



Livre blanc AWS

# Intégration et livraison continues pour les réseaux 5G sur AWS



# Intégration et livraison continues pour les réseaux 5G sur AWS: Livre blanc AWS

Copyright © Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Les marques et l'habillage commerciaux d'Amazon ne peuvent pas être utilisés en connexion avec un produit ou un service qui n'est pas celui d'Amazon, d'une manière susceptible de causer de la confusion chez les clients ou d'une manière qui dénigre ou discrédite Amazon. Toutes les autres marques commerciales qui ne sont pas la propriété d'Amazon sont la propriété de leurs propriétaires respectifs, qui peuvent ou non être affiliés ou connectés à Amazon, ou sponsorisés par Amazon.

---

# Table of Contents

Résumé .....	i
Résumé .....	1
Introduction .....	2
Intégration et livraison continues (CI/CD) .....	4
Intégration continue .....	4
Livraison et déploiement continus .....	4
Infrastructure as Code .....	4
CI/CD sur AWS .....	5
Réseaux 5G sur AWS .....	9
CI/CD dans les réseaux 5G .....	9
Étapes détaillées de l'intégration continue/livraison continue (CI/CD) .....	12
Configuration du réseau .....	12
Déploiement d'infrastructure .....	12
Déploiement de fonctions de réseau natif cloud .....	13
Livraison continue de fonctions CNF .....	15
Sécurité .....	18
Observabilité .....	19
Orchestration CI/CD avec des outils tiers et open source .....	22
Terraform .....	22
Déploiement d'infrastructure .....	22
Déploiement et configuration des fonctions réseau .....	24
Tests .....	26
CI/CD et orchestration .....	28
Conclusion .....	29
Participants .....	30
Révisions du document .....	31
Autres lectures .....	32
Acronymes .....	33
Mentions légales .....	35

# Intégration et livraison continues pour les réseaux 5G sur AWS

Date de publication : 8 mars 2021 ([Révisions du document](#))

## Résumé

Ce livre blanc présente l'intégration et la livraison continues (CI/CD) pour les réseaux 5G, ainsi que la façon dont les outils et les services d'Amazon Web Services (AWS) peuvent être utilisés pour automatiser entièrement le déploiement et les mises à jour des fonctions de réseau 5G. Le livre blanc décrit de façon détaillée les différentes étapes de CI/CD pour les fonctions réseau 5G, y compris la configuration du réseau, le déploiement de l'infrastructure, le déploiement des fonctions de réseau natif cloud et les mises à jour continues des fonctions réseau. Il fournit également des détails sur l'intégration à des outils open source et tiers pour les tests, l'observabilité et l'orchestration.

Ce livre blanc s'adresse aux fournisseurs de services de communication (CSP) ainsi qu'aux fournisseurs indépendants de logiciel (FIL).

# Introduction

Historiquement, le développement, les tests d'intégration en laboratoire et sur le terrain et le déploiement en production de nouveaux nœuds de réseau ou de nouvelles fonctions dans un réseau cellulaire prenaient des semaines, voire des mois, pour s'assurer de la stabilité des services de télécommunications essentiels à la mission et à l'entreprise. Ce long cycle de déploiement était dû à l'architecture monolithique des nœuds de réseau traditionnels, à un environnement multifournisseur et à de nombreuses interfaces point à point entre les entités des réseaux mobiles 2G, 3G et 4G.

Comme présenté dans le livre blanc traitant de [l'évolution du réseau 5G avec AWS](#), les réseaux mobiles 5G, tels que standardisés par le 3GPP, prennent désormais en charge une architecture de réseau natif cloud rendue possible par la virtualisation et la conteneurisation. Plus précisément, les réseaux 5G introduisent et prennent en charge un nouveau paradigme d'architecture de microservices, sans état et basée sur les services.

Cette architecture 5G signifie que différentes fonctions réseau peuvent fonctionner comme des services indépendants faiblement couplés qui communiquent entre eux au moyen d'interfaces et d'API bien définies. Plus important encore, chaque fonction réseau peut être mise à jour indépendamment. Ce changement d'architecture dans la 5G permet aux fournisseurs de services de communication de profiter d'une plus grande agilité et efficacité opérationnelle, en facilitant le déploiement plus fréquent de mises à jour des fonctions réseau, tout en maintenant les tests, les exigences de sécurité et les normes grâce à l'automatisation.

L'intégration et le déploiement de nouvelles fonctions pour un fournisseur de services de communication commencent généralement lorsque le fournisseur de fonctions réseau publie un nouveau package logiciel de fonctions réseau, tel qu'une image [Docker](#) dans une fonction réseau basée sur un conteneur, ou un nouveau fichier de configuration, tel que les Charts de [Helm](#) dans le cas d'une application [Kubernetes](#). (Les Charts de Helm sont un ensemble de fichiers qui décrivent un ensemble associé de ressources Kubernetes).

L'idée d'utiliser le paradigme de CI/CD pour le déploiement de fonctions réseau 5G gagne du terrain, mais la réalisation pratique de cette idée a représenté un défi dans l'industrie des télécommunications.

AWS a été le pionnier du développement de nouveaux outils CI/CD pour la livraison de logiciels afin d'aider un large éventail de secteurs à développer et à déployer rapidement des modifications logicielles, tout en préservant la stabilité et la sécurité des systèmes. Ces outils incluent un ensemble

de services de développement et d'exploitation de logiciels (DevOps) tels qu'[AWS CodeStar](#), [CodeCommit](#), [CodePipeline](#), [CodeBuild](#) et [CodeDeploy](#).

AWS promeut également l'idée d'infrastructure IaC (Infrastructure as Code) à l'aide d'[AWS Cloud Development Kit](#) (AWS CDK), d'[AWS CloudFormation](#) et d'outils tiers basés sur des API tels que [Terraform](#). À l'aide de ces outils, AWS peut stocker les processus de déploiement de la fonction réseau au sein d'AWS en tant que code source, et maintenir ce code source IaC dans le pipeline CI/CD afin de réaliser une livraison continue.

Ce livre blanc décrit les processus détaillés permettant de tirer parti des outils IaC et CI/CD AWS pour le déploiement et la mise à jour de la fonction réseau 5G. En outre, ce livre blanc traite de l'intégration à des outils tiers pour les tests, l'observabilité et l'orchestration.

Les outils CI/CD AWS ne sont pas limités aux fonctions réseau 5G. Ils sont également utilisés pour automatiser le déploiement des réseaux 4G, ce qui permet aux fournisseurs de services de communication de déployer et de mettre à jour rapidement et efficacement les fonctions du réseau 4G. La plupart des fonctions réseau 4G reposent sur la fonction de réseau virtuel (VNF). Les ensembles d'outils CI/CD AWS tels qu'AWS CloudFormation peuvent être utilisés afin d'automatiser le déploiement de VNF 4G, apportant évolutivité et rapidité pour les déploiements de réseaux 4G.

# Intégration et livraison continues (CI/CD)

## Intégration continue

L'intégration continue (CI) est un processus logiciel au cours duquel les développeurs envoient régulièrement leur code dans un référentiel central tel que [AWS CodeCommit](#) ou [GitHub](#). Chaque envoi push de code déclenche une génération automatique, suivie de l'exécution de tests. La CI a pour principal objectif de détecter les problèmes de code à un stade précoce, d'améliorer la qualité du code et de réduire le temps nécessaire à la validation et à la publication de nouvelles mises à jour logicielles.

## Livraison et déploiement continus

La livraison continue (CD) est un processus logiciel au cours duquel les artefacts sont déployés dans l'environnement de test, l'environnement intermédiaire et l'environnement de production. Elle peut être entièrement automatisée ou comporter des étapes d'approbation à des points critiques. Ainsi, il est garanti que toutes les approbations requises avant le déploiement, telles que l'approbation de la gestion des mises en production, sont en place. Quand la livraison continue est correctement mise en œuvre, les développeurs disposent toujours d'un artefact de génération prêt au déploiement ayant suivi un processus de test standardisé.

Grâce au déploiement continu, les révisions sont déployées automatiquement vers un environnement de production, sans nécessiter d'approbation explicite de la part d'un développeur. Le processus de publication de logiciel est alors entièrement automatisé. Ainsi, il est possible de créer une boucle de rétroaction continue des clients dès le début du cycle de vie du produit.

Grâce au déploiement continu, chaque modification validée qui réussit les tests automatisés est automatiquement mise en production. La livraison continue n'a pas pour objectif de mettre immédiatement en production chaque modification validée et ayant réussi les tests automatisés, mais bien de s'assurer que chaque modification peut être mise en production.

## Infrastructure as Code

Comme indiqué dans le livre blanc sur [l'évolution du réseau 5G avec AWS](#), l'IaC est un élément clé pour automatiser le processus d'allocation et la gestion du cycle de vie de l'application et de son environnement. Plutôt que de se fier à des étapes effectuées manuellement, les administrateurs réseau/informatiques et les développeurs peuvent instancier l'infrastructure à l'aide de fichiers de

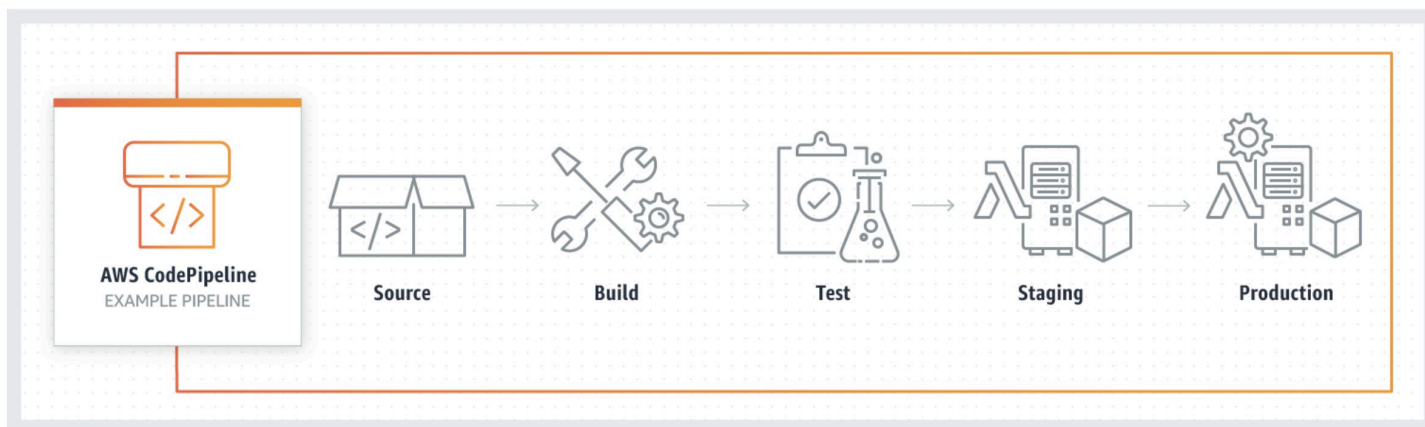
configuration. L'IaC traite ces fichiers de configuration comme du code logiciel. Ces fichiers peuvent être utilisés pour produire un ensemble d'artefacts : à savoir les services de calcul, de stockage, de réseau et d'application qui constituent un environnement d'exploitation. L'IaC élimine la dérive de configuration grâce à l'automatisation, augmentant ainsi la vitesse et l'agilité des déploiements d'infrastructure.

