EECE 7290 – Software Defined Networking (Spring 2017), University of Massachusetts, Lowell

Project - SDN for Secure Video Streaming: CORD Based Secure Video Streaming

Document – Running HelloWorld service and creating new tenant service.

Students –

1) Aman Maldar
2) Priyanka Murthy

Date of submission – May 5, 2017

Part 1 - Running the HelloWorld Service on CORD [1]

This section gives the steps to run the example service in the CORD environment. The second half of the document provides the steps to create a new tenant service.

Assuming, the steps mentioned in document 'CORD Environment Setup' [2] are completed, we are ready to run HelloWorld service and make changes into the template. Following steps will run the example service in the production environment. Production environment already contains XOS, ONOS, OpenStack installed on it.

Steps:

1) ssh into the compute node created on the CloudLab.
   - ssh username@ip_address

     ex- ssh aman_uml@128.104.222.127

2) ssh into prod environment
   - ssh prod
3) The prod environment contains the test client, which can be used to run the services inside the CORD.
   - sudo lxc exec testclient -- /bin/bash
4) Ping to see if all the services are up and running. Ping should be successful
   - ping 8.8.8.8

```
vagrant@prod:~$ sudo lxc exec testclient -- /bin/bash
root@testclient:~# ping 8.8.8.8 -c3
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=44 time=24.6 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=44 time=22.6 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=44 time=22.7 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 22.643/23.363/24.664/0.921 ms
root@testclient:~#
```

5) Exit the testclient

- exit

6) Provide the username to access the services.

- source ~/admin-openrc.sh

7) See the list of all the running services.

- nova list --all-tenants

```
root@testclient:~# exit
exit
vagrant@prod:~$ source ~/admin-openrc.sh
vagrant@prod:~$ nova list --all-tenants
/usr/local/lib/python2.7/dist-packages/requests/packages/urllib3/util/ssl_.py:33
4: SNIMissingWarning: An HTTPS request has been made, but the SNI (Subject Name
Indication) extension to TLS is not available on this platform. This may cause t
he server to present an incorrect TLS certificate, which can cause validation fa
ilures. You can upgrade to a newer version of Python to solve this. For more inf
ormation, see https://urllib3.readthedocs.io/en/latest/advanced-usage.html#ssl-w
arnings
  SNIMissingWarning
```

See the public IP address as shown below.

```
  SubjectAltNameWarning
+--------------------------------------+-------------------------+--------+------------+-------------+------------
--------------------------------------+
| ID                                   | Name                    | Status | Task State | Power State | Networks
                                       |
+--------------------------------------+-------------------------+--------+------------+-------------+------------
--------------------------------------+
| 373da1c8-d6eb-48b0-baaa-600b67d9ef13 | mysite_exampleservice-2 | ACTIVE | -          | Running     | management=
172.27.0.3; public=10.6.1.194        |
| 3810f7f1-4985-4631-84fa-b74ae2b40f6d | mysite_vsg-1            | ACTIVE | -          | Running     | management=
172.27.0.2; mysite_vsg-access=10.0.2.2 |
+--------------------------------------+-------------------------+--------+------------+-------------+------------
--------------------------------------+
vagrant@prod:~$
```

8) Enter the testclient again.

- sudo lxc exec testclient -- /bin/bash

9) Access the service by using curl command. You will see the results printed.

- curl http://10.6.1.194

```
vagrant@prod:~$ sudo lxc exec testclient -- /bin/bash
root@testclient:~# curl http://10.6.1.194
ExampleService
 Service Message: "hello"
 Tenant Message: "world"

root@testclient:~#
```

This is it! We can see the example service printing the message HelloWorld.

Part 2 - Making changes to the HelloWorld Service.

Follow the steps 1,2 from above.

Navigate to the folder /service-profile/cord-pod and see the files in the folder

```
vagrant@prod:~$ cd service-profile/cord-pod
vagrant@prod:~/service-profile/cord-pod$ ls
admin-openrc.sh                  id_rsa                          nodes.yaml
apt-prereqs                      id_rsa.pub                      onboarding-docker-compose
cdn                              images                          onos_monitoring_service_endpoints.json
cleanup.sh                       key_import                      openstack.yaml
cord-services.yaml               Makefile                        pod-cdn.yaml
cord-test-subscriber.yaml        make-inframonitoring-yaml.sh    public-net.yaml
deployment.yaml                  make-virtualbng-json.sh         README.md
docker-compose-bootstrap.yml     management-net.yaml             synchronizers.yaml
exampleservicemonitoring.yaml    monitoringservice.yaml          vrouter.yaml
exampleservice-synchronizer.yaml monitoring_synchronizer.yaml    vsgmonitoring.yaml
exampleservice.yaml              monitoringtenant.yaml           vtn.yaml
fabric.yaml                      network-cfg-quickstart.json     xos_cord_config
files                            node_key                        xos.yaml
vagrant@prod:~/service-profile/cord-pod$
```

