Whitepaper AWS

Integrazione continua e distribuzione continua per le reti 5G in AWS

Integrazione continua e distribuzione continua per le reti 5G in AWS: Whitepaper AWS

Table of Contents

Integrazione continua e distribuzione continua per le reti 5G in AWS

Data di pubblicazione: 8 marzo 2021 (Revisioni del documento)

Riassunto

Questo whitepaper illustra l'integrazione continua e la distribuzione continua (CI/CD) per le reti 5G e gli strumenti e i servizi di Amazon Web Services (AWS) che possono essere utilizzati per automatizzare completamente l'implementazione e gli aggiornamenti delle funzioni di rete 5G. Il whitepaper fornisce una descrizione dettagliata delle diverse fasi di CI/CD per le funzioni di rete 5G, tra cui la configurazione della rete, l'implementazione dell'infrastruttura, l'implementazione delle funzioni di rete native per il cloud e gli aggiornamenti continui delle funzioni di rete. Fornisce inoltre dettagli sull'integrazione con strumenti open source e di terze parti per test, osservabilità e orchestrazione.

Questo whitepaper è rivolto ai fornitori di servizi di comunicazione (CSP) e ai fornitori di software indipendente (ISV).

Introduzione

Storicamente, lo sviluppo, i test di integrazione in laboratorio e sul campo e l'implementazione in produzione di nuovi nodi di rete o nuove caratteristiche in una rete cellulare richiede settimane o addirittura mesi per garantire la stabilità dei servizi di telecomunicazione (telecom) mission e business critical. Il lungo ciclo di implementazione è stato causato dall'architettura monolitica dei nodi di rete tradizionali, da un ambiente con più fornitori e da molte interfacce point-to-point tra entità nelle reti mobili 2G, 3G e 4G.

Come introdotto nel whitepaper Evoluzione della rete 5G con AWS, le reti mobili 5G, come standardizzate da 3GPP, ora supportano un'architettura nativa per il cloud abilitata dalla virtualizzazione e dalla containerizzazione. Più specificamente, le reti 5G introducono e supportano un nuovo paradigma dell'architettura a microservizi, senza stato e basata sui servizi.

Questa architettura 5G significa che diverse funzioni di rete possono funzionare come servizi indipendenti ad accoppiamento debole che comunicano tra loro attraverso interfacce e API ben definite. Ancora più importante, ogni funzione di rete può essere aggiornata in modo indipendente. Questo livello di architettura 5G consente ai CSP di ottenere maggiore agilità ed efficienza operativa semplificando l'implementazione più frequente degli aggiornamenti delle funzioni di rete, mantenendo al contempo i test, i requisiti di sicurezza e gli standard tramite l'automazione.

L'integrazione e l'implementazione di nuove caratteristiche per un CSP generalmente iniziano quando il fornitore della funzione di rete rilascia un nuovo pacchetto software per le funzioni di rete, ad esempio un'immagine Docker in una funzione di rete basata su container, o un nuovo file di configurazione, ad esempio un grafico Helm nel caso dell'applicazione Kubernetes. Un grafico Helm è una raccolta di file che descrivono un insieme correlato di risorse Kubernetes.

L'idea di utilizzare il paradigma di CI/CD per l'implementazione delle funzioni di rete 5G sta guadagnando terreno, ma la realizzazione pratica di questa idea è stata una sfida nel settore delle telecomunicazioni.

AWS ha aperto la strada allo sviluppo di nuovi strumenti CI/CD per la distribuzione del software per aiutare un ampio spettro di settori a sviluppare e implementare rapidamente le modifiche software, mantenendo al contempo la stabilità e la sicurezza dei sistemi. Questi strumenti includono una serie di servizi DevOps (Software Development and Operations) come AWS CodeStar, CodeCommit, CodePipeline, CodeBuild e CodeDeploy.

AWS promuove anche l'idea di Infrastructure as Code (IaC) utilizzando AWS Cloud Development Kit (AWS CDK), AWS CloudFormation e strumenti di terze parti basati su API come Terraform. Con

questi strumenti, AWS può memorizzare i processi di implementazione della funzione di rete in AWS come codice sorgente e mantenere questo codice sorgente IaC nella pipeline CI/CD per realizzare una distribuzione continua.

Questo whitepaper descrive i processi dettagliati per sfruttare gli strumenti IaC e CI/CD di AWS per l'implementazione e l'aggiornamento della funzione di rete 5G. Inoltre, illustra l'integrazione con strumenti di terze parti per il test, l'osservabilità e l'orchestrazione.

Gli strumenti CI/CD di AWS non sono limitati alle funzioni di rete 5G. Sono anche impiegati per automatizzare l'implementazione di reti 4G, che consente ai CSP di implementare e aggiornare in modo rapido ed efficiente le funzioni di rete 4G. La maggior parte delle funzioni di rete 4G è basata su Virtual Network Function (VNF). I set di strumenti CI/CD di AWS come AWS CloudFormation possono essere utilizzati per automatizzare l'implementazione di VNF 4G, offrendo scalabilità ed efficienza in termini di tempo per le implementazioni di rete 4G.

Integrazione continua e distribuzione continua

Integrazione continua

L'integrazione continua (CI) è un processo software in cui gli sviluppatori inviano regolarmente il loro codice in un repository centrale come AWS CodeCommit o GitHub. Ogni push di codice attiva la compilazione automatica, seguita dall'esecuzione di test. L'obiettivo principale della CI è scoprire i problemi del codice in fase iniziale, migliorare la qualità del codice e ridurre il tempo necessario per convalidare e rilasciare nuovi aggiornamenti software.

Distribuzione continua e implementazione

La distribuzione continua (CD) è un processo software in cui gli artefatti vengono implementati nell'ambiente di test, nell'ambiente di gestione temporanea e nell'ambiente di produzione. La distribuzione continua può essere completamente automatizzata o includere fasi di approvazione nei punti critici. Ciò garantisce che prima dell'implementazione vengano eseguite tutte le approvazioni necessarie, come l'approvazione della gestione del rilascio. Quando la distribuzione continua viene implementata correttamente, gli sviluppatori dispongono sempre di un artefatto di compilazione pronto per la distribuzione che ha già passato un processo di test standardizzato.

Con la distribuzione continua, le modifiche vengono implementate automaticamente nell'ambiente di produzione senza l'approvazione esplicita da parte dello sviluppatore, automatizzando di fatto l'intero processo di rilascio del software. Questo abilita un circuito di feedback dei clienti continuo nelle prime fasi del ciclo di vita del prodotto.

Con la distribuzione continua, ogni modifica che viene sottoposta al commit e supera i test automatici, viene rilasciata automaticamente in produzione. La distribuzione continua non ha lo scopo di rilasciare tutte le modifiche che vengono sottoposte al commit e passare immediatamente i test automatici in produzione, ma di garantire che ogni modifica sia pronta per essere inserita nell'ambiente in produzione.

Infrastructure as Code

Come descritto nel whitepaper Evoluzione della rete 5G con AWS, il framework IaC è un fattore chiave per automatizzare il processo di provisioning e la gestione del ciclo di vita sia per

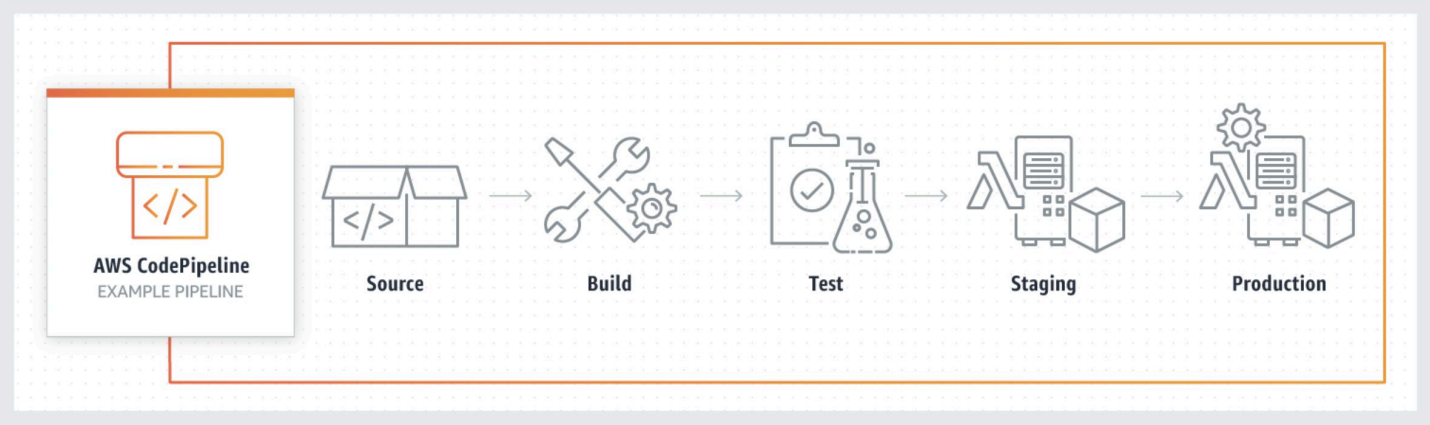
l'applicazione che per l'ambiente. Anziché eseguire i passaggi manualmente, gli amministratori di rete/IT e gli sviluppatori possono creare un'istanza dell'infrastruttura utilizzando i file di configurazione. Il framework IaC tratta questi file di configurazione come codice software. Questi file possono essere utilizzati per produrre un set di artefatti: vale a dire i servizi di calcolo, archiviazione, rete e applicazioni che costituiscono un ambiente operativo. Il framework IaC elimina la deriva della configurazione con l'automazione, aumentando così la velocità e l'agilità delle implementazioni dell'infrastruttura.

Nel caso di implementazioni NFV (Network Function Virtualization, virtualizzazione delle funzioni di rete) in AWS, questo framework IaC apporta valore dal punto di vista dell'orchestrazione. Dalla creazione del Virtual Private Cloud (VPC) all'implementazione delle funzioni di rete, ogni passaggio può essere programmato, gestito come codice sorgente e mantenuto con il controllo della versione in AWS CodeCommit.

Questo framework IaC per le funzioni di rete si traduce nella creazione e nell'implementazione di un'infrastruttura ripetibile e affidabile e di funzioni di rete, che possono essere estese all'automazione end-to-end (E2E) della gestione del network slicing e della gestione del ciclo di vita del servizio. AWS fornisce un set di strumenti completo per la creazione, la manutenzione e l'implementazione dell'infrastruttura in modo programmatico, descrittivo e dichiarativo, utilizzando servizi come AWS CloudFormation, AWS CDK, AWS CDK per Kubernetes e l'esposizione API di tutti i servizi AWS.

