Postgres on OpenStack

- Dave Page | 18/9/2014

Introduction

- PostgreSQL:
  - Core team member
  - pgAdmin lead developer
  - Web/sysadmin teams
  - PGCAC/PGEU board member
- EDB:
  - Tools
  - Cloud products
  - Configuration management
  - Packaging/distribution
  - Infrastructure

What is OpenStack?

- Cloud environment, similar to AWS:
  - Private implementations
  - Public implementations – Rackspace, HP Cloud
- Modular design, offering a choice of services including:
  - Image Service (Glance)
  - Block Storage (Cinder)
  - Compute (Nova)
  - Network (Neutron)
  - Object Storage (Swift)

Glance Image Service

- Store virtual machine images and snapshots
- Preconfigure an image or snapshot with the desired OS/software
- Share images with members of selected projects (AKA, tenants).

Cinder Block Storage Service

- Analogous to Amazon Elastic Block Storage (EBS)
- Provides block storage devices to virtual machine instances
- Can be provided from one or more service nodes via a controller
- Service nodes may use various storage systems, with different properties including:
  - LVM
  - GlusterFS
  - NFS
  - Various SANs

Nova Compute Service

- Analogous to Amazon Elastic Compute Service (EC2)
- Schedules and runs VM instances
- Multiple machine flavours with individual RAM, CPU and disk requirements
- Compute node selection is highly configurable:
  - Pluggable scheduler filters
  - Target images to specific hosts
  - Target flavours to specific hosts
  - Target instances to hosts with enough free resources

Neutron Networking Service

- Provides network services to VM instances
- Attach instances to one or more virtual networks
- Map floating/public IP addresses to a VM
- Configure firewall rules for ingress and egress

Swift Object Storage

- Analogous to Amazon Simple Storage Service (S3)
- Provides highly redundant object storage
- Suitable for storage and retrieval of whole files

Wait, what about Trove?

- Still in relatively early stages of development
- Currently seems focused on MySQL/Cassandra
- No support for clustering or other non-trivial setups
- API favours psql/pgAdmin operations, not deployment/ configuration

Well that all sounds pretty cool, but what can we do with it?

Database Server Images

Database server images

- Preconfigure a “standard” database image
- Enables consistency, which leads to ease of management
- Preinstall commonly required software:
  - Procedural languages
  - WAL-E
  - pgBouncer/pgPool
- Minimise launch/deployment time

Image or Snapshot?

- Technically there's not much difference
- OCD: I like images to be, well images
- Images may be considered "blessed" or approved
- Snapshots are a convenient way to take backups, perhaps of work in progress
- Images are "clean" configurations, whilst snapshots may contain artifacts from previous use, e.g. shell history or log files.

Creating an image

- Create a small root disk in QCOW2 format
- Install the base OS on the disk (I use KVM on RHEL)
- Install the required software
- Add the EPEL Yum repo, and install cloud-init
- Disable zeroconf for the network
- Shutdown the VM
- Run *virt-sysprep* to clean the OS installation

Uploading the image

```
$ glance image-create --name ”PostgreSQL-9.4" --disk-format qcow2 \
  --container-format bare --is-public True --progress < pg-9.3.img
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | 64d7c1cd2b6f60642c14663941cb7913     |
| container_format | bare                                 |
| created_at       | 2014-08-29T14:55:08                  |
| deleted          | False                                |
| deleted_at       | None                                 |
| disk_format      | qcow2                                |
| id               | acafc7c0-40aa-4026-9673-b879124bcec2 |
| is_public        | True                                 |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | PostgreSQL-9.4                       |
| owner            | efa984b0a914450e9a47788ad330699d     |
| protected        | False                                |
| size             | 2013167616                           |
| status           | active                               |
| updated_at       | 2014-08-29T14:55:08                  |
+------------------+--------------------------------------+
```

Database Server Machine Configurations

Dedicated database flavours

- “Flavors” (without the u) describe the instance configuration:
  - vCPUs
  - Memory
  - Root disk
  - Ephemeral disk
  - Swap
  - Other attributes
- Additional attributes allow us to target specific hypervisor properties, e.g.
  - Compute nodes with SSDs

