



# HA for OpenStack, from the control plane to instances

Hands On

Adam Spiers  
Senior Software Engineer  
[aspiers@suse.com](mailto:aspiers@suse.com)

Charles Wang  
Software Engineer  
[cwang@suse.com](mailto:cwang@suse.com)

# Workshop Environment

## Relax ;-)

- We have plenty of time
- Whole build is also automated and idempotent
- You can take home the entire environment afterwards (available online)
- You can run on any machine with at least 20GB RAM

# Workshop Environment

- We'll build a miniature cloud on a single machine
- libvirt + KVM hypervisor
- 5 VMs
  - Administration Server (Crowbar)
  - 2 Control Nodes in an HA cluster
  - 2 Compute Nodes with HA
- Vagrant for rapid deployment

# What is Vagrant?

“Creates and configures lightweight, reproducible, and portable development environments.”

<https://www.vagrantup.com/>

- Not just for development
- Perfect for “kicking the tyres”, demoing, testing, etc.
- Cross-platform (Linux, MacOS X, Windows)
- Providers for libvirt, VirtualBox, VMware, Hyper-V, Docker, OpenStack, ...

# Vagrant Inputs

- 1 or more Vagrant “box” – pre-built virtual appliances
- `Vagrantfile`: Ruby DSL file which defines:
  - which box(es) to use
  - virtual hardware required
  - virtual network topology
  - network ports to forward
  - hypervisor-specific settings
  - files to inject into appliance
  - commands to run in appliance
- files to inject

# Using Vagrant: Crash Course

- `vagrant box add suse/cloud7-admin`
  - <https://app.vagrantup.com/suse>
  - Also possible to add local boxes
- `vagrant up admin`
- `vagrant up controller1`
- `vagrant halt controller2`
- `vagrant destroy compute1`
- <https://www.vagrantup.com/docs/getting-started/>

# Workshop Vagrant Environment

- <https://github.com/SUSE-Cloud/suse-cloud-vagrant>
  - [demos/HA/](#)
  - [vagrant/](#)
    - [Vagrantfile](#) and [configs/2-controllers-2-computes.yaml](#)
- Libvirt + KVM pre-installed
- 2 boxes pre-installed
  - `suse/cloud7-admin` and `suse/sles12sp2`
- 5 VMs
  - `admin` (SUSE OpenStack Cloud 6 Administration Server)
  - `controller1`, `controller2` (will form an HA cluster)
  - `compute1`, `compute2`

# Starting Point

- `vagrant up` was run

- This was run on the admin server:

```
root@crowbar:~ # /root/bin/setup-node-aliases.sh
```

```
root@crowbar:~ # crowbar batch build HA-compute-cloud-demo.yaml
```

- This was run on one controller:

```
root@crowbar:~ # /root/bin/upload-cirros
```

- 2 controllers in HA cluster
  - 2 nodes that will serve as compute nodes
  - All (relevant) barclamps deployed!
- 
- `cd` to local copy of `git` repository
  - `cd vagrant/`



# How to Access Crowbar

- Connect to admin node
  - `vagrant ssh admin` and `su -` or
  - `ssh root@192.168.124.10` or
  - use VM console in `virt-manager` / `VirtualBox`
- Root password is `vagrant`
- Accept the EULAs (for each EULA, read it and type `q` then `y`)
- Point a browser at the Crowbar web UI
  - <http://localhost:8000>
  - Default credentials: `crowbar / crowbar`
- Check the 5 nodes are registered, named correctly, and in Ready state (green)

# Add remotes to Pacemaker cluster

# Pacemaker Barclamp Clusters, Nodes, and Roles

Deployment

[Raw](#)

Drag nodes for deployment from Available Nodes into the selected Role

Available Nodes

cloud6-admin 

compute1 


compute2 


controller1 

controller2 

pacemaker-cluster-member

[Remove all](#)


controller1 

controller2 

hawk-server

[Remove all](#)

controller1 

controller2 

pacemaker-remote

[Remove all](#)


# Pacemaker Role Assignment

Deployment

[Raw](#)

Drag nodes for deployment from Available Nodes into the selected Role

## Available Nodes

 cloud6-admin 

 compute1 

 compute2 

 controller1 

 controller2 

## pacemaker-cluster-member

[Remove all](#)

 controller1 

 controller2 

## hawk-server

[Remove all](#)

 controller1 

 controller2 

## pacemaker-remote

[Remove all](#)

 compute1 

 compute2 

# Pacemaker STONITH Configuration

## STONITH

### Configuration mode for STONITH

Configured with STONITH Block Devices (SBD) ▼

Manual configuration is required for SBD: before applying the proposal, you will have to ensure that the devices are available and initialized for SBD. You will also need to manually setup a watchdog if not all nodes use the same watchdog kernel module. Refer to the [High Availability Guide](#) for details.

