Course Intro
Intro to HPC

UNIVERSITY OF
COPENHAGEN

introduction

Sandbox data scientist

Background in bioinformatics and genomics

Alba Refoyo Martinez, PhD

Senior Consultant

Background in neuroscience and bioinformatics

Stefano Pupe, PhD

UCPH HeaDS

Section for Health Data Science & AI

novo nordisk foundation

Benefitting people and society

National Health Data Science Sandbox

introduction

Jennifer Bartell, PhD

Former member of HeaDS, now Senior Scientific Manager at NNF

Co-creator of the content and many of the slides for this course

UCPH HeaDS

Section for Health Data Science & AI

novo nordisk foundation
Benefitting people and society

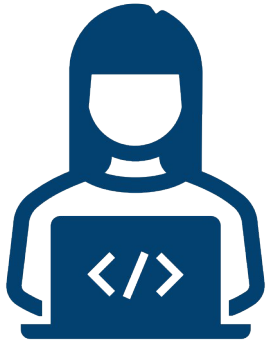
National Health Data Science Sandbox

SERVICES

WHO WE SUPPORT

MED/VET RESEARCHERS

WET LAB RESEARCHERS



DRY LAB RESEARCHERS

COURSE LEADS

GUIDANCE / ASSISTANCE WITH DS ANALYSIS & RDM

PRACTICAL COMPUTING ADVICE / UPSKILLING

SERVICES

HDS SERVICE CORE

CONSULTATIONS

Need guidance? Drop by for a consultation on your DS analysis

COMMISSIONED RESEARCH

Commissioned DS Analysis
Commissioned Supervision

OUTREACH

Conference, seminars and networking events - Join us!

COURSES & WORKSHOPS

Data Science skills, Tools and HDS Topics

DATA LAB & HDS SANDBOX

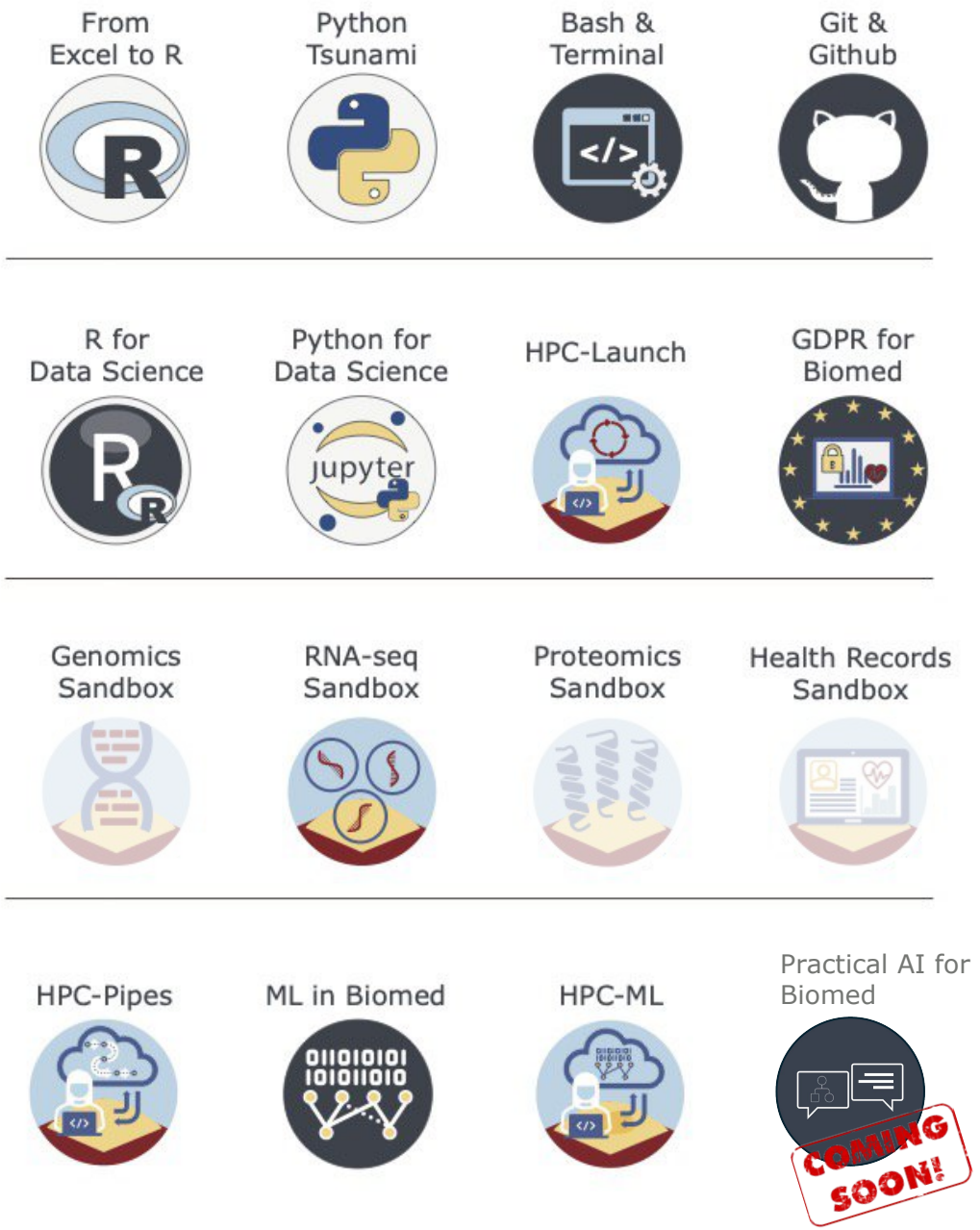
COURSE TRACK

Skill Sets at Different Levels:

- Programming (R, Python)
- Omics Data Analysis (DNA, RNA, Protein)
- High Performance Compute Tools
- Machine Learning for HDS

Data Lab (intro & intermediate)

HDS Sandbox (intermediate & advanced)



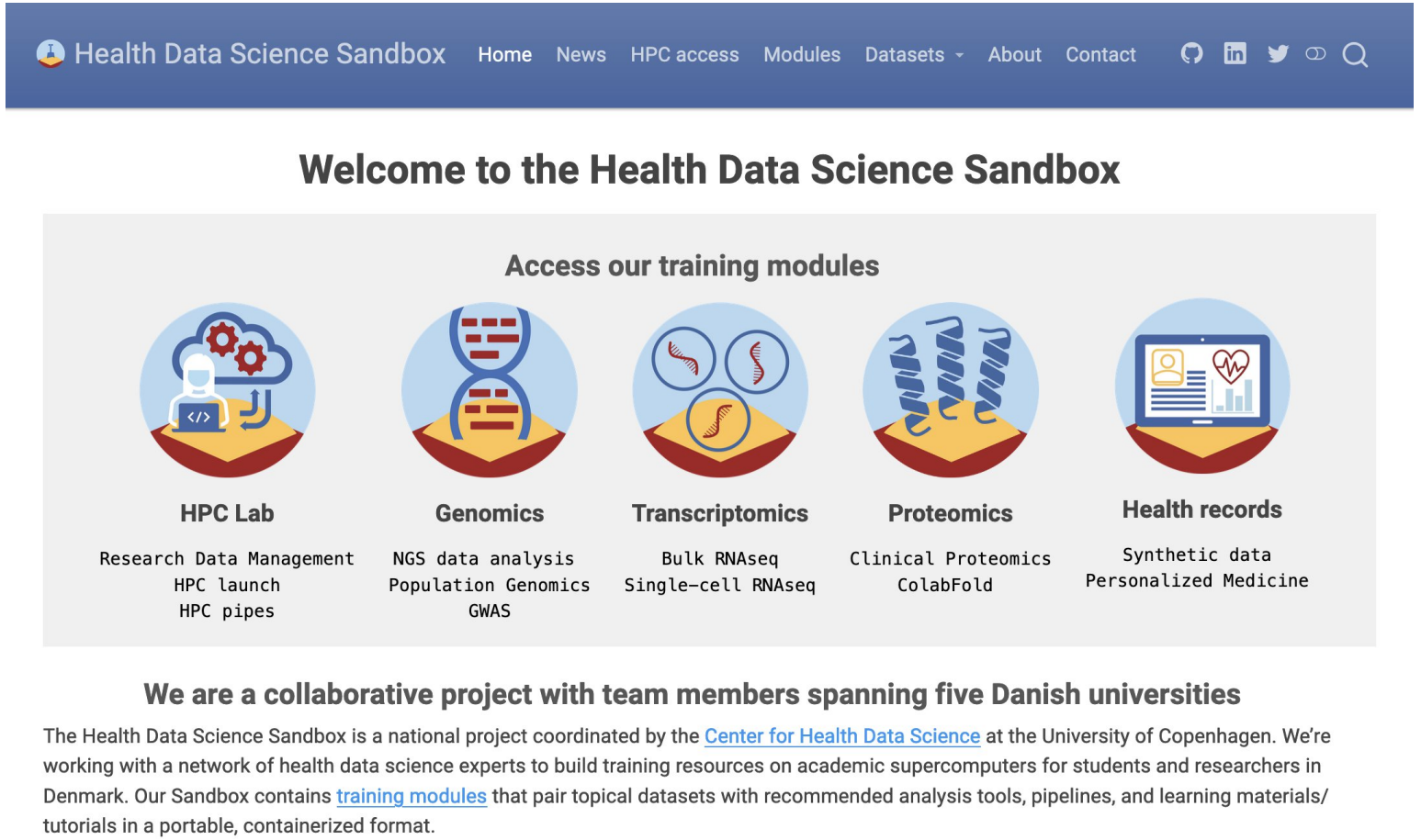
SERVICES

HEALTH DATA SCIENCE SANDBOX

Sandbox apps allow independent use of data, tools, & guides

- **Website** with tutorials, guides, onboarding
- **Workshops** run locally by Sandbox staff
- **External courses** supported by Sandbox resources
- **GitHub** with accessible materials & environment setup
- Containers on **DockerHub**

https://hds-sandbox.github.io



course objectives

- **Introduction to HPC systems**
  Understand HPC systems
  Assess high performance computing goals
  Secure compute and storage for your project

course objectives

- **Introduction to HPC systems**
  Understand HPC systems
  Assess high performance computing goals
  Secure compute and storage for your project

- **Set up computing environment essentials**
  Access and authentication
  Job scheduling basics
  Data handling

course objectives

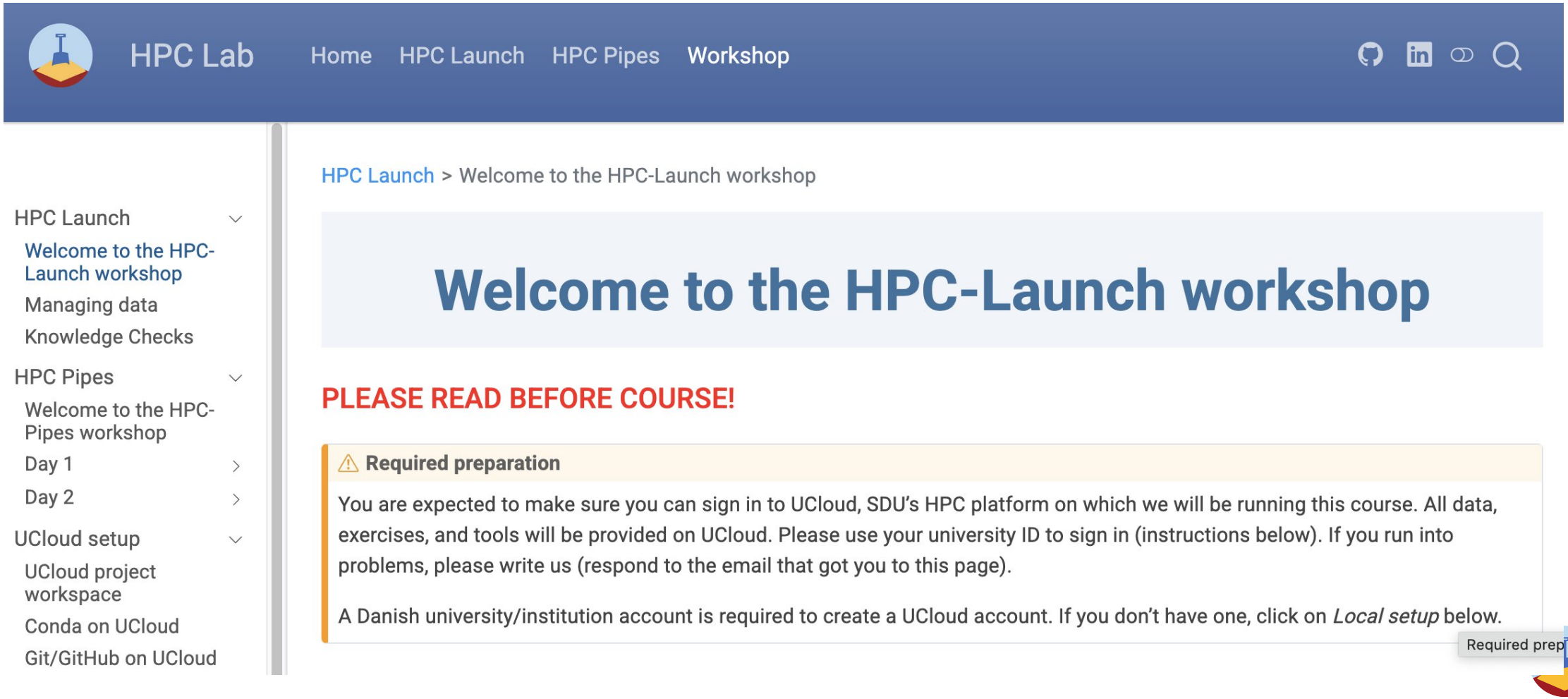
- **Introduction to HPC systems**
  - Understand HPC systems
  - Assess high performance computing goals
  - Secure compute and storage for your project

- **Set up computing environment essentials**
  - Access and authentication
  - Job scheduling basics
  - Data handling

