Construire des architectures hexagonales sur AWS

AWS Directives prescriptives

AWS Directives prescriptives: Construire des architectures hexagonales sur AWS

Table of Contents

Construire des architectures hexagonales sur AWS

Furkan Oruc, Dominik Goby, Darius Kunce et Michal Ploski, Amazon Web Services ()AWS

Juin 2022 (historique du document)

Ce guide décrit un modèle mental et un ensemble de modèles pour le développement d'architectures logicielles. Ces architectures sont faciles à maintenir, à étendre et à faire évoluer au sein de l'entreprise à mesure que l'adoption des produits augmente. Les hyperscalers cloud tels qu'Amazon Web Services (AWS) fournissent des éléments de base permettant aux petites et grandes entreprises d'innover et de créer de nouveaux produits logiciels. Le rythme rapide de l'introduction de ces nouveaux services et fonctionnalités amène les parties prenantes de l'entreprise à s'attendre à ce que leurs équipes de développement prototypent de nouveaux produits minimalement viables (MVPs) plus rapidement, afin que les nouvelles idées puissent être testées et vérifiées dès que possible. Souvent, ceux-ci MVPs sont adoptés et font partie de l'écosystème logiciel d'entreprise. Lors de leur production MVPs, les équipes abandonnent parfois les règles de développement logiciel et les meilleures pratiques, telles que les principes SOLID et les tests unitaires. Ils supposent que cette approche accélérera le développement et réduira les délais de commercialisation. Toutefois, s'ils ne parviennent pas à créer un modèle de base et un cadre pour l'architecture logicielle à tous les niveaux, il sera difficile, voire impossible, de développer de nouvelles fonctionnalités pour le produit. Le manque de certitude et l'évolution des exigences peuvent également ralentir l'équipe pendant le processus de développement.

Ce guide décrit une architecture logicielle proposée, allant d'une architecture hexagonale de bas niveau à une décomposition architecturale et organisationnelle de haut niveau, qui utilise la conception axée sur le domaine (DDD) pour relever ces défis. DDD aide à gérer la complexité de l'entreprise et à faire évoluer l'équipe d'ingénierie à mesure que de nouvelles fonctionnalités sont développées. Il aligne les parties prenantes commerciales et techniques sur les problèmes commerciaux, appelés domaines, en utilisant un langage omniprésent. L'architecture hexagonale est un outil technique de cette approche dans un domaine très spécifique, appelé contexte limité. Un contexte limité est un sous-domaine très cohérent et faiblement couplé du problème commercial. Nous vous recommandons d'adopter une architecture hexagonale pour tous vos projets logiciels d'entreprise, quelle que soit leur complexité.

L'architecture hexagonale encourage l'équipe d'ingénierie à résoudre d'abord le problème commercial, tandis que l'architecture classique en couches détourne l'attention de l'ingénierie du domaine pour se concentrer d'abord sur la résolution des problèmes techniques. En outre, si le

logiciel suit une architecture hexagonale, il est plus facile d'adopter une approche de développement axée sur les tests, ce qui réduit la boucle de rétroaction dont les développeurs ont besoin pour tester les exigences commerciales. Enfin, l'utilisation de commandes et de gestionnaires de commandes est un moyen d'appliquer les principes de responsabilité unique et d'ouverture-fermeture de SOLID. Le respect de ces principes produit une base de code dans laquelle les développeurs et les architectes travaillant sur le projet peuvent facilement naviguer et comprendre, et réduit le risque d'introduire des modifications majeures dans les fonctionnalités existantes.

Ce guide s'adresse aux architectes logiciels et aux développeurs qui souhaitent comprendre les avantages de l'adoption de l'architecture hexagonale et du DDD pour leurs projets de développement logiciel. Il inclut un exemple de conception d'une infrastructure pour votre application AWS prenant en charge une architecture hexagonale. Pour un exemple de mise en œuvre, consultez Structurer un projet Python en architecture hexagonale à l'aide du AWS Lambda site Web AWS Prescriptive Guidance.

Présentation

Conception axée sur le domaine (DDD)

Dans la conception axée sur le domaine (DDD), un domaine est au cœur du système logiciel. Le modèle de domaine est défini en premier, avant de développer un autre module, et il ne dépend pas des autres modules de bas niveau. Au lieu de cela, les modules tels que les bases de données, la couche de présentation et les modules externes dépendent APIs tous du domaine.

Dans DDD, les architectes décomposent la solution en contextes délimités en utilisant la décomposition basée sur la logique métier plutôt que la décomposition technique. Les avantages de cette approche sont présentés dans la Résultats commerciaux ciblés section.

Le DDD est plus facile à mettre en œuvre lorsque les équipes utilisent une architecture hexagonale. Dans l'architecture hexagonale, le cœur de l'application est le centre de l'application. Il est découplé des autres modules via des ports et des adaptateurs, et ne dépend pas d'autres modules. Cela correspond parfaitement à DDD, où un domaine est le cœur de l'application qui résout un problème commercial. Ce guide propose une approche dans laquelle vous modélisez le cœur de l'architecture hexagonale en tant que modèle de domaine d'un contexte délimité. La section suivante décrit plus en détail l'architecture hexagonale.

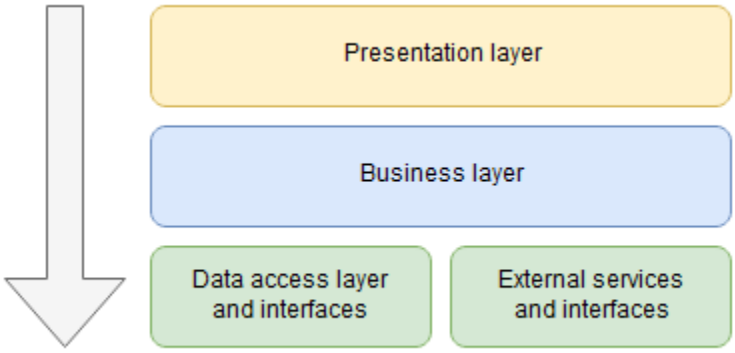
Ce guide ne couvre pas tous les aspects du DDD, qui est un sujet très vaste. Pour mieux comprendre, vous pouvez consulter les ressources DDD répertoriées sur le site Web de Domain Language.

Architecture hexagonale

L'architecture hexagonale, également connue sous le nom de ports et adaptateurs ou architecture onion, est un principe de gestion de l'inversion des dépendances dans les projets logiciels. L'architecture hexagonale met fortement l'accent sur la logique métier du domaine principal lors du développement de logiciels et traite les points d'intégration externes comme secondaires. L'architecture hexagonale aide les ingénieurs logiciels à adopter de bonnes pratiques telles que le développement piloté par les tests (TDD), qui, à son tour, favorise l'évolution de l'architecture et vous aide à gérer des domaines complexes sur le long terme.

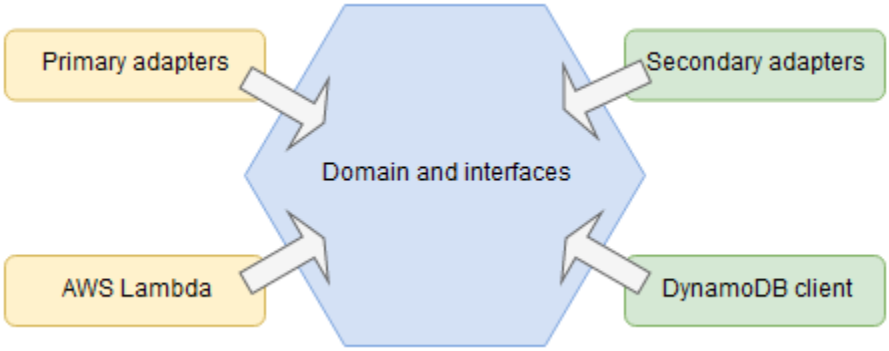
Comparons l'architecture hexagonale à l'architecture en couches classique, qui est le choix le plus populaire pour la modélisation de projets logiciels structurés. Il existe des différences subtiles mais puissantes entre les deux approches.

Dans l'architecture en couches, les projets logiciels sont structurés en niveaux, ce qui représente des préoccupations générales telles que la logique métier ou la logique de présentation. Cette architecture utilise une hiérarchie de dépendances, dans laquelle les couches supérieures dépendent des couches inférieures, mais pas l'inverse. Dans le schéma suivant, la couche de présentation est responsable des interactions avec les utilisateurs. Elle inclut donc l'interface utilisateur APIs, les interfaces de ligne de commande et les composants similaires. La couche de présentation dépend de la couche métier, qui implémente la logique du domaine. La couche métier, quant à elle, dépend de la couche d'accès aux données et de multiples services externes.



Le principal inconvénient de cette configuration est la structure de dépendance. Par exemple, si le modèle de stockage des données dans la base de données change, cela affecte l'interface d'accès aux données. Toute modification du modèle de données affecte également la couche métier, qui dépend de l'interface d'accès aux données. Par conséquent, les ingénieurs logiciels ne peuvent apporter aucune modification à l'infrastructure sans affecter la logique du domaine. Cela augmente à son tour le risque de bogues de régression.

L'architecture hexagonale définit les relations de dépendance d'une manière différente, comme illustré dans le schéma suivant. Il concentre la prise de décision sur la logique métier du domaine, qui définit toutes les interfaces. Les composants externes interagissent avec la logique métier par le biais d'interfaces appelées ports. Les ports sont des abstractions qui définissent les interactions du domaine avec le monde extérieur. Chaque composant de l'infrastructure doit implémenter ces ports, de sorte que les modifications apportées à ces composants n'affectent plus la logique du domaine principal.



Les composants environnants sont appelés adaptateurs. Un adaptateur est un proxy entre le monde externe et le monde interne, et implémente un port défini dans le domaine. Les adaptateurs peuvent être classés en deux groupes : principaux et secondaires. Les adaptateurs principaux sont les points d'entrée du composant logiciel. Ils permettent aux acteurs externes, aux utilisateurs et aux services d'interagir avec la logique de base. AWS Lambda est un bon exemple d'adaptateur principal. Il s'intègre à plusieurs AWS services qui peuvent invoquer les fonctions Lambda comme points d'entrée. Les adaptateurs secondaires sont des wrappers de bibliothèques de services externes qui gèrent les communications avec le monde extérieur. Un bon exemple d'adaptateur secondaire est un client Amazon DynamoDB pour l'accès aux données.

Résultats commerciaux ciblés

L'architecture hexagonale décrite dans ce guide vous aide à atteindre les objectifs suivants :

- Réduisez les délais de mise sur le marché en améliorant le cycle de développement
- Améliorez la qualité des logiciels
- Adaptez-vous plus facilement au changement

Ces processus sont décrits en détail dans les sections suivantes.

Améliorer le cycle de développement

Le développement de logiciels pour le cloud présente de nouveaux défis pour les ingénieurs logiciels, car il est très difficile de reproduire l'environnement d'exécution localement sur la machine de développement. Un moyen simple de valider le logiciel consiste à le déployer dans le cloud et à le tester là-bas. Cependant, cette approche implique un long cycle de feedback, en particulier lorsque l'architecture logicielle contient plusieurs déploiements sans serveur. L'amélioration de ce cycle de feedback réduit le temps de développement des fonctionnalités, ce qui réduit considérablement le délai de mise sur le marché.

Tests dans le cloud

Tester directement dans le cloud est le seul moyen de vous assurer que vos composants architecturaux, tels que les passerelles dans Amazon API Gateway, les AWS Lambda fonctions, les tables Amazon DynamoDB et les autorisations (IAM) Gestion des identités et des accès AWS , sont correctement configurés. C'est peut-être également le seul moyen fiable de tester les intégrations de composants. Bien que certains AWS services (tels que DynamoDB) puissent être déployés localement, la plupart d'entre eux ne peuvent pas être répliqués dans une configuration locale. Dans le même temps, les outils tiers tels que Moto et LocalStackqui simulent AWS des services à des fins de test peuvent ne pas refléter correctement les contrats d'API de service réels, ou le nombre de fonctionnalités peut être limité.

Cependant, la partie la plus complexe du logiciel d'entreprise réside dans la logique métier, et non dans l'architecture cloud. L'architecture change moins souvent que le domaine, qui doit s'adapter aux nouvelles exigences commerciales. Ainsi, tester la logique métier dans le cloud devient un processus intense qui consiste à modifier le code, à lancer un déploiement, à attendre que l'environnement soit prêt et à valider le changement. Si un déploiement ne prend que 5 minutes, il faudra au moins une heure pour effectuer et tester 10 modifications dans la logique métier. Si la logique métier est plus complexe, les tests peuvent nécessiter des jours d'attente pour terminer les déploiements. Si vous avez plusieurs fonctionnalités et plusieurs ingénieurs dans votre équipe, la période prolongée devient rapidement perceptible pour l'entreprise.

Réalisation de tests locaux

Une architecture hexagonale permet aux développeurs de se concentrer sur le domaine plutôt que sur les aspects techniques de l'infrastructure. Cette approche utilise des tests locaux (les outils de

test unitaires du framework de développement que vous avez choisi) pour répondre aux exigences de logique du domaine. Vous n'aurez pas à passer du temps à résoudre des problèmes d'intégration technique ou à déployer votre logiciel dans le cloud pour tester la logique métier. Vous pouvez exécuter des tests unitaires localement et réduire la boucle de rétroaction de quelques minutes à quelques secondes. Si un déploiement prend 5 minutes mais que les tests unitaires sont terminés en 5 secondes, cela représente une réduction significative du temps nécessaire pour détecter les erreurs. La Tester d'abord la logique métier section suivante de ce guide couvre cette approche plus en détail.

Parallélisation du développement

L'approche de l'architecture hexagonale permet aux équipes de développement de paralléliser les efforts de développement. Les développeurs peuvent concevoir et implémenter les différents composants du service individuellement. Cette parallélisation est possible grâce à l'isolation de chaque composant et aux interfaces définies entre chaque composant.

Délai de mise sur le marché du produit

Les tests unitaires locaux améliorent le cycle de feedback sur le développement et réduisent le délai de commercialisation des nouveaux produits ou fonctionnalités, en particulier lorsque ceux-ci contiennent une logique métier complexe, comme expliqué précédemment. En outre, l'augmentation de la couverture du code par les tests unitaires réduit considérablement le risque d'introduction de bogues de régression lorsque vous mettez à jour ou refactorisez la base de code. La couverture des tests unitaires vous permet également de refactoriser en permanence la base de code pour qu'elle reste bien organisée, ce qui accélère le processus d'intégration des nouveaux ingénieurs. Cette question est abordée plus en détail dans la La qualité dès le design section. Enfin, si la logique métier est bien isolée et testée, elle permet aux développeurs de s'adapter rapidement à l'évolution des exigences fonctionnelles et non fonctionnelles. Ceci est expliqué plus en détail dans la S'adapter au changement section.

La qualité dès le design

L'adoption d'une architecture hexagonale permet de promouvoir la qualité de votre base de code dès le début de votre projet. Il est important de créer un processus qui vous aide à répondre aux exigences de qualité attendues dès le début, sans ralentir le processus de développement.

Modifications localisées et lisibilité accrue

L'approche de l'architecture hexagonale permet aux développeurs de modifier le code d'une classe ou d'un composant sans affecter les autres classes ou composants. Cette conception favorise la cohésion des composants développés. En découplant le domaine des adaptateurs et en utilisant des interfaces connues, vous pouvez améliorer la lisibilité du code. Il devient plus facile d'identifier les problèmes et de corriger les cas.

Cette approche facilite également la révision du code pendant le développement et limite l'introduction de modifications non détectées ou de dettes techniques.

Tester d'abord la logique métier

Les tests locaux peuvent être réalisés en introduisant end-to-end, en intégrant et en réalisant des tests unitaires dans le projet. End-to-endles tests couvrent l'ensemble du cycle de vie des demandes entrantes. Ils invoquent généralement un point d'entrée d'application et le testent pour voir s'il répond aux exigences de l'entreprise. Chaque projet logiciel doit comporter au moins un scénario de test utilisant des entrées connues et produisant les résultats attendus. Cependant, l'ajout de scénarios spécifiques peut s'avérer complexe, car chaque test doit être configuré pour envoyer une demande via un point d'entrée (par exemple, via une API REST ou des files d'attente), passer par tous les points d'intégration requis par l'action commerciale, puis affirmer le résultat. La configuration de l'environnement pour le scénario de test et l'affirmation des résultats peuvent prendre beaucoup de temps aux développeurs.

Dans une architecture hexagonale, vous testez la logique métier de manière isolée et vous utilisez des tests d'intégration pour tester les adaptateurs secondaires. Vous pouvez utiliser des adaptateurs fictifs ou factices dans vos tests de logique métier. Vous pouvez également combiner les tests pour les cas d'utilisation professionnels avec les tests unitaires pour votre modèle de domaine afin de maintenir une couverture élevée avec un faible couplage. En tant que bonne pratique, les tests d'intégration ne doivent pas valider la logique métier. Ils doivent plutôt vérifier que l'adaptateur secondaire appelle correctement les services externes.

Idéalement, vous pouvez utiliser le développement piloté par les tests (TDD) et commencer à définir des entités de domaine ou des cas d'utilisation métier avec des tests appropriés dès le début du développement. L'écriture des tests vous permet d'abord de créer des implémentations fictives des interfaces requises par le domaine. Lorsque les tests sont réussis et que les règles logiques du domaine sont satisfaites, vous pouvez implémenter les adaptateurs réels et déployer le logiciel dans l'environnement de test. À ce stade, votre implémentation de la logique de domaine n'est peut-être pas idéale. Vous pouvez ensuite refactoriser l'architecture existante pour la faire évoluer en introduisant des modèles de conception ou en réorganisant le code en général. En utilisant cette approche, vous pouvez éviter d'introduire des bogues de régression et vous pouvez améliorer l'architecture au fur et à mesure que le projet grandit. En combinant cette approche avec les tests automatiques que vous exécutez dans le cadre de votre processus d'intégration continue, vous pouvez réduire le nombre de bogues potentiels avant qu'ils ne soient mis en production.

