JSC Cloud
OpenStack user training

2024-02-20 | Sebastian Achilles, Prateek Gautam, Björn Hagemeier | Juelich Supercomputing Centre

Overview

- Split training in two parts
  - Introduction
  - Core OpenStack services
    - Authentication
    - Virtual Machine service (VM) / Nova: instantiation, life-cycle, advanced actions
    - VM Images / Glance: roll your own, public, private, shared
    - Networking / Neutron: internal networks, routers, security groups, access to internal VMs, firewall
    - Storage / Cinder: suggested handling of payload data, snapshots, backups

Overview

- Split training in two parts
  - Introduction
  - Core OpenStack services
    - Authentication
    - Virtual Machine service (VM) / Nova: instantiation, life-cycle, advanced actions
    - VM Images / Glance: roll your own, public, private, shared
    - Networking / Neutron: internal networks, routers, security groups, access to internal VMs, firewall
    - Storage / Cinder: suggested handling of payload data, snapshots, backups
  - Advanced OpenStack services
    - Orchestration / Heat: how it helps us as Cloud administrators, further use cases
    - Orchestration / Kubernetes:
    - Load balancing (LBaaS) / Octavia
    - VPN as a service (VPNaaS) / Neutron

Agenda

Time	Topic	Who
09:00	Authentication, basic setup, environment	Björn
09:30	VM service, Nova	Sebastian
10:00	VM Exercise	
10:15	VM images	Björn
10:30	*Break*	
10:45	Networking	Björn
11:15	Networking exercise(s)	
11:30	Storage, Cinder	Prateek
11:50	Storage exercise	
12:00	*Lunch break*	
13:00	Kubernetes / LoadBalancers	Tim
13:45	Heat	Björn
14:15	Heat Exercise	Björn
14:30	Wrap-up and Discussion	all

Introduction

Cloud

A definition

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., ***networks****,* ***servers****,* ***storage****,* ***applications****, and* ***services****) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models.*

– NIST Cloud definition

Cloud

Characteristics, service models, deployment models

Characteristics

- on-demand self-service
- broad network access
- resource pooling
- rapid elasticity
- measured service

Service models

- Software as a service (SaaS)
- Platform as a service (PaaS)
- Infrastructure as a service (IaaS)

Deployment models

- private
- community
- public
- hybrid

Cloud

Infrastructure and software stacks

Infrastructures

- Amazon Web Services (AWS): IaaS, PaaS, approximately 200 services (SaaS), Function as a service (FaaS) with AWS Lambda
- Google Cloud Platform (GCP): IaaS, PaaS, FaaS
- Microsoft Azure: IaaS, PaaS, SaaS
- OVH
- IONOS
- T-Systems

Software

- OpenStack: IaaS, PaaS
- OpenNebula: IaaS
- Apache CloudStack: IaaS

Cloud

Infrastructure and software stacks

Infrastructures

- Amazon Web Services (AWS): IaaS, PaaS, approximately 200 services (SaaS), Function as a service (FaaS) with AWS Lambda
- Google Cloud Platform (GCP): IaaS, PaaS, **FaaS**
- Microsoft Azure: IaaS, PaaS, SaaS
- OVH
- IONOS
- T-Systems

Software

- OpenStack: IaaS, PaaS
- OpenNebula: IaaS
- Apache CloudStack: IaaS

Cloud

Infrastructure and software stacks

Infrastructures

- Amazon Web Services (AWS): IaaS, PaaS, approximately 200 services (SaaS), Function as a service (FaaS) with AWS Lambda
- Google Cloud Platform (GCP): IaaS, PaaS, FaaS
- Microsoft Azure: IaaS, PaaS, SaaS
- OVH
- IONOS
- T-Systems

Software

- **OpenStack**: IaaS, PaaS
- OpenNebula: IaaS
- Apache CloudStack: IaaS

OpenStack

Overview – core services

Nova manages the lifecycle of virtual machines (VMs) that have

- a number of CPUs
- an amount of main memory
- storage: **system**, ephemeral, swap
- data storage: volumes
- network ports
- a template image containing an operating system

OpenStack

Overview – core services

Nova manages the lifecycle of virtual machines (VMs) that have

- a number of CPUs
- an amount of main memory
- storage: **system**, ephemeral, swap
- data storage: volumes
- network ports
- a template image containing an operating system

} Nova

OpenStack

Overview – core services

Nova manages the lifecycle of virtual machines (VMs) that have

- a number of CPUs
- an amount of main memory
- storage: **system**, ephemeral, swap
- data storage: volumes
- network ports
- a template image containing an operating system

} Nova

← Cinder

OpenStack

Overview – core services

Nova manages the lifecycle of virtual machines (VMs) that have

- a number of CPUs
- an amount of main memory
- storage: **system**, ephemeral, swap
- data storage: volumes
- network ports
- a template image containing an operating system

} Nova

← Cinder

← Neutron

OpenStack

Overview – core services

Nova manages the lifecycle of virtual machines (VMs) that have

- a number of CPUs
- an amount of main memory
- storage: **system**, ephemeral, swap
- data storage: volumes
- network ports
- a template image containing an operating system

} Nova

← Cinder

← Neutron

← Glance

OpenStack

Overview – core services

Nova manages the lifecycle of virtual machines (VMs) that have

- a number of CPUs
- an amount of main memory
- storage: **system**, ephemeral, swap
- data storage: volumes
- network ports
- a template image containing an operating system
- Keystone for service discovery and authentication

