AWS-Whitepaper

Continuous Integration und Continuous Delivery für 5G-Netzwerke auf AWS

Continuous Integration und Continuous Delivery für 5G-Netzwerke auf AWS: AWS-Whitepaper

Table of Contents

Continuous Integration und Continuous Delivery für 5G-Netzwerke auf AWS

Veröffentlichungsdatum: 08. März 2021 (Dokumentversionen)

Überblick

In diesem Whitepaper werden Continuous Integration und Continuous Delivery (CI/CD) für 5G-Netzwerke vorgestellt und es wird erläutert, wie Tools und Services von Amazon Web Services (AWS) zur vollständig automatisierten Bereitstellung und Aktualisierung von 5G-Netzwerkfunktionen verwendet werden können. Das Whitepaper enthält eine detaillierte Beschreibung der verschiedenen Phasen von CI/CD für 5G-Netzwerkfunktionen, einschließlich der Einrichtung des Netzwerks, der Bereitstellung der Infrastruktur, der Bereitstellung von cloudnativen Netzwerkfunktionen und der kontinuierlichen Aktualisierung von Netzwerkfunktionen. Es enthält auch Details zur Integration mit Open-Source- und Drittanbieter-Tools für Tests, zur Beobachtbarkeit und Orchestrierung.

Dieses Whitepaper richtet sich sowohl an Anbieter von Kommunikationsservices (CSPs) als auch an unabhängige Softwareanbieter (ISVs).

Einführung

In der Vergangenheit dauerten Entwicklung, Labor- und Feldintegrationstests sowie die Produktionsbereitstellung neuer Netzwerkknoten oder neuer Funktionen in einem Mobilfunknetz Wochen oder sogar Monate, damit die Stabilität von missions- und geschäftskritischen Telekommunikationsservices gewährleistet blieb. Der lange Bereitstellungszyklus wurde durch die monolithische Architektur herkömmlicher Netzwerkknoten, eine Umgebung mit mehreren Anbietern und viele Punkt-zu-Punkt-Schnittstellen zwischen den Netzwerkentitäten in den Mobilfunknetzen 2G, 3G und 4G verursacht.

Wie im Whitepaper 5G-Netzwerkentwicklung mit AWS vorgestellt, unterstützen die von 3GPP standardisierten 5G-Mobilfunknetze jetzt eine cloudnative Architektur, die durch Virtualisierung und Containerisierung ermöglicht wird. Genauer gesagt, führen 5G-Netze ein neues Paradigma der Microservice-, zustandslosen und servicebasierten Architektur ein und unterstützen es.

Diese 5G-Architektur bedeutet, dass verschiedene Netzwerkfunktionen als lose gekoppelte unabhängige Services arbeiten können, die über genau definierte Schnittstellen und APIs miteinander kommunizieren. Am wichtigsten ist, dass jede Netzwerkfunktion unabhängig aktualisiert werden kann. Diese Architekturverschiebung in 5G ermöglicht CSPs mehr Agilität und betriebliche Effizienz, da sie es einfacher macht, Updates für Netzwerkfunktionen häufiger durchzuführen und dabei gleichzeitig Testabläufe, Sicherheitsanforderungen und Standards durch die Automatisierung beizubehalten.

Die Integration und Bereitstellung neuer Funktionen für einen CSP beginnt in der Regel, wenn der Anbieter der Netzwerkfunktion ein neues Softwarepaket für die Netzwerkfunktion freigibt, z. B. ein Docker-Image in einer Container-basierten Netzwerkfunktion oder eine neue Konfigurationsdatei wie ein Helm-Chart im Kubernetes-Anwendungsfall. (Ein Helm-Chart ist eine Sammlung von Dateien, die einen verwandten Satz von Kubernetes-Ressourcen beschreiben.)

Die Idee, das CI/CD-Paradigma für die Bereitstellung von 5G-Netzwerkfunktionen zu nutzen, gewinnt an Zugkraft, aber die praktische Umsetzung dieser Idee ist eine Herausforderung in der Telekommunikationsbranche.

AWS hat bei der Entwicklung neuer CI/CD-Tools für die Softwarebereitstellung Pionierarbeit geleistet, um ein breites Spektrum von Branchen bei der schnellen Entwicklung und Einführung von Softwareänderungen zu unterstützen und gleichzeitig die Systemstabilität und -sicherheit zu gewährleisten. Diese Tools umfassen eine Reihe von Services für Software Development and Operations (DevOps, Softwareentwicklung und -betrieb) wie AWS CodeStar, CodeCommit, CodePipeline, CodeBuild und CodeDeploy.

AWS propagiert auch die Idee von Infrastructure as Code (IaC) mithilfe von AWS Cloud Development Kit (AWS CDK), AWS CloudFormation und API-basierten Tools von Drittanbietern wie Terraform. Mit diesen Tools kann AWS die Bereitstellungsprozesse der Netzwerkfunktion innerhalb von AWS als Quellcode speichern und diesen IaC-Quellcode in der CI/CD-Pipeline verwalten, um eine kontinuierliche Bereitstellung zu realisieren.

In diesem Whitepaper werden detaillierte Prozesse für die Nutzung von AWS IaC- und CI/CD-Tools für die Bereitstellung und Aktualisierung der 5G-Netzwerkfunktion beschrieben. Darüber hinaus wird in diesem Whitepaper die Integration mit Tools von Drittanbietern für Tests, zur Beobachtbarkeit und Orchestrierung behandelt.

AWS CI/CD-Tools sind nicht auf 5G-Netzwerkfunktionen beschränkt. Sie werden auch für die Automatisierung der Bereitstellung von 4G-Netzen eingesetzt, wodurch CSP 4G-Netzwerkfunktionen schnell und effizient einrichten und aktualisieren können. Die meisten 4G-Netzwerkfunktionen basieren auf Virtual Network Functions (VNF, Virtuelle Netzwerkfunktionen). AWS CI/CD-Tools wie AWS CloudFormation können zur Automatisierung der Bereitstellung von 4G-VNFs verwendet werden und sorgen für Skalierbarkeit und Zeiteffizienz bei 4G-Netzwerkbereitstellungen.

Continuous Integration und Continuous Delivery

Continuous Integration

Continuous Integration (CI) ist ein Softwareprozess, bei dem Entwickler ihren Code regelmäßig in ein zentrales Repository wie AWS CodeCommit oder GitHub verschieben. Jeder Code-Push löst einen automatischen Build aus, gefolgt von der Durchführung von Tests. Das Hauptziel von CI ist die frühzeitige Erkennung von Codeproblemen, die Verbesserung der Codequalität und die Verkürzung der Zeit, die für die Validierung und Veröffentlichung neuer Software-Updates benötigt wird.

Continuous Delivery und Deployment

Continuous Delivery (CD) ist ein Softwareprozess, bei dem Artefakte in der Testumgebung, der Staging-Umgebung und der Produktionsumgebung bereitgestellt werden. Continuous Delivery kann vollständig automatisiert sein oder an kritischen Punkten Genehmigungsstufen enthalten. Dadurch wird sichergestellt, dass alle erforderlichen Genehmigungen vor der Bereitstellung vorliegen, wie z. B. die Genehmigung für das Versionsmanagement. Bei einer korrekten Implementierung von Continuous Delivery haben Entwickler stets ein Erstellungsartefakt für die Bereitstellung, das bereits einen standardisierten Testprozess durchlaufen hat.

Beim Continuous Deployment werden Änderungen automatisch in einer Produktionsumgebung bereitgestellt. Die vollständige Automatisierung des Software-Release-Prozesses bedeutet, dass keine explizite Genehmigung durch einen Entwickler vorliegen muss. Dies ermöglicht eine kontinuierliche Kunden-Feedback-Schleife schon zu Beginn des Produktlebenszyklus.

Mit Continuous Deployment wird jede festgeschriebene Änderung, die die automatisierten Tests besteht, automatisch für die Produktion freigegeben. Continuous Delivery ist nicht dazu gedacht, festgeschriebene Änderungen, die die automatisierten Tests bestehen, sofort auf die Produktion anzuwenden. Continuous Delivery soll sicherstellen, dass jede Änderung auf die Produktion angewendet werden kann.

Infrastruktur als Code

Wie im Whitepaper 5G Network Evolution with AWS beschrieben, ist IaC ein wichtiger Faktor für die Automatisierung des Bereitstellungsprozesses und des Lebenszyklusmanagements – sowohl

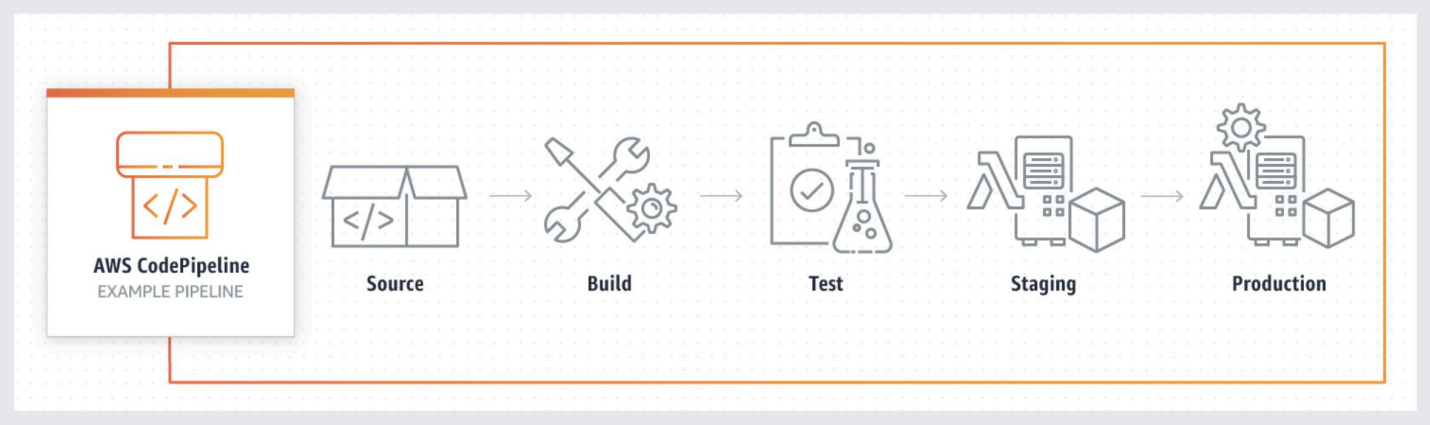
für die Anwendung als auch für deren Umgebung. Anstatt sich auf manuell ausgeführte Schritte zu verlassen, können Netzwerk- und IT-Administratoren sowie auch Entwickler die Infrastruktur mithilfe von Konfigurationsdateien einrichten. IaC behandelt diese Konfigurationsdateien als Softwarecode. Mithilfe dieser Dateien können eine Reihe von Artefakten erzeugt werden: nämlich die Rechen-, Speicher-, Netzwerk- und Anwendungsservices, die eine Betriebsumgebung ausmachen. IaC eliminiert Konfigurationsabweichungen durch Automatisierung und erhöht so die Geschwindigkeit und Agilität von Infrastrukturbereitstellungen.

Bei einer Implementierung der Network Function Virtualization (NFV) in AWS bietet dieses IaC-Framework einen Mehrwert hinsichtlich der Orchestrierung. Von der Erstellung der Virtual Private Cloud (VPC) bis zur Bereitstellung der Netzwerkfunktionen kann jeder Schritt programmiert, als Quellcode verwaltet und mit der Versionskontrolle in AWS CodeCommit gepflegt werden.

