UNIVERSITY OF CAMBRIDGE

# Utilizing OpenStack Infrastructure for Research Storage Services

Paul Browne <pfb29@cam.ac.uk>

# Generic L4 TProxy Stack

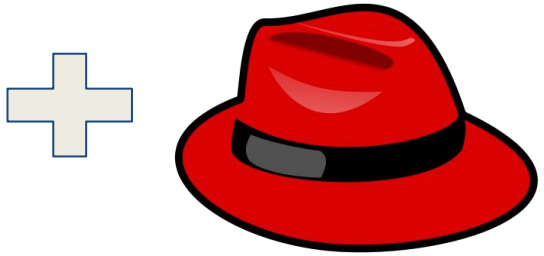
Not all users of the storage services will be pre-existing users of HPC services, so there is a need for alternative access mechanisms.

Access requirements:

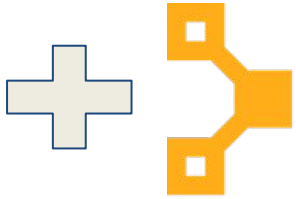
- RDS and RCS has a requirement for access over SSH-using applications
- RFS/IFS has a requirement for access over SMB/CIFS

A standard load-balancer and transparent proxy stack can be applied to both services to expose them over the CUDN, and made HA inside the OpenStack environment.

