Autorisation SaaS multi-locataires et contrôle d'accès aux API : options de mise en œuvre et meilleures pratiques

AWS Conseils prescriptifs

AWS Conseils prescriptifs: Autorisation SaaS multi-locataires et contrôle d'accès aux API : options de mise en œuvre et meilleures pratiques

Table of Contents

Autorisation SaaS multi-locataires et contrôle d'accès aux API : options de mise en œuvre et meilleures pratiques

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

Mai 2024 (historique du document)

Les autorisations et le contrôle d'accès aux API constituent un défi pour de nombreuses applications logicielles, en particulier pour les applications SaaS (Software as a Service) mutualisées. Cette complexité est évidente si l'on considère la prolifération des microservices APIs qui doivent être sécurisés et le grand nombre de conditions d'accès qui découlent des différents locataires, des caractéristiques des utilisateurs et de l'état des applications. Pour résoudre efficacement ces problèmes, une solution doit renforcer le contrôle d'accès aux nombreux éléments APIs présentés par les microservices, les couches Backend for Frontend (BFF) et les autres composants d'une application SaaS mutualisée. Cette approche doit être accompagnée d'un mécanisme capable de prendre des décisions d'accès complexes en fonction de nombreux facteurs et attributs.

Traditionnellement, le contrôle d'accès et l'autorisation des API étaient gérés par une logique personnalisée dans le code de l'application. Cette approche était sujette aux erreurs et n'était pas sécurisée, car les développeurs ayant accès à ce code pouvaient accidentellement ou délibérément modifier la logique d'autorisation, ce qui pouvait entraîner un accès non autorisé. L'audit des décisions prises par la logique personnalisée dans le code de l'application était difficile, car les auditeurs devaient s'immerger dans la logique personnalisée pour déterminer son efficacité en termes de respect d'une norme particulière. De plus, le contrôle d'accès aux API était généralement inutile, car il n'y en avait pas autant APIs à sécuriser. Le changement de paradigme dans la conception des applications pour favoriser les microservices et les architectures orientées services a augmenté le nombre de celles APIs qui doivent utiliser une forme d'autorisation et de contrôle d'accès. En outre, la nécessité de maintenir un accès basé sur les locataires dans une application SaaS à locataires multiples pose des problèmes d'autorisation supplémentaires pour préserver la location. Les meilleures pratiques décrites dans ce guide présentent plusieurs avantages :

- La logique d'autorisation peut être centralisée et écrite dans un langage déclaratif de haut niveau qui n'est spécifique à aucun langage de programmation.
- La logique d'autorisation est extraite du code de l'application et peut être appliquée sous forme de modèle reproductible APIs à tous les éléments d'une application.
- L'abstraction empêche les développeurs de modifier accidentellement la logique d'autorisation.

- L'intégration dans une application SaaS est cohérente et simple.
- L'abstraction évite d'avoir à écrire une logique d'autorisation personnalisée pour chaque point de terminaison d'API.
- Les audits sont simplifiés, car l'auditeur n'a plus besoin de revoir le code pour déterminer les autorisations.
- L'approche décrite dans ce guide soutient l'utilisation de plusieurs paradigmes de contrôle d'accès en fonction des exigences d'une organisation.
- Cette approche d'autorisation et de contrôle d'accès fournit un moyen simple et direct de maintenir l'isolation des données des locataires au niveau de la couche API d'une application SaaS.
- Les meilleures pratiques fournissent une approche cohérente en matière d'intégration et de départ des locataires en ce qui concerne l'autorisation.
- Cette approche propose différents modèles de déploiement d'autorisations (groupés ou silo), qui présentent à la fois des avantages et des inconvénients, comme indiqué dans ce guide.

Résultats commerciaux ciblés

Ce guide prescriptif décrit des modèles de conception reproductibles pour les contrôles d'autorisation et d'accès aux API qui peuvent être mis en œuvre pour les applications SaaS multi-locataires. Ce guide est destiné à toute équipe qui développe des applications soumises à des exigences d'autorisation complexes ou à des besoins stricts en matière de contrôle d'accès aux API. L'architecture détaille la création d'un point de décision politique (PDP) ou d'un moteur de politique et l'intégration des points d'application des politiques (PEP) dans. APIs Deux options spécifiques pour créer un PDP sont abordées : utiliser Amazon Verified Permissions avec le SDK Cedar et utiliser l'Open Policy Agent (OPA) avec le langage de politique Rego. Le guide explique également comment prendre des décisions d'accès sur la base d'un modèle de contrôle d'accès basé sur les attributs (ABAC) ou d'un modèle de contrôle d'accès basé sur les rôles (RBAC), ou d'une combinaison des deux modèles. Nous vous recommandons d'utiliser les modèles et concepts de conception fournis dans ce guide pour informer et standardiser votre mise en œuvre de l'autorisation et du contrôle d'accès aux API dans les applications SaaS multi-locataires. Ces conseils aident à atteindre les résultats commerciaux suivants :

- Architecture d'autorisation d'API standardisée pour les applications SaaS à locataires multiples : cette architecture distingue trois composants : le point d'administration des politiques (PAP) où les politiques sont stockées et gérées, le point de décision politique (PDP) où ces politiques sont évaluées pour parvenir à une décision d'autorisation, et le point d'application des politiques (PEP)

qui applique cette décision. Le service d'autorisation hébergé, Verified Permissions, sert à la fois de PAP et de PDP. Vous pouvez également créer vous-même votre PDP en utilisant un moteur open source tel que Cedar ou OPA.

- Dissociation de la logique d'autorisation des applications — Lorsqu'elle est intégrée au code de l'application ou mise en œuvre par le biais d'un mécanisme d'application ad hoc, la logique d'autorisation peut être sujette à des modifications accidentelles ou malveillantes entraînant un accès involontaire aux données entre locataires ou d'autres violations de sécurité. Pour atténuer ces risques, vous pouvez utiliser un PAP, tel que Verified Permissions, pour stocker les politiques d'autorisation indépendamment du code de l'application et pour appliquer une gouvernance solide à la gestion de ces politiques. Les politiques peuvent être gérées de manière centralisée dans un langage déclaratif de haut niveau, ce qui rend la gestion de la logique d'autorisation bien plus simple que lorsque vous intégrez des politiques dans plusieurs sections du code de l'application. Cette approche garantit également que les mises à jour sont appliquées de manière cohérente.
- Approche flexible des modèles de contrôle d'accès — Le contrôle d'accès basé sur les rôles (RBAC), le contrôle d'accès basé sur les attributs (ABAC) ou une combinaison des deux modèles sont tous des approches valides en matière de contrôle d'accès. Ces modèles tentent de satisfaire aux exigences d'autorisation d'une entreprise en utilisant différentes approches. Ce guide compare et met en contraste ces modèles afin de vous aider à sélectionner un modèle adapté à votre organisation. Le guide explique également comment ces modèles s'appliquent aux différents langages de politique d'autorisation, tels que OPA/Rego Cedar. Les architectures décrites dans ce guide permettent d'adopter avec succès l'un ou les deux modèles.
- Contrôle strict de l'accès aux API : ce guide fournit une méthode de sécurisation APIs cohérente et omniprésente dans une application avec un minimum d'efforts. Cela est particulièrement utile pour les architectures d'applications orientées services ou microservices qui en utilisent généralement un grand nombre APIs pour faciliter les communications intra-applications. Le contrôle strict de l'accès aux API contribue à renforcer la sécurité d'une application et à la rendre moins vulnérable aux attaques ou à l'exploitation.

Isolement des locataires et autorisation multi-locataires

Ce guide fait référence aux concepts d'isolement des locataires et d'autorisation multi-locataires. L'isolation des locataires fait référence aux mécanismes explicites que vous utilisez dans un système SaaS pour garantir que les ressources de chaque locataire, même lorsqu'elles opèrent sur une infrastructure partagée, sont isolées. L'autorisation multi-locataires consiste à autoriser les actions entrantes et à empêcher qu'elles ne soient mises en œuvre sur le mauvais locataire. Un utilisateur

hypothétique pourrait être authentifié et autorisé, tout en continuant d'accéder aux ressources d'un autre locataire. L'authentification et l'autorisation ne bloqueront pas cet accès. Vous devez mettre en œuvre l'isolation des locataires pour atteindre cet objectif. Pour une discussion plus approfondie des différences entre ces deux concepts, consultez la section sur l'isolation des locataires du livre blanc sur les principes fondamentaux de l'architecture SaaS.

Types de contrôle d'accès

Vous pouvez utiliser deux modèles généraux pour implémenter le contrôle d'accès : le contrôle d'accès basé sur les rôles (RBAC) et le contrôle d'accès basé sur les attributs (ABAC). Chaque modèle présente des avantages et des inconvénients, qui sont brièvement décrits dans cette section. Le modèle que vous devez utiliser dépend de votre cas d'utilisation spécifique. L'architecture décrite dans ce guide prend en charge les deux modèles.

RBAC

Le contrôle d'accès basé sur les rôles (RBAC) détermine l'accès aux ressources en fonction d'un rôle qui correspond généralement à la logique métier. Les autorisations sont associées au rôle le cas échéant. Par exemple, un rôle marketing autoriserait un utilisateur à effectuer des activités marketing au sein d'un système restreint. Il s'agit d'un modèle de contrôle d'accès relativement simple à mettre en œuvre car il s'aligne bien sur une logique métier facilement reconnaissable.

Le modèle RBAC est moins efficace lorsque :

- Vous avez des utilisateurs uniques dont les responsabilités englobent plusieurs rôles.
- Vous avez une logique métier complexe qui rend les rôles difficiles à définir.
- Le passage à une taille importante nécessite une administration et un mappage constants des autorisations vers les rôles nouveaux et existants.
- Les autorisations sont basées sur des paramètres dynamiques.

ABAC

Le contrôle d'accès basé sur les attributs (ABAC) détermine l'accès aux ressources en fonction des attributs. Les attributs peuvent être associés à un utilisateur, à une ressource, à un environnement ou même à l'état d'une application. Vos politiques ou règles font référence à des attributs et peuvent utiliser la logique booléenne de base pour déterminer si un utilisateur est autorisé à effectuer une action. Voici un exemple de base d'autorisations :

Dans le système de paiement, tous les utilisateurs du service financier sont autorisés à traiter les paiements sur le point de terminaison de l'API `/payments` pendant les heures ouvrables.

L'appartenance au service financier est un attribut utilisateur qui détermine l'accès à`/payments`. Il existe également un attribut de ressource associé au point de terminaison de l'`/payments`API qui autorise l'accès uniquement pendant les heures ouvrables. Dans ABAC, le fait qu'un utilisateur puisse ou non traiter un paiement est déterminé par une politique qui inclut l'appartenance au service financier en tant qu'attribut utilisateur et l'heure en tant qu'attribut de ressource de`/payments`.

Le modèle ABAC est très flexible pour permettre des décisions d'autorisation dynamiques, contextuelles et granulaires. Cependant, le modèle ABAC est difficile à mettre en œuvre au départ. La définition de règles et de politiques ainsi que l'énumération des attributs pour tous les vecteurs d'accès pertinents nécessitent un investissement initial important pour être mis en œuvre.

Approche hybride RBAC-ABAC

La combinaison du RBAC et de l'ABAC peut offrir certains des avantages des deux modèles. Le RBAC, étant si étroitement aligné sur la logique métier, est plus simple à mettre en œuvre que l'ABAC. Pour fournir une couche de granularité supplémentaire lors de la prise de décisions d'autorisation, vous pouvez combiner ABAC et RBAC. Cette approche hybride détermine l'accès en combinant le rôle d'un utilisateur (et les autorisations qui lui sont attribuées) avec des attributs supplémentaires pour prendre des décisions en matière d'accès. L'utilisation des deux modèles permet de simplifier l'administration et l'attribution des autorisations tout en permettant une flexibilité et une granularité accrues en ce qui concerne les décisions d'autorisation.

Comparaison des modèles de contrôle d'accès

Le tableau suivant compare les trois modèles de contrôle d'accès décrits précédemment. Cette comparaison se veut informative et de haut niveau. L'utilisation d'un modèle d'accès dans une situation spécifique n'est pas nécessairement corrélée aux comparaisons effectuées dans ce tableau.

Facteur	RBAC	ABAC	Hybride
Flexibilité	Medium	Élevé	Élevé
Simplicité	Élevé	Faible	Moyen
Granularité	Faible	Élevé	Moyen
Décisions et règles dynamiques	Non	Oui	Oui

Sensible au contexte	Non	Oui	Un peu
Effort d'implémentation	Faible	Élevé	Moyen

Implémentation d'un PDP

Le point de décision politique (PDP) peut être caractérisé comme un moteur de politiques ou de règles. Ce composant est chargé d'appliquer des politiques ou des règles et de décider si un accès particulier est autorisé. Un PDP peut fonctionner avec des modèles de contrôle d'accès basé sur les rôles (RBAC) et de contrôle d'accès basé sur les attributs (ABAC) ; toutefois, un PDP est requis pour l'ABAC. Un PDP permet de transférer la logique d'autorisation du code de l'application vers un système distinct. Cela peut simplifier le code de l'application. Il fournit également une interface easy-to-use reproductible pour prendre des décisions d'autorisation pour les microservices APIs, les couches Backend for Frontend (BFF) ou tout autre composant d'application.

Les sections suivantes présentent trois méthodes de mise en œuvre d'un PDP. Toutefois, cette liste n'est pas exhaustive.

Méthodes de mise en œuvre du PDP :

- Implémentation d'un PDP à l'aide des autorisations vérifiées par Amazon
- Implémentation d'un PDP à l'aide de l'OPA
- Utilisation d'un moteur de politique personnalisé

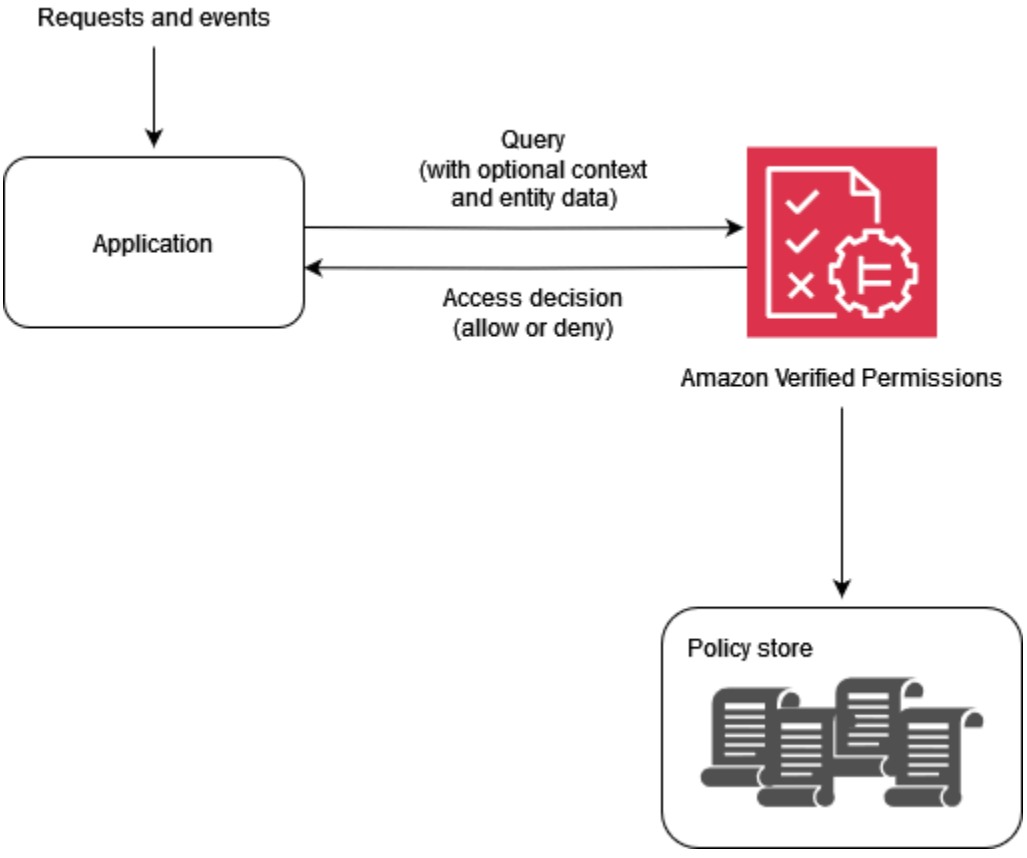
Implémentation d'un PDP à l'aide des autorisations vérifiées par Amazon

Amazon Verified Permissions est un service de gestion des autorisations et d'autorisation évolutif et détaillé que vous pouvez utiliser pour mettre en œuvre un point de décision politique (PDP). En tant que moteur de politiques, il peut aider votre application à vérifier les actions des utilisateurs en temps réel et à mettre en évidence les autorisations trop privilégiées ou non valides. Il aide vos développeurs à créer des applications plus sécurisées plus rapidement en externalisant les autorisations et en centralisant la gestion et l'administration des politiques. En séparant la logique d'autorisation de la logique d'application, Verified Permissions permet le découplage des politiques.

En utilisant les autorisations vérifiées pour implémenter un PDP et en implémentant le moindre privilège et la vérification continue au sein des applications, les développeurs peuvent aligner l'accès à leurs applications sur les principes du Zero Trust. En outre, les équipes de sécurité et d'audit peuvent mieux analyser et auditer qui a accès à quelles ressources au sein d'une application. Verified Permissions utilise Cedar, un langage de politique open source spécialement conçu et axé sur la

sécurité, pour définir des contrôles d'accès basés sur des politiques basés sur le contrôle d'accès basé sur les rôles (RBAC) et le contrôle d'accès basé sur les attributs (ABAC) pour un contrôle d'accès plus granulaire et contextuel.

Les autorisations vérifiées fournissent des fonctionnalités utiles pour les applications SaaS, telles que la possibilité d'activer l'autorisation multi-locataires en utilisant plusieurs fournisseurs d'identité tels qu'Amazon Cognito, Google et Facebook. Une autre fonctionnalité d'autorisations vérifiées particulièrement utile pour les applications SaaS est la prise en charge des rôles personnalisés par locataire. Si vous concevez un système de gestion de la relation client (CRM), un locataire peut définir la granularité de l'accès par opportunités de vente en fonction d'un ensemble de critères particuliers. Un autre locataire peut avoir une autre définition. Les systèmes d'autorisations sous-jacents de Verified Permissions peuvent prendre en charge ces variations, ce qui en fait un excellent candidat pour les cas d'utilisation du SaaS. Les autorisations vérifiées permettent également de rédiger des politiques qui s'appliquent à tous les locataires. Il est donc simple d'appliquer des politiques de protection pour empêcher tout accès non autorisé en tant que fournisseur de SaaS.



Pourquoi utiliser des autorisations vérifiées ?

Utilisez les autorisations vérifiées auprès d'un fournisseur d'identité tel qu'Amazon Cognito pour bénéficier d'une solution de gestion des accès plus dynamique et basée sur des règles pour vos applications. Vous pouvez créer des applications qui aident les utilisateurs à partager des informations et à collaborer tout en préservant la sécurité, la confidentialité et la confidentialité de leurs données. Les autorisations vérifiées contribuent à réduire les coûts opérationnels en vous fournissant un système d'autorisation précis pour appliquer l'accès en fonction des rôles et des attributs de vos identités et de vos ressources. Vous pouvez définir votre modèle de politique, créer et stocker des politiques dans un emplacement central et évaluer les demandes d'accès en quelques millisecondes.

Dans Verified Permissions, vous pouvez exprimer des autorisations à l'aide d'un langage déclaratif simple et lisible par l'homme appelé Cedar. Les politiques rédigées dans Cedar peuvent être partagées entre les équipes quel que soit le langage de programmation utilisé par l'application de chaque équipe.

Points à prendre en compte lorsque vous utilisez des autorisations vérifiées

Dans Autorisations vérifiées, vous pouvez créer des politiques et les automatiser dans le cadre du provisionnement. Vous pouvez également créer des politiques au moment de l'exécution dans le cadre de la logique de l'application. La meilleure pratique consiste à utiliser un pipeline d'intégration et de déploiement continus (CI/CD) pour administrer, modifier et suivre les versions des politiques lorsque vous créez des politiques dans le cadre de l'intégration et du provisionnement des locataires. Une application peut également administrer, modifier et suivre les versions des politiques ; toutefois, la logique de l'application n'exécute pas intrinsèquement cette fonctionnalité. Pour prendre en charge ces fonctionnalités dans votre application, vous devez explicitement concevoir votre application pour implémenter cette fonctionnalité.

S'il est nécessaire de fournir des données externes provenant d'autres sources pour prendre une décision d'autorisation, ces données doivent être récupérées et fournies à Verified Permissions dans le cadre de la demande d'autorisation. Le contexte, les entités et les attributs supplémentaires ne sont pas récupérés par défaut avec ce service.

Aperçu du cèdre

Cedar est un langage de contrôle d'accès flexible, extensible et évolutif basé sur des politiques qui aide les développeurs à exprimer les autorisations des applications sous forme de politiques. Les administrateurs et les développeurs peuvent définir des politiques qui autorisent ou interdisent aux utilisateurs d'agir sur les ressources de l'application. Plusieurs politiques peuvent être associées à

une seule ressource. Lorsqu'un utilisateur de votre application essaie d'effectuer une action sur une ressource, votre application demande l'autorisation du moteur de politiques Cedar. Cedar évalue les politiques applicables et renvoie une `ALLOW` `DENY` décision. Cedar prend en charge les règles d'autorisation pour tout type de principal et de ressource, permet le contrôle d'accès basé sur les rôles (RBAC) et le contrôle d'accès basé sur les attributs (ABAC), et soutient l'analyse grâce à des outils de raisonnement automatisés.

Cedar vous permet de séparer votre logique métier de la logique d'autorisation. Lorsque vous faites des demandes à partir du code de votre application, vous appelez le moteur d'autorisation de Cedar pour déterminer si la demande est autorisée. Si elle est autorisée (la décision l'est`ALLOW`), votre application peut effectuer l'opération demandée. Si elle n'est pas autorisée (la décision l'est`DENY`), votre application peut renvoyer un message d'erreur. Les principales caractéristiques du cèdre incluent :

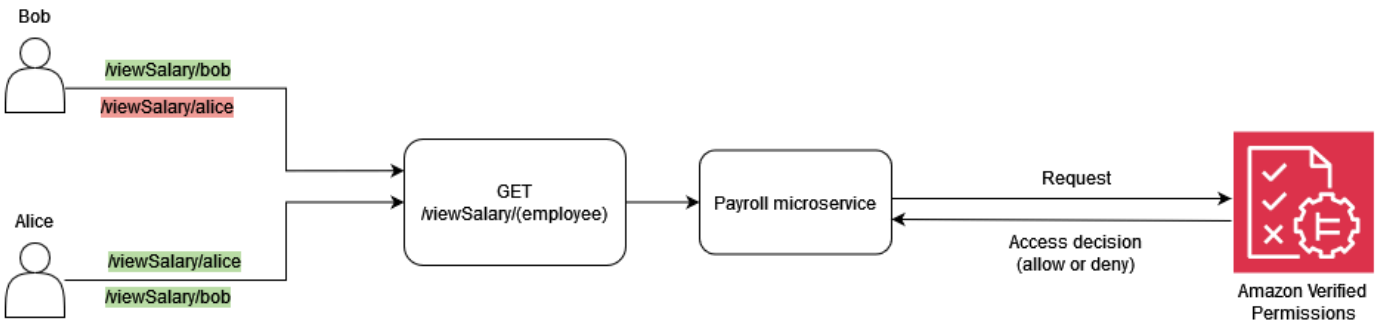
- Expressivité — Cedar est spécialement conçu pour prendre en charge les cas d'utilisation des autorisations et a été développé dans un souci de lisibilité humaine.
- Performance — Cedar prend en charge les politiques d'indexation pour une extraction rapide et fournit une évaluation en temps réel rapide et évolutive avec une latence limitée.
- Analyse — Cedar propose des outils d'analyse qui peuvent optimiser vos politiques et vérifier votre modèle de sécurité.

Pour plus d'informations, consultez le site Web de Cedar.

Exemple 1 : ABAC de base avec autorisations vérifiées et Cedar

Dans cet exemple de scénario, Amazon Verified Permissions est utilisé pour déterminer quels utilisateurs sont autorisés à accéder aux informations d'un microservice de paie fictif. Cette section inclut des extraits de code Cedar pour montrer comment utiliser Cedar pour prendre des décisions en matière de contrôle d'accès. Ces exemples ne sont pas destinés à fournir une exploration complète des fonctionnalités fournies par Cedar et Verified Permissions. Pour une présentation plus complète de Cedar, consultez la documentation de Cedar.

Dans le schéma suivant, nous aimerions appliquer deux règles commerciales générales associées à la `viewSalary` `GET` méthode : les employés peuvent consulter leur propre salaire et les employés peuvent consulter le salaire de toute personne relevant d'eux. Vous pouvez appliquer ces règles commerciales en utilisant les politiques d'autorisations vérifiées.



Les employés peuvent consulter leur propre salaire.

Dans Cedar, la construction de base est une entité qui représente un principe, une action ou une ressource. Pour effectuer une demande d'autorisation et démarrer une évaluation avec une politique d'autorisations vérifiées, vous devez fournir un principal, une action, une ressource et une liste d'entités.

- Le principal (`principal`) est l'utilisateur ou le rôle connecté.
- L'action (`action`) est l'opération évaluée par la demande.
- La ressource (`resource`) est le composant auquel l'action accède.
- La liste des entités (`entityList`) contient toutes les entités requises pour évaluer la demande.

Pour satisfaire à la règle commerciale, les employés peuvent consulter leur propre salaire, vous pouvez fournir une politique d'autorisations vérifiées telle que la suivante.

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

Cette politique évalue `ALLOW` si le propriétaire de l'attribut `Action` est égal au principal `viewSalary` et si la ressource contenue dans la demande est le même. Par exemple, si Bob est l'utilisateur connecté qui a demandé le rapport salarial et qu'il est également le propriétaire du rapport salarial, la politique est évaluée à`ALLOW`.

La demande d'autorisation suivante est soumise à Verified Permissions pour être évaluée par l'exemple de politique. Dans cet exemple, Bob est l'utilisateur connecté qui fait la `viewSalary` demande. Bob est donc le principal du type d'entité`Employee`. L'action que Bob essaie d'effectuer est `viewSalary,` et la ressource qui s'`viewSalary`affichera est `Salary-Bob` de type`Salary`. Afin d'évaluer si Bob peut visualiser la `Salary-Bob` ressource, vous devez fournir une structure d'entité qui lie le type `Employee` avec une valeur de Bob (le principal) à l'attribut propriétaire de la ressource qui possède le type`Salary`. Vous fournissez cette structure dans un`entityList`, où les attributs associés à `Salary` incluent un propriétaire, qui spécifie et `entityIdentifier` qui contient le type `Employee` et la valeurBob. Verified Permissions compare les informations `principal` fournies dans la demande d'autorisation à `owner` l'attribut associé à la `Salary` ressource pour prendre une décision.

```
{
  "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 demande d'autorisation adressée à Verified Permissions renvoie le résultat suivant, où l'attribut `decision` est `ALLOW` ou`DENY`.

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

Dans ce cas, comme Bob essayait de consulter son propre salaire, la demande d'autorisation envoyée à Verified Permissions est évaluée à`ALLOW`. Cependant, notre objectif était d'utiliser les autorisations vérifiées pour appliquer deux règles commerciales. La règle métier qui stipule ce qui suit doit également être vraie :

Les employés peuvent consulter le salaire de toute personne relevant d'eux.

Pour satisfaire à cette règle commerciale, vous pouvez définir une autre politique. La politique suivante détermine `ALLOW` si l'action est effectuée `viewSalary` et si la ressource contenue dans la demande possède un attribut `owner.manager` égal au principal. Par exemple, si Alice est l'utilisateur connecté qui a demandé le rapport salarial et qu'Alice est la responsable du propriétaire du rapport, la politique est évaluée à`ALLOW`.

```
permit (
    principal,
```

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

La demande d'autorisation suivante est soumise à Verified Permissions pour être évaluée par l'exemple de politique. Dans cet exemple, Alice est l'utilisateur connecté qui fait la `viewSalary` demande. Alice est donc la principale et l'entité est du type`Employee`. L'action qu'Alice essaie d'effectuer est`viewSalary`, et la ressource qui s'`viewSalary`affichera est du type `Salary` avec une valeur de`Salary-Bob`. Pour évaluer si Alice peut consulter la `Salary-Bob` ressource, vous devez fournir une structure d'entité qui lie le type `Employee` avec une valeur de `Alice` à l'`manager`attribut, qui doit ensuite être associé à l'`owner`attribut du type `Salary` avec une valeur de`Salary-Bob`. Vous fournissez cette structure dans un`entityList`, où les attributs associés à `Salary` incluent un propriétaire, qui spécifie et `entityIdentifier` qui contient le type `Employee` et la valeur`Bob`. Les autorisations vérifiées vérifient d'abord l'`owner`attribut, qui est évalué en fonction du type `Employee` et de la valeur`Bob`. Ensuite, Verified Permissions évalue l'`manager`attribut associé `Employee` et le compare au principal fourni pour prendre une décision d'autorisation. Dans ce cas, la décision est prise `ALLOW` parce que les `resource.owner.manager` attributs `principal` et sont équivalents.

