VSC HPC Introduction

ICTS KU Leuven
VSC staff:
Ehsan Moravveji
Mag Selwa
Geert Jan Bex (Uhasselt)
Jan Ooghe

Questions and problems troubleshooting

- Blackboard Collaborate session: https://eu.bbcollab.com/guest/ce4229a2f9eb4cc5b90f490d660eb1c2

Material

- Everything is on Github: https://hpcleuven.github.io/HPC-intro/
- Video Recordings
  - Scan the QR code
  - Recommended videos: ~ 2 hrs
  - Optional videos: ~1 hr

What is High Performance Computing?

- using supercomputers to solve advanced computation problems
- Reduce the computation time from days, years, decades, or centuries to minutes, hours, days, or weeks
- The key is parallelism

In practice, it is more like …

The concept is simple: **Parallelism** = employing multiple processors for a single problem

Outline

- What is the VSC?
- What is a cluster?
- Genius Cluster
- Storage
- Login nodes
- Connection Setup
- Software environment
- How to submit jobs?
- Dedicated hardware
- How to choose resources?
- Optional material
  - Linux in brief
  - Conda for Python and R
  - Worker Framework

VSC

(Vlaams Supercomputer Centrum)

VSC PARTNERSHIP

Supported by

fwo

ASSOCIATIE UNIVERSITEIT GENT

universitaire associatie BRUSSEL

ASSOCIATIE KU LEUVEN

antwerp university association

ASSOCIATIE UNIVERSITEIT-HOGESCHOLEN LIMBURG

VSC HPC Environments

Tier-0

PRACE

Vlaanderen
is supercomputing

Tier-1

BrENIAC

Tier-2

GHENT UNIVERSITY — STEVIN

VUB VRIJE UNIVERSITEIT BRUSSEL — HYDRA

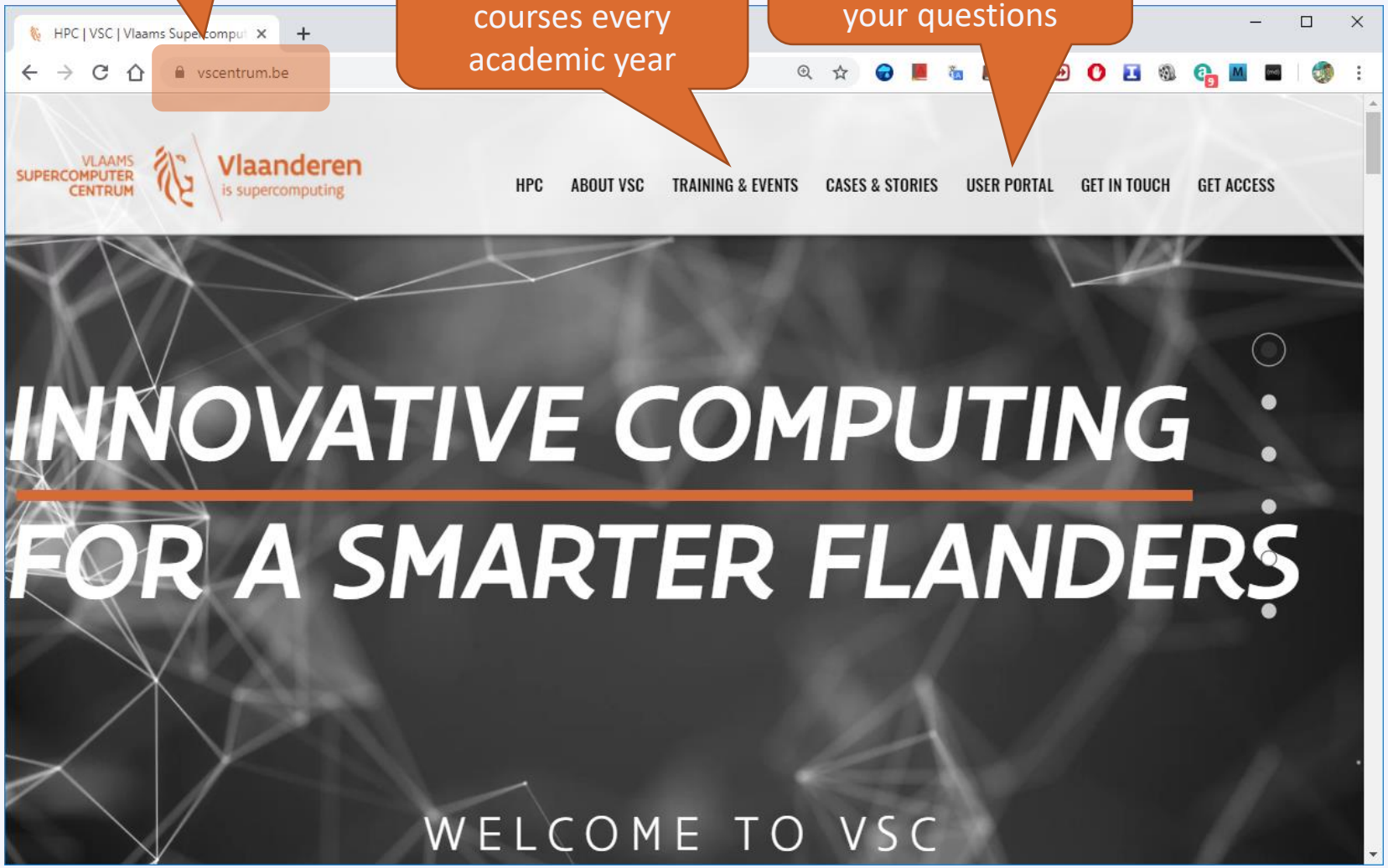
Universiteit Antwerpen — HOPPER/LEIBNIZ

KU LEUVEN UHASSELT — THINKING//GENIUS

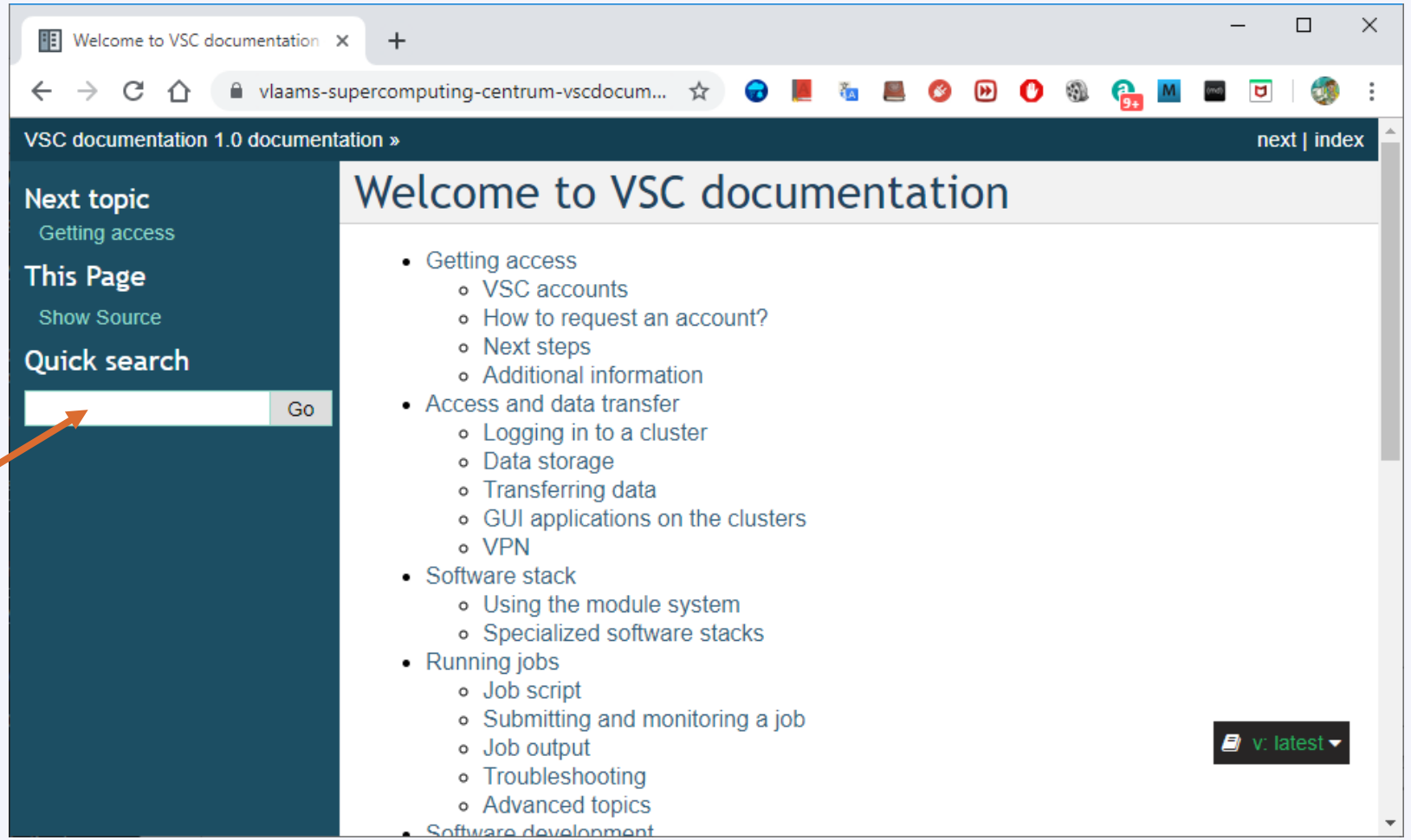
www.vscentrum.be

Very rich suite of courses every academic year

Documentation! Answers >80% of your questions



VLAAMS SUPERCOMPUTER CENTRUM

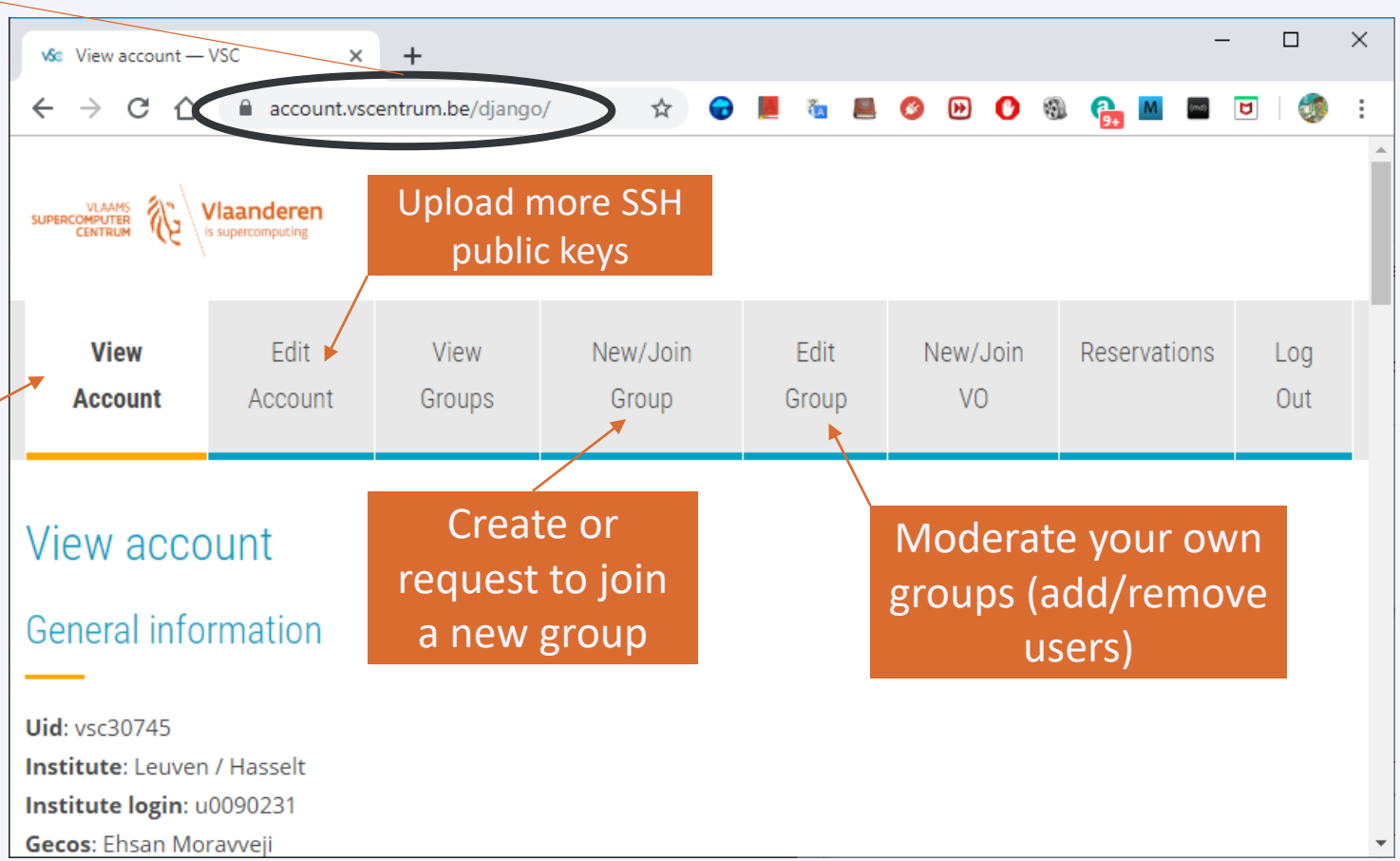


Search you keywords here; e.g. qsub

Support and Services

Basic support	• Helpdesk (hpcinfo@kuleuven.be) • Monitoring and reporting
Application support	• Installation and porting • Optimisation and debugging • Benchmarking • Workflows and best practices
Training	• Documentation and tutorials • Scheduled trainings / workshops • On request workshops • One-to-one sessions

To manage your VSC account: account.vscentrum.be



Upload SSH Key
Request account

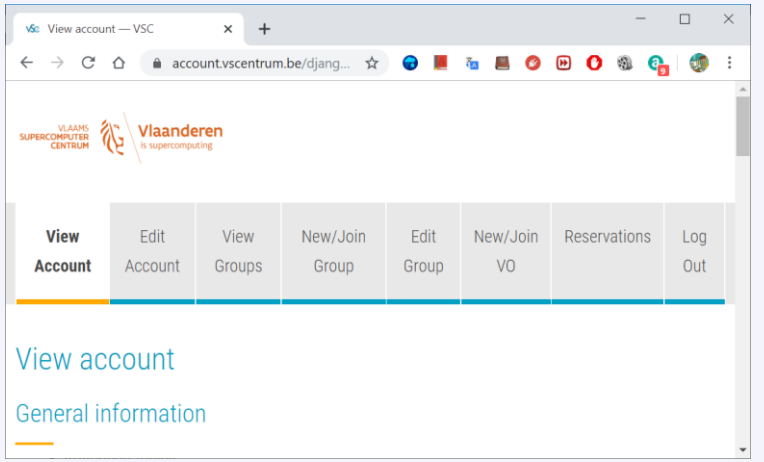
Create or request to join a new group

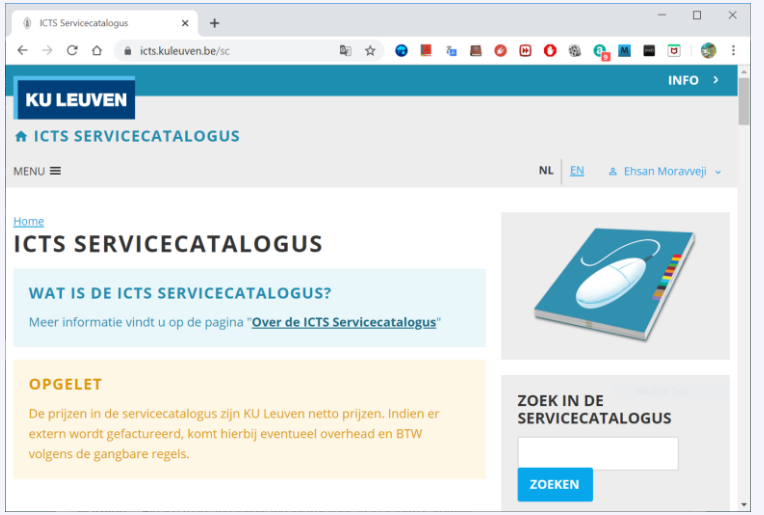
Moderate your own groups (add/remove users)

VLAAMS SUPERCOMPUTER CENTRUM

Become a VSC user

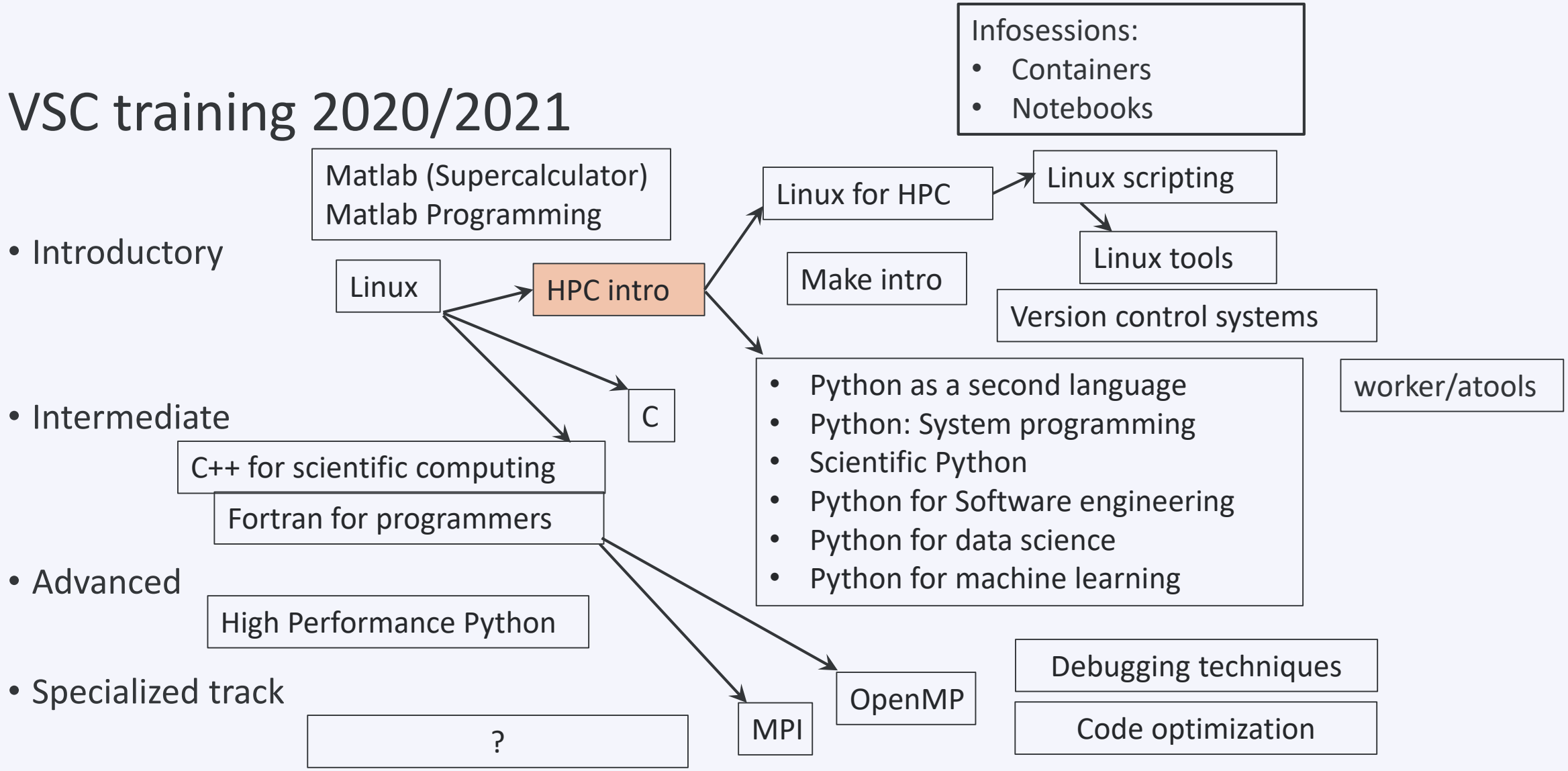
- ❑ Create a secure (4096 bit) SSH key pairs
  Upload it on the account page: www.account.vscentrum.be
- ❑ You need to request a VSC account
  Normally processed swiftly
- ❑ Request introductory credits (2000 free credits for 6 months)
- ❑ Request project credits (for supervisors and project leaders)
  You need to create a VSC group
  Add users to the group to give them access to use credits
  Fill out the request form
- ❑ Extra storage requests
  Scratch extension: free of charge
  Archive fileset: 70 € per TB per year
  Staging fileset: 130 € per TB per year
- ❑ All service costs (compute and storage) are all explained
  Go to ICTS service catalogus: https://icts.kuleuven.be/sc
  Click on High Performance Computing (NL/EN)





VLAAMS SUPERCOMPUTER CENTRUM

VSC training 2020/2021



PRACE MOOC Defensive programming and debugging: https://www.futurelearn.com/courses/defensive-programming-and-debugging

To Acknowledge VSC in publications

Why?

- ❑ a contractual obligation for the VSC
- ❑ helps VSC secure funding
- ❑ you will benefit from it in the long run

At KU Leuven

- ❑ add the relevant papers to the virtual collection "High Performance Computing" in Lirias

In het nederlands

De rekeninfrastructuur en dienstverlening gebruikt in dit werk, werd voorzien door het VSC (Vlaams Supercomputer Centrum), gefinancierd door het FWO en de Vlaamse regering – departement EWI.

In English

The computational resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation - Flanders (FWO) and the Flemish Government – department EWI.