Dans le cas de la mise en œuvre de la virtualisation des fonctions réseau (NFV) sur AWS, ce cadre IaC apporte une valeur du point de vue de l'orchestration. De la création du Virtual Private Cloud (VPC) au déploiement des fonctions réseau, chaque étape peut être programmée, gérée en tant que code source et tenue à jour à l'aide du contrôle de version dans [AWS CodeCommit](#).

Ce cadre IaC pour la fonction réseau permet la création et le déploiement d'une infrastructure et d'une fonction réseau reproductibles et fiables, qui peuvent être étendus à l'automatisation de bout en bout (E2E) de la gestion des tranches de réseau et de la gestion du cycle de vie des services. AWS fournit un ensemble d'outils complets pour créer, maintenir et déployer une infrastructure par programmation de manière descriptive et déclarative, en utilisant des services tels que AWS CloudFormation, AWS CDK, AWS CDK pour Kubernetes, et l'exposition aux API de tous les services AWS.

## CI/CD sur AWS

L'intégration continue/livraison continue (CI/CD) peut être représentée comme un pipeline, où le nouveau code est envoyé d'un côté, testé au cours d'une série de phases (source, génération, test, intermédiaire et production), puis publié en tant que code prêt pour la production.



### Présentation du pipeline CI/CD

Chaque phase du pipeline CI/CD est structurée comme une unité logique dans le processus de livraison. Chaque phase agit comme une barrière qui vérifie un certain aspect du code. Au fur et à



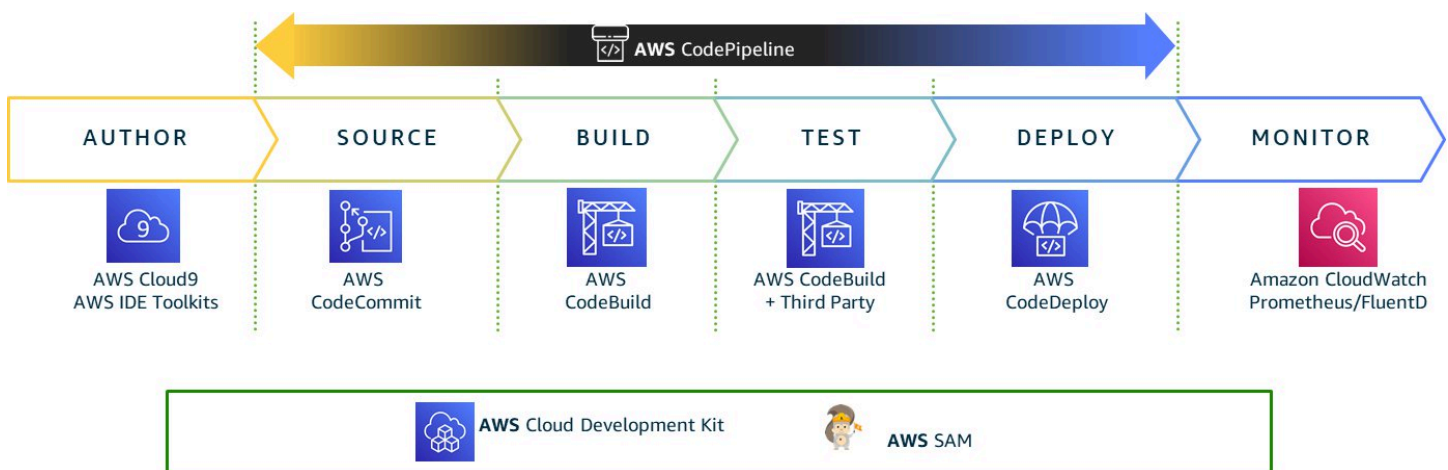
mesure que le code progresse dans le pipeline, on suppose que le code est de meilleure qualité aux phases ultérieures, car d'autres aspects de ce code continuent d'être vérifiés. Les problèmes décelés à un stade précoce empêchent la progression du code dans le pipeline. Les résultats des tests sont immédiatement envoyés à l'équipe, et toutes les versions et tous les lancements sont interrompus si le logiciel échoue à une phase.

AWS propose un ensemble complet d'outils de développement CI/CD permettant d'accélérer les cycles de développement et de publication de logiciels. [AWS CodePipeline](#) automatise les phases de génération, de test et de déploiement du processus de publication chaque fois qu'il y a un changement de code, en fonction du modèle de publication défini. Il en résulte une livraison rapide et fiable des fonctions et des mises à jour.

Les pipelines de code peuvent s'intégrer à d'autres services. Il peut s'agir de services AWS, tels qu'[Amazon Simple Storage Service](#) (Amazon S3), ou de produits tiers, tel que GitHub.

AWS CodePipeline peut répondre à divers cas d'utilisation liés au développement et à l'exploitation, notamment :

- Compilation, génération et test du code avec [AWS CodeBuild](#)
- Livraison continue au cloud d'applications basées sur un conteneur
- Validation avant le déploiement des artefacts (tels que les descripteurs et les images de conteneur) requis pour le service réseau ou des fonctions de réseau natif cloud spécifiques
- Tests fonctionnels, d'intégration et de performances pour la fonction réseau en conteneur/fonction de réseau virtuel (CNF/VNF), y compris les tests de référence et de régression
- Tests de fiabilité et de reprise après sinistre



## Composants du pipeline CI/CD AWS

AWS peut configurer des pipelines CI/CD à l'aide des outils pour développeur AWS suivants :

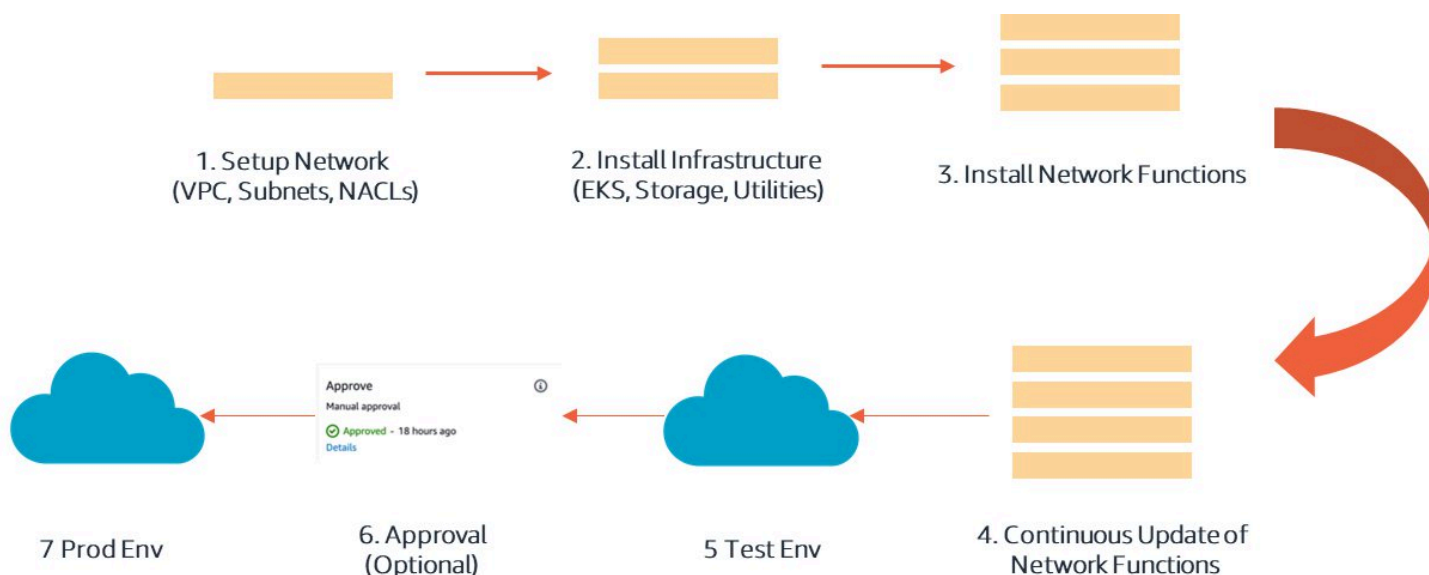
- [AWS CodeCommit](#)
- [AWS CodeBuild](#)
- [AWS CodePipeline](#)
- [AWS CodeDeploy](#)
- [Amazon Elastic Container Registry](#)
- [AWS CodeStar](#)

La création d'un pipeline CI/CD peut être automatisée à l'aide d'[AWS CDK](#) et d'[AWS CloudFormation](#). Dans le domaine de la virtualisation des fonctions réseau, cette automatisation native AWS peut être intégrée dans un cadre de gestion et d'orchestration (MANO) et dans le cadre d'orchestration des services du fournisseur de services de communication.

Le processus CI/CD comprend les étapes suivantes :

- Configuration du réseau : AWS CDK et AWS CloudFormation initient la création des prérequis réseau :
  - Pile de mise en réseau (VPC, sous-réseaux, passerelle NAT (traduction d'adresses réseau), table de routage et passerelle Internet)
- Déploiement de l'infrastructure : AWS CDK et AWS CloudFormation lancent la création des piles de ressources suivantes :
  - Pile de calculs (création de clusters [Amazon Elastic Kubernetes Service](#) (Amazon EKS), nœuds worker EKS, [AWS Lambda](#))
  - Pile de stockage (compartiments Amazon S3, volumes [Amazon Elastic Block Store](#) (Amazon EBS) et [Amazon Elastic File System](#) (Amazon EFS))
  - Pile de surveillance ([CloudWatch](#), [Amazon OpenSearch Service](#) (OpenSearch Service))
  - Pile de sécurité ([AWS Identity and Access Management](#) (AWS IAM), groupes de sécurité [Amazon Elastic Compute Cloud](#) (Amazon EC2), [listes de contrôle d'accès réseau](#) VPC (NACL))

- Déploiement de la fonction réseau cloud (CNF) : à ce stade, la CNF est déployée sur des clusters EKS à l'aide des outils [Kubectl](#) et des Charts de Helm. Cette phase déploie également toutes les applications ou tous les outils spécifiques nécessaires aux fonctions CNF pour fonctionner efficacement (tels que [Prometheus](#) ou [Fluentd](#)). Les fonctions CNF peuvent être déployées au moyen de fonctions Lambda ou avec AWS CodeBuild.
- Mises à jour et déploiement continus : il s'agit d'une séquence d'étapes qui sont effectuées de manière itérative pour déployer les modifications qui font partie des modifications de conteneur/ de configuration entraînant des mises à niveau. Comme dans le cas du déploiement de fonctions CNF, les mises à jour et le déploiement continus peuvent être automatisés à l'aide des services AWS, avec le déclencheur d'[AWS CodeCommit](#), d'[Amazon Elastic Container Registry](#) (Amazon ECR) ou d'un système source tiers tel que [GitLab Webhooks](#).