```
{
  "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": []
      }
```

```
    ]
  }
}
```

Jusqu'à présent, dans cet exemple, nous avons fourni les deux règles commerciales associées à la `viewSalary` méthode : les employés peuvent consulter leur propre salaire et les employés peuvent consulter le salaire de toute personne relevant d'eux, aux autorisations vérifiées sous forme de politiques visant à satisfaire aux conditions de chaque règle commerciale de manière indépendante. Vous pouvez également utiliser une seule politique d'autorisations vérifiées pour satisfaire aux conditions des deux règles commerciales :

Les employés peuvent consulter leur propre salaire et celui de toute personne relevant d'eux.

Lorsque vous utilisez la demande d'autorisation précédente, la politique suivante `ALLOW` détermine si l'action est `viewSalary` et si la ressource contenue dans la demande possède un attribut `owner.manager` égal à ou un attribut `owner` égal à. `principal` `principal`

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

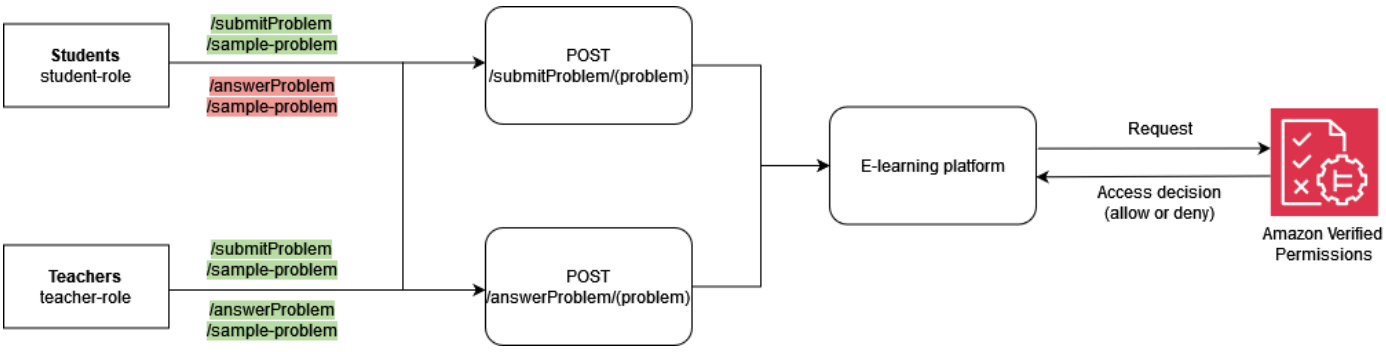
Par exemple, si Alice est l'utilisateur connecté qui demande le rapport salarial, et si Alice est soit la responsable du propriétaire, soit la propriétaire du rapport, alors la politique est évaluée à`ALLOW`.

Pour plus d'informations sur l'utilisation d'opérateurs logiques avec les politiques Cedar, consultez la documentation de Cedar.

Exemple 2 : RBAC de base avec autorisations vérifiées et Cedar

Cet exemple utilise Verified Permissions et Cedar pour illustrer le RBAC de base. Comme mentionné précédemment, la structure de base de Cedar est une entité. Les développeurs définissent leurs propres entités et peuvent éventuellement créer des relations entre les entités. L'exemple suivant inclut trois types d'entités : `UsersRoles`, et`Problems`. `Studentset` `Teachers` peuvent être

considérées comme des entités du type `Role`, et chacune `User` peut être associée à zéro ou à l'un des`Roles`.



Dans Cedar, ces relations sont exprimées en liant le `Role Student` au `User Bob` en tant que parent. Cette association regroupe logiquement tous les utilisateurs étudiants dans un seul groupe. Pour plus d'informations sur le regroupement dans Cedar, consultez la documentation de Cedar.

La politique suivante évalue la décision `ALLOW` d'action `submitProblem`, pour tous les principaux liés au groupe logique `Students` de ce type. `Role`

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

La politique suivante détermine la décision prise `ALLOW` pour l'action `submitProblem` ou `answerProblem` pour tous les principes liés au groupe logique `Teachers` du type. `Role`

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

Afin d'évaluer les demandes avec ces politiques, le moteur d'évaluation doit savoir si le principal référencé dans la demande d'autorisation est membre du groupe approprié. Par conséquent, l'application doit transmettre les informations pertinentes relatives à l'appartenance au groupe

au moteur d'évaluation dans le cadre de la demande d'autorisation. Cela se fait par le biais de la `entities` propriété, qui vous permet de fournir au moteur d'évaluation Cedar des données d'attribut et d'appartenance à un groupe pour le principal et la ressource impliqués dans l'appel d'autorisation. Dans le code suivant, l'appartenance à un groupe est indiquée par le fait `User::"Bob"` qu'un parent est appelé`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": []
```

```
        }
      ]
  }
}
```

Dans cet exemple, Bob est l'utilisateur connecté qui fait la `answerProblem` demande. Par conséquent, Bob est le principal et l'entité est du type`User`. L'action que Bob essaie d'effectuer est`answerProblem`. Afin d'évaluer si Bob peut effectuer l'`answerProblem`action, vous devez fournir une structure d'entité qui lie l'entité `User` à une valeur de Bob et lui attribue son appartenance au groupe en répertoriant une entité parent sous `Role::"Students"` le nom. Étant donné que les entités du groupe d'utilisateurs `Role::"Students"` sont uniquement autorisées à effectuer l'action`submitProblem`, cette demande d'autorisation est évaluée à`DENY`.

En revanche, si le type `User` qui a une valeur égale à `Alice` et qui fait partie du groupe `Role::"Teachers"` essaie d'exécuter l'`answerProblem`action, la demande d'autorisation est évaluée comme suit`ALLOW`, car la politique stipule que les principaux membres du groupe `Role::"Teachers"` sont autorisés à effectuer l'action `answerProblem` sur toutes les ressources. Le code suivant montre ce type de demande d'autorisation qui est évalué à`ALLOW`.

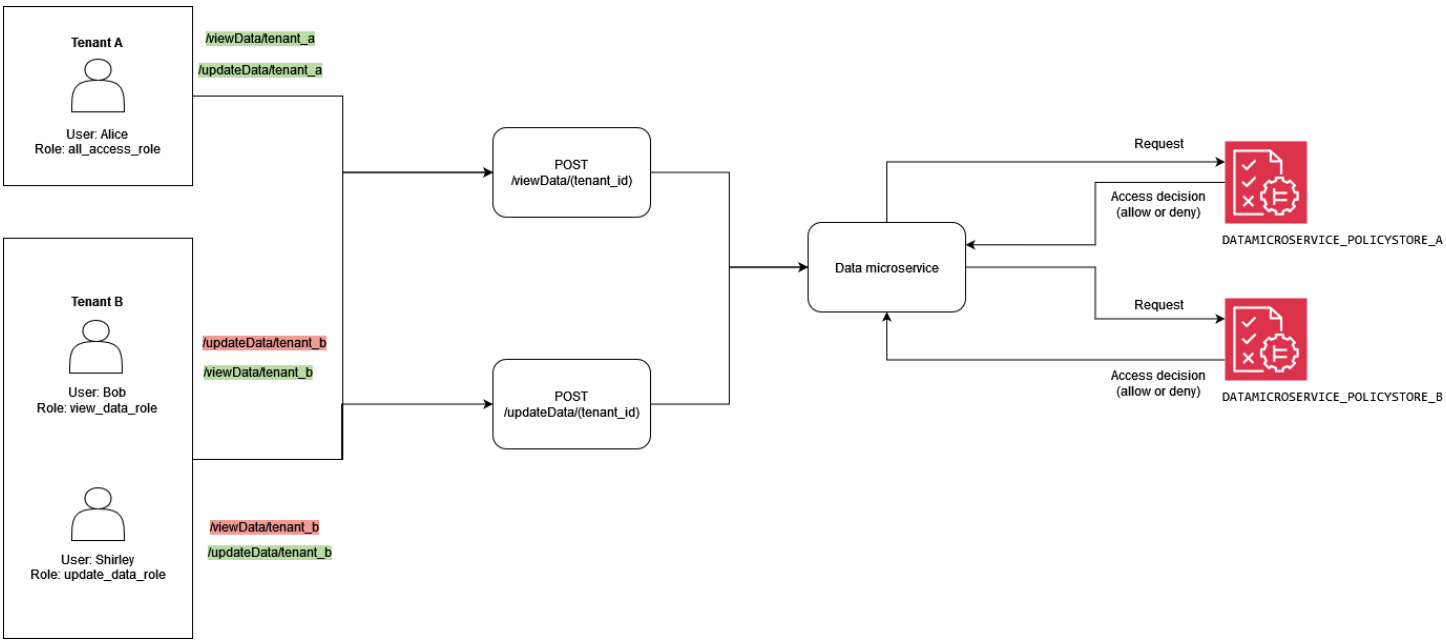
```
{
  "policyStoreId": "ELEARNING_POLICYSTOREID",
  "principal": {
    "entityType": "ElearningApp::User",
    "entityId": "Alice"
  },
  "action": {
    "actionType": "ElearningApp::Action",
    "actionId": "answerProblem"
  },
  "resource": {
    "entityType": "ElearningApp::Problem",
    "entityId":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

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

Exemple 3 : contrôle d'accès multi-locataires avec RBAC

Pour développer l'exemple RBAC précédent, vous pouvez étendre vos exigences pour inclure la mutualisation du SaaS, qui est une exigence courante pour les fournisseurs de SaaS. Dans les solutions multi-locataires, l'accès aux ressources est toujours fourni au nom d'un locataire donné. En d'autres termes, les utilisateurs du locataire A ne peuvent pas voir les données du locataire B, même si ces données sont logiquement ou physiquement colocalisées dans un système. L'exemple suivant montre comment vous pouvez implémenter l'isolation des locataires en utilisant plusieurs magasins de politiques d'autorisations vérifiées, et comment vous pouvez utiliser des rôles d'utilisateur pour définir des autorisations au sein du locataire.

L'utilisation du modèle de conception Per Tenant Policy Store est une bonne pratique pour maintenir l'isolement des locataires tout en mettant en œuvre un contrôle d'accès avec des autorisations vérifiées. Dans ce scénario, les demandes des utilisateurs du locataire A et du locataire B sont vérifiées par rapport à des magasins de politiques distincts, `DATAMICROSERVICE_POLICYSTORE_A` et`DATAMICROSERVICE_POLICYSTORE_B`, respectivement. Pour plus d'informations sur les considérations relatives à la conception des autorisations vérifiées pour les applications SaaS multi-locataires, consultez la section Considérations relatives à la conception multi-locataires des autorisations vérifiées.



La politique suivante se trouve dans le magasin `DATAMICROSERVICE_POLICYSTORE_A` de politiques. Il vérifie que le principal fera partie du groupe `allAccessRole` de type`Role`. Dans ce cas, le principal sera autorisé à effectuer les `updateData` actions `viewData` et sur toutes les ressources associées au locataire A.

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

Les politiques suivantes se trouvent dans le magasin `DATAMICROSERVICE_POLICYSTORE_B` de politiques. La première politique vérifie que le principal fait partie du `updateDataRole` groupe de types`Role`. En supposant que tel soit le cas, il autorise les principaux à effectuer l'`updateData`action sur les ressources associées au locataire B.

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

Cette deuxième politique stipule que les principaux qui font partie du `viewDataRole` groupe de types `Role` doivent être autorisés à effectuer l'`viewData`action sur les ressources associées au locataire B.

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

La demande d'autorisation faite par le locataire A doit être envoyée au magasin de `DATAMICROSERVICE_POLICYSTORE_A` politiques et vérifiée par les politiques appartenant à ce magasin. Dans ce cas, cela est vérifié par la première politique décrite précédemment dans le cadre de cet exemple. Dans cette demande d'autorisation, le principal de type `User` avec une valeur `Alice` de demande d'effectuer l'`viewData`action. Le principal appartient au groupe `allAccessRole` de types`Role`. Alice essaie d'exécuter l'`viewData`action sur la `SampleData` ressource. Alice ayant un `allAccessRole` rôle à jouer, cette évaluation aboutit à une `ALLOW` décision.

```
{
  "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, au contraire, vous consultez une demande faite par le locataire BUser Bob, vous verrez quelque chose comme la demande d'autorisation suivante. La demande est envoyée au magasin de DATAMICROSERVICE_POLICYSTORE_B politiques car elle provient du locataire B. Dans cette demande, le principal Bob souhaite effectuer l'action updateData sur la ressourceSampleData. Cependant, ne Bob fait pas partie d'un groupe ayant accès à l'action updateData sur cette ressource. La demande aboutit donc à une DENY décision.

```
{
  "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": []
      }
    ]
  }
}
```

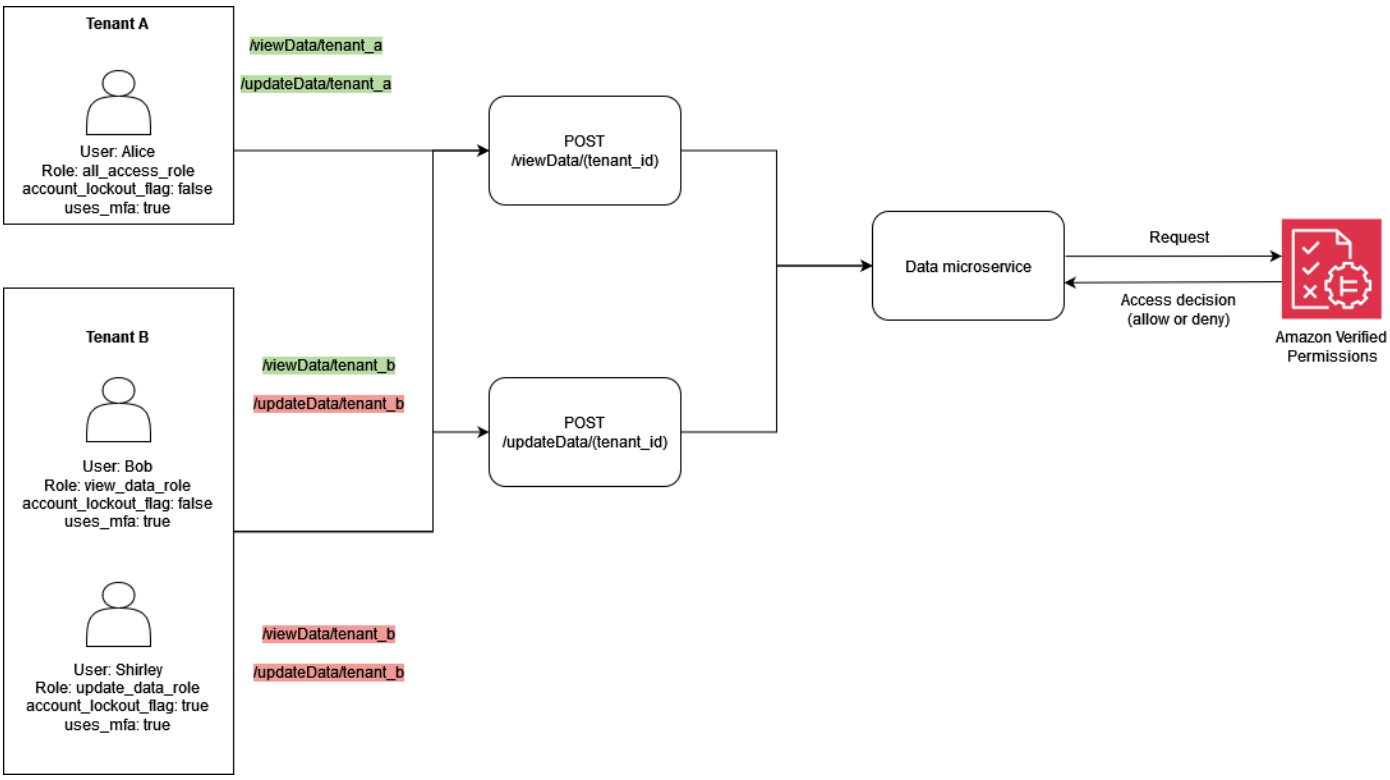
Dans ce troisième exemple, `User Alice` essaie d'exécuter l'`viewData`action sur la ressource`SampleData`. Cette demande est dirigée vers le magasin de `DATAMICROSERVICE_POLICYSTORE_A` politiques car le principal `Alice` appartient à la locataire A. `Alice` fait partie du groupe `allAccessRole` de ce type`Role`, ce qui lui permet d'effectuer l'`viewData`action sur les ressources. En tant que telle, la demande aboutit à une `ALLOW` décision.

```
{
  "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": []
      }
    ]
  }
}
```

Exemple 4 : Contrôle d'accès multi-locataires avec RBAC et ABAC

Pour améliorer l'exemple RBAC présenté dans la section précédente, vous pouvez ajouter des attributs aux utilisateurs afin de créer une approche hybride RBAC-ABAC pour le contrôle d'accès multi-locataires. Cet exemple inclut les mêmes rôles que dans l'exemple précédent, mais ajoute l'attribut user `account_lockout_flag` et le paramètre de contexte`uses_mfa`. L'exemple adopte également une approche différente pour mettre en œuvre le contrôle d'accès multi-locataires en utilisant à la fois le RBAC et l'ABAC, et utilise un magasin de politiques partagé au lieu d'un magasin de politiques différent pour chaque locataire.



Cet exemple représente une solution SaaS multi-locataires dans laquelle vous devez fournir des décisions d'autorisation pour le locataire A et le locataire B, comme dans l'exemple précédent.

Pour implémenter la fonctionnalité de verrouillage utilisateur, l'exemple ajoute l'attribut `account_lockout_flag` au principal de l'`User`entité dans la demande d'autorisation. Cet indicateur verrouille l'accès des utilisateurs au système et DENY octroie tous les privilèges à l'utilisateur bloqué. L'`account_lockout_flag`attribut est associé à l'`User`entité et est actif `User` jusqu'à ce que le drapeau soit activement révoqué au cours de plusieurs sessions. L'exemple utilise la `when` condition pour évalue`account_lockout_flag`.

L'exemple ajoute également des détails sur la demande et la session. Les informations contextuelles indiquent que la session a été authentifiée à l'aide de l'authentification multifactorielle. Pour implémenter cette validation, l'exemple utilise la `when` condition pour évaluer l'`uses_mfa`indicateur dans le cadre du champ de contexte. Pour plus d'informations sur les meilleures pratiques en matière d'ajout de contexte, consultez la documentation 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
};
```

Cette politique empêche l'accès aux ressources à moins que celles-ci ne soient dans le même groupe que l'`Tenant`attribut du principal demandeur. Cette approche visant à maintenir l'isolement des locataires est appelée approche One Shared Multi-Tenant Policy Store. Pour plus d'informations sur les considérations relatives à la conception des autorisations vérifiées pour les applications SaaS multi-locataires, consultez la section Considérations relatives à la conception multi-locataires des autorisations vérifiées.

La politique garantit également que le directeur est membre de `allAccessRole` et limite les actions à `viewData` et`updateData`. En outre, cette politique vérifie que `account_lockout_flag` c'est le cas `false` et que la valeur de contexte de est `uses_mfa` évaluée à. `true`

De même, la politique suivante garantit que le principal et la ressource sont associés au même locataire, ce qui empêche l'accès entre locataires. Cette politique garantit également que le directeur est membre `viewDataRole` et limite les actions à`viewData`. En outre, il vérifie que la valeur de contexte `account_lockout_flag` est `false` et que la valeur de contexte de est `uses_mfa` évaluée à. `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 troisième politique est similaire à la précédente. La politique exige que la ressource soit membre du groupe correspondant à l'entité représentée par`principal.Tenant`. Cela garantit que le principal et la ressource sont associés au locataire B, ce qui empêche l'accès entre locataires. Cette

politique garantit que le directeur est membre `updateDataRole` et limite les actions à`updateData`. En outre, cette politique vérifie que la valeur de contexte `account_lockout_flag` est `false` et que la valeur de contexte de est `uses_mfa` évaluée à. `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 demande d'autorisation suivante est évaluée selon les trois politiques décrites plus haut dans cette section. Dans cette demande d'autorisation, le principal de type `User` et avec une valeur de `Alice` fait une `updateData` demande avec le rôle`allAccessRole`. `Alice`possède l'attribut `Tenant` dont la valeur est`Tenant::"TenantA"`. L'action `Alice` que vous essayez d'exécuter est `updateData,` et la ressource à laquelle elle sera appliquée est `SampleData` du type`Data`. `SampleData`a `TenantA` comme entité mère.

Selon la première politique du magasin de <DATAMICROSERVICE_POLICYSTOREID> politiques, `Alice` vous pouvez effectuer l'`updateData`action sur la ressource, en supposant que les conditions de la `when` clause de la politique sont remplies. La première condition exige que l'`principal.Tenant`attribut soit évalué à`TenantA`. La deuxième condition exige que l'attribut `account_lockout_flag` du principal soit`false`. La dernière condition exige que le contexte `uses_mfa` soit`true`. Les trois conditions étant remplies, la demande renvoie une `ALLOW` décision.

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

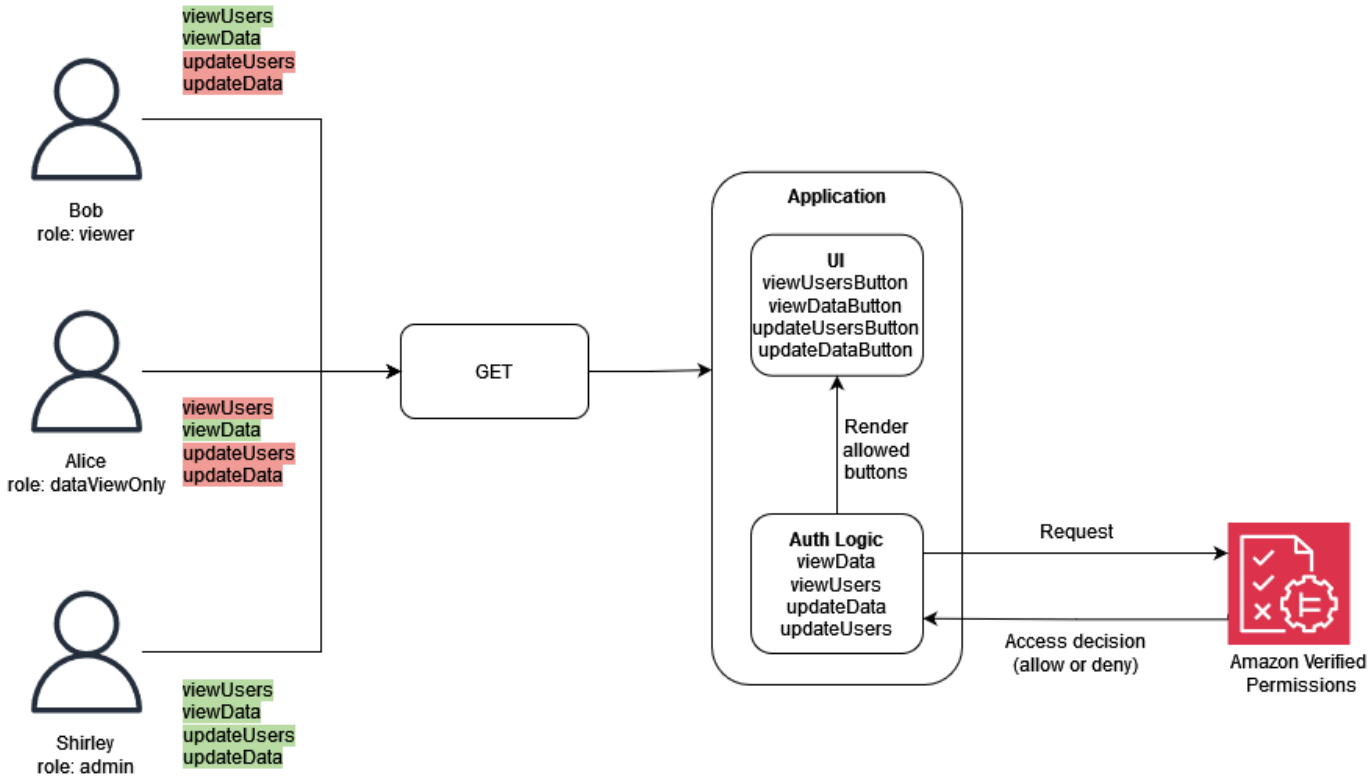
```
      "entityType": "MultitenantApp::Data",
      "entityId": "SampleData"
  },
  "context": {
    "contextMap": {
        "uses_mfa": {
            "boolean": true
        }
    }
  },
  "entities": {
    "entityList": [
      {
        "identifier": {
            "entityType": "MultitenantApp::User",
            "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"
            }
        ]
      }
    ]
  }
}
```

Exemple 5 : filtrage de l'interface utilisateur avec des autorisations vérifiées et Cedar

Vous pouvez également utiliser les autorisations vérifiées pour implémenter le filtrage RBAC des éléments de l'interface utilisateur en fonction des actions autorisées. Cela est extrêmement utile pour les applications qui comportent des éléments d'interface utilisateur contextuels susceptibles d'être associés à des utilisateurs ou à des locataires spécifiques dans le cas d'une application SaaS à locataires multiples.

Dans l'exemple suivant, `Users` des ne `Role` `viewer` sont pas autorisés à effectuer des mises à jour. Pour ces utilisateurs, l'interface utilisateur ne doit afficher aucun bouton de mise à jour.



Dans cet exemple, une application Web d'une seule page comporte quatre boutons. Les boutons visibles dépendent `Role` de l'utilisateur actuellement connecté à l'application. Lorsque l'application Web d'une seule page affiche l'interface utilisateur, elle interroge les autorisations vérifiées pour déterminer les actions que l'utilisateur est autorisé à effectuer, puis génère les boutons en fonction de la décision d'autorisation.

La politique suivante indique que le type dont la valeur est égale `Role` à `viewer` peut afficher à la fois les utilisateurs et les données. Une décision `ALLOW` d'autorisation pour cette politique nécessite une `viewUsers` action `viewData` ou, et nécessite également qu'une ressource soit associée au type `Data` ou`Users`. Une `ALLOW` décision autorise l'interface utilisateur à afficher deux boutons : `viewDataButton` et`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 politique suivante indique que le type `Role` dont la valeur est égale à ne `viewerDataOnly` peut afficher que les données. Une décision `ALLOW` d'autorisation pour cette politique nécessite une `viewData` action et nécessite également qu'une ressource soit associée au type`Data`. Une `ALLOW` décision autorise l'interface utilisateur à afficher le bouton`viewDataButton`.

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

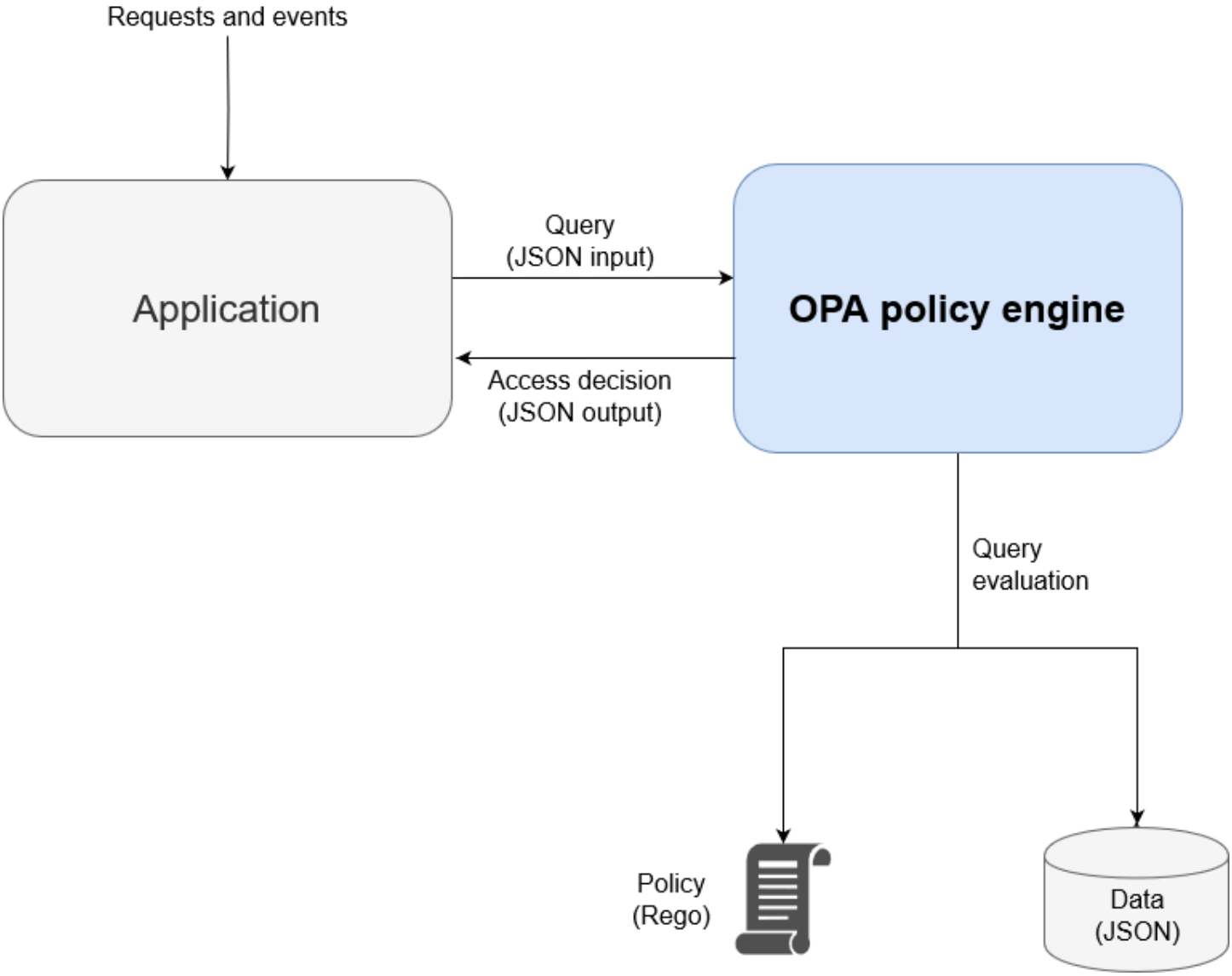
La politique suivante indique que le type `Role` dont la valeur est égale à `admin` peut modifier et afficher les données et les utilisateurs. Une décision `ALLOW` d'autorisation pour cette politique nécessite une action de `updateData`ou`updateUsers`, `viewData`, ou`viewUsers`, et nécessite également qu'une ressource soit associée au type `Data` ou`Users`. Une `ALLOW` décision autorise l'interface utilisateur à afficher les quatre boutons : `updateDataButton`ou`updateUsersButton`,`viewDataButton`, et`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"]
};
```

Implémentation d'un PDP à l'aide de l'OPA

L'Open Policy Agent (OPA) est un moteur de politique open source à usage général. L'OPA a de nombreux cas d'utilisation, mais le cas d'utilisation pertinent pour la mise en œuvre du PDP est sa capacité à dissocier la logique d'autorisation d'une application. C'est ce que l'on appelle le découplage des politiques. L'OPA est utile pour mettre en œuvre un PDP pour plusieurs raisons. Il utilise un langage déclaratif de haut niveau appelé Rego pour rédiger des politiques et des règles.

