Openstack Services Heat, Magnum, Octavia

«OpenStack Administration 101» , 30 Nov. – 3 Dec. 2021

Alessandro Costantini, Doina Cristina Duma – INFN CNAF

01/12/2021

Overview

- **Heat**
  - Orchestration
- **Magnum**
  - Container orchestration
- **Octavia**
  - Load Balancing

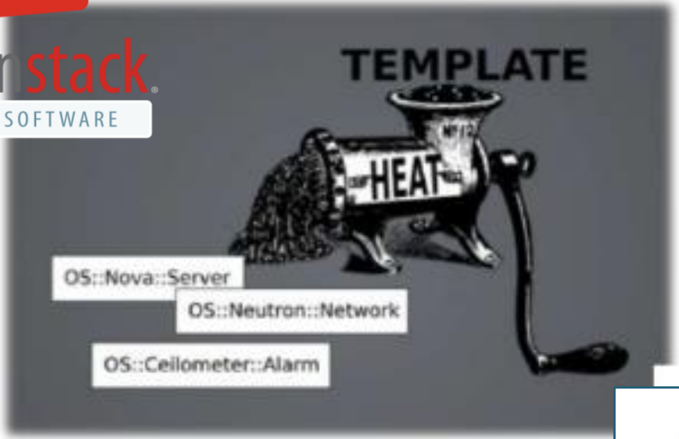
Openstack Heat

- **Heat**
    - Orchestration
    - Heat Architecture
    - Heat Orchestration Template
    - Installation&Configuration
    - References

Openstack Heat

- **Orchestration**
  - Procedure using tools and services aimed at orchestrating resources that are automatically configured and deployed
  - The collection of resources associated with a template is known as a stack
  - Resources are represented in a declarative form (Template)
  - Examples of orchestration
    - AWS CloudFormation
    - Heat Orchestration Template in Cloud
    - TOSCA

Heat vs TOSCA

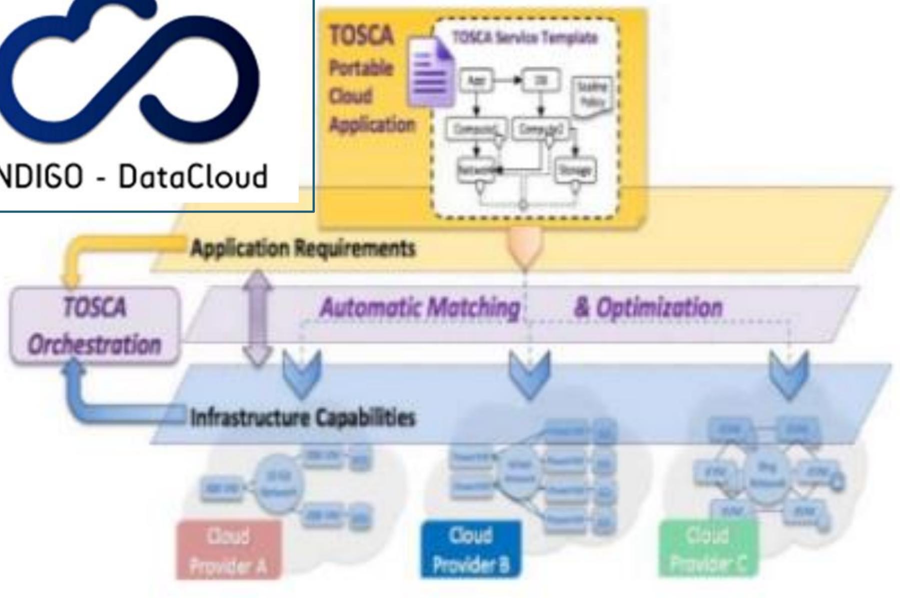


Heat provides a mechanism for orchestrating OpenStack resources through the use of modular templates.

Heat-Translator
(IaaS, App Orchestration)
Tacker
(Network Function Orchestration)
https://wiki.openstack.org/

INDIGO - DataCloud

Topology and Orchestration Specification for Cloud Applications (**TOSCA**), is an OASIS standard language to describe a topology of cloud based web services, their components, relationships, and the processes that manage them



TOSCA defines the interoperable description of applications; including their components, relationships, dependencies, requirements, and capabilities….

Openstack Heat

- **Heat**
  - Heat is the main project in the OpenStack Orchestration program. It implements an orchestration engine to launch multiple composite cloud applications based on **templates** in the form of text files that can be treated like code.
    - Uses template HOT and AWS CloudFormation
    - Defines **Parameters** and **Environments**
    - Provides **Software** configuration
    - Provides **Resources** reservation
    - Provides **Events notification**
    - Provides auto-scaling

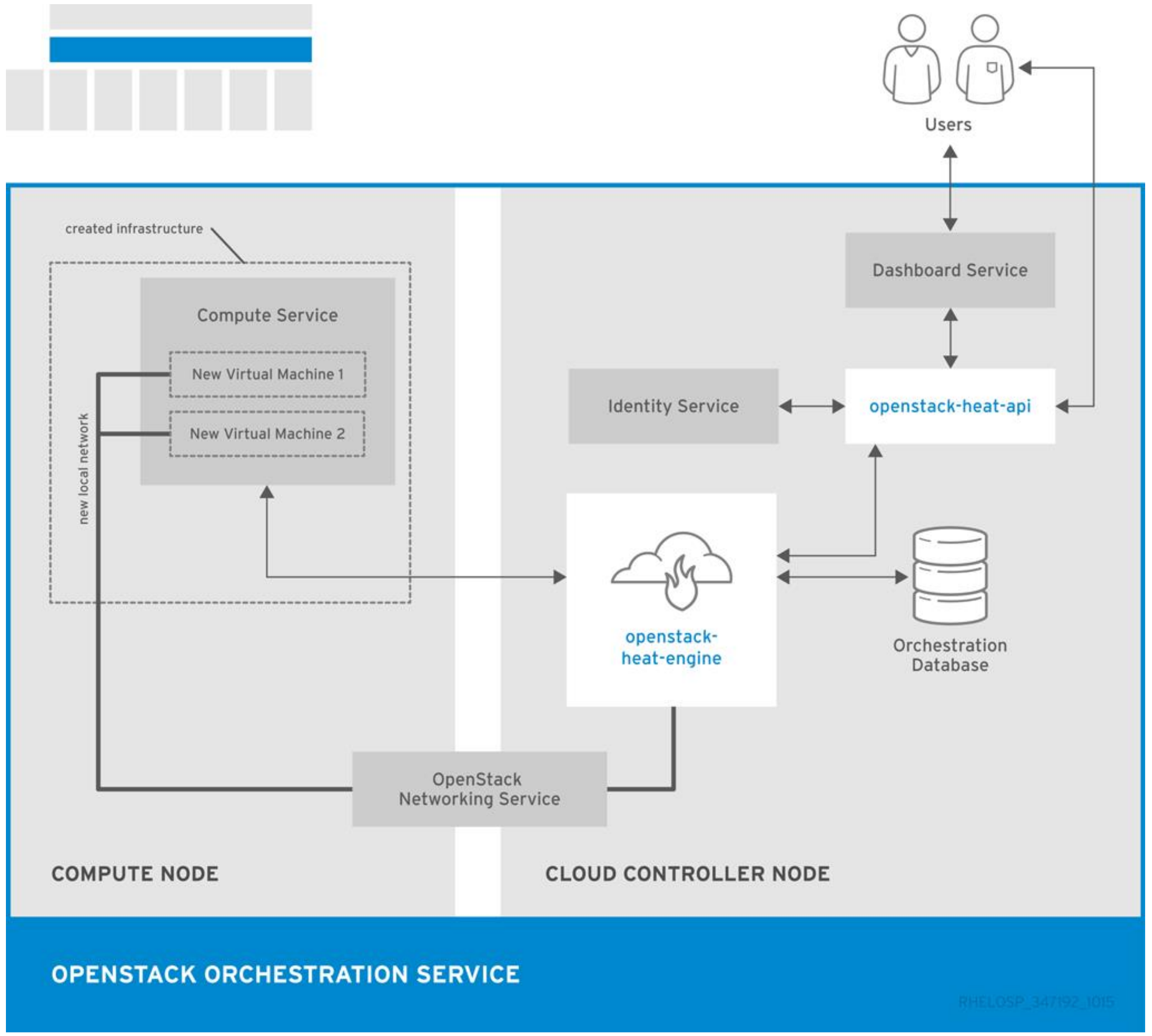
Openstack Heat

- **Heat components**

  - heat
    - The heat tool is a CLI which communicates with the heat-api . End developers could also use the heat REST API directly.
  - heat-api
    - The heat-api component provides an OpenStack-native ReST API that processes API requests by sending them to the heat-engine over Remote Procedure Call (RPC).
  - heat-api-cfn
    - The heat-api-cfn component provides an AWS-style Query API that is compatible with AWS CloudFormation and processes API requests by sending them to the heat-engine over RPC.
  - heat-engine
    - The heat engine does the main work of orchestrating the launch of templates and providing events back to the API consumer.