Orchestration

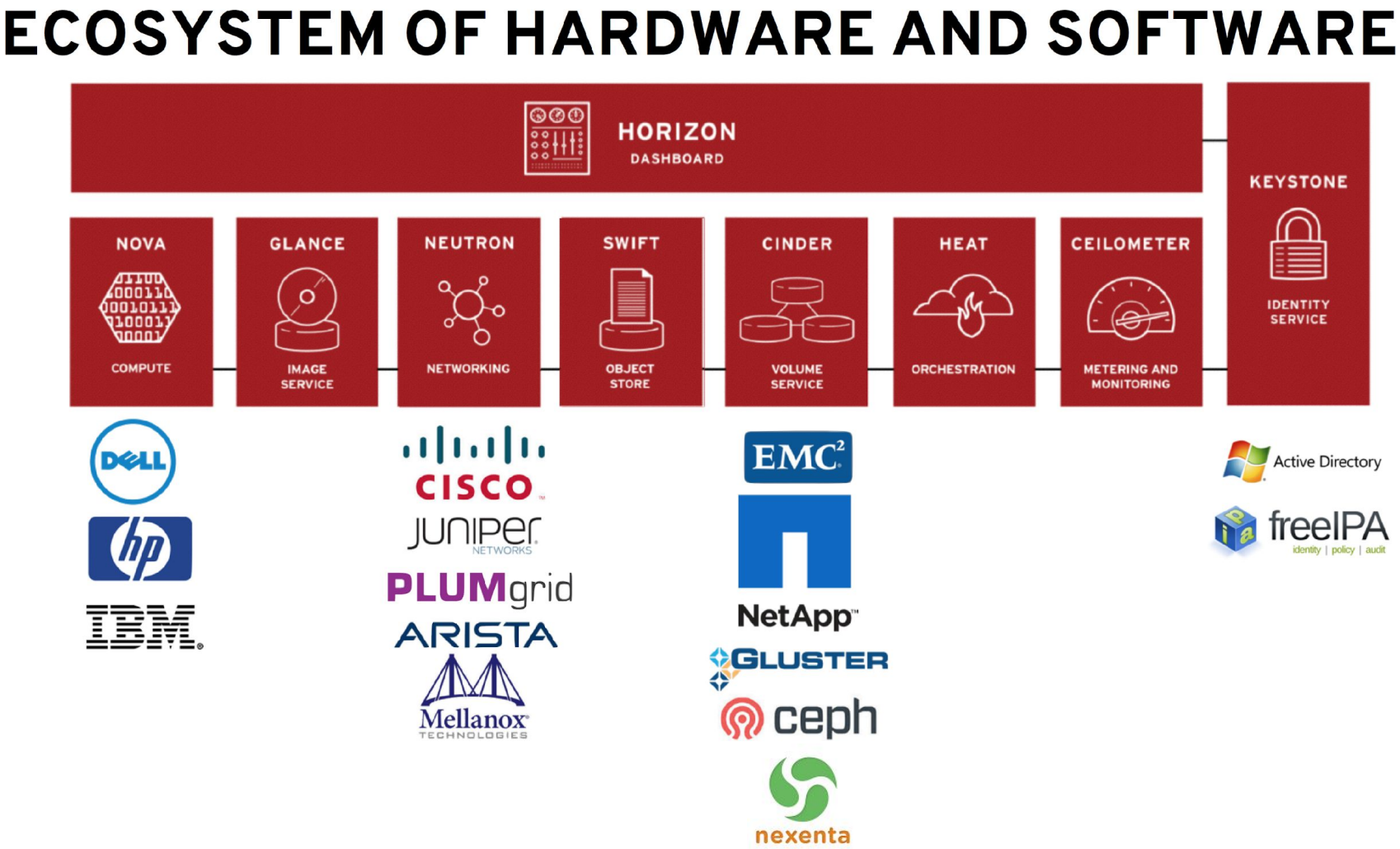


RHEL base OS




Version control / Config management

# OpenStack IaaS



# Generic L4 TProxy Stack


HAProxy can run as a proxy/load-balancer in either L4 (TCP/IP) or L7 (HTTP) mode.

- Munging of traffic flow and HTTP header manipulation at L7 is a standard use case.
- TLS termination and offload at network edge, rich filtering and access control lists, etc

