Demo setup steps for Frankfurt

1. OOF

Configure the OOF with the below configDB details and then deploy.

Edit this file oom/kubernetes/oof/values.yaml.

configDbUrl: http://<ip>:<port>
configDbUserName:
configDbPassword:
configDbGetCellListUrl: 'api/sdnc-config-db/v3/getCellList'
configDbGetNbrListUrl: 'api/sdnc-config-db/v3/getNbrList'

2. SON-Handler MS

Steps for creating dmaap topics

The following dmaap topics need to be present in the running DMAAP instance:

1.PCI-NOTIF-TOPIC-NGHBR-LIST-CHANGE-INFO

2.unauthenticated.SEC_FAULT_OUTPUT

3.unauthenticated.SEC_MEASUREMENT_OUTPUT

4.DCAE_CL_RSP

The topics can be added by logging into the message-router-kafka container and executing the following command:

Create Topic
curl --header "Content-type: application/json" --request POST --data '{"owner": "","txenabled": false,"topicName": "<topic-name>"}' http://<message-router-ip>:3904/topics/create

ConfigDB setup:

The steps to set up configDB can be found in the attachment below.

configdb_setup.txt

Steps to deploy SON-Handler:

Log-in to the DCAE-bootstrap container.

The SON-Handler blueprint file can be found in blueprints/k8s-sonhms.yaml.

Copy the input file to the blueprints/ directory. The input file can be found below:

k8s-sonhms-inputs.yaml

Execute the following command to deploy Son-Handler:

Deploy component
cfy install -b sonhms -d sonhms -i <inputsFilePath> <blueprintFilePath>

To undeploy SON-Handler:

Undeploy component
cfy uninstall sonhms

To delete the blueprint:

Delete blueprint
cfy blueprints delete sonhms

3. SDN-C (SDN-R)

The configuration changes to the 2 property files are required based on the environment:

1. Execute the following command to log into the sdnc container
    a. kubectl exec -it dev-sdnc-0 bash
2. Once in the docker, edit the first property file (/opt/onap/ccsdk/data/properties/sdnr-oofpcipoc-api-dg.properties)
3. Make following configuration changes:
    a. If you are using RAN simulator and performing Netconf mount, update the following flag to true.
        i. ransim-mounted=true
    b. If ConfigDB is deployed, update the following flag to true.
        i. configdb-deployed=true
    c. Update IP address for ConfigDB.
        i. configdb.url=http://<ip-address>:8080/api/sdnc-config-db/v3
    d. Update the controller url for sdnc.
        i. controller.url=https://sdnc.onap:8443
4. Edit the second property file as below(/opt/onap/ccsdk/data/properties/sdnr-CMNotify-api-dg.properties)
    a. If you are using RAN simulator and performing Netconf mount, update the following flag to true.
        i. ransim-mounted=true
    b. Update IP address for ConfigDB.
        i. configdb.url=http://<ip-address>:8080/api/sdnc-config-db/v3
    c. Update the controller url for sdnc.
        i. controller.url=https://sdnc.onap:8443

The following is a temporary fix for dmaap-listener library files, and please do the following for the dmaap-listener docker:

1. Execute the following command to all dmaap-listener POD's to copy relevant template map files
    a. kubectl exec -it -n onap dev-sdnc-sdnc-dmaap-listener-5b8d94959b-27mwf bash
2. Once in the docker, create folder : /opt/app/dmaap-listener/lib/
3. Copy following files to the above folder (/opt/app/dmaap-listener/lib/)
    a. cp /opt/onap/sdnc/dmaap-listener/lib/pci-changes-from-policy-to-sdnr.map /opt/app/dmaap-listener/lib/.
    b. cp /opt/onap/sdnc/dmaap-listener/lib/anr-changes-from-policy-to-sdnr.map /opt/app/dmaap-listener/lib/.
    c. cp /opt/onap/sdnc/dmaap-listener/lib/anr-pci-changes-from-policy-to-sdnr.vt /opt/app/dmaap-listener/lib/.

4. Policy

There following control policies that are being used in the usecase. They have to be created and pushed into the policy.

Log in to policy-pdp container.

pci.json

son.json

pci_push.json

son_push.json

pciBson.json

pciBson_push.json

sonBpci.json

sonBpci_push.json

Execute the following commands:

Create and push Modify Config policy.

Modify Config policy
curl -k --silent --user 'healthcheck:zb!XztG34' -X POST "https://policy-api:6969/policy/api/v1/policytypes/onap.policies.controlloop.operational.common.Drools/versions/1.0.0/policies" -H "Accept: application/json" -H "Content-Type: application/json" -d @pci.json curl --silent -k --user 'healthcheck:zb!XztG34' -X POST "https://policy-pap:6969/policy/pap/v1/pdps/policies" -H "Accept: application/json" -H "Content-Type: application/json" -d @pci_push.json

Create and push Modify Config ANR policy.

