VPC Deployment

# Prerequisite

### AWS Account

The user should be having a valid AWS account. If the user doesn’t have a valid AWS account, user can create one before going forward

### Key Pair

Create a KeyPair in .pem format to use with the cloudformation template. This is required while creating the EC2 instances and while 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 team for the access to NGIC/ILT GEN/TAF source code . Access is required for downloading the source code for building the AMI Images in the following steps.

### AMI Images installed with NGIC/ILT\_GEN

Here we need to create two AMIs to use with the EC2 instances. One is installed and configure with ilt\_gen and the other one is installed with the NGIC code base

* AMI Image installed with NGIC

Follow the below steps for creating the NGIC installed AMI Image  
 Create an EC2 instance with instance type c5n.xlarge  
 Clone the repo using the command root@ngic#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)  
 Change the branch to ngic-lts : root@ngic#git checkout -b ngic-lts  
 Pull the latest code from the ngic-lts branch: root@ngic# git pull origin ngic-lts  
 root@ngic#cd ngic-rtclnx  
 Install NGIC and its dependencies: [root@ngic#./install.sh](mailto:root@ngic#./install.sh)

Configure SIMU CP  
 root@ngic#cd dp  
 Uncomment the CFLAGS += -DSIMU\_CP for enabling fake cp config in the Makefile  
 Uncomment CFLAGS += -DLINUX\_TRANSPORT in config/ng-core\_cfg.mk file  
 root@ngic#make clean  
 root@ngic#make

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>

* AMI Image installed with ilt\_gen

Follow the below steps for creating the ILT\_GEN installed AMI Image  
 Create an EC2 instance with instance type c5n.2xlarge  
 Clone the repo using the command root@ngic#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)  
 Change the branch to tgen-lts : root@ngic#git checkout -b tgen--lts  
 Pull the latest code from the tgen-lts branch: root@ngic# git pull origin tgen-lts  
 root@ngic#cd il\_trafficgen  
 Install ILT\_GEN and its dependencies: [root@ngic#./install.sh](mailto:root@ngic#./install.sh)

### 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)

### Cloudformation Templates

Cloud formation templates are available in the code commit repo. Clone the repo to access the latest cloudformation templates for creating the resources in AWS.   
Follow the below steps for creating S3 bucket and upload the templates to the S3 bucket

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

# Resource Deployment

Resource deployment will be done in two steps. In first step we are creating the environment for deploying the EC2 instances. It including VPCs, Subnets, Transit Gateways, Route tables, Routes etc.

In the second step, we are deploying EC2 instances in the VPC environment created already 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

There are multiple cloudformation templates for deploying instances in different VPCs. Below are the cloudformation templates available for EC2 instance deployments

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

Parameters required for cloudformation varies with the templates. Check 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 we can provide either Amazon provided public image AMI or the custom-made private AMI ID installed with the required packages. The user needs to install the required packages if he chose the Amazon provided public image AMI

# Manual Configuration

### Updating the config files

The user needs to configure the ilt\_gen /NGIC/App server for executing the tests. User can skip this step if the tests are done using the test automation framework.

# Running the tests and verifying the results

User can run the tests manually or by using the test automation framework

### Manual Process

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

root@ngic#cd /home/il\_trafficgen/pktgen

root@ngic# source ../setenv.sh

root@ngic#./il\_nperf.sh -g  
Wait for the application to start. Execute the below command for starting the traffic

pktgen> start 0

For quitting the application  
pktgen>quit

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

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

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

root@ngic#cd /home/il\_trafficgen/pktgen

root@ngic# source ../setenv.sh

root@ngic#./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

Performance numbers are available in the ENB Instance terminal

### Using the Test Automation Framework

* Installation

Clone the latest source code from bitbucket using the below command  
root@ngic#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)  
root@ngic#cd ngic-cloud-setup/performance\_benchmarking/scripts  
[root@ngic#./install.sh](mailto:root@ngic#./install.sh)  
This will create the required folder structure inside the ngic-cloud-setup/performance\_benchmarking folder  
Install python3.8 and its dependencies  
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  
root@ngic# cd ngic-cloud-setup/performance\_benchmarking  
root@ngic# source venv\_ngic/bin/activate  
Navigate to scripts folder  
root@ngic#cd scripts  
Start the test using below command  
root@ngic#python scaling\_schema.py <options>

* Options Supported:

--create\_config : For creating new config files in ngic-cloud-setup/performance\_benchmarking/templates/generated\_config\_files  
 --copy\_config : For copying the config files to the remote instance  
 --execute\_test : For executing the test cases  
 --pps : Updating pps values in the cfg files  
 Usage : --pps 200000  
 --flows : Update flows in the cfg files  
 Usage: --flows 50000  
 --pkt\_size : Updating packet size in cfg files  
 Usage: --pkt\_size 128  
 --test\_duration : Update test\_duration in config file  
 Usage: --test\_duration 30  
 --session\_duration : Update session wait time before verifying the services status, after starting the services  
 Ideal session duration are 10 sec for AWS and 30 sec for SRIOV setup  
 Usage: --session\_duration 10  
 --instance: instance id used for test execution. Multiple instance id can be provided  
 Usage: --instance 1 --instance 3  
 --eth\_stats: Used for collecting the eth stats for debugging purpose  
 --no\_of\_instances : To provide the number of instances used for testing  
 Usage : --no\_of\_instances 3  
 Same as --instance 1 --instance 2 --instance 3  
 --setup : To provide the setup name. Values are AWS, SRIOV  
 Usage: --setup AWS  
 --trials: Number of test iteration in a single test execution  
 Usage: --trials 3  
 --clear\_report: To mark the start of a combined report generation

--create\_report: Create an Excel report by combining all the csv files generated after calling the --clear\_report  
 Also create a text file combine all the Eth Stats file

Note: Use --create\_config --copy\_config along with --pps, --flows, --pkt\_size, --test\_duration for generating and uploading the updated configuration files to the remote instance  
All the above options are optional. Default values will be available in the scripts/config.py  
Kindly avoid using --create\_config --copy\_config with SRIOV setup

* Sample test execution commands:

python scaling\_schema.py --instance 1 --execute\_test --eth\_stats --clear\_report --session\_duration 10 python scaling\_schema.py --flows 50000 --pps 200000 --pkt\_size 128 --instance 1 --create\_config  
 --copy\_config --execute\_test --eth\_stats --session\_duration 10

python scaling\_schema.py --flows 50000 --pps 200000 --pkt\_size 128 --instance 1 --create\_config   
--copy\_config --execute\_test --session\_duration 5 --trials 1

python scaling\_schema.py --create\_report

* Reports:

One CSV file each will be generated for every iteration of the test execution.Single execution with 3 trials will generate 3 csv files in the report/csv\_reports folder

--create\_report will generate a combined report in the reports/excel\_reports directory

Eth stats files will be created and the same availabe in the ether\_stats/stats folder with option --eth\_stats.

The combined report will be availabe in the ether\_stats/generated\_report with --create\_report option

* Templates

Base templates used for generating the .cfg files are available in the templates/base\_templates folder Update the template when there is some change in the cfg files

# Appendix

### Appendix 1 – Root Template

Graphical user interface, text, application

Description automatically generated

### A picture containing graphical user interface, application Description automatically generated

### Appendix 2 – For creating Instances

VPC eNB Instance

Graphical user interface, text, application, email

Description automatically generated

VPC CP Instance

Graphical user interface, text, application, email

Description automatically generated

VPC MEC Instance

Graphical user interface, application, table

Description automatically generated

VPC GiLAN Instance

Graphical user interface, application

Description automatically generated