This kind of manipulation isn't directly possible running HAProxy in the lower L4 mode, but we'd still like our backends sitting behind the load-balancer edge to record true client connections and IP addresses for logging, monitoring and security purposes.

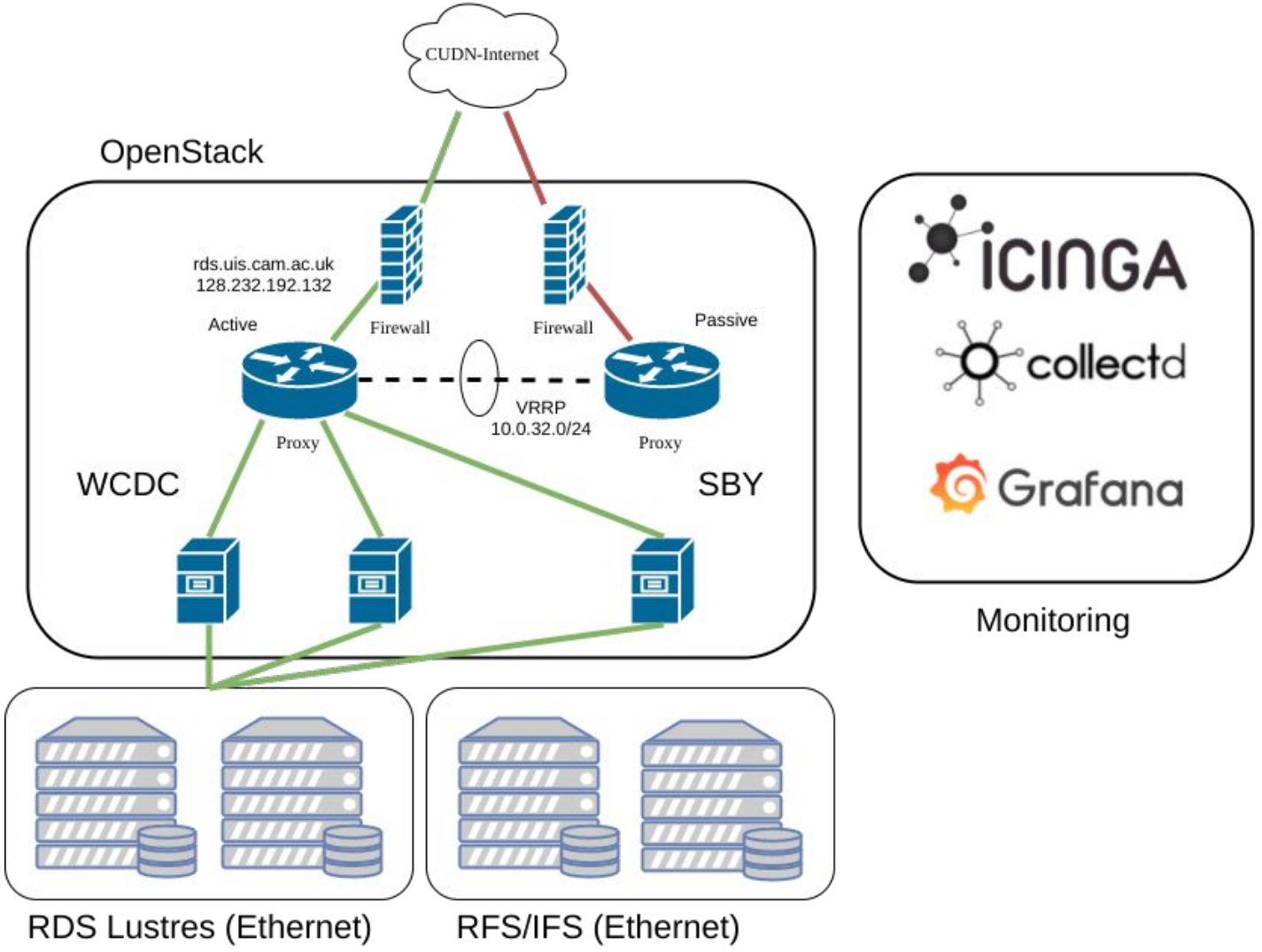
- Use cases on RDS; Dynamic black-hole of abusive SSH clients.
- Log shipping for later analysis.
- Collect login and traffic metrics for visualization.

