Tech Mahindra

aws

# Orchestration of Telco workloads on Amazon Web Series (AWS)

Approaches to orchestrate Telco virtual network functions on AWS

**December 2019**

Connected World.
Connected Experiences.

# Contents

## Abstract

Telecom service providers have rapidly adopted Network Function Virtualization (NFV) and Software-Defined Networking (SDN) technologies in their networks to realize benefits like hardware vendor independence, improved operational efficiency and dynamic realization of new services. Many service providers have begun the process of network transformation from traditional physical network functions (PNFs) to the corresponding Virtual Network Functions (VNFs) by deploying telco cloud solutions on their private data centers. The telco cloud solution comprises of underlying NFV infrastructure (NFVI), the VNF workloads and an orchestrator to manage these VNFs and corresponding network services. The orchestration plays an important role in service provider's ability to launch new services as per customer demands and enable automation to manage these network services. Increasingly, the service providers are migrating their network workloads to public cloud providers like Amazon Web Service (AWS) to realize tangible benefits of public cloud computing including massive scalability, reliability and webscale architecture. This whitepaper describes different orchestration approaches using opensource orchestrator, like ONAP, and other alternate methods that a service provider can implement to leverage AWS as an NFVI and orchestrate Telco workloads on AWS cloud infrastructure.


## Introduction

Traditional telecom industry is facing several challenges such as saturated consumer market, declining ARPU (average revenue per user) and high Capex & Opex required to build, operate and upgrade the geo-disperse complex mobile network infrastructure. In addition, technological advances such as 5G require network service providers to transform their business model to stay competitive, sustain and accelerate the business growth. Architectures and enabling technologies such as Cloud Computing, Network Function Virtualization (NFV), Software Defined Networking (SDN), and Edge Computing (MEC) are providing new avenues to deal with the business challenges by enabling service providers to deliver innovative network services, faster time to market and customer use-cases across various market segments.

With the rise of public cloud service providers, spectrum of cloud technologies spanning cloud service models - IaaS, PaaS, and SaaS, Edge compute services and evolution of architectural patterns such as Microservices using Containers and Serverless, mobile network operators are increasingly adopting multi-domain hybrid cloud strategy to deal with the aforementioned industry challenges. This includes utilizing their own on-premises facilities / data centers as well as public cloud infrastructure for continuous integration, development, and automated deployment and operation of network services. These factors contribute to making the orchestration layer, that provides an ability to coordinate infrastructure resources, service creation, service chaining and lifecycle management, a vital component for the public and hybrid cloud adoption. As 5G is evolving, the mobile operators are looking for faster and cost-effective 4G to 5G upgrade path with network transformation in the form of Virtual Network Functions (VNFs) and Cloud-Native Network Functions (CNFs) in multi-domain hybrid cloud environment using their own data centers and public cloud platform making cloud orchestration a centerpiece of 5G network deployments.

This whitepaper describes two distinct orchestration approaches mobile network operators can adopt with their existing private data center infrastructure and AWS public cloud platform: First, utilizing Open Source ONAP (Open Network Automation Platform) orchestrator capabilities to manage their private cloud infrastructure as well as the AWS public cloud infrastructure using AWS cloud plug-in built into the ONAP orchestrator. Second, using AWS cloud native orchestration capabilities and hybrid cloud solutions.

AWS public cloud platform with 22+ geographic regions, 69+ Availability Zones, 195+ Edge locations across the globe with built-in reliability, and redundancy provides telecom network providers best of both, IT and Telecom, worlds with the ability to transform their networks using the AWS cloud native services, orchestration frameworks, and automation tools for network functions development and deployments in continuous integration continuous delivery (CI/CD) "DevOps" fashion and provide the scale needed to meet exploding traffic demands thereby deliver increased service revenues. 69 proactive cost reductions to date ensures that the cost of the infrastructure needed to support new services remains low.

With 175+ cloud services, Amazon Web Services (AWS) provides the broad range of cloud capabilities from compute, storage, databases, networking, analytics, machine learning and artificial intelligence (AI), Internet of Things (IoT), security, application development, deployment, and management tools with pay-as-you-go pricing enabling media and telecom service providers and customers across the wider industry segments focus on the business outcome while reducing the cost and increasing efficiency, agility, availability, reliability, and security. AWS also offers ever-growing ISV partner ecosystem with hundreds of Telecom ISV partner products available on AWS marketplace giving telecom network service providers a wider choice of AWS compatible ISV products to choose from.

AWS' IaaS (Infrastructure-as-a-Service) and PaaS (Platform-as-a-Service) cloud capabilities can be combined to create a scalable and highly available secure cloud application without worrying about the provisioning and management of underlying infrastructure (compute, storage, and network). AWS cloud platform offers six fundamental advantages of cloud computing: (1) Trade capital expense for variable expense by paying only when you consume and how much you consume (2) Massive economies of scale with lower pay-as-you-go prices (3) Stop guessing capacity: eliminate guessing your infrastructure capacity needs. With cloud computing, you can access as much or as little capacity as you need, and scale up and down as required with only a few minutes' notice (4) Increase speed and agility with faster time to market since the cost and time it takes to experiment and develop is significantly lower and the resources are readily available to your developers in minutes as opposed to weeks (5) Focus on the customers and the business objective, not on the infrastructure. Cloud computing provides a simple way to access servers, storage, databases, networking and a broad set of secure application services over the Internet (6) Go global in minutes: Easily deploy your applications in multiple regions around the world with just a few clicks. This means you can provide lower latency and a better experience for your customers at minimal cost and with easily manageable global footprint.

With the recently announced AWS Outposts service, Amazon has enabled telecom service providers to create hybrid cloud infrastructure by hosting an AWS public cloud native environment on-premises. Outposts extends AWS cloud capabilities to service provider data center, on-premises facility or co-location site providing the same AWS native services, infrastructure, APIs, deployment tools and control plane that customers use in AWS to support VNFs and workloads that need to remain on-premises to achieve low latency response times or process data locally, offering truly consistent hybrid cloud experience. It is a fully managed service; the physical infrastructure is delivered and installed by AWS, operated and monitored by AWS, and automatically updated and patched as part of being connected to an AWS Region.

In this dynamic Telco environment with multiple clouds – private and public – and different workload characteristics, the role of orchestration becomes very important and pivotal to ensure that the components of a network service are deployed, configured and managed precisely to provide the reliable service expected from a Telco operator, which is discussed in subsequent sections of this document in details primarily focusing on ONAP and AWS cloud native orchestration capabilities.

## Virtualization in Telcos and current orchestration approaches

Traditionally, network function implementations are packaged with the infrastructure they run on – but no longer. As the physical network is decoupled from the infrastructure and network services, it is necessary to create both new management tools and orchestration solutions for service providers to realize the benefits of NFV-based solutions.

The European Telecommunications Standards Institute (ETSI) defined the Network Function Virtualization - Management and Orchestration (NFV-MANO) architecture to address these needs. The NFV-MANO architecture consists of three major functional blocks namely Virtual Infrastructure Manager (VIM), Virtual Network Function Manager (VNFM) and the NFV Orchestrator (NFVO).

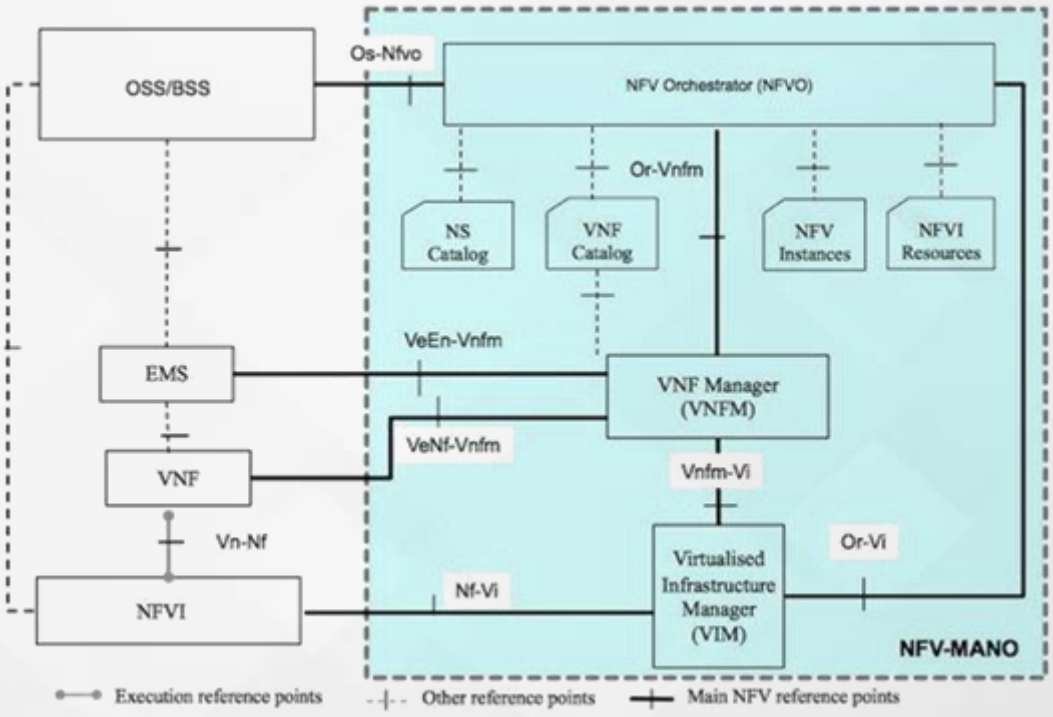



**Figure 1 : ETSI MANO reference architecture**

The **Virtualized Infrastructure manager** (VIM) is responsible for controlling and managing the NFV infrastructure (NFVI) compute, storage and network resources, usually within one operator's infrastructure domain.

The **NFV Orchestrator** (NFVO) provides management of the NFV services, which is responsible for on-boarding of new Network Services (NS) and Virtual Network Function (VNF) packages; NS Lifecycle Management; Global Resource Management; validation and authorization of network functions virtualization infrastructure (NFVI) resource requests.

The **VNF Manager** (VNFM) manages the life cycle of each VNF. As many of the VNFs have their specific methods for provisioning and lifecycle management, each VNF vendor typically brings in their own Specific VNFM (S-VNFM). Telco operators look for a Generic VNFM (G-VNFM) that works with ETSI-standard compliant VNFs, so they can run only one VNFM for VNF lifecycle management of different VNFs which are used to build the telco services they offer.

