Scegliere un'infrastruttura come strumento di codice per la propria organizzazione

AWS Linee guida prescrittive

AWS Linee guida prescrittive: Scegliere un'infrastruttura come strumento di codice per la propria organizzazione

Table of Contents

Scegliere un'infrastruttura come strumento di codice per la propria organizzazione

Amazon Web Services (collaboratori)

Febbraio 2026 (cronologia del documento)

L'infrastruttura come codice (IaC) è il processo di provisioning e gestione dell'infrastruttura di un'applicazione tramite un set 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. È un componente chiave di Agile e delle DevOps pratiche, come il controllo delle versioni, l'integrazione continua e la distribuzione continua.

La scelta di uno strumento Infrastructure as Code (IaC) è considerata una decisione strategica per un'organizzazione. Questa decisione riguarda tutti i team che creano infrastrutture, applicazioni e servizi per l'azienda. Ogni strumento presenta vantaggi e svantaggi, pertanto non esiste un modello. one-size-fits-all

In passato, la gestione e il provisioning dell'infrastruttura erano un processo manuale pieno di errori. IaC semplifica queste attività tramite il codice ed è diventata una soluzione affidabile per l'implementazione dell'infrastruttura. Gli strumenti IaC consentono agli sviluppatori di definire e implementare l'infrastruttura utilizzando linguaggi di programmazione. Ciò non solo migliora l'agilità aziendale, ma accelera anche la crescita e la velocità dell'innovazione. Inoltre, IaC migliora significativamente la sicurezza perché IaC consente all'organizzazione di scansionare il codice prima dell'implementazione, verificando che l'infrastruttura sia affidabile e sicura. In definitiva, lo strumento IaC giusto non è solo una decisione tecnica ma strategica che ha un impatto diretto sul successo complessivo dell'azienda.

Questa guida esplora cinque diversi strumenti IaC che possono essere utilizzati per fornire AWS risorse: AWS CloudFormation, AWS Serverless Application Model (AWS SAM) AWS Cloud Development Kit (AWS CDK), HashiCorp Terraform e Pulumi. Confronta questi strumenti e ti guida nel processo di scelta di quello che soddisfi le esigenze del tuo team, della tua organizzazione e dei talenti del cloud. La chiave è allineare lo strumento IaC scelto agli obiettivi della vostra organizzazione e alle competenze dei vostri sviluppatori. Ad esempio, se il vostro team è competente JavaScript, potreste scegliere AWS CDK TypeScript come strumento IaC principale perché ottimizza il flusso di lavoro di sviluppo.

Vantaggi dell'implementazione di IaC

Scegliendo lo strumento IaC giusto per la tua organizzazione, puoi ottenere i seguenti vantaggi:

