Debugging RO (and others)
Eduardo Sousa (Canonical)
Summary

The main idea of this session is to be able to put OSM code under a debugger.

General approach is:
1. Get your code
2. Stop the component to debug
3. Expose all necessary services
4. Install the dependencies and code
5. Run the code with debugger

Note: this session assumes OSM is already installed in the machine.
Install python3.8

Please install python3.8:
https://linuxize.com/post/how-to-install-python-3-8-on-ubuntu-18-04/
Get your code

1. Get the code
   ○ git clone <repository-url>

2. Setup the git repository information
   ○ git config --local user.name <username>
   ○ git config --local user.email <email>
   ○ git config --local pull.rebase true

3. Install gerrit hooks
   ○ curl -Lo .git/hooks/commit-msg http://osm.etsi.org/gerrit/tools/hooks/commit-msg
   ○ chmod u+x .git/hooks/commit-msg

4. Setup your .gitignore
   ○ cp .gitignore-common .gitignore
   ○ Verify if it includes all the extra files generated by your IDE/development environment

5. Local repository is now setup
Stop RO

To stop RO, run the following command:

```
kubectl -n osm scale deployment ro --replicas=0
```
Expose services

Fetch this Kubernetes spec file ([link](#)) and comment out the part of RO and MongoDB. Run the following commands to apply it:

```
kubectl -n osm apply -f <filename>
```

Check if debug services are created:

```
kubectl -n osm get service | grep debug
```

```
ubuntu@ro-dev:~$ kubectl -n osm get service | grep debug
kafka-debug   NodePort 10.98.219.43 <none> 9092:9092/TCP 71s
keystone-debug NodePort 10.110.0.55 <none> 5000:5000/TCP 71s
mysql-debug   NodePort 10.108.63.191 <none> 3306:3306/TCP 71s
zookeeper-debug NodePort 10.105.246.21 <none> 2181:2181/TCP 71s
```
Getting the IP address from MongoDB:

```
kubectl -n osm get service | grep mongodb-k8s
```

<table>
<thead>
<tr>
<th>Service</th>
<th>ClusterIP</th>
<th>Port</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>mongodb-k8s</td>
<td>10.108.24.82</td>
<td>27017/TCP</td>
<td>15h</td>
</tr>
<tr>
<td>mongodb-k8s-endpoints</td>
<td>None</td>
<td>&lt;none&gt;</td>
<td>15h</td>
</tr>
<tr>
<td>mongodb-k8s-operator</td>
<td>10.102.125.14</td>
<td>30666/TCP</td>
<td>15h</td>
</tr>
</tbody>
</table>
Edit /etc/hosts

Edit /etc/hosts to contain the following information:

```
127.0.0.1 localhost keystone nbi mongo mysql ro
10.98.219.43 kafka kafka-0.kafka.osm.svc.cluster.local
10.108.24.82 mongo mongodb mongodb-k8s
```

Note: don’t forget to adapt to the values obtained in the previous slides.
Test connectivity with kafka

To test connectivity with kafka, run the following command:

```
nc -zvw1 kafka 9092
```

You should have this result:

```
ubuntu@ro-dev:$ nc -zvw1 kafka 9092
Connection to kafka 9092 port [tcp/*] succeeded!
```
Test connectivity with mongodb

To test connectivity with mongodb, run the following command:

nc -zvw1 mongodb 27017

You should have this result:
To change LCM deployment, run the following command:

`kubectl -n osm edit deployment lcm`

Edit the following to the IP of the machine:
Get the pip standardization change

NOTE: Only do this, if the change hasn’t been merged yet.

Run the following command to pull the change:

```
git pull "https://osm.etsi.org/gerrit/osm/RO" refs/changes/55/10355/11
```
Let's create a Virtual Env

You should always create a virtual environment to isolate dependencies (inside RO):

```
python3.8 -m venv venv
```

And activate it:

```
source venv/bin/activate
```
Install all dependencies and projects

Let's create a file called `install.sh` inside RO folder, with the following content:

```bash
#!/bin/bash

pip install --upgrade pip
pip install -r requirements.txt
pip install -r requirements-dev.txt
pip install -e RO-plugin
pip install -e NG-RO
pip install -e RO-VIM-vmware
pip install -e RO-VIM-openstack
pip install -e RO-VIM-openvim
pip install -e RO-VIM-aws
pip install -e RO-VIM-azure
pip install -e RO-VIM-fos
pip install -e RO-SDN-dynpac
pip install -e RO-SDN-ietf2vpn
pip install -e RO-SDN-onos_vpls
pip install -e RO-SDN-onos_openflow
pip install -e RO-SDN-odl_openflow
pip install -e RO-SDN-floodlight_openflow
pip install -e RO-SDN-arista_cloudvision
pip install -e RO-SDN-juniper CONTRAIL
```
Install all dependencies and projects (cont.)

Make it executable and run it:

    chmod +x install.sh
    ./install.sh

Note: this is optional but strongly recommended for RO due to the amount of plugins to install.
Verify that everything is working

Inside the RO folder and with the venv activated, run the following:

```bash
python -u -m osm_ng_ro.ro_main
```

You should get something similar to this:
Putting RO in a debugger (VS Code)

1) Open VS Code and create a SSH target to the machine and RO folder.
2) Install the python extension in the SSH target.
3) Open the debug tab and click “create a launch.json file”.
4) Select Python.
5) Select Module and enter “osm_ng_ro.ro_main”.
6) Insert some breakpoints
7) Start debugger.
Result (VS Code)