MODULE 5

Automating App Deployment

Module Objectives

By the end of this module, you should understand:

- Interfaces into providing initialization of “cloud” instances
  - Cloud-Init
  - Puppet/Chef in a master-less environment
  - Use via CLI/Horizon
- Alternate Ways of initializing/managing instances
  - HEAT (and Ceilometer for triggering events)

App Deployment - Then vs. Now

Then

- Request VM (wait for IT to deploy)
- Load ISO, deploy OS
- Log in, install pre-compiled packages
- Configure Instance
- Request new REAL added to SLB

- Sometimes
- Create golden image with app pre-set
- Deploy new copy, log in, tweak configs, restart services
- Request new REAL added to SLB

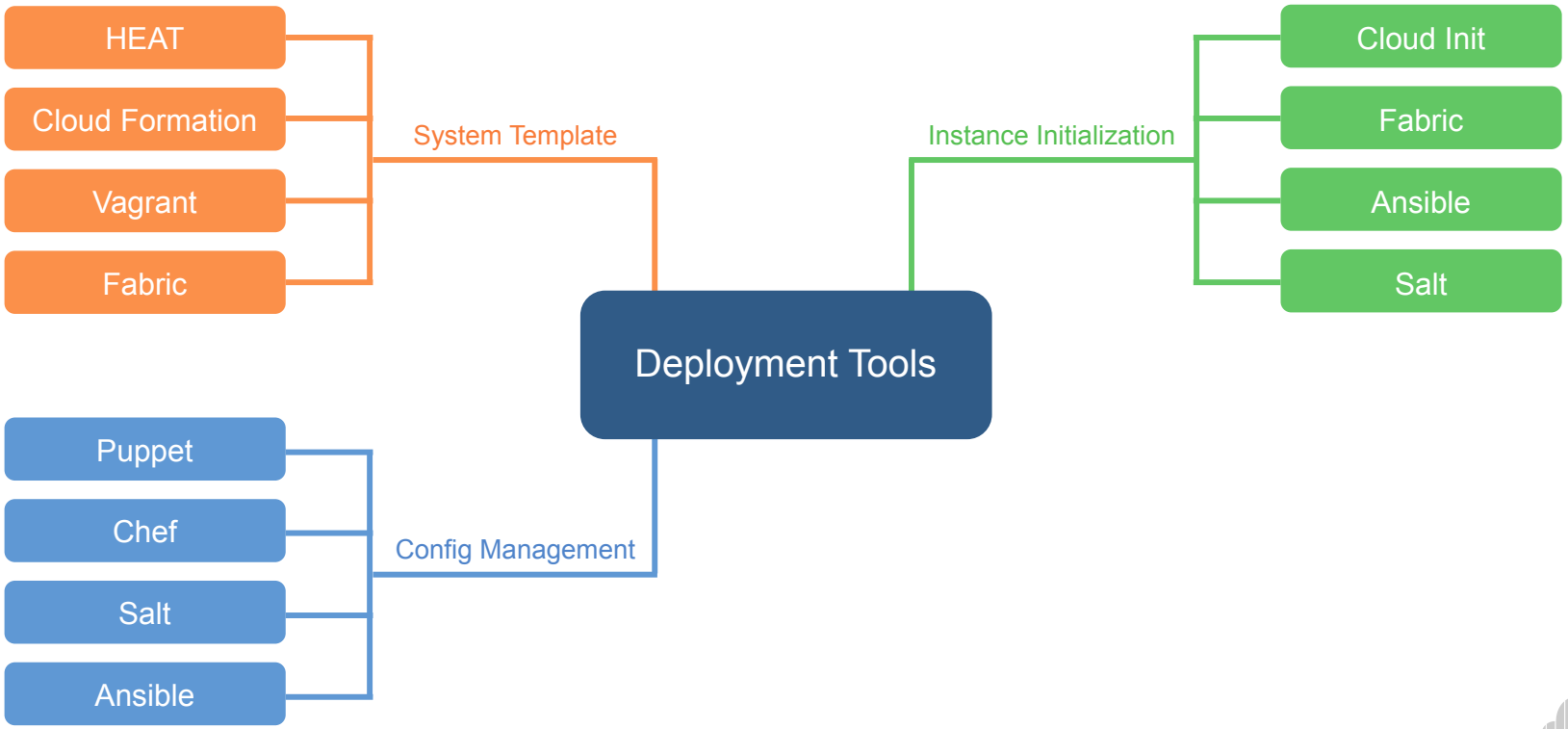
Now

- Run Script…
  - Script launches generic image
  - Script passes init script to image
  - Image boots, runs init
  - Init loads application from repository
  - Config automated via puppet/chef
  - Script requests SLB update

- Sometimes
- Ceilometer monitor adds/removes instances (based on script)

OneCloud Consulting

Tool Map



OneCloud Consulting

Step 1: Initialize

- Get basic info onto the new Instance
- SSH keys (avoid passwords)
- Perhaps update/install packages/fix DNS
- Download code (possibly)
- Download config management tools/components

OneCloud Consulting

Cloud-Init Overview

- https://cloudinit.readthedocs.org/en/latest/
- Requires software on the “golden image”
- Embedded into Nova boot process and Neutron network setup
  - Metadata web service interface
  - Config_drive mini-iso based model
  - Can pass script, yaml config, multi-part MIME message with scripts, config, etc.
- Principal Functions:
  - SSH key management (user, server)
  - User management
  - Package and Repository Management
  - Arbitrary Scripts
  - File creation

