
MIRANTIS
Orchestration (Heat)
training.mirantis.com
Copyright © 2017 Mirantis, Inc. All rights reserved

Heat Module Objectives

- Learn Heat architecture
- Review basics of Heat format
- Understand Autoscaling in OpenStack

MIRANTIS

OpenStack Orchestration

Template-driven engine that allows application developers to describe and automate the deployment of infrastructure

MIRANTIS

What Instructor can talk about:

Companies typically deploy OpenStack to run applications on top of it. Some of those applications have a complex architecture that in physical world may require multiple servers, SAN storage, load balancer, multiple networks, etc. For example, when deploying web servers, typical infrastructure looks like several web-servers behind the load balancer, multiple DB servers with replication, etc.

To replicate the similar infrastructure in the cloud the user would need to create multiple OpenStack resources with some relationships between them. In example above, that would be a separate VM for each of the web and DB servers, neutron LB pool with VIP and Health Monitor.

Heat provides an ability to describe that environment as a template and simplify and automate its creation.

Orchestration vs. Configuration Management

- Orchestration is a sub-category of automation, concerned with coordination of multiple component:
  - For example, servers, networks, volumes, etc..
  - Orchestration is a "higher form" of automation
  - Not just simple or lower-level tasks, but multilayer applications
- Configuration Management is automation of server configuration:
  - Typically a declarative model, based on "fact" discovery of the server
  - Abstracts out the underlying implementation detail of service deployment
- Both are needed to fully automate cloud application deployment.

MIRANTIS

Orchestration Template Languages

- There are numerous templating languages out-there:
  - AWS CloudFormation (CFN)
  - Heat Orchestration Template (HOT)
  - TOSCA (Topology and Orchestration Specification for Cloud Applications)
  - OASIS Cloud Application Management for Platforms (CAMP)
- Heat natively supports HOT and AWS CFN
- HOT part of the family of Orchestration languages, but other formats can be translated before passing them to Heat
  - https://github.com/openstack/heat-translator