### Diagramme de flux du pipeline CI/CD AWS

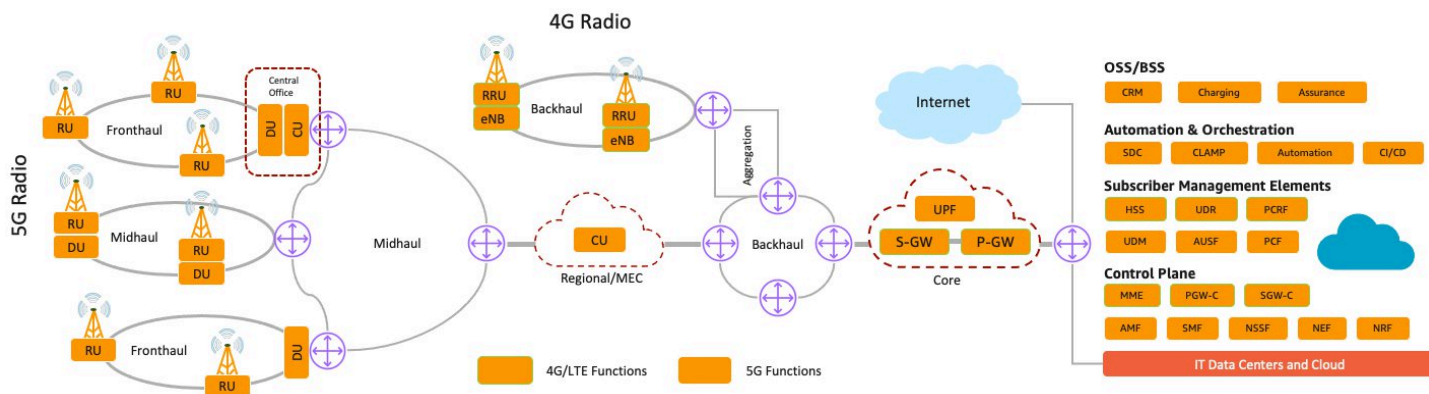
Le pipeline CI/CD est généré à l'aide d'[AWS CodePipeline](#) et utilise un service de livraison continue qui modélise, visualise et automatise les phases nécessaires à la publication du logiciel. En définissant des étapes dans un pipeline, vous pouvez récupérer du code à partir d'un référentiel de code source, générer ce code source dans un artefact publiable, tester l'artefact et le déployer en production. Seul le code qui réussit toutes ces étapes sera déployé. Vous pouvez éventuellement ajouter d'autres exigences à votre pipeline, telles que des approbations manuelles, pour garantir que seules les modifications approuvées sont déployées en production.

# Réseaux 5G sur AWS

Le modèle type de l'infrastructure réseau 5G est composé d'un site radio 4G/5G, d'un réseau fronthaul/midhaul/backhaul, d'un site de réseau principal et d'un centre de données de télécommunications/informatique. Les fournisseurs de services de communication peuvent utiliser les services AWS pour créer une infrastructure réseau 5G évolutive et flexible tout en réduisant les coûts d'investissement initiaux. AWS peut être utilisé pour mettre en œuvre le centre d'exploitation de réseau (NOC) virtuel dans la région qui héberge le système de support opérationnel/système de support pour les entreprises (OSS/BSS) et la majorité des fonctions du réseau principal du plan de contrôle.

AWS peut également être utilisé pour mettre en œuvre le bureau central local (CO) ou le centre de données distribué avec une flotte d'instances [AWS Outposts](#) qui hébergent principalement des fonctions de plan utilisateur telles que UPF (fonction de plan utilisateur), unité centrale RAN et Multi-Access Edge Computing (MEC). Une explication plus détaillée de l'architecture de référence et des avantages de la mise en œuvre du réseau 5G sur AWS est proposée dans le livre blanc relatif à [l'évolution du réseau 5G sur AWS](#).

Au moment de mettre en œuvre le réseau 5G sur AWS, les outils CI/CD AWS, qui sont présentés dans les sections subséquentes de ce livre blanc, peuvent faciliter l'automatisation complète du déploiement, de la mise à niveau et de la gestion du cycle de vie des fonctions réseau 5G.

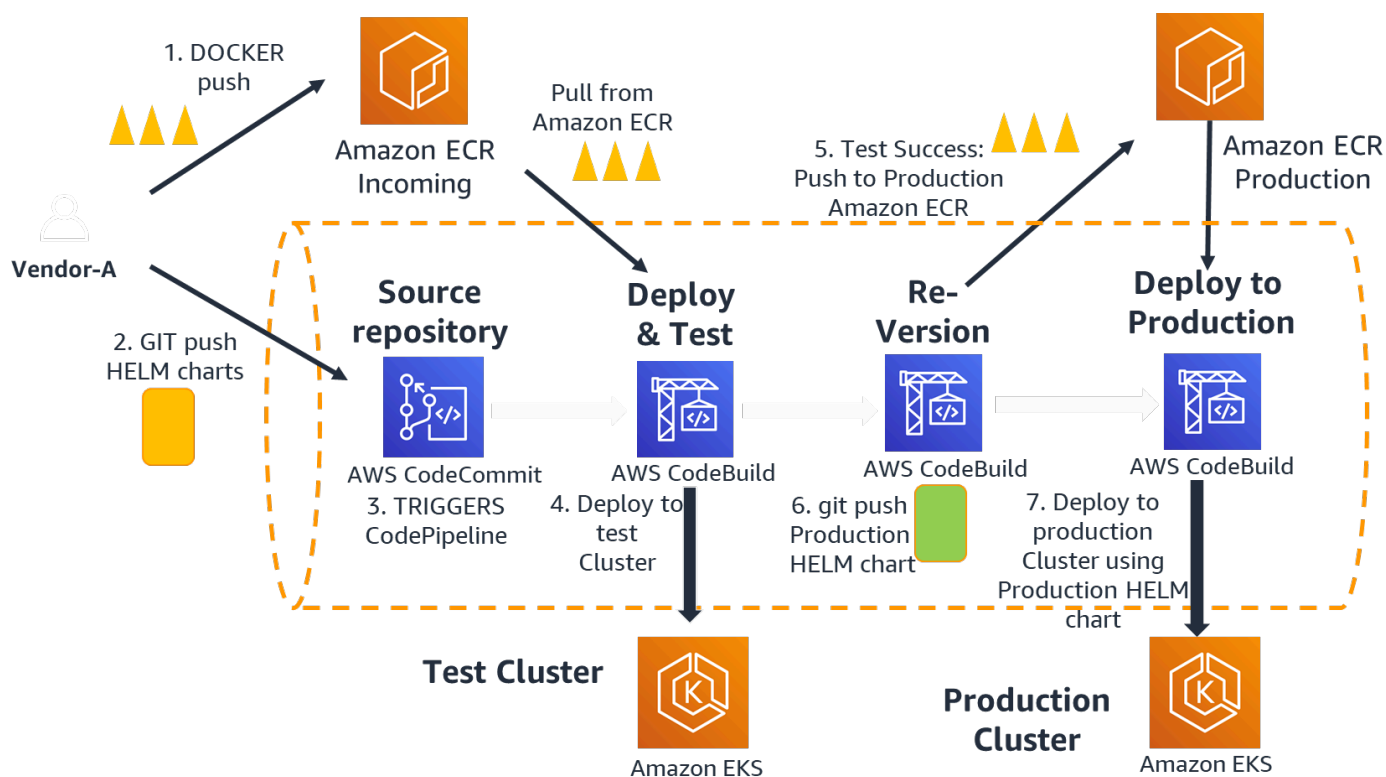


## Architecture E2E du réseau 5G

## CI/CD dans les réseaux 5G

La construction de conception de l'infrastructure est stockée sous forme de code utilisant un langage déclaratif. Le fournisseur de services de communication dispose ainsi d'une reproduction reproductible de l'infrastructure avec le même comportement attendu que nécessaire. Le code est

conservé dans le référentiel de code et un pipeline est configuré pour orchestrer les mises à jour des piles déployées (par exemple, AWS CDK et AWS CloudFormation). AWS peut aider à créer une infrastructure IaC pour une intégration agile des fonctions des fournisseurs indépendants de logiciel (FIL).



## Flux de pipeline de code

Les changements dans les configurations de fonctions de réseau natif cloud au moyen des Charts de Helm sont considérés comme des déclencheurs dans le cadre de l'exécution automatique d'un pipeline CI/CD pour les fonctions réseau.

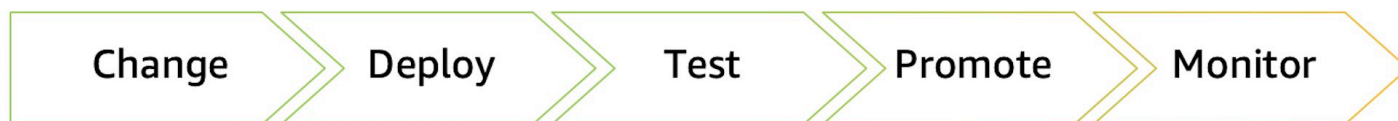
AWS CodeCommit peut être utilisé pour gérer les fichiers de configuration, et Amazon ECR pour conserver les images de conteneur.

Comme indiqué dans la figure du flux du pipeline de code, lorsque le fournisseur indépendant de logiciel envoie de nouvelles modifications de code dans le référentiel de code (Charts de Helm, fichiers de configuration ou fichier de propriétés), le pipeline de code est déclenché. Le pipeline de code extrait l'image d'ECR et utilise les Charts de Helm pour déployer l'application. Les nouveaux tests d'application peuvent être intégrés à l'infrastructure d'automatisation des tests tiers. En fonction du résultat, les fournisseurs de services de communication peuvent approuver le déploiement de production.

La phase source de CodePipeline recherche les modifications apportées aux fichiers de configuration. CodeCommit, Amazon S3, GitHub ou AWS CloudFormation sont des fournisseurs valides pour la phase source. Des systèmes sources alternatifs peuvent être intégrés en utilisant les fonctions Lambda pour implémenter des Webhooks, ce qui permet une intégration basée sur les événements entre Gitlab et AWS CodePipeline. Consultez les liens suivants pour obtenir un guide d'implémentation détaillé.

- [Webhooks avec GitLab](#)
- [Intégrations de registre de conteneurs](#)

La conception du pipeline CI/CD doit tenir compte des étapes de déploiement critiques telles que le déploiement initial, les tests et la promotion en production après que les résultats des tests ont été alignés sur les attentes et vérifiés par rapport à la base de référence. Chaque étape du processus de pipeline fournit des artefacts de données, qui permettent des comparaisons et des décisions orientées données.



### Étapes du pipeline CI/CD d'application

Chaque étape peut être considérée comme une tâche distincte, ce qui permet d'intégrer des flux de travail de validation et de déploiement adéquats pour prendre en charge le service réseau et les fonctions de réseau natif cloud. Les tâches d'exécution peuvent intégrer des outils tiers supplémentaires tels que des générateurs de trafic et des simulateurs, permettant une validation de bout en bout des services réseau.