Solution is to use the Linux kernel tproxy module and HAProxy compiled with tproxy support;

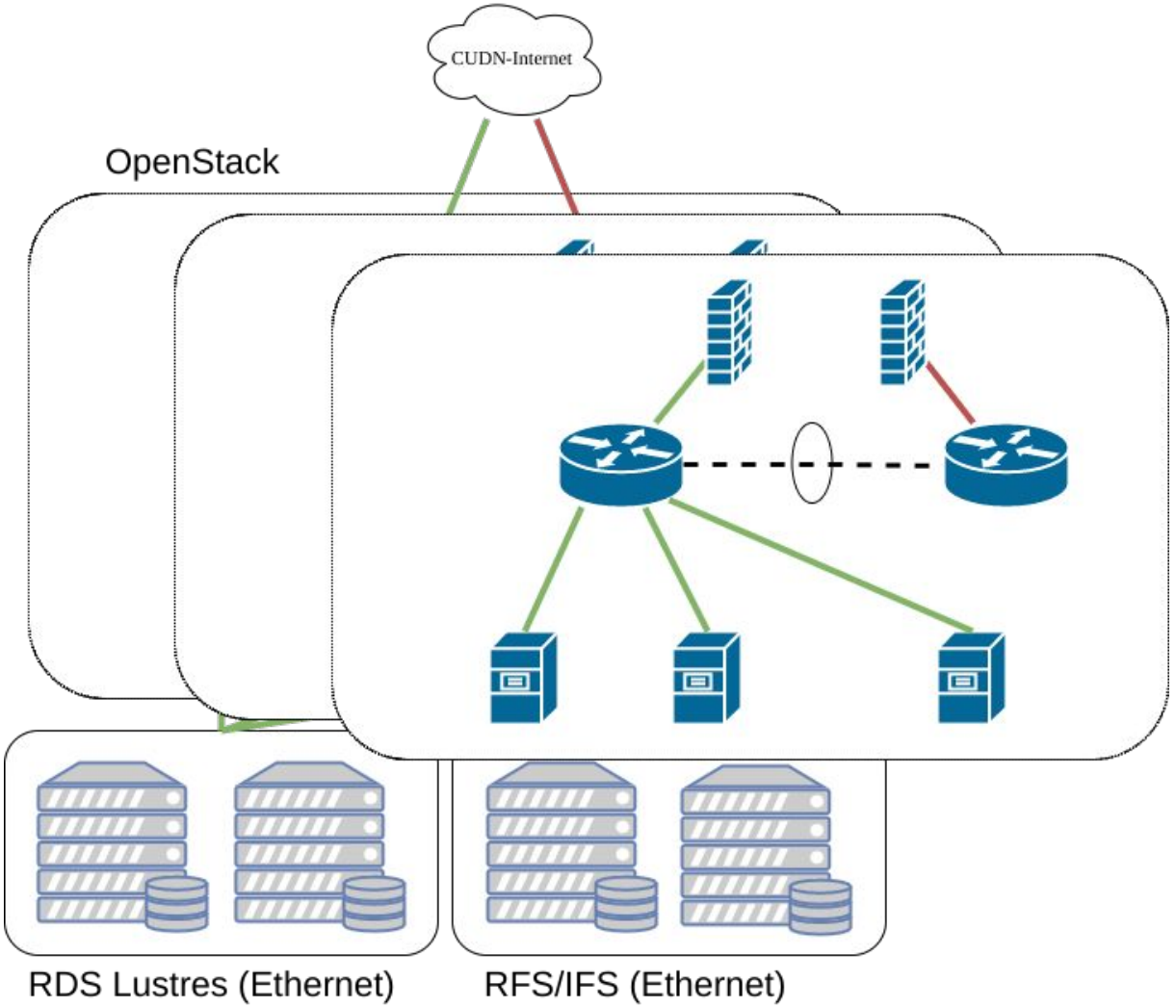
- IPTables pre-routing intercepts traffic, marks it and diverts to a policy routing table to the loopback
- HAProxy collects marked traffic, spoofs client IP to backends, and handles return traffic as backend gateway.

