aws

Entwicklerhandbuch

Deadline Cloud

Deadline Cloud: Entwicklerhandbuch

Table of Contents

Was ist AWS Deadline Cloud?

AWS Deadline Cloud ist ein vollständig verwalteter AWS Service, mit dem Sie innerhalb von Minuten eine skalierbare Verarbeitungsfarm einrichten und in Betrieb nehmen können. Es bietet eine Verwaltungskonsole für die Verwaltung von Benutzern, Farmen, Warteschlangen für die Planung von Jobs und Flotten von Mitarbeitern, die die Verarbeitung durchführen.

Dieses Entwicklerhandbuch richtet sich an Entwickler von Pipelines, Tools und Anwendungen für eine Vielzahl von Anwendungsfällen, darunter die folgenden:

- Pipeline-Entwickler und technische Direktoren können Deadline Cloud APIs und ihre Funktionen in ihre individuellen Produktionspipelines integrieren.
- Unabhängige Softwareanbieter können Deadline Cloud in ihre Anwendungen integrieren, sodass Künstler und Benutzer, die digitale Inhalte erstellen, Renderaufträge von Deadline Cloud nahtlos von ihren Workstations aus einreichen können.
- Entwickler von Web- und Cloud-basierten Diensten können Deadline Cloud-Rendering in ihre Plattformen integrieren, sodass Kunden Ressourcen bereitstellen können, um Produkte virtuell anzusehen.

Wir bieten Tools, mit denen Sie direkt mit jedem Schritt Ihrer Pipeline arbeiten können:

- Eine Befehlszeilenschnittstelle, die Sie direkt oder über Skripts verwenden können.
- Das AWS SDK für 11 beliebte Programmiersprachen.
- Eine REST-basierte Weboberfläche, die Sie von Ihren Anwendungen aus aufrufen können.

Sie können auch andere AWS-Services in Ihren benutzerdefinierten Anwendungen verwenden. Sie können beispielsweise Folgendes verwenden:

- AWS CloudFormationum das Erstellen und Entfernen von Farmen, Warteschlangen und Flotten zu automatisieren.
- Amazon CloudWatch sammelt Kennzahlen für Jobs.
- Amazon Simple Storage Service zum Speichern und Verwalten digitaler Ressourcen und der Auftragsausgabe.
- AWS IAM Identity Centerum Benutzer und Gruppen für Ihre Farmen zu verwalten.

Stellenbeschreibung öffnen

Deadline Cloud verwendet die Open Job Description (OpenJD) -Spezifikation, um die Details eines Jobs zu spezifizieren. OpenJD wurde entwickelt, um Jobs zu definieren, die zwischen Lösungen portierbar sind. Sie verwenden es, um einen Job zu definieren, der aus einer Reihe von Befehlen besteht, die auf Worker-Hosts ausgeführt werden.

Sie können eine OpenJD-Jobvorlage mithilfe eines von Deadline Cloud bereitgestellten Absenders erstellen, oder Sie können ein beliebiges Tool verwenden, mit dem Sie die Vorlage erstellen möchten. Nachdem Sie die Vorlage erstellt haben, senden Sie sie an Deadline Cloud. Wenn Sie einen Submitter verwenden, kümmert sich dieser um den Versand der Vorlage. Wenn Sie die Vorlage auf andere Weise erstellt haben, rufen Sie eine Deadline Cloud-Befehlszeilenaktion auf, oder Sie können eine der Aktionen verwenden, AWS SDKs um den Job zu senden. In beiden Fällen fügt Deadline Cloud den Job der angegebenen Warteschlange hinzu und plant die Arbeit.

Konzepte und Terminologie für Deadline Cloud

Um Ihnen den Einstieg in AWS Deadline Cloud zu erleichtern, werden in diesem Thema einige der wichtigsten Konzepte und Begrifflichkeiten erläutert.

Ressourcen der Farm

Dieses Diagramm zeigt, wie die Farmressourcen von Deadline Cloud zusammenarbeiten.

Farm

> Eine Farm enthält alle anderen Ressourcen, die sich auf das Senden und Ausführen von Jobs beziehen. Farmen sind voneinander unabhängig, was sie für die Trennung von Produktionsumgebungen nützlich macht.

Warteschlange

> Eine Warteschlange enthält Aufträge für die Planung der zugehörigen Flotten. Benutzer können Aufträge an eine Warteschlange senden und ihre Priorität und ihren Status innerhalb der Warteschlange verwalten. Eine Warteschlange muss einer Flotte mit einer Zuordnung zwischen Warteschlange und Flotte zugeordnet sein, damit ihre Jobs ausgeführt werden können. Warteschlangen können mehreren Flotten zugeordnet werden.

Flotte

Eine Flotte enthält Rechenkapazität für die Ausführung von Jobs. Flotten können vom Service oder vom Kunden verwaltet werden. Serviceverwaltete Flotten werden in Deadline Cloud ausgeführt und verfügen über integrierte Funktionen wie Autoskalierung, Lizenzierung und Softwarezugriff. Von Kunden verwaltete Flotten laufen auf Ihren eigenen Rechenressourcen wie EC2 Amazon-Instances oder lokalen Servern.

Budget

Ein Budget legt Ausgabenschwellen für Ihre Arbeitsaktivität fest und ermöglicht es Ihnen, Maßnahmen zu ergreifen, wenn diese Schwellenwerte erreicht sind, z. B. die Auftragsplanung zu beenden.

Warteschlangenumgebung

Eine Warteschlangenumgebung definiert Skripten, die auf jedem Worker ausgeführt werden, um die Workload-Umgebung einzurichten oder herunterzufahren. Sie sind nützlich, um Umgebungsvariablen festzulegen, Software zu installieren und den Ressourcenspeicher zu konfigurieren.

Speicherprofil

Ein Speicherprofil ist eine Konfiguration für eine Gruppe von Hosts und Workstations, die angibt, wo sich Daten im Dateisystem befinden. Deadline Cloud verwendet Speicherprofile, um Pfade zuzuordnen, wenn Jobs auf unterschiedlich konfigurierten Hosts ausgeführt werden, z. B. ein Job, der von Windows oder auf dem Linux gesendet wurde.

Limit

Ein Limit ermöglicht es Ihnen, die Nutzung gemeinsam genutzter Ressourcen wie Floating-Lizenzen nachzuverfolgen und zu kontrollieren, wie sie den Jobs zugewiesen werden. Limits sind Warteschlangen zugeordnet, denen Warteschlangenlimits zugewiesen sind.

Überwachen

Der Monitor konfiguriert die URL für die Deadline Cloud Monitor-Webanwendung, sodass Endbenutzer Jobs überwachen und verwalten können. Der Zugriff darauf kann in einem Browser oder über die Desktop-Anwendung Deadline Cloud Monitor erfolgen.

Ressourcen zur Auftragsausführung

Dieses Diagramm zeigt, wie die Jobressourcen von Deadline Cloud zusammenarbeiten.

Aufgabe

Ein Job ist eine Reihe von Arbeiten, die ein Benutzer an Deadline Cloud übermittelt, um sie zu planen und für verfügbare Mitarbeiter auszuführen. Ein Job kann eine 3D-Szene rendern oder eine Simulation ausführen. Jobs werden anhand wiederverwendbarer Auftragsvorlagen erstellt, die die Laufzeitumgebung und die Prozesse sowie auftragsspezifische Parameter definieren. Jobs enthalten Schritte und Aufgaben, die die auszuführende Arbeit definieren. Sie können mit Prioritäten, der maximalen Anzahl von Mitarbeitern und Einstellungen für Wiederholungsversuche konfiguriert werden.

Priorität der Job

Die Auftragspriorität ist die ungefähre Reihenfolge, in der Deadline Cloud einen Job in einer Warteschlange verarbeitet. Sie können die Job-Priorität zwischen 1 und 100 festlegen. Jobs mit einer höheren Priorität werden in der Regel zuerst verarbeitet. Jobs mit derselben Priorität werden in der Reihenfolge bearbeitet, in der sie eingegangen sind.

Auftragseigenschaften

Auftragseigenschaften sind Einstellungen, die Sie beim Absenden eines Renderjobs definieren. Einige Beispiele umfassen den Bildbereich, den Ausgabepfad, Auftragsanhänge, renderfähige Kamera und mehr. Die Eigenschaften variieren je nach dem DCC, von dem das Rendering eingereicht wurde.

Schritt

Ein Schritt ist Teil eines Jobs, der eine Vorlage für die Ausführung vieler Aufgaben bereitstellt, die bis auf die Aufgabenparameterwerte identisch sind. Schritte können von anderen Schritten abhängig sein, sodass Sie komplexe Workflows mit sequentiellen oder parallel Ausführungspfaden erstellen können. Bei Renderaufträgen definiert ein Schritt häufig den Befehl zum Rendern eines Frames und verwendet die Frame-Nummer als Aufgabenparameter.

Aufgabe

Eine Aufgabe ist die kleinste Arbeitseinheit in Deadline Cloud. Aufgaben sind Teil von Schritten und werden von Mitarbeitern ausgeführt. Dabei handelt es sich um einzelne Operationen, die im Rahmen eines Jobs ausgeführt werden müssen. Aufgaben können mit bestimmten Parametern konfiguriert werden und den Mitarbeitern auf der Grundlage ihrer Fähigkeiten und Verfügbarkeit zugewiesen. Bei Renderjobs rendert eine Aufgabe häufig einen einzelnen Frame.

Worker

Mitarbeiter sind Teil einer Flotte und führen Aufgaben von Jobs aus aus. Worker können mit bestimmten Funktionen wie GPU-Beschleunigern, CPU-Architektur und Betriebssystem konfiguriert werden. In Flotten mit Servicemanagement werden Mitarbeiter automatisch erstellt, wenn die Flotte nach oben und unten skaliert wird.

Instance

Flotten verwenden Instanzen für CPU-Ressourcen. Eine Instance ist eine EC2 Amazon-Performance-Instance. Deadline Cloud verwendet On-Demand- und Spot-Instances.

On-Demand-Instanz

On-Demand-Instances werden sekundengenau berechnet, haben keine langfristige Bindung und werden nicht unterbrochen.

Spot-Instance

Spot-Instances sind unreservierte Kapazität, die Sie zu einem vergünstigten Preis nutzen können, die jedoch durch On-Demand-Anfragen unterbrochen werden kann.

Warten Sie und speichern Sie

Die Funktion „Warten und Speichern“ ermöglicht eine verzögerte Auftragsplanung zu geringeren Kosten und kann durch On-Demand-Anfragen und Spot-Anfragen unterbrochen werden. Wait and Save ist nur für vom Service verwaltete Flotten von Deadline Cloud verfügbar.

Wait and Save dient zur Verwaltung der Ausführung von Visual-Computing-Workloads in AWS Deadline Cloud. Einzelheiten finden Sie in den AWS Servicebedingungen.

Sitzung

Eine Sitzung stellt die Reihenfolge der Arbeit eines Mitarbeiters an einem Job dar. Während einer einzelnen Sitzung können einem Mitarbeiter mehrere Aufgaben zugewiesen werden, die er nacheinander ausführt. Sitzungen beinhalten häufig Einrichtungsaktionen, mit denen Umgebungen konfiguriert und Ressourcen geladen werden, bevor die Aufgabenaktionen ausgeführt werden.

Aktion der Sitzung

Eine Sitzungsaktion steht für bestimmte Operationen, die während einer Sitzung ausgeführt werden, z. B. das Einrichten der Umgebung, das Ausführen einer Aufgabe und das Synchronisieren von Assets.

Andere wichtige Konzepte und Terminologie

Verwendungsexplorer

Der Usage Explorer ist eine Funktion von Deadline Cloud Monitor. Er bietet eine ungefähre Schätzung Ihrer Kosten und Nutzung.

Budgetmanager

Der Budgetmanager ist Teil des Deadline Cloud-Monitors. Verwenden Sie den Budgetmanager, um Budgets zu erstellen und zu verwalten. Sie können ihn auch verwenden, um Aktivitäten einzuschränken, um das Budget einzuhalten.

Deadline Cloud-Clientbibliothek

Die Open-Source-Clientbibliothek umfasst eine Befehlszeilenschnittstelle und eine Bibliothek zur Verwaltung von Deadline Cloud. Zu den Funktionen gehören das Senden von Jobpaketen auf der Grundlage der Open Job Description-Spezifikation an Deadline Cloud, das Herunterladen von Ausgaben für Jobanhänge und die Überwachung Ihrer Farm über die Befehlszeilenschnittstelle (CLI).

Anwendung zur Erstellung digitaler Inhalte (DCC)

Anwendungen zur Erstellung digitaler Inhalte (DCCs) sind Produkte von Drittanbietern, mit denen Sie digitale Inhalte erstellen. Deadline Cloud verfügt über integrierte Integrationen mit vielen Programmen DCCs wie Autodesk Maya, Blender und Maxon Cinema 4D, sodass Sie Aufträge aus dem DCC heraus einreichen und auf servicemanagierten Flotten mit vorkonfigurierter Software und Lizenzierung rendern können.

Arbeitsanhänge

Bei Auftragsanhängen handelt es sich um eine Deadline Cloud-Funktion, mit der Sie im Rahmen eines Jobs Assets wie Texturen, 3D-Modelle und Lichtanlagen hoch- und herunterladen können. Jobanhänge werden in Amazon S3 gespeichert, sodass kein gemeinsam genutzter Netzwerkspeicher erforderlich ist.

Auftragsvorlage

Eine Jobvorlage definiert die Laufzeitumgebung und alle Prozesse, die als Teil eines Deadline Cloud-Jobs ausgeführt werden.

Deadline Cloud-Einreicher

Ein Deadline Cloud-Einreicher ist ein Plugin für ein DCC, mit dem Benutzer Jobs einfach aus dem DCC heraus einreichen können.

Lizenzendpunkt

Ein Lizenzendpunkt macht die nutzungsbasierte Lizenzierung von Deadline Cloud für Produkte von Drittanbietern in Ihrer VPC verfügbar. Dieses Modell ist nutzungsabhängig und Ihnen wird die Anzahl der Stunden und Minuten in Rechnung gestellt, die Sie nutzen. Lizenzendpunkte sind nicht mit Farmen verbunden und können unabhängig voneinander verwendet werden.

Tags (Markierungen)

Ein Tag ist eine Bezeichnung, die Sie einer AWS Ressource zuweisen können. Jedes Tag besteht aus einem Schlüssel und einem optionalen Wert, den Sie definieren. Mithilfe von Stichwörtern können Sie Ihre AWS Ressourcen auf unterschiedliche Weise kategorisieren, z. B. nach Zweck, Eigentümer oder Umgebung.

Nutzungsbasierte Lizenzierung (UBL)

Die nutzungsbasierte Lizenzierung (UBL) ist ein On-Demand-Lizenzmodell, das für ausgewählte Produkte von Drittanbietern verfügbar ist. Bei diesem Modell handelt es sich um eine nutzungsabhängige Bezahlung, bei der Ihnen die Anzahl der Stunden und Minuten in Rechnung gestellt wird, die Sie nutzen.

Deadline: Beratung zur Cloud-Architektur

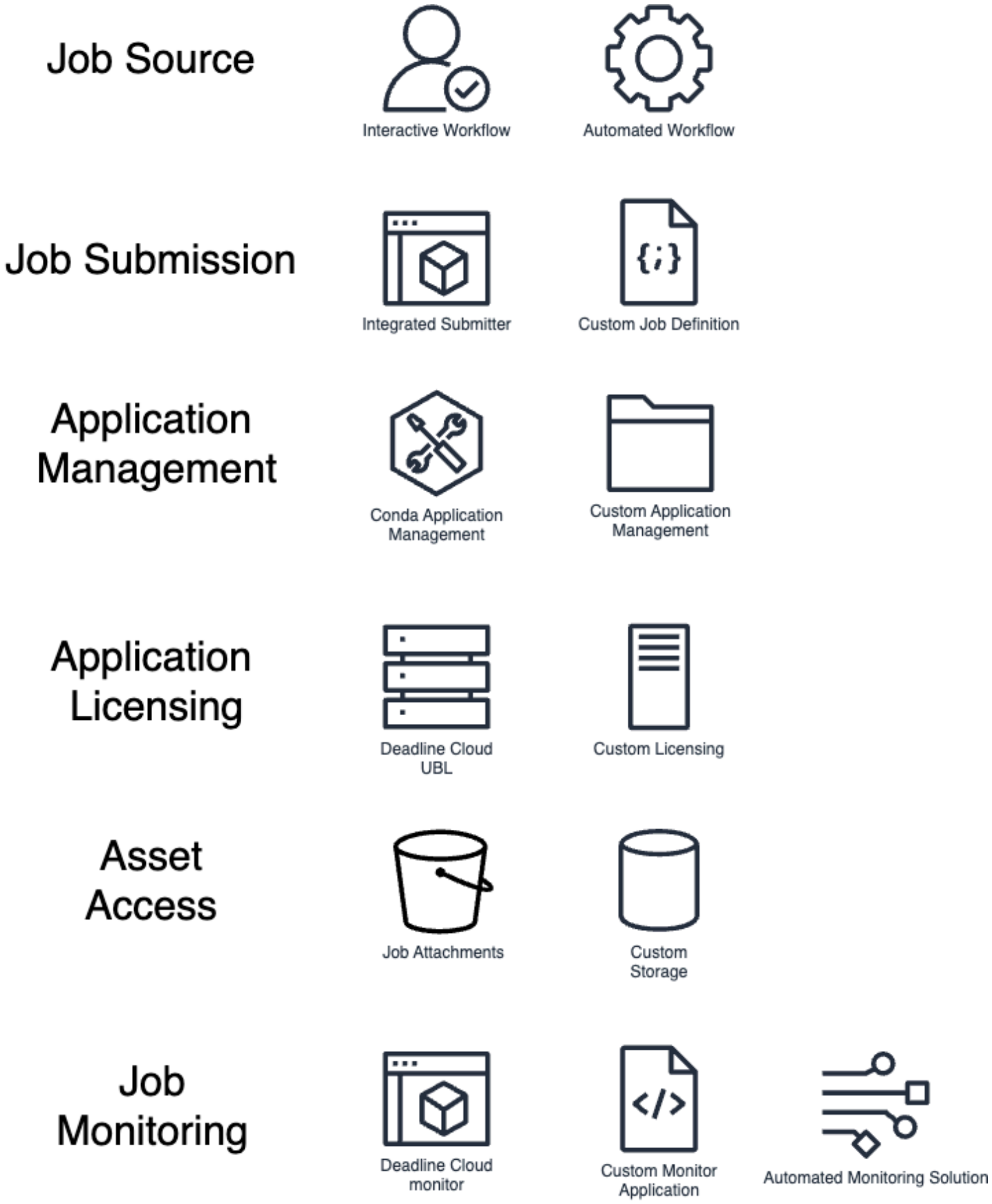
Dieses Thema bietet Anleitungen und bewährte Methoden für den Entwurf und Aufbau zuverlässiger, sicherer, effizienter und kostengünstiger Renderfarmen für Ihre Workloads mithilfe von Deadline Cloud. Diese Hinweise können Ihnen dabei helfen, stabile und effiziente Workloads aufzubauen, sodass Sie sich auf Innovationen konzentrieren, Kosten senken und die Kundenfreundlichkeit verbessern können.

Dieser Inhalt richtet sich an Chief Technology Officers (CTOs), Architekten, Entwickler und Mitglieder des Betriebsteams.

Ein end-to-end Rendering-Workflow erfordert Lösungen auf mehreren Ebenen des Prozesses, z. B. bei der Auftragserstellung, beim Zugriff auf Ressourcen und bei der Auftragsüberwachung. Deadline Cloud bietet mehrere Lösungen für jede Ebene des Renderprozesses. Indem Sie in jeder Ebene aus den Optionen von Deadline Cloud auswählen, können Sie einen Workflow entwerfen, der Ihrem Anwendungsfall entspricht.

Für jede Ebene müssen Sie entscheiden, welcher Ansatz für Ihren Anwendungsfall am besten geeignet ist. Dies sind keine strengen Szenariodefinitionen und nicht die einzige Möglichkeit, Deadline Cloud zu verwenden. Stattdessen handelt es sich um eine Reihe allgemeiner Konzepte, die Ihnen helfen sollen, zu verstehen, wie Deadline Cloud in Ihr Unternehmen oder Ihren Arbeitsablauf passen könnte. Sie können Deadline Cloud-Workloads in die folgenden Ebenen unterteilen: Jobquelle, Job-Einreichung, Anwendungsmanagement, Anwendungslizenzierung, Asset Access, Output Management und Worker Infrastructure Management.

Im Allgemeinen können Sie mix-and-match jedes Szenario auf einer Ebene mit jedem anderen Szenario auf einer anderen Ebene kombinieren, mit Ausnahme bestimmter Kombinationen, die unten angegeben sind.


Job Source
Interactive Workflow
Automated Workflow
Job Submission
Integrated Submitter
Custom Job Definition
Application Management
Conda Application Management
Custom Application Management
Application Licensing
Deadline Cloud UBL
Custom Licensing
Asset Access
Job Attachments
Custom Storage
Job Monitoring
Deadline Cloud monitor
Custom Monitor Application
Automated Monitoring Solution
Worker Infrastructure

Quelle des Job

Die Jobquelle ist der Zugangspunkt, über den neue Jobs in das System gelangen, um von Deadline Cloud gerendert zu werden. Auf hoher Ebene gibt es zwei Hauptquellen für Jobs: menschliche Interaktivität und automatisierte Computersysteme.

Interaktiver Arbeitsablauf

In diesem Szenario ist ein Künstler oder eine andere kreative Rolle der Hauptgenerator für die Arbeit, die in der Deadline Cloud-Farm verarbeitet werden soll. In der Regel sind die Ergebnisse dieser Jobs ein primäres Artefakt für das größere Projekt oder Team. Sie führen ihre Arbeit mit Software wie einem branchenüblichen Tool zur Erstellung digitaler Inhalte (DCC) aus. Sie reichen Jobs manuell an die Deadline Cloud-Farm ein und schauen sich die Ergebnisse anschließend zur Überprüfung an. Die Workstation selbst wird nicht von verwaltet AWS.

In den meisten Fällen verwenden diese Künstler die in Deadline Cloud integrierten Submitter und den Deadline Cloud-Monitor in den Ebenen Workload Application und Monitoring.

Automatisierter Arbeitsablauf

In diesem Szenario ist ein programmgesteuertes System, das dem Kunden gehört, der Hauptgenerator von Aufträgen in der Deadline Cloud-Farm. Dabei kann es sich um die Generierung von Ressourcen in einer Pipeline im Einzelhandel handeln, wie z. B. ein anhand eines 3D-Modells oder eines Scans generiertes Video über einen Plattenspieler. Dies könnte das automatisierte Compositing von Übertragungsgrafiken und Spielerkarten für den Sport sein. Das Thema dieses Szenarios ist, dass eine Person nicht jeden Job manuell an Deadline Cloud übermittelt, sondern dass der Job stattdessen als Teil eines größeren Systems generiert wird.

Bei automatisierten Jobs ist es weniger üblich, dass integrierte Absender von Deadline Cloud und der Deadline Cloud-Monitor verwendet werden. Häufig handelt es sich bei den Jobdefinitionen um von Ihnen verfasste benutzerdefinierte Anwendungsentwicklung, und die Ergebnisse der Jobs werden automatisch zur Genehmigung und Verteilung in ein Digital Asset Management (DAM) -System oder ein Media Asset Management (MAM) -System eingespeist.

Einreichung von Job

Jobs werden mithilfe von OpenJobDescriptionVorlagen an Deadline Cloud übermittelt. OpenJobDescription ist eine flexible, offene Spezifikation zur Definition von Batch-

Verarbeitungsaufträgen, die zwischen verschiedenen Bereitstellungen von Planungssystemen übertragen werden können. Die Jobdefinitionsdatei beschreibt die Parameter des Jobs, die Schritte des Jobs, wie ein Schritt basierend auf den Jobeingaben parametrisiert wird, sowie das eigentliche Skript, das auf einem Worker ausgeführt wird, um die Verarbeitung durchzuführen. Die Idee von Workload Submission besteht darin, wie diese Jobdefinitionen erstellt werden, wer sie erstellt und wie sie eingereicht werden.

Integrierter Einreicher mit DCC

Ein integrierter Deadline Cloud-Einreicher ist eine Software, die Deadline Cloud mit einem branchenüblichen DCC oder Softwarepaket verbindet. Der integrierte Einreicher bestimmt, wie die Daten und die Konfiguration für einen Render-, Composite- oder anderen Workload in eine Jobvorlage umgewandelt werden, was von Deadline Cloud verstanden werden kann. Viele integrierte Einreicher werden vom Deadline Cloud-Team oder dem Entwickler des Softwarepakets erstellt und verwaltet. Falls es für die gewünschte Anwendung jedoch noch keinen gibt, können Sie Ihren eigenen Submitter erstellen und verwalten. Es gibt eine begrenzte Anzahl davon DCCs , die vom Deadline Cloud-Team unterstützt werden.

Interaktive Workflows beinhalten normalerweise integrierte Einreicher, aber nicht immer. Bei automatisierten Workflows mit Vorlagen besteht ein üblicher Arbeitsablauf darin, dass ein Künstler einen Vorlagen-Job in seinem DCC einrichtet und das Auftragspaket einmalig exportiert. Dieses Job-Bundle definiert, wie diese bestimmte Art von Job in Deadline Cloud parametrisiert ausgeführt wird. Dieses Job-Bundle kann zu Automatisierungszwecken in das automatisierte Workflow-Szenario integriert werden.

Benutzerdefinierte Jobdefinition

Bei benutzerdefinierten Anwendungen und Workflows ist es möglich, vollständig zu kontrollieren, wie diese Jobdefinitionen erstellt und an Deadline Cloud übermittelt werden. Auf einer E-Commerce-Website könnten Verkäufer beispielsweise aufgefordert werden, 3D-Modelle des Objekts hochzuladen, das sie verkaufen. Nach diesem Upload könnte die E-Commerce-Plattform dynamisch eine Auftragsdefinition generieren, die an Deadline Cloud gesendet wird, um automatisch eine Drehscheibenanimation auf einem gemeinsamen Hintergrund zu generieren, wobei die gleiche Beleuchtung verwendet wird, die zu den anderen auf der Website verfügbaren 3D-Objekten passt. Während der Entwicklung der E-Commerce-Plattform erstellte ein Softwareentwickler eine Jobdefinition, bettete sie mit eventuell von den Verkäufern bereitgestellten Parametern in die E-Commerce-Plattform ein und codierte die Plattform, um diesen Job während des Produkt-Upload-Workflows der Plattform einzureichen.

Deadline Cloud bietet eine Reihe von Beispiel-Jobdefinitionen im Beispiel-Repository auf Github.

Application Management

Nachdem ein Job an Deadline Cloud übermittelt und einem Worker zugewiesen wurde, wird das Skript aus der Jobdefinition auf dem Worker ausgeführt. In den meisten Fällen ruft dieses Skript eine Anwendung auf, die die eigentliche Verarbeitung durchführt, z. B. Renderer, Composite, Kodierung, Filterung oder andere rechenintensive Aufgaben. Anwendungsmanagement ist das Konzept, mit dem sichergestellt wird, dass den Mitarbeitern die erforderliche Version der erforderlichen Software zur Verfügung steht.

Sie können Anwendungen mit jedem beliebigen Paketverwaltungssystem verwalten, aber Deadline Cloud bietet eine Reihe von Tools, um die Verwendung von Conda-Paketen einfach zu ermöglichen. Conda ist ein plattformübergreifendes, sprachunabhängiges Open-Source-Paketmanager- und Umgebungsmanagementsystem.

Deadline Cloud-verwalteter Conda-Kanal für servicemanagierte Flotten (SMF)

Wenn Sie serviceverwaltete Flotten verwenden, wird automatisch ein von Deadline Cloud verwalteter Conda-Kanal eingerichtet und für die Nutzung durch Ihre Jobs konfiguriert. Der Deadline Cloud-Dienst stellt eine Reihe von DCC-Anwendungen und Renderings von Partnern in diesem Conda-Kanal bereit. Weitere Informationen finden Sie unter Erstellen einer Warteschlangenumgebung im Deadline Cloud-Benutzerhandbuch. Diese Pakete werden vom Deadline Cloud-Dienst automatisch auf dem neuesten Stand gehalten und erfordern keine Wartung durch Sie. Dieser Conda-Kanal ist nur verfügbar, wenn Sie vom Service verwaltete Flotten verwenden. Er ist nicht verfügbar, wenn Sie vom Kunden verwaltete Flotten verwenden.

Selbstverwalteter Conda-Kanal

Wenn Sie den von Deadline Cloud verwalteten Conda-Channel nicht verwenden können, müssen Sie festlegen, wie Sie Anwendungen auf Ihrer Deadline Cloud-Flotte installieren, patchen und anderweitig verwalten. Eine Möglichkeit besteht darin, einen Conda-Channel zu erstellen, den Sie einrichten und verwalten. Dies wird am engsten mit dem von Deadline Cloud verwalteten Conda-Kanal zusammenarbeiten. Sie können beispielsweise ein DCC aus dem von Deadline Cloud verwalteten Conda-Kanal verwenden, aber Ihr eigenes Paket mitbringen, das ein bestimmtes DCC-Plugin enthält. Weitere Informationen zu diesem Prozess finden Sie unter Erstellen eines Conda-Kanals mit S3.

Benutzerdefiniertes Anwendungsmanagement

Für die Anwendungsverwaltung besteht die Anforderung von Deadline Cloud darin, dass die Anwendung im PATH verfügbar ist, wenn das Job-Skript auf dem Worker ausgeführt wird.

Wenn Sie bereits Rez-Pakete erstellen und verwalten, können Sie eine Warteschlangenumgebung verwenden, um die Anwendungen aus Rez-Repositorys zu installieren. Ein Beispiel für eine Warteschlangenumgebung finden Sie auf der AWS Deadline GitHub Cloud-Organisation.

Wenn Sie bereits Anwendungen in einer vom Kunden verwalteten Flotte mit langlebigen Mitarbeitern oder in System-Images verwalten, ist für die Anwendungsverwaltung keine Warteschlangenumgebung erforderlich. Stellen Sie sicher, dass die Bewerbung im Pfad des Auftragsbenutzers angezeigt wird, und senden Sie den Job ab.

Application licensing

Für viele Workloads, die üblicherweise auf Deadline Cloud ausgeführt werden, ist eine Softwarelizenzierung durch den Softwareanbieter erforderlich. Diese Anwendungen werden häufig pro Arbeitsplatz, pro CPU oder pro Host lizenziert. Es liegt in Ihrer Verantwortung, sicherzustellen, dass Ihre Nutzung von Drittanbieter-Software auf Deadline Cloud der Lizenzvereinbarung des Drittanbieters entspricht. Wenn Sie Open-Source-Software, benutzerdefinierte Software oder anderweitig lizenzfreie Software verwenden, ist die Konfiguration dieser Ebene nicht erforderlich. Beachten Sie, dass Deadline Cloud nur Render-Lizenzen und keine Workstation-Lizenzierung unterstützt.

Serviceverwaltete Flotten und nutzungsabhängige Lizenzierung

Bei der Verwendung von serviceverwalteten Flotten von Deadline Cloud wird die nutzungsbasierte Lizenzierung (UBL) automatisch für unterstützte Software konfiguriert. Bei Aufträgen, die auf vom Service verwalteten Flotten ausgeführt werden, werden automatisch Umgebungsvariablen für unterstützte Anwendungen festgelegt, die sie zur Nutzung der Deadline Cloud-Lizenzserver anweisen. Wenn Sie Deadline Cloud UBL verwenden, wird Ihnen nur die Anzahl der Stunden in Rechnung gestellt, für die Sie die lizenzierte Anwendung verwenden.

Kundenverwaltete Flotten und nutzungsabhängige Lizenzierung

Die nutzungsbasierte Cloud-Lizenzierung (UBL) von Deadline ist auch verfügbar, wenn keine vom Service verwalteten Flotten verwendet werden. In diesem Szenario richten Sie Deadline Cloud-Lizenzendpunkte ein, die IP-Adressen in Ihren ausgewählten VPC-Subnetzen

bereitstellen, die Zugriff auf Deadline Cloud-Lizenzserver ermöglichen. Nachdem Sie die entsprechenden softwarespezifischen Umgebungsvariablen auf Ihren Workern konfiguriert und die Netzwerkkonnektivität zwischen den Workern und diesen Lizenzendpunkt-IP-Adressen konfiguriert haben, können die Mitarbeiter Lizenzen für unterstützte Software auschecken und einchecken. Für Lizenzen wird Ihnen genau wie bei der Nutzung von UBL in serviceverwalteten Flotten pro Stunde abgerechnet.

Benutzerdefinierte Lizenzierung

Möglicherweise verwenden Sie eine Anwendung, die von Deadline Cloud UBL nicht unterstützt wird, oder Sie verfügen möglicherweise über bereits bestehende Lizenzen, die noch gültig sind. In diesem Szenario sind Sie für die Konfiguration des Netzwerkpfads von Ihren Mitarbeitern (vom Kunden oder vom Service verwaltet) zu den Lizenzservern verantwortlich. Weitere Informationen zur benutzerdefinierten Lizenzierung finden Sie unterConnect vom Service verwaltete Flotten mit einem benutzerdefinierten Lizenzserver.

Zugriff auf Ressourcen

Nachdem ein Job an einen Worker übermittelt und die Anwendung konfiguriert wurde, muss der Worker so konfiguriert werden, dass er auf die für den Job erforderlichen Asset-Daten zugreift. Dies können 3D-Daten, Texturdaten, Animationsdaten, Videoframes oder jede andere Art von Daten sein, die in Ihrem Job verwendet werden.

Denken Sie zunächst darüber nach, wo Ihre Daten derzeit gespeichert sind. Dies kann auf der Festplatte der Workstation, einem Tool für die Zusammenarbeit zwischen Benutzern, der Quellcodeverwaltung, einem gemeinsam genutzten Dateisystem vor Ort oder in der Cloud, Amazon S3 oder einer beliebigen Anzahl anderer Standorte sein.

Überlegen Sie sich als Nächstes, was ein Mitarbeiter für den Zugriff auf diese Daten benötigt. Werden diese Daten nur in Ihrem Unternehmensnetzwerk zur Verfügung gestellt? Welche Identität oder Anmeldeinformationen sind für den Zugriff auf die Daten erforderlich? Ist die Datenquelle so skaliert, dass sie den Job mit der Anzahl der Mitarbeiter unterstützt, die Sie voraussichtlich bearbeiten werden?

Arbeitsanhänge

Der am einfachsten zu verwendende Mechanismus für den Zugriff auf Ressourcen ist Deadline Cloud Job Attachments. Wenn ein Job mithilfe von Job-Anhängen eingereicht wird, werden die für den Job erforderlichen Daten zusammen mit einer Manifestdatei, die angibt, welche Dateien für

den Job benötigt werden, in einen Amazon S3 S3-Bucket hochgeladen. Bei Anhängen von Jobs ist keine komplizierte Einrichtung von Netzwerken oder gemeinsam genutztem Speicher erforderlich. Dateien werden nur einmal hochgeladen, sodass nachfolgende Uploads schneller abgeschlossen werden können. Nachdem ein Mitarbeiter die Bearbeitung eines Jobs abgeschlossen hat, werden die Ausgabedaten auf Amazon S3 hochgeladen, sodass sie vom Künstler oder einem anderen Kunden heruntergeladen werden können. Arbeitsanhänge sind für Flotten jeder Größe skalierbar und lassen sich einfach und schnell einbauen und verwenden.

Arbeitsanhänge sind nicht für alle Situationen das beste Werkzeug. Wenn Ihre Daten bereits aktiviert sind AWS, fügen Sie mit den Stellenanhängen eine zusätzliche Kopie Ihrer Daten hinzu, einschließlich der damit verbundenen Übertragungszeit und der Speicherkosten. Bei Anhängen von Stellenanhängen muss der Job die benötigten Daten zum Zeitpunkt der Einreichung vollständig angeben können, damit die Daten hochgeladen werden können.

Um Job-Anhänge verwenden zu können, muss Ihrer Deadline Cloud-Warteschlange ein Bucket für Jobanhänge zugeordnet sein, und die Warteschlangenrolle muss verwendet werden, um Zugriff auf diesen Bucket zu gewähren. Standardmäßig unterstützen alle in Deadline Cloud integrierten Einreicher Stellenanhänge. Wenn Sie keinen integrierten Submitter von Deadline Cloud verwenden, können Jobanhänge mit Ihrer benutzerdefinierten Software verwendet werden, indem Sie die Deadline Cloud-Python-Bibliothek integrieren.

Benutzerdefinierter Speicherzugriff

Wenn Sie keine Jobanhänge verwenden, sind Sie dafür verantwortlich, dass die Mitarbeiter Zugriff auf die für Jobs erforderlichen Daten haben. Deadline Cloud bietet eine Reihe von Tools, um dies zu unterstützen und Jobs portabel zu halten. Möglicherweise möchten Sie eine benutzerdefinierte Speicherlösung verwenden, wenn Sie bereits über gemeinsamen Netzwerkspeicher für Künstler und Mitarbeiter verfügen, Sie einen externen Dienst bevorzugen LucidLink, oder aus anderen Gründen.

Verwenden Sie Speicherprofile, um Dateisysteme auf Ihrer Workstation und Ihren Worker-Hosts zu modellieren. Jedes Speicherprofil beschreibt das Betriebssystem und das Dateisystem-Layout einer Ihrer Systemkonfigurationen. Wenn ein Künstler, der eine Windows-Workstation verwendet, einen Job einreicht, der von einem Linux Mitarbeiter bearbeitet wird, stellt Deadline Cloud mithilfe von Speicherprofilen sicher, dass die Pfadzuweisung erfolgt, sodass der Mitarbeiter auf den von Ihnen konfigurierten Datenspeicher zugreifen kann.

Wenn Sie vom Service verwaltete Flotten von Deadline Cloud verwenden, ermöglichen Hostkonfigurationsskripten und VPC-Ressourcenendpunkte es Mitarbeitern, gemeinsam genutzten Speicher oder andere in Ihrer VPC verfügbare Dienste direkt bereitzustellen und darauf zuzugreifen.

Auftragsüberwachung und Outputmanagement

Nachdem die an Deadline Cloud eingereichten Jobs erfolgreich abgeschlossen wurden, lädt eine Person oder ein Prozess die Jobausgabe herunter, um sie im Geschäftsworkflow außerhalb von Deadline Cloud zu verwenden. Nach einem Fehlschlag helfen Jobprotokolle und Überwachungsinformationen bei der Diagnose von Problemen.

Frist: Cloud-Monitor

Die Deadline Cloud Monitor-Anwendung ist im Internet und für den Desktop verfügbar. Diese Lösung eignet sich am besten für Studios, die interaktive Workflows für eine Vielzahl von Speicheranwendungen DCCs verwenden. Der Monitor unterstützt Sie nur, wenn Sie IAM Identity Center verwenden. IAM Identity Center ist ein Workforce Identity-Produkt und keine Lösung für Verbraucheridentität (B2C). Daher ist es für viele B2C-Szenarien nicht geeignet.

Benutzerdefinierte Monitor-Anwendung

Wenn Sie das Überwachungserlebnis Ihrer Benutzer anpassen möchten, Sie ein B2C-Produkt entwickeln oder ein hochspezialisiertes System mit Deadline Cloud aufbauen, entscheiden Sie sich für die Erstellung einer benutzerdefinierten Überwachungsanwendung. Sie können die AWS Deadline Cloud-API verwenden, um diese benutzerdefinierte Anwendung zu erstellen und dabei den Kontext Ihres gesamten Workflows mit den Konzepten von Deadline Cloud zu kombinieren. Beispielsweise könnte Ihr B2C-Produkt ein eigenes Projektkonzept haben, das Benutzer einrichten und Ihre Anwendung Deadline Cloud-Jobs in derselben Oberfläche verschachteln kann.

Automatisierte Überwachungslösung

In einigen Szenarien ist für Deadline Cloud keine spezielle Überwachungsanwendung erforderlich. Dieses Szenario ist in automatisierten Workflows üblich, in denen Deadline Cloud zum automatischen Rendern von Inhalten in einer Pipeline verwendet wird, z. B. Grafiken für Sport- oder Nachrichtensendungen. In diesem Szenario werden die Deadline Cloud-API und die EventBridge Ereignisse zur Integration in ein externes Media Asset Management-System verwendet, um Genehmigungen zu erteilen und Daten in den nächsten Prozessschritt zu übertragen.

Verwaltung der Arbeiter-Infrastruktur

Deadline Cloud-Flotten sind eine Gruppe von Servern (Workern), die in der Lage sind, Jobs zu verarbeiten, die an eine Deadline Cloud-Warteschlange gestellt wurden. Sie bilden die Kerninfrastruktur jeder Deadline Cloud-Farm.

Vom Service verwaltete Flotten

In einer servicemanagierten Flotte übernimmt Deadline Cloud die Verantwortung für die Worker-Hosts, das Betriebssystem, das Netzwerk, das Patchen, die automatische Skalierung und andere Faktoren beim Betrieb einer Renderfarm. Sie geben die Mindest- und Höchstzahl der gewünschten Worker sowie die für Ihre Anwendung erforderlichen Systemspezifikationen an, und Deadline Cloud erledigt den Rest. Servicemanagierte Flotten sind die einzige Flottenoption, die von Deadline Cloud verwaltete Conda-Kanäle nutzen kann, um DCC-Anwendungen der Branche einfach zu verwalten. Darüber hinaus wird Deadline Cloud UBL automatisch mit serviceverwalteten Flotten konfiguriert. Wait and Save-Flotten für kostengünstigere, verzögerungstolerante Workloads ist nur für servicemanagierte Flotten verfügbar.

Vom Kunden verwaltete Flotten

Sie verwenden vom Kunden verwaltete Flotten, wenn Sie mehr Kontrolle über die Worker-Hosts und deren Umgebung benötigen. Kundenverwaltete Flotten eignen sich am besten, wenn Sie Deadline Cloud vor Ort verwenden. Weitere Informationen hierzu finden Sie unter Kundenverwaltete Flotten von Deadline Cloud erstellen und verwenden.

Beispielarchitekturen

Traditionelles Produktionsstudio

Das traditionelle Produktionsstudio benötigt eine umfangreiche Rechen-, Speicher- und Netzwerkinfrastruktur, die sich über mehrere physische Standorte erstrecken kann, um Rendering-Workloads zu bewältigen. Jedes einzelne Softwarepaket und jeder Anbieter hat eigene Hardware-, Software-, Netzwerk- und Lizenzanforderungen, die bei der Lösung von Versionierungs-, Kompatibilitäts- und Ressourcenkonflikten erfüllt werden müssen.

Es ist üblich, separate Infrastrukturanforderungen für Künstler-Workstations, Renderknoten, Netzwerkspeicher, Lizenzserver, Job-Queuing-Systeme, Überwachungstools und Asset Management

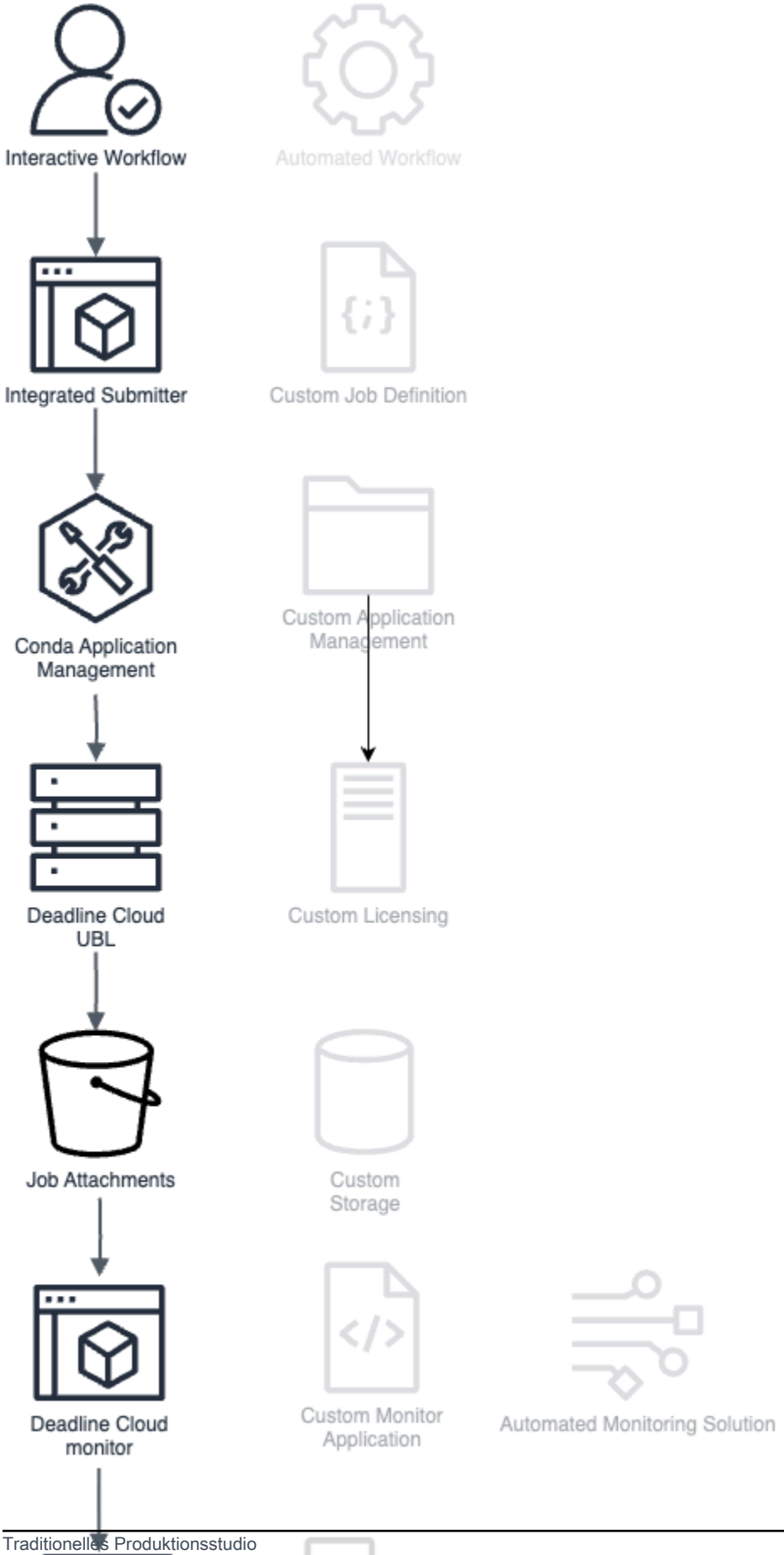
zu haben. Studios müssen in der Regel mehrere Versionen von DCC-Tools, Renderern, Plug-ins und benutzerdefinierten Tools verwalten und gleichzeitig komplexe Lizenzvereinbarungen in ihrer gesamten Renderfarm verwalten. Ihre Studio-Infrastruktur wird komplizierter, wenn Sie Entwicklungs-, Qualitätssicherungs- und Produktionsumgebungen berücksichtigen.

Eine typische Deadline Cloud-Bereitstellung mit service-verwalteten Optionen löst oder reduziert viele dieser Herausforderungen durch:

- Interaktiver Workflow für die Einreichung von Jobs über integrierte DCC-Einreicher
- Bewerbungsmanagement über von Deadline Cloud verwaltete Conda-Kanäle
- Nutzungsbasierte Lizenzierung, die automatisch für unterstützte Software konfiguriert wird
- Verwaltung von Vermögenswerten mithilfe von Stellenanhängen
- Überwachung über die Deadline Cloud Monitor-Anwendung
- Infrastrukturmanagement durch servicemanagierte Flotten

Mit diesem Ansatz können Künstler Jobs direkt von ihren vertrauten DCC-Tools aus an eine skalierbare Cloud-Renderfarm senden, ohne eine komplexe Infrastruktur verwalten zu müssen. Der Service kümmert sich automatisch um die Softwarebereitstellung, Lizenzierung, Datenübertragung und Skalierung der Infrastruktur. Künstler können ihre Jobs über eine Weboberfläche oder eine Desktop-Anwendung überwachen, und die Ergebnisse werden automatisch in Amazon S3 gespeichert, sodass sie leicht darauf zugreifen können.

Mit dieser Konfiguration können Studios Entwicklungs- und Produktionsumgebungen innerhalb von Minuten erstellen, nur für die tatsächlich genutzte Rechenleistung und Lizenzierung zahlen und sich auf kreative Arbeit statt auf Infrastrukturmanagement konzentrieren. Der serviceverwaltete Ansatz bietet den schnellsten Weg zur Einführung von Cloud-Rendering bei gleichzeitiger Beibehaltung vertrauter Arbeitsabläufe für Künstler.


Interactive Workflow
Automated Workflow
Integrated Submitter
Custom Job Definition
Conda Application Management
Custom Application Management
Deadline Cloud UBL
Custom Licensing
Job Attachments
Custom Storage
Deadline Cloud monitor
Custom Monitor Application
Automated Monitoring Solution

Studio in der Cloud

Moderne Studios für visuelle Effekte und Animationen verlagern zunehmend ihre gesamte Pipeline in die Cloud, einschließlich der Workstations für Künstler. Dieser Ansatz macht eine lokale Infrastruktur überflüssig, ermöglicht eine globale Zusammenarbeit und bietet eine nahtlose Skalierung sowohl für interaktives Arbeiten als auch für Rendering. Es bringt jedoch auch neue Herausforderungen bei der Verwaltung von Cloud-Ressourcen, der Sicherstellung des Zugriffs auf Daten mit geringer Latenz und der Integration von Cloud-basierten Workstations in Renderfarmen mit sich.

Ein typisches Cloud-natives Studio erfordert einen einheitlichen Ansatz für die Verwaltung von Cloud-Workstations, gemeinsam genutztem Speicher, der Rendering-Infrastruktur und der Softwarebereitstellung für all diese Komponenten. Herkömmliche Ansätze führten häufig zu komplexen, manuell verwalteten Systemen, die Schwierigkeiten hatten, ein Gleichgewicht zwischen Leistung, Kosten und Flexibilität zu finden.

Eine Deadline Cloud-Bereitstellung für ein Cloud-natives Studio kann wie folgt implementiert werden:

- Interaktive Workflow-Auftragsübermittlung über integrierte DCC-Einreicher auf Cloud-Workstations
- Anwendungsmanagement über von Deadline Cloud verwaltete Conda-Kanäle und Renderknoten
- Nutzungsbasierte Lizenzierung, die automatisch für unterstützte Software konfiguriert wird
- Benutzerdefinierter Speicherzugriff FSx für Windows Dateiserver für gemeinsam genutzte Projektdaten
- Überwachung über die Deadline Cloud-Monitor-Anwendung
- Infrastrukturmanagement mithilfe von serviceverwalteten Flotten

Dieser Ansatz ermöglicht es Künstlern, auf Cloud-basierten Workstations mit direktem Zugriff auf gemeinsam genutzten Hochleistungsspeicher zu arbeiten und Jobs nahtlos an die Deadline Cloud-Farm zu senden. Das Studio kann die Softwarebereitstellung sowohl auf Workstations als auch auf Renderknoten mithilfe derselben Conda-Kanäle verwalten, wodurch Konsistenz gewährleistet und der Wartungsaufwand reduziert wird.

Zu den wichtigsten Vorteilen dieser Konfiguration gehören:

- Weltweite Zusammenarbeit mit Künstlern, die von überall auf Workstations zugreifen können
- Konsistente Softwareumgebungen auf allen Workstations und Renderknoten
- Gemeinsam genutzter Hochleistungsspeicher, auf den sowohl Workstations als auch Renderknoten zugreifen können

- Flexible Skalierung sowohl interaktiver als auch Batch-Rechenressourcen
- Zentralisierte Verwaltung der gesamten Studio-Infrastruktur in der Cloud

Die Speicherkonfiguration in diesem Szenario umfasst in der Regel:

- FSx für Windows Dateiserver für Projektdaten, auf den sowohl Cloud-Workstations als auch Deadline Cloud-Worker zugreifen können
- Speicherprofile in Deadline Cloud zur Verwaltung der Pfadzuweisung zwischen Workstations und Renderknoten
- Direktes Mounten von FSx Shares auf Deadline Cloud-Workern mithilfe von VPC-Ressourcenendpunkten und Hostkonfigurationsskripten

Dieser Cloud-native Ansatz ermöglicht es Studios, auf lokale Infrastrukturen zu verzichten, was eine schnelle Skalierung für Projekte jeder Größe ermöglicht und gleichzeitig die gewohnten Arbeitsabläufe der Künstler beibehalten. Es bietet die Flexibilität, eine Mischung aus vom Service verwalteten und vom Kunden verwalteten Ressourcen zu nutzen, wodurch sowohl die einfache Verwaltung als auch spezifische Leistungsanforderungen optimiert werden.

Durch die Nutzung von Cloud-Workstations zusammen mit Deadline Cloud können Studios eine vollständig integrierte, weltweit zugängliche Produktionspipeline einrichten, die sich nahtlos von kleinen Teams bis hin zu großen Produktionen skalieren lässt.

ECommerce -Automatisierung

Die moderne E-Commerce-Plattform erfordert eine automatisierte Asset-Generierung in großem Maßstab, um eine umfassende Produktvisualisierung für Millionen von Artikeln zu ermöglichen. Herkömmliche Ansätze würden erhebliche Investitionen in die Infrastruktur erfordern, um große Mengen von 3D-Modellen in standardisierte Produktmedien zu verarbeiten, was häufig entweder zu wenig ausgestattete Systeme, die zu Verarbeitungsrückständen führen, oder zu viel bereitgestellten Systemen mit ungenutzter Kapazität führen würde.

Ein typischer automatisierter E-Commerce-Workflow muss die Verarbeitung von Produktuploads, die Validierung von 3D-Modellen, die Verwaltung der Renderfarm, die Ausgabeverarbeitung und die Integration mit Produktinformationssystemen übernehmen. Die Verwaltung dieser Workflows erfordert traditionell die Koordination mehrerer Rendering-Anwendungen, Rechenressourcen und Datenverarbeitungspipelines bei gleichzeitiger Sicherstellung einer gleichbleibenden Qualität und Beibehaltung der Kosteneffizienz im großen Maßstab.

Eine Deadline Cloud-Bereitstellung für die E-Commerce-Automatisierung kann wie folgt implementiert werden:

- Automatisierte Workflow-Auftragsübermittlung durch benutzerdefinierte API-Integration in die bestehende E-Commerce-Erfassungsanwendung
- Maßgeschneiderte Jobdefinitionen, die auf die standardisierte Produktvisualisierung zugeschnitten sind
- Anwendungsmanagement über von Deadline Cloud verwaltete Conda-Kanäle
- Nutzungsbasierte Lizenzierung, die automatisch für unterstützte Software konfiguriert wird
- Direkte Amazon S3 S3-Integration für die Vermögensverwaltung
- Kundenspezifische Überwachungsanwendung, die in bestehende Produktmanagementsysteme integriert ist
- Serviceverwaltete Flotten für elastische Skalierung

Dieser Ansatz ermöglicht die Verarbeitung von Tausenden von Produkten pro Tag und generiert automatisch standardisierte Produktvisualisierungen wie Turntable-Animationen. Die vom Service verwaltete Infrastruktur wird automatisch skaliert, um wechselnden Anforderungen gerecht zu werden und gleichzeitig die Kosteneffizienz durch die Wiederverwendung von Mitarbeitern und die optimierte Anwendungsbereitstellung aufrechtzuerhalten.


eCommerce
Interactive Workflow
Automated Workflow
Integrated Submitter
Custom Job Definition
Conda Application Management
Custom Application Management
Deadline Cloud UBL
Custom Licensing
Job Attachments
Custom Storage
Deadline Cloud monitor
Custom Monitor Application
Automated Monitoring Solution

Whitelabel/OEM/B2C Kunde

Bei herkömmlicher Software zur Erstellung digitaler Inhalte (DCC) müssen Benutzer in der Regel ihre eigene Rendering-Infrastruktur verwalten oder Rendervorgänge lokal auf ihrer Workstation verarbeiten, was entweder zu erheblichen Hardwareinvestitionen oder zu langen Wartezeiten führt, die kreative Workflows unterbrechen. Für Softwareanbieter erforderte die Bereitstellung von Cloud-Rendering-Funktionen traditionell den Aufbau und die Wartung komplexer Infrastruktur- und Abrechnungssysteme.

Eine in die B2C-Software integrierte Deadline Cloud-Bereitstellung ermöglicht ein nahtloses Cloud-Rendering direkt in der vertrauten Benutzeroberfläche des Benutzers. Diese Integration kombiniert:

- Interaktiver Workflow zur Einreichung von Jobs, eingebettet in die DCC-Anwendung
- Deadline Cloud-verwaltete Conda-Kanäle für die Bereitstellung von Render-Anwendungen
- Automatisch konfigurierte nutzungsbasierte Lizenzierung
- Bestandsverwaltung durch Auftragsanhänge mit herstellerverwaltetem Speicher
- Maßgeschneiderte Überwachung, die direkt in die DCC-Schnittstelle integriert ist
- Serviceverwaltete Flotten, die von allen Benutzern gemeinsam genutzt werden

Dieser Ansatz ermöglicht es Endbenutzern, Renderings mit einem einzigen Klick von ihrer Software aus an die Cloud zu senden, ohne Konten, Infrastruktur oder komplexe Einstellungen verwalten zu müssen. Der Softwareanbieter unterhält eine Umgebung mit mehreren Mandanten, in der:

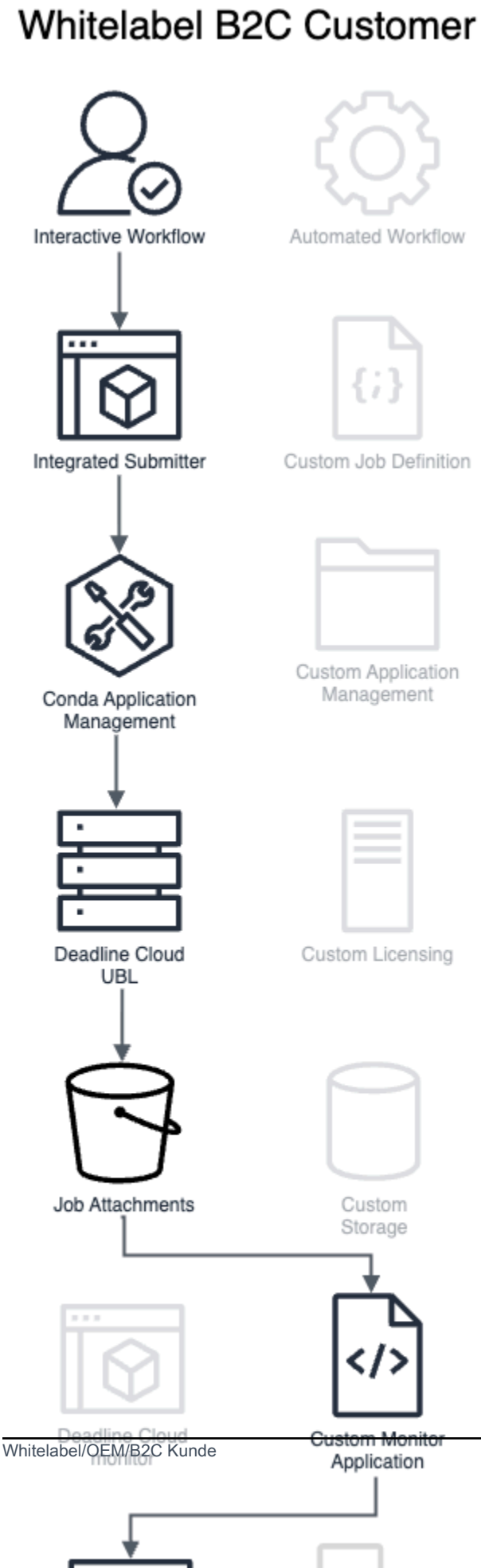
- Benutzer authentifizieren sich mit ihren vorhandenen Software-Anmeldeinformationen
- Jobs werden automatisch an spezielle Warteschlangen für jeden Benutzer weitergeleitet
- Ressourcen werden mithilfe von IAM-gesteuerten Speicherpräfixen sicher isoliert
- Die Abrechnung erfolgt über die vorhandenen Systeme des Anbieters
- Auftragsstatus und Ausgaben werden direkt zurück in die Anwendung des Benutzers gestreamt

Der Ansatz der gemeinsamen Flotte gewährleistet eine optimale Leistung, indem ein warmer Pool von Mitarbeitern aufrechterhalten wird, die Startzeiten minimiert und gleichzeitig die Ressourcennutzung für die gesamte Benutzerbasis maximiert wird. Diese Konfiguration ermöglicht es Softwareanbietern, Cloud-Rendering als nahtlose Produktfunktion anzubieten und nicht als separaten Service, für den zusätzliche Einstellungen oder Konten erforderlich sind.

Endbenutzer profitieren von:

- Einreichung mit einem Klick über ihre vertraute Oberfläche
- Pay-as-you-go Preisgestaltung ohne Infrastrukturmanagement
- Schnelle Startzeiten von Jobs durch gemeinsame Infrastruktur
- Automatischer Download und Organisation abgeschlossener Renderings
- Konsistentes Erlebnis auf allen Plattformen

Dieses Integrationsmuster ermöglicht es Softwareanbietern, ihrer gesamten Benutzerbasis Rendering-Funktionen auf Unternehmensebene zur Verfügung zu stellen und gleichzeitig ein einfaches, benutzerfreundliches Erlebnis zu gewährleisten, das sich wie von selbst in ihrer Anwendung anfühlt.


Whitelabel B2C Customer
Interactive Workflow
Automated Workflow
Integrated Submitter
Custom Job Definition
Conda Application Management
Custom Application Management
Deadline Cloud UBL
Custom Licensing
Job Attachments
Custom Storage
Deadline Cloud monitor
Custom Monitor Application
Automated Monitoring Solution

Was ist ein Deadline Cloud-Workload

Mit AWS Deadline Cloud können Sie Jobs einreichen, um Ihre Anwendungen in der Cloud auszuführen und Daten für die Produktion von Inhalten oder Erkenntnissen zu verarbeiten, die für Ihr Unternehmen von entscheidender Bedeutung sind. Deadline Cloud verwendet Open Job Description (OpenJD) als Syntax für Jobvorlagen, eine Spezifikation, die auf die Bedürfnisse von Visual Compute-Pipelines zugeschnitten ist, aber auch auf viele andere Anwendungsfälle anwendbar ist. Einige Beispiele für Workloads umfassen Computergrafik-Rendering, Physiksimulation und Photogrammetrie.

Die Workloads reichen von einfachen Job-Bundles, die Benutzer entweder mit der CLI oder einer automatisch generierten GUI an eine Warteschlange senden, bis hin zu integrierten Submitter-Plug-ins, die dynamisch ein Job-Bundle für einen anwendungsdefinierten Workload generieren.

Wie Workloads bei der Produktion entstehen

Um Workloads in Produktionskontexten zu verstehen und zu verstehen, wie sie mit Deadline Cloud unterstützt werden können, sollten Sie sich überlegen, wie sie entstehen. Die Produktion kann die Erstellung von visuellen Effekten, Animationen, Spielen, Produktkatalogbildern, 3D-Rekonstruktionen für Building Information Modeling (BIM) und mehr beinhalten. Diese Inhalte werden in der Regel von einem Team von künstlerischen oder technischen Spezialisten erstellt, die eine Vielzahl von Softwareanwendungen und kundenspezifischen Skripten ausführen. Die Mitglieder des Teams tauschen Daten mithilfe einer Produktionspipeline untereinander aus. Viele Aufgaben, die von der Pipeline ausgeführt werden, erfordern intensive Berechnungen, die Tage dauern würden, wenn sie auf der Workstation eines Benutzers ausgeführt würden.

Einige Beispiele für Aufgaben in diesen Produktionspipelines sind:

- Verwendung einer Photogrammetrie-Anwendung zur Verarbeitung von Fotos, die von einem Filmset aufgenommen wurden, um ein strukturiertes digitales Netz zu rekonstruieren.
- Durchführung einer Partikelsimulation in einer 3D-Szene, um einem visuellen Explosionseffekt für eine Fernsehsendung Detailebenen hinzuzufügen.
- Daten für ein Spiellevel in die Form bringen, die für die externe Veröffentlichung erforderlich ist, und Anwenden der Optimierungs- und Komprimierungseinstellungen.
- Rendern einer Reihe von Bildern für einen Produktkatalog, einschließlich Variationen in Farbe, Hintergrund und Beleuchtung.

- Ein speziell entwickeltes Drehbuch auf einem 3D-Modell ausführen, um einen Look anzuwenden, der von einem Filmregisseur maßgeschneidert und genehmigt wurde.

Bei diesen Aufgaben müssen viele Parameter angepasst werden, um ein künstlerisches Ergebnis zu erzielen oder die Ausgabequalität zu optimieren. Oft gibt es eine grafische Benutzeroberfläche, mit der diese Parameterwerte über eine Schaltfläche oder ein Menü ausgewählt werden können, um den Prozess lokal in der Anwendung auszuführen. Wenn ein Benutzer den Prozess ausführt, können die Anwendung und möglicherweise der Host-Computer selbst nicht für andere Operationen verwendet werden, da er den Anwendungsstatus im Arbeitsspeicher verwendet und möglicherweise alle CPU- und Speicherressourcen des Host-Computers beansprucht.

In vielen Fällen ist der Vorgang schnell. Im Laufe der Produktion verlangsamt sich die Geschwindigkeit des Prozesses, wenn die Anforderungen an Qualität und Komplexität steigen. Aus einem Charaktertest, der während der Entwicklung 30 Sekunden gedauert hat, können leicht 3 Stunden werden, wenn er auf den endgültigen Produktionscharakter angewendet wird. Durch diesen Fortschritt kann ein Workload, der seinen Ursprung in einer GUI hatte, zu groß werden, um hineinzupassen. Die Portierung auf Deadline Cloud kann die Produktivität der Benutzer steigern, die diese Prozesse ausführen, da sie wieder die volle Kontrolle über ihre Workstation erhalten und weitere Iterationen vom Deadline Cloud-Monitor aus verfolgen können.

Bei der Entwicklung von Support für einen Workload in Deadline Cloud sollten zwei Unterstützungsebenen angestrebt werden:

- Verlagerung der Arbeitslast von der Benutzerarbeitsstation auf eine Deadline Cloud-Farm ohne Parallelität oder Beschleunigung. Dadurch werden die verfügbaren Rechenressourcen in der Farm möglicherweise nicht ausreichend genutzt, aber die Möglichkeit, lange Operationen auf ein Stapelverarbeitungssystem zu verlagern, ermöglicht es Benutzern, mehr mit ihrer eigenen Workstation zu erledigen.
- Optimierung der Parallelität der Arbeitslast, sodass die horizontale Skalierung der Deadline Cloud-Farm für eine schnelle Ausführung genutzt wird.

Manchmal ist es offensichtlich, wie ein Workload parallel ausgeführt werden kann. Beispielsweise kann jeder Frame eines Computergrafik-Renders unabhängig voneinander ausgeführt werden. Es ist jedoch wichtig, dass Sie sich nicht an dieser Parallelität festsetzen. Machen Sie sich stattdessen bewusst, dass die Auslagerung eines Workloads mit langer Laufzeit in Deadline Cloud erhebliche Vorteile bietet, auch wenn es keine offensichtliche Möglichkeit gibt, den Workload aufzuteilen.

Die Bestandteile eines Workloads

Um einen Deadline Cloud-Workload anzugeben, implementieren Sie ein Job-Bundle, das Benutzer mit der Deadline Cloud-CLI an eine Warteschlange senden. Ein Großteil der Arbeit bei der Erstellung eines Job-Bundles besteht darin, die Jobvorlage zu schreiben, aber es gibt noch mehr Faktoren wie die Bereitstellung der Anwendungen, die für den Workload erforderlich sind. Hier sind die wichtigsten Dinge, die Sie bei der Definition eines Workloads für Deadline Cloud beachten sollten:

- Die Anwendung, die ausgeführt werden soll. Der Job muss in der Lage sein, Anwendungsprozesse zu starten, und erfordert daher eine Installation der verfügbaren Anwendung sowie aller von der Anwendung verwendeten Lizenzen, z. B. den Zugriff auf einen Floating-Lizenzserver. Dies ist in der Regel Teil der Farmkonfiguration und nicht in das Auftragspaket selbst eingebettet.
  - Jobs mithilfe von Warteschlangenumgebungen konfigurieren
  - Vom Kunden verwaltete Flotten mit einem Lizenzendpunkt Connect
- Definitionen von Jobparametern. Die Benutzererfahrung beim Absenden des Jobs wird stark von den bereitgestellten Parametern beeinflusst. Zu den Beispielparametern gehören Datendateien, Verzeichnisse und die Anwendungskonfiguration.
  - Parameterwerte, Elemente für Job-Bundles
- Dateidatenfluss. Wenn ein Job ausgeführt wird, liest er Eingaben aus Dateien, die vom Benutzer bereitgestellt wurden, und schreibt seine Ausgabe dann als neue Dateien. Um mit den Funktionen für Auftragsanhänge und Pfadzuordnungen arbeiten zu können, muss der Job die Pfade der Verzeichnisse oder bestimmter Dateien für diese Eingaben und Ausgaben angeben.
  - Dateien in Ihren Jobs verwenden
- Das Schrittskript. Das Schrittskript führt die Anwendungsbinärdatei mit den richtigen Befehlszeilenoptionen aus, um die angegebenen Jobparameter anzuwenden. Es verarbeitet auch Details wie die Pfadzuweisung, wenn die Workload-Datendateien absolute statt relative Pfadverweise enthalten.
  - Jobvorlagenelemente für Jobpakete

Portabilität von Workloads

Ein Workload ist portabel, wenn er auf mehreren verschiedenen Systemen ausgeführt werden kann, ohne dass er jedes Mal geändert wird, wenn Sie einen Job einreichen. Er kann beispielsweise auf verschiedenen Renderfarmen ausgeführt werden, auf denen unterschiedliche gemeinsam genutzte Dateisysteme installiert sind, oder auf unterschiedlichen Betriebssystemen wie Linux oderWindows.

Wenn Sie ein portables Auftragspaket implementieren, ist es für Benutzer einfacher, den Job in ihrer spezifischen Farm auszuführen oder ihn für andere Anwendungsfälle anzupassen.

Im Folgenden finden Sie einige Möglichkeiten, wie Sie Ihr Job-Bundle portabel machen können.

- Geben Sie mithilfe von Job-Parametern und Asset-Referenzen im `PATH` Job-Bundle die für einen Workload benötigten Eingabedatendateien vollständig an. Durch diesen Ansatz kann der Job auf Farmen übertragen werden, die auf gemeinsam genutzten Dateisystemen basieren, und auf Farmen, die Kopien der Eingabedaten erstellen, wie z. B. die Deadline Cloud-Funktion für Jobanhänge.
- Machen Sie Dateipfadverweise für die Eingabedateien des Jobs verschiebbar und auf verschiedenen Betriebssystemen nutzbar. Zum Beispiel, wenn Benutzer Jobs von Windows Workstations aus einreichen, um sie auf einer Flotte auszuführen. Linux
  - Verwenden Sie relative Dateipfadverweise. Wenn also das Verzeichnis, das sie enthält, an einen anderen Ort verschoben wird, werden die Verweise trotzdem aufgelöst. Einige Anwendungen, wie Blender, unterstützen die Wahl zwischen relativen und absoluten Pfaden.
  - Wenn Sie keine relativen Pfade verwenden können, unterstützen Sie OpenJD-Pfad-Mapping-Metadaten und übersetzen Sie die absoluten Pfade entsprechend der Art und Weise, wie Deadline Cloud die Dateien für den Job bereitstellt.
- Implementieren Sie Befehle in einem Job mithilfe portabler Skripts. Python und Bash sind zwei Beispiele für Skriptsprachen, die auf diese Weise verwendet werden können. Sie sollten erwägen, beide auf allen Worker-Hosts Ihrer Flotten bereitzustellen.
  - Verwenden Sie die Binärdatei des Skriptinterpreters, wie `python` oder`bash`, mit dem Namen der Skriptdatei als Argument. Dieser Ansatz funktioniert auf allen BetriebssystemenWindows, auch im Vergleich zur Verwendung einer Skriptdatei, deren Ausführungsbit aktiviert Linux ist.
  - Schreiben Sie portable Bash-Skripte, indem Sie die folgenden Methoden anwenden:
    - Erweitern Sie die Pfadparameter der Vorlage in einfache Anführungszeichen, um Pfade mit Leerzeichen und Windows Pfadtrennzeichen zu behandeln.
    - Achten Sie bei der Ausführung auf Windows Probleme im Zusammenhang mit der automatischen MinGW-Pfadübersetzung. Es wandelt beispielsweise einen AWS CLI Befehl wie `aws logs tail /aws/deadline/...` in einen Befehl um, der einem Protokoll ähnlich ist `aws logs tail "C:/Program Files/Git/aws/deadline/..."` und nicht korrekt weiterleitet. Stellen Sie die Variable `MSYS_NO_PATHCONV=1` ein, um dieses Verhalten auszuschalten.
    - In den meisten Fällen funktioniert derselbe Code auf allen Betriebssystemen. Wenn der Code anders sein muss, verwenden Sie ein `if/else` Konstrukt, um die Fälle zu behandeln.

```
if [[ "$(uname)" == MINGW* || "$(uname -s)" == MSYS_NT* ]]; then
    # Code for Windows
elif [[ "$(uname)" == Darwin ]]; then
    # Code for MacOS
else
    # Code for Linux and other operating systems
fi
```

  - Sie können portable Python-Skripte schreiben, `pathlib` um Pfadunterschiede im Dateisystem zu behandeln und betriebsspezifische Funktionen zu vermeiden. Die Python-Dokumentation enthält Anmerkungen dazu, beispielsweise in der Dokumentation zur Signalbibliothek. LinuxDie Unterstützung spezifischer Funktionen ist als „Verfügbarkeit: Linux“ gekennzeichnet.
- Verwenden Sie die Jobparameter, um die Anwendungsanforderungen zu spezifizieren. Verwenden Sie konsistente Konventionen, die der Farmadministrator in Warteschlangenumgebungen anwenden kann.
  - Sie können beispielsweise die `RezPackages` Parameter `CondaPackages` und/oder in Ihrem Job mit einem Standardparameterwert verwenden, der die Namen und Versionen der Anwendungspakete auflistet, die der Job benötigt. Anschließend können Sie eine der Beispiel-Warteschlangenumgebungen Conda oder Rez verwenden, um eine virtuelle Umgebung für den Job bereitzustellen.

Erste Schritte mit Deadline Cloud-Ressourcen

Um mit der Erstellung benutzerdefinierter Lösungen für AWS Deadline Cloud zu beginnen, müssen Sie Ihre Ressourcen einrichten. Dazu gehören eine Farm, mindestens eine Warteschlange für die Farm und mindestens eine Arbeiterflotte zur Bedienung der Warteschlange. Sie können Ihre Ressourcen mit der Deadline Cloud-Konsole erstellen, oder Sie können die verwenden AWS Command Line Interface.

In diesem Tutorial werden Sie verwenden, AWS CloudShell um eine einfache Entwicklerfarm zu erstellen und den Worker-Agent auszuführen. Anschließend können Sie einen einfachen Job mit Parametern und Anhängen einreichen und ausführen, eine vom Service verwaltete Flotte hinzufügen und Ihre Farmressourcen bereinigen, wenn Sie fertig sind.

In den folgenden Abschnitten lernen Sie die verschiedenen Funktionen von Deadline Cloud kennen und erfahren, wie sie funktionieren und zusammenarbeiten. Das Befolgen dieser Schritte ist nützlich, um neue Workloads und Anpassungen zu entwickeln und zu testen.

Anweisungen zum Einrichten Ihrer Farm mithilfe der Konsole finden Sie unter Erste Schritte im Deadline Cloud-Benutzerhandbuch.

Themen

Erstellen Sie eine Deadline Cloud-Farm

Verwenden Sie die AWS Command Line Interface (AWS CLI), wie im folgenden Verfahren gezeigt, um Ihre Entwicklerfarm und Warteschlangenressourcen in AWS Deadline Cloud zu erstellen. Außerdem erstellen Sie eine AWS Identity and Access Management (IAM) -Rolle und eine vom Kunden verwaltete Flotte (CMF) und ordnen die Flotte Ihrer Warteschlange zu. Anschließend können Sie das konfigurieren AWS CLI und sicherstellen, dass Ihre Farm wie angegeben eingerichtet ist und funktioniert.

Sie können diese Farm verwenden, um die Funktionen von Deadline Cloud zu erkunden und anschließend neue Workloads, Anpassungen und Pipeline-Integrationen zu entwickeln und zu testen.

Um eine Farm zu erstellen

1. Öffnen Sie eine AWS CloudShell Sitzung. Sie verwenden das CloudShell Fenster, um Befehle AWS Command Line Interface (AWS CLI) einzugeben, um die Beispiele in diesem Tutorial auszuführen. Lassen Sie das CloudShell Fenster geöffnet, während Sie fortfahren.
2. Erstellen Sie einen Namen für Ihre Farm und fügen Sie diesen Farmnamen hinzu`~/.bashrc`. Dadurch wird es für andere Terminalsitzungen verfügbar.

```
echo "DEV_FARM_NAME=DeveloperFarm" >> ~/.bashrc
source ~/.bashrc
```

3. Erstellen Sie die Farmressource und fügen Sie ihre Farm-ID hinzu`~/.bashrc`.

```
aws deadline create-farm \
    --display-name "$DEV_FARM_NAME"

echo "DEV_FARM_ID=\$(aws deadline list-farms \
        --query \"farms[?displayName=='\$DEV_FARM_NAME'].farmId \
        | [0]\" --output text)" >> ~/.bashrc
source ~/.bashrc
```

4. Erstellen Sie die Warteschlangenressource und fügen Sie ihre Warteschlangen-ID hinzu `~/.bashrc`.

```
aws deadline create-queue \
    --farm-id $DEV_FARM_ID \
    --display-name "$DEV_FARM_NAME Queue" \
    --job-run-as-user '{"posix": {"user": "job-user", "group": "job-group"},
 "runAs":"QUEUE_CONFIGURED_USER"}'

echo "DEV_QUEUE_ID=\$(aws deadline list-queues \
        --farm-id \$DEV_FARM_ID \
        --query \"queues[?displayName=='\$DEV_FARM_NAME Queue'].queueId \
        | [0]\" --output text)" >> ~/.bashrc
source ~/.bashrc
```

5. Erstellen Sie eine IAM-Rolle für die Flotte. Diese Rolle bietet Worker-Hosts in Ihrer Flotte die erforderlichen Sicherheitsanmeldedaten, um Jobs aus Ihrer Warteschlange auszuführen.

```
aws iam create-role \
    --role-name "${DEV_FARM_NAME}FleetRole" \
    --assume-role-policy-document \
        '{
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "credentials.deadline.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        }'
aws iam put-role-policy \
    --role-name "${DEV_FARM_NAME}FleetRole" \
    --policy-name WorkerPermissions \
    --policy-document \
        '{
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Action": [
                        "deadline:AssumeFleetRoleForWorker",
                        "deadline:UpdateWorker",
                        "deadline:DeleteWorker",
                        "deadline:UpdateWorkerSchedule",
                        "deadline:BatchGetJobEntity",
                        "deadline:AssumeQueueRoleForWorker"
                    ],
                    "Resource": "*",
                    "Condition": {
                        "StringEquals": {
                            "aws:PrincipalAccount": "${aws:ResourceAccount}"
                        }
                    }
                },
                {
                    "Effect": "Allow",
                    "Action": [
                        "logs:CreateLogStream"
```

```
                ],
                "Resource": "arn:aws:logs:*:*:*:/aws/deadline/*",
                "Condition": {
                    "StringEquals": {
                        "aws:PrincipalAccount": "${aws:ResourceAccount}"
                    }
                }
            },
            {
                "Effect": "Allow",
                "Action": [
                    "logs:PutLogEvents",
                    "logs:GetLogEvents"
                ],
                "Resource": "arn:aws:logs:*:*:*:/aws/deadline/*",
                "Condition": {
                    "StringEquals": {
                        "aws:PrincipalAccount": "${aws:ResourceAccount}"
                    }
                }
            }
        ]
    }'
```

6. Erstellen Sie die vom Kunden verwaltete Flotte (CMF) und fügen Sie deren Flotten-ID hinzu. `~/.bashrc`

```
FLEET_ROLE_ARN="arn:aws:iam::$(aws sts get-caller-identity \
        --query "Account" --output text):role/${DEV_FARM_NAME}FleetRole"
aws deadline create-fleet \
    --farm-id $DEV_FARM_ID \
    --display-name "$DEV_FARM_NAME CMF" \
    --role-arn $FLEET_ROLE_ARN \
    --max-worker-count 5 \
    --configuration \
        '{
            "customerManaged": {
                "mode": "NO_SCALING",
                "workerCapabilities": {
                    "vCpuCount": {"min": 1},
                    "memoryMiB": {"min": 512},
                    "osFamily": "linux",
                    "cpuArchitectureType": "x86_64"
```

```
                }
            }
        }'

echo "DEV_CMF_ID=\$(aws deadline list-fleets \
        --farm-id \$DEV_FARM_ID \
        --query \"fleets[?displayName=='\$DEV_FARM_NAME CMF'].fleetId \
        | [0]\" --output text)" >> ~/.bashrc
source ~/.bashrc
```

7. Ordnen Sie das CMF Ihrer Warteschlange zu.

```
aws deadline create-queue-fleet-association \
    --farm-id $DEV_FARM_ID \
    --queue-id $DEV_QUEUE_ID \
    --fleet-id $DEV_CMF_ID
```

8. Installieren Sie die Deadline Cloud-Befehlszeilenschnittstelle.

```
pip install deadline
```

9. Verwenden Sie den folgenden Befehl, um die Standardfarm auf die Farm-ID und die Warteschlange auf die Warteschlangen-ID festzulegen, die Sie zuvor erstellt haben.

```
deadline config set defaults.farm_id $DEV_FARM_ID
deadline config set defaults.queue_id $DEV_QUEUE_ID
```

10. (Optional) Verwenden Sie die folgenden Befehle, um zu überprüfen, ob Ihre Farm gemäß Ihren Spezifikationen eingerichtet wurde:

    - Alle Farmen auflisten — **`deadline farm list`**
    - Alle Warteschlangen in der Standardfarm auflisten — **`deadline queue list`**
    - Alle Flotten in der Standardfarm auflisten — **`deadline fleet list`**
    - Holen Sie sich die Standardfarm — **`deadline farm get`**
    - Holen Sie sich die Standardwarteschlange — **`deadline queue get`**
    - Ruft alle Flotten ab, die der Standardwarteschlange zugeordnet sind — **`deadline fleet get`**

Nächste Schritte

Nachdem Sie Ihre Farm erstellt haben, können Sie den Deadline Cloud-Worker-Agent auf den Hosts in Ihrer Flotte ausführen, um Jobs zu verarbeiten. Siehe Führen Sie den Deadline Cloud Worker Agent aus.

Führen Sie den Deadline Cloud Worker Agent aus

Bevor Sie die Jobs, die Sie an die Warteschlange auf Ihrer Entwicklerfarm einreichen, ausführen können, müssen Sie den AWS Deadline Cloud-Worker-Agent im Entwicklermodus auf einem Worker-Host ausführen.

Im weiteren Verlauf dieses Tutorials führen Sie mithilfe von zwei AWS CloudShell Tabs AWS CLI Operationen auf Ihrer Entwicklerfarm durch. Auf der ersten Registerkarte können Sie Jobs einreichen. Auf der zweiten Registerkarte können Sie den Worker Agent ausführen.

> **Note**
>
> Wenn Sie Ihre CloudShell Sitzung länger als 20 Minuten inaktiv lassen, kommt es zu einem Timeout und der Worker-Agent wird gestoppt. Folgen Sie den Anweisungen im folgenden Verfahren, um den Worker-Agent neu zu starten.

Bevor Sie einen Worker Agent starten können, müssen Sie eine Deadline Cloud-Farm, eine Warteschlange und eine Flotte einrichten. Siehe Erstellen Sie eine Deadline Cloud-Farm.

Um den Worker Agent im Entwicklermodus auszuführen

1. Wenn Ihre Farm immer noch auf der ersten CloudShell Registerkarte geöffnet ist, öffnen Sie eine zweite CloudShell Registerkarte und erstellen Sie dann die `demoenv-persist` Verzeichnisse `demoenv-logs` und.

   ```
   mkdir ~/demoenv-logs
   mkdir ~/demoenv-persist
   ```

2. Laden Sie die Deadline Cloud Worker Agent-Pakete von PyPI herunter und installieren Sie sie:

> **Note**
>
> Ein Windows, es ist erforderlich, dass die Agentendateien im globalen Site-Packages-Verzeichnis von Python installiert werden. Virtuelle Python-Umgebungen werden derzeit nicht unterstützt.

```
python -m pip install deadline-cloud-worker-agent
```

3. Damit der Worker-Agent die temporären Verzeichnisse für die Ausführung von Jobs erstellen kann, erstellen Sie ein Verzeichnis:

```
sudo mkdir /sessions
sudo chmod 750 /sessions
sudo chown cloudshell-user /sessions
```

4. Führen Sie den Deadline Cloud-Worker-Agent im Entwicklermodus mit den Variablen `DEV_FARM_ID` aus`DEV_CMF_ID`, die Sie dem hinzugefügt haben`~/.bashrc`.

```
deadline-worker-agent \
    --farm-id $DEV_FARM_ID \
    --fleet-id $DEV_CMF_ID \
    --run-jobs-as-agent-user \
    --logs-dir ~/demoenv-logs \
    --persistence-dir ~/demoenv-persist
```

Während der Worker-Agent den `UpdateWorkerSchedule` API-Vorgang initialisiert und dann abfragt, wird die folgende Ausgabe angezeigt:

```
INFO    Worker Agent starting
[2024-03-27 15:51:01,292][INFO    ] # Worker Agent starting
[2024-03-27 15:51:01,292][INFO    ] AgentInfo
Python Interpreter: /usr/bin/python3
Python Version: 3.9.16 (main, Sep  8 2023, 00:00:00)  - [GCC 11.4.1 20230605 (Red
 Hat 11.4.1-2)]
Platform: linux
...
```

```
[2024-03-27 15:51:02,528][INFO    ] # API.Resp # [deadline:UpdateWorkerSchedule]
(200) params={'assignedSessions': {}, 'cancelSessionActions': {},
 'updateIntervalSeconds': 15} ...
[2024-03-27 15:51:17,635][INFO    ] # API.Resp # [deadline:UpdateWorkerSchedule]
(200) params=(Duplicate removed, see previous response) ...
[2024-03-27 15:51:32,756][INFO    ] # API.Resp # [deadline:UpdateWorkerSchedule]
(200) params=(Duplicate removed, see previous response) ...
...
```

5. Wählen Sie Ihre erste CloudShell Registerkarte aus und listen Sie dann die Mitarbeiter in der Flotte auf.

   ```
   deadline worker list --fleet-id $DEV_CMF_ID
   ```

   Es wird eine Ausgabe wie die folgende angezeigt:

   ```
   Displaying 1 of 1 workers starting at 0

   - workerId: worker-8c9af877c8734e89914047111f
     status: STARTED
     createdAt: 2023-12-13 20:43:06+00:00
   ```

In einer Produktionskonfiguration erfordert der Deadline Cloud-Worker-Agent die Einrichtung mehrerer Benutzer und Konfigurationsverzeichnisse als Administratorbenutzer auf dem Host-Computer. Sie können diese Einstellungen überschreiben, da Sie Jobs in Ihrer eigenen Entwicklungsfarm ausführen, auf die nur Sie zugreifen können.

Nächste Schritte

Jetzt, da ein Worker-Agent auf Ihren Worker-Hosts ausgeführt wird, können Sie Jobs an Ihre Mitarbeiter senden. Sie haben folgende Möglichkeiten:

- Mit Deadline Cloud einreichenmit einem einfachen OpenJD-Jobpaket.
- Jobs mit Stellenanhängen in Deadline Cloud einreichendie Dateien zwischen Workstations teilen, die unterschiedliche Betriebssysteme verwenden.

Mit Deadline Cloud einreichen

Um Deadline Cloud-Jobs auf Ihren Worker-Hosts auszuführen, erstellen und verwenden Sie ein Open Job Description (OpenJD) -Jobpaket, um einen Job zu konfigurieren. Das Bundle konfiguriert den Job, indem es beispielsweise Eingabedateien für einen Job angibt und wo die Ausgabe des Jobs geschrieben werden soll. Dieses Thema enthält Beispiele dafür, wie Sie ein Job-Bundle konfigurieren können.

Bevor Sie die Verfahren in diesem Abschnitt ausführen können, müssen Sie die folgenden Schritte ausführen:

- Erstellen Sie eine Deadline Cloud-Farm
- Führen Sie den Deadline Cloud Worker Agent aus

Gehen Sie wie folgt vor, um AWS Deadline Cloud zum Ausführen von Jobs zu verwenden. Verwenden Sie die erste AWS CloudShell Registerkarte, um Jobs an Ihre Entwicklerfarm zu senden. Verwenden Sie die zweite CloudShell Registerkarte, um die Ergebnisse des Worker-Agents anzuzeigen.

Themen

Reichen Sie die simple_job Probe ein

Nachdem Sie eine Farm erstellt und den Worker Agent ausgeführt haben, können Sie das simple_job Beispiel an Deadline Cloud senden.

Um das simple_job Beispiel an Deadline Cloud zu senden

1. Wählen Sie Ihren ersten CloudShell Tab.
2. Laden Sie das Beispiel von herunter GitHub.

```
cd ~
```

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
```

3. Navigieren Sie zum Verzeichnis mit den Beispielen für das Job-Bundle.

```
cd ~/deadline-cloud-samples/job_bundles/
```

4. Reichen Sie das simple_job Beispiel ein.

```
deadline bundle submit simple_job
```

5. Wählen Sie Ihre zweite CloudShell Registerkarte, um die Protokollausgabe über das Aufrufen`BatchGetJobEntities`, das Aufrufen einer Sitzung und das Ausführen einer Sitzungsaktion anzuzeigen.