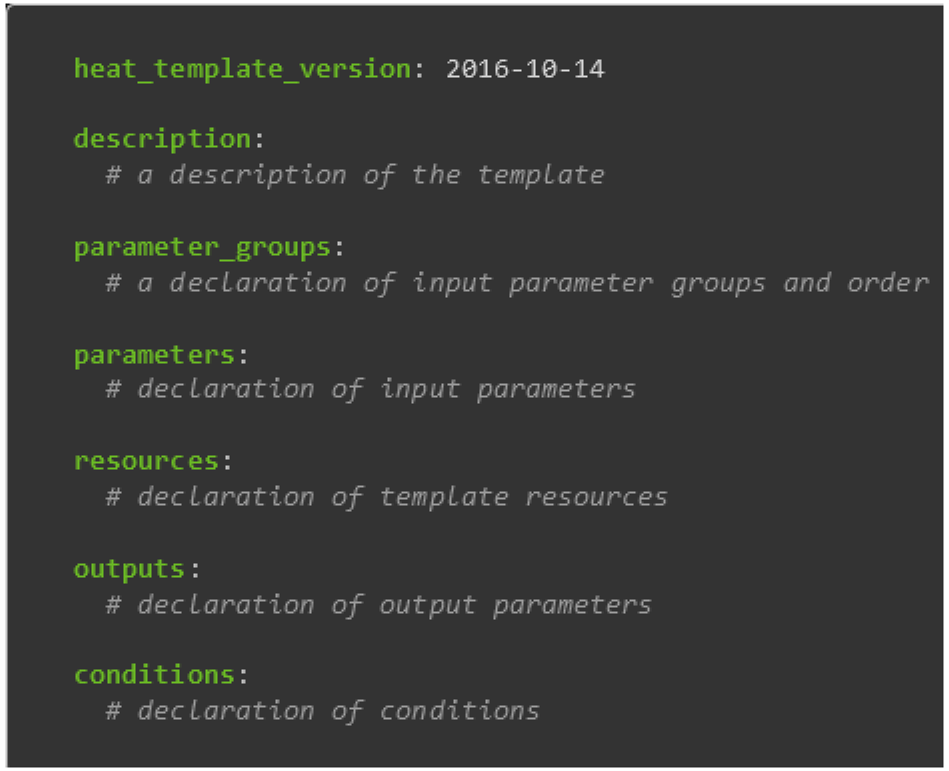
Openstack Heat

- **Heat Architecture**



Openstack Heat

- **Heat Orchestration template**
  - A Heat template describes the infrastructure for a cloud application in text files which are readable and writable by humans, and can be managed by version control tools.
  - HOT templates are defined in YAML and follow the present structure.



```
heat_template_version: 2016-10-14

description:
  # a description of the template

parameter_groups:
  # a declaration of input parameter groups and order

parameters:
  # declaration of input parameters

resources:
  # declaration of template resources

outputs:
  # declaration of output parameters

conditions:
  # declaration of conditions
```

Openstack Heat

•Heat Orchestration template

heat_template_version
- This key with value `2013-05-23` (or a later date) indicates that the YAML document is a HOT template of the specified version.

description
- This optional key allows for giving a description of the template, or the workload that can be deployed using the template.

parameter_groups
- This section allows for specifying how the input parameters should be grouped and the order to provide the parameters in. This section is optional and can be omitted when necessary.

parameters
- This section allows for specifying input parameters that have to be provided when instantiating the template. The section is optional and can be omitted when no input is required.

Openstack Heat

•Heat Orchestration template

resources

- This section contains the declaration of the single resources of the template. This section with at least one resource should be defined in any HOT template, or the template would not really do anything when being instantiated.

outputs

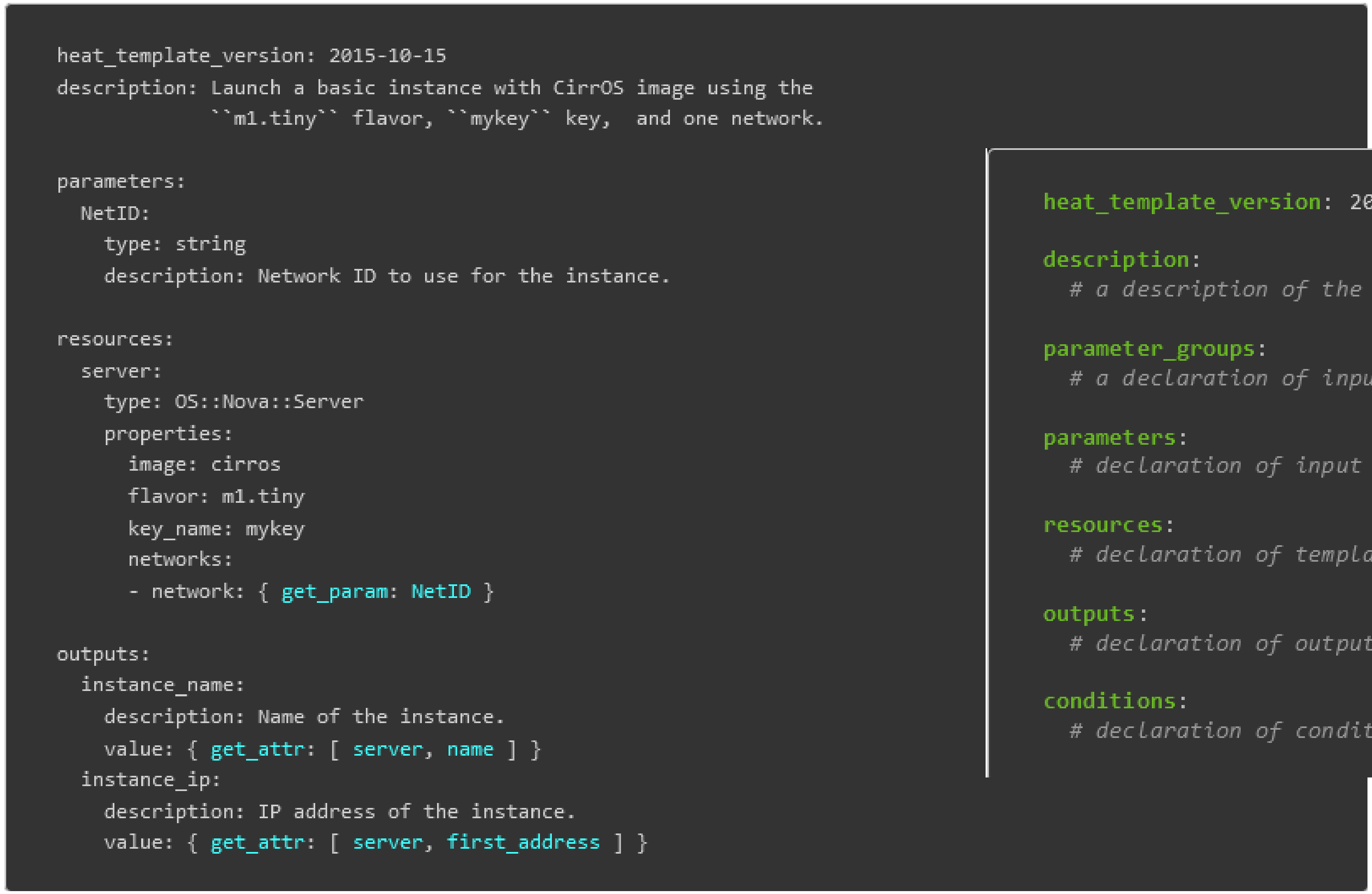
- This section allows for specifying output parameters available to users once the template has been instantiated. This section is optional and can be omitted when no output values are required.

conditions

- This optional section includes statements which can be used to restrict when a resource is created or when a property is defined

Openstack Heat

- **Heat Orchestration template**

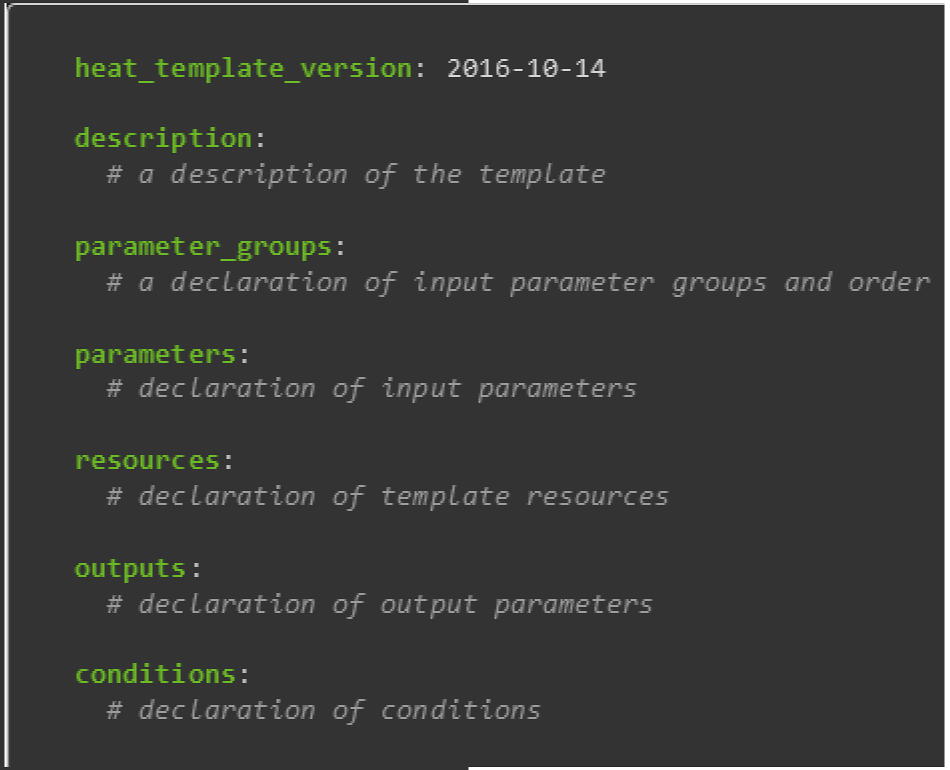


```
heat_template_version: 2015-10-15
description: Launch a basic instance with CirrOS image using the
             ``m1.tiny`` flavor, ``mykey`` key,  and one network.

parameters:
  NetID:
    type: string
    description: Network ID to use for the instance.

resources:
  server:
    type: OS::Nova::Server
    properties:
      image: cirros
      flavor: m1.tiny
      key_name: mykey
      networks:
      - network: { get_param: NetID }

outputs:
  instance_name:
    description: Name of the instance.
    value: { get_attr: [ server, name ] }
  instance_ip:
    description: IP address of the instance.
    value: { get_attr: [ server, first_address ] }
```



```
heat_template_version: 2016-10-14

description:
  # a description of the template

parameter_groups:
  # a declaration of input parameter groups and order

parameters:
  # declaration of input parameters

resources:
  # declaration of template resources

outputs:
  # declaration of output parameters

conditions:
  # declaration of conditions
```

