EGI Federated Cloud for Open Science

Diego Scardaci (EGI.eu), Fotis Psomopoulos (AUTH), Kimmo Mattila (CSC)

diego.scardaci@egi.eu,
fpsom@issel.ee.auth.gr,
kimmo.mattila@csc.fi

http://go.egi.eu/cloud

www.egi.eu

EGI-Engage is co-funded by the Horizon 2020 Framework Programme of the European Union under grant number 654142

Training goals

1. Learn the concept of IaaS cloud computing
2. Learn the conceptual model of the EGI federated cloud
3. Obtain skills in using the standard interfaces of the EGI federated cloud
   - With pre-defined Virtual Machine images
   - With customised deployments (contextualisation)
   - With Docker containers
   - (With your own images)
4. Learn how to deploy bioinformatic applications in the EGI federated cloud
5. Learn how to become an active user
6. Learn how to book the EGI training infrastructure for your tutorials

Outline

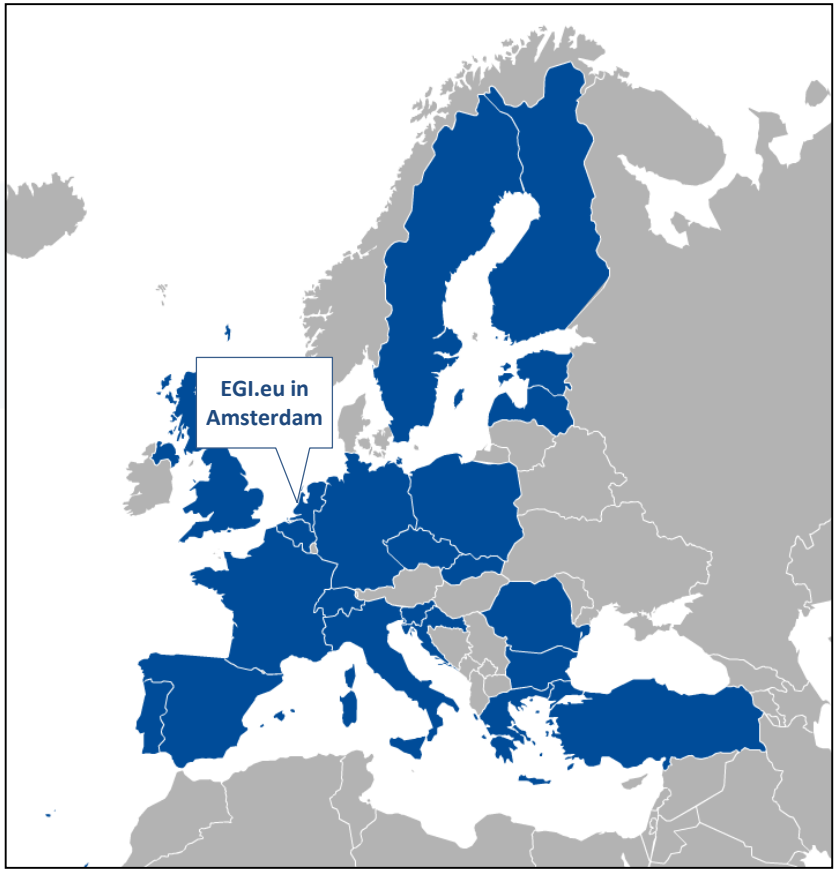
- Introduction to EGI, & EGI Federated Cloud (20')
- Introduction to training infrastructure (10')
- Exercise 1 (60'): Run Chipster in the EGI Federated Cloud
- BREAK (15')
- Exercise 2&3 (45')
  - Compute management – Setup a Jupyter Notebook
  - Persistent storage – Add block storage to the Jupyter Notebook
- Introduction to contextualisation (10')
- Exercise 4: Contextualised compute service – Fractal application (Home Work)
- Introduction to Docker (5')
- Exercise 5: Running a Galaxy Docker container in the EGI Cloud (30')
- Prepare your VM images and next steps (10')
- Feedback forms (5')

EGI-Engage

Introduction to EGI & EGI Federated Cloud

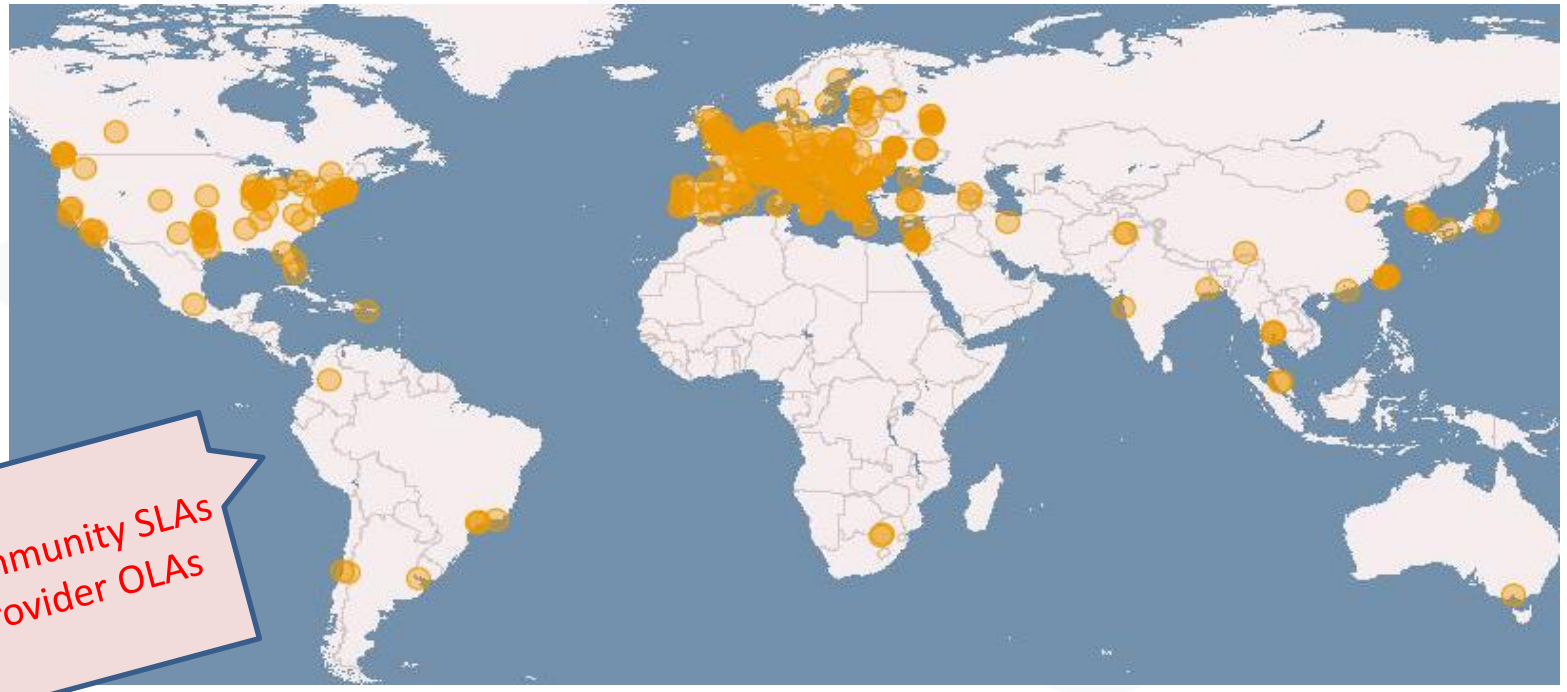
EGI ~~(European Grid Infrastructure)~~

- EGI Council
  - 26 participants: 22 NGIs and 2 EIROs (CERN, EBI)
  - Opening membership to research communities
  - Affiliation programme for countries
- Shared interest in
  - Developing and providing e-infrastructure services that enable open science
- Sustainability
  - Sustainable core services (HW, SW, human!)
  - Project-driven innovation
    - H2020: EGI-Engage, AARC, INDIGO-Datacloud, ELI-Trans, etc.



Membership under discussion
Armenia, Austria, Belarus, Denmark, Ireland, Moldova, Norway, Russia, Ukraine

‘Enabling Global Infrastructures’



Community SLAs
Provider OLAs

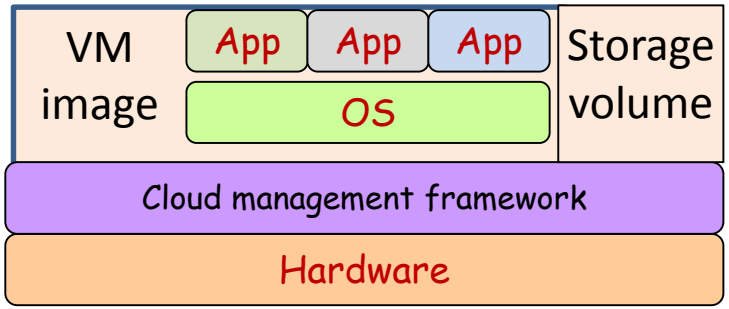
- Distributed, federated storage and compute facilities
- Compute platforms (Grid, Cloud)
- Virtual Research Environments
- > 200 user research projects

Total capacity:

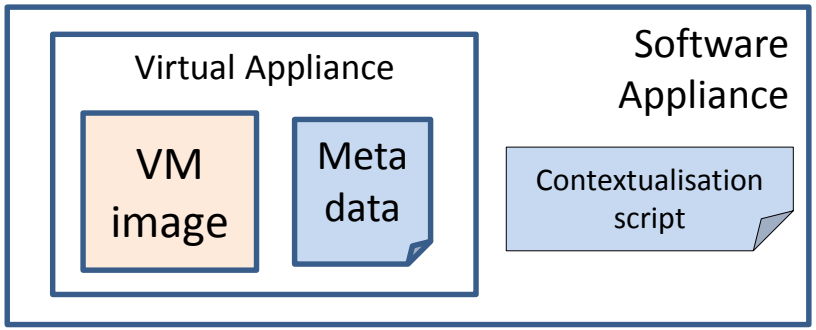
- 340 resource centres in 54 countries
- 550,000 logical CPU cores
- 290 PB disk, 180 PB tape

Cloud computing & Key terms

- Services and solutions delivered and consumed in real time over the Internet
- (Some of the) benefits
  - Virtualisation – Platform-independence; Self-servicing
  - Scalability – 'Pay-as-you-go'; Multi-tenant allocation
  - Predictability – Versioning of VMs and contextualisation scripts
  - Abstractions – IaaS, PaaS, SaaS
  - Open source – KVM, OpenStack, OpenNebula, …



Virtualized Stack

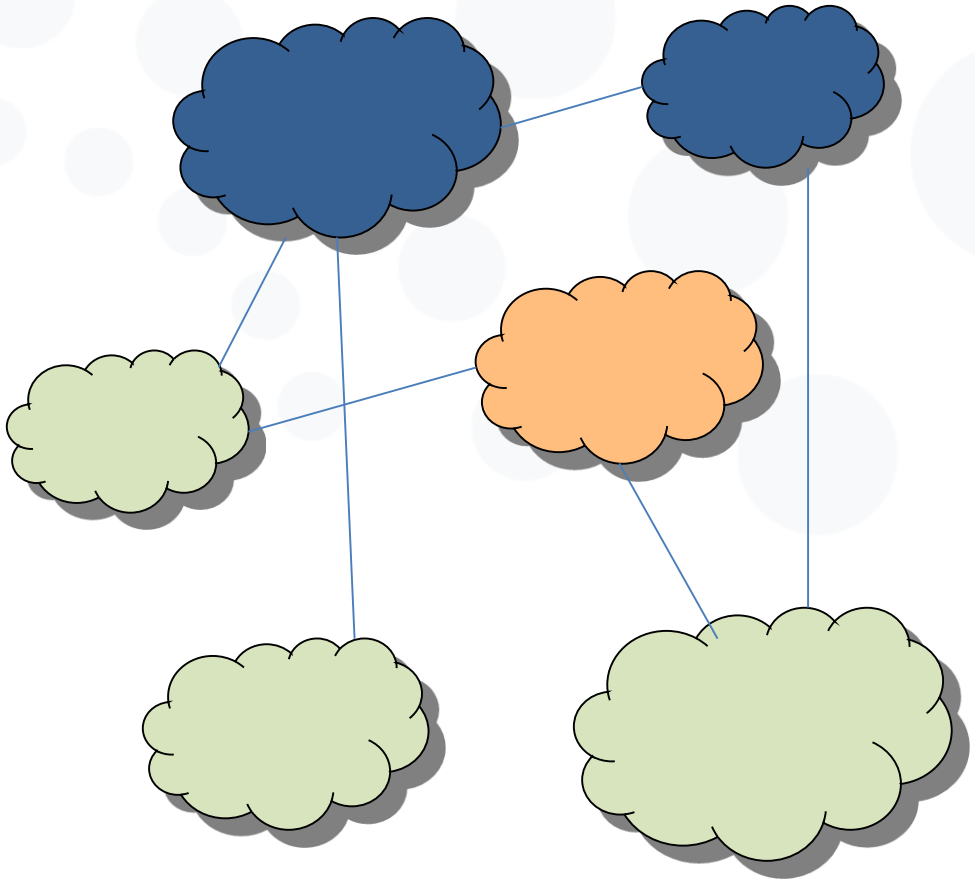


What is a cloud federation? – A 'definition'

- Practice of interconnecting cloud service providers. Motivations:
  - Data locality; Data privacy; Shared investment; Distributed expertise, …
- Multiple cloud sites with some sort of interconnection(s). Examples:
  - Every cloud registered in a single catalo
  - Single VM image catalogue for
  - Support for the sam
  - Automated
  - Si
  - , integrated monitoring, accounting, etc.
  - pport model
  - icketing system, consultancy, training

EGI Federated Cloud can do all this

EGI Federated Cloud



- Grid of clouds
- Unified user interfaces
- Harmonised operational behaviour
- Clouds and their interconnections are based on open standards, open technologies

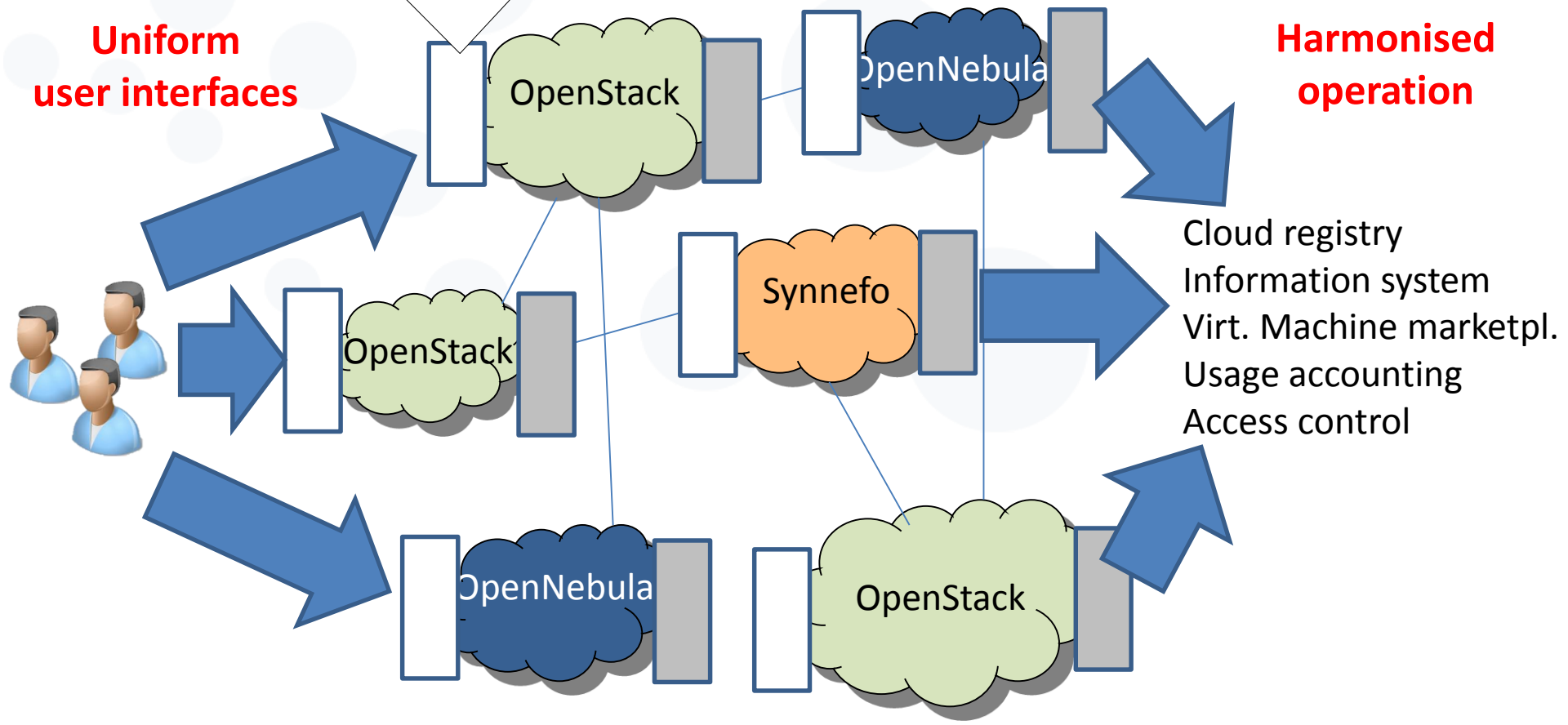
- Infrastructure → Access
  AND
  technology → Deploy