Si vous utilisez des déploiements sans serveur, vous pouvez rapidement configurer une instance de l'application dans votre AWS compte pour une intégration et end-to-end des tests manuels. Après ces étapes de mise en œuvre, nous vous recommandons d'automatiser les tests à chaque nouvelle modification apportée au référentiel.

Maintenabilité

La maintenabilité fait référence à l'exploitation et à la surveillance d'une application afin de s'assurer qu'elle répond à toutes les exigences et de minimiser la probabilité d'une défaillance du système. Pour que le système soit opérationnel, vous devez l'adapter au trafic futur ou aux exigences opérationnelles. Vous devez également vous assurer qu'il est disponible et facile à déployer avec un impact minimal ou nul sur les clients.

Pour comprendre l'état actuel et historique de votre système, vous devez le rendre observable. Pour ce faire, vous pouvez fournir des métriques, des journaux et des traces spécifiques que les opérateurs peuvent utiliser pour garantir que le système fonctionne comme prévu et pour suivre les bogues. Ces mécanismes devraient également permettre aux opérateurs d'effectuer une analyse des causes profondes sans avoir à se connecter à la machine et à lire le code.

Une architecture hexagonale vise à améliorer la maintenabilité de vos applications Web afin que votre code nécessite moins de travail dans l'ensemble. En séparant les modules, en localisant les modifications et en découplant la logique métier des applications de la mise en œuvre des adaptateurs, vous pouvez produire des métriques et des journaux qui aident les opérateurs à mieux comprendre le système et à comprendre l'étendue des modifications spécifiques apportées aux adaptateurs principaux ou secondaires.

S'adapter au changement

Les systèmes logiciels ont tendance à se compliquer. Cela peut s'expliquer notamment par les modifications fréquentes des exigences de l'entreprise et le peu de temps consacré à l'adaptation de l'architecture logicielle en conséquence. Une autre raison pourrait être l'investissement insuffisant dans la mise en place de l'architecture logicielle au début du projet afin de s'adapter aux changements fréquents. Quelle qu'en soit la raison, un système logiciel peut devenir compliqué au point qu'il est presque impossible d'apporter une modification. Il est donc important de créer une architecture logicielle maintenable dès le début du projet. Une bonne architecture logicielle permet de s'adapter facilement aux changements.

Cette section explique comment concevoir des applications maintenables à l'aide d'une architecture hexagonale qui s'adapte facilement aux exigences non fonctionnelles ou commerciales.

S'adapter aux nouvelles exigences non fonctionnelles en utilisant des ports et des adaptateurs

En tant que cœur de l'application, le modèle de domaine définit les actions requises de la part du monde extérieur pour répondre aux exigences de l'entreprise. Ces actions sont définies par le biais d'abstractions, appelées ports. Ces ports sont implémentés par des adaptateurs distincts. Chaque adaptateur est responsable d'une interaction avec un autre système. Par exemple, vous pouvez avoir un adaptateur pour le référentiel de base de données et un autre pour interagir avec une API tierce. Le domaine n'est pas au courant de l'implémentation de l'adaptateur, il est donc facile de remplacer un adaptateur par un autre. Par exemple, l'application peut passer d'une base de données SQL à une base de données NoSQL. Dans ce cas, un nouvel adaptateur doit être développé pour implémenter les ports définis par le modèle de domaine. Le domaine ne dépend pas du référentiel de base de données et utilise des abstractions pour interagir. Il ne serait donc pas nécessaire de modifier quoi que ce soit dans le modèle de domaine. Par conséquent, l'architecture hexagonale s'adapte facilement aux exigences non fonctionnelles.

S'adapter aux nouvelles exigences de l'entreprise en utilisant des commandes et des gestionnaires de commandes

Dans l'architecture en couches classique, le domaine dépend de la couche de persistance. Si vous souhaitez modifier le domaine, vous devez également modifier la couche de persistance. En

comparaison, dans une architecture hexagonale, le domaine ne dépend pas des autres modules du logiciel. Le domaine est au cœur de l'application, et tous les autres modules (ports et adaptateurs) dépendent du modèle de domaine. Le domaine utilise le principe d'inversion de dépendance pour communiquer avec le monde extérieur par le biais de ports. L'avantage de l'inversion de dépendance est que vous pouvez modifier librement le modèle de domaine sans avoir peur de casser d'autres parties du code. Comme le modèle de domaine reflète le problème commercial que vous essayez de résoudre, la mise à jour du modèle de domaine pour l'adapter à l'évolution des exigences commerciales ne pose aucun problème.

Lorsque vous développez un logiciel, la séparation des préoccupations est un principe important à suivre. Pour réaliser cette séparation, vous pouvez utiliser un modèle de commande légèrement modifié. Il s'agit d'un modèle de conception comportementale dans lequel toutes les informations requises pour effectuer une opération sont encapsulées dans un objet de commande. Ces opérations sont ensuite traitées par des gestionnaires de commandes. Les gestionnaires de commandes sont des méthodes qui reçoivent une commande, modifient l'état du domaine, puis renvoient une réponse à l'appelant. Vous pouvez utiliser différents clients, tels que des files d'attente synchrones APIs ou asynchrones, pour exécuter des commandes. Nous vous recommandons d'utiliser des commandes et des gestionnaires de commandes pour chaque opération sur le domaine. En suivant cette approche, vous pouvez ajouter de nouvelles fonctionnalités en introduisant de nouvelles commandes et de nouveaux gestionnaires de commandes, sans modifier votre logique métier existante. Ainsi, l'utilisation d'un modèle de commande facilite l'adaptation aux nouvelles exigences de l'entreprise.

Découplage des composants en utilisant la façade de service ou le modèle CQRS

Dans l'architecture hexagonale, les adaptateurs principaux sont chargés de coupler librement les demandes de lecture et d'écriture entrantes des clients au domaine. Il existe deux manières de réaliser ce couplage souple : en utilisant un modèle de façade de service ou en utilisant le modèle de ségrégation des responsabilités des requêtes de commande (CQRS).

Le modèle de façade de service fournit une interface orientée vers l'avant pour servir les clients, par exemple la couche de présentation ou un microservice. Une façade de service fournit aux clients plusieurs opérations de lecture et d'écriture. Il est chargé de transférer les demandes entrantes vers le domaine et de mapper la réponse reçue du domaine aux clients. L'utilisation d'une façade de services est facile pour les microservices qui ont une seule responsabilité avec plusieurs opérations. Cependant, lorsque l'on utilise la façade du service, il est plus difficile de suivre les principes de responsabilité unique et de fermeture ouverte. Le principe de responsabilité unique stipule que

chaque module ne doit être responsable que d'une seule fonctionnalité du logiciel. Le principe ouvert/fermé stipule que le code doit être ouvert pour extension et fermé pour modification. Au fur et à mesure que la façade des services s'étend, toutes les opérations sont rassemblées dans une seule interface, davantage de dépendances y sont encapsulées et de plus en plus de développeurs commencent à modifier la même façade. Par conséquent, nous recommandons d'utiliser une façade de service uniquement s'il est clair que le service ne s'étendra pas beaucoup au cours du développement.

Une autre façon d'implémenter des adaptateurs principaux dans une architecture hexagonale consiste à utiliser le modèle CQRS, qui sépare les opérations de lecture et d'écriture à l'aide de requêtes et de commandes. Comme expliqué précédemment, les commandes sont des objets qui contiennent toutes les informations nécessaires pour modifier l'état du domaine. Les commandes sont exécutées par des méthodes de gestion de commandes. Les requêtes, en revanche, ne modifient pas l'état du système. Leur seul objectif est de renvoyer des données aux clients. Dans le modèle CQRS, les commandes et les requêtes sont implémentées dans des modules distincts. Cela est particulièrement avantageux pour les projets qui suivent une architecture axée sur les événements, car une commande peut être implémentée sous la forme d'un événement traité de manière asynchrone, alors qu'une requête peut être exécutée de manière synchrone à l'aide d'une API. Une requête peut également utiliser une autre base de données optimisée pour elle. L'inconvénient du modèle CQRS est qu'il prend plus de temps à mettre en œuvre qu'une façade de service. Nous vous recommandons d'utiliser le modèle CQRS pour les projets que vous prévoyez d'étendre et de maintenir à long terme. Les commandes et les requêtes constituent un mécanisme efficace pour appliquer le principe de responsabilité unique et développer des logiciels faiblement couplés, en particulier dans le cadre de projets de grande envergure.

Le CQRS présente de grands avantages à long terme, mais nécessite un investissement initial. Pour cette raison, nous vous recommandons d'évaluer soigneusement votre projet avant de décider d'utiliser le modèle CQRS. Cependant, vous pouvez structurer votre application en utilisant des commandes et des gestionnaires de commandes dès le départ, sans séparer les read/write opérations. Cela vous aidera à refactoriser facilement votre projet pour le CQRS si vous décidez d'adopter cette approche ultérieurement.

Mise à l'échelle organisationnelle

Une combinaison d'architecture hexagonale, de conception axée sur le domaine et (en option) de CQRS permet à votre organisation de faire évoluer rapidement son produit. Selon la loi de Conway, les architectures logicielles ont tendance à évoluer pour refléter les structures de communication

d'une entreprise. Cette observation a toujours eu des connotations négatives, car les grandes entreprises structurent souvent leurs équipes en fonction d'une expertise technique telle que la base de données, le bus de service d'entreprise, etc. Le problème de cette approche est que le développement de produits et de fonctionnalités implique toujours des préoccupations transversales, telles que la sécurité et l'évolutivité, qui nécessitent une communication constante entre les équipes. La structuration des équipes en fonction des caractéristiques techniques crée des silos inutiles au sein de l'organisation, ce qui se traduit par de mauvaises communications, un manque d'appropriation et une perte de vue d'ensemble. Finalement, ces problèmes d'organisation se reflètent dans l'architecture logicielle.

La manœuvre de Conway inverse, quant à elle, définit la structure organisationnelle en fonction des domaines qui favorisent l'architecture logicielle. Par exemple, les équipes interfonctionnelles sont chargées d'un ensemble spécifique de contextes délimités, qui sont identifiés à l'aide du DDD et du storming d'événements. Ces contextes délimités peuvent refléter des caractéristiques très spécifiques du produit. Par exemple, l'équipe chargée du compte peut être responsable du contexte de paiement. Chaque nouvelle fonctionnalité est attribuée à une nouvelle équipe dont les responsabilités sont très cohérentes et peu couplées, afin qu'elle puisse se concentrer uniquement sur la fourniture de cette fonctionnalité et réduire les délais de mise sur le marché. Les équipes peuvent être dimensionnées en fonction de la complexité des fonctionnalités, de sorte que les fonctionnalités complexes peuvent être attribuées à un plus grand nombre d'ingénieurs.

Bonnes pratiques

Modéliser le domaine d'activité

Passez du domaine commercial à la conception du logiciel pour vous assurer que le logiciel que vous écrivez répond aux besoins de l'entreprise.

Utilisez des méthodologies de conception pilotée par le domaine (DDD) telles que le storming d'événements pour modéliser le domaine commercial. Event Storming propose un format d'atelier flexible. Au cours de l'atelier, des experts du domaine et des logiciels explorent en collaboration la complexité du domaine commercial. Les experts en logiciels utilisent les résultats de l'atelier pour démarrer le processus de conception et de développement des composants logiciels.

Rédiger et exécuter des tests depuis le début

Utilisez le développement piloté par les tests (TDD) pour vérifier l'exactitude du logiciel que vous développez. Le TDD fonctionne mieux au niveau des tests unitaires. Le développeur conçoit un composant logiciel en écrivant d'abord un test, qui invoque ce composant. Ce composant n'a aucune implémentation au début, donc le test échoue. À l'étape suivante, le développeur implémente les fonctionnalités du composant, en utilisant des appareils de test avec des objets fictifs pour simuler le comportement des dépendances externes, ou ports. Lorsque le test est réussi, le développeur peut continuer en implémentant de véritables adaptateurs. Cette approche améliore la qualité du logiciel et permet d'obtenir un code plus lisible, car les développeurs comprennent comment les utilisateurs utiliseraient les composants. L'architecture hexagonale prend en charge la méthodologie TDD en séparant le cœur de l'application. Les développeurs rédigent des tests unitaires axés sur le comportement de base du domaine. Ils n'ont pas besoin d'écrire des adaptateurs complexes pour exécuter leurs tests ; ils peuvent plutôt utiliser de simples objets et accessoires fictifs.

Utilisez le développement piloté par le comportement (BDD) pour garantir l' end-to-endacceptation au niveau des fonctionnalités. Dans BDD, les développeurs définissent des scénarios pour les fonctionnalités et les vérifient auprès des parties prenantes de l'entreprise. Les tests BDD utilisent autant de langage naturel que possible pour y parvenir. L'architecture hexagonale soutient la méthodologie BDD avec son concept d'adaptateurs principaux et secondaires. Les développeurs peuvent créer des adaptateurs principaux et secondaires qui peuvent être exécutés localement sans faire appel à des services externes. Ils configurent la suite de tests BDD pour utiliser l'adaptateur principal local pour exécuter l'application.

Exécutez automatiquement chaque test dans le pipeline d'intégration continue pour évaluer en permanence la qualité du système.

Définissez le comportement du domaine

Décomposez le domaine en entités, objets de valeur et agrégats (découvrez comment implémenter la conception axée sur le domaine) et définissez leur comportement. Implémentez le comportement du domaine afin que les tests écrits au début du projet aboutissent. Définissez des commandes qui invoquent le comportement des objets du domaine. Définissez les événements que les objets du domaine émettent une fois qu'ils ont terminé un comportement.

Définissez les interfaces que les adaptateurs peuvent utiliser pour interagir avec le domaine.

Automatisez les tests et le déploiement

Après une première preuve de concept, nous vous recommandons de consacrer du temps à la mise en œuvre DevOps des pratiques. Par exemple, les pipelines d'intégration continue et de livraison continue (CI/CD) ainsi que les environnements de test dynamiques vous aident à maintenir la qualité du code et à éviter les erreurs lors du déploiement.

- Exécutez vos tests unitaires dans votre processus CI et testez votre code avant qu'il ne soit fusionné.
- Créez un processus CD pour déployer votre application dans un dev/test environnement statique ou dans des environnements créés dynamiquement qui prennent en charge l'intégration et les end-to-end tests automatiques.
- Automatisez le processus de déploiement pour les environnements dédiés.

Faites évoluer votre produit en utilisant les microservices et le CQRS

Si votre produit est un succès, adaptez-le en décomposant votre projet logiciel en microservices. Utilisez la portabilité qu'offre l'architecture hexagonale pour améliorer les performances. Divisez les services de requêtes et les gestionnaires de commandes en systèmes synchrone et asynchrone distincts. APIs Envisagez d'adopter le modèle de ségrégation des responsabilités des requêtes de commande (CQRS) et une architecture axée sur les événements.

Si vous recevez de nombreuses demandes de nouvelles fonctionnalités, envisagez de faire évoluer votre organisation en fonction des modèles DDD. Structurez vos équipes de manière à ce qu'elles possèdent une ou plusieurs fonctionnalités sous forme de contextes délimités, comme indiqué précédemment dans la Mise à l'échelle organisationnelle section. Ces équipes peuvent ensuite mettre en œuvre une logique métier en utilisant une architecture hexagonale.

Concevez une structure de projet qui correspond aux concepts d'architecture hexagonale

L'infrastructure en tant que code (IaC) est une pratique largement adoptée dans le développement du cloud. Il vous permet de définir et de gérer les ressources de votre infrastructure (telles que les réseaux, les équilibreurs de charge, les machines virtuelles et les passerelles) sous forme de code source. Ainsi, vous pouvez suivre toutes les modifications apportées à votre architecture à l'aide d'un système de contrôle de version. En outre, vous pouvez facilement créer et déplacer l'infrastructure à des fins de test. Nous vous recommandons de conserver le code de votre application et votre code d'infrastructure dans le même référentiel lorsque vous développez vos applications cloud. Cette approche facilite la maintenance de l'infrastructure de votre application.

Nous vous recommandons de diviser votre application en trois dossiers ou projets correspondant aux concepts de l'architecture hexagonale : `entrypoints` (adaptateurs principaux), `domain` (domaine et interfaces) et `adapters` (adaptateurs secondaires).

La structure de projet suivante fournit un exemple de cette approche lors de la conception d'une API sur AWS. Le projet conserve le code d'application (`app`) et le code d'infrastructure (`infra`) dans le même référentiel, comme recommandé précédemment.

```
app/  # application code
|--- adapters/  # implementation of the ports defined in the domain
     |--- tests/  # adapter unit tests
|--- entrypoints/  # primary adapters, entry points
     |--- api/  # api entry point
          |--- model/  # api model
          |--- tests/  # end to end api tests
|--- domain/  # domain to implement business logic using hexagonal architecture
     |--- command_handlers/  # handlers used to run commands on the domain
     |--- commands/  # commands on the domain
     |--- events/  # events emitted by the domain
     |--- exceptions/  # exceptions defined on the domain
     |--- model/  # domain model
```

```
    |--- ports/  # abstractions used for external communication
    |--- tests/  # domain tests
infra/  # infrastructure code
```

Comme indiqué précédemment, le domaine est le cœur de l'application et ne dépend d'aucun autre module. Nous vous recommandons de structurer le `domain` dossier de manière à inclure les sous-dossiers suivants :

- `command handlers`contient les méthodes ou les classes qui exécutent des commandes sur le domaine.
- `commands`contient les objets de commande qui définissent les informations requises pour effectuer une opération sur le domaine.
- `events`contient les événements émis via le domaine puis routés vers d'autres microservices.
- `exceptions`contient les erreurs connues définies dans le domaine.
- `model`contient les entités de domaine, les objets de valeur et les services de domaine.
- `ports`contient les abstractions par le biais desquelles le domaine communique avec les bases de données ou d'autres composants externes. APIs
- `tests`contient les méthodes de test (telles que les tests de logique métier) exécutées sur le domaine.