Openstack Heat

•Installing&Configuring Heat

- Create database

```
GRANT ALL PRIVILEGES ON heat.* TO 'heat'@'localhost' \
  IDENTIFIED BY 'HEAT_DBPASS';
GRANT ALL PRIVILEGES ON heat.* TO 'heat'@'%' \
  IDENTIFIED BY 'HEAT_DBPASS';
```

- Create user

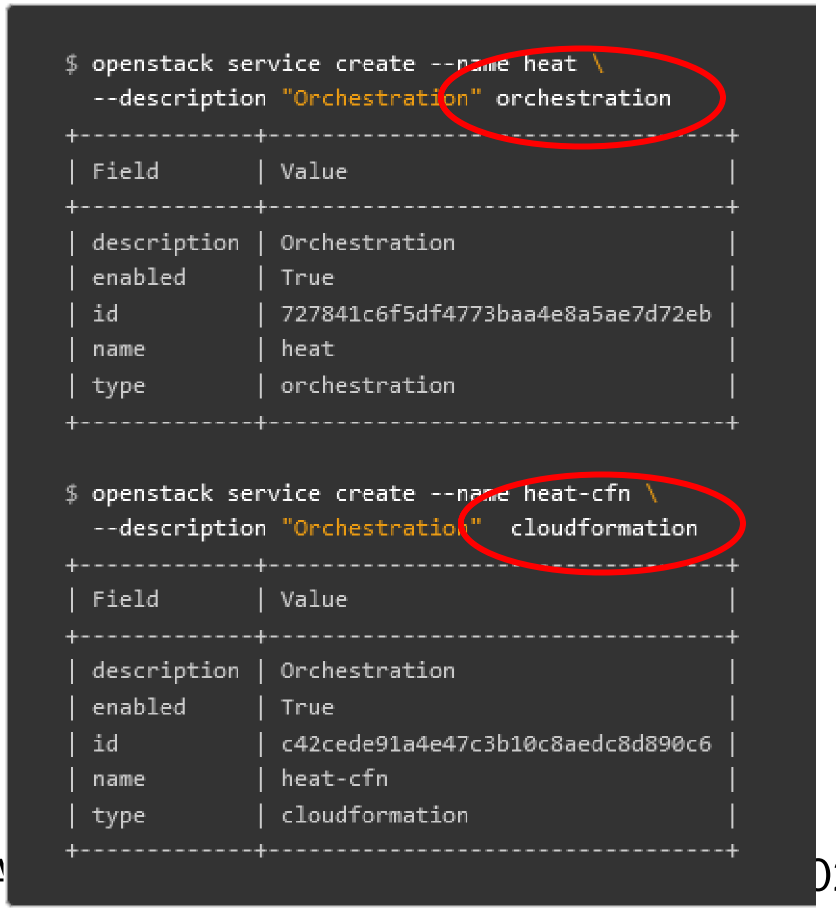
```
$ openstack user create --domain default --password-prompt heat
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | e0353a670a9e496da891347c589539e9 |
| enabled   | True                             |
| id        | ca2e175b851943349be29a328cc5e360 |
| name      | heat                             |
+-----------+----------------------------------+
```

```
$ openstack role add --project service --user heat admin
```

Openstack Heat

•Installing&Configuring Heat

- Create the **services** credentials



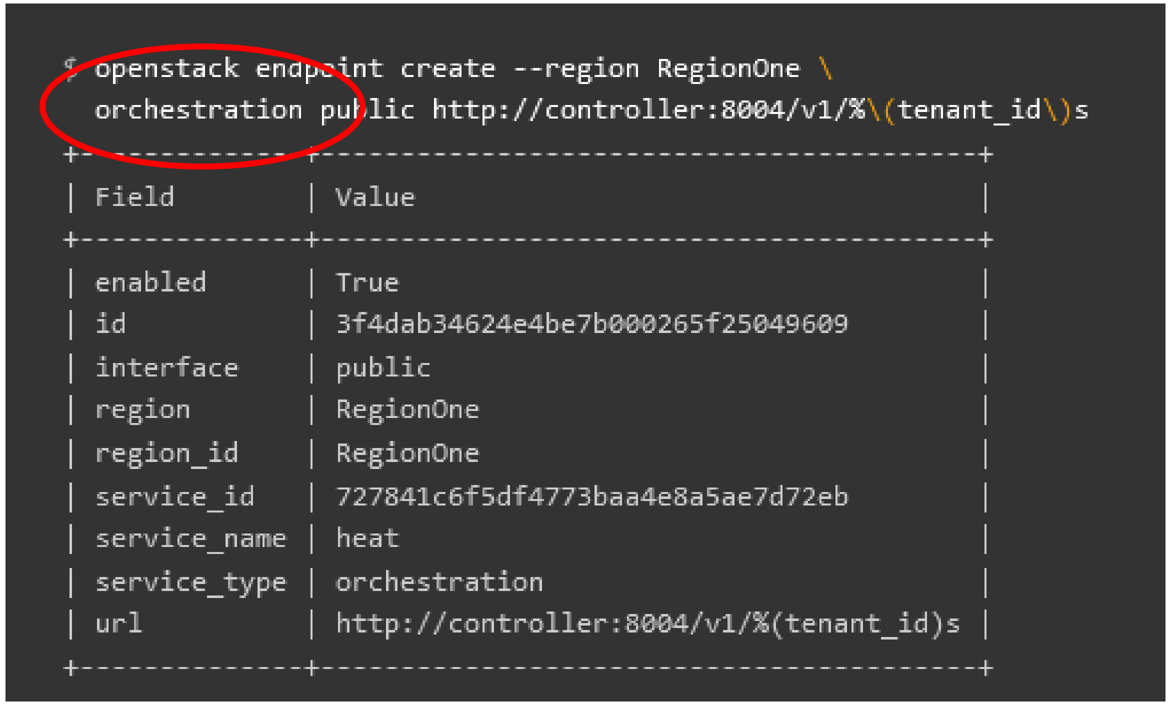
```
$ openstack service create --name heat \
  --description "Orchestration" orchestration
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Orchestration                    |
| enabled     | True                             |
| id          | 727841c6f5df4773baa4e8a5ae7d72eb |
| name        | heat                             |
| type        | orchestration                    |
+-------------+----------------------------------+

$ openstack service create --name heat-cfn \
  --description "Orchestration"  cloudformation
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Orchestration                    |
| enabled     | True                             |
| id          | c42cede91a4e47c3b10c8aedc8d890c6 |
| name        | heat-cfn                         |
| type        | cloudformation                   |
+-------------+----------------------------------+
```

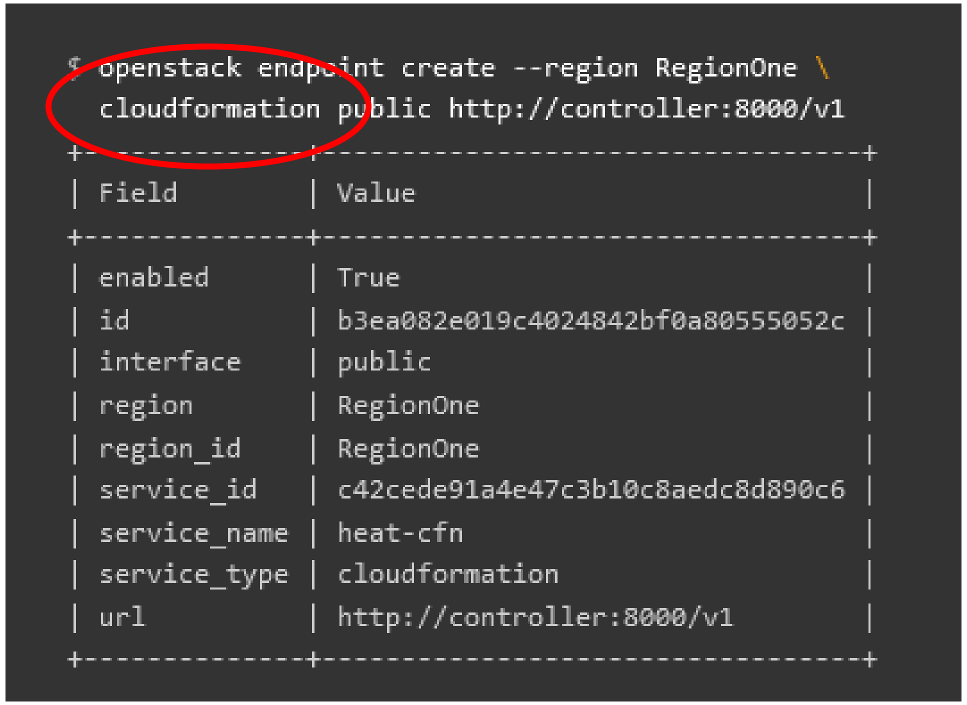
Openstack Heat

•Installing&Configuring Heat

- Create the Orchestration service API endpoints
  - Public, Internal, Admin



```
$ openstack endpoint create --region RegionOne \
  orchestration public http://controller:8004/v1/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 3f4dab34624e4be7b000265f25049609        |
| interface    | public                                  |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | 727841c6f5df4773baa4e8a5ae7d72eb        |
| service_name | heat                                    |
| service_type | orchestration                           |
| url          | http://controller:8004/v1/%(tenant_id)s |
+--------------+-----------------------------------------+
```