Tier-2 Clusters

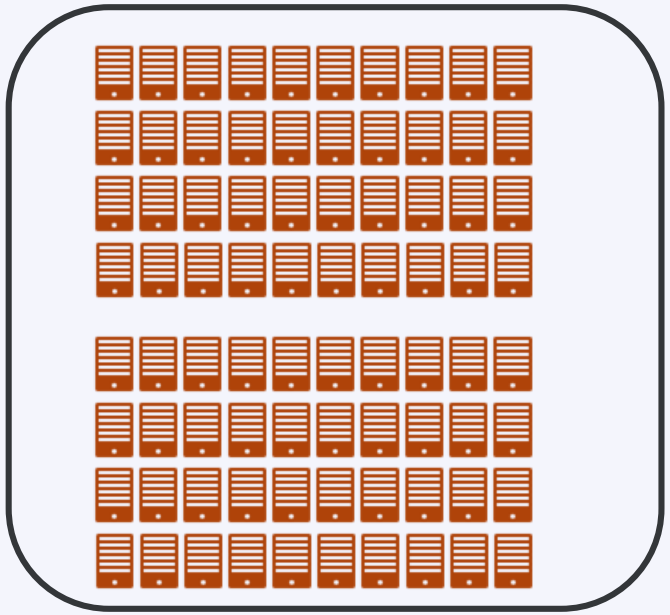
Tier-2 Clusters @ KU Leuven

ThinKing (since 2014)
352 nodes: 7,616 cores

Genius (since 2018)
250 nodes: 8,936 cores

Tier-2 Overview

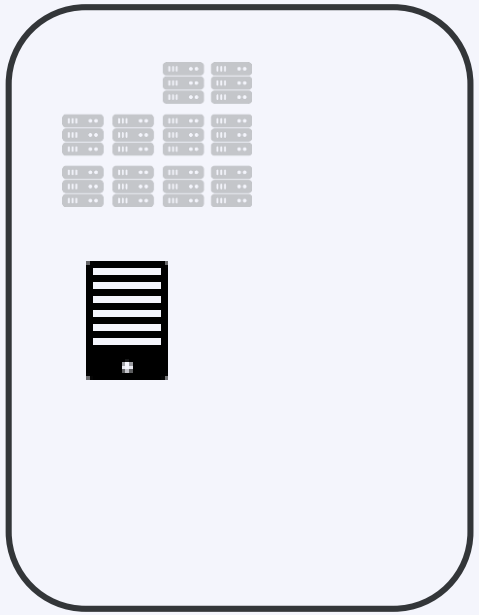
Compute nodes



Thinking:
+ 48x + 86x Haswells 24c 64/128 GB

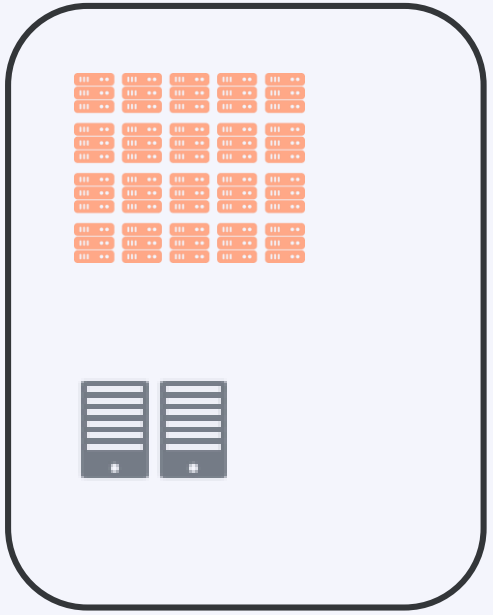
Genius:
+ 96x Skylake 36c 192 GB
+ 144x CascadeLake 36c 192 GB

Large memory nodes



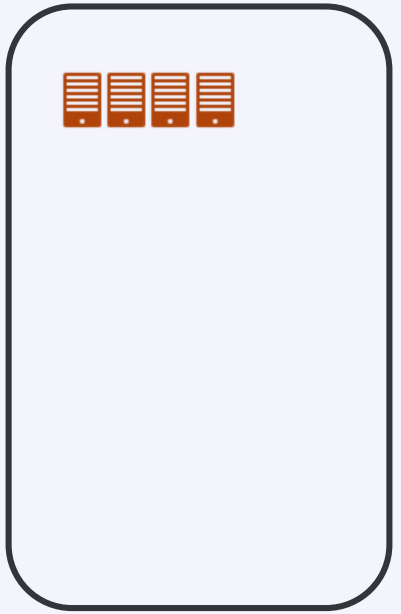
Genius:
+ 10x Skylake 36c 768GB
+ 1x Superdome 112c 6 TB

GPU nodes



Genius:
+ 20x Skylake 36c 192 GB
  4x P100 16GB
+ 2x CascadeLake 36c 768GB
  8x V100 32GB

Exp nodes



Genius:
4 AMD Naples 64c 256 GB

Technical Hardware Specifications

	Tier 2					Tier 1		
Cluster name	ThinKing		Genius			BrENIAC		
Processor type	Haswell		SkyLake		Cascade Lake	Broadwell		SkyLake
Cores per node	24		36		36	28		28
Base Clock Speed	2.5 GHz		2.3 GHz		2.6 GHz	2.4 GHz		2.6Ghz
Total nodes	48	96	86	10	144	580		408
Node memory (GB)	64	96	192	768	192	128	256	192
Memory per core (GB)	2.5	3.8	5.2	21.2	5.3	4.4	9.0	6.7
Total cores	3,456		3,456		4,320	27,664		
Peak performance (Flops/cycle)	8 DP FLOPs/cycle: 4-wide FMA (fused multiply-add) instructions AVX2		16 DP FLOPs/cycle: 8-wide FMA (fused multiply-add) instructions AVX-512		16 DP FLOPs/cycle: 8-wide FMA (fused multiply-add) instructions AVX-512	8 DP FLOPs/cycle: 4-wide FMA (fused multiply-add) instructions AVX2		16 DP FLOPs/cycle: 8-wide FMA (fused multiply-add) instructions AVX-512
Network	Infiniband FDR		Infiniband EDR		Infiniband EDR	Infiniband EDR		
Cache (L1 KB/L2 KB/L3 MB)	12x(32i+32d) / 12x256 / 30MB		18x(32i+32d) / 18x1024 / 25 MB		18x(32i+32d) / 18x1024 / 25 MB	14x(32i+32d) / 14x 256 / 35 MB		18x(32i+32d) / 18x1024 / 25 MB

	Tier 2	Tier 1
Access	All researchers and students	Approved project experienced users
Start with	2,000 introduction credits	500 node days starting grant
Project Credits	Pay as you GO	Panel Review Project proposal Free
Computation Limit	None	500 - 5,000 node days
Application deadline	None	3 February 5 June 2 October
Max. Project Duration		8 months
Available walltime	1h, 24h, 72h, 7d	Max. 3 days

Genius

SkyLake (since 8/2018)
CascadeLake (since 1/2020)

Tier-2 Cluster: Genius

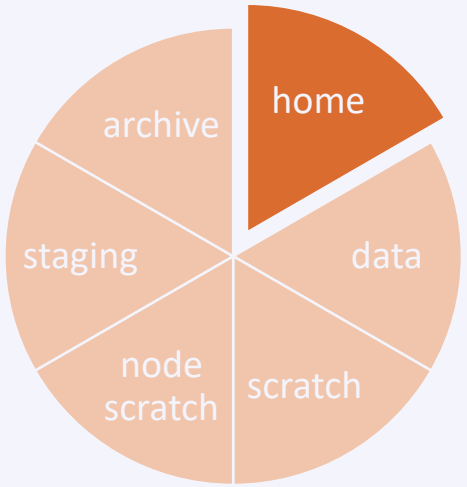
Type of node	CPU type	Inter-connect	# cores	installed mem	local discs	# nodes
SkyLake	Xeon 6140	IB-EDR	**36**	**192 GB**	800 GB	**86**
SkyLake **large mem**	Xeon 6140	IB-EDR	**36**	**768 GB**	800 GB	**10**
SkyLake **GPU**	Xeon 6140 **4xP100** SXM2	IB-EDR	**36**	**192 GB**	800 GB	**20**
CascadeLake	Gold 6240	IB-EDR	**36**	**192 GB**	800 GB	**144**
CascadeLake **GPU**	Gold 6240 **8xV100** SMX2	IB-EDR	**36**	**768 GB**	800 GB	**2**
SkyLake **Superdome**	Gold 6132	Flex Grid	**14**	**6 TB**	6 TB	**8**

Storage

Overview of the storage infrastructure

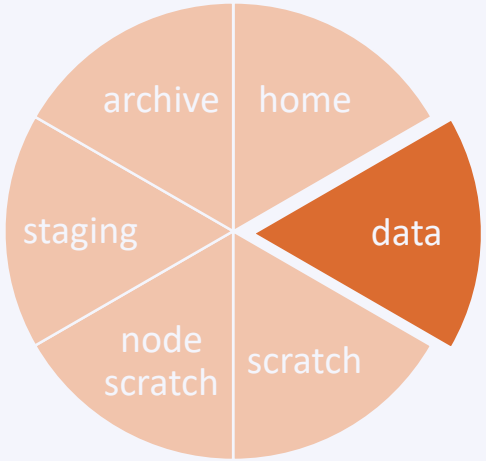
- Your files are owned only by you.
  Other VSC users have no permission to read/write/execute your files (POSIX)
- A VSC account has 3 default storages (free of charge)
  - `$VSC_HOME`
  - `$VSC_DATA`
  - `$VSC_SCRATCH`
- You can additionally request `staging` and `archive` storages
- Different storage volumes have different:
  - mount point
  - size and performance
  - use case
  - backup and maintenance policy
- More info on ICTS Service Catalog (EN/NL)

Storage



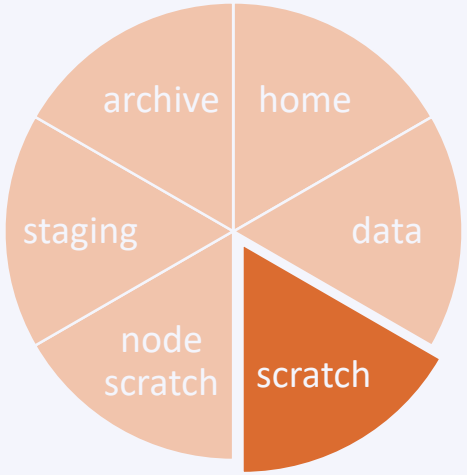
Storage	home folder
Env. Variable	`$VSC_HOME`
Filesystem Type	NFS
Access	Global
Backup	Hourly, daily, weekly (until last month)
Default Quota	3 GB
Extension	Not possible
Usage	Only storing SSH keys, config files
Remarks	- Stay away from using it - Can easily overflow: + Your jobs may crash + Login issues

Storage



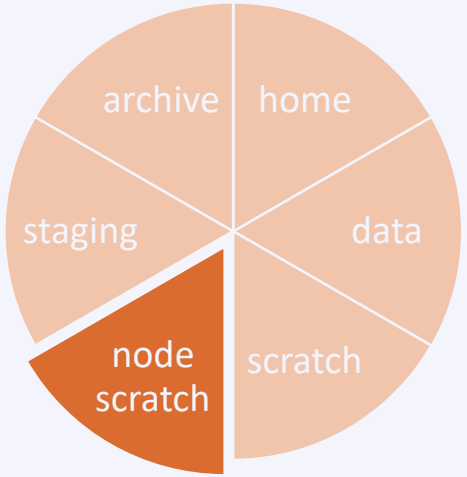
Storage	data folder
Env. Variable	`$VSC_DATA`
Filesystem Type	NFS
Access	Global
Backup	Hourly, daily, weekly (until last month)
Default Quota	75 GB
Extension	On purchase
Usage	Your data, code, software, results
Remarks	- Permanent storage for initial/final results - Not optimal for intensive or parallel I/O

Storage



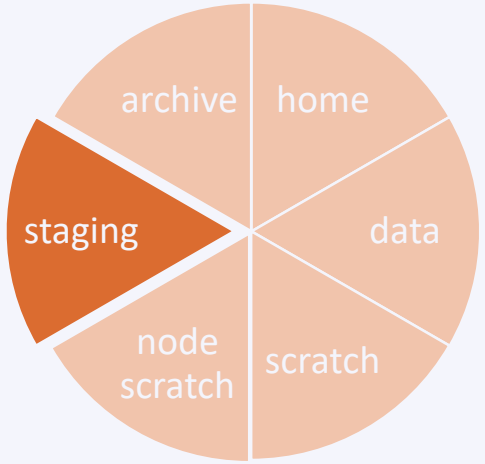
Storage	scratch folder
Env. Variable	`$VSC_SCRATCH`
Filesystem Type	GPFS
Access	Local
Backup	delete after 28 days from last access
Default Quota	100 GB
Extension	Free
Usage	Intensive, parallel I/O, temporary files
Remarks	- Recommended storage for all jobs - Copy scratch files to VSC_DATA or local storage after jobs are done - Deleted files cannot be recovered

Storage



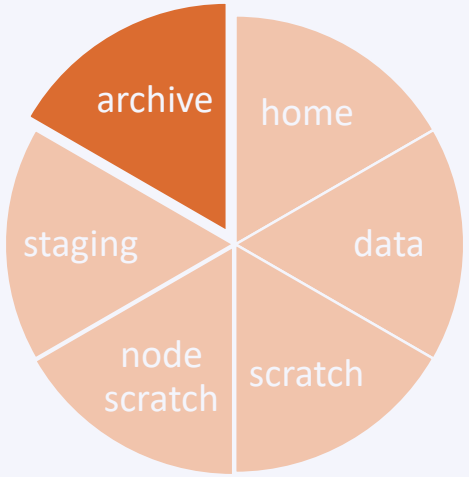
Storage	Node scratch folder
Env. Variable	`$VSC_SCRATCH_NODE`
Filesystem Type	GPFS
Access	On compute node, only at runtime
Backup	None
Default Quota	200 GB
Extension	Read about beeOND
Usage	Temporary storage at runtime
Remarks	- Fastest I/O, attached to the node - Is cleaned after job terminates - Copy the data to your home, scratch, or staging before job ends

Storage



Storage	Staging folder
Path	`/staging/leuven/stg_000XX`
Filesystem Type	GPFS
Access	On demand, only Tier-2@KUL
Backup	None
Default Quota	None
Extension	On purchase, from 1 TB
Usage	Permanent; share with a group
Remarks	- Accessible from login/compute nodes - Fast, parallel I/O

Storage



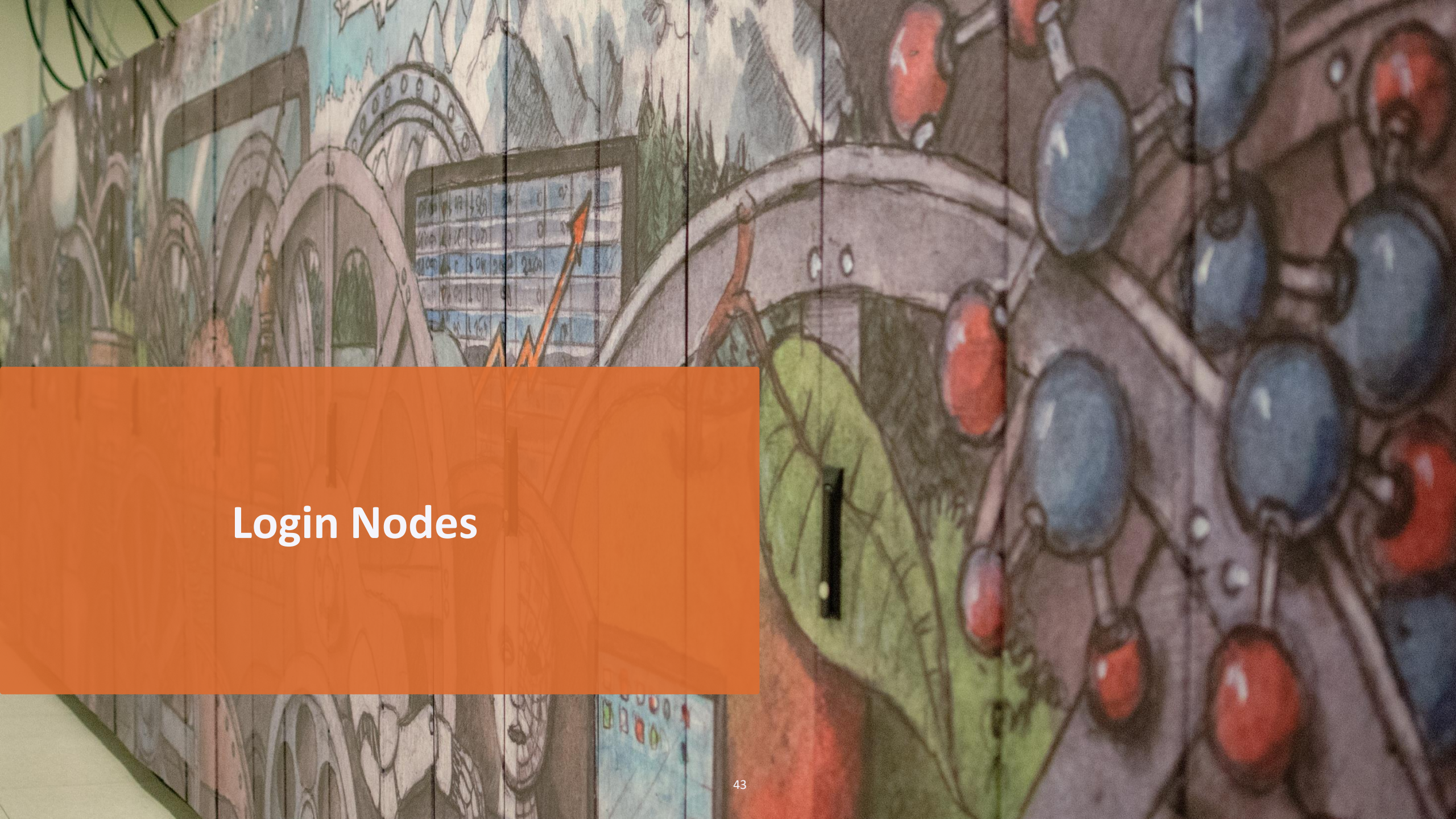
Storage	Archive folder
Path	`/archive/leuven/arc_000XX`
Filesystem Type	NFS
Access	On demand, only Tier-2@KUL
Backup	None
Default Quota	None
Extension	On purchase, from 1 TB
Usage	Permanent; share with a group
Remarks	- Accessible only from login nodes - Cannot use it from compute nodes - Slow, but durable filesystem

Storage

- Request form for extra storage
- More information
- Do not use `/tmp`
  It is only 10 GB and is reserved for the OS and root processes
  Your application can crash if using `/tmp`
- You are automatically logged into your home folder upon login.
  Make sure you immediately go to your other storages, e.g.
  `$> cd $VSC_DATA`
- Always check your storage balance using `myquota` command

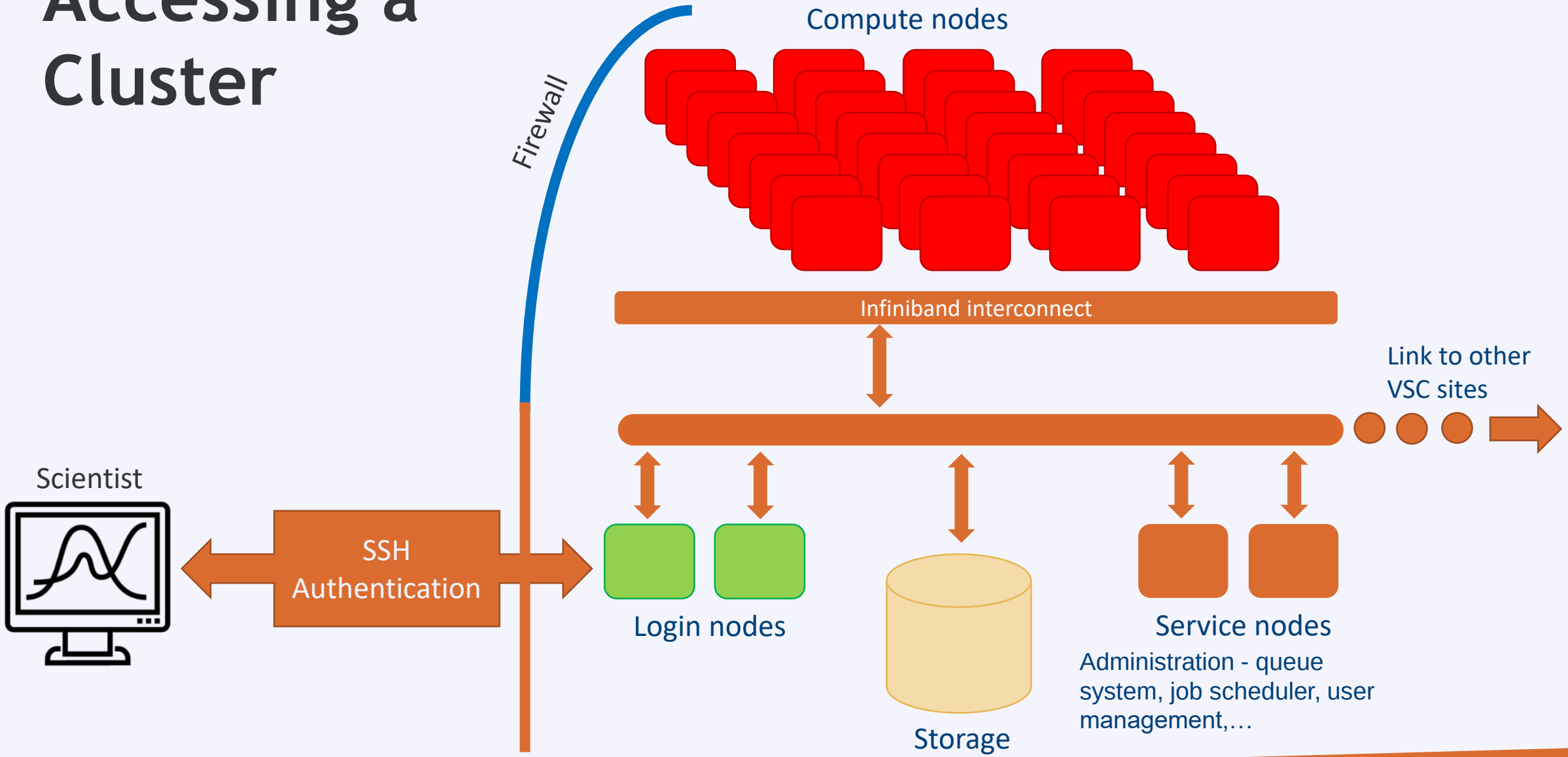
Example

```
$> myquota
file system $VSC_HOME
    Blocks: 1479M of 3072M
    Files: 12934 of 100k
file system $VSC_DATA
    Blocks: 102G of 225G
    Files: 1043k of 10000k
file system $VSC_SCRATCH
    Blocks: 15M of 1.5T
```



Login Nodes

Accessing a Cluster



VLAAMS SUPERCOMPUTER CENTRUM

Using Login Nodes

Tips

- ❑ To develop and/or compile code and/or software
- ❑ To check your storage and credit balance
- ❑ To manage jobs (submit, check status, debug, resubmit, …)
- ❑ To move data around
  within VSC: use data, scratch, staging, archive
  outside VSC: copy/sync from/to your local storage
- ❑ To pre-process or post-process your data/jobs
- ❑ To visualize your data
- ❑ To share files/folders

Warning

- ❑ Login nodes are shared resources
- ❑ **Do not** execute heavy-lifting tasks (core, memory)
- ❑ Instead, submit jobs

Login Hosts on Different Machines/partitions

- Windows: PuTTY or MobaXterm or NX
  Linux/Mac: terminal or NX
- To login, you need an active VSC number and a hostname
  `$> ssh -X vscXXXXX@<hostname>`

