Working with HEAT and OpenStack Orchestration

Andrew Mallett
LINUX AUTHOR AND TRAINER

@theurbanpenguin www.theurbanpenguin.com

Imagine life where you can launch all of your Instances in one go and similarly, clean up the environment by deleting them; imagine HEAT

Objectives

Understanding HEAT

Writing HOT files

Launch a Stack

Delete a Stack

Launch Stack with Parameters

Restacking with HEAT

HEAT

HEAT is an orchestration mechanism for OpenStack. Often used to deploy instances but can be used to deploy many OpenStack components. Templates can be written in CFN or HOT format. HOT being the command HEAT OpenStack Template format written in YAML

HEAT Services

```
== Heat services ==
openstack-heat-api:          active
openstack-heat-api-cfn:      inactive  (disabled on boot)
openstack-heat-api-cloudwatch: inactive  (disabled on boot)
openstack-heat-engine:        active
```

Verify HEAT Services

Template Format HOT

Version

Optional Parameters

Resources

Optional Output

```
heat_template_version: 2015-10-15
description: Deploy Cirros VM
resources:
  resource1:
    properties:
      key_name: host
      image: cirros
      flavor: m1.tiny
      name: vm1
```

Simple HOT Template

Launching a single instance

```
$ heat template-version-list
```

Versions

Version numbers can be obtained from the command line client. We can normally write to the latest version that shows. 2015-10-15 is the Liberty release version.

```
$ heat stack-create -f stack.yml my_stack

$ heat stack-list

$ heat stack show my_stack

$ heat stack-delete my_stack
```

Deploy Stack

Stacks can be deployed from the Dashboard or the CLI. If you are low on resources terminate existing. Deleting the stack will terminate all Instances created from the stack.

Deploy Simple Stack

User_Data

We can make use as we have seen before with scripts passed through as user_data to instances at build time.

Extract from HOT File

```
resources:
    my_vm1:
      type: OS::Nova::Server
      properties:
        key_name: host
        image: cirros
        flavor: m1.tiny
        name: vm1
        user_data_format: RAW
        user_data: |
          #!/bin/sh
          echo '192.168.1.1 router' >> /etc/hosts
```

OS::Nova::KeyPair

The schema for this resource type allows for the creation of Key Pairs in OpenStack. When creating new instances it may be appropriate that they are accessed through their own key.

```
heat_template_version: 2015-10-15
description: KEY PAIR
resources:
  cloud_key:
    type: OS::Nova::KeyPair
    properties:
      name: cloud
      public_key: 'ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDQJn4KpHnTXsPWnkvq5F54H/wuEZ4
i2U4HSBGA/AG3Qzk9xWg6u+nJzbnWN3i85RCKnb6S2CdYraYg3oaZl3k/iT
TBBJOyw2Efz6O21wXzvBMST30o5Z8FHIGjELHSkDPvPZam0Bk1hFN2IHVyT
BDqbn0ZNSj9EPXrp05wsy91RpqRLPnetUPhWBSirHvS4+WOlFzIoNkMbCim
Jmstszag02BuqsypH6Z3mjVeXd+tuGB6yWUREdFk5IlnaM+4ju9XysSfL1V
eo1bb79bG9Jpf22jVwFJ7LNhYJNV6dUBxtvijfft1/xctEZf4mSzs+6/7aZ
beZSXJwce2hJYaqFNV root@packstack'
```

Optional Parameters

Should we need to pass variable data through to the resources we can utilize Parameters

```yaml
heat_template_version: 2015-10-15
description: Simple Template
parameters:
  image_id:
    type: string
    label: Image ID
    description: Image to be used for compute instance
resources:
  my_vm1:
    type: OS::Nova::Server
    properties:
      key_name: host
      image: { get_param: image_id }
      flavor: m1.tiny
      name: vm1
```

```
$ heat stack-create -f stack.yml -P image_id=cirros s1
```

Using Stack Parameters

To use more than one parameter they need to be separated with a semi-colon

Cinder Volumes

Heat can create and attach Cinder Volumes. In this way we can create an instance and have a volume connected to it directly and our example becomes more practical

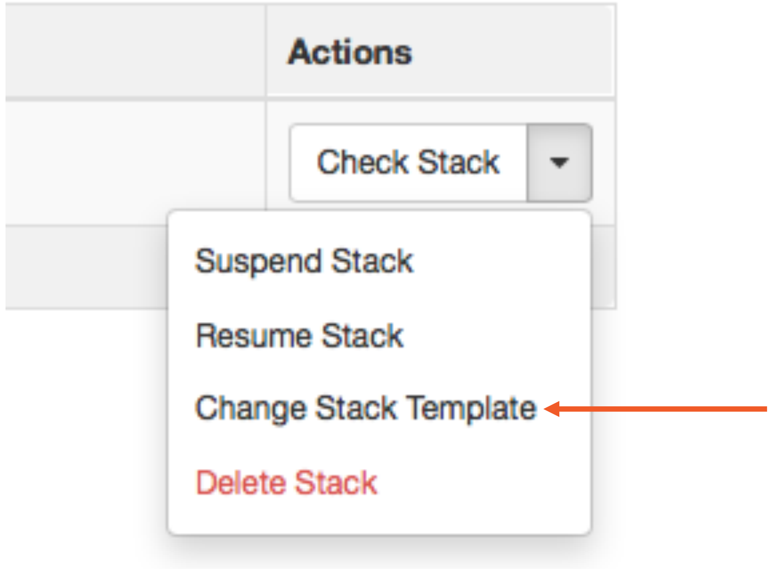
```
my_vm:
         type: OS::Nova::Server
         properties:
           image: cirros
           flavor: m1.tiny
           name: vm1
my_vol:
         type: OS::Cinder::Volume
         properties:
           size: 1
           name: vol1
vol_att:
         type: OS::Cinder::VolumeAttachment
         properties:
           instance_uuid: { get_resource: my_vm }
           volume_id: { get_resource: my_vol }
           mountpoint: /dev/vdb
```

Restack

Allows an existing stack to be reconfigured against a new template

CLI: heat stack-update

GUI:



HEAT

API Service

Engine

API for AWS

Templates

Version

Resources

Instance

```
my_vim:
  type: OS::Nova::Server
  properties:
    image: cirros
    flavor: m1.tiny
```

SSH Key

```
my_key:
  type: OS::Nova::KeyPair
  properties:
    name: cloud
    public_key: ‘ssh-rsa … ‘
```

Volume

```
my_vol:
  type: OS::Cinder::Volume
  properties:
    name: vol1
    size: 1
```

Attachment

```
my_att:
  type: OS::Cinder::Attachment
  properties:
    instance_uuid: { get_resource: my_vm}
    volume_id: { get_resource: my_vol}
    mountpoint: /dev/vdb
```

CLI

heat stack-create

heat stack-list

heat stack-show

heat stack-delete

Next up: Troubleshooting OpenStack