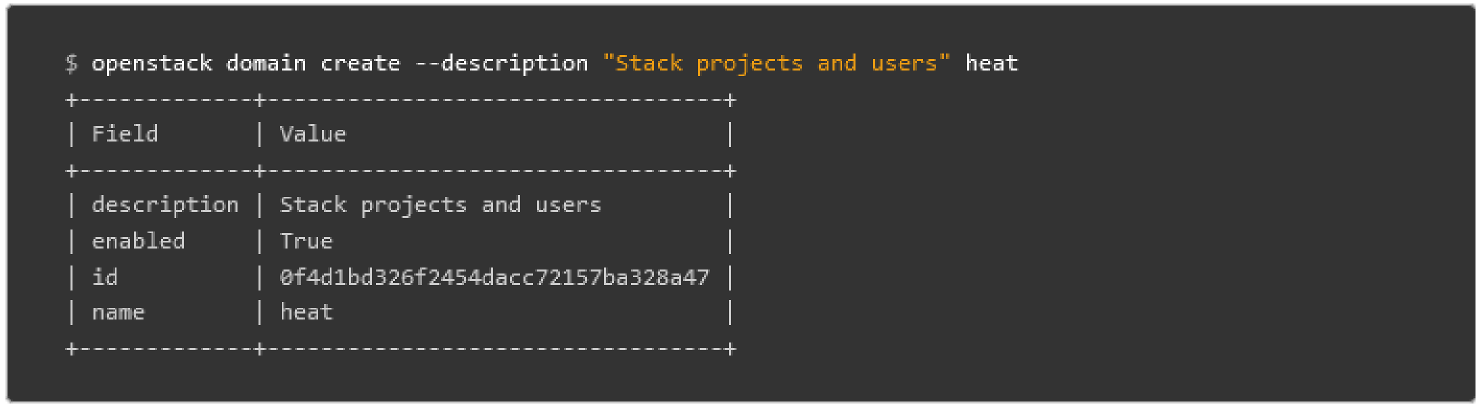


```
$ openstack endpoint create --region RegionOne \
  cloudformation public http://controller:8000/v1
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | b3ea082e019c4024842bf0a80555052c |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | c42cede91a4e47c3b10c8aedc8d890c6 |
| service_name | heat-cfn                         |
| service_type | cloudformation                   |
| url          | http://controller:8000/v1        |
+--------------+----------------------------------+
```

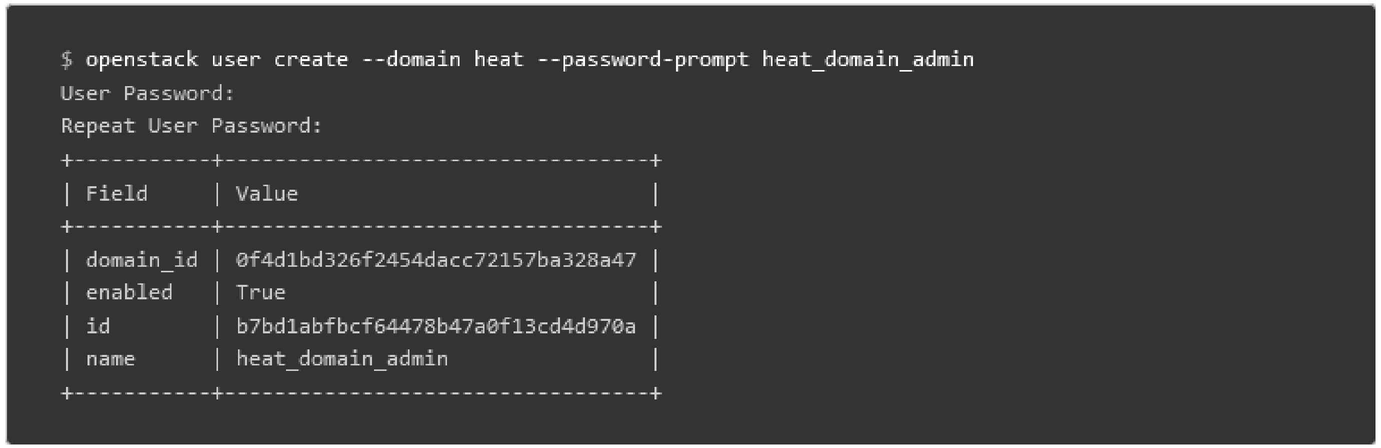
Openstack Heat

•Installing&Configuring Heat

- Create the **Heat** domain and the **heat_domain_admin** user to manage projects and users in the **heat** domain



```
$ openstack domain create --description "Stack projects and users" heat
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Stack projects and users         |
| enabled     | True                             |
| id          | 0f4d1bd326f2454dacc72157ba328a47 |
| name        | heat                             |
+-------------+----------------------------------+
```

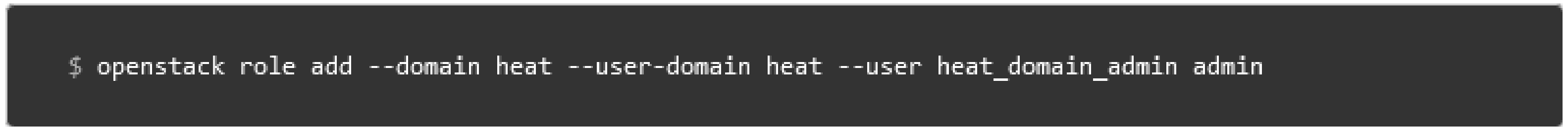


```
$ openstack user create --domain heat --password-prompt heat_domain_admin
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | 0f4d1bd326f2454dacc72157ba328a47 |
| enabled   | True                             |
| id        | b7bd1abfbcf64478b47a0f13cd4d970a |
| name      | heat_domain_admin                |
+-----------+----------------------------------+
```

Openstack Heat

•Installing&Configuring Heat

- Add the **`admin`** role to the **`heat_domain_admin`** user in the `heat` domain to enable administrative stack management privileges by the **`heat_domain_admin`** user

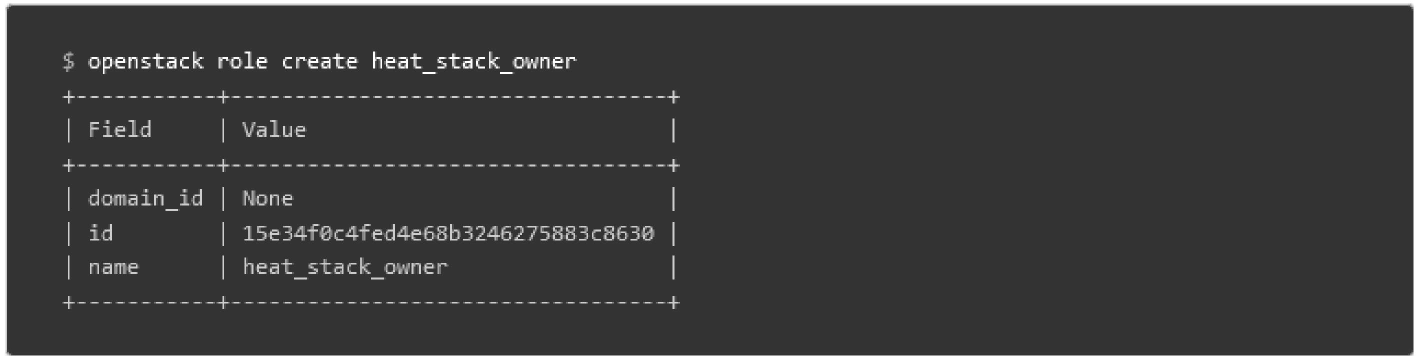


```
$ openstack role add --domain heat --user-domain heat --user heat_domain_admin admin
```

Openstack Heat

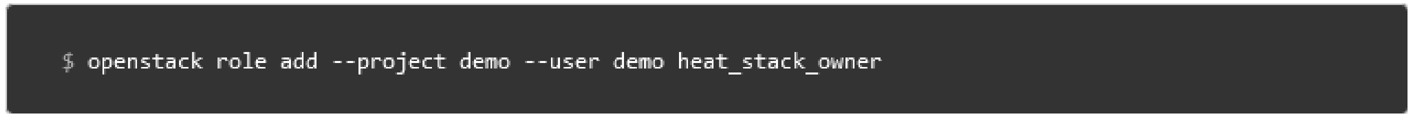
•Installing&Configuring Heat

- Create the **`heat_stack_owner`** role



```
$ openstack role create heat_stack_owner
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | None                             |
| id        | 15e34f0c4fed4e68b3246275883c8630 |
| name      | heat_stack_owner                 |
+-----------+----------------------------------+
```