Ces politiques et règles existent séparément d'une application et peuvent rendre des décisions d'autorisation sans aucune logique spécifique à l'application. L'OPA expose également une RESTful API pour rendre la récupération des décisions d'autorisation simple et directe. Pour prendre une décision d'autorisation, une application interroge OPA avec une entrée JSON, et OPA évalue l'entrée par rapport aux politiques spécifiées pour renvoyer une décision d'accès en JSON. L'OPA est également capable d'importer des données externes susceptibles d'être pertinentes pour prendre une décision d'autorisation.



L'OPA présente plusieurs avantages par rapport aux moteurs de politiques personnalisés :

- L'OPA et son évaluation des politiques avec Rego fournissent un moteur de politiques flexible et prédéfini qui ne nécessite que l'insertion de politiques et de toutes les données nécessaires pour

prendre des décisions d'autorisation. Cette logique d'évaluation des politiques devrait être recréée dans une solution de moteur de politiques personnalisée.

- L'OPA simplifie la logique d'autorisation en écrivant les politiques dans un langage déclaratif. Vous pouvez modifier et administrer ces politiques et règles indépendamment de tout code d'application, sans compétences en développement d'applications.
- OPA expose une RESTful API qui simplifie l'intégration avec les points d'application des politiques (PEPs).
- OPA fournit un support intégré pour la validation et le décodage des jetons Web JSON ()JWTs.
- L'OPA est une norme d'autorisation reconnue, ce qui signifie que la documentation et les exemples sont nombreux si vous avez besoin d'assistance ou de recherches pour résoudre un problème particulier.
- L'adoption d'une norme d'autorisation telle que l'OPA permet de partager les politiques écrites dans Rego entre les équipes, quel que soit le langage de programmation utilisé par l'application d'une équipe.

Il y a deux choses que l'OPA ne fournit pas automatiquement :

- L'OPA ne dispose pas d'un plan de contrôle robuste pour la mise à jour et la gestion des politiques. L'OPA fournit certains modèles de base pour la mise en œuvre des mises à jour des politiques, la surveillance et l'agrégation des journaux en exposant une API de gestion, mais l'intégration avec cette API doit être gérée par l'utilisateur OPA. La meilleure pratique consiste à utiliser un pipeline d'intégration et de déploiement continus (CI/CD) pour administrer, modifier et suivre les versions des politiques et gérer les politiques dans OPA.
- OPA ne peut pas récupérer de données provenant de sources externes par défaut. Une source de données externe pour une décision d'autorisation peut être une base de données contenant les attributs des utilisateurs. Il existe une certaine flexibilité dans la manière dont les données externes sont fournies à l'OPA : elles peuvent être mises en cache localement à l'avance ou récupérées dynamiquement à partir d'une API lorsqu'une décision d'autorisation est demandée, mais l'OPA ne peut pas obtenir ces informations en votre nom.

Présentation de Rego

Rego est un langage de politique à usage général, ce qui signifie qu'il fonctionne pour n'importe quelle couche de la pile et n'importe quel domaine. L'objectif principal de Rego est d'accepter les entrées et les données JSON/YAML qui sont évaluées pour prendre des décisions basées sur

des politiques concernant les ressources, les identités et les opérations de l'infrastructure. Rego vous permet de rédiger une politique pour n'importe quelle couche d'une pile ou d'un domaine sans nécessiter de modification ou d'extension de la langue. Voici quelques exemples de décisions que Rego peut prendre :

- Cette demande d'API est-elle autorisée ou refusée ?
- Quel est le nom d'hôte du serveur de sauvegarde pour cette application ?
- Quel est le score de risque associé à ce changement d'infrastructure proposé ?
- Sur quels clusters ce conteneur doit-il être déployé pour garantir une haute disponibilité ?
- Quelles informations de routage doivent être utilisées pour ce microservice ?

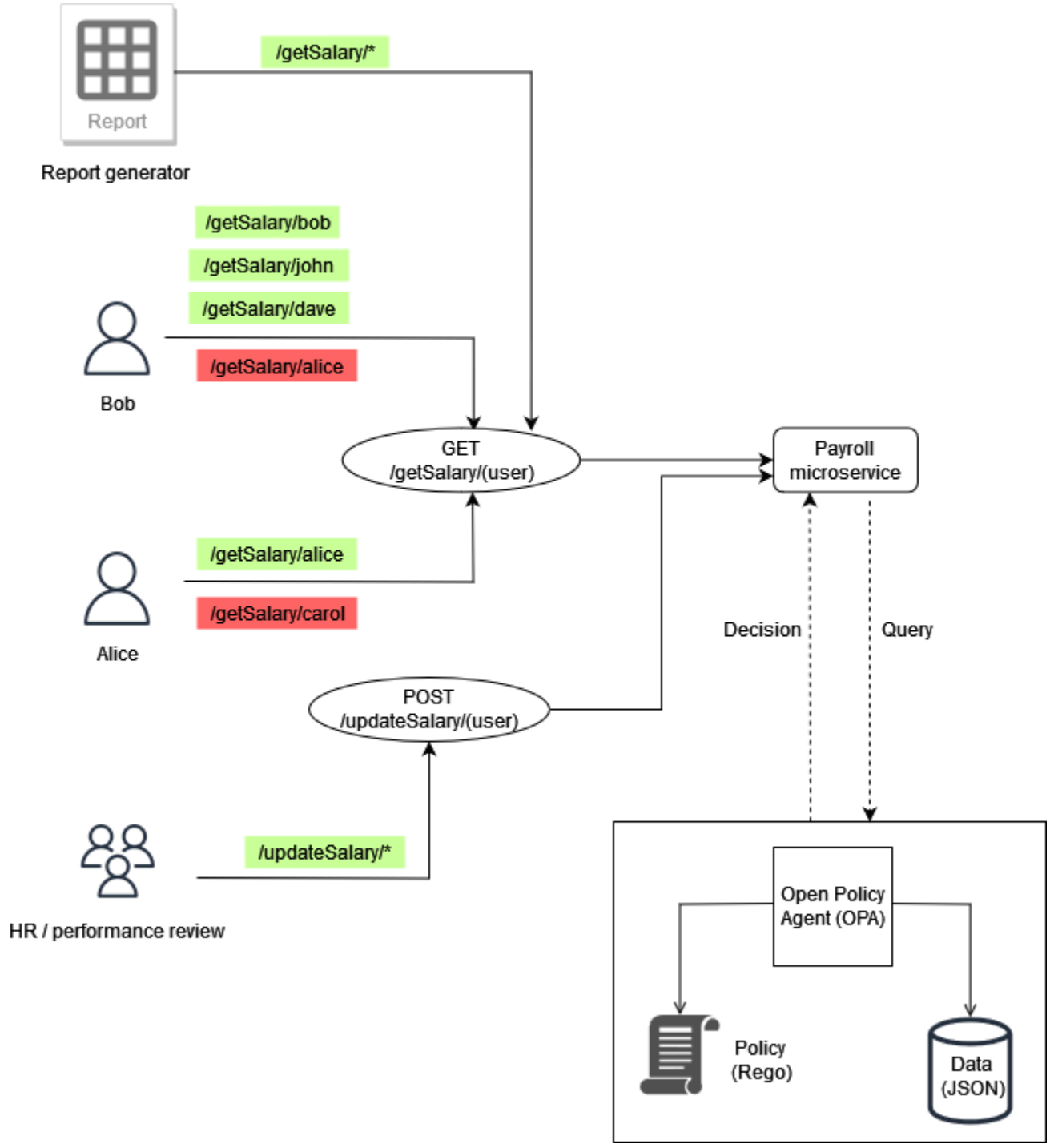
Pour répondre à ces questions, Rego utilise une philosophie de base sur la manière dont ces décisions peuvent être prises. Les deux principes clés de la rédaction d'une politique dans Rego sont les suivants :

- Chaque ressource, identité ou opération peut être représentée sous forme de données JSON ou YAML.
- La politique est une logique appliquée aux données.

Rego aide les systèmes logiciels à prendre des décisions d'autorisation en définissant la logique d'évaluation des entrées de JSON/YAML données. Les langages de programmation tels que C, Java, Go et Python sont la solution habituelle à ce problème, mais Rego a été conçu pour se concentrer sur les données et les entrées qui représentent votre système, ainsi que sur la logique de prise de décisions politiques à partir de ces informations.

Exemple 1 : ABAC de base avec OPA et Rego

Cette section décrit un scénario dans lequel l'OPA est utilisée pour prendre des décisions d'accès concernant les utilisateurs autorisés à accéder aux informations dans un microservice de paie fictif. Des extraits de code Rego sont fournis pour montrer comment vous pouvez utiliser Rego pour prendre des décisions de contrôle d'accès. Ces exemples ne sont ni exhaustifs ni une exploration complète des capacités de Rego et d'OPA. Pour un aperçu plus complet de Rego, nous vous recommandons de consulter la documentation Rego sur le site Web de l'OPA.



Exemple de règles OPA de base

Dans le schéma précédent, l'une des règles de contrôle d'accès appliquées par l'OPA pour le microservice Payroll est la suivante :

Les employés peuvent lire leur propre salaire.

Si Bob essaie d'accéder au microservice de paie pour voir son propre salaire, le microservice de paie peut rediriger l'appel d'API vers l' RESTful API OPA pour prendre une décision d'accès. Le service de paie demande à OPA une décision avec l'entrée JSON suivante :

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

L'OPA sélectionne une ou plusieurs politiques en fonction de la requête. Dans ce cas, la politique suivante, écrite en Rego, évalue l'entrée JSON.

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

Cette politique refuse l'accès par défaut. Il évalue ensuite l'entrée de la requête en la liant à la variable `input` globale. L'opérateur point est utilisé avec cette variable pour accéder aux valeurs de la variable. La règle Rego `allow` renvoie la valeur true si les expressions de la règle sont également vraies. La règle Rego vérifie que l'`method`entrée est égale à GET. Il vérifie ensuite que le premier élément de la liste `path` est bien présent `getSalary` avant d'affecter le deuxième élément de la liste à la variable. `user` Enfin, il vérifie que le chemin auquel on accède est `/getSalary/bob` en vérifiant que l'`user`envoi de la demande correspond à la `user` variable. `input.user` La règle `allow` applique la logique if-then pour renvoyer une valeur booléenne, comme indiqué dans le résultat :

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

Règle partielle utilisant des données externes

Pour démontrer des fonctionnalités OPA supplémentaires, vous pouvez ajouter des exigences à la règle d'accès que vous appliquez. Supposons que vous souhaitiez appliquer cette exigence de contrôle d'accès dans le contexte de l'illustration précédente :

Les employés peuvent lire le salaire de toute personne relevant d'eux.

Dans cet exemple, l'OPA a accès à des données externes qui peuvent être importées pour aider à prendre une décision d'accès :

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

Vous pouvez générer une réponse JSON arbitraire en créant une règle partielle dans OPA, qui renvoie un ensemble de valeurs au lieu d'une réponse fixe. Voici un exemple de règle partielle :

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

Cette règle renvoie un ensemble de tous les utilisateurs qui répondent à la valeur de`input.user`, qui, dans ce cas, est`bob`. La `[_]` construction de la règle est utilisée pour itérer les valeurs de l'ensemble. Voici le résultat de la règle :

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

La récupération de ces informations peut aider à déterminer si un utilisateur est un subordonné direct d'un responsable. Pour certaines applications, il est préférable de renvoyer un JSON dynamique plutôt qu'une simple réponse booléenne.

Tout assembler

La dernière exigence d'accès est plus complexe que les deux premières car elle combine les conditions spécifiées dans les deux exigences :

Les employés peuvent lire leur propre salaire et celui de toute personne relevant d'eux.

Pour répondre à cette exigence, vous pouvez utiliser cette politique Rego :

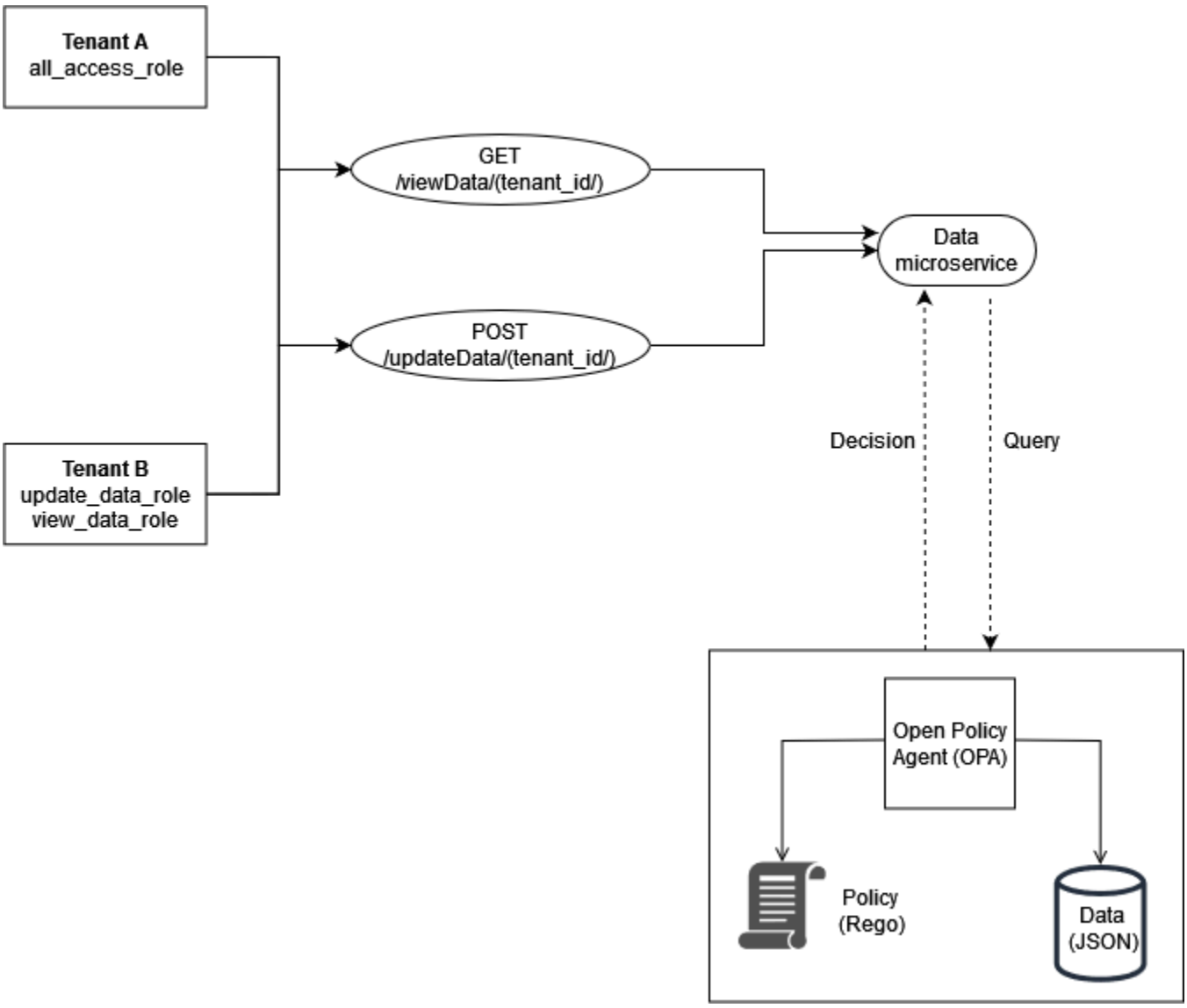
```
default allow = false

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 première règle de la politique permet l'accès à tout utilisateur qui essaie de consulter ses propres informations salariales, comme indiqué précédemment. Le fait d'avoir deux règles portant le même nom fonctionne comme un opérateur logique dans Rego. **`allow`** La deuxième règle extrait la liste de tous les subordonnés directs associés à `input.user` (à partir des données du schéma précédent) et affecte cette liste à la `managers` variable. Enfin, la règle vérifie si l'utilisateur qui essaie de voir son salaire est un subordonné direct `input.user` en vérifiant que son nom est contenu dans la `managers` variable.

Les exemples de cette section sont très basiques et ne fournissent pas une exploration complète ou approfondie des capacités de Rego et OPA. Pour plus d'informations, consultez la documentation OPA, consultez le fichier GitHub README OPA et expérimentez dans le terrain de jeu Rego.

Exemple 2 : contrôle d'accès multi-locataires et RBAC défini par l'utilisateur avec OPA et Rego

Cet exemple utilise OPA et Rego pour montrer comment le contrôle d'accès peut être implémenté sur une API pour une application multi-locataires avec des rôles personnalisés définis par les utilisateurs locataires. Il montre également comment l'accès peut être restreint en fonction d'un locataire. Ce modèle montre comment l'OPA peut prendre des décisions d'autorisation détaillées sur la base des informations fournies dans le cadre d'un rôle de haut niveau.



Les rôles des locataires sont stockés dans des données externes (données RBAC) qui sont utilisées pour prendre des décisions d'accès pour l'OPA :

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

```
}
```

Ces rôles, lorsqu'ils sont définis par un utilisateur locataire, doivent être stockés dans une source de données externe ou dans un fournisseur d'identité (IdP) qui peut servir de source de vérité lors du mappage des rôles définis par le locataire aux autorisations et au locataire lui-même.

Cet exemple utilise deux politiques de l'OPA pour prendre des décisions d'autorisation et pour examiner comment ces politiques renforcent l'isolement des locataires. Ces politiques utilisent les données RBAC définies précédemment.

```
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")
}
```

Pour montrer comment cette règle fonctionnera, considérez une requête OPA contenant les entrées suivantes :

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

Une décision d'autorisation pour cet appel d'API est prise comme suit, en combinant les données RBAC, les politiques OPA et l'entrée de requête OPA :

1. Un utilisateur de `Tenant A` envoie un appel d'API à`/viewData/tenant_a`.
2. Le microservice Data reçoit l'appel et interroge la `allowViewData` règle, en transmettant l'entrée indiquée dans l'exemple de saisie de requête OPA.
3. L'OPA utilise la règle demandée dans les politiques de l'OPA pour évaluer les données fournies. L'OPA utilise également les données du RBAC pour évaluer l'entrée. L'OPA effectue les opérations suivantes :

a. Vérifie que la méthode utilisée pour effectuer l'appel d'API est`GET`.

b. Vérifie que le chemin demandé est`viewData`.

c. Vérifie que `tenant_id` le chemin d'accès est égal à celui `input.tenant_id` associé à l'utilisateur. Cela garantit le maintien de l'isolement des locataires. Un autre locataire, même avec un rôle identique, ne peut pas être autorisé à effectuer cet appel d'API.

d. Extrait une liste d'autorisations de rôles à partir des données externes des rôles et les affecte à la variable. `role_permissions` Cette liste est extraite à l'aide du rôle défini par le locataire associé à l'utilisateur dans `input.role.`

e. Vérifie `role_permissions` s'il contient l'autorisation `viewData.`

4. L'OPA renvoie la décision suivante au microservice Data :

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

Ce processus montre comment le RBAC et la sensibilisation des locataires peuvent contribuer à prendre une décision d'autorisation auprès de l'OPA. Pour mieux illustrer ce point, considérez un appel d'API `/viewData/tenant_b` avec l'entrée de requête suivante :

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

Cette règle renverrait le même résultat que l'entrée de requête OPA, bien qu'elle soit destinée à un locataire différent ayant un rôle différent. Cela est dû au fait que cet appel est destiné `/tenant_b` et que `view_data_role` les données du RBAC ont toujours l'`viewData`autorisation qui leur est associée. Pour appliquer le même type de contrôle d'accès pour`/updateData`, vous pouvez utiliser une règle OPA similaire :

```
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")
}
```

Cette règle est fonctionnellement identique à la `allowViewData` règle, mais elle vérifie un chemin et une méthode de saisie différents. La règle garantit toujours l'isolement du locataire et vérifie que le rôle défini par le locataire accorde l'autorisation à l'appelant de l'API. Pour voir comment cela peut être appliqué, examinez l'entrée de requête suivante pour un appel d'API à `/updateData/tenant_b` :

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

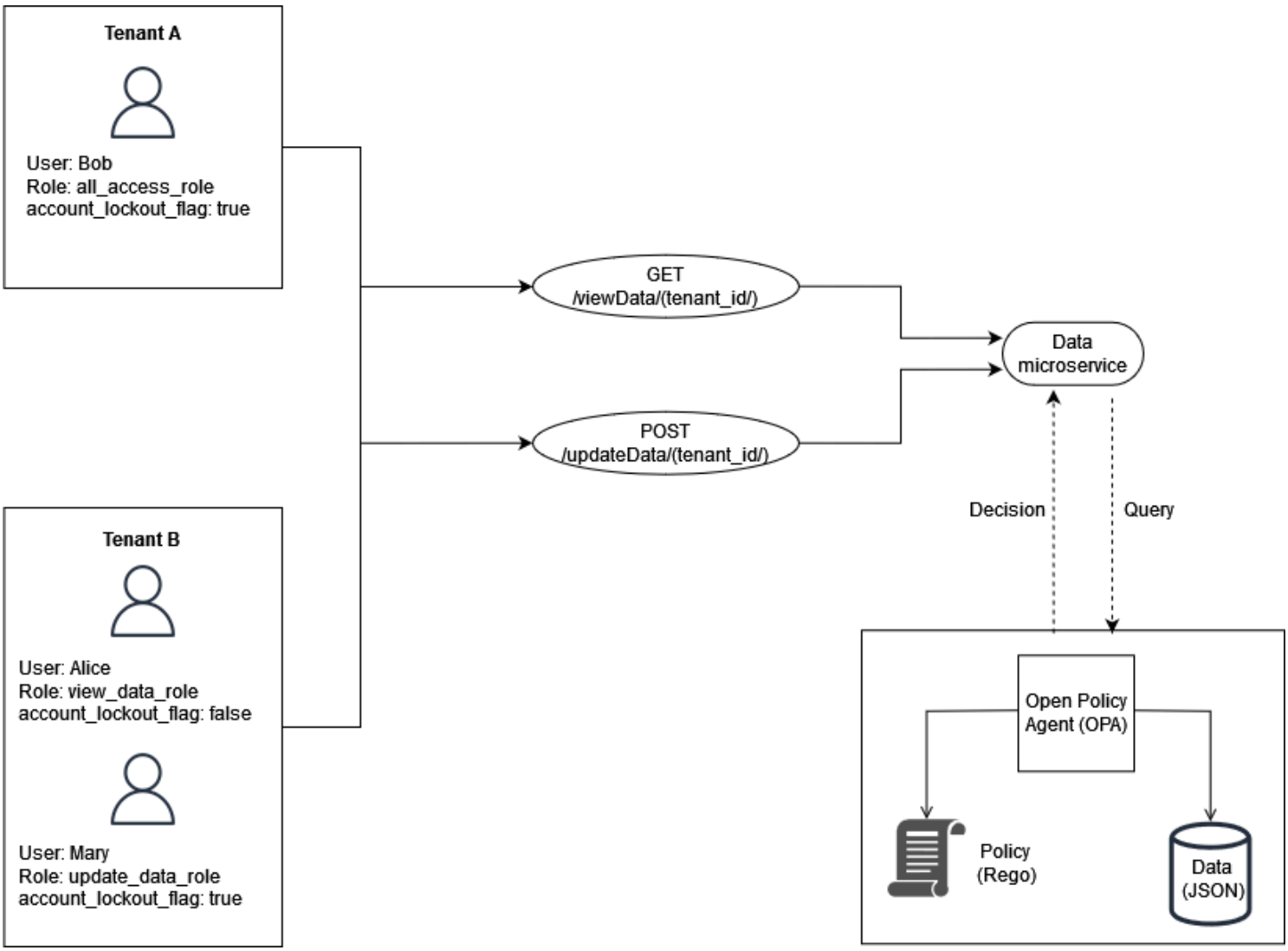
Cette entrée de requête, lorsqu'elle est évaluée à l'aide de la `allowUpdateData` règle, renvoie la décision d'autorisation suivante :

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

Cet appel ne sera pas autorisé. Bien que l'appelant de l'API soit associé à la bonne API `tenant_id` et appelle l'API à l'aide d'une méthode approuvée, il s'`input.role`agit de la méthode définie par le locataire`view_data_role`. Le `view_data_role` n'a pas l'`updateData`autorisation ; par conséquent, l'appel à `/updateData` n'est pas autorisé. Cet appel aurait été un succès pour un `tenant_b` utilisateur possédant le`update_data_role`.

Exemple 3 : Contrôle d'accès multi-locataires pour RBAC et ABAC avec OPA et Rego

Pour améliorer l'exemple RBAC présenté dans la section précédente, vous pouvez ajouter des attributs aux utilisateurs.



Cet exemple inclut les mêmes rôles que dans l'exemple précédent, mais ajoute l'attribut user`account_lockout_flag`. Il s'agit d'un attribut spécifique à l'utilisateur qui n'est associé à aucun rôle en particulier. Vous pouvez utiliser les mêmes données externes RBAC que celles que vous avez utilisées précédemment pour cet exemple :

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

