



Leibniz Supercomputing Centre
of the Bavarian Academy of Sciences and Humanities

The background of the slide is a photograph of a modern, multi-story building with a facade of vertical metal slats. The image is overlaid with a semi-transparent blue filter. In the foreground, there are trees and a sidewalk. A dark blue rectangular box is positioned in the lower-left area of the image, containing the title text in white.

LRZ Compute Cloud for AI support

What includes in the course



First steps

10:00 - 10:45

- A brief introduction to cloud computing
- Meet the LRZ Compute Cloud



Hands-on sessions

11:00 - 12:00

- Extending the LRZ AI capabilities with the Compute Cloud

..... Lunch break

13:00 - 14:15

- Databases and the LRZ Compute Cloud
- Managing the LRZ Compute Cloud from the command line



14:30 - 15:45

- Secure AI workflow with LRZ middleware

Final touch

15:45 - 16:00

- Wrap-up

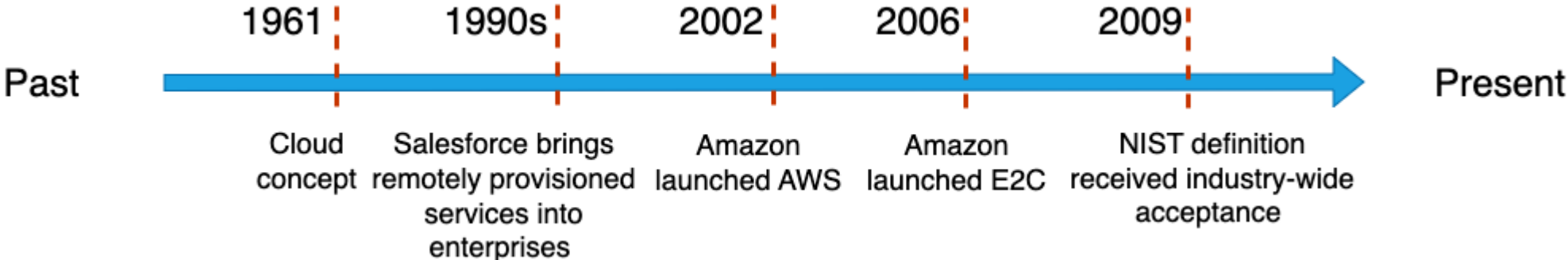
A Brief History of Cloud

Birth of Cloud

Tech giants such as Amazon, Google decides to rent out their machines in an off-season to avoid money draining by the idle machines.



Main timeline of Cloud history



Six Fun Facts of Cloud

1. Banking has the most activity in the history of cloud computing
2. China has the largest data center in the world:
China Telecom Data Center located at Hohhot, size 100 hectares (~140 football fields)
3. A data center from world top 10 largest centers consumes as much power as a mid-sized town
4. Cooling systems consume ~70% of the total energy cost of a data center
5. Amazon dominates the Cloud Market: 32%, Q1 2023
6. ~34 percent of workers prefer working in the cloud and will look for a new job if required to return to the office



Hohhot



What is the Cloud Computing for you?

- Stay quietly in the background until something goes wrong...
- On-demand delivery of resources, e.g. compute resources (CPU, GPU), storage, databases.
- Quickly provisioned, access remotely,
- Pay as you go: idea for individuals, small/middle-sized business
- Flexible environments
-

It is not about
the servers, but
the services

Cloud Computing definition:

Cloud computing is a specialized form of distributed computing that introduces utilization models for remotely provisioning scalable and measured resources*

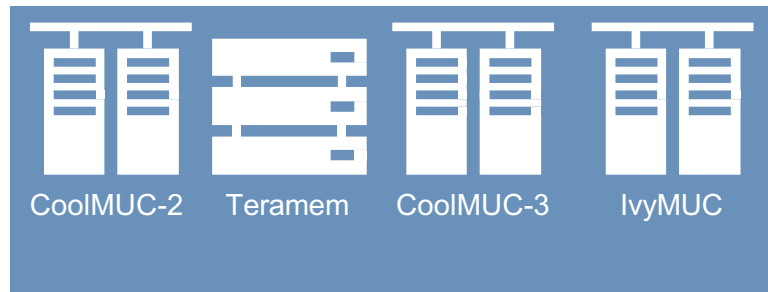
* Thomas, E., Zaigham, M. and Ricardo, P., 2013. *Cloud Computing Concepts, Technology & Architecture*. Prentice Hall.

High Level Overview of Main LRZ Resources

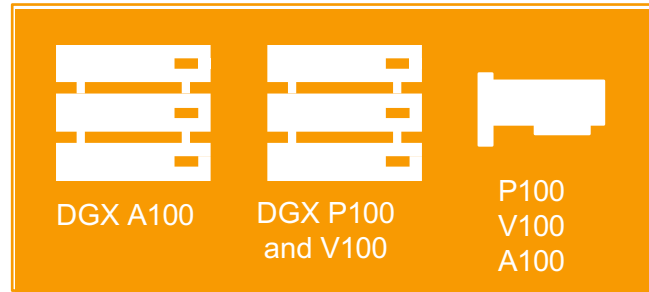


*Slurm: HPC workload manager

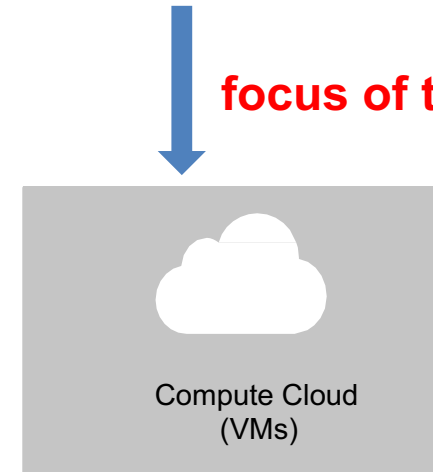
*OpenStack: Cloud orchestrator



HPC systems (Slurm)



AI Systems (Slurm)



Compute Cloud (OpenStack)



Data Science Storage

AI Ready

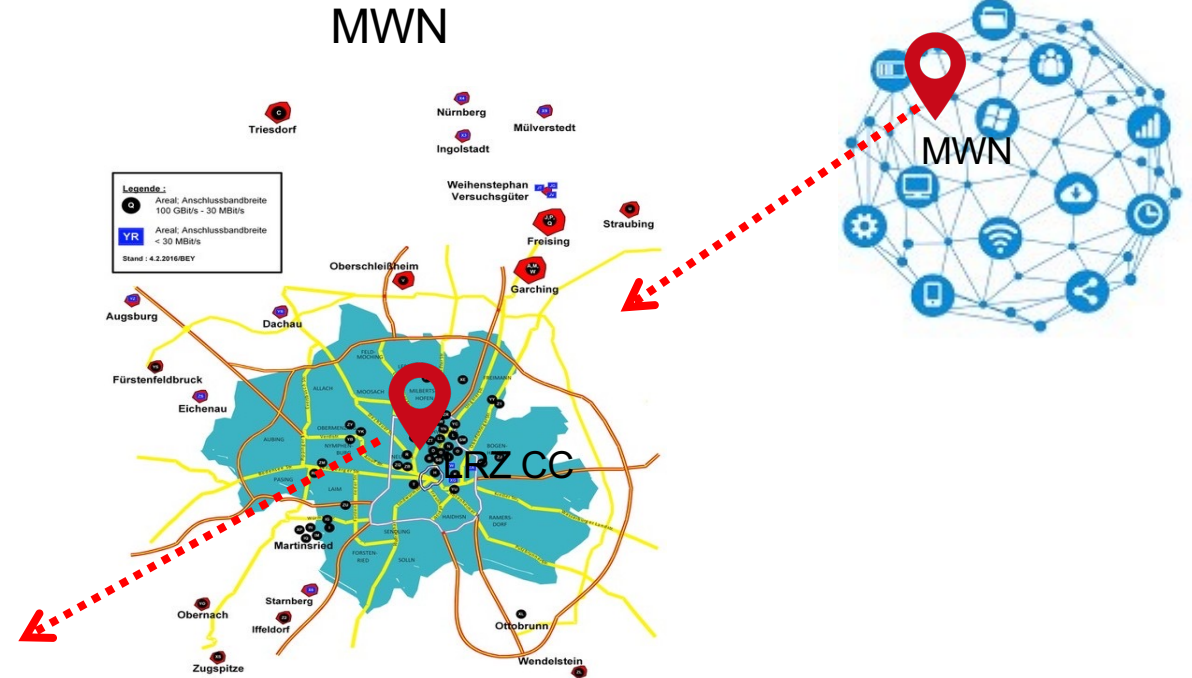
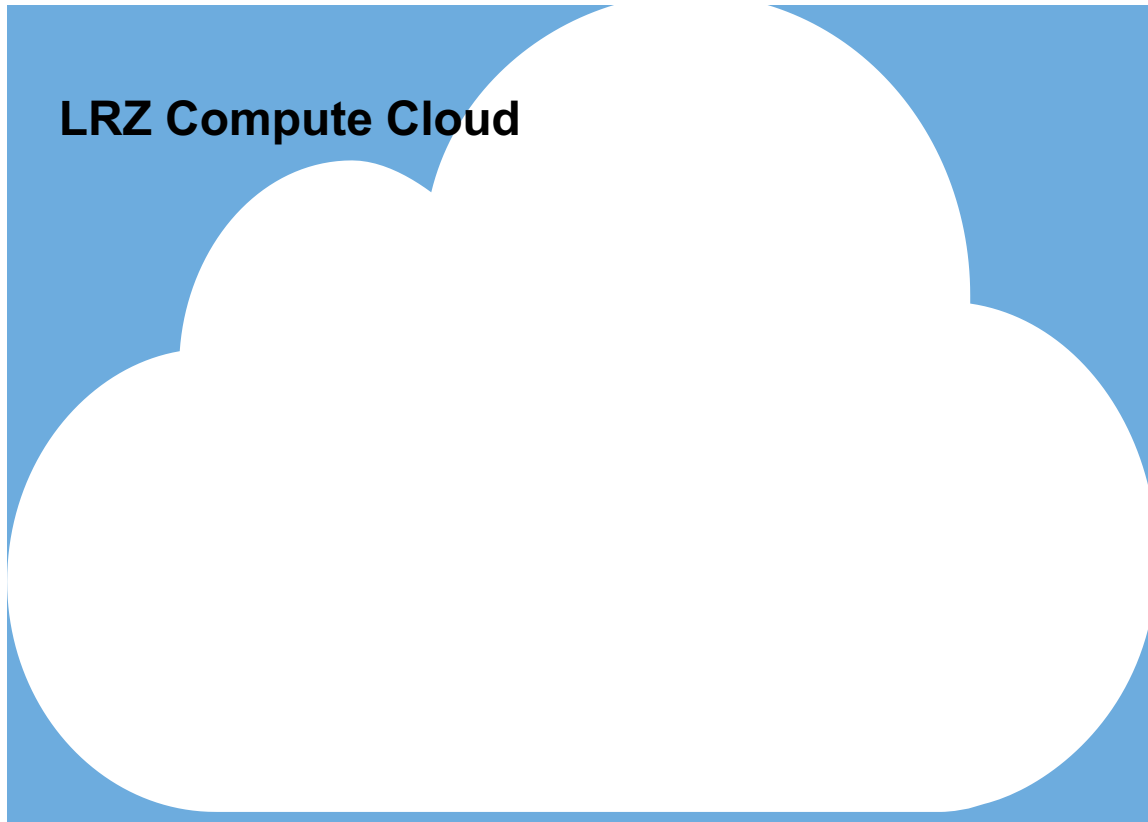