AWS fournit un service [AWS Step Function](#) (machine d'état native cloud) sophistiqué qui s'intègre de manière native à d'autres services AWS, et a également la capacité de s'intégrer à des systèmes externes tels que Jira ou à une infrastructure d'automatisation des tests.

# Étapes détaillées de l'intégration continue/livraison continue (CI/CD)

L'intégration continue/livraison continue (CI/CD) peut être représentée comme un pipeline, où le nouveau code est envoyé d'un côté, testé au cours d'une série de phases (source, génération, test, intermédiaire et production), puis publié en tant que code prêt pour la production.

Les phases de déploiement et de test sont énumérées ci-dessous. Le déploiement et la configuration peuvent être divisés en quatre sections principales :

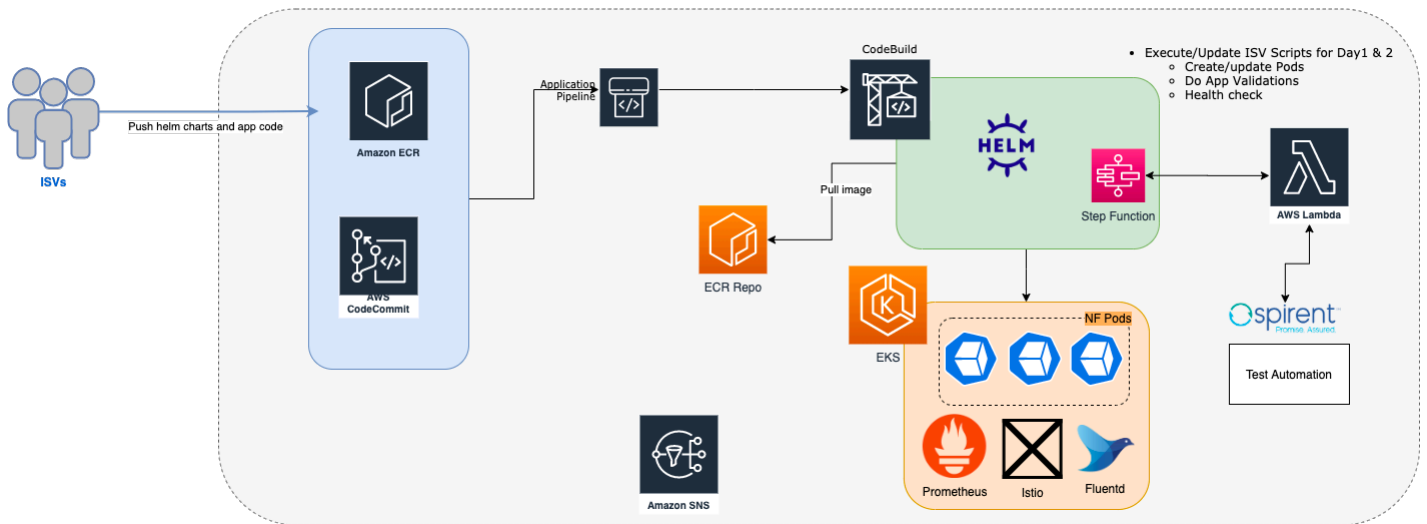
- Configuration du réseau
- Déploiement d'infrastructure
- Déploiement de fonctions de réseau natif cloud
- Livraison continue de fonctions CNF

## Configuration du réseau

Concentrez-vous sur la configuration des conditions préalables à l'infrastructure. Cela inclut la création du VPC, des réseaux, des sous-réseaux, des NACL, etc. Concevez le plan de réseau IP en tenant compte des fournisseurs indépendants de logiciel et du plan de mise en œuvre du client (tels que les allocations mutuelles et statiques par rapport aux allocations dynamiques). Ce plan peut être codé dans AWS CDK ou AWS CloudFormation. L'exécution de ce code déploie les conditions requises pour le réseau de l'infrastructure cloud.

## Déploiement d'infrastructure

Le déploiement de l'infrastructure alloue tous les composants d'infrastructure. Cela inclut la génération du cluster EKS et de l'infrastructure de prise en charge, telle qu'EFS, les nœuds worker EKS, les ELB et la configuration du cluster en fonction des exigences des fonctions de réseau natif cloud. Sur la base des exigences des fonctions CNF, AWS déploie également des interfaces réseau supplémentaires pour les nœuds, y compris des interfaces [Multus](#). La plupart des phases de déploiement et de configuration sont ponctuelles pour une application. Elles sont mises à jour seulement lorsque cela est nécessaire en tant que mise à jour pour l'application.



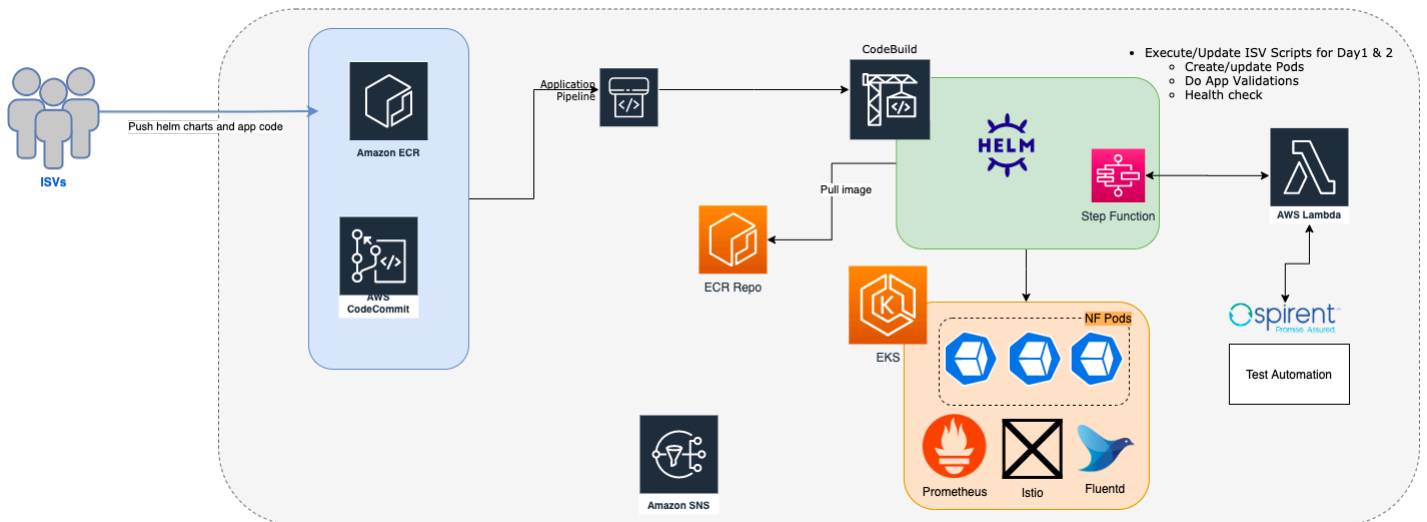
Déploiement de l'infrastructure avec CDK

## Déploiement de fonctions de réseau natif cloud

Le déploiement de fonctions CNF se rapporte au déploiement de l'application. Dans le cadre du déploiement de fonctions CNF, les Charts de Helm de l'application sont implémentés au moyen du pipeline de code CI/CD. Le rappel pour exécuter des scripts propres à une application impliquant principalement des vérifications avant et après est incorporé. Les Charts de Helm sont implémentés dans un ordre dépendant des besoins de l'application. Ils contrôlent l'état des pods Kubernetes avant de passer à l'étape suivante du déploiement. Souvent, les fournisseurs indépendants de logiciel fournissent un script de wrapper permettant l'exécution des Charts de Helm et les vérifications d'intégrité. Ces scripts sont invoqués depuis AWS CodePipeline. Dans le cadre de cette phase, les agents de journalisation et de surveillance tels que Prometheus et Fluentd sont déployés en plus d'Amazon CloudWatch, qui journalise et surveille l'infrastructure cloud de l'application.

Le pipeline de code est intégré à l'infrastructure d'automatisation des tests tiers. Le pipeline de code peut directement appeler les API de l'infrastructure d'automatisation des tests pour exécuter le test sur l'application déployée, interroger les résultats du test et analyser le résultat. Cela simplifie le déploiement et les tests de l'application.

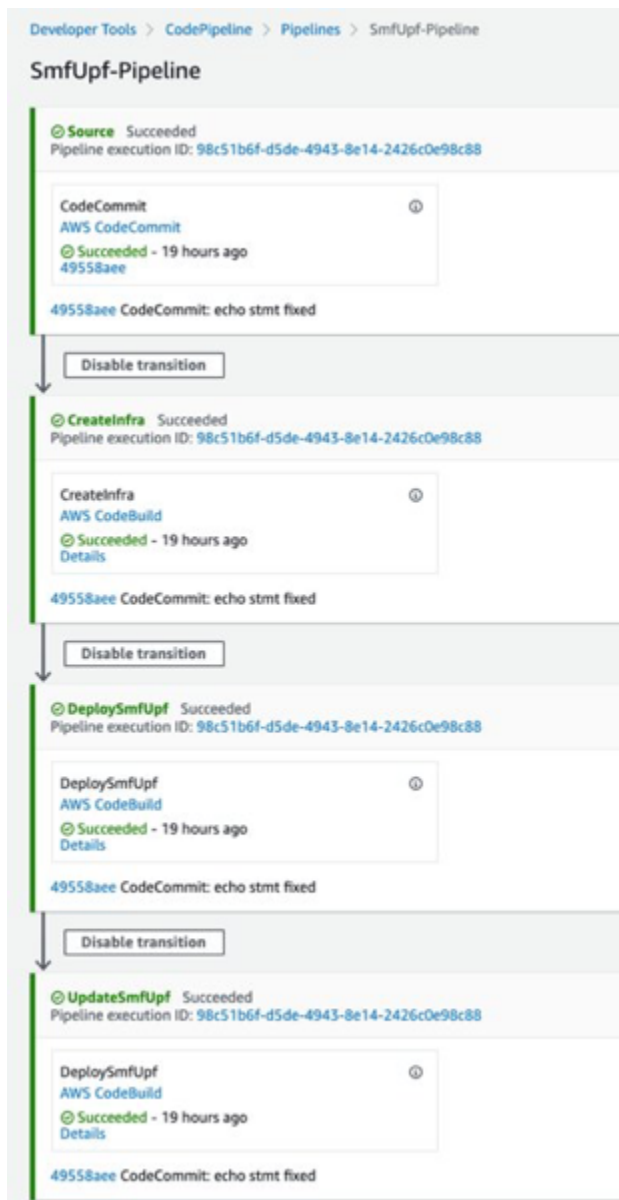




## Déploiement et mise à jour des applications

Voici un exemple de déploiement d'une fonction CNF de la fonction de plan utilisateur/de gestion de session (UPF/SMF) au moyen d'AWS CodePipeline.

