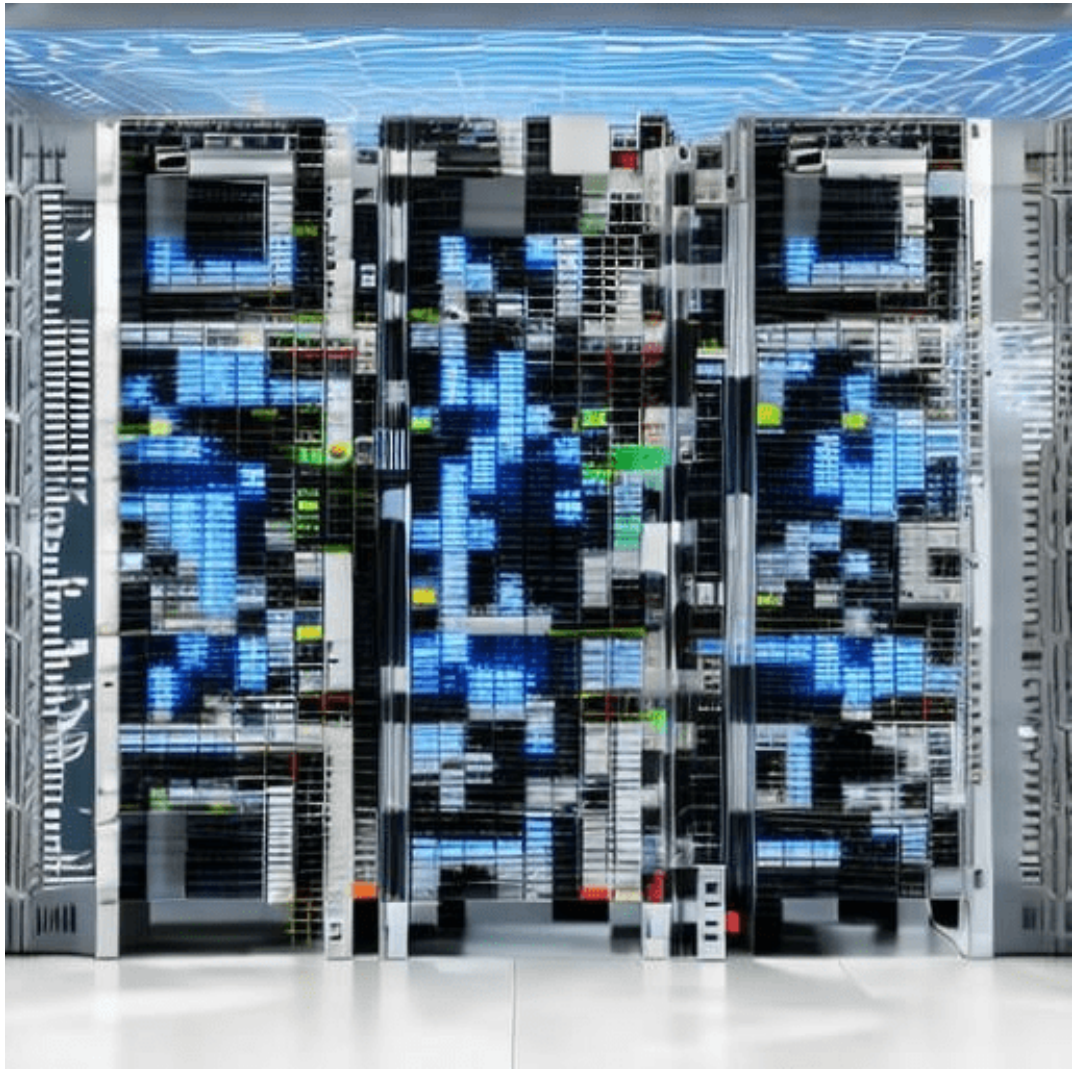
# High Performance Computing - Basic Course

June 13, 2024



WAGENINGEN
UNIVERSITY & RESEARCH

# Instructors

- **Jeremie Vandenplas**
- **Leonardo Honfi Camilo**
- **Dapeng Sun**

WAGENINGEN
UNIVERSITY & RESEARCH

# After this course, you should

- Understand basic concepts of High Performance Computing.
- Be aware of the Anunna supercomputer and its features.
- Understand user environments and modules
- Be able to create, deploy and monitor slurm jobs.
- Be able to interact with other anunna users and seek out support

# Ice Breaker

1. Go to **wooclap.com**
2. Enter code **HPCBASICS**

# Outline

12:30 - Ice Breaker
12:35 - Connecting to the HPC
13:00 - High Performance Computing
13:10 - Anunna
13:25 - Command Line Refresher
14:00 - Transferring Files
14:15 - Break
14:30 - Environments
14:40 - LMOD - Environment Modules
15:10 - SLURM Jobs
16:00 - Monitoring jobs
16:10 - Job Arrays
17:00 - Closing remarks

WAGENINGEN
UNIVERSITY & RESEARCH

# Connecting to the WUR HPC

Guiding users on connecting to the HPC from Linux, Mac, and Windows.

# Before You Connect

- The connection to the HPC is enabled by the Secure Shell ( SSH ) protocol
- On Linux and macOS, SSH is either packages or preinstalled.
- On Windows, we recommend the use of **MobaXTerm**
- If not, then you can use
  - Putty
  - Windows Subsystem for Linux ( WSL )
  - PowerShell

WARNING: If you mistype the password 3 times, you account will be blocked for 24h.

# Connecting to the Anunna HPC

Open a terminal and run the command:

```
$ ssh username@login.anunna.wur.nl
```

WARNING: no characters are displayed when typing your WUR password!!!