} Nova

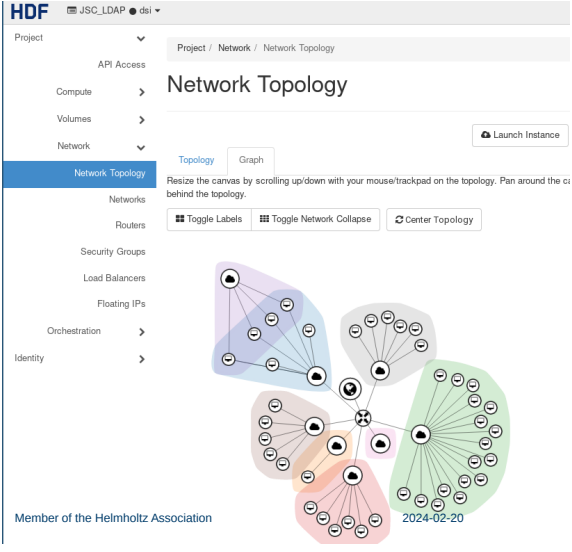
← Cinder

← Neutron

← Glance

Tutorial environment

We will use both web UI and CLI to run examples




```
(openstack-venv-p3) ChJoernh@zam035:~]$ os-dsi server show kolla-test-1
+-------------------------------------+-----------------------------------------------------------+
| Field                               | Value                                                     |
+-------------------------------------+-----------------------------------------------------------+
| OS-DCF:diskConfig                   | AUTO                                                      |
| OS-EXT-AZ:availability_zone         | HDFCloud                                                  |
| OS-EXT-STS:power_state              | Running                                                   |
| OS-EXT-STS:task_state               | None                                                      |
| OS-EXT-STS:vm_state                 | active                                                    |
| OS-SRV-USG:launched_at              | 2021-10-08T08:28:57.000000                                |
| OS-SRV-USG:terminated_at            | None                                                      |
| accessIPv4                          |                                                           |
| accessIPv6                          |                                                           |
| addresses                           | openstack-test=10.0.23.12; openstack-test-internal=10.0.25.12 |
| config_drive                        |                                                           |
| created                             | 2021-10-08T08:28:47Z                                      |
| flavor                              | x18 (82c859c6-4e2d-433b-9a75-ff58d8fc1f4e)                |
| hostId                              | b14117b945619f2854ab497a742dc0c67e8f9eccb8632765b3d1b9380  |
| id                                  | e8af7b3e-ec04-481c-b81e-b404cd9b0ac                       |
| image                               | CentOS 8 (bf79b3c3-975a-4ce1-aeee-6b9fc700eb25)           |
| key_name                            | devstack                                                  |
| name                                | kolla-test-1                                              |
| progress                            | 0                                                         |
| project_id                          | 8b97289e6a2d4d41b2aae1c4060aad99                          |
| properties                          |                                                           |
| security_groups                     | name='default'                                            |
| status                              | ACTIVE                                                    |
| updated                             | 2021-10-08T11:46:33Z                                      |
| user_id                             | 1027fa268cbe223f45e14b167685bf72f3c4e68e3765fd99095d1e30972e1748 |
| volumes_attached                    |                                                           |
+-------------------------------------+-----------------------------------------------------------+
```

Tutorial environment

CLI setup

We'll use a Python virtual environment. Run the following in your shell:

```
$ python3 -m venv openstack
$ source openstack/bin/activate
$ pip install python-openstackclient
```

For authentication:

- Option 1: Download and `source openrc.sh`
- Option 2: Download clouds.yaml, put it in one of
  - current working directory as `clouds.yaml` or
  - ~ `/.config/openstack/clouds.yaml`

Tutorial environment

CLI setup

We'll use a Python virtual environment. Run the following in your shell:

```
$ python3 -m venv openstack
$ source openstack/bin/activate
$ pip install python-openstackclient
```

For authentication:

- Option 1: Download and `source openrc.sh`
- Option 2: Download clouds.yaml, put it in one of
  - current working directory as `clouds.yaml` or
  - ~ `/.config/openstack/clouds.yaml`

Commandline Client

Introduction

- Help system accessible through `openstack help [command(s)]`
- The `--fit-width` or environment variable `CLIFF_FIT_WIDTH=1` helps improve readability by adjusting output width to terminal width
- List and pipe resources
- General resource operations: `create`, `delete`, `list`, `show`, `set`, `unset`, `add`, `remove`

```
$ openstack help
$ openstack help server list
$ openstack help server
server add fixed ip
server add floating ip
server add network
...
```

```
$ openstack server list --status
↪  ACTIVE -f value -c ID | xargs
↪  -n1 openstack server stop
```

Commandline Client

Introduction

EXERCISE

- Help system accessible through `openstack help [command(s)]`
- The `--fit-width` or environment variable `CLIFF_FIT_WIDTH=1` helps improve readability by adjusting output width to terminal width
- List and pipe resources
- General resource operations: `create`, `delete`, `list`, `show`, `set`, `unset`, `add`, `remove`

```
$ openstack help
$ openstack help server list
$ openstack help server
server add fixed ip
server add floating ip
server add network
...
```

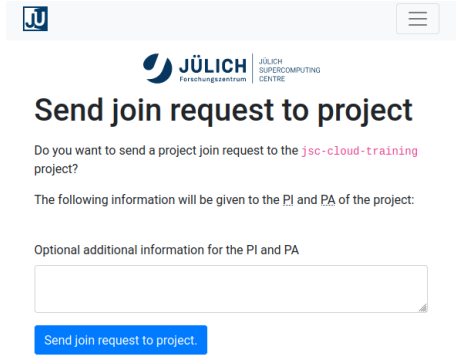
```
$ openstack server list --status
↪  ACTIVE -f value -c ID | xargs
↪  -n1 openstack server stop
```

Authentication and the JSC infrastructure

Join JSC Cloud project through JuDoor

Exercise

- Go to the JuDoor Join page: https://judoor.fz-juelich.de/projects/join/jsc-cloud-training
- Click on ”Send join request to project”



Authentication

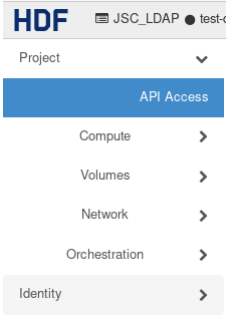
JSC Account

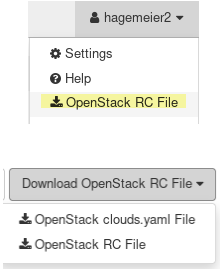
- For login using JSC account, use ”Keystone credentials” → ”JuDoor”
- Enter username and password as in JuDoor
- Other option: Helmholtz AAI federated login

Download `clouds.yaml`

Exercise

- Visit the Horizon dashboard: `https://cloud.jsc.fz-juelich.de/`
- Download credential files from the web interface
- Go to ”API Access”





Keystone Functions

Basic Concepts

- Identity
  - Tenant/Project : abstraction to isolate users and resources (VMs, Volumes, etc)
  - User : Person or service
  - Role : set of rights and privileges allow user to perform operations in a tenant
- Token
  - Randomly generated string used in HTTP headers
  - Authenticate and authorize interactions with the various OpenStack APIs
- Policy
  - Service based access policies for its resources
  - simple rule based mechanism for expressing authorization
  - eg. Policy for identity service can be configured in the /etc/keystone/policy.json
- Catalog
  - Provides an endpoint registry for various cloud services and used for discovering services' endpoints.

Exercise

Environment setup and authentication

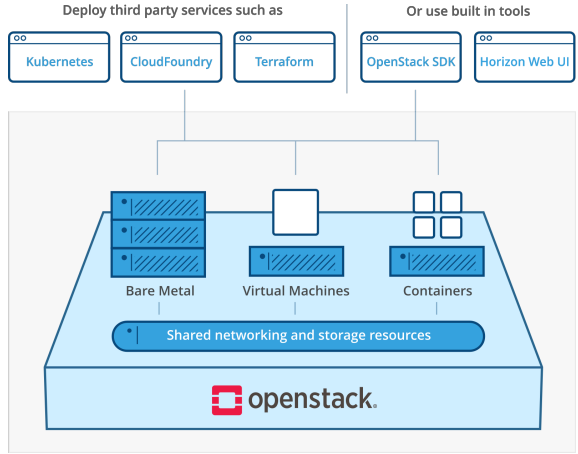
1. Login at `https://cloud.jsc.fz-juelich.de/`
2. Find and download credential files
3. Setup a virtual environment and install the OpenStack cli

Compute (Nova)

Nova

Compute service

- Nova is the OpenStack compute service
- Offers virtual machines (VMs)
- Publicly available or custom images
- Multiple virtualization backends, we use KVM (x86)



Source: OpenStack web site

Nova – server creation

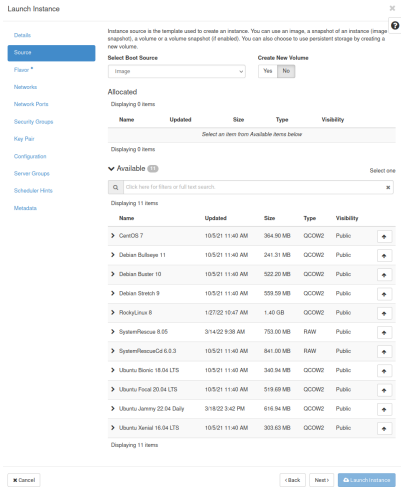
Ingredients

In general, you'll need the following to start a VM

- image (or volume)
- flavor
- name
- network
- keypair (optional)
- user-data (optional)
- server group (rarely used)

Nova – image

- Use one of the provided images, which contain the `cloud-init` packages for ssh key and user injection:
  - AlmaLinux 8.9,9.3
  - Debian Bullseye, Bookworm
  - Rocky Linux 8.9,9.3
  - Ubuntu 22.04
- Upload your own image.