- Automatisation de l'ensemble du processus CI/CD en utilisant CodeCommit, CodeBuild et CodePipeline.
- Les tâches de création d'infrastructure et d'installation d'applications sont intégrées dans le cadre du pipeline.
- Les agents Fluentd et Prometheus sont installés et créés dans les tableaux de bord Amazon CloudWatch.



The screenshot displays the AWS CodePipeline console for a pipeline named 'SmfUpf-Pipeline'. The pipeline is in a 'Succeeded' state. It consists of four stages, each with a successful execution of an AWS CodeBuild action:

- Source:** Succeeded. Pipeline execution ID: 98c51b6f-d5de-4943-8e14-2426c0e98c88. Action: CodeCommit (AWS CodeCommit). Status: Succeeded - 19 hours ago. ID: 49558aee. Command: echo stmt fixed.
- CreateInfra:** Succeeded. Pipeline execution ID: 98c51b6f-d5de-4943-8e14-2426c0e98c88. Action: CreateInfra (AWS CodeBuild). Status: Succeeded - 19 hours ago. ID: 49558aee. Command: echo stmt fixed.
- DeploySmfUpf:** Succeeded. Pipeline execution ID: 98c51b6f-d5de-4943-8e14-2426c0e98c88. Action: DeploySmfUpf (AWS CodeBuild). Status: Succeeded - 19 hours ago. ID: 49558aee. Command: echo stmt fixed.
- UpdateSmfUpf:** Succeeded. Pipeline execution ID: 98c51b6f-d5de-4943-8e14-2426c0e98c88. Action: DeploySmfUpf (AWS CodeBuild). Status: Succeeded - 19 hours ago. ID: 49558aee. Command: echo stmt fixed.

Each stage includes a 'Disable transition' button and a 'Details' link for the CodeBuild action.

Exemple de déploiement de fonctions CNF UPF/SMF

## Livraison continue de fonctions CNF

Cette phase consiste en une séquence d'étapes qui sont effectuées de manière itérative pour déployer les modifications qui font partie des modifications de conteneur/de configuration entraînant des mises à niveau. La livraison continue de fonctions CNF est automatisée au moyen de pipelines et est spécifique aux applications individuelles. AWS utilise des Charts de Helm standard pour mettre à jour les fonctions CNF spécifiques. Le pipeline de code comporte des vérifications avant et après la mise à jour de l'application. Le pipeline CI/CD mis à jour est également intégré à une

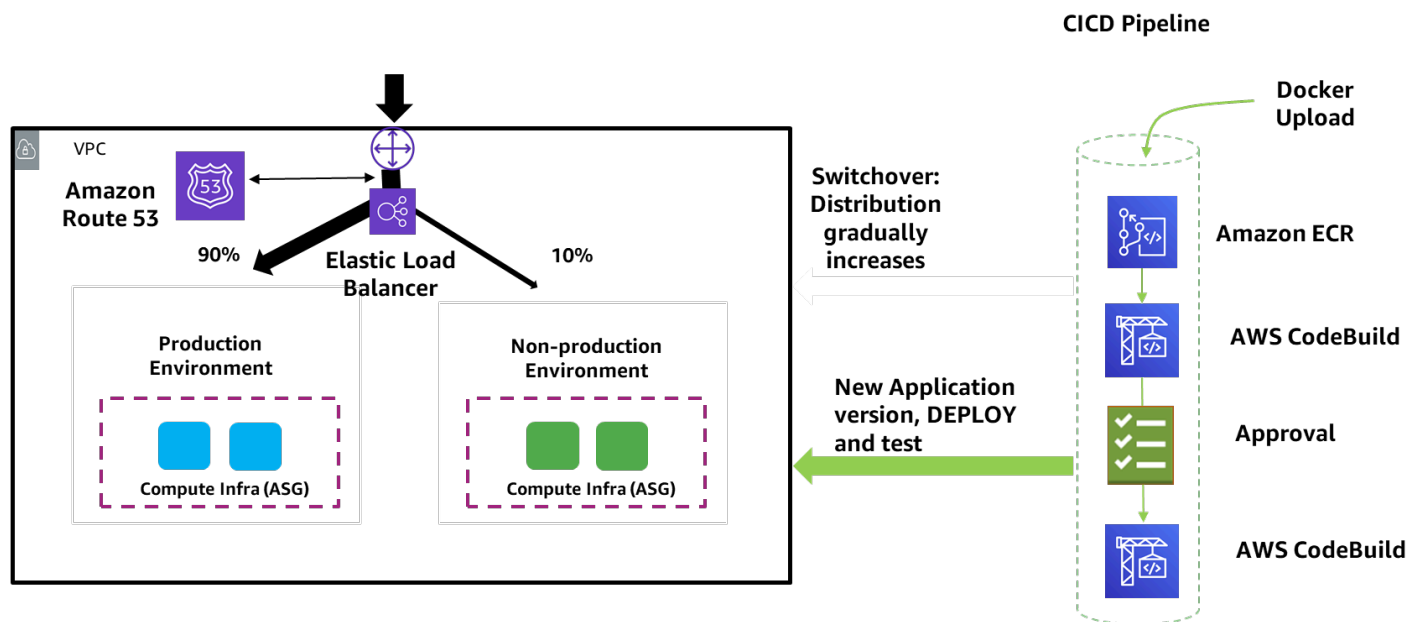
infrastructure d'automatisation des tests afin d'exécuter des tests automatisés. Cette abstraction permet un déploiement propre des fonctions réseau.

La livraison continue et le déploiement de fonctions CNF peuvent être classés dans les catégories ci-après :

- **Mises à niveau des applications** : la plupart des mises à niveau d'applications sont des modifications apportées aux pods d'application Kubernetes. Ces mises à jour peuvent être appliquées automatiquement au moyen du pipeline de code. La plupart des fonctions CNF prennent en charge les mises à niveau sur place en fournissant plusieurs instances de pods d'applications. Les instances multiples permettent une approche de mise à niveau progressive. Tous les changements apportés aux pods d'application ne prennent pas en charge les Charts de Helm. Les pipelines tiennent compte de ces variations et utilisent l'installation/suppression de Helm selon les besoins.
- **Mises à niveau majeures** : les mises à niveau majeures consistent principalement en des modifications de schéma de base de données. Ce changement ne peut pas être appliqué sans provoquer un certain temps d'arrêt. L'approche standard relativement à ces modifications consiste à supprimer l'application et à recréer les pods appropriés. Au cours du processus, l'application peut ne pas être disponible. Les outils suivants sont utilisés pour les mises à niveau :
  - [AWS CloudFormation](#) permet aux clients de décrire et d'allouer toutes les ressources d'infrastructure dans des modèles JSON ou YAML. AWS CloudFormation fournit un puissant mécanisme d'extension grâce à des ressources personnalisées basées sur Lambda. Les clients peuvent étendre AWS CloudFormation au-delà des ressources AWS et allouer les ressources requises dans d'autres environnements, par exemple des ressources sur site dans des environnements hybrides. AWS CDK offre aux développeurs la possibilité de générer du code à l'aide de langages de programmation familiers de niveau supérieur tels que Python, TypeScript, JavaScript, Java et C#, puis de compiler le code dans un format JSON AWS CloudFormation de niveau inférieur, qui peut ensuite être déployé.
  - **Déploiement bleu/vert** : AWS prend en charge et recommande les déploiements bleu/vert et canary dans les environnements de test et de production. Les [déploiements bleu/vert](#) permettent aux clients de tester une nouvelle version de l'application dans un environnement restreint. Ils fournissent une méthode simple et adéquate pour basculer en trafic de production. Les [déploiements canary](#) élargissent ce concept en permettant de tester l'environnement vert hors production avec une faible proportion du trafic de production afin de découvrir tout problème causé par

le trafic de production. La nouvelle version de l'application est testée par rapport au trafic de test simulé interne, ainsi que par rapport à de petites quantités de trafic de production, ce qui donne confiance à l'utilisateur avant qu'il ne bascule sur le trafic de production. Le trafic de production est progressivement augmenté jusqu'à ce que le basculement soit terminé. L'implémentation implique des groupes cibles pondérés DNS et ELB.

- L'automatisation peut être réalisée en configurant AWS CodePipeline avec les phases de déploiement bleu/vert et canary. La phase d'approbation peut être pilotée manuellement au début pendant l'allocation, mais elle doit ensuite être entièrement automatisée. Dans les environnements de test, il est recommandé de toujours réaliser les tests avec une action de restauration pour valider la compatibilité ascendante et descendante, avant le déploiement en production. Le déploiement bleu/vert sur les clusters avec maillage de services dépend de la prise en charge fournie par l'application finale et la passerelle de routage pour que le maillage de services effectue une transition adéquate.
- [AWS Systems Manager](#) fournit une interface utilisateur unifiée qui permet de visualiser les données opérationnelles de plusieurs services AWS utilisés par les fonctions réseau déployées par CI/CD. Systems Manager permet d'automatiser les tâches opérationnelles sur l'ensemble des ressources AWS.



## Déploiement canary

## Sécurité

La sécurité est un élément essentiel. Voici une liste des étapes de sécurité prises en compte par le processus d'intégration et de livraison continues d'AWS lors du déploiement d'une application.

- **Source** : le référentiel ECR alloué au fournisseur est configuré avec un indicateur « Scan on Push » activé, de sorte que tout chargement d'images Docker sera immédiatement soumis à une analyse de sécurité. Toutes les vulnérabilités et expositions courantes connues sont signalées par des notifications. En dehors de l'ECR, lorsque les fournisseurs placent des graphiques dans le référentiel AWS CodeCommit, ils sont invités à chiffrer tous les mots de passe utilisés avec Secrets Manager plutôt qu'avec du texte brut.
- **Intégrité des artefacts** : les artefacts utilisés dans le pipeline sont chiffrés, qu'ils soient au repos (à l'aide de clés gérées par AWS) ou en transit (en utilisant le protocole SSL/TLS).
- **Utilisateurs et rôles IAM** : les autorisations accordées aux utilisateurs ou aux ressources sont basées sur le principe du moindre privilège. Il doit exister une relation d'approbation entre les rôles IAM qui peut devoir être configurée si plusieurs ressources sont opérationnelles dans différents services. Par exemple, AWS CodeBuild nécessite une autorisation pour exécuter des commandes sur un cluster Amazon EKS.
- **Audit** : la capacité d'audit offerte par [AWS CloudTrail](#) suit chaque appel d'API à travers les services et les opérations des utilisateurs, et permet l'évaluation des événements passés.
- **Analyse des vulnérabilités liées aux images** : les images CNF chargées sur Amazon ECR sont automatiquement analysées pour détecter les failles de sécurité. Un rapport des résultats de l'analyse est disponible dans la [AWS Management Console](#), et peut également être récupéré via l'API. Les résultats peuvent ensuite être envoyés aux opérateurs des fournisseurs de services de communication pour une action corrective, y compris le remplacement de l'image CNF.

