Deploy OpenStack

Using Kolla-Ansible

The Plan for today…

Short recap of last lectures (5 mins)

What’s OpenStack and what do you need to deploy it (30 mins)

How to deploy OpenStack with kolla-ansible (20 mins)

How to configure your OpenStack deployment (40 mins)

Releases & Upgrades (5 mins)

Recap

Data center overview

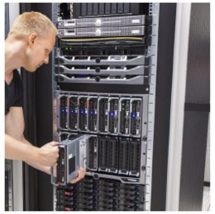
Node deployment

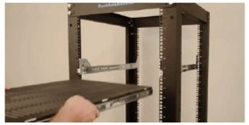
Ceph Storage

Recap – data center overview

Data Center Servers

- Servers (Compute Resource)
  - Rack-mount servers



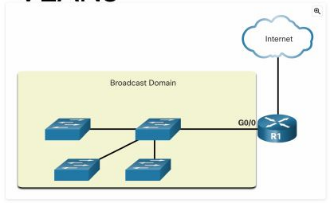


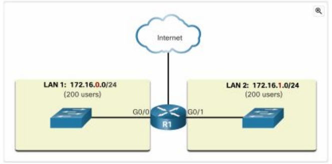
Recap – data center overview

Data Center Servers

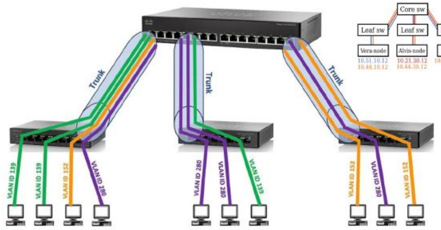
- Servers (Compute Resource)
  - Rack-mount servers

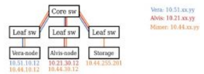
VLANs



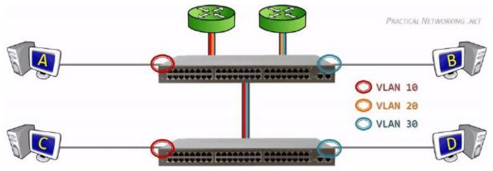


CHALMERS UNIVERSITY OF TECHNOLOGY | E-COMMONS





Traffic arriving on a switch port assigned to one VLAN will only ever be forwarded out another switch port that belongs to the same VLAN – **a switch will never allow traffic to cross a VLAN boundary**.

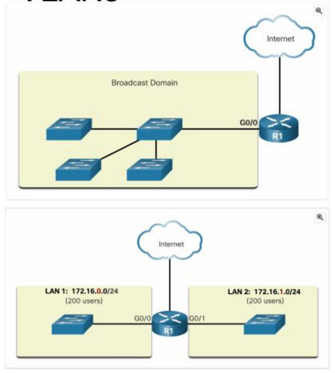


Recap – data center overview

Data Center Servers

- Servers (Compute Resource)
  - Rack-mount servers

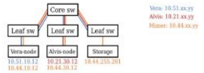
VLANs



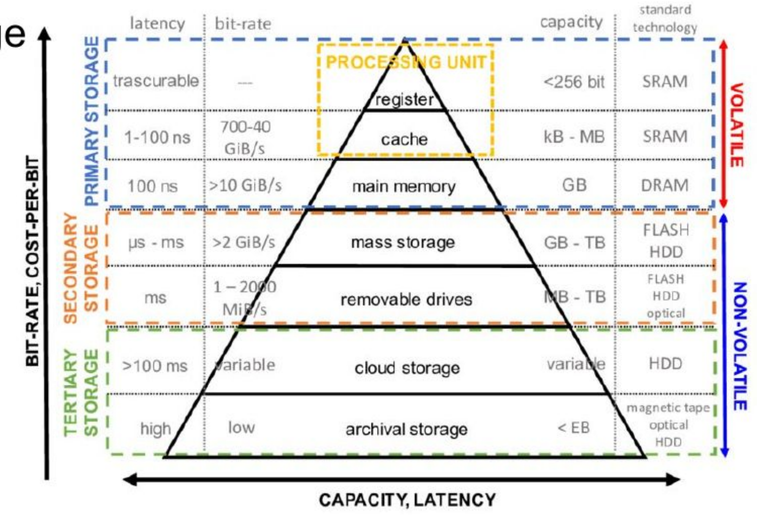
CHALMERS
UNIVERSITY OF TECHNOLOGY
E-COMMONS

Traffic a
forwarde
will nev

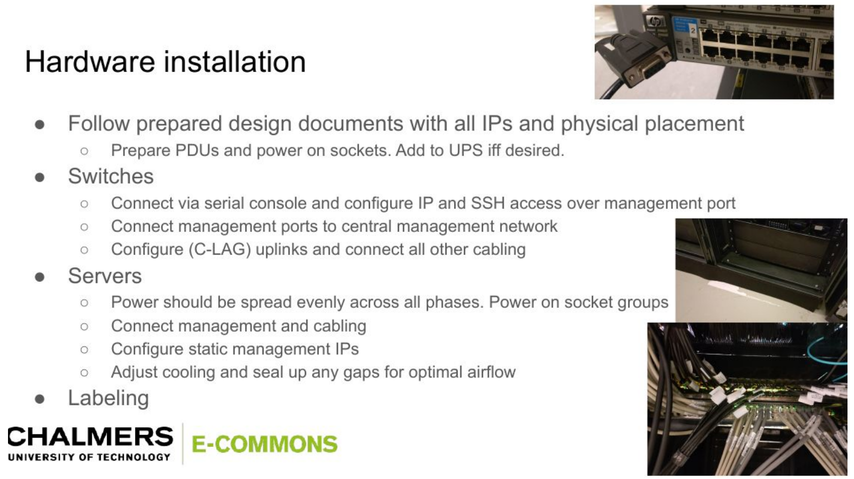
Trunk
Trunk
Trunk



Data Center Storage



Recap – node deployment



Hardware installation

- Follow prepared design documents with all IPs and physical placement
  - Prepare PDUs and power on sockets. Add to UPS iff desired.
- Switches
  - Connect via serial console and configure IP and SSH access over management port
  - Connect management ports to central management network
  - Configure (C-LAG) uplinks and connect all other cabling
- Servers
  - Power should be spread evenly across all phases. Power on socket groups
  - Connect management and cabling
  - Configure static management IPs
  - Adjust cooling and seal up any gaps for optimal airflow
- Labeling

CHALMERS UNIVERSITY OF TECHNOLOGY | E-COMMONS

Recap – node deployment

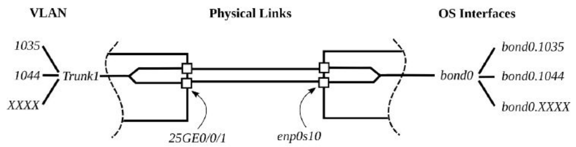
Hardware installation

- Follow prepared design documents with all IPs and physical placement
  - Prepare PDUs and power on sockets. Add to UPS iff desired.
- Switches
  - Connect via serial console and configure IP and SSH access over management port
  - Connect management ports to central management network

:abling

;. Power on socket groups

irflow



Switch config

```
#
interface 25GE1/0/1
 eth-trunk 1
 device transceiver 25GBASE-COPPER
#
interface 25GE1/0/2
 eth-trunk 1
 device transceiver 25GBASE-COPPER
#
interface Eth-Trunk1
 port link-type hybrid
 port hybrid pvid vlan 1035
 port hybrid tagged vlan 1033 1043 to 1044
 port hybrid untagged vlan 1035
 mode lacp-dynamic
```

Ansible (after we have the OS)

```
- name: "bond0: setup bond"
  nmcli:
    type: bond
    conn_name: bond0
    autoconnect: yes
    state: present

- name: "bond0: add bond-slaves"
  nmcli:
    type: bond-slave
    conn_name: '{{ item }}.bond0'
    ifname: '{{ item }}'
    master: bond0
    state: present
  with_items:
    - 'ens801f0np0'
    - 'ens801f0np1'

- name: "bond0: add bond-vlans"
......
```

CHALMERS
UNIVERSITY OF TECHNOLOGY | E-COMMONS

Recap – node deployment

Hardware installation

- Follow prepared design documents with all IPs and physical placement
  - Prepare PDUs and power on sockets. Add to UPS iff desired.
- Switches
  - Connect via serial console and configure IP and SSH access over management port
  - Connect management ports to central management network

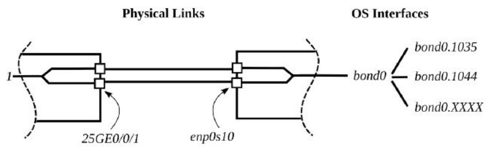
:abling

;. Power on socket groups

irflow

PXE - DHCP

- Machines set to boot with PXE first
- UEFI/BIOS will request DHCP for every interface
  - Slows down booting: disable everything we can in UEFI keeping only IPv4 PXE boot on the correct interface if possible
- Monitor DHCP requests on Janne
  - systemctl status dnsmasq -n 99
  - `May 08 08:13:23 janne dnsmasq-dhcp[3043581]: DHCPREQUEST(eth11) 10.44.255.2 bc:24:11:57:c5:11`
- No requests coming through? Debugging during PXE boot is a pain in the ass
  - Just manually deploy a node, log in via conman and bring up network manually to test things out. Usually missed something some switch configuration, like VLAN or lacp-force-up



```
)PPER

)PPER

:id
:an 1035
vlan 1033 1043 to 1044
:d vlan 1035
```

E-COMMONS

Ansible (after we have the OS)

```
- name: "bond0: setup bond"
  nmcli:
    type: bond
    conn_name: bond0
    autoconnect: yes
    state: present

- name: "bond0: add bond-slaves"
  nmcli:
    type: bond-slave
    conn_name: '{{ item }}.bond0'
    ifname: '{{ item }}'
    master: bond0
    state: present
  with_items:
    - 'ens801f0np0'
    - 'ens801f0np1'

- name: "bond0: add bond-vlans"
......
```

Ansible - generally applicable configurations:

- Networking, static IPs, bond, VLANs
- Hostname, timezone
- Network tuning + tuned - Larger buffer sizes and such
- arp-cache - OS defaults are quite limited
- LLDP - essential for network discovery and debugging
- Icinga health checks, mcelog
- Firewall
- Rsyslog
- (TSM backups)

Recap – node deployment

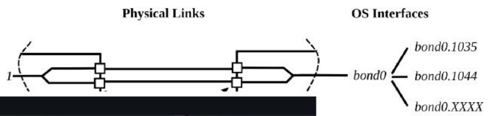
Hardware installation

- Follow prepared design documents with all IPs and physical placement
  - Prepare PDUs and power on sockets. Add to UPS iff desired.
- Switches
  - Connect via serial console and configure IP and SSH access over management port
  - Connect management ports to central management network

:abling

:. Power on socket groups

irflow



fter we have the OS)

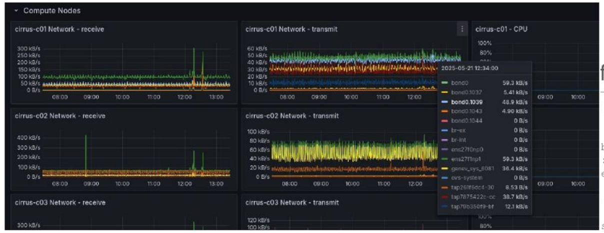
```
 setup bond"


bond0
: yes
ent

 add bond-slaves"

slave
```

PXE - DHCP

- Machines set to boot with PXE first
- UEFI/BIOS will request DHCP for every interface
  - Slows down booting: disable correct interface if possible
- Monitor DHCP requests on
  - systemctl status dnsmasq -n
  - `May 08 08:13:23 janne dnsma`
    `bc:24:11:57:c5:11`
- No requests coming throug
  - Just manually deploy a node, out. Usually missed somethin

Metrics

1. Collect metrics
2. ???
3. Profit!


Compute Nodes
cirrus-c01 Network - receive
cirrus-c01 Network - transmit
cirrus-c01 - CPU
cirrus-c02 Network - receive
cirrus-c02 Network - transmit
cirrus-c03 Network - receive
cirrus-c03 Network - transmit

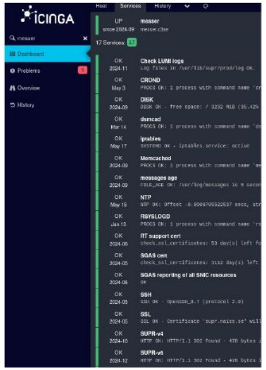
```
2025-05-21 12:34:00
bond0              59.3 kB/s
bond0.1037         5.41 kB/s
bond0.1039         48.9 kB/s
bond0.1043         4.90 kB/s
bond0.1044            0 B/s
br-ex                 0 B/s
br-int                0 B/s
ens27f0np0            0 B/s
ens27f1np1         59.3 kB/s
genev_sys_6081     36.4 kB/s
ovs-system            0 B/s
tap261f6dc4-30     8.53 B/s
tap7875422c-cc     38.7 kB/s
tap79b350f9-bf     12.1 kB/s
```

Ansible - generally a

- Networking, static IPs, b
- Hostname, timezone
- Network tuning + tuned
- arp-cache - OS defaults are quite limited
- LLDP - essential for network discovery and debugging
- Icinga health checks, mcelog
- Firewall
- Rsyslog
- (TSM backups)

Health checks

- Icinga for health checks
  - Ping
  - SSH
  - Probe management for general hardware health events, fans, cooling, temperatures
  - Disk checks
  - systemd unit failures
  - Firewall
  - Updates
  - Certificates
  - NTP
  - Application specific scripts for health checks, e.g. ceph status
- Set alerts in chat if deemed urgent



```
UP since 2024-09   messer
                   messer-c3se
17 Services: 17

OK 2024-11  Check LUMI logs
            Log files in /var/lib/supr/prod/log OK.
OK May 3    CROND
            PROCS OK: 1 process with command name 'cr
OK 2024-09  DISK
            DISK OK - free space: / 3232 MiB (35.42%
OK Mar 14   dsmcad
            PROCS OK: 1 process with command name 'ds
OK May 17   iptables
            SYSTEMD OK - iptables.service: active
OK 2024-09  Memcached
            PROCS OK: 1 process with command name 'me
OK 2024-09  messages age
            FILE_AGE OK: /var/log/messages is 5 secon
OK May 15   NTP
            NTP OK: Offset -0.0000705522637 secs, str
OK Jan 13   RSYSLOGD
            PROCS OK: 1 process with command name 'rs
OK 2024-06  RT support cert
            check_ssl_certificates: 53 day(s) left fo
OK 2024-05  SGAS cert
            check_ssl_certificates: 3112 day(s) left
OK 2024-06  SGAS reporting of all SNIC resources
            OK
OK 2024-05  SSH
            SSH OK - OpenSSH_8.7 (protocol 2.0)
OK 2024-05  SSL
            SSL OK - Certificate 'supr.naiss.se' will
OK 2024-10  SUPR-v4
            HTTP OK: HTTP/1.1 302 Found - 478 bytes i
OK 2024-12  SUPR-v6
            HTTP OK: HTTP/1.1 302 Found - 478 bytes i
```

Recap – Ceph storage

Ceph is a **distributed storage system** designed to provide excellent performance, reliability, and scalability.