While there are many commercial orchestration products, service providers and OEMs have worked together to create open source orchestrators. The Open Network Automation Platform (ONAP) and Open Source Mano (OSM) are the two main open source MANO projects. ONAP is the most widely followed open source initiative with active participation from across the industry players and is considered a more feature-rich platform with backing from leading Tier1 service providers.

Cloud-native Network Functions (CNFs) is the next wave in the network function virtualization space. With Cloud native approach, the network functions, which were implemented as monolithic virtualized applications, are now broken into smaller microservices and deployed as containers in both the public and private clouds. CNFs are increasing being used to support services like 5G slicing. ONAP platform has evolved to support CNFs and will also cater to edge orchestration needs.

Most of the Telco cloud deployments today have NFV infrastructure (NFVI) in service provider's private data center with the NFV orchestrator managing the workloads on-prem. To avail of the scalability, flexibility and cost advantage currently being used by enterprises, Telco operators are increasingly showing interest in migrating the network workloads from their on-prem infrastructure on to public cloud like AWS. In such a hybrid cloud scenario, the NFV orchestrator should offer seamless support for workloads to be orchestrated between public cloud and private cloud. The next logical step will be to go 'all-in' on public cloud where in different layers of the MANO architecture (NFVI, VIM, NFVO, VNFs) will be deployed on a public cloud.

The subsequent sections of the whitepaper describe how Telcos currently use ONAP for orchestration, approaches for supporting hybrid cloud orchestration using ONAP and alternative approaches of orchestrating workloads on AWS using native AWS services.

# ONAP overview

Open Network Automation Platform (ONAP) is an open source initiative which provides a comprehensive platform for real-time, policy-driven orchestration and automation of physical and virtual network functions that will enable software, network, IT and cloud providers and developers to rapidly automate new services and support complete lifecycle management.

The high level architecture of ONAP is reproduced below for reference.

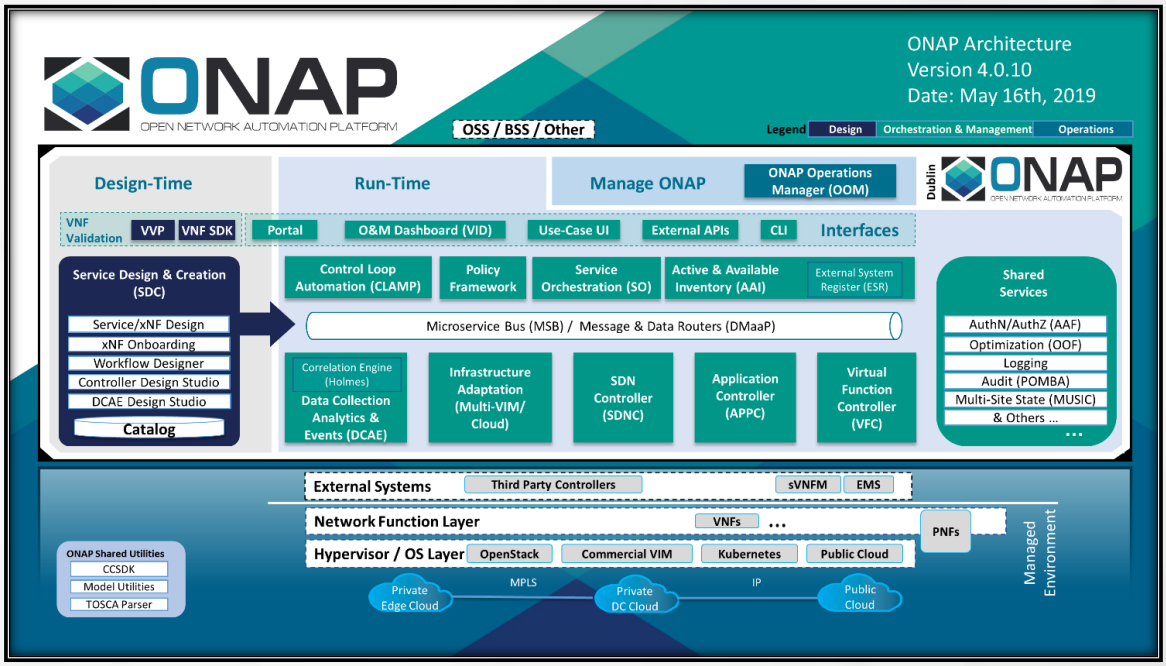



**Figure 2 : ONAP architecture**

The table below highlights some of the key ONAP components and their functional areas

| ONAP Component | Functionality/Feature |
|---|---|
| Service Design and Creation (SDC) | Service Design and catalogue |
| Service Orchestrator (SO) | Orchestration |
| Active and Available Inventory (AAI) | Inventory |
| Controllers (SDN-C, APPC, VFC) | VNF and Network Configurations |
| Multi-VIM/Cloud | Support for multiple cloud infrastructure |
| DCAE and HOLMES | Service Assurance and Monitoring |
| CLAMP and Policy | Closed control loop |
| ONAP Operations Manager (OOM) | Operations and Management |
| Application Authorization Framework (AAF) | Security |
| External APIs | integration with BSS/OSS systems |

**Table 1: ONAP Components**

## ONAP orchestration flow

The NFV orchestration workloads are the network services comprising of various VNFs/CNFs and the connectivity between them. ONAP supports modelling of network functions and services using any of the commonly used methods like HEAT, TOSCA. The VNF Descriptor (VNFD) is a template which is used for modelling the VNF components and its connectivity details – ex. VDU Connection Point Descriptors, Virtual Link Descriptors and VNF External Connection Point Descriptors.

The onboarding of vendor VNFs/CNFs and creating the Network Service (NS) is done by Service Design and Creation (SDC) component of ONAP. Once the VNF vendor artifacts are uploaded, the network service comprising of various VNFs is created in SDC. The SDC platform supports various user roles for different phases of service design. The user with designer role is able to onboard VNFs and design the services. The user with tester role is able to certify that the testing is successful. The governor role user has the privileges to distribute the service. Once the network service is tested and approved, SDC interfaces with various ONAP components and distributes the service artifacts for service instantiation. The service catalogue is populated by SDC during the service design. ONAP uses Active & Available Inventory (AAI), a Graph DB based component to store all inventory details and their status. AAI holds references to network services and infrastructure, data center resources, connectivity components, and service overlays.

The Service Orchestrator (SO) is the main orchestration engine responsible of execution of service recipes. SO integrates with various controllers to accomplish the tasks. A Controller manages the state of a single Resource (Application or Network) for example SDN C is controller for configuring network resources. Multi-VIM/Cloud is a controller component of ONAP which is used to manage multiple cloud infrastructure environments like OpenStack, VMware on private clouds and public clouds like AWS.

SO will use the Multi-Cloud/VIM controller component to execute the underlying cloud APIs for configuration and instantiation of the VNFs on the underlying infra (NFVI). The rules and association of workloads to various cloud regions / Data Centers are defined in the ONAP Optimization Framework (OOF) module. The SO during the instantiation work slow, consults the OOF and accordingly routes the request to the concerned VIM that is managing the underlying NFVI.

In ONAP, the VNF configurations are modeled using the tool called Configuration Design Studio (CDS). This tool creates artifacts which are executed during run time to apply the configuration on to the VNF using Application Controller (APP-C) module. APP-C supports various configuration mechanisms like NETCONF, Ansible to apply the configuration on the VNFs.
During run time, ONAP SO will receive the request to instantiate a network service from BSS which calls the REST APIs exposed by the North Bound Interface (NBI).

## ONAP Service Assurance, Monitoring and Closed loop orchestration

The ONAP service assurance architecture consists of Data Collection Analytics & Events (DCAE) with a set of micro services and then coupled with closed loop service assurance features comprising the Policy, Service Orchestrator (ONAP SO component) and Controllers like APP-C & SDN C.

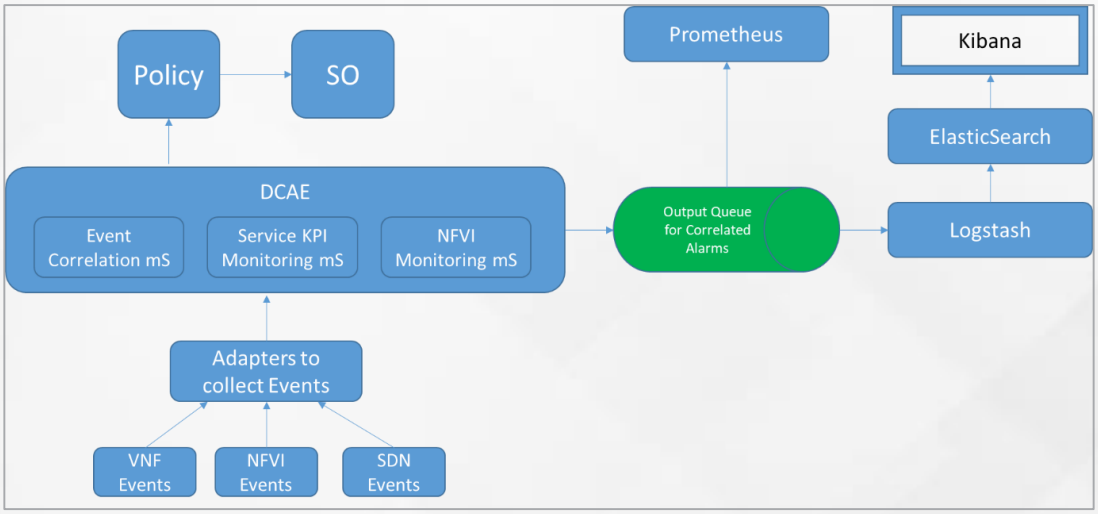



**Figure 3 : ONAP Service Assurance**

The above diagram depicts high-level architecture of how specific events, alarms, matrices, KPIs can be monitored within the ONAP framework and then how the same can be visualized. The events from various sources such as VNFs, NFVIs and SDN controllers can be injected into the DCAE framework. DCAE shall host various micro-services to process different events. All these events from various sources shall be correlated by a KPI monitoring micro-service and high-level performance indicators arrived at. Events from various VNFs shall be correlated to arrive at service level indicators and then the aggregated event shall be sent for logging, searching and to the event monitoring system. The possible solution components for ELK and event monitoring solution are depicted in the above diagram.

Similarly, another micro-service for NFVI monitoring shall keep track of the available resources in NFVI and raise events whenever any threshold is breached.

The Holmes DCAE service is capable of correlating the alarms based on the topology information in Active & Available Inventory (AAI) and is able to arrive a higher level / aggregated alarm / event.