Les adaptateurs principaux sont les points d'entrée de l'application, tels que représentés par le `entrypoints` dossier. Cet exemple utilise le `api` dossier comme adaptateur principal. Ce dossier contient une API `model` qui définit l'interface dont l'adaptateur principal a besoin pour communiquer avec les clients. Le `tests` dossier contient des end-to-end tests pour l'API. Il s'agit de tests superficiels qui valident que les composants de l'application sont intégrés et fonctionnent en harmonie.

Les adaptateurs secondaires, tels que représentés par le `adapters` dossier, implémentent les intégrations externes requises par les ports de domaine. Un référentiel de base de données est un excellent exemple d'adaptateur secondaire. Lorsque le système de base de données change, vous pouvez écrire un nouvel adaptateur en utilisant l'implémentation définie par le domaine. Il n'est pas nécessaire de modifier le domaine ou la logique métier. Le `tests` sous-dossier contient des tests d'intégration externes pour chaque adaptateur.

Exemples d'infrastructure sur AWS

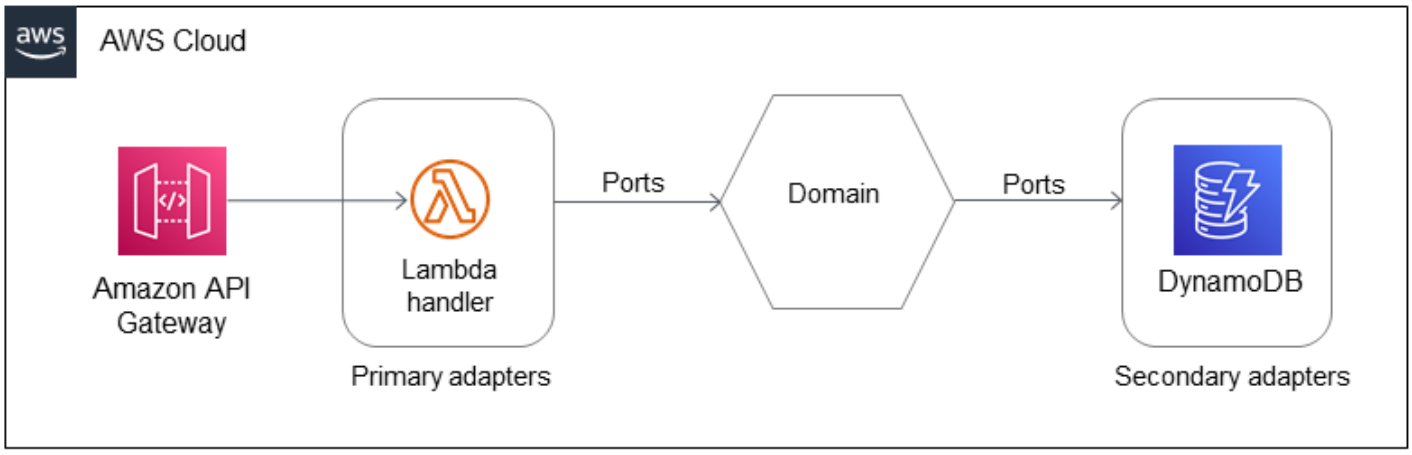
Cette section fournit des exemples de conception d'une infrastructure pour votre application AWS que vous pouvez utiliser pour implémenter une architecture hexagonale. Nous vous recommandons de commencer par une architecture simple pour créer un produit minimum viable (MVP). La plupart des microservices ont besoin d'un point d'entrée unique pour traiter les demandes des clients, d'une couche de calcul pour exécuter le code et d'une couche de persistance pour stocker les données. Les AWS services suivants sont d'excellents candidats pour une utilisation en tant que clients, adaptateurs principaux et adaptateurs secondaires dans une architecture hexagonale :

- Clientèle : Amazon API Gateway, Amazon Simple Queue Service (Amazon SQS), Elastic Load Balancing, Amazon EventBridge
- Adaptateurs principaux : AWS Lambda Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), Amazon Elastic Compute Cloud (Amazon EC2)
- Adaptateurs secondaires : Amazon DynamoDB, Amazon Relational Database Service (Amazon RDS), Amazon Aurora, API Gateway, Amazon SQS, EventBridge Elastic Load Balancing, Amazon Simple Notification Service (Amazon SNS)

Les sections suivantes traitent plus en détail de ces services dans le contexte de l'architecture hexagonale.

Commencez simplement

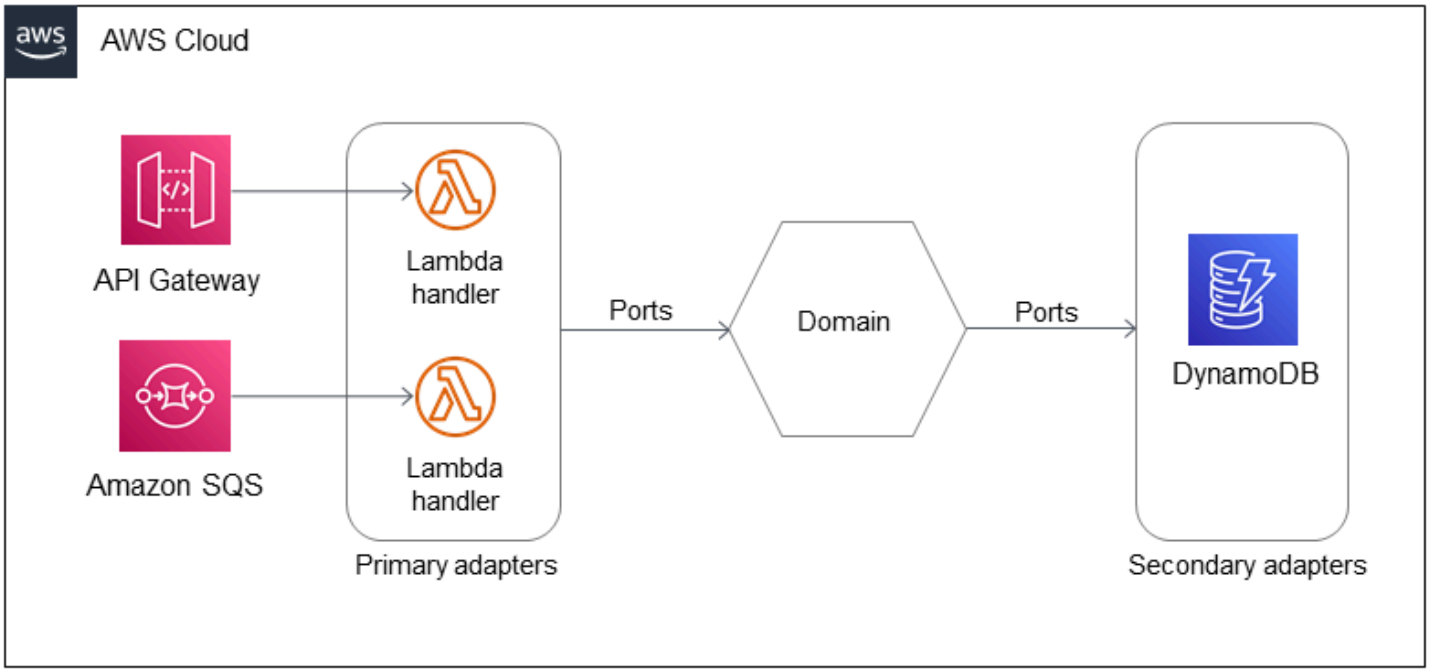
Nous vous recommandons de commencer simplement lorsque vous concevez une application à l'aide d'une architecture hexagonale. Dans cet exemple, API Gateway est utilisé comme client (API REST), Lambda est utilisé comme adaptateur principal (calcul) et DynamoDB est utilisé comme adaptateur secondaire (persistance). Le client de passerelle appelle le point d'entrée, qui, dans ce cas, est un gestionnaire Lambda.



Cette architecture est entièrement sans serveur et constitue un bon point de départ pour l'architecte. Nous vous recommandons d'utiliser le modèle de commande dans le domaine, car il facilite la maintenance du code et s'adapte aux nouvelles exigences commerciales et non fonctionnelles. Cette architecture peut être suffisante pour créer des microservices simples avec quelques opérations.

Appliquez le motif CQRS

Nous vous recommandons de passer au modèle CQRS si le nombre d'opérations sur le domaine doit augmenter. Vous pouvez appliquer le modèle CQRS en tant qu'architecture entièrement sans serveur en AWS utilisant l'exemple suivant.

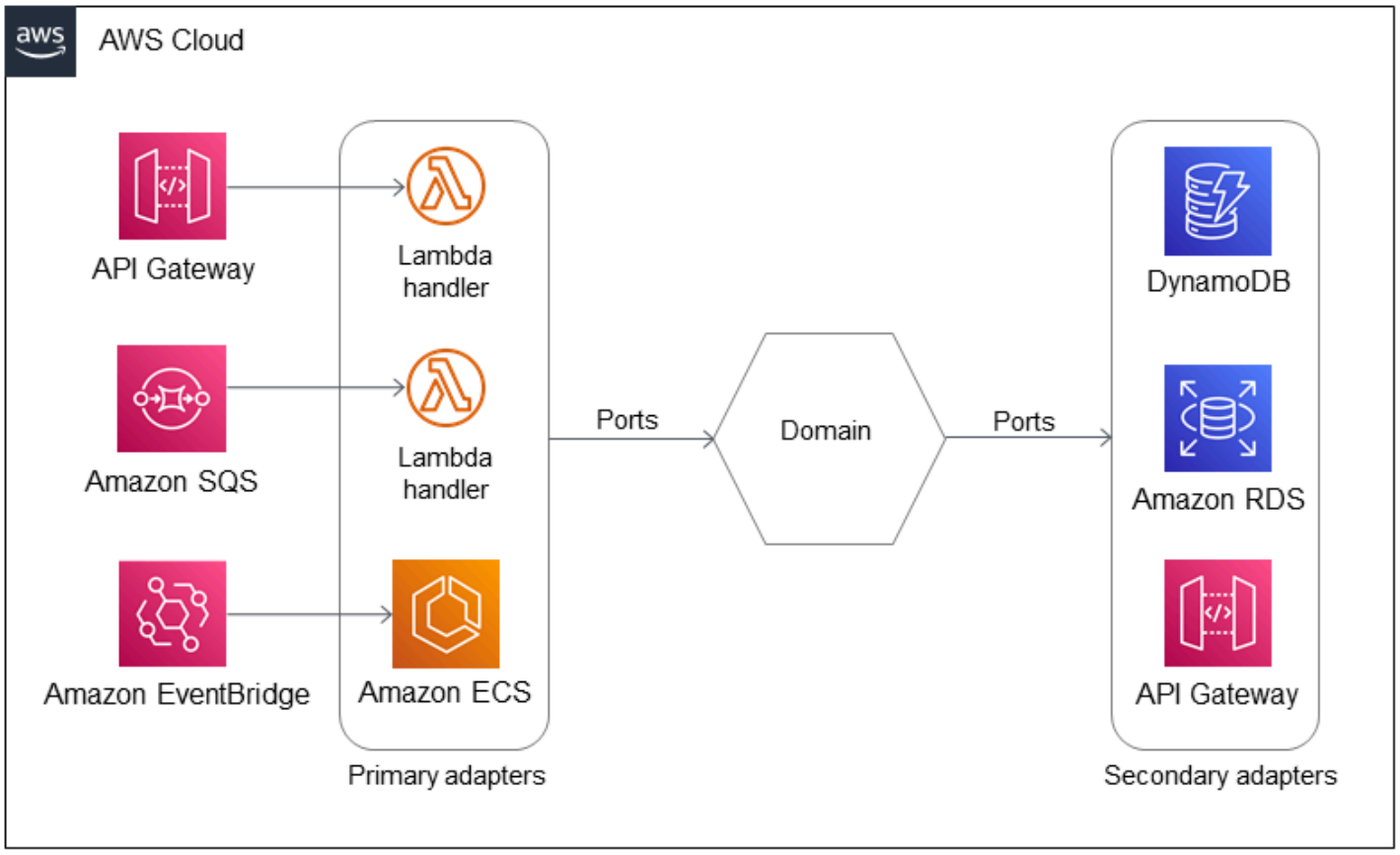


Cet exemple utilise deux gestionnaires Lambda, l'un pour les requêtes et l'autre pour les commandes. Les requêtes sont exécutées de manière synchrone en utilisant une passerelle d'API comme client. Les commandes sont exécutées de manière asynchrone en utilisant Amazon SQS comme client.

Cette architecture inclut plusieurs clients (API Gateway et Amazon SQS) et plusieurs adaptateurs principaux (Lambda), qui sont appelés par leurs points d'entrée correspondants (gestionnaires Lambda). Tous les composants appartiennent au même contexte délimité, ils appartiennent donc au même domaine.

Faites évoluer l'architecture en ajoutant des conteneurs, une base de données relationnelle et une API externe

Les conteneurs sont une bonne option pour les tâches de longue durée. Vous pouvez également utiliser une base de données relationnelle si vous disposez d'un schéma de données prédéfini et souhaitez bénéficier de la puissance du langage SQL. De plus, le domaine devrait communiquer avec des entités externes APIs. Vous pouvez faire évoluer l'exemple d'architecture pour répondre à ces exigences, comme indiqué dans le schéma suivant.

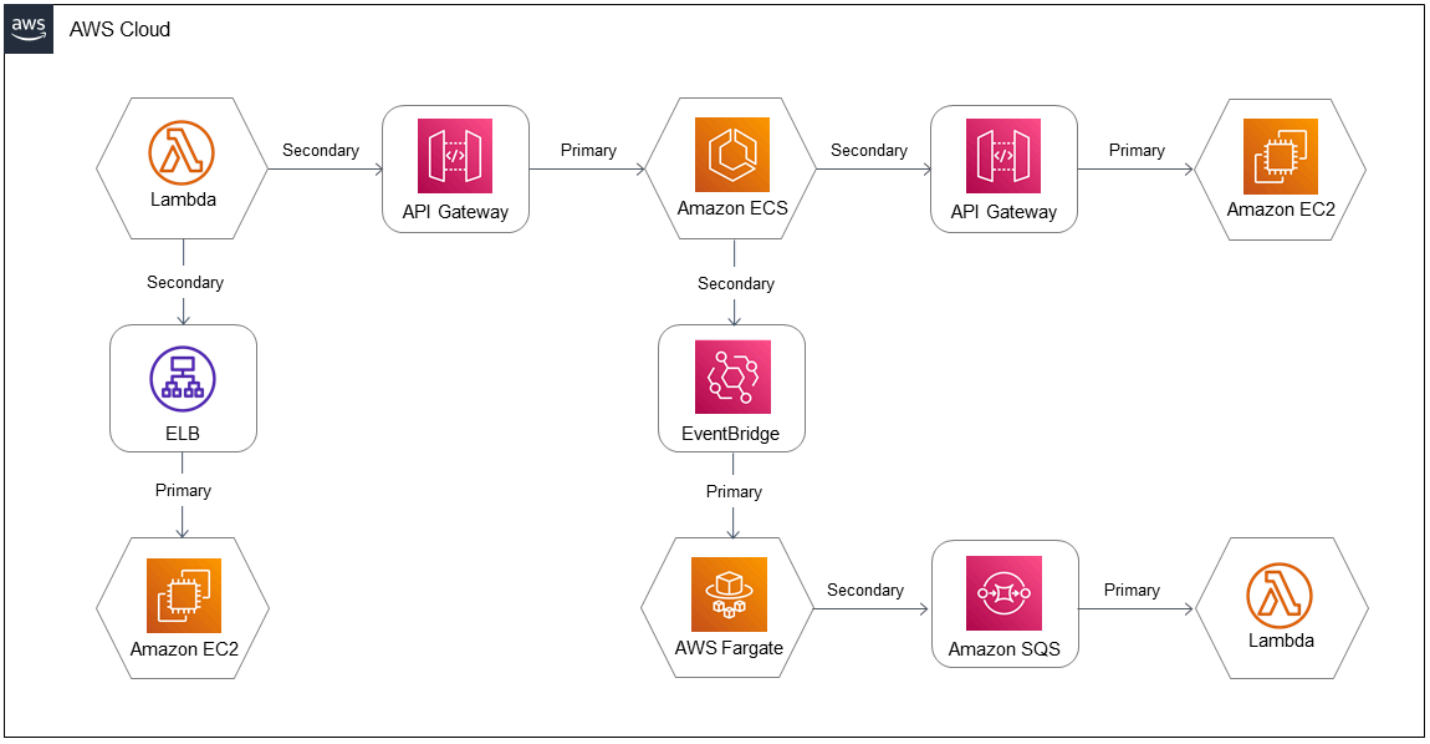


Cet exemple utilise Amazon ECS comme adaptateur principal pour lancer des tâches de longue durée dans le domaine. Amazon EventBridge (client) lance une tâche Amazon ECS (point d'entrée) lorsqu'un événement spécifique se produit. L'architecture inclut Amazon RDS comme autre adaptateur secondaire pour le stockage des données relationnelles. Il ajoute également une autre passerelle d'API en tant qu'adaptateur secondaire pour appeler un appel d'API externe. Par conséquent, l'architecture utilise plusieurs adaptateurs principaux et secondaires qui s'appuient sur différentes couches de calcul sous-jacentes dans un même domaine d'activité.

