Autorizzazione SaaS multi-tenant e controllo dell'accesso alle API: opzioni di implementazione e best practice

AWS Guida prescrittiva

AWS Guida prescrittiva: Autorizzazione SaaS multi-tenant e controllo dell'accesso alle API: opzioni di implementazione e best practice

Table of Contents

Autorizzazione SaaS multi-tenant e controllo dell'accesso alle API: opzioni di implementazione e best practice

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

Maggio 2024 (storia del documento)

L'autorizzazione e il controllo degli accessi alle API rappresentano una sfida per molte applicazioni software, in particolare per le applicazioni software as a service (SaaS) multi-tenant. Questa complessità è evidente se si considera la proliferazione di API di microservizi che devono essere protette e il gran numero di condizioni di accesso derivanti da tenant, caratteristiche utente e stati delle applicazioni diversi. Per affrontare questi problemi in modo efficace, una soluzione deve imporre il controllo degli accessi attraverso le numerose API presentate dai microservizi, dai livelli di Backend for Frontend (BFF) e altri componenti di un'applicazione SaaS multi-tenant. Questo approccio deve essere accompagnato da un meccanismo in grado di prendere decisioni di accesso complesse sulla base di molti fattori e attributi.

Tradizionalmente, il controllo e l'autorizzazione degli accessi alle API venivano gestiti mediante una logica personalizzata nel codice dell'applicazione. Questo approccio era soggetto a errori e non sicuro, in quanto gli sviluppatori che avevano accesso a questo codice potevano modificare accidentalmente o deliberatamente la logica di autorizzazione, con conseguente accesso non autorizzato. Il controllo delle decisioni prese mediante la logica personalizzata nel codice dell'applicazione era difficile, in quanto i revisori avrebbero dovuto immergersi nella logica personalizzata per determinarne l'efficacia nel rispetto di uno standard particolare. Inoltre, il controllo degli accessi alle API era generalmente superfluo, perché non c'erano così tante API da proteggere. Il cambio di paradigma nella progettazione delle applicazioni per favorire i microservizi e le architetture orientate ai servizi ha aumentato il numero di API che devono utilizzare una forma di autorizzazione e controllo degli accessi. Inoltre, la necessità di mantenere l'accesso basato sui tenant in un'applicazione SaaS multi-tenant presenta ulteriori sfide di autorizzazione per preservare la locazione. Le migliori pratiche descritte in questa guida offrono diversi vantaggi:

- La logica di autorizzazione può essere centralizzata e scritta in un linguaggio dichiarativo di alto livello che non è specifico di alcun linguaggio di programmazione.
- La logica di autorizzazione è astratta dal codice dell'applicazione e può essere applicata come modello ripetibile a tutte le API di un'applicazione.

- L'astrazione impedisce modifiche accidentali da parte degli sviluppatori alla logica di autorizzazione.
- L'integrazione in un'applicazione SaaS è semplice e coerente.
- L'astrazione impedisce la necessità di scrivere una logica di autorizzazione personalizzata per ogni endpoint API.
- Gli audit sono semplificati, poiché un revisore non deve più rivedere il codice per determinare le autorizzazioni.
- L'approccio delineato in questa guida supporta l'uso di più paradigmi di controllo degli accessi a seconda dei requisiti di un'organizzazione.
- Questo approccio di autorizzazione e controllo degli accessi offre un modo semplice e diretto per mantenere l'isolamento dei dati dei tenant a livello di API in un'applicazione SaaS.
- Le migliori pratiche forniscono un approccio coerente all'onboarding e all'offboarding degli inquilini per quanto riguarda l'autorizzazione.
- Questo approccio offre diversi modelli di implementazione delle autorizzazioni (in pool o in silo), che presentano vantaggi e svantaggi, come illustrato in questa guida.

Obiettivi aziendali specifici

Questa guida prescrittiva descrive modelli di progettazione ripetibili per i controlli di autorizzazione e accesso alle API che possono essere implementati per applicazioni SaaS multi-tenant. Questa guida è destinata a qualsiasi team che sviluppa applicazioni con requisiti di autorizzazione complessi o esigenze rigorose di controllo degli accessi alle API. L'architettura descrive in dettaglio la creazione di un punto di decisione delle politiche (PDP) o di un motore di politica e l'integrazione dei punti di applicazione delle politiche (PEP) nelle API. Vengono discusse due opzioni specifiche per la creazione di un PDP: utilizzare Amazon Verified Permissions con Cedar SDK e utilizzare Open Policy Agent (OPA) con il linguaggio di policy Rego. La guida illustra anche come prendere decisioni di accesso basate su un modello di controllo degli accessi basato sugli attributi (ABAC) o un modello di controllo degli accessi basato sui ruoli (RBAC) o una combinazione di entrambi i modelli. Ti consigliamo di utilizzare i modelli e i concetti di progettazione forniti in questa guida per informare e standardizzare l'implementazione dell'autorizzazione e del controllo dell'accesso alle API nelle applicazioni SaaS multi-tenant. Questa guida aiuta a raggiungere i seguenti risultati aziendali:

- Architettura di autorizzazione API standardizzata per applicazioni SaaS multi-tenant: questa architettura distingue tra tre componenti: il punto di amministrazione delle politiche (PAP) in cui vengono archiviate e gestite le politiche, il punto decisionale delle politiche (PDP) in cui tali politiche

vengono valutate per raggiungere una decisione di autorizzazione e il punto di applicazione delle politiche (PEP) che applica tale decisione. Il servizio di autorizzazione ospitato, Verified Permissions, funge sia da PAP che da PDP. In alternativa, puoi creare tu stesso il tuo PDP utilizzando un motore open source come Cedar o OPA.

- Separazione della logica di autorizzazione dalle applicazioni: la logica di autorizzazione, se incorporata nel codice dell'applicazione o implementata tramite un meccanismo di applicazione ad hoc, può essere soggetta a modifiche accidentali o dolose che causano l'accesso involontario ai dati tra tenant o altre violazioni della sicurezza. Per contribuire a mitigare queste possibilità, è possibile utilizzare un PAP, ad esempio Verified Permissions, per archiviare le politiche di autorizzazione indipendentemente dal codice dell'applicazione e applicare una solida governance alla gestione di tali politiche. Le politiche possono essere gestite centralmente in un linguaggio dichiarativo di alto livello, il che rende la gestione della logica di autorizzazione molto più semplice rispetto a quando si incorporano le politiche in più sezioni del codice dell'applicazione. Questo approccio garantisce inoltre che gli aggiornamenti vengano applicati in modo coerente.
- Approccio flessibile ai modelli di controllo degli accessi: il controllo degli accessi basato sui ruoli (RBAC), il controllo degli accessi basato sugli attributi (ABAC) o una combinazione di entrambi i modelli sono tutti approcci validi al controllo degli accessi. Questi modelli cercano di soddisfare i requisiti di autorizzazione di un'azienda utilizzando approcci diversi. Questa guida confronta e mette a confronto questi modelli per aiutarvi a scegliere un modello adatto alla vostra organizzazione. La guida illustra anche come questi modelli si applicano a diversi linguaggi di policy di autorizzazione, come OPA/Rego e Cedar. Le architetture illustrate in questa guida consentono di adottare con successo uno o entrambi i modelli.
- Controllo rigoroso dell'accesso alle API: questa guida fornisce un metodo per proteggere le API in modo coerente e pervasivo in un'applicazione con il minimo sforzo. Ciò è particolarmente utile per le architetture applicative orientate ai servizi o ai microservizi che generalmente utilizzano un gran numero di API per facilitare le comunicazioni tra applicazioni. Il rigoroso controllo degli accessi alle API aiuta ad aumentare la sicurezza di un'applicazione e la rende meno vulnerabile agli attacchi o allo sfruttamento.

Isolamento dei tenant e autorizzazione multi-tenant

Questa guida fa riferimento ai concetti di isolamento dei tenant e di autorizzazione multi-tenant. L'isolamento dei tenant si riferisce ai meccanismi espliciti utilizzati in un sistema SaaS per garantire che le risorse di ciascun tenant, anche quando operano su un'infrastruttura condivisa, siano isolate. L'autorizzazione multi-tenant si riferisce all'autorizzazione delle azioni in entrata e alla

prevenzione che vengano implementate sul tenant sbagliato. Un utente ipotetico potrebbe essere autenticato e autorizzato e continuare ad accedere alle risorse di un altro tenant. L'autenticazione e l'autorizzazione non bloccheranno questo accesso: è necessario implementare l'isolamento dei tenant per raggiungere questo obiettivo. Per una discussione più approfondita delle differenze tra questi due concetti, consulta la sezione sull'isolamento dei tenant del white paper SaaS Architecture Fundamentals.

Tipi di controllo degli accessi

È possibile utilizzare due modelli ampiamente definiti per implementare il controllo degli accessi: controllo degli accessi basato sui ruoli (RBAC) e controllo degli accessi basato sugli attributi (ABAC). Ciascun modello presenta vantaggi e svantaggi, descritti brevemente in questa sezione. Il modello da utilizzare dipende dal caso d'uso specifico. L'architettura descritta in questa guida supporta entrambi i modelli.

RBAC

Il controllo degli accessi basato sul ruolo (RBAC) determina l'accesso alle risorse in base a un ruolo che di solito è in linea con la logica aziendale. Le autorizzazioni sono associate al ruolo a seconda dei casi. Ad esempio, un ruolo di marketing autorizzerebbe un utente a svolgere attività di marketing all'interno di un sistema limitato. Si tratta di un modello di controllo degli accessi relativamente semplice da implementare perché si allinea bene a una logica aziendale facilmente riconoscibile.

Il modello RBAC è meno efficace quando:

- Hai utenti unici le cui responsabilità comprendono diversi ruoli.
- Avete una logica aziendale complessa che rende difficile la definizione dei ruoli.
- Il passaggio a grandi dimensioni richiede un'amministrazione e una mappatura costanti delle autorizzazioni per ruoli nuovi ed esistenti.
- Le autorizzazioni si basano su parametri dinamici.

ABAC

Il controllo degli accessi basato sugli attributi (ABAC) determina l'accesso alle risorse in base agli attributi. Gli attributi possono essere associati a un utente, a una risorsa, a un ambiente o persino allo stato dell'applicazione. I criteri o le regole fanno riferimento agli attributi e possono utilizzare la logica booleana di base per determinare se un utente è autorizzato a eseguire un'azione. Ecco un esempio di base di autorizzazioni:

Nel sistema di pagamento, tutti gli utenti del reparto finanziario possono elaborare i pagamenti presso l'endpoint API `/payments` durante l'orario lavorativo.

L'appartenenza al dipartimento finanziario è un attributo utente che determina l'accesso a`/payments`. Esiste anche un attributo di risorsa associato all'endpoint `/payments` API che consente l'accesso solo durante l'orario lavorativo. In ABAC, la capacità di un utente di elaborare o meno un pagamento è determinata da una politica che include l'appartenenza al dipartimento finanziario come attributo utente e l'ora come attributo di risorsa di. `/payments`

Il modello ABAC è molto flessibile nel consentire decisioni di autorizzazione dinamiche, contestuali e granulari. Tuttavia, il modello ABAC è difficile da implementare inizialmente. La definizione di regole e politiche, nonché l'enumerazione degli attributi per tutti i vettori di accesso pertinenti, richiedono un investimento iniziale significativo per l'implementazione.

Approccio ibrido RBAC-ABAC

La combinazione di RBAC e ABAC può offrire alcuni dei vantaggi di entrambi i modelli. RBAC, essendo così strettamente allineato alla logica aziendale, è più semplice da implementare rispetto all'ABAC. Per fornire un ulteriore livello di granularità quando si prendono decisioni di autorizzazione, è possibile combinare ABAC con RBAC. Questo approccio ibrido determina l'accesso combinando il ruolo dell'utente (e le autorizzazioni assegnate) con attributi aggiuntivi per prendere decisioni di accesso. L'utilizzo di entrambi i modelli consente una semplice amministrazione e assegnazione delle autorizzazioni, consentendo al contempo una maggiore flessibilità e granularità relative alle decisioni di autorizzazione.

Confronto dei modelli di controllo degli accessi

La tabella seguente mette a confronto i tre modelli di controllo degli accessi discussi in precedenza. Questo confronto vuole essere informativo e di alto livello. L'utilizzo di un modello di accesso in una situazione specifica potrebbe non essere necessariamente correlato ai confronti effettuati in questa tabella.

Fattore	RBAC	ABAC	Ibrido
Flessibilità	Media	Elevata	Elevata
Semplicità	Elevata	Bassa	Media
Granularity (Granularità)	Bassa	Elevata	Media

Decisioni e regole dinamiche	No	Sì	Sì
Consapevole al contesto	No	Sì	Un po'
Sforzo di implement azione	Bassa	Elevata	Media

Implementazione di un PDP

Il punto di decisione politica (PDP) può essere caratterizzato come un motore di politiche o regole. Questo componente è responsabile dell'applicazione di politiche o regole e della decisione se consentire un determinato accesso. Un PDP può funzionare con modelli di controllo degli accessi basato sui ruoli (RBAC) e di controllo degli accessi basato sugli attributi (ABAC); tuttavia, un PDP è un requisito per ABAC. Un PDP consente di scaricare la logica di autorizzazione nel codice dell'applicazione su un sistema separato. Questo può semplificare il codice dell'applicazione. Fornisce inoltre un'interfaccia easy-to-use ripetibile per prendere decisioni di autorizzazione per API, microservizi, livelli di Backend for Frontend (BFF) o qualsiasi altro componente dell'applicazione.

Le sezioni seguenti illustrano tre metodi per implementare un PDP. Tuttavia, questo non è un elenco completo.

Metodi di implementazione PDP:

- Implementazione di un PDP utilizzando Amazon Verified Permissions
- Implementazione di un PDP utilizzando OPA
- Utilizzo di un motore di policy personalizzato

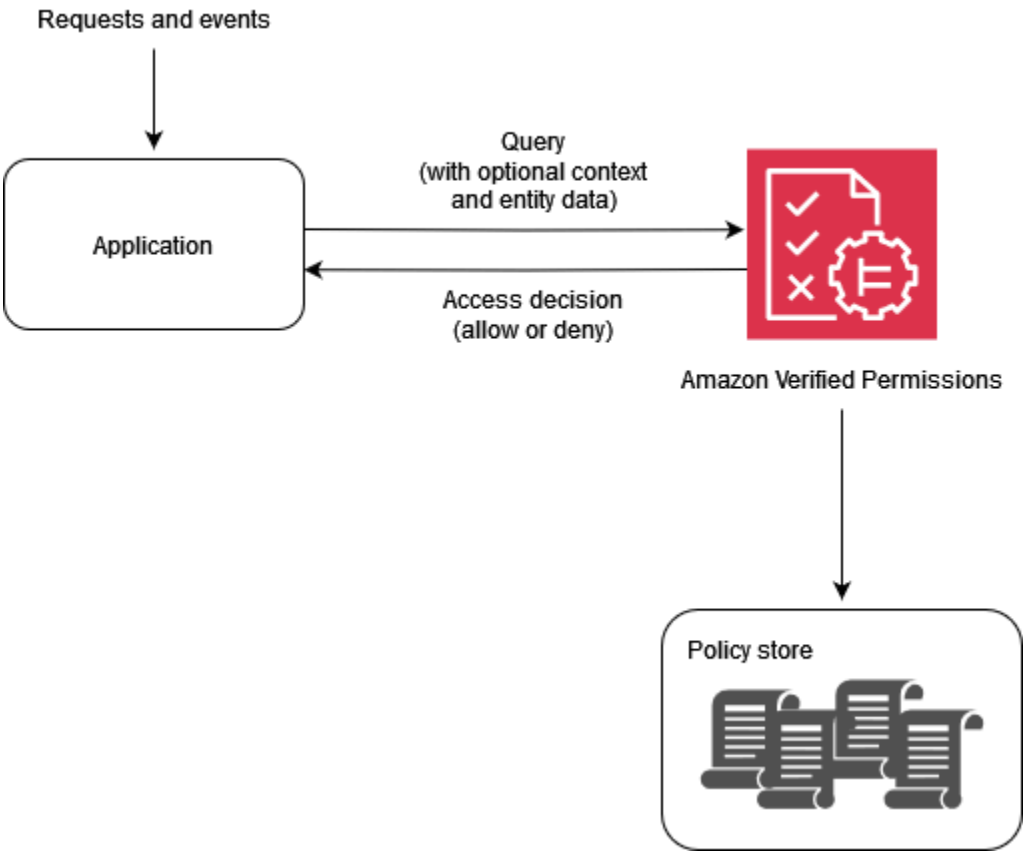
Implementazione di un PDP utilizzando Amazon Verified Permissions

Amazon Verified Permissions è un servizio di gestione e autorizzazione scalabile e granulare delle autorizzazioni che puoi utilizzare per implementare un punto decisionale sulle politiche (PDP). In quanto motore di policy, può aiutare l'applicazione a verificare le azioni degli utenti in tempo reale ed evidenziare le autorizzazioni eccessivamente privilegiate o non valide. Aiuta gli sviluppatori a creare applicazioni più sicure più rapidamente esternalizzando le autorizzazioni e centralizzando la gestione e l'amministrazione delle policy. Separando la logica di autorizzazione dalla logica dell'applicazione, Verified Permissions supporta il disaccoppiamento delle politiche.

Utilizzando le autorizzazioni verificate per implementare un PDP e implementando i privilegi minimi e la verifica continua all'interno delle applicazioni, gli sviluppatori possono allineare l'accesso alle applicazioni ai principi Zero Trust. Inoltre, i team di sicurezza e controllo possono analizzare e verificare meglio chi ha accesso a quali risorse all'interno di un'applicazione. Verified Permissions utilizza Cedar, un linguaggio di policy open source creato appositamente e incentrato sulla sicurezza,

per definire controlli di accesso basati su policy basati sul controllo degli accessi basato sui ruoli (RBAC) e sul controllo degli accessi basato sugli attributi (ABAC) per un controllo degli accessi più granulare e sensibile al contesto.

Verified Permissions offre alcune funzioni utili per le applicazioni SaaS, come la possibilità di abilitare l'autorizzazione multi-tenant utilizzando più provider di identità come Amazon Cognito, Google e Facebook. Un'altra funzionalità di Autorizzazioni verificate particolarmente utile per le applicazioni SaaS è il supporto per ruoli personalizzati su base per-tenant. Se state progettando un sistema di gestione delle relazioni con i clienti (CRM), un tenant potrebbe definire la granularità dell'accesso in base alle opportunità di vendita sulla base di un particolare insieme di criteri. Un altro inquilino potrebbe avere un'altra definizione. I sistemi di autorizzazioni alla base di Verified Permissions possono supportare queste variazioni, il che lo rende un ottimo candidato per i casi d'uso SaaS. Verified Permissions supporta anche la possibilità di scrivere policy che si applicano a tutti i tenant, quindi è semplice applicare policy guardrail per impedire accessi non autorizzati come provider SaaS.



Perché usare le autorizzazioni verificate?

Utilizza le autorizzazioni verificate con un provider di identità come Amazon Cognito per una soluzione di gestione degli accessi più dinamica e basata su policy per le tue applicazioni. Puoi

creare applicazioni che aiutano gli utenti a condividere informazioni e collaborare mantenendo la sicurezza, la riservatezza e la privacy dei loro dati. Verified Permissions aiuta a ridurre i costi operativi fornendovi un sistema di autorizzazione dettagliato per imporre l'accesso in base ai ruoli e agli attributi delle vostre identità e risorse. Puoi definire il tuo modello di policy, creare e archiviare le policy in una posizione centrale e valutare le richieste di accesso in millisecondi.

In Autorizzazioni verificate, puoi esprimere le autorizzazioni utilizzando un linguaggio dichiarativo semplice e leggibile dall'uomo chiamato Cedar. Le politiche scritte in Cedar possono essere condivise tra i team indipendentemente dal linguaggio di programmazione utilizzato dall'applicazione di ciascun team.

Cosa considerare quando si utilizzano le autorizzazioni verificate

In Autorizzazioni verificate, puoi creare politiche e automatizzarle come parte del provisioning. È inoltre possibile creare policy in fase di esecuzione come parte della logica dell'applicazione. Come best practice, è consigliabile utilizzare una pipeline di integrazione e distribuzione continua (CI/CD) per amministrare, modificare e tenere traccia delle versioni delle policy quando si creano le policy come parte dell'onboarding e del provisioning dei tenant. In alternativa, un'applicazione può amministrare, modificare e tenere traccia delle versioni delle policy; tuttavia, la logica dell'applicazione non esegue intrinsecamente questa funzionalità. Per supportare queste funzionalità nell'applicazione, è necessario progettare esplicitamente l'applicazione in modo da implementare questa funzionalità.

Se è necessario fornire dati esterni provenienti da altre fonti per prendere una decisione di autorizzazione, tali dati devono essere recuperati e forniti a Verified Permissions come parte della richiesta di autorizzazione. Il contesto, le entità e gli attributi aggiuntivi non vengono recuperati per impostazione predefinita con questo servizio.

Panoramica su Cedar

Cedar è un linguaggio di controllo degli accessi flessibile, estensibile e scalabile basato su policy che aiuta gli sviluppatori a esprimere le autorizzazioni delle applicazioni come politiche. Gli amministratori e gli sviluppatori possono definire politiche che consentono o vietano agli utenti di agire sulle risorse delle applicazioni. È possibile allegare più politiche a una singola risorsa. Quando un utente dell'applicazione tenta di eseguire un'azione su una risorsa, l'applicazione richiede l'autorizzazione al motore di policy Cedar. Cedar valuta le politiche applicabili e restituisce una `ALLOW` decisione or. `DENY` Cedar supporta le regole di autorizzazione per qualsiasi tipo di principale e risorsa, consente il controllo degli accessi basato sui ruoli (RBAC) e il controllo degli accessi basato sugli attributi (ABAC) e supporta l'analisi tramite strumenti di ragionamento automatizzati.

Cedar consente di separare la logica aziendale dalla logica di autorizzazione. Quando si effettuano richieste dal codice dell'applicazione, si chiama il motore di autorizzazione di Cedar per determinare se la richiesta è autorizzata. Se è autorizzata (la decisione è`ALLOW`), l'applicazione può eseguire l'operazione richiesta. Se non è autorizzata (la decisione è`DENY`), l'applicazione può restituire un messaggio di errore. Le caratteristiche principali di Cedar includono:

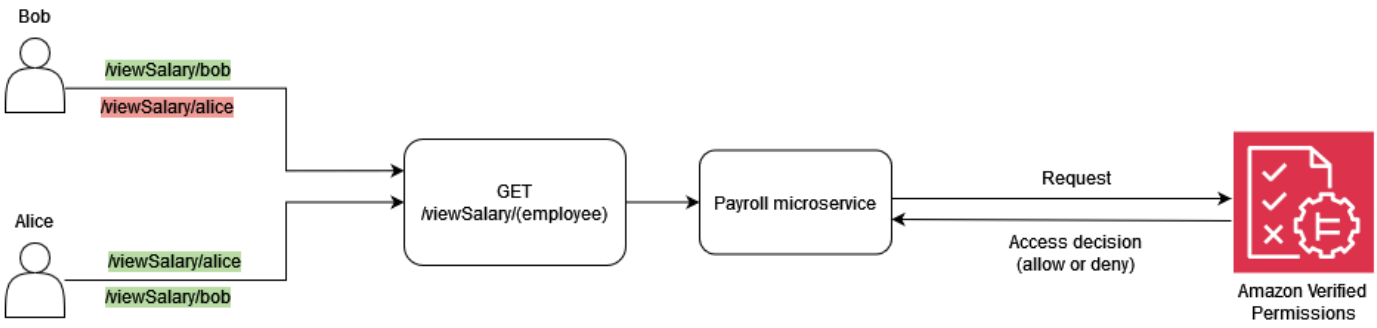
- Espressività: Cedar è stato progettato appositamente per supportare i casi d'uso delle autorizzazioni ed è stato sviluppato pensando alla leggibilità umana.
- Prestazioni: Cedar supporta le politiche di indicizzazione per un recupero rapido e fornisce una valutazione in tempo reale rapida e scalabile con latenza limitata.
- Analisi: Cedar supporta strumenti di analisi in grado di ottimizzare le politiche e verificare il modello di sicurezza.

Per ulteriori informazioni, consultate il sito Web di Cedar.

Esempio 1: ABAC di base con autorizzazioni verificate e Cedar

In questo scenario di esempio, Amazon Verified Permissions viene utilizzato per determinare a quali utenti è consentito accedere alle informazioni in un microservizio Payroll fittizio. Questa sezione include frammenti di codice Cedar per dimostrare come utilizzare Cedar per prendere decisioni sul controllo degli accessi. Questi esempi non intendono fornire un'esplorazione completa delle funzionalità fornite da Cedar e Verified Permissions. Per una panoramica più completa di Cedar, consulta la documentazione di Cedar.

Nel diagramma seguente, vorremmo applicare due regole aziendali generali associate al `viewSalary` `GET` metodo: i dipendenti possono visualizzare il proprio stipendio e i dipendenti possono visualizzare lo stipendio di chiunque faccia loro capo. È possibile applicare queste regole aziendali utilizzando le politiche di autorizzazione verificate.



I dipendenti possono visualizzare il proprio stipendio.

In Cedar, il costrutto di base è un'entità, che rappresenta un principale, un'azione o una risorsa. Per effettuare una richiesta di autorizzazione e avviare una valutazione con una politica di autorizzazioni verificate, è necessario fornire un responsabile, un'azione, una risorsa e un elenco di entità.

- Il principal (`principal`) è l'utente o il ruolo registrato.
- L'azione (`action`) è l'operazione che viene valutata dalla richiesta.
- La risorsa (`resource`) è il componente a cui l'azione accede.
- L'elenco delle entità (`entityList`) contiene tutte le entità richieste necessarie per valutare la richiesta.

Per soddisfare la regola aziendale, i dipendenti possono visualizzare il proprio stipendio, è possibile fornire una politica di autorizzazioni verificate come la seguente.

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

Questa politica valuta `ALLOW` se la risorsa `Action` is `viewSalary` e la risorsa inclusa nella richiesta hanno un proprietario dell'attributo uguale al principale. Ad esempio, se Bob è l'utente connesso che ha richiesto il rapporto sullo stipendio ed è anche il proprietario del rapporto sullo stipendio, la politica restituisce lo stesso. `ALLOW`

La seguente richiesta di autorizzazione viene inviata a Verified Permissions per essere valutata in base alla politica di esempio. In questo esempio, Bob è l'utente connesso che effettua la richiesta. `viewSalary` Pertanto, Bob è il principale del tipo `Employee` di entità. L'azione che Bob sta cercando di eseguire è `viewSalary,` e la risorsa che `viewSalary` verrà visualizzata è `Salary-Bob` del tipo`Salary`. Per valutare se Bob può visualizzare la `Salary-Bob` risorsa, è necessario fornire una struttura di entità che colleghi il tipo `Employee` con un valore pari a `Bob` (principale) all'attributo proprietario della risorsa che ha il tipo`Salary`. Fornisci questa struttura in un formato`entityList`, in cui gli attributi associati a `Salary` includono un proprietario, che specifica e `entityIdentifier` che contiene il tipo `Employee` e il valore`Bob`. Verified Permissions confronta

quanto `principal` fornito nella richiesta di autorizzazione con l'`owner`attributo associato alla `Salary` risorsa per prendere una decisione.

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

```
}
```

La richiesta di autorizzazione a Verified Permissions restituisce quanto segue come output, dove l'attributo `decision` è o. `ALLOW` `DENY`

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

In questo caso, poiché Bob stava cercando di visualizzare il proprio stipendio, la richiesta di autorizzazione inviata a Verified Permissions corrisponde a. `ALLOW` Tuttavia, il nostro obiettivo era utilizzare le autorizzazioni verificate per applicare due regole aziendali. Anche la regola aziendale che afferma quanto segue dovrebbe essere vera:

I dipendenti possono visualizzare lo stipendio di chiunque risponda a loro.

Per soddisfare questa regola aziendale, puoi fornire un'altra politica. La seguente politica valuta `ALLOW` se l'azione è `viewSalary` e se la risorsa nella richiesta ha un attributo `owner.manager` uguale al principale. Ad esempio, se Alice è l'utente connesso che ha richiesto il rapporto sullo stipendio e Alice è la responsabile del proprietario del rapporto, la politica restituisce lo stesso risultato. `ALLOW`

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

La seguente richiesta di autorizzazione viene inviata a Verified Permissions per essere valutata in base alla politica di esempio. In questo esempio, Alice è l'utente connesso che effettua la richiesta.

`viewSalary` Quindi Alice è la principale e l'entità è del tipo`Employee`. L'azione che Alice sta cercando di eseguire è`viewSalary`, e la risorsa che `viewSalary` verrà visualizzata è del tipo `Salary` con un valore di`Salary-Bob`. Per valutare se Alice può visualizzare la `Salary-Bob` risorsa, è necessario fornire una struttura di entità che colleghi il tipo `Employee` con un valore pari `Alice` all'`manager`attributo, che deve quindi essere associato all'`owner`attributo del tipo `Salary` con un valore di`Salary-Bob`. Fornite questa struttura in un formato`entityList`, in cui gli attributi associati a `Salary` includono un proprietario, che specifica e `entityIdentifier` che contiene il tipo `Employee` e il valore`Bob`. Verified Permissions controlla innanzitutto l'`owner`attributo, che restituisce il tipo `Employee` e il valore. `Bob` Quindi, Verified Permissions valuta l'`manager`attributo a cui è associato `Employee` e lo confronta con il principale fornito per prendere una decisione di autorizzazione. In questo caso, la decisione è `ALLOW` dovuta al fatto che `resource.owner.manager` gli attributi `principal` and sono equivalenti.

```
{
  "policyStoreId": "PAYROLLAPP_POLICYSTOREID",
  "principal": {
    "entityType": "PayrollApp::Employee",
    "entityId": "Alice"
  },
  "action": {
    "actionType": "PayrollApp::Action",
    "actionId": "viewSalary"
  },
  "resource": {
    "entityType": "PayrollApp::Salary",
    "entityId": "Salary-Bob"
  },
  "entities": {
    "entityList": [
      {
        "identifier": {
          "entityType": "PayrollApp::Employee",
          "entityId": "Alice"
        },
        "attributes": {
          "manager": {
            "entityIdentifier": {
              "entityType": "PayrollApp::Employee",
              "entityId": "None"
            }
          }
```

```
        },
        "parents": []
      },
      {
        "identifier": {
          "entityType": "PayrollApp::Salary",
          "entityId": "Salary-Bob"
        },
        "attributes": {
          "owner": {
            "entityIdentifier": {
              "entityType": "PayrollApp::Employee",
              "entityId": "Bob"
            }
          }
        },
        "parents": []
      },
      {
        "identifier": {
          "entityType": "PayrollApp::Employee",
          "entityId": "Bob"
        },
        "attributes": {
          "manager": {
            "entityIdentifier": {
              "entityType": "PayrollApp::Employee",
              "entityId": "Alice"
            }
          }
        },
       "parents": []
      }
    ]
  }
}
```

Finora, in questo esempio, abbiamo fornito le due regole aziendali associate al `viewSalary` metodo, vale a dire che i dipendenti possono visualizzare il proprio stipendio e i dipendenti possono visualizzare lo stipendio di chiunque risponda a loro, utilizzando le autorizzazioni verificate come politiche per soddisfare le condizioni di ogni regola aziendale in modo indipendente. Puoi anche

utilizzare un'unica politica di autorizzazioni verificate per soddisfare le condizioni di entrambe le regole aziendali:

I dipendenti possono visualizzare il proprio stipendio e quello di chiunque risponda a loro.

Quando si utilizza la richiesta di autorizzazione precedente, la seguente politica valuta `ALLOW` se l'azione è `viewSalary` e se la risorsa inclusa nella richiesta ha un attributo `owner.manager` uguale a o un attributo `owner` uguale a. `principal` `principal`

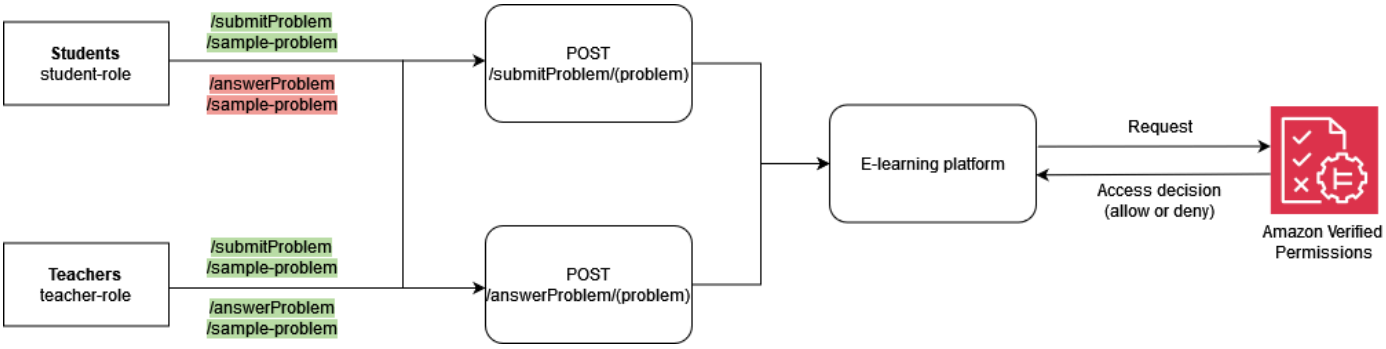
```
permit (
    principal,
    action == PayrollApp::Action::"viewSalary",
    resource
)
when {
    principal == resource.owner.manager ||
    principal == resource.owner
};
```

Ad esempio, se Alice è l'utente connesso che richiede il rapporto sullo stipendio e se Alice è la responsabile del proprietario o la proprietaria del rapporto, la politica restituisce tale rapporto. `ALLOW`

Per ulteriori informazioni sull'utilizzo degli operatori logici con le politiche Cedar, consultate la documentazione di Cedar.

Esempio 2: RBAC di base con autorizzazioni verificate e Cedar

Questo esempio utilizza Verified Permissions e Cedar per dimostrare l'RBAC di base. Come accennato in precedenza, il costrutto base di Cedar è un'entità. Gli sviluppatori definiscono le proprie entità e possono facoltativamente creare relazioni tra le entità. L'esempio seguente include tre tipi di entità: `UsersRoles`, e`Problems`. `Studentse` `Teachers` possono essere considerate entità del tipo `Role`, e ciascuna `User` può essere associata a zero o a uno qualsiasi dei`Roles`.



In Cedar, queste relazioni vengono espresse collegando il a come elemento principale. `Role` `Student` `User` Bob Questa associazione raggruppa logicamente tutti gli utenti studenti in un unico gruppo. Per ulteriori informazioni sul raggruppamento in Cedar, consulta la documentazione di Cedar.