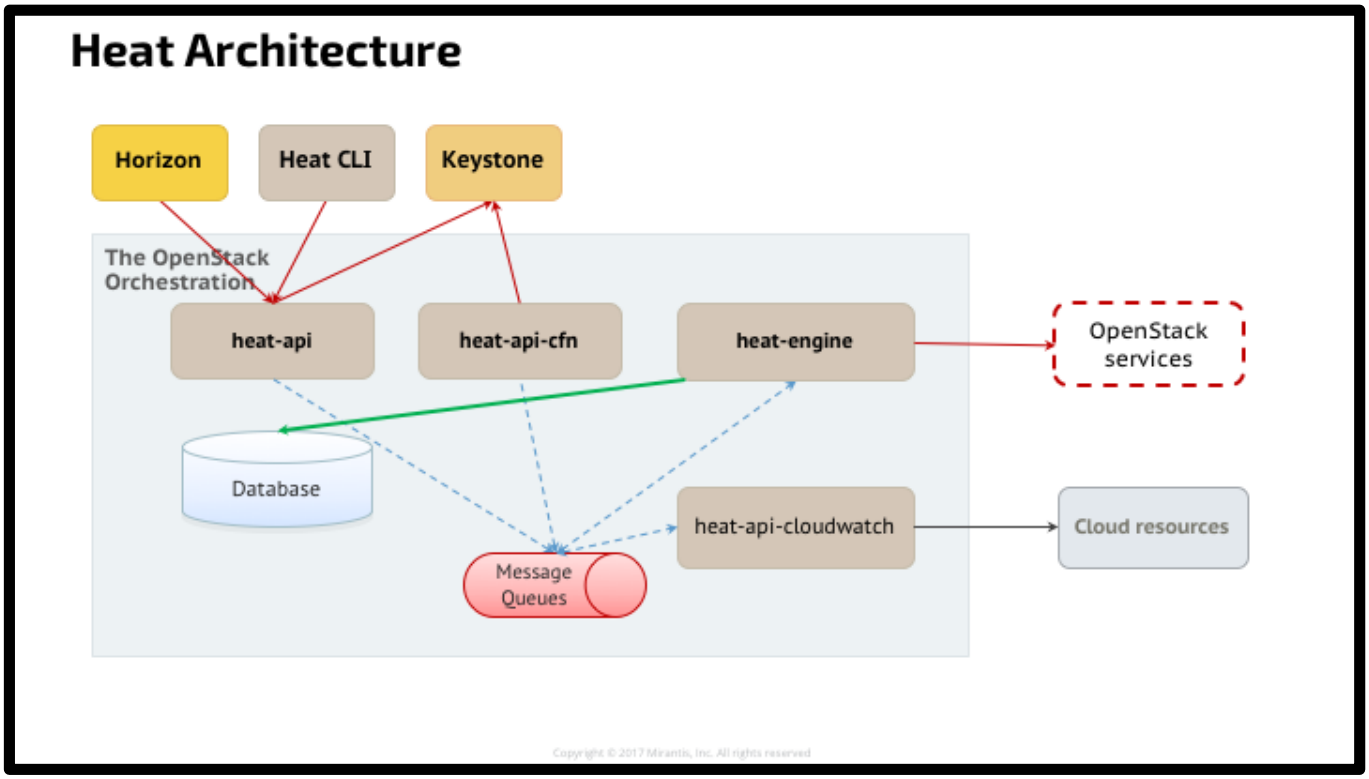
MIRANTIS

https://wiki.openstack.org/wiki/Heat/Vocabulary

There is a work on tools for automatic translation of other template languages. For example heat-translator to translate TOSCA template into Heat DSL.

Heat Capabilities

- Describes the infrastructure for a cloud application – stack (deployment):
  - OpenStack resources: for example, servers, networks, volumes, etc..
  - Relationships between resources: for example, this volume is connected to this server
  - In a text file in the special format – template (blueprint)
- Manages that infrastructure:
  - Automatically changes the infrastructure when the template is modified and re-applied
  - Deletes infrastructure when the stack is deleted
- Integrates with software configuration management tools such as Puppet and Chef:
  - For example: can create VM with puppet server and install puppet clients on VMs
  - Can pass parameters to cloud-init, etc.
- Provides an Autoscaling service that integrates with Ceilometer



https://wiki.openstack.org/wiki/Heat/Vision – this describes a vision, so currently it's a little different.

http://docs.openstack.org/developer/heat/architecture.html

Heat API

- heat-api
  - OpenStack native REST API
- heat-api-cfn
  - provides AWS Query API
- Both communicate with Heat Engine via MQ tell it what actions to perform

MIRANTIS

Heat Engine

- Does all of the orchestration work
- Is a layer where resource integration is implemented
- Contains abstractions to use Auto Scaling and High Availability

MIRANTIS

Heat CloudWatch API

- Ideologically refers to AWS CloudWatch service
  - Gets metrics from stacks
- Is replaced by Ceilometer
- Is used for Autoscaling

MIRANTIS

Heat Orchestration Template

HOT

MIRANTIS

Heat Orchestration Template (HOT)

- Stacks are created from templates.
- HOT is an orchestration document that details everything that is needed to carry out an orchestration.
- HOT has the same structure and abstractions as AWS CloudFormation template.
- Template is written in YAML format (JSON is also supported).

MIRANTIS

HOT Definitions

- Parameters
  - User defined parameters passed into template from CLI or GUI
  - Parameters include type, description, default value, hidden, and constraints
- Resources
  - Resources for Heat to Orchestrate
  - Consist of Type, Properties, DependsOn
  - Produce global attributes
- Outputs
  - Displayed via CLI/GUI to identify important information of template
- Full specification:
  - http://docs.openstack.org/developer/heat/template_guide/

MIRANTIS

Here go to labs

HOT Example: Version and Description

- Mandatory version statement:

```
heat_template_version: 2013-05-23
```

  - Heat script backward compatibility is built into the language
- Optional template description:

```
description: Simple template to deploy a single compute instance
```