```
...
[2024-03-27 16:00:21,846][INFO    ] # Session.Starting
 # [session-053d77cef82648fe2] Starting new Session.
 [queue-3ba4ff683ff54db09b851a2ed8327d7b/job-d34cc98a6e234b6f82577940ab4f76c6]
[2024-03-27 16:00:21,853][INFO    ] # API.Req # [deadline:BatchGetJobEntity]
 resource={'farm-id': 'farm-3e24cfc9bbcd423e9c1b6754bc1',
 'fleet-id': 'fleet-246ee60f46d44559b6cce010d05', 'worker-id':
 'worker-75e0fce9c3c344a69bff57fcd83'} params={'identifiers': [{'jobDetails':
 {'jobId': 'job-d34cc98a6e234b6f82577940ab4'}}]} request_url=https://
scheduling.deadline.us-west-2.amazonaws.com/2023-10-12/farms/
farm-3e24cfc9bbcd423e /fleets/fleet-246ee60f46d44559b1 /workers/worker-
 75e0fce9c3c344a69b /batchGetJobEntity
[2024-03-27 16:00:22,013][INFO    ] # API.Resp # [deadline:BatchGetJobEntity](200)
 params={'entities': [{'jobDetails': {'jobId': 'job-d34cc98a6e234b6f82577940ab6',
 'jobRunAsUser': {'posix': {'user': 'job-user', 'group': 'job-group'},
 'runAs': 'QUEUE_CONFIGURED_USER'}, 'logGroupName': '/aws/deadline/
farm-3e24cfc9bbcd423e9c1b6754bc1/queue-3ba4ff683ff54db09b851a2ed83', 'parameters':
 '*REDACTED*', 'schemaVersion': 'jobtemplate-2023-09'}}], 'errors': []}
 request_id=a3f55914-6470-439e-89e5-313f0c6
[2024-03-27 16:00:22,013][INFO    ] # Session.Add #
 [session-053d77cef82648fea9c69827182] Appended new SessionActions.
 (ActionIds: ['sessionaction-053d77cef82648fea9c69827182-0'])
 [queue-3ba4ff683ff54db09b851a2ed8b/job-d34cc98a6e234b6f82577940ab6]
[2024-03-27 16:00:22,014][WARNING ] # Session.User #
 [session-053d77cef82648fea9c69827182] Running as the Worker Agent's
 user. (User: cloudshell-user) [queue-3ba4ff683ff54db09b851a2ed8b/job-
d34cc98a6e234b6f82577940ac6]
```

```
[2024-03-27 16:00:22,015][WARNING ] # Session.AWSCreds #
 [session-053d77cef82648fea9c69827182] AWS Credentials are not available: Queue has
 no IAM Role. [queue-3ba4ff683ff54db09b851a2ed8b/job-d34cc98a6e234b6f82577940ab6]
[2024-03-27 16:00:22,026][INFO    ] # Session.Logs #
 [session-053d77cef82648fea9c69827182] Logs streamed to: AWS CloudWatch
 Logs. (LogDestination: /aws/deadline/farm-3e24cfc9bbcd423e9c1b6754bc1/
queue-3ba4ff683ff54db09b851a2ed83/session-053d77cef82648fea9c69827181)
 [queue-3ba4ff683ff54db09b851a2ed83/job-d34cc98a6e234b6f82577940ab4]
[2024-03-27 16:00:22,026][INFO    ] # Session.Logs #
 [session-053d77cef82648fea9c69827182] Logs streamed to: local
 file. (LogDestination: /home/cloudshell-user/demoenv-logs/
queue-3ba4ff683ff54db09b851a2ed8b/session-053d77cef82648fea9c69827182.log)
 [queue-3ba4ff683ff54db09b851a2ed83/job-d34cc98a6e234b6f82577940ab4]
...
```

> **Note**
>
> Es werden nur die Protokollausgaben des Worker-Agenten angezeigt. Es gibt ein separates Protokoll für die Sitzung, in der der Job ausgeführt wird.

6. Wählen Sie Ihre erste Registerkarte und überprüfen Sie dann die Protokolldateien, die der Worker-Agent schreibt.

   a. Navigieren Sie zum Protokollverzeichnis des Worker-Agents und sehen Sie sich dessen Inhalt an.

   ```
   cd ~/demoenv-logs
   ls
   ```

   b. Drucken Sie die erste Protokolldatei, die der Worker-Agent erstellt.

   ```
   cat worker-agent-bootstrap.log
   ```

   Diese Datei enthält die Ergebnisse des Worker-Agents darüber, wie er die Deadline Cloud-API aufgerufen hat, um eine Worker-Ressource in Ihrer Flotte zu erstellen, und dann die Flottenrolle übernommen hat.

   c. Druckt die Protokolldatei aus, die ausgegeben wird, wenn der Worker Agent der Flotte beitritt.

```
cat worker-agent.log
```

Dieses Protokoll enthält Ausgaben über alle Aktionen, die der Worker-Agent ausführt, enthält jedoch keine Ausgaben über die Warteschlangen, in denen er Jobs ausführt, mit Ausnahme IDs der Ressourcen.

d. Druckt die Protokolldateien für jede Sitzung in einem Verzeichnis, das den gleichen Namen wie die Ressourcen-ID der Warteschlange hat.

```
cat $DEV_QUEUE_ID/session-*.log
```

Wenn der Auftrag erfolgreich ist, sieht die Ausgabe der Protokolldatei wie folgt aus:

```
cat $DEV_QUEUE_ID/$(ls -t $DEV_QUEUE_ID | head -1)

2024-03-27 16:00:22,026 WARNING Session running with no AWS Credentials.
2024-03-27 16:00:22,404 INFO
2024-03-27 16:00:22,405 INFO ==============================================
2024-03-27 16:00:22,405 INFO --------- Running Task
2024-03-27 16:00:22,405 INFO ==============================================
2024-03-27 16:00:22,406 INFO ----------------------------------------------
2024-03-27 16:00:22,406 INFO Phase: Setup
2024-03-27 16:00:22,406 INFO ----------------------------------------------
2024-03-27 16:00:22,406 INFO Writing embedded files for Task to disk.
2024-03-27 16:00:22,406 INFO Mapping: Task.File.runScript -> /sessions/
session-053d77cef82648fea9c698271812a/embedded_fileswa_gj55_/tmp2u9yqtsz
2024-03-27 16:00:22,406 INFO Wrote: runScript -> /sessions/
session-053d77cef82648fea9c698271812a/embedded_fileswa_gj55_/tmp2u9yqtsz
2024-03-27 16:00:22,407 INFO ----------------------------------------------
2024-03-27 16:00:22,407 INFO Phase: Running action
2024-03-27 16:00:22,407 INFO ----------------------------------------------
2024-03-27 16:00:22,407 INFO Running command /sessions/
session-053d77cef82648fea9c698271812a/tmpzuzxpslm.sh
2024-03-27 16:00:22,414 INFO Command started as pid: 471
2024-03-27 16:00:22,415 INFO Output:
2024-03-27 16:00:22,420 INFO Welcome to AWS Deadline Cloud!
2024-03-27 16:00:22,571 INFO
2024-03-27 16:00:22,572 INFO ==============================================
2024-03-27 16:00:22,572 INFO --------- Session Cleanup
2024-03-27 16:00:22,572 INFO ==============================================
```

```
2024-03-27 16:00:22,572 INFO Deleting working directory: /sessions/
session-053d77cef82648fea9c698271812a
```

7. Druckt Informationen über den Job.

```
deadline job get
```

Wenn Sie den Job weiterleiten, speichert das System ihn als Standard, sodass Sie die Job-ID nicht eingeben müssen.

Reichen Sie eine simple_job mit einem Parameter ein

Sie können Jobs mit Parametern einreichen. Im folgenden Verfahren bearbeiten Sie die simple_job Vorlage so, dass sie eine benutzerdefinierte Nachricht enthält, senden die und drucken dann die Sitzungsprotokolldatei aussimple_job, um die Nachricht anzuzeigen.

Um das simple_job Beispiel mit einem Parameter einzureichen

1. Wählen Sie Ihre erste CloudShell Registerkarte aus und navigieren Sie dann zum Verzeichnis mit den Job-Bundle-Beispielen.

```
cd ~/deadline-cloud-samples/job_bundles/
```

2. Drucken Sie den Inhalt der simple_job Vorlage aus.

```
cat simple_job/template.yaml
```

Der `parameterDefinitions` Abschnitt mit dem `Message` Parameter sollte wie folgt aussehen:

```
parameterDefinitions:
- name: Message
  type: STRING
  default: Welcome to AWS Deadline Cloud!
```

3. Senden Sie das simple_job Beispiel mit einem Parameterwert und warten Sie dann, bis die Ausführung des Jobs abgeschlossen ist.

```
deadline bundle submit simple_job \
    -p "Message=Greetings from the developer getting started guide."
```

4. Um die benutzerdefinierte Nachricht zu sehen, sehen Sie sich die letzte Sitzungsprotokolldatei an.

```
cd ~/demoenv-logs
cat $DEV_QUEUE_ID/$(ls -t $DEV_QUEUE_ID | head -1)
```

Erstellen Sie ein simple_file_job-Job-Bundle mit Datei-I/O

Ein Renderjob muss die Szenendefinition lesen, daraus ein Bild rendern und dieses Bild dann in einer Ausgabedatei speichern. Sie können diese Aktion simulieren, indem Sie den Job den Hash der Eingabe berechnen lassen, anstatt ein Bild zu rendern.

Um ein simple_file_job-Job-Paket mit Datei-I/O zu erstellen

1. Wählen Sie Ihre erste CloudShell Registerkarte aus und navigieren Sie dann zum Verzeichnis mit den Job-Bundle-Beispielen.

```
cd ~/deadline-cloud-samples/job_bundles/
```

2. Erstellen Sie eine Kopie von `simple_job` mit dem neuen Namen`simple_file_job`.

```
cp -r simple_job simple_file_job
```

3. Bearbeiten Sie die Jobvorlage wie folgt:

> **Note**
>
> Wir empfehlen Ihnen, nano für diese Schritte die Verwendung zu verwenden. Wenn Sie lieber verwenden möchtenVim, müssen Sie den Einfügemodus über einstellen`:set paste`.

   a. Öffnen Sie die Vorlage in einem Texteditor.

```
nano simple_file_job/template.yaml
```

   b. Fügen Sie die folgenden `type`, `object`, und hinzu `dataFlow`, `parameter`, `Definitions`.

```
- name: InFile
  type: PATH
  objectType: FILE
  dataFlow: IN
- name: OutFile
  type: PATH
  objectType: FILE
  dataFlow: OUT
```

c. Fügen Sie den folgenden `bash` Skriptbefehl am Ende der Datei hinzu, die aus der Eingabedatei liest und in die Ausgabedatei schreibt.

```
        # hash the input file, and write that to the output
        sha256sum "{{Param.InFile}}" > "{{Param.OutFile}}"
```

Die aktualisierte Version `template.yaml` sollte genau den folgenden Angaben entsprechen:

```
specificationVersion: 'jobtemplate-2023-09'
name: Simple File Job Bundle Example
parameterDefinitions:
- name: Message
  type: STRING
  default: Welcome to AWS Deadline Cloud!
- name: InFile
  type: PATH
  objectType: FILE
  dataFlow: IN
- name: OutFile
  type: PATH
  objectType: FILE
  dataFlow: OUT
steps:
- name: WelcomeToDeadlineCloud
  script:
    actions:
      onRun:
        command: '{{Task.File.Run}}'
    embeddedFiles:
    - name: Run
      type: TEXT
```

```
      runnable: true
      data: |
        #!/usr/bin/env bash
        echo "{{Param.Message}}"

        # hash the input file, and write that to the output
        sha256sum "{{Param.InFile}}" > "{{Param.OutFile}}"
```

> **Note**
>
> Wenn Sie den Abstand in der anpassen möchten, stellen Sie sicher`template.yaml`, dass Sie Leerzeichen anstelle von Einzügen verwenden.

d. Speichern Sie die Datei und beenden Sie den Texteditor.

4. Geben Sie Parameterwerte für die Eingabe- und Ausgabedateien an, um den simple_file_job zu senden.

```
deadline bundle submit simple_file_job \
    -p "InFile=simple_job/template.yaml" \
    -p "OutFile=hash.txt"
```

5. Druckt Informationen über den Job.

```
deadline job get
```

- Sie werden eine Ausgabe wie die folgende sehen:

```
parameters:
  Message:
    string: Welcome to AWS Deadline Cloud!
  InFile:
    path: /local/home/cloudshell-user/BundleFiles/JobBundle-Examples/simple_job/
template.yaml
  OutFile:
    path: /local/home/cloudshell-user/BundleFiles/JobBundle-Examples/hash.txt
```

- Sie haben zwar nur relative Pfade angegeben, für die Parameter ist jedoch der vollständige Pfadsatz festgelegt. Der AWS CLI verknüpft das aktuelle Arbeitsverzeichnis mit allen Pfaden, die als Parameter bereitgestellt werden, wenn die Pfade den folgenden Typ habenPATH.

- Der Worker-Agent, der im anderen Terminalfenster ausgeführt wird, nimmt den Job auf und führt ihn aus. Diese Aktion erstellt die `hash.txt` Datei, die Sie mit dem folgenden Befehl anzeigen können.

  ```
  cat hash.txt
  ```

  Dieser Befehl druckt eine Ausgabe, die der folgenden ähnelt.

  ```
  eaa2df5d34b54be5ac34c56a24a8c237b8487231a607eaf530a04d76b89c9cd3 /local/home/
  cloudshell-user/BundleFiles/JobBundle-Examples/simple_job/template.yaml
  ```

Nächste Schritte

Nachdem Sie gelernt haben, wie Sie einfache Jobs mit der Deadline Cloud CLI einreichen, können Sie Folgendes erkunden:

- Jobs mit Stellenanhängen in Deadline Cloud einreichenum zu erfahren, wie man Jobs auf Hosts mit unterschiedlichen Betriebssystemen ausführt.
- Fügen Sie Ihrer Entwicklerfarm in Deadline Cloud eine vom Service verwaltete Flotte hinzuum Ihre Jobs auf Hosts auszuführen, die von Deadline Cloud verwaltet werden.
- Bereinigen Sie Ihre Farmressourcen in Deadline Cloudum die Ressourcen herunterzufahren, die Sie für dieses Tutorial verwendet haben.

Jobs mit Stellenanhängen in Deadline Cloud einreichen

Viele Farmen verwenden gemeinsam genutzte Dateisysteme, um Dateien zwischen den Hosts, die Jobs einreichen, und denen, die Jobs ausführen, gemeinsam zu nutzen. Im vorherigen Beispiel wird das lokale Dateisystem `simple_file_job` beispielsweise von den AWS CloudShell Terminalfenstern gemeinsam genutzt, die auf Registerkarte eins, wo Sie den Job einreichen, und Registerkarte zwei, wo Sie den Worker-Agent ausführen, ausgeführt werden.

Ein gemeinsam genutztes Dateisystem ist vorteilhaft, wenn sich die Worker-Hosts und die Worker-Hosts im selben lokalen Netzwerk befinden. Wenn Sie Ihre Daten vor Ort in der Nähe der Workstations speichern, die darauf zugreifen, bedeutet die Verwendung einer cloudbasierten Farm, dass Sie Ihre Dateisysteme über ein VPN mit hoher Latenz gemeinsam nutzen oder Ihre

Dateisysteme in der Cloud synchronisieren müssen. Keine dieser Optionen ist einfach einzurichten oder zu bedienen.

AWS Deadline Cloud bietet eine einfache Lösung mit Job-Anhängen, die E-Mail-Anhängen ähneln. Mit Job-Anhängen hängen Sie Daten an Ihren Job an. Anschließend kümmert sich Deadline Cloud um die Details der Übertragung und Speicherung Ihrer Auftragsdaten in Amazon Simple Storage Service (Amazon S3) -Buckets.

Workflows zur Erstellung von Inhalten sind oft iterativ, was bedeutet, dass ein Benutzer Jobs mit einer kleinen Teilmenge modifizierter Dateien einreicht. Da Amazon S3 S3-Buckets Auftragsanhänge in einem inhaltsadressierbaren Speicher speichern, basiert der Name jedes Objekts auf dem Hash der Objektdaten, und der Inhalt eines Verzeichnisbaums wird in einem Manifest-Dateiformat gespeichert, das an einen Auftrag angehängt ist.

Bevor Sie die Verfahren in diesem Abschnitt ausführen können, müssen Sie die folgenden Schritte ausführen:

- Erstellen Sie eine Deadline Cloud-Farm
- Führen Sie den Deadline Cloud Worker Agent aus

Gehen Sie wie folgt vor, um Jobs mit Auftragsanhängen auszuführen.

Themen

Fügen Sie Ihrer Warteschlange eine Konfiguration für Jobanhänge hinzu

Um Jobanhänge in Ihrer Warteschlange zu aktivieren, fügen Sie der Warteschlangenressource in Ihrem Konto eine Konfiguration für Jobanhänge hinzu.

Um Ihrer Warteschlange eine Konfiguration für Jobanhänge hinzuzufügen

1. Wählen Sie Ihre erste CloudShell Registerkarte und geben Sie dann einen der folgenden Befehle ein, um einen Amazon S3 S3-Bucket für Jobanhänge zu verwenden.

- Wenn Sie noch keinen privaten Amazon S3 S3-Bucket haben, können Sie einen neuen S3-Bucket erstellen und verwenden.

```
DEV_FARM_BUCKET=$(echo $DEV_FARM_NAME \
    | tr '[:upper:]' '[:lower:]')-$(xxd -l 16 -p /dev/urandom)
if [ "$AWS_REGION" == "us-east-1" ]; then LOCATION_CONSTRAINT=
else LOCATION_CONSTRAINT="--create-bucket-configuration \
    LocationConstraint=${AWS_REGION}"
fi
aws s3api create-bucket \
    $LOCATION_CONSTRAINT \
    --acl private \
    --bucket ${DEV_FARM_BUCKET}
```

- Wenn Sie bereits einen privaten Amazon S3 S3-Bucket haben, können Sie ihn verwenden, indem Sie ihn *MY_BUCKET_NAME* durch den Namen Ihres Buckets ersetzen.

```
DEV_FARM_BUCKET=MY_BUCKET_NAME
```

2. Nachdem Sie Ihren Amazon S3 S3-Bucket erstellt oder ausgewählt haben, fügen Sie den Bucket-Namen hinzu, `~/.bashrc` um den Bucket für andere Terminalsitzungen verfügbar zu machen.

```
echo "DEV_FARM_BUCKET=$DEV_FARM_BUCKET" >> ~/.bashrc
 source ~/.bashrc
```

3. Erstellen Sie eine AWS Identity and Access Management (IAM-) Rolle für die Warteschlange.

```
aws iam create-role --role-name "${DEV_FARM_NAME}QueueRole" \
    --assume-role-policy-document \
        '{
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "credentials.deadline.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        }'
```

```
aws iam put-role-policy \
    --role-name "${DEV_FARM_NAME}QueueRole" \
    --policy-name S3BucketsAccess \
    --policy-document \
            '{
                "Version": "2012-10-17",
                "Statement": [
                {
                    "Action": [
                        "s3:GetObject*",
                        "s3:GetBucket*",
                        "s3:List*",
                        "s3:DeleteObject*",
                        "s3:PutObject",
                        "s3:PutObjectLegalHold",
                        "s3:PutObjectRetention",
                        "s3:PutObjectTagging",
                        "s3:PutObjectVersionTagging",
                        "s3:Abort*"
                    ],
                    "Resource": [
                        "arn:aws:s3:::'$DEV_FARM_BUCKET'",
                        "arn:aws:s3:::'$DEV_FARM_BUCKET'/*"
                    ],
                    "Effect": "Allow"
                }
            ]
            }'
```

4. Aktualisieren Sie Ihre Warteschlange so, dass sie die Einstellungen für Jobanhänge und die IAM-Rolle enthält.

```
QUEUE_ROLE_ARN="arn:aws:iam::$(aws sts get-caller-identity \
        --query "Account" --output text):role/${DEV_FARM_NAME}QueueRole"
aws deadline update-queue \
    --farm-id $DEV_FARM_ID \
    --queue-id $DEV_QUEUE_ID \
    --role-arn $QUEUE_ROLE_ARN \
    --job-attachment-settings \
        '{
            "s3BucketName": "'$DEV_FARM_BUCKET'",
            "rootPrefix": "JobAttachments"
        }'
```

5. Vergewissern Sie sich, dass Sie Ihre Warteschlange aktualisiert haben.

```
deadline queue get
```

Es wird eine Ausgabe wie die folgende angezeigt:

```
...
jobAttachmentSettings:
  s3BucketName: DEV_FARM_BUCKET
  rootPrefix: JobAttachments
roleArn: arn:aws:iam::ACCOUNT_NUMBER:role/DeveloperFarmQueueRole
...
```

simple_file_jobMit Stellenanhängen einreichen

Wenn Sie Job-Anhänge verwenden, müssen Job-Bundles Deadline Cloud genügend Informationen liefern, um den Datenfluss des Jobs zu bestimmen, z. B. anhand von PATH Parametern. Im Fall von haben Sie die `template.yaml` Datei bearbeitetsimple_file_job, um Deadline Cloud mitzuteilen, dass sich der Datenfluss in der Eingabe- und Ausgabedatei befindet.

Nachdem Sie die Konfiguration für Jobanhänge zu Ihrer Warteschlange hinzugefügt haben, können Sie das simple_file_job-Beispiel mit Jobanhängen einreichen. Nachdem Sie dies getan haben, können Sie sich die Protokollierung und die Jobausgabe ansehen, um sich zu vergewissern, dass das simple_file_job mit den Stellenanhängen funktioniert.

Um das simple_file_job-Job-Paket mit Job-Anhängen einzureichen

1. Wählen Sie Ihre erste CloudShell Registerkarte und öffnen Sie dann das Verzeichnis. `JobBundle-Samples`
2.
```
cd ~/deadline-cloud-samples/job_bundles/
```
3. Reichen Sie simple_file_job in die Warteschlange ein. Wenn Sie aufgefordert werden, den Upload zu bestätigen, geben Sie ein. **y**

```
deadline bundle submit simple_file_job \
    -p InFile=simple_job/template.yaml \
    -p OutFile=hash-jobattachments.txt
```

4. Führen Sie den folgenden Befehl aus, um die Ausgabe des Sitzungsprotokolls der Jobanhänge zur Datenübertragung anzuzeigen.

```
JOB_ID=$(deadline config get defaults.job_id)
SESSION_ID=$(aws deadline list-sessions \
        --farm-id $DEV_FARM_ID \
        --queue-id $DEV_QUEUE_ID \
        --job-id $JOB_ID \
        --query "sessions[0].sessionId" \
        --output text)
cat ~/demoenv-logs/$DEV_QUEUE_ID/$SESSION_ID.log
```

5. Listet die Sitzungsaktionen auf, die innerhalb der Sitzung ausgeführt wurden.

```
aws deadline list-session-actions \
    --farm-id $DEV_FARM_ID \
    --queue-id $DEV_QUEUE_ID \
    --job-id $JOB_ID \
    --session-id $SESSION_ID
```

Es werden Ausgaben wie die folgende angezeigt:

```
{
    "sessionactions": [
        {
            "sessionActionId": "sessionaction-123-0",
            "status": "SUCCEEDED",
            "startedAt": "<timestamp>",
            "endedAt": "<timestamp>",
            "progressPercent": 100.0,
            "definition": {
                "syncInputJobAttachments": {}
            }
        },
        {
            "sessionActionId": "sessionaction-123-1",
            "status": "SUCCEEDED",
            "startedAt": "<timestamp>",
            "endedAt": "<timestamp>",
            "progressPercent": 100.0,
            "definition": {
                "taskRun": {
```

```
                    "taskId": "task-abc-0",
                    "stepId": "step-def"
                }
            }
        }
    ]
}
```

Bei der ersten Sitzungsaktion wurden die Eingabeauftragsanhänge heruntergeladen, während die zweite Aktion die Aufgabe wie in den vorherigen Schritten ausführt und dann die Anlagen für den Ausgabeauftrag hochgeladen hat.

6. Listet das Ausgabeverzeichnis auf.

```
ls *.txt
```

Die Ausgabe `hash.txt` ist beispielsweise im Verzeichnis vorhanden, aber `hash-jobattachments.txt` nicht vorhanden, da die Ausgabedatei des Jobs noch nicht heruntergeladen wurde.

7. Laden Sie die Ausgabe des letzten Jobs herunter.

```
deadline job download-output
```

8. Sehen Sie sich die Ausgabe der heruntergeladenen Datei an.

```
cat hash-jobattachments.txt
```

Es wird eine Ausgabe wie die folgende angezeigt:

```
eaa2df5d34b54be5ac34c56a24a8c237b8487231a607eaf530a04d76b89c9cd3  /tmp/openjd/
session-123/assetroot-abc/simple_job/template.yaml
```

Verstehen, wie Jobanhänge in Amazon S3 gespeichert werden

Sie können die AWS Command Line Interface (AWS CLI) verwenden, um Daten für Jobanhänge hoch- oder herunterzuladen, die in Amazon S3 S3-Buckets gespeichert sind. Wenn Sie wissen, wie Deadline Cloud Jobanhänge auf Amazon S3 speichert, hilft Ihnen das bei der Entwicklung von Workloads und Pipeline-Integrationen.

Um zu überprüfen, wie Deadline Cloud-Jobanhänge in Amazon S3 gespeichert werden

1. Wählen Sie Ihren ersten CloudShell Tab und öffnen Sie dann das Verzeichnis mit den Job-Bundle-Beispielen.

   ```
   cd ~/deadline-cloud-samples/job_bundles/
   ```

2. Untersuchen Sie die Jobeigenschaften.

   ```
   deadline job get
   ```

   Es wird eine Ausgabe wie die folgende angezeigt:

   ```
   parameters:
     Message:
       string: Welcome to AWS Deadline Cloud!
     InFile:
       path: /home/cloudshell-user/deadline-cloud-samples/job_bundles/simple_job/
   template.yaml
     OutFile:
       path: /home/cloudshell-user/deadline-cloud-samples/job_bundles/hash-
   jobattachments.txt
   attachments:
     manifests:
     - rootPath: /home/cloudshell-user/deadline-cloud-samples/job_bundles/
       rootPathFormat: posix
       outputRelativeDirectories:
       - .
       inputManifestPath: farm-3040c59a5b9943d58052c29d907a645d/queue-
   cde9977c9f4d4018a1d85f3e6c1a4e6e/Inputs/
   f46af01ca8904cd8b514586671c79303/0d69cd94523ba617c731f29c019d16e8_input.xxh128
       inputManifestHash: f95ef91b5dab1fc1341b75637fe987ee
     fileSystem: COPIED
   ```

   Das Feld „Anlagen“ enthält eine Liste von Manifeststrukturen, die Eingabe- und Ausgabedatenpfade beschreiben, die der Job bei seiner Ausführung verwendet. Sehen Sie `rootPath` sich den lokalen Verzeichnispfad auf dem Computer an, der den Job eingereicht hat. Um das Amazon S3 S3-Objektsuffix zu sehen, das eine Manifestdatei enthält, lesen Sie den`inputManifestFile`. Die Manifestdatei enthält Metadaten für einen Verzeichnisstruktur-Snapshot der Eingabedaten des Jobs.

3. Drucken Sie das Amazon S3 S3-Manifest-Objekt hübsch aus, um die Eingabeverzeichnisstruktur für den Job zu sehen.

```
MANIFEST_SUFFIX=$(aws deadline get-job \
     --farm-id $DEV_FARM_ID \
     --queue-id $DEV_QUEUE_ID \
     --job-id $JOB_ID \
     --query "attachments.manifests[0].inputManifestPath" \
     --output text)
 aws s3 cp s3://$DEV_FARM_BUCKET/JobAttachments/Manifests/$MANIFEST_SUFFIX - | jq .
```

Es wird eine Ausgabe wie die folgende angezeigt:

```
{
     "hashAlg": "xxh128",
     "manifestVersion": "2023-03-03",
     "paths": [
     {
         "hash": "2ec297b04c59c4741ed97ac8fb83080c",
         "mtime": 1698186190000000,
         "path": "simple_job/template.yaml",
         "size": 445
     }
     ],
     "totalSize": 445
 }
```

4. Konstruieren Sie das Amazon S3 S3-Präfix, das die Manifeste für die Anlagen des Ausgabeauftrags enthält, und listen Sie das Objekt darunter auf.

```
SESSION_ACTION=$(aws deadline list-session-actions \
    --farm-id $DEV_FARM_ID \
    --queue-id $DEV_QUEUE_ID \
    --job-id $JOB_ID \
    --session-id $SESSION_ID \
    --query "sessionActions[?definition.taskRun != null] | [0]")
STEP_ID=$(echo $SESSION_ACTION | jq -r .definition.taskRun.stepId)
TASK_ID=$(echo $SESSION_ACTION | jq -r .definition.taskRun.taskId)
TASK_OUTPUT_PREFIX=JobAttachments/Manifests/$DEV_FARM_ID/$DEV_QUEUE_ID/$JOB_ID/
$STEP_ID/$TASK_ID/
aws s3api list-objects-v2 --bucket $DEV_FARM_BUCKET --prefix $TASK_OUTPUT_PREFIX
```

Die angehängten Ausgabeaufträge werden nicht direkt von der Jobressource referenziert, sondern stattdessen in einem Amazon S3 S3-Bucket platziert, der auf der Farmressource basiert IDs.

5. Rufen Sie den neuesten Manifestobjektschlüssel für die spezifische Sitzungsaktions-ID ab und drucken Sie dann die Manifestobjekte hübsch aus.

```
SESSION_ACTION_ID=$(echo $SESSION_ACTION | jq -r .sessionActionId)
 MANIFEST_KEY=$(aws s3api list-objects-v2 \
     --bucket $DEV_FARM_BUCKET \
     --prefix $TASK_OUTPUT_PREFIX \
     --query "Contents[*].Key" --output text \
     | grep $SESSION_ACTION_ID \
     | sort | tail -1)
 MANIFEST_OBJECT=$(aws s3 cp s3://$DEV_FARM_BUCKET/$MANIFEST_KEY -)
 echo $MANIFEST_OBJECT | jq .
```

In der Ausgabe werden Eigenschaften der Datei `hash-jobattachments.txt` wie die folgenden angezeigt:

```
{
     "hashAlg": "xxh128",
     "manifestVersion": "2023-03-03",
     "paths": [
     {
         "hash": "f60b8e7d0fabf7214ba0b6822e82e08b",
         "mtime": 1698785252554950,
         "path": "hash-jobattachments.txt",
         "size": 182
     }
     ],
     "totalSize": 182
 }
```

Ihr Job wird nur ein einziges Manifest-Objekt pro Aufgabenausführung haben, aber im Allgemeinen ist es möglich, mehr Objekte pro Aufgabenausführung zu haben.

6. Zeigen Sie die für den Inhalt adressierbare Amazon S3 S3-Speicherausgabe unter dem Präfix an. `Data`

```
FILE_HASH=$(echo $MANIFEST_OBJECT | jq -r .paths[0].hash)
FILE_PATH=$(echo $MANIFEST_OBJECT | jq -r .paths[0].path)
aws s3 cp s3://$DEV_FARM_BUCKET/JobAttachments/Data/$FILE_HASH -
```

Es wird eine Ausgabe wie die folgende angezeigt:

```
eaa2df5d34b54be5ac34c56a24a8c237b8487231a607eaf530a04d76b89c9cd3  /tmp/openjd/
session-123/assetroot-abc/simple_job/template.yaml
```

Nächste Schritte

Nachdem Sie gelernt haben, wie Sie Jobs mit Anhängen mithilfe der Deadline Cloud-CLI einreichen, können Sie Folgendes erkunden:

- Mit Deadline Cloud einreichenum zu erfahren, wie Sie Jobs mit einem OpenJD-Bundle auf Ihren Worker-Hosts ausführen.
- Fügen Sie Ihrer Entwicklerfarm in Deadline Cloud eine vom Service verwaltete Flotte hinzuum Ihre Jobs auf Hosts auszuführen, die von Deadline Cloud verwaltet werden.
- Bereinigen Sie Ihre Farmressourcen in Deadline Cloudum die Ressourcen herunterzufahren, die Sie für dieses Tutorial verwendet haben.

Fügen Sie Ihrer Entwicklerfarm in Deadline Cloud eine vom Service verwaltete Flotte hinzu

AWS CloudShell bietet nicht genügend Rechenkapazität, um größere Workloads zu testen. Es ist auch nicht für Jobs konfiguriert, die Aufgaben auf mehrere Worker-Hosts verteilen.

Anstatt zu verwenden CloudShell, können Sie Ihrer Entwicklerfarm eine Auto Scaling Service Managed Fleet (SMF) hinzufügen. Ein SMF bietet ausreichend Rechenkapazität für größere Workloads und kann Jobs bewältigen, bei denen Jobaufgaben auf mehrere Worker-Hosts verteilt werden müssen.

Bevor Sie ein SMF hinzufügen, müssen Sie eine Deadline Cloud-Farm, eine Warteschlange und eine Flotte einrichten. Siehe Erstellen Sie eine Deadline Cloud-Farm.

Um Ihrer Entwicklerfarm eine vom Service verwaltete Flotte hinzuzufügen

1. Wählen Sie Ihre erste AWS CloudShell Registerkarte aus, erstellen Sie dann die vom Service verwaltete Flotte und fügen Sie deren Flotten-ID hinzu. `.bashrc` Diese Aktion macht es für andere Terminalsitzungen verfügbar.

```
FLEET_ROLE_ARN="arn:aws:iam::$(aws sts get-caller-identity \
        --query "Account" --output text):role/${DEV_FARM_NAME}FleetRole"
 aws deadline create-fleet \
     --farm-id $DEV_FARM_ID \
     --display-name "$DEV_FARM_NAME SMF" \
     --role-arn $FLEET_ROLE_ARN \
     --max-worker-count 5 \
     --configuration \
         '{
             "serviceManagedEc2": {
                 "instanceCapabilities": {
                     "vCpuCount": {
                         "min": 2,
                         "max": 4
                     },
                     "memoryMiB": {
                         "min": 512
                     },
                     "osFamily": "linux",
                     "cpuArchitectureType": "x86_64"
                 },
                 "instanceMarketOptions": {
                     "type": "spot"
                 }
             }
         }'

 echo "DEV_SMF_ID=$(aws deadline list-fleets \
         --farm-id $DEV_FARM_ID \
         --query "fleets[?displayName=='$DEV_FARM_NAME SMF'].fleetId \
         | [0]" --output text)" >> ~/.bashrc
 source ~/.bashrc
```

2. Ordnen Sie das SMF Ihrer Warteschlange zu.

```
aws deadline create-queue-fleet-association \
     --farm-id $DEV_FARM_ID \
```

```
    --queue-id $DEV_QUEUE_ID \
    --fleet-id $DEV_SMF_ID
```

3. In simple_file_job die Warteschlange einreichen. Wenn Sie aufgefordert werden, den Upload zu bestätigen, geben Sie ein**y**.

```
deadline bundle submit simple_file_job \
    -p InFile=simple_job/template.yaml \
    -p OutFile=hash-jobattachments.txt
```

4. Vergewissern Sie sich, dass das SMF ordnungsgemäß funktioniert.

```
deadline fleet get
```

   - Es kann einige Minuten dauern, bis der Mitarbeiter anfängt. Wiederholen Sie den `deadline fleet get` Befehl, bis Sie sehen, dass die Flotte läuft.
   - Die `queueFleetAssociationsStatus` für den Service verwaltete Flotte wird es sein. `ACTIVE`
   - Die SMF `autoScalingStatus` wird von `GROWING` zu wechseln. `STEADY`

   Ihr Status wird in etwa wie folgt aussehen:

```
fleetId: fleet-2cc78e0dd3f04d1db427e7dc1d51ea44
farmId: farm-63ee8d77cdab4a578b685be8c5561c4a
displayName: DeveloperFarm SMF
description: ''
status: ACTIVE
autoScalingStatus: STEADY
targetWorkerCount: 0
workerCount: 0
minWorkerCount: 0
maxWorkerCount: 5
```

5. Sehen Sie sich das Protokoll für den Job an, den Sie eingereicht haben. Dieses Protokoll wird in einem Protokoll in Amazon CloudWatch Logs gespeichert, nicht im CloudShell Dateisystem.

```
JOB_ID=$(deadline config get defaults.job_id)
SESSION_ID=$(aws deadline list-sessions \
        --farm-id $DEV_FARM_ID \
        --queue-id $DEV_QUEUE_ID \
```

```
        --job-id $JOB_ID \
        --query "sessions[0].sessionId" \
        --output text)
aws logs tail /aws/deadline/$DEV_FARM_ID/$DEV_QUEUE_ID \
    --log-stream-names $SESSION_ID
```

Nächste Schritte

Nachdem Sie eine Flotte mit Servicemanagement erstellt und getestet haben, sollten Sie die Ressourcen, die Sie erstellt haben, entfernen, um unnötige Kosten zu vermeiden.

- Bereinigen Sie Ihre Farmressourcen in Deadline Cloudum die Ressourcen herunterzufahren, die Sie für dieses Tutorial verwendet haben.

Bereinigen Sie Ihre Farmressourcen in Deadline Cloud

Um neue Workloads und Pipeline-Integrationen zu entwickeln und zu testen, können Sie weiterhin die Deadline Cloud-Entwicklerfarm verwenden, die Sie für dieses Tutorial erstellt haben. Wenn Sie Ihre Entwicklerfarm nicht mehr benötigen, können Sie ihre Ressourcen, einschließlich Farm-, Flotten-, Warteschlangen-, AWS Identity and Access Management (IAM-) Rollen und Logs, in Amazon CloudWatch Logs löschen. Nachdem Sie diese Ressourcen gelöscht haben, müssen Sie das Tutorial erneut beginnen, um die Ressourcen verwenden zu können. Weitere Informationen finden Sie unter Erste Schritte mit Deadline Cloud-Ressourcen.

Um die Ressourcen der Entwicklerfarm zu bereinigen

1. Wählen Sie Ihren ersten CloudShell Tab und beenden Sie dann alle Verbindungen zwischen Warteschlangen und Flotten für Ihre Warteschlange.

```
FLEETS=$(aws deadline list-queue-fleet-associations \
        --farm-id $DEV_FARM_ID \
        --queue-id $DEV_QUEUE_ID \
        --query "queueFleetAssociations[].fleetId" \
        --output text)
for FLEET_ID in $FLEETS; do
    aws deadline update-queue-fleet-association \
        --farm-id $DEV_FARM_ID \
        --queue-id $DEV_QUEUE_ID \
        --fleet-id $FLEET_ID \
```

```
        --status STOP_SCHEDULING_AND_CANCEL_TASKS
done
```

2. Listet die Flottenzuordnungen der Warteschlange auf.

```
aws deadline list-queue-fleet-associations \
     --farm-id $DEV_FARM_ID \
     --queue-id $DEV_QUEUE_ID
```

Möglicherweise müssen Sie den Befehl erneut ausführen, bis die Ausgabe meldet`"status": "STOPPED"`, dann können Sie mit dem nächsten Schritt fortfahren. Dieser Vorgang kann mehrere Minuten in Anspruch nehmen.

```
{
    "queueFleetAssociations": [
        {
            "queueId": "queue-abcdefgh01234567890123456789012id",
            "fleetId": "fleet-abcdefgh01234567890123456789012id",
            "status": "STOPPED",
            "createdAt": "2023-11-21T20:49:19+00:00",
            "createdBy": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed/
MySessionName",
            "updatedAt": "2023-11-21T20:49:38+00:00",
            "updatedBy": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed/
MySessionName"
        },
        {
            "queueId": "queue-abcdefgh01234567890123456789012id",
            "fleetId": "fleet-abcdefgh01234567890123456789012id",
            "status": "STOPPED",
            "createdAt": "2023-11-21T20:32:06+00:00",
            "createdBy": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed/
MySessionName",
            "updatedAt": "2023-11-21T20:49:39+00:00",
            "updatedBy": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed/
MySessionName"
        }
    ]
}
```

3. Löschen Sie alle Verknüpfungen zwischen Warteschlange und Flotte für Ihre Warteschlange.

```
for FLEET_ID in $FLEETS; do
     aws deadline delete-queue-fleet-association \
         --farm-id $DEV_FARM_ID \
         --queue-id $DEV_QUEUE_ID \
         --fleet-id $FLEET_ID
 done
```

4. Löschen Sie alle Flotten, die Ihrer Warteschlange zugeordnet sind.

```
for FLEET_ID in $FLEETS; do
     aws deadline delete-fleet \
         --farm-id $DEV_FARM_ID \
         --fleet-id $FLEET_ID
 done
```

5. Löscht die Warteschlange.

```
aws deadline delete-queue \
     --farm-id $DEV_FARM_ID \
     --queue-id $DEV_QUEUE_ID
```

6. Löschen Sie die Farm.

```
aws deadline delete-farm \
     --farm-id $DEV_FARM_ID
```

7. Löschen Sie andere AWS Ressourcen für Ihre Farm.

   a. Löschen Sie die Flottenrolle AWS Identity and Access Management (IAM).

```
aws iam delete-role-policy \
     --role-name "${DEV_FARM_NAME}FleetRole" \
     --policy-name WorkerPermissions
aws iam delete-role \
     --role-name "${DEV_FARM_NAME}FleetRole"
```

   b. Löschen Sie die Warteschlangen-IAM-Rolle.

```
aws iam delete-role-policy \
     --role-name "${DEV_FARM_NAME}QueueRole" \
     --policy-name S3BucketsAccess
aws iam delete-role \
```

```
     --role-name "${DEV_FARM_NAME}QueueRole"
```

c. Löschen Sie die Amazon CloudWatch Logs-Protokollgruppen. Jede Warteschlange und Flotte hat ihre eigene Protokollgruppe.

```
aws logs delete-log-group \
     --log-group-name "/aws/deadline/$DEV_FARM_ID/$DEV_QUEUE_ID"
aws logs delete-log-group \
     --log-group-name "/aws/deadline/$DEV_FARM_ID/$DEV_CMF_ID"
aws logs delete-log-group \
     --log-group-name "/aws/deadline/$DEV_FARM_ID/$DEV_SMF_ID"
```

Jobs erstellen, um sie an Deadline Cloud einzureichen

Sie reichen Jobs mithilfe von Jobpaketen an Deadline Cloud ein. Ein Job-Bundle ist eine Sammlung von Dateien, einschließlich einer Jobvorlage mit einer Open Job Description (OpenJD) und allen Asset-Dateien, die zum Rendern des Jobs benötigt werden.

Die Jobvorlage beschreibt, wie Worker die Assets verarbeiten und darauf zugreifen, und stellt das Skript bereit, das der Worker ausführt. Job-Pakete ermöglichen es Künstlern, technischen Direktoren und Pipeline-Entwicklern, komplexe Jobs einfach von ihren lokalen Workstations oder der lokalen Renderfarm aus an Deadline Cloud zu senden. Jobpakete sind besonders nützlich für Teams, die an groß angelegten Projekten mit visuellen Effekten, Animationen oder anderen Medienrendering-Projekten arbeiten, die skalierbare Rechenressourcen auf Abruf benötigen.

Sie können das Auftragspaket mithilfe des lokalen Dateisystems zum Speichern von Dateien und eines Texteditors zum Erstellen der Auftragsvorlage erstellen. Nachdem Sie das Paket erstellt haben, reichen Sie den Job entweder mit der Deadline Cloud-CLI oder einem Tool wie einem Deadline Cloud-Submitter an Deadline Cloud ein

Sie können Ihre Ressourcen in einem Dateisystem speichern, das von Ihren Mitarbeitern gemeinsam genutzt wird, oder Sie können Deadline Cloud-Jobanhänge verwenden, um das Verschieben von Assets in S3-Buckets zu automatisieren, wo Ihre Mitarbeiter darauf zugreifen können. Jobanhänge helfen auch dabei, die Ergebnisse Ihrer Jobs zurück auf Ihre Workstations zu übertragen.

Die folgenden Abschnitte enthalten detaillierte Anweisungen zum Erstellen und Einreichen von Job-Bundles an Deadline Cloud.

Themen

Vorlagen für offene Stellenbeschreibungen (OpenJD) für Deadline Cloud

Ein Job-Bundle ist eines der Tools, mit denen Sie Jobs für AWS Deadline Cloud definieren. Sie gruppieren eine OpenJD-Vorlage (Open Job Description) mit zusätzlichen Informationen wie Dateien und Verzeichnissen, die Ihre Jobs mit Stellenanhängen verwenden. Sie verwenden die Deadline Cloud-Befehlszeilenschnittstelle (CLI), um ein Job-Bundle zu verwenden, um Jobs für die Ausführung in einer Warteschlange einzureichen.

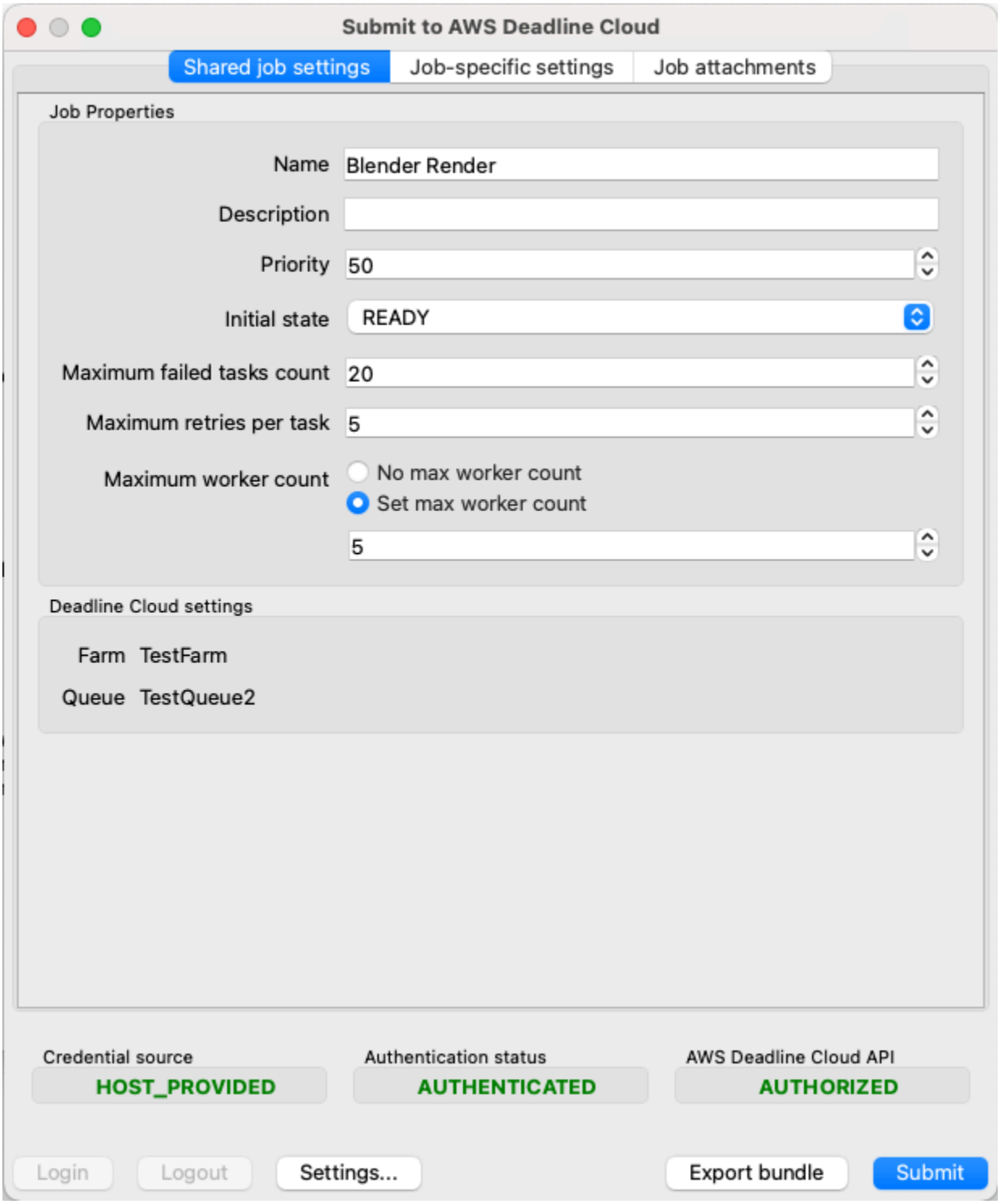
Ein Job-Bundle ist eine Verzeichnisstruktur, die eine OpenJD-Jobvorlage, andere Dateien, die den Job definieren, und jobspezifische Dateien enthält, die als Eingabe für Ihren Job benötigt werden. Sie können die Dateien, die Ihren Job definieren, entweder als YAML- oder JSON-Dateien angeben.

Die einzige erforderliche Datei ist entweder `template.yaml` oder`template.json`. Sie können auch die folgenden Dateien einschließen:

```
/template.yaml (or template.json)
/asset_references.yaml (or asset_references.json)
/parameter_values.yaml (or parameter_values.json)
/other job-specific files and directories
```

Verwenden Sie ein Job-Bundle für benutzerdefinierte Job-Übermittlungen mit der Deadline Cloud-CLI und einem Jobanhang, oder Sie können eine grafische Einreichungsoberfläche verwenden. Im Folgenden finden Sie beispielsweise das Blender-Beispiel von GitHub. Um das Beispiel mit dem folgenden Befehl im Blender-Beispielverzeichnis auszuführen:

```
deadline bundle gui-submit blender_render
```



Das Fenster mit den auftragsspezifischen Einstellungen wird aus den `userInterface` Eigenschaften der Jobparameter generiert, die in der Jobvorlage definiert sind.

Um einen Job über die Befehlszeile einzureichen, können Sie einen Befehl verwenden, der dem folgenden ähnelt

```
deadline bundle submit \
    --yes \
    --name Demo \
     -p BlenderSceneFile=location of scene file \
     -p OutputDir=file pathe for job output \
      blender_render/
```

Oder Sie können die `deadline.client.api.create_job_from_job_bundle` Funktion im `deadline` Python-Paket verwenden.

Alle in Deadline Cloud enthaltenen Plug-ins für Job-Einreicher, wie das Autodesk Maya-Plugin, generieren ein Job-Bundle für Ihre Einreichung und verwenden dann das Deadline Cloud-Python-Paket, um Ihren Job bei Deadline Cloud einzureichen. Sie können die eingereichten Job-Bundles im Job-Verlaufsverzeichnis Ihrer Workstation oder mithilfe eines Absenders einsehen. Sie können Ihr Job-Verlaufsverzeichnis mit dem folgenden Befehl finden:

```
deadline config get settings.job_history_dir
```

Wenn Ihr Job auf einem Deadline Cloud-Worker ausgeführt wird, hat er Zugriff auf Umgebungsvariablen, die ihm Informationen über den Job liefern. Die Umgebungsvariablen sind:

Variablenname	Available (Verfügbar)
DEADLINE_FARM_ID	Alle Aktionen
DEADLINE_FLOTTENNUMMER	Alle Aktionen
DEADLINE_WORKER-ID	Alle Aktionen
DEADLINE_WARTESCHLANGEN-ID	Alle Aktionen
DEADLINE_JOB-ID	Alle Aktionen
DEADLINE_STEP-ID	Aktionen der Aufgabe
DEADLINE_SESSION_ID	Alle Aktionen
DEADLINE_TASK_ID	Aktionen der Aufgabe
DEADLINE_SESSIONACTION_ID	Alle Aktionen

Themen

Jobvorlagenelemente für Jobpakete

Die Jobvorlage definiert die Laufzeitumgebung und die Prozesse, die als Teil eines Deadline Cloud-Jobs ausgeführt werden. Sie können Parameter in einer Vorlage erstellen, sodass sie verwendet werden kann, um Jobs zu erstellen, die sich nur in den Eingabewerten unterscheiden, ähnlich wie bei einer Funktion in einer Programmiersprache.

Wenn Sie einen Job an Deadline Cloud senden, wird er in allen Warteschlangenumgebungen ausgeführt, die auf die Warteschlange angewendet wurden. Warteschlangenumgebungen werden mithilfe der Spezifikation für externe Umgebungen von Open Job Description (OpenJD) erstellt. Einzelheiten finden Sie in der Environment-Vorlage im OpenJD-Repository GitHub .

Eine Einführung in die Erstellung eines Jobs mit einer OpenJD-Jobvorlage finden Sie unter Einführung in die Erstellung eines Jobs im GitHub OpenJD-Repository. Zusätzliche Informationen finden Sie unter Wie Jobs ausgeführt werden. Es gibt Beispiele für Jobvorlagen im `samples` Verzeichnis des GitHub OpenJD-Repositorys.

Sie können die Jobvorlage entweder im YAML-Format (`template.yaml`) oder im JSON-Format (`template.json`) definieren. Die Beispiele in diesem Abschnitt werden im YAML-Format angezeigt.

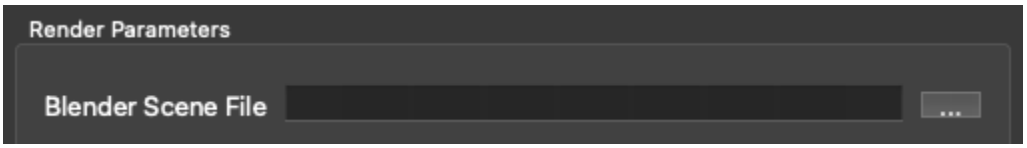
Die Jobvorlage für das `blender_render` Beispiel definiert beispielsweise einen Eingabeparameter `BlenderSceneFile` als Dateipfad:

```
- name: BlenderSceneFile
  type: PATH
  objectType: FILE
  dataFlow: IN
  userInterface:
    control: CHOOSE_INPUT_FILE
    label: Blender Scene File
    groupLabel: Render Parameters
    fileFilters:
    - label: Blender Scene Files
```

```
      patterns: ["*.blend"]
    - label: All Files
      patterns: ["*"]
  description: >
    Choose the Blender scene file to render. Use the 'Job Attachments' tab
    to add textures and other files that the job needs.
```

Die `userInterface` Eigenschaft definiert das Verhalten automatisch generierter Benutzeroberflächen sowohl in der Befehlszeile, die den Befehl verwendet, als auch in den Plug-ins für die Auftragsübermittlung für Anwendungen wie Autodesk Maya. `deadline bundle gui-submit`

In diesem Beispiel ist das UI-Widget zur Eingabe eines Werts für den `BlenderSceneFile` Parameter ein Dialogfeld zur Dateiauswahl, in dem nur Dateien angezeigt werden. `.blend`



Weitere Beispiele für die Verwendung des `userInteface` Elements finden Sie im Beispiel gui_control_showcase im Repository unter. deadline-cloud-samples GitHub

Die `dataFlow` Eigenschaften `objectType` und steuern das Verhalten von Job-Anhängen, wenn Sie einen Job aus einem Job-Bundle einreichen. In diesem Fall `dataFlow:IN` bedeuten `objectType: FILE` und, dass der Wert von eine Eingabedatei für Jobanhänge `BlenderSceneFile` ist.

Im Gegensatz dazu hat die Definition des `OutputDir` Parameters `objectType: DIRECTORY` und`dataFlow: OUT`:

```
- name: OutputDir
  type: PATH
  objectType: DIRECTORY
  dataFlow: OUT
  userInterface:
    control: CHOOSE_DIRECTORY
    label: Output Directory
    groupLabel: Render Parameters
  default: "./output"
  description: Choose the render output directory.
```

Der Wert des `OutputDir` Parameters wird von Job-Anhängen als Verzeichnis verwendet, in das der Job Ausgabedateien schreibt.

Weitere Informationen zu den `dataFlow` Eigenschaften `objectType` und finden Sie JobPathParameterDefinitionin der Spezifikation Open Job Description

Der Rest des Beispiels für eine `blender_render` Jobvorlage definiert den Arbeitsablauf des Jobs als einen einzelnen Schritt, wobei jedes Bild in der Animation als separate Aufgabe gerendert wird:

```
steps:
- name: RenderBlender
  parameterSpace:
    taskParameterDefinitions:
    - name: Frame
      type: INT
      range: "{{Param.Frames}}"
  script:
    actions:
      onRun:
        command: bash
        # Note: {{Task.File.Run}} is a variable that expands to the filename on the
 worker host's
        # disk where the contents of the 'Run' embedded file, below, is written.
        args: ['{{Task.File.Run}}']
    embeddedFiles:
      - name: Run
        type: TEXT
        data: |
          # Configure the task to fail if any individual command fails.
          set -xeuo pipefail

          mkdir -p '{{Param.OutputDir}}'

          blender --background '{{Param.BlenderSceneFile}}' \
                  --render-output '{{Param.OutputDir}}/{{Param.OutputPattern}}' \
                  --render-format {{Param.Format}} \
                  --use-extension 1 \
                  --render-frame {{Task.Param.Frame}}
```

Wenn der Wert des `Frames` Parameters beispielsweise lautet`1-10`, definiert er 10 Aufgaben. Jede Aufgabe hat einen anderen Wert für den `Frame` Parameter. Um eine Aufgabe auszuführen:

1. Alle Variablenverweise in der `data` Eigenschaft der eingebetteten Datei sind beispielsweise erweitert`--render-frame 1`.

2. Der Inhalt der `data` Eigenschaft wird in eine Datei im Arbeitsverzeichnis der Sitzung auf der Festplatte geschrieben.
3. Der onRun Befehl der Aufgabe wird in aufgelöst `bash` *`location of embedded file`* und dann ausgeführt.

Weitere Informationen zu eingebetteten Dateien, Sitzungen und Pfadzuordnungen finden Sie unter So werden Jobs ausgeführt in der Open Job Description-Spezifikation.

Es gibt weitere Beispiele für Jobvorlagen im Repository deadline-cloud-samples/job_bundles sowie die Vorlagenbeispiele, die in der Open Job Description-Spezifikation enthalten sind.

Aufgabenteilung für Jobvorlagen

Durch Aufgaben-Chunking können Sie mehrere Aufgaben zu einer einzigen Arbeitseinheit zusammenfassen, die als Chunk bezeichnet wird. In einem Renderjob bedeutet dies beispielsweise, dass Deadline Cloud mehrere Frames zusammen versenden kann, anstatt einen Frame pro Befehlsaufruf. Dies reduziert den Aufwand beim Starten von Anwendungen für jede Aufgabe und verkürzt die Gesamtlaufzeit des Jobs. Einzelheiten finden Sie unter Mehrere Frames gleichzeitig ausführen im OpenJD-Wiki.

OpenJD unterstützt Erweiterungen, die Jobvorlagen um optionale Funktionen erweitern. Das Aufteilen von Aufgaben wird durch Hinzufügen der Erweiterung aktiviert. `TASK_CHUNKING` Um Chunking zu verwenden, fügen Sie die Erweiterung zu Ihrer Jobvorlage hinzu und verwenden Sie den `CHUNK[INT]` Task-Parametertyp. Senden Sie Jobs in Einzelteilen mit demselben Befehl. `deadline bundle submit` Die folgende Jobvorlage rendert beispielsweise Frames in Blöcken von 10:

```
specificationVersion: 'jobtemplate-2023-09'
extensions:
  - TASK_CHUNKING
name: Blender Render with Contiguous Chunking
parameterDefinitions:
  - name: BlenderSceneFile
    type: PATH
    objectType: FILE
    dataFlow: IN
  - name: Frames
    type: STRING
    default: "1-100"
```

```
  - name: OutputDir
    type: PATH
    objectType: DIRECTORY
    dataFlow: OUT
    default: "./output"
steps:
  - name: RenderBlender
    parameterSpace:
      taskParameterDefinitions:
        - name: Frame
          type: CHUNK[INT]
          range: "{{Param.Frames}}"
          chunks:
            defaultTaskCount: 10
            rangeConstraint: CONTIGUOUS
    script:
      actions:
        onRun:
          command: bash
          args: ["{{Task.File.Run}}"]
      embeddedFiles:
        - name: Run
          type: TEXT
          data: |
            set -xeuo pipefail

            mkdir -p '{{Param.OutputDir}}'

            # Parse the chunk range (e.g., "1-10") into start and end frames
            START_FRAME="$(echo '{{Task.Param.Frame}}' | cut -d- -f1)"
            END_FRAME="$(echo '{{Task.Param.Frame}}' | cut -d- -f2)"

            blender --background '{{Param.BlenderSceneFile}}' \
                    --render-output '{{Param.OutputDir}}/output_####' \
                    --render-format PNG \
                    --use-extension 1 \
                    -s "$START_FRAME" \
                    -e "$END_FRAME" \
                    --render-anim
```

In diesem Beispiel unterteilt Deadline Cloud die 100 Frames in Blöcke wie`1-10`, usw. `11-20` Die `{{Task.Param.Frame}}` Variable wird zu einem Bereichsausdruck wie erweitert. `1-10` Da auf gesetzt `rangeConstraint` ist`CONTIGUOUS`, ist der Bereich immer `start-end` formatiert. Das

Skript analysiert diesen Bereich und übergibt die Start- und Endframes mithilfe der `-e` Optionen `-s` und mit `--render-anim` an Blender.

Die `chunks` Eigenschaft unterstützt die folgenden Felder:

- `defaultTaskCount`— (Erforderlich) Wie viele Aufgaben zu einem einzigen Block zusammengefasst werden sollen. Der Höchstwert ist 150.
- `rangeConstraint`— (Erforderlich) Wenn`CONTIGUOUS`, ist ein Chunk immer ein zusammenhängender Bereich wie. `1-10` Wenn`NONCONTIGUOUS`, kann ein Chunk eine beliebige Menge sein wie. `1,3,7-10`
- `targetRuntimeSeconds`— (Optional) Die Ziellaufzeit in Sekunden für jeden Chunk. Deadline Cloud kann die Chunk-Größe dynamisch anpassen, um sich diesem Ziel zu nähern, sobald einige Chunks abgeschlossen sind.

Weitere Beispiele für das Chunking von Aufgaben, darunter einfache Beispiele und Blender-Beispiele mit zusammenhängenden und nicht zusammenhängenden Chunks, finden Sie in den Beispielen zum Aufgaben-Chunking im Deadline Cloud-Beispiel-Repository unter. GitHub

> **Vom Kunden verwaltete Flottenanforderungen**
>
> Für das Aufteilen von Aufgaben ist eine kompatible Worker-Agent-Version erforderlich. Wenn Sie vom Kunden verwaltete Flotten verwenden, stellen Sie sicher, dass Ihre Worker Agents auf dem neuesten Stand sind, bevor Sie Jobs mit Chunking weiterleiten. Vom Service verwaltete Flotten verwenden immer eine kompatible Worker-Agent-Version.

> **Die Ausgabe für Einzelaufträge wird heruntergeladen**
>
> Wenn Sie die Ausgabe für eine einzelne Aufgabe in einem aufgeteilten Job herunterladen, lädt Deadline Cloud die Ausgabe für den gesamten Block herunter. Wenn beispielsweise die Frames 1—10 zusammen verarbeitet wurden, umfasst das Herunterladen der Ausgabe für Frame 3 alle Frames 1—10. Für diese Funktion ist `deadline-cloud` Version 0.53.3 oder höher erforderlich.

Parameterwerte, Elemente für Job-Bundles

Sie können die Parameterdatei verwenden, um die Werte einiger Jobparameter in der Jobvorlage oder der CreateJobOperationsanforderungsargumente im Job-Bundle festzulegen, sodass Sie beim Senden eines Jobs keine Werte festlegen müssen. Die Benutzeroberfläche für die Auftragsübermittlung ermöglicht es Ihnen, diese Werte zu ändern.

Sie können die Jobvorlage entweder im YAML-Format (`parameter_values.yaml`) oder im JSON-Format (`parameter_values.json`) definieren. Die Beispiele in diesem Abschnitt werden im YAML-Format angezeigt.

In YAML ist das Format der Datei:

```
parameterValues:
- name: <string>
  value: <integer>, <float>, or <string>
- name: <string>
  value: <integer>, <float>, or <string>ab
... repeating as necessary
```

Jedes Element der `parameterValues` Liste muss eines der folgenden sein:

- Ein in der Jobvorlage definierter Jobparameter.
- Ein in einer Warteschlangenumgebung definierter Jobparameter für die Warteschlange, an die Sie den Job senden.
- Ein spezieller Parameter, der beim Erstellen eines Jobs an den `CreateJob` Vorgang übergeben wird.
  - `deadline:priority`— Der Wert muss eine Ganzzahl sein. Er wird als Prioritätsparameter an die `CreateJob` Operation übergeben.
  - `deadline:targetTaskRunStatus`— Der Wert muss eine Zeichenfolge sein. Er wird als targetTaskRunStatus-Parameter an die `CreateJob` Operation übergeben.
  - `deadline:maxFailedTasksCount`— Der Wert muss eine Ganzzahl sein. Er wird als maxFailedTasksCount-Parameter an die `CreateJob` Operation übergeben.
  - `deadline:maxRetriesPerTask`— Der Wert muss eine Ganzzahl sein. Er wird als maxRetriesPerTask-Parameter an die `CreateJob` Operation übergeben.
  - `deadline:maxWorkercount`— Der Wert muss eine Ganzzahl sein. Er wird als maxWorkerCountParameter an die `CreateJob` Operation übergeben.

Bei einer Auftragsvorlage handelt es sich immer um eine Vorlage und nicht um einen bestimmten auszuführenden Job. Eine Datei mit Parameterwerten ermöglicht es einem Job-Bundle, entweder als Vorlage zu dienen, wenn für einige Parameter in dieser Datei keine Werte definiert sind, oder als spezifische Jobübermittlung, wenn alle Parameter Werte haben.

Das Beispiel blender_render hat beispielsweise keine Parameterdatei und die zugehörige Jobvorlage definiert Parameter ohne Standardwerte. Diese Vorlage muss als Vorlage für die Erstellung von Jobs verwendet werden. Nachdem Sie mit diesem Job-Bundle einen Job erstellt haben, schreibt Deadline Cloud ein neues Job-Bundle in das Job-Verlaufsverzeichnis.

Zum Beispiel, wenn Sie einen Job mit dem folgenden Befehl einreichen:

```
deadline bundle gui-submit blender_render/
```

Das neue Job-Bundle enthält eine `parameter_values.yaml` Datei, die die angegebenen Parameter enthält:

```
% cat ~/.deadline/job_history/\(default\)/2024-06/2024-06-20-01-JobBundle-Demo/
parameter_values.yaml
parameterValues:
- name: deadline:targetTaskRunStatus
  value: READY
- name: deadline:maxFailedTasksCount
  value: 10
- name: deadline:maxRetriesPerTask
  value: 5
- name: deadline:priority
  value: 75
- name: BlenderSceneFile
  value: /private/tmp/bundle_demo/bmw27_cpu.blend
- name: Frames
  value: 1-10
- name: OutputDir
  value: /private/tmp/bundle_demo/output
- name: OutputPattern
  value: output_####
- name: Format
  value: PNG
- name: CondaPackages
  value: blender
- name: RezPackages
```

```
  value: blender
```

Sie können denselben Job mit dem folgenden Befehl erstellen:

```
deadline bundle submit ~/.deadline/job_history/\(default\)/2024-06/2024-06-20-01-
JobBundle-Demo/
```

> **Note**
>
> Das von Ihnen eingereichte Job-Bundle wird in Ihrem Job-Verlaufsverzeichnis gespeichert. Sie können den Speicherort dieses Verzeichnisses mit dem folgenden Befehl ermitteln:
>
> ```
> deadline config get settings.job_history_dir
> ```

Asset-Referenzen, Elemente für Job-Bundles

Sie können Deadline Cloud-Jobanhänge verwenden, um Dateien zwischen Ihrer Workstation und Deadline Cloud hin und her zu übertragen. Die Asset-Referenzdatei listet Eingabedateien und -verzeichnisse sowie Ausgabeverzeichnisse für Ihre Anhänge auf. Wenn Sie nicht alle Dateien und Verzeichnisse in dieser Datei auflisten, können Sie sie auswählen, wenn Sie einen Job mit dem `deadline bundle gui-submit` Befehl einreichen.

Diese Datei hat keine Auswirkung, wenn Sie keine Jobanhänge verwenden.

Sie können die Jobvorlage entweder im YAML-Format (`asset_references.yaml`) oder im JSON-Format (`asset_references.json`) definieren. Die Beispiele in diesem Abschnitt werden im YAML-Format angezeigt.

In YAML ist das Format der Datei:

```
assetReferences:
    inputs:
        # Filenames on the submitting workstation whose file contents are needed as
        # inputs to run the job.
        filenames:
        - list of file paths
        # Directories on the submitting workstation whose contents are needed as inputs
```

```
        # to run the job.
        directories:
        - list of directory paths

    outputs:
        # Directories on the submitting workstation where the job writes output files
        # if running locally.
        directories:
        - list of directory paths

    # Paths referenced by the job, but not necessarily input or output.
    # Use this if your job uses the name of a path in some way, but does not explicitly
need
    # the contents of that path.
    referencedPaths:
    - list of directory paths
```

Bei der Auswahl der Eingabe- oder Ausgabedatei, die auf Amazon S3 hochgeladen werden soll, vergleicht Deadline Cloud den Dateipfad mit den Pfaden, die in Ihren Speicherprofilen aufgeführt sind. Jeder Dateisystemspeicherort SHARED vom Typ -type in einem Speicherprofil abstrahiert eine Netzwerk-Dateifreigabe, die auf Ihren Workstations und Worker-Hosts bereitgestellt wird. Deadline Cloud lädt nur Dateien hoch, die sich nicht auf einer dieser Dateifreigaben befinden.

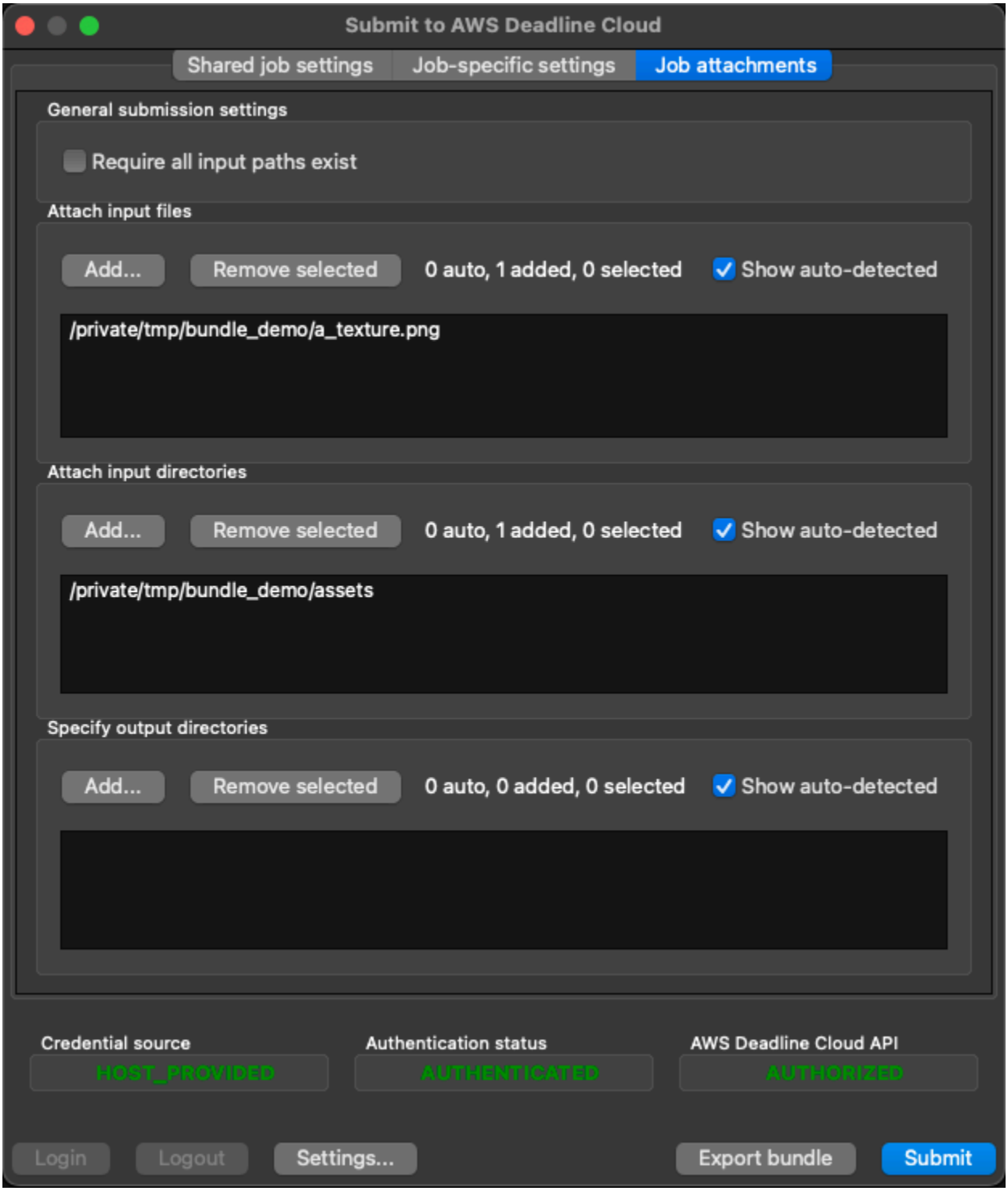
Weitere Informationen zum Erstellen und Verwenden von Speicherprofilen finden Sie unter Gemeinsamer Speicher in Deadline Cloud im AWS Deadline Cloud-Benutzerhandbuch.

Example- Die von der Deadline Cloud-GUI erstellte Asset-Referenzdatei

Verwenden Sie den folgenden Befehl, um einen Job mithilfe des Beispiels blender_render einzureichen.

```
deadline bundle gui-submit blender_render/
```

Fügen Sie dem Job auf der Registerkarte Jobanhänge einige zusätzliche Dateien hinzu:



Nachdem Sie den Job eingereicht haben, können Sie sich die `asset_references.yaml` Datei im Job-Bundle im Job-Verlaufsverzeichnis ansehen, um die Assets in der YAML-Datei zu sehen:

```
% cat ~/.deadline/job_history/\(default\)/2024-06/2024-06-20-01-JobBundle-Demo/
asset_references.yaml
```

```
assetReferences:
  inputs:
    filenames:
    - /private/tmp/bundle_demo/a_texture.png
    directories:
    - /private/tmp/bundle_demo/assets
  outputs:
    directories: []
  referencedPaths: []
```

Dateien in Ihren Jobs verwenden

Viele der Jobs, die Sie an AWS Deadline Cloud einreichen, enthalten Eingabe- und Ausgabedateien. Ihre Eingabedateien und Ausgabeverzeichnisse befinden sich möglicherweise auf einer Kombination aus gemeinsam genutzten Dateisystemen und lokalen Laufwerken. Jobs müssen den Inhalt an diesen Speicherorten lokalisieren. Deadline Cloud bietet zwei Funktionen: Jobanhänge und Speicherprofile, die zusammenarbeiten, damit Ihre Jobs die benötigten Dateien finden können.

Arbeitsanhänge bieten mehrere Vorteile

- Dateien mithilfe von Amazon S3 zwischen Hosts verschieben
- Übertragen Sie Dateien von Ihrer Workstation auf Worker-Hosts und umgekehrt
- Verfügbar für Jobs in Warteschlangen, in denen Sie die Funktion aktivieren
- Wird hauptsächlich für vom Service verwaltete Flotten verwendet, ist aber auch kompatibel mit vom Kunden verwalteten Flotten.

Verwenden Sie Speicherprofile, um das Layout gemeinsam genutzter Dateisystemspeicherorte auf Ihrer Workstation und Ihren Worker-Hosts zuzuordnen. Diese Zuordnung hilft Ihren Jobs dabei, gemeinsam genutzte Dateien und Verzeichnisse zu finden, wenn sich deren Speicherorte auf Ihrer Workstation und Ihren Worker-Hosts unterscheiden, z. B. bei plattformübergreifenden Konfigurationen mit Windows basierten Workstations und Linux basierten Worker-Hosts. Die Zuordnung Ihrer Dateisystemkonfiguration im Speicherprofil wird auch von Job-Anhängen verwendet, um die Dateien zu identifizieren, die für den Transfer zwischen Hosts über Amazon S3 benötigt werden.

Wenn Sie keine Jobanhänge verwenden und keine Datei- und Verzeichnisspeicherorte zwischen Workstations und Worker-Hosts neu zuordnen müssen, müssen Sie Ihre Fileshares nicht mit Speicherprofilen modellieren.

Themen

Beispiel für eine Projektinfrastruktur

Um die Verwendung von Job-Anhängen und Speicherprofilen zu demonstrieren, richten Sie eine Testumgebung mit zwei separaten Projekten ein. Sie können die Deadline Cloud-Konsole verwenden, um die Testressourcen zu erstellen.

1. Falls Sie dies noch nicht getan haben, erstellen Sie eine Testfarm. Gehen Sie wie unter Farm erstellen beschrieben vor, um eine Farm zu erstellen.
2. Erstellen Sie in jedem der beiden Projekte zwei Warteschlangen für Jobs. Gehen Sie wie unter Warteschlange erstellen beschrieben vor, um Warteschlangen zu erstellen.

    a. Erstellen Sie die erste **`Q1`** aufgerufene Warteschlange. Verwenden Sie die folgende Konfiguration und verwenden Sie die Standardeinstellungen für alle anderen Elemente.

    - Wählen Sie für Jobanhänge Create a new Amazon S3 S3-Bucket aus.
    - Wählen Sie Zuordnung zu kundenverwalteten Flotten aktivieren aus.
    - Geben Sie für „Als Benutzer ausführen“ sowohl **`jobuser`** für den POSIX-Benutzer als auch für die POSIX-Gruppe ein.
    - Erstellen Sie für die Warteschlangendienst-Rolle eine neue Rolle mit dem Namen **`AssetDemoFarm-Q1-Role`**
    - Deaktivieren Sie das Kontrollkästchen für die standardmäßige Conda-Warteschlangenumgebung.

    b. Erstellen Sie die zweite **`Q2`** aufgerufene Warteschlange. Verwenden Sie die folgende Konfiguration und verwenden Sie die Standardeinstellungen für alle anderen Elemente.

    - Wählen Sie für Jobanhänge Create a new Amazon S3 S3-Bucket aus.
    - Wählen Sie Zuordnung zu kundenverwalteten Flotten aktivieren aus.
    - Geben Sie für „Als Benutzer ausführen“ sowohl **`jobuser`** für den POSIX-Benutzer als auch für die POSIX-Gruppe ein.
    - Erstellen Sie für die Warteschlangendienst-Rolle eine neue Rolle mit dem Namen **`AssetDemoFarm-Q2-Role`**

    - Deaktivieren Sie das Kontrollkästchen für die standardmäßige Conda-Warteschlangenumgebung.

3. Erstellen Sie eine einzige, vom Kunden verwaltete Flotte, die die Jobs von beiden Warteschlangen aus ausführt. Um die Flotte zu erstellen, folgen Sie dem Verfahren unter Erstellen einer kundenverwalteten Flotte. Verwenden Sie die folgende Konfiguration:

    - Verwenden Sie als Name**`DemoFleet`**.
    - Wählen Sie als Flottenart die Option Vom Kunden verwaltet
    - Erstellen Sie für die Servicerolle Fleet eine neue Rolle mit dem Namen AssetDemoFarm-Fleet-Role.
    - Ordnen Sie die Flotte keinen Warteschlangen zu.

Die Testumgebung geht davon aus, dass es drei Dateisysteme gibt, die von Hosts gemeinsam genutzt werden, die Netzwerkdateifreigaben verwenden. In diesem Beispiel haben die Speicherorte die folgenden Namen:

- `FSCommon`- enthält Eingabe-Job-Assets, die beiden Projekten gemeinsam sind.
- `FS1`- enthält Eingabe- und Ausgabe-Auftragsressourcen für Projekt 1.
- `FS2`- enthält Eingabe- und Ausgabe-Auftragsressourcen für Projekt 2.

In der Testumgebung wird außerdem davon ausgegangen, dass es drei Arbeitsstationen gibt, und zwar wie folgt:

- `WSAll`- Eine Linux basierte Workstation, die von Entwicklern für alle Projekte verwendet wird. Die gemeinsam genutzten Speicherorte im Dateisystem sind:
    - `FSCommon`: `/shared/common`
    - `FS1`: `/shared/projects/project1`
    - `FS2`: `/shared/projects/project2`
- `WS1`- Eine Windows basierte Workstation, die für Projekt 1 verwendet wird. Die gemeinsam genutzten Speicherorte im Dateisystem sind:
    - `FSCommon`: `S:\`
    - `FS1`: `Z:\`
    - `FS2`: Nicht verfügbar

- `WS1`- Eine macOS basierte Workstation, die für Projekt 2 verwendet wird. Die Speicherorte des gemeinsam genutzten Dateisystems sind:
  - `FSCommon`: `/Volumes/common`
  - `FS1`: Nicht verfügbar
  - `FS2`: `/Volumes/projects/project2`

Definieren Sie abschließend die gemeinsam genutzten Dateisystemspeicherorte für die Mitarbeiter in Ihrer Flotte. Die folgenden Beispiele beziehen sich auf diese Konfiguration als`WorkerConfig`. Die gemeinsam genutzten Standorte sind:

- `FSCommon`: `/mnt/common`
- `FS1`: `/mnt/projects/project1`
- `FS2`: `/mnt/projects/project2`

Sie müssen keine gemeinsam genutzten Dateisysteme, Workstations oder Worker einrichten, die dieser Konfiguration entsprechen. Die gemeinsam genutzten Standorte müssen für die Demonstration nicht existieren.

Speicherprofile und Pfadzuweisung

Verwenden Sie Speicherprofile, um die Dateisysteme auf Ihrer Workstation und Ihren Worker-Hosts zu modellieren. Jedes Speicherprofil beschreibt das Betriebssystem und das Dateisystem-Layout einer Ihrer Systemkonfigurationen. In diesem Thema wird beschrieben, wie Sie Speicherprofile verwenden, um die Dateisystemkonfigurationen Ihrer Hosts zu modellieren, sodass Deadline Cloud Pfadzuordnungsregeln für Ihre Jobs generieren kann, und wie diese Pfadzuordnungsregeln aus Ihren Speicherprofilen generiert werden.

Wenn Sie einen Job an Deadline Cloud einreichen, können Sie eine optionale Speicherprofil-ID für den Job angeben. Dieses Speicherprofil beschreibt das Dateisystem der einreichenden Workstation. Es beschreibt die ursprüngliche Dateisystemkonfiguration, die von den Dateipfaden in der Jobvorlage verwendet wird.

Sie können einer Flotte auch ein Speicherprofil zuordnen. Das Speicherprofil beschreibt die Dateisystemkonfiguration aller Worker-Hosts in der Flotte. Wenn Sie über Worker mit einer anderen Dateisystemkonfiguration verfügen, müssen diese Worker einer anderen Flotte in Ihrer Farm zugewiesen werden.

Pfadzuordnungsregeln beschreiben, wie Pfade neu zugeordnet werden sollten, und zwar von der Art, wie sie im Job angegeben sind, bis zum tatsächlichen Speicherort des Pfads auf einem Worker-Host. Deadline Cloud vergleicht die im Speicherprofil eines Jobs beschriebene Dateisystemkonfiguration mit dem Speicherprofil der Flotte, die den Job ausführt, um diese Pfadzuordnungsregeln abzuleiten.

Themen

Modellieren Sie gemeinsam genutzte Dateisystemspeicherorte mit Speicherprofilen

Ein Speicherprofil modelliert die Dateisystemkonfiguration einer Ihrer Hostkonfigurationen. In der Beispielprojektinfrastruktur gibt es vier verschiedene Hostkonfigurationen. In diesem Beispiel erstellen Sie für jedes ein separates Speicherprofil. Sie können ein Speicherprofil mit einer der folgenden Methoden erstellen:

- CreateStorageProfile API
- AWS::Deadline::StorageProfile CloudFormation Ressource
- AWS -Konsole

Ein Speicherprofil besteht aus einer Liste von Dateisystemspeicherorten, die Deadline Cloud jeweils den Speicherort und den Typ eines Dateisystemspeicherorts mitteilen, der für Jobs relevant ist, die von einem Host eingereicht oder auf einem Host ausgeführt werden. Ein Speicherprofil sollte nur die Standorte modellieren, die für Jobs relevant sind. Der gemeinsam genutzte `FSCommon` Speicherort befindet sich beispielsweise auf der Workstation `WS1` unter`S:\`, sodass der entsprechende Speicherort im Dateisystem wie folgt lautet:

```
{
    "name": "FSCommon",
    "path": "S:\\",
    "type": "SHARED"
}
```

Verwenden Sie die folgenden Befehle, um das Speicherprofil für Workstation-Konfigurationen `WS1` `WS3` und die Worker-Konfiguration `WorkerConfig` mithilfe von AWS CLIin zu erstellen AWS CloudShell: `WS2`

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff

aws deadline create-storage-profile --farm-id $FARM_ID \
  --display-name WSAll \
  --os-family LINUX \
  --file-system-locations \
  '[
      {"name": "FSCommon", "type":"SHARED", "path":"/shared/common"},
      {"name": "FS1", "type":"SHARED", "path":"/shared/projects/project1"},
      {"name": "FS2", "type":"SHARED", "path":"/shared/projects/project2"}
  ]'

aws deadline create-storage-profile --farm-id $FARM_ID \
  --display-name WS1 \
  --os-family WINDOWS \
  --file-system-locations \
  '[
      {"name": "FSCommon", "type":"SHARED", "path":"S:\\"},
      {"name": "FS1", "type":"SHARED", "path":"Z:\\"}
   ]'

aws deadline create-storage-profile --farm-id $FARM_ID \
  --display-name WS2 \
  --os-family MACOS \
  --file-system-locations \
  '[
      {"name": "FSCommon", "type":"SHARED", "path":"/Volumes/common"},
      {"name": "FS2", "type":"SHARED", "path":"/Volumes/projects/project2"}
  ]'

aws deadline create-storage-profile --farm-id $FARM_ID \
  --display-name WorkerCfg \
  --os-family LINUX \
  --file-system-locations \
  '[
      {"name": "FSCommon", "type":"SHARED", "path":"/mnt/common"},
      {"name": "FS1", "type":"SHARED", "path":"/mnt/projects/project1"},
      {"name": "FS2", "type":"SHARED", "path":"/mnt/projects/project2"}
```

```
  ]'
```

> **Note**
>
> Sie müssen auf die Dateisystemspeicherorte in Ihren Speicherprofilen verweisen und dabei dieselben Werte für die `name` Eigenschaft in allen Speicherprofilen in Ihrer Farm verwenden. Deadline Cloud vergleicht beim Generieren von Pfadzuordnungsregeln die Namen, um festzustellen, dass Dateisystemspeicherorte aus verschiedenen Speicherprofilen auf denselben Speicherort verweisen.

Konfigurieren Sie Speicherprofile für Flotten

Sie können eine Flotte so konfigurieren, dass sie ein Speicherprofil enthält, das die Dateisystemstandorte aller Mitarbeiter in der Flotte modelliert. Die Host-Dateisystemkonfiguration aller Mitarbeiter in einer Flotte muss mit dem Speicherprofil ihrer Flotte übereinstimmen. Mitarbeiter mit unterschiedlichen Dateisystemkonfigurationen müssen sich in separaten Flotten befinden.

Um die Konfiguration Ihrer Flotte für die Verwendung des `WorkerConfig` Speicherprofils festzulegen, verwenden Sie den folgenden Befehl AWS CLI: AWS CloudShell

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of FLEET_ID to your fleet's identifier
FLEET_ID=fleet-00112233445566778899aabbccddeeff
# Change the value of WORKER_CFG_ID to your storage profile named WorkerConfig
WORKER_CFG_ID=sp-00112233445566778899aabbccddeeff

FLEET_WORKER_MODE=$( \
  aws deadline get-fleet --farm-id $FARM_ID --fleet-id $FLEET_ID \
   --query '.configuration.customerManaged.mode' \
)
FLEET_WORKER_CAPABILITIES=$( \
  aws deadline get-fleet --farm-id $FARM_ID --fleet-id $FLEET_ID \
   --query '.configuration.customerManaged.workerCapabilities' \
)

aws deadline update-fleet --farm-id $FARM_ID --fleet-id $FLEET_ID \
  --configuration \
  "{
    \"customerManaged\": {
```

```
      \"storageProfileId\": \"$WORKER_CFG_ID\",
      \"mode\": $FLEET_WORKER_MODE,
      \"workerCapabilities\": $FLEET_WORKER_CAPABILITIES
    }
  }"
```

Konfigurieren Sie Speicherprofile für Warteschlangen

Die Konfiguration einer Warteschlange umfasst eine Liste von Namen der gemeinsam genutzten Dateisystemspeicherorte, auf die an die Warteschlange übermittelte Jobs Zugriff benötigen, wobei Groß- und Kleinschreibung beachtet werden muss. Beispielsweise erfordern an die Warteschlange übermittelte Jobs `Q1` Dateisystemspeicherorte und. `FSCommon` `FS1` Für an die Warteschlange übermittelte Jobs `Q2` sind Dateisystemspeicherorte `FSCommon` und erforderlich. `FS2`

Verwenden Sie das folgende Skript, um die Konfigurationen der Warteschlange so einzustellen, dass diese Dateisystemspeicherorte erforderlich sind:

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of QUEUE2_ID to queue Q2's identifier
QUEUE2_ID=queue-00112233445566778899aabbccddeeff

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
  --required-file-system-location-names-to-add FSComm FS1

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE2_ID \
  --required-file-system-location-names-to-add FSComm FS2
```

Die Konfiguration einer Warteschlange umfasst auch eine Liste der zulässigen Speicherprofile, die für Aufträge gilt, die an diese Warteschlange weitergeleitet wurden, und für Flotten, die mit dieser Warteschlange verknüpft sind. In der Liste der zulässigen Speicherprofile der Warteschlange sind nur Speicherprofile zulässig, die Dateisystemspeicherorte für alle erforderlichen Dateisystemspeicherorte für die Warteschlange definieren.

Ein Job schlägt fehl, wenn Sie ihn mit einem Speicherprofil einreichen, das nicht in der Liste der zulässigen Speicherprofile für die Warteschlange enthalten ist. Sie können einen Job ohne Speicherprofil jederzeit an eine Warteschlange senden. Die Workstation-Konfigurationen sind beschriftet `WSAll` und `WS1` beide verfügen über die erforderlichen Dateisystemspeicherorte

(FSCommonundFS1) für die WarteschlangeQ1. Sie müssen berechtigt sein, Jobs an die Warteschlange weiterzuleiten. In ähnlicher Weise werden die Workstation-Konfigurationen WSAll und die WS2 Anforderungen für die Warteschlange erfülltQ2. Sie müssen in der Lage sein, Jobs an diese Warteschlange weiterzuleiten. Aktualisieren Sie beide Warteschlangenkonfigurationen, sodass Jobs mit diesen Speicherprofilen mithilfe des folgenden Skripts gesendet werden können:

```
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff
# Change the value of WS1 to the identifier of the WS1 storage profile
WS1_ID=sp-00112233445566778899aabbccddeeff
# Change the value of WS2 to the identifier of the WS2 storage profile
WS2_ID=sp-00112233445566778899aabbccddeeff

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
  --allowed-storage-profile-ids-to-add $WSALL_ID $WS1_ID

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE2_ID \
  --allowed-storage-profile-ids-to-add $WSALL_ID $WS2_ID
```

Wenn Sie das WS2 Speicherprofil zur Liste der zulässigen Speicherprofile für die Warteschlange hinzufügen, schlägt Q1 dies fehl:

```
$ aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
  --allowed-storage-profile-ids-to-add $WS2_ID

An error occurred (ValidationException) when calling the UpdateQueue operation: Storage
 profile id: sp-00112233445566778899aabbccddeeff does not have required file system
 location: FS1
```

Das liegt daran, dass das WS2 Speicherprofil keine Definition für den Dateisystemspeicherort enthältFS1, den diese Warteschlange Q1 benötigt.

Das Zuordnen einer konfigurierten Flotte zu einem Speicherprofil, das nicht in der Liste der zulässigen Speicherprofile der Warteschlange enthalten ist, schlägt ebenfalls fehl. Beispiel:

```
$ aws deadline create-queue-fleet-association --farm-id $FARM_ID \
   --fleet-id $FLEET_ID \
   --queue-id $QUEUE1_ID

An error occurred (ValidationException) when calling the CreateQueueFleetAssociation
 operation: Mismatch between storage profile ids.
```

Um den Fehler zu beheben, fügen Sie das angegebene Speicherprofil der Liste der zulässigen Speicherprofile sowohl für die Warteschlange als auch für die Warteschlange `Q1` hinzu. `WorkerConfig` `Q2` Ordnen Sie dann die Flotte diesen Warteschlangen zu, sodass die Mitarbeiter der Flotte Aufträge aus beiden Warteschlangen ausführen können.

```
# Change the value of FLEET_ID to your fleet's identifier
FLEET_ID=fleet-00112233445566778899aabbccddeeff
# Change the value of WORKER_CFG_ID to your storage profile named WorkerCfg
WORKER_CFG_ID=sp-00112233445566778899aabbccddeeff

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
  --allowed-storage-profile-ids-to-add $WORKER_CFG_ID

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE2_ID \
  --allowed-storage-profile-ids-to-add $WORKER_CFG_ID

aws deadline create-queue-fleet-association --farm-id $FARM_ID \
  --fleet-id $FLEET_ID \
  --queue-id $QUEUE1_ID

aws deadline create-queue-fleet-association --farm-id $FARM_ID \
  --fleet-id $FLEET_ID \
  --queue-id $QUEUE2_ID
```

Leiten Sie Pfadzuordnungsregeln aus Speicherprofilen ab

Pfadzuordnungsregeln beschreiben, wie Pfade vom Job zum tatsächlichen Speicherort des Pfads auf einem Worker-Host neu zugeordnet werden sollten. Wenn eine Aufgabe auf einem Worker ausgeführt wird, wird das Speicherprofil des Jobs mit dem Speicherprofil der Worker-Flotte verglichen, um die Pfadzuordnungsregeln für die Aufgabe abzuleiten.

