TRIPLEO

PROVISION YOUR DATACENTER WITH OPENSTACK

Imre Farkas and Ladislav Smola

AGENDA

- TripleO tools
- How to deploy OpenStack?
- Architecture
- Tuskar & Tuskar-UI
- Advanced features

WHAT IS TRIPLEO?

“TripleO is a program aimed at installing, upgrading and operating OpenStack clouds using OpenStack's own cloud facilities as the foundations - building on Nova, Neutron, Heat and Ironic to automate fleet management at datacenter scale (and scaling down to as few as 2 machines).”

HOW TO LAUNCH AN APPLICATION?

1. grab an image
2. deploy
3. configure

GRAB AN IMAGE

```
wget http://download.fedoraproject.org/pub/fedora/linux/releases/20/
            Images/x86_64/Fedora-x86_64-20-20131211.1-sda.qcow2
```

HOW TO LAUNCH AN APPLICATION?

1. ~~grab an image~~
2. **deploy**
3. configure

HOW TO DEPLOY A VM? - SOLUTION #1

```
nova boot myCentOSServer
          --image "3afe97b2-26dc-49c5-a2cc-a2fc8d80c001"
          --flavor m1.small
```

Use Puppet, Chef, Ansible or whatever for configuration

HOW TO DEPLOY A VM? - SOLUTION #2

Use Heat!

HEAT TEMPLATE

```
heat_template_version: 2013-05-23

description:
  ...

parameters:
  ...

resources:
  ...

outputs:
  ...
```

HEAT TEMPLATE - PARAMETERS

```
parameters:
  db_name:
    type: string
    description: WordPress database name
    default: wordpress
    constraints:
      - length: { min: 1, max: 64 }
        description: db_name must be between 1 and 64 characters
      - allowed_pattern: '[a-zA-Z][a-zA-Z0-9]*'
        description: >
          db_name must begin with a letter and contain only alphanumeric
          characters
```

HEAT TEMPLATE - RESOURCES

```
resources:
  wordpress_instance:
    type: OS::Nova::Server
    properties:
      image: { get_param: image_id }
      flavor: { get_param: instance_type }
      key_name: { get_param: key_name }
      user_data:
        ...
```

HEAT TEMPLATE - RESOURCE PROPERTIES: USER_DATA (1)

```
user_data:
  str_replace:
    template: |
      #!/bin/bash -v

      yum -y install mysql mysql-server httpd wordpress
      systemctl enable mysqld.service
      systemctl enable httpd.service
      systemctl start mysqld.service
      systemctl start httpd.service

      firewall-cmd --add-service=http
      firewall-cmd --permanent --add-service=http
```

HEAT TEMPLATE - OUTPUTS

```
outputs:
  WebsiteURL:
    description: URL for Wordpress wiki
    value:
      str_replace:
        template: http://host/wordpress
        params:
          host: { get_attr: [wordpress_instance, first_address] }
```

HOW TO DEPLOY AN VM? - SOLUTION #2

```
heat stack-create mystack
                  --template-file=WordPress_Single_Instance.yaml
                  --parameters="db_name=$db_name;db_rootpassword=..."
```

TRIPLEO-HEAT-TEMPLATES

HOW TO LAUNCH AN APPLICATION?

1. **~~grab~~/build an image**
2. ~~deploy~~
3. configure

IMAGE BUILDING

THE GOAL OF IMAGE BUILDING IN TRIPLEO

"The goal of the image building process is to produce blank slate machines that have all the necessary bits to fulfill a specific purpose in the running of an Openstack cloud: e.g. a nova-compute node."

DISKIMAGE-BUILDER

- disk images / file system images / ramdisk images
- virtual / baremetal machines
- stock of elements
- easily extensible

DISKIMAGE-BUILDER

SCRIPTS EXECUTION PHASES

- root.d
- finalise.d
- cleanup.d
- block-device.d
- extra-data.d
- pre-install.d
- install.d
- post-install.d
- environment.d

ELEMENT EXAMPLE: FEDORA

```
bin/install-packages
bin/map-packages
bin/map-services
finalise.d/01-clean-old-kernels.sh
finalise.d/99-setup-first-boot
install.d/00-fedora-fixup-audit
install.d/00-fedora-fixup-openssl
install.d/00-fedora-fixup-pyopenssl
install.d/01-install-deps
pre-install.d/15-fedora-remove-grub
pre-install.d/00-usr-local-bin-secure-path
pre-install.d/02-lsb
root.d/10-fedora-cloud-image
README.md
element-deps
source-repository-fedora
```

HOW TO LAUNCH AN APPLICATION?

1. ~~grab/build an image~~
2. ~~deploy~~
3. **configure**

CONFIGURATION

OS-COLLECT-CONFIG

```
[default]
command=os-refresh-config

[cfn]
metadata_url=http://192.0.2.99:8000/v1/
access_key_id = ABCDEFGHIJLMNOP01234567890
secret_access_key = 01234567890ABCDEFGHIJKLMNOP
path = MyResource
stack_name = my.stack
```

HEAT TEMPLATE - METADATA

```
resources:
  wordpress_instance:
    type: OS::Nova::Server
    properties:
      image: { get_param: image_id }
        ...
      metadata:
        key: value
```

OS-REFRESH-CONFIG

SCRIPTS EXECUTION PHASES

- pre-configure.d
- configure.d
- migration.d
- post-configure.d

OS-APPLY-CONFIG

converts JSON file to service config

```
{"keystone": {"database": {"host": "127.0.0.1",
                           "user": "keystone",
                           "password": "foobar"}}}
```

```
[sql]
connection = mysql://keystone:foobar@127.0.0.1/keystone
```

MUSTACHE TEMPLATE EXAMPLE

```
[database]
connection={{tuskar.db}}

[heat_keystone]
username = {{tuskar.user}}
tenant_name = {{tuskar.tenant_name}}
password = {{tuskar.password}}
auth_url = http://{{keystone.host}}:35357/v2.0
```

TRIPLEO-IMAGE-ELEMENTS

DISKIMAGE-BUILDER/ELEMENTS VS TRIPLEO-IMAGE-ELEMENTS

ELEMENT EXAMPLE: NOVA

```
install.d/nova-source-install/74-nova
os-apply-config/etc/nova/nova.conf
os-refresh-config/configure.d/10-nova-state
pre-install.d/00-disable-requiretty
README.md
element-deps
source-repository-nova
```