- A very simple template:

```
heat_template_version: 2013-05-23
description: Simple template to deploy a single compute instance
resources:
  my_instance:
    type: OS::Nova::Server
    properties:
       key_name: my_key
       image: cirros-0.3.0-i386-uec
       flavor: m1.small
```

MIRANTIS

HOT Example: Parameters

```
parameters:
   <param name>:
     type: <string | number | json | comma_delimited_list | boolean>
     label: <human-readable name of the parameter>
     description: <description of the parameter>
     default: <default value for parameter>
     hidden: <true | false>
     constraints:
       <parameter constraints>
```

```
parameters:
        key_name:
                  type: string
                  description: Name of key-pair to be used for compute instance
        image_id:
                  type: string
                  description: Image to be used for compute instance
        instance_type:
                  type: string
                  description: Type of instance (flavor) to be used
```

MIRANTIS

HOT Example: Constraints

constraints:
- length: { min: 6, max: 8 }
- range: { min: 0, max: 10 }
- allowed_values: [m1.medium, m1.large, m1.xlarge]
- allowed_pattern: "[A-Z]+[a-zA-Z0-9]*"

```
user_name:
    type: string
    label: User Name
    description: User name to be configured for the application
    constraints:
     - length: { min: 6, max: 12 }
           description: User name must be between 6 and 12 characters
     - allowed_pattern: "[a-zA-Z0-9]+"
           description: User name must consist of characters and numbers only.
     - allowed_pattern: "[A-Z]+[a-zA-Z0-9]*"
           description: User name must start with an uppercase character.
```

MIRANTIS

HOT Example: Resources

```
parameters:
  server_name:
    type: string
    description: Name of the server
    default:
    str_replace:
    template: stack_name
    params:
      stack_name: { get_param: "OS::stack_name" }
  key_name:
    type: string
    description: Name of key-pair
  image_id:
    type: string
    description: Image to be used for compute instance
  instance_type:
    type: string
    description: Type of instance (flavor) to be used
```

```
resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      name: { get_param: server_name }
      key_name: { get_param: key_name }
      image: { get_param: image_id }
      flavor: { get_param: instance_type }
      user_data: |
        #!/bin/bash
        sudo apt-get update
```

MIRANTIS

HOT Example: Outputs

```
outputs:
  instance_ip:
    description: The IP address of the deployed instance
    value: {get_attr: [my_instance, first_address]}
```