**MobaXTerm:** go to Session => SSH and under remote host type fill in

```
login.anunna.wur.nl
```

# Connecting to the WUR HPC

- The connection to the HPC is enabled by the Secure Shell ( SSH ) protocol
- On Linux and macOS, SSH is either packages or preinstalled.
- On Windows, we recommend using **MobaXterm**.
- If not, then you can use
  - Putty
  - Windows Subsystem for Linux ( WSL )
  - PowerShell

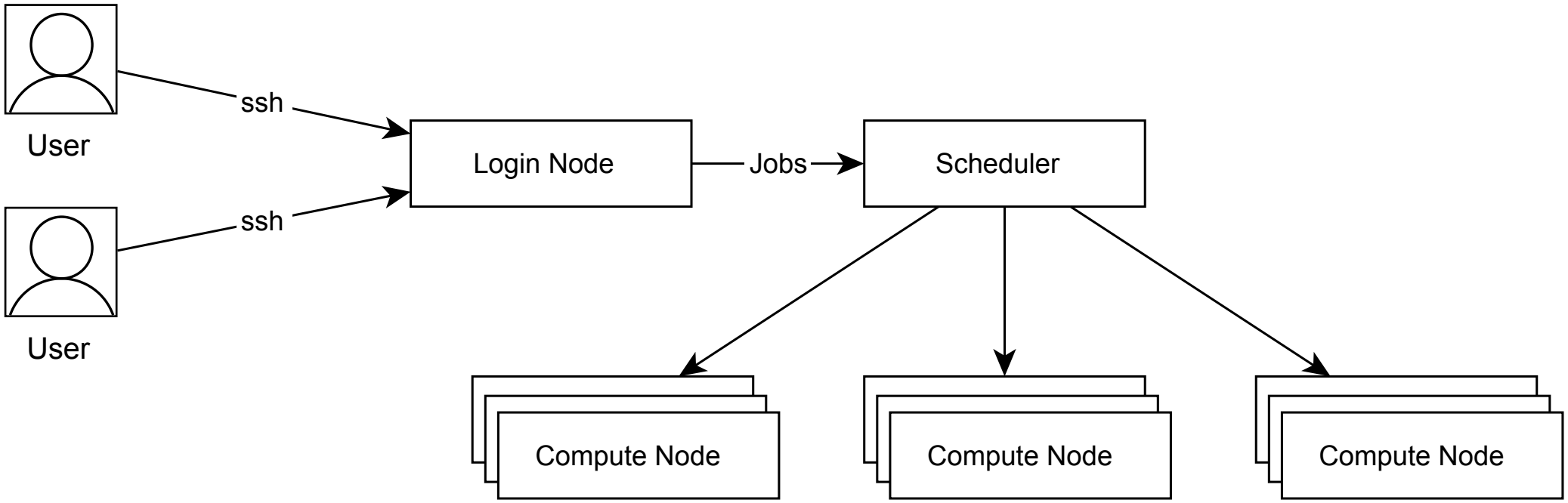
# High Performance Computing

A Brief introduction into High performance computing concepts

# Definition

High Performance Computing (HPC) is the practice of aggregating computing power in a way that delivers much higher performance than one could get out of a typical desktop computer or workstation, in order to solve large problems in science, engineering, or business.

# Concept Diagram



# Scheduler

Resources in an HPC are reserved by a scheduler, or resource manager.

**Resources:**

- Cores (tasks)
- memory/memory per core
- Threads
- gpu

**Examples:**

- Slurm
- PBS Professional
- Torque
- LSF
- HTCondor

WAGENINGEN
UNIVERSITY & RESEARCH

# Jargon - 1

**Job:**

All the processes, or tasks, involved in achieving a computational goal

**Tasks:**

A process, or a unit of work, that comprises a job.

**Threads:**

Compute operations that make up a process.