Deadline Cloud erstellt eine Zuordnungsregel für jeden der erforderlichen Dateisystemspeicherorte in der Konfiguration der Warteschlange. Beispielsweise hat ein Job, der mit dem `WSAll` Speicherprofil an die Warteschlange gesendet `Q1` wurde, die Pfadzuordnungsregeln:

- `FSComm`: `/shared/common -> /mnt/common`
- `FS1`: `/shared/projects/project1 -> /mnt/projects/project1`

Deadline Cloud erstellt Regeln für die Speicherorte `FSComm` und das `FS1` Dateisystem, aber nicht für den Speicherort des `FS2` Dateisystems, obwohl `WSAll` sowohl die `WorkerConfig`

Speicherprofile als auch diese definierenFS2. Dies liegt daran, dass die Liste Q1 der erforderlichen Dateisystemspeicherorte in der Warteschlange lautet`["FSComm", "FS1"]`.

Sie können die Pfadzuordnungsregeln überprüfen, die für Jobs verfügbar sind, die mit einem bestimmten Speicherprofil eingereicht wurden, indem Sie einen Job einreichen, der die Datei mit den Pfadzuordnungsregeln von Open Job Description ausdruckt, und dann das Sitzungsprotokoll lesen, nachdem der Job abgeschlossen ist:

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of WSALL_ID to the identifier of the WSALL storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

aws deadline create-job --farm-id $FARM_ID --queue-id $QUEUE1_ID \
  --priority 50 \
  --storage-profile-id $WSALL_ID \
  --template-type JSON --template \
  '{
    "specificationVersion": "jobtemplate-2023-09",
    "name": "DemoPathMapping",
    "steps": [
      {
        "name": "ShowPathMappingRules",
        "script": {
          "actions": {
            "onRun": {
              "command": "/bin/cat",
              "args": [ "{{Session.PathMappingRulesFile}}" ]
            }
          }
        }
      }
    ]
  }'
```

Wenn Sie die Deadline Cloud-CLI zum Senden von Jobs verwenden, legt deren `settings.storage_profile_id` Konfigurationseinstellung das Speicherprofil fest, das Jobs haben, die mit der CLI eingereicht werden. Um Jobs mit dem `WSAll` Speicherprofil einzureichen, legen Sie Folgendes fest:

```
deadline config set settings.storage_profile_id $WSALL_ID
```

Um einen vom Kunden verwalteten Worker so auszuführen, als ob er in der Beispielinfrastruktur ausgeführt würde, folgen Sie dem Verfahren unter Den Worker-Agent ausführen im Deadline Cloud-Benutzerhandbuch, um einen Worker auszuführen. AWS CloudShell Wenn Sie diese Anweisungen bereits befolgt haben, löschen Sie zuerst die `~/demoenv-persist` Verzeichnisse `~/demoenv-logs` und. Legen Sie vorher auch die Werte der `DEV_CMF_ID` Umgebungsvariablen `DEV_FARM_ID` und, auf die sich die Anweisungen beziehen, wie folgt fest:

```
DEV_FARM_ID=$FARM_ID
DEV_CMF_ID=$FLEET_ID
```

Nach der Ausführung des Jobs können Sie die Pfadzuordnungsregeln in der Protokolldatei des Jobs sehen:

```
cat demoenv-logs/${QUEUE1_ID}/*.log
...
JJSON log results (see below)
...
```

Das Protokoll enthält Zuordnungen sowohl für das `FS1` als auch für das `FSComm` Dateisystem. Der Protokolleintrag wurde aus Gründen der Lesbarkeit neu formatiert und sieht wie folgt aus:

```
{
    "version": "pathmapping-1.0",
    "path_mapping_rules": [
        {
            "source_path_format": "POSIX",
            "source_path": "/shared/projects/project1",
            "destination_path": "/mnt/projects/project1"
        },
        {
            "source_path_format": "POSIX",
            "source_path": "/shared/common",
            "destination_path": "/mnt/common"
        }
    ]
```

Sie können Jobs mit unterschiedlichen Speicherprofilen einreichen, um zu sehen, wie sich die Pfadzuordnungsregeln ändern.

Verwenden Sie Jobanhänge, um Dateien zu teilen

Verwenden Sie Jobanhänge, um Dateien, die sich nicht in gemeinsam genutzten Verzeichnissen befinden, für Ihre Jobs verfügbar zu machen und die Ausgabedateien zu erfassen, falls sie nicht in gemeinsam genutzte Verzeichnisse geschrieben wurden. Job Attachments verwendet Amazon S3, um Dateien zwischen Hosts zu übertragen. Dateien werden in S3-Buckets gespeichert, und Sie müssen eine Datei nicht hochladen, wenn sich ihr Inhalt nicht geändert hat.

Sie müssen Auftragsanhänge verwenden, wenn Sie Jobs auf vom Service verwalteten Flotten ausführen, da Hosts die Speicherorte im Dateisystem nicht gemeinsam nutzen. Jobanhänge sind auch für vom Kunden verwaltete Flotten nützlich, wenn die Eingabe- oder Ausgabedateien eines Jobs in einem gemeinsam genutzten Netzwerkdateisystem gespeichert sind, z. B. wenn Ihr Auftragspaket Shell- oder Python-Skripte enthält.

Wenn Sie ein Job-Bundle entweder mit der Deadline Cloud-CLI oder einem Deadline Cloud-Absender einreichen, verwenden Jobanhänge das Speicherprofil des Jobs und die erforderlichen Dateisystemspeicherorte der Warteschlange, um die Eingabedateien zu identifizieren, die sich nicht auf einem Worker-Host befinden und im Rahmen der Auftragsübermittlung auf Amazon S3 hochgeladen werden sollten. Diese Speicherprofile helfen Deadline Cloud auch dabei, die Ausgabedateien an Worker-Host-Standorten zu identifizieren, die auf Amazon S3 hochgeladen werden müssen, damit sie auf Ihrer Workstation verfügbar sind.

Die Beispiele für Jobanhänge verwenden die Konfigurationen für Farm, Flotte, Warteschlangen und Speicherprofile von Beispiel für eine Projektinfrastruktur undSpeicherprofile und Pfadzuweisung. Sie sollten diese Abschnitte vor diesem durchgehen.

In den folgenden Beispielen verwenden Sie ein Beispiel-Job-Bundle als Ausgangspunkt und ändern es dann, um die Funktionalität von Job Attachment zu untersuchen. Job-Bundles sind die beste Methode, um Job-Anhänge für Ihre Jobs zu verwenden. Sie kombinieren eine Jobvorlage „Open Job Description“ in einem Verzeichnis mit zusätzlichen Dateien, in denen die Dateien und Verzeichnisse aufgeführt sind, die für Jobs benötigt werden, die das Job-Bundle verwenden. Weitere Informationen zu Jobpaketen finden Sie unterVorlagen für offene Stellenbeschreibungen (OpenJD) für Deadline Cloud.

Dateien mit einem Job einreichen

Mit Deadline Cloud können Sie Job-Workflows für den Zugriff auf Eingabedateien aktivieren, die an gemeinsam genutzten Dateisystemen auf Worker-Hosts nicht verfügbar sind. Mit Auftragsanhängen

können Renderaufträge auf Dateien zugreifen, die sich nur auf einem lokalen Workstation-Laufwerk oder in einer vom Service verwalteten Flottenumgebung befinden. Wenn Sie ein Auftragspaket einreichen, können Sie Listen mit Eingabedateien und Verzeichnissen hinzufügen, die für den Job erforderlich sind. Deadline Cloud identifiziert diese nicht gemeinsam genutzten Dateien, lädt sie vom lokalen Computer auf Amazon S3 hoch und lädt sie auf den Worker-Host herunter. Es optimiert den Prozess der Übertragung von Eingabe-Assets an Renderknoten und stellt sicher, dass alle erforderlichen Dateien für die verteilte Auftragsausführung zugänglich sind.

Sie können die Dateien für Jobs direkt im Job-Bundle angeben, Parameter in der Jobvorlage verwenden, die Sie mithilfe von Umgebungsvariablen oder einem Skript bereitstellen, und die `assets_references` Datei des Jobs verwenden. Sie können eine dieser Methoden oder eine Kombination aus allen dreien verwenden. Sie können ein Speicherprofil für das Paket für den Job angeben, sodass nur Dateien hochgeladen werden, die sich auf der lokalen Arbeitsstation geändert haben.

In diesem Abschnitt wird anhand eines Beispiel-Job-Bundles von GitHub gezeigt, wie Deadline Cloud die Dateien in Ihrem Job zum Hochladen identifiziert, wie diese Dateien in Amazon S3 organisiert sind und wie sie den Worker-Hosts zur Verfügung gestellt werden, die Ihre Jobs verarbeiten.

Themen

- So lädt Deadline Cloud Dateien auf Amazon S3 hoch
- Wie Deadline Cloud die hochzuladenden Dateien auswählt
- Wie finden Jobs Eingabedateien für Jobanhänge

So lädt Deadline Cloud Dateien auf Amazon S3 hoch

Dieses Beispiel zeigt, wie Deadline Cloud Dateien von Ihrer Workstation oder Ihrem Worker-Host auf Amazon S3 hochlädt, damit sie gemeinsam genutzt werden können. Es verwendet ein Beispiel-Job-Bundle von GitHub und die Deadline Cloud-CLI, um Jobs einzureichen.

Klonen Sie zunächst das Deadline GitHub Cloud-Beispiel-Repository in Ihre AWS CloudShellUmgebung und kopieren Sie dann das `job_attachments_devguide` Job-Bundle in Ihr Home-Verzeichnis:

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
cp -r deadline-cloud-samples/job_bundles/job_attachments_devguide ~/
```

Installieren Sie die Deadline Cloud-CLI, um Job-Bundles einzureichen:

```
pip install deadline --upgrade
```

Das `job_attachments_devguide` Job-Bundle besteht aus einem einzigen Schritt mit einer Aufgabe, die ein Bash-Shell-Skript ausführt, dessen Dateisystemspeicherort als Jobparameter übergeben wird. Die Definition des Job-Parameters lautet:

```
...
- name: ScriptFile
  type: PATH
  default: script.sh
  dataFlow: IN
  objectType: FILE
...
```

Der `IN` Wert der `dataFlow` Eigenschaft teilt Job-Anhängen mit, dass der Wert des `ScriptFile` Parameters eine Eingabe für den Job ist. Der Wert der `default` Eigenschaft ist ein relativer Speicherort zum Verzeichnis des Job-Bundles, es kann sich aber auch um einen absoluten Pfad handeln. Diese Parameterdefinition deklariert die `script.sh` Datei im Verzeichnis des Job-Bundles als Eingabedatei, die für die Ausführung des Jobs erforderlich ist.

Stellen Sie als Nächstes sicher, dass für die Deadline Cloud-CLI kein Speicherprofil konfiguriert ist, und senden Sie den Job dann in die WarteschlangeQ1:

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff

deadline config set settings.storage_profile_id ''

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID
 job_attachments_devguide/
```

Die Ausgabe der Deadline Cloud-CLI nach der Ausführung dieses Befehls sieht wie folgt aus:

```
Submitting to Queue: Q1
...
Hashing Attachments  [####################################]  100%
Hashing Summary:
    Processed 1 file totaling 39.0 B.
```

```
    Skipped re-processing 0 files totaling 0.0 B.
    Total processing time of 0.0327 seconds at 1.19 KB/s.

Uploading Attachments  [####################################]  100%
Upload Summary:
    Processed 1 file totaling 39.0 B.
    Skipped re-processing 0 files totaling 0.0 B.
    Total processing time of 0.25639 seconds at 152.0 B/s.

Waiting for Job to be created...
Submitted job bundle:
   job_attachments_devguide/
Job creation completed successfully
job-74148c13342e4514b63c7a7518657005
```

Wenn Sie den Job einreichen, hascht Deadline Cloud die `script.sh` Datei zuerst und lädt sie dann auf Amazon S3 hoch.

Deadline Cloud behandelt den S3-Bucket als inhaltsadressierbaren Speicher. Dateien werden in S3-Objekte hochgeladen. Der Objektname wird aus einem Hash des Dateiinhalts abgeleitet. Wenn zwei Dateien identischen Inhalt haben, haben sie unabhängig davon, wo sich die Dateien befinden oder wie sie benannt sind, denselben Hashwert. Dieser inhaltsadressierbare Speicher ermöglicht es Deadline Cloud, das Hochladen einer Datei zu vermeiden, wenn sie bereits verfügbar ist.

Sie können die AWS-CLI verwenden, um die Objekte zu sehen, die auf Amazon S3 hochgeladen wurden:

```
# The name of queue `Q1`'s job attachments S3 bucket
Q1_S3_BUCKET=$(
  aws deadline get-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
    --query 'jobAttachmentSettings.s3BucketName' | tr -d '"'
)

aws s3 ls s3://$Q1_S3_BUCKET --recursive
```

Zwei Objekte wurden auf S3 hochgeladen:

- `DeadlineCloud/Data/87cb19095dd5d78fcaf56384ef0e6241.xxh128`— Der Inhalt von`script.sh`. Der Wert `87cb19095dd5d78fcaf56384ef0e6241` im Objektschlüssel ist der Hash des Dateiinhalts, und die Erweiterung `xxh128` gibt an, dass der Hashwert als 128-Bit-xxhash berechnet wurde.

- `DeadlineCloud/Manifests/<farm-id>/<queue-id>/Inputs/<guid>/a1d221c7fd97b08175b3872a37428e8c_input`— Das Manifest-Objekt für die Auftragsübergabe. Die Werte `<farm-id>`<queue-id>, und `<guid>` sind Ihre Farm-ID, Ihre Warteschlangen-ID und ein zufälliger Hexadezimalwert. Der Wert `a1d221c7fd97b08175b3872a37428e8c` in diesem Beispiel ist ein Hashwert, der anhand der Zeichenfolge`/home/cloudshell-user/job_attachments_devguide`, dem Verzeichnis, in dem er sich `script.sh` befindet, berechnet wird.

Das Manifest-Objekt enthält die Informationen für die Eingabedateien in einem bestimmten Stammpfad, die im Rahmen der Auftragsübermittlung auf S3 hochgeladen wurden. Laden Sie diese Manifestdatei herunter (`aws s3 cp s3://$Q1_S3_BUCKET/<objectname>`). Ihr Inhalt ist ähnlich wie:

```
{
    "hashAlg": "xxh128",
    "manifestVersion": "2023-03-03",
    "paths": [
        {
            "hash": "87cb19095dd5d78fcaf56384ef0e6241",
            "mtime": 1721147454416085,
            "path": "script.sh",
            "size": 39
        }
    ],
    "totalSize": 39
}
```

Dies bedeutet, dass die Datei hochgeladen `script.sh` wurde, und der Hash des Inhalts dieser Datei lautet`87cb19095dd5d78fcaf56384ef0e6241`. Dieser Hashwert entspricht dem Wert im Objektnamen`DeadlineCloud/Data/87cb19095dd5d78fcaf56384ef0e6241.xxh128`. Es wird von Deadline Cloud verwendet, um zu wissen, welches Objekt für den Inhalt dieser Datei heruntergeladen werden muss.

Das vollständige Schema für diese Datei ist verfügbar in GitHub.

Wenn Sie den CreateJob Vorgang verwenden, können Sie den Speicherort der Manifestobjekte festlegen. Sie können die GetJobOperation verwenden, um den Standort zu sehen:

```
{
    "attachments": {
```

```
        "file system": "COPIED",
        "manifests": [
            {
                "inputManifestHash": "5b0db3d311805ea8de7787b64cbbe8b3",
                "inputManifestPath": "<farm-id>/<queue-id>/Inputs/<guid>/
a1d221c7fd97b08175b3872a37428e8c_input",
                "rootPath": "/home/cloudshell-user/job_attachments_devguide",
                "rootPathFormat": "posix"
            }
        ]
    },
    ...
}
```

Wie Deadline Cloud die hochzuladenden Dateien auswählt

Die Dateien und Verzeichnisse, die Job Attachments für den Upload auf Amazon S3 als Eingaben für Ihren Job in Betracht ziehen, sind:

- Die Werte aller Jobparameter `PATH` vom Typ -type, die in der Jobvorlage des Job-Bundles mit dem `dataFlow` Wert `IN` oder `INOUT` definiert sind.
- Die Dateien und Verzeichnisse, die als Eingaben in der Asset-Referenzdatei des Job-Bundles aufgeführt sind.

Wenn Sie einen Job ohne Speicherprofil einreichen, werden alle Dateien hochgeladen, die für den Upload in Frage kommen. Wenn Sie einen Job mit einem Speicherprofil einreichen, werden Dateien nicht auf Amazon S3 hochgeladen, wenn sie sich in den Dateisystemspeicherorten des Speicherprofils `SHARED` vom Typ -type befinden, die auch erforderliche Dateisystemspeicherorte für die Warteschlange sind. Es wird davon ausgegangen, dass diese Speicherorte auf den Worker-Hosts verfügbar sind, auf denen der Job ausgeführt wird, sodass sie nicht auf S3 hochgeladen werden müssen.

In diesem Beispiel erstellen Sie `SHARED` Dateisystemspeicherorte `WSAll` in Ihrer CloudShell AWS-Umgebung und fügen dann Dateien zu diesen Dateisystemspeicherorten hinzu. Verwenden Sie den folgenden Befehl:

```
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

sudo mkdir -p /shared/common /shared/projects/project1 /shared/projects/project2
```

```
sudo chown -R cloudshell-user:cloudshell-user /shared

for d in /shared/common /shared/projects/project1 /shared/projects/project2; do
  echo "File contents for $d" > ${d}/file.txt
done
```

Als Nächstes fügen Sie dem Job-Bundle eine Asset-Referenzdatei hinzu, die alle Dateien enthält, die Sie als Eingaben für den Job erstellt haben. Verwenden Sie den folgenden Befehl:

```
cat > ${HOME}/job_attachments_devguide/asset_references.yaml << EOF
assetReferences:
  inputs:
    filenames:
    - /shared/common/file.txt
    directories:
    - /shared/projects/project1
    - /shared/projects/project2
EOF
```

Als Nächstes konfigurieren Sie die Deadline Cloud-CLI so, dass Jobs mit dem `WSAll` Speicherprofil gesendet werden, und senden Sie dann das Job-Bundle:

```

# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

deadline config set settings.storage_profile_id $WSALL_ID

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID
 job_attachments_devguide/
```

Deadline Cloud lädt zwei Dateien auf Amazon S3 hoch, wenn Sie den Job einreichen. Sie können die Manifestobjekte für den Job von S3 herunterladen, um die hochgeladenen Dateien zu sehen:

```
for manifest in $( \
  aws deadline get-job --farm-id $FARM_ID --queue-id $QUEUE1_ID --job-id $JOB_ID \
    --query 'attachments.manifests[].inputManifestPath' \
    | jq -r '.[]'
```

```
); do
  echo "Manifest object: $manifest"
  aws s3 cp --quiet s3://$Q1_S3_BUCKET/DeadlineCloud/Manifests/$manifest /dev/stdout |
 jq .
done
```

In diesem Beispiel gibt es eine einzelne Manifestdatei mit dem folgenden Inhalt:

```
{
    "hashAlg": "xxh128",
    "manifestVersion": "2023-03-03",
    "paths": [
        {
            "hash": "87cb19095dd5d78fcaf56384ef0e6241",
            "mtime": 1721147454416085,
            "path": "home/cloudshell-user/job_attachments_devguide/script.sh",
            "size": 39
        },
        {
            "hash": "af5a605a3a4e86ce7be7ac5237b51b79",
            "mtime": 1721163773582362,
            "path": "shared/projects/project2/file.txt",
            "size": 44
        }
    ],
    "totalSize": 83
}
```

Verwenden Sie die GetJob Operation für das Manifest, um zu überprüfen, ob der „/“ `rootPath` ist.

```
aws deadline get-job --farm-id $FARM_ID --queue-id $QUEUE1_ID --job-id $JOB_ID --query
 'attachments.manifests[*]'
```

Der Stammpfad für eine Reihe von Eingabedateien ist immer der längste gemeinsame Unterpfad dieser Dateien. Wenn Ihr Job Windows stattdessen von eingereicht wurde und es Eingabedateien ohne gemeinsamen Unterpfad gibt, weil sie sich auf unterschiedlichen Laufwerken befanden, sehen Sie auf jedem Laufwerk einen eigenen Stammpfad. Die Pfade in einem Manifest beziehen sich immer auf den Stammpfad des Manifests. Es wurden also folgende Eingabedateien hochgeladen:

- `/home/cloudshell-user/job_attachments_devguide/script.sh`— Die Skriptdatei im Job-Bundle.

- `/shared/projects/project2/file.txt`— Die Datei an einem `SHARED` Dateisystemspeicherort im `WSAll` Speicherprofil, der nicht in der Liste der erforderlichen Dateisystemspeicherorte für die Warteschlange enthalten ist`Q1`.

Die Dateien an den Dateisystemspeicherorten `FSCommon` (`/shared/common/file.txt`) und `FS1` (`/shared/projects/project1/file.txt`) sind nicht in der Liste enthalten. Dies liegt daran, dass sich diese Dateisystemspeicherorte `SHARED` im `WSAll` Speicherprofil und beide in der Liste der erforderlichen Dateisystemspeicherorte in der Warteschlange befinden`Q1`.

Sie können die Dateisystemspeicherorte sehen, die `SHARED` für einen Job in Betracht gezogen werden, der mit einem bestimmten Speicherprofil im Rahmen des GetStorageProfileForQueue Vorgangs eingereicht wird. `Q1`Verwenden Sie den folgenden Befehl, um das Speicherprofil `WSAll` für die Warteschlange abzufragen:

```
aws deadline get-storage-profile --farm-id $FARM_ID --storage-profile-id $WSALL_ID

aws deadline get-storage-profile-for-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID --
storage-profile-id $WSALL_ID
```

Wie finden Jobs Eingabedateien für Jobanhänge

Damit ein Job die Dateien verwenden kann, die Deadline Cloud mithilfe von Job-Anhängen auf Amazon S3 hochlädt, benötigt Ihr Job diese Dateien, die über das Dateisystem auf den Worker-Hosts verfügbar sind. Wenn eine Sitzung für Ihren Job auf einem Worker-Host läuft, lädt Deadline Cloud die Eingabedateien für den Job in ein temporäres Verzeichnis auf dem lokalen Laufwerk des Worker-Hosts herunter und fügt Pfadzuordnungsregeln für jeden Root-Pfad des Jobs zu seinem Dateisystem-Speicherort auf dem lokalen Laufwerk hinzu.

Starten Sie für dieses Beispiel den Deadline Cloud-Worker-Agent auf einer CloudShell AWS-Tab. Lassen Sie alle zuvor eingereichten Jobs fertig laufen und löschen Sie dann die Job-Logs aus dem Logs-Verzeichnis:

```
rm -rf ~/devdemo-logs/queue-*
```

Das folgende Skript ändert das Job-Bundle so, dass es alle Dateien im temporären Arbeitsverzeichnis der Sitzung und den Inhalt der Datei mit den Pfadzuordnungsregeln anzeigt, und sendet dann einen Job mit dem geänderten Paket:

```
# Change the value of FARM_ID to your farm's identifier
```

```
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

deadline config set settings.storage_profile_id $WSALL_ID

cat > ~/job_attachments_devguide/script.sh << EOF
#!/bin/bash

echo "Session working directory is: \$(pwd)"
echo
echo "Contents:"
find . -type f
echo
echo "Path mapping rules file: \$1"
jq . \$1
EOF

cat > ~/job_attachments_devguide/template.yaml << EOF
specificationVersion: jobtemplate-2023-09
name: "Job Attachments Explorer"
parameterDefinitions:
- name: ScriptFile
  type: PATH
  default: script.sh
  dataFlow: IN
  objectType: FILE
steps:
- name: Step
  script:
    actions:
      onRun:
        command: /bin/bash
        args:
        - "{{Param.ScriptFile}}"
        - "{{Session.PathMappingRulesFile}}"
EOF

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID
 job_attachments_devguide/
```

Sie können sich das Protokoll der Ausführung des Jobs ansehen, nachdem er vom Worker in Ihrer AWS CloudShell Umgebung ausgeführt wurde:

```
cat demoenv-logs/queue-*/session*.log
```

Aus dem Protokoll geht hervor, dass in der Sitzung als Erstes die beiden Eingabedateien für den Job auf den Worker heruntergeladen werden:

```
2024-07-17 01:26:37,824 INFO ================================================
2024-07-17 01:26:37,825 INFO --------- Job Attachments Download for Job
2024-07-17 01:26:37,825 INFO ================================================
2024-07-17 01:26:37,825 INFO Syncing inputs using Job Attachments
2024-07-17 01:26:38,116 INFO Downloaded 142.0 B / 186.0 B of 2 files (Transfer rate:
 0.0 B/s)
2024-07-17 01:26:38,174 INFO Downloaded 186.0 B / 186.0 B of 2 files (Transfer rate:
 733.0 B/s)
2024-07-17 01:26:38,176 INFO Summary Statistics for file downloads:
Processed 2 files totaling 186.0 B.
Skipped re-processing 0 files totaling 0.0 B.
Total processing time of 0.09752 seconds at 1.91 KB/s.
```

Als Nächstes folgt die Ausgabe von `script.sh` run by the job:

- Die Eingabedateien, die beim Absenden des Jobs hochgeladen wurden, befinden sich in einem Verzeichnis, dessen Name mit „assetroot“ beginnt, im temporären Verzeichnis der Sitzung.
- Die Pfade der Eingabedateien wurden relativ zum Verzeichnis „assetroot“ verschoben und nicht relativ zum Stammpfad für das Eingabemanifest des Jobs (). `"/"`
- Die Datei mit den Regeln für die Pfadzuweisung enthält eine zusätzliche Regel, die dem absoluten Pfad des `"/"` Verzeichnisses „assetroot“ neu zugeordnet wird.

Beispiel:

```
2024-07-17 01:26:38,264 INFO Output:
2024-07-17 01:26:38,267 INFO Session working directory is: /sessions/session-5b33f
2024-07-17 01:26:38,267 INFO
2024-07-17 01:26:38,267 INFO Contents:
2024-07-17 01:26:38,269 INFO ./tmp_xdhbsdo.sh
2024-07-17 01:26:38,269 INFO ./tmpdi00052b.json
2024-07-17 01:26:38,269 INFO ./assetroot-assetroot-3751a/shared/projects/project2/
file.txt
```

```
2024-07-17 01:26:38,269 INFO ./assetroot-assetroot-3751a/home/cloudshell-user/
job_attachments_devguide/script.sh
2024-07-17 01:26:38,269 INFO
2024-07-17 01:26:38,270 INFO Path mapping rules file: /sessions/session-5b33f/
tmpdi00052b.json
2024-07-17 01:26:38,282 INFO {
2024-07-17 01:26:38,282 INFO   "version": "pathmapping-1.0",
2024-07-17 01:26:38,282 INFO   "path_mapping_rules": [
2024-07-17 01:26:38,282 INFO     {
2024-07-17 01:26:38,282 INFO       "source_path_format": "POSIX",
2024-07-17 01:26:38,282 INFO       "source_path": "/shared/projects/project1",
2024-07-17 01:26:38,283 INFO       "destination_path": "/mnt/projects/project1"
2024-07-17 01:26:38,283 INFO     },
2024-07-17 01:26:38,283 INFO     {
2024-07-17 01:26:38,283 INFO       "source_path_format": "POSIX",
2024-07-17 01:26:38,283 INFO       "source_path": "/shared/common",
2024-07-17 01:26:38,283 INFO       "destination_path": "/mnt/common"
2024-07-17 01:26:38,283 INFO     },
2024-07-17 01:26:38,283 INFO     {
2024-07-17 01:26:38,283 INFO       "source_path_format": "POSIX",
2024-07-17 01:26:38,283 INFO       "source_path": "/",
2024-07-17 01:26:38,283 INFO       "destination_path": "/sessions/session-5b33f/
assetroot-assetroot-3751a"
2024-07-17 01:26:38,283 INFO     }
2024-07-17 01:26:38,283 INFO   ]
2024-07-17 01:26:38,283 INFO }
```

> **Note**
>
> Wenn der Job, den Sie einreichen, mehrere Manifeste mit unterschiedlichen Stammpfaden enthält, gibt es für jeden Stammpfad ein anderes Verzeichnis mit dem Namen „assetroot“.

Wenn Sie auf den verschobenen Dateisystemspeicherort einer Ihrer Eingabedateien, Verzeichnisse oder Dateisystemverzeichnisse verweisen müssen, können Sie entweder die Datei mit den Pfadzuordnungsregeln in Ihrem Job verarbeiten und die Neuzuordnung selbst durchführen oder der Jobvorlage in Ihrem Job-Bundle einen `PATH` Typ-Job-Parameter hinzufügen und den Wert, den Sie neu zuordnen müssen, als Wert dieses Parameters übergeben. Im folgenden Beispiel wird das Auftragspaket so geändert, dass es einen dieser Auftragsparameter hat, und sendet dann einen Job mit dem Speicherort des Dateisystems `/shared/projects/project2` als Wert:

```
cat > ~/job_attachments_devguide/template.yaml << EOF
specificationVersion: jobtemplate-2023-09
name: "Job Attachments Explorer"
parameterDefinitions:
- name: LocationToRemap
  type: PATH
steps:
- name: Step
  script:
    actions:
      onRun:
        command: /bin/echo
        args:
        - "The location of {{RawParam.LocationToRemap}} in the session is
 {{Param.LocationToRemap}}"
EOF

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID
 job_attachments_devguide/ \
  -p LocationToRemap=/shared/projects/project2
```

Die Protokolldatei für die Ausführung dieses Jobs enthält seine Ausgabe:

```
2024-07-17 01:40:35,283 INFO Output:
2024-07-17 01:40:35,284 INFO The location of /shared/projects/project2 in the session
 is /sessions/session-5b33f/assetroot-assetroot-3751a
```

Ausgabedateien von einem Job abrufen

Dieses Beispiel zeigt, wie Deadline Cloud die von Ihren Jobs generierten Ausgabedateien identifiziert, entscheidet, ob diese Dateien auf Amazon S3 hochgeladen werden sollen, und wie Sie diese Ausgabedateien auf Ihre Workstation übertragen können.

Verwenden Sie für dieses Beispiel das `job_attachments_devguide_output` Job-Bundle anstelle des `job_attachments_devguide` Job-Bundles. Erstellen Sie zunächst eine Kopie des Bundles in Ihrer AWS CloudShell Umgebung aus Ihrem Klon des Deadline GitHub Cloud-Beispiel-Repositorys:

```
cp -r deadline-cloud-samples/job_bundles/job_attachments_devguide_output ~/
```

Der wichtige Unterschied zwischen diesem Job-Bundle und dem `job_attachments_devguide` Job-Bundle besteht darin, dass der Jobvorlage ein neuer Job-Parameter hinzugefügt wurde:

```
...
parameterDefinitions:
...
- name: OutputDir
  type: PATH
  objectType: DIRECTORY
  dataFlow: OUT
  default: ./output_dir
  description: This directory contains the output for all steps.
...
```

Die `dataFlow` Eigenschaft des Parameters hat den Wert`OUT`. Deadline Cloud verwendet den Wert von `dataFlow` Jobparametern mit einem Wert von `OUT` oder `INOUT` als Ausgabe Ihres Jobs. Wenn der Dateisystemspeicherort, der als Wert an diese Art von Jobparametern übergeben wurde, einem lokalen Dateisystemspeicherort auf dem Worker, der den Job ausführt, neu zugeordnet wird, sucht Deadline Cloud an dem Speicherort nach neuen Dateien und lädt diese als Jobausgaben auf Amazon S3 hoch.

Um zu sehen, wie das funktioniert, starten Sie zunächst den Deadline Cloud-Worker-Agent in einem AWS CloudShell Tab. Lassen Sie alle zuvor eingereichten Jobs die Ausführung beenden. Löschen Sie anschließend die Jobprotokolle aus dem Protokollverzeichnis:

```
rm -rf ~/devdemo-logs/queue-*
```

Reichen Sie als Nächstes einen Job mit diesem Job-Bundle ein. Nachdem der Worker in Ihren CloudShell Läufen ausgeführt wurde, schauen Sie sich die Logs an:

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

deadline config set settings.storage_profile_id $WSALL_ID

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID ./
job_attachments_devguide_output
```

Das Protokoll zeigt, dass eine Datei als Ausgabe erkannt und auf Amazon S3 hochgeladen wurde:

```
2024-07-17 02:13:10,873 INFO ------------------------------------------
2024-07-17 02:13:10,873 INFO Uploading output files to Job Attachments
2024-07-17 02:13:10,873 INFO ------------------------------------------
2024-07-17 02:13:10,873 INFO Started syncing outputs using Job Attachments
2024-07-17 02:13:10,955 INFO Found 1 file totaling 117.0 B in output directory: /
sessions/session-7efa/assetroot-assetroot-3751a/output_dir
2024-07-17 02:13:10,956 INFO Uploading output manifest to
 DeadlineCloud/Manifests/farm-0011/queue-2233/job-4455/step-6677/
task-6677-0/2024-07-17T02:13:10.835545Z_sessionaction-8899-1/
c6808439dfc59f86763aff5b07b9a76c_output
2024-07-17 02:13:10,988 INFO Uploading 1 output file to S3: s3BucketName/DeadlineCloud/
Data
2024-07-17 02:13:11,011 INFO Uploaded 117.0 B / 117.0 B of 1 file (Transfer rate: 0.0
 B/s)
2024-07-17 02:13:11,011 INFO Summary Statistics for file uploads:
Processed 1 file totaling 117.0 B.
Skipped re-processing 0 files totaling 0.0 B.
Total processing time of 0.02281 seconds at 5.13 KB/s.
```

Das Protokoll zeigt auch, dass Deadline Cloud ein neues Manifest-Objekt im Amazon S3 S3-Bucket erstellt hat, das für die Verwendung durch Jobanhänge in der Warteschlange konfiguriert istQ1. Der Name des Manifest-Objekts wird von der Farm, der Warteschlange, dem Job, dem Schritt, der Aufgabe, dem Zeitstempel und den `sessionaction` Bezeichnern der Aufgabe abgeleitet, die die Ausgabe generiert hat. Laden Sie diese Manifestdatei herunter, um zu sehen, wo Deadline Cloud die Ausgabedateien für diese Aufgabe abgelegt hat:

```
# The name of queue `Q1`'s job attachments S3 bucket
Q1_S3_BUCKET=$(
  aws deadline get-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
    --query 'jobAttachmentSettings.s3BucketName' | tr -d '"'
)

# Fill this in with the object name from your log
OBJECT_KEY="DeadlineCloud/Manifests/..."

aws s3 cp --quiet s3://$Q1_S3_BUCKET/$OBJECT_KEY /dev/stdout | jq .
```

Das Manifest sieht wie folgt aus:

```
{
  "hashAlg": "xxh128",
```

```
  "manifestVersion": "2023-03-03",
  "paths": [
    {
      "hash": "34178940e1ef9956db8ea7f7c97ed842",
      "mtime": 1721182390859777,
      "path": "output_dir/output.txt",
      "size": 117
    }
  ],
  "totalSize": 117
}
```

Dies zeigt, dass der Inhalt der Ausgabedatei auf die gleiche Weise in Amazon S3 gespeichert wird wie Jobeingabedateien. Ähnlich wie Eingabedateien wird die Ausgabedatei in S3 mit einem Objektnamen gespeichert, der den Hash der Datei und das Präfix enthält`DeadlineCloud/Data`.

```
$ aws s3 ls --recursive s3://$Q1_S3_BUCKET | grep 34178940e1ef9956db8ea7f7c97ed842
2024-07-17 02:13:11        117 DeadlineCloud/
Data/34178940e1ef9956db8ea7f7c97ed842.xxh128
```

Sie können die Ausgabe eines Jobs mit dem Deadline Cloud-Monitor oder der Deadline Cloud-CLI auf Ihre Workstation herunterladen:

```
deadline job download-output --farm-id $FARM_ID --queue-id $QUEUE1_ID --job-id $JOB_ID
```

Der Wert des `OutputDir` Job-Parameters im übermittelten Job ist`./output_dir`, sodass die Ausgabe in ein Verzeichnis heruntergeladen wird, das `output_dir` innerhalb des Job-Bundle-Verzeichnisses aufgerufen wird. Wenn Sie einen absoluten Pfad oder einen anderen relativen Speicherort als Wert für angegeben haben`OutputDir`, würden die Ausgabedateien stattdessen an diesen Speicherort heruntergeladen.

```
$ deadline job download-output --farm-id $FARM_ID --queue-id $QUEUE1_ID --job-id
 $JOB_ID
Downloading output from Job 'Job Attachments Explorer: Output'

Summary of files to download:
    /home/cloudshell-user/job_attachments_devguide_output/output_dir/output.txt (1
 file)
```

```
You are about to download files which may come from multiple root directories. Here are
 a list of the current root directories:
[0] /home/cloudshell-user/job_attachments_devguide_output
> Please enter the index of root directory to edit, y to proceed without changes, or n
 to cancel the download (0, y, n) [y]:

Downloading Outputs  [####################################]  100%
Download Summary:
    Downloaded 1 files totaling 117.0 B.
    Total download time of 0.14189 seconds at 824.0 B/s.
    Download locations (total file counts):
        /home/cloudshell-user/job_attachments_devguide_output (1 file)
```

Dateien aus einem Schritt in einem abhängigen Schritt verwenden

Dieses Beispiel zeigt, wie ein Schritt in einem Job auf die Ausgaben eines Schritts zugreifen kann, von dem er im selben Job abhängt.

Um die Ergebnisse eines Schritts für einen anderen verfügbar zu machen, fügt Deadline Cloud einer Sitzung zusätzliche Aktionen hinzu, um diese Ausgaben herunterzuladen, bevor Aufgaben in der Sitzung ausgeführt werden. Sie teilen ihr mit, aus welchen Schritten die Ausgaben heruntergeladen werden sollen, indem Sie diese Schritte als Abhängigkeiten des Schritts deklarieren, der die Ausgaben verwenden muss.

Verwenden Sie das `job_attachments_devguide_output` Job-Bundle für dieses Beispiel. Erstellen Sie zunächst in Ihrer AWS CloudShell Umgebung eine Kopie von Ihrem Klon des Deadline GitHub Cloud-Beispiel-Repositorys. Ändern Sie ihn, um einen abhängigen Schritt hinzuzufügen, der erst nach dem vorhandenen Schritt ausgeführt wird und die Ausgabe dieses Schritts verwendet:

```
cp -r deadline-cloud-samples/job_bundles/job_attachments_devguide_output ~/

cat >> job_attachments_devguide_output/template.yaml << EOF
- name: DependentStep
  dependencies:
  - dependsOn: Step
  script:
    actions:
      onRun:
        command: /bin/cat
        args:
        - "{{Param.OutputDir}}/output.txt"
```

```
EOF
```

Der mit diesem modifizierten Auftragspaket erstellte Job wird in zwei separaten Sitzungen ausgeführt, eine für die Aufgabe im Schritt „Schritt“ und dann eine zweite für die Aufgabe im Schritt "DependentStep“.

Starten Sie zunächst den Deadline Cloud-Worker-Agent in einem CloudShell Tab. Lassen Sie alle zuvor eingereichten Jobs fertig laufen und löschen Sie dann die Job-Logs aus dem Logs-Verzeichnis:

```
rm -rf ~/devdemo-logs/queue-*
```

Als Nächstes reichen Sie einen Job mit dem geänderten `job_attachments_devguide_output` Auftragspaket ein. Warten Sie, bis die Ausführung auf dem Worker in Ihrer CloudShell Umgebung abgeschlossen ist. Sehen Sie sich die Protokolle der beiden Sitzungen an:

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

deadline config set settings.storage_profile_id $WSALL_ID

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID ./
job_attachments_devguide_output

# Wait for the job to finish running, and then:

cat demoenv-logs/queue-*/session-*
```

Im Sitzungsprotokoll für die Aufgabe im genannten `DependentStep` Schritt werden zwei separate Download-Aktionen ausgeführt:

```
2024-07-17 02:52:05,666 INFO ================================================
2024-07-17 02:52:05,666 INFO --------- Job Attachments Download for Job
2024-07-17 02:52:05,667 INFO ================================================
2024-07-17 02:52:05,667 INFO Syncing inputs using Job Attachments
2024-07-17 02:52:05,928 INFO Downloaded 207.0 B / 207.0 B of 1 file (Transfer rate: 0.0
 B/s)
```

```
2024-07-17 02:52:05,929 INFO Summary Statistics for file downloads:
Processed 1 file totaling 207.0 B.
Skipped re-processing 0 files totaling 0.0 B.
Total processing time of 0.03954 seconds at 5.23 KB/s.

2024-07-17 02:52:05,979 INFO
2024-07-17 02:52:05,979 INFO ================================================
2024-07-17 02:52:05,979 INFO --------- Job Attachments Download for Step
2024-07-17 02:52:05,979 INFO ================================================
2024-07-17 02:52:05,980 INFO Syncing inputs using Job Attachments
2024-07-17 02:52:06,133 INFO Downloaded 117.0 B / 117.0 B of 1 file (Transfer rate: 0.0
 B/s)
2024-07-17 02:52:06,134 INFO Summary Statistics for file downloads:
Processed 1 file totaling 117.0 B.
Skipped re-processing 0 files totaling 0.0 B.
Total processing time of 0.03227 seconds at 3.62 KB/s.
```

Bei der ersten Aktion wird die `script.sh` Datei heruntergeladen, die für den Schritt mit dem Namen „Schritt“ verwendet wurde. Bei der zweiten Aktion werden die Ausgaben aus diesem Schritt heruntergeladen. Deadline Cloud bestimmt, welche Dateien heruntergeladen werden sollen, indem das in diesem Schritt generierte Ausgabemanifest als Eingabemanifest verwendet wird.

Später im selben Protokoll können Sie die Ausgabe des Schritts "DependentStep„sehen:

```
2024-07-17 02:52:06,213 INFO Output:
2024-07-17 02:52:06,216 INFO Script location: /sessions/session-5b33f/
assetroot-assetroot-3751a/script.sh
```

Ressourcenlimits für Jobs erstellen

Jobs, die an Deadline Cloud übermittelt werden, können von Ressourcen abhängen, die von mehreren Jobs gemeinsam genutzt werden. Beispielsweise kann eine Farm mehr Mitarbeiter als variable Lizenzen für eine bestimmte Ressource haben. Oder ein gemeinsam genutzter Dateiserver kann möglicherweise nur einer begrenzten Anzahl von Workern gleichzeitig Daten bereitstellen. In einigen Fällen können ein oder mehrere Jobs all diese Ressourcen beanspruchen, was aufgrund nicht verfügbarer Ressourcen zu Fehlern führt, wenn neue Mitarbeiter eingestellt werden.

Um dieses Problem zu lösen, können Sie Grenzwerte für diese begrenzten Ressourcen verwenden. Deadline Cloud berücksichtigt die Verfügbarkeit eingeschränkter Ressourcen und verwendet diese Informationen, um sicherzustellen, dass Ressourcen verfügbar sind, wenn neue Mitarbeiter ihre

Arbeit aufnehmen, sodass die Wahrscheinlichkeit geringer ist, dass Jobs aufgrund nicht verfügbarer Ressourcen ausfallen.

Grenzwerte werden für die gesamte Farm erstellt. Für Aufträge, die an eine Warteschlange gesendet werden, können nur Limits gelten, die der Warteschlange zugeordnet sind. Wenn Sie ein Limit für einen Job angeben, der nicht mit der Warteschlange verknüpft ist, ist der Job nicht kompatibel und kann nicht ausgeführt werden.

Um ein Limit zu verwenden, müssen Sie

- Erstellen Sie ein Limit
- Ordnen Sie ein Limit und einer Warteschlange zu
- Reichen Sie einen Job ein, der Limits erfordert

> **Note**
>
> Wenn Sie einen Job mit eingeschränkten Ressourcen in einer Warteschlange ausführen, der kein Limit zugeordnet ist, kann dieser Job alle Ressourcen verbrauchen. Wenn Sie über eine eingeschränkte Ressource verfügen, stellen Sie sicher, dass alle Schritte in Aufträgen in Warteschlangen, die die Ressource verwenden, mit einem Limit verknüpft sind.

Bei Grenzwerten, die in einer Farm definiert, einer Warteschlange zugeordnet und in einem Job angegeben sind, kann eines von vier Dingen passieren:

- Wenn Sie ein Limit erstellen, es einer Warteschlange zuordnen und das Limit in der Vorlage eines Jobs angeben, wird der Job ausgeführt und verwendet nur die im Limit definierten Ressourcen.
- Wenn Sie ein Limit erstellen, es in einer Jobvorlage angeben, das Limit aber keiner Warteschlange zuordnen, wird der Job als inkompatibel markiert und kann nicht ausgeführt werden.
- Wenn Sie ein Limit erstellen, es keiner Warteschlange zuordnen und das Limit nicht in der Vorlage eines Jobs angeben, wird der Job ausgeführt, verwendet das Limit aber nicht.
- Wenn Sie überhaupt kein Limit verwenden, wird der Job ausgeführt.

Wenn Sie mehreren Warteschlangen ein Limit zuordnen, teilen sich die Warteschlangen die Ressourcen, für die das Limit gilt. Wenn Sie beispielsweise ein Limit von 100 erstellen und eine Warteschlange 60 Ressourcen verwendet, können andere Warteschlangen nur 40 Ressourcen

verwenden. Wenn eine Ressource freigegeben wird, kann sie von einer Aufgabe aus einer beliebigen Warteschlange übernommen werden.

Deadline Cloud bietet zwei AWS CloudFormation Messwerte, mit denen Sie die durch ein Limit bereitgestellten Ressourcen überwachen können. Sie können die aktuelle Anzahl der verwendeten Ressourcen und die maximale Anzahl der Ressourcen, die im Rahmen des Limits verfügbar sind, überwachen. Weitere Informationen finden Sie unter Kennzahlen zum Ressourcenlimit im Deadline Cloud Developer Guide.

Sie wenden ein Limit auf einen Auftragsschritt in einer Jobvorlage an. Wenn Sie im `amounts` Abschnitt eines Schritts den Namen der Mengenanforderung für ein Limit angeben und der Warteschlange des Jobs ein Limit zugeordnet `amountRequirementName` wird, werden die für diesen Schritt geplanten Aufgaben durch das Limit für die Ressource eingeschränkt. `hostRequirements`

Wenn für einen Schritt eine Ressource erforderlich ist, die durch ein erreichtes Limit eingeschränkt ist, werden die Aufgaben in diesem Schritt nicht von weiteren Mitarbeitern übernommen.

Sie können mehr als ein Limit auf einen Arbeitsschritt anwenden. Wenn in diesem Schritt beispielsweise zwei verschiedene Softwarelizenzen verwendet werden, können Sie für jede Lizenz ein eigenes Limit festlegen. Wenn für einen Schritt zwei Grenzwerte erforderlich sind und das Limit für eine der Ressourcen erreicht ist, werden Aufgaben in diesem Schritt nicht von weiteren Mitarbeitern übernommen, bis die Ressourcen verfügbar sind.

Beenden und Löschen von Grenzwerten

Wenn Sie die Zuordnung zwischen einer Warteschlange und einem Limit beenden oder löschen, beendet ein Job, der das Limit verwendet, die Planung von Aufgaben für Schritte, die dieses Limit erfordern, und blockiert die Erstellung neuer Sitzungen für einen Schritt.

Aufgaben, die sich im Status BEREIT befinden, bleiben bereit, und Aufgaben werden automatisch fortgesetzt, sobald die Verbindung zwischen der Warteschlange und dem Limit wieder aktiv wird. Sie müssen keine Jobs in die Warteschlange stellen.

Wenn Sie die Zuordnung zwischen einer Warteschlange und einem Limit beenden oder löschen, haben Sie zwei Möglichkeiten, die Ausführung von Aufgaben zu beenden:

- Aufgaben beenden und stornieren — Mitarbeiter mit Sitzungen, die das Limit erreicht haben, stornieren alle Aufgaben.

- Ausführen von Aufgaben beenden und beenden — Mitarbeiter mit Sitzungen, die das Limit erreicht haben, erledigen ihre Aufgaben.

Wenn Sie ein Limit über die Konsole löschen, beenden die Mitarbeiter zunächst die Ausführung von Aufgaben sofort oder erst, wenn sie abgeschlossen sind. Wenn die Zuordnung gelöscht wird, passiert Folgendes:

- Schritte, für die das Limit erforderlich ist, sind als nicht kompatibel gekennzeichnet.
- Der gesamte Job, der diese Schritte enthält, wird abgebrochen, einschließlich der Schritte, für die das Limit nicht erforderlich ist.
- Der Job ist als nicht kompatibel markiert.

Wenn der Warteschlange, die dem Limit zugeordnet ist, eine Flotte mit einer Flottenkapazität zugeordnet ist, die dem Mengenanforderungsnamen des Limits entspricht, verarbeitet diese Flotte weiterhin Aufträge mit dem angegebenen Limit.

Erstellen Sie ein Limit

Sie erstellen ein Limit mit der Deadline Cloud-Konsole oder dem CreateLimit Vorgang in der Deadline Cloud-API. Limits sind für eine Farm definiert, aber mit Warteschlangen verknüpft. Nachdem Sie ein Limit erstellt haben, können Sie es einer oder mehreren Warteschlangen zuordnen.

Um ein Limit zu erstellen

1. Wählen Sie im Dashboard der Deadline Cloud-Konsole (Deadline Cloud-Konsole) die Farm aus, für die Sie eine Warteschlange erstellen möchten.
2. Wählen Sie die Farm aus, zu der Sie das Limit hinzufügen möchten, wählen Sie den Tab Limits und dann Limit erstellen aus.
3. Geben Sie die Details für das Limit ein. Der Name der Mengenanforderung ist der Name, der in der Jobvorlage verwendet wird, um das Limit zu identifizieren. Er muss mit dem Präfix beginnen, **`amount.`** gefolgt von der Bezeichnung des Betrags. Der Name der Mengenanforderung muss in den Warteschlangen, die mit dem Limit verknüpft sind, eindeutig sein.
4. Wenn Sie „Höchstbetrag festlegen“ wählen, entspricht dies der Gesamtanzahl der Ressourcen, die nach diesem Limit zulässig sind. Wenn Sie „Kein Höchstbetrag“ wählen, ist die Ressourcennutzung nicht begrenzt. Auch wenn die Ressourcennutzung nicht begrenzt ist, wird die `CurrentCount` CloudWatch Amazon-Metrik ausgegeben, sodass Sie die Nutzung

verfolgen können. Weitere Informationen finden Sie unter CloudWatchMetriken im Deadline Cloud Developer Guide.

5. Wenn Sie bereits wissen, für welche Warteschlangen das Limit verwendet werden soll, können Sie sie jetzt auswählen. Sie müssen keine Warteschlange zuordnen, um ein Limit zu erstellen.
6. Wählen Sie Limit erstellen aus.

Ordnen Sie ein Limit und einer Warteschlange zu

Nachdem Sie ein Limit erstellt haben, können Sie dem Limit eine oder mehrere Warteschlangen zuordnen. Nur Warteschlangen, die einem Grenzwert zugeordnet sind, verwenden die im Grenzwert angegebenen Werte.

Sie erstellen eine Zuordnung zu einer Warteschlange mithilfe der Deadline Cloud-Konsole oder des CreateQueueLimitAssociation Vorgangs in der Deadline Cloud-API.

Um eine Warteschlange mit einem Limit zu verknüpfen

1. Wählen Sie im Dashboard der Deadline Cloud-Konsole (Deadline Cloud-Konsole) die Farm aus, der Sie ein Limit mit einer Warteschlange verknüpfen möchten.
2. Wählen Sie den Tab Limits, wählen Sie das Limit aus, dem eine Warteschlange zugeordnet werden soll, und wählen Sie dann Limit bearbeiten aus.
3. Wählen Sie im Abschnitt Warteschlangen zuordnen die Warteschlangen aus, die dem Limit zugeordnet werden sollen.
4. Wählen Sie Änderungen speichern aus.

Reichen Sie einen Job ein, der Limits erfordert

Sie wenden ein Limit an, indem Sie es als Hostanforderung für den Job oder Job-Schritt angeben. Wenn Sie in einem Schritt kein Limit angeben und dieser Schritt eine zugeordnete Ressource verwendet, wird die Nutzung des Schritts nicht auf das Limit angerechnet, wenn Jobs geplant werden.

Bei einigen Deadline Cloud-Einreichern können Sie eine Hostanforderung festlegen. Sie können den Namen des Limits im Absender angeben, um das Limit anzuwenden.

Wenn Ihr Einreicher das Hinzufügen von Hostanforderungen nicht unterstützt, können Sie auch ein Limit festlegen, indem Sie die Jobvorlage für den Job bearbeiten.

Um ein Limit auf einen Job-Schritt im Job-Bundle anzuwenden

1. Öffnen Sie die Jobvorlage für den Job mit einem Texteditor. Die Jobvorlage befindet sich im Job-Bundle-Verzeichnis für den Job. Weitere Informationen finden Sie unter Job-Pakete im Deadline Cloud Developer Guide.
2. Suchen Sie die Schrittdefinition für den Schritt, auf den das Limit angewendet werden soll.
3. Fügen Sie der Schrittdefinition Folgendes hinzu. *amount.name*Ersetzen Sie es durch den Namen der Mengenanforderung für Ihr Limit. Für den typischen Gebrauch sollten Sie den `min` Wert auf 1 setzen.

   YAML

   ```
   hostRequirements:
     amounts:
     - name: amount.name
       min: 1
   ```

   JSON

   ```
   "hostRequirements": {
       "amounts": [
           {
               "name": "amount.name",
               "min": "1"
           }
       ]
   }
   ```

   Sie können einem Auftragsschritt wie folgt mehrere Grenzwerte hinzufügen. Ersetzen Sie *amount.name_1* und *amount.name_2* durch die Namen der Mengenanforderungen Ihrer Limits.

   YAML

   ```
   hostRequirements:
     amounts:
     - name: amount.name_1
       min: 1
     - name: amount.name_2
   ```

```
      min: 1
```

JSON

```
"hostRequirements": {
    "amounts": [
        {
            "name": "amount.name_1",
            "min": "1"
        },
        {
            "name": "amount.name_2",
            "min": "1"
        }
    }
}
```

4. Speichern Sie die Änderungen an der Jobvorlage.

So reichen Sie einen Job bei Deadline Cloud ein

Es gibt viele verschiedene Möglichkeiten, Jobs bei AWS Deadline Cloud einzureichen. In diesem Abschnitt werden einige der Möglichkeiten beschrieben, wie Sie Jobs mithilfe der von Deadline Cloud bereitgestellten Tools oder durch die Erstellung eigener benutzerdefinierter Tools für Ihre Workloads einreichen können.

- Von einem Terminal aus — wenn Sie zum ersten Mal ein Jobpaket entwickeln oder wenn Benutzer, die einen Job einreichen, sich mit der Befehlszeile auskennen
- Aus einem Skript — zur Anpassung und Automatisierung von Workloads
- Aus einer Anwendung — wenn die Arbeit des Benutzers in einer Anwendung stattfindet oder wenn der Kontext einer Anwendung wichtig ist.

In den folgenden Beispielen werden die `deadline` Python-Bibliothek und das `deadline` Befehlszeilentool verwendet. Beide sind verfügbar PyPiund werden dort gehostet GitHub.

Themen

- Senden Sie einen Job von einem Terminal aus an Deadline Cloud
- Senden Sie einen Job mithilfe eines Skripts an Deadline Cloud

- Reichen Sie eine Stelle innerhalb einer Bewerbung ein

Senden Sie einen Job von einem Terminal aus an Deadline Cloud

Wenn Sie nur ein Job-Bundle und die Deadline Cloud-CLI verwenden, können Sie oder Ihre technisch versierten Benutzer schnell das Schreiben von Jobpaketen wiederholen, um das Einreichen eines Jobs zu testen. Verwenden Sie den folgenden Befehl, um ein Job-Bundle einzureichen:

```
deadline bundle submit <path-to-job-bundle>
```

Wenn Sie ein Job-Bundle mit Parametern einreichen, für die das Paket keine Standardwerte enthält, können Sie diese mit der `--parameter` Option`-p`/angeben.

```
deadline bundle submit <path-to-job-bundle> -p <parameter-name>=<parameter-value> -
p ...
```

Führen Sie den Befehl help aus, um eine vollständige Liste der verfügbaren Optionen zu erhalten:

```
deadline bundle submit --help
```

Senden Sie einen Job über eine GUI an Deadline Cloud

Die Deadline Cloud CLI verfügt außerdem über eine grafische Benutzeroberfläche, über die Benutzer die Parameter sehen können, die sie angeben müssen, bevor sie einen Job einreichen. Wenn Ihre Benutzer es vorziehen, nicht mit der Befehlszeile zu interagieren, können Sie eine Desktop-Verknüpfung schreiben, die einen Dialog zum Einreichen eines bestimmten Job-Bundles öffnet:

```
deadline bundle gui-submit <path-to-job-bundle>
```

Verwenden Sie die `--browse` Option can, damit der Benutzer ein Jobpaket auswählen kann:

```
deadline bundle gui-submit --browse
```

Eine vollständige Liste der verfügbaren Optionen erhalten Sie, wenn Sie den Befehl help ausführen:

```
deadline bundle gui-submit --help
```

Senden Sie einen Job mithilfe eines Skripts an Deadline Cloud

Um das Senden von Jobs an Deadline Cloud zu automatisieren, können Sie sie mithilfe von Tools wie Bash, Powershell und Batch-Dateien skripten.

Sie können Funktionen wie das Auffüllen von Jobparametern aus Umgebungsvariablen oder anderen Anwendungen hinzufügen. Sie können auch mehrere Jobs hintereinander einreichen oder die Erstellung eines Auftragspakets per Skript abschicken.

Einen Job mit Python einreichen

Deadline Cloud verfügt auch über eine Open-Source-Python-Bibliothek für die Interaktion mit dem Dienst. Der Quellcode ist verfügbar auf GitHub.

Die Bibliothek ist auf pypi über pip () verfügbar. `pip install deadline` Es ist dieselbe Bibliothek, die vom Deadline Cloud CLI-Tool verwendet wird:

```
from deadline.client import api

job_bundle_path = "/path/to/job/bundle"
job_parameters = [
    {
        "name": "parameter_name",
        "value": "parameter_value"
    },
]

job_id = api.create_job_from_job_bundle(
    job_bundle_path,
    job_parameters
)
print(job_id)
```

Um einen Dialog wie den `deadline bundle gui-submit` Befehl zu erstellen, können Sie die `show_job_bundle_submitter` Funktion von verwenden `deadline.client.ui.job_bundle_submitter`.

Das folgende Beispiel startet eine Qt-Anwendung und zeigt den Job Bundle Submitter:

```
# The GUI components must be installed with pip install "deadline[gui]"
import sys
from qtpy.QtWidgets import QApplication
```

```
from deadline.client.ui.job_bundle_submitter import show_job_bundle_submitter

app = QApplication(sys.argv)
submitter = show_job_bundle_submitter(browse=True)
submitter.show()
app.exec()
print(submitter.create_job_response)
```

Um Ihren eigenen Dialog zu erstellen, können Sie die `SubmitJobToDeadlineDialog` Klasse in verwenden. `deadline.client.ui.dialogs.submit_job_to_deadline_dialog` Sie können Werte übergeben, Ihren eigenen auftragsspezifischen Tab einbetten und festlegen, wie das Job-Bundle erstellt (oder übergeben) wird.

Reichen Sie eine Stelle innerhalb einer Bewerbung ein

Um Benutzern das Einreichen von Jobs zu erleichtern, können Sie die von einer Anwendung bereitgestellten Skriptlaufzeiten oder Plugin-Systeme verwenden. Die Benutzer haben eine vertraute Oberfläche, und Sie können leistungsstarke Tools erstellen, die die Benutzer beim Einreichen eines Workloads unterstützen.

Job-Bundles in eine Anwendung einbetten

Dieses Beispiel zeigt das Einreichen von Auftragspaketen, die Sie in der Anwendung zur Verfügung stellen.

Um einem Benutzer Zugriff auf diese Job-Bundles zu gewähren, erstellen Sie ein Skript, das in ein Menüelement eingebettet ist, das die Deadline Cloud-CLI startet.

Das folgende Skript ermöglicht es einem Benutzer, das Job-Bundle auszuwählen:

```
deadline bundle gui-submit --install-gui
```

Um stattdessen ein bestimmtes Job-Bundle in einem Menüelement zu verwenden, verwenden Sie Folgendes:

```
deadline bundle gui-submit </path/to/job/bundle> --install-gui
```

Dadurch wird ein Dialogfeld geöffnet, in dem der Benutzer die Jobparameter, Eingaben und Ausgaben ändern und den Job dann weiterleiten kann. Sie können verschiedene Menüelemente für verschiedene Jobpakete einrichten, die ein Benutzer in einer Bewerbung einreichen kann.

Wenn der Job, den Sie mit einem Job-Bundle einreichen, bei allen Einreichungen ähnliche Parameter und Ressourcenreferenzen enthält, können Sie die Standardwerte in das zugrunde liegende Job-Bundle eingeben.

Holen Sie sich Informationen aus einer Bewerbung

Um Informationen aus einer Anwendung abzurufen, sodass Benutzer sie nicht manuell zur Einreichung hinzufügen müssen, können Sie Deadline Cloud in die Anwendung integrieren, sodass Ihre Benutzer Jobs über eine vertraute Oberfläche einreichen können, ohne die Anwendung beenden oder Befehlszeilentools verwenden zu müssen.

Wenn Ihre Anwendung über eine Scripting-Runtime verfügt, die Python und pyside/pyqt unterstützt, können Sie die GUI-Komponenten aus der Deadline Cloud-Clientbibliothek verwenden, um eine Benutzeroberfläche zu erstellen. Ein Beispiel finden Sie unter Integration von Deadline Cloud für Maya unter. GitHub

Die Deadline Cloud-Clientbibliothek bietet Operationen, die Folgendes tun, um Ihnen zu helfen, ein starkes integriertes Benutzererlebnis zu bieten:

- Rufen Sie die Umgebungsparameter, Jobparameter und Asset-Referenzen aus Umgebungsvariablen ab und rufen Sie das Anwendungs-SDK auf.
- Legen Sie die Parameter im Job-Bundle fest. Um zu vermeiden, dass das ursprüngliche Paket geändert wird, sollten Sie eine Kopie des Bundles erstellen und die Kopie einreichen.

Wenn Sie den `deadline bundle gui-submit` Befehl verwenden, um das Job-Bundle zu senden, müssen Sie die `parameter_values.yaml` und `asset_references.yaml` -Dateien programmgesteuert verwenden, um die Informationen aus der Anwendung zu übergeben. Weitere Informationen zu diesen Dateien finden Sie unter. Vorlagen für offene Stellenbeschreibungen (OpenJD) für Deadline Cloud

Wenn Sie komplexere Steuerelemente als die von OpenJD angebotenen benötigen, den Job vom Benutzer abstrahieren müssen oder die Integration an den visuellen Stil der Anwendung anpassen möchten, können Sie Ihren eigenen Dialog schreiben, der die Deadline Cloud-Clientbibliothek aufruft, um den Job einzureichen.

Jobs in Deadline Cloud planen

Nachdem ein Auftrag erstellt wurde, plant AWS Deadline Cloud, dass er in einer oder mehreren Flotten bearbeitet wird, die einer Warteschlange zugeordnet sind. Die Flotte, die eine bestimmte

Aufgabe bearbeitet, wird auf der Grundlage der für die Flotte konfigurierten Funktionen und der Hostanforderungen eines bestimmten Schritts ausgewählt.

Jobs in einer Warteschlange werden in der Reihenfolge der bestmöglichen Priorität geplant, von der höchsten zur niedrigsten Priorität. Wenn zwei Jobs dieselbe Priorität haben, wird der älteste Job zuerst geplant.

In den folgenden Abschnitten wird detailliert beschrieben, wie ein Job geplant wird.

Prüfen Sie die Flottenkompatibilität

Nachdem ein Job erstellt wurde, vergleicht Deadline Cloud die Hostanforderungen für jeden Schritt im Job mit den Fähigkeiten der Flotten, die mit der Warteschlange verknüpft sind, an die der Job übermittelt wurde. Wenn eine Flotte die Hostanforderungen erfüllt, wird der Job in den `READY` Status versetzt.

Wenn für einen Schritt des Jobs Anforderungen gelten, die von einer Flotte, die der Warteschlange zugeordnet ist, nicht erfüllt werden können, wird der Status des Schritts auf gesetzt`NOT_COMPATIBLE`. Außerdem werden die restlichen Schritte des Jobs storniert.

Die Funktionen für eine Flotte werden auf Flottenebene festgelegt. Selbst wenn ein Mitarbeiter in einer Flotte die Anforderungen des Auftrags erfüllt, werden ihm keine Aufgaben aus dem Auftrag zugewiesen, wenn seine Flotte die Anforderungen des Auftrags nicht erfüllt.

Die folgende Jobvorlage enthält einen Schritt, der die Hostanforderungen für den Schritt spezifiziert:

```
name: Sample Job With Host Requirements
specificationVersion: jobtemplate-2023-09
steps:
- name: Step 1
  script:
    actions:
      onRun:
        args:
        - '1'
        command: /usr/bin/sleep
  hostRequirements:
    amounts:
    # Capabilities starting with "amount." are amount capabilities. If they start with
 "amount.worker.",
    # they are defined by the OpenJD specification. Other names are free for custom
 usage.
```

```
    - name: amount.worker.vcpu
      min: 4
      max: 8
    attributes:
    - name: attr.worker.os.family
      anyOf:
      - linux
```

Dieser Job kann für eine Flotte mit den folgenden Funktionen geplant werden:

```
{
    "vCpuCount": {"min": 4, "max": 8},
    "memoryMiB": {"min": 1024},
    "osFamily": "linux",
    "cpuArchitectureType": "x86_64"
}
```

Dieser Job kann nicht für eine Flotte mit einer der folgenden Funktionen geplant werden:

```
{
    "vCpuCount": {"min": 4},
    "memoryMiB": {"min": 1024},
    "osFamily": "linux",
    "cpuArchitectureType": "x86_64"
}
    The vCpuCount has no maximum, so it exceeds the maximum vCPU host requirement.

{
    "vCpuCount": {"max": 8},
    "memoryMiB": {"min": 1024},
    "osFamily": "linux",
    "cpuArchitectureType": "x86_64"
}
    The vCpuCount has no minimum, so it doesn't satisfy the minimum vCPU host
 requirement.

{
    "vCpuCount": {"min": 4, "max": 8},
    "memoryMiB": {"min": 1024},
    "osFamily": "windows",
    "cpuArchitectureType": "x86_64"
}
```

```
    The osFamily doesn't match.
```

Skalierung der Flotte

Wenn ein Auftrag einer kompatiblen, servicemanagierten Flotte zugewiesen wird, wird die Flotte auto skaliert. Die Anzahl der Mitarbeiter in der Flotte ändert sich je nach der Anzahl der Aufgaben, die der Flotte zur Ausführung zur Verfügung stehen.

Wenn ein Auftrag einer vom Kunden verwalteten Flotte zugewiesen wird, sind Mitarbeiter möglicherweise bereits vorhanden oder können mithilfe von ereignisbasierter Autoskalierung erstellt werden. Weitere Informationen finden Sie unter Verwenden EventBridge zur Behandlung von Auto Scaling-Ereignissen im Amazon EC2 Auto Scaling Scaling-Benutzerhandbuch.

Sitzungen

Die Aufgaben in einem Job sind in eine oder mehrere Sitzungen aufgeteilt. Die Mitarbeiter führen die Sitzungen durch, um die Umgebung einzurichten, die Aufgaben auszuführen und dann die Umgebung zu zerstören. Jede Sitzung besteht aus einer oder mehreren Aktionen, die ein Mitarbeiter ausführen muss.

Wenn ein Mitarbeiter Abschnittsaktionen abschließt, können zusätzliche Sitzungsaktionen an den Mitarbeiter gesendet werden. Der Mitarbeiter verwendet in der Sitzung vorhandene Umgebungen und Jobanhänge wieder, um Aufgaben effizienter zu erledigen.

Bei dienstverwalteten Flottenarbeitern werden Sitzungsverzeichnisse nach dem Ende der Sitzung gelöscht, andere Verzeichnisse werden jedoch zwischen den Sitzungen beibehalten. Dieses Verhalten ermöglicht es Ihnen, Caching-Strategien für Daten zu implementieren, die in mehreren Sitzungen wiederverwendet werden können. Um Daten zwischen Sitzungen zwischenzuspeichern, speichern Sie sie im Home-Verzeichnis des Benutzers, der den Job ausführt. Beispielsweise werden Conda-Pakete im Home-Verzeichnis des Job-Benutzers unter `C:\Users\job-user\.conda-pkgs` on Windows workers und `/home/job-user/.conda-pkgs` on Linux workers zwischengespeichert. Diese Daten bleiben verfügbar, bis der Worker heruntergefahren wird.

Jobanhänge werden vom Einreicher erstellt, den Sie als Teil Ihres Deadline Cloud CLI-Jobpakets verwenden. Mit der `--attachments` Option für den Befehl können Sie auch Jobanhänge erstellen. `create-job` AWS CLI Umgebungen werden an zwei Stellen definiert: Warteschlangenumgebungen, die an eine bestimmte Warteschlange angehängt sind, und Job- und Schrittumgebungen, die in der Jobvorlage definiert sind.

Es gibt vier Arten von Sitzungsaktionen:

- `syncInputJobAttachments`— Lädt die Eingabe-Job-Anhänge an den Worker herunter.
- `envEnter`— Führt die `onEnter` Aktionen für eine Umgebung aus.
- `taskRun`— Führt die `onRun` Aktionen für eine Aufgabe aus.
- `envExit`— Führt die `onExit` Aktionen für eine Umgebung aus.

Die folgende Jobvorlage hat eine Schrittumgebung. Sie enthält eine `onEnter` Definition zum Einrichten der Schrittumgebung, eine `onRun` Definition, die die auszuführende Aufgabe definiert, und eine `onExit` Definition zum Abbau der Schrittumgebung. Die für diesen Job erstellten Sitzungen umfassen eine `envEnter` Aktion, eine oder mehrere `taskRun` Aktionen und dann eine `envExit` Aktion.

```
name: Sample Job with Maya Environment
specificationVersion: jobtemplate-2023-09
steps:
- name: Maya Step
  stepEnvironments:
  - name: Maya
    description: Runs Maya in the background.
    script:
      embeddedFiles:
      - name: initData
        filename: init-data.yaml
        type: TEXT
        data: |
          scene_file: MyAwesomeSceneFile
          renderer: arnold
          camera: persp
      actions:
        onEnter:
          command: MayaAdaptor
          args:
          - daemon
          - start
          - --init-data
          - file://{{Env.File.initData}}
        onExit:
          command: MayaAdaptor
          args:
          - daemon
```

```
        - stop
  parameterSpace:
    taskParameterDefinitions:
    - name: Frame
      range: 1-5
      type: INT
  script:
    embeddedFiles:
    - name: runData
      filename: run-data.yaml
      type: TEXT
      data: |
        frame: {{Task.Param.Frame}}
    actions:
      onRun:
        command: MayaAdaptor
        args:
        - daemon
        - run
        - --run-data
        - file://{{ Task.File.runData }}
```

Pipelining von Sitzungsaktionen

Durch das Pipelining von Sitzungsaktionen kann ein Scheduler einem Worker mehrere Sitzungsaktionen vorab zuweisen. Der Worker kann diese Aktionen dann sequenziell ausführen, wodurch die Leerlaufzeit zwischen Aufgaben reduziert oder ganz vermieden wird.

Um eine erste Zuweisung zu erstellen, erstellt der Scheduler eine Sitzung mit einer Aufgabe, der Worker erledigt die Aufgabe und anschließend analysiert der Scheduler die Aufgabendauer, um future Zuweisungen zu bestimmen.

Damit der Scheduler effektiv ist, gibt es Regeln für die Aufgabendauer. Für Aufgaben unter einer Minute verwendet der Scheduler ein Power-of-2-Wachstumsmuster. Bei einer 1-Sekunden-Aufgabe weist der Scheduler beispielsweise 2 neue Aufgaben zu, dann 4 und dann 8. Für Aufgaben, die länger als eine Minute dauern, weist der Scheduler nur eine neue Aufgabe zu, und das Pipelining bleibt deaktiviert.

Um die Pipeline-Größe zu berechnen, geht der Scheduler wie folgt vor:

- Verwendet die durchschnittliche Aufgabendauer abgeschlossener Aufgaben
- Zielt darauf ab, den Mitarbeiter eine Minute lang zu beschäftigen

- Berücksichtigt nur Aufgaben innerhalb derselben Sitzung
- Gibt Daten zur Dauer nicht an alle Mitarbeiter weiter

Durch die Weiterleitung von Sitzungsaktionen können Mitarbeiter sofort mit neuen Aufgaben beginnen und es gibt keine Wartezeiten zwischen den Anfragen des Terminplaners. Es sorgt auch für eine höhere Effizienz der Mitarbeiter und eine bessere Aufgabenverteilung bei lang andauernden Prozessen.

Wenn ein neuer Job mit höherer Priorität verfügbar ist, beendet der Mitarbeiter außerdem die gesamte ihm zuvor zugewiesene Arbeit, bevor die aktuelle Sitzung endet und eine neue Sitzung aus einem Job mit höherer Priorität zugewiesen wird.

Abhängigkeiten der Schritte

Deadline Cloud unterstützt die Definition von Abhängigkeiten zwischen Schritten, sodass ein Schritt wartet, bis ein anderer Schritt abgeschlossen ist, bevor er gestartet wird. Sie können mehr als eine Abhängigkeit für einen Schritt definieren. Ein Schritt mit einer Abhängigkeit wird erst geplant, wenn alle Abhängigkeiten abgeschlossen sind.

Wenn die Jobvorlage eine zirkuläre Abhängigkeit definiert, wird der Job abgelehnt und der Jobstatus wird auf gesetzt`CREATE_FAILED`.

Mit der folgenden Jobvorlage wird ein Job in zwei Schritten erstellt. `StepB`hängt davon ab`StepA`. `StepB`wird erst ausgeführt, nachdem der `StepA` Vorgang erfolgreich abgeschlossen wurde.

Nachdem der Job erstellt wurde, `StepA` befindet er sich im `READY` Status und `StepB` befindet sich im `PENDING` Status. Wenn der `StepA` Vorgang abgeschlossen ist, `StepB` wechselt er in den `READY` Status. `StepA`Schlägt fehl oder wurde der `StepA` Vorgang abgebrochen, `StepB` wechselt er in den `CANCELED` Status.

Sie können eine Abhängigkeit von mehreren Schritten festlegen. Wenn beispielsweise von beiden `StepA` und `StepC` abhängt`StepB`, `StepC` wird erst gestartet, wenn die anderen beiden Schritte abgeschlossen sind.

Für Schrittabhängigkeiten gelten die folgenden Einschränkungen:

- Abhängigkeiten pro Schritt — Ein Schritt kann von maximal 128 anderen Schritten abhängen.
- Verbraucher pro Schritt — Maximal 32 weitere Schritte können von einem einzigen Schritt abhängen.

```
name: Step-Step Dependency Test
specificationVersion: 'jobtemplate-2023-09'
steps:
- name: A
  script:
    actions:
      onRun:
        command: bash
        args: ['{{ Task.File.run }}']
    embeddedFiles:
      - name: run
        type: TEXT
        data: |
          #!/bin/env bash

          set -euo pipefail

          sleep 1
          echo Task A Done!
- name: B
  dependencies:
  - dependsOn: A # This means Step B depends on Step A
  script:
    actions:
      onRun:
        command: bash
        args: ['{{ Task.File.run }}']
    embeddedFiles:
      - name: run
        type: TEXT
        data: |
          #!/bin/env bash

          set -euo pipefail

          sleep 1
          echo Task B Done!
```

Einen Job in Deadline Cloud ändern

Sie können die folgenden `update` Befehle AWS Command Line Interface (AWS CLI) verwenden, um die Konfiguration eines Jobs zu ändern oder den Zielstatus eines Jobs, Schritts oder einer Aufgabe festzulegen:

- `aws deadline update-job`
- `aws deadline update-step`
- `aws deadline update-task`

Ersetzen Sie in den folgenden `update` Befehlsbeispielen jeden Befehl *`user input placeholder`* durch Ihre eigenen Informationen.

Example— Einen Job erneut in die Warteschlange stellen

Alle Aufgaben im Job wechseln in den `READY` Status, sofern es keine Schrittabhängigkeiten gibt. Schritte mit Abhängigkeiten wechseln zu entweder `READY` oder`PENDING`, wenn sie wiederhergestellt werden.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--target-task-run-status PENDING
```

Example— Stornieren Sie einen Job

Alle Aufgaben im Job, die nicht den Status haben `SUCCEEDED` oder markiert `FAILED` sind`CANCELED`.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--target-task-run-status CANCELED
```

Example— Einen Job als fehlgeschlagen markieren

Alle Aufgaben im Job, die den Status haben, `SUCCEEDED` bleiben unverändert. Alle anderen Aufgaben sind markiert`FAILED`.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--target-task-run-status FAILED
```

Example— Markiere einen Job als erfolgreich

Alle Aufgaben im Job werden in den SUCCEEDED Bundesstaat übertragen.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--target-task-run-status SUCCEEDED
```

Example— Einen Job aussetzen

Die Aufgaben des Jobs im FAILED Status SUCCEEDEDCANCELED, oder ändern sich nicht. Alle anderen Aufgaben sind markiertSUSPENDED.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--target-task-run-status SUSPENDED
```

Example— Ändern Sie die Priorität eines Jobs

Aktualisiert die Priorität eines Jobs in einer Warteschlange, um die Reihenfolge zu ändern, in der er geplant wird. Jobs mit höherer Priorität werden in der Regel zuerst geplant.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--priority 100
```

Example— Ändert die Anzahl der zulässigen fehlgeschlagenen Aufgaben

Aktualisiert die maximale Anzahl fehlgeschlagener Aufgaben, die der Job haben kann, bevor die verbleibenden Aufgaben abgebrochen werden.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--max-failed-tasks-count 200
```

Example— Ändert die Anzahl der zulässigen Aufgabenwiederholungen

Aktualisiert die maximale Anzahl von Wiederholungen für eine Aufgabe, bevor die Aufgabe fehlschlägt. Eine Aufgabe, die die maximale Anzahl von Wiederholungen erreicht hat, kann erst in die Warteschlange gestellt werden, wenn dieser Wert erhöht wird.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--max-retries-per-task 10
```

Example— Archivieren Sie einen Job

Aktualisiert den Lebenszyklusstatus des Jobs auf`ARCHIVED`. Archivierte Jobs können nicht geplant oder geändert werden. Sie können nur einen Job archivieren, der sich im `SUSPENDED` Status `FAILEDCANCELED`,`SUCCEEDED`, oder befindet.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--lifecycle-status ARCHIVED
```

Example— Ändern Sie den Namen eines Jobs

Aktualisiert den Anzeigenamen eines Jobs. Der Jobname kann bis zu 128 Zeichen lang sein.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--name "New Job Name"
```

Example— Ändern Sie die Beschreibung eines Jobs

Aktualisiert die Beschreibung eines Jobs. Die Beschreibung kann bis zu 2048 Zeichen lang sein. Um die bestehende Beschreibung zu entfernen, übergeben Sie eine leere Zeichenfolge.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--description "New Job Description"
```

Example— Einen Schritt erneut in die Warteschlange stellen

Alle Aufgaben im Schritt wechseln in den `READY` Status, sofern keine Schrittabhängigkeiten bestehen. Aufgaben in Schritten mit Abhängigkeiten wechseln entweder zu `READY` oder`PENDING`, und die Aufgabe wird wiederhergestellt.

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status PENDING
```

Example— Brechen Sie einen Schritt ab

Alle Aufgaben in dem Schritt, die nicht den Status haben `SUCCEEDED` oder markiert `FAILED` sind`CANCELED`.

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status CANCELED
```

Example— Einen Schritt als fehlgeschlagen markieren

Alle Aufgaben in dem Schritt, die den Status haben, `SUCCEEDED` bleiben unverändert. Alle anderen Aufgaben sind markiert`FAILED`.

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status FAILED
```

Example— Markiere einen Schritt als erfolgreich

Alle Aufgaben in dem Schritt sind markiertSUCCEEDED.

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status SUCCEEDED
```

Example— Einen Schritt aussetzen

Aufgaben im Schritt im FAILED Status SUCCEEDEDCANCELED, oder ändern sich nicht. Alle anderen Aufgaben sind markiertSUSPENDED.

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status SUSPENDED
```

Example— Den Status einer Aufgabe ändern

Wenn Sie den Befehl update-task Deadline Cloud CLI verwenden, wechselt die Aufgabe in den angegebenen Status.

```
aws deadline update-task \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--task-id taskID \
```

```
--target-task-run-status SUCCEEDED | SUSPENDED | CANCELED | FAILED | PENDING
```

Kundenverwaltete Flotten von Deadline Cloud erstellen und verwenden

Wenn Sie eine kundenverwaltete Flotte (CMF) einrichten, haben Sie die volle Kontrolle über Ihre Verarbeitungspipeline. Sie definieren die Netzwerk- und Softwareumgebung für jeden Mitarbeiter. Deadline Cloud fungiert als Repository und Scheduler für Ihre Jobs.

Ein Mitarbeiter kann eine Amazon Elastic Compute Cloud (Amazon EC2) -Instance, ein Mitarbeiter in einer Colocation-Einrichtung oder ein Mitarbeiter vor Ort sein. Jeder Mitarbeiter muss den Deadline Cloud-Worker-Agent ausführen. Alle Mitarbeiter müssen Zugriff auf den Deadline Cloud-Serviceendpunkt haben.

In den folgenden Themen erfahren Sie, wie Sie mithilfe von Amazon EC2 EC2-Instances ein einfaches CMF erstellen.

Topics

Erstellen Sie eine vom Kunden verwaltete Flotte

Gehen Sie wie folgt vor, um eine kundenverwaltete Flotte (CMF) zu erstellen.

Deadline Cloud console

Um mit der Deadline Cloud-Konsole eine vom Kunden verwaltete Flotte zu erstellen

1. Öffnen Sie die Deadline Cloud-Konsole.

2. Wählen Sie Farmen aus. Eine Liste der verfügbaren Farmen wird angezeigt.
3. Wählen Sie den Namen der Farm aus, in der Sie arbeiten möchten.
4. Wählen Sie die Registerkarte Flotten und dann Flotte erstellen aus.
5. Geben Sie einen Namen für Ihre Flotte ein.
6. (Optional) Geben Sie eine Beschreibung für Ihre Flotte ein.
7. Wählen Sie als Flottenart die Option Vom Kunden verwaltet aus.
8. Wählen Sie den Servicezugang Ihrer Flotte aus.

    a. Wir empfehlen, für jede Flotte die Option Neue Servicerolle erstellen und verwenden zu verwenden, um die Berechtigungen detaillierter steuern zu können. Diese Option ist standardmäßig ausgewählt.

    b. Sie können auch eine bestehende Servicerolle verwenden, indem Sie eine Servicerolle auswählen auswählen.
9. Überprüfen Sie Ihre Auswahl und wählen Sie dann Weiter.
10. Wählen Sie ein Betriebssystem für Ihre Flotte aus. Alle Mitarbeiter einer Flotte müssen über ein gemeinsames Betriebssystem verfügen.
11. Wählen Sie die Host-CPU-Architektur aus.
12. Wählen Sie die minimalen und maximalen vCPU- und Speicher-Hardwarekapazitäten aus, um die Workload-Anforderungen Ihrer Flotten zu erfüllen.
13. Wählen Sie einen Auto Scaling-Typ aus. Weitere Informationen finden Sie unter Verwendung EventBridge zur Behandlung von Auto Scaling Scaling-Ereignissen.

    - Keine Skalierung: Sie erstellen eine lokale Flotte und möchten Deadline Cloud Auto Scaling deaktivieren.
    - Empfehlungen zur Skalierung: Sie erstellen eine Amazon Elastic Compute Cloud (Amazon EC2) -Flotte.
14. (Optional) Wählen Sie den Pfeil aus, um den Abschnitt Funktionen hinzufügen zu erweitern.
15. (Optional) Aktivieren Sie das Kontrollkästchen GPU-Fähigkeit hinzufügen — Optional und geben Sie dann den Mindest- GPUs und Höchstwert sowie den Arbeitsspeicher ein.
16. Überprüfen Sie Ihre Auswahl und wählen Sie dann Weiter.
17. (Optional) Definieren Sie benutzerdefinierte Worker-Funktionen und wählen Sie dann Weiter.
18. Wählen Sie in der Dropdownliste eine oder mehrere Warteschlangen aus, die Sie der Flotte zuordnen möchten.

> **Note**
>
> Wir empfehlen, eine Flotte nur Warteschlangen zuzuordnen, die sich alle innerhalb derselben Vertrauensgrenze befinden. Diese Empfehlung gewährleistet eine starke Sicherheitsgrenze zwischen der Ausführung von Aufträgen auf demselben Worker.

19. Überprüfen Sie die Warteschlangenzuordnungen und wählen Sie dann Weiter.
20. (Optional) Für die Standard-Conda-Warteschlangenumgebung erstellen wir eine Umgebung für Ihre Warteschlange, in der die von Jobs angeforderten Conda-Pakete installiert werden.

> **Note**
>
> Die Conda-Warteschlangenumgebung wird verwendet, um Conda-Pakete zu installieren, die von Jobs angefordert werden. Normalerweise sollten Sie die Conda-Warteschlangenumgebung für die zugehörigen Warteschlangen deaktivieren, CMFs da dort die erforderlichen Conda-Befehle standardmäßig CMFs nicht installiert sind.

21. (Optional) Fügen Sie Ihrem CMF Tags hinzu. Weitere Informationen finden Sie unter Taggen Ihrer AWS Ressourcen.
22. Überprüfen Sie Ihre Flottenkonfiguration, nehmen Sie alle Änderungen vor und wählen Sie dann Flotte erstellen.
23. Wählen Sie die Registerkarte Flotten aus und notieren Sie sich die Flotten-ID.

AWS CLI

Um den zu verwenden AWS CLI , um eine vom Kunden verwaltete Flotte zu erstellen

1. Öffnen Sie ein -Terminalfenster.
2. `fleet-trust-policy.json`In einem neuen Editor erstellen.

   a. Fügen Sie die folgende IAM-Richtlinie hinzu und ersetzen Sie den *`ITALICIZED`* Text durch Ihre AWS Konto-ID und Deadline Cloud-Farm-ID.

   JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "credentials.deadline.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnEquals": {
                    "aws:SourceArn":
 "arn:aws:deadline:*:111122223333:farm/FARM_ID"
                }
            }
        }
    ]
}
```

b. Speichern Sie Ihre Änderungen.

3. Geben Sie einen Namen für den Benutzer ein und klicken Sie dann auf `fleet-policy.json`.

   a. Fügen Sie die folgende IAM-Richtlinie hinzu.

      JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "deadline:AssumeFleetRoleForWorker",
                "deadline:UpdateWorker",
                "deadline:DeleteWorker",
                "deadline:UpdateWorkerSchedule",
                "deadline:BatchGetJobEntity",
                "deadline:AssumeQueueRoleForWorker"
            ],
```

```
            "Resource": "arn:aws:deadline:*:111122223333:*",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalAccount": "${aws:ResourceAccount}"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream"
            ],
            "Resource": "arn:aws:logs:*:*:*://deadline/*",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalAccount": "${aws:ResourceAccount}"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents",
                "logs:GetLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:*:/aws/deadline/*",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalAccount": "${aws:ResourceAccount}"
                }
            }
        }
    ]
}
```

b. Speichern Sie Ihre Änderungen.

4. Fügen Sie eine IAM-Rolle hinzu, die die Mitarbeiter in Ihrer Flotte verwenden können.

```
aws iam create-role --role-name FleetWorkerRoleName --assume-role-policy-
document file://fleet-trust-policy.json
aws iam put-role-policy --role-name FleetWorkerRoleName --policy-name
 FleetWorkerPolicy --policy-document file://fleet-policy.json
```

5. Geben Sie einen Namen für den Benutzer ein und klicken Sie dann auf `create-fleet-request.json`.

   a. Fügen Sie die folgende IAM-Richtlinie hinzu und ersetzen Sie den KURSIV gedruckten Text durch die Werte Ihres CMF.

   > **Note**
   >
   > Sie finden die in der. *ROLE_ARN* `create-cmf-fleet.json`
   > Für *OS_FAMILY* die müssen Sie eines von`linux`, `macos` oder wählen`windows`.

```
{
    "farmId": "FARM_ID",
    "displayName": "FLEET_NAME",
    "description": "FLEET_DESCRIPTION",
    "roleArn": "ROLE_ARN",
    "minWorkerCount": 0,
    "maxWorkerCount": 10,
    "configuration": {
        "customerManaged": {
            "mode": "NO_SCALING",
            "workerCapabilities": {
                "vCpuCount": {
                    "min": 1,
                    "max": 4
                },
                "memoryMiB": {
                    "min": 1024,
                    "max": 4096
                },
                "osFamily": "OS_FAMILY",
                "cpuArchitectureType": "x86_64",
            },
        },
    }
}
```

   b. Speichern Sie Ihre Änderungen.

6. Stellen Sie Ihre Flotte zusammen.

```
aws deadline create-fleet --cli-input-json file://create-fleet-request.json
```

Einrichtung und Konfiguration des Worker-Hosts

Ein Worker-Host bezieht sich auf einen Host-Computer, auf dem ein Deadline Cloud-Worker ausgeführt wird. In diesem Abschnitt wird erklärt, wie Sie den Worker-Host einrichten und für Ihre spezifischen Bedürfnisse konfigurieren. Jeder Worker-Host führt ein Programm aus, das als Worker-Agent bezeichnet wird. Der Worker-Agent ist verantwortlich für:

- Verwaltung des Lebenszyklus des Arbeitnehmers.
- Synchronisieren der zugewiesenen Arbeit, ihres Fortschritts und ihrer Ergebnisse.
- Überwachung laufender Arbeiten.
- Weiterleiten von Protokollen an konfigurierte Ziele.

Wir empfehlen Ihnen, den mitgelieferten Deadline Cloud-Worker-Agent zu verwenden. Der Worker-Agent ist Open Source und wir freuen uns über Funktionsanfragen, aber Sie können ihn auch entwickeln und an Ihre Bedürfnisse anpassen.

Um die Aufgaben in den folgenden Abschnitten ausführen zu können, benötigen Sie Folgendes:

Linux

- Eine Linux basierte Amazon Elastic Compute Cloud (Amazon EC2) -Instance. Wir empfehlen Amazon Linux 2023.
- `sudo`Privilegien
- Python 3.9 oder höher

Windows

- Eine Windows basierte Amazon Elastic Compute Cloud (Amazon EC2) -Instance. Wir empfehlenWindows Server 2022.

- Administratorzugriff auf den Worker-Host

- Python 3.9 oder höher für alle Benutzer installiert

Erstellen und konfigurieren Sie eine virtuelle Python-Umgebung

Sie können eine virtuelle Python-Umgebung erstellen, Linux wenn Sie Python 3.9 oder höher installiert und in Ihrem platziert habenPATH.

> **Note**
>
> Bei Aktivierung Windows müssen die Agentdateien im globalen Site-Packages-Verzeichnis von Python installiert werden. Virtuelle Python-Umgebungen werden derzeit nicht unterstützt.

Um eine virtuelle Python-Umgebung zu erstellen und zu aktivieren

1. Öffnen Sie ein Terminal als `root` Benutzer (oder verwenden Sie`sudo/su`).
2. Erstellen und aktivieren Sie eine virtuelle Python-Umgebung.

