Autorización de SaaS para múltiples inquilinos y control de acceso a la API: opciones de implementación y mejores prácticas

AWS Guía prescriptiva

AWS Guía prescriptiva: Autorización de SaaS para múltiples inquilinos y control de acceso a la API: opciones de implementación y mejores prácticas

Table of Contents

Autorización de SaaS para múltiples inquilinos y control de acceso a la API: opciones de implementación y mejores prácticas

Tabby Ward, Thomas Davis, Gideon Landeman y Tomas Riha, Amazon Web Services ()AWS

Mayo de 2024 (historia del documento)

La autorización y el control de acceso a las API representan un desafío para muchas aplicaciones de software, en particular, para las aplicaciones de software como servicio (SaaS) multiusuario. Esta complejidad es evidente si se tiene en cuenta la proliferación de microservicios APIs que deben protegerse y la gran cantidad de condiciones de acceso que se derivan de los distintos tipos de arrendatarios, características de los usuarios y estados de las aplicaciones. Para abordar estos problemas de manera efectiva, una solución debe imponer el control de acceso en los numerosos componentes que APIs presentan los microservicios, las capas de backend para frontend (BFF) y otros componentes de una aplicación SaaS multiusuario. Este enfoque debe ir acompañado de un mecanismo que sea capaz de tomar decisiones de acceso complejas en función de muchos factores y atributos.

Tradicionalmente, el control de acceso y la autorización de las API se gestionaban mediante una lógica personalizada en el código de la aplicación. Este enfoque era propenso a errores y no era seguro, ya que los desarrolladores que tenían acceso a este código podían cambiar la lógica de autorización de forma accidental o deliberada, lo que podía provocar un acceso no autorizado. Era difícil auditar las decisiones adoptadas mediante la lógica personalizada en el código de la aplicación, ya que los auditores tenían que sumergirse en la lógica personalizada para determinar su eficacia a la hora de mantener un estándar concreto. Además, en general, el control de acceso a las API era innecesario, ya que no había tantas APIs que proteger. El cambio de paradigma en el diseño de aplicaciones para favorecer los microservicios y las arquitecturas orientadas a servicios ha incrementado el número de empresas APIs que deben utilizar una forma de autorización y control de acceso. Además, la necesidad de mantener el acceso basado en el inquilino en una aplicación de SaaS multiinquilino presenta desafíos de autorización adicionales para preservar el arrendamiento. Las mejores prácticas descritas en esta guía ofrecen varios beneficios:

- La lógica de autorización se puede centralizar y escribir en un lenguaje declarativo de alto nivel que no es específico de ningún lenguaje de programación.

- La lógica de autorización se extrae del código de la aplicación y se puede aplicar como un patrón repetible a todos los APIs componentes de una aplicación.
- La abstracción evita que los desarrolladores realicen cambios accidentales en la lógica de autorización.
- La integración en una aplicación SaaS es coherente y sencilla.
- La abstracción evita la necesidad de escribir una lógica de autorización personalizada para cada punto final de la API.
- Las auditorías se simplifican porque el auditor ya no necesita revisar el código para determinar los permisos.
- El enfoque descrito en esta guía admite el uso de varios paradigmas de control de acceso en función de los requisitos de la organización.
- Este enfoque de autorización y control de acceso proporciona una forma sencilla y directa de mantener el aislamiento de los datos de los inquilinos en la capa de API de una aplicación SaaS.
- Las mejores prácticas proporcionan un enfoque coherente a la hora de incorporar y retirar inquilinos en lo que respecta a la autorización.
- Este enfoque ofrece diferentes modelos de despliegue de autorizaciones (agrupados o aislados), que presentan ventajas y desventajas, como se explica en esta guía.

Resultados empresariales específicos

Esta guía prescriptiva describe los patrones de diseño repetibles para los controles de autorización y acceso a las API que se pueden implementar para aplicaciones SaaS multiusuario. Esta guía está destinada a cualquier equipo que desarrolle aplicaciones con requisitos de autorización complejos o necesidades estrictas de control de acceso a la API. La arquitectura detalla la creación de un punto de decisión de políticas (PDP) o un motor de políticas y la integración de los puntos de aplicación de políticas (PEP) en ellos. APIs Se discuten dos opciones específicas para crear un PDP: usar Amazon Verified Permissions con el SDK de Cedar y usar Open Policy Agent (OPA) con el lenguaje de políticas Rego. La guía también analiza la toma de decisiones de acceso basadas en un modelo de control de acceso basado en atributos (ABAC) o en un modelo de control de acceso basado en roles (RBAC), o en una combinación de ambos modelos. Le recomendamos que utilice los patrones y conceptos de diseño que se proporcionan en esta guía para informar y estandarizar la implementación de la autorización y el control de acceso a las API en aplicaciones SaaS multiusuario. Esta guía ayuda a lograr los siguientes resultados empresariales:

- Arquitectura de autorización de API estandarizada para aplicaciones SaaS multiusuario: esta arquitectura distingue entre tres componentes: el punto de administración de políticas (PAP) donde se almacenan y administran las políticas, el punto de decisión de políticas (PDP) donde se evalúan esas políticas para tomar una decisión de autorización y el punto de aplicación de políticas (PEP) que hace cumplir esa decisión. El servicio de autorización hospedado, Verified Permissions, funciona como PAP y como PDP. Como alternativa, puede crear su PDP usted mismo utilizando un motor de código abierto como Cedar u OPA.
- Separación de la lógica de autorización de las aplicaciones: la lógica de autorización, cuando está integrada en el código de la aplicación o se implementa mediante un mecanismo de aplicación ad hoc, puede estar sujeta a cambios accidentales o malintencionados que provocan el acceso involuntario a los datos entre usuarios u otras infracciones de seguridad. Para ayudar a mitigar estas posibilidades, puede utilizar un PAP, como Verified Permissions, para almacenar las políticas de autorización independientemente del código de la aplicación y aplicar una gobernanza sólida a la gestión de esas políticas. Las políticas se pueden mantener de forma centralizada en un lenguaje declarativo de alto nivel, lo que hace que mantener la lógica de autorización sea mucho más sencillo que cuando se integran políticas en varias secciones del código de la aplicación. Este enfoque también garantiza que las actualizaciones se apliquen de forma coherente.
- Enfoque flexible de los modelos de control de acceso: el control de acceso basado en roles (RBAC), el control de acceso basado en atributos (ABAC) o una combinación de ambos modelos son todos enfoques válidos para el control de acceso. Estos modelos intentan cumplir con los requisitos de autorización de una empresa mediante el uso de diferentes enfoques. En esta guía se comparan y contrastan estos modelos para ayudarle a seleccionar el modelo que mejor se adapte a su organización. La guía también analiza cómo se aplican estos modelos a diferentes lenguajes de políticas de autorización, como OPA/Rego el Cedar. Las arquitecturas analizadas en esta guía permiten adoptar uno o ambos modelos con éxito.
- Control estricto del acceso a la API: esta guía proporciona un método para proteger una aplicación de forma APIs uniforme y generalizada con un mínimo esfuerzo. Esto resulta especialmente útil para las arquitecturas de aplicaciones orientadas a servicios o microservicios que, por lo general, utilizan un gran número de ellas para facilitar las comunicaciones entre aplicaciones. APIs El estricto control de acceso a la API ayuda a aumentar la seguridad de una aplicación y la hace menos vulnerable a los ataques o la explotación.

Aislamiento de inquilinos y autorización de múltiples inquilinos

Esta guía se refiere a los conceptos de aislamiento de inquilinos y autorización de inquilinos múltiples. El aislamiento de inquilinos se refiere a los mecanismos explícitos que se utilizan en un sistema SaaS para garantizar que los recursos de cada inquilino, incluso cuando operan en una infraestructura compartida, estén aislados. La autorización de varios inquilinos se refiere a autorizar las acciones entrantes y evitar que se implementen en el arrendatario equivocado. Un usuario hipotético podría estar autenticado y autorizado y, aun así, acceder a los recursos de otro inquilino. La autenticación y la autorización no bloquearán este acceso; es necesario implementar el aislamiento de los inquilinos para lograr este objetivo. Para obtener una explicación más amplia de las diferencias entre estos dos conceptos, consulte la sección sobre el aislamiento de inquilinos del documento técnico sobre los fundamentos de la arquitectura de SaaS.

Tipos de control de acceso

Puede utilizar dos modelos ampliamente definidos para implementar el control de acceso: el control de acceso basado en roles (RBAC) y el control de acceso basado en atributos (ABAC). Cada modelo tiene ventajas y desventajas, que se analizan brevemente en esta sección. El modelo que debe utilizar depende de su caso de uso específico. La arquitectura que se describe en esta guía es compatible con ambos modelos.

RBAC

El control de acceso basado en roles (RBAC) determina el acceso a los recursos en función de un rol que normalmente se alinea con la lógica empresarial. Los permisos se asocian al rol según corresponda. Por ejemplo, una función de marketing autorizaría a un usuario a realizar actividades de marketing dentro de un sistema restringido. Se trata de un modelo de control de acceso relativamente sencillo de implementar porque se ajusta bien a una lógica empresarial fácilmente reconocible.

El modelo RBAC es menos eficaz cuando:

- Tiene usuarios únicos cuyas responsabilidades abarcan varias funciones.
- Tiene una lógica empresarial compleja que dificulta la definición de las funciones.
- La ampliación hasta alcanzar un tamaño grande requiere una administración y una asignación constantes de los permisos a las funciones nuevas y existentes.
- Las autorizaciones se basan en parámetros dinámicos.

ABAC

El control de acceso basado en atributos (ABAC) determina el acceso a los recursos en función de los atributos. Los atributos se pueden asociar a un usuario, un recurso, un entorno o incluso al estado de la aplicación. Sus políticas o reglas hacen referencia a los atributos y pueden usar la lógica booleana básica para determinar si un usuario tiene permiso para realizar una acción. Este es un ejemplo básico de permisos:

En el sistema de pagos, todos los usuarios del departamento de finanzas pueden procesar los pagos en el punto final de la API `/payments` durante el horario laboral.

La pertenencia al departamento de finanzas es un atributo del usuario que determina el acceso al `/payments`. También hay un atributo de recurso asociado al punto final de la `/payments` API que permite el acceso solo durante el horario laboral. En ABAC, el hecho de que un usuario pueda o no procesar un pago viene determinado por una política que incluye la pertenencia al departamento de finanzas como atributo de usuario y la hora como atributo de recurso de`/payments`.

El modelo ABAC es muy flexible y permite tomar decisiones de autorización dinámicas, contextuales y granulares. Sin embargo, el modelo ABAC es difícil de implementar inicialmente. La definición de reglas y políticas, así como la enumeración de los atributos de todos los vectores de acceso relevantes, requieren una importante inversión inicial para su implementación.

Enfoque híbrido RBAC-ABAC

La combinación de RBAC y ABAC puede proporcionar algunas de las ventajas de ambos modelos. El RBAC, al estar tan alineado con la lógica empresarial, es más fácil de implementar que el ABAC. Para proporcionar un nivel adicional de granularidad a la hora de tomar decisiones de autorización, puede combinar el ABAC con el RBAC. Este enfoque híbrido determina el acceso combinando el rol de un usuario (y sus permisos asignados) con atributos adicionales para tomar decisiones de acceso. El uso de ambos modelos permite una administración y asignación de permisos sencillas y, al mismo tiempo, permite una mayor flexibilidad y granularidad en lo que respecta a las decisiones de autorización.

Comparación de modelos de control de acceso

La siguiente tabla compara los tres modelos de control de acceso analizados anteriormente. Esta comparación pretende ser informativa y de alto nivel. El uso de un modelo de acceso en una situación específica puede no estar necesariamente correlacionado con las comparaciones realizadas en esta tabla.

Factor	RBAC	ABAC	Híbrido
Flexibilidad	Medio	Alto	Alto
Simplicidad	Alto	Bajo	Medio
Granularity (Grado de detalle)	Bajo	Alto	Medio

Decisiones y reglas dinámicas	No	Sí	Sí
Conscientes del contexto	No	Sí	Un poco
Esfuerzo de implementación	Bajo	Alto	Medio

Implementación de un PDP

El punto de decisión política (PDP) se puede caracterizar como un motor de políticas o reglas. Este componente es responsable de aplicar las políticas o reglas y de decidir si se permite un acceso en particular. Un PDP puede funcionar con modelos de control de acceso basado en roles (RBAC) y de control de acceso basado en atributos (ABAC); sin embargo, un PDP es un requisito para el ABAC. Un PDP permite transferir la lógica de autorización del código de la aplicación a un sistema independiente. Esto puede simplificar el código de la aplicación. También proporciona una interfaz easy-to-use repetible para tomar decisiones de autorización para microservicios APIs, capas de backend for Frontend (BFF) o cualquier otro componente de la aplicación.

En las siguientes secciones, se analizan tres métodos para implementar un PDP. Sin embargo, esta no es una lista completa.

Métodos de implementación del PDP:

- Implementación de un PDP mediante permisos verificados de Amazon
- Implementación de un PDP mediante OPA
- Uso de un motor de políticas personalizado

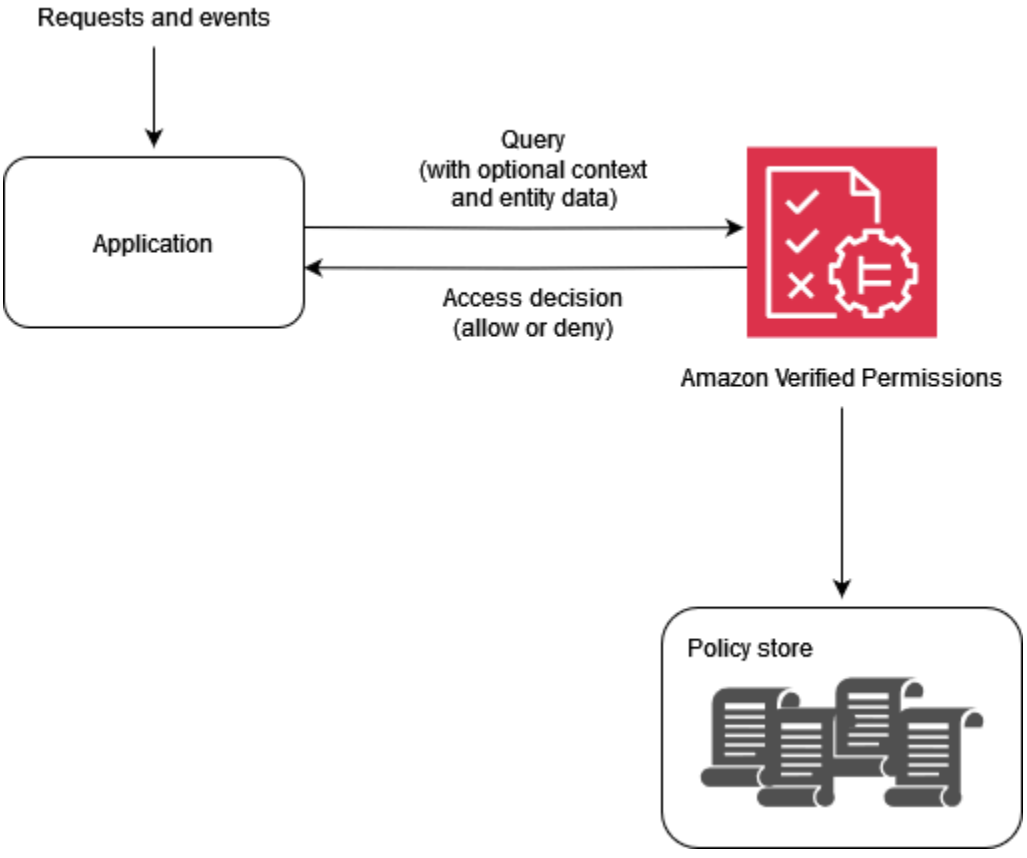
Implementación de un PDP mediante permisos verificados de Amazon

Amazon Verified Permissions es un servicio de autorización y administración de permisos escalable y detallado que puede utilizar para implementar un punto de decisión política (PDP). Como motor de políticas, puede ayudar a su aplicación a verificar las acciones de los usuarios en tiempo real y a destacar los permisos excesivamente privilegiados o no válidos. Ayuda a sus desarrolladores a crear aplicaciones más seguras con mayor rapidez al externalizar la autorización y centralizar la gestión y la administración de las políticas. Al separar la lógica de autorización de la lógica de la aplicación, Verified Permissions permite desvincular las políticas.

Al utilizar los permisos verificados para implementar un PDP y al implementar la verificación continua y con privilegios mínimos en las aplicaciones, los desarrolladores pueden ajustar el acceso a las aplicaciones a los principios de confianza cero. Además, los equipos de seguridad y auditoría pueden analizar y auditar mejor quién tiene acceso a qué recursos de una aplicación. Verified Permissions utiliza Cedar, un lenguaje de políticas de código abierto diseñado específicamente y que

prioriza la seguridad, para definir los controles de acceso basados en políticas basados en el control de acceso basado en roles (RBAC) y el control de acceso basado en atributos (ABAC) para lograr un control de acceso más detallado y sensible al contexto.

Los permisos verificados proporcionan algunas funciones útiles para las aplicaciones SaaS, como la capacidad de habilitar la autorización de varios inquilinos mediante el uso de varios proveedores de identidad, como Amazon Cognito, Google y Facebook. Otra función de permisos verificados que resulta especialmente útil para las aplicaciones SaaS es la compatibilidad con funciones personalizadas por inquilino. Si está diseñando un sistema de gestión de las relaciones con los clientes (CRM), un cliente podría definir la granularidad del acceso a las oportunidades de venta en función de un conjunto concreto de criterios. Otro inquilino podría tener otra definición. Los sistemas de permisos subyacentes de Verified Permissions admiten estas variaciones, lo que lo convierte en un excelente candidato para los casos de uso de SaaS. Los permisos verificados también permiten redactar políticas que se apliquen a todos los inquilinos, por lo que es sencillo aplicar políticas de protección para evitar el acceso no autorizado como proveedor de SaaS.



¿Por qué usar permisos verificados?

Use permisos verificados con un proveedor de identidad como Amazon Cognito para obtener una solución de administración de acceso más dinámica y basada en políticas para sus aplicaciones. Puede crear aplicaciones que ayuden a los usuarios a compartir información y colaborar, a la vez que mantienen la seguridad, la confidencialidad y la privacidad de sus datos. Los permisos verificados ayudan a reducir los costos operativos al proporcionarle un sistema de autorización detallado para imponer el acceso en función de las funciones y los atributos de sus identidades y recursos. Puede definir su modelo de políticas, crear y almacenar políticas en una ubicación central y evaluar las solicitudes de acceso en milisegundos.

En Verified Permissions, puedes expresar los permisos mediante un lenguaje declarativo simple y legible por humanos llamado Cedar. Las políticas que están escritas en Cedar se pueden compartir entre los equipos, independientemente del lenguaje de programación que utilice la aplicación de cada equipo.

¿Qué hay que tener en cuenta al utilizar permisos verificados

En Verified Permissions, puede crear políticas y automatizarlas como parte del aprovisionamiento. También puede crear políticas en tiempo de ejecución como parte de la lógica de la aplicación. Como práctica recomendada, debería utilizar una canalización de integración e implementación continuas (CI/CD) para administrar, modificar y realizar un seguimiento de las versiones de las políticas cuando cree políticas como parte de la incorporación y el aprovisionamiento de los inquilinos. Como alternativa, una aplicación puede administrar, modificar y realizar un seguimiento de las versiones de las políticas; sin embargo, la lógica de la aplicación no ofrece esta funcionalidad de forma inherente. Para admitir estas capacidades en su aplicación, debe diseñarla de forma explícita para implementar esta funcionalidad.

Si es necesario proporcionar datos externos de otras fuentes para tomar una decisión de autorización, estos datos deben recuperarse y proporcionarse a Verified Permissions como parte de la solicitud de autorización. El contexto, las entidades y los atributos adicionales no se recuperan de forma predeterminada con este servicio.

Descripción general de Cedar

Cedar es un lenguaje de control de acceso flexible, ampliable y escalable basado en políticas que ayuda a los desarrolladores a expresar los permisos de las aplicaciones como políticas. Los administradores y desarrolladores pueden definir políticas que permitan o prohíban a los usuarios actuar con los recursos de la aplicación. Se pueden adjuntar varias políticas a un único recurso. Cuando un usuario de la aplicación intenta realizar una acción en un recurso, la aplicación solicita

la autorización del motor de políticas de Cedar. Cedar evalúa las políticas aplicables y devuelve una `DENY` decisión `ALLOW` o decisión. Cedar respalda las reglas de autorización para cualquier tipo de capital y recurso, permite el control de acceso basado en roles (RBAC) y el control de acceso basado en atributos (ABAC), y apoya el análisis mediante herramientas de razonamiento automatizadas.

Cedar le permite separar la lógica empresarial de la lógica de autorización. Cuando realiza solicitudes desde el código de su aplicación, llama al motor de autorización de Cedar para determinar si la solicitud está autorizada. Si está autorizada (la decisión lo es`ALLOW`), su aplicación puede realizar la operación solicitada. Si no está autorizada (la decisión sí`DENY`), la aplicación puede devolver un mensaje de error. Las principales características de Cedar incluyen:

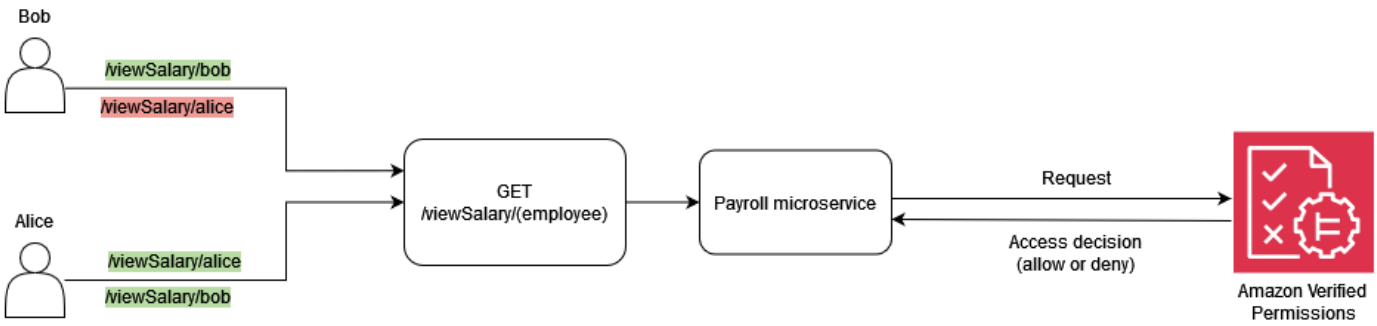
- Expresividad: Cedar está diseñado específicamente para respaldar los casos de uso de autorizaciones y se desarrolló teniendo en cuenta la legibilidad humana.
- Rendimiento: Cedar admite las políticas de indexación para una recuperación rápida y proporciona una evaluación rápida y escalable en tiempo real con una latencia limitada.
- Análisis: Cedar admite herramientas de análisis que pueden optimizar sus políticas y verificar su modelo de seguridad.

Para obtener más información, consulte el sitio web de Cedar.

Ejemplo 1: ABAC básico con permisos verificados y Cedar

En este escenario de ejemplo, Amazon Verified Permissions se utiliza para determinar qué usuarios pueden acceder a la información de un microservicio de nómina ficticio. Esta sección incluye fragmentos de código de Cedar para demostrar cómo se puede utilizar Cedar para tomar decisiones de control de acceso. Estos ejemplos no pretenden ofrecer una exploración completa de las capacidades que ofrecen Cedar y Verified Permissions. Para obtener una descripción más completa de Cedar, consulte la documentación de Cedar.

En el siguiente diagrama, nos gustaría aplicar dos reglas comerciales generales asociadas al `viewSalary` `GET` método: los empleados pueden ver su propio salario y los empleados pueden ver el salario de cualquier persona que dependa de ellos. Puedes hacer cumplir estas reglas empresariales mediante las políticas de permisos verificados.



Los empleados pueden ver su propio salario.

En Cedar, la construcción básica es una entidad, que representa un principal, una acción o un recurso. Para realizar una solicitud de autorización e iniciar una evaluación con una política de permisos verificados, debe proporcionar una entidad principal, una acción, un recurso y una lista de entidades.

- El principal (`principal`) es el usuario o rol que ha iniciado sesión.
- La acción (`action`) es la operación que evalúa la solicitud.
- El recurso (`resource`) es el componente al que accede la acción.
- La lista de entidades (`entityList`) contiene todas las entidades necesarias para evaluar la solicitud.

Para cumplir con la norma empresarial, los empleados pueden ver su propio salario, puedes proporcionar una política de permisos verificados como la siguiente.

```
permit (
    principal,
    action == Action::"viewSalary",
    resource
)
when {
    principal == resource.owner
};
```

Esta política evalúa `ALLOW` si el atributo `Action` is `viewSalary` y el recurso de la solicitud tienen un propietario de atributo igual al principal. Por ejemplo, si Bob es el usuario registrado que solicitó el informe salarial y también es el propietario del informe salarial, la política se evalúa como. `ALLOW`

La siguiente solicitud de autorización se envía a Verified Permissions para que la evalúe el modelo de política. En este ejemplo, Bob es el usuario que ha iniciado sesión y que realiza la `viewSalary` solicitud. Por lo tanto, Bob es el principal del tipo de entidad`Employee`. La acción que Bob intenta realizar es de ese tipo `viewSalary`, y el recurso que se `viewSalary` mostrará es `Salary-Bob` de ese tipo`Salary`. Para evaluar si Bob puede ver el `Salary-Bob` recurso, es necesario proporcionar una estructura de entidad que vincule el tipo `Employee` con un valor `Bob` (el principal) con el atributo propietario del recurso que contiene el tipo`Salary`. Esta estructura se proporciona en un`entityList`, donde los atributos asociados `Salary` incluyen un propietario, que especifica un propietario `entityIdentifier` que contiene el tipo `Employee` y el valor`Bob`. Verified Permissions compara lo `principal` proporcionado en la solicitud de autorización con el `owner` atributo asociado al `Salary` recurso para tomar una decisión.

```
{
  "policyStoreId": "PAYROLLAPP_POLICYSTOREID",
  "principal": {
    "entityType": "PayrollApp::Employee",
    "entityId": "Bob"
  },
  "action": {
    "actionType": "PayrollApp::Action",
    "actionId": "viewSalary"
  },
  "resource": {
    "entityType": "PayrollApp::Salary",
    "entityId": "Salary-Bob"
  },
  "entities": {
    "entityList": [
      {
        "identifier": {
          "entityType": "PayrollApp::Salary",
          "entityId": "Salary-Bob"
        },
        "attributes": {
          "owner": {
            "entityIdentifier": {
              "entityType": "PayrollApp::Employee",
              "entityId": "Bob"
            }
          }
        }
```

```
      },
      {
        "identifier": {
          "entityType": "PayrollApp::Employee",
          "entityId": "Bob"
        },
        "attributes": {}
      }
    ]
  }
}
```

La solicitud de autorización a Verified Permissions devuelve lo siguiente como resultado, donde el atributo `decision` es `ALLOW` o`DENY`.

```
{
    "determiningPolicies":
        [
            {
                "determiningPolicyId": "PAYROLLAPP_POLICYSTOREID"
            }
        ],
    "decision": "ALLOW",
    "errors": []
}
```

En este caso, dado que Bob estaba intentando ver su propio salario, la solicitud de autorización enviada a Verified Permissions equivale a`ALLOW`. Sin embargo, nuestro objetivo era utilizar los permisos verificados para hacer cumplir dos normas empresariales. La regla empresarial que establece lo siguiente también debería ser cierta:

Los empleados pueden ver el salario de cualquier persona que dependa de ellos.

Para cumplir con esta regla empresarial, puede proporcionar otra política. La siguiente política evalúa `ALLOW` si la acción es igual al principal `viewSalary` y si el recurso de la solicitud tiene un atributo `owner.manager` igual al principal. Por ejemplo, si Alice es la usuaria que inició sesión y solicitó el informe salarial y Alice es la administradora del propietario del informe, la política se evaluará como tal. `ALLOW`

```
permit (
    principal,
```

```
    action == Action::"viewSalary",
    resource
)
when {
    principal == resource.owner.manager
};
```

La siguiente solicitud de autorización se envía a Verified Permissions para que la evalúe el modelo de política. En este ejemplo, Alice es el usuario que ha iniciado sesión y que realiza la `viewSalary` solicitud. Por lo tanto, Alice es la principal y la entidad es de ese tipo`Employee`. La acción que Alice intenta realizar es`viewSalary`, y el recurso que `viewSalary` se mostrará es del tipo `Salary` con un valor de`Salary-Bob`. Para evaluar si Alice puede ver el `Salary-Bob` recurso, debe proporcionar una estructura de entidad que vincule el tipo `Employee` con un valor de `Alice` con el `manager` atributo, que luego debe asociarse al `owner` atributo del tipo `Salary` con un valor de`Salary-Bob`. Esta estructura se proporciona en una`entityList`, donde los atributos asociados `Salary` incluyen un propietario, que especifica una `entityIdentifier` que contiene el tipo `Employee` y el valorBob. Verified Permissions comprueba primero el `owner` atributo, que evalúa el tipo `Employee` y el valorBob. A continuación, Verified Permissions evalúa el `manager` atributo asociado `Employee` y lo compara con el principal proporcionado para tomar una decisión de autorización. En este caso, la decisión se `ALLOW` debe a que los `resource.owner.manager` atributos `principal` y son equivalentes.

```
{
  "policyStoreId": "PAYROLLAPP_POLICYSTOREID",
  "principal": {
    "entityType": "PayrollApp::Employee",
    "entityId": "Alice"
  },
  "action": {
    "actionType": "PayrollApp::Action",
    "actionId": "viewSalary"
  },
  "resource": {
    "entityType": "PayrollApp::Salary",
    "entityId": "Salary-Bob"
  },
  "entities": {
    "entityList": [
      {
        "identifier": {
```

```
          "entityType": "PayrollApp::Employee",
          "entityId": "Alice"
        },
        "attributes": {
          "manager": {
            "entityIdentifier": {
              "entityType": "PayrollApp::Employee",
              "entityId": "None"
            }
          }
        },
        "parents": []
      },
      {
        "identifier": {
          "entityType": "PayrollApp::Salary",
          "entityId": "Salary-Bob"
        },
        "attributes": {
          "owner": {
            "entityIdentifier": {
              "entityType": "PayrollApp::Employee",
              "entityId": "Bob"
            }
          }
        },
        "parents": []
      },
      {
        "identifier": {
          "entityType": "PayrollApp::Employee",
          "entityId": "Bob"
        },
        "attributes": {
          "manager": {
            "entityIdentifier": {
              "entityType": "PayrollApp::Employee",
              "entityId": "Alice"
            }
          }
        },
       "parents": []
      }
```

```
    ]
  }
}
```

Hasta ahora, en este ejemplo, hemos proporcionado las dos reglas de negocio asociadas al `viewSalary` método: los empleados pueden ver su propio salario y los empleados pueden ver el salario de cualquier persona que dependa de ellos. Los permisos verificados son políticas para cumplir las condiciones de cada regla de negocio de forma independiente. También puede utilizar una única política de permisos verificados para cumplir las condiciones de ambas reglas empresariales:

Los empleados pueden ver su propio salario y el de cualquier persona que dependa de ellos.

Al utilizar la solicitud de autorización anterior, la siguiente política evalúa `ALLOW` si la acción es `viewSalary` y el recurso de la solicitud tiene un atributo `owner.manager` igual a o un atributo `owner` igual a. `principal` `principal`

```
permit (
    principal,
    action == PayrollApp::Action::"viewSalary",
    resource
)
when {
    principal == resource.owner.manager ||
    principal == resource.owner
};
```

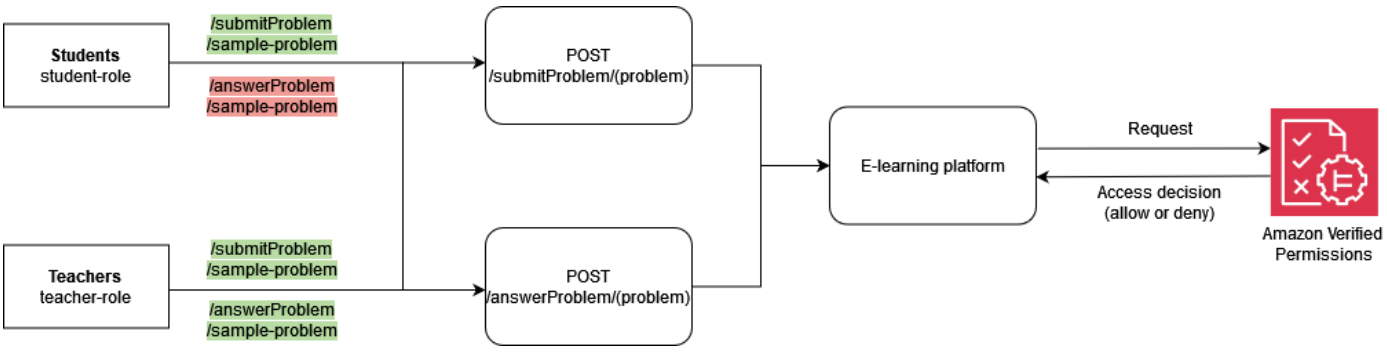
Por ejemplo, si Alice es el usuario que ha iniciado sesión y solicita el informe salarial y si Alice es la gestora del propietario o la propietaria del informe, la política se evalúa como. `ALLOW`

Para obtener más información sobre el uso de operadores lógicos con las políticas de Cedar, consulte la documentación de Cedar.

Ejemplo 2: RBAC básico con permisos verificados y Cedar

En este ejemplo, se utilizan permisos verificados y Cedar para demostrar el RBAC básico. Como se mencionó anteriormente, la construcción básica de Cedar es una entidad. Los desarrolladores definen sus propias entidades y, si lo desean, pueden crear relaciones entre entidades. El siguiente ejemplo incluye tres tipos de entidades: `UsersRoles`, y`Problems`. `Studentsy` `Teachers` pueden

considerarse entidades de este tipo `Role`, y cada una de ellas `User` puede estar asociada a cero o a cualquiera de las`Roles`.



En Cedar, estas relaciones se expresan vinculando el `Role` `Student` a `User` Bob como su matriz. Esta asociación agrupa de forma lógica a todos los usuarios estudiantes en un grupo. Para obtener más información sobre la agrupación en Cedar, consulte la documentación de Cedar.