# Jargon - 2

**CPU:**
Core or hardware thread

**Muti-threaded:**
Tasks are broken down onto several CPUs

**Multi-process:**
Job composed of several tasks

# Anunna


An overview of the Wageningen University's supercomputer and its services

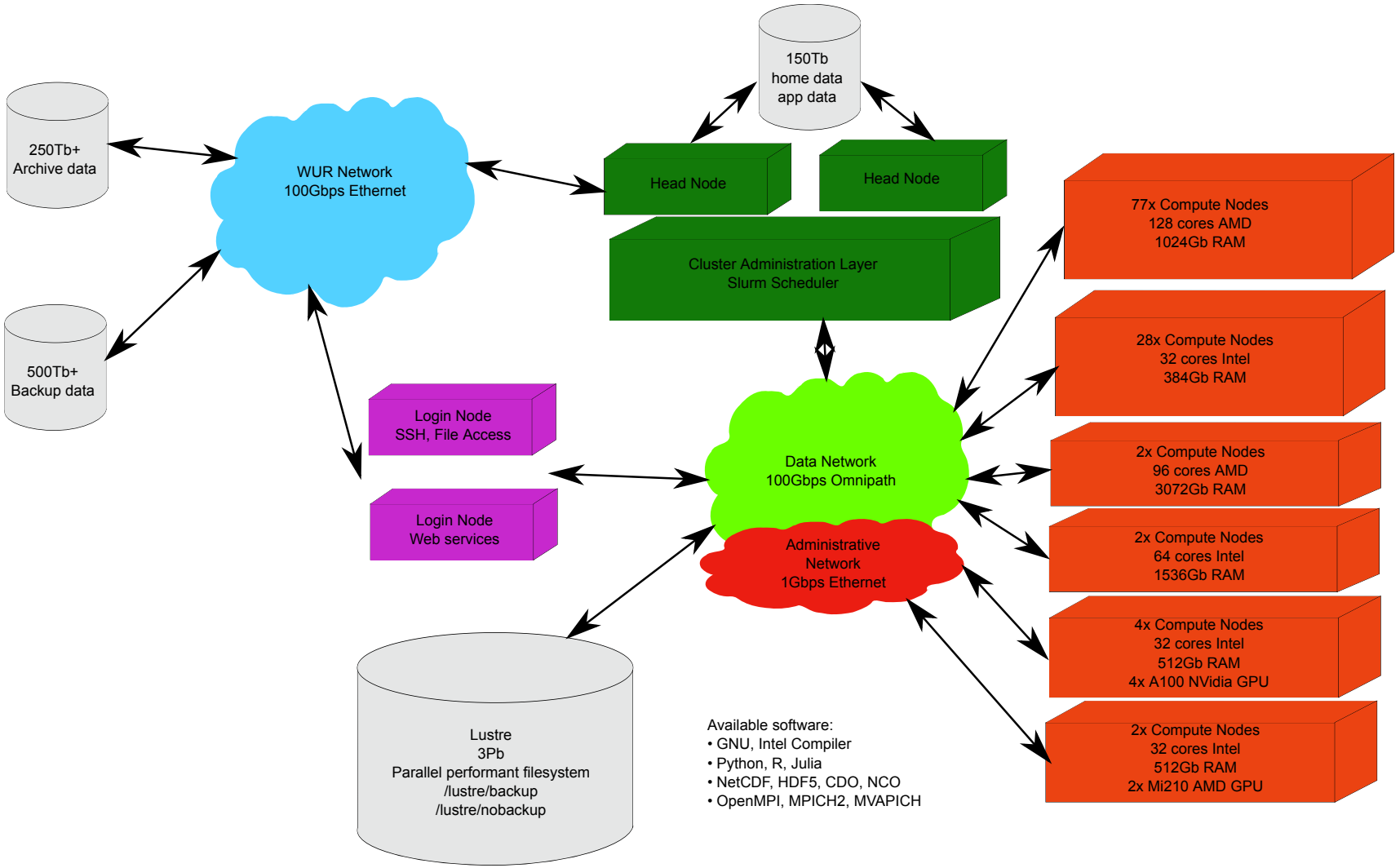
WAGENINGEN
UNIVERSITY & RESEARCH

# Overview

**Cores:** 11520

**Nodes:** 119

**GPUs:** 22




WAGENINGEN
UNIVERSITY & RESEARCH

# Software

Operating System

**Ubuntu 20.04**

Scheduler:

**SLURM**

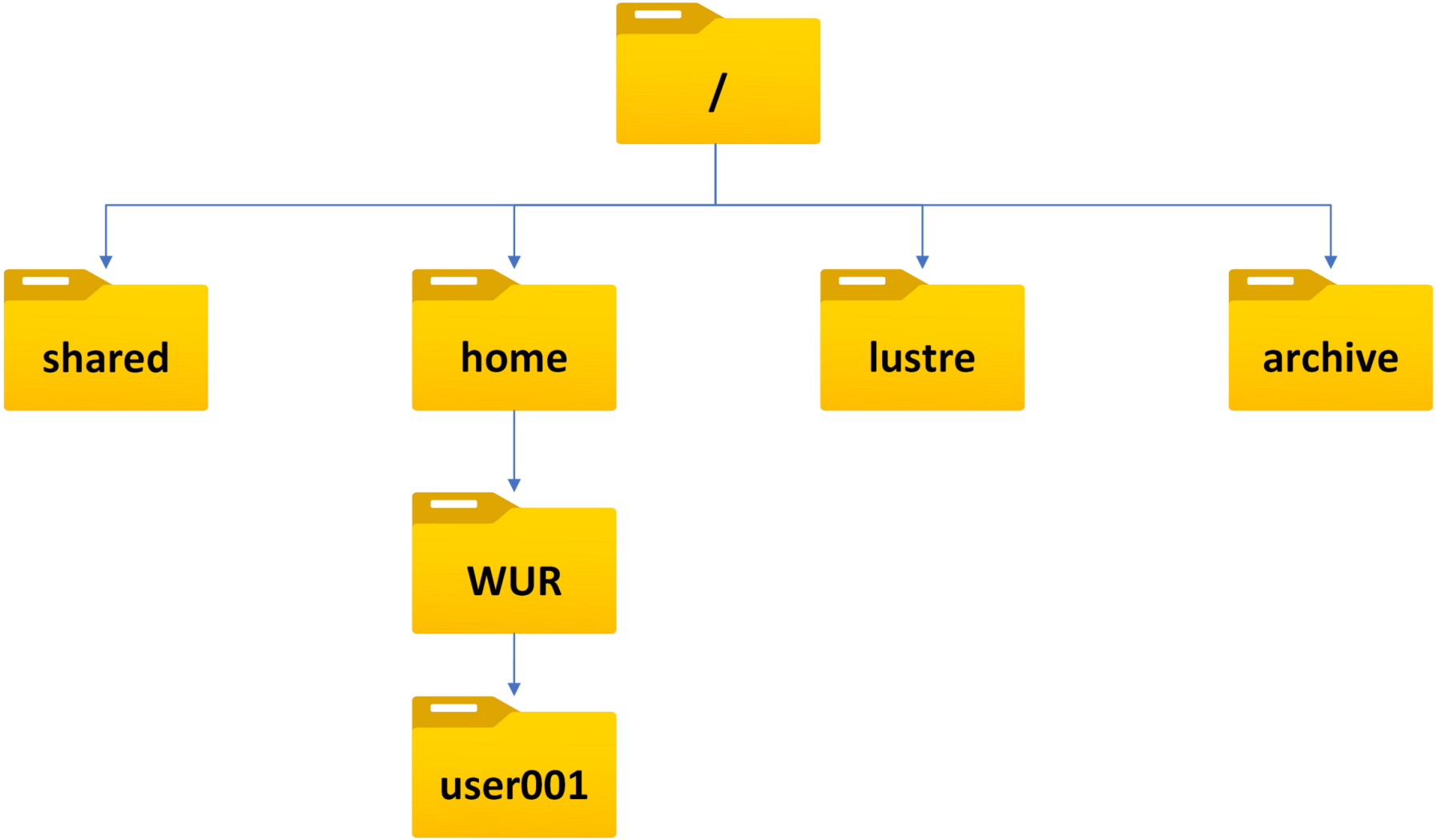
Compilters

**GCC, Intel, …**

Software

**R, Octave, Matlab, julia, python, OpenFOAM, AlphaFold, …**

# Main Folders




WAGENINGEN
UNIVERSITY & RESEARCH

# Storage - Home and Archive

## Home:

- /home/[institution]/[username]
- limited to 200 GB (210GB hard limit)
- slow filesystem (not suitable for jobs)