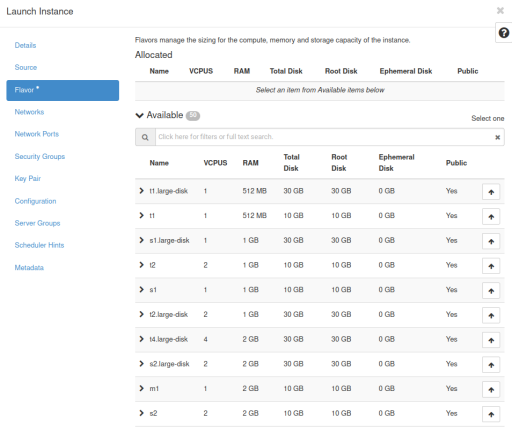


Nova – flavor

A flavor comprises

- number of VCPUs,
- amount of RAM,
- root disk size,
- ephemeral disk size,
- swap disk size, and
- RX/TX factor.

All parameters except VCPUs and RAM (and root disk size to some extend) are the same for all favors in our deployment.



Nova – flavor

RAM \ VCPUs	1	2	4	16
1 GB	SCS-1L:1:20n			
2 GB	SCS-1L:2:20n	SCS-2L:2:20n		
4 GB	SCS-1L:4:20n	SCS-2L:4:20n	SCS-4L:4:20n	
8 GB		SCS-2L:8:20n	SCS-4L:8:20n	
16 GB			SCS-4L:16:20n	SCS-16L:16:20n
32 GB				SCS-16L:32:20n
64 GB				SCS-16L:64:20n

Table: OpenStack flavors on the JSC-Cloud

Nova – flavor

Parameter	Value
root disk	20 GB
ephemeral disk	0 GB
swap disk	0 MB
RX/TX factor	1

Table: Fixed parameters for defined flavors

Nova – Create VM

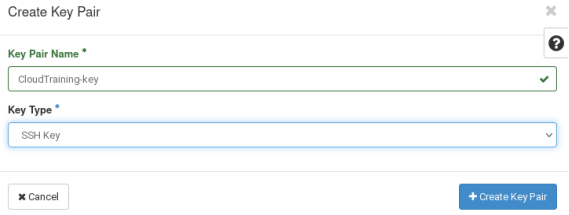
Steps when starting from a empty project:

1. Create a Security Group
2. Create a Network
3. Create a Router (not needed in JSC Cloud, since each project will come with one router)
4. Create a Key Pair
5. Create one (or more) Instances

Nova – Create a Key Pair

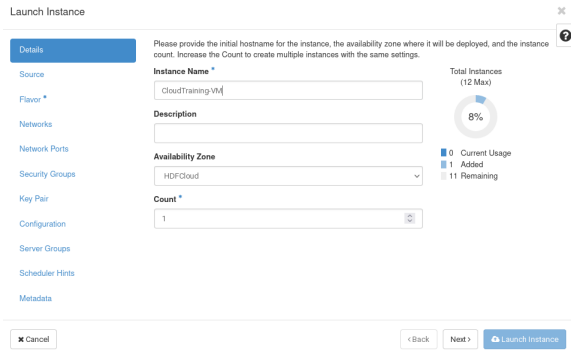
Add Interface

- Project → Compute → Key Pairs
- Create Key Pair
- Define a name for the key and select SSH as key type



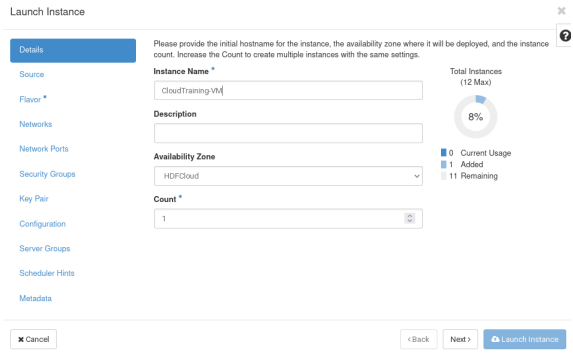
Nova – Create an Instance

- Project → Compute → Instances
- Launch Instance
- Define a name for the instance



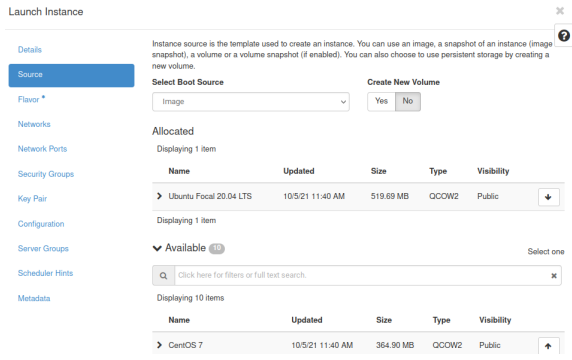
Nova – Create an Instance

- Project → Compute → Instances
- Launch Instance
- Define a name for the instance