La siguiente política evalúa la decisión de la acción `ALLOW` `submitProblem`, para todos los principales que están vinculados al grupo lógico `Students` de este tipo. `Role`

```
permit (
    principal in ElearningApp::Role::"Students",
    action == ElearningApp::Action::"submitProblem",
    resource
);
```

La siguiente política evalúa la decisión `ALLOW` de la acción `submitProblem` o `answerProblem` de todos los principios que están vinculados al grupo `Teachers` lógico de este tipo. `Role`

```
permit (
    principal in ElearningApp::Role::"Teachers",
    action in [
        ElearningApp::Action::"submitProblem",
        ElearningApp::Action::"answerProblem"
    ],
    resource
);
```

Para evaluar las solicitudes con estas políticas, el motor de evaluación necesita saber si el director al que se hace referencia en la solicitud de autorización es miembro del grupo apropiado. Por lo tanto, la solicitud debe transmitir la información relevante sobre la pertenencia al grupo al motor

de evaluación como parte de la solicitud de autorización. Esto se hace a través de la `entities` propiedad, que le permite proporcionar al motor de evaluación de Cedar los datos sobre los atributos y la pertenencia al grupo del director y el recurso involucrados en la solicitud de autorización. En el siguiente código, la pertenencia a un grupo se indica definiendo `User::"Bob"` como la llamada a un padre`Role::"Students"`.

```
{
  "policyStoreId": "ELEARNING_POLICYSTOREID",
  "principal": {
    "entityType": "ElearningApp::User",
    "entityId": "Bob"
  },
  "action": {
    "actionType": "ElearningApp::Action",
    "actionId": "answerProblem"
  },
  "resource": {
    "entityType": "ElearningApp::Problem",
    "entityId": "SomeProblem"
  },
  "entities": {
    "entityList": [
        {
            "identifier": {
                "entityType": "ElearningApp::User",
                "entityId": "Bob"
            },
            "attributes": {},
            "parents": [
                {
                    "entityType": "ElearningApp::Role",
                    "entityId": "Students"
                }
            ]
        },
        {
          "identifier": {
            "entityType": "ElearningApp::Problem",
            "entityId": "SomeProblem"
          },
          "attributes": {},
          "parents": []
```

```
        }
      ]
  }
}
```

En este ejemplo, Bob es el usuario registrado que realiza la `answerProblem` solicitud. Por lo tanto, Bob es el principal y la entidad es de ese tipo`User`. La acción que Bob intenta realizar es`answerProblem`. Para evaluar si Bob puede realizar la `answerProblem` acción, es necesario proporcionar una estructura de entidad que vincule a la entidad `User` con un valor de `Bob` y asigne su pertenencia a un grupo mediante la inclusión de una entidad principal como`Role::"Students"`. Como las entidades del grupo `Role::"Students"` de usuarios solo pueden realizar la acción`submitProblem`, esta solicitud de autorización se evalúa como. `DENY`

Por otro lado, si el tipo `User` que tiene un valor de `Alice` y forma parte del grupo `Role::"Teachers"` intenta realizar la `answerProblem` acción, la solicitud de autorización se evalúa como tal`ALLOW`, ya que la política establece que los directores del grupo pueden realizar la acción `answerProblem` en todos `Role::"Teachers"` los recursos. El código siguiente muestra este tipo de solicitud de autorización que se evalúa como. `ALLOW`

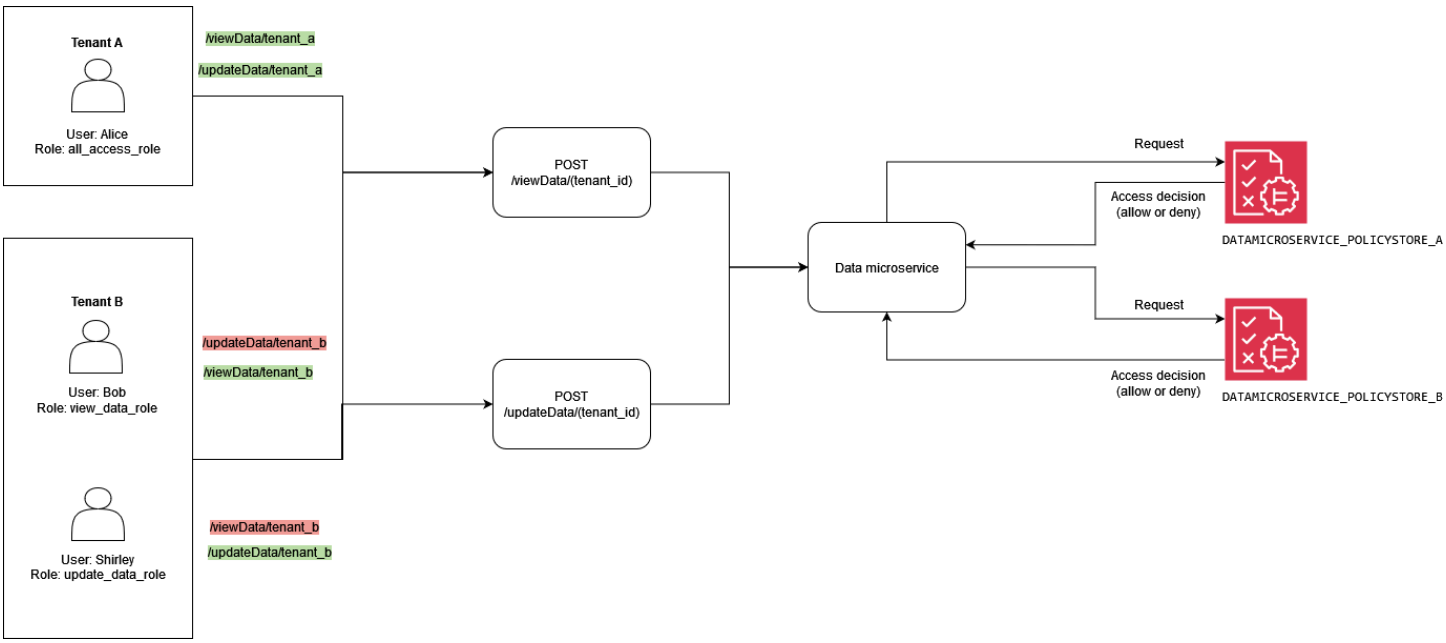
```
{
  "policyStoreId": "ELEARNING_POLICYSTOREID",
  "principal": {
    "entityType": "ElearningApp::User",
    "entityId": "Alice"
  },
  "action": {
    "actionType": "ElearningApp::Action",
    "actionId": "answerProblem"
  },
  "resource": {
    "entityType": "ElearningApp::Problem",
    "entityId": "SomeProblem"
  },
  "entities": {
    "entityList": [
        {
            "identifier": {
                "entityType": "ElearningApp::User",
                "entityId": "Alice"
            },
            "attributes": {},
```

```
            "parents": [
                {
                    "entityType": "ElearningApp::Role",
                    "entityId": "Teachers"
                }
            ]
        },
        {
            "identifier": {
                "entityType": "ElearningApp::Problem",
                "entityId": "SomeProblem"
            },
            "attributes": {},
            "parents": []
        }
      ]
  }
}
```

Ejemplo 3: Control de acceso multiusuario con RBAC

Para profundizar en el ejemplo anterior de RBAC, puede ampliar sus requisitos para incluir la multitenencia de SaaS, que es un requisito común para los proveedores de SaaS. En las soluciones multiarrendatario, el acceso a los recursos siempre se proporciona en nombre de un arrendatario determinado. Es decir, los usuarios del arrendatario A no pueden ver los datos del arrendatario B, incluso si esos datos están colocados lógica o físicamente en un sistema. El siguiente ejemplo ilustra cómo puede implementar el aislamiento de inquilinos mediante varios almacenes de políticas de permisos verificados y cómo puede emplear las funciones de usuario para definir los permisos dentro del inquilino.

Utilizar el patrón de diseño del almacén de políticas por inquilino es una práctica recomendada para mantener el aislamiento de los inquilinos y, al mismo tiempo, implementar el control de acceso con permisos verificados. En este escenario, las solicitudes de los usuarios del inquilino A y del inquilino B se verifican comparándolas con almacenes de políticas independientes `DATAMICROSERVICE_POLICYSTORE_A` y`DATAMICROSERVICE_POLICYSTORE_B`, respectivamente. Para obtener más información sobre las consideraciones de diseño de permisos verificados para aplicaciones SaaS de varios inquilinos, consulte la sección Consideraciones de diseño de permisos verificados para múltiples inquilinos.



La siguiente política se encuentra en el almacén de políticas. `DATAMICROSERVICE_POLICYSTORE_A` Verifica que el principal formará parte del grupo `allAccessRole` de tipo`Role`. En este caso, el director podrá realizar `updateData` las acciones `viewData` y acciones en todos los recursos que estén asociados al inquilino A.

```
permit (
    principal in MultitenantApp::Role::"allAccessRole",
    action in [
        MultitenantApp::Action::"viewData",
        MultitenantApp::Action::"updateData"
    ],
    resource
);
```

Las siguientes políticas se encuentran en el almacén `DATAMICROSERVICE_POLICYSTORE_B` de políticas. La primera política comprueba que el principal forma parte del `updateDataRole` grupo de tipos`Role`. Suponiendo que ese sea el caso, da permiso a los directores para realizar la `updateData` acción con los recursos que están asociados al inquilino B.

```
permit (
    principal in MultitenantApp::Role::"updateDataRole",
    action == MultitenantApp::Action::"updateData",
    resource
);
```

Esta segunda política exige que los directores que formen parte del `viewDataRole` grupo de este tipo `Role` puedan realizar la `viewData` acción con los recursos asociados al arrendatario B.

```
permit (
    principal in MultitenantApp::Role::"viewDataRole",
    action == MultitenantApp::Action::"viewData",
    resource
);
```

La solicitud de autorización realizada por el arrendatario A debe enviarse al almacén `DATAMICROSERVICE_POLICYSTORE_A` de políticas y verificarse mediante las políticas que pertenecen a ese almacén. En este caso, se verifica mediante la primera política analizada anteriormente como parte de este ejemplo. En esta solicitud de autorización, el principal de tipo `User` con un valor de `Alice` solicita realizar la `viewData` acción. El principal pertenece al grupo `allAccessRole` de tipos`Role`. Alice está intentando realizar la `viewData` acción en el `SampleData` recurso. Como Alice tiene la `allAccessRole` función, esta evaluación da como resultado una `ALLOW` decisión.

```
{
  "policyStoreId": "DATAMICROSERVICE_POLICYSTORE_A",
  "principal": {
      "entityType": "MultitenantApp::User",
      "entityId": "Alice"
  },
  "action": {
      "actionType": "MultitenantApp::Action",
      "actionId": "viewData"
  },
  "resource": {
      "entityType": "MultitenantApp::Data",
      "entityId": "SampleData"
  },
  "entities": {
    "entityList": [
      {
        "identifier": {
            "entityType": "MultitenantApp::User",
            "entityId": "Alice"
        },
        "attributes": {},
        "parents": [
```

```
            {
                "entityType": "MultitenantApp::Role",
                "entityId": "allAccessRole"
            }
        ]
      },
      {
        "identifier": {
            "entityType": "MultitenantApp::Data",
            "entityId": "SampleData"
        },
        "attributes": {},
        "parents": []
      }
    ]
  }
}
```

Si, por el contrario, consulta una solicitud realizada por el inquilino B`User Bob`, verá algo parecido a la siguiente solicitud de autorización. La solicitud se envía al almacén de `DATAMICROSERVICE_POLICYSTORE_B` políticas porque proviene del arrendatario B. En esta solicitud, el director `Bob` desea realizar la acción `updateData` en el recurso`SampleData`. Sin embargo, `Bob` no forma parte de un grupo que tenga acceso a la acción `updateData` de ese recurso. Por lo tanto, la solicitud da lugar a una `DENY` decisión.

```
{
  "policyStoreId": "DATAMICROSERVICE_POLICYSTORE_B",
  "principal": {
      "entityType": "MultitenantApp::User",
      "entityId": "Bob"
  },
  "action": {
      "actionType": "MultitenantApp::Action",
      "actionId": "updateData"
  },
  "resource": {
      "entityType": "MultitenantApp::Data",
      "entityId": "SampleData"
  },
  "entities": {
    "entityList": [
      {
```

```
        "identifier": {
            "entityType": "MultitenantApp::User",
            "entityId": "Bob"
        },
        "attributes": {},
        "parents": [
            {
                "entityType": "MultitenantApp::Role",
                "entityId": "viewDataRole"
            }
        ]
      },
      {
        "identifier": {
            "entityType": "MultitenantApp::Data",
            "entityId": "SampleData"
        },
        "attributes": {},
        "parents": []
      }
    ]
  }
}
```

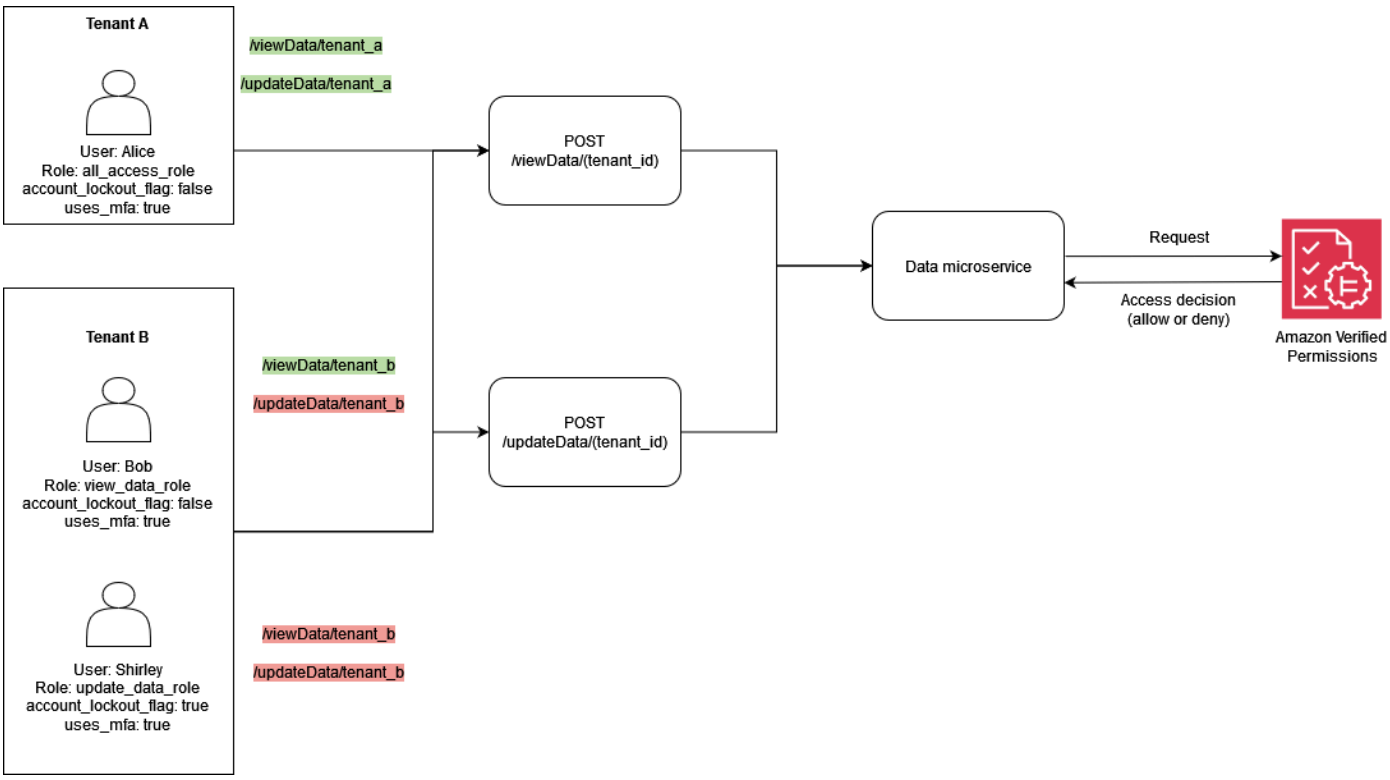
En este tercer ejemplo, `User Alice` intenta realizar la `viewData` acción en el recurso`SampleData`. Esta solicitud se dirige al almacén de `DATAMICROSERVICE_POLICYSTORE_A` políticas porque la principal `Alice` pertenece al arrendatario A. `Alice` forma parte de este tipo `allAccessRole` de grupo`Role`, lo que le permite realizar la `viewData` acción con los recursos. Como tal, la solicitud da lugar a una `ALLOW` decisión.

```
{
  "policyStoreId": "DATAMICROSERVICE_POLICYSTORE_A",
  "principal": {
      "entityType": "MultitenantApp::User",
      "entityId": "Alice"
  },
  "action": {
      "actionType": "MultitenantApp::Action",
      "actionId": "viewData"
  },
  "resource": {
      "entityType": "MultitenantApp::Data",
```

```
      "entityId": "SampleData"
  },
  "entities": {
    "entityList": [
      {
        "identifier": {
            "entityType": "MultitenantApp::User",
            "entityId": "Alice"
        },
        "attributes": {},
        "parents": [
            {
                "entityType": "MultitenantApp::Role",
                "entityId": "allAccessRole"
            }
        ]
      },
      {
        "identifier": {
            "entityType": "MultitenantApp::Data",
            "entityId": "SampleData"
        },
        "attributes": {},
        "parents": []
      }
    ]
  }
}
```

Ejemplo 4: Control de acceso multiusuario con RBAC y ABAC

Para mejorar el ejemplo del RBAC de la sección anterior, puede añadir atributos a los usuarios para crear un enfoque híbrido RBAC-ABAC para el control de acceso de varios usuarios. Este ejemplo incluye las mismas funciones del ejemplo anterior, pero añade el atributo de usuario y el parámetro de contexto. `account_lockout_flag` `uses_mfa` El ejemplo también adopta un enfoque diferente para implementar el control de acceso multiusuario mediante RBAC y ABAC, y usa un almacén de políticas compartido en lugar de un almacén de políticas diferente para cada inquilino.



Este ejemplo representa una solución SaaS multiusuario en la que debe proporcionar decisiones de autorización para el arrendatario A y el arrendatario B, de forma similar al ejemplo anterior.

Para implementar la función de bloqueo de usuarios, el ejemplo agrega el atributo `account_lockout_flag` a la `User` entidad principal en la solicitud de autorización. Este indicador bloquea el acceso del usuario al sistema y `DENY` otorga todos los privilegios al usuario bloqueado. El `account_lockout_flag` atributo está asociado a la `User` entidad y estará en vigor `User` hasta que la marca se revoque activamente en varias sesiones. En el ejemplo se usa la `when` condición para evaluar`account_lockout_flag`.

El ejemplo también agrega detalles sobre la solicitud y la sesión. La información de contexto especifica que la sesión se ha autenticado mediante la autenticación multifactorial. Para implementar esta validación, el ejemplo usa la `when` condición para evaluar el `uses_mfa` indicador como parte del campo de contexto. Para obtener más información sobre las prácticas recomendadas para añadir contexto, consulte la documentación de Cedar.

```
permit (
    principal in MultitenantApp::Role::"allAccessRole",
    action in [
        MultitenantApp::Action::"viewData",
        MultitenantApp::Action::"updateData"
```

```
    ],
    resource
)
when {
    principal.account_lockout_flag == false &&
    context.uses_mfa == true &&
    resource in principal.Tenant
};
```

Esta política impide el acceso a los recursos a menos que el recurso esté en el mismo grupo que el `Tenant` atributo del principal solicitante. Este enfoque para mantener el aislamiento de los inquilinos se denomina enfoque de un almacén de políticas compartido para varios inquilinos. Para obtener más información sobre las consideraciones de diseño de permisos verificados para aplicaciones SaaS de varios inquilinos, consulte la sección Consideraciones de diseño de permisos verificados para múltiples inquilinos.

La política también garantiza que el director sea miembro de `allAccessRole` y restringe las acciones a y. `viewData` `updateData` Además, esta política verifica que así `account_lockout_flag` sea `false` y que el valor de contexto para la `uses_mfa` evaluación sea igual a. `true`

Del mismo modo, la siguiente política garantiza que tanto el principal como el recurso estén asociados al mismo inquilino, lo que impide el acceso entre inquilinos. Esta política también garantiza que el director sea miembro `viewDataRole` y restringe las acciones a. `viewData` Además, verifica que el `account_lockout_flag` es `false` y que el valor de contexto para el que se `uses_mfa` evalúa. `true`

```
permit (
    principal in MultitenantApp::Role::"viewDataRole",
    action == MultitenantApp::Action::"viewData",
    resource
)
when {
    principal.account_lockout_flag == false &&
    context.uses_mfa == true &&
    resource in principal.Tenant
};
```

La tercera política es similar a la anterior. La política exige que el recurso sea miembro del grupo que corresponda a la entidad por la que está representado`principal.Tenant`. Esto garantiza

que tanto el principal como el recurso estén asociados al inquilino B, lo que impide el acceso entre inquilinos. Esta política garantiza que el director sea miembro `updateDataRole` y restringe las acciones a. `updateData` Además, esta política verifica que `account_lockout_flag` es `false` y que el valor de contexto se `uses_mfa` evalúa como tal. `true`

```
permit (
    principal in MultitenantApp::Role::"updateDataRole",
    action == MultitenantApp::Action::"updateData",
    resource
)
when {
    principal.account_lockout_flag == false &&
    context.uses_mfa == true &&
    resource in principal.Tenant
};
```

La siguiente solicitud de autorización se evalúa mediante las tres políticas descritas anteriormente en esta sección. En esta solicitud de autorización, el principal del tipo `User` y con un valor de `Alice` realiza una `updateData` solicitud con el rol`allAccessRole`. `Alice`tiene el atributo `Tenant` cuyo valor es`Tenant::"TenantA"`. La acción que `Alice` se intenta realizar es de ese tipo `updateData,` y el recurso al que se aplicará es `SampleData` de ese tipo`Data`. `SampleData`tiene `TenantA` como entidad principal.

Según la primera política del almacén de <DATAMICROSERVICE_POLICYSTOREID> políticas, `Alice` puede realizar la `updateData` acción en el recurso, suponiendo que se cumplan las condiciones de la `when` cláusula de la política. La primera condición requiere que el `principal.Tenant` atributo se evalúe`TenantA`. La segunda condición requiere que el atributo del principal `account_lockout_flag` sea`false`. La condición final requiere que el contexto `uses_mfa` sea`true`. Como se cumplen las tres condiciones, la solicitud devuelve una `ALLOW` decisión.

```
{
  "policyStoreId": "DATAMICROSERVICE_POLICYSTORE",
  "principal": {
      "entityType": "MultitenantApp::User",
      "entityId": "Alice"
  },
  "action": {
      "actionType": "MultitenantApp::Action",
      "actionId": "updateData"
```

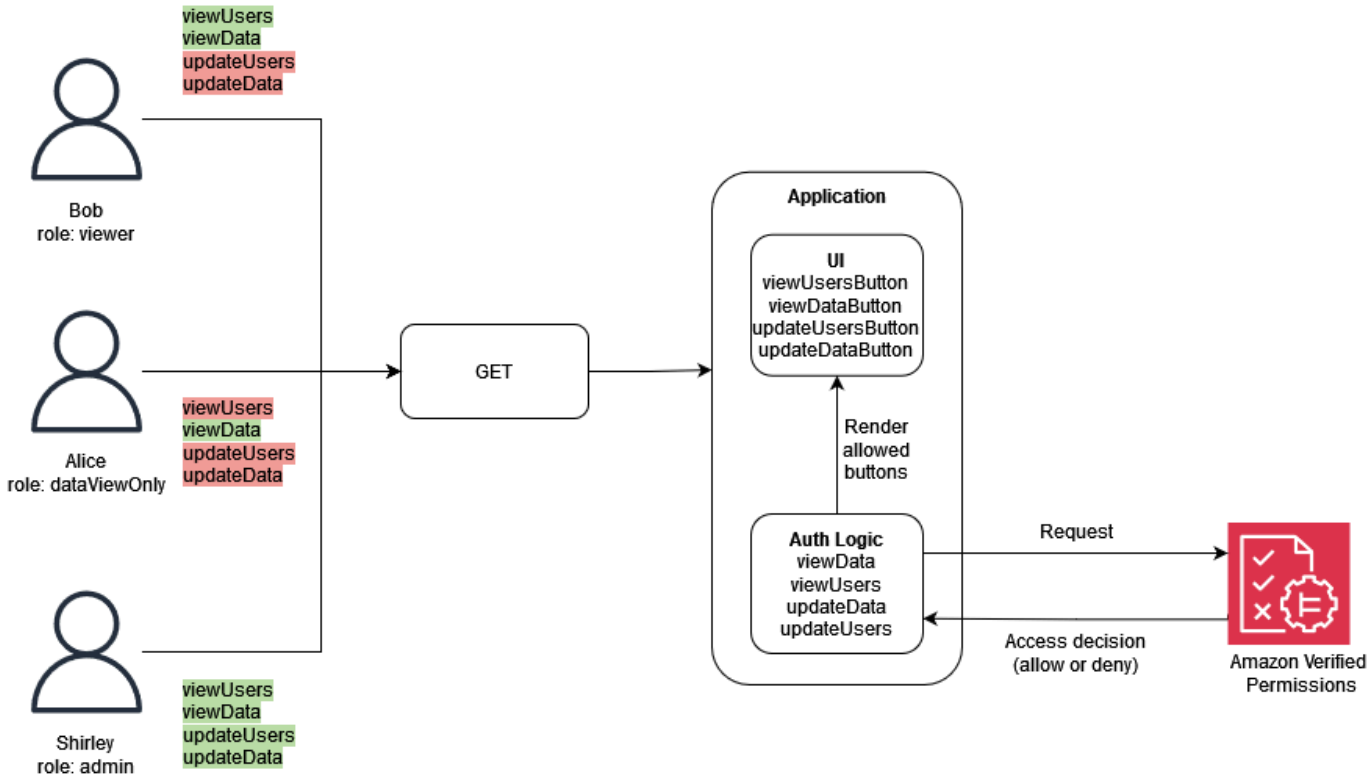
```
  },
  "resource": {
      "entityType": "MultitenantApp::Data",
      "entityId": "SampleData"
  },
  "context": {
    "contextMap": {
        "uses_mfa": {
            "boolean": true
        }
    }
  },
  "entities": {
    "entityList": [
      {
        "identifier": {
            "entityType": "MultitenantApp::User",
            "entityId": "Alice"
        },
        "attributes": {
            {
                "account_lockout_flag": {
                    "boolean": false
                },
                "Tenant": {
                   "entityIdentifier": {
                        "entityType":"MultitenantApp::Tenant",
                        "entityId":"TenantA"
                   }
                }
            }
        },
        "parents": [
            {
                "entityType": "MultitenantApp::Role",
                "entityId": "allAccessRole"
            }
        ]
        },
     {
        "identifier": {
            "entityType": "MultitenantApp::Data",
            "entityId": "SampleData"
```

```
        },
        "attributes": {},
        "parents": [
            {
                "entityType": "MultitenantApp::Tenant",
                "entityId": "TenantA"
            }
        ]
      }
    ]
  }
}
```

Ejemplo 5: filtrado de la interfaz de usuario con permisos verificados y Cedar

También puedes usar los permisos verificados para implementar el filtrado RBAC de los elementos de la interfaz de usuario en función de las acciones autorizadas. Esto es extremadamente valioso para las aplicaciones que tienen elementos de interfaz de usuario sensibles al contexto que podrían estar asociados a usuarios o inquilinos específicos en el caso de una aplicación SaaS multiusuario.

En el siguiente ejemplo, `Users` no `Role` `viewer` están autorizados a realizar actualizaciones. Para estos usuarios, la interfaz de usuario no debería mostrar ningún botón de actualización.



En este ejemplo, una aplicación web de una sola página tiene cuatro botones. Los botones que están visibles dependen `Role` del usuario que esté actualmente conectado a la aplicación. A medida que la aplicación web de una sola página renderiza la interfaz de usuario, consulta los permisos verificados para determinar qué acciones está autorizado a realizar el usuario y, a continuación, genera los botones en función de la decisión de autorización.

La siguiente política especifica que el tipo `Role` con un valor de `viewer` puede ver tanto los usuarios como los datos. La decisión de `ALLOW` autorizar esta política requiere una `viewUsers` acción `viewData` o, además, requiere que se asocie un recurso al tipo `Data` o`Users`. Una `ALLOW` decisión permite a la interfaz de usuario representar dos botones: `viewDataButton` y`viewUsersButton`.

```
permit (
    principal in GuiAPP::Role::"viewer",
    action in [GuiAPP::Action::"viewData", GuiAPP::Action::"viewUsers"],
    resource
)
when {
   resource in [GuiAPP::Type::"Data", GuiAPP::Type::"Users"]
};
```

La siguiente política especifica que el tipo `Role` con un valor de solo `viewerDataOnly` puede ver datos. La decisión de `ALLOW` autorizar esta política requiere una `viewData` acción y también requiere que se asocie un recurso al tipo`Data`. Una `ALLOW` decisión permite que la interfaz de usuario renderice el botón`viewDataButton`.

```
permit (
    principal in GuiApp::Role::"viewerDataOnly",
    action in [GuiApp::Action::"viewData"],
    resource in [GuiApp::Type::"Data"]
);
```

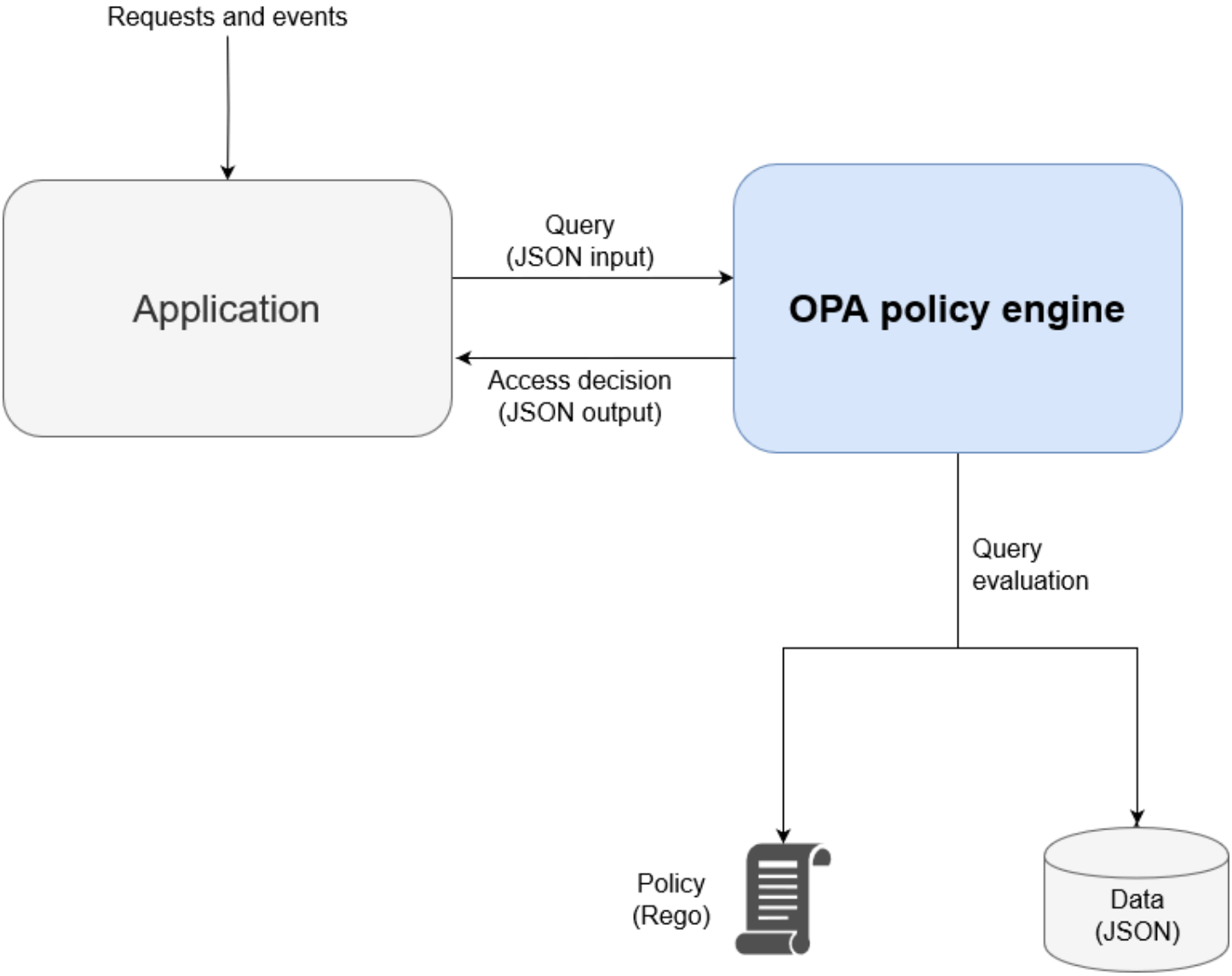
La siguiente política especifica que el tipo `Role` con un valor de `admin` puede editar y ver datos y usuarios. La decisión de `ALLOW` autorizar esta política requiere una acción de `updateDataupdateUsers`, `viewData,` o`viewUsers`, y también requiere que se asocie un recurso al tipo `Data` o`Users`. Una `ALLOW` decisión permite a la interfaz de usuario representar los cuatro botones: `updateDataButton`o`updateUsersButton`,`viewDataButton`, y`viewUsersButton`.

```
permit (
    principal in GuiApp::Role::"admin",
    action in [
        GuiApp::Action::"updateData",
        GuiApp::Action::"updateUsers",
        GuiApp::Action::"viewData",
        GuiApp::Action::"viewUsers"
       ],
    resource
)
when {
   resource in [GuiApp::Type::"Data", GuiApp::Type::"Users"]
};
```

Implementación de un PDP mediante OPA

El Open Policy Agent (OPA) es un motor de políticas de código abierto y de uso general. La OPA tiene muchos casos de uso, pero el caso de uso relevante para la implementación del PDP es su capacidad para desvincular la lógica de autorización de una aplicación. Esto se denomina desacoplamiento de políticas. La OPA es útil para implementar un PDP por varias razones. Utiliza un lenguaje declarativo de alto nivel llamado Rego para redactar políticas y reglas. Estas políticas

y reglas existen por separado de una aplicación y pueden generar decisiones de autorización sin ninguna lógica específica de la aplicación. La OPA también incluye una RESTful API para que las decisiones de recuperación de autorizaciones sean sencillas y directas. Para tomar una decisión de autorización, una aplicación consulta la OPA con una entrada de JSON y la OPA evalúa la entrada con arreglo a las políticas especificadas para devolver una decisión de acceso en JSON. La OPA también es capaz de importar datos externos que podrían ser relevantes para tomar una decisión de autorización.



La OPA tiene varias ventajas en comparación con los motores de políticas personalizados:

- La OPA y su evaluación de políticas con Rego proporcionan un motor de políticas flexible y prediseñado que solo requiere la inserción de las políticas y cualquier dato necesario para tomar

decisiones de autorización. Esta lógica de evaluación de políticas tendría que recrearse en una solución de motor de políticas personalizada.

- La OPA simplifica la lógica de autorización al tener las políticas escritas en un lenguaje declarativo. Puede modificar y administrar estas políticas y reglas independientemente del código de cualquier aplicación, sin necesidad de conocimientos de desarrollo de aplicaciones.
- OPA presenta una RESTful API que simplifica la integración con los puntos de aplicación de las políticas ()PEPs.
- OPA proporciona soporte integrado para validar y decodificar los JSON Web Tokens (). JWTs
- La OPA es un estándar de autorización reconocido, lo que significa que la documentación y los ejemplos son abundantes si necesita ayuda o investigación para resolver un problema en particular.
- La adopción de un estándar de autorización como la OPA permite que las políticas redactadas en Rego se compartan entre los equipos, independientemente del lenguaje de programación que utilice la aplicación del equipo.

Hay dos cosas que la OPA no proporciona automáticamente:

- La OPA no tiene un plano de control sólido para actualizar y administrar las políticas. La OPA proporciona algunos patrones básicos para implementar las actualizaciones de políticas, el monitoreo y la agregación de registros al exponer una API de administración, pero la integración con esta API debe ser gestionada por el usuario de OPA. Como práctica recomendada, debe utilizar una canalización de integración y despliegue continuos (CI/CD) para administrar, modificar y realizar un seguimiento de las versiones de las políticas y gestionar las políticas en la OPA.
- De forma predeterminada, OPA no puede recuperar datos de fuentes externas. Una fuente de datos externa para una decisión de autorización podría ser una base de datos que contenga los atributos del usuario. Existe cierta flexibilidad en la forma en que se proporcionan los datos externos a la OPA (se pueden almacenar en caché local con antelación o se pueden recuperar de forma dinámica desde una API cuando se solicita una decisión de autorización), pero la OPA no puede obtener esta información en su nombre.

Descripción general de Rego

Rego es un lenguaje de políticas de uso general, lo que significa que funciona para cualquier capa y dominio. El objetivo principal de Rego es aceptar entradas y datos de JSON/YAML que se evalúan para tomar decisiones basadas en políticas sobre los recursos, las identidades y las operaciones

de la infraestructura. Rego le permite redactar políticas sobre cualquier capa de una pila o dominio sin necesidad de cambiar o ampliar el idioma. Estos son algunos ejemplos de decisiones que puede tomar Rego:

- ¿Se permite o deniega esta solicitud de API?
- ¿Cuál es el nombre de host del servidor de respaldo de esta aplicación?
- ¿Cuál es la puntuación de riesgo de este cambio de infraestructura propuesto?
- ¿En qué clústeres se debe implementar este contenedor para obtener una alta disponibilidad?
- ¿Qué información de enrutamiento debe usarse para este microservicio?

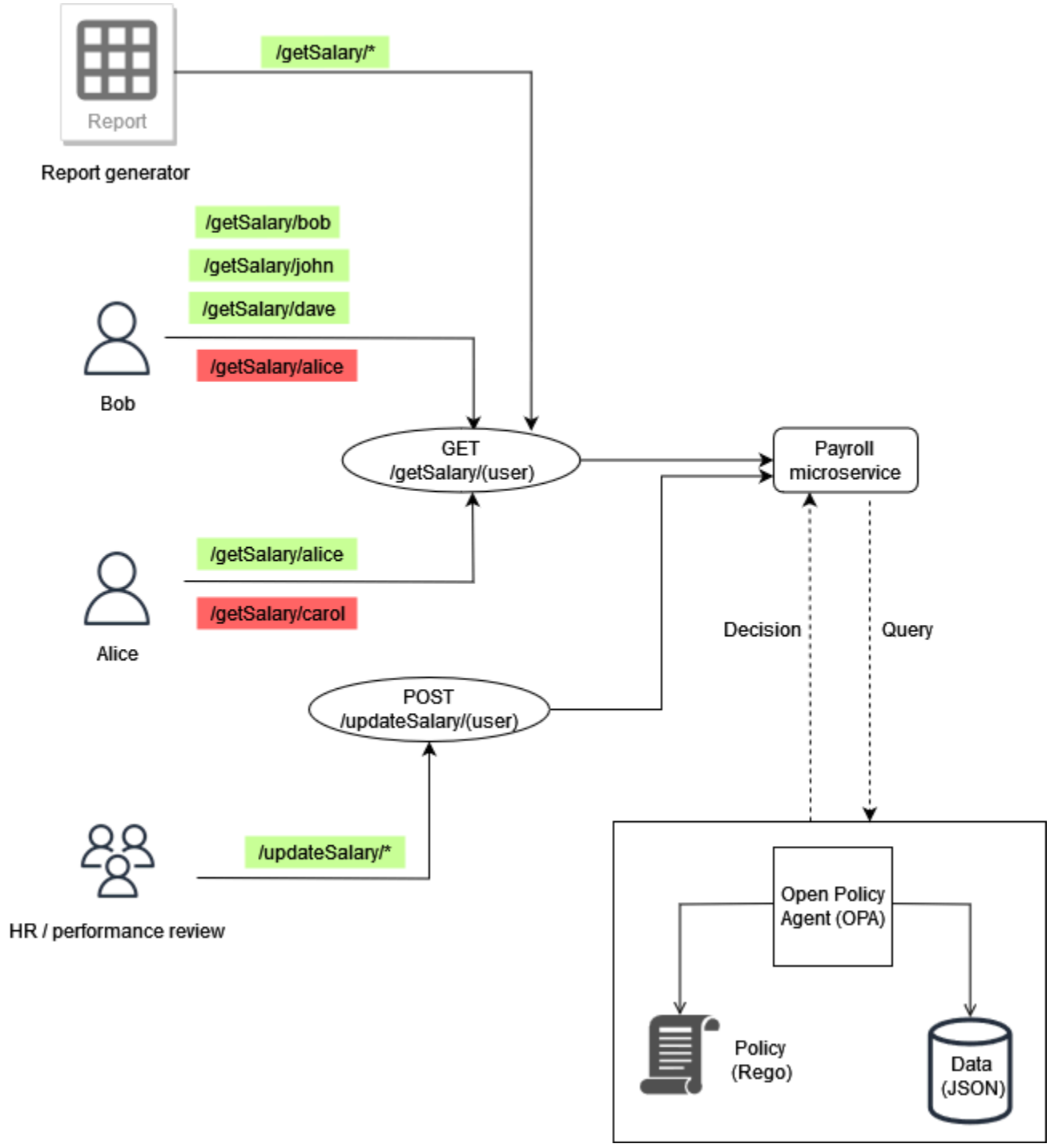
Para responder a estas preguntas, Rego emplea una filosofía básica sobre cómo se pueden tomar estas decisiones. Los dos principios clave a la hora de redactar la política en Rego son:

- Cada recurso, identidad u operación se puede representar como datos JSON o YAML.
- La política es la lógica que se aplica a los datos.