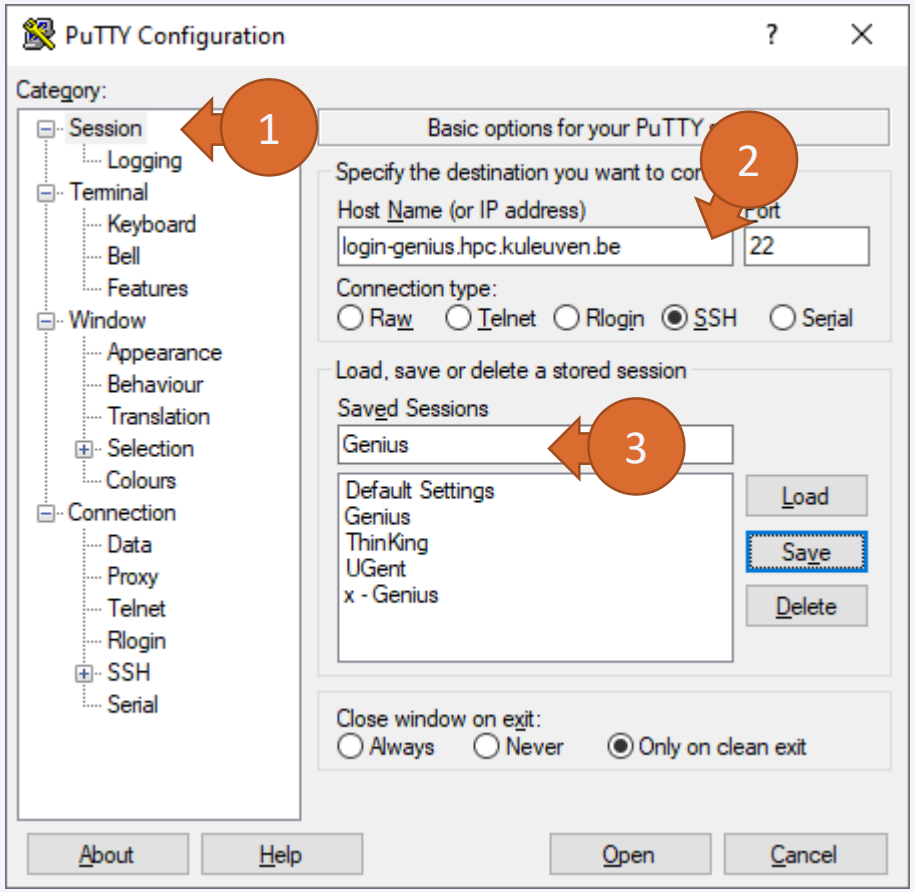
Cluster / Partition	<hostname>	Remark(s)
ThinKing	`login-thinking.hpc.kuleuven.be`	Recommended
	`login7-tier2.hpc.kuleuven.be` `login8-tier2.hpc.kuleuven.be`	Haswell partition
Genius	**`login.hpc.kuleuven.be`** `login-genius.hpc.kuleuven.be`	Recommended
	`login{1,2}-tier2.hpc.kuleuven.be`	No GPU
	`login{3,4}-tier2.hpc.kuleuven.be`	Nvidia Quadro P6000
Superdome	Any Genius login node (above)	`-q qsuperdome -l partition=superdome`
AMD	Any Genius login node (above)	`-l partition=amd`

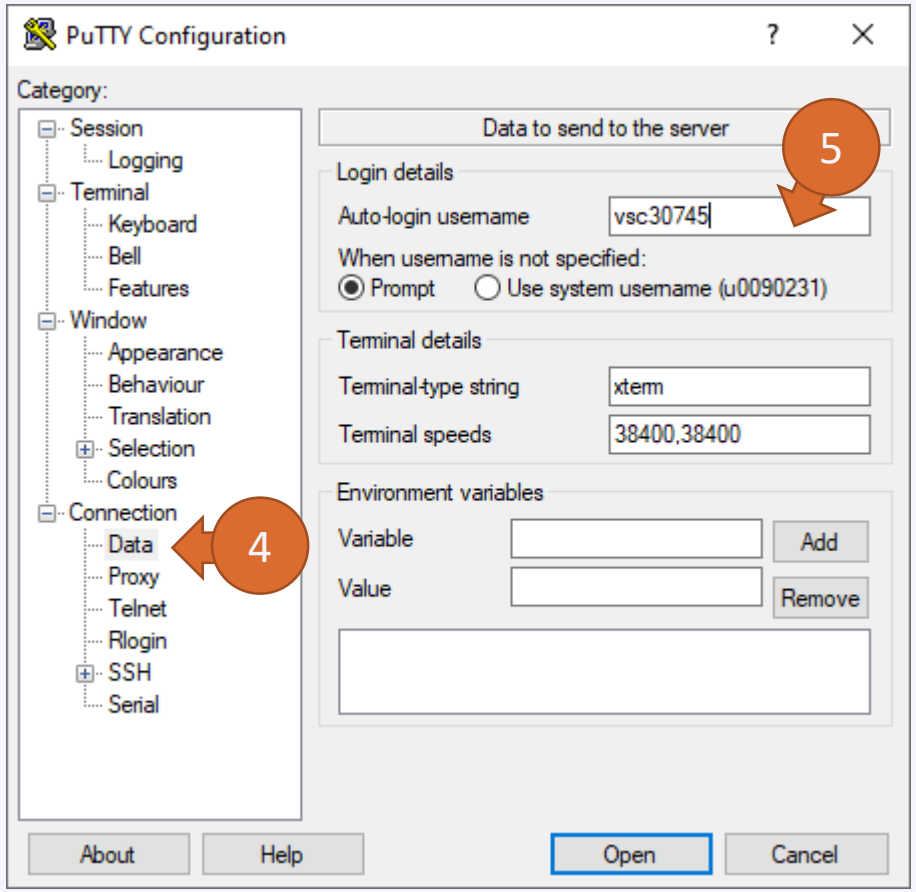
Login Setup

- ❑ PuTTY
- ❑ Terminal

Setup PuTTY in 12 Clicks

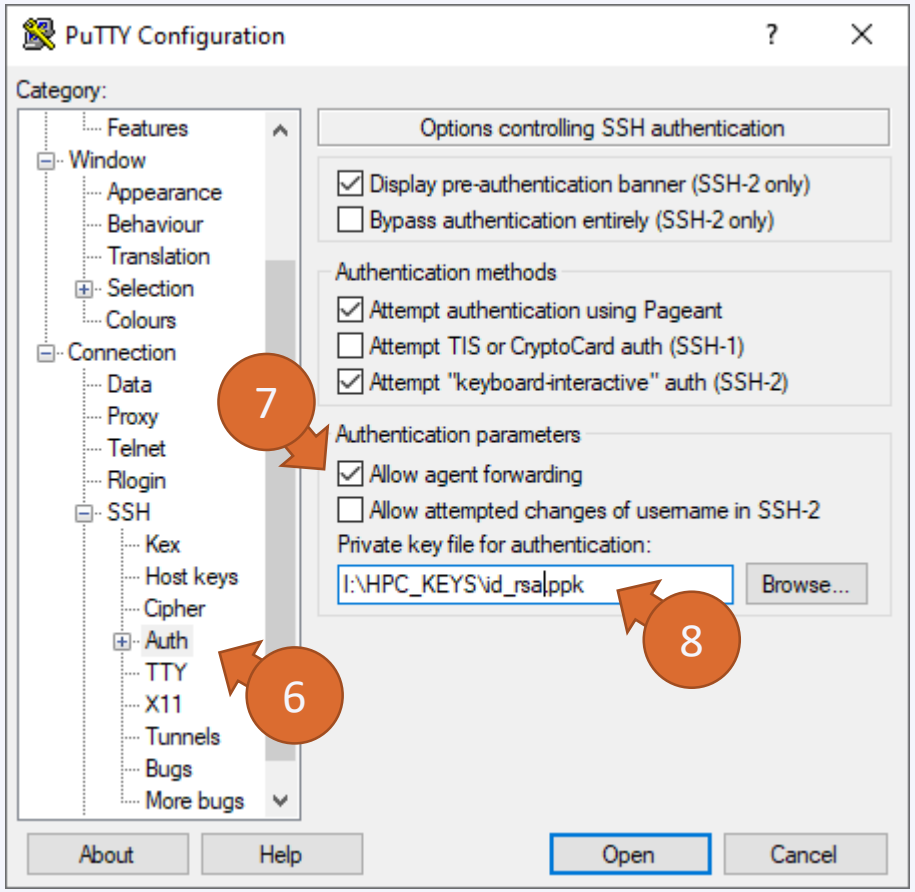
(Windows only)

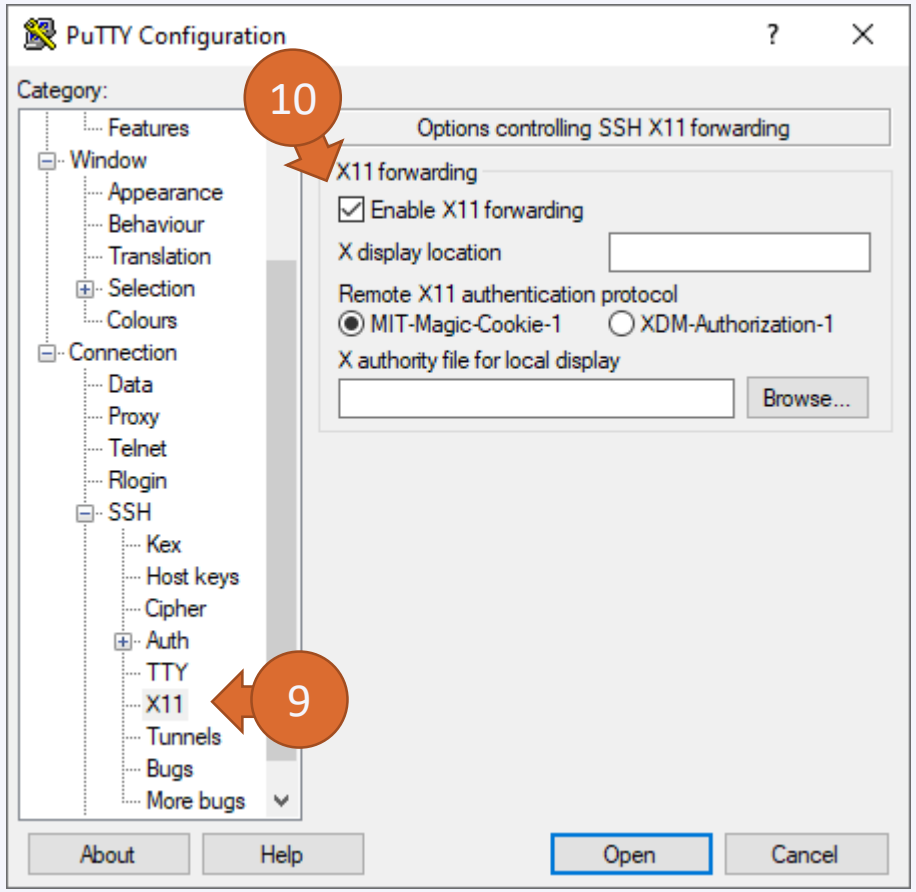




Setup PuTTY in 12 Clicks

(Windows only)

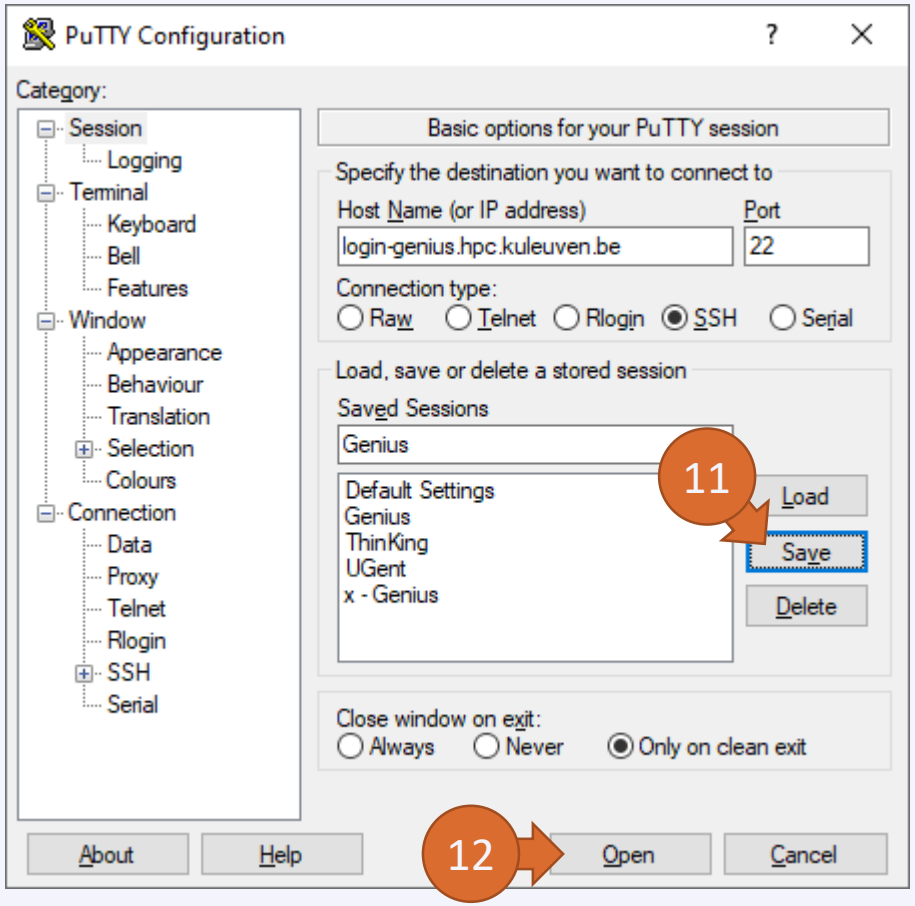


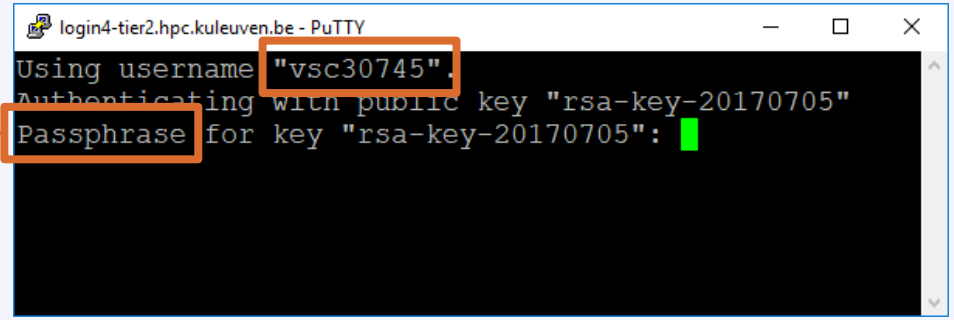


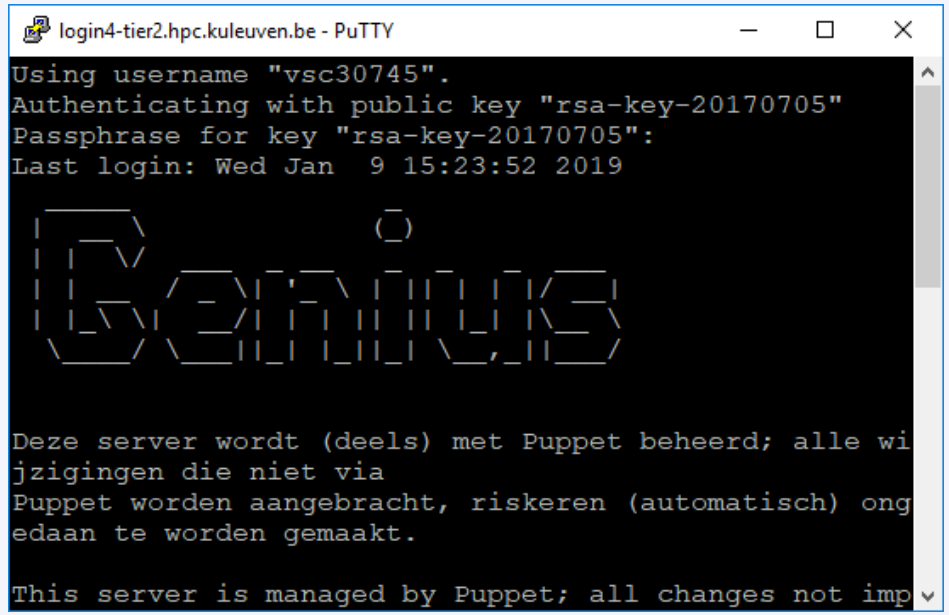
Setup PuTTY in 12 Clicks

(Windows only)

If PuTTY asks for **password**, exit immediately, and check the path to your private key. Else you will be blocked for 24h.

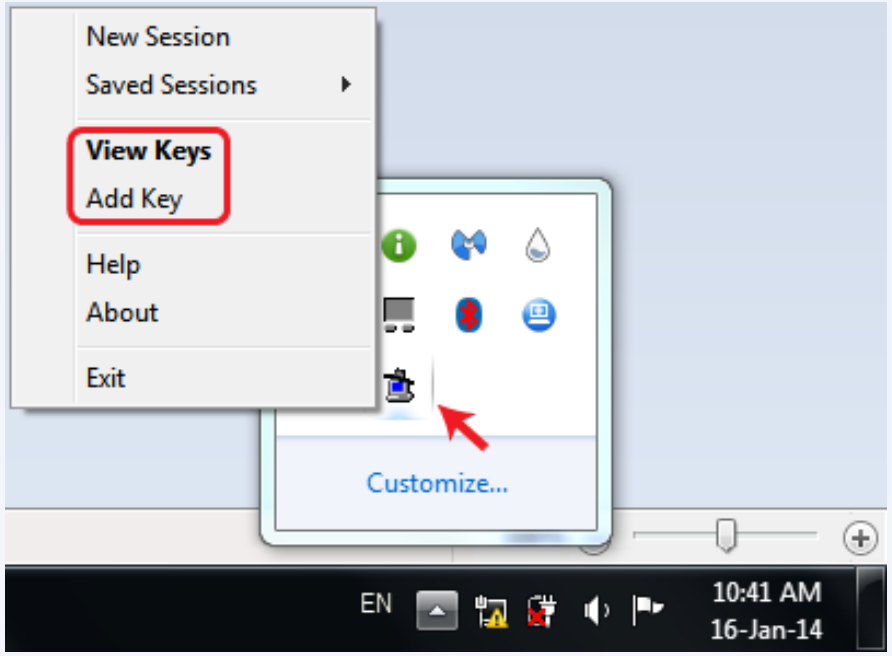






Activate Your SSH Agent
(Windows only)

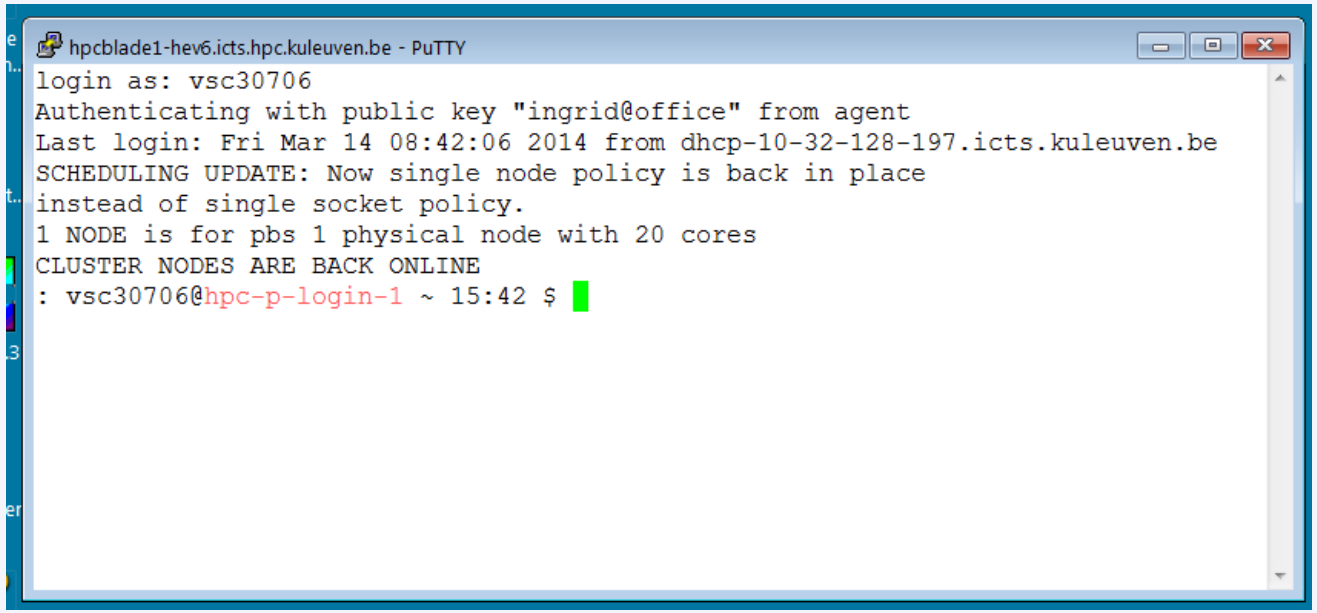
- When you install PuTTY, the agent called **Pageant** is automatically installed
- Open Pageant; it hides inside your bottom-right tray
- Click on “Add Key” and brows to your private key(s) folder
- After choosing a key, you are asked for a passphrase
- If successful, agent always remembers your key



Connecting via Terminal
(Linux and Mac)

If asked for **password**, please stop connecting and contact suport, otherwise after a few attempts you will be blocked for 24h.