- **Manage computational projects with a research data management (RDM) focus**
  - Data lifecycle
  - File system and storage structure
  - Reproducibility
  - Helpful tools to support RDM implementation

workshop agenda

https://hds-sandbox.github.io/HPC-lab/workshop/launch-requirements.html



Kai the coding champ

Ace at R Studio and
the Slack meme channel

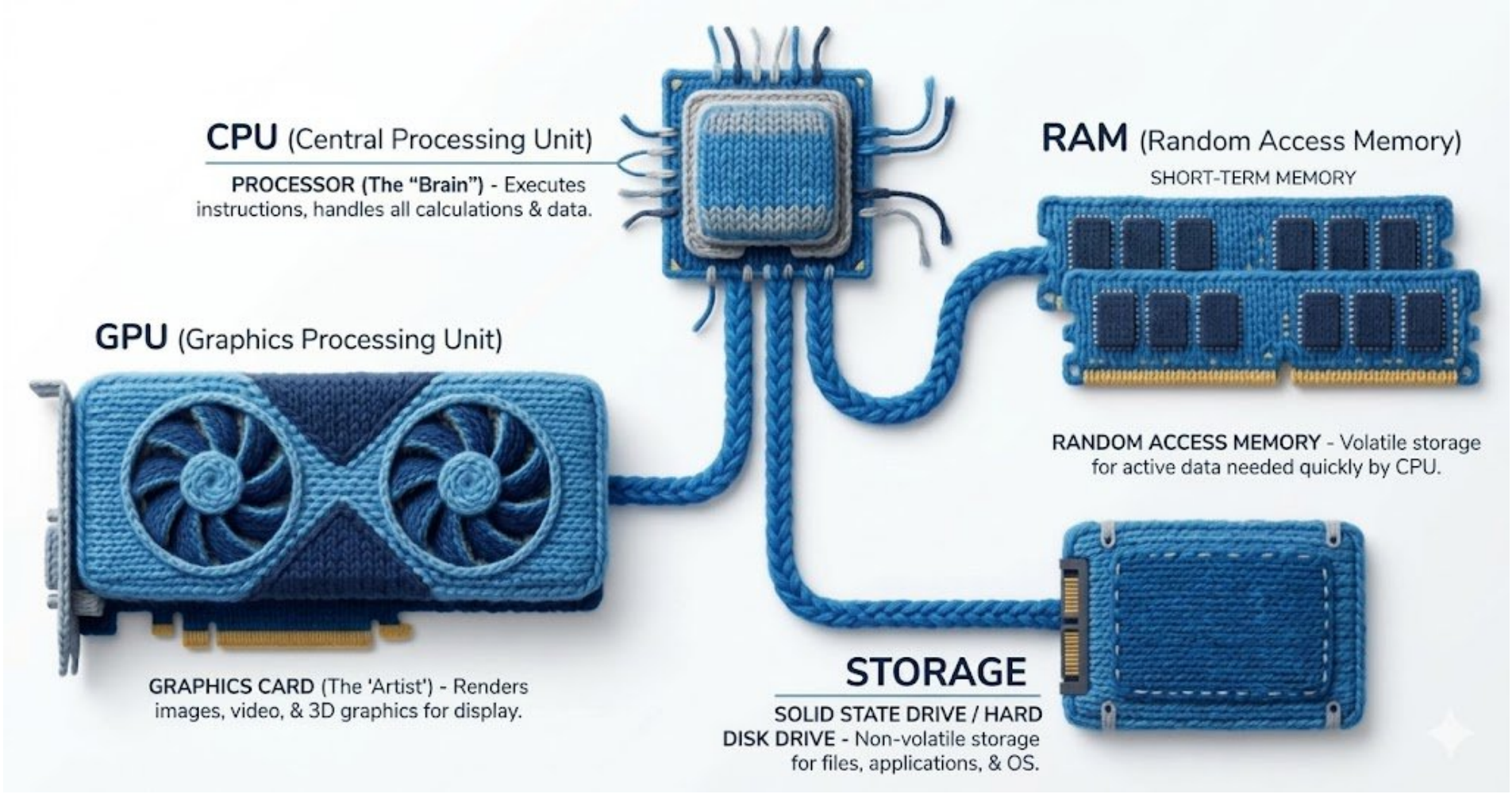
Paula the pipette pro

30 microtiter plates?
Make it 50!

Ralph the register wrangler

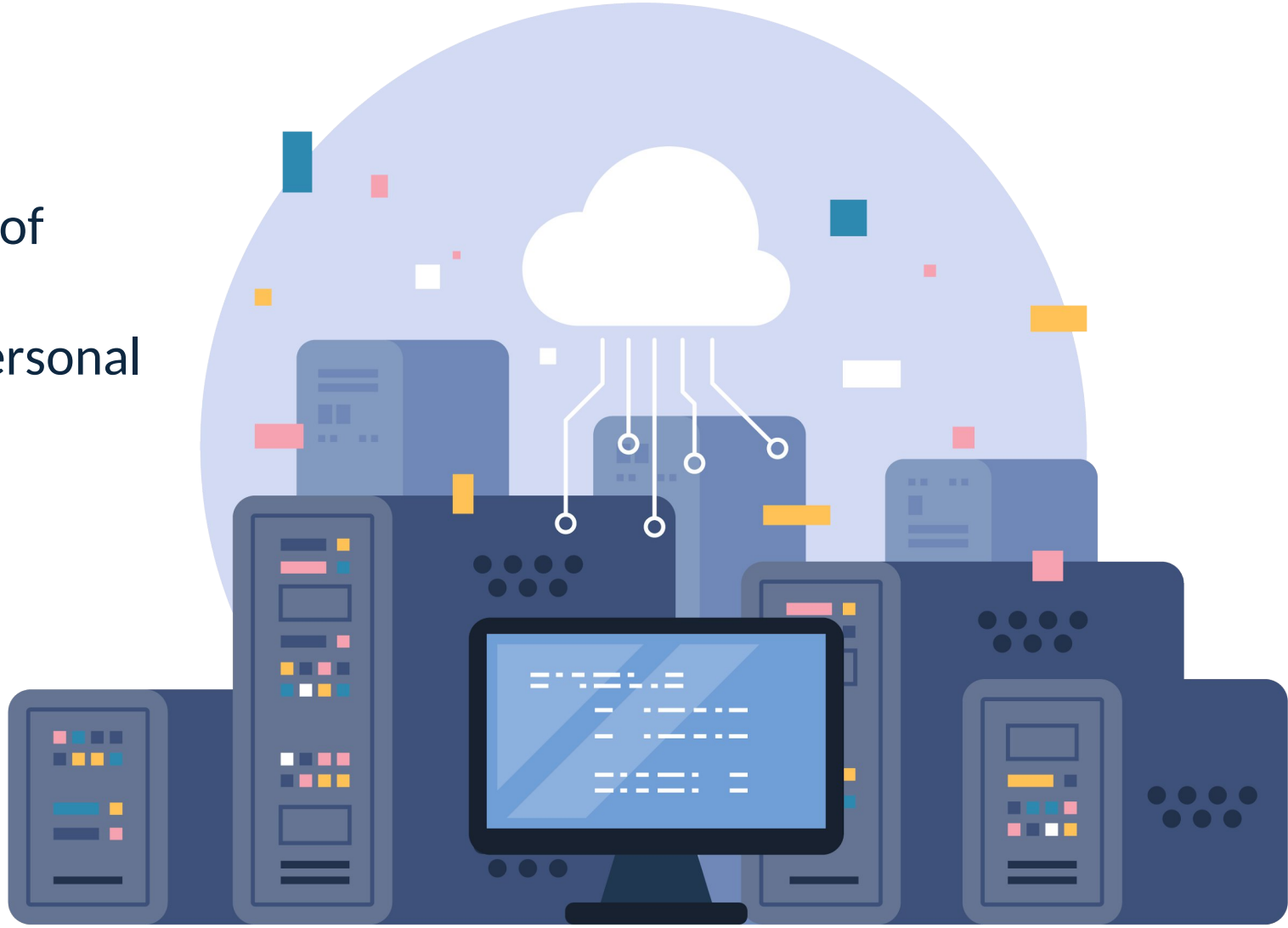
Can left join in his sleep,
but has DMP PTSD

hardware basics



High Performance Computing

Using a supercomputer or cluster of computers to perform jobs too computationally intensive for a personal computer (laptop or desktop)



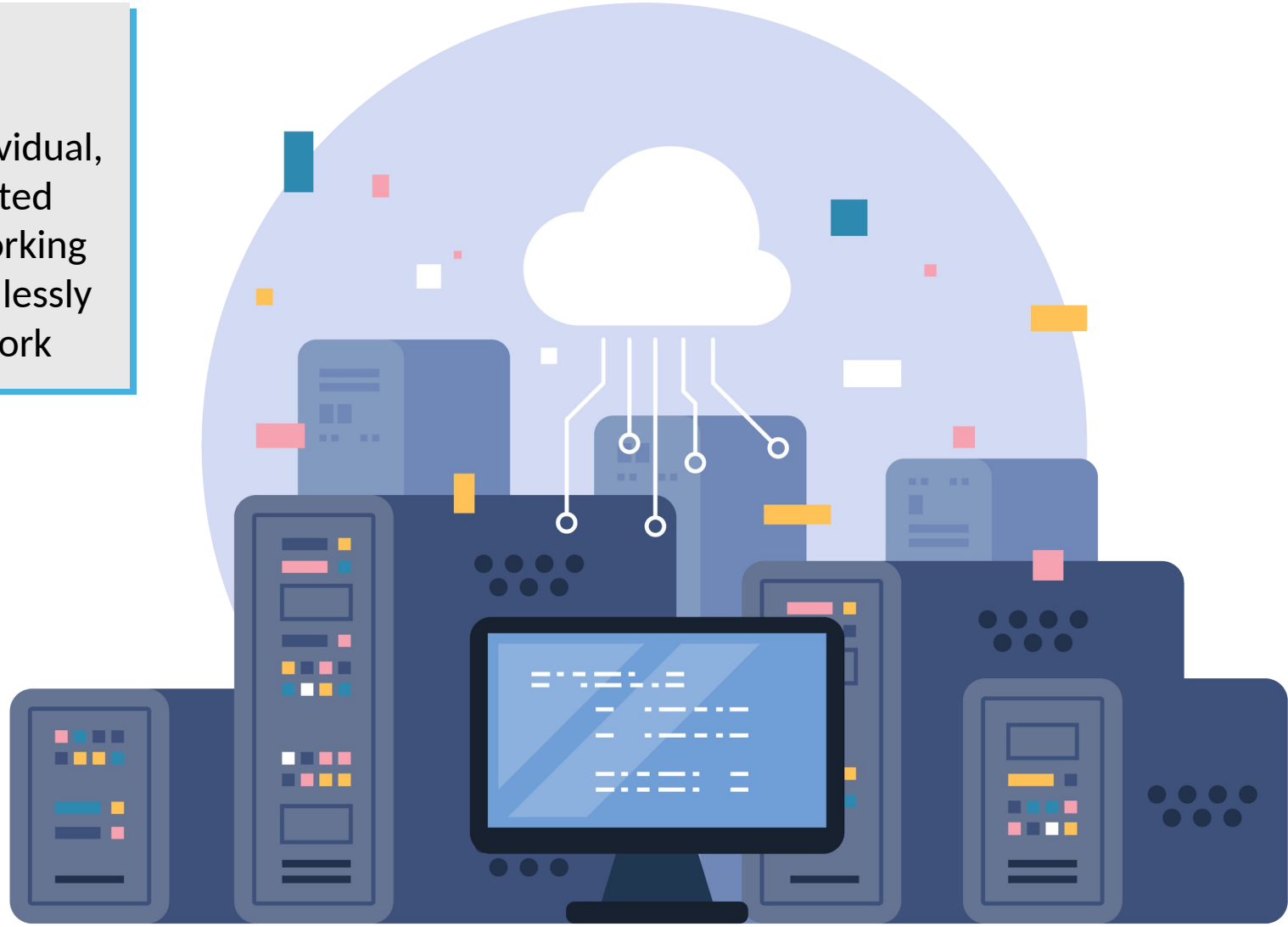
intro to HPC

Supercomputer

An exceptionally fast and powerful machine designed to process massive amounts of data

Cluster

A group of individual, interconnected computers working together seamlessly over a network



common HPC workflows

CPU-Bound

Tasks that require raw processing power for sequential calculation.

- Phylogenetic tree inference
- Pharmacokinetic modeling
- Basic epidemiology models

common HPC workflows

CPU-Bound

Tasks that require raw processing power for sequential calculation.

- Phylogenetic tree inference
- Pharmacokinetic modeling
- Basic epidemiology models

Memory-Bound

Workflows that need to hold massive, interconnected datasets in active memory all at once.

- Single-Cell RNASeq
- Metagenomics Assembly
- Genome-Wide Association Studies

common HPC workflows

CPU-Bound

Tasks that require raw processing power for sequential calculation.

- Phylogenetic tree inference
- Pharmacokinetic modeling
- Basic epidemiology models

Memory-Bound

Workflows that need to hold massive, interconnected datasets in active memory all at once.

- Single-Cell RNASeq
- Metagenomics Assembly
- Genome-Wide Association Studies

GPU-Based

Tasks involving massive amounts of relatively simple, repetitive math that can be parallelized.