- Velocità: uno degli obiettivi di IaC è velocizzare le cose eliminando i processi manuali. Un approccio basato sul codice rende più facile fare di più in meno tempo. La configurazione manuale di ogni ambiente IT è molto costosa e richiede ingegneri e architetti dedicati per la configurazione dell'hardware e del software. Aumentando la velocità, le organizzazioni sono maggiormente in grado di adattarsi rapidamente alle mutevoli esigenze dei clienti e alle condizioni di mercato. Consente inoltre di scrivere il codice IaC una sola volta e di distribuirlo in centinaia di ambienti in una frazione del tempo necessario per crearli manualmente.
- Scalabilità: IAc semplifica l'aggiunta di risorse all'infrastruttura esistente. Gli upgrade vengono forniti rapidamente in modo da poterli espandere rapidamente durante i periodi di picco di utilizzo. Ad esempio, le organizzazioni che offrono servizi online possono espandersi per stare al passo con le richieste degli utenti durante i periodi di maggiore richiesta, come il Black Friday.
- Sicurezza avanzata: IaC eccelle nel garantire la sicurezza e la conformità per le organizzazioni. Le organizzazioni sono in grado di impostare policy e configurazioni di sicurezza di base e applicarle attraverso pipeline eseguendo test automatici sull'IaC. Se l'IaC viola le regole di conformità, la pipeline fallisce e fornisce automaticamente un feedback allo sviluppatore. Ciò può impedire la creazione di infrastrutture non sicure.
- Coerenza: la coerenza è un altro vantaggio fondamentale di IaC. Quando più ingegneri implementano manualmente le configurazioni, le incongruenze sono inevitabili. Nel tempo, diventa difficile tracciare e riprodurre gli stessi ambienti. Queste incongruenze spesso portano a differenze critiche tra gli ambienti di sviluppo, QA e produzione.
- Efficienza: IaC migliora l'efficienza e la produttività durante tutto il ciclo di vita dello sviluppo. I programmatori creano ambienti sandbox per svilupparsi in modo isolato. Le operazioni possono fornire rapidamente l'infrastruttura per i test di sicurezza. Gli ingegneri addetti al controllo qualità dispongono di copie esatte degli ambienti di produzione durante i test. Quando sono pronti per l'implementazione, gli sviluppatori mettono in produzione sia l'infrastruttura che il codice in un'unica operazione. Può anche consentire un livello più elevato di collaborazione e lavoro di squadra. I team possono creare modelli sicuri per le applicazioni e quindi condividere tali modelli o costrutti con altri team, il che riduce il lavoro duplicato.
- Costi ridotti: IaC riduce i costi di sviluppo del software. Non è necessario spendere risorse per configurare manualmente gli ambienti. Paghi solo per le risorse che utilizzi attivamente, quindi non ci sono costi generali inutili.

- Maggiore affidabilità: se l'infrastruttura è di grandi dimensioni, diventa facile configurare erroneamente una risorsa o fornire servizi nell'ordine sbagliato. Con IAc, le risorse vengono sempre fornite e configurate esattamente come dichiarato. Poiché la creazione manuale delle risorse è soggetta a errori, quando si utilizza IAc è possibile avere un elevato grado di fiducia nel fatto che l'infrastruttura verrà creata come previsto.
- Rilevamento delle deviazioni nella configurazione: la deriva della configurazione si verifica quando la configurazione che ha fornito l'ambiente non corrisponde più all'ambiente effettivo. Molti strumenti IAc consentono di rilevare la deriva e porvi rimedio. Ad esempio, se qualcuno modifica manualmente le risorse in modo errato, è possibile utilizzare lo strumento IaC per rilevare le modifiche e ripristinarle allo stato previsto.
- Sperimentazione: poiché IAc rende il provisioning di una nuova infrastruttura molto più rapido e semplice, è possibile apportare e testare modifiche sperimentali senza investire molto tempo e risorse. Se i risultati ti piacciono, puoi scalare rapidamente la nuova infrastruttura per la produzione.

Revisione degli strumenti IaC per Cloud AWS

Questa guida esplora cinque diversi strumenti IaC che possono essere utilizzati per fornire risorse. AWS Confronta questi strumenti e ti guida nel processo di scelta di quello che soddisfi le esigenze del tuo team, della tua organizzazione e dei talenti del cloud. La chiave è allineare lo strumento IaC scelto agli obiettivi della vostra organizzazione e alle competenze dei vostri sviluppatori.

In questa sezione, troverete ulteriori informazioni su come funziona ogni strumento e sui suoi vantaggi e svantaggi. Ad esempio, alcuni strumenti possono essere utilizzati solo in e non sono adatti per implementazioni multi-cloud o cloud ibrido. Cloud AWS Altri richiedono la conoscenza di linguaggi di programmazione specializzati, mentre altri supportano linguaggi di programmazione comuni. Valuta i vantaggi e gli svantaggi di ogni strumento e valuta quale sarebbe la soluzione migliore per la tua organizzazione.