```
}
```

L'attribut `account_lockout_flag` utilisateur peut être transmis au service de données dans le cadre de la saisie d'une requête OPA `/viewData/tenant_a` pour l'utilisateur Bob :

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

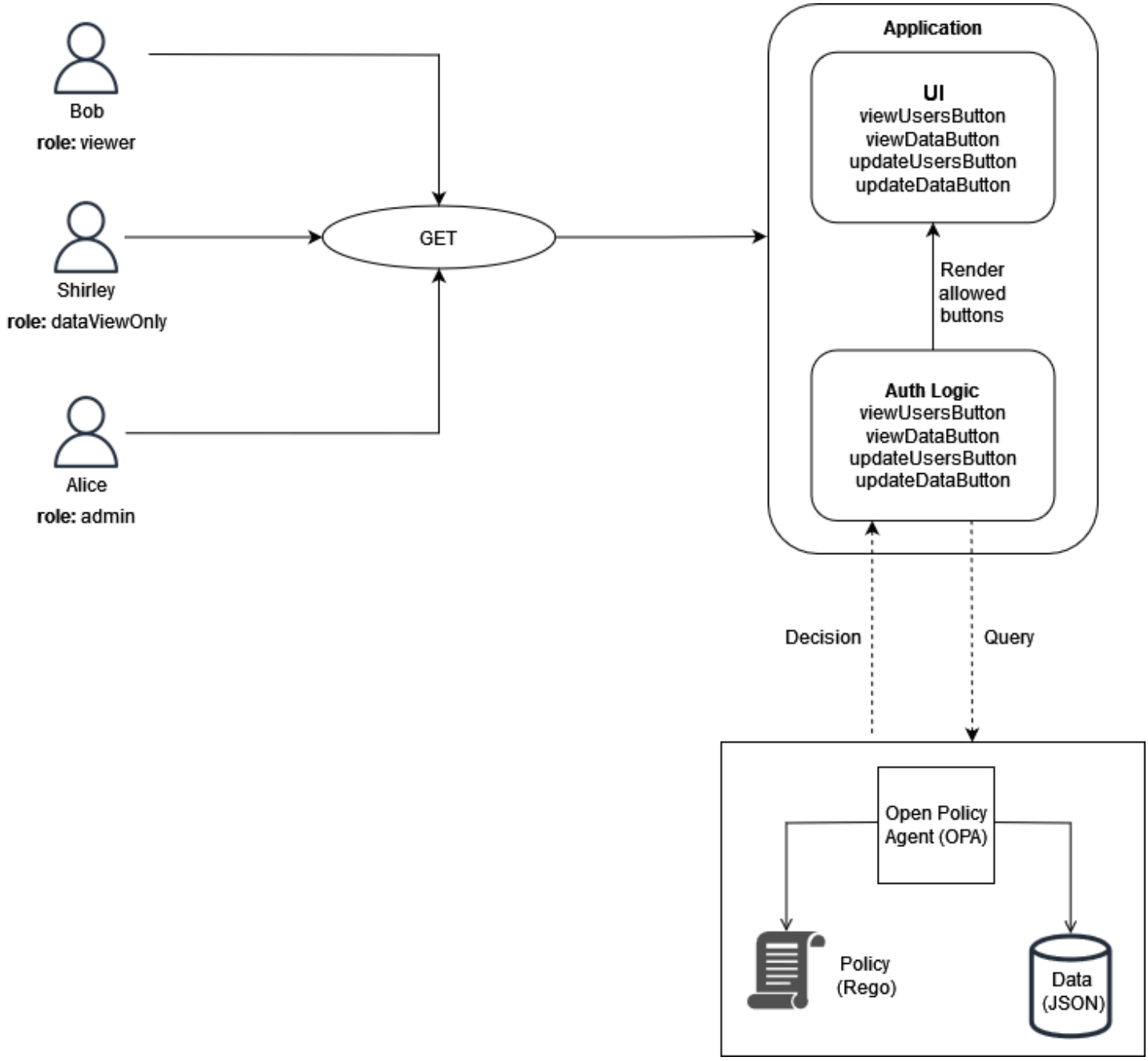
La règle demandée pour la décision d'accès est similaire aux exemples précédents, mais inclut une ligne supplémentaire pour vérifier l'`account_lockout_flag`attribut :

```
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"
}
```

Cette requête renvoie une décision d'autorisation de`false`. Cela est dû au fait que `account_lockout_flag attribute` c'est `true` pour Bob et que la règle Rego `allowViewData` refuse l'accès bien que Bob ait le bon rôle et le bon locataire.

Exemple 4 : filtrage de l'interface utilisateur avec OPA et Rego

La flexibilité d'OPA et de Rego permet de filtrer les éléments de l'interface utilisateur. L'exemple suivant montre comment une règle partielle OPA peut prendre des décisions d'autorisation concernant les éléments à afficher dans une interface utilisateur avec RBAC. Cette méthode est l'une des nombreuses manières de filtrer les éléments de l'interface utilisateur avec OPA.



Dans cet exemple, une application Web d'une seule page comporte quatre boutons. Supposons que vous souhaitiez filtrer l'interface utilisateur de Bob, Shirley et Alice afin qu'ils ne puissent voir que les boutons correspondant à leurs rôles. Lorsque l'interface utilisateur reçoit une demande de l'utilisateur, elle interroge une règle partielle OPA pour déterminer quels boutons doivent être affichés dans l'interface utilisateur. La requête transmet les informations suivantes en entrée à OPA lorsque Bob (avec le rôle`viewer`) envoie une demande à l'interface utilisateur :

```
{
```

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

L'OPA utilise des données externes structurées pour que le RBAC prenne une décision d'accès :

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

La règle partielle OPA utilise à la fois les données externes et les données d'entrée pour produire une liste des actions autorisées :

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

Dans la règle partielle, OPA utilise le `input.role` paramètre spécifié dans le cadre de la requête pour déterminer quels boutons doivent être affichés. Bob a le rôle`viewer`, et les données externes indiquent que les spectateurs ont deux autorisations : `viewUsers` et`viewData`. Par conséquent, le résultat de cette règle pour Bob (et pour tout autre utilisateur ayant un rôle de spectateur) est le suivant :

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

La sortie pour Shirley, qui a le `dataViewOnly` rôle, contiendrait un bouton d'autorisation :`viewData`. La sortie pour Alice, qui a le `admin` rôle, contiendrait toutes ces autorisations.
Ces réponses sont renvoyées à l'interface utilisateur lorsque l'OPA est demandée.
`user_permissions` L'application peut ensuite utiliser cette réponse pour masquer ou afficher le `viewUsersButton`,`viewDataButton`,`updateUsersButton`, et le`updateDataButton`.

Utilisation d'un moteur de politique personnalisé

Une autre méthode pour implémenter un PDP consiste à créer un moteur de politiques personnalisé. L'objectif de ce moteur de politique est de dissocier la logique d'autorisation d'une application. Le moteur de politique personnalisé est chargé de prendre des décisions d'autorisation, similaires aux autorisations vérifiées ou OPA, pour réaliser le découplage des politiques. La principale différence entre cette solution et l'utilisation des autorisations vérifiées ou OPA réside dans le fait que la logique de rédaction et d'évaluation des politiques est conçue sur mesure pour un moteur de politiques personnalisé. Toute interaction avec le moteur doit être exposée via une API ou une autre méthode pour permettre aux décisions d'autorisation d'atteindre une application. Vous pouvez écrire un moteur de politique personnalisé dans n'importe quel langage de programmation ou utiliser d'autres mécanismes pour évaluer les politiques, tels que le langage d'expression commun (CEL).

Mettre en œuvre un PEP

Un point d'application des politiques (PEP) est chargé de recevoir les demandes d'autorisation qui sont envoyées au point de décision politique (PDP) pour évaluation. Un PEP peut se trouver n'importe où dans une application où les données et les ressources doivent être protégées ou où une logique d'autorisation est appliquée. PEPs sont relativement simples par rapport à PDPs. Un PEP est uniquement chargé de demander et d'évaluer une décision d'autorisation et ne nécessite aucune logique d'autorisation. PEPs, au contraire PDPs, ne peut pas être centralisé dans une application SaaS. Cela est dû au fait que l'autorisation et le contrôle d'accès doivent être mis en œuvre dans l'ensemble d'une application et de ses points d'accès. PEPs peut être appliqué à des microservices APIs, à des couches Backend for Frontend (BFF) ou à tout point de l'application où le contrôle d'accès est souhaité ou requis. L' PEPs omniprésence dans une application garantit que l'autorisation est vérifiée souvent et indépendamment à plusieurs reprises.

Pour mettre en œuvre un PEP, la première étape consiste à déterminer où le contrôle d'accès doit être appliqué dans une application. Tenez compte de ce principe lorsque vous PEPs décidez où intégrer dans votre application :

Si une application expose une API, cette API doit être dotée d'une autorisation et d'un contrôle d'accès.

En effet, dans une architecture orientée vers les microservices ou orientée services, ils APIs servent de séparateurs entre les différentes fonctions de l'application. Il est logique d'inclure le contrôle d'accès comme point de contrôle logique entre les fonctions de l'application. Nous vous recommandons vivement de l'inclure PEPs comme condition préalable à l'accès à chaque API dans une application SaaS. Il est également possible d'intégrer l'autorisation à d'autres points d'une application. Dans les applications monolithiques, il peut être nécessaire de les PEPs intégrer dans la logique de l'application elle-même. Il n'y a pas d'emplacement unique qui PEPs devrait être inclus, mais pensez à utiliser le principe de l'API comme point de départ.

Demande de décision d'autorisation

Un PEP doit demander une décision d'autorisation au PDP. La demande peut prendre plusieurs formes. La méthode la plus simple et la plus accessible pour demander une décision d'autorisation consiste à envoyer une demande d'autorisation ou une requête à une RESTful API exposée par le PDP (OPA ou Verified Permissions). Si vous utilisez des autorisations vérifiées, vous pouvez également appeler la IsAuthorizedméthode en utilisant le AWS SDK pour récupérer une décision

d'autorisation. La seule fonction d'un PEP dans ce modèle est de transmettre les informations nécessaires à la demande d'autorisation ou à la requête. Cela peut être aussi simple que de transférer une demande reçue par une API en entrée au PDP. Il existe d'autres méthodes de création PEPs. Par exemple, vous pouvez intégrer un PDP OPA localement à une application écrite dans le langage de programmation Go sous forme de bibliothèque au lieu d'utiliser une API.

Évaluation d'une décision d'autorisation

PEPs nécessité d'inclure une logique pour évaluer les résultats d'une décision d'autorisation. Lorsqu'elles PDPs sont exposées en tant que APIs, la décision d'autorisation est probablement au format JSON et renvoyée par un appel d'API. Le PEP doit évaluer ce code JSON pour déterminer si l'action entreprise est autorisée. Par exemple, si un PDP est conçu pour fournir une décision booléenne d'autorisation ou de refus, le PEP peut simplement vérifier cette valeur, puis renvoyer le code d'état HTTP 200 pour autoriser et le code d'état HTTP 403 pour refuser. Ce modèle d'incorporation d'un PEP comme condition préalable à l'accès à une API est un modèle facile à mettre en œuvre et très efficace pour mettre en œuvre le contrôle d'accès dans une application SaaS. Dans des scénarios plus complexes, le PEP peut être chargé d'évaluer le code JSON arbitraire renvoyé par le PDP. Le PEP doit être écrit de manière à inclure toute la logique nécessaire pour interpréter la décision d'autorisation renvoyée par le PDP. Étant donné qu'un PEP est susceptible d'être implémenté à de nombreux endroits différents de votre application, nous vous recommandons de packager votre code PEP sous forme de bibliothèque ou d'artefact réutilisable dans le langage de programmation de votre choix. Ainsi, votre PEP peut être facilement intégré à n'importe quel point de votre application avec un minimum de retouches.

Modèles de conception pour les architectures SaaS à locataires multiples

Il existe de nombreuses manières de mettre en œuvre le contrôle d'accès et l'autorisation des API. Ce guide se concentre sur trois modèles de conception efficaces pour les architectures SaaS multi-locataires. Ces conceptions servent de référence de haut niveau pour la mise en œuvre des points de décision politique (PDPs) et des points d'application des politiques (PEPs), afin de former un modèle d'autorisation cohérent et omniprésent pour les applications.

Modèles de design :

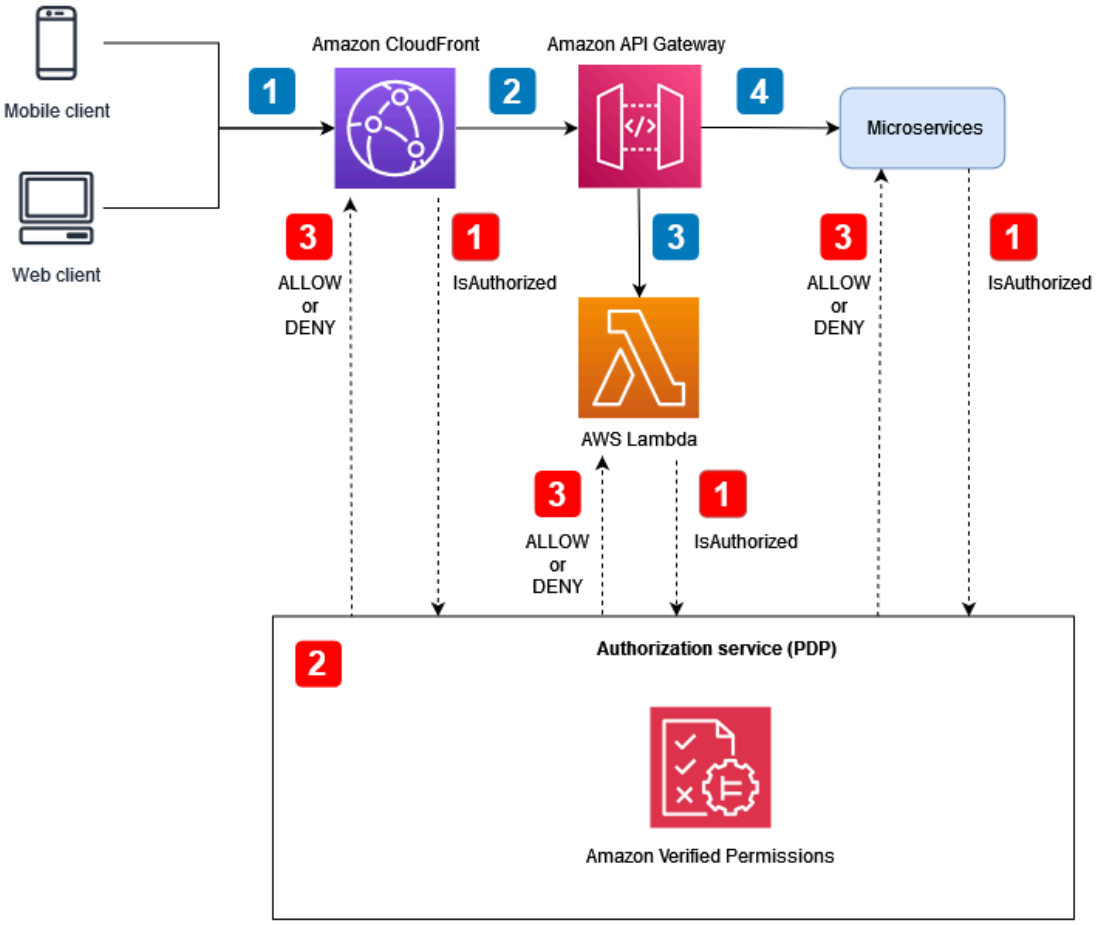
- Modèles de conception pour Amazon Verified Permissions
- Modèles de conception pour OPA

Modèles de conception pour Amazon Verified Permissions

Utilisation d'un PDP centralisé avec activé PEPs APIs

Le point de décision stratégique (PDP) centralisé avec points d'application des politiques (PEPs) sur le APIs modèle suit les meilleures pratiques du secteur afin de créer un système efficace et facile à entretenir pour le contrôle d'accès et l'autorisation des API. Cette approche repose sur plusieurs principes clés :

- L'autorisation et le contrôle d'accès aux API sont appliqués à plusieurs points de l'application.
- La logique d'autorisation est indépendante de l'application.
- Les décisions relatives au contrôle d'accès sont centralisées.



Flux de candidature (illustré par des légendes numérotées en bleu dans le schéma) :

1. Un utilisateur authentifié avec un jeton Web JSON (JWT) génère une requête HTTP à Amazon. CloudFront
2. CloudFront transmet la demande à Amazon API Gateway, qui est configuré comme CloudFront origine.
3. Un autorisateur personnalisé API Gateway est appelé pour vérifier le JWT.
4. Les microservices répondent à la demande.

Flux d'autorisation et de contrôle d'accès aux API (illustré par des légendes numérotées en rouge dans le schéma) :

1. Le PEP appelle le service d'autorisation et transmet les données de demande, y compris les JWTs données.

2. Le service d'autorisation (PDP), dans ce cas Verified Permissions, utilise les données de la demande comme entrée de requête et les évalue en fonction des politiques pertinentes spécifiées par la requête.
3. La décision d'autorisation est renvoyée au PEP et évaluée.

Ce modèle utilise un PDP centralisé pour prendre les décisions d'autorisation. PEPs sont mis en œuvre à différents moments pour faire des demandes d'autorisation au PDP. Le schéma suivant montre comment implémenter ce modèle dans une application SaaS à locataires multiples hypothétique.

Dans cette architecture, PEPs demandez des décisions d'autorisation aux points de terminaison des services pour Amazon CloudFront et Amazon API Gateway et pour chaque microservice. La décision d'autorisation est prise par le service d'autorisation Amazon Verified Permissions (le PDP). Les autorisations vérifiées étant un service entièrement géré, vous n'avez pas à gérer l'infrastructure sous-jacente. Vous pouvez interagir avec les autorisations vérifiées à l'aide d'une RESTful API ou du AWS SDK.

Vous pouvez également utiliser cette architecture avec des moteurs de politiques personnalisés. Cependant, tous les avantages découlant des autorisations vérifiées doivent être remplacés par une logique fournie par le moteur de politique personnalisé.

Un PDP centralisé PEPs activé APIs permet de créer facilement un système d'autorisation robuste pour APIs. Cela simplifie le processus d'autorisation et fournit également une easy-to-use interface reproductible pour prendre des décisions d'autorisation pour les microservices APIs, les couches Backend for Frontend (BFF) ou d'autres composants de l'application.

Utilisation du SDK Cedar

Amazon Verified Permissions utilise le langage Cedar pour gérer les autorisations détaillées dans vos applications personnalisées. Avec les autorisations vérifiées, vous pouvez stocker les politiques Cedar dans un emplacement central, profiter d'une faible latence grâce à un traitement en millisecondes et auditer les autorisations dans différentes applications. Vous pouvez également éventuellement intégrer le SDK Cedar directement dans votre application pour prendre des décisions d'autorisation sans utiliser d'autorisations vérifiées. Cette option nécessite le développement d'applications personnalisées supplémentaires pour gérer et stocker les politiques adaptées à votre cas d'utilisation. Cependant, il peut s'agir d'une alternative viable, en particulier dans les cas où l'accès aux autorisations vérifiées est intermittent ou impossible en raison d'une connectivité Internet incohérente.

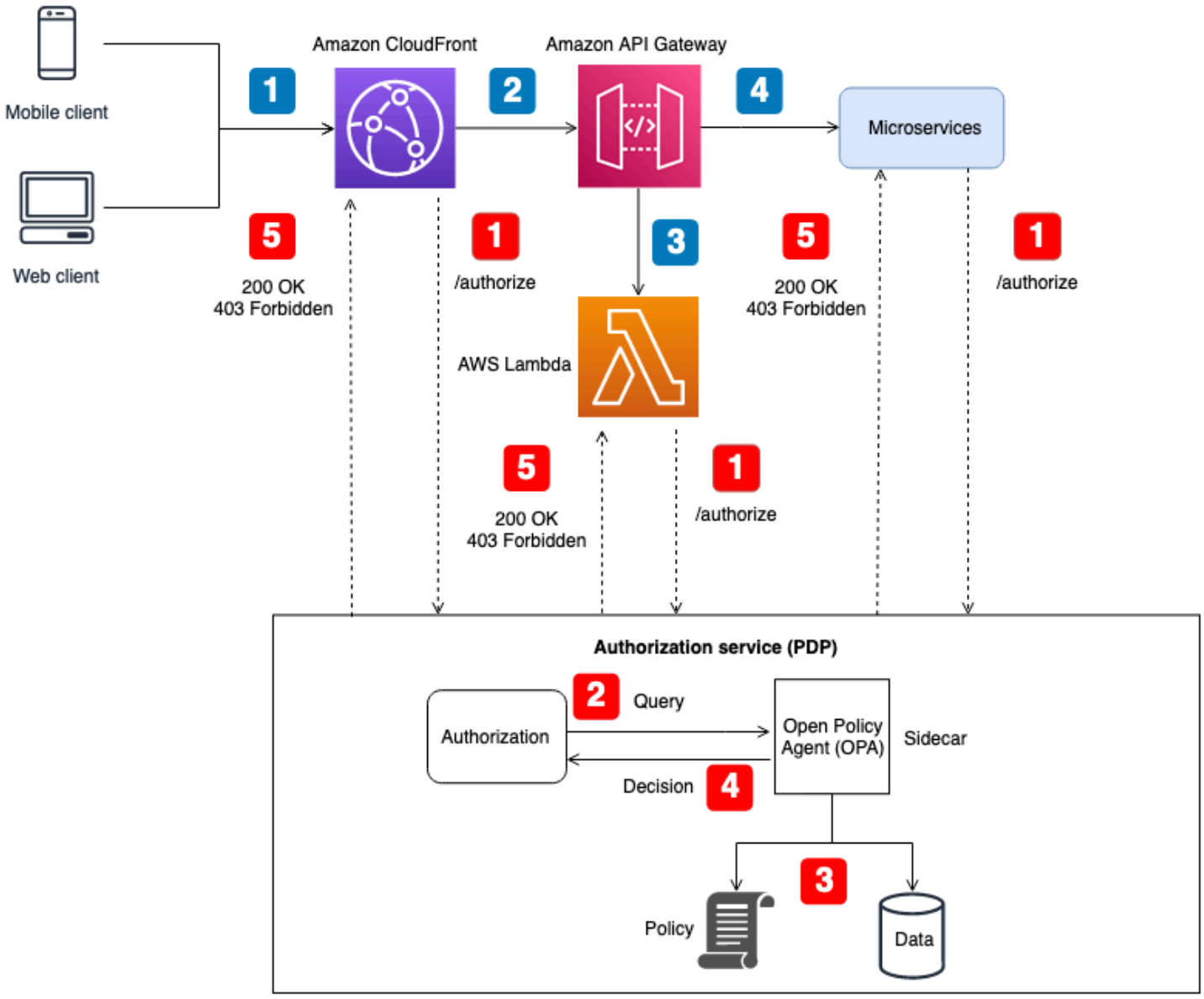
Modèles de conception pour OPA

Utilisation d'un PDP centralisé avec activé PEPs APIs

Le point de décision stratégique (PDP) centralisé avec points d'application des politiques (PEPs) sur le APIs modèle suit les meilleures pratiques du secteur afin de créer un système efficace et facile à entretenir pour le contrôle d'accès et l'autorisation des API. Cette approche repose sur plusieurs principes clés :

- L'autorisation et le contrôle d'accès aux API sont appliqués à plusieurs points de l'application.
- La logique d'autorisation est indépendante de l'application.
- Les décisions relatives au contrôle d'accès sont centralisées.

Ce modèle utilise un PDP centralisé pour prendre les décisions d'autorisation. PEPs sont mis en œuvre APIs pour faire des demandes d'autorisation au PDP. Le schéma suivant montre comment implémenter ce modèle dans une application SaaS à locataires multiples hypothétique.



Flux de candidature (illustré par des légendes numérotées en bleu dans le schéma) :

1. Un utilisateur authentifié avec un JWT génère une requête HTTP à Amazon. CloudFront
2. CloudFront transmet la demande à Amazon API Gateway, qui est configuré comme CloudFront origine.
3. Un autorisateur personnalisé API Gateway est appelé pour vérifier le JWT.
4. Les microservices répondent à la demande.

Flux d'autorisation et de contrôle d'accès aux API (illustré par des légendes numérotées en rouge dans le schéma) :

1. Le PEP appelle le service d'autorisation et transmet les données de demande, y compris les JWTs données.
2. Le service d'autorisation (PDP) prend les données de la demande et interroge une API REST de l'agent OPA, qui fonctionne comme un sidecar. Les données de la demande servent d'entrée à la requête.
3. L'OPA évalue l'entrée en fonction des politiques pertinentes spécifiées par la requête. Les données sont importées pour prendre une décision d'autorisation si nécessaire.
4. L'OPA renvoie une décision au service d'autorisation.
5. La décision d'autorisation est renvoyée au PEP et évaluée.

Dans cette architecture, PEPs demandez des décisions d'autorisation aux points de terminaison des services pour Amazon CloudFront et Amazon API Gateway, ainsi que pour chaque microservice. La décision d'autorisation est prise par un service d'autorisation (le PDP) avec un sidecar OPA. Vous pouvez utiliser ce service d'autorisation en tant que conteneur ou en tant qu'instance de serveur traditionnelle. Le sidecar OPA expose son RESTful API localement afin que l'API ne soit accessible qu'au service d'autorisation. Le service d'autorisation expose une API distincte qui est disponible pour PEPs. Le fait que le service d'autorisation joue le rôle d'intermédiaire entre PEPs un OPA permet d'insérer toute logique de transformation entre PEPs un OPA qui peut être nécessaire, par exemple lorsque la demande d'autorisation d'un PEP n'est pas conforme à la requête saisie par l'OPA.

Vous pouvez également utiliser cette architecture avec des moteurs de politiques personnalisés. Cependant, tous les avantages obtenus grâce à l'OPA doivent être remplacés par une logique fournie par le moteur de politique personnalisé.

Un PDP centralisé PEPs activé APIs permet de créer facilement un système d'autorisation robuste pour APIs. Il est simple à mettre en œuvre et fournit également une easy-to-use interface reproductible pour prendre des décisions d'autorisation pour les microservices APIs, les couches Backend for Frontend (BFF) ou d'autres composants de l'application. Cependant, cette approche peut créer trop de latence dans votre application, car les décisions d'autorisation nécessitent l'appel d'une API distincte. Si la latence du réseau pose problème, vous pouvez envisager un PDP distribué.

Utilisation d'un PDP distribué avec PEPs on APIs

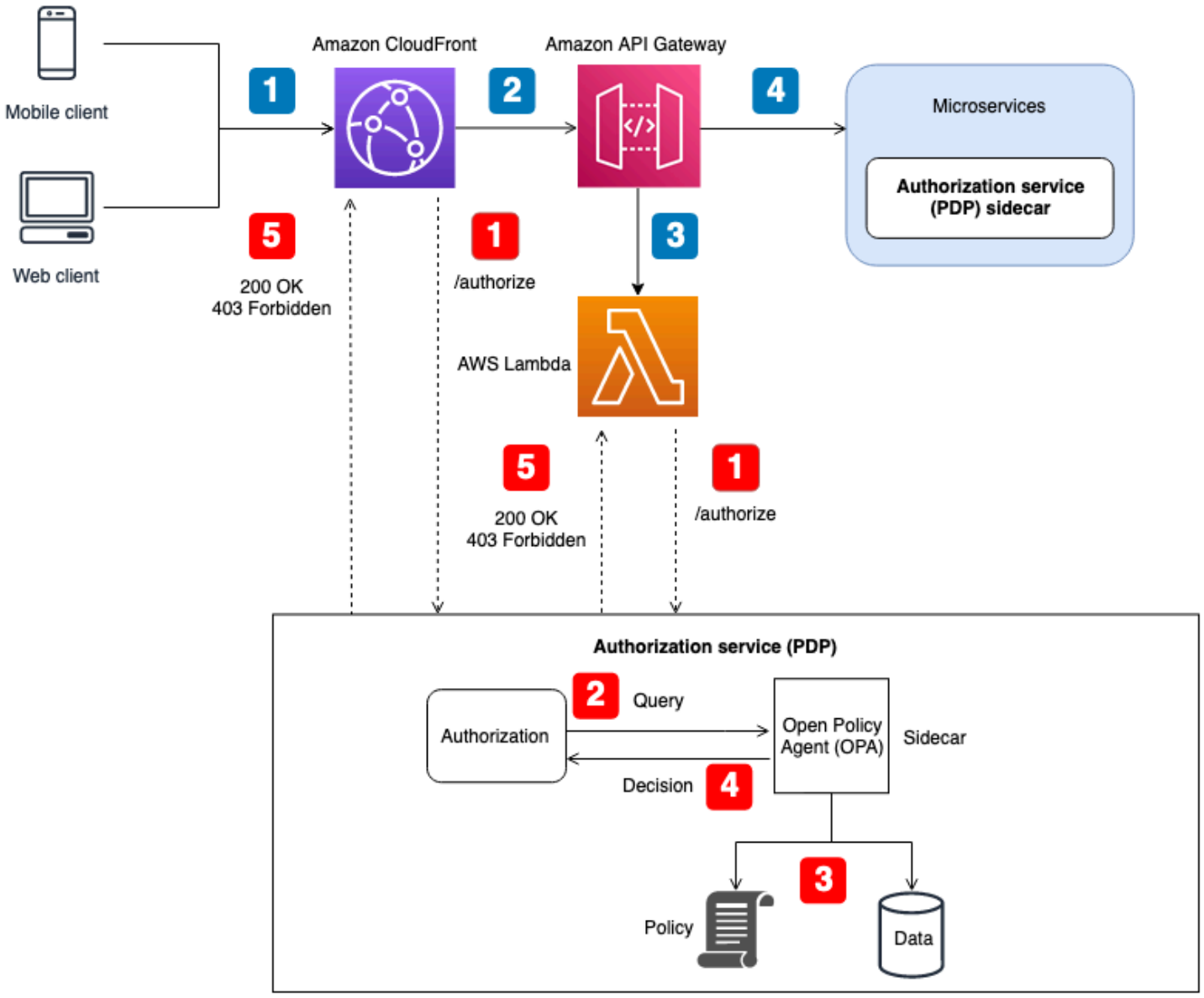
Le APIs modèle de point de décision stratégique distribué (PDP) avec points d'application des politiques (PEPs) suit les meilleures pratiques du secteur afin de créer un système efficace de contrôle d'accès et d'autorisation des API. Comme dans le cas du PDP centralisé avec un APIs modèle, cette approche repose PEPs sur les principes clés suivants :

- L'autorisation et le contrôle d'accès aux API sont appliqués à plusieurs points de l'application.
- La logique d'autorisation est indépendante de l'application.
- Les décisions relatives au contrôle d'accès sont centralisées.

Vous vous demandez peut-être pourquoi les décisions relatives au contrôle d'accès sont centralisées lorsque le PDP est distribué. Bien que le PDP puisse exister à plusieurs endroits dans une application, il doit utiliser la même logique d'autorisation pour prendre des décisions de contrôle d'accès. Tous PDPs fournissent les mêmes décisions de contrôle d'accès avec les mêmes entrées. PEPs sont mis en œuvre APIs pour faire des demandes d'autorisation au PDP. La figure suivante montre comment ce modèle distribué peut être implémenté dans une application SaaS hypothétique à locataires multiples.

Dans cette approche, les PDP sont implémentés à plusieurs endroits de l'application. Pour les composants d'application dotés de capacités de calcul intégrées capables d'exécuter l'OPA et de prendre en charge un PDP, tels qu'un service conteneurisé avec un sidecar ou une instance Amazon Elastic Compute Cloud (Amazon EC2), les décisions relatives au PDP peuvent être intégrées directement dans le composant d'application sans qu'il soit nécessaire de passer un appel d'API à RESTful un service PDP centralisé. Cela présente l'avantage de réduire la latence que vous pourriez rencontrer dans le modèle PDP centralisé, car tous les composants de l'application n'ont pas besoin d'effectuer des appels d'API supplémentaires pour obtenir des décisions d'autorisation. Toutefois, un PDP centralisé est toujours nécessaire dans ce modèle pour les composants d'application qui ne disposent pas de capacités de calcul intégrées permettant l'intégration directe d'un PDP, tels que le service Amazon ou Amazon API CloudFront Gateway.

Le schéma suivant montre comment cette combinaison d'un PDP centralisé et d'un PDP distribué peut être mise en œuvre dans une application SaaS à locataires multiples hypothétique.



Flux de candidature (illustré par des légendes numérotées en bleu dans le schéma) :

1. Un utilisateur authentifié avec un JWT génère une requête HTTP à Amazon. CloudFront
2. CloudFront transmet la demande à Amazon API Gateway, qui est configuré comme CloudFront origine.
3. Un autorisateur personnalisé API Gateway est appelé pour vérifier le JWT.
4. Les microservices répondent à la demande.

Flux d'autorisation et de contrôle d'accès aux API (illustré par des légendes numérotées en rouge dans le schéma) :

1. Le PEP appelle le service d'autorisation et transmet les données de demande, y compris les JWTs données.
2. Le service d'autorisation (PDP) prend les données de la demande et interroge une API REST de l'agent OPA, qui fonctionne comme un sidecar. Les données de la demande servent d'entrée à la requête.
3. L'OPA évalue l'entrée en fonction des politiques pertinentes spécifiées par la requête. Les données sont importées pour prendre une décision d'autorisation si nécessaire.
4. L'OPA renvoie une décision au service d'autorisation.
5. La décision d'autorisation est renvoyée au PEP et évaluée.

Dans cette architecture, les PEP demandent des décisions d'autorisation aux points de terminaison de service pour CloudFront API Gateway et pour chaque microservice. La décision d'autorisation pour les microservices est prise par un service d'autorisation (le PDP) qui fonctionne comme un sidecar avec le composant d'application. Ce modèle est possible pour les microservices (ou services) qui s'exécutent sur des conteneurs ou sur des instances Amazon Elastic Compute Cloud (Amazon EC2). Les décisions d'autorisation pour des services tels que API Gateway CloudFront nécessiteraient toujours de contacter un service d'autorisation externe. Quoi qu'il en soit, le service d'autorisation expose une API accessible à PEPs. Le fait que le service d'autorisation joue le rôle d'intermédiaire entre PEPs un OPA permet d'insérer toute logique de transformation entre PEPs un OPA qui pourrait être nécessaire, par exemple lorsque la demande d'autorisation d'un PEP n'est pas conforme à la requête saisie par l'OPA.

Vous pouvez également utiliser cette architecture avec des moteurs de politiques personnalisés. Cependant, tous les avantages obtenus grâce à l'OPA doivent être remplacés par une logique fournie par le moteur de politique personnalisé.

Un PDP distribué PEPs activé APIs permet de créer un système d'autorisation robuste pour APIs. Il est simple à mettre en œuvre et fournit une easy-to-use interface reproductible pour prendre des décisions d'autorisation pour les microservices APIs, les couches Backend for Frontend (BFF) ou d'autres composants d'application. Cette approche présente également l'avantage de réduire la latence que vous pourriez rencontrer dans le modèle PDP centralisé.

Utilisation d'un PDP distribué en tant que bibliothèque

Vous pouvez également demander des décisions d'autorisation à un PDP mis à disposition sous forme de bibliothèque ou de package à utiliser dans une application. L'OPA peut être utilisée comme bibliothèque tierce Go. Pour les autres langages de programmation, l'adoption de ce modèle signifie généralement que vous devez créer un moteur de politique personnalisé.

Considérations relatives à la conception multi-locataires d'Amazon Verified Permissions

Plusieurs options de conception sont à prendre en compte lorsque vous implémentez l'autorisation à l'aide d'Amazon Verified Permissions dans une solution SaaS à locataires multiples. Avant d'explorer ces options, clarifiez la différence entre l'isolation et l'autorisation dans un contexte SaaS multi-locataires. L'isolation d'un tenant empêche que les données entrantes et sortantes soient exposées au mauvais locataire. L'autorisation garantit qu'un utilisateur dispose des autorisations nécessaires pour accéder à un locataire.

Dans Autorisations vérifiées, les politiques sont stockées dans un magasin de politiques. Comme décrit dans la documentation sur les autorisations vérifiées, vous pouvez soit isoler les politiques des locataires en utilisant un magasin de politiques distinct pour chaque locataire, soit autoriser les locataires à partager des politiques en utilisant un magasin de politiques unique pour tous les locataires. Cette section décrit les avantages et les inconvénients de ces deux stratégies d'isolation et décrit comment elles peuvent être déployées à l'aide d'un modèle de déploiement à plusieurs niveaux. Pour plus de contexte, consultez la documentation sur les autorisations vérifiées.

Bien que les critères abordés dans cette section soient axés sur les autorisations vérifiées, les concepts généraux sont ancrés dans l'état d'esprit d'isolement et les conseils qu'il fournit. Les applications SaaS doivent toujours prendre en compte l'isolation des locataires dans le cadre de leur conception, et ce principe général d'isolation s'étend à l'inclusion des autorisations vérifiées dans une application SaaS. Cette section fait également référence aux principaux modèles d'isolation SaaS tels que le modèle SaaS en silo et le modèle SaaS mutualisé. Pour plus d'informations, consultez les principaux concepts d'isolation dans le AWS Well-Architected Framework, SaaS Lens.

Les principales considérations à prendre en compte lors de la conception de solutions SaaS multi-locataires sont l'isolation et l'intégration des locataires. L'isolement des locataires a un impact sur la sécurité, la confidentialité, la résilience et les performances. L'intégration des locataires a un impact sur vos processus opérationnels en ce qui concerne les frais opérationnels et l'observabilité. Organisations qui optent pour le SaaS ou qui mettent en œuvre des solutions mutualisées doivent toujours prioriser la manière dont la location sera gérée par l'application SaaS. Bien qu'une solution SaaS puisse s'appuyer sur un modèle d'isolation particulier, la cohérence n'est pas nécessairement requise dans l'ensemble de la solution SaaS. Par exemple, le modèle d'isolation que vous choisissez pour les composants frontaux de votre application n'est peut-être pas le même que le modèle d'isolation que vous choisissez pour un microservice ou des services d'autorisation.

Considérations relatives à la conception :

- Intégration des locataires et enregistrement des locataires utilisateurs
- Boutique de politiques par locataire
- Un magasin de politiques partagé par plusieurs locataires
- Modèle de déploiement à plusieurs niveaux

Intégration des locataires et enregistrement des locataires utilisateurs

Les applications SaaS respectent le concept des identités SaaS et suivent les meilleures pratiques générales qui consistent à lier l'identité d'un utilisateur à l'identité d'un locataire. La liaison implique le stockage d'un identifiant de locataire sous forme de réclamation ou d'attribut pour l'utilisateur dans le fournisseur d'identité. Cela déplace la responsabilité de mapper les identités aux locataires de chaque application au processus d'enregistrement des utilisateurs. Chaque utilisateur authentifié possède alors l'identité de locataire correcte dans le cadre du jeton Web JSON (JWT).

De même, la sélection du magasin de politiques approprié pour une demande d'autorisation ne doit pas être déterminée par la logique de l'application. Pour déterminer quel magasin de politiques doit être utilisé par une demande d'autorisation particulière, établissez un mappage entre les utilisateurs et les magasins de politiques, ou entre les locataires et les magasins de politiques. Ces mappages sont généralement conservés dans un magasin de données tel qu'Amazon DynamoDB ou Amazon Relational Database Service (Amazon RDS) auquel votre application fait référence. Vous pouvez également fournir ou compléter ces mappages par les données d'un fournisseur d'identité (IdP). La relation entre les locataires, les utilisateurs et les magasins de politiques est ensuite généralement fournie à un utilisateur via un JWT qui contient toutes les relations nécessaires à une demande d'autorisation.

Cet exemple montre comment le JWT peut apparaître pour l'utilisateur`Alice`, qui appartient au locataire `TenantA` et utilise le magasin de politiques avec l'ID du magasin de politiques `ps-43214321` pour l'autorisation.

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

