Near-RT RIC Deployment

Near-RT RIC deployment steps

Both VMs:

```
mkdir workspace
cd workspace
git clone "https://gerrit.o-ran-sc.org/r/ric-plt/ric-dep"
cd ric-dep/bin
sudo ./install_k8s_and_helm.sh
sudo ./install_common_templates_to_helm.sh
sudo ./setup-ric-common-template
```

RIC Platform:

Under ~/workspace/ric-dep repo:

```
nano helm/appmgr/values.yaml
# replace tiller image with these values
#        tiller:
#          registry: ghcr.io
#          name: helm/tiller
#          tag: v2.12.3

nano helm/infrastructure/values.yaml
# replace tiller image with these values
#        tiller:
#          registry: ghcr.io
#          name: helm/tiller
#          tag: v2.12.3

# edit the ../RECIPE_EXAMPLE/example_recipe_latest_stable.yaml
./install -f ../RECIPE_EXAMPLE/example_recipe_latest_stable.yaml
```

RIC Aux

```
cd ~/workspace
git clone "https://gerrit.o-ran-sc.org/r/it/dep"
cd dep
nano RECIPE_EXAMPLE/AUX/example_recipe.yaml
# replace the ric-dashboard image with the one below
#  image:
#    registry: nexus3.o-ran-sc.org:10002/o-ran-sc
#    name: ric-dashboard
#    tag: 2.1.0
nano ric-aux/helm/infrastructure/subcharts/kong/values.yaml
# replace the ingressController image with the one below
#ingressController:
#  enabled: true
#  image:
#    repository: docker.io/kong/kubernetes-ingress-controller
#    tag: 0.7.0

sudo apt install dos2unix
cd dep/bin
dos2unix -o deploy-ric-aux
kubectl label --overwrite nodes $(hostname) portal-storage=enable
kubectl label --overwrite nodes $(hostname) aaf-storage=enable
./deploy-ric-aux ../RECIPE_EXAMPLE/AUX/example_recipe.yaml
```

E2 Simulator deployment steps

```
git clone "https://gerrit.o-ran-sc.org/r/sim/e2-interface"
sudo apt-get install -y cmake build-essential libsctp-dev autoconf
automake libtool bison flex libboost-all-dev
cd e2-interface/e2sim
mkdir build
cd build
cmake .. && make package && cmake .. -DDEV_PKG=1 && make package
cp *.deb ../e2sm_examples/kpm_e2sm/
```

```
##############################################################################
#####
# Copyright 2022 highstreet technologies GmbH
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
version: '3.8'

services:
  e2-simulator:
    image: "o-ran-sc/ntsim-e2-simulator:2.0.0"
    build:
      context: /home/ubuntu/workspace/e2-interface/e2sim/e2sm_examples
/kpm_e2sm
    container_name: e2-simulator
    entrypoint: ["kpm_sim", "10.106.23.83", "38000"]
```

```
#!/bin/bash

NODE_IP=$(kubectl get pod -n=ricplt -l app=ricplt-e2term-alpha -o
jsonpath="{.items[0].status.hostIP}")
NODE_PORT=$(kubectl get svc -n=ricplt service-ricplt-e2term-sctp-alpha -o
jsonpath="{.spec.ports[0].nodePort}")

echo "E2 term IP address: ${NODE_IP}"
echo "E2 term port: ${NODE_PORT}"
```

```
cd /workspace
mkdir e2-sim
cd e2-sim
nano e2sim-infra.sh
# copy above script for getting the IP and Port of the E2 termination point
# run the script and use the IP and port in the following docker-compose

nano docker-compose.yml
# copy above YAML into this file
# replace the entrypoint IP and port with the above IP and port

sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --
dearmor -o /etc/apt/keyrings/docker.gpg
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings
/docker.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.
list > /dev/null
sudo apt-get update
sudo apt-get install docker-compose-plugin
sudo docker compose build
sudo docker compose up -d
```

Wireshark deployment

Wireshark can also be deployed to sniff the traffic inside the k8s cluster.