Ceph is often used for **object storage, block storage, and file systems**.

Recap – Ceph storage

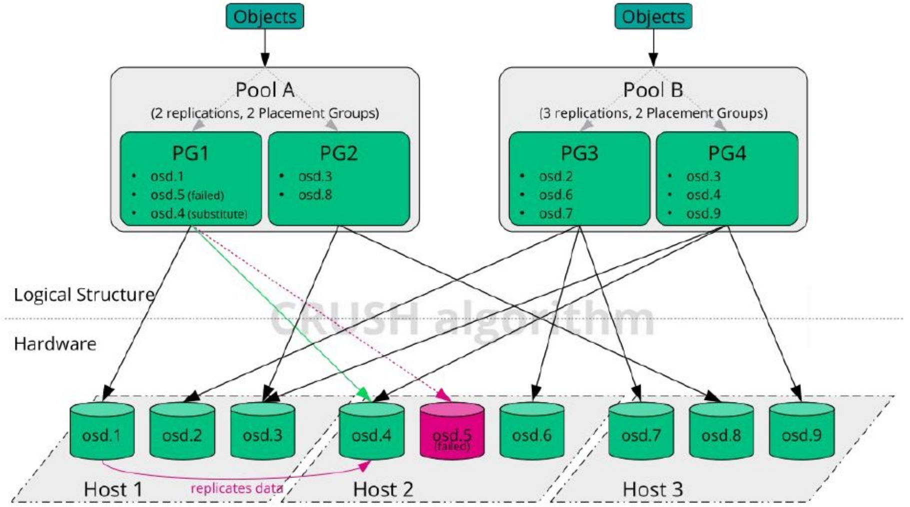
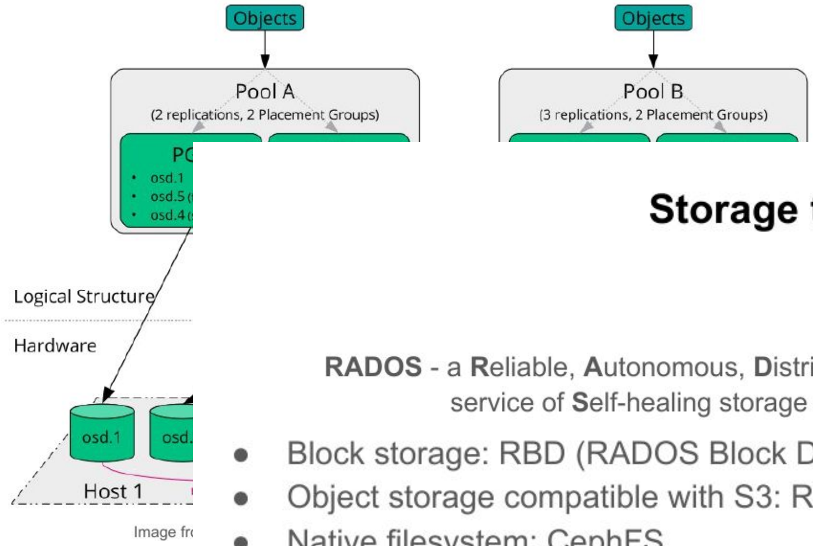


Image from SUSE: https://documentation.suse.com/ses/7/html/ses-all/images/data-structure-example.png

Recap – Ceph storage



Storage types provided

RADOS - a **R**eliable, **A**utonomous, **D**istributed, **O**bject-storage service of **S**elf-healing storage nodes

- Block storage: RBD (RADOS Block Device)
- Object storage compatible with S3: RGW (RADOS GateWay)
- Native filesystem: CephFS
- CephFS can also be exported as NFS (uses Ganesha)
- An exporter for SMB is in development

Recap – Ceph storage

Mimer-flash

- 12 Servers
  - 2 x 16 core Intel Xeon 6326, 384 GB memory
  - 10x 14TB NVMe
  - MDS spread out but together with OSD
  - 2x 100Gbit Ethernet for client network
  - 2x 100Gbit Ethernet for cluster network
- 2x 100Gbit-switch, each with dual 100Gbit uplinks
- Part of Mimer-ceph, i.e. shares MGR, MON etc.

ed

e

eWay)

- CephFS can also be exported as NFS (uses Ganesha)
- An exporter for SMB is in development

Questions?

Next up:
What’s OpenStack and what do you need to deploy it

The purpose of a cloud (from an astronomer’s perspective)

- Operating lots of compute and storage resources in one place makes a lot of sense from an economic, energetic, and organisational perspective

- But once we have all these resources (loads of CPUs, GPUs, RAM, Storage):
  - How do we make them accessible to the user in a user-friendly way?
  - How do we accommodate different users’ needs?
  - How can we stay flexible in terms of what services we provide?

- Via cloud architecture we can “chop up” the hardware in differently sized bits to accommodate everything from small single-core “servers” via heavy fat nodes to small networks of servers up to kubernetes and slurm clusters.

openstack®

OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed and provisioned through APIs with common authentication mechanisms.

Components of OpenStack – Core components

- **Glance** – Images
  - Provides images of operating systems for VMs
  - Also used for backups of VMs

- **Nova** – Compute
  - “Runs” the OS, takes care of CPUs, RAM, GPUs
  - Also “provides” ephemeral storage

- **Neutron** – Network
  - Handles all things related to networks inside OpenStack

- **Horizon** – Web frontend
  - User facing frontend to spin up VMs, configure storage, networks, …

- **Heat** – Orchestration
  - Integrates the different services into one
  - The “glue” between the different services

- **Keystone** – Authentication
  - Authentication across different APIs in the system

Components of OpenStack – Required for smooth operations

- **Rabbitmq** – Message broker
  - Asynchronous communication between services
- **Mariadb** – Database
  - Keeps track of VMs, Volumes, Networks, IPs,...
- **Haproxy** – Loadbalancer
  - Spreading the load across nodes in the control plane

Components of OpenStack – Other services we enabled

- **Cinder** – Block storage
  - Provides storage volumes that can be mounted into VMs
  - Several different types of storage backend configurable
    - E.g. Ceph, LVM, NFS…
- **Manila** – Shared storage
  - Can use the same and/or different storage backend than Cinder
- **Octavia** – Loadbalancing
  - Used in VM-networks
  - Choice of using OVN and/or Amphora drivers
    - Amphora needs more resources and is harder to set up – but is more versatile
    - OVN easy to enable, low resource needs, good enough for most applications
- **Barbican** – Key management
  - Management of secret data, including keys and certificates
- **Letsencrypt** – Certificate service
  - Keeps our certificates up to date for cloud.swesrc.chalmers.se
- **Fluentd** – Log parsing
  - Collects logs from services and forwards them to a common place for easier debugging
  - We don’t actually use that yet…

Deployment of OpenStack – Prerequisites

Deployment (seed) node

- Used to deploy, maintain and run OpenStack

Simple VM in Proxmox

Deployment of OpenStack – Prerequisites

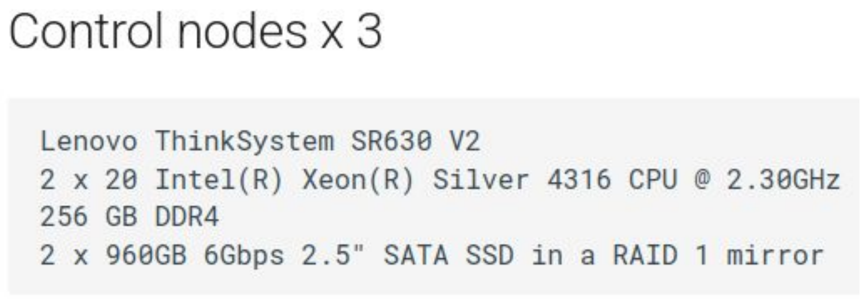
Deployment (seed) node

- Used to deploy, maintain and run OpenStack

Control Plane Node(s)

- They're the "brain" of the cloud
- Either 1, 3, or 5
  - More than 1 for high availability
  - Uneven number to reach quorum
- We operate 3 (cirrus-s[1,2,3])

Simple VM in Proxmox



Control nodes x 3

```
Lenovo ThinkSystem SR630 V2
2 x 20 Intel(R) Xeon(R) Silver 4316 CPU @ 2.30GHz
256 GB DDR4
2 x 960GB 6Gbps 2.5" SATA SSD in a RAID 1 mirror
```

Deployment of OpenStack – Prerequisites

Deployment (seed) node

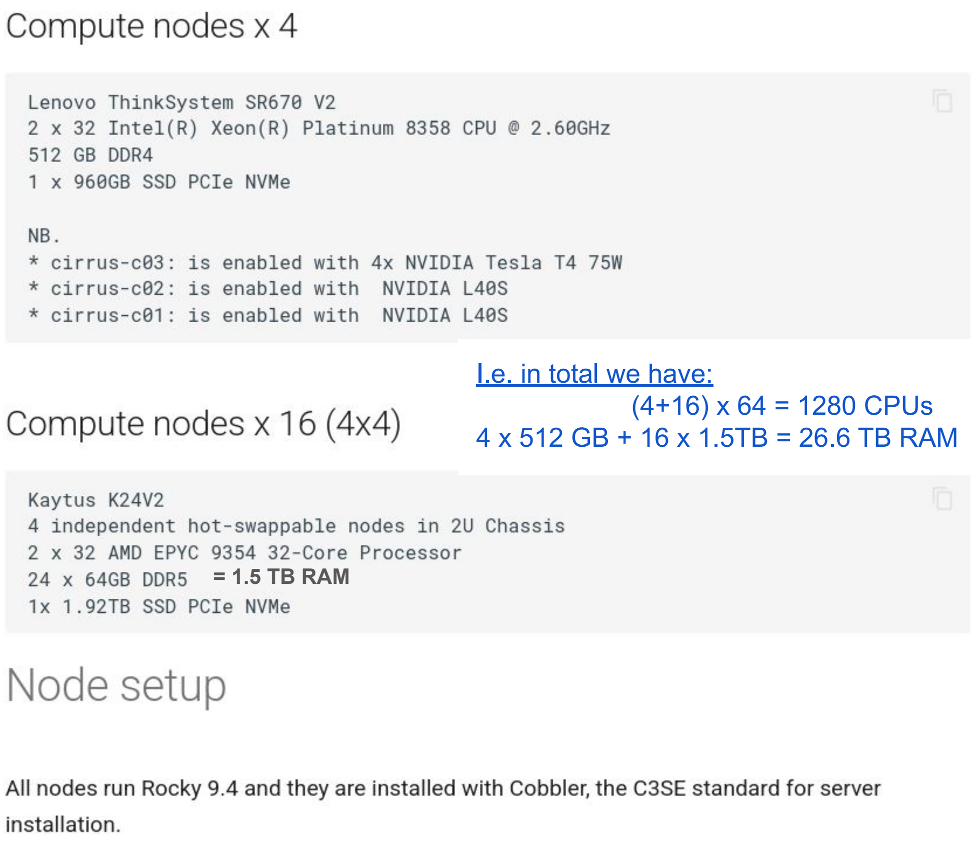
- Used to deploy, maintain and run OpenStack

Control Plane Node(s)

- They're the "brain" of the cloud
- Either 1, 3, or 5
  - More than 1 for high availability
  - Uneven number to reach quorum
- We operate 3 (cirrus-s[1,2,3])

Compute Node(s)

- This is where VMs run
- We operate 19 (cirrus-c[01,19])



Compute nodes x 4

```
Lenovo ThinkSystem SR670 V2
2 x 32 Intel(R) Xeon(R) Platinum 8358 CPU @ 2.60GHz
512 GB DDR4
1 x 960GB SSD PCIe NVMe

NB.
* cirrus-c03: is enabled with 4x NVIDIA Tesla T4 75W
* cirrus-c02: is enabled with  NVIDIA L40S
* cirrus-c01: is enabled with  NVIDIA L40S
```

Compute nodes x 16 (4x4)

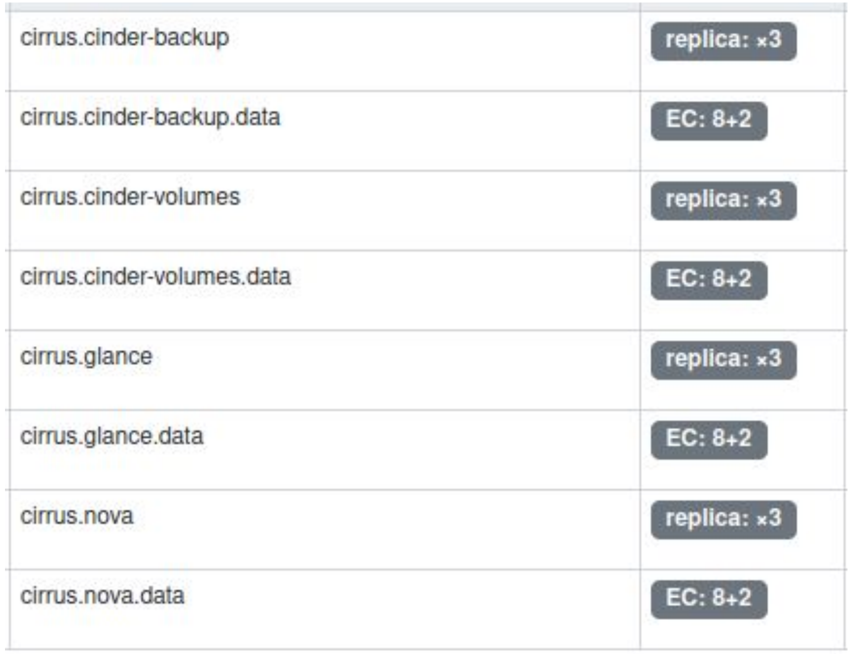
I.e. in total we have:
(4+16) x 64 = 1280 CPUs
4 x 512 GB + 16 x 1.5TB = 26.6 TB RAM

```
Kaytus K24V2
4 independent hot-swappable nodes in 2U Chassis
2 x 32 AMD EPYC 9354 32-Core Processor
24 x 64GB DDR5   = 1.5 TB RAM
1x 1.92TB SSD PCIe NVMe
```

Node setup

All nodes run Rocky 9.4 and they are installed with Cobbler, the C3SE standard for server installation.