Rego ayuda a los sistemas de software a tomar decisiones de autorización al definir la lógica sobre cómo se evalúan las entradas de JSON/YAML datos. Los lenguajes de programación como C, Java, Go y Python son la solución habitual a este problema, pero Rego se diseñó para centrarse en los datos y las entradas que representan su sistema y en la lógica para tomar decisiones políticas con esta información.

Ejemplo 1: ABAC básico con OPA y Rego

En esta sección se describe un escenario en el que la OPA se utiliza para tomar decisiones de acceso sobre qué usuarios pueden acceder a la información de un microservicio ficticio de nómina. Los fragmentos de código de Rego se proporcionan para demostrar cómo se puede utilizar Rego para tomar decisiones de control de acceso. Estos ejemplos no son exhaustivos ni una exploración completa de las capacidades de Rego y OPA. Para obtener una descripción más completa de Rego, le recomendamos que consulte la documentación de Rego en el sitio web de la OPA.



Ejemplo de reglas básicas de OPA

En el diagrama anterior, una de las reglas de control de acceso que la OPA aplica al microservicio de nóminas es:

Los empleados pueden leer su propio salario.

Si Bob intenta acceder al microservicio de nómina para ver su propio salario, el microservicio de nómina puede redirigir la llamada a la API a la RESTful API de OPA para tomar una decisión de acceso. El servicio de nómina consulta a la OPA para tomar una decisión con la siguiente entrada de JSON:

```
{
    "user": "bob",
    "method": "GET",
    "path": ["getSalary", "bob"]
}
```

La OPA selecciona una o varias políticas en función de la consulta. En este caso, la siguiente política, que está escrita en Rego, evalúa la entrada de JSON.

```
default allow = false
allow = true {
    input.method == "GET"
    input.path = ["getSalary", user]
    input.user == user
}
```

Esta política deniega el acceso de forma predeterminada. A continuación, evalúa la entrada de la consulta vinculándola a la variable `input` global. El operador de punto se usa con esta variable para acceder a los valores de la variable. La regla Rego `allow` devuelve el valor verdadero si las expresiones de la regla también lo son. La regla Rego verifica que la `method` entrada sea igual a GET. Luego verifica que el primer elemento de la lista `path` esté `getSalary` antes de asignar el segundo elemento de la lista a la variable. `user` Por último, comprueba que la ruta a la que se accede es `/getSalary/bob` comprobando que la variable que se está `user` realizando la solicitud coincide con la `user` variable. `input.user` La regla `allow` aplica la lógica if-then para devolver un valor booleano, como se muestra en el resultado:

```
{
    "allow": true
}
```

Regla parcial que utiliza datos externos

Para demostrar las capacidades adicionales de la OPA, puede añadir requisitos a la regla de acceso que está aplicando. Supongamos que desea hacer cumplir este requisito de control de acceso en el contexto de la ilustración anterior:

Los empleados pueden leer el salario de cualquier persona que dependa de ellos.

En este ejemplo, OPA tiene acceso a datos externos que se pueden importar para tomar una decisión de acceso:

```
"managers": {
        "bob": ["dave", "john"],
        "carol": ["alice"]
}
```

Puede generar una respuesta JSON arbitraria creando una regla parcial en OPA, que devuelva un conjunto de valores en lugar de una respuesta fija. Este es un ejemplo de una regla parcial:

```
direct_report[user_ids] {
    user_ids = data.managers[input.user][_]
}
```

Esta regla devuelve un conjunto de todos los usuarios que informan con el valor de`input.user`, que, en este caso, es`bob`. La `[_]` construcción de la regla se usa para recorrer en iteración los valores del conjunto. Este es el resultado de la regla:

```
{
    "direct_report": [
      "dave",
      "john"
    ]
}
```

La recuperación de esta información puede ayudar a determinar si un usuario depende directamente de un administrador. Para algunas aplicaciones, es preferible devolver un JSON dinámico a devolver una respuesta booleana simple.

Resumen global

El último requisito de acceso es más complejo que los dos primeros porque combina las condiciones especificadas en ambos requisitos:

Los empleados pueden leer su propio salario y el de cualquier persona que dependa de ellos.

Para cumplir con este requisito, puede utilizar esta política de Rego:

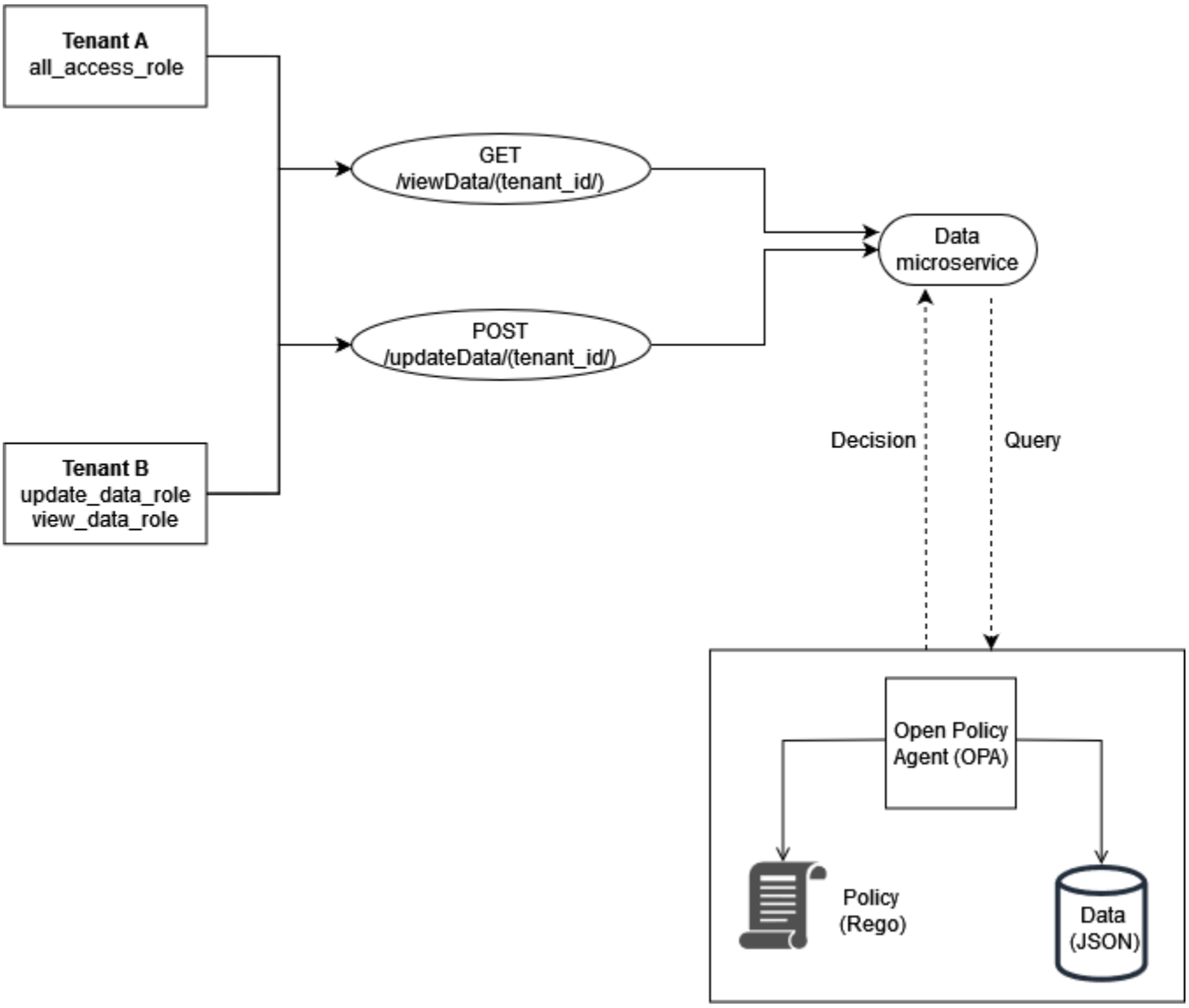
```
default allow = false

allow = true {
    input.method == "GET"
    input.path = ["getSalary", user]
    input.user == user
}

allow = true {
    input.method == "GET"
    input.path = ["getSalary", user]
    managers := data.managers[input.user][_]
    contains(managers, user)
}
```

La primera regla de la política permite el acceso a cualquier usuario que intente ver su propia información salarial, como se mencionó anteriormente. Al tener dos reglas con el mismo nombre`allow`, funciona como un lógico u operador en Rego. La segunda regla recupera la lista de todos los subordinados directos asociados `input.user` (a partir de los datos del diagrama anterior) y asigna esta lista a la variable. `managers` Por último, la regla comprueba si el usuario que está intentando ver su salario depende directamente de él, `input.user` comprobando que su nombre aparece en la variable. `managers`

Los ejemplos de esta sección son muy básicos y no proporcionan una exploración completa o exhaustiva de las capacidades de Rego y OPA. Para obtener más información, consulte la documentación de la OPA, consulte el archivo GitHub README de la OPA y experimente en el entorno de juegos de Rego.

Ejemplo 2: Control de acceso multiusuario y RBAC definido por el usuario con OPA y Rego

En este ejemplo, se utilizan OPA y Rego para demostrar cómo se puede implementar el control de acceso en una API para una aplicación multiusuario con funciones personalizadas definidas por los usuarios arrendatarios. También demuestra cómo se puede restringir el acceso en función del inquilino. Este modelo muestra cómo OPA puede tomar decisiones detalladas sobre permisos basándose en la información que se proporciona en un puesto de alto nivel.



Las funciones de los inquilinos se almacenan en datos externos (datos RBAC) que se utilizan para tomar decisiones de acceso a la OPA:

```
{
    "roles": {
        "tenant_a": {
            "all_access_role": ["viewData", "updateData"]
        },
        "tenant_b": {
            "update_data_role": ["updateData"],
            "view_data_role": ["viewData"]
        }
    }
}
```

Estas funciones, cuando las define un usuario arrendatario, deben almacenarse en una fuente de datos externa o en un proveedor de identidad (IdP) que pueda actuar como fuente de verdad al asignar las funciones definidas por el inquilino a los permisos y al propio inquilino.

En este ejemplo, se utilizan dos políticas de la OPA para tomar decisiones de autorización y para examinar cómo estas políticas imponen el aislamiento de los inquilinos. Estas políticas utilizan los datos del RBAC definidos anteriormente.

```
default allowViewData = false
allowViewData = true {
    input.method == "GET"
    input.path = ["viewData", tenant_id]
    input.tenant_id == tenant_id
    role_permissions := data.roles[input.tenant_id][input.role][_]
    contains(role_permissions, "viewData")
}
```

Para mostrar cómo funcionará esta regla, considere una consulta OPA que tenga la siguiente entrada:

```
{
    "tenant_id": "tenant_a",
    "role": "all_access_role",
    "path": ["viewData", "tenant_a"],
    "method": "GET"
}
```

La decisión de autorización para esta llamada a la API se toma de la siguiente manera, combinando los datos del RBAC, las políticas de la OPA y la entrada de la consulta de la OPA:

1. Un usuario de `Tenant A` realiza una llamada a la API a. `/viewData/tenant_a`
2. El microservicio de datos recibe la llamada y consulta la `allowViewData` regla, pasando la entrada que se muestra en el ejemplo de entrada de consulta de OPA.
3. La OPA utiliza la regla consultada en las políticas de la OPA para evaluar la información proporcionada. La OPA también utiliza los datos del RBAC para evaluar la entrada. La OPA hace lo siguiente:
   a. Verifica que el método utilizado para realizar la llamada a la API sea`GET`.
   b. Verifica que la ruta solicitada sea. `viewData`
   c. Comprueba que `tenant_id` la ruta es igual a la `input.tenant_id` asociada al usuario. Esto garantiza que se mantenga el aislamiento del inquilino. No se puede autorizar a otro inquilino, incluso con una función idéntica, a realizar esta llamada a la API.
   d. Extrae una lista de permisos de rol de los datos externos de los roles y los asigna a la variable. `role_permissions` Esta lista se recupera mediante el rol definido por el inquilino que está asociado al usuario en `input.role.`
   e. Comprueba `role_permissions` si contiene el permiso `viewData.`
4. La OPA devuelve la siguiente decisión al microservicio de datos:

```
{
    "allowViewData": true
}
```

Este proceso muestra cómo la RBAC y el conocimiento de los inquilinos pueden contribuir a tomar una decisión de autorización con la OPA. Para ilustrar mejor este punto, considere la posibilidad de hacer una llamada a la API `/viewData/tenant_b` con la siguiente entrada de consulta:

```
{
    "tenant_id": "tenant_b",
    "role": "view_data_role",
    "path": ["viewData", "tenant_b"],
    "method": "GET"
}
```

Esta regla devolvería el mismo resultado que la entrada de la consulta OPA, aunque es para un inquilino diferente que tiene un rol diferente. Esto se debe a que esta llamada es para `/tenant_b` y los datos incluidos `view_data_role` en el RBAC todavía tienen el `viewData` permiso asociado. Para aplicar el mismo tipo de control de acceso`/updateData`, puedes usar una regla OPA similar:

```
default allowUpdateData = false
allowUpdateData = true {
    input.method == "POST"
    input.path = ["updateData", tenant_id]
    input.tenant_id == tenant_id
    role_permissions := data.roles[input.tenant_id][input.role][_]
    contains(role_permissions, "updateData")
}
```

Esta regla es funcionalmente igual a la `allowViewData` regla, pero verifica una ruta y un método de entrada diferentes. La regla sigue garantizando el aislamiento del inquilino y comprueba que el rol definido por el inquilino concede el permiso al que llama a la API. Para ver cómo se puede aplicar esto, examine la siguiente entrada de consulta para ver si hay una llamada a la API a: `/updateData/tenant_b`

```
{
    "tenant_id": "tenant_b",
    "role": "view_data_role",
    "path": ["updateData", "tenant_b"],
    "method": "POST"
}
```

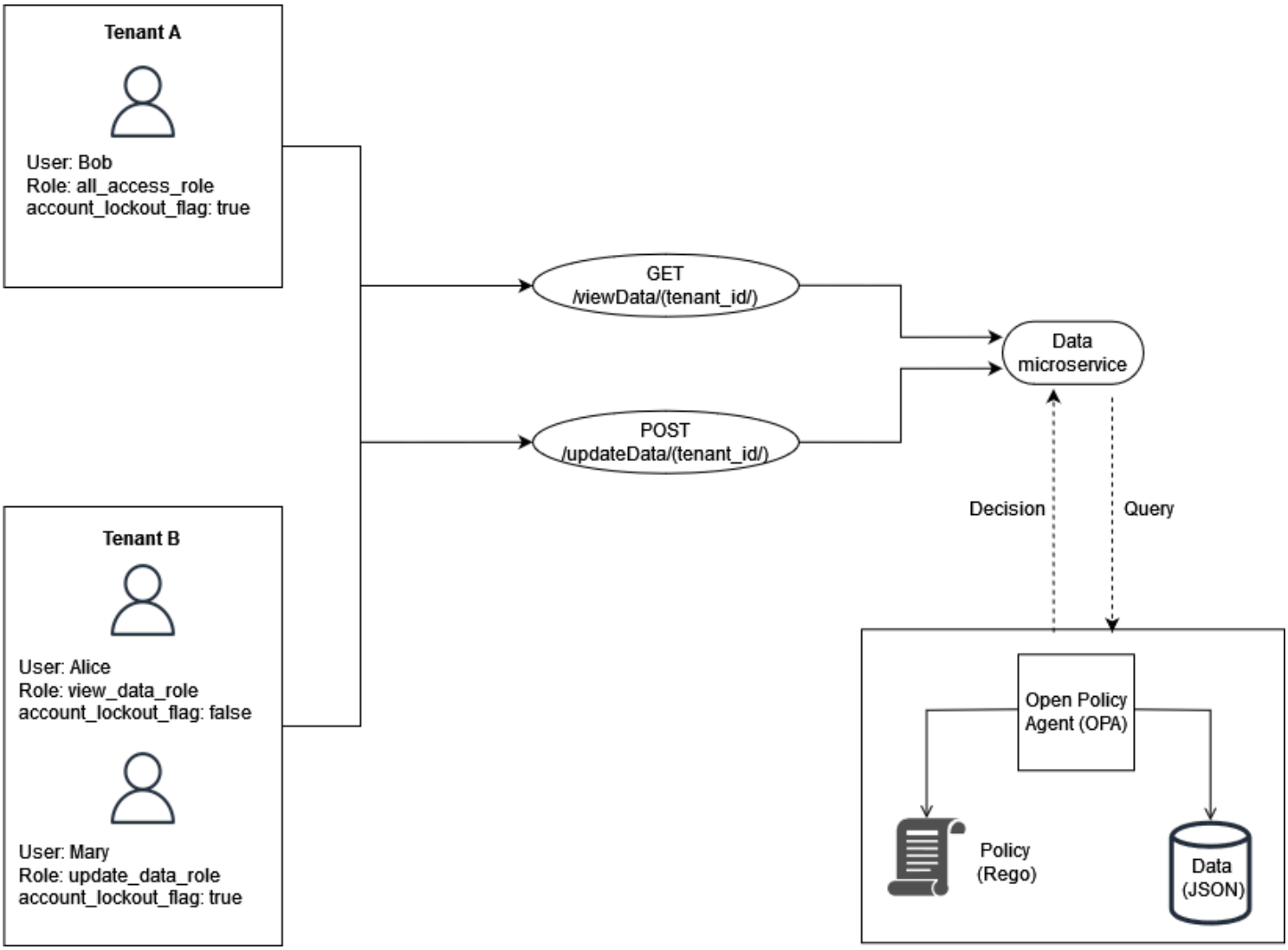
Esta entrada de consulta, cuando se evalúa con la `allowUpdateData` regla, devuelve la siguiente decisión de autorización:

```
{
    "allowUpdateData": false
}
```

Esta llamada no se autorizará. Si bien la persona que llama a la API está asociada a la correcta `tenant_id` y llama a la API mediante un método aprobado, `input.role` es el definido por el `view_data_role` inquilino. `view_data_role`No tiene el `updateData` permiso; por lo tanto, la llamada a `/updateData` no está autorizada. Esta llamada se habría realizado correctamente para un `tenant_b` usuario que tiene el`update_data_role`.

Ejemplo 3: Control de acceso multiusuario para RBAC y ABAC con OPA y Rego

Para mejorar el ejemplo del RBAC de la sección anterior, puede añadir atributos a los usuarios.



Este ejemplo incluye las mismas funciones del ejemplo anterior, pero añade el atributo user. `account_lockout_flag` Se trata de un atributo específico del usuario que no está asociado a ningún rol en particular. Puede utilizar los mismos datos externos del RBAC que utilizó anteriormente para este ejemplo:

```
{
   "roles": {
       "tenant_a": {
           "all_access_role": ["viewData", "updateData"]
       },
```

```
        "tenant_b": {
            "update_data_role": ["updateData"],
            "view_data_role": ["viewData"]
        }
    }
}
```

El atributo de `account_lockout_flag` usuario se puede pasar al servicio de datos como parte de la entrada de una consulta OPA `/viewData/tenant_a` para el usuario Bob:

```
 {
    "tenant_id": "tenant_a",
    "role": "all_access_role",
    "path": ["viewData", "tenant_a"],
    "method": "GET",
    "account_lockout_flag": "true"
}
```

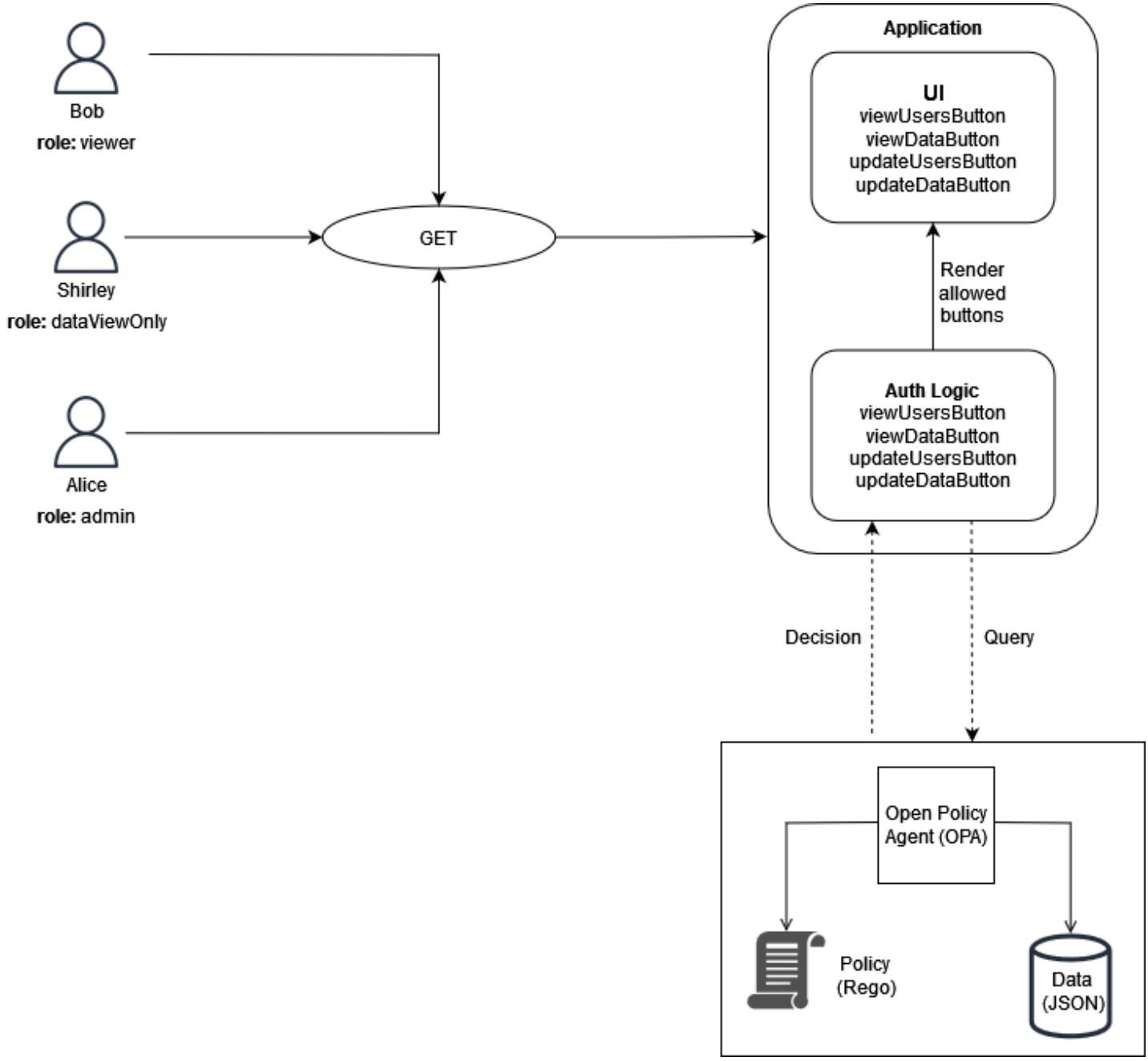
La regla que se consulta para la decisión de acceso es similar a la de los ejemplos anteriores, pero incluye una línea adicional para comprobar el `account_lockout_flag` atributo:

```
default allowViewData = false
allowViewData = true {
    input.method == "GET"
    input.path = ["viewData", tenant_id]
    input.tenant_id == tenant_id
    role_permissions := data.roles[input.tenant_id][input.role][_]
    contains(role_permissions, "viewData")
    input.account_lockout_flag == "false"
}
```

Esta consulta devuelve una decisión de autorización de`false`. Esto se debe a que `account_lockout_flag attribute` es `true` para Bob y la regla Rego `allowViewData` niega el acceso aunque Bob tenga el rol y el inquilino correctos.

Ejemplo 4: filtrado de la interfaz de usuario con OPA y Rego

La flexibilidad de OPA y Rego permite filtrar los elementos de la interfaz de usuario. El siguiente ejemplo demuestra cómo una regla parcial de OPA puede tomar decisiones de autorización sobre qué elementos deben mostrarse en una interfaz de usuario con RBAC. Este método es una de las muchas formas diferentes de filtrar los elementos de la interfaz de usuario con OPA.



En este ejemplo, una aplicación web de una sola página tiene cuatro botones. Supongamos que desea filtrar la interfaz de usuario de Bob, Shirley y Alice para que solo vean los botones que corresponden a sus funciones. Cuando la interfaz de usuario recibe una solicitud del usuario, consulta una regla parcial de la OPA para determinar qué botones deben mostrarse en la interfaz de usuario. La consulta pasa lo siguiente como entrada a OPA cuando Bob (con el rol`viewer`) realiza una solicitud a la interfaz de usuario:

```
{
```

```
    "role": "viewer"
}
```

La OPA utiliza datos externos estructurados para que el RBAC tome una decisión de acceso:

```
{
    "roles": {
        "viewer": ["viewUsers", "viewData"],
        "dataViewOnly": ["viewData"],
        "admin": ["viewUsers", "viewData", "updateUsers", "updateData"]
    }
}
```

La regla parcial de la OPA utiliza tanto los datos externos como la entrada para producir una lista de acciones permitidas:

```
user_permissions[permissions] {
    permissions := data.roles[input.role][_]
}
```

En la regla parcial, OPA usa lo `input.role` especificado como parte de la consulta para determinar qué botones deben mostrarse. Bob tiene esa función`viewer`, y los datos externos especifican que los espectadores tienen dos permisos: `viewUsers` y`viewData`. Por lo tanto, el resultado de esta regla para Bob (y para cualquier otro usuario que tenga un rol de espectador) es el siguiente:

```
{
    "user_permissions": [
        "viewData",
        "viewUsers"
    ]
}
```

El resultado para Shirley, que tiene el `dataViewOnly` rol, contendría un botón de permisos:. `viewData` El resultado para Alice, quien tiene el `admin` rol, contendría todos estos permisos. Estas respuestas se devuelven a la interfaz de usuario cuando se solicita `user_permissions` OPA. A continuación, la aplicación puede utilizar esta respuesta para ocultar o mostrar los `viewUsersButton` caracteres`viewDataButton`,`updateUsersButton`, y. `updateDataButton`

Uso de un motor de políticas personalizado

Un método alternativo para implementar un PDP es crear un motor de políticas personalizado. El objetivo de este motor de políticas es desvincular la lógica de autorización de una aplicación. El motor de políticas personalizado es responsable de tomar decisiones de autorización, de forma similar a los permisos verificados o la OPA, para lograr la disociación de las políticas. La principal diferencia entre esta solución y el uso de permisos verificados o OPA es que la lógica para redactar y evaluar las políticas está diseñada a medida para un motor de políticas personalizado. Cualquier interacción con el motor debe exponerse a través de una API o algún otro método para permitir que las decisiones de autorización lleguen a la aplicación. Puede escribir un motor de políticas personalizado en cualquier lenguaje de programación o utilizar otros mecanismos para la evaluación de políticas, como el Common Expression Language (CEL).

Implementación de un PEP

Un punto de aplicación de políticas (PEP) es responsable de recibir las solicitudes de autorización que se envían al punto de decisión de políticas (PDP) para su evaluación. Un PEP puede estar en cualquier parte de una aplicación donde se deban proteger los datos y los recursos o donde se aplique la lógica de autorización. PEPs son relativamente simples en comparación con PDPs. Un PEP es responsable únicamente de solicitar y evaluar una decisión de autorización y no requiere ninguna lógica de autorización. PEPs, a diferencia de PDPs esto, no se puede centralizar en una aplicación SaaS. Esto se debe a que es necesario implementar la autorización y el control de acceso en toda la aplicación y sus puntos de acceso. PEPs se puede aplicar a los microservicios APIs, a las capas de backend para frontend (BFF) o a cualquier punto de la aplicación en el que se desee o requiera el control de acceso. PEPs La integración en una aplicación garantiza que la autorización se verifique con frecuencia y de forma independiente en varios puntos.

Para implementar un PEP, el primer paso es determinar dónde debe aplicarse el control de acceso en una aplicación. Tenga en cuenta este principio a la hora de decidir dónde PEPs debe integrarse en su aplicación:

Si una aplicación expone una API, debe haber autorización y control de acceso en esa API.

Esto se debe a que en una arquitectura orientada a microservicios u orientada a servicios, APIs sirven como separadores entre las diferentes funciones de la aplicación. Tiene sentido incluir el control de acceso como puntos de control lógicos entre las funciones de la aplicación. Le recomendamos encarecidamente que lo incluya PEPs como requisito previo para acceder a cada API en una aplicación SaaS. También es posible integrar la autorización en otros puntos de una aplicación. En aplicaciones monolíticas, puede ser necesario PEPs integrarlas dentro de la lógica de la propia aplicación. No hay una ubicación única en la que PEPs deba incluirse, pero considere utilizar el principio de la API como punto de partida.

Solicitar una decisión de autorización

El PEP debe solicitar una decisión de autorización al PDP. La solicitud puede adoptar varias formas. El método más fácil y accesible para solicitar una decisión de autorización es enviar una solicitud o consulta de autorización a una RESTful API expuesta por el PDP (OPA o permisos verificados). Si utilizas permisos verificados, también puedes utilizar el IsAuthorizedmétodo mediante el AWS SDK para recuperar una decisión de autorización. La única función de un PEP en este patrón es reenviar la información que necesita la solicitud o consulta de autorización. Esto puede ser tan simple como

reenviar una solicitud recibida por una API como entrada al PDP. Existen otros métodos de creación. PEPs Por ejemplo, puede integrar un PDP de OPA de forma local con una aplicación escrita en el lenguaje de programación Go como biblioteca en lugar de utilizar una API.

Evaluar una decisión de autorización

PEPs es necesario incluir una lógica para evaluar los resultados de una decisión de autorización. Cuando PDPs se exponen como APIs, es probable que la decisión de autorización esté en formato JSON y se devuelva mediante una llamada a la API. El PEP debe evaluar este código JSON para determinar si la acción que se está realizando está autorizada. Por ejemplo, si un PDP está diseñado para proporcionar una decisión booleana de permitir o denegar la autorización, el PEP podría simplemente comprobar este valor y, a continuación, devolver el código de estado HTTP 200 para permitir o denegar. Este patrón de incorporar un PEP como requisito previo para acceder a una API es un patrón fácil de implementar y muy eficaz para implementar el control de acceso en una aplicación SaaS. En escenarios más complicados, el PEP podría ser responsable de evaluar el código JSON arbitrario devuelto por el PDP. El PEP debe escribirse de manera que incluya la lógica necesaria para interpretar la decisión de autorización que devuelve el PDP. Como es probable que un PEP se implemente en muchos lugares diferentes de la aplicación, le recomendamos que empaquete el código PEP como una biblioteca o artefacto reutilizable en el lenguaje de programación que prefiera. De esta forma, el PEP se puede integrar fácilmente en cualquier punto de la aplicación con un mínimo de reelaboración.

Diseñe modelos para arquitecturas SaaS multiusuario

Hay muchas maneras de implementar el control de acceso y la autorización de la API. Esta guía se centra en tres modelos de diseño que son eficaces para arquitecturas SaaS multiusuario. Estos diseños sirven como referencia de alto nivel para la implementación de los puntos de decisión política (PDPs) y los puntos de aplicación de las políticas (PEPs), a fin de formar un modelo de autorización coherente y ubicuo para las solicitudes.

Modelos de diseño:

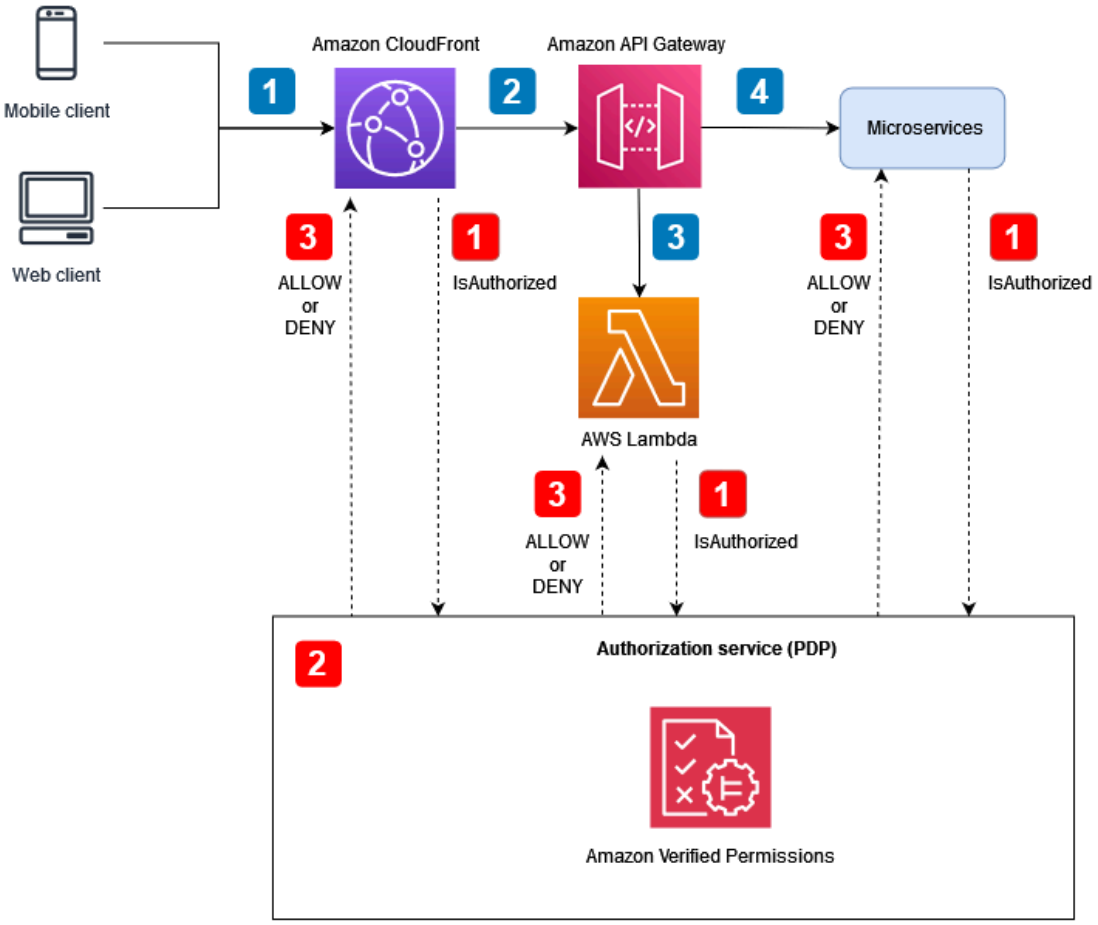
- Modelos de diseño para los permisos verificados de Amazon
- Diseñe modelos para OPA

Modelos de diseño para los permisos verificados de Amazon

Uso de un PDP centralizado con PEPs APIs

El APIs modelo de punto de decisión de políticas (PDP) centralizado con puntos de aplicación de políticas (PEPs) sigue las mejores prácticas del sector para crear un sistema eficaz y de fácil mantenimiento para el control de acceso y la autorización de las API. Este enfoque se basa en varios principios clave:

- La autorización y el control de acceso a la API se aplican en varios puntos de la aplicación.
- La lógica de autorización es independiente de la aplicación.
- Las decisiones de control de acceso están centralizadas.



Flujo de la aplicación (ilustrado con rótulos numerados en azul en el diagrama):

1. Un usuario autenticado con un token web JSON (JWT) genera una solicitud HTTP a Amazon. CloudFront
2. CloudFront reenvía la solicitud a Amazon API Gateway, que está configurado como CloudFront origen.
3. Se llama a un autorizador personalizado de API Gateway para verificar el JWT.
4. Los microservicios responden a la solicitud.