VM VS BAREMETAL

- pxe boot
- autodiscovery
- hw specs
- perf metrics

IRONIC

TRIPLEO == OPENSTACK ON OPENSTACK

IN REALITY: TRIPLEO == OPENSTACK ON OPENSTACK ON OPENSTACK

ARCHITECTURE

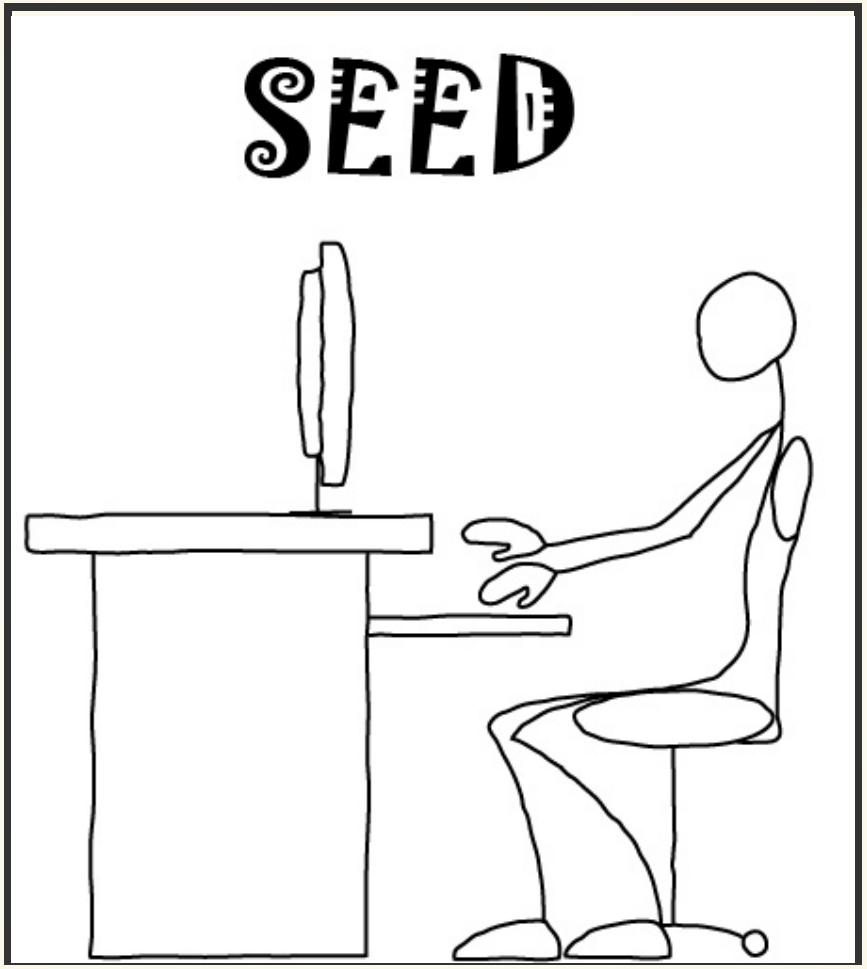
OVERCLOUD

UNDERCLOUD

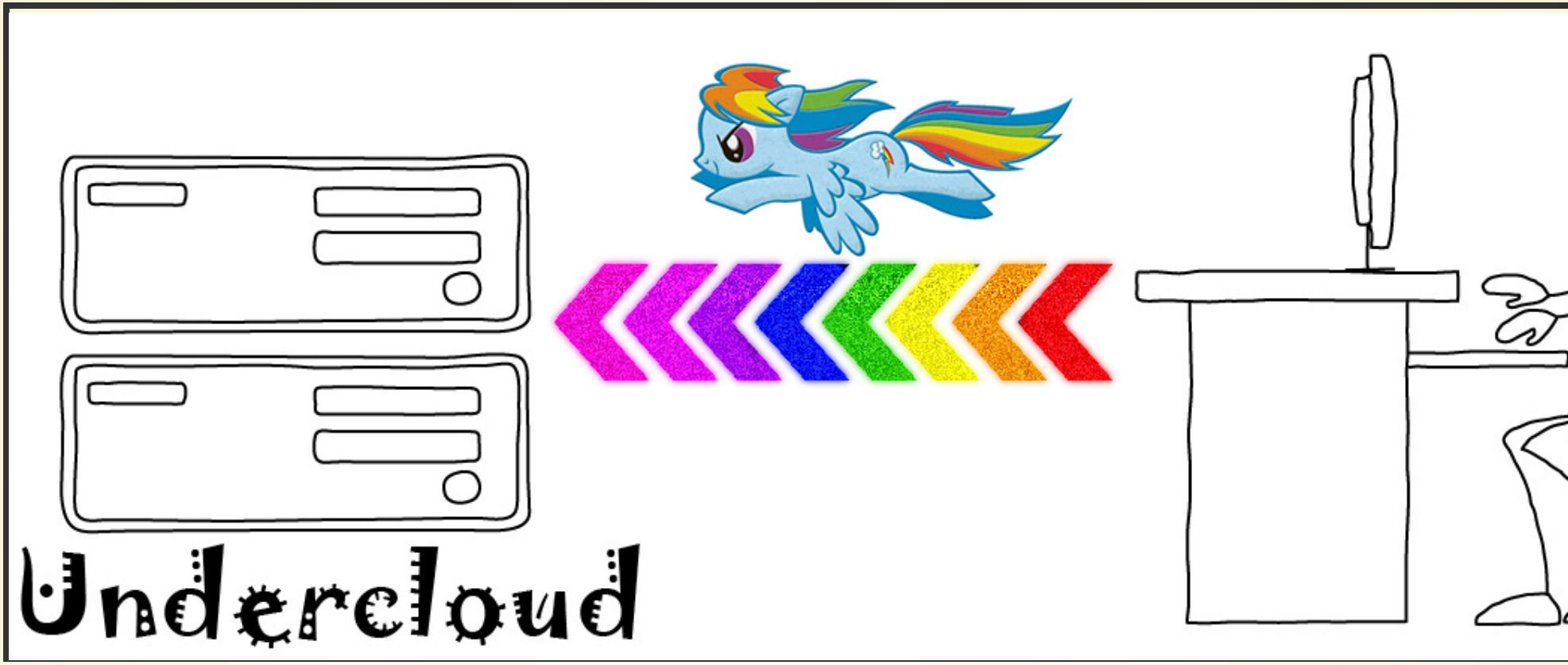
SEED

WORKFLOW

SEED



DEPLOY UNDERCLOUD



DEPLOY OVERCLOUD

DEVTEST

SEED - SETUP VM

```
setup-seed-vm -a $NODE_ARCH

$TRIPLEO_ROOT/diskimage-builder/bin/ramdisk-image-create \
    -a $NODE_ARCH $NODE_DIST $DEPLOY_IMAGE_ELEMENT \
    -o $TRIPLEO_ROOT/deploy-ramdisk

boot-seed-vm -a $NODE_ARCH $NODE_DIST neutron-dhcp-agent
```