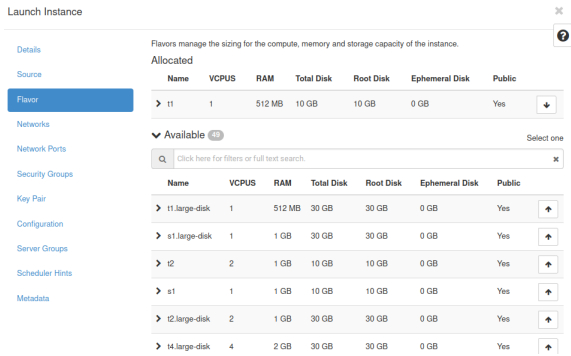
Nova – Create an Instance

- In Source select an image for you instance



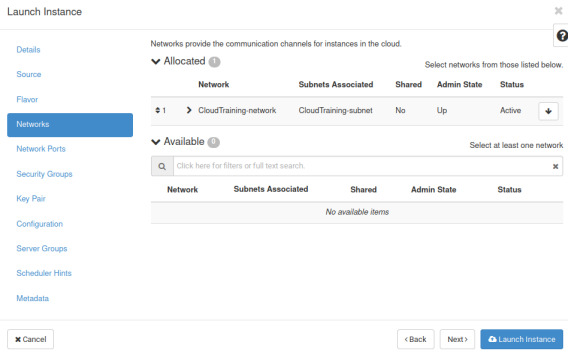
Nova – Create an Instance

- Select a flavor for your instance



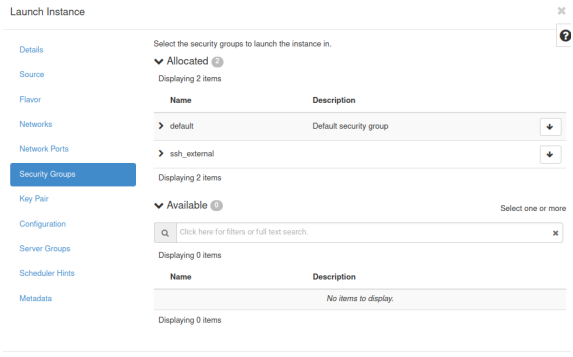
Nova – Create an Instance

- Select a network, e.g. the one you have created



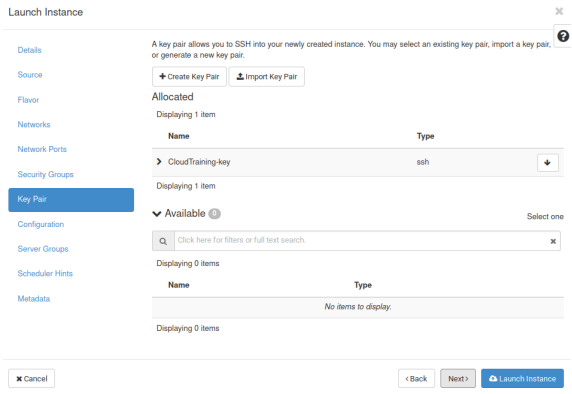
Nova – Create an Instance

- Select a Security Group, e.g. `ssh_external`



Nova – Create an Instance

- Select a Key Pair

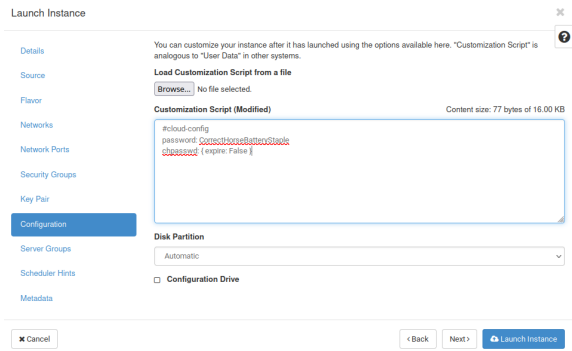


Nova – Create an Instance

- Adjust the customization Script
- For example you can set a password with cloud-config:

```
#cloud-config
password:
 ↪  CorrectHorseBatteryStaple
chpasswd: { expire: False
 ↪  }
```

- `#cloud-config` is a literal keyword
- cloud-init reference: `https://cloudinit.rtfd.io/`



Nova – Create an Instance

An excursion to cloud-init

Cloud-init is useful in the following scenarios

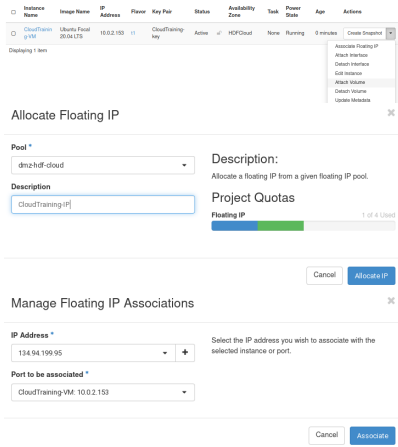
- initial user management
- packages & upgrades
- arbitrary files, e. g. for configuration
- commands to run on first boot
- mount points
- bootstrapping chef, ansible, puppet
- ⇒ check out the module reference

```
#cloud-config
package_upgrade: true

users:
  - name: root
    lock_passwd: true
  - name: bjoernh
    gecos: Bjoern Hagemeier
    lock_passwd: true
    sudo: ALL=(ALL)
     ↪  NOPASSWD:ALL
    shell: /usr/bin/zsh
    ssh-authorized-keys:
      - ssh-ed25519 AAAA...
```

Nova – Create an Instance

Associate Floating IP

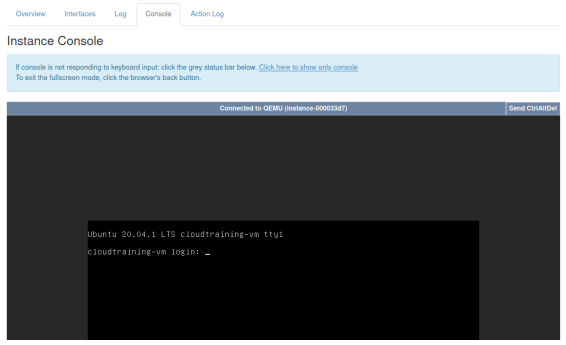
- Project → Compute → Instances
- For your instance → Actions → Associate Floating IP
- Click on + to allocate a floating IP
- Click on Associate



Nova – Create an Instance

Accessing the Instance through the OpenStack dashboard

- Project → Compute → Instances
- Select instance → Console
- default user-names depends on your image
  - `ubuntu`
  - `centos`
  - `rocky`



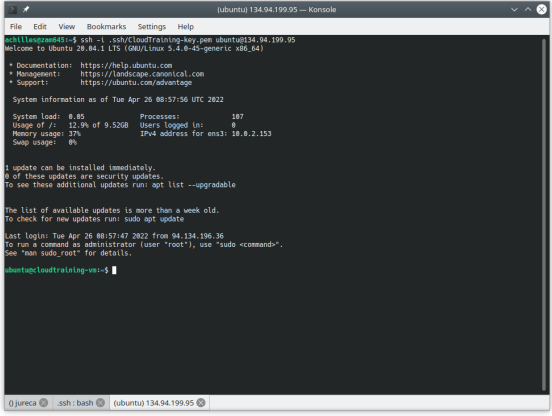

```
openstack console log show <server-name-or-id>
openstack console url show <server-name-or-id>
```

Nova – Create an Instance

Accessing the Instance through SSH

- For ssh you need to use the key pair you have added during the instance creation and the default user name depending on the image you used

```
ssh -i
 ↪   .ssh/CloudTraining-key.pem
 ↪   ubuntu@134.94.199.95
```



Nova – Create an Instance

Server groups

Server groups are a means to control server co-location or separation

	normal	**soft**
affinity	affinity	soft_affinity
anti_affinity	anti_affinity	soft_anti_affinity

```
openstack server group create --policy [soft-][anti-]affinity
 ↪  [--rule max_server_per_host=2] <groupname>
openstack server create --hint group=<server group name or id>
 ↪  ...
```

Special `--rule max_server_per_host=<number>` only for anti_affinity. Default is 1.

Nova – Delete an Instance

Steps to delete an instance and optional also the network are:

1. Delete the instance
2. Release floating IP
3. (Delete Subnet)
4. (Delete Network)
5. (Delete Security Groups)
6. (Delete Key Pair)

Please note: there should be one router per project, therefor it is not necessary to remove the router.

Nova – Using the CLI

Activate the Python venv you created earlier, e.g.:

```
source openstack/bin/activate
```

Gather parameters to launch an instance:

```
openstack flavor list
openstack image list
openstack security group list
openstack keypair list
```

Nova – Using the CLI

Configure access and security for instances

Either upload a public key or create a new key pair when specifying the filename for private key to save to:

```
openstack keypair create
  [--public-key <file> | --private-key <file>]
  [--type <type>] [--user <user>]
  [--user-domain <user-domain>]
  <name>
```

Create and manage security groups:

```
openstack security group list
openstack security group create [--description <description>] <name>
openstack security group delete <name>
```

Nova – Using the CLI

List security group rules:

```
openstack security group rule list
   [--all-projects]
   [--protocol <protocol>]
   [--ethertype <ethertype>]
   [--ingress | --egress]
   [--long]
   [<group>]
```

Nova – Using the CLI

Create and manage security group rules:

```
openstack security group rule create
   [--remote-ip <ip-address> | --remote-group <group>]
   [--dst-port <port-range> | [--icmp-type <icmp-type> [--icmp-code
    ↪  <icmp-code>]]]
   [--protocol <protocol>] [--ingress | --egress]
   [--ethertype <ethertype>] [--project <project> [--project-domain
    ↪  <project-domain>]]
   [--description <description>]
   <group>

openstack security group rule delete <rule>
```