- Assign the **`heat_stack_owner`** role



```
$ openstack role add --project demo --user demo heat_stack_owner
```

Openstack Heat

•Installing&Configuring Heat

- Install packages

```
# yum install openstack-heat-api openstack-heat-api-cfn \
  openstack-heat-engine
```

- Edit the `/etc/heat/heat.conf`
  - `Connection, transport url, keystone_authtoken, ecc...`

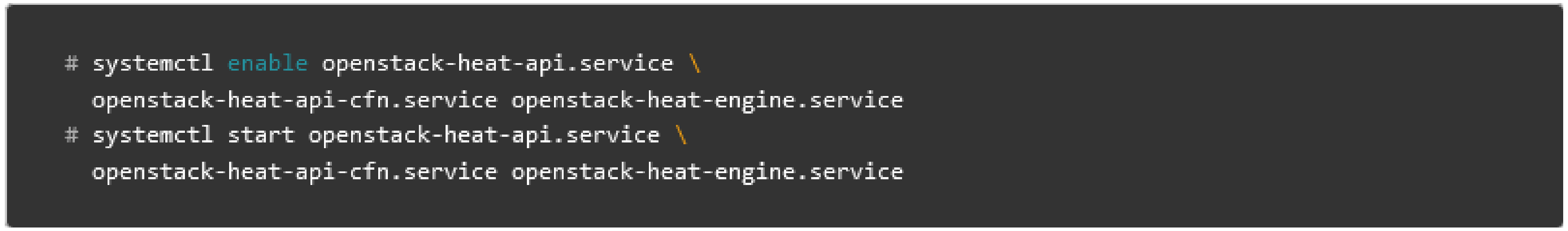
- Populate the Orchestration database

```
# su -s /bin/sh -c "heat-manage db_sync" heat
```

Openstack Heat

•Installing&Configuring Heat

- Start Orchestration services



```
# systemctl enable openstack-heat-api.service \
  openstack-heat-api-cfn.service openstack-heat-engine.service
# systemctl start openstack-heat-api.service \
  openstack-heat-api-cfn.service openstack-heat-engine.service
```

Openstack Heat

References

- https://wiki.oasis-open.org/tosca/TOSCA-implementations
- https://docs.openstack.org/heat/latest/
- https://docs.openstack.org/heat/latest/install/install.html
- https://docs.openstack.org/heat/latest/install/launch-instance.html
- https://docs.openstack.org/heat/latest/developing_guides/architecture.html
- https://www.openstack.org/summit/austin-2016/summit-schedule/events/7398/cern-and-science-clouds-in-europe-with-tosca-openstack-heat-and-the-heat-translator

Openstack Magnum

- **Mangnum**
  - Container Orchestration
  - Architecture
  - Installation&Configuration
  - References

Openstack Magnum

- **Magnum** is an **OpenStack API service** making container orchestration engines (**COE**) such as Docker Swarm, Kubernetes, and Apache Mesos available as first class resources in OpenStack.
  - uses **Heat** to orchestrate an OS image which contains Docker and Kubernetes and runs that image in either virtual machines or bare metal in a cluster configuration.
  - Multitenancy support with a better integration of cluster separation

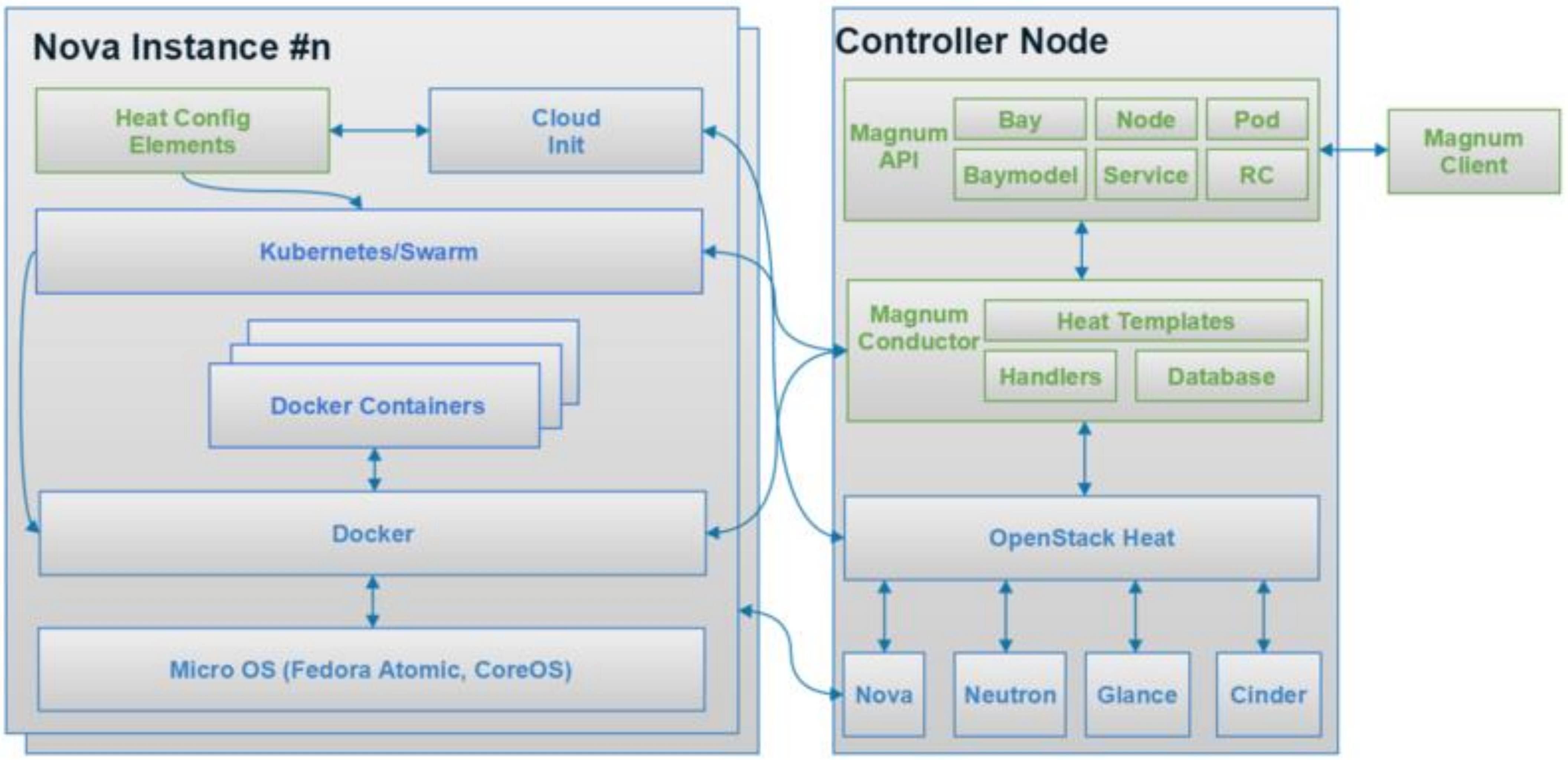
Openstack Magnum

- Terminology:
  - **Cluster** (previously Bay)
    - A cluster is the construct in which Magnum launches container orchestration engines.
    - After a cluster has been created the user is able to add containers to it either directly, or in the case of the Kubernetes container orchestration engine within pods - a logical construct specific to that implementation.
    - A cluster is created based on a ClusterTemplate.
  - **ClusterTemplate** (previously BayModel)
    - A ClusterTemplate in Magnum is roughly equivalent to a flavor in Nova.
    - It acts as a template that defines options such as the container orchestration engine, keypair and image for use when Magnum is creating clusters using the given ClusterTemplate.
  - **Container Orchestration Engine (COE)**
    - A container orchestration engine manages the lifecycle of one or more containers, logically represented in Magnum as a cluster. Magnum supports a number of container orchestration engines, each with their own pros and cons, including Docker Swarm, Kubernetes, and Mesos.

Magnum service components

- **magnum command-line client**
  - A CLI that communicates with the magnum-api to create and manage container clusters. End developers can directly use the magnum REST API.
- **magnum-api service**
  - An OpenStack-native REST API that processes API requests by sending them to the magnum-conductor via AMQP.
- **magnum-conductor service**
  - Runs on a controller machine and connects to heat to orchestrate a cluster. Additionally, it connects to a Docker Swarm, Kubernetes or Mesos REST API endpoint.