Dieses IaC-Framework für Netzwerkfunktionen führt zu einer wiederholbaren und zuverlässigen Erstellung und Bereitstellung von Infrastruktur- und Netzwerkfunktionen, die auf die End-to-End-Automatisierung (E2E) des Netzwerk-Slice-Managements und des Service-Lebenszyklusmanagements ausgedehnt werden kann. AWS bietet ein umfassendes Toolset für die Erstellung, Wartung und Bereitstellung von Infrastrukturen auf programmatische, beschreibende und deklarative Weise und unter Verwendung von Services wie AWS CloudFormation, AWS CDK, AWS CDK für Kubernetes und die API-Offenlegung aller AWS-Services.

CI/CD auf AWS

Sie können sich CI/CD als Pipeline vorstellen, bei der neuer Code an einem Ende eingereicht und dann in verschiedenen Phasen (Quelle, Build, Test, Staging und Produktion) getestet und als produktionsbereiter Code veröffentlicht wird.



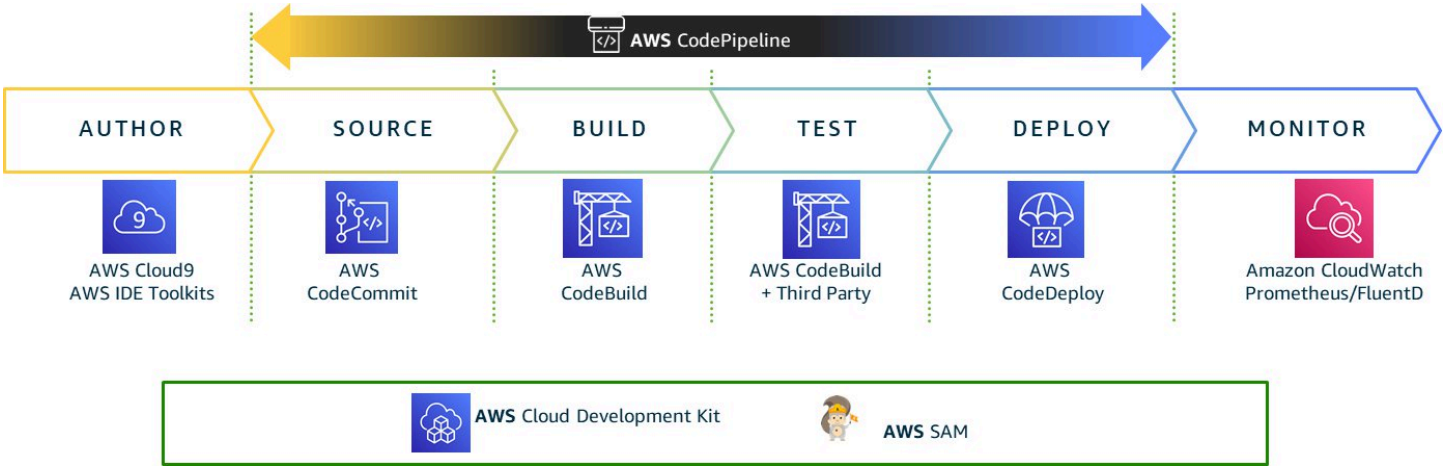
Übersicht über die CI/CD-Pipeline

Jede Phase der CI/CD-Pipeline stellt im Bereitstellungsprozess eine logische Einheit dar. Jede Stufe fungiert als Tor, das einen bestimmten Aspekt des Codes untersucht. Wenn der Code die Pipeline durchläuft, sollte seine Qualität in den späteren Phasen höher sein, da ja immer mehr Aspekte überprüft werden. Werden in einer frühen Phase Probleme erkannt, durchläuft der Code die Pipeline nicht weiter. Die Ergebnisse der Tests werden sofort an das Team gesendet und alle weiteren Builds und Releases werden gestoppt, wenn die Software eine Phase nicht erfolgreich besteht.

AWS bietet einen vollständigen Satz von CI/CD-Entwicklertools zur Beschleunigung der Softwareentwicklung und Veröffentlichungszyklen. AWS CodePipeline automatisiert die Build-, Test- und Bereitstellungsphasen des Veröffentlichungsprozesses bei jeder Codeänderung auf der Grundlage des definierten Veröffentlichungsmodells. Dies ermöglicht die schnelle und zuverlässige Bereitstellung von Funktionen und Updates.

Code-Pipelines können in andere Services integriert werden. Dies können AWS-Services wie Amazon Simple Storage Service (Amazon S3) sein oder Produkte von Drittanbietern, wie z. B. GitHub. AWS CodePipeline kann für eine Vielzahl von Entwicklungs- und Betriebsanwendungsfälle eingesetzt werden, darunter:

- Code mit AWS CodeBuild kompilieren, entwickeln und testen
- Continuous Delivery von Container-basierten Anwendungen in der Cloud
- Validierung von Artefakten (z. B. Deskriptoren und Container-Images) vor der Bereitstellung, die für Netzwerkservices oder bestimmte cloudnative Netzwerkfunktionen erforderlich sind
- Funktions-, Integrations- und Leistungstests für containerisierte Netzwerkfunktionen/virtuelle Netzwerkfunktionen (CNF/VNF), einschließlich Baseline- und Regressionstests
- Tests zur Zuverlässigkeit und Notfallwiederherstellung



AWS CI/CD-Pipeline-Komponenten

AWS kann CI/CD-Pipelines mit den folgenden AWS-Entwicklertools einrichten:

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

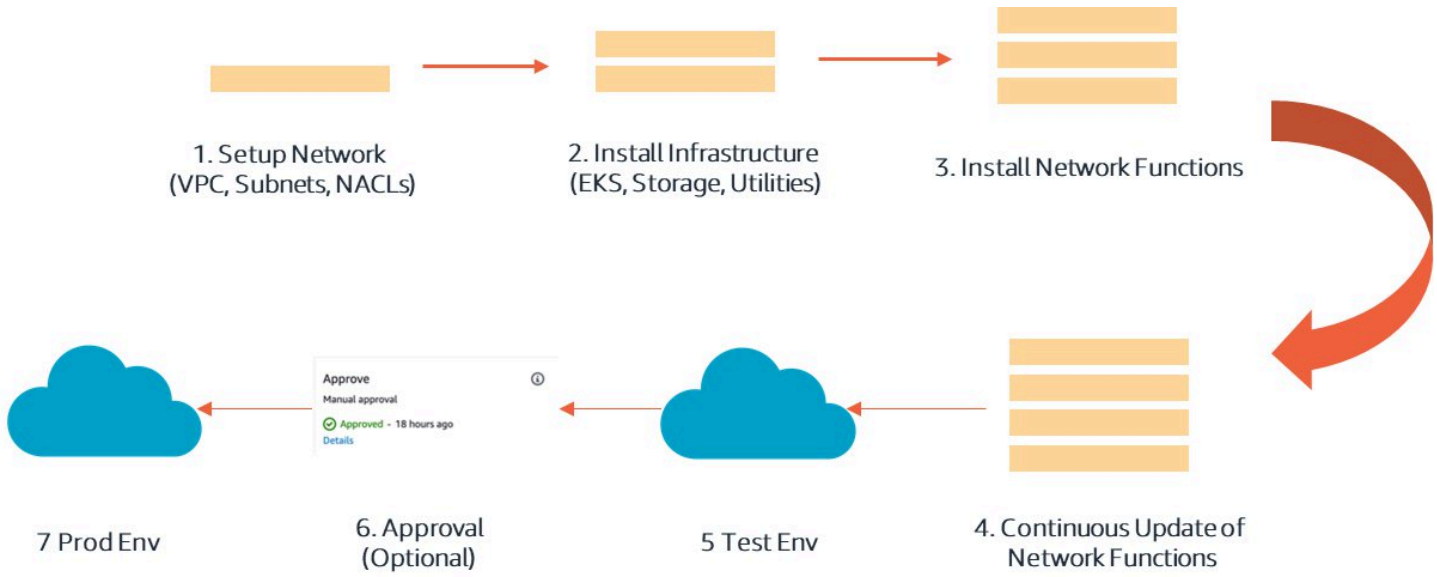
Die Erstellung einer CI/CD-Pipeline kann mithilfe von AWS CDK und AWS CloudFormation automatisiert werden. In der NFV-Domäne kann diese AWS-native Automatisierung in ein Framework zur Verwaltung und Orchestrierung (MANO) und das Framework zur Service-Orchestrierung des CSPs integriert werden.

Der CI/CD-Prozess umfasst die folgenden Schritte:

- Netzwerkeinrichtung – AWS CDK und AWS CloudFormation initiieren die Erstellung der Netzwerkvoraussetzungen:
    - Netzwerk-Stack (VPC, Subnetze, NAT-Gateway (Network Address Translation), Routing-Tabelle und Internet-Gateway)
- Infrastrukturbereitstellung – AWS CDK und AWS CloudFormation initiieren die Erstellung der folgenden Ressourcen-Stacks:
    - Compute-Stack (Amazon Elastic Kubernetes Service (Amazon EKS) Cluster-Erstellung, EKS Worker-Knoten, AWS Lambda)
    - Speicher-Stack (Amazon S3-Buckets, Amazon EBS-Volumes (Amazon Elastic Block Store) und Amazon Elastic File System (Amazon EFS)
    - Überwachungs-Stack (CloudWatch , Amazon OpenSearch Service (OpenSearch Service)

    - Sicherheits-Stack (AWS Identity and Access Management (AWS IAM), Amazon Elastic Compute Cloud (EC2)-Sicherheitsgruppen, VPC Netzwerkzugriffskontrolllisten (NACLs)

- Bereitstellung der Cloud Network Function (CNF) – In dieser Phase werden CNFs mithilfe von Kubectl und Helm-Chart-Tools auf EKS-Clustern bereitgestellt. In dieser Phase werden auch alle spezifischen Anwendungen oder Tools bereitgestellt, die für ein effizientes Funktionieren der CNFs erforderlich sind (z. B. Prometheus oder Fluentd). CNFs können entweder über Lambda-Funktionen oder mit AWS CodeBuild bereitgestellt werden.
- Fortlaufende Aktualisierungen und Bereitstellung – Hierbei handelt es sich um eine Abfolge von wiederholt ausgeführten Schritten, um Container-/Konfigurationsänderungen bereitzustellen, die zu Upgrades führen. Ähnlich wie bei der CNF-Bereitstellung können kontinuierliche Aktualisierungen und Bereitstellungen mithilfe von AWS-Services automatisiert werden, wobei der Auslöser AWS CodeCommit, Amazon Elastic Container Registry (Amazon ECR) oder das Quellsystem eines Drittanbieters wie z. B. GitLab Webhooks sein kann.