SEED - REGISTER SERVICES

```
init-keystone -p unset unset 192.0.2.1 admin@example.com root@192.0.2.1

setup-endpoints 192.0.2.1 --glance-password unset
                          --heat-password unset
                          --neutron-password unset
                          --nova-password unset
```

UNDERCLOUD - CREATE IMAGE

```
$TRIPLEO_ROOT/diskimage-builder/bin/disk-image-create $NODE_DIST
    -a $NODE_ARCH -o $TRIPLEO_ROOT/undercloud
    boot-stack nova-baremetal os-collect-config dhcp-all-interfaces
    neutron-dhcp-agent

UNDERCLOUD_ID=$(load-image $TRIPLEO_ROOT/undercloud.qcow2)
```

UNDERCLOUD - DEPLOY WITH HEAT

```
make -C $TRIPLEO_ROOT/tripleo-heat-templates undercloud-vm.yaml

heat stack-create
    -f $TRIPLEO_ROOT/tripleo-heat-templates/undercloud-vm.yaml
    -P "PowerUserName=$(whoami);AdminToken=${UNDERCLOUD_ADMIN_TOKEN};
        AdminPassword=${UNDERCLOUD_ADMIN_PASSWORD};
        GlancePassword=${UNDERCLOUD_GLANCE_PASSWORD};
        HeatPassword=${UNDERCLOUD_HEAT_PASSWORD};
        NeutronPassword=${UNDERCLOUD_NEUTRON_PASSWORD};
        NovaPassword=${UNDERCLOUD_NOVA_PASSWORD};
        BaremetalArch=${NODE_ARCH};
        PowerManager=$POWER_MANAGER;
        undercloudImage=${UNDERCLOUD_ID}"
    undercloud
```

UNDERCLOUD - REGISTER SERVICES

```
init-keystone -p $UNDERCLOUD_ADMIN_PASSWORD $UNDERCLOUD_ADMIN_TOKEN
    $UNDERCLOUD_IP admin@example.com heat-admin@$UNDERCLOUD_IP

setup-endpoints $UNDERCLOUD_IP
                --glance-password $UNDERCLOUD_GLANCE_PASSWORD
                --heat-password $UNDERCLOUD_HEAT_PASSWORD
                --neutron-password $UNDERCLOUD_NEUTRON_PASSWORD
                --nova-password $UNDERCLOUD_NOVA_PASSWORD
```

OVERCLOUD - CREATE THE CONTROL IMAGE

```
$TRIPLEO_ROOT/diskimage-builder/bin/disk-image-create $NODE_DIST
    -a $NODE_ARCH -o $TRIPLEO_ROOT/overcloud-control
    boot-stack cinder-api cinder-volume os-collect-config
    neutron-network-node dhcp-all-interfaces swift-proxy swift-storage

OVERCLOUD_CONTROL_ID=$(load-image -d $TRIPLEO_ROOT/overcloud-control.qcow
```

OVERCLOUD - CREATE THE COMPUTE IMAGE

```
$TRIPLEO_ROOT/diskimage-builder/bin/disk-image-create $NODE_DIST \
    -a $NODE_ARCH -o $TRIPLEO_ROOT/overcloud-compute \
    nova-compute nova-kvm neutron-openvswitch-agent os-collect-config \
    dhcp-all-interfaces

OVERCLOUD_COMPUTE_ID=$(load-image -d $TRIPLEO_ROOT/overcloud-compute.qcov
```

UNDERCLOUD - DEPLOY WITH HEAT

```
make -C $TRIPLEO_ROOT/tripleo-heat-templates overcloud.yaml

heat stack-create -f $TRIPLEO_ROOT/tripleo-heat-templates/overcloud.yaml
    -P "AdminToken=${OVERCLOUD_ADMIN_TOKEN};
        AdminPassword=${OVERCLOUD_ADMIN_PASSWORD};
        CinderPassword=${OVERCLOUD_CINDER_PASSWORD};
        GlancePassword=${OVERCLOUD_GLANCE_PASSWORD};
        HeatPassword=${OVERCLOUD_HEAT_PASSWORD};
        NeutronPassword=${OVERCLOUD_NEUTRON_PASSWORD};
        NovaPassword=${OVERCLOUD_NOVA_PASSWORD};
        NeutronPublicInterface=${NeutronPublicInterface};
        SwiftPassword=${OVERCLOUD_SWIFT_PASSWORD};
        SwiftHashSuffix=${OVERCLOUD_SWIFT_HASH}${OVERCLOUD_LIBVIRT_TYPE};
        SSLCertificate=${OVERCLOUD_SSL_CERT};SSLKey=${OVERCLOUD_SSL_KEY}'
    overcloud
```

UNDERCLOUD - REGISTER SERVICES

```
init-keystone -p $OVERCLOUD_ADMIN_PASSWORD $OVERCLOUD_ADMIN_TOKEN
    $OVERCLOUD_IP admin@example.com heat-admin@$OVERCLOUD_IP
    ${SSLBASE:+--ssl $PUBLIC_API_URL}

setup-endpoints $OVERCLOUD_IP --cinder-password $OVERCLOUD_CINDER_PASSWOR
    --glance-password $OVERCLOUD_GLANCE_PASSWORD
    --heat-password $OVERCLOUD_HEAT_PASSWORD
    --neutron-password $OVERCLOUD_NEUTRON_PASSWORD
    --nova-password $OVERCLOUD_NOVA_PASSWORD
    --swift-password $OVERCLOUD_SWIFT_PASSWORD
    ${SSLBASE:+--ssl $PUBLIC_API_URL}
```

