VSC HPC Introduction

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

Vlaanderen
is supercomputing

VLAAMS
SUPERCOMPUTER
CENTRUM

Innovative Computing for A Smarter Flanders

vscentrum.be

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 & MFA
- 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

Tier-1

Vlaanderen is supercomputing

Hortense

Tier-2

GHENT UNIVERSITY — **STEVIN**

VUB VRIJE UNIVERSITEIT BRUSSEL — **HYDRA**

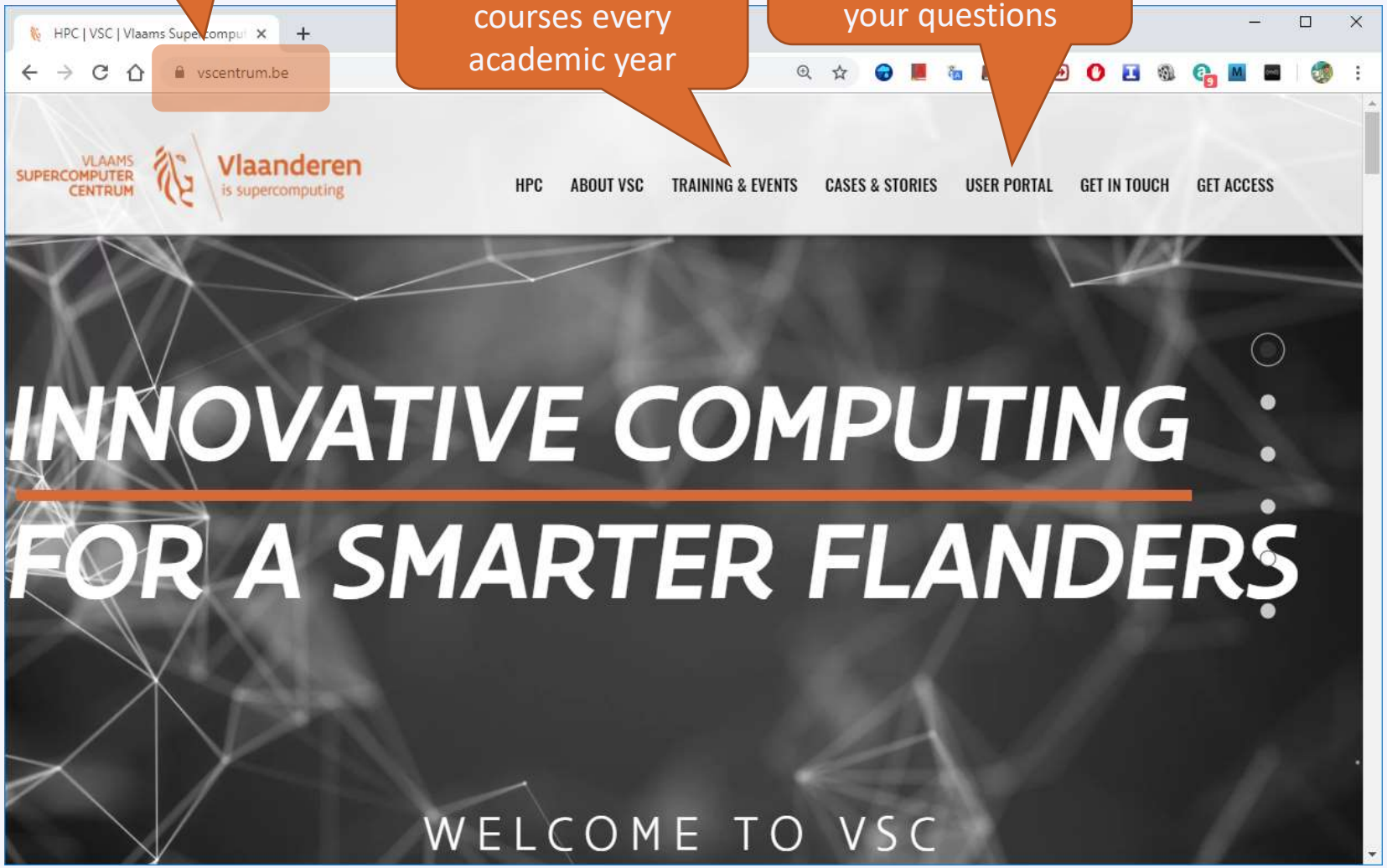
Universiteit Antwerpen — **HOPPER/LEIBNIZ**

KU LEUVEN UHASSELT — **GENIUS/WICE**

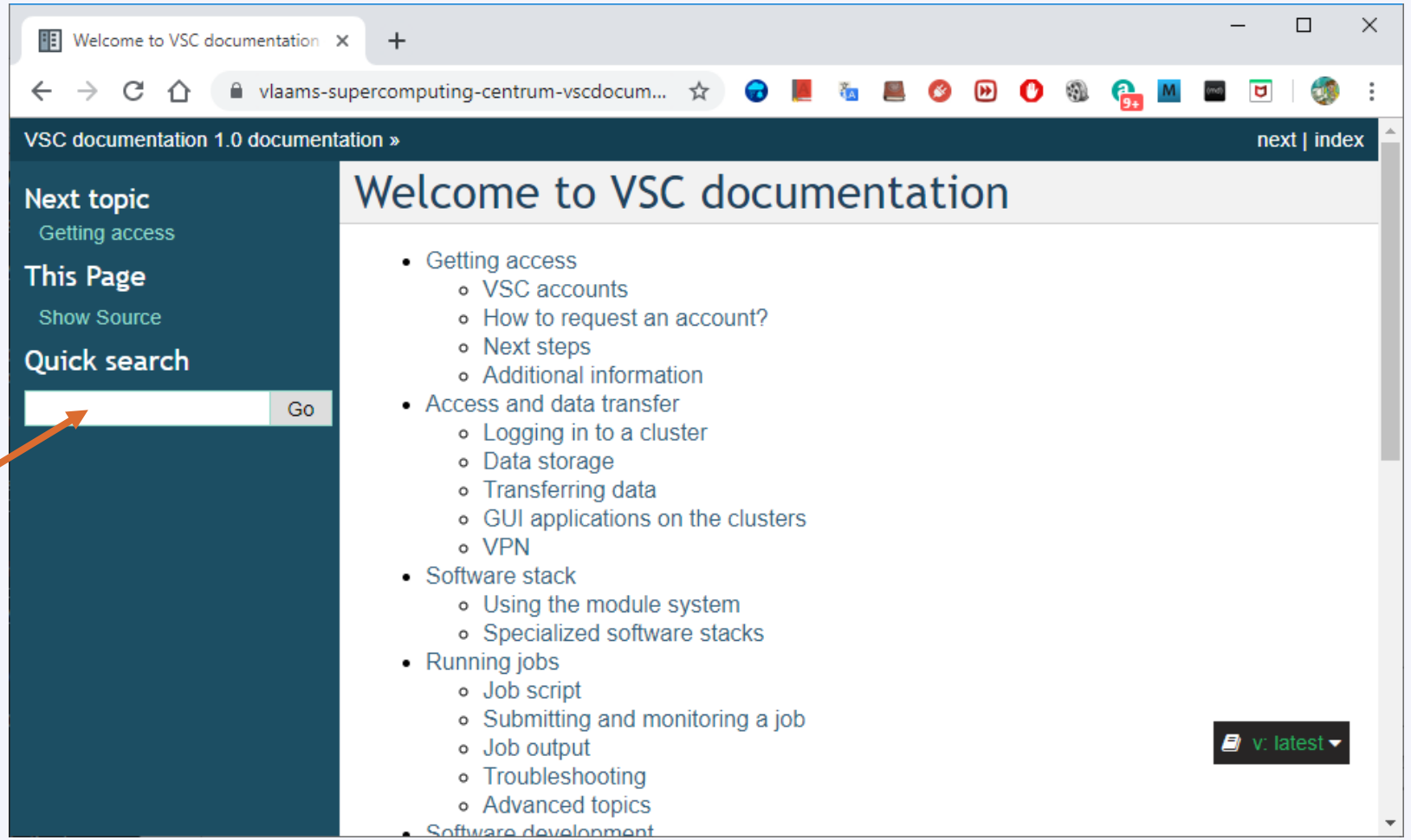
www.vscentrum.be

Very rich suite of courses every academic year

Documentation! Answers >80% of your questions



VLAAMS SUPERCOMPUTER CENTRUM

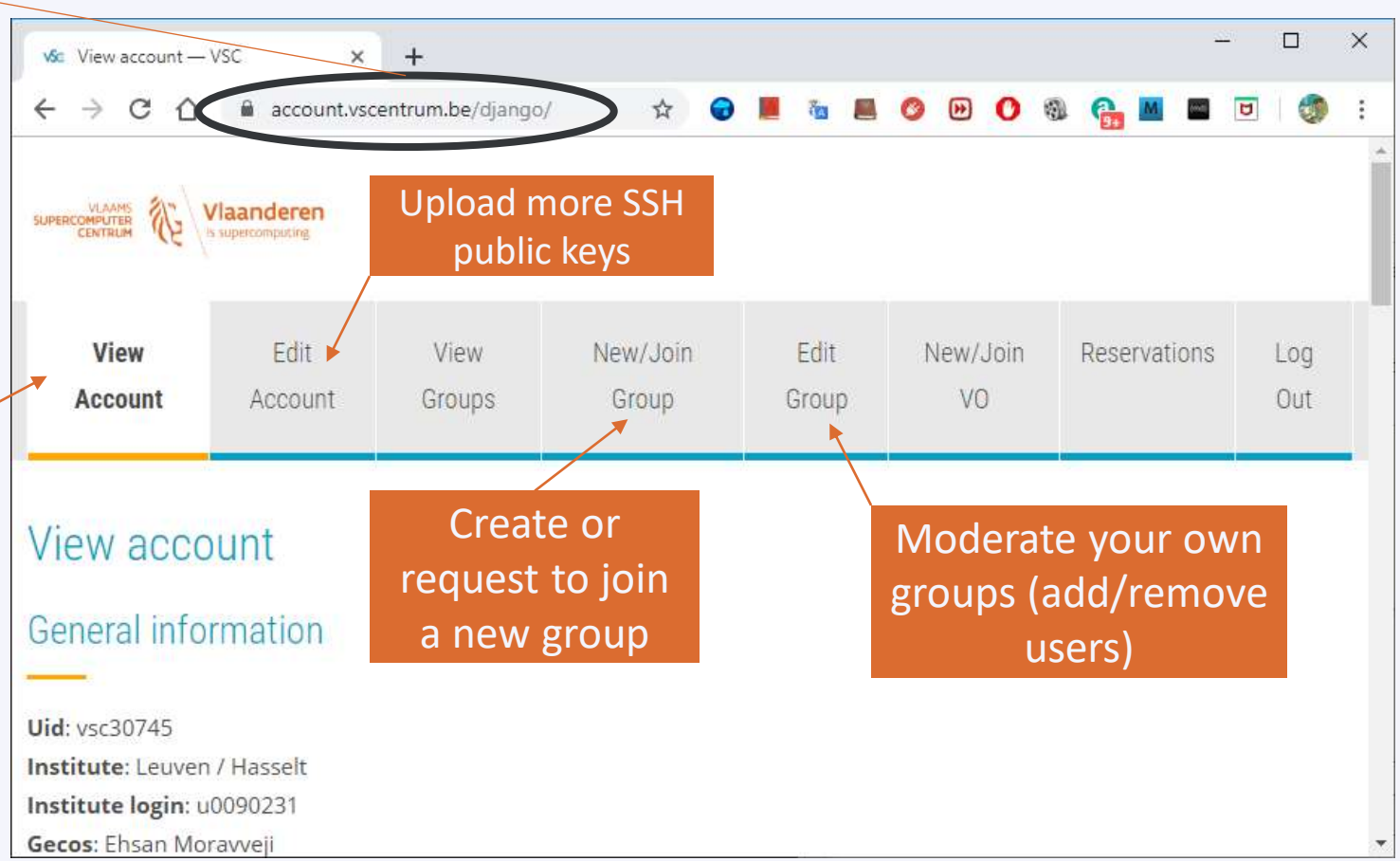


Search you keywords here; e.g. qsub

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 more SSH public keys

Upload SSH Key Request account

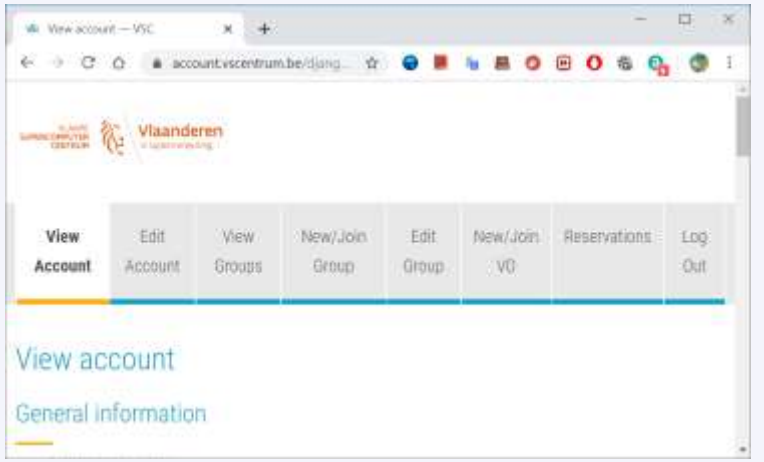
Create or request to join a new group

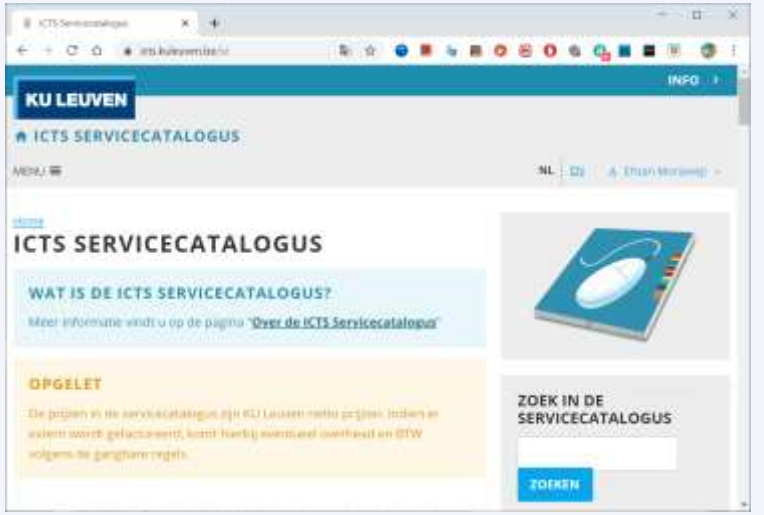
Moderate your own groups (add/remove users)

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 (2M 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
  Staging fileset: 20 € 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)

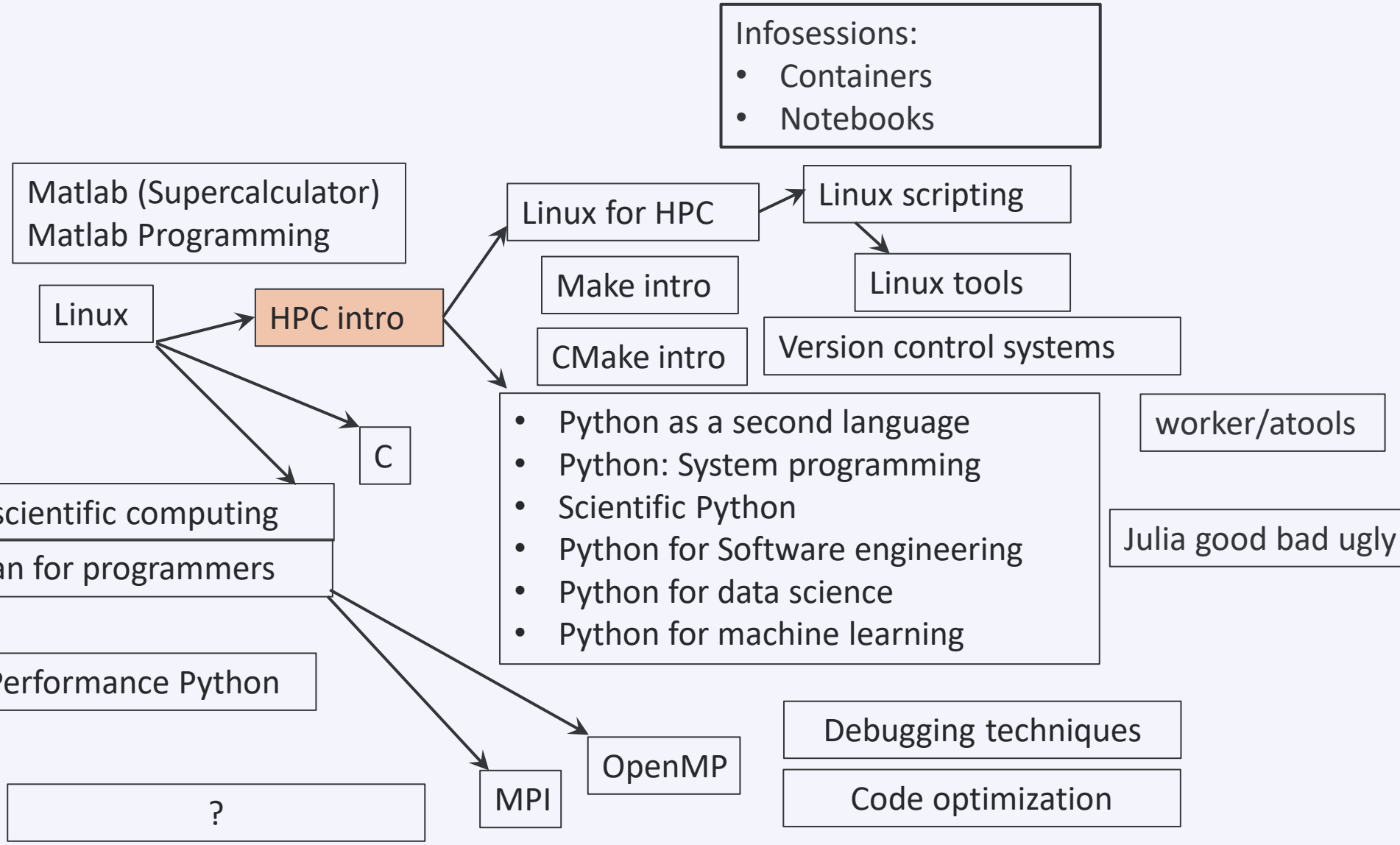




VLAAMS SUPERCOMPUTER CENTRUM

VSC training

- Introductory
- Intermediate
- Advanced
- Specialized track



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.

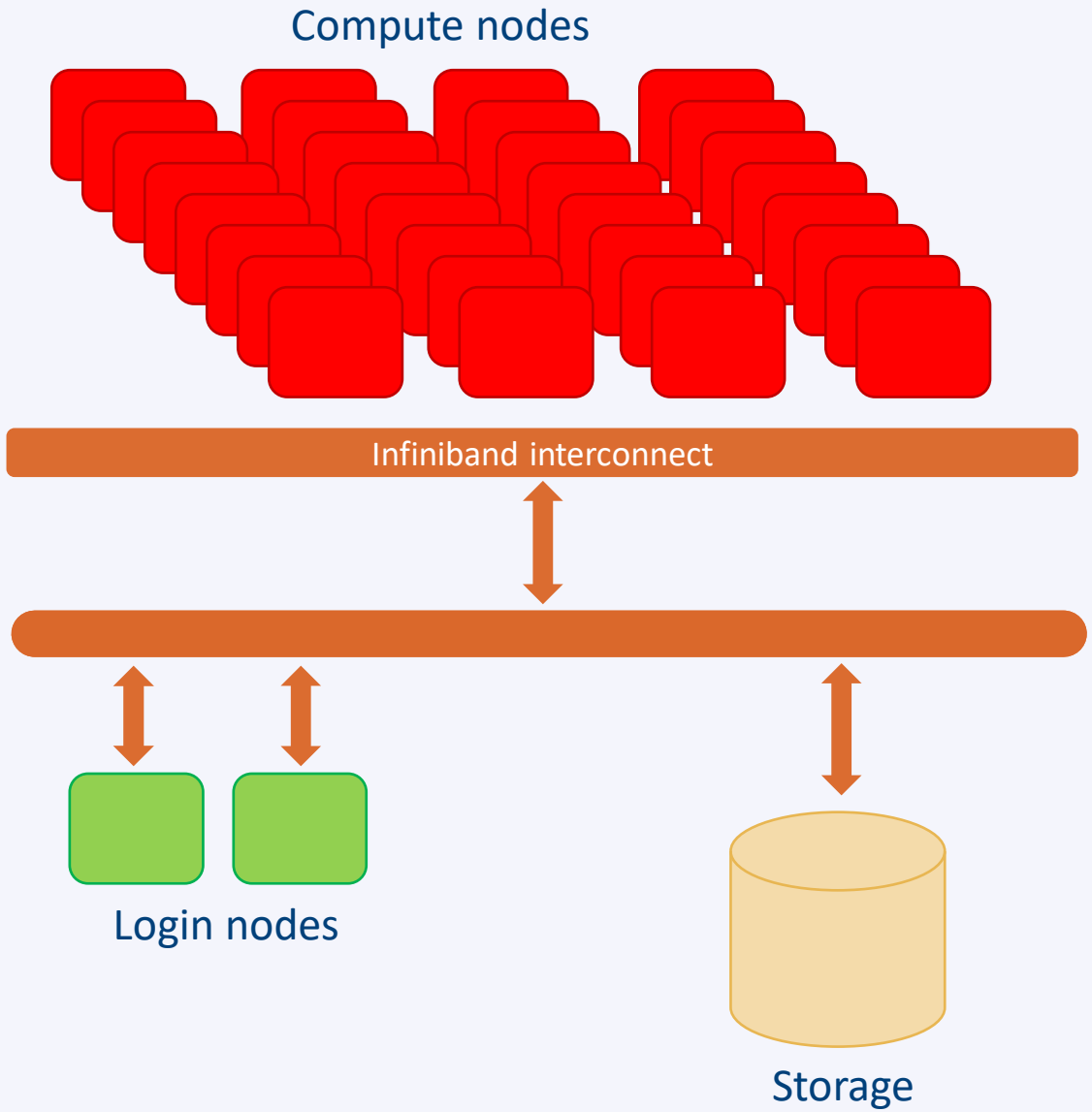
Tier-2 Clusters

What is a Cluster?