Flussdiagramm der AWS CI/CD-Pipeline

Die CI/CD-Pipeline wird mit AWS CodePipeline erstellt und nutzt einen Continuous Delivery-Service, der die erforderlichen Schritte für den Software-Release modelliert, visualisiert und automatisiert. Durch die Definition von Phasen in einer Pipeline können Sie Code aus einem Quellcode-Repository abrufen, diesen Quellcode in ein veröffentlichungsfähiges Artefakt umwandeln, das Artefakt testen und es in der Produktion bereitstellen. Es wird nur Code bereitgestellt, der alle diese Phasen

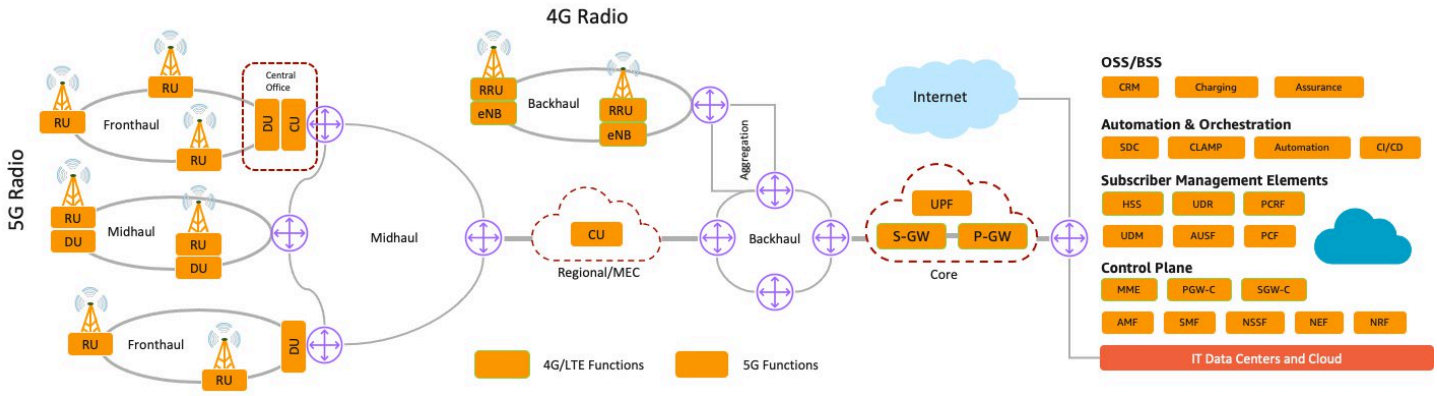
erfolgreich durchläuft. Sie können optional weitere Anforderungen in Ihre Pipeline aufnehmen, z. B. manuelle Genehmigungen, um sicherzustellen, dass nur genehmigte Änderungen in der Produktion bereitgestellt werden.

5G-Netzwerke auf AWS

Das typische Modell einer 5G-Netzwerkinfrastruktur besteht aus einem 4G/5G-Funkstandort, einem Fronthaul-/Midhaul-/Backhaul-Netzwerk, einem Kernnetzstandort und einem Telekommunikations-/IT-Rechenzentrum. CSPs können AWS-Services nutzen, um eine skalierbare, flexible 5G-Netzwerkinfrastruktur zu schaffen und gleichzeitig die Anfangsinvestitionskosten zu senken. AWS kann für die Implementierung des virtuellen Network Operation Centers (NOC) in der Region verwendet werden, in dem das Operations Support System/Business Support System (OSS/BSS) und die meisten Kernnetzfunktionen der Steuerebene gehostet werden.

AWS kann auch genutzt werden, um die lokale Central Office (CO) oder ein verteiltes Rechenzentrum mit einer Flotte von AWS Outposts-Instances zu implementieren, die hauptsächlich Funktionen der Benutzerebene hosten, wie UPF (User-Plane Function), RAN Central Unit (CU) und Multi-Access Edge Computing (MEC). Eine genauere Erläuterung zur Referenzarchitektur und zu den Vorteilen der 5G-Netzwerkimplementierung auf AWS finden Sie im Whitepaper 5G Network Evolution on AWS.

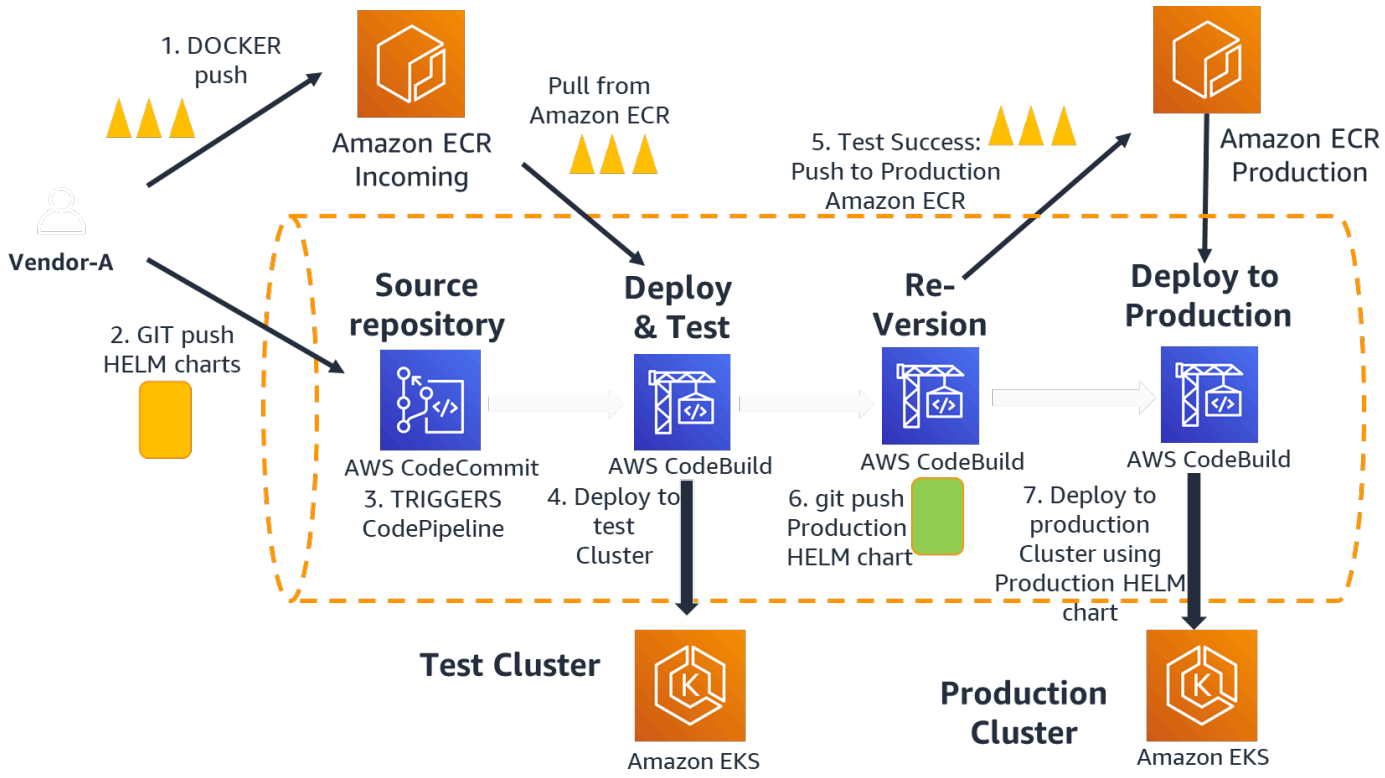
Wenn das 5G-Netzwerk auf AWS implementiert werden soll, können die in den folgenden Abschnitten dieses Whitepapers vorgestellten AWS CI/CD-Tools eine vollständige Automatisierung von Bereitstellung, Upgrades und Lebenszyklusmanagement der 5G-Netzwerkfunktionen erleichtern.



E2E-Architektur für das 5G-Netzwerk

CI/CD in 5G-Netzwerken

Das Designkonstrukt der Infrastruktur wird in Form von Code in einer deklarativen Sprache gespeichert. So kann der CSP bei Bedarf eine wiederholbare Reproduktion der Infrastruktur mit demselben erwarteten Verhalten abrufen. Der Code wird im Code-Repository verwaltet, und es wird eine Pipeline eingerichtet, um die Aktualisierungen der bereitgestellten Stacks zu orchestrieren (z. B. AWS CDK und AWS CloudFormation). AWS kann bei der Entwicklung von Infrastructure as Code (IaC) für das agile Onboarding von Funktionen unabhängiger Softwareanbieter (ISV) helfen.



Code-Pipeline-Verlauf

Änderungen an den Konfigurationen von cloudnativen Netzwerkfunktionen durch Helm-Charts werden als Auslöser für eine automatische CI/CD-Pipeline-Ausführung für die Netzwerkfunktionen betrachtet.

AWS CodeCommit kann zur Verwaltung von Konfigurationsdateien und Amazon ECR zur Aufbewahrung von Container-Images verwendet werden.

Wie in der Abbildung Code-Pipeline-Verlauf dargestellt, wird die Code-Pipeline ausgelöst, wenn der ISV neue Codeänderungen in das Code-Repository (Helm-Chart, Konfigurationsdateien oder eine Eigenschaftendatei) überträgt. Die Code-Pipeline ruft das Image aus ECR ab und verwendet

das Helm-Chart zur Bereitstellung der Anwendung. Die neuen Anwendungstests können in das Framework zur Testautomatisierung eines Drittanbieters integriert werden. Basierend auf dem Ergebnis können CSPs die Produktionsbereitstellung genehmigen.

Die Quellstufe von CodePipeline sucht nach Änderungen in den Konfigurationsdateien. Gültige Anbieter für die Quellstufe sind CodeCommit, Amazon S3, GitHub oder AWS CloudFormation. Alternative Quellsysteme können integriert werden, indem Lambda-Funktionen zur Implementierung von Webhooks verwendet werden. Dies ermöglicht eine ereignisgesteuerte Integration zwischen Gitlab und AWS CodePipeline. Unter den folgenden Links finden Sie eine ausführliche Implementierungsanleitung.

- Webhooks mit GitLab
- Integrationen der Container-Registry

Das Design der CI/CD-Pipeline sollte kritische Bereitstellungsschritte berücksichtigen, z. B. die Erstbereitstellung, das Testen und das Hochstufen in die Produktion, sobald die Testergebnisse die Anforderungen erfüllen und mit der Baseline abgeglichen wurden. Jede Phase des Pipeline-Prozesses liefert Datenartefakte, die Vergleiche und datengesteuerte Entscheidungen ermöglichen.

Schritte der CI/CD-Pipeline für Anwendungen

Jede Phase kann als separate Aufgabe betrachtet werden, was die Einbeziehung von Validierungs- und Bereitstellungsworkflows ermöglicht, die für die Unterstützung von Netzwerkservices und cloudnativen Netzwerkfunktionen geeignet sind. Bei der Ausführung von Aufgaben können zusätzliche Tools von Drittanbietern, wie z. B. Datenverkehrsgeneratoren und -simulatoren, einbezogen werden und so eine End-to-End-Validierung von Netzwerkservices ermöglichen.

AWS bietet den ausgefeilten Service AWS Step Functions (ein cloudnativer Zustandsautomat), der nativ mit anderen AWS-Services integriert werden kann und auch die Möglichkeit bietet, externe Systeme wie Jira oder ein Framework zur Testautomatisierung zu integrieren.

Detaillierte CI/CD-Schritte

Sie können sich CI/CD als Pipeline vorstellen, bei der neuer Code an einem Ende eingereicht und dann in verschiedenen Phasen (Quelle, Build, Test, Staging und Produktion) getestet und als produktionsbereiter Code veröffentlicht wird.