Magnum Architecture



Install & Config

Setup a database, service credentials, and API endpoints:

```
MariaDB [(none)]> CREATE DATABASE magnum;

MariaDB [(none)]> GRANT ALL PRIVILEGES ON magnum.* TO \
    'magnum'@'localhost’ IDENTIFIED BY 'MAGNUM_DBPASS’;

MariaDB [(none)]> GRANT ALL PRIVILEGES ON magnum.* TO \
    'magnum'@'%’ IDENTIFIED BY 'MAGNUM_DBPASS’;
```

Install & Config

Setup OpenStack user, service, endpoint:

```
$ openstack user create --domain default  --password-prompt magnum
$ openstack role add --project service --user magnum admin
$ openstack service create --name magnum --description "OpenStack Magnum" container-infra
$ openstack endpoint create --region RegionOne container-infra public http://CONTROLLER_IP:9511/v1
$ openstack endpoint create --region RegionOne container-infra internal http://CONTROLLER_IP:9511/v1
$ openstack endpoint create --region RegionOne container-infra admin http://CONTROLLER_IP:9511/v1
$ openstack domain create --description "Owns users and projects created by magnum" magnum
$ openstack user create --domain magnum --password-prompt magnum_domain_admin
$ openstack role add --domain magnum --user-domain magnum --user \ magnum_domain_admin admin
```

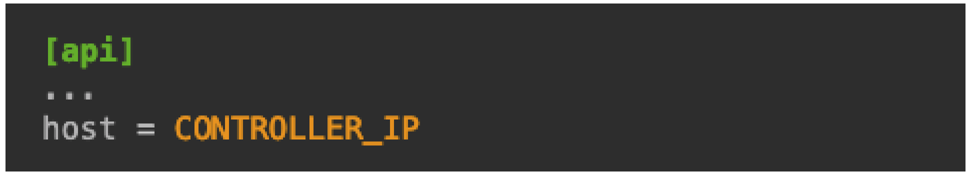
Install Packages

```
# yum install openstack-magnum-api openstack-magnum-conductor python-magnumclient
```

Install & Config

Configure */etc/magnum/magnum.conf:*

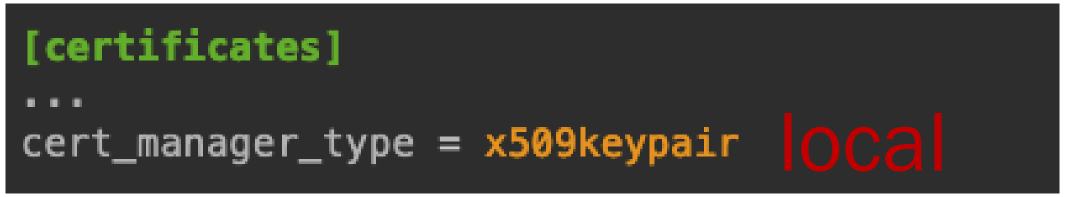
- *[api]* section:



```
[api]
...
host = CONTROLLER_IP
```

- *[certificates]* section, select *barbican* (or *x509keypair*)

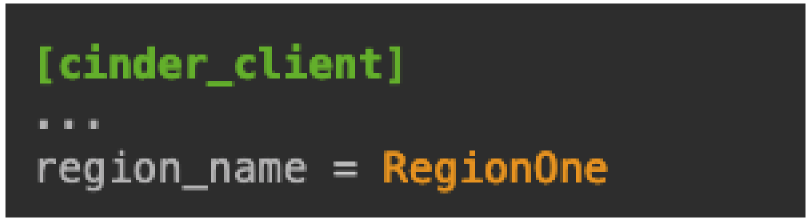
  local



```
[certificates]
...
cert_manager_type = x509keypair
```

local

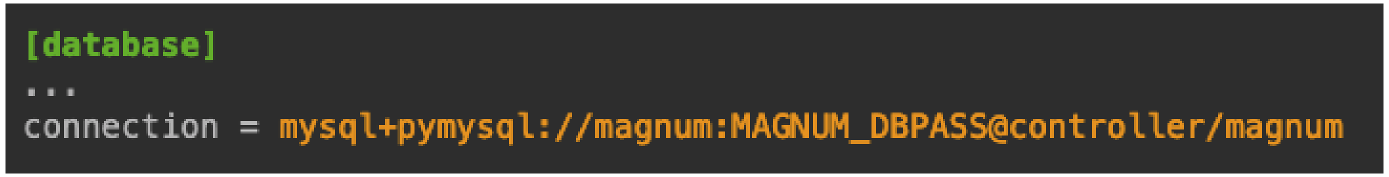
- *[cinder_client]* section, configure the region name



```
[cinder_client]
...
region_name = RegionOne
```

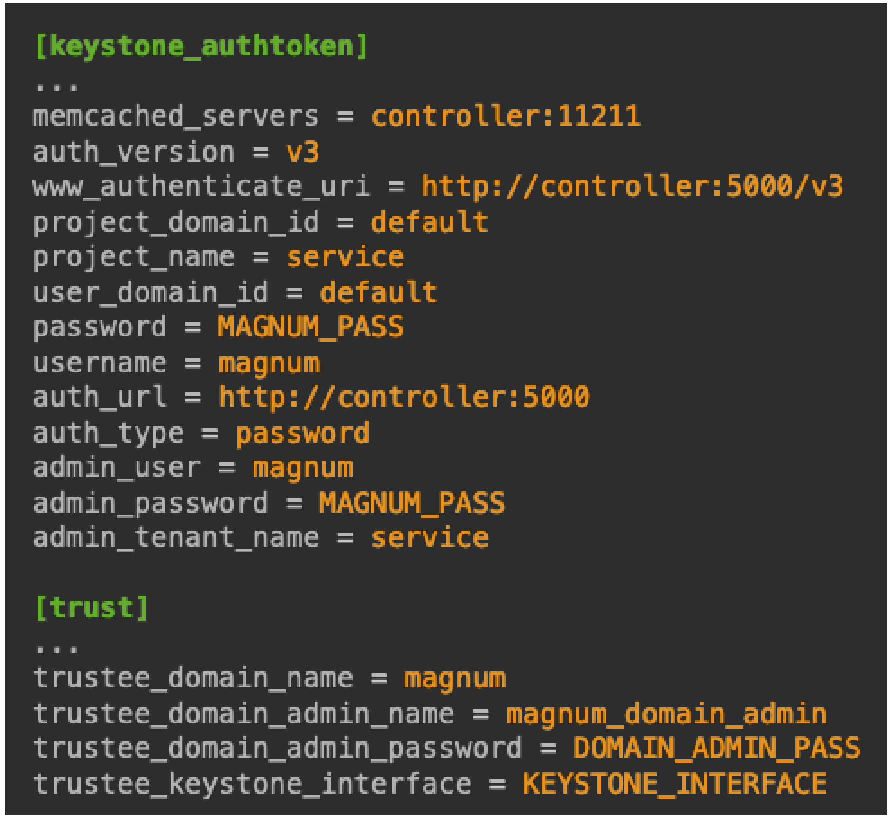
Install & Config

- *[database]* section, configure database access



```
[database]
...
connection = mysql+pymysql://magnum:MAGNUM_DBPASS@controller/magnum
```

- *[keystone_authtoken]* and *[trust]* sections, configure Identity service access:



```
[keystone_authtoken]
...
memcached_servers = controller:11211
auth_version = v3
www_authenticate_uri = http://controller:5000/v3
project_domain_id = default
project_name = service
user_domain_id = default
password = MAGNUM_PASS
username = magnum
auth_url = http://controller:5000
auth_type = password
admin_user = magnum
admin_password = MAGNUM_PASS
admin_tenant_name = service

[trust]
...
trustee_domain_name = magnum
trustee_domain_admin_name = magnum_domain_admin
trustee_domain_admin_password = DOMAIN_ADMIN_PASS
trustee_keystone_interface = KEYSTONE_INTERFACE
```

Install & Config

- *[oslo_messaging_notifications]* section, configure the driver

```
[oslo_messaging_notifications]
...
driver = messaging
```

- *[DEFAULT]* section, configure RabbitMQ message queue access:

```
[DEFAULT]
...
transport_url = rabbit://openstack:RABBIT_PASS@controller
```

- *[oslo_concurrency]* section, configure the lock_path

```
[oslo_concurrency]
...
lock_path = /var/lib/magnum/tmp
```

Install & Config

- Populate Magnum database:

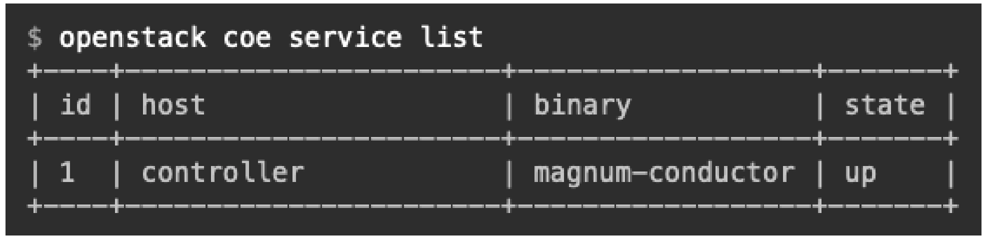
```
# su -s /bin/sh -c "magnum-db-manage upgrade" magnum
```

- Start services and enable their start when system boots:

```
# systemctl enable openstack-magnum-api.service\
              openstack-magnum-conductor.service
# systemctl start openstack-magnum-api.service \
openstack-magnum-conductor.service
```

- Verify service status

```
# systemctl status openstack-magnum-* --no-pager -l
# . keystonerc-admin
# $ openstack coe service list
```



```
$ openstack coe service list
+----+------------------------+-------------------+-------+
| id | host                   | binary            | state |
+----+------------------------+-------------------+-------+
| 1  | controller             | magnum-conductor  | up    |
+----+------------------------+-------------------+-------+
```

Openstack Magnum

References:

- https://docs.openstack.org/magnum/wallaby/install/install-rdo.html

- https://docs.openstack.org/magnum/latest/admin/troubleshooting-guide.html

Openstack Octavia

- **Octavia**
  - Overview
  - Octavia Architecture
  - Install&Configure
  - References

Openstack Octavia

- Octavia is a service-vm based LBaaS implementation which uses haproxy on Nova.
- Deploy Load balancer in virtual Machine.
- The component of Octavia that does load balancing is known as Amphora.
- The component of Octavia that provides command and control of the Amphora is the Octavia controller.