CI/CD in AWS

L'approccio CI/CD può essere rappresentato come una pipeline in cui il nuovo codice viene inviato a un'estremità, testato durante una serie di fasi (origine, compilazione, gestione temporanea e produzione) e quindi pubblicato come codice pronto per la produzione.



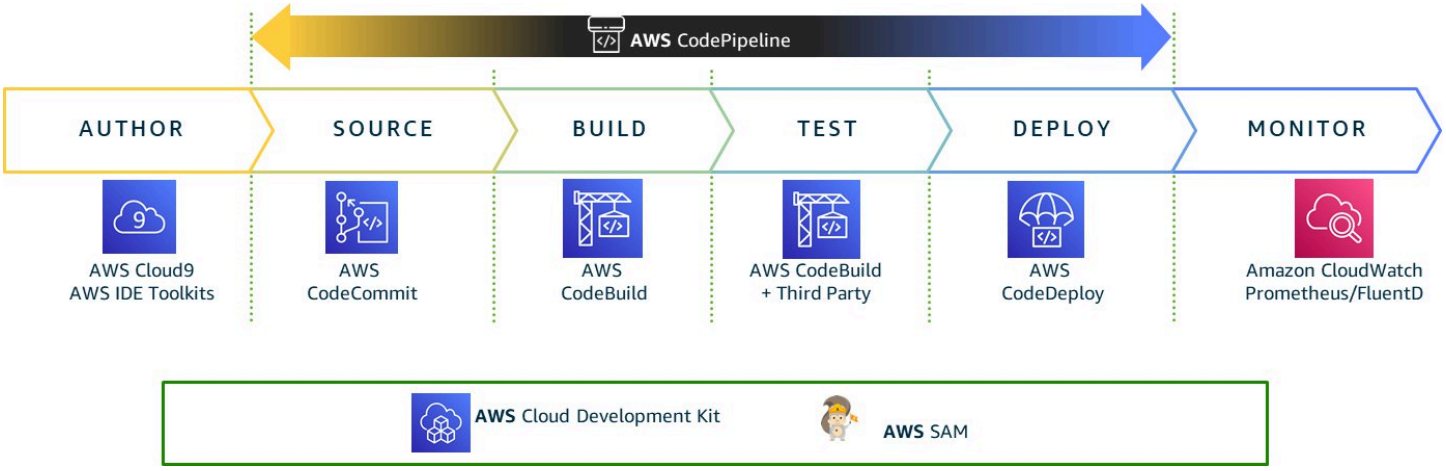
Panoramica della pipeline CI/CD

Ogni fase della pipeline CI/CD è strutturata come un'unità logica nel processo di consegna. Ogni fase agisce da gate che controlla un determinato aspetto del codice. Durante l'avanzamento nella pipeline, si presume che la qualità del codice sia più alta nelle fasi successive perché ne vengono verificati sempre più aspetti. I problemi rilevati in una fase iniziale impediscono al codice di progredire nella pipeline. I risultati dei test vengono immediatamente inviati al team e le build e i rilasci successivi vengono interrotti se il software non supera la fase.

AWS offre un set completo di strumenti di sviluppo CI/CD per accelerare lo sviluppo del software e i cicli di rilascio. AWS CodePipeline automatizza le fasi di compilazione, test e implementazione del processo di rilascio ogni volta che si verifica una modifica del codice, in base al modello di rilascio definito. Ciò consente la consegna rapida e affidabile di caratteristiche e aggiornamenti.

Le pipeline di codice possono integrarsi con altri servizi. Questi possono essere servizi AWS, come Amazon Simple Storage Service (Amazon S3) oppure prodotti di terze parti, come GitHub. AWS CodePipeline è in grado di risolvere una serie di casi d'uso relativi allo sviluppo e alle operazioni, tra cui:

- Creazione, compilazione e test del codice con AWS CodeBuild
- Distribuzione continua di applicazioni basate su container nel cloud.
- Convalida preliminare dell'implementazione di artefatti (come descrittori e immagini di container) richiesti per il servizio di rete o specifiche funzioni di rete native per il cloud
- Test funzionali, di integrazione e delle prestazioni per la funzione di rete containerizzata/funzione di rete virtuale (CNF/VNF), inclusi i test di base e di regressione
- Test di affidabilità e ripristino di emergenza (DR).



Componenti della pipeline CI/CD AWS

AWS può configurare la pipeline CI/CD utilizzando i seguenti strumenti di sviluppo AWS:

- AWS CodeCommit
- AWS CodeBuild
- AWS CodePipeline
- AWS CodeDeploy
- Amazon Elastic Container Registry
- AWS CodeStar

La creazione della pipeline CI/CD può essere automatizzata utilizzando AWS CDK e AWS CloudFormation. Nel dominio NFV, questa automazione nativa di AWS può essere integrata in un framework di gestione e orchestrazione (MANO) e nel framework di orchestrazione dei servizi del CSP.

Il processo CI/CD include le seguenti fasi:

- Configurazione della rete: AWS CDK e AWS CloudFormation avviano la creazione dei prerequisiti di rete:
    - Pila di rete (VPC, sottoreti, gateway NAT (Network Address Translation), tabella di routing e gateway Internet)
- Implementazione dell'infrastruttura: AWS CDK e AWS CloudFormation avviano la creazione delle seguenti pile di risorse:
    - Pila di calcolo (creazione di cluster Amazon Elastic Kubernetes Service (Amazon EKS), nodi (worker) EKS, AWS Lambda)
    - Pila di archiviazione (bucket Amazon S3, volumi Amazon Elastic Block Store (Amazon EBS) e Amazon Elastic File System (Amazon EFS)
    - Pila di monitoraggio (CloudWatch, Amazon OpenSearch Service (OpenSearch Service)
    - Pila di sicurezza AWS Identity and Access Management (AWS IAM), gruppi di sicurezza Amazon Elastic Compute Cloud (Amazon EC2), liste di controllo accessi alla rete VPC (NACL)

- Implementazione della funzione di rete per il cloud (CNF): in questa fase, la CNF viene implementata nei cluster EKS utilizzando gli strumenti dei grafici Kubectl e Helm. Questa fase implementa anche tutte le applicazioni o gli strumenti specifici necessari alle CNF per lavorare in modo efficiente (comePrometheus o FluentD). Le CNF possono essere implementate tramite funzioni Lambda o con AWS CodeBuild.
- Aggiornamenti continui e implementazione: si tratta di una sequenza di passaggi che vengono eseguiti in modo iterativo per implementare le modifiche di container/configurazione che comportano aggiornamenti. Analogamente all'implementazione della CNF, gli aggiornamenti continui e l'implementazione possono essere automatizzati utilizzando i servizi AWS, con l'attivazione di AWS CodeCommit, Amazon Elastic Container Registry (Amazon ECR) o un sistema di origine di terze parti come GitLab Webhook.