Targeting a host aggregate

- Create a host aggregate or group containing the desired compute nodes
- Add a property to the host aggregate to identify the desired attribute for matching
- Create the desired flavour(s)
- Add a property to the flavour to correspond with the property on the host aggregate
- Share the flavour with one or more tenants

Create host aggregate

```
$ nova aggregate-create pg-servers
+----+------------+-------------------+-------+----------+
| Id | Name       | Availability Zone | Hosts | Metadata |
+----+------------+-------------------+-------+----------+
| 14 | pg-servers | -                 |       |          |
+----+------------+-------------------+-------+----------+

$ nova aggregate-add-host pg-servers nova5.ox.uk
Host nova5.ox.uk has been successfully added for aggregate 14
+----+------------+-------------------+---------------+----------+
| Id | Name       | Availability Zone | Hosts         | Metadata |
+----+------------+-------------------+---------------+----------+
| 14 | pg-servers | -                 | 'nova5.ox.uk' |          |
+----+------------+-------------------+---------------+----------+

$ nova aggregate-set-metadata pg-servers pgsvr=true
Metadata has been successfully updated for aggregate 14.
+----+------------+-------------------+---------------+--------------+
| Id | Name       | Availability Zone | Hosts         | Metadata     |
+----+------------+-------------------+------- -------+--------------+
| 14 | pg-servers | -                 | 'nova5.ox.uk' | 'pgsvr=true' |
+----+------------+-------------------+---------------+--------------+
```

Create flavour

```
$ nova flavor-create --ephemeral 0 --swap 4096 --is-public true
pgsvr.small auto 4096 40 4
+--------------------------------------+-------------+-----------+...
| ID                                   | Name        | Memory_MB |...
+--------------------------------------+-------------+-----------+...
| 44221572-a7e0-4660-97fc-352467743fce | pgsvr.small | 4096      |...
+--------------------------------------+-------------+-----------+...

...------+-----------+------+-------+-------------+-----------+
... Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public |
...------+-----------+------+-------+-------------+-----------+
... 40   | 0         | 4096 | 4     | 1.0         | True      |
...------+-----------+------+-------+-------------+-----------+
```

Create flavour

```
$ nova flavor-key pgsvr.small set aggregate_instance_extra_specs:pgsvr=true
$ nova flavor-show pgsvr.small
+----------------------------+-----------------------------------------------------+
| Property                   | Value                                               |
+----------------------------+-----------------------------------------------------+
| OS-FLV-DISABLED:disabled   | False                                               |
| OS-FLV-EXT-DATA:ephemeral  | 0                                                   |
| disk                       | 40                                                  |
| extra_specs                | {"aggregate_instance_extra_specs:pgsvr": "true"}    |
| id                         | 44221572-a7e0-4660-97fc-352467743fce                |
| name                       | pgsvr.small                                         |
| os-flavor-access:is_public | True                                                |
| ram                        | 4096                                                |
| rxtx_factor                | 1.0                                                 |
| swap                       | 4096                                                |
| vcpus                      | 4                                                   |
+----------------------------+-----------------------------------------------------+
```

A note on Scheduler Filters

- Scheduler filters determine which compute node a new instance will be created on
- Filters are chained, and can reject hosts based on:
  - Capabilities
  - Current utilisation
  - Physical location
  - Affinity
  - Other factors such as tenant isolation requirements
- Our example used the *aggregate_instance_extra_specs* filter
- Custom filters can be defined in JSON (*json_filter*), or written from scratch in Python

Instance Storage
-or-
Where do my VMs live?

Local storage

- Known as *Instance Store* on AWS
- Images are copied to local storage on the compute node and started
- Pros:
  - Easy to setup
  - Performant
  - Minimal hardware requirements
- Cons:
  - Moving instances requires downtime
  - Potentially less reliable

Shared instance storage

- Same as Instance Store, but uses storage shared between compute nodes using NFS, iSCSI, etc.
- Pros
  - Allows easy migration of instances between compute nodes
  - Potentially more reliable storage (if a SAN or similar is used)
- Cons
  - “Local” I/O is over the network
  - More complex to setup and maintain
  - Can be far more expensive (SAN, 10GBe or fibre channel network etc.)