Deployment of OpenStack – Prerequisites



cirrus.cinder-backup	replica: ×3
cirrus.cinder-backup.data	EC: 8+2
cirrus.cinder-volumes	replica: ×3
cirrus.cinder-volumes.data	EC: 8+2
cirrus.glance	replica: ×3
cirrus.glance.data	EC: 8+2
cirrus.nova	replica: ×3
cirrus.nova.data	EC: 8+2

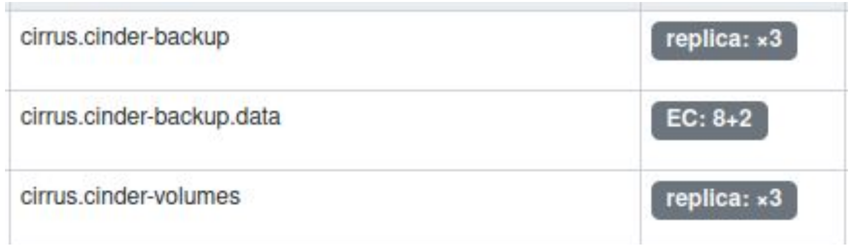
Ceph Storage

- Need to set up various Ceph pools:
  - Glance pool (for images)
  - Nova compute (for the OS; i.e. “/” is here)
  - Cinder volumes + Cinder backup (storage)

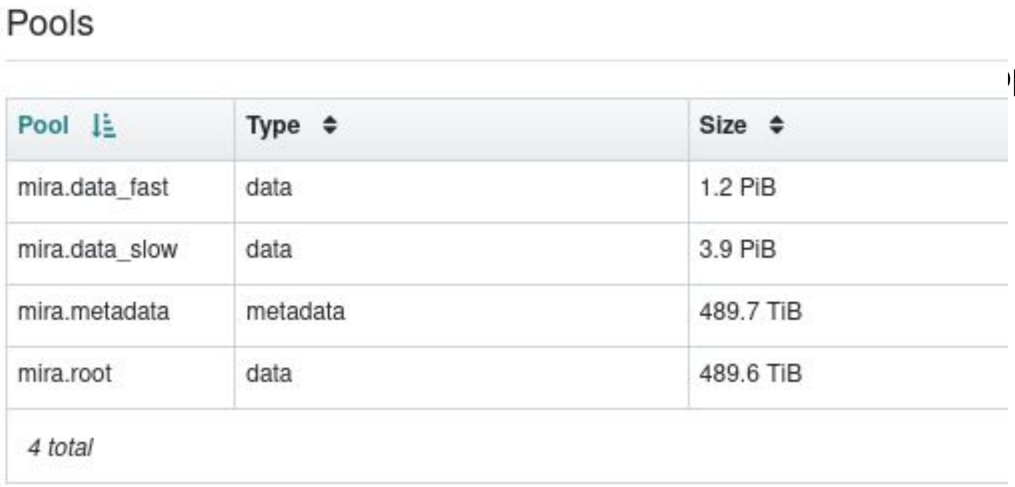
  optional:

  - Nova ephemeral (storage that is tied to the VM)
  - CephFS (shared storage)

Deployment of OpenStack – Prerequisites



cirrus.cinder-backup	replica: ×3
cirrus.cinder-backup.data	EC: 8+2
cirrus.cinder-volumes	replica: ×3



Pools

Pool	Type	Size
mira.data_fast	data	1.2 PiB
mira.data_slow	data	3.9 PiB
mira.metadata	metadata	489.7 TiB
mira.root	data	489.6 TiB

4 total

Ceph Storage

- Need to set up various Ceph pools:
  - Glance pool (for images)
  - Nova compute (for the OS; i.e. "/" is here)
  - Cinder volumes + Cinder backup (storage)

ptional:

  - Nova ephemeral (storage that is tied to the VM)
  - CephFS (shared storage)

Deployment of OpenStack – Prerequisites

Deployment (seed) node

- Used to deploy, maintain and run OpenStack

Control Plane Node(s)

- They're the "brain" of the cloud
- Either 1, 3, or 5
  - More than 1 for high availability
  - Uneven number to reach quorum
- We operate 3 (cirrus-s[1,2,3])

Compute Node(s)

- This is where VMs run
- We operate 19 (cirrus-c[01,19])

Ceph Storage

- Need to set up various Ceph pools:
  - Glance pool (for images)
  - Nova compute (for the OS; i.e. "/" is here)
  - Cinder volumes + Cinder backup (storage)

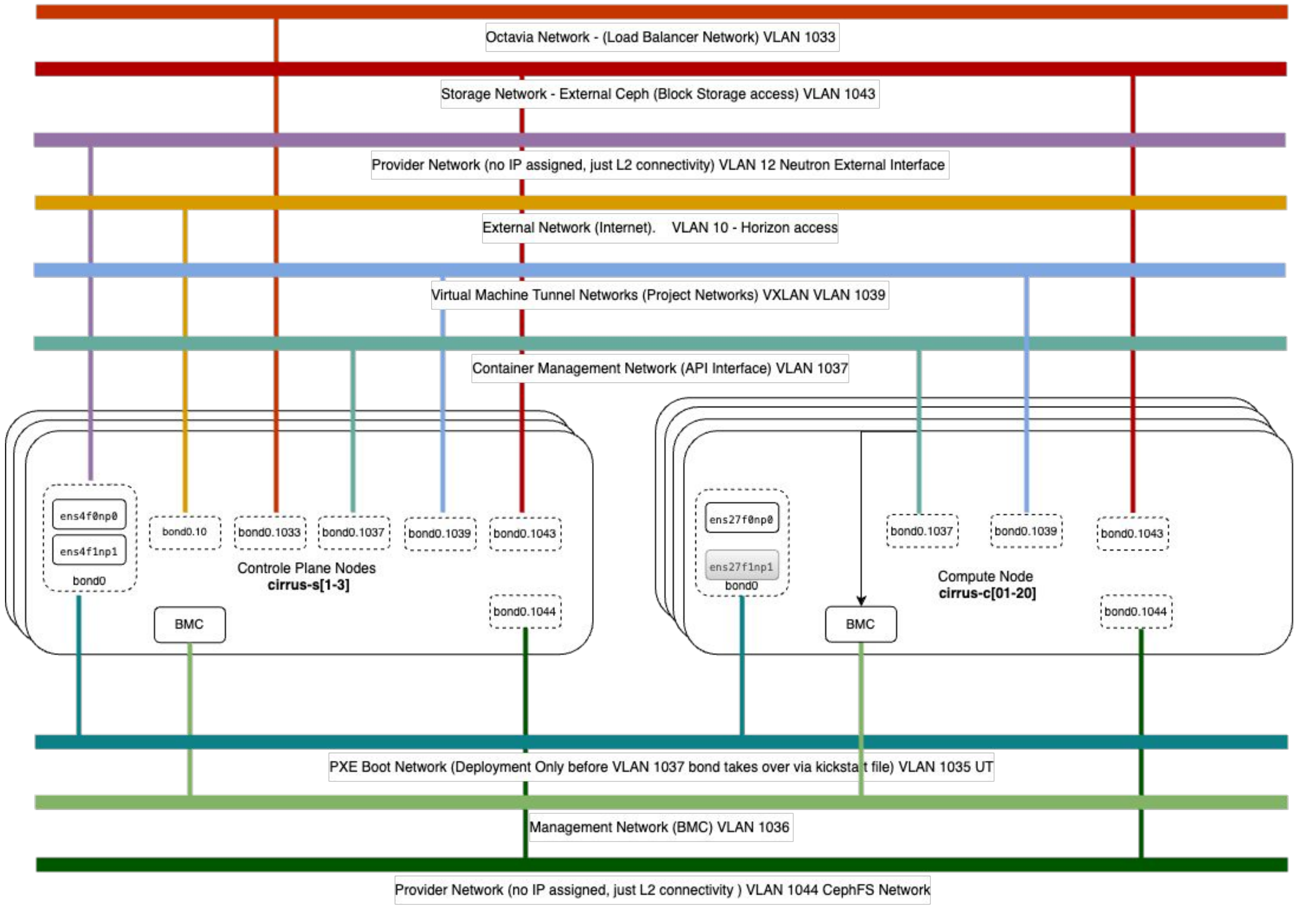
  optional:

  - Nova ephemeral (storage that is tied to the VM)
  - CephFS (shared storage)

Networking

- Connection to Ceph cluster
- Connection to WWW
- Internal Connection

Deployment of OpenStack – Network Setup



Octavia Network - (Load Balancer Network) VLAN 1033
Storage Network - External Ceph (Block Storage access) VLAN 1043
Provider Network (no IP assigned, just L2 connectivity) VLAN 12 Neutron External Interface
External Network (Internet). VLAN 10 - Horizon access
Virtual Machine Tunnel Networks (Project Networks) VXLAN VLAN 1039
Container Management Network (API Interface) VLAN 1037
ens4f0np0
ens4f1np1
bond0
bond0.10
bond0.1033
bond0.1037
bond0.1039
bond0.1043
Controle Plane Nodes
cirrus-s[1-3]
bond0.1044
BMC
ens27f0np0
ens27f1np1
bond0
bond0.1037
bond0.1039
bond0.1043
Compute Node
cirrus-c[01-20]
bond0.1044
BMC
PXE Boot Network (Deployment Only before VLAN 1037 bond takes over via kickstart file) VLAN 1035 UT
Management Network (BMC) VLAN 1036
Provider Network (no IP assigned, just L2 connectivity) VLAN 1044 CephFS Network

Deployment of OpenStack – Network Setup

What we need for OpenStack

- **External Public Network (VLAN 10)**
  - Connects OpenStack to the outside world
  - Used for: Public APIs, Horizon
  - Nodes connected: s-nodes (VIP)
- **Management Network (VLAN 1037)**
  - Internal communication between OpenStack services and SSH access to nodes
  - Used for: Ansible deployment, API communication between services, Control plane traffic
  - Nodes connected: s-nodes (control nodes, network and storage nodes) and c-nodes
- **External (Public / Provider) Network (VLAN 12)**
  - Connects OpenStack VMs to the outside world
  - Used by: Floating IPs, External tenant access
  - Nodes connected: s-nodes (Network nodes)
  - VLAN provider Network is exposed to tenants (physnet1)

Deployment of OpenStack – Network Setup

What is what and why we need it

- **Tenant (Overlay / Tunnel) Network (VLAN 1039)**
  - Traffic between VM instances across compute nodes
  - Neutron overlay networking , east-west VM communication
  - Encapsulated with VXLAN (before), **Geneve (now)**
  - Nodes connected: s-nodes (Network nodes) and c-nodes
- **Storage Network (Ceph Network) (VLAN 1043)**
  - Communication between OpenStack services and the external Ceph cluster
  - Delivers RBD traffic : Cinder volumes, Glance images, Nova ephemeral disks
  - Nodes connected: s-nodes, c-nodes, Ceph MONs/OSDs
- **Storage (CephFS Provider) Network (VLAN 1044)**
  - Connects OpenStack VMs to CephFS
  - Used for CephFS tenant access
  - Routed to 1043 for mounting CephFS Shares
  - FLAT provider Network is exposed to tenants (physnet2)
- **Octavia Management Network (note: now via OVN Provider ie. no need for VLAN 1033)**
  - Internal communication between the Octavia control plane and the load balancer instances (called **Amphora**).
  - This network is **not visible to tenants** and is used only by the infrastructure.
  - Nodes connected : Controller nodes, Network nodes, Compute nodes (where amphora VMs run)

Deployment of OpenStack – Network Setup

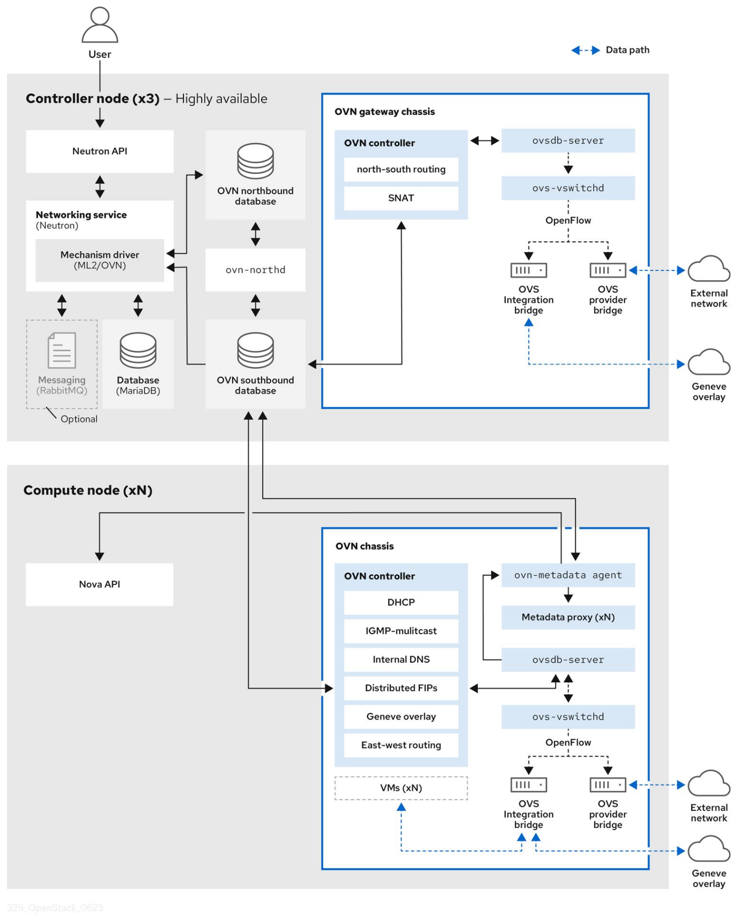
In **OpenStack deployments with Kolla-Ansible**, **Open Virtual Network (OVN)** together with **Open vSwitch (OVS)** is used as the **network backend for OpenStack Neutron**.

It replaces the older Neutron architecture that used multiple agents (L3 agent, DHCP agent, etc.).

In short:

OVS → dataplane (packet forwarding on hosts)
OVN → control plane (programs networking rules into OVS) north-south routing

In simple words:
OVN decides how networking should work, and **OVS** actually moves the packets.



Questions?

If not: time to breathe.

Next up:
How to deploy OpenStack with kolla-ansible

Deployment with Kolla-Ansible

“What is Kolla-Ansible?”

AI-reply (Duck-AI):

“Kolla Ansible is a tool that simplifies the deployment of OpenStack clouds using **Docker containers** and **Ansible automation**. It allows users to quickly set up OpenStack services while providing options for customization as they gain more experience.”

Deployment with Kolla-Ansible

KOLLA
an OpenStack Community Project

“What is Kolla-Ansible?”

AI-reply (Duck-AI):

“Kolla Ansible is a tool that simplifies the deployment of OpenStack clouds using **Docker containers** and **Ansible automation**. It allows users to quickly set up OpenStack services while providing options for customization as they gain more experience.”

OpenStack docs:

“Kolla’s mission is to provide production-ready containers and deployment tools for operating OpenStack clouds.”

Deployment with Kolla-Ansible

“What is Kolla-Ansible

AI-reply (Duck-AI):

“Kolla Ansible is a tool **ers** and **Ansible**
automation. It allows omization as they
gain more experience.”

OpenStack docs:

“Kolla’s mission is to pr nStack clouds.”

Aside – insights from ChatGPT:

Origin of the name

“Kolla” was chosen by the original project maintainers and comes from the **Swedish word “kolla”**, which roughly means **“check,” “look,” or “inspect.”** The idea was that the project would **package and verify OpenStack services in containers**, making them easier to inspect, deploy, and manage.

