Autorisation SaaS multi-locataires et contrôle d'accès aux API

AWS Conseils prescriptifs

AWS Conseils prescriptifs: Autorisation SaaS multi-locataires et contrôle d'accès aux API

Table of Contents

Introduction 1
Résultats commerciaux ciblés 2
Types de contrôle d'accès 4
RBAC 4
ABAC 4
Approche hybride RBAC-ABAC 5
Comparaison des modèles de contrôle d'accès 5
Implémentation d'un PDP 7
Utilisation de l'OPA 7
Présentation de Rego 9
Exemple 1 : ABAC de base avec OPA et Rego 10
Exemple 2 : Contrôle d'accès multi-locataires et RBAC défini par l'utilisateur avec OPA et Rego 14
Exemple 3 : Contrôle d'accès multi-locataires pour RBAC et ABAC avec OPA et Rego 18
Exemple 4 : filtrage de l'interface utilisateur avec OPA et Rego 20
Utilisation d'un moteur de stratégie personnalisé 23
Implémentation d'un PEP 24
Demander une décision d'autorisation 24
Évaluation d'une décision d'autorisation 25
Modèles de conception pour les architectures SaaS à locataires multiples 26
PDP centralisé avec des PEP sur des API 26
PDP distribué avec des PEP sur des API 29
PDP distribué sous forme de bibliothèque 32
Considérations sur l'implémentation 33
DevOps, surveillance et journalisation 33
Extraction de données externes pour un PDP 33
Regroupement 34
Réplication (envoi de données) 34
Extraction dynamique des données 35
Utilisation d'un service d'autorisation pour la mise en œuvre 35
Recommandations pour l'isolation des locataires et la confidentialité des données externes pour le RBAC 37
Bonnes pratiques 39
Sélectionnez un modèle de contrôle d'accès adapté à votre application 39

Mettre en œuvre un PDP 39
Implémentez des PEP pour chaque API de votre entreprise 39
Envisagez d'utiliser l'OPA comme moteur de politiques pour votre PDP 40
Mettre en œuvre un plan de contrôle pour l'OPA pour DevOps la surveillance et la journalisation 40
Déterminez si des données externes sont requises pour les décisions d'autorisation et sélectionnez un modèle adapté 40
FAQ 41
Quelle est la différence entre l'autorisation et l'authentification ? 41
Pourquoi ajouter une autorisation à mon application SaaS ? 41
Pourquoi ai-je besoin d'un modèle de contrôle d'accès ? 41
Le contrôle d'accès à l'API est-il nécessaire pour mon application ? 41
Pourquoi les moteurs de politiques ou PDP sont-ils recommandés pour l'autorisation ? 42
Qu'est-ce qu'un PEP ? 42
Existe-t-il des alternatives open source à l'OPA ? 42
Dois-je rédiger un service d'autorisation pour utiliser OPA, ou puis-je interagir directement avec OPA ? 42
Comment puis-je surveiller mes agents OPA à des fins de disponibilité et d'audit ? 43
Quels systèmes d'exploitation etAWS services puis-je utiliser pour exécuter OPA ? 43
Puis-je lancer OPA surAWS Lambda ? 43
Comment choisir entre une approche PDP distribuée et une approche PDP centralisée ? 43
Puis-je utiliser OPA pour des cas d'utilisation autres que les API ? 43
Étapes suivantes 44
Ressources 45
Références 45
Outils 45
Partenaires 45
Historique du document 46
Glossaire 47
..... 47
A 48
B 51
C 52
D 56
E 60
F 62

G 63
H 64
I 65
L 68
M 69
O 73
P 75
Q 77
R 78
S 80
T 84
U 85
V 86
W 86
Z 88
..... lxxxix

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

Janvier 2024 (historique du document)

L'autorisation et le contrôle d'accès aux API pour les applications SaaS (Software as a Service) multi-locataires sont des sujets complexes. Cette complexité est évidente si l'on considère la prolifération des API de microservices qui doivent être sécurisées et le grand nombre de conditions d'accès qui découlent des différents locataires, des caractéristiques des utilisateurs et des états des applications. Pour résoudre efficacement ces problèmes, une solution doit renforcer le contrôle d'accès aux nombreuses API présentées par les microservices, les couches Backend for Frontend (BFF) et les autres composants d'une application SaaS mutualisée. Cette approche généralisée doit être associée à un paradigme d'application flexible 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. En outre, le contrôle d'accès aux API était généralement inutile, car il n'y avait pas autant d'API à sécuriser. Le changement de paradigme dans la conception des applications pour favoriser les microservices et les architectures orientées services a augmenté le nombre d'API 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 multi-locataires peut devenir très complexe, et des modèles cloisonnés ou très inefficaces sont souvent utilisés pour préserver cette 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 de manière idempotente à toutes les API d'une application.
- L'abstraction empêche les modifications accidentelles de la logique d'autorisation et rend son intégration dans une application SaaS cohérente et simple.

- L'abstraction évite d'avoir à écrire une logique d'autorisation personnalisée pour chaque point de terminaison d'API.
- 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.

Résultats commerciaux ciblés

Ce guide prescriptif décrit les modèles de conception idempotents 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 avec l'Open Policy Agent (OPA) et l'intégration des points d'application des politiques (PEP) dans les API. 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.

> **Note**
>
> Bien que ce guide se concentre principalement sur l'AWSOPA, il propose une nouvelle offre de service appelée Amazon Verified Permissions qui partage en grande partie les mêmes fonctionnalités et avantages que l'OPA. Pour plus de détails, nous vous recommandons de consulter la documentation Amazon Verified Permissions. La plupart des concepts, techniques et meilleures pratiques présentés dans ce guide concernent les autorisations vérifiées par Amazon.

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 est réalisée par le biais de PDP, de PEP, de moteurs de politiques et d'OPA. Ces

concepts et technologies aident à maintenir l'isolement des locataires dans une application SaaS multi-locataires et fournissent une approche globale de l'autorisation des API pour l'application.

- 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 des moteurs de politiques tels que OPA pour séparer les décisions d'autorisation du code de l'application et pour appliquer des politiques cohérentes au sein d'une application. 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 en détail comment ces modèles s'appliquent aux moteurs de politiques, et à l'OPA en particulier. 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 pour sécuriser les API de manière cohérente et omniprésente dans une application avec un minimum d'effort. Cela est particulièrement utile pour les architectures d'applications orientées services ou microservices qui utilisent généralement un grand nombre d'API 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.

Types de contrôle d'accès

Vous pouvez utiliser deux modèles définis de manière générale 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 abordés dans cette section. Le modèle que vous devrez utiliser dépendra de votre cas d'utilisation particulier. 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 de marketing dans le cadre 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.
- L'extension à une taille importante nécessite une administration et une mise en correspondance constantes des autorisations en fonction des 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 une 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 au 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, la capacité ou non d'un utilisateur à traiter un paiement est déterminée 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 car il permet de prendre des décisions d'autorisation dynamiques, contextuelles et granulaires. Cependant, le modèle ABAC est difficile à mettre en œuvre dans un premier temps. 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 à mettre 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 de l'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 en corrélation avec les comparaisons effectuées dans ce tableau.

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

Facteur	RBAC	ABAC	Hybride
Conscient du contexte	Non	Oui	Quelque peu
effort de mise en œuvre	Faible	Élevée	Medium

Implémentation d'un PDP

Le point de décision politique (PDP) peut être considéré comme un moteur de politiques ou de règles. Ce composant est chargé d'appliquer les politiques ou les 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 une exigence de PDP. Un PDP permet de décharger la logique d'autorisation contenue dans le 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 permettant de prendre des décisions d'autorisation pour les API, les microservices, les couches Backend for Frontend (BFF) ou tout autre composant de l'application.

Les sections suivantes décrivent deux méthodes de mise en œuvre d'un PDP : Open Policy Agent (OPA) et les solutions personnalisées. Toutefois, cette liste n'est pas exhaustive.