### Kernel module for watchdog

softdog

Leave the watchdog module attribute empty if the nodes need different watchdog modules.

Node name	Block devices for node
controller1	<input type="text" value="/dev/vdb"/>
controller2	<input type="text" value="/dev/vdb"/>
compute1	<input type="text"/>
compute2	<input type="text"/>

# Pacemaker STONITH Configuration

## STONITH

### Configuration mode for STONITH

Configured with STONITH Block Devices (SBD) ▼

Manual configuration is required for SBD: before applying the proposal, you will have to ensure that the devices are available and initialized for SBD. You will also need to manually setup a watchdog if not all nodes use the same watchdog kernel module. Refer to the [High Availability Guide](#) for details.

### Kernel module for watchdog

softdog

Leave the watchdog module attribute empty if the nodes need different watchdog modules.

Node name	Block devices for node
controller1	/dev/vdb
controller2	/dev/vdb
compute1	/dev/vdb
compute2	/dev/vdb

# Apply Pacemaker Proposal

Save

Apply

Delete

Cancel

# Check Progress of Proposal

```
root@crowbar:~ # tail -f /var/log/crowbar/production.log
```

```
root@crowbar:~ # tail -f /var/log/crowbar/chef-client/*.log
```



# Check Status of Cluster Nodes and Remotes

Login to one of the controller nodes, and do:

```
root@d52-54-77-77-77-01:~ # crm status
Last updated: Mon May  2 17:11:24 2016          Last change: Fri Apr 29 11:3
0:48 2016 by root via crm_resource on d52-54-77-77-77-01
Stack: unknown
Current DC: d52-54-77-77-77-01 (version unknown) - partition with quorum
4 nodes and 93 resources configured

Online: [ d52-54-77-77-77-01 d52-54-77-77-77-02 ]
RemoteOnline: [ remote-d52-54-77-77-77-03 remote-d52-54-77-77-77-04 ]
```

Full list of resources:

```
stonith-d52-54-77-77-77-01    (stonith:external/libvirt):    Started d52-54-77-77-77-02
stonith-d52-54-77-77-77-02    (stonith:external/libvirt):    Started d52-54-77-77-77-01
stonith-remote-d52-54-77-77-77-03    (stonith:external/libvirt):    Started d52-54-77-77-77-01
stonith-remote-d52-54-77-77-77-04    (stonith:external/libvirt):    Started d52-54-77-77-77-02
remote-d52-54-77-77-77-03    (ocf::pacemaker:remote):      Started d52-54-77-77-77-01
remote-d52-54-77-77-77-04    (ocf::pacemaker:remote):      Started d52-54-77-77-77-02
```

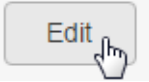
`nova setup`

# Edit Nova Proposal



Nova

OpenStack Compute: Provision and manage large network of virtual machines



# Nova Proposal: Clusters Available

Deployment

Raw

Drag nodes for deployment from Available Nodes into the selected Role

Available Clusters

?

Search

services

Available Clusters with Remote Nodes

?

Search

services (2 remote nodes)

Available Nodes

Search

compute1

compute2

controller1

controller2

crowbar

nova-controller

Remove all

nova-compute-docker

Remove all

nova-compute-hyperv

Remove all

nova-compute-kvm

Remove all

nova-compute-qemu

Remove all

nova-compute-vmware

Remove all

nova-compute-xen

Remove all

nova-compute-zvm

Remove all

# Nova Proposal: Role Assignment

Deployment Raw

Drag nodes for deployment from Available Nodes into the selected Role

Available Clusters	?	nova-controller	Remove all
<input type="text" value="Search"/>		services	
services		nova-compute-docker	Remove all
<input type="text" value="Search"/>		<input type="text"/>	
Available Clusters with Remote Nodes	?	nova-compute-hyperv	Remove all
<input type="text" value="Search"/>		<input type="text"/>	
services (2 remote nodes)		nova-compute-kvm	Remove all
Available Nodes		services (2 remote nodes)	
<input type="text" value="Search"/>		nova-compute-qemu	Remove all
compute1		<input type="text"/>	
compute2		nova-compute-vmware	Remove all
controller1		<input type="text"/>	
controller2		nova-compute-xen	Remove all
crowbar		<input type="text"/>	
		nova-compute-zvm	Remove all
		<input type="text"/>	

# Apply Nova Proposal

Save

Apply

Delete

Cancel

# Check Progress of Proposal

```
root@crowbar:~ # tail -f /var/log/crowbar/production.log
```

```
root@crowbar:~ # tail -f /var/log/crowbar/chef-client/*.log
```