Interfaces to use Slurm and OpenStack are very different!

The LRZ Compute Cloud at a Glance

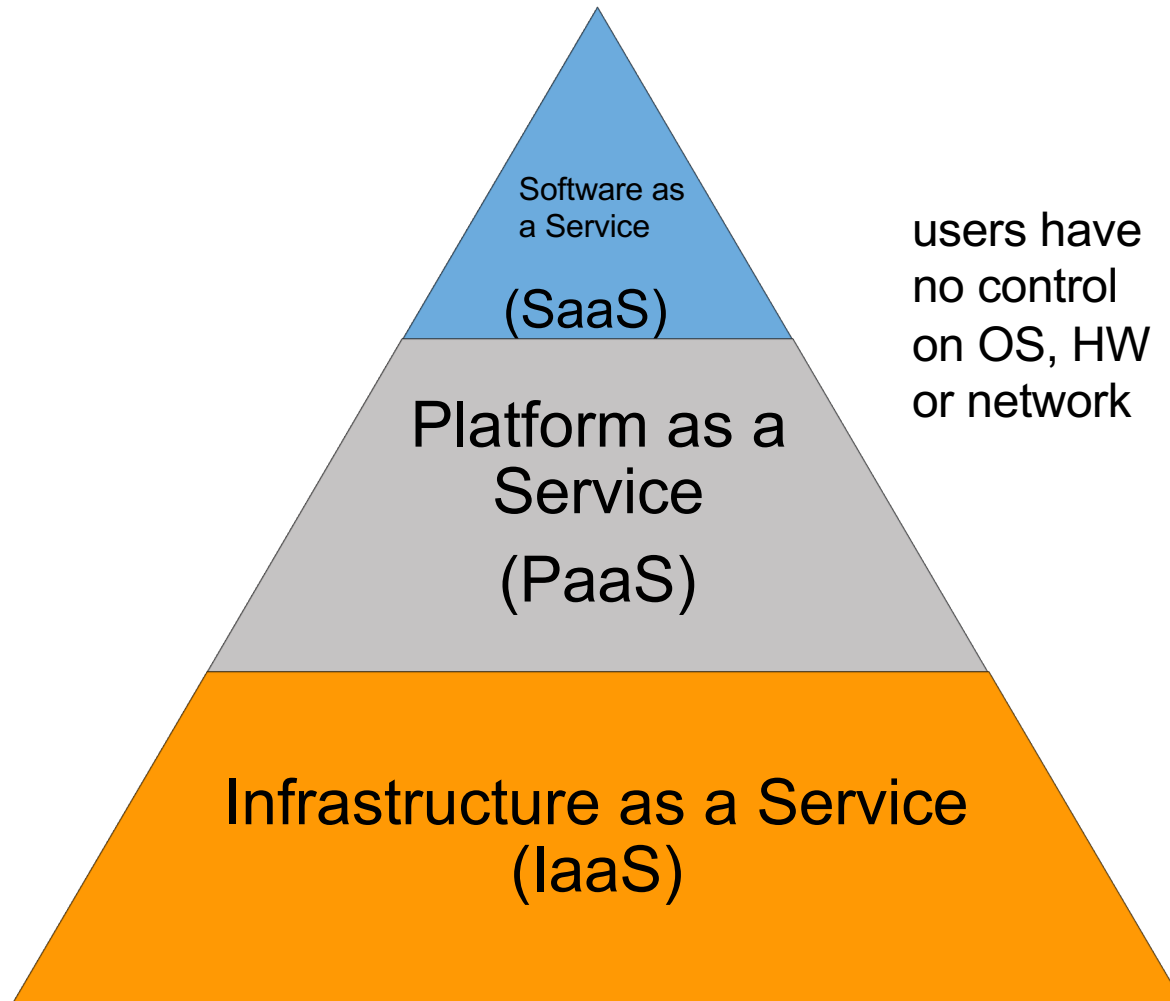


Internet



The LRZ compute cloud located within MWN

Cloud Services



- SaaS – Fully developed software solution to be used, e.g. Google drive, Tiktok (to take online orders).
- PaaS – Provides an environment for an application: provides a framework on top of which to build, deploy, and manage software products, e.g. Google APP engine
- IaaS - Provides a completely virtualized computing infrastructure provisioned and managed over the internet, e.g. Amazon Elastic Compute Cloud (Amazon EC2), [LRZ Compute Cloud](#)

Further reading: Cloud Computing Characteristics*

A consumer can request and receive access to a service offering without an administrator or supporting staff having to fulfil the request manually.

On-demand

Cloud services should be easy to access. Services are not locked to devices, but can be accessed whenever and wherever users want.

**Broad
network
access**

Ability to grow with user demand. It should be relatively easy for the provider to add more resources. Users have "infinite" resources!

**Resource
elasticity**

Cloud services must have the ability to measure usage. Usage can be quantified using various metrics, such as time, bandwidth used, and data used. This ability to measure allows what is known as pay as you go model.

**Measured
service**

When resources (e.g. memory, CPU, network bandwidth) are not used, they should be released, which can be used by others.

**Resource
pooling**

*Xiao, Z. and Xiao, Y., 2012. Security and privacy in cloud computing. *IEEE communications surveys & tutorials*, 15(2), pp.843-859.

A List of Cloud Orchestrators

- **OpenStack**



- Kubernetes (k8s)



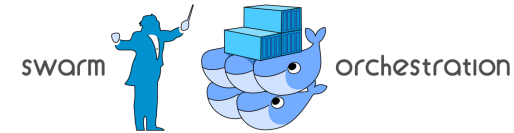
- Apache Mesos, Apache YARN



- Ansible



- Docker Swarm



- ...

Your homework:

What are the differences? Which one to choose? Which one is better maybe?

OpenStack: The Engine of the LRZ Compute Cloud

- What do we need for transforming a set of resources (data center) into a cloud?
 - to manage/admin the hardware
 - to provision machines to users
 - to allow users to authenticate
 - to manage the network across resources
 - ...

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 mechanism.

- OpenStack bundles together a bunch of different technologies, addressing the different needs transforming resources into a Cloud Service

OpenStack - Terminology

- Image

A single file which contains a virtual disk with a bootable operating system installed on it. Images are like a template of a computer's root drive. They contain the operating system and can also include software and layers of your application, such as database servers, web servers, and so on.



FreeBSD



CentOS

OpenStack - Terminology

- Instance or server

A copy of an image running as a virtual server the cloud. We will also call it server.

servers always in a status out of ..

ACTIVE: The server is active.
BUILD: The server has not yet finished the original build process.
DELETED: The server is deleted.
ERROR: The server is in error.
HARD_REBOOT: The server is hard rebooting. This is equivalent to pulling the power plug on a physical server, plugging it back in, and rebooting it.
MIGRATING:
PASSWORD:
PAUSED:
REBOOT: The server is in a soft reboot state. A reboot command was passed to the operating system.
REBUILD:
RESCUE:
RESIZE:
REVERT_RESIZE:
SHELVED: The server is in shelved state. Depends on the shelve offload time, the server will be automatically shelved off loaded.
SHELVED_OFFLOADED: The shelved server is offloaded (removed from the compute host) and it needs unshelved action to be used again.
SHUTOFF: The server was powered down by the user, either through the OpenStack Compute API or from within the server.
SOFT_DELETED:
SUSPENDED:
UNKNOWN:
VERIFY_RESIZE:

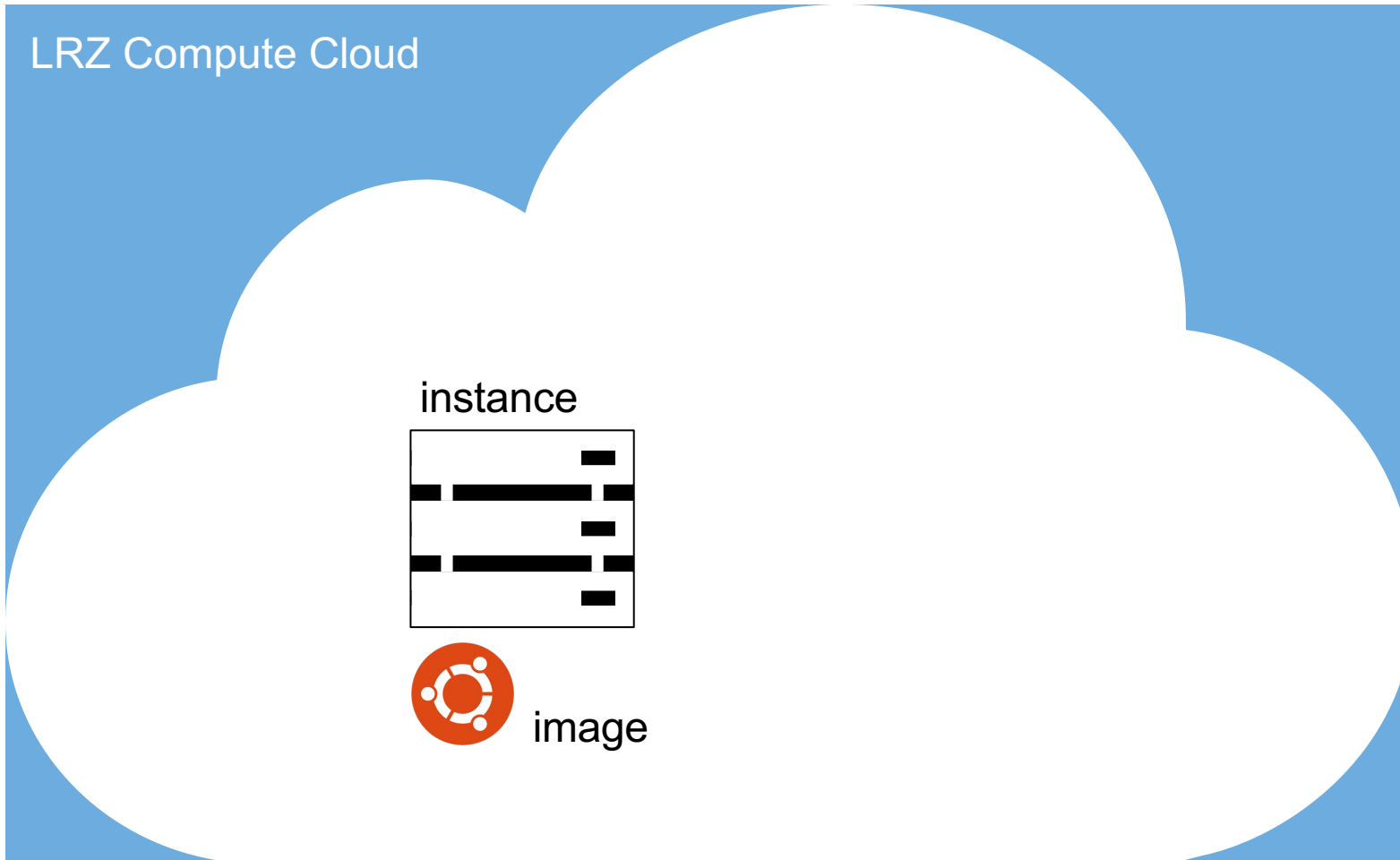
OpenStack - Terminology

- Flavor

Flavors define the compute, memory, and storage capacity of instances. To put it simply, a flavor is an available hardware configuration for a server.

Name	vCPUs	RAM	Remarks	Access
tiny	1	512 MB	for testing purposes only, most Operating Systems will not boot due to restricted resources	public
nvidia-v100.2	40	700 GiB	use 2 GPUs on a GPU node (use entire GPU node)	restricted, contact us
nvidia-v100.1	20	350 GiB	use 1 GPU on a GPU node	restricted, contact us
lrz.xlarge	10	47.5 GiB	use 1/4 compute node	public
lrz.xhuge	48	1488 GiB	use 1/4 of the hugemem node	restricted, contact us
lrz.small	1	4.75 GiB	use 1/40 compute node	public
lrz.medium	2	9.5 GiB	use 1/20 compute node	public
lrz.large	4	19 GiB	use 1/10 compute node	public
lrz.huge	24	744 GiB	use 1/8 of the hugemem node	restricted, contact us
lrz.4xlarge	40	190 GiB	use entire compute node	restricted, contact us
lrz.4xhuge	192	5952 GiB	use entire hugemem node	restricted, contact us
lrz.2xlarge	20	95 GiB	use 1/2 compute node	restricted, contact us
lrz.2xhuge	96	2976 GiB	use 1/2 of the hugemem node	restricted, contact us

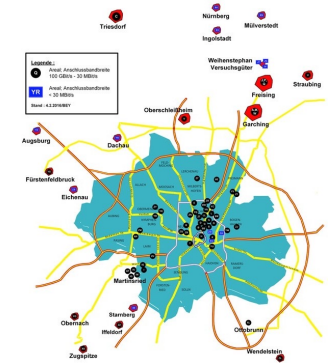
The LRZ Compute Cloud at a Glance



Internet



MWN

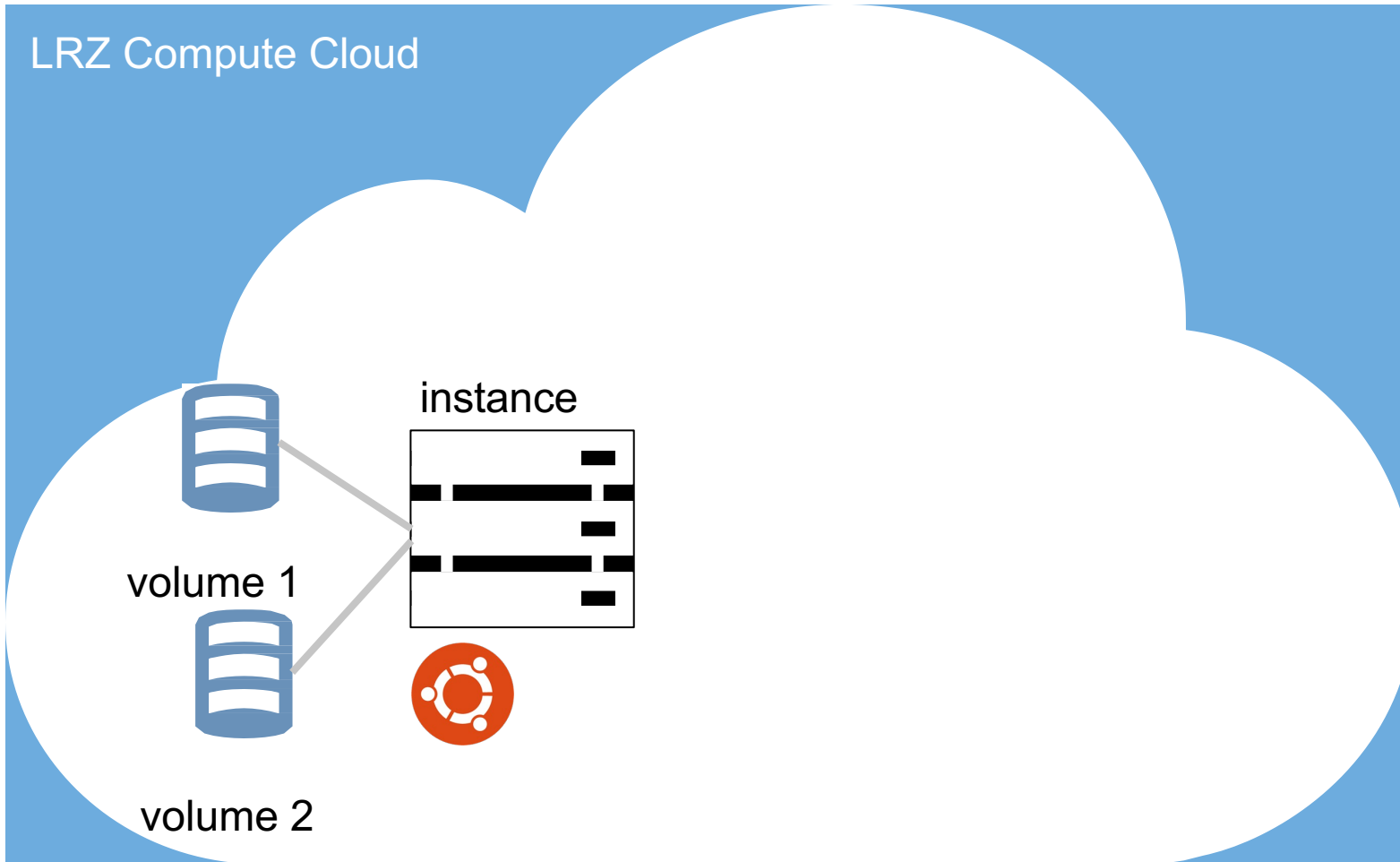


OpenStack - Terminology

- Volume

A volume is a detachable block storage device, similar to a USB hard drive. You can attach a volume to only one instance. But an instance can attach several volumes

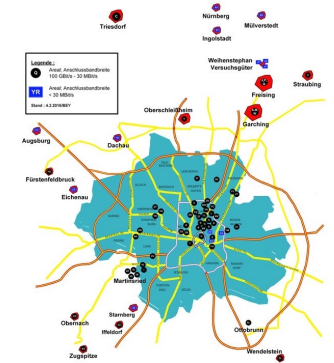
The LRZ Compute Cloud at a Glance



Internet



MWN



OpenStack - Terminology

- **Networking**

OpenStack provides networks, subnets, and routers as object abstractions. Each abstraction has functionality that mimics its physical counterpart: networks contain subnets, and routers route traffic between different subnets and networks. Instances are created within internal private networks. These networks can be routed to external networks (e.g., Internet or MWN) via a virtual router.