Volume storage

- Known as *Elastic Block Store* (EBS) on AWS
- Root volume is stored in Cinder
- Pros:
  - Available as a launch-time option, if Cinder is configured
  - Allows easy migration of instances between compute nodes
  - Potentially more reliable storage (if a SAN or similar is used)
- Cons:
  - “Local” I/O is over the network
  - More complex to setup and maintain
  - Can be far more expensive (SAN, 10GBe or fibre channel network etc.)

Data Storage

Data storage

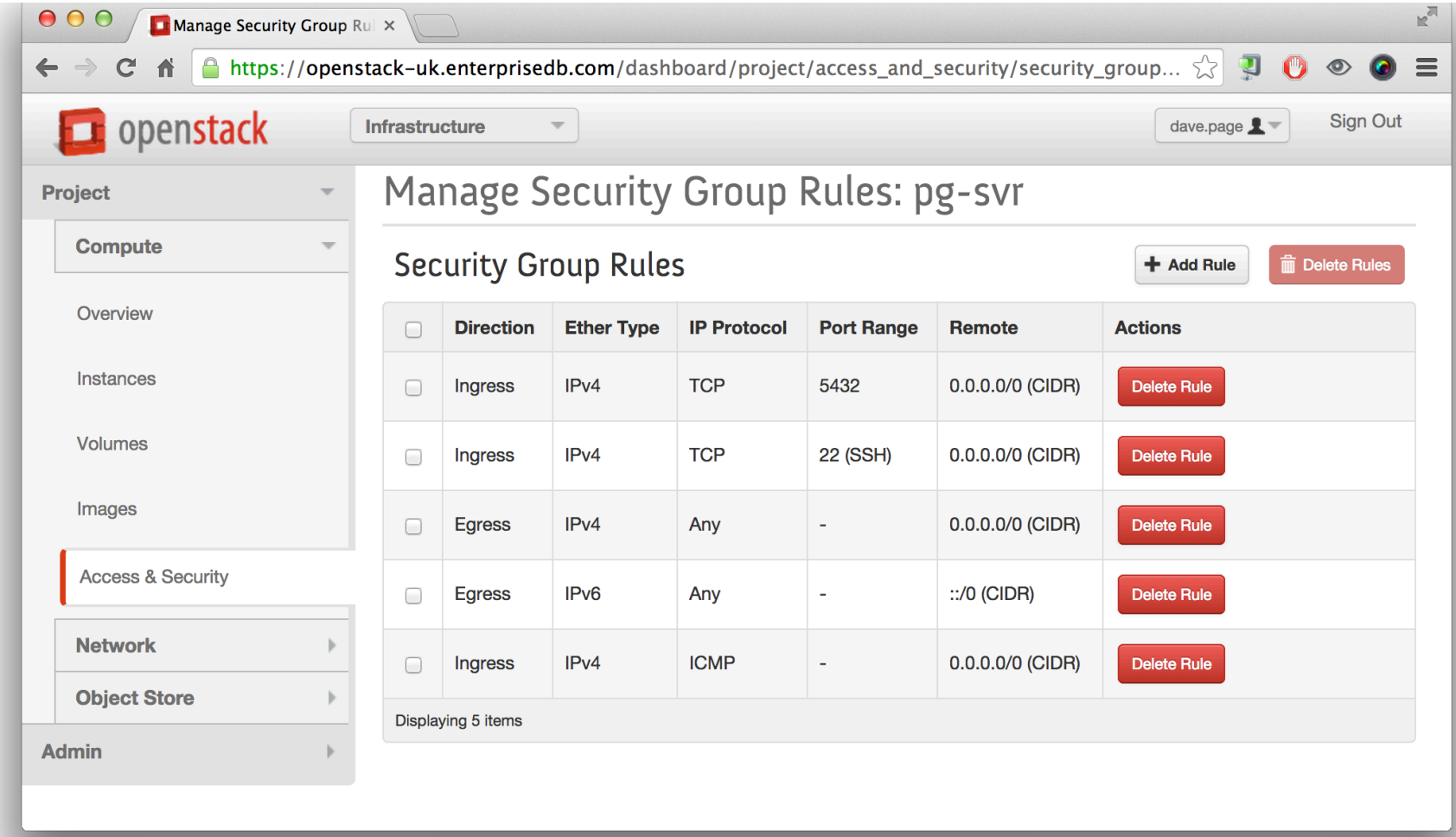
- In the instance:
  - Easy to setup and manage – click and go
  - Resizing will pause the instance
  - Not very “elastic”
  - Slower snapshots – data has to be moved to Glance
- In Cinder:
  - More complex to setup
  - More expensive for a high performance setup
  - Add storage on the fly by adding volumes and utilising LVM
  - Can choose storage types – SSD vs. rust, low vs. high IOPs
  - Fast snapshots – Cinder can use filesystem/SAN snapshots