OneCloud Consulting

Sample UserData as YAML

```
#cloud-config
write_files:
  - path: /etc/resolv.conf
    permissions: 0644
    owner: root
    content: |
      nameserver 8.8.8.8

apt_update: true
packages:
 - python-pip
 - git

runcmd:
- mkdir /home/ubuntu/gitrepo
- git clone "https://github.com/mjhea0/flaskr-tdd.git" /home/ubuntu/gitrepo
- pip install -r /home/ubuntu/gitrepo/requirements.txt
- sed -i 's/app.run()/app.run(host="0.0.0.0",port=1234)/g' /home/ubuntu/gitrepo/app.py
- python /home/ubuntu/gitrepo/app.py &
```

OneCloud Consulting

Puppet

What is it?

- Puppet is a data driven, idempotent configuration management system that allows you to define the state of your infrastructure and/or application and enforce the end state of the environment.

What can it do?

- Manage packages (versions, installation, etc.)
- Manage configuration of components (add vhost to apache config)
- Report on state of configuration/ installation
- Enable a ‘workflow’ model for deployment

OneCloud Consulting

Puppet hierarchy

- Module
  - Manifest
    - Init.pp
      - Class
      - Types/defines
- Example: class to define user

```
class robert ($state_val='present') {
  user {"robert":
    ensure=>$state_val,
  } }
```

- In order to do something though, we have to instantiate the class, so we can add:

```
class {'robert': state_val=>'absent'}
```

- And then we can "execute this (assuming we put it in a file test.pp)

```
puppet apply -vd init.pp
```

- Or we could put the file in /etc/puppet/modules/robert/manifests/init.pp and:

```
puppet apply -ve 'class {"robert": }'
```

OneCloud Consulting

Masterless Puppet

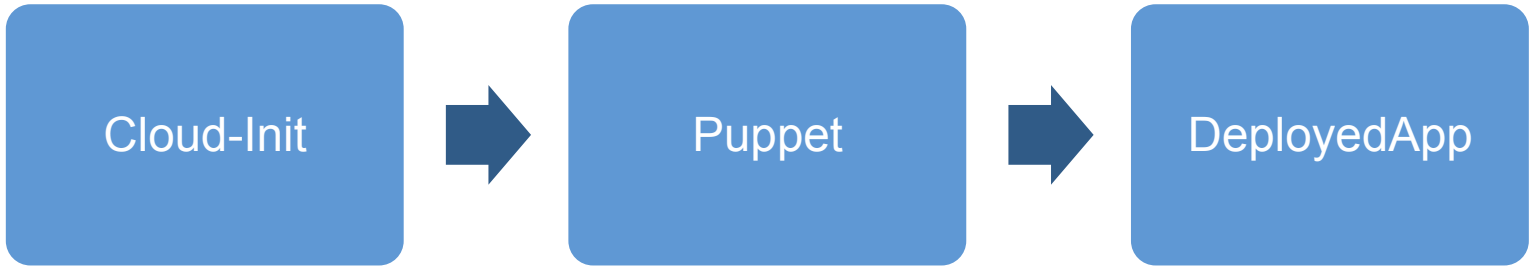
Advantages:

- Masterless puppet modules are about as simple as writing bash scripts, however they are more maintainable, easily made to be cross platform.
- Still gives you the advantage of the abstraction layer that puppet provides

Disadvantages:

- More difficult to distribute to nodes, need pulp, git or other distribution techniques

How To Tie It All Together



Use cloud-Init to install puppet

Puppet triggers puppet apply and run manifests to deploy the given app

Your deployed app on the openstack instances are ready to served.

OneCloud Consulting

Using Heat and Ceilometer

OneCloud
Consulting

Ceilometer Overview

- Initial intent was to develop a tool to support billing against consumption
- Service Metering is a long standing solution, and has well understood models
- Data collected has uses besides metering
  - Alarm (something has happened)
  - Assurance (is my service alive)
  - Scale management (do I have too much/little)
  - Profiling (is my app running efficiently)

OneCloud Consulting

Getting metrics from Ceilometer

- Principally via CLI, additional interactions are possible via HEAT ceilometer meters-list
- The list is long and includes more than you probably want to see, this is a more concise list:

Name	Type	Unit
cpu	cumulative	ns
cpu_util	gauge	%
disk.read.bytes.rate	gauge	B/s
disk.read.requests.rate	gauge	request/s
disk.write.bytes.rate	gauge	B/s
disk.write.requests.rate	gauge	request/s
network.incoming.bytes.rate	gauge	B/s
network.incoming.packets.rate	gauge	packet/s
network.outgoing.bytes.rate	gauge	B/s
network.outgoing.packets.rate	gauge	packet/s

Ceilometer Alarm Notifier

- Allows you to set alarms based on threshold evaluation for a collection of samples.
- An alarm can be set on a single meter, or on a combination
- To setup an alarm, you will call Ceilometer's API server specifying the alarm conditions and an action to take