La seguente politica valuta la decisione relativa all'azione `ALLOW` `submitProblem`, per tutti i principali collegati al gruppo logico del tipo. `Students` `Role`

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

La seguente politica si riferisce alla decisione `ALLOW` relativa all'azione `submitProblem` o `answerProblem` a tutti i principali collegati al gruppo `Teachers` logico del tipo. `Role`

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

Per valutare le richieste con queste politiche, il motore di valutazione deve sapere se il principale a cui si fa riferimento nella richiesta di autorizzazione fa parte del gruppo appropriato. Pertanto, l'applicazione deve trasmettere le informazioni pertinenti sull'appartenenza al gruppo al motore di valutazione come parte della richiesta di autorizzazione. Ciò avviene tramite la `entities` proprietà, che consente di fornire al motore di valutazione Cedar i dati sugli attributi e sull'appartenenza al gruppo per il principale e la risorsa coinvolte nella chiamata di autorizzazione. Nel codice seguente, l'appartenenza al gruppo viene indicata definendo `User::"Bob"` come la chiamata `Role::"Students"` di un genitore.

```
{
  "policyStoreId": "ELEARNING_POLICYSTOREID",
  "principal": {
    "entityType": "ElearningApp::User",
    "entityId": "Bob"
  },
```

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

In questo esempio, Bob è l'utente connesso che effettua la `answerProblem` richiesta. Pertanto, Bob è il principale e l'entità è del tipo`User`. L'azione che Bob sta cercando di eseguire è`answerProblem`. Per valutare se Bob è in grado di eseguire l'`answerProblem`azione, è necessario fornire una struttura di entità che colleghi l'entità `User` con un valore di `Bob` e le assegni l'appartenenza al gruppo elencando un'entità principale come`Role::"Students"`. Poiché le entità del gruppo di utenti `Role::"Students"` sono autorizzate solo a eseguire l'azione`submitProblem`, questa richiesta di autorizzazione restituisce un risultato positivo. `DENY`

D'altra parte, se il tipo `User` che ha un valore pari a `Alice` e fa parte del gruppo `Role::"Teachers"` tenta di eseguire l'`answerProblem`azione, la richiesta di autorizzazione restituisce lo stesso risultato`ALLOW`, poiché la politica impone che i responsabili del gruppo `Role::"Teachers"` siano autorizzati a eseguire l'azione `answerProblem` su tutte le risorse. Il codice seguente mostra questo tipo di richiesta di autorizzazione che restituisce a. `ALLOW`

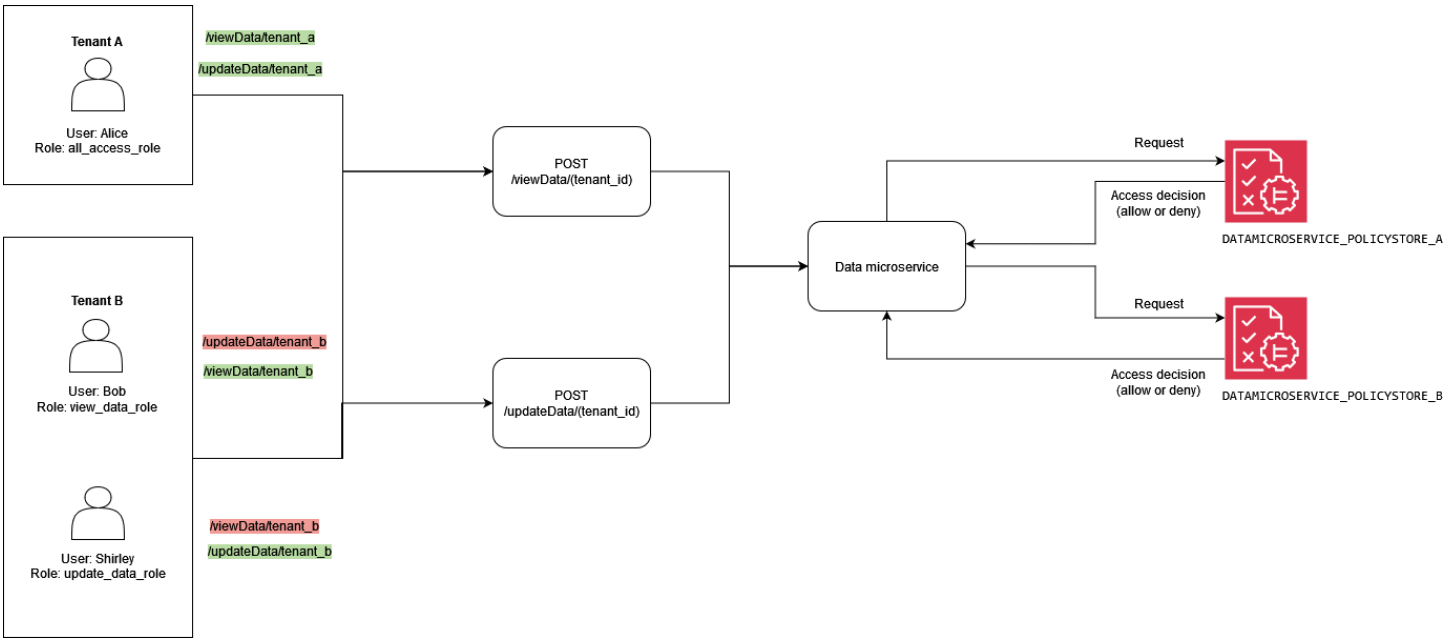
```
{
  "policyStoreId": "ELEARNING_POLICYSTOREID",
  "principal": {
    "entityType": "ElearningApp::User",
    "entityId": "Alice"
  },
  "action": {
    "actionType": "ElearningApp::Action",
    "actionId": "answerProblem"
  },
  "resource": {
    "entityType": "ElearningApp::Problem",
    "entityId":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

```
        }
      ]
  }
}
```

Esempio 3: controllo degli accessi multi-tenant con RBAC

Per approfondire il precedente esempio RBAC, è possibile espandere i requisiti per includere la multi-tenancy SaaS, che è un requisito comune per i provider SaaS. Nelle soluzioni multi-tenant, l'accesso alle risorse viene sempre fornito per conto di un determinato tenant. Cioè, gli utenti del Tenant A non possono visualizzare i dati del Tenant B, anche se tali dati sono collocati logicamente o fisicamente in un sistema. L'esempio seguente illustra come implementare l'isolamento dei tenant utilizzando più archivi di policy per le autorizzazioni verificate e come utilizzare i ruoli utente per definire le autorizzazioni all'interno del tenant.

L'utilizzo del modello di progettazione Per Tenant Policy Store è una best practice per mantenere l'isolamento dei tenant durante l'implementazione del controllo degli accessi con autorizzazioni verificate. In questo scenario, le richieste degli utenti Tenant A e Tenant B vengono verificate in archivi di policy separati e, rispettivamente. `DATAMICROSERVICE_POLICYSTORE_A` `DATAMICROSERVICE_POLICYSTORE_B` Per ulteriori informazioni sulle considerazioni sulla progettazione di autorizzazioni verificate per applicazioni SaaS multi-tenant, consulta la sezione Considerazioni sulla progettazione multi-tenant delle autorizzazioni verificate.



La seguente policy si trova nel Policy Store. DATAMICROSERVICE_POLICYSTORE_A Verifica che il principale faccia parte del gruppo `allAccessRole` di tipi. `Role` In questo caso, il committente sarà autorizzato a eseguire le `updateData` azioni `viewData` e su tutte le risorse associate al Tenant A.

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

Le seguenti politiche si trovano nell'archivio delle DATAMICROSERVICE_POLICYSTORE_B politiche. La prima policy verifica che il principale faccia parte del `updateDataRole` gruppo di tipi. `Role` Supponendo che sia così, autorizza i mandanti a eseguire l'`updateData`azione sulle risorse associate al Tenant B.

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

Questa seconda politica impone che i committenti che fanno parte del `viewDataRole` gruppo di tipo `Role` siano autorizzati a eseguire l'`viewData`azione sulle risorse associate al Tenant B.

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

La richiesta di autorizzazione effettuata dal Tenant A deve essere inviata al DATAMICROSERVICE_POLICYSTORE_A policy store e verificata in base alle politiche che appartengono a tale archivio. In questo caso, viene verificata in base alla prima politica discussa in precedenza come parte di questo esempio. In questa richiesta di autorizzazione, il principale di tipo `User` con valore di `Alice` richiede di eseguire l'`viewData`azione. Il principale appartiene al gruppo

`allAccessRole` di tipi`Role`. Alice sta cercando di eseguire l'`viewData`azione sulla `SampleData` risorsa. Poiché Alice ha il `allAccessRole` ruolo, questa valutazione porta a una `ALLOW` decisione.

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

```
}
```

Se invece visualizzi una richiesta fatta dal Tenant B da`User Bob`, vedrai qualcosa come la seguente richiesta di autorizzazione. La richiesta viene inviata al `DATAMICROSERVICE_POLICYSTORE_B` policy store perché proviene dal Tenant B. In questa richiesta, il principale `Bob` desidera eseguire l'azione `updateData` sulla risorsa. `SampleData` Tuttavia, non `Bob` fa parte di un gruppo che ha accesso all'azione `updateData` su quella risorsa. Pertanto, la richiesta dà luogo a una `DENY` decisione.

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

```
        },
        "attributes": {},
        "parents": []
      }
    ]
  }
}
```

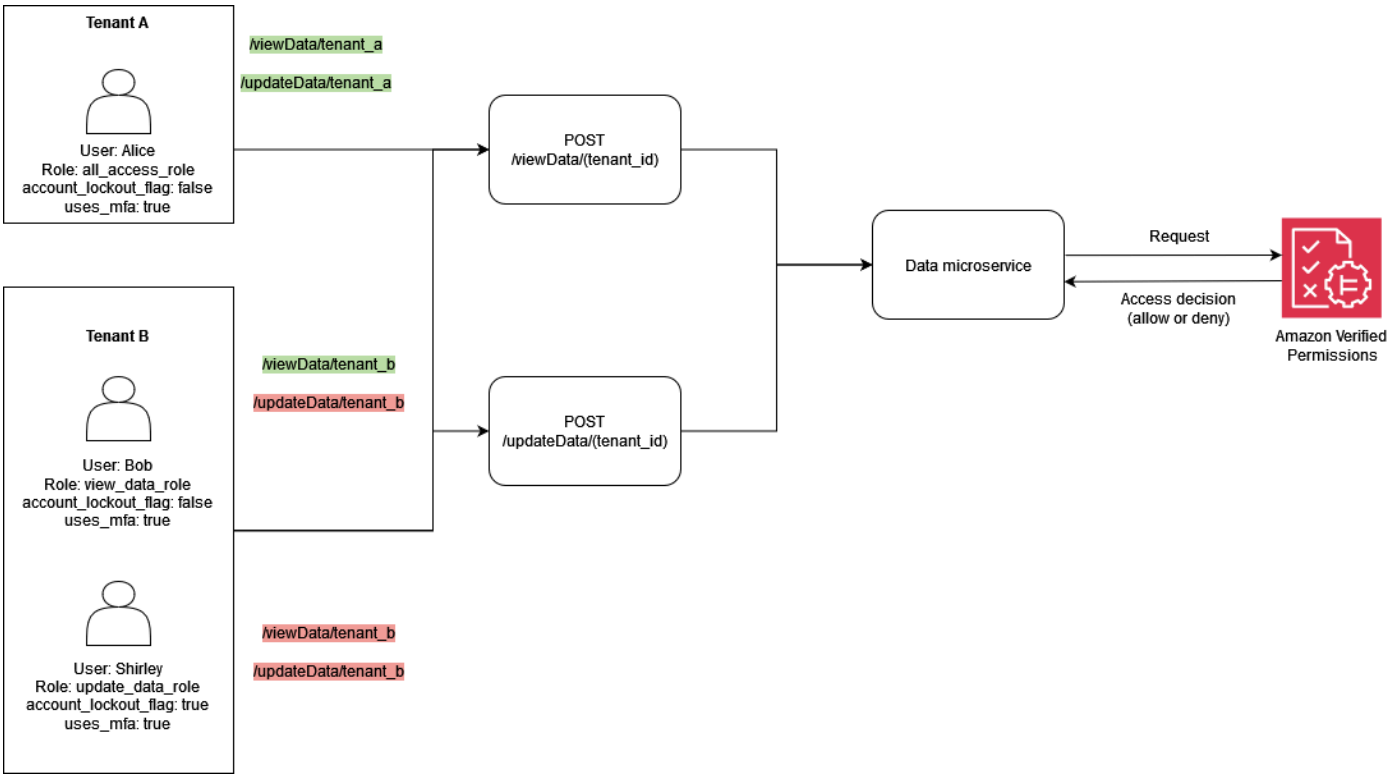
In questo terzo esempio, `User Alice` tenta di eseguire l'`viewData`azione sulla risorsa`SampleData`. Questa richiesta viene indirizzata al `DATAMICROSERVICE_POLICYSTORE_A` policy store perché il responsabile `Alice` appartiene al Tenant A. `Alice` fa parte del gruppo `allAccessRole` del tipo`Role`, che le consente di eseguire l'`viewData`azione sulle risorse. Pertanto, la richiesta dà luogo a una `ALLOW` decisione.

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

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

Esempio 4: controllo degli accessi multi-tenant con RBAC e ABAC

Per migliorare l'esempio RBAC della sezione precedente, è possibile aggiungere attributi agli utenti per creare un approccio ibrido RBAC-ABAC per il controllo degli accessi multi-tenant. Questo esempio include gli stessi ruoli dell'esempio precedente, ma aggiunge l'attributo user e il parametro context. `account_lockout_flag` `uses_mfa` L'esempio adotta inoltre un approccio diverso all'implementazione del controllo degli accessi multi-tenant utilizzando sia RBAC che ABAC e utilizza un archivio di politiche condiviso anziché un archivio di politiche diverso per ogni tenant.



Questo esempio rappresenta una soluzione SaaS multi-tenant in cui è necessario fornire decisioni di autorizzazione per il Tenant A e il Tenant B, in modo simile all'esempio precedente.

Per implementare la funzionalità di blocco degli utenti, l'esempio aggiunge l'attributo `account_lockout_flag` all'`User`entità principale nella richiesta di autorizzazione. Questo flag blocca l'accesso dell'utente al sistema e assegna `DENY` tutti i privilegi all'utente bloccato. L'`account_lockout_flag`attributo è associato all'`User`entità ed è valido `User` fino a quando il flag non viene revocato attivamente in più sessioni. L'esempio utilizza la `when` condizione per valutare`account_lockout_flag`.

L'esempio aggiunge anche dettagli sulla richiesta e sulla sessione. Le informazioni di contesto specificano che la sessione è stata autenticata utilizzando l'autenticazione a più fattori. Per implementare questa convalida, l'esempio utilizza la `when` condizione per valutare il `uses_mfa` flag come parte del campo di contesto. Per ulteriori informazioni sulle migliori pratiche per l'aggiunta di contesto, consulta la documentazione Cedar.

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

Questa politica impedisce l'accesso alle risorse a meno che la risorsa non sia nello stesso gruppo dell'attributo del `Tenant` principale richiedente. Questo approccio per mantenere l'isolamento dei tenant è denominato approccio One Shared Multi-Tenant Policy Store. Per ulteriori informazioni sulle considerazioni sulla progettazione di autorizzazioni verificate per applicazioni SaaS multi-tenant, consulta la sezione Considerazioni sulla progettazione multi-tenant delle autorizzazioni verificate.

La politica garantisce inoltre che il preside sia membro e limita le azioni a e. `allAccessRole` `viewData` `updateData` Inoltre, questa politica verifica che lo `account_lockout_flag` sia `false` e che il valore di contesto `uses_mfa` valuti a. `true`

Analogamente, la seguente politica garantisce che sia il principale che la risorsa siano associati allo stesso tenant, impedendo l'accesso tra tenant. Questa politica garantisce inoltre che il preside sia membro di `viewDataRole` e limita le azioni a. `viewData` Inoltre, verifica che il valore `account_lockout_flag` è `false` e che il valore di contesto di sia uguale a`uses_mfa`. `true`

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

La terza politica è simile alla precedente. La politica richiede che la risorsa sia un membro del gruppo che corrisponde all'entità rappresentata da`principal.Tenant`. Ciò garantisce che sia il principale che la risorsa siano associati al Tenant B, il che impedisce l'accesso tra tenant. Questa politica garantisce che il committente sia membro di `updateDataRole` e limita le azioni a. `updateData` Inoltre, questo criterio verifica che il valore di `account_lockout_flag` is `false` e il valore di contesto `uses_mfa` valuti a. `true`

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

La seguente richiesta di autorizzazione viene valutata in base alle tre politiche illustrate in precedenza in questa sezione. In questa richiesta di autorizzazione, il principale di tipo `User` e con valore di `Alice` effettua una `updateData` richiesta con il ruolo`allAccessRole`. `Alice`ha l'attributo `Tenant` il cui valore è`Tenant::"TenantA"`. L'azione `Alice` che si sta tentando di eseguire è `updateData,` e la risorsa a cui verrà applicata è `SampleData` del tipo`Data`. `SampleData`ha `TenantA` come entità principale.

In base alla prima politica nell'archivio delle <DATAMICROSERVICE_POLICYSTOREID> politiche, Alice può eseguire l'updateDataazione sulla risorsa, presupponendo che siano soddisfatte le condizioni nella when clausola della politica. La prima condizione richiede l'principal.Tenantattributo su cui eseguire la valutazione. TenantA La seconda condizione richiede che l'account_lockout_flagattributo principale siafalse. L'ultima condizione richiede che uses_mfa il contesto siatrue. Poiché tutte e tre le condizioni sono soddisfatte, la richiesta restituisce una ALLOW decisione.

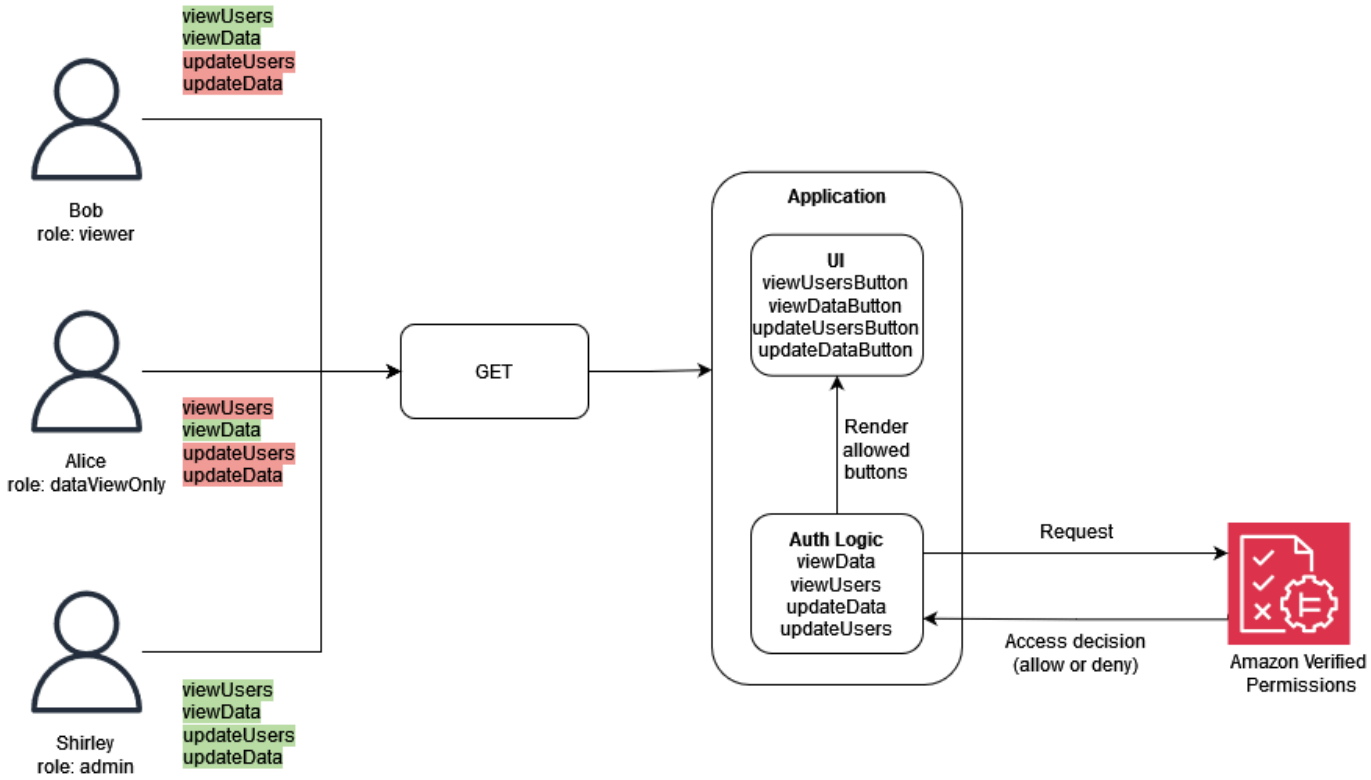
```
{
  "policyStoreId": "DATAMICROSERVICE_POLICYSTORE",
  "principal": {
      "entityType": "MultitenantApp::User",
      "entityId": "Alice"
  },
  "action": {
      "actionType": "MultitenantApp::Action",
      "actionId": "updateData"
  },
  "resource": {
      "entityType": "MultitenantApp::Data",
      "entityId": "SampleData"
  },
  "context": {
    "contextMap": {
        "uses_mfa": {
            "boolean": true
        }
    }
  },
  "entities": {
    "entityList": [
      {
        "identifier": {
            "entityType": "MultitenantApp::User",
            "entityId": "Alice"
        },
        "attributes": {
            {
                "account_lockout_flag": {
                    "boolean": false
                },
                "Tenant": {
```

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

Esempio 5: filtraggio dell'interfaccia utente con autorizzazioni verificate e Cedar

Puoi anche utilizzare le autorizzazioni verificate per implementare il filtraggio RBAC degli elementi dell'interfaccia utente in base ad azioni autorizzate. Ciò è estremamente utile per le applicazioni con elementi dell'interfaccia utente sensibili al contesto che potrebbero essere associati a utenti o tenant specifici nel caso di un'applicazione SaaS multi-tenant.

Nell'esempio seguente, `Users` non `Role` `viewer` sono autorizzati a eseguire aggiornamenti. Per questi utenti, l'interfaccia utente non dovrebbe visualizzare alcun pulsante di aggiornamento.



In questo esempio, un'applicazione Web a pagina singola ha quattro pulsanti. I pulsanti visibili dipendono dall'`Role`utente che ha attualmente effettuato l'accesso all'applicazione. Quando l'applicazione Web a pagina singola esegue il rendering dell'interfaccia utente, richiede le autorizzazioni verificate per determinare quali azioni l'utente è autorizzato a eseguire, quindi genera i pulsanti in base alla decisione di autorizzazione.

La seguente politica specifica che il tipo `Role` con un valore di `viewer` può visualizzare sia gli utenti che i dati. Una decisione di `ALLOW` autorizzazione per questa politica richiede un'`viewUsers`azione `viewData` o e richiede inoltre che una risorsa sia associata al tipo `Data` o`Users`. Una `ALLOW` decisione consente all'interfaccia utente di visualizzare due pulsanti: `viewDataButton` `eviewUsersButton`.

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

La seguente politica specifica che il tipo `Role` con un valore di `viewerDataOnly` può solo visualizzare i dati. Una decisione di `ALLOW` autorizzazione per questa politica richiede un'`viewData`azione e richiede anche l'associazione di una risorsa al tipo`Data`. Una `ALLOW` decisione consente all'interfaccia utente di eseguire il rendering del pulsante`viewDataButton`.

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

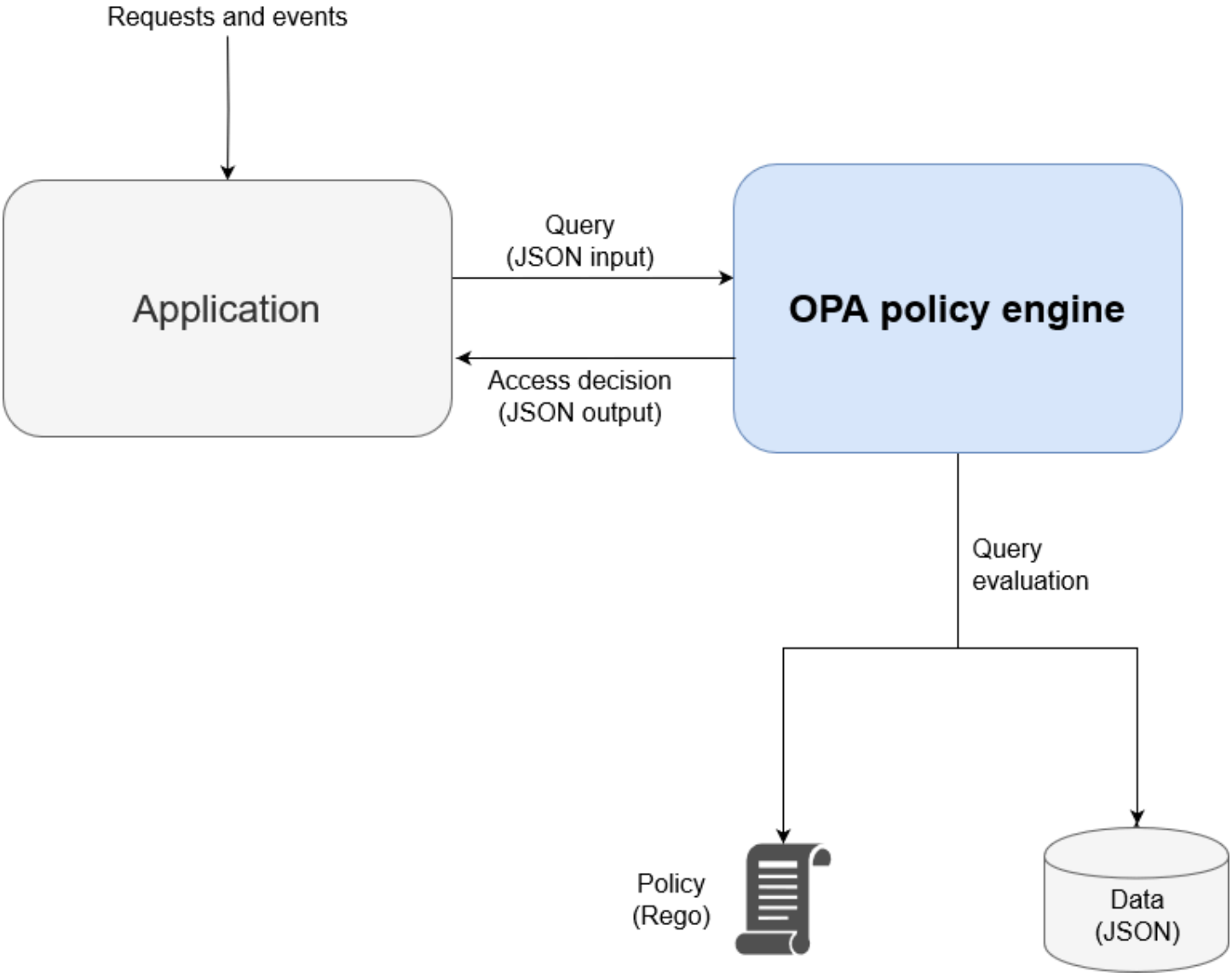
La seguente politica specifica che il tipo `Role` con un valore di `admin` può modificare e visualizzare dati e utenti. Una decisione di `ALLOW` autorizzazione per questa politica richiede un'azione di `updateData` `updateUsers``viewUsers`, `viewData,` o e richiede inoltre che una risorsa sia associata al tipo `Data` o`Users`. Una `ALLOW` decisione consente all'interfaccia utente di visualizzare tutti e quattro i pulsanti:`updateDataButton`, `updateUsersButton``viewDataButton`, `eviewUsersButton`.

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

Implementazione di un PDP utilizzando OPA

L'Open Policy Agent (OPA) è un motore di policy generico open source. OPA ha molti casi d'uso, ma il caso d'uso rilevante per l'implementazione PDP è la sua capacità di disaccoppiare la logica di autorizzazione da un'applicazione. Questo processo si chiama disaccoppiamento delle politiche. L'OPA è utile nell'implementazione di un PDP per diversi motivi. Utilizza un linguaggio dichiarativo di alto livello chiamato Rego per redigere politiche e regole. Queste politiche e regole esistono

separatamente da un'applicazione e possono prendere decisioni di autorizzazione senza alcuna logica specifica dell'applicazione. OPA espone inoltre un'API RESTful per rendere semplici e dirette le decisioni di recupero delle autorizzazioni. Per prendere una decisione di autorizzazione, un'applicazione interroga OPA con input JSON e OPA valuta l'input rispetto alle politiche specificate per restituire una decisione di accesso in JSON. OPA è anche in grado di importare dati esterni che potrebbero essere rilevanti per prendere una decisione di autorizzazione.



L'OPA presenta diversi vantaggi rispetto ai motori di policy personalizzati:

- L'OPA e la sua valutazione delle politiche con Rego forniscono un motore di policy flessibile e predefinito che richiede solo l'inserimento di politiche e di tutti i dati necessari per prendere decisioni di autorizzazione. Questa logica di valutazione delle politiche dovrebbe essere ricreata in una soluzione di motore delle politiche personalizzata.

- L'OPA semplifica la logica di autorizzazione disponendo di politiche scritte in un linguaggio dichiarativo. È possibile modificare e amministrare queste politiche e regole indipendentemente da qualsiasi codice applicativo, senza competenze di sviluppo di applicazioni.
- L'OPA espone un'API RESTful, che semplifica l'integrazione con i punti di applicazione delle politiche (PEP).
- OPA fornisce supporto integrato per la convalida e la decodifica dei token Web JSON (JWT).
- OPA è uno standard di autorizzazione riconosciuto, il che significa che la documentazione e gli esempi sono numerosi se hai bisogno di assistenza o ricerca per risolvere un problema particolare.
- L'adozione di uno standard di autorizzazione come OPA consente di condividere le politiche scritte in Rego tra i team indipendentemente dal linguaggio di programmazione utilizzato dall'applicazione del team.

Ci sono due cose che OPA non fornisce automaticamente:

- OPA non dispone di un solido piano di controllo per l'aggiornamento e la gestione delle politiche. OPA fornisce alcuni modelli di base per l'implementazione degli aggiornamenti delle politiche, il monitoraggio e l'aggregazione dei log esponendo un'API di gestione, ma l'integrazione con questa API deve essere gestita dall'utente OPA. Come best practice, è consigliabile utilizzare una pipeline di integrazione e distribuzione continua (CI/CD) per amministrare, modificare e tenere traccia delle versioni delle policy e gestire le politiche in OPA.
- Per impostazione predefinita, OPA non può recuperare dati da fonti esterne. Una fonte esterna di dati per una decisione di autorizzazione potrebbe essere un database che contiene gli attributi utente. Esiste una certa flessibilità nel modo in cui i dati esterni vengono forniti all'OPA: possono essere memorizzati nella cache locale in anticipo o recuperati dinamicamente da un'API quando viene richiesta una decisione di autorizzazione, ma l'OPA non può ottenere queste informazioni per conto dell'utente.

Panoramica di Rego

Rego è un linguaggio di policy generico, il che significa che funziona per qualsiasi livello dello stack e per qualsiasi dominio. Lo scopo principale di Rego è accettare input e dati JSON/YAML che vengono valutati per prendere decisioni basate sulle politiche sulle risorse, le identità e le operazioni dell'infrastruttura. Rego consente di scrivere policy su qualsiasi livello di uno stack o di un dominio senza richiedere una modifica o un'estensione del linguaggio. Ecco alcuni esempi di decisioni che Rego può prendere:

- Questa richiesta API è consentita o rifiutata?
- Qual è il nome host del server di backup per questa applicazione?
- Qual è il punteggio di rischio associato a questa modifica dell'infrastruttura proposta?
- In quali cluster deve essere distribuito questo container per un'elevata disponibilità?
- Quali informazioni di routing devono essere utilizzate per questo microservizio?

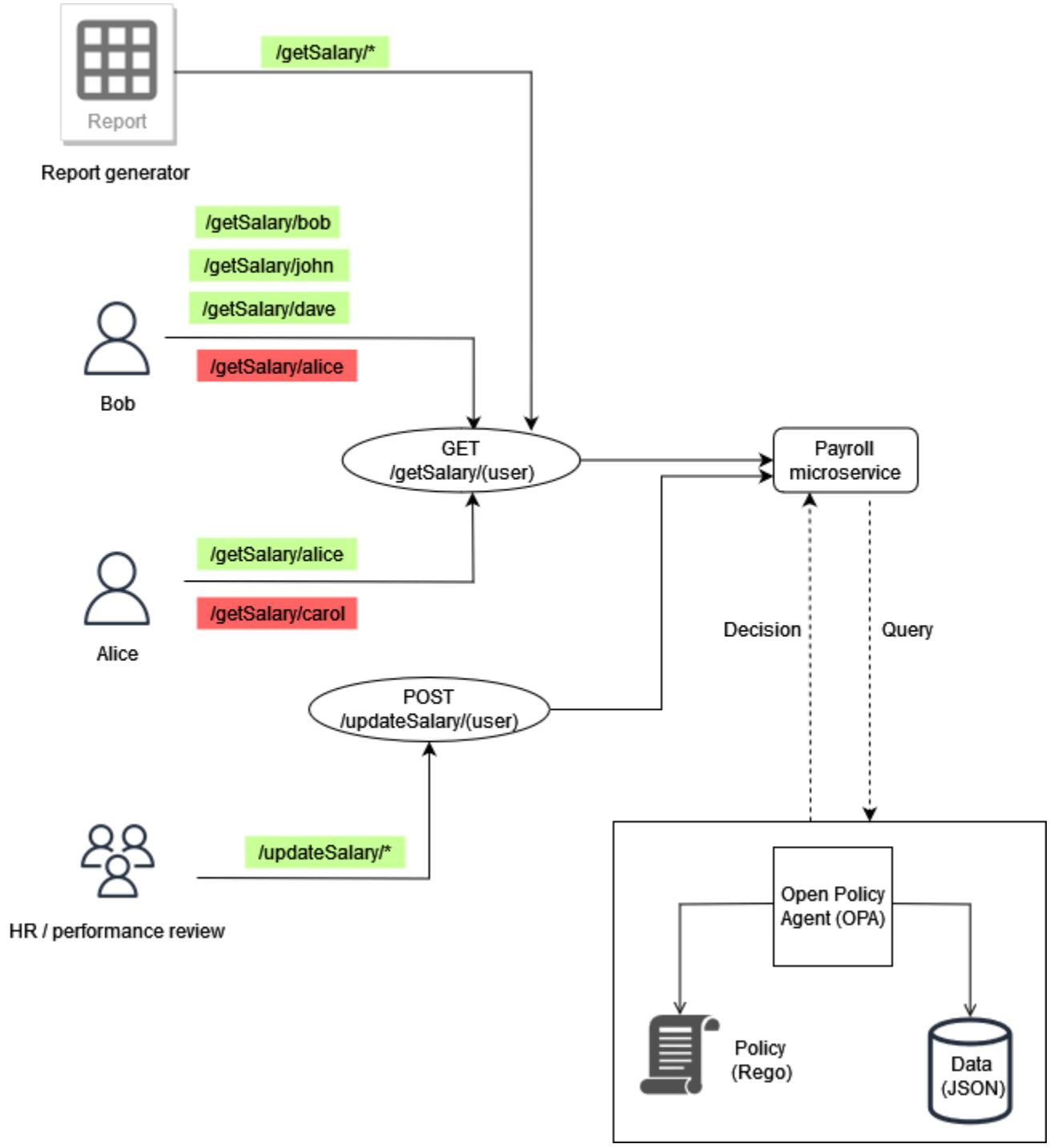
Per rispondere a queste domande, Rego utilizza una filosofia di base su come queste decisioni possono essere prese. I due principi chiave nella stesura della politica in Rego sono:

- Ogni risorsa, identità o operazione può essere rappresentata come dati JSON o YAML.
- La policy è la logica applicata ai dati.

Rego aiuta i sistemi software a prendere decisioni di autorizzazione definendo la logica su come vengono valutati gli input dei dati JSON/YAML. I linguaggi di programmazione come C, Java, Go e Python sono la soluzione abituale a questo problema, ma Rego è stato progettato per concentrarsi sui dati e sugli input che rappresentano il sistema e sulla logica per prendere decisioni politiche con queste informazioni.

Esempio 1: ABAC di base con OPA e Rego

Questa sezione descrive uno scenario in cui l'OPA viene utilizzato per prendere decisioni di accesso su quali utenti sono autorizzati ad accedere alle informazioni in un microservizio Payroll fittizio. Vengono forniti frammenti di codice Rego per dimostrare come utilizzare Rego per prendere decisioni sul controllo degli accessi. Questi esempi non sono né esaustivi né un'esplorazione completa delle funzionalità di Rego e OPA. Per una panoramica più completa di Rego, ti consigliamo di consultare la documentazione Rego sul sito web OPA.



Esempio di regole OPA di base

Nel diagramma precedente, una delle regole di controllo degli accessi applicate dall'OPA per il microservizio Payroll è:

I dipendenti possono leggere il proprio stipendio.

Se Bob tenta di accedere al microservizio Payroll per visualizzare il proprio stipendio, il microservizio Payroll può reindirizzare la chiamata API all'API RESTful OPA per prendere una decisione di accesso. Il servizio Payroll richiede all'OPA una decisione con il seguente input JSON:

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

OPA seleziona una o più politiche in base alla query. In questo caso, la seguente politica, scritta in Rego, valuta l'input JSON.

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

Questa politica nega l'accesso per impostazione predefinita. Quindi valuta l'input nella query associandolo alla variabile globale. `input` L'operatore punto viene utilizzato con questa variabile per accedere ai valori della variabile. La regola Rego `allow` restituisce true se anche le espressioni nella regola sono vere. La regola Rego verifica che l'input sia uguale `method` a GET. Quindi verifica che il primo elemento dell'elenco `path` sia presente `getSalary` prima di assegnare il secondo elemento dell'elenco alla variabile. `user` Infine, verifica che il percorso a cui si accede verifichi `/getSalary/bob` verificando che la `user` richiesta in corso corrisponda alla `input.user` variabile. `user` La regola `allow` applica la logica if-then per restituire un valore booleano, come mostrato nell'output:

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

Regola parziale che utilizza dati esterni

Per dimostrare funzionalità OPA aggiuntive, puoi aggiungere requisiti alla regola di accesso che stai applicando. Supponiamo di voler applicare questo requisito di controllo degli accessi nel contesto dell'illustrazione precedente:

I dipendenti possono leggere lo stipendio di chiunque risponda a loro.

In questo esempio, OPA ha accesso a dati esterni che possono essere importati per prendere una decisione di accesso:

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

È possibile generare una risposta JSON arbitraria creando una regola parziale in OPA, che restituisce un set di valori anziché una risposta fissa. Questo è un esempio di regola parziale:

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

Questa regola restituisce un insieme di tutti gli utenti che riportano il valore di`input.user`, che, in questo caso, è`bob`. Il `[_]` costrutto contenuto nella regola viene utilizzato per iterare sui valori del set. Questo è il risultato della regola:

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

Il recupero di queste informazioni può aiutare a determinare se un utente è un referente diretto di un manager. Per alcune applicazioni, è preferibile restituire un codice JSON dinamico piuttosto che restituire una semplice risposta booleana.

Mettere tutto insieme

L'ultimo requisito di accesso è più complesso dei primi due perché combina le condizioni specificate in entrambi i requisiti:

I dipendenti possono leggere il proprio stipendio e quello di chiunque risponda a loro.

Per soddisfare questo requisito, puoi utilizzare questa politica Rego:

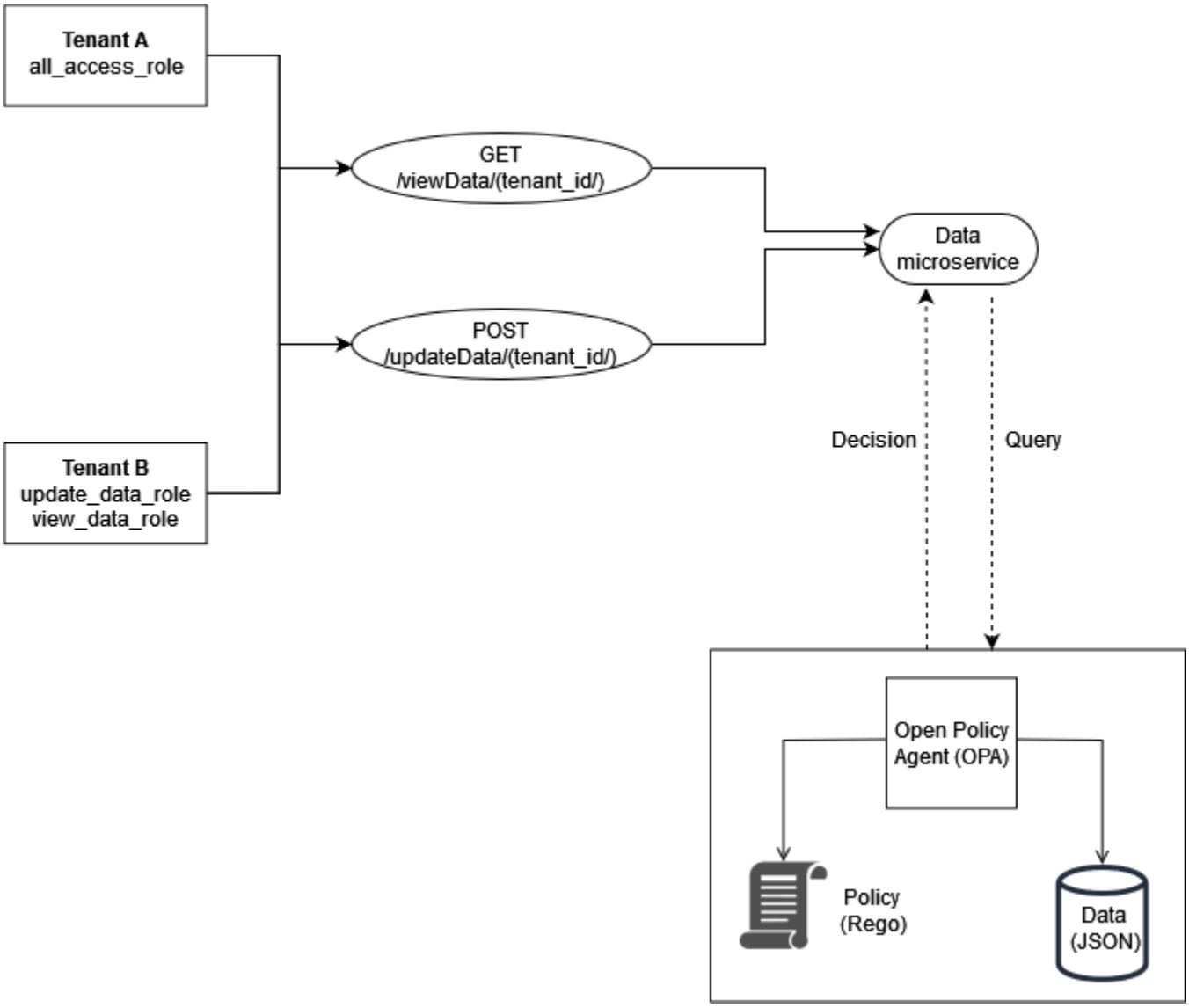
```
default allow = false