efits

TOPIC OF THIS TUTORIAL

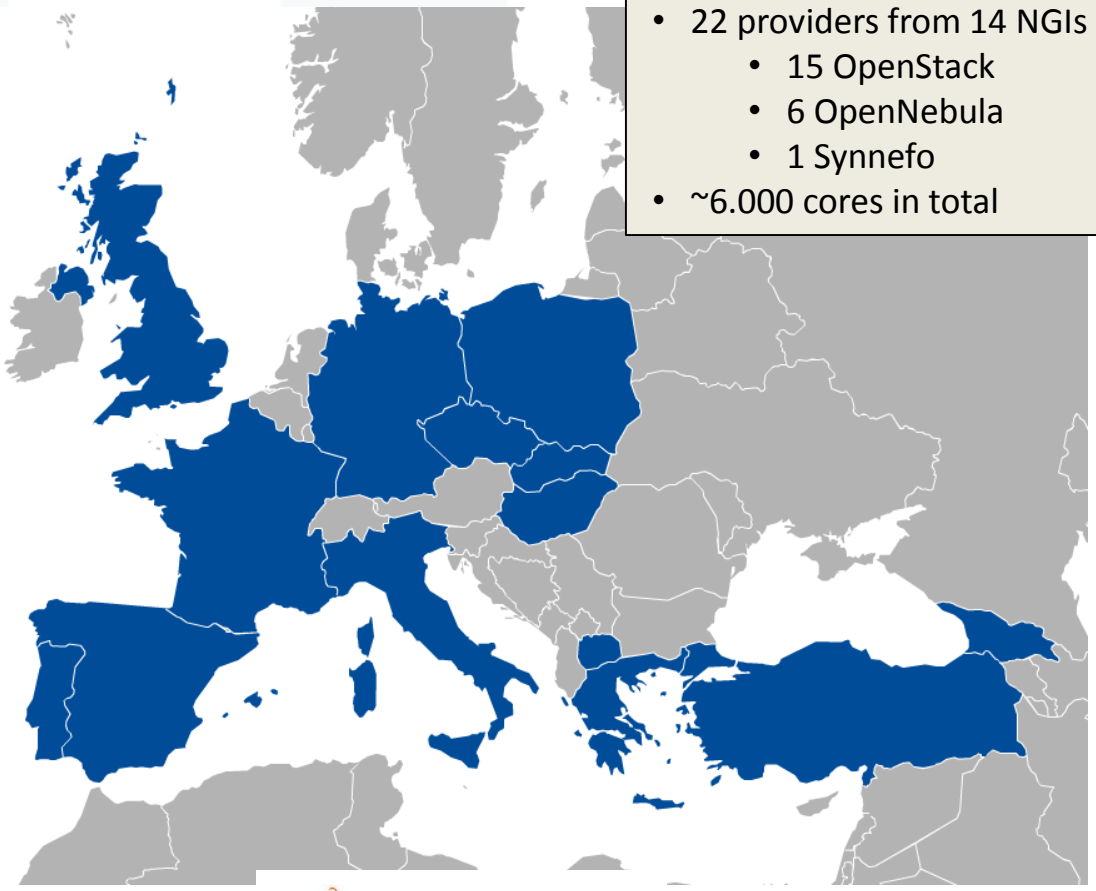
Occi - On every site

OpenStack Nova - On OS sites

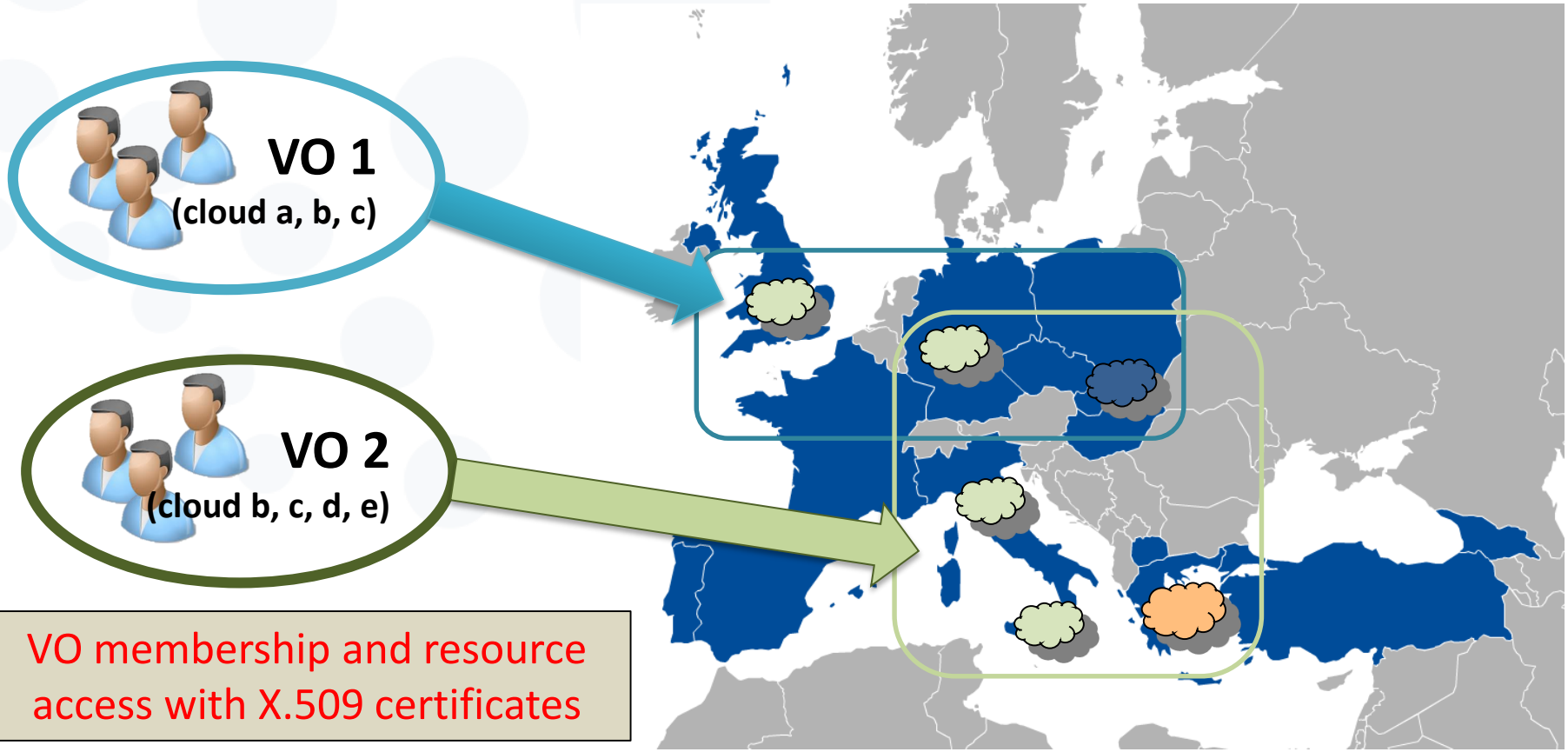


The current infrastructure

Today:
- 22 providers from 14 NGIs
  - 15 OpenStack
  - 6 OpenNebula
  - 1 Synnefo
- ~6.000 cores in total



Access to the Federated Cloud: Virtual Organisations

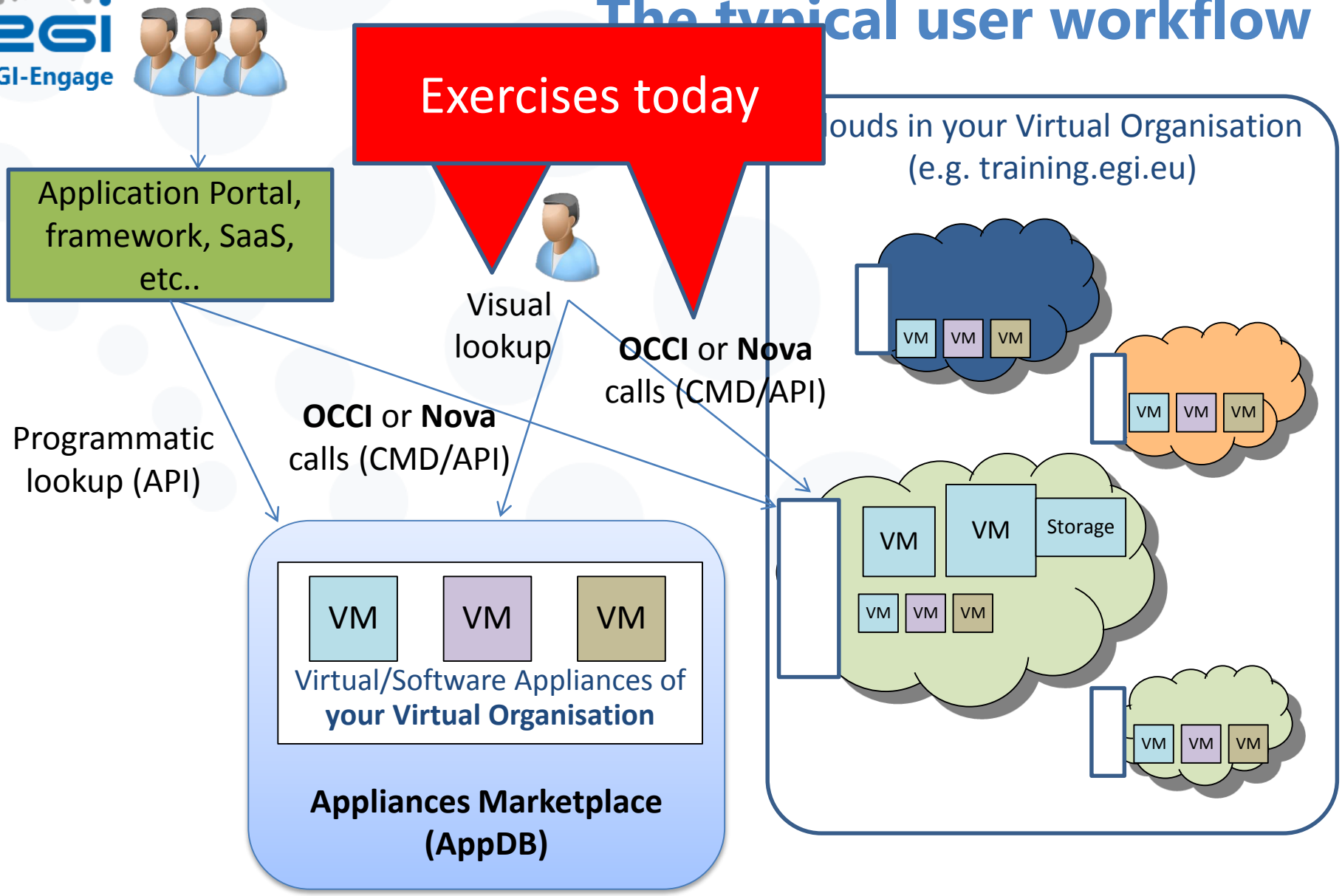


VO membership and resource access with X.509 certificates

1. Generic VOs – e.g. fedcloud.egi.eu → Incubator for new users
2. Community-specific VOs – e.g. CHIPSTER, Highthroughtputseq, EISCAT, etc. (SLA, OLAs)
3. Training VO = training.egi.eu → To be used today

Browse VOs at http://operations-portal.egi.eu/vo/search (both grid and cloud)

The typical user workflow

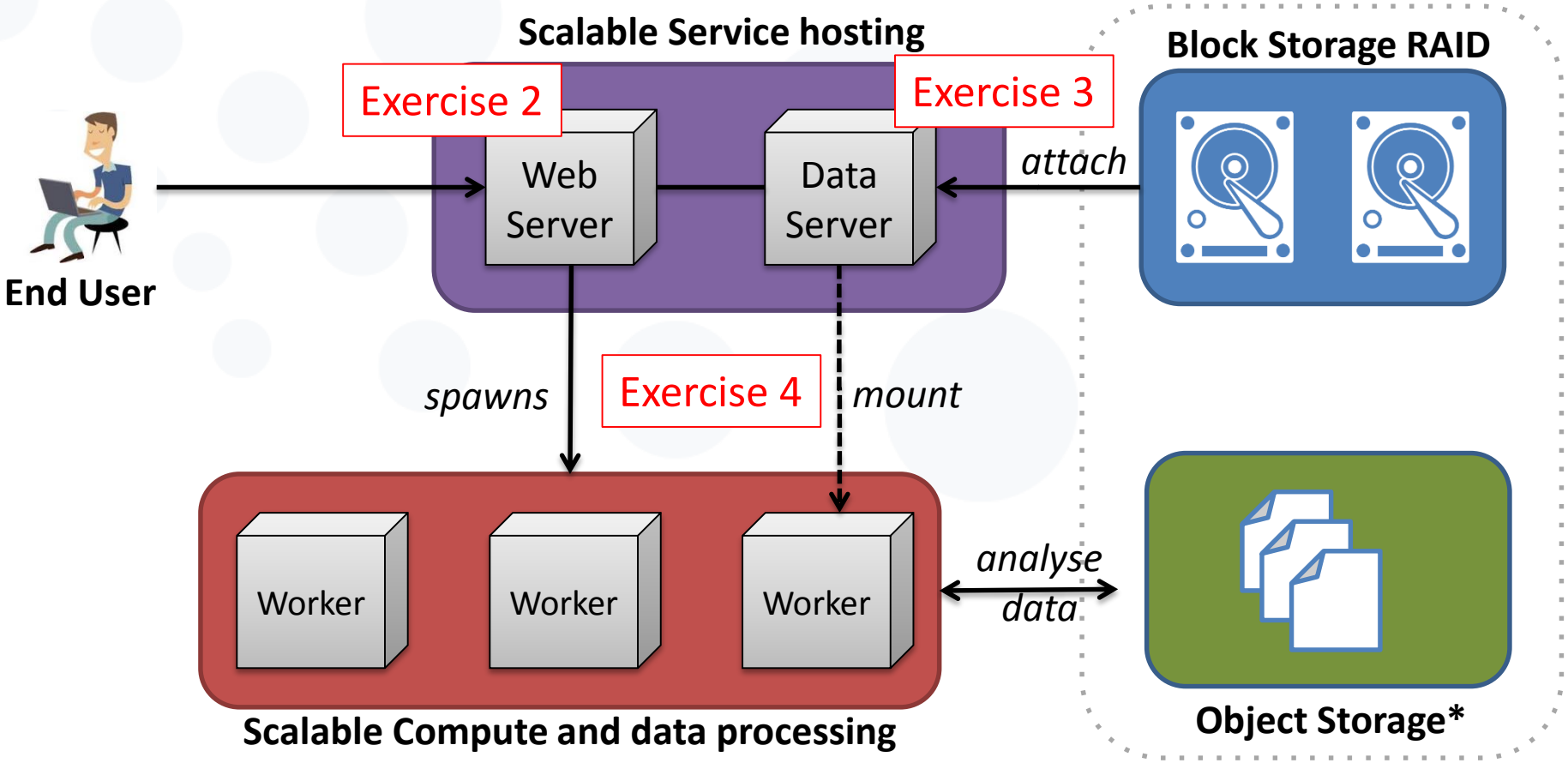


Typical usage models

- **Compute and data intensive workloads**
  - Batch and interactive (e.g. iPython-Jupiter) with scalable and customized environments
- **Service Hosting**
  - Long-running services (e.g. web server, database, application server, Galaxy server)
- **Datasets repository**
  - Store and manage large datasets (in a storage volume)
- **Disposable and testing environments**
  - Host training environments, test applications