Steps:

krew installation (instructions here):

```
(
  set -x; cd "$(mktemp -d)" &&
  OS="$(uname | tr '[:upper:]' '[:lower:]')" &&
  ARCH="$(uname -m | sed -e 's/x86_64/amd64/' -e 's/\(arm\)\(64\)\?.*/\1\2
/' -e 's/aarch64$/arm64/')" &&
  KREW="krew-${OS}_${ARCH}" &&
  curl -fsSLO "https://github.com/kubernetes-sigs/krew/releases/latest
/download/${KREW}.tar.gz" &&
  tar zxvf "${KREW}.tar.gz" &&
  ./"${KREW}" install krew
)

export PATH="${KREW_ROOT:-$HOME/.krew}/bin:$PATH"
```

ksniff installation (instructions here):

```
kubectl krew install sniff
```

Wireshark can be deployed inside a docker container, as described in the OAM repository (wireshark section) in O-RAN-SC.

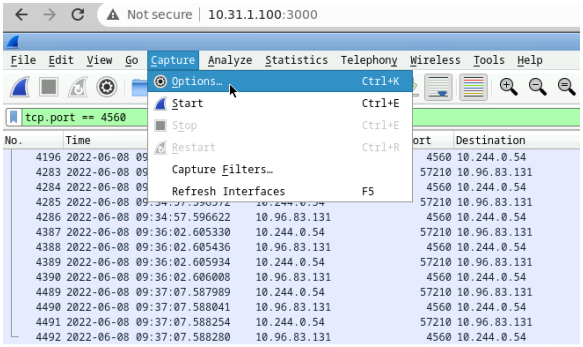
In order to see the packets captured with `kubectl sniff` in Wireshark, a pipe needs to be created (instructions here):

```
mkfifo /tmp/capture.pcap
```

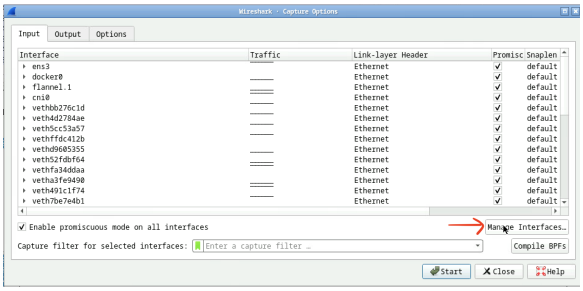
Please make sure that the above pipe (/tmp/capture.pcap) is mounted as a volume in the Wireshark docker container!

Configure Wireshark to capture packets from that pipe:

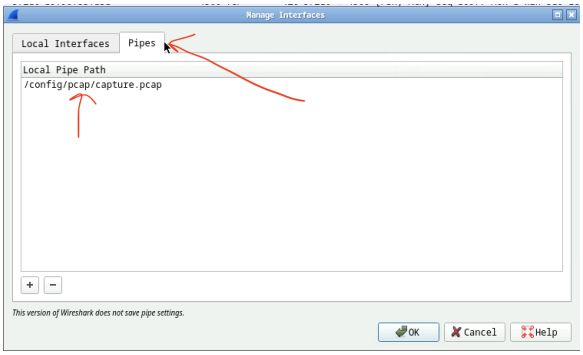
Capture Options



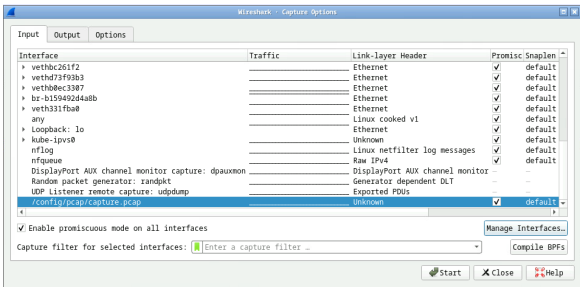
Manage Interfaces:



Add a new pipe pointing to the pipe inside the mapped volume of the Wireshark docker container.



Select that pipe as the interface to run the capture on:



Start the k8s sniffer on the Pod of interest:

```
kubectl sniff ricxapp-rc-86f7dfd8ff-5klz8 -n ricxapp -o /tmp/capture.pcap
# this will sniff traffic from the RC xApp k8s pod to the /tmp/capture.
pcap pipe
```

Important: the order matters! The Wireshark capture needs to be started first, and only then the sniffing of the traffic towards that pipe!