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 prima regola della politica consente l'accesso a qualsiasi utente che tenti di visualizzare le proprie informazioni salariali, come discusso in precedenza. Avere due regole con lo stesso nome`allow`, funziona come operatore logico o come operatore in Rego. La seconda regola recupera l'elenco di tutti i report diretti associati a `input.user` (dai dati del diagramma precedente) e assegna questo elenco alla variabile. `managers` Infine, la regola verifica se l'utente che sta cercando di visualizzare il proprio stipendio è un referente diretto `input.user` verificando che il suo nome sia contenuto nella variabile. `managers`

Gli esempi in questa sezione sono molto semplici e non forniscono un'esplorazione completa o approfondita delle funzionalità di Rego e OPA. Per ulteriori informazioni, consultate la documentazione OPA, consultate il file _GitHub README OPA e sperimentate nel parco giochi Rego.

Esempio 2: controllo degli accessi multi-tenant e RBAC definito dall'utente con OPA e Rego

Questo esempio utilizza OPA e Rego per dimostrare come implementare il controllo degli accessi su un'API per un'applicazione multi-tenant con ruoli personalizzati definiti dagli utenti tenant. Dimostra inoltre come è possibile limitare l'accesso in base a un tenant. Questo modello mostra come OPA può prendere decisioni granulari sulle autorizzazioni sulla base delle informazioni fornite in un ruolo di alto livello.



I ruoli degli inquilini sono archiviati in dati esterni (dati RBAC) utilizzati per prendere decisioni di accesso per OPA:

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

```
}
```

Questi ruoli, se definiti da un utente tenant, devono essere archiviati in un'origine dati esterna o in un provider di identità (IdP) che possa fungere da fonte di verità durante la mappatura dei ruoli definiti dal tenant alle autorizzazioni e al tenant stesso.

Questo esempio utilizza due politiche in OPA per prendere decisioni di autorizzazione e per esaminare come queste politiche impongono l'isolamento dei tenant. Queste politiche utilizzano i dati RBAC definiti in precedenza.

```
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")
}
```

Per mostrare come funzionerà questa regola, considera una query OPA con il seguente input:

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

Una decisione di autorizzazione per questa chiamata API viene presa come segue, combinando i dati RBAC, le politiche OPA e l'input della query OPA:

1. Un utente di `Tenant A` effettua una chiamata API a. `/viewData/tenant_a`
2. Il microservizio Data riceve la chiamata e interroga la `allowViewData` regola, passando l'input mostrato nell'esempio di input della query OPA.
3. OPA utilizza la regola interrogata nelle politiche OPA per valutare l'input fornito. L'OPA utilizza anche i dati dei dati RBAC per valutare l'input. L'OPA esegue le seguenti operazioni:
   a. Verifica che il metodo utilizzato per effettuare la chiamata API sia. `GET`
   b. Verifica che il percorso richiesto sia. `viewData`

c. Verifica che il valore `tenant_id` nel percorso sia uguale a quello `input.tenant_id` associato all'utente. Ciò garantisce il mantenimento dell'isolamento degli inquilini. Un altro tenant, anche con un ruolo identico, non può essere autorizzato a effettuare questa chiamata API.

d. Estrae un elenco di autorizzazioni di ruolo dai dati esterni dei ruoli e le assegna alla variabile. `role_permissions` Questo elenco viene recuperato utilizzando il ruolo definito dal tenant associato all'utente in `input.role.`

e. Verifica se `role_permissions` contiene l'autorizzazione `viewData.`

4. OPA restituisce la seguente decisione al microservizio Data:

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

Questo processo mostra come RBAC e la consapevolezza degli inquilini possono contribuire a prendere una decisione di autorizzazione con OPA. Per illustrare ulteriormente questo punto, considera una chiamata API a `/viewData/tenant_b` con il seguente input di query:

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

Questa regola restituirebbe lo stesso output dell'input della query OPA, sebbene sia destinata a un tenant diverso che ha un ruolo diverso. Questo perché questa chiamata è destinata `/tenant_b` e ai dati `view_data_role` in RBAC è ancora associata l'`viewData`autorizzazione. Per applicare lo stesso tipo di controllo degli accessi per`/updateData`, puoi utilizzare una regola OPA simile:

```
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")
```

```
}
```

Questa regola è funzionalmente la stessa della `allowViewData` regola, ma verifica un percorso e un metodo di input diversi. La regola garantisce comunque l'isolamento dei tenant e verifica che il ruolo definito dal tenant conceda l'autorizzazione al chiamante dell'API. Per vedere come ciò potrebbe essere applicato, esamina il seguente input di query per una chiamata API a: `/updateData/tenant_b`

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

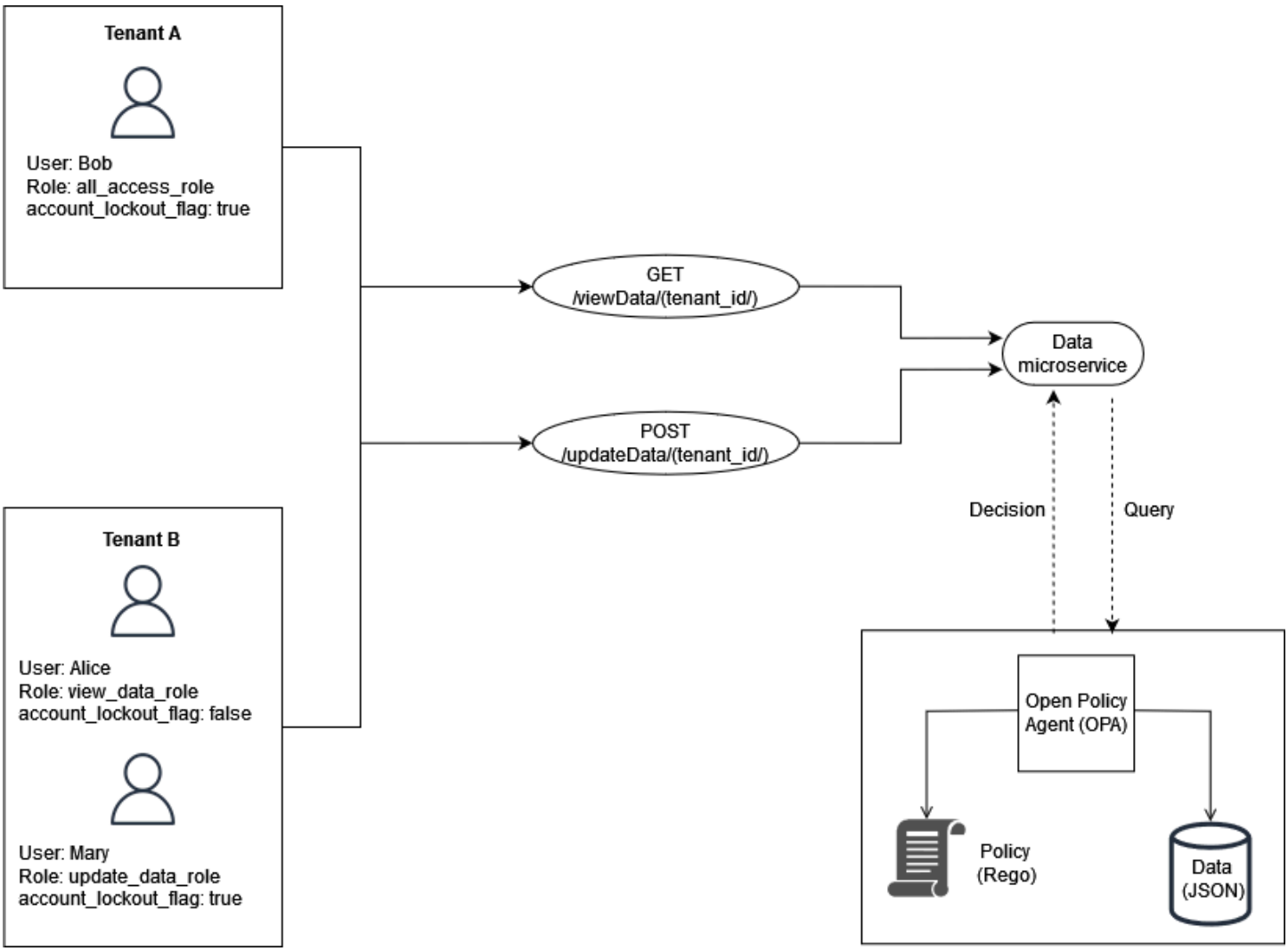
Questo input di query, se valutato con la `allowUpdateData` regola, restituisce la seguente decisione di autorizzazione:

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

Questa chiamata non sarà autorizzata. Sebbene il chiamante dell'API sia associato all'indirizzo corretto `tenant_id` e stia chiamando l'API utilizzando un metodo approvato, `input.role` è definito dal `view_data_role` tenant. `view_data_role`Non dispone dell'`updateData`autorizzazione, pertanto la chiamata a non è autorizzata. `/updateData` Questa chiamata avrebbe avuto successo per un `tenant_b` utente che dispone di`update_data_role`.

Esempio 3: controllo degli accessi multi-tenant per RBAC e ABAC con OPA e Rego

Per migliorare l'esempio RBAC nella sezione precedente, è possibile aggiungere attributi agli utenti.



Questo esempio include gli stessi ruoli dell'esempio precedente, ma aggiunge l'attributo user. `account_lockout_flag` Si tratta di un attributo specifico dell'utente che non è associato a nessun ruolo particolare. È possibile utilizzare gli stessi dati esterni RBAC utilizzati in precedenza per questo esempio:

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

```
}
```

L'attributo `account_lockout_flag` user può essere passato al servizio Data come parte dell'input di una query OPA `/viewData/tenant_a` per l'utente Bob:

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

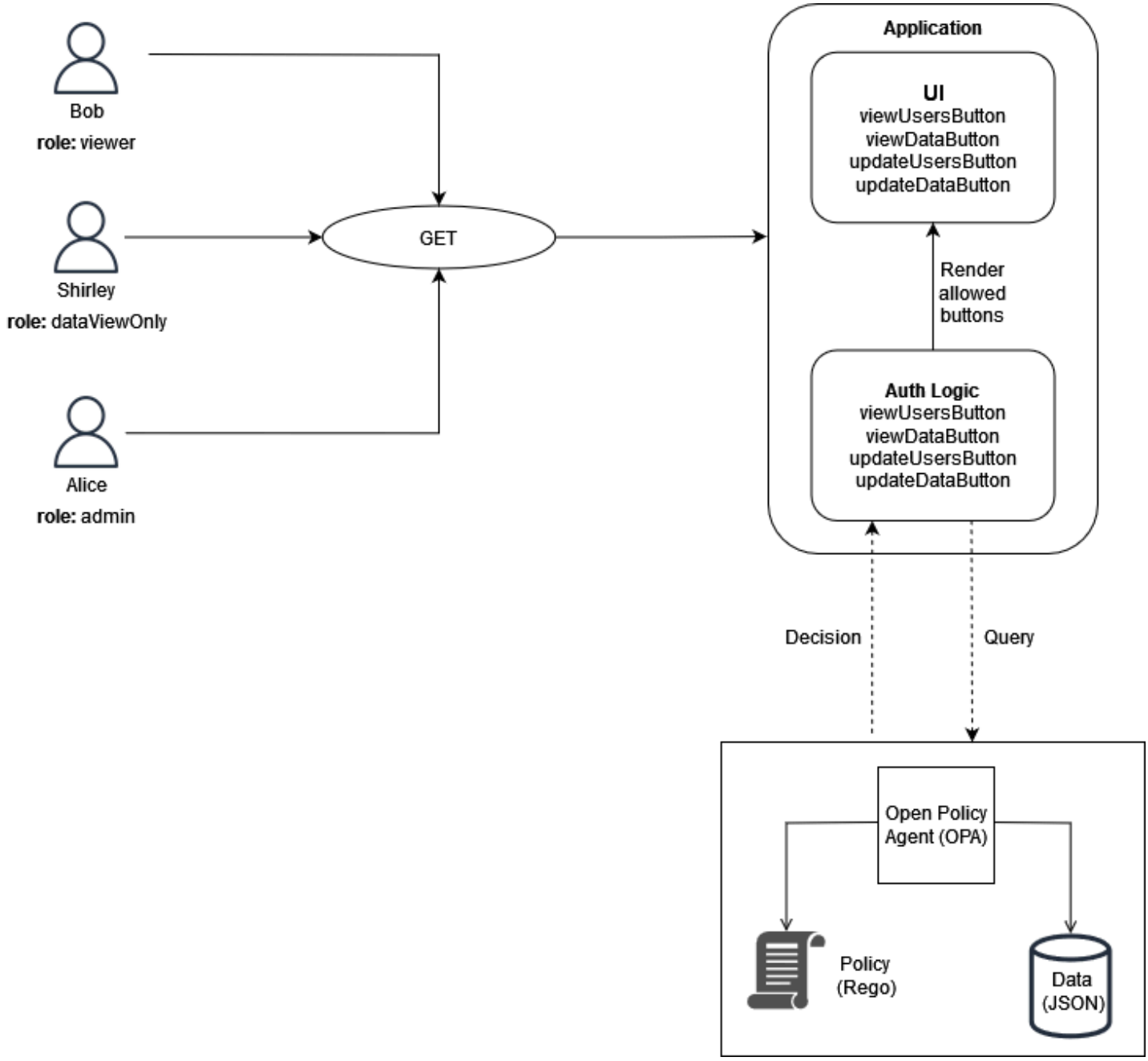
La regola richiesta per la decisione di accesso è simile agli esempi precedenti, ma include una riga aggiuntiva per verificare l'attributo: `account_lockout_flag`

```
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"
}
```

Questa query restituisce una decisione di autorizzazione di. `false` Questo perché `account_lockout_flag attribute` è `true` per Bob e la regola Rego `allowViewData` nega l'accesso sebbene Bob abbia il ruolo e l'inquilino corretti.

Esempio 4: filtraggio dell'interfaccia utente con OPA e Rego

La flessibilità di OPA e Rego supporta la capacità di filtrare gli elementi dell'interfaccia utente. L'esempio seguente dimostra come una regola parziale OPA può prendere decisioni di autorizzazione su quali elementi devono essere visualizzati in un'interfaccia utente con RBAC. Questo metodo è uno dei tanti modi diversi per filtrare gli elementi dell'interfaccia utente con OPA.



In questo esempio, un'applicazione Web a pagina singola ha quattro pulsanti. Supponiamo che tu voglia filtrare l'interfaccia utente di Bob, Shirley e Alice in modo che possano vedere solo i pulsanti che corrispondono ai loro ruoli. Quando l'interfaccia utente riceve una richiesta dall'utente, interroga una regola parziale OPA per determinare quali pulsanti devono essere visualizzati nell'interfaccia utente. La query trasmette quanto segue come input a OPA quando Bob (con il ruolo`viewer`) effettua una richiesta all'interfaccia utente:

```
{
```

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

OPA utilizza dati esterni strutturati per RBAC per prendere una decisione di accesso:

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

La regola parziale OPA utilizza sia i dati esterni che l'input per produrre un elenco di azioni consentite:

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

Nella regola parziale, OPA utilizza quanto `input.role` specificato come parte della query per determinare quali pulsanti devono essere visualizzati. Bob ha il ruolo e `viewer` i dati esterni specificano che gli spettatori hanno due autorizzazioni: e. `viewUsers` `viewData` Pertanto, l'output di questa regola per Bob (e per tutti gli altri utenti che hanno un ruolo di spettatore) è il seguente:

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

L'output per Shirley, che ha il `dataViewOnly` ruolo, conterrebbe un pulsante di autorizzazione:. `viewData` L'output di Alice, che ha il `admin` ruolo, conterrebbe tutte queste autorizzazioni. Queste risposte vengono restituite all'interfaccia utente quando viene richiesta l'OPA. `user_permissions` L'applicazione può quindi utilizzare questa risposta per nascondere o visualizzare i`viewUsersButton`, `viewDataButton``updateUsersButton`, e. `updateDataButton`

Utilizzo di un motore di policy personalizzato

Un metodo alternativo per implementare un PDP consiste nel creare un motore di policy personalizzato. L'obiettivo di questo motore di policy è disaccoppiare la logica di autorizzazione da un'applicazione. Il motore di policy personalizzato è responsabile dell'adozione di decisioni di autorizzazione, analogamente a Verified Permissions o OPA, per ottenere il disaccoppiamento delle politiche. La differenza principale tra questa soluzione e l'utilizzo di Autorizzazioni verificate o OPA è che la logica per la scrittura e la valutazione delle politiche è progettata su misura per un motore di policy personalizzato. Qualsiasi interazione con il motore deve essere esposta tramite un'API o un altro metodo per consentire alle decisioni di autorizzazione di raggiungere un'applicazione. È possibile scrivere un motore di policy personalizzato in qualsiasi linguaggio di programmazione o utilizzare altri meccanismi per la valutazione delle politiche, come il Common Expression Language (CEL).

Implementazione di un PEP

Un punto di applicazione delle politiche (PEP) è responsabile della ricezione delle richieste di autorizzazione che vengono inviate al punto decisionale delle politiche (PDP) per la valutazione. Un PEP può trovarsi in qualsiasi punto dell'applicazione in cui è necessario proteggere dati e risorse o in cui viene applicata la logica di autorizzazione. I PEP sono relativamente semplici rispetto ai PDP. Un PEP è responsabile solo della richiesta e della valutazione di una decisione di autorizzazione e non richiede alcuna logica di autorizzazione. I PEP, a differenza dei PDP, non possono essere centralizzati in un'applicazione SaaS. Questo perché l'autorizzazione e il controllo degli accessi devono essere implementati in un'applicazione e nei suoi punti di accesso. I PEP possono essere applicati alle API, ai microservizi, ai livelli di Backend for Frontend (BFF) o a qualsiasi punto dell'applicazione in cui è desiderato o richiesto il controllo degli accessi. Rendere i PEP pervasivi in un'applicazione garantisce che l'autorizzazione venga verificata spesso e in modo indipendente in più punti.

Per implementare un PEP, il primo passo è determinare dove applicare il controllo degli accessi in un'applicazione. Considerate questo principio quando decidete dove integrare i PEP nella vostra applicazione:

Se un'applicazione espone un'API, dovrebbe esserci l'autorizzazione e il controllo degli accessi su quell'API.

Questo perché in un'architettura orientata ai microservizi o ai servizi, le API fungono da separatori tra le diverse funzioni dell'applicazione. È opportuno includere il controllo degli accessi come checkpoint logici tra le funzioni dell'applicazione. Ti consigliamo vivamente di includere i PEP come prerequisito per l'accesso a ciascuna API in un'applicazione SaaS. È anche possibile integrare l'autorizzazione in altri punti di un'applicazione. Nelle applicazioni monolitiche, potrebbe essere necessario che i PEP siano integrati nella logica dell'applicazione stessa. Non esiste un'unica sede in cui includere i PEP, ma considerate la possibilità di utilizzare il principio dell'API come punto di partenza.

Richiesta di una decisione di autorizzazione

Un PEP deve richiedere una decisione di autorizzazione al PDP. La richiesta può assumere diverse forme. Il metodo più semplice e accessibile per richiedere una decisione di autorizzazione consiste nell'inviare una richiesta o una query di autorizzazione a un'API RESTful esposta dal PDP (OPA o Verified Permissions). Se utilizzi le autorizzazioni verificate, puoi anche richiamare il IsAuthorizedmetodo utilizzando l' AWS SDK per recuperare una decisione di autorizzazione. L'unica

funzione di un PEP in questo schema è quella di inoltrare le informazioni necessarie alla richiesta o alla query di autorizzazione. Ciò può essere semplice: basta inoltrare una richiesta ricevuta da un'API come input al PDP. Esistono altri metodi per creare PEP. Ad esempio, è possibile integrare un OPA PDP localmente con un'applicazione scritta nel linguaggio di programmazione Go come libreria invece di utilizzare un'API.

Valutazione di una decisione di autorizzazione

I PEP devono includere una logica per valutare i risultati di una decisione di autorizzazione. Quando i PDP vengono esposti come API, la decisione di autorizzazione è probabilmente in formato JSON e restituita da una chiamata API. Il PEP deve valutare questo codice JSON per determinare se l'azione intrapresa è autorizzata. Ad esempio, se un PDP è progettato per fornire una decisione di autorizzazione booleana di consentire o negare l'autorizzazione, il PEP potrebbe semplicemente controllare questo valore e quindi restituire il codice di stato HTTP 200 per consentire e il codice di stato HTTP 403 per negare. Questo modello di incorporazione di un PEP come prerequisito per l'accesso a un'API è un modello facilmente implementabile e altamente efficace per implementare il controllo degli accessi su un'applicazione SaaS. In scenari più complicati, il PEP potrebbe essere responsabile della valutazione del codice JSON arbitrario restituito dal PDP. Il PEP deve essere scritto in modo da includere la logica necessaria per interpretare la decisione di autorizzazione restituita dal PDP. Poiché è probabile che un PEP venga implementato in molti punti diversi dell'applicazione, consigliamo di impacchettare il codice PEP come libreria o artefatto riutilizzabile nel linguaggio di programmazione preferito. In questo modo, il PEP può essere facilmente integrato in qualsiasi punto dell'applicazione con rielaborazioni minime.

Modelli di progettazione per architetture SaaS multi-tenant

Esistono molti modi per implementare il controllo e l'autorizzazione degli accessi alle API. Questa guida si concentra su tre modelli di progettazione efficaci per le architetture SaaS multi-tenant. Questi progetti fungono da riferimento di alto livello per l'implementazione dei punti decisionali politici (PDP) e dei punti di applicazione delle politiche (PEP), per formare un modello di autorizzazione coeso e onnipresente per le applicazioni.

Modelli di progettazione:

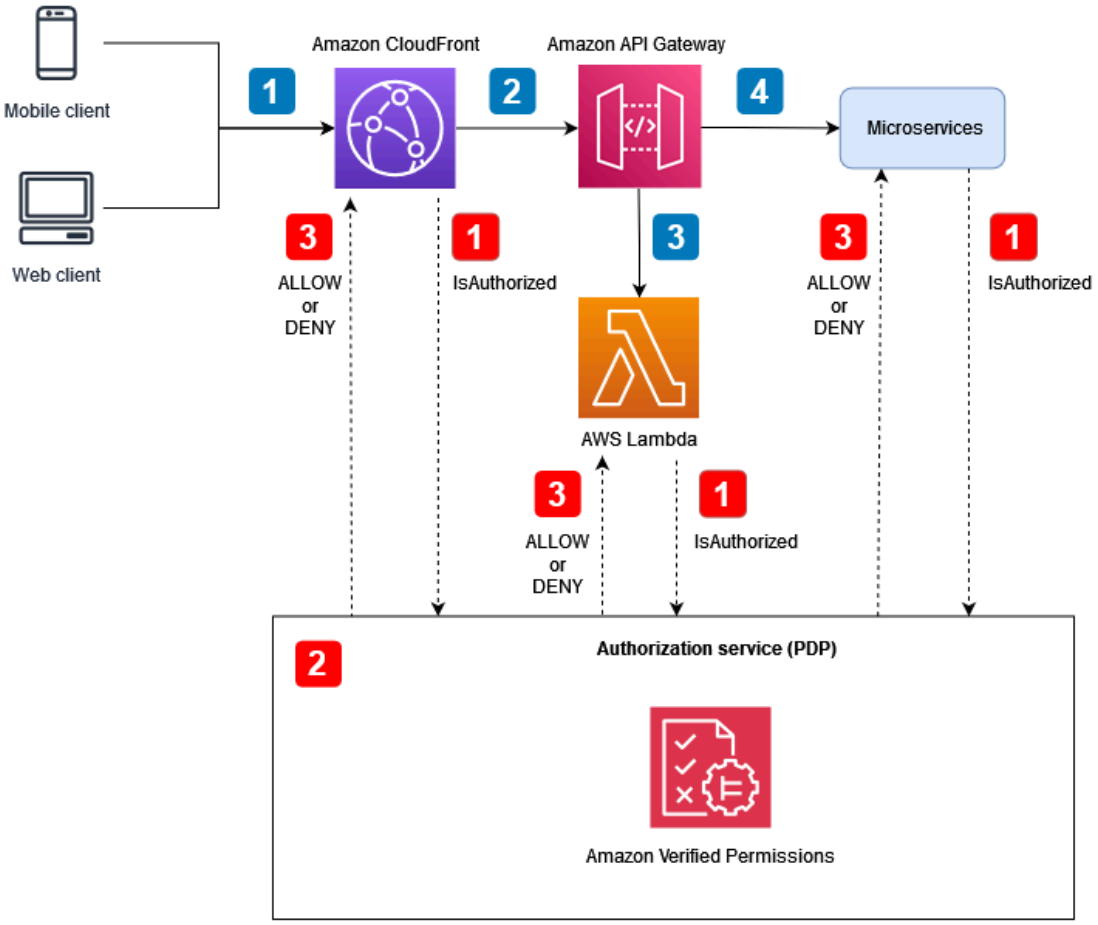
- Modelli di progettazione per Amazon Verified Permissions
- Modelli di progettazione per OPA

Modelli di progettazione per Amazon Verified Permissions

Utilizzo di un PDP centralizzato con PEP su API

Il modello PDP (Policy Decision Point) centralizzato con punti di applicazione delle policy (PEP) sulle API segue le migliori pratiche del settore per creare un sistema efficace e di facile manutenzione per il controllo e l'autorizzazione degli accessi alle API. Questo approccio supporta diversi principi chiave:

- L'autorizzazione e il controllo dell'accesso alle API vengono applicati in più punti dell'applicazione.
- La logica di autorizzazione è indipendente dall'applicazione.
- Le decisioni sul controllo degli accessi sono centralizzate.



Flusso dell'applicazione (illustrato con callout numerati in blu nel diagramma):

1. Un utente autenticato con un JSON Web Token (JWT) genera una richiesta HTTP ad Amazon. CloudFront
2. CloudFront inoltra la richiesta ad Amazon API Gateway, che è configurato come CloudFront origine.
3. Viene chiamato un autorizzatore personalizzato API Gateway per verificare il JWT.
4. I microservizi rispondono alla richiesta.

Flusso di autorizzazione e controllo dell'accesso alle API (illustrato con callout numerati in rosso nel diagramma):

1. Il PEP chiama il servizio di autorizzazione e trasmette i dati della richiesta, inclusi eventuali JWT.
2. Il servizio di autorizzazione (PDP), in questo caso Verified Permissions, utilizza i dati della richiesta come input di query e li valuta in base alle politiche pertinenti specificate dalla query.

3. La decisione di autorizzazione viene restituita al PEP e valutata.

Questo modello utilizza un PDP centralizzato per prendere decisioni di autorizzazione. I PEP vengono implementati in diversi punti per effettuare richieste di autorizzazione al PDP. Il diagramma seguente mostra come implementare questo modello in un'ipotetica applicazione SaaS multi-tenant.

In questa architettura, i PEP richiedono decisioni di autorizzazione sugli endpoint del servizio per Amazon CloudFront e Amazon API Gateway e per ogni microservizio. La decisione di autorizzazione viene presa dal servizio di autorizzazione Amazon Verified Permissions (PDP). Poiché Verified Permissions è un servizio completamente gestito, non è necessario gestire l'infrastruttura sottostante. Puoi interagire con le autorizzazioni verificate utilizzando un'API RESTful o l'SDK. AWS

Puoi anche utilizzare questa architettura con motori di policy personalizzati. Tuttavia, tutti i vantaggi ottenuti dalle autorizzazioni verificate devono essere sostituiti con la logica fornita dal motore di policy personalizzato.

Un PDP centralizzato con PEP sulle API offre un'opzione semplice per creare un solido sistema di autorizzazione per le API. Ciò semplifica il processo di autorizzazione e fornisce anche un'interfaccia ripetibile per prendere decisioni di autorizzazione per API easy-to-use, microservizi, livelli di Backend for Frontend (BFF) o altri componenti dell'applicazione.

Utilizzo del Cedar SDK

Amazon Verified Permissions utilizza il linguaggio Cedar per gestire autorizzazioni dettagliate nelle tue applicazioni personalizzate. Con Verified Permissions, puoi archiviare le policy Cedar in una posizione centrale, sfruttare la bassa latenza con elaborazione in millisecondi e controllare le autorizzazioni per diverse applicazioni. Opzionalmente, puoi anche integrare l'SDK Cedar direttamente nella tua applicazione per fornire decisioni di autorizzazione senza utilizzare le autorizzazioni verificate. Questa opzione richiede lo sviluppo di applicazioni personalizzate aggiuntive per gestire e archiviare le politiche per il vostro caso d'uso. Tuttavia, può essere una valida alternativa, in particolare nei casi in cui l'accesso alle autorizzazioni verificate è intermittente o non è possibile a causa di una connettività Internet incoerente.