Des contrôles de sécurité ont lieu à différentes étapes du pipeline pour garantir que l'image nouvellement chargée est sécurisée et conforme aux contrôles de conformité souhaités, afin qu'une notification puisse être envoyée aux fournisseurs de services de communication pour approbation :

- Le registre de conteneur recherche toutes les vulnérabilités CVE ouvertes.
- La configuration est vérifiée pour détecter les fuites d'informations, les schémas de données d'identification personnelle (PII) connus, pendant la phase de test, ce qui déclenche des règles de vérification de conformité pour des problèmes tels que des ports TCP/UDP ouverts inattendus et des vulnérabilités de déni de service.

- La compatibilité ascendante et descendante est vérifiée aux fins de sécurité de mise à niveau/restauration.

Outre l'application, il est essentiel d'assurer la sécurité du pipeline en garantissant le transfert chiffré des artefacts entre les phases, au repos ou en transit.

## Observabilité

AWS permet l'observabilité des fonctions CNF 5G qui sont déployées sur AWS par défaut. Cela est rendu possible par Amazon CloudWatch. CloudWatch apporte une visibilité complète à vos ressources et applications cloud.

Amazon CloudWatch réalise quatre étapes principales au cours de ce processus :

1. Collecte : collecte des métriques et des journaux de l'ensemble de vos ressources, applications et services AWS qui s'exécutent sur des serveurs AWS et sur site.
2. Surveillance : visualisation des applications et de l'infrastructure avec les tableaux de bord CloudWatch, corrélation des journaux et des métriques en vis-à-vis pour résoudre les problèmes et définition des alertes avec [CloudWatch Alarms](#).
3. Action : réponse automatique aux changements opérationnels avec [CloudWatch Events](#) et [AWS Auto Scaling](#).
4. Analyse : métriques allant jusqu'à une seconde, conservation des données étendue (15 mois) et analyse en temps réel avec [CloudWatch Metric Math](#).

L'agent Amazon CloudWatch est installé dans le cluster Kubernetes du client. L'agent prend en charge les fonctions de [configuration](#), de détection et d'extraction de métriques Prometheus, en enrichissant et en publiant toutes les métriques et métadonnées Prometheus haute fidélité au format EMF ([Embedded Metric Format](#)) dans [CloudWatch Logs](#).

[Amazon CloudWatch Container Insights](#) automatise la détection et la collecte de métriques Prometheus à partir d'applications en conteneur. Il collecte, filtre et crée automatiquement des métriques CloudWatch personnalisées agrégées visualisées dans des tableaux de bord.

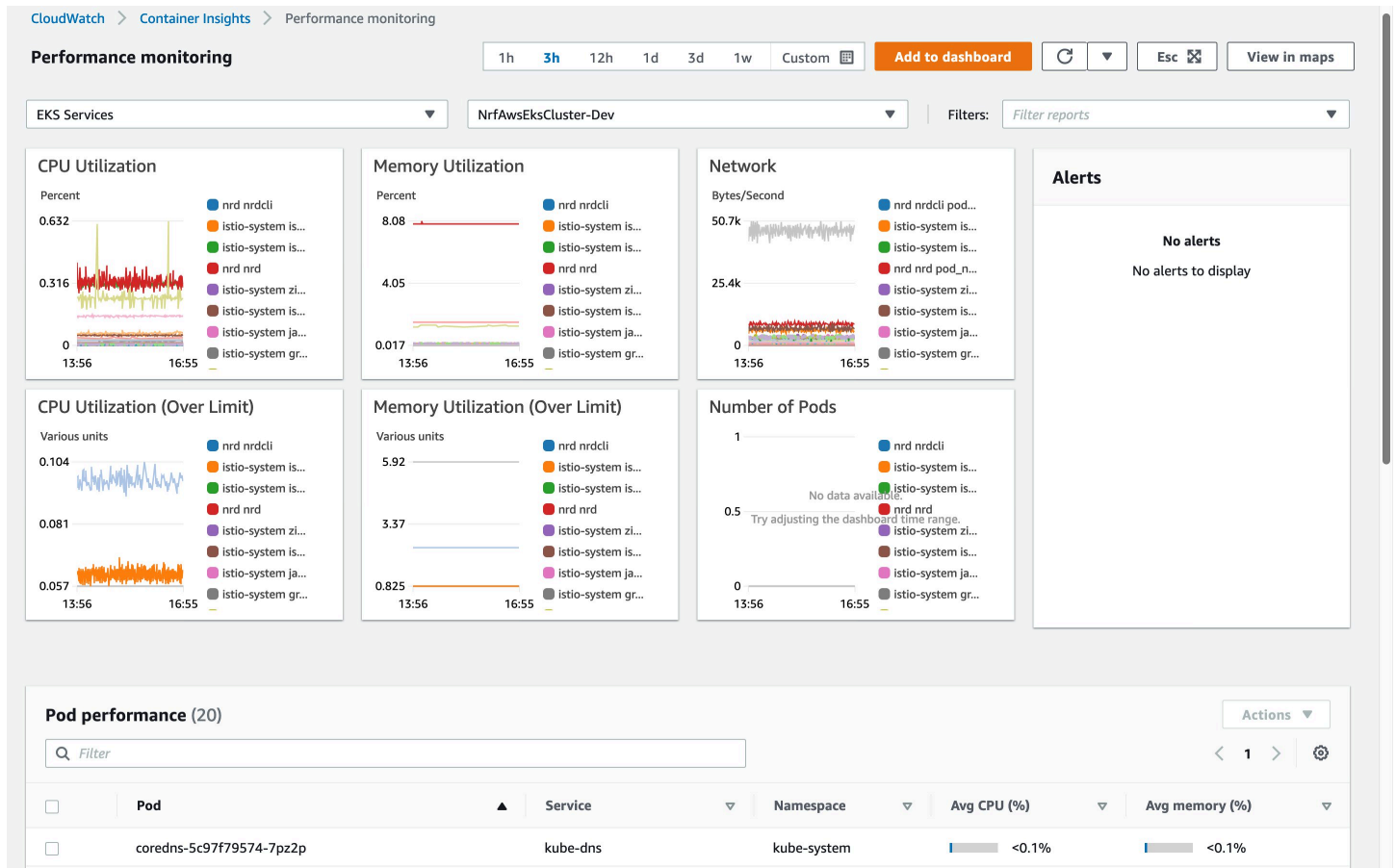
Chaque événement crée des points de données métriques en tant que métriques personnalisées CloudWatch pour un ensemble organisé de dimensions de métriques entièrement configurable. La publication de métriques Prometheus agrégées sous forme de statistiques de métriques

personnalisées CloudWatch réduit le nombre de métriques nécessaires pour surveiller, alerter et résoudre les problèmes de performances et les défaillances. Vous pouvez également analyser les métriques Prometheus haute fidélité à l'aide du [langage de requête CloudWatch Logs Insights](#) afin d'isoler des pods et des étiquettes spécifiques ayant un impact sur l'intégrité et les performances de vos environnements en conteneur.

AWS CloudTrail offre cette visibilité, en enregistrant chaque appel d'API entre les services. [AWS Config](#) offre une capacité de validation de la conformité. AWS fournit aux clients des options de surveillance supplémentaires concernant les métriques, les journaux, les événements pour l'application, l'infrastructure et les pipelines, à l'aide de divers services tels que [AWS X-Ray](#) et [AWS CloudTrail](#).

- AWS peut intégrer en mode natif des outils métriques open source tels que Prometheus, Fluentd, etc.
- Les [métriques Prometheus](#) peuvent être intégrées dans Amazon CloudWatch ou OpenSearch Service pour une analyse plus approfondie.
- AWS utilise Fluentd comme mécanisme standard pour collecter les journaux de différents systèmes. Ce même mécanisme est utilisé et configuré pour ce projet.

Pour en savoir plus sur la configuration de ce mécanisme, consultez [Configurer Fluentd en tant que DaemonSet pour l'envoi de journaux à CloudWatch Logs](#).



Exemple de métriques surveillées par Amazon CloudWatch



# Orchestration CI/CD avec des outils tiers et open source

La couche d'orchestration utilise l'IaC pour déployer et configurer l'infrastructure sous-jacente requise afin d'exécuter les fonctions du réseau 5G. Cette couche doit être conçue pour être modulaire, portable et réutilisable.

L'infrastructure suit les bonnes pratiques en matière de réseau natif cloud, étant hautement disponible, redondante et évolutive.

Comme démontré dans les sections précédentes, le déploiement de l'infrastructure sous-jacente peut être réalisé à l'aide d'[AWS Cloud Development Kit](#). Il est possible d'utiliser pour cela [Terraform](#) de Hashicorp.

## Terraform

Terraform est un outil logiciel IaC open source qui fournit un flux d'interface de ligne de commande (CLI) cohérent pour gérer des centaines de services cloud. Il codifie les API de cloud dans des fichiers de configuration déclaratifs.

Pour procéder à un déploiement avec Terraform, appliquez les mêmes principes que ceux utilisés dans CDK. Le code est structuré en modules qui permettent de personnaliser et de réutiliser les composants réseau en fonction des exigences du fournisseur.

La configuration est entièrement paramétrée, ce qui permet de personnaliser les déploiements en fonction des recommandations des fournisseurs et des fournisseurs indépendants de logiciel.

Le déploiement des fonctions réseau comprend deux phases :

- L'infrastructure AWS requise est créée et gérée au moyen d'un référentiel central.
- La configuration et le code sont stockés de manière centralisée dans un référentiel GitHub.

Une fois les prérequis créés, la fonction réseau est prête à être déployée à l'aide d'un pipeline d'application défini à l'étape précédente.