- Check your SSH Agent. Is your SSH key found?
  ```
  $> ssh-add -l
  ```
- If your key is not found, add it to the agent:
  ```
  $> ssh-add </path/to/my/private/keyfile>
  ```
- Try to ssh now
  ```
  $> ssh vscXXXXX@<hostname>
  ```



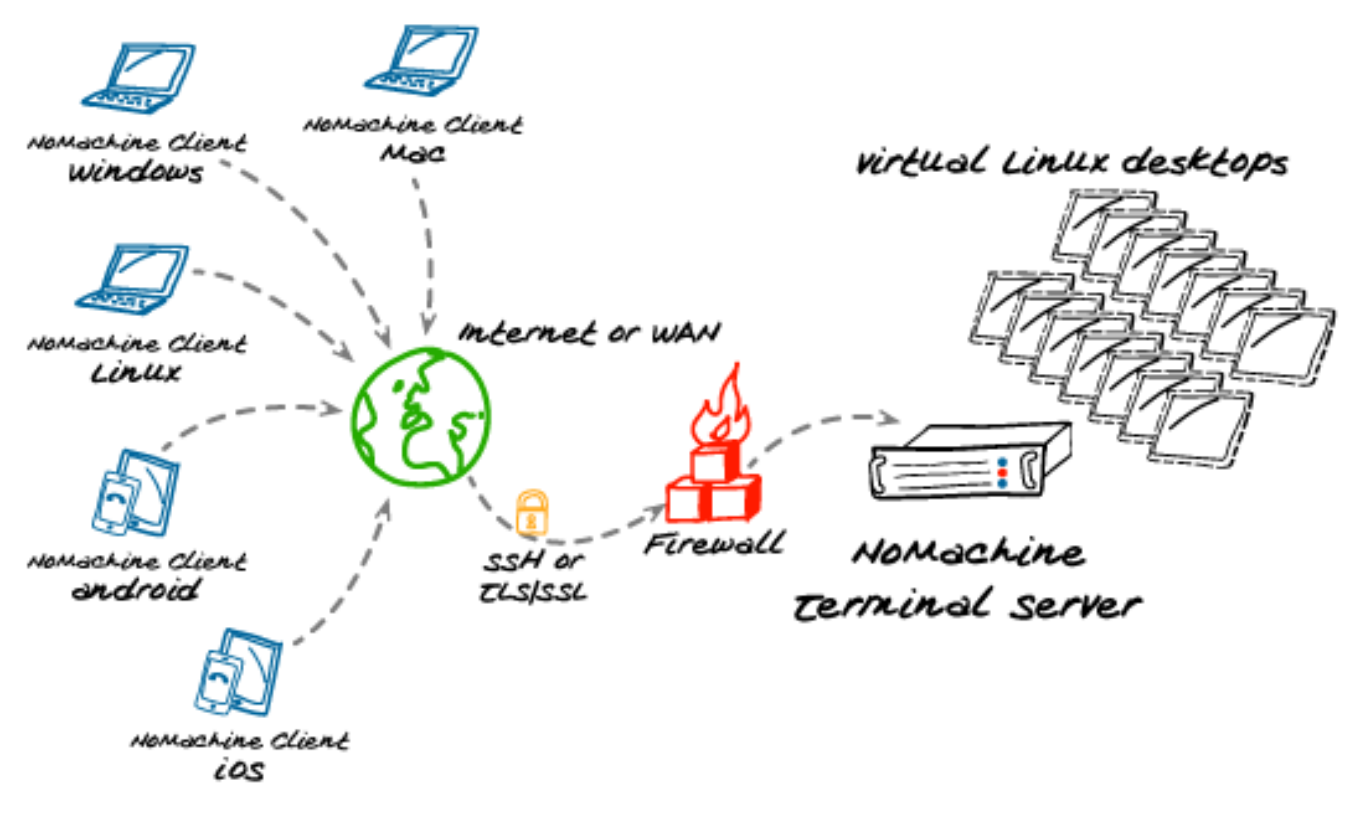
NX

- ❑ About
- ❑ Setup

NX – The Graphical Login

NX is:

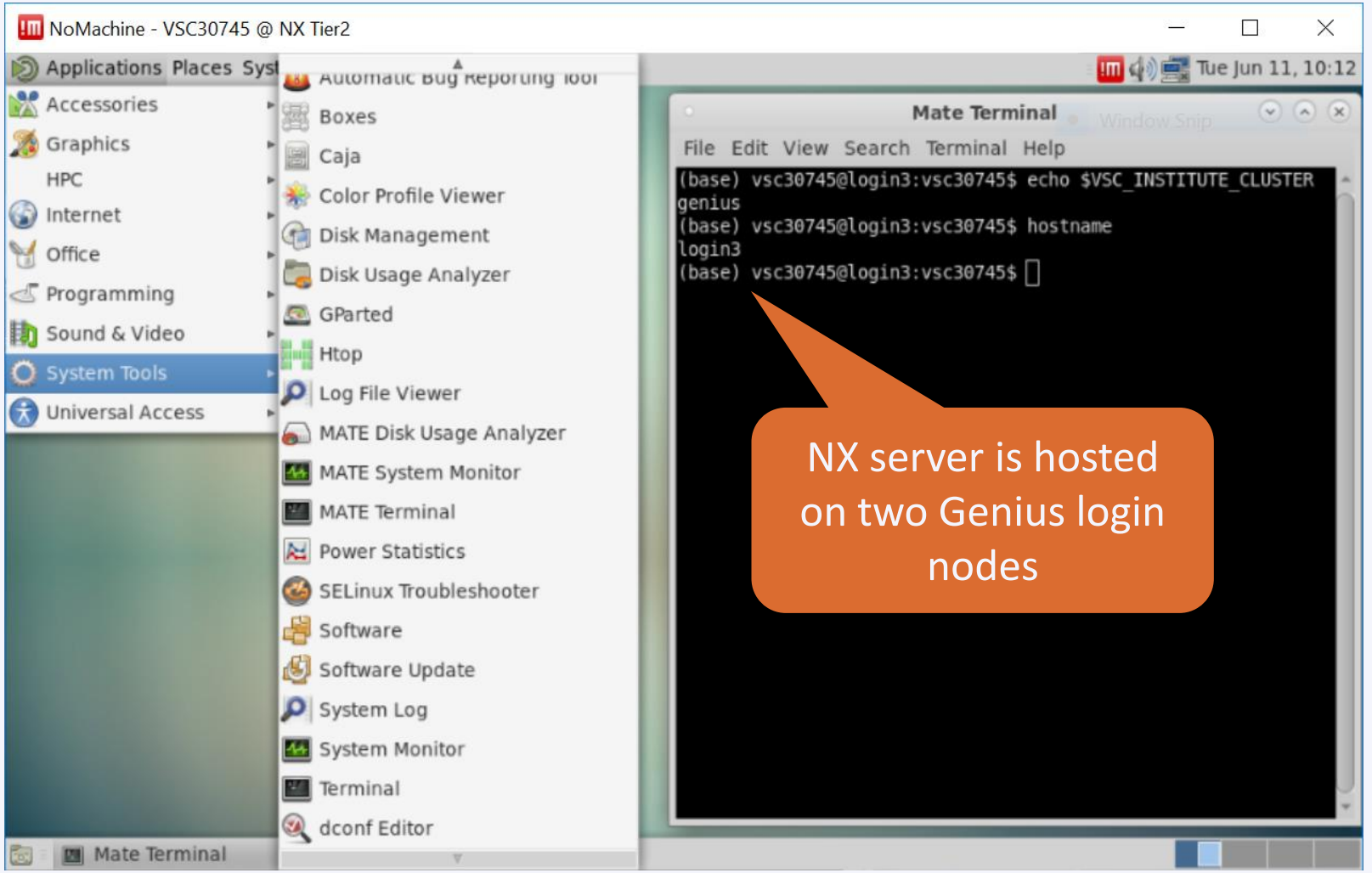
- ❑ a graphical remote desktop server
- ❑ login to VSC
- ❑ start a virtual Linux instance(s)
- ❑ Client: NoMachine
- ❑ Configuration Guide
- ❑ Using OpenSSH key
  Convert your key



Advantages of NX

- ❑ Graphical login
- ❑ A session stays alive/active even if you close your client (e.g. laptop)
- ❑ Resume a session, and immediately keep working where you left off
- ❑ More interactive jobs
- ❑ Available apps:
  - Internet browser, terminal, …
  - Text editor, PDF reader, Image viewer, …
  - Matlab, RStudio, SAS
- ❑ Remark:
  - NX is shared among tens of users
  - **Do not compute/test your code there**
  - Instead, submit jobs from NX to compute nodes

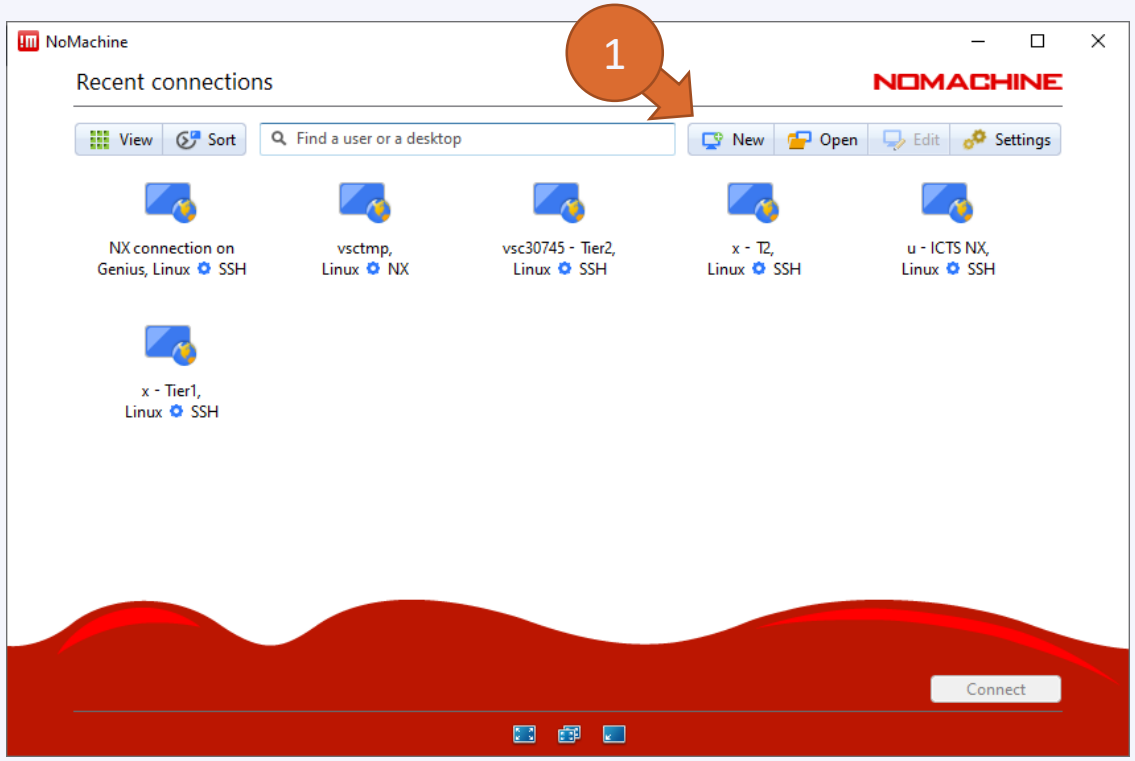
NX virtual desktop

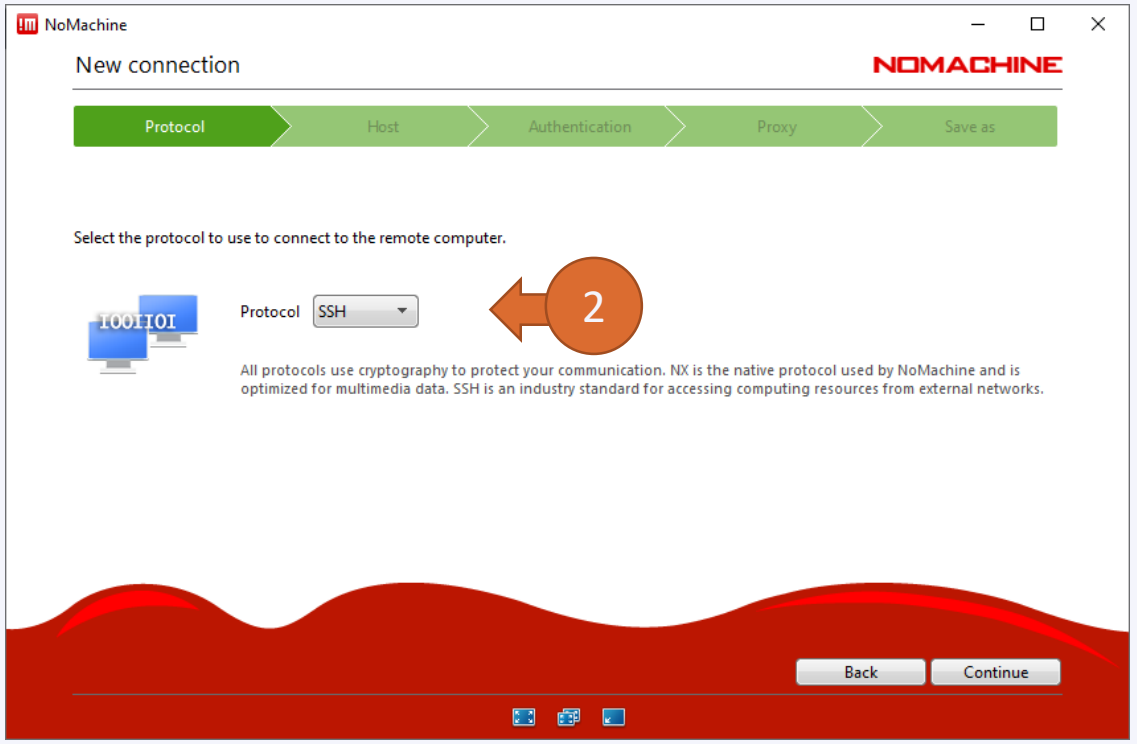


NX: available software

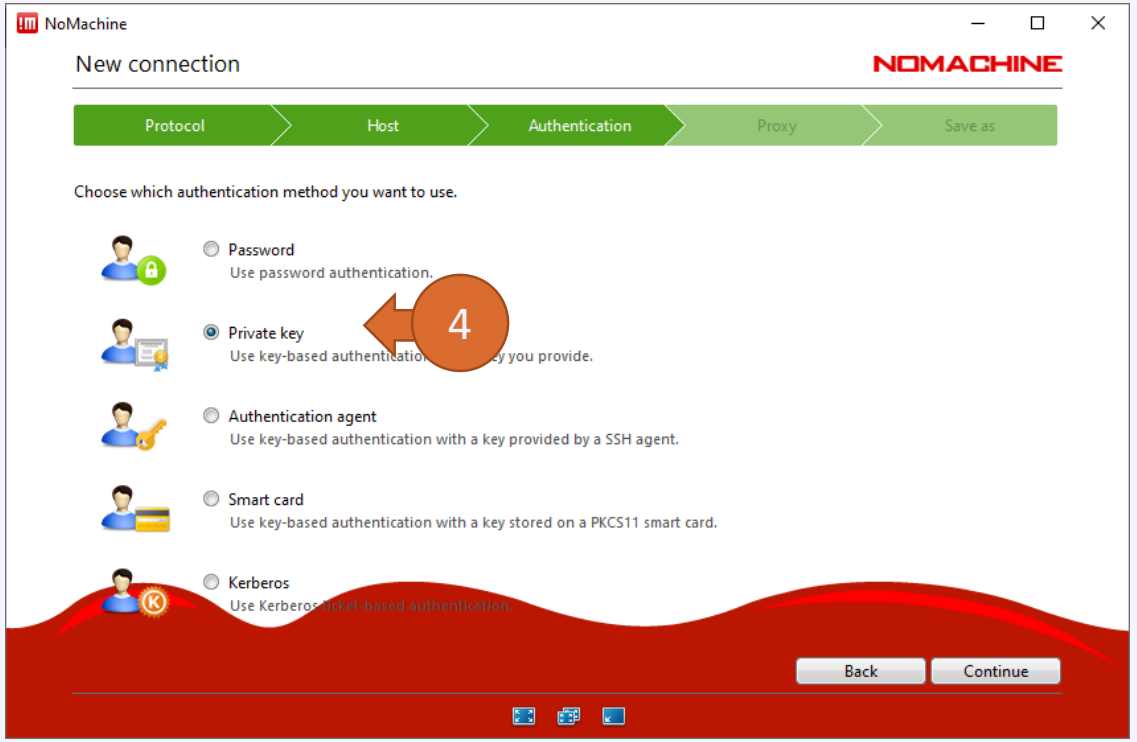
- **Accesories**: Gedit, Vi IMproved, Emacs (dummy version), Calculator
- **Graphics**: gThumb (picture viewer), Xpdf Viewer
- **Internet**: Firefox
- **HPC**: **Computation**: Matlab (2018a), RStudio, SAS
- **Visualisation**: Paraview, VisIt, VMD
- **Programming**: Meld Diff Viewer (visual diff and merge tool)
- **System tools**: File Browser, Terminal
- **Additionally**: Gnuplot (graphing utility), Filezilla (file transfer tool), Evince (PDF, PostScript, TIFF, XPS, DVI Viewer)
- Software launched though modules from Terminal.