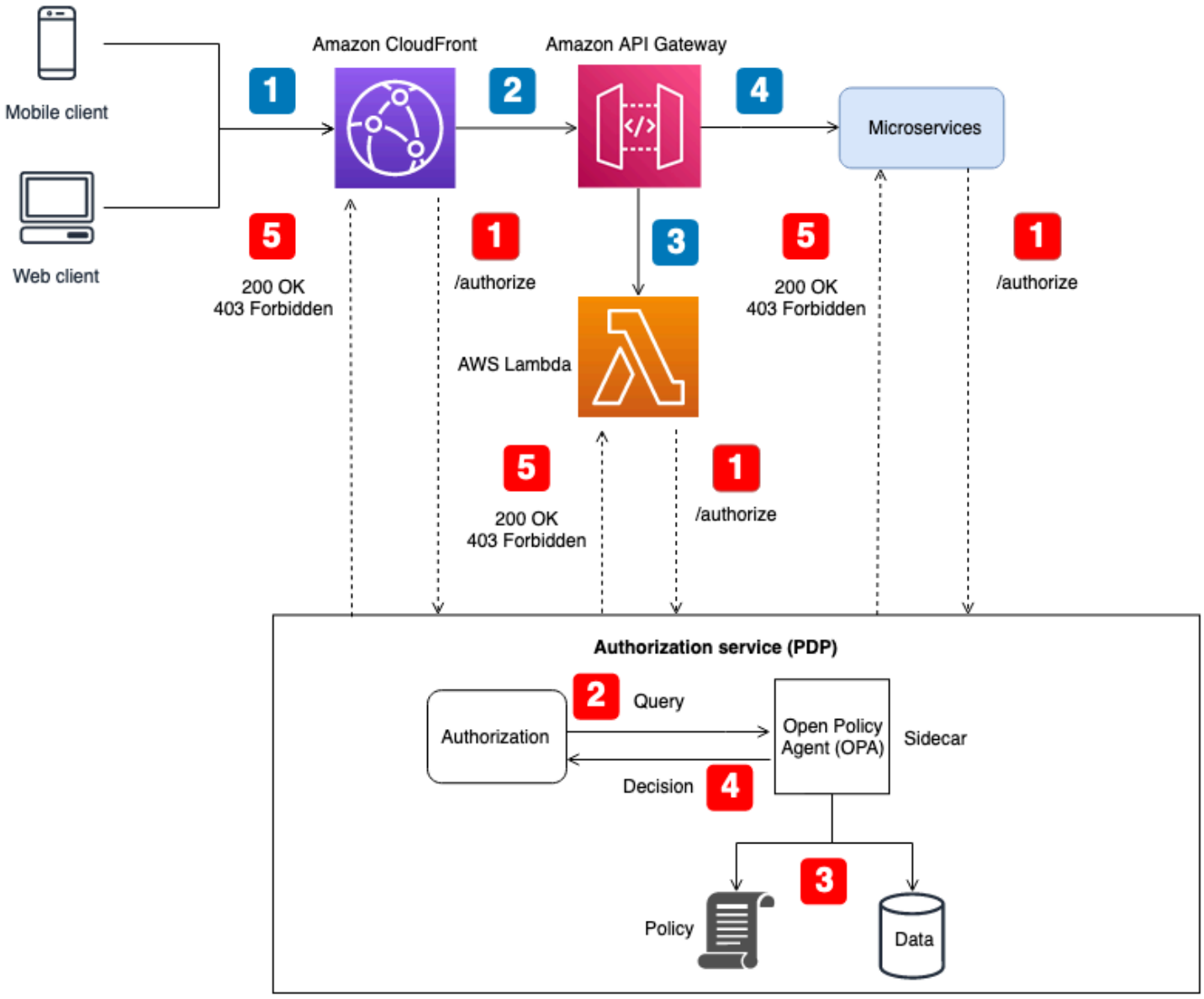
Modelli di progettazione per OPA

Utilizzo di un PDP centralizzato con PEP su API

Il modello PDP (Policy Decision Point) centralizzato con punti di applicazione delle policy (PEP) sulle API segue le migliori pratiche del settore per creare un sistema efficace e di facile manutenzione per il controllo e l'autorizzazione degli accessi alle API. Questo approccio supporta diversi principi chiave:

- L'autorizzazione e il controllo dell'accesso alle API vengono applicati in più punti dell'applicazione.
- La logica di autorizzazione è indipendente dall'applicazione.
- Le decisioni sul controllo degli accessi sono centralizzate.

Questo modello utilizza un PDP centralizzato per prendere decisioni di autorizzazione. I PEP sono implementati in tutte le API per effettuare richieste di autorizzazione al PDP. Il diagramma seguente mostra come implementare questo modello in un'ipotetica applicazione SaaS multi-tenant.



Flusso dell'applicazione (illustrato con callout numerati in blu nel diagramma):

1. Un utente autenticato con un JWT genera una richiesta HTTP ad Amazon. CloudFront
2. CloudFront inoltra la richiesta ad Amazon API Gateway, che è configurato come CloudFront origine.
3. Viene chiamato un autorizzatore personalizzato API Gateway per verificare il JWT.
4. I microservizi rispondono alla richiesta.

Flusso di autorizzazione e controllo dell'accesso alle API (illustrato con callout numerati in rosso nel diagramma):

1. Il PEP chiama il servizio di autorizzazione e trasmette i dati della richiesta, inclusi eventuali JWT.
2. Il servizio di autorizzazione (PDP) prende i dati della richiesta e interroga un'API REST dell'agente OPA, che viene eseguita come sidecar. I dati della richiesta servono come input per la query.
3. L'OPA valuta l'input in base alle politiche pertinenti specificate dalla query. I dati vengono importati per prendere una decisione di autorizzazione, se necessario.
4. L'OPA restituisce una decisione al servizio di autorizzazione.
5. La decisione di autorizzazione viene restituita al PEP e valutata.

In questa architettura, i PEP richiedono decisioni di autorizzazione sugli endpoint del servizio per Amazon CloudFront e Amazon API Gateway e per ogni microservizio. La decisione di autorizzazione viene presa da un servizio di autorizzazione (il PDP) con un sidecar OPA. È possibile utilizzare questo servizio di autorizzazione come contenitore o come istanza di server tradizionale. Il sidecar OPA espone la propria API RESTful localmente in modo che l'API sia accessibile solo al servizio di autorizzazione. Il servizio di autorizzazione espone un'API separata disponibile per i PEP. Il fatto che il servizio di autorizzazione funga da intermediario tra PEP e OPA consente l'inserimento di qualsiasi logica di trasformazione tra PEP e OPA che possa essere necessaria, ad esempio quando la richiesta di autorizzazione di un PEP non è conforme all'input di query previsto dall'OPA.

È inoltre possibile utilizzare questa architettura con motori di policy personalizzati. Tuttavia, tutti i vantaggi ottenuti dall'OPA devono essere sostituiti con la logica fornita dal motore di policy personalizzato.

Un PDP centralizzato con PEP sulle API offre un'opzione semplice per creare un solido sistema di autorizzazione per le API. È semplice da implementare e fornisce anche un'interfaccia ripetibile per prendere decisioni di autorizzazione per API easy-to-use, microservizi, livelli di Backend for Frontend (BFF) o altri componenti dell'applicazione. Tuttavia, questo approccio potrebbe creare troppa latenza nell'applicazione, poiché le decisioni di autorizzazione richiedono la chiamata di un'API separata. Se la latenza di rete è un problema, potreste prendere in considerazione un PDP distribuito.

Utilizzo di un PDP distribuito con PEP su API

Il modello Distributed Policy Decision Point (PDP) con Policy Enforcement Points (PEP) sulle API segue le migliori pratiche del settore per creare un sistema efficace per il controllo e l'autorizzazione

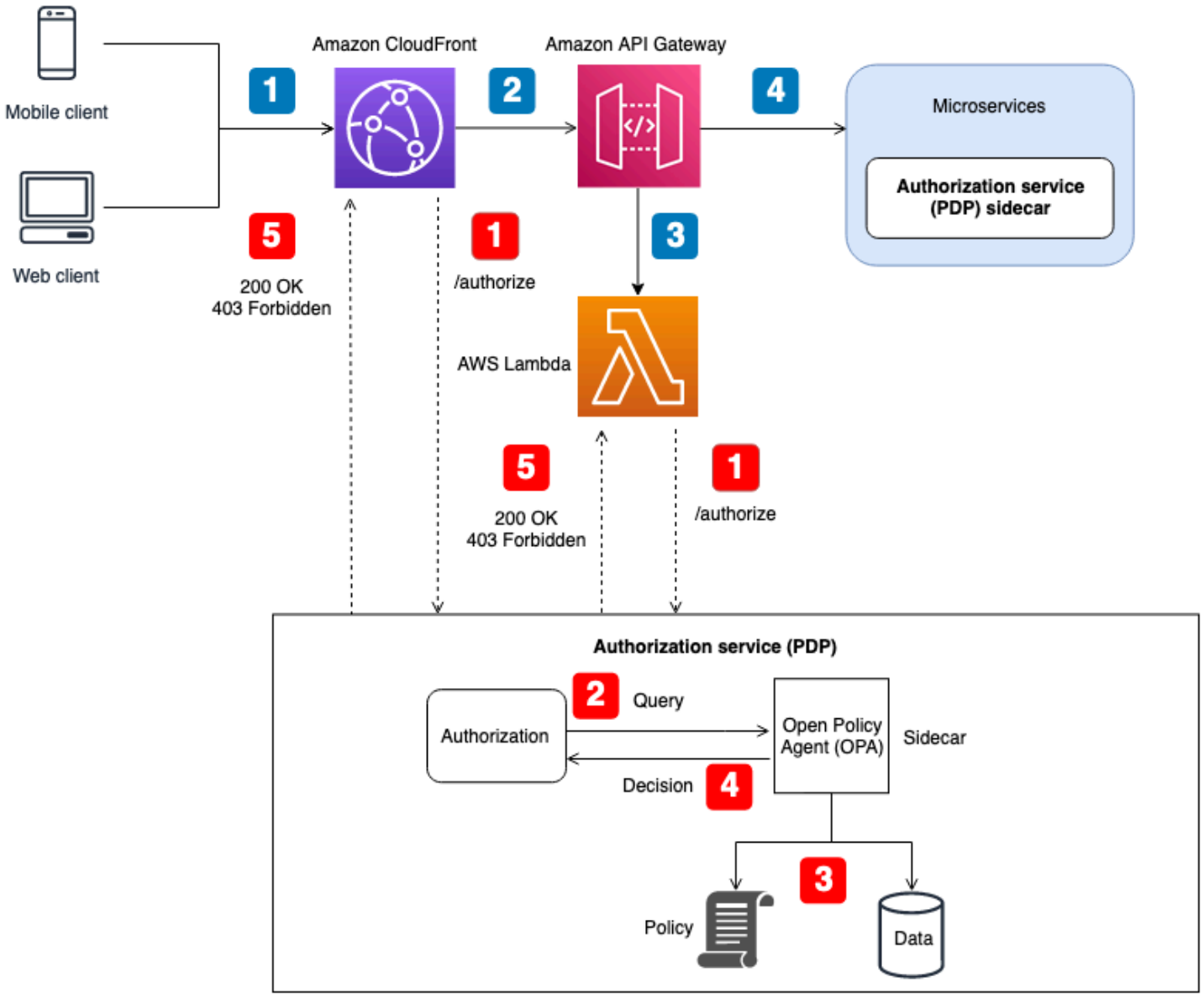
degli accessi alle API. Analogamente al modello PDP centralizzato con PEP su API, questo approccio supporta i seguenti principi chiave:

- L'autorizzazione e il controllo dell'accesso alle API vengono applicati in più punti dell'applicazione.
- La logica di autorizzazione è indipendente dall'applicazione.
- Le decisioni sul controllo degli accessi sono centralizzate.

Potresti chiederti perché le decisioni sul controllo degli accessi sono centralizzate quando il PDP viene distribuito. Sebbene il PDP possa esistere in più punti di un'applicazione, deve utilizzare la stessa logica di autorizzazione per prendere decisioni sul controllo degli accessi. Tutti i PDP forniscono le stesse decisioni di controllo degli accessi con gli stessi input. I PEP sono implementati in tutte le API per effettuare richieste di autorizzazione al PDP. La figura seguente mostra come questo modello distribuito può essere implementato in un'ipotetica applicazione SaaS multi-tenant.

In questo approccio, i PDP vengono implementati in più punti dell'applicazione. Per i componenti applicativi che dispongono di funzionalità di elaborazione integrate in grado di eseguire OPA e supportare un PDP, come un servizio containerizzato con un sidecar o un'istanza Amazon Elastic Compute Cloud (Amazon EC2), le decisioni PDP possono essere integrate direttamente nel componente dell'applicazione senza dover effettuare una chiamata API RESTful a un servizio PDP centralizzato. Ciò ha il vantaggio di ridurre la latenza che si potrebbe riscontrare nel modello PDP centralizzato, poiché non tutti i componenti dell'applicazione devono effettuare chiamate API aggiuntive per ottenere decisioni di autorizzazione. Tuttavia, in questo modello è ancora necessario un PDP centralizzato per i componenti applicativi che non dispongono di funzionalità di elaborazione integrate che consentano l'integrazione diretta di un PDP, come il servizio Amazon o Amazon CloudFront API Gateway.

Il diagramma seguente mostra come questa combinazione di un PDP centralizzato e un PDP distribuito può essere implementata in un'ipotetica applicazione SaaS multi-tenant.



Flusso dell'applicazione (illustrato con callout numerati in blu nel diagramma):

1. Un utente autenticato con un JWT genera una richiesta HTTP ad Amazon. CloudFront
2. CloudFront inoltra la richiesta ad Amazon API Gateway, che è configurato come CloudFront origine.
3. Viene chiamato un autorizzatore personalizzato API Gateway per verificare il JWT.
4. I microservizi rispondono alla richiesta.

Flusso di autorizzazione e controllo dell'accesso alle API (illustrato con callout numerati in rosso nel diagramma):

1. Il PEP chiama il servizio di autorizzazione e trasmette i dati della richiesta, inclusi eventuali JWT.
2. Il servizio di autorizzazione (PDP) prende i dati della richiesta e interroga un'API REST dell'agente OPA, che viene eseguita come sidecar. I dati della richiesta servono come input per la query.
3. L'OPA valuta l'input in base alle politiche pertinenti specificate dalla query. I dati vengono importati per prendere una decisione di autorizzazione, se necessario.
4. L'OPA restituisce una decisione al servizio di autorizzazione.
5. La decisione di autorizzazione viene restituita al PEP e valutata.

In questa architettura, i PEP richiedono decisioni di autorizzazione presso gli endpoint di servizio per CloudFront un API Gateway e per ogni microservizio. La decisione di autorizzazione per i microservizi viene presa da un servizio di autorizzazione (il PDP) che funge da complemento al componente dell'applicazione. Questo modello è possibile per microservizi (o servizi) eseguiti su contenitori o istanze Amazon Elastic Compute Cloud (Amazon EC2). Le decisioni di autorizzazione per servizi come API Gateway richiedono CloudFront comunque il contatto con un servizio di autorizzazione esterno. In ogni caso, il servizio di autorizzazione espone un'API disponibile per i PEP. Il fatto che il servizio di autorizzazione funga da intermediario tra PEP e OPA consente l'inserimento di qualsiasi logica di trasformazione tra PEP e OPA che potrebbe essere necessaria, ad esempio quando la richiesta di autorizzazione di un PEP non è conforme all'input di query previsto dall'OPA.

È inoltre possibile utilizzare questa architettura con motori di policy personalizzati. Tuttavia, tutti i vantaggi ottenuti dall'OPA devono essere sostituiti con la logica fornita dal motore di policy personalizzato.

Un PDP distribuito con PEP sulle API offre la possibilità di creare un solido sistema di autorizzazione per le API. È semplice da implementare e fornisce un'interfaccia ripetibile per prendere decisioni di autorizzazione per API easy-to-use, microservizi, livelli di Backend for Frontend (BFF) o altri componenti dell'applicazione. Questo approccio ha anche il vantaggio di ridurre la latenza che si potrebbe riscontrare nel modello PDP centralizzato.

Utilizzo di un PDP distribuito come libreria

È inoltre possibile richiedere decisioni di autorizzazione a un PDP reso disponibile come libreria o pacchetto da utilizzare all'interno di un'applicazione. OPA può essere utilizzata come libreria Go di

terze parti. Per altri linguaggi di programmazione, l'adozione di questo modello in genere significa che è necessario creare un motore di policy personalizzato.

Considerazioni sulla progettazione multi-tenant di Amazon Verified Permissions

Esistono diverse opzioni di progettazione da considerare quando si implementa l'autorizzazione utilizzando Amazon Verified Permissions in una soluzione SaaS multi-tenant. Prima di esplorare queste opzioni, chiariamo la differenza tra isolamento e autorizzazione in un contesto SaaS multi-tenant. L'isolamento di un tenant impedisce che i dati in entrata e in uscita vengano esposti al tenant sbagliato. L'autorizzazione garantisce che un utente disponga delle autorizzazioni per accedere a un tenant.

In Autorizzazioni verificate, le politiche vengono archiviate in un archivio di politiche. Come descritto nella documentazione sulle autorizzazioni verificate, è possibile isolare le politiche dei tenant utilizzando un archivio di policy separato per ogni tenant oppure consentire ai tenant di condividere le politiche utilizzando un unico archivio di politiche per tutti i tenant. Questa sezione illustra i vantaggi e gli svantaggi di queste due strategie di isolamento e descrive come possono essere implementate utilizzando un modello di distribuzione a più livelli. Per un contesto aggiuntivo, consulta la documentazione sulle autorizzazioni verificate.

Sebbene i criteri discussi in questa sezione si concentrino sulle autorizzazioni verificate, i concetti generali sono radicati nella mentalità dell'isolamento e nelle linee guida che fornisce. Le applicazioni SaaS devono sempre considerare l'isolamento dei tenant come parte della loro progettazione e questo principio generale di isolamento si estende all'inclusione delle autorizzazioni verificate in un'applicazione SaaS. Questa sezione fa riferimento anche ai principali modelli di isolamento SaaS, come il modello SaaS in silos e il modello SaaS in pool. Per ulteriori informazioni, consulta i concetti di isolamento di base in AWS Well-Architected Framework, SaaS Lens.

Le considerazioni chiave nella progettazione di soluzioni SaaS multi-tenant sono l'isolamento dei tenant e l'onboarding dei tenant. L'isolamento dei tenant influisce sulla sicurezza, sulla privacy, sulla resilienza e sulle prestazioni. L'onboarding dei tenant influisce sui processi operativi in relazione al sovraccarico operativo e all'osservabilità. Organizations che intraprendono un percorso SaaS o implementano soluzioni multi-tenant devono sempre dare la priorità al modo in cui la tenancy verrà gestita dall'applicazione SaaS. Sebbene una soluzione SaaS possa orientarsi verso un particolare modello di isolamento, la coerenza non è necessariamente richiesta nell'intera soluzione SaaS. Ad esempio, il modello di isolamento scelto per i componenti di frontend dell'applicazione potrebbe non essere lo stesso del modello di isolamento scelto per un microservizio o per i servizi di autorizzazione.

Considerazioni sulla progettazione:

- Inserimento degli inquilini e registrazione degli inquilini come utenti
- Archivio delle politiche per tenant
- Un archivio di policy multi-tenant condiviso
- Modello di implementazione a più livelli

Inserimento degli inquilini e registrazione degli inquilini come utenti

Le applicazioni SaaS rispettano il concetto di identità SaaS e seguono la best practice generale di associare un'identità utente a un'identità del tenant. L'associazione implica la memorizzazione di un identificatore del tenant come affermazione o attributo per l'utente nel provider di identità. Ciò sposta la responsabilità della mappatura delle identità dei tenant da ciascuna applicazione al processo di registrazione degli utenti. Ogni utente autenticato dispone quindi dell'identità del tenant corretta come parte del JSON Web Token (JWT).

Analogamente, la selezione del policy store corretto per una richiesta di autorizzazione non dovrebbe essere determinata dalla logica dell'applicazione. Per determinare quale archivio di policy deve essere utilizzato da una particolare richiesta di autorizzazione, gestite una mappatura degli utenti nei policy store o dei tenant nei policy store. Queste mappature vengono generalmente gestite in un archivio dati come Amazon DynamoDB o Amazon Relational Database Service (Amazon RDS) a cui fa riferimento l'applicazione. Puoi anche fornire o integrare queste mappature con i dati in un provider di identità (IdP). La relazione tra inquilini, utenti e archivi delle politiche viene quindi generalmente fornita a un utente tramite un JWT che contiene tutte le relazioni necessarie per una richiesta di autorizzazione.

Questo esempio mostra come potrebbe apparire il JWT per l'utente`Alice`, che appartiene al tenant `TenantA` e utilizza l'archivio delle politiche con l'ID dell'archivio delle politiche per l'autorizzazione. `ps-43214321`

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

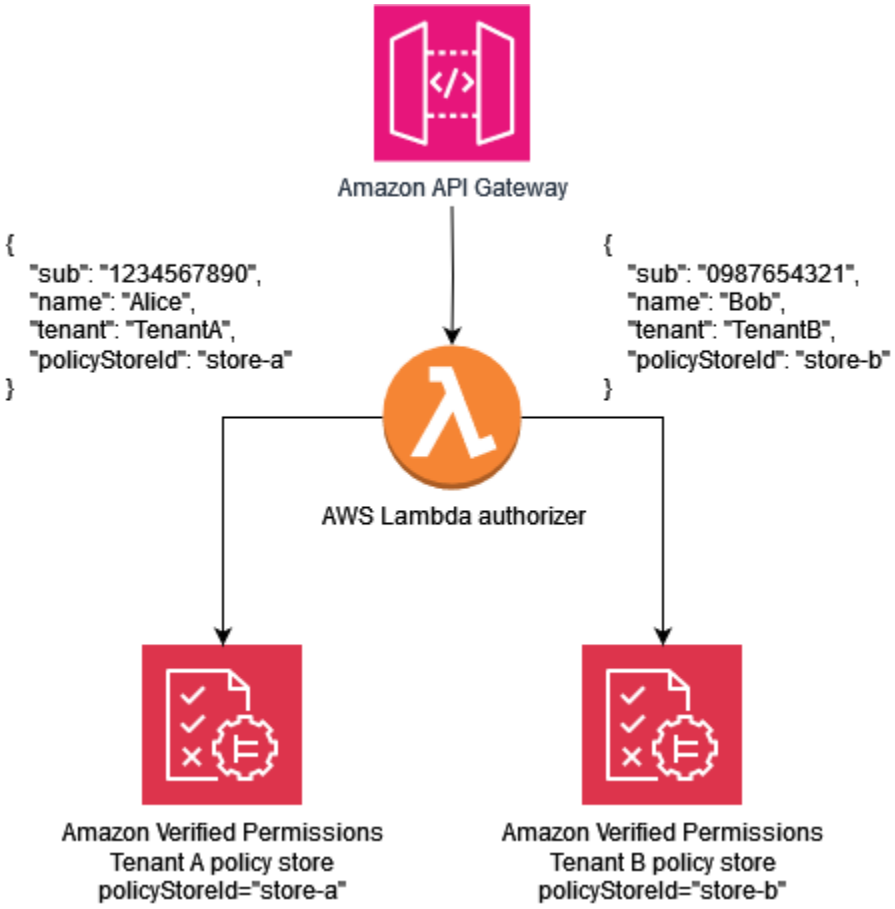
Archivio delle politiche per tenant

Il modello di progettazione dell'archivio di policy per tenant in Amazon Verified Permissions associa ogni tenant di un'applicazione SaaS al proprio archivio di politiche. Questo modello è simile al modello di isolamento dei silo SaaS. Entrambi i modelli richiedono la creazione di un'infrastruttura specifica per il locatario e presentano vantaggi e svantaggi simili. I vantaggi principali di questo approccio sono l'isolamento dei tenant basato sull'infrastruttura, il supporto per modelli di autorizzazione unici su base tenant, l'eliminazione dei problemi legati alla rumorosità dei vicini e la riduzione dell'impatto in caso di errori negli aggiornamenti o nelle implementazioni delle politiche. Gli svantaggi di questo approccio includono processi, implementazioni e operazioni di onboarding dei tenant più complessi. L'archivio delle politiche per tenant è l'approccio consigliato se la soluzione dispone di policy uniche per tenant.

Il modello di archivio delle politiche per tenant può fornire un approccio altamente suddiviso in silos all'isolamento dei tenant, se l'applicazione SaaS lo richiede. È possibile utilizzare questo modello anche con l'isolamento del pool, ma l'implementazione delle autorizzazioni verificate non condividerà i vantaggi standard del più ampio modello di isolamento del pool, come la gestione e le operazioni semplificate.

In un archivio di policy per tenant, l'isolamento del tenant si ottiene mappando l'identificatore del policy store del tenant all'identità SaaS dell'utente durante il processo di registrazione dell'utente, come discusso in precedenza. Questo approccio lega fortemente l'archivio delle politiche del tenant all'utente principale e fornisce un modo coerente per condividere la mappatura nell'intera soluzione SaaS. Puoi fornire la mappatura a un'applicazione SaaS mantenendola come parte di un IdP o in un'origine dati esterna come DynamoDB. Ciò garantisce inoltre che il committente faccia parte del tenant e che venga utilizzato l'archivio delle politiche del tenant.

Questo esempio mostra come il JWT che contiene l'`policyStoreId`and `tenant` di un utente viene passato dall'endpoint di un'API al punto di valutazione delle politiche in un AWS Lambda autorizzatore, che indirizza la richiesta al policy store corretto.



La seguente politica di esempio illustra il paradigma di progettazione del policy store per tenant. L'utente `Alice` appartiene a `TenantA`. The policyStoreId `store-a` viene inoltre mappato all'identità del tenant di `Alice`, e impone l'uso del policy store corretto. Ciò garantisce che vengano utilizzate le politiche di. `TenantA`

> **Note**
>
> Il modello di archivio delle politiche per inquilino isola le politiche degli inquilini. L'autorizzazione impone le azioni che gli utenti sono autorizzati a eseguire sui propri dati. Le risorse coinvolte in qualsiasi applicazione ipotetica che utilizza questo modello devono essere isolate utilizzando altri meccanismi di isolamento, come definito nella documentazione di AWS Well-Architected Framework, SaaS Lens.

In questa politica, `Alice` dispone delle autorizzazioni per visualizzare i dati di tutte le risorse.

```
permit (
```

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

Per effettuare una richiesta di autorizzazione e avviare una valutazione con una politica di autorizzazioni verificate, è necessario fornire l'ID dell'archivio delle politiche che corrisponde all'ID univoco mappato al tenant,. `store-a`

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

```
         ]
      ]
   }
}
```

L'utente Bob appartiene al Tenant B ed policyStoreId `store-b` è inoltre mappato all'identità del tenant diBob, il che impone l'uso del policy store corretto. Ciò garantisce che vengano utilizzate le politiche del Tenant B.

In questa politica, Bob dispone delle autorizzazioni per personalizzare i dati di tutte le risorse. In questo esempio, `customizeData` potrebbe trattarsi di un'azione specifica solo per il Tenant B, quindi la politica sarebbe unica per il Tenant B. Il modello di archivio delle politiche per tenant supporta intrinsecamente politiche personalizzate su base per tenant.

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

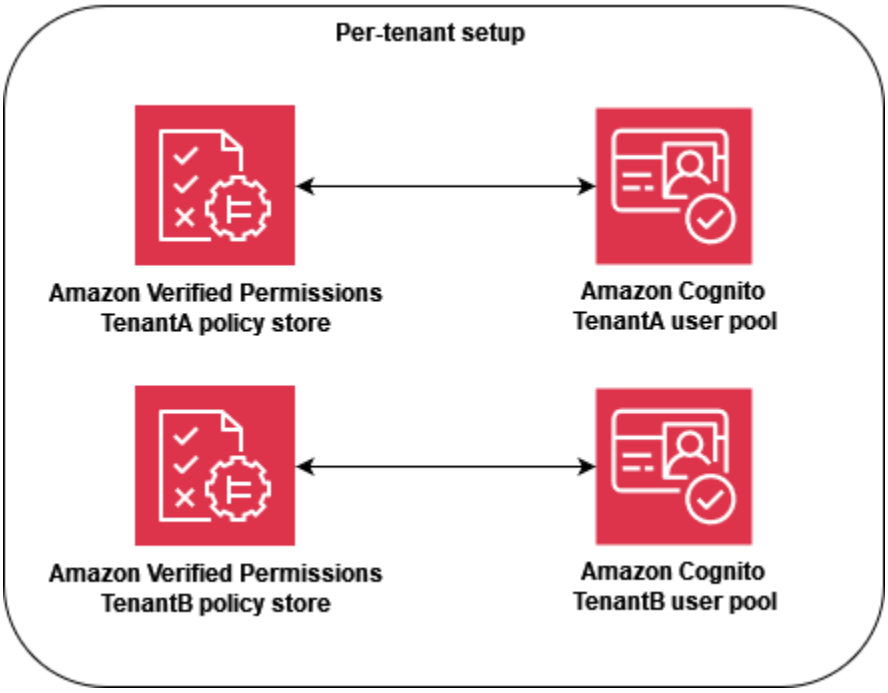
Per effettuare una richiesta di autorizzazione e avviare una valutazione con una politica di autorizzazioni verificate, è necessario fornire l'ID dell'archivio delle politiche che corrisponde all'ID univoco mappato al tenant,. `store-b`

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

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

Con Verified Permissions, è possibile, ma non necessario, integrare un IdP con un policy store. Questa integrazione consente alle politiche di fare esplicito riferimento al principale nell'archivio di identità come principale delle politiche. Per ulteriori informazioni su come effettuare l'integrazione con Amazon Cognito come IdP per autorizzazioni verificate, consulta la documentazione sulle autorizzazioni verificate e la documentazione di Amazon Cognito.

Quando si integra un policy store con un IdP, è possibile utilizzare solo una fonte di identità per policy store. Ad esempio, se scegli di integrare le autorizzazioni verificate con Amazon Cognito, devi rispecchiare la strategia utilizzata per l'isolamento dei tenant degli archivi di policy di Autorizzazioni verificate e dei pool di utenti di Amazon Cognito. Inoltre, gli archivi delle policy e i pool di utenti devono essere gli stessi. Account AWS



A livello operativo, l'archivio delle politiche per tenant offre un vantaggio in termini di controllo, in quanto è possibile interrogare facilmente l'attività registrata in modo indipendente per ogni tenant.AWS CloudTrail Tuttavia, ti consigliamo comunque di registrare metriche personalizzate aggiuntive su una dimensione per-tenant su Amazon. CloudWatch

L'approccio per-tenant policy store richiede inoltre un'attenzione particolare a due quote di autorizzazioni verificate per garantire che non interferiscano con le operazioni della soluzione SaaS. Queste quote sono Policy store per regione per account e IsAuthorized richieste al secondo per regione per account. Puoi richiedere aumenti per entrambe le quote.

Per un esempio più dettagliato di come implementare il modello per-tenant policy store, consulta il AWS post sul blog Controllo degli accessi SaaS usando Amazon Verified Permissions with a per-tenant policy store.

Un archivio di policy multi-tenant condiviso

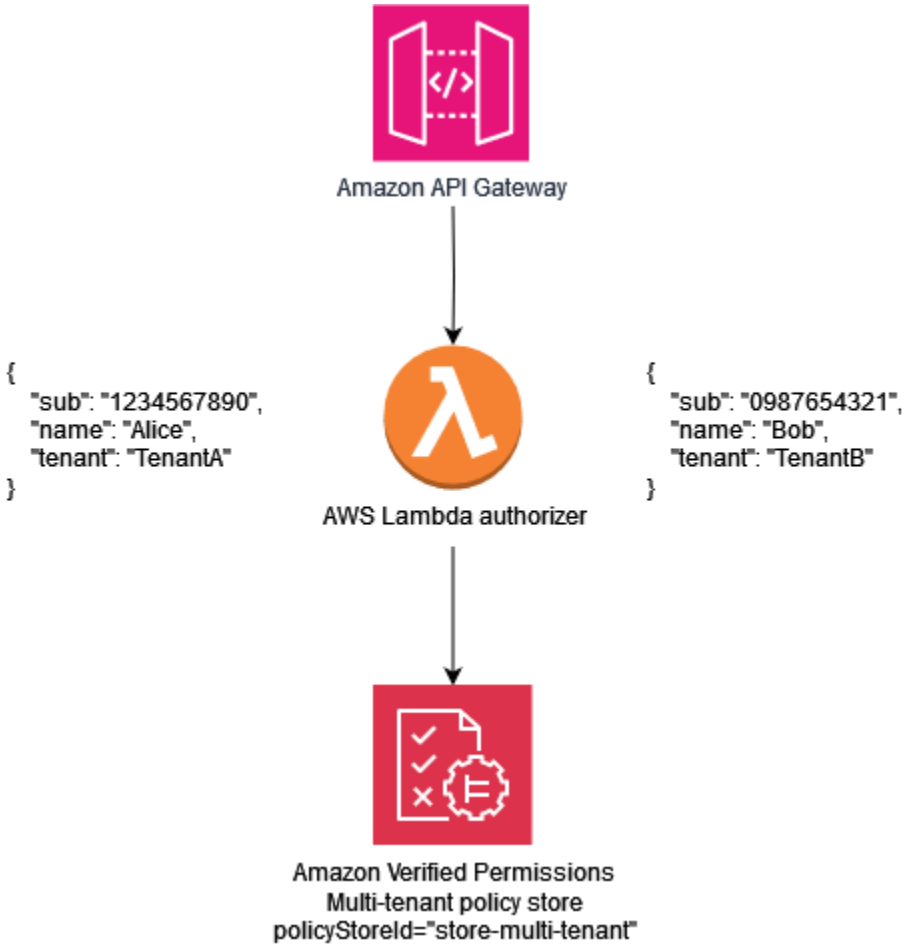
Il modello di progettazione di un unico archivio di policy multi-tenant condiviso utilizza un unico archivio di policy multi-tenant in Amazon Verified Permissions per tutti i tenant della soluzione SaaS. Il vantaggio principale di questo approccio è la gestione e le operazioni semplificate, in particolare perché non è necessario creare archivi di policy aggiuntivi durante l'onboarding dei tenant. Gli svantaggi di questo approccio includono una maggiore portata dell'impatto derivante da eventuali

guasti o errori negli aggiornamenti o nelle implementazioni delle policy e una maggiore esposizione agli effetti rumorosi dei vicini. Inoltre, non consigliamo questo approccio se la soluzione richiede politiche uniche per ogni tenant. In questo caso, utilizza invece il modello di archivio delle politiche per tenant per garantire che vengano utilizzate le politiche del tenant corretto.

L'approccio One shared multi-tenant policy store è simile al modello di isolamento in pool SaaS. Può fornire un approccio condiviso all'isolamento dei tenant, se l'applicazione SaaS lo richiede. Puoi utilizzare questo modello anche se la tua soluzione SaaS applica l'isolamento in silos ai suoi microservizi. Quando si sceglie un modello, è necessario valutare i requisiti per l'isolamento dei dati dei tenant e la struttura delle politiche di autorizzazione verificate necessarie per un'applicazione SaaS in modo indipendente.

Per applicare un modo coerente di condividere l'identificatore del tenant sull'intera soluzione SaaS, è buona norma mappare l'identificatore all'identità SaaS dell'utente durante la registrazione dell'utente, come discusso in precedenza. Puoi fornire questa mappatura a un'applicazione SaaS mantenendola come parte di un IdP o in un'origine dati esterna come DynamoDB. Si consiglia inoltre di mappare l'ID condiviso del policy store agli utenti. Sebbene l'ID non venga utilizzato come parte dell'isolamento degli inquilini, si tratta di una buona pratica perché facilita le modifiche future.

L'esempio seguente mostra come l'endpoint API invia un JWT agli utenti `Alice` che appartengono a tenant diversi ma condividono il policy store con l'ID del policy store per l'autorizzazione. `Bob` `store-multi-tenant` Poiché tutti i tenant condividono un unico archivio di politiche, non è necessario mantenere l'ID del policy store in un token o database. Poiché tutti i tenant condividono un unico ID del policy store, è possibile fornire l'ID come variabile di ambiente che l'applicazione può utilizzare per effettuare chiamate al policy store.



La seguente policy di esempio illustra il paradigma di progettazione di policy multi-tenant condiviso. In questa politica, il principale `MultiTenantApp::User` che possiede il genitore `MultiTenantApp::Role` `Admin` dispone delle autorizzazioni per visualizzare i dati di tutte le risorse.

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

Poiché è in uso un unico archivio di politiche, l'archivio delle politiche Verified Permissions deve garantire che un attributo di tenancy associato al principale corrisponda all'attributo di tenancy associato alla risorsa. Ciò può essere ottenuto includendo la seguente policy nel policy store, per garantire che tutte le richieste di autorizzazione che non hanno attributi di tenancy corrispondenti sulla risorsa e sul principale vengano rifiutate.