Flujo de autorización y control de acceso a la API (ilustrado con rótulos numerados en rojo en el diagrama):

1. El PEP llama al servicio de autorización y pasa los datos de la solicitud, incluidos los datos. JWTs

2. El servicio de autorización (PDP), en este caso Verified Permissions, utiliza los datos de la solicitud como entrada de la consulta y los evalúa en función de las políticas pertinentes especificadas en la consulta.
3. La decisión de autorización se devuelve al PEP y se evalúa.

Este modelo utiliza un PDP centralizado para tomar decisiones de autorización. PEPs se implementan en diferentes puntos para realizar solicitudes de autorización al PDP. El siguiente diagrama muestra cómo se puede implementar este modelo en una hipotética aplicación SaaS multiusuario.

En esta arquitectura, PEPs solicite las decisiones de autorización en los puntos de enlace del servicio para Amazon CloudFront y Amazon API Gateway y para cada microservicio. La decisión de autorización la toma el servicio de autorización, Amazon Verified Permissions (el PDP). Dado que Verified Permissions es un servicio totalmente gestionado, no es necesario que gestione la infraestructura subyacente. Puede interactuar con los permisos verificados mediante una RESTful API o el AWS SDK.

También puede utilizar esta arquitectura con motores de políticas personalizados. Sin embargo, cualquier ventaja que se obtenga con los permisos verificados debe sustituirse por la lógica proporcionada por el motor de políticas personalizado.

Un PDP centralizado con PEPs on APIs proporciona una opción sencilla para crear un sistema de autorización sólido. APIs Esto simplifica el proceso de autorización y también proporciona una easy-to-use interfaz repetible para tomar decisiones de autorización para microservicios APIs, capas de backend for Frontend (BFF) u otros componentes de la aplicación.

Uso del SDK de Cedar

Amazon Verified Permissions utiliza el lenguaje Cedar para gestionar los permisos detallados en sus aplicaciones personalizadas. Con los permisos verificados, puede almacenar las políticas de Cedar en una ubicación central, aprovechar la baja latencia con un procesamiento de milisegundos y auditar los permisos en diferentes aplicaciones. Si lo desea, también puede integrar el SDK de Cedar directamente en su aplicación para tomar decisiones de autorización sin utilizar permisos verificados. Esta opción requiere el desarrollo adicional de aplicaciones personalizadas para administrar y almacenar las políticas para su caso de uso. Sin embargo, puede ser una alternativa viable, especialmente en los casos en que el acceso a los permisos verificados es intermitente o no es posible debido a una conectividad a Internet incoherente.

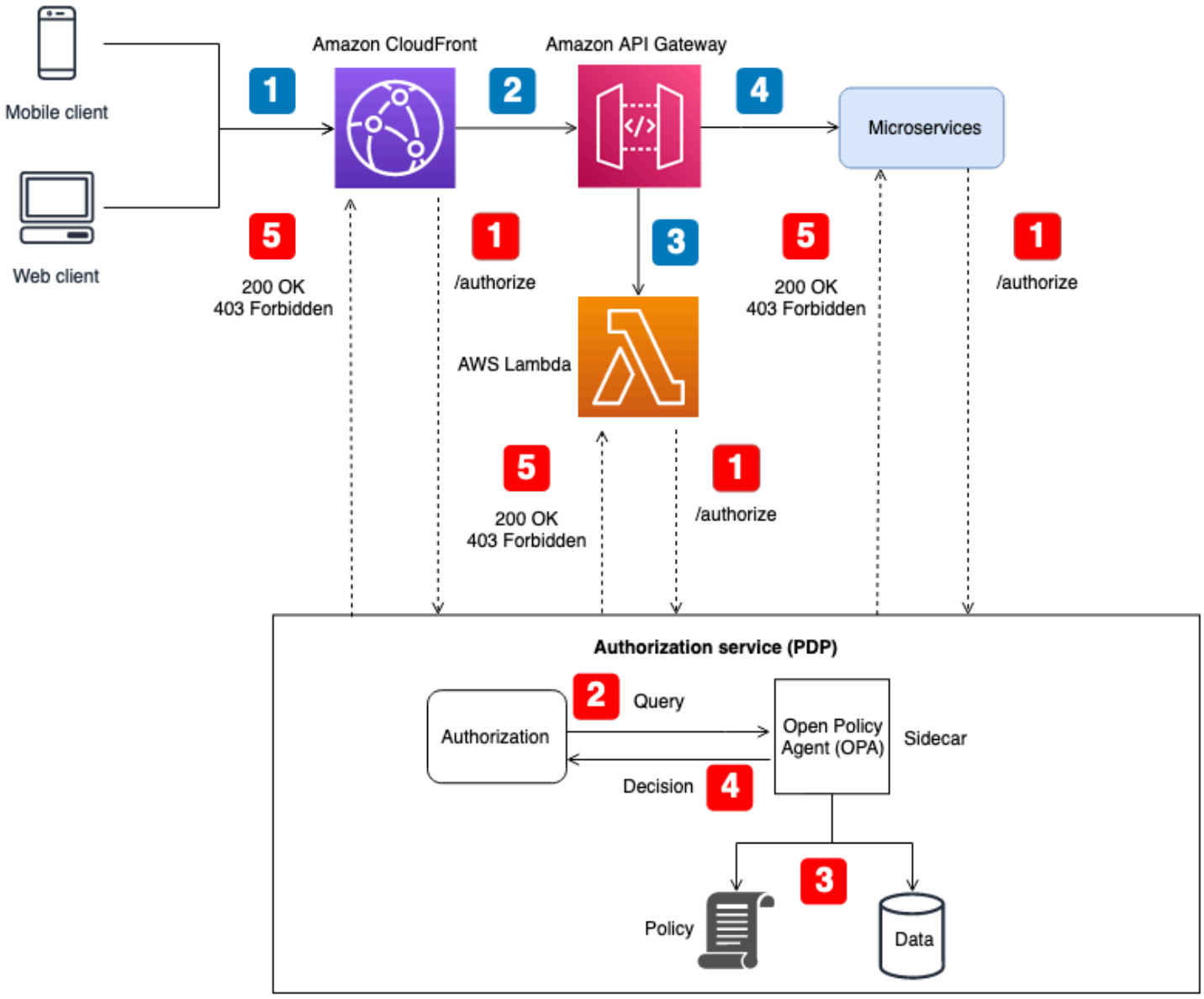
Diseñe modelos para OPA

Uso de un PDP centralizado con PEPs APIs

El APIs modelo de punto de decisión de políticas (PDP) centralizado con puntos de aplicación de políticas (PEPs) sigue las mejores prácticas del sector para crear un sistema eficaz y de fácil mantenimiento para el control de acceso y la autorización de las API. Este enfoque se basa en varios principios clave:

- La autorización y el control de acceso a la API se aplican en varios puntos de la aplicación.
- La lógica de autorización es independiente de la aplicación.
- Las decisiones de control de acceso están centralizadas.

Este modelo utiliza un PDP centralizado para tomar decisiones de autorización. PEPs se implementan en todos los APIs casos para realizar solicitudes de autorización al PDP. El siguiente diagrama muestra cómo se puede implementar este modelo en una hipotética aplicación SaaS multiusuario.



Flujo de la aplicación (ilustrado con rótulos numerados en azul en el diagrama):

1. Un usuario autenticado con un JWT genera una solicitud HTTP a Amazon. CloudFront
2. CloudFront reenvía la solicitud a Amazon API Gateway, que está configurado como CloudFront origen.
3. Se llama a un autorizador personalizado de API Gateway para verificar el JWT.
4. Los microservicios responden a la solicitud.

Flujo de autorización y control de acceso a la API (ilustrado con rótulos numerados en rojo en el diagrama):

1. El PEP llama al servicio de autorización y pasa los datos de la solicitud, incluidos los datos. JWTs
2. El servicio de autorización (PDP) toma los datos de la solicitud y consulta la API REST de un agente de OPA, que funciona como un sidecar. Los datos de la solicitud sirven como entrada para la consulta.
3. La OPA evalúa la entrada en función de las políticas relevantes especificadas en la consulta. Los datos se importan para tomar una decisión de autorización si es necesario.
4. La OPA devuelve la decisión al servicio de autorización.
5. La decisión de autorización se devuelve al PEP y se evalúa.

En esta arquitectura, PEPs solicite las decisiones de autorización en los puntos de enlace del servicio para Amazon CloudFront y Amazon API Gateway, y para cada microservicio. La decisión de autorización la toma un servicio de autorización (el PDP) con un sidecar OPA. Puede utilizar este servicio de autorización como un contenedor o como una instancia de servidor tradicional. El sidecar de OPA expone su RESTful API de forma local, por lo que solo el servicio de autorización puede acceder a ella. El servicio de autorización expone una API independiente que está disponible para. PEPs Hacer que el servicio de autorización actúe como intermediario entre PEPs una OPA permite insertar cualquier lógica de transformación entre PEPs una OPA que pueda ser necesaria, por ejemplo, cuando la solicitud de autorización de un PEP no se ajusta a la entrada de consulta esperada por la OPA.

También puede usar esta arquitectura con motores de políticas personalizados. Sin embargo, cualquier ventaja que se obtenga con la OPA debe sustituirse por la lógica proporcionada por el motor de políticas personalizado.

Un PDP centralizado con PEPs on APIs proporciona una opción sencilla para crear un sistema de autorización sólido. APIs Es fácil de implementar y también proporciona una easy-to-use interfaz repetible para tomar decisiones de autorización para microservicios APIs, capas de backend for Frontend (BFF) u otros componentes de la aplicación. Sin embargo, este enfoque puede generar una latencia excesiva en la aplicación, ya que las decisiones de autorización requieren llamar a una API independiente. Si la latencia de la red es un problema, podría considerar la posibilidad de utilizar un PDP distribuido.

Uso de un PDP distribuido con PEPs APIs

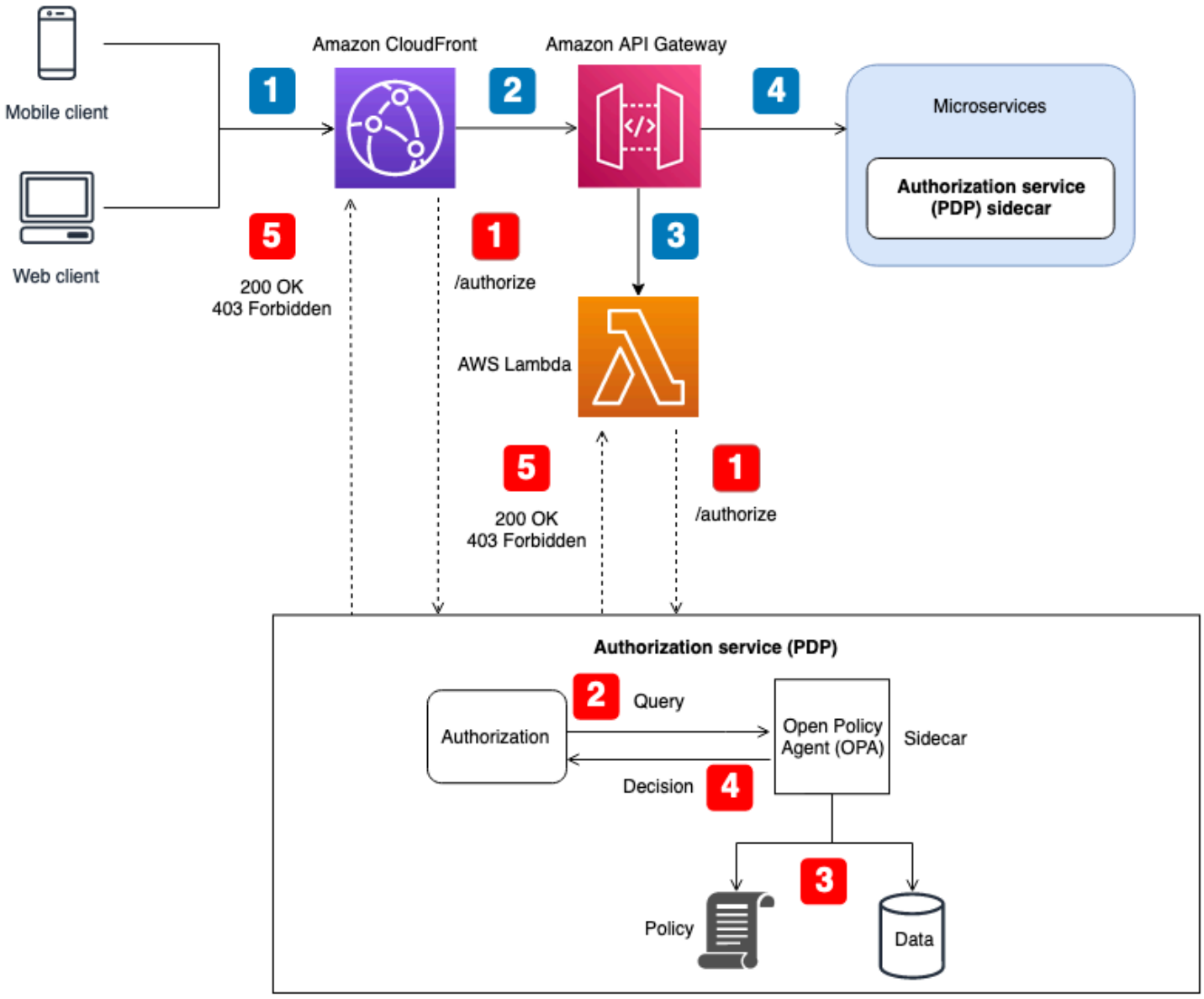
El APIs modelo de punto de decisión de políticas distribuido (PDP) con puntos de aplicación de políticas (PEPs) sigue las mejores prácticas del sector para crear un sistema eficaz de control y autorización del acceso a las API. Al igual que con el PDP centralizado PEPs sin APIs modelo, este enfoque apoya los siguientes principios clave:

- La autorización y el control de acceso a la API se aplican en varios puntos de la aplicación.
- La lógica de autorización es independiente de la aplicación.
- Las decisiones de control de acceso están centralizadas.

Quizás se pregunte por qué las decisiones de control de acceso están centralizadas cuando se distribuye el PDP. Si bien el PDP puede estar en varios lugares de una aplicación, debe usar la misma lógica de autorización para tomar decisiones de control de acceso. Todos PDPs proporcionan las mismas decisiones de control de acceso con las mismas entradas. PEPs se implementan en todos los APIs casos para realizar solicitudes de autorización al PDP. La siguiente figura muestra cómo se puede implementar este modelo distribuido en una hipotética aplicación SaaS multiusuario.

En este enfoque, los PDP se implementan en varios lugares de la aplicación. En el caso de los componentes de la aplicación que tienen capacidades informáticas integradas que pueden ejecutar OPA y admitir un PDP, como un servicio en contenedores con un sidecar o una instancia de Amazon Elastic Compute Cloud (Amazon EC2), las decisiones de PDP se pueden integrar directamente en el componente de la aplicación sin tener que realizar RESTful una llamada de API a un servicio PDP centralizado. Esto tiene la ventaja de reducir la latencia que se puede encontrar en el modelo PDP centralizado, ya que no todos los componentes de la aplicación tienen que realizar llamadas a la API adicionales para obtener decisiones de autorización. Sin embargo, un PDP centralizado sigue siendo necesario en este modelo para los componentes de la aplicación que no tienen capacidades informáticas integradas que permitan la integración directa de un PDP, como el servicio Amazon o Amazon API CloudFront Gateway.

El siguiente diagrama muestra cómo se puede implementar esta combinación de un PDP centralizado y un PDP distribuido en una hipotética aplicación SaaS multiusuario.



Flujo de la aplicación (ilustrado con rótulos numerados en azul en el diagrama):

1. Un usuario autenticado con un JWT genera una solicitud HTTP a Amazon. CloudFront
2. CloudFront reenvía la solicitud a Amazon API Gateway, que está configurado como CloudFront origen.
3. Se llama a un autorizador personalizado de API Gateway para verificar el JWT.
4. Los microservicios responden a la solicitud.

Flujo de autorización y control de acceso a la API (ilustrado con rótulos numerados en rojo en el diagrama):

1. El PEP llama al servicio de autorización y pasa los datos de la solicitud, incluidos los datos. JWTs
2. El servicio de autorización (PDP) toma los datos de la solicitud y consulta la API REST de un agente de OPA, que funciona como un sidecar. Los datos de la solicitud sirven como entrada para la consulta.
3. La OPA evalúa la entrada en función de las políticas relevantes especificadas en la consulta. Los datos se importan para tomar una decisión de autorización si es necesario.
4. La OPA devuelve la decisión al servicio de autorización.
5. La decisión de autorización se devuelve al PEP y se evalúa.

En esta arquitectura, los PEP solicitan decisiones de autorización en los puntos finales del servicio para API Gateway CloudFront y para cada microservicio. La decisión de autorización de los microservicios la toma un servicio de autorización (el PDP) que funciona como un sidecar con el componente de aplicación. Este modelo es posible para microservicios (o servicios) que se ejecutan en contenedores o instancias de Amazon Elastic Compute Cloud (Amazon EC2). Las decisiones de autorización para servicios como API Gateway y, aun así, CloudFront requerirían ponerse en contacto con un servicio de autorización externo. En cualquier caso, el servicio de autorización expone una API que está disponible para PEPs. Hacer que el servicio de autorización actúe como intermediario entre PEPs una OPA permite insertar cualquier lógica de transformación entre PEPs una OPA que pueda ser necesaria, por ejemplo, cuando la solicitud de autorización de un PEP no se ajusta a la entrada de consulta esperada por la OPA.

También puede usar esta arquitectura con motores de políticas personalizados. Sin embargo, cualquier ventaja que se obtenga con la OPA debe sustituirse por la lógica proporcionada por el motor de políticas personalizado.

Un PDP distribuido con PEPs on APIs ofrece la opción de crear un sistema de autorización sólido para APIs él. Es fácil de implementar y proporciona una easy-to-use interfaz repetible para tomar decisiones de autorización para microservicios APIs, capas de backend para frontend (BFF) u otros componentes de la aplicación. Este enfoque también tiene la ventaja de reducir la latencia que puede producirse en el modelo PDP centralizado.

Uso de un PDP distribuido como biblioteca

También puede solicitar decisiones de autorización a un PDP que esté disponible como biblioteca o paquete para su uso en una aplicación. OPA se puede usar como una biblioteca de terceros de Go. En el caso de otros lenguajes de programación, la adopción de este modelo generalmente implica crear un motor de políticas personalizado.

Consideraciones de diseño para varios arrendatarios de Amazon Verified Permissions

Hay varias opciones de diseño que se deben tener en cuenta al implementar la autorización mediante Amazon Verified Permissions en una solución SaaS multiusuario. Antes de explorar estas opciones, aclaremos la diferencia entre aislamiento y autorización en un contexto de SaaS multiusuario. Al aislar a un arrendatario, se evita que los datos entrantes y salientes queden expuestos al arrendatario equivocado. La autorización garantiza que un usuario tenga los permisos para acceder a un inquilino.

En los permisos verificados, las políticas se almacenan en un almacén de políticas. Como se describe en la documentación de permisos verificados, puede aislar las políticas de los inquilinos mediante un almacén de políticas independiente para cada inquilino o permitir que los inquilinos compartan las políticas mediante un único almacén de políticas para todos los inquilinos. En esta sección se analizan las ventajas y desventajas de estas dos estrategias de aislamiento y se describe cómo se pueden implementar mediante un modelo de implementación por niveles. Para obtener más información sobre el contexto, consulte la documentación sobre permisos verificados.

Si bien los criterios analizados en esta sección se centran en los permisos verificados, los conceptos generales se basan en la mentalidad de aislamiento y en la orientación que proporciona. Las aplicaciones SaaS siempre deben considerar el aislamiento de los inquilinos como parte de su diseño, y este principio general de aislamiento se extiende a la inclusión de permisos verificados en una aplicación SaaS. Esta sección también hace referencia a los principales modelos de aislamiento de SaaS, como el modelo SaaS en silos y el modelo SaaS agrupado. Para obtener información adicional, consulte los conceptos básicos de aislamiento en AWS Well-Architected Framework, SaaS Lens.

Las consideraciones clave a la hora de diseñar soluciones SaaS multiusuario son el aislamiento y la incorporación de inquilinos. El aislamiento de los inquilinos afecta a la seguridad, la privacidad, la resiliencia y el rendimiento. La incorporación de inquilinos afecta a sus procesos operativos en lo que respecta a los gastos operativos y a la observabilidad. Las organizaciones que optan por el SaaS o que implementan soluciones multiusuario siempre deben priorizar la forma en que la aplicación SaaS gestionará el arrendamiento. Si bien una solución SaaS puede inclinarse hacia un modelo de aislamiento en particular, no es necesaria la coherencia en toda la solución SaaS. Por ejemplo, es posible que el modelo de aislamiento que elija para los componentes frontend de su

aplicación no sea el mismo que el modelo de aislamiento que elija para un microservicio o un servicio de autorización.

Consideraciones de diseño:

- Incorporación de inquilinos y registro de usuarios/inquilinos
- Almacén de políticas por inquilino
- Un almacén de políticas compartido con varios inquilinos
- Modelo de despliegue por niveles

Incorporación de inquilinos y registro de usuarios/inquilinos

Las aplicaciones SaaS respetan el concepto de identidades SaaS y siguen la mejor práctica general de vincular la identidad de un usuario con la identidad de un inquilino. La vinculación implica almacenar un identificador de inquilino como afirmación o atributo del usuario en el proveedor de identidad. Esto transfiere la responsabilidad de asignar las identidades a los inquilinos de cada aplicación al proceso de registro de usuarios. Cada usuario autenticado tiene entonces la identidad de inquilino correcta como parte del JSON Web Token (JWT).

Del mismo modo, la lógica de la aplicación no debe determinar la selección del almacén de políticas correcto para una solicitud de autorización. Para determinar qué almacén de políticas debe utilizar una solicitud de autorización concreta, mantenga un mapeo de los usuarios a los almacenes de políticas o de los inquilinos a los almacenes de políticas. Por lo general, estas asignaciones se mantienen en un almacén de datos como Amazon DynamoDB o Amazon Relational Database Service (Amazon RDS) al que hace referencia la aplicación. También puede proporcionar o complementar estas asignaciones mediante datos de un proveedor de identidad (IdP). Por lo general, la relación entre los inquilinos, los usuarios y los almacenes de políticas se proporciona al usuario a través de un JWT que contiene todas las relaciones necesarias para una solicitud de autorización.

En este ejemplo, se muestra cómo podría aparecer el JWT para el usuario`Alice`, que pertenece al inquilino `TenantA` y utiliza el almacén de políticas con el ID del almacén de políticas `ps-43214321` para la autorización.

```
{
   "sub":"1234567890",
   "name":"Alice",
   "tenant":"TenantA",
   "policyStoreId":"ps-43214321"
```

```
}
```

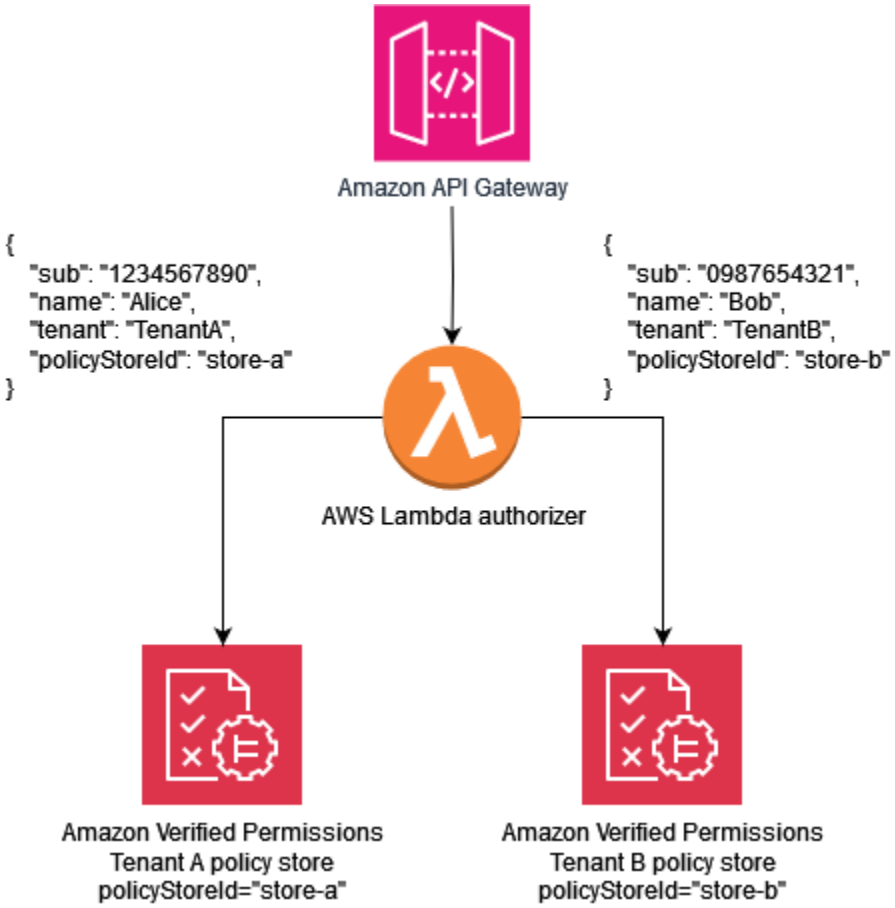
Almacén de políticas por inquilino

El modelo de diseño de almacén de políticas por inquilino de Amazon Verified Permissions asocia a cada inquilino de una aplicación SaaS a su propio almacén de políticas. Este modelo es similar al modelo de aislamiento de silos SaaS. Ambos modelos exigen la creación de una infraestructura específica para cada inquilino y tienen ventajas y desventajas similares. Las principales ventajas de este enfoque son el aislamiento de los arrendatarios que impone la infraestructura, la compatibilidad con modelos de autorización únicos para cada inquilino, la eliminación de los problemas de los vecinos ruidosos y la reducción del alcance del impacto en caso de que no se produzca una actualización o implementación de las políticas. Las desventajas de este enfoque incluyen procesos, despliegues y operaciones de incorporación de inquilinos más complejos. El enfoque recomendado es almacenar políticas por inquilino si la solución tiene políticas únicas por inquilino.

El modelo de almacén de políticas por inquilino puede proporcionar un enfoque altamente aislado para el aislamiento de los inquilinos, si su aplicación SaaS lo requiere. También puedes usar este modelo con el aislamiento de grupos, pero tu implementación de permisos verificados no compartirá los beneficios estándar del modelo de aislamiento de grupos más amplio, como la simplificación de la administración y las operaciones.

En un almacén de políticas por inquilino, el aislamiento del inquilino se logra mapeando el identificador del almacén de políticas del inquilino con la identidad SaaS del usuario durante el proceso de registro del usuario, como se ha descrito anteriormente. Este enfoque vincula estrechamente el almacén de políticas del arrendatario con el usuario principal y proporciona una forma coherente de compartir el mapeo en toda la solución SaaS. Puede proporcionar el mapeo a una aplicación SaaS manteniéndolo como parte de un IdP o en una fuente de datos externa, como DynamoDB. Esto también garantiza que el principal forme parte del arrendatario y que se utilice el almacén de políticas del arrendatario.

En este ejemplo, se muestra cómo el JWT que contiene la `tenant` dirección `policyStoreId` AND de un usuario pasa del punto final de la API al punto de evaluación de políticas de un AWS Lambda autorizador, que enruta la solicitud al almacén de políticas correcto.



El siguiente ejemplo de política ilustra el paradigma de diseño del almacén de políticas por inquilino. El usuario que `Alice` pertenece a `TenantA.` The también policyStoreId `store-a` está asignado a la identidad del inquilino del `Alice,` almacén de políticas correcto y exige su uso. Esto garantiza que se utilicen las políticas de`TenantA`.

> **Note**
>
> El modelo de almacén de políticas por inquilino aísla las políticas de los inquilinos. La autorización impone las acciones que los usuarios pueden realizar con sus datos. Los recursos involucrados en cualquier aplicación hipotética que utilice este modelo deben aislarse mediante el uso de otros mecanismos de aislamiento, tal como se define en la documentación de AWS Well-Architected Framework, SaaS Lens.

En esta política, `Alice` tiene permisos para ver los datos de todos los recursos.

```
permit (
```

```
    principal == MultiTenantApp::User::"Alice",
    action == MultiTenantApp::Action::"viewData",
    resource
);
```

Para realizar una solicitud de autorización e iniciar una evaluación con una política de permisos verificados, debe proporcionar el ID del almacén de políticas que corresponda al identificador único asignado al inquilino. `store-a`

```
{
   "policyStoreId":"store-a",
   "principal":{
      "entityType":"MultiTenantApp::User",
      "entityId":"Alice"
   },
   "action":{
      "actionType":"MultiTenantApp::Action",
      "actionId":"viewData"
   },
   "resource":{
      "entityType":"MultiTenantApp::Data",
      "entityId":"my_example_data"
   },
   "entities":{
      "entityList":[
         [
            {
               "identifier":{
                  "entityType":"MultiTenantApp::User",
                  "entityId":"Alice"
               },
               "attributes":{},
               "parents":[]
            },
            {
               "identifier":{
                  "entityType":"MultiTenantApp::Data",
                  "entityId":"my_example_data"
               },
               "attributes":{},
               "parents":[]
            }
```

```
         ]
      ]
   }
}
```

El usuario Bob pertenece al arrendatario B y también policyStoreId `store-b` está asignado a la identidad del arrendatario deBob, lo que exige el uso del almacén de políticas correcto. Esto garantiza que se utilicen las políticas del inquilino B.

En esta política, Bob tiene permisos para personalizar los datos de todos los recursos. En este ejemplo, `customizeData` puede tratarse de una acción que sea específica únicamente del arrendatario B, por lo que la política sería exclusiva para el arrendatario B. El modelo de almacén de políticas por inquilino admite de forma inherente políticas personalizadas para cada inquilino.

```
permit (
    principal == MultiTenantApp::User::"Bob",
    action == MultiTenantApp::Action::"customizeData",
    resource
);
```

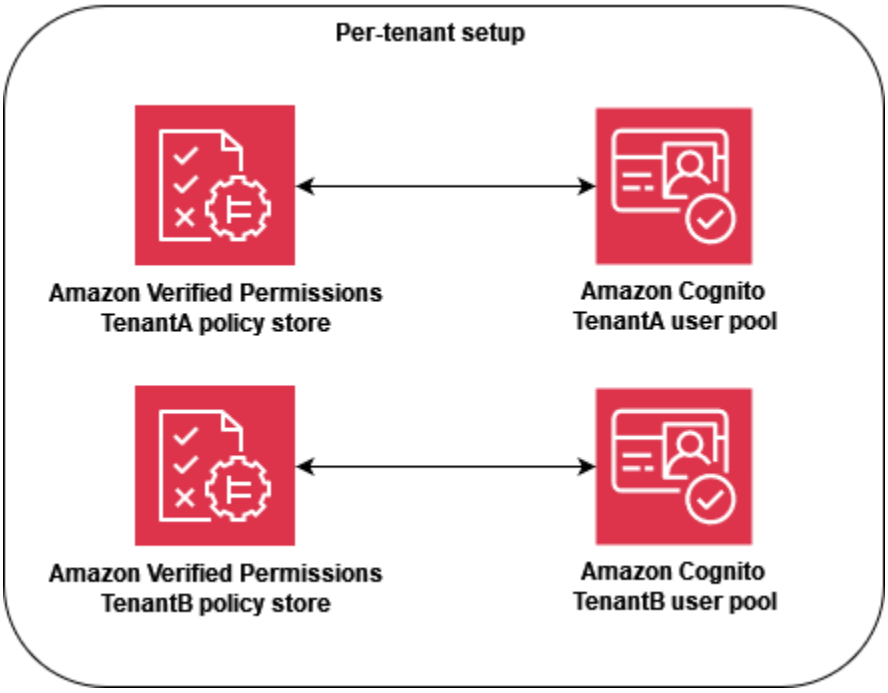
Para realizar una solicitud de autorización e iniciar una evaluación con una política de permisos verificados, debe proporcionar el ID del almacén de políticas que corresponda al ID único asignado al inquilino. `store-b`

```
{
   "policyStoreId":"store-b",
   "principal":{
      "entityType":"MultiTenantApp::User",
      "entityId":"Bob"
   },
   "action":{
      "actionType":"MultiTenantApp::Action",
      "actionId":"customizeData"
   },
   "resource":{
      "entityType":"MultiTenantApp::Data",
      "entityId":"my_example_data"
   },
   "entities":{
      "entityList":[
         [
```

```
            {
               "identifier":{
                  "entityType":"MultiTenantApp::User",
                  "entityId":"Bob"
               },
               "attributes":{},
               "parents":[]
            },
            {
               "identifier":{
                  "entityType":"MultiTenantApp::Data",
                  "entityId":"my_example_data"
               },
               "attributes":{},
               "parents":[]
            }
         ]
      ]
   }
}
```

Con los permisos verificados, es posible, pero no obligatorio, integrar un IdP con un almacén de políticas. Esta integración permite que las políticas hagan referencia explícita al principal del almacén de identidades como el principal de las políticas. Para obtener más información sobre cómo integrarse con Amazon Cognito como un IdP para permisos verificados, consulte la documentación de permisos verificados y la documentación de Amazon Cognito.

Al integrar un almacén de políticas con un IdP, solo puede usar una fuente de identidad por almacén de políticas. Por ejemplo, si decide integrar los permisos verificados con Amazon Cognito, tendrá que reflejar la estrategia utilizada para aislar a los inquilinos de los almacenes de políticas de permisos verificados y los grupos de usuarios de Amazon Cognito. Los almacenes de políticas y los grupos de usuarios también deben estar en el mismo lugar. Cuenta de AWS



A nivel operativo, el almacén de políticas por inquilino tiene una ventaja de auditoría, ya que puede consultar fácilmente la actividad registrada de AWS CloudTrail forma independiente para cada inquilino. Sin embargo, te recomendamos que registres métricas personalizadas adicionales en una dimensión por inquilino en Amazon CloudWatch.

El enfoque de almacén de políticas por inquilino también requiere prestar mucha atención a dos cuotas de permisos verificados para garantizar que no interfieran con las operaciones de su solución SaaS. Estas cuotas son almacenes de políticas por región y cuenta y IsAuthorized solicitudes por segundo por región y cuenta. Puedes solicitar aumentos para ambas cuotas.

Para ver un ejemplo más detallado de cómo implementar el modelo de almacén de políticas por inquilino, consulte la entrada del AWS blog Control de acceso SaaS mediante permisos verificados de Amazon con un almacén de políticas por inquilino.

Un almacén de políticas compartido con varios inquilinos

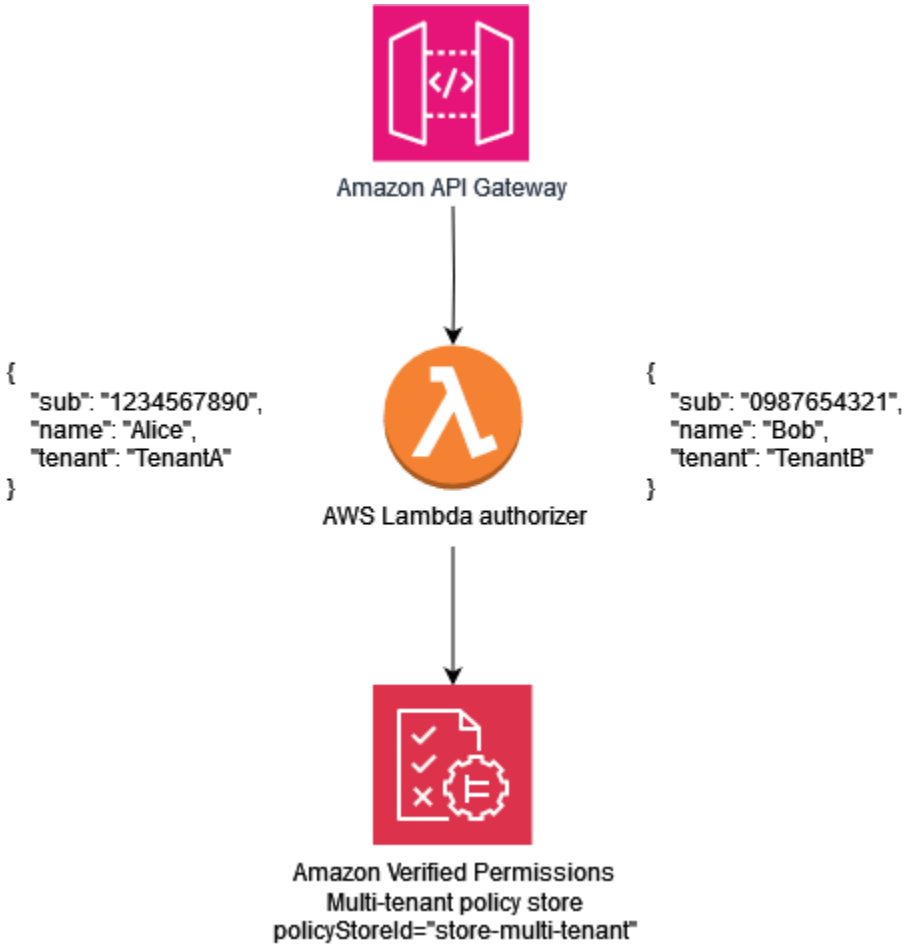
El modelo de diseño de un almacén de políticas compartido con varios inquilinos utiliza un único almacén de políticas con varios inquilinos en Amazon Verified Permissions para todos los inquilinos de la solución SaaS. La principal ventaja de este enfoque es la simplificación de la administración y las operaciones, sobre todo porque no es necesario crear almacenes de pólizas adicionales durante la incorporación de los inquilinos. Las desventajas de este enfoque incluyen un mayor alcance

del impacto derivado de cualquier fallo o error en las actualizaciones o implementaciones de las políticas, y una mayor exposición a los efectos de ruido en los vecinos. Además, no recomendamos este enfoque si su solución requiere políticas únicas para cada inquilino. En este caso, utilice en su lugar el modelo de almacén de políticas por inquilino para garantizar que se utilicen las políticas del inquilino correcto.