```
python3 -m venv /opt/deadline/worker
source /opt/deadline/worker/bin/activate
pip install --upgrade pip
```

Installieren Sie den Deadline Cloud Worker Agent

Nachdem Sie Ihr Python eingerichtet und eine virtuelle Umgebung erstellt habenLinux, installieren Sie die Python-Pakete für den Deadline Cloud Worker Agent.

Um die Python-Pakete für den Worker Agent zu installieren

Linux

1. Öffnen Sie ein Terminal als `root` Benutzer (oder verwenden Sie`sudo/su`).
2. Laden Sie die Deadline Cloud Worker Agent-Pakete von PyPI herunter und installieren Sie sie:

```
/opt/deadline/worker/bin/python -m pip install deadline-cloud-worker-agent
```

Windows

1. Öffnen Sie eine Administrator-Befehlszeile oder PowerShell ein Terminal.
2. Laden Sie die Deadline Cloud Worker Agent-Pakete von PyPI herunter und installieren Sie sie:

```
python -m pip install deadline-cloud-worker-agent
```

Wenn Ihr Windows Worker-Host lange Pfadnamen (mehr als 250 Zeichen) benötigt, müssen Sie lange Pfadnamen wie folgt aktivieren:

Um lange Pfade für Windows Worker-Hosts zu aktivieren

1. Stellen Sie sicher, dass der Registrierungsschlüssel für lange Pfade aktiviert ist. Weitere Informationen finden Sie unter Registrierungseinstellung zur Aktivierung von Protokollpfaden auf der Microsoft-Website.
2. Installieren Sie das Windows SDK für C++ x86-Desktop-Apps. Weitere Informationen finden Sie unter WindowsSDK im Windows Dev Center.
3. Öffnen Sie den Python-Installationsort in Ihrer Umgebung, in dem der Worker-Agent installiert ist. Der Standardwert ist `C:\Program Files\Python311`. Es gibt eine ausführbare Datei mit dem Namen`pythonservice.exe`.
4. Erstellen Sie eine neue Datei mit `pythonservice.exe.manifest` dem Namen am selben Ort. Fügen Sie Folgendes hinzu:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
  <assemblyIdentity type="win32" name="pythonservice" processorArchitecture="x86"
 version="1.0.0.0"/>
  <application xmlns="urn:schemas-microsoft-com:asm.v3">
    <windowsSettings>
      <longPathAware xmlns="http://schemas.microsoft.com/SMI/2016/
WindowsSettings">true</longPathAware>
    </windowsSettings>
  </application>
</assembly>
```

5. Öffnen Sie eine Eingabeaufforderung und führen Sie den folgenden Befehl am Speicherort der von Ihnen erstellten Manifestdatei aus:

```
"C:\Program Files (x86)\Windows Kits\10\bin\10.0.26100.0\x86\mt.exe" -manifest
 pythonservice.exe.manifest -outputresource:pythonservice.exe;#1
```

Die Ausgabe sollte folgendermaßen oder ähnlich aussehen:

```
Microsoft (R) Manifest Tool
Copyright (c) Microsoft Corporation.
All rights reserved.
```

Der Worker kann jetzt auf lange Pfade zugreifen. Um zu bereinigen, entfernen Sie die `pythonservice.exe.manifest` Datei und deinstallieren Sie das SDK.

Konfigurieren Sie den Deadline Cloud Worker Agent

Sie können die Deadline Cloud-Worker-Agent-Einstellungen auf drei Arten konfigurieren. Wir empfehlen Ihnen, das Betriebssystem-Setup zu verwenden, indem Sie das `install-deadline-worker` Tool ausführen.

Der Worker-Agent unterstützt die Ausführung als Domänenbenutzer unter Windows nicht. Um einen Job als Domänenbenutzer auszuführen, können Sie ein Domänenbenutzerkonto angeben, wenn Sie einen Warteschlangenbenutzer für die Ausführung von Jobs konfigurieren. Weitere Informationen finden Sie unter Schritt 7 unter Deadline Cloud-Warteschlangen im AWS Deadline Cloud-Benutzerhandbuch.

Befehlszeilenargumente — Sie können Argumente angeben, wenn Sie den Deadline Cloud-Worker-Agent von der Befehlszeile aus ausführen. Einige Konfigurationseinstellungen sind nicht über Befehlszeilenargumente verfügbar. Geben Sie ein, um alle verfügbaren Befehlszeilenargumente anzuzeigen`deadline-worker-agent --help`.

Umgebungsvariablen — Sie können den Deadline Cloud-Worker-Agent konfigurieren, indem Sie die Umgebungsvariable festlegen, die mit beginnt`DEADLINE_WORKER_`. Um beispielsweise alle verfügbaren Befehlszeilenargumente zu sehen, können `export DEADLINE_WORKER_VERBOSE=true` Sie die Ausgabe des Worker-Agents auf ausführlich setzen. Weitere Beispiele und Informationen finden Sie unter `/etc/amazon/deadline/worker.toml.example` on Linux oder `C:\ProgramData\Amazon\Deadline\Config\worker.toml.example` onWindows.

Konfigurationsdatei — Wenn Sie den Worker-Agent installieren, erstellt er eine Konfigurationsdatei, die sich unter `/etc/amazon/deadline/worker.toml` on Linux oder `C:\ProgramData\Amazon\Deadline\Config\worker.toml` on befindetWindows. Der Worker-Agent lädt diese Konfigurationsdatei, wenn er gestartet wird. Sie können die Beispielkonfigurationsdatei (aktiviert Linux oder `/etc/amazon/deadline/worker.toml.example` `C:\ProgramData\Amazon\Deadline\Config\worker.toml.example` aktiviertWindows) verwenden, um die Standard-Worker-Agent-Konfigurationsdatei an Ihre spezifischen Bedürfnisse anzupassen.

Schließlich empfehlen wir Ihnen, das auto Herunterfahren für den Worker Agent zu aktivieren, nachdem Ihre Software bereitgestellt wurde und wie erwartet funktioniert. Auf diese Weise kann die Mitarbeiterflotte bei Bedarf erweitert und nach Abschluss eines Auftrags heruntergefahren werden. Durch die automatische Skalierung wird sichergestellt, dass Sie nur die benötigten Ressourcen nutzen. Damit eine von der Auto Scaling-Gruppe gestartete Instanz heruntergefahren werden kann, müssen Sie der `worker.toml` Konfigurationsdatei etwas hinzufügen`shutdown_on_stop=true`.

Um die auto Abschaltung zu aktivieren

Als **`root`** Benutzer:

- Installieren Sie den Worker Agent mit Parametern**`--allow-shutdown`**.

  Linux

  Geben Sie ein:

  ```
  /opt/deadline/worker/bin/install-deadline-worker \
    --farm-id FARM_ID \
    --fleet-id FLEET_ID \
    --region REGION \
    --allow-shutdown
  ```

  Windows

  Geben Sie ein:

  ```
  install-deadline-worker ^
    --farm-id FARM_ID ^
    --fleet-id FLEET_ID ^
    --region REGION ^
    --allow-shutdown
  ```

Erstellen Sie Job-Benutzer und -Gruppen

In diesem Abschnitt wird die erforderliche Benutzer- und Gruppenbeziehung zwischen dem Agent-Benutzer und den in Ihren Warteschlangen `jobRunAsUser` definierten Benutzern beschrieben.

Der Deadline Cloud-Worker-Agent sollte als dedizierter agentenspezifischer Benutzer auf dem Host ausgeführt werden. Sie sollten die `jobRunAsUser` Eigenschaft der Deadline Cloud-Warteschlangen so konfigurieren, dass Mitarbeiter die Warteschlangenjobs als ein bestimmter Betriebssystembenutzer und eine bestimmte Gruppe ausführen. Diese Konfiguration bedeutet, dass Sie die gemeinsamen Dateisystemberechtigungen für Ihre Jobs kontrollieren können. Sie stellt auch eine wichtige Sicherheitsgrenze zwischen Ihren Jobs und dem Worker-Agent-Benutzer dar.

LinuxJobbenutzer und Gruppen

Um einen lokalen Worker Agent-Benutzer einzurichten`jobRunAsUser`, stellen Sie sicher, dass Sie die folgenden Anforderungen erfüllen. Wenn Sie ein Linux Pluggable Authentication Module (PAM) wie Active Directory oder LDAP verwenden, ist Ihr Verfahren möglicherweise anders.

Der Worker-Agent-Benutzer und die gemeinsam genutzte `jobRunAsUser` Gruppe werden bei der Installation des Worker-Agents festgelegt. Die Standardwerte sind `deadline-worker-agent` und`deadline-job-users`, aber Sie können diese ändern, wenn Sie den Worker Agent installieren.

```
install-deadline-worker \
    --user AGENT_USER_NAME \
    --group JOB_USERS_GROUP
```

Befehle sollten als Root-Benutzer ausgeführt werden.

- Jeder `jobRunAsUser` sollte eine passende Primärgruppe haben. Wenn Sie einen Benutzer mit dem `adduser` Befehl erstellen, wird normalerweise eine passende Primärgruppe erstellt.

  ```
  adduser -r -m jobRunAsUser
  ```

- Die primäre Gruppe von `jobRunAsUser` ist eine sekundäre Gruppe für den Worker Agent-Benutzer. Die gemeinsam genutzte Gruppe ermöglicht es dem Worker-Agent, Dateien für den Job verfügbar zu machen, während dieser ausgeführt wird.

  ```
  usermod -a -G jobRunAsUser deadline-worker-agent
  ```

- Der `jobRunAsUser` muss Mitglied der gemeinsam genutzten Auftragsgruppe sein.

```
usermod -a -G deadline-job-users jobRunAsUser
```

- Der `jobRunAsUser` darf nicht zur primären Gruppe des Worker Agent-Benutzers gehören. Vertrauliche Dateien, die vom Worker-Agenten geschrieben wurden, gehören der primären Gruppe des Agenten. Wenn a Teil dieser Gruppe `jobRunAsUser` ist, können Jobs, die auf dem Worker ausgeführt werden, auf Dateien des Worker-Agents zugreifen.
- Die Standardeinstellung AWS-Region muss der Region der Farm entsprechen, zu der der Worker gehört. Dies sollte auf alle `jobRunAsUser` Konten des Arbeiters angewendet werden.

```
sudo -u jobRunAsUser aws configure set default.region aws-region
```

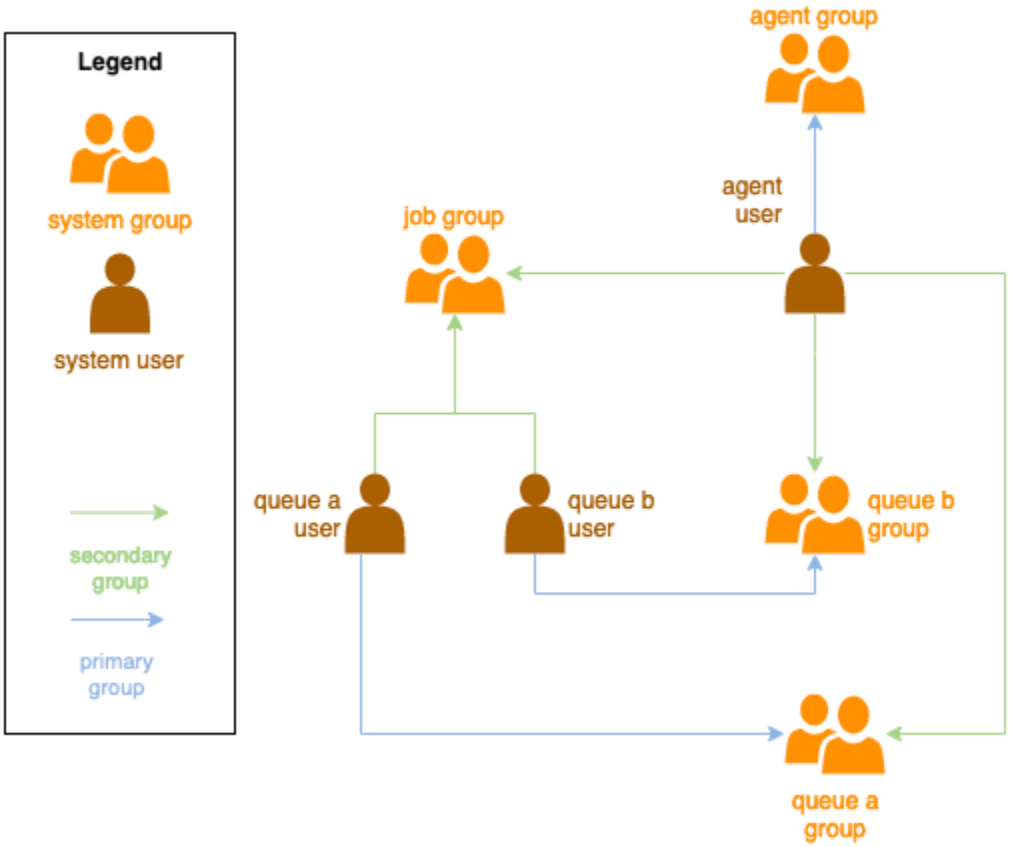
- Der Worker-Agent-Benutzer muss in der Lage sein, `sudo` Befehle als auszuführen`jobRunAsUser`. Führen Sie den folgenden Befehl aus, um einen Editor zu öffnen und eine neue Sudoers-Regel zu erstellen:

  ```
  visudo -f /etc/sudoers.d/deadline-worker-job-user
  ```

  Fügen Sie der Datei Folgendes hinzu:

  ```
  # Allows the Deadline Cloud worker agent OS user to run commands
   # as the queue OS user without requiring a password.
  deadline-worker-agent ALL=(jobRunAsUser) NOPASSWD:ALL
  ```

Das folgende Diagramm veranschaulicht die Beziehung zwischen dem Agent-Benutzer und den `jobRunAsUser` Benutzern und Gruppen für Warteschlangen, die der Flotte zugeordnet sind.



Windows-Benutzer

Um einen Windows Benutzer als Benutzer verwenden zu können`jobRunAsUser`, muss er die folgenden Anforderungen erfüllen:

- Alle `jobRunAsUser` Warteschlangenbenutzer müssen vorhanden sein.
- Ihre Passwörter müssen mit dem Wert des Geheimnisses übereinstimmen, das im `JobRunAsUser` Feld ihrer Warteschlange angegeben ist. Eine Anleitung finden Sie in Schritt 7 unter Deadline Cloud-Warteschlangen im AWS Deadline Cloud-Benutzerhandbuch.
- Der Agent-Benutzer muss in der Lage sein, sich als diese Benutzer anzumelden.

Sicherung Ihres Worker-Hosts

Beachten Sie bei der Einrichtung Ihres Worker-Hosts die bewährten Sicherheitsmethoden, um vertrauliche Informationen zu schützen und angemessene Zugriffskontrollen aufrechtzuerhalten.

Konfiguration der Berechtigungen für Protokollordner

Der Worker-Agent schreibt Protokolldateien, die vertrauliche Informationen aus Host-Konfigurationsskripten und der Jobausführung enthalten können. Der `install-deadline-worker` Befehl erstellt das Protokollverzeichnis mit sicheren Berechtigungen. Wenn Sie das Verzeichnis vor der Installation manuell erstellen müssen, gehen Sie wie folgt vor, um die Berechtigungen der vom Service verwalteten Flotten abzugleichen:

Linux

So konfigurieren Sie Protokollverzeichnisberechtigungen für Linux

1. Erstellen Sie das Protokollverzeichnis:

```
sudo mkdir -p /var/log/amazon/deadline
```

2. Stellen Sie den Besitzer und die Gruppe auf den Worker Agent-Benutzer ein:

```
sudo chown -R deadline-worker:deadline-worker /var/log/amazon/deadline
```

3. Legen Sie die Berechtigungen auf 750 fest:

```
sudo chmod -R 750 /var/log/amazon/deadline
```

   Diese Berechtigungen stellen sicher, dass nur der Benutzer und die Gruppe des Worker Agents auf die Protokolldateien zugreifen können. Dadurch wird verhindert, dass Jobbenutzer und andere nicht autorisierte Benutzer potenziell vertrauliche Informationen lesen.

Windows

So konfigurieren Sie Protokollverzeichnisberechtigungen für Windows

1. Öffnen Sie ein PowerShell Administratorterminal.
2. Erstellen Sie das Protokollverzeichnis:

```
New-Item -ItemType Directory -Force -Path "$env:PROGRAMDATA\Amazon\Deadline
\Logs"
```

3. Konfigurieren Sie ACLs die Konfiguration so, dass nur der Worker-Agent-Benutzer und Administratoren zugelassen sind:

```
$acl = Get-Acl "$env:PROGRAMDATA\Amazon\Deadline\Logs"
$acl.SetAccessRuleProtection($true, $false)
$acl.Access | ForEach-Object { $acl.RemoveAccessRule($_) }
$agentRule = New-Object
 System.Security.AccessControl.FileSystemAccessRule("deadline-worker",
 "FullControl", "ContainerInherit,ObjectInherit", "None", "Allow")
$adminRule = New-Object
 System.Security.AccessControl.FileSystemAccessRule("Administrators",
 "FullControl", "ContainerInherit,ObjectInherit", "None", "Allow")
$acl.AddAccessRule($agentRule)
$acl.AddAccessRule($adminRule)
Set-Acl "$env:PROGRAMDATA\Amazon\Deadline\Logs" $acl
```

Diese Befehle beschränken den Zugriff auf das Protokollverzeichnis nur auf den Worker Agent-Benutzer und die Administratorgruppe und verhindern so, dass Job-Benutzer und andere nicht autorisierte Benutzer potenziell vertrauliche Informationen lesen können.

Zugriff auf Windows Job-Benutzergeheimnisse verwalten

Wenn Sie eine Warteschlange mit einem konfigurieren Windows`jobRunAsUser`, müssen Sie ein AWS Secrets Manager Manager-Geheimnis angeben. Es wird erwartet, dass der Wert dieses Geheimnisses ein JSON-kodiertes Objekt der folgenden Form ist:

```
{
    "password": "JOB_USER_PASSWORD"
}
```

Damit Worker Jobs so ausführen können, wie die Warteschlange konfiguriert ist`jobRunAsUser`, muss die IAM-Rolle der Flotte über die erforderlichen Berechtigungen verfügen, um den Wert des Geheimnisses abzurufen. Wenn das Geheimnis mit einem vom Kunden verwalteten KMS-Schlüssel verschlüsselt wird, muss die IAM-Rolle der Flotte auch über Berechtigungen zur Entschlüsselung mit dem KMS-Schlüssel verfügen.

Es wird dringend empfohlen, bei diesen Geheimnissen das Prinzip der geringsten Rechte einzuhalten. Das bedeutet, dass der Zugriff zum Abrufen des geheimen Werts von `jobRunAsUser` → → `windows` einer Warteschlange wie folgt sein sollte: `passwordArn`

- wird einer Flottenrolle zugewiesen, wenn zwischen der Flotte und der Warteschlange eine Verbindung zwischen Warteschlange und Flotte erstellt wird
- wird einer Flottenrolle entzogen, wenn zwischen der Flotte und der Warteschlange eine Flottenverbindung zwischen der Flotte und der Warteschlange gelöscht wird

Außerdem sollte das AWS Secrets Manager Manager-Geheimnis, das das `jobRunAsUser` Passwort enthält, gelöscht werden, wenn es nicht mehr verwendet wird.

Gewähren Sie Zugriff auf ein geheimes Passwort

Deadline Cloud-Flotten benötigen Zugriff auf das `jobRunAsUser` Passwort, das im Passwortgeheimnis der Warteschlange gespeichert ist, wenn die Warteschlange und die Flotte verknüpft werden. Wir empfehlen, die AWS Secrets Manager Manager-Ressourcenrichtlinie zu verwenden, um Zugriff auf die Flottenrollen zu gewähren. Wenn Sie sich strikt an diese Richtlinie halten, ist es einfacher festzustellen, welche Flottenrollen Zugriff auf den geheimen Schlüssel haben.

Um Zugriff auf das Geheimnis zu gewähren

1. Öffnen Sie die AWS Secret Manager-Konsole für das Geheimnis.
2. Fügen Sie im Abschnitt Ressourcenberechtigungen eine Richtlinienerklärung der folgenden Form hinzu:

   JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "arn:aws:iam::111122223333:role/FleetRole"
      },
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource" : "arn:aws:secretsmanager:us-
east-1:111122223333:secret:YourSecretName-ABC123"
    }
  ]
```

```
}
```

Widerrufen Sie den Zugriff auf ein geheimes Passwort

Wenn eine Flotte keinen Zugriff mehr auf eine Warteschlange benötigt, entfernen Sie den Zugriff auf das geheime Passwort für die Warteschlange`jobRunAsUser`. Wir empfehlen, die AWS Secrets Manager Manager-Ressourcenrichtlinie zu verwenden, um Zugriff auf die Flottenrollen zu gewähren. Wenn Sie sich strikt an diese Richtlinie halten, ist es einfacher festzustellen, welche Flottenrollen Zugriff auf den geheimen Schlüssel haben.

Um den Zugriff auf das Geheimnis zu entziehen

1. Öffnen Sie die AWS Secret Manager-Konsole für das Geheimnis.
2. Entfernen Sie im Abschnitt Ressourcenberechtigungen die Richtlinienerklärung in der folgenden Form:

   JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "arn:aws:iam::111122223333:role/FleetRole"
      },
      "Action" : [
        "secretsmanager:GetSecretValue"
      ],
      "Resource" : "arn:aws:secretsmanager:us-
east-1:111122223333:secret:YourSecretName-ABC123"
    }
  ]
}
```

Installation und Konfiguration der für Jobs erforderlichen Software

Nachdem Sie den Deadline Cloud-Worker-Agent eingerichtet haben, können Sie den Worker-Host mit jeder Software vorbereiten, die für die Ausführung von Jobs erforderlich ist.

Wenn Sie einen Job an eine Warteschlange senden, der ein zugeordneter Job zugeordnet ist`jobRunAsUser`, wird der Job als dieser Benutzer ausgeführt. Wenn ein Job mit Befehlen übermittelt wird, bei denen es sich nicht um einen absoluten Pfad handelt, muss sich dieser Befehl im `PATH` Verzeichnis dieses Benutzers befinden.

Unter Linux können Sie das `PATH` für einen Benutzer in einer der folgenden Optionen angeben:

- ihr `~/.bashrc` oder `~/.bash_profile`
- Systemkonfigurationsdateien wie `/etc/profile.d/*` und `/etc/profile`
- Shell-Startskripte:`/etc/bashrc`.

Unter Windows können Sie das `PATH` für einen Benutzer in einer der folgenden Optionen angeben:

- ihre benutzerspezifischen Umgebungsvariablen
- die systemweiten Umgebungsvariablen

Installieren Sie die Adapter für Tools zur Erstellung digitaler Inhalte

Deadline Cloud bietet OpenJobDescription Adapter für die Nutzung beliebter DCC-Anwendungen (Digital Content Creation). Um diese Adapter in einer vom Kunden verwalteten Flotte zu verwenden, müssen Sie die DCC-Software und die Anwendungsadapter installieren. Stellen Sie anschließend sicher, dass die ausführbaren Programme der Software im Systemsuchpfad verfügbar sind (z. B. in der `PATH` Umgebungsvariablen).

Um DCC-Adapter in einer vom Kunden verwalteten Flotte zu installieren

1. Öffnen Sie das A-Terminal.
    a. Öffnen Sie unter Linux ein Terminal als `root` Benutzer (oder verwenden Sie`sudo`/`su`)
    b. Öffnen Sie Windows unter eine Administrator-Befehlszeile oder ein PowerShell Terminal.
2. Installieren Sie die Deadline Cloud-Adapterpakete.

```
pip install deadline deadline-cloud-for-maya deadline-cloud-for-nuke deadline-
cloud-for-blender deadline-cloud-for-3ds-max
```

AWS Anmeldeinformationen konfigurieren

Die Anfangsphase des Lebenszyklus eines Mitarbeiters ist Bootstrapping. In dieser Phase erstellt die Worker-Agent-Software einen Worker in Ihrer Flotte und bezieht die AWS Anmeldeinformationen für den weiteren Betrieb von der Rolle Ihres Fuhrparks.

AWS credentials for Amazon EC2

So erstellen Sie eine IAM-Rolle für Amazon EC2 mit Deadline Cloud-Worker-Host-Berechtigungen

1. Öffnen Sie unter https://console.aws.amazon.com/iam/ die IAM-Konsole.
2. Wählen Sie im Navigationsbereich Rollen und anschließend Rolle erstellen aus.
3. Wählen Sie AWS Dienst aus.
4. Wählen Sie EC2 als Dienst oder Anwendungsfall aus und wählen Sie dann Weiter aus.
5. Um die erforderlichen Berechtigungen zu gewähren, fügen Sie die `AWSDeadlineCloud-WorkerHost` AWS verwaltete Richtlinie bei.

On-premises AWS credentials

Ihre Mitarbeiter vor Ort verwenden Anmeldeinformationen, um auf Deadline Cloud zuzugreifen. Für den sichersten Zugriff empfehlen wir die Verwendung von IAM Roles Anywhere zur Authentifizierung Ihrer Mitarbeiter. Weitere Informationen finden Sie unter IAM Roles Anywhere.

Zum Testen können Sie IAM-Benutzerzugriffsschlüssel für AWS Anmeldeinformationen verwenden. Wir empfehlen, dass Sie ein Ablaufdatum für den IAM-Benutzer festlegen, indem Sie eine restriktive Inline-Richtlinie einbeziehen.

> **Important**
>
> Beachten Sie die folgenden Warnungen:
>
> - Verwenden Sie NICHT die Root-Anmeldeinformationen Ihres Kontos, um auf AWS Ressourcen zuzugreifen. Diese Anmeldeinformationen bieten uneingeschränkten Zugriff auf Konten und können nur schwer widerrufen werden.

- Fügen Sie KEINE tatsächlichen Zugriffsschlüssel oder Anmeldeinformationen in Ihre Anwendungsdateien ein. Wenn Sie dies tun, riskieren Sie damit, dass Ihre Kontodaten versehentlich offengelegt werden, falls Sie z. B. das Projekt in ein öffentliches Repository hochladen.
- Fügen Sie KEINE Dateien in Ihrem Projektbereich hinzu, die Anmeldeinformationen enthalten.
- Sichern Sie Ihre Zugangsschlüssel. Geben Sie Ihre Zugangsschlüssel nicht an Unbefugte weiter, auch nicht, um Ihre Kontokennungen zu finden. Dadurch kann eine Person permanenten Zugriff auf Ihr Konto erlangen.
- Beachten Sie, dass alle in der Datei mit den gemeinsam genutzten AWS Anmeldeinformationen gespeicherten Anmeldeinformationen im Klartext gespeichert werden.

Weitere Informationen finden Sie unter Bewährte Methoden für die Verwaltung von AWS Zugriffsschlüsseln in der AWS Allgemeinen Referenz.

Erstellen eines IAM-Benutzers

1. Öffnen Sie unter https://console.aws.amazon.com/iam/ die IAM-Konsole.
2. Wählen Sie im Navigationsbereich Benutzer und anschließend Benutzer erstellen aus.
3. Geben Sie dem Benutzer einen Namen. Deaktivieren Sie das Kontrollkästchen für Benutzerzugriff auf gewähren AWS-Managementkonsole, und wählen Sie dann Weiter.
4. Wählen Sie Richtlinien direkt anhängen aus.
5. Wählen Sie aus der Liste der Berechtigungsrichtlinien die AWSDeadlineCloud-WorkerHostRichtlinie aus und klicken Sie dann auf Weiter.
6. Überprüfen Sie die Benutzerdetails und wählen Sie dann Benutzer erstellen aus.

Beschränken des Benutzerzugriffs auf ein begrenztes Zeitfenster

Alle von Ihnen erstellten Zugriffsschlüssel für IAM-Benutzer stellen langfristige Anmeldeinformationen dar. Um sicherzustellen, dass diese Anmeldeinformationen bei unsachgemäßer Verwendung ablaufen, können Sie die Anmeldeinformationen zeitlich begrenzen, indem Sie eine Inline-Richtlinie erstellen, die ein Datum festlegt, nach dem die Schlüssel nicht mehr gültig sind.

1. Öffnen Sie den IAM-Benutzer, den Sie gerade erstellt haben. Wählen Sie auf der Registerkarte „Berechtigungen“ die Option „Berechtigungen hinzufügen“ und dann „Inline-Richtlinie erstellen“ aus.
2. Geben Sie im JSON-Editor die folgenden Berechtigungen an. Um diese Richtlinie zu verwenden, ersetzen Sie den `aws:CurrentTime` Zeitstempelwert in der Beispielrichtlinie durch Ihre eigene Uhrzeit und Ihr eigenes Datum.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "DateGreaterThan": {
          "aws:CurrentTime": "2024-01-01T00:00:00Z"
        }
      }
    }
  ]
}
```

Erstellen eines Zugriffsschlüssels

1. Wählen Sie auf der Seite mit den Benutzerdetails die Registerkarte Sicherheitsanmeldeinformationen aus. Wählen Sie im Abschnitt Zugriffsschlüssel die Option Zugriffsschlüssel erstellen aus.
2. Geben Sie an, dass Sie den Schlüssel für Andere verwenden möchten, klicken Sie dann auf Weiter und anschließend auf Zugriffsschlüssel erstellen.
3. Wählen Sie auf der Seite Zugriffsschlüssel abrufen die Option Anzeigen aus, um den Wert des geheimen Zugriffsschlüssels Ihres Benutzers anzuzeigen. Sie können die Anmeldeinformationen kopieren oder eine CSV-Datei herunterladen.

Speichern Sie die Benutzerzugriffsschlüssel

- Speichern Sie die Benutzerzugriffsschlüssel in der Datei mit den AWS Anmeldeinformationen des Agent-Benutzers auf dem Worker-Host-System:
  - AnLinux, die Datei befindet sich unter `~/.aws/credentials`
  - AnWindows, die Datei befindet sich unter `%USERPROFILE\.aws\credentials`

  Ersetzen Sie die folgenden Schlüssel:

```
[default]
aws_access_key_id=ACCESS_KEY_ID
aws_secret_access_key=SECRET_ACCESS_KEY
```

> **Important**
>
> Wenn Sie diesen IAM-Benutzer nicht mehr benötigen, empfehlen wir, ihn zu entfernen, um den bewährten AWS Sicherheitsmethoden zu entsprechen. Wir empfehlen, dass Ihre menschlichen Benutzer AWS IAM Identity Centerbeim Zugriff AWS temporäre Anmeldeinformationen verwenden müssen.

Konfiguration des Netzwerks für AWS-Endpunktverbindungen

Deadline Cloud erfordert für einen ordnungsgemäßen Betrieb eine sichere Konnektivität zu verschiedenen AWS Service-Endpunkten. Um Deadline Cloud verwenden zu können, müssen Sie sicherstellen, dass Ihre Netzwerkumgebung es Ihren Deadline Cloud-Mitarbeitern ermöglicht, sich mit diesen Endpunkten zu verbinden.

Wenn Sie über eine Netzwerk-Firewall verfügen, die ausgehende Verbindungen blockiert, müssen Sie möglicherweise Firewall-Ausnahmen für bestimmte Endpunkte hinzufügen. Für Deadline Cloud müssen Sie Ausnahmen für die folgenden Dienste hinzufügen:

- Deadline Cloud-Endpunkte
- Amazon CloudWatch Logs-Endpunkte
- Amazon Simple Storage Service-Endpunkte

Wenn Ihre Jobs andere AWS Dienste nutzen, müssen Sie möglicherweise auch Ausnahmen für diese Dienste hinzufügen. Sie finden diese Endpunkte im Kapitel Service-Endpunkte und Kontingente des AWS General Reference Guide. Nachdem Sie die erforderlichen Endpunkte identifiziert haben, erstellen Sie Regeln für ausgehenden Datenverkehr in Ihrer Firewall, um den Datenverkehr zu diesen spezifischen Endpunkten zuzulassen.

Für einen ordnungsgemäßen Betrieb muss sichergestellt werden, dass auf diese Endpunkte zugegriffen werden kann. Erwägen Sie außerdem die Implementierung geeigneter Sicherheitsmaßnahmen, wie z. B. die Verwendung virtueller privater Clouds (VPCs), Sicherheitsgruppen und Netzwerkzugriffskontrolllisten (ACLs), um eine sichere Umgebung aufrechtzuerhalten und gleichzeitig den erforderlichen Deadline-Cloud-Verkehr zuzulassen.

Testen Sie die Konfiguration Ihres Worker-Hosts

Nachdem Sie den Worker Agent installiert, die für die Verarbeitung Ihrer Jobs erforderliche Software installiert und die AWS Anmeldeinformationen für den Worker Agent konfiguriert haben, sollten Sie testen, ob die Installation Ihre Jobs verarbeiten kann, bevor Sie einen erstellen AMI für Ihre Flotte. Sie sollten Folgendes testen:

- Der Deadline Cloud-Worker-Agent ist ordnungsgemäß für die Ausführung als Systemdienst konfiguriert.
- Dass der Worker die zugehörige Warteschlange nach Arbeit abfragt.
- Dass der Mitarbeiter erfolgreich Aufträge verarbeitet, die an die der Flotte zugeordnete Warteschlange gesendet wurden.

Nachdem Sie die Konfiguration getestet haben und repräsentative Jobs erfolgreich verarbeiten können, können Sie mit dem konfigurierten Worker eine erstellen AMI für EC2 Amazon-Mitarbeiter oder als Modell für Ihre Mitarbeiter vor Ort.

> **Note**
>
> Wenn Sie die Worker-Host-Konfiguration einer Auto Scaling-Flotte testen, können Sie in den folgenden Situationen Schwierigkeiten haben, Ihren Worker zu testen:
>
> - Wenn sich keine Arbeit in der Warteschlange befindet, stoppt Deadline Cloud den Worker-Agent kurz nach dem Start des Workers.

- Wenn der Worker-Agent so konfiguriert ist, dass er den Host beim Stoppen herunterfährt, fährt der Agent die Maschine herunter, wenn sich keine Arbeit in der Warteschlange befindet.

Um diese Probleme zu vermeiden, sollten Sie eine Staging-Flotte verwenden, die nicht auto skaliert, um Ihre Mitarbeiter zu konfigurieren und zu testen. Stellen Sie nach dem Testen des Worker-Hosts sicher, dass Sie die richtige Flotten-ID angeben, bevor Sie ein AMI.

Um Ihre Worker-Host-Konfiguration zu testen

1. Führen Sie den Worker-Agent aus, indem Sie den Betriebssystemdienst starten.

   Linux

   Führen Sie in einer Root-Shell den folgenden Befehl aus:

   ```
   systemctl start deadline-worker
   ```

   Windows

   Von einer Administrator-Befehlszeile aus oder PowerShell Terminal, geben Sie den folgenden Befehl ein:

   ```
   sc.exe start DeadlineWorker
   ```

2. Überwachen Sie den Mitarbeiter, um sicherzustellen, dass er startet und nach Arbeit fragt.

   Linux

   Führen Sie in einer Root-Shell den folgenden Befehl aus:

   ```
   systemctl status deadline-worker
   ```

   Der Befehl sollte eine Antwort wie die folgende zurückgeben:

   ```
   Active: active (running) since Wed 2023-06-14 14:44:27 UTC; 7min ago
   ```

Wenn die Antwort nicht so aussieht, überprüfen Sie die Protokolldatei mit dem folgenden Befehl:

```
tail -n 25 /var/log/amazon/deadline/worker-agent.log
```

Windows

Von einer Administrator-Befehlszeile aus oder PowerShell Terminal, geben Sie den folgenden Befehl ein:

```
sc.exe query DeadlineWorker
```

Der Befehl sollte eine Antwort wie die folgende zurückgeben:

```
STATE   : 4 RUNNING
```

Wenn die Antwort keine enthältRUNNING, überprüfen Sie die Worker-Protokolldatei. Öffne und verwalte PowerShell fordern Sie den folgenden Befehl auf und führen Sie ihn aus:

```
Get-Content -Tail 25 -Path $env:PROGRAMDATA\Amazon\Deadline\Logs\worker-
agent.log
```

3. Reichen Sie Jobs in die Warteschlange ein, die mit Ihrer Flotte verknüpft ist. Die Jobs sollten repräsentativ für die Jobs sein, die von der Flotte verarbeitet werden.
4. Überwachen Sie den Fortschritt des Jobs mithilfe des Deadline Cloud-Monitors oder der CLI. Wenn ein Job fehlschlägt, überprüfen Sie die Sitzungs- und Worker-Protokolle.
5. Aktualisieren Sie die Konfiguration des Worker-Hosts nach Bedarf, bis die Jobs erfolgreich abgeschlossen wurden.
6. Wenn die Testjobs erfolgreich waren, können Sie den Worker beenden:

   Linux

   Führen Sie in einer Root-Shell den folgenden Befehl aus:

   ```
   systemctl stop deadline-worker
   ```

Windows

Von einer Administrator-Befehlszeile aus oder PowerShell Terminal, geben Sie den folgenden Befehl ein:

```
sc.exe stop DeadlineWorker
```

Erstelle eine Amazon Machine Image

Um ein zu erstellen Amazon Machine Image (AMI) zur Verwendung in einer kundenverwalteten Amazon Elastic Compute Cloud (Amazon EC2) -Flotte (CMF), führen Sie die Aufgaben in diesem Abschnitt aus. Sie müssen eine EC2 Amazon-Instance erstellen, bevor Sie fortfahren können. Weitere Informationen finden Sie unter Starten Ihrer Instance im EC2 Amazon-Benutzerhandbuch für Linux-Instances.

> **Important**
>
> Erstellen eines AMI erstellt einen Snapshot der angehängten Volumes der EC2 Amazon-Instance. Jegliche Software, die auf der Instance installiert ist, bleibt bestehen, sodass Instances wiederverwendet werden, wenn Sie Instances von AMI. Wir empfehlen, eine Patch-Strategie zu verfolgen und alle neuen regelmäßig zu aktualisieren AMI mit aktualisierter Software, bevor Sie sie auf Ihre Flotte anwenden.

Bereiten Sie die EC2 Amazon-Instance vor

Bevor Sie eine erstellen AMI, müssen Sie den Arbeiterstatus löschen. Der Worker-Status bleibt auch zwischen den Starts des Worker-Agents bestehen. Wenn dieser Status auch auf dem AMI, dann haben alle Instances, die von dort aus gestartet werden, denselben Status.

Wir empfehlen außerdem, alle vorhandenen Protokolldateien zu löschen. Protokolldateien können auf einer EC2 Amazon-Instance verbleiben, wenn Sie das AMI vorbereiten. Durch das Löschen dieser Dateien wird Verwirrung bei der Diagnose möglicher Probleme in Arbeiterflotten, die das AMI verwenden, minimiert.

Sie sollten auch den Worker-Agent-Systemdienst aktivieren, damit der Deadline Cloud-Worker-Agent gestartet wird, wenn Amazon gestartet EC2 wird.

Schließlich empfehlen wir Ihnen, das auto Herunterfahren des Worker-Agents zu aktivieren. Auf diese Weise kann die Worker-Flotte bei Bedarf hochskaliert und heruntergefahren werden, wenn der Rendering-Job abgeschlossen ist. Diese auto Skalierung stellt sicher, dass Sie Ressourcen nur nach Bedarf verwenden.

So bereiten Sie die EC2 Amazon-Instance vor

1. Öffnen Sie die EC2 Amazon-Konsole.
2. Starten Sie eine EC2 Amazon-Instance. Weitere Informationen finden Sie unter Starten Sie Ihre Instance.
3. Richten Sie den Host so ein, dass er eine Verbindung zu Ihrem Identity Provider (IdP) herstellt, und mounten Sie dann jedes gemeinsam genutzte Dateisystem, das er benötigt.
4. Folgen Sie den Anleitungen zu Installieren Sie den Deadline Cloud Worker AgentWorker Agent konfigurieren, dann und. Erstellen Sie Job-Benutzer und -Gruppen
5. Wenn Sie eine vorbereiten AMI basiert auf Amazon Linux 2023, um Software auszuführen, die mit der VFX Reference Platform kompatibel ist, müssen Sie mehrere Anforderungen aktualisieren. Weitere Informationen finden Sie unter Kompatibilität der VFX Reference Platform im AWS Deadline Cloud-Benutzerhandbuch.
6. Öffnen Sie ein -Terminalfenster.
   a. Öffnen Sie unter Linux ein Terminal als `root` Benutzer (oder verwenden Sie`sudo`/`su`)
   b. Ein Windows, öffnen Sie eine Administrator-Befehlszeile oder ein PowerShell Terminal.
7. Stellen Sie sicher, dass der Worker-Dienst nicht läuft und nicht so konfiguriert ist, dass er beim Booten gestartet wird:
   a. Führen Sie unter Linux Folgendes aus

```
systemctl stop deadline-worker
systemctl enable deadline-worker
```

   b. Ein Windows, lauf

```
sc.exe stop DeadlineWorker
sc.exe config DeadlineWorker start= auto
```

8. Löscht den Arbeiterstatus.
   a. Führen Sie unter Linux Folgendes aus

```
rm -rf /var/lib/deadline/*
```

b. Ein Windows, lauf

```
del /Q /S %PROGRAMDATA%\Amazon\Deadline\Cache\*
```

9. Löschen Sie die Protokolldateien.

a. Führen Sie unter Linux Folgendes aus

```
rm -rf /var/log/amazon/deadline/*
```

b. Ein Windows, lauf

```
del /Q /S %PROGRAMDATA%\Amazon\Deadline\Logs\*
```

10. Ein Windows, es wird empfohlen, die Anwendung Amazon EC2 Launch Settings im Startmenü auszuführen, um die letzte Vorbereitung und das Herunterfahren der Instance auf dem Host abzuschließen.

> **Note**
>
> Sie MÜSSEN Shutdown without Sysprep wählen und dürfen niemals Shutdown with Sysprep wählen. Beim Herunterfahren mit Sysprep werden alle lokalen Benutzer unbrauchbar. Weitere Informationen finden Sie im Abschnitt Bevor Sie beginnen des Themas Erstellen eines benutzerdefinierten AMIs im Benutzerhandbuch für Windows-Instances.

Erstellen Sie das AMI

Um das zu bauen AMI

1. Öffnen Sie die EC2 Amazon-Konsole.
2. Wählen Sie im Navigationsbereich Instances und dann Ihre Instance aus.
3. Wählen Sie Instanzstatus und dann Instanz beenden aus.
4. Nachdem die Instance gestoppt wurde, wählen Sie Actions aus.
5. Wählen Sie „Bild und Vorlagen“ und anschließend „Bild erstellen“.

6. Geben Sie einen Bildnamen ein.
7. (Optional) Geben Sie eine Beschreibung für Ihr Bild ein.
8. Wählen Sie Create Image (Image erstellen) aus.

Erstellen Sie eine Flotteninfrastruktur mit einer Amazon EC2 Auto Scaling Scaling-Gruppe

In diesem Abschnitt wird erklärt, wie Sie eine Amazon EC2 Auto Scaling Scaling-Flotte erstellen.

Verwenden Sie die folgende CloudFormation YAML-Vorlage, um eine Amazon EC2 Auto Scaling (Auto Scaling) -Gruppe, eine Amazon Virtual Private Cloud (Amazon VPC) mit zwei Subnetzen, einem Instance-Profil und einer Instance-Zugriffsrolle zu erstellen. Diese sind erforderlich, um die Instance mithilfe von Auto Scaling in den Subnetzen zu starten.

Sie sollten die Liste der Instance-Typen überprüfen und aktualisieren, damit sie Ihren Rendering-Anforderungen entspricht.

Eine vollständige Erläuterung der in der CloudFormation YAML-Vorlage verwendeten Ressourcen und Parameter finden Sie in der Referenz zum Deadline Cloud-Ressourcentyp im AWS CloudFormation Benutzerhandbuch.

So erstellen Sie eine Amazon EC2 Auto Scaling Scaling-Flotte

1. Verwenden Sie das folgende Beispiel, um eine CloudFormation Vorlage zu erstellen, die die `AMIId` Parameter `FarmIDFleetID`, und definiert. Speichern Sie die Vorlage in einer `.YAML` Datei auf Ihrem lokalen Computer.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Amazon Deadline Cloud customer-managed fleet
Parameters:
  FarmId:
    Type: String
    Description: Farm ID
  FleetId:
    Type: String
    Description: Fleet ID
  AMIId:
    Type: String
    Description: AMI ID for launching workers
```

```
Resources:
  deadlineVPC:
    Type: 'AWS::EC2::VPC'
    Properties:
      CidrBlock: 100.100.0.0/16
  deadlineWorkerSecurityGroup:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      GroupDescription: !Join
        - ' '
        - - Security group created for Deadline Cloud workers in the fleet
          - !Ref FleetId
      GroupName: !Join
        - ''
        - - deadlineWorkerSecurityGroup-
          - !Ref FleetId
      SecurityGroupEgress:
        - CidrIp: 0.0.0.0/0
          IpProtocol: '-1'
      SecurityGroupIngress: []
      VpcId: !Ref deadlineVPC
  deadlineIGW:
    Type: 'AWS::EC2::InternetGateway'
    Properties: {}
  deadlineVPCGatewayAttachment:
    Type: 'AWS::EC2::VPCGatewayAttachment'
    Properties:
      VpcId: !Ref deadlineVPC
      InternetGatewayId: !Ref deadlineIGW
  deadlinePublicRouteTable:
    Type: 'AWS::EC2::RouteTable'
    Properties:
      VpcId: !Ref deadlineVPC
  deadlinePublicRoute:
    Type: 'AWS::EC2::Route'
    Properties:
      RouteTableId: !Ref deadlinePublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref deadlineIGW
    DependsOn:
      - deadlineIGW
      - deadlineVPCGatewayAttachment
  deadlinePublicSubnet0:
    Type: 'AWS::EC2::Subnet'
```

```
    Properties:
      VpcId: !Ref deadlineVPC
      CidrBlock: 100.100.16.0/22
      AvailabilityZone: !Join
        - ''
        - - !Ref 'AWS::Region'
          - a
  deadlineSubnetRouteTableAssociation0:
    Type: 'AWS::EC2::SubnetRouteTableAssociation'
    Properties:
      RouteTableId: !Ref deadlinePublicRouteTable
      SubnetId: !Ref deadlinePublicSubnet0
  deadlinePublicSubnet1:
    Type: 'AWS::EC2::Subnet'
    Properties:
      VpcId: !Ref deadlineVPC
      CidrBlock: 100.100.20.0/22
      AvailabilityZone: !Join
        - ''
        - - !Ref 'AWS::Region'
          - c
  deadlineSubnetRouteTableAssociation1:
    Type: 'AWS::EC2::SubnetRouteTableAssociation'
    Properties:
      RouteTableId: !Ref deadlinePublicRouteTable
      SubnetId: !Ref deadlinePublicSubnet1
  deadlineInstanceAccessAccessRole:
    Type: 'AWS::IAM::Role'
    Properties:
      RoleName: !Join
        - '-'
        - - deadline
          - InstanceAccess
          - !Ref FleetId
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Principal:
              Service: ec2.amazonaws.com
            Action:
              - 'sts:AssumeRole'
      Path: /
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy'
```

```
      - 'arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore'
      - 'arn:aws:iam::aws:policy/AWSDeadlineCloud-WorkerHost'
  deadlineInstanceProfile:
    Type: 'AWS::IAM::InstanceProfile'
    Properties:
      Path: /
      Roles:
        - !Ref deadlineInstanceAccessAccessRole
  deadlineLaunchTemplate:
    Type: 'AWS::EC2::LaunchTemplate'
    Properties:
      LaunchTemplateName: !Join
        - ''
        - - deadline-LT-
          - !Ref FleetId
      LaunchTemplateData:
        NetworkInterfaces:
          - DeviceIndex: 0
            AssociatePublicIpAddress: true
            Groups:
              - !Ref deadlineWorkerSecurityGroup
            DeleteOnTermination: true
        ImageId: !Ref AMIId
        InstanceInitiatedShutdownBehavior: terminate
        IamInstanceProfile:
          Arn: !GetAtt
            - deadlineInstanceProfile
            - Arn
        MetadataOptions:
          HttpTokens: required
          HttpEndpoint: enabled

  deadlineAutoScalingGroup:
    Type: 'AWS::AutoScaling::AutoScalingGroup'
    Properties:
      AutoScalingGroupName: !Join
        - ''
        - - deadline-ASG-autoscalable-
          - !Ref FleetId
      MinSize: 0
      MaxSize: 10
      VPCZoneIdentifier:
        - !Ref deadlinePublicSubnet0
        - !Ref deadlinePublicSubnet1
```

```
      NewInstancesProtectedFromScaleIn: true
      MixedInstancesPolicy:
        InstancesDistribution:
          OnDemandBaseCapacity: 0
          OnDemandPercentageAboveBaseCapacity: 0
          SpotAllocationStrategy: capacity-optimized
          OnDemandAllocationStrategy: lowest-price
        LaunchTemplate:
          LaunchTemplateSpecification:
            LaunchTemplateId: !Ref deadlineLaunchTemplate
            Version: !GetAtt
              - deadlineLaunchTemplate
              - LatestVersionNumber
          Overrides:
            - InstanceType: m5.large
            - InstanceType: m5d.large
            - InstanceType: m5a.large
            - InstanceType: m5ad.large
            - InstanceType: m5n.large
            - InstanceType: m5dn.large
            - InstanceType: m4.large
            - InstanceType: m3.large
            - InstanceType: r5.large
            - InstanceType: r5d.large
            - InstanceType: r5a.large
            - InstanceType: r5ad.large
            - InstanceType: r5n.large
            - InstanceType: r5dn.large
            - InstanceType: r4.large
      MetricsCollection:
        - Granularity: 1Minute
          Metrics:
            - GroupMinSize
            - GroupMaxSize
            - GroupDesiredCapacity
            - GroupInServiceInstances
            - GroupTotalInstances
            - GroupInServiceCapacity
            - GroupTotalCapacity
```

2. Öffnen Sie die CloudFormation Konsole unter https://console.aws.amazon.com/cloudformation.

   Verwenden Sie die CloudFormation Konsole, um einen Stack anhand der Anweisungen zum Hochladen der von Ihnen erstellten Vorlagendatei zu erstellen. Weitere Informationen

finden Sie im AWS CloudFormation Benutzerhandbuch unter Erstellen eines Stacks auf der CloudFormation Konsole.

> **Note**
>
> - Anmeldeinformationen aus der IAM-Rolle, die an die Amazon EC2 EC2-Instance Ihres Workers angehängt sind, sind für alle Prozesse verfügbar, die auf diesem Worker ausgeführt werden, einschließlich Jobs. Der Mitarbeiter sollte über die geringsten Betriebsberechtigungen verfügen: und `deadline:CreateWorker` `deadline:AssumeFleetRoleForWorker.`
> - Der Worker-Agent ruft die Anmeldeinformationen für die Warteschlangenrolle ab und konfiguriert sie für die Verwendung bei der Ausführung von Jobs. Die Amazon EC2 EC2-Instance-Profilrolle sollte keine Berechtigungen enthalten, die für Ihre Jobs erforderlich sind.

Skalieren Sie Ihre Amazon EC2 EC2-Flotte automatisch mit der Deadline Cloud-Funktion für Skalierungsempfehlungen

Deadline Cloud nutzt eine Amazon EC2 Auto Scaling (Auto Scaling) -Gruppe, um die vom Kunden verwaltete Amazon EC2 EC2-Flotte (CMF) automatisch zu skalieren. Sie müssen den Flottenmodus konfigurieren und die erforderliche Infrastruktur in Ihrem Konto bereitstellen, damit Ihre Flotte auto skaliert werden kann. Die von Ihnen bereitgestellte Infrastruktur funktioniert für alle Flotten, sodass Sie sie nur einmal einrichten müssen.

Der grundlegende Arbeitsablauf ist: Sie konfigurieren Ihren Flottenmodus so, dass er auto skaliert, und dann sendet Deadline Cloud ein EventBridge Ereignis für diese Flotte aus, wenn sich die empfohlene Flottengröße ändert (ein Ereignis enthält die Flotten-ID, die empfohlene Flottengröße und andere Metadaten). Sie werden eine EventBridge Regel haben, um die relevanten Ereignisse zu filtern, und ein Lambda, um sie zu verarbeiten. Das Lambda wird in Amazon EC2 Auto Scaling integriert, `AutoScalingGroup` um die Amazon EC2 EC2-Flotte automatisch zu skalieren.

Stellen Sie den Flottenmodus auf **`EVENT_BASED_AUTO_SCALING`**

Konfigurieren Sie Ihren Flottenmodus auf`EVENT_BASED_AUTO_SCALING`. Sie können dazu die Konsole verwenden oder die verwenden, AWS CLI um die `CreateFleet` `UpdateFleet` OR-API

direkt aufzurufen. Nachdem der Modus konfiguriert wurde, beginnt Deadline Cloud mit dem Senden von EventBridge Ereignissen, sobald sich die empfohlene Flottengröße ändert.

- `UpdateFleet`Beispielbefehl:

```
aws deadline update-fleet \
  --farm-id FARM_ID \
  --fleet-id FLEET_ID \
  --configuration file://configuration.json
```

- `CreateFleet`Beispielbefehl:

```
aws deadline create-fleet \
  --farm-id FARM_ID \
  --display-name "Fleet name" \
  --max-worker-count 10 \
  --configuration file://configuration.json
```

Das Folgende ist ein Beispiel für die `configuration.json` Verwendung in den obigen CLI-Befehlen (`--configuration file://configuration.json`).

- Um Auto Scaling für Ihre Flotte zu aktivieren, sollten Sie den Modus auf einstellen`EVENT_BASED_AUTO_SCALING`.
- Dies `workerCapabilities` sind die Standardwerte, die dem CMF bei der Erstellung zugewiesen wurden. Sie können diese Werte ändern, wenn Sie mehr Ressourcen benötigen, die Ihrem CMF zur Verfügung stehen.

Nachdem Sie den Flottenmodus konfiguriert haben, sendet Deadline Cloud Ereignisse mit Empfehlungen zur Flottengröße für diese Flotte aus.

```
{
    "customerManaged": {
        "mode": "EVENT_BASED_AUTO_SCALING",
        "workerCapabilities": {
            "vCpuCount": {
                "min": 1,
                "max": 4
            },
            "memoryMiB": {
```

```
                "min": 1024,
                "max": 4096
            },
            "osFamily": "linux",
            "cpuArchitectureType": "x86_64"
        }
    }
}
```

Stellen Sie den Auto Scaling Scaling-Stack mithilfe der CloudFormation Vorlage bereit

Sie können eine EventBridge Regel zum Filtern von Ereignissen, ein Lambda zum Verwerten der Ereignisse und zur Steuerung von Auto Scaling und eine SQS-Warteschlange zum Speichern unverarbeiteter Ereignisse einrichten. Verwenden Sie die folgende CloudFormation Vorlage, um alles in einem Stack bereitzustellen. Nachdem Sie die Ressourcen erfolgreich bereitgestellt haben, können Sie einen Auftrag einreichen und die Flotte wird automatisch skaliert.

```
Resources:
  AutoScalingLambda:
    Type: 'AWS::Lambda::Function'
    Properties:
      Code:
        ZipFile: |-
          """
          This lambda is configured to handle "Fleet Size Recommendation Change"
          messages. It will handle all such events, and requires
          that the ASG is named based on the fleet id. It will scale up/down the fleet
          based on the recommended fleet size in the message.

          Example EventBridge message:
          {
              "version": "0",
              "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
              "detail-type": "Fleet Size Recommendation Change",
              "source": "aws.deadline",
              "account": "111122223333",
              "time": "2017-12-22T18:43:48Z",
              "region": "us-west-1",
              "resources": [],
              "detail": {
                  "farmId": "farm-12345678900000000000000000000000",
                  "fleetId": "fleet-12345678900000000000000000000000",
                  "oldFleetSize": 1,
```

```
                "newFleetSize": 5,
            }
        }
        """

        import json
        import boto3
        import logging

        logger = logging.getLogger()
        logger.setLevel(logging.INFO)

        auto_scaling_client = boto3.client("autoscaling")

        def lambda_handler(event, context):
            logger.info(event)
            event_detail = event["detail"]
            fleet_id = event_detail["fleetId"]
            desired_capacity = event_detail["newFleetSize"]

            asg_name = f"deadline-ASG-autoscalable-{fleet_id}"
            auto_scaling_client.set_desired_capacity(
                AutoScalingGroupName=asg_name,
                DesiredCapacity=desired_capacity,
                HonorCooldown=False,
            )

            return {
                'statusCode': 200,
                'body': json.dumps(f'Successfully set desired_capacity for {asg_name}
to {desired_capacity}')
            }
     Handler: index.lambda_handler
     Role: !GetAtt
       - AutoScalingLambdaServiceRole
       - Arn
     Runtime: python3.11
   DependsOn:
     - AutoScalingLambdaServiceRoleDefaultPolicy
     - AutoScalingLambdaServiceRole
 AutoScalingEventRule:
   Type: 'AWS::Events::Rule'
   Properties:
     EventPattern:
```

```
      source:
        - aws.deadline
      detail-type:
        - Fleet Size Recommendation Change
    State: ENABLED
    Targets:
      - Arn: !GetAtt
          - AutoScalingLambda
          - Arn
        DeadLetterConfig:
          Arn: !GetAtt
            - UnprocessedAutoScalingEventQueue
            - Arn
        Id: Target0
        RetryPolicy:
          MaximumRetryAttempts: 15
AutoScalingEventRuleTargetPermission:
  Type: 'AWS::Lambda::Permission'
  Properties:
    Action: 'lambda:InvokeFunction'
    FunctionName: !GetAtt
      - AutoScalingLambda
      - Arn
    Principal: events.amazonaws.com
    SourceArn: !GetAtt
      - AutoScalingEventRule
      - Arn
AutoScalingLambdaServiceRole:
  Type: 'AWS::IAM::Role'
  Properties:
    AssumeRolePolicyDocument:
      Statement:
        - Action: 'sts:AssumeRole'
          Effect: Allow
          Principal:
            Service: lambda.amazonaws.com
      Version: 2012-10-17
    ManagedPolicyArns:
      - !Join
        - ''
        - - 'arn:'
          - !Ref 'AWS::Partition'
          - ':iam::aws:policy/service-role/AWSLambdaBasicExecutionRole'
AutoScalingLambdaServiceRoleDefaultPolicy:
```

```
    Type: 'AWS::IAM::Policy'
    Properties:
      PolicyDocument:
        Statement:
          - Action: 'autoscaling:SetDesiredCapacity'
            Effect: Allow
            Resource: '*'
        Version: 2012-10-17
      PolicyName: AutoScalingLambdaServiceRoleDefaultPolicy
      Roles:
        - !Ref AutoScalingLambdaServiceRole
  UnprocessedAutoScalingEventQueue:
    Type: 'AWS::SQS::Queue'
    Properties:
      QueueName: deadline-unprocessed-autoscaling-events
    UpdateReplacePolicy: Delete
    DeletionPolicy: Delete
  UnprocessedAutoScalingEventQueuePolicy:
    Type: 'AWS::SQS::QueuePolicy'
    Properties:
      PolicyDocument:
        Statement:
          - Action: 'sqs:SendMessage'
            Condition:
              ArnEquals:
                'aws:SourceArn': !GetAtt
                  - AutoScalingEventRule
                  - Arn
            Effect: Allow
            Principal:
              Service: events.amazonaws.com
            Resource: !GetAtt
              - UnprocessedAutoScalingEventQueue
              - Arn
        Version: 2012-10-17
      Queues:
        - !Ref UnprocessedAutoScalingEventQueue
```

Führen Sie einen Flottenzustandscheck durch

Nachdem Sie Ihre Flotte erstellt haben, sollten Sie einen individuellen Zustandscheck erstellen, um sicherzustellen, dass Ihre Flotte fehlerfrei bleibt und keine blockierten Instanzen auftreten, um unnötige Kosten zu vermeiden. Weitere Informationen finden Sie unter Bereitstellung eines Deadline

Cloud-Flottenzustandschecks. GitHub Eine Integritätsprüfung kann das Risiko verringernAmazon Machine Image, dass eine versehentliche Änderung Ihrer Startvorlage oder Netzwerkkonfiguration unentdeckt ausgeführt wird.

Konfiguration und Verwendung von vom Service verwalteten Deadline Cloud-Flotten

Eine Service-Managed Fleet (SMF) ist eine Sammlung von Mitarbeitern, die von Deadline Cloud verwaltet werden. Ein SMF macht die Verwaltung der Flottenskalierung zur Bearbeitung von Anforderungen oder die Reduzierung der Flottengröße nach Abschluss einer Aufgabe überflüssig.

Wenn ein SMF mithilfe der standardmäßigen Conda-Warteschlangenumgebung einer Warteschlange zugeordnet wird, konfiguriert Deadline Cloud die Mitarbeiter in der Flotte mit dem entsprechenden Softwarepaket. Informationen zu unterstützten Partneranwendungen finden Sie unter Standard-Conda-Warteschlangenumgebung im AWS Deadline Cloud-Benutzerhandbuch.

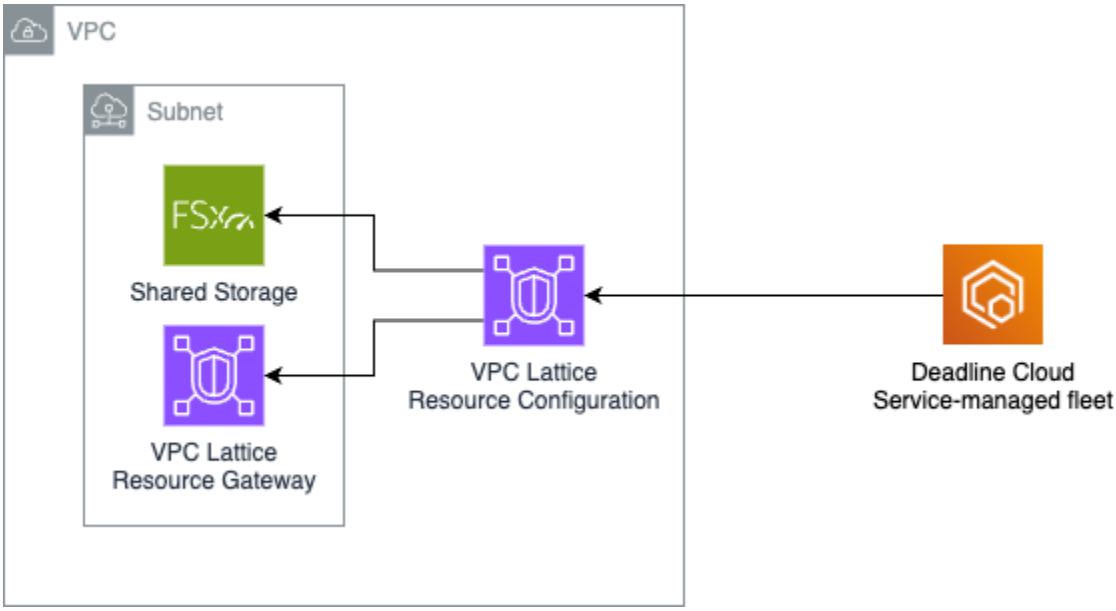
In den meisten Fällen müssen Sie ein SMF nicht ändern, um Ihre Workloads zu verarbeiten. In einigen Situationen müssen Sie jedoch möglicherweise Änderungen an Ihren Flotten vornehmen.

> **Note**
>
> Informationen zur Installation benutzerdefinierter Software auf Workern mithilfe von Host-Konfigurationsskripten finden Sie unterFühren Sie Host-Konfigurationsskripten mit Administratorrechten aus.

Themen

Connect VPC-Ressourcen mit Ihrem SMF mit VPC-Ressourcenendpunkten

Mit Amazon VPC-Ressourcenendpunkten für Deadline Cloud Service-Managed Fleets (SMF) können Sie Ihre VPC-Ressourcen wie Netzwerkdateisysteme (NFS), Lizenzserver und Datenbanken mit Ihren Deadline Cloud-Workern verbinden. Mit dieser Funktion können Sie die vollständig verwaltete Plattform von Deadline Cloud nutzen und sich gleichzeitig in Ihre bestehende Infrastruktur innerhalb einer VPC integrieren.



> **Tip**
>
> Eine CloudFormation Referenzvorlage, die einen FSx Amazon-Cluster einrichtet und ihn mit einer serviceverwalteten Flotte verbindet, finden Sie unter smf_vpc_fsx im Deadline Cloud-Beispiel-Repository unter. GitHub

So funktionieren VPC-Ressourcenendpunkte

VPC-Ressourcenendpunkte verwenden VPC Lattice, um eine sichere Verbindung zwischen Ihren Deadline Cloud SMF-Workern und Ressourcen in Ihrer VPC herzustellen. Die Verbindung ist unidirektional, was bedeutet, dass Worker eine Verbindung zu Ressourcen in Ihrer VPC herstellen und Daten hin und her übertragen können, Ressourcen in Ihrer VPC jedoch keine Verbindung zu einem Worker herstellen können.

Wenn Sie eine VPC-Ressource mit einer vom Service verwalteten Flotte von Deadline Cloud verbinden, können Ihre Mitarbeiter über einen privaten Domainnamen auf Ressourcen in Ihrer VPC zugreifen. Darüber hinaus fließt der Datenverkehr über VPC Lattice von Workern zu Ihren VPC-Ressourcen, und Ressourcen in Ihrer VPC sehen den Datenverkehr, der vom VPC Lattice Resource Gateway kommt.

Weitere Informationen finden Sie im VPC Lattice-Benutzerhandbuch.

Voraussetzungen

Bevor Sie VPC-Ressourcen mit Ihrer vom Service verwalteten Deadline Cloud-Flotte verbinden, stellen Sie sicher, dass Sie über Folgendes verfügen:

- Ein AWS Konto mit einer VPC, die Ressourcen enthält, die Sie verbinden möchten.
- IAM-Berechtigungen zum Erstellen und Verwalten von VPC-Lattice-Ressourcen.
- Eine Deadline Cloud-Farm mit mindestens einer vom Service verwalteten Flotte.
- VPC-Ressourcen, die Sie zugänglich machen möchten (FSx, NFS, Lizenzserver usw.).

Richten Sie einen VPC-Ressourcenendpunkt ein

Um einen VPC-Ressourcenendpunkt einzurichten, müssen Sie Ressourcen in VPC Lattice erstellen und AWS RAMdiese Ressourcen dann mit Ihrer Flotte in Deadline Cloud verbinden. Gehen Sie wie folgt vor, um einen VPC-Ressourcenendpunkt für Ihr SMF einzurichten.

1. Informationen zum Erstellen eines Ressourcen-Gateways in VPC Lattice finden Sie unter Erstellen eines Ressourcen-Gateways im VPC Lattice-Benutzerhandbuch.
2. Informationen zum Erstellen einer Ressourcenkonfiguration in VPC Lattice finden Sie unter Erstellen einer Ressourcenkonfiguration im VPC Lattice-Benutzerhandbuch.
3. Um die Ressource mit Ihrer Deadline Cloud-Flotte zu teilen, erstellen Sie eine Ressourcenfreigabe in. AWS RAM Eine Anleitung dazu finden Sie unter Eine gemeinsame Ressource erstellen.

   Wählen Sie beim Erstellen einer Ressourcenfreigabe für Principals die Option Service Principal aus der Dropdownliste aus und geben Sie dann die Eingabetaste ein. **`fleets.deadline.amazonaws.com`**
4. Gehen Sie wie folgt vor, um die Ressourcenkonfiguration mit Ihrer Deadline Cloud-Flotte zu verbinden.

   a. Falls Sie dies noch nicht getan haben, öffnen Sie die Deadline Cloud-Konsole.

   b. Wählen Sie im Navigationsbereich Farmen und dann Ihre Farm aus.

   c. Wählen Sie die Registerkarte Flotten und dann Ihre Flotte aus.

   d. Wechseln Sie auf die Registerkarte Configurations (Konfiguration).

   e. Wählen Sie unter VPC-Ressourcenendpunkte die Option Bearbeiten aus.

f. Wählen Sie die Ressourcenkonfiguration aus, die Sie erstellt haben, und wählen Sie dann Änderungen speichern aus.

Zugreifen auf Ihre VPC-Ressourcen

Nachdem Sie Ihre VPC-Ressource mit Ihrer Flotte verbunden haben, können Mitarbeiter über einen privaten Domainnamen im folgenden Format darauf zugreifen: `<resource_config_id>.resource-endpoints.deadline.<region>.amazonaws.com`

Diese Domain ist privat und nur für Mitarbeiter zugänglich (nicht über das Internet oder Ihre Workstation).

Verwenden Sie ein Host-Konfigurationsskript, um den Zugriff auf die VPC-Ressource auf Ihren Workern bereitzustellen oder zu konfigurieren. Host-Konfigurationsskripten werden beim Start von Workern mit Administratorrechten ausgeführt und ermöglichen es Ihnen, Dateisysteme zu mounten, Netzwerkeinstellungen zu konfigurieren oder andere Einrichtungsaufgaben auszuführen.

Authentifizierung und Sicherheit

Speichern Sie Anmeldeinformationen für Ressourcen, die eine Authentifizierung erfordern, sicher im AWS Secrets Manager, greifen Sie auf geheime Daten aus Ihren Host-Konfigurationsskripten oder Jobskripten zu und implementieren Sie entsprechende Dateisystemberechtigungen, um den Zugriff zu kontrollieren. Berücksichtigen Sie die Auswirkungen auf die Sicherheit, wenn Sie Ressourcen über mehrere Flotten gemeinsam nutzen. Wenn beispielsweise zwei Flotten mit demselben gemeinsam genutzten Speicher verbunden sind, können Jobs, die auf einer Flotte ausgeführt werden, möglicherweise auf Ressourcen zugreifen, die mit der anderen Flotte erstellt wurden.

Technische Überlegungen

Beachten Sie bei der Verwendung von VPC-Ressourcenendpunkten Folgendes:

- Verbindungen können nur von Workern zu VPC-Ressourcen initiiert werden, nicht von VPC-Ressourcen zu Workern.
- Sobald eine Verbindung hergestellt wurde, bleibt sie bestehen, bis sie zurückgesetzt wird, auch wenn die Ressourcenkonfiguration unterbrochen wird.
- Die VPC-Lattice-Verbindung verarbeitet Verbindungen zwischen Availability Zones automatisch und ohne zusätzliche Gebühren. Ihr Ressourcen-Gateway muss sich eine Availability Zone mit

Ihrer VPC-Ressource teilen. Wir empfehlen daher, das Ressourcen-Gateway so zu konfigurieren, dass es sich über alle Availability Zones erstreckt.

- Der Datenverkehr, der über den VPC-Ressourcenendpunkt geleitet wird, verwendet Network Address Translation (NAT), was nicht mit allen Anwendungsfällen kompatibel ist. Beispielsweise kann Microsoft Active Directory (AD) keine Verbindung über NAT herstellen.

Weitere Informationen zu VPC-Lattice-Kontingenten finden Sie unter Quotas for VPC Lattice.

Fehlerbehebung

Wenn Sie Probleme mit VPC-Ressourcenendpunkten haben, überprüfen Sie Folgendes.

- Wenn Sie eine Fehlermeldung wie „mount.nfs: access denied by server while mount“ erhalten, müssen Sie möglicherweise die Client-Konfiguration Ihres NFS-Volumes aktualisieren.
- Überprüfen Sie die Einrichtung Ihrer Ressourcenkonfiguration, indem Sie Tests von einer Amazon EC2 EC2-Instance oder AWS CloudShell in Ihrer VPC aus durchführen.
- Testen Sie Ihre Deadline Cloud-Verbindung mit einfachen CLI-Jobs. Weitere Informationen finden Sie unter Deadline Cloud-Beispiele auf GitHub.
- Überprüfen Sie die Einstellungen in der Sicherheitsgruppe des Ressourcen-Gateways, falls Verbindungsfehler auftreten.
- Aktivieren Sie VPC-Zugriffsprotokolle, um Verbindungen zu überwachen.

Verwenden Sie Auftragsanhänge für vom Service verwaltete Flotten

Jobanhänge übertragen Dateien zwischen Ihrer Workstation und Deadline Cloud-Mitarbeitern mithilfe von Amazon Simple Storage Service (Amazon S3). Sie können Jobanhänge allein oder zusammen mit gemeinsam genutztem Speicher verwenden, um Zusatzdaten an Jobs anzuhängen, die nicht mit anderen Jobs geteilt werden, wie z. B. Jobskripte, Konfigurationsdateien oder lokal gespeicherte Projektelemente.

Informationen zur Funktionsweise von Stellenanhängen finden Sie unter Jobanhänge im Deadline Cloud-Benutzerhandbuch. Einzelheiten zur Angabe von Eingabe- und Ausgabedateien in Job-Bundles finden Sie unterVerwenden Sie Jobanhänge, um Dateien zu teilen.

Wählen Sie einen Dateisystemmodus

Wenn Sie einen Job mit Anhängen einreichen, können Sie wählen, wie Mitarbeiter Dateien aus Amazon S3 laden, indem Sie die `fileSystem` Eigenschaft festlegen:

- KOPIERT (Standard) — Lädt alle Dateien auf die lokale Festplatte herunter, bevor die Aufgaben beginnen. Optimal, wenn für jede Aufgabe die meisten Eingabedateien benötigt werden.
- VIRTUELL — Hängt ein virtuelles Dateisystem ein, das Dateien bei Bedarf herunterlädt. Optimal, wenn Aufgaben nur eine Teilmenge von Eingabedateien benötigen. Nur für Linux-SMF-Worker verfügbar.

> **Important**
>
> Caching im VIRTUELLEN Modus kann den Speicherverbrauch erhöhen und ist nicht für alle Workloads optimiert. Wir empfehlen Ihnen, Ihre Arbeitslast zu testen, bevor Sie Produktionsjobs ausführen.

Ausführliche Informationen zur Konfiguration des Dateisystemmodus finden Sie unter Virtuelles Dateisystem im Deadline Cloud-Benutzerhandbuch.

Optimieren Sie die Übertragungsleistung

Die Geschwindigkeit, mit der Dateien von Amazon S3 mit SMF-Workern synchronisiert werden, hängt von der Amazon Elastic Block Store (Amazon EBS) -Volume-Konfiguration Ihrer Flotte ab. Standardmäßig verwenden SMF-Worker gp3-Amazon-EBS-Volumes mit grundlegenden Leistungseinstellungen. Bei Workloads mit großen Eingabedateien oder vielen kleinen Dateien können Sie die Übertragungsgeschwindigkeiten verbessern, indem Sie den Amazon EBS-Durchsatz und die IOPS-Einstellungen erhöhen. Sie können diese Einstellungen mithilfe von () aktualisieren. AWS Command Line Interface AWS CLI

Durchsatz (MiB/s)

Die Geschwindigkeit, mit der Daten vom Volume gelesen oder auf das Volume geschrieben werden können. Die Standardeinstellung ist 125 MiB/s, maximum is 1,000 MiB/s für GP3-Volumes. Erhöhung bei großen sequentiellen Dateiübertragungen.

E/A\Sek

Eingabe-/Ausgabevorgänge pro Sekunde. Die Standardeinstellung ist 3.000 IOPS, das Maximum ist 16.000 IOPS für GP3-Volumes. Erhöht sich, wenn viele kleine Dateien übertragen werden.

> **Note**
>
> Die Erhöhung des Amazon EBS-Durchsatzes und der IOPS erhöht die Flottenkosten. Preisinformationen finden Sie unter Deadline Cloud-Preise.

Um die Amazon EBS-Einstellungen für eine bestehende Flotte zu aktualisieren, verwenden Sie den AWS CLI

- Führen Sie den folgenden Befehl aus:

```
aws deadline update-fleet \
  --farm-id farm-0123456789abcdef0 \
  --fleet-id fleet-0123456789abcdef0 \
  --configuration '{
    "serviceManagedEc2": {
      "instanceCapabilities": {
        "vCpuCount": {"min": 4},
        "memoryMiB": {"min": 8192},
        "osFamily": "linux",
        "cpuArchitectureType": "x86_64",
        "rootEbsVolume": {
          "sizeGiB": 250,
          "iops": 6000,
          "throughputMiB": 500
        }
      },
      "instanceMarketOptions": {"type": "spot"}
    }
  }'
```

Laden Sie die Job-Ergebnisse herunter

Laden Sie nach Abschluss Ihres Jobs die Ausgabedateien mit der Deadline Cloud-CLI oder dem AWS Deadline Cloud-Monitor (Deadline Cloud-Monitor) herunter:

```
deadline job download-output --job-id job-0123456789abcdef0
```

Informationen zum automatischen Herunterladen von Ausgaben nach Abschluss von Jobs finden Sie unter Automatische Downloads im Deadline Cloud-Benutzerhandbuch.

Bereitstellung und Konfiguration benutzerdefinierter Software für Mitarbeiter

AWS Deadline Cloud bietet mehrere Methoden, um benutzerdefinierte Software, Plugins und Tools für Ihre Mitarbeiter bereitzustellen und zu konfigurieren. Welche Methode Sie wählen, hängt von Ihren Anforderungen ab, z. B. ob Sie Administratorrechte benötigen, wie oft sich die Software ändert und ob die Software für alle Jobs oder nur für bestimmte Jobs verfügbar sein soll.

Wählen Sie eine Bereitstellungsmethode

Verwenden Sie die folgende Tabelle, um die richtige Bereitstellungsmethode für Ihren Anwendungsfall auszuwählen.

Kriterien	Warteschl angenumgebung	Host-Konfiguration sskript	Benutzerdefiniertes Conda-Paket
Administratorrechte erforderl ich	Nein	Ja	Nein
Wenn es läuft	Beginn der Sitzung	Inbetriebnahme eines Arbeiters	Beginn der Sitzung
Scope	Pro Warteschlange oder Job	Alle Arbeiter in der Flotte	Pro Warteschlange oder Auftrag
Kann durch die Einreichung von Jobs gesteuert werden	Ja	Nein	Ja
Komplexität der Einrichtung	Niedrig	Medium	Hoch
Am besten geeignet für	Einfache Plugins, Skripte, Umgebungs variablen	Systemtreiber, Docker, Speicherh alterungen	Komplexe Anwendungen mit Abhängigkeiten

Kurzanleitung zur Entscheidungsfindung:

- Benötigen Sie Administrator- oder Root-Rechte? Verwenden Sie ein Host-Konfigurationsskript.
- Einfaches Plugin oder Skript ohne Administratorrechte? Verwenden Sie eine Warteschlangenumgebung.
- Komplexe Anwendung mit Anforderungen an die Versionskontrolle? Erstellen Sie ein benutzerdefiniertes Conda-Paket.

Jobs mithilfe von Warteschlangenumgebungen konfigurieren

AWS Deadline Cloud verwendet Warteschlangenumgebungen, um die Software auf Ihren Mitarbeitern zu konfigurieren. In einer Umgebung können Sie zeitaufwändige Aufgaben wie Einrichtung und Abbau einmal für alle Aufgaben in einer Sitzung ausführen. Sie definiert die Aktionen, die auf einem Worker ausgeführt werden, wenn eine Sitzung gestartet oder beendet wird. Sie können eine Umgebung für eine Warteschlange, Jobs, die in der Warteschlange ausgeführt werden, und die einzelnen Schritte für einen Job konfigurieren.

Sie definieren Umgebungen als Warteschlangenumgebungen oder Jobumgebungen. Erstellen Sie Warteschlangenumgebungen mit der Deadline Cloud-Konsole oder mit dem Deadline: CreateQueueEnvironment -Betrieb und definieren Sie Jobumgebungen in den Jobvorlagen der Jobs, die Sie einreichen. Sie folgen der Open Job Description (OpenJD) -Spezifikation für Umgebungen. Einzelheiten finden Siehttps://github.com/OpenJobDescription/openjd-specifications/wiki/2023-09-Template-Schemas#4-environment <Environment>in der OpenJD-Spezifikation unter. GitHub

Zusätzlich zu einem `name` und enthält jede Umgebung zwei Felder`description`, die die Umgebung auf dem Host definieren. Diese sind:

- `script`— Die Aktion, die ergriffen wird, wenn diese Umgebung auf einem Worker ausgeführt wird.
- `variables`— Eine Reihe von name/value Umgebungsvariablenpaaren, die beim Betreten der Umgebung festgelegt werden.

Sie müssen mindestens einen von `script` oder festlegen`variables`.

Sie können in Ihrer Jobvorlage mehr als eine Umgebung definieren. Jede Umgebung wird in der Reihenfolge angewendet, in der sie in der Vorlage aufgeführt ist. Sie können dies verwenden, um die Komplexität Ihrer Umgebungen zu bewältigen.

Die Standard-Warteschlangenumgebung für Deadline Cloud verwendet den Conda-Paketmanager, um Software in die Umgebung zu laden, aber Sie können auch andere Paketmanager verwenden.

Die Standardumgebung definiert zwei Parameter, um die Software anzugeben, die geladen werden soll. Diese Variablen werden von den von Deadline Cloud bereitgestellten Einsendern festgelegt. Sie können sie jedoch auch in Ihren eigenen Skripten und Anwendungen festlegen, die die Standardumgebung verwenden. Diese sind:

- `CondaPackages`— Eine durch Leerzeichen getrennte Liste von Conda-Paketen, die den Spezifikationen entsprechen, die für den Job installiert werden sollen. Zum Beispiel würde der Blender-Absender in Blender 3.6 zusätzliche Bilder `blender=3.6` zum Rendern hinzufügen.
- `CondaChannels`— Eine durch Leerzeichen getrennte Liste von Conda-Kanälen, aus denen Pakete installiert werden sollen. Für vom Service verwaltete Flotten werden Pakete vom Channel aus installiert. `deadline-cloud` Sie können weitere Kanäle hinzufügen.

Steuern Sie die Jobumgebung mit OpenJD-Warteschlangenumgebungen

Mithilfe von Warteschlangenumgebungen können Sie benutzerdefinierte Umgebungen für Ihre Renderaufträge definieren. Eine Warteschlangenumgebung ist eine Vorlage, die die Umgebungsvariablen, Dateizuordnungen und andere Einstellungen für Jobs steuert, die in einer bestimmten Warteschlange ausgeführt werden. Sie ermöglicht es Ihnen, die Ausführungsumgebung für die an eine Warteschlange eingereichten Jobs an die Anforderungen Ihrer Workloads anzupassen. AWS Deadline Cloud bietet drei verschachtelte Ebenen, auf die Sie Open Job Description (OpenJD) -Umgebungen anwenden können: Queue, Job und Step. Durch die Definition von Warteschlangenumgebungen können Sie eine konsistente und optimierte Leistung für verschiedene Arten von Jobs sicherstellen, die Ressourcenzuweisung optimieren und die Warteschlangenverwaltung vereinfachen.

Die Warteschlangenumgebung ist eine Vorlage, die Sie über die AWS Verwaltungskonsole oder mithilfe der an eine Warteschlange in Ihrem AWS Konto anhängen. AWS CLI Sie können eine Umgebung für eine Warteschlange erstellen, oder Sie können mehrere Warteschlangenumgebungen erstellen, die angewendet wurden, um die Ausführungsumgebung zu erstellen. Dieser Ansatz ermöglicht es Ihnen, eine Umgebung schrittweise zu erstellen und zu testen, um sicherzustellen, dass sie für Ihre Jobs ordnungsgemäß funktioniert.

Job- und Step-Umgebungen sind in der Jobvorlage definiert, mit der Sie einen Job in Ihrer Warteschlange erstellen. Die OpenJD-Syntax ist in diesen verschiedenen Formen von Umgebungen dieselbe. In diesem Abschnitt werden wir sie innerhalb von Jobvorlagen zeigen.

Themen

Legen Sie Umgebungsvariablen in einer Warteschlangenumgebung fest

Viele Anwendungen und Frameworks verwenden Umgebungsvariablen, um Funktionseinstellungen, Protokollierungsebenen und die Anzeigekonfiguration zu steuern. Sie können Open Job Description (OpenJD) -Umgebungen verwenden, um Umgebungsvariablen festzulegen, die jeder Taskbefehl innerhalb seines Gültigkeitsbereichs erbt.

Gültigkeitsbereich der Umgebungsvariablen

AWS Deadline Cloud wendet Umgebungsvariablen aus Warteschlangenumgebungen an, die Sie an eine Warteschlange anhängen. Innerhalb einer Jobvorlage können Sie mithilfe von OpenJD-Umgebungen auch Umgebungsvariablen auf Job- und Schrittebene definieren. Variablen, die in einem engeren Gültigkeitsbereich definiert sind, überschreiben Variablen mit demselben Namen aus einem breiteren Gültigkeitsbereich.

- Warteschlangenumgebung — Eine Vorlage, die Sie an eine Warteschlange in Deadline Cloud anhängen. Variablen gelten für alle Jobs, die an die Warteschlange gesendet werden. Sie können Variablen mit einer `variables` Zuordnung für feste Werte festlegen oder Skripts für dynamische Werte verwenden.
- Arbeitsumgebung — Definiert unter `jobEnvironments` in einer Jobvorlage. Variablen gelten für alle Schritte und Aufgaben im Job. Eine Variable auf Jobebene überschreibt eine Variable auf Warteschlangenebene mit demselben Namen.
- Schrittumgebung — Definiert unter `stepEnvironments` in einer Jobvorlage. Variablen gelten nur für die Aufgaben in diesem Schritt. Eine Variable auf Schrittebene überschreibt eine Variable auf Job- oder Warteschlangenebene mit demselben Namen.

Setzen von Variablen in einer Warteschlangenumgebung

Sie können Umgebungsvariablen in einer Warteschlangenumgebung mithilfe einer `variables` Map für feste Werte oder `script` mithilfe einer `onEnter` With-Aktion für dynamische Werte festlegen.

Die folgende Vorlage für die Warteschlangenumgebung verwendet eine `variables` Map, auf die die `QT_QPA_PLATFORM` Variable gesetzt wird`offscreen`, sodass Anwendungen, die das Qt Framework verwenden, auf Worker-Hosts ohne interaktive Anzeige ausgeführt werden können.

```
specificationVersion: 'environment-2023-09'
environment:
  name: QtOffscreen
  variables:
    QT_QPA_PLATFORM: offscreen
```

Verwenden Sie für dynamische Werte, wie das Ändern `PATH` oder Aktivieren virtueller Umgebungen, ein Skript, das Zeilen im Format stdout `openjd_env: VAR=value` ausgibt. Das `openjd_env:` Präfix ist erforderlich. Durch die Verwendung von `echo``export`, oder anderen Shell-Mechanismen ohne das Präfix werden Variablen nicht an Jobs und Aufgaben weitergegeben.

Die folgende Vorlage für die Warteschlangenumgebung legt die `QT_QPA_PLATFORM` Variable mithilfe eines Skripts fest.

```
specificationVersion: 'environment-2023-09'
environment:
  name: QtOffscreen
  script:
    actions:
      onEnter:
        command: bash
        args:
        - "{{Env.File.Enter}}"
    embeddedFiles:
    - name: Enter
      type: TEXT
      data: |
        #!/bin/env bash
        set -euo pipefail
        echo "openjd_env: QT_QPA_PLATFORM=offscreen"
```

Um eine Warteschlangenumgebung an Ihre Warteschlange anzuhängen, verwenden Sie die Deadline Cloud-Konsole oder die AWS CLI. Weitere Informationen finden Sie unter Erstellen einer Warteschlangenumgebung im AWS Deadline Cloud-Benutzerhandbuch. Der folgende AWS CLI Befehl erstellt eine Warteschlangenumgebung aus einer Vorlagendatei.

```
aws deadline create-queue-environment \
```

```
    --farm-id FARM_ID \
    --queue-id QUEUE_ID \
    --priority 1 \
    --template-type YAML \
    --template file://my-queue-env.yaml
```

Komplexere Beispiele, wie das Erstellen und Aktivieren virtueller Conda-Umgebungen, finden Sie unter Beispiele für Deadline Cloud-Warteschlangenumgebungen unter GitHub.

Variablen in einer Jobvorlage festlegen

Fügen Sie in einer Jobvorlage eine `variables` Map zu einem `jobEnvironments` `stepEnvironments` OR-Eintrag hinzu. Jeder Eintrag ist ein Schlüssel-Wert-Paar, wobei der Schlüssel der Variablenname und der Wert der Variablenwert ist.

Die folgende Jobvorlage setzt die `QT_QPA_PLATFORM` Umgebungsvariable auf`offscreen`, sodass Anwendungen, die das Qt Framework verwenden, auf Worker-Hosts ohne interaktive Anzeige ausgeführt werden können.

```
specificationVersion: 'jobtemplate-2023-09'
name: MyJob
jobEnvironments:
- name: JobEnv
  variables:
    QT_QPA_PLATFORM: offscreen
```

Sie können mehrere Variablen in einer einzigen Umgebungsdefinition festlegen.

```
jobEnvironments:
- name: JobEnv
  variables:
    JOB_VERBOSITY: MEDIUM
    JOB_PROJECT_ID: my-project-id
    JOB_ENDPOINT_URL: https://my-host-name/my/path
    QT_QPA_PLATFORM: offscreen
```

Mithilfe der `{{Param.ParameterName}}` Syntax können Sie in Variablenwerten auf Jobparameter verweisen.

```
jobEnvironments:
- name: JobEnv
```

```
  variables:
    JOB_EXAMPLE_PARAM: "{{Param.ExampleParam}}"
```

Um eine Variable auf Jobebene für einen bestimmten Schritt zu überschreiben, definieren Sie einen `stepEnvironments` Eintrag mit demselben Variablennamen. Das folgende Beispiel definiert `JOB_PROJECT_ID` auf Auftragsebene mit dem Wert `project-12` und überschreibt dann den Wert auf Schrittebene mit. `step-project-12` Die Aufgaben im Schritt verwenden den Wert auf Schrittebene.

```
specificationVersion: 'jobtemplate-2023-09'
name: MyJob
jobEnvironments:
- name: JobEnv
  variables:
    JOB_PROJECT_ID: project-12
steps:
- name: MyStep
  stepEnvironments:
  - name: StepEnv
    variables:
      JOB_PROJECT_ID: step-project-12
```

Probieren Sie es aus: Beispiel für die Umgebungsvariable ausführen

Das Deadline Cloud-Beispiel-Repository enthält ein Job-Bundle, das das Setzen und Anzeigen von Umgebungsvariablen demonstriert. Die Beispieljobvorlage definiert Variablen sowohl auf Auftrags- als auch auf Schrittebene und führt dann eine Aufgabe aus, die das zusammengeführte Ergebnis druckt. Gehen Sie wie folgt vor, um das Beispiel auszuführen und die Ergebnisse zu überprüfen.

Voraussetzungen

1. Wenn Sie keine Deadline Cloud-Farm mit einer Warteschlange und der zugehörigen Linux-Flotte haben, folgen Sie der Anleitung zum Onboarding in der Deadline Cloud-Konsole, um eine Farm mit Standardeinstellungen zu erstellen.
2. Wenn Sie die Deadline Cloud-CLI und den AWS Deadline Cloud-Monitor nicht auf Ihrer Workstation haben, folgen Sie den Schritten unter Deadline Cloud-Einreicher einrichten.
3. Wird verwendet`git`, um das Deadline GitHub Cloud-Beispiel-Repository zu klonen.

   ```
   git clone https://github.com/aws-deadline/deadline-cloud-samples.git
   cd deadline-cloud-samples/job_bundles
   ```

Ausführen des Beispiels

1. Verwenden Sie die Deadline Cloud-CLI, um das `job_env_vars` Beispiel einzureichen.

```
deadline bundle submit job_env_vars
```

2. Wählen Sie im Deadline Cloud-Monitor den neuen Job aus, um seinen Fortschritt zu überwachen. Sobald für die Linux Flotte, die der Warteschlange zugeordnet ist, ein Mitarbeiter verfügbar ist, ist der Job in wenigen Sekunden abgeschlossen. Wählen Sie die Aufgabe aus und wählen Sie dann im Menü oben rechts im Aufgabenbereich die Option Protokolle anzeigen aus.

Vergleich von Sitzungsaktionen mit ihren Definitionen

In der Protokollansicht werden drei Sitzungsaktionen angezeigt. Öffnen Sie die Datei job_env_vars/template.yaml in einem Texteditor, um jede Aktion mit ihrer Definition in der Jobvorlage zu vergleichen.