Policy module enables an operator to define policies for remedial actions on specific events generated by the DCAE micro services. These policies could instruct SO to either scale-out, scale-in, migrate the VNFs and to do some specific configurations as may be required thus ensuring closed loop automation.

ONAP also supports Kafka based messaging interface for any external 3rd party tools to process the generated events for alarm correlation, policy execution and enabling closed loop automation.

## Orchestration of Telco workloads on AWS Cloud Platform

The following are the methods of orchestrating and managing Telco workloads on AWS:

1. Using an external orchestration application such as ONAP to manage and orchestrate Telco workloads running on AWS cloud platform
2. Using AWS cloud native orchestration services and capabilities

Both these approaches will be described in detail in the subsequent sections

# ONAP based approaches for orchestration on AWS Cloud

Two key aspects need to be taken into consideration to use ONAP as an orchestrator for AWS cloud: (1) ONAP orchestration engine installation on AWS cloud (2) Mechanisms that ONAP needs to support for orchestrating the network workloads on the AWS cloud infrastructure.

All the ONAP components are cloud native and are deployed on a Kubernetes (K8s) cluster using the component called ONAP Operations Manager (OOM). Since Kubernetes cluster deployment is a prerequisite for ONAP deployment, Amazon Elastic Kubernetes Service (EKS) could be used to set-up the desired Kubernetes cluster for ONAP deployment. This will speed up the ONAP deployment process since the manual tasks of setting up the K8s cluster could be automated using AWS EKS tools.

Apart from the ease of deployment, highly available clustered ONAP deployment scenarios could also benefit from Amazon EKS features that run the Kubernetes management infrastructure across multiple AWS regions and availability zones to eliminate a single point of failure.

The overall steps are depicted in the diagram below:


**Figure 4:** ***ONAP Orchestrator deployment on AWS using Amazon EKS***

## Network Workload Categories

The network functions today are realized as both, Virtual Network Functions (VNFs) and as Container Network Functions (CNFs). ONAP framework supports orchestration of network services composed of both, VNFs and CNFs.

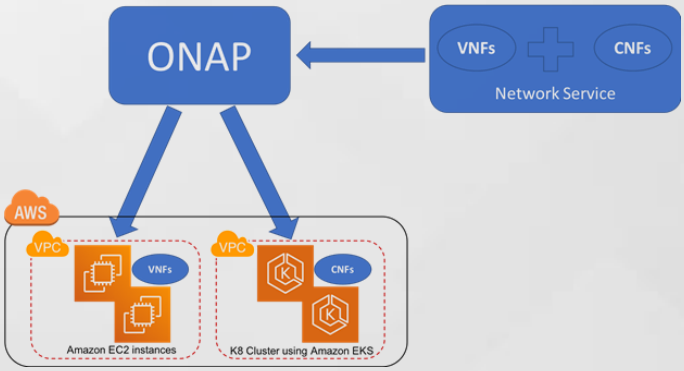



**Figure 5: Network Functions Deployments on AWS with ONAP Orchestrator**

Orchestration of CNFs would also most often require a K8s cluster set-up as a pre-requisite. From a logical perspective, this K8s cluster would be defined as a cloud region within the ONAP A&AI.

The figure below depicts the deployment of ONAP components on AWS public cloud and how ONAP will orchestrate and manage the telco workloads.

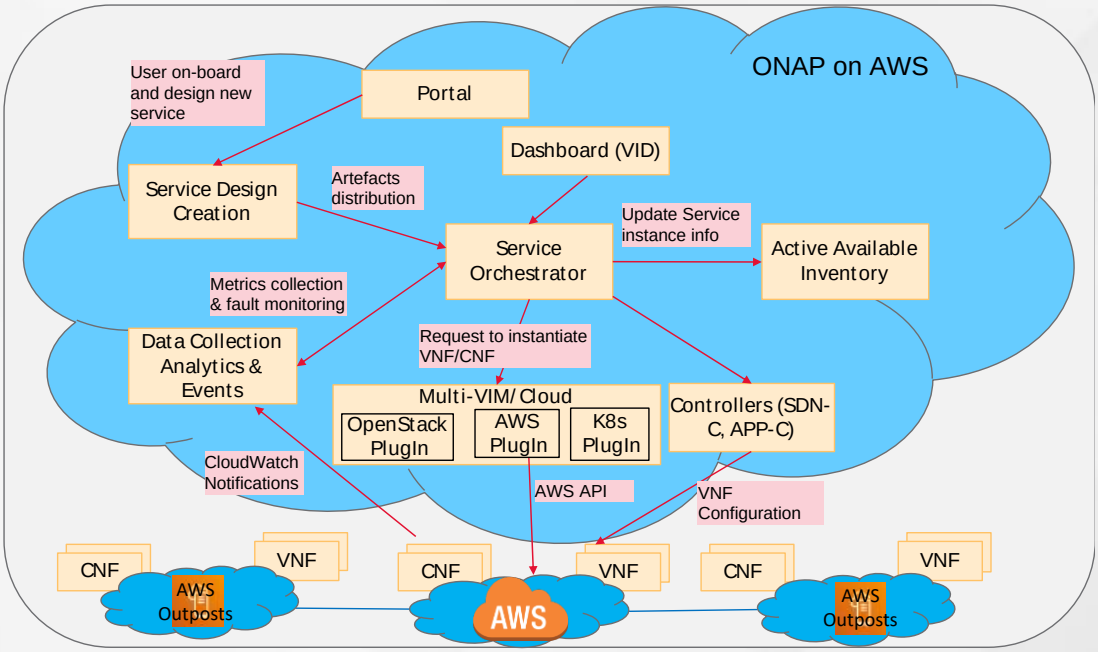



**Figure 6 : Orchestrating workloads on AWS using ONAP**

When it comes to orchestration of VNF workloads on AWS there are multiple options available – each with their distinct set of advantages. The criteria being which type of VNF descriptors should be used – the standard TOSCA mechanism or the native AWS mechanism using the cloud formation templates.

ONAP orchestrator supports onboarding the VNF artifacts on AWS using the standard TOSCA format. In addition, AWS cloud native format such as AWS CloudFormation templates can be used to onboard the VNFs. While the former is part of the opensource distribution, the later could be implemented within the existing ONAP framework as an extension.

## VNF Descriptors using TOSCA templates

Topology and Orchestration specification for Cloud Applications (TOSCA) is a modeling language used to describe the cloud applications. TOSCA intends to provide a fully declarative, standards-based and portable alternative to technology-specific template languages such as Heat Orchestration Templates (HOT) or Amazon CloudFormation Templates.

Using the TOSCA profile for NFV which is extension to the base TOSCA constructs, a network service comprising various VNFs and the topology around it could be designed using new TOSCA constructs like virtual links, connection points etc. ONAP supports onboarding of VNF descriptors in the TOSCA format and subsequent design of the network service which may comprise of multiple VNFs.

The high level ONAP architecture that supports orchestration of VNFs on to several cloud providers is depicted below.

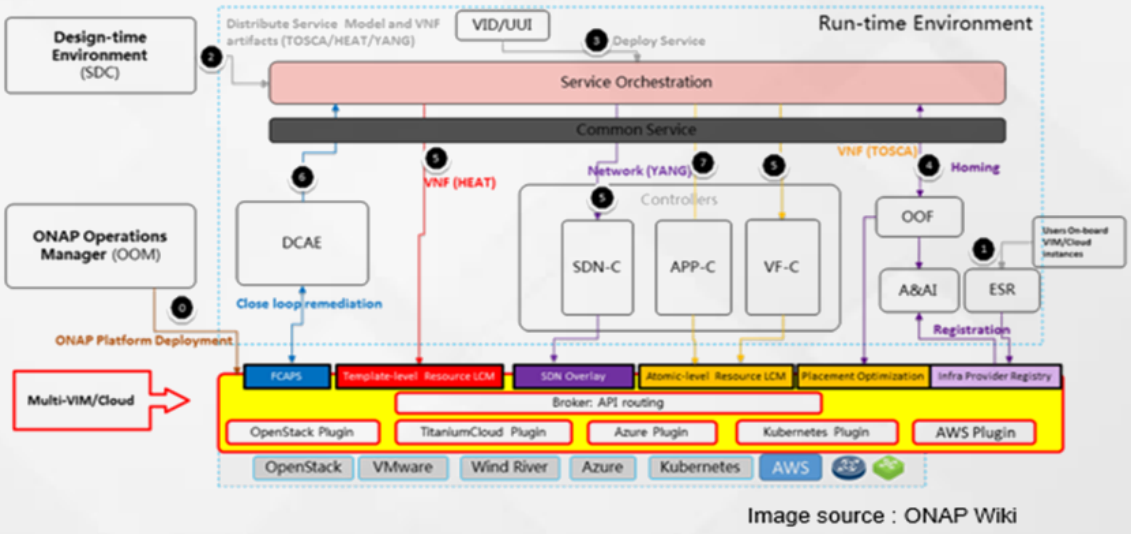



**Figure 7: ONAP Multi VIM/Cloud component architecture**

The Multi-VIM / Cloud component within the ONAP framework supports cloud provider specific plug-ins to orchestrate the VNF workloads within the respective cloud providers. The AWS Cloud plug-in will be part of this Multi-VIM / Cloud component. The high level steps of how TOSCA based VNF descriptors are orchestrated by ONAP on AWS cloud is summarized below:

1. The AWS cloud region is defined in the ONAP ESR / AAI component.
2. The vendor VNF TOSCA artifacts will be on-boarded in SDC component of ONAP. The Network service would be designed in the SDC component and then distributed for deployment.
3. The Virtual Infrastructure Deployment (VID) or the Use case User Interface (UUI) component of ONAP invokes the Service Order workflow of Service Orchestrator (SO) component of ONAP. The SO parses the TOSCA artifacts of the VNFs.

4) SO consults the OOF component to determine the cloud region on which the workload needs to be orchestrated
5) SO shall now invoke the Multi-VIM / Cloud component API for orchestrating the resources in the cloud region

Within the multi-cloud / VIM, the Broker component performs the appropriate API routing by calling the AWS plug-in, which finally calls the AWS APIs to further create the resources in the cloud instance.

Advantage of this approach is that it gives a flexibility of deploying same TOSCA based VNF descriptor in a private cloud as well as AWS public cloud or any other public cloud. Based on the destination cloud, the correct plug-in is chosen by ONAP framework. However, not all the features of AWS are fully utilized in this approach since the network service and VNF descriptors are modelled using TOSCA, which is a normalized descriptor devoid of some native capabilities that the cloud specific descriptor template such as AWS CloudFormation would support.