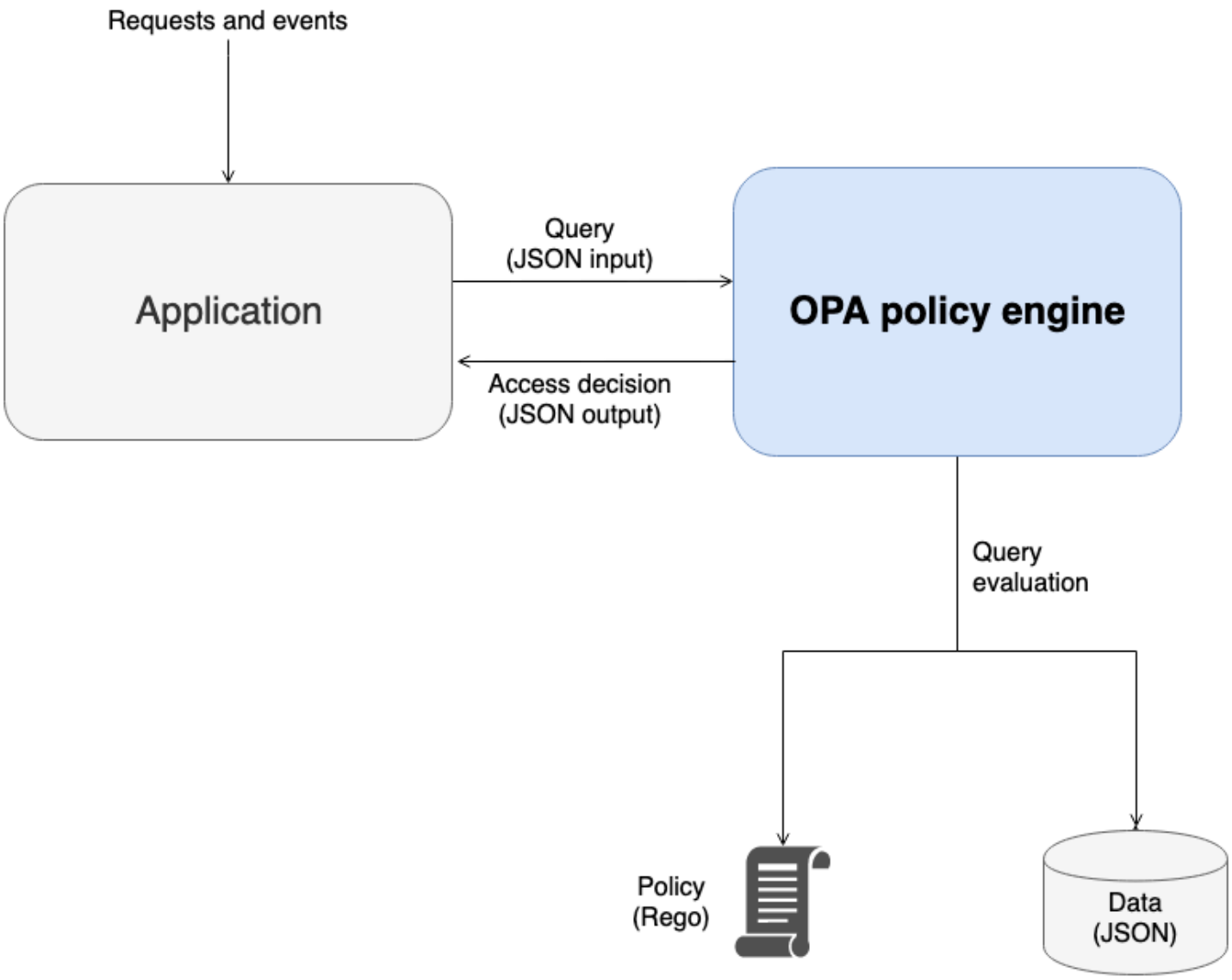
Méthodes de mise en œuvre du PDP :

- Utilisation de l'OPA
- Utilisation d'un moteur de stratégie personnalisé

Utilisation de l'OPA

La méthode préférée pour implémenter un PDP est d'utiliser l'Open Policy Agent (OPA). OPA est un moteur de politiques open source à usage général. L'OPA présente de nombreux cas d'utilisation, mais le cas d'utilisation pertinent pour la mise en œuvre du PDP est sa capacité à découpler 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. OPA expose également une API RESTful qui permet de récupérer les décisions d'autorisation de manière 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 au format JSON. OPA est également capable d'importer des données externes qui pourraient être pertinentes pour prendre une décision d'autorisation.



L'OPA présente plusieurs 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 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 personnalisé.
- L'OPA simplifie la logique d'autorisation en ayant des politiques rédigées dans un langage déclaratif. Vous pouvez modifier et administrer ces politiques et règles indépendamment de tout code d'application, sans avoir besoin de compétences en développement d'applications.
- OPA expose une API RESTful, qui simplifie l'intégration avec les points d'application des politiques (PEP).

- OPA fournit un support intégré pour la validation et le décodage des jetons Web JSON (JWT).
- 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. OPA fournit quelques 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. Il est recommandé d'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 à partir de sources externes par défaut. Une source externe de données pour une décision d'autorisation peut être une base de données contenant des attributs utilisateur. La manière dont les données externes sont fournies à l'OPA offre une certaine flexibilité : 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.

Dans cette section :

- Présentation de Rego
- Exemple 1 : ABAC de base avec OPA et Rego
- Exemple 2 : Contrôle d'accès multi-locataires et RBAC défini par l'utilisateur avec OPA et Rego
- Exemple 3 : Contrôle d'accès multi-locataires pour RBAC et ABAC avec OPA et Rego
- Exemple 4 : filtrage de l'interface utilisateur avec OPA et Rego

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 fondées sur des politiques concernant les ressources, les identités et les opérations de l'infrastructure. Rego vous permet de rédiger une politique sur n'importe quelle couche d'une pile ou d'un domaine sans avoir besoin de modifier ou d'étendre 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 ?

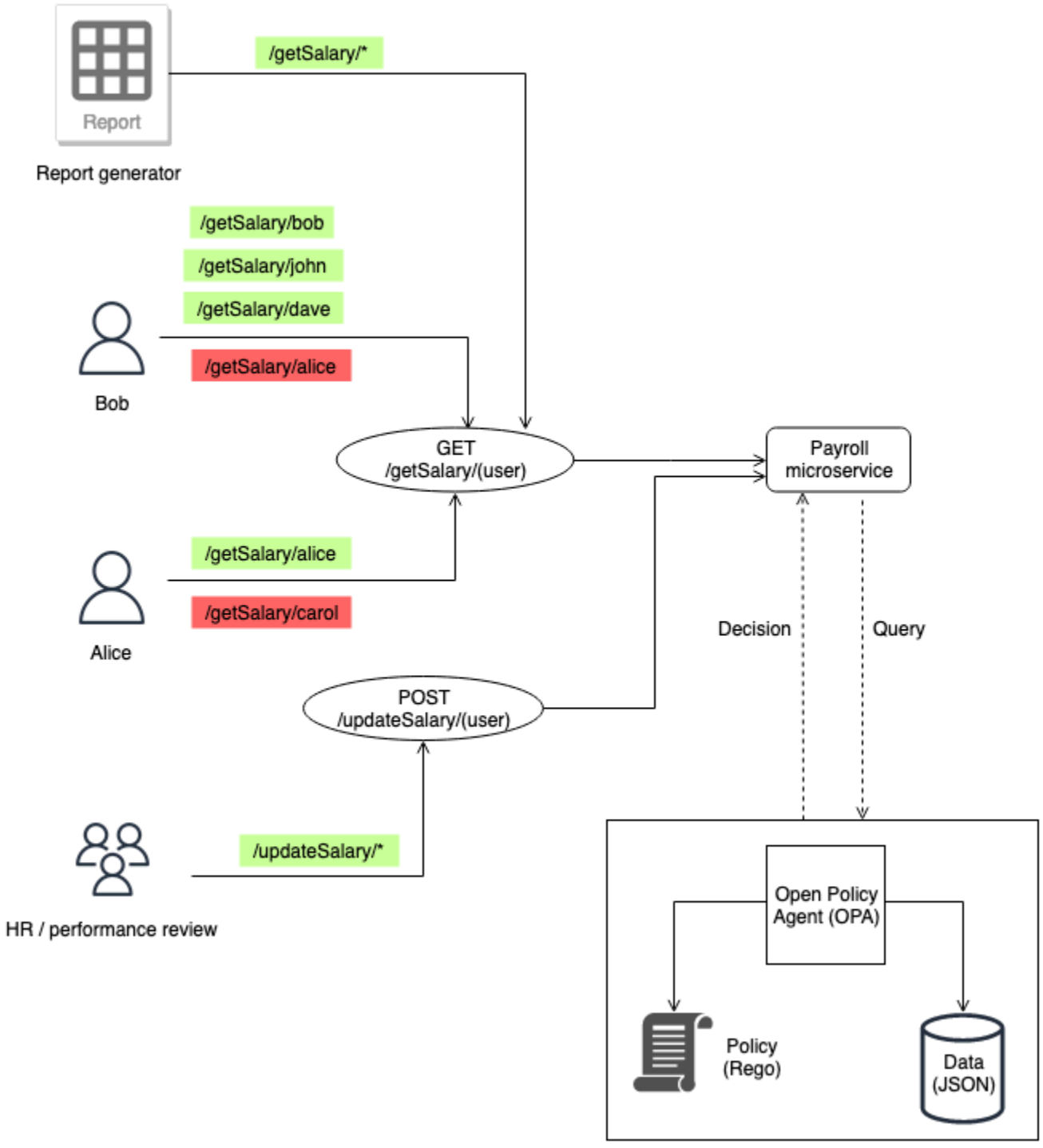
Pour répondre à ces questions, Rego utilise une philosophie de base sur la manière dont ces décisions peuvent être prises. Les deux principes clés lors de la rédaction d'une politique dans Rego sont les suivants :

- 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 données JSON/YAML. Les langages de programmation tels que C, Java, Go et Python constituent 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 permettant de prendre des décisions politiques avec ces informations.

Exemple 1 : ABAC de base avec OPA et Rego

Cette section décrit un scénario dans lequel l'OPA est utilisé pour prendre des décisions d'accès concernant les utilisateurs autorisés à accéder aux informations d'un microservice de paie fictif. Des extraits de code Rego sont fournis pour montrer comment vous pouvez utiliser Rego pour prendre des décisions en matière de contrôle d'accès. Ces exemples ne sont ni exhaustifs ni ne constituent une exploration complète des capacités de Rego et OPA. Pour une présentation plus complète 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 Payroll pour voir son propre salaire, le microservice Payroll peut rediriger l'appel d'API vers l'API OPA RESTful pour prendre une décision d'accès. Le service de paie demande à OPA de prendre une décision avec l'entrée JSON suivante :

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

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 dans la requête en la liant à la variable globale en entrée. L'opérateur point est utilisé avec cette variable pour accéder à ses valeurs. La règle Rego est`allow`renvoyée `true` si les expressions de la règle sont également vraies. La règle Rego vérifie que`method` l'entrée est égale à`GET`. Il vérifie ensuite que le premier élément du chemin de la liste se trouve`getSalary` bien 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 le`user` fait de faire 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 qui leur rend compte.