Firewalling

Instance firewalling

- Neutron provides *Security Groups* to define firewall rules
- No ingress access by default
- Each security group has one or more ingress or egress rules defined
- All instances have one or more security groups associated with them
- Allows easy configuration of a standard set of firewall rules for multiple servers

Security group configuration

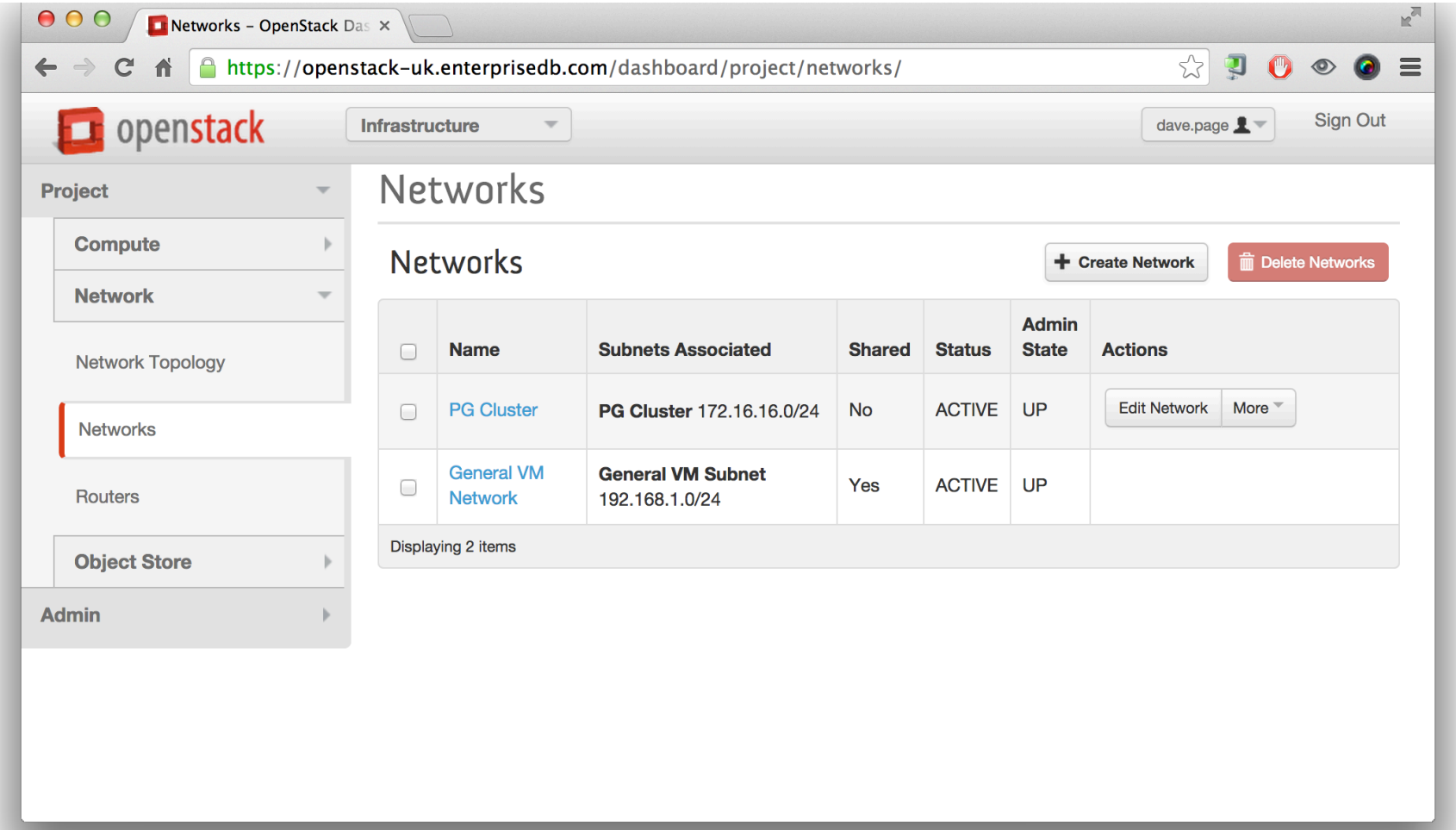


Networking

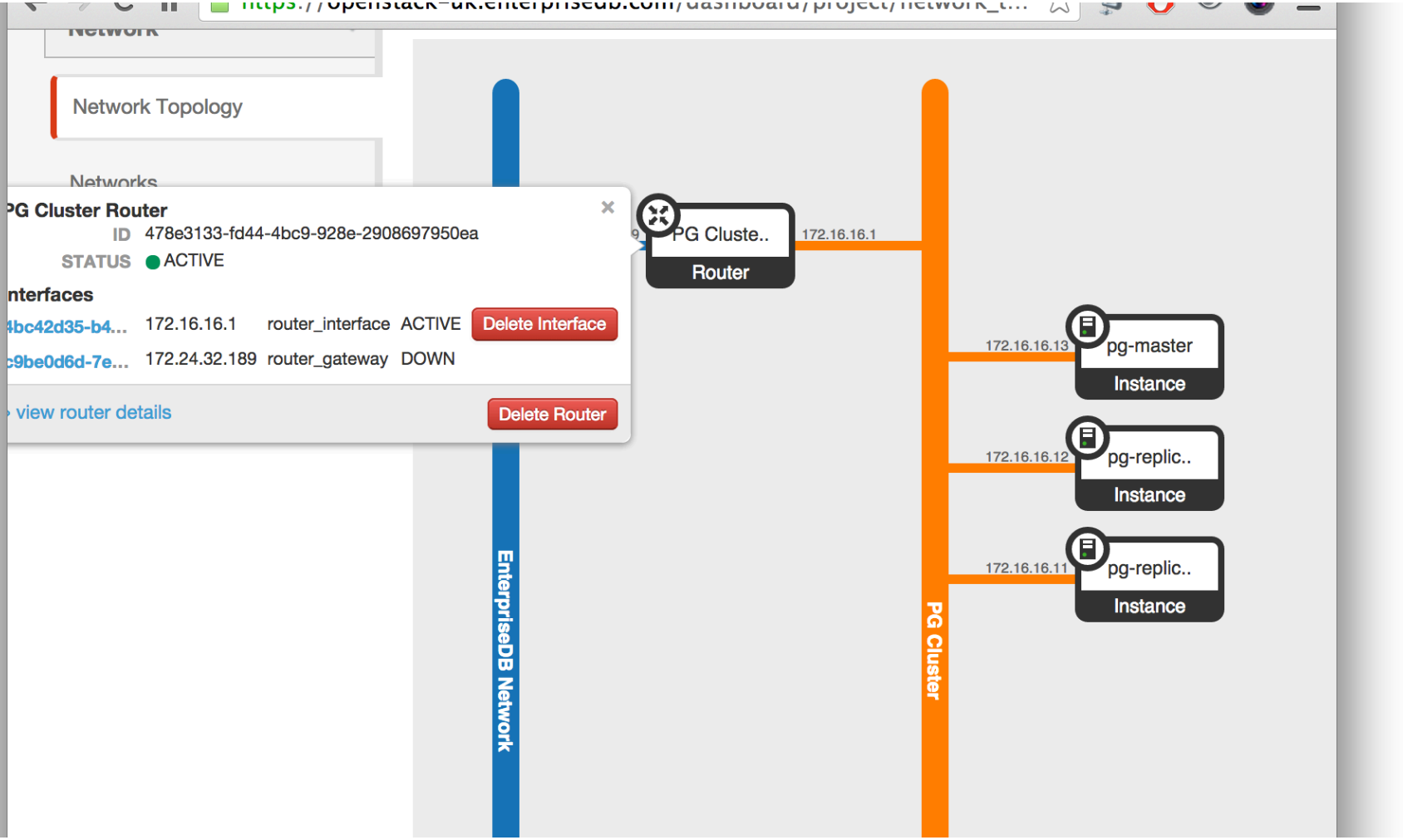
Private cluster network

- Create one or more virtual networks to contain a cluster of database servers, entire application or some other group of instances