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

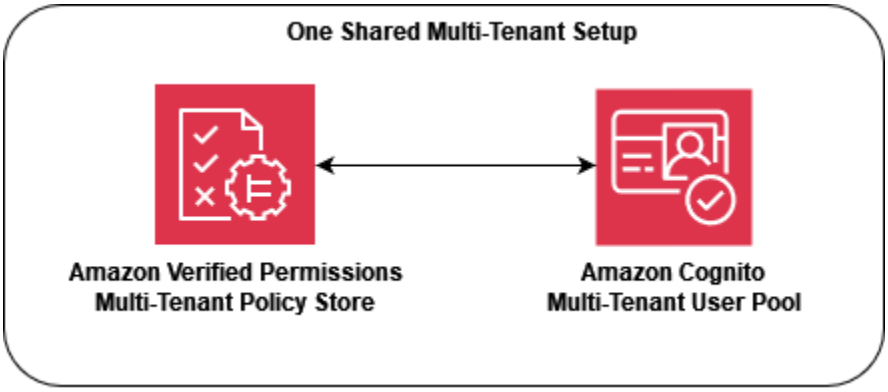
Per una richiesta di autorizzazione che utilizza un modello di policy store multi-tenant condiviso, l'ID del policy store è l'identificatore del policy store condiviso. Nella richiesta seguente, le `User Alice` è consentito l'accesso perché ha un `Role` of e `Admin` gli `Tenant` attributi associati alla risorsa e al principale sono entrambi. `TenantA`

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

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

Con Verified Permissions, è possibile, ma non necessario, integrare un IdP con un policy store. Questa integrazione consente alle politiche di fare esplicito riferimento al principale nell'archivio di identità come principale delle politiche. Per ulteriori informazioni su come effettuare l'integrazione con Amazon Cognito come IdP per autorizzazioni verificate, consulta la documentazione sulle autorizzazioni verificate e la documentazione di Amazon Cognito.

Quando si integra un policy store con un IdP, è possibile utilizzare solo una fonte di identità per policy store. Ad esempio, se scegli di integrare le autorizzazioni verificate con Amazon Cognito, devi rispecchiare la strategia utilizzata per l'isolamento dei tenant degli archivi di policy di Autorizzazioni verificate e dei pool di utenti di Amazon Cognito. Inoltre, gli archivi delle policy e i pool di utenti devono essere gli stessi. Account AWS



Dal punto di vista operativo e di controllo, il modello di archivio di policy multi-tenant condiviso presenta uno svantaggio in quanto l'attività registrata AWS CloudTrail richiede un maggior numero di interrogazioni complesse per filtrare le singole attività sul tenant, poiché ogni chiamata registrata utilizza lo stesso archivio di politiche. CloudTrail In questo scenario, è utile registrare su Amazon CloudWatch metriche personalizzate aggiuntive su una dimensione per-tenant per garantire un livello adeguato di osservabilità e capacità di controllo.

L'approccio «one shared multi-tenant policy store» richiede inoltre un'attenzione particolare alle quote di autorizzazioni verificate per garantire che non interferiscano con le operazioni della soluzione SaaS. In particolare, ti consigliamo di monitorare IsAuthorized le richieste al secondo per regione per quota di account per assicurarti che non vengano superati i limiti. Puoi richiedere un aumento di questa quota.

Modello di implementazione a più livelli

Creando un modello di distribuzione a più livelli, puoi isolare i tenant «Enterprise Tier» ad alta priorità dal volume potenzialmente più elevato di clienti «Standard Tier». In questo modello, è possibile implementare tutte le modifiche implementate alle policy negli archivi delle policy separatamente per ogni livello, isolando così ogni livello di clienti dalle modifiche apportate al di fuori del proprio livello. Nel modello di implementazione a più livelli, gli archivi delle politiche vengono in genere creati come parte del provisioning iniziale dell'infrastruttura per ogni livello anziché essere implementati quando un tenant è integrato.

Se la soluzione utilizza principalmente un modello di isolamento in pool, potrebbe essere necessario un isolamento o una personalizzazione aggiuntivi. Ad esempio, è possibile creare un «livello Premium» in cui ogni tenant disporrebbe della propria infrastruttura a livello di tenant, il che crea un modello a silos implementando un'istanza in pool con un solo tenant. Ciò potrebbe assumere la forma

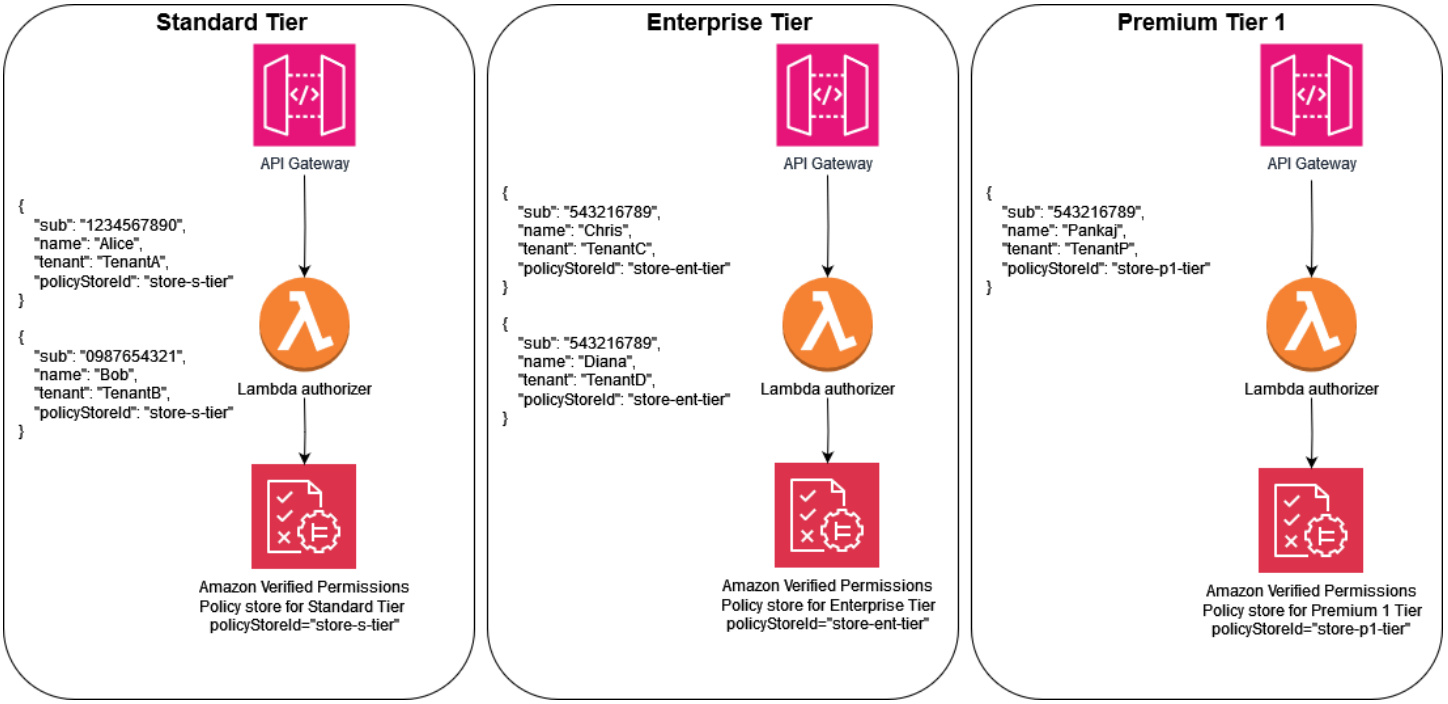
di infrastrutture «Premium Tier Tenant A» e «Premium Tier Tenant B» completamente separate, compresi gli archivi delle polizze. Questo approccio si traduce in un modello di isolamento isolato per i clienti di più alto livello.

Nel modello di implementazione a più livelli, ogni archivio di politiche deve seguire lo stesso modello di isolamento, sebbene sia distribuito separatamente. Poiché vengono utilizzati più archivi di policy, è necessario applicare un modo coerente di condivisione dell'identificatore dell'archivio delle politiche associato al tenant nell'intera soluzione SaaS. Come per il modello di archivio delle politiche per tenant, è buona norma mappare l'identificatore del tenant all'identità SaaS dell'utente durante la registrazione dell'utente.

Il diagramma seguente mostra tre livelli:, e. `Standard Tier` `Enterprise Tier` `Premium Tier 1` Ogni livello viene distribuito separatamente nella propria infrastruttura e utilizza un archivio di policy condiviso all'interno del livello. I livelli Standard ed Enterprise contengono più tenant. `TenantA`e `TenantB` rientrano nel`Standard Tier`, `TenantC` e `TenantD` sono nel livello Enterprise.

`Premium Tier 1`contiene solo`TenantP`, in modo da poter servire il tenant premium come se la soluzione avesse un modello di isolamento completamente isolato e fornire funzionalità come politiche personalizzate. L'onboarding di un nuovo cliente di livello premium comporterebbe la creazione di un'infrastruttura. `Premium Tier 2`

> **Note**
>
> L'applicazione, l'implementazione e l'onboarding dei tenant nel livello premium sono identici ai livelli standard ed enterprise. L'unica differenza è che il flusso di lavoro di onboarding di livello premium inizia con la fornitura di una nuova infrastruttura di livello.


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

Considerazioni sulla progettazione multi-tenant OPA

L'Open Policy Agent (OPA) è un servizio flessibile che può essere applicato a numerosi casi d'uso in cui le applicazioni sono necessarie per prendere decisioni politiche e autorizzazioni. L'utilizzo di OPA con applicazioni SaaS multi-tenant richiede la considerazione di criteri unici per garantire che le migliori pratiche SaaS chiave, come l'isolamento dei tenant, rimangano parte dell'implementazione di OPA. Questi criteri includono i modelli di implementazione OPA, l'isolamento dei tenant e il modello documentale OPA e l'onboarding dei tenant. Ciascuno di questi fattori influisce sulla progettazione ottimale dell'OPA per quanto riguarda le applicazioni multi-tenant.

Sebbene la discussione in questa sezione si concentri sull'OPA, i concetti generali sono radicati nella mentalità isolante e nelle linee guida che fornisce. Le applicazioni SaaS devono sempre considerare l'isolamento dei tenant come parte della loro progettazione e questo principio generale di isolamento si estende all'inclusione dell'OPA in un'applicazione SaaS. L'OPA, se usato in modo appropriato, può essere una parte fondamentale del modo in cui l'isolamento viene applicato nelle applicazioni SaaS. Questa sezione fa riferimento anche ai principali modelli di isolamento SaaS, come il modello SaaS in silos e il modello SaaS in pool. Per ulteriori informazioni, consulta i concetti di isolamento di base in AWS Well-Architected Framework, SaaS Lens.

Considerazioni sulla progettazione:

- Confronto tra modelli di implementazione centralizzati e distribuiti
- Isolamento dei tenant con il modello di documento OPA
- Inserimento degli inquilini

Confronto tra modelli di implementazione centralizzati e distribuiti

È possibile implementare OPA secondo uno schema di distribuzione centralizzato o distribuito e il metodo ideale per un'applicazione multi-tenant dipende dal caso d'uso. Per esempi di questi modelli, vedere le sezioni Utilizzo di un PDP centralizzato con PEP sulle API e Utilizzo di un PDP distribuito e PEP sulle API più avanti di questa guida. Poiché OPA può essere distribuito come demone in un sistema operativo o contenitore, può essere implementato in diversi modi per supportare un'applicazione multi-tenant.

In uno schema di distribuzione centralizzato, OPA viene distribuito come contenitore o daemon con la sua API RESTful disponibile per altri servizi dell'applicazione. Quando un servizio richiede una

decisione da parte dell'OPA, viene chiamata l'API OPA RESTful centrale per produrre tale decisione. Questo approccio è semplice da implementare e gestire, poiché esiste una sola implementazione di OPA. L'aspetto negativo di questo approccio è che non fornisce alcun meccanismo per mantenere la separazione dei dati degli inquilini. Poiché esiste una sola implementazione dell'OPA, tutti i dati dei tenant utilizzati in una decisione OPA, inclusi i dati esterni a cui fa riferimento l'OPA, devono essere disponibili per OPA. È possibile mantenere l'isolamento dei dati dei tenant con questo approccio, ma deve essere applicato dalla politica e dalla struttura dei documenti dell'OPA o dall'accesso ai dati esterni. Un modello di distribuzione centralizzato richiede anche una latenza più elevata, poiché ogni decisione di autorizzazione deve effettuare una chiamata API RESTful a un altro servizio.

In un modello di distribuzione distribuito, l'OPA viene distribuito come contenitore o daemon insieme ai servizi dell'applicazione multi-tenant. Potrebbe essere distribuito come contenitore secondario o come daemon che viene eseguito localmente sul sistema operativo. Per recuperare una decisione dall'OPA, il servizio effettua una chiamata API RESTful alla distribuzione OPA locale. (Poiché OPA può essere distribuito come pacchetto Go, puoi utilizzare Go in modo nativo per recuperare una decisione invece di utilizzare una chiamata API RESTful.) A differenza del modello di distribuzione centralizzato, il modello distribuito richiede uno sforzo molto più consistente per l'implementazione, la manutenzione e l'aggiornamento perché è presente in più aree dell'applicazione. Un vantaggio del modello di distribuzione distribuito è la capacità di mantenere l'isolamento dei dati dei tenant, in particolare per le applicazioni che utilizzano un modello SaaS in silos. I dati specifici del tenant possono essere isolati nelle distribuzioni OPA specifiche per quel tenant, poiché l'OPA in un modello distribuito viene distribuito insieme al tenant. Inoltre, un modello di distribuzione distribuito ha una latenza molto inferiore rispetto a un modello di distribuzione centralizzato, poiché ogni decisione di autorizzazione può essere presa localmente.

Quando scegliete un modello di distribuzione OPA nella vostra applicazione multi-tenant, assicuratevi di valutare i criteri più critici per la vostra applicazione. Se l'applicazione multi-tenant è sensibile alla latenza, un modello di distribuzione distribuito offre prestazioni migliori a scapito di implementazioni e manutenzioni più complesse. Sebbene sia possibile gestire parte di questa complessità tramite DevOps l'automazione, richiede comunque uno sforzo aggiuntivo rispetto a un modello di implementazione centralizzato.

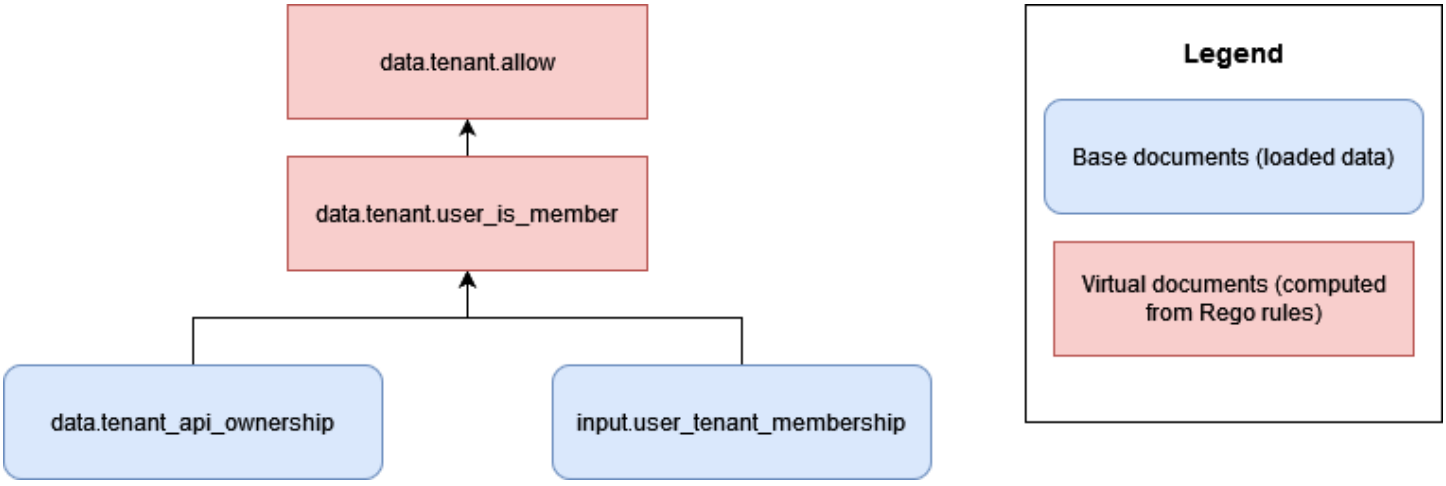
Se la tua applicazione multi-tenant utilizza un modello SaaS in silos, puoi utilizzare un modello di distribuzione OPA distribuito per imitare l'approccio in silos all'isolamento dei dati dei tenant. Questo perché, quando OPA viene eseguito insieme a ciascun servizio applicativo specifico del tenant, è possibile personalizzare ogni distribuzione OPA in modo che contenga solo i dati associati a quel tenant. Non è possibile isolare i dati OPA in un modello di distribuzione OPA centralizzato. Se si

utilizza un modello di distribuzione centralizzato o un modello distribuito insieme a un modello SaaS in pool, l'isolamento dei dati dei tenant deve essere mantenuto nel modello di documento OPA.

Isolamento dei tenant con il modello di documento OPA

OPA utilizza i documenti per prendere decisioni. Questi documenti possono contenere dati specifici del tenant, quindi è necessario considerare come mantenere l'isolamento dei dati degli inquilini. I documenti OPA sono costituiti da documenti di base e documenti virtuali. I documenti di base contengono dati provenienti dal mondo esterno. Ciò include i dati forniti direttamente all'OPA, i dati sulla richiesta OPA e i dati che potrebbero essere passati all'OPA come input. I documenti virtuali vengono calcolati in base a criteri e includono politiche e regole OPA. Per ulteriori informazioni, consulta la documentazione OPA.

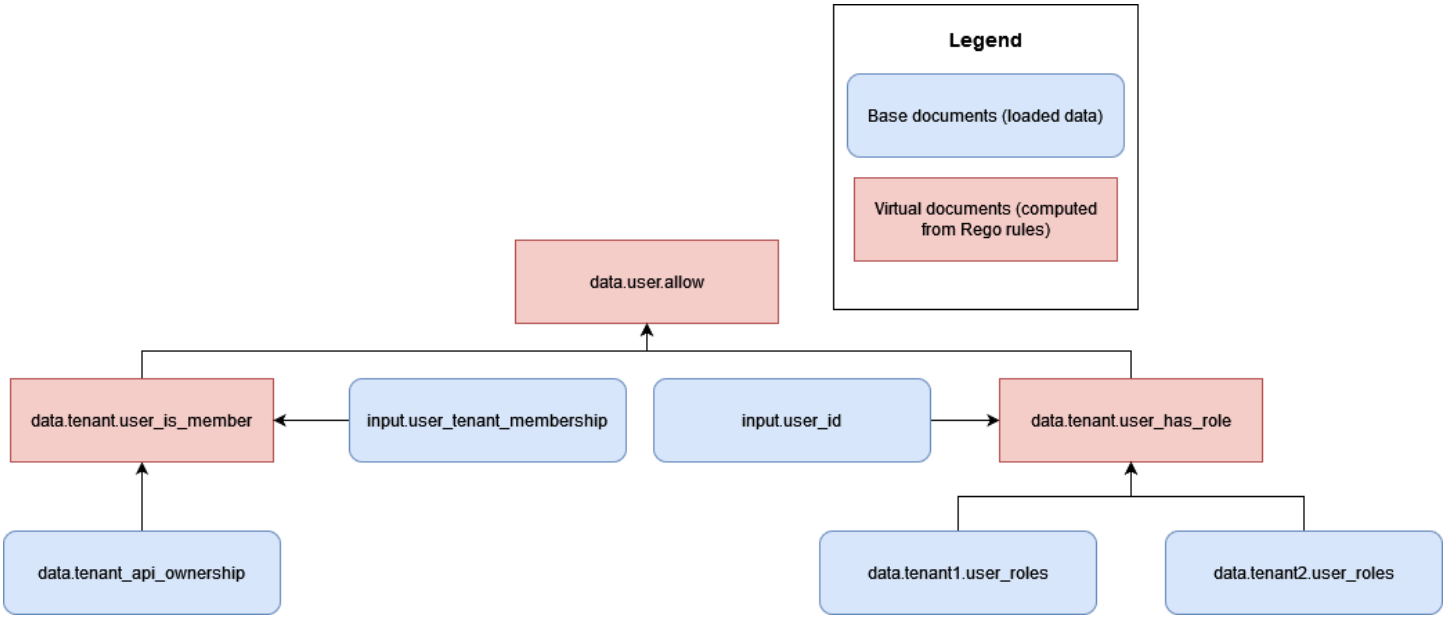
Per progettare un modello di documento in OPA per un'applicazione multi-tenant, è necessario innanzitutto considerare il tipo di documenti di base necessari per prendere una decisione in OPA. Se questi documenti di base contengono dati specifici del tenant, è necessario adottare misure per garantire che tali dati non vengano esposti accidentalmente all'accesso tra tenant. Fortunatamente, in molti casi, i dati specifici del locatario non sono necessari per prendere una decisione in OPA. L'esempio seguente mostra un ipotetico modello di documento OPA che consente l'accesso a un'API in base al tenant proprietario dell'API e al fatto che l'utente sia un membro del tenant, come indicato nel documento di input.



In questo approccio, OPA non ha accesso a nessun dato specifico del tenant, ad eccezione delle informazioni su quali inquilini possiedono un'API. In questo caso, non vi è alcuna preoccupazione che l'OPA faciliti l'accesso tra tenant, poiché le uniche informazioni che OPA utilizza per prendere una decisione di accesso sono l'associazione dell'utente con un tenant e l'associazione del tenant con

le API. È possibile applicare questo tipo di modello di documento OPA a un modello SaaS in silos, poiché ogni tenant avrebbe la proprietà di risorse indipendenti.

Tuttavia, in molti approcci di autorizzazione RBAC, esiste la possibilità di un'esposizione delle informazioni tra diversi tenant. Nell'esempio seguente, un ipotetico modello di documento OPA consente l'accesso a un'API in base al fatto che un utente sia membro di un tenant e che l'utente abbia il ruolo corretto per accedere all'API.



Questo modello comporta il rischio di accesso tra tenant diversi, poiché i ruoli e le autorizzazioni di più tenant `data.tenant2.user_roles` devono ora essere resi accessibili all'OPA per `data.tenant1.user_roles` prendere decisioni di autorizzazione. Per mantenere l'isolamento dei tenant e la riservatezza della mappatura dei ruoli, questi dati non devono risiedere all'interno dell'OPA. I dati RBAC devono risiedere in una fonte di dati esterna come un database. Inoltre, l'OPA non deve essere utilizzato per mappare ruoli predefiniti a permessi specifici, poiché ciò rende difficile per gli inquilini definire i propri ruoli e le proprie autorizzazioni. Inoltre, rende la logica di autorizzazione rigida e necessita di un aggiornamento costante. Per indicazioni su come incorporare in modo sicuro i dati RBAC nel processo decisionale dell'OPA, consulta la sezione Raccomandazioni per l'isolamento degli inquilini e la privacy dei dati più avanti in questa guida.

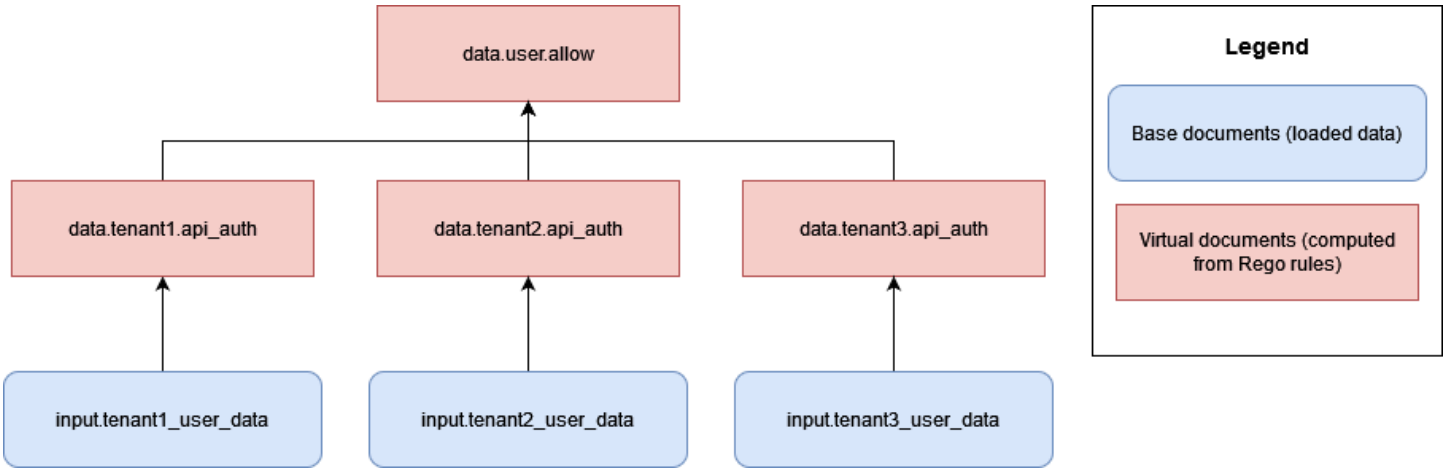
È possibile mantenere facilmente l'isolamento dei tenant in OPA evitando di archiviare alcun dato specifico del tenant come documento di base asincrono. Un documento di base asincrono è costituito da dati archiviati in memoria e che possono essere aggiornati periodicamente in OPA. Altri documenti di base, come l'input OPA, vengono trasmessi in modo sincrono e sono disponibili solo al momento della decisione. Ad esempio, fornire dati specifici del tenant come parte dell'input OPA a una query

non costituirebbe una violazione dell'isolamento del tenant, poiché tali dati sono disponibili solo in modo sincrono durante il processo decisionale.

Inserimento degli inquilini

La struttura dei documenti OPA deve consentire un semplice onboarding degli inquilini senza introdurre requisiti ingombranti. È possibile organizzare i documenti virtuali nella gerarchia dei modelli di documenti OPA con pacchetti e questi pacchetti possono contenere molte regole. Quando pianificate un modello di documento OPA per un'applicazione multi-tenant, stabilite innanzitutto quali dati sono necessari all'OPA per prendere una decisione. È possibile fornire dati come input, precaricarli in OPA o fornirli da fonti di dati esterne al momento della decisione o periodicamente. Per ulteriori informazioni sull'utilizzo di dati esterni con OPA, vedere la sezione Recupero di dati esterni per un PDP in OPA più avanti in questa guida.

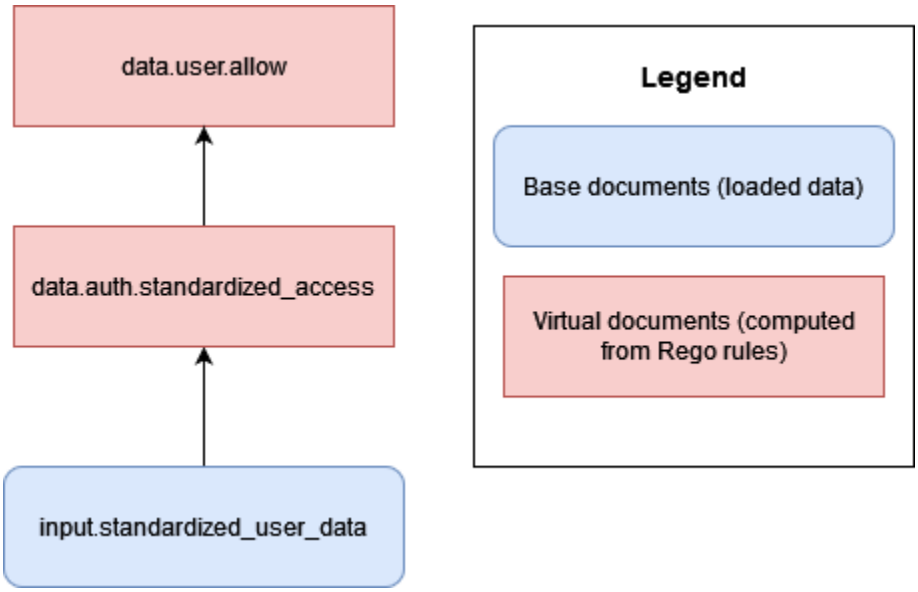
Dopo aver determinato i dati necessari per prendere una decisione in OPA, considerate come implementare le regole OPA organizzate come pacchetti, per prendere decisioni con quei dati. Ad esempio, in un modello SaaS in silos in cui ogni tenant potrebbe avere requisiti unici per il modo in cui vengono prese le decisioni di autorizzazione, è possibile implementare pacchetti di regole OPA specifici del tenant.



Lo svantaggio di questo approccio è che è necessario aggiungere un nuovo set di regole OPA, specifiche per ogni tenant, per ogni tenant aggiunto all'applicazione SaaS. Si tratta di un'operazione complessa e difficile da scalare, ma potrebbe essere inevitabile a seconda delle esigenze dei vostri inquilini.

In alternativa, in un modello SaaS condiviso, se tutti i tenant prendono decisioni di autorizzazione sulla base delle stesse regole e utilizzano la stessa struttura di dati, è possibile utilizzare pacchetti

OPA standard con regole generalmente applicabili per semplificare l'onboarding dei tenant e scalare l'implementazione OPA.



Ove possibile, si consiglia di utilizzare regole e pacchetti OPA generalizzati (o documenti virtuali) per prendere decisioni basate su dati standardizzati forniti da ciascun tenant. Questo approccio rende l'OPA facilmente scalabile, poiché si modificano solo i dati forniti all'OPA per ogni tenant, non il modo in cui OPA fornisce le proprie decisioni tramite le proprie regole. È necessario introdurre un rules-per-tenant modello solo quando i singoli inquilini richiedono decisioni uniche o devono fornire all'OPA dati diversi rispetto agli altri inquilini.

DevOps, monitoraggio, registrazione e recupero dei dati per un PDP

In questo paradigma di autorizzazione proposto, le politiche sono centralizzate nel servizio di autorizzazione. Questa centralizzazione è intenzionale perché uno degli obiettivi dei modelli di progettazione discussi in questa guida è quello di ottenere il disaccoppiamento delle politiche o la rimozione della logica di autorizzazione dagli altri componenti dell'applicazione. Sia Amazon Verified Permissions che Open Policy Agent (OPA) forniscono meccanismi per aggiornare le politiche quando sono necessarie modifiche alla logica di autorizzazione.

Nel caso delle autorizzazioni verificate, i meccanismi per l'aggiornamento programmatico delle politiche sono offerti dall' AWS SDK (consulta la Amazon Verified Permissions API Reference Guide). Utilizzando l'SDK, puoi implementare nuove politiche su richiesta. Inoltre, poiché Verified Permissions è un servizio gestito, non è necessario gestire, configurare o mantenere piani di controllo o agenti per eseguire gli aggiornamenti. Tuttavia, ti consigliamo di utilizzare una pipeline di integrazione e distribuzione continua (CI/CD) per amministrare la distribuzione degli archivi di policy per le autorizzazioni verificate e gli aggiornamenti delle policy utilizzando l'SDK. AWS

Le autorizzazioni verificate forniscono un facile accesso alle funzionalità di osservabilità. Può essere configurato per registrare tutti i tentativi di accesso ai gruppi di CloudWatch log di Amazon AWS CloudTrail, ai bucket Amazon Simple Storage Service (Amazon S3) o ai flussi di distribuzione di Amazon Data Firehose per consentire una risposta rapida agli incidenti di sicurezza e alle richieste di controllo. Inoltre, puoi monitorare lo stato del servizio Autorizzazioni verificate tramite. AWS Health Dashboard Poiché Verified Permissions è un servizio gestito, il suo stato è mantenuto da AWS, ed è possibile configurare le funzionalità di osservabilità utilizzando altri AWS servizi gestiti.

Nel caso dell'OPA, le API REST offrono modi per aggiornare le politiche in modo programmatico. È possibile configurare le API per estrarre nuove versioni dei pacchetti di policy da posizioni consolidate o per inviare policy su richiesta. Inoltre, OPA offre un servizio di discovery di base in cui i nuovi agenti possono essere configurati dinamicamente e gestiti centralmente da un piano di controllo che distribuisce i Discovery Bundle. (Il piano di controllo per OPA deve essere impostato e configurato dall'operatore OPA.) Si consiglia di creare una solida pipeline CI/CD per il controllo delle versioni, la verifica e l'aggiornamento delle politiche, indipendentemente dal fatto che il motore delle politiche sia Verified Permissions, OPA o un'altra soluzione.

Per OPA, il piano di controllo offre anche opzioni per il monitoraggio e il controllo. È possibile esportare i log che contengono le decisioni di autorizzazione dell'OPA su server HTTP remoti per l'aggregazione dei log. Questi registri decisionali sono preziosi per scopi di controllo.

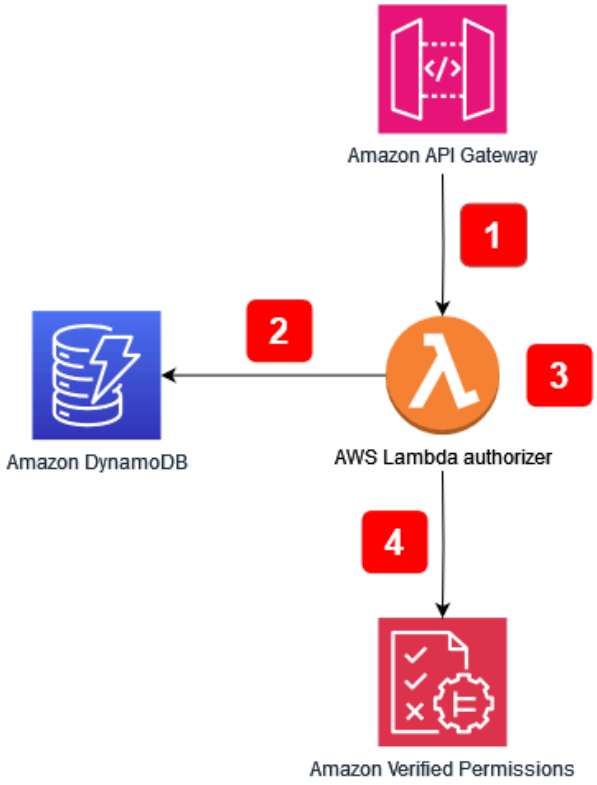
Se state pensando di adottare un modello di autorizzazione in cui le decisioni sul controllo degli accessi siano disaccoppiate dall'applicazione, assicuratevi che il servizio di autorizzazione disponga di funzionalità efficaci di monitoraggio, registrazione e gestione CI/CD per l'onboarding di nuovi PDP o l'aggiornamento delle politiche.

Argomenti

Recupero di dati esterni per un PDP in Amazon Verified Permissions

Amazon Verified Permissions non supporta il recupero di dati esterni per un PDP, ma può archiviare i dati forniti dall'utente come parte del suo schema. Come in OPA, se tutti i dati per una decisione di autorizzazione possono essere forniti come parte di una richiesta di autorizzazione o come parte di un token Web JSON (JWT) passato come parte della richiesta, non è richiesta alcuna configurazione aggiuntiva. Tuttavia, è possibile fornire dati aggiuntivi da fonti esterne a Verified Permissions tramite la richiesta di autorizzazione come parte del servizio di autorizzazione di un'applicazione che chiama Verified Permissions. Ad esempio, il servizio di autorizzazione di un'applicazione può interrogare una fonte esterna come DynamoDB o Amazon RDS per i dati e questi servizi possono quindi includere i dati forniti esternamente come parte di una richiesta di autorizzazione.

Il diagramma seguente mostra un esempio di come è possibile recuperare e incorporare dati aggiuntivi in una richiesta di autorizzazione Verified Permissions. Potrebbe essere necessario utilizzare questo metodo per recuperare dati come le mappature dei ruoli RBAC, per recuperare attributi aggiuntivi pertinenti alle risorse o ai principali o nei casi in cui i dati risiedono in parti diverse di un'applicazione e non possono essere forniti tramite un token di identity provider (IdP).



Flusso dell'applicazione:

1. L'applicazione riceve una chiamata API ad Amazon API Gateway e la inoltra all' AWS Lambda autorizzatore.
2. L'autorizzatore Lambda chiama Amazon DynamoDB per recuperare dati aggiuntivi sul principale che ha effettuato la richiesta.
3. L'autorizzatore Lambda incorpora i dati aggiuntivi nella richiesta di autorizzazione effettuata a Verified Permissions.
4. L'autorizzatore Lambda invia una richiesta di autorizzazione a Verified Permissions e riceve una decisione di autorizzazione.