MIRANTIS

Autoscaling with Heat

MIRANTIS

Heat Auto Scaling Principles

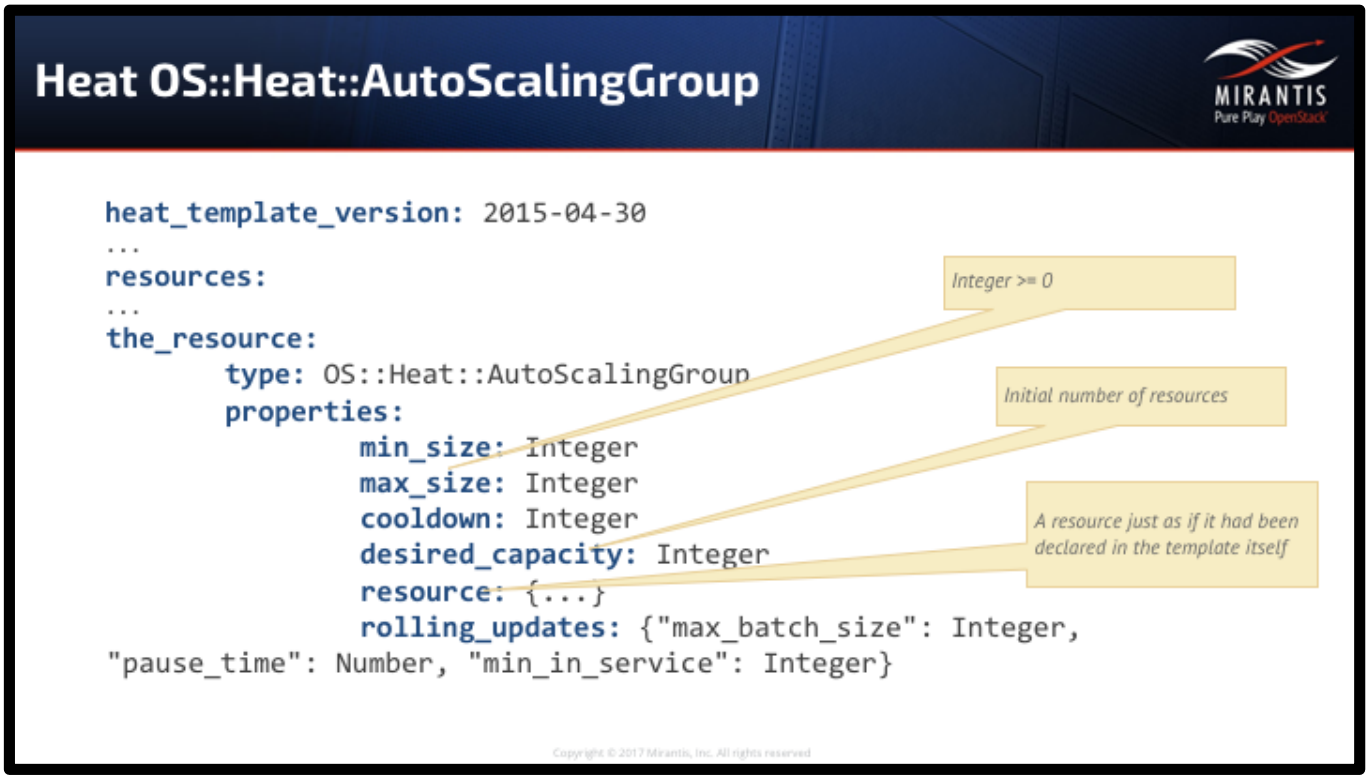
- Heat + Telemetry = Autoscaling
- Aodh is configured to provide alarming (i.e. set and monitor thresholds) and to report back to Heat Engine
- Upscaling and Downscaling scheduling groups are created for actions
- Core functionality is implemented in Heat Engine

MIRANTIS

Heat Auto Scaling Resources

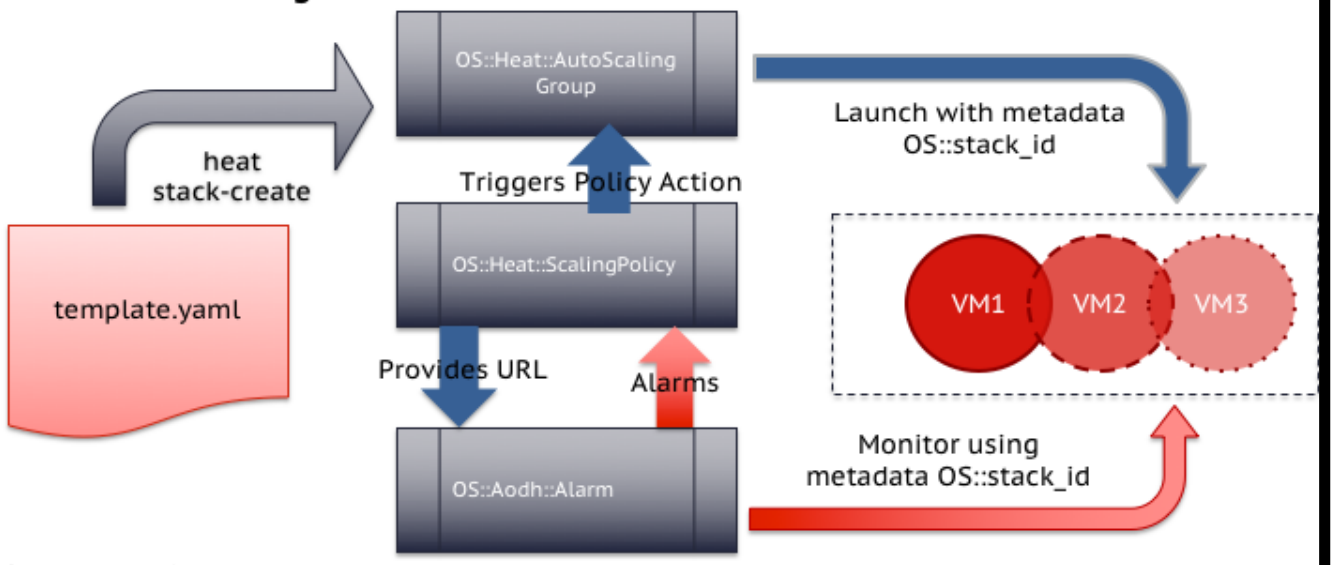
- OS::Heat::AutoScalingGroup
  - An autoscaling group that can scale arbitrary resources
- OS::Heat::ScalingPolicy
  - A resource to manage scaling of OS::Heat::AutoScalingGroup
- OS::Aodh::GnocchiAggregationByResourceAlarm
  - The resource for defining an Aodh alarm
- OS::stack_id
  - Heat “stack” identifier used as glue to tie an OS::Ceilometer::Alarm to an OS::Heat::AutoScalingGroup

