

Introduction to using cloud and containers for training - OpenStack and Docker oriented view

6.2.2019

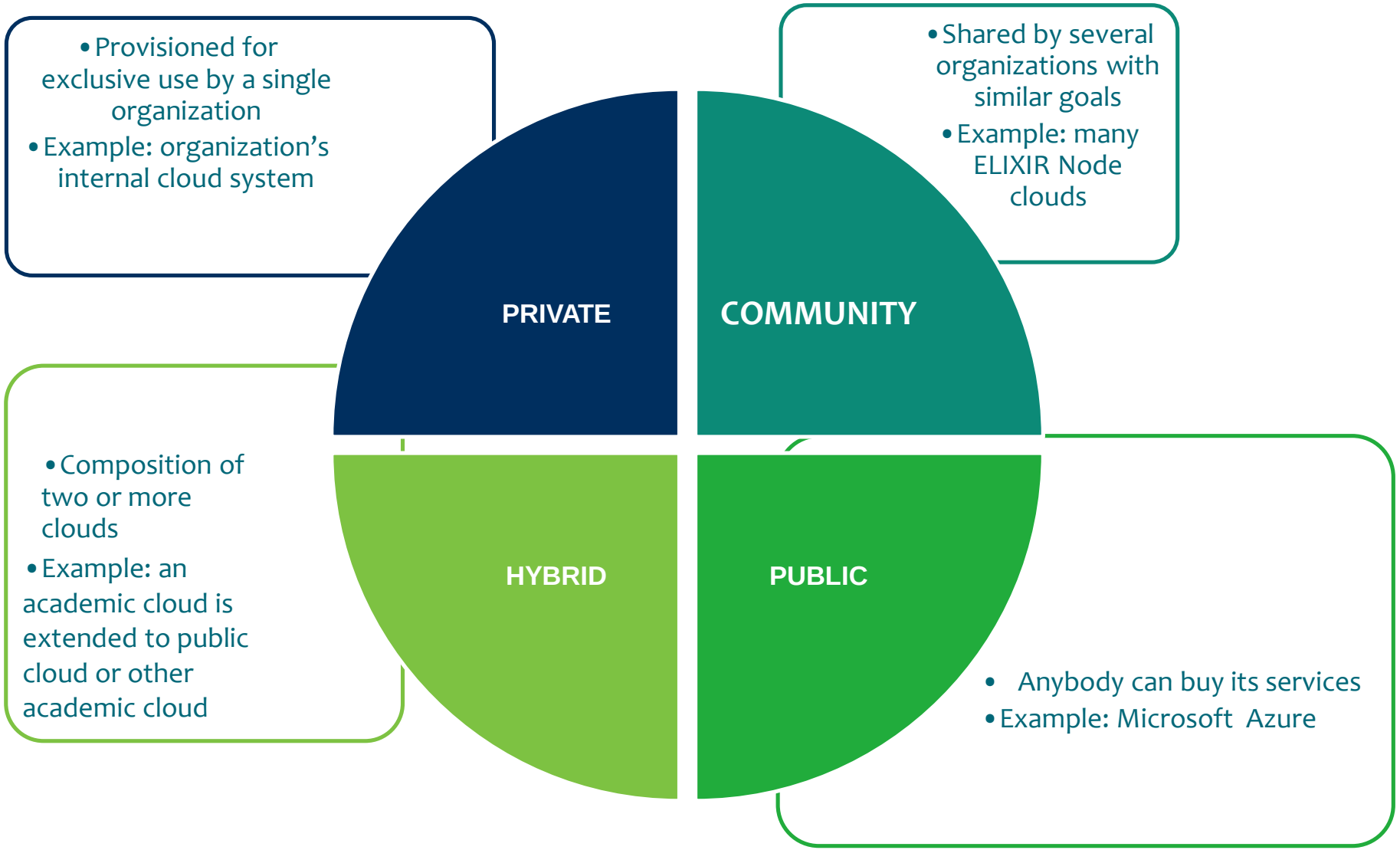
CSC – Finnish research, education, culture and public administration ICT knowledge center

Cloud Computing

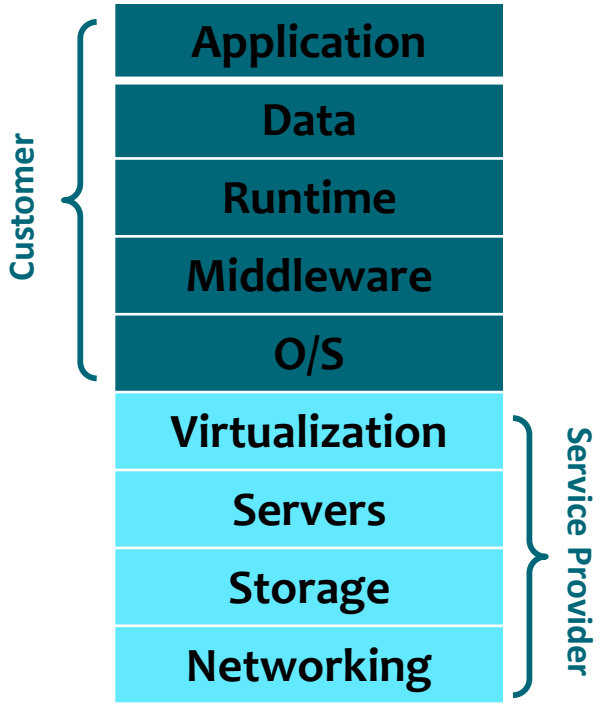
"Cloud Computing refers to on-demand delivery of computing services – servers, storage, databases, networking, software, analytics and more— over the network."

"A model for enabling 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"

Cloud Deployment Models

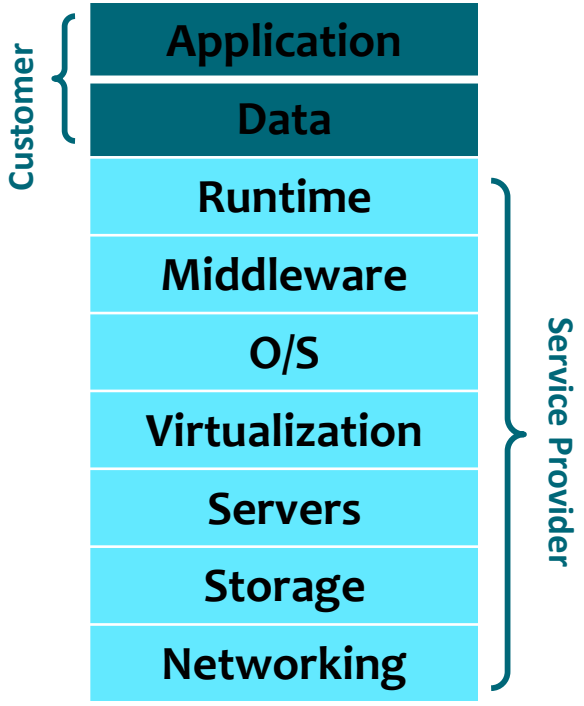


IaaS – PaaS – SaaS : the responsibility division



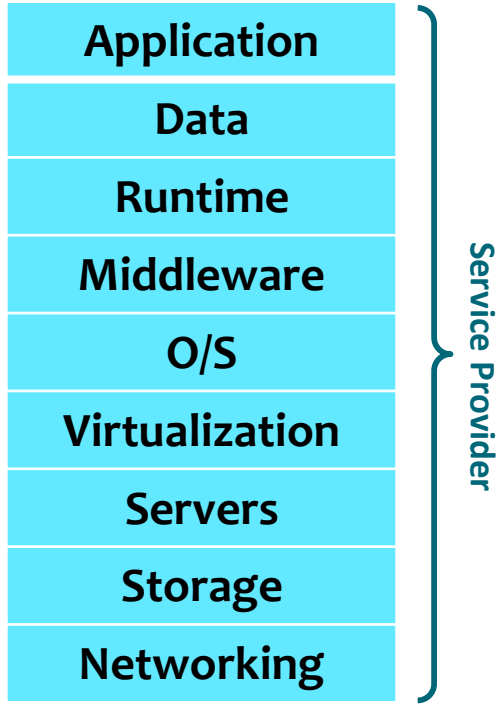
Infrastructure as a Service (IaaS)

CSC's ePouta/cPouta
Amazon EC2,
Microsoft Azure….



Platform as a Service (PaaS)

CSC's RAHTI
CSC's notebook.csc.fi
Google AppEngine,
Heroku,…

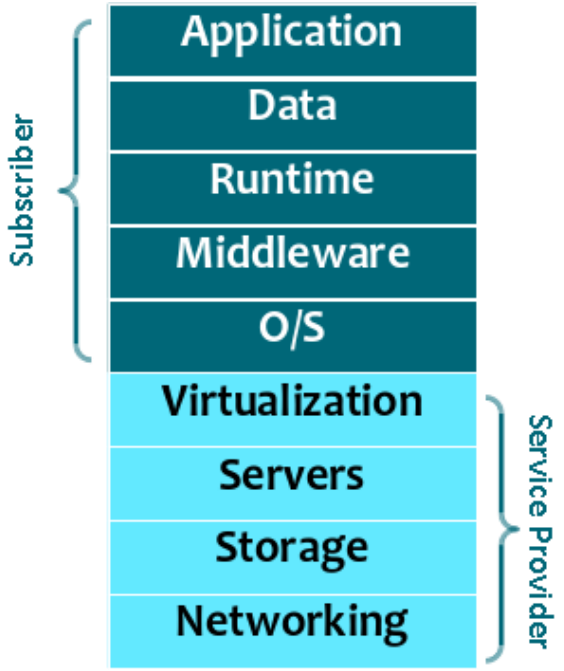


Software as a Service (SaaS)

CSC's Chipster,
Google Web Apps,
Microsoft Web Apps,,..
DropBox and many other cloud services