Setup NX in 10 Steps

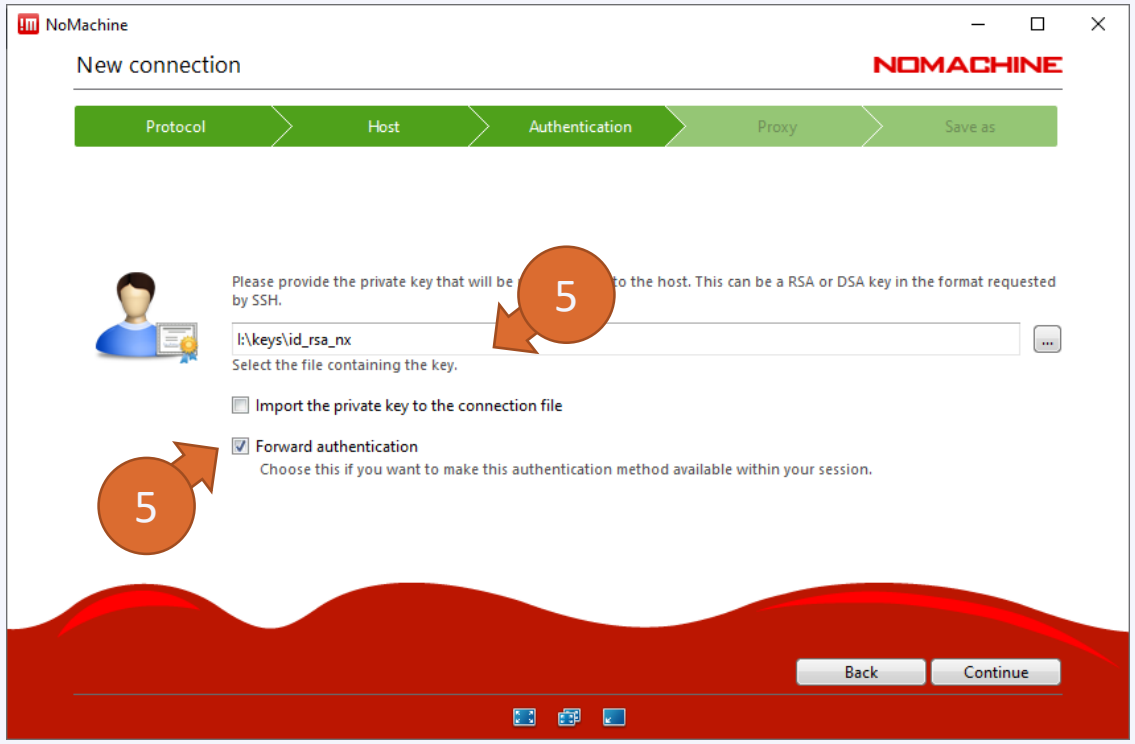


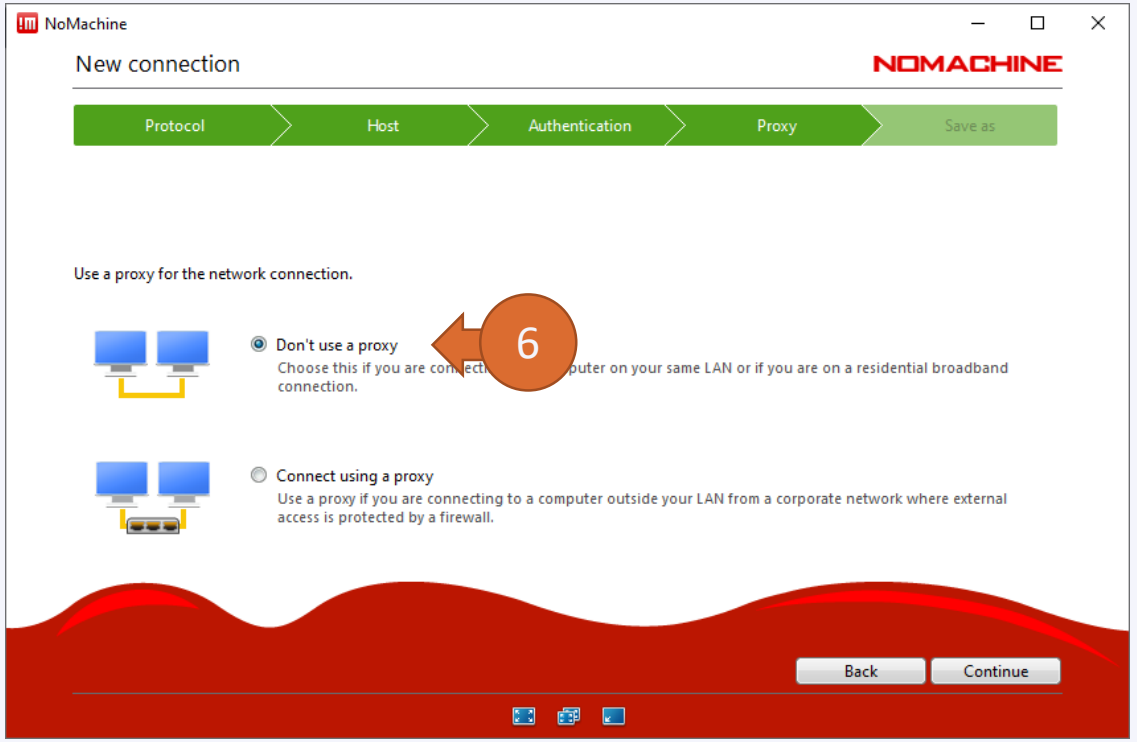


Setup NX in 10 Steps

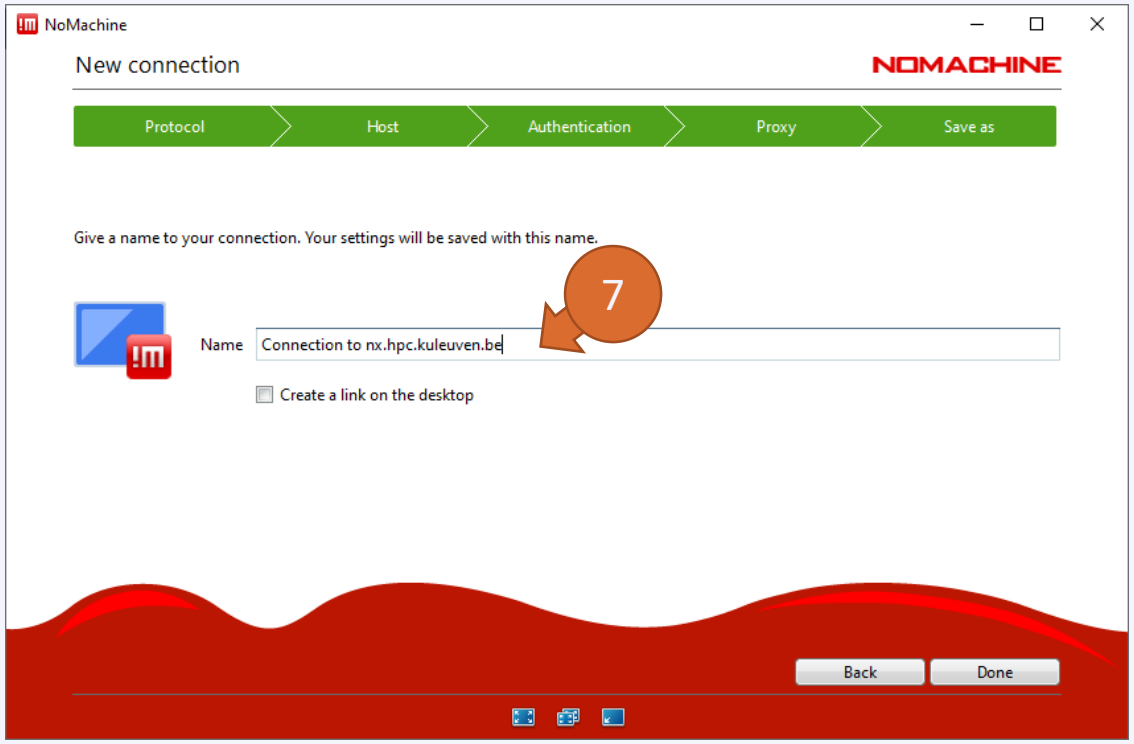


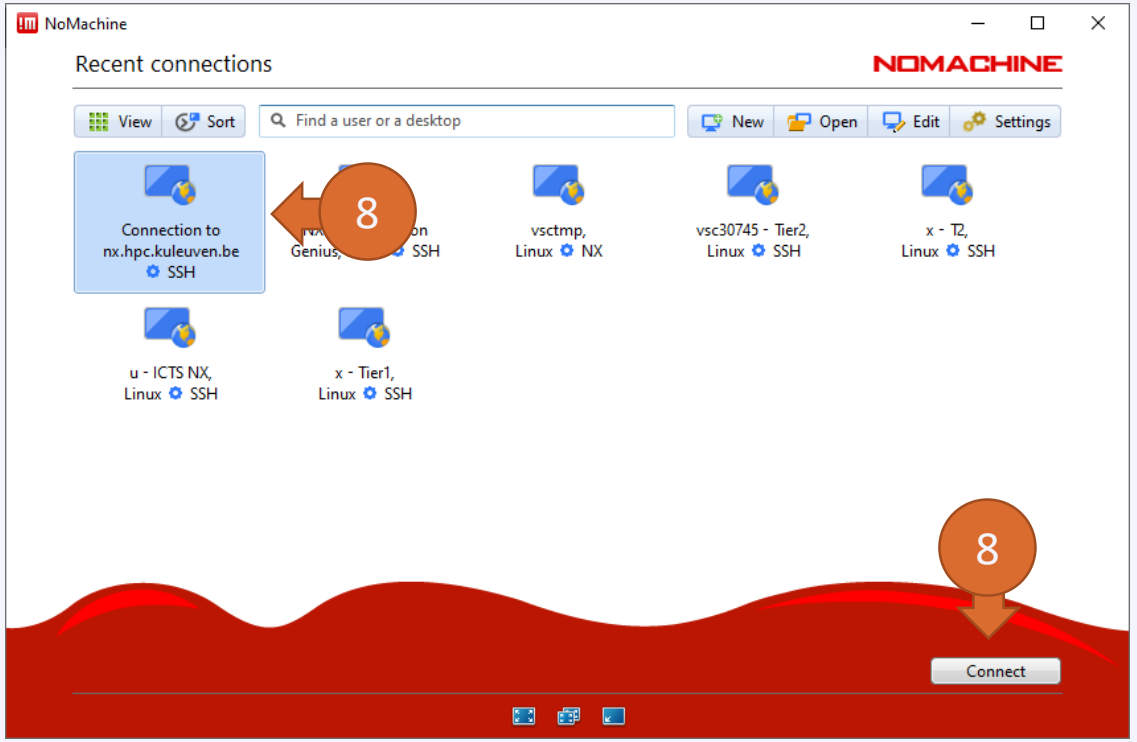
Setup NX in 10 Steps



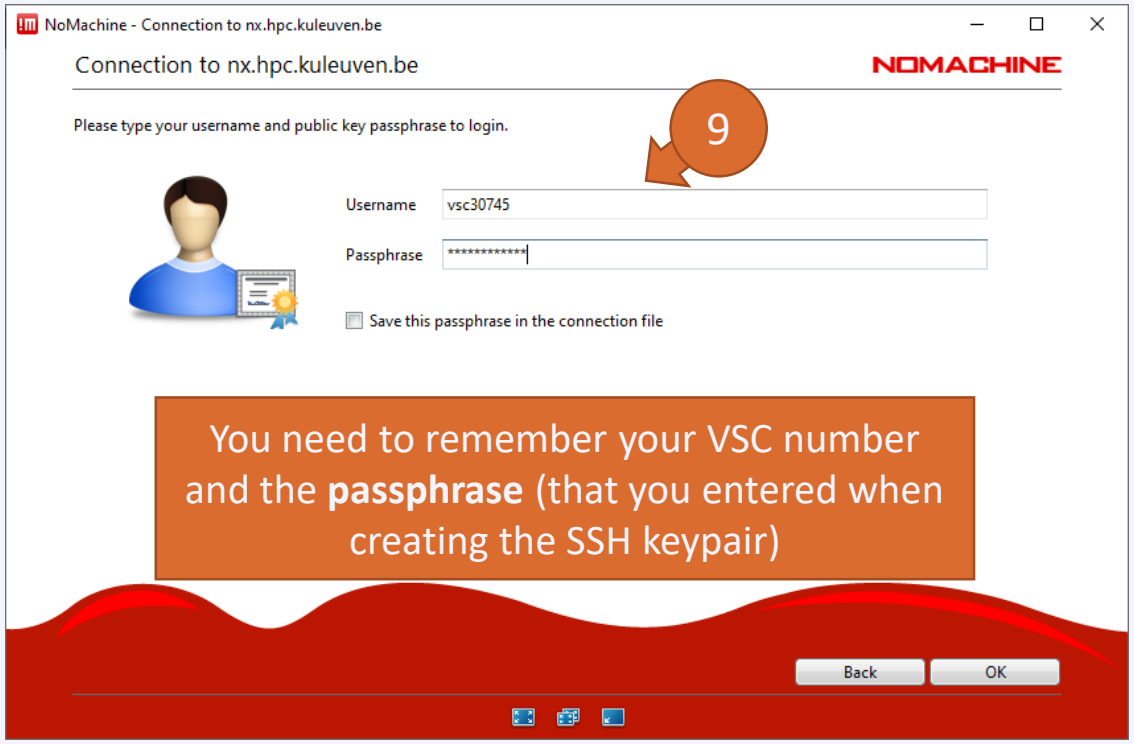


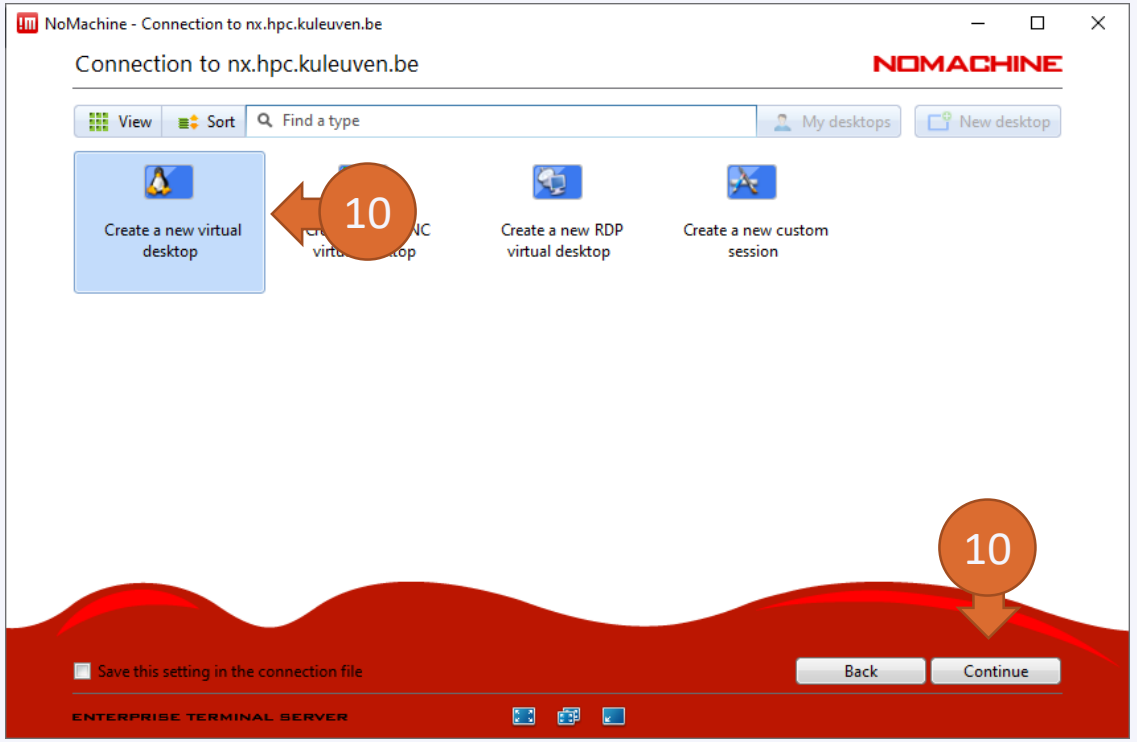
Setup NX in 10 Steps

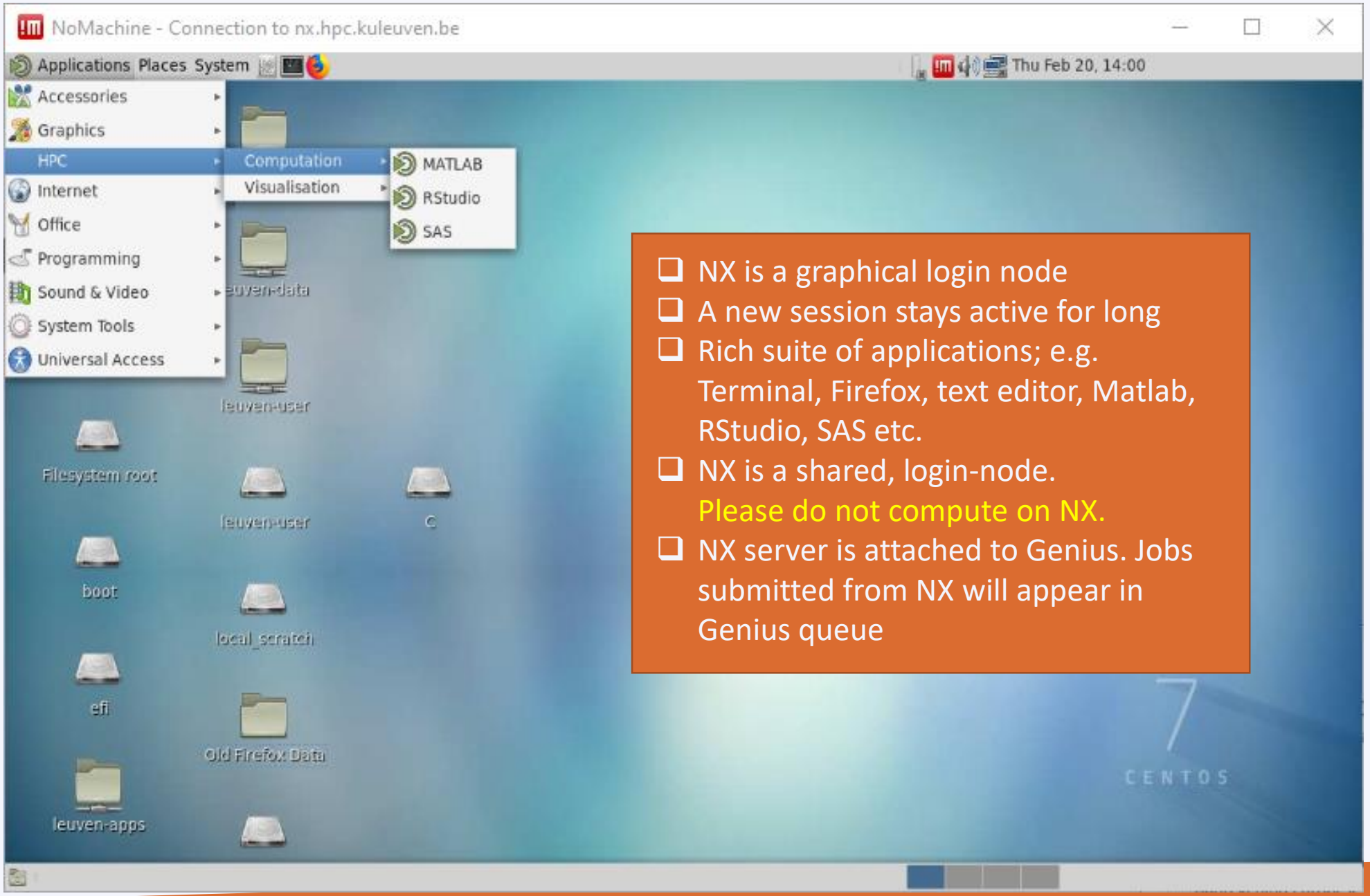




Setup NX in 10 Steps



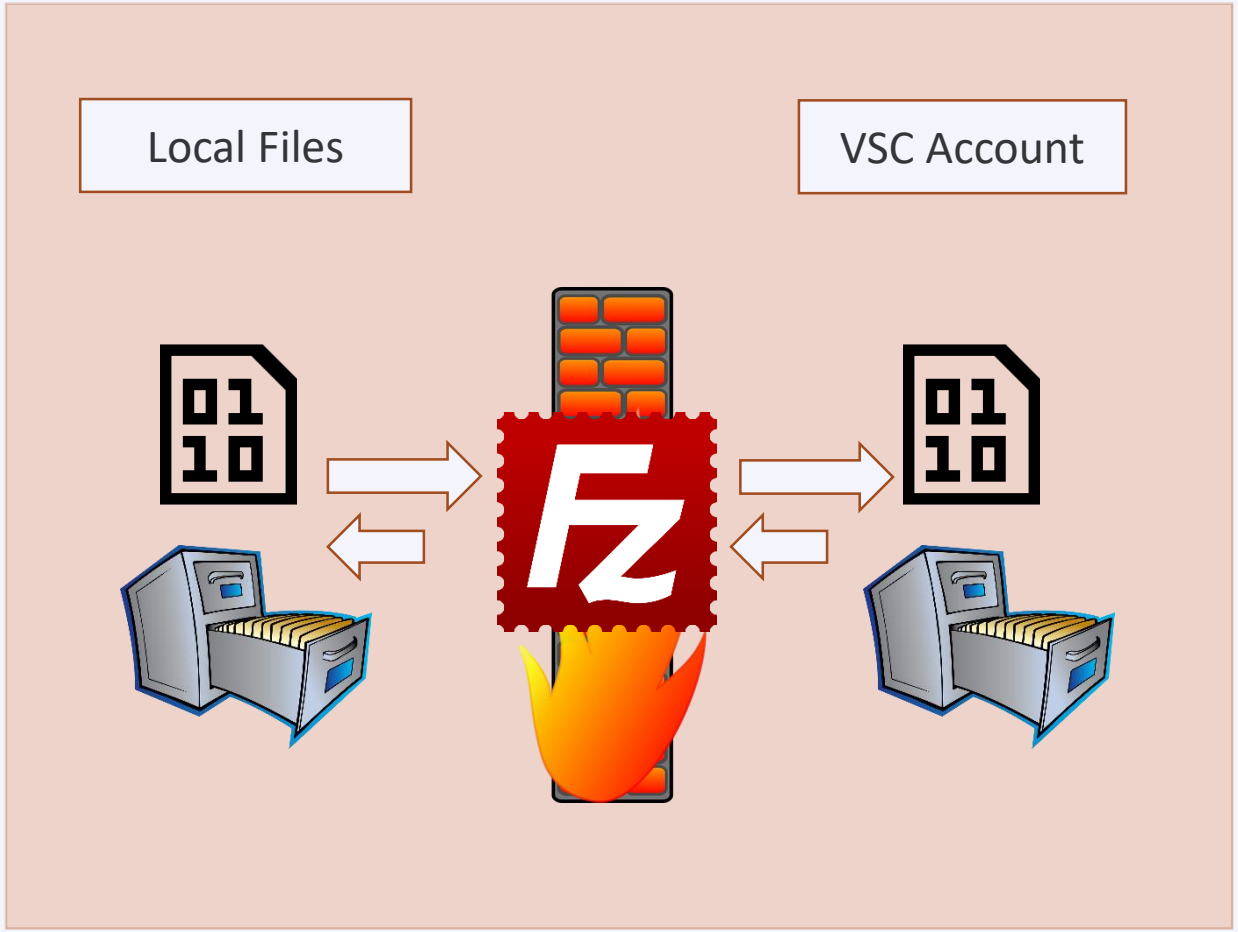




- NX is a graphical login node
- A new session stays active for long
- Rich suite of applications; e.g. Terminal, Firefox, text editor, Matlab, RStudio, SAS etc.
- NX is a shared, login-node. Please do not compute on NX.
- NX server is attached to Genius. Jobs submitted from NX will appear in Genius queue

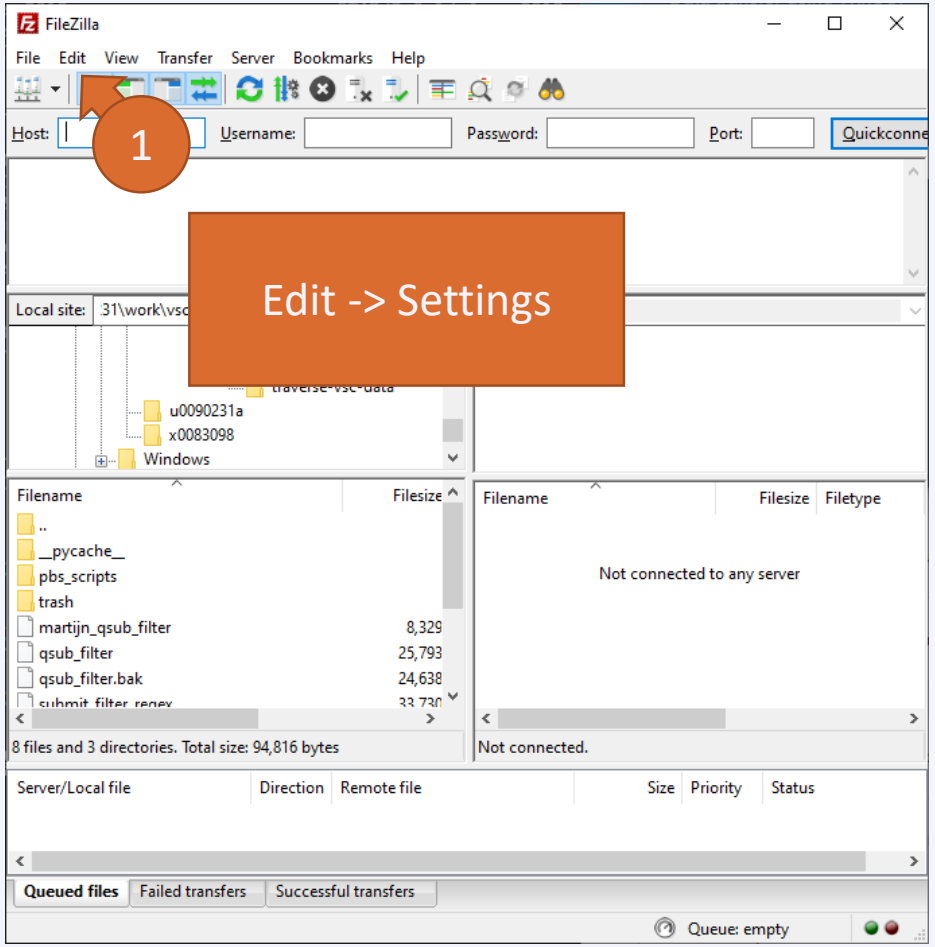
File Transfer with FileZilla

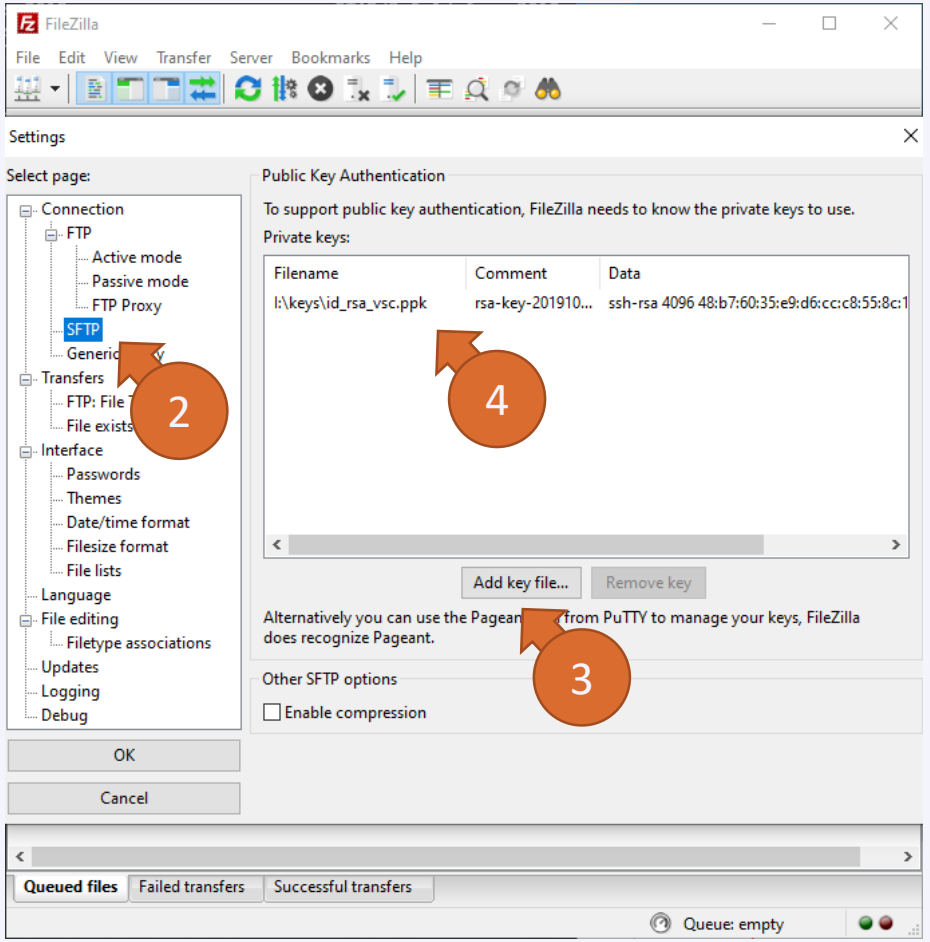
File transfer



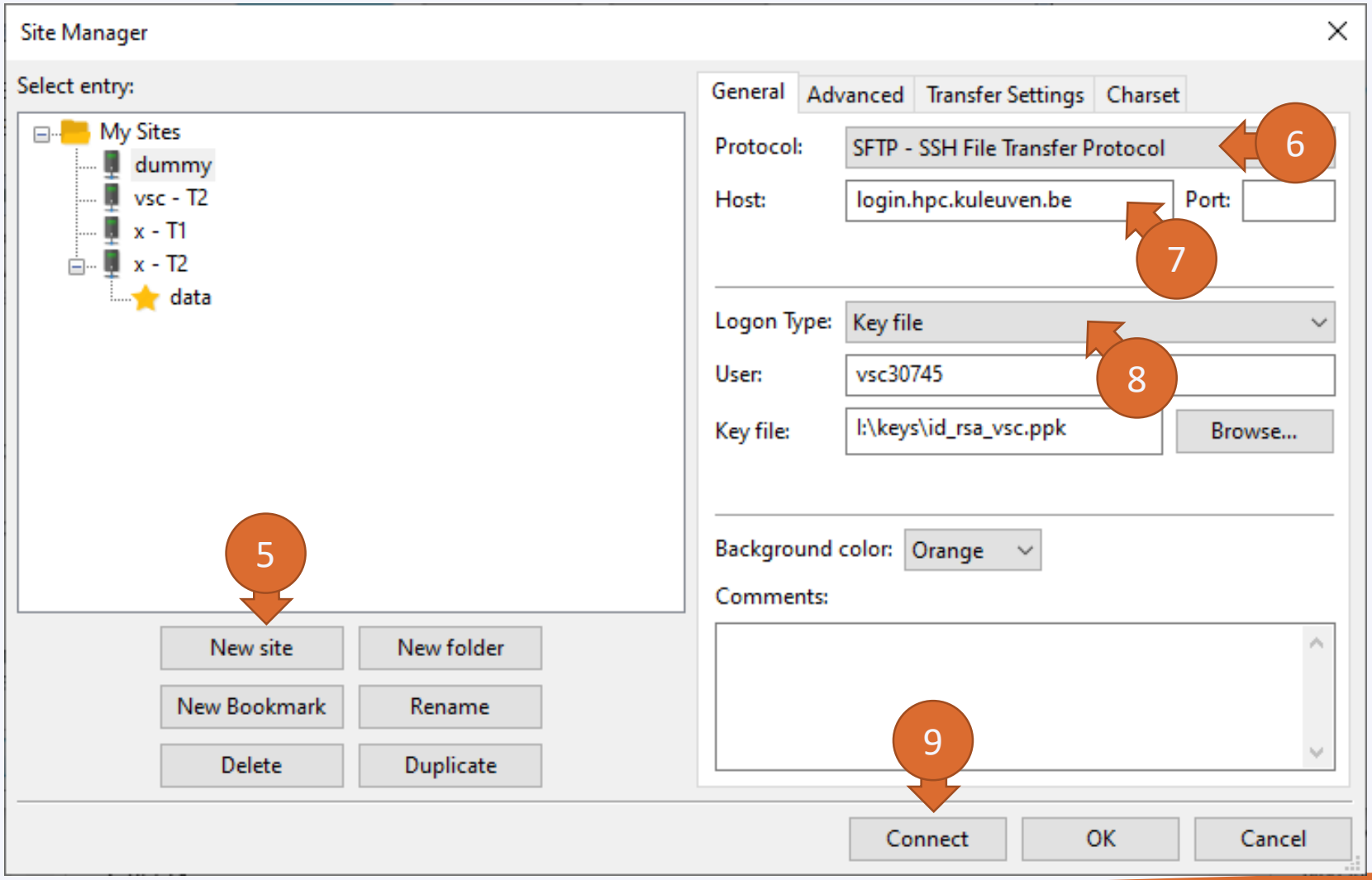
Application	OS
FileZilla	Windows, Linux, Mac
WinSCP	Windows
rsync, scp	Linux, Mac
Globus	Window, Linux, Mac