Deployment with Kolla-Ansible

We started off from the Quickstart-Guide. On our deployment node, in a nutshell, we do the following:

- Create a virtual python environment and activate it
- pip install kolla-ansible from the branch that corresponds to the OpenStack release you're going for
- Create `/etc/kolla` and copy the content of `/path/to/venv/share/kolla-ansible/etc_examples/kolla/*` across
- Pip install Ansible Galaxy dependencies
- create/modify `/path/to/venv/share/kolla-ansible/ansible/inventory/multinode` that assigns roles to the different nodes in your cluster (see later slides)
- Create passwords for the different services; those are stored in `/etc/kolla/passwords.yml`
  - We copied the file with a blank list of required passwords across earlier
  - We can use `kolla-genpwd` to create all at once or do it manually in our favourite way
- Modify `/etc/kolla/globals.yml` according to your needs (see later)
  - This file defines close to everything in the deployment. It's GOLD.
- And then we simply deploy…. Famous last words….

Deployment with Kolla-Ansible

Now we essentially just run a bunch of Ansible playbooks

Bootstrap the servers with dependencies (Docker, Python…)

```
kolla-ansible bootstrap-servers -i ./multinode
```

Pre-deployment checks (e.g. can all nodes reach the nodes they need to reach)

```
kolla-ansible prechecks -i ./multinode
```

Deploy (pulls all the containers as needed and deploys OpenStack)

```
kolla-ansible deploy -i ./multinode
```

Deployment with Kolla-Ansible

Now we essentially just run a bunch of Ansible playbooks

Post-deployment step to collect config info

```
kolla-ansible post-deploy
```

→gets you: `/etc/kolla/clouds.yaml` which contains access credentials and also

`/etc/kolla/admin-openrc.sh` which allows you to “talk” to your deployment

Deployment with Kolla-Ansible

KOLLA
an OpenStack Community Project

Check the deployment went fine

Install Python OpenStack CLI (to check and work with OpenStack)

```
pip install python-openstackclient -c https://releases.openstack.org/constraints/upper/|KOLLA_OPENSTACK_RELEASE|
```

Check that everything is up and running nicely – also check this nice guide

`source /etc/kolla/admin-openrc.sh` – log into the ‘admin’ project on OpenStack

Deployment with Kolla-Ansible

KOLLA
an OpenStack Community Project

Check the deployment went fine

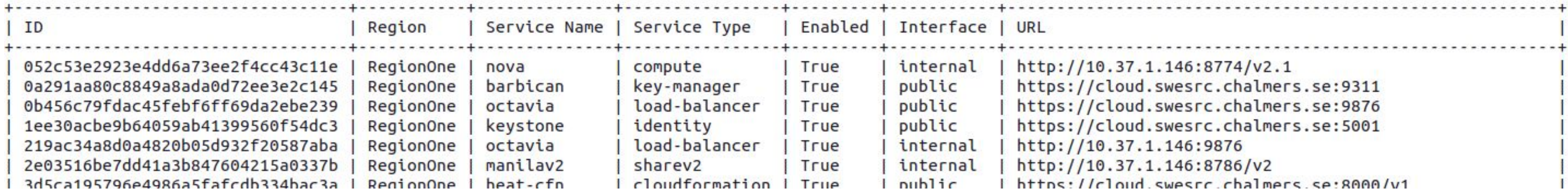
Install Python OpenStack CLI (to check and work with OpenStack)

```
pip install python-openstackclient -c https://releases.openstack.org/constraints/upper/|KOLLA_OPENSTACK_RELEASE|
```

Check that everything is up and running nicely – also check this nice guide

```
source /etc/kolla/admin-openrc.sh
```

```
openstack endpoint list
```



```
+----------------------------------+-----------+--------------+----------------+---------+-----------+----------------------------------------+
| ID                               | Region    | Service Name | Service Type   | Enabled | Interface | URL                                    |
+----------------------------------+-----------+--------------+----------------+---------+-----------+----------------------------------------+
| 052c53e2923e4dd6a73ee2f4cc43c11e | RegionOne | nova         | compute        | True    | internal  | http://10.37.1.146:8774/v2.1           |
| 0a291aa80c8849a8ada0d72ee3e2c145 | RegionOne | barbican     | key-manager    | True    | public    | https://cloud.swesrc.chalmers.se:9311  |
| 0b456c79fdac45febf6ff69da2ebe239 | RegionOne | octavia      | load-balancer  | True    | public    | https://cloud.swesrc.chalmers.se:9876  |
| 1ee30acbe9b64059ab41399560f54dc3 | RegionOne | keystone     | identity       | True    | public    | https://cloud.swesrc.chalmers.se:5001  |
| 219ac34a8d0a4820b05d932f20587aba | RegionOne | octavia      | load-balancer  | True    | internal  | http://10.37.1.146:9876                |
| 2e03516be7dd41a3b847604215a0337b | RegionOne | manilav2     | sharev2        | True    | internal  | http://10.37.1.146:8786/v2             |
| 3d5ca195796e4986a5fafcdb334bac3a | RegionOne | heat-cfn     | cloudformation | True    | public    | https://cloud.swesrc.chalmers.se:8000/v1 |
```

Deployment with Kolla-Ansible

Check the deployment went fine

Install Python OpenStack CLI (to check and work with OpenStack)

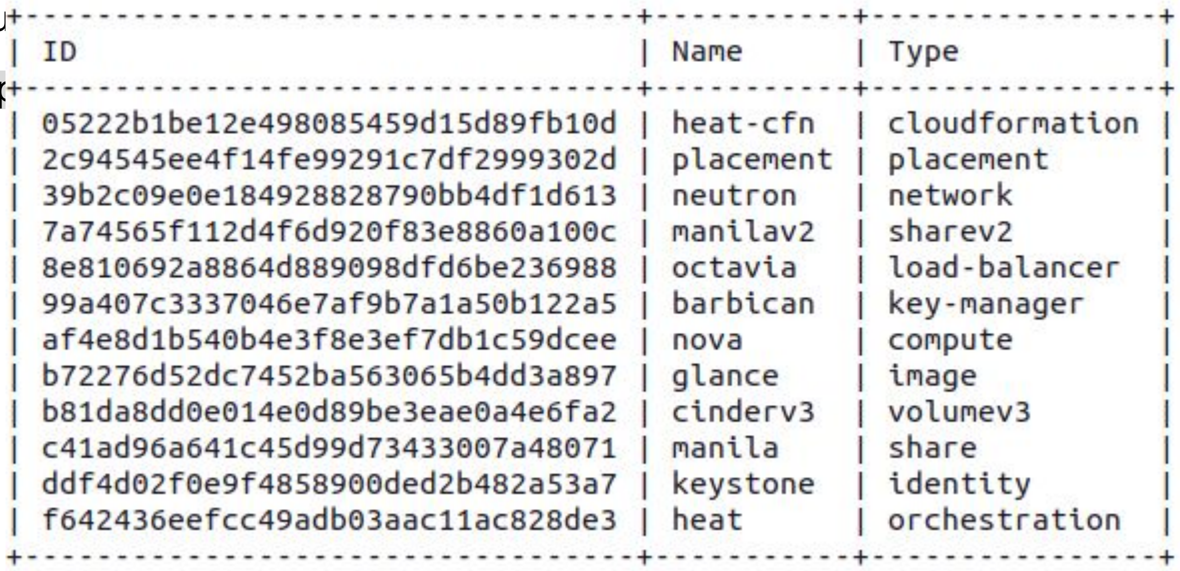
```
pip install python-openstackclient -c https://releases.openstack.org/constraints/upper/|KOLLA_OPENSTACK_RELEASE|
```

Check that everything is u

```
source /etc/kolla/admin-op
```

```
openstack endpoint list
```

```
openstack service list
```



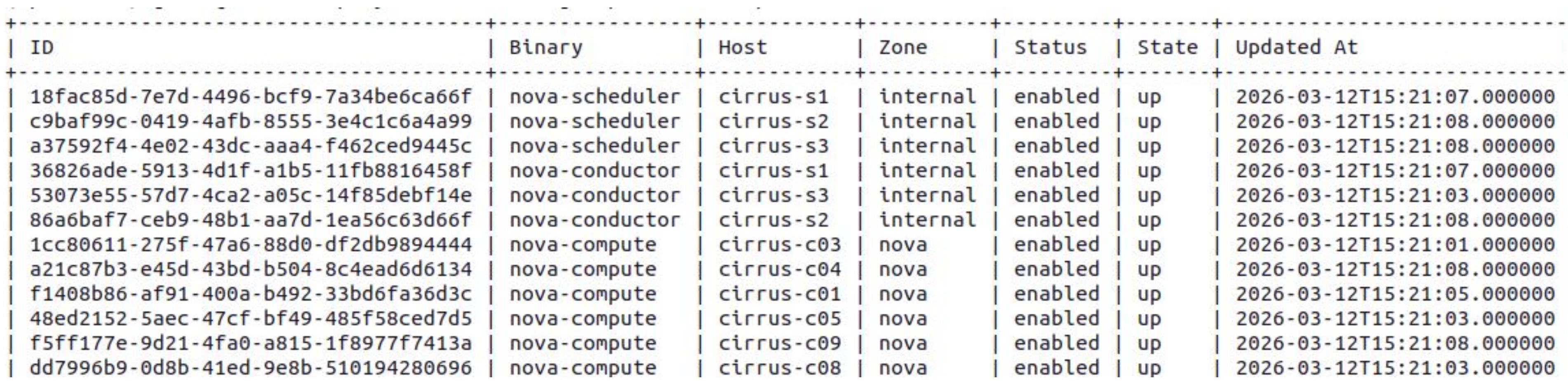
```
+----------------------------------+-----------+----------------+
| ID                               | Name      | Type           |
+----------------------------------+-----------+----------------+
| 05222b1be12e498085459d15d89fb10d | heat-cfn  | cloudformation |
| 2c94545ee4f14fe99291c7df2999302d | placement | placement      |
| 39b2c09e0e184928828790bb4df1d613 | neutron   | network        |
| 7a74565f112d4f6d920f83e8860a100c | manilav2  | sharev2        |
| 8e810692a8864d889098dfd6be236988 | octavia   | load-balancer  |
| 99a407c3337046e7af9b7a1a50b122a5 | barbican  | key-manager    |
| af4e8d1b540b4e3f8e3ef7db1c59dcee | nova      | compute        |
| b72276d52dc7452ba563065b4dd3a897 | glance    | image          |
| b81da8dd0e014e0d89be3eae0a4e6fa2 | cinderv3  | volumev3       |
| c41ad96a641c45d99d73433007a48071 | manila    | share          |
| ddf4d02f0e9f4858900ded2b482a53a7 | keystone  | identity       |
| f642436eefcc49adb03aac11ac828de3 | heat      | orchestration  |
+----------------------------------+-----------+----------------+
```

Deployment with Kolla-Ansible

KOLLA

an OpenStack Community Project

Check the deployment went fine



```
+--------------------------------------+----------------+------------+----------+---------+-------+----------------------------
| ID                                   | Binary         | Host       | Zone     | Status  | State | Updated At
+--------------------------------------+----------------+------------+----------+---------+-------+----------------------------
| 18fac85d-7e7d-4496-bcf9-7a34be6ca66f | nova-scheduler | cirrus-s1  | internal | enabled | up    | 2026-03-12T15:21:07.000000
| c9baf99c-0419-4afb-8555-3e4c1c6a4a99 | nova-scheduler | cirrus-s2  | internal | enabled | up    | 2026-03-12T15:21:08.000000
| a37592f4-4e02-43dc-aaa4-f462ced9445c | nova-scheduler | cirrus-s3  | internal | enabled | up    | 2026-03-12T15:21:08.000000
| 36826ade-5913-4d1f-a1b5-11fb8816458f | nova-conductor | cirrus-s1  | internal | enabled | up    | 2026-03-12T15:21:07.000000
| 53073e55-57d7-4ca2-a05c-14f85debf14e | nova-conductor | cirrus-s3  | internal | enabled | up    | 2026-03-12T15:21:03.000000
| 86a6baf7-ceb9-48b1-aa7d-1ea56c63d66f | nova-conductor | cirrus-s2  | internal | enabled | up    | 2026-03-12T15:21:08.000000
| 1cc80611-275f-47a6-88d0-df2db9894444 | nova-compute   | cirrus-c03 | nova     | enabled | up    | 2026-03-12T15:21:01.000000
| a21c87b3-e45d-43bd-b504-8c4ead6d6134 | nova-compute   | cirrus-c04 | nova     | enabled | up    | 2026-03-12T15:21:08.000000
| f1408b86-af91-400a-b492-33bd6fa36d3c | nova-compute   | cirrus-c01 | nova     | enabled | up    | 2026-03-12T15:21:05.000000
| 48ed2152-5aec-47cf-bf49-485f58ced7d5 | nova-compute   | cirrus-c05 | nova     | enabled | up    | 2026-03-12T15:21:03.000000
| f5ff177e-9d21-4fa0-a815-1f8977f7413a | nova-compute   | cirrus-c09 | nova     | enabled | up    | 2026-03-12T15:21:08.000000
| dd7996b9-0d8b-41ed-9e8b-510194280696 | nova-compute   | cirrus-c08 | nova     | enabled | up    | 2026-03-12T15:21:03.000000
```

openstack compute service list

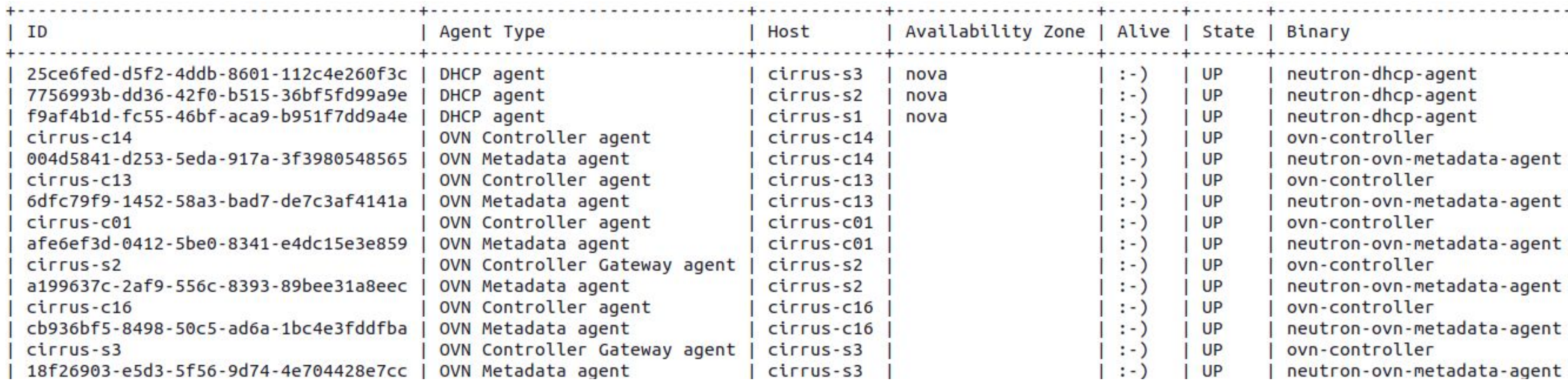
Deployment with Kolla-Ansible

KOLLA

an OpenStack Community Project

Check the deployment went fine

Install Python OpenStack CLI (to check and work with OpenStack)