```
}
```

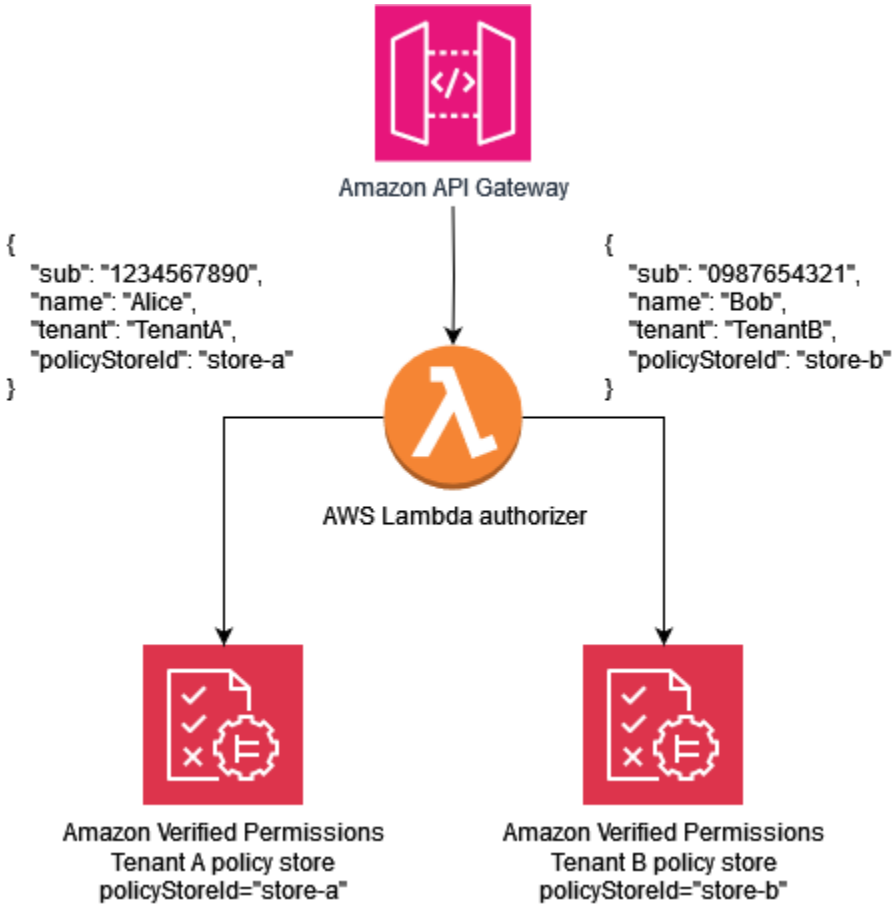
Boutique de politiques par locataire

Le modèle de conception de magasin de politiques par locataire d'Amazon Verified Permissions associe chaque locataire d'une application SaaS à son propre magasin de politiques. Ce modèle est similaire au modèle d'isolation des silos en mode SaaS. Les deux modèles imposent la création d'une infrastructure spécifique au locataire et présentent des avantages et des inconvénients similaires. Les principaux avantages de cette approche sont l'isolation des locataires imposée par l'infrastructure, la prise en charge de modèles d'autorisation uniques par locataire, l'élimination des problèmes de voisinage bruyants et la réduction de l'impact en cas d'échec des mises à jour des politiques ou des déploiements. Les inconvénients de cette approche incluent des processus d'intégration des locataires, des déploiements et des opérations plus complexes. Le magasin de politiques par locataire est l'approche recommandée si la solution dispose de politiques uniques par locataire.

Le modèle de magasin de politiques par locataire peut fournir une approche très cloisonnée de l'isolation des locataires, si votre application SaaS l'exige. Vous pouvez également utiliser ce modèle avec l'isolation du pool, mais votre implémentation des autorisations vérifiées ne bénéficiera pas des avantages standard d'un modèle d'isolation de pool plus large, tels que la simplification de la gestion et des opérations.

Dans un magasin de politiques par locataire, l'isolation des locataires est obtenue en mappant l'identifiant du magasin de politiques d'un locataire à l'identité SaaS de l'utilisateur pendant le processus d'enregistrement de l'utilisateur, comme indiqué précédemment. Cette approche lie étroitement le magasin de politiques du locataire au principal utilisateur et fournit un moyen cohérent de partager la cartographie dans l'ensemble de la solution SaaS. Vous pouvez fournir le mappage à une application SaaS en le gérant dans le cadre d'un IdP ou dans une source de données externe telle que DynamoDB. Cela garantit également que le principal fait partie du locataire et que le magasin de polices du locataire est utilisé.

Cet exemple montre comment le JWT qui contient le `policyStoreId` et `tenant` d'un utilisateur est transmis du point de terminaison d'une API au point d'évaluation des politiques dans un AWS Lambda autorisateur, qui achemine la demande vers le magasin de politiques approprié.



L'exemple de politique suivant illustre le paradigme de conception du magasin de politiques par locataire. L'utilisateur `Alice` appartient à `TenantA`. The. policyStoreId `store-a` Il est également mappé à l'identité du locataire `Alice`, et impose l'utilisation du magasin de politiques approprié. Cela garantit que les politiques de `TenantA` sont utilisées.

> **Note**
>
> Le modèle de magasin de politiques par locataire isole les politiques des locataires. L'autorisation applique les actions que les utilisateurs sont autorisés à effectuer sur leurs données. Les ressources impliquées dans toute application hypothétique utilisant ce modèle doivent être isolées à l'aide d'autres mécanismes d'isolation, tels que définis dans la documentation de AWS Well-Architected Framework, SaaS Lens.

Dans cette politique, `Alice` est autorisé à consulter les données de toutes les ressources.

```
permit (
```

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

Pour effectuer une demande d'autorisation et démarrer une évaluation avec une politique d'autorisations vérifiées, vous devez fournir l'ID du magasin de politiques qui correspond à l'identifiant unique mappé au locataire. `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":[]
            }
```

```
         ]
      ]
   }
}
```

L'utilisateur Bob appartient au locataire B et policyStoreId `store-b` est également mappé à l'identité du locataire deBob, ce qui impose l'utilisation du magasin de politiques approprié. Cela garantit que les politiques du locataire B sont utilisées.

Dans cette politique, Bob est autorisé à personnaliser les données de toutes les ressources. Dans cet exemple, il `customizeData` peut s'agir d'une action spécifique uniquement au locataire B, de sorte que la politique serait unique pour le locataire B. Le modèle de magasin de politiques par locataire prend en charge intrinsèquement les politiques personnalisées sur une base par locataire.

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

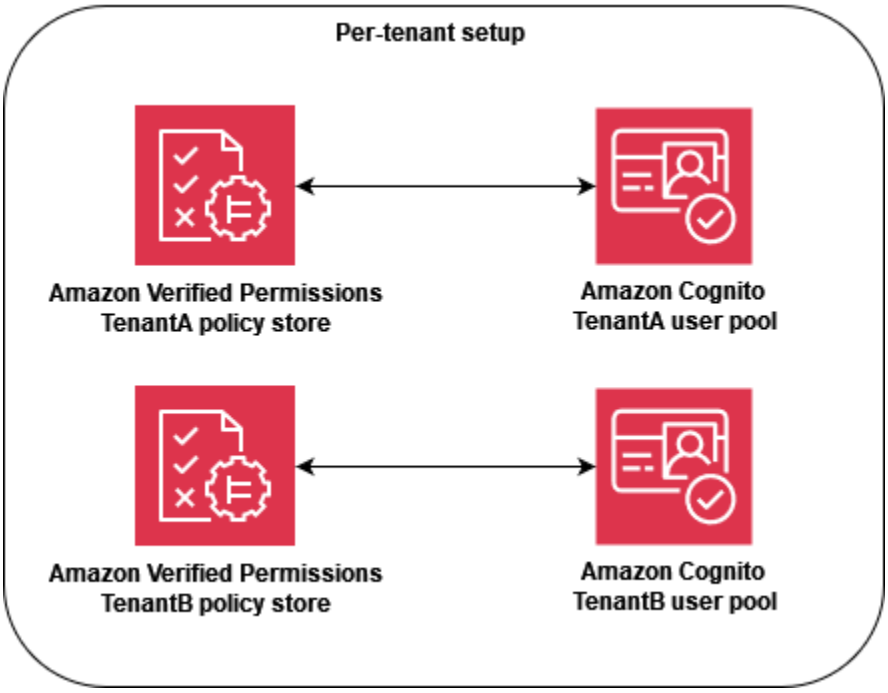
Pour effectuer une demande d'autorisation et démarrer une évaluation avec une politique d'autorisations vérifiées, vous devez fournir l'ID du magasin de politiques qui correspond à l'identifiant unique mappé au locataire. `store-b`

```
{
   "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":[]
            }
         ]
      ]
   }
}
```

Avec les autorisations vérifiées, il est possible, mais pas obligatoire, d'intégrer un IdP à un magasin de politiques. Cette intégration permet aux politiques de faire explicitement référence au principal dans le magasin d'identités en tant que principal des politiques. Pour plus d'informations sur l'intégration à Amazon Cognito en tant qu'IdP pour les autorisations vérifiées, consultez la documentation relative aux autorisations vérifiées et la documentation Amazon Cognito.

Lorsque vous intégrez un magasin de politiques à un IdP, vous ne pouvez utiliser qu'une seule source d'identité par magasin de politiques. Par exemple, si vous choisissez d'intégrer les autorisations vérifiées à Amazon Cognito, vous devez suivre la stratégie utilisée pour isoler les locataires des boutiques de politiques relatives aux autorisations vérifiées et des groupes d'utilisateurs Amazon Cognito. Les magasins de politiques et les groupes d'utilisateurs doivent également se trouver dans le même emplacement Compte AWS.



Sur le plan opérationnel, le magasin de politiques par locataire présente un avantage en matière d'audit, car vous pouvez facilement interroger l'activité enregistréeAWS CloudTrail indépendamment pour chaque locataire. Cependant, nous vous recommandons tout de même de consigner des statistiques personnalisées supplémentaires sur une dimension par locataire sur Amazon CloudWatch.

L'approche du magasin de politiques par locataire nécessite également de porter une attention particulière à deux quotas d'autorisations vérifiées afin de garantir qu'ils n'interfèrent pas avec le fonctionnement de votre solution SaaS. Ces quotas sont des Policy Stores par région et par compte et des IsAuthorized demandes par seconde, par région et par compte. Vous pouvez demander des augmentations pour les deux quotas.

Pour un exemple plus détaillé de la mise en œuvre du modèle de magasin de politiques par locataire, consultez le billet de AWS blog sur le contrôle d'accès SaaS à l'aide d'Amazon Verified Permissions avec un magasin de politiques par locataire.

Un magasin de politiques partagé par plusieurs locataires

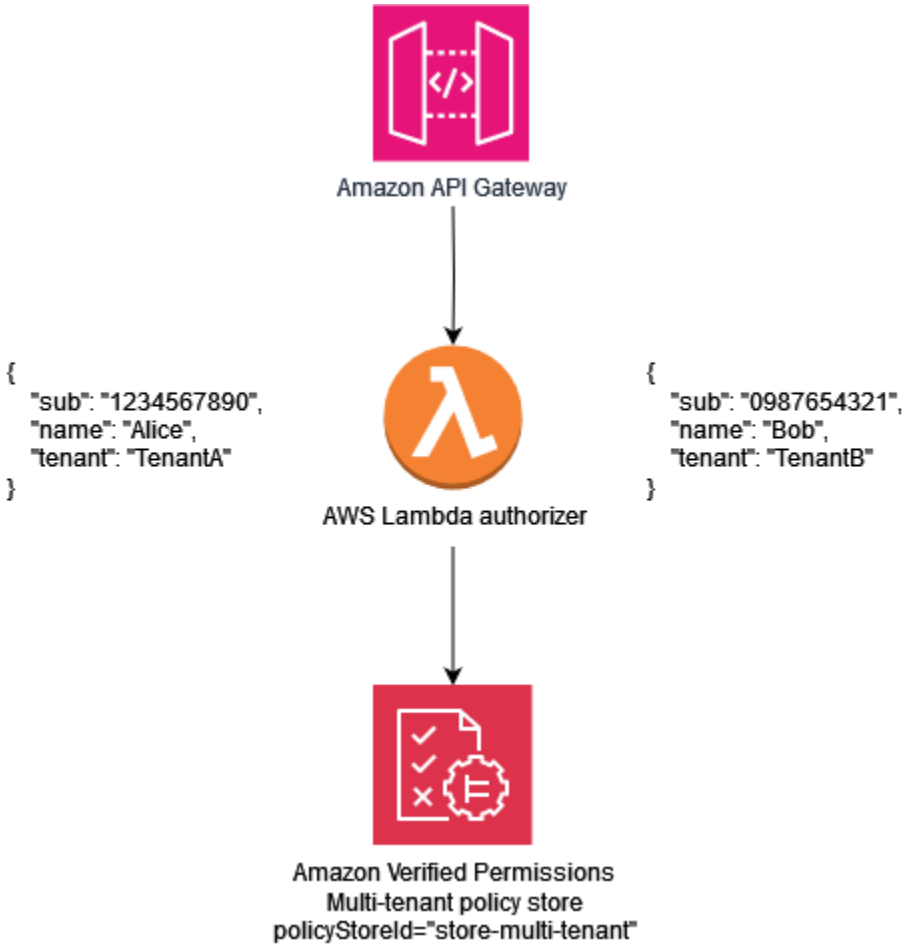
Le modèle de conception d'une boutique de politiques multi-locataires partagée utilise une seule boutique de politiques multi-locataires dans Amazon Verified Permissions pour tous les locataires de la solution SaaS. Le principal avantage de cette approche est la simplification de la

gestion et des opérations, notamment parce que vous n'avez pas à créer de magasins de polices supplémentaires lors de l'intégration des locataires. Les inconvénients de cette approche incluent une augmentation de l'impact en cas de défaillance ou d'erreur lors de la mise à jour des politiques ou des déploiements, et une plus grande exposition aux effets de voisinage bruyants. De plus, nous ne recommandons pas cette approche si votre solution nécessite des politiques uniques pour chaque locataire. Dans ce cas, utilisez plutôt le modèle de magasin de politiques par locataire pour garantir que les politiques du bon locataire sont utilisées.

L'approche d'un magasin de politiques mutualisé partagé est similaire au modèle d'isolation mutualisé SaaS. Il peut fournir une approche groupée pour isoler les locataires, si votre application SaaS l'exige. Vous pouvez également utiliser ce modèle si votre solution SaaS applique une isolation cloisonnée à ses microservices. Lorsque vous choisissez un modèle, vous devez évaluer les exigences relatives à l'isolation des données des locataires et à la structure des politiques d'autorisations vérifiées qui sont nécessaires pour une application SaaS de manière indépendante.

Pour appliquer une méthode cohérente de partage de l'identifiant du locataire dans l'ensemble de votre solution SaaS, il est recommandé de mapper l'identifiant à l'identité SaaS de l'utilisateur lors de son enregistrement, comme indiqué précédemment. Vous pouvez fournir ce mappage à une application SaaS en le gérant dans le cadre d'un IdP ou dans une source de données externe telle que DynamoDB. Nous vous recommandons également de mapper l'ID du magasin de politiques partagé aux utilisateurs. Bien que l'identifiant ne soit pas utilisé pour isoler les locataires, il s'agit d'une bonne pratique car elle facilite les changements futurs.

L'exemple suivant montre comment le point de terminaison de l'API envoie un JWT pour les utilisateurs `Alice` et `Bob` les utilisateurs qui appartiennent à des locataires différents mais partagent le magasin de politiques avec l'ID du magasin de politiques à des `store-multi-tenant` fins d'autorisation. Comme tous les locataires partagent un seul magasin de politiques, il n'est pas nécessaire de conserver l'ID du magasin de politiques dans un jeton ou une base de données. Étant donné que tous les locataires partagent un même identifiant de magasin de politiques, vous pouvez fournir cet identifiant en tant que variable d'environnement que votre application peut utiliser pour appeler le magasin de politiques.



L'exemple de politique suivant illustre le paradigme de conception d'une politique partagée par plusieurs locataires. Dans cette politique, le principal `MultiTenantApp::User` qui a le parent `MultiTenantApp::Role Admin` est autorisé à consulter les données de toutes les ressources.

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

Étant donné qu'un seul magasin de politiques est utilisé, le magasin de politiques Verified Permissions doit garantir qu'un attribut de location associé au principal correspond à l'attribut de location associé à la ressource. Cela peut être accompli en incluant la politique suivante dans le magasin de politiques, afin de garantir que toutes les demandes d'autorisation dont les attributs de location ne correspondent pas à la ressource et au principal sont rejetées.

```
forbid(
```

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

Pour une demande d'autorisation qui utilise un modèle de magasin de politiques partagé à locataires multiples, l'ID du magasin de politiques est l'identifiant du magasin de politiques partagé. Dans la demande suivante, l'accès `User` `Alice` est autorisé car elle possède un `Role` de`Admin`, et les `Tenant` attributs associés à la ressource et au principal sont les deux`TenantA`.

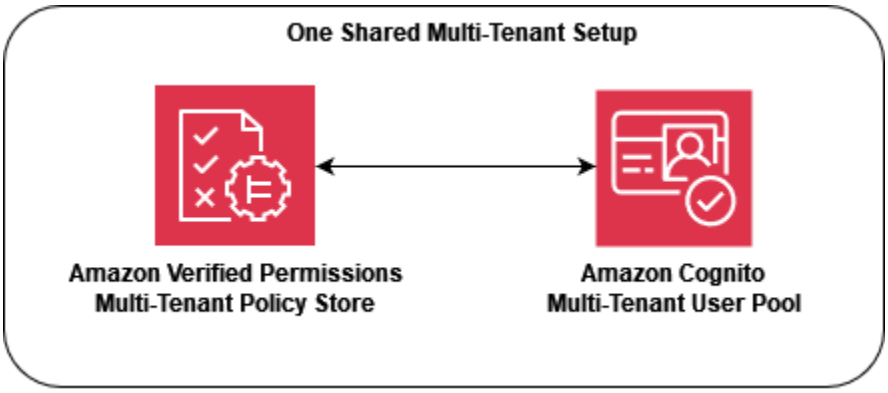
```
{
   "policyStoreId":"store-multi-tenant",
   "principal":{
      "entityType":"MultiTenantApp::User",
      "entityId":"Alice"
   },
   "action":{
      "actionType":"MultiTenantApp::Action",
      "actionId":"viewData"
   },
   "resource":{
      "entityType":"MultiTenantApp::Data",
      "entityId":"my_example_data"
   },
   "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":[]
         }
      ]
   }
}
```

Avec les autorisations vérifiées, il est possible, mais pas obligatoire, d'intégrer un IdP à un magasin de politiques. Cette intégration permet aux politiques de faire explicitement référence au principal dans le magasin d'identités en tant que principal des politiques. Pour plus d'informations sur l'intégration à Amazon Cognito en tant qu'IdP pour les autorisations vérifiées, consultez la documentation relative aux autorisations vérifiées et la documentation Amazon Cognito.