## Archive:

- /archive/[institution]/[username]
- Cheap
- Only for storage (no jobs)

# Storage - Lustre

## backup:

- /lustre/backup/[institution]/unit/[username]
- Extra costs for backup

## nobackup:

- /lustre/nobackup/[institution]/unit/[username]

## scratch:

- /lustre/scratch/[institution]/unit/[username]
- Could be cleaned up

# WebServices - Jupyter

**notebook.anunna.wur.nl**

- Sandbox mode for small scripts running at the login0 node
- Larger jobs can run in the cluster
- Supports various versions of Julia, Python and R (**Ju****Py**te**R**)
- Reservations can be made for courses with support from the HPC team
  - Reservations for courses can be made via a form in the itsupport page
  - New request →Hosting → High Performance Computing

# WebServices - Galaxy

galaxy.anunna.wur.nl

Galaxy offers a vast collection of bioinformatics tools and workflows, enabling researchers to perform a wide range of analyses from sequence alignment to data visualization, all within a single platform.

WAGENINGEN
UNIVERSITY & RESEARCH

# Best Practices and Policies

- We **never** remove user data
- No jobs should run on login node
- Maximum runtime of jobs is 3 weeks.
- No jobs should output/write to the home directory
- Do not add modules to your **~/.bashrc** or **~/.bash_profile** files
- Anaconda installations at home directory are not supported

# Future Plans

- New and more robust Logins
- Storage Tiering
- Ubuntu 24.04
- GUI Applications on the browser with **Open OnDemand**


WAGENINGEN
UNIVERSITY & RESEARCH

# Command Line Refresher

A refresher on the bash shell and some basic commands and tools

# Bash shell - $

**Bourne Again Shell**

- Command-line interface that allows users to interact with the operating system by typing commands to perform operations and manage files and programs.
- Most popular, though there are many alternatives
- Interpreter located at **/bin/bash**
- It has its own language syntax
- Commands usually follow the format:

```
$ <application> --<flag>/-<flag> <argument>
```

# cd - change directory

**Template:**

```
$ cd <directoryPath>
```

**Go to home directory:**

```
$ cd ~
```

**Go to previous directory:**

```
$ cd -
```

**Go one directory up:**

```
$ cd ..
```

**Go two directories up:**

```
$ cd ../..
```

**relative path:**

```
$ cd ./apps
```

**full path:**

```
$ cd /home/WUR/user001/apps
```

WAGENINGEN
UNIVERSITY & RESEARCH

# mkdir - make directory

**Template:**

```
$ mkdir -<flags> <directoryPath>
```

**Create folder at home directory:**

```
$ mkdir ~/newFolder
```

**Create folder with parents:**

```
$ mkdir -p ./first/second/newFolder
```

# ls - List

Template:

```
$ ls -<flags> <fileOrDirectory>
```

list all files at the home directory:

```
$ ls -a ~
```

list all files in long format at the current directory

```
$ ls -al .
```

# cp - Copy

Template:

```
$ cp -<flags> <source> <target>
```

copying files (interactive):"

```
$ cp -i file01 file02
```

copying directories (interactive):

```
$ cp -ri directory01 directory02
```

# mv - Move

**Template:**

```
$ mv -<flags> <source> <target>
```

**Rename files (interactive)**

```
$ mv -i ./file01 ./file02
```

**Rename directory (interactive):**

```
$ mv -i directory01 directory02
```

# rm - Remove

**Template:**

```
$ rm -<flags> <fileOrDirectory>
```

**Remove multiple files (interactive):**

```
$ rm -i ./file01 ./file02 ./archive/file03.txt
```

**Remove multiple directories (interactive):**

```
$ rm -ri directory01 directory02
```

# echo - Display Text

**Template:**

```
$ echo -<flags> <string>
```

**Display contents of $PATH:**

```
$ echo $PATH
```

**Display string with escape characters:**

```
$ echo -e "\nThis was a triumph!\n"
```

# man - Manual Pages

**Template:**

```
$ man -<flags> <application>
```

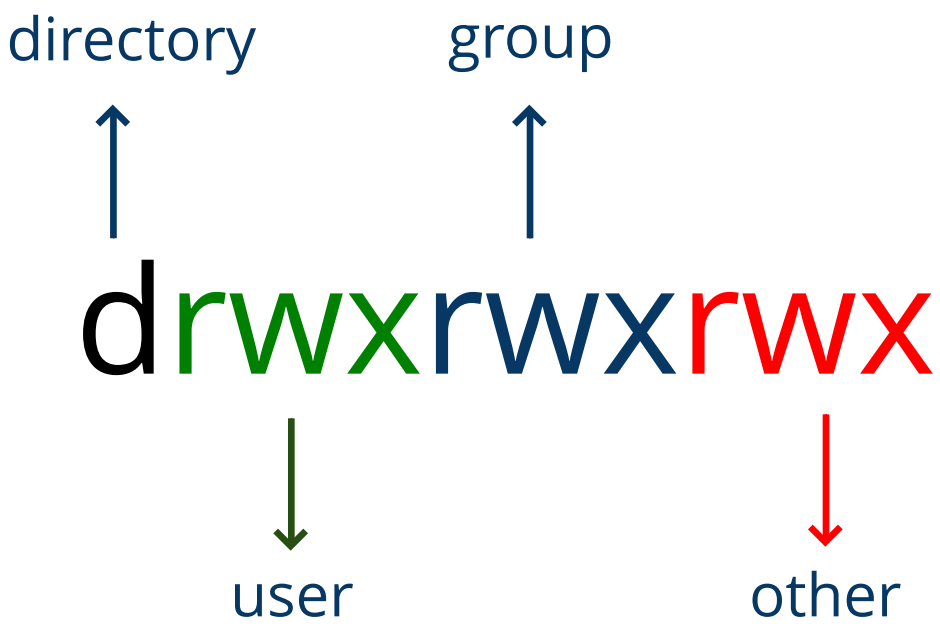
**Manual pages of ls:**

```
$ man ls
```