El enfoque de un almacén de políticas compartido y multiusuario es similar al modelo de aislamiento agrupado de SaaS. Puede proporcionar un enfoque agrupado para el aislamiento de los inquilinos, si su aplicación SaaS lo requiere. También puede usar este modelo si su solución SaaS aplica aislamiento en silos a sus microservicios. Al elegir un modelo, debe evaluar los requisitos para el aislamiento de los datos de los inquilinos y la estructura de las políticas de permisos verificados que son necesarias para una aplicación SaaS de forma independiente.

Para garantizar una forma coherente de compartir el identificador de arrendatario en toda la solución SaaS, se recomienda asignar el identificador a la identidad de SaaS del usuario durante el registro del usuario, tal y como se ha explicado anteriormente. Puede proporcionar este mapeo a una aplicación SaaS manteniéndolo como parte de un IdP o en una fuente de datos externa, como DynamoDB. También se recomienda asignar el ID del almacén de políticas compartido a los usuarios. Si bien la identificación no se usa como parte del aislamiento de los inquilinos, es una buena práctica porque facilita los cambios futuros.

En el siguiente ejemplo, se muestra cómo el punto final de la API envía un JWT para los usuarios `Alice` y los usuarios que pertenecen a arrendatarios diferentesBob, pero que comparten el almacén de políticas con el ID del almacén de políticas `store-multi-tenant` para su autorización. Como todos los inquilinos comparten un único almacén de políticas, no es necesario mantener el ID del almacén de políticas en un token o una base de datos. Como todos los inquilinos comparten un único ID de almacén de políticas, puede proporcionarlo como una variable de entorno que la aplicación puede usar para realizar llamadas al almacén de políticas.



El siguiente ejemplo de política ilustra el paradigma de diseño de políticas multiusuario compartido. En esta política, el director `MultiTenantApp::User` que tiene la matriz `MultiTenantApp::Role` `Admin` tiene permisos para ver los datos de todos los recursos.

```
permit (
    principal in MultiTenantApp::Role::"Admin",
    action == MultiTenantApp::Action::"viewData",
    resource
);
```

Como se utiliza un único almacén de políticas, el almacén de políticas de permisos verificados debe garantizar que el atributo de arrendamiento asociado al principal coincida con el atributo de arrendamiento asociado al recurso. Esto se puede lograr mediante la inclusión de la siguiente política en el almacén de políticas, para garantizar que se rechacen todas las solicitudes de autorización que no tengan atributos de arrendamiento coincidentes en el recurso y el principal.

```
forbid(
```

```
    principal,
    action,
    resource
)
unless {
    resource.Tenant == principal.Tenant
};
```

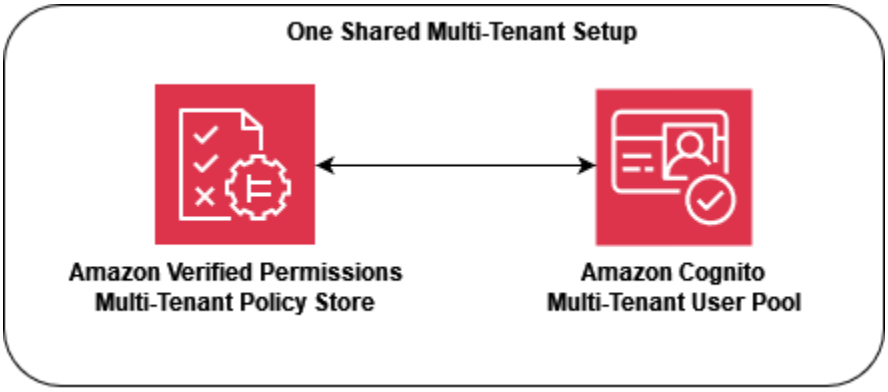
En el caso de una solicitud de autorización que utiliza un modelo de almacén de políticas compartido y multiusuario, el ID del almacén de políticas es el identificador del almacén de políticas compartido. En la siguiente solicitud, `User Alice` se le permite el acceso porque tiene un `Role` de `Admin` y los `Tenant` atributos asociados al recurso y al principal son ambos`TenantA`.

```
{
   "policyStoreId":"store-multi-tenant",
   "principal":{
      "entityType":"MultiTenantApp::User",
      "entityId":"Alice"
   },
   "action":{
      "actionType":"MultiTenantApp::Action",
      "actionId":"viewData"
   },
   "resource":{
      "entityType":"MultiTenantApp::Data",
      "entityId":"my_example_data"
   },
   "entities":{
      "entityList":[
         {
            "identifier":{
               "entityType":"MultiTenantApp::User",
               "entityId":"Alice"
            },
            "attributes": {
                {
                    "Tenant": {
                        "entityIdentifier": {
                            "entityType":"MultitenantApp::Tenant",
                            "entityId":"TenantA"
                        }
                    }
```

```
                }
            },
            "parents":[
               {
                  "entityType":"MultiTenantApp::Role",
                  "entityId":"Admin"
               }
            ]
         },
         {
            "identifier":{
               "entityType":"MultiTenantApp::Data",
               "entityId":"my_example_data"
            },
            "attributes": {
                {
                    "Tenant": {
                        "entityIdentifier": {
                            "entityType":"MultitenantApp::Tenant",
                            "entityId":"TenantA"
                        }
                    }
                }
            },
            "parents":[]
         }
      ]
   }
}
```

Con los permisos verificados, es posible, pero no obligatorio, integrar un IdP con un almacén de políticas. Esta integración permite que las políticas hagan referencia explícita al principal del almacén de identidades como el principal de las políticas. Para obtener más información sobre cómo integrarse con Amazon Cognito como un IdP para permisos verificados, consulte la documentación de permisos verificados y la documentación de Amazon Cognito.

Al integrar un almacén de políticas con un IdP, solo puede usar una fuente de identidad por almacén de políticas. Por ejemplo, si decide integrar los permisos verificados con Amazon Cognito, tendrá que reflejar la estrategia utilizada para aislar a los inquilinos de los almacenes de políticas de permisos verificados y los grupos de usuarios de Amazon Cognito. Los almacenes de políticas y los grupos de usuarios también deben estar en el mismo lugar. Cuenta de AWS



Desde el punto de vista operativo y de auditoría, el modelo de almacén de políticas compartido y multiusuario presenta la desventaja de que la actividad registrada AWS CloudTrail requiere consultas más complicadas para filtrar la actividad individual del arrendatario, ya que cada CloudTrail llamada registrada utiliza el mismo almacén de políticas. En este escenario, resulta útil registrar métricas personalizadas adicionales en una dimensión por inquilino en Amazon CloudWatch para garantizar un nivel adecuado de observabilidad y capacidad de auditoría.

El enfoque de un almacén de políticas compartido y multiusuario también requiere prestar mucha atención a las cuotas de permisos verificados para garantizar que no interfieran con las operaciones de su solución SaaS. En concreto, le recomendamos que controle la cuota de IsAuthorized solicitudes por segundo por región y cuenta para asegurarse de que no se superen sus límites. Puedes solicitar un aumento de esta cuota.

Modelo de despliegue por niveles

Al crear un modelo de implementación por niveles, puede aislar a los clientes del «nivel empresarial» de alta prioridad del volumen potencialmente mayor de clientes del «nivel estándar». En este modelo, puede implementar cualquier cambio implementado en las políticas en los almacenes de pólizas por separado para cada nivel, lo que aísla a cada nivel de clientes de los cambios realizados fuera de su nivel. En el modelo de implementación por niveles, los almacenes de políticas se crean normalmente como parte del aprovisionamiento inicial de la infraestructura para cada nivel, en lugar de implementarse cuando se incorpora un arrendatario.

Si su solución utiliza principalmente un modelo de aislamiento agrupado, es posible que necesite aislamiento o personalización adicionales. Por ejemplo, puede crear un «nivel premium» en el que cada inquilino tenga su propia infraestructura de niveles de inquilinos, lo que crea un modelo aislado al implementar una instancia agrupada con un solo inquilino. Esto podría adoptar la forma

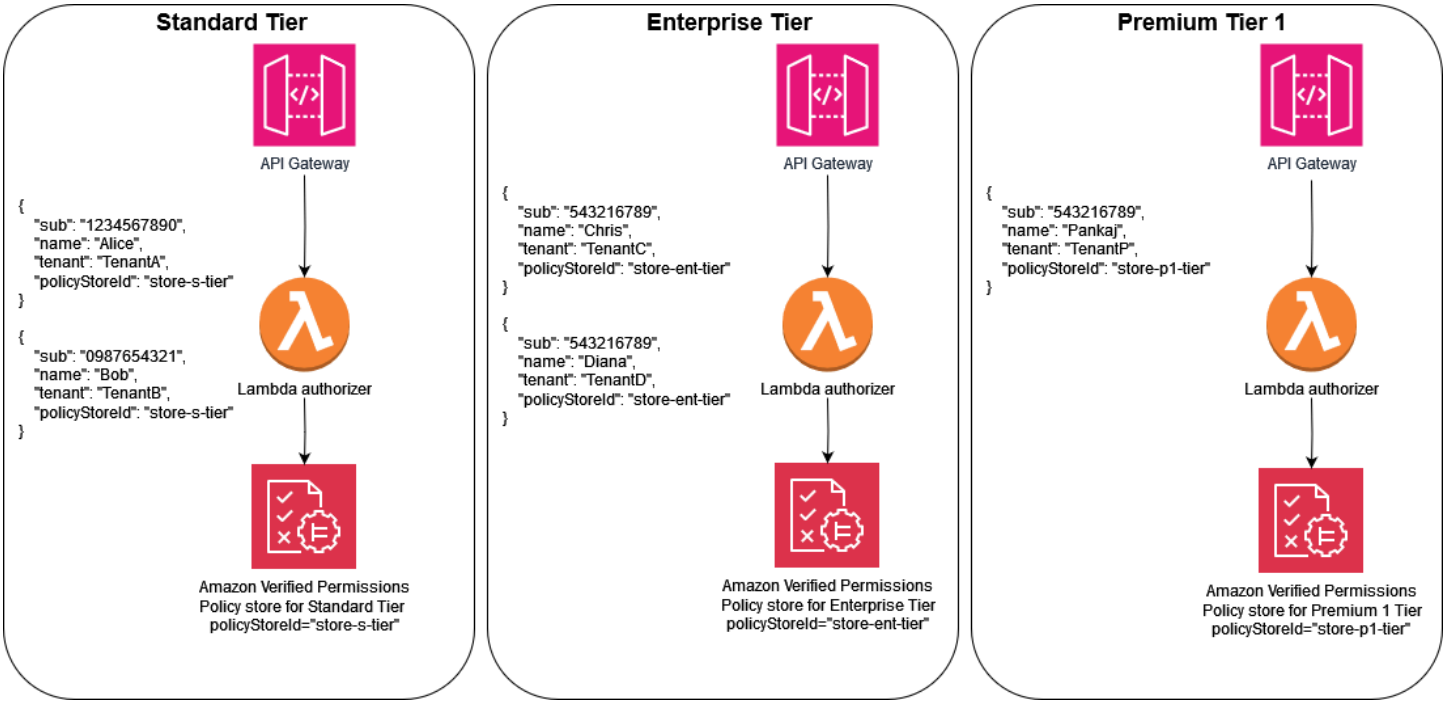
de infraestructuras de «arrendatario de nivel premium A» y «arrendatario de nivel premium B» completamente separadas, incluidos los almacenes de pólizas. Este enfoque da como resultado un modelo de aislamiento aislado para el nivel más alto de clientes.

En el modelo de implementación por niveles, cada almacén de políticas debe seguir el mismo modelo de aislamiento, aunque se implementa por separado. Como se utilizan varios almacenes de políticas, debe aplicar una forma coherente de compartir el identificador del almacén de políticas que está asociado al inquilino en toda la solución SaaS. Al igual que con el modelo de almacén de políticas por inquilino, es una buena práctica asignar el identificador del inquilino a la identidad de SaaS del usuario durante el registro del usuario.

El siguiente diagrama muestra tres niveles: `Standard Tier`, `Enterprise Tier`, y. `Premium Tier 1` Cada nivel se implementa por separado en su propia infraestructura y utiliza un almacén de políticas compartido dentro del nivel. Los niveles Estándar y Empresarial contienen varios arrendatarios. `TenantA` y `TenantB` están en el `Standard Tier` nivel y `TenantC` y `TenantD` están en el nivel empresarial.

`Premium Tier 1` solo contiene `TenantP`, por lo que puede atender al cliente principal como si la solución tuviera un modelo de aislamiento totalmente aislado y ofrecer funciones como políticas personalizadas. La incorporación de un nuevo cliente de nivel premium implicaría la creación de una infraestructura. `Premium Tier 2`

> **Note**
>
> La aplicación, el despliegue y la incorporación de inquilinos en el nivel premium son idénticos a los niveles estándar y empresarial. La única diferencia es que el flujo de trabajo de incorporación del nivel premium comienza con el aprovisionamiento de una infraestructura de nivel nuevo.


Standard Tier
API Gateway
{
"sub": "1234567890",
"name": "Alice",
"tenant": "TenantA",
"policyStoreId": "store-s-tier"
}
{
"sub": "0987654321",
"name": "Bob",
"tenant": "TenantB",
"policyStoreId": "store-s-tier"
}
Lambda authorizer
Amazon Verified Permissions
Policy store for Standard Tier
policyStoreId="store-s-tier"
Enterprise Tier
API Gateway
{
"sub": "543216789",
"name": "Chris",
"tenant": "TenantC",
"policyStoreId": "store-ent-tier"
}
{
"sub": "543216789",
"name": "Diana",
"tenant": "TenantD",
"policyStoreId": "store-ent-tier"
}
Lambda authorizer
Amazon Verified Permissions
Policy store for Enterprise Tier
policyStoreId="store-ent-tier"
Premium Tier 1
API Gateway
{
"sub": "543216789",
"name": "Pankaj",
"tenant": "TenantP",
"policyStoreId": "store-p1-tier"
}
Lambda authorizer
Amazon Verified Permissions
Policy store for Premium 1 Tier
policyStoreId="store-p1-tier"

Consideraciones de diseño de OPA para múltiples inquilinos

El Open Policy Agent (OPA) es un servicio flexible que se puede aplicar a numerosos casos de uso en los que las aplicaciones deben tomar decisiones sobre políticas y autorizaciones. El uso de OPA con aplicaciones de SaaS de múltiples inquilinos requiere la consideración de criterios únicos para garantizar que las mejores prácticas clave de SaaS, como el aislamiento de inquilinos, sigan siendo parte de la implementación de OPA. Estos criterios incluyen los patrones de despliegue de la OPA, el aislamiento de los inquilinos y el modelo de documentos de la OPA, así como la incorporación de los inquilinos. Cada uno de ellos afecta al diseño óptimo de la OPA en lo que respecta a las aplicaciones con varios usuarios.

Si bien el análisis de esta sección se centra en la OPA, los conceptos generales se basan en la mentalidad de aislamiento y en la orientación que proporciona. Las aplicaciones SaaS siempre deben considerar el aislamiento de los inquilinos como parte de su diseño, y este principio general de aislamiento se extiende a la inclusión de la OPA en una aplicación SaaS. La OPA, si se usa adecuadamente, puede ser una parte clave de la forma en que se aplica el aislamiento en las aplicaciones SaaS. Esta sección también hace referencia a los principales modelos de aislamiento de SaaS, como el modelo SaaS en silos y el modelo SaaS agrupado. Para obtener información adicional, consulte los conceptos básicos de aislamiento en AWS Well-Architected Framework, SaaS Lens.

Consideraciones de diseño:

- Comparación de los patrones de despliegue centralizados y distribuidos
- Aislamiento de inquilinos con el modelo de documentos OPA
- Incorporación de inquilinos

Comparación de los patrones de despliegue centralizados y distribuidos

Puede implementar la OPA en un patrón de implementación centralizado o distribuido, y el método ideal para una aplicación multiusuario depende del caso de uso. Para ver ejemplos de estos patrones, consulte las APIs secciones Uso de un PDP centralizado con PEPs APIs y Uso de un PDP distribuido y PEPs demás, que aparecen anteriormente en esta guía. Como OPA se puede implementar como un daemon en un sistema operativo o contenedor, se puede implementar de varias maneras para admitir una aplicación multiusuario.

En un patrón de despliegue centralizado, OPA se despliega como un contenedor o un daemon con su RESTful API disponible para otros servicios de la aplicación. Cuando un servicio requiere una decisión de la OPA, se recurre a la RESTful API central de la OPA para que tome esta decisión. Este enfoque es fácil de implementar y mantener, ya que solo hay una implementación de OPA. La desventaja de este enfoque es que no proporciona ningún mecanismo para mantener la separación de los datos de los inquilinos. Como la OPA solo se implementa una vez, todos los datos de los inquilinos que se utilicen en una decisión de la OPA, incluidos los datos externos a los que haga referencia la OPA, deben estar disponibles para la OPA. Puede mantener el aislamiento de los datos de los inquilinos con este enfoque, pero debe respetarse mediante la estructura de políticas y documentos de la OPA o mediante el acceso a datos externos. Un patrón de despliegue centralizado también requiere una latencia más alta, ya que cada decisión de autorización debe realizar una llamada a la RESTful API a otro servicio.

En un patrón de despliegue distribuido, OPA se implementa como un contenedor o un daemon junto con los servicios de la aplicación multiusuario. Se puede implementar como un contenedor lateral o como un daemon que se ejecute localmente en el sistema operativo. Para obtener una decisión de la OPA, el servicio realiza una llamada a la RESTful API a la implementación local de la OPA. (Como la OPA se puede implementar como un paquete Go, puedes usar Go de forma nativa para recuperar una decisión en lugar de usar una llamada a la RESTful API). A diferencia del patrón de implementación centralizada, el patrón distribuido requiere un esfuerzo mucho más sólido de implementación, mantenimiento y actualización, ya que está presente en múltiples áreas de la aplicación. Una ventaja del patrón de despliegue distribuido es la capacidad de mantener el aislamiento de los datos de los inquilinos, especialmente en el caso de las aplicaciones que utilizan un modelo SaaS aislado. Los datos específicos del inquilino se pueden aislar en las implementaciones de OPA específicas para ese inquilino, ya que la OPA en un modelo distribuido se implementa junto con el inquilino. Además, un patrón de despliegue distribuido tiene una latencia mucho menor que un patrón de despliegue centralizado, ya que cada decisión de autorización se puede tomar de forma local.

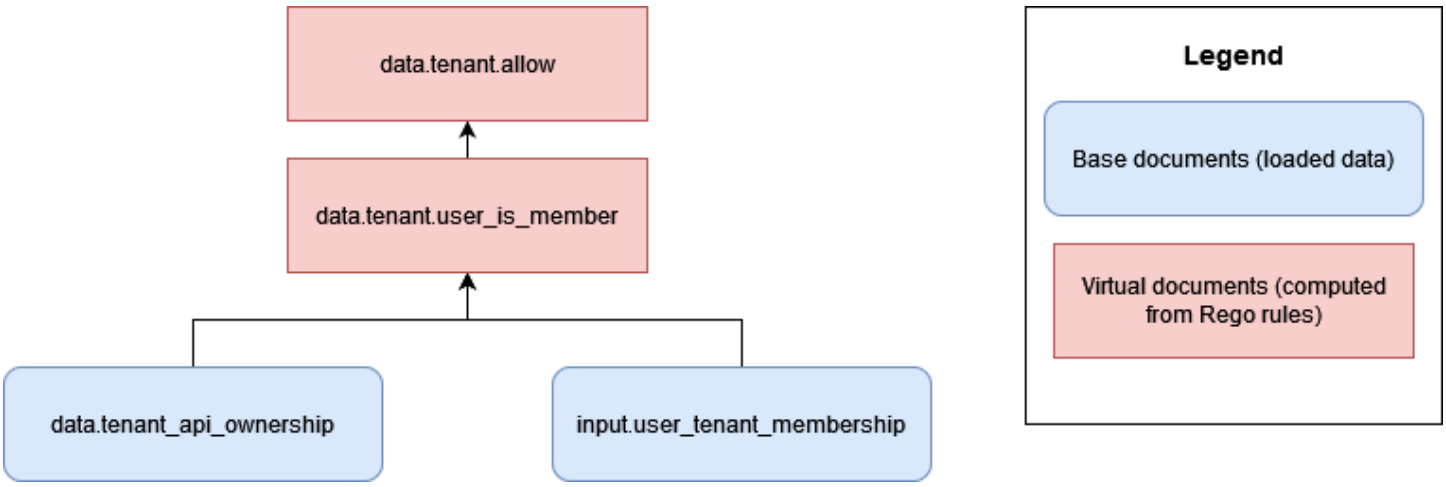
Cuando elija un patrón de despliegue de OPA en su aplicación multiusuario, asegúrese de evaluar los criterios más importantes para su aplicación. Si su aplicación multiusuario es sensible a la latencia, un patrón de implementación distribuido ofrece un mejor rendimiento a costa de una implementación y un mantenimiento más complejos. Si bien es posible gestionar parte de esta complejidad mediante DevOps la automatización, sigue requiriendo un esfuerzo adicional en comparación con un patrón de despliegue centralizado.

Si su aplicación multiusuario usa un modelo SaaS aislado, puede usar un patrón de implementación de OPA distribuido para imitar el enfoque aislado de los datos de los inquilinos. Esto se debe a que cuando OPA se ejecuta junto con cada servicio de aplicaciones específico del inquilino, puede personalizar cada implementación de OPA para que solo contenga los datos asociados a ese inquilino. No es posible aislar los datos de la OPA en un patrón de implementación de la OPA centralizado. Si utiliza un patrón de implementación centralizado o un patrón distribuido junto con un modelo SaaS agrupado, el aislamiento de los datos de los inquilinos debe mantenerse en el modelo de documentos de la OPA.

Aislamiento de inquilinos con el modelo de documentos OPA

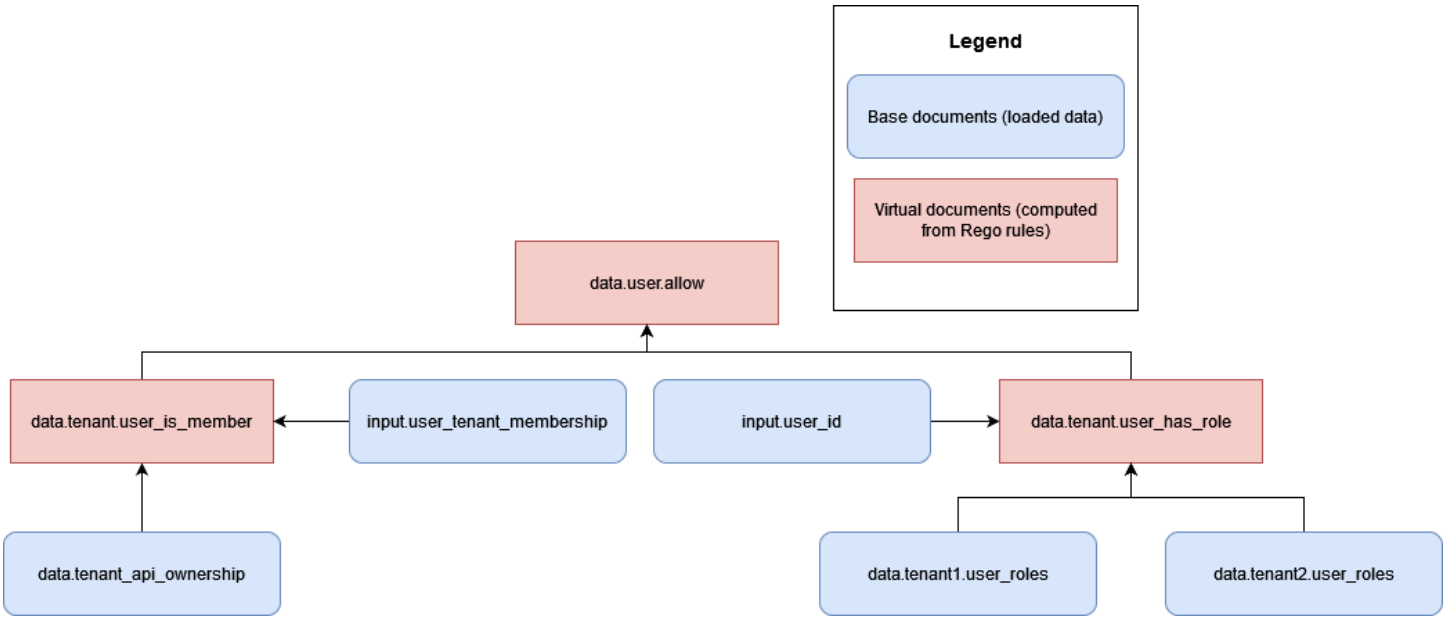
La OPA utiliza los documentos para tomar decisiones. Estos documentos pueden contener datos específicos de los inquilinos, por lo que debe considerar cómo mantener el aislamiento de los datos de los inquilinos. Los documentos de la OPA consisten en documentos base y documentos virtuales. Los documentos base contienen datos del mundo exterior. Esto incluye los datos proporcionados directamente a la OPA, los datos sobre la solicitud de la OPA y los datos que podrían transferirse a la OPA como entrada. Los documentos virtuales se calculan por política e incluyen las políticas y normas de la OPA. Para obtener más información, consulte la documentación de la OPA.

Para diseñar un modelo de documento en la OPA para una solicitud con varios inquilinos, primero debe considerar qué tipo de documentos base necesitará para tomar una decisión en la OPA. Si estos documentos básicos contienen datos específicos de los inquilinos, debe tomar medidas para garantizar que estos datos no se expongan accidentalmente al acceso de varios inquilinos. Afortunadamente, en muchos casos, no se requieren datos específicos del inquilino para tomar una decisión en la OPA. El siguiente ejemplo muestra un modelo de documento de OPA hipotético que permite el acceso a una API en función del arrendatario propietario de la API y de si el usuario es miembro del arrendatario, como se indica en el documento de entrada.



En este enfoque, la OPA no tiene acceso a ningún dato específico del inquilino, excepto a la información sobre qué inquilinos poseen una API. En este caso, no es motivo de preocupación que la OPA facilite el acceso entre inquilinos, ya que la única información que la OPA utiliza para tomar una decisión de acceso es la asociación del usuario con un inquilino y la asociación del inquilino con él. APIs Podría aplicar este tipo de modelo de documentos de OPA a un modelo de SaaS aislado, ya que cada inquilino tendría la propiedad de recursos independientes.

Sin embargo, en muchos enfoques de autorización del RBAC, existe la posibilidad de que la información quede expuesta entre distintos arrendatarios. En el siguiente ejemplo, un modelo hipotético de documento de la OPA permite el acceso a una API en función de si el usuario es miembro de un arrendatario y de si el usuario tiene la función correcta para acceder a la API.



Este modelo presenta un riesgo de acceso entre inquilinos, ya que los roles y permisos de varios inquilinos entran en `data.tenant1.user_roles` juego y ahora la OPA `data.tenant2.user_roles` debe ponerlos a disposición de la OPA para tomar decisiones de autorización. Para mantener el aislamiento de los inquilinos y la privacidad de la asignación de funciones, estos datos no deben residir en la OPA. Los datos del RBAC deben residir en una fuente de datos externa, como una base de datos. Además, la OPA no debe usarse para asignar funciones predefinidas a permisos específicos, ya que esto dificulta que los inquilinos definan sus propias funciones y permisos. También hace que la lógica de autorización sea rígida y necesite una actualización constante. Para obtener orientación sobre cómo incorporar de forma segura los datos del RBAC en el proceso de toma de decisiones de la OPA, consulte la sección Recomendaciones para el aislamiento de los inquilinos y la privacidad de los datos, más adelante en esta guía.

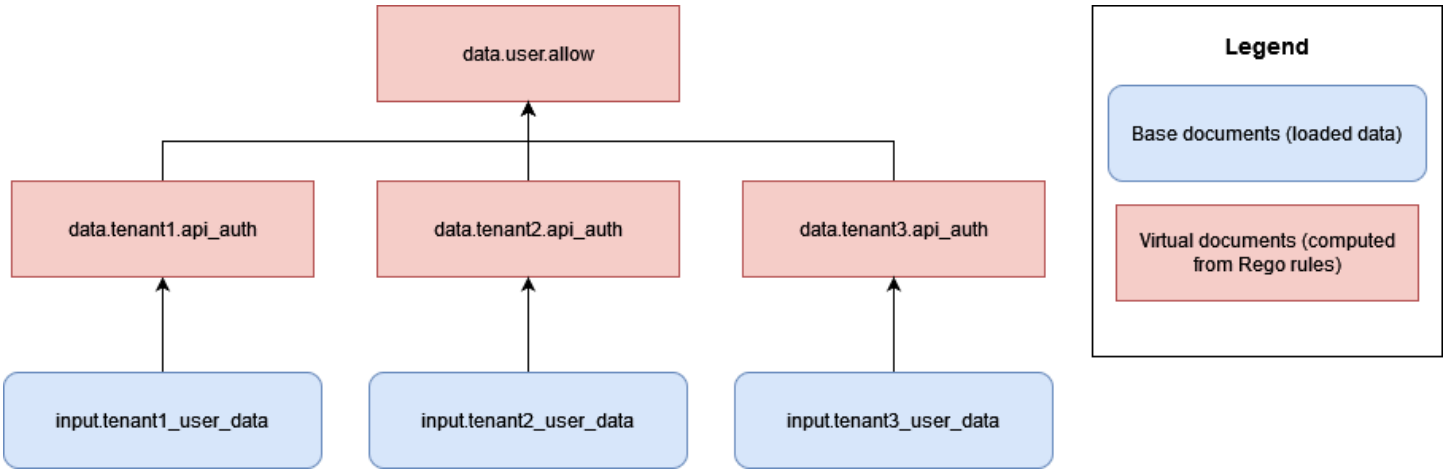
Puede mantener fácilmente el aislamiento de los inquilinos en la OPA al no almacenar ningún dato específico de los inquilinos como un documento base asíncrono. Un documento base asíncrono son datos que se almacenan en la memoria y se pueden actualizar periódicamente en la OPA. Otros documentos base, como la entrada de la OPA, se transmiten de forma sincrónica y solo están disponibles en el momento de tomar una decisión. Por ejemplo, proporcionar datos específicos del inquilino como parte de la entrada de la OPA a una consulta no constituiría una violación del aislamiento del inquilino, ya que esos datos solo están disponibles de forma sincrónica durante el proceso de toma de una decisión.

Incorporación de inquilinos

La estructura de los documentos de la OPA debe permitir la incorporación sencilla de los inquilinos sin introducir requisitos engorrosos. Puede organizar los documentos virtuales en la jerarquía del modelo de documentos de la OPA con paquetes, y estos paquetes pueden contener muchas reglas. Cuando planifique un modelo de documento de la OPA para una aplicación con varios usuarios, determine primero qué datos son necesarios para que la OPA tome una decisión. Puede proporcionar datos como entrada, precargarlos en la OPA o proporcionarlos desde fuentes de datos externas en el momento de tomar la decisión o de forma periódica. Para obtener más información sobre el uso de datos externos con la OPA, consulte la sección Recuperación de datos externos para un PDP en la OPA, más adelante en esta guía.

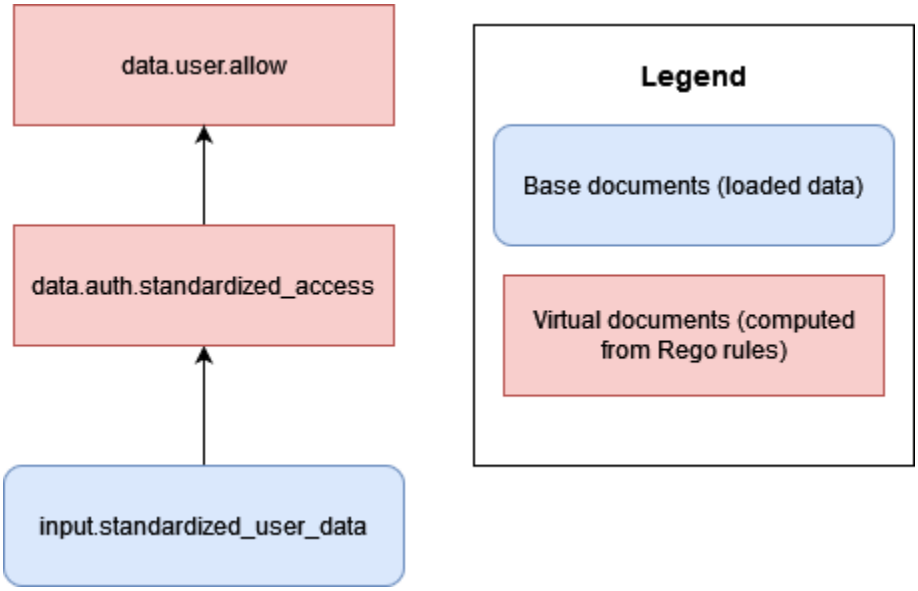
Después de determinar los datos necesarios para tomar una decisión en la OPA, considere cómo implementar las reglas de la OPA organizadas en paquetes para tomar decisiones con esos datos. Por ejemplo, en un modelo de SaaS aislado en el que cada inquilino pueda tener requisitos únicos

sobre la forma en que se toman las decisiones de autorización, podría implementar paquetes de reglas de OPA específicos para cada inquilino.



La desventaja de este enfoque es que debe agregar un nuevo conjunto de reglas de OPA, específicas para cada inquilino, para cada inquilino que agregue a su aplicación de SaaS. Esto es engorroso y difícil de escalar, pero puede ser inevitable según los requisitos de sus inquilinos.

Como alternativa, en un modelo SaaS agrupado, si todos los inquilinos toman decisiones de autorización en función de las mismas reglas y utilizan la misma estructura de datos, podría utilizar paquetes de OPA estándar que tengan reglas de aplicación general para facilitar la incorporación de los inquilinos y el escalado de la implementación de la OPA.



Siempre que sea posible, le recomendamos que utilice reglas y paquetes generalizados de la OPA (o documentos virtuales) para tomar decisiones basadas en los datos estandarizados proporcionados por cada inquilino. Este enfoque hace que la OPA sea fácilmente escalable, ya que solo se cambian

los datos proporcionados a la OPA para cada inquilino, no la forma en que la OPA toma sus decisiones a través de sus reglas. Solo es necesario introducir un rules-per-tenant modelo cuando los inquilinos individuales requieren decisiones únicas o tienen que proporcionar a OPA datos diferentes a los de otros inquilinos.

DevOps, supervisión, registro y recuperación de datos para un PDP

En este paradigma de autorización propuesto, las políticas están centralizadas en el servicio de autorización. Esta centralización es deliberada porque uno de los objetivos de los modelos de diseño que se analizan en esta guía es lograr la disociación de las políticas, es decir, eliminar la lógica de autorización de otros componentes de la aplicación. Tanto Amazon Verified Permissions como Open Policy Agent (OPA) proporcionan mecanismos para actualizar las políticas cuando es necesario realizar cambios en la lógica de autorización.

En el caso de los permisos verificados, el AWS SDK ofrece mecanismos para actualizar las políticas mediante programación (consulte la Guía de referencia de la API de permisos verificados de Amazon). Con el SDK, puede impulsar nuevas políticas a pedido. Además, dado que Verified Permissions es un servicio gestionado, no es necesario administrar, configurar ni mantener los planos de control o los agentes para realizar las actualizaciones. Sin embargo, le recomendamos que utilice una canalización de integración e implementación continuas (CI/CD) para administrar la implementación de los almacenes de políticas de permisos verificados y las actualizaciones de políticas mediante el AWS SDK.

Los permisos verificados proporcionan un fácil acceso a las funciones de observabilidad. Se puede configurar para registrar todos los intentos de acceso a los flujos de entrega de Amazon AWS CloudTrail, a los grupos de CloudWatch registros de Amazon Simple Storage Service (Amazon S3) o a las transmisiones de entrega de Amazon Data Firehose para permitir una respuesta rápida a los incidentes de seguridad y las solicitudes de auditoría. Además, puede supervisar el estado del servicio de permisos verificados a través del. Panel de AWS Health Dado que Verified Permissions es un servicio gestionado, su estado se mantiene mediante otros servicios gestionados AWS, y usted puede configurar las funciones de observabilidad mediante otros servicios AWS gestionados.

En el caso de OPA, REST APIs ofrece formas de actualizar las políticas mediante programación. Puede configurarlos APIs para que extraigan nuevas versiones de los paquetes de políticas de ubicaciones establecidas o para que publiquen políticas a pedido. Además, OPA ofrece un servicio de detección básico en el que los nuevos agentes se pueden configurar de forma dinámica y administrar de forma centralizada mediante un plano de control que distribuye los paquetes de detección. (El operador de la OPA debe configurar y configurar el plano de control de la OPA). Le recomendamos que cree un CI/CD proceso sólido para el control de versiones, la verificación y la

actualización de las políticas, ya sea que el motor de políticas sea Verified Permissions, OPA u otra solución.