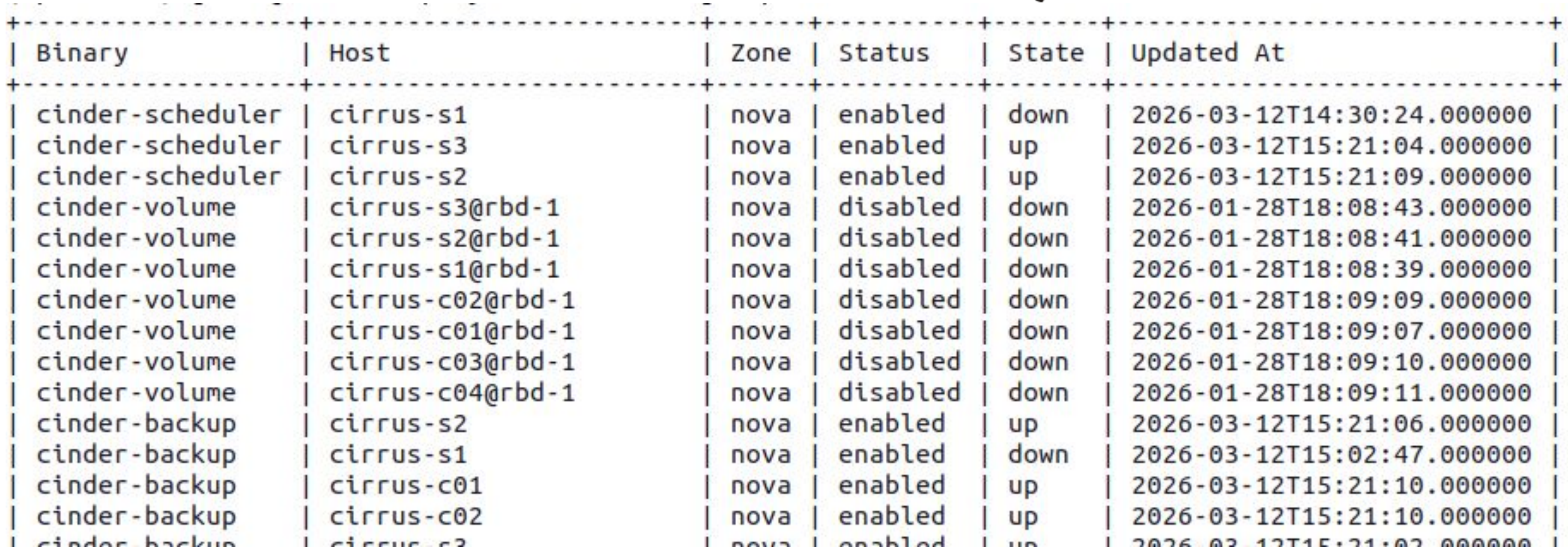


```
+--------------------------------------+------------------------------+--------------+-------------------+-------+-------+---------------------------
| ID                                   | Agent Type                   | Host         | Availability Zone | Alive | State | Binary
+--------------------------------------+------------------------------+--------------+-------------------+-------+-------+---------------------------
| 25ce6fed-d5f2-4ddb-8601-112c4e260f3c | DHCP agent                   | cirrus-s3    | nova              | :-)   | UP    | neutron-dhcp-agent
| 7756993b-dd36-42f0-b515-36bf5fd99a9e | DHCP agent                   | cirrus-s2    | nova              | :-)   | UP    | neutron-dhcp-agent
| f9af4b1d-fc55-46bf-aca9-b951f7dd9a4e | DHCP agent                   | cirrus-s1    | nova              | :-)   | UP    | neutron-dhcp-agent
| cirrus-c14                           | OVN Controller agent         | cirrus-c14   |                   | :-)   | UP    | ovn-controller
| 004d5841-d253-5eda-917a-3f3980548565 | OVN Metadata agent           | cirrus-c14   |                   | :-)   | UP    | neutron-ovn-metadata-agent
| cirrus-c13                           | OVN Controller agent         | cirrus-c13   |                   | :-)   | UP    | ovn-controller
| 6dfc79f9-1452-58a3-bad7-de7c3af4141a | OVN Metadata agent           | cirrus-c13   |                   | :-)   | UP    | neutron-ovn-metadata-agent
| cirrus-c01                           | OVN Controller agent         | cirrus-c01   |                   | :-)   | UP    | ovn-controller
| afe6ef3d-0412-5be0-8341-e4dc15e3e859 | OVN Metadata agent           | cirrus-c01   |                   | :-)   | UP    | neutron-ovn-metadata-agent
| cirrus-s2                            | OVN Controller Gateway agent | cirrus-s2    |                   | :-)   | UP    | ovn-controller
| a199637c-2af9-556c-8393-89bee31a8eec | OVN Metadata agent           | cirrus-s2    |                   | :-)   | UP    | neutron-ovn-metadata-agent
| cirrus-c16                           | OVN Controller agent         | cirrus-c16   |                   | :-)   | UP    | ovn-controller
| cb936bf5-8498-50c5-ad6a-1bc4e3fddfba | OVN Metadata agent           | cirrus-c16   |                   | :-)   | UP    | neutron-ovn-metadata-agent
| cirrus-s3                            | OVN Controller Gateway agent | cirrus-s3    |                   | :-)   | UP    | ovn-controller
| 18f26903-e5d3-5f56-9d74-4e704428e7cc | OVN Metadata agent           | cirrus-s3    |                   | :-)   | UP    | neutron-ovn-metadata-agent
```

openstack network agent list

Deployment with Kolla-Ansible

Check the deployment went fine



```
+------------------+--------------------------+------+----------+-------+----------------------------+
| Binary           | Host                     | Zone | Status   | State | Updated At                 |
+------------------+--------------------------+------+----------+-------+----------------------------+
| cinder-scheduler | cirrus-s1                | nova | enabled  | down  | 2026-03-12T14:30:24.000000 |
| cinder-scheduler | cirrus-s3                | nova | enabled  | up    | 2026-03-12T15:21:04.000000 |
| cinder-scheduler | cirrus-s2                | nova | enabled  | up    | 2026-03-12T15:21:09.000000 |
| cinder-volume    | cirrus-s3@rbd-1          | nova | disabled | down  | 2026-01-28T18:08:43.000000 |
| cinder-volume    | cirrus-s2@rbd-1          | nova | disabled | down  | 2026-01-28T18:08:41.000000 |
| cinder-volume    | cirrus-s1@rbd-1          | nova | disabled | down  | 2026-01-28T18:08:39.000000 |
| cinder-volume    | cirrus-c02@rbd-1         | nova | disabled | down  | 2026-01-28T18:09:09.000000 |
| cinder-volume    | cirrus-c01@rbd-1         | nova | disabled | down  | 2026-01-28T18:09:07.000000 |
| cinder-volume    | cirrus-c03@rbd-1         | nova | disabled | down  | 2026-01-28T18:09:10.000000 |
| cinder-volume    | cirrus-c04@rbd-1         | nova | disabled | down  | 2026-01-28T18:09:11.000000 |
| cinder-backup    | cirrus-s2                | nova | enabled  | up    | 2026-03-12T15:21:06.000000 |
| cinder-backup    | cirrus-s1                | nova | enabled  | down  | 2026-03-12T15:02:47.000000 |
| cinder-backup    | cirrus-c01               | nova | enabled  | up    | 2026-03-12T15:21:10.000000 |
| cinder-backup    | cirrus-c02               | nova | enabled  | up    | 2026-03-12T15:21:10.000000 |
```

openstack volume service list

Deployment with Kolla-Ansible

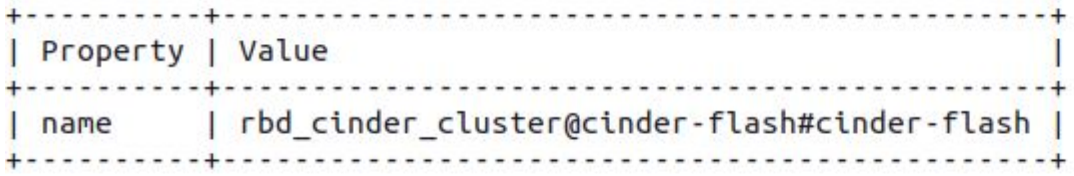
Check the deployment went fine

Install Python OpenStack CLI (to check and work with OpenStack)

```
pip install python-openstackclient -c https://releases.openstack.org/constraints/upper/|KOLLA_OPENSTACK_RELEASE|
```

Check that everything is up and running nicely – also check this nice guide

```
source /etc/kolla/admin-openrc.sh
openstack endpoint list
openstack service list
openstack compute service list
openstack network agent list
openstack volume service list
cinder get-pools
```



```
+----------+--------------------------------------------+
| Property | Value                                      |
+----------+--------------------------------------------+
| name     | rbd_cinder_cluster@cinder-flash#cinder-flash |
+----------+--------------------------------------------+
```

Deployment with Kolla-Ansible

In case things are not to your liking, you can always reconfigure – by service and/or by node

```
kolla-ansible reconfigure -i ./multinode
kolla-ansible reconfigure -i ./multinode -t openvswitch,neutron,ovn
kolla-ansible reconfigure -i ./multinode -t openvswitch,neutron,ovn --limit cirrus-c01,cirrus-c14
```

Or you destroy the whole thing and start from scratch

```
kolla-ansible destroy -i ./multinode --include-images --yes-i-really-really-mean-it
```

This can also be done on a per-node basis

```
kolla-ansible destroy -i ./multinode --include-images --yes-i-really-really-mean-it --limit cirrus-c02
```

Questions?

Next up:
How to configure your OpenStack deployment

Deployment with Kolla-Ansible

Configuration

Most of the configuration is done in /etc/kolla/globals.yml

However, the configuration can be fine grained on a per node and/or per service basis.

Kolla-ansible picks up from /etc/kolla/config/[cinder, nova, manila, neutron, glance, …]

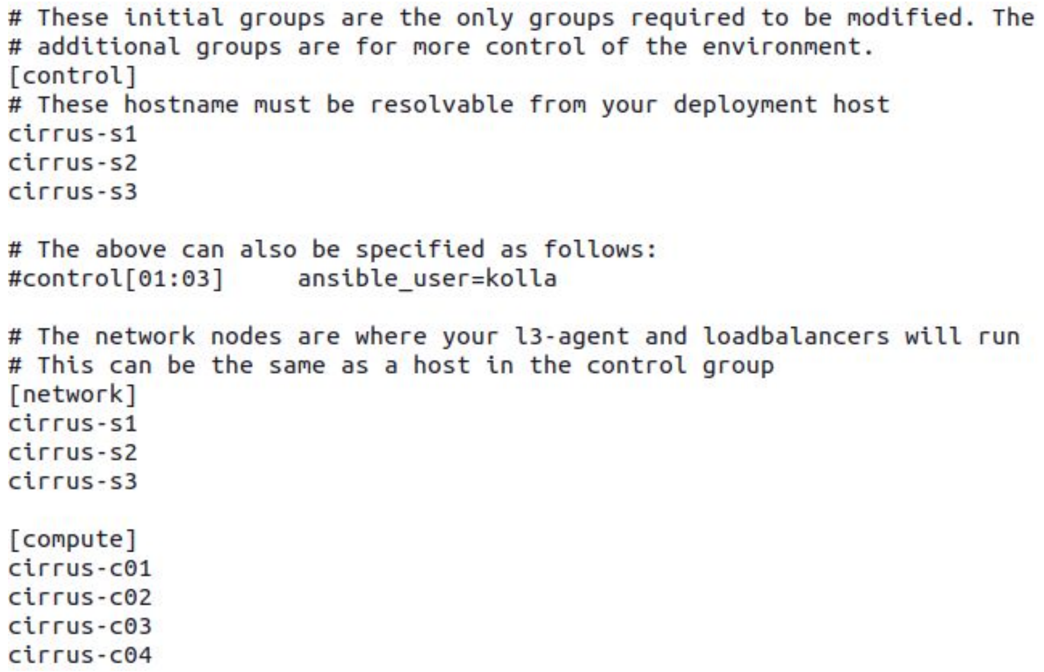
Role assignment is done in path/to/venv/share/kolla-ansible/ansible/inventory/multinode

Deployment with Kolla-Ansible

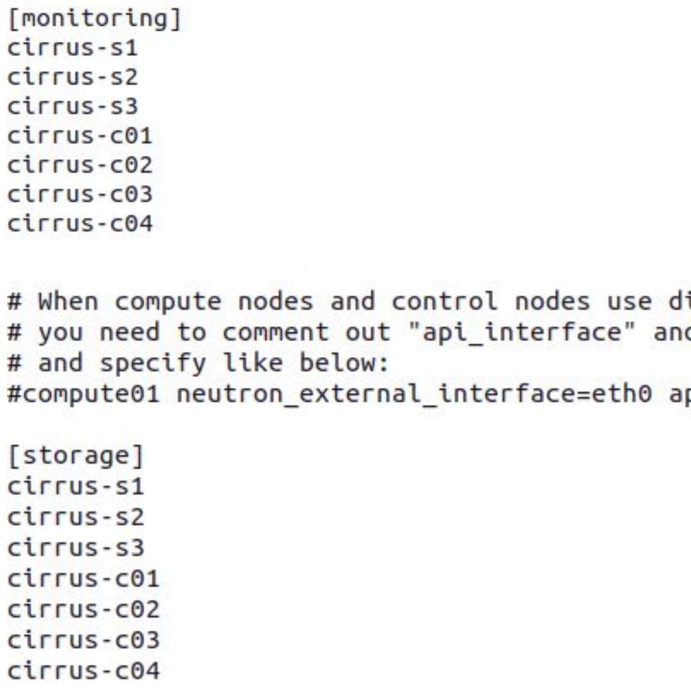
KOLLA

an OpenStack Community Project

Configuration – role assignment in “multinode”



```
# These initial groups are the only groups required to be modified. The
# additional groups are for more control of the environment.
[control]
# These hostname must be resolvable from your deployment host
cirrus-s1
cirrus-s2
cirrus-s3

# The above can also be specified as follows:
#control[01:03]     ansible_user=kolla

# The network nodes are where your l3-agent and loadbalancers will run
# This can be the same as a host in the control group
[network]
cirrus-s1
cirrus-s2
cirrus-s3

[compute]
cirrus-c01
cirrus-c02
cirrus-c03
cirrus-c04
```