Cloud Service Landscape

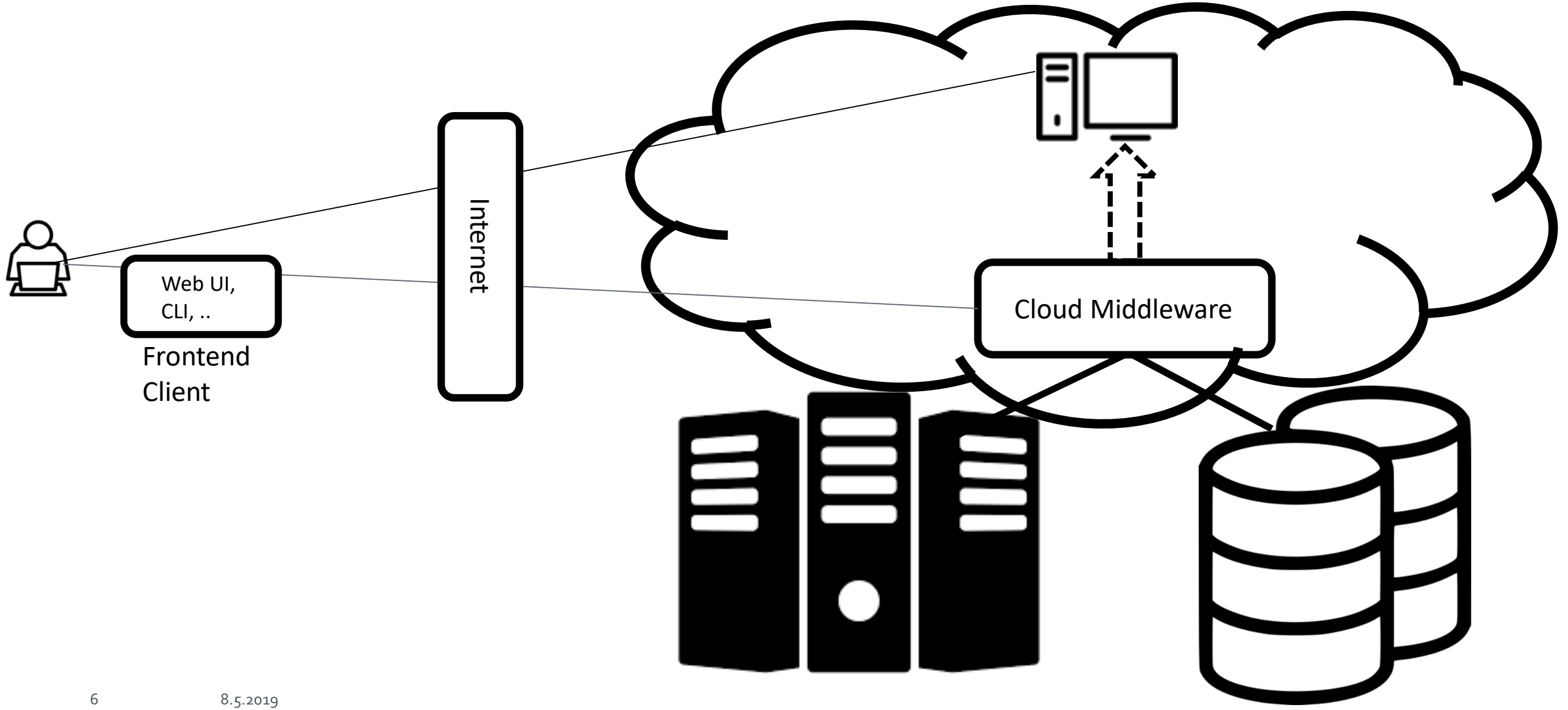
Infrastructure as a Service (IaaS)



For example:
pouta.csc.fi, Amazon EC2, Microsoft Azure, Google Compute Engine

- This course is strictly about IaaS cloud
- Yesterday's services in a Virtual Machine are today's containerized microservices; today's containerized microservices are tomorrow's serverless architectures
- IaaS probably has some evolutionary cycles left, too
- Let's proceed to look at how IaaS is setup!

Typical IaaS Cloud Setup



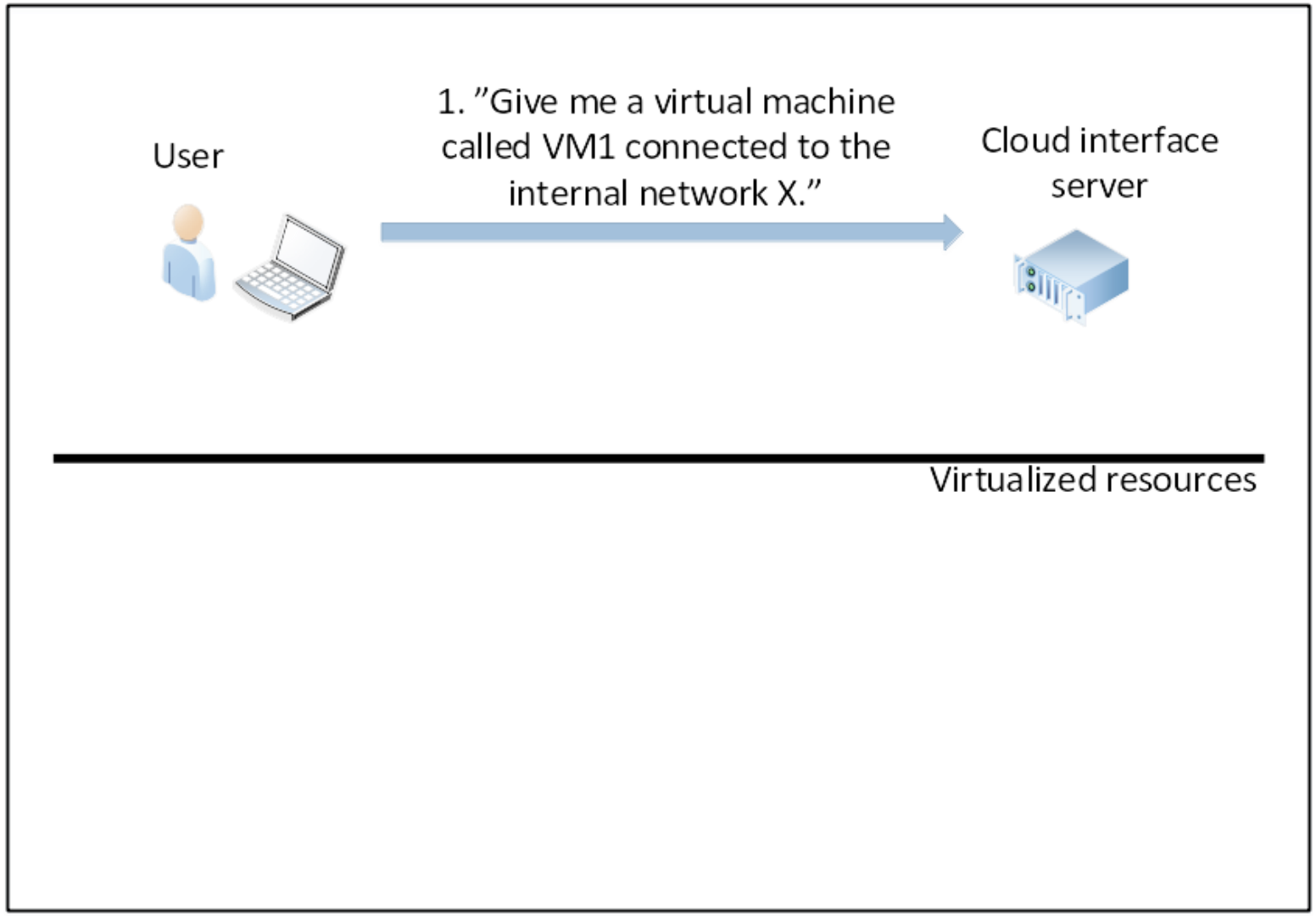
Creating virtual resources in Pouta - User Interfaces

- Web User Interface -
  - Suitable for administering individual VMs, keys, images, volumes…
  - The only UI to support Haka federated login

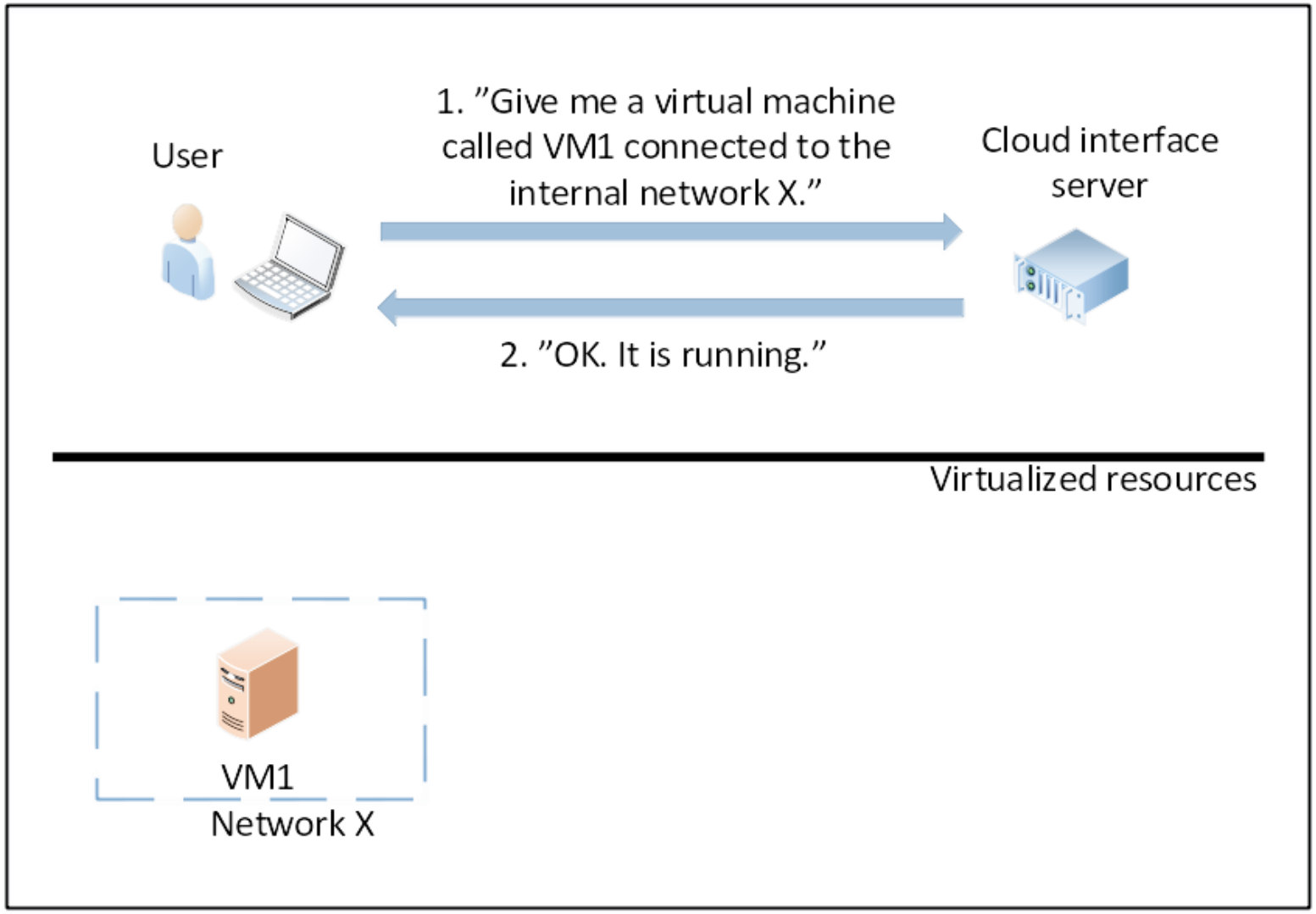
- CLI tools
  - Suitable for more elaborate resource provisioning and possibly some lightweight (scripted) software integrations
  - More info at ***https://research.csc.f/pouta-install-client***

- Programming APIs
  - Suitable for building very large systems and stacks
  - Support from individual services (compute, storage) to full-fledged orchestration
  - List of APIs available at ***https://pouta.csc.f/dashboard/project/access_and_security***