Le domaine est toujours couplé de manière souple à tous les adaptateurs principaux et secondaires par le biais d'abstractions appelées ports. Le domaine définit ce dont il a besoin du monde extérieur en utilisant des ports. Comme il incombe à l'adaptateur d'implémenter le port, le passage d'un adaptateur à un autre n'affecte pas le domaine. Par exemple, vous pouvez passer d'Amazon DynamoDB à Amazon RDS en écrivant un nouvel adaptateur, sans affecter le domaine.

Ajouter d'autres domaines (zoom arrière)

L'architecture hexagonale s'aligne bien avec les principes d'une architecture de microservices. Les exemples d'architecture présentés jusqu'à présent contenaient un seul domaine (ou un contexte limité). Les applications incluent généralement plusieurs domaines, qui doivent communiquer via des adaptateurs principaux et secondaires. Chaque domaine représente un microservice et est vaguement couplé à d'autres domaines.



Dans cette architecture, chaque domaine utilise un ensemble différent d'environnements informatiques. (Chaque domaine peut également avoir plusieurs environnements informatiques, comme dans l'exemple précédent.) Chaque domaine définit ses interfaces requises pour communiquer avec d'autres domaines via des ports. Les ports sont implémentés à l'aide d'adaptateurs principaux et secondaires. Ainsi, le domaine n'est pas affecté en cas de modification de l'adaptateur. De plus, les domaines sont découplés les uns des autres.

Dans l'exemple d'architecture présenté dans le schéma précédent, Lambda, Amazon EC2, Amazon ECS sont utilisés comme AWS Fargate adaptateurs principaux. API Gateway, Elastic Load Balancing et Amazon SQS sont utilisés comme adaptateurs secondaires. EventBridge

FAQ

Pourquoi utiliser une architecture hexagonale ?

L'architecture hexagonale permet aux développeurs de se concentrer sur la logique du domaine, simplifie l'automatisation des tests et améliore la qualité et l'adaptabilité du code. Ces améliorations se traduisent par un délai de mise sur le marché plus court et une mise à l'échelle technique et organisationnelle facilitée.

Pourquoi utiliser le design piloté par domaine ?

La conception axée sur le domaine (DDD) vous permet de créer des composants et des constructions logiciels en utilisant un langage commun entre les parties prenantes de l'entreprise et les ingénieurs. DDD vous aide à gérer la complexité des logiciels et constitue une stratégie efficace pour assurer la maintenance des produits logiciels sur le long terme.

Puis-je pratiquer le développement piloté par les tests sans architecture hexagonale ?

Oui. Le développement piloté par les tests (TDD) ne se limite pas à des modèles de conception logicielle spécifiques. Cependant, l'architecture hexagonale facilite la pratique du TDD.

Puis-je adapter mon produit sans architecture hexagonale ni conception axée sur le domaine ?

Oui. La mise à l'échelle technique et organisationnelle du produit peut être réalisée avec la plupart des modèles de conception. Cependant, l'architecture hexagonale et le DDD facilitent la mise à l'échelle et sont plus efficaces pour les grands projets à long terme.

Quelles technologies dois-je utiliser pour implémenter une architecture hexagonale ?

L'architecture hexagonale ne se limite pas à une pile technologique spécifique. Nous vous recommandons de choisir une technologie qui prend en charge l'inversion des dépendances et les tests unitaires.

Je développe un produit minimum viable. Est-il judicieux de passer du temps à réfléchir à l'architecture logicielle ?

Oui. Nous vous recommandons d'utiliser des modèles de conception qui vous sont familiers MVPs. Nous vous encourageons à essayer de pratiquer l'architecture hexagonale jusqu'à ce que vos ingénieurs soient à l'aise avec celle-ci. La mise en place d'une architecture hexagonale pour les nouveaux projets ne nécessite pas un investissement de temps beaucoup plus important que le fait de démarrer sans architecture.

Je développe un produit minimum viable et je n'ai pas le temps de rédiger des tests.

Si votre MVP contient une logique métier, nous vous recommandons vivement de rédiger des tests automatisés pour celui-ci. Cela permettra de réduire la boucle de rétroaction et de gagner du temps.

Quels modèles de design supplémentaires puis-je utiliser avec l'architecture hexagonale ?

Utilisez le modèle CQRS pour prendre en charge la mise à l'échelle de l'ensemble du système. Utilisez le modèle de référentiel pour stocker et restaurer votre modèle de domaine. Utilisez le modèle d'unité de travail pour gérer les étapes du processus transactionnel. Utilisez la composition plutôt que l'héritage pour modéliser des agrégats de domaines, des entités et des objets de valeur. Ne créez pas de hiérarchies d'objets complexes.

Étapes suivantes

- Familiarisez-vous davantage avec les concepts de conception axés sur le domaine en lisant les liens rassemblés dans la Ressources section.
- Si vous implémentez un nouveau projet, utilisez le modèle de structure de projet fourni dans ce guide et implémentez quelques fonctionnalités.
- Si vous êtes en train de mettre en œuvre un projet existant, identifiez le code qui peut être divisé entre les opérations en lecture seule et en écriture seule. Abstrayez le code en lecture seule dans les services de requête et placez le code en écriture seule dans les gestionnaires de commandes.
- Lorsque la structure de base de votre projet est en place, rédigez des tests unitaires, établissez une intégration continue (CI) avec l'automatisation des tests et suivez les pratiques de développement piloté par les tests (TDD).

Ressources

Références

- Structurez un projet Python en architecture hexagonale en utilisant AWS Lambda(modèle de AWS guidage prescriptif)
- Agile Teams (site Web Scaled Agile Framework)
- Modèles d'architecture avec Python, par Harry Percival et Bob Gregory (O'Reilly Media, 31 mars 2020), en particulier les chapitres suivants :
  - Commandes et gestionnaire de commandes
  - Ségrégation des responsabilités entre les commandes et les requêtes (CQRS)
  - Modèle de référentiel
- Event Storming : l'approche la plus intelligente de la collaboration au-delà des limites du silo, par Alberto Brandolini (site Web d'Event Storming)
- Démystifier la loi de Conway, par Sam Newman (site Web de Thoughtworks, 30 juin 2014)
- Développer une architecture évolutive avec AWS Lambda, par James Beswick (AWS Compute Blog, 8 juillet 2021)
- Langage de domaine : aborder la complexité au cœur du logiciel (site Web du langage de domaine)
- Facade, de Dive Into Design Patterns par Alexander Shvets (livre numérique, 5 décembre 2018)
- GivenWhenThen, de Martin Fowler (21 août 2013)
- Implémentation de la conception axée sur le domaine, par Vaughn Vernon (Addison-Wesley Professional, février 2013)
- Manœuvre inversée de Conway (site Web de Thoughtworks, 8 juillet 2014)
- Schéma du mois : Red Green Refactor (DZone site web, 2 juin 2017)
- Explication des principes de conception SOLID : principe d'inversion des dépendances avec exemples de code, par Thorben Janssen (site Web Stackify, 7 mai 2018)
- Principes SOLID : explication et exemples, par Simon Hoiberg (site Web ITNEXT, 1er janvier 2019)
- L'art du développement agile : le développement piloté par les tests, par James Shore et Shane Warden (O'Reilly Media, 25 mars 2010)
- Les principes SOLID de la programmation orientée objet expliqués en langage clair, par Yigit Kemal Erinc (articles sur la programmation freeCodeCamp orientée objet, 20 août 2020)

- Qu'est-ce qu'une architecture axée sur les événements ? (AWS site Web)

AWS services

- Amazon API Gateway
- Amazon Aurora
- Amazon DynamoDB
- Amazon Elastic Compute Cloud (Amazon EC2)
- Amazon Elastic Container Service (Amazon ECS)
- Amazon Elastic Kubernetes Service (Amazon EKS)
- Elastic Load Balancing
- Amazon EventBridge
- AWS Fargate
- AWS Lambda
- Amazon Relational Database Service (Amazon RDS)
- Amazon Simple Notification Service (Amazon SNS)
- Amazon Simple Queue Service (Amazon SQS)

Autres outils

- Moto
- LocalStack

Historique du document

Le tableau suivant décrit les modifications importantes apportées à ce guide. Pour être averti des mises à jour à venir, abonnez-vous à un fil RSS.

Modification	Description	Date
Publication initiale	—	15 juin 2022

AWS Glossaire des directives prescriptives

Les termes suivants sont couramment utilisés dans les stratégies, les guides et les modèles fournis par les directives AWS prescriptives. Pour suggérer des entrées, veuillez utiliser le lien Faire un commentaire à la fin du glossaire.

Nombres

7 R

Sept politiques de migration courantes pour transférer des applications vers le cloud. Ces politiques s'appuient sur les 5 R identifiés par Gartner en 2011 et sont les suivantes :

- Refactorisation/réarchitecture : transférez une application et modifiez son architecture en tirant pleinement parti des fonctionnalités natives cloud pour améliorer l'agilité, les performances et la capacité de mise à l'échelle. Cela implique généralement le transfert du système d'exploitation et de la base de données. Exemple : migrez votre base de données Oracle sur site vers l'édition compatible avec Amazon Aurora PostgreSQL.
- Replateformer (déplacer et remodeler) : transférez une application vers le cloud et introduisez un certain niveau d'optimisation pour tirer parti des fonctionnalités du cloud. Exemple : migrez votre base de données Oracle sur site vers Amazon Relational Database Service (Amazon RDS) pour Oracle dans le. AWS Cloud
- Racheter (rachat) : optez pour un autre produit, généralement en passant d'une licence traditionnelle à un modèle SaaS. Exemple : migrez votre système de gestion de la relation client (CRM) vers Salesforce.com.
- Réhéberger (lift and shift) : transférez une application vers le cloud sans apporter de modifications pour tirer parti des fonctionnalités du cloud. Exemple : migrez votre base de données Oracle sur site vers Oracle sur une instance EC2 dans le. AWS Cloud
- Relocaliser (lift and shift au niveau de l'hyperviseur) : transférez l'infrastructure vers le cloud sans acheter de nouveau matériel, réécrire des applications ou modifier vos opérations existantes. Vous migrez des serveurs d'une plateforme sur site vers un service cloud pour la même plateforme. Exemple : migrer une Microsoft Hyper-V application vers AWS.
- Retenir : conservez les applications dans votre environnement source. Il peut s'agir d'applications nécessitant une refactorisation majeure, que vous souhaitez retarder, et d'applications existantes que vous souhaitez retenir, car rien ne justifie leur migration sur le plan commercial.

- Retirer : mettez hors service ou supprimez les applications dont vous n'avez plus besoin dans votre environnement source.

A

ABAC

Voir contrôle d'accès basé sur les attributs.

services abstraits

Consultez la section Services gérés.

ACIDE

Voir atomicité, consistance, isolation, durabilité.

migration active-active

Méthode de migration de base de données dans laquelle la synchronisation des bases de données source et cible est maintenue (à l'aide d'un outil de réplication bidirectionnelle ou d'opérations d'écriture double), tandis que les deux bases de données gèrent les transactions provenant de la connexion d'applications pendant la migration. Cette méthode prend en charge la migration par petits lots contrôlés au lieu d'exiger un basculement ponctuel. Elle est plus flexible mais demande plus de travail qu'une migration active-passive.

migration active-passive

Méthode de migration de base de données dans laquelle les bases de données source et cible sont synchronisées, mais seule la base de données source gère les transactions liées à la connexion des applications pendant que les données sont répliquées vers la base de données cible. La base de données cible n'accepte aucune transaction pendant la migration.

fonction d'agrégation

Fonction SQL qui agit sur un groupe de lignes et calcule une valeur de retour unique pour le groupe. Des exemples de fonctions d'agrégation incluent `SUM` et`MAX`.

AI

Voir intelligence artificielle.

AIOps

Voir les opérations d'intelligence artificielle.

anonymisation

Processus de suppression définitive d'informations personnelles dans un ensemble de données. L'anonymisation peut contribuer à protéger la vie privée. Les données anonymisées ne sont plus considérées comme des données personnelles.

anti-motif

Solution fréquemment utilisée pour un problème récurrent lorsque la solution est contre-productive, inefficace ou moins efficace qu'une alternative.

contrôle des applications

Une approche de sécurité qui permet d'utiliser uniquement des applications approuvées afin de protéger un système contre les logiciels malveillants.

portefeuille d'applications

Ensemble d'informations détaillées sur chaque application utilisée par une organisation, y compris le coût de génération et de maintenance de l'application, ainsi que sa valeur métier. Ces informations sont essentielles pour le processus de découverte et d'analyse du portefeuille et permettent d'identifier et de prioriser les applications à migrer, à moderniser et à optimiser.

intelligence artificielle (IA)

Domaine de l'informatique consacré à l'utilisation des technologies de calcul pour exécuter des fonctions cognitives généralement associées aux humains, telles que l'apprentissage, la résolution de problèmes et la reconnaissance de modèles. Pour plus d'informations, veuillez consulter Qu'est-ce que l'intelligence artificielle ?

opérations d'intelligence artificielle (AIOps)

Processus consistant à utiliser des techniques de machine learning pour résoudre les problèmes opérationnels, réduire les incidents opérationnels et les interventions humaines, mais aussi améliorer la qualité du service. Pour plus d'informations sur son AIOps utilisation dans la stratégie de AWS migration, consultez le guide d'intégration des opérations.

chiffrement asymétrique

Algorithme de chiffrement qui utilise une paire de clés, une clé publique pour le chiffrement et une clé privée pour le déchiffrement. Vous pouvez partager la clé publique, car elle n'est pas utilisée pour le déchiffrement, mais l'accès à la clé privée doit être très restreint.

atomicité, cohérence, isolement, durabilité (ACID)

Ensemble de propriétés logicielles garantissant la validité des données et la fiabilité opérationnelle d'une base de données, même en cas d'erreur, de panne de courant ou d'autres problèmes.

contrôle d'accès par attributs (ABAC)

Pratique qui consiste à créer des autorisations détaillées en fonction des attributs de l'utilisateur, tels que le service, le poste et le nom de l'équipe. Pour plus d'informations, consultez ABAC pour AWS dans la documentation Gestion des identités et des accès AWS (IAM).

source de données faisant autorité

Emplacement où vous stockez la version principale des données, considérée comme la source d'information la plus fiable. Vous pouvez copier les données de la source de données officielle vers d'autres emplacements à des fins de traitement ou de modification des données, par exemple en les anonymisant, en les expurgant ou en les pseudonymisant.

Zone de disponibilité

Un emplacement distinct au sein d'un Région AWS réseau isolé des défaillances dans d'autres zones de disponibilité et fournissant une connectivité réseau peu coûteuse et à faible latence aux autres zones de disponibilité de la même région.

AWS Cadre d'adoption du cloud (AWS CAF)

Un cadre de directives et de meilleures pratiques visant AWS à aider les entreprises à élaborer un plan efficace pour réussir leur migration vers le cloud. AWS La CAF organise ses conseils en six domaines prioritaires appelés perspectives : les affaires, les personnes, la gouvernance, les plateformes, la sécurité et les opérations. Les perspectives d'entreprise, de personnes et de gouvernance mettent l'accent sur les compétences et les processus métier, tandis que les perspectives relatives à la plateforme, à la sécurité et aux opérations se concentrent sur les compétences et les processus techniques. Par exemple, la perspective liée aux personnes cible les parties prenantes qui s'occupent des ressources humaines (RH), des fonctions de dotation en personnel et de la gestion des personnes. Dans cette perspective, la AWS CAF fournit des conseils pour le développement du personnel, la formation et les communications afin de préparer l'organisation à une adoption réussie du cloud. Pour plus d'informations, veuillez consulter le site Web AWS CAF et le livre blanc AWS CAF.

AWS Cadre de qualification de la charge de travail (AWS WQF)

Outil qui évalue les charges de travail liées à la migration des bases de données, recommande des stratégies de migration et fournit des estimations de travail. AWS Le WQF est inclus avec

AWS Schema Conversion Tool (AWS SCT). Il analyse les schémas de base de données et les objets de code, le code d'application, les dépendances et les caractéristiques de performance, et fournit des rapports d'évaluation.

B

mauvais bot

Un bot destiné à perturber ou à nuire à des individus ou à des organisations.