1. JobEnvWählen Sie die Aktion Sitzung starten aus. Die Protokollausgabe zeigt, dass die Umgebungsvariablen auf Jobebene gesetzt wurden.

```
Setting: JOB_VERBOSITY=MEDIUM
Setting: JOB_EXAMPLE_PARAM=An example parameter value
Setting: JOB_PROJECT_ID=project-12
Setting: JOB_ENDPOINT_URL=https://internal-host-name/some/path
Setting: QT_QPA_PLATFORM=offscreen
```

   Die folgenden Zeilen aus der Jobvorlage definieren diese Umgebung.

```
jobEnvironments:
- name: JobEnv
  variables:
    JOB_VERBOSITY: MEDIUM
    JOB_EXAMPLE_PARAM: "{{Param.ExampleParam}}"
    JOB_PROJECT_ID: project-12
    JOB_ENDPOINT_URL: https://internal-host-name/some/path
    QT_QPA_PLATFORM: offscreen
```

2. Wählen Sie die Aktion StepEnv Sitzung starten aus. Die Protokollausgabe zeigt die Variablen auf Schrittebene, einschließlich der überschriebenen Variablen. `JOB_PROJECT_ID`

```
Setting: STEP_VERBOSITY=HIGH
Setting: JOB_PROJECT_ID=step-project-12
```

Die folgenden Zeilen aus der Jobvorlage definieren diese Umgebung.

```
stepEnvironments:
- name: StepEnv
  variables:
    STEP_VERBOSITY: HIGH
    JOB_PROJECT_ID: step-project-12
```

3. Wählen Sie die Aktion Sitzung ausführen aus. In der Protokollausgabe werden die zusammengeführten Umgebungsvariablen angezeigt, die für die Aufgabe verfügbar sind. Beachten Sie, dass der Wert auf Schrittebene `JOB_PROJECT_ID` verwendet wird. `step-project-12`

```
Environment variables starting with JOB_*:
JOB_ENDPOINT_URL=https://internal-host-name/some/path
JOB_EXAMPLE_PARAM='An example parameter value'
JOB_PROJECT_ID=step-project-12
JOB_VERBOSITY=MEDIUM

Environment variables starting with STEP_*:
STEP_VERBOSITY=HIGH
```

Legen Sie den Pfad in einer Warteschlangenumgebung fest

Verwenden Sie OpenJD-Umgebungen, um neue Befehle in einer Umgebung bereitzustellen. Zuerst erstellen Sie ein Verzeichnis mit Skriptdateien und fügen dieses Verzeichnis dann zu den `PATH` Umgebungsvariablen hinzu, sodass die ausführbaren Dateien in Ihrem Skript sie ausführen können, ohne jedes Mal den Verzeichnispfad angeben zu müssen. Die Liste der Variablen in einer Umgebungsdefinition bietet keine Möglichkeit, die Variable zu ändern. Sie tun dies also, indem Sie stattdessen ein Skript ausführen. Nachdem das Skript die Dinge eingerichtet und geändert hat`PATH`, exportiert es die Variable mit dem Befehl in die OpenJD-Laufzeit. `echo "openjd_env: PATH=$PATH"`

Voraussetzungen

Führen Sie die folgenden Schritte aus, um das Beispiel-Job-Bundle mit Umgebungsvariablen aus dem Github-Repository für Beispiele von Deadline Cloud auszuführen.

1. Wenn Sie keine Deadline Cloud-Farm mit einer Warteschlange und der zugehörigen Linux-Flotte haben, folgen Sie der Anleitung zum Onboarding in der Deadline Cloud-Konsole, um eine Farm mit Standardeinstellungen zu erstellen.
2. Wenn Sie die Deadline Cloud-CLI und den Deadline Cloud-Monitor nicht auf Ihrer Workstation haben, folgen Sie den Schritten unter Deadline Cloud-Einreicher einrichten im Benutzerhandbuch.
3. Wird verwendet`git`, um das Deadline GitHub Cloud-Beispiel-Repository zu klonen.

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
 Cloning into 'deadline-cloud-samples'...
 ...
cd deadline-cloud-samples/job_bundles
```

Führen Sie das Pfadbeispiel aus

1. Verwenden Sie die Deadline Cloud-CLI, um das `job_env_with_new_command` Beispiel einzureichen.

```
 $ deadline bundle submit job_env_with_new_command
 Submitting to Queue: MySampleQueue
 ...
```

2. Im Deadline Cloud-Monitor sehen Sie den neuen Job und können seinen Fortschritt überwachen. Sobald für die Linux Flotte, die der Warteschlange zugeordnet ist, ein Mitarbeiter zur Verfügung steht, der die Aufgabe des Jobs ausführen kann, ist der Job in wenigen Sekunden abgeschlossen. Wählen Sie die Aufgabe aus und wählen Sie dann im Menü oben rechts im Aufgabenbereich die Option Protokolle anzeigen.

   Auf der rechten Seite befinden sich zwei Sitzungsaktionen: Starten RandomSleepCommand und Ausführen von Aufgaben. Die Protokollanzeige in der Mitte des Fensters entspricht der ausgewählten Sitzungsaktion auf der rechten Seite.

Vergleichen Sie die Sitzungsaktionen mit ihren Definitionen

In diesem Abschnitt verwenden Sie den Deadline Cloud-Monitor, um die Sitzungsaktionen mit der Position zu vergleichen, in der sie in der Jobvorlage definiert sind. Es geht weiter mit dem vorherigen Abschnitt.

Öffnen Sie die Datei job_env_with_new_command/template.yaml in einem Texteditor. Vergleichen Sie die Sitzungsaktionen mit den Aktionen, in denen sie in der Jobvorlage definiert sind.

1. Wählen Sie im Deadline Cloud-Monitor die Aktion RandomSleepCommand Sitzung starten aus. Sie werden die Protokollausgabe wie folgt sehen.

```
 2024/07/16 17:25:32-07:00
 2024/07/16 17:25:32-07:00 ==============================================
 2024/07/16 17:25:32-07:00 --------- Entering Environment: RandomSleepCommand
 2024/07/16 17:25:32-07:00 ==============================================
 2024/07/16 17:25:32-07:00 ----------------------------------------------
 2024/07/16 17:25:32-07:00 Phase: Setup
 2024/07/16 17:25:32-07:00 ----------------------------------------------
 2024/07/16 17:25:32-07:00 Writing embedded files for Environment to disk.
 2024/07/16 17:25:32-07:00 Mapping: Env.File.Enter -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmpbt8j_c3f
 2024/07/16 17:25:32-07:00 Mapping: Env.File.SleepScript -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmperastlp4
 2024/07/16 17:25:32-07:00 Wrote: Enter -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmpbt8j_c3f
 2024/07/16 17:25:32-07:00 Wrote: SleepScript -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmperastlp4
 2024/07/16 17:25:32-07:00 ----------------------------------------------
 2024/07/16 17:25:32-07:00 Phase: Running action
 2024/07/16 17:25:32-07:00 ----------------------------------------------
 2024/07/16 17:25:32-07:00 Running command sudo -u job-user -i setsid -w /sessions/
session-ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/tmpbwrquq5u.sh
 2024/07/16 17:25:32-07:00 Command started as pid: 2205
 2024/07/16 17:25:32-07:00 Output:
 2024/07/16 17:25:33-07:00 openjd_env: PATH=/sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/bin:/opt/conda/condabin:/home/job-
user/.local/bin:/home/job-user/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
bin:/sbin:/bin:/var/lib/snapd/snap/bin
 No newer logs at this moment.
```

In den folgenden Zeilen aus der Jobvorlage wurde diese Aktion angegeben.

```
jobEnvironments:
- name: RandomSleepCommand
  description: Adds a command 'random-sleep' to the environment.
  script:
    actions:
```

```
      onEnter:
        command: bash
        args:
        - "{{Env.File.Enter}}"
    embeddedFiles:
    - name: Enter
      type: TEXT
      data: |
        #!/bin/env bash
        set -euo pipefail

        # Make a bin directory inside the session's working directory for providing
 new commands
        mkdir -p '{{Session.WorkingDirectory}}/bin'

        # If this bin directory is not already in the PATH, then add it
        if ! [[ ":$PATH:" == *':{{Session.WorkingDirectory}}/bin:'* ]]; then
          export "PATH={{Session.WorkingDirectory}}/bin:$PATH"

          # This message to Open Job Description exports the new PATH value to the
 environment
          echo "openjd_env: PATH=$PATH"
        fi

        # Copy the SleepScript embedded file into the bin directory
        cp '{{Env.File.SleepScript}}' '{{Session.WorkingDirectory}}/bin/random-
sleep'
        chmod u+x '{{Session.WorkingDirectory}}/bin/random-sleep'
    - name: SleepScript
      type: TEXT
      runnable: true
      data: |
        ...
```

2. Wählen Sie im Deadline Cloud-Monitor die Aktion StepEnv Sitzung starten aus. Sie sehen die Protokollausgabe wie folgt.

```
2024/07/16 17:25:33-07:00
2024/07/16 17:25:33-07:00 ==============================================
2024/07/16 17:25:33-07:00 --------- Running Task
2024/07/16 17:25:33-07:00 ==============================================
2024/07/16 17:25:33-07:00 ----------------------------------------------
2024/07/16 17:25:33-07:00 Phase: Setup
```

```
 2024/07/16 17:25:33-07:00 ------------------------------------------------
 2024/07/16 17:25:33-07:00 Writing embedded files for Task to disk.
 2024/07/16 17:25:33-07:00 Mapping: Task.File.Run -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmpdrwuehjf
 2024/07/16 17:25:33-07:00 Wrote: Run -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmpdrwuehjf
 2024/07/16 17:25:33-07:00 ------------------------------------------------
 2024/07/16 17:25:33-07:00 Phase: Running action
 2024/07/16 17:25:33-07:00 ------------------------------------------------
 2024/07/16 17:25:33-07:00 Running command sudo -u job-user -i setsid -w /sessions/
session-ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/tmpz81iaqfw.sh
 2024/07/16 17:25:33-07:00 Command started as pid: 2256
 2024/07/16 17:25:33-07:00 Output:
 2024/07/16 17:25:34-07:00 + random-sleep 12.5 27.5
 2024/07/16 17:26:00-07:00 Sleeping for duration 26.90
 2024/07/16 17:26:00-07:00 ------------------------------------------------
 2024/07/16 17:26:00-07:00 Uploading output files to Job Attachments
 2024/07/16 17:26:00-07:00 ------------------------------------------------
```

3. In den folgenden Zeilen aus der Jobvorlage wurde diese Aktion angegeben.

```
steps:
- name: EnvWithCommand
  script:
    actions:
      onRun:
        command: bash
        args:
        - '{{Task.File.Run}}'
    embeddedFiles:
    - name: Run
      type: TEXT
      data: |
        set -xeuo pipefail

        # Run the script installed into PATH by the job environment
        random-sleep 12.5 27.5
  hostRequirements:
    attributes:
    - name: attr.worker.os.family
      anyOf:
      - linux
```

Führen Sie einen Hintergrund-Daemon-Prozess in der Warteschlangenumgebung aus

In vielen Anwendungsfällen beim Rendern kann das Laden der Anwendungs- und Szenendaten viel Zeit in Anspruch nehmen. Wenn ein Job sie für jeden Frame neu lädt, verbringt er die meiste Zeit mit Overhead. Es ist oft möglich, die Anwendung einmal als Hintergrund-Daemon-Prozess zu laden, sie die Szenendaten laden zu lassen und ihr dann Befehle über Interprozesskommunikation (IPC) zu senden, um die Renderings durchzuführen.

Viele der Open-Source-Integrationen von Deadline Cloud verwenden dieses Muster. Das Projekt Open Job Description stellt eine Adapter-Laufzeitbibliothek mit robusten IPC-Mustern auf allen unterstützten Betriebssystemen bereit.

Um dieses Muster zu demonstrieren, gibt es ein eigenständiges Beispiel-Job-Paket, das Python- und Bash-Code verwendet, um einen Hintergrund-Daemon und den IPC für Aufgaben zur Kommunikation mit ihm zu implementieren. Der Daemon ist in Python implementiert und wartet auf ein SIGUSR1 POSIX-Signal, das angibt, wann eine Aufgabe bearbeitet werden muss. Die Aufgabendetails werden in einer bestimmten JSON-Datei an den Daemon übergeben, und die Ergebnisse der Ausführung der Aufgabe werden als weitere JSON-Datei zurückgegeben.

Voraussetzungen

Führen Sie die folgenden Schritte aus, um das Beispiel-Auftragspaket mit einem Daemon-Prozess aus dem Github-Repository für Beispiele von Deadline Cloud auszuführen.

1. Wenn Sie keine Deadline Cloud-Farm mit einer Warteschlange und der zugehörigen Linux-Flotte haben, folgen Sie der Anleitung zum Onboarding in der Deadline Cloud-Konsole, um eine Farm mit Standardeinstellungen zu erstellen.
2. Wenn Sie die Deadline Cloud-CLI und den Deadline Cloud-Monitor nicht auf Ihrer Workstation haben, folgen Sie den Schritten unter Deadline Cloud-Einreicher einrichten im Benutzerhandbuch.
3. Wird verwendet`git`, um das Deadline GitHub Cloud-Beispiel-Repository zu klonen.

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
 Cloning into 'deadline-cloud-samples'...
 ...
cd deadline-cloud-samples/job_bundles
```

Führen Sie das Daemon-Beispiel aus

1. Verwenden Sie die Deadline Cloud-CLI, um das `job_env_daemon_process` Beispiel einzureichen.

```
 git clone https://github.com/aws-deadline/deadline-cloud-samples.git
Cloning into 'deadline-cloud-samples'...
 ...
cd deadline-cloud-samples/job_bundles
```

2. In der Deadline Cloud-Monitoranwendung sehen Sie den neuen Job und können seinen Fortschritt überwachen. Sobald für die Linux Flotte, die der Warteschlange zugeordnet ist, ein Mitarbeiter zur Verfügung steht, der die Aufgabe des Auftrags ausführt, ist der Job in etwa einer Minute abgeschlossen. Wenn Sie eine der Aufgaben ausgewählt haben, wählen Sie im Menü oben rechts im Aufgabenbereich die Option Protokolle anzeigen.

   Auf der rechten Seite befinden sich zwei Sitzungsaktionen: Starten DaemonProcess und Ausführen von Aufgaben. Die Protokollanzeige in der Mitte des Fensters entspricht der ausgewählten Sitzungsaktion auf der rechten Seite.

   Wählen Sie die Option Protokolle für alle Aufgaben anzeigen. In der Zeitleiste werden die restlichen Aufgaben angezeigt, die im Rahmen der Sitzung ausgeführt wurden, sowie die `Shut down DaemonProcess` Aktion, die die Umgebung verlassen hat.

Sehen Sie sich die Daemon-Protokolle an

1. In diesem Abschnitt verwenden Sie den Deadline Cloud-Monitor, um die Sitzungsaktionen mit denen zu vergleichen, in denen sie in der Jobvorlage definiert sind. Es geht weiter mit dem vorherigen Abschnitt.

   Öffnen Sie die Datei job_env_daemon_process/template.yaml in einem Texteditor. Vergleichen Sie die Sitzungsaktionen mit den Aktionen, in denen sie in der Jobvorlage definiert sind.

2. Wählen Sie die `Launch DaemonProcess` Sitzungsaktion im Deadline Cloud-Monitor aus. Sie werden die Protokollausgabe wie folgt sehen.

```
2024/07/17 16:27:20-07:00
2024/07/17 16:27:20-07:00 ==============================================
2024/07/17 16:27:20-07:00 --------- Entering Environment: DaemonProcess
2024/07/17 16:27:20-07:00 ==============================================
```

```
 2024/07/17 16:27:20-07:00 ----------------------------------------------
 2024/07/17 16:27:20-07:00 Phase: Setup
 2024/07/17 16:27:20-07:00 ----------------------------------------------
 2024/07/17 16:27:20-07:00 Writing embedded files for Environment to disk.
 2024/07/17 16:27:20-07:00 Mapping: Env.File.Enter -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/enter-daemon-
process-env.sh
 2024/07/17 16:27:20-07:00 Mapping: Env.File.Exit -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/exit-daemon-
process-env.sh
 2024/07/17 16:27:20-07:00 Mapping: Env.File.DaemonScript -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
script.py
 2024/07/17 16:27:20-07:00 Mapping: Env.File.DaemonHelperFunctions -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
helper-functions.sh
 2024/07/17 16:27:20-07:00 Wrote: Enter -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/enter-daemon-
process-env.sh
 2024/07/17 16:27:20-07:00 Wrote: Exit -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/exit-daemon-
process-env.sh
 2024/07/17 16:27:20-07:00 Wrote: DaemonScript -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
script.py
 2024/07/17 16:27:20-07:00 Wrote: DaemonHelperFunctions -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
helper-functions.sh
 2024/07/17 16:27:20-07:00 ----------------------------------------------
 2024/07/17 16:27:20-07:00 Phase: Running action
 2024/07/17 16:27:20-07:00 ----------------------------------------------
 2024/07/17 16:27:20-07:00 Running command sudo -u job-user -i setsid -w /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/tmp_u8slys3.sh
 2024/07/17 16:27:20-07:00 Command started as pid: 2187
 2024/07/17 16:27:20-07:00 Output:
 2024/07/17 16:27:21-07:00 openjd_env: DAEMON_LOG=/sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/daemon.log
 2024/07/17 16:27:21-07:00 openjd_env: DAEMON_PID=2223
 2024/07/17 16:27:21-07:00 openjd_env: DAEMON_BASH_HELPER_SCRIPT=/sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
helper-functions.sh
```

In den folgenden Zeilen aus der Jobvorlage wurde diese Aktion angegeben.

```
  stepEnvironments:
  - name: DaemonProcess
    description: Runs a daemon process for the step's tasks to share.
    script:
      actions:
        onEnter:
          command: bash
          args:
          - "{{Env.File.Enter}}"
        onExit:
          command: bash
          args:
          - "{{Env.File.Exit}}"
      embeddedFiles:
      - name: Enter
        filename: enter-daemon-process-env.sh
        type: TEXT
        data: |
          #!/bin/env bash
          set -euo pipefail

          DAEMON_LOG='{{Session.WorkingDirectory}}/daemon.log'
          echo "openjd_env: DAEMON_LOG=$DAEMON_LOG"
          nohup python {{Env.File.DaemonScript}} > $DAEMON_LOG 2>&1 &
          echo "openjd_env: DAEMON_PID=$!"
          echo "openjd_env:
DAEMON_BASH_HELPER_SCRIPT={{Env.File.DaemonHelperFunctions}}"

          echo 0 > 'daemon_log_cursor.txt'
    ...
```

3. Wählen Sie im Deadline Cloud-Monitor eine der Aktionen Task run: N session aus. Sie werden die Protokollausgabe wie folgt sehen.

```
2024/07/17 16:27:22-07:00
 2024/07/17 16:27:22-07:00 ================================================
 2024/07/17 16:27:22-07:00 --------- Running Task
 2024/07/17 16:27:22-07:00 ================================================
 2024/07/17 16:27:22-07:00 Parameter values:
 2024/07/17 16:27:22-07:00 Frame(INT) = 2
 2024/07/17 16:27:22-07:00 ------------------------------------------------
 2024/07/17 16:27:22-07:00 Phase: Setup
```

```
 2024/07/17 16:27:22-07:00 ------------------------------------------------
 2024/07/17 16:27:22-07:00 Writing embedded files for Task to disk.
 2024/07/17 16:27:22-07:00 Mapping: Task.File.Run -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/run-task.sh
 2024/07/17 16:27:22-07:00 Wrote: Run -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/run-task.sh
 2024/07/17 16:27:22-07:00 ------------------------------------------------
 2024/07/17 16:27:22-07:00 Phase: Running action
 2024/07/17 16:27:22-07:00 ------------------------------------------------
 2024/07/17 16:27:22-07:00 Running command sudo -u job-user -i setsid -w /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/tmpv4obfkhn.sh
 2024/07/17 16:27:22-07:00 Command started as pid: 2301
 2024/07/17 16:27:22-07:00 Output:
 2024/07/17 16:27:23-07:00 Daemon PID is 2223
 2024/07/17 16:27:23-07:00 Daemon log file is /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/daemon.log
 2024/07/17 16:27:23-07:00
 2024/07/17 16:27:23-07:00 === Previous output from daemon
 2024/07/17 16:27:23-07:00 ===
 2024/07/17 16:27:23-07:00
 2024/07/17 16:27:23-07:00 Sending command to daemon
 2024/07/17 16:27:23-07:00 Received task result:
 2024/07/17 16:27:23-07:00 {
 2024/07/17 16:27:23-07:00   "result": "SUCCESS",
 2024/07/17 16:27:23-07:00   "processedTaskCount": 1,
 2024/07/17 16:27:23-07:00   "randomValue": 0.2578537967668988,
 2024/07/17 16:27:23-07:00   "failureRate": 0.1
 2024/07/17 16:27:23-07:00 }
 2024/07/17 16:27:23-07:00
 2024/07/17 16:27:23-07:00 === Daemon log from running the task
 2024/07/17 16:27:23-07:00 Loading the task details file
 2024/07/17 16:27:23-07:00 Received task details:
 2024/07/17 16:27:23-07:00 {
 2024/07/17 16:27:23-07:00  "pid": 2329,
 2024/07/17 16:27:23-07:00  "frame": 2
 2024/07/17 16:27:23-07:00 }
 2024/07/17 16:27:23-07:00 Processing frame number 2
 2024/07/17 16:27:23-07:00 Writing result
 2024/07/17 16:27:23-07:00 Waiting until a USR1 signal is sent...
 2024/07/17 16:27:23-07:00 ===
 2024/07/17 16:27:23-07:00
 2024/07/17 16:27:23-07:00 ------------------------------------------------
 2024/07/17 16:27:23-07:00 Uploading output files to Job Attachments
```

```
2024/07/17 16:27:23-07:00 ------------------------------------------------
```

Die folgenden Zeilen aus der Job-Vorlage spezifizieren diese Aktion. ```Schritte:

```
steps:
- name: EnvWithDaemonProcess
  parameterSpace:
    taskParameterDefinitions:
    - name: Frame
      type: INT
      range: "{{Param.Frames}}"

  stepEnvironments:
    ...

  script:
    actions:
      onRun:
        timeout: 60
        command: bash
        args:
        - '{{Task.File.Run}}'
    embeddedFiles:
    - name: Run
      filename: run-task.sh
      type: TEXT
      data: |
        # This bash script sends a task to the background daemon process,
        # then waits for it to respond with the output result.

        set -euo pipefail

        source "$DAEMON_BASH_HELPER_SCRIPT"

        echo "Daemon PID is $DAEMON_PID"
        echo "Daemon log file is $DAEMON_LOG"

        print_daemon_log "Previous output from daemon"

        send_task_to_daemon "{\"pid\": $$, \"frame\": {{Task.Param.Frame}} }"
        wait_for_daemon_task_result

        echo Received task result:
```

```
        echo "$TASK_RESULT" | jq .

        print_daemon_log "Daemon log from running the task"

  hostRequirements:
    attributes:
    - name: attr.worker.os.family
      anyOf:
      - linux
```

Bewerben Sie sich für Ihre Jobs

Sie können eine Warteschlangenumgebung verwenden, um Anwendungen zur Verarbeitung Ihrer Jobs zu laden. Wenn Sie mithilfe der Deadline Cloud-Konsole eine vom Service verwaltete Flotte erstellen, haben Sie die Möglichkeit, eine Warteschlangenumgebung zu erstellen, die den Conda-Paketmanager zum Laden von Anwendungen verwendet.

Wenn Sie einen anderen Paketmanager verwenden möchten, können Sie eine Warteschlangenumgebung für diesen Manager erstellen. Ein Beispiel für die Verwendung von Rez finden Sie unterVerwenden Sie einen anderen Paketmanager.

Deadline Cloud bietet einen Conda-Kanal, über den Sie eine Auswahl von Rendering-Anwendungen in Ihre Umgebung laden können. Sie unterstützen die Einreicher, die Deadline Cloud für Anwendungen zur Erstellung digitaler Inhalte bereitstellt.

Sie können auch Software für Conda-Forge laden, um sie in Ihren Jobs zu verwenden. Die folgenden Beispiele zeigen Jobvorlagen, die die von Deadline Cloud bereitgestellte Warteschlangenumgebung verwenden, um Anwendungen vor der Ausführung des Jobs zu laden.

Themen

- Eine Anwendung von einem Conda-Kanal abrufen
- Verwenden Sie einen anderen Paketmanager

Eine Anwendung von einem Conda-Kanal abrufen

Sie können eine benutzerdefinierte Warteschlangenumgebung für Ihre Deadline Cloud-Mitarbeiter erstellen, die die Software Ihrer Wahl installiert. Diese Beispiel-Warteschlangenumgebung hat dasselbe Verhalten wie die Umgebung, die von der Konsole für vom Service verwaltete Flotten verwendet wird. Sie führt Conda direkt aus, um die Umgebung zu erstellen.

Die Umgebung erstellt für jede Deadline Cloud-Sitzung, die auf einem Worker ausgeführt wird, eine neue virtuelle Conda-Umgebung und löscht die Umgebung, wenn sie fertig ist.

Conda speichert die heruntergeladenen Pakete im Cache, sodass sie nicht erneut heruntergeladen werden müssen. In jeder Sitzung müssen jedoch alle Pakete mit der Umgebung verknüpft werden.

Die Umgebung definiert drei Skripte, die ausgeführt werden, wenn Deadline Cloud eine Sitzung auf einem Worker startet. Das erste Skript wird ausgeführt, wenn die `onEnter` Aktion aufgerufen wird. Es ruft die anderen beiden auf, um Umgebungsvariablen einzurichten. Wenn die Ausführung des Skripts abgeschlossen ist, ist die Conda-Umgebung verfügbar, in der alle angegebenen Umgebungsvariablen gesetzt sind.

Die neueste Version des Beispiels finden Sie unter conda_queue_env_console_equivalent.yaml im Repository unter. deadline-cloud-samples GitHub

Wenn Sie eine Anwendung verwenden möchten, die im Conda-Channel nicht verfügbar ist, können Sie einen Conda-Channel in Amazon S3 erstellen und dann Ihre eigenen Pakete für diese Anwendung erstellen. Weitere Informationen hierzu finden Sie unter Erstellen Sie einen Conda-Kanal mit S3.

Holen Sie sich Open-Source-Bibliotheken von Conda-Forge

In diesem Abschnitt wird beschrieben, wie Sie Open-Source-Bibliotheken aus dem `conda-forge` Kanal verwenden. Das folgende Beispiel ist eine Jobvorlage, die das `polars` Python-Paket verwendet.

Der Job legt die in der Warteschlangenumgebung definierten `CondaChannels` Parameter `CondaPackages` und fest, die Deadline Cloud mitteilen, wo das Paket abgerufen werden soll.

Der Abschnitt der Jobvorlage, der die Parameter festlegt, lautet:

```
- name: CondaPackages
  description: A list of conda packages to install. The job expects a Queue Environment
 to handle this.
  type: STRING
  default: polars
- name: CondaChannels
  description: A list of conda channels to get packages from. The job expects a Queue
 Environment to handle this.
  type: STRING
  default: conda-forge
```

Die neueste Version der vollständigen Beispiel-Jobvorlage finden Sie unter stage_1_self_contained_template/template.yaml. Die neueste Version der Warteschlangenumgebung, die die Conda-Pakete lädt, finden Sie unter conda_queue_env_console_equivalent.yaml im Repository unter. deadline-cloud-samples GitHub

Rufen Sie den Deadline-Cloud-Kanal auf Blender

Das folgende Beispiel zeigt eine Jobvorlage, die Blender aus dem `deadline-cloud` Conda-Channel stammt. Dieser Kanal unterstützt die Einreicher, die Deadline Cloud für Software zur Erstellung digitaler Inhalte bereitstellt. Sie können jedoch denselben Kanal verwenden, um Software für Ihren eigenen Gebrauch zu laden.

Eine Liste der vom `deadline-cloud` Kanal bereitgestellten Software finden Sie unter Standard-Warteschlangenumgebung im AWS Deadline Cloud-Benutzerhandbuch.

Dieser Job legt den in der Warteschlangenumgebung definierten `CondaPackages` Parameter fest, um Deadline Cloud anzuweisen, Blender in die Umgebung zu laden.

Der Abschnitt der Jobvorlage, der den Parameter festlegt, lautet:

```
- name: CondaPackages
  type: STRING
  userInterface:
    control: LINE_EDIT
    label: Conda Packages
    groupLabel: Software Environment
  default: blender
  description: >
    Tells the queue environment to install Blender from the deadline-cloud conda
 channel.
```

Die neueste Version der vollständigen Beispiel-Jobvorlage finden Sie unter blender_render/template.yaml. Die neueste Version der Warteschlangenumgebung, die die Conda-Pakete lädt, finden Sie unter conda_queue_env_console_equivalent.yaml im Repository unter. deadline-cloud-samplesGitHub

Verwenden Sie einen anderen Paketmanager

Der Standard-Paketmanager für Deadline Cloud ist conda. Wenn Sie einen anderen Paketmanager verwenden müssen, z. B. können Sie eine benutzerdefinierte Warteschlangenumgebung erstellenRez, die Skripts enthält, die stattdessen Ihren Paketmanager verwenden.

Diese Beispiel-Warteschlangenumgebung bietet dasselbe Verhalten wie die Umgebung, die von der Konsole für vom Service verwaltete Flotten verwendet wird. Sie ersetzt den Conda-Paketmanager durch. Rez

Die Umgebung definiert drei Skripte, die ausgeführt werden, wenn Deadline Cloud eine Sitzung auf einem Worker startet. Das erste Skript wird ausgeführt, wenn die `onEnter` Aktion aufgerufen wird. Es ruft die anderen beiden auf, um Umgebungsvariablen einzurichten. Wenn die Ausführung des Skripts abgeschlossen ist, ist die Rez Umgebung mit allen festgelegten Umgebungsvariablen verfügbar.

Das Beispiel geht davon aus, dass Sie über eine vom Kunden verwaltete Flotte verfügen, die ein gemeinsam genutztes Dateisystem für die Rez-Pakete verwendet.

Die neueste Version des Beispiels finden Sie unter rez_queue_env.yaml im Repository unter. deadline-cloud-samplesGitHub

Erstellen Sie einen Conda-Kanal mit S3

Wenn Ihre Jobs Anwendungen ausführen müssen, die auf den `conda-forge`Kanälen `deadline-cloud`oder nicht verfügbar sind, können Sie einen benutzerdefinierten Conda-Channel hosten, um Ihre eigenen Pakete bereitzustellen. Wenn Sie eine Warteschlange in der AWS Deadline Cloud (Deadline Cloud) -Konsole erstellen, fügt die Konsole standardmäßig eine Conda-Warteschlangenumgebung hinzu. Um Ihre Pakete für Jobs verfügbar zu machen, fügen Sie den benutzerdefinierten Channel zur Warteschlangenumgebung hinzu.

Ein Conda-Channel ist statischer gehosteter Inhalt, den Sie auf verschiedene Arten hosten können, z. B. in einem Dateisystem oder in einem Amazon Simple Storage Service (Amazon S3) -Bucket. Wenn Ihre Deadline Cloud-Farm ein gemeinsam genutztes Dateisystem für Assets verwendet, können Sie einen beliebigen Pfad darauf als Kanalnamen verwenden. Sie können den Channel in einem Amazon S3 S3-Bucket hosten, um mithilfe von AWS Identity and Access Management (IAM-) Berechtigungen einen breiteren Zugriff zu erhalten.

Sie können Pakete lokal erstellen und testen und sie dann in einem Channel veröffentlichen. Das lokale Erstellen von Paketen ist eine einfache Möglichkeit, ohne die Einrichtung einer Infrastruktur mit der Iteration von Paketen zu beginnen. Sie können auch eine Warteschlange zum Erstellen von Paketen in der Deadline Cloud verwenden, um Pakete zu erstellen und sie in einem Channel zu veröffentlichen. Eine Warteschlange zur Paketerstellung vereinfacht die Verwaltung von Paketen für mehrere Betriebssysteme und Beschleunigerkonfigurationen. Sie können Versionen aktualisieren und vollständige Sätze von Paket-Builds von überall aus einreichen.

Sie können Kanäle für Ihr Studio und Ihre Deadline Cloud-Farm auf verschiedene Arten konfigurieren. Sie können einen Amazon S3 S3-Kanal haben und all Ihre Workstations und Farmhosts so konfigurieren, dass sie ihn verwenden. Sie können auch mehr als einen Kanal haben und die Spiegelung mit AWS DataSync () DataSync einrichten. Ihre Deadline Cloud-Warteschlange zum Erstellen von Paketen kann beispielsweise auf einem Amazon S3 S3-Channel veröffentlicht werden, der lokal für Workstations und lokale Farmhosts gespiegelt wird.

Themen

Pakete lokal erstellen und testen

Bevor Sie Pakete auf Amazon S3 veröffentlichen oder die CI/CD Automatisierung auf Ihrer Deadline Cloud-Farm einrichten, können Sie Conda-Pakete auf Ihrer Workstation mithilfe eines lokalen Dateisystemkanals erstellen und testen. Mit diesem Ansatz können Sie schnell lokal an Rezepten iterieren und Pakete verifizieren.

Der `rattler-build publish` Befehl erstellt ein Rezept, kopiert das resultierende Paket in einen Channel und indexiert den Channel in einem Schritt. Wenn Sie auf ein lokales Dateisystemverzeichnis abzielen, `rattler-build` wird der Kanal automatisch erstellt und initialisiert, falls das Verzeichnis nicht existiert.

Die folgenden Anweisungen verwenden das Blender 4.5-Beispielrezept aus dem Deadline Cloud-Beispiel-Repository auf. GitHub Sie können ein anderes Rezept aus dem Proben-Repository verwenden oder Ihr eigenes Rezept verwenden.

Voraussetzungen

Bevor Sie beginnen, installieren Sie die folgenden Tools auf Ihrer Workstation:

- pixi — Ein Paketmanager, den Sie zum Installieren `rattler-build` und Testen von Paketen verwenden. Installiere Pixi aus pixi.sh.
- rattler-build — Das Tool zur Paketerstellung, das von Deadline Cloud Conda Recipes verwendet wird. Führen Sie nach der Installation von Pixi den folgenden Befehl aus, um Pixi zu installieren. `rattler-build`

  ```
  pixi global install rattler-build
  ```
- git — Erforderlich, um das Beispiel-Repository zu klonen. OnWindows, git for bietet Windows auch eine `bash` Shell, die einige der Windows Beispielrezepte benötigen.

Ein Paket erstellen und auf einem lokalen Channel veröffentlichen

In diesem Verfahren klonen Sie das Deadline Cloud-Beispiel-Repository und verwenden es`rattler-build publish`, um das Paket zu erstellen und in einem lokalen Dateisystemkanal zu veröffentlichen.

Um ein Paket zu erstellen und in einem lokalen Channel zu veröffentlichen

1. Klonen Sie das Deadline Cloud-Beispiel-Repository.

   ```
   git clone https://github.com/aws-deadline/deadline-cloud-samples.git
   ```
2. Wechseln Sie in das `conda_recipes`-Verzeichnis.

   ```
   cd deadline-cloud-samples/conda_recipes
   ```
3. Führen Sie den folgenden Befehl aus, um das Blender 4.5-Rezept zu erstellen und das Paket in einem lokalen Channel-Verzeichnis zu veröffentlichen.

   Führen Sie macOS unter Linux und den folgenden Befehl aus.

   ```
   rattler-build publish blender-4.5/recipe/recipe.yaml \
       --to file://$HOME/my-conda-channel
   ```

   Führen Sie auf Windows (cmd) den folgenden Befehl aus.

```
rattler-build publish blender-4.5/recipe/recipe.yaml ^
    --to file://%USERPROFILE%/my-conda-channel
```

Der `rattler-build publish` Befehl führt die folgenden Aktionen aus:

- Erstellt das Paket aus dem Rezept.
- Erzeugt das Kanalverzeichnis, falls das Verzeichnis nicht existiert.
- Kopiert die Paketdatei in den Channel.
- Indiziert den Kanal, sodass Paketmanager das Paket finden können.

Wenn Ihr Paketrezept von Paketen aus einem bestimmten Kanal abhängt, wie z. B. conda-forge, fügen Sie dem Befehl etwas `-c conda-forge` hinzu.

Um das Paket neu aufzubauen, nachdem Sie Änderungen am Rezept vorgenommen haben, fügen Sie hinzu, `--build-number=+1` um die Build-Nummer automatisch zu erhöhen.

```
rattler-build publish blender-4.5/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1
```

Weitere Informationen dazu finden Sie in der `rattler-build publish` Dokumentation zur Veröffentlichung von Rattler-Build.

Builds debuggen

Wenn ein Build fehlschlägt, wird `rattler-build` das Build-Verzeichnis beibehalten, sodass Sie es untersuchen können. Führen Sie den folgenden Befehl aus, um eine interaktive Shell in der Build-Umgebung zu öffnen, in der alle Umgebungsvariablen so eingerichtet sind, wie sie während des Builds waren.

```
rattler-build debug shell
```

In der Debug-Shell können Sie Dateien ändern, einzelne Build-Befehle ausführen und Abhängigkeiten hinzufügen, um das Problem zu isolieren. Weitere Informationen finden Sie unter Debuggen von Builds in der Rattler-Build-Dokumentation.

Das Paket testen

Nachdem Sie das Paket erstellt und veröffentlicht haben, erstellen Sie ein temporäres Pixi-Projekt. Verwenden Sie das Projekt, um das Paket vom lokalen Kanal aus zu installieren und zu überprüfen, ob es ordnungsgemäß funktioniert.

Um das Paket zu testen

1. Erstellen Sie ein temporäres Testverzeichnis und initialisieren Sie ein Pixi-Projekt mit dem lokalen Kanal.

   Führen Sie unter Linux und macOS die folgenden Befehle aus.

   ```
   mkdir package-test-env
   cd package-test-env
   pixi init --channel file://$HOME/my-conda-channel
   ```

   Führen Sie auf Windows (cmd) die folgenden Befehle aus.

   ```
   mkdir package-test-env
   cd package-test-env
   pixi init --channel file://%USERPROFILE%/my-conda-channel
   ```

2. Fügen Sie das Paket dem Projekt hinzu.

   ```
   pixi add blender=4.5
   ```

3. Stellen Sie sicher, dass das Paket ordnungsgemäß funktioniert.

   ```
   pixi run blender --version
   ```

Wenn Sie mit dem Paket zufrieden sind, können Sie das Paket in einem Amazon S3 S3-Conda-Kanal veröffentlichen, sodass Deadline Cloud-Mitarbeiter das Paket installieren können. Weitere Informationen finden Sie unter Pakete in einem S3-Conda-Kanal veröffentlichen.

Bereinigen

Nach dem Testen können Sie das Testprojekt und den lokalen Channel entfernen.

Um Testressourcen zu bereinigen

1. Entfernen Sie das Testprojektverzeichnis.

   Führen Sie macOS unter Linux und den folgenden Befehl aus.

   ```
   rm -rf package-test-env
   ```

   Führen Sie auf Windows (cmd) den folgenden Befehl aus.

   ```
   rmdir /s /q package-test-env
   ```

2. Entfernen Sie das lokale Conda-Kanalverzeichnis.

   Führen Sie unter Linux macOS und den folgenden Befehl aus.

   ```
   rm -rf $HOME/my-conda-channel
   ```

   Führen Sie auf Windows (cmd) den folgenden Befehl aus.

   ```
   rmdir /s /q %USERPROFILE%\my-conda-channel
   ```

3. (Optional) Entfernen Sie das `rattler-build` Ausgabeverzeichnis, das die erstellte Paketdatei enthält.

   Führen Sie macOS unter Linux und den folgenden Befehl aus.

   ```
   rm -rf deadline-cloud-samples/conda_recipes/output
   ```

   Führen Sie auf Windows (cmd) den folgenden Befehl aus.

   ```
   rmdir /s /q deadline-cloud-samples\conda_recipes\output
   ```

Veröffentlichen Sie Pakete in einem Amazon S3 S3-Conda-Kanal

Sie können Conda-Pakete in einem Amazon Simple Storage Service (Amazon S3) -Bucket veröffentlichen, sodass AWS Deadline Cloud-Mitarbeiter (Deadline Cloud) sie für die Ausführung von Jobs installieren können. Der `rattler-build publish` Befehl funktioniert mit Amazon S3

genauso wie mit einem lokalen Dateisystemkanal. Der Befehl kann ein Rezept erstellen und das Ergebnis veröffentlichen oder eine Paketdatei veröffentlichen, die Sie bereits erstellt haben. In beiden Fällen lädt der Befehl das Paket in den Bucket hoch und indexiert den Channel in einem Schritt.

Der `rattler-build publish` Befehl authentifiziert sich AWS mithilfe der standardmäßigen Anmeldeinformationskette, sodass er Ihre AWS Konfiguration wie jedes andere Tool verwendet. AWS Weitere Informationen zur Konfiguration von Anmeldeinformationen finden Sie unter Konfiguration und Einstellungen für Anmeldeinformationsdateien im AWS Command Line Interface (AWS CLI) - Benutzerhandbuch.

Voraussetzungen

Bevor Sie Pakete in Amazon S3 veröffentlichen, müssen Sie die folgenden Voraussetzungen erfüllen:

- pixi and rattler-build — Installieren Sie Pixi aus pixi.sh und installieren Sie es anschließend. `rattler-build`

  ```
  pixi global install rattler-build
  ```

- git — Erforderlich, um das Beispiel-Repository zu klonen. OnWindows, git for bietet Windows auch eine `bash` Shell, die einige der Windows Beispielrezepte benötigen.
- Amazon S3 S3-Bucket — Ein Amazon S3 S3-Bucket, der als Conda-Kanal verwendet werden soll. Sie können den Bucket für Jobanhänge aus Ihrer Deadline Cloud-Farm verwenden oder einen separaten Bucket erstellen.
- AWS Anmeldeinformationen — Konfigurieren Sie die Anmeldeinformationen auf Ihrer Workstation mithilfe des `aws configure` Befehls oder des `aws login` Befehls. Weitere Informationen finden Sie unter Einrichten von AWS CLI im AWS Command Line Interface -Benutzerhandbuch.
- IAM-Berechtigungen — (Optional) Um den Umfang der Berechtigungen zu reduzieren, über die Ihre Anmeldeinformationen verfügen, können Sie eine AWS Identity and Access Management (IAM-) Richtlinie verwenden, die nur die folgenden Berechtigungen für den Amazon S3 S3-Bucket und das von Ihnen verwendete Kanalpräfix gewährt (z. B.`/Conda/*`):
  - `s3:GetObject`
  - `s3:PutObject`
  - `s3:DeleteObject`
  - `s3:ListBucket`
  - `s3:GetBucketLocation`

Ein Paket auf einem Amazon S3 S3-Kanal veröffentlichen

Verwenden Sie `rattler-build publish` mit einem `s3://` Ziel, um ein Paket in Ihrem Amazon S3 S3-Conda-Kanal zu veröffentlichen. Wenn der Kanal nicht im Bucket vorhanden ist, wird der Kanal `rattler-build` automatisch initialisiert. Bevor Sie beginnen, stellen Sie sicher, dass Sie die Voraussetzungen erfüllt haben.

Im folgenden Beispiel wird das Blender 4.5-Beispielrezept aus dem Deadline Cloud-Beispiel-Repository am veröffentlichtGitHub. Sie können ein anderes Rezept aus dem Samples-Repository verwenden oder Ihr eigenes Rezept verwenden.

Um ein Paket auf einem Amazon S3 S3-Kanal zu veröffentlichen

1. Klonen Sie das Deadline Cloud-Beispiel-Repository.

   ```
   git clone https://github.com/aws-deadline/deadline-cloud-samples.git
   ```

2. Wechseln Sie in das `conda_recipes`-Verzeichnis.

   ```
   cd deadline-cloud-samples/conda_recipes
   ```

3. Führen Sie den folgenden Befehl aus. *`amzn-s3-demo-bucket`*Ersetzen Sie es durch Ihren Bucket-Namen.

   Führen Sie macOS unter Linux und den folgenden Befehl aus.

   ```
   rattler-build publish blender-4.5/recipe/recipe.yaml \
       --to s3://amzn-s3-demo-bucket/Conda/Default
   ```

   Führen Sie auf Windows (cmd) den folgenden Befehl aus.

   ```
   rattler-build publish blender-4.5/recipe/recipe.yaml ^
       --to s3://amzn-s3-demo-bucket/Conda/Default
   ```

   Das `/Conda/Default` Präfix organisiert den Kanal innerhalb des Buckets. Sie können ein anderes Präfix verwenden, aber das Präfix muss in allen Befehlen und Warteschlangenkonfigurationen, die auf den Kanal verweisen, konsistent sein.

Um ein aktualisiertes Paket neu zu erstellen und zu veröffentlichen, fügen Sie `--build-number=+1` die Build-Nummer hinzu, um sie automatisch zu erhöhen.

```
rattler-build publish blender-4.5/recipe/recipe.yaml \
    --to s3://amzn-s3-demo-bucket/Conda/Default \
    --build-number=+1
```

Wenn Ihr Paketrezept von Paketen aus einem bestimmten Channel abhängt, wie z. B. conda-forge, fügen Sie dem Befehl etwas `-c conda-forge` hinzu.

Sie können auch eine Paketdatei veröffentlichen, die Sie bereits erstellt haben, z. B. eine `.conda` Datei aus einem lokalen Build. *`amzn-s3-demo-bucket`*Ersetzen Sie es durch Ihren Bucket-Namen.

```
rattler-build publish output/linux-64/blender-4.5.0-hb0f4dca_0.conda \
    --to s3://amzn-s3-demo-bucket/Conda/Default
```

Das Paket testen

Nachdem Sie das Paket veröffentlicht haben, erstellen Sie ein temporäres Pixi-Projekt, um zu überprüfen, ob das Paket ordnungsgemäß funktioniert. Das Projekt installiert das Paket aus dem Amazon S3 S3-Channel.

Um das Paket zu testen

1. Erstellen Sie ein temporäres Testverzeichnis und initialisieren Sie ein Pixi-Projekt mit dem Amazon S3 S3-Kanal. Ersetzen Sie es *`amzn-s3-demo-bucket`* durch Ihren Bucket-Namen.

   ```
   mkdir package-test-env
   cd package-test-env
   pixi init --channel s3://amzn-s3-demo-bucket/Conda/Default
   ```

2. Fügen Sie das Paket dem Projekt hinzu.

   ```
   pixi add blender=4.5
   ```

3. Stellen Sie sicher, dass das Paket ordnungsgemäß funktioniert.

   ```
   pixi run blender --version
   ```

Bereinigen

Entfernen Sie nach dem Testen das Testprojektverzeichnis.

Um Testressourcen zu bereinigen

- Entfernen Sie das Testprojektverzeichnis.

  Führen Sie macOS unter Linux und den folgenden Befehl aus.

  ```
  rm -rf package-test-env
  ```

  Führen Sie auf Windows (cmd) den folgenden Befehl aus.

  ```
  rmdir /s /q package-test-env
  ```

Builds debuggen

Wenn ein Build fehlschlägt, wird `rattler-build` das Build-Verzeichnis beibehalten, sodass Sie es untersuchen können. Führen Sie den folgenden Befehl aus, um eine interaktive Shell in der Build-Umgebung zu öffnen, in der alle Umgebungsvariablen so eingerichtet sind, wie sie während des Builds waren.

```
rattler-build debug shell
```

In der Debug-Shell können Sie Dateien ändern, einzelne Build-Befehle ausführen und Abhängigkeiten hinzufügen, um das Problem zu isolieren. Weitere Informationen finden Sie unter Debuggen von Builds in der Rattler-Build-Dokumentation.

Pakete für andere Plattformen erstellen

Der `rattler-build publish` Befehl erstellt Pakete für das Betriebssystem der Workstation, auf der der Befehl ausgeführt wird. Wenn Ihre Deadline Cloud-Flotte ein anderes Betriebssystem als Ihre Workstation verwendet oder wenn Ihr Paket andere Hostanforderungen hat, haben Sie die folgenden Optionen:

- `rattler-build publish`Auf einem Host ausführen, der dem Zielbetriebssystem entspricht. Verwenden Sie beispielsweise eine Amazon Elastic Compute Cloud (Amazon EC2) -Instance, die ausgeführt wirdLinux, um Pakete für eine Linux Flotte zu erstellen.

- Verwenden Sie eine Deadline Cloud-Warteschlange zur Paketerstellung, um Builds auf der Zielplattform zu automatisieren. Weitere Informationen finden Sie unter Erstellen einer Warteschlange zur Paketerstellung.
- (Fortgeschritten) Verwenden Sie Cross-Compilierung, um Pakete für eine andere Plattform als Ihre Workstation zu erstellen. Weitere Informationen finden Sie unter Cross-Compilierung in der Rattler-Build-Dokumentation.

Nächste Schritte

Nachdem Sie Pakete in Ihrem Amazon S3 S3-Conda-Kanal veröffentlicht haben, konfigurieren Sie Ihre Deadline Cloud-Warteschlangen so, dass sie den Kanal verwenden:

- Konfigurieren Sie Produktionswarteschlangenberechtigungen für benutzerdefinierte Conda-Pakete — Gewähren Sie Ihren Produktionswarteschlangen schreibgeschützten Zugriff auf den Amazon S3 S3-Conda-Kanal.
- Einen Conda-Kanal zu einer Warteschlangenumgebung hinzufügen — Konfigurieren Sie die Warteschlangenumgebung so, dass Pakete aus dem Amazon S3 S3-Conda-Kanal installiert werden.

Konfigurieren Sie die Berechtigungen für die Produktionswarteschlange für benutzerdefinierte Conda-Pakete

Ihre Produktionswarteschlange benötigt nur Leseberechtigungen für das `/Conda` Präfix im S3-Bucket der Warteschlange. Öffnen Sie die Seite AWS Identity and Access Management (IAM) für die Rolle, die der Produktionswarteschlange zugeordnet ist, und ändern Sie die Richtlinie wie folgt:

1. Öffnen Sie die Deadline Cloud-Konsole und navigieren Sie zur Seite mit den Warteschlangendetails für die Warteschlange zur Paketerstellung.
2. Wählen Sie die Warteschlangendienst-Rolle und anschließend Warteschlange bearbeiten aus.
3. Scrollen Sie zum Abschnitt Warteschlangendienstrolle und wählen Sie dann Diese Rolle in der IAM-Konsole anzeigen aus.
4. Wählen Sie aus der Liste der Berechtigungsrichtlinien die AmazonDeadlineCloudQueuePolicyfür Ihre Warteschlange aus.
5. Wählen Sie auf der Registerkarte „Berechtigungen“ die Option Bearbeiten aus.

6. Fügen Sie der Warteschlangendienst-Rolle einen neuen Abschnitt wie den folgenden hinzu. Ersetzen Sie *amzn-s3-demo-bucket* und *111122223333* durch Ihren eigenen Bucket und Ihr eigenes Konto.

```
{
   "Effect": "Allow",
   "Sid": "CustomCondaChannelReadOnly",
   "Action": [
    "s3:GetObject",
    "s3:ListBucket"
   ],
   "Resource": [
    "arn:aws:s3:::amzn-s3-demo-bucket",
    "arn:aws:s3:::amzn-s3-demo-bucket/Conda/*"
   ],
   "Condition": {
    "StringEquals": {
     "aws:ResourceAccount": "111122223333"
    }
   }
  },
```

Fügen Sie einer Warteschlangenumgebung einen Conda-Kanal hinzu

Um den S3-Conda-Kanal zu verwenden, müssen Sie den `s3://amzn-s3-demo-bucket/Conda/Default` Kanalstandort zum `CondaChannels` Parameter von Jobs hinzufügen, die Sie an Deadline Cloud senden. Die mit Deadline Cloud bereitgestellten Einreicher stellen Felder zur Verfügung, um benutzerdefinierte Conda-Kanäle und -Pakete anzugeben.

Sie können vermeiden, jeden Job zu ändern, indem Sie die Conda-Warteschlangenumgebung für Ihre Produktionswarteschlange bearbeiten. Führen Sie die folgenden Schritte aus:

1. Öffnen Sie die Deadline Cloud-Konsole und navigieren Sie zur Seite mit den Warteschlangendetails für die Produktionswarteschlange.
2. Wählen Sie den Tab Umgebungen.
3. Wählen Sie die Conda-Warteschlangenumgebung und dann Bearbeiten aus.
4. Wählen Sie den JSON-Editor und suchen Sie dann im Skript nach der Parameterdefinition für`CondaChannels`.

5. Bearbeiten Sie die Zeile `default: "deadline-cloud"` so, dass sie mit dem neu erstellten S3-Conda-Kanal beginnt:

```
default: "s3://amzn-s3-demo-bucket/Conda/Default deadline-cloud"
```

Serviceverwaltete Flotten ermöglichen standardmäßig eine flexible Kanalpriorität für Conda. Bei einem Job, der anfragt, `blender=4.2` ob Blender 4.2 sowohl im neuen Channel als auch im Channel enthalten ist, wird das Paket aus dem `deadline-cloud` Channel abgerufen, der zuerst in der Channel-Liste steht. Wenn eine angegebene Paketversion nicht im ersten Kanal gefunden wird, werden die nachfolgenden Kanäle auf die Paketversion überprüft.

Für vom Kunden verwaltete Flotten können Sie die Verwendung von Conda-Paketen aktivieren, indem Sie eines der Conda-Warteschlangenumgebungsbeispiele im Deadline Cloud-Beispiel-Repository verwenden. GitHub

Erstellen Sie ein Conda-Paket für eine Anwendung oder ein Plugin

Ein Conda-Paket ist ein komprimiertes Archiv mit Software, die in einer beliebigen Sprache geschrieben wurde. Conda unterstützt eine Vielzahl von Betriebssystem- und Architekturkombinationen, sodass Sie vollständige Anwendungen wie BlenderMaya, und Nuke zusammen mit Bibliotheken für Python und andere Sprachen paketieren können. Weitere Informationen zu Conda-Paketen finden Sie unter Pakete in der Conda-Dokumentation.

Um ein Conda-Paket zu verwenden, installieren Sie es in einer virtuellen Umgebung. Eine virtuelle Conda-Umgebung hat ein Präfixverzeichnis, in dem Pakete installiert werden. Bei der Installation eines Pakets werden Dateien fest verknüpft oder neu verlinkt, sofern dies unterstützt wird. Das Erstellen mehrerer Umgebungen mit denselben Paketen verbraucht also keinen nennenswerten zusätzlichen Festplattenspeicher. Um eine virtuelle Umgebung zu verwenden, aktivieren Sie sie, um Umgebungsvariablen festzulegen. Bei der Aktivierung werden von Paketen bereitgestellte Skripts ausgeführt, sodass jedes Paket die Möglichkeit hat, PATH oder andere Umgebungsvariablen zu ändern. Conda-Pakete enthalten normalerweise Anwendungen oder Bibliotheken, aber aufgrund der flexiblen Aktivierung können sie auch auf Anwendungen verweisen, die auf einem gemeinsam genutzten Dateisystem installiert sind.

Die Erstellung eines benutzerdefinierten Pakets umfasst drei Phasen: Ein Rezept enthält die Bauanweisungen, ein Paket ist das gebaute Artefakt (`.conda`oder die erstellte `.tar.bz2` Datei) und ein Channel hostet die Pakete für die Installation. Der `rattler-build publish` Befehl behandelt

alle drei Schritte: Er kann ein Rezept in ein Paket einbauen und in einem Channel veröffentlichen, oder es kann direkt ein Paket-Artefakt verwenden, um es zu veröffentlichen.

Die Conda-Forge-Community verwaltet Paketrezepte für eine breite Palette von Open-Source-Software und hostet Paket-Artefakte im Channel. `conda-forge` Sie können Ihre Warteschlange so konfigurieren, dass sie sie `conda-forge` als Paketquelle einschließt, und dann benutzerdefinierte Pakete erstellen, deren Ausführung von Conda-Forge-Paketen abhängt. Denn Linux Conda-Forge hostet eine vollständige Compiler-Toolchain einschließlich CUDA-Unterstützung, wobei konsistente Kompilierungs- und Linkoptionen ausgewählt sind. Sie können Conda-Forge-Pakete als Abhängigkeiten in Ihren eigenen Rezepten verwenden oder sie zusammen mit Ihren benutzerdefinierten Paketen in derselben Umgebung installieren.

Sie können eine gesamte Anwendung, einschließlich Abhängigkeiten, zu einem Conda-Paket kombinieren. Die Pakete, die Deadline Cloud im Deadline-Cloud-Channel für serviceverwaltete Flotten bereitstellt, verwenden diesen binären Repacking-Ansatz. Dadurch werden dieselben Dateien wie bei einer Installation so organisiert, dass sie zur virtuellen Conda-Umgebung passen.

> **Note**
>
> Große Anwendungen können Dutzende GB freien Festplattenspeicher für das Quellarchiv, die entpackten Dateien und die Build-Ausgabe benötigen. Stellen Sie sicher, dass Sie eine Festplatte mit ausreichend verfügbarem Speicherplatz für die Paketbauausgabe verwenden.

Eine Anwendung verpacken

Beim Umpacken einer Anwendung für Conda gibt es zwei Ziele:

- Die meisten Dateien für die Anwendung sollten von der primären Struktur der virtuellen Conda-Umgebung getrennt sein. Umgebungen können die Anwendung dann mit Paketen aus anderen Quellen wie Conda-Forge mischen.
- Wenn eine virtuelle Conda-Umgebung aktiviert ist, sollte die Anwendung über die Umgebungsvariable PATH verfügbar sein.

Um eine Anwendung für Conda neu zu verpacken

1. Schreiben Sie Conda-Build-Rezepte, die die Anwendung in ein Unterverzeichnis wie installieren. `$CONDA_PREFIX/opt/<application-name>` Dies unterscheidet es von den Standardpräfixverzeichnissen wie `bin` und. `lib`
2. Fügen Sie Symlinks oder Startskripten hinzu, `$CONDA_PREFIX/bin` um die Anwendungsbinärdateien auszuführen.

   Alternativ können Sie activate.d-Skripten erstellen, die der `conda activate` Befehl ausführt, um die Binärverzeichnisse der Anwendung zum PATH hinzuzufügen. Wenn Symlinks nicht überall unterstützt werdenWindows, wo Umgebungen erstellt werden können, verwenden Sie stattdessen die Skripten application launch oder activate.d.
3. Manche Anwendungen sind auf Bibliotheken angewiesen, die nicht standardmäßig auf den vom Service verwalteten Flotten von Deadline Cloud installiert sind. Beispielsweise ist das X11-Fenstersystem für nicht interaktive Jobs normalerweise nicht erforderlich, aber für einige Anwendungen muss es immer noch ohne grafische Oberfläche ausgeführt werden. Sie müssen diese Abhängigkeiten in dem Paket angeben, das Sie erstellen.
4. Wenn die Anwendung Plugins unterstützt, geben Sie eine klare Konvention an, nach der Plugin-Pakete bei der Integration in die Anwendung in einer virtuellen Umgebung funktionieren müssen. Zum Beispiel dokumentiert das Maya2026-Beispielrezept diese Konvention für Maya Plugins.
5. Stellen Sie sicher, dass Sie die Urheber- und Lizenzvereinbarungen für die von Ihnen verpackten Anwendungen einhalten. Wir empfehlen, einen privaten Amazon S3 S3-Bucket für Ihren Conda-Kanal zu verwenden, um die Verteilung zu kontrollieren und den Paketzugriff auf Ihre Farm einzuschränken.

Beispielrezepte für die Pakete im `deadline-cloud` Channel sind im Deadline Cloud-Beispiel-Repository unter GitHub verfügbar.

Ein Plugin verpacken

Anwendungs-Plugins können als eigene Conda-Pakete verpackt werden. Beachten Sie beim Erstellen eines Plugin-Pakets die folgenden Richtlinien:

- Nehmen Sie das Host-Anwendungspaket sowohl als Build- als auch als Run-Abhängigkeit in das Build-Rezept auf`recipe.yaml`. Verwenden Sie eine Versionsbeschränkung, sodass das Build-Rezept nur mit kompatiblen Paketen installiert wird.

- Halten Sie sich bei der Registrierung des Plug-ins an die Konventionen für Host-Anwendungspakete.

Adapter-Pakete

Einige Anwendungsintegrationen von Deadline Cloud verwenden einen Adapter, der die Anwendungsschnittstelle erweitert, um das Schreiben von Jobvorlagen zu vereinfachen. Ein Adapter ist eine Befehlszeilenschnittstelle, die die Ausführung eines Hintergrund-Daemons, die Statusberichterstattung und die Pfadzuweisung unterstützt. Weitere Informationen finden Sie unter Open Job Description Adaptor Runtime aufGitHub. Zum Beispiel GitHub enthält deadline-cloud-for-mayaon eine integrierte Benutzeroberfläche für die Auftragsübermittlung und einen Maya Adapter, der als `maya-openjd` Paket für vom Service verwaltete Flotten verfügbar ist.

Auftragsübermittlungen von Deadline Cloud Submitter GUIs enthalten einen `CondaPackages` Parameterwert, der die Conda-Pakete angibt, die in einer virtuellen Umgebung für die Ausführung des Jobs enthalten sein sollen. Der `CondaPackages` Parameterwert für sieht Maya normalerweise so aus `maya=2025.* maya-openjd=0.15.* maya-mtoa` und kann alternative Einträge für Plugin-Pakete enthalten. Wenn die Warteschlangenumgebung eine virtuelle Conda-Umgebung für die Ausführung des Jobs einrichtet, löst sie diese Paketnamen und Versionseinschränkungen auf, sodass sie kompatibel sind, und fügt alle Abhängigkeitspakete hinzu, die sie zur Ausführung benötigen. Jedes Adapter- und Plugin-Paket gibt an, womit es kompatibel ist, einschließlich welcher Versionen vonMaya, welchen Versionen von Python und anderer Abhängigkeiten.

Um Ihre eigenen Adapterpakete mit unseren Beispielen wie dem Maya-Openjd-Rezept auf zu erstellenGitHub, können Sie auf den Paketen für Python und anderen Abhängigkeiten aufbauen, die von conda-forge bereitgestellt werden. Möglicherweise müssen Sie zuerst die Frist und openjd-adaptor-runtimedie Rezepte erstellen, um Abhängigkeiten zu erfüllen.

Erstellen Sie ein Conda-Build-Rezept für Blender

Sie können verschiedene Anwendungen verwenden, um ein Conda-Build-Rezept zu erstellen. Blenderist kostenlos zu verwenden und lässt sich einfach mit Conda verpacken. Die Blender Foundation stellt Anwendungsarchive für mehrere Betriebssysteme zur Verfügung. Wir haben ein Beispiel für ein Conda-Build-Rezept erstellt, das die Windows-Dateien .zip und .tar.xz für Linux verwendet. In diesem Abschnitt erfahren Sie, wie Sie das Conda-Build-Rezept von 4.2 verwenden. Blender

Die Datei deadline-cloud.yaml spezifiziert die Conda-Plattformen und andere Metadaten für die Übermittlung von Paketaufträgen an Deadline Cloud. Dieses Rezept enthält Informationen zum lokalen Quellarchiv, um zu demonstrieren, wie das funktioniert. Die Linux-64-Conda-Plattform ist so eingestellt, dass sie eine Standardkonfiguration für die Einreichung von Jobs einbaut, die der gängigsten Konfiguration entspricht. Die Datei deadline-cloud.yaml sieht etwa wie folgt aus:

```
condaPlatforms:
  - platform: linux-64
    defaultSubmit: true
    sourceArchiveFilename: blender-4.2.1-linux-x64.tar.xz
    sourceDownloadInstructions: 'Run "curl -LO https://download.blender.org/release/
Blender4.2/blender-4.2.1-linux-x64.tar.xz"'
  - platform: win-64
    defaultSubmit: false
    sourceArchiveFilename: blender-4.2.1-windows-x64.zip
    sourceDownloadInstructions: 'Run "curl -LO https://download.blender.org/release/
Blender4.2/blender-4.2.1-windows-x64.zip"'
```

Überprüfen Sie die Dateien im Verzeichnis. `recipe` Die Metadaten für das Rezept befinden sich in recipe/recipe.yaml. Sie können auch die Dokumentation conda build meta.yaml lesen, um mehr zu erfahren, z. B. darüber, wie die Datei eine Vorlage zur Generierung von YAML ist. Die Vorlage wird verwendet, um die Versionsnummer nur einmal anzugeben und je nach Betriebssystem unterschiedliche Werte bereitzustellen.

Sie können die unter ausgewählten Build-Optionen überprüfen`meta.yaml`, um verschiedene Prüfungen für die binäre Verlagerung und das Verknüpfen von dynamischen gemeinsamen Objekten (DSO) zu deaktivieren. Diese Optionen steuern, wie das Paket funktioniert, wenn es in einer virtuellen Conda-Umgebung unter einem beliebigen Verzeichnispräfix installiert wird. Die Standardwerte vereinfachen das Verpacken jeder Abhängigkeitsbibliothek in ein separates Paket, aber wenn Sie eine Anwendung binär neu verpacken, müssen Sie sie ändern.

Wenn für die Anwendung, die Sie verpacken, zusätzliche Abhängigkeitsbibliotheken erforderlich sind oder Sie Plugins für eine Anwendung separat verpacken, können DSO-Fehler auftreten. Diese Fehler treten auf, wenn sich die Abhängigkeit nicht im Bibliothekssuchpfad für die ausführbare Datei oder Bibliothek befindet, die sie benötigt. Anwendungen sind darauf angewiesen, dass sich Bibliotheken in global definierten Pfaden wie `/lib` oder befinden`/usr/lib`, wenn sie auf einem System installiert sind. Da virtuelle Conda-Umgebungen jedoch überall platziert werden können, gibt es keinen absoluten Pfad, der verwendet werden kann. Conda verwendet relative RPATH-

Funktionen, die sowohl als auch macOS unterstützt werden, um Linux dies zu handhaben. Weitere Informationen finden Sie in der Conda-Build-Dokumentation unter Pakete verschiebbar machen.

Blendererfordert keine RPATH-Anpassung, da die Anwendungsarchive unter Berücksichtigung dieser Tatsache erstellt wurden. Für Anwendungen, die dies benötigen, können Sie dieselben Tools verwenden wie Conda Build: `patchelf` unter Linux und `install_name_tool` anderen. macOS

Während der Paketerstellung wird das Skript build.sh oder build_win.sh (aufgerufen von`bld.bat`) ausgeführt, um Dateien in einer Umgebung zu installieren, die mit den Paketabhängigkeiten vorbereitet ist. Diese Skripten kopieren die Installationsdateien, erstellen Symlinks von `$PREFIX/bin` ihnen und richten die Aktivierungsskripten ein. Bei aktivierter Option werden keine Symlinks erstelltWindows, sondern stattdessen das Blender-Verzeichnis zum PATH im Aktivierungsskript hinzugefügt.

Wir verwenden `bash` statt `cmd.exe` einer.bat-Datei für den Windows Teil des Conda-Build-Rezepts, da dies für mehr Konsistenz zwischen den Build-Skripten sorgt. Tipps zur Verwendung `bash` von finden Sie in der Empfehlung zur Workload-Portabilität im Deadline Cloud-Entwicklerhandbuch. Windows Wenn Sie Git for installiert habenWindows, um das deadline-cloud-samplesGit-Repository zu klonen, haben Sie bereits Zugriff auf`bash`.

In der Dokumentation zu den Conda-Build-Umgebungsvariablen sind die Werte aufgeführt, die für die Verwendung im Build-Skript verfügbar sind. Zu diesen Werten gehören `$SRC_DIR` für die Quellarchivdaten, `$PREFIX` für das Installationsverzeichnis, für den `$RECIPE_DIR` Zugriff auf andere Dateien aus dem Rezept, `$PKG_NAME` `$PKG_VERSION` für den Paketnamen und die Version sowie `$target_platform` für die Ziel-Conda-Plattform.

Reichen Sie den Blender 4.2-Paketjob ein

Sie können Ihr eigenes Blender 4.2-Conda-Paket erstellen, um Jobs zu rendern, indem Sie das Blender Archiv herunterladen und dann einen Job an die Paketbau-Warteschlange senden. Die Warteschlange sendet den Job an die zugehörige Flotte, um das Paket zu erstellen und den Conda-Kanal neu zu indizieren.

Diese Anweisungen verwenden Git aus einer Bash-kompatiblen Shell, um einen OpenJD-Paketbaujob und einige Conda-Rezepte aus dem Deadline Cloud-Beispiel-Repository abzurufen. GitHub Sie benötigen außerdem Folgendes:

- Wenn Sie Git verwendenWindows, wird bei der Installation von Git eine Version von Bash, git BASH, installiert.

- Sie müssen die Deadline Cloud CLI installiert haben.
- Sie müssen beim Deadline Cloud-Monitor angemeldet sein.

1. Öffnen Sie die Deadline Cloud-Konfigurations-GUI mit dem folgenden Befehl und legen Sie die Standardfarm und -warteschlange auf Ihre Paketerstellungswarteschlange fest.

```
deadline config gui
```

2. Verwenden Sie den folgenden Befehl, um das Deadline GitHUb Cloud-Beispiel-Repository zu klonen.

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
```

3. Wechseln Sie zu dem `conda_recipes` Verzeichnis im `deadline-cloud-samples` Verzeichnis.

```
cd deadline-cloud-samples/conda_recipes
```

4. Führen Sie das aufgerufene Skript aus`submit-package-job`. Das Skript enthält Anweisungen zum HerunterladenBlender, wenn Sie das Skript zum ersten Mal ausführen.

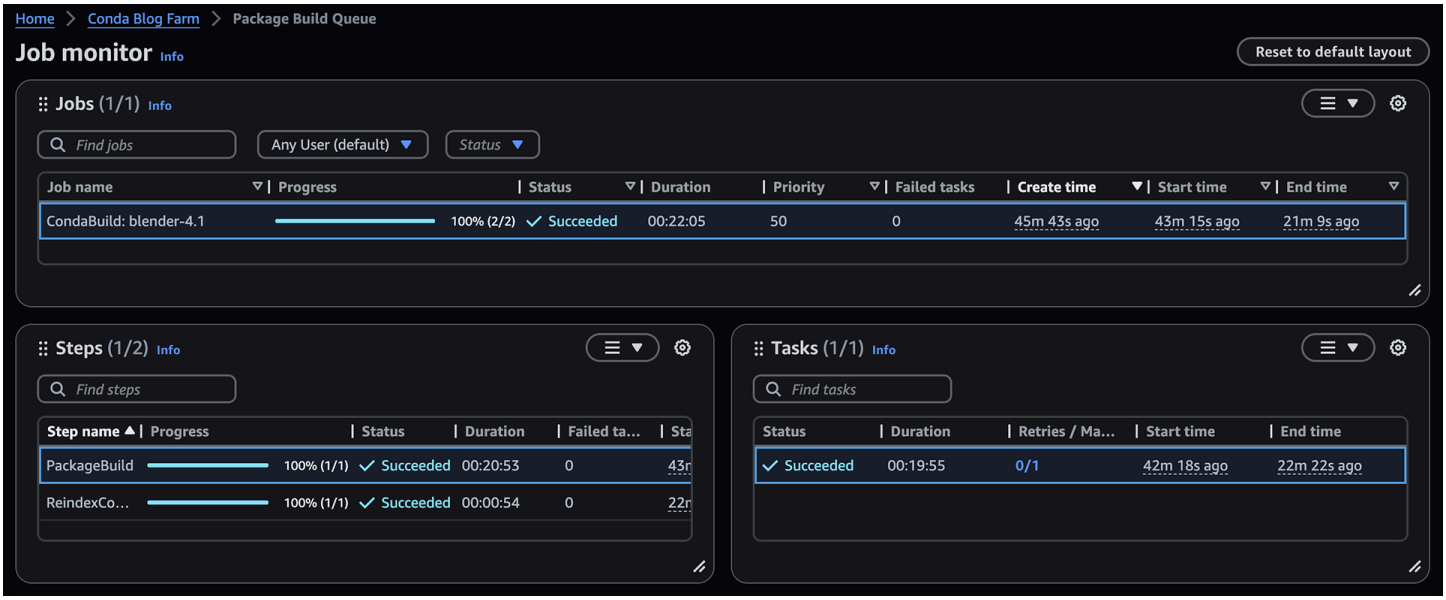
```
./submit-package-job blender-4.2/
```

5. Folgen Sie den Anweisungen zum HerunterladenBlender. Wenn Sie das Archiv haben, führen Sie das `submit-package-job` Skript erneut aus.

```
./submit-package-job blender-4.2/
```

Nachdem Sie den Job eingereicht haben, können Sie den Deadline Cloud-Monitor verwenden, um den Fortschritt und Status des Jobs während der Ausführung zu verfolgen.

In der unteren linken Ecke des Monitors werden die beiden Schritte des Jobs angezeigt: das Erstellen des Pakets und die anschließende Neuindizierung. Unten rechts werden die einzelnen Schritte für jede Aufgabe angezeigt. In diesem Beispiel gibt es für jede Aufgabe einen Schritt.



In der unteren linken Ecke des Monitors befinden sich die beiden Schritte des Jobs: das Erstellen des Pakets und das anschließende Neuindizieren des Conda-Kanals. Unten rechts befinden sich die einzelnen Aufgaben für jeden Schritt. In diesem Beispiel gibt es für jeden Schritt nur eine Aufgabe.

Wenn Sie mit der rechten Maustaste auf die Aufgabe für den Schritt zur Paketerstellung klicken und Protokolle anzeigen auswählen, zeigt der Monitor eine Liste von Sitzungsaktionen an, die zeigen, wie die Aufgabe für den Worker geplant ist. Die Aktionen sind:

- Anlagen synchronisieren — Diese Aktion kopiert die Eingabe-Job-Anhänge oder mountet ein virtuelles Dateisystem, je nachdem, welche Einstellung für das Dateisystem mit den Job-Anhängen verwendet wurde.
- Conda starten — Diese Aktion stammt aus der Warteschlangenumgebung, die standardmäßig hinzugefügt wurde, als Sie die Warteschlange erstellt haben. Der Job spezifiziert keine Conda-Pakete, sodass er schnell abgeschlossen wird und keine virtuelle Conda-Umgebung erstellt.
- CondaBuild Env starten — Diese Aktion erstellt eine benutzerdefinierte virtuelle Conda-Umgebung, die die Software enthält, die zum Erstellen eines Conda-Pakets und zur Neuindizierung eines Kanals erforderlich ist. Sie wird über den Conda-Forge-Kanal installiert.
- Aufgabe ausführen — Diese Aktion erstellt das Blender Paket und lädt die Ergebnisse auf Amazon S3 hoch.

Während die Aktionen ausgeführt werden, senden sie Protokolle in einem strukturierten Format an Amazon CloudWatch. Wenn ein Job abgeschlossen ist, wählen Sie Protokolle für alle Aufgaben

anzeigen aus, um zusätzliche Protokolle über den Auf- und Abbau der Umgebung, in der der Job ausgeführt wird, einzusehen.

Testen Sie Ihr Paket mit einem Blender 4.2-Renderjob

Nachdem Sie das Blender 4.2-Paket erstellt und Ihre Produktionswarteschlange für die Verwendung des S3-Conda-Kanals konfiguriert haben, können Sie Jobs zum Rendern mit dem Paket einreichen. Wenn Sie keine Szene haben, laden Sie die Blender Szene Blender 3.5 — Cozy Kitchen von der Seite mit den BlenderDemo-Dateien herunter.

Das Deadline GitHub Cloud-Beispiel-Repository, das Sie zuvor heruntergeladen haben, enthält einen Beispieljob zum Rendern einer Blender Szene mit den folgenden Befehlen:

```
deadline bundle submit blender_render \
     -p CondaPackages=blender=4.2 \
     -p BlenderSceneFile=/path/to/downloaded/blender-3.5-splash.blend \
     -p Frames=1
```

Sie können den Deadline Cloud-Monitor verwenden, um den Fortschritt Ihres Jobs zu verfolgen:

1. Wählen Sie im Monitor die Aufgabe für den Job aus, den Sie eingereicht haben, und wählen Sie dann die Option, um das Protokoll anzuzeigen.
2. Wählen Sie auf der rechten Seite der Protokollansicht die Aktion Conda-Sitzung starten aus.

Sie können sehen, dass die Aktion in den beiden für die Warteschlangenumgebung konfigurierten Conda-Kanälen nach Blender 4.2 gesucht hat und dass sie das Paket im S3-Kanal gefunden hat.

Erstellen Sie ein Conda-Build-Rezept für Autodesk Maya

Sie können kommerzielle Anwendungen als Conda-Pakete verpacken. In Create a Conda build recipe for haben Sie gelerntBlender, wie Sie eine Anwendung verpacken, die als einfache, verschiebbare Archivdatei und unter Open-Source-Lizenzbedingungen verfügbar ist. Kommerzielle Anwendungen werden häufig über Installationsprogramme verteilt und verfügen möglicherweise über ein Lizenzverwaltungssystem, mit dem sie arbeiten können.

Die folgende Liste baut auf den Grundlagen auf, die unter Erstellen eines Conda-Pakets für eine Anwendung oder ein Plug-In behandelt werden, und zwar mit den Anforderungen, die üblicherweise mit der Paketierung kommerzieller Anwendungen verbunden sind. Die Einzelheiten in den Unteraufzählungen veranschaulichen, wie Sie die Richtlinien anwenden können. Maya

- Machen Sie sich mit den Lizenzrechten und Einschränkungen der Anwendung vertraut. Möglicherweise müssen Sie ein Lizenzverwaltungssystem konfigurieren. Wenn das Programm keine Durchsetzung vorsieht, müssen Sie Ihre Farm entsprechend Ihren Rechten konfigurieren.
  - Lesen Sie die häufig gestellten Fragen zu Autodesk Abonnementvorteilen zu Cloud-Rechten, um zu erfahrenMaya, welche Cloud-Rechte für Sie gelten könnten. Konfigurieren Sie Ihre Deadline Cloud-Farm nach Bedarf.
  - AutodeskProdukte basieren auf einer Datei namens`ProductInformation.pit`. Für die meisten Konfigurationen dieser Datei ist Administratorzugriff auf das System erforderlich. Dieser Zugriff ist für vom Service verwaltete Flotten nicht verfügbar. Produktfunktionen für Thin Clients bieten eine Möglichkeit, dieses Problem an einem anderen Ort zu lösen. Weitere Informationen finden Sie unter Thin Client-Lizenzierung für Maya. MotionBuilder
- Manche Anwendungen hängen von Bibliotheken ab, die nicht auf den vom Service verwalteten Fleet Worker-Hosts installiert sind, weshalb das Paket sie bereitstellen muss. Dies kann direkt im Anwendungspaket oder in einem separaten Abhängigkeitspaket enthalten sein.
  - Mayahängt von einer Reihe solcher Bibliotheken ab, darunter freetype und fontconfig. Wenn diese Bibliotheken im Systempaketmanager verfügbar sind, z. B. in `dnf` for AL2023, können Sie sie als Quelle für die Anwendung verwenden. Da diese RPM-Pakete nicht so konzipiert sind, dass sie verschoben werden können, müssen Sie Tools verwenden, die sicherstellen, dass Abhängigkeiten innerhalb des Maya Installationspräfixes aufgelöst werden. `patchelf`
- Für die Installation ist möglicherweise Administratorzugriff erforderlich. Da vom Service verwaltete Flotten keinen Administratorzugriff bieten, müssen Sie eine Installation auf einem System mit diesem Zugriff durchführen. Erstellen Sie anschließend ein Archiv mit den Dateien, die für den Paketerstellungsauftrag benötigt werden.
  - Das Windows Installationsprogramm für Maya erfordert Administratorzugriff, daher erfordert das Erstellen des Conda-Pakets einen manuellen Prozess, um zunächst ein solches Archiv zu erstellen.
- Die Anwendungskonfiguration, einschließlich der Art und Weise, wie sich Plugins bei der Anwendung registrieren, kann auf Betriebssystem- oder Benutzerebene definiert werden. Wenn Plugins in einer virtuellen Conda-Umgebung platziert werden, müssen sie so in die Anwendung integriert werden können, dass sie in sich geschlossen sind und niemals Dateien oder andere Daten außerhalb des Präfixes der virtuellen Umgebung schreiben. Wir empfehlen Ihnen, dies über das Conda-Paket der Anwendung einzurichten.
  - Das Maya Beispielpaket definiert die Umgebungsvariable, `MAYA_NO_HOME=1` um sie von der Konfiguration auf Benutzerebene zu isolieren, und fügt Modulsuchpfade hinzu, `MAYA_MODULE_PATH` sodass separat verpackte Plugins aus der virtuellen Umgebung heraus

integriert werden können. Das MtoA Beispielpaket platziert eine .mod-Datei in einem dieser Verzeichnisse, die beim Maya Start geladen wird.

Schreiben Sie die Rezept-Metadaten

1. Öffnen Sie das Verzeichnis GitHub deadline-cloud-samples/conda_recipes/maya-2025 in Ihrem Browser oder in einem Texteditor in Ihrem lokalen Klon des Repositorys.

   Die Datei `deadline-cloud.yaml` beschreibt die Conda-Build-Plattformen, für die Pakete erstellt werden sollen, und wo die Anwendung herkommt. Das Rezeptbeispiel spezifiziert Linux sowohl als auch Windows Builds, und nur das Linux wird standardmäßig übermittelt.

2. Laden Sie die vollständigen Maya Installationsprogramme von Ihrem Autodesk Anmeldenamen herunter. Denn Linux der Paket-Build kann das Archiv direkt verwenden. Platzieren Sie es also direkt im `conda_recipes/archive_files` Verzeichnis. Denn Windows das Installationsprogramm benötigt Administratorzugriff, um ausgeführt zu werden. Sie müssen das Installationsprogramm ausführen und die erforderlichen Dateien in einem Archiv für das Paketrezept sammeln, das Sie verwenden möchten. Die Datei README.md im Rezept dokumentiert ein wiederholbares Verfahren zur Erstellung dieses Artefakts. Das Verfahren verwendet eine neu gestartete Amazon EC2 EC2-Instance, um eine saubere Umgebung für die Installation bereitzustellen, die Sie dann beenden können, nachdem Sie das Ergebnis gespeichert haben. Um andere Anwendungen zu verpacken, für die Administratorzugriff erforderlich ist, können Sie ein ähnliches Verfahren anwenden, sobald Sie den Satz von Dateien festgelegt haben, den die Anwendung benötigt.

3. Öffnen Sie die Dateien recipe/recipe.yaml und recipe/meta.yaml, um die Einstellungen für Rattler-Build und Conda-Build zu überprüfen oder zu bearbeiten. Sie können den Paketnamen und die Version für die Anwendung festlegen, die Sie verpacken.

   Der Quellbereich enthält einen Verweis auf die Archive, einschließlich des Sha256-Hashs der Dateien. Immer wenn Sie diese Dateien ändern, z. B. auf eine neue Version, müssen Sie diese Werte berechnen und aktualisieren.

   Der Build-Abschnitt enthält hauptsächlich Optionen zum Ausschalten der standardmäßigen Binärverschiebungsoptionen, da die automatischen Mechanismen für die speziellen Bibliotheks- und Binärverzeichnisse, die das Paket verwendet, nicht richtig funktionieren.

   Schließlich können Sie im Abschnitt „Über“ einige Metadaten über die Anwendung eingeben, die beim Durchsuchen oder Verarbeiten des Inhalts eines Conda-Kanals verwendet werden können.

Schreiben Sie das Paketerstellungsskript

1. Die Paketerstellungsskripten im Maya Conda-Beispielbaurezept enthalten Kommentare, in denen die Schritte erklärt werden, die die Skripts ausführen. Lesen Sie sich die Kommentare und Befehle durch, um Folgendes herauszufinden:

   - Wie das Rezept mit der RPM-Datei umgeht von Autodesk
   - Die Änderungen, die das Rezept vornimmt, damit die Installation in die virtuellen Conda-Umgebungen verlagert werden kann, in denen das Rezept installiert ist
   - Wie das Rezept Hilfsvariablen festlegt`MAYA_VERSION`, z. B. `MAYA_LOCATION` und anhand derer Ihre Software nachvollziehen kann, ob Maya sie ausgeführt wird.

2. Öffnen Sie zum Beispiel die Datei recipe/build.sh Linux, um das Paketerstellungsskript zu überprüfen oder zu bearbeiten.

   Öffnen Sie für die Datei recipe/build_win.sh Windows, um das Paketerstellungsskript zu überprüfen oder zu bearbeiten.

Reichen Sie einen Job ein, der die Maya Pakete erstellt

1. Geben Sie das `conda_recipes` Verzeichnis in Ihrem Klon des GitHub deadline-cloud-samplesRepositorys ein.
2. Stellen Sie sicher, dass Ihre Deadline Cloud-Farm für Ihre Deadline Cloud-CLI konfiguriert ist. Wenn Sie die Schritte zum Erstellen eines Conda-Kanals mit Amazon S3 befolgt haben, sollte Ihre Farm für Ihre CLI konfiguriert sein.
3. Führen Sie den folgenden Befehl aus, um einen Job einzureichen, der Linux sowohl Windows Pakete als auch erstellt.

   ```
   ./submit-package-job maya-2025 --all-platforms
   ```

Erstellen Sie ein Conda-Build-Rezept für das Plugin Autodesk Maya to Arnold (MtoA)

Sie können Plugins für kommerzielle Anwendungen als Conda-Pakete verpacken. Plugins sind dynamisch geladene Bibliotheken, die eine von einer Anwendung bereitgestellte Anwendungsbinärschnittstelle (ABI) verwenden, um die Funktionalität dieser Anwendung zu erweitern. Das Maya to Arnold (MtoA) Plugin fügt den Arnold Renderer als Option hinzu. Maya

- Das MtoA Beispiel-Build-Rezept hängt vom MayaPaket ab und verwendet eine == Einschränkung für die Version.
- Das Maya Paket konfiguriert einen Maya Modulpfad in der virtuellen Umgebung`$PREFIX/usr/autodesk/maya$MAYA_VERSION/modules`, in dem das Plugin eine `.mod` Datei platzieren kann. Das MtoA Beispiel-Build-Rezept erstellt eine Datei `mtoa.mod` in diesem Verzeichnis.

Schreiben Sie die Rezept-Metadaten

1. Öffnen Sie das Verzeichnis GitHub deadline-cloud-samples/conda_recipes/maya-mtoa-2025 in Ihrem Browser oder in einem Texteditor in Ihrem lokalen Klon des Repositorys.

   Das Rezept folgt den gleichen Mustern wie das Maya Conda-Build-Rezept und verwendet dieselben Quellarchive, um das Plugin zu installieren.
2. Öffnen Sie die Dateien recipe/recipe.yaml und recipe/meta.yaml, um die Einstellungen für Rattler-Build und für Conda-Build zu überprüfen oder zu bearbeiten. Diese Dateien spezifizieren eine Abhängigkeit, von der während der Paketerstellung und beim Erstellen einer virtuellen Umgebung zur Ausführung des Plugins abhängig ist. `maya`

Schreiben Sie das Paketerstellungsskript

- Die Paketerstellungsskripten im MtoA Conda-Beispielbaurezept enthalten Kommentare, in denen die Schritte erklärt werden, die die Skripts ausführen. Lesen Sie sich die Kommentare und Befehle durch, um zu erfahren, wie das Rezept eine Datei `mtoa.mod` in dem im Maya Paket angegebenen Verzeichnis installiert MtoA und erstellt.

  Arnoldund Maya verwenden dieselbe Lizenzierungstechnologie, sodass das Maya Conda-Build-Rezept bereits die Informationen enthält, die von Arnold benötigt werden.

  Die Unterschiede zwischen den Skripten Linux und den Windows Build-Skripten ähneln denen für das Maya Conda-Build-Rezept.

Reichen Sie einen Job ein, der die Maya MtoA Plugin-Pakete erstellt

1. Geben Sie das `conda_recipes` Verzeichnis in Ihrem Klon des GitHub deadline-cloud-samplesRepositorys ein.
2. Stellen Sie sicher, dass Sie Pakete für die Maya Host-Anwendung aus dem vorherigen Abschnitt erstellt haben.

3. Stellen Sie sicher, dass Ihre Deadline Cloud-Farm für Ihre Deadline Cloud-CLI konfiguriert ist. Wenn Sie die Schritte zum Erstellen eines Conda-Kanals mit Amazon S3 befolgt haben, sollte Ihre Farm für Ihre CLI konfiguriert sein.
4. Führen Sie den folgenden Befehl aus, um einen Job einzureichen, der Linux sowohl Windows Pakete als auch erstellt.

   ```
   ./submit-package-job maya-mtoa-2025 --all-platforms
   ```

Testen Sie Ihr Paket mit einem Maya Renderjob

Nachdem Sie die Version Maya 2025 und die MtoA Pakete erstellt haben, können Sie Jobs zum Rendern mit dem Paket einreichen. Das Beispiel „Turntable with Maya/Arnold Job Bundle“ rendert eine Animation mit Maya und. Arnold Dieses Beispiel wird auch FFmpeg zum Kodieren eines Videos verwendet. Sie können den Conda-Forge-Kanal zur Standardliste `CondaChannels` in Ihrer Conda-Warteschlangenumgebung hinzufügen, um eine Quelle für das Paket bereitzustellen. `ffmpeg`

Führen Sie im `job_bundles` Verzeichnis in Ihrem Git-Klon von den deadline-cloud-samplesfolgenden Befehl aus.

```
deadline bundle submit turntable_with_maya_arnold
```

Sie können den Deadline Cloud-Monitor verwenden, um den Fortschritt Ihres Jobs zu verfolgen:

1. Wählen Sie im Monitor die Aufgabe für den Job aus, den Sie eingereicht haben, und wählen Sie dann die Option, um das Protokoll anzuzeigen.
2. Wählen Sie auf der rechten Seite der Protokollansicht die Aktion Conda-Sitzung starten aus.

Sie können sehen, dass die Aktion nach maya und maya-mtoa in den für die Warteschlangenumgebung konfigurierten Conda-Kanälen gesucht hat und dass sie die Pakete im S3-Kanal gefunden hat.

Automatisieren Sie Paketerstellungen mit Deadline Cloud

Für CI/CD Workflows oder wenn Sie Pakete für mehrere Betriebssysteme erstellen müssen, können Sie eine Deadline Cloud-Warteschlange zur Paketerstellung erstellen. Die Warteschlangenpläne erstellen Jobs in Ihrer Flotte, die die Pakete erstellen und sie in Ihrem Amazon Simple Storage Service (Amazon S3) Conda-Kanal veröffentlichen. Dies vereinfacht die Verwaltung kontinuierlicher Paket-Builds für Softwareversionen in all Ihren erforderlichen Konfigurationen.

Sie können eine Warteschlange zur Paketerstellung mithilfe einer Vorlage AWS CloudFormation (CloudFormation) oder manuell über die Deadline Cloud-Konsole erstellen. Die CloudFormation Vorlage stellt eine komplette Farm mit einer Produktionswarteschlange und einer Warteschlange zur Paketerstellung bereit, die bereits konfiguriert sind. Wenn Sie die Warteschlange von der Konsole aus erstellen, haben Sie mehr Kontrolle über einzelne Einstellungen.

Erstellen Sie eine Warteschlange zum Erstellen von Paketen mit CloudFormation

Sie können eine CloudFormation Vorlage verwenden, um eine Deadline Cloud-Farm zu erstellen, die eine Warteschlange zur Paketerstellung enthält. Die Vorlage konfiguriert eine Produktionswarteschlange und eine Warteschlange zur Paketerstellung mit einem privaten Amazon S3 S3-Conda-Kanal.

Bevor Sie die Vorlage bereitstellen, erstellen Sie einen Amazon S3 S3-Bucket für Jobanhänge und Ihren Conda-Kanal. Sie können einen Bucket von der Amazon S3 S3-Konsole aus erstellen. Sie benötigen den Bucket-Namen, wenn Sie die Vorlage bereitstellen.

Um die CloudFormation Vorlage bereitzustellen

1. Laden Sie die Vorlage deadline-cloud-starter-farm-template.yaml aus dem Deadline Cloud-Beispiel-Repository unter herunter. GitHub
2. Wählen Sie in der CloudFormation Konsole „Stack erstellen“ und dann „Mit neuen Ressourcen“ (Standard) aus.
3. Wählen Sie die Option zum Hochladen einer Vorlagendatei und laden Sie dann die `deadline-cloud-starter-farm-template.yaml` Datei hoch.
4. Geben Sie einen Namen für den Stack ein, z. B.**`StarterFarm`**, und geben Sie den Namen eines Amazon S3 S3-Buckets für Jobanhänge und den Conda-Kanal an.
5. Folgen Sie den Schritten auf der CloudFormation Konsole, um die Stack-Erstellung abzuschließen.

Weitere Informationen zu den Vorlagenparametern und Anpassungsoptionen finden Sie in der README-Datei für die Starter-Farm im Deadline-Cloud-Beispiel-Repository unterGitHub.

Erstellen Sie von der Konsole aus eine Warteschlange zur Paketerstellung

Folgen Sie den Anweisungen unter Eine Warteschlange erstellen im Deadline Cloud-Benutzerhandbuch. Nehmen Sie die folgenden Änderungen vor:

- Wählen Sie in Schritt 5 einen vorhandenen Amazon S3 S3-Bucket aus. Geben Sie einen Namen für den Stammordner an, z. B. **`DeadlineCloudPackageBuild`** damit Build-Artefakte von Ihren normalen Deadline Cloud-Anhängen getrennt bleiben.
- In Schritt 6 können Sie die Warteschlange zur Paketerstellung einer vorhandenen Flotte zuordnen oder eine völlig neue Flotte erstellen, falls Ihre aktuelle Flotte nicht geeignet ist.
- Erstellen Sie in Schritt 9 eine neue Servicerolle für Ihre Paketbau-Warteschlange. Sie werden die Berechtigungen ändern, um der Warteschlange die Berechtigungen zu geben, die für das Hochladen von Paketen und die Neuindizierung eines Conda-Kanals erforderlich sind.

Konfigurieren Sie die Berechtigungen für die Warteschlange zum Erstellen von Paketen

Damit die Warteschlange zur Paketerstellung auf das `/Conda` Präfix im Amazon S3 S3-Bucket der Warteschlange zugreifen kann, müssen Sie die Rolle der Warteschlange ändern, um ihr read/write Zugriff zu gewähren. Die Rolle benötigt die folgenden Berechtigungen, damit Paketerstellungsaufträge neue Pakete hochladen und den Channel neu indizieren können.

- `s3:GetObject`
- `s3:PutObject`
- `s3:ListBucket`
- `s3:GetBucketLocation`
- `s3:DeleteObject`

1. Öffnen Sie die Deadline Cloud-Konsole und navigieren Sie zur Seite mit den Warteschlangendetails für die Warteschlange zur Paketerstellung.
2. Wählen Sie die Warteschlangendienst-Rolle und anschließend Warteschlange bearbeiten aus.
3. Scrollen Sie zum Abschnitt Warteschlangendienstrolle und wählen Sie dann Diese Rolle in der IAM-Konsole anzeigen aus.
4. Wählen Sie aus der Liste der Berechtigungsrichtlinien die AmazonDeadlineCloudQueuePolicyfür Ihre Warteschlange aus.
5. Wählen Sie auf der Registerkarte „Berechtigungen“ die Option Bearbeiten aus.
6. Fügen Sie der Warteschlangendienst-Rolle einen neuen Abschnitt wie den folgenden hinzu. Ersetzen Sie *`amzn-s3-demo-bucket`* und *`111122223333`* durch Ihren eigenen Bucket und Ihr eigenes Konto.