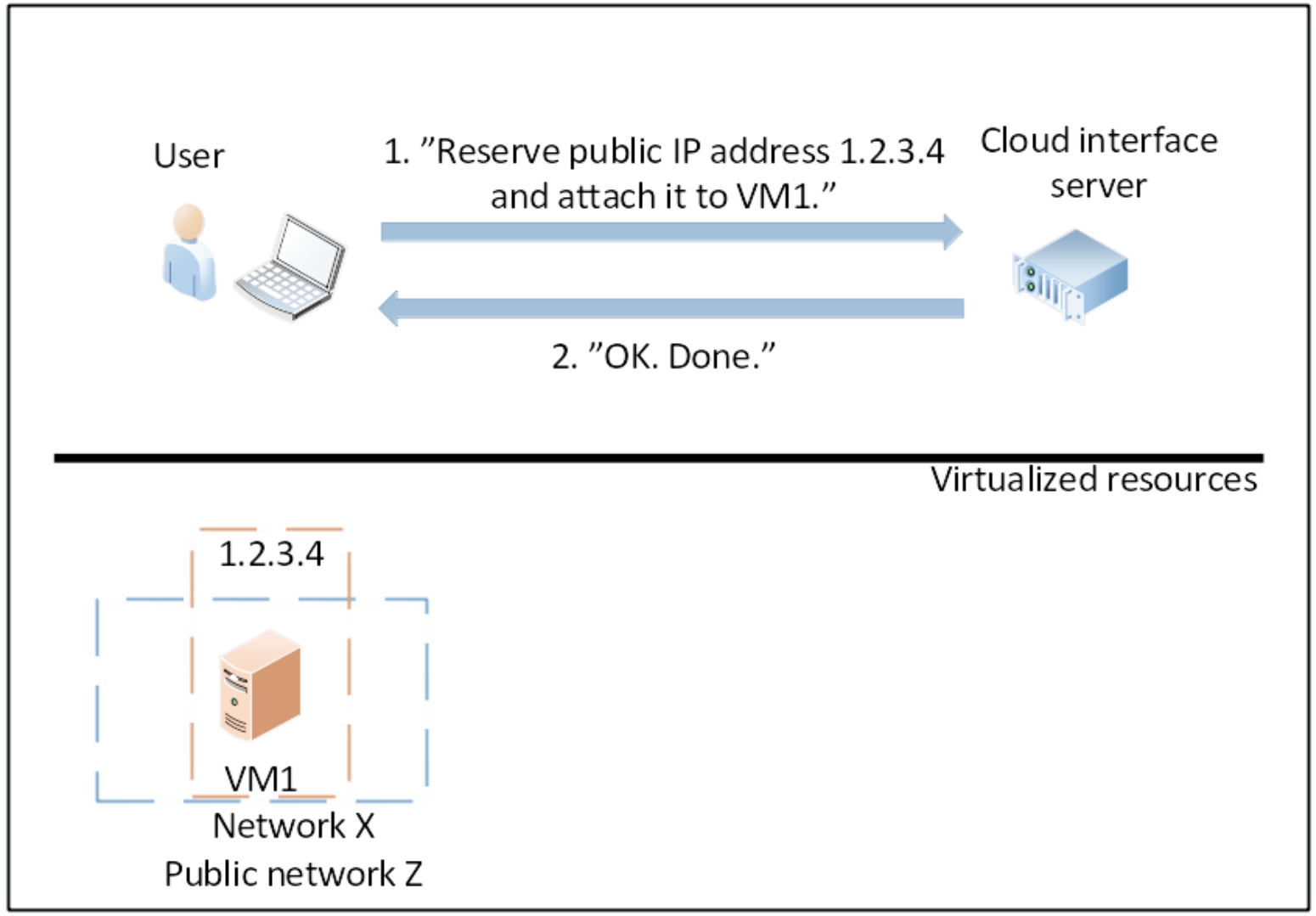
Workflow For Creating Resources



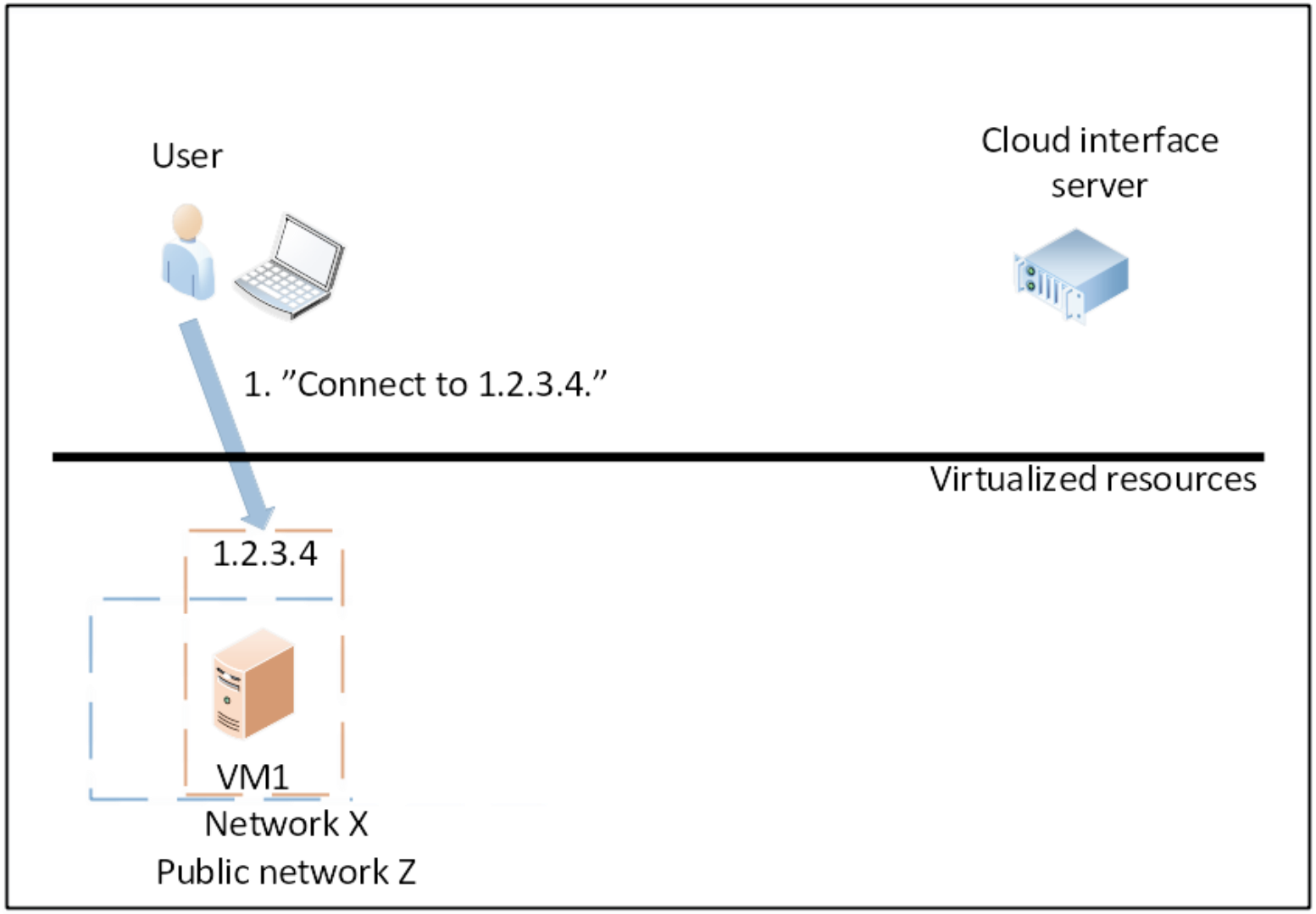
Workflow For Creating Resources



Workflow For Creating Resources



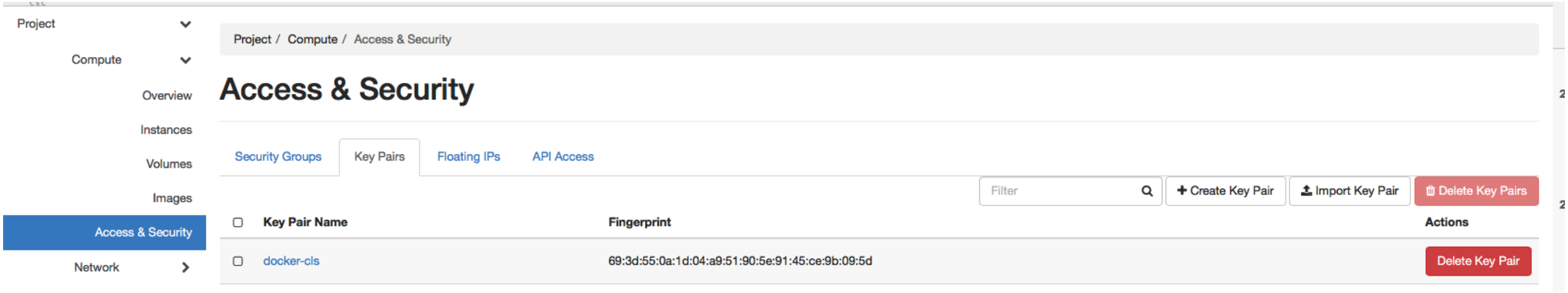
Workflow For Creating Resources



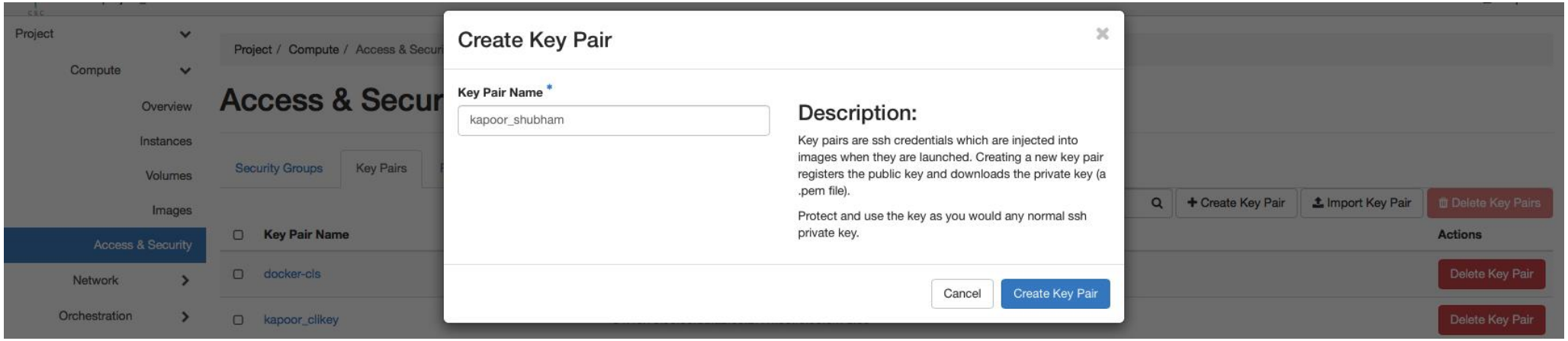
Things needed to create and access a VM in cPouta

- Access to Pouta Web UI
- One IPv4 address - a public "**Floating IP**"
- **Security Group** permitting access from User's computer
- Ssh **key**-Based Authentication
  - later the authentication can be changed to password based, but it is not so recommended as password protected key
- **SSH** client software
- Internet access

Creating a Key pair



Navigate to **Compute>Access and Security>Key Pairs**



Click on create Key Pair, name key as **lastname_firstname**

Storing the key pair

Linux and Mac OS X

- Create .ssh directory in ~ if its not there already

```
mkdir -p  .ssh
chmod 700 .ssh
```

