**README**

# **Dependencies**

Currently the control plane for the COB setup is installed in Ubuntu via ansible.

To get started with ansible, please follow the official documentation:

- https://docs.ansible.com/ansible/latest/user\_guide/intro\_getting\_started.html

Apart from a basic understanding of ansible, you should have a working

environment with:

\* Ubuntu 18.04 or newer

\* python3

\* python3-venv

\* ansible (instructions on how to install are below)

# **Prerequisites**

## Install ansible on your local machine

You need to install the ansible python module on your local machine. There is no

need to install ansible on the remote hosts or targets. To install the tested

version of ansible, including all other needed python modules, please use a

virtual environment and the requirements.txt provided in this repo. If you are

running Ubuntu on your local machine and have `python3` and `python3-venv`

already installed, you can run:

```

python3 -m venv my-virtualenv

```

To active the the created virtualenv run:

```

source my-virtualenv/bin/activate

```

And finally to install all needed python modules into the activated virtualenv run:

```

pip install -r requirements.txt

```

After you have followed all three steps from above, running the command `pip

freeze` should look exactly like this:

```

ubuntu@localhost:~$ pip freeze

ansible==2.8.5

asn1crypto==1.1.0

cffi==1.13.0

cryptography==2.7

dnspython==1.16.0

Jinja2==2.10.3

MarkupSafe==1.1.1

pycparser==2.19

PyYAML==5.1.2

six==1.12.0

## Running docker on your target machine behind a proxy

If your target machine is running behind a proxy or does not have direct access

to the internet and more specifically to the ubuntu apt repositories, you need

to add this proxy also for docker. To allow docker running on the target machine

to pull images from dockerhub (which we need to pull all the base images), you

have to modify the docker service like documented here:

- https://docs.docker.com/config/daemon/systemd/#httphttps-proxy

To allow docker containers to install new packages during runtime (which is

needed to install the proper kernel header files to build kernel modules in our

case) you also need to add your proxies for the docker client like documented

here:

- https://docs.docker.com/network/proxy/

# **Control plane setup**

The Control plane for the Central Office in a Box (COB) setup contains the below components running in docker containers:

– One or multiple accel-ppp container

– One Redis server in a container

– One PFCP BNG CP in a container

– One Radius server in a container

– One mysql database in a container

Whenever any of the ‘accel-ppp’ containers receive the session request, it communicates with the ‘Radius’ container for the user authentication and authorization. The Radius server looks up for available IP addresses in the mysql database to assign the IP address for the user. During the session establishment the messages are published to the ‘Redis’ container, which forwards the messages to the ‘PFCP BNG CP’ container. The ‘PFCP BNG CP’ container has the association with the ‘PFCP BNG UP’ containers, which are running in the Data plane server. The ‘PFCP BNG CP’ container forwards the command to write specific forwarding rules into the DPDK-BNG containers running in the Data plane.