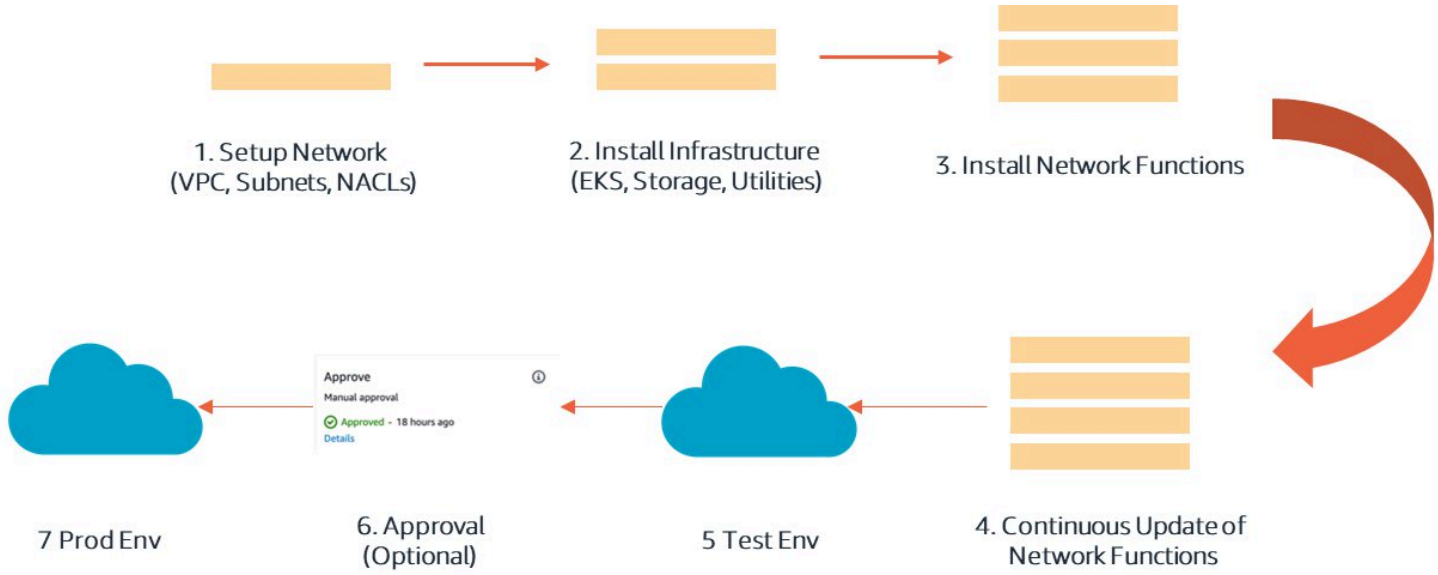


Diagramma di flusso della pipeline CI/CD AWS

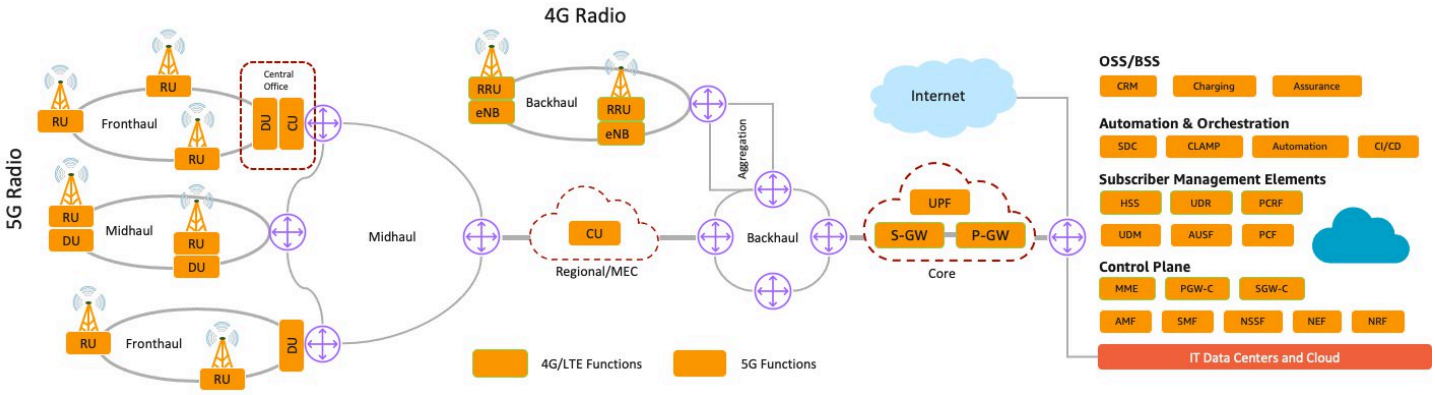
La pipeline CI/CD è creata con AWS CodePipeline e utilizza un servizio di distribuzione continua che modella, visualizza e automatizza i passaggi necessari per il rilascio del software. Definendo le fasi in una pipeline, è possibile recuperare il codice da un repository di codice sorgente, compilare il codice sorgente in un artefatto rilasciabile, testare l'artefatto e implementarlo in produzione. Viene implementato solo il codice che supera tutte queste fasi. Facoltativamente, puoi aggiungere altri requisiti alla pipeline, come le approvazioni manuali, per garantire che solo le modifiche approvate vengano implementate in produzione.

Reti 5G in AWS

Il modello tipico dell'infrastruttura di rete 5G è composto da un sito radio 4G/5G, una rete fronthaul/midhaul/backhaul, un sito di rete principale e un data center per le telecomunicazioni/IT. I CSP possono utilizzare i servizi AWS per creare un'infrastruttura di rete 5G scalabile e flessibile, contenendo al contempo i costi di investimento iniziali. AWS può essere utilizzato per implementare il centro operatività di rete (NOC) virtuale nella regione che ospita il sistema di supporto operativo/sistema di supporto aziendale (OSS/BSS) e la maggior parte delle funzioni di rete core del piano di controllo.

AWS può anche essere sfruttato per implementare l'ufficio centrale locale o il data center distribuito con un parco istanze AWS Outposts che ospitano principalmente funzioni di piano per l'utente come la funzione UPF, RAN Central Unit (CU) e Multi-Access Edge Computing (MEC). Una spiegazione più dettagliata dell'architettura di riferimento e dei vantaggi dell'implementazione della rete 5G in AWS è spiegata nel whitepaper Evoluzione della rete 5G in AWS.

Quando è il momento di implementare la rete 5G in AWS, gli strumenti CI/CD AWS, introdotti nelle sezioni seguenti di questo whitepaper, possono facilitare la completa automazione dell'implementazione, dell'aggiornamento e della gestione del ciclo di vita delle funzioni di rete 5G.

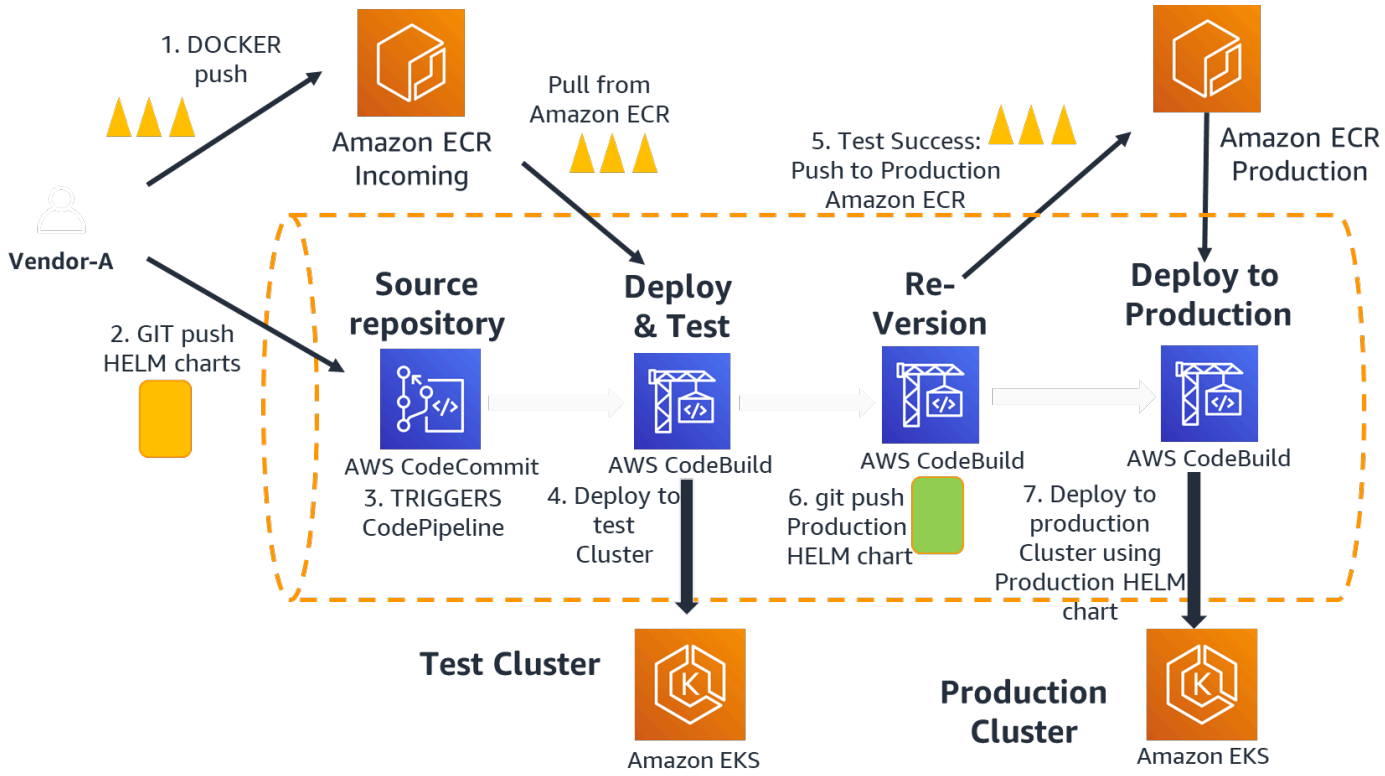


Architettura E2E della rete 5G

CI/CD nelle reti 5G

Il costrutto di progettazione dell'infrastruttura viene archiviato sotto forma di codice utilizzando un linguaggio dichiarativo. Ciò consente al CSP di avere una riproduzione ripetibile dell'infrastruttura con lo stesso comportamento previsto in base alle esigenze. Il codice viene mantenuto nel repository di codice e viene configurata una pipeline per orchestrare gli aggiornamenti alle pile implementate (ad

esempio, AWS CDK e AWS CloudFormation). AWS può aiutare a creare il framework Infrastructure as Code (IaC) per l'inserimento agile delle funzioni di fornitore di software indipendente (ISV).



Flusso della pipeline di codice

Le modifiche nelle configurazioni delle funzioni di rete native per il cloud tramite i grafici Helm sono considerate trigger per l'esecuzione automatica di pipeline CI/CD per le funzioni di rete.

AWS CodeCommit può essere utilizzato per gestire i file di configurazione e Amazon ECR per conservare le immagini dei container.

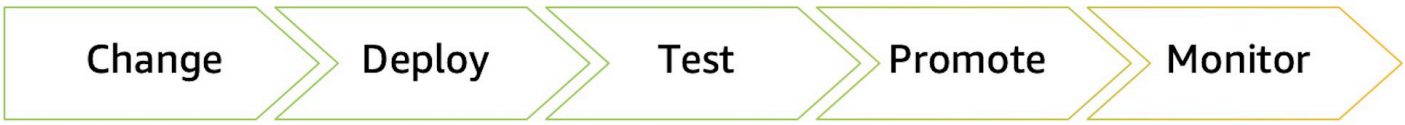
Come mostrato nella figura del flusso della pipeline del codice, quando l'ISV invia nuove modifiche del codice nel repository di codice (grafico Helm, file di configurazione o file delle proprietà), viene attivata la pipeline di codice. La pipeline di codice estrae l'immagine da ECR e utilizza il grafico Helm per implementare l'applicazione. Il nuovo test delle applicazioni può essere integrato con il framework di automazione dei test di terze parti. In base al risultato, i CSP possono approvare l'implementazione in produzione.

La fase di origine di CodePipeline cerca le modifiche nei file di configurazione. I provider validi per la fase di origine sono CodeCommit, Amazon S3, GitHub o AWS CloudFormation. I sistemi di origine alternativi possono essere integrati utilizzando le funzioni Lambda per implementare i webhook,

consentendo l"integrazione guidata dagli eventi tra Gitlab e AWS CodePipeline. Consulta i seguenti collegamenti per una guida dettagliata dell'implementazione.

- Webhook con GitLab
- Integrazioni nel registro del container

La progettazione della pipeline CI/CD deve tenere conto delle fasi di implementazione critiche come l'implementazione iniziale, il test e la promozione in produzione dopo che i risultati dei test sono stati allineati alle aspettative e verificati rispetto alla linea di base. Ogni fase del processo della pipeline fornisce artefatti di dati che abilitano il confronto e le decisioni orientate ai dati.



Fasi della pipeline CI/CD dell'applicazione

Ogni fase può essere considerata un'attività separata, consentendo l'incorporazione di flussi di lavoro di convalida e implementazione adeguati a supportare il servizio di rete e le funzioni di rete native per il cloud. Le attività di esecuzione possono incorporare strumenti aggiuntivi di terze parti, come generatori e simulatori di traffico, consentendo la convalida dei servizi di rete end-to-end.

AWS fornisce il sofisticato servizio AWS Step Functions (macchina a stati nativa per il cloud) che si integra in modo nativo con altri servizi AWS e con sistemi esterni come Jira o un framework di automazione dei test.

Passaggi dettagliati per CI/CD

L'approccio CI/CD può essere rappresentato come una pipeline in cui il nuovo codice viene inviato a un'estremità, testato durante una serie di fasi (origine, compilazione, gestione temporanea e produzione) e quindi pubblicato come codice pronto per la produzione.