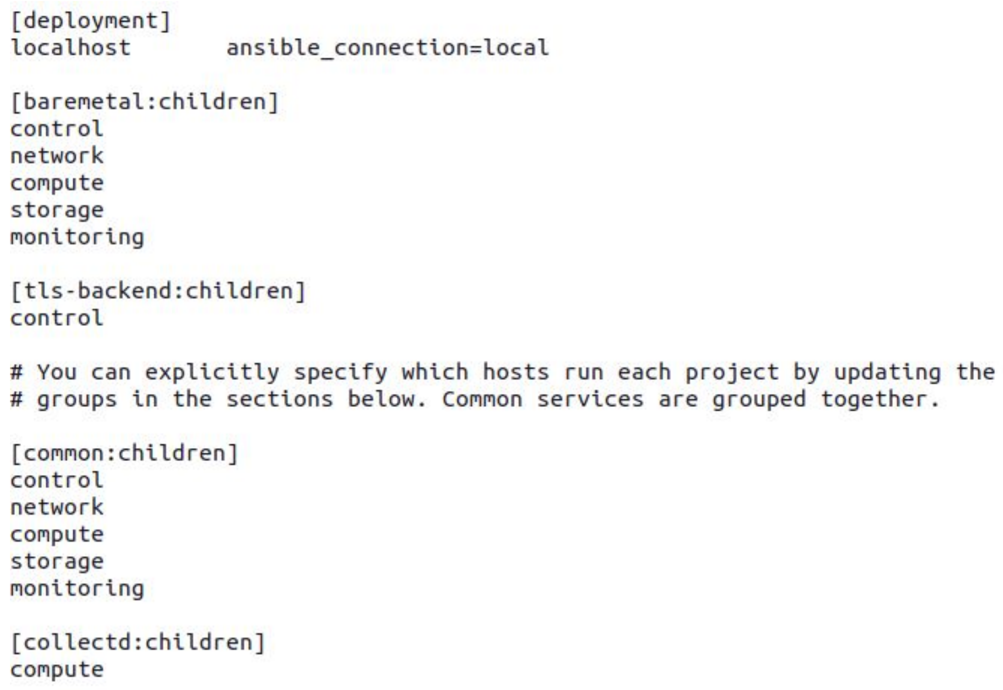


```
[monitoring]
cirrus-s1
cirrus-s2
cirrus-s3
cirrus-c01
cirrus-c02
cirrus-c03
cirrus-c04


# When compute nodes and control nodes use di
# you need to comment out "api_interface" and
# and specify like below:
#compute01 neutron_external_interface=eth0 ap

[storage]
cirrus-s1
cirrus-s2
cirrus-s3
cirrus-c01
cirrus-c02
cirrus-c03
cirrus-c04
```

Deployment with Kolla-Ansible

KOLLA
an OpenStack Community Project

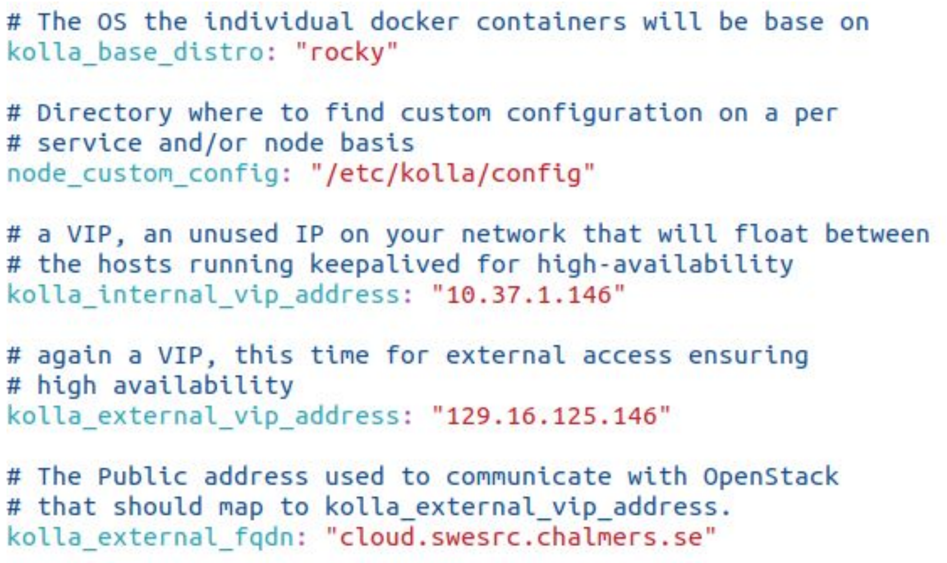
Configuration – role assignment in "multinode"



```
[deployment]
localhost       ansible_connection=local

[baremetal:children]
control
network
compute
storage
monitoring

[tls-backend:children]
control

# You can explicitly specify which hosts run each project by updating the
# groups in the sections below. Common services are grouped together.

[common:children]
control
network
compute
storage
monitoring

[collectd:children]
compute
```

Deployment with Kolla-Ansible

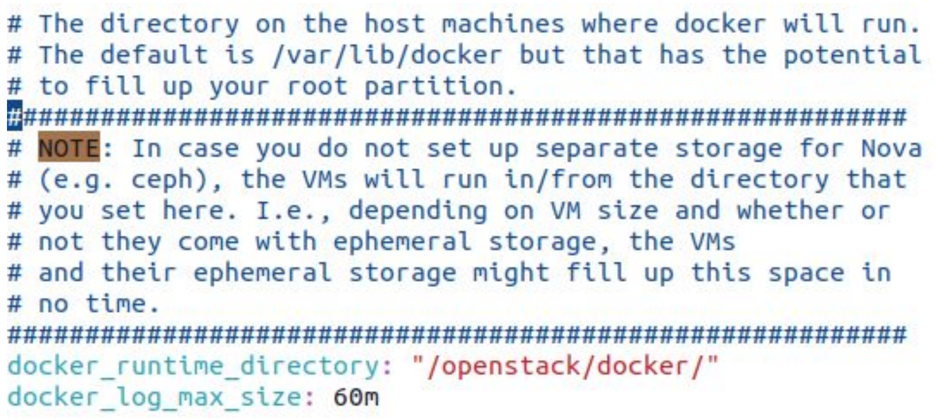
Configuration – “globals.yml”



```
# The OS the individual docker containers will be base on
kolla_base_distro: "rocky"

# Directory where to find custom configuration on a per
# service and/or node basis
node_custom_config: "/etc/kolla/config"

# a VIP, an unused IP on your network that will float between
# the hosts running keepalived for high-availability
kolla_internal_vip_address: "10.37.1.146"

# again a VIP, this time for external access ensuring
# high availability
kolla_external_vip_address: "129.16.125.146"

# The Public address used to communicate with OpenStack
# that should map to kolla_external_vip_address.
kolla_external_fqdn: "cloud.swesrc.chalmers.se"
```

Deployment with Kolla-Ansible

Configuration – “globals.yml”



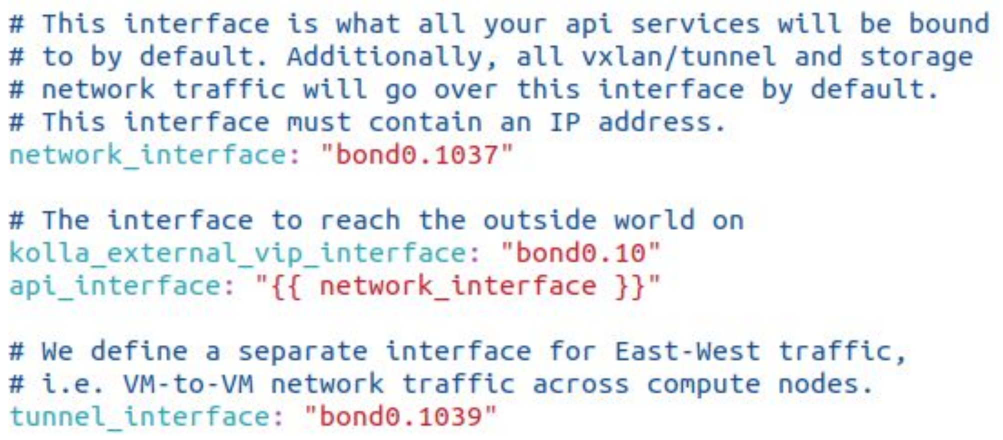
```
# The directory on the host machines where docker will run.
# The default is /var/lib/docker but that has the potential
# to fill up your root partition.
#########################################################
# NOTE: In case you do not set up separate storage for Nova
# (e.g. ceph), the VMs will run in/from the directory that
# you set here. I.e., depending on VM size and whether or
# not they come with ephemeral storage, the VMs
# and their ephemeral storage might fill up this space in
# no time.
#########################################################
docker_runtime_directory: "/openstack/docker/"
docker_log_max_size: 60m
```

In our case /openstack are
700 GB NVMes on the nodes

Deployment with Kolla-Ansible

KOLLA
an OpenStack Community Project

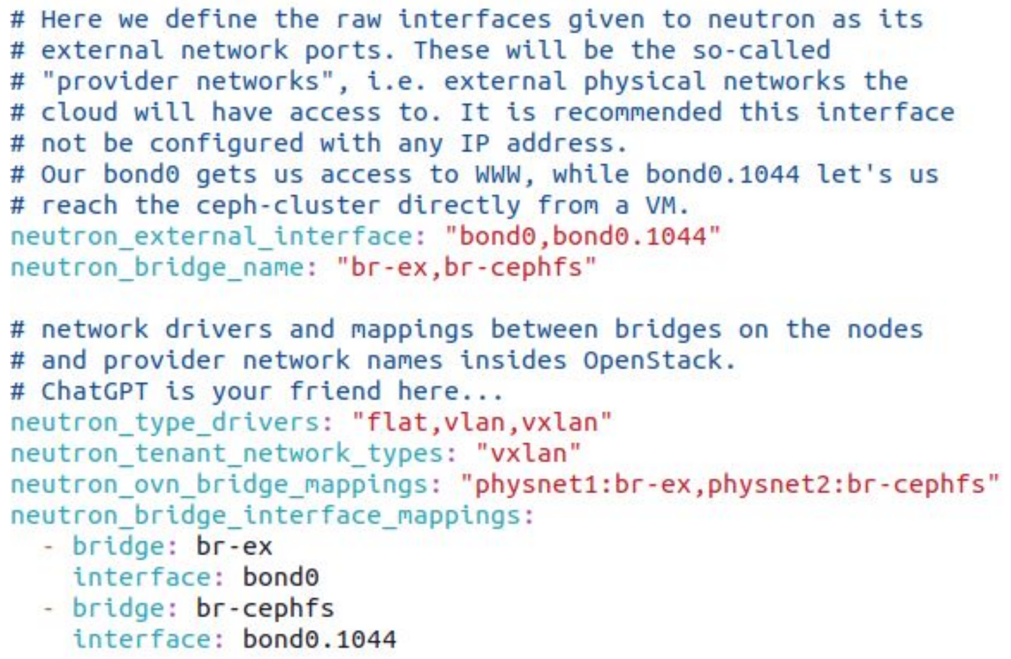
Configuration – “globals.yml”



```
# This interface is what all your api services will be bound
# to by default. Additionally, all vxlan/tunnel and storage
# network traffic will go over this interface by default.
# This interface must contain an IP address.
network_interface: "bond0.1037"

# The interface to reach the outside world on
kolla_external_vip_interface: "bond0.10"
api_interface: "{{ network_interface }}"

# We define a separate interface for East-West traffic,
# i.e. VM-to-VM network traffic across compute nodes.
tunnel_interface: "bond0.1039"
```

Deployment with Kolla-Ansible

Configuration – “globals.yml”

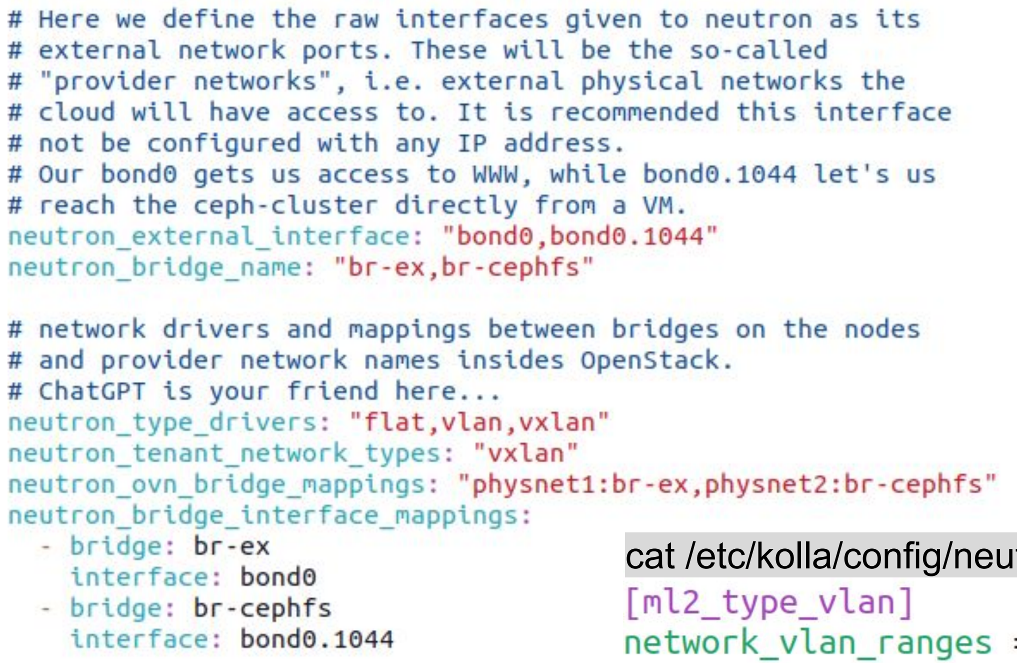


```
# Here we define the raw interfaces given to neutron as its
# external network ports. These will be the so-called
# "provider networks", i.e. external physical networks the
# cloud will have access to. It is recommended this interface
# not be configured with any IP address.
# Our bond0 gets us access to WWW, while bond0.1044 let's us
# reach the ceph-cluster directly from a VM.
neutron_external_interface: "bond0,bond0.1044"
neutron_bridge_name: "br-ex,br-cephfs"

# network drivers and mappings between bridges on the nodes
# and provider network names insides OpenStack.
# ChatGPT is your friend here...
neutron_type_drivers: "flat,vlan,vxlan"
neutron_tenant_network_types: "vxlan"
neutron_ovn_bridge_mappings: "physnet1:br-ex,physnet2:br-cephfs"
neutron_bridge_interface_mappings:
  - bridge: br-ex
    interface: bond0
  - bridge: br-cephfs
    interface: bond0.1044
```

Deployment with Kolla-Ansible

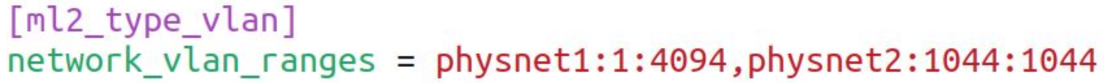
KOLLA
an OpenStack Community Project

Configuration – “globals.yml”



```
# Here we define the raw interfaces given to neutron as its
# external network ports. These will be the so-called
# "provider networks", i.e. external physical networks the
# cloud will have access to. It is recommended this interface
# not be configured with any IP address.
# Our bond0 gets us access to WWW, while bond0.1044 let's us
# reach the ceph-cluster directly from a VM.
neutron_external_interface: "bond0,bond0.1044"
neutron_bridge_name: "br-ex,br-cephfs"

# network drivers and mappings between bridges on the nodes
# and provider network names insides OpenStack.
# ChatGPT is your friend here...
neutron_type_drivers: "flat,vlan,vxlan"
neutron_tenant_network_types: "vxlan"
neutron_ovn_bridge_mappings: "physnet1:br-ex,physnet2:br-cephfs"
neutron_bridge_interface_mappings:
  - bridge: br-ex
    interface: bond0
  - bridge: br-cephfs
    interface: bond0.1044
```

cat /etc/kolla/config/neutron/ml2_conf.ini



```
[ml2_type_vlan]
network_vlan_ranges = physnet1:1:4094,physnet2:1044:1044
```