Lorsque vous intégrez un magasin de politiques à un IdP, vous ne pouvez utiliser qu'une seule source d'identité par magasin de politiques. Par exemple, si vous choisissez d'intégrer les autorisations vérifiées à Amazon Cognito, vous devez suivre la stratégie utilisée pour isoler les locataires des boutiques de politiques relatives aux autorisations vérifiées et des groupes

d'utilisateurs Amazon Cognito. Les magasins de politiques et les groupes d'utilisateurs doivent également se trouver dans le même emplacement Compte AWS.



Du point de vue de l'exploitation et de l'audit, le modèle de magasin de règles partagé par plusieurs locataires présente un inconvénient dans la mesure où l'activité enregistrée AWS CloudTrail nécessite des requêtes plus complexes pour filtrer l'activité individuelle du locataire, car chaque CloudTrail appel enregistré utilise le même magasin de politiques. Dans ce scénario, il est utile d'enregistrer des métriques personnalisées supplémentaires sur une dimension par locataire sur Amazon CloudWatch afin de garantir un niveau approprié d'observabilité et de capacité d'audit.

L'approche d'un magasin de politiques mutualisé partagé nécessite également une attention particulière aux quotas d'autorisations vérifiées afin de garantir qu'ils n'interfèrent pas avec le fonctionnement de votre solution SaaS. Nous vous recommandons notamment de surveiller les IsAuthorized demandes par seconde, par région et par quota de compte afin de vous assurer que ses limites ne sont pas dépassées. Vous pouvez demander une augmentation de ce quota.

Modèle de déploiement à plusieurs niveaux

En créant un modèle de déploiement à plusieurs niveaux, vous pouvez isoler les locataires « niveau entreprise » hautement prioritaires du volume potentiellement plus élevé de clients « niveau standard ». Dans ce modèle, vous pouvez déployer les modifications apportées aux politiques dans les magasins de politiques séparément pour chaque niveau, ce qui permet d'isoler chaque niveau de clients des modifications apportées en dehors de son niveau. Dans le modèle de déploiement hiérarchisé, les magasins de politiques sont généralement créés dans le cadre du provisionnement initial de l'infrastructure pour chaque niveau au lieu d'être déployés lors de l'intégration d'un locataire.

Si votre solution utilise principalement un modèle d'isolation groupée, vous aurez peut-être besoin d'une isolation ou d'une personnalisation supplémentaires. Par exemple, vous pouvez créer un

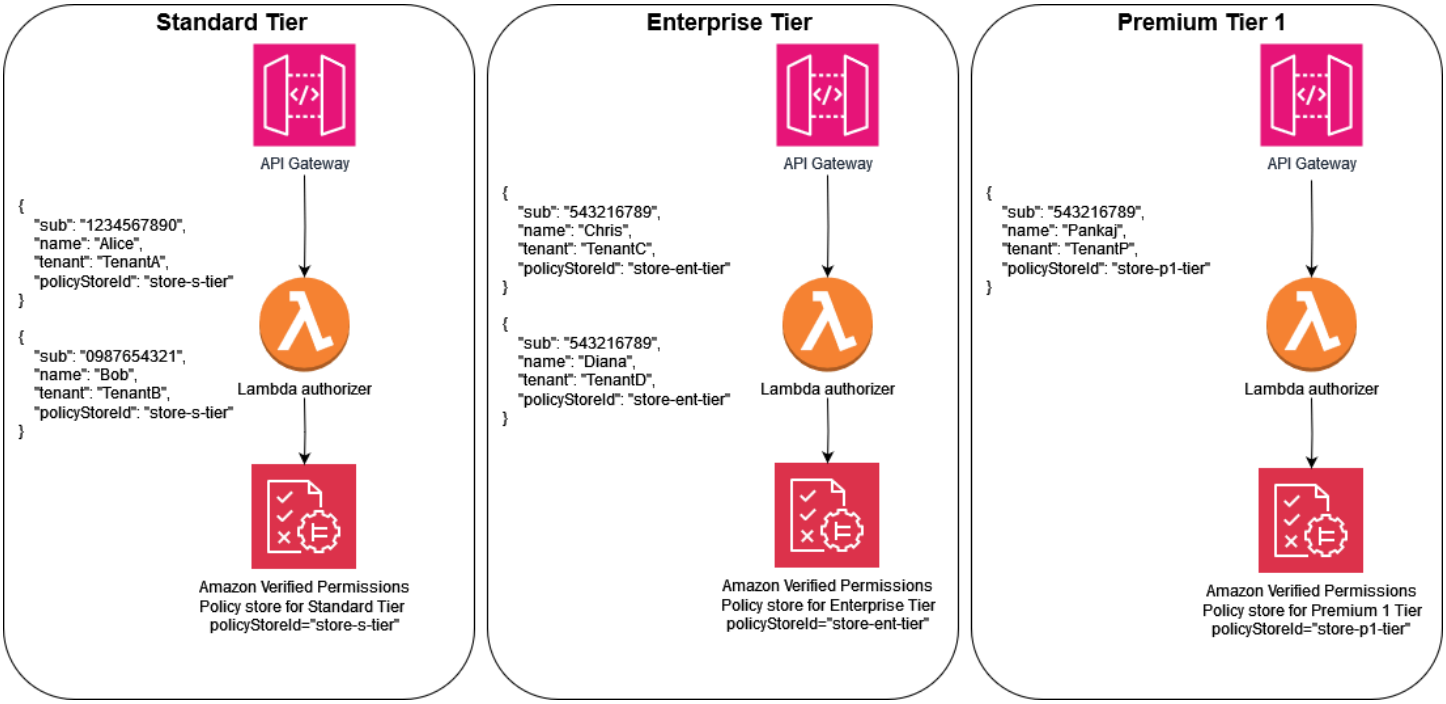
« niveau Premium » dans lequel chaque locataire disposerait de sa propre infrastructure de niveau locataire, ce qui crée un modèle cloisonné en déployant une instance groupée avec un seul locataire. Cela pourrait prendre la forme d'infrastructures « Premium Tier A » et « Premium Tier Tenant B » complètement séparées, y compris des magasins d'assurance. Cette approche aboutit à un modèle d'isolation cloisonné pour le plus haut niveau de clients.

Dans le modèle de déploiement hiérarchisé, chaque magasin de politiques doit suivre le même modèle d'isolation, bien qu'il soit déployé séparément. Étant donné que plusieurs magasins de politiques sont utilisés, vous devez appliquer une méthode cohérente de partage de l'identifiant du magasin de politiques associé au locataire dans l'ensemble de la solution SaaS. Comme dans le cas du modèle de magasin de politiques par locataire, il est recommandé de mapper l'identifiant du locataire à l'identité SaaS de l'utilisateur lors de son inscription.

Le schéma suivant montre trois niveaux : `Standard Tier`Enterprise Tier, et`Premium Tier` 1. Chaque niveau est déployé séparément dans sa propre infrastructure et utilise un magasin de politiques partagé au sein du niveau. Les niveaux Standard et Enterprise contiennent plusieurs locataires. `TenantA`et `TenantB` se situent dans `Standard Tier` le niveau `TenantC` et `TenantD` sont dans le niveau Enterprise.

`Premium Tier 1`contient uniquement`TenantP`, de sorte que vous pouvez servir le locataire premium comme si la solution reposait sur un modèle d'isolation totalement cloisonné et fournir des fonctionnalités telles que des politiques personnalisées. L'intégration d'un nouveau client haut de gamme entraînerait la création d'une `Premium Tier 2` infrastructure.

> **Note**
>
> L'application, le déploiement et l'intégration des locataires dans le niveau premium sont identiques aux niveaux standard et entreprise. La seule différence est que le flux de travail d'intégration de niveau supérieur commence par le provisionnement d'une nouvelle infrastructure de niveau.


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"

Considérations relatives à la conception multi-locataires OPA

L'Open Policy Agent (OPA) est un service flexible qui peut être appliqué à de nombreux cas d'utilisation dans lesquels les applications doivent prendre des décisions en matière de politique et d'autorisation. L'utilisation de l'OPA avec des applications SaaS multi-locataires nécessite la prise en compte de critères uniques pour garantir que les meilleures pratiques SaaS clés, telles que l'isolation des locataires, continuent de faire partie de la mise en œuvre de l'OPA. Ces critères incluent les modèles de déploiement de l'OPA, l'isolement des locataires et le modèle de document de l'OPA, ainsi que l'intégration des locataires. Chacun de ces éléments affecte la conception optimale de l'OPA en ce qui concerne les applications multi-locataires.

Bien que la discussion dans cette section se concentre sur l'OPA, les concepts généraux sont ancrés dans l'état d'esprit d'isolement et les conseils qu'il fournit. Les applications SaaS doivent toujours prendre en compte l'isolation des locataires dans le cadre de leur conception, et ce principe général d'isolation s'étend à l'inclusion de l'OPA dans une application SaaS. L'OPA, s'il est utilisé de manière appropriée, peut jouer un rôle clé dans la mise en œuvre de l'isolation dans les applications SaaS. Cette section fait également référence aux principaux modèles d'isolation SaaS tels que le modèle SaaS en silo et le modèle SaaS mutualisé. Pour plus d'informations, consultez les principaux concepts d'isolation dans le AWS Well-Architected Framework, SaaS Lens.

Considérations relatives à la conception :

- Comparaison des modèles de déploiement centralisés et distribués
- Isolation des locataires grâce au modèle documentaire OPA
- Accueil des locataires

Comparaison des modèles de déploiement centralisés et distribués

Vous pouvez déployer OPA selon un schéma de déploiement centralisé ou distribué, et la méthode idéale pour une application mutualisée dépend du cas d'utilisation. Pour des exemples de ces modèles, reportez-vous aux APIs sections Utilisation d'un PDP centralisé avec un PDP distribué PEPs activé APIs et Utilisation d'un PDP distribué plus haut dans ce guide. PEPs Comme l'OPA peut être déployé en tant que démon dans un système d'exploitation ou un conteneur, il peut être implémenté de plusieurs manières pour prendre en charge une application mutualisée.

Dans un schéma de déploiement centralisé, OPA est déployé en tant que conteneur ou daemon avec son RESTful API disponible pour les autres services de l'application. Lorsqu'un service nécessite une décision de l'OPA, l' RESTful API centrale de l'OPA est appelée pour produire cette décision. Cette approche est simple à déployer et à maintenir, car il n'y a qu'un seul déploiement d'OPA. L'inconvénient de cette approche est qu'elle ne fournit aucun mécanisme permettant de maintenir la séparation des données des locataires. Comme il n'existe qu'un seul déploiement de l'OPA, toutes les données des locataires utilisées dans une décision de l'OPA, y compris les données externes référencées par l'OPA, doivent être mises à la disposition de l'OPA. Vous pouvez maintenir l'isolation des données des locataires grâce à cette approche, mais elle doit être appliquée par la politique et la structure documentaire de l'OPA ou par l'accès à des données externes. Un modèle de déploiement centralisé nécessite également une latence plus élevée, car chaque décision d'autorisation doit passer par un appel d' RESTful API vers un autre service.

Dans un modèle de déploiement distribué, l'OPA est déployé en tant que conteneur ou daemon aux côtés des services de l'application mutualisée. Il peut être déployé en tant que conteneur annexe ou en tant que démon exécuté localement sur le système d'exploitation. Pour récupérer une décision de l'OPA, le service envoie un appel d' RESTful API au déploiement OPA local. (Comme l'OPA peut être déployée sous forme de package Go, vous pouvez utiliser Go de manière native pour récupérer une décision au lieu d'utiliser un appel d' RESTful API.) Contrairement au modèle de déploiement centralisé, le modèle distribué nécessite un effort beaucoup plus important pour le déploiement, la maintenance et la mise à jour, car il est présent dans plusieurs domaines de l'application. L'un des avantages du modèle de déploiement distribué est la capacité à maintenir l'isolation des données des locataires, en particulier pour les applications qui utilisent un modèle SaaS cloisonné. Les données spécifiques au locataire peuvent être isolées dans les déploiements OPA spécifiques à ce locataire, car dans un modèle distribué, l'OPA est déployée en même temps que le locataire. En outre, un modèle de déploiement distribué présente une latence bien inférieure à celle d'un modèle de déploiement centralisé, car chaque décision d'autorisation peut être prise localement.

Lorsque vous choisissez un modèle de déploiement OPA dans votre application mutualisée, assurez-vous d'évaluer les critères les plus critiques pour votre application. Si votre application mutualisée est sensible à la latence, un modèle de déploiement distribué offre de meilleures performances au détriment d'un déploiement et d'une maintenance plus complexes. Bien que vous puissiez gérer une partie de cette complexité par le biais DevOps de l'automatisation, cela nécessite tout de même des efforts supplémentaires par rapport à un modèle de déploiement centralisé.

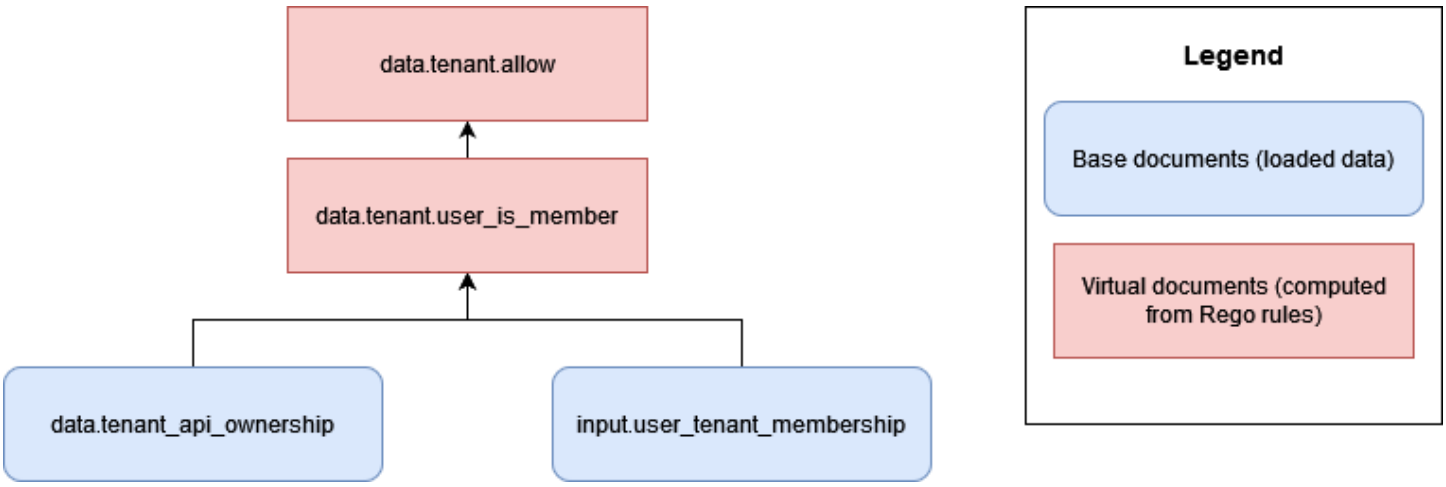
Si votre application mutualisée utilise un modèle SaaS cloisonné, vous pouvez utiliser un modèle de déploiement OPA distribué pour imiter l'approche cloisonnée de l'isolation des données des

locataires. En effet, lorsque OPA s'exécute parallèlement à chaque service d'application spécifique au locataire, vous pouvez personnaliser chaque déploiement OPA pour ne contenir que les données associées à ce locataire. Il n'est pas possible de cloisonner les données OPA dans un schéma de déploiement OPA centralisé. Si vous utilisez un modèle de déploiement centralisé ou un modèle distribué en conjonction avec un modèle SaaS groupé, l'isolation des données des locataires doit être maintenue dans le modèle de document OPA.

Isolation des locataires grâce au modèle documentaire OPA

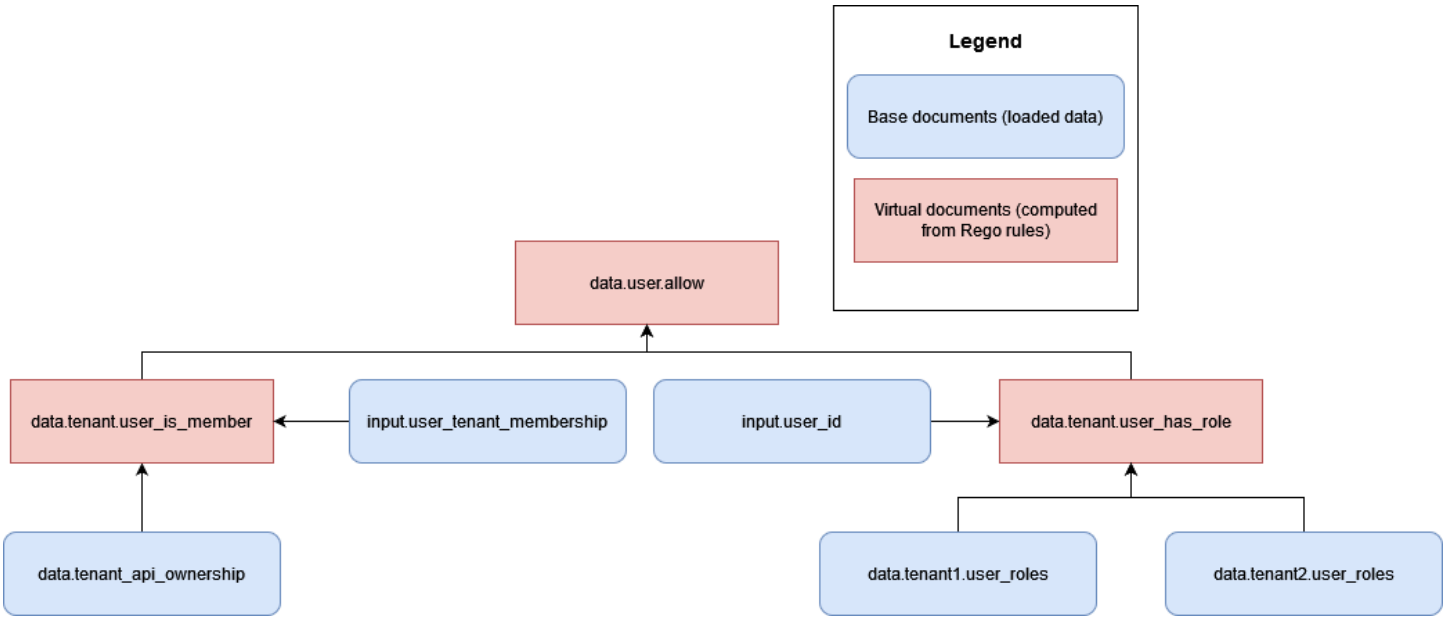
L'OPA utilise des documents pour prendre des décisions. Ces documents peuvent contenir des données spécifiques au locataire. Vous devez donc réfléchir à la manière de maintenir l'isolation des données du locataire. Les documents OPA se composent de documents de base et de documents virtuels. Les documents de base contiennent des données provenant du monde extérieur. Cela inclut les données fournies directement à l'OPA, les données relatives à la demande de l'OPA et les données susceptibles d'être transmises à l'OPA en tant qu'entrée. Les documents virtuels sont calculés par politique et incluent les politiques et règles de l'OPA. Pour plus d'informations, consultez la documentation de l'OPA.

Pour concevoir un modèle de document dans OPA pour une application multi-locataires, vous devez d'abord déterminer le type de documents de base dont vous aurez besoin pour prendre une décision dans OPA. Si ces documents de base contiennent des données spécifiques au locataire, vous devez prendre des mesures pour vous assurer que ces données ne sont pas accidentellement exposées à un accès entre locataires. Heureusement, dans de nombreux cas, les données spécifiques au locataire ne sont pas nécessaires pour prendre une décision dans OPA. L'exemple suivant montre un modèle de document OPA hypothétique qui autorise l'accès à une API en fonction du locataire propriétaire de l'API et du fait que l'utilisateur est membre du locataire, comme indiqué dans le document d'entrée.



Dans cette approche, l'OPA n'a accès à aucune donnée spécifique au locataire, à l'exception des informations sur les locataires qui possèdent une API. Dans ce cas, le fait que l'OPA facilite l'accès entre locataires ne suscite aucune inquiétude, car les seules informations que l'OPA utilise pour prendre une décision d'accès sont l'association d'un utilisateur avec un locataire et l'association du locataire avec APIs. Vous pouvez appliquer ce type de modèle de document OPA à un modèle SaaS cloisonné, car chaque locataire serait propriétaire de ressources indépendantes.

Cependant, dans de nombreuses approches d'autorisation RBAC, il existe un risque d'exposition d'informations entre locataires. Dans l'exemple suivant, un modèle de document OPA hypothétique autorise l'accès à une API selon que l'utilisateur est membre d'un locataire et qu'il possède le rôle approprié pour accéder à l'API.



Ce modèle présente un risque d'accès entre locataires, car les rôles et autorisations de plusieurs locataires `data.tenant2.user_roles` doivent désormais être rendus accessibles à `data.tenant1.user_roles` l'OPA pour prendre des décisions en matière d'autorisation. Pour préserver l'isolement des locataires et la confidentialité du mappage des rôles, ces données ne doivent pas résider dans l'OPA. Les données RBAC doivent résider dans une source de données externe telle qu'une base de données. En outre, l'OPA ne doit pas être utilisée pour associer des rôles prédéfinis à des autorisations spécifiques, car cela rend difficile pour les locataires de définir leurs propres rôles et autorisations. Cela rend également votre logique d'autorisation rigide et nécessite une mise à jour constante. Pour obtenir des conseils sur la manière d'intégrer en toute sécurité les données RBAC dans le processus décisionnel de l'OPA, consultez la section Recommandations pour l'isolation des locataires et la confidentialité des données plus loin dans ce guide.

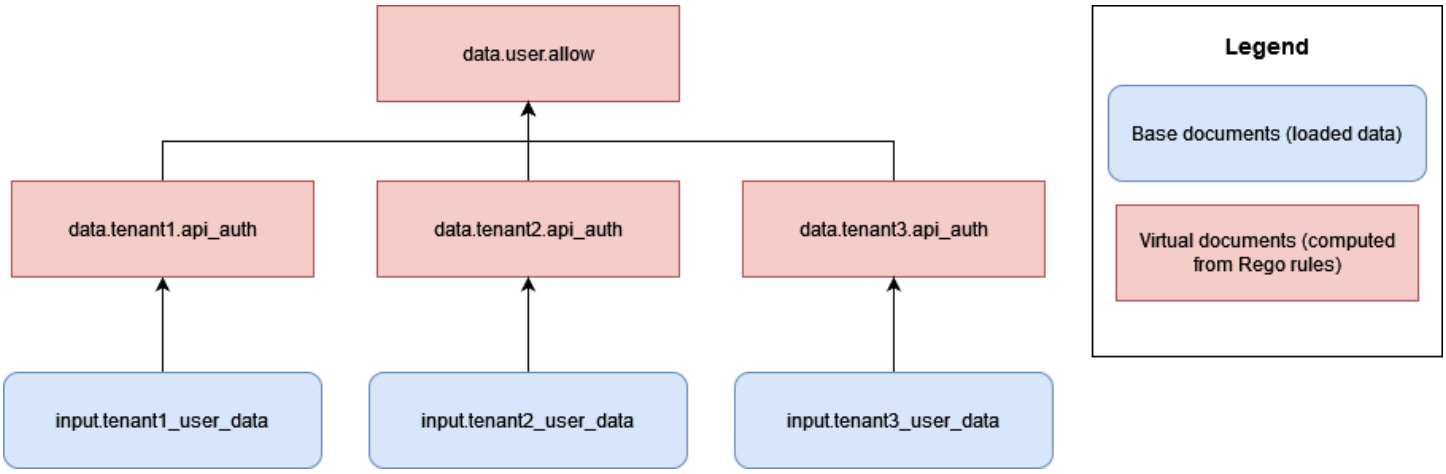
Vous pouvez facilement maintenir l'isolement des locataires dans OPA en ne stockant aucune donnée spécifique au locataire sous forme de document de base asynchrone. Un document de base asynchrone est une donnée qui est stockée en mémoire et qui peut être mise à jour périodiquement, dans OPA. D'autres documents de base, tels que les entrées OPA, sont transmis de manière synchrone et ne sont disponibles qu'au moment de la décision. Par exemple, la fourniture de données spécifiques au locataire dans le cadre de la saisie par l'OPA d'une requête ne constituerait pas une violation de l'isolement du locataire, car ces données ne sont disponibles que de manière synchrone pendant le processus de prise de décision.

Accueil des locataires

La structure des documents de l'OPA doit permettre une intégration simple des locataires sans introduire de lourdes exigences. Vous pouvez organiser les documents virtuels dans la hiérarchie des modèles de documents OPA à l'aide de packages, qui peuvent contenir de nombreuses règles. Lorsque vous planifiez un modèle de document OPA pour une application à locataires multiples, déterminez d'abord quelles données sont nécessaires à l'OPA pour prendre une décision. Vous pouvez fournir des données en entrée, les précharger dans OPA ou les fournir à partir de sources de données externes au moment de la prise de décision ou périodiquement. Pour plus d'informations sur l'utilisation de données externes avec OPA, consultez la section Extraction de données externes pour un PDP dans OPA plus loin dans ce guide.

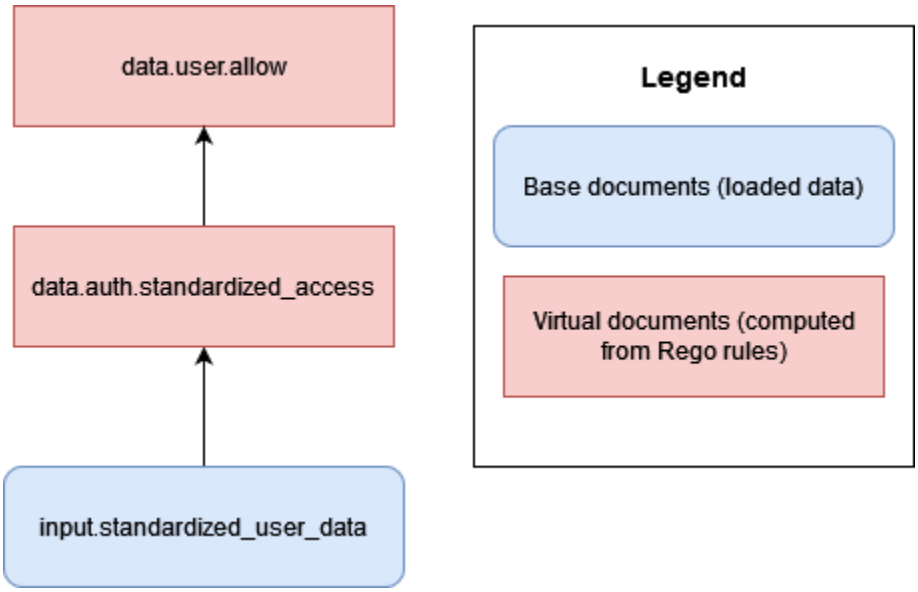
Après avoir déterminé les données requises pour prendre une décision dans OPA, réfléchissez à la manière de mettre en œuvre les règles OPA organisées sous forme de packages, afin de prendre des décisions avec ces données. Par exemple, dans un modèle SaaS cloisonné où chaque locataire

peut avoir des exigences uniques quant à la manière dont les décisions d'autorisation sont prises, vous pouvez implémenter des packages de règles OPA spécifiques au locataire.



L'inconvénient de cette approche est que vous devez ajouter un nouvel ensemble de règles OPA, spécifiques à chaque locataire, pour chaque locataire que vous ajoutez à votre application SaaS. Cela est fastidieux et difficile à adapter, mais cela peut être inévitable en fonction des besoins de vos locataires.

Par ailleurs, dans un modèle SaaS mutualisé, si tous les locataires prennent des décisions d'autorisation basées sur les mêmes règles et utilisent la même structure de données, vous pouvez utiliser des packages OPA standard dotés de règles généralement applicables pour faciliter l'intégration des locataires et l'extension de votre mise en œuvre de l'OPA.



Dans la mesure du possible, nous vous recommandons d'utiliser des règles et des packages OPA généralisés (ou des documents virtuels) pour prendre des décisions basées sur des données

standardisées fournies par chaque locataire. Cette approche rend l'OPA facilement évolutive, car vous ne modifiez que les données fournies à l'OPA pour chaque locataire, et non la façon dont l'OPA prend ses décisions par le biais de ses règles. Il n'est nécessaire d'introduire un rules-per-tenant modèle que lorsque les locataires individuels ont besoin de décisions uniques ou doivent fournir à l'OPA des données différentes de celles des autres locataires.

DevOps, surveillance, journalisation et récupération de données pour un PDP

Dans ce paradigme d'autorisation proposé, les politiques sont centralisées dans le service d'autorisation. Cette centralisation est délibérée car l'un des objectifs des modèles de conception présentés dans ce guide est de découpler les politiques, c'est-à-dire de supprimer la logique d'autorisation des autres composants de l'application. Amazon Verified Permissions et Open Policy Agent (OPA) fournissent tous deux des mécanismes permettant de mettre à jour les politiques lorsque des modifications de la logique d'autorisation sont nécessaires.