- Protein Folding
- Training AI models
- Cryo-EM reconstruction

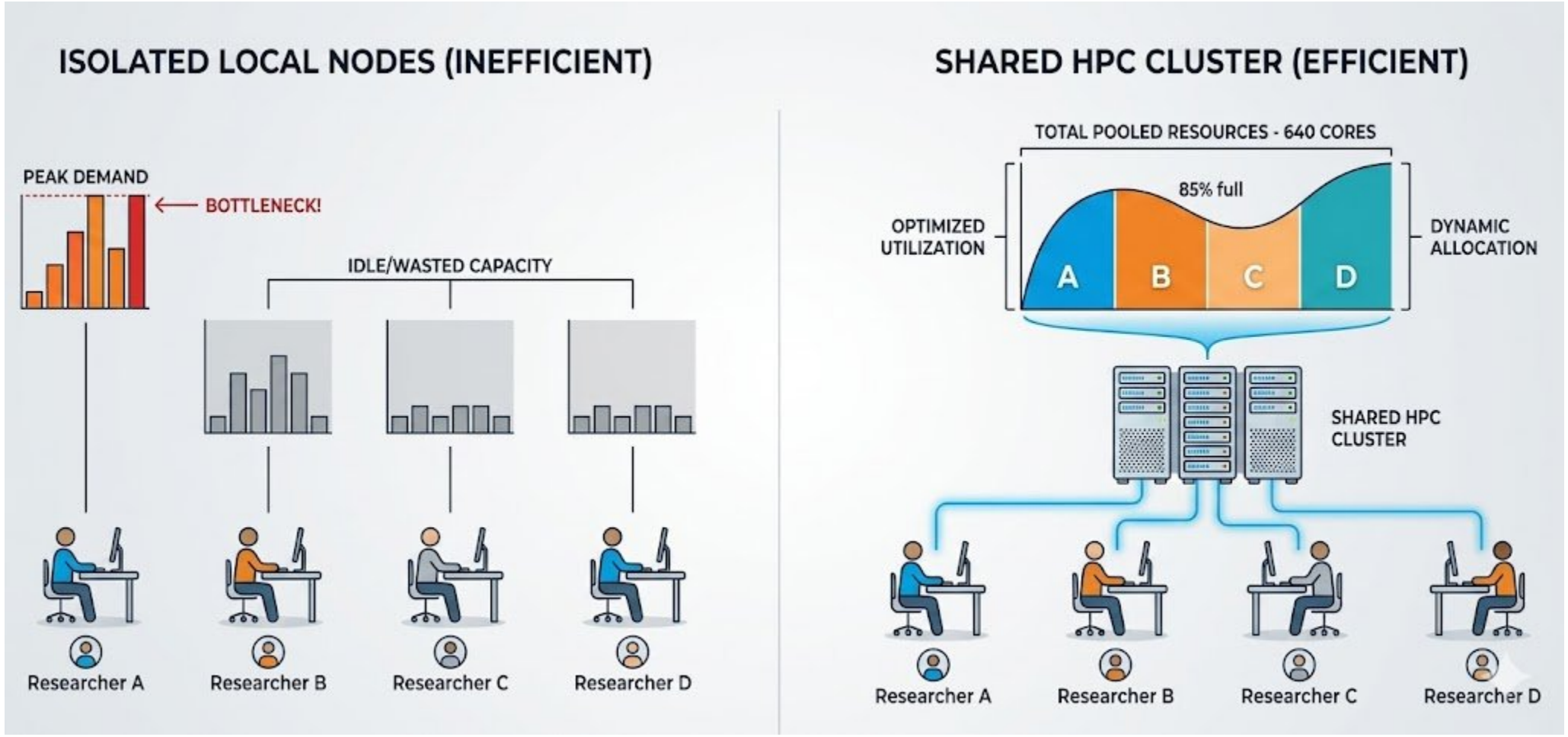
common HPC workflows

Data Sensitive

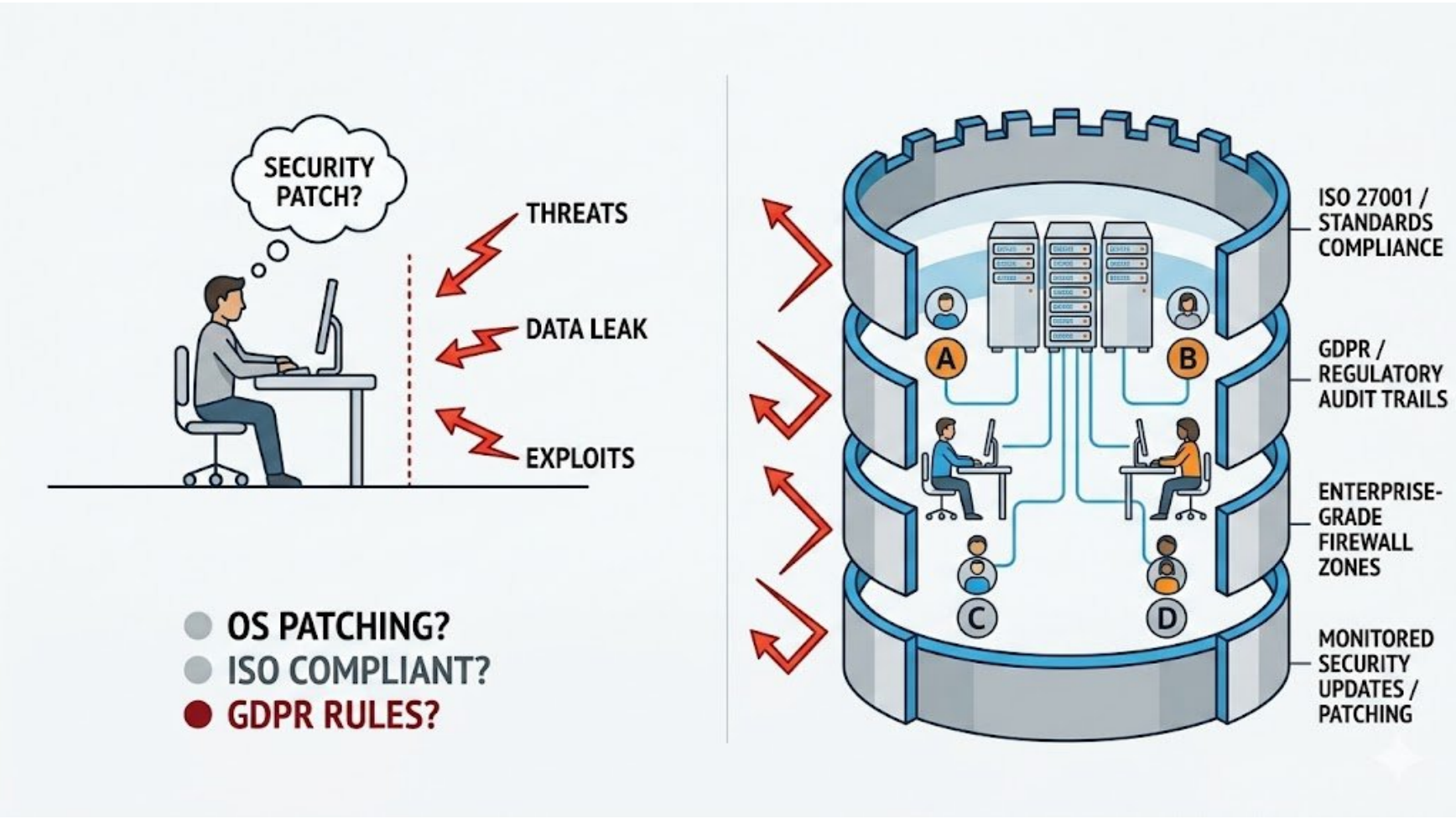
Requiring strict regulatory compliance (GDPR, NIS2) and isolated, secure computing environments.

- Population Health Studies
- Clinical Trial Databases
- Electronic Health Records

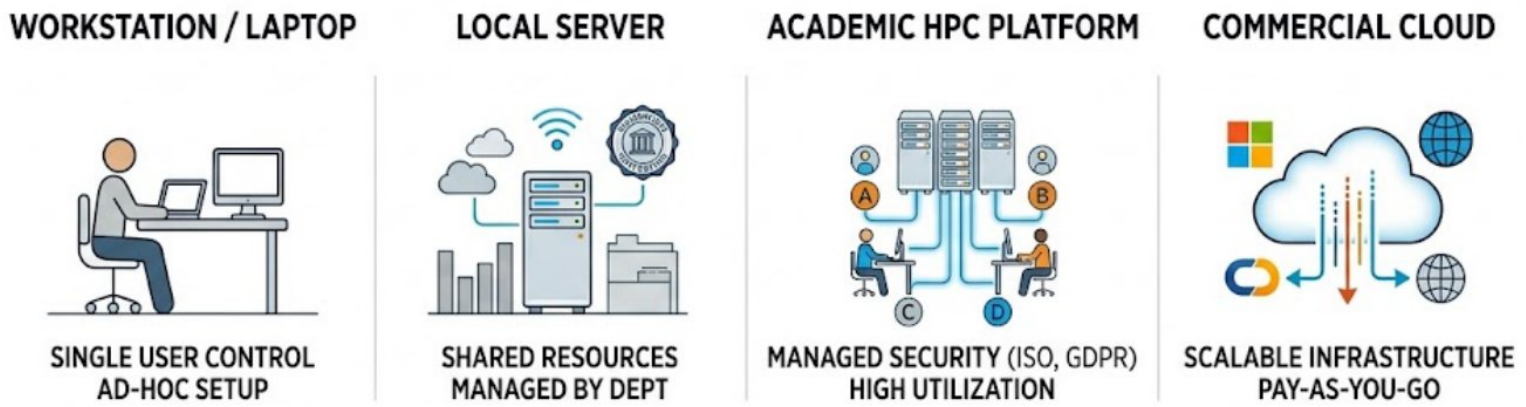
HPC advantages



HPC advantages



compute spectrum

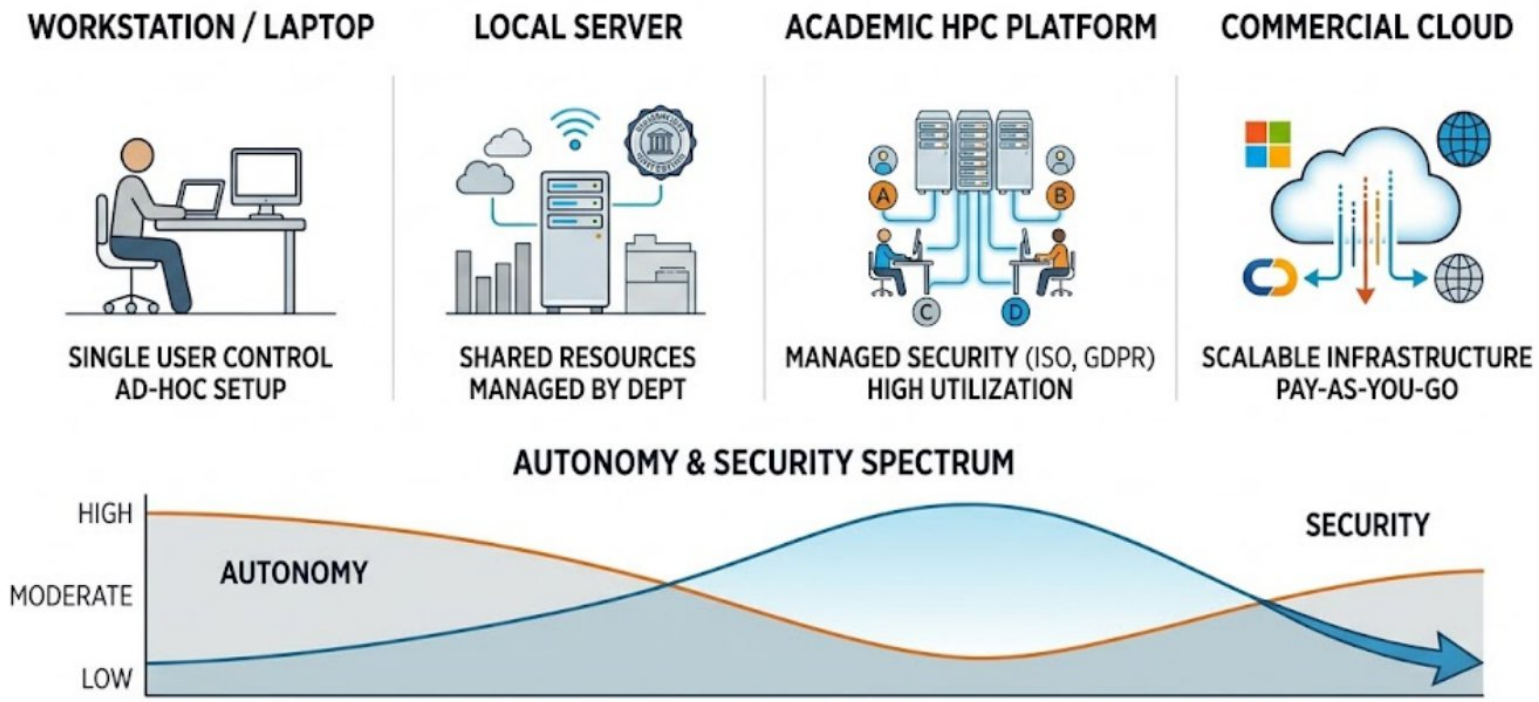


What is your experience with these different types of resources?

What are their pros and cons?