Heat OS::Heat::AutoScalingGroup



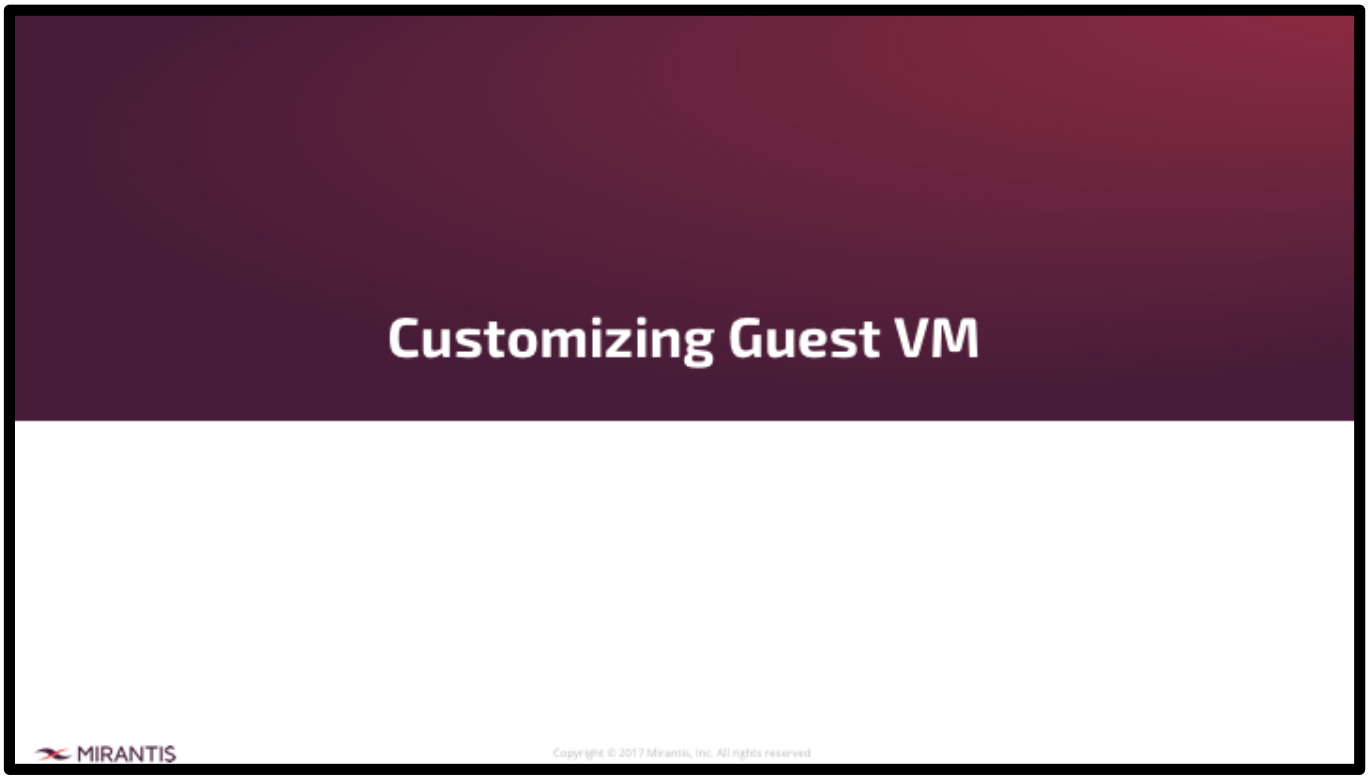
http://docs.openstack.org/developer/heat/template_guide/openstack.html#OS::Heat::AutoScalingGroup