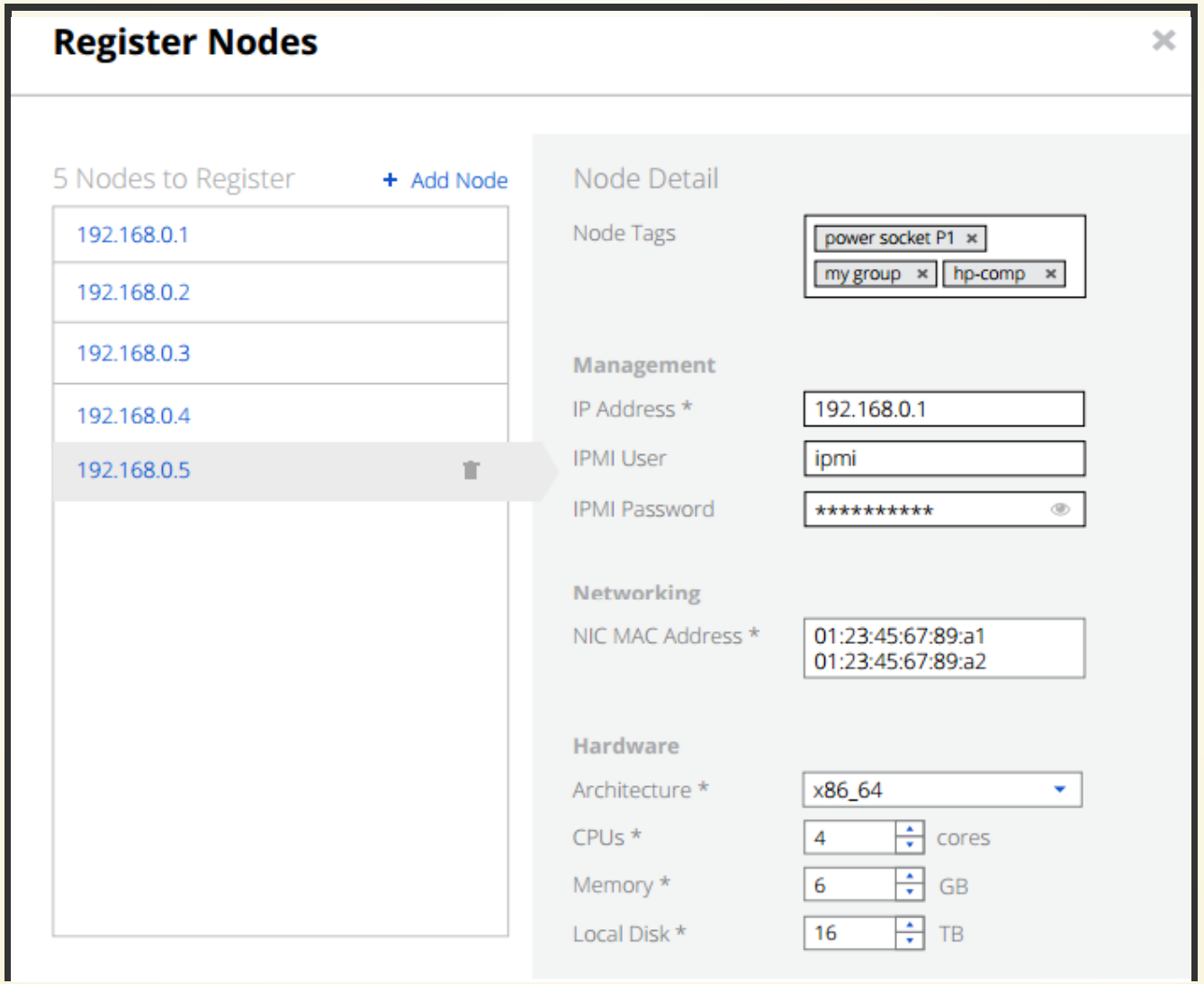
CI/CD PIPELINE

UPDATES


SO FAR...
TripleO tools
How to deploy OpenStack?
Architecture

OVERCLOUD DEPLOYMENT WITH TUSKAR

REGISTERING OF BAREMETAL NODES(MANUAL)



HARDWARE PROFILES (FLAVORS)

- Standard openstack flavors with architecture field (i386/amd64) added
- Heat uses nova-boot for provisioning, nova scheduler looks for available hardware by doing exact match of node specs with chosen flavor

IMAGES

- Created by image builder and uploaded to Glance.

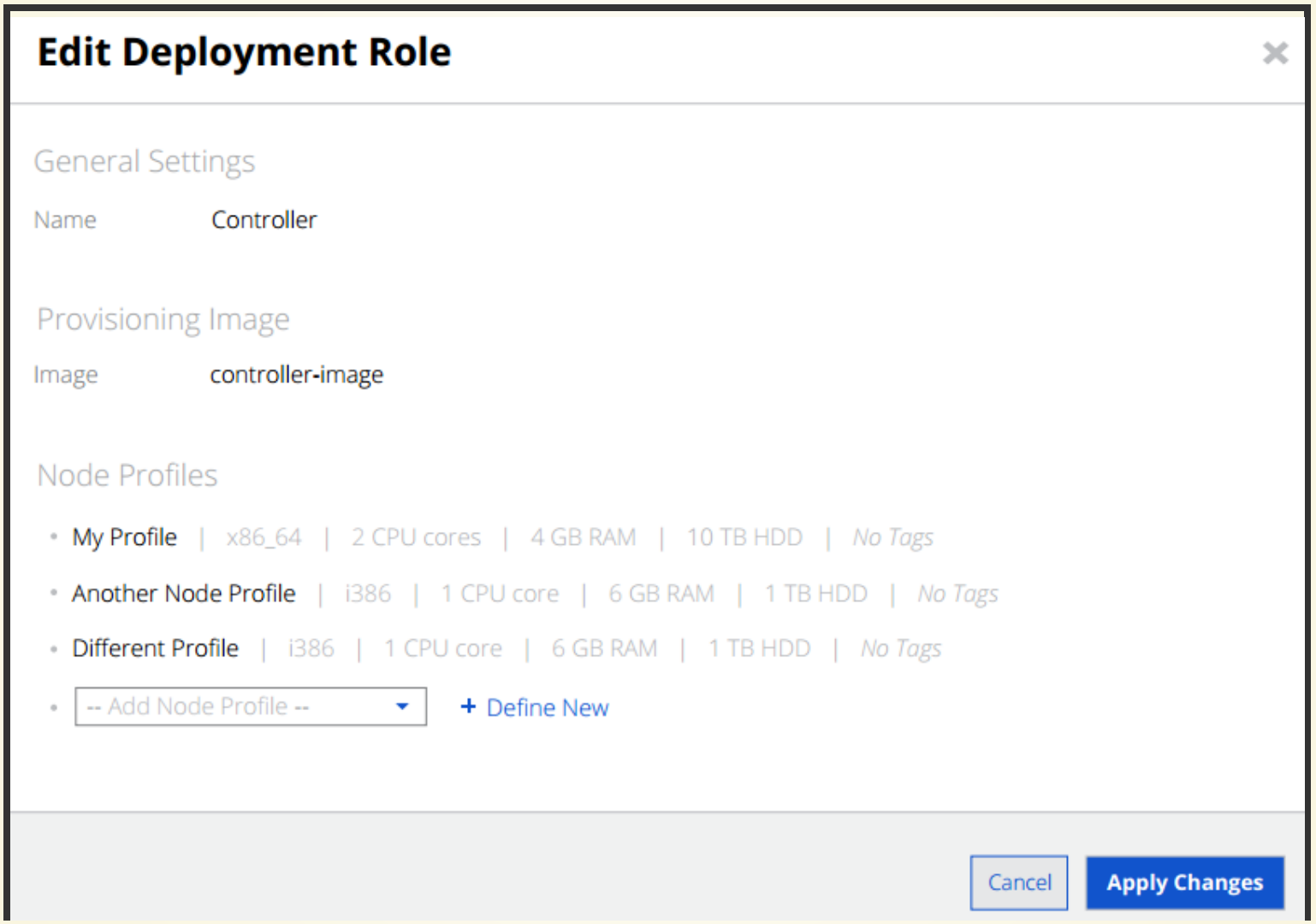
DEPLOYMENT ROLE

- Creates a group of baremetals running the same group of services. (E.g. nova compute, block storage, etc.)
- Relation to hardware profiles(flavors)
- Relation to image
- Relation to heat template