BCP

Consultez la section Planification de la continuité des activités.

graphique de comportement

Vue unifiée et interactive des comportements des ressources et des interactions au fil du temps. Vous pouvez utiliser un graphique de comportement avec Amazon Detective pour examiner les tentatives de connexion infructueuses, les appels d'API suspects et les actions similaires. Pour plus d'informations, veuillez consulter Data in a behavior graph dans la documentation Detective.

système de poids fort

Système qui stocke d'abord l'octet le plus significatif. Voir aussi endianité.

classification binaire

Processus qui prédit un résultat binaire (l'une des deux classes possibles). Par exemple, votre modèle de machine learning peut avoir besoin de prévoir des problèmes tels que « Cet e-mail est-il du spam ou non ? » ou « Ce produit est-il un livre ou une voiture ? ».

filtre de Bloom

Structure de données probabiliste et efficace en termes de mémoire qui est utilisée pour tester si un élément fait partie d'un ensemble.

déploiement bleu/vert

Stratégie de déploiement dans laquelle vous créez deux environnements distincts mais identiques. Vous exécutez la version actuelle de l'application dans un environnement (bleu) et la nouvelle version de l'application dans l'autre environnement (vert). Cette stratégie vous permet de revenir rapidement en arrière avec un impact minimal.

bot

Application logicielle qui exécute des tâches automatisées sur Internet et simule l'activité ou l'interaction humaine. Certains robots sont utiles ou bénéfiques, comme les robots d'exploration Web qui indexent des informations sur Internet. D'autres robots, appelés « bots malveillants », sont destinés à perturber ou à nuire à des individus ou à des organisations.

botnet

Réseaux de robots infectés par des logiciels malveillants et contrôlés par une seule entité, connue sous le nom d'herder ou d'opérateur de bots. Les botnets sont le mécanisme le plus connu pour faire évoluer les bots et leur impact.

branche

Zone contenue d'un référentiel de code. La première branche créée dans un référentiel est la branche principale. Vous pouvez créer une branche à partir d'une branche existante, puis développer des fonctionnalités ou corriger des bogues dans la nouvelle branche. Une branche que vous créez pour générer une fonctionnalité est communément appelée branche de fonctionnalités. Lorsque la fonctionnalité est prête à être publiée, vous fusionnez à nouveau la branche de fonctionnalités dans la branche principale. Pour plus d'informations, consultez À propos des branches (GitHub documentation).

accès par brise-vitre

Dans des circonstances exceptionnelles et par le biais d'un processus approuvé, c'est un moyen rapide pour un utilisateur d'accéder à un accès auquel Compte AWS il n'est généralement pas autorisé. Pour plus d'informations, consultez l'indicateur Implementation break-glass procedures dans le guide Well-Architected AWS .

stratégie existante (brownfield)

L'infrastructure existante de votre environnement. Lorsque vous adoptez une stratégie existante pour une architecture système, vous concevez l'architecture en fonction des contraintes des systèmes et de l'infrastructure actuels. Si vous étendez l'infrastructure existante, vous pouvez combiner des politiques brownfield (existantes) et greenfield (inédites).

cache de tampon

Zone de mémoire dans laquelle sont stockées les données les plus fréquemment consultées.

capacité métier

Ce que fait une entreprise pour générer de la valeur (par exemple, les ventes, le service client ou le marketing). Les architectures de microservices et les décisions de développement

peuvent être dictées par les capacités métier. Pour plus d'informations, veuillez consulter la section Organisation en fonction des capacités métier du livre blanc Exécution de microservices conteneurisés sur AWS.

planification de la continuité des activités (BCP)

Plan qui tient compte de l'impact potentiel d'un événement perturbateur, tel qu'une migration à grande échelle, sur les opérations, et qui permet à une entreprise de reprendre ses activités rapidement.

C

CAF

Voir le cadre d'adoption du AWS cloud.

déploiement de Canary

Diffusion lente et progressive d'une version pour les utilisateurs finaux. Lorsque vous êtes sûr, vous déployez la nouvelle version et remplacez la version actuelle dans son intégralité.

CCo E

Voir le Centre d'excellence du cloud.

CDC

Voir capture des données de modification.

capture des données de modification (CDC)

Processus de suivi des modifications apportées à une source de données, telle qu'une table de base de données, et d'enregistrement des métadonnées relatives à ces modifications. Vous pouvez utiliser la CDC à diverses fins, telles que l'audit ou la réplication des modifications dans un système cible afin de maintenir la synchronisation.

ingénierie du chaos

Introduire intentionnellement des défaillances ou des événements perturbateurs pour tester la résilience d'un système. Vous pouvez utiliser AWS Fault Injection Service (AWS FIS) pour effectuer des expériences qui stressent vos AWS charges de travail et évaluer leur réponse.

CI/CD

Découvrez l'intégration continue et la livraison continue.

classification

Processus de catégorisation qui permet de générer des prédictions. Les modèles de ML pour les problèmes de classification prédisent une valeur discrète. Les valeurs discrètes se distinguent toujours les unes des autres. Par exemple, un modèle peut avoir besoin d'évaluer la présence ou non d'une voiture sur une image.

chiffrement côté client

Chiffrement des données localement, avant que la cible ne les Service AWS reçoive.

Centre d'excellence du cloud (CCoE)

Une équipe multidisciplinaire qui dirige les efforts d'adoption du cloud au sein d'une organisation, notamment en développant les bonnes pratiques en matière de cloud, en mobilisant des ressources, en établissant des délais de migration et en guidant l'organisation dans le cadre de transformations à grande échelle. Pour plus d'informations, consultez les CCoarticles électroniques du blog sur la stratégie AWS Cloud d'entreprise.

cloud computing

Technologie cloud généralement utilisée pour le stockage de données à distance et la gestion des appareils IoT. Le cloud computing est généralement associé à la technologie informatique de pointe.

modèle d'exploitation du cloud

Dans une organisation informatique, modèle d'exploitation utilisé pour créer, faire évoluer et optimiser un ou plusieurs environnements cloud. Pour plus d'informations, consultez la section Création de votre modèle d'exploitation cloud.

étapes d'adoption du cloud

Les quatre phases que les entreprises traversent généralement lorsqu'elles migrent vers AWS Cloud :

- Projet : exécution de quelques projets liés au cloud à des fins de preuve de concept et d'apprentissage
- Base : réaliser des investissements fondamentaux pour accélérer votre adoption du cloud (par exemple, créer une zone de landing zone, définir un CCo E, établir un modèle opérationnel)
- Migration : migration d'applications individuelles
- Réinvention : optimisation des produits et services et innovation dans le cloud

Ces étapes ont été définies par Stephen Orban dans le billet de blog The Journey Toward Cloud-First & the Stages of Adoption publié sur le blog AWS Cloud Enterprise Strategy. Pour plus d'informations sur leur lien avec la stratégie de AWS migration, consultez le guide de préparation à la migration.

CMDB

Consultez la base de données de gestion des configurations.

référentiel de code

Emplacement où le code source et d'autres ressources, comme la documentation, les exemples et les scripts, sont stockés et mis à jour par le biais de processus de contrôle de version. Les référentiels cloud courants incluent GitHub ouBitbucket Cloud. Chaque version du code est appelée branche. Dans une structure de microservice, chaque référentiel est consacré à une seule fonctionnalité. Un seul pipeline CI/CD peut utiliser plusieurs référentiels.

cache passif

Cache tampon vide, mal rempli ou contenant des données obsolètes ou non pertinentes. Cela affecte les performances, car l'instance de base de données doit lire à partir de la mémoire principale ou du disque, ce qui est plus lent que la lecture à partir du cache tampon.

données gelées

Données rarement consultées et généralement historiques. Lorsque vous interrogez ce type de données, les requêtes lentes sont généralement acceptables. Le transfert de ces données vers des niveaux ou classes de stockage moins performants et moins coûteux peut réduire les coûts.

vision par ordinateur (CV)

Domaine de l'IA qui utilise l'apprentissage automatique pour analyser et extraire des informations à partir de formats visuels tels que des images numériques et des vidéos. Par exemple, Amazon SageMaker AI fournit des algorithmes de traitement d'image pour les CV.

dérive de configuration

Pour une charge de travail, une modification de configuration par rapport à l'état attendu. Cela peut entraîner une non-conformité de la charge de travail, et cela est généralement progressif et involontaire.

base de données de gestion des configurations (CMDB)

Référentiel qui stocke et gère les informations relatives à une base de données et à son environnement informatique, y compris les composants matériels et logiciels ainsi que leurs

configurations. Vous utilisez généralement les données d'une CMDB lors de la phase de découverte et d'analyse du portefeuille de la migration.

pack de conformité

Ensemble de AWS Config règles et d'actions correctives que vous pouvez assembler pour personnaliser vos contrôles de conformité et de sécurité. Vous pouvez déployer un pack de conformité en tant qu'entité unique dans une région Compte AWS et, ou au sein d'une organisation, à l'aide d'un modèle YAML. Pour plus d'informations, consultez la section Packs de conformité dans la AWS Config documentation.

intégration continue et livraison continue (CI/CD)

Processus d'automatisation des étapes de source, de construction, de test, de préparation et de production du processus de publication du logiciel. CI/CD est communément décrit comme un pipeline. CI/CD peut vous aider à automatiser les processus, à améliorer la productivité, à améliorer la qualité du code et à accélérer les livraisons. Pour plus d'informations, veuillez consulter Avantages de la livraison continue. CD peut également signifier déploiement continu. Pour plus d'informations, veuillez consulter Livraison continue et déploiement continu.

CV

Voir vision par ordinateur.

D

données au repos

Données stationnaires dans votre réseau, telles que les données stockées.

classification des données

Processus permettant d'identifier et de catégoriser les données de votre réseau en fonction de leur sévérité et de leur sensibilité. Il s'agit d'un élément essentiel de toute stratégie de gestion des risques de cybersécurité, car il vous aide à déterminer les contrôles de protection et de conservation appropriés pour les données. La classification des données est une composante du pilier de sécurité du AWS Well-Architected Framework. Pour plus d'informations, veuillez consulter Classification des données.

dérive des données

Une variation significative entre les données de production et les données utilisées pour entraîner un modèle ML, ou une modification significative des données d'entrée au fil du temps. La dérive

des données peut réduire la qualité, la précision et l'équité globales des prédictions des modèles ML.

données en transit

Données qui circulent activement sur votre réseau, par exemple entre les ressources du réseau.

maillage de données

Un cadre architectural qui fournit une propriété des données distribuée et décentralisée avec une gestion et une gouvernance centralisées.

minimisation des données

Le principe de collecte et de traitement des seules données strictement nécessaires. La pratique de la minimisation des données AWS Cloud peut réduire les risques liés à la confidentialité, les coûts et l'empreinte carbone de vos analyses.

périmètre de données

Ensemble de garde-fous préventifs dans votre AWS environnement qui permettent de garantir que seules les identités fiables accèdent aux ressources fiables des réseaux attendus. Pour plus d'informations, voir Création d'un périmètre de données sur AWS.

prétraitement des données

Pour transformer les données brutes en un format facile à analyser par votre modèle de ML. Le prétraitement des données peut impliquer la suppression de certaines colonnes ou lignes et le traitement des valeurs manquantes, incohérentes ou en double.

provenance des données

Le processus de suivi de l'origine et de l'historique des données tout au long de leur cycle de vie, par exemple la manière dont les données ont été générées, transmises et stockées.

sujet des données

Personne dont les données sont collectées et traitées.

entrepôt des données

Un système de gestion des données qui prend en charge les informations commerciales, telles que les analyses. Les entrepôts de données contiennent généralement de grandes quantités de données historiques et sont généralement utilisés pour les requêtes et les analyses.

langage de définition de base de données (DDL)

Instructions ou commandes permettant de créer ou de modifier la structure des tables et des objets dans une base de données.

langage de manipulation de base de données (DML)

Instructions ou commandes permettant de modifier (insérer, mettre à jour et supprimer) des informations dans une base de données.

DDL

Voir langage de définition de base de données.

ensemble profond

Sert à combiner plusieurs modèles de deep learning à des fins de prédiction. Vous pouvez utiliser des ensembles profonds pour obtenir une prévision plus précise ou pour estimer l'incertitude des prédictions.

deep learning

Un sous-champ de ML qui utilise plusieurs couches de réseaux neuronaux artificiels pour identifier le mappage entre les données d'entrée et les variables cibles d'intérêt.

defense-in-depth

Approche de la sécurité de l'information dans laquelle une série de mécanismes et de contrôles de sécurité sont judicieusement répartis sur l'ensemble d'un réseau informatique afin de protéger la confidentialité, l'intégrité et la disponibilité du réseau et des données qu'il contient. Lorsque vous adoptez cette stratégie AWS, vous ajoutez plusieurs contrôles à différentes couches de la AWS Organizations structure afin de sécuriser les ressources. Par exemple, une defense-in-depth approche peut combiner l'authentification multifactorielle, la segmentation du réseau et le chiffrement.

administrateur délégué

Dans AWS Organizations, un service compatible peut enregistrer un compte AWS membre pour administrer les comptes de l'organisation et gérer les autorisations pour ce service. Ce compte est appelé administrateur délégué pour ce service. Pour plus d'informations et une liste des services compatibles, veuillez consulter la rubrique Services qui fonctionnent avec AWS Organizations dans la documentation AWS Organizations .

déploiement

Processus de mise à disposition d'une application, de nouvelles fonctionnalités ou de corrections de code dans l'environnement cible. Le déploiement implique la mise en œuvre de modifications dans une base de code, puis la génération et l'exécution de cette base de code dans les environnements de l'application.

environnement de développement

Voir environnement.

contrôle de détection

Contrôle de sécurité conçu pour détecter, journaliser et alerter après la survenue d'un événement. Ces contrôles constituent une deuxième ligne de défense et vous alertent en cas d'événements de sécurité qui ont contourné les contrôles préventifs en place. Pour plus d'informations, veuillez consulter la rubrique Contrôles de détection dans Implementing security controls on AWS.

cartographie de la chaîne de valeur du développement (DVSM)

Processus utilisé pour identifier et hiérarchiser les contraintes qui nuisent à la rapidité et à la qualité du cycle de vie du développement logiciel. DVSM étend le processus de cartographie de la chaîne de valeur initialement conçu pour les pratiques de production allégée. Il met l'accent sur les étapes et les équipes nécessaires pour créer et transférer de la valeur tout au long du processus de développement logiciel.

jumeau numérique

Représentation virtuelle d'un système réel, tel qu'un bâtiment, une usine, un équipement industriel ou une ligne de production. Les jumeaux numériques prennent en charge la maintenance prédictive, la surveillance à distance et l'optimisation de la production.

tableau des dimensions

Dans un schéma en étoile, table plus petite contenant les attributs de données relatifs aux données quantitatives d'une table de faits. Les attributs des tables de dimensions sont généralement des champs de texte ou des nombres discrets qui se comportent comme du texte. Ces attributs sont couramment utilisés pour la contrainte des requêtes, le filtrage et l'étiquetage des ensembles de résultats.

catastrophe

Un événement qui empêche une charge de travail ou un système d'atteindre ses objectifs commerciaux sur son site de déploiement principal. Ces événements peuvent être des

catastrophes naturelles, des défaillances techniques ou le résultat d'actions humaines, telles qu'une mauvaise configuration involontaire ou une attaque de logiciel malveillant.

reprise après sinistre (DR)

La stratégie et le processus que vous utilisez pour minimiser les temps d'arrêt et les pertes de données causés par un sinistre. Pour plus d'informations, consultez Disaster Recovery of Workloads on AWS : Recovery in the Cloud in the AWS Well-Architected Framework.

DML

Voir langage de manipulation de base de données.

conception axée sur le domaine

Approche visant à développer un système logiciel complexe en connectant ses composants à des domaines évolutifs, ou objectifs métier essentiels, que sert chaque composant. Ce concept a été introduit par Eric Evans dans son ouvrage Domain-Driven Design: Tackling Complexity in the Heart of Software (Boston : Addison-Wesley Professional, 2003). Pour plus d'informations sur l'utilisation du design piloté par domaine avec le modèle de figuier étrangleur, veuillez consulter Modernizing legacy Microsoft ASP.NET (ASMX) web services incrementally by using containers and Amazon API Gateway.

DR

Voir reprise après sinistre.

détection de dérive

Suivi des écarts par rapport à une configuration de référence. Par exemple, vous pouvez l'utiliser AWS CloudFormation pour détecter la dérive des ressources du système ou AWS Control Tower pour détecter les modifications de votre zone d'atterrissage susceptibles d'affecter le respect des exigences de gouvernance.

DVSM

Voir la cartographie de la chaîne de valeur du développement.

E

EDA

Voir analyse exploratoire des données.

EDI

Voir échange de données informatisé.

informatique de périphérie

Technologie qui augmente la puissance de calcul des appareils intelligents en périphérie d'un réseau IoT. Comparé au cloud computing, l'informatique de pointe peut réduire la latence des communications et améliorer le temps de réponse.

échange de données informatisé (EDI)

L'échange automatique de documents commerciaux entre les organisations. Pour plus d'informations, voir Qu'est-ce que l'échange de données informatisé ?

chiffrement

Processus informatique qui transforme des données en texte clair, lisibles par l'homme, en texte chiffré.

clé de chiffrement

Chaîne cryptographique de bits aléatoires générée par un algorithme cryptographique. La longueur des clés peut varier, et chaque clé est conçue pour être imprévisible et unique.

endianisme

Ordre selon lequel les octets sont stockés dans la mémoire de l'ordinateur. Les systèmes de poids fort stockent d'abord l'octet le plus significatif. Les systèmes de poids faible stockent d'abord l'octet le moins significatif.

point de terminaison

Voir point de terminaison de service.

service de point de terminaison

Service que vous pouvez héberger sur un cloud privé virtuel (VPC) pour le partager avec d'autres utilisateurs. Vous pouvez créer un service de point de terminaison avec AWS PrivateLink et accorder des autorisations à d'autres Comptes AWS ou à Gestion des identités et des accès AWS (IAM) principaux. Ces comptes ou principaux peuvent se connecter à votre service de point de terminaison de manière privée en créant des points de terminaison d'un VPC d'interface. Pour plus d'informations, veuillez consulter Création d'un service de point de terminaison dans la documentation Amazon Virtual Private Cloud (Amazon VPC).

planification des ressources d'entreprise (ERP)

Système qui automatise et gère les principaux processus métier (tels que la comptabilité, le MES et la gestion de projet) pour une entreprise.

chiffrement d'enveloppe

Processus de chiffrement d'une clé de chiffrement à l'aide d'une autre clé de chiffrement. Pour plus d'informations, consultez la section Chiffrement des enveloppes dans la documentation AWS Key Management Service (AWS KMS).

environment

Instance d'une application en cours d'exécution. Les types d'environnement les plus courants dans le cloud computing sont les suivants :

- Environnement de développement : instance d'une application en cours d'exécution à laquelle seule l'équipe principale chargée de la maintenance de l'application peut accéder. Les environnements de développement sont utilisés pour tester les modifications avant de les promouvoir dans les environnements supérieurs. Ce type d'environnement est parfois appelé environnement de test.
- Environnements inférieurs : tous les environnements de développement d'une application, tels que ceux utilisés pour les générations et les tests initiaux.
- Environnement de production : instance d'une application en cours d'exécution à laquelle les utilisateurs finaux peuvent accéder. Dans un CI/CD pipeline, l'environnement de production est le dernier environnement de déploiement.
- Environnements supérieurs : tous les environnements accessibles aux utilisateurs autres que l'équipe de développement principale. Ils peuvent inclure un environnement de production, des environnements de préproduction et des environnements pour les tests d'acceptation par les utilisateurs.