Combined models

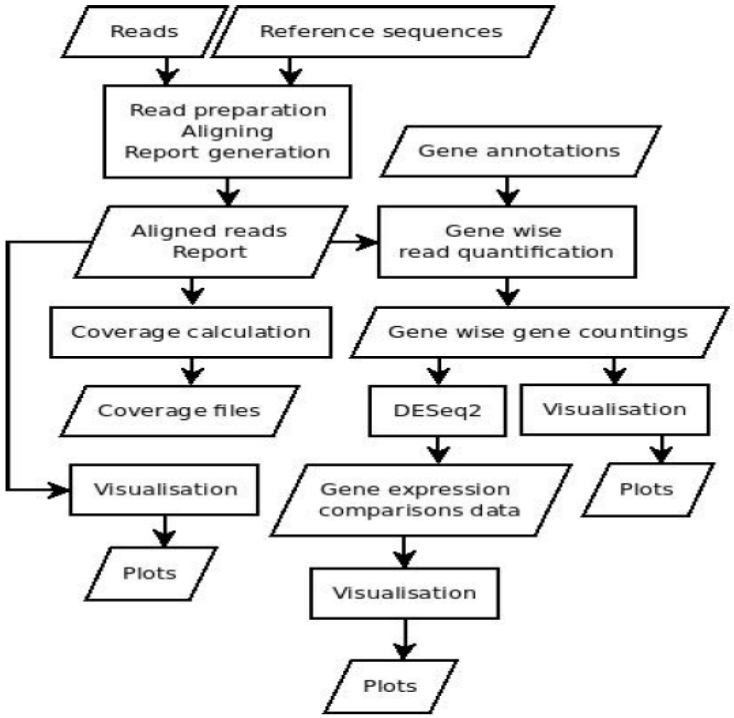
Combine usage models in a single application



* Object storage (CDMI or other) is not available on every site

Example: READemption

Pipeline for the computational evaluation of RNA-Seq. data

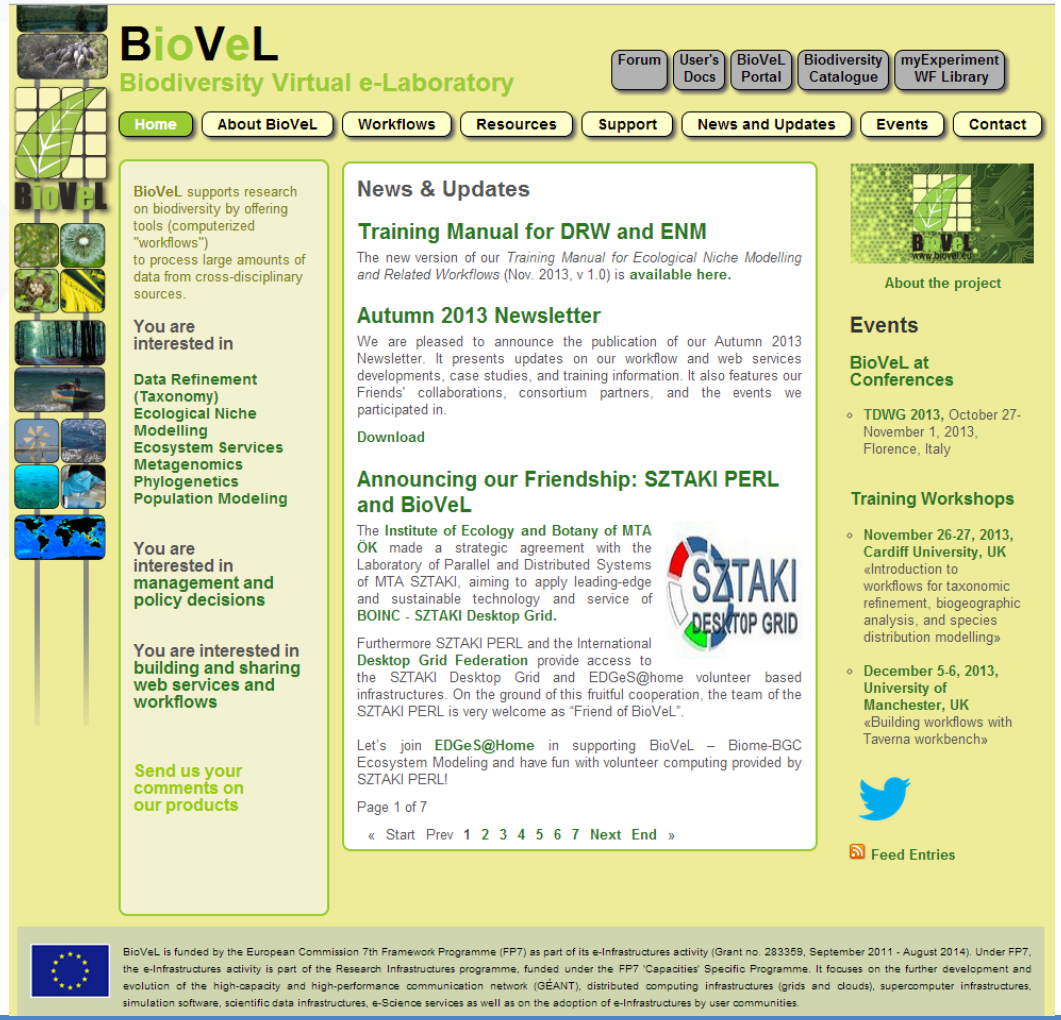


Source: Konrad U. Förstner

Usage Model	• **Heavy computation** • **Large Memory**
Scientific Disciplines	• **Bioinformatics**
Deployment in the FedCloud	• **VMs with 24 cores, 128 GB of RAM** • **Block storage up to 3 TB**

BioVEL

- Workflows:
  - Data Refinement Workflow
  - Ecological Niche Modeling Workflow
  - Ecosystem Services
  - Metagenomics
  - Phylogenetics
  - Population Modeling Workflow



Cloud BioLinux

- Publicly accessible VM
- Platform for developing bioinformatics infrastructures on the cloud
- Quick provision of on-demand infrastructures for HPC in bioinformatics
- Pre-configured tools and GUI
- Tested on Amazon EC2, Eucalyptus, Okeanos and Virtual box

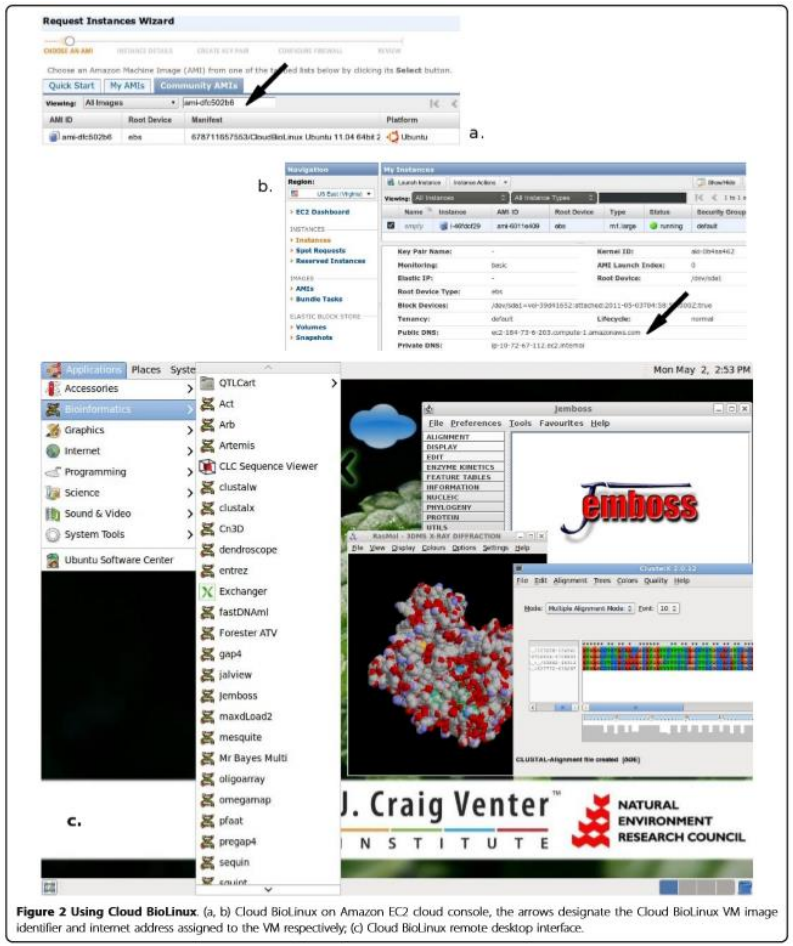
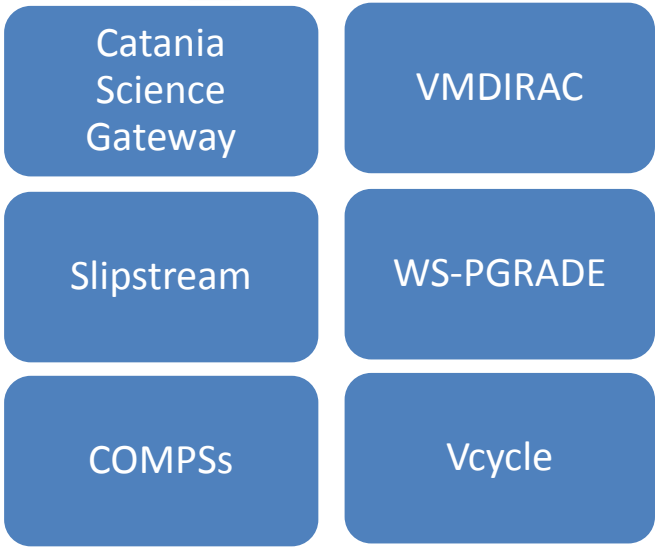


Figure 2 Using Cloud BioLinux. (a, b) Cloud BioLinux on Amazon EC2 cloud console, the arrows designate the Cloud BioLinux VM image identifier and internet address assigned to the VM respectively; (c) Cloud BioLinux remote desktop interface.

High Level Tools (PaaS & SaaS)

- Today we focus on IaaS with OCCI (rOCCI cmd. line), but…
- Higher level tools exist on top of this
  - Alternatives to OCCI and Nova– but usually much more!
  - External contributions – EGI is open for additional tools! Suggest/integrate!

Currently:



Others are emerging:

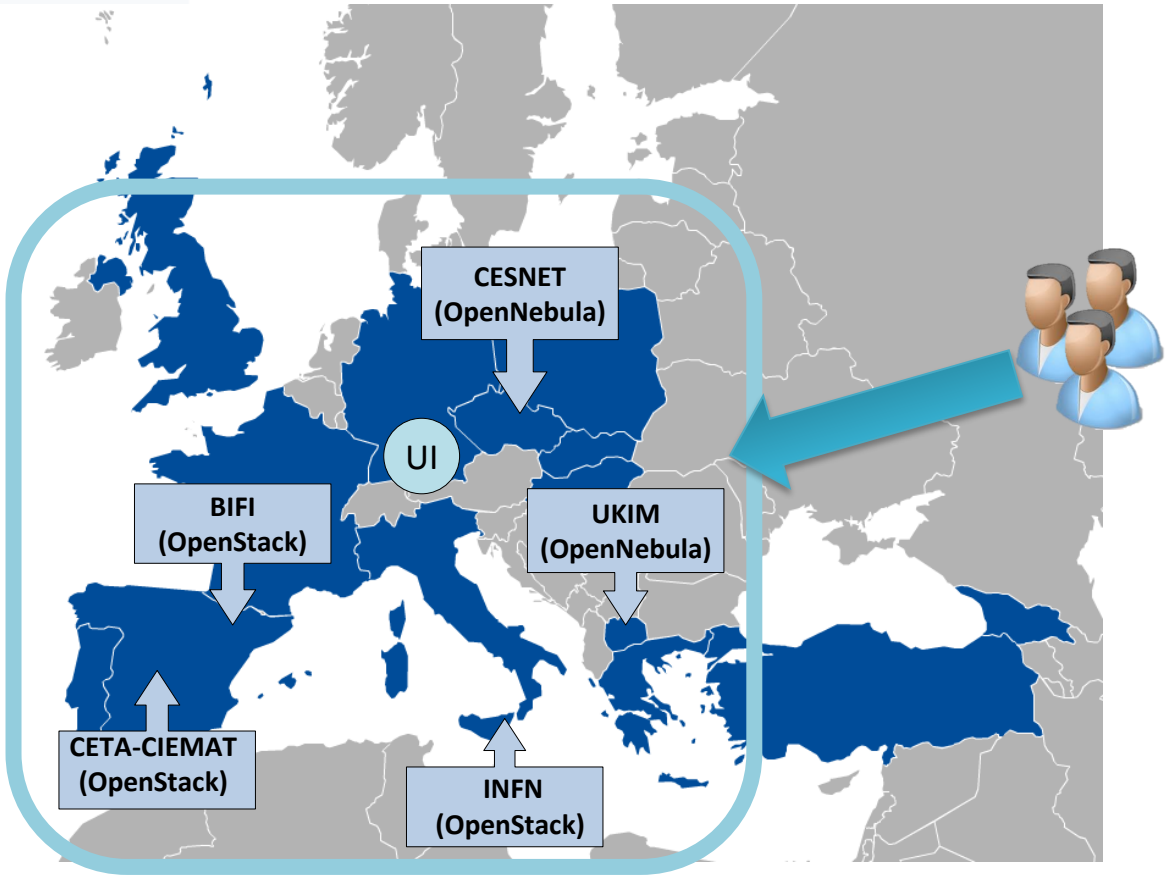
- Occopus orchestrator
- Karamel platform
- D4Science environment
- …

https://wiki.egi.eu/wiki/HOWTO10.

Training infrastructure

training.egi.eu Virtual Organisation

Site	Available capacity in the VO
CESNET (CZ)	64 vCPUs 110 GB of RAM 1 TB of persistent storage
BIFI (ES)	50 vCPUs 50 GB of RAM 50 storage volumes 50 public IP addresses
UKIM (MK)	48 vCPUs 48 GB of RAM 48 public IP addresses
CETA-CIEMAT (ES)	20 vCPUs 40 GB of RAM 5.4 TB storage 10 public IP addresses
INFN (IT)	20 vCPUs 50 GB of RAM 1 TB storage 10 public IP addresses



- Trainers join VO with X.509 personal certificates → Generate own proxy for access
- Trainees get proxies from trainers. **Your proxy is valid for 24 hours**
  - You will need personal certificate from a recognised CA for the long-term – More later!

Accessing the training VO

- Ubuntu 14.04 with rOCCI client
- Configured by trainers

- 1 account / trainee
- 1 proxy / account

Login with SSH

UI

OCCI commands

Get IDs of
- Cloud endpoint
- VA image
- Resource template

Access
(e.g. SSH, Web)

VM Marketplace
(AppDB)

VM

Block storage

Training.egi.eu VO

http://appdb.egi.eu → Cloud Marketplace

OCCI and rOCCI

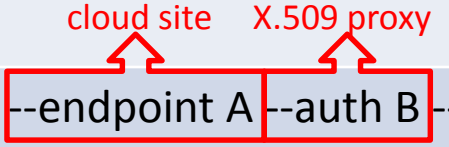
- **OCCI** (Open Cloud Computing Interface, OGF, 2011)
  - For VM Management (compute and storage)
  - Text-based protocol and API focusing on cloud interoperability

- **rOCCI** (OCCI command-line client; r for Ruby)
  - To be used today
  - Interacts with the OCCI servers deployed on cloud sites
  - Supports EGI AAI (X.509 certificates + VOMS)
  - Available with installer, as VM image, as Docker container or source

- **jOCCI**: Java API for OCCI
  - Further info: https://wiki.egi.eu/wiki/Federated_Cloud_APIs_and_SDKs

Main commands to be used during Exercises

Command	Explanation
voms-proxy-info	Check the lifetime of your proxy
ssh-keygen	Generate key-pairs for password-less SSH
occi --endpoint A --auth B --action C –resource D	Perform action C on resource D of cloud site A authenticating as B
--action list	
--action create	
--action describe	
--resource compute	
--resource storage	



rOCCI quick reference guide: https://gist.github.com/arax/4de4a41fb0fa67719856

Log into the UI

- Log into the User Interface
  - SSH to 147.228.242.12
  - Username: xxxx
  - Password: xxxx
- Check your proxy file

```
~$ echo $X509_USER_PROXY
```

- Check the lifetime of your credential

```
~$ voms-proxy-info -all
```

Get ready to access your VMs with SSH

- VMs are (normally) accessible through SSH
  - But password logins are disabled
  - Instead use key pairs
- Create a ssh key to access:

```
~$ ssh-keygen
```

(defaults are ok, can be left without password for the tutorial)