## VNF Descriptors using CloudFormation templates

In this approach, telecom vendors with their VNFs already certified on AWS using AWS CloudFormation templates could re-use same templates with ONAP and orchestrate the related network services on AWS cloud platform. Overall process is explained in the diagram below.

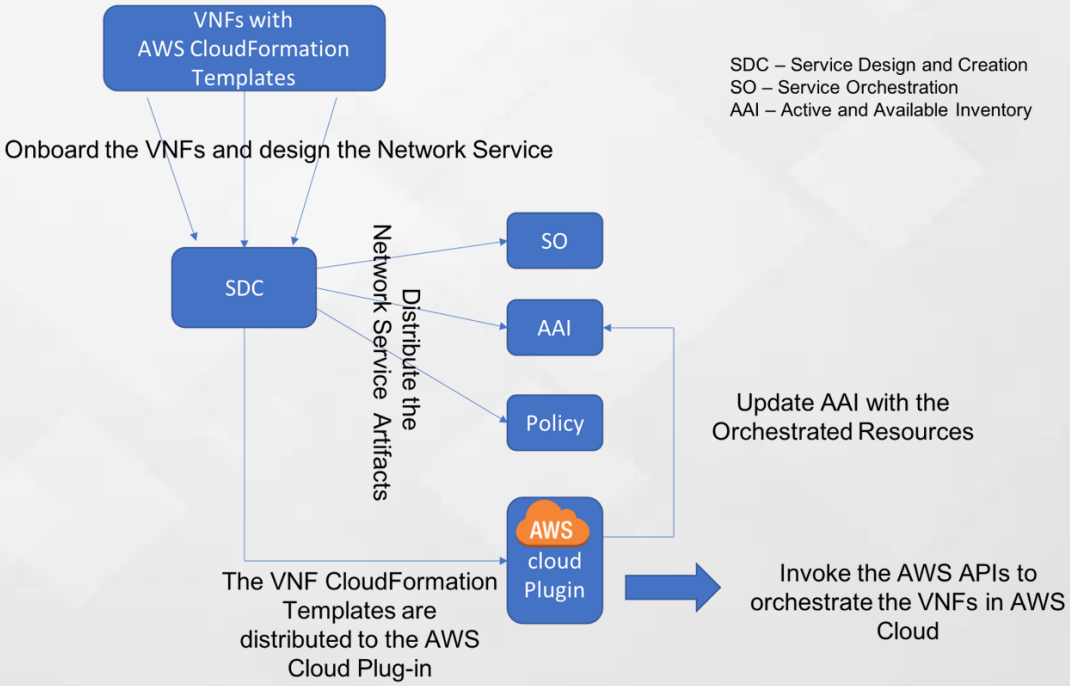



**Figure 8: Deployment of VNFs in ONAP using AWS CloudFormation Templates**

VNFs deployment with ONAP using CloudFormation templates is similar to the VNFs on-boarding using TOSCA based VNF artifacts described above, except for few differences described below –

1. SDC component must be able to accept the AWS CloudFormation template artifacts
2. AWS cloud plug-in registering to SDC to receive the CloudFormation artifacts
3. Configurations in ONAP enabling the multi-Cloud component to redirect the service order instantiation requests to AWS cloud plug-in

With the above extensions to ONAP, we are essentially combining the benefits of using the CloudFormation templates to deploy and manage AWS infrastructure and services along with all standard ONAP capabilities including closed control loop assurance, configuration design using ONAP Control Design Studio (CDS), VNF Life Cycle Management (LCM) automation.

VNF on-boarding process starts with onboarding of AWS CloudFormation VNF templates in SDC and generation of network service artifact that refers to individual CloudFormation template for respective VNF. Here, design features of ONAP SDC will not be used since CloudFormation template would contain the entire network service design.

The AWS Cloud plug-in requires subscribing to these artifacts therefore should be notified by SDC with the details when such artifacts are ready to be distributed.

During the service instantiation, the AWS Cloud plug-in should be able to fetch the artifacts based on the service getting instantiated and then call the AWS API to instantiate the CloudFormation template.

The AWS Cloud plug-in should further call AAI APIs to create the resources and relationships that were instantiated in the AWS through the CloudFormation templates. The AWS Cloud plug-in should ensure that the entire network topology – all the VNFs and their interconnections are created in the AAI. This is very essential since all the other ONAP components would then be able to manage the created resources and all the ONAP enabled automations that are in place for configurations and VNF lifecycle management could be seamlessly utilized. Consequently, the Network Service can be instantiated in AWS utilizing the CloudFormation template features and the service and topology represented in AAI, which would be visible to all the other ONAP components.

## Orchestration of Container Network Functions

ONAP also supports orchestration of Container Network Functions (CNFs), using the Helm based deployment artifacts. Currently, TOSCA specifications are not best suited to represent the deployments of CNFs on the Kubernetes (K8) container cluster since TOSCA specifications do not expose all the K8 capabilities and the standards would take time to support container modeling in TOSCA.

For the Container Network Functions (CNFs) deployment on AWS cloud platform, native services such as AWS CloudFormation, Amazon Elastic Container Service (ECS) and Amazon Elastic Kubernetes Service (EKS) can be utilized. AWS CloudFormation is seamlessly integrated with Amazon ECS and Amazon EKS as described in the AWS cloud native orchestration section below.

# AWS Cloud Native Orchestration of Telco Workloads

Instead of utilizing an external NFV orchestrator such as ONAP, AWS natively supports a host of tools and services to enable orchestration of telco workloads. The section below will describe in detail the different capabilities of AWS Cloud platform to support network workloads design, deployment and management.

## Network Functions and Service Design, On-boarding and Orchestration

AWS cloud platform natively supports various orchestration tools and services listed in Table:1 below that can be utilized individually or in tandem for design, development and management of VNFs and CNFs, automated cross regional deployments in the cloud and hybrid cloud environments, network service creation, customer service onboarding and service life cycle management involving service chaining, deployment and SLA management with service level functions such as provisioning, configuration, orchestration and service assurance. AWS services can be accessed using AWS Management Console (Console), Command Line Interface (CLI), or Software Development Kits (SDKs)/APIs.

| SN | AWS Services | Description | Deployment scenarios |
|---|---|---|---|
| 1 | AWS CloudFormation | Templated approach for the automated and controlled deployment of AWS infrastructure | VNFs and CNFs single and multi-region deployment, configurations, operations and management |
| 2 | AWS OpsWorks | Configuration management service that provides managed instances of Chef and Puppet for the applications configurations | Network function configuration management and continuous application deployment |
| 3 | AWS CodePipeline | CI/CD Orchestration for workloads deployment in DevOps manner | Design, development and deployment of Cloud Native Functions |
| 4 | Amazon ECS | Amazon's own managed Container Orchestration Service for Microservices | Orchestration for Docker containerized applications on a scalable cluster |
| 5 | Amazon EKS | Managed Kubernetes Orchestration Service for Microservices | Orchestration for Kubernetes compatible Docker containerized applications on a scalable cluster |
| 6 | AWS Fargate | Managed compute Engine for Amazon ECS Container Orchestration Service for Serverless Containers | Compute Engine for Amazon ECS to run containers without having to manage servers or clusters |
| 7 | AWS Lambda | Serverless Compute service that automatically manages underlying highly available and scalable compute resources | Events based invocation of software functions to create workflows. Integrates with other AWS services including services listed here allowing you to create a dynamic orchestration layer for auto scaling and auto healing architectures, services creation, and business functions integration |
| 8 | AWS Step Functions | Serverless Orchestration that allows coordination of distributed applications and microservices using visual workflows | Allows you to create Serverless Business Support Systems workflows such as order management, CRM, inventory management, billing and transactions |
| 9 | AWS Service Catalog | Creates and manages a catalog of Products: applications and services and organize them into portfolios for end user consumption | Provides automated mechanism required to make a portfolio of network services available for consumption through a service catalog and allow customers with access to the catalog to create a custom service via a service portal |

| | | | |
|---|---|---|---|
| **6** | **AWS Systems Manager** | **Operations and Maintenance of AWS infrastructure using Amazon CloudWatch dashboards, policy implementation, bulk actions and change, configurations compliance, inventory management, session management, patch management, parameter store, remote management** | **Centrally define, view, investigate, operate, manage your entire infrastructure on AWS encompassing services such as CloudWatch, Trusted Advisor, Personal Health Dashboard** |
| **7** | AWS Config | Assess, audit, and evaluate the configurations of AWS resources. Continuously monitors and records AWS resource configurations. Automate the evaluation of recorded configurations against desired configurations. Simplify compliance auditing, security analysis, change management, and operational troubleshooting | Simplified Operations and Management of infrastructure with faster troubleshooting of the faults due to configuration mismatches. Using Config Rules assess overall compliance and risk status of a configuration, view compliance trends over time and pinpoint which configuration change caused a resource to drift out of compliance |
| **8** | Amazon SNS | Simple Notification Service that coordinates and manages the delivery or sending of messages to subscribing endpoints or clients. Supported protocols: Amazon SQS, HTTP/S, email, SMS, Lambda. | Useful for network operations and management. Works with CloudWatch to send text, email or HTTP/S notifications for the events and alarms |
| **9** | AWS Systems Manager | Operations and Maintenance of AWS infrastructure using Amazon CloudWatch dashboards, policy implementation, bulk actions and change, configurations compliance, inventory management, session management, patch management, parameter store, remote management | Centrally define, view, investigate, operate, manage your entire infrastructure on AWS encompassing services such as CloudWatch, Trusted Advisor, Personal Health Dashboard |

**Table 2: AWS Orchestration Tools and Services**

AWS CloudFormation (CloudFormation) is an Infrastructure as Code *configuration orchestration* tool that enables you to describe and automate provisioning of cloud infrastructure resources including Amazon Virtual Private Cloud (VPC), Amazon Elastic Compute Cloud (EC2), container-related resources like Amazon ECS and Amazon EKS clusters, AWS Lambda serverless compute, Databases and Load Balancers.

CloudFormation works with key concepts such as *templates, stacks* and *change sets* to create, manage and redeploy a collection of related AWS resources, provisioning and updating them in an orderly and predictable fashion to make managing those resources simpler.

- An AWS CloudFormation *template* is a JSON or YAML formatted text file. These files can be saved with extensions such as .json, .yaml, .template, or .txt. CloudFormation uses these templates as blueprints for building your AWS resources.

- When you use AWS CloudFormation, you manage related resources as a single unit called a *stack*. You create, update, and delete a collection of resources by creating, updating, and deleting stacks. When you create a stack, CloudFormation makes underlying service calls to AWS to provision and configure your resources described in the template.