We have to make changes in the file exampleservice.yaml

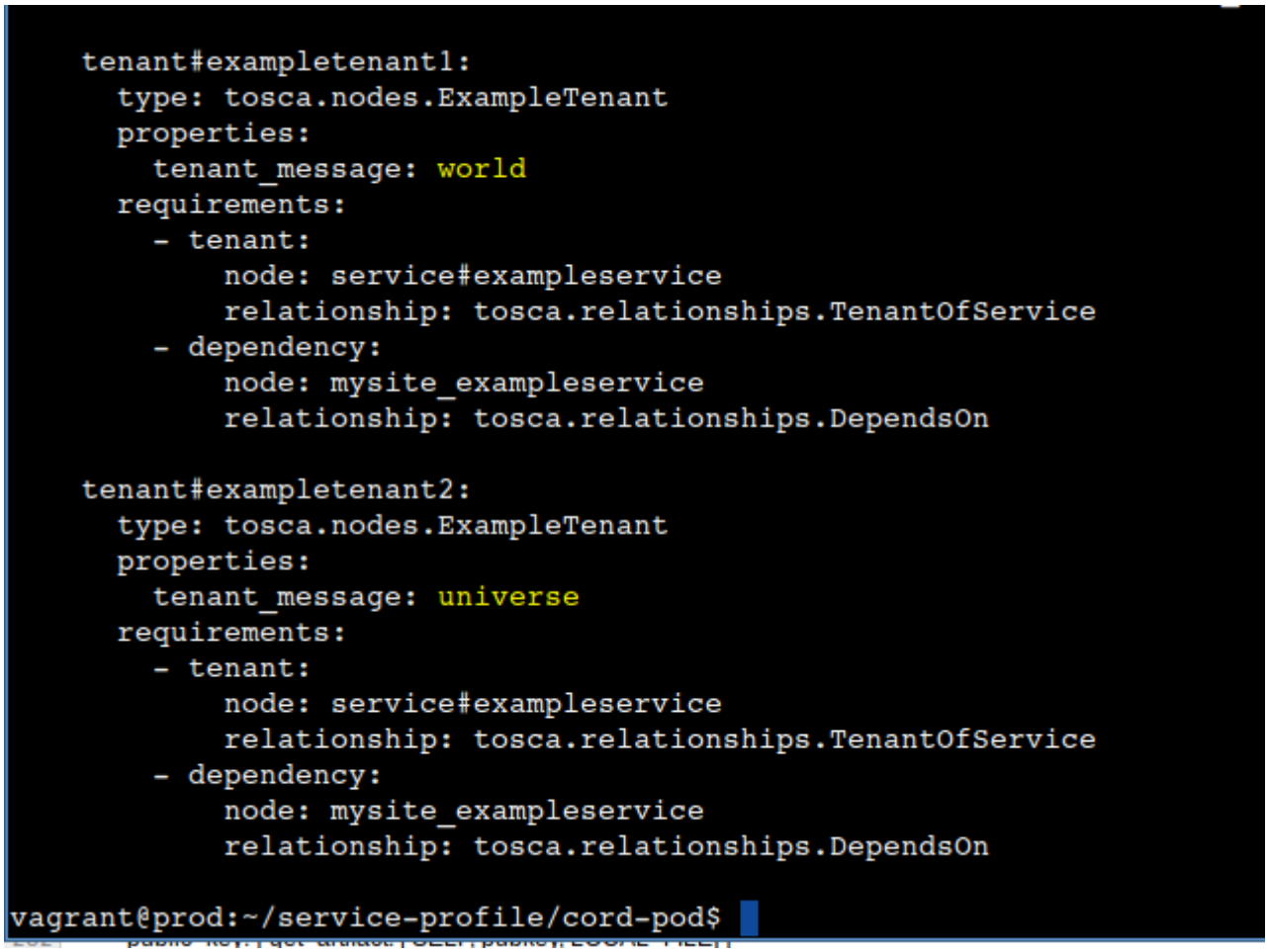
Open the file using editor and see the changes made at the end of file

```
service#exampleservice:
  type: tosca.nodes.ExampleService
  requirements:
    - management:
        node: management
        relationship: tosca.relationships.UsesNetwork
  properties:
    view_url: /admin/exampleservice/exampleservice/$id$/
    kind: exampleservice
    public_key: { get_artifact: [ SELF, pubkey, LOCAL_FILE] }
    private_key_fn: /opt/xos/services/exampleservice/keys/exampleservice_rsa
    service_message: Hello
  artifacts:
    pubkey: /opt/xos/services/exampleservice/keys/exampleservice_rsa.pub

tenant#exampletenant1:
  type: tosca.nodes.ExampleTenant
  properties:
    tenant_message: world
  requirements:
    - tenant:
        node: service#exampleservice
        relationship: tosca.relationships.TenantOfService
    - dependency:
        node: mysite_exampleservice
        relationship: tosca.relationships.DependsOn
```

```
tenant#exampletenant2:
  type: tosca.nodes.ExampleTenant
  properties:
   tenant_message: universe
  requirements:
   - tenant:
      node: service#exampleservice
      relationship: tosca.relationships.TenantOfService
   - dependency:
      node: mysite_exampleservice
      relationship: tosca.relationships.DependsOn
```

Observe the changes made



Once the above changes are made we have to run the development loop again. This takes approximately 20 minutes.