Nova – Using the CLI

After you gather required parameters, run the following command to launch an instance. Specify the server name, flavor ID, and image ID:

```
openstack server create
  (--image <image> | --volume <volume>) --flavor <flavor>
  [--security-group <security-group>] [--key-name <key-name>]
  [--property <key=value>] [--file <dest-filename=source-filename>]
  [--user-data <user-data>] [--availability-zone <zone-name>]
  [--block-device-mapping <dev-name=mapping>]
  [--nic <net-id=net-uuid,v4-fixed-ip=ip-addr,v6-fixed-ip=ip-addr,
  ↪  port-id=port-uuid,auto,none>]
  [--network <network>] [--port <port>] [--hint <key=value>]
  [--config-drive <config-drive-volume>|True] [--min <count>]
  [--max <count>] [--wait]
  <server-name>
```

Nova – Using the CLI

After creating a server you can check the status with

```
openstack server list
  [--ip <ip-address-regex>]     [--ip6 <ip-address-regex>]
  [--name <name-regex>] [--instance-name <server-name>]
  [--status <status>] [--flavor <flavor>]
  [--image <image>] [--host <hostname>]
  [--all-projects] [--project <project>]
  [--user <user>] [--long] [-n|--no-name-lookup]
  [--marker <server>] [--limit <num-servers>] [--deleted]
  [--changes-since <changes-since>]
```

Nova – Using the CLI

Create and assign a floating IP using the CLI:

```
openstack floating ip list
openstack floating ip create dmz-jsc-cloud
openstack server add floating ip <server> <ip-address>
openstack server list
```

Disassociate and delete floating IP addresses:

```
openstack server remove floating ip <server> <ip-address>
openstack floating ip delete <floating-ip>
```

Nova – Using the CLI

Manage instances

Resize instance:

```
openstack server resize [--flavor <flavor> | --confirm | --revert]
  [--wait]
  <server>
```

Suspend and resume an instance:

```
openstack server suspend <server>
openstack server resume <server>
```

Reboot:

```
openstack server reboot [--hard | --soft] [--wait] <server>
```

Nova – Using the CLI

Delete instance:

```
openstack server list

openstack server delete <server>
```

Compute (Nova) Exercise

Nova Exercise

For this exercise:

- Use the exsisting router, network and security group `ssh_external` in the shared project,
- Create your own key following this naming scheme: `CloudTraining_key_<lastname>`
- In case we have more participants than floating IPs, please use the console in the OpenStack dashboard to work on your instance, instead of assigning a floating IP,
- Please release the resources at the end of the exercise.

If you get stuck or run into problems please do not hesitate and reach out to one of the tutors and ask for help!

Nova Exercise

1) Launch an instance using the OpenStack dashboard:

- `SCS-1L:1:20n` flavor
- `Ubuntu Focal 22.04 LTS` image

Benchmark your VM with the following command in your shell:

```
$ sudo apt update
$ sudo apt install sysbench
$ sysbench cpu
 ↪  --cpu-max-prime=20000 run
```

Delete your instance afterwards.

2) Launch an instance using the OpenStack CLI:

- `SCS-2L:2:20n` flavor
- `RockyLinux 9.3` image

Benchmark your VM with the following command in your shell:

```
$ sudo dnf -y install epel-release
$ sudo dnf -y update
$ sudo dnf install sysbench
$ sysbench cpu --threads=2
↪  --cpu-max-prime=20000 run
```

Delete your instance afterwards.

Compare the events per second you are measuring.

Images (Glance)

Images

- images are templates from which instance images (aka. VMs) can be created as a working copy
- frequently employ copy-on-write semantics to improve performance and save space
- Glance service provides discovery, registration and retrieval of VM images

Vanilla images

Official sources for common Linux distributions are:

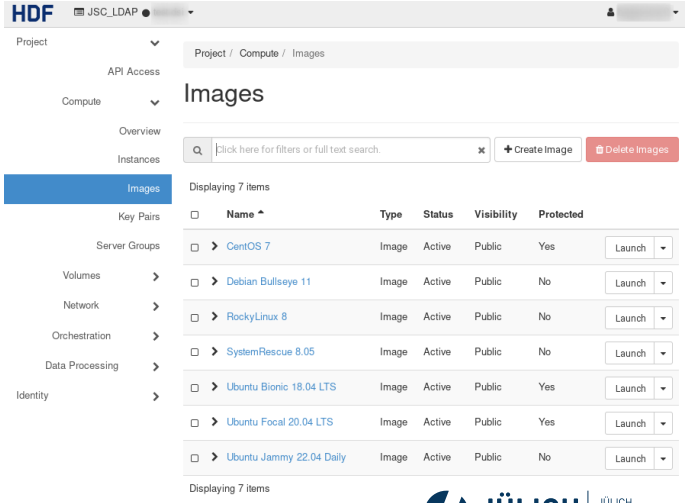
- Ubuntu: `https://cloud-images.ubuntu.com/`
- CentOS: `https://cloud.centos.org/centos/7/images`
- Debian: `https://cdimage.debian.org/cdimage/openstack` and `https://cdimage.debian.org/images/cloud/`
- Rocky: `https://rockylinux.org/alternative-images`
- SystemRescueCD: `https://www.system-rescue.org/Download/`

We already provide these images as public images in JSC Cloud.

Using the image service

Dashboard

- you will find the images under Project → Compute → Images
- launch VM from image
- create
- search
- modify



Visibility

public

- images visible to all projects, listed in default list
- can be booted by all projects
- can only be created by cloud administrators
- typical for vanilla distribution images as shown on previous slide

private (default)

- only visible, bootable within a single project

shared

- image can be shared with other projects explicitly (ACL)
- projects can ”accept” or ”reject” shared image
- accepted image in default list

community

- generally available, but not in default list
- also used for withdrawn or outdated images

Visibility

visibility	default image list	details	boot
public	all	all	all
private	project	project	project
shared	project, accepted member	project, member	project, member
community	accepted member	all	all

Details: https://wiki.openstack.org/wiki/Glance-v2-community-image-sharing#Accepting_a_.27Community.27_Image

Image formats

Disk and containers

Disk format

- raw
- vhd
- vhdx
- vmdk
- vdi
- iso
- ploop
- qcow2
- aki, ari, ami

Container format

- bare
- ovf
- aki, ari, ami
- ova
- docker
- compressed

Details: `https://docs.openstack.org/glance/yoga/user/formats.html`

Image formats

Disk and containers

Disk format

- raw
- ~~vhd~~
- ~~vhdx~~
- ~~vmdk~~
- ~~vdi~~
- ~~iso~~
- ~~ploop~~
- qcow2
- ~~aki, ari, ami~~

Container format

- bare
- ~~ovf~~
- ~~aki, ari, ami~~
- ~~ova~~
- ~~docker~~
- ~~compressed~~

Details: `https://docs.openstack.org/glance/yoga/user/formats.html`

Image metadata

- architecture=’amd64’,
- os_hash_algo=’sha512’,
- os_hash_value=’07c56a879bb0d79522...’,
- os_hidden=’False’,
- ...
- owner_specified.openstack.object=’images/Debian 11 Bullseye’,
- hw_disk_bus=’virtio’,
- hw_scsi_model=’virtio-scsi’

Generally, useful image properties can be found in the OpenStack documentation: `https://docs.openstack.org/glance/latest/admin/useful-image-properties`