épopée

Dans les méthodologies agiles, catégories fonctionnelles qui aident à organiser et à prioriser votre travail. Les épopées fournissent une description détaillée des exigences et des tâches d'implémentation. Par exemple, les points forts de la AWS CAF en matière de sécurité incluent la gestion des identités et des accès, les contrôles de détection, la sécurité des infrastructures, la protection des données et la réponse aux incidents. Pour plus d'informations sur les épopées dans la stratégie de migration AWS , veuillez consulter le guide d'implémentation du programme.

ERP

Voir Planification des ressources d'entreprise.

analyse exploratoire des données (EDA)

Processus d'analyse d'un jeu de données pour comprendre ses principales caractéristiques. Vous collectez ou agrégez des données, puis vous effectuez des enquêtes initiales pour trouver des modèles, détecter des anomalies et vérifier les hypothèses. L'EDA est réalisée en calculant des statistiques récapitulatives et en créant des visualisations de données.

F

tableau des faits

La table centrale dans un schéma en étoile. Il stocke des données quantitatives sur les opérations commerciales. Généralement, une table de faits contient deux types de colonnes : celles qui contiennent des mesures et celles qui contiennent une clé étrangère pour une table de dimensions.

échouer rapidement

Une philosophie qui utilise des tests fréquents et progressifs pour réduire le cycle de vie du développement. C'est un élément essentiel d'une approche agile.

limite d'isolation des défauts

Dans le AWS Cloud, une limite telle qu'une zone de disponibilité Région AWS, un plan de contrôle ou un plan de données qui limite l'effet d'une panne et contribue à améliorer la résilience des charges de travail. Pour plus d'informations, consultez la section Limites d'isolation des AWS pannes.

branche de fonctionnalités

Voir succursale.

fonctionnalités

Les données d'entrée que vous utilisez pour faire une prédiction. Par exemple, dans un contexte de fabrication, les fonctionnalités peuvent être des images capturées périodiquement à partir de la ligne de fabrication.

importance des fonctionnalités

Le niveau d'importance d'une fonctionnalité pour les prédictions d'un modèle. Il s'exprime généralement sous la forme d'un score numérique qui peut être calculé à l'aide de différentes techniques, telles que la méthode Shapley Additive Explanations (SHAP) et les gradients intégrés. Pour plus d'informations, voir Interprétabilité du modèle d'apprentissage automatique avec AWS.

transformation de fonctionnalité

Optimiser les données pour le processus de ML, notamment en enrichissant les données avec des sources supplémentaires, en mettant à l'échelle les valeurs ou en extrayant plusieurs ensembles d'informations à partir d'un seul champ de données. Cela permet au modèle de ML de tirer parti des données. Par exemple, si vous décomposez la date « 2021-05-27 00:15:37 » en « 2021 », « mai », « jeudi » et « 15 », vous pouvez aider l'algorithme d'apprentissage à apprendre des modèles nuancés associés à différents composants de données.

invitation en quelques coups

Fournir à un LLM un petit nombre d'exemples illustrant la tâche et le résultat souhaité avant de lui demander d'effectuer une tâche similaire. Cette technique est une application de l'apprentissage contextuel, dans le cadre de laquelle les modèles apprennent à partir d'exemples (prises de vue) intégrés dans des instructions. Les instructions en quelques étapes peuvent être efficaces pour les tâches qui nécessitent un formatage, un raisonnement ou des connaissances de domaine spécifiques. Voir également l'invite Zero-Shot.

FGAC

Découvrez le contrôle d'accès détaillé.

contrôle d'accès détaillé (FGAC)

Utilisation de plusieurs conditions pour autoriser ou refuser une demande d'accès.

migration instantanée (flash-cut)

Méthode de migration de base de données qui utilise la réplication continue des données par le biais de la capture des données de modification afin de migrer les données dans les plus brefs délais, au lieu d'utiliser une approche progressive. L'objectif est de réduire au maximum les temps d'arrêt.

FM

Voir le modèle de fondation.

modèle de fondation (FM)

Un vaste réseau neuronal d'apprentissage profond qui s'est entraîné sur d'énormes ensembles de données généralisées et non étiquetées. FMs sont capables d'effectuer une grande variété de tâches générales, telles que comprendre le langage, générer du texte et des images et converser en langage naturel. Pour plus d'informations, voir Que sont les modèles de base ?

G

IA générative

Sous-ensemble de modèles d'IA qui ont été entraînés sur de grandes quantités de données et qui peuvent utiliser une simple invite textuelle pour créer de nouveaux contenus et artefacts, tels que des images, des vidéos, du texte et du son. Pour plus d'informations, consultez Qu'est-ce que l'IA générative.

blocage géographique

Voir les restrictions géographiques.

restrictions géographiques (blocage géographique)

Sur Amazon CloudFront, option permettant d'empêcher les utilisateurs de certains pays d'accéder aux distributions de contenu. Vous pouvez utiliser une liste d'autorisation ou une liste de blocage pour spécifier les pays approuvés et interdits. Pour plus d'informations, consultez la section Restreindre la distribution géographique de votre contenu dans la CloudFront documentation.

Flux de travail Gitflow

Approche dans laquelle les environnements inférieurs et supérieurs utilisent différentes branches dans un référentiel de code source. Le flux de travail Gitflow est considéré comme existant, et le flux de travail basé sur les troncs est l'approche moderne préférée.

image dorée

Un instantané d'un système ou d'un logiciel utilisé comme modèle pour déployer de nouvelles instances de ce système ou logiciel. Par exemple, dans le secteur de la fabrication, une image dorée peut être utilisée pour fournir des logiciels sur plusieurs appareils et contribue à améliorer la vitesse, l'évolutivité et la productivité des opérations de fabrication des appareils.

stratégie inédite

L'absence d'infrastructures existantes dans un nouvel environnement. Lorsque vous adoptez une stratégie inédite pour une architecture système, vous pouvez sélectionner toutes les nouvelles technologies sans restriction de compatibilité avec l'infrastructure existante, également appelée brownfield. Si vous étendez l'infrastructure existante, vous pouvez combiner des politiques brownfield (existantes) et greenfield (inédites).

barrière de protection

Règle de haut niveau qui permet de régir les ressources, les politiques et la conformité au sein des unités organisationnelles (OUs). Les barrières de protection préventives appliquent des politiques pour garantir l'alignement sur les normes de conformité. Elles sont mises en œuvre à l'aide de politiques de contrôle des services et de limites des autorisations IAM. Les barrières de protection de détection détectent les violations des politiques et les problèmes de conformité, et génèrent des alertes pour y remédier. Ils sont implémentés à l'aide d'Amazon AWS Config AWS Security Hub CSPM GuardDuty AWS Trusted Advisor, d'Amazon Inspector et de AWS Lambda contrôles personnalisés.

H

HA

Découvrez la haute disponibilité.

migration de base de données hétérogène

Migration de votre base de données source vers une base de données cible qui utilise un moteur de base de données différent (par exemple, Oracle vers Amazon Aurora). La migration hétérogène fait généralement partie d'un effort de réarchitecture, et la conversion du schéma peut s'avérer une tâche complexe. AWS propose AWS SCT qui facilite les conversions de schémas.

haute disponibilité (HA)

Capacité d'une charge de travail à fonctionner en continu, sans intervention, en cas de difficultés ou de catastrophes. Les systèmes HA sont conçus pour basculer automatiquement, fournir constamment des performances de haute qualité et gérer différentes charges et défaillances avec un impact minimal sur les performances.

modernisation des historiens

Approche utilisée pour moderniser et mettre à niveau les systèmes de technologie opérationnelle (OT) afin de mieux répondre aux besoins de l'industrie manufacturière. Un historien est un type de base de données utilisé pour collecter et stocker des données provenant de diverses sources dans une usine.

données de rétention

Partie de données historiques étiquetées qui n'est pas divulguée dans un ensemble de données utilisé pour entraîner un modèle d'apprentissage automatique. Vous pouvez utiliser les données de blocage pour évaluer les performances du modèle en comparant les prévisions du modèle aux données de blocage.

migration de base de données homogène

Migration de votre base de données source vers une base de données cible qui partage le même moteur de base de données (par exemple, Microsoft SQL Server vers Amazon RDS for SQL Server). La migration homogène s'inscrit généralement dans le cadre d'un effort de réhébergement ou de replateforme. Vous pouvez utiliser les utilitaires de base de données natifs pour migrer le schéma.

données chaudes

Données fréquemment consultées, telles que les données en temps réel ou les données translationnelles récentes. Ces données nécessitent généralement un niveau ou une classe de stockage à hautes performances pour fournir des réponses rapides aux requêtes.

correctif

Solution d'urgence à un problème critique dans un environnement de production. En raison de son urgence, un correctif est généralement créé en dehors du flux de travail de DevOps publication habituel.

période de soins intensifs

Immédiatement après le basculement, période pendant laquelle une équipe de migration gère et surveille les applications migrées dans le cloud afin de résoudre les problèmes éventuels. En règle générale, cette période dure de 1 à 4 jours. À la fin de la période de soins intensifs, l'équipe de migration transfère généralement la responsabilité des applications à l'équipe des opérations cloud.

I

IaC

Considérez l'infrastructure comme un code.

politique basée sur l’identité

Politique attachée à un ou plusieurs principaux IAM qui définit leurs autorisations au sein de l' AWS Cloud environnement.

application inactive

Application dont l'utilisation moyenne du processeur et de la mémoire se situe entre 5 et 20 % sur une période de 90 jours. Dans un projet de migration, il est courant de retirer ces applications ou de les retenir sur site.

IIo T

Voir Internet industriel des objets.

infrastructure immuable

Modèle qui déploie une nouvelle infrastructure pour les charges de travail de production au lieu de mettre à jour, d'appliquer des correctifs ou de modifier l'infrastructure existante. Les infrastructures immuables sont intrinsèquement plus cohérentes, fiables et prévisibles que les infrastructures mutables. Pour plus d'informations, consultez les meilleures pratiques de déploiement à l'aide d'une infrastructure immuable dans le AWS Well-Architected Framework.

VPC entrant (d'entrée)

Dans une architecture AWS multi-comptes, un VPC qui accepte, inspecte et achemine les connexions réseau depuis l'extérieur d'une application. L'architecture AWS de référence de sécurité recommande de configurer votre compte réseau avec les fonctions entrantes, sortantes et d'inspection VPCs afin de protéger l'interface bidirectionnelle entre votre application et l'Internet en général.

migration incrémentielle

Stratégie de basculement dans le cadre de laquelle vous migrez votre application par petites parties au lieu d'effectuer un basculement complet unique. Par exemple, il se peut que vous ne transfériez que quelques microservices ou utilisateurs vers le nouveau système dans un premier temps. Après avoir vérifié que tout fonctionne correctement, vous pouvez transférer

progressivement des microservices ou des utilisateurs supplémentaires jusqu'à ce que vous puissiez mettre hors service votre système hérité. Cette stratégie réduit les risques associés aux migrations de grande ampleur.

Industry 4.0

Un terme introduit par Klaus Schwab en 2016 pour désigner la modernisation des processus de fabrication grâce aux avancées en matière de connectivité, de données en temps réel, d'automatisation, d'analyse et d'IA/ML.

infrastructure

Ensemble des ressources et des actifs contenus dans l'environnement d'une application.

infrastructure en tant que code (IaC)

Processus de mise en service et de gestion de l'infrastructure d'une application via un ensemble de fichiers de configuration. IaC est conçue pour vous aider à centraliser la gestion de l'infrastructure, à normaliser les ressources et à mettre à l'échelle rapidement afin que les nouveaux environnements soient reproductibles, fiables et cohérents.

Internet industriel des objets (IIoT)

L'utilisation de capteurs et d'appareils connectés à Internet dans les secteurs industriels tels que la fabrication, l'énergie, l'automobile, les soins de santé, les sciences de la vie et l'agriculture. Pour plus d'informations, voir Élaboration d'une stratégie de transformation numérique de l'Internet des objets (IIoT) industriel.

VPC d'inspection

Dans une architecture AWS multi-comptes, un VPC centralisé qui gère les inspections du trafic réseau VPCs entre (identique ou Régions AWS différent), Internet et les réseaux locaux. L'architecture AWS de référence de sécurité recommande de configurer votre compte réseau avec les fonctions entrantes, sortantes et d'inspection VPCs afin de protéger l'interface bidirectionnelle entre votre application et l'Internet en général.

Internet des objets (IoT)

Réseau d'objets physiques connectés dotés de capteurs ou de processeurs intégrés qui communiquent avec d'autres appareils et systèmes via Internet ou via un réseau de communication local. Pour plus d'informations, veuillez consulter la section Qu'est-ce que l'IoT ?.

interprétabilité

Caractéristique d'un modèle de machine learning qui décrit dans quelle mesure un être humain peut comprendre comment les prédictions du modèle dépendent de ses entrées. Pour plus d'informations, voir Interprétabilité du modèle d'apprentissage automatique avec AWS.

IoT

Voir Internet des objets.

Bibliothèque d'informations informatiques (ITIL)

Ensemble de bonnes pratiques pour proposer des services informatiques et les aligner sur les exigences métier. L'ITIL constitue la base de l'ITSM.

gestion des services informatiques (ITSM)

Activités associées à la conception, à la mise en œuvre, à la gestion et à la prise en charge de services informatiques d'une organisation. Pour plus d'informations sur l'intégration des opérations cloud aux outils ITSM, veuillez consulter le guide d'intégration des opérations.

ITIL

Consultez la bibliothèque d'informations informatiques.

ITSM

Voir Gestion des services informatiques.

L

contrôle d'accès basé sur des étiquettes (LBAC)

Une implémentation du contrôle d'accès obligatoire (MAC) dans laquelle une valeur d'étiquette de sécurité est explicitement attribuée aux utilisateurs et aux données elles-mêmes. L'intersection entre l'étiquette de sécurité utilisateur et l'étiquette de sécurité des données détermine les lignes et les colonnes visibles par l'utilisateur.

zone de destination

Une zone d'atterrissage est un AWS environnement multi-comptes bien conçu, évolutif et sécurisé. Il s'agit d'un point de départ à partir duquel vos entreprises peuvent rapidement lancer et déployer des charges de travail et des applications en toute confiance dans leur environnement de sécurité et d'infrastructure. Pour plus d'informations sur les zones de destination, veuillez consulter Setting up a secure and scalable multi-account AWS environment.

grand modèle de langage (LLM)

Un modèle d'intelligence artificielle basé sur le deep learning qui est préentraîné sur une grande quantité de données. Un LLM peut effectuer plusieurs tâches, telles que répondre à des questions, résumer des documents, traduire du texte dans d'autres langues et compléter des phrases. Pour plus d'informations, voir Que sont LLMs.

migration de grande envergure

Migration de 300 serveurs ou plus.

LBAC

Voir contrôle d'accès basé sur des étiquettes.

principe de moindre privilège

Bonne pratique de sécurité qui consiste à accorder les autorisations minimales nécessaires à l'exécution d'une tâche. Pour plus d'informations, veuillez consulter la rubrique Accorder les autorisations de moindre privilège dans la documentation IAM.

lift and shift

Voir 7 Rs.

système de poids faible

Système qui stocke d'abord l'octet le moins significatif. Voir aussi endianité.

LLM

Voir le grand modèle de langage.

environnements inférieurs

Voir environnement.

M

machine learning (ML)

Type d'intelligence artificielle qui utilise des algorithmes et des techniques pour la reconnaissance et l'apprentissage de modèles. Le ML analyse et apprend à partir de données enregistrées, telles que les données de l'Internet des objets (IoT), pour générer un modèle statistique basé sur des modèles. Pour plus d'informations, veuillez consulter Machine Learning.

branche principale

Voir succursale.

malware

Logiciel conçu pour compromettre la sécurité ou la confidentialité de l'ordinateur. Les logiciels malveillants peuvent perturber les systèmes informatiques, divulguer des informations sensibles ou obtenir un accès non autorisé. Parmi les malwares, on peut citer les virus, les vers, les rançongiciels, les chevaux de Troie, les logiciels espions et les enregistreurs de frappe.

services gérés

Services AWS pour lequel AWS fonctionnent la couche d'infrastructure, le système d'exploitation et les plateformes, et vous accédez aux points de terminaison pour stocker et récupérer des données. Amazon Simple Storage Service (Amazon S3) et Amazon DynamoDB sont des exemples de services gérés. Ils sont également connus sous le nom de services abstraits.

système d'exécution de la fabrication (MES)

Un système logiciel pour le suivi, la surveillance, la documentation et le contrôle des processus de production qui convertissent les matières premières en produits finis dans l'atelier.