Dies sind die Schritte zur Bereitstellung und zum Testen. Bereitstellung und Konfiguration sind im Wesentlichen in vier Hauptabschnitte unterteilt:

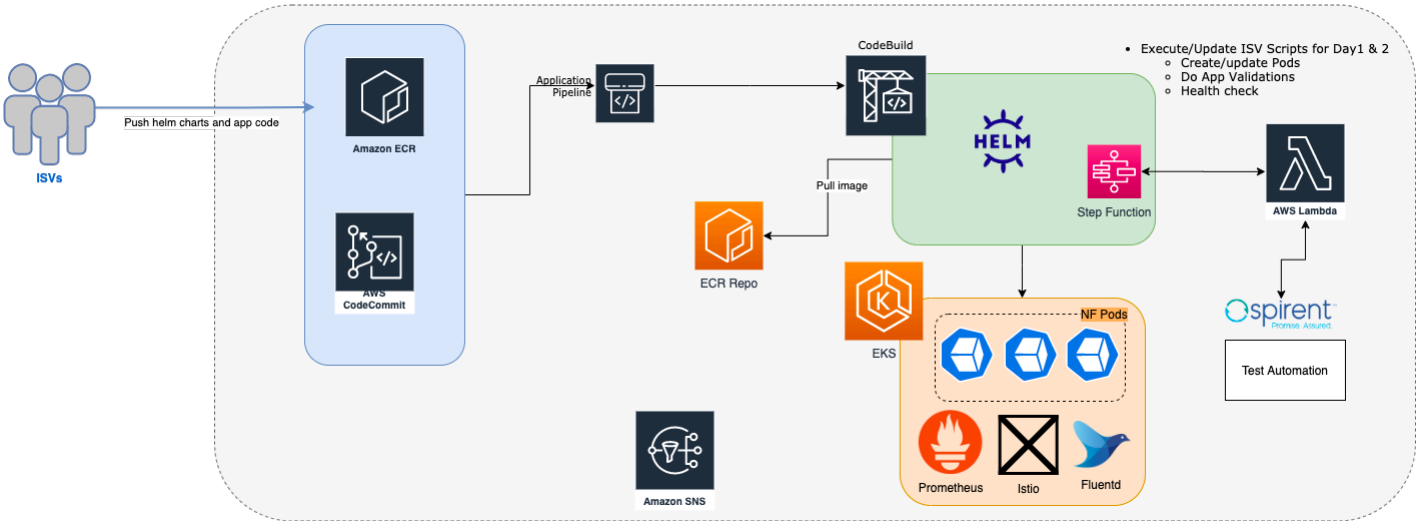
- Netzwerkeinrichtung
- Bereitstellung der Infrastruktur
- Bereitstellung von cloudnativen Netzwerkfunktionen
- Continuous Delivery von CNFs

Netzwerkeinrichtung

Konzentrieren Sie sich auf die Einrichtung der Voraussetzungen für die Infrastruktur. Dazu gehören die Erstellung der VPC, Netzwerke, Subnetze, NACLs usw. Entwerfen Sie den IP-Netzwerkplan unter Berücksichtigung der ISVs und des Implementierungsplans des Kunden (z. B. Multi-Tenancy und statische vs. dynamische Zuweisungen). Dieser Plan kann in AWS CDK oder AWS CloudFormation codiert werden. Durch Ausführen dieses Codes werden die Netzwerkvoraussetzungen für die Cloud-Infrastruktur bereitgestellt.

Bereitstellung der Infrastruktur

Bei der Infrastrukturbereitstellung werden alle Infrastrukturkomponenten bereitgestellt. Dazu gehört das Spawning des EKS-Clusters und der unterstützenden Infrastruktur, wie EFS, EKS-Worker-Knoten, ELBs und die Konfiguration des Clusters gemäß den Anforderungen an cloudnative Netzwerkfunktionen. Auf der Grundlage der CNF-Anforderungen stellt AWS auch zusätzliche Netzwerkschnittstellen für die Knoten bereit, einschließlich Multus-Schnittstellen. Die meisten Bereitstellungs- und Konfigurationsschritte sind ein einmaliger Aufwand für eine Anwendung und werden nur aktualisiert, wenn sie als Update für die Anwendung erforderlich sind.

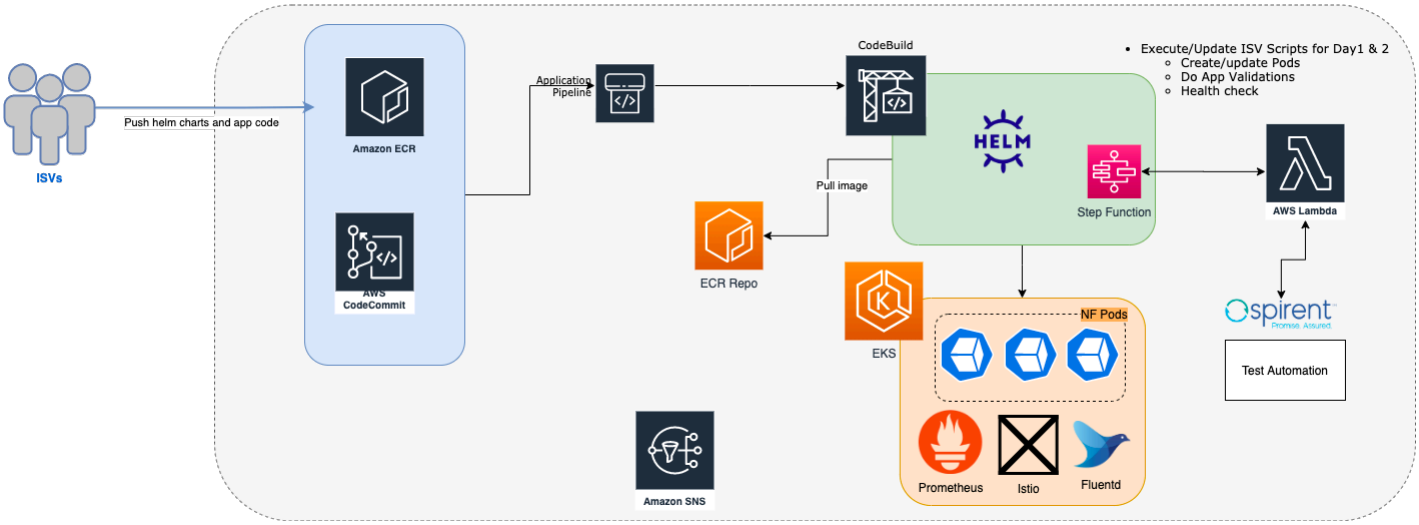


Infrastrukturbereitstellung mit CDK

Bereitstellung von cloudnativen Netzwerkfunktionen

Bei der CNF-Bereitstellung geht es um die Anwendungsbereitstellung. Als Teil der CNF-Bereitstellung werden die Helm-Charts für eine Anwendung durch die CI/CD-Code-Pipeline implementiert. Eingebunden ist auch der Callback zur Ausführung individueller anwendungsspezifischer Skripte, die in erster Linie Vor- und Nachprüfungen beinhalten. Die Helm-Charts werden in einer Reihenfolge implementiert, die den Anforderungen der Anwendung entspricht. Sie dienen der Überprüfung des Status der Kubernetes-PODs, bevor zum nächsten Schritt der Bereitstellung übergegangen wird. Oft stellen ISVs ein Wrapper-Skript bereit, um Helm-Charts und Zustandsprüfungen auszuführen. Diese ISV-Skripte werden aus AWS CodePipeline aufgerufen. In dieser Phase werden Protokollierungs- und Überwachungsagenten wie Prometheus und Fluentd zusätzlich zu Amazon CloudWatch bereitgestellt, um die Cloud-Infrastruktur der Anwendung zu protokollierten und zu überwachen.

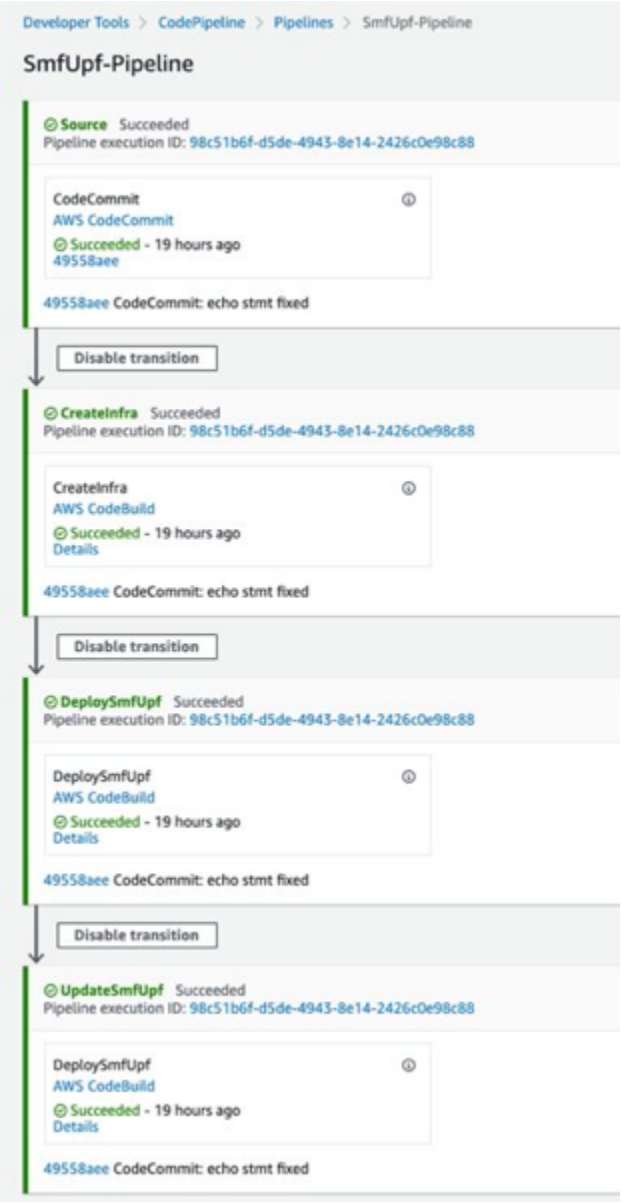
Die Code-Pipeline ist in das Framework zur Testautomatisierung von Drittanbietern integriert. Die Code-Pipeline kann die Framework-APIs für die Testautomatisierung direkt aufrufen, um den Test in der bereitgestellten Anwendung auszuführen, die Testergebnisse abzufragen und das Ergebnis zu analysieren. Dies vereinfacht die Bereitstellung und das Testen der Anwendung.



Anwendungsbereitstellung und -aktualisierung

Im Folgenden finden Sie ein Beispiel für den Einsatz der Benutzerebenenfunktion/Sitzungsverwaltungsfunktion (UPF/SMF) für CNFs über AWS CodePipeline.

- Automatisieren Sie den gesamten CI/CD-Prozess mit CodeCommit, CodeBuild und CodePipeline.
- Aufgaben zur Infrastrukturerstellung und Anwendungsinstallation sind in der Pipeline integriert.
- FluentD- und Prometheus-Agenten werden in Amazon CloudWatch-Dashboards installiert und erstellt.



Beispiel für die Bereitstellung von UPF/SMF-CNFs

Continuous Delivery von CNFs

Diese Phase besteht aus einer Abfolge von wiederholt ausgeführten Schritten zur Implementierung von Änderungen, die Teil von Container-/Konfigurationsänderungen sind und zu Upgrades führen. Continuous Delivery von CNFs wird über Pipelines automatisiert und ist spezifisch für einzelne Anwendungen. AWS verwendet Standard-Helm-Charts, um bestimmte CNFs zu aktualisieren. Die Code-Pipeline enthält Vor- und Nachprüfungen für den Aktualisierungsstatus der Anwendung. Die aktualisierte CI/CD-Pipeline ist auch in ein Framework zur Testautomatisierung integriert, um

automatisierte Tests durchzuführen. Diese Abstraktion ermöglicht eine saubere Bereitstellung von Netzwerkfunktionen.