Di seguito sono riportati i passaggi di implementazione e test. L'implementazione e la configurazione sono suddivise principalmente in quattro sezioni principali:

- Configurazione della rete
- Implementazione dell'infrastruttura
- Implementazione di funzioni di rete native per il cloud

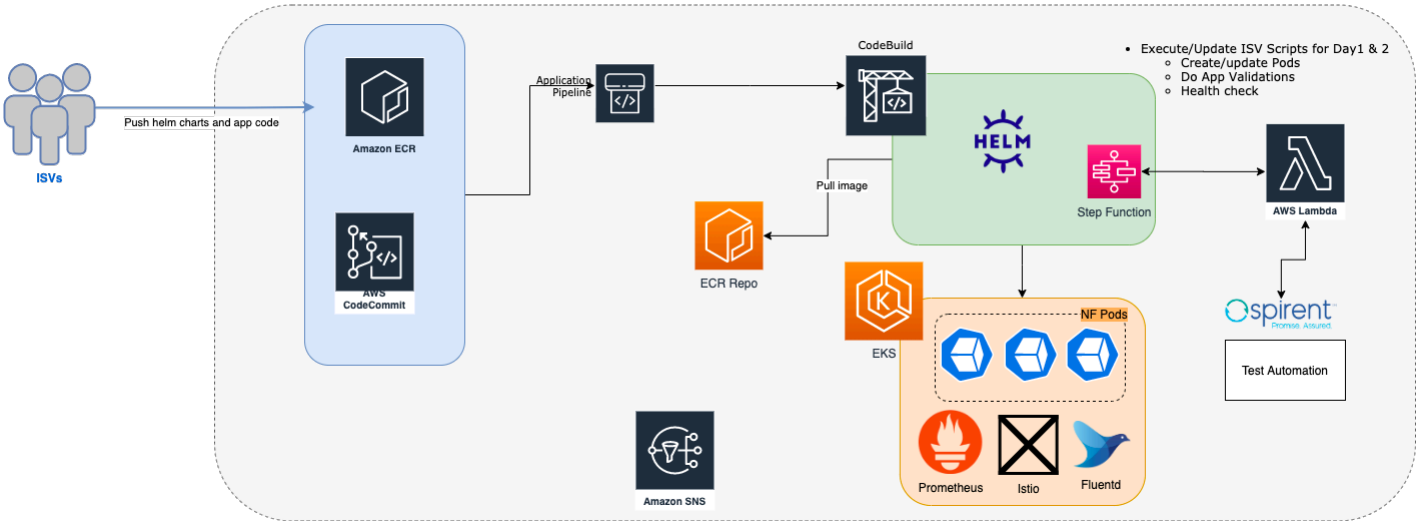
- Distribuzione continua CNF

Configurazione della rete

Concentrati sulla configurazione dei prerequisiti dell'infrastruttura. Ciò include la creazione di VPC, reti, sottoreti, NACL e così via. Progetta il piano di rete IP considerando gli ISV e il piano di implementazione del cliente (ad esempio allocazioni multi-tenancy statiche o dinamiche). Questo piano può essere codificato in AWS CDK o AWS CloudFormation. L'esecuzione di questo codice implementa i prerequisiti di rete dell'infrastruttura cloud .

Implementazione dell'infrastruttura

L'implementazione dell'infrastruttura effettua il provisioning di qualsiasi componente dell'infrastruttura. Include la generazione del cluster EKS e dell'infrastruttura di supporto, come EFS, nodi (worker) EKS, sistemi di bilanciamento del carico e la configurazione del cluster in base ai requisiti delle funzioni di rete native per il cloud. In base ai requisiti CNF, AWS implementa anche interfacce di rete aggiuntive per i nodi, comprese le interfacce Multus. La maggior parte dei passaggi di implementazione e configurazione di un'applicazione si esegue una sola volta e viene aggiornata solo quando è necessario per l'applicazione.



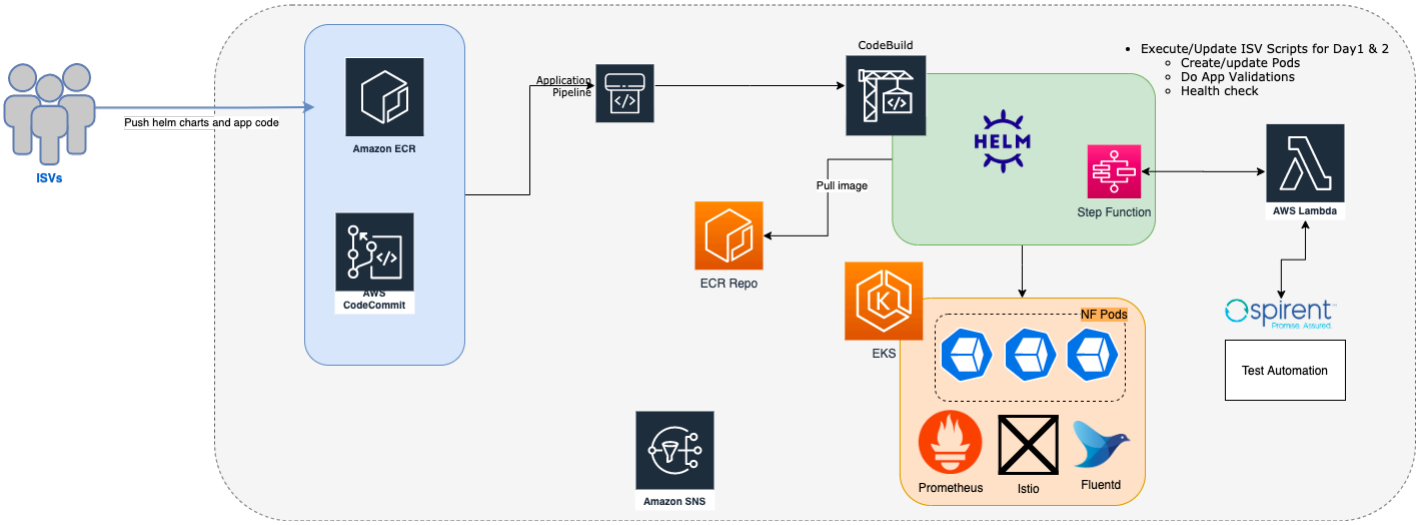
Implementazione dell'infrastruttura con CDK

Implementazione di funzioni di rete native per il cloud

L'implementazione CNF riguarda l'implementazione dell'applicazione. Come parte dell'implementazione CNF, i grafici Helm per l'applicazione sono implementati tramite la pipeline di

codice CI/CD. I callback per l'esecuzione di singoli script specifici dell'applicazione che coinvolgono principalmente controlli preventivi e consuntivi sono incorporati. I grafici Helm sono implementati in una sequenza in base alle esigenze dell'applicazione e controllano lo stato dei pod Kubernetes prima di passare alla fase successiva dell'implementazione. Spesso, gli ISV forniscono uno script wrapper per eseguire i grafici Helm e i controlli di integrità. Questi script ISV vengono richiamati da AWS CodePipeline. Nell'ambito di questa fase, gli agenti di registrazione e monitoraggio come Prometheus e FluentD vengono implementati in aggiunta ad Amazon CloudWatch, che registra e monitora l'infrastruttura cloud dell'applicazione.

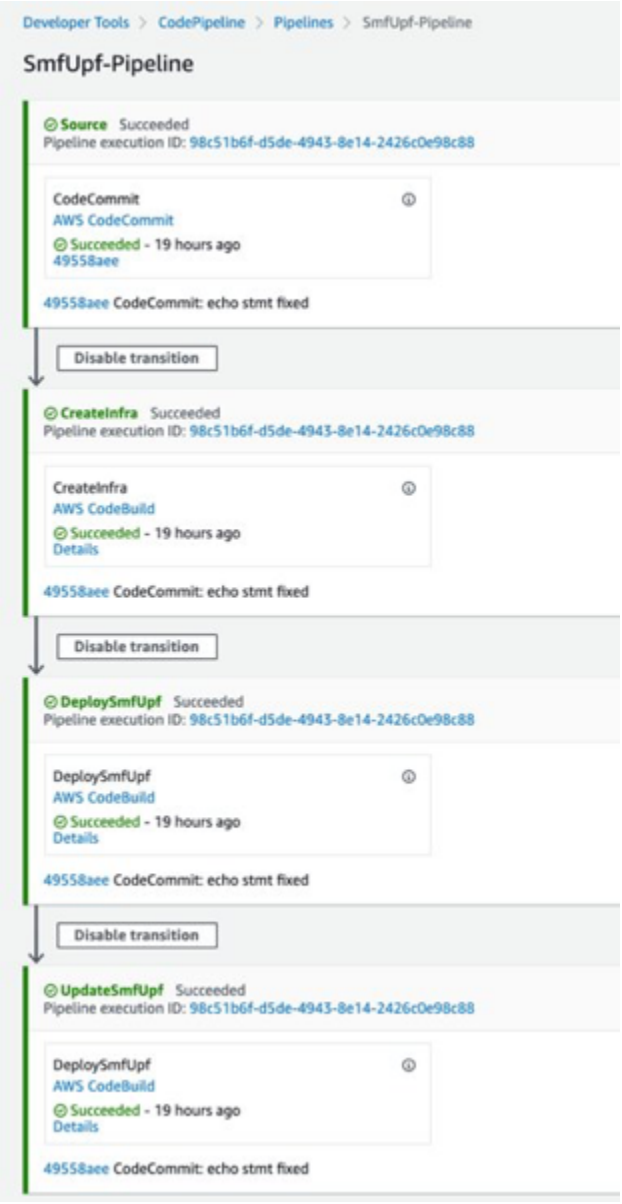
La pipeline di codice è integrata con il framework di automazione dei test di terze parti. La pipeline di codice può chiamare direttamente le API del framework di automazione dei test per eseguire il test dell'applicazione implementata, ottenere i risultati del test e analizzare il risultato, semplificando l'implementazione e il test dell'applicazione.



Implementazione e aggiornamento dell'applicazione

Di seguito è riportato un esempio di implementazione della funzione di piano per utente/funzione di gestione della sessione (UPF/SMF) CNF tramite AWS CodePipeline.

- Automatizza il processo CI/CD completo utilizzando CodeCommit, CodeBuild e CodePipeline.
- Le attività di creazione dell'infrastruttura e installazione delle applicazioni sono integrate come parte della pipeline.
- Gli agenti FluentD e Prometheus vengono installati e creati nei pannelli di controllo Amazon CloudWatch.