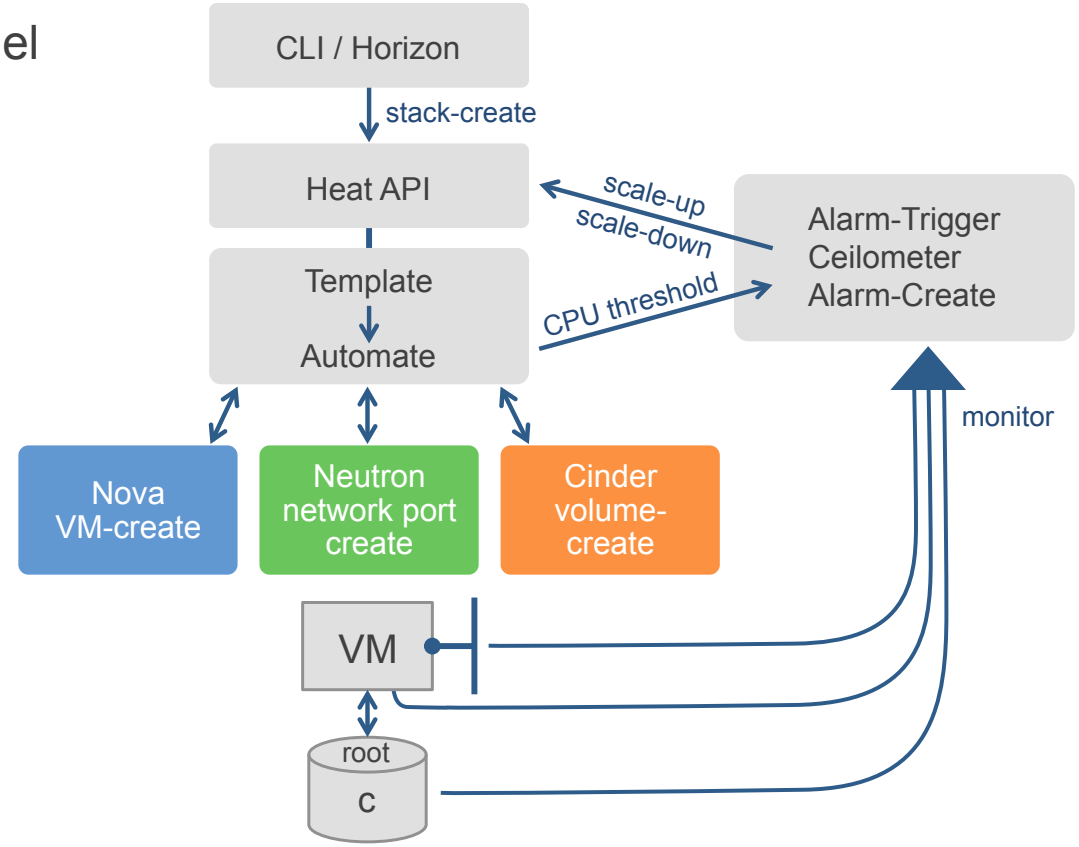
```
ceilometer alarm-threshold-create --name cpu_high --description 'instance running
hot'     --meter-name cpu_util  --threshold 70.0 --comparison-operator gt  --
statistic avg    --period 600 --evaluation-periods 3    --alarm-action 'log://'
--query resource_id=VM_ID
```

- Forms of action
  - HTTP/HTTPs Callback
  - Log

OneCloud Consulting

Heat Model

Heat Model



OneCloud
Consulting

Example HOT template

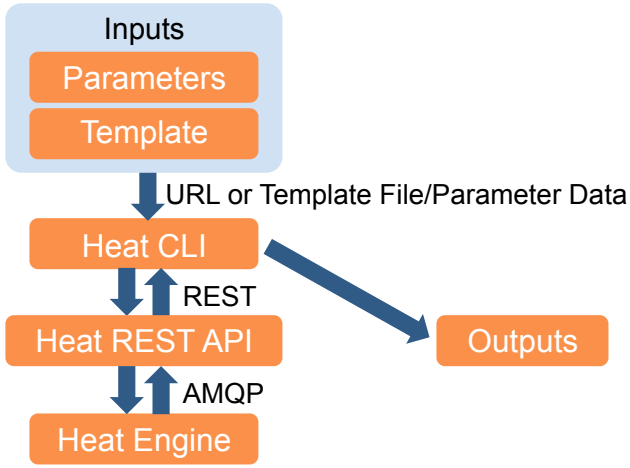
```
heat_template_version: 2013-05-23

description: Test Template
```

```
parameters:
  ImageID:
    type: string
    description: Image use to boot a server
  NetID:
    type: string
    description: Network ID for the server
```

```
resources:
  server1:
    type: OS::Nova::Server
    properties:
      name: "Test server"
      image: { get_param: ImageID }
      flavor: "m1.tiny"
      networks:
        - port: { get_resource: server1_port }
```

```
server1_port:
    type: OS::Neutron::Port
    properties:
      network_ID: { get_param: NetID }
```

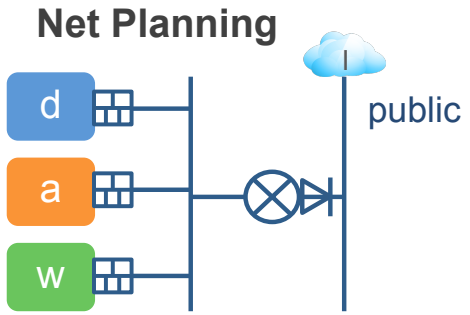
```
outputs:
  server1_private_ip:
    description: IP address of the server in the private network
    value: { get_attr: [ server1, first_address ] }
```