```
{
   "Effect": "Allow",
   "Sid": "CustomCondaChannelReadWrite",
   "Action": [
    "s3:GetObject",
    "s3:PutObject",
    "s3:DeleteObject",
    "s3:ListBucket",
    "s3:GetBucketLocation"
   ],
   "Resource": [
    "arn:aws:s3:::amzn-s3-demo-bucket",
    "arn:aws:s3:::amzn-s3-demo-bucket/Conda/*"
   ],
   "Condition": {
    "StringEquals": {
     "aws:ResourceAccount": "111122223333"
    }
   }
  },
```

Reichen Sie einen Auftrag zur Paketerstellung ein

Nachdem Sie eine Warteschlange zum Erstellen von Paketen erstellt und die Warteschlangenberechtigungen konfiguriert haben, können Sie Aufträge zum Erstellen von Conda-Paketen einreichen. Das `submit-package-job` Skript im Deadline Cloud-Beispiel-Repository unter GitHub sendet einen Build-Job für ein Conda-Rezept.

Sie benötigen Folgendes:

- Die Deadline Cloud-CLI ist auf Ihrer Workstation installiert.
- Eine aktive AWS Anmeldesitzung für den Deadline Cloud-Monitor (Deadline Cloud-Monitor).
- Ein Klon des Deadline Cloud-Beispiel-Repositorys.

Um einen Auftrag zur Paketerstellung einzureichen

1. Öffnen Sie die Deadline Cloud-Konfigurations-GUI und legen Sie die Standardfarm und -warteschlange auf Ihre Paketerstellungswarteschlange fest.

```
deadline config gui
```

2. Wechseln Sie in das `conda_recipes` Verzeichnis im Beispiel-Repository.

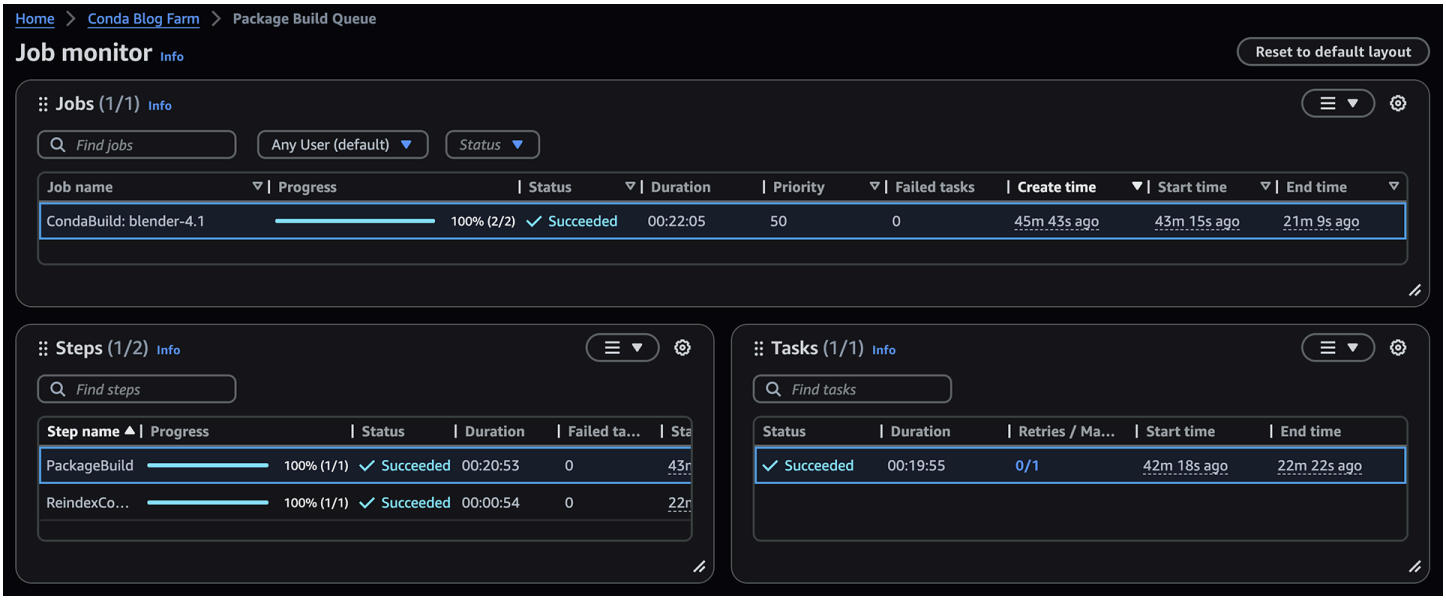
```
cd deadline-cloud-samples/conda_recipes
```

3. Führen Sie das `submit-package-job` Skript mit dem Rezeptverzeichnis aus. Das folgende Beispiel erstellt das Blender 4.5-Rezept.

```
./submit-package-job blender-4.5/
```

   Wenn für das Rezept ein Quellarchiv erforderlich ist, das Sie noch nicht heruntergeladen haben, enthält das Skript Anweisungen zum Herunterladen. Laden Sie das Archiv herunter und führen Sie das Skript erneut aus.

Nachdem Sie den Job eingereicht haben, können Sie den Deadline Cloud-Monitor verwenden, um den Fortschritt und Status des Jobs zu überprüfen.



Der Monitor zeigt die beiden Schritte des Jobs: das Erstellen des Pakets und die anschließende Neuindizierung des Conda-Kanals. Wenn Sie mit der rechten Maustaste auf die Aufgabe für den Schritt zur Paketerstellung klicken und Protokolle anzeigen wählen, zeigt der Monitor die Sitzungsaktionen an:

- Anlagen synchronisieren — Kopiert die Anlagen des Eingabeauftrags oder hängt ein virtuelles Dateisystem ein.
- Starten Sie Conda — Die Aktion für die Warteschlangenumgebung. Der Build-Job spezifiziert keine Conda-Pakete, sodass diese Aktion schnell abgeschlossen wird.
- Launch CondaBuild Env — Erstellt eine virtuelle Conda-Umgebung mit der Software, die zum Erstellen eines Conda-Pakets und zur Neuindizierung eines Kanals erforderlich ist.
- Task run — Erstellt das Paket und lädt die Ergebnisse auf Amazon S3 hoch.

Während die Aktionen ausgeführt werden, senden sie Protokolle an Amazon CloudWatch (CloudWatch). Wenn ein Job abgeschlossen ist, wählen Sie Protokolle für alle Aufgaben anzeigen aus, um zusätzliche Protokolle über die Einrichtung und den Abbau der Umgebung einzusehen.

Führen Sie Host-Konfigurationsskripten mit Administratorrechten aus

Mithilfe von Host-Konfigurationsskripten können Sie administrative Aufgaben, wie z. B. die Softwareinstallation, für Ihre vom Service verwalteten Flottenmitarbeiter ausführen. Diese Skripts werden mit erhöhten Rechten (`sudo`anLinux, Administrator aktiviertWindows) ausgeführt, sodass Sie Ihre Worker flexibel für Ihr System konfigurieren können.

Deadline Cloud führt das Skript aus, nachdem der Worker den `STARTING` Status erreicht hat und bevor er irgendwelche Aufgaben ausführt.

> **Important**
>
> Das Skript wird mit erhöhten Berechtigungen ausgeführt. Es liegt in Ihrer Verantwortung sicherzustellen, dass das Skript keine Sicherheitsprobleme verursacht.
> Wenn Sie ein Host-Konfigurationsskript verwenden, sind Sie dafür verantwortlich, den Zustand Ihrer Flotte zu überwachen.

Zu den häufigsten Verwendungszwecken für Host-Konfigurationsskripten gehören:

- Installation von Software, für die Administratorzugriff erforderlich ist
- Installation von Docker Containern

- Installation von Cloud-Speicherlösungen von Drittanbietern wieLucidLink. Eine exemplarische Vorgehensweise finden Sie im for LucidLink M&E-Blog unter Einrichtung von servicemanaged-Flottenskripten AWS für Deadline Cloud.

Sie können ein Host-Konfigurationsskript mit der Konsole oder mit dem erstellen und aktualisieren. AWS CLI

Console

1. Wählen Sie auf der Seite mit den Flottendetails die Registerkarte Konfigurationen aus.
2. Geben Sie im Feld Skript das Skript ein, das mit erhöhten Rechten ausgeführt werden soll. Sie können Import wählen, um ein Skript von Ihrer Workstation zu laden.
3. Legen Sie eine Zeitüberschreitung in Sekunden für die Ausführung des Skripts fest. Der Standardwert ist 300 Sekunden (5 Minuten).
4. Wählen Sie Änderungen speichern, um das Skript zu speichern.

Create with CLI

Verwenden Sie den folgenden AWS CLI Befehl, um eine Flotte mit einem Host-Konfigurationsskript zu erstellen. Ersetzen Sie den *placeholder* Text durch Ihre Informationen.

```
aws deadline create-fleet \
--farm-id farm-12345 \
--display-name "fleet-name" \
--max-worker-count 1 \
--configuration '{
"serviceManagedEc2": {
  "instanceCapabilities": {
    "vCpuCount": {"min": 2},
    "memoryMiB": {"min": 4096},
    "osFamily": "linux",
    "cpuArchitectureType": "x86_64"
  },
  "instanceMarketOptions": {"type":"spot"}
}
}' \
--role-arn arn:aws:iam::111122223333:role/role-name \
--host-configuration '{ "scriptBody": "script body", "scriptTimeoutSeconds": timeout
 value}'
```

Update with CLI

Verwenden Sie den folgenden AWS CLI Befehl, um das Host-Konfigurationsskript einer Flotte zu aktualisieren. Ersetzen Sie den *placeholder* Text durch Ihre Informationen.

```
aws deadline update-fleet \
--farm-id farm-12345 \
--fleet-id fleet-455678 \
--host-configuration '{ "scriptBody": "script body", "scriptTimeoutSeconds": timeout
 value}'
```

Die folgenden Skripte demonstrieren:

- Die für das Skript verfügbaren Umgebungsvariablen
- Diese AWS Anmeldeinformationen funktionieren in der Shell
- Dass das Skript in einer Shell mit erhöhten Rechten ausgeführt wird

Linux

Verwenden Sie das folgende Skript, um zu zeigen, dass ein Skript mit `root` Rechten ausgeführt wird:

```
# Print environment variables
set
# Check AWS Credentials
aws sts get-caller-identity
```

Windows

Verwenden Sie das folgende PowerShell Skript, um zu zeigen, dass ein Skript mit Administratorrechten ausgeführt wird:

```
Get-ChildItem env: | ForEach-Object { "$($_.Name)=$($_.Value)" }
aws sts get-caller-identity
function Test-AdminPrivileges {
  $currentUser = New-Object
 Security.Principal.WindowsPrincipal([Security.Principal.WindowsIdentity]::GetCurrent())
  $isAdmin =
 $currentUser.IsInRole([Security.Principal.WindowsBuiltInRole]::Administrator)
```

```
  return $isAdmin
}

if (Test-AdminPrivileges) {
  Write-Host "The current PowerShell session is elevated (running as
 Administrator)."
} else {
  Write-Host "The current PowerShell session is not elevated (not running as
 Administrator)."
}
exit 0
```

Problembehandlung bei Host-Konfigurationsskripten

Wenn Sie das Host-Konfigurationsskript ausführen:

- Bei Erfolg: Der Worker führt den Job aus
- Bei einem Fehler (Exit-Code ungleich Null oder Absturz):
  - Der Worker wird heruntergefahren

  Die Flotte startet automatisch einen neuen Worker unter Verwendung des neuesten Host-Konfigurationsskripts

Um das Skript zu überwachen:

1. Öffnen Sie die Flottenseite in der Deadline Cloud-Konsole.
2. Wählen Sie Mitarbeiter anzeigen, um den Deadline Cloud-Monitor zu öffnen.
3. Sehen Sie sich den Mitarbeiterstatus auf der Monitorseite an.

> **Tip**
>
> Legen Sie beim Testen von Host-Konfigurationsskripten die maximale Anzahl von Workern für die Flotte auf 1 fest, um zu vermeiden, dass mehrere Worker während der Iteration des Skripts gestartet werden.

Wichtige Hinweise:

- Worker, die aufgrund eines Fehlers heruntergefahren wurden, sind in der Worker-Liste im Monitor nicht verfügbar. Verwenden Sie CloudWatch Logs, um die Worker-Logs in der folgenden Protokollgruppe einzusehen:

  ```
  /aws/deadline/farm-XXXXX/fleet-YYYYY
  ```

  Suchen Sie in dieser Protokollgruppe nach einem Stream mit dem Namen`worker-ZZZZZ`.
- CloudWatch Logs speichert Worker-Protokolle entsprechend der von Ihnen konfigurierten Aufbewahrungsfrist.

Überwachen Sie die Ausführung des Hostkonfigurationsskripts

Mit Host-Konfigurationsskripten können Sie die volle Kontrolle über einen Deadline Cloud-Worker übernehmen. Sie können jedes Softwarepaket installieren, Betriebssystemparameter neu konfigurieren oder gemeinsam genutzte Dateisysteme mounten. Mit dieser erweiterten Funktion und der Fähigkeit von Deadline Cloud, auf Tausende von Workern zu skalieren, können Sie überwachen, wann Konfigurationsskripte erfolgreich ausgeführt wurden oder nicht.

Wir empfehlen die folgenden Lösungen für die Überwachung der Ausführung von Host-Konfigurationsskripten.

CloudWatch Überwachung von Protokollen

Alle Flottenhost-Konfigurationsprotokolle werden in die CloudWatch Protokollgruppe der Flotte und speziell in den CloudWatch Log-Stream eines Mitarbeiters gestreamt. Dies `/aws/deadline/farm-123456789012/fleet-777788889999` ist beispielsweise die Protokollgruppe für Farm`123456789012`, Flotte`777788889999`.

Jeder Mitarbeiter stellt beispielsweise einen eigenen Protokollstream bereit`worker-123456789012`. Die Hostkonfigurationsprotokolle enthalten Log-Banner wie „Host-Konfigurationsskript wird ausgeführt“ und „Host-Konfigurationsskript wurde abgeschlossen“, Exit-Code: 0. Der Exit-Code des Skripts ist im fertigen Banner enthalten und kann mithilfe CloudWatch von Tools abgefragt werden.

CloudWatch Protokolliert und Einblicke

CloudWatch Logs Insights bietet erweiterte Funktionen zur Analyse von Protokollinformationen. Die folgende Log Insights-Abfrage sucht beispielsweise nach dem Exit-Code der Hostkonfiguration, sortiert nach Zeit:

```
fields @timestamp, @message, @logStream, @log
| filter @message like /Finished running Host Configuration Script/
| parse @message /exit code: (?<exit_code>\d+)/
| display @timestamp, exit_code
| sort @timestamp desc
```

Weitere Informationen zu CloudWatch Logs Insights finden Sie unter Analysieren von Protokolldaten mit CloudWatch Logs Insights im Amazon CloudWatch Logs-Benutzerhandbuch.

Strukturierte Protokollierung durch Worker Agent

Der Worker Agent von Deadline Cloud veröffentlicht strukturierte JSON-Protokolle für CloudWatch. Der Worker Agent bietet eine breite Palette strukturierter Protokolle zur Analyse der Gesundheit von Arbeitnehmern. Weitere Informationen finden Sie unter Anmeldung des Deadline Cloud-Worker-Agents GitHub.

Die Attribute der strukturierten Protokolle werden in Felder in Log Insights entpackt. Sie können diese CloudWatch Funktion verwenden, um Fehler beim Starten der Hostkonfiguration zu zählen und zu analysieren. Mit einer Count-and-Bin-Abfrage kann beispielsweise ermittelt werden, wie oft Fehler auftreten:

```
fields @timestamp, @message, @logStream, @log
| sort @timestamp desc
| filter message like /Worker Agent host configuration failed with exit code/
| stats count(*) by exit_code, bin(1h)
```

CloudWatch metrische Filter für Metriken und Alarme

Sie können CloudWatch Metrikfilter einrichten, um CloudWatch Metriken aus Protokollen zu generieren. Mit Metrikfiltern können Sie Alarme und Dashboards zur Überwachung der Ausführung von Host-Konfigurationsskripten erstellen.

So erstellen Sie einen Metrikfilter

1. Öffnen Sie die CloudWatch Konsole.
2. Wählen Sie im Navigationsbereich Protokolle und dann Protokollgruppen aus.
3. Wählen Sie die Protokollgruppe Ihrer Flotte aus.
4. Wählen Sie Metrikfilter erstellen aus.
5. Definieren Sie Ihr Filtermuster mit einer der folgenden Methoden:

- Für Erfolgskennzahlen:

```
{$.message = "*Worker Agent host configuration succeeded.*"}
```

- Für Misserfolgsmetriken:

```
{$.exit_code != 0 && $.message = "*Worker Agent host configuration failed with
 exit code*"}
```

6. Wählen Sie Weiter, um eine Metrik mit den folgenden Werten zu erstellen:

- Metrik-Namespace: Ihr Metrik-Namespace (zum Beispiel) **`MyDeadlineFarm`**
- Metrikname: Ihr angeforderter Metrikname (z. B.) **`host_config_failure`**
- Metrikwert: **`1`** (Jede Instanz entspricht einer Zählung von 1)
- Standardwert: Leer lassen
- Einheit: **`Count`**

Nachdem Sie Metrikfilter erstellt haben, können Sie CloudWatch Standardalarme konfigurieren, um bei erhöhten Ausfallraten bei der Hostkonfiguration Maßnahmen zu ergreifen, oder die Metriken zu einem CloudWatch Dashboard für day-to-day Betrieb und Überwachung hinzufügen.

Weitere Informationen finden Sie unter Filter- und Mustersyntax im Amazon CloudWatch Logs-Benutzerhandbuch.

Verwendung von Softwarelizenzen mit Deadline Cloud

Deadline Cloud bietet zwei Methoden zur Bereitstellung von Softwarelizenzen für Ihre Jobs:

- Nutzungsbasierte Lizenzierung (UBL) — Nachverfolgung und Abrechnung basieren auf der Anzahl der Stunden, die Ihre Flotte für die Bearbeitung eines Auftrags benötigt. Es gibt keine festgelegte Anzahl an Lizenzen, sodass Ihre Flotte nach Bedarf skaliert werden kann. UBL ist Standard für servicemanagierte Flotten. Für vom Kunden verwaltete Flotten können Sie einen Deadline Cloud-Lizenzendpunkt für UBL verbinden. UBL stellt Lizenzen für Ihre Deadline Cloud-Mitarbeiter zum Rendern bereit, stellt jedoch keine Lizenzen für Ihre DCC-Anwendungen bereit.
- Bring Your Own License (BYOL) — ermöglicht es Ihnen, bestehende Softwarelizenzen mit Ihren vom Service oder vom Kunden verwalteten Flotten zu verwenden. Sie können BYOL verwenden, um eine Verbindung zu Lizenzservern für Software herzustellen, die nicht von den nutzungsbasierten Lizenzen von Deadline Cloud unterstützt wird. Sie können BYOL mit vom Service verwalteten Flotten verwenden, indem Sie eine Verbindung zu einem benutzerdefinierten Lizenzserver herstellen.

Themen

Connect vom Service verwaltete Flotten mit einem benutzerdefinierten Lizenzserver

Sie können Ihren eigenen Lizenzserver verwenden, um ihn mit einer vom Service verwalteten Deadline Cloud-Flotte zu verwenden. Um Ihre eigene Lizenz mitzubringen, können Sie einen Lizenzserver mithilfe einer Warteschlangenumgebung in Ihrer Farm konfigurieren. Um Ihren Lizenzserver zu konfigurieren, sollten Sie bereits eine Farm und eine Warteschlange eingerichtet haben.

Wie Sie eine Verbindung zu einem Softwarelizenzserver herstellen, hängt von der Konfiguration Ihrer Flotte und den Anforderungen des Softwareanbieters ab. In der Regel greifen Sie auf zwei Arten auf den Server zu:

- Direkt zum Lizenzserver. Ihre Mitarbeiter erhalten über das Internet eine Lizenz vom Lizenzserver des Softwareanbieters. Alle Ihre Mitarbeiter müssen in der Lage sein, eine Verbindung zum Server herzustellen.
- Über einen Lizenz-Proxy. Ihre Mitarbeiter stellen eine Verbindung zu einem Proxyserver in Ihrem lokalen Netzwerk her. Nur der Proxyserver darf über das Internet eine Verbindung zum Lizenzserver des Anbieters herstellen.

Mit den folgenden Anweisungen verwenden Sie Amazon EC2 Systems Manager (SSM), um Ports von einer Worker-Instance an Ihren Lizenzserver oder Ihre Proxy-Instance weiterzuleiten. Wenn Ihr Lizenzserver im folgenden Beispiel keine Lizenz bereitstellen kann, wird die nutzungsbasierte Lizenzierung von Deadline Cloud verwendet. Entfernen Sie die Abschnitte, die nicht für Ihre Pipeline oder Produkte gelten, für die Sie nach Ablauf Ihrer Lizenzen keine nutzungsbasierte Lizenzierung verwenden möchten.

Topics

Schritt 1: Konfigurieren Sie die Warteschlangenumgebung

Sie können in Ihrer Warteschlange eine Warteschlangenumgebung für den Zugriff auf Ihren Lizenzserver konfigurieren. Stellen Sie zunächst sicher, dass Sie eine AWS Instanz mit Lizenzserverzugriff konfiguriert haben. Verwenden Sie dazu eine der folgenden Methoden:

- Lizenzserver — Die Instanz hostet die Lizenzserver direkt.
- Lizenzproxy — Die Instanz hat Netzwerkzugriff auf den Lizenzserver und leitet die Lizenzserverports an den Lizenzserver weiter. Einzelheiten zur Konfiguration einer Lizenz-Proxyinstanz finden Sie unterSchritt 2: (Optional) Einrichtung der Lizenz-Proxyinstanz.

Hinweise zur Konfiguration von Lizenzumgebungsvariablen finden Sie unterSchritt 3: Eine Rendering-Anwendung mit einem Endpunkt Connect. Für ein benutzerdefiniertes Lizenzserver-Setup bleibt die Lizenzserveradresse localhost und nicht der Amazon VPC-Endpunkt.

Um der Warteschlangenrolle die erforderlichen Berechtigungen hinzuzufügen

1. Wählen Sie in der Deadline Cloud-Konsole die Option Gehe zum Dashboard aus.
2. Wählen Sie im Dashboard die Farm und dann die Warteschlange aus, die Sie konfigurieren möchten.
3. Wählen Sie unter Warteschlangendetails > Servicerolle die Rolle aus.
4. Wählen Sie „Berechtigung hinzufügen“ und anschließend „Inline-Richtlinie erstellen“ aus.
5. Wählen Sie den JSON-Richtlinieneditor aus, kopieren Sie dann den folgenden Text und fügen Sie ihn in den Editor ein.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Action": [
                "ssm:StartSession"
            ],
            "Resource": [
                "arn:aws:ssm:us-east-1::document/AWS-
StartPortForwardingSession",
                "arn:aws:ec2:us-east-1:111122223333:instance/instance_id"
            ]
        }
    ]
}
```

6. Bevor Sie die neue Richtlinie speichern, ersetzen Sie die folgenden Werte im Richtlinientext:
   - `region`Ersetzen Sie durch die AWS Region, in der sich Ihre Farm befindet
   - `instance_id`Ersetzen Sie es durch die Instanz-ID für den Lizenzserver oder die Proxyinstanz, die Sie verwenden
   - `account_id`Ersetzen Sie es durch die AWS Kontonummer, die Ihre Farm enthält
7. Wählen Sie Weiter aus.
8. Geben Sie als Richtliniennamen ein**`LicenseForwarding`**.

9. Wählen Sie Richtlinie erstellen aus, um Ihre Änderungen zu speichern und die Richtlinie mit den erforderlichen Berechtigungen zu erstellen.

Um der Warteschlange eine neue Warteschlangenumgebung hinzuzufügen

1. Wählen Sie in der Deadline Cloud-Konsole Gehe zum Dashboard, falls Sie dies noch nicht getan haben.
2. Wählen Sie im Dashboard die Farm und dann die Warteschlange aus, die Sie konfigurieren möchten.
3. Wählen Sie „Warteschlangenumgebungen“ **>** „Aktionen“ **>** „Mit YAML neu erstellen“.
4. Kopieren Sie den folgenden Text und fügen Sie ihn in den YAML-Skripteditor ein.

   Windows

```
specificationVersion: "environment-2023-09"
parameterDefinitions:
 - name: LicenseInstanceId
   type: STRING
   description: >
    The Instance ID of the license server/proxy instance
   default: ""
 - name: LicenseInstanceRegion
   type: STRING
   description: >
    The region containing this farm
   default: ""
 - name: LicensePorts
   type: STRING
   description: >
    Comma-separated list of ports to be forwarded to the license server/proxy
    instance. Example: "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
   default: "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
environment:
 name: BYOL License Forwarding
 variables:
  example_LICENSE: 2701@localhost
 script:
  actions:
   onEnter:
    command: bash
```

```
    args: [ "{{Env.File.Enter}}" ]
   onExit:
    command: bash
    args: [ "{{Env.File.Exit}}" ]
  embeddedFiles:
   - name: Enter
     type: TEXT
     runnable: True
     data: |
      curl "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/
windows/SessionManagerPlugin.zip" -o "{{Session.WorkingDirectory}}/ssm-
plugin.zip"
      powershell -Command "Expand-Archive -Path '{{Session.WorkingDirectory}}/
ssm-plugin.zip' -DestinationPath '{{Session.WorkingDirectory}}/ssm-plugin'
 -Force; Expand-Archive -Path '{{Session.WorkingDirectory}}/ssm-plugin/
package.zip' -DestinationPath '{{Session.WorkingDirectory}}/ssm-plugin/package'
 -Force"
      conda activate
      python "{{Env.File.StartSession}}" "{{Session.WorkingDirectory}}/ssm-
plugin/package/bin/session-manager-plugin.exe"
   - name: Exit
     type: TEXT
     runnable: True
     data: |
      echo Killing SSM Manager Plugin PIDs: $BYOL_SSM_PIDS
      for pid in ${BYOL_SSM_PIDS//,/ }; do kill $pid; done
   - name: StartSession
     type: TEXT
     data: |
      import boto3
      import json
      import subprocess
      import sys
      import os
      import tempfile

      instance_id = "{{Param.LicenseInstanceId}}"
      region = "{{Param.LicenseInstanceRegion}}"
      license_ports_list = "{{Param.LicensePorts}}".split(",")

      ssm_client = boto3.client("ssm", region_name=region)
      pids = []

      for port in license_ports_list:
```

```
        session_response = ssm_client.start_session(
          Target=instance_id,
          DocumentName="AWS-StartPortForwardingSession",
          Parameters={"portNumber": [port], "localPortNumber": [port]}
        )

        cmd = [
          sys.argv[1],
          json.dumps(session_response),
          region,
          "StartSession",
          "",
          json.dumps({"Target": instance_id}),
          f"https://ssm.{region}.amazonaws.com"
        ]

        process = subprocess.Popen(cmd, stdout=subprocess.DEVNULL,
 stderr=subprocess.DEVNULL)
        pids.append(process.pid)
        print(f"SSM Port Forwarding Session started for port {port}")

      print(f"openjd_env: BYOL_SSM_PIDS={','.join(str(pid) for pid in pids)}")

      # Enabling UBL after the BYOL has run out requires prepending the BYOL
 configuration to the existing license setup
      # Remove the sections that do not apply to your pipeline, or you do not
 want to use UBL after exhausting the BYOL licenses.
      # The port numbers used may not match what your license server is serving.

      # Arnold
      os.environ["ADSKFLEX_LICENSE_FILE"] = f"2701@localhost;
{os.environ.get('ADSKFLEX_LICENSE_FILE', '')}"
      print(f"openjd_env:
 ADSKFLEX_LICENSE_FILE={os.environ['ADSKFLEX_LICENSE_FILE']}")

      # Cinema4D
      os.environ["g_licenseServerRLM"] = f"localhost:7057;
{os.environ.get('g_licenseServerRLM', '')}"
      print(f"openjd_env:
 g_licenseServerRLM={os.environ['g_licenseServerRLM']}")

      # Nuke
      os.environ["foundry_LICENSE"] = f"6101@localhost;
{os.environ.get('foundry_LICENSE', '')}"
```

```
      print(f"openjd_env: foundry_LICENSE={os.environ['foundry_LICENSE']}")

      # SideFX
      os.environ["SESI_LMHOST"] = f"localhost:1715;
{os.environ.get('SESI_LMHOST', '')}"
      print(f"openjd_env: SESI_LMHOST={os.environ['SESI_LMHOST']}")

      # Redshift and Red Giant
      os.environ["redshift_LICENSE"] = f"7054@localhost;7055@localhost;
{os.environ.get('redshift_LICENSE', '')}"
      print(f"openjd_env: redshift_LICENSE={os.environ['redshift_LICENSE']}")

      # V-Ray doesn't support multiple license servers in a single environment
 variable
      # See https://documentation.chaos.com/space/LIC5/125050770/Sharing+a
+License+Configuration+in+a+Network
      vray_license = os.environ.get('VRAY_AUTH_CLIENT_SETTINGS', '')
      xml_content = """<VRLClient>
        <LicServer>
          <Host>localhost</Host>
          <Port>30304</Port>"""

      if vray_license and vray_license.startswith('licset://'):
          server_parts = vray_license.removeprefix('licset://').split(':')
          if len(server_parts) >= 2:
              xml_content += f"""
          <Host1>{server_parts[0]}</Host1>
          <Port1>{server_parts[1]}</Port1>"""

      xml_content += """
          <User></User>
          <Pass></Pass>
        </LicServer>
      </VRLClient>"""

      temp_dir = tempfile.gettempdir()
      xml_path = os.path.join(temp_dir, 'vrlclient.xml')

      with open(xml_path, 'w') as f:
          f.write(xml_content)

      os.environ["VRAY_AUTH_CLIENT_FILE_PATH"] = temp_dir
      print(f"openjd_env:
 VRAY_AUTH_CLIENT_FILE_PATH={os.environ['VRAY_AUTH_CLIENT_FILE_PATH']}")
```

```
     # Clear the existing VRAY_AUTH_CLIENT_SETTINGS so only the vrlclient.xml
 file is used.
     os.environ["VRAY_AUTH_CLIENT_SETTINGS"] = ''
     print(f"openjd_env:
 VRAY_AUTH_CLIENT_SETTINGS={os.environ['VRAY_AUTH_CLIENT_SETTINGS']}")

     # Print out the created xml file's contents
     print(f"V-Ray configuration file: {xml_path}")
     with open(xml_path, 'r') as f:
         print(f"{f.read()}")
```

Linux

```
specificationVersion: "environment-2023-09"
parameterDefinitions:
 - name: LicenseInstanceId
   type: STRING
   description: >
    The Instance ID of the license server/proxy instance
   default: ""
 - name: LicenseInstanceRegion
   type: STRING
   description: >
    The region containing this farm
   default: ""
 - name: LicensePorts
   type: STRING
   description: >
    Comma-separated list of ports to be forwarded to the license server/proxy
    instance. Example: "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
   default: "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
environment:
 name: BYOL License Forwarding
 variables:
  example_LICENSE: 2701@localhost
 script:
  actions:
   onEnter:
    command: bash
    args: [ "{{Env.File.Enter}}" ]
```

```
   onExit:
    command: bash
    args: [ "{{Env.File.Exit}}" ]
  embeddedFiles:
   - name: Enter
     type: TEXT
     runnable: True
     data: |
      curl https://s3.amazonaws.com/session-manager-downloads/plugin/
latest/linux_64bit/session-manager-plugin.rpm -Ls | rpm2cpio - | cpio -iv
 --to-stdout ./usr/local/sessionmanagerplugin/bin/session-manager-plugin >
 {{Session.WorkingDirectory}}/session-manager-plugin
      chmod +x {{Session.WorkingDirectory}}/session-manager-plugin
      conda activate
      python {{Env.File.StartSession}} {{Session.WorkingDirectory}}/session-
manager-plugin
   - name: Exit
     type: TEXT
     runnable: True
     data: |
      echo Killing SSM Manager Plugin PIDs: $BYOL_SSM_PIDS
      for pid in ${BYOL_SSM_PIDS//,/ }; do kill $pid; done
   - name: StartSession
     type: TEXT
     data: |
      import boto3
      import json
      import subprocess
      import sys
      import os
      import tempfile

      instance_id = "{{Param.LicenseInstanceId}}"
      region = "{{Param.LicenseInstanceRegion}}"
      license_ports_list = "{{Param.LicensePorts}}".split(",")

      ssm_client = boto3.client("ssm", region_name=region)
      pids = []

      for port in license_ports_list:
        session_response = ssm_client.start_session(
          Target=instance_id,
          DocumentName="AWS-StartPortForwardingSession",
          Parameters={"portNumber": [port], "localPortNumber": [port]}
```

```
        )

        cmd = [
          sys.argv[1],
          json.dumps(session_response),
          region,
          "StartSession",
          "",
          json.dumps({"Target": instance_id}),
          f"https://ssm.{region}.amazonaws.com"
        ]

        process = subprocess.Popen(cmd, stdout=subprocess.DEVNULL,
 stderr=subprocess.DEVNULL)
        pids.append(process.pid)
        print(f"SSM Port Forwarding Session started for port {port}")

      print(f"openjd_env: BYOL_SSM_PIDS={','.join(str(pid) for pid in pids)}")

      # Enabling UBL after the BYOL has run out requires prepending the BYOL
 configuration to the existing license setup
      # Remove the sections that do not apply to your pipeline, or you do not
 want to use UBL after exhausting the BYOL licenses.
      # The port numbers used may not match what your license server is serving.

      # Arnold
      os.environ["ADSKFLEX_LICENSE_FILE"] = f"2701@localhost:
{os.environ.get('ADSKFLEX_LICENSE_FILE', '')}"
      print(f"openjd_env:
 ADSKFLEX_LICENSE_FILE={os.environ['ADSKFLEX_LICENSE_FILE']}")

      # Nuke
      os.environ["foundry_LICENSE"] = f"6101@localhost:
{os.environ.get('foundry_LICENSE', '')}"
      print(f"openjd_env: foundry_LICENSE={os.environ['foundry_LICENSE']}")

      # SideFX
      os.environ["SESI_LMHOST"] = f"localhost:1715;
{os.environ.get('SESI_LMHOST', '')}"
      print(f"openjd_env: SESI_LMHOST={os.environ['SESI_LMHOST']}")

      # Redshift and Red Giant
      os.environ["redshift_LICENSE"] = f"7054@localhost:7055@localhost:
{os.environ.get('redshift_LICENSE', '')}"
```

```
      print(f"openjd_env: redshift_LICENSE={os.environ['redshift_LICENSE']}")

      # V-Ray doesn't support multiple license servers in a single environment
 variable
      # See https://documentation.chaos.com/space/LIC5/125050770/Sharing+a
+License+Configuration+in+a+Network
      vray_license = os.environ.get('VRAY_AUTH_CLIENT_SETTINGS', '')
      xml_content = """<VRLClient>
        <LicServer>
          <Host>localhost</Host>
          <Port>30304</Port>"""

      if vray_license and vray_license.startswith('licset://'):
          server_parts = vray_license.removeprefix('licset://').split(':')
          if len(server_parts) >= 2:
              xml_content += f"""
          <Host1>{server_parts[0]}</Host1>
          <Port1>{server_parts[1]}</Port1>"""

      xml_content += """
          <User></User>
          <Pass></Pass>
        </LicServer>
      </VRLClient>"""

      temp_dir = tempfile.gettempdir()
      xml_path = os.path.join(temp_dir, 'vrlclient.xml')

      with open(xml_path, 'w') as f:
          f.write(xml_content)

      os.environ["VRAY_AUTH_CLIENT_FILE_PATH"] = temp_dir
      print(f"openjd_env:
 VRAY_AUTH_CLIENT_FILE_PATH={os.environ['VRAY_AUTH_CLIENT_FILE_PATH']}")

      # Clear the existing VRAY_AUTH_CLIENT_SETTINGS so only the vrlclient.xml
 file is used.
      os.environ["VRAY_AUTH_CLIENT_SETTINGS"] = ''
      print(f"openjd_env:
 VRAY_AUTH_CLIENT_SETTINGS={os.environ['VRAY_AUTH_CLIENT_SETTINGS']}")

      # Print out the created xml file's contents
      print(f"V-Ray configuration file: {xml_path}")
      with open(xml_path, 'r') as f:
```

```
            print(f"{f.read()}")
```

5. Bevor Sie die Warteschlangenumgebung speichern, nehmen Sie nach Bedarf die folgenden Änderungen am Umgebungstext vor:

   - Aktualisieren Sie die Standardwerte für die folgenden Parameter, sodass sie Ihrer Umgebung entsprechen:
     - LicenseInstanceID — Die Amazon EC2 EC2-Instance-ID Ihres Lizenzservers oder Ihrer Proxy-Instance
     - LicenseInstanceRegion— Die AWS Region, in der sich Ihre Farm befindet
     - LicensePorts— Eine durch Kommas getrennte Liste von Ports, die an den Lizenzserver oder die Proxyinstanz weitergeleitet werden sollen (z. B. 2700,2701)
   - Wenn Sie die nutzungsbasierte Lizenzierung (UBL) verwenden möchten, nachdem Bring Your Own License (BYOL) erschöpft ist, stellen Sie sicher, dass der Port für Ihren Lizenzserver korrekt ist. Wenn Sie UBL nicht mehr verwenden möchten, nachdem BYOL aufgebraucht ist, fügen Sie dem Variablenbereich alle erforderlichen Umgebungsvariablen für die Lizenzierung hinzu.

     Diese Variablen sollten den Link DCCs zu localhost auf dem Lizenzserverport weiterleiten. Wenn Ihr Foundry-Lizenzserver beispielsweise Port 6101 abhört, würden Sie die Variable als hinzufügen. **`foundry_LICENSE: 6101@localhost`**

6. (Optional) Sie können die Priorität auf 0 belassen oder sie so ändern, dass die Priorität in Umgebungen mit mehreren Warteschlangen unterschiedlich angeordnet wird.
7. Wählen Sie „Warteschlangenumgebung erstellen“, um die neue Umgebung zu speichern.

   Wenn die Warteschlangenumgebung festgelegt ist, rufen Aufträge, die an diese Warteschlange gesendet werden, Lizenzen vom konfigurierten Lizenzserver ab.

Schritt 2: (Optional) Einrichtung der Lizenz-Proxyinstanz

Als Alternative zur Verwendung eines Lizenzservers können Sie einen Lizenzproxy verwenden. Um einen Lizenz-Proxy zu erstellen, erstellen Sie eine neue Amazon Linux 2023-Instance, die Netzwerkzugriff auf den Lizenzserver hat. Bei Bedarf können Sie diesen Zugriff über eine VPN-Verbindung konfigurieren. Weitere Informationen finden Sie unter VPN-Verbindungen im Amazon VPC-Benutzerhandbuch.

Um eine Lizenz-Proxyinstanz für Deadline Cloud einzurichten, folgen Sie den Schritten in diesem Verfahren. Führen Sie die folgenden Konfigurationsschritte auf dieser neuen Instanz durch, um die Weiterleitung des Lizenzverkehrs an Ihren Lizenzserver zu ermöglichen

1. Geben Sie Folgendes ein, um das HAProxy Paket zu installieren

```
sudo yum install haproxy
```

2. Aktualisieren Sie den Abschnitt listen license-server der Konfigurationsdatei/etc/haproxy/haproxy.cfg wie folgt:

   a. Ersetzen Sie LicensePort1 und LicensePort2 durch die Portnummern, die an den Lizenzserver weitergeleitet werden sollen. Fügen Sie kommagetrennte Werte hinzu oder entfernen Sie sie, um der erforderlichen Anzahl von Anschlüssen gerecht zu werden.

   b. LicenseServerHostErsetzen Sie durch den Hostnamen oder die IP-Adresse des Lizenzservers.

```
lobal
    log         127.0.0.1 local2
    chroot      /var/lib/haproxy
    user        haproxy
    group       haproxy
    daemon

defaults
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s

listen license-server
     bind *:LicensePort1,*:LicensePort2
     server license-server LicenseServerHost
```

3. Führen Sie die folgenden Befehle aus, um den HAProxy Dienst zu aktivieren und zu starten:

```
sudo systemctl enable haproxy
sudo service haproxy start
```

Nach Abschluss der Schritte sollten Lizenzanfragen, die aus der Weiterleitungswarteschlangenumgebung an localhost gesendet werden, an den angegebenen Lizenzserver weitergeleitet werden.

Schritt 3: Einrichtung der CloudFormation Vorlage

Sie können eine CloudFormation Vorlage verwenden, um eine gesamte Farm so zu konfigurieren, dass sie Ihre eigene Lizenzierung verwendet.

1. Ändern Sie die im nächsten Schritt bereitgestellte Vorlage, um dem Variablenbereich unter Umgebung alle erforderlichen BYOLQueueUmgebungsvariablen für die Lizenzierung hinzuzufügen.
2. Verwenden Sie die folgende CloudFormation Vorlage.

```
AWSTemplateFormatVersion: 2010-09-09
Description: "Create &ADC; resources for BYOL"

Parameters:
  LicenseInstanceId:
    Type: AWS::EC2::Instance::Id
    Description: Instance ID for the license server/proxy instance
  LicensePorts:
    Type: String
    Description: Comma-separated list of ports to forward to the license instance

Resources:
  JobAttachmentBucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: !Sub byol-example-ja-bucket-${AWS::AccountId}-${AWS::Region}
      BucketEncryption:
        ServerSideEncryptionConfiguration:
          - ServerSideEncryptionByDefault:
              SSEAlgorithm: AES256

  Farm:
    Type: AWS::Deadline::Farm
    Properties:
      DisplayName: BYOLFarm

  QueuePolicy:
```

```
    Type: AWS::IAM::ManagedPolicy
    Properties:
      ManagedPolicyName: BYOLQueuePolicy
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action:
              - s3:GetObject
              - s3:PutObject
              - s3:ListBucket
              - s3:GetBucketLocation
            Resource:
              - !Sub ${JobAttachmentBucket.Arn}
              - !Sub ${JobAttachmentBucket.Arn}/job-attachments/*
            Condition:
              StringEquals:
                aws:ResourceAccount: !Sub ${AWS::AccountId}
          - Effect: Allow
            Action: logs:GetLogEvents
            Resource: !Sub arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-
group:/aws/deadline/${Farm.FarmId}/*
          - Effect: Allow
            Action:
              - s3:ListBucket
              - s3:GetObject
            Resource:
              - "*"
            Condition:
              ArnLike:
                s3:DataAccessPointArn:
                  - arn:aws:s3:*:*:accesspoint/deadline-software-*
              StringEquals:
                s3:AccessPointNetworkOrigin: VPC

  BYOLSSMPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      ManagedPolicyName: BYOLSSMPolicy
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action:
```

```
              - ssm:StartSession
            Resource:
              - !Sub arn:aws:ssm:${AWS::Region}::document/AWS-
StartPortForwardingSession
              - !Sub arn:aws:ec2:${AWS::Region}:${AWS::AccountId}:instance/
${LicenseInstanceId}


  WorkerPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      ManagedPolicyName: BYOLWorkerPolicy
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action:
              - logs:CreateLogStream
            Resource: !Sub arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-
group:/aws/deadline/${Farm.FarmId}/*
            Condition:
              ForAnyValue:StringEquals:
                aws:CalledVia:
                  - deadline.amazonaws.com
          - Effect: Allow
            Action:
              - logs:PutLogEvents
              - logs:GetLogEvents
            Resource: !Sub arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-
group:/aws/deadline/${Farm.FarmId}/*


  QueueRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: BYOLQueueRole
      ManagedPolicyArns:
        - !Ref QueuePolicy
        - !Ref BYOLSSMPolicy
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action:
```

```
            - sts:AssumeRole
          Principal:
            Service:
              - credentials.deadline.amazonaws.com
              - deadline.amazonaws.com
          Condition:
            StringEquals:
              aws:SourceAccount: !Sub ${AWS::AccountId}
            ArnEquals:
              aws:SourceArn: !Ref Farm

  WorkerRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: BYOLWorkerRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/AWSDeadlineCloud-FleetWorker
        - !Ref WorkerPolicy
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action:
              - sts:AssumeRole
            Principal:
              Service: credentials.deadline.amazonaws.com


  Queue:
    Type: AWS::Deadline::Queue
    Properties:
      DisplayName: BYOLQueue
      FarmId: !GetAtt Farm.FarmId
      RoleArn: !GetAtt QueueRole.Arn
      JobRunAsUser:
        Posix:
          Group: ""
          User: ""
        RunAs: WORKER_AGENT_USER
      JobAttachmentSettings:
        RootPrefix: job-attachments
        S3BucketName: !Ref JobAttachmentBucket

  Fleet:
```

```
    Type: AWS::Deadline::Fleet
    Properties:
      DisplayName: BYOLFleet
      FarmId: !GetAtt Farm.FarmId
      MinWorkerCount: 1
      MaxWorkerCount: 2
      Configuration:
        ServiceManagedEc2:
          InstanceCapabilities:
            VCpuCount:
              Min: 4
              Max: 16
            MemoryMiB:
              Min: 4096
              Max: 16384
            OsFamily: LINUX
            CpuArchitectureType: x86_64
          InstanceMarketOptions:
            Type: on-demand
      RoleArn: !GetAtt WorkerRole.Arn

  QFA:
    Type: AWS::Deadline::QueueFleetAssociation
    Properties:
      FarmId: !GetAtt Farm.FarmId
      FleetId: !GetAtt Fleet.FleetId
      QueueId: !GetAtt Queue.QueueId

  CondaQueueEnvironment:
    Type: AWS::Deadline::QueueEnvironment
    Properties:
      FarmId: !GetAtt Farm.FarmId
      Priority: 5
      QueueId: !GetAtt Queue.QueueId
      TemplateType: YAML
      Template: |
        specificationVersion: 'environment-2023-09'
        parameterDefinitions:
        - name: CondaPackages
          type: STRING
          description: >
            This is a space-separated list of conda package match specifications to
install for the job.
            E.g. "blender=3.6" for a job that renders frames in Blender 3.6.
```

```
            See https://docs.conda.io/projects/conda/en/latest/user-guide/concepts/
pkg-specs.html#package-match-specifications
          default: ""
          userInterface:
            control: LINE_EDIT
            label: Conda Packages
        - name: CondaChannels
          type: STRING
          description: >
            This is a space-separated list of conda channels from which to install
 packages. &ADC; SMF packages are
            installed from the "deadline-cloud" channel that is configured by
 &ADC;.

            Add "conda-forge" to get packages from the https://conda-forge.org/
 community, and "defaults" to get packages
            from Anaconda Inc (make sure your usage complies with https://
www.anaconda.com/terms-of-use).
          default: "deadline-cloud"
          userInterface:
            control: LINE_EDIT
            label: Conda Channels
        environment:
          name: Conda
          script:
            actions:
              onEnter:
                command: "conda-queue-env-enter"
                args: ["{{Session.WorkingDirectory}}/.env", "--packages",
 "{{Param.CondaPackages}}", "--channels", "{{Param.CondaChannels}}"]
              onExit:
                command: "conda-queue-env-exit"

  BYOLQueueEnvironment:
    Type: AWS::Deadline::QueueEnvironment
    Properties:
      FarmId: !GetAtt Farm.FarmId
      Priority: 10
      QueueId: !GetAtt Queue.QueueId
      TemplateType: YAML
      Template: !Sub |
        specificationVersion: "environment-2023-09"
        parameterDefinitions:
```

```
        - name: LicenseInstanceId
          type: STRING
          description: >
            The Instance ID of the license server/proxy instance
          default: ""
        - name: LicenseInstanceRegion
          type: STRING
          description: >
            The region containing this farm
          default: ""
        - name: LicensePorts
          type: STRING
          description: >
            Comma-separated list of ports to be forwarded to the license server/
proxy
            instance. Example:
 "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
          default: "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
        environment:
        name: BYOL License Forwarding
        variables:
          example_LICENSE: 2701@localhost
        script:
          actions:
          onEnter:
            command: bash
            args: [ "{{Env.File.Enter}}" ]
          onExit:
            command: bash
            args: [ "{{Env.File.Exit}}" ]
          embeddedFiles:
          - name: Enter
            type: TEXT
            runnable: True
            data: |
              curl https://s3.amazonaws.com/session-manager-downloads/plugin/
latest/linux_64bit/session-manager-plugin.rpm -Ls | rpm2cpio - | cpio -iv
 --to-stdout ./usr/local/sessionmanagerplugin/bin/session-manager-plugin >
 {{Session.WorkingDirectory}}/session-manager-plugin
              chmod +x {{Session.WorkingDirectory}}/session-manager-plugin
              conda activate
              python {{Env.File.StartSession}} {{Session.WorkingDirectory}}/
session-manager-plugin
          - name: Exit
```

```
            type: TEXT
            runnable: True
            data: |
              echo Killing SSM Manager Plugin PIDs: $BYOL_SSM_PIDS
              for pid in ${BYOL_SSM_PIDS//,/ }; do kill $pid; done
          - name: StartSession
            type: TEXT
            data: |
              import boto3
              import json
              import subprocess
              import sys
              import os
              import tempfile

              instance_id = "{{Param.LicenseInstanceId}}"
              region = "{{Param.LicenseInstanceRegion}}"
              license_ports_list = "{{Param.LicensePorts}}".split(",")

              ssm_client = boto3.client("ssm", region_name=region)
              pids = []

              for port in license_ports_list:
                session_response = ssm_client.start_session(
                  Target=instance_id,
                  DocumentName="AWS-StartPortForwardingSession",
                  Parameters={"portNumber": [port], "localPortNumber": [port]}
                )

                cmd = [
                  sys.argv[1],
                  json.dumps(session_response),
                  region,
                  "StartSession",
                  "",
                  json.dumps({"Target": instance_id}),
                  f"https://ssm.{region}.amazonaws.com"
                ]

                process = subprocess.Popen(cmd, stdout=subprocess.DEVNULL,
stderr=subprocess.DEVNULL)
                pids.append(process.pid)
                print(f"SSM Port Forwarding Session started for port {port}")
```

```
            print(f"openjd_env: BYOL_SSM_PIDS={','.join(str(pid) for pid in
 pids)}")

            # Enabling UBL after the "bring your own license" (BYOL) has run out
 requires prepending the BYOL configuration to the existing license setup
            # Remove the sections that do not apply to your pipeline, or you do
 not want to use UBL after exhausting the BYOL licenses.
            # The port numbers used may not match what your license server is
 serving.

            # Arnold
            os.environ["ADSKFLEX_LICENSE_FILE"] = f"2701@localhost:
{os.environ.get('ADSKFLEX_LICENSE_FILE', '')}"
            print(f"openjd_env:
 ADSKFLEX_LICENSE_FILE={os.environ['ADSKFLEX_LICENSE_FILE']}")

            # Nuke
            os.environ["foundry_LICENSE"] = f"6101@localhost:
{os.environ.get('foundry_LICENSE', '')}"
            print(f"openjd_env: foundry_LICENSE={os.environ['foundry_LICENSE']}")

            # SideFX
            os.environ["SESI_LMHOST"] = f"localhost:1715;
{os.environ.get('SESI_LMHOST', '')}"
            print(f"openjd_env: SESI_LMHOST={os.environ['SESI_LMHOST']}")

            # Redshift and Red Giant
            os.environ["redshift_LICENSE"] = f"7054@localhost:7055@localhost:
{os.environ.get('redshift_LICENSE', '')}"
            print(f"openjd_env:
 redshift_LICENSE={os.environ['redshift_LICENSE']}")

            # V-Ray doesn't support multiple license servers in a single
 environment variable
            # See https://documentation.chaos.com/space/LIC5/125050770/Sharing+a
+License+Configuration+in+a+Network
            vray_license = os.environ.get('VRAY_AUTH_CLIENT_SETTINGS', '')
            xml_content = """<VRLClient>
              <LicServer>
                <Host>localhost</Host>
                <Port>30304</Port>"""

            if vray_license and vray_license.startswith('licset://'):
                server_parts = vray_license.removeprefix('licset://').split(':')
```

```
                if len(server_parts) >= 2:
                    xml_content += f"""
                <Host1>{server_parts[0]}</Host1>
                <Port1>{server_parts[1]}</Port1>"""

            xml_content += """
                <User></User>
                <Pass></Pass>
              </LicServer>
            </VRLClient>"""

            temp_dir = tempfile.gettempdir()
            xml_path = os.path.join(temp_dir, 'vrlclient.xml')

            with open(xml_path, 'w') as f:
                f.write(xml_content)

            os.environ["VRAY_AUTH_CLIENT_FILE_PATH"] = temp_dir
            print(f"openjd_env:
VRAY_AUTH_CLIENT_FILE_PATH={os.environ['VRAY_AUTH_CLIENT_FILE_PATH']}")

            # Clear the existing VRAY_AUTH_CLIENT_SETTINGS so only the
vrlclient.xml file is used.
            os.environ["VRAY_AUTH_CLIENT_SETTINGS"] = ''
            print(f"openjd_env:
VRAY_AUTH_CLIENT_SETTINGS={os.environ['VRAY_AUTH_CLIENT_SETTINGS']}")

            # Print out the created xml file's contents
            print(f"V-Ray configuration file: {xml_path}")
            with open(xml_path, 'r') as f:
                print(f"{f.read()}")
```

3. Geben Sie bei der Bereitstellung der CloudFormation Vorlage die folgenden Parameter an:

   - Aktualisieren Sie die LicenseInstanceID mit der Amazon EC2 EC2-Instance-ID Ihres Lizenzservers oder Ihrer Proxy-Instance
   - Aktualisieren Sie die LicensePortsmit einer durch Kommas getrennten Liste von Ports, die an den Lizenzserver oder die Proxy-Instance weitergeleitet werden sollen (z. B. 2700,2701)
   - Fügen Sie die Lizenzumgebungsvariablen hinzu, indem Sie sie in der Vorlage ersetzen **`example_LICENSE: 2700@localhost`**

4. Stellen Sie die Vorlage bereit, um Ihre Farm mit der Funktion „Bring Your Own License“ einzurichten.

Connect vom Kunden verwaltete Flotten mit einem Lizenzendpunkt

Der nutzungsbasierte Lizenzserver von AWS Deadline Cloud bietet On-Demand-Lizenzen für ausgewählte Produkte von Drittanbietern. Bei nutzungsabhängigen Lizenzen können Sie nutzungsabhängig bezahlen. Ihnen wird nur die Nutzungsdauer in Rechnung gestellt. Bei der nutzungsbasierten Lizenzierung werden Lizenzen für das Rendern Ihrer Deadline Cloud-Mitarbeiter bereitgestellt, nicht jedoch Lizenzen für Ihre DCC-Anwendungen.

Der nutzungsbasierte Lizenzserver von Deadline Cloud kann mit jedem Flotten-Typ verwendet werden, sofern die Deadline Cloud-Mitarbeiter mit dem Lizenzserver kommunizieren können. Der Lizenzserver wird automatisch für vom Service verwaltete Flotten eingerichtet. Die folgende Konfiguration ist nur für vom Kunden verwaltete Flotten erforderlich.

Um den Lizenzserver zu erstellen, benötigen Sie eine Sicherheitsgruppe für die VPC Ihrer Farm, die Datenverkehr für Drittanbieterlizenzen zulässt.

Themen

Schritt 1: Erstellen Sie eine Sicherheitsgruppe

Verwenden Sie die Amazon VPC-Konsole, um eine Sicherheitsgruppe für die VPC Ihrer Farm zu erstellen. Konfigurieren Sie die Sicherheitsgruppe so, dass sie die folgenden Regeln für eingehenden Datenverkehr zulässt:

- Autodesk Maya und Arnold — 2701 — 2702, TCP, IPv4 IPv6
- Kino 4D — 7057, TCP, IPv4 IPv6
- KeyShot — 2703, TCP, IPv4 IPv6
- Foundry Nuke — 6101, TCP, IPv4 IPv6
- Roter Riese — 7055, TCP, IPV4
- Redshift — 7054, TCP, IPv4 IPv6
- SideFX Houdini, Mantra und Karma — 1715 - 1717, TCP, IPv4 IPv6
- VRay — 30304, TCP, IPV4

Die Quelle für jede Regel für eingehenden Datenverkehr ist die Worker-Sicherheitsgruppe der Flotte.

Weitere Informationen zum Erstellen einer Sicherheitsgruppe finden Sie unter Erstellen einer Sicherheitsgruppe im Amazon Virtual Private Cloud Cloud-Benutzerhandbuch.

Schritt 2: Richten Sie den Lizenzendpunkt ein

Ein Lizenzendpunkt ermöglicht den Zugriff auf Lizenzserver für Produkte von Drittanbietern. Lizenzanfragen werden an den Lizenzendpunkt gesendet. Der Endpunkt leitet sie an den entsprechenden Lizenzserver weiter. Der Lizenzserver verfolgt Nutzungsbeschränkungen und Berechtigungen. Durch das Erstellen eines Lizenzendpunkts in Deadline Cloud wird ein AWS PrivateLink Schnittstellenendpunkt in Ihrer VPC bereitgestellt. Diese Endpunkte werden gemäß den Standardpreisen abgerechnet. AWS PrivateLink Weitere Informationen finden Sie unter AWS PrivateLink Preise.

Mit den entsprechenden Berechtigungen können Sie Ihren Lizenzendpunkt erstellen. Die für die Erstellung eines Lizenzendpunkts erforderliche Richtlinie finden Sie unter Richtlinie zur Zulassung der Erstellung eines Lizenzendpunkts.

Sie können Ihren Lizenzendpunkt über Ihr Dashboard in der Deadline Cloud-Konsole erstellen.

1. Wählen Sie im linken Navigationsbereich Lizenzendpunkte und dann Lizenzendpunkt erstellen aus.
2. Gehen Sie auf der Seite Lizenzendpunkt erstellen wie folgt vor:
   - Wählen Sie eine VPC aus.
   - Wählen Sie die Subnetze aus, die Ihre Deadline Cloud-Worker enthalten. Sie können bis zu 10 Subnetze auswählen.
   - Wählen Sie die Sicherheitsgruppe aus, die Sie in Schritt 1 erstellt haben. Sie können bis zu 10 Sicherheitsgruppen für kompliziertere Szenarien auswählen.
   - (Optional) Wählen Sie Neues Tag hinzufügen und fügen Sie ein oder mehrere Tags hinzu. Sie können bis zu 50 Tags hinzufügen.
3. Wählen Sie Lizenzendpunkt erstellen aus. Wenn der Lizenzendpunkt erstellt wird, wird er auf der Seite mit den Lizenzendpunkten angezeigt.
4. Wählen Sie im Bereich „Gemessene Produkte“ die Option Produkte hinzufügen und wählen Sie dann die Produkte aus, die Sie Ihrem Lizenzendpunkt hinzufügen möchten. Wählen Sie Hinzufügen aus.

Um ein Produkt von einem Lizenzendpunkt zu entfernen, wählen Sie im Bereich Produkte mit Nutzungsdauer das Produkt aus und klicken Sie dann auf Entfernen. Wählen Sie in der Bestätigung erneut Entfernen aus.

Schritt 3: Eine Rendering-Anwendung mit einem Endpunkt Connect

Nachdem der Lizenzendpunkt eingerichtet wurde, verwenden Anwendungen ihn genauso wie einen Lizenzserver eines Drittanbieters. In der Regel konfigurieren Sie den Lizenzserver für die Anwendung, indem Sie eine Umgebungsvariable oder eine andere Systemeinstellung, z. B. einen Microsoft Windows-Registrierungsschlüssel, auf einen Lizenzserverport und eine Adresse festlegen.

Um den DNS-Namen des Lizenzendpunkts abzurufen, wählen Sie den Lizenzendpunkt in der Konsole aus und klicken Sie dann im Bereich DNS-Name auf das Kopiersymbol.

Beispiele für Konfigurationen

Example— Autodesk Maya und Arnold

> **Note**
>
> Sie können Autodesk Maya und Arnold zusammen oder getrennt verwenden. Verwenden Sie Port 2702 für Autodesk Maya und Port 2701 für Arnold.

Stellen Sie für Autodesk Maya die Umgebungsvariable `ADSKFLEX_LICENSE_FILE` auf:

```
2702@VPC_Endpoint_DNS_Name
```

Setzen Sie für Arnold die Umgebungsvariable `ADSKFLEX_LICENSE_FILE` auf:

```
2701@VPC_Endpoint_DNS_Name
```

Setzen Sie die Umgebungsvariable für Autodesk Maya und Arnold `ADSKFLEX_LICENSE_FILE` auf:

```
2702@VPC_Endpoint_DNS_Name:2701@VPC_Endpoint_DNS_Name
```

> **Note**
>
> Verwenden Sie für Windows Worker ein Semikolon (;) anstelle eines Doppelpunkts (:), um die Endpunkte voneinander zu trennen.

Example— Kino 4D

Setzen Sie die Umgebungsvariable `g_licenseServerRLM` auf:

```
VPC_Endpoint_DNS_Name:7057
```

Nachdem Sie die Umgebungsvariable erstellt haben, sollten Sie in der Lage sein, ein Bild mit einer Befehlszeile ähnlich der folgenden zu rendern:

```
"C:\Program Files\Maxon Cinema 4D 2025\Commandline.exe" -render ^
    "C:\Users\User\MyC4DFileWithRedshift.c4d" -frame 0 ^
    -oimage "C:\Users\Administrator\User\MyOutputImage.png
```

Example– KeyShot

Setzen Sie die Umgebungsvariable `LUXION_LICENSE_FILE` auf:

```
2703@VPC_Endpoint_DNS_Name
```

Nach der Installation KeyShot und Ausführung können `pip install deadline-cloud-for-keyshot` Sie mit dem folgenden Befehl testen, ob die Lizenz funktioniert. Das Skript validiert Ihre Einstellungen, rendert aber nichts.

```
"C:\Program Files\KeyShot12\bin\keyshot_headless.exe" ^
    -floating_feature keyshot2 ^
    -floating_license_server 2703@VPC_Endpoint_DNS_Name ^
    -script "C:\Program Files\Python311\Lib\site-packages\deadline\keyshot_adaptor
\KeyShotClient\keyshot_handler.py"
```

Die Antwort sollte Folgendes ohne Fehlermeldungen enthalten:

```
Connecting to floating license server
```

Example— Foundry Nuke

Setzen Sie die Umgebungsvariable `foundry_LICENSE` auf:

```
6101@VPC_Endpoint_DNS_Name
```

Um zu testen, ob die Lizenzierung ordnungsgemäß funktioniert, können Sie Nuke in einem Terminal ausführen:

```
~/nuke/Nuke14.0v5/Nuke14.0 -x
```

Example— Roter Riese

Setzen Sie die Umgebungsvariable `redshift_LICENSE` auf:

```
7055@VPC_Endpoint_DNS_Name
```

Beachten Sie, dass Red Giant und Redshift dieselbe `redshift_LICENSE` Umgebungsvariable haben. Wenn Sie beide Anwendungen verwenden möchten, können Sie die Umgebungsvariable wie folgt setzen:

```
7054@VPC_Endpoint_DNS_Name:7055@VPC_Endpoint_DNS_Name
```

> **Note**
>
> Verwenden Sie für Windows Worker ein Semikolon (;) anstelle eines Doppelpunkts (:), um die Endpunkte voneinander zu trennen.

Um zu testen, ob die Lizenzierung ordnungsgemäß funktioniert, stellen Sie sicher, dass After Effects und Red Giant installiert sind. Anschließend können Sie ein Projekt mit einem Befehl rendern, der dem folgenden ähnelt:

```
C:\Program Files\Adobe\Adobe After Effects 2025\Support Files\aerender.exe -comp "Comp
 1" -project
         C:\Users\MyUser\myAfterEffectsProjectUsingRedGiant.aep -output
           C:\Users\MyUser\myMovieWithRedGiant.mp4
```

Example— Redshift

Setzen Sie die Umgebungsvariable `redshift_LICENSE` auf:

```
7054@VPC_Endpoint_DNS_Name
```

Nachdem Sie die Umgebungsvariable erstellt haben, sollten Sie in der Lage sein, ein Bild mit einer Befehlszeile ähnlich der folgenden zu rendern:

```
C:\ProgramData\redshift\bin\redshiftCmdLine.exe ^
    C:\demo\proxy\RS_Proxy_Demo.rs ^
    -oip C:\demo\proxy\images
```

Example— SideFX Houdini, Mantra und Karma

Führen Sie den folgenden Befehl aus:

```
/opt/hfs19.5.640/bin/hserver -S
 "http://VPC_Endpoint_DNS_Name:1715;http://VPC_Endpoint_DNS_Name:1716;http://
VPC_Endpoint_DNS_Name:1717;"
```

Um zu testen, ob die Lizenzierung ordnungsgemäß funktioniert, können Sie eine Houdini-Szene mit diesem Befehl rendern:

```
/opt/hfs19.5.640/bin/hython ~/forpentest.hip -c "hou.node('/out/mantra1').render()"
```

Example– VRay

Setzen Sie die Umgebungsvariable `VRAY_AUTH_CLIENT_SETTINGS` auf:

```
licset://VPC_Endpoint_DNS_Name:30304
```

Setzen Sie die Umgebungsvariable `VRAY_AUTH_CLIENT_FILE_PATH` auf:

```
/null
```

Um zu testen, ob die Lizenzierung ordnungsgemäß funktioniert, können Sie ein Bild VRay mit einem Befehl rendern, der dem folgenden ähnelt:

```
/usr/Chaos/V-Ray/bin/vray -sceneFile=/root/my_scene.vrscene -display=0
```

Schritt 4: Löschen Sie einen Lizenzendpunkt

Denken Sie beim Löschen Ihrer kundenverwalteten Flotte daran, Ihren Lizenzendpunkt zu löschen. Wenn Sie den Lizenzendpunkt nicht löschen, werden Ihnen weiterhin die AWS PrivateLink Fixkosten in Rechnung gestellt

Sie können Ihren Lizenzendpunkt in Ihrem Dashboard in der Deadline Cloud-Konsole löschen.

1. Wählen Sie im linken Navigationsbereich die Option Lizenzendpunkte aus.
2. Wählen Sie den Endpunkt aus, den Sie löschen möchten, und klicken Sie dann zur Bestätigung erneut auf Löschen.

Einsatz von KI-Agenten mit Deadline Cloud

Verwenden Sie KI-Agenten, um Job-Bundles zu schreiben, Conda-Pakete zu entwickeln und Jobs in Deadline Cloud zu beheben. In diesem Thema wird erklärt, was KI-Agenten sind, was wichtig ist, um effektiv mit ihnen zu arbeiten, und Ressourcen, die Agenten dabei helfen, Deadline Cloud zu verstehen.

Ein KI-Agent ist ein Softwaretool, das ein großes Sprachmodell (LLM) verwendet, um Aufgaben autonom auszuführen. KI-Agenten können Dateien lesen und schreiben, Befehle ausführen und auf der Grundlage von Feedback Lösungen entwickeln. Beispiele hierfür sind Befehlszeilentools wie Kiro und IDE-integrierte Assistenten.

Wichtige Punkte für die Arbeit mit KI-Agenten

Die folgenden wichtigen Punkte helfen Ihnen dabei, bessere Ergebnisse zu erzielen, wenn Sie KI-Agenten mit Deadline Cloud verwenden.

- Grundlagen schaffen — KI-Agenten schneiden am besten ab, wenn sie Zugriff auf relevante Dokumentationen, Spezifikationen und Beispiele haben. Sie können die Grundlage schaffen, indem Sie den Agenten auf bestimmte Dokumentationsseiten verweisen, vorhandenen Beispielcode als Referenz weitergeben, relevante Open-Source-Repositorys in den lokalen Workspace klonen und Dokumentation für Anwendungen von Drittanbietern bereitstellen.
- Erfolgskriterien angeben — Definieren Sie das erwartete Ergebnis und die technischen Anforderungen für den Agenten. Wenn Sie beispielsweise einen Agenten bitten, ein Auftragspaket zu entwickeln, geben Sie die Auftragseingaben, Parameter und erwarteten Ergebnisse an. Wenn Sie sich bezüglich der Spezifikationen nicht sicher sind, bitten Sie den Agenten, zunächst Optionen vorzuschlagen und die Anforderungen dann gemeinsam zu verfeinern.
- Ermöglichen Sie eine Feedback-Schleife — KI-Agenten iterieren effektiver, wenn sie ihre Lösungen testen und Feedback erhalten können. Anstatt beim ersten Versuch eine funktionierende Lösung zu erwarten, geben Sie dem Agenten die Möglichkeit, seine Lösung auszuführen und die Ergebnisse zu überprüfen. Dieser Ansatz funktioniert gut, wenn der Agent auf Statusaktualisierungen, Protokolle und Validierungsfehler zugreifen kann. Wenn Sie beispielsweise ein Auftragspaket entwickeln, erlauben Sie dem Agenten, den Job einzureichen und die Protokolle zu überprüfen.
- Rechnen Sie mit Wiederholungen — Selbst bei einem guten Kontext können Agenten vom Kurs abweichen oder Annahmen treffen, die nicht zu Ihrer Umgebung passen. Beobachten Sie, wie der Agent an die Aufgabe herangeht, und geben Sie ihm dabei Unterstützung. Fügen Sie fehlenden

Kontext hinzu, falls der Agent Probleme hat, helfen Sie bei der Fehlersuche, indem Sie auf bestimmte Protokolldateien verweisen, verfeinern Sie die Anforderungen, sobald Sie sie entdecken, und fügen Sie negative Anforderungen hinzu, um explizit anzugeben, was der Agent vermeiden sollte.

Ressourcen für den Agentenkontext

Die folgenden Ressourcen helfen KI-Agenten dabei, die Konzepte von Deadline Cloud zu verstehen und genaue Ergebnisse zu erzielen.

- Deadline Cloud Model Context Protocol (MCP) -Server — Für Agenten, die das Model Context Protocol unterstützen, enthält das Deadline-Cloud-Repository den Deadline-Cloud-Client, der einen MCP-Server für die Interaktion mit Jobs enthält.
- AWSDokumentations-MCP-Server — Für Agenten, die MCP unterstützen, konfigurieren Sie den AWSDocumentation MCP-Server so, dass der Agent direkten Zugriff auf die Dokumentation hat, einschließlich des Deadline AWS Cloud-Benutzerhandbuchs und des Entwicklerhandbuchs.
- Open Job Description Specification — Die Open Job Description Specification on GitHub definiert das Schema für Jobvorlagen. Verweisen Sie auf dieses Repository, wenn Agenten die Struktur und Syntax von Jobvorlagen verstehen müssen.
- deadline-cloud-samples— Das deadline-cloud-samplesRepository enthält Beispiel-Job-Bundles, Conda-Rezepte und CloudFormation Vorlagen für gängige Anwendungen und Anwendungsfälle.
- aws-deadline GitHub organization — Die aws-deadline GitHub organization enthält Referenz-Plugins für viele Drittanbieteranwendungen, die Sie als Beispiele für andere Integrationen verwenden können.

Beispielaufforderung: Ein Job-Bundle schreiben

Die folgende Beispielaufforderung zeigt, wie ein KI-Agent verwendet wird, um Job-Bundles zu erstellen, die einen LoRa-Adapter (Low-Rank Adaptation) für die Generierung von KI-Images trainieren. Die Eingabeaufforderung veranschaulicht die zuvor erörterten wichtigsten Punkte: Sie bietet eine Grundlage, indem sie auf relevante Repositorien verweist, definiert Erfolgskriterien für die Job-Bundle-Ausgaben und skizziert eine Feedback-Schleife für die iterative Entwicklung.

```
Write a pair of job bundles for Deadline Cloud that use the diffusers Python library to
 train a LoRA adapter on a set of images and then generate images from it.

Requirements:
```

```
- The training job takes a set of JPEG images as input, uses an image description,
 LoRA rank, learning rate, batch size, and number of training steps as parameters, and
 outputs a `.safetensors` file.
- The generation job takes the `.safetensors` file as input and the number of images
 to generate, then outputs JPEG images. The jobs use Stable Diffusion 1.5 as the base
 model.
- The jobs run `diffusers` as a Python script. Install the necessary packages using
 conda by setting the job parameters:
    - `CondaChannels`: `conda-forge`
    - `CondaPackages`: list of conda packages to install

For context, clone the following repositories to your workspace and review their
 documentation and code:
- OpenJobDescription specification: https://github.com/OpenJobDescription/openjd-
specifications/blob/mainline/wiki/2023-09-Template-Schemas.md
- Deadline Cloud sample job bundles: https://github.com/aws-deadline/deadline-cloud-
samples/tree/mainline/job_bundles
- diffusers library: https://github.com/huggingface/diffusers

Read through the provided context before you start. To develop a job bundle, iterate
 with the following steps until the submitted job succeeds. If a step fails, update the
 job bundle and restart the loop:
1. Create a job bundle.
2. Validate the job template syntax: `openjd check`
3. Submit the job to Deadline Cloud: `deadline bundle submit`
4. Wait for the job to complete: `deadline job wait`
5. View the job status and logs: `deadline job logs`
6. Download the job output: `deadline job download-output`

To verify the training and generation jobs work together, iterate with the following
 steps until the generation job produces images that resemble the dog in the training
 data:
1. Develop and submit a training job using the training images in `./exdog`
2. Wait for the job to succeed then download its output.
3. Develop and submit a generation job using the LoRA adapter from the training job.
4. Wait for the job to succeed then download its output.
5. Inspect the generated images. If they resemble the dog in the training data, you're
 done. Otherwise, review the job template, job parameters, and job logs to identify and
 fix the issue.
```

AWSDeadline Cloud überwachen

Die Überwachung ist ein wichtiger Bestandteil der Aufrechterhaltung der Zuverlässigkeit, Verfügbarkeit und Leistung von AWS Deadline Cloud (Deadline Cloud) und Ihrer AWS Lösungen. Sammeln Sie Überwachungsdaten aus allen Teilen Ihrer AWS Lösung, damit Sie einen Fehler an mehreren Stellen leichter debuggen können, falls einer auftritt. Bevor Sie mit der Überwachung von Deadline Cloud beginnen, sollten Sie einen Überwachungsplan erstellen, der Antworten auf die folgenden Fragen enthält:

- Was sind Ihre Überwachungsziele?
- Welche Ressourcen möchten Sie überwachen?
- Wie oft werden diese Ressourcen überwacht?
- Welche Überwachungs-Tools möchten Sie verwenden?
- Wer soll die Überwachungsaufgaben ausführen?
- Wer soll benachrichtigt werden, wenn Fehler auftreten?

AWSund Deadline Cloud bieten Tools, mit denen Sie Ihre Ressourcen überwachen und auf potenzielle Vorfälle reagieren können. Einige dieser Tools übernehmen die Überwachung für Sie, andere Tools erfordern manuelles Eingreifen. Sie sollten die Überwachungsaufgaben so weit wie möglich automatisieren.

- Amazon CloudWatch überwacht Ihre AWS Ressourcen und die Anwendungen, auf denen Sie laufen, AWS in Echtzeit. Sie können Kennzahlen erfassen und verfolgen, benutzerdefinierte Dashboards erstellen und Alarme festlegen, die Sie benachrichtigen oder Maßnahmen ergreifen, wenn eine bestimmte Metrik einen von Ihnen festgelegten Schwellenwert erreicht. Sie können beispielsweise die CPU-Auslastung oder andere Kennzahlen Ihrer EC2 Amazon-Instances CloudWatch verfolgen und bei Bedarf automatisch neue Instances starten. Weitere Informationen finden Sie im CloudWatch Amazon-Benutzerhandbuch.

  Deadline Cloud hat drei CloudWatch Metriken.

- Mit Amazon CloudWatch Logs können Sie Ihre Protokolldateien von EC2 Amazon-Instances und anderen Quellen überwachen CloudTrail, speichern und darauf zugreifen. CloudWatch Logs kann Informationen in den Protokolldateien überwachen und Sie benachrichtigen, wenn bestimmte Schwellenwerte erreicht werden. Sie können Ihre Protokolldaten auch in einem sehr

robusten Speicher archivieren. Weitere Informationen finden Sie im Amazon CloudWatch Logs-Benutzerhandbuch.

- Amazon EventBridge kann verwendet werden, um Ihre AWS Services zu automatisieren und automatisch auf Systemereignisse wie Probleme mit der Anwendungsverfügbarkeit oder Ressourcenänderungen zu reagieren. Ereignisse im AWS Rahmen von Services werden nahezu EventBridge in Echtzeit zugestellt. Sie können einfache Regeln schreiben, um anzugeben, welche Ereignisse für Sie interessant sind und welche automatisierten Aktionen ausgeführt werden sollen, wenn ein Ereignis mit einer Regel übereinstimmt. Weitere Informationen finden Sie im EventBridge Amazon-Benutzerhandbuch.
- AWS CloudTrailerfasst API-Aufrufe und zugehörige Ereignisse, die von oder im Namen Ihres AWS Kontos getätigt wurden, und übermittelt die Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket. Sie können feststellen, welche Benutzer und Konten angerufen wurdenAWS, von welcher Quell-IP-Adresse aus die Anrufe getätigt wurden und wann die Aufrufe erfolgten. Weitere Informationen finden Sie im AWS CloudTrail-Benutzerhandbuch.

Themen

Protokollieren von Deadline Cloud API-Aufrufen mit AWS CloudTrail

Deadline Cloud ist in einen Dienst integriert AWS CloudTrail, der eine Aufzeichnung der von einem Benutzer, einer Rolle oder einem ausgeführten Aktionen bereitstellt AWS-Service. CloudTrail erfasst alle API-Aufrufe Deadline Cloud als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der Deadline Cloud Konsole und Codeaufrufen für die Deadline Cloud API-Operationen. Anhand der von gesammelten Informationen können Sie die Anfrage CloudTrail, an die die Anfrage gestellt wurde Deadline Cloud, die IP-Adresse, von der aus die Anfrage gestellt wurde, den Zeitpunkt der Anfrage und weitere Details ermitteln.

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Ob die Anforderung mit Anmeldeinformationen des Root-Benutzers oder des Benutzers gestellt wurde.
- Die Anforderung wurde im Namen eines IAM-Identity-Center-Benutzers erstellt.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anforderung aus einem anderen AWS-Service gesendet wurde.

CloudTrail ist in Ihrem aktiv AWS-Konto , wenn Sie das Konto erstellen, und Sie haben automatisch Zugriff auf den CloudTrail Eventverlauf. Der CloudTrail Ereignisverlauf bietet eine einsehbare, durchsuchbare, herunterladbare und unveränderliche Aufzeichnung der aufgezeichneten Verwaltungsereignisse der letzten 90 Tage in einem. AWS-Region Weitere Informationen finden Sie im AWS CloudTrail Benutzerhandbuch unter Arbeiten mit dem CloudTrail Ereignisverlauf. Für die Anzeige des Ereignisverlaufs CloudTrail fallen keine Gebühren an.

Für eine fortlaufende Aufzeichnung der Ereignisse in AWS-Konto den letzten 90 Tagen erstellen Sie einen Trail- oder CloudTrailLake-Event-Datenspeicher.

CloudTrail Pfade

Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Alle mit dem erstellten Pfade AWS-Managementkonsole sind regionsübergreifend. Sie können mithilfe von AWS CLI einen Einzel-Region- oder einen Multi-Region-Trail erstellen. Es wird empfohlen, einen Trail mit mehreren Regionen zu erstellen, da Sie alle Aktivitäten AWS-Regionen in Ihrem Konto erfassen. Wenn Sie einen Einzel-Region-Trail erstellen, können Sie nur die Ereignisse anzeigen, die im AWS-Region des Trails protokolliert wurden. Weitere Informationen zu Trails finden Sie unter Erstellen eines Trails für Ihr AWS-Konto und Erstellen eines Trails für eine Organisation im AWS CloudTrail -Benutzerhandbuch.

Sie können eine Kopie Ihrer laufenden Verwaltungsereignisse kostenlos an Ihren Amazon S3 S3-Bucket senden, CloudTrail indem Sie einen Trail erstellen. Es fallen jedoch Amazon S3 S3-Speichergebühren an. Weitere Informationen zur CloudTrail Preisgestaltung finden Sie unter AWS CloudTrail Preise. Informationen zu Amazon-S3-Preisen finden Sie unter Amazon S3 – Preise.

CloudTrail Datenspeicher für Ereignisse in Lake

CloudTrail Mit Lake können Sie SQL-basierte Abfragen für Ihre Ereignisse ausführen. CloudTrail Lake konvertiert bestehende Ereignisse im zeilenbasierten JSON-Format in das Apache ORC-Format. ORC ist ein spaltenförmiges Speicherformat, das für den schnellen Abruf von Daten optimiert ist. Die Ereignisse werden in Ereignisdatenspeichern zusammengefasst, bei denen es

sich um unveränderliche Sammlungen von Ereignissen handelt, die auf Kriterien basieren, die Sie mit Hilfe von erweiterten Ereignisselektoren auswählen. Die Selektoren, die Sie auf einen Ereignisdatenspeicher anwenden, steuern, welche Ereignisse bestehen bleiben und für Sie zur Abfrage verfügbar sind. Weitere Informationen zu CloudTrail Lake finden Sie unter Arbeiten mit AWS CloudTrail Lake im AWS CloudTrail Benutzerhandbuch.

CloudTrail Für das Speichern und Abfragen von Ereignisdaten in Lake fallen Kosten an. Beim Erstellen eines Ereignisdatenspeichers wählen Sie die Preisoption aus, die für den Ereignisdatenspeicher genutzt werden soll. Die Preisoption bestimmt die Kosten für die Erfassung und Speicherung von Ereignissen sowie die standardmäßige und maximale Aufbewahrungsdauer für den Ereignisdatenspeicher. Weitere Informationen zur CloudTrail Preisgestaltung finden Sie unter AWS CloudTrail Preise.

Deadline Cloud Datenereignisse in CloudTrail

Datenereignisse liefern Informationen über die Ressourcenoperationen, die auf oder in einer Ressource ausgeführt werden (z. B. Lesen oder Schreiben in ein Amazon-S3-Objekt). Sie werden auch als Vorgänge auf Datenebene bezeichnet. Datenereignisse sind oft Aktivitäten mit hohem Volume. Protokolliert standardmäßig CloudTrail keine Datenereignisse. Der CloudTrail Ereignisverlauf zeichnet keine Datenereignisse auf.

Für Datenereignisse werden zusätzliche Gebühren fällig. Weitere Informationen zur CloudTrail Preisgestaltung finden Sie unter AWS CloudTrail Preisgestaltung.

Sie können Datenereignisse für die Deadline Cloud Ressourcentypen mithilfe der CloudTrail Konsole oder CloudTrail API-Operationen protokollieren. AWS CLI Weitere Informationen zum Protokollieren von Datenereignissen finden Sie unter Protokollieren von Datenereignissen mit dem AWS-Managementkonsole und Protokollieren von Datenereignissen mit dem AWS Command Line Interface im AWS CloudTrail -Benutzerhandbuch.

In der folgenden Tabelle sind die Deadline Cloud Ressourcentypen aufgeführt, für die Sie Datenereignisse protokollieren können. In der Spalte Datenereignistyp (Konsole) wird der Wert angezeigt, den Sie in der Liste Datenereignistyp auf der CloudTrail Konsole auswählen können. In der Wertspalte resources.type wird der `resources.type` Wert angezeigt, den Sie bei der Konfiguration erweiterter Event-Selektoren mithilfe von oder angeben würden. AWS CLI CloudTrail APIs In der CloudTrail Spalte APIs Protokollierte Daten werden die API-Aufrufe angezeigt, die CloudTrail für den Ressourcentyp protokolliert wurden.

Typ des Datenereignisses (Konsole)	resources.type-Wert	Daten, die APIs protokolliert wurden CloudTrail
Deadline Fleet	`AWS::Deadline::Fleet`	• SearchWorkers
Warteschlange	`AWS::Deadline::Fleet`	• SearchJobs
Deadline Worker	`AWS::Deadline::Worker`	• GetWorker • ListSessionsForWorker • UpdateWorkerSchedule • BatchGetJobEntity • ListWorkers
Job mit Deadline	`AWS::Deadline::Job`	• ListStepConsumers • UpdateTask • ListJobs • GetStep • ListSteps • GetJob • GetTask • GetSession • ListSessions • CreateJob • ListSessionActions • ListTasks • CopyJobTemplate • UpdateSession • UpdateStep • UpdateJob • ListJobParameterDefinitions • GetSessionAction • ListStepDependencies

Typ des Datenereignisses (Konsole)	resources.type-Wert	Daten, die APIs protokolliert wurden CloudTrail
		• SearchTasks • SearchSteps

Sie können erweiterte Event-Selektoren so konfigurieren, dass sie nach den Feldern `eventName`, `readOnly` und `resources.ARN` filtern, sodass nur die Ereignisse protokolliert werden, die für Sie wichtig sind. Weitere Informationen zu diesen Kontingenten finden Sie unter AdvancedFieldSelector in derAWS CloudTrail -API-Referenz.

Deadline Cloud Management-Veranstaltungen in CloudTrail

Verwaltungsereignisse bieten Informationen über Verwaltungsvorgänge, die an Ressourcen in Ihrem ausgeführt werden AWS-Konto. Sie werden auch als Vorgänge auf Steuerebene bezeichnet. CloudTrail Protokolliert standardmäßig Verwaltungsereignisse.

AWS Deadline Cloud protokolliert die folgenden Operationen auf der Deadline Cloud Steuerungsebene CloudTrail als Verwaltungsereignisse.

- associate-member-to-farm
- associate-member-to-fleet
- associate-member-to-job
- associate-member-to-queue
- assume-fleet-role-for-gelesen
- assume-fleet-role-for-Arbeiter
- assume-queue-role-for-gelesen
- assume-queue-role-for-Benutzer
- assume-queue-role-for-Arbeiter
- Budget erstellen
- Farm erstellen
- create-fleet
- create-license-endpoint
- Limit erstellen

- Monitor erstellen
- Warteschlange erstellen
- create-queue-environment
- create-queue-fleet-association
- create-queue-limit-association
- create-storage-profile
- Mitarbeiter erstellen
- Budget löschen
- Farm löschen
- delete-fleet
- delete-license-endpoint
- Limit löschen
- delete-metered-product
- Monitor löschen
- Warteschlange löschen
- delete-queue-environment
- delete-queue-fleet-association
- delete-queue-limit-association
- delete-storage-profile
- Mitarbeiter löschen
- disassociate-member-from-farm
- disassociate-member-from-fleet
- disassociate-member-from-job
- disassociate-member-from-queue
- get-application-version
- Holen Sie sich ein Budget
- Farm abrufen
- get-feature-map
- Holen Sie sich die Flotte

- get-license-endpoint
- Limit abrufen
- Monitor abrufen
- Warteschlange abrufen
- get-queue-environment
- get-queue-fleet-association
- get-queue-limit-association
- get-sessions-statistics-aggregation
- get-storage-profile
- get-storage-profile-for-Warteschlange
- list-available-metered-products
- Budgets auflisten
- list-farm-members
- Farmen auflisten
- list-fleet-members
- Flotten auflisten
- list-job-members
- list-license-endpoints
- Limit für die Liste
- list-metered-products
- listet Monitore
- list-queue-environments
- list-queue-fleet-associations
- list-queue-limit-associations
- list-queue-members
- list-queues
- list-storage-profiles
- list-storage-profiles-for-Warteschlange
- list-tags-for-resource
- put-metered-product

- start-sessions-statistics-aggregation
- tag-resource
- untag-resource
- Budget aktualisieren
- Farm aktualisieren
- Flotte aktualisieren
- Limit aktualisieren
- Monitor aktualisieren
- Aktualisierungswarteschlange
- update-queue-environment
- update-queue-fleet-association
- update-queue-limit-association
- update-storage-profile
- Update-Worker

Deadline Cloud Beispiele für Ereignisse

Ein Ereignis stellt eine einzelne Anfrage aus einer beliebigen Quelle dar und enthält Informationen über den angeforderten API-Vorgang, Datum und Uhrzeit des Vorgangs, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordneter Stack-Trace der öffentlichen API-Aufrufe, sodass Ereignisse nicht in einer bestimmten Reihenfolge angezeigt werden.

Das folgende Beispiel zeigt ein CloudTrail Ereignis, das den `CreateFarm` Vorgang demonstriert.

```
{
    "eventVersion": "0",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EXAMPLE-PrincipalID:EXAMPLE-Session",
        "arn": "arn:aws:sts::111122223333:assumed-role/EXAMPLE-UserName/EXAMPLE-
Session",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE-accessKeyId",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
```

```
                "principalId": "EXAMPLE-PrincipalID",
                "arn": "arn:aws:iam::111122223333:role/EXAMPLE-UserName",
                "accountId": "111122223333",
                "userName": "EXAMPLE-UserName"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-03-08T23:25:49Z"
            }
        }
    },
    "eventTime": "2021-03-08T23:25:49Z",
    "eventSource": "deadline.amazonaws.com",
    "eventName": "CreateFarm",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "EXAMPLE-userAgent",
    "requestParameters": {
        "displayName": "example-farm",
        "kmsKeyArn": "arn:aws:kms:us-west-2:111122223333:key/111122223333",
        "X-Amz-Client-Token": "12abc12a-1234-1abc-123a-1a11bc1111a",
        "description": "example-description",
        "tags": {
            "purpose_1": "e2e"
            "purpose_2": "tag_test"
        }
    },
    "responseElements": {
        "farmId": "EXAMPLE-farmID"
    },
    "requestID": "EXAMPLE-requestID",
    "eventID": "EXAMPLE-eventID",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
    "eventCategory": "Management",
}
```


Das JSON-Beispiel zeigt die AWS-Region IP-Adresse und andere "" wie `requestParameters` "" und "`displayName`", die Ihnen bei der Identifizierung des Ereignisses helfen können. `kmsKeyArn`

Informationen zu CloudTrail Datensatzinhalten finden Sie im AWS CloudTrail Benutzerhandbuch unter CloudTrailDatensatzinhalte.