Finally, as in OpenStack proper, the KeepaliveD Linux implementation of VRRP provides an active-passive pair level of high-availability for the front-end proxies, where a service IP floats over the pair.

# Generic L4 TProxy Stack



# Generic L4 TProxy Stack



# Thank You

Research Computing Platforms @ UIS

Team Lead: Wojciech Turek
Team: Alasdair King
Joe Stankiewicz
Matt Raso-Barnett
Paul Browne

# Questions ?

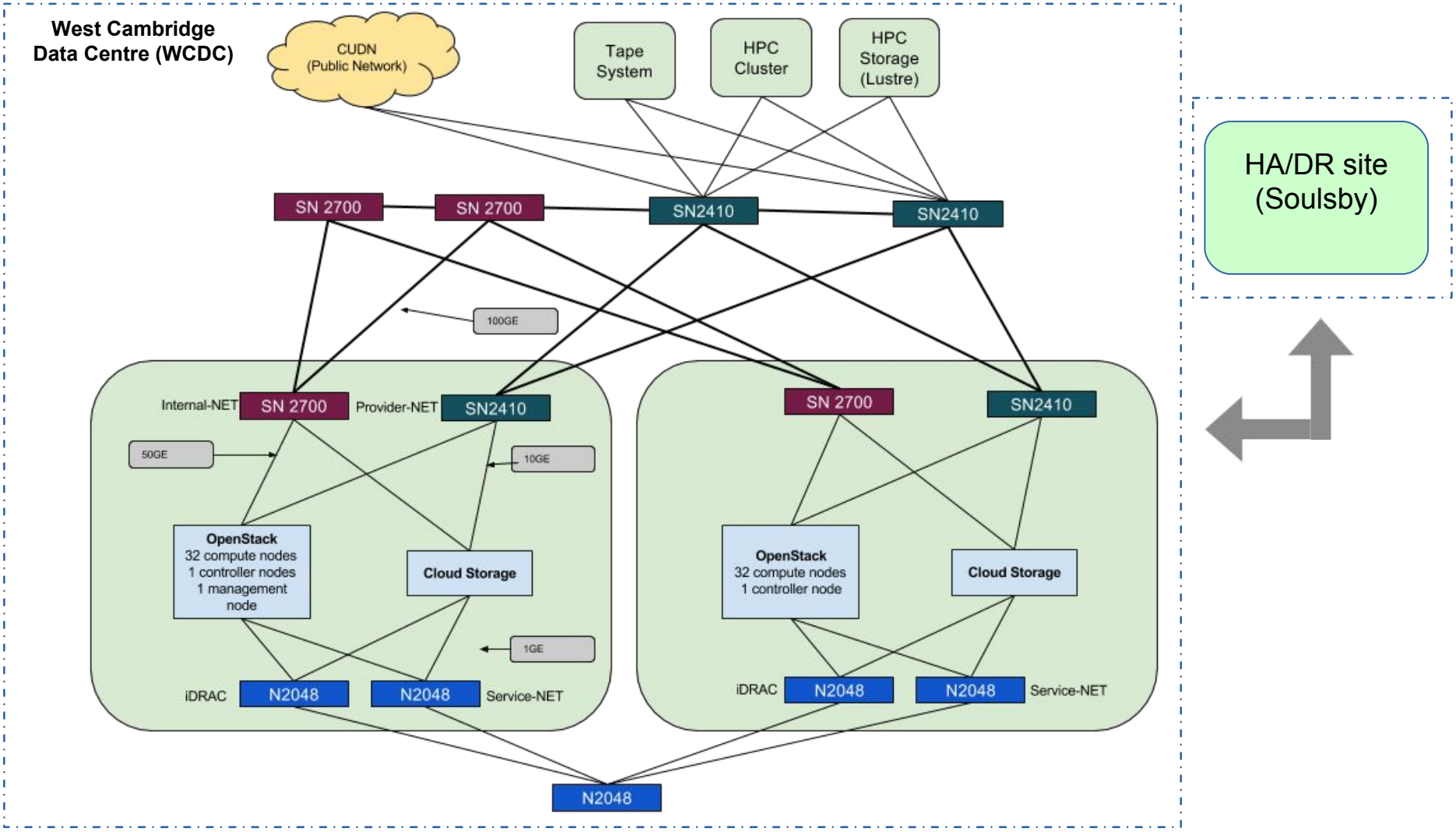
# OpenStack Hardware

| Type | No. | Specification |
|---|---|---|
| Controller Node | 3 | Dell R630<br>Dual Xeon E5-2643v3 3.4GhZ, 6 core<br>128GB RAM<br>2.4TB local SSD<br>50GbE Mellanox ConnectX-4 LX<br>10GbE Intel<br>1GbE Intel |
| Compute Node | 80 | Dell C6320<br>Dual Xeon E5-2680v3 2.5GhZ, 12 core<br>256GB RAM<br>1.6TB local SSD<br>50GbE ConnectX-4 LX<br>10GbE Intel<br>1GbE Intel |
| Network | 5<br><br>5<br><br>6 | Mellanox Spectrum SN2700<br>32x 100GbE ports, split 2x50GbE<br><br>Mellanox Spectrum SN2410<br>48x 10GbE ports + 8x 100GbE uplink<br><br>Dell N2048 PowerConnect<br>48x 1GbE ports |

**OpenStack Distro:**
Red Hat OSP8 (Liberty)

**Cinder Storage:**
Small Ceph pool (64TB)
NexentaStor iSCSI (1PB)

# OpenStack Networking



# OpenStack Compute and Control Plane Networking

NIC Hardware:

- 50GbE Mellanox ConnectX4-LX: Storage networks, Tenant networks (VLAN + VXLAN)
- 10GbE Network: Data Centre Provider and External VLANs
- 1GbE Network: Hardware provisioning, IPMI and management/monitoring