OneCloud Consulting

Template Model considerations

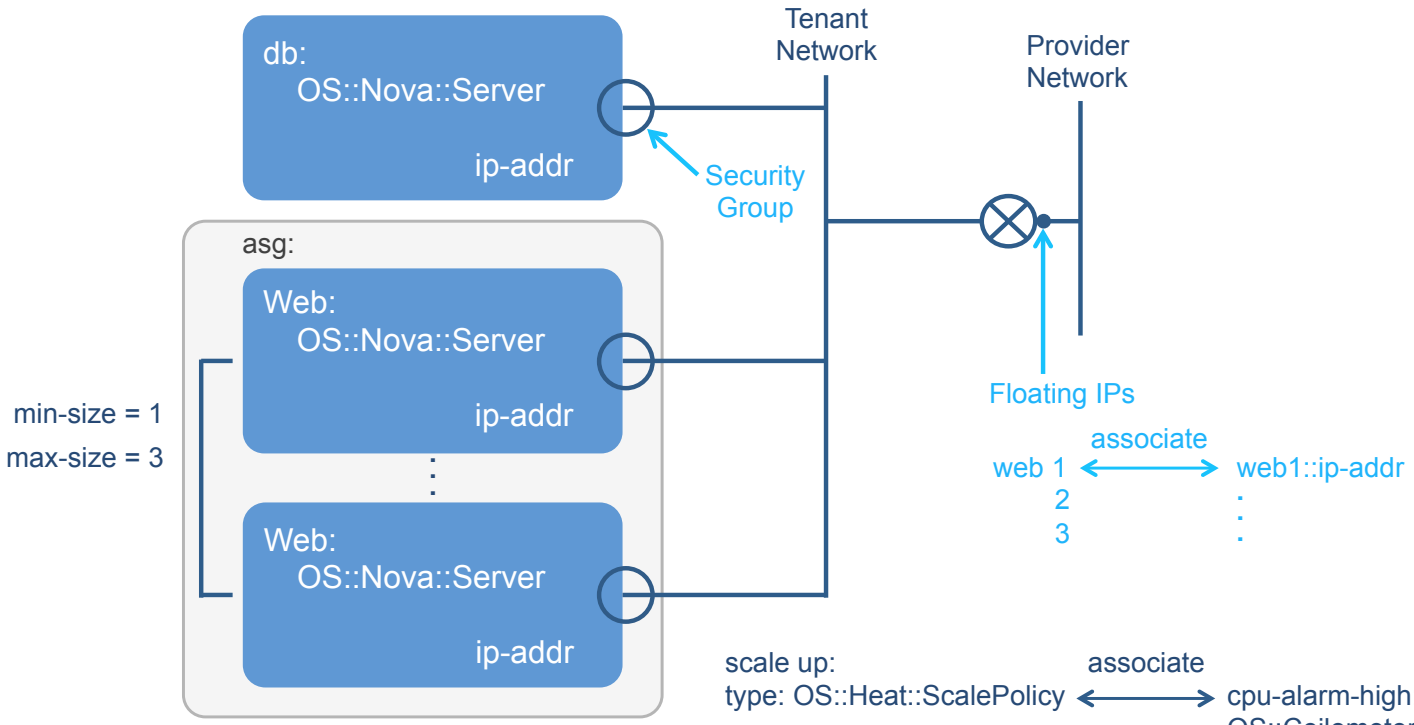
- How your network works (floating Ips, SLB, etc.) still need to be considered
- Map the resources together to determine how pieces should fit
- Individual “public” VMs need Floating IPs associated



- Interaction between VMs (scale groups)
  - Scale up and Scale down resources
  - Associated Ceilometer alarms

- With SLB
  - Define pool (associate floating IP with pool)
  - Associate scale VMs with pool

OneCloud Consulting



OneCloud
Consulting

Example Scaling HOT

- From Github: https://github.com/openstack/heat-templates/tree/master/hot
- https://raw.githubusercontent.com/openstack/heat-templates/master/hot/autoscaling.yaml
- https://raw.githubusercontent.com/openstack/heat-templates/master/hot/lb_server.yaml

OneCloud Consulting

Stack Creation

- `$ heat stack-list`

```
+--------------------------------------+--------------+-----------------+----------------------+
| id                                   | stack_name   | stack_status    | creation_time        |
+--------------------------------------+--------------+-----------------+----------------------+
| 4c712026-dcd5-4664-90b8-0915494c1332 | stackone     | CREATE_COMPLETE | 2013-08-12T13:33:47Z |
| 7edc7480-bda5-4e1c-9d5d-f567d3b6a050 | stacktwo     | CREATE_FAILED   | 2013-08-12T13:33:59Z |
+--------------------------------------+--------------+-----------------+----------------------+
```