Deployment with Kolla-Ansible

KOLLA
an OpenStack Community Project

Configuration – "globals.yml"

```
# Use OVN (Open Virtual Network) as the network driver.
neutron_plugin_agent: "ovn"

# Enable distributed floating ip for OVN deployments
neutron_ovn_distributed_fip: "yes"

# Enable DHCP agent(s) to use with OVN
neutron_ovn_dhcp_agent: "yes"
```

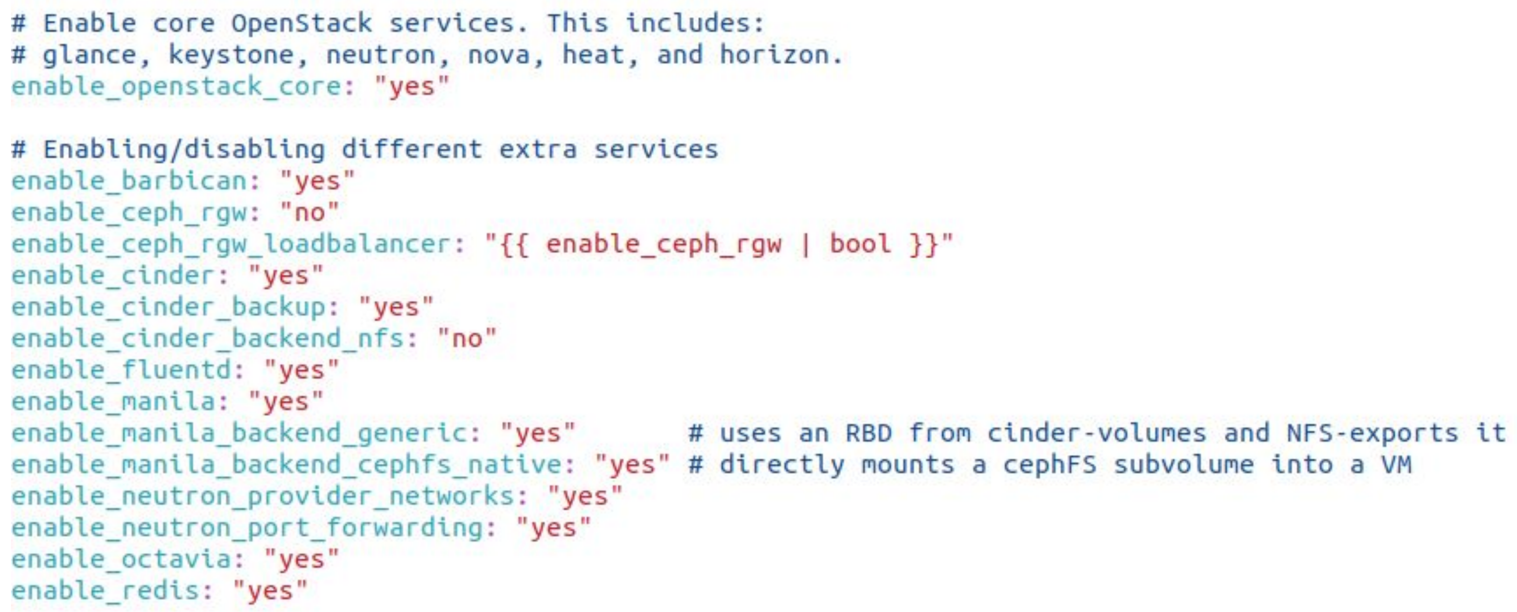
```
# Configure Octavia (the loadbalancer service) to use OVN instead
# of amphora.
octavia_provider_drivers: "ovn:OVN provider"
octavia_provider_agents: "ovn"
octavia_auto_configure: yes
octavia_loadbalancer_topology: "SINGLE"
```

```
# firewall acrobatics; requires quite specific
# firewalld settings on the nodes
disable_firewall: "false"
enable_external_api_firewalld: "true"
external_api_firewalld_zone: "public"

# Handling of certificates in the containers.
kolla_enable_tls_external: "yes"
openstack_cacert: "/etc/pki/tls/certs/ca-bundle.crt"
kolla_admin_openrc_cacert: "{{ openstack_cacert }}"
kolla_copy_ca_into_containers: "yes"
```

```
# SSL certificate settings
enable_letsencrypt: "yes"
letsencrypt_email: "tekniker@c3se.chalmers.se"
letsencrypt_cron_renew_schedule:  "0   */12   *   *   *"

# Log levels
openstack_logging_debug: "True"
nova_libvirt_logging_debug: "False"
```

Deployment with Kolla-Ansible

Configuration – “globals.yml”

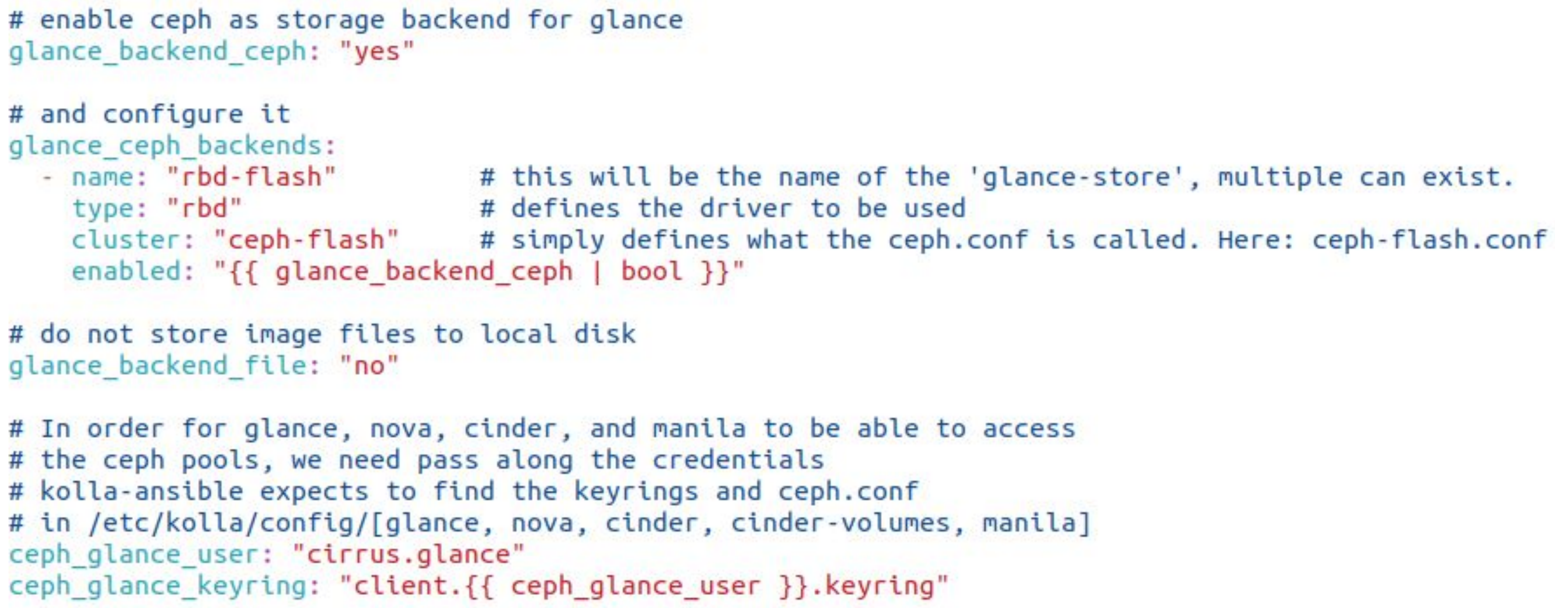


```
# Enable core OpenStack services. This includes:
# glance, keystone, neutron, nova, heat, and horizon.
enable_openstack_core: "yes"

# Enabling/disabling different extra services
enable_barbican: "yes"
enable_ceph_rgw: "no"
enable_ceph_rgw_loadbalancer: "{{ enable_ceph_rgw | bool }}"
enable_cinder: "yes"
enable_cinder_backup: "yes"
enable_cinder_backend_nfs: "no"
enable_fluentd: "yes"
enable_manila: "yes"
enable_manila_backend_generic: "yes"       # uses an RBD from cinder-volumes and NFS-exports it
enable_manila_backend_cephfs_native: "yes" # directly mounts a cephFS subvolume into a VM
enable_neutron_provider_networks: "yes"
enable_neutron_port_forwarding: "yes"
enable_octavia: "yes"
enable_redis: "yes"
```

Deployment with Kolla-Ansible

KOLLA
an OpenStack Community Project

Configuration – “globals.yml”

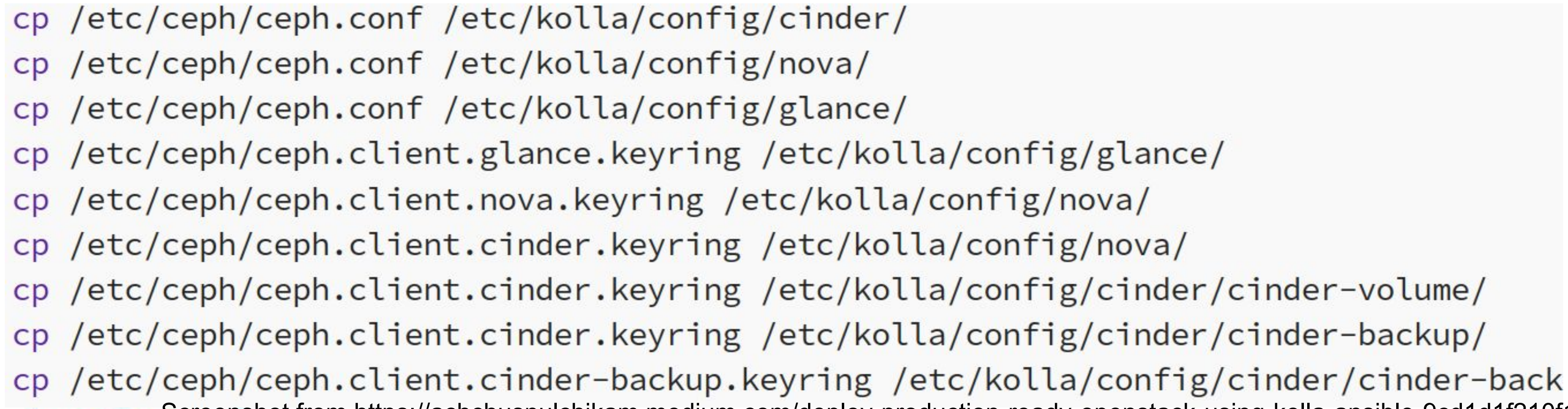


```
# enable ceph as storage backend for glance
glance_backend_ceph: "yes"

# and configure it
glance_ceph_backends:
  - name: "rbd-flash"          # this will be the name of the 'glance-store', multiple can exist.
    type: "rbd"                # defines the driver to be used
    cluster: "ceph-flash"      # simply defines what the ceph.conf is called. Here: ceph-flash.conf
    enabled: "{{ glance_backend_ceph | bool }}"

# do not store image files to local disk
glance_backend_file: "no"

# In order for glance, nova, cinder, and manila to be able to access
# the ceph pools, we need pass along the credentials
# kolla-ansible expects to find the keyrings and ceph.conf
# in /etc/kolla/config/[glance, nova, cinder, cinder-volumes, manila]
ceph_glance_user: "cirrus.glance"
ceph_glance_keyring: "client.{{ ceph_glance_user }}.keyring"
```

Deployment with Kolla-Ansible

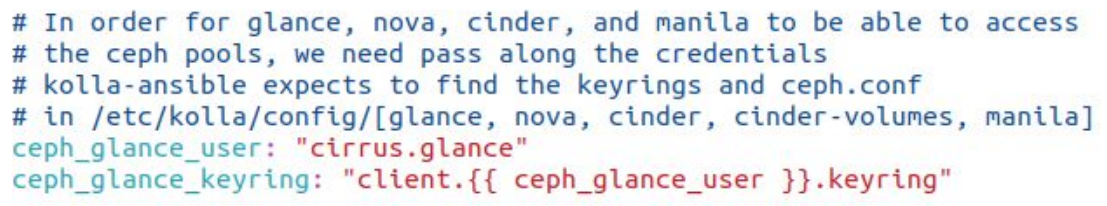
KOLLA
an OpenStack Community Project

Configuration – “globals.yml”



```
cp /etc/ceph/ceph.conf /etc/kolla/config/cinder/
cp /etc/ceph/ceph.conf /etc/kolla/config/nova/
cp /etc/ceph/ceph.conf /etc/kolla/config/glance/
cp /etc/ceph/ceph.client.glance.keyring /etc/kolla/config/glance/
cp /etc/ceph/ceph.client.nova.keyring /etc/kolla/config/nova/
cp /etc/ceph/ceph.client.cinder.keyring /etc/kolla/config/nova/
cp /etc/ceph/ceph.client.cinder.keyring /etc/kolla/config/cinder/cinder-volume/
cp /etc/ceph/ceph.client.cinder.keyring /etc/kolla/config/cinder/cinder-backup/
cp /etc/ceph/ceph.client.cinder-backup.keyring /etc/kolla/config/cinder/cinder-back
```

Screenshot from https://achchusnulchikam.medium.com/deploy-production-ready-openstack-using-kolla-ansible-9cd1d1f210f1

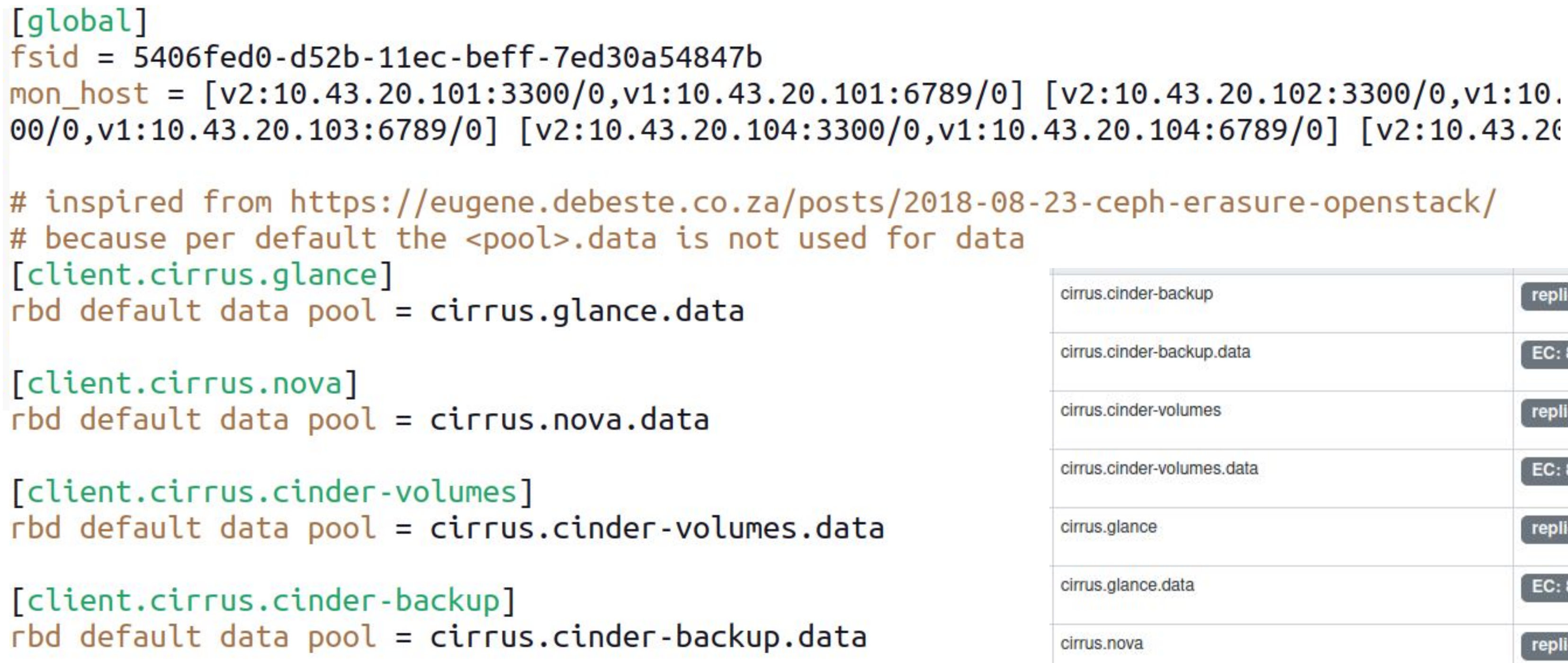


```
# In order for glance, nova, cinder, and manila to be able to access
# the ceph pools, we need pass along the credentials
# kolla-ansible expects to find the keyrings and ceph.conf
# in /etc/kolla/config/[glance, nova, cinder, cinder-volumes, manila]
ceph_glance_user: "cirrus.glance"
ceph_glance_keyring: "client.{{ ceph_glance_user }}.keyring"
```