**shortcuts**

```
navigation: arrow keys, page down, page up
page down: space bar
search: / (n: previous, N: next)
quit: q
```

# Permissions



# Permission Octals

$$r : 2^2 \quad w : 2^1 \quad x : 2^0$$

$$rwx : 1 * 2^2 + 1 * 2^1 + 1 * 2^0 = 4 + 2 + 1 = 7$$

$$r\text{-}x : 1 * 2^2 + 0 * 2^1 + 1 * 2^0 = 4 + 0 + 1 = 5$$

$$r\text{--} : 1 * 2^2 + 0 * 2^1 + 0 * 2^0 = 4 + 0 + 0 = 4$$

# Putting It All Together

rwxr-xr-x = 755

# Exercise: Permission Octals - (5 min)

rw-r-xr-x

rw-r-----

rw-rw-rw-

rwxrwxrwx

# Exercise: Permission Octals - solution

rw-r-xr-x = 655

rw-r----- = 640

rw-rw-rw- = 666

rwxrwxrwx = 777

# chmod - Changing Permissions

**Template:**

```
$ chmod  octal <file/folder> <optionFlags>
```

**Changing permissions of a file**

```
$ chmod 775 myfile
```

**Changing permissions of a directory and all enclosed files and subdirectories**

```
$ chmod 775 myDirectory -R
```

**Making a file executable**

```
$ chmod +x myfile
```

# chown - Changing Ownership

Template:

```
$ chown  user:group <file/folder> <optionFlags>
```

Giving user001 ownership of myFile

```
$ chown  user001: myFile
```

Giving mygroup ownership of myDirectory without changing the user ownership

```
$ chown :mygroup myDirectory -R
```

Assigning onwership all the files and folders inside myDirectory to user001

```
$ chown user001:user001 myDirectory -R
```

# Text Editors

- Nano (recommended)
- Vim
- emacs
- VScode/pyCharm with Remote SSH Extension

WARNING: Only use VScode/pyCharm for editing files.

Do not use them to run scripts/jobs

# Scripts

```
1 #!/usr/bin/bash
2
3 echo -e "\nHello, World!\n"
```

- Collection of commands
- Executed in sequence (top to bottom)
- First line of the script defines interpreter (#!)
- Must be executable (permissions)

# Scripts - Alternative Interpretors

For python scripts:

```
1  #!/usr/bin/env python
```

For R scripts:

```
1  #!/usr/bin/env Rscript
```

# Exercise - Create and run script

1. Open your text editor of choice
2. Add the interpreter
3. Write a command to display the phrase "Hello, World"
4. Write a command that lists the contents of your home directory (including hidden files).
5. Redirect the output of this last command into a file labelled "home.txt" inside your home directory
6. Make sure your script is executable
7. Copy/Move your script to ~/
8. execute it from your home directory

# Exercise solution -script

```
1 #!/usr/bin/bash
2
3 echo -e "\nHello, World!\n"
4
5 ls -al $HOME > ~/home.txt
```

# Transferring files

How to transfer file to and from the HPC

# Linux/Mac/Powershell - scp

**Template:**

```
$ scp -<flags>  user@addressSource user@addressTarget
```

**Copying files to the HPC:**

```
$ scp myfile user@login.anunna.wur.nl:destination
```

**Copying directories from the HPC :**

```
$ scp -r user@login.anunna.wur.nl:directory destination
```

# STFP

Windows: MobaXterm

Mac: Cyberduck

linux: file browser

# Exercise - Transferring files

1. Copy file from the location below to your laptop and open it on your laptop.
    - file location: **/lustre/shared/hpcCourses/picture.jpg**

# Downloading Files into the HPC

**curl**

```
curl -JOL <linkAddress>
```

**wget**

```
wget <linkAddress>
```

# Exercise - Download files

1. Go to **github.com/Code-Hex/Neo-cowsay**
2. Go to releases (right side of the screen)
3. Under **assets** look for **cowsay_2.0.4_Linux_x86_64.tar.gz**
4. right click into the link and copy the link address
5. user wget (or curl) to download the file into your home directory
6. create the folder ~/**apps/cowsay**
7. Copy downloaded file to the **~/apps/cowsay** folder
8. Extract the file with the command **tar xvfz file.tar.gz**

# Environments

Bash environment variables, how to manage them and some examples

# Definitions

Bash environment variables are key-value pairs stored within the Bash shell that influence the behaviour of software on the system.

They can be used to configure shell settings, store data like paths to executables or directories, and control the operation of scripts and applications.

# Env and Notable Variables

Display the variables in your session:

```
$ env
```

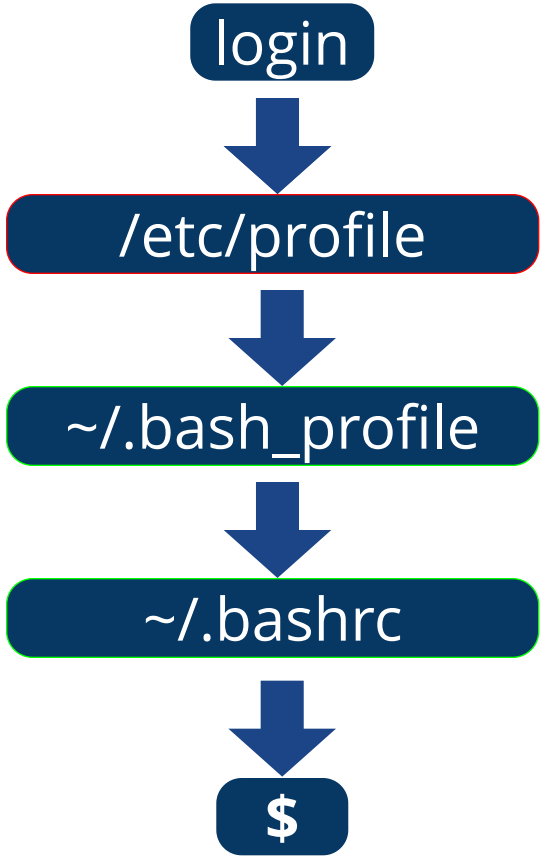
Notable:

**HOME** - stores the location of your home directory

**PATH** - stores locations of your executable files (separated by :)

**LD_LIBRARY_PATH** - stores locations of libraries

# WHERE DO THESE VARIABLES COME FROM?




WAGENINGEN
UNIVERSITY & RESEARCH

# Exercise

1.
2. Add the **cowsay** folder (from the previous exercise) to your **PATH** variable
3. Change your working directory to your home
4. Run the command

```
echo "Hello" | cowsay
```

# Solution

# Lmod - Environment Modules

An introduction to environment modules and some usage examples

# What does it do?

## Lmod

A flexible, Lua-based **environment module** system that enables the dynamic modification of a user's environment via module files, simplifying the management of software and library versions on HPC systems.

## Environment module

A system that allows for the dynamic configuration of a user's shell environment variables, facilitating easy management of application environments and versions on UNIX-like systems.

# MODULEPATH

- A bash environment variable that tell **lmod** where the modules are located.
- Multiple locations can be specified, divided by a colon ( **:** )
- Can be modified with an **export** command
- Can be appended with the **module use** command

Find the default MODULEPATH location:

```
$ echo $MODULEPATH
```

WAGENINGEN
UNIVERSITY & RESEARCH

# Listing Available Modules

**Get overview of available modules**

```
$ module overview # ml ov
```

**List available software:**

```
$ module avail # ml av
$ module spider # ml spider
```

**Search with a clear cache (slower):**

```
$ module --ignore_cache av
$ module --ignore_cache spider
```

WAGENINGEN
UNIVERSITY & RESEARCH

# Searching for Modules

## Searching for modules

```
$ module avail R/ # ml av R/
$ module spider R/4 # ml spider R/
```

## Search for a keyword inside a module's description

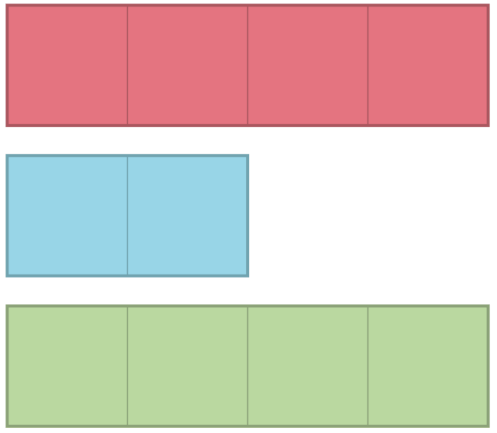
```
$ module key numpy # ml key numpy
```

# Exercise - 5 min

- Find the module that has the **terra** R extension

# Modules at Anunna

- Modules are to be arranged in "buckets" wrt to year
  - One version of software per bucket
  - Conveyor belt: software older than 3 years will be removed or moved to the **legacy** bucket.
  - A new bucket will be released each year.
- The building of the packages is being done with **EasyBuild**
  - Software tested across many HPCs worldwide
- If you need to a specific software
  - Submit a ticket
  - If you can provide an EasyBuild recipe, it will speed things up



EASYBUILD.io
building software with ease

WAGENINGEN
UNIVERSITY & RESEARCH

# Getting Additional Information

## Get the description of a module

```
$ module whatis Python-bundle-PyPI
```

## Get the entire module file

```
$ module show Python-bundle-PyPI # ml show Python-bundle-PyPI
```

# Module Files

- Written in Lua
- Describe the module
- Loads other modules
- Modify environment variables

# Loading Modules

load module (e.g. python)

```
$ module load 2023
$ module load Python/3.11.3-GCCcore-12.3.0
```

load module (e.g. python)

```
$ ml 2023
$ ml Python-bundle-PyPI/2023.06-GCCcore-12.3.0
```

# Removing Modules

Remove individual module (e.g. python)

```
$ module unload Python/3.11.3
```

Remove all modules

```
$ module purge
```

Swap a module with another

```
$ ml module swap Python-bundle-PyPI SciPy-bundle
```

# Modify MODULEPATH

**Append MODULEPATH:**

```
$ module use myFolder
```

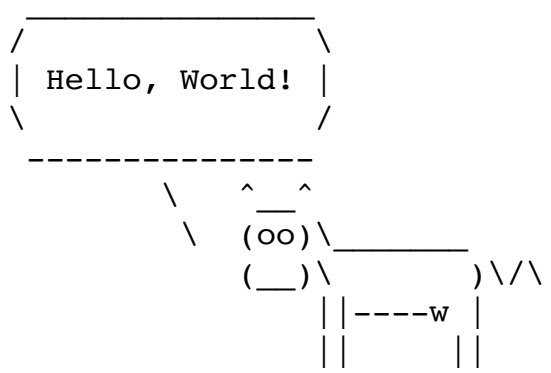
**Redefine MODULEPATH:**

```
$ export MODULEPATH=/my/Directory/Location
```

# Exercise - Create Your Own Modules

1. Create a folder called **modules** in your home directory
2. Create a subfolder inside of **modules** named **cowsay** (**~/modules/cowsay**)
3. Copy the file **2.0.4.lua** from **/lustre/shared/hpcCourses** to the cowsay module subfolder
4. Add the modules folder to your MODULESPATH (Hint: "module use")
5. Use the spider command to search for the cowsay module
6. At your home directory, load the **cowsay** module
    1. Check your PATH variable, before and after loading the module (you can always unload)
7. Execute the command: **cowsay "Hello"**

# Exercise - Solution




- Modules modify your environment variables  like PATH and LD_LIBRARY_PATH automatically
- These changes can be reverted by unloading
- You can add your own modules by creating a lua file and pointing your MODULEPATH to it
  - Easiest way it using the **module use** command