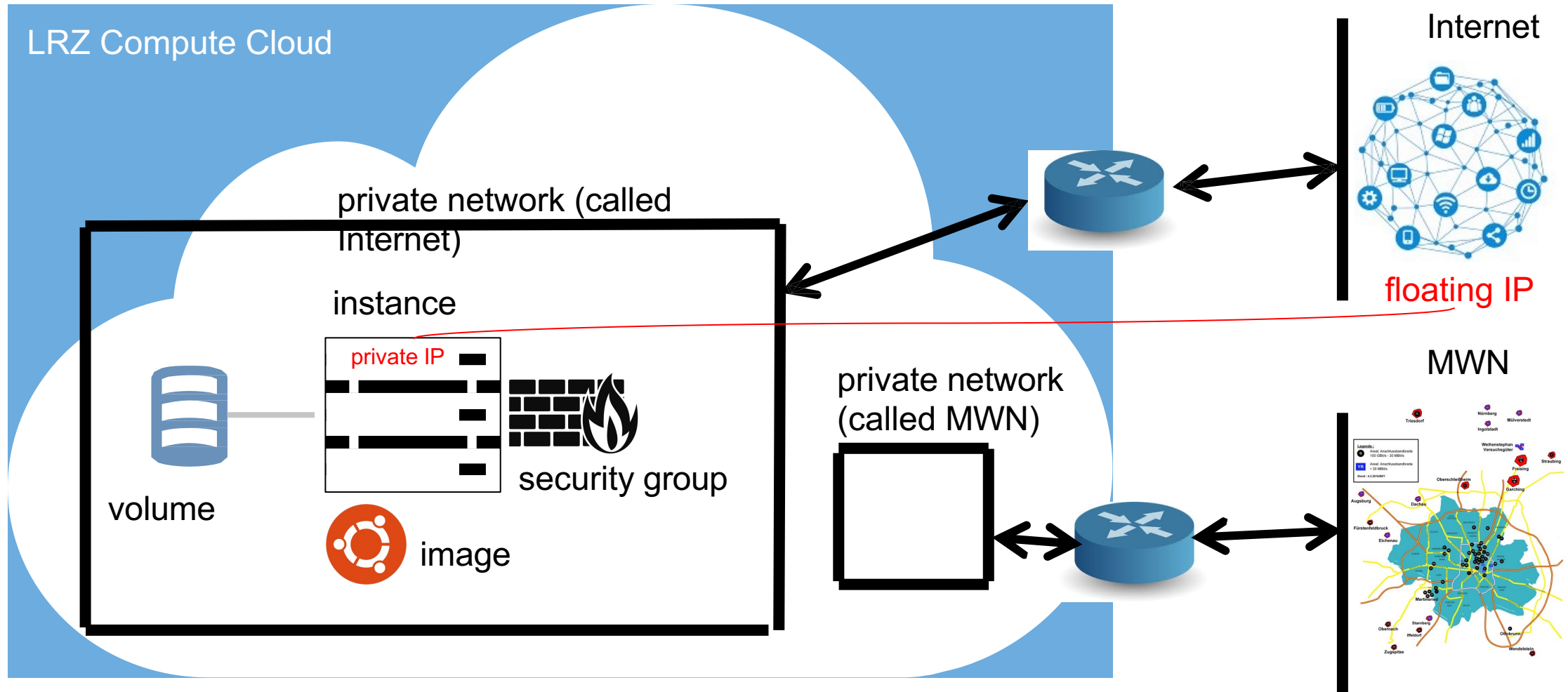
- **Private and Floating IP**

Each instance has a fixed IP within its private Network. That IP can be associated to an IP of the external network that network is connected by means of what it is called *floating IP address*. The floating IP address will allow addressing the instance from the outside.

- **Security group**

A security group acts as a virtual firewall for servers and other resources on a network. It is a container for rules for allowing different types of network traffic to and from an instance.

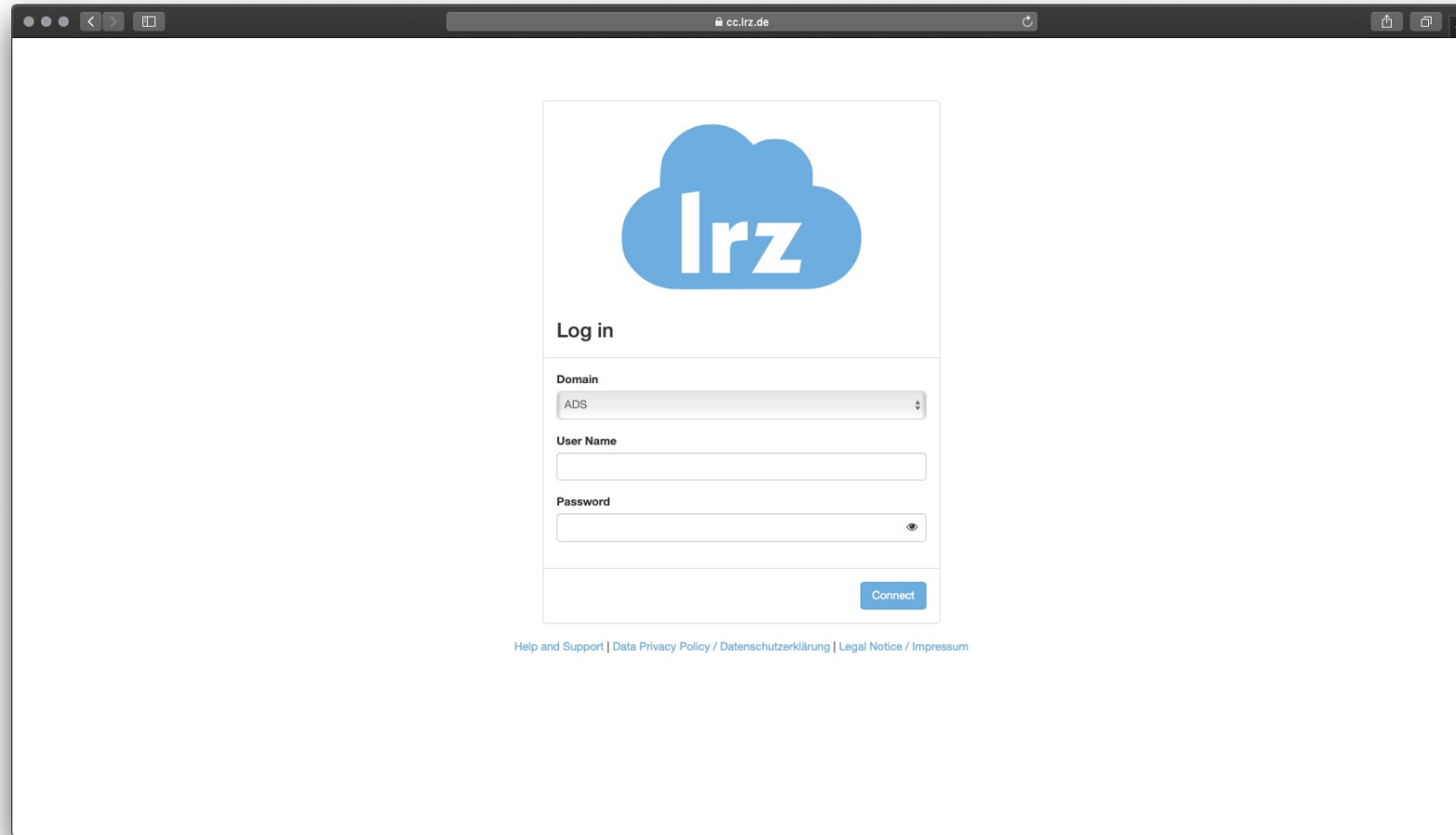
The LRZ Compute Cloud at a Glance



The Compute Cloud Web Interface



<https://cc.lrz.de>



The Compute Cloud Web Interface



Left panel.
Allows you to
operate with the CC.