In questa sezione vengono descritti i seguenti strumenti IAc:

- AWS CloudFormation
- AWS Serverless Application Model (AWS SAM)
- AWS Cloud Development Kit (AWS CDK)
- HashiCorp Terraform
- Pulumi

Utilizzo AWS CloudFormation come strumento IAc

AWS CloudFormationè uno strumento Servizio AWS che utilizza file modello per automatizzare l'approvvigionamento delle risorse. AWS Crei un modello che descrive tutte le AWS risorse che desideri distribuire e fornisce e CloudFormation configura tali risorse per te.

CloudFormation i modelli vengono scritti utilizzando JSON o YAML. Uno CloudFormation stack è l'implementazione delle risorse definite nel modello. Puoi gestire gli CloudFormation stack tramite AWS Management Console, a livello di codice tramite l' CloudFormation SDK o tramite (). AWS Command Line Interface AWS CLI Per ulteriori informazioni su come CloudFormation funziona, consulta AWS CloudFormation i concetti e Come AWS CloudFormation funziona nella documentazione. CloudFormation

Vantaggi dell'utilizzo CloudFormation:

- CloudFormation i set di modifiche consentono di visualizzare in anteprima le modifiche a uno stack in esecuzione prima di distribuirle. I set di modifiche riepilogano le modifiche proposte alle risorse in esecuzione in uno stack esistente. Questo può aiutarti a identificare conflitti o conseguenze indesiderate prima della distribuzione. Ad esempio, se cambi il nome di un'istanza di database Amazon Relational Database Service (Amazon RDS) CloudFormation , creerà un nuovo database ed eliminerà quello vecchio. Perderesti i dati nel vecchio database a meno che tu non ne abbia già eseguito il backup. Se generi un set di modifiche, noterai che la modifica comporterà la sostituzione del database e sarai in grado di pianificare di conseguenza prima di aggiornare lo stack.
- Se si verifica un errore durante la distribuzione di un set di modifiche CloudFormation , torna automaticamente all'ultimo stato di funzionamento noto.
- È possibile utilizzare i set di CloudFormation stack per distribuire risorse su più Account AWS e. Regioni AWS
- Non sono previsti costi aggiuntivi per l'utilizzo CloudFormation con provider di risorse nei seguenti namespace: :*, Alexa: :* e Custom: :*. In questi casi, paghi solo per le AWS risorse che fornisci, come se le avessi fornite manualmente.
- CloudFormation gestisce lo stato per te. Ciò significa che CloudFormation effettua chiamate di servizio sottostanti AWS per fornire e configurare le risorse come definito nei CloudFormation modelli.
- CloudFormation fornisce strumenti per rilevare e correggere le deviazioni di configurazione. Per ulteriori informazioni, consulta Rilevamento delle modifiche di configurazione non gestite agli stack e alle risorse nella documentazione. CloudFormation
- È possibile utilizzare CloudFormation per creare risorse personalizzate. È possibile scrivere una logica di provisioning personalizzata in modelli che vengono CloudFormation eseguiti ogni volta che si creano, aggiornano o eliminano pile.
- CloudFormation supporta la modellazione, il provisioning e la gestione di risorse applicative di terze parti con registro. CloudFormation
- CloudFormation supporta l'importazione di risorse esistenti nella gestione. CloudFormation

Svantaggi dell'utilizzo CloudFormation di:

- Se non hai familiarità con la sintassi JSON o YAML, potrebbe volerci un po' per abituarti. JSON non è stato progettato per essere leggibile dall'uomo e non consente di inserire commenti in linea. YAML consente di fare commenti ed è più facile da leggere. Tuttavia, la sua sintassi si basa su schede e spazi, quindi può essere facile commettere errori di indentazione.