When is a laptop or workstation genuinely the *better* choice than HPC?

compute spectrum



pros and cons

Local PC / Cluster

+ admin rights to install
+ all resources are yours
+ smaller fixed costs

- fixed resources
- maintenance
- security

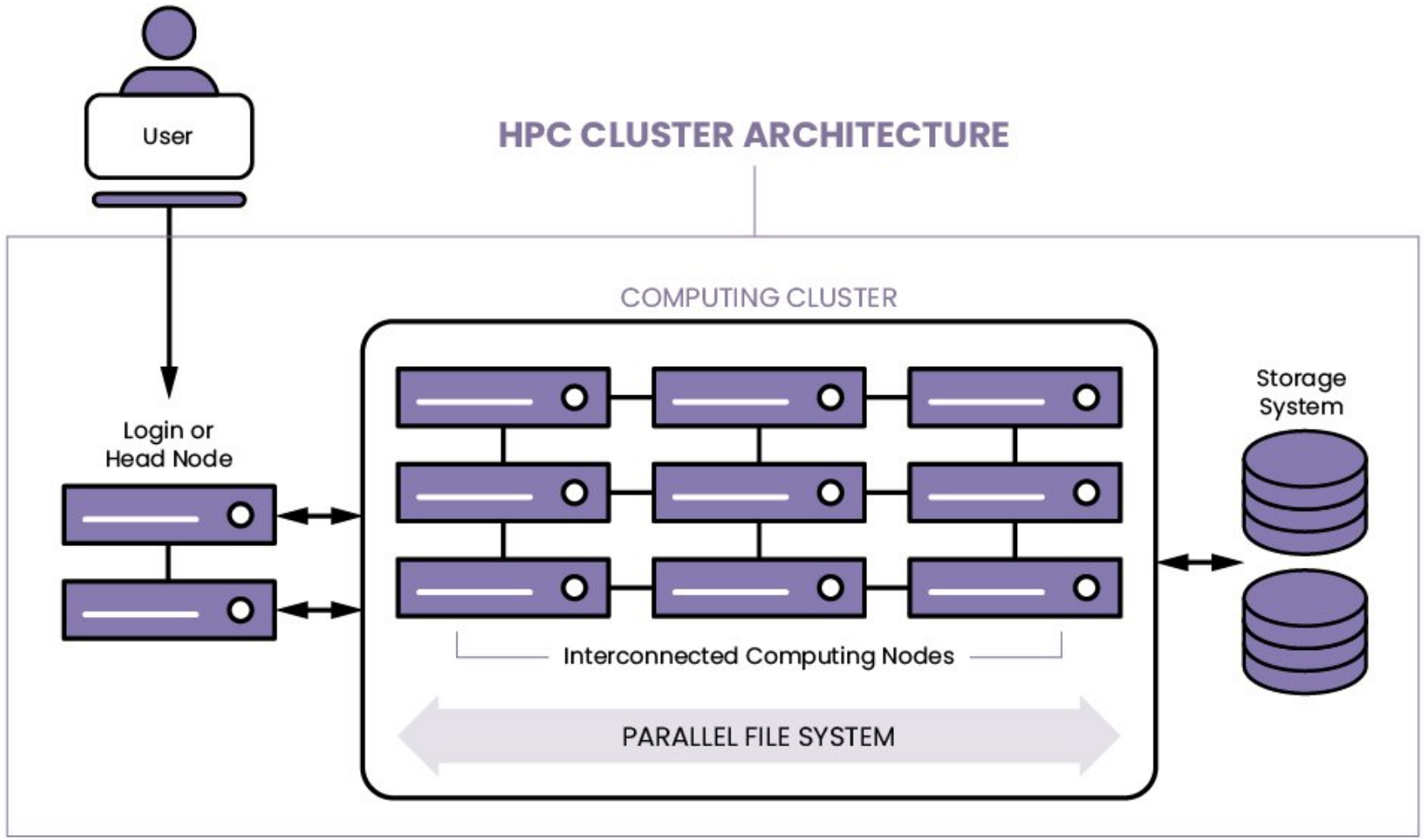
HPCs / Cloud

+ scalable resources
+ IT/sys admin management
+ IT security & privacy controls

- constraints on tool installs
- job setup hassle
- running costs

HPC architecture

HPC architecture

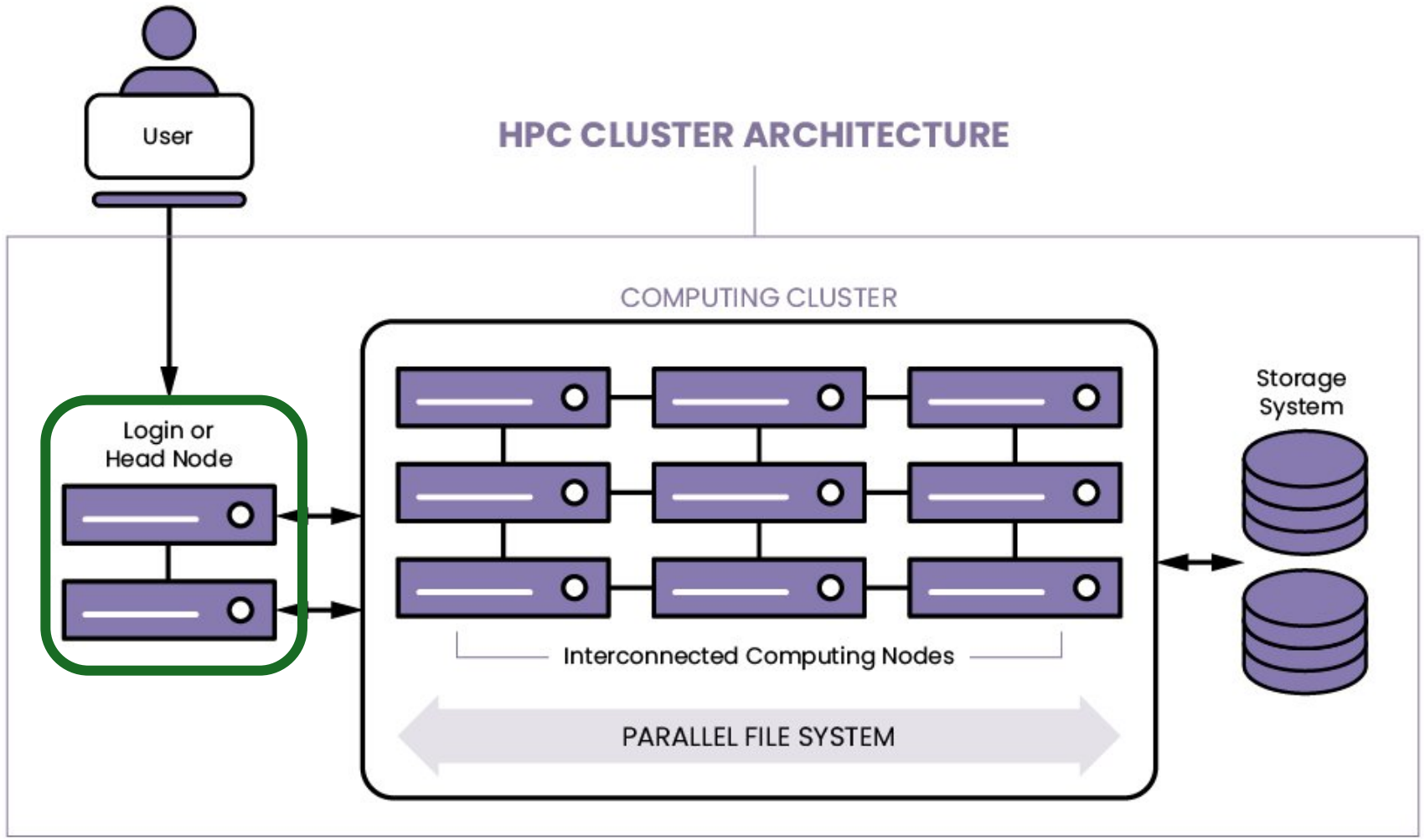


https://www.weka.io/learn/hpc/what-are-hpc-and-hpc-clusters/

Node

- Any physical device that can send, receive, or pass information
- In HPC, this term is normally used in reference to a compute node or a login node
- Nodes are interconnected to facilitate communication and data transfer between them

HPC architecture

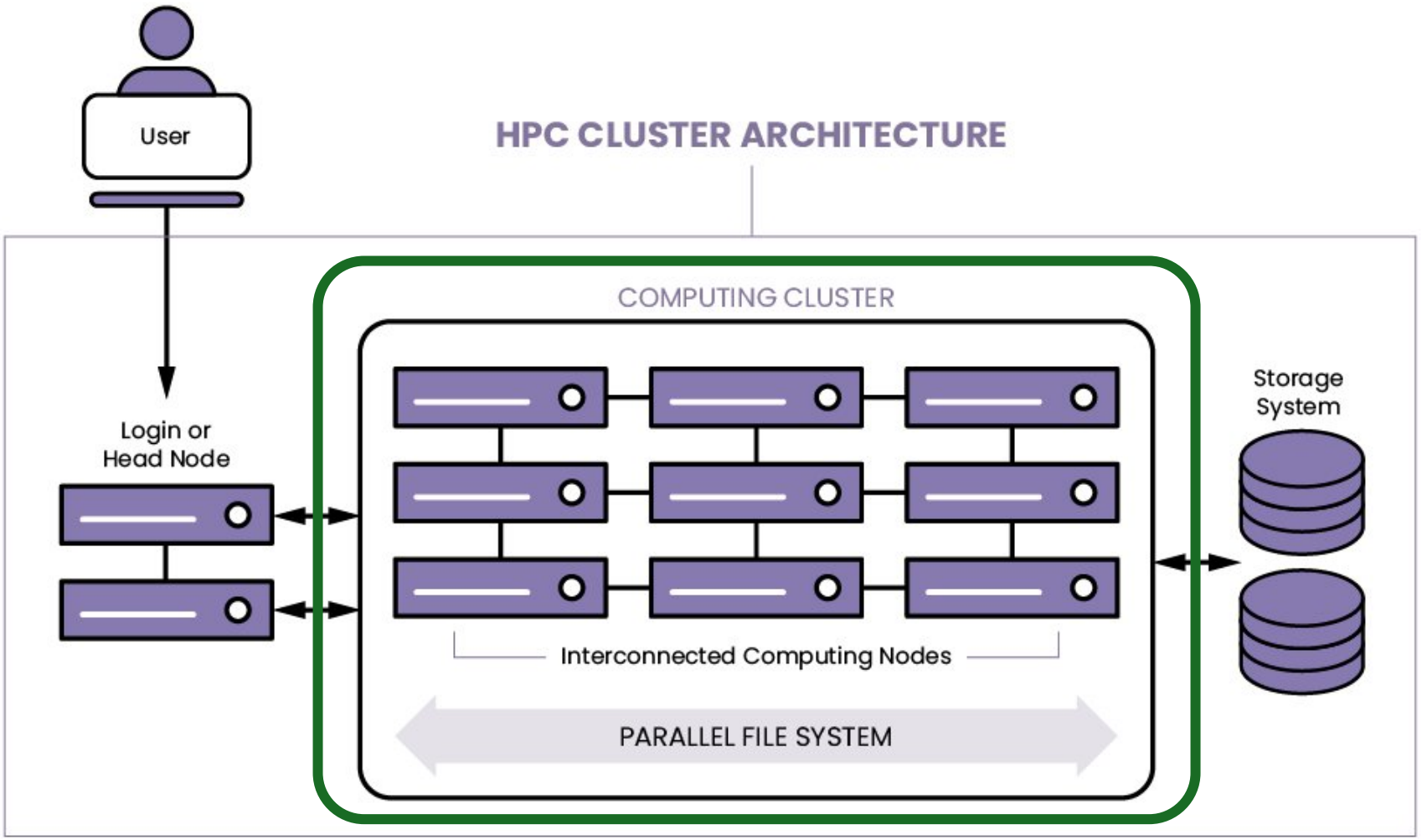


https://www.weka.io/learn/hpc/what-are-hpc-and-hpc-clusters/

Login Node

- A computer that acts as the front end to the HPC system, where users access (request) cluster resources and submit tasks for the computing nodes to perform

HPC architecture

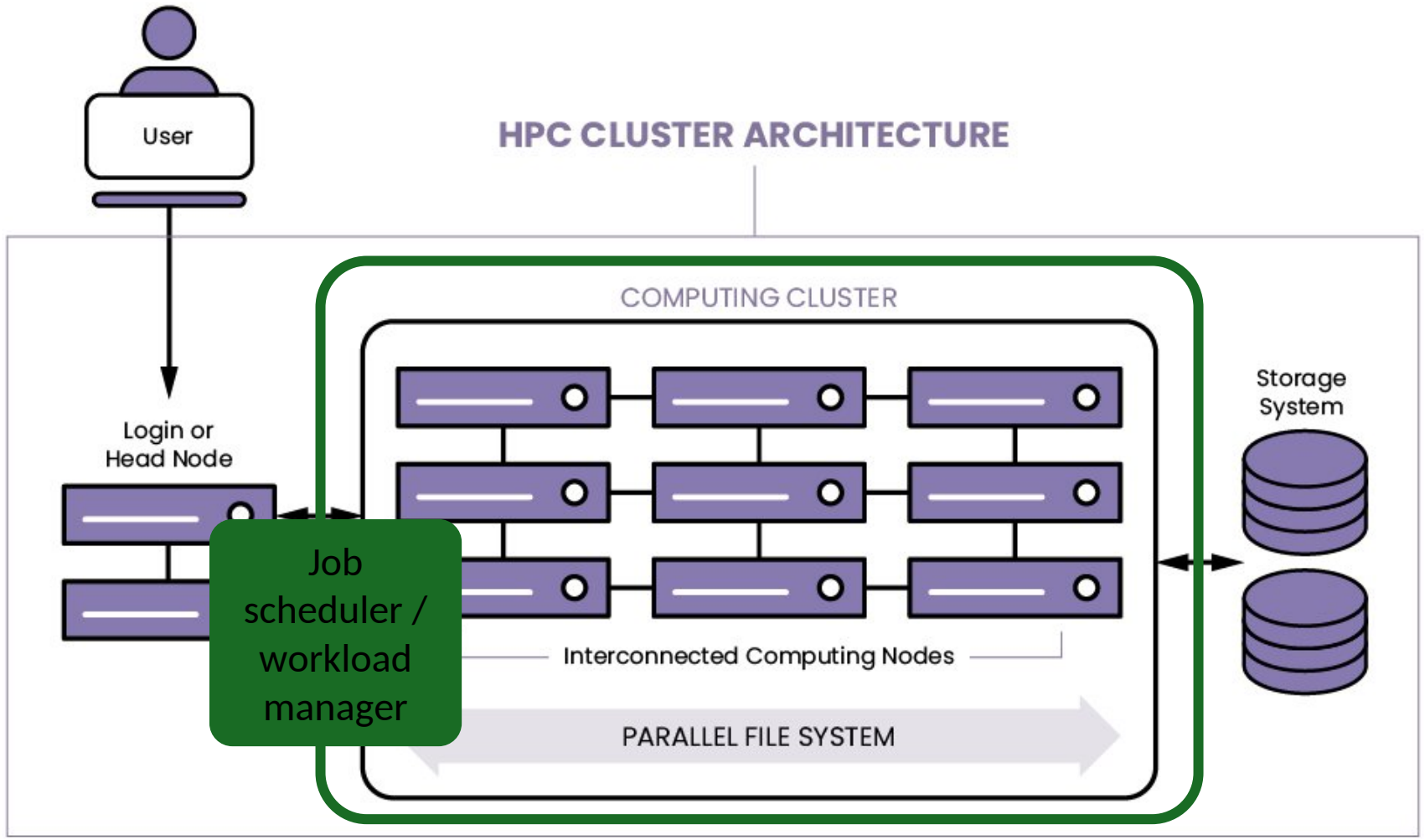


https://www.weka.io/learn/hpc/what-are-hpc-and-hpc-clusters/

Computing Cluster

- A (large) group of closely interconnected computers that work together as a single system to complete jobs