Esempio di implementazione UPF/SMF CNF

Distribuzione continua CNF

Si tratta di una sequenza di passaggi che vengono eseguiti ripetutamente per implementare le modifiche di configurazione di container che comportano aggiornamenti. La distribuzione continua CNF è automatizzata tramite pipeline ed è specifica delle singole applicazioni. AWS utilizza i grafici Helm standard per aggiornare CNF specifiche. La pipeline del codice prevede controlli preventivi e consuntivi per lo stato di aggiornamento dell'applicazione. La pipeline CI/CD aggiornata è inoltre integrata con un framework di automazione dei test per eseguire test automatizzati. Questa astrazione consente un'implementazione pulita delle funzioni di rete.

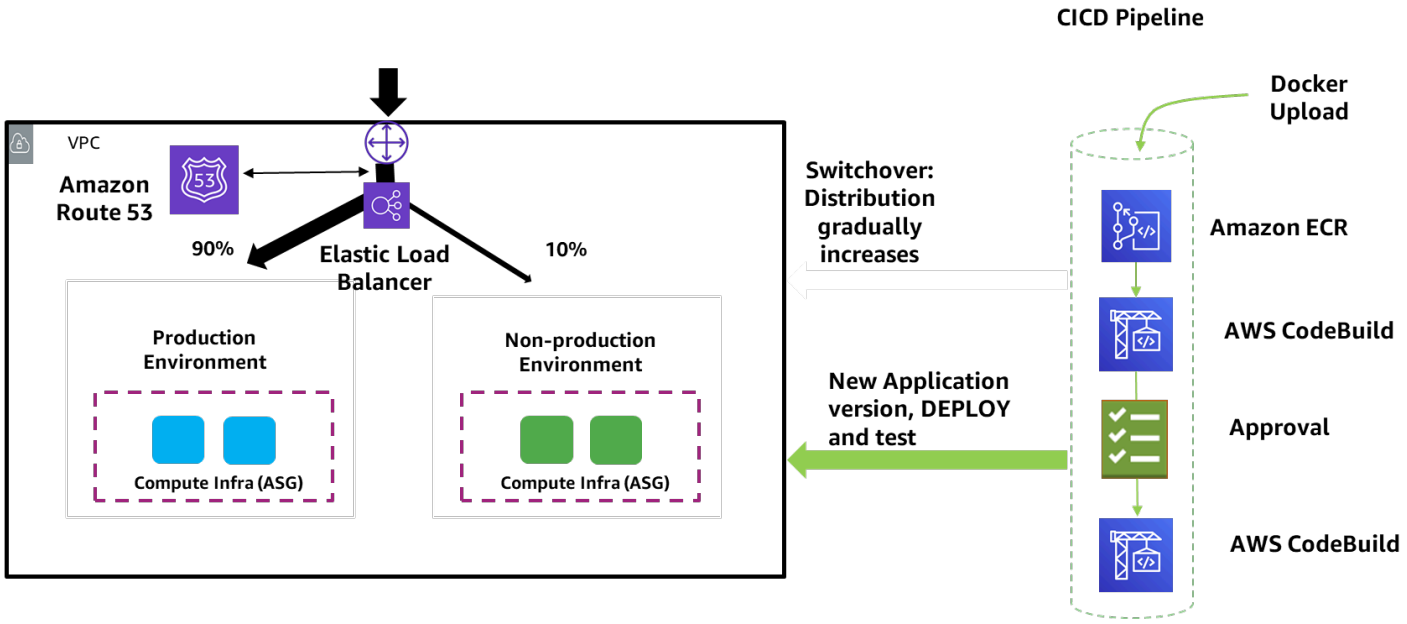
L'implementazione e la distribuzione continua CNF possono essere ampiamente classificate nelle seguenti categorie:

- Aggiornamenti delle applicazioni: la maggior parte degli aggiornamenti delle applicazioni è costituita da modifiche nei pod delle applicazioni Kuberbetes Questi aggiornamenti possono essere applicati automaticamente tramite la pipeline di codice. La maggior parte delle CNF supporta gli aggiornamenti locali fornendo più istanze di pod delle applicazioni. Le istanze multiple consentono un approccio di aggiornamento continuo. Non tutte le modifiche del pod delle applicazioni supportano l'aggiornamento Helm. Le pipeline tengono conto di queste variazioni e utilizzano l'installazione e l'eliminazione di Helm secondo le necessità.
- Aggiornamenti principali: gli aggiornamenti principali sono costituiti fondamentalmente da modifiche dello schema di database. Queste modifiche non possono essere applicate senza causare tempi di inattività. L'approccio standard consiste nell'eliminare l'applicazione e ricreare i pod pertinenti. Durante il processo è possibile che l'applicazione non sia disponibile. Per gli aggiornamenti vengono utilizzati i seguenti strumenti:

    - AWS CloudFormation consente ai clienti di descrivere ed effettuare il provisioning di tutte le risorse dell'infrastruttura in modelli JSON o YAML. AWS CloudFormation fornisce un potente meccanismo di estensione tramite risorse personalizzate supportate da Lambda. I clienti possono estendere AWS CloudFormation oltre le risorse AWS ed effettuare il provisioning delle risorse richieste in altri ambienti, ad esempio risorse On-Premise in ambienti ibridi. AWS CDK offre agli sviluppatori la possibilità di creare codice utilizzando linguaggi di programmazione di livello superiore già familiari, come Python, TypeScript, JavaScript, Java e C #, e quindi compilare il codice in un formato AWS CloudFormation JSON di livello inferiore, che può quindi essere implementato.
    - Implementazione blu/verde: AWS supporta e consiglia implementazioni blu/verdi e basate su canary sia negli ambienti di test che in quelli di produzione. Le implementazioni blu/verdi consentono ai clienti di testare una nuova versione dell'applicazione in un ambiente contenuto. Forniscono un metodo semplice e valido per passare al traffico di produzione. Le implementazioni basate su canary estendono questo concetto consentendo di testare l'ambiente verde non di produzione con una piccola percentuale del traffico di produzione per scoprire eventuali problemi causati dal traffico di produzione. La nuova versione dell'applicazione viene testata rispetto al traffico di test simulato interno e a piccole quantità di traffico di produzione, che aumenta la confidenza dell'utente prima di passare al traffico di produzione. Il traffico

di produzione è gradualmente aumentato fino al completamento del passaggio. L'implementazione coinvolge gruppi di destinazione ponderati DNS e ponderati ELB.

- L'automazione può essere ottenuta configurando AWS CodePipeline con le fasi di implementazione blu/verde e canary. La fase di approvazione può essere condotta manualmente all'inizio durante il provisioning, ma in seguito deve essere completamente automatizzata. Negli ambienti di test, è buona norma eseguire sempre il test con un'operazione di ripristino dello stato precedente per convalidare la compatibilità con le versioni successive e precedenti, prima di eseguire l'implementazione in produzione. L'implementazione blu/verde nei cluster con mesh di servizio dipende dal supporto fornito dall'applicazione finale e dal gateway di routing per la mesh di servizio per realizzare una transizione valida.
- AWS Systems Manager fornisce un'interfaccia utente unificata che consente di visualizzare i dati operativi di più servizi AWS utilizzati dalle funzioni di rete implementate da CI/CD. Systems Manager ti consente di automatizzare le attività operative delle risorse AWS.



Distribuzione canary

Sicurezza

La sicurezza è un elemento fondamentale. Di seguito è riportato un elenco di passaggi di sicurezza che il processo CI/CD di AWS prende in considerazione durante l'implementazione di un'applicazione.

- Origine: il repository ECR assegnato al fornitore è configurato con un flag "Scan on Push" (Scansione su invio) abilitato, in modo che qualsiasi caricamento di immagini Docker venga immediatamente sottoposto a una scansione di sicurezza. Le esposizioni e le vulnerabilità comuni (CVE) note vengono contrassegnate con notifiche. Oltre a ECR, quando i fornitori inseriscono grafici nel repository AWS CodeCommit, viene richiesto loro di crittografare tutte le password utilizzate con Secrets Manager piuttosto che con testo normale.
- Integrità degli artefatti: gli artefatti utilizzati nella pipeline vengono crittografati sia a riposo (utilizzando chiavi gestite da AWS) che in transito (utilizzando SSL/TLS).
- Utenti e ruoli IAM: le autorizzazioni fornite agli utenti o alle risorse si basano sul principio del privilegio minimo. Deve esserci una relazione di trust tra ruoli IAM che potresti dover configurare se usi le risorse in servizi diversi. Ad esempio, AWS CodeBuild necessita dell'autorizzazione per eseguire comandi su un cluster Amazon EKS.
- Verifica: la funzionalità di verifica offerta da AWS CloudTrail tiene traccia di ogni chiamata API tra servizi e operazioni degli utenti e consente la valutazione di eventi passati.
- Scansione delle vulnerabilità delle immagini: le immagini CNF caricate in Amazon ECR vengono automaticamente scansionate alla ricerca di vulnerabilità di sicurezza. Il report dei risultati della scansione è disponibile nella AWS Management Console e può essere recuperato anche tramite API. I risultati possono quindi essere inviati agli operatori CSP per le azioni correttive, inclusa la sostituzione dell'immagine CNF.

I controlli di sicurezza vengono effettuati in varie fasi della pipeline per garantire che l'immagine appena caricata sia sicura e conforme ai controlli di conformità desiderati, in modo da poter inviare una notifica ai CSP per l'approvazione:

- Il registro del container esegue la scansione per individuare eventuali vulnerabilità CVE aperte.
- La configurazione viene controllata per individuare eventuali perdite di informazioni, modelli di informazioni personali di identificazione (PII) noti, durante la fase di test, attivando le regole di controllo della conformità per rilevare i problemi come le porte TCP/UDP aperte impreviste e le vulnerabilità DOS.

- La compatibilità con le versioni precedenti e successive viene verificata per la sicurezza di aggiornamento/ripristino dello stato precedente.

Oltre all'applicazione, è fondamentale fornire la sicurezza della pipeline garantendo il trasferimento crittografato degli artefatti tra le fasi, sia a riposo che in transito.

Osservabilità

AWS consente l'osservabilità per le CNF 5G implementate in AWS di default. Questa funzionalità è abilitata da Amazon CloudWatch. CloudWatch offre una visibilità completa delle risorse e delle applicazioni cloud.

Amazon CloudWatch prevede quattro passaggi principali per questo processo:

1. Raccolta: raccogli parametri e registri da tutte le risorse, le applicazioni e i servizi AWS eseguiti su server AWS e On-Premise.
2. Monitoraggio: visualizza le applicazioni e l'infrastruttura con i pannelli di controllo CloudWatch, correla log e parametri per la risoluzione dei problemi e imposta avvisi con CloudWatch Alarms.
3. Azione: automatizza la risposta alle modifiche operative con CloudWatch Events e AWS Auto Scaling.
4. Analisi: parametri fino a un secondo, conservazione estesa dei dati (15 mesi) e analisi in tempo reale con CloudWatch Metric Math.