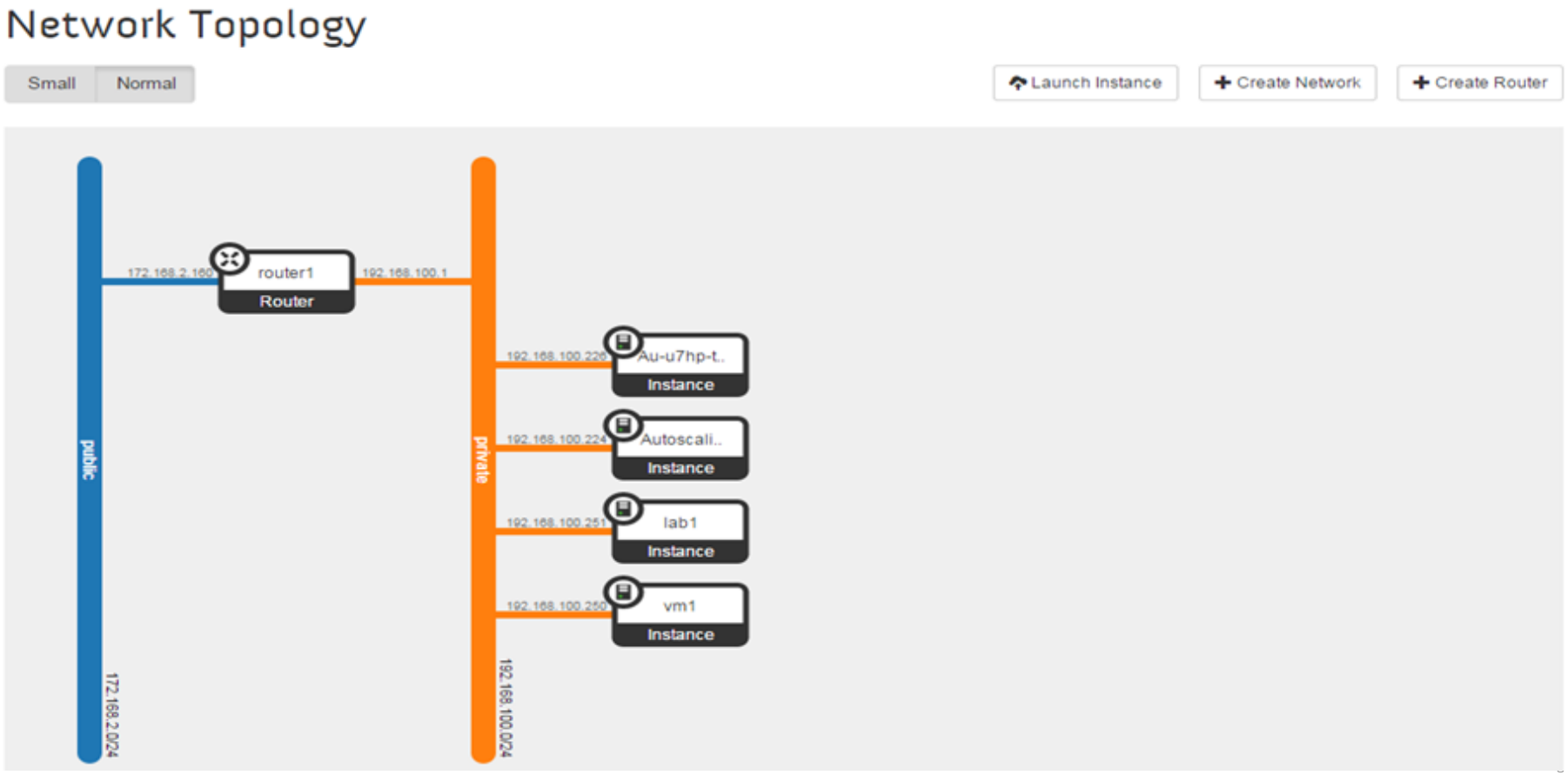
- `$ heat resource-list mystack`

```
+----------------------+--------------------+-----------------+----------------------+
| logical_resource_id | resource_type       | resource_status | updated_time         |
+----------------------+--------------------+-----------------+----------------------+
| Database             | OS::NOVA::Instance | CREATE_COMPLETE | 2013-08-12T13:34:56Z |
+----------------------+--------------------+-----------------+----------------------+
```