HPC architecture

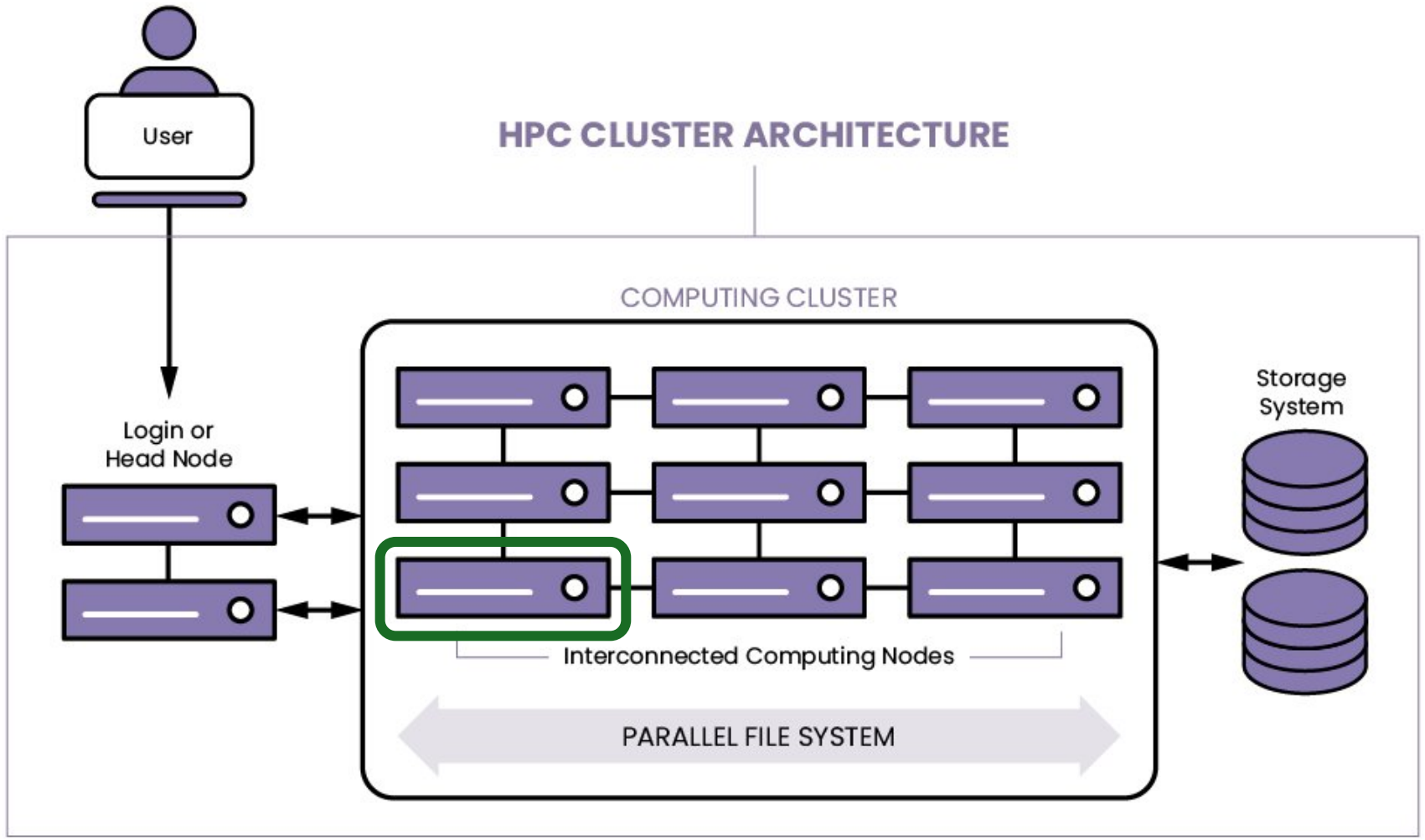


https://www.weka.io/learn/hpc/what-are-hpc-and-hpc-clusters/

Job scheduler

- Type varies by platform, but manages and schedules jobs (tasks) across compute nodes allocating resources to complete the job

HPC architecture

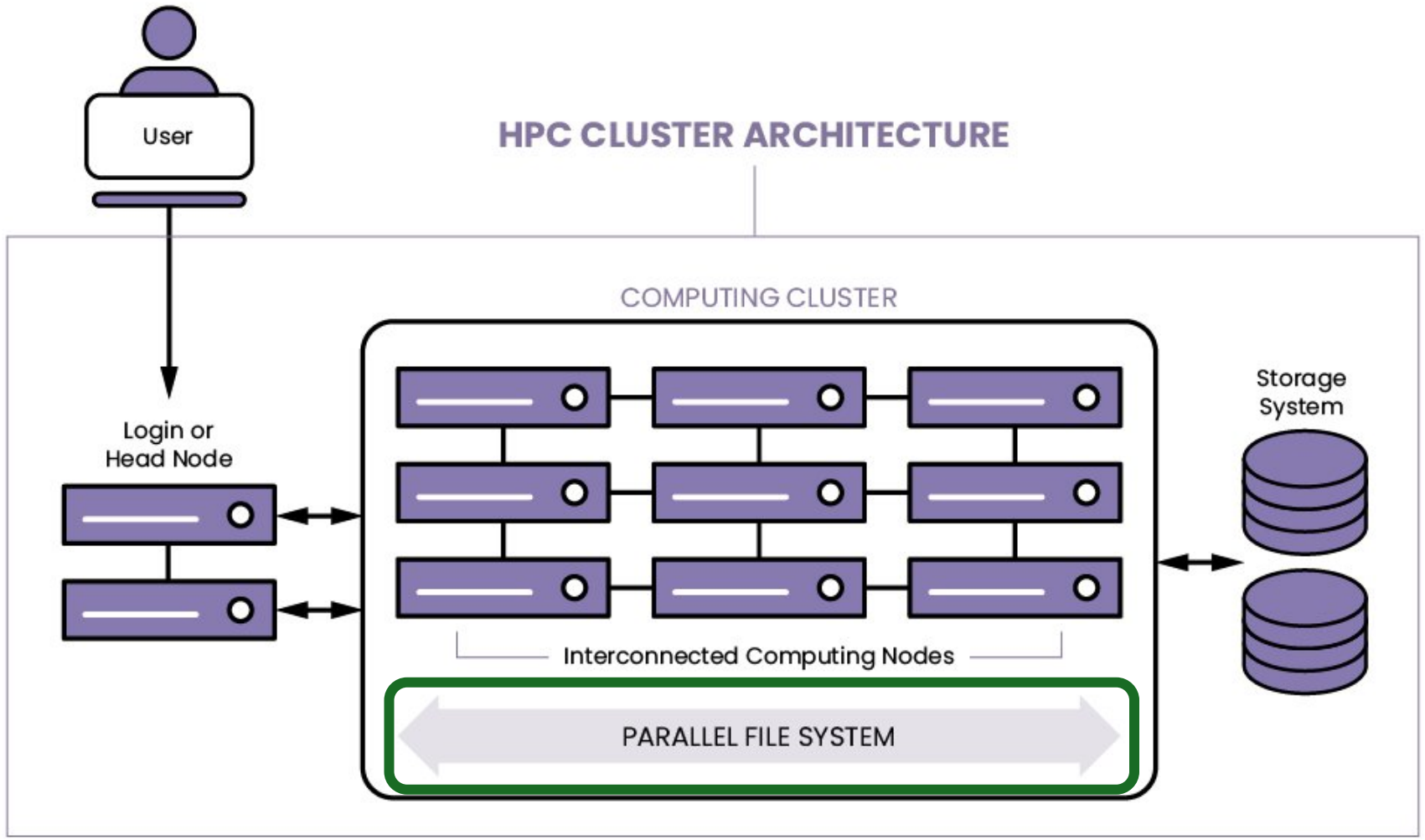


https://www.weka.io/learn/hpc/what-are-hpc-and-hpc-clusters/

Computing Node

- An individual computer within the compute cluster made up of a set of processors and their local memory
- ‘Size’ of the node traditionally varies by number of processors / amount of memory

HPC architecture

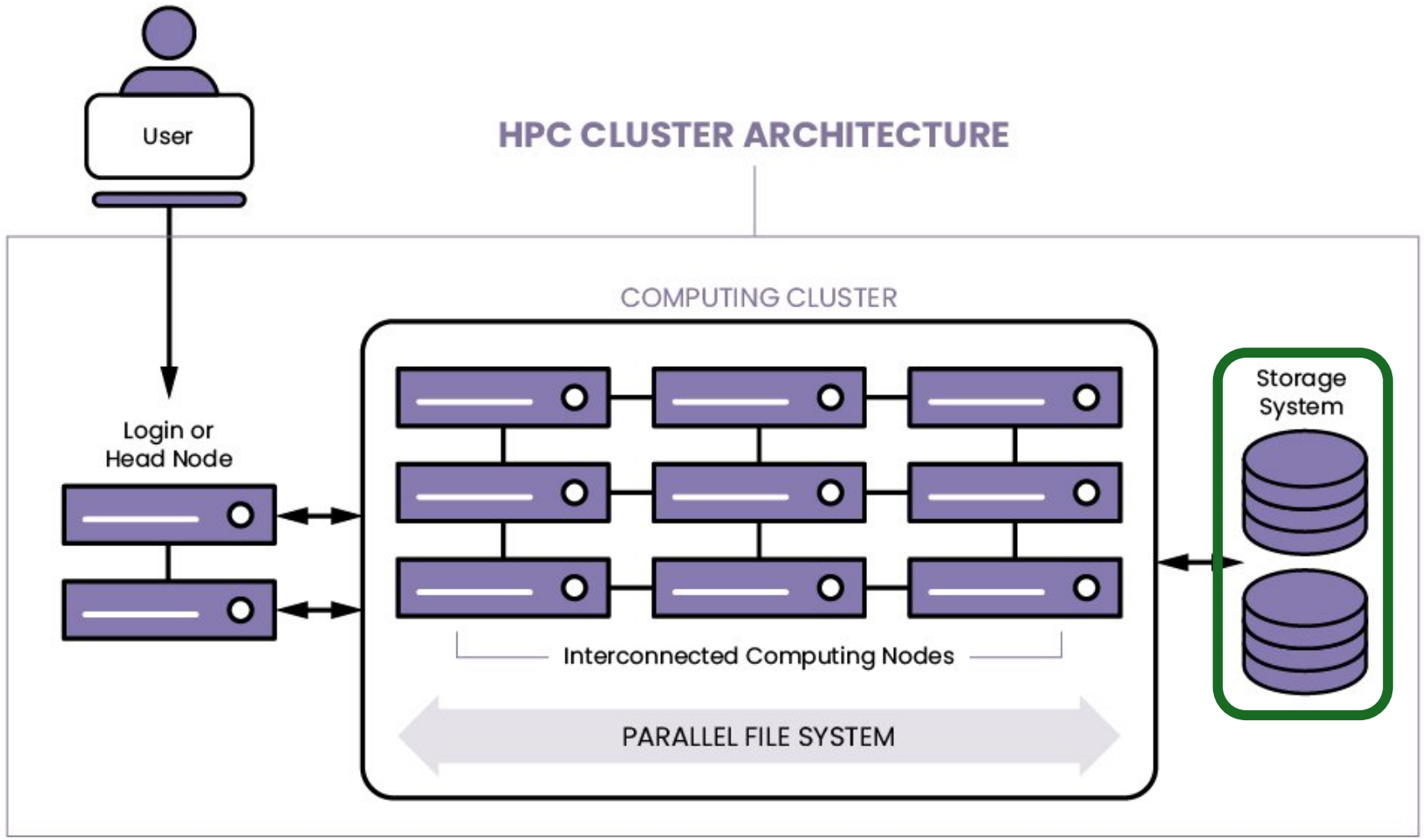


https://www.weka.io/learn/hpc/what-are-hpc-and-hpc-clusters/

Parallel File System

- Specialized for efficient read/write access to data storage by potentially many compute nodes (and independent users!) operating in parallel