Excercise 1
Running Chipster in EGI Federated Cloud

Chipster:
http://chipster.csc.fi

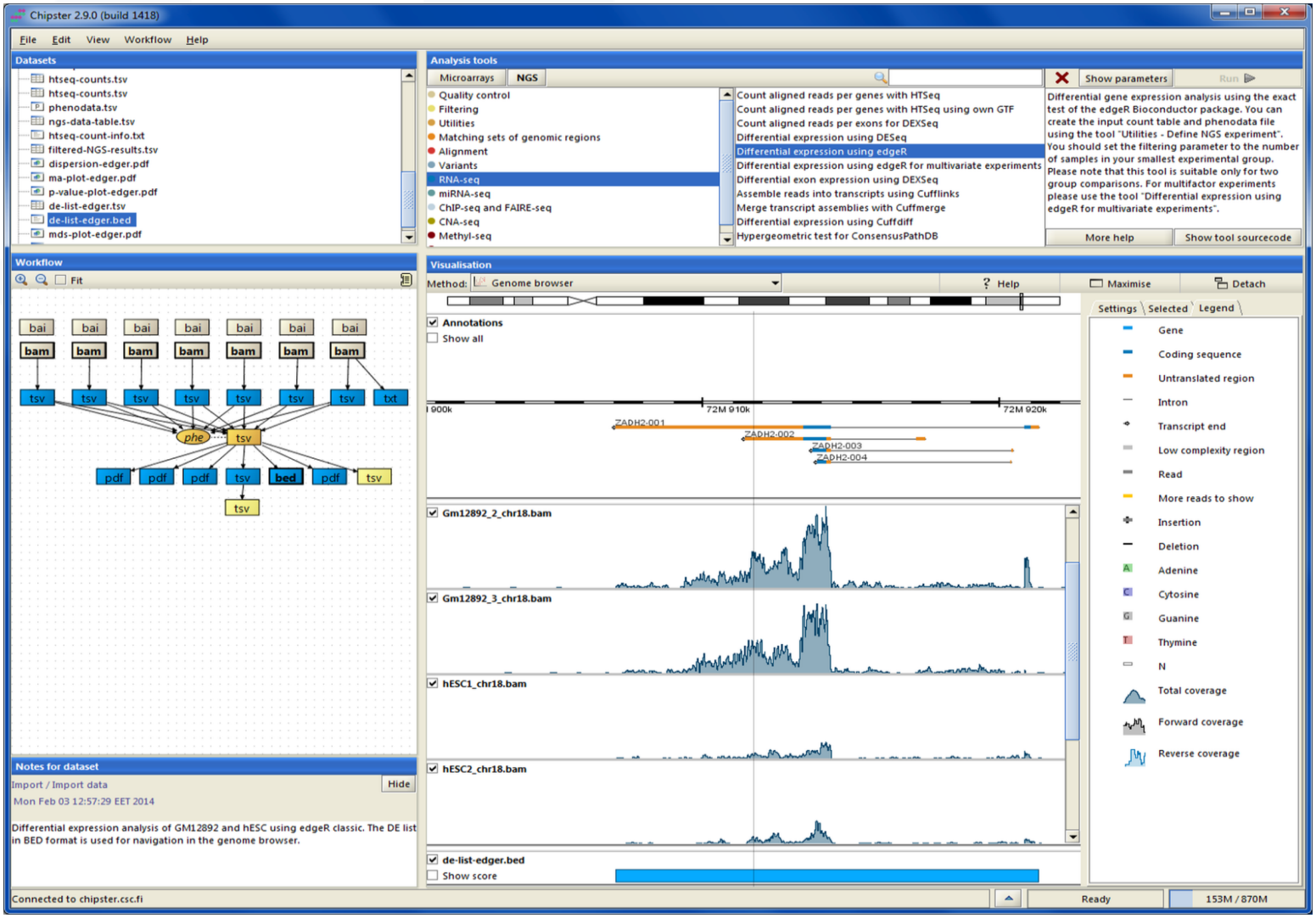
- **Open source platform for data analysis**
- **Provides an easy access to over 340 analysis tools**
  - No programming or command line experience required
- **What can I do with Chipster?**
  - analyze and integrate high-throughput data
  - visualize data efficiently
  - share analysis sessions
  - save and share automatic workflows

Analysis tools

- **140 NGS tools for**
  - RNA-seq
  - miRNA-seq
  - exome/genome-seq
  - ChIP-seq
  - FAIRE-seq
  - MeDIP-seq
  - CNA-seq
  - Metagenomics (16S rRNA)
- **60 tools for sequence analysis**
  - BLAST, EMBOSS, MAFFT, Phylip

- **140 microarray tools for**
  - gene expression
  - miRNA expression
  - protein expression
  - aCGH
  - SNP
  - integration of different data

Chipster client



Chipster

- **Chipster id free, open source software**
- **CSC hosts a Chipster server for researchers working in Finland**
- **If you are not working in Finland you must purchase account to CSC or use some other Chipster server**

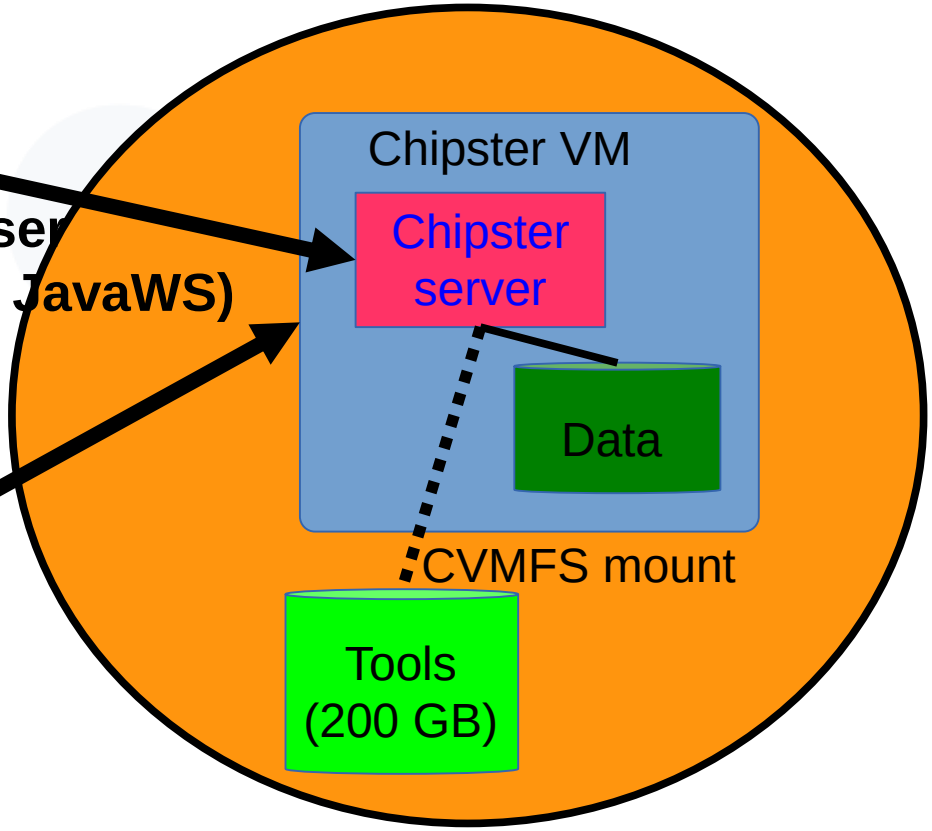
Chipster in EGI Federated Cloud

Users

Web browser
(Chipster client+ JavaWS)

OCCI
SSH

Local Chipster manager



Tools needed:
- Certificate
- VO membership
- rOCCI
- Mac OSX or Linux

Launching Chipster in EGI Federated cloud

1. **Create a contextualization script** that contains commands to create the required directories and CVMFS linking (about 50 lines)

2. **Create a data volume**

3. **Select VM-flavor and operating system template and launch the virtual machine**

4. **Set a public IP address**

5. **Connect to the new VM and restart chipster server**

Launching Chipster in EGI Federated cloud

1. **... or use FedCloud_chipster_manager**

   `./FedCloud_chipster_manager -launch -key` *`your_cloud_key`*

2. **Tasks available in FedCloud_chipster_manager**
   1. **launch** a chipster server
   2. **delete** a chipster server
   3. **list** chipster servers in current VO
   4. check **status** of chipster servers in current VO
   5. **restart** a Chipster server
   6. add chipster **user accounts** to the server

Using Chipster in EGI Federated Cloud

- **When the server is running, end users can access the server using port 8081**
  - **https://[ip.address.of.the.VM]:/8081**

- **The manager of the server can open a terminal connetion to the server:**
  - ssh -i keyfile ubuntu@[ip.address.of.the.VM]

- **Instructions for managing your Chipster server can be found from the Chipster technical manual:**
  - https://github.com/chipster/chipster/wiki/TechnicalManual

BREAK

www.egi.eu

EGI-Engage is co-funded by the Horizon 2020 Framework Programme
of the European Union under grant number 654142

EGI-Engage

Exercise 2 & 3: Jupyter Notebook

Jupyter Notebook

- Open source, interactive data science and scientific computing across over 40 programming languages.
- Notebooks can be shared with others using email, Dropbox, GitHub
- Interactive widgets

DATA CARPENTRY
MAKING DATA SCIENCE MORE EFFICIENT

software carpentry

- Favorite tool for the Software and Data Carpentry workshops

Exercise 2 and 3

Managing VMs and block storage:

1. Start a Jupyter Notebook on an EGI Cloud site

2. Use persistent storage for Jupyter files

EGI-Engage

Exercise 2: Run a Jupyter Notebook in the EGI Federated Cloud

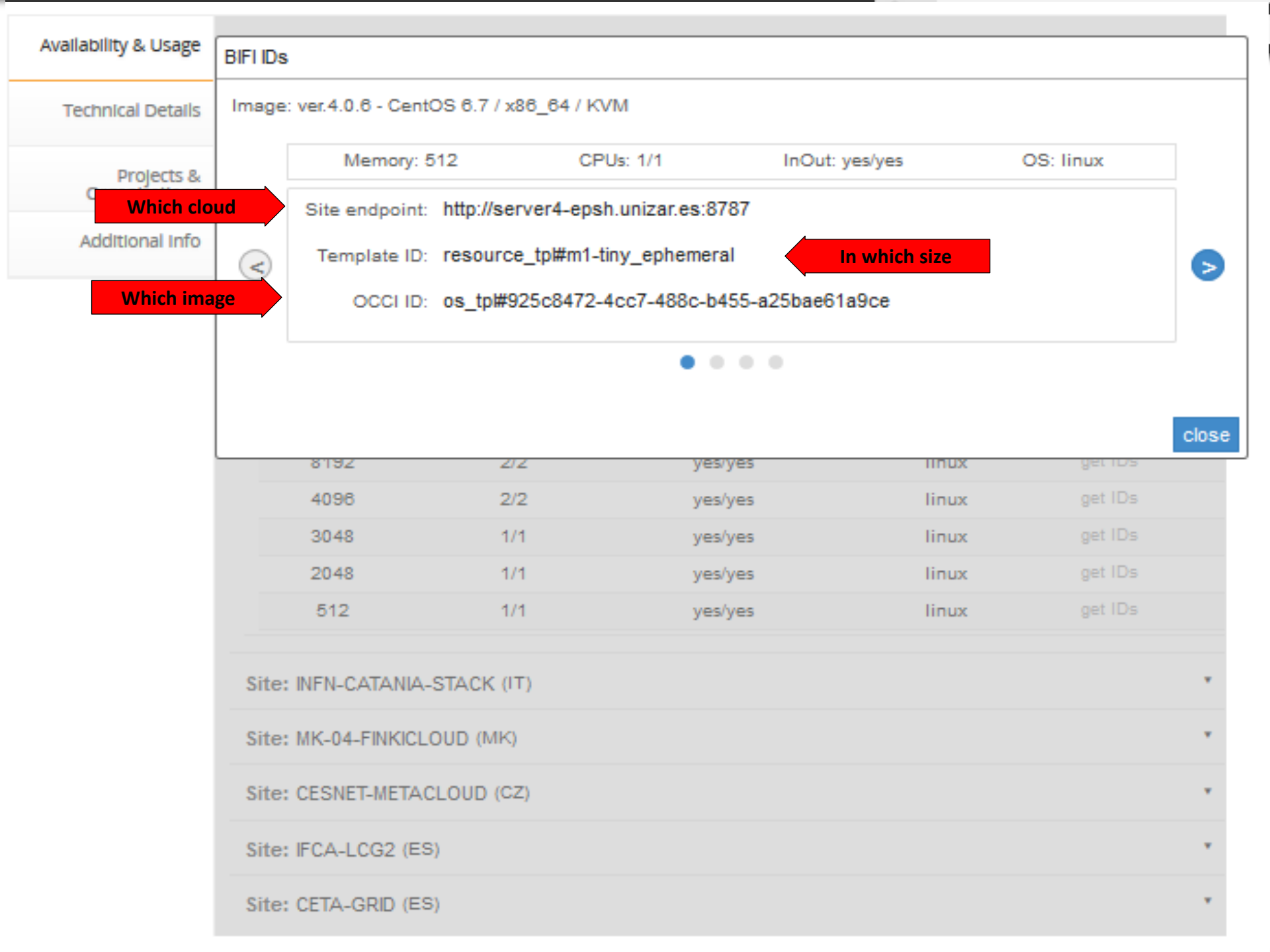
Exercise 1: Jupyter Notebook setup

What you have to do:

1. Browse AppDB, find **3 IDs** (visual lookup):
    1. ID of the cloud site you want to use
    2. ID of the Jupyter Notebook VM image for that site
    3. ID of the resource template the VM should use (smallest!)
2. Create VM instance (OCCI call)
3. Access wiki from a web browser

Browsing AppDB
DEMO

- Go to AppDB:
  - http://appdb.egi.eu
  - Cloud Mp → Virtual Organizations → training.egi.eu
- Choose Jupiter Notebook VA and a specific site
  - See request on next slide!
- VAs and SAs in this VO:
  - Baseline OS appliances
    - Minimal OS images
    - Centos6, Ubuntu 12.04, Ubuntu 14.04
  - Specific appliances
    - FedCloud tools: Ubuntu 14.04 with FedCloud clients ready to use
    - MoinMoin wiki: Ubuntu 14.04 image with MoinMoin installed and configured to run on startup
    - Jupyter Notebook: Centos6 image with Jupyter Notebook installed
  - Software appliances
    - Use contextualization to deliver the functionality


Availability & Usage
Technical Details
Projects &
Additional Info
BIFI IDs
Image: ver.4.0.6 - CentOS 6.7 / x86_64 / KVM
Memory: 512
CPUs: 1/1
InOut: yes/yes
OS: linux
Which cloud
Site endpoint: http://server4-epsh.unizar.es:8787
Template ID: resource_tpl#m1-tiny_ephemeral
In which size
Which image
OCCI ID: os_tpl#925c8472-4cc7-488c-b455-a25bae61a9ce
close
4096 2/2 yes/yes linux get IDs
3048 1/1 yes/yes linux get IDs
2048 1/1 yes/yes linux get IDs
512 1/1 yes/yes linux get IDs
Site: INFN-CATANIA-STACK (IT)
Site: MK-04-FINKICLOUD (MK)
Site: CESNET-METACLOUD (CZ)
Site: IFCA-LCG2 (ES)
Site: CETA-GRID (ES)

REQUEST

- Instantiate VMs based on the smallest resource templates during the whole tutorial
  - I.e. Use the following **Template IDs**:

Site	Template name	Template ID
CESNET	Small	http://schema.fedcloud.egi.eu/occi/infrastructure/resource_tpl#small
BIFI	Tiny	resource_tpl#m1-tiny-ephemeral **MORE COMPLEX NETWORKING!**
INFN-CATANIA-STACK	Small	resource_tpl#m1-small

Create your first compute appliance

Use MoinMoinWiki VA values from AppDB!

```
~$ ENDPOINT=<Copy here Site Endpoint information from AppDB>

~$ RESOURCE_TPL=<copy here the Template ID from AppDB>

~$ OS_TPL=<copy here the OCCI ID from AppDB>

~$ occi --endpoint $ENDPOINT \
        --auth x509 --voms --user-cred $X509_USER_PROXY \
        --action create --resource compute \
        --mixin $RESOURCE_TPL --mixin $OS_TPL \
        --attribute occi.core.title="notebook$(date +%s)" \
        --context public_key="file:///$HOME/.ssh/id_rsa.pub"

~$ COMPUTE_ID=...
```

Save the ID in an Env. variable

List and describe your VM instances

```
~$ occi --endpoint $ENDPOINT \
        --auth x509 --voms --user-cred $X509_USER_PROXY \
        --action list --resource compute
```

```
~$ occi --endpoint $ENDPOINT \
        --auth x509 --voms --user-cred $X509_USER_PROXY \
        --action describe --resource $COMPUTE_ID
```

This returns lot of info, including the IP Address of your VM!

`occi.networkinterface.address = …`

→ It's not so simple ☹ See next slide!

Accessing the appliance

- If the VM does not have a public IP (on BIFI endpoint):

```
~$ occi --endpoint $ENDPOINT \
        --auth x509 --voms --user-cred $X509_USER_PROXY \
        --action link --resource $COMPUTE_ID \
        --link /network/public
```

- Obtain the IP address from the output of the describe command.