Dans cet exemple, 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 font rapport à la valeur de`input.user`, qui, dans ce cas, est`bob`. La`[_]` structure de la règle est utilisée pour itérer sur 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.

Synthèse

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 qui relève d'eux.

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

```
default allow = false
```

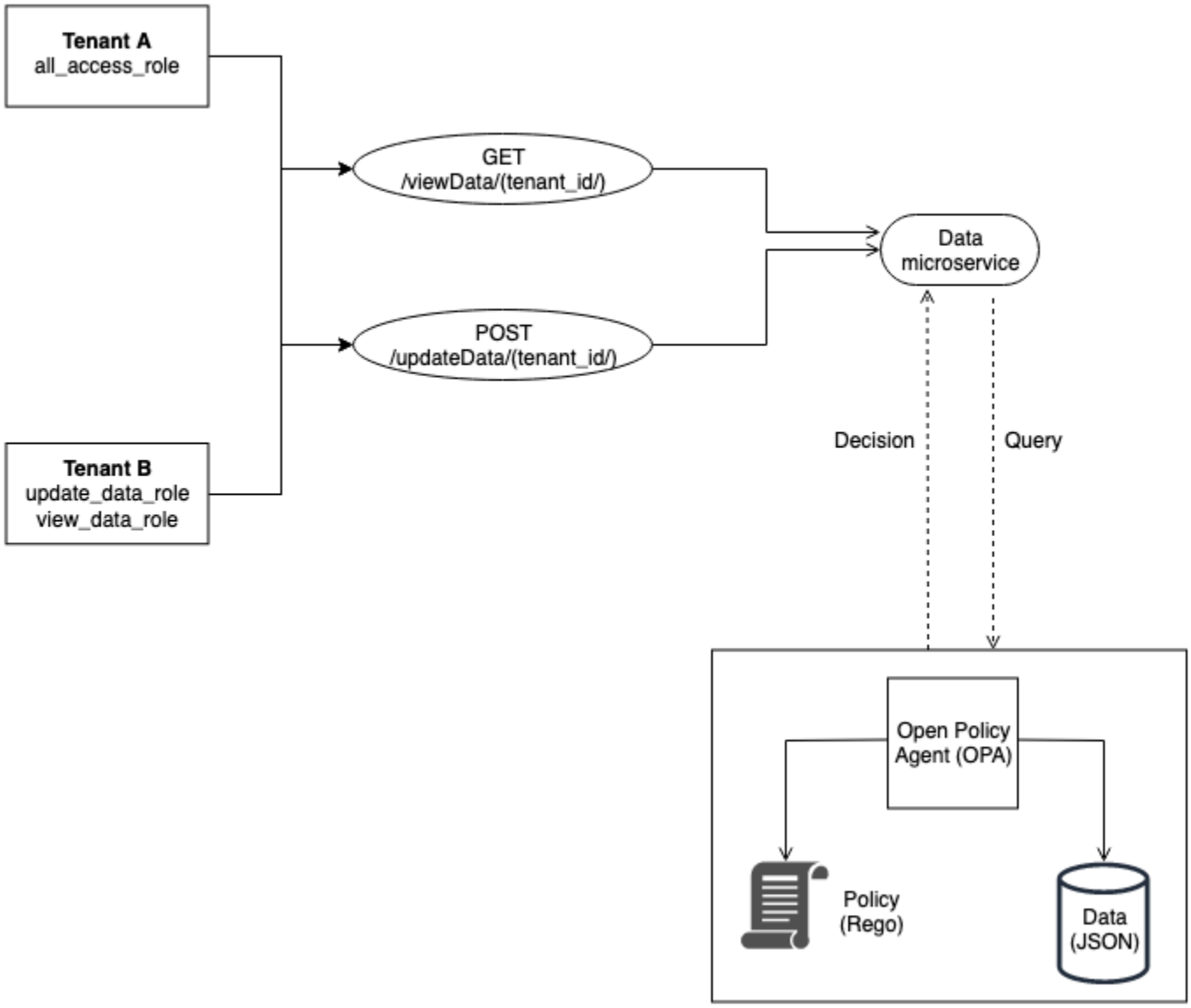
```
allow = true {
    input.method == "GET"
    input.path = ["getSalary", user]
    input.user == user
}

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

La première règle de la politique permet à tout utilisateur d'accéder à ses propres informations salariales, comme indiqué précédemment. Le fait d'avoir deux règles portant le même nom fonctionne comme une logique ou un opérateur dans Rego.`allow` La deuxième règle extrait la liste de tous les subordonnés directs associés à`input.user` (à partir des données du diagramme précédent) et attribue 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 figure dans la`managers` variable.

Les exemples présentés dans 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 Rego Playground.

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 mis en œuvre sur une API pour une application mutualisée avec des rôles personnalisés définis par les utilisateurs locataires. Il montre également comment l'accès peut être restreint en fonction du locataire. Ce modèle montre comment l'OPA peut prendre des décisions d'autorisation granulaires sur la base des informations fournies dans 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 un fournisseur d'identité (IdP) qui peut servir de source de vérité lors du mappage des rôles définis par le locataire avec les autorisations et avec le locataire lui-même.

Cet exemple utilise deux politiques dans 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 dont l'entrée est la suivante :

```
{
    "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` passe 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 d'entrée de requête OPA.
3. L'OPA utilise la règle demandée dans les politiques OPA pour évaluer les données fournies. L'OPA utilise également les données de`RBAC data` pour évaluer les entrées. OPA exécute 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 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 des autorisations de rôles à partir des données externes des rôles et les affecte à la variable`role_permissions`. Cette liste est récupérée à l'aide du rôle défini par le locataire qui est associé à l'utilisateur dans`input.role`.

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

4. 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 à`/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 la même sortie 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 RBAC contiennent toujours l'`viewData`autorisation qui leur est associée. Pour appliquer le même type de contrôle d'accès`/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'isolation des locataires et vérifie que le rôle défini par le locataire accorde l'autorisation à l'appelant de l'API. Pour voir comment cela pourrait 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"
}
```

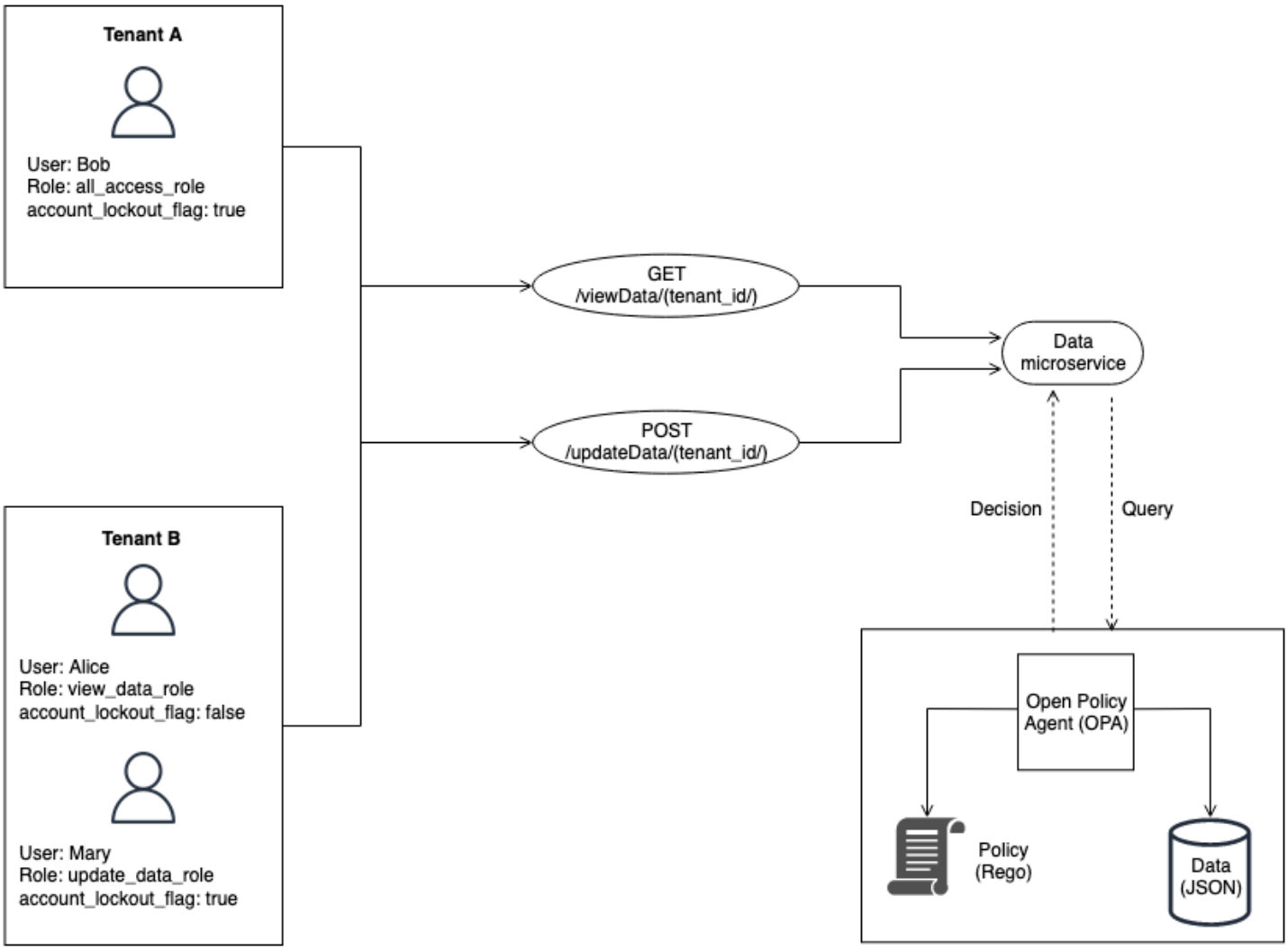
Cette entrée de requête, lorsqu'elle est évaluée avec la`allowUpdateData` règle, renvoie la décision d'autorisation suivante :

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

Cet appel ne sera pas autorisé. Bien que l'appelant de l'API soit associé à l'API correct`tenant_id` et qu'il appelle l'API en utilisant une méthode approuvée,`input.role` il s'agit de l'appelant défini par le locataire`view_data_role`. Il`view_data_role` n'en a pas l'`updateData`autorisation ; par conséquent, l'appel à`/updateData` n'est pas autorisé. Cet appel aurait été réussi pour un`tenant_b` utilisateur qui possède le`update_data_role`.

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

Pour améliorer l'exemple RBAC de 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 utilisateur`account_lockout_flag`. Il s'agit d'un attribut spécifique à l'utilisateur qui n'est associé à aucun rôle particulier. Vous pouvez utiliser les mêmes données externes RBAC 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 l'entrée 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"
}
```