HPC architecture

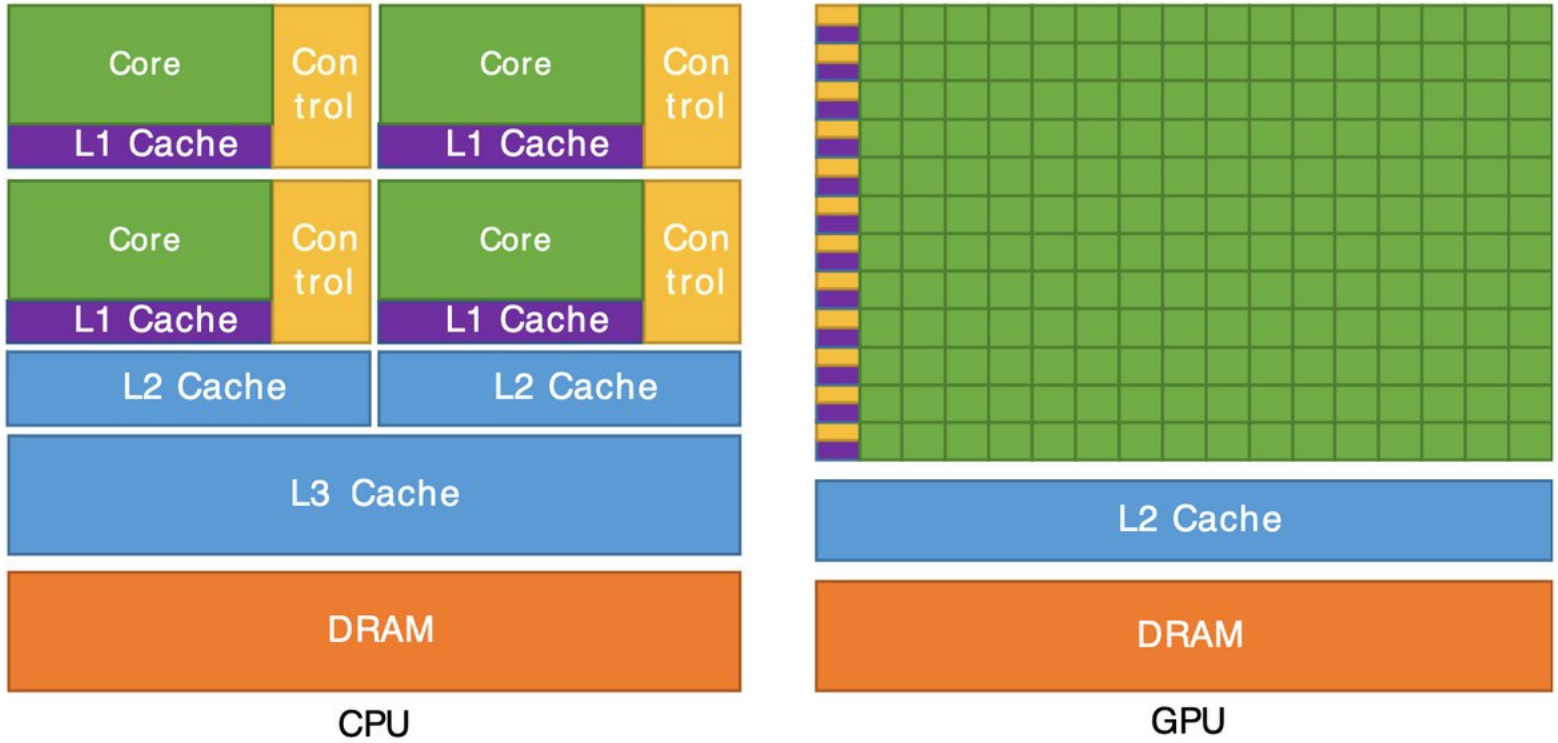


https://www.weka.io/learn/hpc/what-are-hpc-and-hpc-clusters/

Storage System

- Provides persistent storage for data and programs used by the HPC system
- Not your standard SSD system, requires fast, sophisticated orchestration of requests to high-end servers (data nodes) such that wait times are minimal

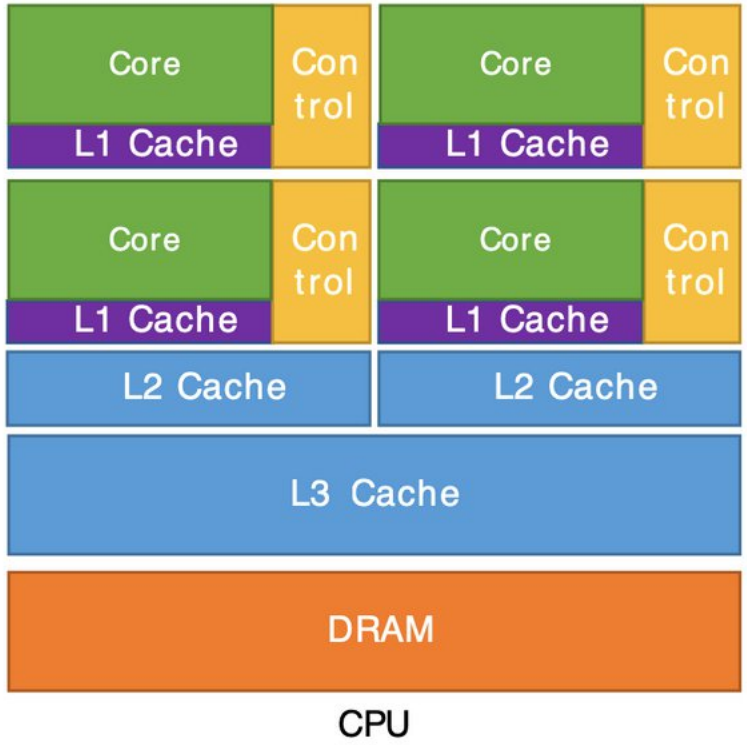
HPC components



Comparing the relative capabilities of the basic elements of CPU and GPU architectures.

https://cvw.cac.cornell.edu/gpu-architecture/gpu-characteristics/design

HPC components



CPUs

Modern CPUs are packaged together as multi-core processors on a single microprocessor chip

Each core has private L1 (and often L2) cache, while higher-level caches (L3) are shared across the chip.

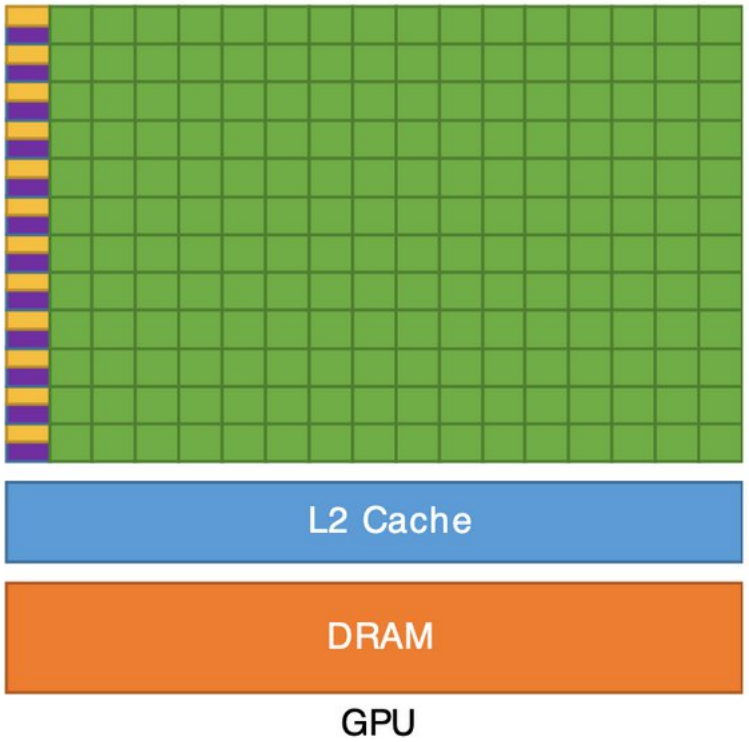
Thread = smallest execution unit (originally performed by a single core)

Job hierarchy: Job > task > process > thread

Use of terms task/thread/core can get complicated. Roughly, more cores means more parallel threads (if RAM isn't limiting)

https://cvw.cac.cornell.edu/gpu-architecture/gpu-characteristics/design

HPC components



GPUs

Many cores executing the same instructions on many data elements at once

High memory bandwidth (fast data movement)

Ideal for parallelization (note that controllers are per row of cores, not per core like a CPU!)

Good for massive data throughput, matrix math and large-scale simple computations

https://cvw.cac.cornell.edu/gpu-architecture/gpu-characteristics/design

file systems

A file system is the layer that turns raw storage into files and folders - names, permissions, sizes, and where each byte physically lives. On your laptop there are some common types of file system (ext4, APFS, NTFS) geared towards one user.

With HPCs, the challenge is that hundreds of nodes and users read and write the same storage at once. This is why HPC uses *networked* and *parallel* file systems (nfs, zfs, beegfs, lustre, etc.)

What makes one a better fit than another:

- A few big files vs many small files (the usual bottleneck)
- Raw throughput vs number of clients served simultaneously

beyond file systems

Classic file systems assume a manageable number of files, read more or less whole, on storage attached to your compute. As datasets grew to billions of files and petabytes, and as compute moved to the cloud, that model started to strain. Two ideas emerged in response:

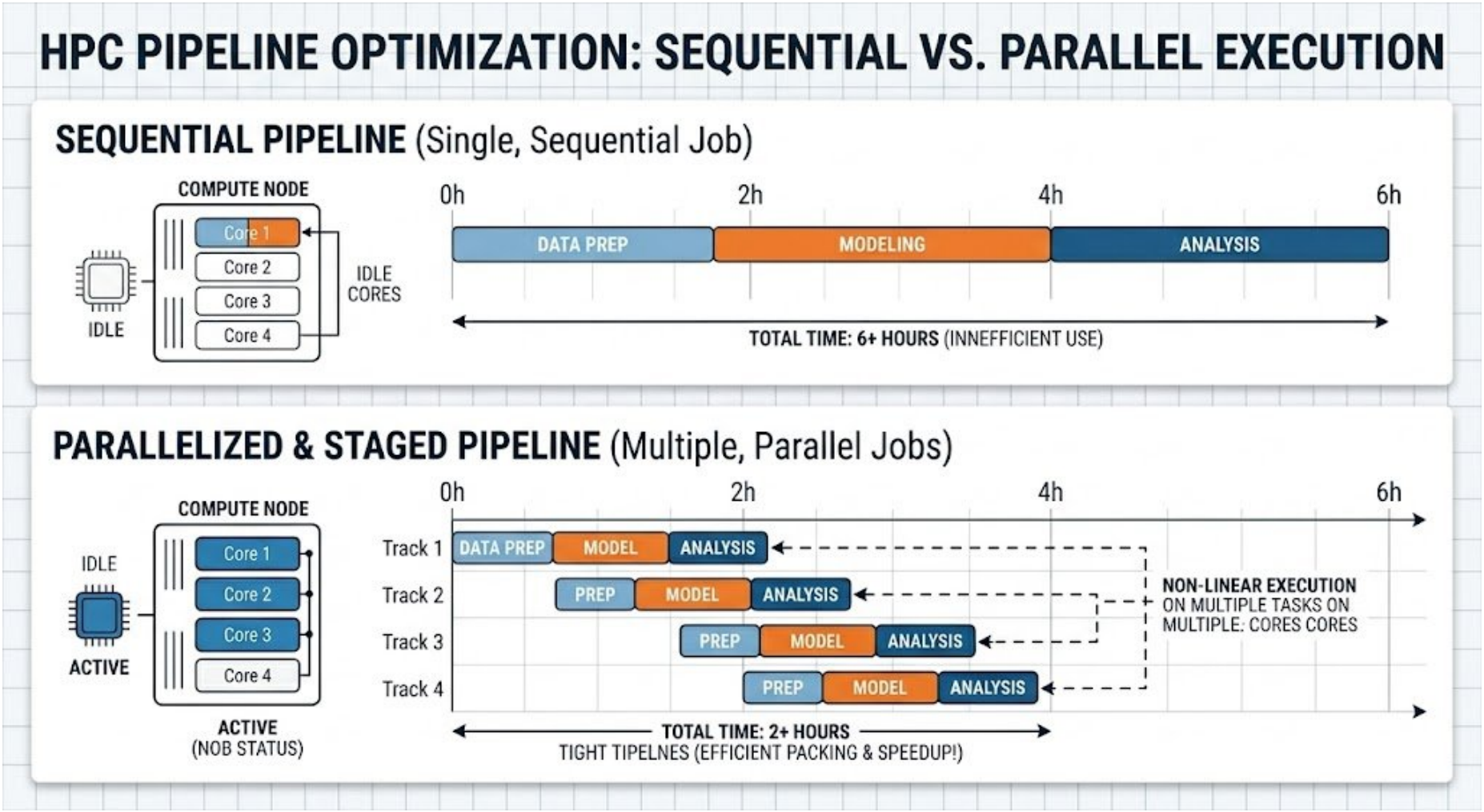
Object storage (e.g. S3) Drop the folder hierarchy; keep data as *objects* reached over the network by API. Near-limitless scale, low cost, and storage scales *independently* of compute.