Logging into the appliance

- ssh with centos user:

```
~$ ssh centos@<your vm ip>
```

- Once logged in, check the size of the image:

```
~wiki $ cat /proc/cpuinfo

~wiki $ cat /proc/meminfo
```

Start the service

- After connecting to the newly launched VM, start the Jupyter notebook as follows:

```
~$ jupyter notebook
```

- Jupyter start a web-server (by default listening to port 8888)
- Go to your web-browser and type:
  - https://[public ip]:8888

Transfer files

- We can transfer input/data files, as well as notebooks from any given location to the current VM.
- In our case, let's take some sample files using wget as follows:

"Real-world" notebook

```
~$ wget http://grid.ct.infn.it/cron_files/ELIXIR_WS/GeneExpressionHeatmap.ipynb
```

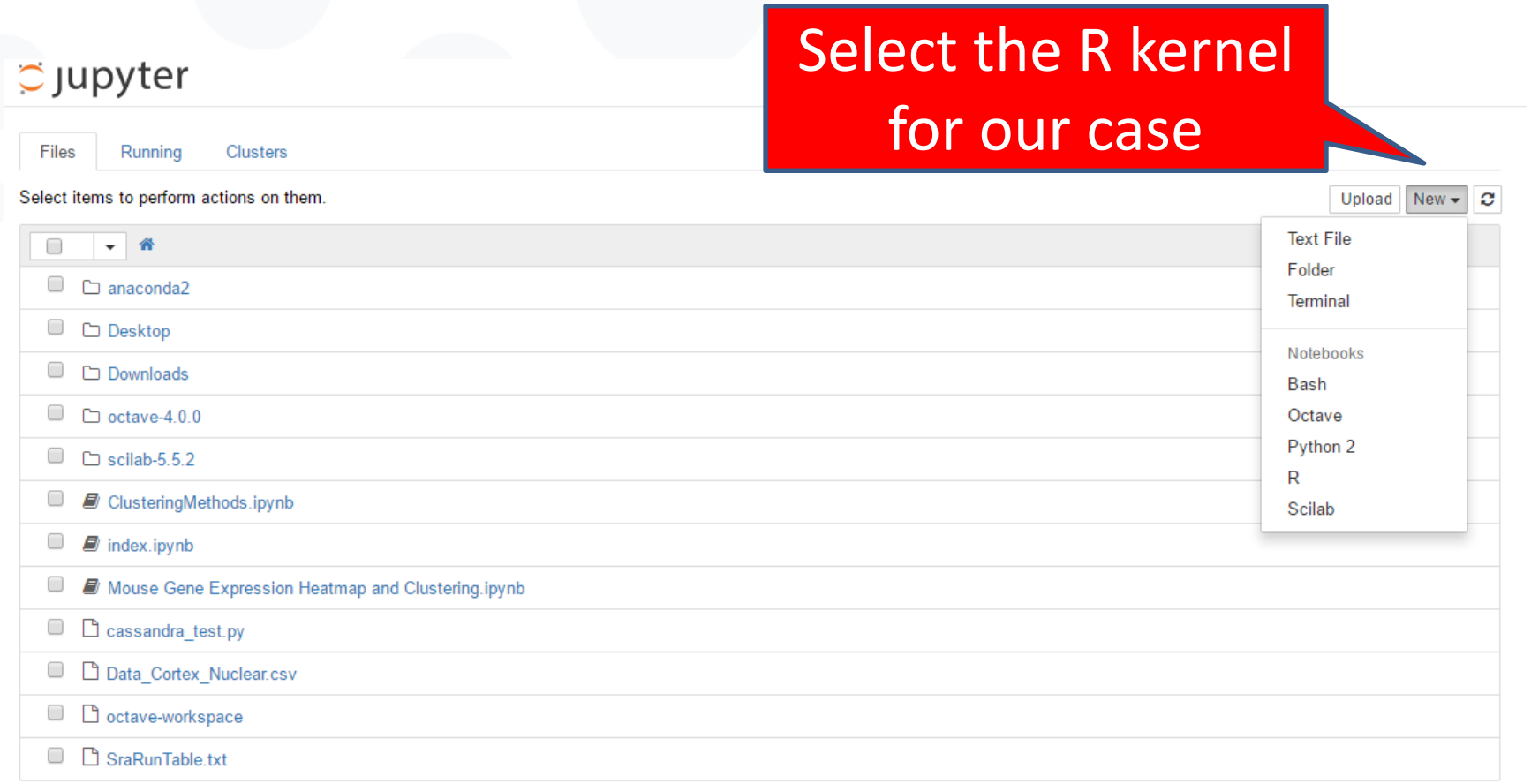
Corresponding dataset

```
~$ wget http://grid.ct.infn.it/cron_files/ELIXIR_WS/Data_Cortex_Nuclear.csv
```

A dataset for our exercise now

```
~$ wget http://grid.ct.infn.it/cron_files/ELIXIR_WS/SraRunTable.txt
```

Jupyter's main page



You can also have a terminal case

- Useful for basic CLI training.

We can show standard downsteam analysis

jupyter Using clouds and VMs in bioinformatics training Last Checkpoint: a few seconds ago (autosaved)

File Edit View Insert Cell Kernel Help

Code CellToolbar

In [1]:
```
mydata <- read.csv("SraRunTable.txt", head=TRUE, sep="\t")
```

In [2]:
```
summary(mydata)
```

Out[2]:
```
      BioSample_s    Experiment_s  InsertSize_l    LibraryLayout_s
SAMN00205564: 2   SRX040608: 1    Min.   :   0.0   PAIRED: 2
SAMN00205573: 2   SRX040663: 1    1st Qu.:   0.0   SINGLE:35
SAMN00205533: 1   SRX040664: 1    Median :   0.0
SAMN00205558: 1   SRX040665: 1    Mean   : 150.4
SAMN00205559: 1   SRX040666: 1    3rd Qu.:   0.0
SAMN00205560: 1   SRX040667: 1    Max.   :2834.0
(Other)     :29   (Other)  :31
       Library_Name_s      LoadDate_s    MBases_l         MBytes_l
<not provided>: 1     2012-07-25: 6   Min.   :  59.0   Min.   : 48.0
CZB152        : 1     2014-05-29:31   1st Qu.: 153.0   1st Qu.:112.0
CZB154        : 1                     Median : 281.0   Median :193.0
CZB199        : 1                     Mean   : 444.7   Mean   :280.8
REL10979      : 1                     3rd Qu.: 633.0   3rd Qu.:444.0
REL10988      : 1                     Max.   :1695.0   Max.   :679.0
(Other)       :31
   ReleaseDate_s       Run_s       SRA_Sample_s  Sample_Name_s Assay_Type_s
2011-03-25:31    SRR097977: 1   SRS167172: 2    ZDB172  : 2    WGS:37
2012-07-26: 6    SRR098026: 1   SRS167181: 2    ZDB30   : 2
                 SRR098027: 1   SRS167141: 1    CZB152  : 1
                 SRR098028: 1   SRS167166: 1    CZB154  : 1
                 SRR098029: 1   SRS167167: 1    CZB199  : 1
                 SRR098030: 1   SRS167168: 1    REL10979: 1
                 (Other)  :31   (Other)  :29    (Other) :29
```

Each R command is executed within the VM

Results are shown on page

Example case

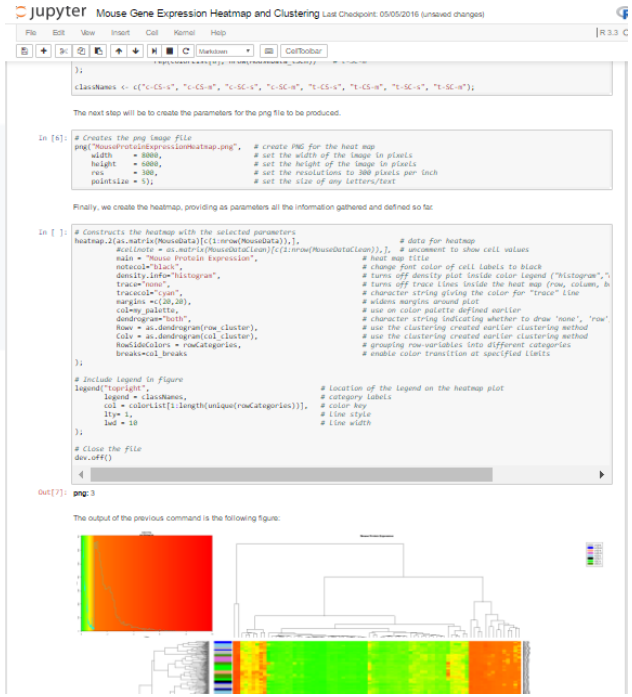
- Let's run the following commands in the newly created (*adapted from the Data Carpentry genomics lesson*)

```
~$ sradata <- read.csv("SraRunTable.txt", head=TRUE,
                                                  sep="\t")
~$ summary(sradata)
~$ install.packages("dplyr",
                    repos='http://cran.us.r-project.org')
~$ library("dplyr")
~$ select(sradata, LibraryLayout_s, LoadDate_s, MBases_l,
                                                Sample_Name_s)
~$ filter(sradata, LibraryLayout_s == "PAIRED")
~$ sradata %>%
    select(LibraryLayout_s, LoadDate_s, MBases_l,
                                                Sample_Name_s) %>%
    filter(LibraryLayout_s == "PAIRED")
```

Load data in the notebook

Run entire preset notebooks

- One of the key advantages is to allow the re-use of defined notebooks
- Open the “Mouse Gene Expression Heatmap and Clustering” notebook
- It’s an entire process, with documentation, that can allow specific tasks (the creation of a Gene Expression heatmap in this case)



EGI-Engage

Exercise 3:
Jupyter with persistent storage

Making Jupyter files persistent

- When a VM is deleted all its disks are also deleted
  - If you need persistency for your data you must use a storage volume
- Let's try it with our Jupyter Notebook:
  1. Create a volume
  2. Attach volume to our Jupyter VM
  3. Create FS in the volume and copy the Jupyter files
  4. Detach volume and delete VM
  5. Create new VM with the created volume attached
  6. Mount the volume and check the Jupyter files are still there

Create the volume and describe it

- Create a volume

```
~$ occi --endpoint $ENDPOINT \
        --auth x509 --voms --user-cred $X509_USER_PROXY \
        --action create --resource storage \
        --attribute occi.storage.size="num(1)" \
        --attribute occi.core.title="notebookdata_$(date
+%s)"
~$ STORAGE_ID=...
```

Save the ID in an Env. variable

- Describe it

```
~$ occi --endpoint $ENDPOINT \
        --auth x509 --voms --user-cred $X509_USER_PROXY \
        --action describe --resource $STORAGE_ID
```

Attach to VM

```
~$ occi --endpoint $ENDPOINT \
        --auth x509 --voms --user-cred $X509_USER_PROXY \
        --action link --resource $COMPUTE_ID \
        --link $STORAGE_ID
```

See attach information

```
~$ occi --endpoint $ENDPOINT \
        --auth x509 --voms --user-cred $X509_USER_PROXY \
        --action describe --resource $COMPUTE_ID

[…]

Links:

    [[ http://schemas.ogf.org/occi/infrastructure#storagelink ]]
    >> location: /storage/link/c17e204e-c96f-40ff-aebe-
671351254a5e_1e0162cb-2805-4fe7-8c4e-997a5ddf02ff
    occi.core.source = /compute/c17e204e-c96f-40ff-aebe-671351254a5e
    occi.core.target = /storage/1e0162cb-2805-4fe7-8c4e-997a5ddf02ff
    occi.core.id = /storage/link/c17e204e-c96f-40ff-aebe-
671351254a5e_1e0162cb-2805-4fe7-8c4e-997a5ddf02ff
    occi.storagelink.deviceid = /dev/vdb

[…]

~$ LINK_ID= =<copy here Link ID>
```

LINK_ID

We will need this at the VM to manage the volume

Move Jupyter files to new volume

```
~$ ssh centos@<your jupyter notebook ip>

~$ sudo mkfs.ext3 /dev/vdb

~$ sudo mount /dev/vdb /mnt

~$ sudo cp GeneExpressionHeatmap.ipynb
Data_Cortex_Nuclear.csv SraRunTable.txt /mnt

~$ sudo ls -la /mnt

~$
```

Clean up and stop the VM

- Umount the volume

```
~$ sudo umount /mnt
```

- Detach the volume:

```
~$ occi --endpoint $ENDPOINT \
        --auth x509 --voms --user-cred $X509_USER_PROXY \
        --action delete --resource $LINK_ID
```

- Delete VM:

```
~$ occi --endpoint $ENDPOINT \
        --auth x509 --voms --user-cred $X509_USER_PROXY \
        --action delete --resource $COMPUTE_ID
```

Create a new notebook with the volume

```
~$ occi --endpoint $ENDPOINT \
        --auth x509 --voms --user-cred $X509_USER_PROXY \
        --action create --resource compute \
        --mixin $RESOURCE_TPL --mixin $OS_TPL \
        --attribute occi.core.title="notebook$(date +%s)" \
        --link $STORAGE_ID \
        --context public_key="file:///$HOME/.ssh/id_rsa.pub"

~$ COMPUTE_ID2=...
```

Save the ID in an Env. variable

Use the volume

- Login into the VM and mount the volume at /org

```
~$ ssh centos@<your notebook ip>

~$ sudo mount /dev/vdc /mnt

~$ ls -la /mnt
```

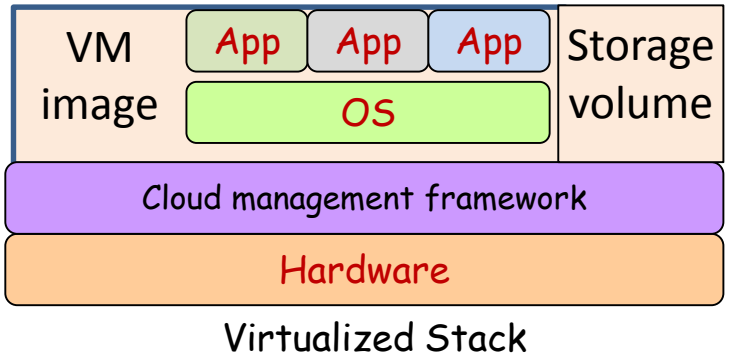
- The Jupyter Notebook files are available in the new VM!

Once done, delete your instances

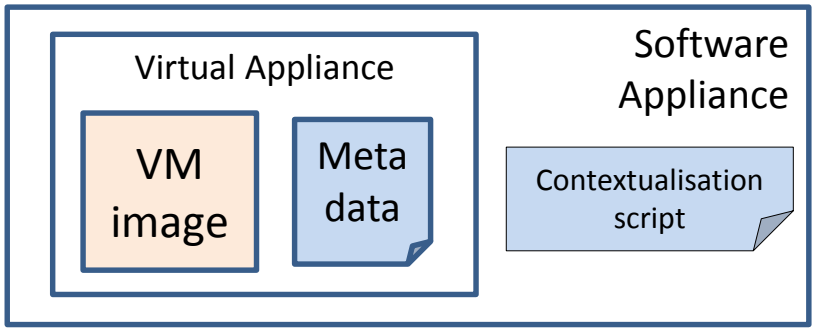
```
~$ occi --endpoint $ENDPOINT \
        --auth x509 --voms --user-cred $X509_USER_PROXY \
        --action delete --resource $COMPUTE_ID2
```

```
~$ occi --endpoint $ENDPOINT \
        --auth x509 --voms --user-cred $X509_USER_PROXY \
        --action delete --resource $STORAGE_ID
```

Contextualisation



Virtualized Stack



Contextualization

- What?
  - Contextualization is the process of installing, configuring and preparing software upon boot time on a pre-defined virtual machine image
  - e.g. hostname, IP address, ssh keys, …
- Why?
  - Configuration not known until instantiation (e.g. data location)
  - Private Information (e.g. host certs)
  - Software that changes frequently or under development
  - Not practical to create a new VM image for every possible configuration

Contextualization in FedCloud

- Contextualization requires passing some data to the VMs on instantiation (the context)
- OCCI extensions specify how to pass context to the VM
  - BUT not how the data will be available!
- Each RP can have different mechanisms
  - metadata server at a known location
  - iso filesystem attached to the VM
  - file injection into the VM image
  - …
- cloud-init is the preferred tool to abstract this

cloud-init

- cloud-init abstracts the different ways of providing context to the VM and defines a format for the data
- cloud-init is able to:
  - configure network, users, **ssh keys**, filesystems,
  - **install packages,**
  - execute arbitrary commands,
  - **execute user provided scripts,**
  - invoke puppet or chef for configuration
  - …
- And can be easily extended!

cloud-init support

- cloud-init is the de-facto standard for contextualization of VMs
- Supports:
  - Most commercial providers (Amazon EC2, Azure, Rackspace,…)
  - Cloud management frameworks in fedcloud:
    - OpenStack
    - OpenNebula (use version ≥ 0.7.5)
    - Synnefo
- Packages available for most Linux distributions: ubuntu/debian, SL5/SL6 (in EPEL), SUSE, …

Use with rOCCI CLI

- Use --context option to specify
  - user_data
  - public_key

EXAMPLE – NO NEED TO EXECUTE

```
~$ occi --endpoint $ENDPOINT \
        --auth x509 --voms --user-cred $X509_USER_PROXY \
        --action create --resource compute \
        --mixin $RESOURCE_TPL --mixin $OS_TPL \
        --attribute occi.core.title="wiki$(date +%s)" \
        --context user_data="file:///$PWD/context"  \
        --context public_key="file:///$HOME/.ssh/id_rsa.pub"

~$ COMPUTE_ID=...
```

Meta data vs user data

Meta data

- Basic predefined information on the VM
  - VM Identifier
  - Hostname, IP
  - User Public Keys

User data

- User data is treated as opaque data: Passed to cloud-init.
- It is up to cloud-init to interpret it.

cloud-init uses both meta-data and user-data to contextualize the VMs

Public Key injection with cloud-init

- SSH with key pairs is so common that treated specially
  - Goes into the VM meta-data
  - By default, cloud-init will add it to the authorized keys of the default user (ubuntu for Ubuntu, centos for CentOS, see description of the VM in AppDB)
  - You can inject more keys to other users with the user-data but adding it here makes it independent to errors of the user-data → access the VM to debug issues

User data in cloud-init

- Some of the supported formats:
  - user script: execute it. (begins with #!) → Next slide
  - Cloud Config Data: cloud-config is the simplest way to accomplish some things via user-data. Using cloud-config syntax, the user can specify certain things in a human friendly format. (begins with #cloud-config)
  - include file: contains a list of urls, one per line. Each of the URLs will be read, and their content will be passed through this same set of rules. (begins with #include)
  - gzipped content → uncompress and use as it were not compressed.