MAP

Voir Migration Acceleration Program.

mécanisme

Processus complet au cours duquel vous créez un outil, favorisez son adoption, puis inspectez les résultats afin de procéder aux ajustements nécessaires. Un mécanisme est un cycle qui se renforce et s'améliore lorsqu'il fonctionne. Pour plus d'informations, voir Création de mécanismes dans le cadre AWS Well-Architected.

compte membre

Tous, à l' Comptes AWS exception du compte de gestion, qui font partie d'une organisation dans AWS Organizations. Un compte ne peut être membre que d'une seule organisation à la fois.

MAILLES

Voir le système d'exécution de la fabrication.

Transport télémétrique en file d'attente de messages (MQTT)

Protocole de communication léger machine-to-machine (M2M), basé sur le modèle de publication/d'abonnement, pour les appareils IoT aux ressources limitées.

microservice

Un petit service indépendant qui communique via un réseau bien défini APIs et qui est généralement détenu par de petites équipes autonomes. Par exemple, un système d'assurance peut inclure des microservices qui mappent à des capacités métier, telles que les ventes ou le marketing, ou à des sous-domaines, tels que les achats, les réclamations ou l'analytique. Les avantages des microservices incluent l'agilité, la flexibilité de la mise à l'échelle, la facilité de déploiement, la réutilisation du code et la résilience. Pour plus d'informations, consultez la section Intégration de microservices à l'aide de services AWS sans serveur.

architecture de microservices

Approche de création d'une application avec des composants indépendants qui exécutent chaque processus d'application en tant que microservice. Ces microservices communiquent via une interface bien définie en utilisant Lightweight. APIs Chaque microservice de cette architecture peut être mis à jour, déployé et mis à l'échelle pour répondre à la demande de fonctions spécifiques d'une application. Pour plus d'informations, consultez la section Implémentation de microservices sur AWS.

Programme d'accélération des migrations (MAP)

Un AWS programme qui fournit un support de conseil, des formations et des services pour aider les entreprises à établir une base opérationnelle solide pour passer au cloud, et pour aider à compenser le coût initial des migrations. MAP inclut une méthodologie de migration pour exécuter les migrations héritées de manière méthodique, ainsi qu'un ensemble d'outils pour automatiser et accélérer les scénarios de migration courants.

migration à grande échelle

Processus consistant à transférer la majeure partie du portefeuille d'applications vers le cloud par vagues, un plus grand nombre d'applications étant déplacées plus rapidement à chaque vague. Cette phase utilise les bonnes pratiques et les enseignements tirés des phases précédentes pour implémenter une usine de migration d'équipes, d'outils et de processus en vue de rationaliser la migration des charges de travail grâce à l'automatisation et à la livraison agile. Il s'agit de la troisième phase de la stratégie de migration AWS.

usine de migration

Équipes interfonctionnelles qui rationalisent la migration des charges de travail grâce à des approches automatisées et agiles. Les équipes de Migration Factory comprennent généralement des responsables des opérations, des analystes commerciaux et des propriétaires, des ingénieurs de migration, des développeurs et DevOps des professionnels travaillant dans le cadre de sprints.

Entre 20 et 50 % du portefeuille d'applications d'entreprise est constitué de modèles répétés qui peuvent être optimisés par une approche d'usine. Pour plus d'informations, veuillez consulter la rubrique discussion of migration factories et le guide Cloud Migration Factory dans cet ensemble de contenus.

métadonnées de migration

Informations relatives à l'application et au serveur nécessaires pour finaliser la migration. Chaque modèle de migration nécessite un ensemble de métadonnées de migration différent. Les exemples de métadonnées de migration incluent le sous-réseau cible, le groupe de sécurité et le AWS compte.

modèle de migration

Tâche de migration reproductible qui détaille la stratégie de migration, la destination de la migration et l'application ou le service de migration utilisé. Exemple : réorganisez la migration vers Amazon EC2 AWS avec le service de migration d'applications.

Évaluation du portefeuille de migration (MPA)

Outil en ligne qui fournit des informations pour valider l'analyse de rentabilisation en faveur de la migration vers le. AWS Cloud La MPA propose une évaluation détaillée du portefeuille (dimensionnement approprié des serveurs, tarification, comparaison du coût total de possession, analyse des coûts de migration), ainsi que la planification de la migration (analyse et collecte des données d'applications, regroupement des applications, priorisation des migrations et planification des vagues). L'outil MPA (connexion requise) est disponible gratuitement pour tous les AWS consultants et consultants APN Partner.

Évaluation de la préparation à la migration (MRA)

Processus qui consiste à obtenir des informations sur l'état de préparation d'une organisation au cloud, à identifier les forces et les faiblesses et à élaborer un plan d'action pour combler les lacunes identifiées, à l'aide du AWS CAF. Pour plus d'informations, veuillez consulter le guide de préparation à la migration. La MRA est la première phase de la stratégie de migration AWS.

stratégie de migration

L'approche utilisée pour migrer une charge de travail vers le AWS Cloud. Pour plus d'informations, reportez-vous aux 7 R de ce glossaire et à Mobiliser votre organisation pour accélérer les migrations à grande échelle.

ML

Voir apprentissage automatique.

modernisation

Transformation d'une application obsolète (héritée ou monolithique) et de son infrastructure en un système agile, élastique et hautement disponible dans le cloud afin de réduire les coûts, de gagner en efficacité et de tirer parti des innovations. Pour plus d'informations, consultez la section Stratégie de modernisation des applications dans le AWS Cloud.

évaluation de la préparation à la modernisation

Évaluation qui permet de déterminer si les applications d'une organisation sont prêtes à être modernisées, d'identifier les avantages, les risques et les dépendances, et qui détermine dans quelle mesure l'organisation peut prendre en charge l'état futur de ces applications. Le résultat de l'évaluation est un plan de l'architecture cible, une feuille de route détaillant les phases de développement et les étapes du processus de modernisation, ainsi qu'un plan d'action pour combler les lacunes identifiées. Pour plus d'informations, consultez la section Évaluation de l'état de préparation à la modernisation des applications dans le AWS Cloud.

applications monolithiques (monolithes)

Applications qui s'exécutent en tant que service unique avec des processus étroitement couplés. Les applications monolithiques ont plusieurs inconvénients. Si une fonctionnalité de l'application connaît un pic de demande, l'architecture entière doit être mise à l'échelle. L'ajout ou l'amélioration des fonctionnalités d'une application monolithique devient également plus complexe lorsque la base de code s'élargit. Pour résoudre ces problèmes, vous pouvez utiliser une architecture de microservices. Pour plus d'informations, veuillez consulter Decomposing monoliths into microservices.

MPA

Voir Évaluation du portefeuille de migration.

MQTT

Voir Message Queuing Telemetry Transport.

classification multi-classes

Processus qui permet de générer des prédictions pour plusieurs classes (prédiction d'un résultat parmi plus de deux). Par exemple, un modèle de ML peut demander « Ce produit est-il un livre, une voiture ou un téléphone ? » ou « Quelle catégorie de produits intéresse le plus ce client ? ».

infrastructure mutable

Modèle qui met à jour et modifie l'infrastructure existante pour les charges de travail de production. Pour améliorer la cohérence, la fiabilité et la prévisibilité, le AWS Well-Architected Framework recommande l'utilisation d'une infrastructure immuable comme meilleure pratique.

O

OAC

Voir Contrôle d'accès à l'origine.

OAI

Voir l'identité d'accès à l'origine.

OCM

Voir gestion du changement organisationnel.

migration hors ligne

Méthode de migration dans laquelle la charge de travail source est supprimée au cours du processus de migration. Cette méthode implique un temps d'arrêt prolongé et est généralement utilisée pour de petites charges de travail non critiques.

OI

Consultez la section Intégration des opérations.

OLA

Voir l'accord au niveau opérationnel.

migration en ligne

Méthode de migration dans laquelle la charge de travail source est copiée sur le système cible sans être mise hors ligne. Les applications connectées à la charge de travail peuvent continuer à fonctionner pendant la migration. Cette méthode implique un temps d'arrêt nul ou minimal et est généralement utilisée pour les charges de travail de production critiques.

OPC-UA

Voir Open Process Communications - Architecture unifiée.

Communications par processus ouvert - Architecture unifiée (OPC-UA)

Un protocole de communication machine-to-machine (M2M) pour l'automatisation industrielle. L'OPC-UA fournit une norme d'interopérabilité avec des schémas de cryptage, d'authentification et d'autorisation des données.

accord au niveau opérationnel (OLA)

Accord qui précise ce que les groupes informatiques fonctionnels s'engagent à fournir les uns aux autres, afin de prendre en charge un contrat de niveau de service (SLA).

examen de l'état de préparation opérationnelle (ORR)

Une liste de questions et de bonnes pratiques associées qui vous aident à comprendre, à évaluer, à prévenir ou à réduire l'ampleur des incidents et des défaillances possibles. Pour plus d'informations, voir Operational Readiness Reviews (ORR) dans le AWS Well-Architected Framework.

technologie opérationnelle (OT)

Systèmes matériels et logiciels qui fonctionnent avec l'environnement physique pour contrôler les opérations, les équipements et les infrastructures industriels. Dans le secteur manufacturier, l'intégration des systèmes OT et des technologies de l'information (IT) est au cœur des transformations de l'industrie 4.0.

intégration des opérations (OI)

Processus de modernisation des opérations dans le cloud, qui implique la planification de la préparation, l'automatisation et l'intégration. Pour en savoir plus, veuillez consulter le guide d'intégration des opérations.

journal de suivi d'organisation

Un parcours créé par AWS CloudTrail qui enregistre tous les événements pour tous les membres Comptes AWS d'une organisation dans AWS Organizations. Ce journal de suivi est créé dans chaque Compte AWS qui fait partie de l'organisation et suit l'activité de chaque compte. Pour plus d'informations, consultez la section Création d'un suivi pour une organisation dans la CloudTrail documentation.

gestion du changement organisationnel (OCM)

Cadre pour gérer les transformations métier majeures et perturbatrices du point de vue des personnes, de la culture et du leadership. L'OCM aide les organisations à se préparer et à effectuer la transition vers de nouveaux systèmes et de nouvelles politiques en accélérant

l'adoption des changements, en abordant les problèmes de transition et en favorisant des changements culturels et organisationnels. Dans la stratégie de AWS migration, ce cadre est appelé accélération du personnel, en raison de la rapidité du changement requise dans les projets d'adoption du cloud. Pour plus d'informations, veuillez consulter le guide OCM.

contrôle d'accès d'origine (OAC)

Dans CloudFront, une option améliorée pour restreindre l'accès afin de sécuriser votre contenu Amazon Simple Storage Service (Amazon S3). L'OAC prend en charge tous les compartiments S3 dans leur ensemble Régions AWS, le chiffrement côté serveur avec AWS KMS (SSE-KMS) et les requêtes dynamiques PUT adressées au compartiment S3. DELETE

identité d'accès d'origine (OAI)

Dans CloudFront, une option permettant de restreindre l'accès afin de sécuriser votre contenu Amazon S3. Lorsque vous utilisez OAI, il CloudFront crée un principal auprès duquel Amazon S3 peut s'authentifier. Les principaux authentifiés peuvent accéder au contenu d'un compartiment S3 uniquement via une distribution spécifique CloudFront . Voir également OAC, qui fournit un contrôle d'accès plus précis et amélioré.

ORR

Voir l'examen de l'état de préparation opérationnelle.

DE

Voir technologie opérationnelle.

VPC sortant (de sortie)

Dans une architecture AWS multi-comptes, un VPC qui gère les connexions réseau initiées depuis une application. L'architecture AWS de référence de sécurité recommande de configurer votre compte réseau avec les fonctions entrantes, sortantes et d'inspection VPCs afin de protéger l'interface bidirectionnelle entre votre application et l'Internet en général.

P

limite des autorisations

Politique de gestion IAM attachée aux principaux IAM pour définir les autorisations maximales que peut avoir l'utilisateur ou le rôle. Pour plus d'informations, veuillez consulter la rubrique Limites des autorisations dans la documentation IAM.

informations personnelles identifiables (PII)

Informations qui, lorsqu'elles sont consultées directement ou associées à d'autres données connexes, peuvent être utilisées pour déduire raisonnablement l'identité d'une personne. Les exemples d'informations personnelles incluent les noms, les adresses et les informations de contact.

PII

Voir les informations personnelles identifiables.

manuel stratégique

Ensemble d'étapes prédéfinies qui capturent le travail associé aux migrations, comme la fourniture de fonctions d'opérations de base dans le cloud. Un manuel stratégique peut revêtir la forme de scripts, de runbooks automatisés ou d'un résumé des processus ou des étapes nécessaires au fonctionnement de votre environnement modernisé.

PLC

Voir contrôleur logique programmable.

PLM

Consultez la section Gestion du cycle de vie des produits.

policy

Objet capable de définir les autorisations (voir la politique basée sur l'identité), de spécifier les conditions d'accès (voir la politique basée sur les ressources) ou de définir les autorisations maximales pour tous les comptes d'une organisation dans AWS Organizations (voir la politique de contrôle des services).

persistance polyglotte

Choix indépendant de la technologie de stockage de données d'un microservice en fonction des modèles d'accès aux données et d'autres exigences. Si vos microservices utilisent la même technologie de stockage de données, ils peuvent rencontrer des difficultés d'implémentation ou présenter des performances médiocres. Les microservices sont plus faciles à mettre en œuvre, atteignent de meilleures performances, ainsi qu'une meilleure capacité de mise à l'échelle s'ils utilisent l'entrepôt de données le mieux adapté à leurs besoins.

évaluation du portefeuille

Processus de découverte, d'analyse et de priorisation du portefeuille d'applications afin de planifier la migration. Pour plus d'informations, veuillez consulter Evaluating migration readiness.

predicate

Une condition de requête qui renvoie `true` ou`false`, généralement située dans une `WHERE` clause.

prédicat pushdown

Technique d'optimisation des requêtes de base de données qui filtre les données de la requête avant le transfert. Cela réduit la quantité de données qui doivent être extraites et traitées à partir de la base de données relationnelle et améliore les performances des requêtes.

contrôle préventif

Contrôle de sécurité conçu pour empêcher qu'un événement ne se produise. Ces contrôles constituent une première ligne de défense pour empêcher tout accès non autorisé ou toute modification indésirable de votre réseau. Pour plus d'informations, veuillez consulter Preventative controls dans Implementing security controls on AWS.

principal

Entité AWS capable d'effectuer des actions et d'accéder aux ressources. Cette entité est généralement un utilisateur root pour un Compte AWS rôle IAM ou un utilisateur. Pour plus d'informations, veuillez consulter la rubrique Principal dans Termes et concepts relatifs aux rôles, dans la documentation IAM.

confidentialité dès la conception

Une approche d'ingénierie système qui prend en compte la confidentialité tout au long du processus de développement.

zones hébergées privées

Conteneur contenant des informations sur la manière dont vous souhaitez qu'Amazon Route 53 réponde aux requêtes DNS pour un domaine et ses sous-domaines au sein d'un ou de plusieurs VPCs domaines. Pour plus d'informations, veuillez consulter Working with private hosted zones dans la documentation Route 53.

contrôle proactif

Contrôle de sécurité conçu pour empêcher le déploiement de ressources non conformes. Ces contrôles analysent les ressources avant qu'elles ne soient provisionnées. Si la ressource n'est pas conforme au contrôle, elle n'est pas provisionnée. Pour plus d'informations, consultez le guide de référence sur les contrôles dans la AWS Control Tower documentation et consultez la section Contrôles proactifs dans Implémentation des contrôles de sécurité sur AWS.

gestion du cycle de vie des produits (PLM)

Gestion des données et des processus d'un produit tout au long de son cycle de vie, depuis la conception, le développement et le lancement, en passant par la croissance et la maturité, jusqu'au déclin et au retrait.

environnement de production

Voir environnement.

contrôleur logique programmable (PLC)

Dans le secteur manufacturier, un ordinateur hautement fiable et adaptable qui surveille les machines et automatise les processus de fabrication.

chaînage rapide

Utiliser le résultat d'une invite LLM comme entrée pour l'invite suivante afin de générer de meilleures réponses. Cette technique est utilisée pour décomposer une tâche complexe en sous-tâches ou pour affiner ou développer de manière itérative une réponse préliminaire. Cela permet d'améliorer la précision et la pertinence des réponses d'un modèle et permet d'obtenir des résultats plus précis et personnalisés.

pseudonymisation

Processus de remplacement des identifiants personnels dans un ensemble de données par des valeurs fictives. La pseudonymisation peut contribuer à protéger la vie privée. Les données pseudonymisées sont toujours considérées comme des données personnelles.

publish/subscribe (pub/sub)

Modèle qui permet des communications asynchrones entre les microservices afin d'améliorer l'évolutivité et la réactivité. Par exemple, dans un MES basé sur des microservices, un microservice peut publier des messages d'événements sur un canal auquel d'autres microservices peuvent s'abonner. Le système peut ajouter de nouveaux microservices sans modifier le service de publication.

Q

plan de requête

Série d'étapes, telles que des instructions, utilisées pour accéder aux données d'un système de base de données relationnelle SQL.

régression du plan de requêtes

Le cas où un optimiseur de service de base de données choisit un plan moins optimal qu'avant une modification donnée de l'environnement de base de données. Cela peut être dû à des changements en termes de statistiques, de contraintes, de paramètres d'environnement, de liaisons de paramètres de requêtes et de mises à jour du moteur de base de données.

R

Matrice RACI

Voir responsable, responsable, consulté, informé (RACI).

RAG

Voir Retrieval Augmented Generation.

rançongiciel

Logiciel malveillant conçu pour bloquer l'accès à un système informatique ou à des données jusqu'à ce qu'un paiement soit effectué.

Matrice RASCI

Voir responsable, responsable, consulté, informé (RACI).