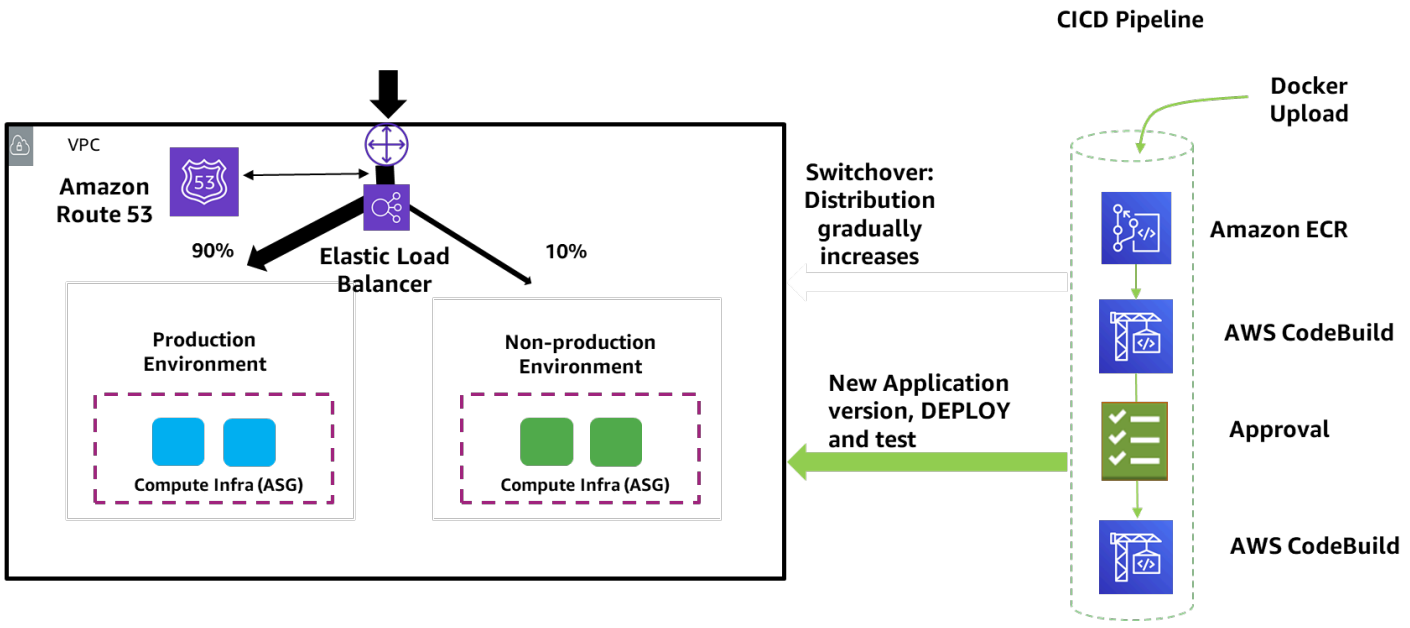
Continuous Delivery und Deployment von CNFs lassen sich grob in die folgenden Kategorien einteilen:

- Upgrades von Anwendungen – Die meisten Upgrades von Anwendungen sind Änderungen innerhalb der Kuberbetes-Anwendungs-PODs. Diese Aktualisierungen können automatisch über die Code-Pipeline angewendet werden. Die meisten CNFs unterstützen direkte Upgrades, indem sie mehrere Instances von Anwendungs-PODs bereitstellen. Mehrere Instances ermöglichen den Ansatz von fortlaufenden Upgrades. Nicht alle Änderungen an Anwendungs-PODs unterstützen das Upgrade von Helm. Pipelines berücksichtigen diese Variationen und verwenden die Installation bzw. das Löschen von Helm nach Bedarf.
- Größere Upgrades – Größere Upgrades sind in erster Linie Änderungen des Datenbankschemas. Diese Änderungen können nicht angewendet werden, ohne dass es zu Ausfallzeiten kommt. Der Standardansatz für diese Änderungen besteht darin, die Anwendung zu löschen und die entsprechenden Pods neu zu erstellen. Während des Vorgangs ist die Anwendung möglicherweise nicht verfügbar. Die folgenden Tools werden für Upgrades verwendet:

    - Mit AWS CloudFormation können Kunden alle Infrastrukturressourcen in JSON- oder YAML-Vorlagen beschreiben und bereitstellen. AWS CloudFormation bietet einen leistungsstarken Erweiterungsmechanismus durch Lambda-gestützte benutzerdefinierte Ressourcen. Kunden können AWS CloudFormation über AWS-Ressourcen erweitern und die erforderliche Ressource in anderen Umgebungen bereitstellen, z. B. lokale Ressourcen in Hybrid-Umgebungen. AWS CDK bietet Entwicklern die Möglichkeit, Code mit vertrauten Programmiersprachen auf höherer Ebene wie Python, TypeScript, JavaScript, Java und C# zu erstellen und den Code dann in ein AWS CloudFormation JSON-Format auf niedrigerer Ebene zu kompilieren, das dann bereitgestellt werden kann.
    - Blau-Grün-Bereitstellung – AWS unterstützt und empfiehlt Blau-Grün- und Canary-Bereitstellungen in Test- und auch in Produktionsumgebungen. Blau-Grün-Bereitstellungen ermöglichen es Kunden, eine neue Anwendungsversion in einer geschlossenen Umgebung zu testen. Sie bieten eine einfache und elegante Methode, den Produktionsdatenverkehr umzustellen. Canary-Bereitstellungen erweitern dieses Konzept, indem sie es ermöglichen, die grüne Nicht-Produktionsumgebung mit einem kleinen Teil des Produktionsdatenverkehrs zu testen, um etwaige durch den Produktionsdatenverkehr verursachte Probleme aufzudecken. Die neue

Anwendungsversion wird sowohl mit internem, simuliertem Testdatenverkehr als auch mit kleinen Mengen Produktionsdatenverkehr getestet, was dem Benutzer eine gewisse Sicherheit gibt, bevor er ganz auf den Produktionsdatenverkehr umschaltet. Der Produktionsdatenverkehr wird schrittweise erhöht, bis die Umstellung abgeschlossen ist. Die Implementierung umfasst gewichtete DNS- und ELB-Zielgruppen.

- Eine Automatisierung kann durch die Konfiguration von AWS CodePipeline mit den Phasen der Blau-Grün- und Canary-Bereitstellung erreicht werden. Die Genehmigungsphase kann anfangs während der Bereitstellung manuell gesteuert werden, später sollte sie jedoch vollständig automatisiert werden. In Testumgebungen empfiehlt es sich, immer mit einer Rollback-Aktion zu testen, um die Auf- und Abwärtskompatibilität zu prüfen, bevor die Bereitstellung in der Produktion erfolgt. Die Blau-Grün-Bereitstellung auf Clustern mit Service-Mesh hängt von der Unterstützung durch die Endanwendung und das Routing-Gateway ab, die Service-Mesh einen reibungslosen Übergang ermöglichen.
- AWS Systems Manager bietet eine einheitliche Benutzeroberfläche, auf der Sie die Betriebsdaten von mehreren AWS-Services aufrufen können, die von CI/CD-bereitgestellten Netzwerkfunktionen verwendet werden. Mit Systems Manager können Sie betriebliche Aufgaben für Ihre AWS-Ressourcen automatisieren.



Canary-Bereitstellung

Sicherheit

Sicherheit ist ein entscheidendes Element. Nachfolgend finden Sie eine Liste von Sicherheitsschritten, die der AWS CI/CD-Prozess bei der Bereitstellung einer Anwendung berücksichtigt.

- Quelle – Das dem Anbieter zugewiesene ECR-Repository ist so konfiguriert, dass das Flag „Scan on Push“ aktiviert ist und alle Uploads von Docker-Images sofort einem Sicherheitsscan unterzogen werden. Alle bekannten Schwachstellen und Sicherheitslücken (CVE) werden mit Benachrichtigungen gekennzeichnet. Abgesehen von ECR werden Anbieter, die Diagramme in das AWS CodeCommit-Repository stellen, aufgefordert, alle verwendeten Passwörter mit Secrets Manager und nicht als Klartext zu verschlüsseln.
- Integrität von Artefakten – Die in der Pipeline verwendeten Artefakte werden sowohl im Ruhezustand (mit AWS-verwalteten Schlüsseln) als auch bei der Übertragung (mit SSL/TLS) verschlüsselt.
- IAM-Benutzer und -Rollen – Die Berechtigungen, die Benutzern oder Ressourcen gewährt werden, basieren auf dem Prinzip der geringsten Rechte. Es sollte eine IAM-rollenübergreifende Vertrauensstellung bestehen, die möglicherweise konfiguriert werden muss, wenn Sie mit Ressourcen in verschiedenen Services arbeiten. AWS CodeBuild benötigt beispielsweise die Berechtigung, Befehle in einem Amazon EKS-Cluster auszuführen.
- Prüfung – Die von AWS CloudTrail angebotene Überwachungsfunktion verfolgt jeden einzelnen API-Aufruf über Services und Benutzeroperationen hinweg und ermöglicht die Auswertung vergangener Ereignisse.
- Image-Schwachstellenscan – in Amazon ECR hochgeladene CNF-Images werden automatisch auf Sicherheitslücken gescannt. Ein Bericht über die Scanergebnisse ist in der AWS Management Console verfügbar und kann auch über die API abgerufen werden. Die Ergebnisse können dann an die CSP-Betreiber weitergeleitet werden, damit sie Korrekturmaßnahmen ergreifen, einschließlich ein Austausch des CNF-Images.

Die Sicherheitsprüfungen finden in verschiedenen Phasen der Pipeline statt. Das neu hochgeladene Image muss sicher sein und die gewünschten Compliance-Prüfungen bestehen, bevor eine Benachrichtigung an die CSPs zur Genehmigung gesendet werden kann:

- Das Container-Registry sucht nach offenen CVE-Schwachstellen.
- Die Konfiguration wird während der Testphase auf Informationslecks sowie auf bekannte Muster in den persönlich identifizierbaren Informationen (PII) geprüft. Dabei werden Regeln zur Überprüfung

der Konformität für Probleme wie unerwartet offene TCP/UDP-Ports und DOS-Schwachstellen ausgelöst.

- Die Abwärts- und Aufwärtskompatibilität wird für die Sicherheit von Upgrades und Rollbacks überprüft.

Nicht nur die Sicherheit der Anwendung muss gewährleistet sein, sondern auch die der Pipeline. So muss die verschlüsselte Übertragung von Artefakten über alle Phasen hinweg sichergestellt sein – ob im Ruhezustand oder während der Übertragung.

Beobachtbarkeit

AWS bietet die Beobachtbarkeit der 5G-CNFs, die standardmäßig auf AWS bereitgestellt werden. Dies wird von Amazon CloudWatch ermöglicht. CloudWatch bietet vollständigen Überblick über Ihre Cloud-Ressourcen und -Anwendungen.