Überwachung mit CloudWatch

Amazon CloudWatch (CloudWatch) sammelt Rohdaten und verarbeitet sie zu lesbaren, nahezu in Echtzeit verfügbaren Metriken. Sie können die CloudWatch Konsole unter öffnen https://console.aws.amazon.com/cloudwatch/, um Deadline Cloud-Metriken anzuzeigen und zu filtern.

Diese Statistiken werden 15 Monate lang aufbewahrt, sodass Sie auf historische Informationen zugreifen können, um einen besseren Überblick über die Leistung Ihrer Webanwendung oder Ihres Dienstes zu erhalten. Sie können auch Alarme einrichten, die auf bestimmte Grenzwerte achten und Benachrichtigungen senden oder Aktivitäten auslösen, wenn diese Grenzwerte erreicht werden. Weitere Informationen finden Sie im CloudWatch Amazon-Benutzerhandbuch.

Deadline Cloud hat zwei Arten von Protokollen — Aufgabenprotokolle und Worker-Logs. Ein Aufgabenprotokoll liegt vor, wenn Sie Ausführungsprotokolle als Skript oder während der Ausführung von DCC ausführen. In einem Aufgabenprotokoll können Ereignisse wie das Laden von Objekten, das Rendern von Kacheln oder das Nichtauffinden von Texturen angezeigt werden.

Ein Worker-Protokoll zeigt die Prozesse des Worker-Agents. Dazu können Dinge gehören, z. B. wann der Worker-Agent startet, sich selbst registriert, Fortschritte meldet, Konfigurationen lädt oder Aufgaben abschließt.

Der Namespace für diese Protokolle lautet`/aws/deadline/*`.

Für Deadline Cloud laden Mitarbeiter diese Protokolle in Logs hoch. CloudWatch Standardmäßig laufen Protokolle nie ab. Wenn ein Job eine große Datenmenge ausgibt, können zusätzliche Kosten anfallen. Weitere Informationen finden Sie unter CloudWatch Amazon-Preise.

Sie können die Aufbewahrungsrichtlinie für jede Protokollgruppe anpassen. Eine kürzere Aufbewahrung entfernt alte Protokolle und kann zur Senkung der Speicherkosten beitragen. Um Protokolle aufzubewahren, können Sie sie in Amazon Simple Storage Service archivieren, bevor Sie das Protokoll entfernen. Weitere Informationen finden Sie unter Exportieren von Protokolldaten nach Amazon S3 mithilfe der Konsole im CloudWatch Amazon-Benutzerhandbuch.

> **Note**
>
> CloudWatch Das Lesen von Protokollen ist begrenzt durchAWS. Wenn Sie planen, viele Künstler an Bord zu nehmen, empfehlen wir Ihnen, sich an den AWS Kundensupport zu

wenden und eine Erhöhung des `GetLogEvents` Kontingents in zu beantragen CloudWatch. Außerdem empfehlen wir Ihnen, das Log-Tailing-Portal zu schließen, wenn Sie nicht debuggen.

Weitere Informationen finden Sie unter CloudWatch Log-Kontingente im CloudWatch Amazon-Benutzerhandbuch.

CloudWatch Metriken

Deadline Cloud sendet Metriken an Amazon CloudWatch. Sie können dieAWS-Managementkonsole, die oder eine API verwendenAWS CLI, um die Metriken aufzulisten, an die Deadline Cloud sendet CloudWatch. Standardmäßig deckt jeder Datenpunkt die 1 Minute ab, die auf die Startzeit der Aktivität folgt. Informationen darüber, wie Sie die verfügbaren Metriken mit dem AWS-Managementkonsole oder dem anzeigen könnenAWS CLI, finden Sie unter Verfügbare Metriken anzeigen im CloudWatch Amazon-Benutzerhandbuch.

Vom Kunden verwaltete Flottenkennzahlen

Der `AWS/DeadlineCloud` Namespace enthält die folgenden Kennzahlen für Ihre vom Kunden verwalteten Flotten:

Metrik	Description	Einheit
`RecommendedFleetSize`	Die Anzahl der Mitarbeiter, die Deadline Cloud für die Bearbeitung von Aufträgen empfiehlt. Sie können diese Kennzahl verwenden, um die Anzahl der Mitarbeiter aus Ihrer Flotte zu erhöhen oder zu verringern.	Anzahl
`UnhealthyWorkerCount`	Die Anzahl der Mitarbeiter, die für die Bearbeitung von Aufträgen zuständig sind, die nicht korrekt sind.	Anzahl

Sie können die folgenden Dimensionen verwenden, um die vom Kunden verwalteten Flottenkennzahlen zu verfeinern:

Dimension	Description
FarmId	Diese Dimension filtert die Daten, die Sie für die angegebene Farm anfordern.
FleetId	Diese Dimension filtert die Daten, die Sie für die angegebene Mitarbeiterflotte anfordern.

Kennzahlen zur Lizenzierung

Der `AWS/DeadlineCloud` Namespace enthält die folgenden Metriken für die Lizenzierung:

Metrik	Description	Einheit
`LicensesInUse`	Die Anzahl der verwendeten Lizenzsitzungen.	Anzahl

Sie können die folgenden Dimensionen verwenden, um die Lizenzkennzahlen zu verfeinern:

Dimension	Description
FleetId	Verwenden Sie diese Dimension, um die Daten nach der angegebenen, vom Service verwaltet en Flotte zu filtern. Verwenden Sie für vom Kunden verwaltete Flotten stattdessen die LicenseEndpointId Dimension.
LicenseEndpointId	Verwenden Sie diese Dimension, um die Daten nach dem angegebenen Lizenzendpunkt zu filtern.

Dimension	Description
Produkt	Verwenden Sie diese Dimension, um die Daten nach dem angegebenen gemessenen Produkt zu filtern.

Kennzahlen zum Ressourcenlimit

Der `AWS/DeadlineCloud` Namespace enthält die folgenden Metriken für Ressourcenlimits:

Metrik	Description	Einheit
`CurrentCount`	Die Anzahl der Ressourcen, die nach diesem verwendeten Limit modelliert werden.	Anzahl
`MaxCount`	Die maximale Anzahl von Ressourcen, die nach diesem Limit modelliert werden. Wenn Sie den `maxCount` Wert mithilfe der API auf -1 setzen, gibt Deadline Cloud die `MaxCount` Metrik nicht aus.	Anzahl

Sie können die folgenden Dimensionen verwenden, um die Metriken für das gleichzeitige Limit zu verfeinern:

Dimension	Description
FarmId	Diese Dimension filtert die Daten, die Sie für die angegebene Farm anfordern.
LimitId	Diese Dimension filtert die von Ihnen angeforderten Daten bis zum angegebenen Grenzwert.

Empfohlene Alarme

Mit können Sie Alarme erstellen CloudWatch, die Messwerte überwachen und Ihnen eine Benachrichtigung senden oder eine andere Aktion ausführen, wenn ein Schwellenwert überschritten wird. Weitere Informationen zur Konfiguration von CloudWatch Alarmen finden Sie im CloudWatch Amazon-Benutzerhandbuch.

Wir empfehlen, dass Sie Alarme für die folgenden Deadline Cloud-Metriken einrichten:

LicensesInUse

Abmessungen: FleetId, LicenseEndpointId

Beschreibung des Alarms: Dieser Alarm erkennt, wenn sich die aktiven Lizenzsitzungen für eine vom Service verwaltete Flotte oder einen Lizenzendpunkt Ihrem Kontokontingent nähern. In diesem Fall können Sie das Kontingent für Lizenzsitzungen erhöhen. Mithilfe von Service Quotas können Sie Ihre aktuellen Kontingente einsehen und Erhöhungen beantragen. Weitere Informationen finden Sie im Service Quotas User Guide.

Absicht: Vermeiden Sie Fehler beim Auschecken von Lizenzen, indem Sie die Nutzung überwachen, bevor das Kontingentlimit erreicht ist.

Statistik: Maximum

Empfohlener Schwellenwert: 90% Ihres Kontingents für Lizenzsitzungen

Begründung des Schwellenwerts: Legen Sie den Schwellenwert auf einen Prozentsatz Ihres Kontingents fest, sodass Sie Maßnahmen ergreifen können, bevor das Limit erreicht wird.

Zeitraum: 1 Minute

Datenpunkte bis Alarm: 1

Auswertungszeiträume: 1

Vergleichsoperator: `GREATER_THAN_THRESHOLD`

Weitere Ressourcen

- CloudWatch Amazon-Benutzerhandbuch

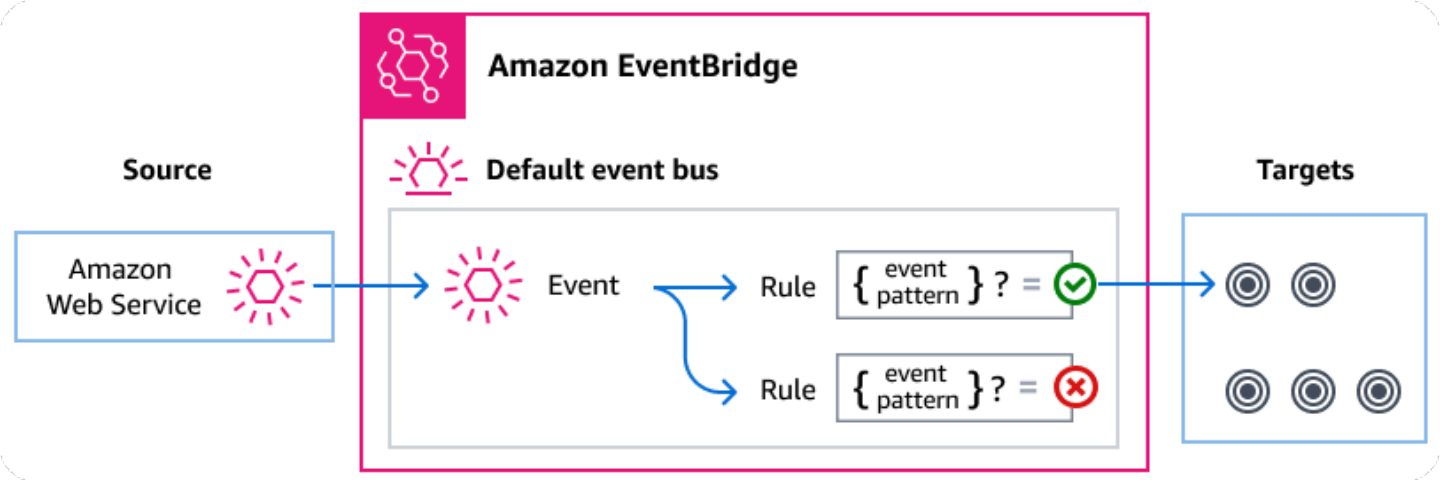
- Service Quotas-Benutzerhandbuch

Verwaltung von Deadline Cloud-Ereignissen mit Amazon EventBridge

Amazon EventBridge ist ein serverloser Dienst, der Ereignisse verwendet, um Anwendungskomponenten miteinander zu verbinden, sodass Sie leichter skalierbare, ereignisgesteuerte Anwendungen erstellen können. Bei der ereignisgesteuerten Architektur werden lose gekoppelte Softwaresysteme entwickelt, die zusammenarbeiten, indem sie Ereignisse senden und darauf reagieren. Ereignisse stellen eine Veränderung in einer Ressource oder Umgebung dar.

Funktionsweise:

Wie bei vielen AWS Diensten generiert Deadline Cloud Ereignisse und sendet sie an den EventBridge Standard-Event-Bus. (Der Standard-Event-Bus wird automatisch in jedem AWS Konto bereitgestellt.) Ein Event Bus ist ein Router, der Ereignisse empfängt und sie an null oder mehr Ziele weiterleitet. Die Regeln, die Sie für den Event Bus festlegen, werten die Ereignisse aus, sobald sie eintreffen. Jede Regel prüft, ob ein Ereignis dem Ereignismuster der Regelentspricht. Wenn das Ereignis übereinstimmt, sendet der Event Bus das Ereignis an das/die angegebene(n) Ziel(e).



Themen

- Frist für Cloud-Ereignisse
- Bereitstellung von Deadline Cloud-Ereignissen mithilfe von EventBridge Regeln
- Detailreferenz zu Deadline Cloud-Ereignissen

Frist für Cloud-Ereignisse

Deadline Cloud sendet die folgenden Ereignisse automatisch an den EventBridge Standard-Event-Bus. Ereignisse, die dem Ereignismuster einer Regel entsprechen, werden nach bestem Wissen und Gewissen an die angegebenen Ziele übermittelt. Ereignisse werden möglicherweise nicht in der richtigen Reihenfolge zugestellt.

Weitere Informationen finden Sie im Amazon EventBridge Benutzerhandbuch unter EventBridge Ereignisse.

Ereignisdetailtyp	Description
Budgetschwellenwert erreicht	Wird gesendet, wenn eine Warteschlange einen bestimmten Prozentsatz des zugewiesenen Budgets erreicht.
Änderung des Status des Joblebenszyklus	Wird gesendet, wenn sich der Lebenszyklusstatus eines Jobs ändert.
Änderung des Status der Auftragsausführung	Wird gesendet, wenn sich der Gesamtstatus der Aufgaben in einem Job ändert.
Schritt: Änderung des Lebenszyklusstatus	Wird gesendet, wenn sich der Lebenszyklusstatus eines Schritts in einem Job ändert.
Schritt „Ausführen“: Status ändern	Wird gesendet, wenn sich der Gesamtstatus der Aufgaben in einem Schritt ändert.
Änderung des Status der Aufgabenausführung	Wird gesendet, wenn sich der Status einer Aufgabe ändert.

Bereitstellung von Deadline Cloud-Ereignissen mithilfe von EventBridge Regeln

Damit der EventBridge Standard-Eventbus Deadline Cloud-Ereignisse an ein Ziel sendet, müssen Sie eine Regel erstellen. Jede Regel enthält ein Ereignismuster, das EventBridge mit jedem Ereignis übereinstimmt, das auf dem Event-Bus empfangen wurde. Wenn die Ereignisdaten mit dem angegebenen Ereignismuster EventBridge übereinstimmen, wird dieses Ereignis an die Ziele der Regel gesendet.

Umfassende Anweisungen zum Erstellen von Event-Bus-Regeln finden Sie im EventBridge Benutzerhandbuch unter Erstellen von Regeln, die auf Ereignisse reagieren.

Erstellen von Ereignismustern, die Deadline Cloud-Ereignissen entsprechen

Jedes Ereignismuster ist ein JSON-Objekt, das Folgendes enthält:

- Ein `source`-Attribut, das den Service identifiziert, der das Ereignis sendet. Für Deadline Cloud-Ereignisse lautet die Quelle`aws.deadline`.
- (Optional): Ein `detail-type`-Attribut, das ein Array der zuzuordnenden Ereignistypen enthält.
- (Optional): Ein `detail`-Attribut, das alle anderen Ereignisdaten für den Abgleich enthält.

Das folgende Ereignismuster entspricht beispielsweise allen Ereignissen zur Änderung der Fleet Size Recommendation für die `farmId` für Deadline Cloud angegebenen Ereignisse:

```
{
  "source": ["aws.deadline"],
  "detail-type": ["Fleet Size Recommendation Change"],
  "detail": {
     "farmId": "farm-12345678900000000000000000000000"
  }
}
```

Weitere Informationen zum Schreiben von Ereignismustern finden Sie unter Ereignismuster im EventBridge Benutzerhandbuch.

Detailreferenz zu Deadline Cloud-Ereignissen

Alle Ereignisse von AWS Diensten haben einen gemeinsamen Satz von Feldern, die Metadaten über das Ereignis enthalten, z. B. den AWS Dienst, der die Quelle des Ereignisses ist, den Zeitpunkt, zu dem das Ereignis generiert wurde, das Konto und die Region, in der das Ereignis stattgefunden hat, und andere. Definitionen dieser allgemeinen Felder finden Sie unter Referenz zur Ereignisstruktur im Amazon EventBridge Benutzerhandbuch.

Darüber hinaus weist jedes Ereignis ein `detail`-Feld auf, das spezifische Daten für das betreffende Ereignis enthält. In der folgenden Referenz werden die Detailfelder für die verschiedenen Deadline Cloud-Ereignisse definiert.

Bei der EventBridge Auswahl und Verwaltung von Deadline Cloud-Ereignissen ist es hilfreich, Folgendes zu beachten:

- Das `source` Feld für alle Ereignisse aus Deadline Cloud ist auf gesetzt`aws.deadline`.
- Das Feld `detail-type` gibt den Ereignistyp an.

  Beispiel, `Fleet Size Recommendation Change`.
- Das Feld `detail` enthält die Daten, die für das betreffende Ereignis spezifisch sind.

Informationen zur Erstellung von Ereignismustern, die es Regeln ermöglichen, Deadline Cloud-Ereignissen zu entsprechen, finden Sie unter Ereignismuster im Amazon EventBridge Benutzerhandbuch.

Weitere Informationen zu Ereignissen und deren EventBridge Verarbeitung finden Sie unter Amazon EventBridge Ereignisse im Amazon EventBridge Benutzerhandbuch.

Themen

Ereignis „Budgetschwellenwert erreicht“

Sie können das Ereignis „Budgetschwellenwert erreicht“ verwenden, um zu überwachen, wie viel Prozent eines Budgets verwendet wurden. Deadline Cloud sendet Ereignisse, wenn der verwendete Prozentsatz die folgenden Schwellenwerte überschreitet:

- 10, 20, 30, 40, 50, 60, 70, 75, 80, 85, 90, 95, 96, 97, 98, 99, 100

Die Häufigkeit, mit der Deadline Cloud Ereignisse vom Typ Budget Threshold Reached sendet, nimmt zu, wenn sich das Budget seinem Limit nähert. Diese Häufigkeit ermöglicht es Ihnen, ein Budget genau zu beobachten, wenn es sich seinem Limit nähert, und Maßnahmen zu ergreifen, um zu verhindern, dass zu viel ausgegeben wird. Sie können auch Ihre eigenen Budgetschwellenwerte

festlegen. Deadline Cloud sendet ein Ereignis, wenn die Nutzung Ihre benutzerdefinierten Schwellenwerte überschreitet.

Wenn Sie die Höhe eines Budgets ändern, basiert das nächste Mal, wenn Deadline Cloud das Ereignis „Budgetschwellenwert erreicht“ sendet, auf dem aktuellen Prozentsatz des Budgets, der verwendet wurde. Wenn Sie beispielsweise 50$ zu einem Budget von 100$ hinzufügen, das sein Limit erreicht hat, zeigt das nächste Ereignis „Budgetschwellenwert erreicht“ an, dass das Budget bei 75 Prozent liegt.

Im Folgenden finden Sie die Detailfelder für das `Budget Threshold Reached` Ereignis.

Die `detail-type` Felder `source` und sind unten aufgeführt, da sie spezifische Werte für Deadline Cloud-Ereignisse enthalten. Definitionen der anderen Metadatenfelder, die in allen Ereignissen enthalten sind, finden Sie unter Referenz zur Ereignisstruktur im Amazon EventBridge Benutzerhandbuch.

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Budget Threshold Reached",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "budgetId": "budget-12345678900000000000000000000000",
        "thresholdInPercent": 0
    }
}
```

`detail-type`

Identifiziert den Ereignistyp.

Für dieses Ereignis ist dieser Wert`Budget Threshold Reached`.

`source`

Identifiziert den Service, aus dem das Ereignis stammt. Für Deadline Cloud-Ereignisse ist dieser Wert`aws.deadline`.

`detail`

Ein JSON-Objekt, das Informationen zum Ereignis enthält. Der Service, der das Ereignis generiert, bestimmt den Inhalt dieses Feldes.

Für dieses Ereignis beinhalten diese Daten:

`farmId`

Die ID der Farm, die den Job enthält.

`budgetId`

Die Kennung des Budgets, das einen Schwellenwert erreicht hat.

`thresholdInPercent`

Der Prozentsatz des Budgets, der verwendet wurde.

Empfehlung zur Flottengröße: Ereignis ändern

Wenn Sie Ihre Flotte so konfigurieren, dass sie ereignisbasiertes Auto Scaling verwendet, sendet Deadline Cloud Ereignisse, mit denen Sie Ihre Flotten verwalten können. Jedes dieser Ereignisse enthält Informationen über die aktuelle Größe und die angeforderte Größe einer Flotte. Ein Beispiel für die Verwendung eines EventBridge Ereignisses und eine Lambda-Beispielfunktion zur Behandlung des Ereignisses finden Sie unter Automatische Skalierung Ihrer Amazon EC2 EC2-Flotte mit der Deadline Cloud-Skalierungsempfehlungsfunktion.

Das Ereignis zur Änderung der Empfehlung zur Flottengröße wird gesendet, wenn Folgendes eintritt:

- Wenn sich die empfohlene Flottengröße ändert und oldFleetSize sich von unterscheidet newFleetSize.
- Wenn der Service feststellt, dass die tatsächliche Flottengröße nicht der empfohlenen Flottengröße entspricht. Die tatsächliche Flottengröße können Sie dem WorkerCount in der GetFleet Betriebsantwort entnehmen. Dies kann passieren, wenn sich eine aktive Amazon EC2 EC2-Instance nicht als Deadline Cloud-Worker registrieren kann.

Im Folgenden finden Sie die Detailfelder für die `Fleet Size Recommendation Change` Veranstaltung.

Die `detail-type` Felder `source` und sind unten aufgeführt, da sie spezifische Werte für Deadline Cloud-Ereignisse enthalten. Definitionen der anderen Metadatenfelder, die in allen

Ereignissen enthalten sind, finden Sie unter Referenz zur Ereignisstruktur im Amazon EventBridge Benutzerhandbuch.

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Fleet Size Recommendation Change",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "fleetId": "fleet-12345678900000000000000000000000",
        "oldFleetSize": 1,
        "newFleetSize": 5,
    }
}
```

`detail-type`

Identifiziert den Ereignistyp.

Für dieses Ereignis ist dieser Wert`Fleet Size Recommendation Change`.

`source`

Identifiziert den Service, aus dem das Ereignis stammt. Für Deadline Cloud-Ereignisse ist dieser Wert`aws.deadline`.

`detail`

Ein JSON-Objekt, das Informationen zum Ereignis enthält. Der Service, der das Ereignis generiert, bestimmt den Inhalt dieses Feldes.

Für dieses Ereignis beinhalten diese Daten:

`farmId`

Die ID der Farm, die den Job enthält.

`fleetId`

Die Kennung der Flotte, für die eine Größenänderung erforderlich ist.

`oldFleetSize`

Die aktuelle Größe der Flotte.

`newFleetSize`

Die empfohlene neue Größe für die Flotte.

Ereignis zur Änderung des Joblebenszyklusstatus

Wenn Sie einen Job erstellen oder aktualisieren, legt Deadline Cloud den Lebenszyklusstatus so fest, dass der Status der letzten vom Benutzer initiierten Aktion angezeigt wird.

Bei jeder Änderung des Lebenszyklusstatus wird ein Ereignis zur Änderung des Joblebenszyklusstatus gesendet, auch wenn der Job erstellt wird.

Im Folgenden finden Sie die Detailfelder für das `Job Lifecycle Status Change` Ereignis.

Die `detail-type` Felder `source` und sind unten aufgeführt, da sie spezifische Werte für Deadline Cloud-Ereignisse enthalten. Definitionen der anderen Metadatenfelder, die in allen Ereignissen enthalten sind, finden Sie unter Referenz zur Ereignisstruktur im Amazon EventBridge Benutzerhandbuch.

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Job Lifecycle Status Change",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "queueId": "queue-12345678900000000000000000000000",
        "jobId": "job-12345678900000000000000000000000",
        "previousLifecycleStatus": "UPDATE_IN_PROGRESS",
        "lifecycleStatus": "UPDATE_SUCCEEDED"
    }
}
```

`detail-type`

Identifiziert den Ereignistyp.

Für dieses Ereignis ist dieser Wert`Job Lifecycle Status Change`.

`source`

Identifiziert den Service, aus dem das Ereignis stammt. Für Deadline Cloud-Ereignisse ist dieser Wert`aws.deadline`.

`detail`

Ein JSON-Objekt, das Informationen zum Ereignis enthält. Der Service, der das Ereignis generiert, bestimmt den Inhalt dieses Feldes.

Für dieses Ereignis beinhalten diese Daten:

`farmId`

Die ID der Farm, die den Job enthält.

`queueId`

Die ID der Warteschlange, die den Job enthält.

`jobId`

Die ID des Jobs.

`previousLifecycleStatus`

Der Lebenszyklusstatus, in dem der Job beendet wird. Dieses Feld ist nicht enthalten, wenn Sie einen Job zum ersten Mal einreichen.

`lifecycleStatus`

Der Lebenszyklusstatus, in den der Job eintritt.

Ereignis zur Änderung des Status der Auftragsausführung

Ein Job besteht aus vielen Aufgaben. Jede Aufgabe hat einen Status. Der Status aller Aufgaben wird zusammengefasst, um einen Gesamtstatus für einen Job zu erhalten. Weitere Informationen finden Sie unter Job in Deadline Cloud im AWS Deadline Cloud-Benutzerhandbuch.

Ein Ereignis zur Änderung des Status der Auftragsausführung wird gesendet, wenn:

- Das kombinierte `taskRunStatus` Feld ändert sich.
- Der Job wird in die Warteschlange gestellt, es sei denn, der Job befindet sich im Status READY.

Ein Ereignis zur Änderung des Status der Auftragsausführung wird NICHT gesendet, wenn:

- Der Job wird zuerst erstellt. Um die Auftragserstellung zu überwachen, überprüfen Sie die Ereignisse zur Änderung des Joblebenszyklusstatus auf Änderungen.
- Das `taskRunStatusCounts` Feld des Jobs ändert sich, aber der Ausführungsstatus der kombinierten Jobaufgabe ändert sich nicht.

Im Folgenden finden Sie die Detailfelder für das `Job Run Status Change` Ereignis.

Die `detail-type` Felder `source` und sind unten aufgeführt, da sie spezifische Werte für Deadline Cloud-Ereignisse enthalten. Definitionen der anderen Metadatenfelder, die in allen Ereignissen enthalten sind, finden Sie unter Referenz zur Ereignisstruktur im Amazon EventBridge Benutzerhandbuch.

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Job Run Status Change",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "queueId": "queue-12345678900000000000000000000000",
        "jobId": "job-12345678900000000000000000000000",
        "previousTaskRunStatus": "RUNNING",
        "taskRunStatus": "SUCCEEDED",
        "taskRunStatusCounts": {
            "PENDING": 0,
            "READY": 0,
            "RUNNING": 0,
            "ASSIGNED": 0,
            "STARTING": 0,
            "SCHEDULED": 0,
            "INTERRUPTING": 0,
```

```
            "SUSPENDED": 0,
            "CANCELED": 0,
            "FAILED": 0,
            "SUCCEEDED": 20,
            "NOT_COMPATIBLE": 0
        }
    }
}
```

`detail-type`

Identifiziert den Ereignistyp.

Für dieses Ereignis ist dieser Wert`Job Run Status Change`.

`source`

Identifiziert den Service, aus dem das Ereignis stammt. Für Deadline Cloud-Ereignisse ist dieser Wert`aws.deadline`.

`detail`

Ein JSON-Objekt, das Informationen zum Ereignis enthält. Der Service, der das Ereignis generiert, bestimmt den Inhalt dieses Feldes.

Für dieses Ereignis beinhalten diese Daten:

`farmId`

Die ID der Farm, die den Job enthält.

`queueId`

Die ID der Warteschlange, die den Job enthält.

`jobId`

Die ID des Jobs.

`previousTaskRunStatus`

Der Status, in dem die Aufgabe ausgeführt wird, gibt an, dass der Job beendet wird.

`taskRunStatus`

Der Status der Aufgabenausführung, in den der Job wechselt.

`taskRunStatusCounts`

Die Anzahl der Aufgaben des Jobs in jedem Status.

Schritt: Ereignis „Lebenszyklusstatus ändern“

Wenn Sie ein Ereignis erstellen oder aktualisieren, legt Deadline Cloud den Lebenszyklusstatus des Jobs fest, um den Status der letzten vom Benutzer initiierten Aktion zu beschreiben.

Ein Ereignis zur Änderung des Schrittlebenszyklusstatus wird gesendet, wenn:

- Eine Schrittaktualisierung wird gestartet (UPDATE_IN_PROGRESS).
- Ein schrittweises Update wurde erfolgreich abgeschlossen (UPDATE_SUCCEED).
- Ein schrittweises Update ist fehlgeschlagen (UPDATE_FAILED).

Bei der ersten Erstellung des Schritts wird kein Ereignis gesendet. Um die Erstellung von Schritten zu überwachen, überprüfen Sie die Ereignisse zur Änderung des Joblebenszyklusstatus auf Änderungen.

Im Folgenden finden Sie die Detailfelder für das `Step Lifecycle Status Change` Ereignis.

Die `detail-type` Felder `source` und sind unten aufgeführt, da sie spezifische Werte für Deadline Cloud-Ereignisse enthalten. Definitionen der anderen Metadatenfelder, die in allen Ereignissen enthalten sind, finden Sie unter Referenz zur Ereignisstruktur im Amazon EventBridge Benutzerhandbuch.

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Step Lifecycle Status Change",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "queueId": "queue-12345678900000000000000000000000",
        "jobId": "job-12345678900000000000000000000000",
        "stepId": "step-12345678900000000000000000000000",
```

```
        "previousLifecycleStatus": "UPDATE_IN_PROGRESS",
        "lifecycleStatus": "UPDATE_SUCCEEDED"
    }
}
```

`detail-type`

Identifiziert den Ereignistyp.

Für dieses Ereignis ist dieser Wert`Step Lifecycle Status Change`.

`source`

Identifiziert den Service, aus dem das Ereignis stammt. Für Deadline Cloud-Ereignisse ist dieser Wert`aws.deadline`.

`detail`

Ein JSON-Objekt, das Informationen zum Ereignis enthält. Der Service, der das Ereignis generiert, bestimmt den Inhalt dieses Feldes.

Für dieses Ereignis beinhalten diese Daten:

`farmId`

Die ID der Farm, die den Job enthält.

`queueId`

Die ID der Warteschlange, die den Job enthält.

`jobId`

Die ID des Jobs.

`stepId`

Die Kennung des aktuellen Auftragsschritts.

`previousLifecycleStatus`

Der Lebenszyklusstatus, den der Schritt verlässt.

`lifecycleStatus`

Der Lebenszyklusstatus, in den der Schritt eintritt.

Schritt: Ereignis „Status ändern“ ausführen

Jeder Schritt in einem Job besteht aus vielen Aufgaben. Jede Aufgabe hat einen Status. Die Aufgabenstatus werden kombiniert, um einen Gesamtstatus für Schritte und Jobs zu ergeben.

Ein Ereignis zur Änderung des Status eines Schrittlaufs wird gesendet, wenn:

- Die kombinierten `taskRunStatus` Änderungen.
- Der Schritt wird in eine Warteschlange gestellt, sofern sich dieser Schritt nicht im Status READY befindet.

Ein Ereignis wird nicht gesendet, wenn:

- Der Schritt wird zuerst erstellt. Um die Erstellung von Schritten zu überwachen, überprüfen Sie die Ereignisse zur Änderung des Joblebenszyklusstatus auf Änderungen.
- Der Schritt `taskRunStatusCounts` ändert sich, aber der Ausführungsstatus der kombinierten Schrittaufgabe ändert sich nicht.

Im Folgenden finden Sie die Detailfelder für das `Step Run Status Change` Ereignis.

Die `detail-type` Felder `source` und sind unten aufgeführt, da sie spezifische Werte für Deadline Cloud-Ereignisse enthalten. Definitionen der anderen Metadatenfelder, die in allen Ereignissen enthalten sind, finden Sie unter Referenz zur Ereignisstruktur im Amazon EventBridge Benutzerhandbuch.

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Step Run Status Change",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "queueId": "queue-12345678900000000000000000000000",
        "jobId": "job-12345678900000000000000000000000",
        "stepId": "step-12345678900000000000000000000000",
        "previousTaskRunStatus": "RUNNING",
```

```
        "taskRunStatus": "SUCCEEDED",
        "taskRunStatusCounts": {
            "PENDING": 0,
            "READY": 0,
            "RUNNING": 0,
            "ASSIGNED": 0,
            "STARTING": 0,
            "SCHEDULED": 0,
            "INTERRUPTING": 0,
            "SUSPENDED": 0,
            "CANCELED": 0,
            "FAILED": 0,
            "SUCCEEDED": 20,
            "NOT_COMPATIBLE": 0
        }
    }
}
```

`detail-type`

Identifiziert den Ereignistyp.

Für dieses Ereignis ist dieser Wert`Step Run Status Change`.

`source`

Identifiziert den Service, aus dem das Ereignis stammt. Für Deadline Cloud-Ereignisse ist dieser Wert`aws.deadline`.

`detail`

Ein JSON-Objekt, das Informationen zum Ereignis enthält. Der Service, der das Ereignis generiert, bestimmt den Inhalt dieses Feldes.

Für dieses Ereignis beinhalten diese Daten:

`farmId`

Die ID der Farm, die den Job enthält.

`queueId`

Die ID der Warteschlange, die den Job enthält.

`jobId`

Die ID des Jobs.

`stepId`

Die Kennung des aktuellen Auftragsschritts.

`previousTaskRunStatus`

Der Ausführungsstatus, den der Schritt verlässt.

`taskRunStatus`

Der Ausführungsstatus, in den der Schritt eintritt.

`taskRunStatusCounts`

Die Anzahl der Aufgaben des Schritts in jedem Status.

Ereignis „Status der Aufgabe ausführen“

Das `runStatus` Feld wird aktualisiert, wenn die Aufgabe ausgeführt wird. Ein Ereignis wird gesendet, wenn:

- Der Ausführungsstatus der Aufgabe ändert sich.
- Die Aufgabe wird in die Warteschlange gestellt, es sei denn, die Aufgabe befindet sich im Status BEREIT.

Ein Ereignis wird nicht gesendet, wenn:

- Die Aufgabe wird zuerst erstellt. Um die Aufgabenerstellung zu überwachen, überprüfen Sie die Ereignisse zur Änderung des Joblebenszyklusstatus auf Änderungen.

Im Folgenden finden Sie die Detailfelder für das `Task Run Status Change` Ereignis.

Die `detail-type` Felder `source` und sind unten aufgeführt, da sie spezifische Werte für Deadline Cloud-Ereignisse enthalten. Definitionen der anderen Metadatenfelder, die in allen Ereignissen enthalten sind, finden Sie unter Referenz zur Ereignisstruktur im Amazon EventBridge Benutzerhandbuch.

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Task Run Status Change",
```

```
    "source": "aws.aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "queueId": "queue-12345678900000000000000000000000",
        "jobId": "job-12345678900000000000000000000000",
        "stepId": "step-12345678900000000000000000000000",
        "taskId": "task-12345678900000000000000000000000-0",
        "previousRunStatus": "RUNNING",
        "runStatus": "SUCCEEDED"
    }
}
```

`detail-type`

Identifiziert den Ereignistyp.

Für dieses Ereignis ist dieser Wert`Fleet Size Recommendation Change`.

`source`

Identifiziert den Service, aus dem das Ereignis stammt. Für Deadline Cloud-Ereignisse ist dieser Wert`aws.deadline`.

`detail`

Ein JSON-Objekt, das Informationen zum Ereignis enthält. Der Service, der das Ereignis generiert, bestimmt den Inhalt dieses Feldes.

Für dieses Ereignis beinhalten diese Daten:

`farmId`

Die ID der Farm, die den Job enthält.

`queueId`

Die ID der Warteschlange, die den Job enthält.

`jobId`

Die ID des Jobs.

stepId

Die Kennung des aktuellen Job-Schritts.

`taskId`

Die ID der laufenden Aufgabe.

`previousRunStatus`

Der Ausführungsstatus, den die Aufgabe verlässt.

`runStatus`

Der Ausführungsstatus, in den die Aufgabe eintritt.

Abfragen aggregierter Daten mit Sitzungsstatistiken unter Verwendung der AWS CLI

Um Kosten zu verfolgen, die Ressourcennutzung zu analysieren oder festzustellen, welche Benutzer die meisten Ressourcen verbrauchen, können Sie die AWS Command Line Interface (AWS CLI) verwenden, um aggregierte Sitzungsstatistiken für Ihre AWS Deadline Cloud-Farmen (Deadline Cloud) abzufragen. Die Sitzungsstatistik-API bietet Daten zu Kosten, Laufzeit und Nutzung, die Sie nach verschiedenen Dimensionen wie Warteschlange, Flotte, Instanztyp oder Benutzer gruppieren können.

Das Abfragen von Sitzungsstatistiken ist ein asynchroner Prozess. Zuerst starten Sie eine Aggregationsanforderung und rufen dann die Ergebnisse mithilfe der Aggregations-ID ab.

Eine Aggregationsanforderung starten

Um eine Aggregationsanforderung zu starten, führen Sie den `start-sessions-statistics-aggregation` Befehl aus. Im folgenden Beispiel werden Statistiken für eine bestimmte Warteschlange nach Benutzer-ID gruppiert. Ersetzen Sie den *`placeholder`* Text durch Ihre Informationen.

```
aws deadline start-sessions-statistics-aggregation \
    --farm-id farm-id \
    --resource-ids '{"queueIds":["queue-id"]}' \
    --start-time 2025-11-24T10:00:00Z \
    --end-time 2025-11-25T18:00:00Z \
    --group-by '["USER_ID"]' \
    --period HOURLY \
    --statistics '["SUM"]' \
    --timezone UTC-08:00 \
    --region region-name
```

Sie können Statistiken nach anderen Dimensionen wie`QUEUE_ID`,, `FLEET_ID` `JOB_ID`INSTANCE_TYPE, oder gruppierenLICENSE_PRODUCT. Weitere Informationen zu allen verfügbaren Parametern finden Sie start-sessions-statistics-aggregationin der AWS CLI Befehlsreferenz.

Die Antwort enthält eine Aggregations-ID:

```
{
    "aggregationId": "92b35143f2d04641979bc9b777232f38"
}
```

Ergebnisse werden abgerufen

Führen Sie den `get-sessions-statistics-aggregation` Befehl mit der Aggregations-ID aus, um die Ergebnisse abzurufen. Ersetzen Sie den *`placeholder`* Text durch Ihre Informationen.

```
aws deadline get-sessions-statistics-aggregation \
    --farm-id farm-id \
    --aggregation-id aggregation-id \
    --region region-name
```

Das folgende Beispiel zeigt eine Antwort, wenn Sie Statistiken nach Benutzer-ID gruppieren. Das `userId` Feld enthält eine UUID, die Sie einem Benutzernamen zuordnen müssen, um den Benutzer zu identifizieren:

```
{
    "statistics": [
        {
            "userId": "f9c1f3f0-1031-70dc-4d25-30d7225b04a0",
            "count": 1,
            "costInUsd": {
                "sum": 0.0
            },
            "runtimeInSeconds": {
                "sum": 53.773
            },
            "aggregationStartTime": "2025-11-24T22:00:00Z",
            "aggregationEndTime": "2025-11-24T23:00:00Z"
        }
    ],
    "status": "COMPLETED"
}
```

Informationen zum Ermitteln des mit einem verknüpften Nutzernamen finden Sie `userId` unter. the section called “Benutzermetadaten mithilfe von UserID abrufen”

Weitere Informationen zur API finden Sie in der Deadline Cloud API-Referenz.

Topics

- the section called “Benutzermetadaten mithilfe von UserID abrufen”

Benutzermetadaten und Attribute mithilfe von UserID in einem Identitätsspeicher abrufen

> **Note**
>
> Dieses Verfahren gilt auch für das von der SearchJobsAPI zurückgegebene `createdBy` Feld, das dasselbe Benutzer-ID-Format verwendet.

Das `userId` Feld in der Sitzungsstatistik enthält einen der folgenden Werte:

- Ein AWS Identity and Access Management (IAM-) Rollen-ARN, zum Beispiel:`arn:aws:sts::123456789012:assumed-role/Admin/user-Isengard`.
- Eine IAM Identity Center-Benutzer-ID (UUID), zum Beispiel:. `f9c1f3f0-1031-70dc-4d25-30d7225b04a0`

Für die IAM-Rolle ARNs ist der Benutzername im ARN selbst sichtbar. Für IAM Identity Center-Benutzer IDs können Sie den Benutzernamen mithilfe der IAM Identity Center Identity Store-API nachschlagen.

Gehen Sie wie folgt vor, um den mit einer IAM Identity Center-Benutzer-ID verknüpften Benutzernamen zu identifizieren. Bevor Sie beginnen, rufen Sie die Identity Store-ID aus Ihren IAM Identity Center-Einstellungen ab. Weitere Informationen finden Sie unter the section called “Finden Sie Ihre Identity Store-ID”.

Um eine Benutzer-ID zuzuordnen

1. Führen Sie den folgenden Befehl aus und *`IdentityStoreId`* ersetzen Sie ihn durch Ihre Identity Store-ID und *`userUUID`* durch die Antwort `userId` aus der Sitzungsstatistik:

```
aws identitystore describe-user \
    --identity-store-id IdentityStoreId \
    --user-id userUUID
```

2. Überprüfen Sie die Antwort, die den Benutzernamen enthält:

```
{
    "UserName": "jdoe",
    "UserId": "f9c1f3f0-1031-70dc-4d25-30d7225b04a0",
    "Name": {
        "FamilyName": "Doe",
        "GivenName": "Jane"
    },
    "DisplayName": "Jane Doe",
    "Emails": [{
        "Value": "jdoe@example.com",
        "Type": "work",
        "Primary": true
    }],
    "IdentityStoreId": "d-xxxxxxxxxx"
}
```

Finden Sie Ihre Identity Store-ID

Um Benutzer Benutzernamen IDs zuzuordnen, benötigen Sie die Identity Store-ID. Sie finden die Identity Store-ID in der IAM Identity Center-Konsole oder im. AWS CLI

Konsole

Gehen Sie wie folgt vor, um Ihre Identity Store-ID mithilfe der Konsole zu ermitteln.

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die IAM Identity Center-Konsole.
2. Wählen Sie im Navigationsbereich Settings (Einstellungen).
3. Kopieren Sie den IAM Identity Center Identity Store ID-Wert. Das Format ist `d-xxxxxxxxxx`.

AWS CLI

Führen Sie den folgenden Befehl aus und *`region-name`* ersetzen Sie ihn durch die Region, in der Ihre IAM Identity Center-Instanz konfiguriert ist:

```
aws sso-admin list-instances --region region-name
```

Die Antwort beinhaltet: `IdentityStoreId`

```
{
    "Instances": [
        {
            "CreatedDate": "2025-11-19T15:45:55.160000-08:00",
            "IdentityStoreId": "d-xxxxxxxxxx",
            "InstanceArn": "arn:aws:sso:::instance/ssoins-xxxxxxxxxxxxxxx",
            "OwnerAccountId": "123456789012",
            "Status": "ACTIVE"
        }
    ]
}
```

Überprüfung der Benutzerzuweisung

Nachdem Sie einem Benutzernamen eine Benutzer-ID zugeordnet haben, können Sie in der IAM Identity Center-Konsole überprüfen, ob die Benutzer-ID dem erwarteten Benutzer entspricht. Gehen Sie wie folgt vor, um die Benutzerzuordnung zu überprüfen.

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die IAM Identity Center-Konsole.
2. Klicken Sie im Navigationsbereich auf Users (Benutzer).
3. Wählen Sie den Benutzernamen aus der AWS CLI Antwort aus.
4. Vergewissern Sie sich im Abschnitt Allgemeine Informationen, dass die Benutzer-ID mit der `userId` aus Ihren Sitzungsstatistiken übereinstimmt.

Weitere Ressourcen

- IAM Identity Center-Benutzerhandbuch
- Referenz zur IAM Identity Center Identity Store-API

Sicherheit in Deadline Cloud

Cloud-Sicherheit AWS hat höchste Priorität. Als AWS Kunde profitieren Sie von Rechenzentren und Netzwerkarchitekturen, die darauf ausgelegt sind, die Anforderungen der sicherheitssensibelsten Unternehmen zu erfüllen.

Sicherheit ist eine gemeinsame AWS Verantwortung von Ihnen und Ihnen. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, die AWS-Services in der läuft AWS Cloud. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Externe Prüfer testen und verifizieren regelmäßig die Wirksamkeit unserer Sicherheitsmaßnahmen im Rahmen der AWS . Weitere Informationen zu den Compliance-Programmen, die für gelten AWS Deadline Cloud, finden Sie AWS-Services unter Umfang nach Compliance-ProgrammenAWS-Services und unter .
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS-Service , was Sie verwenden. Sie sind auch für andere Faktoren verantwortlich, etwa für die Vertraulichkeit Ihrer Daten, für die Anforderungen Ihres Unternehmens und für die geltenden Gesetze und Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der gemeinsamen Verantwortung bei der Verwendung anwenden können Deadline Cloud. In den folgenden Themen erfahren Sie, wie Sie die Konfiguration vornehmen Deadline Cloud , um Ihre Sicherheits- und Compliance-Ziele zu erreichen. Sie erfahren auch, wie Sie andere verwenden können AWS-Services , die Ihnen bei der Überwachung und Sicherung Ihrer Deadline Cloud Ressourcen helfen.

Themen

- Bewährte Sicherheitsmethoden für Deadline Cloud

Datenschutz in Deadline Cloud

Das Modell der AWS gemeinsamen Verantwortung und geteilter Verantwortung gilt für den Datenschutz in AWS Deadline Cloud. Wie in diesem Modell beschrieben, AWS ist verantwortlich für den Schutz der globalen Infrastruktur, auf der alle Systeme laufen AWS Cloud. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie unter Häufig gestellte Fragen zum Datenschutz. Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag AWS -Modell der geteilten Verantwortung und in der DSGVO im AWS -Sicherheitsblog.

Aus Datenschutzgründen empfehlen wir, dass Sie AWS-Konto Anmeldeinformationen schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Wird verwendet SSL/TLS , um mit AWS Ressourcen zu kommunizieren. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein AWS CloudTrail. Informationen zur Verwendung von CloudTrail Pfaden zur Erfassung von AWS Aktivitäten finden Sie unter Arbeiten mit CloudTrail Pfaden im AWS CloudTrail Benutzerhandbuch.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-3-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-3.

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit der Konsole, der Deadline Cloud API oder auf andere AWS-Services Weise arbeiten oder diese

verwenden. AWS CLI AWS SDKs Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen externen Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

Die in die Namensfelder von Deadline Cloud Jobvorlagen eingegebenen Daten können auch in Abrechnungs- oder Diagnoseprotokollen enthalten sein und sollten keine vertraulichen oder sensiblen Informationen enthalten.

Themen

Verschlüsselung im Ruhezustand

AWS Deadline Cloud schützt sensible Daten, indem sie im Ruhezustand mit den in AWS Key Management Service ()AWS KMS gespeicherten Verschlüsselungsschlüsseln verschlüsselt werden. Verschlüsselung im Ruhezustand ist überall verfügbar, AWS-Regionen wo sie verfügbar Deadline Cloud ist.

Die Verschlüsselung von Daten bedeutet, dass sensible Daten, die auf Festplatten gespeichert sind, für einen Benutzer oder eine Anwendung ohne gültigen Schlüssel nicht lesbar sind. Nur eine Partei mit einem gültigen verwalteten Schlüssel kann die Daten entschlüsseln.

Deadline Cloud löscht Amazon Elastic Block Store-Volumes, wenn servicemanaged Fleet Worker-Instances beendet werden.

Informationen darüber, wie Deadline Cloud Daten im Ruhezustand verschlüsselt werden können, finden Sie unter. AWS KMS Schlüsselverwaltung

Verschlüsselung während der Übertragung

AWS Deadline Cloud Verwendet für Daten während der Übertragung Transport Layer Security (TLS) 1.2 oder 1.3, um Daten zu verschlüsseln, die zwischen dem Dienst und den Workern gesendet

werden. Wir benötigen TLS 1.2 und empfehlen TLS 1.3. Wenn Sie eine Virtual Private Cloud (VPC) verwenden, können Sie darüber hinaus eine private Verbindung zwischen Ihrer VPC und herstellen. AWS PrivateLink Deadline Cloud

Schlüsselverwaltung

Wenn Sie eine neue Farm erstellen, können Sie einen der folgenden Schlüssel zum Verschlüsseln Ihrer Farmdaten wählen:

- AWS eigener KMS-Schlüssel — Standardverschlüsselungstyp, wenn Sie beim Erstellen der Farm keinen Schlüssel angeben. Der KMS-Schlüssel gehört AWS Deadline Cloud. Sie können AWS eigene Schlüssel nicht anzeigen, verwalten oder verwenden. Sie müssen jedoch keine Maßnahmen ergreifen, um die Schlüssel zu schützen, mit denen Ihre Daten verschlüsselt werden. Weitere Informationen finden Sie AWS im AWS Key Management Service Entwicklerhandbuch unter Eigene Schlüssel.
- Kundenverwalteter KMS-Schlüssel — Sie geben einen vom Kunden verwalteten Schlüssel an, wenn Sie eine Farm erstellen. Der gesamte Inhalt der Farm ist mit dem KMS-Schlüssel verschlüsselt. Der Schlüssel wird in Ihrem Konto gespeichert und wird von Ihnen erstellt, gehört und verwaltet. Es AWS KMS fallen Gebühren an. Sie haben die volle Kontrolle über den KMS-Schlüssel. Sie können folgende Aufgaben ausführen:
  - Festlegung und Aufrechterhaltung wichtiger Richtlinien
  - Festlegung und Aufrechterhaltung von IAM-Richtlinien und -Zuschüssen
  - Aktivieren und Deaktivieren wichtiger Richtlinien
  - Hinzufügen von -Tags
  - Erstellen von Schlüsselaliasen

  Sie können einen kundeneigenen Schlüssel, der in einer Deadline Cloud Farm verwendet wird, nicht manuell rotieren. Die automatische Rotation des Schlüssels wird unterstützt.

  Weitere Informationen finden Sie im AWS Key Management Service Entwicklerhandbuch unter Schlüssel, die dem Kunden gehören.

  Um einen vom Kunden verwalteten Schlüssel zu erstellen, folgen Sie den Schritten unter Erstellen symmetrischer kundenverwalteter Schlüssel im AWS Key Management Service Entwicklerhandbuch.

Wie werden Deadline Cloud Zuschüsse verwendet AWS KMS

Deadline Cloud erfordert einen Zuschuss, um Ihren vom Kunden verwalteten Schlüssel verwenden zu können. Wenn Sie eine Farm erstellen, die mit einem vom Kunden verwalteten Schlüssel verschlüsselt ist, Deadline Cloud erstellt das Programm in Ihrem Namen einen Zuschuss, indem es eine `CreateGrant` Anfrage an sendet AWS KMS , um Zugriff auf den von Ihnen angegebenen KMS-Schlüssel zu erhalten.

Deadline Cloud verwendet mehrere Zuschüsse. Jeder Grant wird von einem anderen Teil verwendet Deadline Cloud , der Ihre Daten ver- oder entschlüsseln muss. Deadline Cloud verwendet auch Zuschüsse, um den Zugriff auf andere AWS Dienste zu ermöglichen, die zum Speichern von Daten in Ihrem Namen verwendet werden, wie Amazon Simple Storage Service, Amazon Elastic Block Store oder OpenSearch.

Zuschüsse, die Deadline Cloud die Verwaltung von Maschinen in einer vom `GranteePrincipal` Service verwalteten Flotte ermöglichen, beinhalten eine Deadline Cloud Kontonummer und eine Rolle als Service Principal. Dies ist zwar nicht üblich, aber notwendig, um Amazon EBS-Volumes für Mitarbeiter in serviceverwalteten Flotten mit dem für die Farm angegebenen kundenverwalteten KMS-Schlüssel zu verschlüsseln.

Kundenseitig verwaltete CMK-Schlüsselrichtlinie

Schlüsselrichtlinien steuern den Zugriff auf den vom Kunden verwalteten Schlüssel. Jeder Schlüssel muss über genau eine Schlüsselrichtlinie verfügen, die Aussagen enthält, die festlegen, wer den Schlüssel verwenden darf und wie er verwendet werden darf. Wenn Sie Ihren vom Kunden verwalteten Schlüssel erstellen, können Sie eine Schlüsselrichtlinie angeben. Weitere Informationen finden Sie unter Verwalten des Zugriffs auf kundenseitig verwaltete Schlüssel im Entwicklerhandbuch zum AWS Key Management Service .

Minimale IAM-Richtlinie für CreateFarm

Um Ihren vom Kunden verwalteten Schlüssel zum Erstellen von Farmen mithilfe der Konsole oder des `CreateFarm` API-Vorgangs zu verwenden, müssen die folgenden AWS KMS API-Operationen zugelassen sein:

- `kms:CreateGrant`: Fügt einem kundenverwalteten Schlüssel eine Erteilung hinzu. Gewährt Konsolenzugriff auf einen angegebenen AWS KMS Schlüssel. Weitere Informationen finden Sie im AWS Key Management Service Entwicklerhandbuch unter Using Grants.
- `kms:Decrypt`— Ermöglicht Deadline Cloud das Entschlüsseln von Daten in der Farm.

- kms:DescribeKey— Stellt dem Kunden verwaltete Schlüsseldetails zur Verfügung, damit Deadline Cloud der Schlüssel validiert werden kann.
- kms:GenerateDataKey— Ermöglicht Deadline Cloud die Verschlüsselung von Daten mit einem eindeutigen Datenschlüssel.

Die folgende Richtlinienerklärung gewährt die erforderlichen Berechtigungen für den `CreateFarm` Vorgang.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DeadlineCreateGrants",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey",
                "kms:CreateGrant",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/1234567890abcdef0",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "deadline.us-west-2.amazonaws.com"
                }
            }
        }
    ]
}
```

Minimale IAM-Richtlinie für schreibgeschützte Operationen

Um Ihren vom Kunden verwalteten Schlüssel für schreibgeschützte Deadline Cloud Operationen zu verwenden, z. B. für das Abrufen von Informationen über Farmen, Warteschlangen und Flotten. Die folgenden AWS KMS API-Operationen müssen zulässig sein:

- kms:Decrypt— Ermöglicht Deadline Cloud das Entschlüsseln von Daten in der Farm.
- kms:DescribeKey— Stellt dem Kunden verwaltete Schlüsseldetails zur Verfügung, damit Deadline Cloud der Schlüssel validiert werden kann.

Die folgende Richtlinienerklärung gewährt die erforderlichen Berechtigungen für schreibgeschützte Operationen.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DeadlineReadOnly",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "deadline.us-west-2.amazonaws.com"
                }
            }
        }
    ]
}
```

Minimale IAM-Richtlinie für Lese- und Schreibvorgänge

Um Ihren vom Kunden verwalteten Schlüssel für Lese- und Deadline Cloud Schreibvorgänge wie das Erstellen und Aktualisieren von Farmen, Warteschlangen und Flotten zu verwenden. Die folgenden AWS KMS API-Operationen müssen zulässig sein:

- kms:Decrypt— Ermöglicht Deadline Cloud das Entschlüsseln von Daten in der Farm.
- kms:DescribeKey— Stellt dem Kunden verwaltete Schlüsseldetails zur Verfügung, damit Deadline Cloud der Schlüssel validiert werden kann.

- kms:GenerateDataKey— Ermöglicht Deadline Cloud die Verschlüsselung von Daten mit einem eindeutigen Datenschlüssel.

Die folgende Richtlinienerklärung gewährt die erforderlichen Berechtigungen für den `CreateFarm` Vorgang.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DeadlineReadWrite",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "deadline.us-west-2.amazonaws.com"
                }
            }
        }
    ]
}
```

Überwachen Ihrer Verschlüsselungsschlüssel

Wenn Sie einen vom AWS KMS Kunden verwalteten Schlüssel für Ihre Deadline Cloud Farmen verwenden, können Sie Amazon CloudWatch Logs verwenden AWS CloudTrail, um Anfragen zu verfolgen, die Deadline Cloud an gesendet AWS KMS werden.

CloudTrail Veranstaltung für Zuschüsse

Das folgende CloudTrail Beispielereignis tritt ein, wenn Zuschüsse erstellt werden, in der Regel, wenn Sie die `CreateFleet` Operation `CreateFarmCreateMonitor`, oder aufrufen.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:SampleUser01",
        "arn": "arn:aws::sts::111122223333:assumed-role/Admin/SampleUser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws::iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2024-04-23T02:05:26Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "deadline.amazonaws.com"
    },
    "eventTime": "2024-04-23T02:05:35Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "deadline.amazonaws.com",
    "userAgent": "deadline.amazonaws.com",
    "requestParameters": {
        "operations": [
            "CreateGrant",
            "Decrypt",
            "DescribeKey",
            "Encrypt",
            "GenerateDataKey"
        ],
        "constraints": {
            "encryptionContextSubset": {
                "aws:deadline:farmId": "farm-abcdef1234567890098765432 1fedcba",
                "aws:deadline:accountId": "111122223333"
            }
```

```
        },
        "granteePrincipal": "deadline.amazonaws.com",
        "keyId": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
        "retiringPrincipal": "deadline.amazonaws.com"
    },
    "responseElements": {
        "grantId": "6bbe819394822a400fe5e3a75d0e9ef16c1733143fff0c1fc00dc7ac282a18a0",
        "keyId": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    },
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "readOnly": false,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE44444"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```


CloudTrail Ereignis für die Entschlüsselung

Das folgende CloudTrail Beispielereignis tritt ein, wenn Werte mithilfe des vom Kunden verwalteten KMS-Schlüssels entschlüsselt werden.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:SampleUser01",
        "arn": "arn:aws::sts::111122223333:assumed-role/SampleRole/SampleUser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
```

```
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws::iam::111122223333:role/SampleRole",
                "accountId": "111122223333",
                "userName": "SampleRole"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2024-04-23T18:46:51Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "deadline.amazonaws.com"
    },
    "eventTime": "2024-04-23T18:51:44Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "deadline.amazonaws.com",
    "userAgent": "deadline.amazonaws.com",
    "requestParameters": {
        "encryptionContext": {
            "aws:deadline:farmId": "farm-abcdef12345678900987654321fedcba",
            "aws:deadline:accountId": "111122223333",
            "aws-crypto-public-key": "AotL+SAMPLEVALUEiOMEXAMPLEaaqNOTREALaGTESTONLY
+p/5H+EuKd4Q=="
        },
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    },
    "responseElements": null,
    "requestID": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeffffff",
    "eventID": "ffffffff-eeee-dddd-cccc-bbbbbbaaaaaa",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        }
    ],
    "eventType": "AwsApiCall",
```

```
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

CloudTrail Ereignis für die Verschlüsselung

Das folgende CloudTrail Beispielereignis tritt ein, wenn Werte mit dem vom Kunden verwalteten KMS-Schlüssel verschlüsselt werden.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:SampleUser01",
        "arn": "arn:aws::sts::111122223333:assumed-role/SampleRole/SampleUser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws::iam::111122223333:role/SampleRole",
                "accountId": "111122223333",
                "userName": "SampleRole"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2024-04-23T18:46:51Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "deadline.amazonaws.com"
    },
    "eventTime": "2024-04-23T18:52:40Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "deadline.amazonaws.com",
    "userAgent": "deadline.amazonaws.com",
    "requestParameters": {
        "numberOfBytes": 32,
        "encryptionContext": {
```

```
            "aws:deadline:farmId": "farm-abcdef12345678900987654321fedcba",
            "aws:deadline:accountId": "111122223333",
            "aws-crypto-public-key": "AotL+SAMPLEVALUEiOMEXAMPLEaaqNOTREALaGTESTONLY
+p/5H+EuKd4Q=="
        },
        "keyId": "arn:aws::kms:us-
west-2:111122223333:key/abcdef12-3456-7890-0987-654321fedcba"
    },
    "responseElements": null,
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE33333"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```


Löschen eines vom Kunden verwalteten KMS-Schlüssels

Das Löschen eines vom Kunden verwalteten KMS-Schlüssels in AWS Key Management Service (AWS KMS) ist destruktiv und potenziell gefährlich. Dadurch werden das Schlüsselmaterial und alle mit dem Schlüssel verknüpften Metadaten unwiderruflich gelöscht. Nachdem ein vom Kunden verwalteter KMS-Schlüssel gelöscht wurde, können Sie die mit diesem Schlüssel verschlüsselten Daten nicht mehr entschlüsseln. Das Löschen des Schlüssels bedeutet, dass die Daten nicht mehr wiederhergestellt werden können.

Aus diesem Grund AWS KMS haben Kunden eine Wartezeit von bis zu 30 Tagen, bevor der KMS-Schlüssel gelöscht wird. Die Standardwartezeit beträgt 30 Tage.

Über die Wartezeit

Da das Löschen eines vom Kunden verwalteten KMS-Schlüssels zerstörerisch und potenziell gefährlich ist, müssen Sie eine Wartezeit von 7—30 Tagen festlegen. Die Standardwartezeit beträgt 30 Tage.

Die tatsächliche Wartezeit kann jedoch bis zu 24 Stunden länger sein als der von Ihnen geplante Zeitraum. Verwenden Sie den DescribeKeyVorgang, um das tatsächliche Datum und die Uhrzeit der Löschung des Schlüssels zu ermitteln. Sie können das geplante Löschdatum eines Schlüssels auch in der AWS KMS Konsole auf der Detailseite des Schlüssels im Abschnitt Allgemeine Konfiguration sehen. Beachten Sie die Zeitzone.

Während der Wartezeit lautet der Status und der Schlüsselstatus des vom Kunden verwalteten Schlüssels „Ausstehende Löschung“.

- Ein vom Kunden verwalteter KMS-Schlüssel, dessen Löschung aussteht, kann für keine kryptografischen Operationen verwendet werden.
- AWS KMS rotiert nicht die Backing-Schlüssel von vom Kunden verwalteten KMS-Schlüsseln, deren Löschung noch aussteht.

Weitere Informationen zum Löschen eines vom Kunden verwalteten KMS-Schlüssels finden Sie unter Löschen von Kundenhauptschlüsseln im AWS Key Management Service Entwicklerhandbuch.

Datenschutz für den Datenverkehr zwischen Netzwerken

AWS Deadline Cloud unterstützt Amazon Virtual Private Cloud (Amazon VPC) zur Sicherung von Verbindungen. Amazon VPC bietet Funktionen, mit denen Sie die Sicherheit Ihrer Virtual Private Cloud (VPC) erhöhen und überwachen können.

Sie können eine vom Kunden verwaltete Flotte (CMF) mit Amazon Elastic Compute Cloud (Amazon EC2) -Instances einrichten, die in einer VPC ausgeführt werden. Durch die Bereitstellung von Amazon VPC-Endpunkten zur Nutzung AWS PrivateLink bleibt der Datenverkehr zwischen Workern in Ihrem CMF und dem Deadline Cloud Endpunkt innerhalb Ihrer VPC. Darüber hinaus können Sie Ihre VPC so konfigurieren, dass der Internetzugang auf Ihre Instances beschränkt wird.

In serviceverwalteten Flotten sind die Mitarbeiter nicht über das Internet erreichbar, aber sie haben Internetzugang und stellen über das Internet eine Verbindung zum Deadline Cloud Service her. Jede Flotte mit Servicemanagement wird in einem eigenen isolierten Netzwerk betrieben, und die Worker-Instances bleiben für einzelne Kunden reserviert.

Abmelden

AWS Deadline Cloud sammelt bestimmte Betriebsinformationen, um uns bei der Entwicklung und Verbesserung zu Deadline Cloud unterstützen. Zu den gesammelten Daten gehören Dinge wie Ihre AWS Konto-ID und Benutzer-ID, sodass wir Sie korrekt identifizieren können, falls Sie ein Problem mit der haben Deadline Cloud. Wir erfassen auch Deadline Cloud spezifische Informationen wie Ressourcen IDs (eine FarmID oder QueueID, falls zutreffend), den Produktnamen (z. B. JobAttachments WorkerAgent, und mehr) und die Produktversion.

Sie können diese Datenerfassung mithilfe der Anwendungskonfiguration deaktivieren. Jeder Computer Deadline Cloud, mit dem sowohl Client-Workstations als auch Flottenmitarbeiter interagiert, muss sich separat abmelden.

Deadline Cloud Monitor — Desktop

Deadline Cloud monitor — desktop sammelt Betriebsinformationen, z. B. wann Abstürze auftreten und wann die Anwendung geöffnet wird, damit wir wissen, wenn Sie Probleme mit der Anwendung haben. Um die Erfassung dieser Betriebsinformationen zu deaktivieren, deaktivieren Sie auf der Einstellungsseite die Option Datenerfassung aktivieren, um die Leistung von Deadline Cloud Monitor zu messen.

Nach der Abmeldung sendet der Desktop-Monitor keine Betriebsdaten mehr. Alle zuvor gesammelten Daten werden gespeichert und können weiterhin zur Verbesserung des Dienstes verwendet werden. Weitere Informationen finden Sie in den Häufig gestellten Fragen zum Datenschutz.

AWS Deadline Cloud CLI und Tools

Die AWS Deadline Cloud CLI, die Einreicher und der Worker Agent sammeln alle Betriebsinformationen, z. B. wann Abstürze auftreten und wann Jobs eingereicht werden, damit wir wissen, wenn Sie Probleme mit diesen Anwendungen haben. Verwenden Sie eine der folgenden Methoden, um sich von der Erfassung dieser Betriebsinformationen abzumelden:

- Geben Sie im Terminal ein**`deadline config set telemetry.opt_out true`**.

  Dadurch werden die CLI, die Einreicher und der Worker-Agent deaktiviert, wenn sie als aktueller Benutzer ausgeführt werden.
- Fügen Sie bei der Installation des Deadline Cloud Worker-Agenten das **`--telemetry-opt-out`** Befehlszeilenargument hinzu. Beispiel, **`./install.sh --farm-id $FARM_ID --fleet-id $FLEET_ID --telemetry-opt-out`**.

- Bevor Sie den Worker-Agent, die CLI oder den Submitter ausführen, legen Sie eine Umgebungsvariable fest: **`DEADLINE_CLOUD_TELEMETRY_OPT_OUT=true`**

Nach dem Abmelden senden die Deadline Cloud Tools keine Betriebsdaten mehr. Alle zuvor gesammelten Daten werden gespeichert und können weiterhin zur Verbesserung des Dienstes verwendet werden. Weitere Informationen finden Sie in den Häufig gestellten Fragen zum Datenschutz.

Identity and Access Management in Deadline Cloud

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service , den Zugriff auf Ressourcen sicher zu AWS kontrollieren. IAM-Administratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), um Deadline Cloud-Ressourcen zu verwenden. IAM ist ein Programm AWS-Service , das Sie ohne zusätzliche Kosten nutzen können.

Themen

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von Ihrer Rolle ab:

- Servicebenutzer – Fordern Sie von Ihrem Administrator Berechtigungen an, wenn Sie nicht auf Features zugreifen können (siehe Fehlerbehebung bei Identität und Zugriff auf AWS Deadline Cloud).

- Serviceadministrator – Bestimmen Sie den Benutzerzugriff und stellen Sie Berechtigungsanfragen (siehe So funktioniert Deadline Cloud mit IAM).
- IAM-Administrator – Schreiben Sie Richtlinien zur Zugriffsverwaltung (siehe Beispiele für identitätsbasierte Richtlinien für Deadline Cloud).

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen sich als IAM-Benutzer authentifizieren oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich als föderierte Identität anmelden, indem Sie Anmeldeinformationen aus einer Identitätsquelle wie AWS IAM Identity Center (IAM Identity Center), Single Sign-On-Authentifizierung oder Anmeldeinformationen verwenden. Google/Facebook Weitere Informationen zum Anmelden finden Sie unter So melden Sie sich bei Ihrem AWS-Konto an im Benutzerhandbuch für AWS-Anmeldung .

AWS Bietet für den programmatischen Zugriff ein SDK und eine CLI zum kryptografischen Signieren von Anfragen. Weitere Informationen finden Sie unter AWS Signature Version 4 for API requests im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, dem sogenannten AWS-Konto Root-Benutzer, der vollständigen Zugriff auf alle AWS-Services Ressourcen hat. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Eine Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Tasks that require root user credentials im IAM-Benutzerhandbuch.

Verbundidentität

Es hat sich bewährt, dass menschliche Benutzer für den Zugriff AWS-Services mithilfe temporärer Anmeldeinformationen einen Verbund mit einem Identitätsanbieter verwenden müssen.

Eine föderierte Identität ist ein Benutzer aus Ihrem Unternehmensverzeichnis, Ihrem Directory Service Web-Identitätsanbieter oder der AWS-Services mithilfe von Anmeldeinformationen aus einer Identitätsquelle zugreift. Verbundene Identitäten übernehmen Rollen, die temporäre Anmeldeinformationen bereitstellen.

Für die zentrale Zugriffsverwaltung empfehlen wir AWS IAM Identity Center. Weitere Informationen finden Sie unter Was ist IAM Identity Center? im AWS IAM Identity Center -Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität mit bestimmten Berechtigungen für eine einzelne Person oder Anwendung. Wir empfehlen die Verwendung temporärer Anmeldeinformationen anstelle von IAM-Benutzern mit langfristigen Anmeldeinformationen. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Erfordern, dass menschliche Benutzer den Verbund mit einem Identitätsanbieter verwenden müssen, um AWS mithilfe temporärer Anmeldeinformationen darauf zugreifen zu können.

Eine IAM-Gruppe spezifiziert eine Sammlung von IAM-Benutzern und erleichtert die Verwaltung von Berechtigungen für große Gruppen von Benutzern. Weitere Informationen finden Sie unter Anwendungsfälle für IAM-Benutzer im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität mit spezifischen Berechtigungen, die temporäre Anmeldeinformationen bereitstellt. Sie können eine Rolle übernehmen, indem Sie von einer Benutzer- zu einer IAM-Rolle (Konsole) wechseln oder indem Sie eine AWS Oder-API-Operation AWS CLI aufrufen. Weitere Informationen finden Sie unter Methoden, um eine Rolle zu übernehmen im IAM-Benutzerhandbuch.

IAM-Rollen sind nützlich für den Verbundbenutzer-Zugriff, temporäre IAM-Benutzerberechtigungen, kontoübergreifenden Zugriff, serviceübergreifenden Zugriff und Anwendungen, die auf Amazon EC2 laufen. Weitere Informationen finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie definiert Berechtigungen, wenn sie mit einer Identität oder Ressource verknüpft sind. AWS bewertet diese Richtlinien, wenn ein Principal eine Anfrage stellt. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Mit Hilfe von Richtlinien legen Administratoren fest, wer Zugriff auf was hat, indem sie definieren, welches Prinzipal welche Aktionen auf welchen Ressourcenund unter welchen Bedingungendurchführen darf.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator erstellt IAM-Richtlinien und fügt sie zu Rollen hinzu, die die Benutzer dann übernehmen können. IAM-Richtlinien definieren Berechtigungen unabhängig von der Methode, die zur Ausführung der Operation verwendet wird.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität (Benutzer, Gruppe oder Rolle) anfügen können. Diese Richtlinien steuern, welche Aktionen Identitäten für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können Inline-Richtlinien (direkt in eine einzelne Identität eingebettet) oder verwaltete Richtlinien (eigenständige Richtlinien, die mit mehreren Identitäten verbunden sind) sein. Informationen dazu, wie Sie zwischen verwalteten und Inline-Richtlinien wählen, finden Sie unter Choose between managed policies and inline policies im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele hierfür sind Vertrauensrichtlinien für IAM-Rollen und Amazon S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben.

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Weitere Richtlinientypen

AWS unterstützt zusätzliche Richtlinientypen, mit denen die maximalen Berechtigungen festgelegt werden können, die durch gängigere Richtlinientypen gewährt werden:

- Berechtigungsgrenzen – Eine Berechtigungsgrenze legt die maximalen Berechtigungen fest, die eine identitätsbasierte Richtlinie einer IAM-Entität erteilen kann. Weitere Informationen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im -IAM-Benutzerhandbuch.
- Richtlinien zur Dienstkontrolle (SCPs) — Geben Sie die maximalen Berechtigungen für eine Organisation oder Organisationseinheit in an AWS Organizations. Weitere Informationen finden Sie unter Service-Kontrollrichtlinien im AWS Organizations -Benutzerhandbuch.

- Richtlinien zur Ressourcenkontrolle (RCPs) — Legen Sie die maximal verfügbaren Berechtigungen für Ressourcen in Ihren Konten fest. Weitere Informationen finden Sie im AWS Organizations Benutzerhandbuch unter Richtlinien zur Ressourcenkontrolle (RCPs).
- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die als Parameter übergeben werden, wenn Sie eine temporäre Sitzung für eine Rolle oder einen Verbundbenutzer erstellen. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn für eine Anfrage mehrere Arten von Richtlinien gelten, sind die daraus resultierenden Berechtigungen schwieriger zu verstehen. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie unter Bewertungslogik für Richtlinien im IAM-Benutzerhandbuch.

So funktioniert Deadline Cloud mit IAM

Bevor Sie IAM verwenden, um den Zugriff auf Deadline Cloud zu verwalten, sollten Sie sich darüber informieren, welche IAM-Funktionen mit Deadline Cloud verwendet werden können.

IAM-Funktionen, die Sie mit Deadline Cloud verwenden können AWS

IAM-Feature	Deadline Cloud-Unterstützung
Identitätsbasierte Richtlinien	Ja
Ressourcenbasierte Richtlinien	Nein
Richtlinienaktionen	Ja
Richtlinienressourcen	Ja
Richtlinienbedingungsschlüssel (services pezifisch)	Ja
ACLs	Nein
ABAC (Tags in Richtlinien)	Ja
Temporäre Anmeldeinformationen	Ja

IAM-Feature	Deadline Cloud-Unterstützung
Forward Access Sessions (FAS)	Ja
Servicerollen	Ja
Service-verknüpfte Rollen	Nein

Einen allgemeinen Überblick darüber, wie Deadline Cloud und andere mit den meisten IAM-Funktionen AWS-Services funktionieren, finden Sie im IAM-Benutzerhandbuch unter AWS Dienste, die mit IAM funktionieren.

Identitätsbasierte Richtlinien für Deadline Cloud

Unterstützt Richtlinien auf Identitätsbasis: Ja

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien im IAM-Benutzerhandbuch.

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen zugelassen oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen, unter denen Aktionen zugelassen oder abgelehnt werden. Informationen zu sämtlichen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der IAM-Referenz für JSON-Richtlinienelemente im IAM-Benutzerhandbuch.

Beispiele für identitätsbasierte Richtlinien für Deadline Cloud

Beispiele für identitätsbasierte Richtlinien von Deadline Cloud finden Sie unter. Beispiele für identitätsbasierte Richtlinien für Deadline Cloud

Ressourcenbasierte Richtlinien in Deadline Cloud

Unterstützt ressourcenbasierte Richtlinien: Nein

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und

Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Um kontoübergreifenden Zugriff zu ermöglichen, können Sie ein gesamtes Konto oder IAM-Entitäten in einem anderen Konto als Prinzipal in einer ressourcenbasierten Richtlinie angeben. Weitere Informationen finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Richtlinienaktionen für Deadline Cloud

Unterstützt Richtlinienaktionen: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Nehmen Sie Aktionen in eine Richtlinie auf, um Berechtigungen zur Ausführung des zugehörigen Vorgangs zu erteilen.

Eine Liste der Deadline Cloud-Aktionen finden Sie unter Von AWS Deadline Cloud definierte Aktionen in der Serviceautorisierungsreferenz.

Richtlinienaktionen in Deadline Cloud verwenden vor der Aktion das folgende Präfix:

```
deadline
```

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie mit Kommata:

```
"Action": [
      "deadline:action1",
      "deadline:action2"
         ]
```

Beispiele für identitätsbasierte Richtlinien von Deadline Cloud finden Sie unter. Beispiele für identitätsbasierte Richtlinien für Deadline Cloud

Richtlinienressourcen für Deadline Cloud

Unterstützt Richtlinienressourcen: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Als Best Practice geben Sie eine Ressource mit dem zugehörigen Amazon-Ressourcennamen (ARN) an. Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

```
"Resource": "*"
```

Eine Liste der Deadline Cloud-Ressourcentypen und ihrer ARNs Eigenschaften finden Sie unter Von AWS Deadline Cloud definierte Ressourcen in der Service Authorization Reference. Informationen dazu, mit welchen Aktionen Sie den ARN jeder Ressource angeben können, finden Sie unter Von AWS Deadline Cloud definierte Aktionen.

Beispiele für identitätsbasierte Richtlinien von Deadline Cloud finden Sie unter. Beispiele für identitätsbasierte Richtlinien für Deadline Cloud

Bedingungsschlüssel für Richtlinien für Deadline Cloud

Unterstützt servicespezifische Richtlinienbedingungsschlüssel: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das Element `Condition` gibt an, wann Anweisungen auf der Grundlage definierter Kriterien ausgeführt werden. Sie können bedingte Ausdrücke erstellen, die Bedingungsoperatoren verwenden, z. B. ist gleich oder kleiner als, damit die Bedingung in der Richtlinie mit Werten in der Anforderung übereinstimmt. Eine Übersicht aller AWS globalen Bedingungsschlüssel finden Sie unter Kontextschlüssel für AWS globale Bedingungen im IAM-Benutzerhandbuch.

Eine Liste der Deadline Cloud-Bedingungsschlüssel finden Sie unter Bedingungsschlüssel für AWS Deadline Cloud in der Service Authorization Reference. Informationen zu den Aktionen und Ressourcen, mit denen Sie einen Bedingungsschlüssel verwenden können, finden Sie unter Von AWS Deadline Cloud definierte Aktionen.

Beispiele für identitätsbasierte Richtlinien von Deadline Cloud finden Sie unter. Beispiele für identitätsbasierte Richtlinien für Deadline Cloud

ACLs in Deadline Cloud

Unterstützt ACLs: Nein

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLs ähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

ABAC mit Deadline Cloud

Unterstützt ABAC (Tags in Richtlinien): Ja

Die attributbasierte Zugriffskontrolle (ABAC) ist eine Autorisierungsstrategie, bei der Berechtigungen basierend auf Attributen, auch als Tags bezeichnet, definiert werden. Sie können Tags an IAM-Entitäten und AWS -Ressourcen anhängen und dann ABAC-Richtlinien entwerfen, um Operationen zu ermöglichen, wenn das Tag des Prinzipals mit dem Tag auf der Ressource übereinstimmt.

Um den Zugriff auf der Grundlage von Tags zu steuern, geben Sie im Bedingungselement einer Richtlinie Tag-Informationen an, indem Sie die Schlüssel `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*, oder Bedingung `aws:TagKeys` verwenden.

Wenn ein Service alle drei Bedingungsschlüssel für jeden Ressourcentyp unterstützt, lautet der Wert für den Service Ja. Wenn ein Service alle drei Bedingungsschlüssel für nur einige Ressourcentypen unterstützt, lautet der Wert Teilweise.

Weitere Informationen zu ABAC finden Sie unter Definieren von Berechtigungen mit ABAC-Autorisierung im IAM-Benutzerhandbuch. Um ein Tutorial mit Schritten zur Einstellung von ABAC anzuzeigen, siehe Attributbasierte Zugriffskontrolle (ABAC) verwenden im IAM-Benutzerhandbuch.

Temporäre Anmeldeinformationen mit Deadline Cloud verwenden

Unterstützt temporäre Anmeldeinformationen: Ja

Temporäre Anmeldeinformationen ermöglichen kurzfristigen Zugriff auf AWS Ressourcen und werden automatisch erstellt, wenn Sie den Verbund verwenden oder die Rollen wechseln. AWS empfiehlt, temporäre Anmeldeinformationen dynamisch zu generieren, anstatt langfristige Zugriffsschlüssel zu verwenden. Weitere Informationen finden Sie unter Temporäre Anmeldeinformationen in IAM und AWS-Services , die mit IAM funktionieren im IAM-Benutzerhandbuch.

Zugriffssitzungen für Deadline Cloud weiterleiten

Unterstützt Forward Access Sessions (FAS): Ja

Forward-Access-Sitzungen (FAS) verwenden die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, kombiniert mit der Anforderung, Anfragen AWS-Service an nachgelagerte Dienste zu stellen. Einzelheiten zu den Richtlinien für FAS-Anforderungen finden Sie unter Zugriffssitzungen weiterleiten.

Servicerollen für Deadline Cloud

Unterstützt Servicerollen: Ja

Eine Servicerolle ist eine IAM-Rolle, die ein Service annimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.

> **Warning**
>
> Das Ändern der Berechtigungen für eine Servicerolle kann die Funktionalität von Deadline Cloud beeinträchtigen. Bearbeiten Sie Servicerollen nur, wenn Deadline Cloud Sie dazu anleitet.

Servicebezogene Rollen für Deadline Cloud

Unterstützt serviceverknüpfte Rollen: Ja

Eine serviceverknüpfte Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Dienstbezogene Rollen werden in Ihrem Dienst angezeigt AWS-Konto und gehören dem Dienst. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.

Details zum Erstellen oder Verwalten von serviceverknüpften Rollen finden Sie unter AWS -Services, die mit IAM funktionieren. Suchen Sie in der Tabelle nach einem Service mit einem `Yes` in der Spalte Service-linked role (Serviceverknüpfte Rolle). Wählen Sie den Link Yes (Ja) aus, um die Dokumentation für die serviceverknüpfte Rolle für diesen Service anzuzeigen.

Beispiele für identitätsbasierte Richtlinien für Deadline Cloud

Standardmäßig sind Benutzer und Rollen nicht berechtigt, Deadline Cloud-Ressourcen zu erstellen oder zu ändern. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen.

Informationen dazu, wie Sie unter Verwendung dieser beispielhaften JSON-Richtliniendokumente eine identitätsbasierte IAM-Richtlinie erstellen, finden Sie unter Erstellen von IAM-Richtlinien (Konsole) im IAM-Benutzerhandbuch.

Einzelheiten zu den von Deadline Cloud definierten Aktionen und Ressourcentypen, einschließlich des Formats ARNs für die einzelnen Ressourcentypen, finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für AWS Deadline Cloud in der Service Authorization Reference.

Themen

Best Practices für Richtlinien

Identitätsbasierte Richtlinien legen fest, ob jemand Deadline Cloud-Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder sie löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Wenn Sie identitätsbasierte Richtlinien erstellen oder bearbeiten, befolgen Sie diese Richtlinien und Empfehlungen:

- Erste Schritte mit AWS verwalteten Richtlinien und Umstellung auf Berechtigungen mit den geringsten Rechten — Verwenden Sie die AWS verwalteten Richtlinien, die Berechtigungen für viele gängige Anwendungsfälle gewähren, um damit zu beginnen, Ihren Benutzern und Workloads

Berechtigungen zu gewähren. Sie sind in Ihrem verfügbar. AWS-Konto Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom AWS Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden Sie unter Von AWS verwaltete Richtlinien oder Von AWS verwaltete Richtlinien für Auftragsfunktionen im IAM-Benutzerhandbuch.

- Anwendung von Berechtigungen mit den geringsten Rechten – Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum Anwenden von Berechtigungen finden Sie unter Richtlinien und Berechtigungen in IAM im IAM-Benutzerhandbuch.
- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs – Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Serviceaktionen zu gewähren, wenn diese für einen bestimmten Zweck verwendet werden AWS-Service, z. CloudFormation B. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.
- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und funktionale Berechtigungen zu gewährleisten – IAM Access Analyzer validiert neue und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter Richtlinienvalidierung mit IAM Access Analyzer im IAM-Benutzerhandbuch.
- Multi-Faktor-Authentifizierung (MFA) erforderlich — Wenn Sie ein Szenario haben, das IAM-Benutzer oder einen Root-Benutzer in Ihrem System erfordert AWS-Konto, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter Sicherer API-Zugriff mit MFA im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter Best Practices für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Verwenden Sie die Deadline Cloud-Konsole

Um auf die AWS Deadline Cloud-Konsole zugreifen zu können, benötigen Sie ein Mindestmaß an Berechtigungen. Diese Berechtigungen müssen es Ihnen ermöglichen, Details zu den Deadline Cloud-Ressourcen in Ihrem aufzulisten und einzusehen AWS-Konto. Wenn Sie eine identitätsbasierte Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole nicht wie vorgesehen für Entitäten (Benutzer oder Rollen) mit dieser Richtlinie.

Sie müssen Benutzern, die nur die API AWS CLI oder die AWS API aufrufen, keine Mindestberechtigungen für die Konsole gewähren. Stattdessen sollten Sie nur Zugriff auf die Aktionen zulassen, die der API-Operation entsprechen, die die Benutzer ausführen möchten.

Um sicherzustellen, dass Benutzer und Rollen die Deadline Cloud-Konsole weiterhin verwenden können, fügen Sie den Entitäten auch die Deadline Cloud *ConsoleAccess* oder die *ReadOnly* AWS verwaltete Richtlinie hinzu. Weitere Informationen finden Sie unter Hinzufügen von Berechtigungen zu einem Benutzer im IAM-Benutzerhandbuch.

Richtlinie für den Zugriff auf die Konsole

Um Zugriff auf alle Funktionen der Deadline Cloud-Konsole zu gewähren, fügen Sie diese Identitätsrichtlinie einem Benutzer oder einer Rolle hinzu, auf die Sie vollen Zugriff haben möchten.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "EC2InstanceTypeSelection",
        "Effect": "Allow",
        "Action": [
            "ec2:DescribeInstanceTypeOfferings",
            "ec2:DescribeInstanceTypes",
            "ec2:GetInstanceTypesFromInstanceRequirements",
            "pricing:GetProducts"
        ],
        "Resource": ["*"]
    },
    {
        "Sid": "VPCResourceSelection",
        "Effect": "Allow",
        "Action": [
```

```
            "ec2:DescribeVpcs",
            "ec2:DescribeSubnets",
            "ec2:DescribeSecurityGroups"
        ],
        "Resource": ["*"]
    },
    {
        "Sid": "ViewVpcLatticeResources",
        "Effect": "Allow",
        "Action": [
            "vpc-lattice:ListResourceConfigurations",
            "vpc-lattice:GetResourceConfiguration",
            "vpc-lattice:GetResourceGateway"
        ],
        "Resource": ["*"]
    },
    {
        "Sid": "ManageVpcEndpointsViaDeadline",
        "Effect": "Allow",
        "Action": [
            "ec2:CreateVpcEndpoint",
            "ec2:DescribeVpcEndpoints",
            "ec2:DeleteVpcEndpoints",
        "ec2:CreateTags"
        ],
        "Resource": ["*"],
        "Condition": {
        "StringEquals": { "aws:CalledViaFirst": "deadline.amazonaws.com" }
        }
    },
    {
        "Sid": "ChooseJobAttachmentsBucket",
        "Effect": "Allow",
        "Action": ["s3:GetBucketLocation", "s3:ListAllMyBuckets"],
        "Resource": "*"
    },
    {
        "Sid": "CreateDeadlineCloudLogGroups",
        "Effect": "Allow",
        "Action": ["logs:CreateLogGroup"],
        "Resource": "arn:aws:logs:*:*:log-group:/aws/deadline/*",
        "Condition": {
        "StringLike": { "aws:CalledViaFirst": "deadline.amazonaws.com" }
        }
```

```
    },
    {
        "Sid": "ValidateDependencies",
        "Effect": "Allow",
        "Action": ["s3:ListBucket"],
        "Resource": "*",
        "Condition": {
        "StringLike": { "aws:CalledViaFirst": "deadline.amazonaws.com" }
        }
    },
    {
        "Sid": "RoleSelection",
        "Effect": "Allow",
        "Action": ["iam:GetRole", "iam:ListRoles"],
        "Resource": "*"
    },
    {
        "Sid": "PassRoleToDeadlineCloud",
        "Effect": "Allow",
        "Action": ["iam:PassRole"],
        "Condition": {
        "StringLike": { "iam:PassedToService": "deadline.amazonaws.com" }
    },
        "Resource": "*"
    },
    {
        "Sid": "KMSKeySelection",
        "Effect": "Allow",
        "Action": ["kms:ListKeys", "kms:ListAliases"],
        "Resource": "*"
    },
    {
        "Sid": "IdentityStoreReadOnly",
        "Effect": "Allow",
        "Action": [
            "identitystore:DescribeUser",
            "identitystore:DescribeGroup",
            "identitystore:ListGroups",
            "identitystore:ListUsers",
            "identitystore:IsMemberInGroups",
            "identitystore:ListGroupMemberships",
            "identitystore:ListGroupMembershipsForMember",
            "identitystore:GetGroupMembershipId"
    ],
```

```
            "Resource": "*"
        },
        {
            "Sid": "OrganizationAndIdentityCenterIdentification",
            "Effect": "Allow",
            "Action": [
                "sso:ListDirectoryAssociations",
                "organizations:DescribeAccount",
                "organizations:DescribeOrganization",
                "sso:DescribeRegisteredRegions",
                "sso:GetManagedApplicationInstance",
                "sso:GetSharedSsoConfiguration",
                "sso:ListInstances",
                "sso:GetApplicationAssignmentConfiguration",
                "sso:GetSSOStatus",
                "sso:ListRegions",
                "sso:DescribeRegion"
        ],
            "Resource": "*"
        },
        {
            "Sid": "ManagedDeadlineCloudIDCApplication",
            "Effect": "Allow",
            "Action": [
                "sso:CreateApplication",
                "sso:PutApplicationAssignmentConfiguration",
                "sso:PutApplicationAuthenticationMethod",
                "sso:PutApplicationGrant",
                "sso:DeleteApplication",
                "sso:UpdateApplication"
        ],
            "Resource": "*",
            "Condition": {
            "StringLike": { "aws:CalledViaFirst": "deadline.amazonaws.com" }
            }
        },
        {
            "Sid": "ChooseSecret",
            "Effect": "Allow",
            "Action": ["secretsmanager:ListSecrets"],
            "Resource": "*"
        },
        {
            "Sid": "DeadlineMembershipActions",
```

```
        "Effect": "Allow",
        "Action": [
            "deadline:AssociateMemberToFarm",
            "deadline:AssociateMemberToFleet",
            "deadline:AssociateMemberToQueue",
            "deadline:AssociateMemberToJob",
            "deadline:DisassociateMemberFromFarm",
            "deadline:DisassociateMemberFromFleet",
            "deadline:DisassociateMemberFromQueue",
            "deadline:DisassociateMemberFromJob",
            "deadline:ListFarmMembers",
            "deadline:ListFleetMembers",
            "deadline:ListQueueMembers",
            "deadline:ListJobMembers"
    ],
        "Resource": ["*"]
    },
    {
        "Sid": "DeadlineControlPlaneActions",
        "Effect": "Allow",
        "Action": [
            "deadline:CreateMonitor",
            "deadline:GetMonitor",
            "deadline:UpdateMonitor",
            "deadline:DeleteMonitor",
            "deadline:ListMonitors",
            "deadline:CreateFarm",
            "deadline:GetFarm",
            "deadline:UpdateFarm",
            "deadline:DeleteFarm",
            "deadline:ListFarms",
            "deadline:CreateQueue",
            "deadline:GetQueue",
            "deadline:UpdateQueue",
            "deadline:DeleteQueue",
            "deadline:ListQueues",
            "deadline:CreateFleet",
            "deadline:GetFleet",
            "deadline:UpdateFleet",
            "deadline:DeleteFleet",
            "deadline:ListFleets",
            "deadline:ListWorkers",
            "deadline:CreateQueueFleetAssociation",
            "deadline:GetQueueFleetAssociation",
```

```
            "deadline:UpdateQueueFleetAssociation",
            "deadline:DeleteQueueFleetAssociation",
            "deadline:ListQueueFleetAssociations",
            "deadline:CreateQueueEnvironment",
            "deadline:GetQueueEnvironment",
            "deadline:UpdateQueueEnvironment",
            "deadline:DeleteQueueEnvironment",
            "deadline:ListQueueEnvironments",
            "deadline:CreateLimit",
            "deadline:GetLimit",
            "deadline:UpdateLimit",
            "deadline:DeleteLimit",
            "deadline:ListLimits",
            "deadline:CreateQueueLimitAssociation",
            "deadline:GetQueueLimitAssociation",
            "deadline:DeleteQueueLimitAssociation",
            "deadline:UpdateQueueLimitAssociation",
            "deadline:ListQueueLimitAssociations",
            "deadline:CreateStorageProfile",
            "deadline:GetStorageProfile",
            "deadline:UpdateStorageProfile",
            "deadline:DeleteStorageProfile",
            "deadline:ListStorageProfiles",
            "deadline:ListStorageProfilesForQueue",
            "deadline:ListBudgets",
            "deadline:TagResource",
            "deadline:UntagResource",
            "deadline:ListTagsForResource",
            "deadline:CreateLicenseEndpoint",
            "deadline:GetLicenseEndpoint",
            "deadline:DeleteLicenseEndpoint",
            "deadline:ListLicenseEndpoints",
            "deadline:ListAvailableMeteredProducts",
            "deadline:ListMeteredProducts",
            "deadline:PutMeteredProduct",
            "deadline:DeleteMeteredProduct"
        ],
        "Resource": ["*"]
      }]
}
```

Richtlinie zum Einreichen von Jobs an eine Warteschlange

In diesem Beispiel erstellen Sie eine Richtlinie mit eingeschränktem Geltungsbereich, die die Berechtigung zum Senden von Aufträgen an eine bestimmte Warteschlange in einer bestimmten Farm erteilt.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "SubmitJobsFarmAndQueue",
            "Effect": "Allow",
            "Action": "deadline:CreateJob",
            "Resource": "arn:aws:deadline:us-east-1:111122223333:farm/FARM_A/
queue/QUEUE_B/job/*"
        }
    ]
}
```

Richtlinie, die die Erstellung eines Lizenzendpunkts ermöglicht

In diesem Beispiel erstellen Sie eine nach unten abgegrenzte Richtlinie, die die erforderlichen Berechtigungen zum Erstellen und Verwalten von Lizenzendpunkten gewährt. Verwenden Sie diese Richtlinie, um den Lizenzendpunkt für die VPC zu erstellen, die Ihrer Farm zugeordnet ist.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "CreateLicenseEndpoint",
        "Effect": "Allow",
        "Action": [
            "deadline:CreateLicenseEndpoint",
            "deadline:DeleteLicenseEndpoint",
            "deadline:GetLicenseEndpoint",
            "deadline:ListLicenseEndpoints",
            "deadline:PutMeteredProduct",
```

```
            "deadline:DeleteMeteredProduct",
            "deadline:ListMeteredProducts",
            "deadline:ListAvailableMeteredProducts",
            "ec2:CreateVpcEndpoint",
            "ec2:DescribeVpcEndpoints",
            "ec2:DeleteVpcEndpoints"
        ],
        "Resource": [
            "arn:aws:deadline:*:111122223333:*",
            "arn:aws:ec2:*:111122223333:vpc-endpoint/*"
        ]
    }]
}
```

Richtlinie, die die Überwachung einer bestimmten Farmwarteschlange ermöglicht

In diesem Beispiel erstellen Sie eine Richtlinie mit eingeschränktem Geltungsbereich, die die Erlaubnis erteilt, Jobs in einer bestimmten Warteschlange für eine bestimmte Farm zu überwachen.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "MonitorJobsFarmAndQueue",
        "Effect": "Allow",
        "Action": [
            "deadline:SearchJobs",
            "deadline:ListJobs",
            "deadline:GetJob",
            "deadline:SearchSteps",
            "deadline:ListSteps",
            "deadline:ListStepConsumers",
            "deadline:ListStepDependencies",
            "deadline:GetStep",
            "deadline:SearchTasks",
            "deadline:ListTasks",
            "deadline:GetTask",
            "deadline:ListSessions",
            "deadline:GetSession",
            "deadline:ListSessionActions",
```

```
            "deadline:GetSessionAction"
        ],
        "Resource": [
            "arn:aws:deadline:us-east-1:123456789012:farm/FARM_A/queue/QUEUE_B",
            "arn:aws:deadline:us-east-1:123456789012:farm/FARM_A/queue/QUEUE_B/*"
        ]
    }]
}
```

AWS verwaltete Richtlinien für Deadline Cloud

Eine AWS verwaltete Richtlinie ist eine eigenständige Richtlinie, die von erstellt und verwaltet wird AWS. AWS Verwaltete Richtlinien dienen dazu, Berechtigungen für viele gängige Anwendungsfälle bereitzustellen, sodass Sie damit beginnen können, Benutzern, Gruppen und Rollen Berechtigungen zuzuweisen.

