Dinesh Kumar, Jim Huang, Arun Emmanuel

Intel Corporation / Amazon Web Services / HCL Technologies

***Reference Architecture to deploy and scale Telco functions in AWS Cloud.***

Table of Contents

[1. Introduction 3](#_Toc55295994)

[2. Test Environment 4](#_Toc55295995)

[3. Prerequisite 5](#_Toc55295996)

[3.1. AWS Account 5](#_Toc55295997)

[3.2. Key Pair 5](#_Toc55295999)

[3.3. Access to source code 5](#_Toc55296000)

[3.4. AMI Images installed with NGIC/ILT\_GEN 5](#_Toc55296001)

[3.5. AWS Resource Quota 6](#_Toc55296002)

[3.6. AWS CloudFormation Templates 6](#_Toc55296003)

[4. Test Environment Deployment 7](#_Toc55296004)

[4.1. Creating the Environment 7](#_Toc55296005)

[4.2. Creating the Instances 7](#_Toc55296006)

[5. Performance Test Execution 8](#_Toc55296007)

[5.1. Manual Test Execution 8](#_Toc55296008)

[5.2. Automated Test Execution 9](#_Toc55296009)

[6. Troubleshooting 9](#_Toc55296010)

[6.1. Traffic not flowing between the VPCs Configure secondary interfaces on the VPCs using the below steps and verify the traffic on the mentioned interfaces 9](#_Toc55296011)

[6.2. Error while creating VPCs using AWS CloudFormation template 10](#_Toc55296012)

[Appendix 11](#_Toc55296013)

[Appendix 1 – Root Template 11](#_Toc55296014)

[Appendix 2 – Templates for creating Instances 12](#_Toc55296015)

[7. References 15](#_Toc55296016)

Acronyms & Abbreviations

|  |  |
| --- | --- |
| OMEC | Open Mobile Evolved Core |
| ONF | Open Networking Foundation |
| VNFs | Virtualized Network Functions |
| VMs | Virtual Machines |
| EPC | Evolved Packet Core |
| CUPS | Control Plane and User Plane Separation |
| MEC | Multi-access Edge Computing |
| AWS | Amazon Web Services |
| UPFs | User Plane Functions |
| RAN | Radio Access Network |
| CP | Control Plane |
| TGW | Transit Gateway |
| VPC | Virtual Private Cloud |
| IoT | Internet of Things |
| NGIC | Next Generation Infrastructure Core |
| TAF | Test Automation Framework |
| 5G | 5th Generation Mobile Network |
| 3GPP | 3rd Generation Partnership Project |
| EC2 | Elastic Compute Cloud |
| ILT-GEN | Traffic Generator/Responder |
| AMI | Amazon Machine Image |
| IGW | Internet Gateway |
| NAT | Network Address Translation |
| eNB | Extended Node Base |
| AS | Application Server |

# Introduction

The Open Mobile Evolved Core (OMEC)[1] is an open source Evolved Packet Core providing both control and user plane functions. It is a project under the umbrella of the Open Networking Foundation (ONF)[2], a Linux Foundation project. The high-performance packet processing core functions were deployed on production networks [3] using commodity servers.

This document provides steps to deploy the Evolved Packet Core reference architecture using OMEC[1] control and user plane functions for telecom core network infrastructure on AWS Cloud. The reference architecture supports telecom application use cases and core network performance optimization from telecom on-premises edge to the cloud region, end to end. The telecom core network infrastructure is modularized in the public cloud so that telecom operators will be able to deploy Virtualized Network Functions (VNFs) on Virtual Machines (VM) right-sized by scale-up method within performance-efficiency thresholds

Reference architecture advances Evolved Packet Core (EPC) [1] toward the 5G system architecture specified by 3GPP [3] and reflects on these cloud and mobile infrastructure design methods:

* Control Plane and User Plane Separation (CUPS) [4] to scale the control plane and user plane resources independently.
* Multi-access Edge Computing (MEC) [5] to support high-performance applications with low latency (less than tens of milliseconds round trip time) requirements
* Resource rightsizing for performance scaling

A high-level view of the reference architecture is shown in Figure 1. It consists of AWS Cloud in a region and telecom on-premises interconnected through AWS Direct Connect links. The AWS Cloud region hosts Control Plane and User Plane VNFs and the telecom on-premises may host AWS Cloud Outposts which runs User Plane Functions (UPFs) along with Radio Access Network (RAN).



