Deploy DMaaP message router in nonrtric

This page is out of date.

Please see the relevant page for the latest release: e.g. Release I - Run in Docker

This article helps you to deploy the NON-RT-RIC components using docker compose.

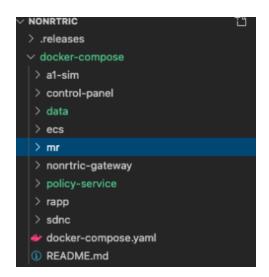
- Prerequisite
- Installation
- NON-RT-RIC without A1 Controller, with dmaap-mr
- Init Script
- Manual test
- Message format

Prerequisite

- Docker
- Docker Compose

Installation

Clone nonrtric and go to nonrtric/docker-compose/



You can see a few folders, each of them container docker-compose file for different components. You have the option to install the nonrtric with/without A1 controller/dmapp-mr based on your requirement.

NON-RT-RIC without A1 Controller, with dmaap-mr

withDmaap_nosdnc/docker-compose.yml

This file is to create nonrtric system without sdnc a1-controller, with dmapp-mr

nosdnc with dmaap-mr

 $\label{locker-compose} \mbox{$\tt docker-compose.yaml -f control-panel/docker-compose.yaml -f policy-service/docker-compose.yaml -f mr/docker-compose.yml up -d } \mbox{$\tt docker-compose.yaml -f mr/docker-compose.yml up -d } \mbox{$\tt docker-compose.yml up -d } \mbox{$\tt docker-compos$

You can see the docker containers with docker ps,

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAM
a884db69f8c7	nexus3.onap.org:10001/onap/dmaap/dmaap-mr:1.1.18	"sh startup.sh"	12 seconds ago	Up 11 seconds	0.0.0.0:3904-3905->3904-3905/tcp	dma
ap-mr 55671d890037	nexus3.onap.org:10001/onap/dmaap/kafka111:1.0.4	"/etc/confluent/dock"	14 seconds ago	Up 12 seconds	0.0.0.0:9092->9092/tcp, 9093/tcp	kaf
b9c0c93d4e19 sim-STD	nexus3.o-ran-sc.org:10004/o-ran-sc/a1-simulator:2.0.0	"/bin/sh -c 'src/sta"	16 seconds ago	Up 14 seconds	0.0.0.0:30003->8085/tcp, 0.0.0.0:30004->8185/tcp	a1-
584be418c702 icy-control-panel	nexus3.o-ran-sc.org:10004/o-ran-sc/nonrtric-controlpanel:2.0.0	"java -Xms128m -Xmx2"	16 seconds ago	Up 13 seconds	0.0.0.0:8080->8080/tcp, 0.0.0.0:8082->8082/tcp	pol
435afdb48ad0 icy-agent	nexus3.o-ran-sc.org:10004/o-ran-sc/nonrtric-policy-agent:2.0.1	"java -jar /opt/app/"	16 seconds ago	Up 13 seconds	0.0.0.0:8081->8081/tcp, 0.0.0.0:8433->8433/tcp	pol
1724b4eec638 sim-OSC	nexus3.o-ran-sc.org:10004/o-ran-sc/a1-simulator:2.0.0	"/bin/sh -c 'src/sta"	16 seconds ago	Up 13 seconds	0.0.0.0:30001->8085/tcp, 0.0.0.0:30002->8185/tcp	a1-
dffe1c3e04ac	nexus3.onap.org:10001/onap/dmaap/zookeeper:6.0.3	"/etc/confluent/dock"	16 seconds ago	Up 13 seconds	2888/tcp, 0.0.0.0:2181->2181/tcp, 3888/tcp	Z00
keeper	_					_

you should be able to see containers running as above.

Init Script

You can create sample policy types/policy instances of both Standard & OSC Version.

Run the fill_data.sh script under directory /data to create some dummy data in the running system.

Command to load the data,

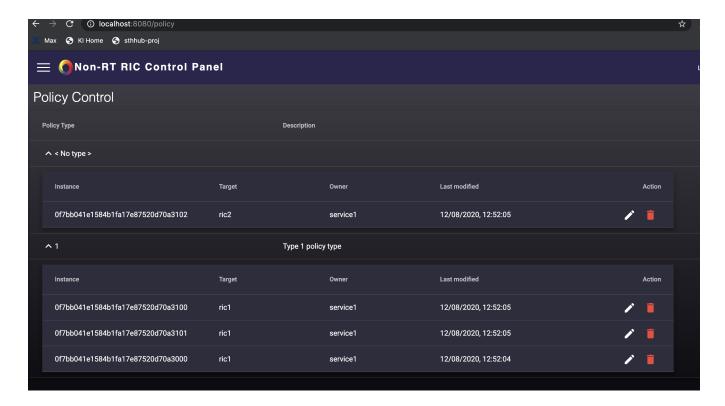
load data cd nonrtric/docker-compose/data ./prepareDmaapMsg.sh [dmaap-mr-port] [al-sim-OSC-port] [al-sim-STD-port] [http/https]

This will create,

- one policy type in a1-sim-OSC
- one service in policy agent
- three policies in a1-sim-OSC
- one policy in a1-sim-STD

Open the GUI to view all the policies & policy types created by the above script.

http://localhost:[control-panel-port]/



Manual test

send messages to dmaap-mr

```
cd nonrtric/docker-compose/data
curl -X POST "http://localhost:3904/events/Al-POLICY-AGENT-READ/" -H "accept: application/json" -H "Content-
Type: application/json" --data-binary @testdata/dmaap-msg-policy-get.json
```

pull messages from dmaap-mr

```
 \begin{tabular}{ll} $\tt Curl -X GET "http://localhost:3904/events/Al-POLICY-AGENT-WRITE/users/policy-agent?timeout=15000\&limit=100" -H "accept: application/json" -H "Content-Type: application/json" | jq . \\ \end{tabular}
```

You can see the messages from dmaap-mr topic A1-POLICY-AGENT-WRITE, with consumer group "user", consumer id "policy-agent"

Message format

create a policy via dmaap-mr

```
{
        "apiVersion": "1.0",
        "operation": "PUT",
        "correlationId": "1596304565904621535",
        "originatorId": "849e6c6b420",
        "payload": {
           "scope": {
               "ueId": "ue3000",
                "qosId": "qos3000"
            "qosObjective": {
                "priorityLevel": 3000
        "requestId": "23343221",
        "target": "policy-agent",
        "timestamp": "2020-08-01 17:56:05.905035",
        "type": "request",
        "url": "/policy?id=0f7bb04le1584b1fa17e87520d70a3000&ric=ric1&service=service1&type=1"
```