Image metadata

- architecture=’amd64’,
- os_hash_algo=’sha512’,
- os_hash_value=’07c56a879bb0d79522...’,
- os_hidden=’False’,
- ...
- owner_specified.openstack.object=’images/Debian 11 Bullseye’,
- hw_disk_bus=’virtio’,
- ~~hw_scsi_model=’virtio-scsi’~~ ← not a good idea at JSC

Generally, useful image properties can be found in the OpenStack documentation: `https://docs.openstack.org/glance/latest/admin/useful-image-properties`

Image actions

list

```
$ openstack image list --public
...
$ openstack image list --community
...
$ openstack image list --property os_distro=ubuntu
+--------------------------------------+-------------------------+------
| ID                                   | Name
 ↪  | Status |
+--------------------------------------+-------------------------+------
| f0d4e282-3a12-43ac-8e8b-b1663399e9fe | Ubuntu Bionic 18.04 LTS
 ↪  | active |
| 149a65b5-aeb8-499f-aaa6-ec966bd28dd6 | Ubuntu Focal 20.04 LTS
 ↪  | active |
...
```

Image actions

create

```
$ wget
 ↪  https://download.cirros-cloud.net/0.5.2/cirros-0.5.2-x86_64-disk.im
$ openstack image create --private --file
 ↪  cirros-0.5.2-x86_64-disk.img --disk-format raw
 ↪  --container-format bare cirros-0.5.2-hagemeier2
$ openstack server create --flavor t1 --network internal --image
 ↪  cirros-0.5.2-bjoernh cirros-test-hagemeier2
$ openstack console log show cirros-test-hagemeier2
$ openstack console url show cirros-test-hagemeier2
```

Image actions

create


```
$ wget
 ↪  https://download.cirros-cloud.net/0.5.2/cirros-0.5.2-x86_64-disk.im
$ openstack image create --private --file
 ↪  cirros-0.5.2-x86_64-disk.img --disk-format raw
 ↪  --container-format bare cirros-0.5.2-hagemeier2
$ openstack server create --flavor t1 --network internal --image
 ↪  cirros-0.5.2-bjoernh cirros-test-hagemeier2
$ openstack console log show cirros-test-hagemeier2
$ openstack console url show cirros-test-hagemeier2
```

Network (Neutron)

Networking

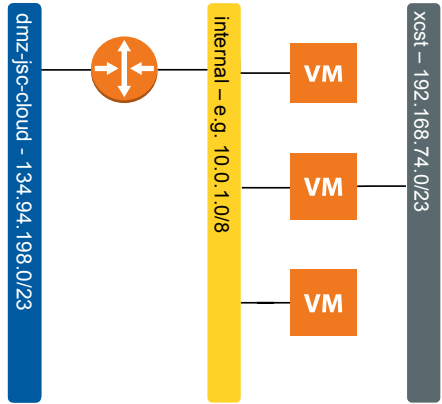
Specifics at JSC

- public network `dmz-jsc-cloud`
  - IPv4 addresses via **floating IPs**
  - subnet 134.94.199.0/24
  - each virtual router consumes an address
- project network `internal`
  - user defined, local IPv4 addresses
  - subnets must not overlap with public subnets
  - router provides NATing functionality through **masquerading** or **SNAT/DNAT**
  - think in terms of your network at home
- DATA storage network `xcst`
  - shared network controlled by Cloud administrators
  - direct connection to VM, will appear as additional network device
  - subnet 192.168.74.0/23

Networking

Specifics at JSC

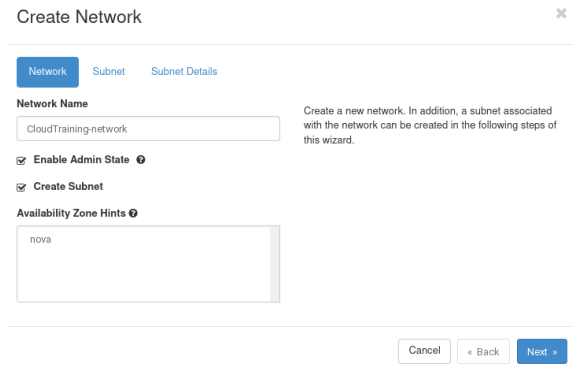
- **floating IPs** realized in router as DNAT/SNAT
- VMs without floating IPs not accessible from the outside and SNATed in outbound connections
- all new projects will be equipped with a **router** and **internal** network, such that you can immediately start working. JSC's **DNS servers** will be configured in the internal network



Network creation

Network

- Project → Network → Networks
- Set a name for the network, Admin status and Create subnet should both be ”enabled”.

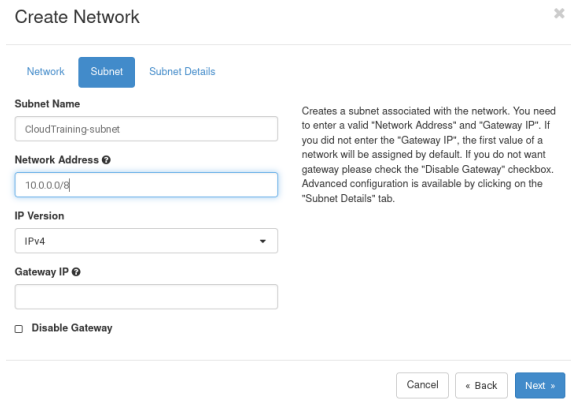



```
$ openstack network create test-net
$ openstack network delete test-net
```

Network creation

Subnet

- Set a name for the subnet
- The network address should be a private network range, such as `192.168.42.0/16` or `10.0.0.0/8`. This range is only available within your network and by the connected router (see next section)



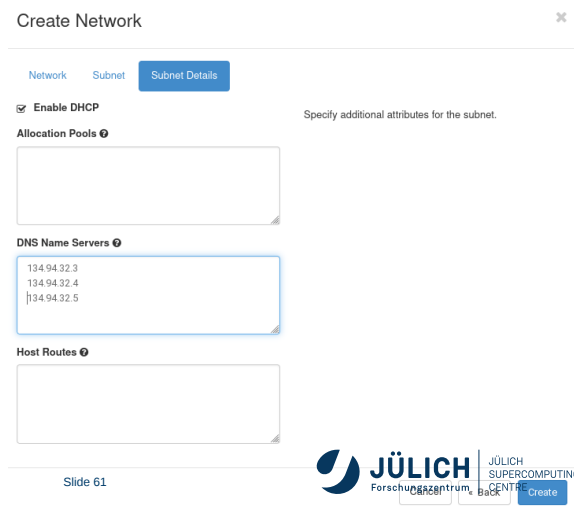

```
$ openstack subnet create --network test-net --subnet-range
 ↪  10.0.0.0/8 --dns-nameserver 134.94.32.3 test-subnet
```

Network creation

Subnet Details

- In the subnet details, it is important to set the following three DNS servers:
  - 134.94.32.3
  - 134.94.32.4
  - 134.94.32.5
- Add nameservers

```
$ openstack subnet set
 ↪  --dns-nameserver
 ↪  134.94.32.4
 ↪  --dns-nameserver
 ↪  134.94.32.5
 ↪  test-subnet
$ openstack subnet unset
 ↪  --dns-nameserver
 ↪  134.94.32.3
```

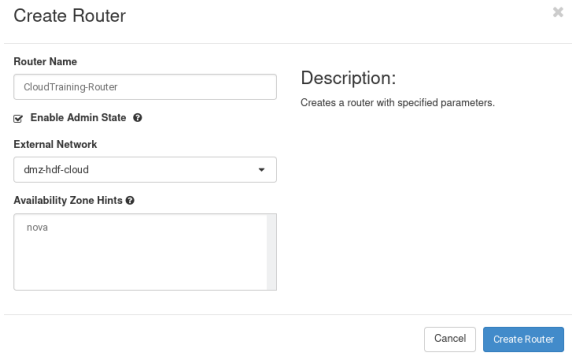


JÜLICH Forschungszentrum
JÜLICH SUPERCOMPUTING CENTRE

Router creation

Basic settings

- Project → Network → Routers
- Define a Name for the router, Admin status ”enabled”, and an External network, which for the JSC cloud will be ”dmz-jsc-cloud”.