L'agente Amazon CloudWatch è installato nel cluster Kubernetes del cliente. L'agente supporta le caratteristiche di configurazione, rilevamento e pull dei parametri di Prometheus, arricchendo e pubblicando tutti i parametri e i metadati Prometheus ad alta fedeltà come EMF (Embedded Metric Format) in CloudWatch Logs.

Amazon CloudWatch Container Insights automatizza il rilevamento e la raccolta dei parametri Prometheus dalle applicazioni containerizzate. Raccoglie, filtra e crea automaticamente parametri CloudWatch personalizzati aggregati e visualizzati nei pannelli di controllo.

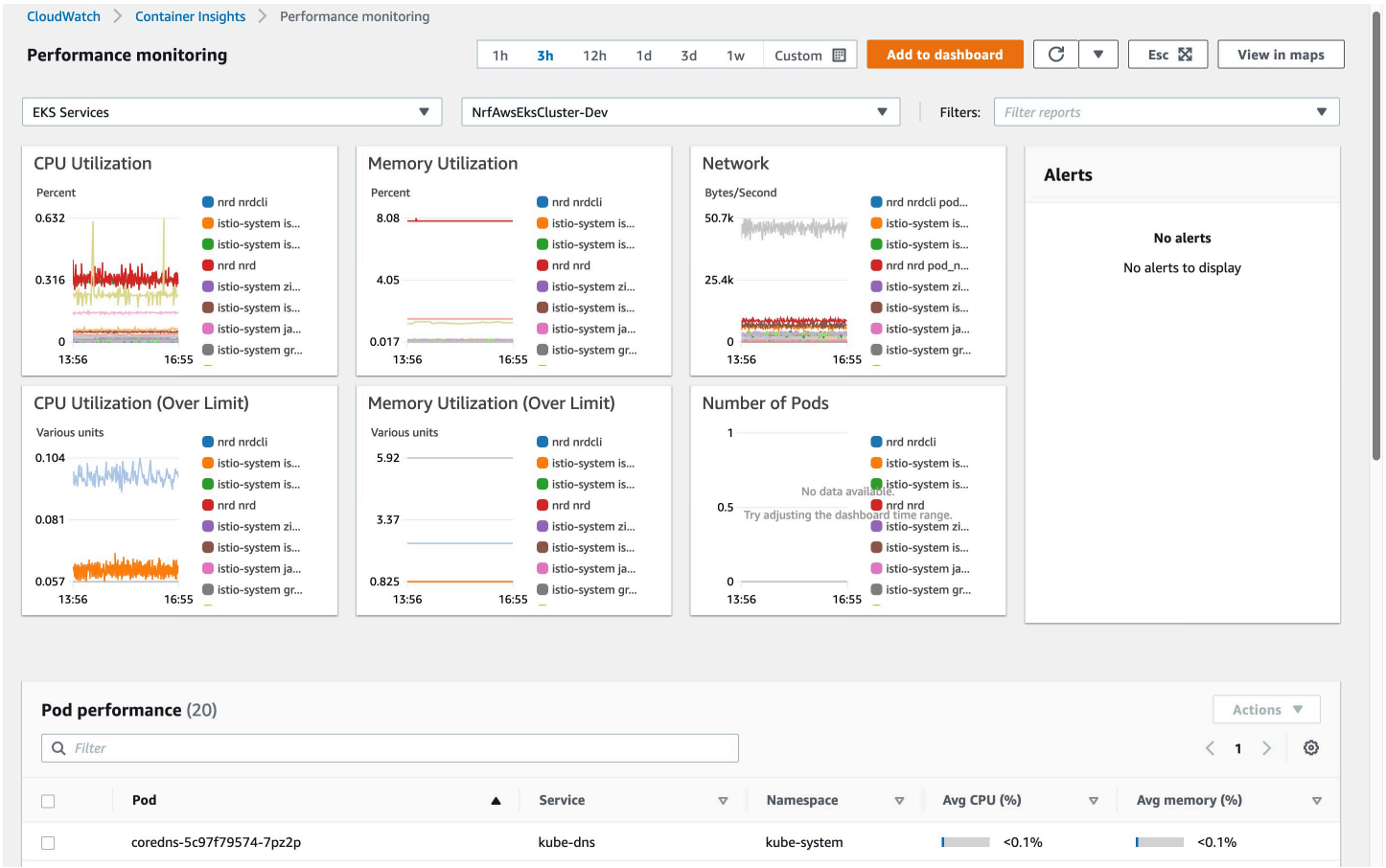
Ogni evento crea punti dati dei parametri come parametri personalizzati di CloudWatch per un set curato di dimensioni di parametro completamente configurabili. La pubblicazione di parametri aggregati Prometheus come statistiche di parametri personalizzati CloudWatch riduce il numero di parametri necessari per monitorare, avvisare e risolvere problemi di prestazioni e guasti. Puoi anche analizzare i parametri Prometheus ad alta fedeltà utilizzando il linguaggio di query CloudWatch Logs

Insights per isolare pod ed etichette specifici che influiscono sull'integrità e sulle prestazioni dei tuoi ambienti containerizzati.

AWS CloudTrail offre questa visibilità, registrando ogni chiamata API per i servizi. AWS Config include la funzionalità per la convalida della conformità. AWS offre ai clienti opzioni di monitoraggio aggiuntive di parametri, registri, eventi per l'applicazione, l'infrastruttura e le pipeline, utilizzando vari servizi come AWS X-Ray e AWS CloudTrail.

- AWS può integrare in modo nativo strumenti di parametri open source come Prometheus, FluentD e così via.
- I parametri Prometheus possono essere ulteriormente importati in Amazon CloudWatch o OpenSearch Service per ulteriori analisi.
- AWS utilizza FluentD come meccanismo standard per raccogliere i registri da vari sistemi. Lo stesso meccanismo viene usato e configurato per questo progetto.

Per i dettagli su come configurare questo meccanismo, consulta Configurazione di FluentD come DaemonSet per inviare i log a CloudWatch Logs.



Esempio di parametri monitorati da Amazon CloudWatch

Orchestrazione CI/CD con strumenti open source e di terze parti

Il livello di orchestrazione utilizza IaC per implementare e configurare l'infrastruttura sottostante necessaria per eseguire le funzioni di rete 5G. Questo livello deve essere progettato per essere modulare, portatile e riutilizzabile.

L'infrastruttura segue le best practice native per il cloud, che la rendono altamente disponibile, ridondante e scalabile.

Come dimostrato nelle sezioni precedenti, l'implementazione dell'infrastruttura sottostante può essere ottenuta con AWS Cloud Development Kit tramite Terraform di Hashicorp.

Terraform

Terraform è uno strumento software IaC open source che fornisce un flusso di lavoro coerente con l'interfaccia a riga di comando (CLI) per gestire centinaia di servizi cloud. Terraform codifica le API cloud in file di configurazione dichiarativi.

Per l'implementazione con Terraform si usano gli stessi principi utilizzati in CDK. Il codice è strutturato in moduli che consentono di personalizzare e riutilizzare i componenti di rete in base alle esigenze del fornitore.

La configurazione è completamente parametrizzata e quindi è possibile personalizzare le implementazioni in base ai fornitori e ai suggerimenti degli ISV.

L'implementazione delle funzioni di rete è suddivisa in due fasi:

- L'infrastruttura AWS richiesta viene creata e gestita tramite un repository centrale.
- La configurazione e il codice sono archiviati centralmente in un repository GitHub.

Dopo aver creato i prerequisiti, la funzione di rete è pronta per essere implementata utilizzando una pipeline dell'applicazione impostata nella fase precedente.

Implementazione dell'infrastruttura

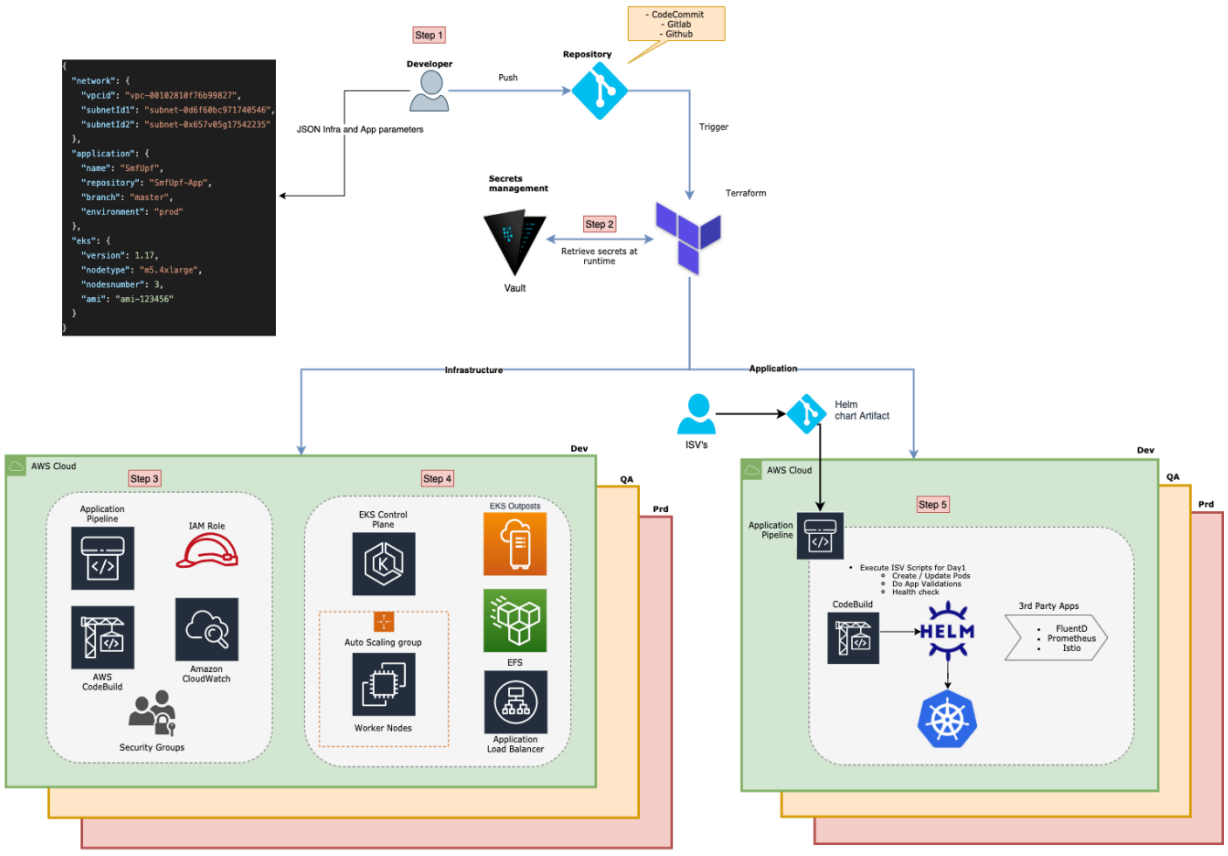
L'implementazione dell'infrastruttura include tutti i prerequisiti per implementare e configurare la funzione di rete.