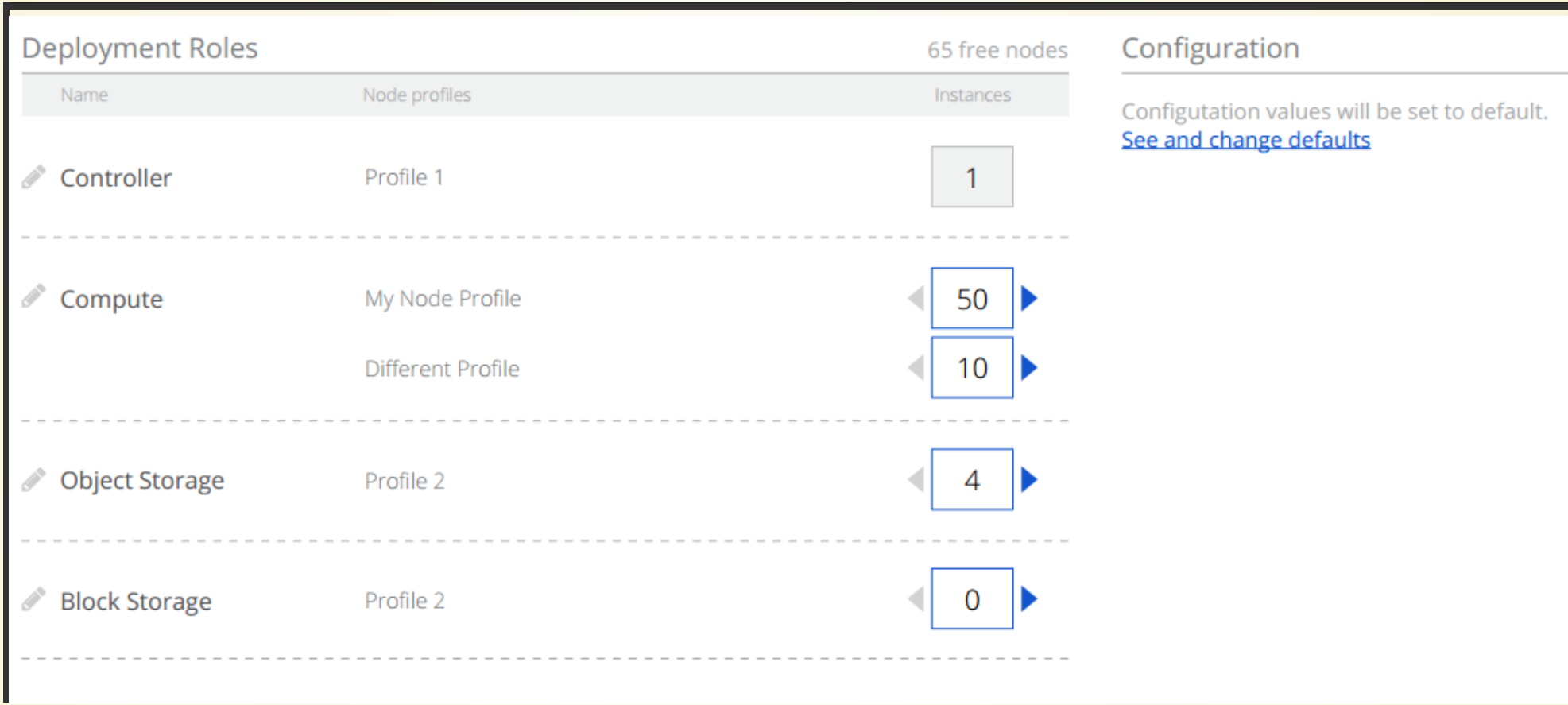
LIMITED FOR THESE ROLES FOR ICEHOUSE

- Controller (all services control planes)
- Compute (Nova)
- Object storage (Swift)
- Block storage (Cinder)