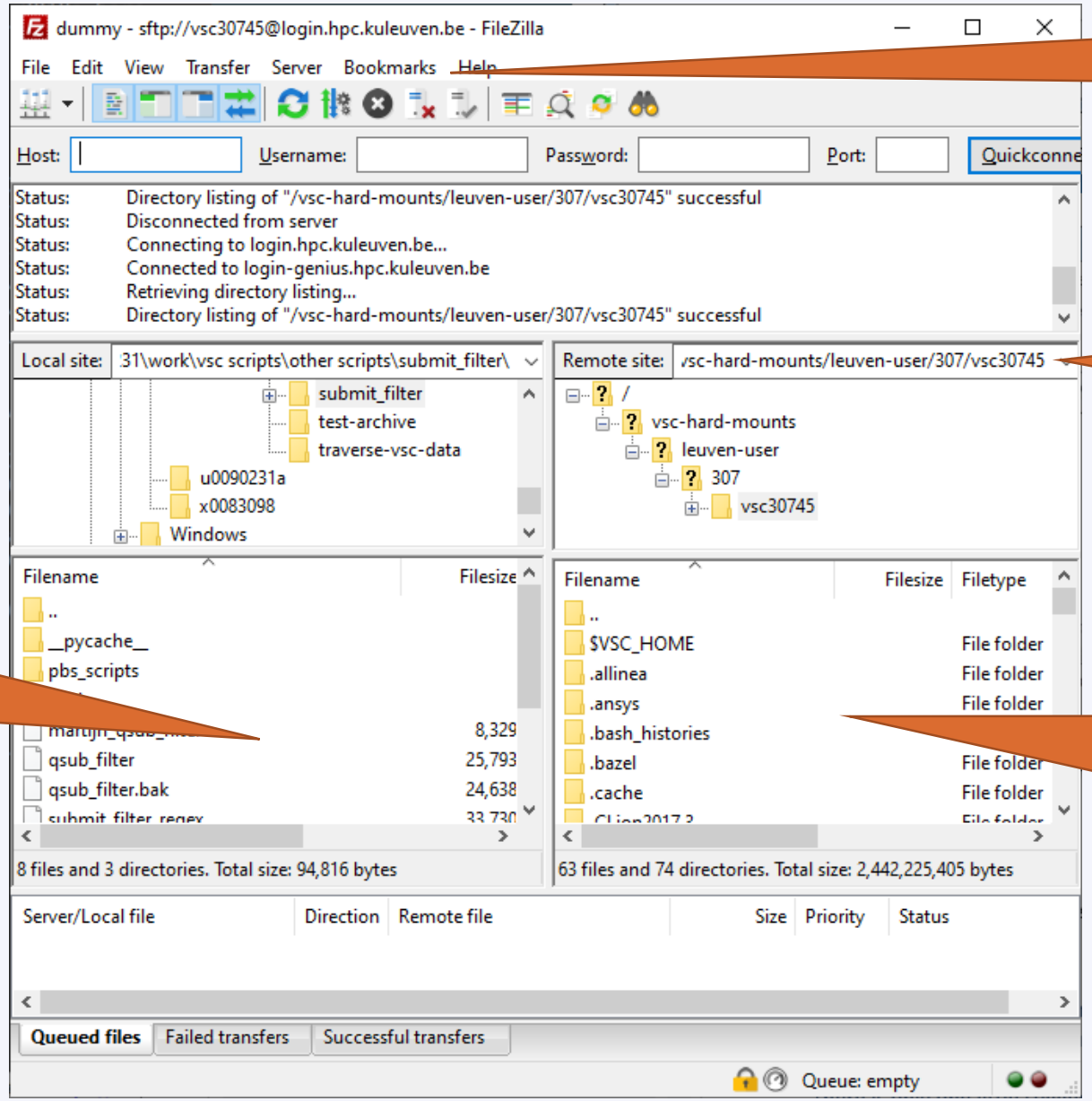
File transfer





File transfer





For convenience, you'd better bookmark your data and scratch folders!

Instead, go to your `$VSC_DATA` folder, e.g. /data/leuven/399/vsc39934

Your local machine (e.g. laptop)

Your VSC storage (e.g. `$VSC_DATA`, `$VSC_SCRATCH`). **Note:** by default you arrive at your own `$VSC_HOME`. Copy nothing there!

Software Stack

Software: Available Modules

- OS: Linux CentOS 7.7, Kernel 3.10.0-1127.18.2.el7.x86_64
- Compilers:
  Intel 2018a (icc, icpc, ifort; Intel MPI; Intel MKL)
  FOSS 2018a (gcc, g++, gfortran; OpenMPI; ScaLAPACK, OpenBLAS, FFTW)
- Note: **Never mix** FOSS and Intel compilers (gives dependency conflict)

Command	Remark
`module av`	List all installed modules
`module av Python`	List all Python-related modules
`module load Python/3.6.4-intel-2018a`	Load a specific module
`module list`	List all loaded modules and their dependencies
`module unload Python/3.6.4-intel-2018a`	Unload a module (but dependencies still stay)
`module purge`	Remove all modules from your work session

Software: Your Specific Needs

- You can always install your desired software in your $VSC_DATA
  Use Intel or FOSS toolchains
- If you cannot, ask us for help
- Python/R packages for AI and ML must be installed by the users themselves
- Install miniconda in your $VSC_DATA

```
$> wget https://repo.continuum.io/miniconda/Miniconda3-latest-
Linux-x86_64.sh
$> bash Miniconda3-latest-Linux-x86_64.sh -p $VSC_DATA/miniconda3
```

- Create conda environment per every project

```
$> conda create –n EnvTF tensorflow-gpu=2.0.0 pytorch
```

- Use your own conda/Python in your jobs/scripts

```
$> conda activate EnvTF
```

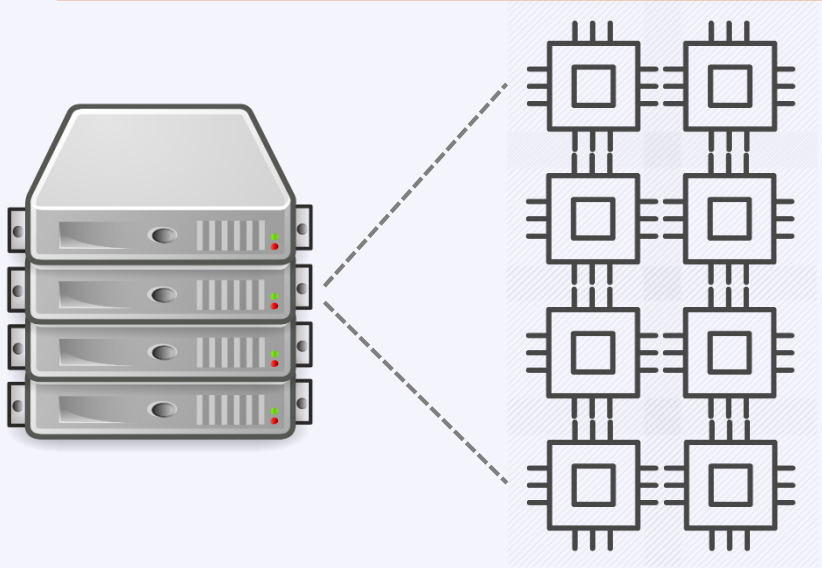
Starting to Compute

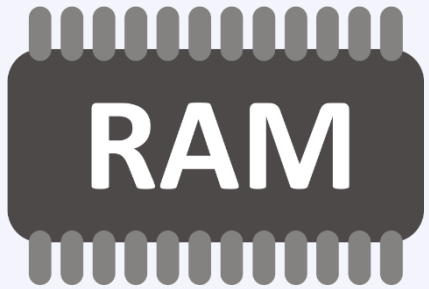
Extra services

- Request introduction credits
  https://admin.kuleuven.be/icts/onderzoek/hpc/request-introduction-credits
- Request project credits (create your group first)
  https://admin.kuleuven.be/icts/onderzoek/hpc/request-project-credits
  https://icts.kuleuven.be/sc/forms/Aanvraagformulier_HPC_Credits
- Extra project credits (to add to existing project)
  https://admin.kuleuven.be/icts/onderzoek/hpc/extra-project-credits
- Request extra storage
- https://admin.kuleuven.be/icts/onderzoek/hpc/hpc-storage

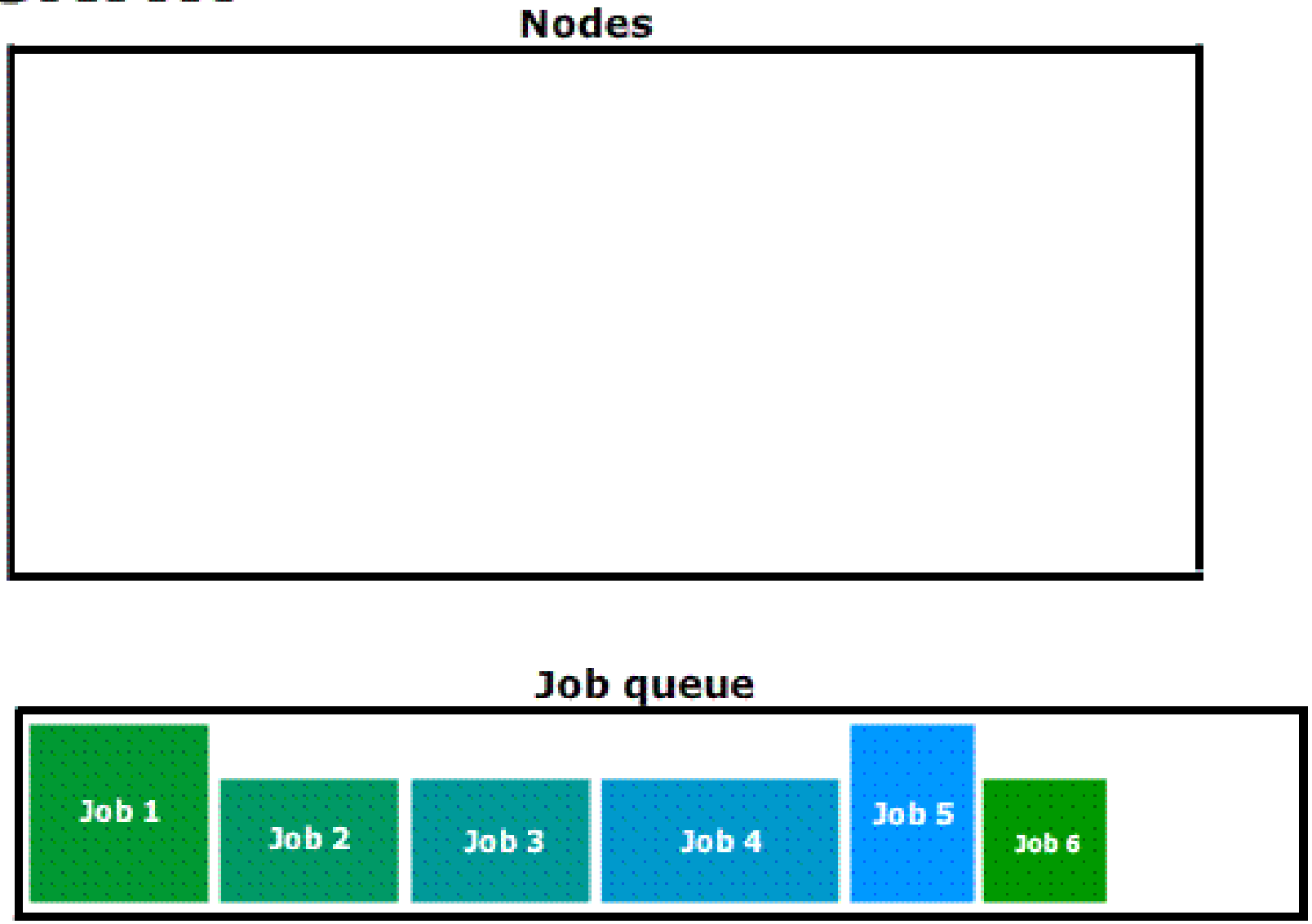
Resource Glossary

- ❑ **Nodes**: how many compute servers to request?
- ❑ **Cores**: how many cores per node to use?
- ❑ **Memory requirement**: how much memory each core needs?
- ❑ **Partition**: GPU, BigMem, Superdome, AMD
- ❑ **Walltime**: how long to use resources?





Backfill

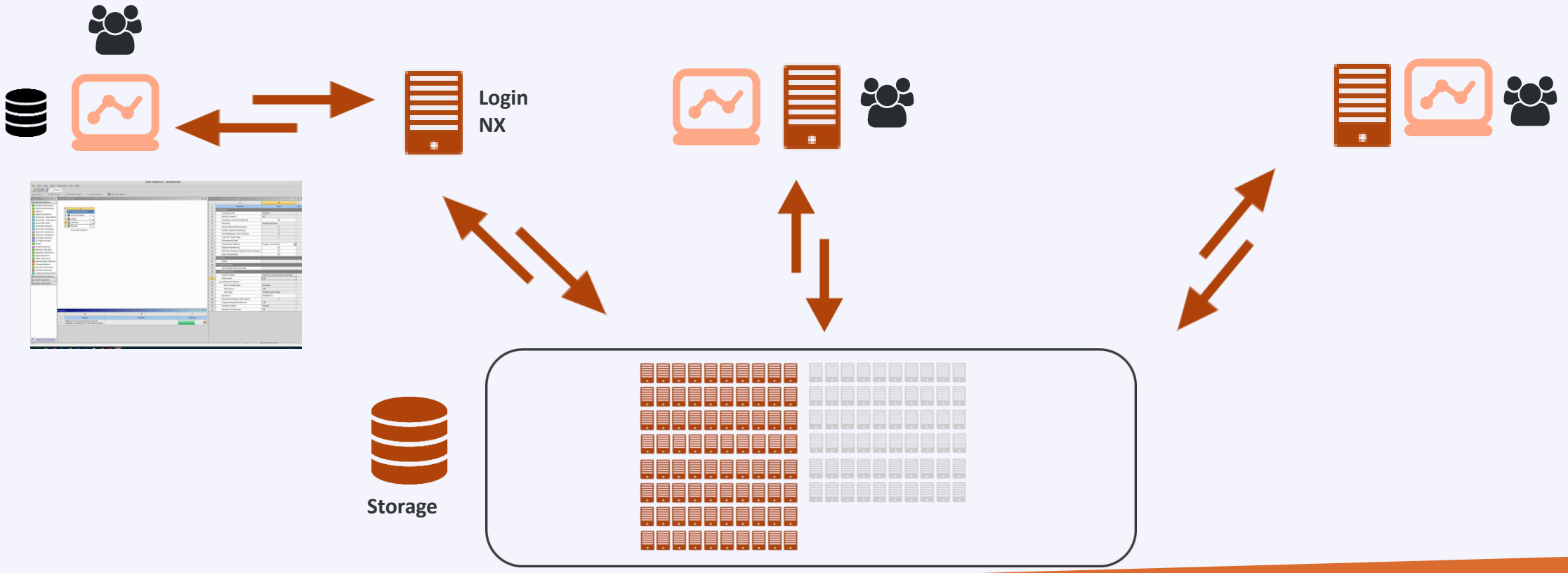


Ways to use the cluster

INTERACTIVE GUI WORKLOAD

BATCH WORKLOAD

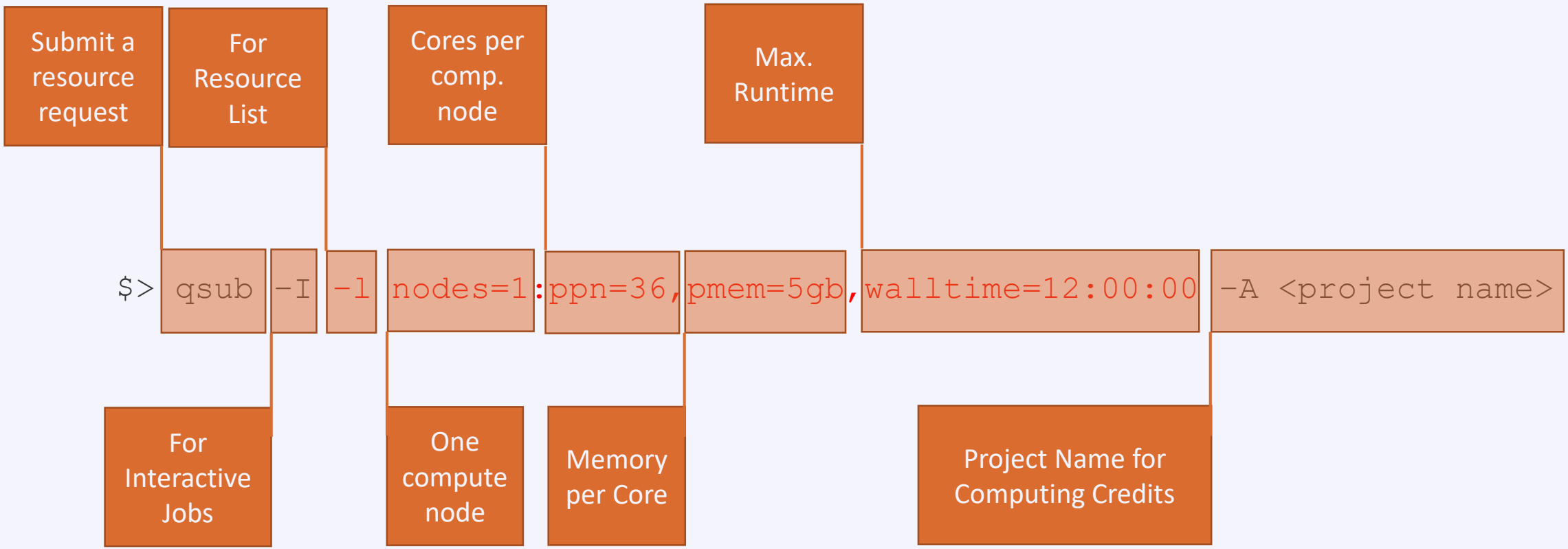
INTERACTIVE WORKLOAD



Types of Jobs

1) **Batch jobs** are by far the most common and allow for the most efficient use of the infrastructure. Essentially, a batch job is a bash shell script that is executed on a compute node, and that can spawn a parallel computation on many nodes. These jobs are placed in the queue, and the user can forget about it until it is finished.
2) **Interactive jobs** are intended to work on one or more compute nodes interactively. This can be useful in the context of software development for debugging and profiling applications, or for interactive calculations or visualizations. Basically, one gets a shell on one of the compute nodes.
3) To get GPU(s), you can use **JupyterHub** with your internet browser

Requesting Compute Resources Interactively



Remarks

- Specifying <project name> for credits is mandatory, e.g.: **`-A lp_hpcinfo_training`**
- If you request 2000 introductory credits, you use them like: **`-A default_project`**
- Implicit defaults are `nodes=1:ppn=1,pmem=5gb,walltime=1:00:00`

Interactive Jobs

- Interactive job: 1 core for 1 hour (default)

```
$> qsub -I -A lp_hpc
```

- Interactive job with X-forwarding

```
$> qsub -I -X -A lp_hpc
```

- Ask 2 SkyLake nodes, 36 cores each, 4 GB RAM per core for 12 hours

```
$> qsub -I -l nodes=2:ppn=36:skylake,pmem=4gb,walltime=12:00:00 -A lp_hpc
```

- Ask 2 CascadeLake nodes, 36 cores each, 4GB RAM per core for 30 min

```
$> qsub -I -l nodes=2:ppn=36:cascadelake,pmem=4gb,walltime=30:00 -A lp_hpc
```

- Request fraction of a node

```
$> qsub -I -l nodes=1:ppn=8:cascadelake -A lp_hpc
```

Batch Workload

- Create a new text (ASCII) file
- Give it a good name!
  E.g. `simulation.pbs`
- The first line is the shebang!
  `#!/bin/bash`
- Put all commands line by line, e.g.
  `echo "Hello World!"`
- Submit the job to the batch server
- Receive a unique JobID
- Error and output files

Job Script

```
#!/bin/bash
echo "Hello World!"
```

Command Line

```
$ qsub simulation.pbs -A default_project
```

JobID

```
50041238.tier2-p-
moab2.tier2.hpc.kuleuven.be
```

stderr, stdout

```
$ ls simulation*
simulation.pbs
simulation.pbs.e50041238
simulation.pbs.o50041238
```

Standard Error File

- Always created
- <JobScript>.**e**<JobID>
- Contains all errors and warnings
- If empty: everything went well
- Always study it
- Addess all warnings and errors (if you can)
- Typical error examples …

stderr

```
$ ls *.e*
simulation.pbs.e50041238
```

Job Crash

```
forrtl: error (78): process killed (SIGTERM)
Stack trace terminated abnormally.
```

Short Walltime

```
PBS: job killed: walltime 432031 exceeded
limit 432000
```

Low Disk Space

```
IOError: [Errno 122] Disk quota exceeded
```

Standard Output File

- Always created
- <JobScript>.o<JobID>
- Contains all standard output (instead of screen)
- Always study it

stdout

```
$ ls *.o*
simulation.pbs.o50041238
```

Output File

```
time: 3600
nodes: 1
procs: 1
account string: lpt2_sysadmin
queue: q1h
============================
Hello World!
============================
Date: Fri Mar 20 14:32:01 CET 2020
Allocated nodes:
r26i27n11
Job ID: 50240161.tier2-p-moab-2.tier2.hpc. …
Resource List:
pmem=5gb,walltime=01:00:00,nednodes=1:ppn=36
Resources Used:
cput=00:00:01,vmem=0kb,walltime=00:00:04,mem=0kb,e
nergy_used=0
Queue Name: q1h
----------------------------
time: 4
nodes: 1
procs: 1
account: lpt2_sysadmin
```