- Move key pair to .ssh directory

```
cd .ssh
mv ../Downloads/yourkey.pem .
```

- Make key unreadable by other users

```
chmod 400 yourkey.pem
```

- Protect key with passphrase

```
ssh-keygen -p -f yourkey.pem
```

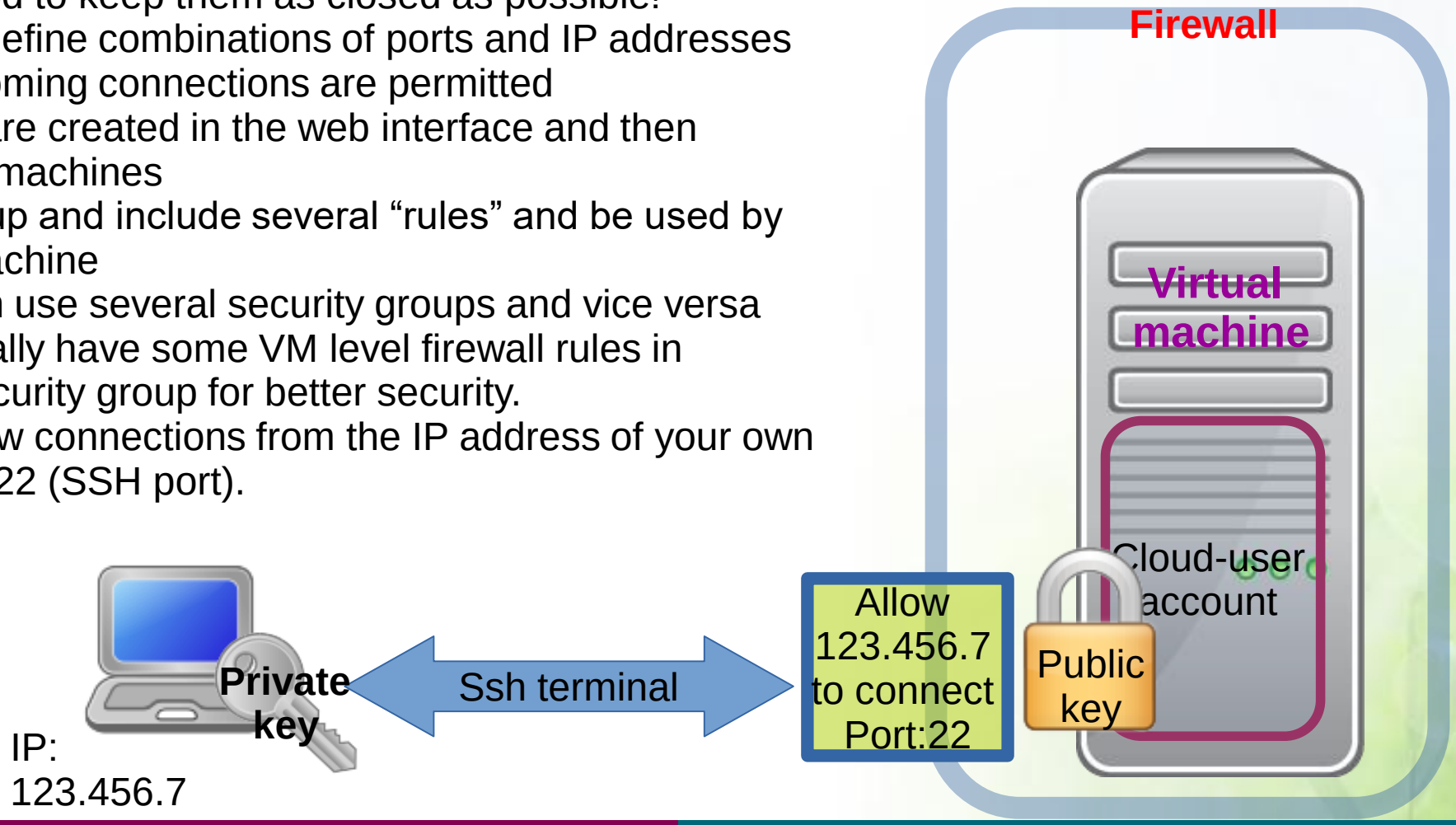
Windows

- Download Putty and Puttygen tools if you don't have them

- Load your **private key (yourkey.pem)** into **puttygen** and change it to .ppk format
- Open Putty, load .ppk file under **Connection | SSH | Auth | Private key file for authentication**
  - Provide user name **cloud-user**
  - Provide password which you added to Puttygen (Optional)

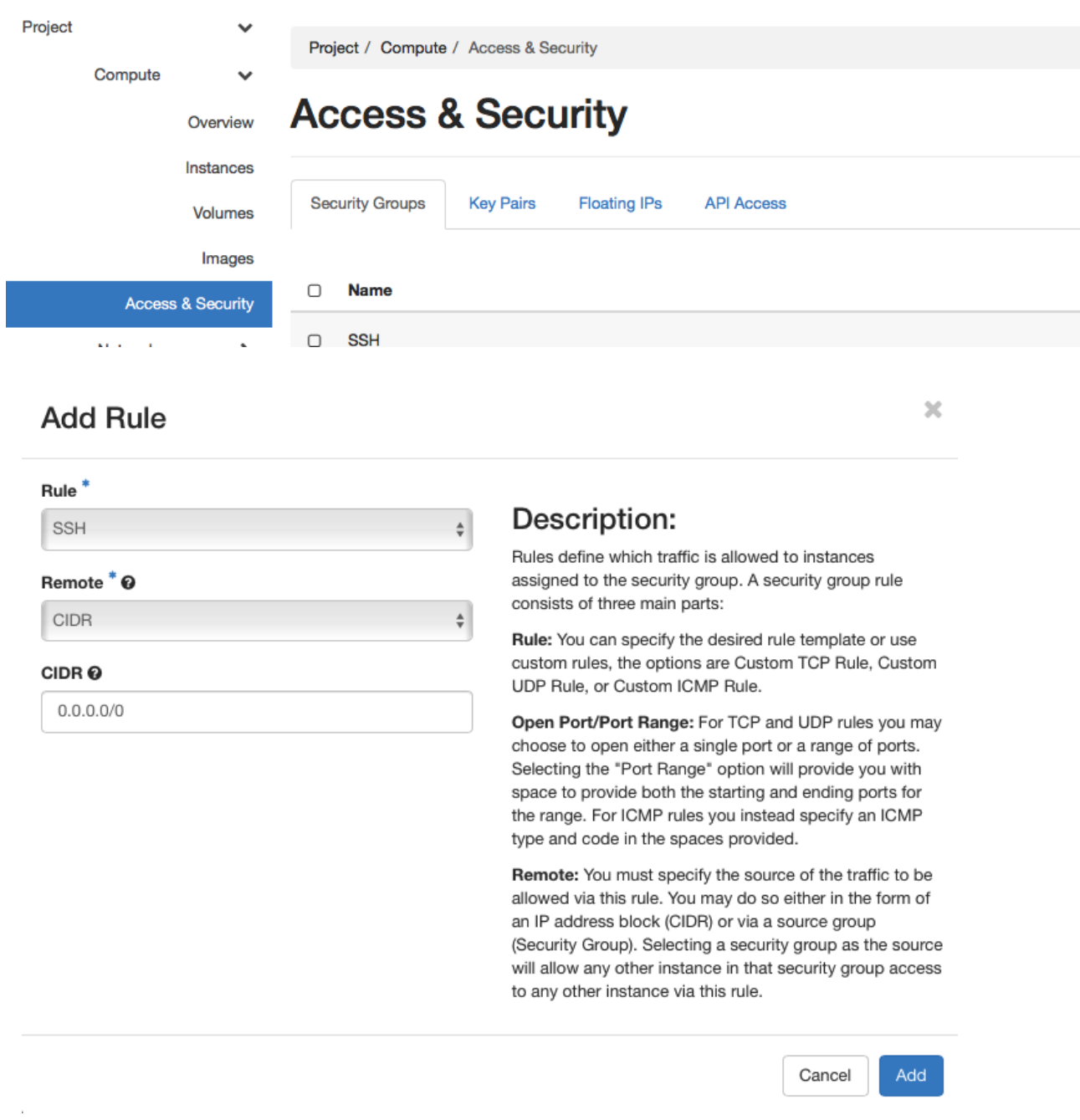
Opening VM to internet with security groups

- A Security Group defines a set of cloud level firewall rules for filtering traffic, typically inbound
- By default Security Groups blocks all incoming connections to your VM. It is good to keep them as closed as possible!
- Security groups define combinations of ports and IP addresses for which the incoming connections are permitted
- Security groups are created in the web interface and then applied to virtual machines
- One security group and include several “rules” and be used by several virtual machine
- One machine can use several security groups and vice versa
- You can additionally have some VM level firewall rules in conjunction to security group for better security.
- Typical case: allow connections from the IP address of your own computer to port 22 (SSH port).



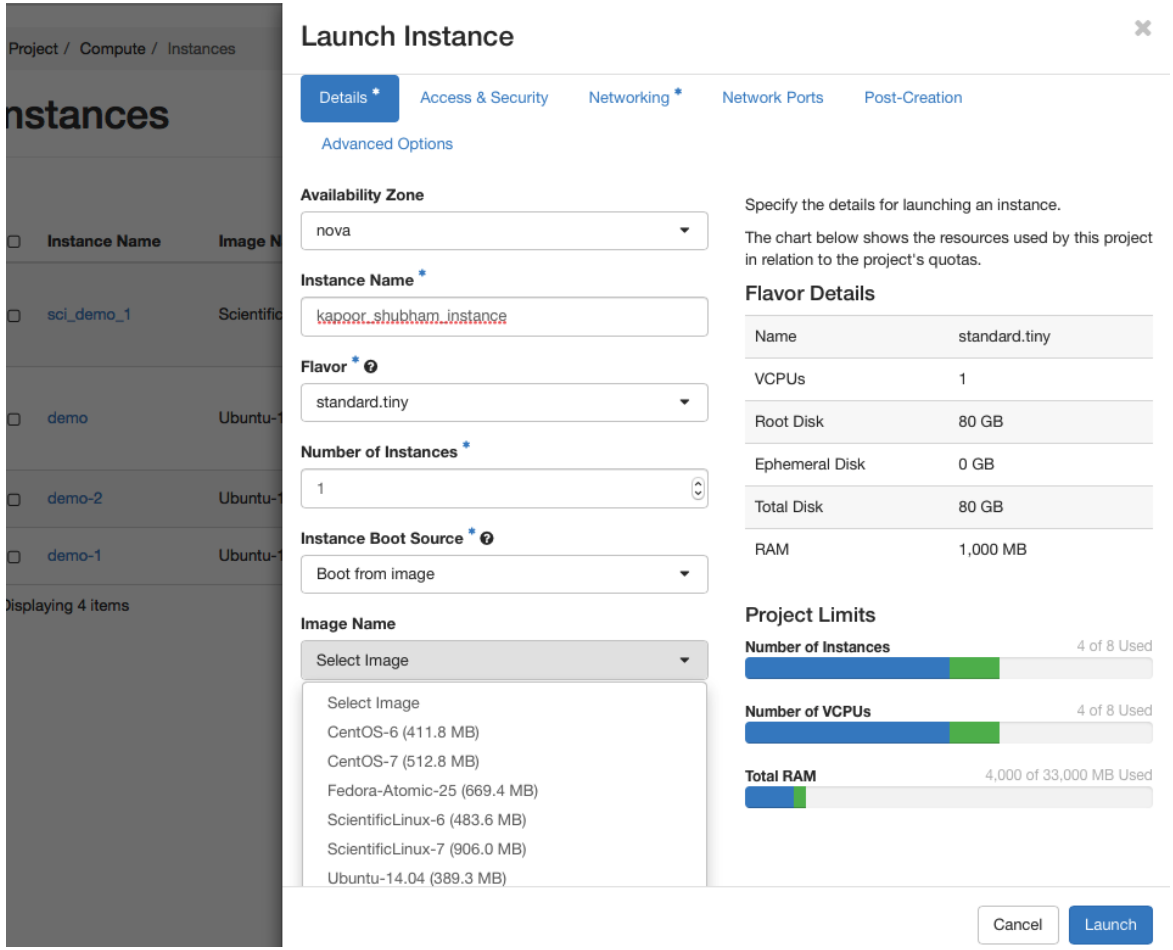
Security Groups

- Created by navigating to **Compute>Access and Security>Create Security Group**
  - Several predefined rule sets are available, such as for SSH
  - At bare minimum you need to select the Source IP for the traffic
  - Modify the **CIDR** field to allow SSH connections only from specific IPs