Il diagramma include una funzionalità di Amazon API Gateway chiamata autorizzatore Lambda. Sebbene questa funzionalità possa non essere disponibile per le API fornite da altri servizi o applicazioni, puoi replicare il modello generale di utilizzo di un autorizzatore per recuperare dati aggiuntivi da incorporare in una richiesta di autorizzazione Verified Permissions in una moltitudine di casi d'uso.

Recupero di dati esterni per un PDP in OPA

Per OPA, se tutti i dati necessari per una decisione di autorizzazione possono essere forniti come input o come parte di un JSON Web Token (JWT) passato come componente della query, non è richiesta alcuna configurazione aggiuntiva. (È relativamente semplice passare i dati contestuali JWT e SaaS a OPA come parte dell'input delle query.) OPA può accettare input JSON arbitrari in quello che viene chiamato approccio di input in caso di sovraccarico. Se un PDP richiede dati oltre a quelli che possono essere inclusi come input o un JWT, OPA offre diverse opzioni per recuperare questi dati. Queste includono il raggruppamento, l'invio di dati (replica) e il recupero dinamico dei dati.

Raggruppamento OPA

La funzionalità di raggruppamento OPA supporta il seguente processo per il recupero dei dati esterni:

1. Il Policy Enforcement Point (PEP) richiede una decisione di autorizzazione.
2. OPA scarica nuovi pacchetti di policy, inclusi dati esterni.
3. Il servizio di raggruppamento replica i dati dalle fonti di dati.

Quando si utilizza la funzionalità di raggruppamento, OPA scarica periodicamente policy e pacchetti di dati da un servizio di bundle centralizzato. (OPA non fornisce l'implementazione e la configurazione di un servizio bundle.) Tutte le politiche e i dati esterni estratti dal servizio bundle vengono archiviati in memoria. Questa opzione non funziona se la dimensione dei dati esterni è troppo grande per essere archiviata in memoria o se i dati vengono modificati troppo frequentemente.

Per ulteriori informazioni sulla funzionalità di raggruppamento, consulta la documentazione OPA.

Replica OPA (invio di dati)

L'approccio di replica OPA supporta il seguente processo per il recupero dei dati esterni:

1. Il PEP richiede una decisione di autorizzazione.
2. Il replicatore di dati invia i dati all'OPA.
3. Il replicatore di dati replica i dati dalle fonti di dati.

In questa alternativa all'approccio basato sul raggruppamento, i dati vengono trasferiti all'OPA anziché essere raccolti periodicamente dall'OPA. (OPA non fornisce l'implementazione e la

configurazione di un replicatore). L'approccio push presenta gli stessi limiti di dimensione dei dati dell'approccio raggruppato, poiché OPA archivia tutti i dati in memoria. Il vantaggio principale dell'opzione push è che è possibile aggiornare i dati in OPA con delta anziché sostituire tutti i dati esterni ogni volta. Ciò rende l'opzione push più appropriata per i set di dati che cambiano frequentemente.

Per ulteriori informazioni sull'opzione di replica, consultate la documentazione OPA.

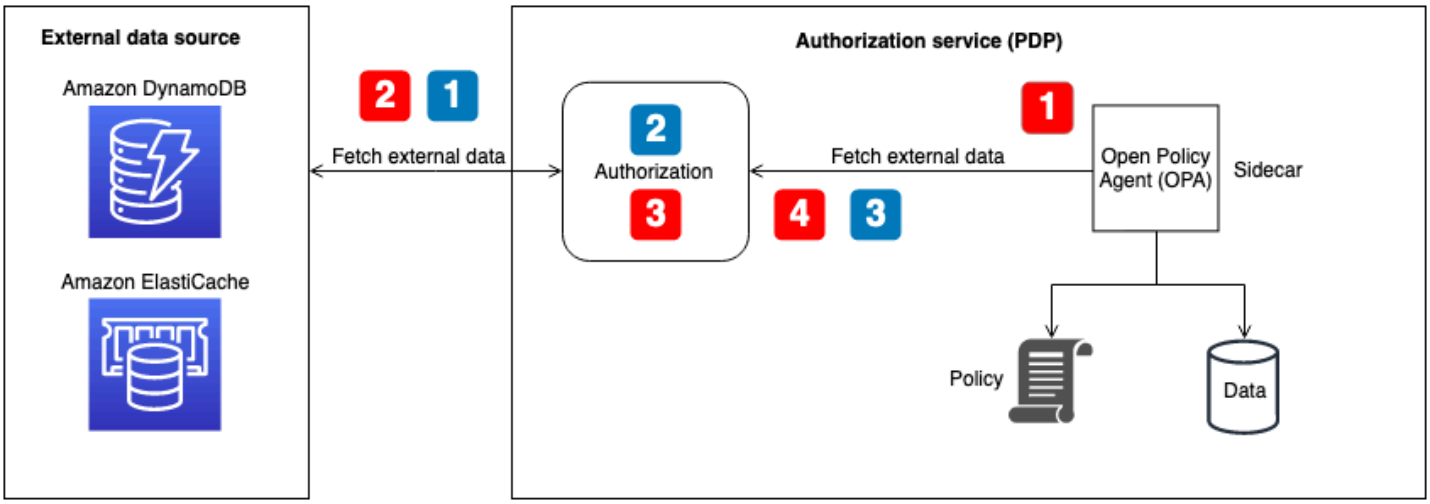
Recupero dinamico dei dati OPA

Se i dati esterni da recuperare sono troppo grandi per essere memorizzati nella memoria dell'OPA, i dati possono essere estratti dinamicamente da una fonte esterna durante la valutazione di una decisione di autorizzazione. Quando si utilizza questo approccio, i dati sono sempre aggiornati. Questo approccio presenta due inconvenienti: latenza di rete e accessibilità. Attualmente, OPA può recuperare i dati in fase di esecuzione solo tramite una richiesta HTTP. Se le chiamate che vanno a un'origine dati esterna non possono restituire dati come risposta HTTP, richiedono un'API personalizzata o qualche altro meccanismo per fornire questi dati a OPA. Poiché OPA può recuperare i dati solo tramite richieste HTTP e la velocità di recupero dei dati è fondamentale, ti consigliamo di utilizzare un servizio come Servizio AWS Amazon DynamoDB per conservare dati esterni quando possibile.

Per ulteriori informazioni sull'approccio pull, consulta la documentazione OPA.

Utilizzo di un servizio di autorizzazione per l'implementazione con OPA

Quando si recuperano dati esterni utilizzando il raggruppamento, la replica o un approccio di pull dinamico, si consiglia che il servizio di autorizzazione faciliti questa interazione. Questo perché il servizio di autorizzazione può recuperare dati esterni e trasformarli in JSON per consentire a OPA di prendere decisioni di autorizzazione. Il diagramma seguente mostra come un servizio di autorizzazione può funzionare con questi tre approcci esterni per il recupero dei dati.



Recupero di dati esterni per il flusso OPA: recupero unificato o dinamico dei dati al momento della decisione (illustrato con callout numerati in rosso nel diagramma):

1. L'OPA chiama l'endpoint dell'API locale per il servizio di autorizzazione, che è configurato come endpoint in bundle o come endpoint per il recupero dinamico dei dati durante le decisioni di autorizzazione.
2. Il servizio di autorizzazione interroga o chiama l'origine dati esterna per recuperare dati esterni. (Per un endpoint integrato, questi dati devono contenere anche le politiche e le regole OPA. Gli aggiornamenti in bundle sostituiscono tutto, sia i dati che le policy, nella cache di OPA.)
3. Il servizio di autorizzazione esegue qualsiasi trasformazione necessaria sui dati restituiti per trasformarli nell'input JSON previsto.
4. I dati vengono restituiti all'OPA. Viene memorizzato nella cache per la configurazione del pacchetto e utilizzato immediatamente per decisioni di autorizzazione dinamiche.

Recupero di dati esterni per OPA flow — replicator (illustrato con callout numerati in blu nel diagramma):

1. Il replicatore (parte del servizio di autorizzazione) richiama l'origine dati esterna e recupera tutti i dati da aggiornare in OPA. Ciò può includere politiche, regole e dati esterni. Questa chiamata può avere una cadenza prestabilita oppure può avvenire in risposta agli aggiornamenti dei dati nella fonte esterna.
2. Il servizio di autorizzazione esegue tutte le trasformazioni necessarie sui dati restituiti per trasformarli nell'input JSON previsto.

3. Il servizio di autorizzazione chiama OPA e memorizza i dati nella cache. Il servizio di autorizzazione può aggiornare selettivamente dati, politiche e regole.

Raccomandazioni per l'isolamento degli inquilini e la riservatezza dei dati

La sezione precedente ha fornito diversi approcci per l'utilizzo di dati esterni con OPA e Amazon Verified Permissions per facilitare le decisioni di autorizzazione. Ove possibile, consigliamo di utilizzare l'approccio di input in caso di sovraccarico per passare i dati contestuali SaaS a OPA per prendere decisioni di autorizzazione invece di archiviare i dati nella memoria di OPA. Questo caso d'uso non si applica a AWS Cloud Map, perché non supporta l'archiviazione di dati esterni nel servizio.

Nei modelli ibridi di controllo degli accessi basato sui ruoli (RBAC) o RBAC e nei modelli ibridi di controllo degli accessi basato sugli attributi (ABAC), i dati forniti esclusivamente da una richiesta o da una query di autorizzazione potrebbero essere insufficienti, poiché è necessario fare riferimento a ruoli e autorizzazioni per prendere decisioni di autorizzazione. Per mantenere l'isolamento degli inquilini e la riservatezza della mappatura dei ruoli, questi dati non devono risiedere all'interno dell'OPA. I dati RBAC devono risiedere in una fonte di dati esterna come un database o devono essere trasmessi come parte dei claim in un JWT da un IdP. In Verified Permissions, i dati RBAC possono essere gestiti come parte delle politiche e dello schema nel modello di archivio delle politiche per tenant, poiché ogni tenant ha il proprio archivio delle politiche separato logicamente. Tuttavia, nel modello di policy store multi-tenant condiviso, i dati di mappatura dei ruoli non devono risiedere nelle autorizzazioni verificate per mantenere l'isolamento dei tenant.

Inoltre, OPA e Verified Permissions non devono essere utilizzati per mappare ruoli predefiniti a permessi specifici, poiché ciò rende difficile per gli inquilini definire i propri ruoli e le proprie autorizzazioni. Inoltre, rende la logica di autorizzazione rigida e necessita di un aggiornamento costante. L'eccezione a questa linea guida è il modello di archivio delle politiche per tenant in Verified Permissions, poiché questo modello consente a ciascun tenant di disporre di proprie politiche che possono essere valutate indipendentemente su base individuale.

Autorizzazioni verificate da Amazon

L'unico posto in cui Verified Permissions può archiviare dati RBAC potenzialmente privati è nello schema. Ciò è accettabile nel modello di archivio delle politiche per tenant, poiché ogni tenant dispone di un proprio archivio di politiche separato logicamente. Tuttavia, potrebbe compromettere

l'isolamento dei tenant nell'unico modello di archivio delle politiche multi-tenant condiviso. Nei casi in cui questi dati siano necessari per prendere una decisione di autorizzazione, devono essere recuperati da una fonte esterna come DynamoDB o Amazon RDS e incorporati nella richiesta di autorizzazione Verified Permissions.

OPA

Gli approcci sicuri con OPA per il mantenimento della privacy e l'isolamento dei tenant dei dati RBAC includono l'utilizzo del recupero o della replica dinamica dei dati per ottenere dati esterni. Questo perché è possibile utilizzare il servizio di autorizzazione illustrato nel diagramma precedente per fornire solo dati esterni specifici del tenant o dell'utente per prendere una decisione di autorizzazione. Ad esempio, è possibile utilizzare un replicatore per fornire dati RBAC o una matrice di autorizzazioni alla cache OPA quando un utente accede e fare in modo che i dati vengano referenziati in base a un utente fornito nei dati di input. È possibile utilizzare un approccio simile con i dati estratti dinamicamente per recuperare solo i dati pertinenti per prendere decisioni di autorizzazione. Inoltre, nell'approccio al recupero dinamico dei dati, questi dati non devono essere memorizzati nella cache in OPA. L'approccio di raggruppamento non è efficace quanto l'approccio di recupero dinamico nel mantenere l'isolamento dei tenant, perché aggiorna tutto ciò che si trova nella cache OPA e non è in grado di elaborare aggiornamenti precisi. Il modello di raggruppamento è ancora un buon approccio per l'aggiornamento delle politiche OPA e dei dati non RBAC.

Best practice

Questa sezione elenca alcuni dei punti salienti di questa guida. Per discussioni dettagliate su ciascun punto, segui i collegamenti alle sezioni corrispondenti.

Seleziona un modello di controllo degli accessi adatto alla tua applicazione

Questa guida illustra diversi modelli di controllo degli accessi. A seconda dell'applicazione e dei requisiti aziendali, è necessario selezionare un modello adatto alle proprie esigenze. Considerate come potete utilizzare questi modelli per soddisfare le vostre esigenze di controllo degli accessi e in che modo le vostre esigenze di controllo degli accessi potrebbero evolversi, richiedendo modifiche all'approccio scelto.

Implementa un PDP

Il punto decisionale delle politiche (PDP) può essere caratterizzato come un motore di politiche o regole. Questo componente è responsabile dell'applicazione di politiche o regole e della decisione se consentire un determinato accesso. Un PDP consente di scaricare la logica di autorizzazione nel codice dell'applicazione su un sistema separato. Questo può semplificare il codice dell'applicazione. Fornisce inoltre un'interfaccia easy-to-use idempotente per prendere decisioni di autorizzazione per API, microservizi, livelli di Backend for Frontend (BFF) o qualsiasi altro componente dell'applicazione. Un PDP può essere utilizzato per applicare i requisiti di locazione in modo coerente in un'applicazione.

Implementa i PEP per ogni API della tua applicazione

L'implementazione di un punto di applicazione delle politiche (PEP) richiede la determinazione del punto in cui deve avvenire l'applicazione del controllo degli accessi. Come primo passo, individuate i punti dell'applicazione in cui potete incorporare i PEP. Considerate questo principio quando decidete dove aggiungere i PEP:

Se un'applicazione espone un'API, dovrebbe esserci l'autorizzazione e il controllo degli accessi su quell'API.

Prendi in considerazione l'utilizzo di Amazon Verified Permissions o OPA come motore di policy per il tuo PDP.

Amazon Verified Permissions offre vantaggi rispetto ai motori di policy personalizzati. È un servizio scalabile e dettagliato di gestione e autorizzazione delle autorizzazioni per le applicazioni che crei. Supporta la scrittura di politiche nel linguaggio dichiarativo open source di alto livello Cedar. Di conseguenza, l'implementazione di un motore di policy utilizzando Verified Permissions richiede meno sforzi di sviluppo rispetto all'implementazione di una soluzione propria. Inoltre, Verified Permissions è completamente gestito, quindi non è necessario gestire l'infrastruttura sottostante.

L'Open Policy Agent (OPA) presenta vantaggi rispetto ai motori di policy personalizzati. OPA e la sua valutazione delle politiche con Rego forniscono un motore di policy flessibile e predefinito che supporta la scrittura di politiche in un linguaggio dichiarativo di alto livello. Ciò riduce notevolmente il livello di impegno richiesto per l'implementazione di un motore di policy rispetto alla creazione di una soluzione personalizzata. Inoltre, l'OPA sta rapidamente diventando uno standard di autorizzazione ben supportato.

Implementa un piano di controllo per OPA per il monitoraggio DevOps e la registrazione

Poiché OPA non fornisce un mezzo per aggiornare e tenere traccia delle modifiche alla logica di autorizzazione tramite il controllo del codice sorgente, si consiglia di implementare un piano di controllo per eseguire queste funzioni. Ciò consentirà di distribuire più facilmente gli aggiornamenti agli agenti OPA, in particolare se OPA opera in un sistema distribuito, il che ridurrà l'onere amministrativo derivante dall'utilizzo di OPA. Inoltre, è possibile utilizzare un piano di controllo per raccogliere i registri per l'aggregazione e monitorare lo stato degli agenti OPA.

Configura le funzionalità di registrazione e osservabilità in Autorizzazioni verificate

Le autorizzazioni verificate forniscono un facile accesso alle funzionalità di osservabilità. Puoi configurare il servizio per registrare tutti i tentativi di accesso ai gruppi di CloudWatch log di Amazon AWS CloudTrail, ai bucket S3 o ai flussi di distribuzione di Amazon Data Firehose per consentire una risposta rapida agli incidenti di sicurezza e alle richieste di controllo. Inoltre, puoi monitorare lo stato del servizio tramite. AWS Health Dashboard Poiché Verified Permissions è un servizio gestito, la sua

integrità viene gestita da AWS, ed è possibile configurarne le funzionalità di osservabilità utilizzando altri servizi AWS gestiti.

Utilizza una pipeline CI/CD per fornire e aggiornare gli archivi di policy e le policy in Verified Permissions

Verified Permissions è un servizio gestito, quindi non è necessario gestire, configurare o mantenere piani di controllo o agenti per eseguire gli aggiornamenti. Tuttavia, ti consigliamo comunque di utilizzare una pipeline di integrazione e distribuzione continua (CI/CD) per amministrare la distribuzione degli archivi di policy e degli aggiornamenti delle policy per le autorizzazioni verificate utilizzando l'SDK. AWS Questo sforzo può eliminare il lavoro manuale e ridurre la probabilità di errori dell'operatore quando si apportano modifiche alle risorse di Autorizzazioni verificate.

Determina se sono necessari dati esterni per le decisioni di autorizzazione e seleziona un modello adatto

Se un PDP può prendere decisioni di autorizzazione basate esclusivamente sui dati contenuti in un JSON Web Token (JWT), di solito non è necessario importare dati esterni per facilitare il processo decisionale. Se utilizzi Autorizzazioni verificate o OPA come PDP, può anche accettare input aggiuntivi che vengono trasmessi come parte della richiesta, anche se questi dati non sono inclusi in un JWT. Per le autorizzazioni verificate, puoi utilizzare un parametro di contesto per i dati aggiuntivi. Per OPA, puoi utilizzare i dati JSON come input di sovraccarico. Se si utilizza un JWT, i metodi di input contestuali o di sovraccarico sono generalmente molto più semplici rispetto alla conservazione dei dati esterni in un'altra fonte. Se sono necessari dati esterni più complessi per prendere decisioni di autorizzazione, OPA offre diversi modelli per il recupero di dati esterni e Verified Permissions può integrare i dati nelle sue richieste di autorizzazione facendo riferimento a fonti esterne con un servizio di autorizzazione.

Domande frequenti

Questa sezione fornisce le risposte alle domande più frequenti sull'implementazione del controllo e dell'autorizzazione degli accessi alle API nelle applicazioni SaaS multi-tenant.

D: Qual è la differenza tra autorizzazione e autenticazione?

R. L'autenticazione è il processo di verifica dell'identità di un utente. L'autorizzazione concede agli utenti le autorizzazioni per accedere a una risorsa specifica.

D: Qual è la differenza tra l'autorizzazione e l'isolamento dei tenant in un'applicazione SaaS?

R. L'isolamento dei tenant si riferisce a meccanismi espliciti utilizzati in un sistema SaaS per garantire che le risorse di ciascun tenant, anche quando operano su un'infrastruttura condivisa, siano isolate. L'autorizzazione multi-tenant si riferisce all'autorizzazione delle azioni in entrata e alla prevenzione che vengano implementate sul tenant sbagliato. Un utente ipotetico potrebbe essere autenticato e autorizzato, ma potrebbe comunque essere in grado di accedere alle risorse di un altro tenant. Nulla di ciò che riguarda l'autenticazione e l'autorizzazione blocca necessariamente questo accesso, ma l'isolamento dei tenant è necessario per raggiungere questo obiettivo. Per ulteriori informazioni su questi due concetti, consulta la discussione sull'isolamento dei tenant nel white paper AWS SaaS Architecture Fundamentals.

D: Perché devo prendere in considerazione l'isolamento dei tenant per la mia applicazione SaaS?

R. Le applicazioni SaaS hanno più tenant. Un tenant può essere un'organizzazione cliente o qualsiasi entità esterna che utilizza quell'applicazione SaaS. A seconda di come è progettata l'applicazione, ciò significa che i tenant potrebbero accedere ad API, database o altre risorse condivise. È importante mantenere l'isolamento dei tenant, vale a dire costrutti che controllino rigorosamente l'accesso alle risorse e blocchino qualsiasi tentativo di accedere alle risorse di un altro tenant, per impedire agli utenti di un tenant di accedere alle informazioni private di un altro tenant. Le applicazioni SaaS sono spesso progettate per garantire che l'isolamento dei tenant sia mantenuto in tutta l'applicazione e che i tenant possano accedere solo alle proprie risorse.

D: Perché ho bisogno di un modello di controllo degli accessi?

R. I modelli di controllo degli accessi vengono utilizzati per creare un metodo coerente per determinare come concedere l'accesso alle risorse di un'applicazione. Ciò può essere fatto assegnando ruoli agli utenti strettamente allineati alla logica aziendale, oppure può basarsi su altri attributi contestuali come l'ora del giorno o se un utente soddisfa una condizione predefinita. I modelli

di controllo degli accessi costituiscono la logica di base utilizzata dall'applicazione per prendere decisioni di autorizzazione per determinare le autorizzazioni degli utenti.

D: Il controllo degli accessi alle API è necessario per la mia applicazione?

R. Sì. Le API devono sempre verificare che il chiamante disponga dell'accesso appropriato. Il controllo pervasivo degli accessi alle API garantisce inoltre che l'accesso venga concesso solo in base ai tenant, in modo da poter mantenere l'isolamento appropriato.

D: Perché si consiglia l'autorizzazione ai motori di policy o ai PDP?

R. Un punto di decisione sulle politiche (PDP) consente di scaricare la logica di autorizzazione nel codice dell'applicazione su un sistema separato. Questo può semplificare il codice dell'applicazione. Fornisce inoltre un'interfaccia easy-to-use idempotente per prendere decisioni di autorizzazione per API, microservizi, livelli di Backend for Frontend (BFF) o qualsiasi altro componente dell'applicazione.

D: Cos'è un PEP?

R. Un Policy Enforcement Point (PEP) è responsabile della ricezione delle richieste di autorizzazione inviate al PDP per la valutazione. Un PEP può trovarsi in qualsiasi punto di un'applicazione in cui i dati e le risorse devono essere protetti o in cui viene applicata la logica di autorizzazione. I PEP sono relativamente semplici rispetto ai PDP. Un PEP è responsabile solo della richiesta e della valutazione di una decisione di autorizzazione e non richiede alcuna logica di autorizzazione per essere incorporata in essa.

D: Come devo scegliere tra Amazon Verified Permissions e OPA?

R. Per scegliere tra Autorizzazioni verificate e Open Policy Agent (OPA), tieni sempre a mente il tuo caso d'uso e i tuoi requisiti unici. Verified Permissions offre un modo completamente gestito per definire autorizzazioni dettagliate, controllare le autorizzazioni tra le applicazioni e centralizzare il sistema di amministrazione delle policy per le applicazioni, soddisfacendo al contempo i requisiti di latenza delle applicazioni con un'elaborazione in millisecondi. OPA è un motore di policy open source generico che può anche aiutarti a unificare le policy in tutto lo stack di applicazioni. Per eseguire OPA è necessario ospitarlo nel proprio AWS ambiente, in genere con un contenitore o delle funzioni. AWS Lambda

Verified Permissions utilizza il linguaggio di policy open source Cedar, mentre OPA utilizza Rego. Pertanto, la familiarità con una di queste lingue potrebbe indurti a scegliere quella soluzione. Tuttavia, ti consigliamo di leggere informazioni su entrambe le lingue e poi di risolvere il problema che stai cercando di risolvere per trovare la soluzione migliore per il tuo caso d'uso.

D: Esistono alternative open source alle autorizzazioni verificate e all'OPA?

R. Esistono alcuni sistemi open source simili a Verified Permissions e Open Policy Agent (OPA), come il Common Expression Language (CEL). Questa guida si concentra sia sulle autorizzazioni verificate, in quanto servizio scalabile di gestione delle autorizzazioni e di autorizzazione dettagliato, sia su OPA, che è ampiamente adottato, documentato e adattabile a molti tipi diversi di applicazioni e requisiti di autorizzazione.

D: Devo scrivere un servizio di autorizzazione per utilizzare OPA o posso interagire direttamente con OPA?

R. È possibile interagire direttamente con OPA. Nel contesto di questa guida, un servizio di autorizzazione si riferisce a un servizio che traduce le richieste di decisione di autorizzazione in interrogazioni OPA e viceversa. Se l'applicazione è in grado di interrogare e accettare direttamente le risposte OPA, non è necessario introdurre questa complessità aggiuntiva.

D: Come posso monitorare i miei agenti OPA per scopi di uptime e controllo?

R. OPA fornisce la registrazione e il monitoraggio di base dell'uptime, sebbene la sua configurazione predefinita sia probabilmente insufficiente per le implementazioni aziendali. Per ulteriori informazioni, consultate la sezione relativa al DevOpsmonitoraggio e alla registrazione precedente di questa guida.

D: Come posso monitorare le autorizzazioni verificate per scopi di operatività e controllo?

R. Verified Permissions è un servizio AWS gestito e la disponibilità può essere monitorata tramite. AWS Health Dashboard Inoltre, Verified Permissions è in grado di accedere ad AWS CloudTrail Amazon CloudWatch Logs, Amazon S3 e Amazon Data Firehose.

D: Quali sistemi operativi e servizi AWS posso usare per eseguire OPA?

R. È possibile eseguire OPA su macOS, Windows e Linux. Gli agenti OPA possono essere configurati su agenti Amazon Elastic Compute Cloud (Amazon EC2) e su servizi di containerizzazione come Amazon Elastic Container Service (Amazon ECS) e Amazon Elastic Kubernetes Service (Amazon EKS).

D: Quali sistemi operativi e servizi posso usare per eseguire le autorizzazioni verificate? AWS

R. Verified Permissions è un servizio AWS gestito ed è gestito da. AWS Non è necessaria alcuna configurazione, installazione o hosting aggiuntivi per utilizzare le autorizzazioni verificate, ad eccezione della possibilità di effettuare richieste di autorizzazione al servizio.

D: Posso eseguire OPA su? AWS Lambda

R. È possibile eseguire OPA su Lambda come libreria Go. Per informazioni su come eseguire questa operazione per un autorizzatore Lambda API Gateway, consulta il post del AWS blog Creazione di un autorizzatoreLambda personalizzato utilizzando Open Policy Agent.

D: Come devo decidere tra un approccio PDP distribuito e un PDP centralizzato?

R. Dipende dall'applicazione. Molto probabilmente verrà determinato in base alla differenza di latenza tra un modello PDP distribuito e uno centralizzato. Ti consigliamo di creare un proof of concept e di testare le prestazioni dell'applicazione per verificare la soluzione.

D: Posso usare OPA per casi d'uso diversi dalle API?

R. Sì. La documentazione OPA fornisce esempi per Kubernetes, Envoy, Docker, Kafka, SSH e sudo e Terraform. Inoltre, OPA può restituire risposte JSON arbitrarie alle query utilizzando le regole parziali Rego. A seconda della query, OPA può essere utilizzato per rispondere a molte domande con risposte JSON.

D: Posso utilizzare le autorizzazioni verificate per casi d'uso diversi dalle API?

R. Sì. Verified Permissions può fornire una `DENY` risposta `ALLOW` OR a qualsiasi richiesta di autorizzazione ricevuta. Verified Permissions può fornire risposte di autorizzazione per qualsiasi applicazione o servizio che richieda decisioni di autorizzazione.

D: Posso creare policy in Verified Permissions utilizzando il linguaggio di policy IAM?

R. No. È necessario utilizzare il linguaggio delle politiche Cedar per creare politiche. Cedar è progettato per supportare la gestione delle autorizzazioni per le risorse delle applicazioni dei clienti, mentre il linguaggio di policy AWS Identity and Access Management (IAM) si è evoluto per supportare il controllo degli accessi alle risorse. AWS

Passaggi successivi

La complessità del processo di autorizzazione e del controllo degli accessi alle API per le applicazioni SaaS multi-tenant può essere superata adottando un approccio standardizzato e indipendente dal linguaggio per prendere decisioni di autorizzazione. Questi approcci incorporano punti decisionali (PDP) e punti di applicazione delle politiche (PDP) che impongono l'accesso in modo flessibile e pervasivo. È possibile incorporare diversi approcci al controllo degli accessi, come il controllo degli accessi basato sui ruoli (RBAC), il controllo degli accessi basato su attributi (ABAC) o una combinazione dei due, in una strategia di controllo degli accessi coesa. La rimozione della logica di autorizzazione da un'applicazione elimina il sovraccarico dovuto all'inclusione di soluzioni ad hoc nel codice dell'applicazione per affrontare il controllo degli accessi. L'implementazione e le best practice trattate in questa guida hanno lo scopo di informare e standardizzare un approccio all'implementazione dell'autorizzazione e del controllo degli accessi alle API nelle applicazioni SaaS multi-tenant. È possibile utilizzare questa guida come primo passo nella raccolta di informazioni e nella progettazione di un solido sistema di controllo e autorizzazione degli accessi per l'applicazione. Fasi successive:

- Esamina le tue esigenze di autorizzazione e isolamento dei tenant e seleziona un modello di controllo degli accessi per l'applicazione.
- Crea un proof of concept da testare utilizzando Amazon Verified Permissions o Open Policy Agent (OPA) oppure scrivendo il tuo motore di policy personalizzato.
- Identifica le API e le posizioni dell'applicazione in cui devono essere implementati i PEP.

Risorse

Riferimenti

- Documentazione sulle autorizzazioni verificate di Amazon (AWS documentazione)
- Come usare Amazon Verified Permissions per l'autorizzazione (post AWS sul blog)
- Implementa un provider di policy di autorizzazione personalizzato per le app ASP.NET Core utilizzando Amazon Verified Permissions (AWS post sul blog)
- Gestisci ruoli e autorizzazioni con PBAC utilizzando Amazon VerifiedAWS Permissions (post sul blog)
- Controllo degli accessi SaaS tramite Amazon Verified Permissions con un archivio di policy per tenant (post sul blog)AWS
- La documentazione ufficiale dell'OPA
- Perché le imprese devono abbracciare il progetto CNCF con la laurea più recente — Open Policy Agent (articolo di Forbes di Janakiram MSV, 8 febbraio 2021)
- Creazione di un autorizzatore Lambda personalizzato utilizzando Open Policy Agent (AWS post sul blog)
- Realizza le policy come codice con AWS Cloud Development Kit tramite Open Policy Agent (post AWS sul blog)
- Governance e AWS conformità del cloud utilizzando le policy come codice (post AWS sul blog)
- Utilizzo di Open Policy Agent su Amazon EKS (post AWS del blog)
- Conformità come codice per Amazon ECS utilizzando Open Policy Agent EventBridge, Amazon e AWS Lambda (post AWS sul blog)
- Contromisure basate su politiche per Kubernetes — Parte 1 (post sul blog)AWS
- Utilizzo degli autorizzatori Lambda API Gateway (documentazione)AWS

Strumenti

- The Cedar Playground (per testare Cedar in un browser)
- Repository Cedar Github
- Riferimento al linguaggio Cedar
- The Rego Playground (per testare Rego in un browser)

- Archivio OPA GitHub

Partner

- Partner per la gestione di identità e accessi
- Partner per la sicurezza delle applicazioni
- Partner per la governance del cloud
- Partner per la sicurezza e la conformità
- Partner per le operazioni di sicurezza e l'automazione
- Partner di ingegneria della sicurezza

Cronologia dei documenti

La tabella seguente descrive le modifiche significative apportate a questa guida. Per ricevere notifiche sugli aggiornamenti futuri, puoi abbonarti a un feed RSS.

Modifica	Descrizione	Data
Dettagli ed esempi aggiunti per Amazon Verified Permissions	Sono state aggiunte discussioni dettagliate, esempi e codice per l'utilizzo di Amazon Verified Permissions per implementare un PDP. Le nuove sezioni includono: • Implementazione di un PDP utilizzando Amazon Verified Permissions • Modelli di progettazione per Amazon Verified Permissions • Considerazioni sulla progettazione multi-tenant di Amazon Verified Permissions • Recupero di dati esterni per un PDP in Amazon Verified Permissions	28 maggio 2024
Informazioni chiarite	È stato chiarito il modello di progettazione PDP distribuito con PEP sulle API.	10 gennaio 2024
Sono state aggiunte brevi informazioni sul nuovo servizio AWS	Sono state aggiunte informazioni su Amazon Verified Permissions, che offre le	22 maggio 2023

	stesse funzionalità e gli stessi vantaggi di OPA.	
—	Pubblicazione iniziale	17 agosto 2021

AWS Glossario delle linee guida prescrittive

I seguenti sono termini comunemente usati nelle strategie, nelle guide e nei modelli forniti da AWS Prescriptive Guidance. Per suggerire voci, utilizza il link Fornisci feedback alla fine del glossario.

Numeri

7 R

Sette strategie di migrazione comuni per trasferire le applicazioni sul cloud. Queste strategie si basano sulle 5 R identificate da Gartner nel 2011 e sono le seguenti:

- Rifattorizzare/riprogettare: trasferisci un'applicazione e modifica la sua architettura sfruttando appieno le funzionalità native del cloud per migliorare l'agilità, le prestazioni e la scalabilità. Ciò comporta in genere la portabilità del sistema operativo e del database. Esempio: migra il tuo database Oracle locale all'edizione compatibile con Amazon Aurora PostgreSQL.
- Ridefinire la piattaforma (lift and reshape): trasferisci un'applicazione nel cloud e introduci un certo livello di ottimizzazione per sfruttare le funzionalità del cloud. Esempio: migra il tuo database Oracle locale ad Amazon Relational Database Service (Amazon RDS) per Oracle in. Cloud AWS
- Riacquistare (drop and shop): passa a un prodotto diverso, in genere effettuando la transizione da una licenza tradizionale a un modello SaaS. Esempio: migra il tuo sistema di gestione delle relazioni con i clienti (CRM) su Salesforce.com.
- Eseguire il rehosting (lift and shift): trasferisci un'applicazione sul cloud senza apportare modifiche per sfruttare le funzionalità del cloud. Esempio: migra il tuo database Oracle locale a Oracle su un'istanza EC2 in. Cloud AWS
- Trasferire (eseguire il rehosting a livello hypervisor): trasferisci l'infrastruttura sul cloud senza acquistare nuovo hardware, riscrivere le applicazioni o modificare le operazioni esistenti. Esegui la migrazione dei server da una piattaforma locale a un servizio cloud per la stessa piattaforma. Esempio: migra un'applicazione suMicrosoft Hyper-V. AWS
- Riesaminare (mantenere): mantieni le applicazioni nell'ambiente di origine. Queste potrebbero includere applicazioni che richiedono una rifattorizzazione significativa che desideri rimandare a un momento successivo e applicazioni legacy che desideri mantenere, perché non vi è alcuna giustificazione aziendale per effettuarne la migrazione.
- Ritirare: disattiva o rimuovi le applicazioni che non sono più necessarie nell'ambiente di origine.

A

ABAC

Vedi controllo degli accessi basato sugli attributi.

servizi astratti

Vedi servizi gestiti.

ACIDO

Vedi atomicità, consistenza, isolamento, durata.

migrazione attiva-attiva

Un metodo di migrazione del database in cui i database di origine e di destinazione vengono mantenuti sincronizzati (utilizzando uno strumento di replica bidirezionale o operazioni di doppia scrittura) ed entrambi i database gestiscono le transazioni provenienti dalle applicazioni di connessione durante la migrazione. Questo metodo supporta la migrazione in piccoli batch controllati anziché richiedere una conversione una tantum. È più flessibile ma richiede più lavoro rispetto alla migrazione attiva-passiva.

migrazione attiva-passiva

Un metodo di migrazione di database in cui i database di origine e di destinazione vengono mantenuti sincronizzati, ma solo il database di origine gestisce le transazioni provenienti dalle applicazioni di connessione mentre i dati vengono replicati nel database di destinazione. Il database di destinazione non accetta alcuna transazione durante la migrazione.

funzione aggregata

Una funzione SQL che opera su un gruppo di righe e calcola un singolo valore restituito per il gruppo. Esempi di funzioni aggregate includono `SUM` e. `MAX`

Intelligenza artificiale

Vedi intelligenza artificiale.