Heat Auto Scaling Process



An auto-scaling example:

https://git.openstack.org/cgit/openstack/heat-templates/tree/hot/autoscaling.yaml



http://docs.openstack.org/developer/heat/template_guide/software_deployment.html

Configuring Software That Runs On Your Servers

- Three broad methods for server configuration:
  - Custom image building
  - User-data boot scripts and cloud-init
  - Software deployment resources
- Custom Image building has the following advantages:
  - Boot speed - No need to download and install anything at boot time.
  - Boot reliability - downloads failure due to transient network failures or inconsistent software repositories.
  - Test verification - verified in test environments before deployment in production.
  - No configuration dependencies - post-boot configuration may require agents installed and enabled on guest VM

MIRANTIS

User-data and cloud-init

cloud-init or cloudbase-init guest agents

MIRANTIS

Nova User Data

- A special key in the metadata service that holds user data provided at boot time that cloud-aware applications in the guest instance can access.
  - `nova boot --image ubuntu-cloudimage --flavor 1 --user-data mydata.file`
- To do something useful with the user data, the virtual machine image must be configured to run a service on boot that retrieves the user data from the metadata service and takes some action based on its content.
- One such application is the cloud-init system. Typical use case is to pass something like a shell script or a configuration file as user data.
- user-data is limited to 16384 bytes.

MIRANTIS

http://docs.openstack.org/user-guide/cli_provide_user_data_to_instances.html

Nova Metadata Service

- The instance (guest VM) can retrieve its user data by querying the metadata service through either the OpenStack metadata API or the EC2 compatibility API:

  ```
  $ curl http://169.254.169.254/openstack/2012-08-10/user_data
  ```

  This is some text

  ```
  $ curl http://169.254.169.254/2009-04-04/user-data
  ```

  This is some text

- Note that the Compute service treats user data as a blob. While cloud-init requires a YAML file, user data can be in any format.

MIRANTIS

https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform/4/html/End_User_Guide/user-data.html

The cloud-init System

- An open-source package from Ubuntu that is the industry standard for bootstrapping cloud servers (initialization on first boot)
- Available on various Linux distributions such as Ubuntu Cloud Images and the official Ubuntu images available on EC2.
- Some of the things it configures are:
  - setting a default locale
  - setting hostname
  - resizing boot disk to that specified in boot flavor
  - adding ssh keys to user's .ssh/authorized_keys so they can log in
  - setting up ephemeral mount points

MIRANTIS

http://cloudinit.readthedocs.org/en/latest/index.html

The cloud-init Supported Formats

- Gzip Compressed Content
- Mime Multi Part archive
- User-Data Script
- Include File
- Cloud Config Data
- Upstart Job
- Cloud Boothook
- Part Handler

begins with: "#!" or "Content-Type: text/x-shellscript"

begins with "#include" or "Content-Type: text/x-include-url"

begins with "#cloud-config" or "Content-Type: text/cloud-config"

MIRANTIS

http://cloudinit.readthedocs.org/en/latest/index.html

Using user-data in heat

```
resources:
...
  db:
    type: OS::Nova::Server
    properties:
    …
    user_data_format: RAW
    user_data:
    str_replace:
    template: |
      #!/bin/bash -v
              yum -y install mariadb mariadb-server
              systemctl enable mariadb.service
              systemctl start mariadb.service
              mysqladmin -u root password $db_rootpassword
    params:
              $db_rootpasswd: {get_attr: [db_root_password, value]}
```

```
webserver:
  type: OS::Nova::Server
  depends_on: db
  properties:
  ...
    user_data_format: RAW
    user_data:
      str_replace:
        template: |
          #!/bin/bash -v
          yum -y install httpd wordpress
          systemctl enable httpd.service
          systemctl start httpd.service
          setsebool -P
 httpd_can_network_connect_db=1
        params:
     $db_name: {get_param: database_name}
```