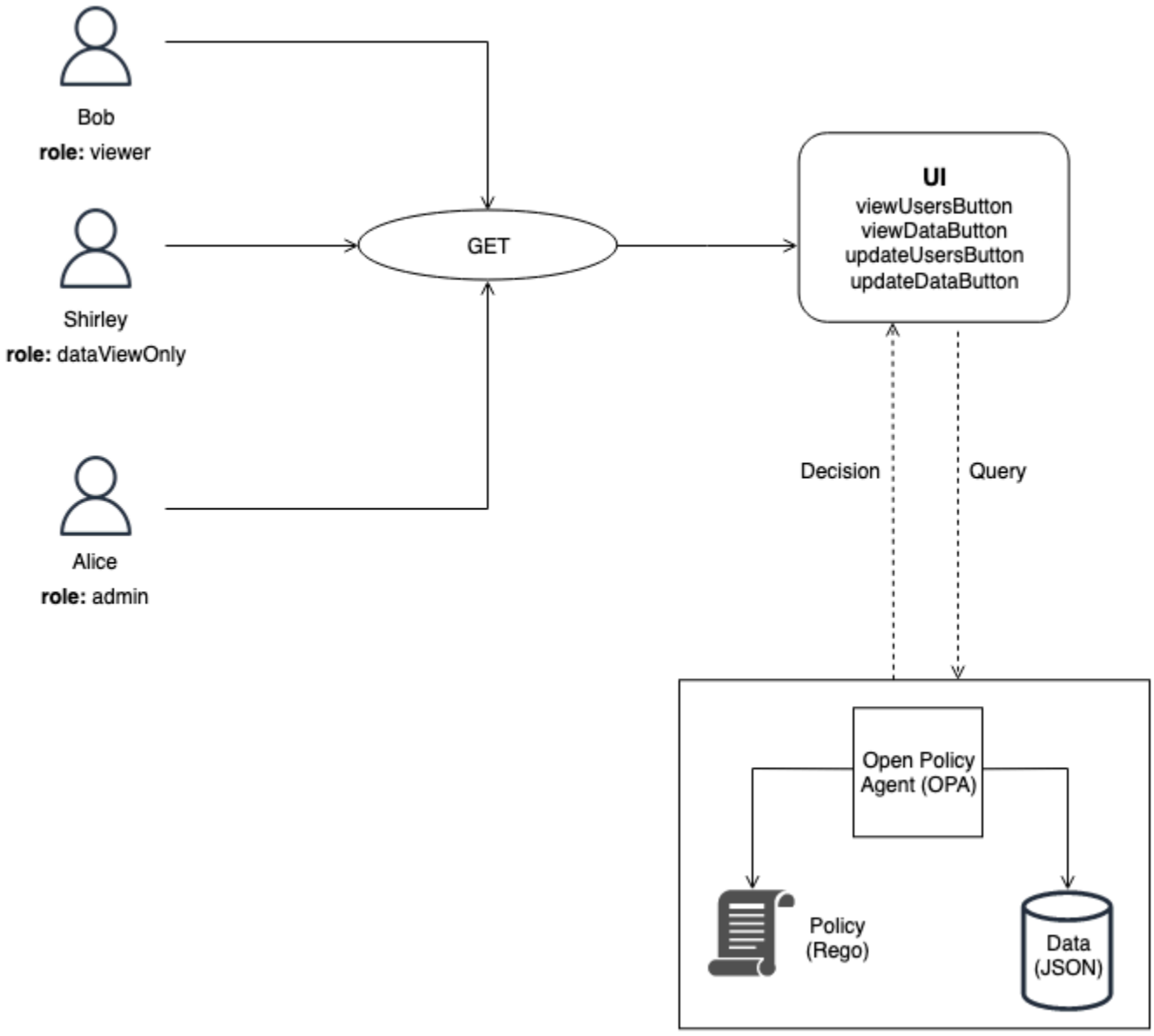
La règle qui est demandée pour la décision d'accès est similaire aux exemples précédents, mais inclut une ligne supplémentaire pour vérifier l'`account_lockout_flag`attribut :

```
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 l'`account_lockout_flag`attribut est`true` destiné à Bob et que la règle Rego`allowViewData` refuse l'accès bien que Bob ait le rôle et le locataire appropriés.

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 façons de filtrer les éléments de l'interface utilisateur avec OPA.



Dans cet exemple, une application Web d'une seule page possède 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`) fait une demande à l'interface utilisateur :

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

OPA utilise des données externes structurées pour le RBAC afin de prendre une décision d'accès :

```
{
    "roles": {
        "viewer": ["viewUsersButton", "viewDataButton"],
        "dataViewOnly": ["viewDataButton"],
        "admin": ["viewUsersButton", "viewDataButton", "updateUsersButton",
 "updateDataButton"]
    }
}
```

La règle partielle OPA utilise à la fois les données externes et les entrées pour produire un ensemble de boutons qu'un utilisateur peut visualiser sur l'interface utilisateur :

```
ui_buttons[buttons] {
    buttons := data.roles[input.role][_]
}
```

Dans la règle partielle, OPA utilise la`input.role` valeur spécifiée dans le cadre de la requête pour déterminer quels boutons doivent être affichés. Bob joue le rôle`viewer`, et les données externes indiquent que les spectateurs peuvent voir deux boutons :`viewUsersButton` et`viewDataButton`. Par conséquent, le résultat de cette règle pour Bob (et pour tout autre utilisateur ayant un rôle de spectateur) est le suivant :