En el caso de la OPA, el plano de control también ofrece opciones de supervisión y auditoría. Puede exportar los registros que contienen las decisiones de autorización de la OPA a servidores HTTP remotos para la agregación de registros. Estos registros de decisiones tienen un valor incalculable para fines de auditoría.

Si está pensando en adoptar un modelo de autorización en el que las decisiones de control de acceso estén desvinculadas de su aplicación, asegúrese de que su servicio de autorización cuente con capacidades eficaces de supervisión, registro y CI/CD administración para incorporar políticas nuevas PDPs o actualizarlas.

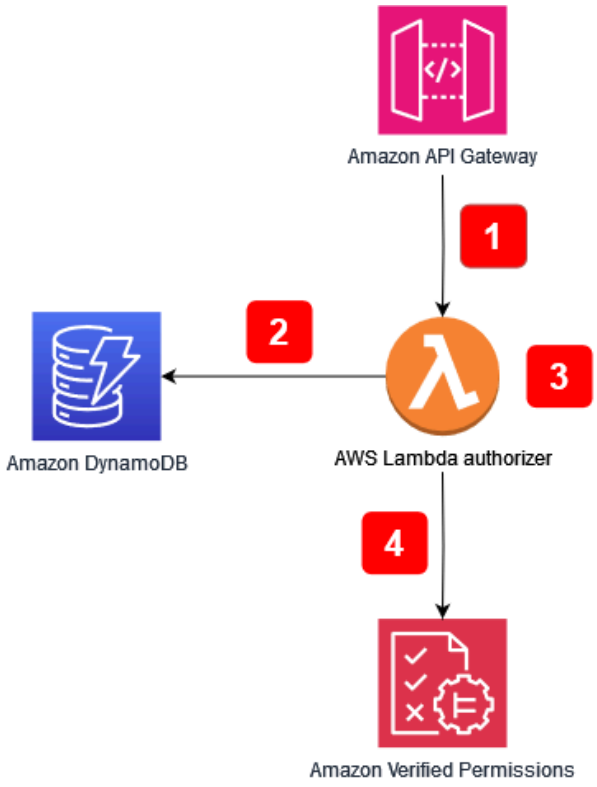
Temas

Recuperación de datos externos para un PDP en Amazon Verified Permissions

Amazon Verified Permissions no admite la recuperación de datos externos para un PDP, pero puede almacenar los datos proporcionados por el usuario como parte de su esquema. Al igual que en la OPA, si todos los datos para una decisión de autorización se pueden proporcionar como parte de una solicitud de autorización o como parte de un token web JSON (JWT) que se transfiere como parte de la solicitud, no se requiere ninguna configuración adicional. Sin embargo, puedes proporcionar datos adicionales de fuentes externas a Verified Permissions mediante la solicitud de autorización como parte del servicio de autorización de una aplicación que se llama Verified Permissions. Por ejemplo, el servicio de autorización de una aplicación puede consultar datos a una fuente externa, como DynamoDB o Amazon RDS, y estos servicios pueden incluir los datos proporcionados externamente como parte de una solicitud de autorización.

En el siguiente diagrama se muestra un ejemplo de cómo se pueden recuperar datos adicionales e incorporarlos a una solicitud de autorización de permisos verificados. Puede que sea necesario utilizar este método para recuperar datos, como las asignaciones de funciones de RBAC, para recuperar atributos adicionales que sean relevantes para los recursos o los principales, o en los

casos en que los datos residan en diferentes partes de una aplicación y no se puedan proporcionar a través de un token de proveedor de identidad (IdP).



Flujo de aplicaciones:

1. La aplicación recibe una llamada de API a Amazon API Gateway y la reenvía al AWS Lambda autorizador.
2. El autorizador de Lambda llama a Amazon DynamoDB para recuperar datos adicionales sobre la persona principal que realizó la solicitud.
3. El autorizador Lambda incorpora los datos adicionales en la solicitud de autorización que se realizó a Verified Permissions.
4. El autorizador de Lambda realiza una solicitud de autorización a Verified Permissions y recibe una decisión de autorización.

El diagrama incluye una función de Amazon API Gateway denominada autorizador Lambda. Si bien es posible que esta función no esté disponible para las APIs que ofrecen otros servicios o aplicaciones, puede replicar el modelo general de utilizar un autorizador para obtener datos adicionales e incorporarlos a una solicitud de autorización de permisos verificados en una multitud de casos de uso.

Recuperación de datos externos para un PDP en OPA

En el caso de la OPA, si todos los datos necesarios para una decisión de autorización se pueden proporcionar como entrada o como parte de un token web JSON (JWT) que se pasa como componente de la consulta, no se requiere ninguna configuración adicional. (Es relativamente sencillo pasar datos de contexto JWTs de SaaS a OPA como parte de la entrada de consultas). OPA puede aceptar entradas JSON arbitrarias en lo que se denomina el enfoque de entrada por sobrecarga. Si un PDP requiere más datos de los que se pueden incluir como entrada o como JWT, la OPA ofrece varias opciones para recuperar estos datos. Estas incluyen la agrupación, el envío de datos (replicación) y la recuperación dinámica de datos.

Empaquetado de OPA

La función de agrupación OPA admite el siguiente proceso de recuperación de datos externos:

1. El punto de aplicación de políticas (PEP) solicita una decisión de autorización.
2. La OPA descarga nuevos paquetes de políticas, incluidos los datos externos.
3. El servicio de agrupamiento replica los datos de las fuentes de datos.

Al utilizar la función de agrupación, OPA descarga periódicamente los paquetes de políticas y datos de un servicio de paquetes centralizado. (La OPA no proporciona la implementación y configuración de un servicio combinado). Todas las políticas y los datos externos que se extraen del servicio combinado se almacenan en la memoria. Esta opción no funcionará si el tamaño de los datos externos es demasiado grande para almacenarlos en la memoria o si los datos cambian con demasiada frecuencia.

Para obtener más información sobre la función de agrupamiento, consulte la documentación de la OPA.

Replicación de OPA (envío de datos)

El enfoque de replicación de la OPA admite el siguiente proceso de recuperación de datos externos:

1. El PEP solicita una decisión de autorización.
2. El replicador de datos envía los datos a la OPA.
3. El replicador de datos replica los datos de las fuentes de datos.

En esta alternativa al enfoque de agrupamiento, los datos se envían a la OPA, en lugar de recogerlos periódicamente. (La OPA no proporciona la implementación y configuración de un replicador). El enfoque push tiene las mismas limitaciones de tamaño de datos que el enfoque de agrupamiento, ya que la OPA almacena todos los datos en la memoria. La principal ventaja de la opción push es que puede actualizar los datos en OPA con deltas en lugar de reemplazar todos los datos externos cada vez. Esto hace que la opción push sea más adecuada para conjuntos de datos que cambian con frecuencia.

Para obtener más información sobre la opción de replicación, consulte la documentación de la OPA.

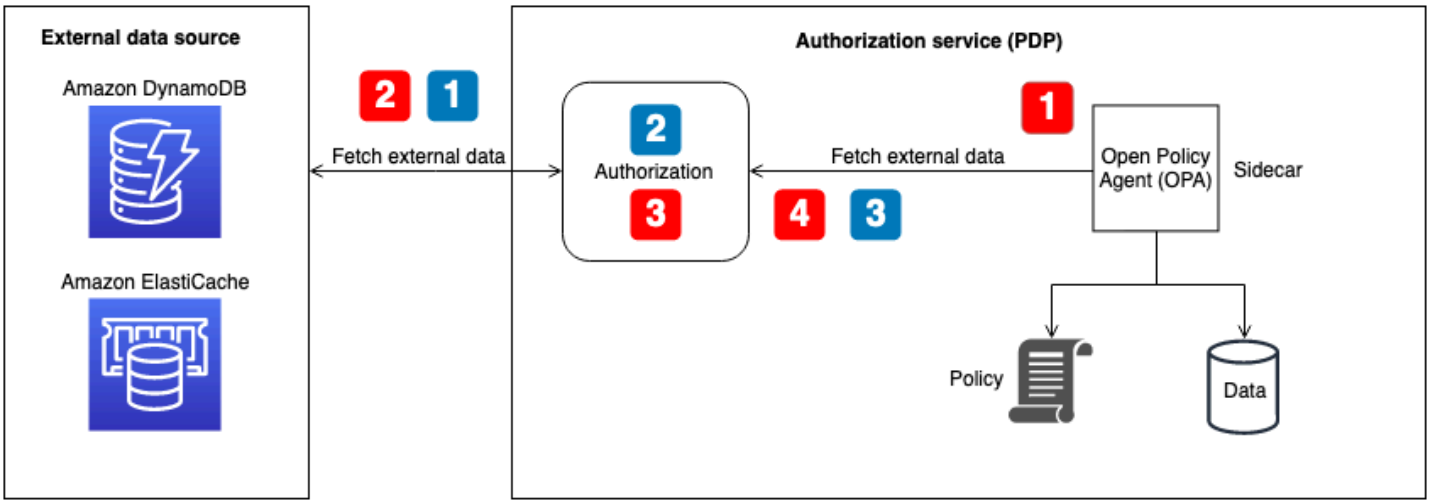
Recuperación dinámica de datos de OPA

Si los datos externos que se van a recuperar son demasiado grandes para guardarlos en caché en la memoria de la OPA, los datos se pueden extraer dinámicamente de una fuente externa durante la evaluación de una decisión de autorización. Cuando se utiliza este enfoque, los datos están siempre actualizados. Este enfoque tiene dos inconvenientes: la latencia de la red y la accesibilidad. Actualmente, OPA solo puede recuperar datos en tiempo de ejecución mediante una solicitud HTTP. Si las llamadas que van a una fuente de datos externa no pueden devolver datos como una respuesta HTTP, requieren una API personalizada o algún otro mecanismo para proporcionar estos datos a OPA. Dado que OPA solo puede recuperar datos a través de solicitudes HTTP y la velocidad de recuperación de los datos es fundamental, le recomendamos que utilice Amazon DynamoDB para almacenar datos externos siempre que sea posible. Servicio de AWS

Para obtener más información sobre el enfoque de atracción, consulte la documentación de la OPA.

Uso de un servicio de autorización para la implementación con OPA

Cuando busque datos externos mediante agrupamiento, replicación o un enfoque de extracción dinámica, le recomendamos que el servicio de autorización facilite esta interacción. Esto se debe a que el servicio de autorización puede recuperar datos externos y transformarlos en JSON para que OPA tome decisiones de autorización. El siguiente diagrama muestra cómo puede funcionar un servicio de autorización con estos tres enfoques de recuperación de datos externos.



Recuperación de datos externos para el flujo de OPA: recuperación dinámica o agrupada de datos en el momento de la decisión (ilustrada con rótulos numerados en rojo en el diagrama):

1. La OPA llama al punto final de la API local para el servicio de autorización, que se configura como un punto final de paquete o el punto final para la recuperación dinámica de datos durante las decisiones de autorización.
2. El servicio de autorización consulta o llama a la fuente de datos externa para recuperar datos externos. (En el caso de un punto final de paquete, estos datos también deben contener las políticas y normas de la OPA. Las actualizaciones de paquetes reemplazan todo lo que hay en la caché de la OPA (tanto los datos como las políticas).
3. El servicio de autorización realiza cualquier transformación necesaria en los datos devueltos para convertirlos en la entrada JSON esperada.
4. Los datos se devuelven a OPA. Se almacenan en caché en la memoria para la configuración del paquete y se utilizan inmediatamente para tomar decisiones de autorización dinámicas.

Recuperación de datos externos para el flujo de OPA: replicador (ilustrado con rótulos numerados en azul en el diagrama):

1. El replicador (parte del servicio de autorización) llama a la fuente de datos externa y recupera todos los datos que se van a actualizar en la OPA. Esto puede incluir políticas, reglas y datos externos. Esta llamada puede tener una cadencia determinada o puede producirse en respuesta a actualizaciones de datos en la fuente externa.
2. El servicio de autorización realiza cualquier transformación necesaria en los datos devueltos para convertirlos en la entrada JSON esperada.

3. El servicio de autorización llama a OPA y almacena en caché los datos en la memoria. El servicio de autorización puede actualizar datos, políticas y reglas de forma selectiva.

Recomendaciones para el aislamiento de los inquilinos y la privacidad de los datos

La sección anterior proporcionó varios enfoques para usar datos externos con los permisos verificados de OPA y Amazon para ayudar a tomar decisiones de autorización. Siempre que sea posible, le recomendamos que utilice el enfoque de entrada por sobrecarga para pasar los datos de contexto de SaaS a la OPA para tomar decisiones de autorización en lugar de almacenar los datos en la memoria de la OPA. Este caso de uso no se aplica a AWS Cloud Map, ya que no admite el almacenamiento de datos externos en el servicio.

En los modelos híbridos de control de acceso basado en roles (RBAC) o RBAC y control de acceso basado en atributos (ABAC), los datos proporcionados únicamente por una solicitud o consulta de autorización pueden ser insuficientes, ya que es necesario hacer referencia a los roles y permisos para tomar decisiones de autorización. Para mantener el aislamiento de los inquilinos y la privacidad de la asignación de funciones, estos datos no deben residir en la OPA. Los datos del RBAC deben residir en una fuente de datos externa, como una base de datos, o deben transmitirse como parte de las reclamaciones en un JWT desde un IdP. En los permisos verificados, los datos del RBAC se pueden mantener como parte de las políticas y el esquema del modelo de almacén de políticas por arrendatario, ya que cada arrendatario tiene su propio almacén de políticas separado de forma lógica. Sin embargo, en el modelo de almacén de políticas compartido con varios inquilinos, los datos de asignación de funciones no deben residir en los permisos verificados para mantener el aislamiento de los inquilinos.

Además, la OPA y los permisos verificados no deben usarse para asignar funciones predefinidas a permisos específicos, ya que esto dificulta que los inquilinos definan sus propias funciones y permisos. También hace que la lógica de autorización sea rígida y necesite una actualización constante. La excepción a esta directriz es el modelo de almacén de políticas por inquilino de Verified Permissions, ya que este modelo permite que cada arrendatario tenga sus propias políticas que se pueden evaluar de forma independiente por arrendatario.

Amazon Verified Permissions

El único lugar donde los permisos verificados pueden almacenar datos RBAC potencialmente privados es en el esquema. Esto es aceptable en el modelo de almacén de políticas por arrendatario,

ya que cada arrendatario tiene su propio almacén de políticas separado de forma lógica. Sin embargo, podría comprometer el aislamiento de los inquilinos en el modelo de almacén de políticas compartido con varios inquilinos. En los casos en que estos datos sean necesarios para tomar una decisión de autorización, deben recuperarse de una fuente externa, como DynamoDB o Amazon RDS, e incorporarse a la solicitud de autorización de permisos verificados.

OPA

Los enfoques seguros de la OPA para mantener la privacidad y el aislamiento de los datos del RBAC entre los usuarios incluyen el uso de la recuperación o replicación dinámica de datos para obtener datos externos. Esto se debe a que puede usar el servicio de autorización ilustrado en el diagrama anterior para proporcionar solo datos externos específicos del inquilino o del usuario para tomar una decisión de autorización. Por ejemplo, puede usar un replicador para proporcionar datos RBAC o una matriz de permisos a la caché de la OPA cuando un usuario inicia sesión y hacer que se haga referencia a los datos en función del usuario proporcionado en los datos de entrada. Puede utilizar un enfoque similar con datos extraídos de forma dinámica para recuperar solo los datos relevantes para tomar decisiones de autorización. Además, en el enfoque de recuperación dinámica de datos, estos datos no tienen que estar en caché en OPA. El enfoque de agrupamiento no es tan eficaz como el enfoque de recuperación dinámica para mantener el aislamiento de los inquilinos, ya que actualiza todo lo que hay en la memoria caché de la OPA y no puede procesar actualizaciones precisas. El modelo de agrupamiento sigue siendo un buen enfoque para actualizar las políticas de la OPA y los datos no relacionados con el RBAC.

Prácticas recomendadas

En esta sección se enumeran algunas de las conclusiones de alto nivel de esta guía. Para obtener información detallada sobre cada punto, siga los enlaces a las secciones correspondientes.

Seleccione un modelo de control de acceso que funcione para su aplicación

En esta guía se analizan varios modelos de control de acceso. En función de la aplicación y de los requisitos empresariales, debe seleccionar el modelo que mejor se adapte a sus necesidades. Considere cómo puede utilizar estos modelos para satisfacer sus necesidades de control de acceso y cómo podrían evolucionar estas necesidades, lo que requeriría cambios en el enfoque que haya seleccionado.

Implemente un PDP

El punto de decisión política (PDP) se puede caracterizar como un motor de políticas o reglas. Este componente es responsable de aplicar las políticas o reglas y de decidir si se permite un acceso en particular. Un PDP permite transferir la lógica de autorización del código de la aplicación a un sistema independiente. Esto puede simplificar el código de la aplicación. También proporciona una interfaz easy-to-use ideal para tomar decisiones de autorización para microservicios APIs, capas de backend for Frontend (BFF) o cualquier otro componente de la aplicación. Se puede usar un PDP para hacer cumplir los requisitos de arrendamiento de manera uniforme en una aplicación.

PEPs Impleméntelo para cada API de su aplicación

La implementación de un punto de aplicación de políticas (PEP) requiere determinar dónde debe aplicarse el control de acceso en una aplicación. Como primer paso, localice los puntos de la aplicación en los que pueda PEPs incorporarla. Tenga en cuenta este principio a la hora de decidir dónde añadir PEPs:

Si una aplicación expone una API, debe haber autorización y control de acceso en esa API.

Considere la posibilidad de utilizar Amazon Verified Permissions (OPA) como motor de políticas para su PDP

Los permisos verificados de Amazon tienen ventajas sobre los motores de políticas personalizados. Se trata de un servicio de autorización y administración de permisos escalable y detallado para las aplicaciones que cree. Es compatible con la redacción de políticas en el lenguaje declarativo de código abierto de alto nivel Cedar. Como resultado, implementar un motor de políticas mediante permisos verificados requiere menos esfuerzo de desarrollo que implementar su propia solución. Además, Verified Permissions está totalmente gestionado, por lo que no es necesario gestionar la infraestructura subyacente.

El Open Policy Agent (OPA) presenta ventajas en comparación con los motores de políticas personalizados. La OPA y su evaluación de políticas con Rego proporcionan un motor de políticas flexible y prediseñado que permite redactar políticas en un lenguaje declarativo de alto nivel. Esto hace que el nivel de esfuerzo necesario para implementar un motor de políticas sea significativamente menor que para crear su propia solución. Además, la OPA se está convirtiendo rápidamente en un estándar de autorización bien respaldado.

Implemente un plano de control para la OPA para DevOps la supervisión y el registro

Como la OPA no proporciona un medio para actualizar y realizar un seguimiento de los cambios en la lógica de autorización mediante el control de código fuente, le recomendamos que implemente un plano de control para realizar estas funciones. Esto permitirá que las actualizaciones se distribuyan más fácilmente a los agentes de la OPA, especialmente si la OPA opera en un sistema distribuido, lo que reducirá la carga administrativa que implica el uso de la OPA. Además, se puede utilizar un plano de control para recopilar registros para su agregación y supervisar el estado de los agentes de la OPA.

Configure las funciones de registro y observabilidad en Verified Permissions

Los permisos verificados proporcionan un fácil acceso a las funciones de observabilidad. Puede configurar el servicio para que registre todos los intentos de acceso a los flujos de entrega de Amazon Data Firehose, grupos de CloudWatch registros de Amazon, buckets S3 o Amazon Data

Firehose para poder responder rápidamente a los incidentes de seguridad y a las solicitudes de auditoría. AWS CloudTrail Además, puede supervisar el estado del servicio a través del. Panel de AWS Health Dado que Verified Permissions es un servicio gestionado, su estado se mantiene mediante otros servicios gestionados AWS, y usted puede configurar sus funciones de observación mediante el uso de otros servicios AWS gestionados.

Utilice una CI/CD canalización para aprovisionar y actualizar los almacenes de políticas y las políticas en Verified Permissions

Los permisos verificados son un servicio gestionado, por lo que no es necesario gestionar, configurar ni mantener los planos de control o los agentes para realizar las actualizaciones. Sin embargo, le recomendamos que utilice una canalización de integración e implementación continuas (CI/CD) para administrar la implementación de los almacenes de políticas de permisos verificados y las actualizaciones de políticas mediante el AWS SDK. Este esfuerzo puede eliminar el esfuerzo manual y reducir la probabilidad de que se cometan errores por parte del operador al realizar cambios en los recursos de permisos verificados.

Determine si se requieren datos externos para tomar decisiones de autorización y seleccione un modelo que se adapte a ellos

Si un PDP puede tomar decisiones de autorización basándose únicamente en los datos contenidos en un token web JSON (JWT), normalmente no es necesario importar datos externos para ayudar a tomar estas decisiones. Si utilizas permisos verificados u OPA como PDP, también puede aceptar datos adicionales que se envíen como parte de la solicitud, incluso si estos datos no están incluidos en un JWT. En el caso de los permisos verificados, puedes usar un parámetro de contexto para los datos adicionales. En el caso de la OPA, puede utilizar los datos JSON como entrada de sobrecarga. Si utilizas un JWT, los métodos de entrada contextuales o de sobrecarga suelen ser mucho más fáciles que mantener los datos externos en otra fuente. Si se requieren datos externos más complejos para tomar decisiones de autorización, OPA ofrece varios modelos para recuperar datos externos, y Verified Permissions puede complementar los datos de sus solicitudes de autorización haciendo referencia a fuentes externas con un servicio de autorización.

Preguntas frecuentes

En esta sección se proporcionan respuestas a las preguntas más frecuentes sobre la implementación del control de acceso y la autorización de las API en aplicaciones SaaS multiusuario.

P: ¿Cuál es la diferencia entre autorización y autenticación?

R. La autenticación es el proceso de verificar quién es un usuario. La autorización otorga permisos a los usuarios para acceder a un recurso específico.

P: ¿Cuál es la diferencia entre la autorización y el aislamiento de inquilinos en una aplicación SaaS?

R. El aislamiento de inquilinos se refiere a los mecanismos explícitos que se utilizan en un sistema SaaS para garantizar que los recursos de cada inquilino, incluso cuando operan en una infraestructura compartida, estén aislados. La autorización de varios inquilinos se refiere a la autorización de acciones entrantes y a impedir que se implementen en el arrendatario equivocado. Un usuario hipotético podría estar autenticado y autorizado, pero podría seguir accediendo a los recursos de otro inquilino. Nada de lo relacionado con la autenticación y la autorización bloquea necesariamente este acceso, pero es necesario aislar al inquilino para lograr este objetivo. Para obtener más información sobre estos dos conceptos, consulte el debate sobre el aislamiento de inquilinos en el documento técnico Fundamentos de la arquitectura de AWS SaaS.

P: ¿Por qué debo considerar el aislamiento de inquilinos para mi solicitud de SaaS?

R. Las aplicaciones SaaS tienen varios inquilinos. Un inquilino puede ser una organización cliente o cualquier entidad externa que utilice esa aplicación SaaS. Según el diseño de la aplicación, esto significa que los inquilinos pueden estar accediendo a recursos compartidos APIs, a bases de datos u otros recursos. Es importante mantener el aislamiento de los inquilinos (es decir, estructuras que controlen estrictamente el acceso a los recursos y bloqueen cualquier intento de acceder a los recursos de otro inquilino) para evitar que los usuarios de un inquilino accedan a la información privada de otro inquilino. Las aplicaciones SaaS suelen diseñarse para garantizar que el aislamiento de los inquilinos se mantenga en toda la aplicación y que los inquilinos solo puedan acceder a sus propios recursos.

P: ¿Por qué necesito un modelo de control de acceso?

R. Los modelos de control de acceso se utilizan para crear un método coherente para determinar cómo conceder el acceso a los recursos de una aplicación. Esto se puede hacer asignando

funciones a los usuarios que estén estrechamente alineadas con la lógica empresarial, o puede basarse en otros atributos contextuales, como la hora del día o si un usuario cumple una condición predefinida. Los modelos de control de acceso constituyen la lógica básica que utiliza la aplicación al tomar decisiones de autorización para determinar los permisos de los usuarios.

P: ¿Es necesario el control de acceso mediante API para mi aplicación?

R: Sí. APIs debe comprobar siempre que la persona que llama tiene el acceso adecuado. El control de acceso generalizado a través de la API también garantiza que el acceso solo se conceda en función de los inquilinos, de modo que se pueda mantener el aislamiento adecuado.

P: ¿Por qué se PDPs recomiendan los motores de políticas o la autorización?

R. Un punto de decisión de políticas (PDP) permite transferir la lógica de autorización del código de la aplicación a un sistema independiente. Esto puede simplificar el código de la aplicación. También proporciona una interfaz easy-to-use ideal para tomar decisiones de autorización para microservicios APIs, capas de backend for Frontend (BFF) o cualquier otro componente de la aplicación.

P: ¿Qué es un PEP?

R. Un punto de aplicación de políticas (PEP) es responsable de recibir las solicitudes de autorización que se envían al PDP para su evaluación. Un PEP puede estar en cualquier parte de una aplicación donde se deban proteger los datos y los recursos o donde se aplique la lógica de autorización. PEPs son relativamente simples en comparación con. PDPs Un PEP es responsable únicamente de solicitar y evaluar una decisión de autorización y no requiere que se le incorpore ninguna lógica de autorización.

P: ¿Cómo debo elegir entre los permisos verificados de Amazon y OPA?

R. Para elegir entre permisos verificados y Open Policy Agent (OPA), ten siempre en cuenta tu caso de uso y tus requisitos exclusivos. Verified Permissions ofrece una forma totalmente gestionada de definir permisos detallados, auditar los permisos en todas las aplicaciones y centralizar el sistema de administración de políticas para sus aplicaciones, a la vez que se cumplen los requisitos de latencia de las aplicaciones con un procesamiento de milisegundos. OPA es un motor de políticas de código abierto y de uso general que también puede ayudarlo a unificar las políticas en toda su gama de aplicaciones. Para ejecutar OPA, debe alojarla en su AWS entorno, normalmente con un contenedor o funciones. AWS Lambda

Verified Permissions usa el lenguaje de políticas de código abierto Cedar, mientras que OPA usa Rego. Por lo tanto, la familiaridad con uno de estos lenguajes podría llevarle a elegir esa solución.

Sin embargo, te recomendamos que leas información sobre ambos idiomas y, a continuación, retomes el problema que intentas resolver para encontrar la mejor solución para tu caso de uso.

P: ¿Existen alternativas de código abierto a los permisos verificados y a la OPA?

R. Hay algunos sistemas de código abierto que son similares a Verified Permissions y al Open Policy Agent (OPA), como el Common Expression Language (CEL). Esta guía se centra tanto en los permisos verificados, que es una administración de permisos escalable y un servicio de autorización detallado, como en la OPA, que está ampliamente adoptada, documentada y adaptable a muchos tipos diferentes de aplicaciones y requisitos de autorización.

P: ¿Necesito redactar un servicio de autorización para usar la OPA o puedo interactuar directamente con la OPA?

R: Puede interactuar con OPA directamente. En el contexto de esta guía, un servicio de autorización se refiere a un servicio que traduce las solicitudes de decisiones de autorización en consultas de la OPA y viceversa. Si su solicitud puede consultar y aceptar las respuestas de la OPA directamente, no es necesario introducir esta complejidad adicional.

P: ¿Cómo superviso el tiempo de actividad de mis agentes de la OPA y con fines de auditoría?

R: La OPA proporciona un registro y una supervisión básica del tiempo de actividad, aunque es probable que su configuración predeterminada no sea suficiente para las implementaciones empresariales. Para obtener más información, consulte la DevOps sección sobre supervisión y registro que aparece anteriormente en esta guía.

P: ¿Cómo puedo supervisar los permisos verificados para comprobar el tiempo de actividad y realizar auditorías?

R. Los permisos verificados son un servicio AWS gestionado y se puede supervisar su disponibilidad a través del Panel de AWS Health. Además, Verified Permissions permite iniciar AWS CloudTrail sesión en Amazon CloudWatch Logs, Amazon S3 y Amazon Data Firehose.

P: ¿Qué sistemas operativos y AWS servicios puedo usar para ejecutar OPA?

R. Puede ejecutar OPA en macOS, Windows y Linux. Los agentes de OPA se pueden configurar en agentes de Amazon Elastic Compute Cloud (Amazon EC2), así como en servicios de contenerización como Amazon Elastic Container Service (Amazon ECS) y Amazon Elastic Kubernetes Service (Amazon EKS).

P: ¿Qué sistemas operativos y AWS servicios puedo usar para ejecutar los permisos verificados?

R: Verified Permissions es un servicio AWS gestionado y está gestionado por AWS. No se necesita ninguna configuración, instalación o alojamiento adicionales para utilizar los permisos verificados, excepto la capacidad de realizar solicitudes de autorización al servicio.

P: ¿Puedo ejecutar OPA AWS Lambda?

R. Puede ejecutar OPA en Lambda como una biblioteca Go. Para obtener información sobre cómo hacerlo con un autorizador Lambda de API Gateway, consulte la entrada del AWS blog Creación de un autorizador Lambda personalizado mediante Open Policy Agent.

P: ¿Cómo debo decidir entre un enfoque de PDP distribuido o un PDP centralizado?

R: Esto depende de su aplicación. Lo más probable es que se determine en función de la diferencia de latencia entre un modelo PDP distribuido y uno centralizado. Le recomendamos que cree una prueba de concepto y pruebe el rendimiento de la aplicación para verificar la solución.

P: ¿Puedo usar OPA también para otros casos de uso APIs?

R: Sí. La documentación de la OPA proporciona ejemplos de Kubernetes, Envoy, Docker, Kafka, SSH y sudo, y Terraform. Además, OPA puede devolver respuestas JSON arbitrarias a las consultas mediante el uso de reglas parciales de Rego. Según la consulta, OPA se puede utilizar para responder a muchas preguntas con respuestas en formato JSON.

P: ¿Puedo usar también los permisos verificados para otros casos de uso APIs?

R: Sí. Los permisos verificados pueden proporcionar una `ALLOW` `DENY` respuesta a cualquier solicitud de autorización que reciba. Los permisos verificados pueden proporcionar respuestas de autorización para cualquier aplicación o servicio que requiera decisiones de autorización.

P: ¿Puedo crear políticas en Verified Permissions utilizando el lenguaje de políticas de IAM?

R: No. Debe utilizar el lenguaje de políticas de Cedar para crear políticas. Cedar se ha diseñado para facilitar la gestión de permisos de los recursos de las aplicaciones de los clientes, mientras que el lenguaje de políticas AWS Identity and Access Management (IAM) ha evolucionado para facilitar el control de acceso a AWS los recursos.

Pasos a seguir a continuación

La complejidad de la autorización y el control de acceso de las API de las aplicaciones SaaS multiusuario se puede superar mediante la adopción de un enfoque estandarizado e independiente del lenguaje para tomar decisiones de autorización. Estos enfoques incorporan puntos de decisión política (PDPs) y puntos de aplicación de políticas (PDPs) que imponen el acceso de manera flexible y generalizada. Se pueden incorporar varios enfoques del control de acceso, como el control de acceso basado en roles (RBAC), el control de acceso basado en atributos (ABAC) o una combinación de ambos, en una estrategia de control de acceso coherente. Al eliminar la lógica de autorización de una aplicación, se elimina la sobrecarga que supone incluir soluciones específicas en el código de la aplicación para abordar el control de acceso. La implementación y las recomendaciones analizadas en esta guía tienen como objetivo informar y estandarizar un enfoque destinado a la implementación de la autorización y el control de acceso para las API en aplicaciones SaaS multiusuario. Puede utilizar esta guía como primer paso para recopilar información y diseñar un sistema sólido de autorización y control de acceso para su aplicación. Próximos pasos:

- Revise sus necesidades de autorización y aislamiento de usuarios y seleccione un modelo de control de acceso para su aplicación.
- Cree una prueba de concepto para realizar pruebas utilizando Amazon Verified Permissions o Open Policy Agent (OPA), o bien diseñe su propio motor de políticas personalizado.
- Identifique APIs y localice en su aplicación los lugares donde PEPs debe implementarse.

Recursos

Referencias

- Documentación sobre permisos verificados de Amazon (AWS documentación)
- Cómo utilizar los permisos verificados de Amazon para la autorización (AWS entrada del blog)
- Implementación de un proveedor de políticas de autorización personalizado para aplicaciones principales de ASP.NET mediante permisos verificados de Amazon (entrada AWS del blog)
- Gestione funciones y derechos con PBAC mediante Amazon Verified Permissions AWS (entrada del blog)
- Control de acceso SaaS mediante permisos verificados de Amazon con un almacén de políticas por inquilino (AWS entrada del blog)
- La documentación oficial de la OPA
- Por qué las empresas deben adoptar el proyecto de la CNCF recién graduado: Open Policy Agent (artículo de Forbes de Janakiram MSV, 8 de febrero de 2021)
- Creación de un autorizador Lambda personalizado mediante Open Policy Agent (AWS entrada del blog)
- Haga realidad las políticas como código con el AWS Cloud Development Kit a través de Open Policy Agent (entrada AWS del blog)
- La gobernanza de la nube y el AWS cumplimiento de la política como código (AWS entrada del blog)
- Uso de Open Policy Agent en Amazon EKS (AWS entrada del blog)
- La conformidad como código para Amazon ECS mediante Open Policy Agent EventBridge, Amazon y AWS Lambda (entrada AWS del blog)
- Contramedidas basadas en políticas para Kubernetes: primera parte (entrada del blog)AWS
- Uso de autorizadores Lambda de API Gateway (documentación)AWS

Herramientas

- The Cedar Playground (para probar Cedar en un navegador)
- Repositorio Cedar Github
- Referencia del lenguaje Cedar

- The Rego Playground (para probar Rego en un navegador)
- Repositorio OPA GitHub

Socios

- Socios de Identity and Access Management
- Socios de seguridad de aplicaciones
- Socios de gobernanza de la nube
- Socios de seguridad y cumplimiento
- Socios de automatización y operaciones de seguridad
- Socios de ingeniería de seguridad

Historial de documentos

En la siguiente tabla, se describen cambios significativos de esta guía. Si quiere recibir notificaciones de futuras actualizaciones, puede suscribirse a las notificaciones RSS.

Cambio	Descripción	Fecha
Detalles y ejemplos añadidos para los permisos verificados de Amazon	Se agregaron discusion es detalladas, ejemplos y código para usar los permisos verificados de Amazon para implementar un PDP. Las nuevas secciones incluyen: • Implementación de un PDP mediante permisos verificad os de Amazon • Modelos de diseño para los permisos verificados de Amazon • Consideraciones de diseño para varios inquilinos de Amazon Verified Permissio ns • Recuperación de datos externos para un PDP en Amazon Verified Permissio ns	28 de mayo de 2024
Información aclarada	Se aclaró el PDP distribuido con un modelo PEPs de APIs diseño.	10 de enero de 2024
Se agregó información breve sobre el nuevo servicio AWS	Se agregó información sobre los permisos verificados de Amazon, que proporcionan las	22 de mayo de 2023

	mismas funciones y beneficios que OPA.	
—	Publicación inicial	17 de agosto de 2021

AWS Glosario de orientación prescriptiva

Los siguientes son términos de uso común en las estrategias, guías y patrones proporcionados por la Guía AWS prescriptiva. Para sugerir entradas, utilice el enlace Enviar comentarios al final del glosario.

Números

Las 7 R

Siete estrategias de migración comunes para trasladar aplicaciones a la nube. Estas estrategias se basan en las 5 R que Gartner identificó en 2011 y consisten en lo siguiente:

- Refactorizar/rediseñar: traslade una aplicación y modifique su arquitectura mediante el máximo aprovechamiento de las características nativas en la nube para mejorar la agilidad, el rendimiento y la escalabilidad. Por lo general, esto implica trasladar el sistema operativo y la base de datos. Ejemplo: Migrar la base de datos de Oracle en las instalaciones a Amazon Aurora PostgreSQL-Compatible Edition.
- Redefinir la plataforma (transportar y redefinir): traslade una aplicación a la nube e introduzca algún nivel de optimización para aprovechar las capacidades de la nube. Ejemplo: Migrar la base de datos Oracle en las instalaciones a Amazon Relational Database Service (Amazon RDS) para Oracle en la nube de Nube de AWS.
- Recomprar (readquirir): cambie a un producto diferente, lo cual se suele llevar a cabo al pasar de una licencia tradicional a un modelo SaaS. Ejemplo: Migrar el sistema de administración de las relaciones con los clientes (CRM) a Salesforce.com.
- Volver a alojar (migrar mediante lift-and-shift): traslade una aplicación a la nube sin realizar cambios para aprovechar las capacidades de la nube. Ejemplo: Migrar la base de datos de Oracle en las instalaciones a Oracle en una instancia de EC2 en la Nube de AWS.
- Reubicar: (migrar el hipervisor mediante lift and shift): traslade la infraestructura a la nube sin comprar equipo nuevo, reescribir aplicaciones o modificar las operaciones actuales. Los servidores se migran de una plataforma en las instalaciones a un servicio en la nube para la misma plataforma. Ejemplo: migrar una Microsoft Hyper-V aplicación a AWS.
- Retener (revisitar): conserve las aplicaciones en el entorno de origen. Estas pueden incluir las aplicaciones que requieren una refactorización importante, que desee posponer para más adelante, y las aplicaciones heredadas que desee retener, ya que no hay ninguna justificación empresarial para migrarlas.