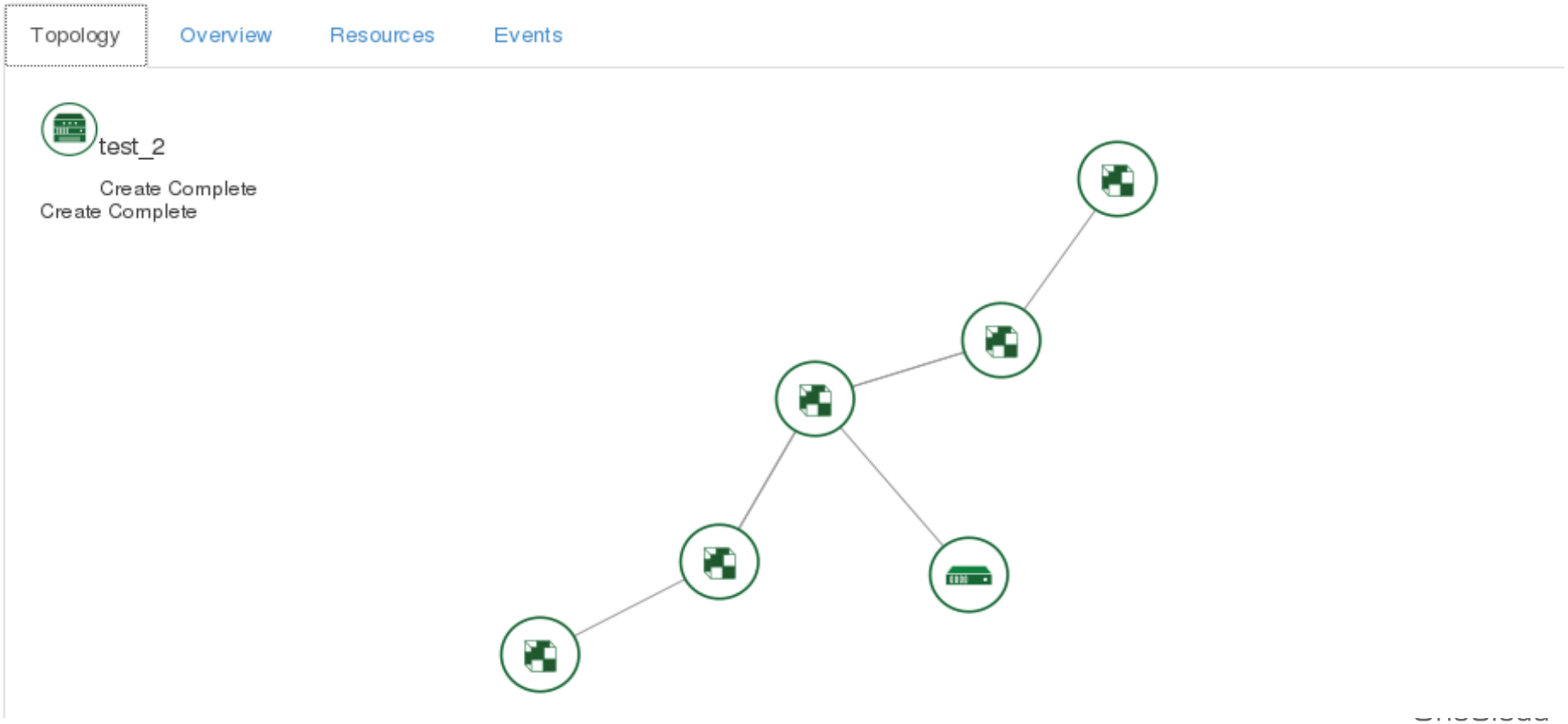
- `$ heat event-list mystack`

```
+----------------------+----+-----------------------+-----------------+----------------------+
| logical_resource_id | id | resource_status_reason | resource_status | event_time           |
+----------------------+----+-----------------------+-----------------+----------------------+
| Database             | 1  | state changed          | IN_PROGRESS     | 2013-08-12T13:42:19Z |
| Database             | 2  | state changed          | CREATE_COMPLETE | 2013-08-12T13:45:44Z |
+----------------------+----+-----------------------+-----------------+----------------------+
```

Network Topology



HEAT Topology



Scaling without Ceilometer

- Can trigger HEAT scale up/scale down with URL from scale group.

- From example heat template:

```
scale_up_url:
    value: { get_attr: [web_server_scaleup_policy, alarm_url] }

scale_dn_url:
    value: { get_attr: [web_server_scaledown_policy, alarm_url] }
```

```
heat output-list stack
heat output-show stack scale_up_url
curl -X POST "http://10.1.10.13:8000/v1/signal/arn%3Aopenstack%3Aheat%3A%3Ae71d06beb59a40d1a9e29df6b014444e
%3Astacks%2Fasg%2F8e386fdc-a872-493d-946b-c9f3e63878c3%2Fresources%2Fweb_server_scaleup_policy?
Timestamp=2015-02-19T07%3A47%3A54Z&SignatureMethod=HmacSHA256&AWSAccessKeyId=bcd012555b7949bd953598f0090bb2c
d&SignatureVersion=2&Signature=p3kbk%2F2v8zLiwUlsToobhPiHB4n8lRTfhnDsZMfbYhU%3D"
```

OneCloud Consulting

MONASCA

Monitoring as a Service - MONaaS

OneCloud
Consulting

Current Status

- Monasca and Stacktach.v3 is open-sourced in StackForge
- Not an OpenStack incubated project, but we are targeting incubation
- Metrics, Alarm Definitions, Alarms and Notification Methods completely
- supported/functional and ready for production deployment
- Who is working on it?
- • HP
- • RackSpace
- • IBM
- Who is deploying it?
- • HP: Public Cloud and Helion distribution
- • Time Warner Cable (TWC)
- • Workday

OneCloud
Consulting

Current Status- Progress

- Events/StackTach.v3 integration is in progress
- Anomaly detection is in progress
- Formalize micro-services architecture
  - Define message formats
  - Define how services are published and registered
- Python port is in progress:
  - All components Python except for API and Threshold Engine
  - API is 75% ported to Python. Note, Java API is 100% functional
  - Threshold Engine is the only remaining Java component