- If you need to make changes to the running resources in a stack, you update the stack. Before making changes to your resources, you can generate a *change set*, which is a summary of your proposed changes. Change sets allow you to see how your changes might impact your running resources, especially for critical resources, before implementing them.

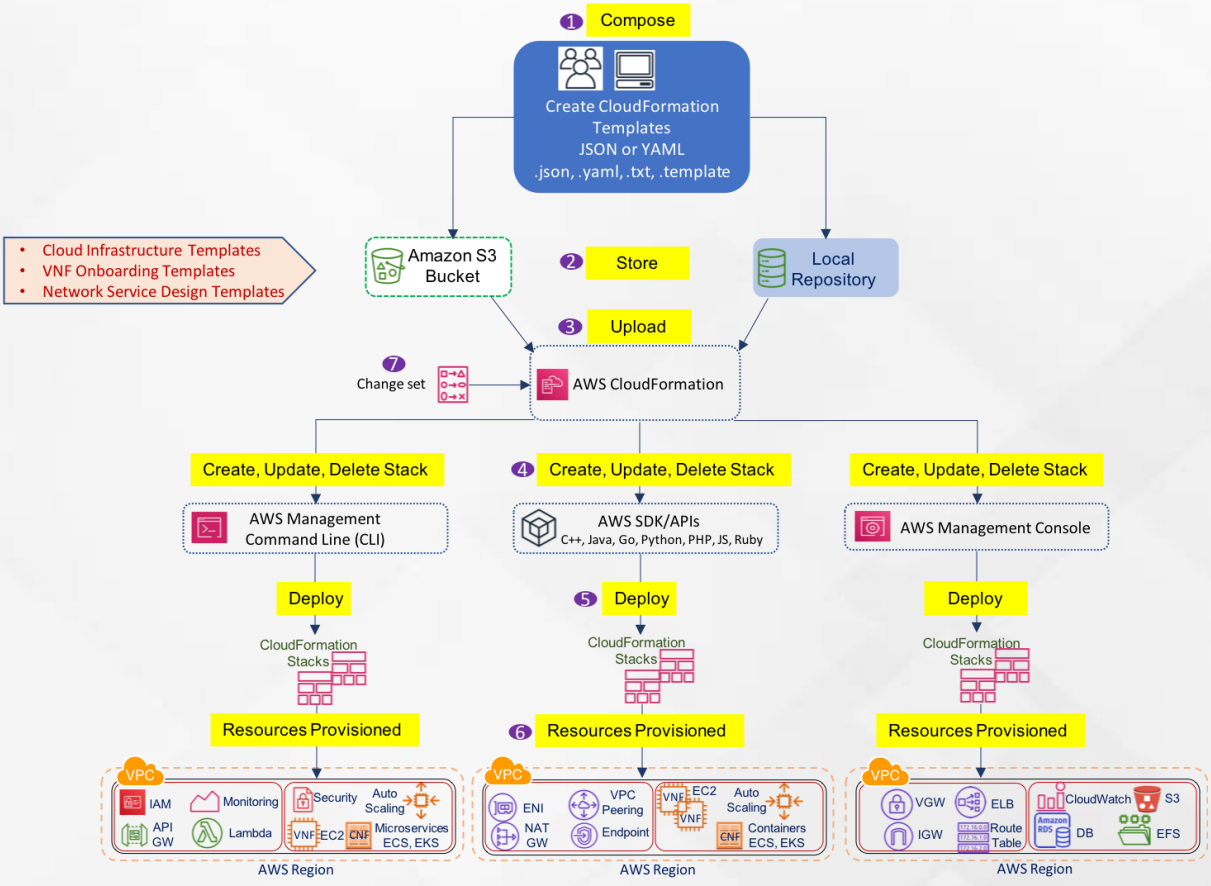



**Figure 9: Deploying AWS Cloud Infrastructure and Services using AWS CloudFormation**

You can also visualize your templates as diagrams and edit them using a drag-and-drop interface with the AWS CloudFormation Designer. You don't need to individually create and configure AWS resources and figure out what's dependent on what. AWS CloudFormation handles all of that. After the AWS resources are deployed, you can modify and update them in a controlled and predictable way, in effect applying version control to your AWS infrastructure the same way you do with your software. Detailed information on the AWS services supported by CloudFormation can be found here.

In the NFVI framework, AWS CloudFormation equates to OpenStack Heat Orchestration Templates (HOT) or TOSCA standard templates that allow you to describe the network topologies, interfaces, components, orchestrations for managing the infrastructure and relationships between the service components for service creation and the policies for the service lifecycle management. AWS CloudFormation templates make it easier to model each VNF in the same universal manner and allowing for easy configuration and deployment of a large number of VNFs.

CloudFormation is often used in conjunction with *configuration management* tools, which are designed to configure the software and systems that run on this infrastructure. This combination provides seamless deployment and configuration of AWS infrastructure and the applications that run on top of it.

AWS OpsWorks (OpsWorks) service provides managed instances of Chef and Puppet, the most popular configuration management tools, that are used with CloudFormation. AWS OpsWorks lets you use Chef and Puppet to automate how VNFs are configured, deployed, and managed across Amazon EC2 instances on AWS cloud platform or on-premises compute environments. AWS OpsWorks has three offerings: (1) AWS OpsWorks for Chef Automate (2) AWS OpsWorks for Puppet Enterprise, (3) AWS OpsWorks Stacks.

AWS CloudFormation and AWS OpsWorks work together to automate applications management on AWS for software configuration management, continuous application deployment, scaling, compliance and monitoring. You can model OpsWorks components (stacks, layers, instances, and applications) inside CloudFormation templates, and provision them as CloudFormation stacks. This enables you to document, version control, and share your OpsWorks configuration. You have the flexibility to provision OpsWorks components and other related AWS resources such as Amazon VPC, Elastic Load Balancing, EC2 Auto Scaling Groups with a unified CloudFormation template or separate CloudFormation templates.

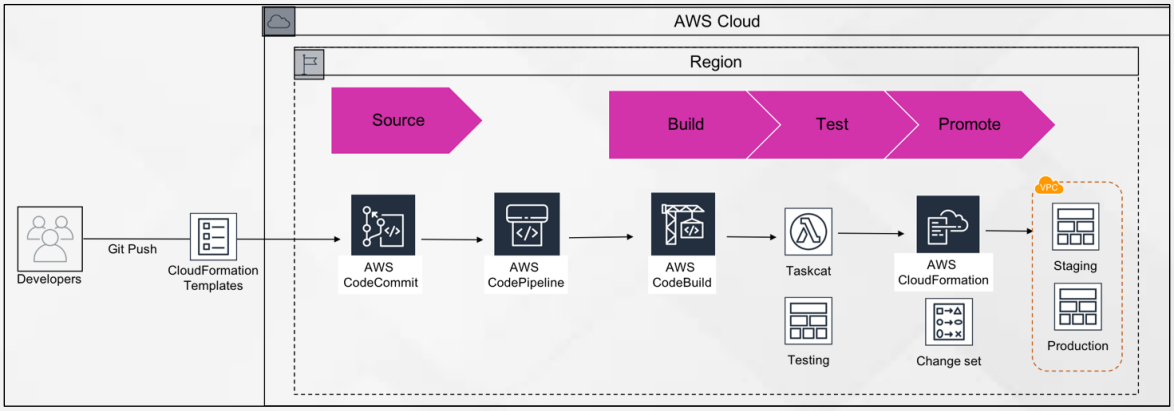



**Figure 10: Continuous Deployment with AWS CodePipeline and AWS CloudFormation**

AWS CodePipeline is a fully managed continuous delivery service, a CI/CD pipeline orchestration tool that enables virtual network functions (VNFs) and cloud-native functions (CNFs) design, continuous integration and deployment in DevOps manner on the AWS cloud infrastructure. It automates application release pipelines - build, test, and deploy phases of the release process every time there is a code change, based on the release model you define, for fast and reliable application and infrastructure updates. AWS CodePipeline provides you with a graphical user interface to create, configure, and manage your pipeline and its various stages and actions, allowing you to easily visualize and model your release process workflow. AWS CodePipeline can pull source code for your pipeline directly from AWS CodeCommit, Amazon ECR, or Amazon S3, and third party services such as **GitHub**. You can also model AWS CloudFormation actions that let you provision, update, or delete AWS resources as part of your release process. CodePipeline can deploy applications to Amazon EC2 instances by using CodeDeploy, AWS Elastic Beanstalk, or AWS OpsWorks Stacks. CodePipeline can also deploy container-based applications to services by using Amazon ECS and Amazon EKS.

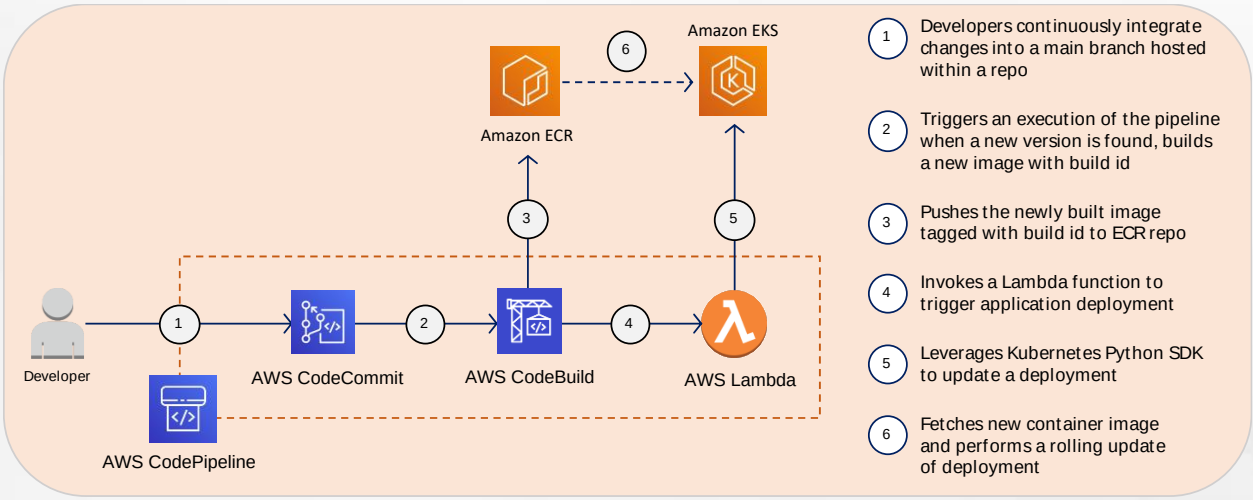



**Figure 11: Continuous Deployment with Amazon EKS and AWS CodePipeline**

## Containers and Microservices Orchestration with Amazon ECS and Amazon EKS

As Telco industry is gravitating towards 5G, focus is growing on design and development of cloud-native network functions (CNFs) for rapid network deployments and faster time to market. In the 5G architecture, Virtualized Radio Access Network (RAN) stack is distributed with virtualized base band unit (vBBU), Control and Data Plane functions (CU/DU) that run on separate Multi-Access Edge (MEC) servers while Radio Unit (RU) is deployed at customer premises for better indoor wireless network coverage. 5G Next Generation Packet Core (NGC) supports Control (CP) and User/Data Plane (UPF) separation with UPF is controlled by Access and Mobility Management Function (AMF) and Session Management Function (SMF). Container based microservices architecture of Control and User Plane functions in the Radio Network (vRAN) Stack and 5G NGC allows flexible deployment of functions at the Edge (MEC) and on the cloud infrastructure for the efficient re-usability of functions. Also, support of network slicing based on modular design and multi-slice connectivity from UEs is one of the key 5G use-cases that service providers will be implementing earlier in their 5G network roll-outs. This distributed architecture also requires common orchestration of Edge and regional cloud infrastructure.