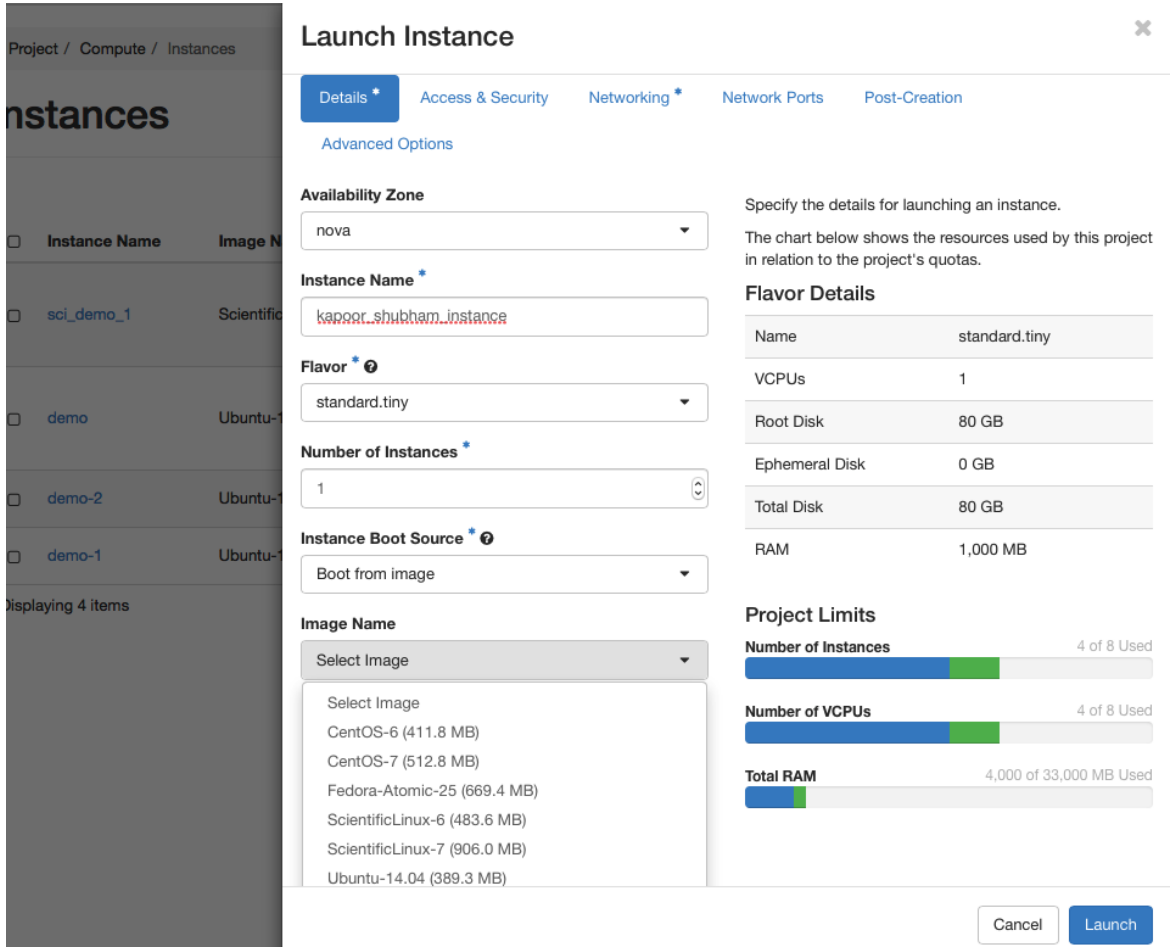
Creating an Instance

- Navigate to **Compute>Instances** and **Launch Instance**
- Give Instance name as **lastname_firstname_instance**
- Select a Flavor of your choice (**standard.tiny** is a good first choice)
- Select Instance Boot Source as **Boot from image**
- Pick an image - any image
- Navigate to **Access & Security** in same popup. Make sure that the "**SSH - World"** Security Group is selected.
- Populate the Post-Creation script as per



Creating an Instance !

- Add SSH key pair to Web UI
- Create instance as before
- In **Access & Security**, make sure that the SSH key pair is selected
- When connecting to the instance, designate the private key into the session or pre-populate it into an SSH agent prior to making a connection



Exercise Set 1: Creating & Securing Virtual Resources

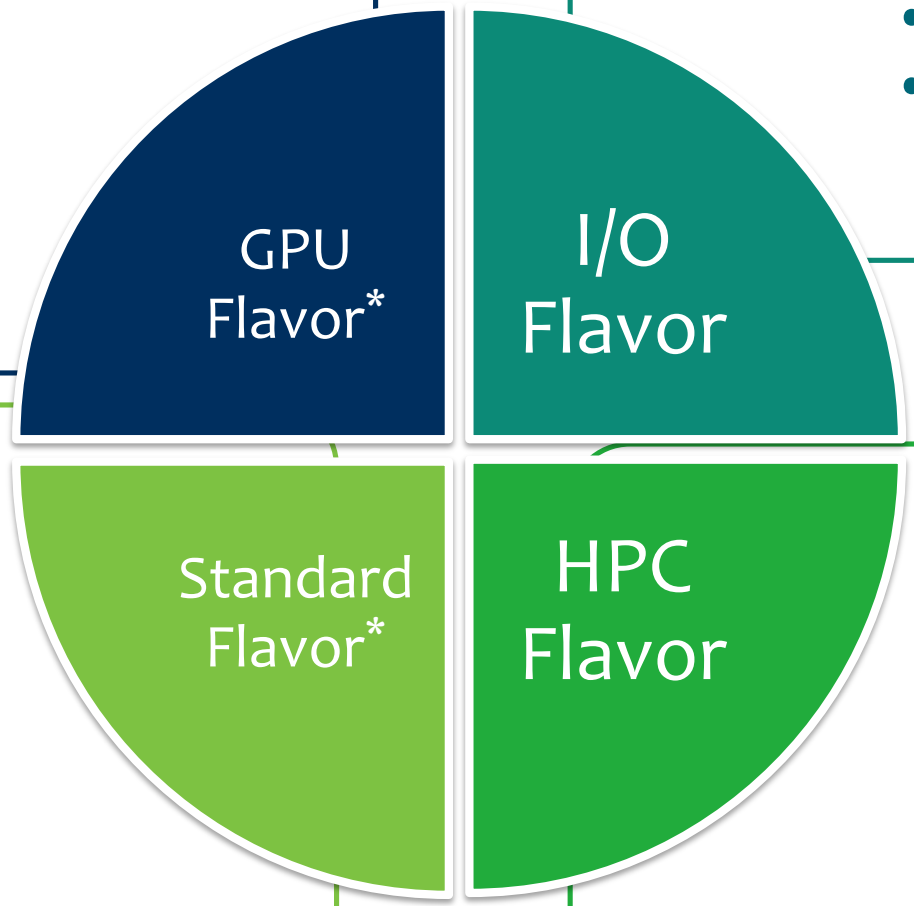
- **Exercise 1 - Creating a temporary Virtual Machine for testing login**
  - Log in to Cloud Dashboard at https://pouta.csc.fi/
  - Create your own Virtual Machine Instance with disposable password in post creation section
  - Associate Floating IP to Virtual Machine Instance
  - Log in to your VM using SSH or Putty
  - Exit and delete the VM
- **Exercise 2 - Creating an SSH key pair for secure login to an Instance**
  - Create an SSH key pair, storing the private key in a safe place
  - Create new VM Instance using this key pair
  - Associate Floating IP address to VM
  - Log in

Exercise Set 1: Creating & Securing Virtual Resources

- **Exercise 3 – Create your own Security Group for securing your virtual resources**
  - Create your own Security Group for SSH traffic
  - Start by creating a wrong Security Group rule
  - Attach it to your VM
  - You would be denied access to your VM
  - Modify Security Group again, this time with correct Security Group rule
  - Connect to your VM

Pouta: Hardware Options

- AI, Deep Machine Learning
- Intensive 3D/Video/Image processing



- Fast local disk needs
- Hadoop/Spark
- Clustered Databases

- Normal Computing & Software Development
- Normal Application/ Web & content services hosting

- Scientific Applications
- Advanced Computing & Software Development

OpenStack

- CSC's cPouta/ePouta cloud services are powered by OpenStack.
    - Current OpenStack version used by Pouta services is Newton

- OpenStack is a cloud software that allows end user to create and use their VM instances, networks and storage.

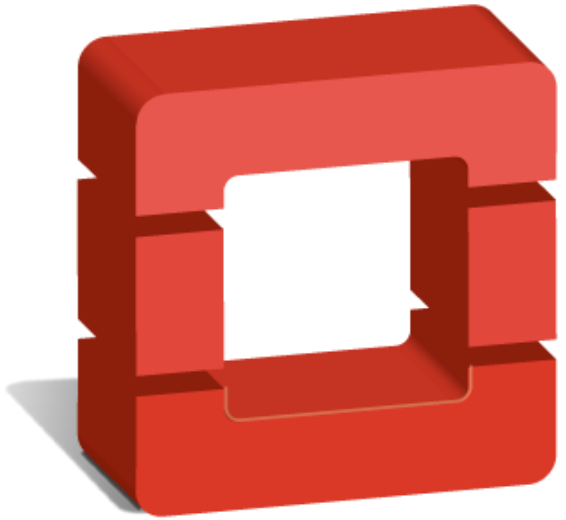
- Fast moving open source project with backing from industrial giants like AT&T, Red Hat, IBM, Intel, HP etc.

- Flexible architecture which may support different types of scalabilities.

- Used by many organizations from research institutes to service/content providers.