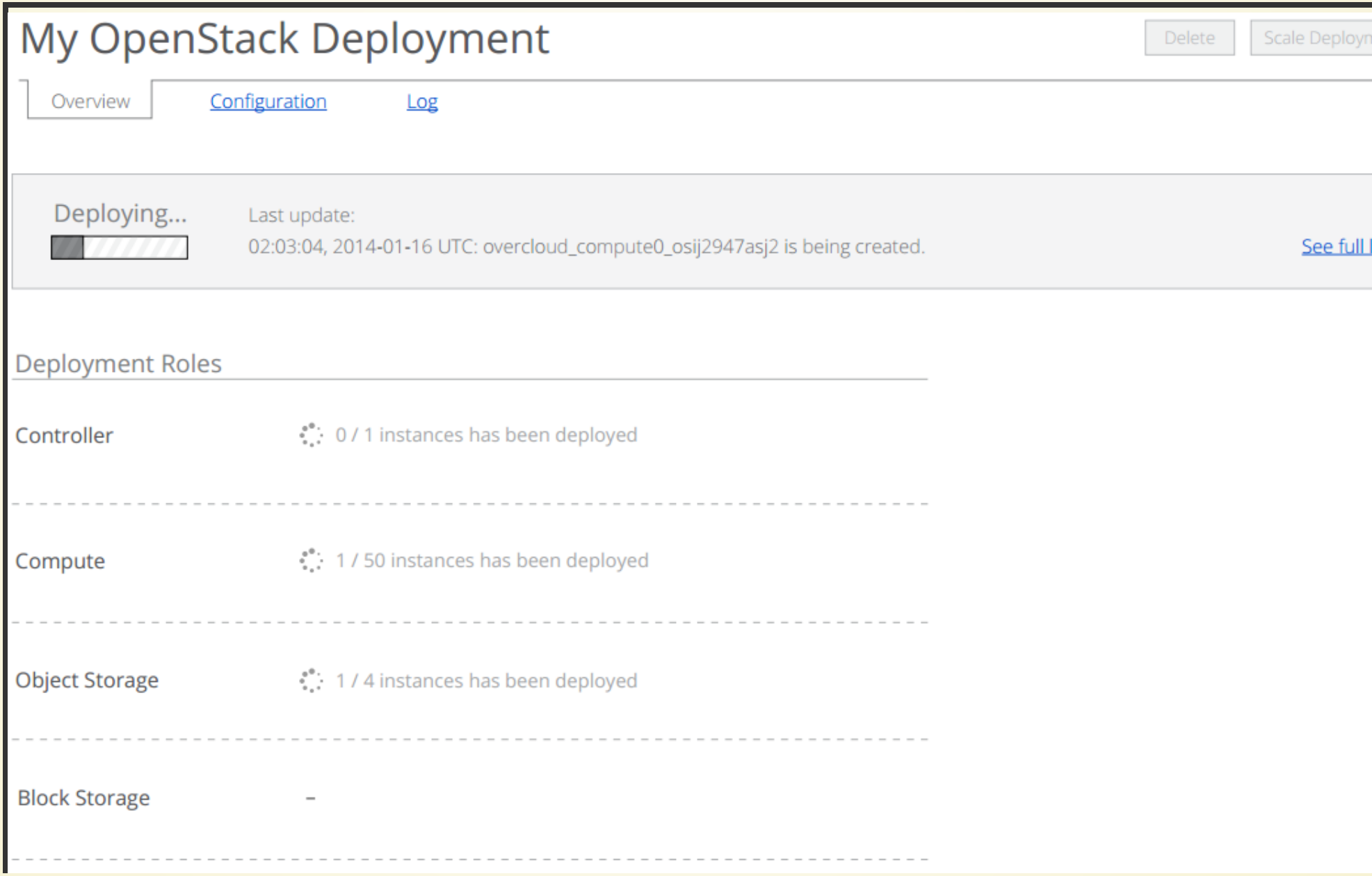
DEPLOYMENT ROLE EXAMPLE



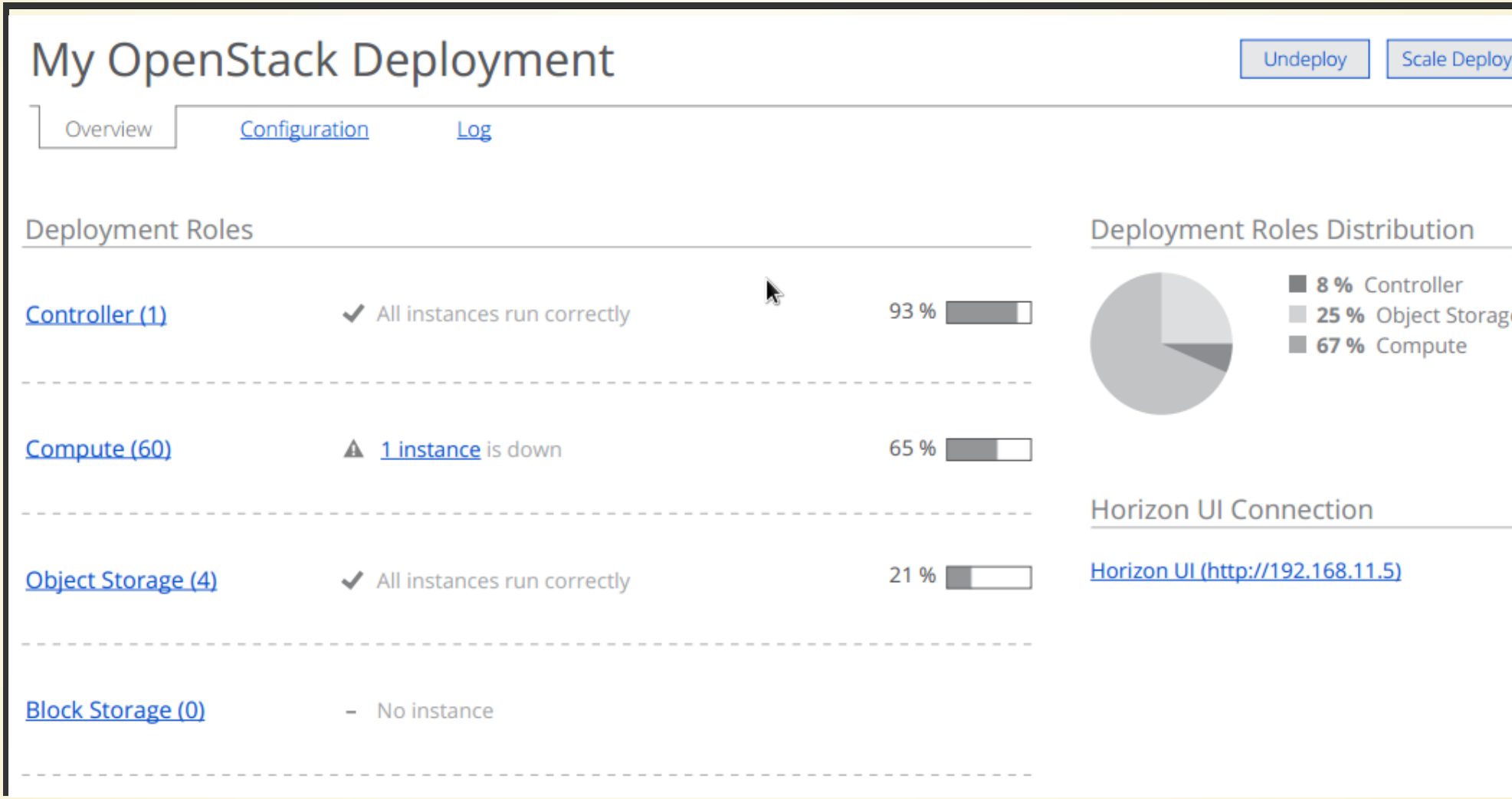
DEPLOYING OVERCLOUD EXAMPLE



DEPLOYING OVERCLOUD EXAMPLE



DEPLOYING OVERCLOUD EXAMPLE



FUTURE (JUNO) MANAGEMENT OF DEPLOYMENT ROLES

- Assigning Image
- Assigning Template
- Assigning Hardware profiles (flavors)

Then we are prepared to deploy any service on the separate baremetals, that can be easily scaled, e.g. Neutron, Ceilometer, Ironic, etc....

In future we will allow to provision multiple images on one machine with Docker.

FUTURE POSSIBLE NEEDED SERVICES

- Heat Template repository as Openstack service?
- Image builder as a Openstack service?