- CloudFormation non supporta implementazioni multi-cloud.
- È necessario utilizzare un'implementazione di livello superiore, come la, per creare costrutti riutilizzabili e altro codice modularizzato. AWS Cloud Development Kit (AWS CDK)

Utilizzo di AWS Serverless Application Model (AWS SAM) come strumento IAc

Il AWS Serverless Application Model (AWS SAM) è un toolkit che si estende. AWS CloudFormation Include funzionalità aggiuntive progettate per aiutarti a creare applicazioni serverless più rapidamente. Quando si distribuisce un AWS SAM modello, questo viene convertito CloudFormation in modo da creare le risorse definite. AWS SAM è composto da due parti, la specifica del AWS SAM modello e l'interfaccia a riga di AWS SAM comando (AWS SAM CLI). Sebbene sia possibile utilizzare la CloudFormation sintassi direttamente nel AWS SAM modello, AWS SAM offre una sintassi unica che si concentra specificamente sull'accelerazione dello sviluppo senza server. Questa sintassi abbreviata consente definizioni ottimizzate di IAc per risorse serverless, come Amazon API Gateway, AWS Lambda e risorse. AWS Step Functions La AWS SAM CLI è uno strumento per sviluppatori che include funzionalità che consentono di testare AWS Lambda le funzioni localmente, creare pipeline di integrazione e distribuzione continua (CI/CD) ed eseguire comandi per distribuire applicazioni serverless.

Vantaggi dell'utilizzo di: AWS SAM

- AWS SAM presenta gli stessi vantaggi di CloudFormation.
- Rispetto a CloudFormation, puoi utilizzarlo più facilmente AWS SAM per creare applicazioni e risorse serverless, come un Amazon API Gateway supportato da AWS Lambda.
- Utilizzando la AWS SAM CLI, è possibile testare AWS Lambda le funzioni localmente. Quando richiami localmente una funzione Lambda in modalità debug, puoi collegare un debugger ad essa. Con il debugger, puoi scorrere il codice riga per riga, vedere i valori di varie variabili e risolvere i problemi nello stesso modo in cui faresti per qualsiasi altra applicazione.

Svantaggi dell'utilizzo: AWS SAM

- AWS SAM presenta gli stessi svantaggi di CloudFormation.
- AWS SAM non può essere utilizzato al di fuori di AWS.

Utilizzo di AWS CDK come strumento IaC

AWS Cloud Development Kit (AWS CDK)Si tratta di un framework di sviluppo software open source che consente di definire le risorse delle applicazioni cloud utilizzando linguaggi di programmazione familiari. I AWS CDK supporti sono Python JavaScript TypeScript, Java, C# e Go. AWS CDK Fornisce le tue risorse in modo sicuro e ripetibile. AWS CloudFormation Quando sintetizzi il AWS CDK codice, il risultato è un modello. CloudFormation AWS CDK Fornisce astrazioni di alto livello che semplificano il processo di definizione delle risorse. AWS

AWS CDK Utilizza il concetto di costrutti. Un costrutto è un componente all'interno dell'applicazione che rappresenta una o più CloudFormation risorse e la relativa configurazione, ad esempio un bucket Amazon Simple Storage Service (Amazon S3). I costrutti possono essere composti e personalizzati per creare un'infrastruttura più complessa. Per ulteriori informazioni, consultate Construct levels nella AWS CDK documentazione. AWS CDK Genera CloudFormation modelli basati sul codice scritto dagli sviluppatori. Ciò elimina la necessità di creare CloudFormation modelli manualmente. Molte organizzazioni personalizzano, condividono e riutilizzano i costrutti all'interno di una comunità, proprio come qualsiasi altra libreria software. I costrutti di condivisione aiutano gli sviluppatori a scrivere codice più velocemente e a incorporare le migliori pratiche per impostazione predefinita.

AWS CDK gli aspetti possono aiutare le organizzazioni ad applicare gli standard a tutti i costrutti all'interno di un determinato ambito. L'aspetto potrebbe modificare i costrutti, ad esempio aggiungendo tag. Oppure potrebbe verificare qualcosa sullo stato dei costrutti.