Summary

stdout

Resources

summary

Example: PBS Job Script

```
#!/bin/bash
#PBS -l walltime=00:10:00
#PBS -l nodes=1:ppn=36
#PBS -l pmem=5gb
#PBS -N testjob
#PBS -A lp_hpcinfo_training
#PBS -m abe
#PBS -M my.name@kuleuven.be

module load intel/2018a
which icc

cd $PBS_O_WORKDIR
cp /apps/leuven/training/HPC-intro/cpujob.pbs $VSC_SCRATCH
touch output.log
echo I am done
```

Shebang

Resource List

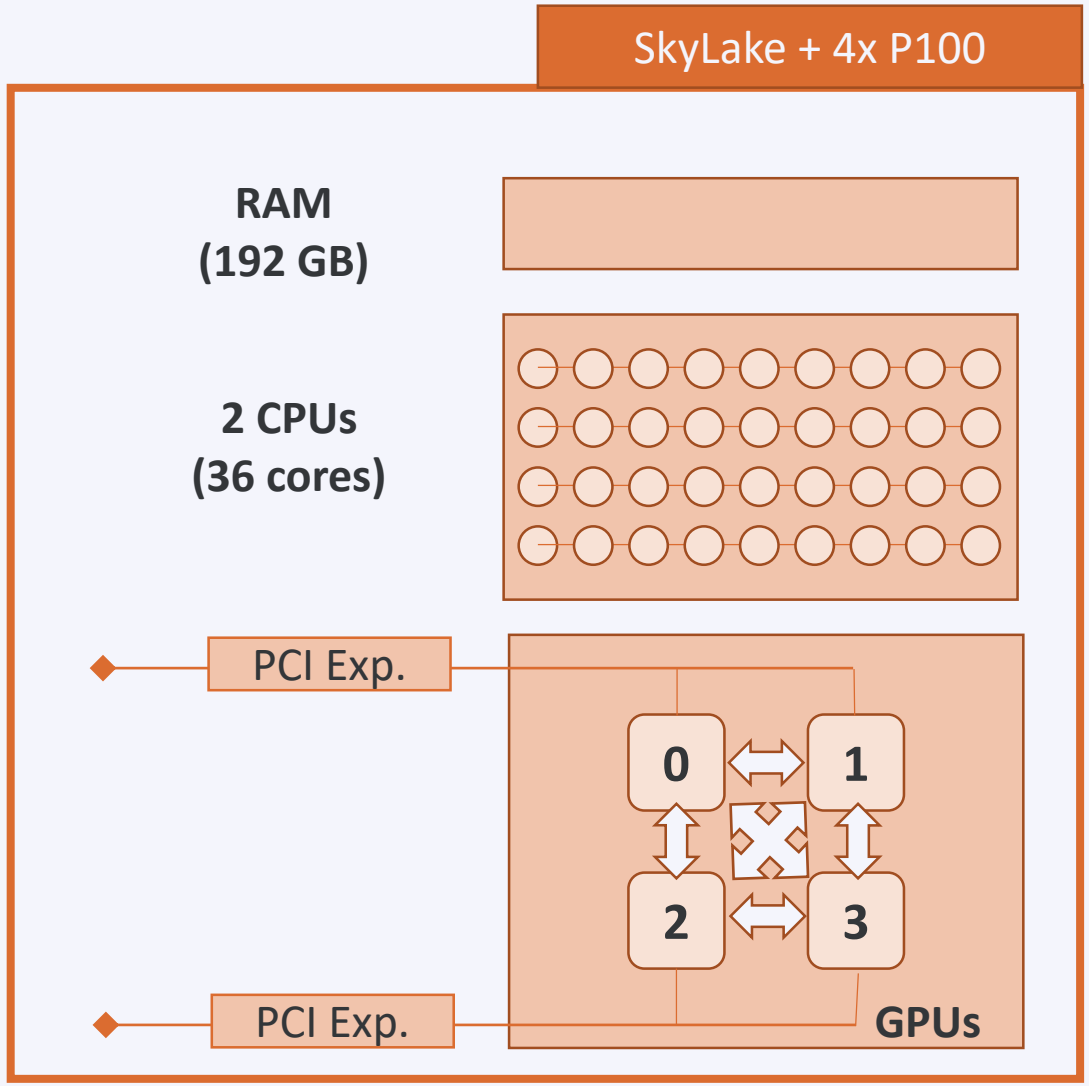
Module load(s)

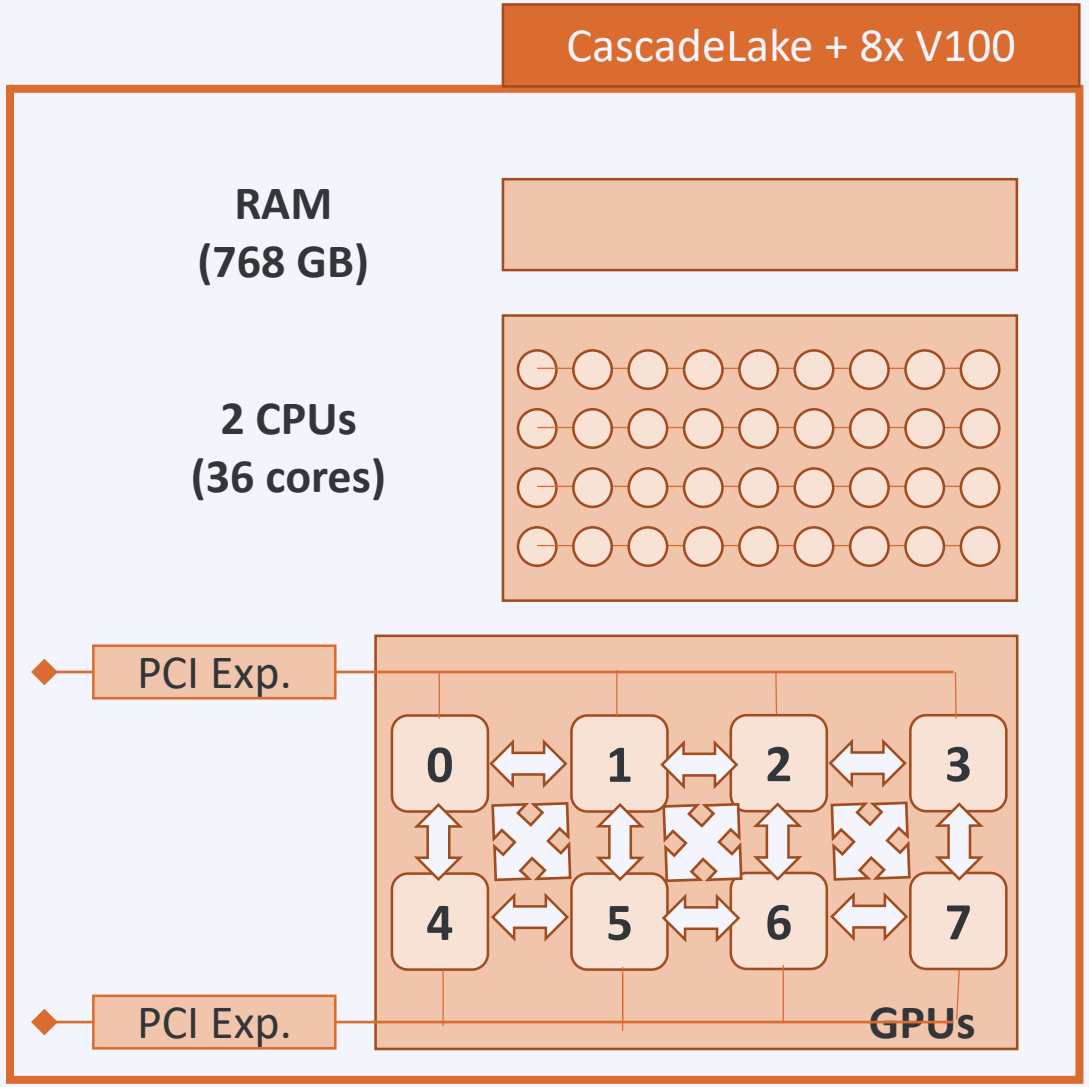
Move data

Execure commands

GPU Partition

GPU Nodes


SkyLake + 4x P100
RAM
(192 GB)
2 CPUs
(36 cores)
PCI Exp.
PCI Exp.
0
1
2
3
GPUs

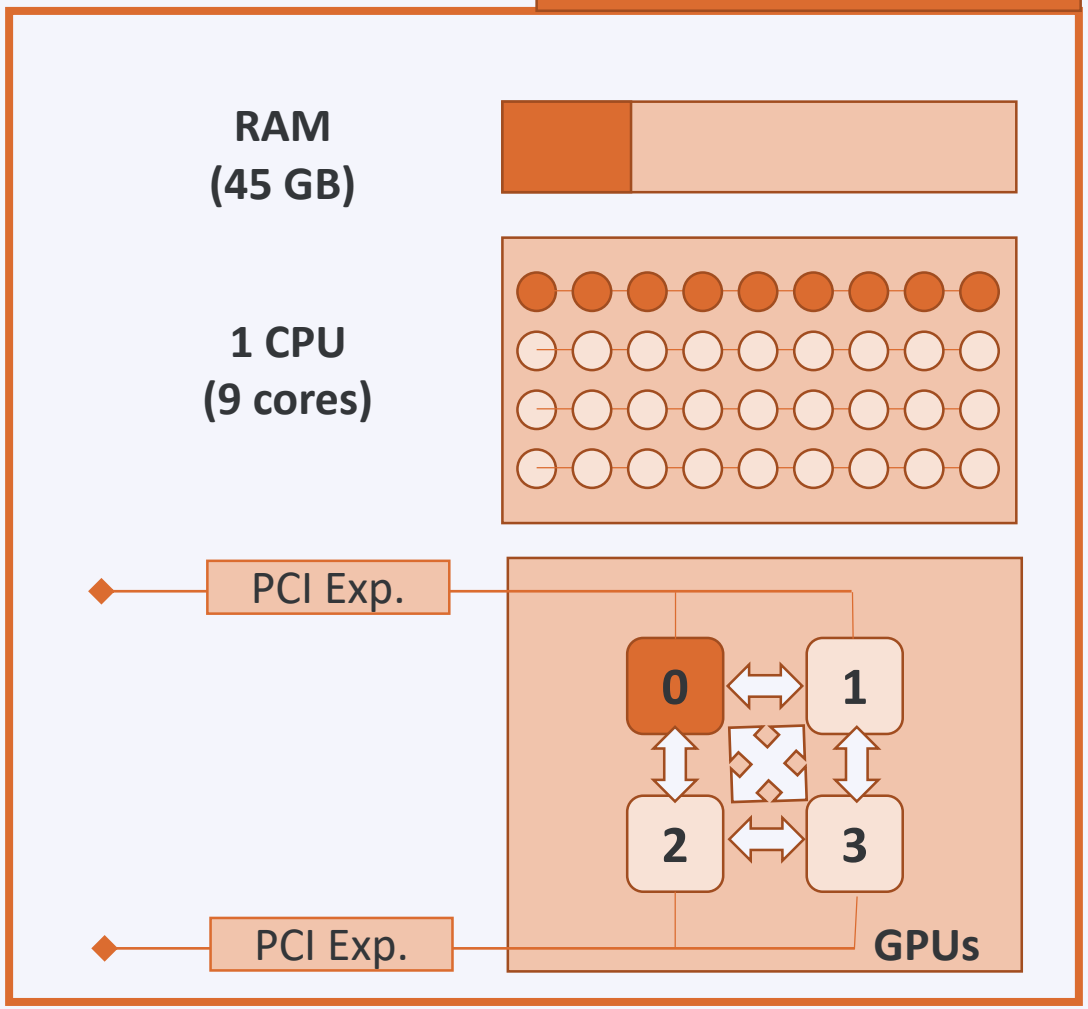

CascadeLake + 8x V100
RAM
(768 GB)
2 CPUs
(36 cores)
PCI Exp.
PCI Exp.
0
1
2
3
4
5
6
7
GPUs

Using P100 GPU(s)

Using 1 GPU

```
#!/bin/bash -l
#PBS -l walltime=00:30:00
#PBS -l partition=gpu
#PBS -l nodes=1:ppn=9:gpus=1:skylake
#PBS -l pmem=5gb
#PBS -N testjob
#PBS -A default_project
```

You get …

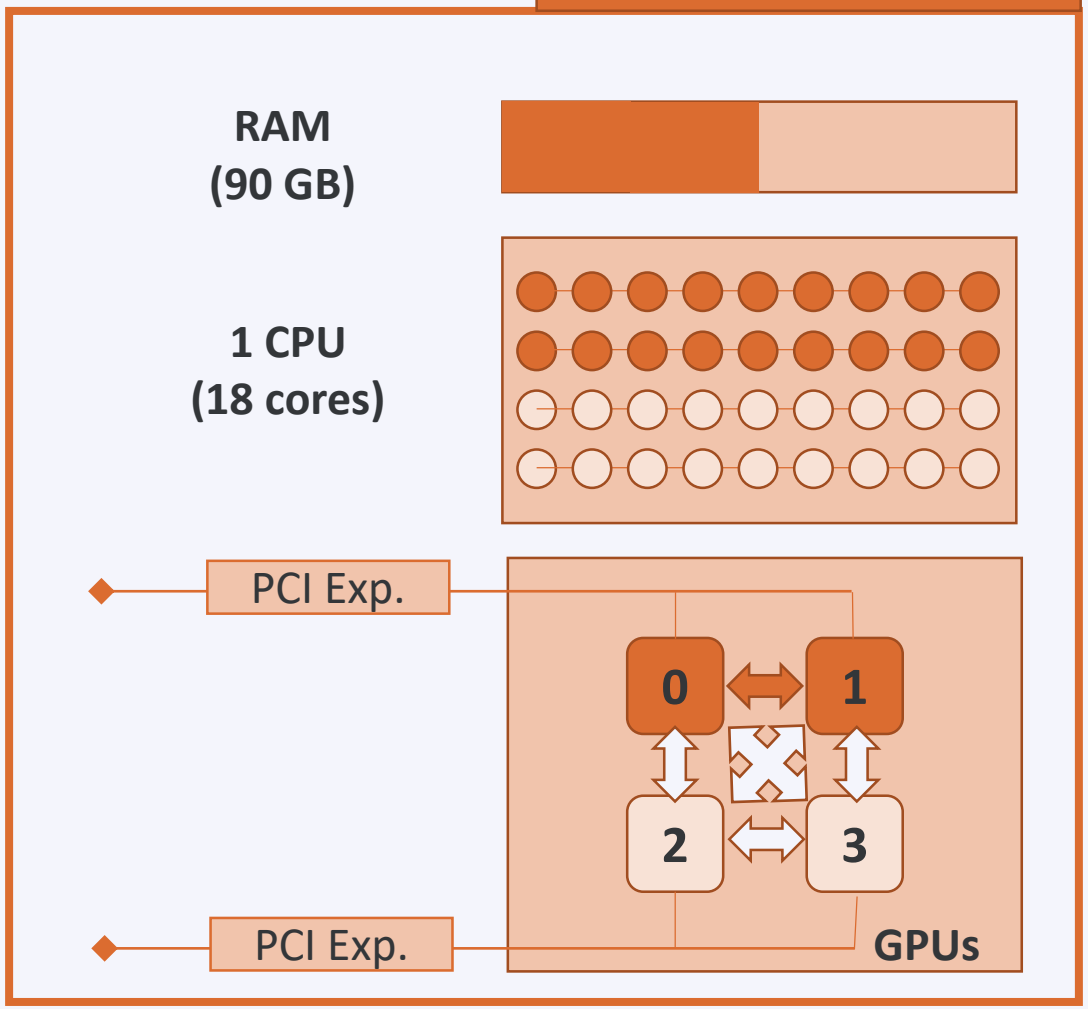


Using P100 GPU(s)

Using 2 GPUs

```
#!/bin/bash -l
#PBS -l walltime=00:30:00
#PBS -l partition=gpu
#PBS -l nodes=1:ppn=18:gpus=2:skylake
#PBS -l pmem=5gb
#PBS -N testjob
#PBS -A default_project
```

You get …

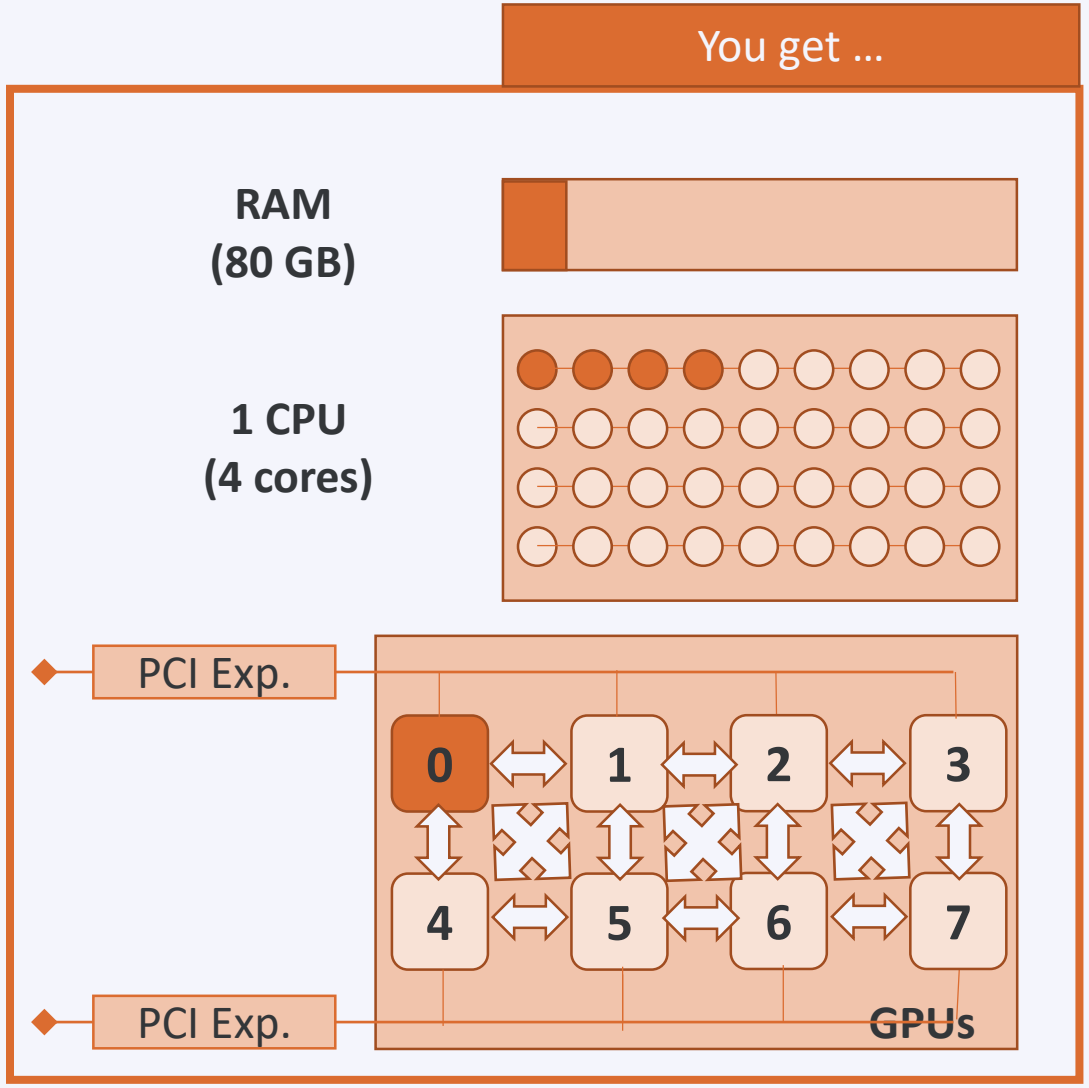


Using V100 GPU(s)

Using 1 GPU

```
#!/bin/bash -l
#PBS -l walltime=00:30:00
#PBS -l partition=gpu
#PBS -l
nodes=1:ppn=4:gpus=1:cascadelake
#PBS -l pmem=20gb
#PBS -N testjob
#PBS -A default_project
```