AWS supports Container based and Serverless Microservices deployment models. AWS offers managed container orchestration services such as **Amazon Elastic Container Service** (ECS), **Amazon Elastic Kubernetes Service** (EKS) and **AWS Fargate** that address the most important challenges of microservices architectures such as on-demand resources, programmability, redeploying cloud infrastructure repeatedly and consistently, service orientation, managed services, polyglot and continuous delivery.

Amazon ECS, EKS and Fargate container services are also supported on AWS Outposts hybrid-cloud platform that allows the network operators to implement use-cases such as CUPS (Control and User Plane Separation) architecture with User Plane placed at the Edge cloud closer to the application, which is key to having an efficient 5G core network.

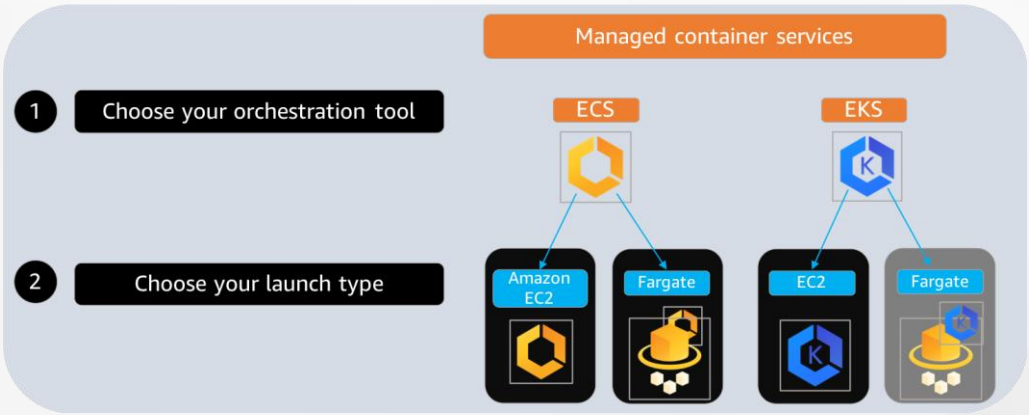



**Figure 12: Managed Containers on AWS**

Amazon Elastic Container Service (Amazon ECS) is a highly scalable, high performance container orchestration service that supports Docker containers and allows you to easily run applications on a managed cluster of Amazon EC2 instances. Amazon ECS container instances are scaled out and scaled in, depending on the traffic volume or the number of incoming requests using EC2 Auto Scaling (ASG) together with Elastic Load Balancing (ALB/NLB).

Amazon ECS eliminates the need to install, operate, and scale your own microservices management infrastructure. With simple API calls, you can launch and stop Docker-enabled applications, query the complete state of your cluster, and access many features like security groups, load balancers, Amazon Elastic Block Store (EBS) volumes, and AWS Identity and Access Management (IAM) roles. ECS is integrated with Amazon CloudWatch service to monitor the container clusters, report events and alarms, collect logs, send notifications and take remedial actions for fault recovery – monitor Amazon ECS.

After a cluster of EC2 instances is up and running, you can define task definitions and services that specify which Docker container images to run on the cluster. Container images are stored in and pulled from container registries (*e.g. Amazon ECR*), which may exist within or outside your AWS infrastructure. To define how your applications run on Amazon ECS, you create a task definition in JSON format. This task definition defines parameters for which container image to run, CPU, memory needed to run the image, how many containers to run, and strategies for container placement within the cluster. Other parameters include security, networking, and logging for your containers.

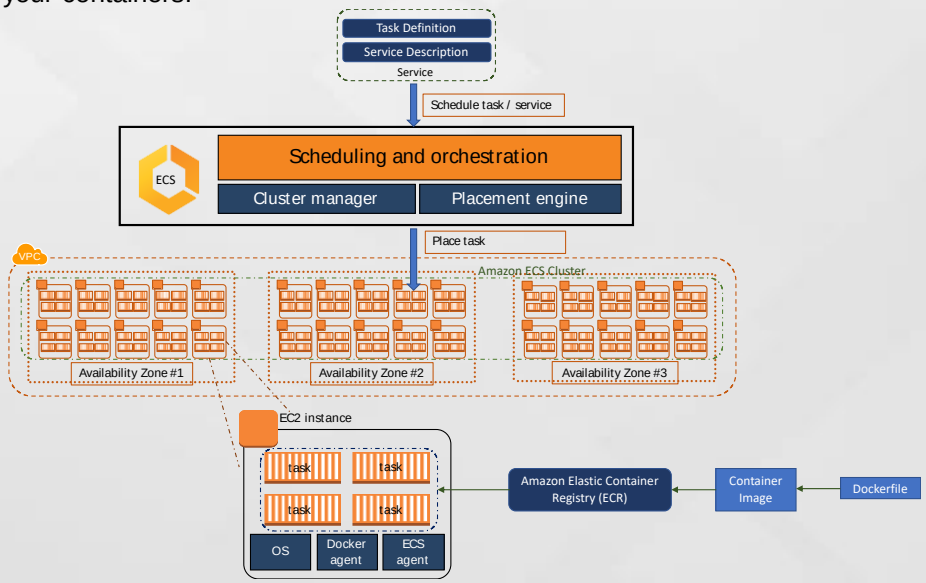
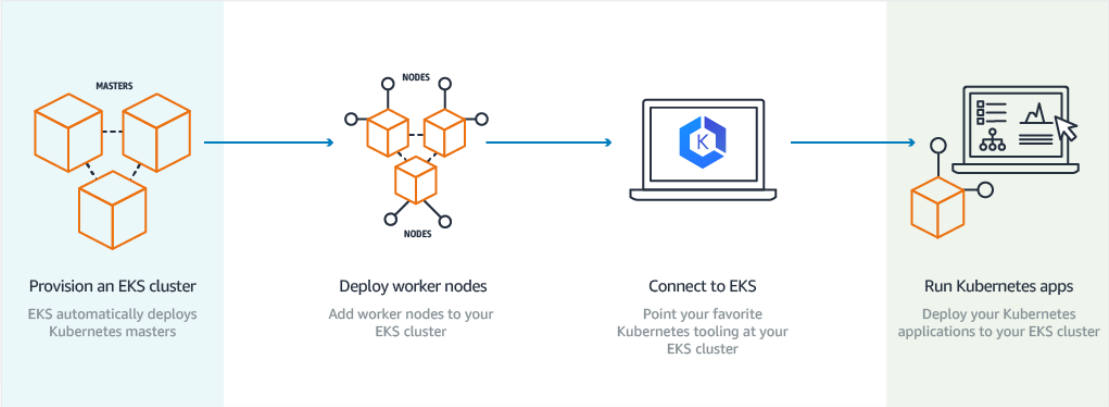



**Figure 13: Amazon ECS Container Cluster Management**

Amazon Elastic Kubernetes Service (Amazon EKS) runs up-to-date versions of the open-source Kubernetes software, so you can use all the existing plugins and tooling from the Kubernetes community. Applications running on Amazon EKS are fully compatible with applications running on any standard Kubernetes environment, whether running in on-premises data centers or public clouds.




Docker images used in Amazon ECS and Amazon EKS can be stored in *Amazon Elastic Container Registry* (Amazon ECR). Amazon ECR is a fully-managed Docker container registry that makes it easy for developers to store, manage, and deploy Docker container images.

## Serverless Microservices with AWS Lambda and AWS Fargate

AWS Lambda (Lambda) is a serverless compute service that lets you run code or a software function without provisioning or managing servers. You pay only for the compute time you consume. There is no charge when your software function is not running. With Lambda, you can execute code for virtually any type of application or backend service with zero administration. Lambda runs your software function on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, code monitoring and logging. Lambda natively supports Java, Go, PowerShell, Node.js, C#, Python, and Ruby code, and provides a Runtime API which allows you to use any additional programming languages to author your functions.

Lambda function can be triggered in response to events from other AWS services such as CloudWatch events or be called directly from any web or mobile application via Amazon API Gateway. Making synchronous calls from API Gateway to AWS Lambda enables the creation of fully serverless applications.

AWS Fargate (Fargate) is a compute engine for Amazon ECS that allows you to run serverless containers so you don't have to worry about provisioning, configuring, and scaling clusters of virtual machines to run containers. With Fargate, you no longer have to worry about provisioning enough compute resources for your container applications. Fargate can launch tens of thousands of containers and easily scale to run your most mission-critical applications.

Amazon ECS has two modes: *Fargate launch type* and *EC2 launch type*. With Fargate launch type, all you have to do is package your application in containers, specify the CPU and memory requirements, define networking and IAM policies, and launch the application.

# Auto Scaling, Auto Healing and Lifecycle Management

AWS load balancing and EC2 autoscaling capabilities let you implement highly available auto healing applications with minimal effort. You can create an *event-driven orchestration layer* to build elastic, fault tolerant and self-healing architectures with an Elastic Load Balancer (ELB), EC2 Auto Scaling Group (ASG), Amazon EC2 compute instances, Amazon CloudWatch, and AWS Lambda services stitched together.