A cluster is comprised of **many powerful interconnected compute nodes**

Some Terminology:

	What	Components
Cluster	The primary part of a cluster are its many powerful interconnected compute nodes.	Compute nodes Login nodes Storage space Interconnect Network …
Node	A **single computer** connected with other nodes via an interconnect network	CPU, Storage, Memory, …
CPU	Processing Component of a computer, also called the **processor**.	Controllers, Cache Memory, Processing Cores
Core	Processing **element of the CPU**.	Control Unit, Arithimic-Logic Unit, Memory.



Tier-2 Clusters @ KU Leuven

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

wICE (since 9/2022)
186 nodes; 13,392 cores

wICE (since 9/2022)
186 nodes; 13,392 cores

IB EDR

Belnet

KU Leuven

Superdome

AMD

SkyLake

86+10 nodes
3456 cores
2x Intel Skylake
18 cores
192 GB RAM
768 GB RAM

20 nodes
720 cores
2x Intel Skylake
18 cores
4 x NVIDIA P100

Cascade Lake

120 nodes
4320 cores
2x Intel Cascadelake
18 cores
192 GB RAM

2 nodes
72 cores
2x Intel Cascadelake
18 cores
8 x NVIDIA V100

IB EDR

Ethernet

NAS
70 TB
HOME
DATA

Ethernet

NFS
Archive
1 PB

IB

GPFS
Scratch
DDN
1,2 PB

IB EDR

2 nodes
72 cores
2x Intel Skylake
18 cores
384 GB RAM

IB EDR

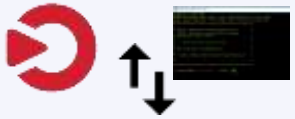
2 nodes
72 cores
2x Intel Skylake
18 cores
384 GB RAM
1 x NVIDIA P600

login-genius.hpc.kuleuven.be

Watch out with
login scripts !

Genius

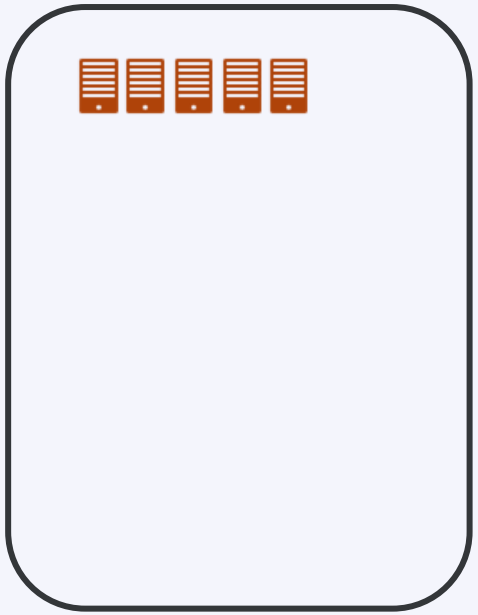
wICE Overview



Compute nodes

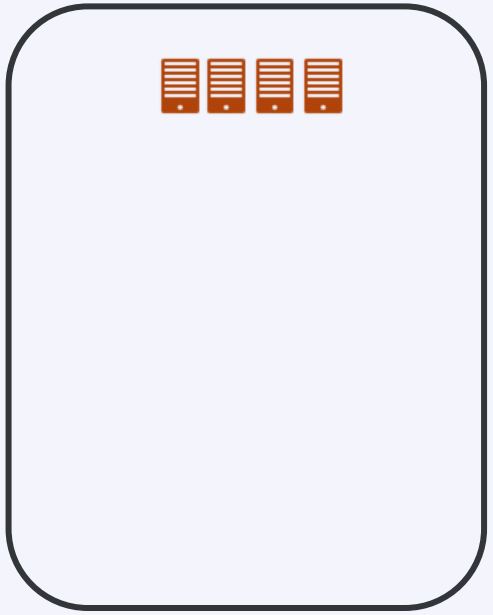
Thin nodes
+ 172x Icelake 72c 256 GB

Large memory nodes



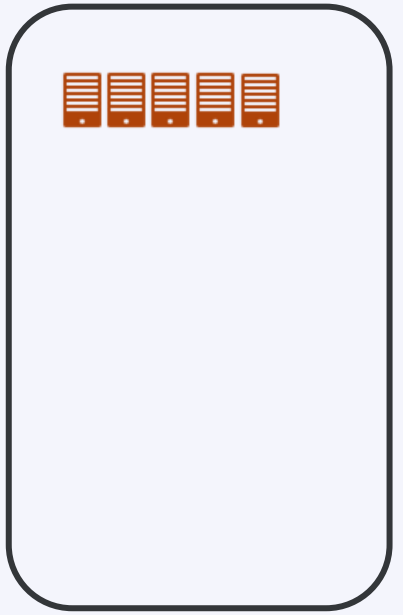
BigMem
+ 5x Icelake 72c 2 TB

GPU nodes



GPUs
+ 4x Icelake 72c 512 GB
  4x A100 80 GB

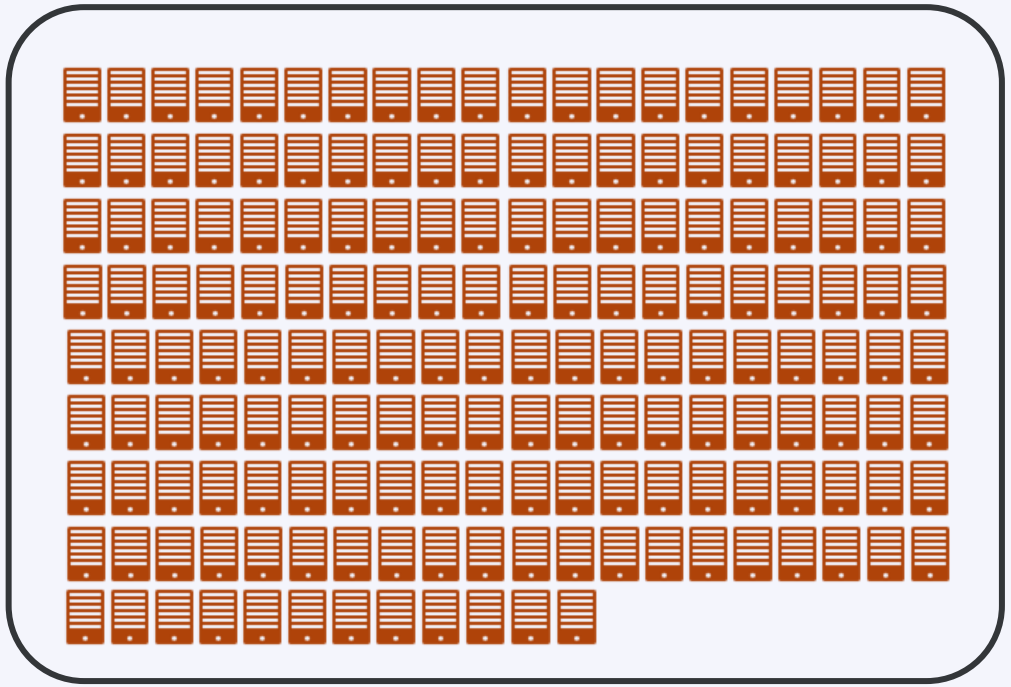
Interactive nodes



+ 5x Icelake 64c 512 GB
  1x A100 80 GB

Tier-2 Overview wICE

Compute nodes



172x IceLake 72c 256 GB

Large memory nodes

5x IceLake 72c 2048 GB

GPU nodes

4x IceLake 72c 512 GB
4 A100 SXM4 80GB

Interactive nodes

5x IceLake 64c 512 GB
1 A100 80GB

Tier-2 Cluster: wICE

Type of node	CPU type	Inter-connect	# cores	installed mem	local discs	# nodes
Icelake	Xeon 8358	IB HDR-100	72	**256 GB**	960 GB	**172**
Icelake **large mem**	Xeon 8358	IB HDR-100	72	**2048 GB**	960 GB	**5**
Icelake **GPU**	Xeon 8358 **4xA100** SXM2 80GB	IB HDR-100	72	**512 GB**	960 GB	**4**
Icelake **Interactive**	Xeon 8358 **1xA100** SXM2 80GB	IB HDR-100	64	**512 GB**	960 GB	**5**

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` storage
- Different storage volumes have different:
  - mount point
  - size and performance
  - use case
  - backup and maintenance policy
- More info on ICTS Service Catalog (EN/NL)

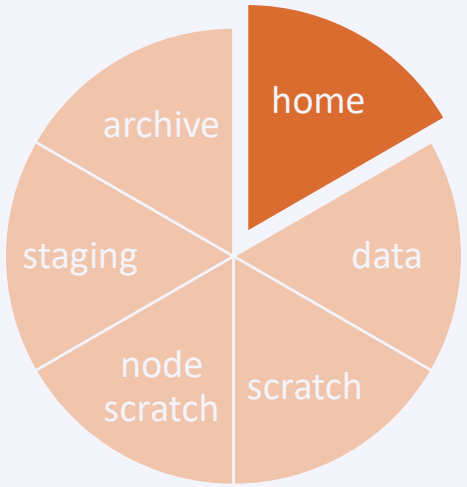
Storage

- 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

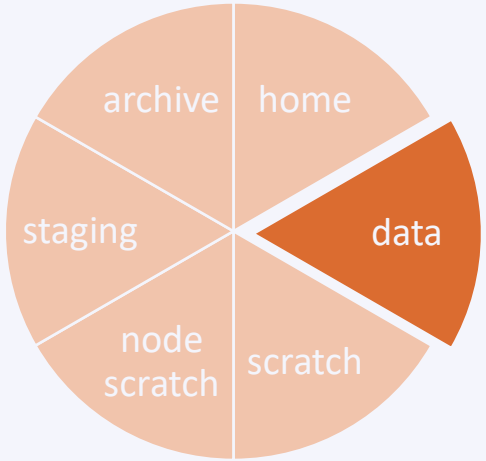
```
$ myquota
file system $VSC_HOME
    Blocks: 1479M of 3072M
    Files: 12934 of 100k
file system $VSC_DATA
    Blocks: 12G of 75G
    Files: 1043k of 10000k
file system $VSC_SCRATCH
    Blocks: 15M of 1.5T
```

Storage



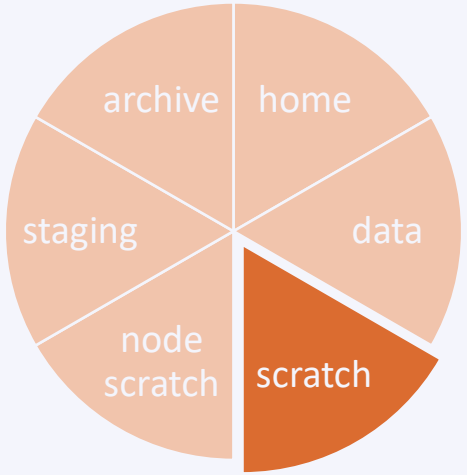
Storage	home folder
Env. Variable	`$VSC_HOME`
Filesystem Type	NFS
Access	Global
Backup	Hourly, daily, weekly (until last month) inside the `.snapshot` folder.
Default Quota	3 GB
Extension	Not possible
Usage	Only storing SSH keys, config files
Remarks	- Stay away from using it - Can easily overflow: +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

Storage



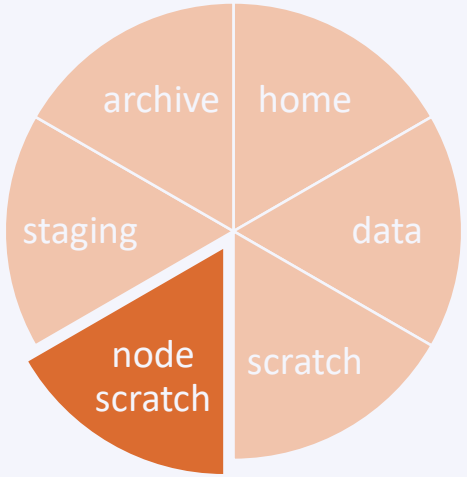
Storage	data folder
Env. Variable	`$VSC_DATA`
Filesystem Type	NFS
Access	Global
Backup	Hourly, daily, weekly (until last month) inside the `.snapshot` folder.
Default Quota	75 GB
Extension	On purchase
Usage	Your data, code, software, results
Remarks	- Permanent storage for initial/final results - Not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

Storage



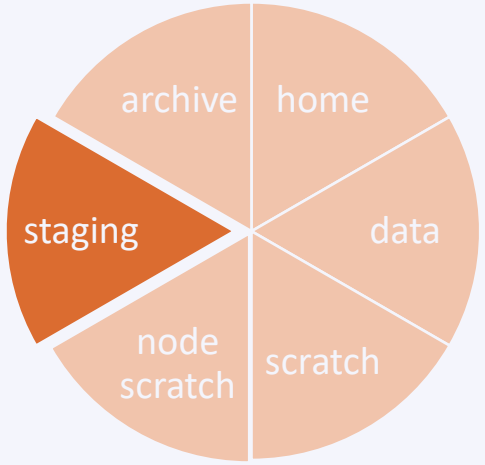
Storage	scratch folder
Env. Variable	`$VSC_SCRATCH`
Filesystem Type	Lustre
Access	Global
Backup	delete after 30 days from last access
Default Quota	500 GB
Extension	Free
Usage	Intensive, parallel I/O, temporary files
Remarks	- Recommended storage for all jobs - Copy scratch files to VSC_DATA or local storage after jobs are done - Deleted files cannot be recovered

Storage



Storage	Node scratch folder
Env. Variable	`$VSC_SCRATCH_NODE`
Filesystem Type	Lustre
Access	On compute node, only at runtime
Backup	None
Default Quota	591 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

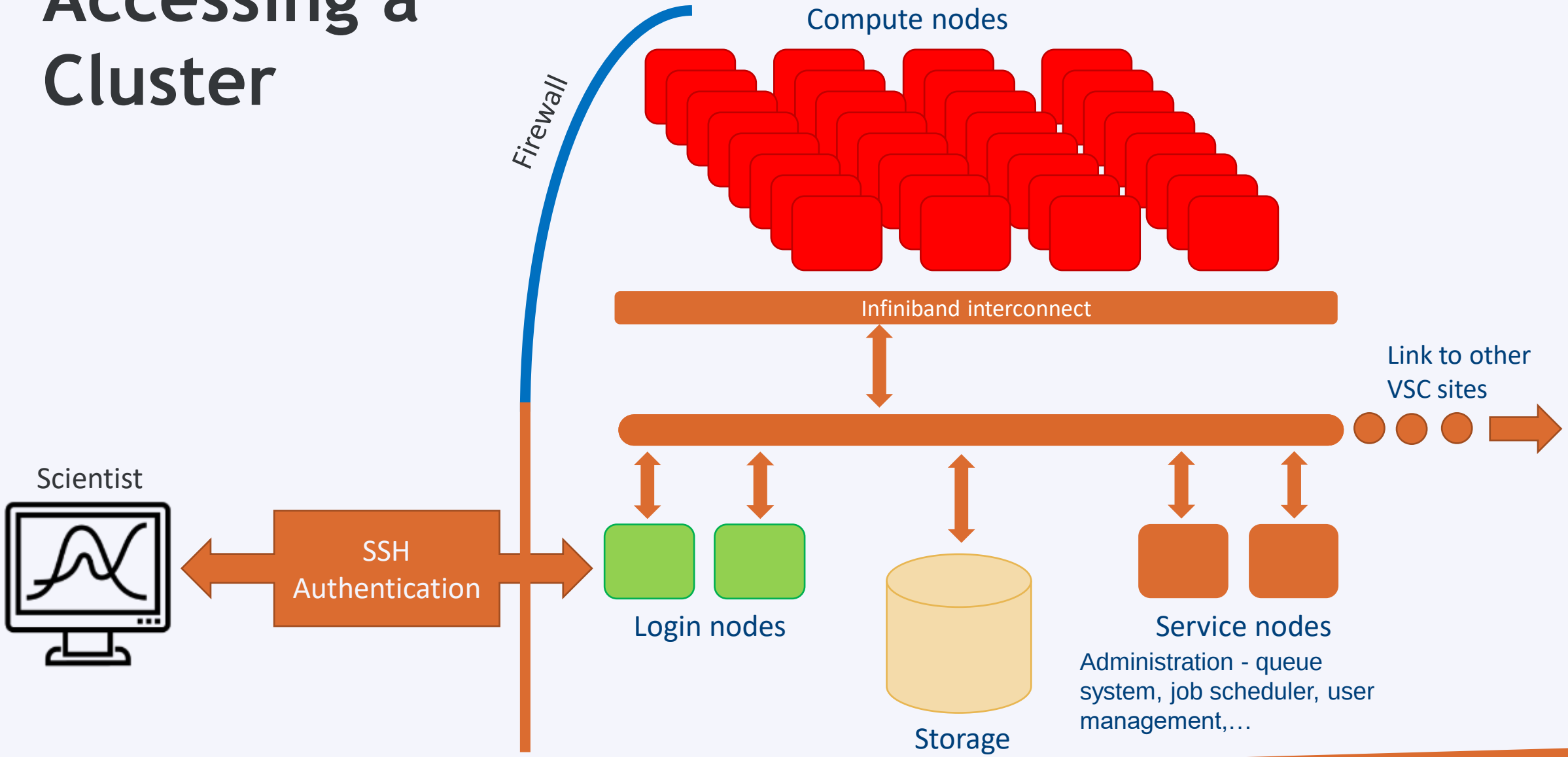
Storage



Storage	Staging folder
Path	`/staging/leuven/stg_000XX`
Filesystem Type	Lustre
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

Login Nodes & MFA

Accessing a Cluster



VLAAMS SUPERCOMPUTER CENTRUM

Using Login Nodes

- ❑ To develop code
- ❑ 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

Tips

Command
`$ top`
Show use of the system

- ❑ Compile your software on compute nodes (via e.g. interactive job)
- ❑ Login nodes are shared resources
- ❑ **Do not** execute heavy-lifting tasks (core, memory)
- ❑ Instead, submit jobs
- ❑ Check `$ slurmtop` to see how busy the cluster is

Warning

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)
Genius	**login.hpc.kuleuven.be**	Recommended
	login{1,2}-tier2.hpc.kuleuven.be	No GPU
	login{3,4}-tier2.hpc.kuleuven.be	Nvidia Quadro P6000
wICE		Accessible only via Genius

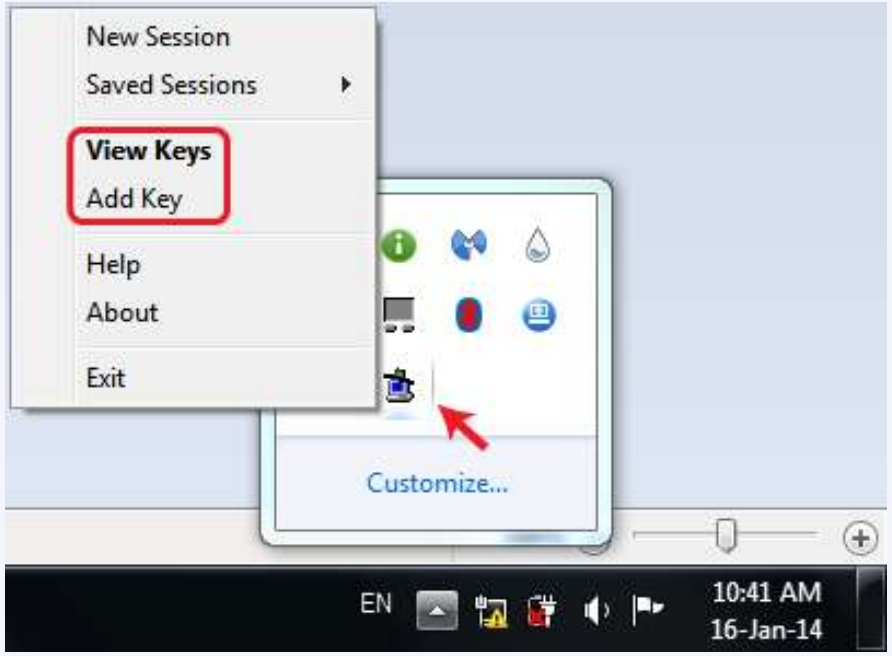
Login Setup

- ❑ PuTTY
- ❑ Terminal

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 SSH keys and Certificates (Multi-Factor Authentication)