Beachten Sie, dass AWS verwaltete Richtlinien für Ihre speziellen Anwendungsfälle möglicherweise keine Berechtigungen mit den geringsten Rechten gewähren, da sie allen AWS Kunden zur Verfügung stehen. Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind.

Sie können die in AWS verwalteten Richtlinien definierten Berechtigungen nicht ändern. Wenn die in einer AWS verwalteten Richtlinie definierten Berechtigungen AWS aktualisiert werden, wirkt sich das Update auf alle Prinzidentitäten (Benutzer, Gruppen und Rollen) aus, denen die Richtlinie zugeordnet ist. AWS aktualisiert eine AWS verwaltete Richtlinie höchstwahrscheinlich, wenn eine neue Richtlinie eingeführt AWS-Service wird oder neue API-Operationen für bestehende Dienste verfügbar werden.

Weitere Informationen finden Sie unter Von AWS verwaltete Richtlinien im IAM-Benutzerhandbuch.

AWS verwaltete Richtlinie: AWSDeadlineCloud-FleetWorker

Sie können die `AWSDeadlineCloud-FleetWorker` Richtlinie an Ihre AWS Identity and Access Management (IAM-) Identitäten anhängen.

Diese Richtlinie gewährt den Mitarbeitern dieser Flotte die Berechtigungen, die sie benötigen, um eine Verbindung mit dem Service herzustellen und Aufgaben vom Service zu empfangen.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen:

- `deadline`— Ermöglicht es Prinzipalen, Mitarbeiter in einer Flotte zu verwalten.

Eine JSON-Liste der Richtliniendetails finden Sie AWSDeadlineCloud-FleetWorkerim Referenzhandbuch zu AWS Managed Policy.

AWS verwaltete Richtlinie: AWSDeadlineCloud-WorkerHost

Sie können die `AWSDeadlineCloud-WorkerHost`-Richtlinie an Ihre IAM-Identitäten anfügen.

Diese Richtlinie gewährt die Berechtigungen, die für die anfängliche Verbindung mit dem Dienst erforderlich sind. Es kann als Amazon Elastic Compute Cloud (Amazon EC2) Instance-Profil verwendet werden.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen:

- `deadline`— Ermöglicht dem Benutzer, Worker zu erstellen, die Flottenrolle für Arbeiter zu übernehmen und Tags auf Arbeiter anzuwenden

Eine JSON-Liste der Richtliniendetails finden Sie AWSDeadlineCloud-WorkerHostim Referenzhandbuch zu AWS Managed Policy.

AWS verwaltete Richtlinie: AWSDeadlineCloud-UserAccessFarms

Sie können die `AWSDeadlineCloud-UserAccessFarms`-Richtlinie an Ihre IAM-Identitäten anfügen.

Diese Richtlinie ermöglicht Benutzern den Zugriff auf Farmdaten auf der Grundlage der Farmen, in denen sie Mitglied sind, und ihrer Mitgliedschaftsstufe.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen:

- `deadline`— Ermöglicht dem Benutzer den Zugriff auf Farmdaten.
- `ec2`— Ermöglicht Benutzern, Details zu Amazon EC2 EC2-Instance-Typen zu sehen.

- `identitystore`— Ermöglicht Benutzern, Benutzer- und Gruppennamen zu sehen.
- `kms`— Ermöglicht Benutzern, vom Kunden verwaltete Schlüssel für ihre AWS Key Management Service (AWS KMS IAM Identity Center) -Instanz zu konfigurieren AWS IAM Identity Center ().

Eine JSON-Liste der Richtliniendetails finden Sie AWSDeadlineCloud-UserAccessFarmsim Referenzhandbuch zu AWS Managed Policy.

AWS verwaltete Richtlinie: AWSDeadlineCloud-UserAccessFleets

Sie können die `AWSDeadlineCloud-UserAccessFleets`-Richtlinie an Ihre IAM-Identitäten anfügen.

Diese Richtlinie ermöglicht Benutzern den Zugriff auf Flottendaten auf der Grundlage der Farmen, in denen sie Mitglied sind, und ihrer Mitgliedschaftsstufe.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen:

- `deadline`— Ermöglicht dem Benutzer den Zugriff auf Farmdaten.
- `ec2`— Ermöglicht Benutzern, Details zu Amazon EC2 EC2-Instance-Typen zu sehen.
- `identitystore`— Ermöglicht Benutzern, Benutzer- und Gruppennamen zu sehen.

Eine JSON-Liste der Richtliniendetails finden Sie AWSDeadlineCloud-UserAccessFleetsim Referenzhandbuch zu AWS Managed Policy.

AWS verwaltete Richtlinie: AWSDeadlineCloud-UserAccessJobs

Sie können die `AWSDeadlineCloud-UserAccessJobs`-Richtlinie an Ihre IAM-Identitäten anfügen.

Diese Richtlinie ermöglicht Benutzern den Zugriff auf Auftragsdaten auf der Grundlage der Farmen, in denen sie Mitglied sind, und ihrer Mitgliedschaftsstufe.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen:

- `deadline`— Ermöglicht dem Benutzer den Zugriff auf Farmdaten.
- `ec2`— Ermöglicht Benutzern, Details zu Amazon EC2 EC2-Instance-Typen zu sehen.
- `identitystore`— Ermöglicht Benutzern, Benutzer- und Gruppennamen zu sehen.

Eine JSON-Liste der Richtliniendetails finden Sie AWSDeadlineCloud-UserAccessJobsim Referenzhandbuch zu AWS Managed Policy.

AWS verwaltete Richtlinie: AWSDeadlineCloud-UserAccessQueues

Sie können die `AWSDeadlineCloud-UserAccessQueues`-Richtlinie an Ihre IAM-Identitäten anfügen.

Diese Richtlinie ermöglicht Benutzern den Zugriff auf Warteschlangendaten auf der Grundlage der Farmen, in denen sie Mitglied sind, und ihrer Mitgliedschaftsstufe.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen:

- `deadline`— Ermöglicht dem Benutzer den Zugriff auf Farmdaten.
- `ec2`— Ermöglicht Benutzern, Details zu Amazon EC2 EC2-Instance-Typen zu sehen.
- `identitystore`— Ermöglicht Benutzern, Benutzer- und Gruppennamen zu sehen.

Eine JSON-Liste der Richtliniendetails finden Sie AWSDeadlineCloud-UserAccessQueuesim Referenzhandbuch zu AWS Managed Policy.

Deadline Cloud-Updates für AWS verwaltete Richtlinien

Sehen Sie sich Details zu Aktualisierungen der AWS verwalteten Richtlinien für Deadline Cloud an, seit dieser Dienst begonnen hat, diese Änderungen zu verfolgen. Um automatische Benachrichtigungen über Änderungen an dieser Seite zu erhalten, abonnieren Sie den RSS-Feed auf der Deadline Cloud-Dokumentverlaufsseite.

Änderungen	Beschreibung	Date
AWSDeadlineCloud-UserAccessFarms— Änderung	Deadline Cloud hat eine neue Aktion hinzugefügt, `kms:Decrypt` sodass Sie einen AWS KMS vom Kunden verwalteten Schlüssel mit Ihrer	22. Dezember 2025

Änderungen	Beschreibung	Date
	IAM Identity Center-Instanz verwenden können.	
AWSDeadlineCloud-WorkerHost— Veränderung	Deadline Cloud hat neue Aktionen `deadline:TagResource` hinzugefügt`deadline:ListTagsForResource`, sodass Sie Tags hinzufügen und anzeigen können, die mit Mitarbeitern in Ihrer Flotte verknüpft sind.	30. Mai 2025
AWSDeadlineCloud-UserAccessFarms— Ändern AWSDeadlineCloud-UserAccessJobs— Veränderung AWSDeadlineCloud-UserAccessQueues— Veränderung	Deadline Cloud hat neue Aktionen `deadline:GetJobTemplate` hinzugefügt`deadline:ListJobParameterDefinitions`, sodass Sie Jobs erneut einreichen können.	7. Oktober 2024
Deadline Cloud hat begonnen, Änderungen zu verfolgen	Deadline Cloud begann, Änderungen an seinen AWS verwalteten Richtlinien nachzuverfolgen.	2. April 2024

Servicerollen

Wie Deadline Cloud IAM-Servicerollen verwendet

Deadline Cloud übernimmt automatisch IAM-Rollen und stellt temporäre Anmeldeinformationen für Mitarbeiter, Jobs und den Deadline Cloud-Monitor bereit. Dieser Ansatz macht die manuelle Verwaltung von Anmeldeinformationen überflüssig und gewährleistet gleichzeitig die Sicherheit durch eine rollenbasierte Zugriffskontrolle.

Wenn Sie Monitore, Flotten und Warteschlangen erstellen, geben Sie IAM-Rollen an, die Deadline Cloud in Ihrem Namen übernimmt. Mitarbeiter und der Deadline Cloud-Monitor erhalten dann temporäre Zugangsdaten von diesen Rollen. AWS-Services

Rolle „Flotte“

Konfigurieren Sie eine Flottenrolle, um Mitarbeitern von Deadline Cloud die Berechtigungen zu geben, die sie benötigen, um Arbeit entgegenzunehmen und den Fortschritt dieser Arbeit zu melden.

In der Regel müssen Sie diese Rolle nicht selbst konfigurieren. Diese Rolle kann in der Deadline Cloud-Konsole für Sie erstellt werden, um die erforderlichen Berechtigungen zu enthalten. Verwenden Sie die folgende Anleitung, um die Besonderheiten dieser Rolle zur Fehlerbehebung zu verstehen.

Wenn Sie Flotten programmgesteuert erstellen oder aktualisieren, geben Sie den ARN für die Flottenrolle mithilfe der API-Operationen `CreateFleet` oder `UpdateFleet` an.

Was macht die Flottenrolle

Die Flottenrolle bietet Mitarbeitern folgende Berechtigungen:

- Empfangen Sie neue Arbeiten und melden Sie den Fortschritt laufender Arbeiten an den Deadline Cloud-Dienst
- Verwalten Sie den Lebenszyklus und den Status Ihrer Mitarbeiter
- Protokollereignisse in Amazon CloudWatch Logs für die Worker-Logs aufzeichnen

Richten Sie die Vertrauensrichtlinie für Flottenrollen ein

Ihre Flottenrolle muss dem Deadline Cloud-Dienst vertrauen und auf Ihre spezifische Farm zugeschnitten sein.

Als bewährte Methode sollte die Vertrauensrichtlinie Sicherheitsbedingungen für den Schutz von Confused Deputy beinhalten. Weitere Informationen zum Schutz vor Confused Deputy finden Sie unter Confused Deputy im Deadline Cloud-Benutzerhandbuch.

- `aws:SourceAccount`stellt sicher, dass nur Ressourcen desselben AWS-Konto Unternehmens diese Rolle übernehmen können.
- `aws:SourceArn`schränkt die Rollenübernahme auf eine bestimmte Deadline Cloud-Farm ein.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowDeadlineCredentialsService",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Principal": {
        "Service": "credentials.deadline.amazonaws.com"
      },
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "YOUR_ACCOUNT_ID"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:deadline:REGION:YOUR_ACCOUNT_ID:farm/YOUR_FARM_ID"
        }
      }
    }
  ]
}
```

Hängen Sie die Berechtigungen für die Flottenrolle an

Fügen Sie Ihrer Flottenrolle die folgende AWS verwaltete Richtlinie hinzu:

AWSDeadlineCloud-FleetWorker

Diese verwaltete Richtlinie bietet Berechtigungen für:

- `deadline:AssumeFleetRoleForWorker`- Ermöglicht Mitarbeitern, ihre Anmeldeinformationen zu aktualisieren.
- `deadline:UpdateWorker`- Ermöglicht Mitarbeitern, ihren Status zu aktualisieren (z. B. beim Beenden auf STOPPED).
- `deadline:UpdateWorkerSchedule`- Zur Erfassung von Arbeitsaufträgen und zur Berichterstattung über den Fortschritt.
- `deadline:BatchGetJobEntity`- Zum Abrufen von Jobinformationen.
- `deadline:AssumeQueueRoleForWorker`- Für den Zugriff auf Anmeldeinformationen für Warteschlangenrollen während der Jobausführung.

Fügen Sie KMS-Berechtigungen für verschlüsselte Farmen hinzu

Wenn Ihre Farm mit einem KMS-Schlüssel erstellt wurde, fügen Sie diese Berechtigungen Ihrer Flottenrolle hinzu, um sicherzustellen, dass der Worker auf verschlüsselte Daten in der Farm zugreifen kann.

Die KMS-Berechtigungen sind nur erforderlich, wenn Ihrer Farm ein KMS-Schlüssel zugeordnet ist. Die `kms:ViaService` Bedingung muss das Format verwenden`deadline.{region}.amazonaws.com`.

Beim Erstellen einer Flotte wird eine CloudWatch Logs-Protokollgruppe für diese Flotte erstellt. Die Berechtigungen des Mitarbeiters werden vom Deadline Cloud-Dienst verwendet, um einen Log-Stream speziell für diesen bestimmten Mitarbeiter zu erstellen. Nachdem der Worker eingerichtet und ausgeführt wurde, verwendet der Worker diese Berechtigungen, um Protokollereignisse direkt an CloudWatch Logs zu senden.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CreateLogStream",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream"
      ],
      "Resource": "arn:aws:logs:REGION:YOUR_ACCOUNT_ID:log-group:/aws/
deadline/YOUR_FARM_ID/*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "deadline.REGION.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "ManageLogEvents",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents",
        "logs:GetLogEvents"
      ],
```

```
      "Resource": "arn:aws:logs:REGION:YOUR_ACCOUNT_ID:log-group:/aws/
deadline/YOUR_FARM_ID/*"
    },
    {
      "Sid": "ManageKmsKey",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:DescribeKey",
        "kms:GenerateDataKey"
      ],
      "Resource": "YOUR_FARM_KMS_KEY_ARN",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "deadline.REGION.amazonaws.com"
        }
      }
    }
  ]
}
```

Änderung der Flottenrolle

Die Berechtigungen für die Flottenrolle sind nicht anpassbar. Die beschriebenen Berechtigungen sind immer erforderlich und das Hinzufügen zusätzlicher Berechtigungen hat keine Auswirkung.

Vom Kunden verwaltete Flottenhost-Rolle

Richten Sie eine WorkerHost Rolle ein, wenn Sie kundenverwaltete Flotten auf Amazon EC2 EC2-Instances oder lokalen Hosts verwenden.

Was macht die Rolle WorkerHost

Durch diese WorkerHost Rolle werden Mitarbeiter auf vom Kunden verwaltete Flottenhosts angewiesen. Sie bietet die Mindestberechtigungen, die ein Host benötigt, um:

- Einen Worker in Deadline Cloud erstellen
- Nehmen Sie die Flottenrolle an, um Betriebsdaten abzurufen
- Kennzeichnen Sie Mitarbeiter mit Flottenkennzeichnungen (sofern die Tag-Weitergabe aktiviert ist)

Richten Sie WorkerHost Rollenberechtigungen ein

Fügen Sie Ihrer WorkerHost Rolle die folgende AWS verwaltete Richtlinie hinzu:

AWSDeadlineCloud-WorkerHost

Diese verwaltete Richtlinie bietet Berechtigungen für:

- `deadline:CreateWorker`- Ermöglicht dem Host, einen neuen Worker zu registrieren.
- `deadline:AssumeFleetRoleForWorker`- Ermöglicht dem Host, die Flottenrolle zu übernehmen.
- `deadline:TagResource`— Ermöglicht das Markieren von Arbeitern während der Erstellung (sofern aktiviert).
- `deadline:ListTagsForResource`- Ermöglicht das Lesen von Flotten-Tags zur Weitergabe.

Verstehen Sie den Bootstrap-Prozess

Die WorkerHost Rolle wird nur beim ersten Start des Workers verwendet:

1. Der Worker-Agent startet auf dem Host mithilfe von WorkerHost Anmeldeinformationen.
2. Er ruft `deadline:CreateWorker` zur Registrierung bei Deadline Cloud auf.
3. Es wird dann aufgerufen, `deadline:AssumeFleetRoleForWorker` um die Anmeldeinformationen für die Flottenrolle abzurufen.
4. Ab diesem Zeitpunkt verwendet der Worker für alle Operationen nur noch Anmeldeinformationen für Flottenrollen.

Die WorkerHost Rolle wird nicht mehr verwendet, nachdem der Worker gestartet wurde. Diese Richtlinie ist für vom Service verwaltete Flotten nicht erforderlich. In vom Service verwalteten Flotten wird das Bootstrapping automatisch durchgeführt.

Rolle in der Warteschlange

Die Warteschlangenrolle wird vom Worker bei der Bearbeitung einer Aufgabe übernommen. Diese Rolle stellt die Berechtigungen bereit, die für die Ausführung der Aufgabe erforderlich sind.

Wenn Sie Warteschlangen programmgesteuert erstellen oder aktualisieren, geben Sie den ARN der Warteschlangenrolle mithilfe der API-Operationen `CreateQueue` oder `UpdateQueue` an.

Richten Sie die Vertrauensrichtlinie für die Warteschlangenrolle ein

Ihre Warteschlangenrolle muss dem Deadline Cloud-Dienst vertrauen.

Als bewährte Methode sollte die Vertrauensrichtlinie Sicherheitsbedingungen für den Schutz von Confused Deputy beinhalten. Weitere Informationen zum Schutz vor Confused Deputy finden Sie unter Confused Deputy im Deadline Cloud-Benutzerhandbuch.

- `aws:SourceAccount`stellt sicher, dass nur Ressourcen desselben AWS-Konto Unternehmens diese Rolle übernehmen können.
- `aws:SourceArn`schränkt die Rollenübernahme auf eine bestimmte Deadline Cloud-Farm ein.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "credentials.deadline.amazonaws.com",
          "deadline.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "YOUR_ACCOUNT_ID"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:deadline:us-west-2:123456789012:farm/{farm-id}"
        }
      }
    }
  ]
}
```

Verstehen Sie die Berechtigungen für Warteschlangenrollen

Die Warteschlangenrolle verwendet keine einzige verwaltete Richtlinie. Wenn Sie Ihre Warteschlange in der Konsole konfigurieren, erstellt Deadline Cloud stattdessen eine benutzerdefinierte Richtlinie für Ihre Warteschlange, die auf Ihrer Konfiguration basiert.

Diese automatisch erstellte Richtlinie bietet Zugriff auf:

Arbeitsanhänge

Lese- und Schreibzugriff auf Ihren angegebenen Amazon S3 S3-Bucket für Jobeingabe- und -ausgabedateien:

```
{
  "Effect": "Allow",
  "Action": [
    "s3:GetObject",
    "s3:PutObject",
    "s3:ListBucket",
    "s3:GetBucketLocation"
  ],
  "Resource": [
    "arn:aws:s3:::YOUR_JOB_ATTACHMENTS_BUCKET",
    "arn:aws:s3:::YOUR_JOB_ATTACHMENTS_BUCKET/YOUR_PREFIX/*"
  ],
  "Condition": {
    "StringEquals": {
      "aws:ResourceAccount": "YOUR_ACCOUNT_ID"
    }
  }
}
```

Job-Logs

Lesezugriff auf CloudWatch Protokolle für Jobs in dieser Warteschlange. Jede Warteschlange hat ihre eigene Protokollgruppe und jede Sitzung hat ihren eigenen Protokollstream:

```
{
  "Effect": "Allow",
  "Action": [
    "logs:GetLogEvents"
  ],
  "Resource": "arn:aws:logs:REGION:YOUR_ACCOUNT_ID:log-group:/aws/
deadline/YOUR_FARM_ID/*"
}
```

Software von Drittanbietern

Zugriff zum Herunterladen von Software von Drittanbietern, die von Deadline Cloud unterstützt wird (wie Maya, Blender und andere):

```
{
  "Effect": "Allow",
  "Action": [
    "s3:ListBucket",
    "s3:GetObject"
  ],
  "Resource": "*",
  "Condition": {
    "ArnLike": {
      "s3:DataAccessPointArn": "arn:aws:s3:*:*:accesspoint/deadline-software-*"
    },
    "StringEquals": {
      "s3:AccessPointNetworkOrigin": "VPC"
    }
  }
}
```

Fügen Sie Berechtigungen für Ihre Jobs hinzu

Fügen Sie Ihrer Warteschlangenrolle Berechtigungen hinzu AWS-Services , auf die Ihre Jobs zugreifen müssen. Beim Schreiben von OpenJobDescription Schrittskripten verwendet das SDK AWS CLI und automatisch die Anmeldeinformationen aus Ihrer Warteschlangenrolle. Verwenden Sie diese Option, um auf zusätzliche Dienste zuzugreifen, die Sie zur Erledigung Ihres Jobs benötigen.

Zu den beispielhaften Anwendungsfällen gehören:

- zum Abrufen von benutzerdefinierten Daten
- SSM-Berechtigungen für den Tunnel zu einem benutzerdefinierten Lizenzserver
- CloudWatch für die Ausgabe benutzerdefinierter Messwerte
- Deadline Cloud-Genehmigung zum Erstellen neuer Jobs für dynamische Workflows

Wie werden Anmeldeinformationen für Warteschlangenrollen verwendet

Deadline Cloud stellt Anmeldeinformationen für Warteschlangenrollen bereit für:

- Arbeiter bei der Auftragsausführung

- Benutzer über die Deadline Cloud-CLI und überwachen die Interaktion mit Job-Anhängen und Protokollen

Deadline Cloud erstellt separate CloudWatch Logs-Protokollgruppen für jede Warteschlange. Jobs verwenden Anmeldeinformationen für Warteschlangenrollen, um Protokolle in die Protokollgruppe ihrer Warteschlange zu schreiben. Die CLI und der Monitor von Deadline Cloud verwenden die Warteschlangenrolle (durch`deadline:AssumeQueueRoleForRead`), um Jobprotokolle aus der Protokollgruppe der Warteschlange zu lesen. Die CLI und der Monitor von Deadline Cloud verwenden die Warteschlangenrolle (durch`deadline:AssumeQueueRoleForUser`), um Daten aus Jobanhängen hoch- oder herunterzuladen.

Rolle überwachen

Konfigurieren Sie eine Monitorrolle, um den Web- und Desktop-Anwendungen von Deadline Cloud Monitor Zugriff auf Ihre Deadline Cloud-Ressourcen zu gewähren.

Wenn Sie Monitore programmgesteuert erstellen oder aktualisieren, geben Sie den ARN der Monitorrolle mithilfe der `UpdateMonitor` API-Operationen `CreateMonitor` oder an.

Was macht die Monitorrolle

Die Monitorrolle ermöglicht es Deadline Cloud Monitor, Endbenutzern Zugriff auf Folgendes zu gewähren:

- Grundlegende Funktionen, die für Deadline Cloud Integrated Submitters, CLI und Monitor erforderlich sind
- Benutzerdefinierte Funktionen für Endbenutzer

Richten Sie die Vertrauensrichtlinie für die Monitorrolle ein

Ihre Monitorrolle muss dem Deadline Cloud-Dienst vertrauen.

Als bewährte Methode sollte die Vertrauensrichtlinie Sicherheitsbedingungen für den Schutz von Confused Deputy beinhalten. Weitere Informationen zum Schutz vor Confused Deputy finden Sie unter Confused Deputy im Deadline Cloud-Benutzerhandbuch.

`aws:SourceAccount`stellt sicher, dass nur Ressourcen desselben AWS-Konto Unternehmens diese Rolle übernehmen können.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "credentials.deadline.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "YOUR_ACCOUNT_ID"
        }
      }
    }
  ]
}
```

Hängen Sie Berechtigungen für die Monitorrolle an

Ordnen Sie Ihrer Monitorrolle für den grundlegenden Betrieb alle folgenden AWS verwalteten Richtlinien zu:

- AWSDeadlineCloud-UserAccessFarms
- AWSDeadlineCloud-UserAccessFleets
- AWSDeadlineCloud-UserAccessJobs
- AWSDeadlineCloud-UserAccessQueues

So funktioniert die Monitorrolle

Bei Verwendung des Deadline Cloud-Monitors meldet sich ein Dienstbenutzer mit AWS IAM Identity Center (IAM Identity Center) an, und die Monitorrolle wird übernommen. Die angenommenen Rollenanmeldedaten werden von der Monitor-Anwendung verwendet, um die Monitor-Benutzeroberfläche anzuzeigen, einschließlich der Liste der Farmen, Flotten, Warteschlangen und anderer Informationen.

Wenn Sie die Desktop-Anwendung Deadline Cloud Monitor verwenden, werden diese Anmeldeinformationen zusätzlich auf der Workstation mithilfe eines benannten AWS Anmeldeinformationsprofils verfügbar gemacht, das dem vom Endbenutzer angegebenen

Profilnamen entspricht. Weitere Informationen zu benannten Profilen finden Sie im AWS SDK- und Tools-Referenzhandbuch.

Mit diesem benannten Profil greifen die Deadline-CLI und die Einreicher auf Deadline Cloud-Ressourcen zu.

Anpassung der Monitorrolle für fortgeschrittene Anwendungsfälle

Sie können die Monitorrolle anpassen, um zu ändern, was Benutzer auf den einzelnen Zugriffsebenen (Betrachter, Mitwirkender, Manager, Besitzer) tun können, oder um Berechtigungen für erweiterte Workflows hinzuzufügen.

Berechtigungen auf Zugriffsebene anpassen

Die vier AWS verwalteten Richtlinien, die der Monitorrolle zugeordnet sind, steuern, was die einzelnen Zugriffsebenen tun können. Sie können der Monitorrolle benutzerdefinierte Richtlinien hinzufügen, um mithilfe des `deadline:MembershipLevel` Bedingungsschlüssels Berechtigungen für bestimmte Zugriffsebenen zu gewähren oder einzuschränken.

Um es Mitwirkenden beispielsweise zu ermöglichen, Jobs zu aktualisieren und zu stornieren (was normalerweise nur Managern und Inhabern vorbehalten ist), fügen Sie eine Richtlinie wie die folgende hinzu:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "deadline:UpdateJob",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "deadline:MembershipLevel": "CONTRIBUTOR"
        }
      }
    }
  ]
}
```

Mit dieser Richtlinie können Mitwirkende Jobs nicht nur einreichen, sondern auch aktualisieren und stornieren.

Hinzufügen von Berechtigungen für erweiterte Workflows

Sie können der Monitorrolle benutzerdefinierte IAM-Richtlinien hinzufügen, um allen Monitor-Benutzern zusätzliche Berechtigungen zu gewähren. Dies ist nützlich für erweiterte Scripting-Workflows, bei denen Benutzer Zugriff auf Funktionen benötigen, die AWS-Services über die Standardfunktionen von Deadline Cloud hinausgehen.

Beachten Sie diese Richtlinien, wenn Sie Ihre Monitorrolle ändern:

- Entfernen Sie keine der verwalteten Richtlinien. Durch das Entfernen dieser Richtlinien wird die Monitorfunktion beeinträchtigt.

Wie Deadline Cloud Monitor die Anmeldeinformationen für Monitorrollen verwendet

Deadline Cloud Monitor ruft bei der Authentifizierung automatisch die Anmeldeinformationen für die Monitorrolle ab. Diese Funktion ermöglicht es der Desktop-Anwendung, erweiterte Überwachungsfunktionen bereitzustellen, die über das hinausgehen, was in einem Standard-Webbrowser verfügbar ist.

Wenn Sie sich mit Deadline Cloud Monitor anmelden, wird automatisch ein Profil erstellt, das Sie mit dem AWS CLI oder einem anderen AWS Tool verwenden können. Dieses Profil verwendet die Anmeldeinformationen der Monitorrolle, sodass Sie programmgesteuerten Zugriff darauf erhalten, AWS-Services basierend auf den Berechtigungen in Ihrer Monitorrolle.

Deadline Cloud-Einreicher funktionieren genauso: Sie verwenden das von Deadline Cloud Monitor erstellte Profil, um AWS-Services mit den entsprechenden Rollenberechtigungen darauf zuzugreifen.

Erweiterte Anpassung der Deadline Cloud-Rollen

Sie können Deadline Cloud-Rollen um zusätzliche Berechtigungen erweitern, um erweiterte Anwendungsfälle zu ermöglichen, die über grundlegende Rendering-Workflows hinausgehen. Bei diesem Ansatz wird das Zugriffsverwaltungssystem von Deadline Cloud genutzt, um den Zugriff auf zusätzliche Funktionen auf der AWS-Services Grundlage der Warteschlangenmitgliedschaft zu steuern.

Zusammenarbeit im Team mit AWS CodeCommit

Fügen Sie Ihrer Warteschlangenrolle AWS CodeCommit Berechtigungen hinzu, um die Teamzusammenarbeit an Projekt-Repositorys zu ermöglichen. Bei diesem Ansatz wird das Zugriffsverwaltungssystem von Deadline Cloud für zusätzliche Anwendungsfälle verwendet. Nur Benutzer mit Zugriff auf die jeweilige Warteschlange erhalten diese AWS

CodeCommit Berechtigungen, sodass Sie den Zugriff auf das Projekt-Repository über die Warteschlangenmitgliedschaft von Deadline Cloud verwalten können.

Dies ist nützlich für Szenarien, in denen Künstler im Rahmen ihres Rendering-Workflows auf projektspezifische Assets, Skripte oder Konfigurationsdateien zugreifen müssen, die in AWS CodeCommit Repositorys gespeichert sind.

Fügen Sie der Warteschlangenrolle AWS CodeCommit Berechtigungen hinzu

Fügen Sie Ihrer Warteschlangenrolle die folgenden Berechtigungen hinzu, um den AWS CodeCommit Zugriff zu ermöglichen:

```
{
  "Effect": "Allow",
  "Action": [
    "codecommit:GitPull",
    "codecommit:GitPush",
    "codecommit:GetRepository",
    "codecommit:ListRepositories"
  ],
  "Resource": "arn:aws:codecommit:REGION:YOUR_ACCOUNT_ID:PROJECT_REPOSITORY"
}
```

Richten Sie einen Anbieter für Anmeldeinformationen auf den Workstations von Künstlern ein

Konfigurieren Sie jede Künstler-Workstation so, dass sie die Deadline Cloud-Warteschlangenanmeldedaten für den AWS CodeCommit Zugriff verwendet. Diese Einrichtung wird einmal pro Workstation durchgeführt.

Um den Anmeldeinformationsanbieter zu konfigurieren

1. Fügen Sie Ihrer AWS Konfigurationsdatei () `~/.aws/config` ein Credential Provider-Profil hinzu:

   ```
   [profile queue-codecommit]
   credential_process = deadline queue export-credentials --farm-id farm-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX --queue-id queue-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   ```

2. Konfiguriere Git so, dass es dieses Profil für AWS CodeCommit Repositorys verwendet:

   ```
   git config --global credential.https://git-codecommit.REGION.amazonaws.com.helper
    '!aws codecommit credential-helper --profile queue-codecommit $@'
   ```

```
git config --global credential.https://git-
codecommit.REGION.amazonaws.com.UseHttpPath true
```

Ersetze *farm-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX* und *queue-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX* durch deine eigentliche Farm und Warteschlange IDs. *REGION*Ersetze es durch deine AWS Region (zum Beispiel`us-west-2`).

AWS CodeCommit Mit Anmeldedaten für die Warteschlange verwenden

Nach der Konfiguration verwenden Git-Operationen beim Zugriff auf AWS CodeCommit Repositorys automatisch die Anmeldeinformationen der Warteschlangenrolle. Der `deadline queue export-credentials` Befehl gibt temporäre Anmeldeinformationen zurück, die wie folgt aussehen:

```
{
  "Version": 1,
  "AccessKeyId": "ASIA...",
  "SecretAccessKey": "...",
  "SessionToken": "...",
  "Expiration": "2025-11-10T23:02:23+00:00"
}
```

Diese Anmeldeinformationen werden bei Bedarf automatisch aktualisiert, und Git-Operationen funktionieren reibungslos:

```
git clone https://git-codecommit.REGION.amazonaws.com/v1/repos/PROJECT_REPOSITORY
git pull
git push
```

Künstler können jetzt mit ihren Warteschlangenberechtigungen auf Projekt-Repositorys zugreifen, ohne separate Anmeldeinformationen zu benötigen. AWS CodeCommit Nur Benutzer mit Zugriff auf die jeweilige Warteschlange können auf das zugehörige Repository zugreifen. Dies ermöglicht eine detaillierte Zugriffskontrolle über das Warteschlangenmitgliedschaftssystem von Deadline Cloud.

Fehlerbehebung bei Identität und Zugriff auf AWS Deadline Cloud

Verwenden Sie die folgenden Informationen, um häufig auftretende Probleme zu diagnostizieren und zu beheben, die bei der Arbeit mit Deadline Cloud und IAM auftreten können.

Themen

- Ich bin nicht berechtigt, eine Aktion in Deadline Cloud durchzuführen

Ich bin nicht berechtigt, eine Aktion in Deadline Cloud durchzuführen

Wenn Sie eine Fehlermeldung erhalten, dass Sie nicht zur Durchführung einer Aktion berechtigt sind, müssen Ihre Richtlinien aktualisiert werden, damit Sie die Aktion durchführen können.

Der folgende Beispielfehler tritt auf, wenn der IAM-Benutzer `mateojackson` versucht, über die Konsole Details zu einer fiktiven *`my-example-widget`*-Ressource anzuzeigen, jedoch nicht über `deadline:`*`GetWidget`*-Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 deadline:GetWidget on resource: my-example-widget
```

In diesem Fall muss die Richtlinie für den Benutzer `mateojackson` aktualisiert werden, damit er mit der `deadline:`*`GetWidget`*-Aktion auf die *`my-example-widget`*-Ressource zugreifen kann.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich bin nicht berechtigt, iam durchzuführen: PassRole

Wenn Sie eine Fehlermeldung erhalten, dass Sie nicht berechtigt sind, die `iam:PassRole` Aktion durchzuführen, müssen Ihre Richtlinien aktualisiert werden, damit Sie eine Rolle an Deadline Cloud übergeben können.

Einige AWS-Services ermöglichen es Ihnen, eine bestehende Rolle an diesen Dienst zu übergeben, anstatt eine neue Servicerolle oder eine dienstverknüpfte Rolle zu erstellen. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in Deadline Cloud auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine Deadline Cloud-Ressourcen ermöglichen

Sie können eine Rolle erstellen, mit der Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation auf Ihre Ressourcen zugreifen können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Für Dienste, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (ACLs) unterstützen, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen darüber, ob Deadline Cloud diese Funktionen unterstützt, finden Sie unter. So funktioniert Deadline Cloud mit IAM
- Informationen dazu, wie Sie Zugriff auf Ihre Ressourcen gewähren können, AWS-Konten die Ihnen gehören, finden Sie im IAM-Benutzerhandbuch unter Zugriff auf einen IAM-Benutzer in einem anderen AWS-Konto , den Sie besitzen.
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, finden Sie AWS-Konten im IAM-Benutzerhandbuch unter Gewähren des Zugriffs für Dritte.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Konformitätsprüfung für Deadline Cloud

Informationen darüber, ob AWS-Service ein AWS-Services in den Geltungsbereich bestimmter Compliance-Programme fällt, finden Sie unter Umfang nach Compliance-Programm AWS-Services unter . Wählen Sie dort das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter AWS Compliance-Programme AWS .

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter Berichte herunterladen unter .

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. Weitere Informationen zu Ihrer Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services finden Sie in der AWS Sicherheitsdokumentation.

Resilienz in Deadline Cloud

Die AWS globale Infrastruktur basiert auf Availability AWS-Regionen Zones. AWS-Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu Availability Zones AWS-Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

AWS Deadline Cloud sichert keine Daten, die in Ihrem S3-Bucket für Jobanhänge gespeichert sind. Sie können Backups Ihrer Job-Anhangsdaten mit jedem standardmäßigen Amazon S3 S3-Backup-Mechanismus wie S3 Versioning oder AWS Backupaktivieren.

Sicherheit der Infrastruktur in Deadline Cloud

Als verwalteter Service ist AWS Deadline Cloud durch AWS globale Netzwerksicherheit geschützt. Informationen zu AWS Sicherheitsdiensten und zum AWS Schutz der Infrastruktur finden Sie unter AWS Cloud-Sicherheit. Informationen zum Entwerfen Ihrer AWS Umgebung unter Verwendung der bewährten Methoden für die Infrastruktursicherheit finden Sie unter Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Sie verwenden AWS veröffentlichte API-Aufrufe, um über das Netzwerk auf Deadline Cloud zuzugreifen. Kunden müssen Folgendes unterstützen:

- Transport Layer Security (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.

- Verschlüsselungs-Suiten mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

Deadline Cloud unterstützt die Verwendung von AWS PrivateLink Virtual Private Cloud (VPC) - Endpunktrichtlinien nicht. Es verwendet die AWS PrivateLink Standardrichtlinie, die vollen Zugriff auf den Endpunkt gewährt. Weitere Informationen finden Sie im AWS PrivateLink Benutzerhandbuch unter Standard-Endpunktrichtlinie.

Konfiguration und Schwachstellenanalyse in Deadline Cloud

AWS kümmert sich um grundlegende Sicherheitsaufgaben wie das Patchen von Gastbetriebssystemen (OS) und Datenbanken, die Firewall-Konfiguration und die Notfallwiederherstellung. Diese Verfahren wurden von qualifizierten Dritten überprüft und zertifiziert. Weitere Informationen finden Sie in den folgenden Ressourcen:

- Modell der übergreifenden Verantwortlichkeit
- Amazon Web Services: Übersicht über Sicherheitsverfahren (Whitepaper)

AWS Deadline Cloud verwaltet Aufgaben auf vom Service oder vom Kunden verwalteten Flotten:

- Für vom Service verwaltete Flotten verwaltet Deadline Cloud das Gastbetriebssystem.
- Bei vom Kunden verwalteten Flotten sind Sie für die Verwaltung des Betriebssystems verantwortlich.

Weitere Informationen zur Konfiguration und Schwachstellenanalyse für AWS Deadline Cloud finden Sie unter

- Bewährte Sicherheitsmethoden für Deadline Cloud

Serviceübergreifende Confused-Deputy-Prävention

Das Confused-Deputy-Problem ist ein Sicherheitsproblem, bei dem eine juristische Stelle, die nicht über die Berechtigung zum Ausführen einer Aktion verfügt, eine privilegiertere juristische Stelle zwingen kann, die Aktion auszuführen. In kann AWS ein dienstübergreifender Identitätswechsel zum Problem des verwirrten Stellvertreters führen. Ein dienstübergreifender Identitätswechsel

kann auftreten, wenn ein Dienst (der Anruf-Dienst) einen anderen Dienst anruft (den aufgerufenen Dienst). Der aufrufende Service kann manipuliert werden, um seine Berechtigungen zu verwenden, um Aktionen auf die Ressourcen eines anderen Kunden auszuführen, für die er sonst keine Zugriffsberechtigung haben sollte. Um dies zu verhindern, bietet AWS Tools, mit denen Sie Ihre Daten für alle Services mit Serviceprinzipalen schützen können, die Zugriff auf Ressourcen in Ihrem Konto erhalten haben.

Wir empfehlen, die Kontextschlüssel aws:SourceArnund die aws:SourceAccountglobalen Bedingungsschlüssel in Ressourcenrichtlinien zu verwenden, um die Berechtigungen einzuschränken, die der AWS Deadline Cloud Ressource einen anderen Dienst gewähren. Verwenden Sie `aws:SourceArn`, wenn Sie nur eine Ressource mit dem betriebsübergreifenden Zugriff verknüpfen möchten. Verwenden Sie `aws:SourceAccount`, wenn Sie zulassen möchten, dass Ressourcen in diesem Konto mit der betriebsübergreifenden Verwendung verknüpft werden.

Der effektivste Weg, um sich vor dem Verwirrter-Stellvertreter-Problem zu schützen, ist die Verwendung des `aws:SourceArn` globalen Bedingungskontextschlüssels mit dem vollständigen Amazon-Ressourcenname (ARN) der Ressource. Wenn Sie den vollständigen ARN der Ressource nicht kennen oder wenn Sie mehrere Ressourcen angeben, verwenden Sie den globalen Kontextbedingungsschlüssel `aws:SourceArn` mit Platzhalterzeichen (*) für die unbekannten Teile des ARN. Beispiel, `arn:aws:deadline:*:`*`123456789012`*`:*`.

Wenn der `aws:SourceArn`-Wert die Konto-ID nicht enthält, z. B. einen Amazon-S3-Bucket-ARN, müssen Sie beide globale Bedingungskontextschlüssel verwenden, um Berechtigungen einzuschränken.

Das folgende Beispiel zeigt, wie Sie die Kontextschlüssel `aws:SourceArn` und die `aws:SourceAccount` globale Bedingung verwenden können, Deadline Cloud um das Problem des verwirrten Stellvertreters zu vermeiden.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "deadline.amazonaws.com"
    },
```

```
    "Action": "deadline:CreateFarm",
    "Resource": [
      "*"
    ],
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:deadline:*:111122223333:*"
      },
      "StringEquals": {
        "aws:SourceAccount": "111122223333"
      }
    }
  }
}
```

Zugriff AWS Deadline Cloud über einen Schnittstellenendpunkt (AWS PrivateLink)

Sie können verwenden AWS PrivateLink , um eine private Verbindung zwischen Ihrer VPC und AWS Deadline Cloud herzustellen. Sie können darauf zugreifen, Deadline Cloud als ob es in Ihrer VPC wäre, ohne ein Internet-Gateway, ein NAT-Gerät, eine VPN-Verbindung oder Direct Connect eine Verbindung zu verwenden. Instances in Ihrer VPC benötigen für den Zugriff Deadline Cloud keine öffentlichen IP-Adressen.

Sie stellen diese private Verbindung her, indem Sie einen Schnittstellen-Endpunkt erstellen, der von AWS PrivateLink unterstützt wird. Wir erstellen eine Endpunkt-Netzwerkschnittstelle in jedem Subnetz, das Sie für den Schnittstellen-Endpunkt aktivieren. Hierbei handelt es sich um vom Anforderer verwaltete Netzwerkschnittstellen, die als Eingangspunkt für den Datenverkehr dienen, der für Deadline Cloud bestimmt ist.

Deadline Cloud verfügt auch über Dual-Stack-Endpunkte. Dual-Stack-Endpunkte unterstützen Anfragen über und. IPv6 IPv4

Weitere Informationen finden Sie unter Zugriff auf AWS-Services über AWS PrivateLink im AWS PrivateLink -Leitfaden.

Überlegungen zu Deadline Cloud

Bevor Sie einen Schnittstellenendpunkt für einrichten Deadline Cloud, finden Sie weitere Informationen unter Zugreifen auf einen AWS-Service mithilfe eines Schnittstellen-VPC-Endpunkts im AWS PrivateLink Handbuch.

Deadline Cloud unterstützt Aufrufe aller API-Aktionen über den Schnittstellenendpunkt.

Standardmäßig Deadline Cloud ist der vollständige Zugriff auf über den Schnittstellenendpunkt zulässig. Alternativ können Sie den Endpunkt-Netzwerkschnittstellen eine Sicherheitsgruppe zuordnen, um den Datenverkehr Deadline Cloud über den Schnittstellenendpunkt zu kontrollieren.

Deadline Cloud unterstützt auch VPC-Endpunktrichtlinien. Weitere Informationen finden Sie unter Kontrolle des Zugriffs auf VPC-Endpunkte mit Endpunktrichtlinien im AWS PrivateLink Leitfaden.

Deadline Cloud Endpunkte

Deadline Cloud verwendet vier Endpunkte für den Zugriff auf den Dienst mithilfe von AWS PrivateLink — zwei für IPv4 und zwei für. IPv6

Mitarbeiter verwenden den `scheduling.deadline.region.amazonaws.com` Endpunkt, um Aufgaben aus der Warteschlange abzurufen, ihnen den Fortschritt zu Deadline Cloud melden und die Aufgabenausgabe zurückzusenden. Wenn Sie eine vom Kunden verwaltete Flotte verwenden, ist der Terminplanungsendpunkt der einzige Endpunkt, den Sie erstellen müssen, es sei denn, Sie verwenden Verwaltungsoperationen. Wenn durch einen Auftrag beispielsweise mehr Jobs erstellt werden, müssen Sie den Verwaltungsendpunkt so einrichten, dass er den `CreateJob` Vorgang aufrufen kann.

Der Deadline Cloud Monitor verwendet den, `management.deadline.region.amazonaws.com` um die Ressourcen in Ihrer Farm zu verwalten, z. B. Warteschlangen und Flotten zu erstellen und zu ändern oder Listen mit Aufträgen, Schritten und Aufgaben abzurufen.

Die AWS SDKs AND-CLI fügt dem Endpunkt automatisch die `scheduling` Präfixe `management` und hinzu. Wenn Sie dieses Verhalten deaktivieren möchten, finden Sie im Abschnitt Host-Präfix-Injection im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch weitere Informationen.

Deadline Cloud erfordert außerdem Endpunkte für die folgenden AWS Dienstendpunkte:

- Deadline Cloud verwendet AWS STS , um Mitarbeiter zu authentifizieren, sodass sie auf Arbeitsressourcen zugreifen können. Weitere Informationen zu AWS STS finden Sie unter

Temporäre Sicherheitsanmeldeinformationen in IAM im AWS Identity and Access Management Benutzerhandbuch.

- Wenn Sie Ihre vom Kunden verwaltete Flotte in einem Subnetz ohne Internetverbindung einrichten, müssen Sie einen VPC-Endpunkt für Amazon CloudWatch Logs einrichten, damit Mitarbeiter Protokolle schreiben können. Weitere Informationen finden Sie unter Überwachung mit. CloudWatch
- Wenn Sie Jobanhänge verwenden, müssen Sie einen VPC-Endpunkt für Amazon Simple Storage Service (Amazon S3) erstellen, damit Mitarbeiter auf die Anlagen zugreifen können. Weitere Informationen finden Sie unter Jobanhänge in Deadline Cloud.

Erstellen Sie Endpunkte für Deadline Cloud

Sie können Schnittstellen-Endpunkte für die Deadline Cloud Verwendung entweder der Amazon VPC-Konsole oder der AWS Command Line Interface ()AWS CLI erstellen. Weitere Informationen finden Sie unter Erstellen eines Schnittstellenendpunkts im AWS PrivateLink -Leitfaden.

Erstellen Sie Verwaltungs- und Scheduling-Endpunkte für die Deadline Cloud Verwendung der folgenden Servicenamen. Ersetzen Sie es *region* durch den AWS-Region Ort, an dem Sie es bereitgestellt Deadline Cloud haben.

```
com.amazonaws.region.deadline.management
```

```
com.amazonaws.region.deadline.scheduling
```

Deadline Cloud unterstützt Dual-Stack-Endpunkte.

Wenn Sie privates DNS für die Schnittstellenendpunkte aktivieren, können Sie API-Anfragen an die Deadline Cloud Verwendung des standardmäßigen regionalen DNS-Namens stellen. Zum Beispiel `scheduling.deadline.us-east-1.amazonaws.com` für Arbeitsoperationen oder `management.deadline.us-east-1.amazonaws.com` für alle anderen Operationen.

Sie müssen auch einen Endpunkt für die AWS STS Verwendung des folgenden Dienstnamens erstellen:

```
com.amazonaws.region.sts
```

Wenn sich Ihre vom Kunden verwaltete Flotte in einem Subnetz ohne Internetverbindung befindet, müssen Sie einen CloudWatch Logs-Endpunkt mit dem folgenden Dienstnamen erstellen:

```
com.amazonaws.region.logs
```

Wenn Sie Auftragsanhänge zum Übertragen von Dateien verwenden, müssen Sie einen Amazon S3 S3-Endpunkt mit dem folgenden Servicenamen erstellen:

```
com.amazonaws.region.s3
```

Eingeschränkte Netzwerkumgebungen

Deadline Cloud bietet Tools, die von Künstlern oder anderen Benutzern auf ihren lokalen Workstations verwendet werden. Diese Tools benötigen Zugriff auf AWS API und Web-Endpunkte, um ihre Funktion zu erfüllen. Wenn Sie den Zugriff auf bestimmte AWS Domänen oder URL-Endpunkte mithilfe einer Lösung zur Filterung von Webinhalten wie Firewalls der nächsten Generation (NGFW) oder Secure Web Gateways (SWG) filtern, müssen Sie die folgenden Domänen oder URL-Endpunkte zu Ihren Zulassungslisten für Webinhaltsfilterlösungen hinzufügen.

AWS API-Endpunkte zur Zulassungsliste

Deadline Cloud-Client-Tools wie Monitor AWS-Managementkonsole, CLI und integrierte Submitter benötigen zusätzlich zu AWS APIs Deadline Cloud Zugriff auf. Diese Endgeräte unterstützen nur. IPv4

- `scheduling.deadline.[Region].amazonaws.com`
- `management.deadline.[Region].amazonaws.com`
- `logs.[Region].amazonaws.com`
- `ec2.[Region].amazonaws.com`
- `s3.[Region].amazonaws.com`
- `sts.[Region].amazonaws.com`
- `identitystore.[Region].amazonaws.com`

Webdomänen, die zugelassen werden sollen

Der Deadline Cloud-Monitor benötigt für den Betrieb Zugriff auf die folgenden Domänen.

Weitere Informationen zur Zulassung von Domains für die AWS Anmeldung finden Sie im Anmelde-Benutzerhandbuch unter Domains, die Sie Ihrer Zulassungsliste hinzufügen AWS sollten.

- downloads.deadlinecloud.amazonaws.com
- d2ev1rdnjzhmnr.cloudfront.net
- prod.log.shortbread.aws.dev
- prod.tools.shortbread.aws.dev
- prod.log.shortbread.analytics.console.aws.a2z.com
- prod.tools.shortbread.analytics.console.aws.a2z.com
- global.help-panel.docs.aws.a2z.com
- *[Region]*.signin.aws
- *[Region]*.signin.aws.amazon.com
- sso.*[Region]*.amazonaws.com
- portal.sso.*[Region]*.amazonaws.com
- oidc.*[Region]*.amazonaws.com
- assets.sso-portal.*[Region]*.amazonaws.com

Der Deadline Cloud-Einreicher benötigt Zugriff auf die folgenden Domänen, um GUI-Abhängigkeiten herunterladen zu können.

- pypi.python.org
- pypi.org
- pythonhosted.org
- files.pythonhosted.org

Umgebungsspezifische Endpunkte, die zugelassen werden sollen

Diese Domänen variieren je nach der spezifischen Konfiguration von Deadline Cloud. Wenn zusätzliche Deadline Cloud-Monitore oder -Warteschlangen erstellt werden, müssen zusätzliche Domains auf die Zulassungsliste gesetzt werden.

- *[Directory ID or alias]*.awsapps.com

  Diese Domain ist an das IAM Identity Center-Setup gebunden und sollte für alle Setups, die dieselbe IAM Identity Center-Instanz verwenden, identisch sein. Den genauen Wert finden Sie vom Unternehmensadministrator in der IAM Identity Center-Konsole unter Einstellungen → URL.AWS-Zugangsportal

- *[Monitor alias]*.*[Region]*.deadlinecloud.amazonaws.com

  Diese Domain ist für das Monitor-Setup in Deadline Cloud vorgesehen. Künstler geben diesen Link in ihren Browser oder ihre Deadline Cloud-Monitoranwendung ein. Wenn Deadline Cloud in future in weiteren Konten oder Regionen eingerichtet wird, wird sich diese Domain ändern. Sie finden diesen Wert in der Deadline Cloud-Konsole im Dashboard → Monitorübersicht → Monitordetails → URL.

- *[Bucket name]*.*[Region]*.s3.amazonaws.com

  Dies ist die Domain für den Bucket mit Job-Anhängen, der von Deadline Cloud-Warteschlangen verwendet wird. Für jede Warteschlange kann ein eigener Bucket für Jobanhänge konfiguriert werden. Den genauen Bucket-Namen finden Sie in der Deadline Cloud-Konsole unter Warteschlangen → Warteschlangendetails → Jobanhänge. Weitere Informationen zu Job-Anhängen finden Sie in der Queues-Dokumentation.

Bewährte Sicherheitsmethoden für Deadline Cloud

AWS Deadline Cloud (Deadline Cloud) bietet eine Reihe von Sicherheitsfunktionen, die Sie bei der Entwicklung und Implementierung Ihrer eigenen Sicherheitsrichtlinien berücksichtigen sollten. Die folgenden bewährten Methoden sind allgemeine Richtlinien und keine vollständige Sicherheitslösung. Da diese bewährten Methoden für Ihre Umgebung möglicherweise nicht angemessen oder ausreichend sind, sollten Sie sie als hilfreiche Überlegungen und nicht als bindend ansehen.

> **Note**
>
> Weitere Informationen zur Bedeutung vieler Sicherheitsthemen finden Sie im Modell der gemeinsamen Verantwortung.

Datenschutz

Aus Datenschutzgründen empfehlen wir, dass Sie Ihre AWS-Konto Anmeldeinformationen schützen und individuelle Konten mit AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).

- Wird verwendet SSL/TLS , um mit AWS Ressourcen zu kommunizieren. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein AWS CloudTrail.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.
- Verwenden Sie fortschrittliche verwaltete Sicherheitsdienste wie Amazon Macie, die Sie bei der Erkennung und Sicherung personenbezogener Daten unterstützen, die in Amazon Simple Storage Service (Amazon S3) gespeichert sind.
- Wenn Sie für den Zugriff auf AWS über eine Befehlszeilenschnittstelle oder über eine API FIPS 140-2-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-2.

Wir empfehlen dringend, in Freitextfeldern wie z. B. im Feld Name keine sensiblen, identifizierenden Informationen wie Kontonummern von Kunden einzugeben. Diese Empfehlung gilt auch, wenn Sie mit AWS Deadline Cloud oder anderen AWS-Services über die Konsole AWS CLI, API oder AWS SDKs arbeiten. Alle Daten, die Sie in Deadline Cloud oder andere Dienste eingeben, werden möglicherweise zur Aufnahme in Diagnoseprotokolle aufgenommen. Wenn Sie eine URL für einen externen Server bereitstellen, schließen Sie keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL ein.

AWS Identity and Access Management Berechtigungen

Verwalten Sie den Zugriff auf AWS Ressourcen mithilfe von Benutzern und AWS Identity and Access Management (IAM-) Rollen und indem Sie Benutzern die geringsten Rechte gewähren. Richten Sie Richtlinien und Verfahren zur Verwaltung von Anmeldeinformationen für die Erstellung, Verteilung, Rotation und den Widerruf AWS von Zugangsdaten ein. Weitere Informationen finden Sie unter Bewährte Methoden für IAM im IAM-Benutzerhandbuch.

Führen Sie Jobs als Benutzer und Gruppen aus

Bei der Verwendung der Warteschlangenfunktion in Deadline Cloud hat es sich bewährt, einen Betriebssystembenutzer (OS) und seine primäre Gruppe anzugeben, sodass der Betriebssystembenutzer die geringsten Rechte für die Jobs der Warteschlange hat.

Wenn Sie die Option „Als Benutzer ausführen“ (und Gruppe) angeben, werden alle Prozesse für Jobs, die an die Warteschlange gesendet werden, mit diesem Betriebssystembenutzer ausgeführt und erben die zugehörigen Betriebssystemberechtigungen dieses Benutzers.

Die Kombination der Flotten- und Warteschlangenkonfigurationen sorgt für ein gewisses Maß an Sicherheit. Auf der Warteschlangenseite können die Rolle „Job wird als Benutzer ausgeführt“ und die IAM-Rolle angegeben werden, um das Betriebssystem und die AWS Berechtigungen für die Jobs der Warteschlange zu verwenden. Die Flotte definiert die Infrastruktur (Worker-Hosts, Netzwerke, bereitgestellter gemeinsam genutzter Speicher), über die Jobs innerhalb der Warteschlange ausgeführt werden, sofern sie einer bestimmten Warteschlange zugeordnet sind. Auf die auf den Worker-Hosts verfügbaren Daten müssen Jobs aus einer oder mehreren zugehörigen Warteschlangen zugreifen können. Die Angabe eines Benutzers oder einer Gruppe trägt dazu bei, die Daten in Jobs vor anderen Warteschlangen, anderer installierter Software oder anderen Benutzern mit Zugriff auf die Worker-Hosts zu schützen. Wenn es in einer Warteschlange keinen Benutzer gibt, wird sie als Agent-Benutzer ausgeführt, der sich als (`sudo`) beliebiger Warteschlangenbenutzer ausgeben kann. Auf diese Weise kann eine Warteschlange ohne Benutzer Rechte an eine andere Warteschlange weiterleiten.

Netzwerk

Um zu verhindern, dass der Datenverkehr abgefangen oder umgeleitet wird, müssen Sie unbedingt sicherstellen, wie und wohin Ihr Netzwerkverkehr geleitet wird.

Wir empfehlen Ihnen, Ihre Netzwerkumgebung auf folgende Weise zu sichern:

- Sichere Subnetz-Routing-Tabellen für Amazon Virtual Private Cloud (Amazon VPC), um zu kontrollieren, wie der Datenverkehr auf IP-Ebene weitergeleitet wird.
- Wenn Sie Amazon Route 53 (Route 53) als DNS-Anbieter in Ihrem Farm- oder Workstation-Setup verwenden, sichern Sie den Zugriff auf die Route 53-API.
- Wenn Sie eine Verbindung zu Deadline Cloud außerhalb herstellen, AWS z. B. über lokale Workstations oder andere Rechenzentren, sichern Sie jede lokale Netzwerkinfrastruktur. Dazu gehören DNS-Server und Routing-Tabellen auf Routern, Switches und anderen Netzwerkgeräten.

Jobs und Jobdaten

Deadline Cloud-Jobs werden innerhalb von Sitzungen auf Worker-Hosts ausgeführt. In jeder Sitzung werden ein oder mehrere Prozesse auf dem Worker-Host ausgeführt. Für die Ausgabe müssen Sie in der Regel Daten eingeben.

Um diese Daten zu sichern, können Sie Betriebssystembenutzer mit Warteschlangen konfigurieren. Der Worker-Agent verwendet den Warteschlangen-OS-Benutzer, um Sitzungsunterprozesse auszuführen. Diese Unterprozesse erben die Berechtigungen des Queue-OS-Benutzers.

Wir empfehlen, dass Sie sich an bewährte Methoden halten, um den Zugriff auf die Daten, auf die diese Unterprozesse zugreifen, zu sichern. Weitere Informationen finden Sie unter Modell der geteilten Verantwortung.

Struktur der Farm

Sie können Deadline Cloud-Flotten und Warteschlangen auf viele Arten anordnen. Bestimmte Vereinbarungen haben jedoch Auswirkungen auf die Sicherheit.

Eine Farm hat eine der sichersten Grenzen, da sie Deadline Cloud-Ressourcen nicht mit anderen Farmen teilen kann, einschließlich Flotten, Warteschlangen und Speicherprofilen. Sie können jedoch externe AWS Ressourcen innerhalb einer Farm gemeinsam nutzen, wodurch die Sicherheitsgrenze gefährdet wird.

Mit der entsprechenden Konfiguration können Sie auch Sicherheitsgrenzen zwischen Warteschlangen innerhalb derselben Farm einrichten.

Folgen Sie diesen bewährten Methoden, um sichere Warteschlangen in derselben Farm zu erstellen:

- Ordnen Sie eine Flotte nur Warteschlangen innerhalb derselben Sicherheitsgrenze zu. Beachten Sie Folgendes:
  - Nachdem der Job auf dem Worker-Host ausgeführt wurde, können Daten zurückbleiben, z. B. in einem temporären Verzeichnis oder im Home-Verzeichnis des Warteschlangenbenutzers.
  - Derselbe Betriebssystembenutzer führt alle Jobs auf einem firmeneigenen Fleet-Worker-Host aus, unabhängig davon, an welche Warteschlange Sie den Job senden.
  - Ein Job kann dazu führen, dass Prozesse auf einem Worker-Host ausgeführt werden, sodass Jobs aus anderen Warteschlangen andere laufende Prozesse beobachten können.
- Stellen Sie sicher, dass sich nur Warteschlangen innerhalb derselben Sicherheitsgrenze einen Amazon S3 S3-Bucket für Jobanhänge teilen.
- Stellen Sie sicher, dass nur Warteschlangen innerhalb derselben Sicherheitsgrenze denselben Betriebssystembenutzer verwenden.
- Sichern Sie alle anderen AWS Ressourcen, die in die Farm integriert sind, bis zur Grenze.

Warteschlangen für Arbeitsanhänge

Jobanhänge sind mit einer Warteschlange verknüpft, die Ihren Amazon S3 S3-Bucket verwendet.

- Auftragsanhänge schreiben in ein Root-Präfix im Amazon S3 S3-Bucket und lesen aus diesem. Sie geben dieses Root-Präfix im `CreateQueue` API-Aufruf an.
- Der Bucket hat ein entsprechendes`Queue Role`, das die Rolle spezifiziert, die Warteschlangenbenutzern Zugriff auf den Bucket und das Root-Präfix gewährt. Beim Erstellen einer Warteschlange geben Sie den `Queue Role` Amazon-Ressourcennamen (ARN) zusammen mit dem Bucket und dem Root-Präfix für Jobanhänge an.
- Autorisierte Aufrufe von `AssumeQueueRoleForRead`,`AssumeQueueRoleForUser`, und `AssumeQueueRoleForWorker` API-Operationen geben einen Satz temporärer Sicherheitsanmeldedaten für die zurück`Queue Role`.

Wenn Sie eine Warteschlange erstellen und einen Amazon S3 S3-Bucket und ein Root-Präfix wiederverwenden, besteht die Gefahr, dass Informationen an Unbefugte weitergegeben werden. QueueA und QueueB verwenden beispielsweise denselben Bucket und dasselbe Root-Präfix. In einem sicheren Workflow hat ArtistA Zugriff auf QueueA, aber nicht auf QueueB. Wenn sich jedoch mehrere Warteschlangen einen Bucket teilen, kann ArtistA auf die Daten in QueueB-Daten zugreifen, da es denselben Bucket und dasselbe Root-Präfix wie QueueA verwendet.

Die Konsole richtet Warteschlangen ein, die standardmäßig sicher sind. Stellen Sie sicher, dass die Warteschlangen eine eindeutige Kombination aus Amazon S3 S3-Bucket und Root-Präfix haben, sofern sie nicht Teil einer gemeinsamen Sicherheitsgrenze sind.

Um Ihre Warteschlangen zu isolieren, müssen Sie das so konfigurieren, `Queue Role` dass nur der Warteschlangenzugriff auf den Bucket und das Root-Präfix zulässig ist. Ersetzen Sie im folgenden Beispiel jedes *`placeholder`* durch Ihre ressourcenspezifischen Informationen.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
```

```
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::JOB_ATTACHMENTS_BUCKET_NAME",

 "arn:aws:s3:::JOB_ATTACHMENTS_BUCKET_NAME/JOB_ATTACHMENTS_ROOT_PREFIX/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "111122223333"
                }
            }
        },
        {
            "Action": [
                "logs:GetLogEvents"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:logs:us-east-1:111122223333:log-group:/aws/
deadline/FARM_ID/*"
        }
    ]
}
```

Sie müssen außerdem eine Vertrauensrichtlinie für die Rolle festlegen. Ersetzen Sie im folgenden Beispiel den *placeholder* Text durch Ihre ressourcenspezifischen Informationen.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "sts:AssumeRole"
            ],
            "Effect": "Allow",
            "Principal": {
                "Service": "deadline.amazonaws.com"
```

```
            },
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:deadline:us-
east-1:111122223333:farm/FARM_ID"
                }
            }
        },
        {
            "Action": [
                "sts:AssumeRole"
            ],
            "Effect": "Allow",
            "Principal": {
                "Service": "credentials.deadline.amazonaws.com"
            },
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:deadline:us-
east-1:111122223333:farm/FARM_ID"
                }
            }
        }
    ]
}
```

Amazon S3 S3-Buckets mit benutzerdefinierter Software

Sie können die folgende Anweisung zu Ihrem hinzufügen, `Queue Role` um auf benutzerdefinierte Software in Ihrem Amazon S3 S3-Bucket zuzugreifen. Im folgenden Beispiel *SOFTWARE_BUCKET_NAME* ersetzen Sie es durch den Namen Ihres S3-Buckets und *BUCKET_ACCOUNT_OWNER* durch die AWS-Konto ID, der der Bucket gehört.

```
"Statement": [
    {
        "Action": [
```

```
            "s3:GetObject",
            "s3:ListBucket"
        ],
        "Effect": "Allow",
        "Resource": [
            "arn:aws:s3:::SOFTWARE_BUCKET_NAME",
            "arn:aws:s3:::SOFTWARE_BUCKET_NAME/*"
        ],
        "Condition": {
         "StringEquals": {
            "aws:ResourceAccount": "BUCKET_ACCOUNT_OWNER"
         }
      }
    }
]
```

Weitere Informationen zu den bewährten Sicherheitsmethoden von Amazon S3 finden Sie unter Bewährte Sicherheitsmethoden für Amazon S3 im Amazon Simple Storage Service-Benutzerhandbuch.

Worker-Hosts

Schützen Sie Worker-Hosts, um sicherzustellen, dass jeder Benutzer nur Operationen für die ihm zugewiesene Rolle ausführen kann.

Wir empfehlen die folgenden bewährten Methoden zur Sicherung von Worker-Hosts:

- Die Verwendung eines Host-Konfigurationsskripts kann die Sicherheit und den Betrieb eines Workers ändern. Eine falsche Konfiguration kann dazu führen, dass der Worker instabil ist oder nicht mehr funktioniert. Es liegt in Ihrer Verantwortung, solche Fehler zu debuggen.
- Verwenden Sie nicht denselben `jobRunAsUser` Wert für mehrere Warteschlangen, es sei denn, an diese Warteschlangen übermittelte Jobs liegen innerhalb derselben Sicherheitsgrenze.
- Stellen Sie die Warteschlange nicht `jobRunAsUser` auf den Namen des Betriebssystembenutzers ein, unter dem der Worker-Agent ausgeführt wird.
- Gewähren Sie Warteschlangenbenutzern die Betriebssystemberechtigungen mit den geringsten Rechten, die für die vorgesehenen Warteschlangenworkloads erforderlich sind. Stellen Sie sicher, dass sie keine Dateisystem-Schreibberechtigungen für Work-Agent-Programmdateien oder andere gemeinsam genutzte Software haben.
- Stellen Sie sicher, dass nur der Root-Benutzer Linux und das Konto `Administrator` Eigentümer der Worker-Agent-Programmdateien sind und diese ändern können. Windows

- Auf Linux Worker-Hosts sollten Sie erwägen, einen `umask` Override-Vorgang zu konfigurieren`/etc/sudoers`, der es dem Worker-Agent-Benutzer ermöglicht, Prozesse als Warteschlangenbenutzer zu starten. Diese Konfiguration trägt dazu bei, dass andere Benutzer nicht auf Dateien zugreifen können, die in die Warteschlange geschrieben wurden.
- Gewähren Sie vertrauenswürdigen Personen den Zugriff auf Worker-Hosts mit den geringsten Rechten.
- Beschränken Sie die Berechtigungen auf lokale DNS-Override-Konfigurationsdateien (`/etc/hosts`aktiviert Linux und aktiviertWindows) sowie `C:\Windows\system32\etc\hosts` auf das Routing von Tabellen auf Workstations und Worker-Host-Betriebssystemen.
- Beschränken Sie die Berechtigungen für die DNS-Konfiguration auf Workstations und Worker-Host-Betriebssystemen.
- Patchen Sie regelmäßig das Betriebssystem und die gesamte installierte Software. Dieser Ansatz umfasst Software, die speziell mit Deadline Cloud verwendet wird, wie z. B. Einreicher, Adapter, Worker Agents, OpenJD Pakete und andere.
- Verwenden Sie sichere Passwörter für die Warteschlange. Windows `jobRunAsUser`
- Wechseln Sie regelmäßig die Passwörter für Ihre Warteschlange`jobRunAsUser`.
- Sorgen Sie für den Zugriff auf die Windows Kennwortgeheimnisse mit den geringsten Rechten und löschen Sie ungenutzte Geheimnisse.
- Erteilen Sie der Warteschlange nicht die `jobRunAsUser` Erlaubnis, die Befehle für den Zeitplan in der future auszuführen:
  - EinLinux, verweigern Sie diesen Konten den Zugriff auf `cron` und`at`.
  - Ist diese Windows Option aktiviert, wird diesen Konten der Zugriff auf den Windows Taskplaner verweigert.

> **Note**
>
> Weitere Informationen darüber, wie wichtig es ist, das Betriebssystem und die installierte Software regelmäßig zu patchen, finden Sie im Modell der gemeinsamen Verantwortung.

Host-Konfigurationsskript

- Die Verwendung eines Host-Konfigurationsskripts kann die Sicherheit und den Betrieb eines Workers ändern. Eine falsche Konfiguration kann dazu führen, dass der Worker instabil ist oder nicht mehr funktioniert. Es liegt in Ihrer Verantwortung, solche Fehler zu debuggen.

Workstations

Es ist wichtig, Workstations mit Zugriff auf Deadline Cloud zu sichern. Dieser Ansatz trägt dazu bei, dass mit Jobs, die Sie an Deadline Cloud einreichen, keine beliebigen Workloads ausgeführt werden können, die Ihnen in Rechnung gestellt werden. AWS-Konto

Wir empfehlen die folgenden bewährten Methoden zur Sicherung von Künstler-Workstations. Weitere Informationen finden Sie unter -Modell der geteilten Verantwortung.

- Sichern Sie alle dauerhaften Anmeldeinformationen, die Zugriff auf, einschließlich Deadline AWS Cloud, ermöglichen. Weitere Informationen finden Sie unter Verwalten der Zugriffsschlüssel für IAM-Benutzer im -IAM-Benutzerhandbuch.
- Installieren Sie nur vertrauenswürdige, sichere Software.
- Erfordern Sie, dass Benutzer sich mit einem Identitätsanbieter zusammenschließen, um AWS mit temporären Anmeldeinformationen zugreifen zu können.
- Verwenden Sie sichere Berechtigungen für die Programmdateien von Deadline Cloud-Absendern, um Manipulationen zu verhindern.
- Gewähren Sie vertrauenswürdigen Personen den Zugriff auf die Workstations von Künstlern mit den geringsten Rechten.
- Verwenden Sie nur Einreicher und Adapter, die Sie über den Deadline Cloud Monitor erhalten.
- Beschränken Sie die Berechtigungen auf lokale DNS-Override-Konfigurationsdateien (`/etc/hosts`an Linux undmacOS, und `C:\Windows\system32\etc\hosts` anWindows) und auf das Routing von Tabellen auf Workstations und Worker-Host-Betriebssystemen.
- Beschränken Sie die Berechtigungen `/etc/resolve.conf` auf Workstations und Worker-Host-Betriebssystemen.
- Patchen Sie regelmäßig das Betriebssystem und die gesamte installierte Software. Dieser Ansatz umfasst Software, die speziell mit Deadline Cloud verwendet wird, wie z. B. Einreicher, Adapter, Worker Agents, OpenJD Pakete und andere.

Überprüfen Sie die Echtheit der heruntergeladenen Software

Überprüfen Sie nach dem Herunterladen des Installationsprogramms die Echtheit Ihrer Software, um sie vor Dateimanipulationen zu schützen. Dieses Verfahren funktioniert sowohl für als auch für SystemeWindows. Linux

Windows

Gehen Sie wie folgt vor, um die Echtheit Ihrer heruntergeladenen Dateien zu überprüfen.

1. Ersetzen Sie den Befehl im folgenden Befehl *file* durch die Datei, die Sie überprüfen möchten. Beispiel, ***C:\PATH\TO\MY*DeadlineCloudSubmitter-windows-x64-installer.exe** . Ersetzen Sie es außerdem *signtool-sdk-version* durch die Version des installierten SignTool SDK. Beispiel, **10.0.22000.0**.

   ```
   "C:\Program Files (x86)\Windows Kits\10\bin\signtool-sdk-version\x86\signtool.exe" verify /vfile
   ```

2. Sie können beispielsweise die Installationsdatei für den Deadline Cloud-Absender überprüfen, indem Sie den folgenden Befehl ausführen:

   ```
   "C:\Program Files (x86)\Windows Kits\10\bin
   \10.0.22000.0\x86\signtool.exe" verify /v DeadlineCloudSubmitter-windows-x64-installer.exe
   ```

Linux

Verwenden Sie das gpg Befehlszeilentool, um die Echtheit Ihrer heruntergeladenen Dateien zu überprüfen.

1. Importieren Sie den OpenPGP Schlüssel, indem Sie den folgenden Befehl ausführen:

   ```
    gpg --import --armor <<EOF
   -----BEGIN PGP PUBLIC KEY BLOCK-----

   mQINBGlANDUBEACg6zffjN43gqe5ryPhk+wQM10rEdvmItw4WPWaVsN+/at/OIJw
   MGCagSYXcgR+jKbsHQOQoEQdo5SrxxHjpKTEs3KQhGvf+ehrU1Ac7koXKIBWtes+
   BI9F0slRECz0nXTOy/cd/90RXjpF07mreTLIKNIbybULfad82nYykpITjFr5XRGj
   /shYkucxRQZdwkgkIYyV25pPICPd2RsX+Zua85jV8mCqVffDfRXvgcPe3+ofClj/
   2CE8UfUIqO8Csua4YEkSqr3aeoTOEFT4kuQR5nFXVzorOEkQtO3gB35KNWKMlIOU
   2vA+wyoL7nWSii4yfYtW3EZ+3gq6HxvnT9Zs8MC53uTOiOdamASXecYREwGmY/io
   ```

```
6n5XTEA/35LNbl4A756vSTZ7h4VFJAN5BpuqxstI1D7ou94skoSmcPoC/iniTvY9
kZylU5OCH/nifMAHM2a5jrQel80cW4oko9eyc8ENQpSy15JElFOKFf7D/4tcZJLF
F0VBTXbhfvq3dPfoq94IWt7p54Ovwj0S//CEu3jZYbNl2QC/3YiHE2H2XyGCQbq6
2MjcuxLnEapoRIqfbi8GPtCWVPzm28WGyKIDofWICczzeJFFJnvzrY3wRG64ibKJ
bR/uedwua1UuiC482V1FD5ffmzSSs8ktTp9hgj7RGDXlc9NTcF1jHxG9hwARAQAB
tCxBV1MgRGVhZGxpbmUgQ2xvdWQgPGF3cy1kZWFkbGluZUBhbWF6b24uY29tPokC
VwQTAQgAQRYhBJmXd7So2csyehiIYsg71N18bhtjBQJpQDQ1AhsvBQkDwmcABQsJ
CAcCAiICBhUKCQgLAgQWAgMBAh4HAheAAAoJEMg71N18bhtjk2UP/3h4KlEzZO/7
BxRmkbixuo1QuqOGvA6tXbSWaM8QH5jglcvL12PZLALklLT4v82uCsLR1lF8/Tch
cCl0SZEOFIS+XxAaw1Xfai6jlyLhabOwKF2ylq5eJlLcw1lh2nAArDRb4fLD0m1g
Dfqetq/XEpyXpOSkWxGRV4RlUdjQfytxrmcUnsT5/fk5f9VDdblu6K/lEmwfyYjB
lXv0uUCkqPot0SmbvOh3PY3Hi3n54ncy8NfTeV+TUvSe3C1s1zNl8aqHoTxJB/eU
kp+LFZ9m+igpSYnKeglKnytylH3KGCjTHglT/QXnI1wNTqmj1kFBVwtt/y1mtnA+
CPIUHP1CtbKsHaLtpp4llBm5TVtPN/Wqqicn5QLl4khg7R4K+V2aaA4ubY6p1tG9
0fFhN5tTnHDSKWMfmb83wfh5Zkcg85c3egjoit+wgGQRAQVqbznx7NqAHs9VoDIu
SPcAr+C329AOBzod4gyNGH7Ah5DkMITo4O4+axnAU9yhFOHcMJmTIask/fNg1Aum
OqYPMUwcgv1GZjLaTJyfGGC1xALsYR0KHnwIehD06MHR/Z98bGkcV8+Y0q8UPsd1
VN1fc1rjCJh/AT3w6owvG4DaEwspseSjzHv16mW4e2N6Uu23SPzgQsJ5qYN2g8D+
P7N9LGDfP8DaYc5JM9mlyFmYI2Q94ufL
=rY5l
-----END PGP PUBLIC KEY BLOCK-----
EOF
```

2. Stellen Sie fest, ob Sie dem OpenPGP Schlüssel vertrauen möchten. Bei der Entscheidung, ob dem oben genannten Schlüssel vertraut werden soll, sollten Sie unter anderem die folgenden Faktoren berücksichtigen:

   - Die Internetverbindung, mit der Sie den GPG-Schlüssel von dieser Website abgerufen haben, ist sicher.
   - Das Gerät, mit dem Sie auf diese Website zugreifen, ist sicher.
   - AWS hat Maßnahmen ergriffen, um das Hosting des OpenPGP öffentlichen Schlüssels auf dieser Website zu sichern.

3. Wenn Sie sich dafür entscheiden, dem OpenPGP Schlüssel zu vertrauen, bearbeiten Sie den Schlüssel so, dass er vertrauenswürdig ist. gpg Gehen Sie dabei wie im folgenden Beispiel vor:

```
$ gpg --edit-key 0xB840C08C29A90796A071FAA5F6CD3CE6B76F3CEF

    gpg (GnuPG) 2.0.22; Copyright (C) 2013 Free Software Foundation, Inc.
    This is free software: you are free to change and redistribute it.
    There is NO WARRANTY, to the extent permitted by law.
```

```
   pub  4096R/4BF0B8D2  created: 2023-06-23  expires: 2025-06-22  usage: SCEA
                        trust: unknown       validity: unknown
   [ unknown] (1). AWS Deadline Cloud example@example.com

   gpg> trust
   pub  4096R/4BF0B8D2  created: 2023-06-23  expires: 2025-06-22  usage: SCEA
                        trust: unknown       validity: unknown
   [ unknown] (1). AWS Deadline Cloud aws-deadline@amazon.com

   Please decide how far you trust this user to correctly verify other users'
keys
   (by looking at passports, checking fingerprints from different sources,
etc.)

     1 = I don't know or won't say
     2 = I do NOT trust
     3 = I trust marginally
     4 = I trust fully
     5 = I trust ultimately
     m = back to the main menu

   Your decision? 5
   Do you really want to set this key to ultimate trust? (y/N) y

   pub  4096R/4BF0B8D2  created: 2023-06-23  expires: 2025-06-22  usage: SCEA
                        trust: ultimate      validity: unknown
   [ unknown] (1). AWS Deadline Cloud aws-deadline@amazon.com
   Please note that the shown key validity is not necessarily correct
   unless you restart the program.

   gpg> quit
```

4. Überprüfen Sie das Installationsprogramm für Deadline Cloud Submitter

   Gehen Sie wie folgt vor, um das Installationsprogramm für den Deadline Cloud-Absender zu verifizieren:

   a. Laden Sie die Signaturdatei für das Deadline Cloud-Installationsprogramm für Submitter herunter.

      Laden Sie die Signaturdatei (.sig) herunter

b. Überprüfen Sie die Signatur des Deadline Cloud Submitter-Installationsprogramms, indem Sie Folgendes ausführen:

```
gpg --verify ./DeadlineCloudSubmitter-linux-x64-installer.run.sig ./
DeadlineCloudSubmitter-linux-x64-installer.run
```

5. Überprüfen Sie den Deadline Cloud-Monitor

> **Note**
>
> Sie können den Download des Deadline Cloud-Monitors mithilfe von Signaturdateien oder plattformspezifischen Methoden überprüfen. Plattformspezifische Methoden finden Sie Linux (Debian) auf der Registerkarte, auf der Registerkarte Linux (RPM) oder auf der Linux (AppImage) Registerkarte, die auf Ihrem heruntergeladenen Dateityp basiert.

Gehen Sie wie folgt vor, um die Desktop-Anwendung Deadline Cloud Monitor anhand von Signaturdateien zu verifizieren:

a. Laden Sie die entsprechende Signaturdatei für Ihr Deadline Cloud Monitor-Installationsprogramm herunter:

- Laden Sie die .deb-Signaturdatei herunter
- Laden Sie die .rpm-Signaturdatei herunter
- Herunterladen. AppImage Signaturdatei

b. Überprüfen Sie die Signatur:

Für .deb:

```
gpg --verify ./deadline-cloud-monitor_amd64.deb.sig ./deadline-cloud-
monitor_amd64.deb
```

Für .rpm:

```
gpg --verify ./deadline-cloud-monitor.x86_64.rpm.sig ./deadline-cloud-
monitor.x86_64.rpm
```

Für. AppImage:

```
gpg --verify ./deadline-cloud-monitor_amd64.AppImage.sig ./deadline-cloud-
monitor_amd64.AppImage
```

c. Vergewissern Sie sich, dass die Ausgabe wie folgt aussieht:

```
gpg: Signature made Mon Apr 1 21:10:14 2024 UTC

gpg: using RSA key B840C08C29A90796A071FAA5F6CD3CE6B7
```

Wenn die Ausgabe den Ausdruck enthält, bedeutet dies`Good signature from "AWS Deadline Cloud"`, dass die Signatur erfolgreich verifiziert wurde und Sie das Installationsskript für den Deadline Cloud-Monitor ausführen können.

Historische Schlüssel

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGX6GQsBEADduUtJgqSXI+q76O6fsFwEYKmbnlyL0xKvlq32EZuyv0otZo5L
le4m5Gg52AzrvPvDiUTLooAlvYeozaYyirIGsK08Ydz0Ftdjroiuh/mw9JSJDJRI
rnRn5yKet1JFezkjopA3pjsTBP6lW/mb1bDBDEwwwtH0x9lV7A03FJ9T7Uzu/qSh
qO/UYdkafro3cPASvkqgDt2tCvURfBcUCAjZVFcLZcVD5iwXacxvKsxxS/e7kuVV
I1+VGT8Hj8XzWYhjCZxOLZk/fvpYPMyEEujN0fYUp6RtMIXve0C9awwMCy5nBG2J
eE2Ol5DsCpTaBd4Fdr3LWcSs8JFA/YfP9auL3NczOozPoVJt+fw8CBlVIXO0J7l5
hvHDjcC+5v0wxqAlMG6+f/SX7CT8FXK+L3iOJ5gBYUNXqHSxUdv8kt76/KVmQa1B
Akl+MPKpMq+lhw++S3G/lXqwWaDNQbRRw7dSZHymQVXvPp1nsqc3hV7KlOM+6s6g
1g4mvFY4lf6DhptwZLWyQXU8rBQpojvQfiSmDFrFPWFi5BexesuVnkGIolQoklKx
AVUSdJPVEJCteyy7td4FPhBaSqT5vW3+ANbr9b/uoRYWJvn17dN0cc9HuRh/Ai+I
nkfECo2WUDLZ0fEKGjGyFX+todWvJXjvc5kmE9Ty5vJp+M9Vvb8jd6t+mwARAQAB
tCxBV1MgRGVhZGxpbmUgQ2xvdWQgPGF3cy1kZWFkbGluZUBhbWF6b24uY29tPokC
VwQTAQgAQRYhBLhAwIwpqQeWoHH6pfbNPOa3bzzvBQJl+hkLAxsvBAUJA8JnAAUL
CQgHAgIiAgYVCgkICwIDFgIBAh4HAheAAAoJEPbNPOa3bzzvKswQAJXzKSAY8sY8
F6Eas2oYwIDDdDurs8FiEnFghjUEO6MTt9AykF/jw+CQg2UzFtEyObHBymhgmhXE
3buVeom96tgM3ZDfZu+sxi5pGX6oAQnZ6riztN+VpkpQmLgwtMGpSMLl3KLwnv2k
WK8mrR/fPMkfdaewB7A6RIUYiW33GAL4KfMIs8/vIwIJw99NxHpZQVoU6dFpuDtE
1OuxGcCqGJ7mAmo6H/YawSNp2Ns80gyqIKYo7o3LJ+WRroIRlQyctq8gnR9JvYXX
42ASqLq5+OXKo4qh81blXKYqtc176BbbSNFjWnzIQgKDgNiHFZCdcOVgqDhwO15r
NICbqqwwNLj/Fr2kecYx180Ktpl0jOOw5IOyh3bf3MVGWnYRdjvA1v+/CO+55N4g
z0kf50Lcdu5RtqV10XBCifn28pecqPaSdYcssYSRl5DLiFktGbNzTGcZZwITTKQc
```

```
af8PPdTGtnnb6P+cdbW3bt9MVtN5/dgSHLThnS8MPEuNCtkTnpXshuVuBGgwBMdb
qUC+HjqvhZzbwns8dr5WI+6HWNBFgGANn6ageYl58vVp0UkuNP8wcWjRARciHXZx
ku6W2jPTHDWGNrBQO2Fx7fd2QYJheIPPAShHcfJO+xgWCof45D0vAxAJ8gGg9Eq+
gFWhsx4NSHn2gh1gDZ41Ou/4exJ1lwPM
=uVaX
-----END PGP PUBLIC KEY BLOCK-----
EOF
```

Linux (AppImage)

Um Pakete zu überprüfen, die a verwendenLinux. AppImage Binär, führen Sie zuerst die Schritte 1 bis 3 Linux auf der Registerkarte aus und führen Sie dann die folgenden Schritte aus.

1. Laden Sie GitHub validate-x86_64 von der AppImageUpdate Seite in herunter. AppImageDatei.
2. Führen Sie nach dem Herunterladen der Datei den folgenden Befehl aus, um Ausführungsberechtigungen hinzuzufügen.

   ```
   chmod a+x ./validate-x86_64.AppImage
   ```

3. Führen Sie den folgenden Befehl aus, um Ausführungsberechtigungen hinzuzufügen.

   ```
   chmod a+x ./deadline-cloud-monitor_<APP_VERSION>_amd64.AppImage
   ```

4. Führen Sie den folgenden Befehl aus, um die Signatur des Deadline Cloud-Monitors zu überprüfen.

   ```
   ./validate-x86_64.AppImage ./deadline-cloud-monitor_<APP_VERSION>_amd64.AppImage
   ```

   Wenn die Ausgabe den Ausdruck enthält`Validation successful`, bedeutet dies, dass die Signatur erfolgreich verifiziert wurde und Sie das Installationsskript für den Deadline Cloud-Monitor problemlos ausführen können.

Linux (Debian)

Um Pakete zu verifizieren, die Linux eine.deb-Binärdatei verwenden, führen Sie zunächst die Schritte 1—3 Linux auf der Registerkarte aus.

dpkg ist das zentrale Paketverwaltungswerkzeug in den meisten debian basierten Linux Distributionen. Sie können die .deb-Datei mit dem Tool überprüfen.

1. Laden Sie die .deb-Datei für den Deadline Cloud Monitor herunter:

   Laden Sie den Deadline Cloud Monitor (.deb) herunter

2. Überprüfen Sie die .deb-Datei:

   ```
   dpkg-sig --verify deadline-cloud-monitor_amd64.deb
   ```

3. Die Ausgabe wird wie folgt aussehen:

   ```
   Processing deadline-cloud-monitor_amd64.deb...
   GOODSIG _gpgbuilder B840C08C29A90796A071FAA5F6CD3C 171200
   ```

4. Um die .deb-Datei zu überprüfen, stellen Sie sicher, dass sie in der Ausgabe vorhanden `GOODSIG` ist.

Linux (RPM)

Um Pakete zu verifizieren, die eine Linux .rpm-Binärdatei verwenden, führen Sie zunächst die Schritte 1 bis 3 auf der Linux Registerkarte aus.

1. Laden Sie die .rpm-Datei für den Deadline Cloud Monitor herunter:

   Laden Sie den Deadline Cloud Monitor (.rpm) herunter

2. Überprüfen Sie die .rpm-Datei:

   ```
   gpg --export --armor "Deadline Cloud" > key.pub
   sudo rpm --import key.pub
   rpm -K deadline-cloud-monitor.x86_64.rpm
   ```

3. Die Ausgabe wird wie folgt aussehen:

   ```
   deadline-cloud-monitor.x86_64.rpm: digests signatures OK
   ```

4. Um die .rpm-Datei zu überprüfen, vergewissern Sie sich, dass sie in der Ausgabe enthalten `digests signatures OK` ist.

Dokumentverlauf

Informationen zu Updates für AWS Deadline Cloud finden Sie in den Versionshinweisen zu Deadline Cloud.

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.