Deploy an app. into you VM with contextualisation

EXAMPLE – NO NEED TO EXECUTE

Create a simple script, called script.sh

```
#!/bin/sh
echo "Hello World." > /root/context.txt
echo "The time is now $(date -R)!" >> /root/context.txt
```

Instantiate VM

```
~$ occi -e $ENDPOINT -n x509 -X -x $X509_USER_PROXY \
        -a create -r compute  \
        [...]
        --context user_data="file://$PWD/script.sh"
```

Check results

```
~$ ssh -i test.key ubuntu@155.210.71.129 "sudo cat
/root/context.txt"
Hello World
The time is now Sat, 20 Jun 2015 18:01:29 +0100
```

cloud-config

- cloud-config is cloud-init own configuration format
- Uses YAML (invalid syntax will make the contextualization fail!)
- Examples:
  - Create users and groups
  - apt-get upgrade should be run on first boot
  - Install packages
  - Run commands
  - …
- cloud-init documentation contains examples for all the supported options: http://cloudinit.readthedocs.org/

Sample cloud-config file

```
#cloud-config

users:
  - default
  - name: myuser
    sudo: ALL=(ALL) NOPASSWD:ALL
    lock-passwd: true
    ssh-import-id: myuser
    shell: /bin/bash
    ssh-authorized-keys:
      - <your key here>

package_upgrade: true

packages:
  - ca-policy-egi-core
  - occi-cli
  - voms-clients
```

Tells cloud-init this is a cloud-config file

Configure default user (e.g. ubuntu)

Create a user called “myuser” able to sudo and with a ssh key

Run apt-get upgrade (or yum upgrade)

Install some packages

EGI-Engage

Exercise 4:
Fractal application with contextualisation
(Home work)

Using Docker in the EGI Federated Cloud to start a Galaxy server

Docker intro slides repurposed from http://pointful.github.io/docker-intro

EGI-Engage

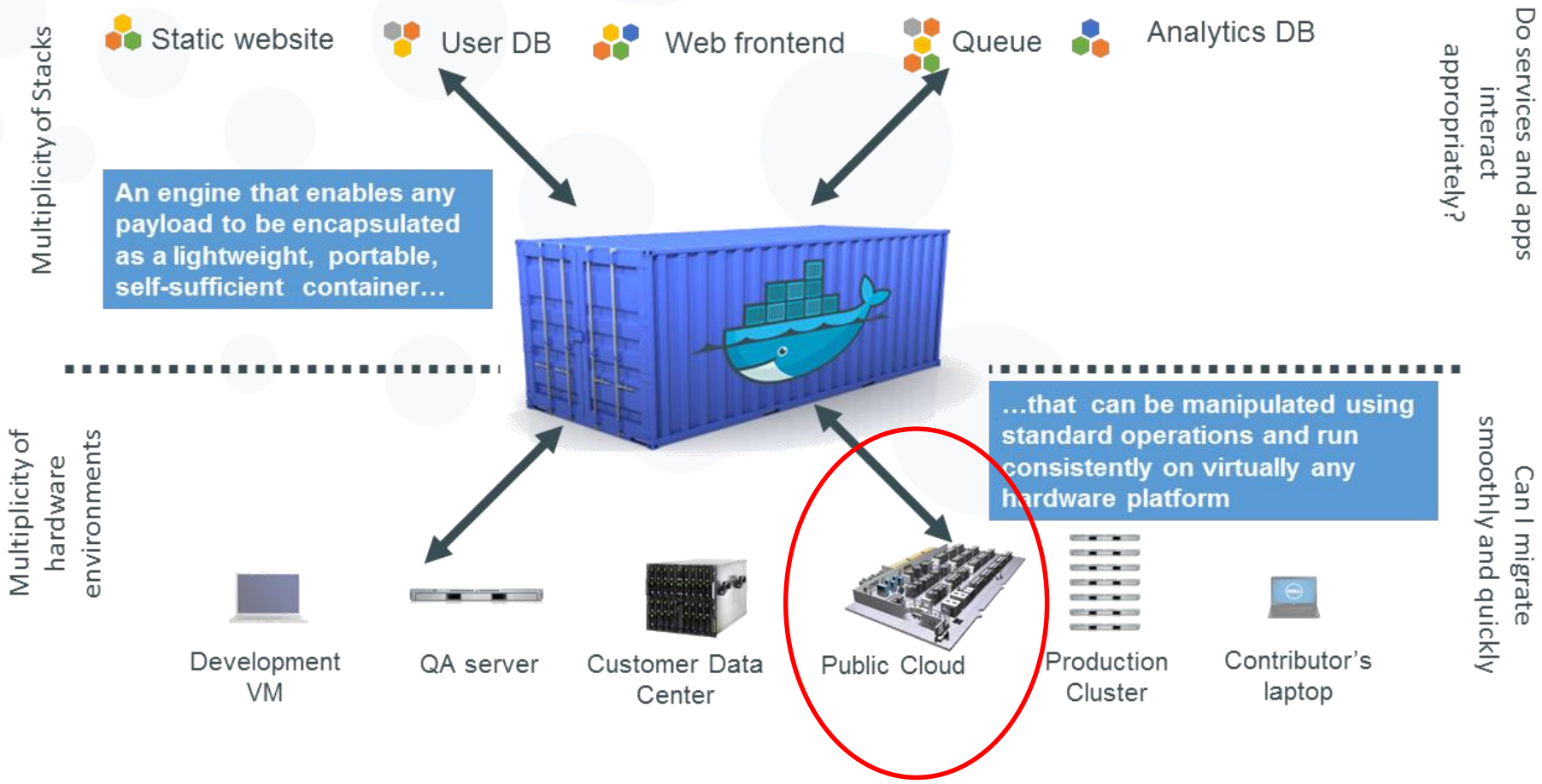
What is Docker ?
https://www.docker.com

- Containers-As-A-Service platform that allows building and running distributed applications anywhere
  - This guarantees that your application will always run regardless of the environment it is running on.

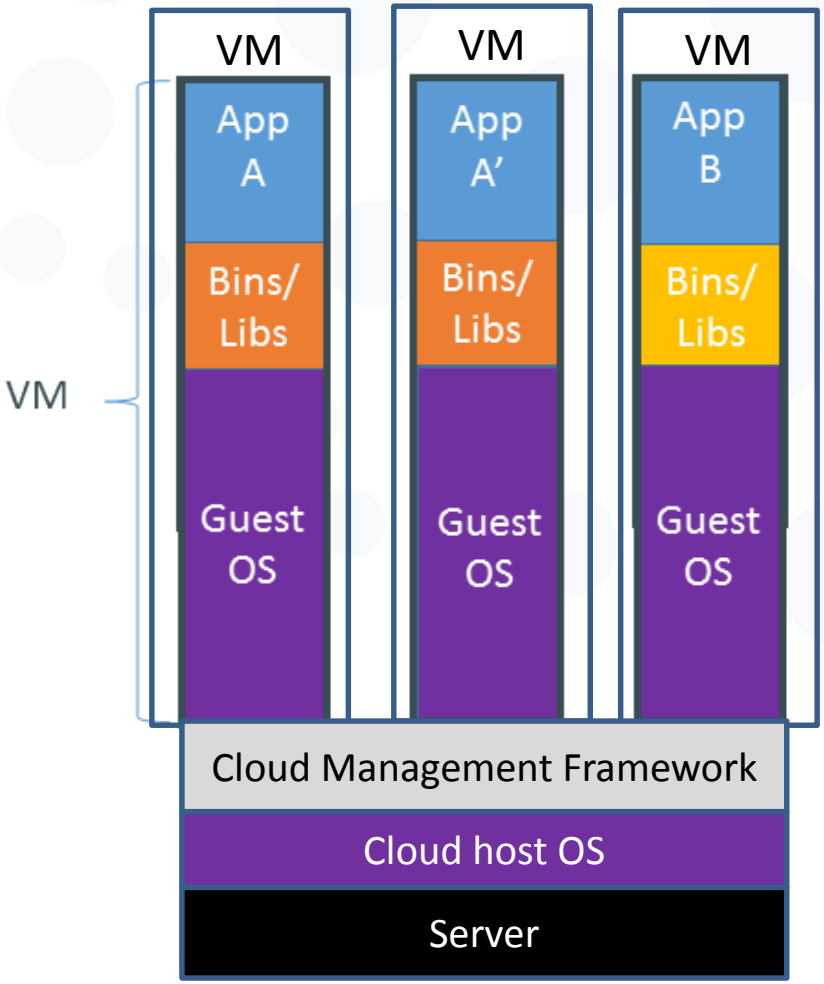
- When your application is in a Docker containers, you don’t have to worry about setting up and maintaining different environments of different tool for each languages.

Intermodal shipping container – solution for pre-cargo problem before 1960

Docker – Container system for code

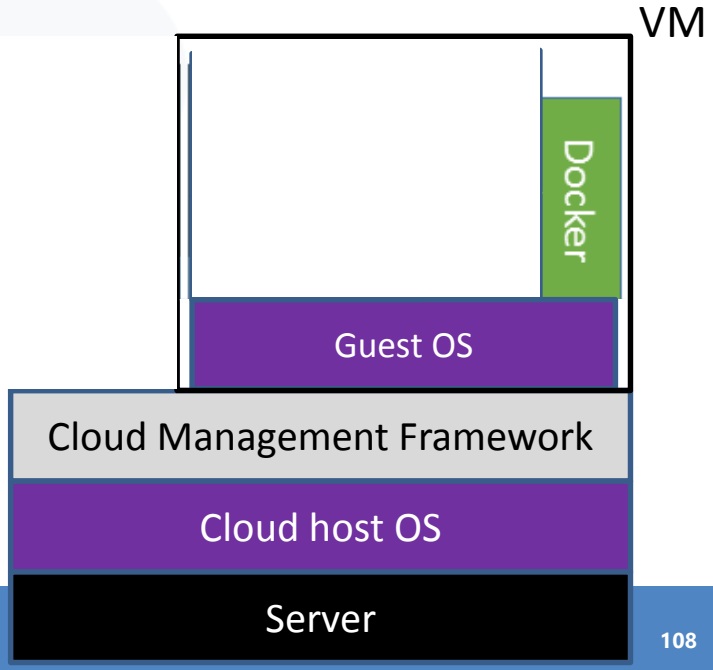


VMs vs containers in the cloud

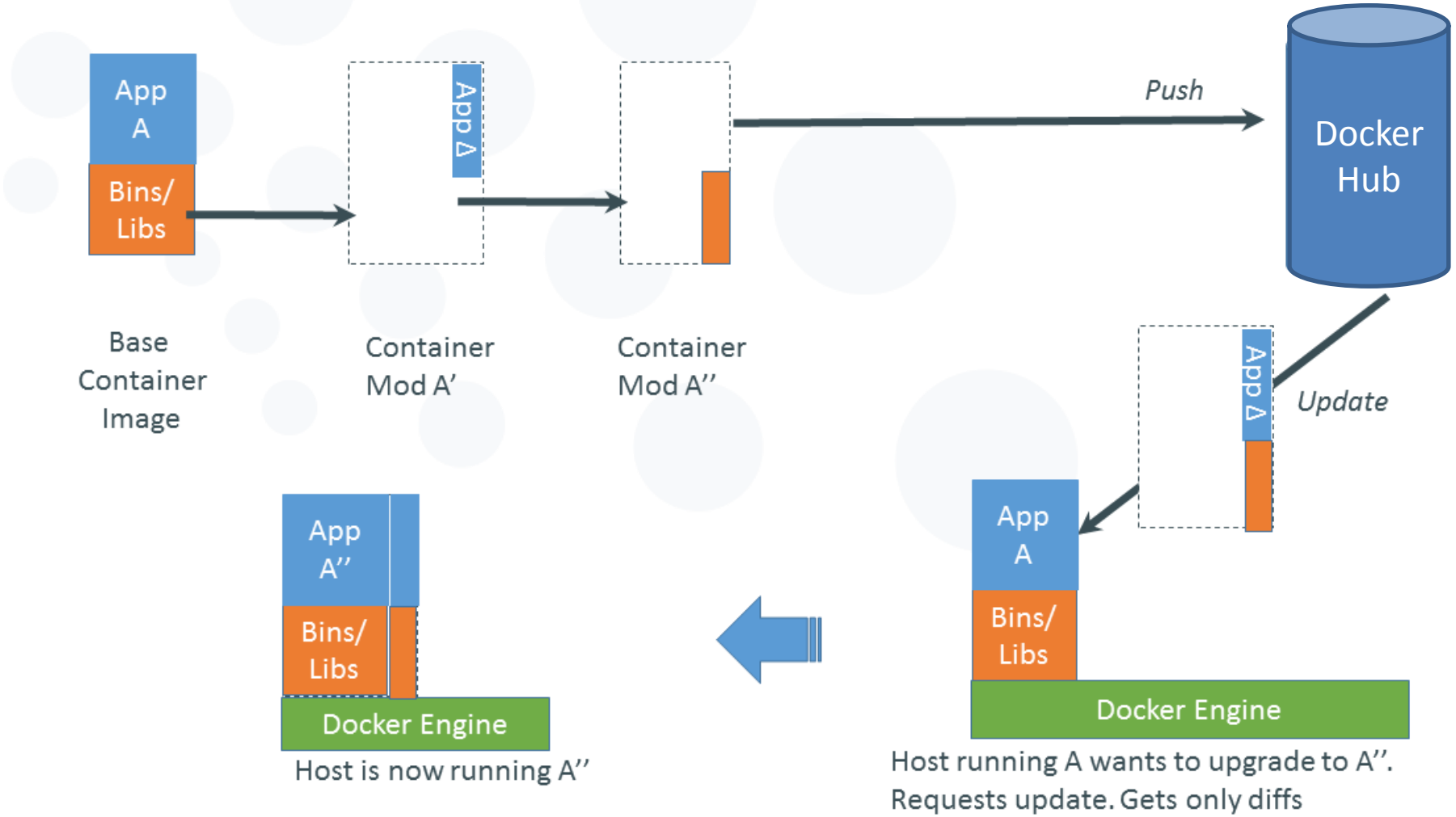


Containers are isolated, but share OS and, where appropriate, bins/libraries

...result is significantly faster deployment, much less overhead, easier migration, faster restart



Updates, Changes
Why are Docker containers lightweight?