Amazon CloudWatch führt vier Hauptschritte während dieses Prozesses aus:

1. Erfassen – Erfassen von Metriken und Protokollen zu allen AWS-Ressourcen sowie Anwendungen und Services, die auf AWS- und lokalen Servern ausgeführt werden.
2. Überwachen – Visualisieren von Anwendungen und Infrastruktur mit CloudWatch-Dashboards, Korrelieren von Protokollen und Metriken zur Fehlersuche und Erstellen von Warnungen mit CloudWatch Alarms .
3. Reagieren – Automatisieren der Reaktion auf betriebsbezogene Veränderungen mit CloudWatch Events und AWS Auto Scaling.
4. Analysieren – Metriken bis zu einer Sekunde, erweiterte Datenaufbewahrung (15 Monate) und Echtzeit-Analyse mit CloudWatch Metric Math .

Der Amazon CloudWatch-Agent ist im Kubernetes-Cluster des Kunden installiert. Der Agent unterstützt die Konfigurations-, Erkennungs- und Metrik-Pull-Funktionen vonPrometheus. Sämtliche Prometheus-Metriken und -Metadaten mit hoher Genauigkeit werden angereichert und im Embedded Metric Format (EMF) in den CloudWatch Logs veröffentlicht.

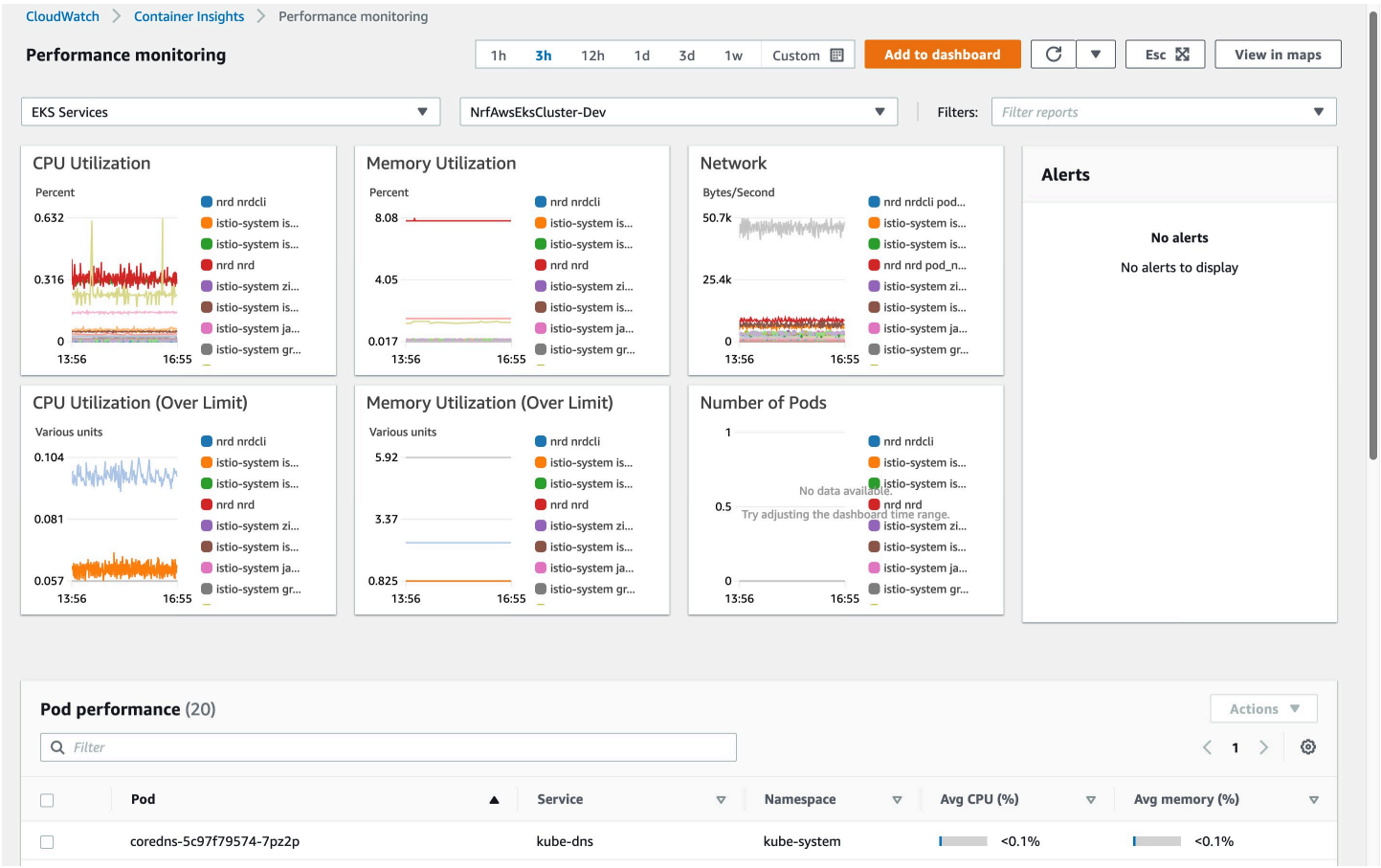
Amazon CloudWatch Container Insights automatisiert die Erkennung und Erfassung von Prometheus-Metriken aus containerisierten Anwendungen. Es sammelt, filtert und erstellt automatisch aggregierte benutzerdefinierte CloudWatch-Metriken, die in Dashboards visualisiert werden.

Jedes Ereignis erzeugt metrische Datenpunkte als benutzerdefinierte CloudWatch-Metriken für ein kuratiertes Set von metrischen Dimensionen, das vollständig konfigurierbar ist. Die Veröffentlichung aggregierter Prometheus-Metriken als benutzerdefinierte CloudWatch-Statistiken reduziert die Anzahl der Metriken, die zur Überwachung, Alarmierung und Fehlerbehebung bei Leistungsproblemen und -ausfällen benötigt werden. Sie können die Prometheus-Metriken mit hoher Genauigkeit auch mithilfe der Abfragesprache „CloudWatch Logs Insights“ analysieren, um bestimmte Pods und Labels zu isolieren, die sich auf Zustand und Leistung Ihrer containerisierten Umgebungen auswirken.

AWS CloudTrail bietet diese Sichtbarkeit und zeichnet jeden API-Aufruf serviceübergreifend auf. AWS Config bietet die Möglichkeit zur Compliance-Validierung. AWS bietet Kunden zusätzliche Überwachungsoptionen für Metriken, Protokolle, Anwendungsereignissen, Infrastruktur und Pipelines unter Verwendung verschiedener Services wie AWS X-Ray und AWS CloudTrail.

- AWS kann Open-Source-Metrik-Tools wie Prometheus, FluentD usw. nativ integrieren.
- Prometheus-Metriken können in Amazon CloudWatch oder OpenSearch Service für weitere Analysen aufgenommen werden.
- AWS verwendet FluentD als Standardmechanismus, um Protokolle von verschiedenen Systemen zu erfassen. Derselbe Mechanismus wird für dieses Projekt verwendet und konfiguriert.

Einzelheiten zur Konfiguration dieses Mechanismus finden Sie unter Einrichten von FluentD als DaemonSet zum Senden von Protokollen an CloudWatch Logs.



Beispiel für von Amazon CloudWatch überwachte Metriken

CI/CD-Orchestrierung mit Drittanbieter- und Open-Source-Tools

Die Orchestrierungsebene nutzt IaC zur Bereitstellung und Konfiguration der zugrunde liegenden Infrastruktur, die für den Betrieb der 5G-Netzwerkfunktionen erforderlich ist. Diese Ebene sollte modular, tragbar und wiederverwendbar sein.

Die Infrastruktur folgt bewährten cloudnativen Methoden und ist hochverfügbar, redundant und skalierbar.

Wie in den vorherigen Abschnitten gezeigt, kann die Bereitstellung der zugrundeliegenden Infrastruktur mit dem AWS Cloud Development Kit erfolgen. Dies kann mit Terraform von Hashicorp durchgeführt werden.

Terraform

Terraform ist ein Open-Source-IaC-Software-Tool, das einen konsistenten Befehlszeilenschnittstellen (CLI)-Workflow zur Verwaltung von Hunderten von Cloud-Services bietet. Terraform kodiert Cloud-APIs in deklarativen Konfigurationsdateien.

Für die Bereitstellung mit Terraform gelten die gleichen Grundsätze wie für das CDK. Der Code ist in Module gegliedert, die eine Anpassung und Wiederverwendung der Netzwerkkomponenten entsprechend den Anforderungen des Herstellers ermöglichen.

Die Konfiguration ist voll parametrisiert, sodass die Bereitstellungen vollständig an die Empfehlungen der Anbieter und ISVs angepasst werden können.

Die Bereitstellung der Netzwerkfunktionen ist in zwei Phasen unterteilt:

- Die erforderliche AWS-Infrastruktur wird über ein zentrales Repository erstellt und verwaltet.
- Die Konfiguration und der Code werden zentral in einem GitHub-Repository gespeichert.

Nachdem die Voraussetzungen geschaffen wurden, kann die Netzwerkfunktion mithilfe einer Anwendungspipeline, die in der vorherigen Phase festgelegt wurde, bereitgestellt werden.

Bereitstellung der Infrastruktur

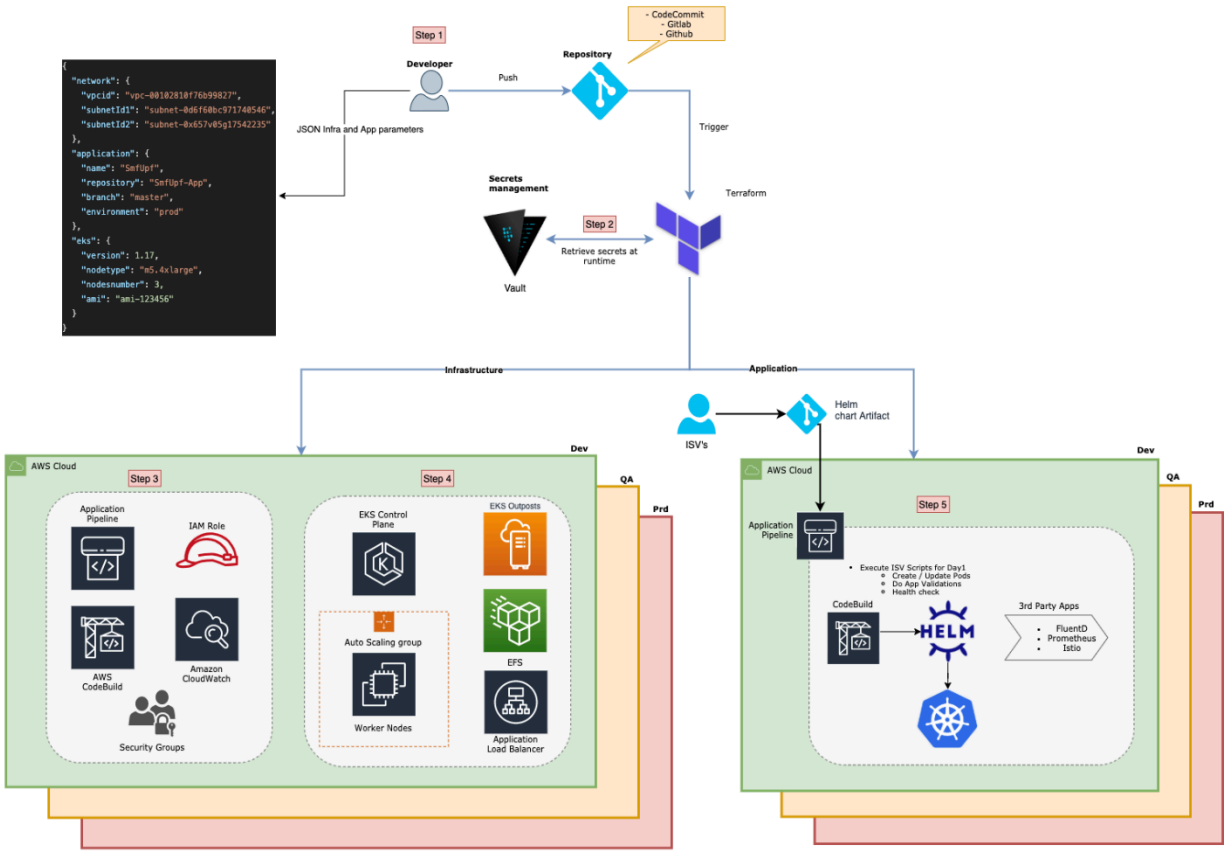
Für eine erfolgreiche Bereitstellung und Konfiguration der Netzwerkfunktion muss die bereitgestellte Infrastruktur alle Voraussetzungen erfüllen.