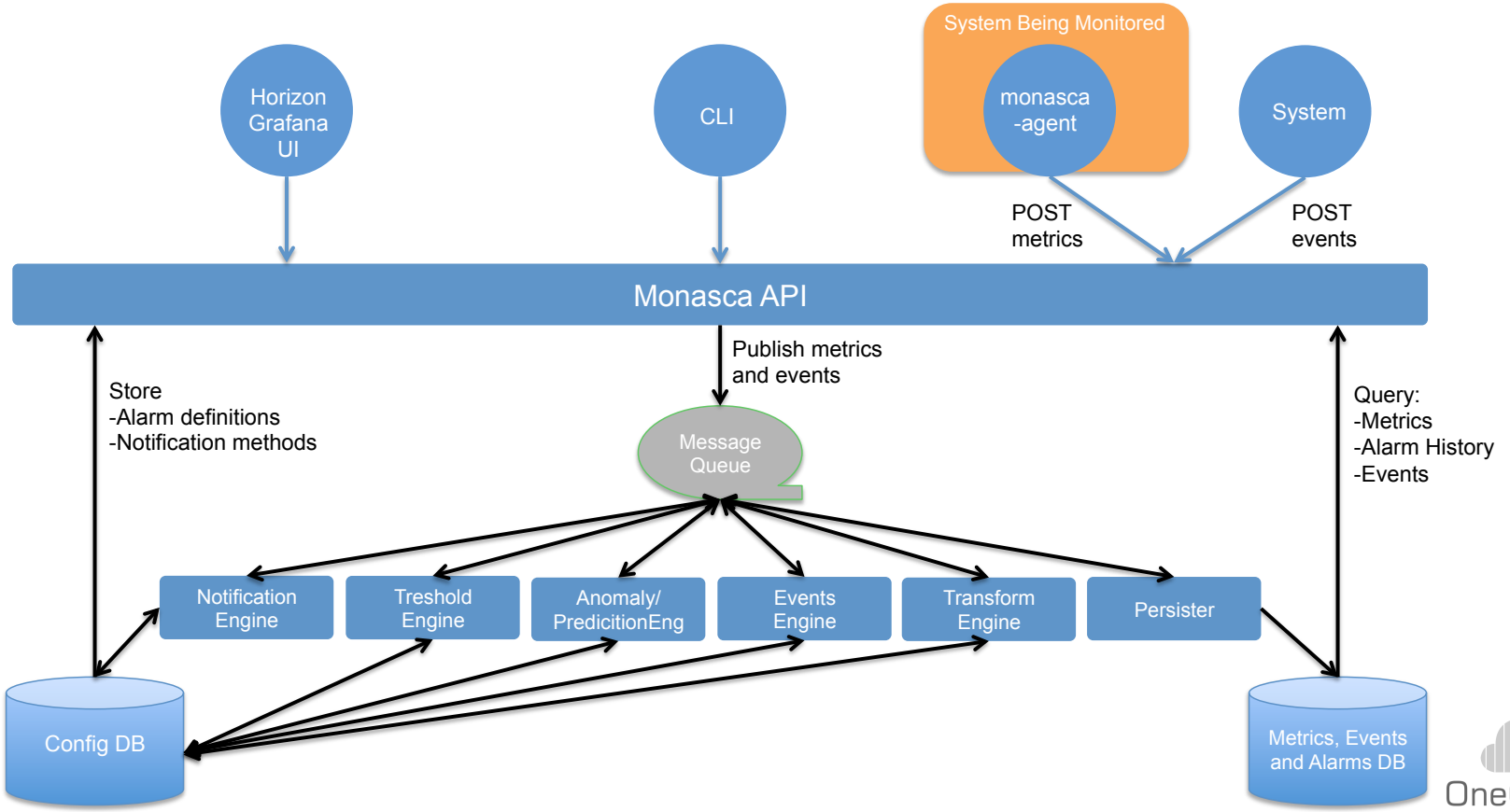
OneCloud
Consulting

What is Monasca?

- Monitoring-as-a-Service solution based on REST API
- Highly-performant, scalable, fault-tolerant and capable of big data retention
- Metrics storage/retrieval/statistics and alarm/thresholding engine
- Notification system
- Real-time event stream processing
- Open-source and built-on open-source technologies such as:
  - Kafka: Performant, scalable, fault-tolerant, durable message queue. Used by LinkedIn, Twitter, …
  - Apache Storm: distributed realtime computation system
  - Time-series databases: InfluxDB supported today. Elastic-search in progress.
  - Consolidates multiple monitoring systems into a single solution
  - Used for both operational and customer facing monitoring.
- Extensible based on micro-services message bus architecture

OneCloud Consulting

Architecture



Metrics - Rest API

- Metrics: Create, query and get statistics for metrics

```
{
     name: cpu.user_perc,
     dimensions: {
          hostname: hostname.domain.com,
          region: uswest,
          zone: 1,
          service: compute
     }
}
```

Simple, concise flexible description
Name (string)
Dimensions: Dictionary of arbitrary (key, value) pairs

- Alarm Definitions –
  - Are templates that are used to automatically create alarms based on matching metric names and dimensions
  - Simple compound expression grammar: `avg(cpu.user_perc{}) > 85 or avg(disk_read_ops{device=vda}, 120) > 1000`
  - Actions associated with alarms for state transitions to ALARM, OK and UNDETERMINED
  - Severity (LOW, MEDIUM, HIGH, CRITICAL).
- Alarms: Query and Delete alarms and query alarm state history
- Notification Methods: e.g. Email address. Associated with alarm definitions

