Best Practices for OpenStack Private Clouds

Florian Haas <florian@hastexo.com>

OpenStack Folsom, November 2012

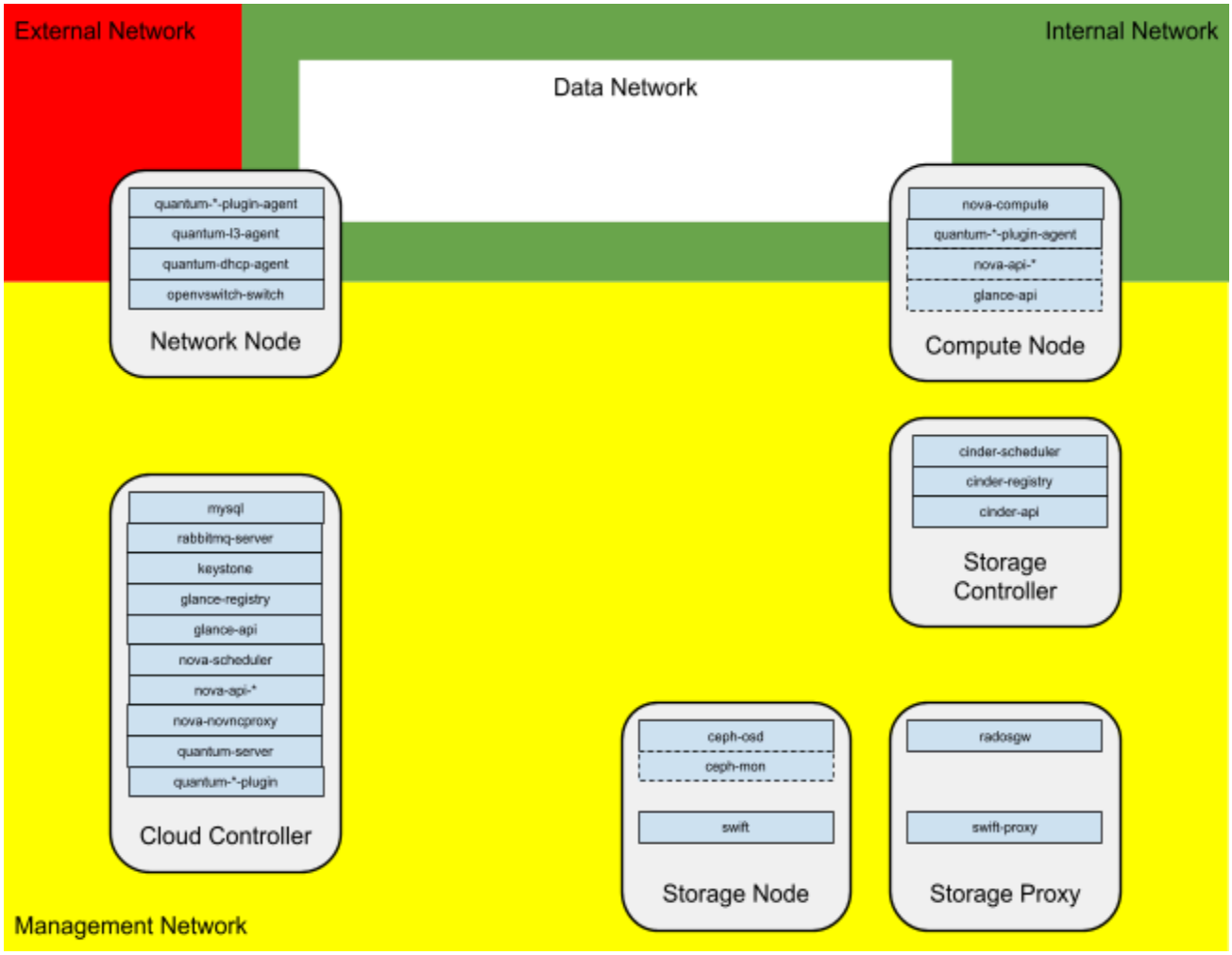
Caveat Lector

If you've found this document from a Google+ post, or otherwise on the web, please be advised that until this notice is removed, this is a work in progress, a draft, or whatever else you would like to call it. OpenStackers are encouraged to comment on this liberally, call the author an idiot, suggest completely different approaches that the ones presented here, etc.

Assumptions

This document assumes that you want to build a private cloud of a minimum of 4 nodes and a maximum of maybe fifty or a hundred. It also assumes that you are planning for diverse, heterogeneous workloads in multiple OpenStack tenants. Finally, it assumes that you're going to want high availability and that a significant portion of your workloads are expected to work with persistent, non-throwaway data.