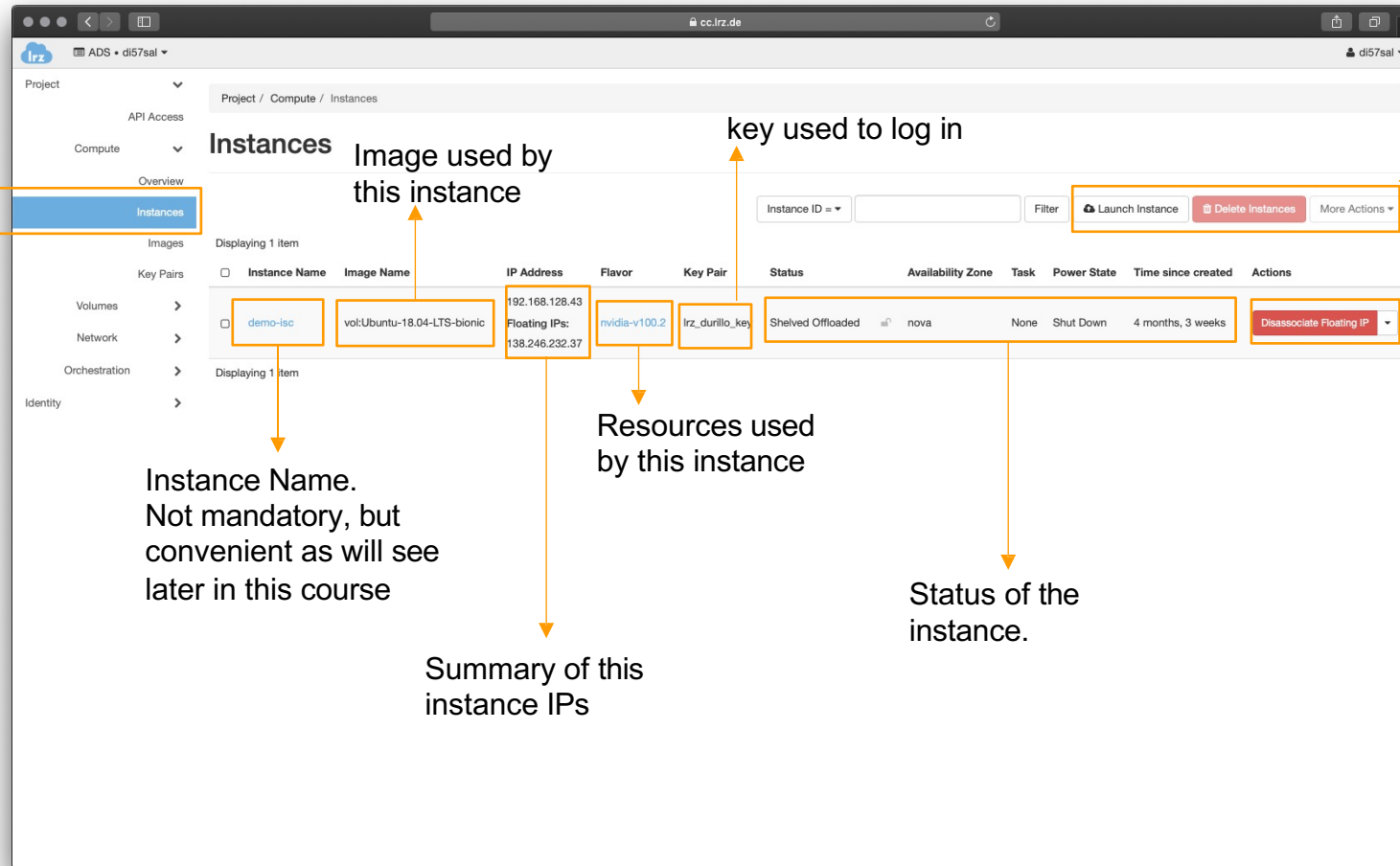
The screenshot shows the LRZ Compute Cloud web interface. The left sidebar contains a navigation menu with the following items: API Access, Compute (selected), Overview (highlighted), Instances, Images, Key Pairs, Volumes, Network, Orchestration, and Identity. The main content area is titled 'Project / Compute / Overview' and is divided into two sections. The 'Overview' section, titled 'Limit Summary', displays six resource usage metrics: Instances (Used 1 of 4), VCPUs (Used 40 of 160), RAM (Used 753,664 of 753,664, No Limit), Floating IPs (Allocated 6 of 50), Security Groups (Used 8 of 10), and Volumes (Used 5 of 10). The 'Usage Summary' section allows users to select a time period for usage queries (from 2019-09-29 to 2019-09-30) and provides a summary of usage statistics: Active Instances: 1, Active RAM: 736GB, This Period's VCPU-Hours: 1504.24, This Period's GB-Hours: 752.12, and This Period's RAM-Hours: 28342240.03. Below this is a table titled 'Usage' showing one instance named 'demo-isc' with 40 VCPUs, 20GB Disk, 736GB RAM, and a creation time of 4 months and 3 weeks. A 'Download CSV Summary' button is located at the bottom right of the Usage Summary section.

Resources Limits.
Shows you what
you are allow and
what you have in
use.

Summary.
Show the
instances you
have. They might
be in different
status.

The Compute Cloud Web Interface

Allows a more informative view of our instances



Instance Name. Not mandatory, but convenient as will see later in this course

Summary of this instance IPs

Resources used by this instance

Status of the instance.

Allows creating new instances / Deleting existing ones

Allows operating this instance (e.g., attach a volume, associate new floating IP)

The Compute Cloud Web Interface



Project / Compute / Key Pairs

Displaying 2 Items

Name	Fingerprint	Actions
access_from_windows	51:93:d4:5f:c1:8b:09:fd:da:11:19:a3:1e:34:ec:32	Delete Key Pair
lrz_durillo_key	c8:aa:48:d1:64:03:42:b5:bb:b9:0a:a4:4d:9f:41:a3	Delete Key Pair

Project / Volumes / Volumes

Displaying 8 Items

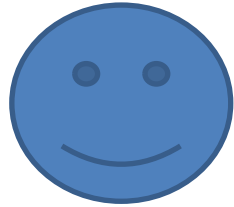
Name	Description	Size	Status	Type	Attached To	Availability Zone	Bootable	Encrypted	Actions
3896da63-2f67-4417-91bb-6a7d32d35cc8	-	30GiB	In-use	ceph	/dev/vda on test	nova	Yes	No	Edit Volume
0c5b5550-947b-44c5-a20a-46ba298a5d97	-	30GiB	Available	ceph	-	nova	Yes	No	Edit Volume
10a34d24-1d7b-46a4-bd3e-f1556a1d3918	-	30GiB	Available	ceph	-	nova	Yes	No	Edit Volume
tensorflow-gpu-volume	-	25GiB	Available	ceph	-	nova	Yes	No	Edit Volume
c706810c-dcd3-4f6a-9214-aadc8e6c801	-	20GiB	Available	ceph	-	nova	Yes	No	Edit Volume
63e9cb62-3bb5-44d3-be13-7421d12ff5a2	-	20GiB	Available	ceph	-	nova	Yes	No	Edit Volume
data	data for isc 2019 demo	80GiB	Reserved	ceph	-	nova	No	No	Update Metadata
f5bb6f63-cd0c-4c0f-a79a-b396ee8a72d0	-	20GiB	Reserved	ceph	-	nova	Yes	No	Update Metadata

Use-case: Generate Container Images for the LRZ AI Systems



HAPPY USER

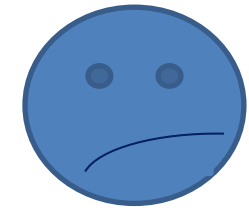
Dockerfile



```
FROM ubuntu:22.04
RUN apt install -y
python
RUN ...
ADD ...
```

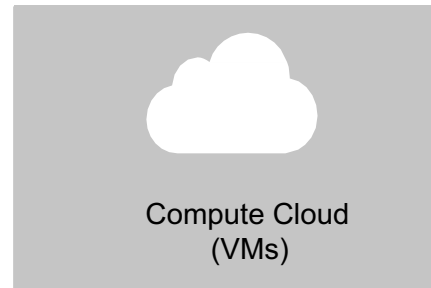
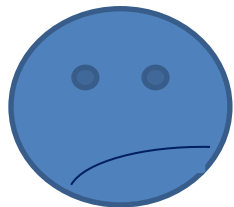


UNHAPPY USER

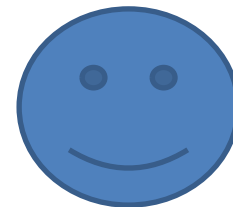


AI Systems (Slurm)

UNHAPPY USER



HAPPY USER

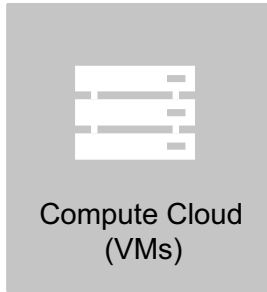


Use-case: Generate Container Images for the LRZ AI Systems



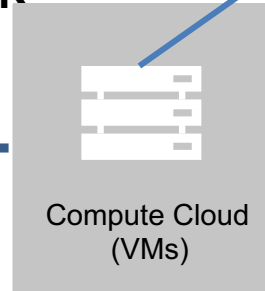
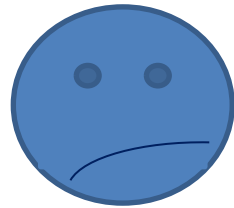
Step 1: Server on the CC

UNHAPPY USER



Step 2: Shared Storage CC and LAI

UNHAPPY USER



AI Systems (Slurm)

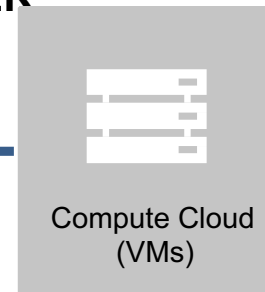


Step 4: Copy the enroot image on the shared storage



Step 3: Create the docker and enroot images in the CC using the dockerfile

UNHAPPY USER



Step 1 - Server on the CC - work on the terminal

- De-facto operation is no Graphical Interface on the provided images
- Accessing instances via ssh
 - No login based on password by default (public and private keys!)
- OpenStack must be aware of your public key(s) to add it(them) to newly created instances (otherwise you will not be able to login)
 - You can import a public key of a keypair generated using your method of preference
 - You can generate a keypair using OpenStack
 - the private key will be downloaded to your computer
 - the public will be recorded by OpenStack

Step 1 - Server on the CC - creating a KeyPair

- Default login onto servers is based on public-private key infrastructure
- Two options
 - To create a new pair of public and private keys ← Used in this course
 - To upload the public key part of existing pair

Live demo



image: Freepik.com

Step 1 - Server on the CC - creating a KeyPair

Too fast? Check a video example here



<https://tinyurl.com/kxhxpda>

*For this hands-on only the creation of the key is relevant (you can ignore connecting to the server for now.)

Step 1 - Server on the CC - creating the server

- Create a server running Ubuntu 22.04

Live demo



image: Freepik.com

Step 1 - Server on the CC - creating the Server

Too fast? Check a video example here



<https://tinyurl.com/yckp8ym4>

Step 1 - Server on the CC - accessing the server

- Default IPs are on the private OpenStack network
- Floating IPs allow bridging external networks to the private OpenStack network

Live demo



image: Freepik.com

Step 1 - Server on the CC - creating and assigning a floating ip

Too fast? Check a video example here



<https://tinyurl.com/bdfha7sz>

Step 1 - Server on the CC - creating and assigning security groups

- By default all ports are closed by the firewall of OpenStack
- Security groups is the mechanism provided for opening these firewalls
- This demo shows how to open the port 22, required by ssh

Live demo



image: Freepik.com

Step 1 - Server on the CC - creating and assigning security groups

Too fast? Check a video example here



<https://tinyurl.com/2s3f6vny>

Step 1 - Server on the CC - connecting to the created server

- This demo shows how to connect via ssh to the server created before

Live demo



image: Freepik.com

Step 1 - Server on the CC - connecting to the created server

Too fast? Check a video example here

Linux and MacOS



<https://tinyurl.com/kxhxpda>

Windows using PuTTY



<https://tinyurl.com/mt9cxzpn>

*For windows users connecting via the Windows Subsystem for Linux (WSL) (e.g., Ubuntu app in windows):

You need to copy the downloaded file from Windows into the WSL space. If your file has been downloaded into C:/Downloads/pair.pem from the Ubuntu terminal copy the file with:
\$ cp /mnt/c/Downloads/pair.pem ./key.pem

After that follow the Linux and MacOS video for connecting, assuming key.pem is your key (i.e., change the permissions and use this file in the ssh command).