RCAC

Voir contrôle d'accès aux lignes et aux colonnes.

réplica en lecture

Copie d'une base de données utilisée en lecture seule. Vous pouvez acheminer les requêtes vers le réplica de lecture pour réduire la charge sur votre base de données principale.

réarchitecte

Voir 7 Rs.

objectif de point de récupération (RPO)

Durée maximale acceptable depuis le dernier point de récupération des données. Il détermine ce qui est considéré comme étant une perte de données acceptable entre le dernier point de reprise et l’interruption du service.

objectif de temps de récupération (RTO)

Le délai maximum acceptable entre l'interruption du service et le rétablissement du service.

refactoriser

Voir 7 Rs.

Région

Un ensemble de AWS ressources dans une zone géographique. Chacun Région AWS est isolé et indépendant des autres pour garantir tolérance aux pannes, stabilité et résilience. Pour plus d'informations, voir Spécifier ce que Régions AWS votre compte peut utiliser.

régression

Technique de ML qui prédit une valeur numérique. Par exemple, pour résoudre le problème « Quel sera le prix de vente de cette maison ? », un modèle de ML pourrait utiliser un modèle de régression linéaire pour prédire le prix de vente d'une maison sur la base de faits connus à son sujet (par exemple, la superficie en mètres carrés).

réhéberger

Voir 7 Rs.

version

Dans un processus de déploiement, action visant à promouvoir les modifications apportées à un environnement de production.

déplacer

Voir 7 Rs.

replateforme

Voir 7 Rs.

rachat

Voir 7 Rs.

résilience

La capacité d'une application à résister aux perturbations ou à s'en remettre. La haute disponibilité et la reprise après sinistre sont des considérations courantes lors de la planification de la résilience dans le AWS Cloud. Pour plus d'informations, consultez AWS Cloud Résilience.

politique basée sur les ressources

Politique attachée à une ressource, comme un compartiment Amazon S3, un point de terminaison ou une clé de chiffrement. Ce type de politique précise les principaux auxquels l'accès est autorisé, les actions prises en charge et toutes les autres conditions qui doivent être remplies.

matrice responsable, redevable, consulté et informé (RACI)

Une matrice qui définit les rôles et les responsabilités de toutes les parties impliquées dans les activités de migration et les opérations cloud. Le nom de la matrice est dérivé des types de responsabilité définis dans la matrice : responsable (R), responsable (A), consulté (C) et informé (I). Le type de support (S) est facultatif. Si vous incluez le support, la matrice est appelée matrice RASCI, et si vous l'excluez, elle est appelée matrice RACI.

contrôle réactif

Contrôle de sécurité conçu pour permettre de remédier aux événements indésirables ou aux écarts par rapport à votre référence de sécurité. Pour plus d'informations, veuillez consulter la rubrique Responsive controls dans Implementing security controls on AWS.

retain

Voir 7 Rs.

se retirer

Voir 7 Rs.

Génération augmentée de récupération (RAG)

Technologie d'IA générative dans laquelle un LLM fait référence à une source de données faisant autorité qui se trouve en dehors de ses sources de données de formation avant de générer une réponse. Par exemple, un modèle RAG peut effectuer une recherche sémantique dans la base de connaissances ou dans les données personnalisées d'une organisation. Pour plus d'informations, voir Qu'est-ce que RAG ?

rotation

Processus de mise à jour périodique d'un secret pour empêcher un attaquant d'accéder aux informations d'identification.

contrôle d'accès aux lignes et aux colonnes (RCAC)

Utilisation d'expressions SQL simples et flexibles dotées de règles d'accès définies. Le RCAC comprend des autorisations de ligne et des masques de colonnes.

RPO

Voir l'objectif du point de récupération.

RTO

Voir l'objectif en matière de temps de rétablissement.

runbook

Ensemble de procédures manuelles ou automatisées nécessaires à l'exécution d'une tâche spécifique. Elles visent généralement à rationaliser les opérations ou les procédures répétitives présentant des taux d'erreur élevés.

S

SAML 2.0

Un standard ouvert utilisé par de nombreux fournisseurs d'identité (IdPs). Cette fonctionnalité permet l'authentification unique fédérée (SSO), afin que les utilisateurs puissent se connecter AWS Management Console ou appeler les opérations de l' AWS API sans que vous ayez à créer un utilisateur dans IAM pour tous les membres de votre organisation. Pour plus d'informations sur la fédération SAML 2.0, veuillez consulter À propos de la fédération SAML 2.0 dans la documentation IAM.

SCADA

Voir Contrôle de supervision et acquisition de données.

SCP

Voir la politique de contrôle des services.

secret

Dans AWS Secrets Manager des informations confidentielles ou restreintes, telles qu'un mot de passe ou des informations d'identification utilisateur, que vous stockez sous forme cryptée. Il comprend la valeur secrète et ses métadonnées. La valeur secrète peut être binaire, une chaîne unique ou plusieurs chaînes. Pour plus d'informations, voir Que contient le secret d'un Secrets Manager ? dans la documentation de Secrets Manager.

sécurité dès la conception

Une approche d'ingénierie système qui prend en compte la sécurité tout au long du processus de développement.

contrôle de sécurité

Barrière de protection technique ou administrative qui empêche, détecte ou réduit la capacité d'un assaillant d'exploiter une vulnérabilité de sécurité. Il existe quatre principaux types de contrôles de sécurité : préventifs, détectifs, réactifs et proactifs.

renforcement de la sécurité

Processus qui consiste à réduire la surface d'attaque pour la rendre plus résistante aux attaques. Cela peut inclure des actions telles que la suppression de ressources qui ne sont plus requises, la mise en œuvre des bonnes pratiques de sécurité consistant à accorder le moindre privilège ou la désactivation de fonctionnalités inutiles dans les fichiers de configuration.

système de gestion des informations et des événements de sécurité (SIEM)

Outils et services qui associent les systèmes de gestion des informations de sécurité (SIM) et de gestion des événements de sécurité (SEM). Un système SIEM collecte, surveille et analyse les données provenant de serveurs, de réseaux, d'appareils et d'autres sources afin de détecter les menaces et les failles de sécurité, mais aussi de générer des alertes.

automatisation des réponses de sécurité

Action prédéfinie et programmée conçue pour répondre automatiquement à un événement de sécurité ou y remédier. Ces automatisations servent de contrôles de sécurité détectifs ou réactifs qui vous aident à mettre en œuvre les meilleures pratiques en matière AWS de sécurité. Parmi les actions de réponse automatique, citons la modification d'un groupe de sécurité VPC, l'application de correctifs à une instance Amazon EC2 ou la rotation des informations d'identification.

chiffrement côté serveur

Chiffrement des données à destination, par celui Service AWS qui les reçoit.

Politique de contrôle des services (SCP)

Politique qui fournit un contrôle centralisé des autorisations pour tous les comptes d'une organisation dans AWS Organizations. SCPs définissez des garde-fous ou des limites aux actions qu'un administrateur peut déléguer à des utilisateurs ou à des rôles. Vous pouvez les utiliser SCPs comme listes d'autorisation ou de refus pour spécifier les services ou les actions autorisés ou interdits. Pour plus d'informations, consultez la section Politiques de contrôle des services dans la AWS Organizations documentation.

point de terminaison du service

URL du point d'entrée pour un Service AWS. Pour vous connecter par programmation au service cible, vous pouvez utiliser un point de terminaison. Pour plus d'informations, veuillez consulter la rubrique Service AWS endpoints dans Références générales AWS.

contrat de niveau de service (SLA)

Accord qui précise ce qu'une équipe informatique promet de fournir à ses clients, comme le temps de disponibilité et les performances des services.

indicateur de niveau de service (SLI)

Mesure d'un aspect des performances d'un service, tel que son taux d'erreur, sa disponibilité ou son débit.

objectif de niveau de service (SLO)

Mesure cible qui représente l'état d'un service, tel que mesuré par un indicateur de niveau de service.

modèle de responsabilité partagée

Un modèle décrivant la responsabilité que vous partagez en matière AWS de sécurité et de conformité dans le cloud. AWS est responsable de la sécurité du cloud, alors que vous êtes responsable de la sécurité dans le cloud. Pour de plus amples informations, veuillez consulter Modèle de responsabilité partagée.

SIEM

Consultez les informations de sécurité et le système de gestion des événements.

point de défaillance unique (SPOF)

Défaillance d'un seul composant critique d'une application susceptible de perturber le système.

SLA

Voir le contrat de niveau de service.

SLI

Voir l'indicateur de niveau de service.

SLO

Voir l'objectif de niveau de service.

split-and-seed modèle

Modèle permettant de mettre à l'échelle et d'accélérer les projets de modernisation. Au fur et à mesure que les nouvelles fonctionnalités et les nouvelles versions de produits sont définies, l'équipe principale se divise pour créer des équipes de produit. Cela permet de mettre à l'échelle les capacités et les services de votre organisation, d'améliorer la productivité des développeurs et de favoriser une innovation rapide. Pour plus d'informations, voir Approche progressive de la modernisation des applications dans le. AWS Cloud

SPOF

Voir point de défaillance unique.

schéma en étoile

Structure organisationnelle de base de données qui utilise une grande table de faits pour stocker les données transactionnelles ou mesurées et utilise une ou plusieurs tables dimensionnelles plus petites pour stocker les attributs des données. Cette structure est conçue pour être utilisée dans un entrepôt de données ou à des fins de business intelligence.

modèle de figuier étrangleur

Approche de modernisation des systèmes monolithiques en réécrivant et en remplaçant progressivement les fonctionnalités du système jusqu'à ce que le système hérité puisse être mis hors service. Ce modèle utilise l'analogie d'un figuier de vigne qui se développe dans un arbre existant et qui finit par supplanter son hôte. Le schéma a été présenté par Martin Fowler comme un moyen de gérer les risques lors de la réécriture de systèmes monolithiques. Pour obtenir un exemple d'application de ce modèle, veuillez consulter Modernizing legacy Microsoft ASP.NET (ASMX) web services incrementally by using containers and Amazon API Gateway.

sous-réseau

Plage d'adresses IP dans votre VPC. Un sous-réseau doit se trouver dans une seule zone de disponibilité.

contrôle de supervision et acquisition de données (SCADA)

Dans le secteur manufacturier, un système qui utilise du matériel et des logiciels pour surveiller les actifs physiques et les opérations de production.

chiffrement symétrique

Algorithme de chiffrement qui utilise la même clé pour chiffrer et déchiffrer les données.

tests synthétiques

Tester un système de manière à simuler les interactions des utilisateurs afin de détecter les problèmes potentiels ou de surveiller les performances. Vous pouvez utiliser Amazon CloudWatch Synthetics pour créer ces tests.

invite du système

Technique permettant de fournir un contexte, des instructions ou des directives à un LLM afin d'orienter son comportement. Les instructions du système aident à définir le contexte et à établir des règles pour les interactions avec les utilisateurs.

T

tags

Des paires clé-valeur qui agissent comme des métadonnées pour organiser vos AWS ressources. Les balises peuvent vous aider à gérer, identifier, organiser, rechercher et filtrer des ressources. Pour plus d'informations, veuillez consulter la rubrique Balisage de vos AWS ressources.

variable cible

La valeur que vous essayez de prédire dans le cadre du ML supervisé. Elle est également qualifiée de variable de résultat. Par exemple, dans un environnement de fabrication, la variable cible peut être un défaut du produit.

liste de tâches

Outil utilisé pour suivre les progrès dans un runbook. Liste de tâches qui contient une vue d'ensemble du runbook et une liste des tâches générales à effectuer. Pour chaque tâche générale, elle inclut le temps estimé nécessaire, le propriétaire et l'avancement.

environnement de test

Voir environnement.

entraînement

Pour fournir des données à partir desquelles votre modèle de ML peut apprendre. Les données d'entraînement doivent contenir la bonne réponse. L'algorithme d'apprentissage identifie des modèles dans les données d'entraînement, qui mettent en correspondance les attributs des données d'entrée avec la cible (la réponse que vous souhaitez prédire). Il fournit un modèle de ML

qui capture ces modèles. Vous pouvez alors utiliser le modèle de ML pour obtenir des prédictions sur de nouvelles données pour lesquelles vous ne connaissez pas la cible.

passerelle de transit

Un hub de transit réseau que vous pouvez utiliser pour interconnecter vos réseaux VPCs et ceux sur site. Pour plus d'informations, voir Qu'est-ce qu'une passerelle de transit dans la AWS Transit Gateway documentation.

flux de travail basé sur jonction

Approche selon laquelle les développeurs génèrent et testent des fonctionnalités localement dans une branche de fonctionnalités, puis fusionnent ces modifications dans la branche principale. La branche principale est ensuite intégrée aux environnements de développement, de préproduction et de production, de manière séquentielle.

accès sécurisé

Accorder des autorisations à un service que vous spécifiez pour effectuer des tâches au sein de votre organisation AWS Organizations et dans ses comptes en votre nom. Le service de confiance crée un rôle lié au service dans chaque compte, lorsque ce rôle est nécessaire, pour effectuer des tâches de gestion à votre place. Pour plus d'informations, consultez la section Utilisation AWS Organizations avec d'autres AWS services dans la AWS Organizations documentation.

réglage

Pour modifier certains aspects de votre processus d'entraînement afin d'améliorer la précision du modèle de ML. Par exemple, vous pouvez entraîner le modèle de ML en générant un ensemble d'étiquetage, en ajoutant des étiquettes, puis en répétant ces étapes plusieurs fois avec différents paramètres pour optimiser le modèle.

équipe de deux pizzas

Une petite DevOps équipe que vous pouvez nourrir avec deux pizzas. Une équipe de deux pizzas garantit les meilleures opportunités de collaboration possible dans le développement de logiciels.

U

incertitude

Un concept qui fait référence à des informations imprécises, incomplètes ou inconnues susceptibles de compromettre la fiabilité des modèles de ML prédictifs. Il existe deux types

d'incertitude : l'incertitude épistémique est causée par des données limitées et incomplètes, alors que l'incertitude aléatoire est causée par le bruit et le caractère aléatoire inhérents aux données. Pour plus d'informations, veuillez consulter le guide Quantifying uncertainty in deep learning systems.

tâches indifférenciées

Également connu sous le nom de « levage de charges lourdes », ce travail est nécessaire pour créer et exploiter une application, mais qui n'apporte pas de valeur directe à l'utilisateur final ni d'avantage concurrentiel. Les exemples de tâches indifférenciées incluent l'approvisionnement, la maintenance et la planification des capacités.

environnements supérieurs

Voir environnement.

V

mise à vide

Opération de maintenance de base de données qui implique un nettoyage après des mises à jour incrémentielles afin de récupérer de l'espace de stockage et d'améliorer les performances.

contrôle de version

Processus et outils permettant de suivre les modifications, telles que les modifications apportées au code source dans un référentiel.

Appairage de VPC

Une connexion entre deux VPCs qui vous permet d'acheminer le trafic en utilisant des adresses IP privées. Pour plus d'informations, veuillez consulter la rubrique Qu'est-ce que l'appairage de VPC ? dans la documentation Amazon VPC.

vulnérabilités

Défaut logiciel ou matériel qui compromet la sécurité du système.

W

cache actif

Cache tampon qui contient les données actuelles et pertinentes fréquemment consultées. L'instance de base de données peut lire à partir du cache tampon, ce qui est plus rapide que la lecture à partir de la mémoire principale ou du disque.

données chaudes

Données rarement consultées. Lorsque vous interrogez ce type de données, des requêtes modérément lentes sont généralement acceptables.

fonction de fenêtre

Fonction SQL qui effectue un calcul sur un groupe de lignes liées d'une manière ou d'une autre à l'enregistrement en cours. Les fonctions de fenêtre sont utiles pour traiter des tâches, telles que le calcul d'une moyenne mobile ou l'accès à la valeur des lignes en fonction de la position relative de la ligne en cours.

charge de travail

Ensemble de ressources et de code qui fournit une valeur métier, par exemple une application destinée au client ou un processus de backend.

flux de travail

Groupes fonctionnels d'un projet de migration chargés d'un ensemble de tâches spécifique. Chaque flux de travail est indépendant, mais prend en charge les autres flux de travail du projet. Par exemple, le flux de travail du portefeuille est chargé de prioriser les applications, de planifier les vagues et de collecter les métadonnées de migration. Le flux de travail du portefeuille fournit ces actifs au flux de travail de migration, qui migre ensuite les serveurs et les applications.

VER

Voir écrire une fois, lire plusieurs.

WQF

Voir le cadre AWS de qualification de la charge de travail.

écrire une fois, lire plusieurs (WORM)

Modèle de stockage qui écrit les données une seule fois et empêche leur suppression ou leur modification. Les utilisateurs autorisés peuvent lire les données autant de fois que nécessaire,

mais ils ne peuvent pas les modifier. Cette infrastructure de stockage de données est considérée comme immuable.

Z

exploit Zero-Day

Une attaque, généralement un logiciel malveillant, qui tire parti d'une vulnérabilité de type « jour zéro ».

vulnérabilité « jour zéro »

Une faille ou une vulnérabilité non atténuée dans un système de production. Les acteurs malveillants peuvent utiliser ce type de vulnérabilité pour attaquer le système. Les développeurs prennent souvent conscience de la vulnérabilité à la suite de l'attaque.

invite Zero-Shot

Fournir à un LLM des instructions pour effectuer une tâche, mais aucun exemple (plans) pouvant aider à la guider. Le LLM doit utiliser ses connaissances pré-entraînées pour gérer la tâche. L'efficacité de l'invite zéro dépend de la complexité de la tâche et de la qualité de l'invite. Voir également les instructions en quelques clics.

application zombie

Application dont l'utilisation moyenne du processeur et de la mémoire est inférieure à 5 %. Dans un projet de migration, il est courant de retirer ces applications.

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.