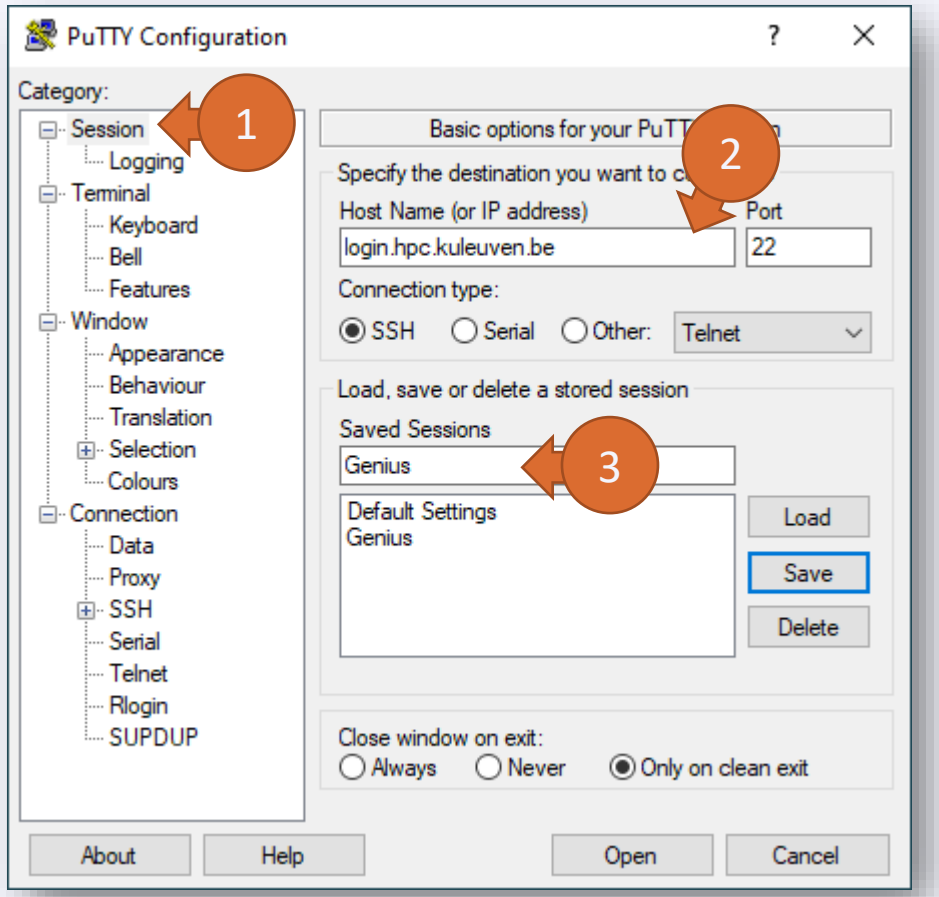


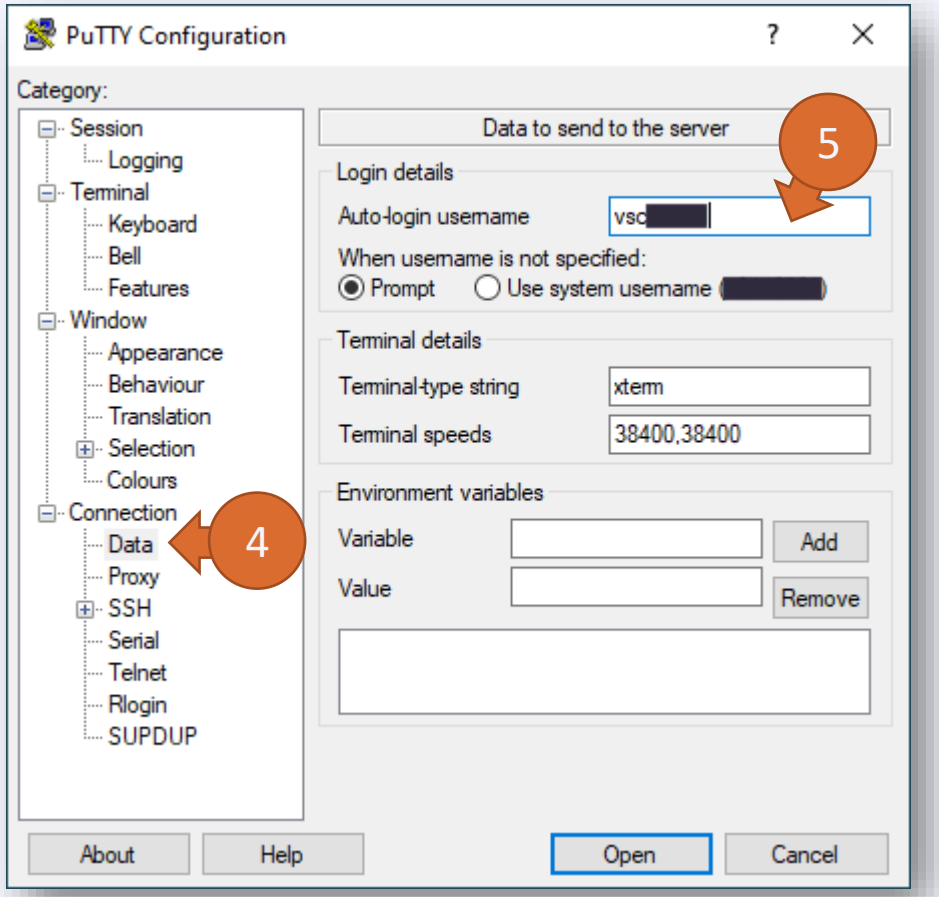
Setup PuTTY in 12 Clicks

(Windows only)

Host Name:

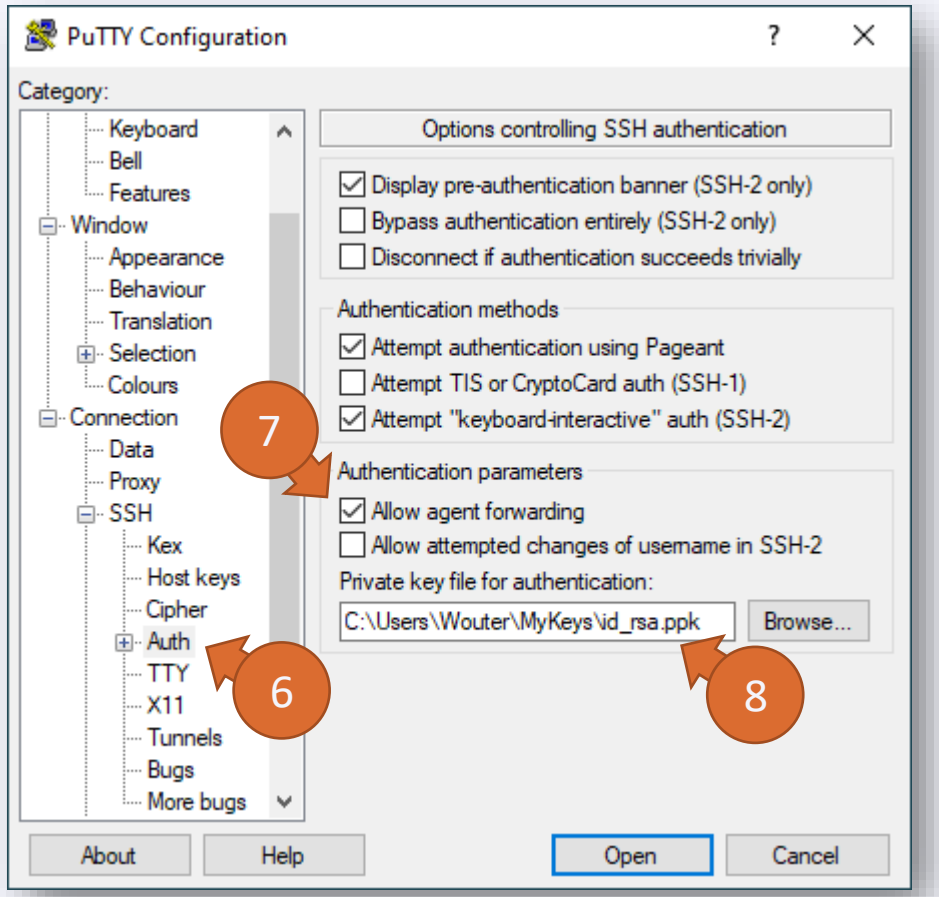
`login.hpc.kuleuven.be`

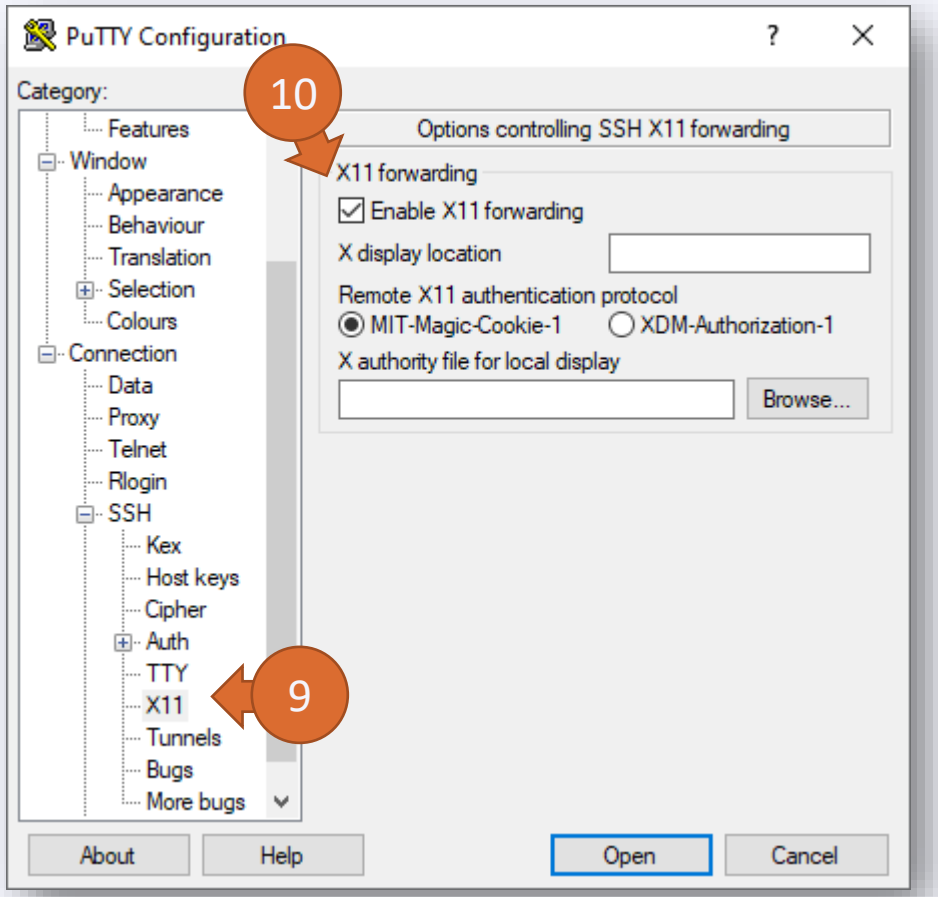




Setup PuTTY in 12 Clicks

(Windows only)

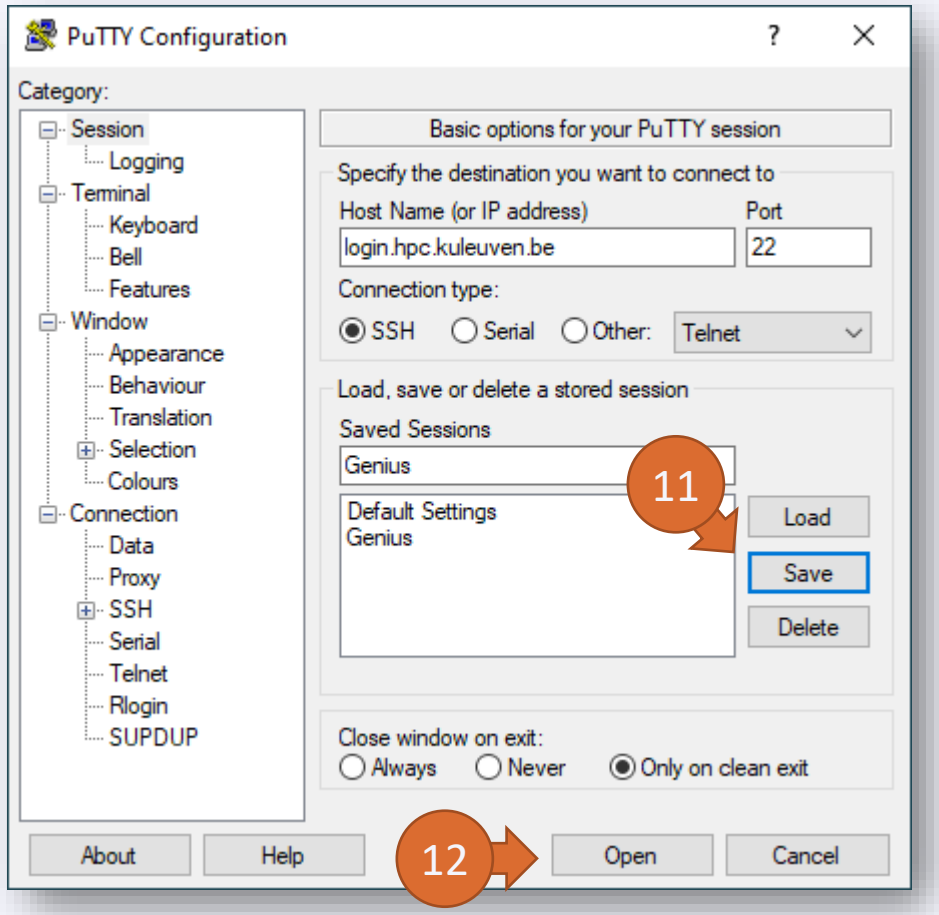


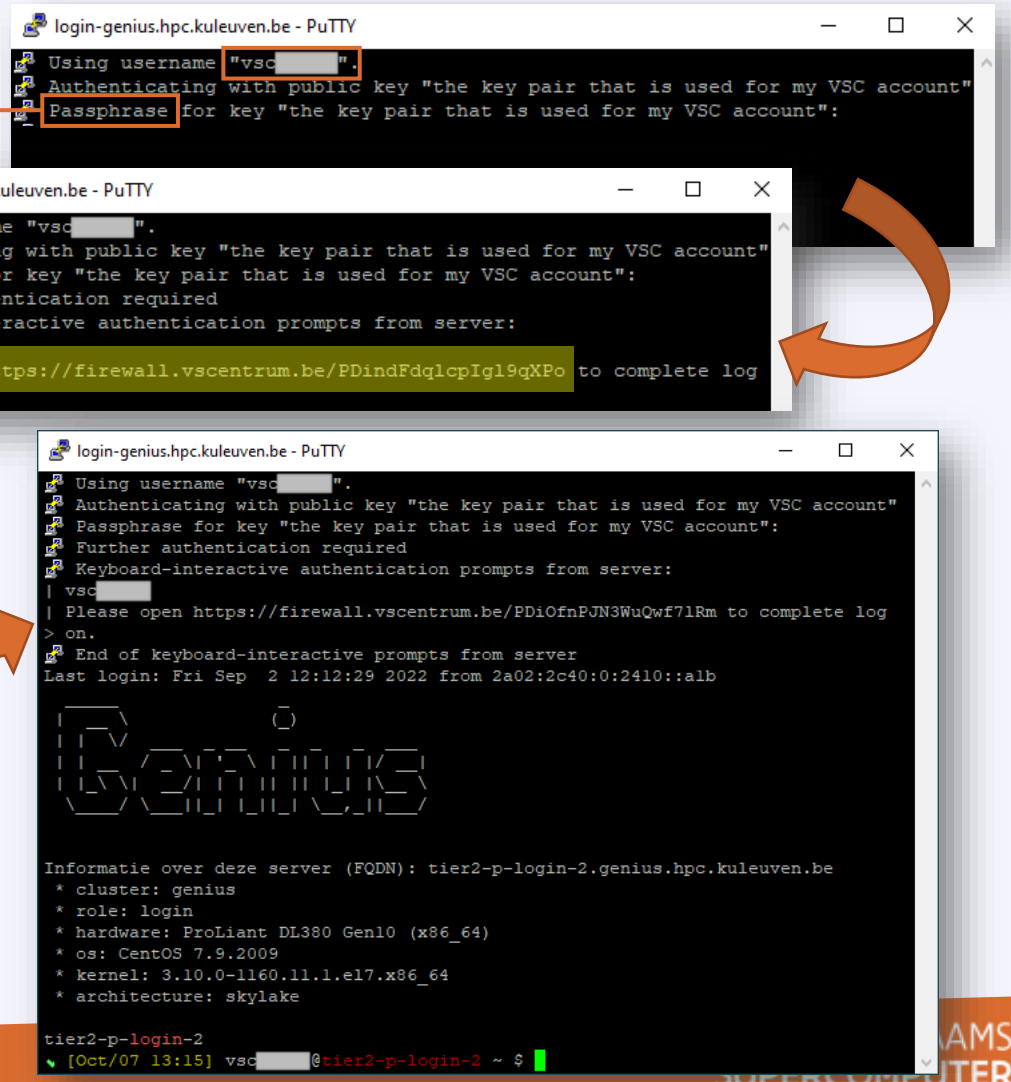


Setup PuTTY in 12 Clicks

(Windows only)

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





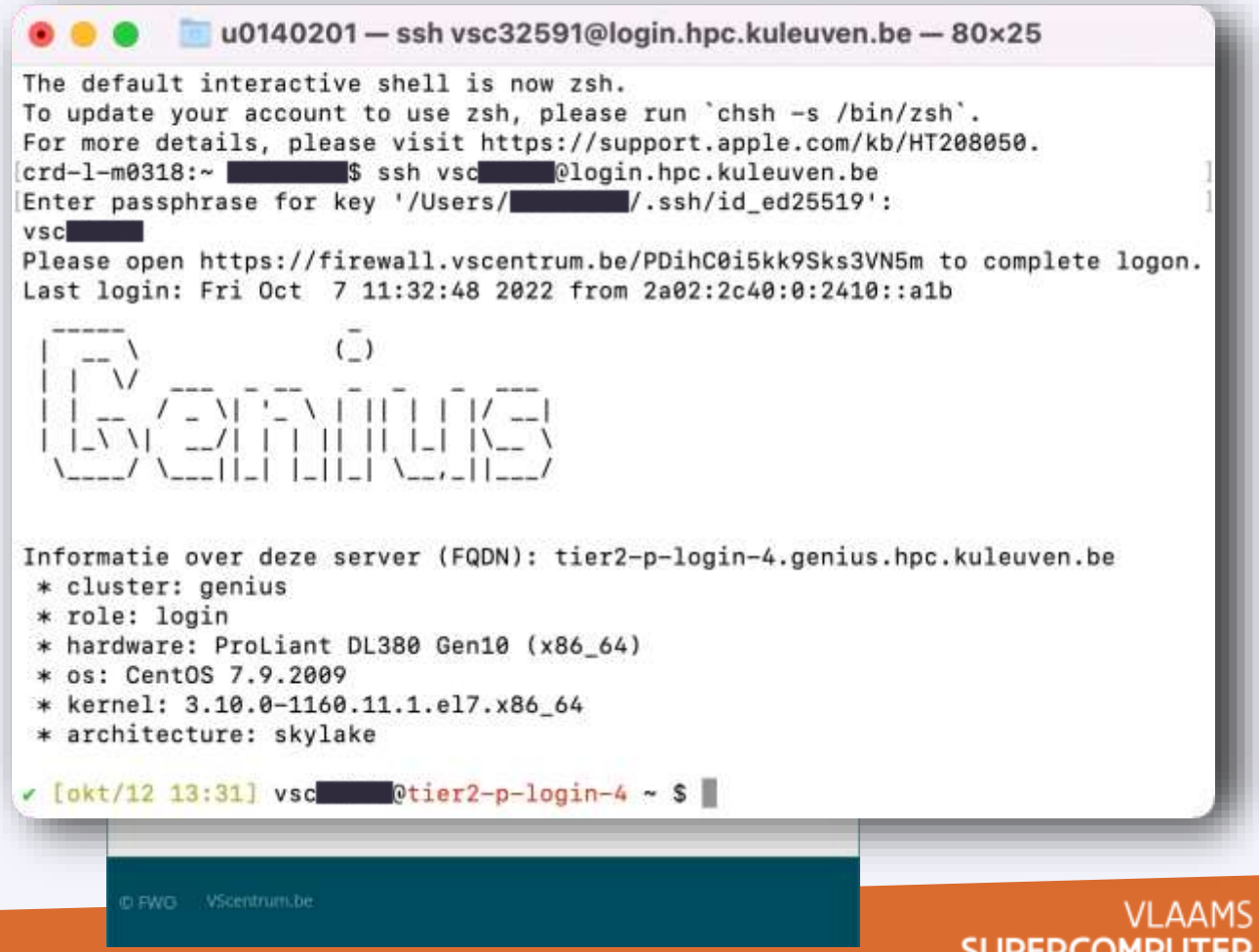
Connecting via Terminal

(Linux and Mac)

Host Name:
`login.hpc.kuleuven.be`

- Use ssh to connect:
  `$ ssh vscXXXXX@<host_name>`
- If key not found:
  `$ ssh -I </path/to/keyfile> ...`

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

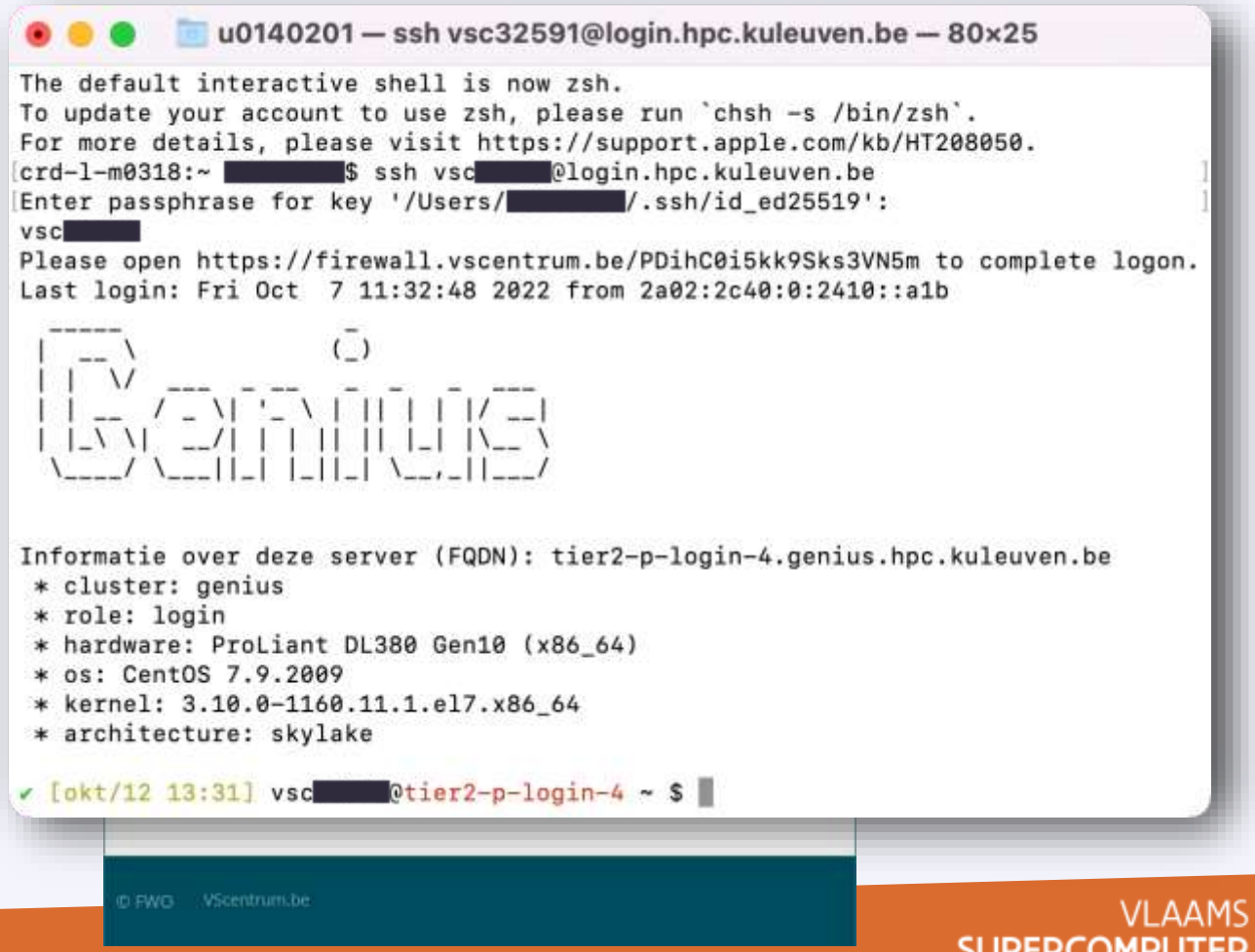


Connecting via Terminal

With SSH Agent (Linux and Mac)