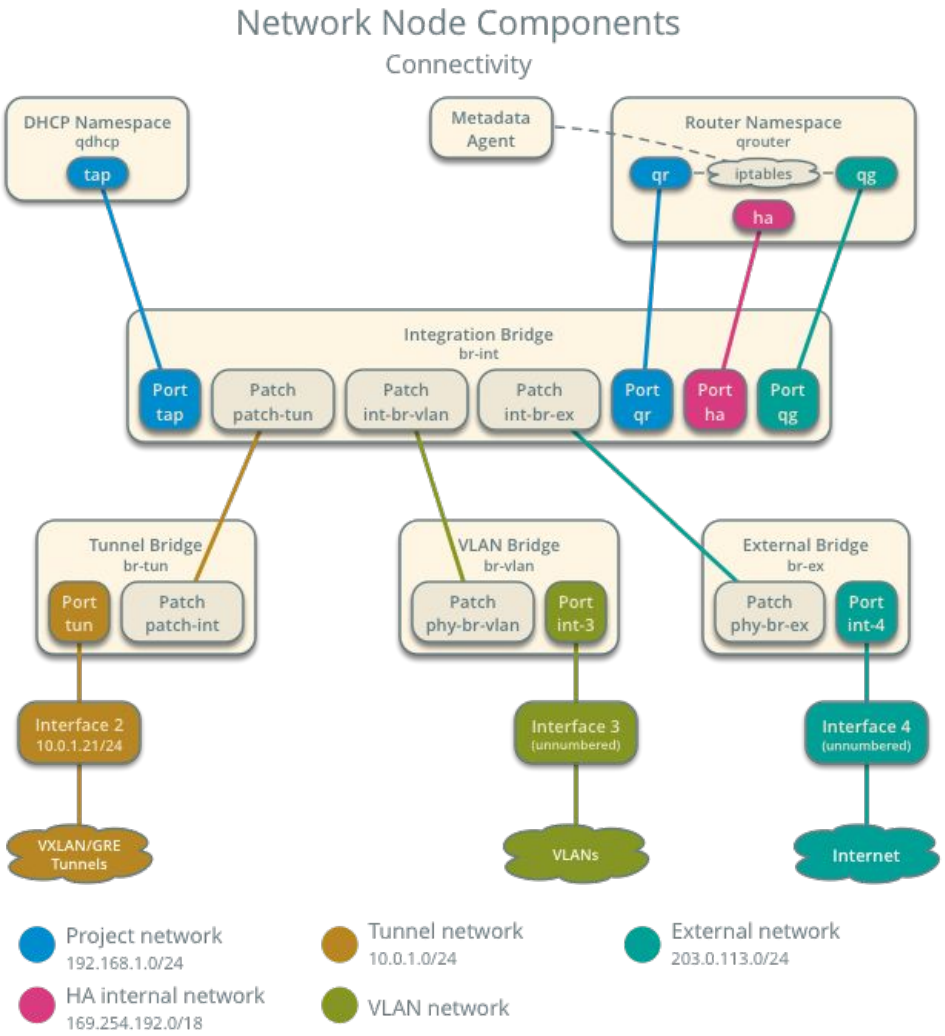
Tenant Network Isolation:

- Limit of 4096 VLAN IDs
- Prefer these for “provider”/data centre VLANs made directly available to hypervisors.

- Use VXLAN encap. for most tenant networks
- Some issues were seen with VXLAN UDP offloads, affecting achievable B/W
- Fixed with RHEL 7.3 kernel backports
- VXLAN network identifier is 24 bit

=> 16 million VNIDs

# North-South Neutron Traffic




HA Neutron L3 routing spreads routing agents across controllers rather than to compute.

L3 agents exist, namespaced for isolation, on the controllers and handle NAT'ed traffic to tenant networks from one or more external networks.

L3 agents fail-over on failure, via Linux VRRP

Distributed Virtual Routing on computes is possible, but requires each compute node be exposed on the external network(s).

OpenVSwitch implementation uses successive OVS bridges and finally a Linux bridge leading to the instance vNIC.

Instead of NAT through the controllers, we can distribute the CUDN as another provider network to the compute node and instance.