You get …

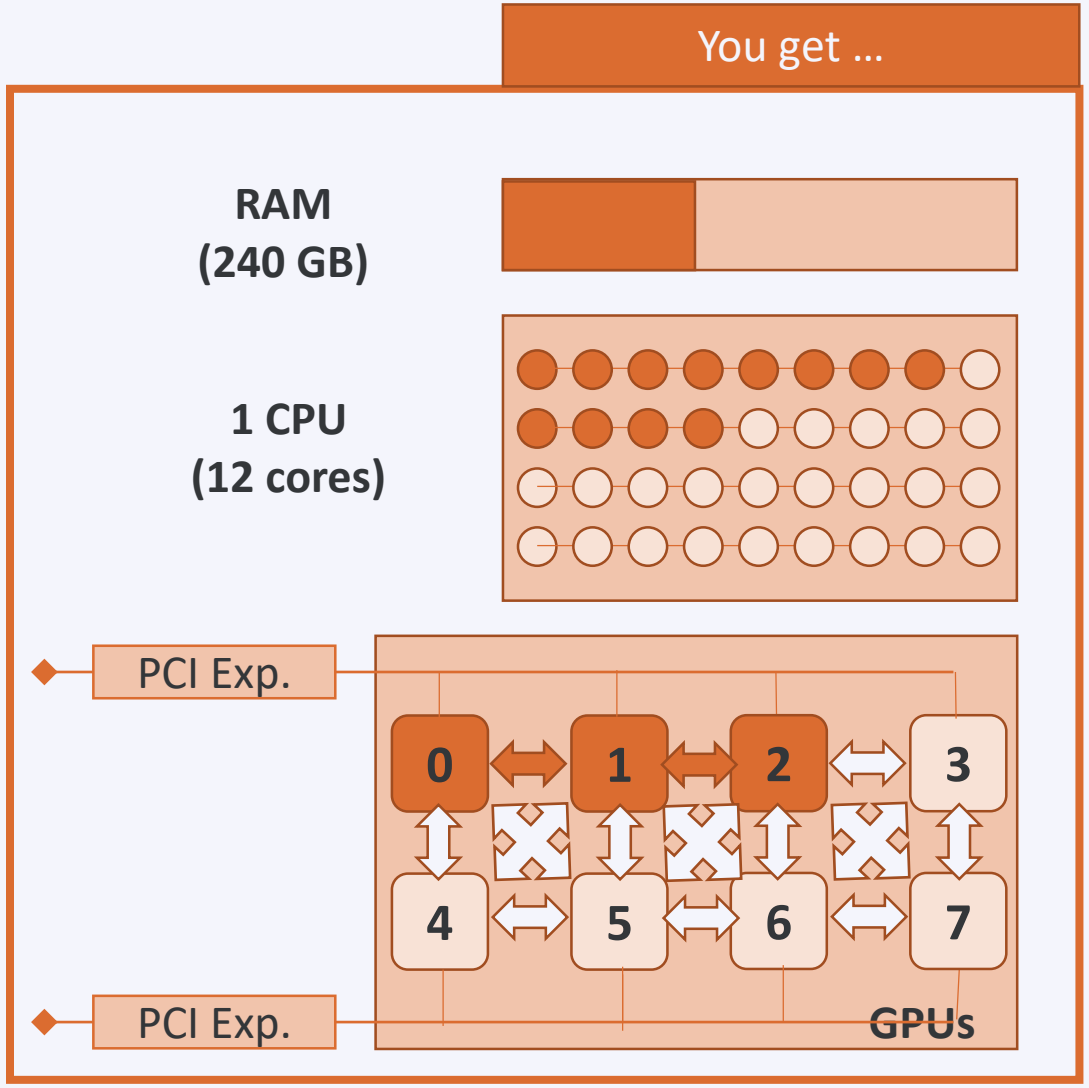


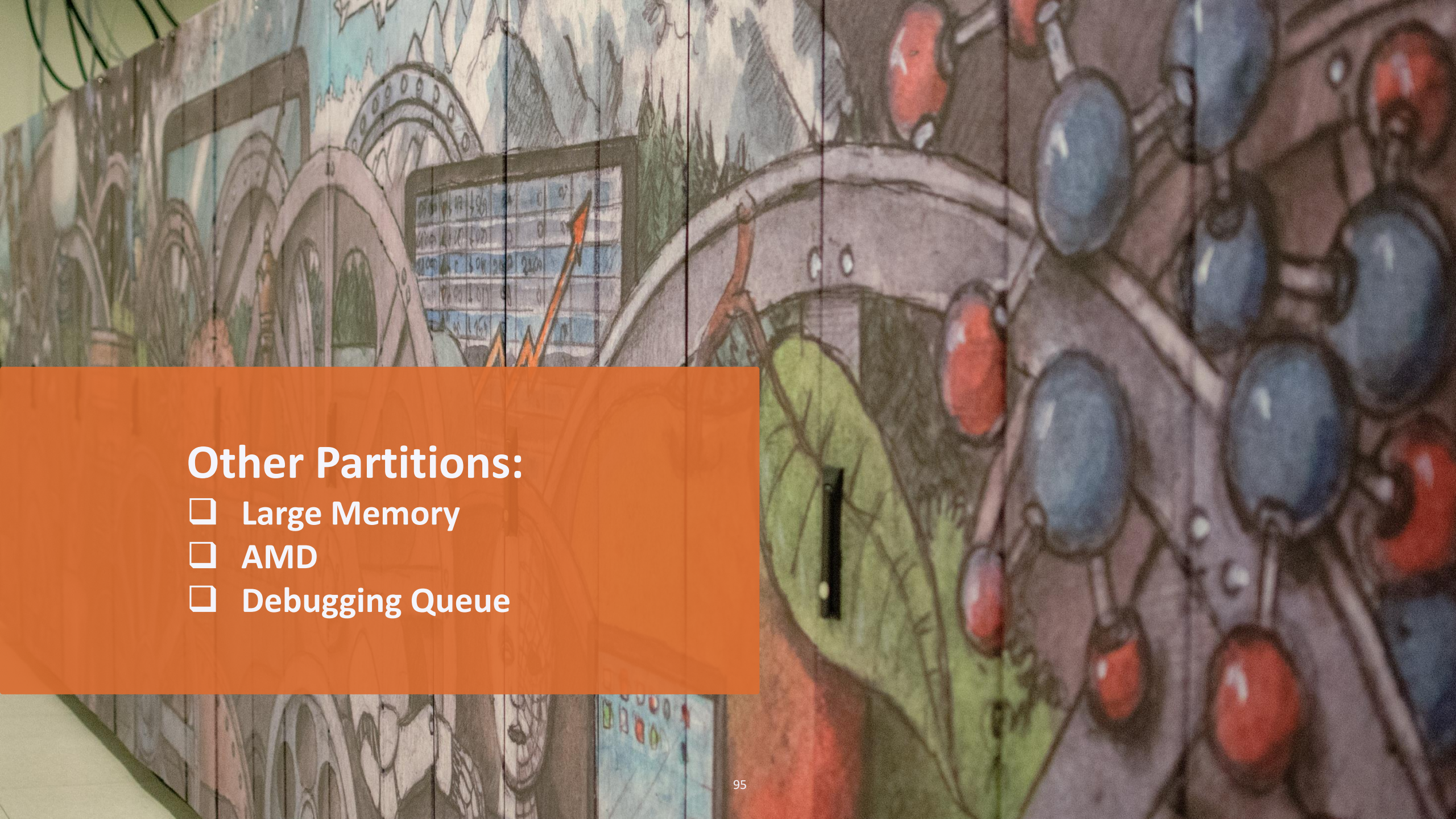
Using V100 GPU(s)

Using 3 GPU

```
#!/bin/bash -l
#PBS -l walltime=00:30:00
#PBS -l partition=gpu
#PBS -l
nodes=1:ppn=12:gpus=3:cascadelake
#PBS -l pmem=20gb
#PBS -N testjob
#PBS -A default_project
```

You get …





Other Partitions:

- ❑ Large Memory
- ❑ AMD
- ❑ Debugging Queue

Requesting Large Memory Nodes

- All Thin Nodes have 192 GB memory each
  You need to leave 8 GB aside for the Operating System, and use the rest
- By default, every core can have
  `pmem = (192GB – 8GB) / (36 cores) ≈ 5GB per-core`
- There are 3 methods you can request larger memory than normal:
  - Use less cores (`ppn`), but larger memory-per-core (`pmem`)
  - Use `bigmem` partition
  - Use `Superdome` machine

First Method

- You sacrifice `ppn`, in favor of `pmem`
- E.g. if you need 20GB memory per core:
  `$> qsub -I -l nodes=1:ppn=8,pmem=20gb …`

- Note: the total memory requested must be less than 192GB – 8GB = 184GB, i.e.
  `ppn × pmem ≤ 184GB`

Requesting Large Memory Nodes

Second Method

- We have 10 Large Memory nodes, with 768 GB memory each
- Specify `-l partition=bigmem` with `qsub` command
- Each core can have
  `pmem=(768 GB - 18 GB) / (36 cores)  ≈ 20 GB per-core`

- E.g.
  ```
  $> qsub -I -l partition=bigmem,nodes=1:ppn=36,pmem=20gb …
  ```

- You can also compromise `ppn` for `pmem`
  E.g.
  ```
  $> qsub -I -l partition=bigmem,nodes=1:ppn=10,pmem=75gb …
  ```

- Make sure `ppn × pmem  ≤ 750GB`

Requesting Large Memory Nodes

Third Method

- Superdome: 8 nodes, 14 cores-per-node, **6 TB** total RAM
- Default `pmem=50gb`
- `numanode` is total CPUs to use (max. 8)
- `lprocs` is total number of cores to use

- E.g.

```
$> module load superdome
$> qsub -l partition=superdome -q qsuperdome \
          -L tasks=1:lprocs=14:place=numanode=1
```

- E.g.

```
$> qsub -l partition=superdome -q qsuperdome \
          -L tasks=1:lprocs=42:place=numanode=3
```

```
$> qsub -l partition=superdome -q qsuperdome \
          -L tasks=1:lprocs=7:place=numanode=1:memory=700gb
```

Request **1/8th** of the machine, with 50 GB per core

Request **3/8th** of the machine, with 50 GB per core

If you want to use > 50GB/core
Eg 7 cores with 100GB/core

(Experimental) AMD Nodes

- 4 nodes
- 2x EPYC 7501 CPUs (2.0 GHz, 32 cores)
- 8 memory channels / CPU
- Max Mem BW 158.95 GiB/s
- RAM: 256 GB
- Remark: `pmem=3800mb`
  If not specified, the job will not start

Job Script

```
#!/bin/bash -l
#PBS -l walltime=01:30:00
#PBS -l partition=amd
#PBS -l nodes=1:ppn=64
#PBS -l pmem=3800mb
#PBS -N testjob
#PBS -A lp_my_project
```

Debugging / Testing Jobs

- To quickly test/debug your (parallel) application
- 2 dedicated nodes on ThinKing and Genius
  One GPU node and one CPU node
- Such jobs do **not** go to the normal queue, so they start faster
- Max. walltime is **30 minutes**
- You must specify Quality of Service (**qos**)

Request Debugging Nodes

```
$> qsub -l nodes=1:ppn=10 -l qos=debugging -l \
      walltime=30:00 -A default_project
```

Managing & Monitoring Jobs

Command	Purpose
`$> qsub ….`	Submit a job (batch/interactive)
`$> qdel <JobID>`	Delete a specific job
`$> checkjob -vvv <JobID>`	Very detailed job info (very useful to diagnose issues)
`$> qstat -n`	Status of all recent jobs
`$> qstat -Q -f`	Info about available queues
`$> showstart <JobID>`	Give a *rough* estimate of start time
`$> showq` `$> showq -p gpu`	Show minimal info about a queue or partition (`-p`)

`$> pbstop`
`$> clusterview`
`$> queueview`
	Overview of the cluster
	Overview of the nodes, cores and GPUs
	Overview of the job queues

`$> mam-balance`	Overview of different credit projects that you can use (`qsub -A <Project>`)
`$> mam-list-allocations`	Detailed overview of your credit projects

Jupyter Notebook on NX

- Install Miniconda in your $VSC_DATA

```
$> wget https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh
$> bash Miniconda3-latest-Linux-x86_64.sh -p $VSC_DATA/miniconda3
```

- Create a conda env including Jupyter

```
$> conda create -n JupNtbk jupyter scikit-learn
```

- Start an interactive GPU job (`qsub -I …`)
- Activate this env

```
$> conda activate JupNtbk
```

- In another terminal, launch the notebook to get URL

```
$> jupyter notebook --ip=`hostname -i` --port=${USER:3} --no-browser
--allow-root
```

- Copy-paste the URL into NX Firefox; done

Singularity Containers

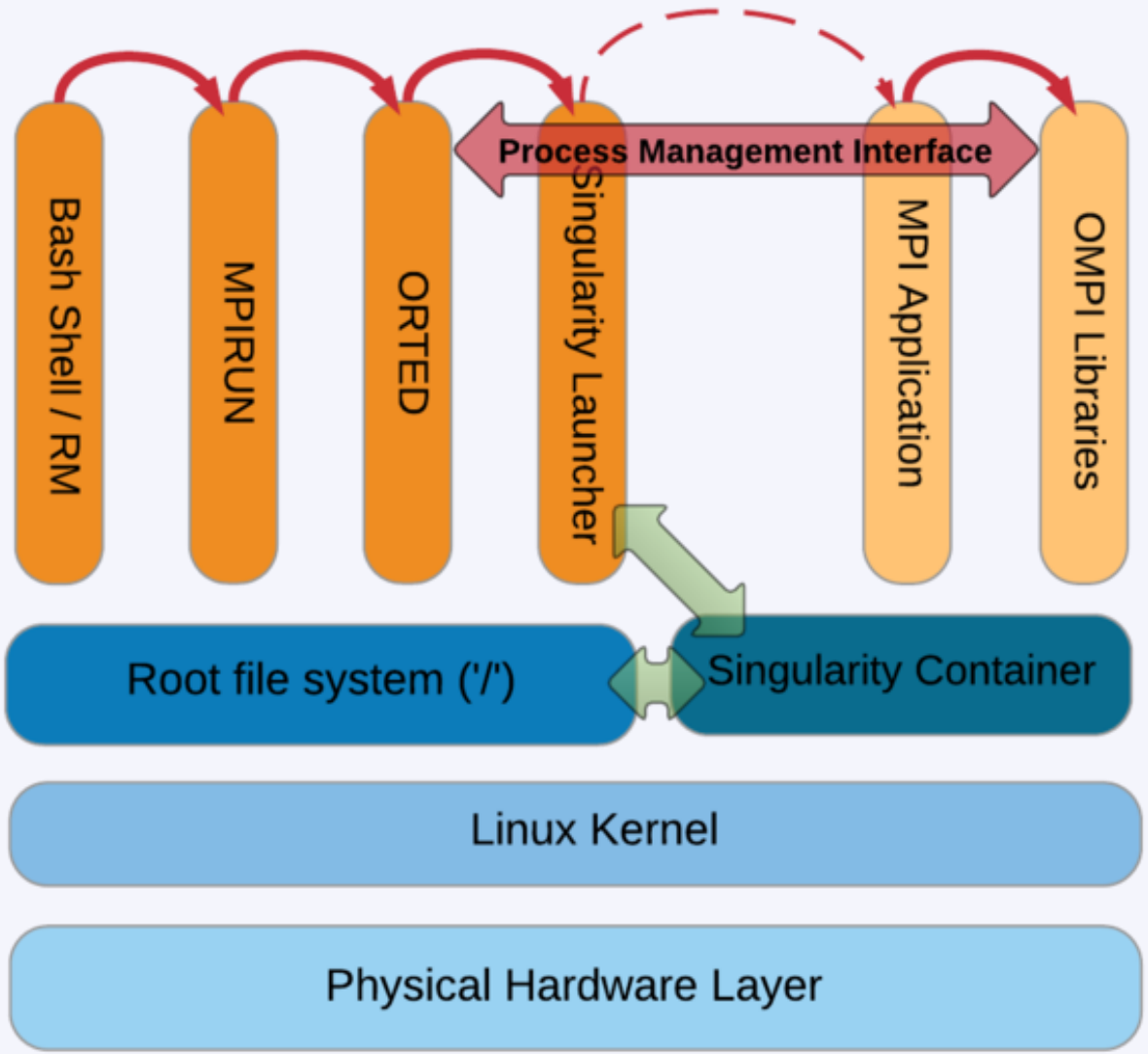
- What?
  + Self-contained OS & software & data
- Why?
  + fully resolved dependency chains
  + portable workflow
- How?
  + You create the image
  + Run it on Genius
  + MPI/OpenMP is supported

PBS Script

```
#!/bin/bash -l
#PBS -l walltime=00:30:00
#PBS -l nodes=4:ppn=36:pmem=5gb
#PBS -A lp_hpcinfo_training

cd $PBS_O_WORKDIR

singularity run Project.simg ./model.exe
```



Where to run your jobs

Which system should I use?

Knowing yourself is
the beginning
of all wisdom

What kind of work I do?

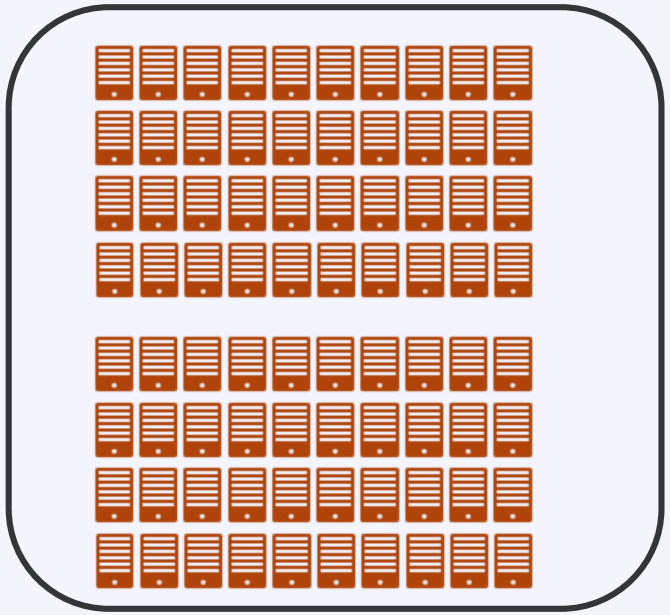
- CPU usage
  - Do you run single core jobs or parallel?
    - Single core
      - Is your code optimized to AVX ?
    - Parallel:
      - single node or multinode?
        - Single node: does my application scale? Up to how much cores ?
      - Multinode:
        - Does my application scale?
- Memory usage
  - How much memory (per core) my jobs use?
  - Is your processing memory bound?

What kind of work I do?

- How to find out
  - Know your algorithm
  - Benchmark your typical workload on the different architectures
  - Use tools to track behavior of your code
    - Use Monitor tool
    - Use profiling tools eg. ARM MAP or Scalasca
  - Consult training agenda
    - Upcoming Allinea ARM Forge training

Which system for what?

Compute nodes (thin nodes)



Thinking:

- Haswell: 24 cores
  64/128 GB

Single core jobs
Worker jobs
Single node parallel jobs

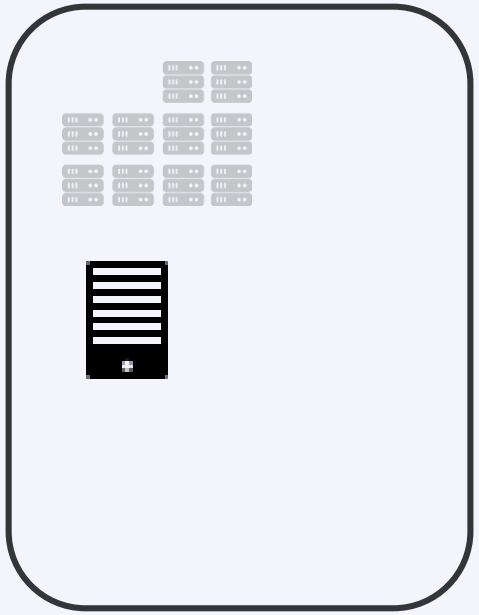
Genius:

- Skylake: 36 cores
- Cascade Lake: 36 cores
- 192 GB
- IB EDR

Multinode parallel jobs
AVX 512
Higher core/memory ratio
Higher memory bandwidth

Which system for what?

Compute nodes (Large memory)



Genius:

- Big memory 36 cores
- 768 GB

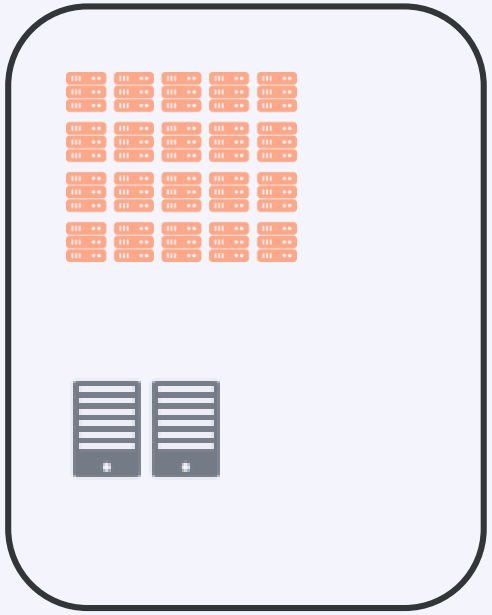
5GB<mem/core<20GB

- Superdome 112 cores
- 6TB

> 20GB mem/core
> 700 GB of total RAM
Single node scaling > 36 cores

Which system for what?

Compute nodes (GPU nodes)



Genius Skylake GPU: P100

Single GPU workload
Multi GPU workload with limited scaling

Genius Cascadelake GPU: V100

> 4 GPU workload
Specific ML workloads (convolutional networks)

Credits

Credits card concept:

- Pre-authorization: holding the balance as unavailable until the merchant clears the transaction
- Balance to be held as unavailable: based on requested resourced (walltime, nodes)
- Actual charge based on what was really used: used walltime (you pay only what you use, e.g. when job crashes)
- See output file

How to check available credits?

```
$ mam-balance
```

How to check the rates?
Check service catalogue

```
$ mam-list-chargerates
```

Credit Pricing

- You pay as you go!
- For academic projects:
  1000 credits = 3.5 EUR
- Credits needed for a job:

  #credits = Walltime (hr) × #nodes × Factor

- For shared nodes, you pay a fraction of the costs, based on the ppn specified

Cluster / Partition	Credits/h
Genius Cascadelake	11.3
Genius Cascadelake 8 GPUs	40
Genius Skylake 4 GPUs	20
Genius Skylake BigMem	12
Genius Skylake	10
Genius / Superdome	10
Genius / AMD nodes	10
ThinKing / Haswell	6.68
ThinKing / IvyBridge	4.76

How to Manage Credits?

Command	Purpose
`mam-balance`	List active projects and available credits
`mam-list-allocations`	List the validity dates of different projects/allocations
`mam-list-chargerates`	List current charge rates for different nodes
`mam-list-accounts -a <account-name>`	List the members in an account

demo/test yourself

- Request membership to lp_hpcintro_training group (account.vscentrum.be)

- Login with putty
- Filetransfer with Filezilla
- Login with NX
- Check disk quota
- Check the credits
- Check/load/list/unload/purge module

demo/test yourself

- Copy intro training files (`/apps/leuven/training/HPC_intro/`) to your `$VSC_HOME`
- Submit `cpujob` to the cluster
- List all your jobs (`qstat`)
- Check the information about the cpujob (`checkjob`)
- Modify the mat.pbs script to request 1 node, 36 cores for 30 minutes and get the notification about job start/end by e-mail
- Check the status of all the jobs

demo - monitoring

- Submit an interactive job
  Run your program on a compute node
  Open a new terminal and ssh to a compute node
  Check the resources usage (`top`)
- Submit an interactive job to GPU node
  Run your program on a compute node
  Open a new terminal and ssh to a compute node
  Check the resources and GPU usage (`top`, `nvidia-smi` or `watch "nvidia-smi"`)
  Submit a batch GPU job
- While the job is running get the information about the node (`checkjob`) and check usage of resources on the node (`ssh`, `top`, `nvidia-smi`)

demo – conda installation

- Install miniconda in your `$VSC_DATA` directory
  - `$ wget https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh`
  - `$ bash Miniconda3-latest-Linux-x86_64.sh -b -p $VSC_DATA/miniconda3`
- Add a PATH to conda:
  - `$ export PATH="${VSC_DATA}/miniconda3/bin:${PATH}"`
- Check if conda in added to your `$PATH ($ which conda)`
- Add it to `$PATH` in your `.bashrc`
  - `$ echo 'export PATH="${VSC_DATA}/miniconda3/bin:${PATH}" ' >> .bashrc`

demo – conda usage

- Create a conda environment including Jupyter

```
$ conda create -n science jupyter numpy scipy
```

- Activate this environment

```
$ source activate science
```

- Add matplotlib package to this environment

```
$ conda install matplotlib
```

- Return to original environment

```
$ conda deactivate
```

demo – notebooks

- Start an interactive GPU job

```
$ qsub -I -l walltime=30:00 -l nodes=1:ppn=9:gpus=1 -l partition=gpu -A default_project
```

- Activate conda environment

```
$ source activate science
```

- Go to your working directory (you can use `$PBS_O_WORKDIR` if you qsub from there)
- Start notebook

```
$ jupyter notebook --port ${USER:3} --ip $(hostname)
$ jupyter notebook --port 30468 --ip $(hostname)
```

- Open the link in the browser in NX and test your notebook

demo – worker

- Copy intro training files (`/apps/leuven/training/worker/`) to your `$VSC_HOME`
- Go to `exercise1` directory
- Submit worker job
- Check the output file

Questions

Helpdesk:
hpcinfo@kuleuven.be or https://admin.kuleuven.be/icts/HPCinfo_form/HPC-info-formulier

VSC web site:
http://www.vscentrum.be/

VSC documentation: https://vlaams-supercomputing-centrum-vscdocumentation.readthedocs-hosted.com/en/latest/

VSC agenda: training sessions, events

Systems status page:
http://status.kuleuven.be/hpc