Host Name:
`login.hpc.kuleuven.be`

- Check your SSH Agent. Is your SSH key found?
  `$ ssh-add -l`
- If your SSH Agent is not running:
  `eval $(ssh-agent)`
- If your key is not found, add it to the Agent:
  `$ ssh-add </path/to/keyfile>`
- Use ssh to connect:
  `$ ssh vscXXXXX@<hostname>`



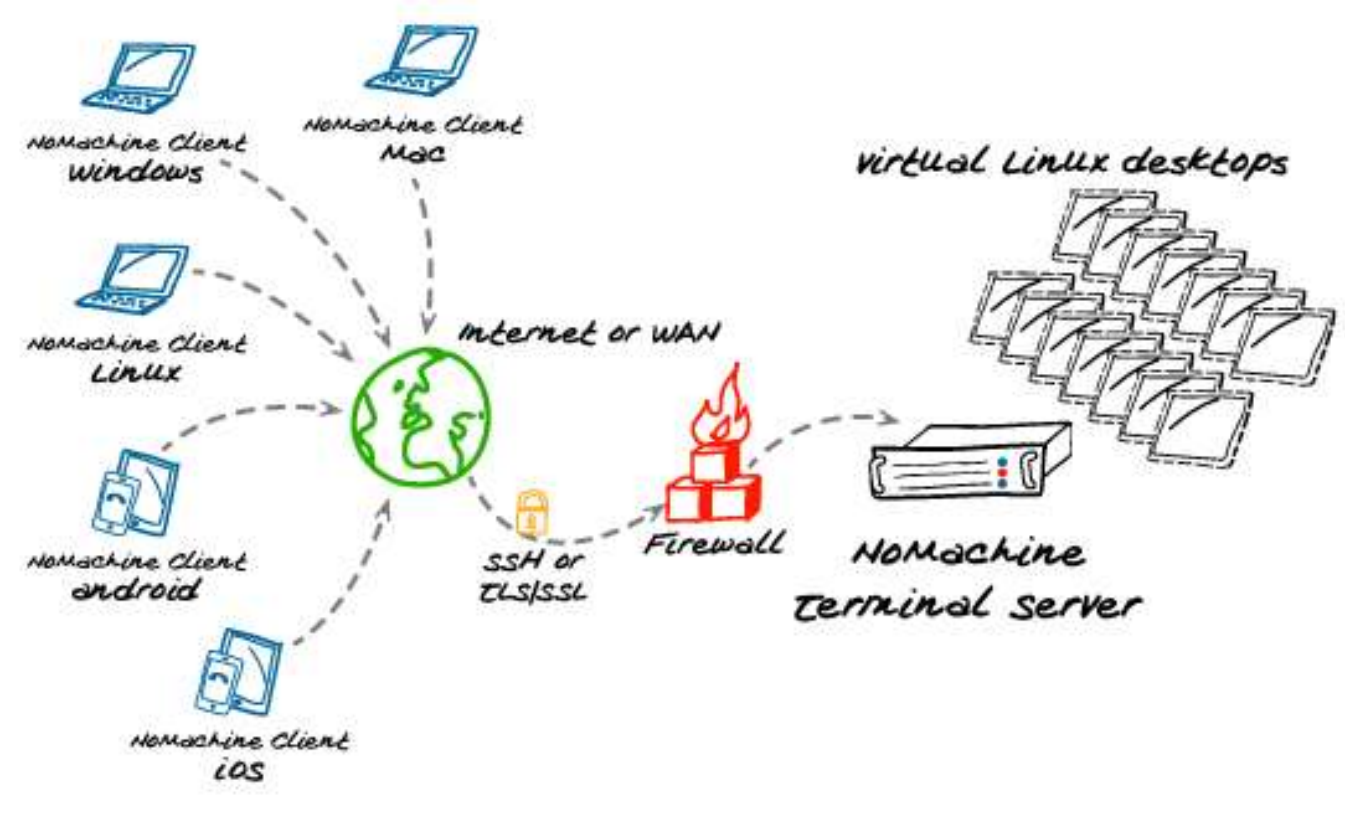
NX

- ❑ About
- ❑ Setup

NX – The Graphical Login

NX is:

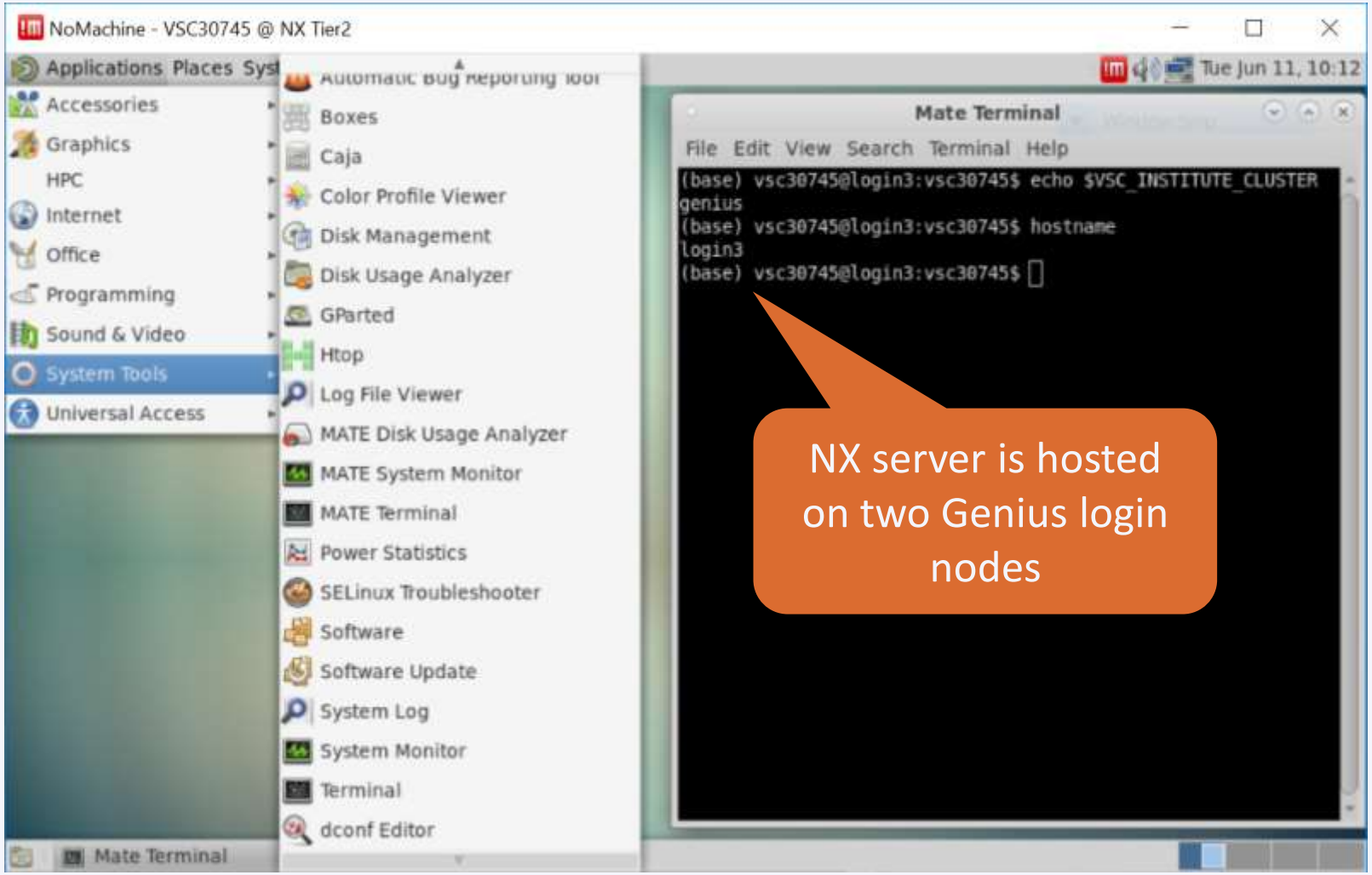
- ❑ a graphical remote desktop server
- ❑ login to VSC
- ❑ start a virtual Linux instance(s)
- ❑ Client: NoMachine
- ❑ Configuration Guide
- ❑ Using OpenSSH key
  Convert your key

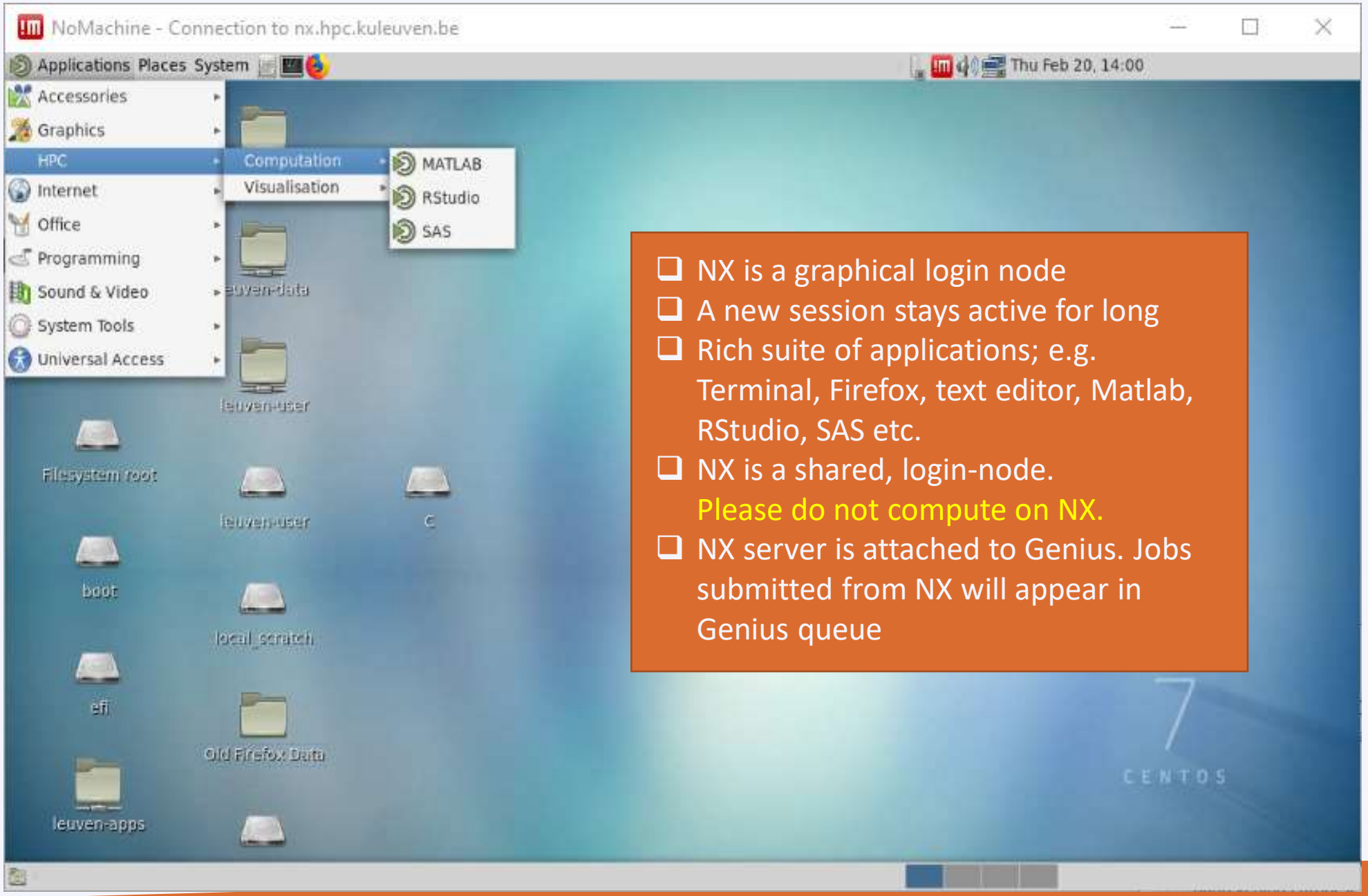


Advantages of NX

- ❑ 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



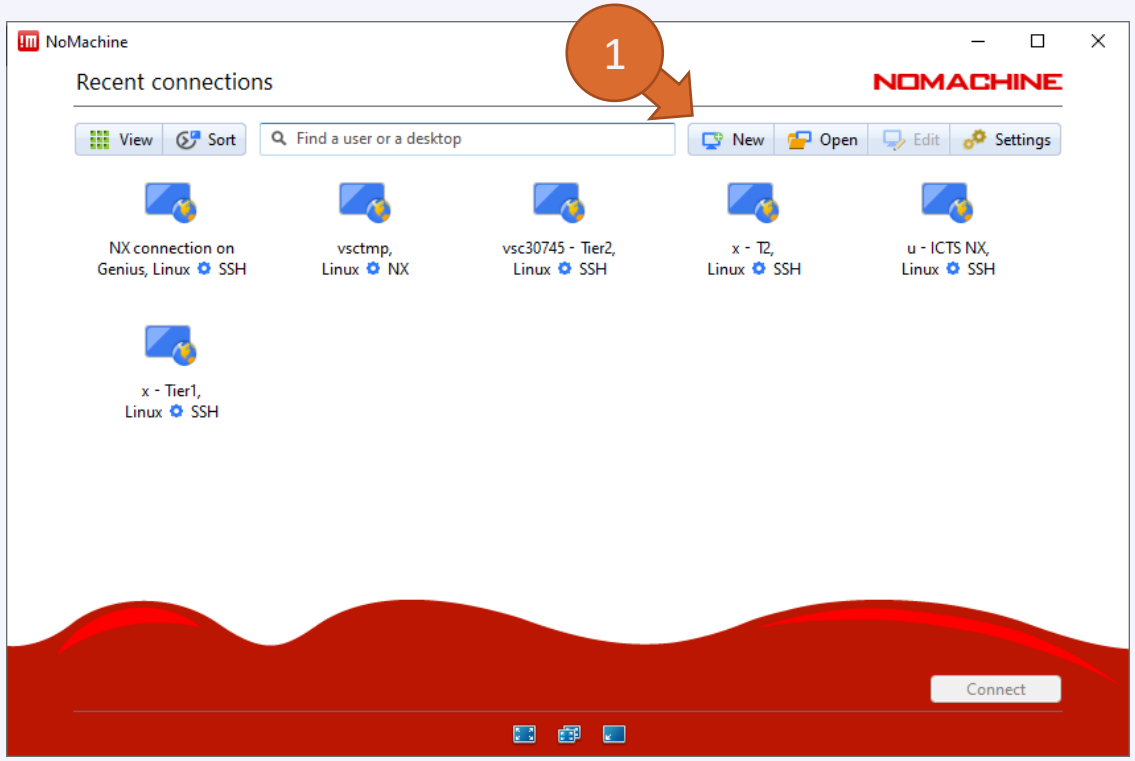


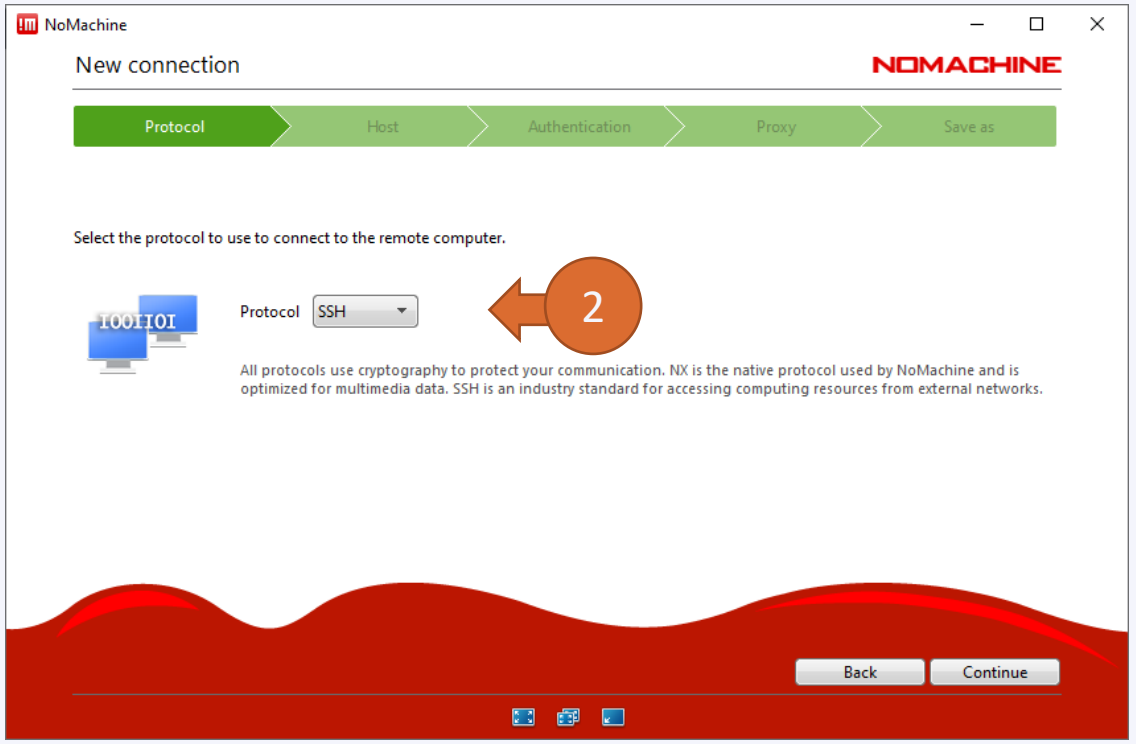
- NX is a graphical login node
- A new session stays active for long
- Rich suite of applications; e.g. Terminal, Firefox, text editor, Matlab, RStudio, SAS etc.
- NX is a shared, login-node. Please do not compute on NX.
- NX server is attached to Genius. Jobs submitted from NX will appear in Genius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

NX: available software

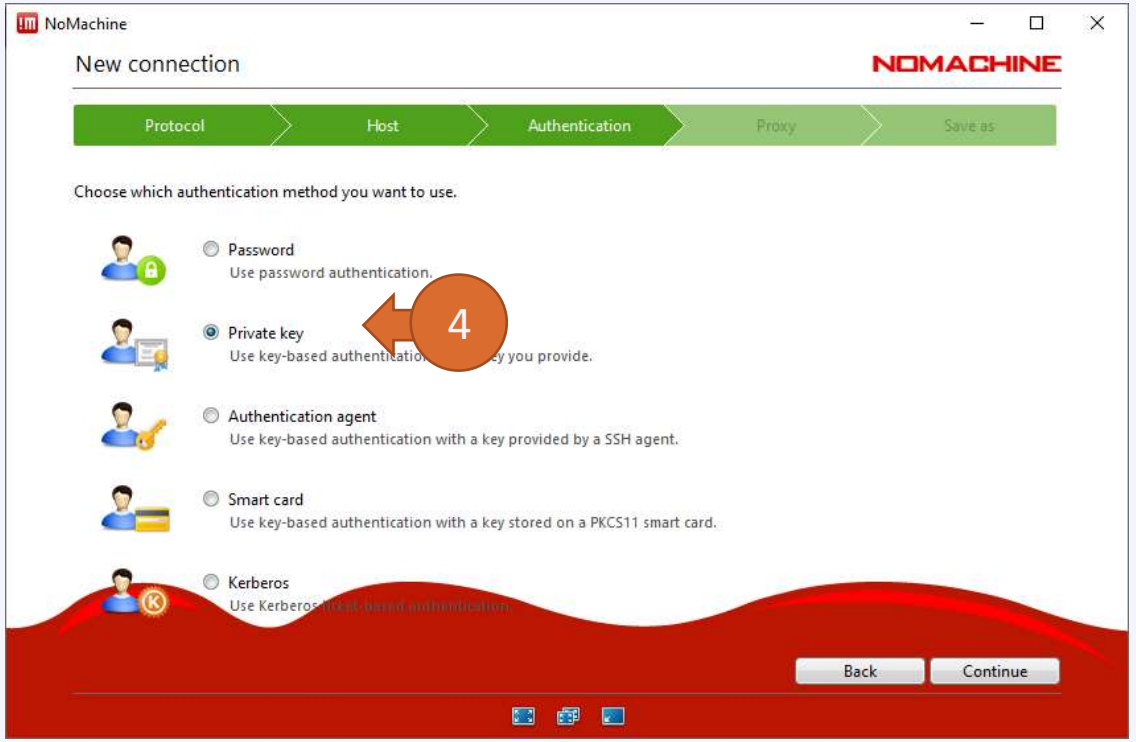
- **Accesories**: Gedit, Vi IMproved, Emacs (dummy version), Calculator
- **Graphics**: gThumb (picture viewer), Xpdf Viewer
- **Internet**: Firefox
- **HPC**: **Computation**: Matlab (2018a), RStudio, SAS
- **Visualisation**: Paraview, VisIt, VMD
- **Programming**: Meld Diff Viewer (visual diff and 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