Step 1 - Server on the CC - summary of steps

- On the Compute Cloud Web Interface
 - Generate a new keypair (see videos before)
- On your computer (see video links the slide before)
 - A file with the extension .pem will be downloaded to your machine (the private key) from previous step
 - **In Linux/UNIX:** change the permission of that file to 600 (\$ chmod 600 ...)
 - **In Windows with WLS:** copy the downloaded file to inside the WSL (/mnt/c/ allows you accessing C:\ in windows from WSL,)
 - **Once copied, change the permissions as in the Linux/UNIX case**
 - **In Windows with Putty:**
 - import it using PuttyGen and configure Putty for using the imported key

Step 1 - Server on the CC - summary of steps

- On the Compute Cloud Web Interface

- Create an instance (next steps are documented with screen shots in successive slides)
 - Choose Ubuntu as image
 - CPU only flavor (preferably a small one)
 - Should be accessible from Internet
 - Place the instance on the private network called internet
 - Once the instance is created assign it a floating IP from the Internet pool

Step 1 - Server on the CC - summary of steps

Subtask: Access the created instance via SSH

- On the Computer Cloud Web Interface
 - **Remember you should have created a security group that allow ingress connections to port 22! and added it to the instance**
- On your computer
 - Open a terminal application

```
ssh -i <path_to_the_pem_file> ubuntu@<floating-ip>
```

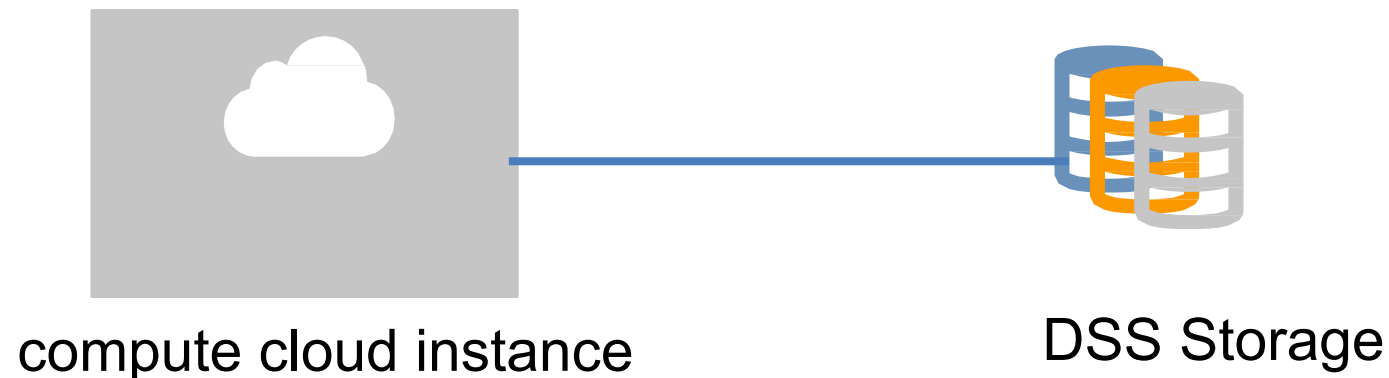
After this step, the rest of slides assume everyone is connected via ssh to the created instance

Step 2: Shared Storage CC and LAI

This use case shows the integration among different LRZ Services

Examples where such integration would be needed

1. Pre- or Postprocessing of data to use/used in SMUC-NG, Linux Cluster or LRZ AI Systems
2. Downloading a large dataset to be consumed on another LRZ service
3. Others (like our example: use that storage to shared an enroot image with the LRZ AI System)



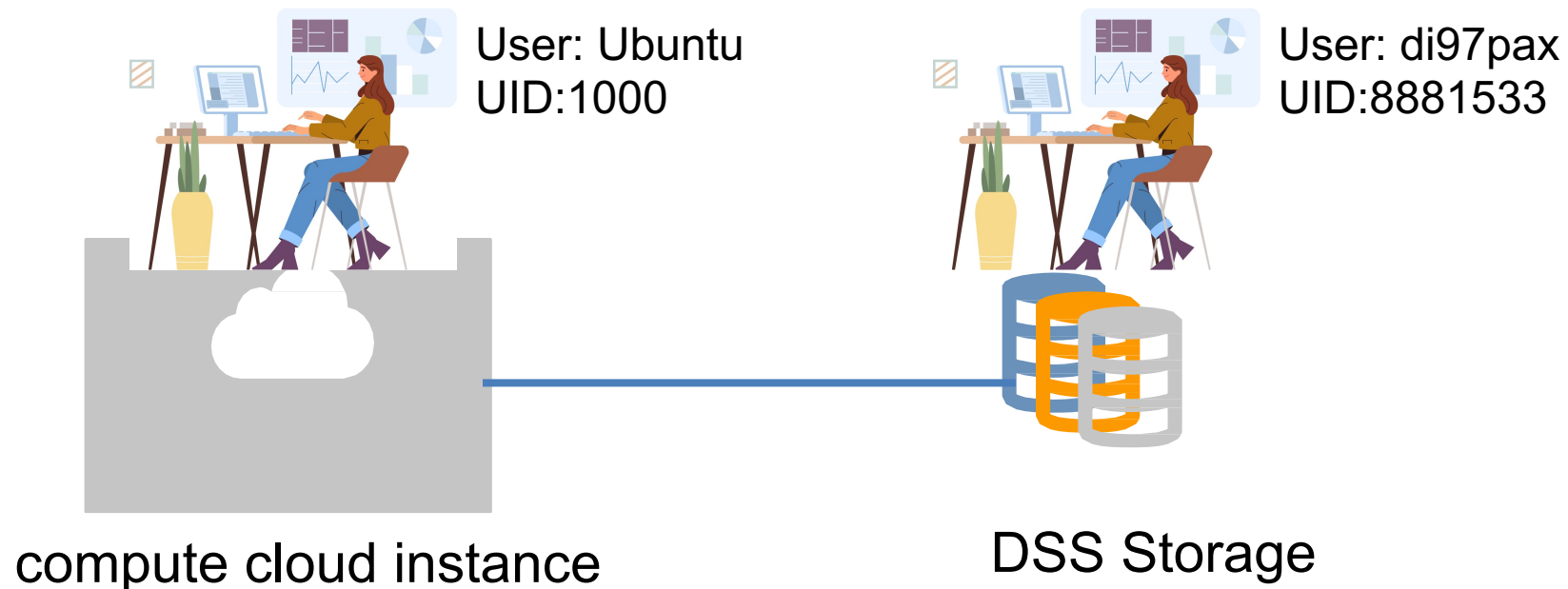
Step 2: Shared Storage CC and LAI

Different users

On a VM: a user within the OS running in that VM

On DSS: LRZ user

Access to DSS containers and data defined on UID and GID basis



Step 2: Shared Storage CC and LAI

1st. create in the VM a Unix group with the group of the container (usually the container name)

```
$ sudo groupadd --gid <GID> <container-name>
```

2nd. create in the VM a user with the same UID that your LRZ account and assign it to the created group in 1st

```
$ sudo adduser --home /home/myuser --uid <UID> --ingroup <container-name> myuser
```

3rd. install the nfs-common package (to mount nfs shares)

```
$ sudo apt install nfs-common
```

4th. create a folder where the DSS container will be mounted

```
$ sudo mkdir -p <container-name>
```

Steps to do in your

Step 2: Shared Storage CC and LAI

5th. create a new NFS export of your DSS container
(only *data curators* can perform this step)

NFS Exports ?

IP	Access Type	Status	Mount Path	Expires at	Config	Actions
	RW	●				✎ 🗑️

[+ Add new export](#)

Create new NFS Export for DSS Container PR74QO-DSS-0014 ?

Client IP Address *

Access Type * RW ▼

Expiration Date --- --- --- ▼

[Create](#)

Steps to do via DSS
or dsscli

Step 2: Shared Storage CC and LAI

6th. mount the container

```
$ sudo mount -t nfs -o rsize=1048576,wsize=1048576,hard,tcp,bg,timeo=600,vers=3 <mount-path>  
/dss/<container-name>
```

7th. access the data on your DSS container

```
$ sudo su - myuser  
$ cd /dss/<contianer-name>
```

To be obtained via the DSSWeb or dsscli Steps to do in your

Step 2: Shared Storage CC and LAI

- This demo shows how to mount a DSS container

Live demo



image: Freepik.com

Step 3: Create the docker and enroot images in the CC using the dockerfile: installing the tools

1st. Install Docker on the Server

```
$ sudo apt install docker.io  
$ sudo usermod -aG docker ubuntu
```

Exit and login again for this step to be effective

Alternative way of installing docker:

```
curl -fsSL https://get.docker.com/  
| bash
```

2nd. Install enroot

```
$ arch=$(dpkg --print-architecture)  
$ curl -fSsL -O https://github.com/NVIDIA/enroot/releases/download/v3.4.1/enroot\_3.4.1-1\_\${arch}.deb  
$ curl -fSsL -O https://github.com/NVIDIA/enroot/releases/download/v3.4.1/enroot+caps\_3.4.1-1\_\${arch}.deb  
$ sudo apt install -y ./*.deb
```

Step 3: Create the docker and enroot images in the CC using the dockerfile: exporting the container

3th. Create the Docker image

```
$ docker build . -t test
```

from ubuntu:23.04
 RUN apt update
 RUN apt install -y python3

4th. Check that the image is there

```
$ docker image list
REPOSITORY TAG IMAGE ID CREATED SIZE
test latest 0bce02617813 6 minutes ago 158MB
ubuntu 23.04 639282825872 7 weeks ago 70.3MB
```

You use **dockerd** as want to import the image from the local docker daemon

5th. Export the image to an enroot one

```
$ enroot import dockerd://test
$ ls
test.sqsh Dockerfile
```

You can copy now test.sqsh to the mounted DSS and it will be available to use in the LRZ AI System



Step 3: Create the docker and enroot images in the CC using the dockerfile

Live demo



image: Freepik.com

Other ways of interacting with the LRZ Compute Cloud.