Analytics formats (e.g. Parquet) Organize data so tools read, compress, and parallelize only the parts they need, instead of loading whole files. (Others in the same spirit: Zarr, HDF5, Arrow)

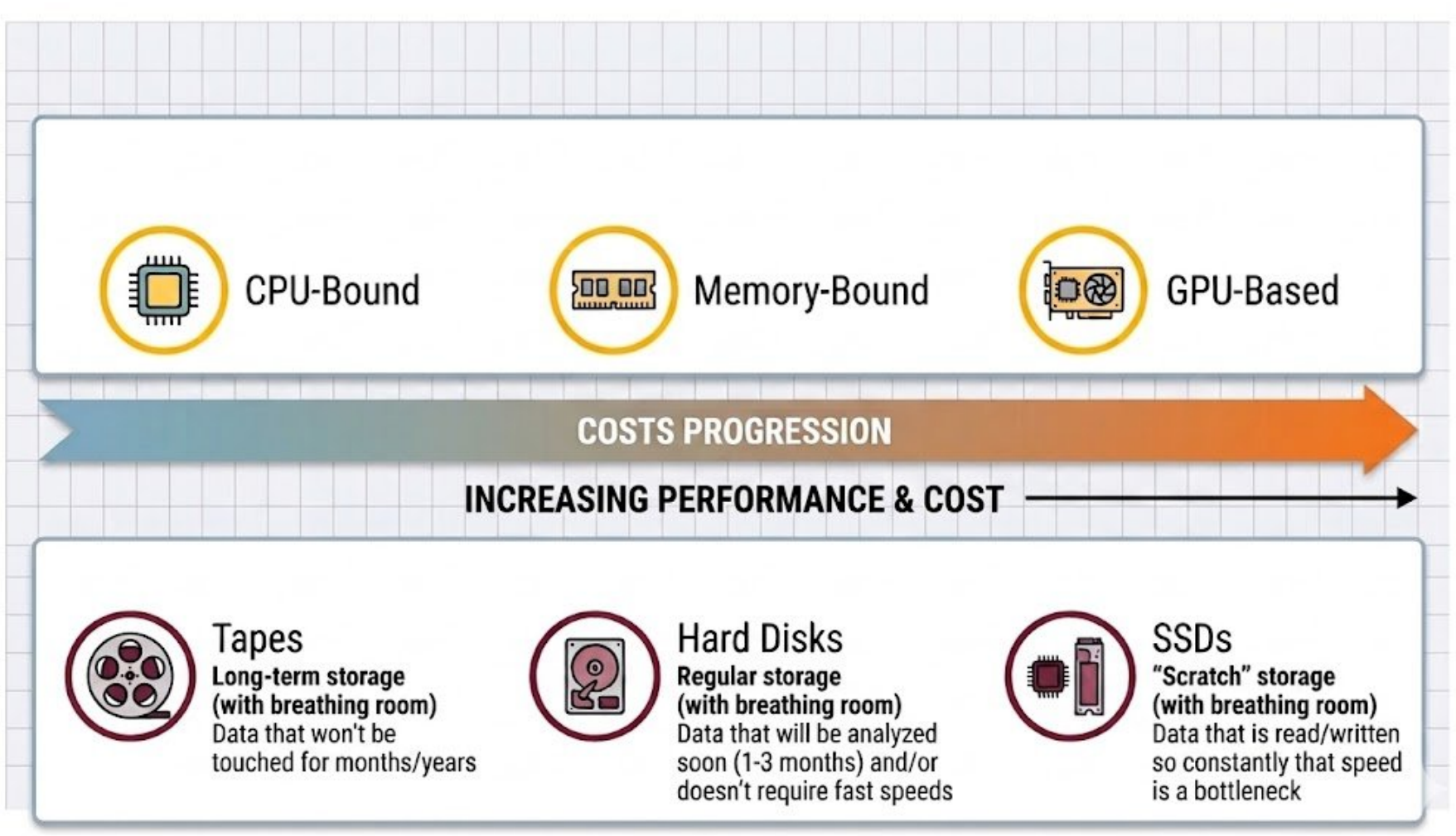
common HPC file structure

Location	Use Case(s)	Watch out for
home (~/)	Configs, small files	Small quota, backed up; don't run jobs or stage data here
scratch/	Active job I/O, large temp files	Not backed up, auto-purged after N days
project/, work/	Datasets shared with your group	User access and permissions usually fragmented
archive/	Finished results, raw data to keep	Slow, cheap, long-term storage after project finishes

HPC job design



HPC job design



summary

- **A cluster is parts working as one.** Login node(s) to access and submit, many compute nodes to do the work, a scheduler to allocate them, and shared parallel storage.

- **Match the hardware to the workload** CPUs for general and many-core parallel work, GPUs for massively parallel, matrix-heavy throughput. More cores only help if your tool/script can actually use them.

- **Storage is not one thing.** There are differences in how file systems work, and bottlenecks that need to be understood. For any given provider, home, scratch, project, and archive each might have different speed, backup, and quota rules.

Intro to Research Data Management & GDPR

what is RDM

Research Data Management (RDM) is a set of practices to ensure your research output is **organized**, **shareable** and **reproducible**.

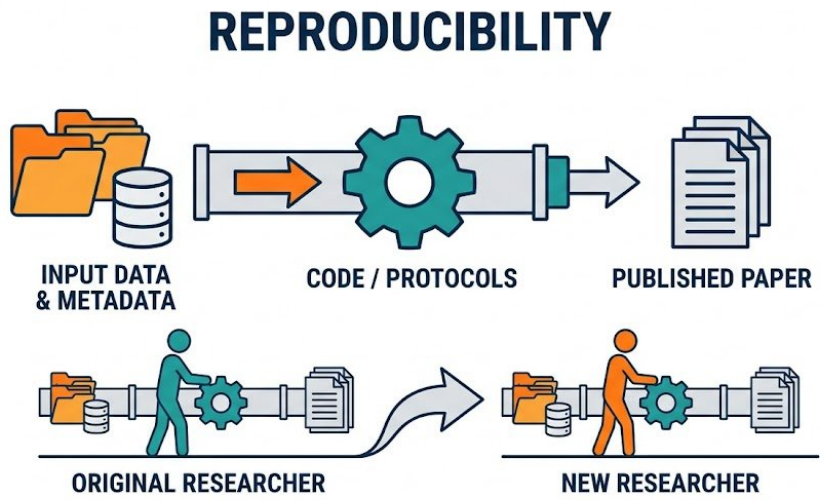
If you work with sensitive data, good RDM practices are not only recommended but legally mandated.

There are personal and institutional incentives for you to make sure your data is **FAIR**. Working with HPCs or cloud systems add a financial incentive to it as well.

FAIR data

Findable
Accessible
Interoperable
Reusable

reproducibility



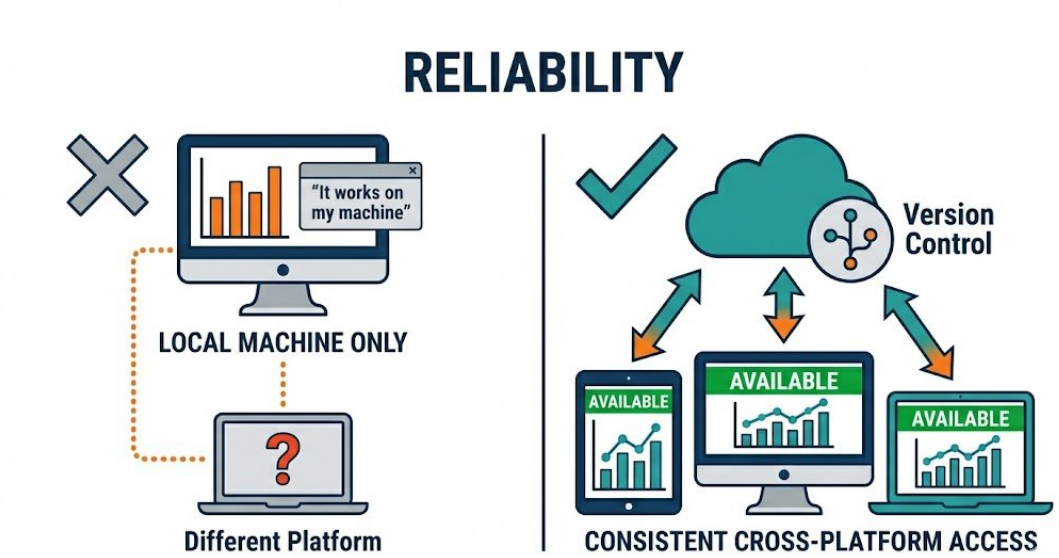
Universities have a lot of turnover: people usually stay for periods of 2-5 years and then move to another position.

Reproducibility is not only about preserving protocols and workflows within the same group, but also ensuring the academic community at large can benefit from your work.

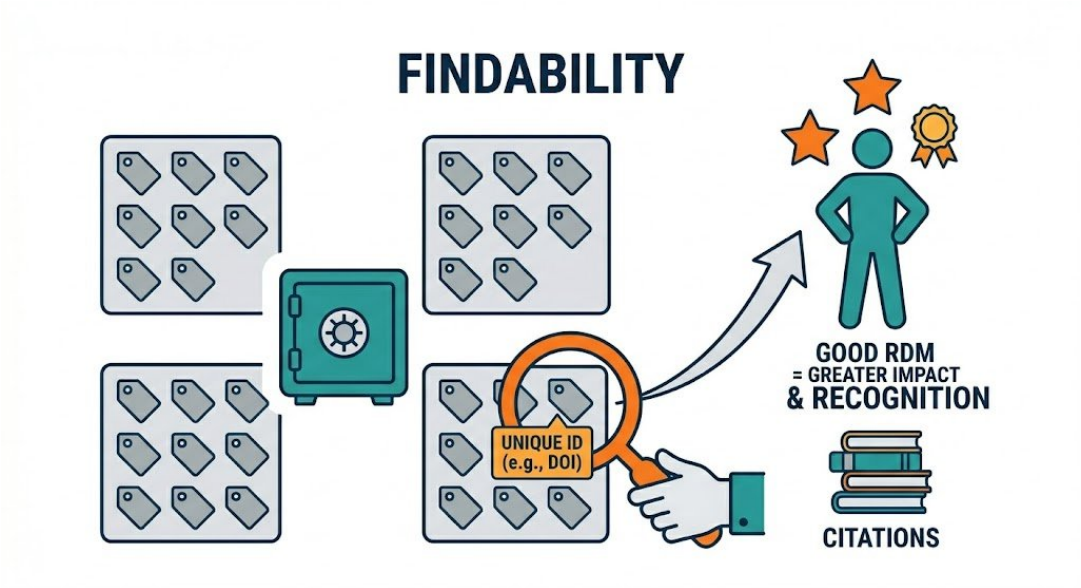
reliability

Creating a reliable workflow means your findings will be reproducible in a variety of environments. Any pipeline or application that only runs on your own computer is not mature enough for publication.

Good RDM practices are helpful at an individual level too. For example, having a strategy for version control and backups prevents loss of time and money.



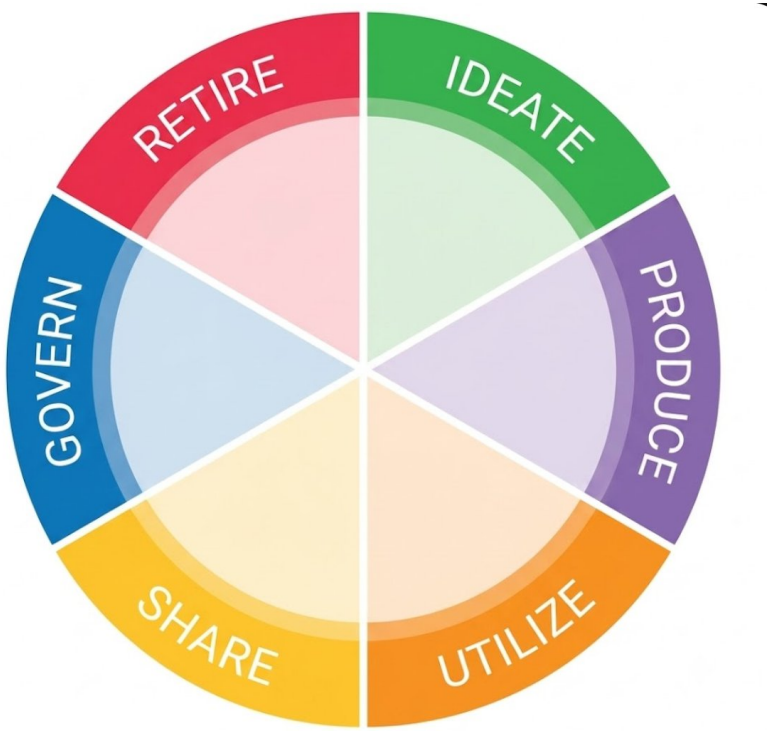
findability



Any lab benefits from having data and workflows preserved with good RDM practices. That means less redundancy and dependencies.

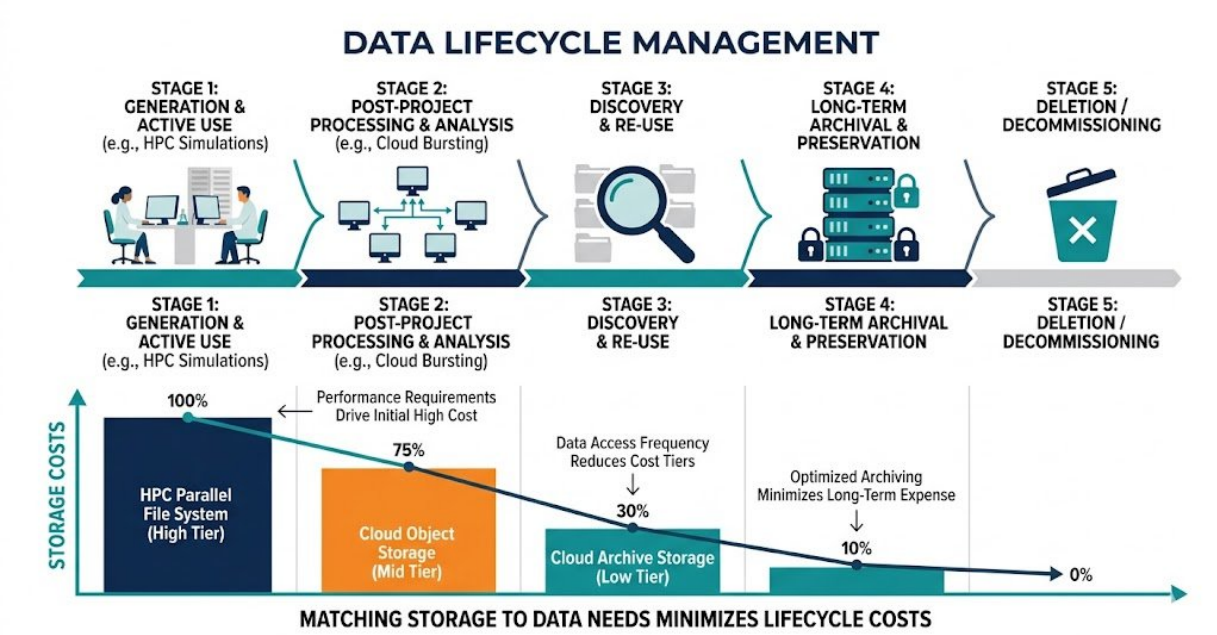
Another great incentive for RDM is that well-organized work, especially if it can be reproduced entirely by readers, often receives more attention and citations.

data lifecycle



- **Create** Generate or collect data; classify its sensitivity and plan its structure from day one
- **Store** Keep it in the right place: secure, backed-up storage with appropriate access controls
- **Use** Analyze the data with reproducible scripts and environments, working only with the data you need
- **Share** Give collaborators access under the right agreements, make outputs FAIR
- **Archive** Preserve essential data and code in a repository with a DOI and a license
- **Destroy** Securely delete data when its retention period ends, or when a subject invokes the right to erasure

costs



We are producing and consuming more data than ever before. In some cases, data needs to be preserved from 5 to 25 years after publication, which can add up to huge costs over time.