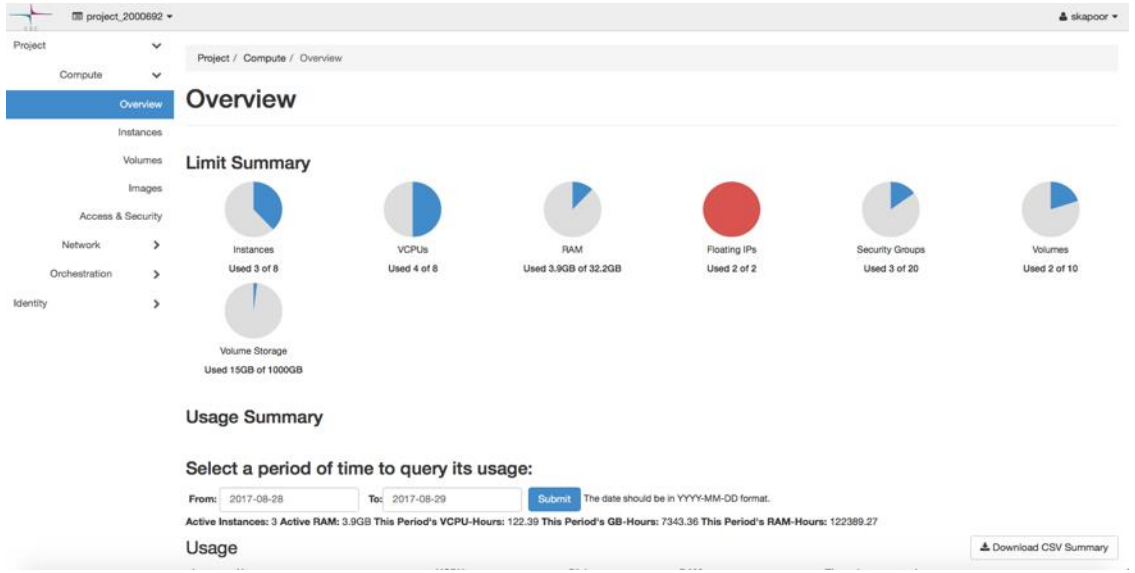
- Large customer base augments better availability of expertise, support and chances of continuity.

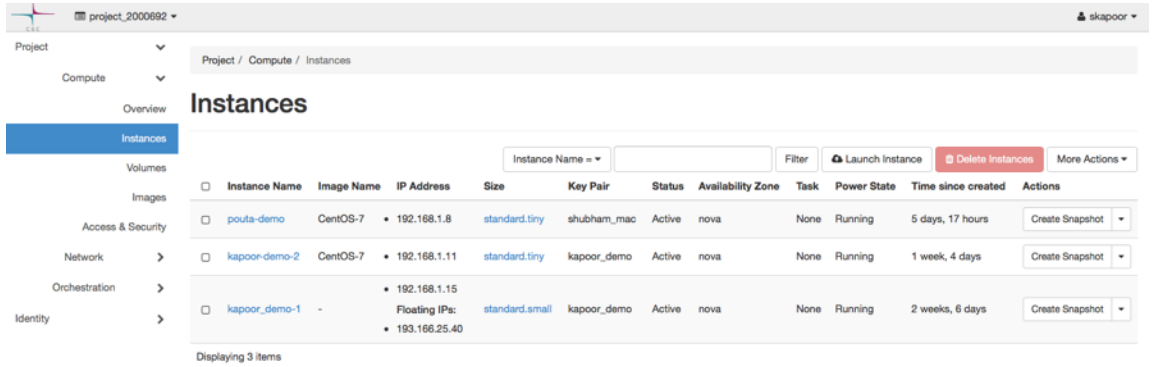
- Supports Web UI, CLI and REST Interfaces

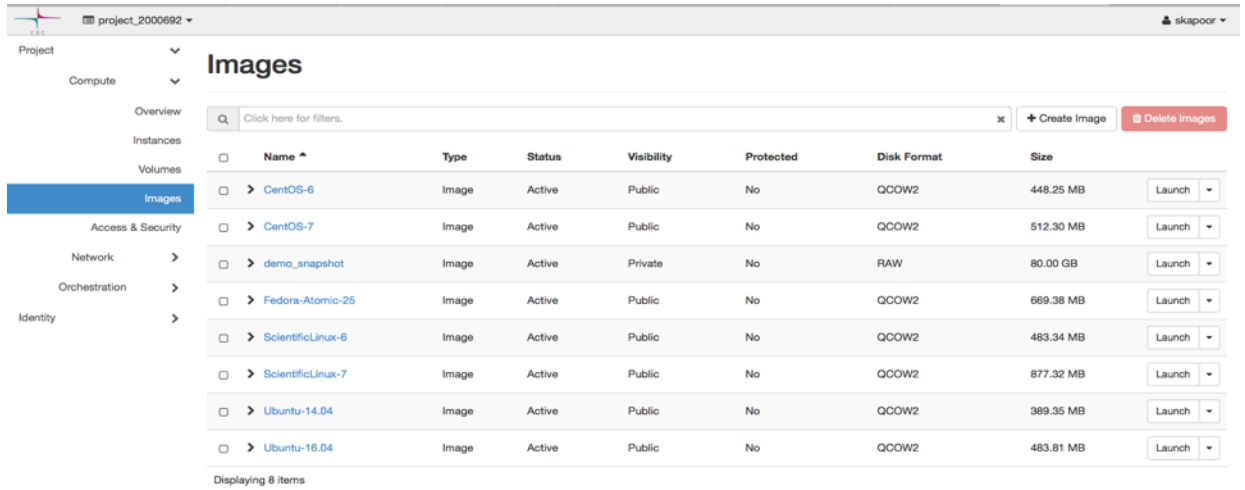


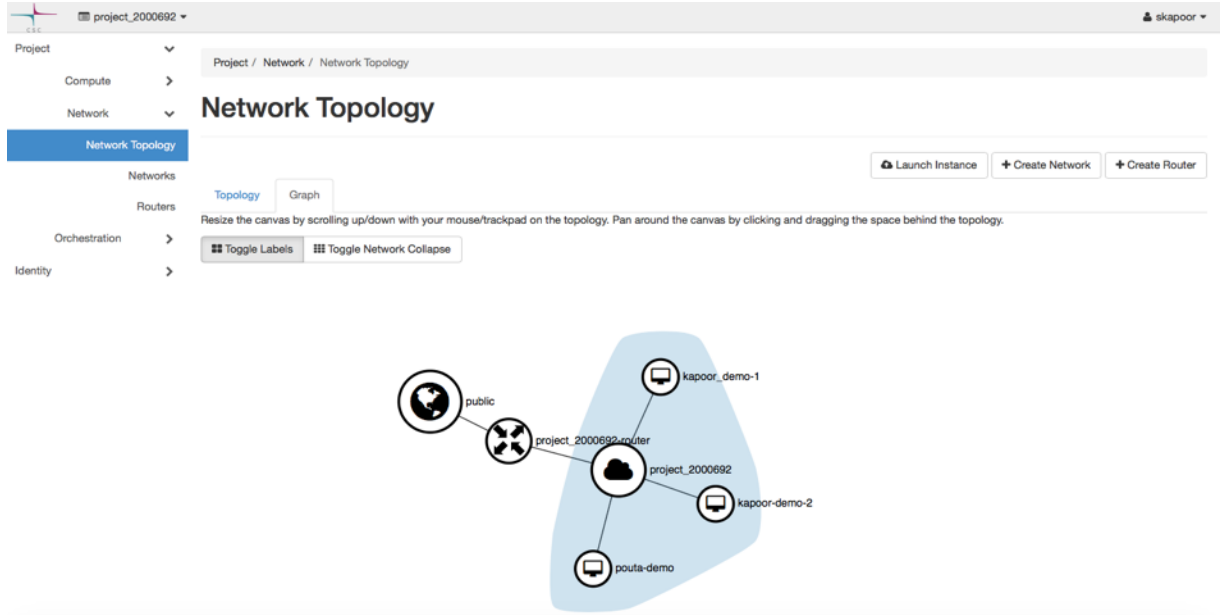
openstack™

OpenStack WebUI









OpenStack CLI

```
(osclient) skapoor-air13:python_virtualenvs skapoor$ openstack image list
+--------------------------------------+--------------------+--------+
| ID                                   | Name               | Status |
+--------------------------------------+--------------------+--------+
| 4a36f474-4ffe-4f88-bc9f-dad674ef48d2 | CentOS-6           | active |
| 7add5463-20a9-4d2e-8bd8-b38d959aa83f | CentOS-7           | active |
| 5ad9d51b-b6eb-44e8-98b6-9d7f69cac5df | Fedora-Atomic-25   | active |
| c42266c9-7e05-45bd-a434-287539c0dc90 | ScientificLinux-6  | active |
| 1d9a34dc-2a79-41c2-b787-4193a9c5b726 | ScientificLinux-7  | active |
| 669bef35-f60a-4bea-93cc-a57348af2ff1 | Ubuntu-14.04       | active |
| 6cd4708e-fcb0-4dbc-92f5-faf4e9aa7424 | Ubuntu-16.04       | active |
| be8c32a5-e1c2-4584-b79c-1fb6caaf4501 | demo_snapshot      | active |
+--------------------------------------+--------------------+--------+
```