AWS CDK Consente agli sviluppatori di utilizzare le proprie competenze e conoscenze di programmazione esistenti per definire l'infrastruttura cloud. Utilizzando linguaggi di programmazione familiari, gli sviluppatori possono applicare la propria esperienza per descrivere AWS le risorse, facilitando la transizione dallo sviluppo di applicazioni al provisioning dell'infrastruttura. Inoltre AWS CDK possono accelerare la creazione dell'infrastruttura. AWS Ciò accelera il ciclo di vita dello sviluppo rispetto alla scrittura manuale dei modelli. CloudFormation

Vantaggi dell'utilizzo di: AWS CDK

- AWS CDK Supporta i linguaggi di programmazione più diffusi.
- I linguaggi generici consentono l'uso di costrutti logici, come for-loop, oggetti, tipi forti e altre tecniche di programmazione. Questo aiuta gli sviluppatori a dichiarare l'infrastruttura in modo conciso e privo di errori. Questo approccio consente inoltre di utilizzare un ambiente di sviluppo integrato (IDE) e gli strumenti correlati per aiutare a gestire la complessità legata alla dichiarazione di un gran numero di risorse.

- AWS CDK i costrutti sono condivisibili e aiutano a soddisfare i requisiti di governance e conformità.
- I AWS CDK costrutti possono ridurre i tempi e gli sforzi per lo sviluppo. Per ulteriori informazioni, consultate il riferimento all'API Construct Library.
- AWS CDK È basato su. CloudFormation Se conosci i CloudFormation suoi concetti, allora i AWS CDK concetti sono più facili da capire.
- Ti AWS CDK aiuta a eseguire test unitari e test istantanei.
- Se una funzionalità non è supportata nativamente in AWS CDK, è possibile utilizzare un costrutto di livello 1 e sostituzioni raw. In alternativa, è possibile utilizzare una risorsa CloudFormation personalizzata che richiama direttamente l'API.
- È possibile ripulire le risorse in modo efficiente eliminando gli CloudFormation stack.

Svantaggi dell'utilizzo di: AWS CDK

- AWS CDK Richiede un ambiente bootstrap in ciascuno. Account AWS Il bootstrap è un'azione da eseguire una sola volta per ogni ambiente in cui vengono distribuite le risorse.
- AWS CDK Può essere utilizzato per distribuire IAc solo in. Cloud AWS

Utilizzo di Terraform come strumento IaC per Cloud AWS

HashiCorp Terraform è uno strumento di infrastruttura come codice (IaC) che ti aiuta a gestire la tua infrastruttura cloud. Utilizzando Terraform, puoi definire risorse cloud e locali in file di configurazione che puoi modificare, riutilizzare e condividere. È quindi possibile utilizzare un flusso di lavoro coerente per fornire e gestire tutta l'infrastruttura durante il suo ciclo di vita.

Gli sviluppatori utilizzano un linguaggio di configurazione di alto livello chiamato linguaggio Terraform. La sintassi nativa di basso livello del linguaggio Terraform è HashiCorpConfiguration Language (HCL). Il linguaggio Terraform è progettato per essere facile da leggere e scrivere per gli umani. Utilizzi il linguaggio Terraform per descrivere lo stato finale desiderato del cloud o dell'infrastruttura locale. Terraform genera quindi un piano per raggiungere quello stato finale e tu esegui il piano per il provisioning dell'infrastruttura.

Vantaggi dell'utilizzo di Terraform:

- Terraform è indipendente dalla piattaforma. Puoi usarlo con qualsiasi provider di servizi cloud. Puoi configurare, testare e implementare l'infrastruttura tra AWS molti altri provider di cloud. Se la tua organizzazione utilizza più provider cloud, Terraform può essere un'unica soluzione unificata

e coerente per gestire l'infrastruttura cloud. Per ulteriori informazioni sul supporto multi-cloud, consulta il provisioning multi-cloud sul sito Web Terraform.