Alcuni dei componenti creati nell'ambito di questa fase sono:

- Reti: VPC, sottoreti pubbliche e private, routing, bilanciatori del carico
- Calcolo: Kubernetes (Vmware Tanzu, Amazon EKS o AWS Outposts), istanze Amazon EC2, nodi primari e nodi (worker), gruppo Auto Scaling
- Archiviazione: Amazon EFS, Amazon EBS, bucket Amazon S3
- Sicurezza: ruoli IAM, gruppi di sicurezza
- Pipeline: CodePipeline, CodeBuild
- Osservabilità: CloudWatch, Prometheus, FluentD

Ecco la sequenza dell'infrastruttura orchestrata da Terraform spiegata nella figura sotto:

1. Uno sviluppatore popola un file JSON archiviato in un repository centrale con il codice IaC. Il file contiene informazioni sulla configurazione dell'infrastruttura desiderata, come la dimensione delle istanze, la versione di Kubernetes, le informazioni di rete e i dettagli del repository delle applicazioni.
2. Vengono recuperati i segreti da HashiCorp Vault o AWS Secrets Manager in fase di esecuzione.
3. Vengono implementati e configurati i componenti dell'infrastruttura (rete, calcolo, archiviazione e sicurezza).
4. Viene implementato un cluster Amazon EKS con nodi (worker) che ospitano i pod della funzione di rete. Amazon EKS può essere implementato anche in AWS Outposts per supportare carichi di lavoro che richiedono la vicinanza a un datacenter.
5. Viene creata e configurata una pipeline dell'applicazione per ascoltare le modifiche nel repository della funzione di rete. Ogni volta che il codice viene inviato al ramo del repository configurato, la pipeline attiva automaticamente la creazione, il test e l'implementazione della funzione di rete.
6. Gli strumenti di osservabilità che raccolgono e centralizzano registri e parametri vengono implementati come servizi in tutti i nodi e forniscono pressoché in tempo reale dati che possono essere visualizzati nei pannelli di controllo Grafana oppure OpenSearch



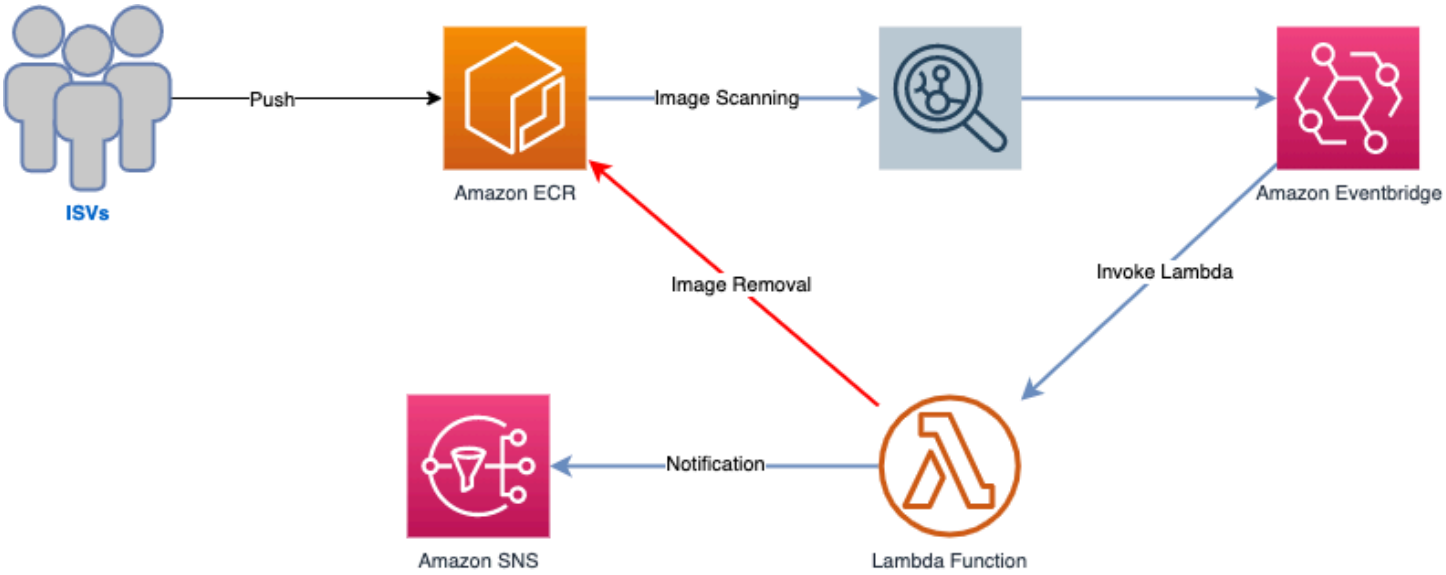
Implementazione e configurazione delle funzioni di rete

Implementazione e configurazione delle funzioni di rete

La pipeline creata nella fase precedente consente agli ISV e ai provider di decentralizzare e ottimizzare l'implementazione delle funzioni di rete. La pipeline è connessa e ascolta le modifiche nel repository dell'applicazione, configurato nel file JSON del passaggio 1 della figura precedente.

Per controllare le immagini pubblicate da terze parti, viene implementata e configurata una soluzione di scansione delle vulnerabilità che aiuta a identificare le vulnerabilità del software nelle immagini dei container. La soluzione di scansione controlla automaticamente tutte le nuove immagini inviate ad Amazon ECR. Per ulteriori informazioni sulla scansione delle immagini ECR, vedi Scansione delle immagini.

Nella figura seguente è illustrata l'architettura della soluzione di scansione delle vulnerabilità delle immagini.



Architettura della soluzione di scansione delle vulnerabilità delle immagini

La pipeline dell'applicazione può essere configurata per essere attivata dalle modifiche nell'immagine dopo il risultato della scansione o da modifiche dirette nel repository. Ad esempio, quando viene creata una nuova immagine Helm.

L'elenco seguente è la sequenza che crea/aggiorna la funzione di rete, come illustrato nella figura seguente:

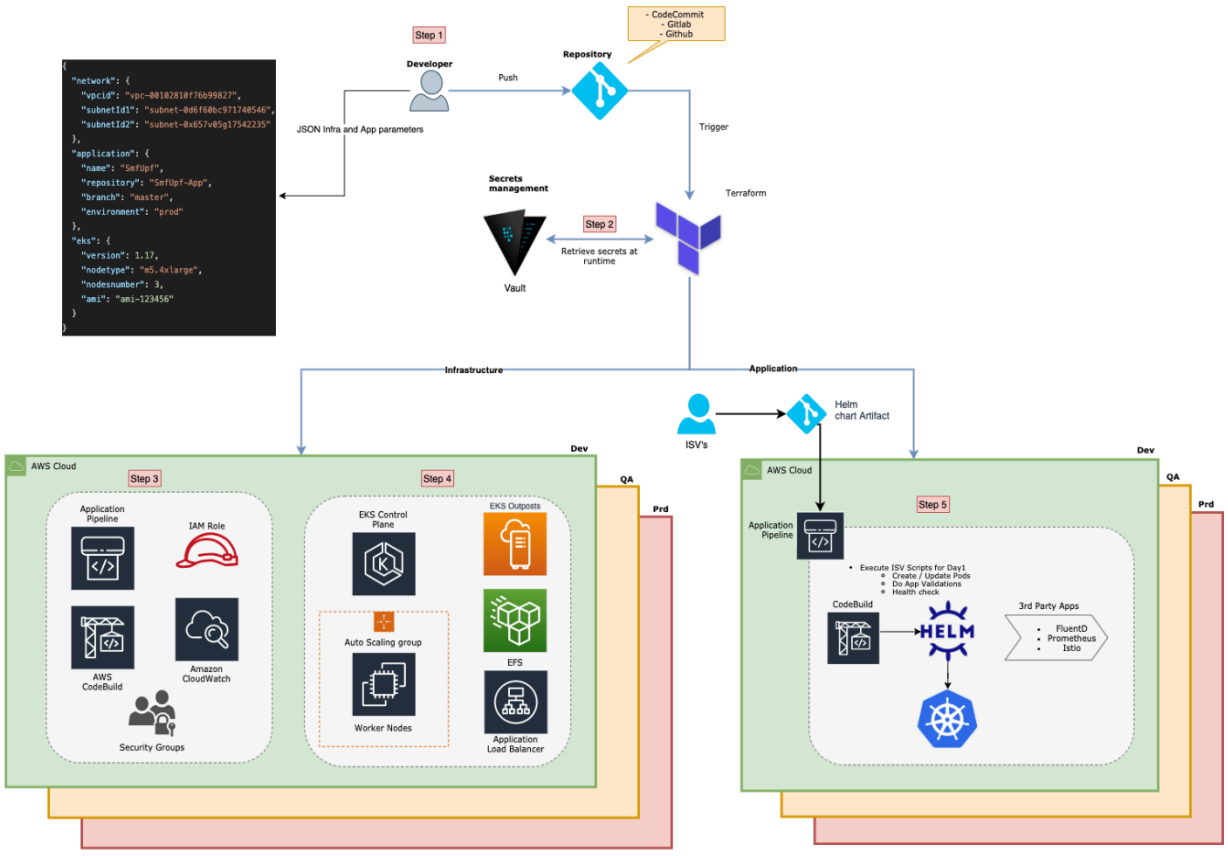
Gli ISV pubblicano nuove immagini in Amazon ECR. Se l'immagine è approvata, viene attivata la pipeline dell'applicazione.

CodePipeline estrae la nuova immagine da Amazon ECR e utilizza CodeBuild per implementare l'immagine in Kubernetes. I comandi Helm possono essere usati per aggiornare la funzione di rete.

Dopo l'implementazione dell'immagine, viene attivato il Test as Service (TaS). Il TaS convalida la nuova implementazione e centralizza i dati e i parametri delle prestazioni delle funzioni di rete in condizioni di stress.