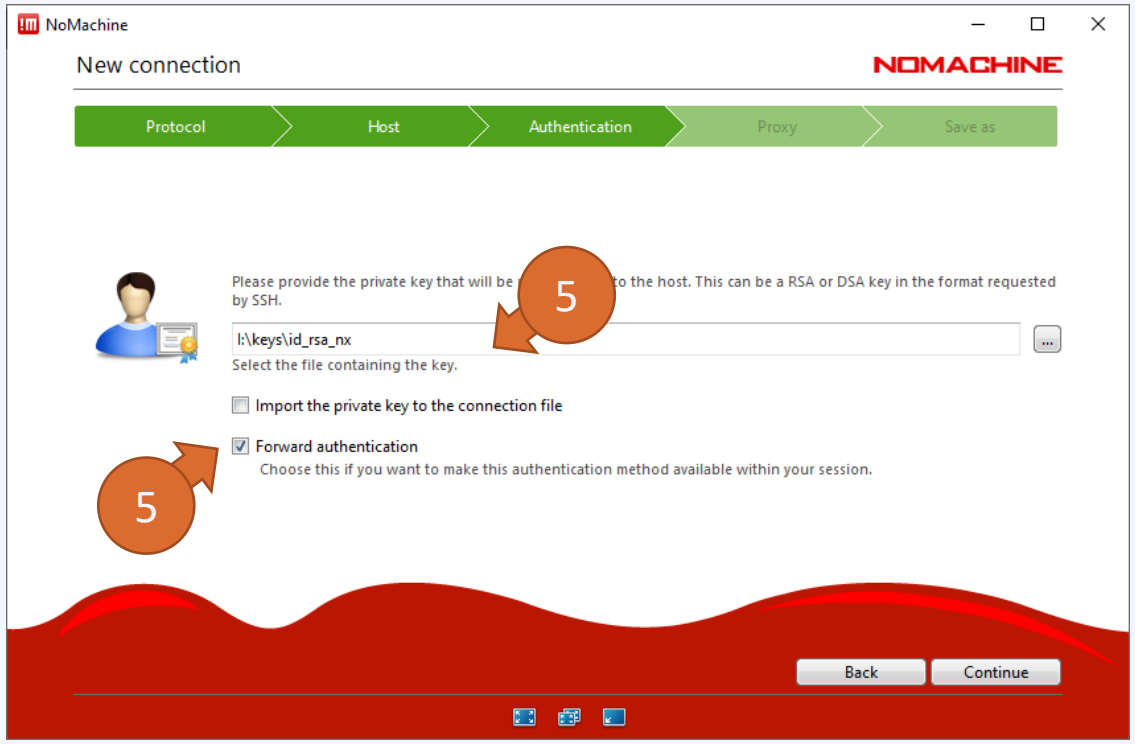


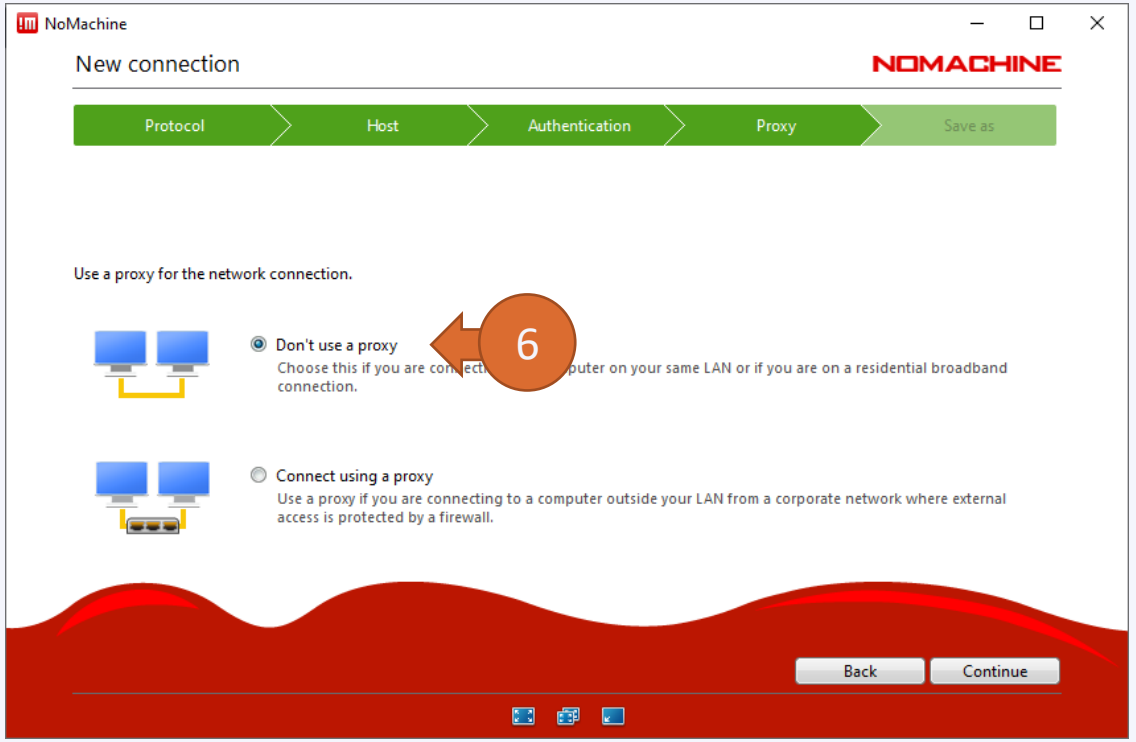


Setup NX in 10 Steps

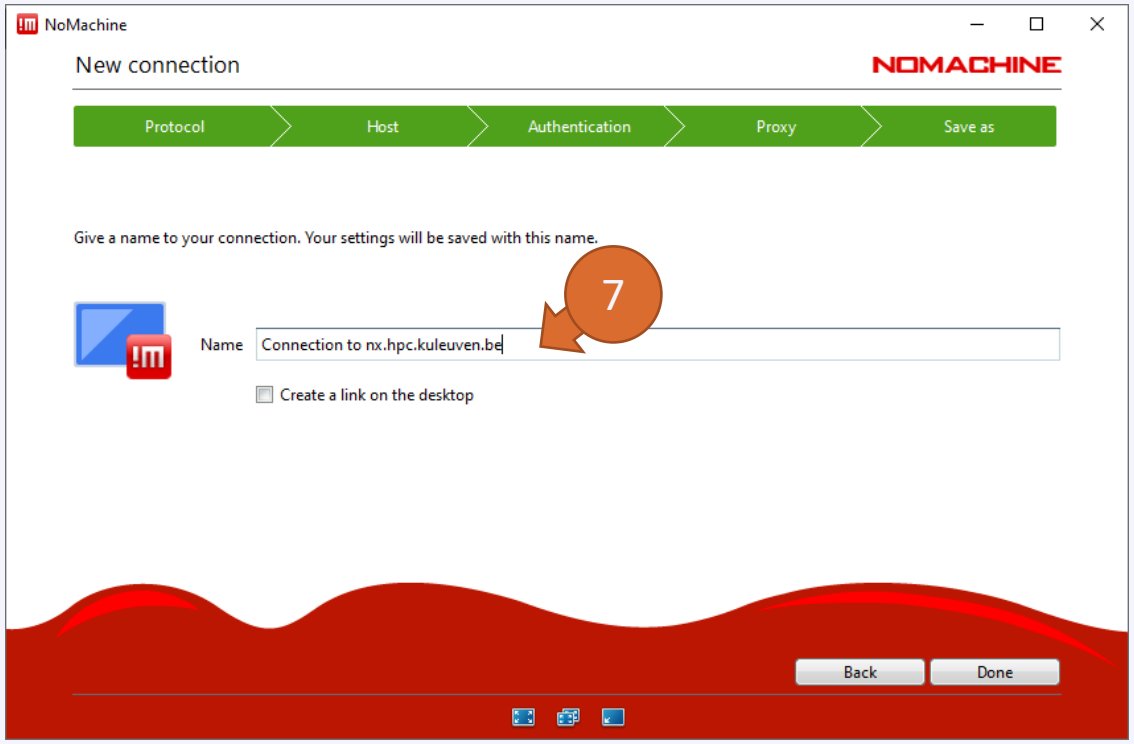


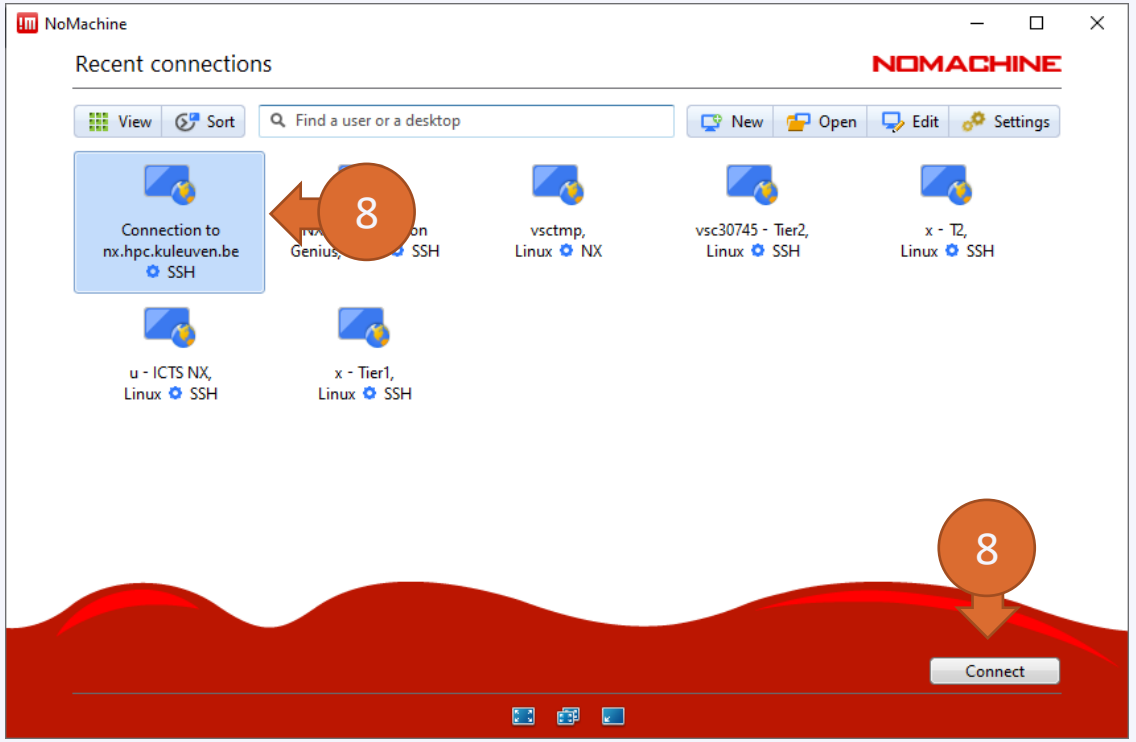
Setup NX in 10 Steps



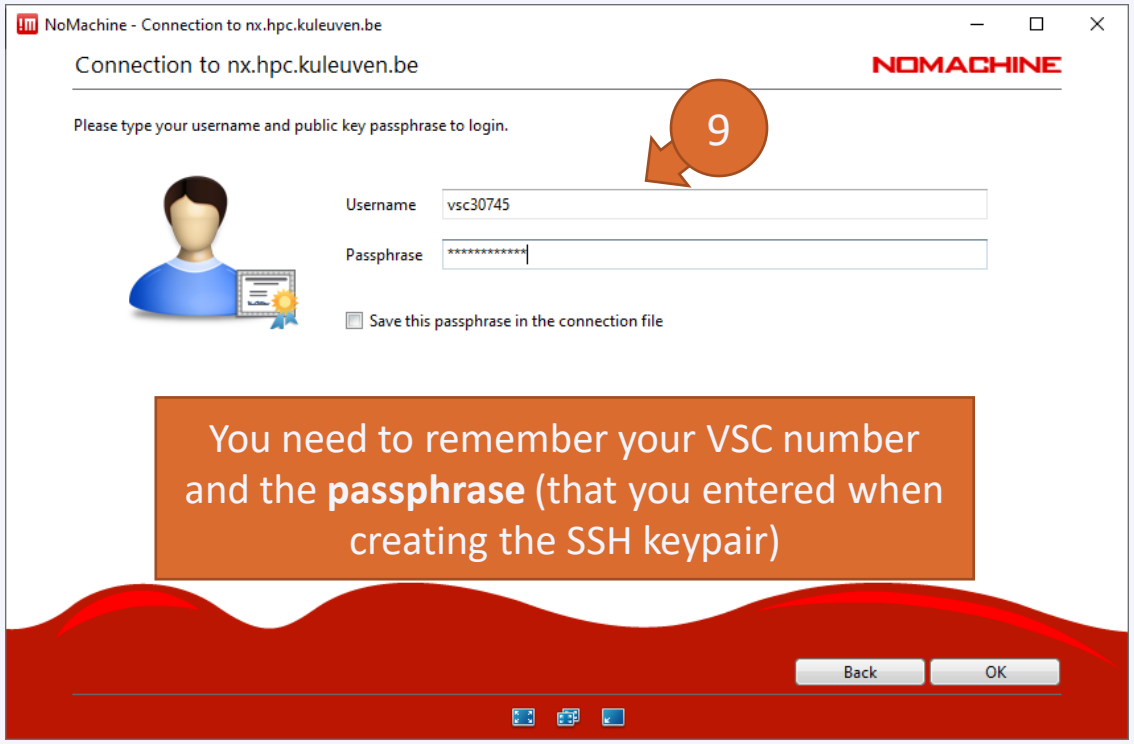


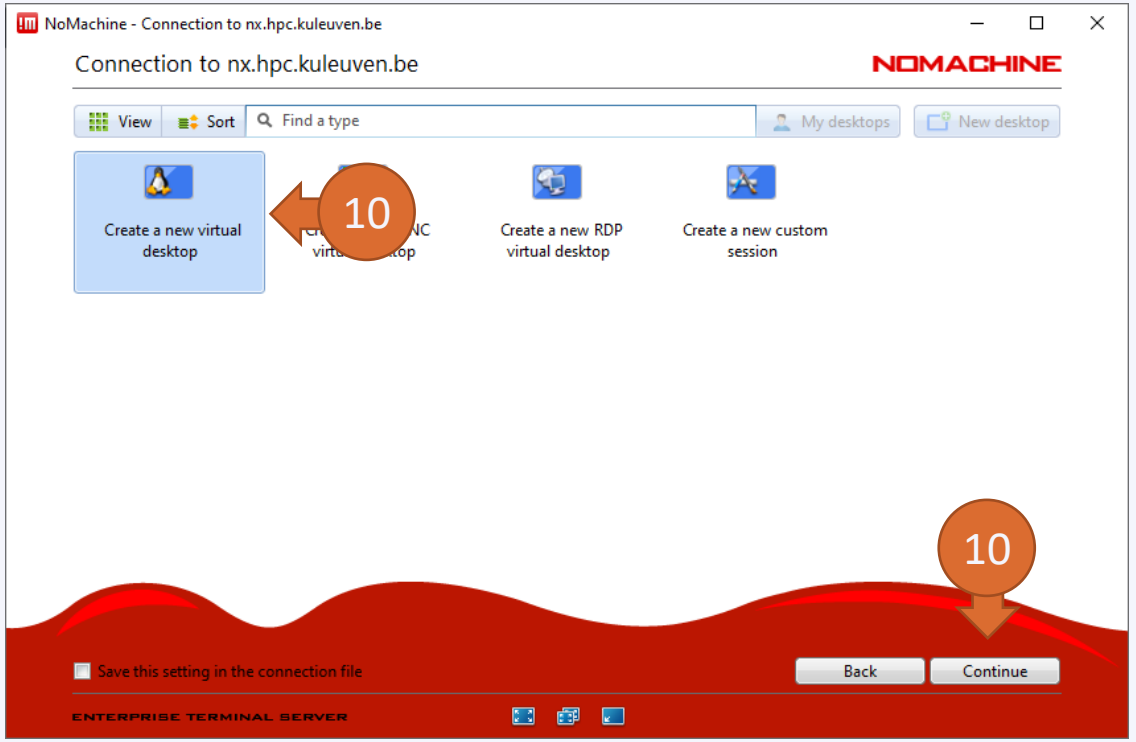
Setup NX in 10 Steps





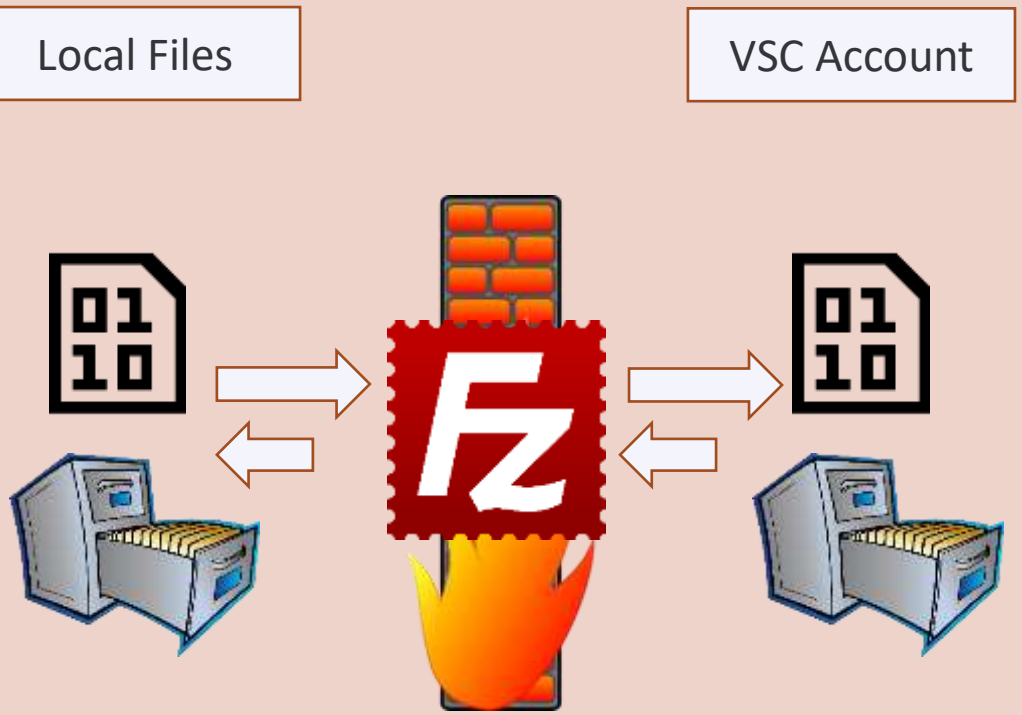
Setup NX in 10 Steps





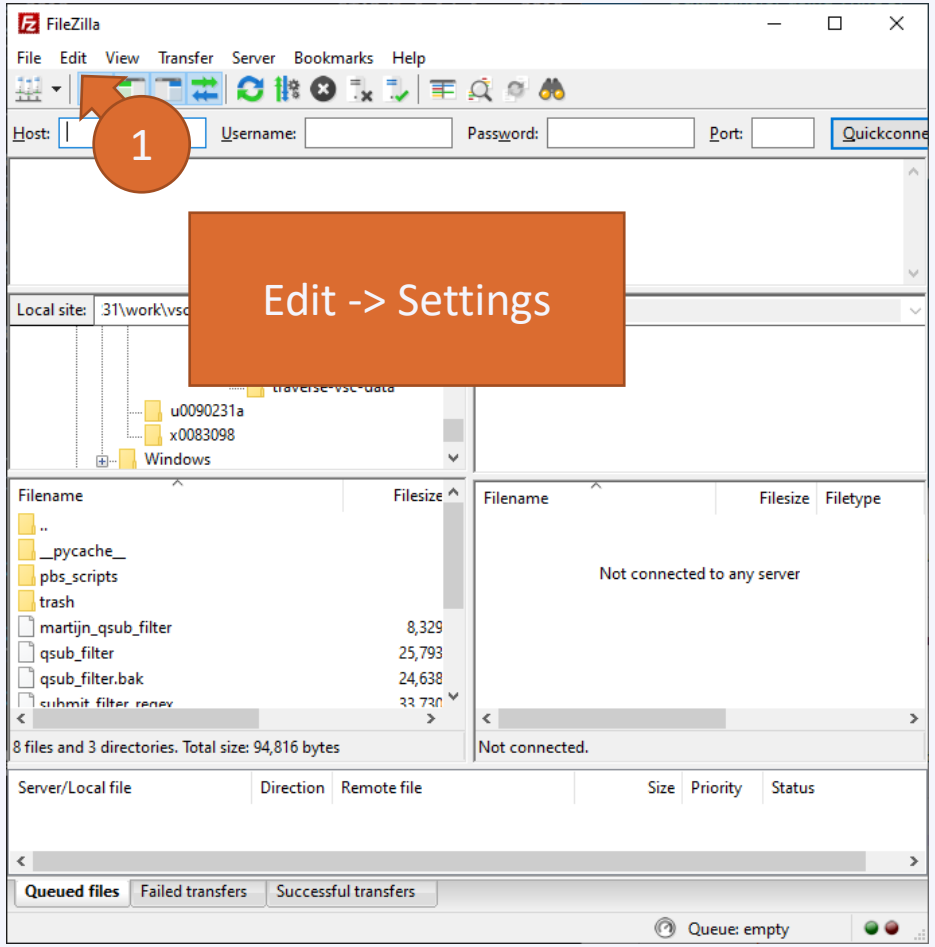
File Transfer with FileZilla

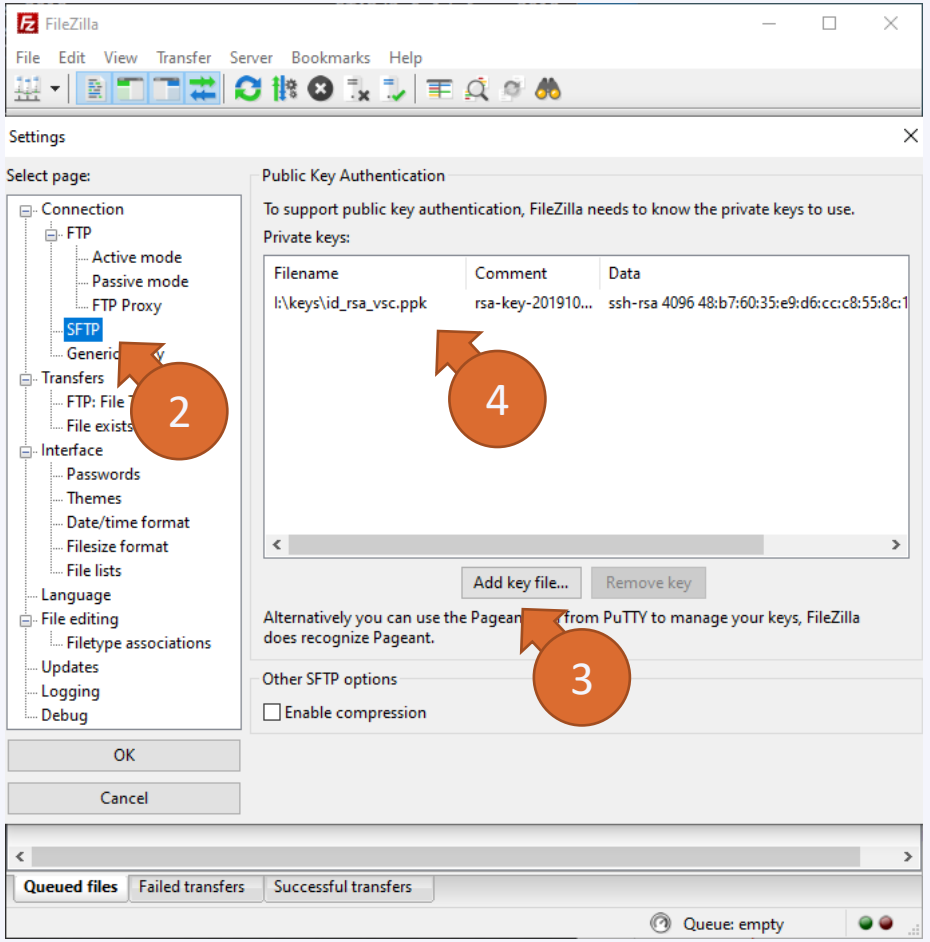
File transfer



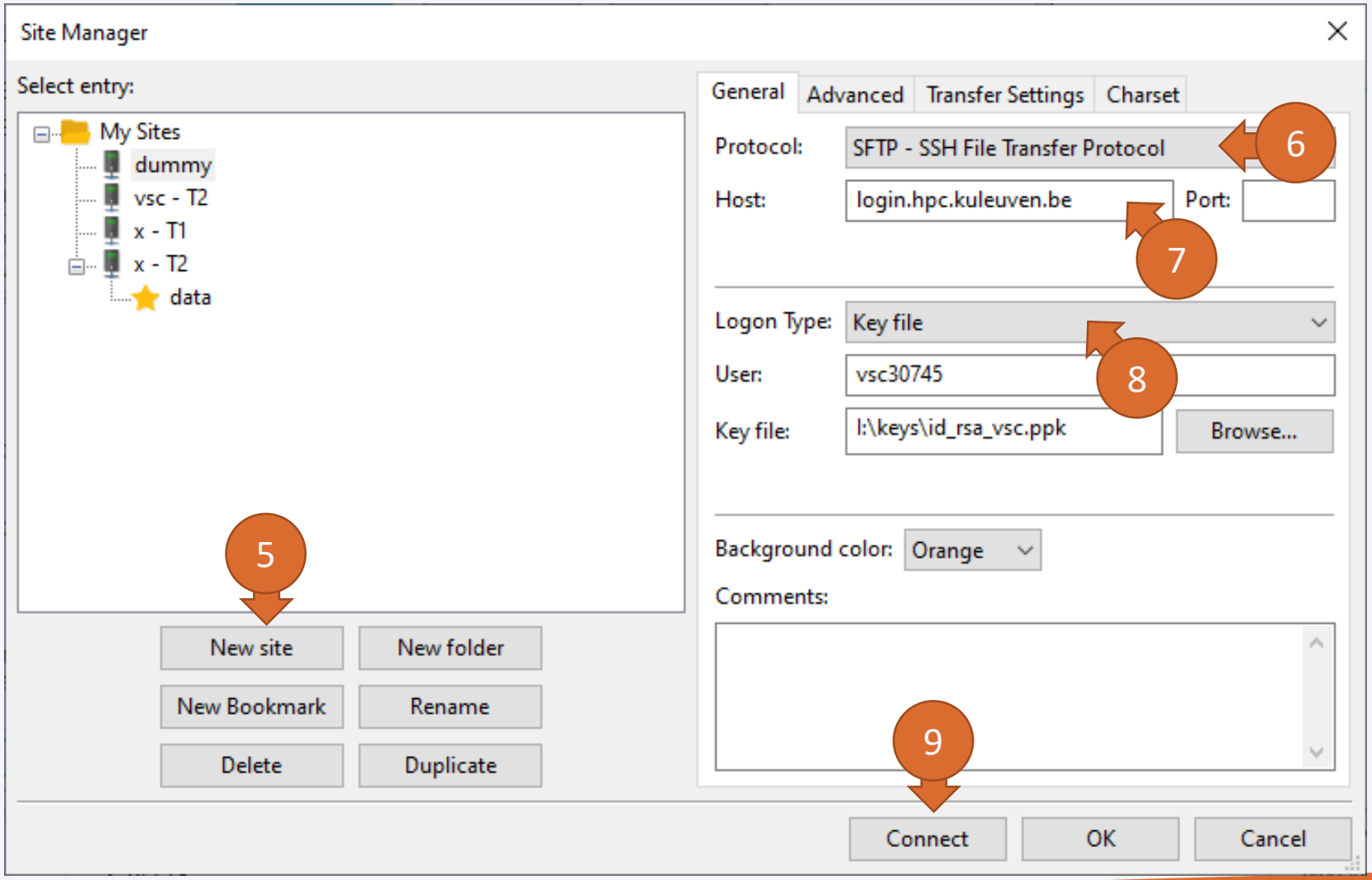
Application	OS
FileZilla	Windows, Linux, Mac
WinSCP	Windows
rsync, scp	Linux, Mac
Globus	Window, Linux, Mac

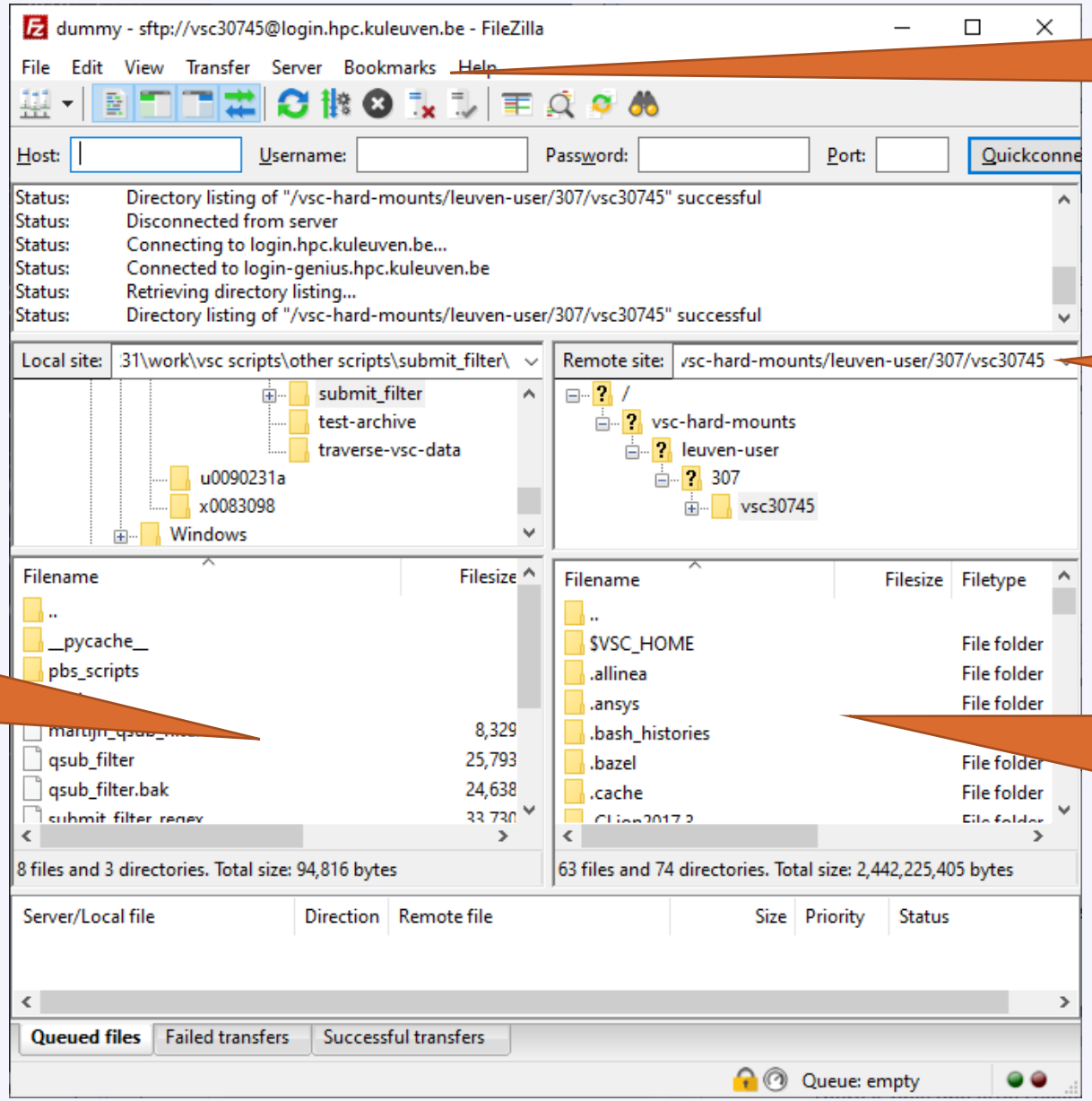
File transfer





File transfer





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

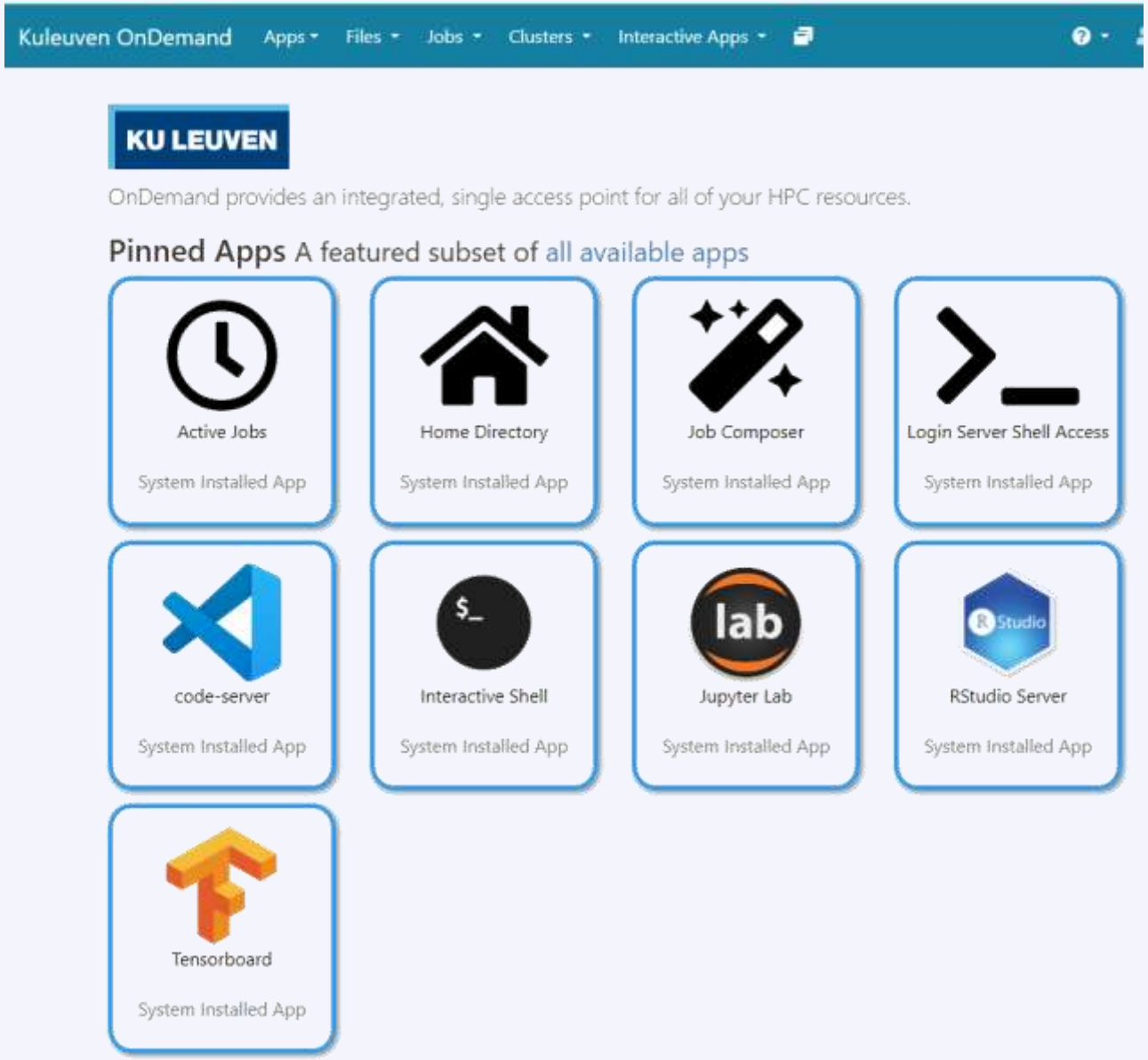
Instead, go to your `$VSC_DATA` folder, e.g. /data/leuven/399/vsc39934

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!

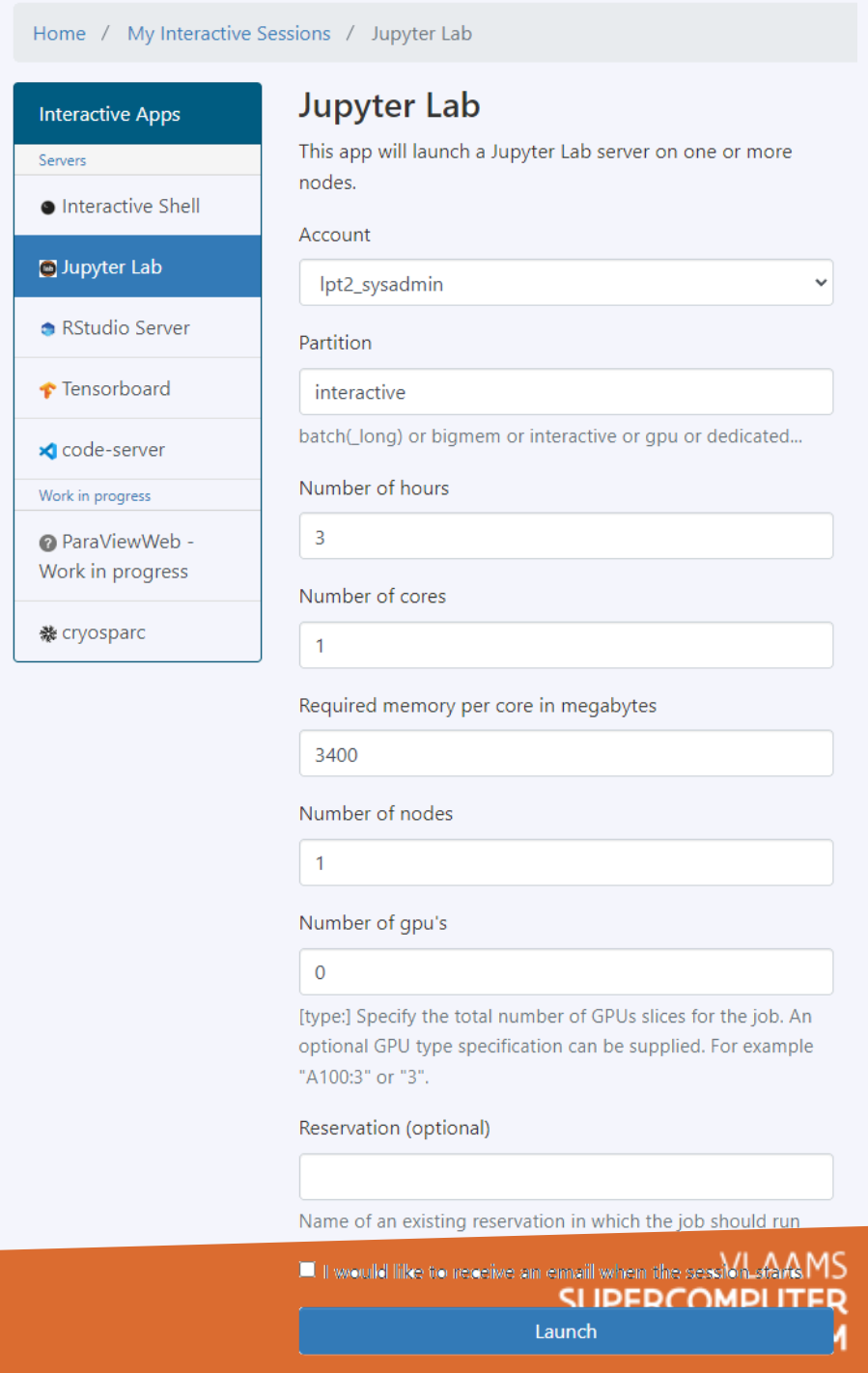
Open OnDemand

- Access wICE via web browser
- https://ondemand.hpc.kuleuven.be
- Login via KULeuven MFA
- File browser
- Interactive apps + integrated shell
- Create, start and monitor jobs
- Develop, compile and test your code
- VSCode editor
- Jupyter Lab, RStudio and Tensorboard



Open OnDemand

- E.g. to start a Jupyter Notebook
- Pick a valid Slurm credit account
