

Hands-On with Heat – OpenStack Service Orchestration

Useful Information

The address for your cloud is <http://192.168.124.81>

Navigate to Project → Orchestration → Stacks and click the button to create a new stack

You may use one of the following methods for inputting your template:

1. Select “Direct Input” in the stack creation pop-up and either type or copy & paste the template from this lab guide into the appropriate box.
2. Copy the contents of the template to a text file, save it to your desktop, and then select “File” as your template source and upload the file.

Delete your stack after each lab exercise.

REMEMBER – INDENTATION/SPACING/CAPITALIZATION MATTERS IN YAML

Exercise 1 – “Hello World”

1. Using one of the steps above, input the text between the dotted lines into the stack creation dialogue for the 'Template Data' field. Leave every other field as is then hit 'Next'.

heat_template_version: 2013-05-23

description: Simple template to deploy a single compute instance

resources:

blog:

type: OS::Nova::Server

properties:

key_name: rashford

image: rashford/wordpress

flavor: m1.tiny

Exercise 2 – “Parameters”

Using one of the steps above, input the text between the dotted lines into the stack creation dialogue

heat_template_version: 2013-05-23

description: Simple template to deploy a single compute instance

parameters:

key_name:

type: string

label: Key Name

description: Name of key-pair to be used for compute instance

image_id:

type: string

label: Image ID

description: Image to be used for compute instance

instance_type:

type: string

label: Instance Type

description: Type of instance (flavor) to be used

resources:

my_instance:

type: OS::Nova::Server

properties:

key_name: { get_param: key_name }

image: { get_param: image_id }

flavor: { get_param: instance_type }

Valid inputs to launch your parameters template:

- Key Name = rashford
- Image ID = rashford/wordpress
- Instance Type = m1.tiny

Exercise 3 – “Defaults”

Using the template text below as a guide, create a **NEW** stack and **MODIFY** the “Parameters” lab template above to include default input for **both** parameters

instance_type:

type: string

label: Instance Type

description: Type of instance (flavor) to be used

default: m1.tiny

Exercise 4 – “Restricting User Input”

Using the template text below as a guide, **MODIFY** the “Parameters” lab template above to restrict the input for **both** parameters

parameters:

instance_type:

type: string

label: Instance Type

description: Type of instance (flavor) to be used

default: m1.tiny

constraints:

- **allowed_values:** [m1.tiny, m1.tiny, m1.medium]

description: Value must be one of m1.tiny, m1.tiny, m1.medium

Exercise 5 – “2 Servers”

Using the template text below as a guide, **MODIFY** the “Parameters” lab template above to add in a second server

database:

type: OS::Nova::Server

properties:

key_name: rashford

image: rashford/mysql

flavor: m1.tiny

Exercise 6 – “Network”

Using the template text below, launch the blog service

heat_template_version: 2013-05-23

description: Simple template to deploy a single compute instance

parameters:

private_net_cidr:

type: string

description: Data network address (CIDR notation)

default: 172.16.0.0/24

resources:

wordpress:

type: OS::Nova::Server

properties:

key_name: rashford

image: rashford/wordpress

flavor: m1.tiny

networks:
- port: { get_resource: wordpress_data_port }
- port: { get_resource: wordpress_port }

database:
type: OS::Nova::Server
properties:
key_name: rashford
image: rashford/mysql
flavor: m1.tiny
networks:
- port: { get_resource: mysql_data_port }

private_net:
type: OS::Neutron::Net
properties:
name: data

private_subnet:
type: OS::Neutron::Subnet
properties:
network_id: { get_resource: private_net }
cidr: { get_param: private_net_cidr }

wordpress_port:
type: OS::Neutron::Port
properties:
network_id: 79a090d5-b626-4bdd-9c17-84a6d01ff0b0
security_groups: [{ get_resource: www_group }]

wordpress_data_port:
type: OS::Neutron::Port
properties:
network_id: { get_resource: private_net }
fixed_ips:
- subnet_id: { get_resource: private_subnet }

wordpress_floating_ip:
type: OS::Neutron::FloatingIP
properties:
floating_network_id: 2b85b72c-21fc-4b91-b990-24fac381658d
port_id: { get_resource: wordpress_port }

mysql_data_port:
type: OS::Neutron::Port
properties:
network_id: { get_resource: private_net }
security_groups: [{ get_resource: mysql_group }]

```
fixed_ips:
  - subnet_id: { get_resource: private_subnet }
```

```
www_group:
  type: OS::Neutron::SecurityGroup
  properties:
    description: Add security group rules for server
    name: www
    rules:
      - remote_ip_prefix: 0.0.0.0/0
        protocol: tcp
        port_range_min: 80
        port_range_max: 80
      - remote_ip_prefix: 0.0.0.0/0
        protocol: tcp
        port_range_min: 443
        port_range_max: 443
      - remote_ip_prefix: 0.0.0.0/0
        protocol: icmp
```

```
mysql_group:
  type: OS::Neutron::SecurityGroup
  properties:
    description: Add security group rules for server
    name: mysql
    rules:
      - remote_ip_prefix: 0.0.0.0/0
        protocol: tcp
        port_range_min: 3306
        port_range_max: 3306
      - remote_ip_prefix: 0.0.0.0/0
        protocol: icmp
```

Exercise 7 – “Volumes”

MODIFY the “Network” lab template above to add in volume storage

```
wordpress_vol:
  type: OS::Cinder::Volume
  properties:
    size: { get_param: wordpress_vol_size }

wordpress_vol_att:
  type: OS::Cinder::VolumeAttachment
  properties:
    instance_uuid: { get_resource: wordpress }
    volume_id: { get_resource: wordpress_vol }
    mountpoint: /dev/vdb
```

```
mysql_vol:
  type: OS::Cinder::Volume
  properties:
    size: { get_param: mysql_vol_size }

mysql_vol_att:
  type: OS::Cinder::VolumeAttachment
  properties:
    instance_uuid: { get_resource: database }
    volume_id: { get_resource: mysql_vol }
    mountpoint: /dev/vdb
```

Exercise 8 – “Auto-Scaling/Load Balancing”

Pull the `apache-autoscaling.yaml` file and the `lb_server.yaml` file from the web server at <http://192.168.124.1/heat> and study them.

Launch the `apache-autoscaling.yaml` service and see it come up.

Experiment with different settings.

For example, how can you change the default number of servers that start up?

The current setup will deploy autoscale workloads one at a time. Can you set it to do it 3 at a time?

It currently is set to scale up if the cpu usage goes over 50% for 1 minute. Can you change these to make it scale up more aggressively?