- Analogous to *Virtual Private Cloud* (VPC) in Amazon
- Optionally attach virtual routers to networks
- Attach one or more networks to each instance
- Use *Floating IPs* (Elastic IPs) to access instances from outside the virtual network
- Each network can have a DHCP server and custom routes

Network configuration



Network topology



WAL Archiving

WAL archiving to Swift

- Object storage is ideal for storing WAL segments and backups
- Highly redundant storage offers reliability and scaling
- Easy to use with WAL-E (https://github.com/wal-e/wal-e)
  - WAL archival and retrieval
  - Base backup creation and restoration
  - WAL cleanup/pruning

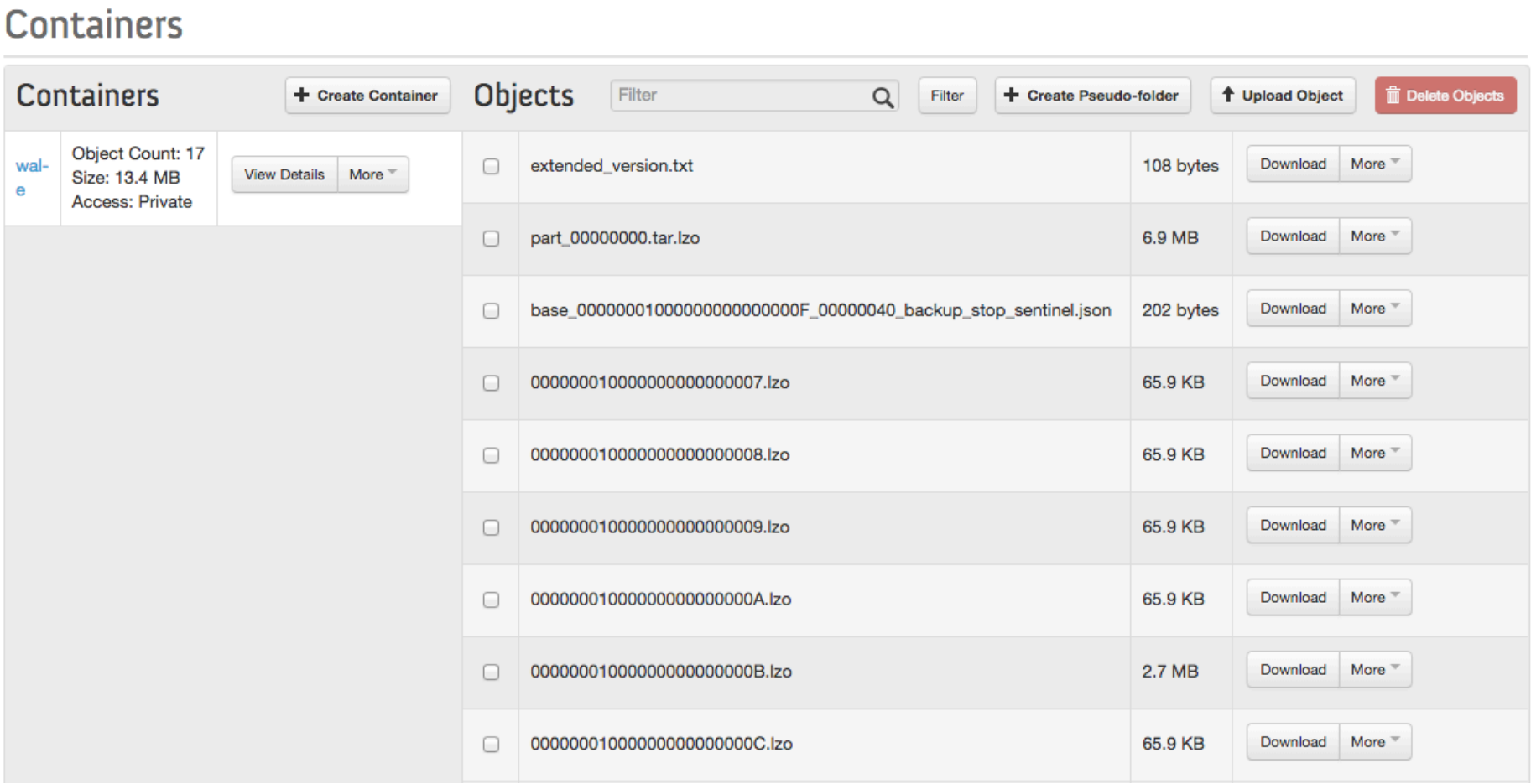
Installing WAL-E

```
# pip install wal-e
<stuff happens>

# vi /var/lib/pgsql/9.3/data/postgresql.conf
wal_level = archive
archive_mode = on
archive_command = '/var/lib/pgsql/wal-e.sh %p'
archive_timeout = 60   # For testing

# vi /var/lib/pgsql/wal-e-creds.sh
export SWIFT_AUTHURL=http://auth.enterprisedb.com:35357/v2.0
export SWIFT_USER=wal-e
export SWIFT_PASSWORD=ReallyReallySecret
export SWIFT_TENANT=admin
export WALE_SWIFT_PREFIX=swift://wal-e

# chown postgres:postgres /var/lib/pgsql/wal-e-creds.sh
# chmod 600 /var/lib/pgsql/wal-e-creds.sh
```