OneCloud Consulting

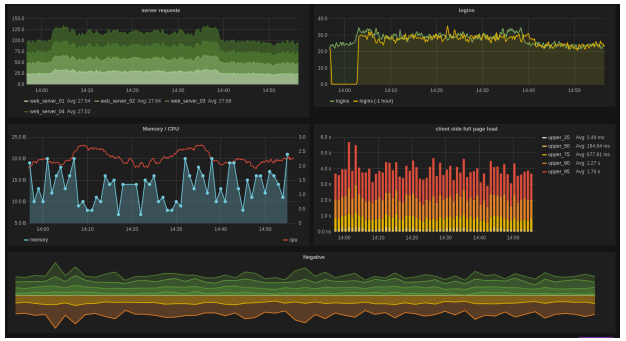
Metrics – monasca-agent

- Python monitoring agent
- System metrics (cpu, memory, …)
- Service metrics (RabbitMQ, MySQL, Kafka, and many other
- Application metrics (Built-in statsd daemon and Python Monasca Statsd library)
- VM metrics
- Active checks
  - HTTP status checks and response times
  - System up/down checks (ping and ssh)
- Runs any Nagios plugin
- Extensible/Pluggable: Additional services can be easily added

Metrics - UI

- Horizon Dashboard
  - Overview/Top-level drill-down
  - Create/Read/Update/Delete alarm definitions using an expression builder
  - Read/Delete alarms and alarm history
  - Create/Read/Update/Delete notification methods

- Grafana Dashboard (http://grafana.org/)
  - Provides visualization of metrics

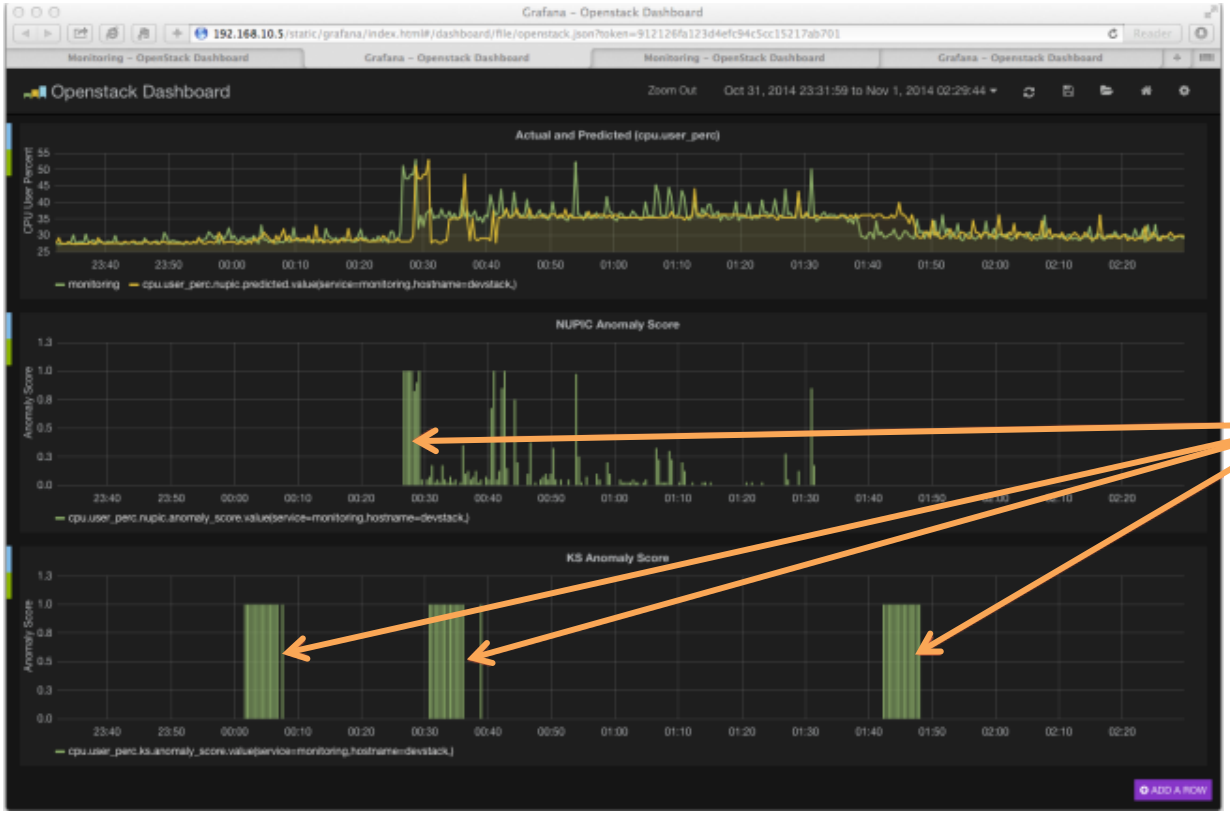


OneCloud Consulting

Events – Anomaly Detection

- Monasca Anomaly Engine implements real-time streaming anomaly detection
- Two algorithms used:
  - Numenta Platform for Intelligent Computing (NuPIC) used by Grok
    (An open-source Python/C++ implementation of Hierarchical Temporal Memory)
  - Kolmogorov-Smirnov (K-S) Two Sample Test
- Anomaly Engine
  - Consumes metrics from the Kafka metrics topic
  - Calculates predicted value and anomaly score (probability of an anomaly)
  - Publishes calculated values as metrics to the Kafka metrics topic
- Alarms can be created for Anomaly scores

Events – Anomaly Detection



OneCloud Consulting