MIRANTIS

Server Synchronization

WaitCondition and WaitConditionHandle

MIRANTIS

Heat Synchronization Resources

- OS::Heat::WaitCondition
  - A resource to create a synchronization wait point, to be triggered by CM script
- OS::Heat::WaitConditionHandle
  - A resource to signal condition completion. You can signal success by adding –data-binary '{"status": "SUCCESS"}' , or signal failure by adding –data-binary '{"status": "FAILURE"}'

MIRANTIS

WaitCondition/Handle Example

```
wait_condition:
  type: OS::Heat::WaitCondition
  properties:
    handle: {get_resource: wait_handle}
    count: 1
    timeout: 600

wait_handle:
  type: OS::Heat::WaitConditionHandle
```

```
server_instance:
  type: OS::Nova::Server
   properties:
   ...
   user_data:
   str_replace:
   params:
   …
   wc_notify: {get_attr: ['wait_handle', 'curl_cli']}
   template:
     #!/bin/bash -ex
     …
     wc_notify --data-binary '{"status": "SUCCESS"}'
```

MIRANTIS

Software Deployment Resources

Better Integration with Configuration Management System

MIRANTIS

Shortcomings of user-data/cloud-init Model

- Difficult to manage large embedded scripts
- Can only run script at stack create or stack update time
  - "stack update" results in server replacement if any change to user-data text
- Difficult to describe/code complex deployments with many dependencies

Heat CM Tool Integration Resources

- OS::Heat::SoftwareConfig
  - To create a reference to an immutable CM script, optionally parameterized with input values, stored in heat database
- OS::Heat::SoftwareDeployment
  - To associated a OS::Heat::SoftwareConfig to a server. Allows for input values to be defined and passed to the configuration.
  - After configuration script completes its execution, its output is available as this resource's attributes.
  - Can be triggered on stack create, update, suspend, resume, delete
  - Update of input values does not cause server replacement.

MIRANTIS

http://docs.openstack.org/developer/heat/template_guide/openstack.html#OS::Heat::SoftwareConfig

http://docs.openstack.org/developer/heat/template_guide/openstack.html#OS::Heat::SoftwareDeployment

OS::Heat::SoftwareDeployment Image Requirements

- Some tools must already be installed on the glance image being launched in order for SoftwareDeployment to work:
  - os-collect-config, os-refresh-config, os-apply-config
    - Collectively responsible for polling of changes in Heat and Nova metadata, and applying the changes to the instance
  - heat-config, heat-config-hook, heat-config-notify
    - These hooks function in relation to the "group" property of SoftwareConfig
    - The "group" property is used to specify the type of SoftwareConfig hook that will be used to deploy the configuration

MIRANTIS

https://fatmin.com/2016/02/23/openstack-heat-and-os-collect-config/

Single Server Deployment Dependencies

- SoftwareDeployment resources have a "name" property, which can influence the sort-order so that, for example, heat-config will apply "config1" before "config2".
- Template directive "depends_on" can be used to specify an explicit dependency between two or more SoftwareDeployment resources

MIRANTIS

References

- http://docs.openstack.org/developer/heat/template_guide/hot_guide.html
- http://docs.openstack.org/developer/heat/
- http://docs.openstack.org/developer/heat/template_guide/cfn.html
- http://docs.openstack.org/developer/heat/template_guide/hot_spec.html
- https://github.com/openstack/heat-templates
- https://wiki.openstack.org/wiki/Heat/GettingStartedUsingDevstack
- https://wiki.openstack.org/w/images/a/a1/TOSCA_in_Heat_-_20130415.pdf

MIRANTIS

Attendance and Survey

Survey link in e-mail

MIRANTIS