Figure 1: High-level View of the Reference Architecture

The reference architecture provided cloud formation templates to deploy and scale User Plane and control plane VNFs in AWS Cloud Region. Reference Architecture also support deploying optional instances for traffic generators and traffic responders in separate VPCs connecting via transit gateways to simulate the on-prem RAN with user devices and target applications and internet. These optional components may be replaced to actual on-prem components and connected to the AWS Cloud running user and control EPC functions via direct connect.

This document provides detailed steps to deploy these components on AWS Cloud and validate the user plane function’s performance and scale using the automation test framework.

# Test Environment

The Figure 2 below provides a high-level view of the reference architecture deployed in AWS for end to end performance scale testing with varying number of user plane function instances running inside a single VPC and connecting to traffic generator and traffic responder instances.

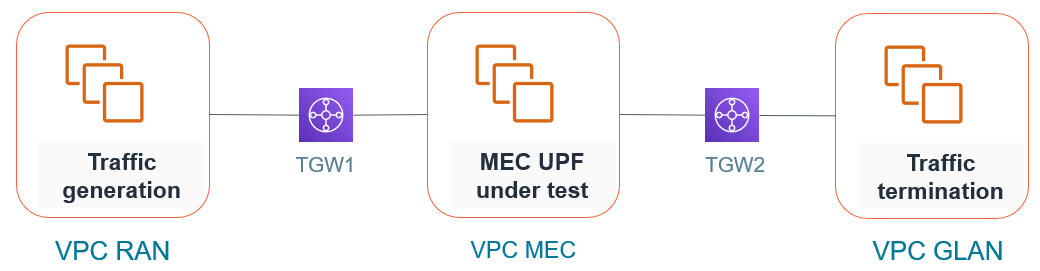


Figure 2: High-level View of the Reference Architecture

The Figure 3 below provides a example deployed reference architecture in AWS with user plane functions and traffic generator and responder instances running in separate VPC inside AWS cloud.