Openstack Octavia

Octavia components

controller - The Controller is the “brains” of Octavia. It consists of five sub-components, which are individual daemons. They can be run on separate back-end infrastructure if desired:

- API Controller - As the name implies, this subcomponent runs Octavia’s API. It takes API requests, performs simple sanitizing on them, and ships them off to the controller worker over the Oslo messaging bus.
- Controller Worker - This subcomponent takes sanitized API commands from the API controller and performs the actions necessary to fulfill the API request.
- Health Manager - This subcomponent monitors individual amphorae to ensure they are up and running, and otherwise healthy. It also handles failover events if amphorae fail unexpectedly.
- Housekeeping Manager - This subcomponent cleans up stale (deleted) database records and manages amphora certificate rotation.
- Driver Agent - The driver agent receives status and statistics updates from provider drivers.

Openstack Octavia

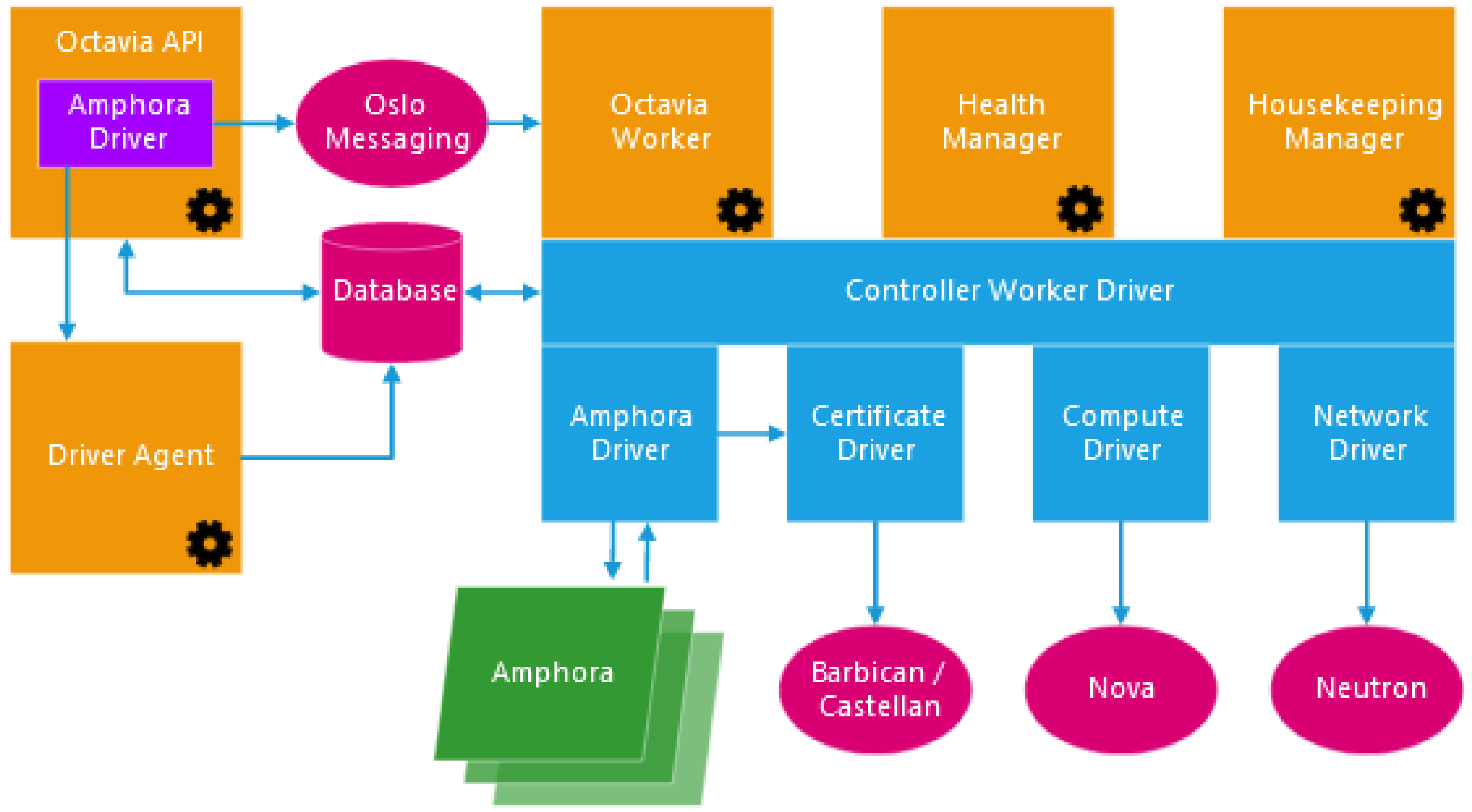
Octavia components

- **amphorae** - Amphorae are the individual virtual machines, containers, or bare metal servers that accomplish the delivery of load balancing services to tenant application environments. In Octavia, the reference implementation of the amphorae image is an Ubuntu virtual machine running HAProxy.

- **network** - Octavia cannot accomplish what it does without manipulating the network environment. Amphorae are spun up with a network interface on the “load balancer network,” and they may also plug directly into tenant networks to reach back-end pool members, depending on how any given load balancing service is deployed by the tenant

Openstack Octavia

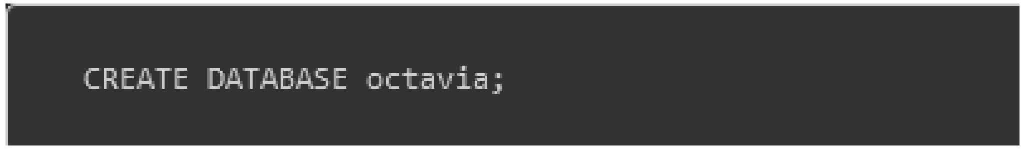
•**Octavia architecture**



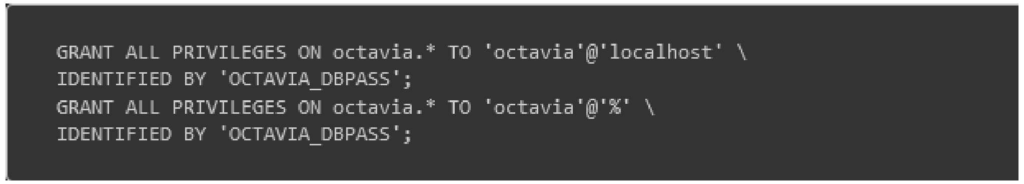
Openstack Octavia

•Installing&Configuring Octavia

- Create database

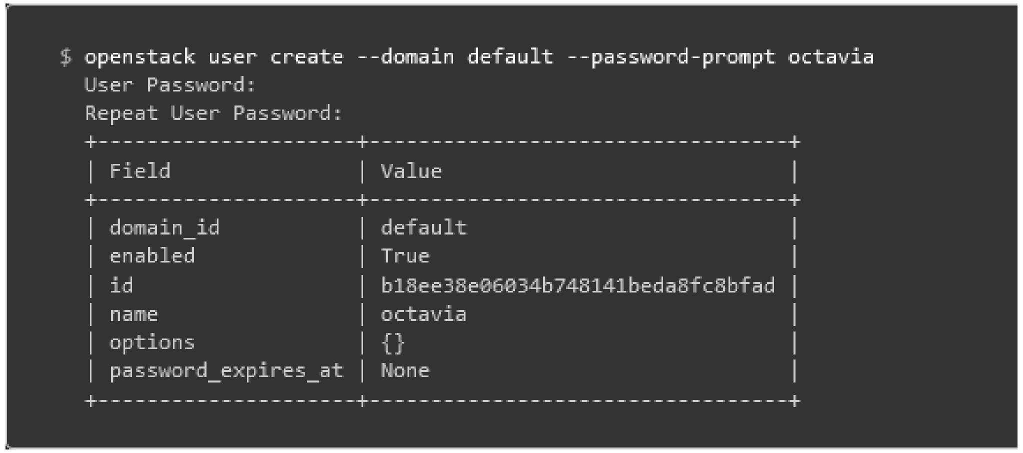


```
CREATE DATABASE octavia;
```

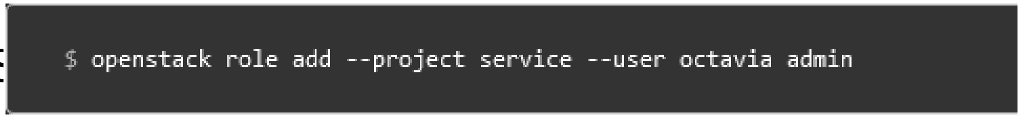


```
GRANT ALL PRIVILEGES ON octavia.* TO 'octavia'@'localhost' \
IDENTIFIED BY 'OCTAVIA_DBPASS';
GRANT ALL PRIVILEGES ON octavia.* TO 'octavia'@'%' \
IDENTIFIED BY 'OCTAVIA_DBPASS';
```

- Create user and assign a role



```
$ openstack user create --domain default --password-prompt octavia
  User Password:
  Repeat User Password:
  +---------------------+----------------------------------+
  | Field               | Value                            |
  +---------------------+----------------------------------+
  | domain_id           | default                          |
  | enabled             | True                             |
  | id                  | b18ee38e06034b748141beda8fc8bfad |
  | name                | octavia                          |
  | options             | {}                               |
  | password_expires_at | None                             |
  +---------------------+----------------------------------+
```