Also, this document really wouldn't make much sense to you unless you have at least some familiarity with OpenStack and its component services.



Network Architecture

This is for a Quantum/OpenVSwitch based architecture.

Plan for a total of 4 physical subnets:

- Your management network. This is where your physical infrastructure has its primary addresses. You shell into cloud controllers on this network. If you're using Ceph storage, this is also where your Ceph MONs and OSDs have their network addresses. All hosts in the cloud infrastructure need an Ethernet interface in the management network.
- Your internal network. This is the separate network that OpenVswitch and Quantum use to maintain their GRE connections in which they encapsulate virtually-switched traffic. All compute nodes (running nova-compute and the Quantum OVS agent) and all network hosts (running OpenVSwitch and the Quantum OVS plugin) need an interface on the internal network.
- Your external network. This is how you access virtual machines from the outside

internet. Quantum maintains “floating” (public) IP addresses in that external network, and assigns them to your machines. Network hosts (running the Quantum L3 and DHCP agents) need an interface on the external network. You most likely will subnet this external network on a per-tenant basis.

- Your data network. This is the network that your Nova guests get their IP addresses from. This network exists *only* with the OVS managed GRE tunnels, you do not need to plan for it in your hardware network infrastructure. This network will also be subnetted per tenant.
- Your API network. If you are planning to expose the ReSTful APIs on your public network, this is the network these APIs will live on.

Node Roles

Plan for a minimum of 4 node roles:

- Cloud Controller: runs MySQL, RabbitMQ, keystone, glance-registry and optionally glance-api, quantum-server, nova-scheduler, nova-novncproxy, nova-api-ec2, nova-api-os-compute and optionally nova-api-metadata.
    - Often only one, definitely *at least* one. May easily be made HA with Pacemaker.
    - If you want to spread out services, you can have several nodes instead of a single Cloud Controller, but the HA configuration then gets a little more involved.
    - ØMQ is an excellent alternative for RabbitMQ for HA purposes, but as of Folsom it is still suffering from interoperability issues with Quantum.
- Network Node: runs openvswitch-switch, quantum-ovs-plugin, plugin-l3-agent and plugin-dhcp-agent.
    - At least one. Possibly more if one can’t handle the routing load. Needs HA.
- Storage Controller: runs cinder-scheduler, cinder-registry and cinder-api
    - At least one. Several more can run cinder-api. If we’re using Cinder with Ceph, this is only useful to scale the load on the API itself. If we’re using Cinder with LVM and iSCSI, multiple cinder-api hosts are the only way to scale out storage. The LVM+iSCSI approach absolutely needs HA (must also fail over the cinder-volumes group).
- Compute Node: runs nova-compute, quantum-ovs-agent. May also run a local glance-api if you want (and have the storage capacity for) local image caching, and the nova-api-* services so local nova-compute can talk to localhost for API calls.
    - At least one. As many as we want and need.

An optional node role, not directly related to OpenStack but still rather recommended for private clouds, is that of a Ceph storage node. For high availability purposes you’ll need 3, and then as many more as your scalability needs require.

Name Resolution

Define a DNS subdomain for your cloud in a domain you own, such as cloud.example.com. Define an internal and and external view on your nameservers for this subdomain, such that *service*.cloud.example.com resolves to a management network address when queried internally, and a publicly accessible API network address when queried externally.

Make the DNS nameservers redundant, both on the internal and the external side. On the management network, also make sure that the DNS sits within the management network and not behind a router (else the router becomes a point of failure).

Service Endpoints

Bind every service that is registered in Keystone to its separate IP address. Register all of them in DNS.

- For API endpoints that run only once in your cloud, name them glance.cloud.example.com, quantum.cloud.example.com, etc.
- For API endpoints that run multiple times, name them nova1.cloud.example.com, nova2.cloud.example.com, cinder1.cloud.example.com, cinder2.cloud.example.com, etc. Or, put them behind an IP load balancer or round-robin DNS and have services talk to them via the non-numbered service name.
- In the internal view of your DNS, register the endpoint's host names with addresses on your management network. In the external DNS, assign them addresses on your API network. Configure the services to listen on the wildcard address (0.0.0.0) on their respective hosts.

System Automation and Deployment

Use *something* to automate your deployment. Whether it's Chef, Puppet, Juju, or Crowbar is up to you. Pick based on what you think fits best. But don't deploy manually.