```
{
    "ui_buttons": [
        "viewDataButton",
        "viewUsersButton"
    ]
}
```

La sortie pour Shirley, qui joue le`dataViewOnly` rôle, contiendrait un seul bouton :`viewDataButton`. La sortie pour Alice, qui joue le`admin` rôle, contiendrait tous les boutons. Ces réponses sont renvoyées à l'interface utilisateur lorsque l'OPA est demandé`ui_buttons`. L'interface utilisateur peut utiliser cette réponse pour masquer ou afficher les boutons en conséquence.

Utilisation d'un moteur de stratégie personnalisé

Une autre méthode pour implémenter un PDP consiste à créer un moteur de politiques personnalisé. L'objectif de ce moteur de politiques est de découpler la logique d'autorisation d'une application. Le moteur de politiques personnalisées est chargé de prendre les décisions d'autorisation, à l'instar de l'OPA, afin de réaliser le découplage des politiques. La principale différence entre cette solution et OPA réside dans le fait que la logique de rédaction et d'évaluation des politiques est personnalisée. Toute interaction avec le moteur doit être exposée par le biais d'une API ou d'une autre méthode pour permettre aux décisions d'autorisation d'atteindre une application. Vous pouvez écrire un moteur de politiques personnalisé dans n'importe quel langage de programmation ou utiliser d'autres mécanismes d'évaluation des politiques, tels que le Common Expression Language (CEL).

Implémentation d'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. Les PPE sont relativement simples par rapport aux PDP. Un PEP est uniquement chargé de demander et d'évaluer une décision d'autorisation et ne nécessite aucune logique d'autorisation. Contrairement aux PDP, les PEP ne peuvent pas être centralisés dans une application SaaS. En effet, l'autorisation et le contrôle d'accès doivent être mis en œuvre dans l'ensemble d'une application et de ses points d'accès. Les PEP peuvent être appliqués aux API, aux microservices, aux couches Backend for Frontend (BFF) ou à tout point de l'application où le contrôle d'accès est souhaité ou requis. L'omniprésence des PEP dans une application garantit que l'autorisation est vérifiée souvent et indépendamment à de multiples points.

Pour mettre en œuvre un PEP, la première étape consiste à déterminer où l'application doit être appliquée au contrôle d'accès. Tenez compte de ce principe lorsque vous décidez où les PEP doivent être intégrés dans votre application :

Si une application expose une API, cette API doit faire l'objet d'une autorisation et d'un contrôle d'accès.

En effet, dans une architecture orientée microservices ou orientée services, les API servent de séparateurs entre les différentes fonctions de l'application. Il est logique d'inclure le contrôle d'accès en tant que points de contrôle logiques entre les fonctions de l'application. Nous vous recommandons vivement d'inclure des PEP 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 dans une application. Dans les applications monolithiques, il peut être nécessaire d'intégrer les PEP dans la logique de l'application elle-même. Il n'existe pas de one-size-fits-all solution pour savoir où les PEP devraient être inclus, mais pensez à utiliser le principe de l'API comme point de départ.

Demander une 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 ou une requête d'autorisation (en utilisant les termes OPA) à une API RESTful exposée par le PDP. Il s'agit de la méthode suggérée pour intégrer des PEP à un PDP, car il s'agit d'un modèle courant qui consiste à exposer des fonctionnalités à plusieurs services dans

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

Évaluation d'une décision d'autorisation

Les PPE doivent inclure une logique pour évaluer les résultats d'une décision d'autorisation. Lorsque les PDP sont exposés en tant qu'API, 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`allow` ou`deny` d'autorisation, le PEP peut simplement vérifier cette valeur, puis renvoyer le code d'état HTTP 200 pour`allow` et le code d'état HTTP 403 pour`deny`. Ce modèle d'intégration d'un PEP comme condition préalable à l'accès à une API est un modèle facile à mettre en œuvre et très efficace pour implémenter 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 d'empaqueter 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 à locataires multiples. Ces conceptions servent de référence de haut niveau pour la mise en œuvre des points de décision politique (PDP) et des points d'application des politiques (PEP), afin de former un modèle d'autorisation cohérent et omniprésent pour les applications.

Modèles de design :

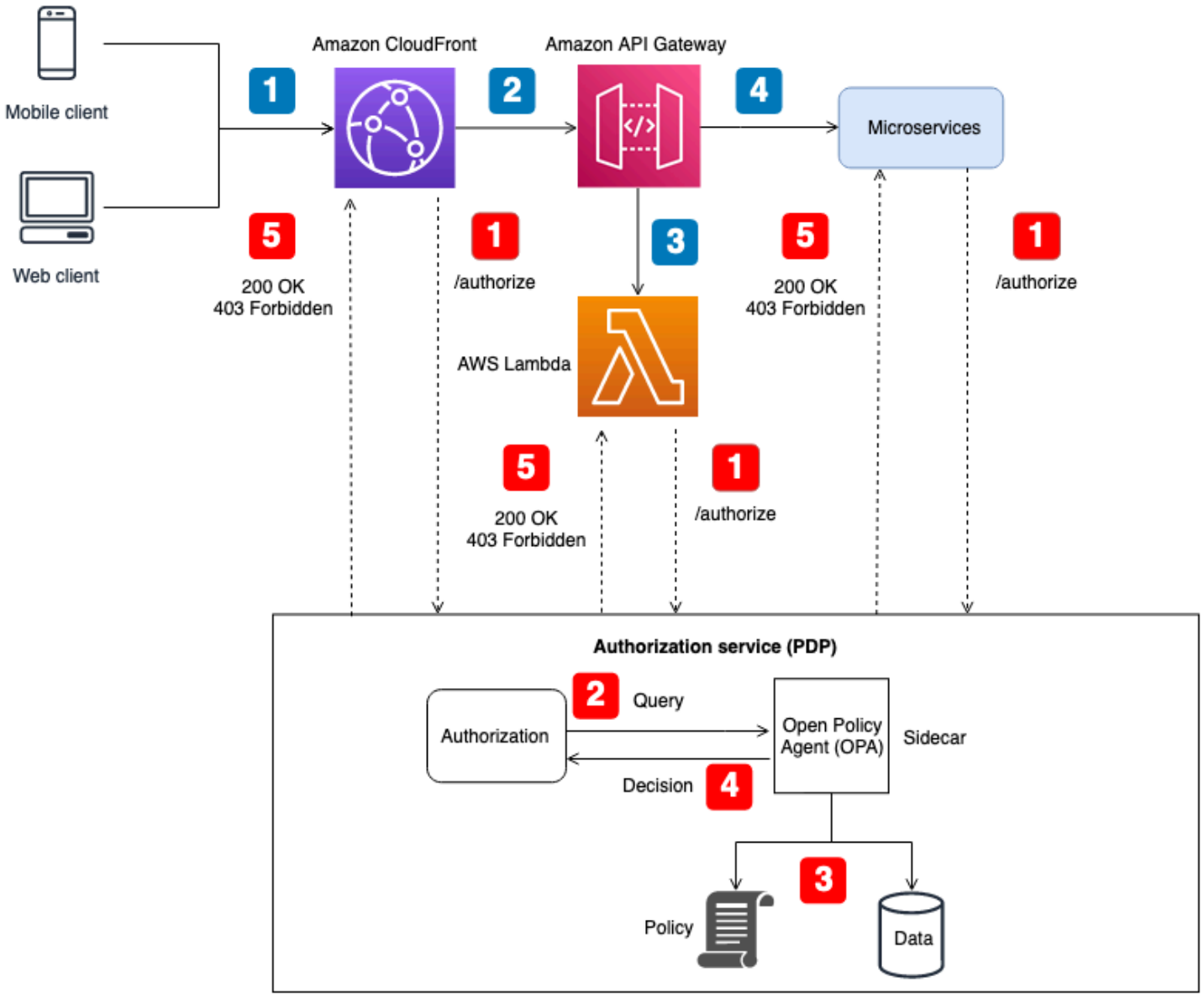
- PDP centralisé avec des PEP sur des API
- PDP distribué avec des PEP sur des API
- PDP distribué sous forme de bibliothèque

PDP centralisé avec des PEP sur des API

Le modèle de point de décision stratégique (PDP) centralisé avec points d'application des politiques (PEP) sur les API suit les meilleures pratiques du secteur pour 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. Les PEP sont implémentés dans toutes les API 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 d'applications :

1	Un utilisateur authentifié avec un jeton JWT génère une requête HTTP à Amazon. CloudFront
2	CloudFront transmet la demande à Amazon API Gateway configuré comme CloudFront origine.

3	Un autorisateur Lambda API Gateway est appelé pour vérifier le jeton JWT.
4	Les microservices répondent aux demandes.

Flux d'autorisation et de contrôle d'accès aux API :

1	Le PEP appelle le service d'autorisation et transmet les données de demande, y compris les jetons JWT.
2	Le service d'autorisation (PDP) prend les données de la demande et interroge une API REST de l'agent OPA exécutée sous forme de sidecar, les données de demande servant 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 doit être renvoyée au PEP et évaluée.

Dans cette architecture, les PEP demandent des décisions d'autorisation aux points de terminaison des services pour CloudFront API Gateway et 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 API RESTful 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 les PEP. Le fait que le service

d'autorisation fasse office d'intermédiaire entre les PEP et l'OPA permet d'insérer toute logique de transformation entre les PEP et l'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 avec des PEP sur des API constitue une option simple pour créer un système d'autorisation robuste pour les API. Il est simple à mettre en œuvre et fournit également une interface easy-to-use idempotente pour prendre des décisions d'autorisation pour les API, les microservices, les couches Backend for Frontend (BFF) ou d'autres composants d'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 est un problème, vous pouvez envisager un PDP distribué.

PDP distribué avec des PEP sur des API

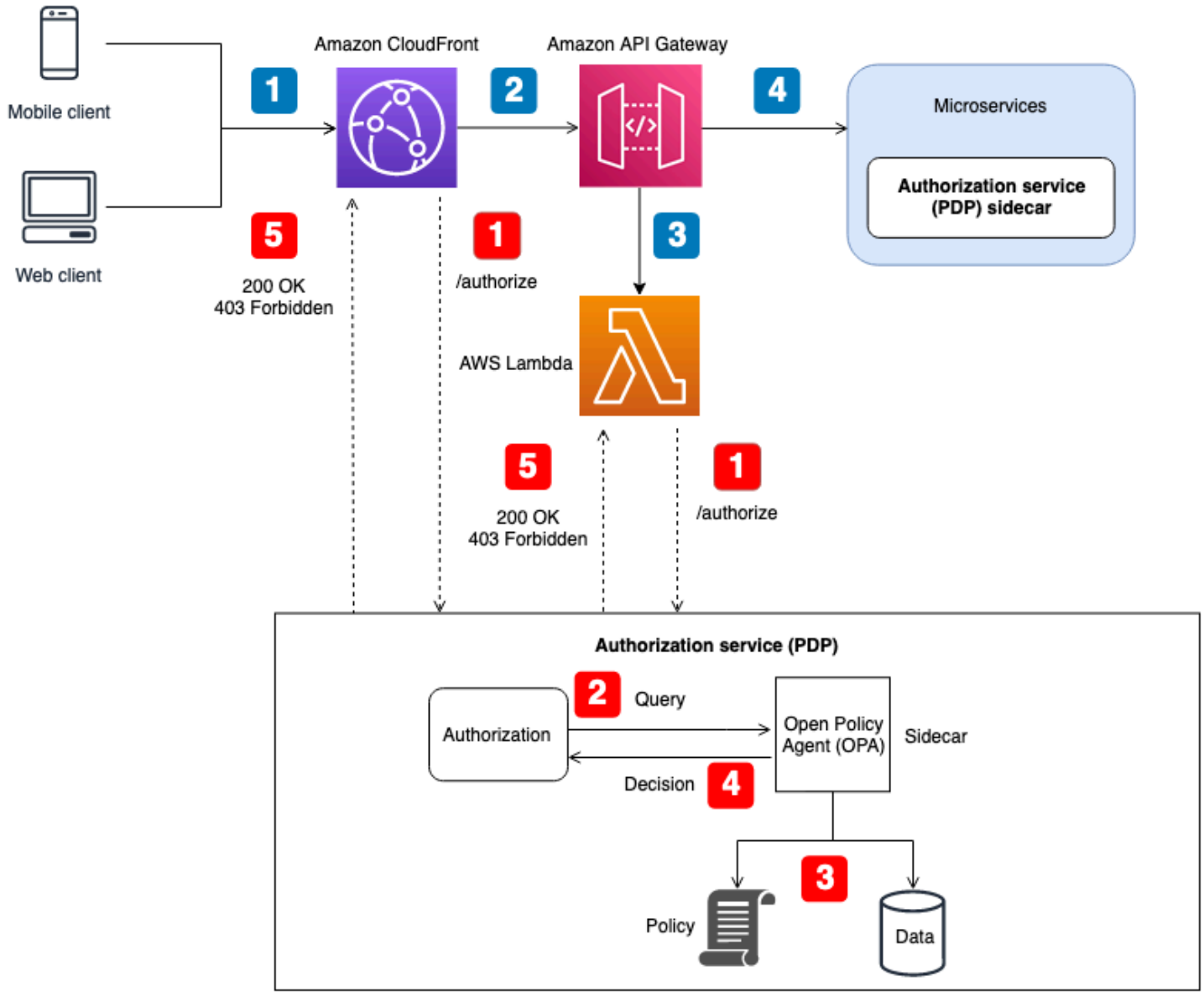
Le modèle de point de décision stratégique (PDP) distribué avec points d'application des politiques (PEP) sur les API suit les meilleures pratiques du secteur pour créer un système efficace de contrôle d'accès et d'autorisation des API. Comme pour le modèle PDP centralisé avec des PEP sur des API, cette approche prend en charge 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 en matière de contrôle d'accès. Tous les PDP fournissent les mêmes décisions de contrôle d'accès avec les mêmes entrées. Les PEP sont implémentés dans toutes les API pour envoyer des demandes d'autorisation à un PDP.

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 avoir à effectuer un appel d'API RESTful à 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 les services 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 d'applications :

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

Flux d'autorisation et de contrôle d'accès aux API :

1	Le PEP appelle le service d'autorisation et transmet les données de demande, y compris les jetons JWT.
2	Le service d'autorisation (PDP) prend les données de la demande et interroge une API REST de l'agent OPA exécutée sous forme de sidecar, les données de demande servant d'entrée à la requête.
3	L'OPA évalue l'entrée en fonction des politique s 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'autoris ation.
5	La décision d'autorisation doit être renvoyée au PEP et évaluée.

Dans cette architecture, les PEP demandent des décisions d'autorisation aux points de terminaison des services 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 des instances 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 disponible pour les PEP. Le fait que le service d'autorisation fasse office d'intermédiaire entre les PEP et l'OPA permet d'insérer toute logique de transformation entre les PEP et l'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 avec des PEP sur des API offre la possibilité de créer un système d'autorisation robuste pour les API. Il est simple à mettre en œuvre et fournit une interface easy-to-use idempotente pour prendre des décisions d'autorisation pour les API, les microservices, 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.

PDP distribué sous forme de 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 sur l'implémentation

DevOps, surveillance et journalisation

Dans ce modèle 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 réaliser le découplage des politiques, c'est-à-dire la suppression de la logique d'autorisation des autres composants de l'application. L'Open Policy Agent (OPA) fournit un mécanisme permettant de mettre à jour les politiques lorsque des modifications de la logique d'autorisation sont nécessaires. Cette fonctionnalité est proposée par une API REST simple que vous pouvez configurer pour extraire de nouvelles versions de politiques (ou ensembles) à partir d'un emplacement défini ou pour diffuser des politiques à la demande. Nous vous recommandons de créer un pipeline CI/CD robuste pour compléter un plan de contrôle pour la gestion des versions, la vérification et la mise à jour des politiques utilisées par OPA pour prendre des décisions d'autorisation. En outre, OPA propose un service de découverte de base dans lequel les nouveaux agents peuvent être configurés dynamiquement et gérés de manière centralisée par un plan de contrôle qui distribue des offres de découverte. Cette fonctionnalité permet de réduire la charge administrative liée à la gestion d'un point de décision politique distribué (PDP).