Einige der in dieser Phase zu erstellenden Komponenten sind:

- Netzwerk – VPC, öffentliche und private Subnetze, Routen, Load Balancer
- Datenverarbeitung – Kubernetes (Vmware Tanzu, Amazon EKS oder AWS Outposts), Primär- und Worker-Knoten der Amazon-EC2-Instances , Auto-Scaling-Gruppe
- Speicher – Amazon EFS, Amazon EBS, Amazon S3-Bucket
- Sicherheit – IAM-Rollen, Sicherheitsgruppen
- Pipeline – CodePipeline, CodeBuild
- Beobachtbarkeit – CloudWatch, Prometheus, FluentD

In der folgenden Abbildung wird die von Terraform orchestrierte Infrastruktursequenz erläutert:

1. Ein Entwickler füllt eine JSON-Datei, die in einem zentralen Repository gespeichert ist, mit dem IaC-Code. Die Datei enthält Informationen über die gewünschte Infrastrukturkonfiguration, z. B. die Größe der Instances, die Kubernetes-Version, Netzwerkinformationen und Details zum Anwendungs-Repository.
2. Zur Laufzeit werden Secrets aus HashiCorp Vault oder AWS Secrets Manager abgerufen.
3. Dann werden die Infrastrukturkomponenten (Netzwerke, Rechenleistung, Speicher und Sicherheit) bereitgestellt und konfiguriert.
4. Es wird ein Amazon EKS-Cluster mit Worker-Knoten bereitgestellt, der die Netzwerkfunktions-Pods hostet. Amazon EKS kann auch auf AWS Outposts bereitgestellt werden, um Workloads zu unterstützen, die die Nähe zu einem Rechenzentrum erfordern.
5. Eine Anwendungspipeline wird erstellt und so konfiguriert, dass sie auf Änderungen im Netzwerkfunktions-Repository wartet. Jedes Mal, wenn Code in die konfigurierte Repository-Verzweigung gesendet wird, löst die Pipeline automatisch die Entwicklung, den Test und die Bereitstellung der Netzwerkfunktion aus.
6. Beobachtbarkeits-Tools, die Protokolle und Metriken erfassen und zentralisieren, werden als Services in allen Knoten bereitgestellt und liefern Daten nahezu in Echtzeit, die in Grafana oder OpenSearch Dashboards visualisiert werden können.



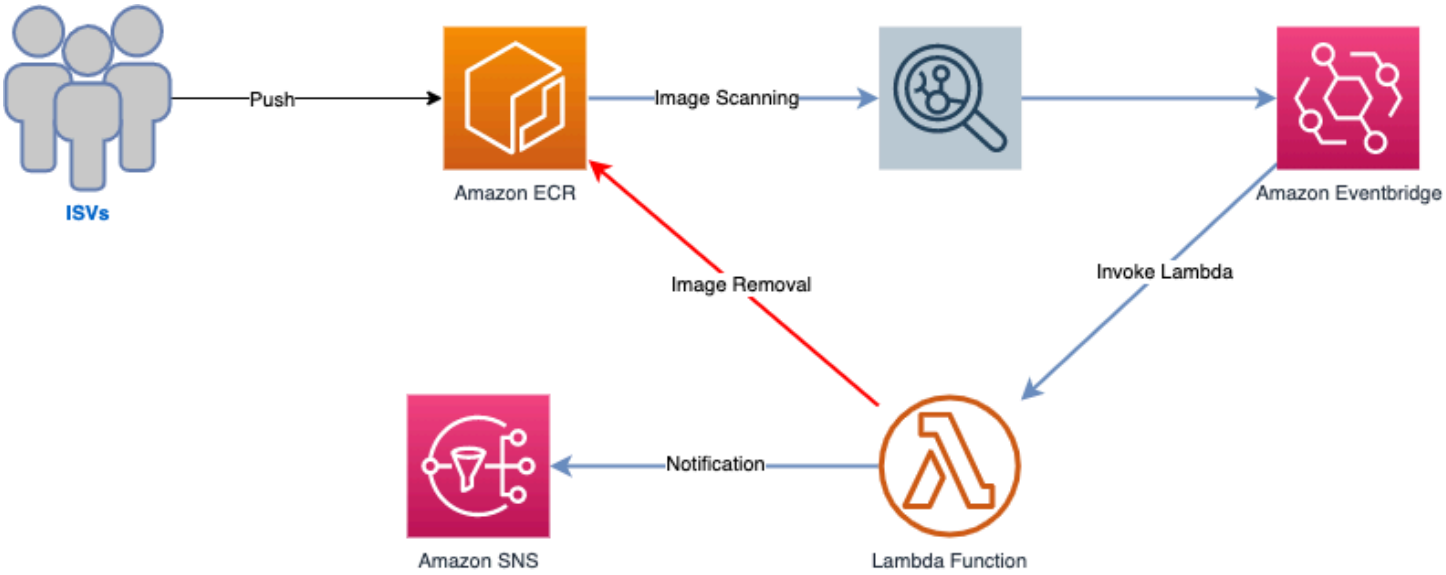
Bereitstellung und Konfiguration von Netzwerkfunktionen

Bereitstellung und Konfiguration von Netzwerkfunktionen

Die in der vorangegangenen Phase erstellte Pipeline ermöglicht es sowohl ISVs als auch Anbietern, die Bereitstellung von Netzwerkfunktionen zu dezentralisieren und zu optimieren. Die Pipeline ist verbunden und überwacht Änderungen im Anwendungs-Repository, das in der JSON-Datei in Schritt 1 der vorangegangenen Abbildung konfiguriert wurde.

Von Drittanbietern veröffentlichte Images werden mit einer bereitgestellten, konfigurierten Schwachstellenscan-Lösung überprüft, die eine Identifizierung von Software-Schwachstellen in Container-Images ermöglicht. Die Scanlösung prüft automatisch alle neuen Images, die an Amazon ECR übertragen wurden. Weitere Informationen zum ECR Image Scanning finden Sie unter Scannen von Images.

Die folgende Abbildung zeigt die Architektur der Image-Schwachstellenscan-Lösung.



Architektur der Image-Schwachstellenscan-Lösung

Die Anwendungspipeline kann so konfiguriert werden, dass sie durch Änderungen im Image nach dem Scan-Ergebnis oder durch direkte Änderungen im Repository ausgelöst wird. Beispielsweise, wenn ein neues Helm-Image erstellt wird.

In der folgenden Abbildung wird die Reihenfolge gezeigt, in der die Netzwerkfunktion erstellt/aktualisiert wird:

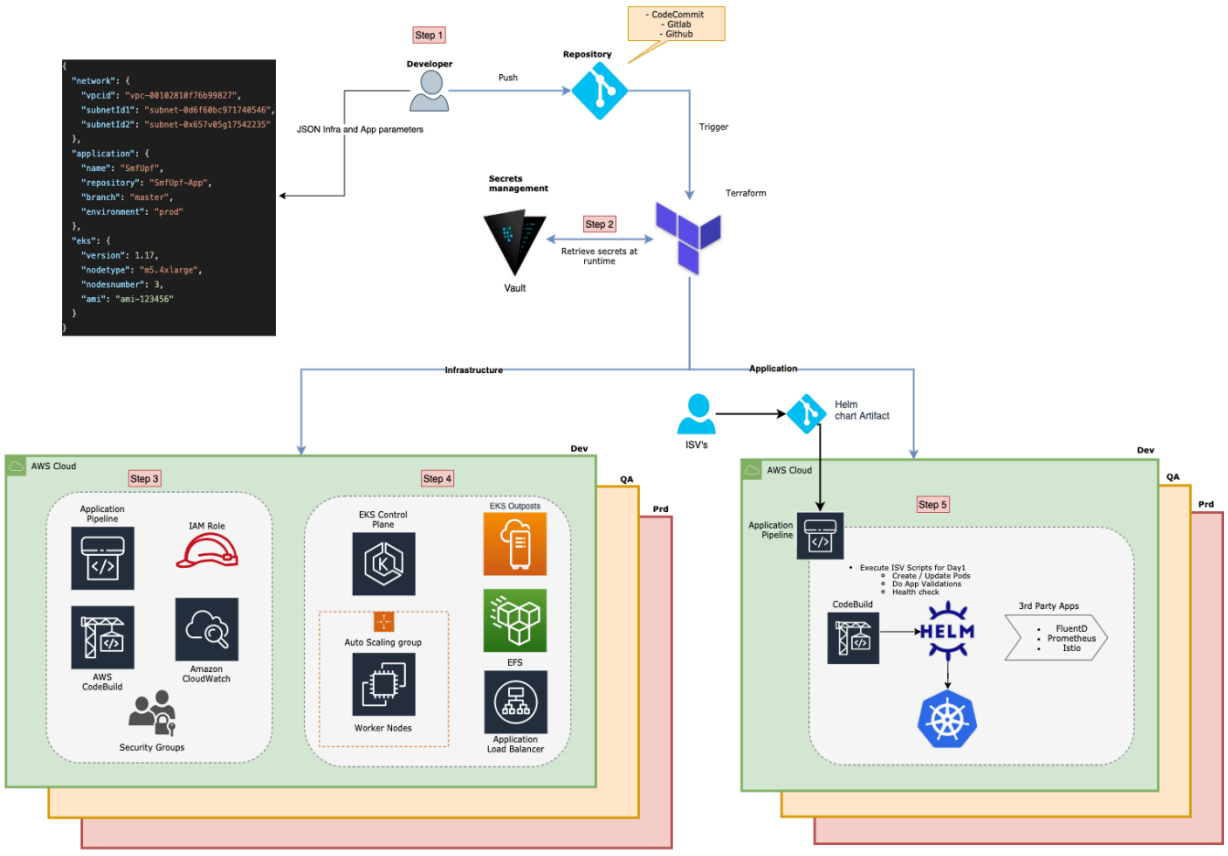
ISVs veröffentlichen neue Images auf Amazon ECR. (Bei Genehmigung des Images wird die Anwendungspipeline ausgelöst.)

CodePipeline ruft das neue Image aus Amazon ECR ab und verwendet CodeBuild, um das Image in Kubernetes bereitzustellen. Helm-Befehle können verwendet werden, um die Netzwerkfunktion zu aktualisieren.

Nachdem das Image bereitgestellt wurde, wird Test as Service (TaS) ausgelöst. TaS validiert die neue Bereitstellung und zentralisiert Daten und Metriken zur Leistung der Netzwerkfunktionen unter Belastung.