The OpenStack Client Tools

- Command-line client for OpenStack that brings the command set for Compute, Identity, Image, Object Storage and Block Storage APIs together in a single shell with a uniform command structure
- <https://docs.openstack.org/python-openstackclient>
- There are different ways of installing the tools. Some examples are:
 - In Ubuntu using apt as front end package manager

```
$ sudo apt install python3-openstackclient
```

- Using python pip

```
$ pip3 install openstackclient
```

Hands-on: configuring the openstack client tools

- A configuration file specific to your account can be downloaded for configuring your system to access the compute cloud via the command line

Live demo



image: Freepik.com

Hands-on: configuring the openstack client tools

Too fast? Check a video example here



<https://tinyurl.com/bdz42kaa>

The OpenStack Client Tools

- Execute the downloaded file
 - e.g. in my case

```
$ source /Download/di57sal-openrc.sh
```

The OpenStack Client Tools - Listing

```
di57sal — di57sal@BADWLRZ-CM60333 — ~ — -zsh — 139x24
[→] openstack server list
+-----+-----+-----+-----+-----+-----+
| ID | Name | Status | Networks | Image | Flavor |
+-----+-----+-----+-----+-----+-----+
| 361246fc-ab91-4520-8c27-cf166cf7a50a | test2 | ACTIVE | internet=192.168.128.148 | | lrz.small |
| d7c17602-74d9-41b7-8e19-563b9f283b33 | test | ACTIVE | internet=192.168.128.96, 138.246.232.52 | | lrz.small |
| 6e23850f-9019-47af-b252-7e8897fae4a8 | demo-isc | SHELVED_OFFLOADED | internet=192.168.128.43, 138.246.232.37 | | nvidia-v100.2 |
+-----+-----+-----+-----+-----+-----+
[→] █
```

The OpenStack Client Tools - Listing

```

di57sal — di57sal@BADWLRZ-CM60333 — ~ — zsh — 139x34
[→] ~ openstack flavor list
+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | Name | RAM | Disk | Ephemeral | VCPUs | Is Public |
+-----+-----+-----+-----+-----+-----+-----+
| 5eaba3f9-2a26-4371-89f7-d1863702d6e3 | nvidia-v100.2 | 753664 | 20 | 0 | 40 | False |
| 6186e4c3-3f02-4ecf-bf68-2088ad10d11b | lrz.medium | 9216 | 20 | 0 | 2 | False |
| 690000bb-457a-479a-9f09-aa32f467b499 | tiny | 512 | 1 | 0 | 1 | False |
| 736b1189-1daf-46f1-ac2c-a9661f6f2b29 | lrz.large | 18432 | 20 | 0 | 4 | False |
| a7f8aa12-48a9-4abe-af82-642c381e74f0 | lrz.small | 4608 | 20 | 0 | 1 | False |
| e6cb5fc6-f0df-4970-ac8a-90d67f401808 | nvidia-v100.1 | 376832 | 20 | 0 | 20 | False |
| ff616544-723d-4eaf-81a5-1df1e86c4c3 | lrz.xlarge | 46080 | 20 | 0 | 10 | False |
+-----+-----+-----+-----+-----+-----+-----+
[→] ~ openstack image list
+-----+-----+-----+
| ID | Name | Status |
+-----+-----+-----+
| 5235e1d1-b2a2-46ed-abfd-c0d8b50c71ed | CentOS-6 | active |
| 63825d44-5312-4ace-b1a5-de7791a6bb43 | CentOS-7 | active |
| 24d28aa3-06d8-4f15-b211-a2428dba0112 | Debian-10-buster | active |
| 112a98db-e6e9-453c-840a-befbbf81a414 | Debian-8-jessie | active |
| c06b03df-4812-4324-8d95-8e47320acf8b | Debian-9-stretch | active |
| f469764e-822b-4177-92a3-83e2a421ae8e | Fedora-29 | active |
| 103cee77-a5b6-4ee6-aa59-0b11b8177f73 | Fedora-30 | active |
| eec6d450-b6b4-4560-8e3d-4945d819361b | FreeBSD-11.2 | active |
| 37ebe015-f481-4e65-8307-deca96359b42 | FreeBSD-12.0 | active |
| 956a7c2d-8d30-441e-891b-14081acac6fd | GPU-Ubuntu-18.04-LTS-bionic-(cuda/docker) | active |
| 0c231b3c-6445-4f66-aa2d-0e88f35c8338 | Ubuntu-16.04-LTS-xenial | active |
| 7e7e699a-dfab-4e10-ac23-696dee869580 | Ubuntu-18.04-LTS-bionic | active |
| 0ef79e2f-cea9-4418-97c7-6f05045bd38a | Ubuntu-19.04-disco | active |
| 41f4ed09-97f0-4aa1-9c14-f8597810e411 | cirros | active |
| 61109f36-0b03-4c14-8a8e-8934f74e85b8 | remote-visualization-ubuntu-18.04 | active |
+-----+-----+-----+

```


The OpenStack Client Tools - Listing

```
di57sal — di57sal@BADWLRZ-CM60333 — ~ — -zsh — 139x24
[→] ~ openstack network list
openst+-----+-----+-----+
| ID | Name | Subnets |
+-----+-----+-----+
| 2da955ac-0ba6-4755-918d-9ae23565492c | test | 4abf2933-4e90-4280-a7f9-8cf63d71d05f |
| 3f1c6c34-2be9-44b3-9f21-c3e031ab8e5c | MWN | 16677895-8403-4f14-866b-62256404f0aa |
| 8f5b0e5e-e3bf-4b53-b680-30bc593213eb | internet | ef5b863e-3d4a-4947-95cf-83b311208894 |
| a3e4d020-c8b4-48b5-beb1-5f0d47d06ed7 | MWN_pool | 3e274178-88b9-4cde-8c37-04bb3a2b0911 |
| cca21b1f-03cd-410b-a80f-5cfce18afeec | internet_pool | 9851df97-49c5-4cb9-a385-6bf6fbfc46e9 |
+-----+-----+-----+
^R
[→] ~ openstack keypair list
+-----+-----+
| Name | Fingerprint |
+-----+-----+
| access_from_windows | 51:93:d4:5f:c1:6b:09:fd:da:11:19:e3:1e:34:ec:32 |
| lrz_durillo_key | c8:ea:48:d1:64:03:42:b5:bb:b9:0a:a4:4d:9f:41:e3 |
| test_machine_key | 66:68:b0:f3:a5:2b:1d:82:d3:25:b6:68:5d:76:03:70 |
+-----+-----+
[→] ~
```

The OpenStack Client Tools - Creating

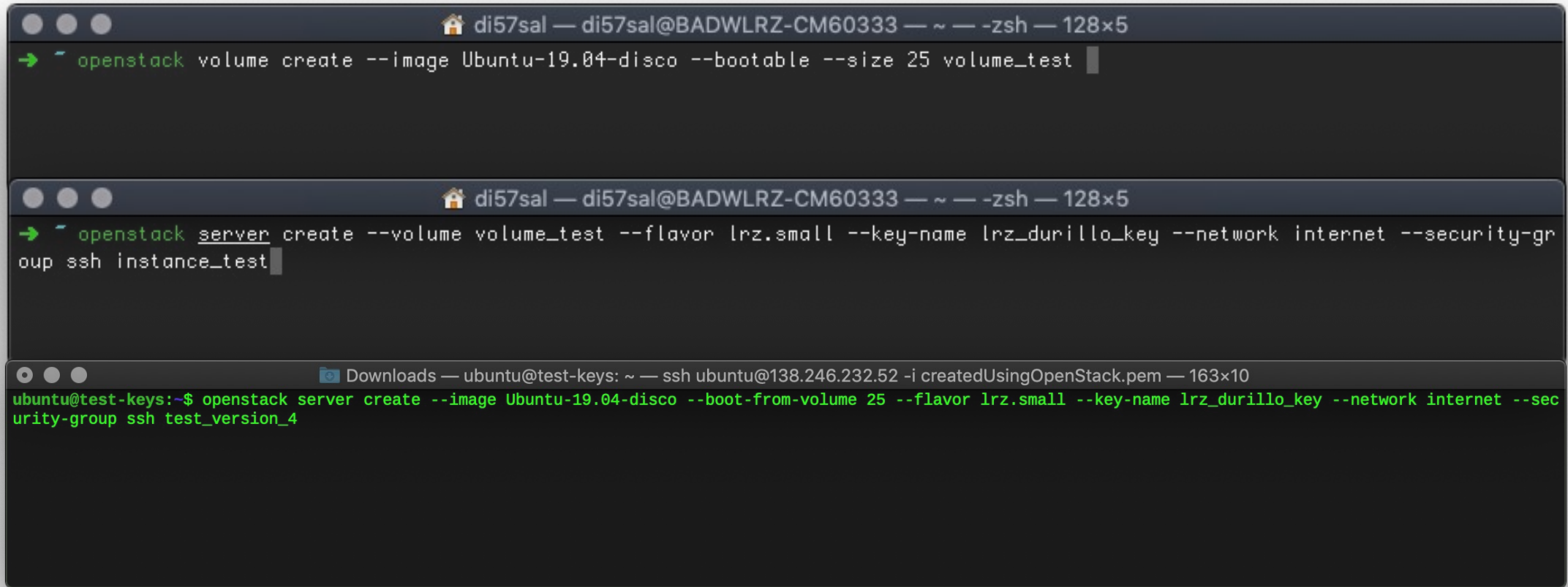


```
di57sal — di57sal@BADWLRZ-CM60333 — ~ — -zsh — 128x9  
→ openstack server create
```