Amazon EC2 Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application. Collection of EC2 instances is called *Auto Scaling Group (ASG)*. Amazon EC2 Auto Scaling helps in maintaining application availability at low cost by automatically adding or removing the EC2 instances according to defined conditions. Dynamic and predictive scaling features of Amazon EC2 Auto Scaling add or remove EC2 instances based on user traffic load and scheduling based on predicted demand automatically. Further details on the other EC2 Auto Scaling features like auto scaling lifecycle and monitoring using health checks, CloudWatch Events and Metrics, CloudTrail logs and SNS notifications can be found here: ASG Lifecycle and Monitoring.

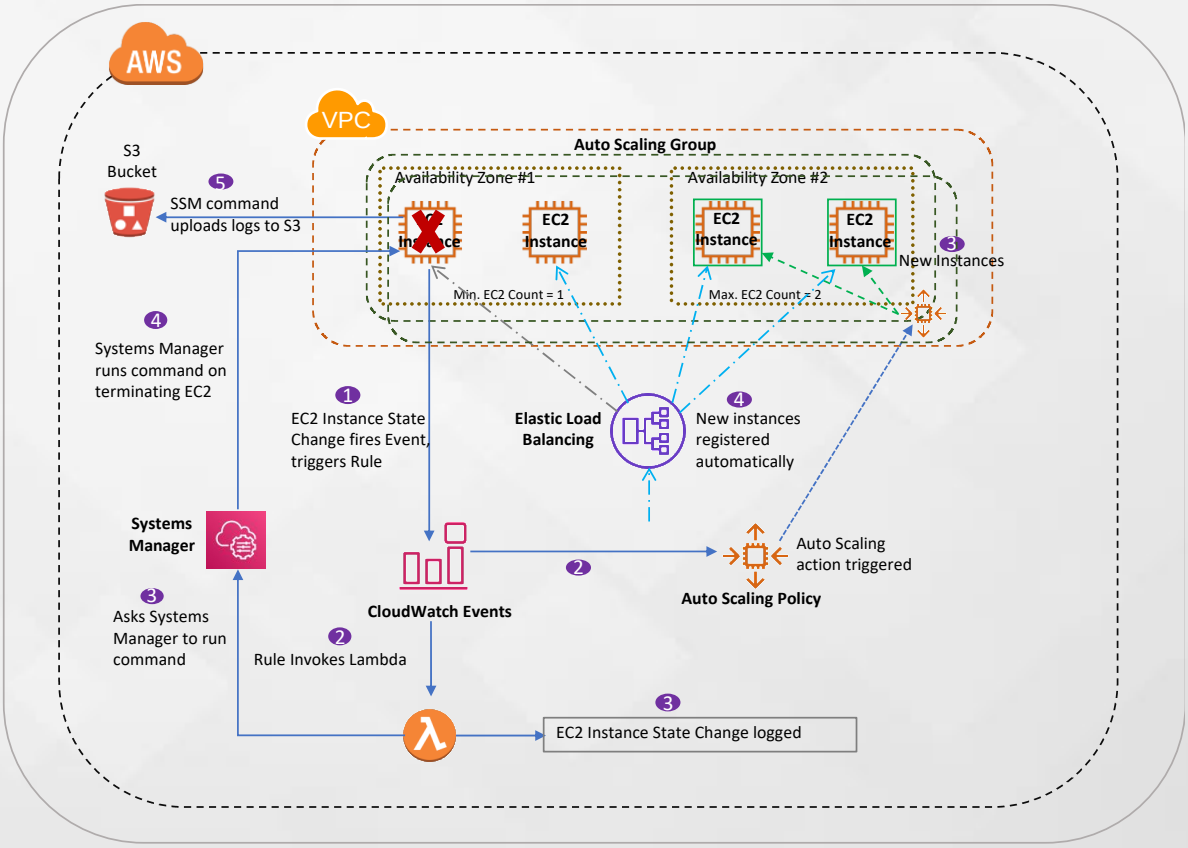



**Figure 14: Event driven Orchestration with CloudWatch, Lambda and EC2 Auto Scaling**

Amazon CloudWatch monitors Amazon EC2 State, Capacity and Auto Scaling, dynamically scales it based on demand, and *Elastic Load Balancing* distributes load across multiple instances in one or more Availability Zones. The measurements collected by Amazon CloudWatch provide Auto Scaling with the information needed to run enough Amazon EC2 instances to deal with the traffic load. *EC2 Auto Scaling* updates the *Elastic Load Balancing* service when new instances are launched or terminated to automatically scale the load-balanced capacity.

You can instantiate, configure, and deploy these important system architecture components in seconds. CloudWatch allows you to create alarms and events which watch metrics and send notifications or automatically make changes to the resources you are monitoring when a threshold is breached. Amazon CloudWatch is further described in the Monitoring, Logging and Performance Management section below.

Elastic Load Balancing supports two types of load balancers that you can select based on the application needs either individually or together. If you need to load balance HTTP requests, use Application Load Balancer. For network/transport protocols (layer4 – TCP, UDP) load balancing, and for extreme performance/low latency applications using Network Load Balancer is recommended.

## Redundancy and Disaster Recovery

AWS Global Infrastructure is designed and built to deliver the most flexible, reliable, scalable, and secure cloud computing environment with the highest quality global network performance available today. The AWS Cloud infrastructure is built around AWS Regions and Availability Zones. An AWS Region is a physical location in the world where we have multiple Availability Zones. Availability Zones consist of one or more discrete data centers, each with redundant power, networking, and connectivity, housed in separate facilities to further reduce single points of failure. Customers who care about high availability and performance of their applications can deploy applications across multiple Availability Zones in the same region for fault tolerance and low latency. Availability Zones are connected to each other with fast, private fiber-optic networking, enabling you to easily architect applications that automatically fail-over between Availability Zones without interruption. AWS managed compute, storage, database, networking, analytics, application and deployment services are inherently built scalable and highly available to significantly minimize the impact on data, system and overall business operations.
Disaster Recovery solutions can be implemented with:

1. Primary infrastructure running in your data center in conjunction with the AWS cloud infrastructure
2. Primary site running in AWS using AWS multi region feature
3. Combination and variation of the DR scenarios below:
    - Backup & Restore: Data backed up and restored using AWS services such as Amazon S3, RDS, EBS
    - Pilot Light: Only minimal critical functionalities in AWS with critical data mirrored/replicated
    - Warm Standby: Fully functional scaled down standby infrastructure in AWS
    - Multi-Site Hot Standby (Active-Active): Fully functional scaled up production infrastructure in AWS

AWS capabilities essential for Disaster Recovery, Backup and Restore are explained in details here.

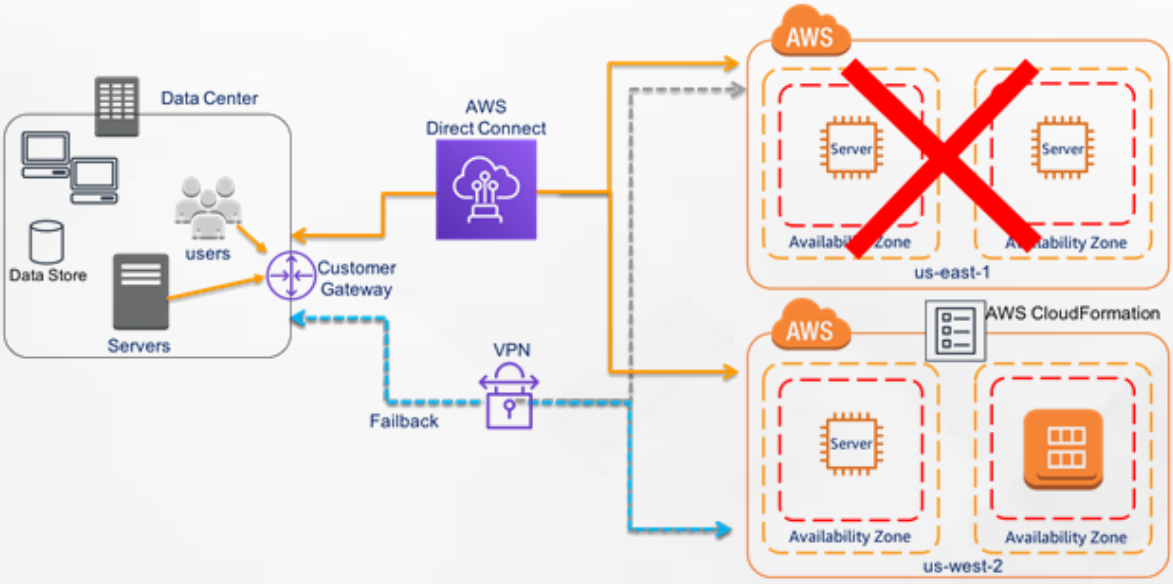



**Figure 15: Multi-Site Replication and Failover**

## Monitoring, Logging and Performance Management

AWS provides various tools that you can use to monitor your AWS Resources. You can configure some of these tools to do the monitoring for you, while some of the tools require manual intervention.

Amazon CloudWatch provides you with data and actionable insights to monitor your applications, understand and respond to system-wide performance changes, optimize resource utilization, and get a unified view of operational health. CloudWatch collects monitoring and operational data in the form of logs, metrics, and events, providing you with a unified view of AWS resources, applications and services that run on AWS, and on-premises servers. You can use CloudWatch to set high resolution alarms, visualize logs and metrics side by side, take automated actions, troubleshoot issues, and discover insights to optimize your applications, and ensure they are running smoothly.

Amazon CloudWatch can be accessed using Amazon CloudWatch console within AWS management console, AWS CLI, CloudWatch API and AWS SDKs. CloudWatch dashboards are customizable home pages in the CloudWatch console that you can use to monitor your resources in a single view, even if those resources that are spread across different Regions. With CloudWatch snapshot graphs and CloudWatch API, you can embed CloudWatch graphs thereby CloudWatch Dashboard outside of the AWS Management Console into your website or third-party monitoring dashboards – see details.

In conjunction with Amazon CloudWatch, you can use various other automated and manual monitoring tools such as AWS CloudTrail log monitoring, AWS Trusted Advisor, AWS Systems Manager, VPC Flow Logs, AWS Personal Health Dashboard, and DNS Logs to monitor your AWS cloud resources and manage their performance. You can use Amazon Simple Notification Service (Amazon SNS) with CloudWatch to send messages when an alarm threshold has been reached. Specifics of these services can be found here.