Modify Config ANR policy
curl -k --silent --user 'healthcheck:zb!XztG34' -X POST "https://policy-api:6969/policy/api/v1/policytypes/onap.policies.controlloop.operational.common.Drools/versions/1.0.0/policies" -H "Accept: application/json" -H "Content-Type: application/json" -d @son.json curl --silent -k --user 'healthcheck:zb!XztG34' -X POST "https://policy-pap:6969/policy/pap/v1/pdps/policies" -H "Accept: application/json" -H "Content-Type: application/json" -d @son_push.json

Create and push PCI-controlloop-guard(Controlloop-denial) policy:

PCI-guard
curl -k --silent --user 'healthcheck:zb!XztG34' -X POST "https://policy-api:6969/policy/api/v1/policytypes/onap.policies.controlloop.guard.coordination.FirstBlocksSecond/versions/1.0.0/policies" -H "Accept: application/json" -H "Content-Type: application/json" -d @pciBson.json curl --silent -k --user 'healthcheck:zb!XztG34' -X POST "https://policy-pap:6969/policy/pap/v1/pdps/policies" -H "Accept: application/json" -H "Content-Type: application/json" -d @pciBson_push.json

Create and push ANR-controlloop-guard policy:

ANR-guard
curl -k --silent --user 'healthcheck:zb!XztG34' -X POST "https://policy-api:6969/policy/api/v1/policytypes/onap.policies.controlloop.guard.coordination.FirstBlocksSecond/versions/1.0.0/policies" -H "Accept: application/json" -H "Content-Type: application/json" -d @sonBpci.json curl --silent -k --user 'healthcheck:zb!XztG34' -X POST "https://policy-pap:6969/policy/pap/v1/pdps/policies" -H "Accept: application/json" -H "Content-Type: application/json" -d @sonBpci_push.json

5. RAN-Sim

Steps to build the RAN-Sim Controller setup:

1. Clone and Checkout Ran-Sim Controller(use Dublin branch)

git clone https://github.com/onap-oof-pci-poc/ran-sim.git

cd ran-sim/ransim/

2. Copy ‘m2_settings.xml’ from ‘<YOURFOLDER>/ransim/’ to <HOME>/.m2/ folder as settings.xml file after updating environment specific configurations.

If the file already exists in <HOME>/.m2/ folder, then merge this file content into the existing file appropriately.

Note: settings.xml entries are mandatory to include dependencies.

3. Open the terminal and navigate to the '<YOURFOLDER>/ransim/ransimctrlr'

4. The following capabilities can be modified in the 'ransim.properties' file based on user capabilities and configurations.

File directory:

<YOURFOLDER>/ransim/ransimctrlr/packages/base/src/files/install/servers/ransim/bin/ransim.properties

a) serverIdPrefix:

Netconf server comman prefix (use default value present in the file)

b) numberOfCellsPerNCServer:

Maximum number of cells that can be handled in a single netconf server(use default value present in the file).

c) numberOfProcessPerMc:

Maximum number of netconf servers that can run in a single machine(use default value present in the file, which is for a machine of 8 GB RAM)

(A single netconf server uses approximately 350MB).

d) numberOfMachines:

Maximum number of machines available(use default value present in the file).

e) GridSize: (Applicable only for HONEYCOMB representation)

The number of cells that can be accommodated along one side of the cluster for an auto-generated layout.

However, it has no relevance now, as the initial layout is generated from a file. So use default value as 1.

f) strictValidateRansimAgentsAvailability:

A boolean value to check if any RAN-Sim agents are running (use default value present in the file).

g) sdnrServerIp:

Neglect(Refer point 5 to update SDNR details).

h) sdnrServerPort:

Neglect(Refer point 5 to update SDNR details).

i) sdnrServerUserid:

Negelct (Refer point 5 to update SDNR details).

j) sdnrServerPassword:

Negelct (Refer point 5 to update SDNR details).

k) maxPciValueAllowed:

maximum value of the physical cell Id. (Default is 503).

l)dumpFileName

Location of the dumpfile to load the topology.

The dump file is loaded from '<YOURFOLDER>/ransim/docker/config'. A sample dump file - 'sample.json' contains deatils of 1000 cells.

For the controller to access the dump file from the the above location use the path - /tmp/ransim-install/config/sample.json

5. The SDNR details can be updated in the file '<YOURFOLDER>/ransim/docker/docker-compose.yml'.

```
environment:
  - SDNR_IP={SDNR IP address}
```

- SDNR_PORT={SDNR port number}
- SDNR_USER={SDNR user ID}
- SDNR_PASSWORD={SDNR user password}

6. Run the following command in the terminal

mvn clean install

7. After successful build navigate to '<YOURFOLDER>/ransim/docker' directory.

8. Run the following three commands in the terminal:

'mvn prepare-package'

'docker build -t onap/ransim-demo ransim-docker'

'docker-compose up'

Note: Use docker compose version 1.6.0 or above.

9. Access the GUI using the following url in the web browser:

'http://<yourIP>:8081/ransimui/index.html

10. Access the swagger ui using the following url:

'http://<yourIP>:8081/ransim/api/swagger-ui.html

11. Try the following url in Websocket client extension in the web browser to verify the connectivity:

'ws://<yourIP>:8081/ransim/RansimAgent/hai'

Steps to check the RAN-Sim Controller logs

1. Open the console and enter the following command:

'docker ps'

2. Once the container details is displayed enter the following command:

docker exec -it ransim bash

3. Navigate to '/opt/app/policy/servers/ransim/logs' using the cd command.

4. Execute the following command to check the logs:

'tail -f ransim-rest.log'