- Simple unified API for all OpenStack Entities
- Access to the help of each command
 - Simply executing it with no additional arguments
 - Executing it with `-h` | `-- help` option

The OpenStack Client Tools – Creating a new server

- Depending on the version of the tools



The image shows three terminal windows stacked vertically, illustrating the evolution of OpenStack client commands. The top window shows a command to create a volume. The middle window shows a command to create a server using a volume. The bottom window shows a command to create a server using a volume and booting from it. Blue brackets on the right side group these windows into three categories: Version 3.19.0 (top two windows), Version > 4.0.0 (middle window), and Version 6.0.0 (bottom window).

```
di57sal — di57sal@BADWLRZ-CM60333 — ~ — -zsh — 128x5  
→ openstack volume create --image Ubuntu-19.04-disco --bootable --size 25 volume_test
```

```
di57sal — di57sal@BADWLRZ-CM60333 — ~ — -zsh — 128x5  
→ openstack server create --volume volume_test --flavor lrz.small --key-name lrz_durillo_key --network internet --security-group ssh instance_test
```

```
Downloads — ubuntu@test-keys: ~ — ssh ubuntu@138.246.232.52 -i createdUsingOpenStack.pem — 163x10  
ubuntu@test-keys:~$ openstack server create --image Ubuntu-19.04-disco --boot-from-volume 25 --flavor lrz.small --key-name lrz_durillo_key --network internet --security-group ssh test_version_4
```

Version 3.19.0

Version > 4.0.0

Version 6.0.0

```
openstack server create --image Ubuntu-22.04-jammy --boot-from-volume 25 --flavor lrz.small --key-name \ yourkeyname --network MWN --security-group your_security_group_name instance_name
```

Hands on Session 2

- Recreate the instance of Hands-on session before using the Openstack Command Line tools

Live demo



image: Freepik.com

Too fast? Check a video example here



<https://tinyurl.com/6bcepb5>

Hands-on session—Secure your AI Workflow with LRZ middleware

Purpose: your workflow includes sensitive info

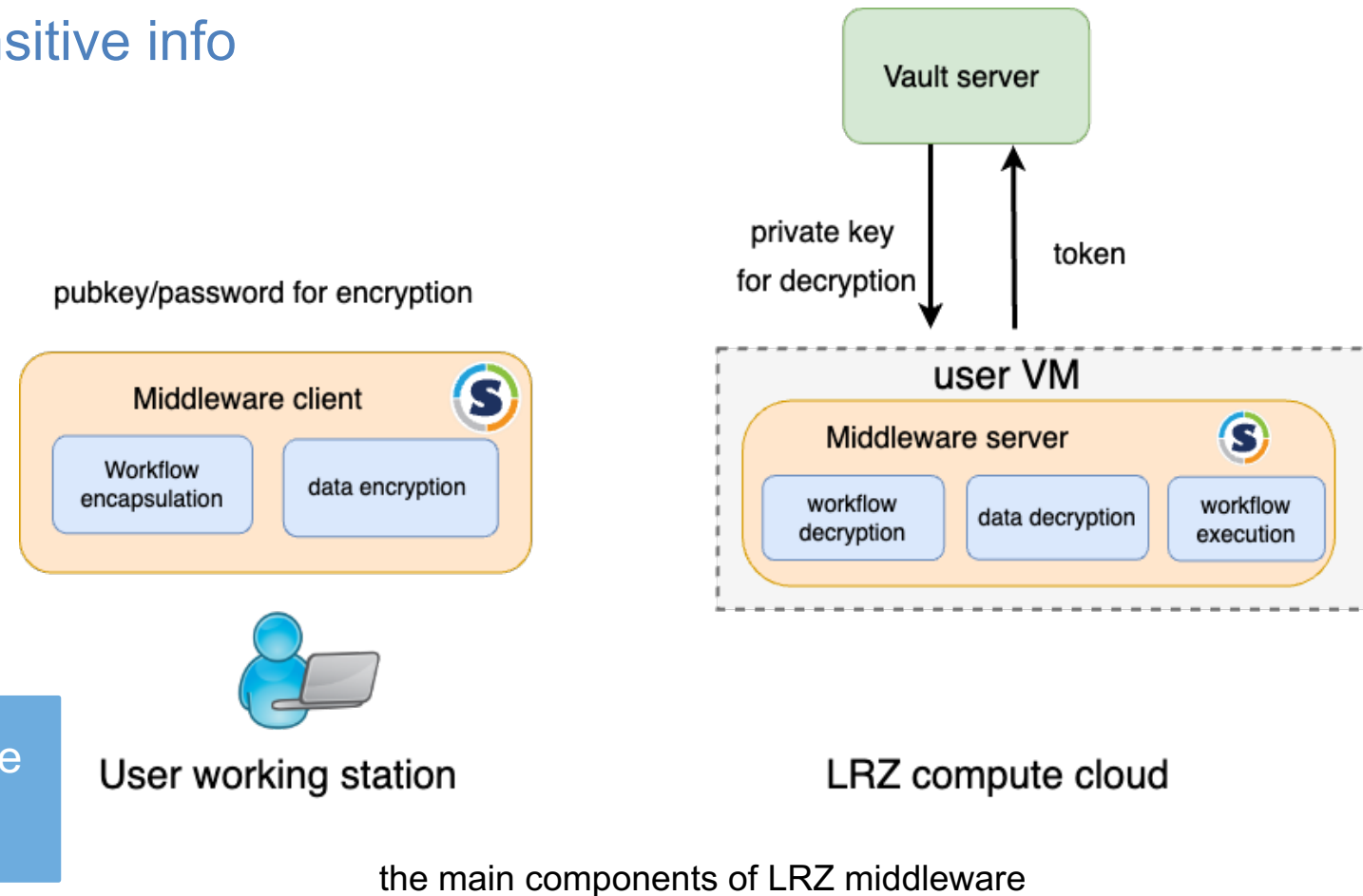
What does this workflow do?

1. A KNN model
2. Read patient health data from a csv file
3. Display data property and plot a correlation map

Hands-on session—Secure your AI Workflow with LRZ middleware*

Purpose: your workflow includes sensitive info

- Pack your code inside encrypted **workflow container**
- Pack your data (if large size, e.g. 100GB) into an encrypted **data container**



This hands-on session uses both Middleware client and Middleware server on VM

* N. Zhou, F. Dufour, V. Bode, P. Zinterhof, N. J. Hammer, and D. Kranzlmler, "Towards Confidential Computing: A Secure Cloud Architecture for Big Data Analytics and AI," in *IEEE International Conference on Cloud Computing (IEEE CLOUD)*, (Chicago, Illinois, USAI), 2023.
Introduction to the LRZ Compute Cloud | 01.12.23 | PD. Dr. Juan J. Durillo, Dr. Naweiluo Zhou

Hands-on session—Secure your Workflow with LRZ middleware

Steps:

1. Prepare the customized image for uploading to Openstack: e.g. *ibguestfs*, *packer*
2. Upload an customized VM image to Openstack (an image has been uploaded for you)
3. Create an VM with the uploaded image
4. Create an encrypted data container with password
5. Create an encrypted workflow container with a given pubkey
6. Execute the encrypted workflow with a given token

A token is used to fetch a private key stored safely on the Vault server. A private key is used to decrypt our workflow container. The token is NOT the decryption key for the workflow container

Hands-on session—Secure your Workflow with LRZ middleware

Steps:

1. Prepare the customized image for uploading to Openstack:

```
$ qemu-img convert -f qcow2 -O raw jammy-server-cloudimg-amd64.img aicourse.raw (optional, on your computer)
```

2. Upload an customized VM image to Openstack (on your computer)

```
$ openstack image create --container-format bare --disk-format raw --file aicourse.raw AI-course (optional)
```

```
$ openstack image set --accept [image_id] (cml for today, replace image_id with the id in the course)
```

3. Create an VM with the uploaded image (on your computer)

```
$ openstack server create --image AI-course --flavor lrz.xlarge --network MWN --security-group ai_course --key-name aicourse --boot-from-volume 20 aicourse
```

4. create an encrypted data container with a password

```
$ cd ~/middleware ; $ lrzclient encrypt -data mycontainer.data
```

5. create an encrypted workflow container with a given pubkey (located at lrz-middleware/rsa_pub.pem)

```
$ lrzclient encrypt -workflow myworkflow.sif
```

6. execute the encrypted workflow with a given token (located at lrz-middleware/token)

```
$ lrzserver run -workflow myworkflow.sif -data mycontainer.data
```

Hands-on session—Secure your Workflow with LRZ middleware

Other cmls to play with

1. Decrypt your data container:

```
$ lrzserver unencrypt -data mycontainer.data (optional)
```

2. Seal your decrypted data container

```
$ lrzserver seal (optional)
```

Wrap-up

- Motivation of Cloud Computing?
- Introduction to Cloud Computing and OpenStack
- Hands-on session using the web interface
- Hands-on session using the OpenStack Client Tools
- Hands-on using customised VM image
- Hands-on to secure AI workflow with LRZ middleware

Further reading:

1. Thomas, E., Zaigham, M. and Ricardo, P., 2013. *Cloud Computing Concepts, Technology & Architecture*. Prentice Hall.
2. Sefraoui, O., Aissaoui, M. and Eleuldj, M., 2012. OpenStack: toward an open-source solution for cloud computing. *International Journal of Computer Applications*, 55(3), pp.38-42.
3. Netto, M.A., Calheiros, R.N., Rodrigues, E.R., Cunha, R.L. and Buyya, R., 2018. HPC cloud for scientific and business applications: taxonomy, vision, and research challenges. *ACM Computing Surveys (CSUR)*, 51(1), pp.1-29.

Please visit
[url-for-dec23](#)
and rate this course.

Your feedback is highly appreciated!
Thank you!

