

# Open Source MANO

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

Getting the IP address from MongoDB:

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

```
ubuntu@ro-dev:~$ kubectl -n osm get service | grep mongodb-k8s
mongodb-k8s          ClusterIP   10.108.24.82    <none>      27017/TCP    15h
mongodb-k8s-endpoints ClusterIP   None           <none>      <none>       15h
mongodb-k8s-operator ClusterIP   10.102.125.14  <none>      30666/TCP    15h
```

# 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:~/RO$ 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:

```
ubuntu@ro-dev:~/RO$ nc -zvw1 mongodb 27017  
Connection to mongodb 27017 port [tcp/*] succeeded!
```

# Change LCM deployment

To change LCM deployment, run the following command:

```
kubectl -n osm edit deployment lcm
```

Edit the following to the IP of the machine:

```
spec:
  progressDeadlineSeconds: 600
  replicas: 0
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: lcm
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: lcm
    spec:
      containers:
      - env:
        - name: OSMLCM_RO_HOST
          value: 172.21.248.176
        - name: OSMLCM_DATABASE_URI
          value: mongodb://mongodb-k8s:27017/?replicaSet=rs0
```

```
dnsPolicy: ClusterFirst
initContainers:
- command:
  - sh
  - -c
  - until (nc -zvw1 kafka 9092 && nc -zvw1 172.21.248.176 9090 && nc -zvw1 mongodb-k8s 27017
    ); do sleep 3; done; exit 0
  image: alpine:latest
  imagePullPolicy: Always
  name: kafka-ro-mongo-test
```

# 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
```

# Lets 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

Lets create a file called install.sh inside RO folder, with the following content:

```
#!/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-ietf12vpn
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:

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

You should get something similar to this:

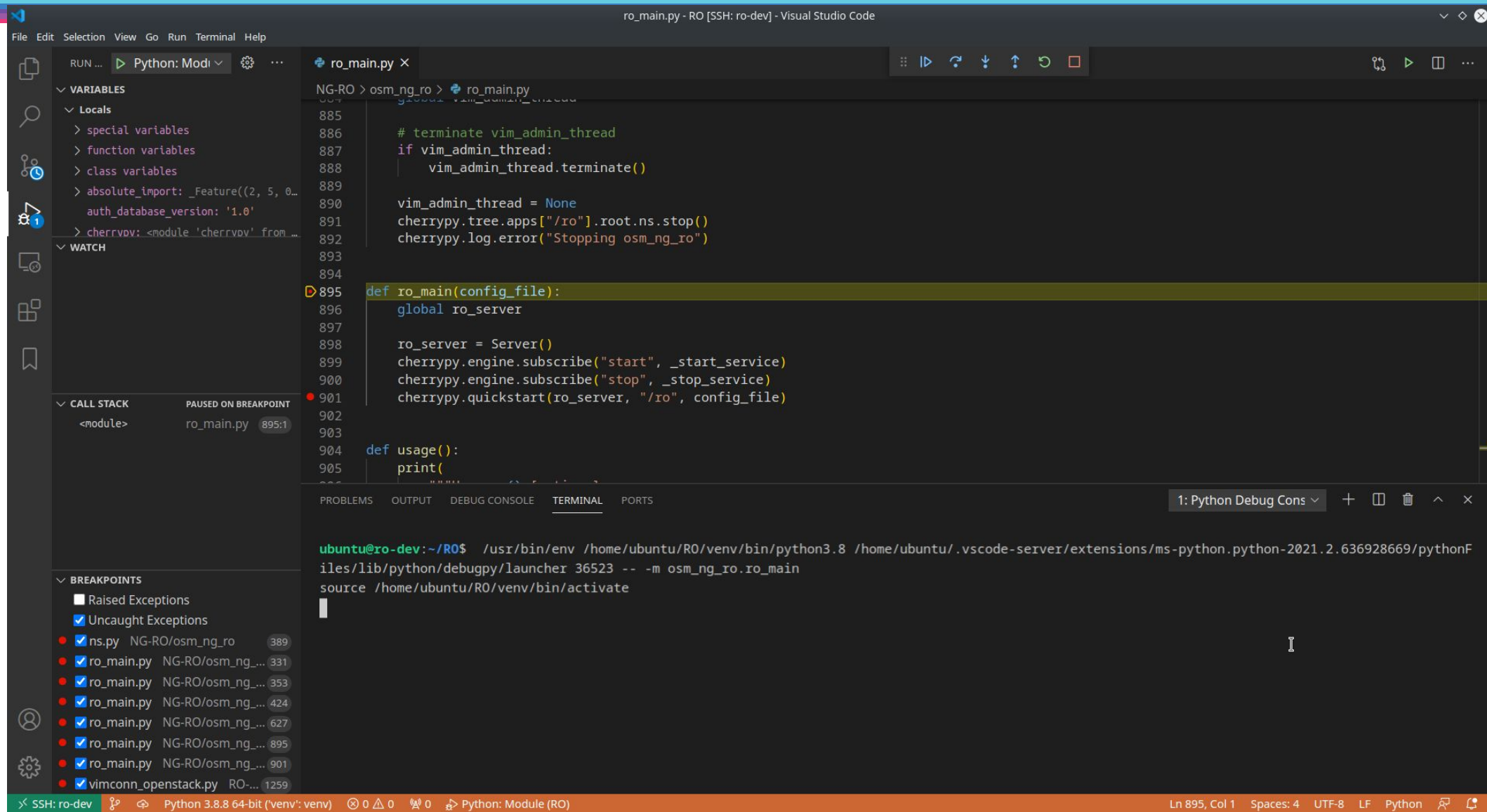
```
(venv) ubuntu@ro-dev:~/RO$ python -u -m osm_ng_ro.ro_main
CherryPy Checker:
'/app/RO/RO-NG/osm_ng_ro/html_public' (root + dir) is not an existing filesystem path.
section: [/static]
root: None
dir: '/app/RO/RO-NG/osm_ng_ro/html_public'
I
2021-03-12T12:18:50 INFO ro.db dbmongo.py:127 Connected to database osm version 1.2
2021-03-12T12:18:50 INFO ro.server _cplogging.py:213 [12/Mar/2021:12:18:50] ENGINE Started monitor thread 'Autoreloader'.
2021-03-12T12:18:50 INFO ro.db dbmongo.py:127 Connected to database osm version 1.2
2021-03-12T12:18:50 INFO ro.vimadmin vim_admin.py:393 Starting
2021-03-12T12:18:50 DEBUG ro.vimadmin vim_admin.py:284 Starting vim_account subscription task
2021-03-12T12:18:50 INFO ro.server _cplogging.py:213 [12/Mar/2021:12:18:50] ENGINE Serving on http://0.0.0.0:9090
2021-03-12T12:18:50 INFO ro.server _cplogging.py:213 [12/Mar/2021:12:18:50] ENGINE Bus STARTED
2021-03-12T12:20:13 INFO ro.access _cplogging.py:283 10.244.0.31 - - [12/Mar/2021:12:20:13] "GET /ro/version HTTP/1.1" 200 54 "" "Python/3.6 aiohttp/3.0.1"
```