- Default partition: batch
  interactive: Max 8 cores, 1 GPU, 16 hr
- Default resources:
  1 core, 1 hour, 3400MB RAM, no GPU
- (At the moment) you cannot use scratch and staging



Software Stack

Software: Available Modules

- OS: Linux CentOS 7.x (Genius) and Rocky Linux 8.x (wICE)
- Toolchains (Genius):
  Intel 2018a (icc, icpc, ifort; Intel MPI; Intel MKL)
  FOSS 2018a (gcc, g++, gfortran; OpenMPI; ScaLAPACK, OpenBLAS, FFTW)
- wICE: default is 2021a
- 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 use /apps/leuven/icelake/2022b/modules/all`	Expose other modules from non-default toolchains, by including them in the search
`module spider Python`	Get more info
`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

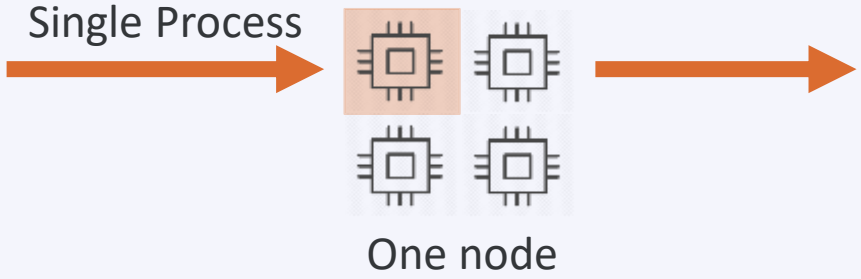
- You can always install your desired software in your $VSC_DATA
  Use Intel or FOSS toolchains
- Compile your code on a compute node (with interactive job)
- If you cannot, ask us for help
- Python/R packages for AI and ML must be installed by the users themselves
- Read more about Python Package Management
- Read more about R Package Management

Starting to Compute

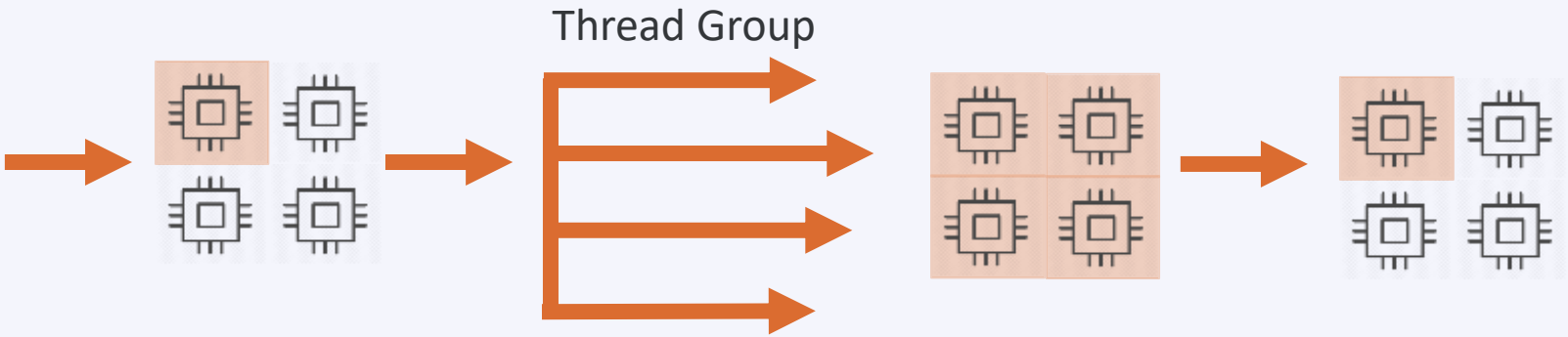
Resource Glossary

- ❑ **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?
- ❑ **Storage**: how much storage (data, scratch, etc) the job needs?
- ❑ **Credits**: how many compute credits will be consumed?