## Ansible deployment

The ansible playbook ‘deploy\_bng\_control\_plane.playbook.yaml’, in this branch of the repository will install all needed components on your target machine for the Control plane and also apply the proper network configuration to them. To fit the deployment to your environment, you need to create and adapt the ansible inventory file and add control plane specific variables with unique names and its values to the ‘inventories’ directory. For our deployment we have `vm-inventory-mysql.yml` file inside the ‘inventories’ folder which deploys all the required components with their proper values. A brief description of the variables of the inventory file in provided below:  
  
1. hosts – the target control plane name <HOST NAME>

2. vars – different way of deployment and the ansible interpreter needs to be python3

– ‘run\_mode’ can have three values ‘deploy’, ‘clean’, ‘clean deploy’.

‘deploy’ is set when the components of the Control plane are deployed for the first time. To clean the old containers ‘run\_mode = clean’ and to cleaning and restart all the containers ‘run\_mode=clean\_deploy’

– ‘mysql\_root\_password’ is the password to access the database

– ‘mysql\_database’ is the name of the sql database

– ‘radius\_mysql\_user’ is the name of the radius container database

– ‘radius\_mysql\_password’ is the password to access the radius part of the database

– ‘radius\_mysql\_user’ is the name of the radius container database

– ‘redis\_host’ is the machine where the redis server is running

– ‘redis\_port’ is the port where the redis server is listening for messages

– ‘bngc\_ip\_addr’ is the IP address of the PFCP BNG CP container

– ‘net\_prefix’ is the subnet of the IP address range

– ‘bngu1’’ is the first PFCP BNG UP container running in the data plane with the IP address ‘bngu\_ip’ and NAS identifier ‘nas1’

3. docker networks values – the different containers need to be accessible for each other. We create a docker subnetwork for the communication between the docker containers. The Users are recommended to use the same values as in the file.

4. Radius variables – The radius containers reads and verifies the requesting user from a mysql database. To reach the database (which is also inside a container), the access to the mysql database is required. So the username and the password is provided to access the database are provided in the inventory file.

5. Redis container values – the redis container listens to port 6379 for published messages from the ‘PFCP BNG CP’

6. bngu\_endpoints – for the ‘PFCP BNG CP’ to communicate with ‘PFCP BNG UP’ containers you can update the IP addresses of the PFCP containers but all the PFCP containers needs to be in the same subnet to talk to each other (both PFCP BNG CP and PFCP BNG UP).

7. CP instances – these are specific values for different ‘accel-ppp’ containers. The ‘mac’ addresses should be the same as the DPDK BNG UL MAC address. The ‘outer\_tag’ is the ‘inner\_vlan’ of the configured vlan in the pon-manager. The IP\_RANGE is also used to set up the radius ip pool.

NOTE: Before running the ansible playbook as described below, make sure you have the correct host key fingerprint of your target machine added to your local known\_hosts file (the ansible ssh connections will fail otherwise). To run the ansible playbook after you followed all the steps above, just execute the below command on your local machine:

To start the control plane for the first time on your target machine, please change directory to ‘docker-compose-cp’ and execute the below command:

```

ansible-playbook -vv -i inventories/vm-inventory-mysql.yml deploy\_bng\_control\_plane.playbook.yaml -u ubuntu -l <target CP hostname> -e setup\_all=yes -e run\_mode=deploy

```

To restart the control plane and clean up the old containers execute the command below:

```

ansible-playbook -vv -i inventories/vm-inventory-mysql.yml deploy\_bng\_control\_plane.playbook.yaml -u ubuntu -l <target CP hostname> -e setup\_all=yes -e run\_mode=clean\_deploy

```

- `-e setup\_all=yes`: this is an external variable set in the playbook itself. With the value set to ‘yes’ all the containers (accel-ppp, Redis, Radius, PFCP BNG CP, mysql) will be created and started. There are also variables like → ‘setup\_docker, setup\_redis, setup\_mysql, setup\_radius, setup\_pfcp\_cp, setup\_accel\_pppd’ where you can restart individual containers by setting the value to ‘yes’

- `-i inventory`: specifies the inventory file that contains all the target

hosts and the environment specific variables

- `deploy\_bng\_control\_plane.playbook.yaml`: is the name of the ansible playbook

inside of this repo that will be executed

- `-u ubuntu`: will set the username that is used to login to the remote server

to `ubuntu` (change this if you are using another user to access your server)

- `-l <Target dataplane>`: will limit the ansible playbook run on your server specified in your inventory file

======================================================================

**# PFCP USER PLANE**

The PFCP application has the User Plane side (PFCPU) which communicates with DPDK IP PIPELINE (The virtual BNG’s) and also have the association with the PFCP BNG CP. PFCP BNG UP connects to the telnet counter part of the DPDK and forwards the commands to write the the packet forwarding rules inside the pipelines. Each UL/DL pair of BNG containers connect to an individual PFCP UP. So there can be multiple PFCPU containers running and communicating with the single PFCPC in the control plane.

Before we start the PFCPU container the DPDK BNGS needs to be up and running. Below is the link where you can find the instructions how to start the dpdk containers in the dataplane.

https://gitlab.bisdn.de/ansible/ansible-vbng/-/tree/dell\_630\_configs

## Writing specific data plane specific PFCP BNG UP container configs

The bngu folder has its own inventory files that are used to specify the data plane configuration. The file `bng-2-up-inventory.yml` has the values from the dataplane configs. If you have different port numbers, mac addresses etc. for your dataplane then you have to update this file with the right numbers. Right now we have only two BNG's running in the data plane so we create two different PFCP BNG UP containers to communicate with them.

1. hosts – the target data plane name <HOST NAME>

2. vars – different way of deployment and the ansible interpreter needs to be python3

– ‘run\_mode’ can have three values ‘deploy’, ‘clean’, ‘clean deploy’.

‘deploy’ is set when the components of the Control plane are deployed for the first time. To clean the old containers ‘run\_mode = clean’ and to cleaning and restart all the containers ‘run\_mode=clean\_deploy’

– ‘dpdk\_host’ is the hostname of the dataplane

– ‘bngc\_ip\_addr’ is the IP address of the PFCP BNG CP container

– ‘net\_prefix’ is the subnet of the IP address range

– ‘container\_name’ is the name of the PFCP BNG UP as there can be multiple PFCP BNG UP containers so they need to have unique names.

– ‘bngu\_ip\_addr’ is the ip address of PFCP BNG UP

– ‘upstream\_dpdk\_port’ is the port number of the DPDK Uplink container

– ‘downstream\_dpdk\_port’ is the port number of the DPDK Downlink container

– ‘gateway\_mac\_address’ is the MAC address of the gateway to the CORE network

– ‘gateway\_ip\_address’ is the IP address of the gateway to the CORE network

– ‘downstream\_mac\_address’’ is the MAC address of the DPDK downlink container

NOTE: Before running the ansible playbook as described below, make sure you have the

correct host key fingerprint of your target machine added to your local

known\_hosts file (the ansible ssh connections will fail otherwise).

To run the ansible playbook after you followed all the steps above, just

execute the below command on your local machine:

# **Data plane PFCP BNG UP**

To start the PFCP BNG UP containers for the first time on your target machine, please change directory to ‘docker-compose-cp/bngu’ and execute the below command:

```

ansible-playbook -vv -i inventories/bng-2-up-inventory.yml deploy\_bngu\_containers.playbook.yaml -u ubuntu -l <Target dataplane> -e run\_mode=deploy

```

To restart the control plane and cleaning up the old containers execute the below command:

```

ansible-playbook -vv -i inventories/bng-2-up-inventory.yml deploy\_bngu\_containers.playbook.yaml -u ubuntu -l <Target dataplane> -e run\_mode=clean\_deploy

```

- `-i inventory`: specifies the inventory file that contains all the target

hosts and the environment specific variables

- `deploy\_bngu\_containers.playbook.yaml`: is the name of the ansible playbook

inside of this repo that will be executed

- `-u ubuntu`: will set the username that is used to login to the remote server

to `ubuntu` (change this if you are using another user to access your server)

- `-l <Target dataplane>`: will limit the ansible playbook run on your server specified in your inventory file

For more information on ansible please check the official documentation here:

https://docs.ansible.com/ansible/latest/user\_guide/intro\_getting\_started.html

NOTE:: This ansible\_playbook also creates a vxlan interface to connect the

control plane and the data plane after all the docker containers are created.

### Helpful commands for debugging

To see the debug output of the dpdk-ip-pipeline CLI installing forwarding rules in the UL\_VF and the DL\_VF:

```

docker logs -f <container name>

```

# Helpful links:

- https://docs.docker.com/get-started/

- https://docs.docker.com/compose/gettingstarted/