# 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)



The screenshot displays the Visual Studio Code interface with a Python script named `ro_main.py` open. The code is being debugged, and a breakpoint is set at line 895. The terminal shows the command to run the script.

```
ro_main.py - RO [SSH: ro-dev] - Visual Studio Code
```

File Edit Selection View Go Run Terminal Help

RUN ... Python: Mod... ro\_main.py

VARIABLES

- Locals
  - special variables
  - function variables
  - class variables
  - absolute\_import: \_Feature((2, 5, 0...
  - auth\_database\_version: '1.0'
  - cherrypy: <module 'cherrypy' from ...
- WATCH

CALL STACK PAUSED ON BREAKPOINT

- <module> ro\_main.py 895:1

BREAKPOINTS

- Raised Exceptions
- Uncaught Exceptions
- ns.py NG-RO/osm\_ng\_ro 389
- ro\_main.py NG-RO/osm\_ng\_... 331
- ro\_main.py NG-RO/osm\_ng\_... 353
- ro\_main.py NG-RO/osm\_ng\_... 424
- ro\_main.py NG-RO/osm\_ng\_... 627
- ro\_main.py NG-RO/osm\_ng\_... 895
- ro\_main.py NG-RO/osm\_ng\_... 901
- vimconn\_openstack.py RO-... 1259

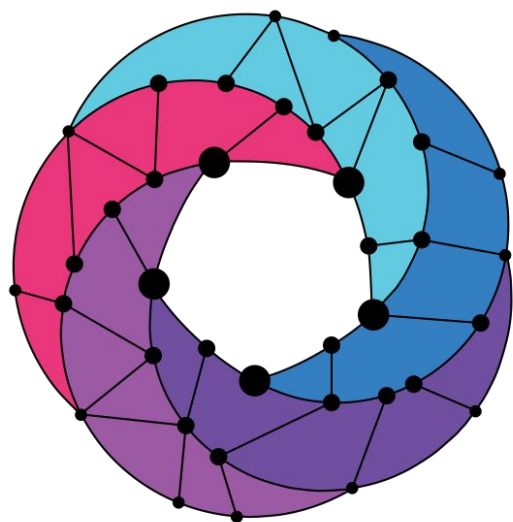
```
NG-RO > osm_ng_ro > ro_main.py
885
886 # terminate vim_admin_thread
887 if vim_admin_thread:
888     vim_admin_thread.terminate()
889
890 vim_admin_thread = None
891 cherryypy.tree.apps["/ro"].root.ns.stop()
892 cherryypy.log.error("Stopping osm_ng_ro")
893
894
895 def ro_main(config_file):
896     global ro_server
897
898     ro_server = Server()
899     cherryypy.engine.subscribe("start", _start_service)
900     cherryypy.engine.subscribe("stop", _stop_service)
901     cherryypy.quickstart(ro_server, "/ro", config_file)
902
903
904 def usage():
905     print(
906
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

1: Python Debug Cons

```
ubuntu@ro-dev:~/RO$ /usr/bin/env /home/ubuntu/RO/venv/bin/python3.8 /home/ubuntu/.vscode-server/extensions/ms-python.python-2021.2.636928669/pythonF
iles/lib/python/debugpy/launcher 36523 -- -m osm_ng_ro.ro_main
source /home/ubuntu/RO/venv/bin/activate
```

Ln 895, Col 1 Spaces: 4 UTF-8 LF Python



# Open Source MANO

Find us at:

[osm.etsi.org](https://osm.etsi.org)  
[osm.etsi.org/wikipub](https://osm.etsi.org/wikipub)