## Déploiement d'infrastructure

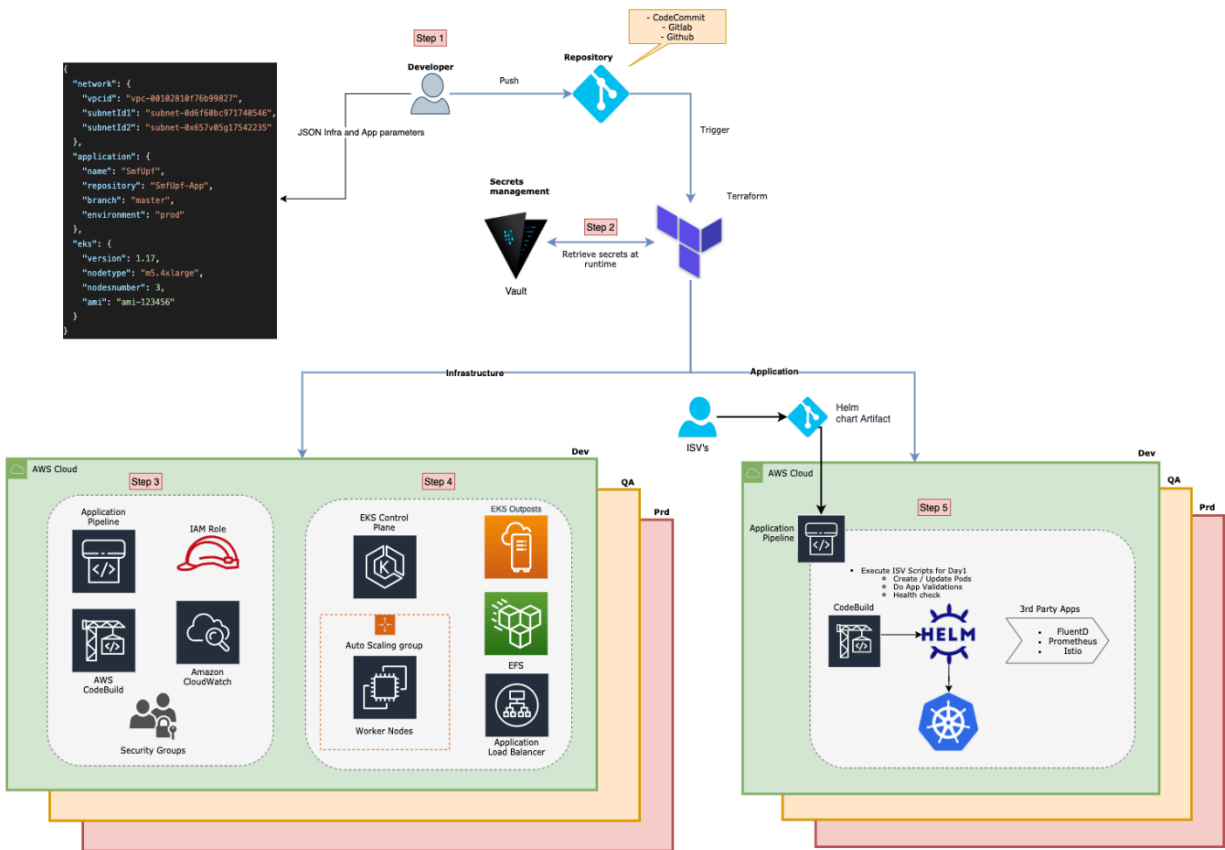
Le déploiement de l'infrastructure inclut toutes les conditions préalables pour que la fonction réseau soit correctement déployée et configurée.

Voici quelques-uns des composants créés dans le cadre de cette phase :

- Mise en réseau : VPC, sous-réseaux publics et privés, routes, équilibreurs de charge
- Calcul : Kubernetes ([VMware Tanzu](#), Amazon EKS ou AWS Outposts), nœuds principaux et worker des instances Amazon EC2, groupe Auto Scaling
- Stockage : Amazon EFS, Amazon EBS, compartiment Amazon S3
- Sécurité : [rôles IAM](#), [groupes de sécurité](#)
- Pipeline : CodePipeline, CodeBuild
- Observabilité : CloudWatch, Prometheus, Fluentd

Voici la séquence d'infrastructure orchestrée par Terraform et expliquée dans la figure suivante :

1. Un développeur remplit un fichier JSON stocké dans un référentiel central avec le code IaC. Le fichier contient des informations sur la configuration d'infrastructure souhaitée, telles que la taille des instances, la version de Kubernetes, les informations réseau et les détails du référentiel d'applications.
2. Récupère les secrets de HashiCorp Vault ou d'[AWS Secrets Manager](#) au moment de l'exécution.
3. Déploie et configure les composants de l'infrastructure (réseaux, calcul, stockage et sécurité).
4. Un cluster Amazon EKS avec des nœuds worker hébergeant les pods de fonctions réseau est déployé. Amazon EKS peut également être déployé sur [AWS Outposts](#) pour prendre en charge les charges de travail qui nécessitent la proximité d'un centre de données.
5. Un pipeline d'application est créé et configuré pour écouter les changements dans le référentiel de fonctions réseau. Chaque fois que du code est envoyé à la branche de référentiel configurée, le pipeline déclenche automatiquement la génération, le test et le déploiement de la fonction réseau.
6. Les outils d'observabilité qui collectent et centralisent les journaux et les métriques sont déployés en tant que services dans tous les nœuds et fournissent des données en quasi-temps réel qui peuvent être visualisées dans les [tableaux de bord Grafana](#) ou [OpenSearch](#).



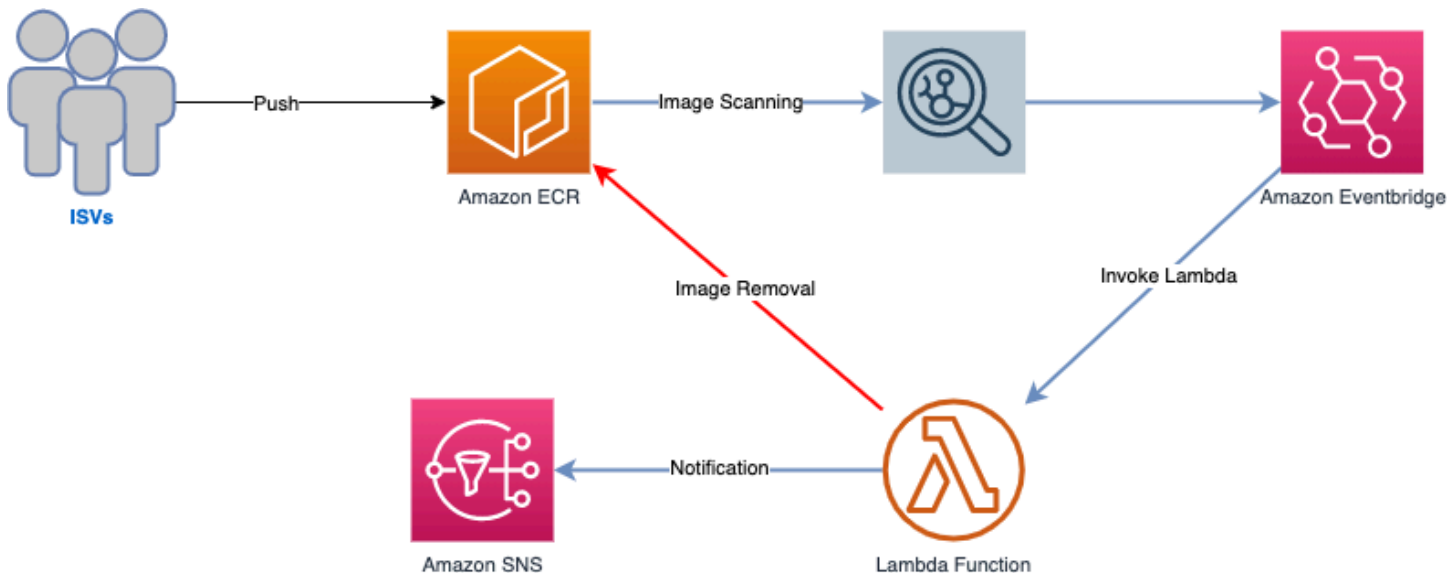
## Déploiement et configuration des fonctions réseau

## Déploiement et configuration des fonctions réseau

Le pipeline créé à l'étape précédente permet aux fournisseurs indépendants de logiciel et aux fournisseurs de décentraliser et d'optimiser le déploiement des fonctions réseau. Le pipeline est connecté et écoute les modifications apportées au référentiel d'applications, qui est configuré dans le fichier JSON à l'étape 1 de la figure précédente.

Pour vérifier les images publiées par des tiers, une solution d'analyse des vulnérabilités qui aide à identifier les vulnérabilités logicielles dans les images de conteneur est déployée et configurée. La solution d'analyse vérifie automatiquement toutes les nouvelles images envoyées à [Amazon ECR](#). Pour en savoir plus sur l'analyse des images ECR, consultez la section [Analyse d'images](#).

La figure ci-dessous illustre l'architecture de la solution Image Vulnerability Scanning.



### Architecture de la solution Image Vulnerability Scanning

Le pipeline d'application peut être configuré pour être déclenché par des modifications de l'image après le résultat de l'analyse, ou par des modifications directement dans le référentiel. Par exemple, lorsqu'une image Helm est créée.

La liste ci-dessous est la séquence qui crée/met à niveau la fonction réseau, comme illustré dans la figure suivante :

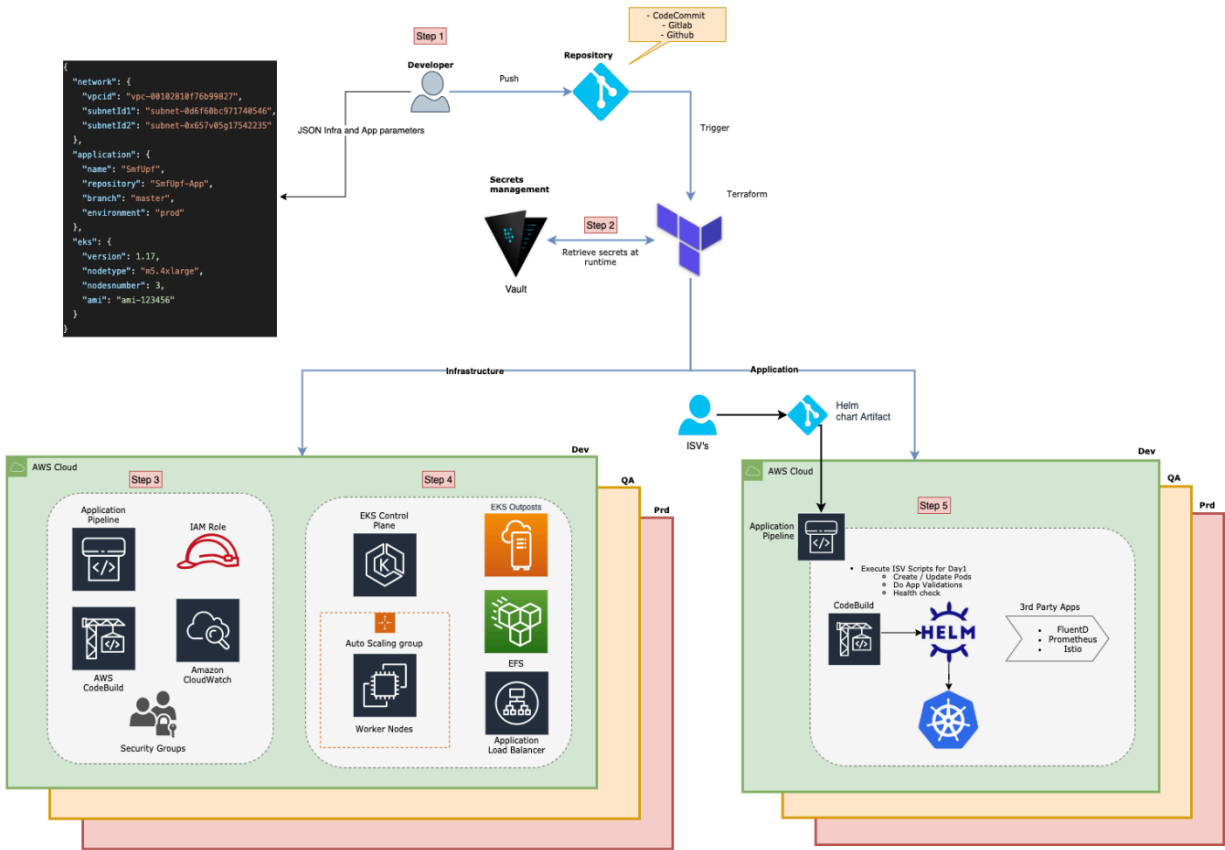
Les fournisseurs indépendants de logiciel publient de nouvelles images sur Amazon ECR. (Si l'image est approuvée, le pipeline d'application est déclenché.)