Dans le cas des autorisations vérifiées, des mécanismes de mise à jour programmatique des politiques sont proposés par le AWS SDK (voir le guide de référence de l'API Amazon Verified Permissions). À l'aide du SDK, vous pouvez imposer de nouvelles politiques à la demande. En outre, comme Verified Permissions est un service géré, vous n'avez pas besoin de gérer, de configurer ou de maintenir des plans de contrôle ou des agents pour effectuer des mises à jour. Toutefois, nous vous recommandons d'utiliser un pipeline d'intégration et de déploiement continus (CI/CD) pour administrer le déploiement des magasins de politiques d'autorisations vérifiées et des mises à jour des politiques à l'aide du AWS SDK.

Les autorisations vérifiées permettent d'accéder facilement aux fonctionnalités d'observabilité. Il peut être configuré pour enregistrer toutes les tentatives d'accès aux groupes de CloudWatch journaux Amazon AWS CloudTrail, aux compartiments Amazon Simple Storage Service (Amazon S3) ou aux flux de livraison Amazon Data Firehose afin de permettre une réponse rapide aux incidents de sécurité et aux demandes d'audit. En outre, vous pouvez surveiller l'état du service Verified Permissions via le Tableau de bord AWS Health. Dans la mesure où Verified Permissions est un service géré, son intégrité est maintenue par AWS d'autres services gérés et vous pouvez configurer les fonctionnalités d'observabilité à l'aide d'autres services AWS gérés.

Dans le cas de l'OPA, REST APIs propose des moyens de mettre à jour les politiques de manière programmatique. Vous pouvez configurer le APIs pour extraire les nouvelles versions des ensembles de politiques à partir d'emplacements établis ou pour appliquer des politiques à la demande. En outre, OPA propose un service de découverte de base dans lequel les nouveaux agents peuvent être configurés de manière dynamique et gérés de manière centralisée par un plan de contrôle qui distribue les ensembles de découverte. (Le plan de contrôle de l'OPA doit être établi et configuré par l'opérateur OPA.) Nous vous recommandons de créer un CI/CD pipeline robuste pour le

versionnement, la vérification et la mise à jour des politiques, que le moteur de politiques soit Verified Permissions, OPA ou une autre solution.

Pour OPA, le plan de contrôle fournit également des options de surveillance et d'audit. Vous pouvez exporter les journaux contenant les décisions d'autorisation de l'OPA vers des serveurs HTTP distants pour l'agrégation des journaux. Ces journaux de décisions sont d'une valeur inestimable à des fins d'audit.

Si vous envisagez d'adopter un modèle d'autorisation dans lequel les décisions relatives au contrôle d'accès sont dissociées de votre application, assurez-vous que votre service d'autorisation dispose de fonctionnalités de surveillance, de journalisation et de CI/CD gestion efficaces pour intégrer de nouvelles politiques PDPs ou les mettre à jour.

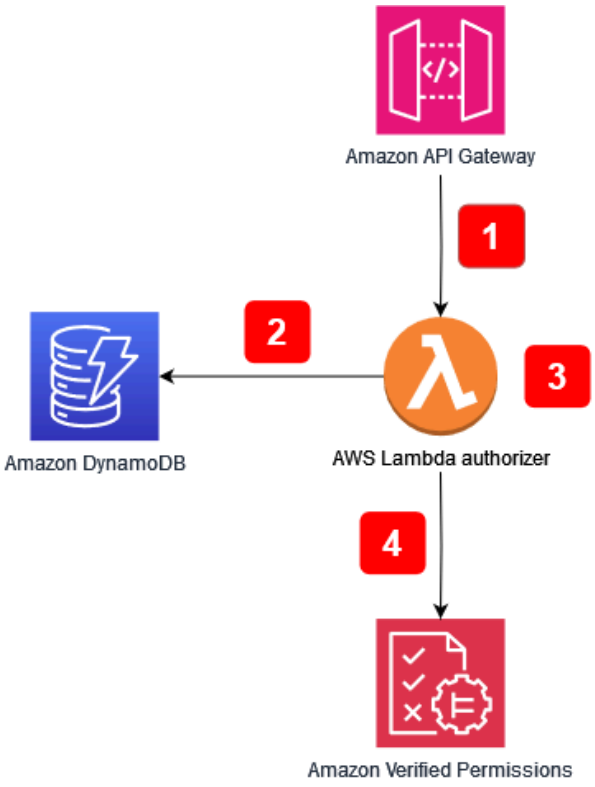
Rubriques

Extraction de données externes pour un PDP dans Amazon Verified Permissions

Amazon Verified Permissions ne prend pas en charge la récupération de données externes pour un PDP, mais il peut stocker les données fournies par l'utilisateur dans le cadre de son schéma. Comme dans OPA, si toutes les données relatives à une décision d'autorisation peuvent être fournies dans le cadre d'une demande d'autorisation ou dans le cadre d'un jeton Web JSON (JWT) transmis dans le cadre de la demande, aucune configuration supplémentaire n'est requise. Cependant, vous pouvez fournir des données supplémentaires provenant de sources externes à Verified Permissions par le biais de la demande d'autorisation dans le cadre du service d'autorisation d'une application qui appelle Verified Permissions. Par exemple, le service d'autorisation d'une application peut interroger une source externe telle que DynamoDB ou Amazon RDS pour obtenir des données, et ces services peuvent ensuite inclure les données fournies en externe dans le cadre d'une demande d'autorisation.

Le schéma suivant montre un exemple de la manière dont des données supplémentaires peuvent être récupérées et incorporées dans une demande d'autorisation d'autorisations vérifiées. Il peut être nécessaire d'utiliser cette méthode pour récupérer des données telles que les mappages de rôles RBAC, pour récupérer des attributs supplémentaires pertinents pour les ressources ou les principaux,

ou dans les cas où les données résident dans différentes parties d'une application et ne peuvent pas être fournies via un jeton de fournisseur d'identité (IdP).



Flux d'applications :

1. L'application reçoit un appel d'API vers Amazon API Gateway et transmet l'appel à l' AWS Lambda autorisateur.
2. L'autorisateur Lambda appelle Amazon DynamoDB pour récupérer des données supplémentaires concernant le principal à l'origine de la demande.
3. L'autorisateur Lambda intègre les données supplémentaires dans la demande d'autorisation envoyée à Verified Permissions.
4. L'autorisateur Lambda fait une demande d'autorisation à Verified Permissions et reçoit une décision d'autorisation.

Le schéma inclut une fonctionnalité d'Amazon API Gateway appelée autorisateur Lambda. Bien que cette fonctionnalité ne soit pas disponible pour APIs ceux fournis par d'autres services ou applications, vous pouvez reproduire le modèle général qui consiste à utiliser un autorisateur pour récupérer des données supplémentaires à intégrer dans une demande d'autorisation d'autorisations vérifiées dans une multitude de cas d'utilisation.

Récupération de données externes pour un PDP dans OPA

Pour OPA, si toutes les données requises pour une décision d'autorisation peuvent être fournies en entrée ou dans le cadre d'un jeton Web JSON (JWT) transmis en tant que composant de la requête, aucune configuration supplémentaire n'est requise. (Il est relativement simple de transmettre JWTs des données contextuelles SaaS à OPA dans le cadre de la saisie des requêtes.) L'OPA peut accepter une entrée JSON arbitraire dans le cadre de ce que l'on appelle l'approche d'entrée par surcharge. Si un PDP nécessite des données autres que celles qui peuvent être incluses en entrée ou en tant que JWT, OPA propose plusieurs options pour récupérer ces données. Il s'agit notamment du regroupement, du transfert de données (réplication) et de la récupération dynamique des données.

Regroupement OPA

La fonctionnalité de regroupement OPA prend en charge le processus suivant pour la récupération de données externes :

1. Le point d'application des politiques (PEP) demande une décision d'autorisation.
2. L'OPA télécharge de nouveaux ensembles de politiques, y compris des données externes.
3. Le service de regroupement réplique les données à partir de sources de données.

Lorsque vous utilisez la fonctionnalité de regroupement, OPA télécharge régulièrement des politiques et des ensembles de données à partir d'un service de regroupement centralisé. (OPA ne fournit pas la mise en œuvre et la configuration d'un service groupé.) Toutes les politiques et données externes extraites du service groupé sont stockées en mémoire. Cette option ne fonctionnera pas si la taille des données externes est trop importante pour être stockée en mémoire ou si les données changent trop fréquemment.

Pour plus d'informations sur la fonctionnalité de regroupement, consultez la documentation OPA.

Réplication OPA (transmission de données)

L'approche de réplication OPA prend en charge le processus suivant pour la récupération de données externes :

1. Le PEP demande une décision d'autorisation.
2. Le réplicateur de données transmet les données à l'OPA.

3. Le réplicateur de données réplique les données à partir de sources de données.

Dans cette alternative à l'approche de regroupement, les données sont transmises à l'OPA au lieu d'être périodiquement extraites par celle-ci. (OPA ne fournit pas la mise en œuvre et la configuration d'un réplicateur.) L'approche push présente les mêmes limites de taille de données que l'approche de regroupement, car OPA stocke toutes les données en mémoire. Le principal avantage de l'option push est que vous pouvez mettre à jour les données dans OPA avec des deltas au lieu de remplacer toutes les données externes à chaque fois. Cela rend l'option push plus appropriée pour les ensembles de données qui changent fréquemment.

Pour plus d'informations sur l'option de réplication, consultez la documentation OPA.

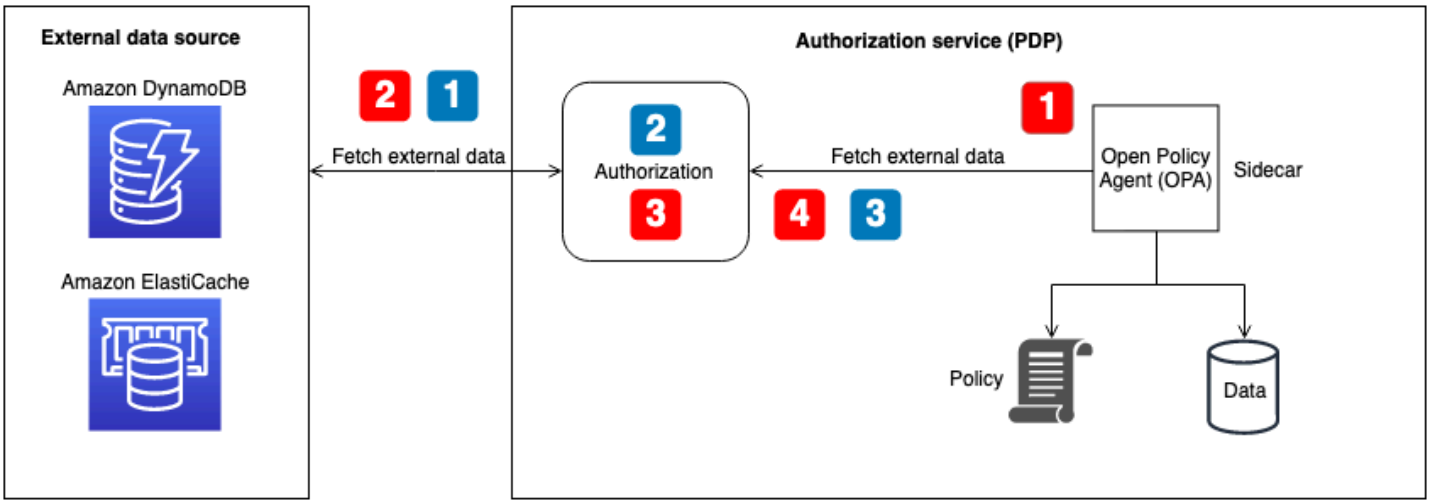
Récupération dynamique des données OPA

Si les données externes à récupérer sont trop volumineuses pour être mises en cache dans la mémoire de l'OPA, les données peuvent être extraites dynamiquement d'une source externe lors de l'évaluation d'une décision d'autorisation. Lorsque vous utilisez cette approche, les données sont toujours à jour. Cette approche présente deux inconvénients : la latence du réseau et l'accessibilité. Actuellement, OPA ne peut récupérer des données au moment de l'exécution que par le biais d'une requête HTTP. Si les appels destinés à une source de données externe ne peuvent pas renvoyer de données sous forme de réponse HTTP, ils ont besoin d'une API personnalisée ou d'un autre mécanisme pour fournir ces données à l'OPA. Étant donné qu'OPA ne peut récupérer des données que par le biais de requêtes HTTP et que la vitesse de récupération des données est essentielle, nous vous recommandons d'utiliser un outil Service AWS tel qu'Amazon DynamoDB pour conserver des données externes dans la mesure du possible.

Pour plus d'informations sur l'approche pull, consultez la documentation de l'OPA.

Utilisation d'un service d'autorisation pour la mise en œuvre avec OPA

Lorsque vous récupérez des données externes en utilisant le regroupement, la réplication ou une approche d'extraction dynamique, nous recommandons que le service d'autorisation facilite cette interaction. En effet, le service d'autorisation peut récupérer des données externes et les transformer en JSON pour que l'OPA prenne des décisions d'autorisation. Le schéma suivant montre comment un service d'autorisation peut fonctionner avec ces trois approches de récupération de données externes.



Extraction de données externes pour le flux OPA : extraction groupée ou dynamique de données au moment de la décision (illustrée par des légendes numérotées en rouge dans le schéma) :

1. L'OPA appelle le point de terminaison de l'API local pour le service d'autorisation, qui est configuré comme point de terminaison du bundle ou comme point de terminaison pour la récupération dynamique des données lors des décisions d'autorisation.
2. Le service d'autorisation interroge ou appelle la source de données externe pour récupérer des données externes. (Pour un point de terminaison groupé, ces données doivent également contenir des politiques et des règles OPA. Les mises à jour du bundle remplacent tout, à la fois les données et les politiques, dans le cache d'OPA.)
3. Le service d'autorisation effectue toute transformation nécessaire sur les données renvoyées pour les transformer en entrée JSON attendue.
4. Les données sont renvoyées à l'OPA. Il est mis en cache en mémoire pour la configuration du bundle et utilisé immédiatement pour les décisions d'autorisation dynamiques.

Récupération de données externes pour le flux OPA — réplicateur (illustrées par des légendes numérotées en bleu dans le schéma) :

1. Le réplicateur (qui fait partie du service d'autorisation) appelle la source de données externe et récupère toutes les données à mettre à jour dans OPA. Cela peut inclure des politiques, des règles et des données externes. Cet appel peut être effectué à une cadence définie ou en réponse à des mises à jour de données dans la source externe.
2. Le service d'autorisation effectue toutes les transformations nécessaires sur les données renvoyées pour les transformer en entrée JSON attendue.

3. Le service d'autorisation appelle OPA et met les données en cache en mémoire. Le service d'autorisation peut mettre à jour les données, les politiques et les règles de manière sélective.

Recommandations relatives à l'isolation des locataires et à la confidentialité des données

La section précédente présentait plusieurs approches pour utiliser des données externes avec OPA et Amazon Verified Permissions afin de faciliter la prise de décisions en matière d'autorisation. Dans la mesure du possible, nous vous recommandons d'utiliser l'approche de saisie par surcharge pour transmettre les données contextuelles SaaS à l'OPA afin de prendre des décisions d'autorisation au lieu de stocker les données dans la mémoire de l'OPA. Ce cas d'utilisation ne s'applique pas à AWS Cloud Map, car il ne prend pas en charge le stockage de données externes dans le service.

Dans les modèles hybrides de contrôle d'accès basé sur les rôles (RBAC) ou de contrôle d'accès basé sur les attributs (ABAC), les données fournies uniquement par une demande ou une requête d'autorisation peuvent être insuffisantes, car les rôles et les autorisations doivent être référencés pour prendre des décisions d'autorisation. Pour préserver l'isolement des locataires et la confidentialité du mappage des rôles, ces données ne doivent pas résider dans l'OPA. Les données RBAC doivent résider dans une source de données externe telle qu'une base de données ou doivent être transmises dans le cadre de réclamations d'un IdP dans un JWT. Dans Verified Permissions, les données RBAC peuvent être conservées dans le cadre des politiques et du schéma du modèle de magasin de politiques par locataire, car chaque locataire possède son propre magasin de politiques séparé logiquement. Toutefois, dans le modèle d'un magasin de politiques mutualisé partagé, les données de mappage des rôles ne doivent pas résider dans les autorisations vérifiées afin de maintenir l'isolement des locataires.

En outre, l'OPA et les autorisations vérifiées ne doivent pas être utilisées pour associer des rôles prédéfinis à des autorisations spécifiques, car cela rend difficile pour les locataires de définir leurs propres rôles et autorisations. Cela rend également votre logique d'autorisation rigide et nécessite une mise à jour constante. L'exception à cette directive est le modèle de magasin de politiques par locataire dans Verified Permissions, car ce modèle permet à chaque locataire d'avoir ses propres politiques qui peuvent être évaluées indépendamment par locataire.

Amazon Verified Permissions

Le seul endroit où les autorisations vérifiées peuvent stocker des données RBAC potentiellement privées se trouve dans le schéma. Cela est acceptable dans le modèle de magasin de politiques

par locataire, car chaque locataire possède son propre magasin de politiques séparé logiquement. Cependant, cela pourrait compromettre l'isolement des locataires dans le modèle de magasin de politiques partagé par plusieurs locataires. Dans les cas où ces données sont nécessaires pour prendre une décision d'autorisation, elles doivent être extraites d'une source externe telle que DynamoDB ou Amazon RDS et intégrées à la demande d'autorisation des autorisations vérifiées.

OPA

Les approches sécurisées avec OPA pour préserver la confidentialité et isoler les locataires des données RBAC incluent l'utilisation de la récupération ou de la réplication dynamiques des données pour obtenir des données externes. En effet, vous pouvez utiliser le service d'autorisation illustré dans le schéma précédent pour fournir uniquement des données externes spécifiques au locataire ou à l'utilisateur afin de prendre une décision d'autorisation. Par exemple, vous pouvez utiliser un réplicateur pour fournir des données RBAC ou une matrice d'autorisations au cache OPA lorsqu'un utilisateur se connecte, et faire en sorte que les données soient référencées en fonction de l'utilisateur indiqué dans les données d'entrée. Vous pouvez utiliser une approche similaire avec des données extraites dynamiquement pour récupérer uniquement les données pertinentes pour prendre des décisions d'autorisation. De plus, dans l'approche de récupération dynamique des données, ces données n'ont pas besoin d'être mises en cache dans OPA. L'approche de regroupement n'est pas aussi efficace que l'approche de récupération dynamique pour maintenir l'isolement des locataires, car elle met à jour tout ce qui se trouve dans le cache OPA et ne permet pas de traiter des mises à jour précises. Le modèle de regroupement reste une bonne approche pour mettre à jour les politiques OPA et les données non RBAC.

Bonnes pratiques

Cette section répertorie certains des principaux points à retenir de ce guide. Pour des discussions détaillées sur chaque point, suivez les liens vers les sections correspondantes.

Sélectionnez un modèle de contrôle d'accès adapté à votre application

Ce guide décrit plusieurs modèles de contrôle d'accès. En fonction de votre application et des exigences commerciales, vous devez sélectionner le modèle qui vous convient. Réfléchissez à la manière dont vous pouvez utiliser ces modèles pour répondre à vos besoins en matière de contrôle d'accès et à la manière dont vos besoins de contrôle d'accès pourraient évoluer, ce qui nécessiterait de modifier l'approche que vous avez choisie.

Implémenter un PDP

Le point de décision politique (PDP) peut être caractérisé comme un moteur de politiques ou de règles. Ce composant est chargé d'appliquer des politiques ou des règles et de décider si un accès particulier est autorisé. Un PDP permet de transférer la logique d'autorisation du code de l'application vers un système distinct. Cela peut simplifier le code de l'application. Il fournit également une interface easy-to-use idempotente pour prendre des décisions d'autorisation pour les microservices APIs, les couches Backend for Frontend (BFF) ou tout autre composant d'application. Un PDP peut être utilisé pour appliquer les exigences de location de manière cohérente dans l'ensemble d'une application.

PEPs Implémentation pour chaque API de votre application

La mise en œuvre d'un point d'application des politiques (PEP) nécessite de déterminer où le contrôle d'accès doit être appliqué dans une application. Dans un premier temps, localisez les points de votre application où vous pouvez intégrer PEPs. Tenez compte de ce principe lorsque vous décidez où ajouter PEPs :

Si une application expose une API, cette API doit être dotée d'une autorisation et d'un contrôle d'accès.

Envisagez d'utiliser Amazon Verified Permissions ou OPA comme moteur de politique pour votre PDP

Amazon Verified Permissions présente des avantages par rapport aux moteurs de politiques personnalisés. Il s'agit d'un service de gestion des autorisations et d'autorisation évolutif et précis pour les applications que vous créez. Il prend en charge la rédaction de politiques dans le langage open source déclaratif de haut niveau Cedar. Par conséquent, la mise en œuvre d'un moteur de politiques à l'aide d'autorisations vérifiées nécessite moins d'efforts de développement que la mise en œuvre de votre propre solution. En outre, les autorisations vérifiées sont entièrement gérées, vous n'avez donc pas à gérer l'infrastructure sous-jacente.

L'Open Policy Agent (OPA) présente des avantages par rapport aux moteurs de politiques personnalisés. OPA et son évaluation des politiques avec Rego fournissent un moteur de politiques flexible et prédéfini qui prend en charge la rédaction de politiques dans un langage déclaratif de haut niveau. Cela réduit considérablement le niveau d'effort requis pour mettre en œuvre un moteur de politiques par rapport à la création de votre propre solution. En outre, l'OPA est en train de devenir rapidement une norme d'autorisation bien prise en charge.

Implémenter un plan de contrôle pour l'OPA pour DevOps, la surveillance et la journalisation

Étant donné que l'OPA ne fournit aucun moyen de mettre à jour et de suivre les modifications apportées à la logique d'autorisation par le biais du contrôle de source, nous vous recommandons d'implémenter un plan de contrôle pour exécuter ces fonctions. Cela permettra de distribuer plus facilement les mises à jour aux agents de l'OPA, en particulier si l'OPA fonctionne dans un système distribué, ce qui réduira le fardeau administratif lié à l'utilisation de l'OPA. En outre, un plan de contrôle peut être utilisé pour collecter des journaux à des fins d'agrégation et pour surveiller le statut des agents OPA.

Configurer les fonctionnalités de journalisation et d'observabilité dans les autorisations vérifiées

Les autorisations vérifiées permettent d'accéder facilement aux fonctionnalités d'observabilité. Vous pouvez configurer le service pour enregistrer toutes les tentatives d'accès aux groupes de CloudWatch log Amazon AWS CloudTrail, aux compartiments S3 ou aux flux de livraison Amazon

Data Firehose afin de permettre une réponse rapide aux incidents de sécurité et aux demandes d'audit. En outre, vous pouvez surveiller l'état du service par le biais du Tableau de bord AWS Health. Dans la mesure où Verified Permissions est un service géré, son intégrité est maintenue par AWS d'autres services gérés et vous pouvez configurer ses fonctionnalités d'observabilité à l'aide d'autres services AWS gérés.

Utiliser un CI/CD pipeline pour provisionner et mettre à jour les magasins de politiques et les politiques dans Verified Permissions

Les autorisations vérifiées étant un service géré, vous n'avez pas besoin de gérer, de configurer ou de maintenir des plans de contrôle ou des agents pour effectuer des mises à jour. Cependant, nous vous recommandons tout de même d'utiliser un pipeline d'intégration et de déploiement continus (CI/CD) pour administrer le déploiement des magasins de politiques d'autorisations vérifiées et des mises à jour des politiques à l'aide du AWS SDK. Cet effort peut supprimer les efforts manuels et réduire le risque d'erreurs de l'opérateur lorsque vous apportez des modifications aux ressources d'autorisations vérifiées.

Déterminez si des données externes sont requises pour les décisions d'autorisation et sélectionnez un modèle adapté

Si un PDP peut prendre des décisions d'autorisation basées uniquement sur les données contenues dans un jeton Web JSON (JWT), il n'est généralement pas nécessaire d'importer des données externes pour faciliter la prise de ces décisions. Si vous utilisez Verified Permissions ou OPA en tant que PDP, il peut également accepter des entrées supplémentaires transmises dans le cadre de la demande, même si ces données ne sont pas incluses dans un JWT. Pour les autorisations vérifiées, vous pouvez utiliser un paramètre de contexte pour les données supplémentaires. Pour OPA, vous pouvez utiliser les données JSON comme entrée de surcharge. Si vous utilisez un JWT, les méthodes de saisie de contexte ou de surcharge sont généralement beaucoup plus simples que la gestion de données externes dans une autre source. Si des données externes plus complexes sont nécessaires pour prendre des décisions d'autorisation, OPA propose plusieurs modèles pour récupérer des données externes, et Verified Permissions peut compléter les données de ses demandes d'autorisation en référençant des sources externes avec un service d'autorisation.

FAQ

Cette section fournit des réponses aux questions fréquemment posées sur la mise en œuvre du contrôle d'accès et de l'autorisation des API dans les applications SaaS multi-locataires.

Q. Quelle est la différence entre l'autorisation et l'authentification ?

R. L'authentification est le processus qui permet de vérifier qui est un utilisateur. L'autorisation accorde aux utilisateurs l'autorisation d'accéder à une ressource spécifique.

Q. Quelle est la différence entre l'autorisation et l'isolation des locataires dans une application SaaS ?

R. L'isolation des locataires fait référence aux mécanismes explicites utilisés dans un système SaaS pour garantir que les ressources de chaque locataire, même lorsqu'elles opèrent sur une infrastructure partagée, sont isolées. L'autorisation multi-locataires fait référence à l'autorisation des actions entrantes et à la prévention de leur mise en œuvre sur le mauvais locataire. Un utilisateur hypothétique pourrait être authentifié et autorisé, mais pourrait toujours accéder aux ressources d'un autre locataire. Rien dans l'authentification et l'autorisation ne bloque nécessairement cet accès, mais l'isolation des locataires est nécessaire pour atteindre cet objectif. Pour plus d'informations sur ces deux concepts, consultez la discussion sur l'isolation des locataires dans le livre blanc sur les principes fondamentaux de l'architecture AWS SaaS.

Q : Pourquoi dois-je envisager l'isolation des locataires pour mon application SaaS ?

R. Les applications SaaS ont plusieurs locataires. Un locataire peut être une organisation cliente ou toute entité externe utilisant cette application SaaS. Selon la façon dont l'application est conçue, cela signifie que les locataires peuvent accéder à des ressources partagées APIs, à des bases de données ou à d'autres ressources. Il est important de maintenir l'isolement des locataires, c'est-à-dire des structures qui contrôlent étroitement l'accès aux ressources et bloquent toute tentative d'accès aux ressources d'un autre locataire, afin d'empêcher les utilisateurs d'un locataire d'accéder aux informations privées d'un autre locataire. Les applications SaaS sont souvent conçues pour garantir que l'isolement des locataires est maintenu dans l'ensemble d'une application et que les locataires ne peuvent accéder qu'à leurs propres ressources.

Q. Pourquoi ai-je besoin d'un modèle de contrôle d'accès ?

R. Les modèles de contrôle d'accès sont utilisés pour créer une méthode cohérente permettant de déterminer comment accorder l'accès aux ressources d'une application. Cela peut être fait en attribuant des rôles aux utilisateurs qui sont étroitement liés à la logique métier, ou cela peut être

basé sur d'autres attributs contextuels tels que l'heure ou le fait qu'un utilisateur répond à une condition prédéfinie. Les modèles de contrôle d'accès constituent la logique de base que votre application utilise pour prendre des décisions d'autorisation afin de déterminer les autorisations des utilisateurs.

Q. Le contrôle d'accès aux API est-il nécessaire pour mon application ?

R. Oui. APIs doit toujours vérifier que l'appelant dispose de l'accès approprié. Le contrôle d'accès généralisé aux API garantit également que l'accès n'est accordé qu'en fonction des locataires afin de maintenir une isolation appropriée.

Q : Pourquoi les moteurs de politiques sont-ils PDPs recommandés pour l'autorisation ?

R. Un point de décision politique (PDP) permet de transférer la logique d'autorisation du code d'application vers un système distinct. Cela peut simplifier le code de l'application. Il fournit également une interface easy-to-use idempotente pour prendre des décisions d'autorisation pour les microservices APIs, les couches Backend for Frontend (BFF) ou tout autre composant d'application.

Q. Qu'est-ce qu'un PEP ?

R. Un point d'application des politiques (PEP) est chargé de recevoir les demandes d'autorisation qui sont envoyées au PDP pour évaluation. Un PEP peut se trouver n'importe où dans une application où les données et les ressources doivent être protégées ou où une logique d'autorisation est appliquée. PEPs sont relativement simples par rapport à PDPs. Un PEP est uniquement chargé de demander et d'évaluer une décision d'autorisation et ne nécessite aucune logique d'autorisation pour y être incorporée.