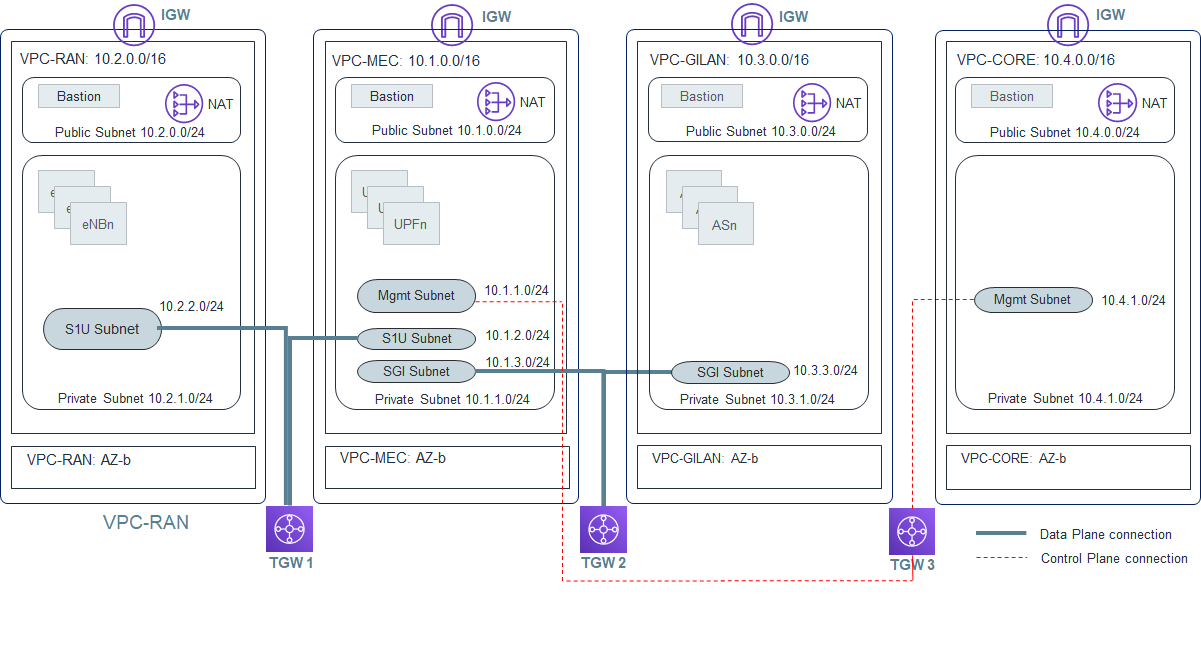


Figure 3: Detailed Reference Testbed Diagram

# Prerequisite

## AWS Account

## Key Pair

Create a KeyPair in .pem format to use with the CloudFormation template. This will be required while creating the EC2 instances and accessing the instances using SSH client. User can create a new keypair by following <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html#prepare-key-pair>

## Access to source code

Contact Intel Business Development team for the access to OMEC Next Generation Infrastructure Core (NGIC), traffic generator and responder (ilt\_gen), and Test Automation Frameworks (TAF) source code. Access is required for downloading the source code for building the AMI Images in the following section.

## AMI Images installed with NGIC/ILT\_GEN

User will create two AMIs to use with the EC2 instances. One AMI is installed and configured with ilt\_gen and the other one is installed and configured with the NGIC code base for user plane and control plane functions.

* Create AMI Image with NGIC code base installed for user plane and control plane functions

1. Create an EC2 instance with instance type c5n.2xlarge and ubuntu 18.04  
 2. Login as root user. ***sudo su***  
 3. Clone the source code using the command ***git clone*** [***https://<user\_name>@ilpm.intel-research.net/bitbucket/scm/vccbbz/ngic-rtclnx.git***](https://%3cuser_name%3e@ilpm.intel-research.net/bitbucket/scm/vccbbz/ngic-rtclnx.git)

4. ***cd ngic-rtclnx***  
 5. Change the branch to ngic-lts: ***git checkout -b ngic-lts***  
 6. Pull the latest code from the ngic-lts branch: ***git pull origin ngic-lts***  
 7. Install NGIC and its dependencies using included install script [***./install.sh***](mailto:./install.sh)

8. Configure user plane VNF configurations

8.1 ***cd dp***

8.2 Uncomment the line ‘CFLAGS += -DSIMU\_CP’ in the Makefile if it is commented out As we are simulating the control plane sessions and those sessions are prepopulated before user plane traffic starts so uncommenting this will prepopulate all sessions for user plane function to start receiving the processing the user plane traffic

8.3 Uncomment ‘CFLAGS += -DLINUX\_TRANSPORT’ in config/ng-core\_cfg.mk file to run the user plane function in Linux kernel mode instead of dpdk.

***8.4 make clean  
 8.5 make***

9. Create an AMI from the above EC2 instance installed with NGIC code base using <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/creating-an-ami-ebs.html>

* Create AMI Image with ilt\_gen code base installed for traffic generator and responder instances

1. Create an EC2 instance with instance type c5n.2xlarge and ubuntu 18.04  
 2. Clone the git repo using the command ***git clone*** [***https://<user\_name>@ilpm.intel-research.net/bitbucket/scm/vccbbz/il\_trafficgen.git***](https://%3cuser_name%3e@ilpm.intel-research.net/bitbucket/scm/vccbbz/il_trafficgen.git)  
 3. Change the branch to tgen-lts : ***git checkout -b tgen--lts***  
 4. Pull the latest code from the tgen-lts branch: ***git pull origin tgen-lts***  
 5. Install ILT\_GEN and its dependencies: follow steps in [***./install.sh***](mailto:./install.sh)

6. Create an AMI from the above EC2 instance installed with NGIC code base using <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/creating-an-ami-ebs.html>

## AWS Resource Quota

Verify required resource quotas are available in the AWS account for the region. The quota required for the resources are listed below

* VPC = 4 (VPC Count)
* EIP = 8 (2 \* VPC Count)
* NAT Gateways = 4 (1 \* VPC Count)
* Internet Gateways = 4 (1 \* VPC Count)

## AWS CloudFormation Templates

AWS CloudFormation templates are in AWS managed Github: <https://github.com/aws-samples/aws-intel-5g>. Clone the repo to access the latest AWS CloudFormation templates for creating the resources in AWS.