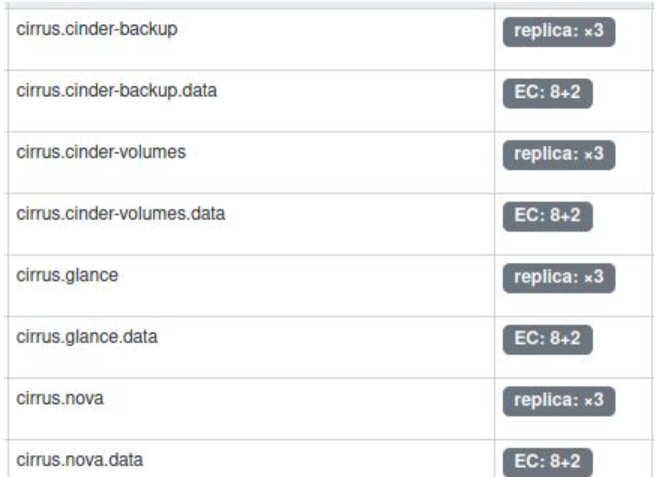
Deployment with Kolla-Ansible

KOLLA
an OpenStack Community Project

Small detour: ceph.conf



```
[global]
fsid = 5406fed0-d52b-11ec-beff-7ed30a54847b
mon_host = [v2:10.43.20.101:3300/0,v1:10.43.20.101:6789/0] [v2:10.43.20.102:3300/0,v1:10.
00/0,v1:10.43.20.103:6789/0] [v2:10.43.20.104:3300/0,v1:10.43.20.104:6789/0] [v2:10.43.2

# inspired from https://eugene.debeste.co.za/posts/2018-08-23-ceph-erasure-openstack/
# because per default the <pool>.data is not used for data
[client.cirrus.glance]
rbd default data pool = cirrus.glance.data

[client.cirrus.nova]
rbd default data pool = cirrus.nova.data

[client.cirrus.cinder-volumes]
rbd default data pool = cirrus.cinder-volumes.data

[client.cirrus.cinder-backup]
rbd default data pool = cirrus.cinder-backup.data
```

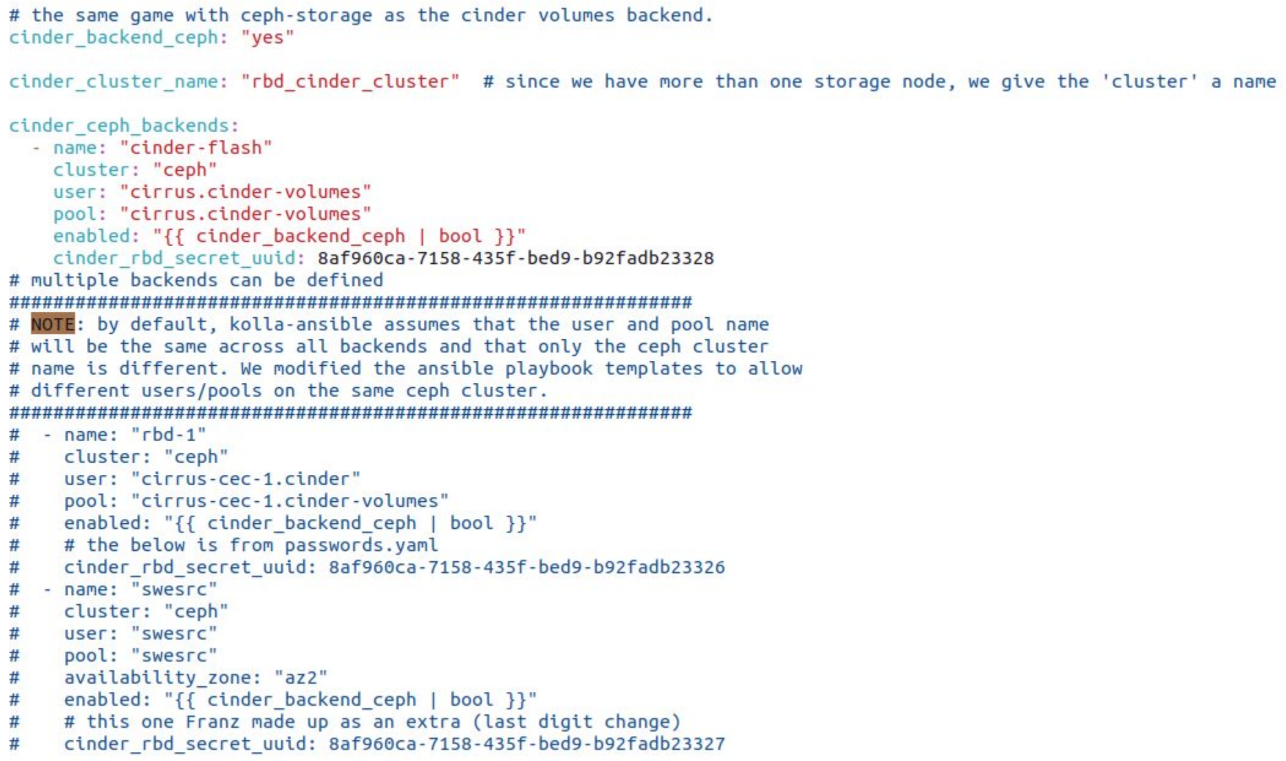


cirrus.cinder-backup	replica: ×3
cirrus.cinder-backup.data	EC: 8+2
cirrus.cinder-volumes	replica: ×3
cirrus.cinder-volumes.data	EC: 8+2
cirrus.glance	replica: ×3
cirrus.glance.data	EC: 8+2
cirrus.nova	replica: ×3
cirrus.nova.data	EC: 8+2

Deployment with Kolla-Ansible

KOLLA
an OpenStack Community Project

Configuration – “globals.yml”



```
# the same game with ceph-storage as the cinder volumes backend.
cinder_backend_ceph: "yes"

cinder_cluster_name: "rbd_cinder_cluster"  # since we have more than one storage node, we give the 'cluster' a name

cinder_ceph_backends:
  - name: "cinder-flash"
    cluster: "ceph"
    user: "cirrus.cinder-volumes"
    pool: "cirrus.cinder-volumes"
    enabled: "{{ cinder_backend_ceph | bool }}"
    cinder_rbd_secret_uuid: 8af960ca-7158-435f-bed9-b92fadb23328
# multiple backends can be defined
###############################################################
# NOTE: by default, kolla-ansible assumes that the user and pool name
# will be the same across all backends and that only the ceph cluster
# name is different. We modified the ansible playbook templates to allow
# different users/pools on the same ceph cluster.
###############################################################
#  - name: "rbd-1"
#    cluster: "ceph"
#    user: "cirrus-cec-1.cinder"
#    pool: "cirrus-cec-1.cinder-volumes"
#    enabled: "{{ cinder_backend_ceph | bool }}"
#    # the below is from passwords.yaml
#    cinder_rbd_secret_uuid: 8af960ca-7158-435f-bed9-b92fadb23326
#  - name: "swesrc"
#    cluster: "ceph"
#    user: "swesrc"
#    pool: "swesrc"
#    availability_zone: "az2"
#    enabled: "{{ cinder_backend_ceph | bool }}"
#    # this one Franz made up as an extra (last digit change)
#    cinder_rbd_secret_uuid: 8af960ca-7158-435f-bed9-b92fadb23327
```

Deployment with Kolla-Ansible

KOLLA

an OpenStack Community Project

Configuration – “globals.yml”

```
# set up cinder backup storage backend
cinder_backup_ceph_backend:
  name: "cinder-backup"
  cluster: "ceph"
  user: "{{ ceph_cinder_backup_user }}"
  pool: "{{ ceph_cinder_backup_pool_name }}"
  enabled: "{{ enable_cinder_backup | bool }}"

ceph_cinder_backup_user: "cirrus.cinder-backup"
ceph_cinder_backup_keyring: "client.{{ ceph_cinder_backup_user }}.keyring"
ceph_cinder_backup_pool_name: "cirrus.cinder-backup"


# the same for nova; we leave namings and all at defaults.
nova_backend_ceph: "yes"
ceph_nova_user: "cirrus.nova"
ceph_nova_keyring: "client.{{ ceph_nova_user }}.keyring"
ceph_nova_pool_name: "cirrus.nova"

# Similar for manila and cephFS. We leave driver choices and
# backend namings at defaults. We just need to tell kolla what
# the file system on the ceph cluster is called. Otherwise it
# will try and connect to the first one it finds (we have several).
manila_cephfs_filesystem_name: "mira"
ceph_manila_user: "manila"
ceph_manila_keyring: "client.{{ ceph_manila_user }}.keyring"
```

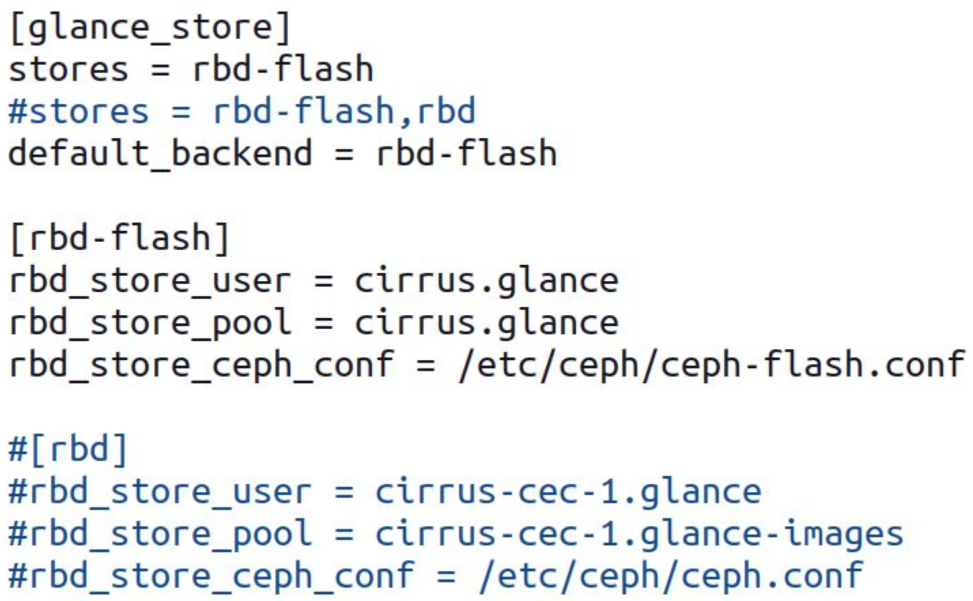
Deployment with Kolla-Ansible

KOLLA

an OpenStack Community Project

Configuration – custom override examples:
Configure 2 Glance storage backends

```
cat /etc/kolla/config/glance/glance-api.conf
```



```
[glance_store]
stores = rbd-flash
#stores = rbd-flash,rbd
default_backend = rbd-flash

[rbd-flash]
rbd_store_user = cirrus.glance
rbd_store_pool = cirrus.glance
rbd_store_ceph_conf = /etc/ceph/ceph-flash.conf

#[rbd]
#rbd_store_user = cirrus-cec-1.glance
#rbd_store_pool = cirrus-cec-1.glance-images
#rbd_store_ceph_conf = /etc/ceph/ceph.conf
```

Deployment with Kolla-Ansible

Configuration – custom override examples:
Configure Nova for GPU PCI passthrough on one node

```
cat /etc/kolla/config/nova/cirrus-c01/nova.conf
[PCI]
device_spec = { "vendor_id": "10de", "product_id": "26b9" }
passthrough_whitelist = { "vendor_id": "10de", "product_id": "26b9" }
alias = { "vendor_id":"10de", "product_id":"26b9", "device_type":"type-PF", "name":"l40s" }
```

```
cat /etc/kolla/config/nova/nova-api.conf
[pci]
alias: { "vendor_id":"10de", "product_id":"1eb8", "device_type":"type-PF", "name":"tesla-t4" }
alias: { "vendor_id":"10de", "product_id":"26b9", "device_type":"type-PF", "name":"l40s" }

[filter_scheduler]
enabled_filters = PciPassthroughFilter
available_filters = nova.scheduler.filters.all_filters
```

Deployment with Kolla-Ansible

We keep all configuration in a git repository

- I.e. everything in `/path/to/venv/share/kolla-ansible` is in a repo
- On the deploy machine, `/etc/kolla` is a symlink to `/path/to/venv/share/kolla-ansible/etc/kolla`
- This is to ensure we can trace our work and role back easily if required
- For SweSRC the repo lives here
- Our production configuration is in the “master”-branch
- The development cluster is in the “dev”-branch
- From here it is also easy to go back and forth between releases of OpenStack
    - Just keep in mind that an upgrade also overrides your edits to, e.g.
        - ./etc/kolla/globals.yaml
        - ./ansible/inventory/multinode
    - Unless you’re smart and put your edits on the stash before upgrading (which I didn’t….)

Deployment with Kolla-Ansible

Releases and upgrades

- OpenStack releases every 6 Months: April & October (YYYY.1 and YYYY.2)
  - April release is SLURP (Skip Level Upgrade Release Process); i.e. October can be skipped.
- Following the operator guide it’s as easy as

  ```
  pip install --upgrade git+https://opendev.org/openstack/kolla-ansible@stable/<version>
  ```

  Where <version> is, e.g., 2025.1

- Then install dependencies, pull the new images (`kolla-ansible pull`) and upgrade (`kolla-ansible upgrade`)