Q : Comment choisir entre Amazon Verified Permissions et OPA ?

R. Pour choisir entre les autorisations vérifiées et l'Open Policy Agent (OPA), gardez toujours à l'esprit votre cas d'utilisation et vos exigences uniques. Les autorisations vérifiées constituent un moyen entièrement géré de définir des autorisations précises, d'auditer les autorisations entre les applications et de centraliser le système d'administration des politiques pour vos applications tout en répondant aux exigences de latence de vos applications grâce à un traitement en millisecondes. OPA est un moteur de politiques open source à usage général qui peut également vous aider à unifier les politiques au sein de votre pile d'applications. Pour exécuter OPA, vous devez l'héberger dans votre AWS environnement, généralement avec un conteneur ou AWS Lambda des fonctions.

Verified Permissions utilise le langage de politique open source Cedar, tandis qu'OPA utilise Rego. Par conséquent, la connaissance de l'une de ces langues peut vous inciter à choisir cette solution.

Cependant, nous vous recommandons de vous renseigner sur les deux langues, puis de revenir sur le problème que vous essayez de résoudre afin de trouver la meilleure solution pour votre cas d'utilisation.

Q. Existe-t-il des alternatives open source aux autorisations vérifiées et à l'OPA ?

R. Il existe quelques systèmes open source similaires à Verified Permissions et à Open Policy Agent (OPA), tels que le Common Expression Language (CEL). Ce guide se concentre à la fois sur les autorisations vérifiées, en tant que service de gestion des autorisations évolutif et précis, et sur l'OPA, qui est largement adopté, documenté et adaptable à de nombreux types d'applications et d'exigences d'autorisation.

Q. Dois-je écrire un service d'autorisation pour utiliser l'OPA, ou puis-je interagir directement avec l'OPA ?

R. Vous pouvez interagir directement avec l'OPA. Dans le contexte de ce guide, un service d'autorisation fait référence à un service qui traduit les demandes de décision d'autorisation en requêtes OPA, et vice versa. Si votre application peut interroger et accepter directement les réponses de l'OPA, il n'est pas nécessaire d'introduire cette complexité supplémentaire.

Q. Comment puis-je surveiller mes agents OPA à des fins de disponibilité et d'audit ?

R. OPA assure la journalisation et la surveillance de base du temps de disponibilité, bien que sa configuration par défaut soit probablement insuffisante pour les déploiements en entreprise. Pour plus d'informations, consultez la DevOps section Surveillance et journalisation plus haut dans ce guide.

Q : Comment puis-je surveiller les autorisations vérifiées à des fins de disponibilité et d'audit ?

A. Verified Permissions est un service AWS géré dont la disponibilité peut être contrôlée via le Tableau de bord AWS Health. En outre, Verified Permissions est capable de se connecter à AWS CloudTrail Amazon CloudWatch Logs, Amazon S3 et Amazon Data Firehose.

Q. Quels systèmes d'exploitation et AWS services puis-je utiliser pour exécuter OPA ?

R. Vous pouvez exécuter OPA sur macOS, Windows et Linux. Les agents OPA peuvent être configurés sur des agents Amazon Elastic Compute Cloud (Amazon EC2) ainsi que sur des services de conteneurisation tels qu'Amazon Elastic Container Service (Amazon ECS) et Amazon Elastic Kubernetes Service (Amazon EKS).

Q. Quels systèmes d'exploitation et AWS services puis-je utiliser pour exécuter les autorisations vérifiées ?

A. Verified Permissions est un service AWS géré par AWS. Aucune configuration, installation ou hébergement supplémentaire n'est nécessaire pour utiliser les autorisations vérifiées, à l'exception de la possibilité de faire des demandes d'autorisation au service.

Q. Puis-je lancer OPA AWS Lambda ?

R. Vous pouvez exécuter OPA sur Lambda en tant que bibliothèque Go. Pour savoir comment procéder pour un autorisateur Lambda API Gateway, consultez le billet de AWS blog Création d'un autorisateur Lambda personnalisé à l'aide d'Open Policy Agent.

Q : Comment choisir entre une approche PDP distribuée et une approche PDP centralisée ?

R. Cela dépend de votre candidature. Il sera probablement déterminé en fonction de la différence de latence entre un modèle PDP distribué et centralisé. Nous vous recommandons de créer une preuve de concept et de tester les performances de votre application pour vérifier votre solution.

Q. Puis-je également utiliser OPA pour des cas d'utilisation APIs ?

R. Oui. La documentation OPA fournit des exemples pour Kubernetes, Envoy, Docker, Kafka, SSH et sudo, et Terraform. De plus, OPA peut renvoyer des réponses JSON arbitraires aux requêtes en utilisant les règles partielles Rego. En fonction de la requête, l'OPA peut être utilisé pour répondre à de nombreuses questions avec des réponses JSON.

Q. Puis-je utiliser les autorisations vérifiées pour d'autres cas d'utilisation APIs ?

R. Oui. Verified Permissions peut fournir une `DENY` réponse `ALLOW` ou une réponse à toute demande d'autorisation reçue. Les autorisations vérifiées peuvent fournir des réponses d'autorisation pour toute application ou service nécessitant des décisions d'autorisation.

Q : Puis-je créer des politiques dans Verified Permissions en utilisant le langage de politique IAM ?

R. Non Vous devez utiliser le langage politique de Cedar pour créer des politiques. Cedar est conçu pour prendre en charge la gestion des autorisations pour les ressources des applications clients, tandis que le langage de politique Gestion des identités et des accès AWS (IAM) a évolué pour prendre en charge le contrôle d'accès aux AWS ressources.

Étapes suivantes

La complexité de l'autorisation et du contrôle d'accès à l'API pour les applications SaaS à locataires multiples peut être surmontée en adoptant une approche standardisée et indépendante du langage pour prendre des décisions en matière d'autorisation. Ces approches intègrent des points de décision politique (PDPs) et des points d'application des politiques (PDPs) qui renforcent l'accès de manière flexible et généralisée. Plusieurs approches du contrôle d'accès, telles que le contrôle d'accès basé sur les rôles (RBAC), le contrôle d'accès par attributs (ABAC) ou une combinaison des deux, peuvent être intégrées dans une stratégie de contrôle d'accès cohérente. En supprimant la logique d'autorisation d'une application, il n'est plus nécessaire d'inclure des solutions ad hoc dans le code de l'application pour gérer le contrôle d'accès. La mise en œuvre et les meilleures pratiques présentées dans ce guide visent à informer et à standardiser une approche de la mise en œuvre de l'autorisation et du contrôle d'accès à l'API dans les applications SaaS à locataires multiples. Vous pouvez utiliser ce guide comme première étape dans la collecte d'informations et la conception d'un système de contrôle d'accès et d'autorisation robuste pour votre application. Prochaines étapes :

- Examinez vos besoins en matière d'autorisation et d'isolement des locataires et sélectionnez un modèle de contrôle d'accès pour votre application.
- Créez une preuve de concept à tester en utilisant Amazon Verified Permissions ou Open Policy Agent (OPA), ou en écrivant votre propre moteur de politique personnalisé.
- Identifiez APIs et localisez dans votre application les endroits où la mise en œuvre PEPs doit être effectuée.

Ressources

Références

- Documentation relative aux autorisations vérifiées par Amazon (AWS documentation)
- Comment utiliser les autorisations vérifiées d'Amazon à des fins d'autorisation (article de AWS blog)
- Implémenter un fournisseur de politique d'autorisation personnalisé pour les applications ASP.NET Core à l'aide d'Amazon Verified Permissions (article de AWS blog)
- Gérez les rôles et les droits avec le PBAC à l'aide des autorisations vérifiées par Amazon (AWS article de blog)
- Contrôle d'accès SaaS à l'aide d'Amazon Verified Permissions avec un magasin de politiques par locataire (article de AWS blog)
- La documentation officielle de l'OPA
- Pourquoi les entreprises doivent adopter le projet CNCF le plus récemment diplômé — Open Policy Agent (article Forbes de Janakiram MSV, 8 février 2021)
- Création d'un autorisateur Lambda personnalisé à l'aide d'Open Policy Agent (AWS article de blog)
- Concrétisez une politique sous forme de code avec AWS Cloud Development Kit via Open Policy Agent (article de AWS blog)
- Gouvernance du cloud et conformité AWS avec la politique sous forme de code (article de AWS blog)
- Utilisation d'Open Policy Agent sur Amazon EKS (article de AWS blog)
- Conformité sous forme de code pour Amazon ECS à l'aide d'Open Policy Agent EventBridge, Amazon et AWS Lambda (article de AWS blog)
- Contre-mesures basées sur des politiques pour Kubernetes — Partie 1 (article de blog)AWS
- Utilisation des autorisateurs Lambda d'API Gateway (documentation)AWS

Outils

- The Cedar Playground (pour tester Cedar dans un navigateur)
- Dépôt Cedar Github
- Référence linguistique Cedar

- The Rego Playground (pour tester Rego dans un navigateur)
- GitHubRéférentiel OPA

Partenaires

- Partenaires de gestion des identités et des accès
- Partenaires de sécurité des applications
- Partenaires en matière de gouvernance du cloud
- Partenaires en matière de sécurité et de conformité
- Partenaires des opérations de sécurité et de l'automatisation
- Partenaires en ingénierie de sécurité

Historique du document

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

Modification	Description	Date
Détails et exemples ajoutés pour Amazon Verified Permissions	Ajout de discussions détaillées, d'exemples et de code pour l'utilisation d'Amazon Verified Permissions pour implémenter un PDP. Les nouvelles sections incluent : • Implémentation d'un PDP à l'aide des autorisations vérifiées par Amazon • Modèles de conception pour Amazon Verified Permissions • Considérations relatives à la conception multi-locataires d'Amazon Verified Permissions • Extraction de données externes pour un PDP dans Amazon Verified Permissions	28 mai 2024
Informations clarifiées	Clarification du PDP distribué avec aucun modèle PEPs de APIs conception.	10 janvier 2024
Ajout de brèves informations sur le nouveau AWS service	Ajout d'informations sur Amazon Verified Permissions, qui fournit les mêmes	22 mai 2023

	fonctionnalités et avantages que l'OPA.	
—	Publication initiale	17 août 2021

AWS Glossaire des directives prescriptives

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

Nombres

7 R

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

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

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

A

ABAC

Voir contrôle d'accès basé sur les attributs.

services abstraits

Consultez la section Services gérés.

ACIDE

Voir atomicité, consistance, isolation, durabilité.

migration active-active

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

migration active-passive

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

fonction d'agrégation

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

AI

Voir intelligence artificielle.

AIOps

Voir les opérations d'intelligence artificielle.

anonymisation

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

anti-motif

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

contrôle des applications

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

portefeuille d'applications

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

intelligence artificielle (IA)

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

opérations d'intelligence artificielle (AIOps)

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

chiffrement asymétrique

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

atomicité, cohérence, isolement, durabilité (ACID)

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

contrôle d'accès par attributs (ABAC)

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

source de données faisant autorité

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

Zone de disponibilité

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

AWS Cadre d'adoption du cloud (AWS CAF)

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

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

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

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

B

mauvais bot

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

BCP

Consultez la section Planification de la continuité des activités.

graphique de comportement

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

système de poids fort

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

classification binaire

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

filtre de Bloom

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

déploiement bleu/vert

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

bot

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

botnet

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

branche

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

accès par brise-vitre

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

stratégie existante (brownfield)

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

cache de tampon

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

capacité métier

Ce que fait une entreprise pour générer de la valeur (par exemple, les ventes, le service client ou le marketing). Les architectures de microservices et les décisions de développement peuvent être dictées par les capacités métier. Pour plus d'informations, veuillez consulter la section Organisation en fonction des capacités métier du livre blanc Exécution de microservices conteneurisés sur AWS.

planification de la continuité des activités (BCP)

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

C

CAF

Voir le cadre d'adoption du AWS cloud.

déploiement de Canary

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

CCo E

Voir le Centre d'excellence du cloud.

CDC

Voir capture des données de modification.

capture des données de modification (CDC)

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

ingénierie du chaos

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

CI/CD

Découvrez l'intégration continue et la livraison continue.

classification

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

chiffrement côté client

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

Centre d'excellence du cloud (CCoE)

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

cloud computing

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

modèle d'exploitation du cloud

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

étapes d'adoption du cloud

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

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

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

CMDB

Consultez la base de données de gestion des configurations.

référentiel de code

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

cache passif

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

données gelées

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

vision par ordinateur (CV)

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

dérive de configuration

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

base de données de gestion des configurations (CMDB)

Référentiel qui stocke et gère les informations relatives à une base de données et à son environnement informatique, y compris les composants matériels et logiciels ainsi que leurs configurations. Vous utilisez généralement les données d'une CMDB lors de la phase de découverte et d'analyse du portefeuille de la migration.

pack de conformité

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

intégration continue et livraison continue (CI/CD)

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

CV

Voir vision par ordinateur.

D

données au repos

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

classification des données

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

dérive des données

Une variation significative entre les données de production et les données utilisées pour entraîner un modèle ML, ou une modification significative des données d'entrée au fil du temps. La dérive des données peut réduire la qualité, la précision et l'équité globales des prédictions des modèles ML.

données en transit

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

maillage de données

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

minimisation des données

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

périmètre de données

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

prétraitement des données

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

provenance des données

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

sujet des données

Personne dont les données sont collectées et traitées.

entrepôt des données

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

langage de définition de base de données (DDL)

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

langage de manipulation de base de données (DML)

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

DDL

Voir langage de définition de base de données.

ensemble profond

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

deep learning

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

defense-in-depth

Approche de la sécurité de l'information dans laquelle une série de mécanismes et de contrôles de sécurité sont judicieusement répartis sur l'ensemble d'un réseau informatique afin de protéger la confidentialité, l'intégrité et la disponibilité du réseau et des données qu'il contient. Lorsque vous adoptez cette stratégie AWS, vous ajoutez plusieurs contrôles à différentes couches de

la AWS Organizations structure afin de sécuriser les ressources. Par exemple, une defense-in-depth approche peut combiner l'authentification multifactorielle, la segmentation du réseau et le chiffrement.

administrateur délégué

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

déploiement

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

environnement de développement

Voir environnement.

contrôle de détection

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

cartographie de la chaîne de valeur du développement (DVSM)

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

jumeau numérique

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

tableau des dimensions

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

catastrophe

Un événement qui empêche une charge de travail ou un système d'atteindre ses objectifs commerciaux sur son site de déploiement principal. Ces événements peuvent être des catastrophes naturelles, des défaillances techniques ou le résultat d'actions humaines, telles qu'une mauvaise configuration involontaire ou une attaque de logiciel malveillant.

reprise après sinistre (DR)

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

DML

Voir langage de manipulation de base de données.

conception axée sur le domaine

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

DR

Voir reprise après sinistre.

détection de dérive

Suivi des écarts par rapport à une configuration de référence. Par exemple, vous pouvez l'utiliser AWS CloudFormation pour détecter la dérive des ressources du système ou AWS Control Tower

pour détecter les modifications de votre zone d'atterrissage susceptibles d'affecter le respect des exigences de gouvernance.

DVSM

Voir la cartographie de la chaîne de valeur du développement.

E

EDA

Voir analyse exploratoire des données.

EDI

Voir échange de données informatisé.

informatique de périphérie

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

échange de données informatisé (EDI)

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

chiffrement

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

clé de chiffrement

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

endianisme

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

point de terminaison

Voir point de terminaison de service.

service de point de terminaison

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

planification des ressources d'entreprise (ERP)

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

chiffrement d'enveloppe

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

environment

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

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

environnements de préproduction et des environnements pour les tests d'acceptation par les utilisateurs.

épopée

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

ERP

Voir Planification des ressources d'entreprise.

analyse exploratoire des données (EDA)

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

F

tableau des faits

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

échouer rapidement

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

limite d'isolation des défauts

Dans le AWS Cloud, une limite telle qu'une zone de disponibilité Région AWS, un plan de contrôle ou un plan de données qui limite l'effet d'une panne et contribue à améliorer la résilience des

charges de travail. Pour plus d'informations, consultez la section Limites d'isolation des AWS pannes.

branche de fonctionnalités

Voir succursale.

fonctionnalités

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

importance des fonctionnalités

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

transformation de fonctionnalité

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

invitation en quelques coups

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

FGAC

Découvrez le contrôle d'accès détaillé.

contrôle d'accès détaillé (FGAC)

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

migration instantanée (flash-cut)

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

FM

Voir le modèle de fondation.

modèle de fondation (FM)

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

G

IA générative

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

blocage géographique

Voir les restrictions géographiques.

restrictions géographiques (blocage géographique)

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

Flux de travail Gitflow

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

image dorée

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

stratégie inédite

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

barrière de protection

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

H

HA

Découvrez la haute disponibilité.

migration de base de données hétérogène

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

haute disponibilité (HA)

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

modernisation des historiens

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

données de rétention

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

migration de base de données homogène

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

données chaudes

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

correctif

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

période de soins intensifs

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

I

IaC

Considérez l'infrastructure comme un code.

politique basée sur l’identité

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

application inactive

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

IIo T

Voir Internet industriel des objets.

infrastructure immuable

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

VPC entrant (d'entrée)

Dans une architecture AWS multi-comptes, un VPC qui accepte, inspecte et achemine les connexions réseau depuis l'extérieur d'une application. L'architecture AWS de référence de sécurité recommande de configurer votre compte réseau avec les fonctions entrantes, sortantes

et d'inspection VPCs afin de protéger l'interface bidirectionnelle entre votre application et l'Internet en général.

migration incrémentielle

Stratégie de basculement dans le cadre de laquelle vous migrez votre application par petites parties au lieu d'effectuer un basculement complet unique. Par exemple, il se peut que vous ne transfériez que quelques microservices ou utilisateurs vers le nouveau système dans un premier temps. Après avoir vérifié que tout fonctionne correctement, vous pouvez transférer progressivement des microservices ou des utilisateurs supplémentaires jusqu'à ce que vous puissiez mettre hors service votre système hérité. Cette stratégie réduit les risques associés aux migrations de grande ampleur.

Industry 4.0

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

infrastructure

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

infrastructure en tant que code (IaC)

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

Internet industriel des objets (IIoT)

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

VPC d'inspection

Dans une architecture AWS multi-comptes, un VPC centralisé qui gère les inspections du trafic réseau VPCs entre (identique ou Régions AWS différent), Internet et les réseaux locaux. L'architecture AWS de référence de sécurité recommande de configurer votre compte réseau

avec les fonctions entrantes, sortantes et d'inspection VPCs afin de protéger l'interface bidirectionnelle entre votre application et l'Internet en général.

Internet des objets (IoT)

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

interprétabilité

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

IoT

Voir Internet des objets.

Bibliothèque d'informations informatiques (ITIL)

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

gestion des services informatiques (ITSM)

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

ITIL

Consultez la bibliothèque d'informations informatiques.

ITSM

Voir Gestion des services informatiques.

L

contrôle d'accès basé sur des étiquettes (LBAC)

Une implémentation du contrôle d'accès obligatoire (MAC) dans laquelle une valeur d'étiquette de sécurité est explicitement attribuée aux utilisateurs et aux données elles-mêmes. L'intersection

entre l'étiquette de sécurité utilisateur et l'étiquette de sécurité des données détermine les lignes et les colonnes visibles par l'utilisateur.

zone de destination

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

grand modèle de langage (LLM)

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

migration de grande envergure

Migration de 300 serveurs ou plus.

LBAC

Voir contrôle d'accès basé sur des étiquettes.

principe de moindre privilège

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

lift and shift

Voir 7 Rs.

système de poids faible

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

LLM

Voir le grand modèle de langage.

environnements inférieurs

Voir environnement.

M

machine learning (ML)

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

branche principale

Voir succursale.

malware

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

services gérés

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

système d'exécution de la fabrication (MES)

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

MAP

Voir Migration Acceleration Program.

mécanisme

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

compte membre

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

MAILLES

Voir le système d'exécution de la fabrication.

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

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

microservice

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

architecture de microservices

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

Programme d'accélération des migrations (MAP)

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

migration à grande échelle

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

usine de migration

Équipes interfonctionnelles qui rationalisent la migration des charges de travail grâce à des approches automatisées et agiles. Les équipes de Migration Factory comprennent généralement des responsables des opérations, des analystes commerciaux et des propriétaires, des ingénieurs de migration, des développeurs et DevOps des professionnels travaillant dans le cadre de sprints. Entre 20 et 50 % du portefeuille d'applications d'entreprise est constitué de modèles répétés qui peuvent être optimisés par une approche d'usine. Pour plus d'informations, veuillez consulter la rubrique discussion of migration factories et le guide Cloud Migration Factory dans cet ensemble de contenus.

métadonnées de migration

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

modèle de migration

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

Évaluation du portefeuille de migration (MPA)

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

Évaluation de la préparation à la migration (MRA)

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

stratégie de migration

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

ML

Voir apprentissage automatique.

modernisation

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

évaluation de la préparation à la modernisation

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

applications monolithiques (monolithes)

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

MPA

Voir Évaluation du portefeuille de migration.

MQTT

Voir Message Queuing Telemetry Transport.

classification multi-classes

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

infrastructure mutable

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

O

OAC

Voir Contrôle d'accès à l'origine.

OAI

Voir l'identité d'accès à l'origine.

OCM

Voir gestion du changement organisationnel.

migration hors ligne

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

OI

Consultez la section Intégration des opérations.

OLA

Voir l'accord au niveau opérationnel.

migration en ligne

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

OPC-UA

Voir Open Process Communications - Architecture unifiée.

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

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

accord au niveau opérationnel (OLA)

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

examen de l'état de préparation opérationnelle (ORR)

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

technologie opérationnelle (OT)

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

intégration des opérations (OI)

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

journal de suivi d'organisation

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

gestion du changement organisationnel (OCM)

Cadre pour gérer les transformations métier majeures et perturbatrices du point de vue des personnes, de la culture et du leadership. L'OCM aide les organisations à se préparer et à effectuer la transition vers de nouveaux systèmes et de nouvelles politiques en accélérant l'adoption des changements, en abordant les problèmes de transition et en favorisant des changements culturels et organisationnels. Dans la stratégie de AWS migration, ce cadre est appelé accélération du personnel, en raison de la rapidité du changement requise dans les projets d'adoption du cloud. Pour plus d'informations, veuillez consulter le guide OCM.

contrôle d'accès d'origine (OAC)

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

identité d'accès d'origine (OAI)

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

ORR

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

DE

Voir technologie opérationnelle.

VPC sortant (de sortie)

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

P

limite des autorisations

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

informations personnelles identifiables (PII)

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

PII

Voir les informations personnelles identifiables.

manuel stratégique

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

PLC

Voir contrôleur logique programmable.

PLM

Consultez la section Gestion du cycle de vie des produits.

policy

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

persistance polyglotte

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

évaluation du portefeuille

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

predicate

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

prédicat pushdown

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

contrôle préventif

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

principal

Entité AWS capable d'effectuer des actions et d'accéder aux ressources. Cette entité est généralement un utilisateur root pour un Compte AWS rôle IAM ou un utilisateur. Pour plus

d'informations, veuillez consulter la rubrique Principal dans Termes et concepts relatifs aux rôles, dans la documentation IAM.

confidentialité dès la conception

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

zones hébergées privées

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

contrôle proactif

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

gestion du cycle de vie des produits (PLM)

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

environnement de production

Voir environnement.

contrôleur logique programmable (PLC)

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

chaînage rapide

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

pseudonymisation

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

publish/subscribe (pub/sub)

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

Q

plan de requête

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

régression du plan de requêtes

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

R

Matrice RACI

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

RAG

Voir Retrieval Augmented Generation.

rançongiciel

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

Matrice RASCI

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

RCAC

Voir contrôle d'accès aux lignes et aux colonnes.

réplica en lecture

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

réarchitecte

Voir 7 Rs.

objectif de point de récupération (RPO)

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

objectif de temps de récupération (RTO)

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

refactoriser

Voir 7 Rs.

Région

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

régression

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

réhéberger

Voir 7 Rs.

version

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

déplacer

Voir 7 Rs.

replateforme

Voir 7 Rs.

rachat

Voir 7 Rs.

résilience

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

politique basée sur les ressources

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

matrice responsable, redevable, consulté et informé (RACI)

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

contrôle réactif

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

retain

Voir 7 Rs.

se retirer

Voir 7 Rs.

Génération augmentée de récupération (RAG)

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

rotation

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

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

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

RPO

Voir l'objectif du point de récupération.

RTO

Voir l'objectif en matière de temps de rétablissement.

runbook

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

S

SAML 2.0

Un standard ouvert utilisé par de nombreux fournisseurs d'identité (IdPs). Cette fonctionnalité permet l'authentification unique fédérée (SSO), afin que les utilisateurs puissent se connecter

AWS Management Console ou appeler les opérations de l' AWS API sans que vous ayez à créer un utilisateur dans IAM pour tous les membres de votre organisation. Pour plus d'informations sur la fédération SAML 2.0, veuillez consulter À propos de la fédération SAML 2.0 dans la documentation IAM.

SCADA

Voir Contrôle de supervision et acquisition de données.

SCP

Voir la politique de contrôle des services.

secret

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

sécurité dès la conception

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

contrôle de sécurité

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

renforcement de la sécurité

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

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

Outils et services qui associent les systèmes de gestion des informations de sécurité (SIM) et de gestion des événements de sécurité (SEM). Un système SIEM collecte, surveille et analyse les

données provenant de serveurs, de réseaux, d'appareils et d'autres sources afin de détecter les menaces et les failles de sécurité, mais aussi de générer des alertes.

automatisation des réponses de sécurité

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

chiffrement côté serveur

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

Politique de contrôle des services (SCP)

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

point de terminaison du service

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

contrat de niveau de service (SLA)

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

indicateur de niveau de service (SLI)

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

objectif de niveau de service (SLO)

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

modèle de responsabilité partagée

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

SIEM

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

point de défaillance unique (SPOF)

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

SLA

Voir le contrat de niveau de service.

SLI

Voir l'indicateur de niveau de service.

SLO

Voir l'objectif de niveau de service.

split-and-seed modèle

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

SPOF

Voir point de défaillance unique.

schéma en étoile

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

modèle de figuier étrangleur

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

sous-réseau

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

contrôle de supervision et acquisition de données (SCADA)

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

chiffrement symétrique

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

tests synthétiques

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

invite du système

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

T

tags

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

variable cible

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

liste de tâches

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

environnement de test

Voir environnement.

entraînement

Pour fournir des données à partir desquelles votre modèle de ML peut apprendre. Les données d'entraînement doivent contenir la bonne réponse. L'algorithme d'apprentissage identifie des modèles dans les données d'entraînement, qui mettent en correspondance les attributs des données d'entrée avec la cible (la réponse que vous souhaitez prédire). Il fournit un modèle de ML qui capture ces modèles. Vous pouvez alors utiliser le modèle de ML pour obtenir des prédictions sur de nouvelles données pour lesquelles vous ne connaissez pas la cible.

passerelle de transit

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

flux de travail basé sur jonction

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

accès sécurisé

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

section Utilisation AWS Organizations avec d'autres AWS services dans la AWS Organizations documentation.

réglage

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

équipe de deux pizzas

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

U

incertitude

Un concept qui fait référence à des informations imprécises, incomplètes ou inconnues susceptibles de compromettre la fiabilité des modèles de ML prédictifs. Il existe deux types d'incertitude : l'incertitude épistémique est causée par des données limitées et incomplètes, alors que l'incertitude aléatoire est causée par le bruit et le caractère aléatoire inhérents aux données.

tâches indifférenciées

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

environnements supérieurs

Voir environnement.

V

mise à vide

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

contrôle de version

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

Appairage de VPC

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

vulnérabilités

Défaut logiciel ou matériel qui compromet la sécurité du système.

W

cache actif

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

données chaudes

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

fonction de fenêtre

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

charge de travail

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

flux de travail

Groupes fonctionnels d'un projet de migration chargés d'un ensemble de tâches spécifique. Chaque flux de travail est indépendant, mais prend en charge les autres flux de travail du projet.

Par exemple, le flux de travail du portefeuille est chargé de prioriser les applications, de planifier les vagues et de collecter les métadonnées de migration. Le flux de travail du portefeuille fournit ces actifs au flux de travail de migration, qui migre ensuite les serveurs et les applications.

VER

Voir écrire une fois, lire plusieurs.

WQF

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

écrire une fois, lire plusieurs (WORM)

Modèle de stockage qui écrit les données une seule fois et empêche leur suppression ou leur modification. Les utilisateurs autorisés peuvent lire les données autant de fois que nécessaire, mais ils ne peuvent pas les modifier. Cette infrastructure de stockage de données est considérée comme immuable.

Z

exploit Zero-Day

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

vulnérabilité « jour zéro »

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

invite Zero-Shot

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

application zombie

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

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