Create S3 bucket and upload the templates to the S3 bucket

* Create a S3 bucket <https://s3.console.aws.amazon.com/s3/home>
* Upload all AWS CloudFormation template files to the newly created S3 bucket
* Make a note of the S3 Bucket name for future use

# Test Environment Deployment

Test Environment deployment is done in two steps. In first step user create the infrastructure resources for deploying the EC2 instances, including VPCs, Subnets, Transit Gateways, Route tables, Routes IGW, NAT Gateways etc.

In the second step, user deploy EC2 instances in the VPC environment created in the first step

## Creating the Environment

Environment Resources are created using the 5g-env-root.template . Follow the below procedure for creating the deployment environment

* Get the URL for 5g-env-root.template from the S3 bucket we created recently
* Go to the **Services** and select **CloudFormation**
* Select **Create Stack** -- > **With new resources (Standard)**
* Select
  + Template is Ready
  + Amazon S3 URL
* Provide the URL for 5g-env-root.template in the space provided
* Click Next
* Provide the parameters required for creating the environment. Default values will be shown and used if not provided by the user (Parameter details are available in Appendix 1)
* Click Next
* Create Tags for the deployment if needed
* Click Next
* Verify the parameters
* Click Create Stack
* Wait for the stacks to get created as it may take 5-7 minutes for completing the operation

## Creating the Instances

Below is the list of the AWS CloudFormation templates available for EC2 instance deployments

* 5g-server-cp.template – For deploying EC2 instances in CP VPC
* 5g-server-enb.template - For deploying EC2 instances in ENB VPC
* 5g-server-gilan.template - For deploying EC2 instances in GiLAN VPC
* 5g-server-mec.template - For deploying EC2 instances in MEC VPC

Parameters required for CloudFormation varies with the templates. Please refer Appendix 2 for the various parameters required for the CloudFormation templates. One parameter required for all the above templates is the Amazon Machine Image ID. Here user can provide either Amazon provided public image AMI, or the custom-made private AMI ID installed with the required packages. User will need to manually install all the required packages if Amazon provided public image AMI option is selected.

# Performance Test Execution

The tests can be executed manually or by using the test automation framework

## Manual Test Execution

* Login to MEC instance through Bastion server and then execute the below commands

***sudo su***  
***cd /home/ngic-rtclnx/dp***  
***source ../setenv.sh  
./run.sh linux***   
For exiting the application use ‘q’ followed by ‘X’

* Login to eNB instance through Bastion server and then execute the below commands  
  ***sudo su***

***cd /home/il\_trafficgen/pktgen***

***source ../setenv.sh***

***./il\_nperf.sh -g***  
Wait for the application to start.

Start the Traffic with ‘start 0’ command

pktgen> ***start 0***

When done, quit the application with quit command

pktgen>***quit***

* Login to GiLAN instance through Bastion server and then execute the below commands  
  ***sudo su***

***cd /home/il\_trafficgen/pktgen***

***source ../setenv.sh***

***./il\_nperf.sh -r***

Wait for the application to start. Execute the below command for starting the traffic

pktgen> ***start 0***

For quitting the application  
pktgen>***quit***

Traffic from eNB instance and GiLAN can be run simultaneously to exercise traffic in both directions.

Performance results are displayed on screen in the ENB and GiLAN Instances .

## Automated Test Execution

* Installation

Clone the latest source code from bitbucket using the below command  
***git clone*** [***https://<user\_name>@ilpm.intel-research.net/bitbucket/scm/vccbbz/ngic-cloud-setup.git***](https://%3cuser_name%3e@ilpm.intel-research.net/bitbucket/scm/vccbbz/ngic-cloud-setup.git)  
***cd ngic-cloud-setup/performance\_benchmarking/scripts***  
[***./install.sh***](mailto:./install.sh)   
install.sh will create the required folder structure inside the ngic-cloud-setup/performance\_benchmarking folder, install python3.8 and its dependencies and   
creates a virtual environment named venv\_ngic and install all the required python libraries.