Installing WAL-E

```
# vi /var/lib/pgsql/wal-e.sh
#!/bin/sh
source /var/lib/pgsql/wal-e-creds.sh
/usr/bin/wal-e wal-push $1

# chmod +x /var/lib/pgsql/wal-e-creds.sh
# service postgresql-9.3 restart
# tail -f /var/lib/pgsql/9.3/data/pg_log/postgresql.log
wal_e.main   INFO     MSG: starting WAL-E
        DETAIL: The subcommand is "wal-push".
        STRUCTURED: time=2014-08-29T16:55:56.236552-00 pid=18301
wal_e.worker.upload INFO     MSG: begin archiving a file
        DETAIL: Uploading "pg_xlog/0000000100000000000000E" to "swift://wal-e/
wal_005/0000000100000000000000E.lzo".
        STRUCTURED: time=2014-08-29T16:55:56.360962-00 pid=18301 action=push-wal
key=swift://wal-e/wal_005/0000000100000000000000E.lzo prefix= seg=0000000100000000000000E
state=begin
urllib3.connectionpool INFO     Starting new HTTP connection (1): auth.enterprisedb.com
urllib3.connectionpool INFO     Starting new HTTP connection (1): swift.enterprisedb.com
wal_e.worker.upload INFO     MSG: completed archiving to a file
        DETAIL: Archiving to "swift://wal-e/wal_005/0000000100000000000000E.lzo" complete
at 9.29666KiB/s.
        STRUCTURED: time=2014-08-29T16:56:03.467501-00 pid=18301 action=push-wal
key=swift://wal-e/wal_005/0000000100000000000000E.lzo prefix= rate=9.29666
seg=0000000100000000000000E state=complete
```

WAL Archives



Summary

Summary

- Deploy standardised, maintainable database servers from images in Glance
- Configure database-optimised compute nodes
- Configure database-optimised machine flavours in Nova to target desired nodes
- Select instance storage based on performance and flexibility requirements vs. budget.
- Select data storage based on complexity vs. flexibility

Summary contd.

- Use Neutron's security groups to provide standardised security configuration outside of the instance
- Utilise virtual networks to isolate logical clusters or groups of machines
- Archive base backups and WAL to Swift with WAL-E for ease of backup management, redundancy and scalability

Questions?