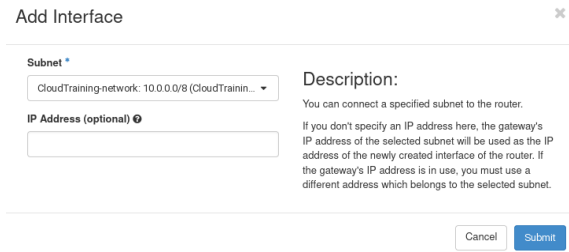



```
$ openstack router create --external-gateway dmz-jsc-cloud
 ↪  test-router
```

Router creation

Connection to subnet

- In Project → Network → Routers, select the router
- Interfaces → Add interface
- Add an interface in your internal network.




```
$ openstack router add subnet test-router test-subnet
```

Ports

- links between networks and VMs (or other parts of the infrastructure) are created by ports
- typically created automatically in the general case
- additional ports can be created an added to VMs in two ways

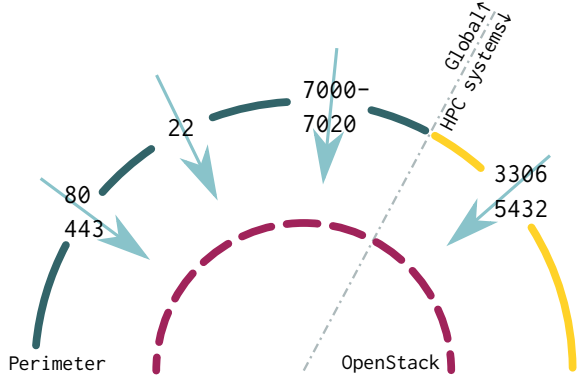
```
$ openstack server add fixed ip <server> <network>

$ openstack port create --network <network> [<port-name>]
$ openstack server add port <server> <port-name-or-id>
```

- useful for specific scenarios like participation in multiple networks or for specific ports w/o port security

Networking

Security Groups and perimeter firewall

- Security Groups provide a way to define sets of firewall rules based on
  - source and target IPs
  - source security group (other VMs)
  - protocol and port
  - assigned to VMs
- they are assigned to compute instances
- perimeter firewall allows
  - HTTP, HTTPS
  - SSH
  - 7000 - 7020
  - MySQL and PostgreSQL from HPC systems

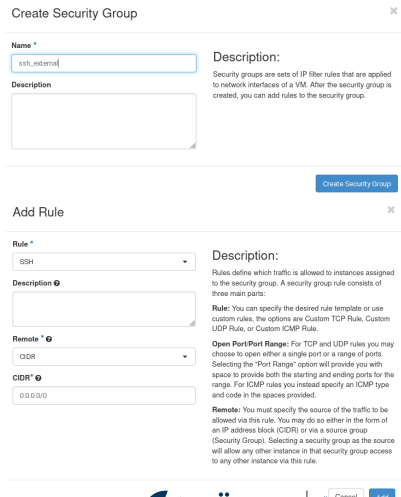


Security Groups

- Project → Network → Security groups
- Create a security group, e.g. `ssh_external`
- On Manage rules select a predefined rule, e.g. `SSH`
- Adjust CIDR: `0.0.0.0/0` for access from anywhere, or `134.94.0.0/16` from the FZJ network.

```
$ openstack security group create
 ↪  ssh_internal
$ openstack security group rule
 ↪  create --remote-ip 134.94.0.0/16
 ↪  --dst-port 22 --protocol tcp
 ↪  ssh_internal
```

- Further information: `https://docs.openstack.org/nova/latest/admin/security-groups.html`



Networking

SSH forwarding using ProxyJump

- public IPs must be used sparingly
- whenever possible, use one publicly available VM as gateway into your internal network
- please do not attempt this forwarding magic yourself
- from the SSH man page
- can be concatenated (comma separated)

```
-J destination
    Connect to the target host by first making a ssh connection
    ↪ to the jump
    host described by destination and then establishing a TCP
    ↪ forwarding to
    the ultimate destination from there. Multiple jump hops may
    ↪ be
    specified separated by comma characters. ...
```

Cinder (volume)

Cinder - Introduction

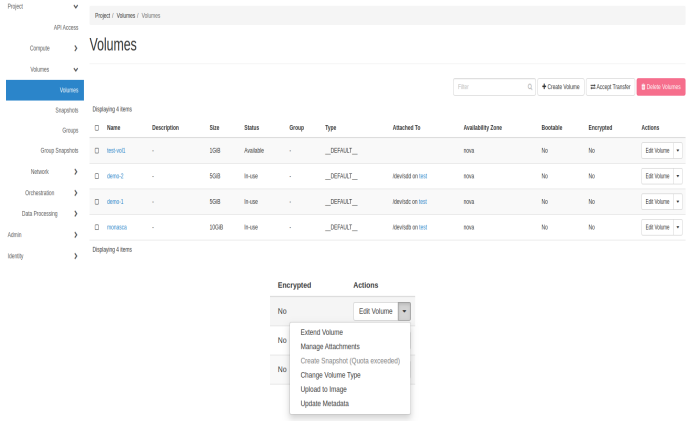
- Block storage service for OpenStack.
- Provides software defined block storage via abstraction and automation on top of various backend storage devices (NFS at Juelich)
- Provisions and manages block devices known as Cinder volumes.
- Volumes lifecycle is independent of VMs.

- Volume types:
  - a group of volume policies: provision type, compression etc.
  - we are offering
    - DEFAULT

- Operations:
  - Create/Delete Volumes
  - Use volume with VM
  - Create/Delete Snapshot from Volumes
  - Create Volume from Snapshot
  - Extend Volume

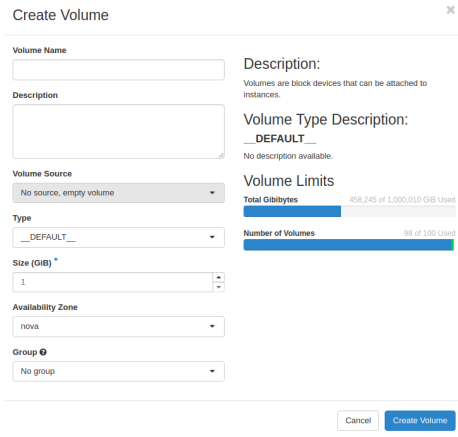
Cinder - Dashboard

- You will find the volumes under Project → Volumes → volumes
- Create Volumes
- Search
- Delete Volumes
- Volume Actions (Edit Volume)