OVERCLOUD MONITORING WITH TUSKAR

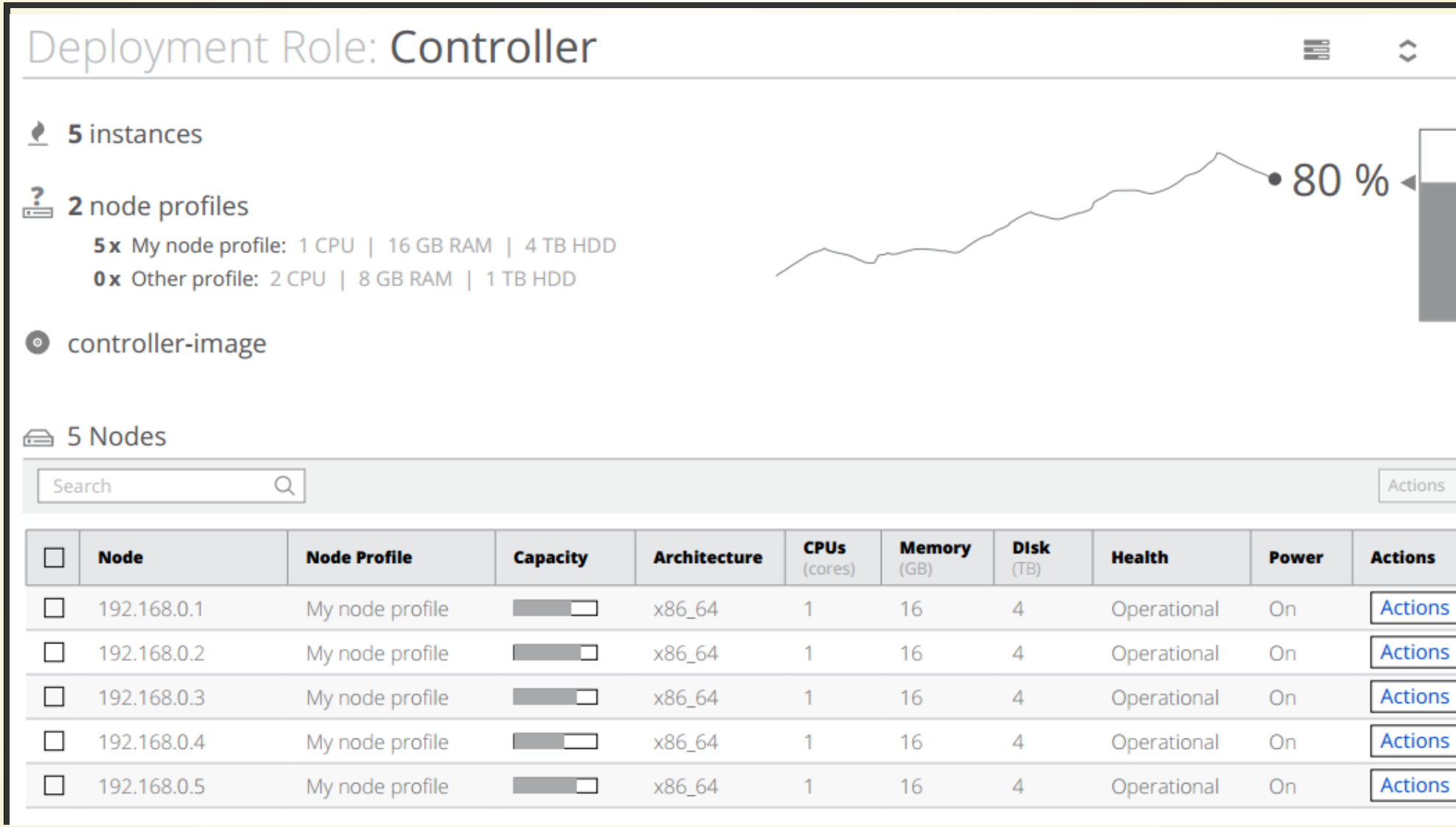
MONITORING OF BAREMETALS

- Icehouse: monitoring via SNMP: cpu_util, memory, disk, network
- Juno: monitoring via IPMI: Temperature, Fan Speed, Volt, etc. Plus pluggable architecture for vendor specific metrics.

STORING AND QUERYING STATS

- Using Ceilometer agent for polling of the samples
- Using Ceilometer as generic samples and meters storage
- Using Ceilometer for querying statistics
- Ceilometer is supporting many backends MongoDB, MySQL, PostgreSQL, HBase, DB2 (Elastic search in progress)

USING D3 AND RICKSHAW LIBRARIES FOR RENDERING CHARTS



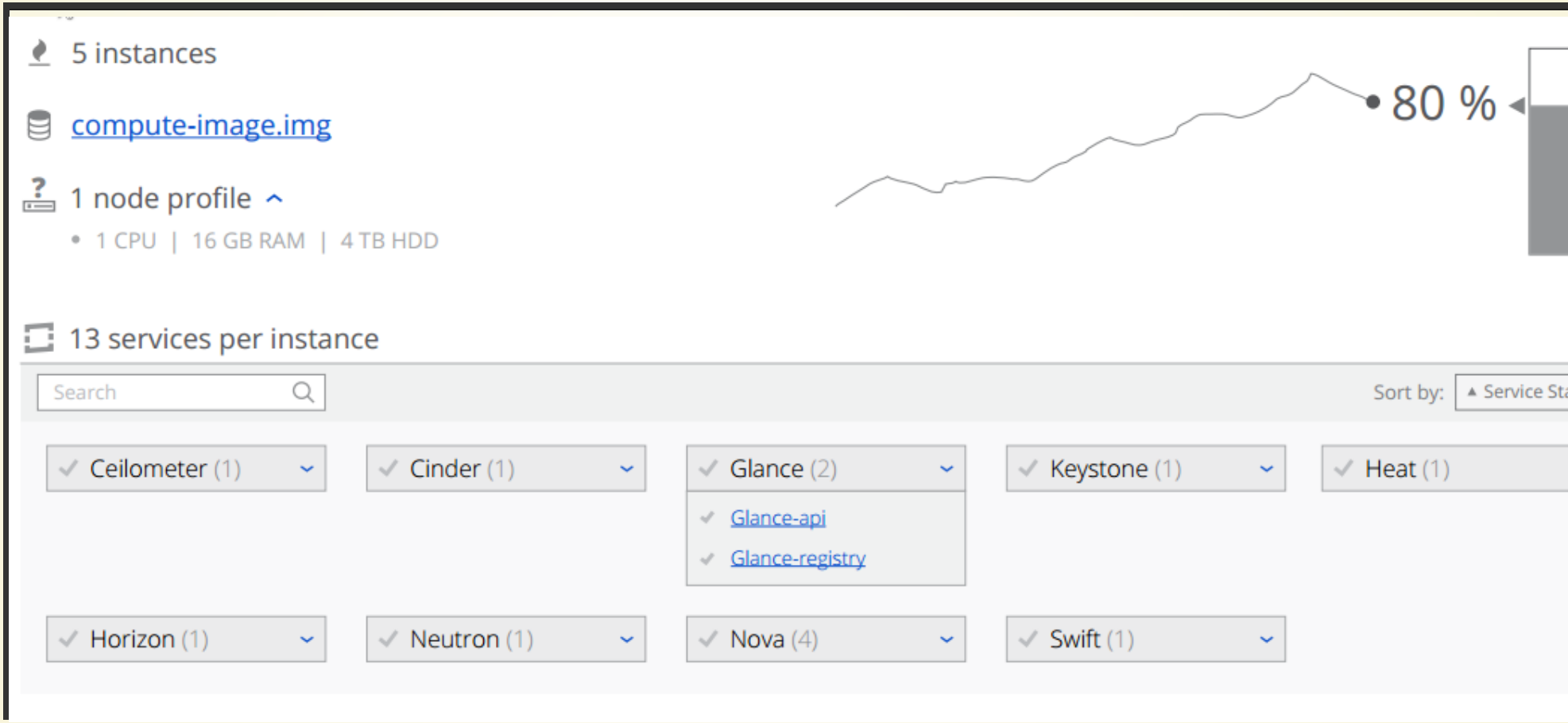
FUTURE (JUNO) ADDING 'EVENTS TIME-LINE' TO CHARTS

- Under the each chart, there will be a time-line of the events, as a context for the chart data.
- E.g. machine updated, powered down, bad health, etc..