- Retirar: retire o elimine las aplicaciones que ya no sean necesarias en un entorno de origen.

A

ABAC

Consulte control de acceso basado en atributos.

servicios abstractos

Consulte servicios administrados.

ACID

Consulte atomicidad, consistencia, aislamiento, durabilidad.

migración activa-activa

Método de migración de bases de datos en el que las bases de datos de origen y destino se mantienen sincronizadas (mediante una herramienta de replicación bidireccional o mediante operaciones de escritura doble) y ambas bases de datos gestionan las transacciones de las aplicaciones conectadas durante la migración. Este método permite la migración en lotes pequeños y controlados, en lugar de requerir una transición única. Es más flexible, pero requiere más trabajo que una migración activa-pasiva.

migración activa-pasiva

Método de migración de bases de datos en el que las bases de datos de origen y destino se mantienen sincronizadas, pero solo la de origen gestiona las transacciones de las aplicaciones conectadas, mientras los datos se replican en la de destino. La base de datos de destino no acepta ninguna transacción durante la migración.

función de agregación

Función SQL que actúa en un grupo de filas y calcula un único valor de devolución para el grupo. Entre los ejemplos de funciones de agregación se incluyen `SUM` y `MAX`.

IA

Consulte inteligencia artificial.

AIOps

Consulte operaciones de inteligencia artificial

anonimización

El proceso de eliminar permanentemente la información personal de un conjunto de datos. La anonimización puede ayudar a proteger la privacidad personal. Los datos anonimizados ya no se consideran datos personales.

antipatrones

Una solución que se utiliza con frecuencia para un problema recurrente en el que la solución es contraproducente, ineficaz o menos eficaz que una alternativa.

control de aplicaciones

Enfoque de seguridad que permite usar de manera exclusiva aplicaciones aprobadas para ayudar a proteger un sistema contra el malware.

cartera de aplicaciones

Recopilación de información detallada sobre cada aplicación que utiliza una organización, incluido el costo de creación y mantenimiento de la aplicación y su valor empresarial. Esta información es clave para el proceso de detección y análisis de la cartera y ayuda a identificar y priorizar las aplicaciones que se van a migrar, modernizar y optimizar.

inteligencia artificial (IA)

El campo de la informática que se dedica al uso de tecnologías informáticas para realizar funciones cognitivas que suelen estar asociadas a los seres humanos, como el aprendizaje, la resolución de problemas y el reconocimiento de patrones. Para más información, consulte ¿Qué es la inteligencia artificial?

operaciones de inteligencia artificial (AIOps)

El proceso de utilizar técnicas de machine learning para resolver problemas operativos, reducir los incidentes operativos y la intervención humana, y mejorar la calidad del servicio. Para obtener más información sobre cómo AIOps se utiliza en la estrategia de AWS migración, consulte la guía de integración de operaciones.

cifrado asimétrico

Algoritmo de cifrado que utiliza un par de claves, una clave pública para el cifrado y una clave privada para el descifrado. Puede compartir la clave pública porque no se utiliza para el descifrado, pero el acceso a la clave privada debe estar sumamente restringido.

atomicidad, consistencia, aislamiento, durabilidad (ACID)

Conjunto de propiedades de software que garantizan la validez de los datos y la fiabilidad operativa de una base de datos, incluso en caso de errores, cortes de energía u otros problemas.

control de acceso basado en atributos (ABAC)

La práctica de crear permisos detallados basados en los atributos del usuario, como el departamento, el puesto de trabajo y el nombre del equipo. Para obtener más información, consulte ABAC AWS en la documentación AWS Identity and Access Management (IAM).

origen de datos fidedigno

Ubicación en la que se almacena la versión principal de los datos, que se considera la fuente de información más fiable. Puede copiar los datos del origen de datos autorizado a otras ubicaciones con el fin de procesarlos o modificarlos, por ejemplo, anonimizarlos, redactarlos o seudonimizarlos.

Zona de disponibilidad

Una ubicación distinta dentro de una Región de AWS que está aislada de los fallos en otras zonas de disponibilidad y que proporciona una conectividad de red económica y de baja latencia a otras zonas de disponibilidad de la misma región.

AWS Marco de adopción de la nube (AWS CAF)

Un marco de directrices y mejores prácticas AWS para ayudar a las organizaciones a desarrollar un plan eficiente y eficaz para migrar con éxito a la nube. AWS CAF organiza la orientación en seis áreas de enfoque denominadas perspectivas: negocios, personas, gobierno, plataforma, seguridad y operaciones. Las perspectivas empresariales, humanas y de gobernanza se centran en las habilidades y los procesos empresariales; las perspectivas de plataforma, seguridad y operaciones se centran en las habilidades y los procesos técnicos. Por ejemplo, la perspectiva humana se dirige a las partes interesadas que se ocupan de los Recursos Humanos (RR. HH.), las funciones del personal y la administración de las personas. Desde esta perspectiva, AWS CAF proporciona orientación para el desarrollo, la formación y la comunicación de las personas a fin de preparar a la organización para una adopción exitosa de la nube. Para obtener más información, consulte la Página web de AWS CAF y el Documento técnico de AWS CAF.

AWS Marco de calificación de la carga de trabajo (AWS WQF)

Herramienta que evalúa las cargas de trabajo de migración de bases de datos, recomienda estrategias de migración y proporciona estimaciones de trabajo. AWS WQF se incluye con AWS

Schema Conversion Tool ().AWS SCT Analiza los esquemas de bases de datos y los objetos de código, el código de las aplicaciones, las dependencias y las características de rendimiento y proporciona informes de evaluación.

B

bot malicioso

Bot destinado a causar interrupciones o daños a personas u organizaciones.

BCP

Consulte planificación de la continuidad del negocio.

gráfico de comportamiento

Una vista unificada e interactiva del comportamiento de los recursos y de las interacciones a lo largo del tiempo. Puede utilizar un gráfico de comportamiento con Amazon Detective para examinar los intentos de inicio de sesión fallidos, las llamadas sospechosas a la API y acciones similares. Para obtener más información, consulte Datos en un gráfico de comportamiento en la documentación de Detective.

sistema big-endian

Un sistema que almacena primero el byte más significativo. Consulte también endianidad.

clasificación binaria

Un proceso que predice un resultado binario (una de las dos clases posibles). Por ejemplo, es posible que su modelo de ML necesite predecir problemas como “¿Este correo electrónico es spam o no es spam?” o “¿Este producto es un libro o un automóvil?”.

filtro de floración

Estructura de datos probabilística y eficiente en términos de memoria que se utiliza para comprobar si un elemento es miembro de un conjunto.

implementación azul/verde

Estrategia de implementación en la que se crean dos entornos separados, pero idénticos. La versión actual de la aplicación se ejecuta en un entorno (azul) y la nueva versión de la aplicación se ejecuta en el otro entorno (verde). Esta estrategia lo ayuda a hacer reversiones rápidas con un impacto mínimo.

bot

Aplicación de software que ejecuta tareas automatizadas a través de Internet y simula la actividad o interacción humana. Algunos bots son útiles o beneficiosos, como los rastreadores web que indexan la información de Internet. Otros bots, conocidos como bots maliciosos, tienen como objetivo causar interrupciones o daños a personas u organizaciones.

botnet

Redes de bots infectadas por malware y que están bajo el control de una sola parte, conocida como pastor de bots u operador de bots. Las botnets son el mecanismo más conocido para escalar los bots y su impacto.

branch

Área contenida de un repositorio de código. La primera rama que se crea en un repositorio es la rama principal. Puede crear una rama nueva a partir de una rama existente y, a continuación, desarrollar características o corregir errores en la rama nueva. Una rama que se genera para crear una característica se denomina comúnmente rama de característica. Cuando la característica se encuentra lista para su lanzamiento, se vuelve a combinar la rama de característica con la rama principal. Para obtener más información, consulte Acerca de las sucursales (GitHub documentación).

acceso de emergencia

En circunstancias excepcionales y mediante un proceso aprobado, es una forma rápida de que un usuario pueda acceder a un Cuenta de AWS sitio al que normalmente no tiene permisos de acceso. Para más información, consulte el indicador Implement break-glass procedures en la guía de AWS Well-Architected.

estrategia de implementación sobre infraestructura existente

La infraestructura existente en su entorno. Al adoptar una estrategia de implementación sobre infraestructura existente para una arquitectura de sistemas, se diseña la arquitectura en función de las limitaciones de los sistemas y la infraestructura actuales. Si está ampliando la infraestructura existente, puede combinar las estrategias de implementación sobre infraestructuras existentes y de implementación desde cero.

caché de búfer

El área de memoria donde se almacenan los datos a los que se accede con más frecuencia.

capacidad empresarial

Lo que hace una empresa para generar valor (por ejemplo, ventas, servicio al cliente o marketing). Las arquitecturas de microservicios y las decisiones de desarrollo pueden estar impulsadas por las capacidades empresariales. Para obtener más información, consulte la sección Organizado en torno a las capacidades empresariales del documento técnico Ejecutar microservicios en contenedores en AWS.

planificación de la continuidad del negocio (BCP)

Plan que aborda el posible impacto de un evento disruptivo, como una migración a gran escala en las operaciones y permite a la empresa reanudar las operaciones rápidamente.

C

CAF

Consulte AWS Cloud Adoption Framework.

implementación canario

Lanzamiento lento e incremental de una versión para los usuarios finales. Cuando tenga mayor confianza en la nueva versión, la implementa y reemplaza la versión actual en su totalidad.

CCoE

Consulte Centro de excelencia en la nube.

CDC

Consulte captura de datos de cambios.

captura de datos de cambio (CDC)

Proceso de seguimiento de los cambios en un origen de datos, como una tabla de base de datos, y registro de los metadatos relacionados con el cambio. Puede utilizar los CDC para diversos fines, como auditar o replicar los cambios en un sistema de destino para mantener la sincronización.

ingeniería del caos

Introducción intencionada de fallos o eventos disruptivos para poner a prueba la resiliencia de un sistema. Puedes usar AWS Fault Injection Service (AWS FIS) para realizar experimentos que estresen tus AWS cargas de trabajo y evalúen su respuesta.

CI/CD

Consulte integración continua y entrega continua.

clasificación

Un proceso de categorización que permite generar predicciones. Los modelos de ML para problemas de clasificación predicen un valor discreto. Los valores discretos siempre son distintos entre sí. Por ejemplo, es posible que un modelo necesite evaluar si hay o no un automóvil en una imagen.

cifrado del cliente

Cifrado de datos localmente, antes de que el objetivo los Servicio de AWS reciba.

Centro de excelencia en la nube (CCoE)

Equipo multidisciplinario que impulsa los esfuerzos de adopción de la nube en toda la organización, incluido el desarrollo de las prácticas recomendadas en la nube, la movilización de recursos, el establecimiento de plazos de migración y la dirección de la organización durante las transformaciones a gran escala. Para obtener más información, consulte las publicaciones de CCo E en el blog de estrategia Nube de AWS empresarial.

computación en la nube

La tecnología en la nube que se utiliza normalmente para la administración de dispositivos de IoT y el almacenamiento de datos de forma remota. La computación en la nube suele estar relacionada con la tecnología de computación de periferia.

modelo operativo en la nube

En una organización de TI, el modelo operativo que se utiliza para crear, madurar y optimizar uno o más entornos de nube. Para obtener más información, consulte Creación de su modelo operativo de nube.

etapas de adopción de la nube

Las siguientes son las cuatro fases por las que suelen pasar las empresas cuando migran a la Nube de AWS:

- Proyecto: ejecución de algunos proyectos relacionados con la nube con fines de prueba de concepto y aprendizaje
- Fundamento: realizar inversiones fundamentales para escalar su adopción de la nube (p. ej., crear una landing zone, definir una CCo E, establecer un modelo de operaciones)

- Migración: migración de aplicaciones individuales
- Reinvención: optimización de productos y servicios e innovación en la nube

Stephen Orban definió estas etapas en la entrada del blog The Journey Toward Cloud-First & the Stages of Adoption en el blog Nube de AWS Enterprise Strategy. Para obtener información sobre su relación con la estrategia de AWS migración, consulte la guía de preparación para la migración.

CMDB

Consulte base de datos de administración de configuración.

repositorio de código

Una ubicación donde el código fuente y otros activos, como documentación, muestras y scripts, se almacenan y actualizan mediante procesos de control de versiones. Algunos repositorios en la nube comunes son GitHub o Bitbucket Cloud. Cada versión del código se denomina rama. En una estructura de microservicios, cada repositorio se encuentra dedicado a una única funcionalidad. Una sola canalización de CI/CD puede utilizar varios repositorios.

caché en frío

Una caché de búfer que está vacía no está bien poblada o contiene datos obsoletos o irrelevantes. Esto afecta al rendimiento, ya que la instancia de la base de datos debe leer desde la memoria principal o el disco, lo que es más lento que leer desde la memoria caché del búfer.

datos fríos

Datos a los que se accede con poca frecuencia y que suelen ser históricos. Al consultar este tipo de datos, normalmente se aceptan consultas lentas. Trasladar estos datos a niveles o clases de almacenamiento de menor rendimiento y menos costosos puede reducir los costos.

visión artificial (CV)

Campo de la IA que utiliza el machine learning para analizar y extraer información de formatos visuales, como imágenes y videos digitales. Por ejemplo, Amazon SageMaker AI proporciona algoritmos de procesamiento de imágenes para CV.

deriva de configuración

En el caso de una carga de trabajo, un cambio en la configuración con respecto al estado esperado. Podría provocar que la carga de trabajo deje de cumplir las normas y, por lo general, es gradual e involuntaria.

base de datos de administración de configuración (CMDB)

Repositorio que almacena y administra información sobre una base de datos y su entorno de TI, incluidos los componentes de hardware y software y sus configuraciones. Por lo general, los datos de una CMDB se utilizan en la etapa de detección y análisis de la cartera de productos durante la migración.

paquete de conformidad

Un conjunto de AWS Config reglas y medidas correctivas que puede reunir para personalizar sus controles de conformidad y seguridad. Puede implementar un paquete de conformidad como una entidad única en una región Cuenta de AWS y, o en una organización, mediante una plantilla YAML. Para obtener más información, consulta los paquetes de conformidad en la documentación. AWS Config

integración y entrega continuas (CI/CD)

El proceso de automatización de las etapas de origen, compilación, prueba, puesta en escena y producción del proceso de publicación del software. CI/CD se describe comúnmente como una canalización. CI/CD puede ayudarlo a automatizar los procesos, mejorar la productividad, mejorar la calidad del código y entregar más rápido. Para obtener más información, consulte Beneficios de la entrega continua. CD también puede significar implementación continua. Para obtener más información, consulte Entrega continua frente a implementación continua.

CV

Consulte visión artificial.

D

datos en reposo

Datos que están estacionarios en la red, como los datos que se encuentran almacenados.

clasificación de datos

Un proceso para identificar y clasificar los datos de su red en función de su importancia y sensibilidad. Es un componente fundamental de cualquier estrategia de administración de riesgos de ciberseguridad porque lo ayuda a determinar los controles de protección y retención adecuados para los datos. La clasificación de datos es un componente del pilar de seguridad

del AWS Well-Architected Framework. Para obtener más información, consulte Clasificación de datos.

deriva de datos

Una variación significativa entre los datos de producción y los datos que se utilizaron para entrenar un modelo de machine learning, o un cambio significativo en los datos de entrada a lo largo del tiempo. La deriva de datos puede reducir la calidad, la precisión y la imparcialidad generales de las predicciones de los modelos de machine learning.

datos en tránsito

Datos que se mueven de forma activa por la red, por ejemplo, entre los recursos de la red.

malla de datos

Marco de arquitectura que proporciona una propiedad de datos distribuida y descentralizada con una administración y una gobernanza centralizadas.

minimización de datos

El principio de recopilar y procesar solo los datos estrictamente necesarios. Practicar la minimización de los datos Nube de AWS puede reducir los riesgos de privacidad, los costos y la huella de carbono de la analítica.

perímetro de datos

Un conjunto de barreras preventivas en su AWS entorno que ayudan a garantizar que solo las identidades confiables accedan a los recursos confiables desde las redes esperadas. Para obtener más información, consulte Crear un perímetro de datos sobre. AWS

preprocesamiento de datos

Transformar los datos sin procesar en un formato que su modelo de ML pueda analizar fácilmente. El preprocesamiento de datos puede implicar eliminar determinadas columnas o filas y corregir los valores faltantes, incoherentes o duplicados.

procedencia de los datos

El proceso de rastrear el origen y el historial de los datos a lo largo de su ciclo de vida, por ejemplo, la forma en que se generaron, transmitieron y almacenaron los datos.

titular de los datos

Persona cuyos datos se recopilan y procesan.

almacenamiento de datos

Sistema de administración de datos que respalda la inteligencia empresarial, como los análisis. Los almacenes de datos suelen contener grandes cantidades de datos históricos y, por lo general, se utilizan para las consultas y los análisis.

lenguaje de definición de datos (DDL)

Instrucciones o comandos para crear o modificar la estructura de tablas y objetos de una base de datos.

lenguaje de manipulación de datos (DML)

Instrucciones o comandos para modificar (insertar, actualizar y eliminar) la información de una base de datos.

DDL

Consulte lenguaje de definición de bases de datos.

conjunto profundo

Combinar varios modelos de aprendizaje profundo para la predicción. Puede utilizar conjuntos profundos para obtener una predicción más precisa o para estimar la incertidumbre de las predicciones.

aprendizaje profundo

Un subcampo del ML que utiliza múltiples capas de redes neuronales artificiales para identificar el mapeo entre los datos de entrada y las variables objetivo de interés.

defense-in-depth

Un enfoque de seguridad de la información en el que se distribuyen cuidadosamente una serie de mecanismos y controles de seguridad en una red informática para proteger la confidencialidad, la integridad y la disponibilidad de la red y de los datos que contiene. Al adoptar esta estrategia AWS, se añaden varios controles en diferentes capas de la AWS Organizations estructura para ayudar a proteger los recursos. Por ejemplo, un defense-in-depth enfoque podría combinar la autenticación multifactorial, la segmentación de la red y el cifrado.

administrador delegado

En AWS Organizations, un servicio compatible puede registrar una cuenta de AWS miembro para administrar las cuentas de la organización y gestionar los permisos de ese servicio. Esta

cuenta se denomina administrador delegado para ese servicio. Para obtener más información y una lista de servicios compatibles, consulte Servicios que funcionan con AWS Organizations en la documentación de AWS Organizations .

Implementación

El proceso de hacer que una aplicación, características nuevas o correcciones de código se encuentren disponibles en el entorno de destino. La implementación abarca implementar cambios en una base de código y, a continuación, crear y ejecutar esa base en los entornos de la aplicación.

entorno de desarrollo

Consulte entorno.

control de detección

Un control de seguridad que se ha diseñado para detectar, registrar y alertar después de que se produzca un evento. Estos controles son una segunda línea de defensa, ya que lo advierten sobre los eventos de seguridad que han eludido los controles preventivos establecidos. Para obtener más información, consulte Controles de detección en Implementación de controles de seguridad en AWS.

asignación de flujos de valor para el desarrollo (DVSM)

Proceso que se utiliza para identificar y priorizar las restricciones que afectan negativamente a la velocidad y la calidad en el ciclo de vida del desarrollo de software. DVSM amplía el proceso de asignación del flujo de valor diseñado originalmente para las prácticas de fabricación ajustada. Se centra en los pasos y los equipos necesarios para crear y transferir valor a través del proceso de desarrollo de software.

gemelo digital

Representación virtual de un sistema del mundo real, como un edificio, una fábrica, un equipo industrial o una línea de producción. Los gemelos digitales son compatibles con el mantenimiento predictivo, la supervisión remota y la optimización de la producción.

tabla de dimensiones

En un esquema en estrella, tabla más pequeña que contiene los atributos de datos sobre los datos cuantitativos en una tabla de hechos. Los atributos de la tabla de dimensiones suelen ser campos de texto o números discretos que se comportan como texto. Estos atributos se suelen utilizar para restringir consultas, filtrarlas y etiquetar los conjuntos de resultados.

desastre

Un evento que impide que una carga de trabajo o un sistema cumplan sus objetivos empresariales en su ubicación principal de implementación. Estos eventos pueden ser desastres naturales, fallos técnicos o el resultado de acciones humanas, como una configuración incorrecta involuntaria o un ataque de malware.

recuperación de desastres (DR)

Estrategia y proceso que utiliza para minimizar el tiempo de inactividad y la pérdida de datos a causa de un desastre. Para obtener más información, consulte Recuperación ante desastres de cargas de trabajo en AWS: Recovery in the Cloud in the AWS Well-Architected Framework.

DML

Consulte lenguaje de manipulación de bases de datos.

diseño basado en el dominio

Un enfoque para desarrollar un sistema de software complejo mediante la conexión de sus componentes a dominios en evolución, o a los objetivos empresariales principales, a los que sirve cada componente. Este concepto lo introdujo Eric Evans en su libro, Diseño impulsado por el dominio: abordando la complejidad en el corazón del software (Boston: Addison-Wesley Professional, 2003). Para obtener información sobre cómo utilizar el diseño basado en dominios con el patrón de higos estranguladores, consulte Modernización gradual de los servicios web antiguos de Microsoft ASP.NET (ASMX) mediante contenedores y Amazon API Gateway.

DR

Consulte recuperación ante desastres.

Detección de desviaciones

Seguimiento de las desviaciones con respecto a una configuración con línea de base. Por ejemplo, puedes usarlo AWS CloudFormation para detectar desviaciones en los recursos del sistema o puedes usarlo AWS Control Tower para detectar cambios en tu landing zone que puedan afectar al cumplimiento de los requisitos de gobierno.

DVSM

Consulte asignación de flujos de valor para el desarrollo.

E

EDA

Consulte análisis de datos de tipo exploratorio.

EDI

Consulte intercambio electrónico de datos.

computación en la periferia

La tecnología que aumenta la potencia de cálculo de los dispositivos inteligentes en la periferia de una red de IoT. En comparación con la computación en la nube, la computación de periferia puede reducir la latencia de la comunicación y mejorar el tiempo de respuesta.

intercambio electrónico de datos (EDI)

Intercambio automatizado de documentos comerciales entre organizaciones. Para más información, consulte ¿Qué es el intercambio electrónico de datos?

cifrado

Proceso de computación que transforma datos de texto plano, que son legibles por humanos, en texto cifrado.

clave de cifrado

Cadena criptográfica de bits aleatorios que se genera mediante un algoritmo de cifrado. Las claves pueden variar en longitud y cada una se ha diseñado para ser impredecible y única.

endianidad

El orden en el que se almacenan los bytes en la memoria del ordenador. Los sistemas big-endianos almacenan primero el byte más significativo. Los sistemas Little-Endian almacenan primero el byte menos significativo.

punto de conexión

Consulte punto de conexión de servicio.

servicio de punto de conexión

Servicio que puede alojar en una nube privada virtual (VPC) para compartir con otros usuarios. Puede crear un servicio de punto final AWS PrivateLink y conceder permisos a otros Cuentas de AWS o a responsables AWS Identity and Access Management (de IAM). Estas cuentas o entidades principales pueden conectarse a su servicio de punto de conexión de forma privada

mediante la creación de puntos de conexión de VPC de interfaz. Para obtener más información, consulte Creación de un servicio de punto de conexión en la documentación de Amazon Virtual Private Cloud (Amazon VPC).

planificación de recursos empresariales (ERP)

Sistema que automatiza y administra los procesos empresariales clave (como la contabilidad, MES y la administración de proyectos) de una empresa.

cifrado de sobre

El proceso de cifrar una clave de cifrado con otra clave de cifrado. Para obtener más información, consulte el cifrado de sobres en la documentación de AWS Key Management Service (AWS KMS).

entorno

Una instancia de una aplicación en ejecución. Los siguientes son los tipos de entornos más comunes en la computación en la nube:

- entorno de desarrollo: instancia de una aplicación en ejecución que solo se encuentra disponible para el equipo principal responsable del mantenimiento de la aplicación. Los entornos de desarrollo se utilizan para probar los cambios antes de promocionarlos a los entornos superiores. Este tipo de entorno a veces se denomina entorno de prueba.
- entornos inferiores: todos los entornos de desarrollo de una aplicación, como los que se utilizan para las compilaciones y pruebas iniciales.
- entorno de producción: instancia de una aplicación en ejecución a la que pueden acceder los usuarios finales. En un CI/CD proceso, el entorno de producción es el último entorno de implementación.
- entornos superiores: todos los entornos a los que pueden acceder usuarios que no sean del equipo de desarrollo principal. Esto puede incluir un entorno de producción, entornos de preproducción y entornos para las pruebas de aceptación por parte de los usuarios.

epopeya

En las metodologías ágiles, son categorías funcionales que ayudan a organizar y priorizar el trabajo. Las epopeyas brindan una descripción detallada de los requisitos y las tareas de implementación. Por ejemplo, las epopeyas AWS de seguridad de CAF incluyen la gestión de identidades y accesos, los controles de detección, la seguridad de la infraestructura, la protección de datos y la respuesta a incidentes. Para obtener más información sobre las epopeyas en la estrategia de migración de AWS , consulte la Guía de implementación del programa.

ERP

Consulte planificación de recursos empresariales.

análisis de datos de tipo exploratorio (EDA)

El proceso de analizar un conjunto de datos para comprender sus características principales. Se recopilan o agregan datos y, a continuación, se realizan las investigaciones iniciales para encontrar patrones, detectar anomalías y comprobar las suposiciones. El EDA se realiza mediante el cálculo de estadísticas resumidas y la creación de visualizaciones de datos.

F

tabla de hechos

Tabla central de un esquema en estrella. Almacena datos cuantitativos sobre operaciones empresariales. Por lo general, una tabla de hechos contiene dos tipos de columnas: las que contienen medidas y las que contienen una clave externa para una tabla de dimensiones.

Fail Fast

Filosofía que utiliza pruebas frecuentes e incrementales para reducir el ciclo de vida del desarrollo. Es una parte fundamental de los enfoques ágiles.

límite de aislamiento de errores

En el Nube de AWS, un límite, como una zona de disponibilidad Región de AWS, un plano de control o un plano de datos, que limita el efecto de una falla y ayuda a mejorar la resiliencia de las cargas de trabajo. Para más información, consulte AWS Fault Isolation Boundaries.

rama de característica

Consulte rama.

características

Los datos de entrada que se utilizan para hacer una predicción. Por ejemplo, en un contexto de fabricación, las características pueden ser imágenes que se capturan periódicamente desde la línea de fabricación.

importancia de las características

La importancia que tiene una característica para las predicciones de un modelo. Por lo general, esto se expresa como una puntuación numérica que se puede calcular mediante diversas

técnicas, como las explicaciones aditivas de Shapley (SHAP) y los gradientes integrados. Para obtener más información, consulte Interpretabilidad del modelo de aprendizaje automático con. AWS

transformación de funciones

Optimizar los datos para el proceso de ML, lo que incluye enriquecer los datos con fuentes adicionales, escalar los valores o extraer varios conjuntos de información de un solo campo de datos. Esto permite que el modelo de ML se beneficie de los datos. Por ejemplo, si divide la fecha del “27 de mayo de 2021 00:15:37” en “jueves”, “mayo”, “2021” y “15”, puede ayudar al algoritmo de aprendizaje a aprender patrones matizados asociados a los diferentes componentes de los datos.

peticiones con pocos pasos

Proporcionar a un LLM una pequeña cantidad de ejemplos que demuestren la tarea y el resultado deseado antes de pedirle que lleve a cabo una tarea similar. Esta técnica es una aplicación del aprendizaje contextual, mediante el que los modelos aprenden a partir de ejemplos (pasos) incrustados en las peticiones. La técnica de peticiones con pocos pasos puede ser eficaz para las tareas que requieren un formato, un razonamiento o un conocimiento del dominio específicos. Consulte también peticiones desde cero.

FGAC

Consulte control de acceso detallado.

control de acceso preciso (FGAC)

El uso de varias condiciones que tienen por objetivo permitir o denegar una solicitud de acceso.

migración relámpago

Método de migración de bases de datos que utiliza la replicación continua de datos mediante la captura de datos de cambio para migrar los datos en el menor tiempo posible, en lugar de utilizar un enfoque gradual. El objetivo es reducir al mínimo el tiempo de inactividad.

FM

Consulte modelo fundacional.

Modelo fundacional (FM)

Una gran red neuronal de aprendizaje profundo que se ha estado entrenando con conjuntos de datos masivos de datos generalizados y sin etiquetar. FMs son capaces de realizar una

amplia variedad de tareas generales, como comprender el lenguaje, generar texto e imágenes y conversar en lenguaje natural. Para más información, consulte ¿Qué son los modelos fundacionales?

G

IA generativa

Subconjunto de modelos de IA que se entrenaron con grandes cantidades de datos y que pueden utilizar una simple petición de texto para crear contenido y artefactos nuevos, como imágenes, videos, texto y audio. Para más información, consulte ¿Qué es la IA generativa?

bloqueo geográfico

Consulte restricciones geográficas.

restricciones geográficas (bloqueo geográfico)

En Amazon CloudFront, una opción para impedir que los usuarios de países específicos accedan a las distribuciones de contenido. Puede utilizar una lista de permitidos o bloqueados para especificar los países aprobados y prohibidos. Para obtener más información, consulta la sección Restringir la distribución geográfica del contenido en la CloudFront documentación.

Flujo de trabajo de Gitflow

Un enfoque en el que los entornos inferiores y superiores utilizan diferentes ramas en un repositorio de código fuente. El flujo de trabajo de Gitflow se considera heredado, mientras que el flujo de trabajo basado en enlaces troncales es el enfoque moderno preferido.

imagen dorada

Instantánea de un sistema o software que se usa como plantilla para implementar nuevas instancias de ese sistema o software. Por ejemplo, en la fabricación, una imagen dorada se puede utilizar para aprovisionar software en varios dispositivos y ayuda a mejorar la velocidad, la escalabilidad y la productividad de las operaciones de fabricación de dispositivos.

estrategia de implementación desde cero

La ausencia de infraestructura existente en un entorno nuevo. Al adoptar una estrategia de implementación desde cero para una arquitectura de sistemas, puede seleccionar todas las tecnologías nuevas sin que estas deban ser compatibles con una infraestructura existente, lo que también se conoce como implementación sobre infraestructura existente. Si está

ampliando la infraestructura existente, puede combinar las estrategias de implementación sobre infraestructuras existentes y de implementación desde cero.

barrera de protección

Una regla de alto nivel que ayuda a regular los recursos, las políticas y el cumplimiento en todas las unidades organizativas (OUs). Las barreras de protección preventivas aplican políticas para garantizar la alineación con los estándares de conformidad. Se implementan mediante políticas de control de servicios y límites de permisos de IAM. Las barreras de protección de detección detectan las vulneraciones de las políticas y los problemas de conformidad, y generan alertas para su corrección. Se implementan mediante Amazon AWS Config AWS Security Hub CSPM GuardDuty AWS Trusted Advisor, Amazon Inspector y AWS Lambda cheques personalizados.

H

HA

Consulte alta disponibilidad.

migración heterogénea de bases de datos

Migración de la base de datos de origen a una base de datos de destino que utilice un motor de base de datos diferente (por ejemplo, de Oracle a Amazon Aurora). La migración heterogénea suele ser parte de un esfuerzo de rediseño de la arquitectura y convertir el esquema puede ser una tarea compleja. AWS ofrece AWS SCT, lo cual ayuda con las conversiones de esquemas.

alta disponibilidad (HA)

La capacidad de una carga de trabajo para funcionar de forma continua, sin intervención, en caso de desafíos o desastres. Los sistemas de alta disponibilidad están diseñados para realizar una conmutación por error automática, ofrecer un rendimiento de alta calidad de forma constante y gestionar diferentes cargas y fallos con un impacto mínimo en el rendimiento.

modernización histórica

Un enfoque utilizado para modernizar y actualizar los sistemas de tecnología operativa (TO) a fin de satisfacer mejor las necesidades de la industria manufacturera. Un histórico es un tipo de base de datos que se utiliza para recopilar y almacenar datos de diversas fuentes en una fábrica.

datos de reserva

Parte de los datos históricos etiquetados que se ocultan de un conjunto de datos que se utiliza para entrenar un modelo de machine learning. Puede utilizar los datos de reserva para evaluar el rendimiento del modelo mediante la comparación de las predicciones del modelo con los datos de reserva.

migración homogénea de bases de datos

Migración de la base de datos de origen a una base de datos de destino que comparte el mismo motor de base de datos (por ejemplo, Microsoft SQL Server a Amazon RDS para SQL Server). La migración homogénea suele formar parte de un esfuerzo para volver a alojar o redefinir la plataforma. Puede utilizar las utilidades de bases de datos nativas para migrar el esquema.

datos recientes

Datos a los que se accede con frecuencia, como datos en tiempo real o datos traslacionales recientes. Por lo general, estos datos requieren un nivel o una clase de almacenamiento de alto rendimiento para proporcionar respuestas rápidas a las consultas.

hotfix

Una solución urgente para un problema crítico en un entorno de producción. Debido a su urgencia, una revisión suele realizarse fuera del flujo de trabajo de DevOps publicación típico.

periodo de hiperatención

Periodo, inmediatamente después de la transición, durante el cual un equipo de migración administra y monitorea las aplicaciones migradas en la nube para solucionar cualquier problema. Por lo general, este periodo dura de 1 a 4 días. Al final del periodo de hiperatención, el equipo de migración suele transferir la responsabilidad de las aplicaciones al equipo de operaciones en la nube.

I

IaC

Consulte infraestructura como código.

políticas basadas en identidades

Política asociada a uno o más directores de IAM que define sus permisos en el entorno. Nube de AWS

aplicación inactiva

Aplicación que utiliza un promedio de CPU y memoria de entre 5 y 20 por ciento durante un periodo de 90 días. En un proyecto de migración, es habitual retirar estas aplicaciones o mantenerlas en las instalaciones.

IIoT

Consulte Internet de las cosas industrial.

infraestructura inmutable

Modelo que implementa una nueva infraestructura para las cargas de trabajo de producción en lugar de actualizar o modificar la infraestructura existente o aplicarle revisiones. Las infraestructuras inmutables son de manera intrínseca más coherentes, fiables y predecibles que las infraestructuras mutables. Para más información, consulte la práctica recomendada Implementación mediante una infraestructura inmutable en el Marco de AWS Well-Architected.

VPC entrante (de entrada)

En una arquitectura de AWS cuentas múltiples, una VPC que acepta, inspecciona y enruta las conexiones de red desde fuera de una aplicación. La arquitectura AWS de referencia de seguridad recomienda configurar la cuenta de red con entradas, salidas e inspección VPCs para proteger la interfaz bidireccional entre la aplicación y el resto de Internet.

migración gradual

Estrategia de transición en la que se migra la aplicación en partes pequeñas en lugar de realizar una transición única y completa. Por ejemplo, puede trasladar inicialmente solo unos pocos microservicios o usuarios al nuevo sistema. Tras comprobar que todo funciona correctamente, puede trasladar microservicios o usuarios adicionales de forma gradual hasta que pueda retirar su sistema heredado. Esta estrategia reduce los riesgos asociados a las grandes migraciones.

Industria 4.0

Término que introdujo Klaus Schwab en 2016 para referirse a la modernización de los procesos de fabricación mediante los avances en la conectividad, los datos en tiempo real, la automatización, el análisis, la IA y el ML.

infraestructura

Todos los recursos y activos que se encuentran en el entorno de una aplicación.

infraestructura como código (IaC)

Proceso de aprovisionamiento y administración de la infraestructura de una aplicación mediante un conjunto de archivos de configuración. La IaC se ha diseñado para ayudarlo a centralizar la administración de la infraestructura, estandarizar los recursos y escalar con rapidez a fin de que los entornos nuevos sean repetibles, fiables y consistentes.

Internet de las cosas industrial (T) IIo

El uso de sensores y dispositivos conectados a Internet en los sectores industriales, como el productivo, el eléctrico, el automotriz, el sanitario, el de las ciencias de la vida y el de la agricultura. Para obtener más información, consulte Creación de una estrategia de transformación digital de la Internet de las cosas (IIoT) industrial.

VPC de inspección

En una arquitectura de AWS cuentas múltiples, una VPC centralizada que gestiona las inspecciones del tráfico de red VPCs entre Internet y las redes locales (en una misma o Regiones de AWS diferente). La arquitectura AWS de referencia de seguridad recomienda configurar su cuenta de red con entrada, salida e inspección VPCs para proteger la interfaz bidireccional entre la aplicación e Internet en general.

Internet de las cosas (IoT)

Red de objetos físicos conectados con sensores o procesadores integrados que se comunican con otros dispositivos y sistemas a través de Internet o de una red de comunicación local. Para obtener más información, consulte ¿Qué es IoT?.

interpretabilidad

Característica de un modelo de machine learning que describe el grado en que un ser humano puede entender cómo las predicciones del modelo dependen de sus entradas. Para obtener más información, consulte Interpretabilidad del modelo de aprendizaje automático con. AWS

IoT

Consulte Internet de las cosas.

biblioteca de información de TI (ITIL)

Conjunto de prácticas recomendadas para ofrecer servicios de TI y alinearlos con los requisitos empresariales. La ITIL proporciona la base para la ITSM.

administración de servicios de TI (ITSM)

Actividades asociadas con el diseño, la implementación, la administración y el soporte de los servicios de TI para una organización. Para obtener información sobre la integración de las operaciones en la nube con las herramientas de ITSM, consulte la Guía de integración de operaciones.