Cinder - Volume Creation

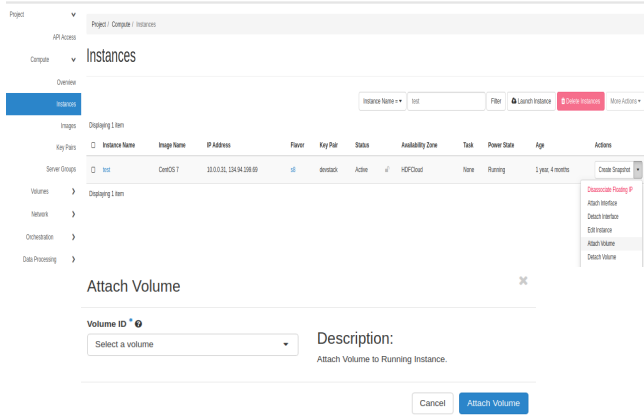
- Volume creation fields:
  - Volume Name
  - Description : short description about volume
  - Volume source:
    - Empty source
    - Snapshot
    - Image
    - Volume
  - Type : volume type
  - Size : 1-max (project quota)
  - Availibity Zone: Nova (default)
  - Groups



Cinder - Attach volume to a VM

- Project → Compute → Instances → Choose VM (Action) → Attach Volume

- Select your volume to attach



Cinder - Mounting volume device in VM (1/2)

SSH/login to your VM
Check volume/block device is attached to the VM

```
ubuntu@ubuntu-exp:~$ lsblk
NAME    MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
vda     253:0    0    10G  0 disk
|-vda1  253:1    0    10G  0 part /
vdb     253:16   0     5G  0 disk
```

Format the disk with preferred filesystem

```
sudo mkfs.ext4 /dev/vdb
```

Cinder - Mounting volume device in VM (2/2)

List the UUID of the partition

```
ubuntu@ubuntu-exp:~$ lsblk -f
NAME   FSTYPE LABEL UUID                                 MOUNTPOINT
vdb    ext4         f3bb28e8-5143-41f9-9684-fc48cbaba3ca
```

Mount the file system

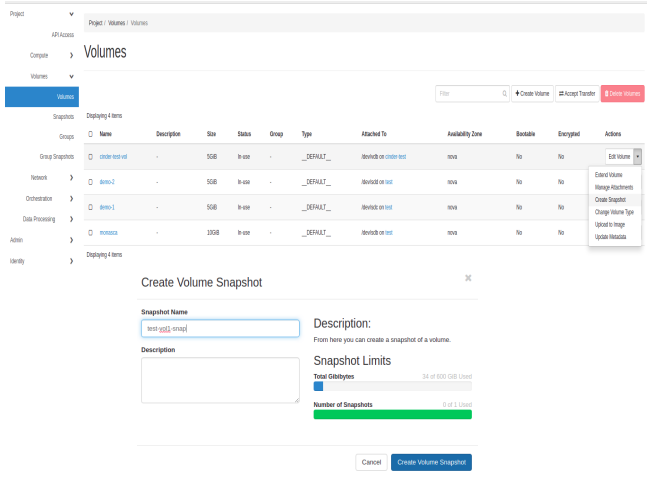
```
sudo mkdir -p /mnt/vol-attach
sudo echo "UUID=f3bb28e8-5143-41f9-9684-fc48cbaba3ca /mnt/vol-attach
 ↪  ext4 defaults 0 0"|sudo tee -a /etc/fstab
sudo mount -a
```

Verify

```
ubuntu@ubuntu-exp:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/vdb1       5.0G   33M  5.0G   1% /mnt/vol-attach
```

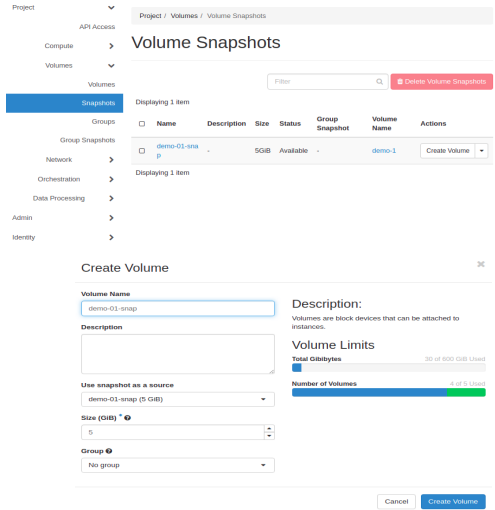
Cinder - Creating Snapshots from Volume

- Select **Create Snapshot** action of the target volume.
- Provide a Snapshot name for the snapshot and click **Create Volume Snapshot**.



Cinder - Creating Volume from Snapshots

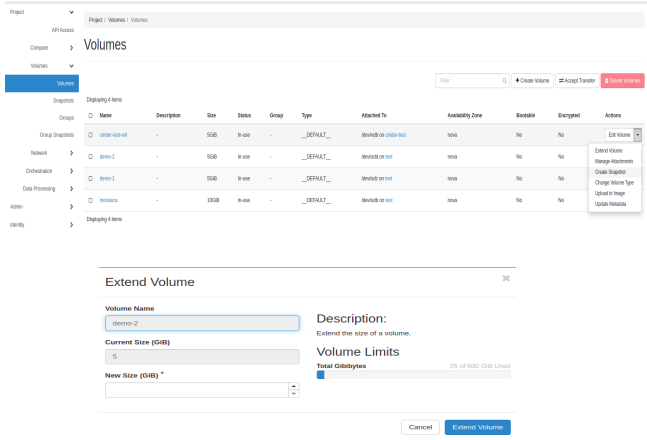
- You will find the snapshots under Project → Compute → Volumes → snapshots
- Select the **Create Volume** action of the target snapshot.
- Provide a volume name for the volume and click **Create Volume**.



Cinder - Resize Volume

- Detach Volume from the VM
- From volume action drop down menu choose **Extend Volume**

- New Size = current Size + extra size



Cinder - Using the CLI

Create Volume

```
openstack volume create --size  size  <VOLUME NAME>
openstack volume list
openstack volume show <VOLUME>
```

Delete Volume

```
openstack volume delete  <VOLUME>
```

Attach your volume to a server

```
openstack server add volume <VM>  <VOLUME>
```

Cinder - Using the CLI

Create Snapshot from the volume

```
openstack volume snapshot create   --volume <VOLUME> --description
↪  <description>  <SNAPSHOT NAME>
openstack volume snapshot list
openstack volume snapshot show  <SNAPSHOT>
```

Delete Snapshot

```
openstack volume snapshot delete <SNAPSHOT>
```

Resize a volume

```
openstack server remove volume <VM>  <VOLUME>
openstack volume set <VOLUME> --size <SIZE>
```

Reference : https://docs.openstack.org/python-openstackclient/zed/cli/command-objects/volume.html

Cinder Exercise

If you get stuck or run into problems please do not hesitate and reach out to one of the tutors and ask for help!

1. Create a Volume of size 2G
   - Find out volume-ID and list other metadata

2. Attach Volume to your Virtual Machine (VM)
   - Use previously created VM or create a new VM

3. Mount attached volume
   - Create ext4 filesystem on the disk
   - Create new directory /mnt/vol-attach in the VM
   - Mount newly created filesystem at /mnt/vol-attach

Cinder Exercise

4. Install fio (flexible I/O tester) tool.

```
sudo apt install fio  or sudo yum install fio
```

5. Fill 1G volume space and also perform I/O performance test

```
cd  /mnt/vol-attach
sudo fio --name=randomwrite --ioengine=libaio --iodepth=1
 ↪  --rw=randwrite  --bs=4k --direct=0 --size=1G --numjobs=1
 ↪  --runtime=30 --group_reporting
```

6. Check usage of the filesystem

```
df -h
```