- Terraform è senza agenti. Non richiede l'installazione di alcun software sull'infrastruttura gestita.
- I moduli Terraform sono un modo efficace per riutilizzare il codice e attenersi al principio Don't Repeat Yourself (DRY). Ad esempio, potresti avere una configurazione specifica per un'applicazione che contiene un'istanza Amazon Elastic Compute Cloud (Amazon EC2), volumi Amazon Elastic Block Store (Amazon EBS) e altre risorse raggruppate logicamente. Se devi creare più copie di questa configurazione o applicazione, puoi impacchettare le risorse in un modulo Terraform e creare più istanze del modulo anziché copiare l'intero codice più volte. Questi moduli possono aiutarti a organizzare, incapsulare e riutilizzare le configurazioni. Garantiscono inoltre coerenza e garantiscono le migliori pratiche.
- Terraform è in grado di rilevare e gestire la deriva (post sul blog Terraform) nella tua infrastruttura. Ad esempio, se le risorse gestite da Terraform vengono modificate all'esterno di Terraform, è possibile rilevare la deriva e ripristinarle allo stato desiderato utilizzando la CLI Terraform.

Svantaggi dell'utilizzo di Terraform:

- Il supporto per nuove funzionalità o nuove risorse relative a qualsiasi provider di servizi cloud potrebbe non essere disponibile.
- Terraform non gestisce automaticamente il tuo stato come AWS CloudFormation. È archiviato per impostazione predefinita in un file locale, ma puoi anche archiviarlo in remoto in un bucket Amazon S3 o tramite Terraform Enterprise.
- Lo stato Terraform può contenere dati sensibili, come le password dei database, che possono comportare problemi di sicurezza. È consigliabile crittografare il file di stato, archiviarlo in remoto, abilitare il controllo delle versioni dei file su di esso e utilizzare i privilegi minimi per le operazioni di lettura e scrittura su di esso. Per ulteriori informazioni, consulta Proteggere i dati sensibili utilizzando e Terraform. AWS Secrets Manager HashiCorp
- Nell'agosto 2023, Hashicorp ha annunciato che non sarebbe più stato concesso in licenza come open source ai sensi della Mozilla Public License. Invece, ora è concesso in licenza con la Business Source License.

Utilizzo di Pulumi come strumento IaC per Cloud AWS

Pulumi è un'infrastruttura open source come strumento di codice. Supporta linguaggi di programmazione comuni esistenti, come Python TypeScript JavaScript, Go, .NET, Java e YAML.

Utilizza inoltre il suo ecosistema nativo per interagire con le risorse cloud tramite Pulumi SDK. Una CLI scaricabile, un runtime, librerie e un servizio ospitato collaborano per fornire, aggiornare e gestire l'infrastruttura cloud.

Puoi utilizzare Pulumi SDK per creare e distribuire software cloud che utilizza contenitori, funzioni serverless, servizi ospitati e infrastrutture, su qualsiasi cloud.

Opzionalmente puoi accoppiare Pulumi con Pulumi Cloud. Pulumi Cloud è un servizio gestito che memorizza lo stato e i segreti e gestisce le implementazioni della tua infrastruttura Pulumi.

Vantaggi dell'utilizzo di Pulumi:

- Pulumi fornisce l'infrastruttura di oltre cinquanta provider di cloud e software as a service (SaaS).
- Offre un'interfaccia completa e coerente progettata per ridurre la complessità del cloud.

Svantaggi dell'utilizzo di Pulumi:

- Sebbene Pulumi abbia una comunità attiva che offre supporto, è più piccola della comunità Terraform.
- Pulumi ha una curva di apprendimento più ripida.

Scelta di uno strumento IaC

Quindi, quale strumento scegliere?