- make cleanup; make local_containers; make; make vtn; make fabric; make cord; make cord-subscriber; make exampleservice

```
vagrant@prod:~/service-profile/cord-pod$  make cleanup; make local_containers; make; make vtn; make fabric; make
ord; make cord-subscriber; make exampleservice
test ! -s /home/vagrant/service-profile/cord-pod//onboarding-docker-compose/docker-compose.yml || sudo docker-co
ose -p cordpod -f /home/vagrant/service-profile/cord-pod//onboarding-docker-compose/docker-compose.yml stop
Stopping cordpod_xos_ui_1 ...
Stopping cordpod_xos_synchronizer_exampleservice_1 ...
Stopping cordpod_xos_synchronizer_vtn_1 ...
Stopping cordpod_xos_synchronizer_onos_1 ...
Stopping cordpod_xos_synchronizer_vrouter_1 ...
Stopping cordpod_xos_synchronizer_fabric_1 ...
Stopping cordpod_xos_synchronizer_vsg_1 ...
Stopping cordpod_xos_synchronizer_vtr_1 ...
Stopping cordpod_xos_synchronizer_openstack_1 ...
```

We are creating new tenant service. Two tenant s are prints 2 different messages. Parent service always runs first to print “Hello”. Two tenant services prints “world” and “universe” respectively.

See the result of running the development loop.

```
bash /home/vagrant/service-profile/common/wait_for_onboarding_ready.sh 81 services/exampleservice
Waiting for services/exampleservice to be onboarded
.......services/exampleservice is onboarded
bash /home/vagrant/service-profile/common/wait_for_onboarding_ready.sh 81 xos
Waiting for xos to be onboarded
..............xos is onboarded
bash /home/vagrant/service-profile/common/wait_for_xos_port.sh 8888
Waiting for XOS to start listening on port 8888
.............................XOS is ready
python /home/vagrant/service-profile/common/run_tosca.py 8888 padmin@vicci.org letmein exampleservice.yaml
ordered_names: ['management', 'm1.small', 'trusty-server-multi-nic', 'service#exampleservice', 'public', '
 'mysite_exampleservice', 'tenant#exampletenant1', 'Private', 'service#vrouter']
Network:management (management) already exists. Skipping update due to 'no-update' property
Flavor:m1.small (m1.small) already exists
Image:trusty-server-multi-nic (trusty-server-multi-nic) already exists
Created ExampleService 'exampleservice'
Network:public (public) already exists. Skipping update due to 'no-update' property
Site:mysite (mysite) already exists
Created Slice 'mysite_exampleservice'
Added network connection from 'mysite_exampleservice' to 'management'
Added network connection from 'mysite_exampleservice' to 'public'
Created ExampleTenant 'exampleservice-tenant-11'
NetworkTemplate:Private (Private) already exists
Service:vrouter (service#vrouter) already exists. Skipping update due to 'no-update' property

sleep 60
vagrant@prod:~/service-profile/cord-pod$
```

Once the development loop is completed, again see the list of all the services. We can see newly created service

```
+--------------------------------------+
| ID                                   | Name                    | Status | Task State | Power State | Networks
|
+--------------------------------------+-------------------------+--------+------------+-------------+------------
--------------------------------------+
| bef6fb3d-806e-493a-8966-66b84b2dfbfa | mysite_exampleservice-2 | ACTIVE | -          | Running     | management=
172.27.0.4; public=10.6.1.195        |
| c178bcdc-9936-4682-84b7-c054a38a0f5c | mysite_exampleservice-3 | ACTIVE | -          | Running     | management=
172.27.0.3; public=10.6.1.194        |
| 82673488-d8fa-4083-bc67-03351a4720d0 | mysite_vsg-1            | ACTIVE | -          | Running     | management=
172.27.0.2; mysite_vsg-access=10.0.2.2 |
+--------------------------------------+-------------------------+--------+------------+-------------+------------
```

Do the curl again to see the results.

```
--------------------------------------+
vagrant@prod:~/service-profile/cord-pod$ sudo lxc exec testclient -- /bin/bash
root@testclient:~# curl http://10.6.1.194
ExampleService
 Service Message: "hello"
 Tenant Message: "universe"

root@testclient:~# curl http://10.6.1.195
ExampleService
 Service Message: "hello"
 Tenant Message: "world"

root@testclient:~#
```

This is it; we have created new tenant service.

References-

[1] Running hello world - example service

https://github.com/opencord/exampleservice/tree/master/xos

[2] Project Documentation

https://github.com/amanmaldar/EECE7290_Project

[3] Github repository link for all project details- https://github.com/priyanka-N-Murthy/EECE-7290-Software-Defined-Networking-Project