# Check Status of nova resources in Cluster

Login to one of the controller nodes, and do:

```
root@d52-54-77-77-77-01:~ # crm status
Last updated: Mon May  2 17:11:24 2016          Last change: Fri Apr 29 11:3
0:48 2016 by root via crm_resource on d52-54-77-77-77-01
Stack: unknown
Current DC: d52-54-77-77-77-01 (version unknown) - partition with quorum
4 nodes and 93 resources configured

Online: [ d52-54-77-77-77-01 d52-54-77-77-77-02 ]
RemoteOnline: [ remote-d52-54-77-77-77-03 remote-d52-54-77-77-77-04 ]

Clone Set: cl-g-nova-compute [g-nova-compute]
Started: [ remote-d52-54-77-77-77-03 remote-d52-54-77-77-77-04 ]
nova-evacuate (ocf::openstack:NovaEvacuate): Started d52-54-77-77-77-02
fence-nova (stonith:fence_compute): Started d52-54-77-77-77-02
```



# Shared Storage

# How is Shared Storage Setup for the Workshop?

We're using the admin server's NFS server:

- Only suitable for testing purposes!
- In production, use SES / SAN

# Verify Setup of Shared Storage

- Locate shared directories via `nfs_client` barclamp
- Check `/etc/exports` on admin server
- Check `/etc/fstab` on controller / compute nodes
- Run `mount` on controller / compute nodes

# Boot a VM

# Boot a VM

Let's boot a VM to test compute node HA!

Connect to one of the controller nodes, and get image / flavor / net names:

```
source .openrc
openstack image list
openstack flavor list
neutron net-list
```

Boot the VM using these ids:

```
nova boot --image image --flavor flavor --nic net-id=net testvm
```

Test it's booted:

```
nova show testvm
```

# Assign a Floating IP

Create floating IP:

```
neutron floatingip-create floatingnet
```

Get VM IP:

```
nova list
```

Get port id:

```
neutron port-list | grep vmIP
```

Associate floating IP with VM port:

```
neutron floatingip-associate floatingipID portID
```

# Allow ICMP and SSH for VMs

The VMs use the `default` security group (by default).

Make sure it has ICMP:

```
openstack security group rule create --proto icmp default
```

Also allow SSH:

```
openstack security group rule create --proto tcp --dst-port 22  
default
```



# Set Up Monitoring (1/2)

- Recommended in separate windows/terminals
- From either of the controller nodes

Ping VM:

```
ping vmFloatingIP
```

Ping host where the VM is running:

```
nova list --fields host,name  
ping host
```



# Set Up Monitoring (2/2)

Find node running nova - evacuate:

```
crm resource show nova-evacuate
```

On that node, check log messages for NovaEvacuate workflow:

```
tail -f /var/log/messages | grep NovaEvacuate
```

Monitor cluster status:

```
crm_mon
```

# Test Compute Node Failover (the exciting bit!)

# Simulate Compute Node Failure

Login to compute node where VM runs, and type:

```
kill -9 -f pacemaker_remoted
```

This will cause fencing! (Why?)

# Verify Recovery

- Ping to the VM is interrupted, then resumed
- Ping to the compute node is interrupted (then resumed)
- Log messages show:
  - `NovaEvacuate [...] Initiating evacuation`
  - `NovaEvacuate [...] Completed evacuation`
- `crm status` shows compute node offline (then back online)
- Verify compute node was fenced
  - Check `/var/log/messages` on DC
- Verify VM moved to another compute node
  - `nova list --fields host,name`

# Troubleshooting

# Verifying Compute Node Failure Detection

Pacemaker monitors compute nodes via `pacemaker_remote`.

If compute node failure detected:

1. compute node is fenced
  - `crm_mon` etc. will show node unclean / offline
2. Pacemaker invokes `fence-nova` as secondary fencing resource

```
crm configure show fencing_topology
```

Find node running `fence_compute`:

```
crm resource show fence-nova
```

# Verifying Secondary Fencing

`fence_compute` script:

1. tells nova server that node is down
2. updates attribute on compute node to indicate node needs recovery

Log files:

- `/var/log/nova/fence_compute.log`
- `/var/log/messages` on DC and node running `fence-nova`

Verify attribute state via:

```
attrd_updater --query --all --name=evacuate
```

# Verifying Compute Node Failure Recovery Process

1. NovaEvacuate spots attribute and calls nova evacuate

```
root@controller1:~ # crm resource show nova-evacuate
resource nova-evacuate is running on: d52-54-77-77-77-02
```

2. nova resurrects VM on other node

```
root@controller2:~ # grep nova-evacuate /var/log/messages
NovaEvacuate [...] Initiating evacuation
NovaEvacuate [...] Completed evacuation
```

Warning: no retries if resurrection fails!



# Process Failures

`pacemaker_remote` looks after key compute node services.

Exercise:

- Use `crm on cl-g-nova-compute` to find out which services it looks after
- Try killing a process and see what happens
  - nothing, thanks to [bsc#901796](#)
- Try *stopping* a process and see what happens
- Try breaking a process (e.g. corrupt config file and restart)



We adapt. You succeed.