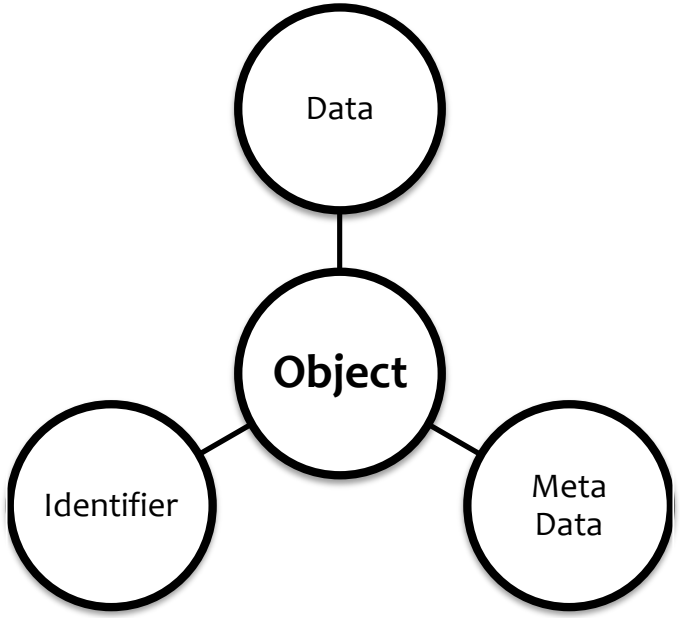
```
(osclient) skapoor-air13:python_virtualenvs skapoor$ openstack server list
+--------------------------------------+-----------------------+--------+-----------------------------------------+------------------
| ID                                   | Name                  | Status | Networks                                | Image
| Flavor       |
+--------------------------------------+-----------------------+--------+-----------------------------------------+------------------
| a8d5f4f8-5659-4599-93ef-c32a2c96ddf8 | kapoor_shubham_instance | ACTIVE | project_2000692=192.168.1.8             | Ubuntu-16.04
(osclient) skapoor-air13:python_virtualenvs skapoor$ openstack keypair show kapoor_shubham
+-------------+-------------------------------------------------+
| Field       | Value                                           |
+-------------+-------------------------------------------------+
| created_at  | 2017-09-15T09:24:15.000000                      |
| deleted     | False                                           |
| deleted_at  | None                                            |
| fingerprint | ad:3f:45:ff:de:09:65:be:84:f3:e7:ab:22:36:57:9e |
| id          | 183015                                          |
| name        | kapoor_shubham                                  |
| updated_at  | None                                            |
| user_id     | skapoor                                         |
+-------------+-------------------------------------------------+
```

```
(osclient) skapoor-air13:python_virtualenvs skapoor$ openstack flavor list
+--------------------------------------+-----------------+--------+------+-----------+-------+-----------+
| ID                                   | Name            |    RAM | Disk | Ephemeral | VCPUs | Is Public |
+--------------------------------------+-----------------+--------+------+-----------+-------+-----------+
| 0143b0d1-4788-4d1f-aa04-4473e4a7c2a6 | standard.tiny   |   1000 |   80 |         0 |     1 | True      |
| 053c4852-dd1e-42dc-947a-fe4263548fa9 | hpc-gen2.48core | 240000 |   80 |         0 |    48 | True      |
| 110eb004-f7cc-474b-8158-14bb244cb05e | hpc-gen2.24core | 120000 |   80 |         0 |    24 | True      |
| 1792db39-f38e-43ba-ae95-96b7549b4f84 | standard.xlarge |  16000 |   80 |         0 |     6 | True      |
| 27d232d6-d245-4cf4-8ab9-a0424005184b | hpc-gen2.8core  |  40000 |   80 |         0 |     8 | True      |
| 2f24b080-287f-49a9-8219-2295cde364c3 | hpc-gen2.16core |  80000 |   80 |         0 |    16 | True      |
| 41ec2177-604b-492c-8f19-f2d7c2bc8c07 | io.70GB         |  10000 |   20 |        70 |     2 | True      |
| 544e940c-4b9b-4f54-ab6f-f1ee1792fe48 | hpc-gen2.2core  |  10000 |   80 |         0 |     2 | True      |
| 58bbbf4c-e174-485f-b050-b0cc86c0f677 | hpc-gen1.16core |  60000 |   80 |         0 |    16 | True      |
| a82b2b5f-6788-41fd-80cb-ed7576ee1e7c | hpc-gen1.8core  |  30000 |   80 |         0 |     8 | True      |
| af9fa76e-818a-421e-9142-0341e7818d90 | io.340GB        |  40000 |   20 |       340 |     8 | True      |
| ba8f9270-93fe-47ee-b402-714a1352f190 | hpc-gen1.1core  |   3750 |   80 |         0 |     1 | True      |
| c0c7bb30-2679-4e0d-94ab-4395237f505e | hpc-gen1.4core  |  15000 |   80 |         0 |     4 | True      |
| c1da3536-f22d-426e-bc14-ef994f1bfaa7 | io.700GB        |  80000 |   20 |       700 |    16 | True      |
| c5ffaed0-6707-4a99-9498-9ef6d34c8add | io.160GB        |  20000 |   20 |       160 |     4 | True      |
| d4a2cb9c-99da-4e0f-82d7-3313cca2b2c2 | standard.small  |   2000 |   80 |         0 |     2 | True      |
| e7b3364e-f70c-4e3b-8e5a-fa249759d14c | standard.large  |   8000 |   80 |         0 |     4 | True      |
| f363d088-4967-48ff-bc80-86c0d05ff418 | standard.medium |   4000 |   80 |         0 |     3 | True      |
+--------------------------------------+-----------------+--------+------+-----------+-------+-----------+
```

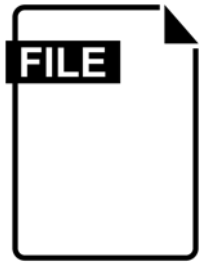
```
(osclient) skapoor-air13:python_virtualenvs skapoor$ openstack server create --flavor standard.tiny --image 6cd4708e-fcb0-4dbc-92f5-faf4e9aa7424 --key-name kapoor_shubham kapoor_shubham_instance_2
+-----------------------------+-----------------------------------------------------+
| Field                       | Value                                               |
+-----------------------------+-----------------------------------------------------+
| OS-DCF:diskConfig           | MANUAL                                              |
| OS-EXT-AZ:availability_zone |                                                     |
| OS-EXT-STS:power_state      | NOSTATE                                             |
| OS-EXT-STS:task_state       | scheduling                                          |
| OS-EXT-STS:vm_state         | building                                            |
| OS-SRV-USG:launched_at      | None                                                |
| OS-SRV-USG:terminated_at    | None                                                |
| accessIPv4                  |                                                     |
| accessIPv6                  |                                                     |
| addresses                   |                                                     |
| adminPass                   | VAYj6QiSnN7t                                        |
| config_drive                |                                                     |
| created                     | 2017-09-15T12:07:17Z                                |
| flavor                      | standard.tiny (0143b0d1-4788-4d1f-aa04-4473e4a7c2a6) |
| hostId                      |                                                     |
| id                          | 61076662-6ca5-44af-93b4-7b1b832a644a                |
| image                       | Ubuntu-16.04 (6cd4708e-fcb0-4dbc-92f5-faf4e9aa7424) |
| key_name                    | kapoor_shubham                                      |
| name                        | kapoor_shubham_instance_2                           |
| progress                    | 0                                                   |
| project_id                  | 2d9e321be82f4066a3824284ce47b17d                    |
| properties                  |                                                     |
| security_groups             | name='default'                                      |
| status                      | BUILD                                               |
| updated                     | 2017-09-15T12:07:18Z                                |
| user_id                     | skapoor                                             |
| volumes_attached            |                                                     |
+-----------------------------+-----------------------------------------------------+
```

What is Object Storage

- Object storage is a computer data storage architecture that manages data as objects.
- Each object has three things: Data, Metadata and Globally unique identifier.
- Different from other data storage architectures like File Storage: Data as a file hierarchy and Block Storage: Data as blocks within sectors & tracks.
- Accessed via APIs at application-level, rather than via OS at system level.
- Scalable and Self healing storage.

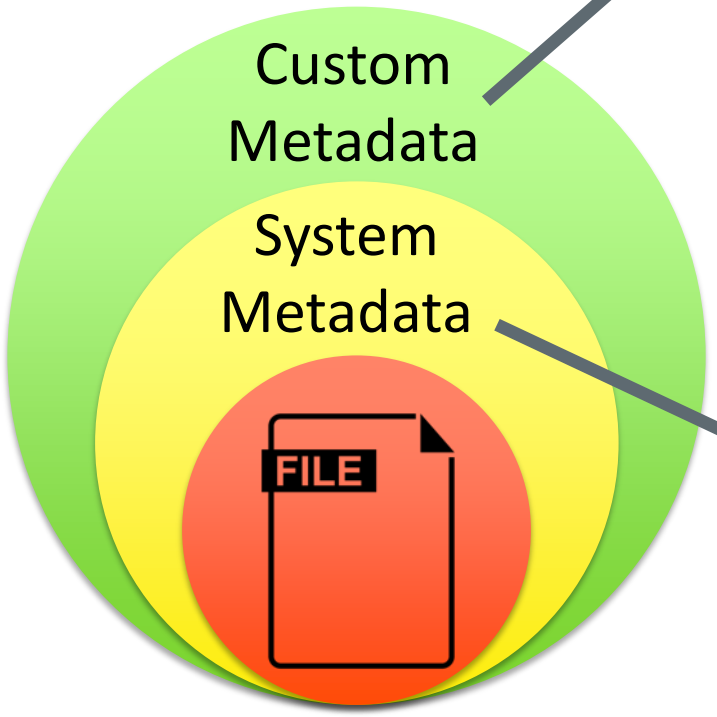


File Storage vs Object Storage



File Name: CTSCAN_Kapoor
Created by: User1
Created on: 19-09-2017
File Type: DICOM

File Storage



Object ID: 123456
Patient Name: Shubham
Patient ID: 23242
Physician Name: Dr. John
Prior1 : XYZ.DICOM
Self Destruct: 2 Year

File Name: CTSCAN_Kapoor
Created by: User1
Created on: 19-09-2017
File Type: DICOM

Object Storage

Where Object Storage Fits

On Basis of Data Type

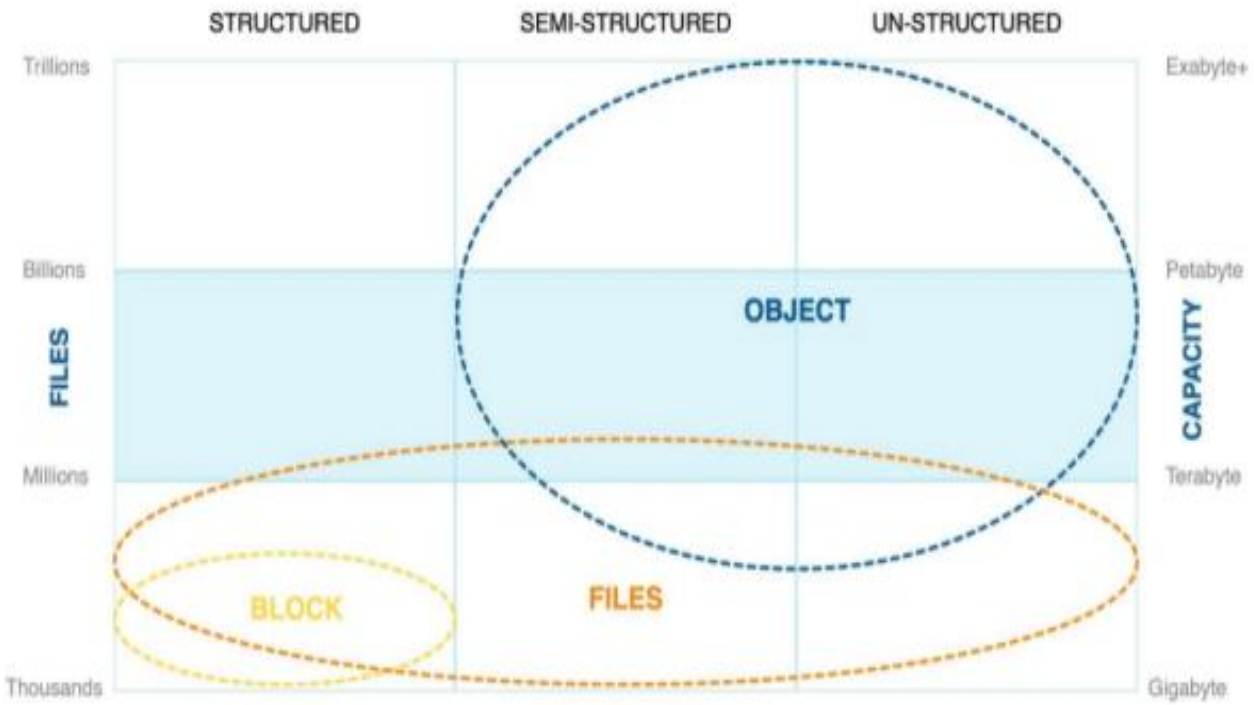
- Storage of Unstructured/ Semi structured Data like Media files, web contents, Backup Archives etc.

- Cold Storage of structured and semi structured data like Databases, Sensor Data, Log files etc.

- Archiving files in place of local tape drives.

- Big Data, large data sets

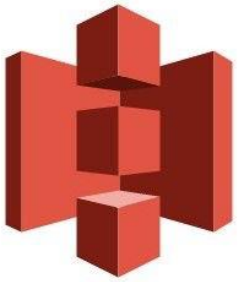
On Basis of Data Size



Where Object Storage Doesn't Fit

- Hot Data.
- Relational/OLTP Databases.
- Latency intolerant applications.
- Data with **Strict** consistency requirements.