Le plan de contrôle offre également des avantages supplémentaires en matière de surveillance et d'audit. Vous pouvez surveiller l'état de l'OPA via un plan de contrôle. Peut-être plus important encore, les journaux contenant les décisions d'autorisation de l'OPA peuvent être exportés 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 efficaces de surveillance, de journalisation et de gestion des CI/CD pour intégrer de nouveaux PDP ou mettre à jour les politiques.

Extraction de données externes pour un PDP

Un PDP peut avoir besoin de données supplémentaires pour prendre une décision d'autorisation au-delà de ce qui est fourni en entrée. Par exemple, vous pouvez avoir des attributs spécifiques à l'utilisateur ou au locataire stockés dans une base de données qui doivent être référencés par un PDP afin de prendre une décision d'autorisation. Pour l'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 des données contextuelles JWT et SaaS à OPA dans le cadre de la saisie de la requête.) OPA peut accepter des entrées JSON arbitraires selon ce que l'on appelle l'approche des entrées par surcharge. Si un PDP nécessite des données autres que celles qui peuvent être incluses en entrée ou sous forme de jeton JWT, OPA propose plusieurs options pour récupérer ces données. Il s'agit notamment du regroupement, de la transmission des données (réplication) et de la récupération dynamique des données.

Regroupement

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

1. Le point d'application de la politique (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 fonction de regroupement, OPA télécharge régulièrement les ensembles de politiques et de données à partir d'un service groupé centralisé. (La mise en œuvre et la configuration d'un service groupé ne sont pas fournies par OPA.) 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 fonction de regroupement, consultez la documentation OPA.

Réplication (envoi de données)

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

1. Le point d'application de la politique (PEP) demande une décision d'autorisation.
2. Le réplicateur de données envoie les données vers OPA.
3. Le réplicateur de données réplique les données à partir de sources de données.

Dans cette alternative à l'approche groupée, les données sont transmises à l'OPA au lieu d'être extraites périodiquement par celle-ci. (L'implémentation et la configuration d'un réplicateur ne sont pas fournies par OPA.) L'approche push présente les mêmes limites de taille de données que

l'approche groupée, 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.

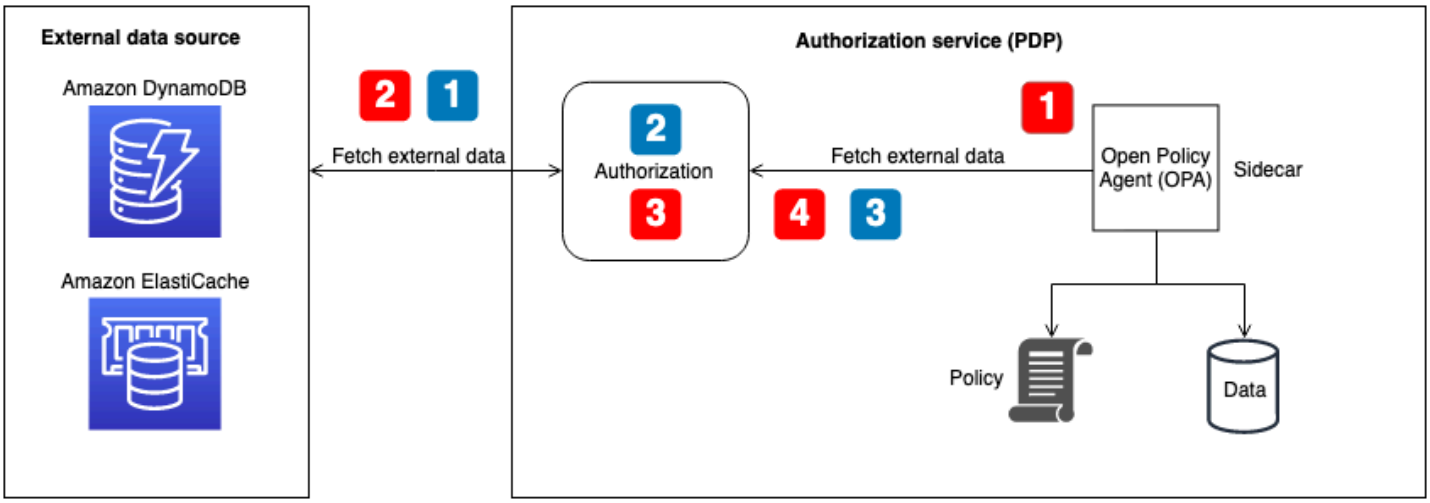
Extraction dynamique des données

Si les données externes à récupérer sont trop volumineuses pour être mises en cache dans la mémoire de l'OPA, elles 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 présentes up-to-date. Cette approche présente deux inconvénients : la latence du réseau et l'accessibilité. Actuellement, OPA ne peut récupérer des données lors de l'exécution que par le biais d'une requête HTTP. Si les appels qui renvoient à une source de données externe ne peuvent pas renvoyer de données sous forme de réponse HTTP, ils nécessitent une API personnalisée ou un autre mécanisme pour fournir ces données à OPA. Comme OPA ne peut récupérer des données que par le biais de requêtes HTTP et que la rapidité de la récupération des données est essentielle, nous vous recommandons d'utiliser unAWS service tel qu'Amazon ElastiCache ou Amazon DynamoDB pour conserver les données externes lorsque cela est possible.

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

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

Lorsque vous récupérez des données externes à l'aide du regroupement, de la réplication ou d'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 permettre à OPA de prendre des décisions d'autorisation. Le schéma suivant montre comment un service d'autorisation peut fonctionner avec ces trois approches externes de récupération de données.



Extraction de données externes pour le flux OPA — récupération de données groupées ou dynamiques au moment de la décision :

1	OPA appelle le point de terminaison de l'API local du service d'autorisation Ce point de terminaison est configuré pour servir de point de terminaison groupé ou de 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 les données. (Pour un point de terminaison groupé, ces données doivent également contenir les politiques et règles OPA. Les mises à jour groupées remplacent tout, à la fois les données et les politiques, dans le cache OPA.)
3	Le service d'autorisation effectue toute transformation nécessaire sur les données renvoyées pour les transformer en l'entrée JSON attendue.

4	Les données sont renvoyées à OPA. Il est mis en cache en mémoire pour la configuration du bundle et utilisé immédiatement pour les décisions d'autorisation dynamiques.

Extraction de données externes pour le flux OPA — réplicateur :

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 suivre une cadence définie ou se produire en réponse à des mises à jour de données dans la source externe.
2	Le service d'autorisation effectue toute transformation nécessaire sur les données renvoyées pour les transformer en l'entrée JSON attendue.
3	Le service d'autorisation appelle OPA et met en cache les données en mémoire. Le service d'autorisation peut mettre à jour de manière sélective les données, les politiques et les règles.

Recommandations pour l'isolation des locataires et la confidentialité des données externes pour le RBAC

La section précédente a fourni plusieurs approches pour importer des données externes dans OPA afin de faciliter la prise de décisions d'autorisation. Dans les cas où cela est possible, nous vous recommandons d'utiliser l'approche de saisie par surcharge pour transmettre les données

contextuelles SaaS à OPA afin de prendre des décisions d'autorisation. Toutefois, dans les modèles hybrides de contrôle d'accès basé sur les rôles (RBAC) ou de contrôle d'accès basé sur les attributs (ABAC) et de contrôle d'accès basé sur les attributs (ABAC), ces données seront souvent insuffisantes, car les rôles et les autorisations devront être référencés pour prendre des décisions d'autorisation. Afin de préserver l'isolement des locataires et la confidentialité de la cartographie des rôles, ces données ne doivent pas figurer dans 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 utilisé 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.

Certaines approches sécurisées pour préserver la confidentialité et l'isolation des locataires des données RBAC consistent à utiliser la récupération dynamique des données ou l'approche réplicateur pour obtenir des données externes pour l'OPA. En effet, le service d'autorisation illustré dans le schéma précédent peut être utilisé 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 afin de récupérer uniquement les données pertinentes pour prendre des décisions d'autorisation. En outre, dans l'approche de récupération dynamique des données, ces données n'ont pas besoin d'être mises en cache dans OPA. L'approche groupée n'est pas aussi efficace que l'approche de récupération dynamique pour maintenir l'isolation des locataires, car elle met à jour tout ce qui se trouve dans le cache OPA et ne peut pas traiter les mises à jour précises. Le modèle de regroupement reste une bonne approche pour mettre à jour les politiques de l'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 de votre entreprise, vous devez sélectionner un 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 en matière de contrôle d'accès pourraient évoluer, nécessitant des modifications de l'approche que vous avez choisie.

Mettre en œuvre un PDP

Le point de décision politique (PDP) peut être considéré comme un moteur de politiques ou de règles. Ce composant est chargé d'appliquer les politiques ou les règles et de décider si un accès particulier est autorisé. Un PDP permet de décharger la logique d'autorisation contenue dans le 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 permettant de prendre des décisions d'autorisation pour les API, les microservices, les couches Backend for Frontend (BFF) ou tout autre composant de l'application. Un PDP peut être utilisé pour appliquer les exigences de location de manière cohérente sur l'ensemble d'une application.

Implémentez des PEP pour chaque API de votre entreprise

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

Si une application expose une API, cette API doit faire l'objet d'une autorisation et d'un contrôle d'accès.

Envisagez d'utiliser l'OPA comme moteur de politiques pour votre PDP

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 permet de rédiger des 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 étayée.

Mettre en œuvre un plan de contrôle pour l'OPA pour DevOps la surveillance et la journalisation

Comme l'OPA ne permet pas 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 de mettre en œuvre 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 la charge administrative liée à l'utilisation de l'OPA. En outre, un plan de contrôle peut être utilisé pour collecter les journaux à des fins d'agrégation et pour surveiller l'état des agents OPA.

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

S'il est possible pour un PDP de prendre des décisions d'autorisation uniquement sur la base des données contenues dans un jeton Web JSON (JWT), il n'est généralement pas nécessaire d'importer des données externes pour faciliter la prise de décisions d'autorisation. Si OPA est utilisé comme PDP, il peut également accepter des données JSON arbitraires en tant qu'entrée de surcharge transmise dans le cadre de la demande, même si ces données ne sont pas incluses dans un JWT. L'utilisation d'un JWT ou de méthodes de saisie par surcharge est généralement beaucoup plus facile que de conserver des 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.

FAQ

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

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

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

Pourquoi ajouter une autorisation à mon application SaaS ?

Les applications SaaS ont plusieurs locataires. Un locataire peut être une organisation cliente ou toute entité externe qui utilise cette application SaaS. Selon la manière dont l'application est conçue, cela signifie que les locataires peuvent accéder à des API, des bases de données ou d'autres ressources partagées. Il est important de maintenir l'isolement des locataires, c'est-à-dire de séparer les autorisations et les données par locataire, afin d'empêcher les utilisateurs d'un locataire d'accéder aux informations privées d'un autre locataire. L'autorisation dans les applications SaaS est souvent conçue pour garantir que l'isolation des locataires est maintenue dans l'ensemble de l'application et que les locataires ne peuvent accéder qu'à leurs propres ressources.