Protokolle und Metriken werden in OpenSearch und Grafana gesammelt und zentralisiert. Drittanbieter wie Datadog, Istio und Prometheus können ebenfalls konfiguriert werden, um zusätzliche Beobachtbarkeit bereitzustellen.

Es kann auch MANO zur Koordination von Netzwerkressourcen bereitgestellt und in die Lösung integriert werden. Die erfassten Daten werden verwendet, um automatisierte Maßnahmen wie Netzwerk-Slicing und automatische Skalierung des Quality of Service (QoS) durchzuführen.



Anwendungspipeline

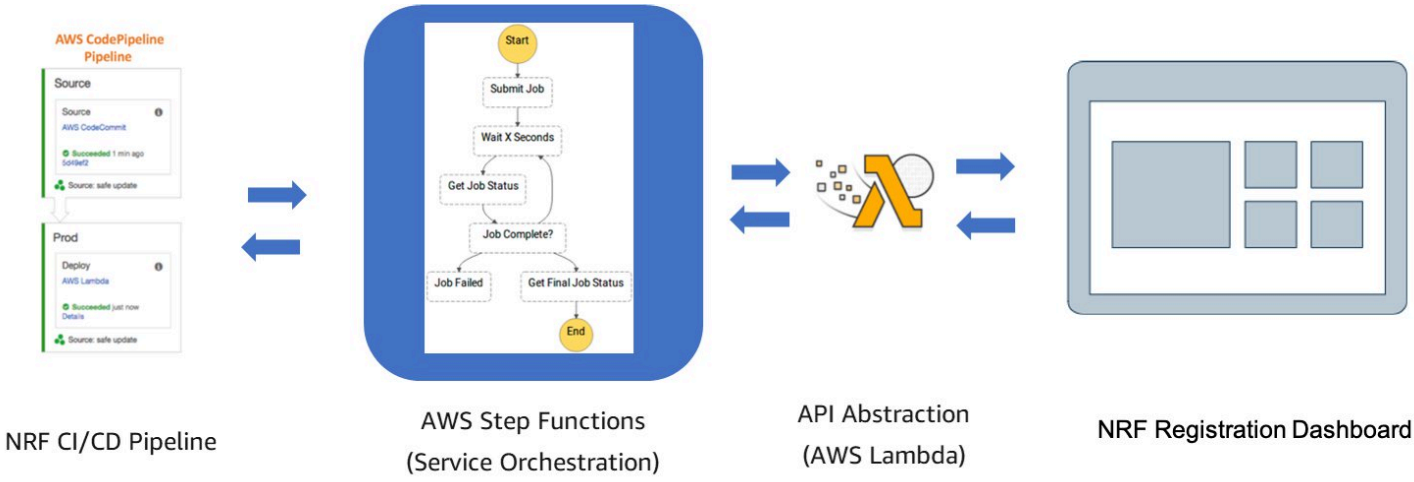
Tests

Das telekommunikationsspezifische Framework zur Testautomatisierung kann in die Code-Pipeline integriert werden. Die Code-Pipeline enthält Schrittfunktionen, um die Integration mit einem Framework zur Testautomatisierung zu orchestrieren. AWS Step Functions verwenden mehrere Lambda-Funktionen, um das Framework zur Testautomatisierung (Tools von Drittanbietern) über API-Aufrufe aufzurufen. Die Schrittfunktion erhält zunächst die Test-ID, führt den Test aus und erhält die Ergebnisse vom Framework zur Testautomatisierung. Dann analysiert die Schrittfunktion die Ergebnisse und leitet sie an die Code-Pipeline weiter, damit die Anwendung für die Produktionsbereitstellung freigegeben werden kann. Die Genehmigung kann je nach Bedarf automatisiert oder manuell in der Code-Pipeline gepflegt werden. Dies ist ein wichtiger Schritt für CSPs, um die Bereitstellung von der Testumgebung in die Produktion zu verlagern. Für die Integration erforderliche High-Level-APIs werden wie folgt kategorisiert:

- Kontext abrufen

- Einen bestimmten Testfall ausführen
- Testfall beenden
- Ergebnisse abrufen

Die Komplexität des Aufrufs externer REST-APIs wird mithilfe von AWS Step Functions modelliert, die Standardkonstrukte zum Aufrufen paralleler Abläufe, zum Warten auf Ergebnisse, zur Verzweigung auf der Grundlage von Bedingungen und zur Integration von REST-APIs mit AWS CodePipeline zulassen.



Testablauf

CI/CD und Orchestrierung

Continuous Integration und Continuous Delivery sind Teil einer übergreifenden Automatisierungsphilosophie, die sich auf cloudnative Architektur und deren Anwendung auf 5G bezieht. Die Orchestrierung ist ein weiterer Aspekt dieser Philosophie. Sie soll dynamisch und reaktiv auf alle Veränderungen im Netzwerk reagieren. Orchestrierung und CI/CD müssen eng miteinander verknüpft sein, um einen fehlerfreien Service zu gewährleisten und Serviceunterbrechungen zu minimieren. Die Integration von CI/CD und Orchestrierung sollte an zwei Fronten erfolgen:

- Das Anwenden von Patches und Upgrades im System muss so verwaltet und orchestriert werden, dass Unterbrechungen der laufenden Services auf ein Minimum reduziert werden. Die Orchestrierung kann beispielsweise dynamisch bestimmen, wann ein Update am besten durchgeführt werden sollte.
- Eine CI/CD-fähige Orchestrierung ermöglicht es, den Datenverkehr während der Bereitstellung von Upgrades auf der Grundlage der gewählten Bereitstellungsmodellstrategie (Canary, linear oder alles gleichzeitig) zu verschieben.

In der Regel werden Orchestrierungslösungen auf CI/CD-Pipelines aufgesetzt, damit die Orchestrierung Governance-Phasen in diese Pipelines einführen und in die laufenden Upgrade-Zyklen eingreifen kann.

Fazit

CI/CD bietet einen klaren und effizienten Weg für Entwickler und Anwendungsteams, um neuen Anwendungscode innerhalb weniger Minuten bereitzustellen. AWS verfügt über eine Vielzahl von Tools, die Entwicklern bei Integration, Testen und Bereitstellung von neuem Code helfen können, darunter AWS CodePipeline, AWS CodeCommit, AWS CodeBuild, AWS CodeDeploy und viele weitere. In diesem Dokument wurde untersucht, wie AWS-Services verwendet werden können, um einen CI/CD-Prozess für die Bereitstellung von 5G-Netzwerkfunktionen auf vollständig automatisierte Weise zu erstellen, einschließlich der verschiedenen Schritte, die für eine Bereitstellung von neuem Code erforderlich sind. Außerdem wurde untersucht, wie ein Framework eines Drittanbieters zur Testautomatisierung in den CI/CD-Prozess integriert werden kann und wie Drittanbieter-Tools wie Terraform eingesetzt werden können.

Mitwirkende

An diesem Dokument haben folgende Personen mitgewirkt:

- 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

Dokumentversionen

Abonnieren Sie den RSS-Feed, um über Aktualisierungen des Whitepapers benachrichtigt zu werden.

Update-Historie-Änderung	Update-Historie-Beschreibung	update-history-date
Erstveröffentlichung	Erstveröffentlichung des Whitepapers	8. März 2021

Weitere Informationen

Weitere Informationen finden Sie unter:

- Continuous Integration und Continuous Delivery auf AWS (Whitepaper)
- Carrier-Grade Mobile Packet Core Network auf AWS (Whitepaper)
- 5G-Netzwerkentwicklung mit AWS (Whitepaper)

Akronyme

- AMF – Access & Mobility Management Function (Zugriffs- und Mobilitätsmanagementfunktion)
- API – Application Programming Interface (Anwendungsprogrammierschnittstelle)
- AUSF – Authentication Server Function (Authentifizierungsserverfunkion)
- BSS – Business Support System
- CDK – Cloud Development Kit
- CI/CD – Continuous Integration und Continuous Delivery
- CLI – Command Line Interface (Befehlszeilenschnittstelle)
- CNF – Cloud-native or Containerized Network Function (cloudnative oder containerisierte Netzwerkfunktion)
- CSP – Communication Service Provider (Kommunikationsserviceanbieter)
- CU – RAN Central Unit (RAN-Zentraleinheit)
- CVE – Common Vulnerabilities and Exposures (Häufige Schwachstellen und Sicherheitslücken)
- DoS – Denial of Service (Dienstleistungsverhinderung)
- DR – Disaster Recovery (Notfallwiederherstellung)
- DU – RAN Distributed Unit (Verteilte RAN-Einheit)
- E2E – End-to-End
- ECR – Elastic Container Registry
- EFS – Elastic File System
- EKS – Elastic Kubernetes Service
- EPC – Evolved Packet Core (Entwickelter Paketkern)
- IaC – Infrastructure as Code (Infrastruktur als Code)
- ISV – Independent Software Vendor (Unabhängiger Softwareanbieter)
- MANO – Management and Orchestration (Verwaltung und Orchestrierung)
- MEC – Multi-Access Edge Computing
- NACL – Network Access Control List (Netzwerkzugriffskontrolllisten)
- NAT – Network Address Translation (Netzwerkadressenübersetzung)
- NF – Network Function (Netzwerkfunktion)
- NFV – Network Function Virtualization (Virtualisierung von Netzwerkfunktionen)

- NFVO – Network Function Virtualization Orchestrator (Orchestrator zur Virtualisierung von Netzwerkfunktionen)
- NOC – Network Operations Centre (Zentrum für Netzwerkabläufe)
- NRF – Network Repository Function (Netzwerk-Repository-Funktion)
- OSS – Operations Support System (Betriebssupportsystem)
- PII – Personally Identifiable Information (Personenbezogene Daten)
- QoS – Quality of Service (Servicequalität)
- RAN – Radio Access Network (Funkzugangsnetz)
- SBI – Service Based Interface (Servicebasierte Schnittstelle)
- SMF – Session Management Function (Sitzungsverwaltungsfunktion)
- SSL – Secure Sockets Layer
- TaS – Test as Service (Test als Service)
- TCP – Transmission Control Protocol (Protokoll zur Übertragungssteuerung)
- TLS – Transport Layer Security
- UDM – Unified Data Management (Einheitliche Datenverwaltung)
- UDP – User Datagram Protocol
- UPF – User Plane Function (Benutzerebenenfunktion)
- VIM – Virtualized Infrastructure Manager (Virtualisierter Infrastrukturmanager)
- VNF – Virtual Network Function (Virtuelle Netzwerkfunktion)
- VPC – Virtual Private Cloud

Hinweise

Kunden sind eigenverantwortlich für die unabhängige Bewertung der Informationen in diesem Dokument zuständig. Dieses Dokument: (a) dient rein zu Informationszwecken, (b) spiegelt die aktuellen Produktangebote und Verfahren von AWS wider, die sich ohne vorherige Mitteilung ändern können, und (c) impliziert keinerlei Verpflichtungen oder Zusicherungen seitens AWS und dessen Tochtergesellschaften, Lieferanten oder Lizenzgebern. AWS-Produkte oder -Services werden im vorliegenden Zustand und ohne ausdrückliche oder stillschweigende Gewährleistungen, Zusicherungen oder Bedingungen bereitgestellt. Die Verantwortung und Haftung von AWS gegenüber seinen Kunden wird durch AWS-Vereinbarungen geregelt. Dieses Dokument ist weder ganz noch teilweise Teil der Vereinbarungen zwischen AWS und seinen Kunden und ändert diese Vereinbarungen auch nicht.