ITIL

Consulte biblioteca de información de TI.

ITSM

Consulte administración de servicios de TI.

L

control de acceso basado en etiquetas (LBAC)

Una implementación del control de acceso obligatorio (MAC) en la que a los usuarios y a los propios datos se les asigna explícitamente un valor de etiqueta de seguridad. La intersección entre la etiqueta de seguridad del usuario y la etiqueta de seguridad de los datos determina qué filas y columnas puede ver el usuario.

zona de aterrizaje

Una landing zone es un AWS entorno multicuenta bien diseñado, escalable y seguro. Este es un punto de partida desde el cual las empresas pueden lanzar e implementar rápidamente cargas de trabajo y aplicaciones con confianza en su entorno de seguridad e infraestructura. Para obtener más información sobre las zonas de aterrizaje, consulte Configuración de un entorno de AWS seguro y escalable con varias cuentas.

modelo de lenguaje de gran tamaño (LLM)

Modelo de IA de aprendizaje profundo que se entrenó previamente con una gran cantidad de datos. Un LLM puede llevar a cabo varias tareas, como responder preguntas, resumir documentos, traducir textos a otros idiomas y completar oraciones. Para obtener más información, consulte Qué son. LLMs

migración grande

Migración de 300 servidores o más.

LBAC

Consulte control de acceso basado en etiquetas.

privilegio mínimo

La práctica recomendada de seguridad que consiste en conceder los permisos mínimos necesarios para realizar una tarea. Para obtener más información, consulte Aplicar permisos de privilegio mínimo en la documentación de IAM.

migrar mediante lift-and-shift

Consulte Las 7 R.

sistema little-endian

Un sistema que almacena primero el byte menos significativo. Consulte también endianidad.

LLM

Consulte modelo de lenguaje de gran tamaño.

entornos inferiores

Consulte entorno.

M

machine learning (ML)

Un tipo de inteligencia artificial que utiliza algoritmos y técnicas para el reconocimiento y el aprendizaje de patrones. El ML analiza y aprende de los datos registrados, como los datos del Internet de las cosas (IoT), para generar un modelo estadístico basado en patrones. Para más información, consulte Machine learning.

rama principal

Consulte rama.

malware

Software diseñado para comprometer la seguridad o la privacidad de la computadora. El malware podría interrumpir los sistemas informáticos, filtrar información confidencial u obtener acceso

no autorizado. Algunos ejemplos de malware son los virus, los gusanos, el ransomware, los troyanos, el spyware y los registradores de pulsaciones de teclas.

Servicios administrados

Servicios de AWS para lo cual AWS opera la capa de infraestructura, el sistema operativo y las plataformas, y se accede a los puntos finales para almacenar y recuperar datos. Amazon Simple Storage Service (Amazon S3) y Amazon DynamoDB son ejemplos de servicios administrados. También se conocen como servicios abstractos.

sistema de ejecución de fabricación (MES)

Sistema de software para seguir, supervisar, documentar y controlar los procesos de producción que convierten las materias primas en productos acabados en la zona de producción.

MAP

Consulte Programa de aceleración de la migración.

mecanismo

Proceso completo mediante el que se crea una herramienta, se impulsa su adopción y, a continuación, se inspeccionan los resultados para hacer ajustes. Un mecanismo es un ciclo que se refuerza y mejora por sí mismo a medida que funciona. Para obtener más información, consulte Creación de mecanismos en el AWS Well-Architected Framework.

cuenta de miembro

Todas las Cuentas de AWS demás cuentas, excepto la de administración, que forman parte de una organización. AWS Organizations Una cuenta no puede pertenecer a más de una organización a la vez.

MES

Consulte sistema de ejecución de fabricación.

Message Queuing Telemetry Transport (MQTT)

Un protocolo de comunicación ligero machine-to-machine (M2M), basado en el patrón de publicación/suscripción, para dispositivos de IoT con recursos limitados.

microservicio

Un servicio pequeño e independiente que se comunica a través de una red bien definida APIs y que, por lo general, es propiedad de equipos pequeños e independientes. Por ejemplo,

un sistema de seguros puede incluir microservicios que se adapten a las capacidades empresariales, como las de ventas o marketing, o a subdominios, como las de compras, reclamaciones o análisis. Los beneficios de los microservicios incluyen la agilidad, la escalabilidad flexible, la facilidad de implementación, el código reutilizable y la resiliencia. Para obtener más información, consulte Integrar microservicios mediante AWS servicios sin servidor.

arquitectura de microservicios

Un enfoque para crear una aplicación con componentes independientes que ejecutan cada proceso de la aplicación como un microservicio. Estos microservicios se comunican a través de una interfaz bien definida mediante un uso ligero. APIs Cada microservicio de esta arquitectura se puede actualizar, implementar y escalar para satisfacer la demanda de funciones específicas de una aplicación. Para obtener más información, consulte Implementación de microservicios en. AWS

Programa de aceleración de la migración (MAP)

Un AWS programa que proporciona soporte de consultoría, formación y servicios para ayudar a las organizaciones a crear una base operativa sólida para migrar a la nube y para ayudar a compensar el costo inicial de las migraciones. El MAP incluye una metodología de migración para ejecutar las migraciones antiguas de forma metódica y un conjunto de herramientas para automatizar y acelerar los escenarios de migración más comunes.

migración a escala

Proceso de transferencia de la mayoría de la cartera de aplicaciones a la nube en oleadas, con más aplicaciones desplazadas a un ritmo más rápido en cada oleada. En esta fase, se utilizan las prácticas recomendadas y las lecciones aprendidas en las fases anteriores para implementar una fábrica de migración de equipos, herramientas y procesos con el fin de agilizar la migración de las cargas de trabajo mediante la automatización y la entrega ágil. Esta es la tercera fase de la estrategia de migración de AWS.

fábrica de migración

Equipos multifuncionales que agilizan la migración de las cargas de trabajo mediante enfoques automatizados y ágiles. Los equipos de las fábricas de migración suelen incluir a analistas y propietarios de operaciones, empresas, ingenieros de migración, desarrolladores y DevOps profesionales que trabajan a pasos agigantados. Entre el 20 y el 50 por ciento de la cartera de aplicaciones empresariales se compone de patrones repetidos que pueden optimizarse mediante un enfoque de fábrica. Para obtener más información, consulte la discusión sobre las fábricas de migración y la Guía de fábricas de migración a la nube en este contenido.

metadatos de migración

Información sobre la aplicación y el servidor que se necesita para completar la migración. Cada patrón de migración requiere un conjunto diferente de metadatos de migración. Algunos ejemplos de metadatos de migración son la subred de destino, el grupo de seguridad y AWS la cuenta.

patrón de migración

Tarea de migración repetible que detalla la estrategia de migración, el destino de la migración y la aplicación o el servicio de migración utilizados. Ejemplo: rehospede la migración a Amazon EC2 AWS con Application Migration Service.

Migration Portfolio Assessment (MPA)

Herramienta en línea que proporciona información a fin de validar los argumentos comerciales necesarios para migrar a la Nube de AWS. La MPA ofrece una evaluación detallada de la cartera (adecuación del tamaño de los servidores, precios, comparaciones del costo total de propiedad, análisis de los costos de migración), así como una planificación de la migración (análisis y recopilación de datos de aplicaciones, agrupación de aplicaciones, priorización de la migración y planificación de oleadas). La herramienta MPA (requiere iniciar sesión) está disponible de forma gratuita para todos los AWS consultores y consultores de los socios de APN.

Evaluación de la preparación para la migración (MRA)

Proceso que consiste en obtener información sobre el estado de preparación de una organización para la nube, identificar sus puntos fuertes y débiles y elaborar un plan de acción para cerrar las brechas identificadas mediante el AWS CAF. Para obtener más información, consulte la Guía de preparación para la migración. La MRA es la primera fase de la estrategia de migración de AWS.

estrategia de migración

Enfoque utilizado para migrar una carga de trabajo a la Nube de AWS. Para más información, consulte la entrada Las 7 R de este glosario y también Mobilize your organization to accelerate large-scale migrations.

ML

Consulte machine learning.

modernización

Transformar una aplicación obsoleta (antigua o monolítica) y su infraestructura en un sistema ágil, elástico y de alta disponibilidad en la nube para reducir los gastos, aumentar la eficiencia

y aprovechar las innovaciones. Para más información, consulte Strategy for modernizing applications in the Nube de AWS.

evaluación de la preparación para la modernización

Evaluación que ayuda a determinar la preparación para la modernización de las aplicaciones de una organización; identifica los beneficios, los riesgos y las dependencias; y determina qué tan bien la organización puede soportar el estado futuro de esas aplicaciones. El resultado de la evaluación es un esquema de la arquitectura objetivo, una hoja de ruta que detalla las fases de desarrollo y los hitos del proceso de modernización y un plan de acción para abordar las brechas identificadas. Para más información, consulte Evaluating modernization readiness for applications in the Nube de AWS.

aplicaciones monolíticas (monolitos)

Aplicaciones que se ejecutan como un único servicio con procesos estrechamente acoplados. Las aplicaciones monolíticas presentan varios inconvenientes. Si una característica de la aplicación experimenta un aumento en la demanda, se debe escalar toda la arquitectura. Agregar o mejorar las características de una aplicación monolítica también se vuelve más complejo a medida que crece la base de código. Para solucionar problemas con la aplicación, puede utilizar una arquitectura de microservicios. Para obtener más información, consulte Descomposición de monolitos en microservicios.

MPA

Consulte Migration Portfolio Assessment.

MQTT

Consulte Message Queuing Telemetry Transport.

clasificación multiclase

Un proceso que ayuda a generar predicciones para varias clases (predice uno de más de dos resultados). Por ejemplo, un modelo de ML podría preguntar “¿Este producto es un libro, un automóvil o un teléfono?” o “¿Qué categoría de productos es más interesante para este cliente?”.

infraestructura mutable

Modelo que actualiza y modifica la infraestructura actual para las cargas de trabajo de producción. Para mejorar la coherencia, la fiabilidad y la previsibilidad, el AWS Well-Architected Framework recomienda el uso de una infraestructura inmutable como práctica recomendada.

O

OAC

Consulte control de acceso de origen.

OAI

Consulte identidad de acceso de origen.

OCM

Consulte administración del cambio organizacional.

migración fuera de línea

Método de migración en el que la carga de trabajo de origen se elimina durante el proceso de migración. Este método implica un tiempo de inactividad prolongado y, por lo general, se utiliza para cargas de trabajo pequeñas y no críticas.

OI

Consulte integración de operaciones.

OLA

Consulte acuerdo de nivel operativo.

migración en línea

Método de migración en el que la carga de trabajo de origen se copia al sistema de destino sin que se desconecte. Las aplicaciones que están conectadas a la carga de trabajo pueden seguir funcionando durante la migración. Este método implica un tiempo de inactividad nulo o mínimo y, por lo general, se utiliza para cargas de trabajo de producción críticas.

OPC-UA

Consulte Open Process Communications: arquitectura unificada.

Open Process Communications: arquitectura unificada (OPC-UA)

Un protocolo de machine-to-machine comunicación (M2M) para la automatización industrial. OPC-UA establece un estándar de interoperabilidad con esquemas de autenticación, autorización y cifrado de datos.

acuerdo de nivel operativo (OLA)

Acuerdo que aclara lo que los grupos de TI operativos se comprometen a ofrecerse entre sí, para respaldar un acuerdo de nivel de servicio (SLA).

revisión de la preparación operativa (ORR)

Lista de comprobación de preguntas y prácticas recomendadas asociadas que son útiles para comprender, evaluar, prevenir o reducir el alcance de los incidentes y posibles errores. Para más información, consulte Operational Readiness Reviews (ORR) en el Marco de AWS Well-Architected.

tecnología operativa (TO)

Sistemas de hardware y software que funcionan con el entorno físico para controlar las operaciones, los equipos y la infraestructura industriales. En el sector de la fabricación, la integración de los sistemas de TO y tecnología de la información (TI) es un enfoque clave para las transformaciones de la industria 4.0.

integración de operaciones (OI)

Proceso de modernización de las operaciones en la nube, que implica la planificación de la preparación, la automatización y la integración. Para obtener más información, consulte la Guía de integración de las operaciones.

registro de seguimiento organizativo

Un registro creado por y AWS CloudTrail que registra todos los eventos para todos los miembros Cuentas de AWS de una organización. AWS Organizations Este registro de seguimiento se crea en cada Cuenta de AWS que forma parte de la organización y realiza un seguimiento de la actividad en cada cuenta. Para obtener más información, consulte Crear un registro para una organización en la CloudTrail documentación.

administración del cambio organizacional (OCM)

Marco para administrar las transformaciones empresariales importantes y disruptivas desde la perspectiva de las personas, la cultura y el liderazgo. La OCM ayuda a las empresas a prepararse para nuevos sistemas y estrategias y a realizar la transición a ellos, al acelerar la adopción de cambios, abordar los problemas de transición e impulsar cambios culturales y organizacionales. En la estrategia de AWS migración, este marco se denomina aceleración de personal, debido a la velocidad de cambio que requieren los proyectos de adopción de la nube. Para obtener más información, consulte la Guía de OCM.

control de acceso de origen (OAC)

En CloudFront, una opción mejorada para restringir el acceso y proteger el contenido del Amazon Simple Storage Service (Amazon S3). El OAC admite todos los buckets de S3 Regiones de AWS, el cifrado del lado del servidor AWS KMS (SSE-KMS) y las solicitudes dinámicas `PUT` y `DELETE` dirigidas al bucket de S3.

identidad de acceso de origen (OAI)

En CloudFront, una opción para restringir el acceso y proteger el contenido de Amazon S3. Cuando utiliza OAI, CloudFront crea un principal con el que Amazon S3 puede autenticarse. Los directores autenticados solo pueden acceder al contenido de un bucket de S3 a través de una distribución específica. CloudFront Consulte también el OAC, que proporciona un control de acceso más detallado y mejorado.

ORR

Consulte revisión de la preparación operativa.

OT

Consulte tecnología operativa.

VPC saliente (de salida)

En una arquitectura de AWS cuentas múltiples, una VPC que gestiona las conexiones de red que se inician desde una aplicación. La arquitectura AWS de referencia de seguridad recomienda configurar la cuenta de red con entradas, salidas e inspección VPCs para proteger la interfaz bidireccional entre la aplicación e Internet en general.

P

límite de permisos

Una política de administración de IAM que se adjunta a las entidades principales de IAM para establecer los permisos máximos que puede tener el usuario o el rol. Para obtener más información, consulte Límites de permisos en la documentación de IAM.

información de identificación personal (PII)

Información que, vista directamente o combinada con otros datos relacionados, puede utilizarse para deducir de manera razonable la identidad de una persona. Algunos ejemplos de información de identificación personal son los nombres, las direcciones y la información de contacto.

PII

Consulte información de identificación personal.

manual de estrategias

Conjunto de pasos predefinidos que capturan el trabajo asociado a las migraciones, como la entrega de las funciones de operaciones principales en la nube. Un manual puede adoptar la forma de scripts, manuales de procedimientos automatizados o resúmenes de los procesos o pasos necesarios para operar un entorno modernizado.

PLC

Consulte controlador lógico programable.

PLM

Consulte administración del ciclo de vida del producto.

policy

Objeto que puede definir permisos (consulte política basada en identidad), especificar las condiciones de acceso (consulte política basada en recursos) o definir los permisos máximos para todas las cuentas de una organización de AWS Organizations (consulte política de control de servicio).

persistencia políglota

Elegir de forma independiente la tecnología de almacenamiento de datos de un microservicio en función de los patrones de acceso a los datos y otros requisitos. Si sus microservicios tienen la misma tecnología de almacenamiento de datos, pueden enfrentarse a desafíos de implementación o experimentar un rendimiento deficiente. Los microservicios se implementan más fácilmente y logran un mejor rendimiento y escalabilidad si utilizan el almacén de datos que mejor se adapte a sus necesidades.

evaluación de cartera

Proceso de detección, análisis y priorización de la cartera de aplicaciones para planificar la migración. Para obtener más información, consulte la Evaluación de la preparación para la migración.

predicate

Condición de consulta que devuelve `true` o `false`. En general, se encuentra en una cláusula `WHERE`.

inserción de predicados

Técnica de optimización de consultas en bases de datos que filtra los datos de la consulta antes de transferirlos. Esta técnica reduce la cantidad de datos de la base de datos relacional que se tienen que recuperar y procesar. Además, mejora el rendimiento de las consultas.

control preventivo

Un control de seguridad diseñado para evitar que ocurra un evento. Estos controles son la primera línea de defensa para evitar el acceso no autorizado o los cambios no deseados en la red. Para obtener más información, consulte Controles preventivos en Implementación de controles de seguridad en AWS.

entidad principal

Una entidad AWS que puede realizar acciones y acceder a los recursos. Esta entidad suele ser un usuario raíz para un Cuenta de AWS rol de IAM o un usuario. Para obtener más información, consulte Entidad principal en Términos y conceptos de roles en la documentación de IAM.

Privacidad desde el diseño

Enfoque de ingeniería de sistemas que tiene en cuenta la privacidad durante todo el proceso de desarrollo.

zonas alojadas privadas

Un contenedor que contiene información sobre cómo desea que Amazon Route 53 responda a las consultas de DNS de un dominio y sus subdominios dentro de uno o más VPCs. Para obtener más información, consulte Uso de zonas alojadas privadas en la documentación de Route 53.

control proactivo

Control de seguridad que se diseñó para evitar la implementación de recursos que no cumplan con la normativa. Estos controles analizan los recursos antes de aprovisionarlos. Si el recurso no cumple con los requisitos del control, no se aprovisiona. Para obtener más información, consulte la guía de referencia de controles en la AWS Control Tower documentación y consulte Controles proactivos en la sección Implementación de controles de seguridad en AWS.

administración del ciclo de vida del producto (PLM)

Administración de los datos y los procesos de un producto a lo largo de todo su ciclo de vida, desde el diseño, el desarrollo y el lanzamiento, pasando por el crecimiento y la madurez, hasta la reducción de su uso y su retirada.

entorno de producción

Consulte entorno.

controlador lógico programable (PLC)

En el sector de la fabricación, computadora adaptable y altamente fiable que supervisa las máquinas y automatiza los procesos de fabricación.

encadenamiento de peticiones

Uso de la salida de una petición de LLM como entrada para la siguiente petición a fin de generar mejores respuestas. Esta técnica se utiliza para dividir una tarea compleja en tareas secundarias o para refinar o ampliar de forma iterativa una respuesta preliminar. Ayuda a mejorar la precisión y la relevancia de las respuestas de un modelo y permite obtener resultados más detallados y personalizados.

seudonimización

El proceso de reemplazar los identificadores personales de un conjunto de datos por valores de marcadores de posición. La seudonimización puede ayudar a proteger la privacidad personal. Los datos seudonimizados siguen considerándose datos personales.

publish/subscribe (pub/sub)

Patrón que permite establecer comunicaciones asíncronas entre microservicios para mejorar la escalabilidad y la capacidad de respuesta. Por ejemplo, en un MES basado en microservicios, un microservicio puede publicar mensajes de eventos en un canal al que se pueden suscribir otros microservicios. El sistema puede agregar nuevos microservicios sin cambiar el servicio de publicación.

Q

plan de consulta

Serie de pasos, como instrucciones, que se utilizan para acceder a los datos de un sistema de base de datos relacional SQL.

regresión del plan de consulta

El optimizador de servicios de la base de datos elige un plan menos óptimo que antes de un cambio determinado en el entorno de la base de datos. Los cambios en estadísticas,

restricciones, configuración del entorno, enlaces de parámetros de consultas y actualizaciones del motor de base de datos PostgreSQL pueden provocar una regresión del plan.

R

Matriz RACI

Consulte responsable, fiable, consultada e informada (RACI).

RAG

Consulte generación aumentada por recuperación.

ransomware

Software malicioso que se ha diseñado para bloquear el acceso a un sistema informático o a los datos hasta que se efectúe un pago.

Matriz RASCI

Consulte responsable, fiable, consultada e informada (RACI).

RCAC

Consulte control de acceso por filas y columnas.

réplica de lectura

Una copia de una base de datos que se utiliza con fines de solo lectura. Puede enrutar las consultas a la réplica de lectura para reducir la carga en la base de datos principal.

rediseñar

Consulte Las 7 R.

objetivo de punto de recuperación (RPO)

La cantidad de tiempo máximo aceptable desde el último punto de recuperación de datos. Esto determina qué se considera una pérdida de datos aceptable entre el último punto de recuperación y la interrupción del servicio.

objetivo de tiempo de recuperación (RTO)

La demora máxima aceptable entre la interrupción del servicio y el restablecimiento del servicio.

refactorizar

Consulte Las 7 R.

Region

Conjunto de AWS recursos en un área geográfica. Cada uno Región de AWS está aislado e independiente de los demás para proporcionar tolerancia a las fallas, estabilidad y resiliencia. Para más información, consulte Specify which Regiones de AWS your account can use.

regresión

Una técnica de ML que predice un valor numérico. Por ejemplo, para resolver el problema de "¿A qué precio se venderá esta casa?", un modelo de ML podría utilizar un modelo de regresión lineal para predecir el precio de venta de una vivienda en función de datos conocidos sobre ella (por ejemplo, los metros cuadrados).

volver a alojar

Consulte Las 7 R.

versión

En un proceso de implementación, el acto de promover cambios en un entorno de producción.

reubicar

Consulte Las 7 R.

redefinir la plataforma

Consulte Las 7 R.

recomprar

Consulte Las 7 R.

resiliencia

Capacidad de una aplicación para resistir interrupciones o recuperarse de ellas. Al planificar la resiliencia en la Nube de AWS, la alta disponibilidad y la recuperación ante desastres son consideraciones comunes. Para más información, consulte Resiliencia en la Nube de AWS.

política basada en recursos

Una política asociada a un recurso, como un bucket de Amazon S3, un punto de conexión o una clave de cifrado. Este tipo de política especifica a qué entidades principales se les permite el acceso, las acciones compatibles y cualquier otra condición que deba cumplirse.

matriz responsable, confiable, consultada e informada (RACI)

Una matriz que define las funciones y responsabilidades de todas las partes involucradas en las actividades de migración y las operaciones de la nube. El nombre de la matriz se deriva de los tipos de responsabilidad definidos en la matriz: responsable (R), contable (A), consultado (C) e informado (I). El tipo de soporte (S) es opcional. Si incluye el soporte, la matriz se denomina matriz RASCI y, si la excluye, se denomina matriz RACI.

control receptivo

Un control de seguridad que se ha diseñado para corregir los eventos adversos o las desviaciones con respecto a su base de seguridad. Para obtener más información, consulte Controles receptivos en Implementación de controles de seguridad en AWS.

retain

Consulte Las 7 R.

retirar

Consulte Las 7 R.

Generación aumentada de recuperación (RAG)

Tecnología de IA generativa mediante la que un LLM hace referencia a un origen de datos autorizado que se encuentra fuera de sus orígenes de datos de entrenamiento antes de generar una respuesta. Por ejemplo, un modelo de RAG podría hacer una búsqueda semántica en la base de conocimientos o en los datos personalizados de una organización. Para más información, consulte ¿Qué es RAG (generación aumentada por recuperación)?

rotación

Proceso mediante el que periódicamente se actualiza un secreto para que resulte más difícil que un atacante pueda acceder a las credenciales.

control de acceso por filas y columnas (RCAC)

El uso de expresiones SQL básicas y flexibles que tienen reglas de acceso definidas. El RCAC consta de permisos de fila y máscaras de columnas.

RPO

Consulte objetivo de punto de recuperación.

RTO

Consulte objetivo de tiempo de recuperación.

manual de procedimientos

Conjunto de procedimientos manuales o automatizados necesarios para realizar una tarea específica. Por lo general, se diseñan para agilizar las operaciones o los procedimientos repetitivos con altas tasas de error.

S

SAML 2.0

Un estándar abierto que utilizan muchos proveedores de identidad (IdPs). Esta función permite el inicio de sesión único (SSO) federado, de modo que los usuarios pueden iniciar sesión Consola de administración de AWS o llamar a las operaciones de la AWS API sin tener que crear un usuario en IAM para todos los miembros de la organización. Para obtener más información sobre la federación basada en SAML 2.0, consulte Acerca de la federación basada en SAML 2.0 en la documentación de IAM.

SCADA

Consulte control de supervisión y adquisición de datos.

SCP

Consulte política de control de servicio.

secreta

En AWS Secrets Manager, información confidencial o restringida, como una contraseña o credenciales de usuario, que se almacena de forma cifrada. Se compone del valor del secreto y de sus metadatos. El valor del secreto puede ser binario, una sola cadena o varias cadenas. Para más información, consulte What's in a Secrets Manager secret? en la documentación de Secrets Manager.

seguridad desde el diseño

Enfoque de ingeniería de sistemas que tiene en cuenta la seguridad durante todo el proceso de desarrollo.

control de seguridad

Barrera de protección técnica o administrativa que impide, detecta o reduce la capacidad de un agente de amenazas para aprovechar una vulnerabilidad de seguridad. Existen cuatro tipos de controles de seguridad principales: preventivos, de detección, de respuesta y proactivos.

refuerzo de la seguridad

Proceso de reducir la superficie expuesta a ataques para hacerla más resistente a los ataques. Esto puede incluir acciones, como la eliminación de los recursos que ya no se necesitan, la implementación de prácticas recomendadas de seguridad consistente en conceder privilegios mínimos o la desactivación de características innecesarias en los archivos de configuración.

sistema de información sobre seguridad y administración de eventos (SIEM)

Herramientas y servicios que combinan sistemas de administración de información sobre seguridad (SIM) y de administración de eventos de seguridad (SEM). Un sistema de SIEM recopila, monitorea y analiza los datos de servidores, redes, dispositivos y otras fuentes para detectar amenazas y brechas de seguridad y generar alertas.

automatización de la respuesta de seguridad

Acción predefinida y programada que está diseñada para responder automáticamente a un evento de seguridad o corregirlo. Estas automatizaciones sirven como controles de seguridad preventivos o adaptables que le ayudan a implementar las mejores prácticas AWS de seguridad. La modificación de un grupo de seguridad de VPC, la aplicación de revisiones a una instancia de Amazon EC2 o la rotación de credenciales son algunos ejemplos de acciones de respuesta automatizadas.

cifrado del servidor

Cifrado de los datos en su destino, por parte de Servicio de AWS quien los recibe.

política de control de servicio (SCP)

Política que proporciona un control centralizado de los permisos de todas las cuentas de una organización en AWS Organizations. SCPs defina barreras o establezca límites a las acciones que un administrador puede delegar en usuarios o roles. Puede utilizarlas SCPs como listas de permitidos o rechazados para especificar qué servicios o acciones están permitidos o prohibidos. Para obtener más información, consulte las políticas de control de servicios en la AWS Organizations documentación.

punto de enlace de servicio

La URL del punto de entrada de un Servicio de AWS. Para conectarse mediante programación a un servicio de destino, puede utilizar un punto de conexión. Para obtener más información, consulte Puntos de conexión de Servicio de AWS en Referencia general de AWS.

acuerdo de nivel de servicio (SLA)

Acuerdo que aclara lo que un equipo de TI se compromete a ofrecer a los clientes, como el tiempo de actividad y el rendimiento del servicio.

indicador de nivel de servicio (SLI)

Medición de un aspecto del rendimiento de un servicio, como la tasa de errores, la disponibilidad o el rendimiento.

objetivo de nivel de servicio (SLO)

Métrica objetivo que representa el estado de un servicio medido mediante un indicador de nivel de servicio.

modelo de responsabilidad compartida

Un modelo que describe la responsabilidad con AWS la que compartes la seguridad y el cumplimiento de la nube. AWS es responsable de la seguridad de la nube, mientras que usted es responsable de la seguridad en la nube. Para obtener más información, consulte el Modelo de responsabilidad compartida.

SIEM

Consulte sistema de administración de eventos e información de seguridad.

único punto de error (SPOF)

Error en un único componente crítico de una aplicación que puede interrumpir el sistema.

SLA

Consulte acuerdo de nivel de servicio.

SLI

Consulte indicador de nivel de servicio.

SLO

Consulte objetivo de nivel de servicio.

split-and-seed modelo

Un patrón para escalar y acelerar los proyectos de modernización. A medida que se definen las nuevas funciones y los lanzamientos de los productos, el equipo principal se divide para

crear nuevos equipos de productos. Esto ayuda a ampliar las capacidades y los servicios de su organización, mejora la productividad de los desarrolladores y apoya la innovación rápida. Para más información, consulte Phased approach to modernizing applications in the Nube de AWS.

SPOF

Consulte único punto de error.

esquema en estrella

Estructura organizativa de una base de datos que utiliza una tabla de hechos de gran tamaño para almacenar datos transaccionales o medidos y una o varias tablas dimensionales más pequeñas para almacenar los atributos de los datos. Esta estructura está diseñada para utilizarse en un almacén de datos o con fines de inteligencia empresarial.

patrón de higo estrangulador

Un enfoque para modernizar los sistemas monolíticos mediante la reescritura y el reemplazo gradual de las funciones del sistema hasta que se pueda desmantelar el sistema heredado. Este patrón utiliza la analogía de una higuera que crece hasta convertirse en un árbol estable y, finalmente, se apodera y reemplaza a su host. El patrón fue presentado por Martin Fowler como una forma de gestionar el riesgo al reescribir sistemas monolíticos. Para ver un ejemplo con la aplicación de este patrón, consulte Modernización gradual de los servicios web antiguos de Microsoft ASP.NET (ASMX) mediante contenedores y Amazon API Gateway.

subred

Un intervalo de direcciones IP en la VPC. Una subred debe residir en una sola zona de disponibilidad.

control de supervisión y adquisición de datos (SCADA)

En el sector de la fabricación, sistema que utiliza hardware y software para supervisar los activos físicos y las operaciones de producción.

cifrado simétrico

Un algoritmo de cifrado que utiliza la misma clave para cifrar y descifrar los datos.

pruebas sintéticas

Prueba de un sistema de manera que simule las interacciones de los usuarios para detectar posibles problemas o supervisar el rendimiento. Puede usar Amazon CloudWatch Synthetics para crear estas pruebas.

petición del sistema

Técnica para proporcionar contexto, instrucciones o pautas a un LLM para dirigir su comportamiento. Las peticiones del sistema ayudan a establecer el contexto y las reglas para las interacciones con los usuarios.

T

etiquetas

Pares clave-valor que actúan como metadatos para organizar los recursos. AWS Las etiquetas pueden ayudar a administrar, identificar, organizar, buscar y filtrar recursos de . Para obtener más información, consulte Etiquetado de los recursos de AWS.

variable de destino

El valor que intenta predecir en el ML supervisado. Esto también se conoce como variable de resultado. Por ejemplo, en un entorno de fabricación, la variable objetivo podría ser un defecto del producto.

lista de tareas

Herramienta que se utiliza para hacer un seguimiento del progreso mediante un manual de procedimientos. La lista de tareas contiene una descripción general del manual de procedimientos y una lista de las tareas generales que deben completarse. Para cada tarea general, se incluye la cantidad estimada de tiempo necesario, el propietario y el progreso.

entorno de prueba

Consulte entorno.

entrenamiento

Proporcionar datos de los que pueda aprender su modelo de ML. Los datos de entrenamiento deben contener la respuesta correcta. El algoritmo de aprendizaje encuentra patrones en los datos de entrenamiento que asignan los atributos de los datos de entrada al destino (la respuesta que desea predecir). Genera un modelo de ML que captura estos patrones. Luego, el modelo de ML se puede utilizar para obtener predicciones sobre datos nuevos para los que no se conoce el destino.

puerta de enlace de tránsito

Un centro de tránsito de red que puede usar para interconectar sus redes con VPCs las locales. Para obtener más información, consulte Qué es una pasarela de tránsito en la AWS Transit Gateway documentación.

flujo de trabajo basado en enlaces troncales

Un enfoque en el que los desarrolladores crean y prueban características de forma local en una rama de característica y, a continuación, combinan esos cambios en la rama principal. Luego, la rama principal se adapta a los entornos de desarrollo, preproducción y producción, de forma secuencial.

acceso de confianza

Otorgar permisos a un servicio que especifique para realizar tareas en su organización AWS Organizations y en sus cuentas en su nombre. El servicio de confianza crea un rol vinculado al servicio en cada cuenta, cuando ese rol es necesario, para realizar las tareas de administración por usted. Para obtener más información, consulte AWS Organizations Utilización con otros AWS servicios en la AWS Organizations documentación.

ajuste

Cambiar aspectos de su proceso de formación a fin de mejorar la precisión del modelo de ML. Por ejemplo, puede entrenar el modelo de ML al generar un conjunto de etiquetas, incorporar etiquetas y, luego, repetir estos pasos varias veces con diferentes ajustes para optimizar el modelo.

equipo de dos pizzas

Un DevOps equipo pequeño al que puedes alimentar con dos pizzas. Un equipo formado por dos integrantes garantiza la mejor oportunidad posible de colaboración en el desarrollo de software.

U

incertidumbre

Un concepto que hace referencia a información imprecisa, incompleta o desconocida que puede socavar la fiabilidad de los modelos predictivos de ML. Hay dos tipos de incertidumbre: la incertidumbre epistémica se debe a datos limitados e incompletos, mientras que la incertidumbre aleatoria se debe al ruido y la aleatoriedad inherentes a los datos. Para más información, consulte la guía Cuantificación de la incertidumbre en los sistemas de aprendizaje profundo.

tareas indiferenciadas

También conocido como tareas arduas, es el trabajo que es necesario para crear y operar una aplicación, pero que no proporciona un valor directo al usuario final ni proporciona una ventaja competitiva. Algunos ejemplos de tareas indiferenciadas son la adquisición, el mantenimiento y la planificación de la capacidad.

entornos superiores

Consulte entorno.

V

succión

Una operación de mantenimiento de bases de datos que implica limpiar después de las actualizaciones incrementales para recuperar espacio de almacenamiento y mejorar el rendimiento.

control de versión

Procesos y herramientas que realizan un seguimiento de los cambios, como los cambios en el código fuente de un repositorio.

Emparejamiento de VPC

Una conexión entre dos VPCs que le permite enrutar el tráfico mediante direcciones IP privadas. Para obtener más información, consulte ¿Qué es una interconexión de VPC? en la documentación de Amazon VPC.

vulnerabilidad

Defecto de software o hardware que pone en peligro la seguridad del sistema.

W

caché caliente

Un búfer caché que contiene datos actuales y relevantes a los que se accede con frecuencia. La instancia de base de datos puede leer desde la caché del búfer, lo que es más rápido que leer desde la memoria principal o el disco.

datos templados

Datos a los que el acceso es infrecuente. Al consultar este tipo de datos, normalmente se aceptan consultas moderadamente lentas.

función de ventana

Función SQL que hace un cálculo en un grupo de filas que se relacionan de alguna manera con el registro actual. Las funciones de ventana son útiles para las tareas de procesamiento, como calcular una media móvil o acceder al valor de las filas en función de la posición relativa de la fila actual.

carga de trabajo

Conjunto de recursos y código que ofrece valor comercial, como una aplicación orientada al cliente o un proceso de backend.

flujo de trabajo

Grupos funcionales de un proyecto de migración que son responsables de un conjunto específico de tareas. Cada flujo de trabajo es independiente, pero respalda a los demás flujos de trabajo del proyecto. Por ejemplo, el flujo de trabajo de la cartera es responsable de priorizar las aplicaciones, planificar las oleadas y recopilar los metadatos de migración. El flujo de trabajo de la cartera entrega estos recursos al flujo de trabajo de migración, que luego migra los servidores y las aplicaciones.

WORM

Consulte escritura única y lectura múltiple.

WQF

Consulte AWS Workload Qualification Framework.

escritura única y lectura múltiple (WORM)

Modelo de almacenamiento que escribe los datos una sola vez y evita que se eliminen o modifiquen. Los usuarios autorizados pueden leer los datos tantas veces como sea necesario, pero no los pueden cambiar. Esta infraestructura de almacenamiento de datos se considera inmutable.

Z

ataque de día cero

Ataque, normalmente de malware, que se aprovecha de una vulnerabilidad de día cero.

vulnerabilidad de día cero

Un defecto o una vulnerabilidad sin mitigación en un sistema de producción. Los agentes de amenazas pueden usar este tipo de vulnerabilidad para atacar el sistema. Los desarrolladores suelen darse cuenta de la vulnerabilidad a raíz del ataque.

peticiones desde cero

Proporcionar a un LLM instrucciones para llevar a cabo una tarea, pero sin ejemplos (pasos) que puedan ayudar a guiarlo. El LLM debe usar los conocimientos del entrenamiento previo para llevar a cabo la tarea. La eficacia de la petición desde cero depende de la complejidad de la tarea y de la calidad de la petición. Consulte también peticiones con pocos pasos.

aplicación zombi

Aplicación que utiliza un promedio de CPU y memoria menor al 5 por ciento. En un proyecto de migración, es habitual retirar estas aplicaciones.

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.