AIOps

Guarda le operazioni di intelligenza artificiale.

anonimizzazione

Il processo di eliminazione permanente delle informazioni personali in un set di dati. L'anonimizzazione può aiutare a proteggere la privacy personale. I dati anonimi non sono più considerati dati personali.

anti-modello

Una soluzione utilizzata frequentemente per un problema ricorrente in cui la soluzione è controproducente, inefficace o meno efficace di un'alternativa.

controllo delle applicazioni

Un approccio alla sicurezza che consente l'uso solo di applicazioni approvate per proteggere un sistema dal malware.

portfolio di applicazioni

Una raccolta di informazioni dettagliate su ogni applicazione utilizzata da un'organizzazione, compresi i costi di creazione e manutenzione dell'applicazione e il relativo valore aziendale. Queste informazioni sono fondamentali per il processo di scoperta e analisi del portfolio e aiutano a identificare e ad assegnare la priorità alle applicazioni da migrare, modernizzare e ottimizzare.

intelligenza artificiale (IA)

Il campo dell'informatica dedicato all'uso delle tecnologie informatiche per svolgere funzioni cognitive tipicamente associate agli esseri umani, come l'apprendimento, la risoluzione di problemi e il riconoscimento di schemi. Per ulteriori informazioni, consulta la sezione Che cos'è l'intelligenza artificiale?

operazioni di intelligenza artificiale (AIOps)

Il processo di utilizzo delle tecniche di machine learning per risolvere problemi operativi, ridurre gli incidenti operativi e l'intervento umano e aumentare la qualità del servizio. Per ulteriori informazioni su come viene utilizzato AIOps nella strategia di migrazione AWS , consulta la guida all'integrazione delle operazioni.

crittografia asimmetrica

Un algoritmo di crittografia che utilizza una coppia di chiavi, una chiave pubblica per la crittografia e una chiave privata per la decrittografia. Puoi condividere la chiave pubblica perché non viene utilizzata per la decrittografia, ma l'accesso alla chiave privata deve essere altamente limitato.

atomicità, consistenza, isolamento, durabilità (ACID)

Un insieme di proprietà del software che garantiscono la validità dei dati e l'affidabilità operativa di un database, anche in caso di errori, interruzioni di corrente o altri problemi.

Controllo degli accessi basato su attributi (ABAC)

La pratica di creare autorizzazioni dettagliate basate su attributi utente, come reparto, ruolo professionale e nome del team. Per ulteriori informazioni, consulta ABAC for AWS nella documentazione AWS Identity and Access Management (IAM).

fonte di dati autorevole

Una posizione in cui è archiviata la versione principale dei dati, considerata la fonte di informazioni più affidabile. È possibile copiare i dati dalla fonte di dati autorevole in altre posizioni allo scopo di elaborarli o modificarli, ad esempio anonimizzandoli, oscurandoli o pseudonimizzandoli.

Zona di disponibilità

Una posizione distinta all'interno di un edificio Regione AWS che è isolata dai guasti in altre zone di disponibilità e offre una connettività di rete economica e a bassa latenza verso altre zone di disponibilità nella stessa regione.

AWS Cloud Adoption Framework (CAF)AWS

Un framework di linee guida e best practice AWS per aiutare le organizzazioni a sviluppare un piano efficiente ed efficace per passare con successo al cloud. AWS CAF organizza le linee guida in sei aree di interesse chiamate prospettive: business, persone, governance, piattaforma, sicurezza e operazioni. Le prospettive relative ad azienda, persone e governance si concentrano sulle competenze e sui processi aziendali; le prospettive relative alla piattaforma, alla sicurezza e alle operazioni si concentrano sulle competenze e sui processi tecnici. Ad esempio, la prospettiva relativa alle persone si rivolge alle parti interessate che gestiscono le risorse umane (HR), le funzioni del personale e la gestione del personale. In questa prospettiva, AWS CAF fornisce linee guida per lo sviluppo delle persone, la formazione e le comunicazioni per aiutare a preparare l'organizzazione all'adozione del cloud di successo. Per ulteriori informazioni, consulta il sito web di AWS CAF e il white paper AWS CAF.

AWS Workload Qualification Framework (WQF)AWS

Uno strumento che valuta i carichi di lavoro di migrazione dei database, consiglia strategie di migrazione e fornisce stime del lavoro. AWS WQF è incluso in (). AWS Schema Conversion Tool AWS SCT Analizza gli schemi di database e gli oggetti di codice, il codice dell'applicazione, le dipendenze e le caratteristiche delle prestazioni e fornisce report di valutazione.

B

bot difettoso

Un bot che ha lo scopo di interrompere o causare danni a individui o organizzazioni.

BCP

Vedi la pianificazione della continuità operativa.

grafico comportamentale

Una vista unificata, interattiva dei comportamenti delle risorse e delle interazioni nel tempo. Puoi utilizzare un grafico comportamentale con Amazon Detective per esaminare tentativi di accesso non riusciti, chiamate API sospette e azioni simili. Per ulteriori informazioni, consulta Dati in un grafico comportamentale nella documentazione di Detective.

sistema big-endian

Un sistema che memorizza per primo il byte più importante. Vedi anche endianness.

Classificazione binaria

Un processo che prevede un risultato binario (una delle due classi possibili). Ad esempio, il modello di machine learning potrebbe dover prevedere problemi come "Questa e-mail è spam o non è spam?" o "Questo prodotto è un libro o un'auto?"

filtro Bloom

Una struttura di dati probabilistica ed efficiente in termini di memoria che viene utilizzata per verificare se un elemento fa parte di un set.

distribuzioni blu/verdi

Una strategia di implementazione in cui si creano due ambienti separati ma identici. La versione corrente dell'applicazione viene eseguita in un ambiente (blu) e la nuova versione dell'applicazione nell'altro ambiente (verde). Questa strategia consente di ripristinare rapidamente il sistema con un impatto minimo.

bot

Un'applicazione software che esegue attività automatizzate su Internet e simula l'attività o l'interazione umana. Alcuni bot sono utili o utili, come i web crawler che indicizzano le informazioni su Internet. Alcuni altri bot, noti come bot dannosi, hanno lo scopo di disturbare o causare danni a individui o organizzazioni.

botnet

Reti di bot infettate da malware e controllate da un'unica parte, nota come bot herder o bot operator. Le botnet sono il meccanismo più noto per scalare i bot e il loro impatto.

ramo

Un'area contenuta di un repository di codice. Il primo ramo creato in un repository è il ramo principale. È possibile creare un nuovo ramo a partire da un ramo esistente e quindi sviluppare funzionalità o correggere bug al suo interno. Un ramo creato per sviluppare una funzionalità viene comunemente detto ramo di funzionalità. Quando la funzionalità è pronta per il rilascio, il ramo di funzionalità viene ricongiunto al ramo principale. Per ulteriori informazioni, consulta Informazioni sulle filiali (documentazione). GitHub

accesso break-glass

In circostanze eccezionali e tramite una procedura approvata, un mezzo rapido per consentire a un utente di accedere a un sito a Account AWS cui in genere non dispone delle autorizzazioni necessarie. Per ulteriori informazioni, vedere l'indicatore Implementate break-glass procedures nella guida Well-Architected AWS .

strategia brownfield

L'infrastruttura esistente nell'ambiente. Quando si adotta una strategia brownfield per un'architettura di sistema, si progetta l'architettura in base ai vincoli dei sistemi e dell'infrastruttura attuali. Per l'espansione dell'infrastruttura esistente, è possibile combinare strategie brownfield e greenfield.

cache del buffer

L'area di memoria in cui sono archiviati i dati a cui si accede con maggiore frequenza.

capacità di business

Azioni intraprese da un'azienda per generare valore (ad esempio vendite, assistenza clienti o marketing). Le architetture dei microservizi e le decisioni di sviluppo possono essere guidate dalle capacità aziendali. Per ulteriori informazioni, consulta la sezione Organizzazione in base alle funzionalità aziendali del whitepaper Esecuzione di microservizi containerizzati su AWS.

pianificazione della continuità operativa (BCP)

Un piano che affronta il potenziale impatto di un evento che comporta l'interruzione dell'attività, come una migrazione su larga scala, sulle operazioni e consente a un'azienda di riprendere rapidamente le operazioni.

C

CAF

Vedi AWS Cloud Adoption Framework.

implementazione canaria

Il rilascio lento e incrementale di una versione agli utenti finali. Quando sei sicuro, distribuisci la nuova versione e sostituisci la versione corrente nella sua interezza.

CoE

Vedi Cloud Center of Excellence.

CDC

Vedi Change Data Capture.

Change Data Capture (CDC)

Il processo di tracciamento delle modifiche a un'origine dati, ad esempio una tabella di database, e di registrazione dei metadati relativi alla modifica. È possibile utilizzare CDC per vari scopi, ad esempio il controllo o la replica delle modifiche in un sistema di destinazione per mantenere la sincronizzazione.

ingegneria del caos

Introduzione intenzionale di guasti o eventi dirompenti per testare la resilienza di un sistema. Puoi usare AWS Fault Injection Service (AWS FIS) per eseguire esperimenti che stressano i tuoi AWS carichi di lavoro e valutarne la risposta.

CI/CD

Vedi integrazione continua e distribuzione continua.

classificazione

Un processo di categorizzazione che aiuta a generare previsioni. I modelli di ML per problemi di classificazione prevedono un valore discreto. I valori discreti sono sempre distinti l'uno dall'altro. Ad esempio, un modello potrebbe dover valutare se in un'immagine è presente o meno un'auto.

crittografia lato client

Crittografia dei dati a livello locale, prima che il destinatario li Servizio AWS riceva.

centro di eccellenza del cloud (CCoE)

Un team multidisciplinare che guida le iniziative di adozione del cloud in tutta l'organizzazione, tra cui lo sviluppo di best practice per il cloud, la mobilitazione delle risorse, la definizione delle tempistiche di migrazione e la guida dell'organizzazione attraverso trasformazioni su larga scala. Per ulteriori informazioni, consulta i post di CCoE sull' Cloud AWS Enterprise Strategy Blog.

cloud computing

La tecnologia cloud generalmente utilizzata per l'archiviazione remota di dati e la gestione dei dispositivi IoT. Il cloud computing è generalmente collegato alla tecnologia di edge computing.

modello operativo cloud

In un'organizzazione IT, il modello operativo utilizzato per creare, maturare e ottimizzare uno o più ambienti cloud. Per ulteriori informazioni, consulta Building your Cloud Operating Model.

fasi di adozione del cloud

Le quattro fasi che le organizzazioni in genere attraversano quando migrano verso Cloud AWS:

- Progetto: esecuzione di alcuni progetti relativi al cloud per scopi di dimostrazione e apprendimento
- Fondamento: effettuare investimenti fondamentali per dimensionare l'adozione del cloud (ad esempio, creazione di una zona di destinazione, definizione di un CCoE, definizione di un modello operativo)
- Migrazione: migrazione di singole applicazioni
- Reinvenzione: ottimizzazione di prodotti e servizi e innovazione nel cloud

Queste fasi sono state definite da Stephen Orban nel post del blog The Journey Toward Cloud-First & the Stages of Adoption on the Enterprise Strategy. Cloud AWS Per informazioni su come si relazionano alla strategia di AWS migrazione, consulta la guida alla preparazione alla migrazione.

CMDB

Vedi database di gestione della configurazione.

repository di codice

Una posizione in cui il codice di origine e altri asset, come documentazione, esempi e script, vengono archiviati e aggiornati attraverso processi di controllo delle versioni. Gli archivi cloud più comuni includono GitHub o AWS CodeCommit. Ogni versione del codice è denominata ramo. In

una struttura a microservizi, ogni repository è dedicato a una singola funzionalità. Una singola pipeline CI/CD può utilizzare più repository.

cache fredda

Una cache del buffer vuota, non ben popolata o contenente dati obsoleti o irrilevanti. Ciò influisce sulle prestazioni perché l'istanza di database deve leggere dalla memoria o dal disco principale, il che richiede più tempo rispetto alla lettura dalla cache del buffer.

dati freddi

Dati a cui si accede raramente e che in genere sono storici. Quando si eseguono interrogazioni di questo tipo di dati, le interrogazioni lente sono in genere accettabili. Lo spostamento di questi dati su livelli o classi di storage meno costosi e con prestazioni inferiori può ridurre i costi.

visione artificiale (CV)

Un campo dell'intelligenza artificiale che utilizza l'apprendimento automatico per analizzare ed estrarre informazioni da formati visivi come immagini e video digitali. Ad esempio, AWS Panorama offre dispositivi che aggiungono CV alle reti di telecamere locali e Amazon SageMaker fornisce algoritmi di elaborazione delle immagini per CV.

deriva della configurazione

Per un carico di lavoro, una modifica della configurazione rispetto allo stato previsto. Potrebbe causare la non conformità del carico di lavoro e in genere è graduale e involontaria.

database di gestione della configurazione (CMDB)

Un repository che archivia e gestisce le informazioni su un database e il relativo ambiente IT, inclusi i componenti hardware e software e le relative configurazioni. In genere si utilizzano i dati di un CMDB nella fase di individuazione e analisi del portafoglio della migrazione.

Pacchetto di conformità

Una raccolta di AWS Config regole e azioni correttive che puoi assemblare per personalizzare i controlli di conformità e sicurezza. È possibile distribuire un pacchetto di conformità come singola entità in una regione Account AWS and o all'interno di un'organizzazione utilizzando un modello YAML. Per ulteriori informazioni, consulta i Conformance Pack nella documentazione. AWS Config

integrazione e distribuzione continua (continuous integration and continuous delivery, CI/CD)

Il processo di automazione delle fasi di origine, creazione, test, gestione temporanea e produzione del processo di rilascio del software. Il processo CI/CD è comunemente descritto come una

pipeline. CI/CD può aiutare ad automatizzare i processi, migliorare la produttività, migliorare la qualità del codice e velocizzare le distribuzioni. Per ulteriori informazioni, consulta Vantaggi della distribuzione continua. CD può anche significare continuous deployment (implementazione continua). Per ulteriori informazioni, consulta Distribuzione continua e implementazione continua a confronto.

CV

Vedi visione artificiale.

D

dati a riposo

Dati stazionari nella rete, ad esempio i dati archiviati.

classificazione dei dati

Un processo per identificare e classificare i dati nella rete in base alla loro criticità e sensibilità. È un componente fondamentale di qualsiasi strategia di gestione dei rischi di sicurezza informatica perché consente di determinare i controlli di protezione e conservazione appropriati per i dati. La classificazione dei dati è un componente del pilastro della sicurezza nel AWS Well-Architected Framework. Per ulteriori informazioni, consulta Classificazione dei dati.

deriva dei dati

Una variazione significativa tra i dati di produzione e i dati utilizzati per addestrare un modello di machine learning o una modifica significativa dei dati di input nel tempo. La deriva dei dati può ridurre la qualità, l'accuratezza e l'equità complessive nelle previsioni dei modelli ML.

dati in transito

Dati che si spostano attivamente attraverso la rete, ad esempio tra le risorse di rete.

rete di dati

Un framework architettonico che fornisce la proprietà distribuita e decentralizzata dei dati con gestione e governance centralizzate.

riduzione al minimo dei dati

Il principio della raccolta e del trattamento dei soli dati strettamente necessari. Praticare la riduzione al minimo dei dati in the Cloud AWS può ridurre i rischi per la privacy, i costi e l'impronta di carbonio delle analisi.

perimetro dei dati

Una serie di barriere preventive nell' AWS ambiente che aiutano a garantire che solo le identità attendibili accedano alle risorse attendibili delle reti previste. Per ulteriori informazioni, consulta Building a data perimeter on. AWS

pre-elaborazione dei dati

Trasformare i dati grezzi in un formato che possa essere facilmente analizzato dal modello di ML. La pre-elaborazione dei dati può comportare la rimozione di determinate colonne o righe e l'eliminazione di valori mancanti, incoerenti o duplicati.

provenienza dei dati

Il processo di tracciamento dell'origine e della cronologia dei dati durante il loro ciclo di vita, ad esempio il modo in cui i dati sono stati generati, trasmessi e archiviati.

soggetto dei dati

Un individuo i cui dati vengono raccolti ed elaborati.

data warehouse

Un sistema di gestione dei dati che supporta la business intelligence, come l'analisi. I data warehouse contengono in genere grandi quantità di dati storici e vengono generalmente utilizzati per interrogazioni e analisi.

linguaggio di definizione del database (DDL)

Istruzioni o comandi per creare o modificare la struttura di tabelle e oggetti in un database.

linguaggio di manipolazione del database (DML)

Istruzioni o comandi per modificare (inserire, aggiornare ed eliminare) informazioni in un database.

DDL

Vedi linguaggio di definizione del database.

deep ensemble

Combinare più modelli di deep learning per la previsione. È possibile utilizzare i deep ensemble per ottenere una previsione più accurata o per stimare l'incertezza nelle previsioni.

deep learning

Un sottocampo del ML che utilizza più livelli di reti neurali artificiali per identificare la mappatura tra i dati di input e le variabili target di interesse.

defense-in-depth

Un approccio alla sicurezza delle informazioni in cui una serie di meccanismi e controlli di sicurezza sono accuratamente stratificati su una rete di computer per proteggere la riservatezza, l'integrità e la disponibilità della rete e dei dati al suo interno. Quando si adotta questa strategia AWS, si aggiungono più controlli a diversi livelli della AWS Organizations struttura per proteggere le risorse. Ad esempio, un defense-in-depth approccio potrebbe combinare l'autenticazione a più fattori, la segmentazione della rete e la crittografia.

amministratore delegato

In AWS Organizations, un servizio compatibile può registrare un account AWS membro per amministrare gli account dell'organizzazione e gestire le autorizzazioni per quel servizio. Questo account è denominato amministratore delegato per quel servizio specifico. Per ulteriori informazioni e un elenco di servizi compatibili, consulta Servizi che funzionano con AWS Organizations nella documentazione di AWS Organizations .

implementazione

Il processo di creazione di un'applicazione, di nuove funzionalità o di correzioni di codice disponibili nell'ambiente di destinazione. L'implementazione prevede l'applicazione di modifiche in una base di codice, seguita dalla creazione e dall'esecuzione di tale base di codice negli ambienti applicativi.

Ambiente di sviluppo

Vedi ambiente.

controllo di rilevamento

Un controllo di sicurezza progettato per rilevare, registrare e avvisare dopo che si è verificato un evento. Questi controlli rappresentano una seconda linea di difesa e avvisano l'utente in caso di eventi di sicurezza che aggirano i controlli preventivi in vigore. Per ulteriori informazioni, consulta Controlli di rilevamento in Implementazione dei controlli di sicurezza in AWS.

mappatura del flusso di valore dello sviluppo (DVSM)

Un processo utilizzato per identificare e dare priorità ai vincoli che influiscono negativamente sulla velocità e sulla qualità nel ciclo di vita dello sviluppo del software. DVSM estende il processo di

mappatura del flusso di valore originariamente progettato per pratiche di produzione snella. Si concentra sulle fasi e sui team necessari per creare e trasferire valore attraverso il processo di sviluppo del software.

gemello digitale

Una rappresentazione virtuale di un sistema reale, ad esempio un edificio, una fabbrica, un'attrezzatura industriale o una linea di produzione. I gemelli digitali supportano la manutenzione predittiva, il monitoraggio remoto e l'ottimizzazione della produzione.

tabella delle dimensioni

In uno schema a stella, una tabella più piccola che contiene gli attributi dei dati quantitativi in una tabella dei fatti. Gli attributi della tabella delle dimensioni sono in genere campi di testo o numeri discreti che si comportano come testo. Questi attributi vengono comunemente utilizzati per il vincolo delle query, il filtraggio e l'etichettatura dei set di risultati.

disastro

Un evento che impedisce a un carico di lavoro o a un sistema di raggiungere gli obiettivi aziendali nella sua sede principale di implementazione. Questi eventi possono essere disastri naturali, guasti tecnici o il risultato di azioni umane, come errori di configurazione involontari o attacchi di malware.

disaster recovery (DR)

La strategia e il processo utilizzati per ridurre al minimo i tempi di inattività e la perdita di dati causati da un disastro. Per ulteriori informazioni, consulta Disaster Recovery of Workloads su AWS: Recovery in the Cloud in the AWS Well-Architected Framework.

DML

Vedi linguaggio di manipolazione del database.

progettazione basata sul dominio

Un approccio allo sviluppo di un sistema software complesso collegandone i componenti a domini in evoluzione, o obiettivi aziendali principali, perseguiti da ciascun componente. Questo concetto è stato introdotto da Eric Evans nel suo libro,Domain-Driven Design: Tackling Complexity in the Heart of Software (Boston: Addison-Wesley Professional, 2003). Per informazioni su come utilizzare la progettazione basata sul dominio con il modello del fico strangolatore (Strangler Fig), consulta la sezione Modernizzazione incrementale dei servizi Web Microsoft ASP.NET (ASMX) legacy utilizzando container e il Gateway Amazon API.

DOTT.

Vedi disaster recovery.

rilevamento della deriva

Tracciamento delle deviazioni da una configurazione di base. Ad esempio, è possibile AWS CloudFormation utilizzarlo per rilevare deviazioni nelle risorse di sistema oppure AWS Control Tower per rilevare cambiamenti nella landing zone che potrebbero influire sulla conformità ai requisiti di governance.

DVSM

Vedi la mappatura del flusso di valore dello sviluppo.

E

EDA

Vedi analisi esplorativa dei dati.

edge computing

La tecnologia che aumenta la potenza di calcolo per i dispositivi intelligenti all'edge di una rete IoT. Rispetto al cloud computing, l'edge computing può ridurre la latenza di comunicazione e migliorare i tempi di risposta.

crittografia

Un processo di elaborazione che trasforma i dati in chiaro, leggibili dall'uomo, in testo cifrato.

chiave crittografica

Una stringa crittografica di bit randomizzati generata da un algoritmo di crittografia. Le chiavi possono variare di lunghezza e ogni chiave è progettata per essere imprevedibile e univoca.

endianità

L'ordine in cui i byte vengono archiviati nella memoria del computer. I sistemi big-endian memorizzano per primo il byte più importante. I sistemi little-endian memorizzano per primo il byte meno importante.

endpoint

Vedi service endpoint.

servizio endpoint

Un servizio che puoi ospitare in un cloud privato virtuale (VPC) da condividere con altri utenti. Puoi creare un servizio endpoint con AWS PrivateLink e concedere autorizzazioni ad altri Account AWS o a AWS Identity and Access Management (IAM) principali. Questi account o principali possono connettersi al servizio endpoint in privato creando endpoint VPC di interfaccia. Per ulteriori informazioni, consulta Creazione di un servizio endpoint nella documentazione di Amazon Virtual Private Cloud (Amazon VPC).

pianificazione delle risorse aziendali (ERP)

Un sistema che automatizza e gestisce i processi aziendali chiave (come contabilità, MES e gestione dei progetti) per un'azienda.

crittografia envelope

Il processo di crittografia di una chiave di crittografia con un'altra chiave di crittografia. Per ulteriori informazioni, vedete Envelope encryption nella documentazione AWS Key Management Service (AWS KMS).

ambiente

Un'istanza di un'applicazione in esecuzione. Di seguito sono riportati i tipi di ambiente più comuni nel cloud computing:

- ambiente di sviluppo: un'istanza di un'applicazione in esecuzione disponibile solo per il team principale responsabile della manutenzione dell'applicazione. Gli ambienti di sviluppo vengono utilizzati per testare le modifiche prima di promuoverle negli ambienti superiori. Questo tipo di ambiente viene talvolta definito ambiente di test.
- ambienti inferiori: tutti gli ambienti di sviluppo di un'applicazione, ad esempio quelli utilizzati per le build e i test iniziali.
- ambiente di produzione: un'istanza di un'applicazione in esecuzione a cui gli utenti finali possono accedere. In una pipeline CI/CD, l'ambiente di produzione è l'ultimo ambiente di implementazione.
- ambienti superiori: tutti gli ambienti a cui possono accedere utenti diversi dal team di sviluppo principale. Si può trattare di un ambiente di produzione, ambienti di preproduzione e ambienti per i test di accettazione da parte degli utenti.

epica

Nelle metodologie agili, categorie funzionali che aiutano a organizzare e dare priorità al lavoro. Le epiche forniscono una descrizione di alto livello dei requisiti e delle attività di implementazione.

Ad esempio, le epopee della sicurezza AWS CAF includono la gestione delle identità e degli accessi, i controlli investigativi, la sicurezza dell'infrastruttura, la protezione dei dati e la risposta agli incidenti. Per ulteriori informazioni sulle epiche, consulta la strategia di migrazione AWS , consulta la guida all'implementazione del programma.

ERP

Vedi la pianificazione delle risorse aziendali.

analisi esplorativa dei dati (EDA)

Il processo di analisi di un set di dati per comprenderne le caratteristiche principali. Si raccolgono o si aggregano dati e quindi si eseguono indagini iniziali per trovare modelli, rilevare anomalie e verificare ipotesi. L'EDA viene eseguita calcolando statistiche di riepilogo e creando visualizzazioni di dati.

F

tabella dei fatti

Il tavolo centrale in uno schema a stella. Memorizza dati quantitativi sulle operazioni aziendali. In genere, una tabella dei fatti contiene due tipi di colonne: quelle che contengono misure e quelle che contengono una chiave esterna per una tabella di dimensioni.

fallire velocemente

Una filosofia che utilizza test frequenti e incrementali per ridurre il ciclo di vita dello sviluppo. È una parte fondamentale di un approccio agile.

limite di isolamento dei guasti

Nel Cloud AWS, un limite come una zona di disponibilità Regione AWS, un piano di controllo o un piano dati che limita l'effetto di un errore e aiuta a migliorare la resilienza dei carichi di lavoro. Per ulteriori informazioni, consulta AWS Fault Isolation Boundaries.

ramo di funzionalità

Vedi filiale.

caratteristiche

I dati di input che usi per fare una previsione. Ad esempio, in un contesto di produzione, le caratteristiche potrebbero essere immagini acquisite periodicamente dalla linea di produzione.

importanza delle caratteristiche

Quanto è importante una caratteristica per le previsioni di un modello. Di solito viene espresso come punteggio numerico che può essere calcolato con varie tecniche, come Shapley Additive Explanations (SHAP) e gradienti integrati. Per ulteriori informazioni, vedere Interpretabilità del modello di machine learning con:AWS.

trasformazione delle funzionalità

Per ottimizzare i dati per il processo di machine learning, incluso l'arricchimento dei dati con fonti aggiuntive, il dimensionamento dei valori o l'estrazione di più set di informazioni da un singolo campo di dati. Ciò consente al modello di ML di trarre vantaggio dai dati. Ad esempio, se suddividi la data "2021-05-27 00:15:37" in "2021", "maggio", "giovedì" e "15", puoi aiutare l'algoritmo di apprendimento ad apprendere modelli sfumati associati a diversi componenti dei dati.

FGAC

Vedi il controllo granulare degli accessi.

controllo granulare degli accessi (FGAC)

L'uso di più condizioni per consentire o rifiutare una richiesta di accesso.

migrazione flash-cut

Un metodo di migrazione del database che utilizza la replica continua dei dati tramite l'acquisizione dei dati delle modifiche per migrare i dati nel più breve tempo possibile, anziché utilizzare un approccio graduale. L'obiettivo è ridurre al minimo i tempi di inattività.

G

blocco geografico

Vedi restrizioni geografiche.

limitazioni geografiche (blocco geografico)

In Amazon CloudFront, un'opzione per impedire agli utenti di determinati paesi di accedere alle distribuzioni di contenuti. Puoi utilizzare un elenco consentito o un elenco di blocco per specificare i paesi approvati e vietati. Per ulteriori informazioni, consulta Limitare la distribuzione geografica dei contenuti nella CloudFront documentazione.

Flusso di lavoro di GitFlow

Un approccio in cui gli ambienti inferiori e superiori utilizzano rami diversi in un repository di codice di origine. Il flusso di lavoro Gitflow è considerato obsoleto e il flusso di lavoro basato su trunk è l'approccio moderno e preferito.

strategia greenfield

L'assenza di infrastrutture esistenti in un nuovo ambiente. Quando si adotta una strategia greenfield per un'architettura di sistema, è possibile selezionare tutte le nuove tecnologie senza il vincolo della compatibilità con l'infrastruttura esistente, nota anche come brownfield. Per l'espansione dell'infrastruttura esistente, è possibile combinare strategie brownfield e greenfield.

guardrail

Una regola di livello elevato che consente di governare risorse, policy e conformità tra le unità organizzative (OU). I guardrail preventivi applicano le policy per garantire l'allineamento agli standard di conformità. Vengono implementati utilizzando le policy di controllo dei servizi e i limiti delle autorizzazioni IAM. I guardrail di rilevamento rilevano le violazioni delle policy e i problemi di conformità e generano avvisi per porvi rimedio. Sono implementati utilizzando Amazon AWS Config AWS Security Hub GuardDuty AWS Trusted Advisor, Amazon Inspector e controlli personalizzati AWS Lambda .

H

AH

Vedi disponibilità elevata.

migrazione di database eterogenea

Migrazione del database di origine in un database di destinazione che utilizza un motore di database diverso (ad esempio, da Oracle ad Amazon Aurora). La migrazione eterogenea fa in genere parte di uno sforzo di riprogettazione e la conversione dello schema può essere un'attività complessa. AWS offre AWS SCT che aiuta con le conversioni dello schema.

alta disponibilità (HA)

La capacità di un carico di lavoro di funzionare in modo continuo, senza intervento, in caso di sfide o disastri. I sistemi HA sono progettati per il failover automatico, fornire costantemente prestazioni di alta qualità e gestire carichi e guasti diversi con un impatto minimo sulle prestazioni.

modernizzazione storica

Un approccio utilizzato per modernizzare e aggiornare i sistemi di tecnologia operativa (OT) per soddisfare meglio le esigenze dell'industria manifatturiera. Uno storico è un tipo di database utilizzato per raccogliere e archiviare dati da varie fonti in una fabbrica.

migrazione di database omogenea

Migrazione del database di origine in un database di destinazione che condivide lo stesso motore di database (ad esempio, da Microsoft SQL Server ad Amazon RDS per SQL Server). La migrazione omogenea fa in genere parte di un'operazione di rehosting o ridefinizione della piattaforma. Per migrare lo schema è possibile utilizzare le utilità native del database.

dati caldi

Dati a cui si accede frequentemente, ad esempio dati in tempo reale o dati di traduzione recenti. Questi dati richiedono in genere un livello o una classe di storage ad alte prestazioni per fornire risposte rapide alle query.

hotfix

Una soluzione urgente per un problema critico in un ambiente di produzione. A causa della sua urgenza, un hotfix viene in genere creato al di fuori del tipico DevOps flusso di lavoro di rilascio.

periodo di hypercare

Subito dopo la conversione, il periodo di tempo in cui un team di migrazione gestisce e monitora le applicazioni migrate nel cloud per risolvere eventuali problemi. In genere, questo periodo dura da 1 a 4 giorni. Al termine del periodo di hypercare, il team addetto alla migrazione in genere trasferisce la responsabilità delle applicazioni al team addetto alle operazioni cloud.

I

IaC

Considera l'infrastruttura come codice.

Policy basata su identità

Una policy associata a uno o più principi IAM che definisce le relative autorizzazioni all'interno dell' Cloud AWS ambiente.

applicazione inattiva

Un'applicazione che prevede un uso di CPU e memoria medio compreso tra il 5% e il 20% in un periodo di 90 giorni. In un progetto di migrazione, è normale ritirare queste applicazioni o mantenerle on-premise.

IIoT

Vedi Industrial Internet of Things.

infrastruttura immutabile

Un modello che implementa una nuova infrastruttura per i carichi di lavoro di produzione anziché aggiornare, applicare patch o modificare l'infrastruttura esistente. Le infrastrutture immutabili sono intrinsecamente più coerenti, affidabili e prevedibili delle infrastrutture mutabili. Per ulteriori informazioni, consulta la best practice Deploy using immutable infrastructure in Well-Architected AWS Framework.

VPC in ingresso (ingress)

In un'architettura AWS multi-account, un VPC che accetta, ispeziona e indirizza le connessioni di rete dall'esterno di un'applicazione. Nel documento Architettura di riferimento per la sicurezza di AWS si consiglia di configurare l'account di rete con VPC in entrata, in uscita e di ispezione per proteggere l'interfaccia bidirezionale tra l'applicazione e Internet in generale.

migrazione incrementale

Una strategia di conversione in cui si esegue la migrazione dell'applicazione in piccole parti anziché eseguire una conversione singola e completa. Ad esempio, inizialmente potresti spostare solo alcuni microservizi o utenti nel nuovo sistema. Dopo aver verificato che tutto funzioni correttamente, puoi spostare in modo incrementale microservizi o utenti aggiuntivi fino alla disattivazione del sistema legacy. Questa strategia riduce i rischi associati alle migrazioni di grandi dimensioni.

Industria 4.0

Un termine introdotto da Klaus Schwab nel 2016 per riferirsi alla modernizzazione dei processi di produzione attraverso progressi in termini di connettività, dati in tempo reale, automazione, analisi e AI/ML.

infrastruttura

Tutte le risorse e gli asset contenuti nell'ambiente di un'applicazione.

infrastruttura come codice (IaC)

Il processo di provisioning e gestione dell'infrastruttura di un'applicazione tramite un insieme di file di configurazione. Il processo IaC è progettato per aiutarti a centralizzare la gestione dell'infrastruttura, a standardizzare le risorse e a dimensionare rapidamente, in modo che i nuovi ambienti siano ripetibili, affidabili e coerenti.

Internet delle cose industriale (IIoT)

L'uso di sensori e dispositivi connessi a Internet nei settori industriali, come quello manifatturiero, energetico, automobilistico, sanitario, delle scienze della vita e dell'agricoltura. Per ulteriori informazioni, consulta Creazione di una strategia di trasformazione digitale dell'Internet delle cose industriale (IIoT).

VPC di ispezione

In un'architettura AWS multi-account, un VPC centralizzato che gestisce le ispezioni del traffico di rete tra VPC (uguali o diversi Regioni AWS), Internet e reti locali. Nel documento Architettura di riferimento per la sicurezza di AWS si consiglia di configurare l'account di rete con VPC in entrata, in uscita e di ispezione per proteggere l'interfaccia bidirezionale tra l'applicazione e Internet in generale.

Internet of Things (IoT)

La rete di oggetti fisici connessi con sensori o processori incorporati che comunicano con altri dispositivi e sistemi tramite Internet o una rete di comunicazione locale. Per ulteriori informazioni, consulta Cos'è l'IoT?

interpretabilità

Una caratteristica di un modello di machine learning che descrive il grado in cui un essere umano è in grado di comprendere in che modo le previsioni del modello dipendono dai suoi input. Per ulteriori informazioni, consulta la sezione Interpretabilità dei modelli di machine learning con AWS.

IoT

Vedi Internet of Things.

libreria di informazioni IT (ITIL)

Una serie di best practice per offrire servizi IT e allinearli ai requisiti aziendali. ITIL fornisce le basi per ITSM.

gestione dei servizi IT (ITSM)

Attività associate alla progettazione, implementazione, gestione e supporto dei servizi IT per un'organizzazione. Per informazioni sull'integrazione delle operazioni cloud con gli strumenti ITSM, consulta la guida all'integrazione delle operazioni.

ITIL

Vedi la libreria di informazioni IT.

ITSM

Vedi Gestione dei servizi IT.

L

controllo degli accessi basato su etichette (LBAC)

Un'implementazione del controllo di accesso obbligatorio (MAC) in cui agli utenti e ai dati stessi viene assegnato esplicitamente un valore di etichetta di sicurezza. L'intersezione tra l'etichetta di sicurezza utente e l'etichetta di sicurezza dei dati determina quali righe e colonne possono essere visualizzate dall'utente.

zona di destinazione

Una landing zone è un AWS ambiente multi-account ben progettato, scalabile e sicuro. Questo è un punto di partenza dal quale le organizzazioni possono avviare e distribuire rapidamente carichi di lavoro e applicazioni con fiducia nel loro ambiente di sicurezza e infrastruttura. Per ulteriori informazioni sulle zone di destinazione, consulta la sezione Configurazione di un ambiente AWS multi-account sicuro e scalabile.

migrazione su larga scala

Una migrazione di 300 o più server.