Exercise #5: Docker and Galaxy

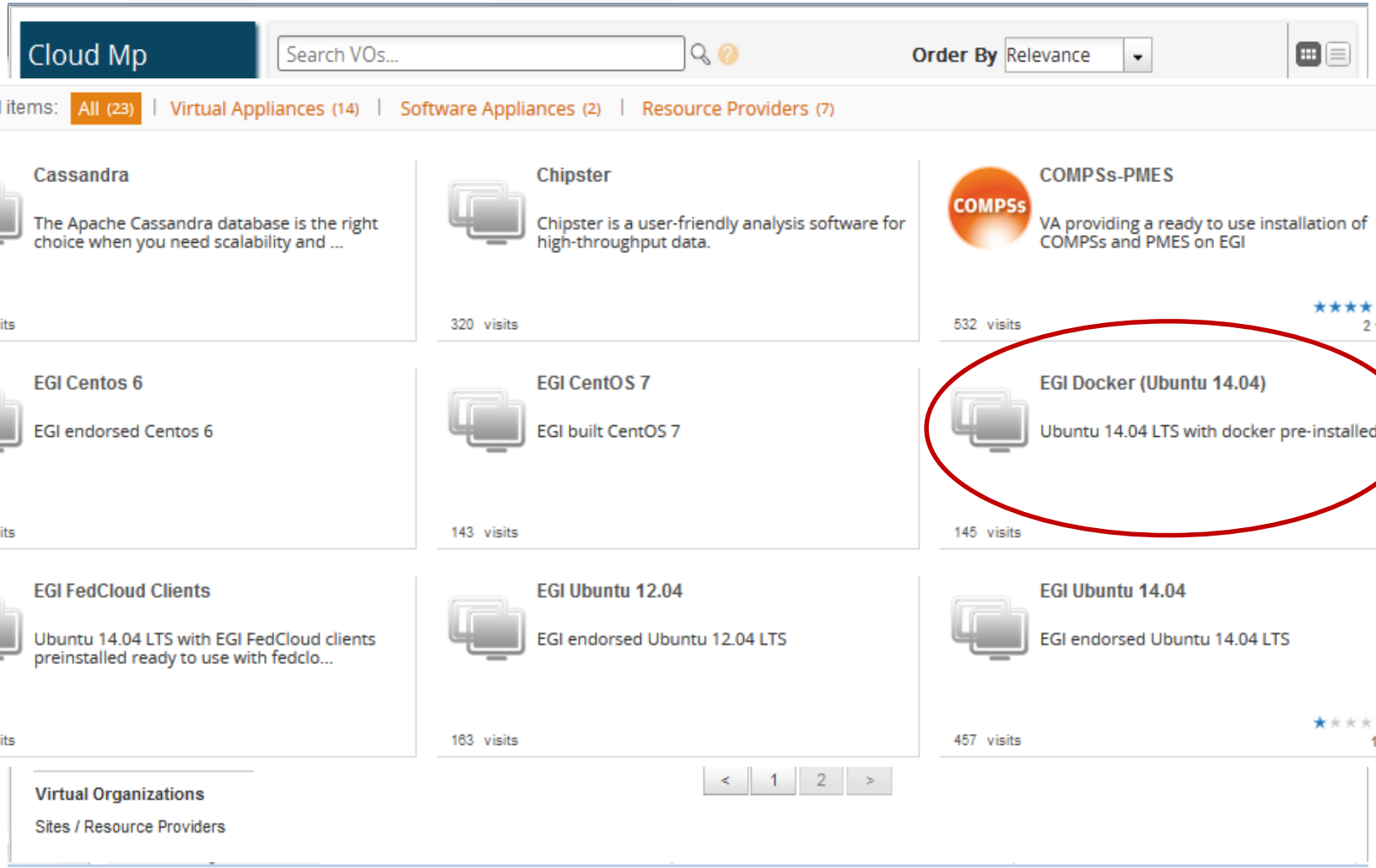
- Galaxy is an open, web-based platform for data intensive biomedical research
- Becoming a standard for running downstream analysis
- There are several "flavours" of Galaxy, depending on the specific needs:
  - NCBI-BLAST
  - ChamicalToolBox
  - Ballaxy
  - NGS-deepTools
  - Galaxy ChIP-exo
  - Galaxy Proteomics
  - Imaging
  - Galaxy for metagenomics

Exercise 5: Running Docker Containers in the EGI Federated Cloud

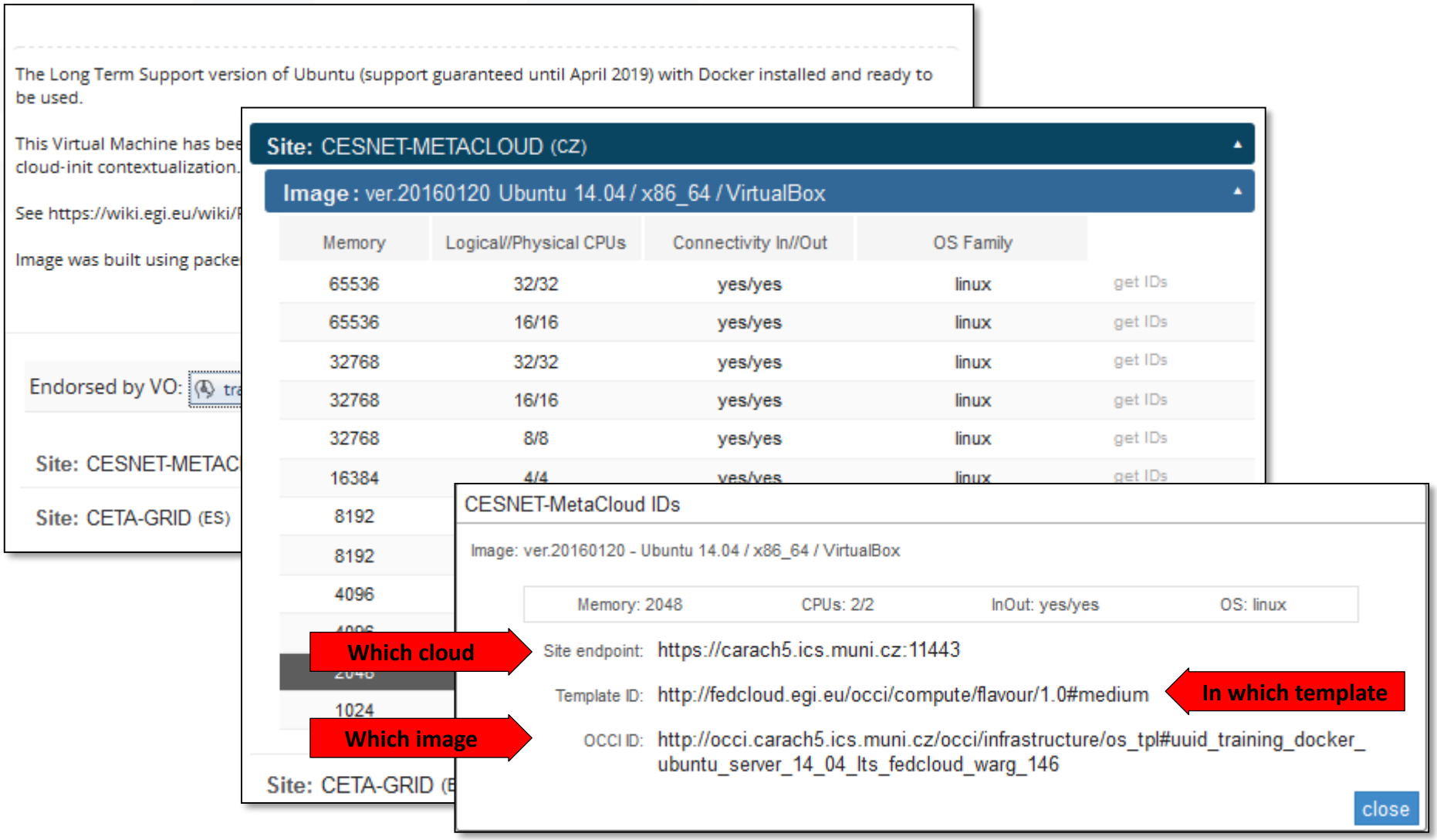
Tasks:

1. Browsing the EGI AppDB
2. Find the 3 IDs required to instantiate the [EGI Docker (Ubuntu 14.04)] image on a cloud site.
3. Login to UI, check proxy, create key
4. Create your VM instance
5. (Optional) Some sites may require the allocation of a public IP before you can log in
6. SSH into your VM instance
7. Run Docker commands and start a Galaxy server
8. Delete your VM

VM with pre-deployed Docker Engine in EGI AppDB



Check the Docker Containers in the EGI AppDB



Create your first compute appliance

Use Docker Ubuntu 14.04 VA values from AppDB!

```
~$ ENDPOINT=<copy here the Site endpoint ID from AppDB>

~$ RESOURCE_TPL=<copy here the Template ID from AppDB>

~$ OS_TPL=<copy here the OCCI ID from AppDB>

~$ occi --endpoint $ENDPOINT \
        --auth x509 --voms --user-cred $X509_USER_PROXY \
        --action create --resource compute \
        --mixin $RESOURCE_TPL --mixin $OS_TPL \
        --attribute occi.core.title="docker$(date +%s)" \
        --context public_key="file:///$HOME/.ssh/id_rsa.pub"
```

Save the ID in an ENV. variable

```
~$ COMPUTE ID=...
```

List and describe your VM instances

- Listing available VMs

```
~$ occi --endpoint $ENDPOINT \
        --auth x509 --voms \
        --user-cred $X509_USER_PROXY \
        --action list --resource compute
```

- Describing your VM

```
~$ occi --endpoint $ENDPOINT \
        --auth x509 --voms \
        --user-cred $X509_USER_PROXY \
        --action describe --resource $COMPUTE_ID
```

This returns lot of info, including the IP Address of your VM!

```
occi.networkinterface.address = …
```

→ It's not so simple ☹ See next slide!

Adding Public IP (Optional)

- If the VM does not have a public IP (e.g. BIFI endpoint) attach a public interface as follows:

```
~$ occi --endpoint $ENDPOINT \
        --auth x509 --voms \
       --user-cred $X509_USER_PROXY \
        --action link --resource $COMPUTE_ID \
        --link /network/public
```

- Obtain the IP address from the output of the describe command.

Logging into the appliance

- ssh with ubuntu user:

```
~$ ssh ubuntu@<your vm ip>
```

Verify if the Docker is installed properly

```
ubuntu@stoor196:~$ sudo docker run hello-world

Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
b901d36b6f2f: Pull complete
0a6ba66e537a: Pull complete
Digest: sha256:8be990ef2aeb16dbcb9271ddfe2610fa6658d13f6dfb8bc72074cc1ca36966a7
Status: Downloaded newer image for hello-world:latest

Hello from Docker.
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ sudo docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker Hub account:
 https://hub.docker.com

For more examples and ideas, visit:
 https://docs.docker.com/userguide/
```

Start the Galaxy server

- After connecting to the newly launched VM, start the Galaxy docker instance as follows:

```
~$ sudo docker run -d -p 8080:80 -p 8021:21
bgruening/galaxy-stable
```

- We also need to clean-up any “orphan” volumes

```
~$ sudo docker volume rm $(sudo docker volume ls -qf
dangling=true)
```

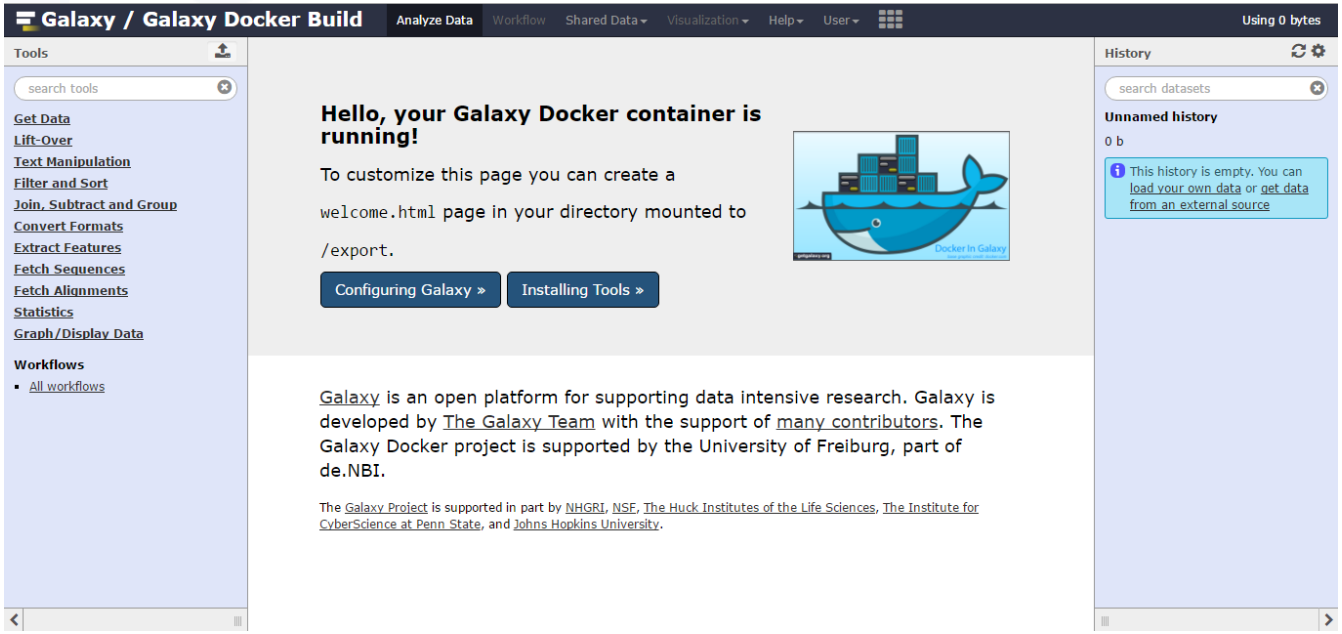
- Galaxy starts a web-server (by default listening to port 8080)
- Go to your web-browser and type:
  - http://[public ip]:8080

Persistent storage vs Read-only

- By default, the Docker Galaxy image is read-only.
  - All changes inside one session will be lost after restart.
- This mode is useful to present Galaxy to your colleagues or to run workshops with it.
- To install Tool Shed repositories or to save your data you need to create a persistent storage volume

Galaxy main interface

- You can create your own user
- For admin rights, the Galaxy Admin User has the username `admin@galaxy.org` and the password `admin`.



Install new tool: “join”

- As “admin” open the Admin section, and go to “Search Tool shed”
  - Search for the “join” tool
  - Preview and install
- After accepting the default setting for all aspects (i.e. dependencies, etc), you will be able to see the new tool on the list
- We need to restart the container in order for the tool to be installed correctly

```
~$ sudo docker ps –a
~$ sudo docker stop <containerID>
~$ sudo docker start <containerID>
```

A simple Galaxy exercise

- **Goal**:
  Find the chr22 exons with the highest number of SNPs

- Get coding exons
  - obtain data from UCSC by clicking **Get Data -> UCSC Main**
- Keep the default setting in the Table Browser, with the exception of position (**chr22**) and assembly (**hg19**)
- In the next screen, select "Coding Exons" and send the query to Galaxy
- The main screen of the Galaxy will now show the newly created step as a new entry in the history (right column)

Modified exercise from Galaxy 101: https://github.com/nekrut/galaxy/wiki/Galaxy101-1

A simple Galaxy exercise

- Get the SNPs;
  - Same process, but this time select “Variation” in the “group” setting
- We can rename the history items, to better reflect our process (“Exons” and “SNPs” respectively)
- Let’s join the two datasets. However, we need the tool we installed in the first step
- Go to “Join, Subtract and Group” (or the group you assigned the new tool to), and run the “join” tool

A simple Galaxy exercise

- Next step would be to count the number of SNPs per exon
  - "Join, Subtract, and Group" -> "Group" tool
  - Use Column 4 as the grouping attribute
  - Insert a new operation "Count" on the same column
- In order to identify the exon with the highest number of SNPs, we need to "sort" and "filter" the new dataset
  - "Filter and Sort" -> "Sort" (on Column 2)
  - "Text Manipulation" -> "Select First" (set top 10)

A simple Galaxy exercise

- The final table should look like this:

1	2
uc002zsw.2_cds_0_0_chr22_21044319_f	33
uc003bhh.3_cds_0_0_chr22_46652458_r	31
uc003alp.4_cds_5_0_chr22_32108069_r	20
uc003alo.2_cds_5_0_chr22_32108069_r	19
uc010gwj.1_cds_4_0_chr22_32108069_r	19
uc003asz.4_cds_2_0_chr22_37964115_f	16
uc010gqp.2_cds_10_0_chr22_16287254_r	16
uc002zoc.3_cds_0_0_chr22_18834445_f	15
uc002zwe.3_cds_0_0_chr22_22868326_r	14
uc003bhw.1_cds_34_0_chr22_46929524_r	14

- We have identified the top exons, but with little information yet.
- Let's combine this, with the original Exon data

A simple Galaxy exercise

- "Join, Subtract and Group" -> "Compare two Queries" tool
  - 1: Exons dataset, using column #4
  - 2. Last dataset, using column #1

- The best way to learn about these exons is to look at their genomic surrounding.
- There is really no better way to do this than using genome browsers.
  - Because this analysis was performed on "standard" human genome (hg38 in this case), you have two choices
    - UCSC Genome Browser, and
    - IGV

A simple Galaxy exercise

A simple Galaxy exercise

- Finally, you can rename and save your entire workflow for use in the future (or other training sessions)

Stop the service

- We can stop the service using the following docker command

```
~$ sudo docker stop <id>
```

- Where the id can be identified with the command:

```
~$ sudo docker ps -a
```

Clean up and stop the VM

- Delete the VM:

```
~$ occi --endpoint $ENDPOINT \
        --auth x509 --voms \
        --user-cred $X509_USER_PROXY \
        --action delete --resource $COMPUTE_ID
```

Preparing your own VM image

Image Creation

- Create VM with some virtualization software:
  - VirtualBox (available for most OS, easy to use)
  - KVM
  - Xen
  - VMWare
  - …
- Once VM is configured as needed, export to the proper format
  - OVF is the preferred format in the EGI Federated Cloud

EGI Endorsed VM images

- EGI prepares and provides a few basic VAs for VOs to use
- Preparation process assures that these are always well-configured, secure and up-to-date
  - *These have 'EGI' in their title in AppDB!*
  - *Ask your VO Manager to add these to the VO-wide list!*
- Guideline to prepare your own image:
  - https://indico.egi.eu/indico/event/2544/session/46/contribution/32

Make your VM available via AppDB

- Package the OVF + disk into OVA
  - https://github.com/EGI-FCTF/VMI-endorsement/blob/master/tools/ovf2ova.sh
- Upload the image to a repository
  - http://appliance-repo.egi.eu/images/ is available if you need
- Register image in AppDB
  - Create a VA entry
    https://wiki.appdb.egi.eu/main:faq:how_to_register_a_virtual_appliance
  - Create a new version within the VA and point to your image location
    https://wiki.appdb.egi.eu/main:guides:guide_for_managing_virtual_appliance_versions_using_the_portal
- Request VA endorsement in your VO
  (so VMI gets distributed to the cloud sites)
  - Endorsement requests are dealt with by designated VO members
    https://wiki.appdb.egi.eu/main:guides:notify_virtual_organization_representatives

VO Manager needs to include your VM in the VO-wide image list.
This triggers replication of your image to the clouds of your VO.