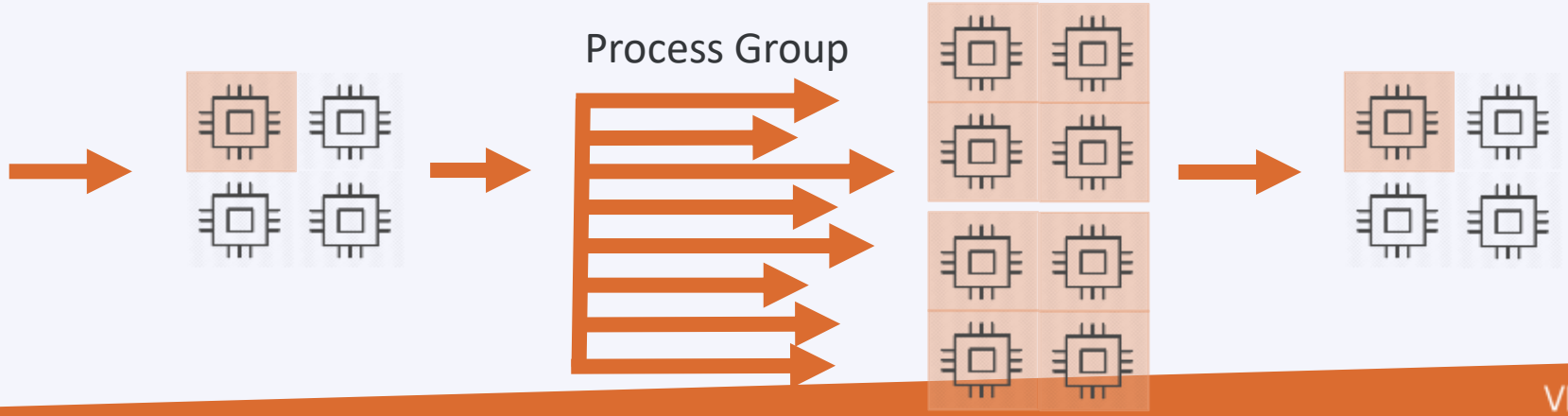
Serial Application
(1 process on 1 core)



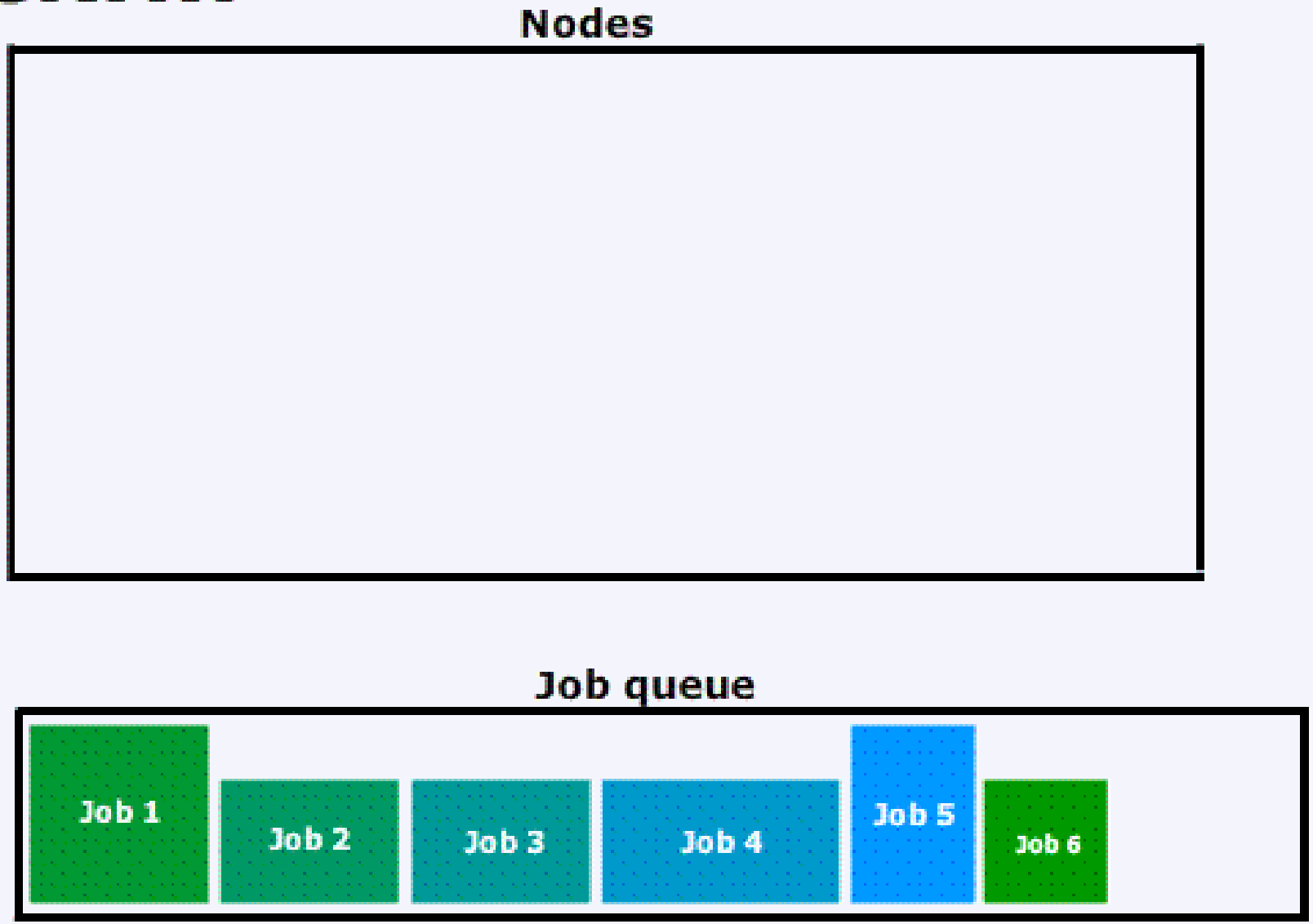
Multi-Core Appl.
(N threads on N cores from **1 node**)



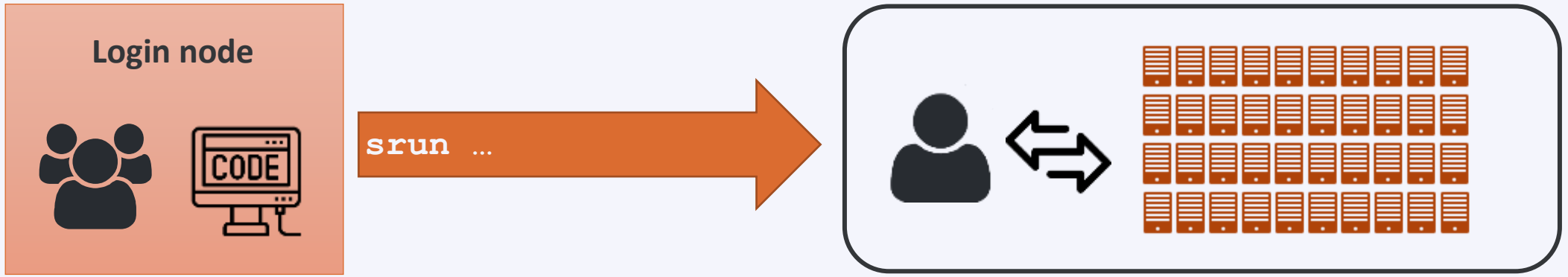
Distributed Appl.
(many processes on multiple cores/nodes)



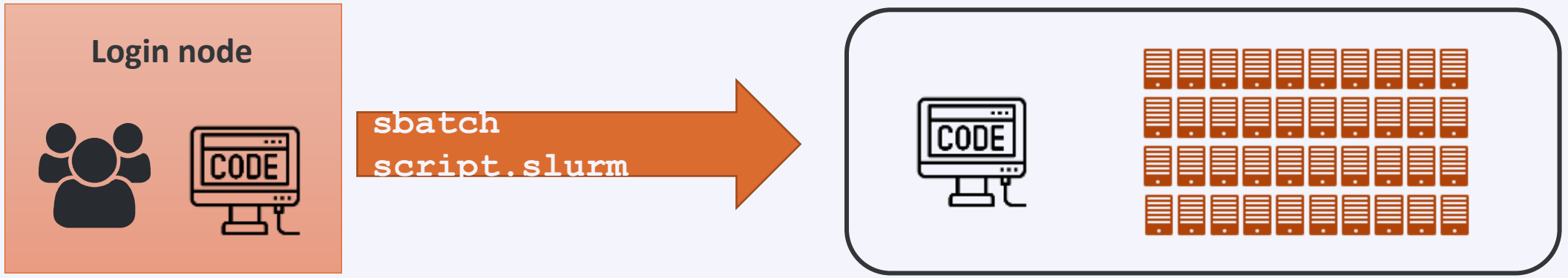
Backfill



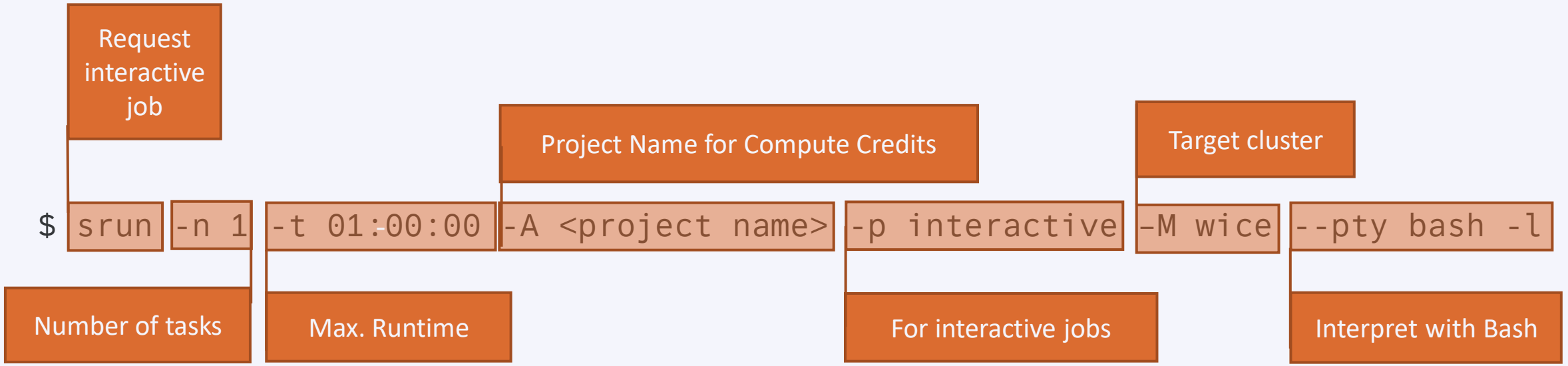
Interactive Job



Batch Job



Interactive Job on **wICE**



Remark

- Specifying <project name> for credits is mandatory, *e.g.*: **`-A lp_hpcinfo`**
- Implicit defaults for an interactive job on wICE are: `-n 1, -t 01:00:00, --mem-per-cpu=2000M`

Jobs: Slurm submit options

SLURM	Remarks

`--nodes=X --ntasks-per-node=Y`
`--ntasks=X*Y`
`--ntasks=X --cpus-per-task=Y`
	or
	or
	(for Hybrid MPI/OpenMP only)

`--partition=gpu`	Default: "batch" partition
`--mem-per-cpu=XG`	Default: average memory/core
`--time=hh:mm:ss`	
`-J jobname`	
`-o <file_template>`	output
`-e <file_template>`	Default: same as stdout
`--mail-type=FAIL,BEGIN,END`	
`--mail-user=<mailaddress>`	
`--export=ALL,key=value`	Additional variables to pass to job
`-A account`	Mandatory (credits)

Argument Shorthands

Some of the `sbatch`, `srun` and `salloc` command line arguments have shorthands

Shorthand	Full Argument	Meaning
`-A`	`--account`	Slurm account name
`-a`	`--array`	Job array range
`-M`	`--cluster`	Machine or cluster name
`-c`	`--cpus-per-task`	Num cores per task (default=1)
`-d`	`--dependency`	Job dependency (after, afterok, afterany, …)
`-I`	`--input`	STDIN filename
`-e`	`--error`	STDERR filename
`-o`	`--output`	STDOUT filename
`-G`	`--gpus`	Num GPUs per job
`-t`	`--time`	Maximum walltime
`-N`	`--nodes`	Minimum num nodes
`-n`	`--ntasks`	Maximum num tasks
`-p`	`--partition`	Partition name

Jobs: Defaults

If no explicit resources are given your job will start

- with 1 node 1 core,
- with walltime of 1hr
- in batch partition
- jobs with no cluster given start on Genius cluster (and get job-id ~760)
- soon the Slurm accounts will be available also on Genius (with the same account name)

Interactive Jobs

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

```
$ srun -M wice -A lp_hpcinfo --pty /bin/bash -l
```

- Interactive job with X-forwarding

```
$ srun -M wice -A lp_hpcinfo -x11 --pty /bin/bash -l
```

- Request fraction of a node from interactive partition

```
$ srun -M wice -A lp_hpcinfo -N 1 -n 4 -p interactive --pty /bin/bash -l
```

- Request a GPU accelerator

```
$ srun -M wice -A lp_hpcinfo -N1 -n 18 -p gpu -G 1 --pty /bin/bash -l
```

Example: Slurm Job Script

```
#!/bin/bash -l

#SBATCH --cluster wice
#SBATCH --account lp_hpcinfo
#SBATCH --nodes 1
#SBATCH --ntasks 4
#SBATCH --mem-per-cpu 4G
#SBATCH --job-name hpc_workflow
#SBATCH --mail-type BEGIN,END,FAIL
#SBATCH --mail-user my.name@kuleuven.be

Module --force purge
module load intel/2021a
module load Python/3.9.5-GCCcore-10.3.0

which python3

cd $VSC_SCRATCH/projects/simulations

cp -r $VSC_DATA/input_data .


python modelling.py

cp -r output_data $VSC_DATA

rm -rf ./input_data ./output_data
```

Shebang

Resource List

Module load(s)

Move data

Execure commands

Move data

Batch Workload

- Submit the job to the batch server
- Receive a unique JobID
- Error and output files

Job Script

```
#!/bin/bash -l
…
```

Command Line

```
$ sbatch simulation.slurm
```

JobID

```
Submitted batch job 60042478 on cluster wice
```

stderr, stdout

```
$ ls *.out
slurm-60042478.out
```

Output File

- STDERR and STDOUT are by default redirected to a single file:
- `slurm-<JobID>.out`
- Contains job info, all errors and warnings, and printouts
- Always study it
- Address all warnings and errors (if you can)
- Typical error examples …

STDOUT + STDERR

```
$ ls slurm-*.out
slurm-60042478.out
```

Out of Memory

```
slurmstepd: error: Detected 1 oom-kill event(s).
Some of your processes may have been killed by the
cgroup out-of-memory handler.
```

Short Walltime

```
slurmstepd: error: *** JOB 60042478 ON s28c11n2
CANCELLED AT 2023-02-08T10:03:43 DUE TO TIME LIMIT
***
```

Low Disk Space

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

Output File

- Always created
- `slurm-<job_id>.out`
- Contains all standard output and error (instead of screen)
- Always study it
- Standard Output and Error channels can be redirected to other files:

```
#SBATCH --output …
#SBATCH --error …
```

stdout

```
$ ls slurm-*.out
slurm-60041238.out
```

Output File

```
SLURM_JOB_ID: 60033947
SLURM_JOB_USER: vscXXXXX
SLURM_JOB_ACCOUNT: lp_wice_pilot
SLURM_JOB_NAME: testjob
SLURM_CLUSTER_NAME: wice
SLURM_JOB_PARTITION: batch
SLURM_NNODES: 1
SLURM_NODELIST: m28c30n4
SLURM_JOB_CPUS_PER_NODE: 72
Date: Tue Jan 10 17:02:04 CET 2023
Walltime: 00-01:00:00
==================================
/apps/leuven/icelake/2021a/software/intel-compilers/2021.2.0/compiler/2021.2.0/linux/bin/intel64/icc
cp: cannot stat '/apps/leuven/training/test': No such file ordirectory
Hello World
```

Resource Summary

Stdout

Inspecting Jobs

Example

```
$ scontrol show job -M wice 60049330
JobId=60049330 JobName=f70.slurm
   UserId=vsc3...(253...) GroupId=vsc3...(253...)
   Account=lp_metacommunity_models QOS=lp_metacommunity_models
   JobState=PENDING Reason=QOSGrpBillingMinutes
   SubmitTime=2023-02-16T00:24:58 EligibleTime=2023-02-16T00:24:58
   Partition=batch NodeList= NumNodes=4-4 NumCPUs=288 NumTasks=288 CPUs/Task=1
   TRES=cpu=288,mem=979200M,node=4,billing=733
   MinCPUsNode=72 MinMemoryCPU=3400M
   WorkDir=/vsc-hard-mounts/leuven-data/3../vsc3.../Odonate_SOM
   StdErr=/vsc-hard-mounts/leuven-data/3../vsc3.../Odonate_SOM/slurm-60049330.out
   StdIn=/dev/null
   StdOut=/vsc-hard-mounts/leuven-data/3../vsc3.../Odonate_SOM/slurm-60049330.out
```

Diagnosis

Why is my job in pending/hold state?
Check out the “Reason” on Slurm docs: https://slurm.schedmd.com/resource_limits.html

Other Partitions on wICE

GPU

```
#SBATCH --partition=gpu
#SBATCH --nodes=1
#SBATCH --ntasks=18
#SBATCH --gpus-per-node=1
```

Big Memory

```
#SBATCH --partition=bigmem
#SBATCH --nodes=1
#SBATCH --tasks-per-node=72
#SBATCH --mem-per-cpu=28000M
```

Managing & Monitoring Jobs

Command	Purpose
`sbatch ...`	Submit a batch job
`srun ...`	Submit an interactive job
`scancel --cluster=wice <JobID>`	Cancel a specific pending or running job
`scontrol show job --cluster=wice <JobID>` `slurm_jobinfo <JobID>`	Detailed job info (very useful to diagnose issues)
`squeue --cluster=wice –long`	Status of all recent jobs
`squeue --cluster=wice –start`	Give a *rough* estimate of start time
`sinfo --cluster=wice`	Info about the state of available partitions and nodes
`sacct --cluster=wice --batch --job <JobID>`	Show minimal info about a queue or partition
`slurmtop --cluster=wice`	Overview of the cluster

Interactive Partition

- Accessible via command line and Open OnDemand
- To quickly compile, test, debug your (parallel) application
- To pre-/post-process your data and make visualizations
- Short queue time
- 5 dedicated nodes on wICE
- 64 cores per node, 1 GPU (=7 GPU instances), 512 GB memory
- Max resource per user: 8 cores, 1 GPU instance, 16 hour walltime

Interactive Job

```
$ srun -n 1 -p interactive -t 16:00:00 -G 1 --pty bash -l
```

Singularity Containers

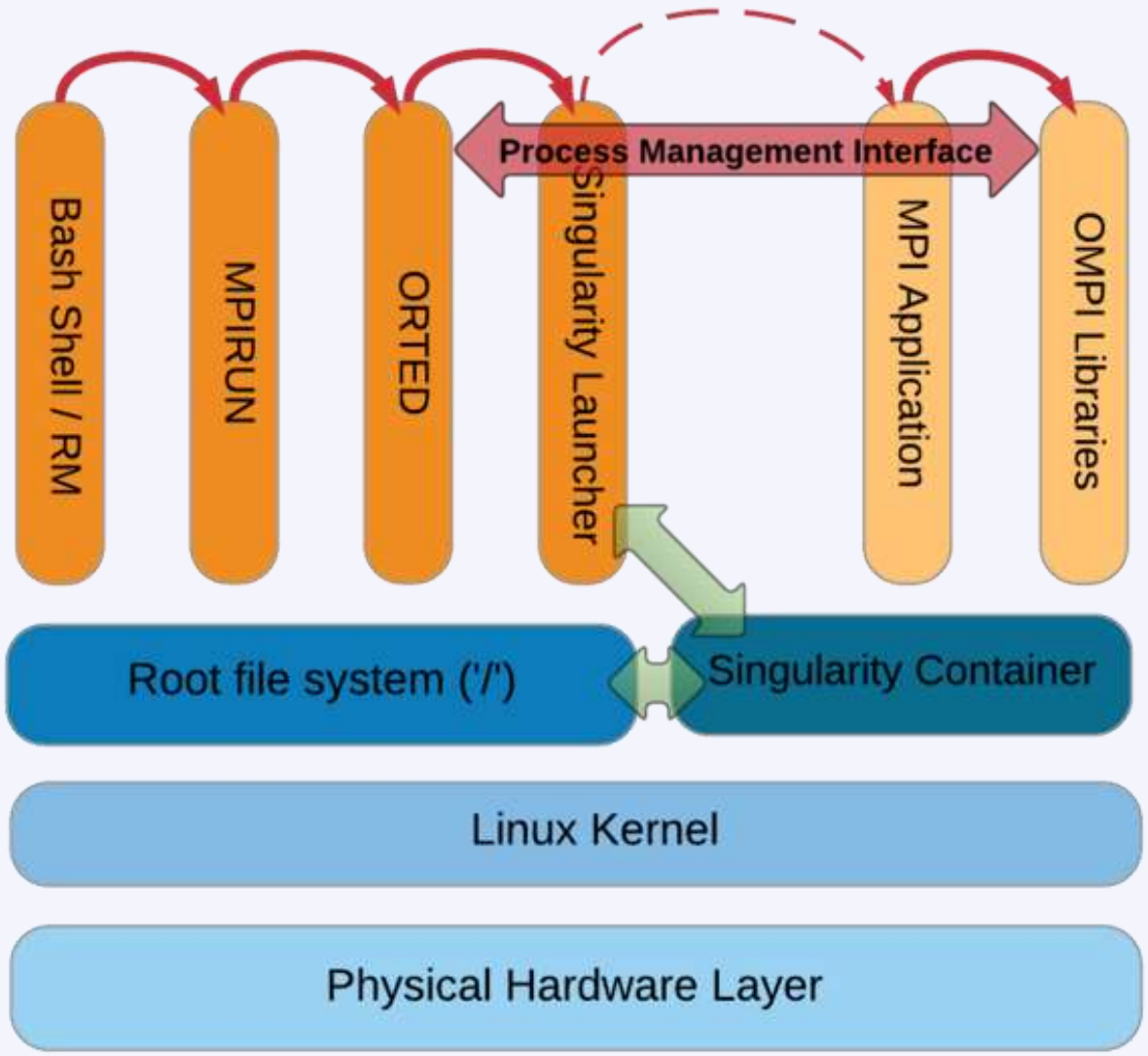
- What?
  + Self-contained OS & software & data
- Why?
  + fully resolved dependency chains
  + portable workflow
- How?
  + You create the image
  + Run it on Genius
  + MPI/OpenMP is supported

Slurm Script

```
#!/bin/bash -l

#SBATCH -t 30:00
#SBATCH -N 2
#SBATCH -n 72
#SBATCH -M wice
#SBATCH -A lp_hpcinfo

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



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 the rates?
Check service catalogue

How to check available credits?

```
$ sam-balance
```

Did the new credits arrive?