Con così tante opzioni di strumenti e requisiti aziendali diversi, non esiste un one-size-fits-all approccio. Oltre ai vantaggi e agli svantaggi di ogni strumento illustrato in questa guida, prendete in considerazione i seguenti consigli per i vostri requisiti aziendali e il vostro modello operativo:

- Se gestite o implementate una AWS soluzione serverless con una o più dipendenti minime, AWS Serverless Application Model (AWS SAM) potrebbe essere una buona opzione per voi. Ha tutte le stesse funzionalità di. AWS CloudFormation Inoltre, semplifica il test e la distribuzione di applicazioni serverless su. Cloud AWS
- Se gestisci la tua infrastruttura interamente attiva AWS, allora queste AWS CloudFormation AWS Cloud Development Kit (AWS CDK) sono buone opzioni. Forniscono la gestione out-of-the-box dello stato e puoi anche utilizzare in modo nativo nuove funzionalità o AWS risorse.
- Se desideri un'utilità multiprovider, in particolare per la gestione di infrastrutture multi-cloud o cloud ibrido, Terraform potrebbe essere una buona scelta perché è indipendente dalla piattaforma. Con Terraform, puoi anche utilizzare un'ampia gamma di plugin e dispone di una vasta community con opzioni di supporto aziendale.
- Se disponi di una distribuzione dall'alto verso il basso con le migliori pratiche e se disponi di un'orchestrazione in cui crei, pubblichi e distribuisci moduli riutilizzabili utilizzando linguaggi di programmazione comuni, allora potrebbe essere una buona opzione. AWS CDK
- Se la tua organizzazione è in grado di tollerare un elevato livello di rischio e deve supportare ambienti multi-cloud o cloud ibrido, prendi in considerazione l'utilizzo di Pulumi.

Passaggi successivi

Questa guida ha illustrato i vantaggi e gli svantaggi di diversi strumenti Infrastructure as Code (IaC) che è possibile utilizzare per creare, distribuire e gestire AWS risorse e infrastrutture. In base allo strumento scelto, ti consigliamo di visitare le seguenti risorse per iniziare.

AWS Cloud Development Kit (AWS CDK)

- AWS CDK seminario introduttivo

AWS CloudFormation

- AWS CloudFormation laboratorio
- Workshop AWS CloudFormation

HashiCorp Terraform

- AWS Officina Terraform
- CDK per il repository Terraform () GitHub

Pulumi

- Archivio Pulumi () GitHub
- AWS modernizzazione con officina Pulumi

Risorse

AWS documentazione

- Le migliori pratiche per l'utilizzo di AWS CDK in per TypeScript creare progetti IaC (AWS Prescriptive Guidance)
- Controlla AWS CDK le applicazioni o i CloudFormation modelli per le migliori pratiche utilizzando i pacchetti di regole cdk-nag (Prescriptive Guidance)AWS
- Introduzione a DevOps on AWS: Infrastructure as code (white paper)AWS
- Documentazione di AWS CloudFormation

Altra documentazione

- Guida introduttiva all'uso di Infrastructure as Code (HashiCorpsito Web)
- HashiCorpDocumentazione Terraform
- Documentazione Pulumi
- Pulumi: Cos'è l'infrastruttura come codice (sito web Pulumi)

Strumenti

- cdk-nag () GitHub
- cfn-bag () GitHub
- Terratest

Collaboratori

Creazione di testi

- Siamak Heshmati, architetto senior, DevOps AWS
- Mason Cahill, consulente senior, DevOps AWS
- Sandip Gangapadhyay, architetto senior, DevOps AWS
- Sandeep Gawande, consulente capo senior, AWS
- Rajneesh Tyagi, consulente principale, AWS

Revisione

- David Howerton, Senior Engagement Manager, AWS
- Steven Guggenheimer, architetto senior di applicazioni cloud, AWS

Scrittura tecnica

- Lilly AbouHarb, scrittrice tecnica senior, AWS

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
Aggiornamenti Pulumi	Abbiamo aggiornato il capitolo Pulumi.	17 febbraio 2026
Pubblicazione iniziale	—	23 febbraio 2024

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.