BIANCO

Vedi controllo degli accessi basato su etichette.

Privilegio minimo

La best practice di sicurezza per la concessione delle autorizzazioni minime richieste per eseguire un'attività. Per ulteriori informazioni, consulta Applicazione delle autorizzazioni del privilegio minimo nella documentazione di IAM.

eseguire il rehosting (lift and shift)

Vedi 7 R.

sistema little-endian

Un sistema che memorizza per primo il byte meno importante. Vedi anche endianità.

ambienti inferiori

Vedi ambiente.

M

machine learning (ML)

Un tipo di intelligenza artificiale che utilizza algoritmi e tecniche per il riconoscimento e l'apprendimento di schemi. Il machine learning analizza e apprende dai dati registrati, come i dati dell'Internet delle cose (IoT), per generare un modello statistico basato su modelli. Per ulteriori informazioni, consulta la sezione Machine learning.

ramo principale

Vedi filiale.

malware

Software progettato per compromettere la sicurezza o la privacy del computer. Il malware potrebbe interrompere i sistemi informatici, divulgare informazioni sensibili o ottenere accessi non autorizzati. Esempi di malware includono virus, worm, ransomware, trojan horse, spyware e keylogger.

servizi gestiti

Servizi AWS per cui AWS gestisce il livello di infrastruttura, il sistema operativo e le piattaforme e si accede agli endpoint per archiviare e recuperare i dati. Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3) e Amazon DynamoDB sono esempi di servizi gestiti. Questi sono noti anche come servizi astratti.

sistema di esecuzione della produzione (MES)

Un sistema software per tracciare, monitorare, documentare e controllare i processi di produzione che convertono le materie prime in prodotti finiti in officina.

MAP

Vedi Migration Acceleration Program.

meccanismo

Un processo completo in cui si crea uno strumento, si promuove l'adozione dello strumento e quindi si esaminano i risultati per apportare le modifiche. Un meccanismo è un ciclo che si rafforza e si migliora man mano che funziona. Per ulteriori informazioni, consulta Creazione di meccanismi nel AWS Well-Architected Framework.

account membro

Tutti gli account Account AWS diversi dall'account di gestione che fanno parte di un'organizzazione in. AWS Organizations Un account può essere membro di una sola organizzazione alla volta.

MEH

Vedi sistema di esecuzione della produzione.

Message Queuing Telemetry Transport (MQTT)

Un protocollo di comunicazione machine-to-machine (M2M) leggero, basato sul modello di pubblicazione/sottoscrizione, per dispositivi IoT con risorse limitate.

microservizio

Un piccolo servizio indipendente che comunica tramite API ben definite ed è in genere di proprietà di piccoli team autonomi. Ad esempio, un sistema assicurativo potrebbe includere microservizi che si riferiscono a funzionalità aziendali, come vendite o marketing, o sottodomini, come acquisti, reclami o analisi. I vantaggi dei microservizi includono agilità, dimensionamento flessibile, facilità di implementazione, codice riutilizzabile e resilienza. Per ulteriori informazioni, consulta Integrazione dei microservizi utilizzando servizi serverless. AWS

architettura di microservizi

Un approccio alla creazione di un'applicazione con componenti indipendenti che eseguono ogni processo applicativo come microservizio. Questi microservizi comunicano tramite un'interfaccia ben definita utilizzando API leggere. Ogni microservizio in questa architettura può essere aggiornato, distribuito e dimensionato per soddisfare la richiesta di funzioni specifiche di un'applicazione. Per ulteriori informazioni, vedere Implementazione dei microservizi su. AWS

Programma di accelerazione della migrazione (MAP)

Un AWS programma che fornisce consulenza, supporto, formazione e servizi per aiutare le organizzazioni a costruire una solida base operativa per il passaggio al cloud e per contribuire a compensare il costo iniziale delle migrazioni. MAP include una metodologia di migrazione per eseguire le migrazioni precedenti in modo metodico e un set di strumenti per automatizzare e accelerare gli scenari di migrazione comuni.

migrazione su larga scala

Il processo di trasferimento della maggior parte del portfolio di applicazioni sul cloud avviene a ondate, con più applicazioni trasferite a una velocità maggiore in ogni ondata. Questa fase utilizza le migliori pratiche e le lezioni apprese nelle fasi precedenti per implementare una fabbrica di migrazione di team, strumenti e processi per semplificare la migrazione dei carichi di lavoro attraverso l'automazione e la distribuzione agile. Questa è la terza fase della strategia di migrazione AWS.

fabbrica di migrazione

Team interfunzionali che semplificano la migrazione dei carichi di lavoro attraverso approcci automatizzati e agili. I team di Migration Factory includono in genere operazioni, analisti e proprietari aziendali, ingegneri addetti alla migrazione, sviluppatori e DevOps professionisti che lavorano nell'ambito degli sprint. Tra il 20% e il 50% di un portfolio di applicazioni aziendali è costituito da schemi ripetuti che possono essere ottimizzati con un approccio di fabbrica. Per ulteriori informazioni, consulta la discussione sulle fabbriche di migrazione e la Guida alla fabbrica di migrazione al cloud in questo set di contenuti.

metadati di migrazione

Le informazioni sull'applicazione e sul server necessarie per completare la migrazione. Ogni modello di migrazione richiede un set diverso di metadati di migrazione. Esempi di metadati di migrazione includono la sottorete, il gruppo di sicurezza e l'account di destinazione. AWS

modello di migrazione

Un'attività di migrazione ripetibile che descrive in dettaglio la strategia di migrazione, la destinazione della migrazione e l'applicazione o il servizio di migrazione utilizzati. Esempio: riorganizza la migrazione su Amazon EC2 AWS con Application Migration Service.

Valutazione del portfolio di migrazione (MPA)

Uno strumento online che fornisce informazioni per la convalida del business case per la migrazione a. Cloud AWS MPA offre una valutazione dettagliata del portfolio (dimensionamento

corretto dei server, prezzi, confronto del TCO, analisi dei costi di migrazione) e pianificazione della migrazione (analisi e raccolta dei dati delle applicazioni, raggruppamento delle applicazioni, prioritizzazione delle migrazioni e pianificazione delle ondate). Lo strumento MPA (richiede l'accesso) è disponibile gratuitamente per tutti i AWS consulenti e i consulenti dei partner APN.

valutazione della preparazione alla migrazione (MRA)

Il processo di acquisizione di informazioni sullo stato di preparazione al cloud di un'organizzazione, l'identificazione dei punti di forza e di debolezza e la creazione di un piano d'azione per colmare le lacune identificate, utilizzando il CAF. AWS Per ulteriori informazioni, consulta la guida di preparazione alla migrazione. MRA è la prima fase della strategia di migrazione AWS.

strategia di migrazione

L'approccio utilizzato per migrare un carico di lavoro verso. Cloud AWS Per ulteriori informazioni, consulta la voce 7 R in questo glossario e consulta Mobilita la tua organizzazione per accelerare le migrazioni su larga scala.

ML

Vedi machine learning.

modernizzazione

Trasformazione di un'applicazione obsoleta (legacy o monolitica) e della relativa infrastruttura in un sistema agile, elastico e altamente disponibile nel cloud per ridurre i costi, aumentare l'efficienza e sfruttare le innovazioni. Per ulteriori informazioni, vedere Strategia per la modernizzazione delle applicazioni in. Cloud AWS

valutazione della preparazione alla modernizzazione

Una valutazione che aiuta a determinare la preparazione alla modernizzazione delle applicazioni di un'organizzazione, identifica vantaggi, rischi e dipendenze e determina in che misura l'organizzazione può supportare lo stato futuro di tali applicazioni. Il risultato della valutazione è uno schema dell'architettura di destinazione, una tabella di marcia che descrive in dettaglio le fasi di sviluppo e le tappe fondamentali del processo di modernizzazione e un piano d'azione per colmare le lacune identificate. Per ulteriori informazioni, vedere Valutazione della preparazione alla modernizzazione per le applicazioni in. Cloud AWS

applicazioni monolitiche (monoliti)

Applicazioni eseguite come un unico servizio con processi strettamente collegati. Le applicazioni monolitiche presentano diversi inconvenienti. Se una funzionalità dell'applicazione registra un

picco di domanda, l'intera architettura deve essere dimensionata. L'aggiunta o il miglioramento delle funzionalità di un'applicazione monolitica diventa inoltre più complessa man mano che la base di codice cresce. Per risolvere questi problemi, puoi utilizzare un'architettura di microservizi. Per ulteriori informazioni, consulta la sezione Scomposizione dei monoliti in microservizi.

MAPPA

Vedi Migration Portfolio Assessment.

MQTT

Vedi Message Queuing Telemetry Transport.

classificazione multiclasse

Un processo che aiuta a generare previsioni per più classi (prevedendo uno o più di due risultati). Ad esempio, un modello di machine learning potrebbe chiedere "Questo prodotto è un libro, un'auto o un telefono?" oppure "Quale categoria di prodotti è più interessante per questo cliente?"

infrastruttura mutabile

Un modello che aggiorna e modifica l'infrastruttura esistente per i carichi di lavoro di produzione. Per migliorare la coerenza, l'affidabilità e la prevedibilità, il AWS Well-Architected Framework consiglia l'uso di un'infrastruttura immutabile come best practice.

O

OAC

Vedi Origin Access Control.

QUERCIA

Vedi Origin Access Identity.

OCM

Vedi gestione delle modifiche organizzative.

migrazione offline

Un metodo di migrazione in cui il carico di lavoro di origine viene eliminato durante il processo di migrazione. Questo metodo prevede tempi di inattività prolungati e viene in genere utilizzato per carichi di lavoro piccoli e non critici.

OI

Vedi l'integrazione delle operazioni.

OLA

Vedi accordo a livello operativo.

migrazione online

Un metodo di migrazione in cui il carico di lavoro di origine viene copiato sul sistema di destinazione senza essere messo offline. Le applicazioni connesse al carico di lavoro possono continuare a funzionare durante la migrazione. Questo metodo comporta tempi di inattività pari a zero o comunque minimi e viene in genere utilizzato per carichi di lavoro di produzione critici.

OPC-UA

Vedi Open Process Communications - Unified Architecture.

Comunicazioni a processo aperto - Architettura unificata (OPC-UA)

Un protocollo di comunicazione machine-to-machine (M2M) per l'automazione industriale. OPC-UA fornisce uno standard di interoperabilità con schemi di crittografia, autenticazione e autorizzazione dei dati.

accordo a livello operativo (OLA)

Un accordo che chiarisce quali sono gli impegni reciproci tra i gruppi IT funzionali, a supporto di un accordo sul livello di servizio (SLA).

revisione della prontezza operativa (ORR)

Un elenco di domande e best practice associate che aiutano a comprendere, valutare, prevenire o ridurre la portata degli incidenti e dei possibili guasti. Per ulteriori informazioni, vedere Operational Readiness Reviews (ORR) nel Well-Architected AWS Framework.

tecnologia operativa (OT)

Sistemi hardware e software che interagiscono con l'ambiente fisico per controllare le operazioni, le apparecchiature e le infrastrutture industriali. Nella produzione, l'integrazione di sistemi OT e di tecnologia dell'informazione (IT) è un obiettivo chiave per le trasformazioni dell'Industria 4.0.

integrazione delle operazioni (OI)

Il processo di modernizzazione delle operazioni nel cloud, che prevede la pianificazione, l'automazione e l'integrazione della disponibilità. Per ulteriori informazioni, consulta la guida all'integrazione delle operazioni.

trail organizzativo

Un percorso creato da noi AWS CloudTrail che registra tutti gli eventi di un'organizzazione per tutti Account AWS . AWS Organizations Questo percorso viene creato in ogni Account AWS che fa parte dell'organizzazione e tiene traccia dell'attività in ogni account. Per ulteriori informazioni, consulta Creazione di un percorso per un'organizzazione nella CloudTrail documentazione.

gestione del cambiamento organizzativo (OCM)

Un framework per la gestione di trasformazioni aziendali importanti e che comportano l'interruzione delle attività dal punto di vista delle persone, della cultura e della leadership. OCM aiuta le organizzazioni a prepararsi e passare a nuovi sistemi e strategie accelerando l'adozione del cambiamento, affrontando i problemi di transizione e promuovendo cambiamenti culturali e organizzativi. Nella strategia di AWS migrazione, questo framework si chiama accelerazione delle persone, a causa della velocità di cambiamento richiesta nei progetti di adozione del cloud. Per ulteriori informazioni, consultare la Guida OCM.

controllo dell'accesso all'origine (OAC)

In CloudFront, un'opzione avanzata per limitare l'accesso per proteggere i contenuti di Amazon Simple Storage Service (Amazon S3). OAC supporta tutti i bucket S3 in generale Regioni AWS, la crittografia lato server con AWS KMS (SSE-KMS) e le richieste dinamiche e dirette al bucket S3. `PUT` `DELETE`

identità di accesso origine (OAI)

Nel CloudFront, un'opzione per limitare l'accesso per proteggere i tuoi contenuti Amazon S3. Quando usi OAI, CloudFront crea un principale con cui Amazon S3 può autenticarsi. I principali autenticati possono accedere ai contenuti in un bucket S3 solo tramite una distribuzione specifica. CloudFront Vedi anche OAC, che fornisce un controllo degli accessi più granulare e avanzato.

O

Vedi la revisione della prontezza operativa.

- NON

Vedi la tecnologia operativa.

VPC in uscita (egress)

In un'architettura AWS multi-account, un VPC che gestisce le connessioni di rete avviate dall'interno di un'applicazione. Nel documento Architettura di riferimento per la sicurezza di

AWS si consiglia di configurare l'account di rete con VPC in entrata, in uscita e di ispezione per proteggere l'interfaccia bidirezionale tra l'applicazione e Internet in generale.

P

limite delle autorizzazioni

Una policy di gestione IAM collegata ai principali IAM per impostare le autorizzazioni massime che l'utente o il ruolo possono avere. Per ulteriori informazioni, consulta Limiti delle autorizzazioni nella documentazione di IAM.

informazioni di identificazione personale (PII)

Informazioni che, se visualizzate direttamente o abbinate ad altri dati correlati, possono essere utilizzate per dedurre ragionevolmente l'identità di un individuo. Esempi di informazioni personali includono nomi, indirizzi e informazioni di contatto.

Informazioni che consentono l'identificazione personale degli utenti

Visualizza le informazioni di identificazione personale.

playbook

Una serie di passaggi predefiniti che raccolgono il lavoro associato alle migrazioni, come l'erogazione delle funzioni operative principali nel cloud. Un playbook può assumere la forma di script, runbook automatici o un riepilogo dei processi o dei passaggi necessari per gestire un ambiente modernizzato.

PLC

Vedi controllore logico programmabile.

PLM

Vedi la gestione del ciclo di vita del prodotto.

policy

Un oggetto in grado di definire le autorizzazioni (vedi politica basata sull'identità), specificare le condizioni di accesso (vedi politicabasata sulle risorse) o definire le autorizzazioni massime per tutti gli account di un'organizzazione in (vedi politica di controllo dei servizi). AWS Organizations

persistenza poliglotta

Scelta indipendente della tecnologia di archiviazione di dati di un microservizio in base ai modelli di accesso ai dati e ad altri requisiti. Se i microservizi utilizzano la stessa tecnologia di archiviazione di dati, possono incontrare problemi di implementazione o registrare prestazioni scadenti. I microservizi vengono implementati più facilmente e ottengono prestazioni e scalabilità migliori se utilizzano l'archivio dati più adatto alle loro esigenze. Per ulteriori informazioni, consulta la sezione Abilitazione della persistenza dei dati nei microservizi.

valutazione del portfolio

Un processo di scoperta, analisi e definizione delle priorità del portfolio di applicazioni per pianificare la migrazione. Per ulteriori informazioni, consulta la pagina Valutazione della preparazione alla migrazione.

predicate

Una condizione di interrogazione che restituisce o, in genere, si trova in una clausola`true`. `false` `WHERE`

predicato pushdown

Una tecnica di ottimizzazione delle query del database che filtra i dati della query prima del trasferimento. Ciò riduce la quantità di dati che devono essere recuperati ed elaborati dal database relazionale e migliora le prestazioni delle query.

controllo preventivo

Un controllo di sicurezza progettato per impedire il verificarsi di un evento. Questi controlli sono la prima linea di difesa per impedire accessi non autorizzati o modifiche indesiderate alla rete. Per ulteriori informazioni, consulta Controlli preventivi in Implementazione dei controlli di sicurezza in AWS.

principale

Un'entità in AWS grado di eseguire azioni e accedere alle risorse. Questa entità è in genere un utente root per un Account AWS ruolo IAM o un utente. Per ulteriori informazioni, consulta Principali in Termini e concetti dei ruoli nella documentazione di IAM.

Privacy fin dalla progettazione

Un approccio all'ingegneria dei sistemi che tiene conto della privacy durante l'intero processo di progettazione.

zone ospitate private

Un container che contiene informazioni su come si desidera che Amazon Route 53 risponda alle query DNS per un dominio e i relativi sottodomini all'interno di uno o più VPC. Per ulteriori informazioni, consulta Utilizzo delle zone ospitate private nella documentazione di Route 53.

controllo proattivo

Un controllo di sicurezza progettato per impedire l'implementazione di risorse non conformi. Questi controlli analizzano le risorse prima del loro provisioning. Se la risorsa non è conforme al controllo, non viene fornita. Per ulteriori informazioni, consulta la guida di riferimento sui controlli nella AWS Control Tower documentazione e consulta Controlli proattivi in Implementazione dei controlli di sicurezza su. AWS

gestione del ciclo di vita del prodotto (PLM)

La gestione dei dati e dei processi di un prodotto durante l'intero ciclo di vita, dalla progettazione, sviluppo e lancio, attraverso la crescita e la maturità, fino al declino e alla rimozione.

Ambiente di produzione

Vedi ambiente.

controllore logico programmabile (PLC)

Nella produzione, un computer altamente affidabile e adattabile che monitora le macchine e automatizza i processi di produzione.

pseudonimizzazione

Il processo di sostituzione degli identificatori personali in un set di dati con valori segnaposto. La pseudonimizzazione può aiutare a proteggere la privacy personale. I dati pseudonimizzati sono ancora considerati dati personali.

pubblica/sottoscrivi (pub/sub)

Un pattern che consente comunicazioni asincrone tra microservizi per migliorare la scalabilità e la reattività. Ad esempio, in un MES basato su microservizi, un microservizio può pubblicare messaggi di eventi su un canale a cui altri microservizi possono abbonarsi. Il sistema può aggiungere nuovi microservizi senza modificare il servizio di pubblicazione.

Q

Piano di query

Una serie di passaggi, come le istruzioni, utilizzati per accedere ai dati in un sistema di database relazionale SQL.

regressione del piano di query

Quando un ottimizzatore del servizio di database sceglie un piano non ottimale rispetto a prima di una determinata modifica all'ambiente di database. Questo può essere causato da modifiche a statistiche, vincoli, impostazioni dell'ambiente, associazioni dei parametri di query e aggiornamenti al motore di database.

R

Matrice RACI

Vedi responsabile, responsabile, consultato, informato (RACI).

ransomware

Un software dannoso progettato per bloccare l'accesso a un sistema informatico o ai dati fino a quando non viene effettuato un pagamento.

Matrice RASCI

Vedi responsabile, responsabile, consultato, informato (RACI).

RCAC

Vedi il controllo dell'accesso a righe e colonne.

replica di lettura

Una copia di un database utilizzata per scopi di sola lettura. È possibile indirizzare le query alla replica di lettura per ridurre il carico sul database principale.

riprogettare

Vedi 7 Rs.

obiettivo del punto di ripristino (RPO)

Il periodo di tempo massimo accettabile dall'ultimo punto di ripristino dei dati. Ciò determina quella che viene considerata una perdita di dati accettabile tra l'ultimo punto di ripristino e l'interruzione del servizio.

obiettivo del tempo di ripristino (RTO)

Il ritardo massimo accettabile tra l'interruzione del servizio e il ripristino del servizio.

rifattorizzare

Vedi 7 R.

Regione

Una raccolta di AWS risorse in un'area geografica. Ciascuna Regione AWS è isolata e indipendente dalle altre per fornire tolleranza agli errori, stabilità e resilienza. Per ulteriori informazioni, consulta Specificare cosa può usare Regioni AWS il tuo account.

regressione

Una tecnica di ML che prevede un valore numerico. Ad esempio, per risolvere il problema "A che prezzo verrà venduta questa casa?" un modello di ML potrebbe utilizzare un modello di regressione lineare per prevedere il prezzo di vendita di una casa sulla base di dati noti sulla casa (ad esempio, la metratura).

riospitare

Vedi 7 R.

rilascio

In un processo di implementazione, l'atto di promuovere modifiche a un ambiente di produzione.

trasferisco

Vedi 7 Rs.

ripiattaforma

Vedi 7 Rs.

riacquisto

Vedi 7 Rs.

resilienza

La capacità di un'applicazione di resistere o ripristinare le interruzioni. L'elevata disponibilità e il disaster recovery sono considerazioni comuni quando si pianifica la resilienza in. Cloud AWSPer ulteriori informazioni, vedere Cloud AWS Resilience.

policy basata su risorse

Una policy associata a una risorsa, ad esempio un bucket Amazon S3, un endpoint o una chiave di crittografia. Questo tipo di policy specifica a quali principali è consentito l'accesso, le azioni supportate e qualsiasi altra condizione che deve essere soddisfatta.

matrice di assegnazione di responsabilità (RACI)

Una matrice che definisce i ruoli e le responsabilità di tutte le parti coinvolte nelle attività di migrazione e nelle operazioni cloud. Il nome della matrice deriva dai tipi di responsabilità definiti nella matrice: responsabile (R), responsabile (A), consultato (C) e informato (I). Il tipo di supporto (S) è facoltativo. Se includi il supporto, la matrice viene chiamata matrice RASCI e, se la escludi, viene chiamata matrice RACI.

controllo reattivo

Un controllo di sicurezza progettato per favorire la correzione di eventi avversi o deviazioni dalla baseline di sicurezza. Per ulteriori informazioni, consulta Controlli reattivi in Implementazione dei controlli di sicurezza in AWS.

retain

Vedi 7 R.

andare in pensione

Vedi 7 Rs.

rotazione

Processo di aggiornamento periodico di un segreto per rendere più difficile l'accesso alle credenziali da parte di un utente malintenzionato.

controllo dell'accesso a righe e colonne (RCAC)

L'uso di espressioni SQL di base e flessibili con regole di accesso definite. RCAC è costituito da autorizzazioni di riga e maschere di colonna.

RPO

Vedi l'obiettivo del punto di ripristino.

RTO

Vedi l'obiettivo del tempo di ripristino.

runbook

Un insieme di procedure manuali o automatizzate necessarie per eseguire un'attività specifica. In genere sono progettati per semplificare operazioni o procedure ripetitive con tassi di errore elevati.

S

SAML 2.0

Uno standard aperto utilizzato da molti provider di identità (IdPs). Questa funzionalità abilita il single sign-on (SSO) federato, in modo che gli utenti possano accedere AWS Management Console o chiamare le operazioni AWS API senza che tu debba creare un utente in IAM per tutti i membri dell'organizzazione. Per ulteriori informazioni sulla federazione basata su SAML 2.0, consulta Informazioni sulla federazione basata su SAML 2.0 nella documentazione di IAM.

SCADA

Vedi controllo di supervisione e acquisizione dati.

SCP

Vedi la politica di controllo del servizio.

Secret

In AWS Secrets Manager, informazioni riservate o riservate, come una password o le credenziali utente, archiviate in forma crittografata. È costituito dal valore segreto e dai relativi metadati. Il valore segreto può essere binario, una stringa singola o più stringhe. Per ulteriori informazioni, consulta Cosa c'è in un segreto di Secrets Manager? nella documentazione di Secrets Manager.

controllo di sicurezza

Un guardrail tecnico o amministrativo che impedisce, rileva o riduce la capacità di un autore di minacce di sfruttare una vulnerabilità di sicurezza. Esistono quattro tipi principali di controlli di sicurezza: preventivi, investigativi, reattivi e proattivi.

rafforzamento della sicurezza

Il processo di riduzione della superficie di attacco per renderla più resistente agli attacchi. Può includere azioni come la rimozione di risorse che non sono più necessarie, l'implementazione di

best practice di sicurezza che prevedono la concessione del privilegio minimo o la disattivazione di funzionalità non necessarie nei file di configurazione.

sistema di gestione delle informazioni e degli eventi di sicurezza (SIEM)

Strumenti e servizi che combinano sistemi di gestione delle informazioni di sicurezza (SIM) e sistemi di gestione degli eventi di sicurezza (SEM). Un sistema SIEM raccoglie, monitora e analizza i dati da server, reti, dispositivi e altre fonti per rilevare minacce e violazioni della sicurezza e generare avvisi.

automazione della risposta alla sicurezza

Un'azione predefinita e programmata progettata per rispondere o porre rimedio automaticamente a un evento di sicurezza. Queste automazioni fungono da controlli di sicurezza investigativi o reattivi che aiutano a implementare le migliori pratiche di sicurezza. AWS Esempi di azioni di risposta automatizzate includono la modifica di un gruppo di sicurezza VPC, l'applicazione di patch a un'istanza Amazon EC2 o la rotazione delle credenziali.

Crittografia lato server

Crittografia dei dati a destinazione, da parte di chi li riceve. Servizio AWS

Policy di controllo dei servizi (SCP)

Una policy che fornisce il controllo centralizzato sulle autorizzazioni per tutti gli account di un'organizzazione in AWS Organizations. Le SCP definiscono i guardrail o fissano i limiti alle azioni che un amministratore può delegare a utenti o ruoli. Puoi utilizzare le SCP come elenchi consentiti o elenchi di rifiuto, per specificare quali servizi o azioni sono consentiti o proibiti. Per ulteriori informazioni, consulta le politiche di controllo del servizio nella AWS Organizations documentazione.

endpoint del servizio

L'URL del punto di ingresso per un Servizio AWS. Puoi utilizzare l'endpoint per connetterti a livello di programmazione al servizio di destinazione. Per ulteriori informazioni, consulta Endpoint del Servizio AWS nei Riferimenti generali di AWS.

accordo sul livello di servizio (SLA)

Un accordo che chiarisce ciò che un team IT promette di offrire ai propri clienti, ad esempio l'operatività e le prestazioni del servizio.

indicatore del livello di servizio (SLI)

Misurazione di un aspetto prestazionale di un servizio, ad esempio il tasso di errore, la disponibilità o la velocità effettiva.

obiettivo a livello di servizio (SLO)

Una metrica target che rappresenta lo stato di un servizio, misurato da un indicatore del livello di servizio.

Modello di responsabilità condivisa

Un modello che descrive la responsabilità condivisa AWS per la sicurezza e la conformità del cloud. AWS è responsabile della sicurezza del cloud, mentre tu sei responsabile della sicurezza nel cloud. Per ulteriori informazioni, consulta Modello di responsabilità condivisa.

SIEM

Vedi il sistema di gestione delle informazioni e degli eventi sulla sicurezza.

punto di errore singolo (SPOF)

Un guasto in un singolo componente critico di un'applicazione che può disturbare il sistema.

SLAM

Vedi il contratto sul livello di servizio.

SLI

Vedi l'indicatore del livello di servizio.

LENTA

Vedi obiettivo del livello di servizio.

split-and-seed modello

Un modello per dimensionare e accelerare i progetti di modernizzazione. Man mano che vengono definite nuove funzionalità e versioni dei prodotti, il team principale si divide per creare nuovi team di prodotto. Questo aiuta a dimensionare le capacità e i servizi dell'organizzazione, migliora la produttività degli sviluppatori e supporta una rapida innovazione. Per ulteriori informazioni, vedere Approccio graduale alla modernizzazione delle applicazioni in. Cloud AWS

SPOF

Vedi punto di errore singolo.

schema a stella

Una struttura organizzativa di database che utilizza un'unica tabella dei fatti di grandi dimensioni per archiviare i dati transazionali o misurati e utilizza una o più tabelle dimensionali più piccole per memorizzare gli attributi dei dati. Questa struttura è progettata per l'uso in un data warehouse o per scopi di business intelligence.

modello del fico strangolatore

Un approccio alla modernizzazione dei sistemi monolitici mediante la riscrittura e la sostituzione incrementali delle funzionalità del sistema fino alla disattivazione del sistema legacy. Questo modello utilizza l'analogia di una pianta di fico che cresce fino a diventare un albero robusto e alla fine annienta e sostituisce il suo ospite. Il modello è stato introdotto da Martin Fowler come metodo per gestire il rischio durante la riscrittura di sistemi monolitici. Per un esempio di come applicare questo modello, consulta Modernizzazione incrementale dei servizi Web legacy di Microsoft ASP.NET (ASMX) mediante container e Gateway Amazon API.

sottorete

Un intervallo di indirizzi IP nel VPC. Una sottorete deve risiedere in una singola zona di disponibilità.

controllo di supervisione e acquisizione dati (SCADA)

Nella produzione, un sistema che utilizza hardware e software per monitorare gli asset fisici e le operazioni di produzione.

crittografia simmetrica

Un algoritmo di crittografia che utilizza la stessa chiave per crittografare e decrittografare i dati.

test sintetici

Test di un sistema in modo da simulare le interazioni degli utenti per rilevare potenziali problemi o monitorare le prestazioni. Puoi usare Amazon CloudWatch Synthetics per creare questi test.

T

tags

Coppie chiave-valore che fungono da metadati per l'organizzazione delle risorse. AWS Con i tag è possibile a gestire, identificare, organizzare, cercare e filtrare le risorse. Per ulteriori informazioni, consulta Tagging delle risorse AWS.

variabile di destinazione

Il valore che stai cercando di prevedere nel machine learning supervisionato. Questo è indicato anche come variabile di risultato. Ad esempio, in un ambiente di produzione la variabile di destinazione potrebbe essere un difetto del prodotto.

elenco di attività

Uno strumento che viene utilizzato per tenere traccia dei progressi tramite un runbook. Un elenco di attività contiene una panoramica del runbook e un elenco di attività generali da completare. Per ogni attività generale, include la quantità stimata di tempo richiesta, il proprietario e lo stato di avanzamento.

Ambiente di test

Vedi ambiente.

training

Fornire dati da cui trarre ispirazione dal modello di machine learning. I dati di training devono contenere la risposta corretta. L'algoritmo di apprendimento trova nei dati di addestramento i pattern che mappano gli attributi dei dati di input al target (la risposta che si desidera prevedere). Produce un modello di ML che acquisisce questi modelli. Puoi quindi utilizzare il modello di ML per creare previsioni su nuovi dati di cui non si conosce il target.

Transit Gateway

Un hub di transito di rete che è possibile utilizzare per collegare i VPC e le reti on-premise. Per ulteriori informazioni, consulta Cos'è un gateway di transito nella AWS Transit Gateway documentazione.

flusso di lavoro basato su trunk

Un approccio in cui gli sviluppatori creano e testano le funzionalità localmente in un ramo di funzionalità e quindi uniscono tali modifiche al ramo principale. Il ramo principale viene quindi integrato negli ambienti di sviluppo, preproduzione e produzione, in sequenza.

Accesso attendibile

Concessione delle autorizzazioni a un servizio specificato dall'utente per eseguire attività all'interno dell'organizzazione AWS Organizations e nei suoi account per conto dell'utente. Il servizio attendibile crea un ruolo collegato al servizio in ogni account, quando tale ruolo è necessario, per eseguire attività di gestione per conto dell'utente. Per ulteriori informazioni,

consulta Utilizzo AWS Organizations con altri AWS servizi nella AWS Organizations documentazione.

regolazione

Modificare alcuni aspetti del processo di training per migliorare la precisione del modello di ML. Ad esempio, puoi addestrare il modello di ML generando un set di etichette, aggiungendo etichette e quindi ripetendo questi passaggi più volte con impostazioni diverse per ottimizzare il modello.

team da due pizze

Una piccola DevOps squadra che puoi sfamare con due pizze. Un team composto da due persone garantisce la migliore opportunità possibile di collaborazione nello sviluppo del software.

U

incertezza

Un concetto che si riferisce a informazioni imprecise, incomplete o sconosciute che possono minare l'affidabilità dei modelli di machine learning predittivi. Esistono due tipi di incertezza: l'incertezza epistemica, che è causata da dati limitati e incompleti, mentre l'incertezza aleatoria è causata dal rumore e dalla casualità insiti nei dati. Per ulteriori informazioni, consulta la guida Quantificazione dell'incertezza nei sistemi di deep learning.

compiti indifferenziati

Conosciuto anche come sollevamento di carichi pesanti, è un lavoro necessario per creare e far funzionare un'applicazione, ma che non apporta valore diretto all'utente finale né offre vantaggi competitivi. Esempi di attività indifferenziate includono l'approvvigionamento, la manutenzione e la pianificazione della capacità.

ambienti superiori

Vedi ambiente.

V

vacuum

Un'operazione di manutenzione del database che prevede la pulizia dopo aggiornamenti incrementali per recuperare lo spazio di archiviazione e migliorare le prestazioni.

controllo delle versioni

Processi e strumenti che tengono traccia delle modifiche, ad esempio le modifiche al codice di origine in un repository.

Peering VPC

Una connessione tra due VPC che consente di instradare il traffico tramite indirizzi IP privati. Per ulteriori informazioni, consulta Che cos'è il peering VPC? nella documentazione di Amazon VPC.

vulnerabilità

Un difetto software o hardware che compromette la sicurezza del sistema.

W

cache calda

Una cache del buffer che contiene dati correnti e pertinenti a cui si accede frequentemente. L'istanza di database può leggere dalla cache del buffer, il che richiede meno tempo rispetto alla lettura dalla memoria dal disco principale.

dati caldi

Dati a cui si accede raramente. Quando si eseguono interrogazioni di questo tipo di dati, in genere sono accettabili interrogazioni moderatamente lente.

funzione finestra

Una funzione SQL che esegue un calcolo su un gruppo di righe che si riferiscono in qualche modo al record corrente. Le funzioni della finestra sono utili per l'elaborazione di attività, come il calcolo di una media mobile o l'accesso al valore delle righe in base alla posizione relativa della riga corrente.

Carico di lavoro

Una raccolta di risorse e codice che fornisce valore aziendale, ad esempio un'applicazione rivolta ai clienti o un processo back-end.

flusso di lavoro

Gruppi funzionali in un progetto di migrazione responsabili di una serie specifica di attività. Ogni flusso di lavoro è indipendente ma supporta gli altri flussi di lavoro del progetto. Ad esempio,

il flusso di lavoro del portfolio è responsabile della definizione delle priorità delle applicazioni, della pianificazione delle ondate e della raccolta dei metadati di migrazione. Il flusso di lavoro del portfolio fornisce queste risorse al flusso di lavoro di migrazione, che quindi migra i server e le applicazioni.

VERME

Vedi scrivere una volta, leggere molti.

WQF

Vedi AWS Workload Qualification Framework.

scrivi una volta, leggi molte (WORM)

Un modello di storage che scrive i dati una sola volta e ne impedisce l'eliminazione o la modifica. Gli utenti autorizzati possono leggere i dati tutte le volte che è necessario, ma non possono modificarli. Questa infrastruttura di archiviazione dei dati è considerata immutabile.

Z

exploit zero-day

Un attacco, in genere malware, che sfrutta una vulnerabilità zero-day.

vulnerabilità zero-day

Un difetto o una vulnerabilità assoluta in un sistema di produzione. Gli autori delle minacce possono utilizzare questo tipo di vulnerabilità per attaccare il sistema. Gli sviluppatori vengono spesso a conoscenza della vulnerabilità causata dall'attacco.

applicazione zombie

Un'applicazione che prevede un utilizzo CPU e memoria inferiore al 5%. In un progetto di migrazione, è normale ritirare queste applicazioni.

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.