```
$ sam-list-allocations
```

Credit Pricing

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

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

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

Cluster / Partition	Credits/hr
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
wICE / Thinnode	10,99
wICE Bigmem	19
wICE / 1 GPU	11,25
wICE 4GPUs	45
wICE Interactive	-

How to Manage Credits?

Command	Purpose
`sam-balance`	List active projects and available credits
`sam-list-allocations`	List the validity dates of different projects/allocations
`sam-list-usagerecords`	List current charge rates for different nodes
`sam-statement -A <account-name>`	Job statements from an account

Conda Package Management: Python and R

Conda on wice

https://docs.vscentrum.be/en/latest/software/python_package_management.html?highlight=iniconda#install-python-packages-using-conda

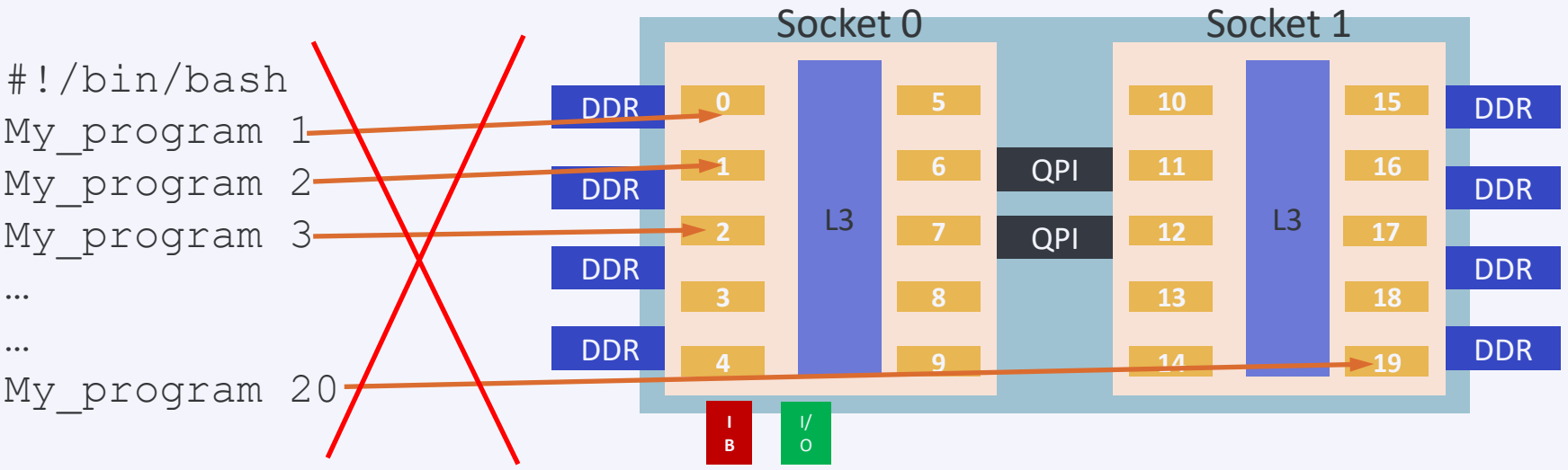
- Good practice would be 2 separate paths of conda installations for each cluster.
- Beginning of the slurm script matters. Use `#!/bin/bash -l` not `#!/bin/bash` to load the env properly.
- Universal .bashrc

```
case ${VSC_INSTITUTE_CLUSTER} in
    genius)
        export PATH="${VSC_DATA}/miniconda3/bin:${PATH}"
        ;;
    wice)
        export PATH="${VSC_DATA}/miniconda3-wice/bin:${PATH}"
        ;;
esac
```

Worker Framework

Sequential…

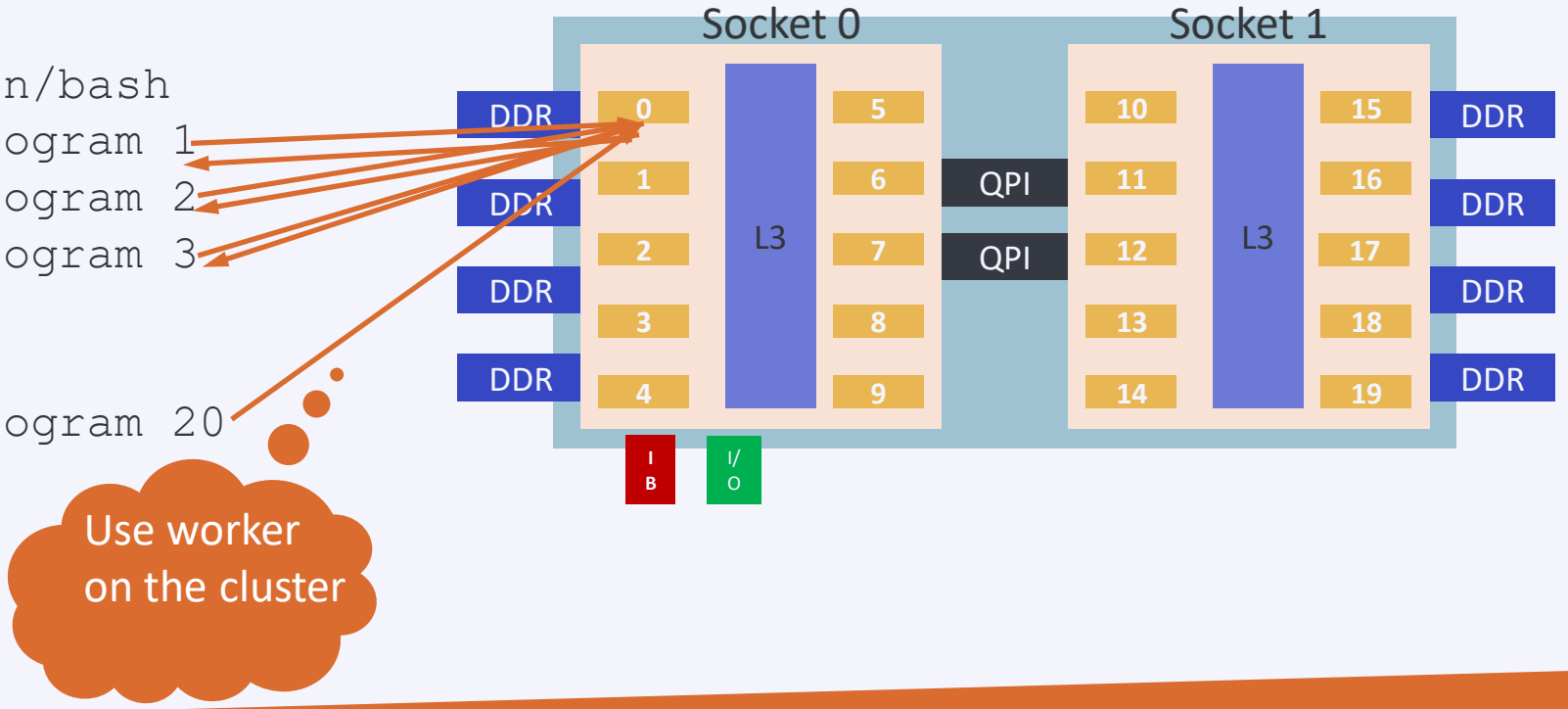
- A shell program consists of a sequential list of commands
- -> bash script on HPC cluster will run sequentially



Sequential…

- A shell program consists of a sequential list of commands
- -> bash script on HPC cluster will run sequentially

```
#!/bin/bash
My_program 1
My_program 2
My_program 3
…
…
My_program 20
```

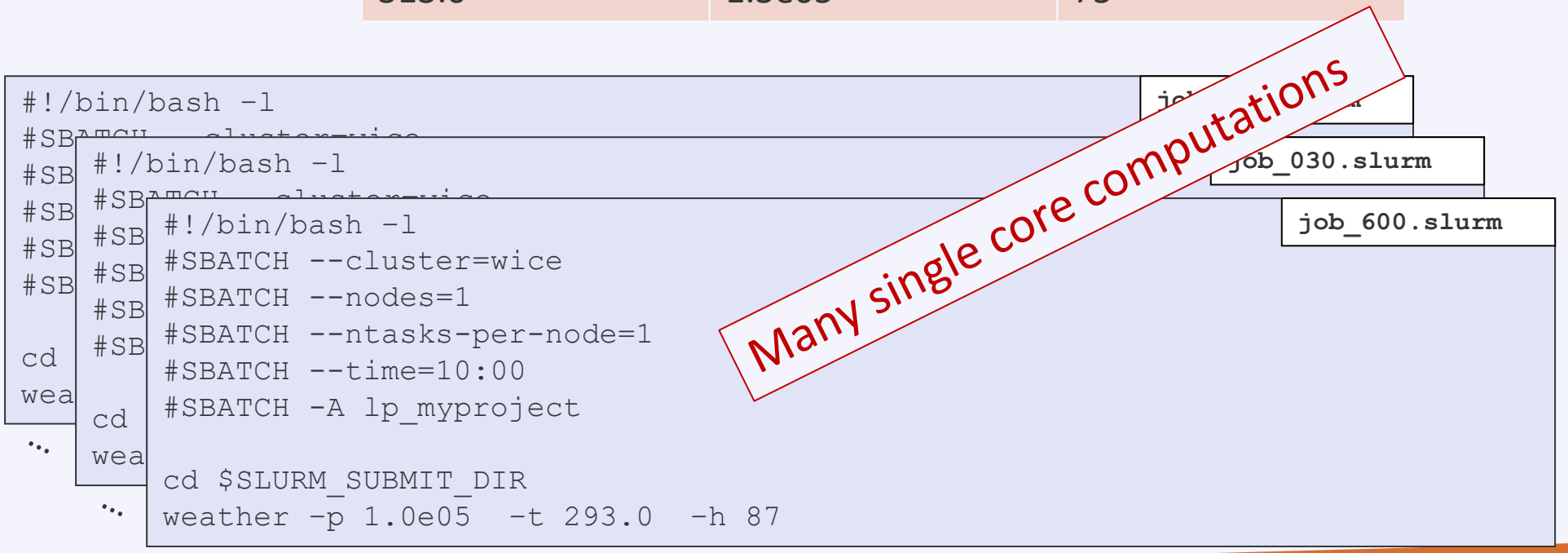


Parallel Computing

- Serial:
  - one program, on one core
- 'Embarrassingly parallel' problems:
  - lots of runs of one program, with different parameters
- Problems that require 'real' parallel algorithms
  - OpenMP
  - MPI : Message Passing Interface

Use case: parameter exploration

temperature	pressure	humidity
293.0	1.0e05	87
…	…	…
313.0	1.3e05	75



VLAAMS SUPERCOMPUTER CENTRUM

Solution: worker with `-data`

data.csv

temperature	pressure	humidity
293.0	1.0e05	87
…	…	…

job.slurm

```
#!/bin/bash -l
#SBATCH --cluster=wice
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=72
#SBATCH --time=01:00:00
#SBATCH -A lp_hpcinfo

cd $SLURM_SUBMIT_DIR
weather -p $pressure -t $temperature  -h $humidity
```

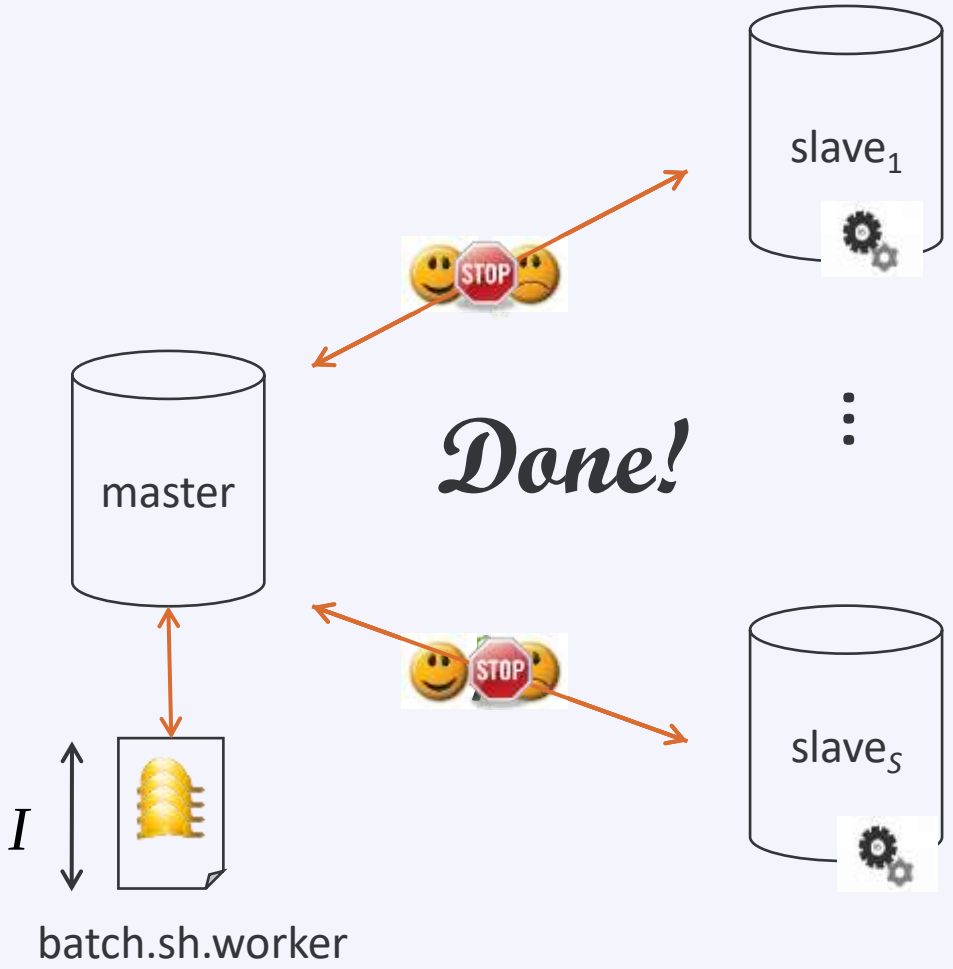
```
#From wICE interactive session:
$ module --force purge
$ module use /apps/leuven/icelake/2021a/modules/all
$ module load worker/1.6.12-foss-2021a-wice
$ wsub  -data data.csv  -batch job.slurm
```

Data exploration: steps

- Write PBS script with parameters
- Create Excel sheet with data
  - Convert to CSV format
- Submit with `wsub`
  - walltime is time to complete all work items

$$walltime_{\text{job}} \geq \frac{N \cdot walltime_{\text{work item}}}{nodes \cdot ppn}$$

worker processing: informally



slave queries for work

master sends work

slave notifies on success/failure, queries for more work

master sends stop

$$nodes = S + 1$$

How to use worker well?

- Many work items, i.e., #work items/#proc >> 1
- time(work item) > 1 minute
- Work item is not multithreaded
- Work item is multithreaded
  - will work, but user must be careful to request the right resources
  - Use `-threaded <n>` flag with `wsub`
- There be dragons: licensing!

worker & multithreading

- Some software uses multithreading automatically, e.g.,
  - R
  - Matlab
- Will use as many threads as there are cores, regardless of system load
  - 36 cores/node
  - 36 work items/node

} 36×36 threads/node

Oversubscription: *very* inefficient!!!

Controlling number of threads

- R, most of the time: `OMP_NUM_THREADS=1`

- Matlab
  - Use `maxNumCompThreads(1)` function call
  - Use compiler flag: `mcc -singleCompThread …`

program_pe.slurm

```
#!/bin/bash -l
#SBATCH -account=lp_myproject
#SBATCH --cluster=wice
#SBATCH --time=1:00:00
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=72

module load R
cd $SLURM_SUBMIT_DIR

export OMP_NUM_THREADS=1
Rscript  program  $a  $b
```

Help on worker

- See documentation
  http://worker.readthedocs.io/
- Each command has help, use `-h`

- All resources have to be specified inside slurm script
- `wresume` does not work yet. Only `wsub` is so far supported
- `-master` flag does not work yet

Submitting worker jobs

From wICE (Interactive) Node

```
$ module av worker
------------------- /apps/leuven/icelake/2021a/modules/all --------------
   worker/1.6.12-foss-2021a
```

From Genius Login Node

```
$ module use /apps/leuven/skylake/2021a/modules/all
$ module av worker
------------------ /apps/leuven/skylake/2021a/modules/all -------------

    worker/1.6.12-foss-2021a-slurm    worker/1.6.12-intel-2021a-slurm
    worker/1.6.12-foss-2021a-wice     worker/1.6.12-foss-2021a
     worker/1.6.12-intel-2021a  (D)
```

Submitting worker jobs

- Let us use one of the worker examples:
  https://github.com/gjbex/worker/tree/development_slurm/examples/bash_example

- alp.slurm

```
#!/bin/bash -l
#SBATCH --nodes=1
#SBATCH --ntasks-per-node 27
#SBATCH --time=00:30:00
#SBATCH -A lpt2_sysadmin
#SBATCH --cluster=wice
cd $SLURM_SUBMIT_DIR
alphabet.sh $letter $counter
```

alpha.csv

```
letter,counter
a,1
b,2
c,3
d,4
e,5
f,6
....
```

Submitting worker jobs

- From Genius

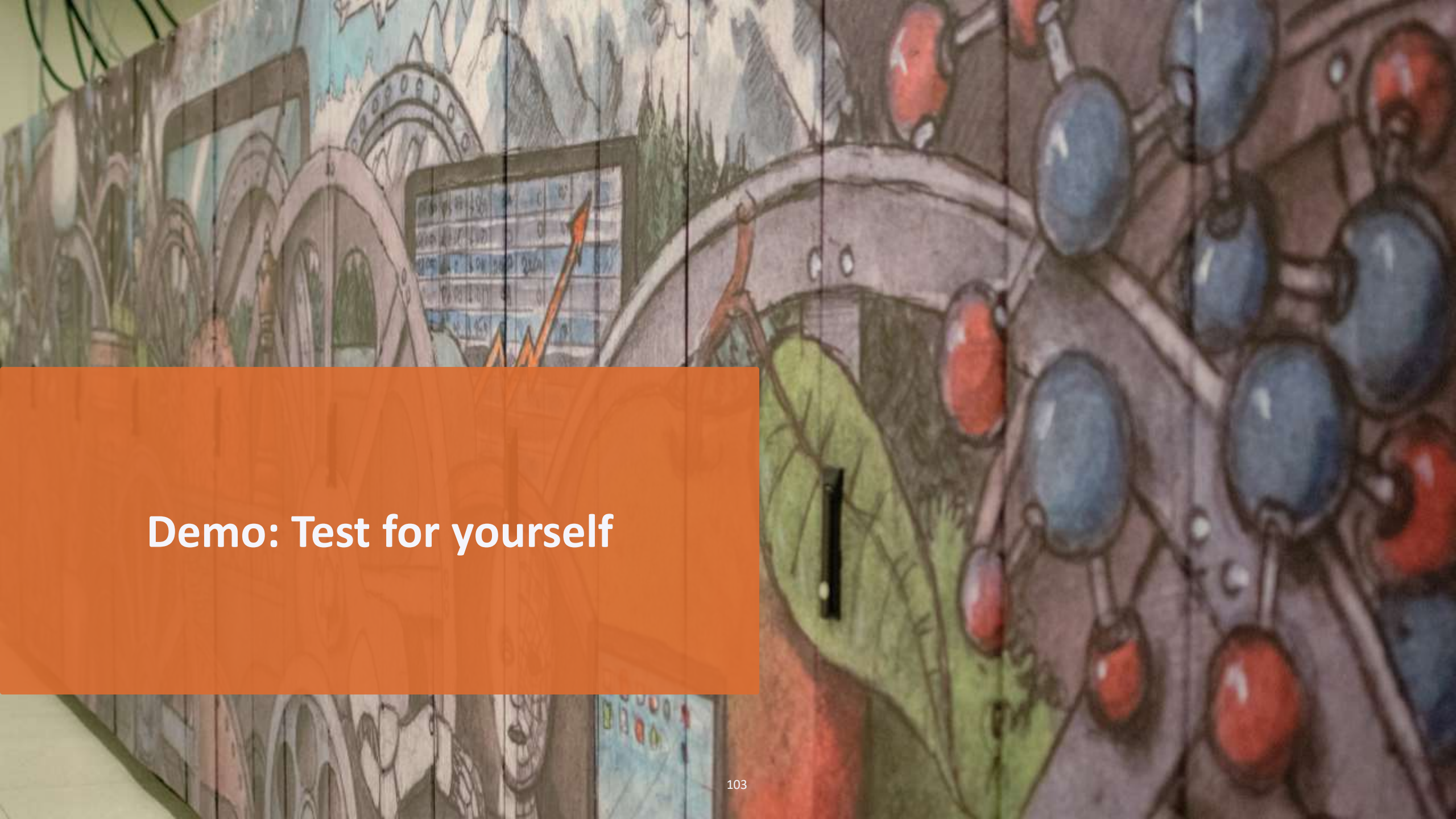
```
#!/bin/bash -l
module purge
module use /apps/leuven/skylake/2021a/modules/all
module load worker/1.6.12-foss-2021a-wice
wsub -batch alp.slurm -data alpha.csv
```

Submitting worker jobs

- From interactive job on wICE

```
#!/bin/bash -l
module --force purge
module use /apps/leuven/icelake/2021a/modules/all
module load worker/1.6.12-foss-2021a
wsub -batch alp.slurm -data alpha.csv
```

```
total number of work items: 26
Submitted batch job 60008889 on cluster wice
```



Demo: Test for yourself

demo/test yourself

- Request membership to lp_hpcinfo 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_DATA`
- Submit `cpujob` to the cluster
- List all your jobs `squeue -M wice`
- Check the information about the cpujob `slurm_jobinfo -M wice <job_ID>`
- Modify the mpi.slurm script to request 1 node, 72 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 a batch GPU job
  While the job is running get the information about the node `slurm_jobinfo...`
  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 – worker

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

Command	Purpose
`$ sbatch ...`	Submit a batch job
`$ srun ...`	Submit an interactive job
`$ scancel --cluster=wice <JobID>`	Cancel a specific pending or running job
`$ scontrol show job --cluster=wice <JobID>` `$ slurm_jobinfo <JobID>`	Detailed job info (very useful to diagnose issues)
`$ squeue --cluster=wice --long`	Status of all recent jobs
`$ squeue --cluster=wice --start`	Give a rough estimate of start time
`$ sinfo --cluster=wice`	Info about the state of available partitions and nodes
`$ sacct --cluster=wice --batch --job <JobID>`	Show minimal info about a queue or partition (-p)
`$ slurmtop` `$ scontrol --cluster=wice show node <hostname>`	Overview of the cluster Get detailed information about the status of a node
`$ sam-balance`	Overview of all your available credit projects
`$ sam-list-allocations`	Detailed overview of your credit allocation history

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://docs.vscentrum.be/en/latest/

VSC agenda: training sessions, events

Systems status page:
http://status.vscentrum.be

Stay Connected to VSC

LinkedIn®