Object Storage Around us



Amazon S3

Microsoft Azure
Blob Storage

IBM Cloud
Object Storage

Google Cloud Storage

- Social Media Storage
- Big Data Analytics
- Offsite Backup/Archive
- Static Website Hosting
- Digital Archives

Persistent Data Volumes

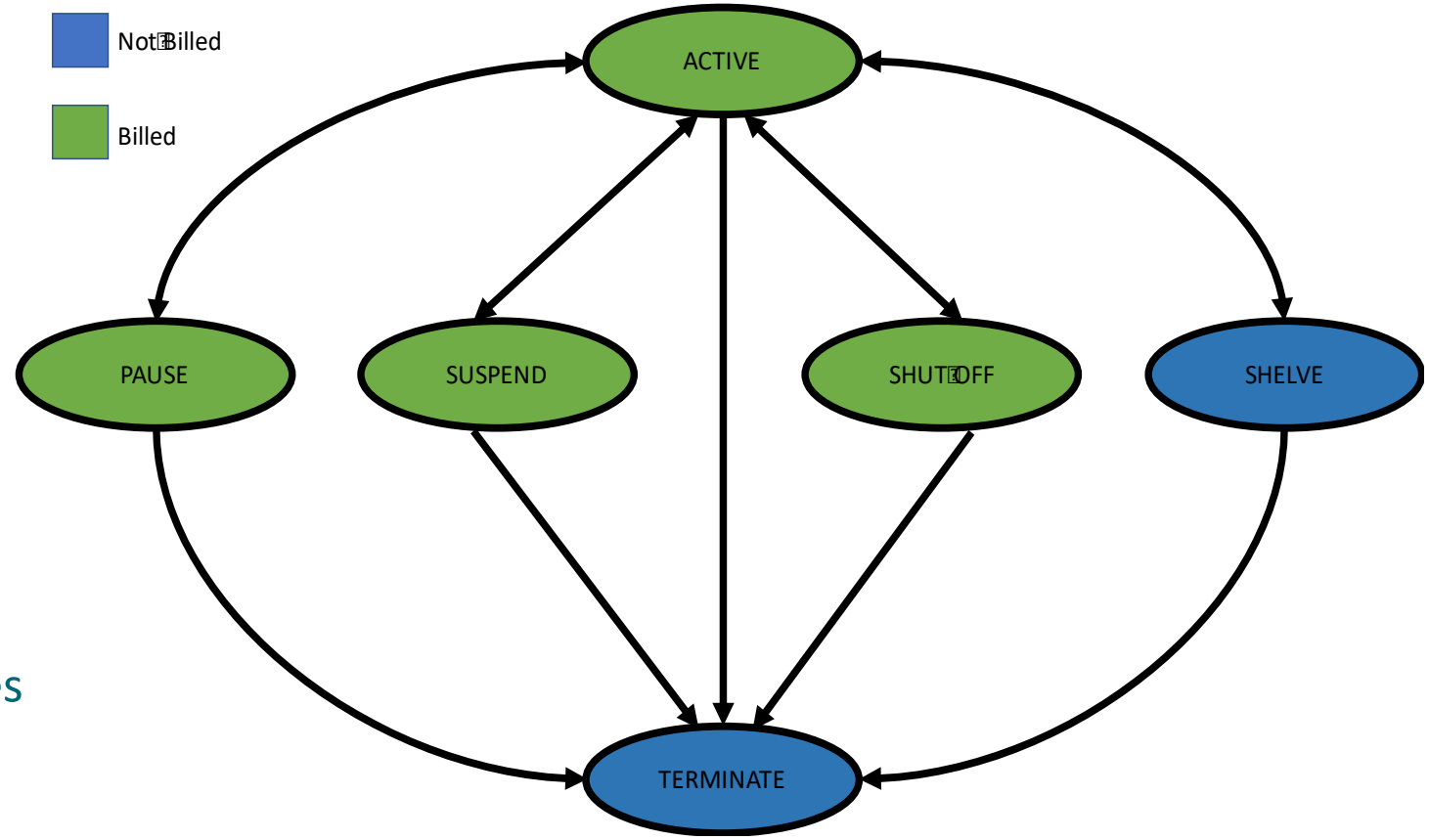
- In Pouta, VM:s have only small local (virtual) disk
- Virtual data volumes can be created.
- Volumes can be attached one VM at the time.
- Volume sizes vary between 1-50 TB (or more)
- A project can have several volumes
- Management with web interface or command line client
- In a volume, data is preserved even though the VM is accidentally deleted, or become in accessible.
- Volumes are project specific, not user specific
- No backup!

Pouta : Managing Project

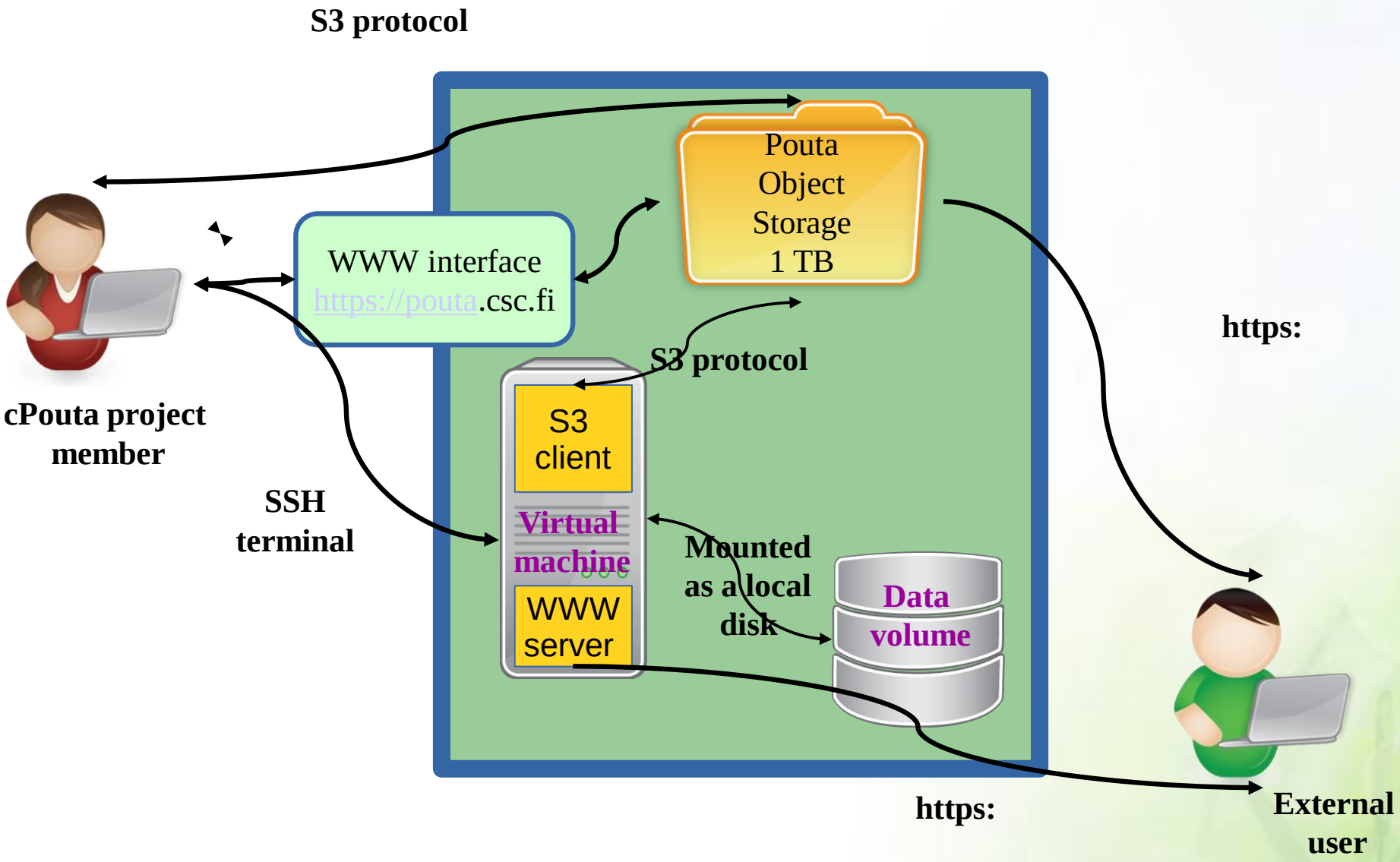
- A Pouta project contain a set of resources: cores, memory, storage, ip-addresses
- A default project contains:
  - For cPouta: 8 cores, 32 GB memory, 1 TB disk space, 2 floating IP addresses .
  - For ePouta: Negotiated between customer and CSC
- If needed you can ask for more resources for your project.
- Project members can build one or several VMs and volumes based on the granted resources.
- When VMs and Volumes are active they are consuming billing units (even if no one is using them).
- Project members can manage other members machines and volumes too.
- Your CSC account can be a member in many cPouta projects.

Pouta : VM States

- **Active** – Consumes billing units regardless of the real usage.
- **Shut off** – Not active, but still reserves the resources. Consumes still billing units.
- **Suspended** – Temporarily suspended. Current state saved. Can be revoked. Consumes billing units
- **Shelved** – VM is shut off, resources are freed & Sate is saved. Can be later on revoked if resources are free (un-shelved). Does not use billing units.
- **Terminate** – Removes the Virtual Machine.



cPouta in action





Building Application Stack on Pouta VM

CSC – Suomalainen tutkimuksen, koulutuksen, kulttuurin ja julkishallinnon ICT-osaamiskeskus

Installing software to your VM

- The VM images include just the basic linux tools.
- You can/must add the tools you need with using tools like
- **System level repository installation**:
  - Centos and Scientific linux: **sudo yum**
  - Ubuntu: **sudo apt**
  - add missing libraries and linux commands and many applications too
- **Compile codes or download pre-compiled binaries.**
- Install Docker and use Docker images.
- Use Conda!

sudo command for system administration

The default user, cloud-user, does not have superuser rights, but can do admin operations with **sudo**.

sudo *linux-command-to-execute*

- Repository installations
- System libraries and directories
- User accounts

e.g.

```
sudo reboot
sudo yum install nano
sudo nano /etc/yum.conf
sudo useradd teppo
```

Repository installation in Centos and RedHat with yum

System wide installation of libraries and tools
Many application software are also available this way

`yum help`	List subcommands and options
`sudo yum install` ***`package`***	Install a package from repository
`sudo yum update`	Update one or all packages in the system
`yum provides` ***`filename`***	Check what packages include the defined file
`yum search` ***`term`***	Search package names and descriptions
`sudo yum localinstall` ***`package`***	Install locally available rpm file
`sudo yum remove` ***`package`***	Remove a package

Repository installation in ubuntu with apt-get

System wide installation of libraries and tools
Many application software are also available this way

`apt-get --help`	List commands and options
`sudo apt-get install` ***`package`***	Install a package from repository
`sudo apt-get update`	Update one or all packages in the system
`apt-file` ***`filename`***	Check what packages include the defined file
`apt-cache search` ***`term`***	Search package names and descriptions
`sudo yum localinstall` ***`package`***	Install locally available rpm file
`sudo apt-get remove` ***`package`***	Remove a package

Conda /Bioconda

- Easy way to install software tools together with their dependencies
- Bioconda- repository contains over 700 bioscience tools
- Does not need superuser privileges
- For installing conda and browsing bioconda packages, check bioconda home page:

  https://bioconda.github.io/

- Once you have conda installed, you can install application software with commands like:

```
conda create -n aligners bwa bowtie hisat star
source activate aligners
bwa
```