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

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 correspondent étroitement à la logique métier, ou en fonction d'autres attributs contextuels tels que l'heure de la journée ou le fait qu'un utilisateur respecte 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.

Le contrôle d'accès à l'API est-il nécessaire pour mon application ?

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

Pourquoi les moteurs de politiques ou PDP sont-ils recommandés pour l'autorisation ?

Un point de décision politique (PDP) permet de décharger la logique d'autorisation contenue dans le 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 permettant de prendre des décisions d'autorisation pour les API, les microservices, les couches Backend for Frontend (BFF) ou tout autre composant de l'application.

Qu'est-ce qu'un PEP ?

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. Les PPE sont relativement simples par rapport aux PDP. Un PEP est uniquement chargé de demander et d'évaluer une décision d'autorisation et ne nécessite aucune logique d'autorisation.

Existe-t-il des alternatives open source à l'OPA ?

Il existe quelques systèmes open source similaires à l'Open Policy Agent (OPA), tels que le Common Expression Language (CEL). Ce guide se concentre sur l'OPA car il est largement adopté, documenté et adaptable à de nombreux types de demandes et d'exigences d'autorisation.

Dois-je rédiger un service d'autorisation pour utiliser OPA, ou puis-je interagir directement avec OPA ?

Vous pouvez interagir directement avec 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 demander et accepter directement les réponses OPA, il n'est pas nécessaire d'introduire cette complexité supplémentaire.

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

OPA fournit une journalisation et une surveillance de base de la disponibilité, bien que la configuration par défaut soit probablement insuffisante pour les déploiements en entreprise. Pour plus d'informations, consultez laDevOps section Surveillance et journalisation.

Quels systèmes d'exploitation etAWS services puis-je utiliser pour exécuter OPA ?

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

Puis-je lancer OPA surAWS Lambda ?

Vous pouvez exécuter OPA sur Lambda en tant que bibliothèque Go. L'article deAWS blog Création d'un autorisateur Lambda personnalisé à l'aide d'Open Policy Agent explique comment cela peut être fait pour un autorisateur Lambda API Gateway.

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

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

Puis-je utiliser OPA pour des cas d'utilisation autres que les API ?

Oui. La documentation OPA fournit des exemples pour Kubernetes, Envoy, Docker, Kafka, SSH et sudo, ainsi que Terraform. De plus, OPA est capable de renvoyer du JSON arbitraire en réponse à des requêtes en utilisant les règles partielles Rego. Selon la requête, OPA peut être utilisé pour répondre à de nombreuses questions avec des réponses JSON.

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 stratégique (PDP) et des points d'application stratégique (PEP) 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.

Étapes suivantes :

- Examinez vos besoins en matière d'autorisation et d'isolement des locataires et sélectionnez un modèle de contrôle d'accès pour votre application.
- Mettez en œuvre l'Open Policy Agent (OPA) et créez une preuve de concept à tester. (Vous pouvez également écrire votre propre moteur de stratégie personnalisé.)
- Identifiez les API et les emplacements de votre application où les PEP doivent être implémentés.

Ressources

Références

- 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 (AWSarticle 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

Outils

- 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 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
Informations clarifiées	Clarification du PDP distribué avec des PEP sur le modèle de conception des API.	10 janvier 2024
Informations ajoutées sur le nouveau AWS service	Ajout d'informations sur Amazon Verified Permissions, qui fournit les mêmes fonctionnalités et avantages que l'OPA.	22 mai 2023
—	Publication initiale	17 août 2021

Glossaire des recommandations AWS

Les termes suivants sont couramment utilisés dans les politiques, les guides et les modèles fournis par les recommandations AWS. 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 Amazon Aurora Édition compatible avec 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) for Oracle dans le Cloud AWS.
- 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 Cloud AWS.
- 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. Ce scénario de migration est propre à VMware Cloud on AWS, qui prend en charge la compatibilité des machines virtuelles (VM) et la portabilité de la charge de travail entre votre environnement sur site et AWS. Vous pouvez utiliser les technologies VMware Cloud Foundation à partir de vos centres de données sur site lorsque vous migrez votre infrastructure vers VMware Cloud on AWS. Exemple : relocalisez l'hyperviseur hébergeant votre base de données Oracle vers VMware Cloud on 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 la synchronisation des bases de données source et cible est maintenue, mais seule la base de données source gère les transactions provenant de la connexion d'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 la façon dont les AIOps sont utilisées dans la stratégie de migration AWS, veuillez consulter 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, veuillez consulter ABAC for AWS dans la documentation AWS Identity and Access Management (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

Emplacement distinct au sein d'une Région AWS qui est à l'abri des dysfonctionnements d'autres zones de disponibilité et offre une connectivité réseau peu coûteuse et de faible latence par rapport aux autres zones de disponibilité de la même région.

Framework d'adoption du Cloud AWS (AWS CAF)

Un cadre de directives et de bonnes pratiques d'AWS pour aider les entreprises à élaborer un plan efficient et efficace pour réussir leur migration vers le Cloud AWS. Le CAF organise ses conseils en six domaines prioritaires appelés perspectives : l'entreprise, les personnes, la gouvernance, la plateforme, 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, 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 Workload Qualification Framework (AWS WQF)

Outil qui évalue les charges de travail de migration de base de données, recommande des politiques de migration et fournit des estimations de travail. AWS 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

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.

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-ArchitectedAWS.

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.

CCoE

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 en local, avant que le Service AWS cible ne les reçoive.

Centre d'excellence 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, veuillez consulter les publications du CCoE sur le blog AWS Cloud Enterprise Strategy.

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 organisations traversent généralement lorsqu'elles migrent vers le Cloud AWS :

- Projet : exécution de quelques projets liés au cloud à des fins de preuve de concept et d'apprentissage
- Base : réaliser des investissements fondamentaux pour mettre à l'échelle l'adoption du cloud (par exemple, en créant une zone de destination, en définissant un CCoE ou en établissant 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 sur le blog AWS Cloud Enterprise Strategy. Pour en savoir plus sur la façon dont elles sont liées à stratégie de migration AWS, veuillez consulter 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 ouAWS CodeCommit. 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

Domaine de l'IA utilisé par les machines pour identifier des personnes, des lieux et des objets sur des images avec une précision égale ou supérieure à celle de l'être humain. Souvent conçu à partir de modèles d'apprentissage profond, il automatise l'extraction, l'analyse, la classification et la compréhension des informations utiles à partir d'une seule image ou d'une séquence d'images.

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

Une collection de règles AWS Config et d'actions correctives que vous pouvez mettre en place pour personnaliser vos contrôles de conformité et de sécurité. Vous pouvez déployer un pack de conformité en tant qu'entité unique dans un Compte AWS et une région, ou au sein d'une organisation, à l'aide d'un modèle YAML. Pour plus d'informations, veuillez consulter Conformance packs dans la documentation AWS Config.

intégration continue et livraison continue (CI/CD)

Processus d'automatisation des étapes source, de génération, de test, intermédiaire 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.

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 cadre AWS Well-Architected. 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.

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 de 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 sur AWS, vous ajoutez plusieurs contrôles à différentes couches de

la structure AWS Organizations afin de protéger 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 membre AWS 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.

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.

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 aux principaux AWS Identity and Access Management (IAM). 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).

chiffrement d'enveloppe

Processus de chiffrement d'une clé de chiffrement à l'aide d'une autre clé de chiffrement. Pour plus d'informations, veuillez consulter la rubrique Envelope encryption 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 pipeline CI/CD, 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 épopées AWS CAF en matière de sécurité comprennent la gestion des identités et des accès, les contrôles de détection, la sécurité de l'infrastructure, 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.

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 leAWS Cloud, une limite telle qu'une zone de disponibilitéRégion 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.