CodePipeline extrait la nouvelle image d'Amazon ECR et utilise CodeBuild pour déployer l'image sur Kubernetes. Les commandes Helm peuvent être utilisées pour mettre à niveau la fonction réseau.

Une fois l'image déployée, le test en tant que service (TaS) est déclenché. Le TaS valide le nouveau déploiement et centralise les données et les métriques concernant les performances des fonctions réseau sous contrainte.

Les journaux et les métriques sont collectés et centralisés dans OpenSearch et Grafana. Des tiers tels que [Datadog](#), [Istio](#) et Prometheus peuvent également être configurés pour fournir une observabilité supplémentaire.

Une fonction de gestion et d'orchestration (MANO) capable de coordonner les ressources du réseau peut également être déployée et intégrée à la solution. Elle utilise les données collectées pour effectuer des actions automatisées telles que le découpage du réseau et la scalabilité automatique de la qualité de service (QoS).



### Pipeline d'application

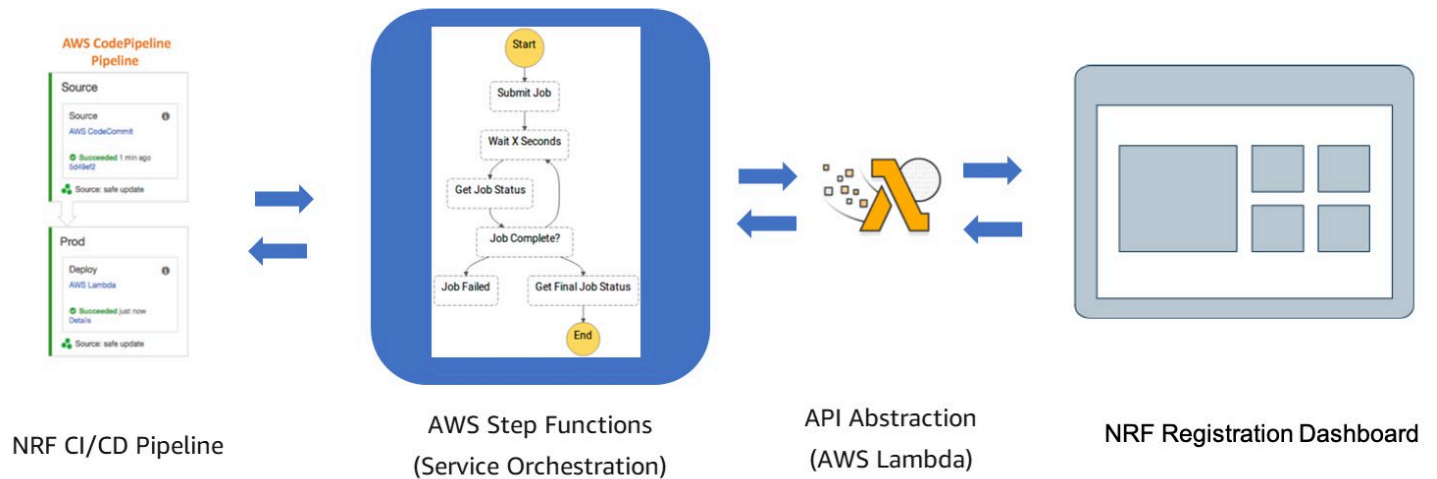
## Tests

L'infrastructure d'automatisation des tests spécifique aux télécommunications peut être intégrée dans le pipeline de code. Le pipeline de code est intégré aux fonctions intermédiaires afin d'orchestrer l'intégration à une infrastructure d'automatisation des tests. AWS Step Functions utilise plusieurs fonctions Lambda pour invoquer l'infrastructure d'automatisation des tests (outils tiers) via des appels d'API. La fonction intermédiaire obtient initialement l'ID de test, exécute le test et obtient les résultats de l'infrastructure d'automatisation des tests. La fonction intermédiaire analyse ensuite les résultats et les transmet au pipeline de code aux fins d'approbation de l'application pour le déploiement de production. L'approbation peut être automatisée ou maintenue manuellement dans le pipeline de code selon les besoins. Il s'agit d'une étape importante pour les fournisseurs de services de communication afin de promouvoir le déploiement de l'environnement de test à la production. Les API de haut niveau requises pour l'intégration sont classées comme suit :

- Obtenir le contexte

- Exécuter un cas de test spécifique
- Arrêter le cas de test
- Obtenir les résultats

La complexité de l'appel des API REST externes est modélisée à l'aide d'AWS Step Functions, qui permet aux constructions standard d'appeler des flux parallèles, d'attendre les résultats, de créer des branches en fonction de conditions et d'intégrer l'API REST à AWS CodePipeline.



## Déroulement des tests

## CI/CD et orchestration

L'intégration continue et la livraison continue font partie d'une philosophie globale d'automatisation, associée à l'architecture de réseau natif cloud et à la façon dont elle s'applique à la 5G.

L'orchestration est un autre aspect de cette philosophie qui doit être dynamique et réactif à tout changement qui se produit sur le réseau. L'orchestration et la CI/CD doivent être étroitement liées afin de garantir un service sain et de minimiser les interruptions de service. L'intégration entre CI/CD et orchestration doit se faire sur deux fronts :

- L'application de correctifs et de mises à niveau dans le système doit être gérée et orchestrée de manière à minimiser les perturbations des services en direct. Par exemple, l'orchestration peut déterminer dynamiquement le meilleur moment où une mise à jour doit être déployée.
- L'orchestration compatible CI/CD permet de déplacer le trafic pendant le déploiement des mises à niveau en fonction de la stratégie de modèle de déploiement adoptée (canary, linéaire ou simultanée).

En règle générale, les solutions d'orchestration s'exécutent au-dessus des pipelines CI/CD afin que l'orchestration puisse introduire des phases de gouvernance dans ces pipelines et être exposée aux cycles de mise à niveau en cours.

## Conclusion

L'intégration et la livraison continues offrent aux développeurs et aux équipes responsables des applications une méthode claire et efficace pour déployer un nouveau code d'application en quelques minutes. AWS propose de nombreux outils qui peuvent aider les développeurs à intégrer, à tester et à déployer de nouveaux codes, notamment AWS CodePipeline, AWS CodeCommit, AWS CodeBuild, AWS CodeDeploy et bien d'autres. Ce document a passé en revue la façon dont les services AWS peuvent être utilisés pour créer un processus CI/CD afin de déployer la fonction réseau 5G de manière entièrement automatisée, y compris les différentes étapes nécessaires pour terminer le déploiement d'un nouveau code. Il présente également la façon d'intégrer une infrastructure d'automatisation des tests tiers au processus CI/CD, ainsi que l'utilisation d'outils tiers tels que Terraform.



# Participants

Ont contribué à la préparation du présent document :

- Hisham Elshaer, consultant senior, AWS Telecom, Amazon Web Services
- Vara Prasad Talari, consultant principal, AWS Telecom, Amazon Web Services
- Rabi Abdel, consultant principal, AWS Telecom, Amazon Web Services
- Franco Bontorin, consultant senior, livraison partagée, Amazon Web Services
- Pragtideep Singh, consultant, livraison partagée, Amazon Web Services
- Subbarao Duggisetty, architecte d'infrastructure cloud, comptes mondiaux, Amazon Web Services
- Young Jung, architecte principal de solutions partenaires, AWS Telecom, Amazon Web Services

## Révisions du document

Pour être informé des mises à jour de ce livre blanc, abonnez-vous au flux RSS.

update-history-change

update-history-description

update-history-date

[Publication initiale](#)

Première publication du livre  
blanc

8 mars 2021

# Suggestions de lecture

Pour en savoir plus, voir :

- [Mise en pratique de l'intégration continue/livraison continue sur AWS](#) (livre blanc)
- [Réseau principal de paquets mobiles à l'échelle de l'opérateur sur AWS](#) (livre blanc)
- [Évolution du réseau 5G avec AWS](#) (livre blanc)

# Acronymes

- AMF – Fonction de gestion de l'accès et de la mobilité
- API – Interface de programmation d'applications
- AUSF – Fonction du serveur d'authentification
- BSS – Système de support pour les entreprises
- CDK – Cloud Development Kit
- CI/CD – Intégration continue et livraison continue
- CLI – Interface de ligne de commande AWS
- CNF – Fonction de réseau natif cloud ou en conteneur
- CSP – Fournisseur de services de communication
- UC – Unité centrale RAN
- CVE – Vulnérabilités et expositions courantes
- DoS – Déni de service
- DR – Reprise après sinistre
- DU – Unité distribuée RAN
- E2E – de bout en bout
- ECR – Elastic Container Registry
- EFS – Elastic File System
- EKS – Elastic Kubernetes Service
- EPC – Cœur de paquet évolué (Evolved Packet Core)
- IaC – Infrastructure as Code
- FIL – Fournisseur indépendant de logiciel
- MANO – Gestion et orchestration
- MEC – Multi-Access Edge Computing
- NACL – Liste de contrôle d'accès (ACL) réseau
- NAT – Traduction d'adresses réseau (Network address translation)
- NF – Fonction réseau
- NFV – Virtualisation des fonctions réseau
- NFVO – Orchestrateur de virtualisation des fonctions réseau

- NOC – Centre d'exploitation du réseau
- NRF – Fonction de référentiel réseau
- OSS – Système de support opérationnel
- PII – Informations personnelles identifiables
- QoS – Qualité de service
- RAN – Réseau d'accès radio
- SBI – Interface basée sur les services
- SMF – Fonction de gestion de session
- SSL – Secure Sockets Layer
- TaS – Test en tant que service
- TCP – Transmission Control Protocol
- TLS – Sécurité de la couche transport (Transport Layer Security)
- UDM – Gestion unifiée des données
- UDP – User Datagram Protocol
- UPF – Fonction de plan utilisateur
- VIM – Gestionnaire d'infrastructure virtualisée
- VNF – Fonction de réseau virtuel
- VPC – Virtual Private Cloud

## Mentions légales

Les clients sont responsables de leur propre évaluation indépendante des informations contenues dans ce document. Le présent document : (a) est fourni à titre informatif uniquement, (b) représente les offres et pratiques actuelles de produits AWS, qui sont susceptibles d'être modifiées sans préavis, et (c) ne crée aucun engagement ou assurance de la part d'AWS et de ses affiliés, fournisseurs ou concédants de licences. Les produits ou services AWS sont fournis « en l'état » sans garantie, représentation ou condition, de quelque nature que ce soit, explicite ou implicite. Les responsabilités et obligations d'AWS envers ses clients sont déterminées par les contrats AWS, et le présent document ne fait pas partie d'un contrat entre AWS et ses clients, ni le modifie.

© 2021, Amazon Web Services, Inc. ou ses sociétés apparentées. Tous droits réservés.