I registri e i parametri vengono raccolti e centralizzati in OpenSearch e Grafana. Anche prodotti di terze parti come Datadog, Istio e Prometheus possono essere configurati per fornire una maggiore osservabilità.

Inoltre, è possibile implementare e integrare nella soluzione un framework MANO in grado di coordinare le risorse di rete. Utilizza i dati raccolti per intraprendere operazioni automatizzate come il network slicing e la scalabilità automatica della qualità del servizio (QoS, Quality of Service).



Pipeline dell'applicazione

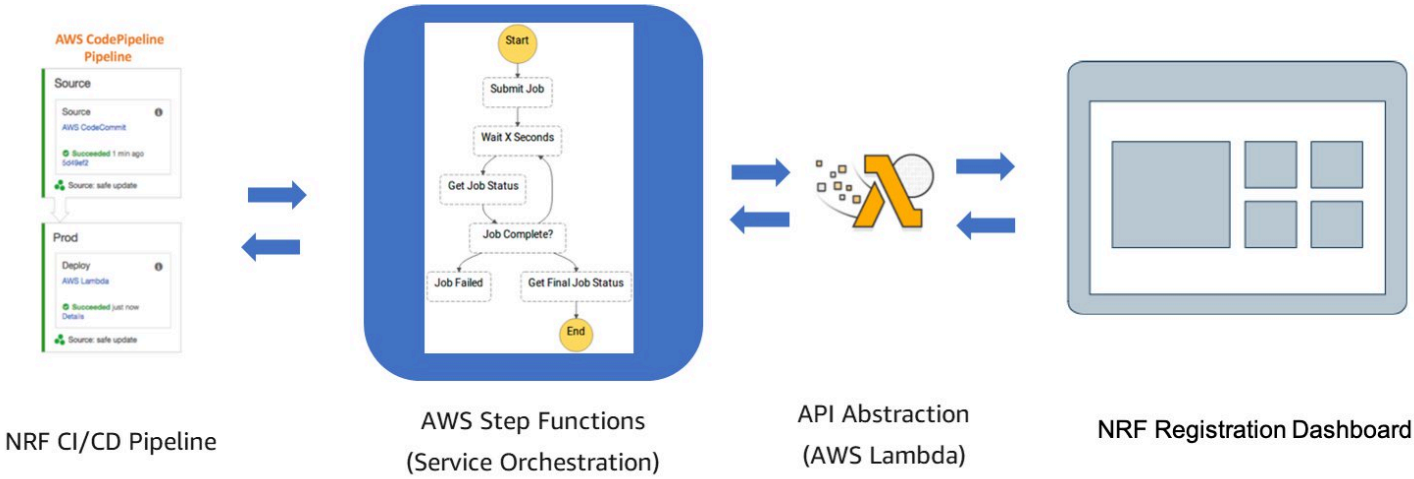
Esecuzione di test

Il framework di automazione dei test specifico per le telecomunicazioni può essere integrato nella pipeline del codice. La pipeline del codice viene integrata con le step function per orchestrare l'integrazione con un framework di automazione dei test. Le step function di AWS utilizzano più funzioni Lambda per richiamare il framework di automazione dei test (strumenti di terze parti) tramite chiamate API. La step function inizialmente recupera l'ID del test, quindi esegue il test e ottiene i risultati dal framework di automazione dei test. La step function analizza i risultati e li passa alla pipeline del codice per l'approvazione dell'applicazione e l'implementazione in produzione. L'approvazione può essere automatizzata o gestita manualmente nella pipeline del codice, se necessario. Questo è un passo importante per i CSP perché possono promuovere l'implementazione dall'ambiente di test in produzione. Le API di alto livello richieste per l'integrazione sono classificate nel modo seguente:

- Ottieni il contesto

- Esegui un test case specifico
- Arresta il test case
- Ottieni risultati

La complessità della chiamata alle API REST esterne è modellata utilizzando le step function di AWS, che consentono ai costrutti standard di richiamare flussi paralleli, attendere i risultati, diramarsi in base alle condizioni e integrare l'API REST con AWS CodePipeline.



Flusso di test

CI/CD e orchestrazione

L'integrazione continua e la distribuzione continua fanno parte di una filosofia di automazione globale che viene fornita con l'architettura nativa per il cloud e si applica al 5G. L'orchestrazione è un altro aspetto di questa filosofia che deve essere dinamico e reattivo a qualsiasi cambiamento che avvenga nella rete. L'orchestrazione e le operazioni di CI/CD devono essere strettamente associate per garantire un servizio integro e per ridurre al minimo le interruzioni del servizio. L'integrazione tra CI/CD e orchestrazione deve avvenire su due fronti:

- L'applicazione di patch e aggiornamenti nel sistema deve essere gestita e orchestrata in modo da ridurre al minimo le interruzioni di qualsiasi servizio live. Ad esempio, l'orchestrazione può determinare dinamicamente il momento migliore in cui un aggiornamento deve essere implementato.
- L'orchestrazione basata su CI/CD consente di spostare il traffico durante l'implementazione degli aggiornamenti in base alla strategia del modello di implementazione adottata (canary, lineare o all-at-once).

In genere, le soluzioni di orchestrazione vengono eseguite sulle pipeline CI/CD per consentire all'orchestrazione di introdurre fasi di governance nelle pipeline e avere esposizione nei cicli di aggiornamento in corso.

Conclusione

CI/CD fornisce un percorso chiaro ed efficiente per sviluppatori e team dell'applicazione per implementare un nuovo codice dell'applicazione in pochi minuti. AWS dispone di un'ampia gamma di strumenti che possono aiutare gli sviluppatori nell'integrazione, nel test e nell'implementazione di nuovo codice, tra cui AWS CodePipeline, AWS CodeCommit, AWS CodeBuild, AWS CodeDeploy e molti altri. Questo documento ha illustrato come si possono utilizzare i servizi AWS per creare un processo CI/CD per l'implementazione della funzione di rete 5G in modo completamente automatizzato, inclusi i diversi passaggi necessari per l'implementazione di nuovo codice. Ha anche esaminato come integrare un framework di automazione dei test di terze parti nel processo CI/CD e come utilizzare strumenti di terze parti come Terraform.

Collaboratori

I collaboratori di questo documento includono:

- Hisham Elshaer, Senior Consultant, AWS Telecom, Amazon Web Services
- Vara Prasad Talari, Principal Consultant, AWS Telecom, Amazon Web Services
- Rabi Abdel, Principal Consultant, AWS Telecom, Amazon Web Services
- Franco Bontorin, Senior Consultant, Shared Delivery, Amazon Web Services
- Pragtideep Singh, Consultant, Shared Delivery, Amazon Web Services
- Subbarao Duggisetty, Cloud Infra Architect, Global Accounts, Amazon Web Services
- Young Jung, Senior Partner Solutions Architect, AWS Telecom, Amazon Web Services

Revisioni del documento

Per ricevere una notifica sugli aggiornamenti di questo whitepaper, iscriviti al feed RSS.

update-history-change	update-history-description	update-history-date
Pubblicazione originale	Prima pubblicazione del whitepaper	8 marzo 2021

Approfondimenti

Per ulteriori informazioni, consulta:

- Lezioni pratiche di integrazione e distribuzione continua in AWS (whitepaper)
- Mobile Packet Core Network di livello carrier in AWS (whitepaper)
- Evoluzione della rete 5G con AWS (whitepaper)

Acronimi

- AMF: Access & Mobility Management Function, funzione di gestione mobilità e accesso
- API: Application Programming Interface
- AUSF: Authentication Server Function, funzione di server di autenticazione
- BSS: Business Support System, sistema di supporto aziendale
- CDK: Cloud Development Kit
- CI/CD: Continuous Integration/Continuous Delivery, integrazione continua/distribuzione continua
- CLI: Command Line Interface, interfaccia a riga di comando
- CNF: Cloud-Native/Containerized Network Function, funzione di rete containerizzata o nativa per il cloud
- CSP: Communication Service Provider, provider di servizi di comunicazione
- CU: RAN Central Unit, unità centrale RAN
- CVE: Common Vulnerabilities and Exposures, esposizioni e vulnerabilità comuni
- DoS: Denial of Service
- DR: Disaster Recovery, ripristino di emergenza
- DU: RAN Distributed Unit, unità distribuita RAN
- E2E: End-to-End
- ECR: Elastic Container Registry
- EFS: Elastic File System
- EKS: Elastic Kubernetes Service
- EPC: Evolved Packet Core
- IaC: Infrastructure as Code
- ISV: Independent Software Vendor, fornitore di software indipendente
- MANO: Management and Orchestration, gestione e orchestrazione
- MEC: Multi-Access Edge Computing
- NACL: Network Access Control List, lista di controllo accessi di rete
- NAT: Network Address Translation
- NF: Network Function, funzione di rete
- NFV: Network Function Virtualization, virtualizzazione delle funzioni di rete

- NFVO: Network Function Virtualization Orchestrator, orchestratore di virtualizzazione delle funzioni di rete
- NOC: Network Operations Centre, centro operatività di rete
- NRF: Network Repository Function, funzione di repository di rete
- OSS: Operations Support System, sistema di supporto operativo
- PII: Personally Identifiable Information, informazioni personali di identificazione
- QoS: Quality of Service, qualità del servizio
- RAN: Radio Access Network
- SBI: Service Based Interface, interfaccia basata su servizio
- SMF: Session Management Function, funzione di gestione delle sessioni
- SSL: Secure Sockets Layer
- TaS: Test as Service
- TCP: Transmission Control Protocol
- TLS: Transport Layer Security
- UDM: Unified Data Management
- UDP: User Datagram Protocol
- UPF: User Plane Function, funzione di piano per utente
- VIM: Virtualized Infrastructure Manager
- VNF: Virtual Network Function, funzione di rete virtuale
- VPC: Virtual Private Cloud, cloud privato virtuale

Avvisi

I clienti sono responsabili della propria valutazione autonoma delle informazioni contenute in questo documento. Questo documento: (a) è solo a scopo informativo, (b) mostra le offerte e le pratiche attuali dei prodotti AWS soggette a modifiche senza preavviso, e (c) non crea alcun impegno o garanzia da parte di AWS e dei suoi affiliati, fornitori o licenziatari. I prodotti o servizi AWS sono forniti "così come sono" senza garanzie, dichiarazioni o condizioni di alcun tipo, sia esplicite che implicite. Le responsabilità e gli obblighi di AWS verso i propri clienti sono disciplinati dagli accordi AWS e il presente documento non fa parte né modifica alcun accordo tra AWS e i suoi clienti.