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.

G

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.

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 d'organisation (UO). 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 GuardDutyAWS 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.

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'environnement AWS Cloud.

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.

IIoT

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 à comptes multiples AWS, VPC qui accepte, inspecte et achemine les connexions réseau depuis l'extérieur d'une application. L'architecture de référence de sécurité AWS recommande de configurer votre compte réseau avec des VPC entrants, sortants et d'inspection afin de protéger l'interface bidirectionnelle entre votre application et 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.

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, veuillez consulter Building an industrial Internet of Things (IIoT) digital transformation strategy.

VPC d'inspection

Dans une architecture AWS à comptes multiples, VPC centralisé qui gère les inspections du trafic réseau entre des VPC (dans des Régions AWS identiques ou différentes), Internet et les réseaux sur site. L'architecture de référence de sécurité AWS recommande de configurer votre compte réseau avec des VPC entrants, sortants et d'inspection afin de protéger l'interface bidirectionnelle entre votre application et 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, veuillez consulter Machine learning model interpretability with 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 de destination est un environnement AWS à comptes multiples Well-Architected 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.

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.

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.

services gérés

Services AWSqui AWS gère 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.

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 au fur et à mesure de son fonctionnement. Pour plus d'informations, voir Création de mécanismes dans le cadre AWS Well-Architected.

compte membre

Tous les Comptes AWS autres que le 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.

microservice

Petit service indépendant qui communique via des API bien définies 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, veuillez consulter Integrating microservices by using AWS serverless services.

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 à l'aide d'API légères. 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, veuillez consulter Implémentation de microservices sur AWS.

Programme d'accélération des migrations (MAP)

Programme AWS qui fournit un support de conseil, des formations et des services pour aider les entreprises à générer 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 les opérations, les analystes commerciaux et les propriétaires, les ingénieurs de migration, les développeurs et les DevOps 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 compte AWS.

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 : réhéberger la migration vers Amazon EC2 avec AWS Application Migration Service.

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 Cloud AWS. 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 mis gratuitement à la disposition de tous les consultants AWS et consultants partenaires APN.

Évaluation de la préparation à la migration (MRA)

Processus qui consiste à obtenir des informations sur l'état de préparation au cloud d'une entreprise, à identifier les forces et les faiblesses, ainsi qu'à élaborer un plan d'action pour combler les lacunes identifiées, à l'aide d'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

Approche utilisée pour migrer une charge de travail vers le Cloud AWS. 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.

MPA

Voir Évaluation du portefeuille de migration.

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, veuillez consulter Strategy for modernizing applications in the 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, veuillez consulter Evaluating modernization readiness for applications in the 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.

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

Voir 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.

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.

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

Journal de suivi créé par AWS CloudTrail qui journalise tous les événements pour tous les Comptes AWS dans 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 migration AWS, ce cadre s'appelle accélération des personnes, 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). OAC prend en charge tous les compartiments S3 dans toutes les Régions AWS, le chiffrement côté serveur avec AWS KMS (SSE-KMS), ainsi que les demandes `PUT` et `DELETE` dynamiques adressées au compartiment S3.

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 ne peuvent accéder au contenu d'un compartiment S3 que par le biais d'une distribution spécifique CloudFront . Voir également OAC, qui fournit un contrôle d'accès plus précis et amélioré.

OU

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

VPC sortant (de sortie)

Dans une architecture AWS à comptes multiples, VPC qui gère les connexions réseau initiées depuis une application. L'architecture de référence de sécurité AWS recommande de configurer votre compte réseau avec des VPC entrants, sortants et d'inspection afin de protéger l'interface bidirectionnelle entre votre application et 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.

politique

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. Pour plus d'informations, veuillez consulter Enabling data persistence in microservices.

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

Une entité d'AWS qui peut exécuter des actions et accéder à des ressources. Cette entité est généralement un utilisateur root pour un Compte AWS, un 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 de l'ingénierie des systèmes qui prend en compte la confidentialité tout au long du processus d'ingénierie.

zones hébergées privées

Conteneur qui contient des informations concernant la façon dont vous souhaitez qu'Amazon Route 53 réponde aux requêtes DNS pour un domaine et ses sous-domaines dans un ou plusieurs VPC. 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.

environnement de production

Voir environnement.

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.

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).

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. Cela permet de déterminer ce qui est considéré comme une perte de données acceptable entre le dernier point de restauration 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

Ensemble de ressources AWS dans une zone géographique. Chaque Région AWS est isolée et indépendante des autres pour assurer la tolérance aux pannes, la stabilité et la résilience. Pour plus d'informations, veuillez consulter Managing Régions AWS dans Références générales AWS.

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.

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.

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 relatif au 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 active l'authentification unique (SSO) fédérée, permettant aux utilisateurs de se connecter à AWS Management Console ou d'appeler les opérations d'API AWS sans qu'il soit nécessaire de créer un utilisateur dans IAM pour chaque membre de l'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.

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, consultez la section Secret dans la documentation de Secrets Manager.

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 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 le Service AWS qui les reçoit.

Politique de contrôle des services (SCP)

Politique qui propose un contrôle centralisé des autorisations pour tous les comptes d'une organisation dans AWS Organizations. Les SCP définissent des barrières de protection ou des

limites aux actions qu'un administrateur peut déléguer à des utilisateurs ou à des rôles. Vous pouvez utiliser les SCP comme listes d'autorisation ou de refus, pour indiquer les services ou les actions autorisés ou interdits. Pour plus d'informations, veuillez consulter la rubrique Politiques de contrôle de service dans la documentation AWS Organizations.

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

Modèle décrivant la responsabilité que vous partagez avec AWS pour la conformité et la sécurité du cloud. AWS est responsable de la sécurité du cloud, tandis 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, consultez la section 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.

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.

T

balises

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

Hub de transit de réseau que vous pouvez utiliser pour relier vos VPC et vos réseaux sur site. Pour plus d'informations, veuillez consulter la rubrique Qu'est-ce qu'une passerelle de transit ? dans la documentation AWS Transit Gateway.

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

Octroi d'autorisations à un service que vous spécifiez pour effectuer des tâches au sein de votre organisation dans 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, veuillez consulter la rubrique Utilisation d'AWS Organizations avec d'autres services AWS dans la documentation AWS Organizations.

réglage

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

équipe de deux pizzas

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

U

incertitude

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

Pour plus d'informations, veuillez consulter le guide Quantifying uncertainty in deep learning systems.

tâches indifférenciées

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

environnements supérieurs

Voir environnement.

V

mise à vide

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

contrôle de version

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

Appairage de VPC

Connexion entre deux VPC qui vous permet d'acheminer le trafic à l'aide d'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

Consultez le cadre de qualification des charges de travail AWS.

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.

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.