Which approach to follow?

Contextualization

- Fast to use, just add user data to VM
- Configuration on creation
- Slow start-up of VM
- Works on top of existing images
- Hard to debug if fails

Docker

- Medium complexity. Needs Docker Engine
- Configuration on creation
- Fast VM start-up, separate application start-up
- Works on Docker-enabled images
- Debug in local environment

Custom Images

- Time consuming, needs virtualization software
- Static configuration
- Fast start-up of VM
- Requires moving large files to sites
- Easier to debug

- **EGI.eu maintains core VM images in AppDB.**
- **These can be used as starting point in all three scenarios.**

Keep in mind!

- You have **root** access to your virtual machines
- Your virtual machines are often visible from the Internet
- It is up to you to keep your virtual machines updated and secure
- **DO NOT USE** password-based authentication for remote access
- You should terminate your virtual machine as soon as it is not needed anymore

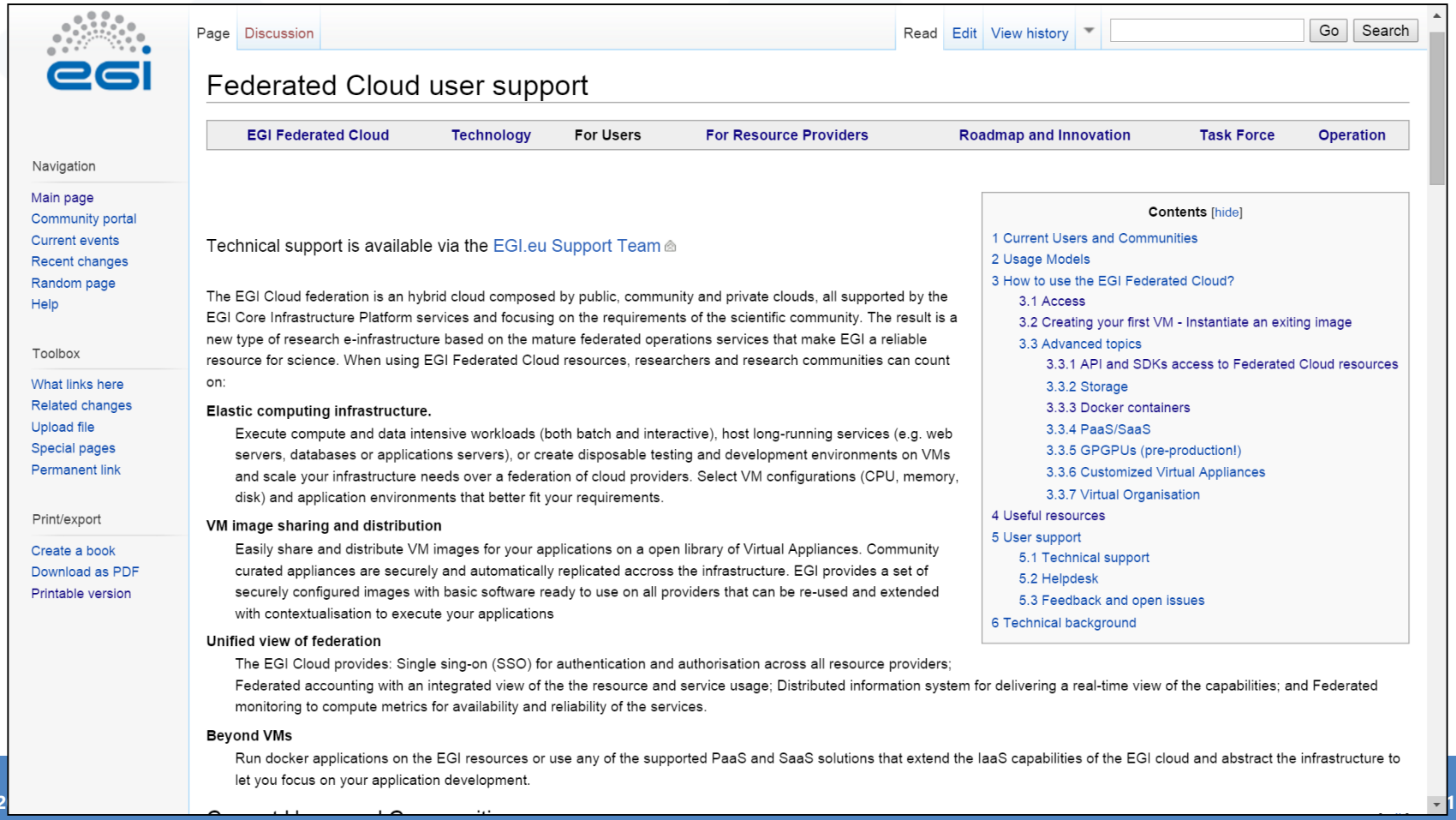
EGI-Engage

Next steps

Main resource

EGI Federated Cloud Documentations and Guides:

- https://wiki.egi.eu/wiki/Federated_Cloud_user_support



Page Discussion Read Edit View history Go Search

Federated Cloud user support

EGI Federated Cloud | Technology | For Users | For Resource Providers | Roadmap and Innovation | Task Force | Operation

Navigation
Main page
Community portal
Current events
Recent changes
Random page
Help

Toolbox
What links here
Related changes
Upload file
Special pages
Permanent link

Print/export
Create a book
Download as PDF
Printable version

Contents [hide]
1 Current Users and Communities
2 Usage Models
3 How to use the EGI Federated Cloud?
3.1 Access
3.2 Creating your first VM - Instantiate an exiting image
3.3 Advanced topics
3.3.1 API and SDKs access to Federated Cloud resources
3.3.2 Storage
3.3.3 Docker containers
3.3.4 PaaS/SaaS
3.3.5 GPGPUs (pre-production!)
3.3.6 Customized Virtual Appliances
3.3.7 Virtual Organisation
4 Useful resources
5 User support
5.1 Technical support
5.2 Helpdesk
5.3 Feedback and open issues
6 Technical background

Technical support is available via the EGI.eu Support Team

The EGI Cloud federation is an hybrid cloud composed by public, community and private clouds, all supported by the EGI Core Infrastructure Platform services and focusing on the requirements of the scientific community. The result is a new type of research e-infrastructure based on the mature federated operations services that make EGI a reliable resource for science. When using EGI Federated Cloud resources, researchers and research communities can count on:

Elastic computing infrastructure.
Execute compute and data intensive workloads (both batch and interactive), host long-running services (e.g. web servers, databases or applications servers), or create disposable testing and development environments on VMs and scale your infrastructure needs over a federation of cloud providers. Select VM configurations (CPU, memory, disk) and application environments that better fit your requirements.

VM image sharing and distribution
Easily share and distribute VM images for your applications on a open library of Virtual Appliances. Community curated appliances are securely and automatically replicated accross the infrastructure. EGI provides a set of securely configured images with basic software ready to use on all providers that can be re-used and extended with contextualisation to execute your applications

Unified view of federation
The EGI Cloud provides: Single sing-on (SSO) for authentication and authorisation across all resource providers; Federated accounting with an integrated view of the the resource and service usage; Distributed information system for delivering a real-time view of the capabilities; and Federated monitoring to compute metrics for availability and reliability of the services.

Beyond VMs
Run docker applications on the EGI resources or use any of the supported PaaS and SaaS solutions that extend the IaaS capabilities of the EGI cloud and abstract the infrastructure to let you focus on your application development.

Getting access to the FedCloud

Your steplist:

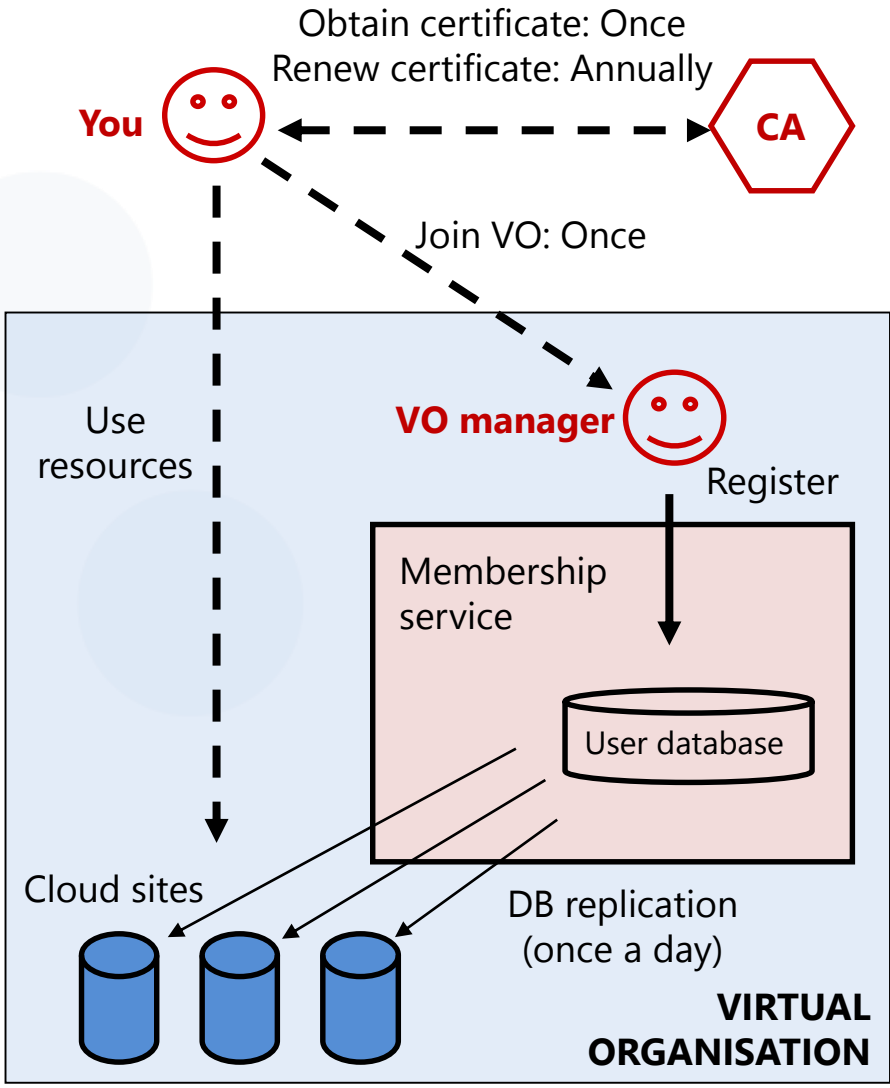
1. Obtain certificate from

   National CA (face-to-face identity check)
   http://www.igtf.net

   OR

   Terena Certificate Service: (online)
   https://www.digicert.com/sso

2. Register at the VO
   - fedcloud.egi.eu is a good starting point
   - Other VOs: http://operations-portal.egi.eu/vo/search
3. VO manager authorizes You
   - Membership DB updated
   - Identity replicated to resource within 1 day
4. Interact with the resorurces
   - rOCCI
   - API
   - High-level tool



Support services

Dedicated technical consultancy for any user or community:

support@egi.eu

F2F/Web Meetings	• Identify suitable setup • Allocate technical experts • Define milestones
Doc	• Step by step guides • Tutorials – CMD line, API • Examples
Continuous tracking and support	• Technical integration • Periodic meetings
EGI VM Images	• Main OS versions • Secure, up-to-date • Contextualisation • Docker
Fedcloud.egi.eu VO	• Resources for prototyping • Enabled on all sites • Usable for 2x6 months
Migration to production	• Identifying committed resource providers • Support for VO setup • SLAs, OLAs

Support through the NGIs

http://www.egi.eu/about/ngis/

- EGI's federated support model
  - National support teams (NGIs)
  - Topic/discipline-specific support teams (see next slide)
  - EGI.eu UCST – primarily coordination & support for supporters (support@egi.eu)

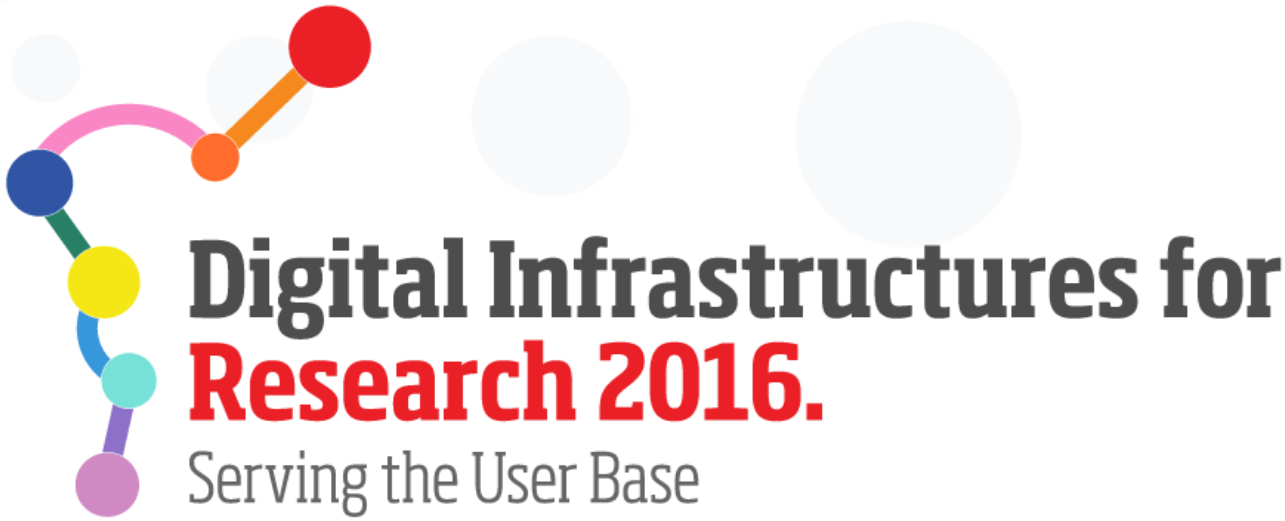
Organise a training event

- You can book the EGI training infrastructure for your tutorials:
  - https://wiki.egi.eu/wiki/Training_infrastructure

- Please email support@egi.eu specifying training course details :
  - date, number of students, kind of audiance, special needs, etc.

DI4R 2016 Conference
Digital Infrastructures for Research

- Krakow, Poland
- 2016, September 28-30.
- Co-organised by EGI, EUDAT, GEANT, OpenAIRE and RDA Europe
- Call for participation (**deadline 3 June!**)



- http://www.digitalinfrastructures.eu/

Thank you for your attention.

Questions?

PLEASE RETURN THE FEEDBACK FORMS!

www.egi.eu