# Jobs

How to submit and monitor a job at the WUR HPC

# SLURM

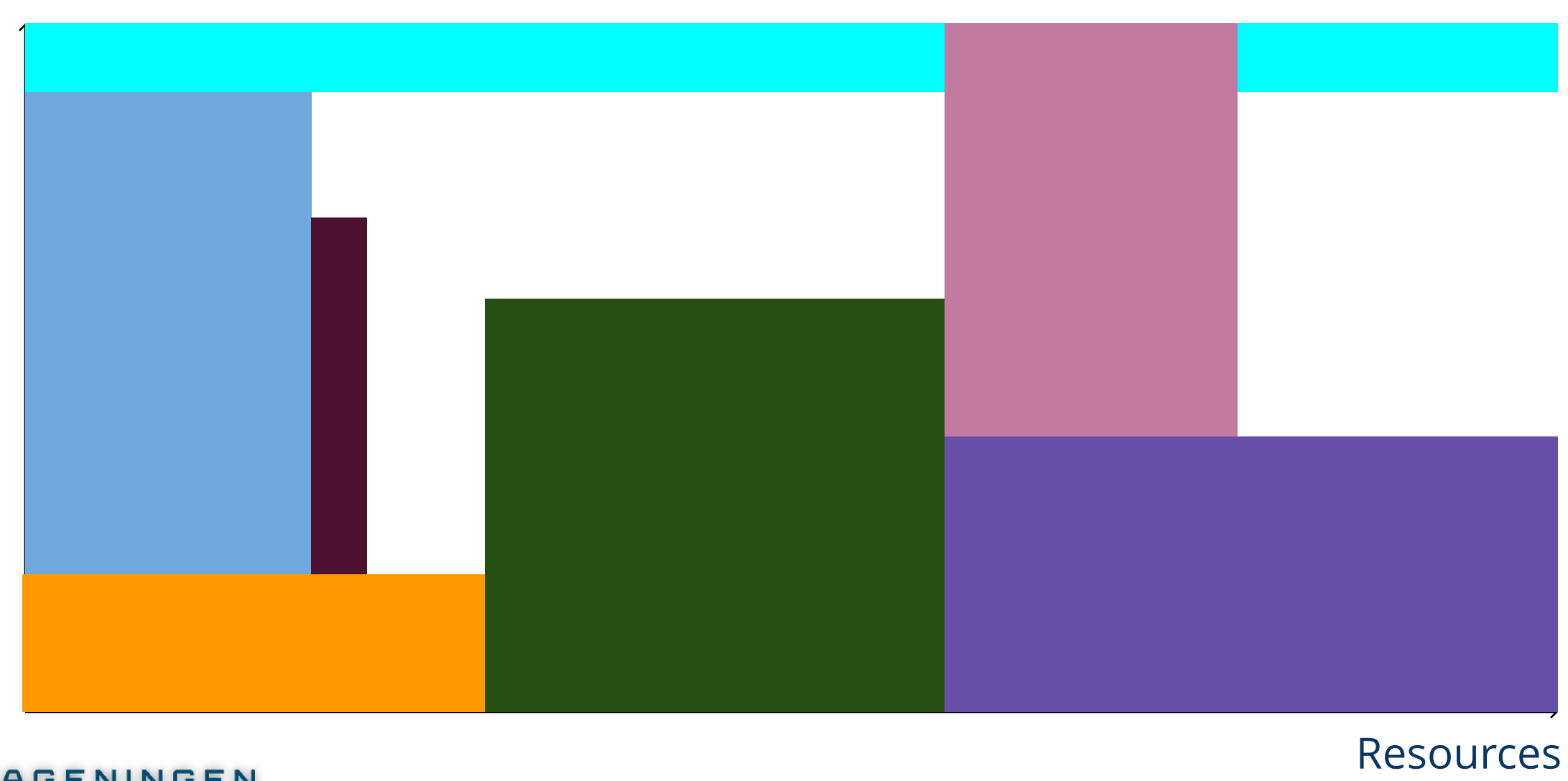
## Simple Linux Utility for Resource Management

- Manages and **allocates resources** (compute nodes)
- Manages and **schedules jobs** on a set of allocated nodes
- **Sets up environments** for parallel and distributed computing

# Requirements for a job

- Sequential or parallel
- If parallel: **multi-threaded** or **multi-process**?
- Resource requests:
  - Number of CPUS
  - Amount of RAM
  - Expected computing time
  - …
- Job steps
  - Job steps can be created with the command **srun** (advanced)

# Tetris




WAGENINGEN
UNIVERSITY & RESEARCH

# Requirements - How to determine

- Read documentation
- Read the source code
- Test it! Run it in a safe environment
  - use sinteractive

# sinteractive

- Wrapper on **srun**
- Provides an immediate interactive shell on the node

```
$ sinteractive -c <cpus> --mem <MB>
```

Example:

```
$ sinteractive -c 1 --mem 2000M
srun: job 19271078 queued and waiting for resources
srun: job 19271078 has been allocated resources
```

# Exercise - Step 1

## Context

- Python script that loads weather data from a Dutch weather station and draws some figures

## Tasks

1. **Create a directory** called example1
2. **Copy** the **Python script** /lustre/shared/training_slurm/spring_2024/serial/training/weer_vanaf_2000.py **in the directory** example1
3. **Load** the **environment module** 2023
4. **Load** the **environment module** Python
5. If needed, **install locally** the libraries datetime, matplotlib, and pandas

**Extra:** write a **bash script** to do the steps 1-4!

# Exercise - solution

```
#!/bin/bash
#Create a directory
mkdir example1
cd example1
#Copy the Python script
cp /lustre/shared/training_slurm/spring_2024/serial/training/weer_vanaf_2000.py .
#List the content of the directory
ml load 2023
ml load Python
```

# Exercise - Step 2

Find the job requirements

**Tasks**

1. Find the job requirements (e.g., memory) of the Python script by using **sinteractive**

```
$ sinteractive --mem 100M --time=0-0:10 -c 1
$ module load 2023
$ module load Python
$ python weer_vanaf_2000.py 240
```