* Configure Key File

Copy the <id\_rsapem.pem> key file to scripts folder  
Update the key file name KEY\_FILE\_NAME field in the scripts/config.py file

* Test Execution

Activate the virtual environment installed using the below command  
***cd ngic-cloud-setup/performance\_benchmarking***  
***source venv\_ngic/bin/activate***  
Navigate to scripts folder  
***cd scripts***  
Start the test using below command  
***python scaling\_schema.py <options>***

* Refer the README available in the Test Automation Framework Repository  
  Path: ‘ngic-cloud-setup/performance\_benchmarking/scripts/README’ for available options and configurations for running tests using Test Automation Framework

# Troubleshooting

## Traffic not flowing between the VPCs Configure secondary interfaces on the VPCs using the below steps and verify the traffic on the mentioned interfaces

On eNB Instances:  
 ***sudo ifconfig ens6 <ens6\_ip>/24 up***   
 ***sudo ip route add 10.1.2.0/24 via 10.2.2.1 dev ens6***  
 Verify traffic on ens6 by running ***sudo tcpdump ens6***

On MEC Instances:  
 ***sudo ifconfig ens6 <ens6\_ip>/24 up***  
 ***sudo ifconfig ens7 <ens\_ip>/24 up  
 sudo ip route add 10.2.2.0/24 via 10.1.2.1 dev ens6  
 sudo ip route add 10.3.3.0/24 via 10.1.3.1 dev ens7***  
 Verify traffic on ens6 by running ***sudo tcpdump ens6***  
 Verify traffic on ens7 by running ***sudo tcpdump ens7***

On GiLAN Instance  
 ***sudo ifconfig ens6 <ens6\_ip>/24 up  
 sudo ip route add 10.1.3.0/24 via 10.3.3.1 dev ens6***  
 Verify traffic on ens6 by running ***sudo tcpdump ens6***

Note: ens6\_ip and ens7\_ip will be available in the Private IPv4 addresses section of the corresponding instances on AWS console  
Traffic can be verified by running ilt\_gen for uplink traffic and ilt responder for the downlink traffic. Ping test can also be used for verifying the traffic. Run the below ping test to confirm the traffic flow  
 eNB s1u ip 🡨--🡪 MEC s1u ip  
 MEC sgi ip 🡨--🡪GiLAN sgi ip  
 eNB Management ip 🡨--🡪 GiLAN Management IP  
 MEC Management ip 🡨--🡪 CP Management IP

## Error while creating VPCs using AWS CloudFormation template

If encounter some error messages as given below

“Embedded stack arn:aws:cloudformation:us-east-2:943592889863:stack/NestedStack39-VPCeNBStack-KPSOLESSAJBS/b2ff8390-0ddf-11eb-b920-06c54738e514 was not successfully created: The following resource(s) failed to create: [LambdaExecutionRole, VPCeNB, VPCeNBInternetGateway]”

Check the available quota for VPCs, EIP, NAT Gateways and Internet Gateways  
  
Syntax: ***Bold Italics*** is used to represent commands

# Appendix

### Appendix 1 – Root Template

A picture containing graphical user interface, application

Description automatically generated

### Appendix 2 – Templates for creating Instances

VPC eNB Instance

Table

Description automatically generated

VPC MEC Instance

Graphical user interface, application

Description automatically generated

VPC GiLAN Instance

Graphical user interface, application

Description automatically generated

# References

[1] Open Mobile Evolved Core, <https://www.opennetworking.org/omec/>

[2] Open Networking Foundation, <https://www.opennetworking.org/>

[3] 3GPP, “System architecture for the 5G System,” TS 23.501 V16.4.0, March, 2020.

[4] 3GPP, “Architecture enhancements for control and user plane separation of EPC nodes,” TS 23.214 V6.0.0, June, 2019.

[5] ETSI, “MEC in 5G networks,” The European Telecommunications Standards Institute white paper, June, 2018.

[6] 3GPP, “Technical Specification Group Services and System Aspects; Network architecture (Release 14),” TS 23.002 V14.0.0, September, 2016.