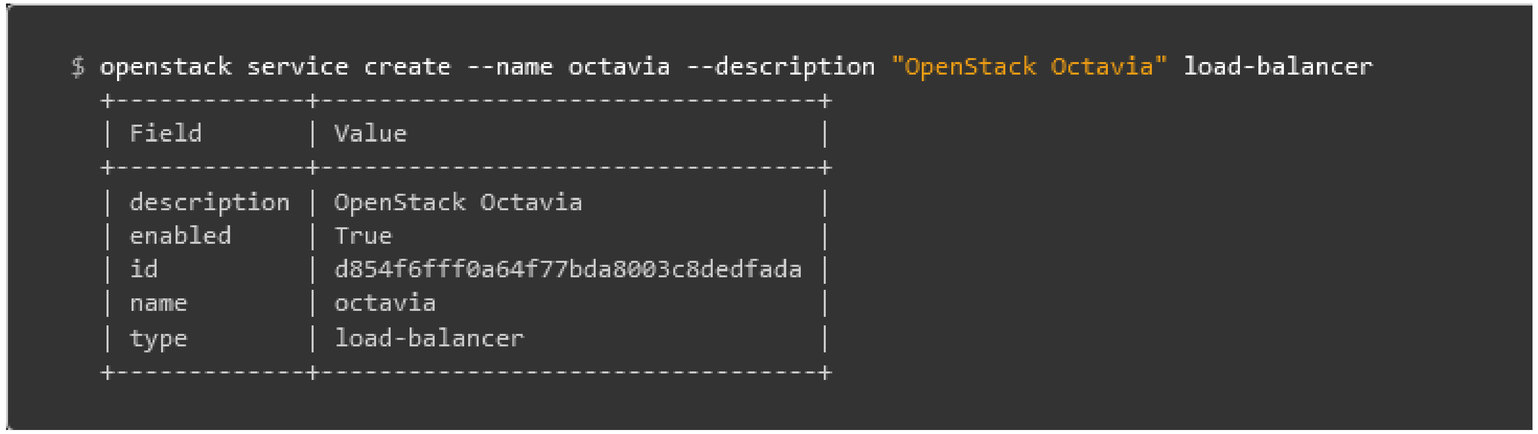


```
$ openstack role add --project service --user octavia admin
```

Openstack Octavia

•Installing&Configuring Octavia

- Create the service credentials

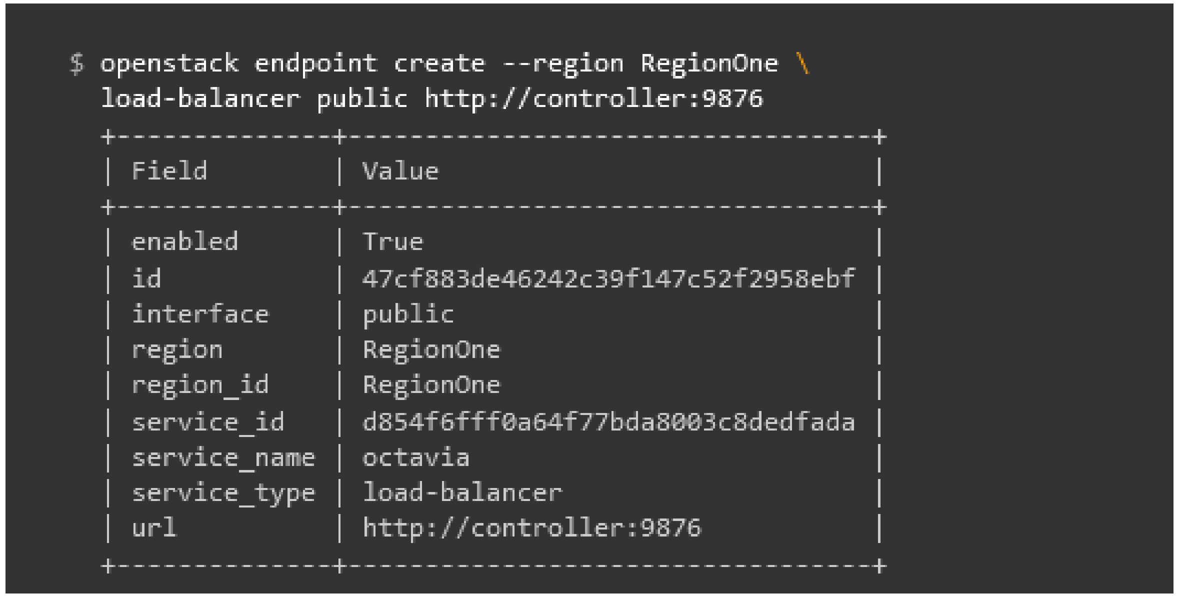


```
$ openstack service create --name octavia --description "OpenStack Octavia" load-balancer
  +-------------+----------------------------------+
  | Field       | Value                            |
  +-------------+----------------------------------+
  | description | OpenStack Octavia                |
  | enabled     | True                             |
  | id          | d854f6fff0a64f77bda8003c8dedfada |
  | name        | octavia                          |
  | type        | load-balancer                    |
  +-------------+----------------------------------+
```

Openstack Octavia

•Installing&Configuring Octavia

- Create the Octavia service API endpoints
    - Public, Internal, Admin



```
$ openstack endpoint create --region RegionOne \
  load-balancer public http://controller:9876
  +--------------+----------------------------------+
  | Field        | Value                            |
  +--------------+----------------------------------+
  | enabled      | True                             |
  | id           | 47cf883de46242c39f147c52f2958ebf |
  | interface    | public                           |
  | region       | RegionOne                        |
  | region_id    | RegionOne                        |
  | service_id   | d854f6fff0a64f77bda8003c8dedfada |
  | service_name | octavia                          |
  | service_type | load-balancer                    |
  | url          | http://controller:9876           |
  +--------------+----------------------------------+
```

Openstack Octavia

•Installing&Configuring Octavia

- Create the Amphora image
- Upload the Amphora image to Glance
- Create a flavor for the amphora image

```
$ openstack flavor create --id 200 --vcpus 1 --ram 1024 \
  --disk 2 "amphora" --private
```

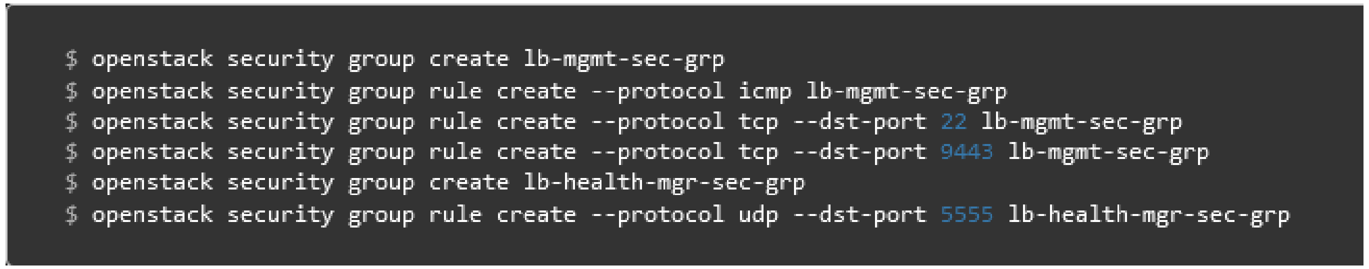
- Install the packages

```
# apt install octavia-api octavia-health-manager octavia-housekeeping \
  octavia-worker python3-octavia python3-octaviaclient
```

Openstack Octavia

•Installing&Configuring Octavia

- Create the certificates
- Create security groups and their rules



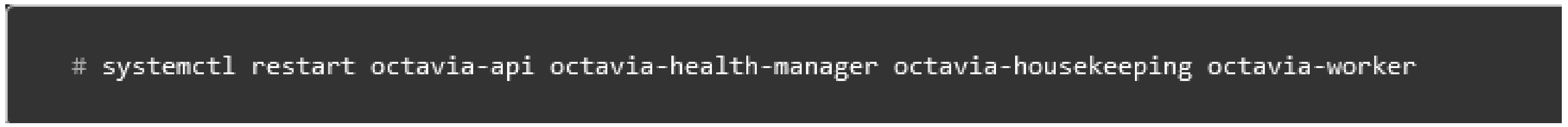
```
$ openstack security group create lb-mgmt-sec-grp
$ openstack security group rule create --protocol icmp lb-mgmt-sec-grp
$ openstack security group rule create --protocol tcp --dst-port 22 lb-mgmt-sec-grp
$ openstack security group rule create --protocol tcp --dst-port 9443 lb-mgmt-sec-grp
$ openstack security group create lb-health-mgr-sec-grp
$ openstack security group rule create --protocol udp --dst-port 5555 lb-health-mgr-sec-grp
```

- Create a key pair for logging in to the amphora instance
- Create dhclient.conf file for dhclient
- Create a network
- Create virtual Ethernet devices (veth)

Openstack Octavia

•Installing&Configuring Octavia

- Edit the `/etc/octavia/octavia.conf` file
- Populate the octavia database
- Restart the services



```
# systemctl restart octavia-api octavia-health-manager octavia-housekeeping octavia-worker
```

Openstack Octavia

References:

- https://docs.openstack.org/octavia/wallaby/

- https://docs.openstack.org/octavia/wallaby/install/install-ubuntu.html