FUTURE (JUNO) MONITORING OF RUNNING SERVICES

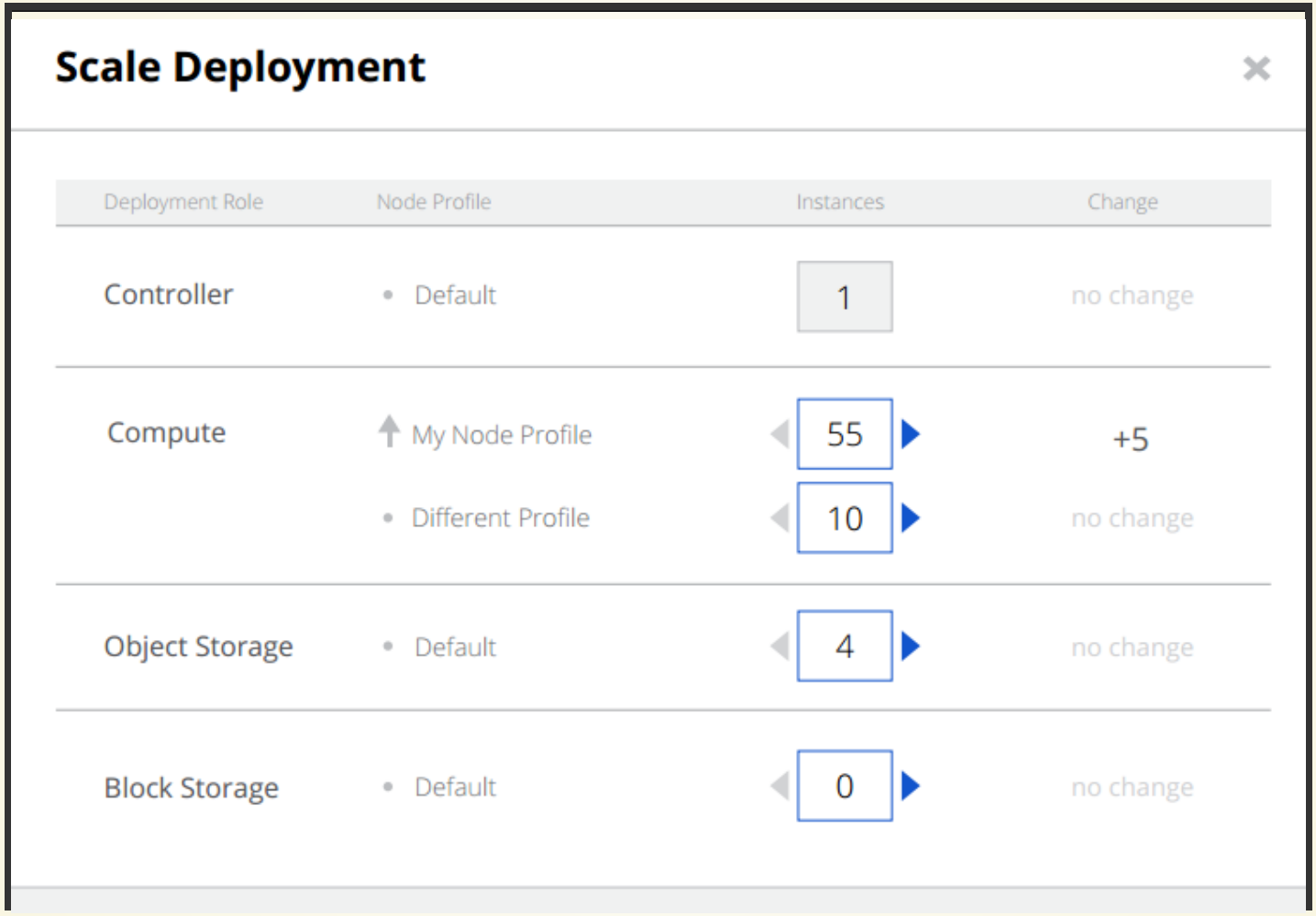
- Extending Glance, so it can return list of services packed in image
- Monitoring of services on each baremetal using Ceilometer

FUTURE(JUNO) MONITORING OF THE RUNNING SERVICES



OVERCLOUD MANAGEMENT WITH TUSKAR

SCALING OF THE OVERCLOUD



DOWNSCALING OF THE OVERCLOUD

- Deleting resource manually then update stack.
- Choosing which resource to unprovision when downscaling, so users can evacuate services running inside, before downscaling happens
- Next: defining hooks for upscaling/downscaling so the evacuating behavior can be defined on template level, so it will be automatic.

FUTURE (JUNO) AUTO-SCALING OF THE OVERCLOUD

- We are waiting on ability to scale nested stacks, implemented in Heat
- Auto-scaling using Ceilometer alarms defined in the Heat template
- Auto-scaling with conditions we need, e.g. cpu_util bigger than 50% for 1 minute

FUTURE (JUNO) AUTO-SCALING EXAMPLES

- Newly plugged hardware can be auto-discovered and auto-scaled.
- It can be combined, as you application is scaling, the more hardware will be needed. That can be taken from e.g. other scaling down and releasing the hardware.

RUNNING OVERCLOUD IN HA

- AMQP - Rabit MQ cluster in active-active
- Openstack API instances - Virtual IP and keepalived + HA proxy
- DB - Mysql Percona XtraDB Cluster

TUSKAR API AND TUSKAR-UI SUMMARY

- Deploy overcloud
- Monitor overcloud
- Manage overcloud
  - Scale
  - Update

RESOURCES

- https://wiki.openstack.org/wiki/TripleO
- http://docs.openstack.org/developer/tripleo-incubator/

QUESTIONS

THANKS FOR YOUR ATTENTION!

GIVE US FEEDBACK! HTTP://DEVCONF.CZ/F/16