# Example SLURM Script

```
#!/bin/bash
#---------Name of the job--------------
#SBATCH --job-name=example1
#---------Mail address -----------------
#SBATCH --mail-user=my.email@wur.nl
#SBATCH --mail-type=ALL
#---------Output files-----------------
#SBATCH --output=result_%j.txt
#SBATCH --error=error_result_%j.txt
#---------Other information------------
#SBATCH --comment='Some comments'
#---------Required resources-----------
#SBATCH --time=0-1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --mem=4000
#---------Environment, Operations and Job step -------
echo 'This is my job'
```

WAGENINGEN
UNIVERSITY & RESEARCH

# SLURM options

Job name:

```
--job-name="job1"
```

To set the name of output files:

```
--output=result_%j.txt
--error=error_result_%j.txt
```

Set comment/label (for billing):

```
--comment="abcd"
```

# SLURM options - required resources

To require **1 task**:

```
--ntask=1
```

To require **1 task that can use up to 2 cpus:**

```
--ntasks=1
--cpus-per-task=2
```

To require **memory** (in MB):

```
--mem=4000
OR
--mem-per-cpu=4000
```

# SLURM options - Time

To require **15 minutes**:

```
--time=15
```

To require **1 day**:

```
--time=1-00:00:00
```

To require **1 day, 5 hours and 30 minutes:**

```
--time=1-05:30:00
```

**Maximum time is set to 3 weeks**

# Partitions

Main partition:

```
--partition=main
```

Gpu partition:

```
--partition=gpu
```

# Specific features - Constraint

To choose a **specific feature:**

```
--constraint=gen3
```

# Submit a script to SLURM

Template:

```
$ sbatch </Path/To/Script.sh>
```

Example

```
$sbatch script_slurm.sh
Submitted batch job 19271078
```

SLURM assigns an ID to the job ($JOBID)

# View the status of your jobs

squeue

- View information about jobs located in the SLURM scheduling queue

To report a list of users:

```
$ squeue -u <user_id>
```

To report a list of specific jobs:

```
$ squeue -j <job_id_list>
```

To report the expected start time of pending jobs:

```
$ squeue --start
```

# Cancel a job

Cancel a single job:

```
$ scancel <job_id>
```

Cancel multiple jobs:

```
$ scancel <job_id0> <job_id1> <job_id2> ...
```

WAGENINGEN
UNIVERSITY & RESEARCH

# Exercise - Step 3

Write and submit a slurm script

- **Write** a SLURM script for the job with the resource requirements determined at Step 2

    1. **Submit** the job to SLURM
    2. **Check the status** of your job and cancel it if needed.
    3. **Check the figures** generated by the Python script.

**TIP -** Template of a SLURM script:

/lustre/shared/training_slurm/spring_2024/serial/training/script_slurm.sh

WAGENINGEN
UNIVERSITY & RESEARCH

# Monitoring Jobs

Introduction to monitoring slurm jobs, and the anunna module

# Monitoring Active and Finished Jobs

**Commonly used SLURM commands to monitor and control jobs**

- squeue
- scancel
- sacct
- scontrol
- sprio

# sacct

- Displays accounting data for all jobs/steps
- Some information are available only at the end of the job

To get an overview of all jobs run in 2024

```
$ sacct -X -u <name> --starttime 2024-01-01 --endtime now
```

To get an overview of a job

```
$ sacct -j <job_id> --format=JobID%-20,AveRSS, MaxRSS,Elapsed
```

# scontrol - Slurm configuration and state

- Update job resource requests
- Work only for running jobs
- **Provides a lot of information…**

To get information of a job

```
$ scontrol show job <job_id>
```

To get information on nodes

```
$ scontrol show nodes
```

# sprio - Slurm configuration and state

- View the components of a job's scheduling priority
- Rule: a job with a lower priority can start before a job with a higher priority if it does not delay the job's start time

To report a list of specific job:

```
$ sprio -j <job_id>
```

To report detailed information:

```
$ sprio -l
```

# Job Arrays

How to submit array of jobs in slurm

# Job Array – embarrassingly parrallel

Parallelism is obtained by launching the **same program** multiple times simultaneously

- Every program does the same thing
- No inter-process communication
- Useful cases

- Multiple input/data files
- Random sampling

# Job Arrays - resource

To submit a job array with index values between 1 and 3

--array=1-3

To submit a job array with index values of 1, 5, 10, 11, 12

--array=1,5,10-12

To submit a job array with index values between 2 and 60, with a limited number (4) of simultaneous running jobs

--array=2-60%4

# Job Arrays - environment variable

Job arrays have additional environment variables

The job array index value

$SLURM_ARRAY_TASK_ID

# Exercise - Step 4

## Aim

- **Submit** an array of jobs to run the same program simultaneously using different inputs

## Tasks

1. **Write** a SLURM script for a **job array** that loads and generate figures for **multiple Dutch weather stations**, with IDs 240, 260 and 270
   - **SLURM option**: --array=240,260,270
   - **SLURM variable**: $SLURM_ARRAY_TASK_ID

# Closing Remarks

Support, wiki, community and whatever else

# HPC Advanced Overview

**12:30** - Introduction and IceBreaker

**12:45** - SSH Keys

**13:00** - HPC Basics Review

**13:30** - Job Arrays + exercise

**14:00** - Types of Parallelism + exercise

**14:45** - Break

**15:00** - Types of Storage - Exercise

**15:30** - Apptainer Containers

**16:30** - BYOP

**17:00** - End

# Communication and Support

## Message of the day

- Announcements often posted

## Annuna Users at Teams

- Feel free to post there if you have any questions or problems

## Tickets

- For more involved support and questions, create a ticket.
  - https://itsupport.wur.nl/
  - New Requests
  - Create a ticket
  - Type HPC at the field "*What is your question about?*"

# Wiki - wiki.anunna.wur.nl

- Slides from courses
- Instructions on how to run jobs
- Tips about python and R
- ...

# Feedback Form

WAGENINGEN
UNIVERSITY & RESEARCH

# So long, and thanks for all the fish!

WAGENINGEN
UNIVERSITY & RESEARCH