### ETSI MANO – AWS Cloud Orchestration Capabilities Mapping

Table below maps ETSI MANO functionalities with AWS Cloud native Orchestration capabilities

| Orchestration/Assurance Functionality | AWS components | ONAP component |
|---|---|---|
| **VNF/Service Design** | Amazon VPC, AWS CloudFormation, AWS Opsworks, AWS CodePipeline | Service Design and catalogue (SDC) |
| **VNF on-boarding template** | AWS CloudFormation, Amazon ECS, Amazon EKS | VNF SDK, SDC |
| **VNF instantiation** | Amazon VPC, Amazon EC2, ECR, Amazon ECS, Amazon EKS, AWS Lambda, Fargate, Outposts | O&M Dashboard (VID), Service Orchestrator (SO), Optimization (OOF), Controllers like APPC |
| **VNF termination** | AWS Management Console, CLI, SDK/APIs, AWS CloudFormation | O&M Dashboard (VID), Service Orchestrator (SO), Controllers |
| **VNF healing** | Amazon CloudWatch, AWS Auto Scaling Group, Elastic Load Balancing | Policy, SO , Controllers |
| **VNF scale out / in** | Amazon EC2, Amazon ECS, Amazon EKS, Amazon CloudWatch, Auto Scaling Group, Elastic Load Balancing | Policy, SO , Controllers |
| **Fault Management** | Amazon CloudWatch, Amazon SNS | DCAE |
| **VNF Day 0/1/2 configuration** | AWS OpsWorks – Chef Automate, Puppet Enterprise and OpsWorks Stacks. AWS ISV partner: Ansible | Controllers Design Studio (CDS), Controllers like APPC, VFC |
| **Monitoring** | Amazon CloudWatch, AWS Systems Manager, AWS CloudTrail, VPC Flow Logs, AWS Trusted Advisor, AWS Personal Health Dashboard, Amazon Inspector, Amazon Guard Duty, DNS Logs, Security Hub | DCAE |
| **Log Management** | Amazon EC2, Amazon S3, Amazon CloudWatch, Amazon Kinesis, Amazon Elasticsearch, Amazon EMR, Amazon Kinesis | ONAP shared service - Logging |
| **Performance Management** | Amazon CloudWatch, AWS Systems Manager, AWS Trusted Advisor | DCAE |
| **Network Service (NS) onboarding** | Amazon VPC, AWS CloudFormation | SDC |
| **NS Lifecycle Management – create, delete, heal, scale** | Amazon EC2, Amazon ECS, Amazon EKS, AWS Lambda, Amazon CloudWatch, Auto Scaling Group, Elastic Load Balancing | SO, Controllers like VFC |
| **Service Chaining of VNFs** | Amazon VPC Networking Services incl. Transit Gateway, VPC Peering, VGW, AWS PrivateLink, ENI, Elastic IP | SDC |
| **Service Orchestration** | AWS CloudFormation, AWS Console, CLI, SDKs/APIs, AWS CodePipeline, Amazon EKS, Amazon ECS, AWS Systems Manager, AWS Step Functions, AWS Service Catalog | ONAP Service Orchestration (SO) component |
| **Single pane of glass for all operations** | AWS Console, CLI and SDK/APIs | Portal |

**Table 3: AWS-ONAP Service Mapping**

## Use case and demo

This section provides an example deployment of hybrid-cloud orchestration of workloads demonstrated by Tech Mahindra, AWS and AWS ISV partners at Mobile World Congress Americas in 2019. The setup demonstrates an end-to-end voice call over a VoLTE network with components spread across multi-domain hybrid-cloud infrastructure – 1) a purpose built private cloud infrastructure to host virtualized Radio Access Network (RAN) stack 2) Mobile Packet Core components hosted on AWS public cloud and 3) Virtual IMS and other components hosted on Openstack based private cloud.

Tech Mahindra Telco Network-as-a-Service (NaaS) is industry's first Telco-grade 4G/5G network hosted on a hybrid public cloud environment. The solution brings the "Economics of Cloud" and "Speed of Software" to the Telco Network. Telco Network-as-a-Service can help operators build their entire network or portions of the network on a public cloud infrastructure. This unique offering results in huge CAPEX/OPEX savings for operators by utilizing the scale, existing infrastructure and cost model provided by a public cloud infrastructure. The network is agnostic to any particular vendor and supports geo-redundant deployment. Tech Mahindra's netOps.ai, Network Automation and Managed Services Framework, is geared to accelerate 5G Network adoption by automating all the key network life cycle stages.

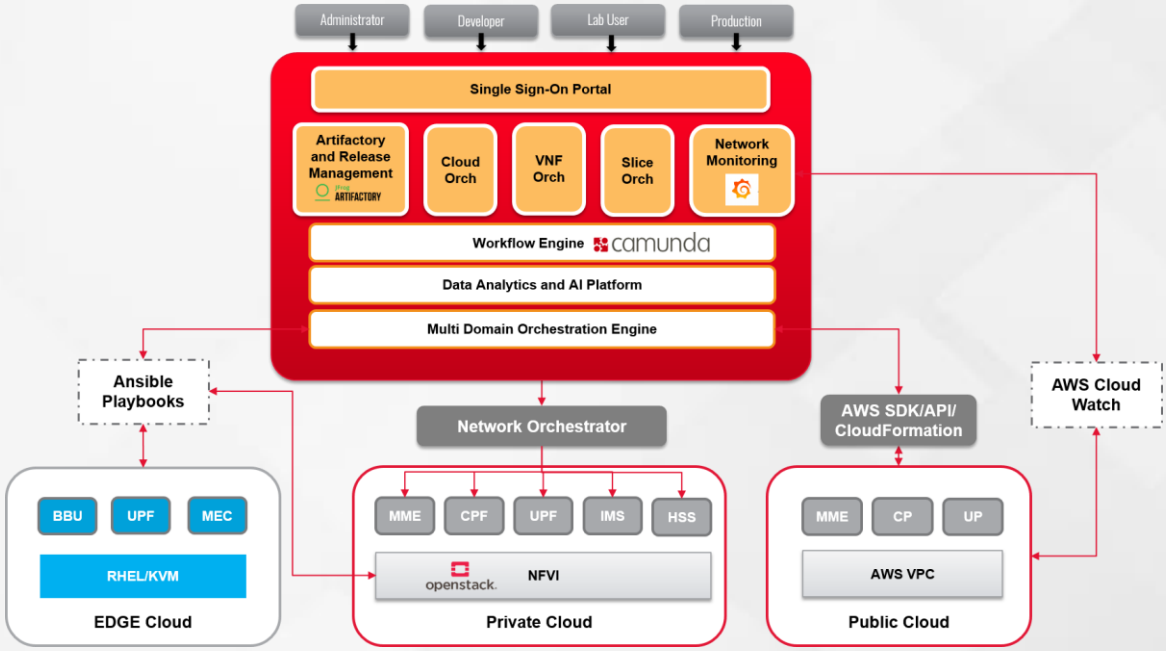



**Figure 16: Multi-domain Hybrid-Cloud Orchestration of Telco Workloads using NetOps.ai**

Following *netOps.ai* orchestrator capabilities were demonstrated at MWC Americas:

- **Telco Cloud Orchestration**
  - Deploying a Telco private cloud instance (based on Redhat Openstack) on-demand
  - Deploying an AWS Virtual Private Cloud (VPC) instance on AWS public cloud

- **VNF Orchestration**
  - VNF orchestration center is vendor agnostic, able to onboard/configure multi-vendor VNFs through Orchestration Engine.

- **Orchestrated VNFs** on hybrid-cloud environments,
  - **Edge cloud**: Orchestration of BBU, UPF and MEC based applications through Ansible Playbooks on a purpose built Edge cloud
  - **Private Cloud**: Orchestration of virtual IMS and F5 firewall via RIFT.io orchestrator
  - **Public Cloud (AWS)**: Orchestration of mobile core (virtual EPC) using AWS SDK

- **Service Orchestration**: Automated the creation of slices on-demand
    - Private LTE slice on a Private Cloud: Deploy and configure the complete slice by creating EPC core and BBU, UPF on Edge cloud
    - New Customer Service Onboarding on a Public cloud (AWS): Configuration of new APN on EPC CORE to be able to perform basic attach functionality
- **NetDevOps**: Implementation of complete DevOps Continuous Integration, Continuous Delivery (CI/CD) pipeline for 5G networks in a multi-vendor environment
- **AI/ML**: Integrated with Tech Mahindra's Data Analytics and AI Platform (DAAIP) platform
    - Predict CPU capacity KPI degradation
    - Anomaly detection performed by netOps.ai framework

## Conclusion

This whitepaper describes multiple approaches to support orchestration of Telco workloads on AWS infrastructure using current opensource orchestrators like ONAP and using AWS cloud native orchestration services. As network operators are evolving their networks towards 5G and adopting private and public cloud infrastructure to deploy cloud-native services on an elastic global scale, orchestration becomes a very critical component. Orchestration should enable Telcos to seamlessly manage various telco workloads such as VNFs and CNFs, generic and supplementary applications utilizing a hybrid-cloud infrastructure comprising of private cloud, purpose built cloud, public cloud and edge cloud. Such orchestration give power to Telcos to launch services with the flexibility and agility to cater to the ever changing needs of their customers, using the most optimal infrastructure and technology.

## Contributors

Contributors to this document include:

- Shonil Kulkarni, Global Partner Solutions Architect, GSII, Amazon Web Services
- Ravi Calyanakoti, Head of Solutions - Network Cloud and Orchestration, Tech Mahindra
- Sharad Shanbhag, Principal Solution Architect, Network Services, Tech Mahindra
- Milind Jalwadi, Principal Solution Architect, Network Services, Tech Mahindra

# References

| S No | Further Reading |
|---|---|
| 1 | AWS Whitepapers and Guides |
| 2 | AWS Blogs Repository |
| 3 | Carrier Grade Mobile Packet Core Network on AWS |
| 4 | Amazon EC2 Overview and Networking for Telecom Companies |
| 5 | Real Time Communication on AWS |
| 6 | Amdocs Optima Digital Customer Management and Commerce Platform in the AWS Cloud |
| 7 | AWS Well-Architected Framework |
| 8 | Overview of Amazon Web Services |
| 9 | Integrating AWS with Multi-protocol Label Switching |
| 10 | Backup and Recovery Approaches Using AWS |
| 11 | AWS Storage Services Overview |
| 12 | AWS Security Best Practices |
| 13 | Introduction to DevOps on AWS |
| 14 | Amazon Virtual Private Cloud Network Connectivity Options |
| 15 | ONAP Wiki : https://wiki.onap.org/ |

Tech Mahindra


www.techmahindra.com

connect@techmahindra.com

www.youtube.com/user/techmahindra09

www.facebook.com/techmahindra

www.twitter.com/tech_mahindra

www.linkedin.com/company/tech-mahindra