A good understanding of your data lifecycle can help generate strategies for cost optimization to minimize those costs.

challenges

If good RDM is so advantageous, then why isn't it more widespread?

Many people never hear about it, or have misconceptions about what RDM means.

Academic incentives are often skewed towards the creation of prototypes rather than stable, reproducible artifacts.

Lastly, it takes active effort to make things reliable and reproducible.

Outside academia, RDM is very prevalent. In the pharma industry, every step of software development needs to be reproducible, accessible and documented (SDLC, GxP)

questions

- Are you a PhD student/postdoc or are you permanent staff?
- Do you have funding to spend on compute?
- Do you need to work with sensitive data?

sensitive data

Sensitive personal data definition

- Any information requiring elevated legal protection because its exposure could significantly impact an individual's fundamental rights, freedoms, or privacy.
- In practice: health records, biological samples, medical imaging, some survey questions (sexual orientation, religious beliefs, etc.) can all fit into this definition.

Security concerns

- Leaks via system intrusions, ransomware, lax security standards
- Data alteration and/or destruction
- Non-compliant handling by researchers (& associated users)

pseudonymized vs anonymized data

	Pseudonymized	Anonymized
Definition	Identifiers replaced with a key	Identifiers irreversibly removed
Re-identifiable?	Yes, with the key	No (if done properly)
Under GDPR?	Yes, still personal data	No, outside GDPR scope
In practice	Common in research	Genuinely hard; rarely fully achieved

most relevant GDPR concepts for HPCs

Shared Responsibility Model

Right to Erasure

Data Processing Agreements

most relevant GDPR concepts for HPCs

Shared Responsibility Model

Many entities share the responsibility for handling your data. You, as the data owner, have the burden to ensure everything works well.

Right to Erasure

Data Processing Agreements

most relevant GDPR concepts for HPCs

Shared Responsibility Model

Many entities share the responsibility for handling your data. You, as the data owner, have the burden to ensure everything works well.

Right to Erasure

The ability for people to remove their data from a dataset. This is particularly difficult if you have poor data governance/RDM.

Data Processing Agreements

most relevant GDPR concepts for HPCs

Shared Responsibility Model

Many entities share the responsibility for handling your data. You, as the data owner, have the burden to ensure everything works well.

Right to Erasure

The ability for people to remove their data from a dataset. This is particularly difficult if you have poor data governance/RDM.

Data Processing Agreements

You cannot simply use any provider, and even if the provider is approved you will need a DPA. In practice, this is the most important criterion when choosing a HPC provider for sensitive data.

data processing agreements

A DPA is a legally binding contract that ensures an external provider processes your sensitive biomedical data strictly according to your rules and GDPR standards.

Data Controller

You / Your PI / KU

Determines the “why” and “how" of data processing. Responsible for study design, ethical approvals and consent collection.

Data Processor

HPC / Cloud provider

Provides compute and storage. Legally forbidden from doing anything with the data except what you explicitly instruct them to do.

data processing agreements

A DPA is a legally binding contract that ensures an external provider processes your sensitive biomedical data strictly according to your rules and GDPR standards.

Data Controller

You / Your PI / KU

Determines the “why” and “how" of data processing. Responsible for study design, ethical approvals and consent collection.

Data Processor

HPC / Cloud provider

Provides compute and storage. Legally forbidden from doing anything with the data except what you explicitly instruct them to do.

Fun fact: even with a DPA, any American provider may be legally compelled by the US government to reveal your data

GDPR in practice

You're approved to study whether biomarker X predicts type 2 diabetes in a hospital cohort. The data controller gives you a pseudonymized export.

GDPR in practice

You're approved to study whether biomarker X predicts type 2 diabetes in a hospital cohort. The data controller gives you a pseudonymized export.

Data protection The data needs to be secured and controlled at all stages.

GDPR in practice

You're approved to study whether biomarker X predicts type 2 diabetes in a hospital cohort. The data controller gives you a pseudonymized export.

Data protection The data needs to be secured and controlled at all stages.

✓ Choose a provider with adequate capabilities (jurisdiction, technical measures, DPA)
✗ Don't upload data to an AI provider, or share it within your group via insecure systems (ie e-mail)

GDPR in practice

You're approved to study whether biomarker X predicts type 2 diabetes in a hospital cohort. The data controller gives you a pseudonymized export.

Data minimization The export has 200 variables × 50,000 patients. Your study needs ~8 variables and the 2,000 patients meeting your criteria.

Data protection The data needs to be secured and controlled at all stages.

✓ Choose a provider with adequate capabilities (jurisdiction, technical measures, DPA)
✗ Don't upload data to an AI provider, or share it within your group via insecure systems (ie e-mail)

GDPR in practice

You're approved to study whether biomarker X predicts type 2 diabetes in a hospital cohort. The data controller gives you a pseudonymized export.

Data protection The data needs to be secured and controlled at all stages.

✓ Choose a provider with adequate capabilities (jurisdiction, technical measures, DPA)
✗ Don't upload data to an AI provider, or share it within your group via insecure systems (ie e-mail)

Data minimization The export has 200 variables × 50,000 patients. Your study needs ~8 variables and the 2,000 patients meeting your criteria.

✓ Pull only those columns and rows into your secure project space
✗ Don't copy the entire EHR dump to scratch "just in case"

GDPR in practice

You're approved to study whether biomarker X predicts type 2 diabetes in a hospital cohort. The data controller gives you a pseudonymized export.

Data protection The data needs to be secured and controlled at all stages.

✓ Choose a provider with adequate capabilities (jurisdiction, technical measures, DPA)
✗ Don't upload data to an AI provider, or share it within your group via insecure systems (ie e-mail)

Data minimization The export has 200 variables × 50,000 patients. Your study needs ~8 variables and the 2,000 patients meeting your criteria.

✓ Pull only those columns and rows into your secure project space
✗ Don't copy the entire EHR dump to scratch "just in case"

Purpose limitation Ethics approval and patient consent cover *diabetes research only*.

GDPR in practice

You're approved to study whether biomarker X predicts type 2 diabetes in a hospital cohort. The data controller gives you a pseudonymized export.

Data minimization The export has 200 variables × 50,000 patients. Your study needs ~8 variables and the 2,000 patients meeting your criteria.

✓ Pull only those columns and rows into your secure project space
✗ Don't copy the entire EHR dump to scratch "just in case"

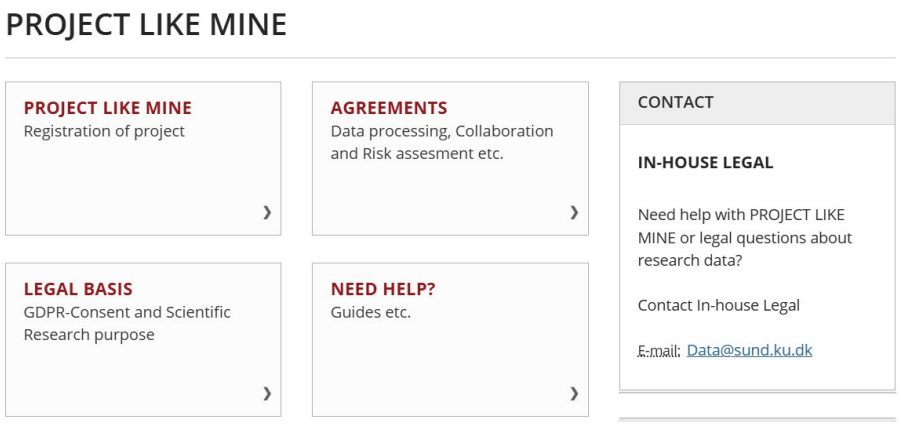
Data protection The data needs to be secured and controlled at all stages.

✓ Choose a provider with adequate capabilities (jurisdiction, technical measures, DPA)
✗ Don't upload data to an AI provider, or share it within your group via insecure systems (ie e-mail)

Purpose limitation Ethics approval and patient consent cover *diabetes research only*.

✓ Use the data for that question
✗ Don't reuse it to explore an unrelated trait, that needs a new legal basis

GDPR support



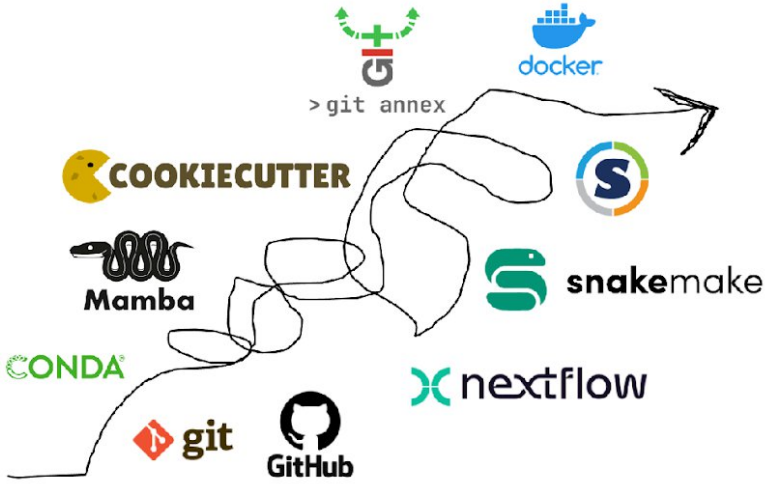
Project like mine

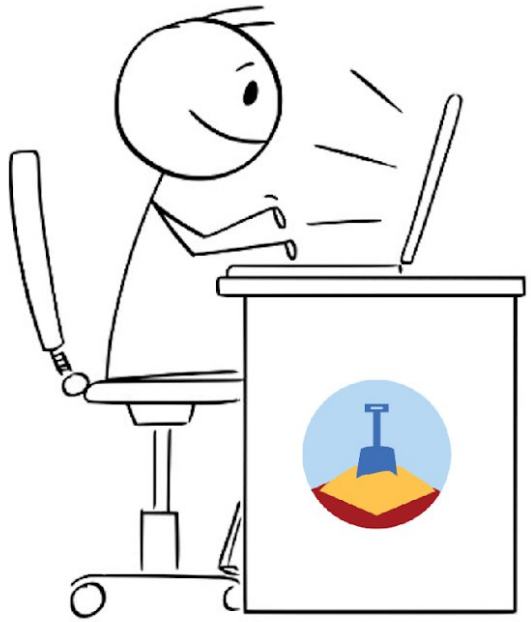
A digital platform from SUND. Mandatory for every project involving sensitive data. You are asked some questions and, in the end, you receive recommendations about which aspects to prioritize (DPAs, ethical approvals, etc.)

GDPR courses and consultation

Our colleague Diana Andrejeva from HeaDS offers a course called GDPR for Biomed, usually once a year, focused on legal aspects relevant to bioinformaticians. She also offers consultations for specific projects.

tools





Is there anything that can help us do RDM more efficiently?
In other sessions we will get more familiar with helpful tools and CLI commands

Danish HPC Providers

HPC alternatives

Maintained by SDU
Has CPUs and GPUs
User friendly interface
Adequate for sensitive data

GENOME | DK
HIGH-PERFORMANCE COMPUTING

From Aarhus University
Mostly CPUs
SSH and virtual desktops
GDPR compliant

LUMI

Finnish supercomputer
Mostly GPUs
SSH only, Slurm-based
Not for sensitive data

HPC alternatives

Supercomputer from NNF
(Gefion)
Mostly GPUs
Can’t use sensitive data

Operating out of DTU
CPUs and GPUs
Expensive and chaotic
GDPR compliant, needs DPA

GDPR compliance

Providers with a KU-wide DPA

Providers that can be used with custom DPA

computerome

GENOME | DK
HIGH-PERFORMANCE COMPUTING

Microsoft Azure

DCAI

aws

Any other cloud provider that complies with GDPR

Providers that cannot handle sensitive data

LUMI

GDPR compliance

Providers with a KU-wide DPA

Providers that can be used with custom DPA

GENOME | DK
HIGH-PERFORMANCE COMPUTING

Providers that cannot handle sensitive data

LUMI

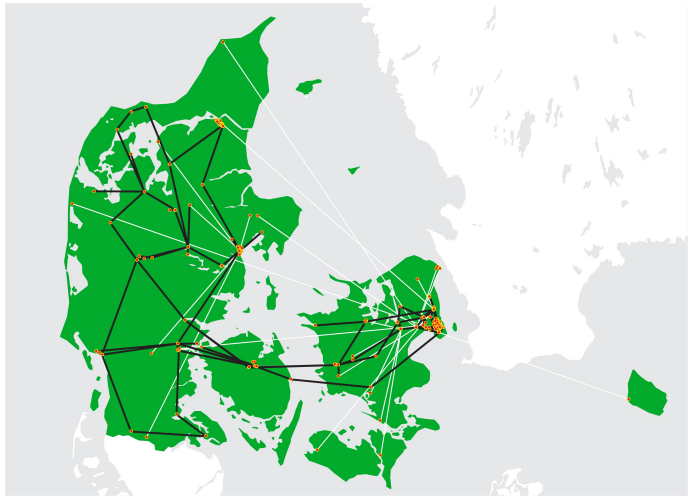
HPC alternatives

	UCloud	GenomeDK	Computerome	DCAI/Gefion	LUMI
CPU nodes	6592 vCPUs	52 thin/60 fat (~11k cores) (expanding)	692 thin/55 fat (50k cores)	191 nodes/112 cores each	2048 nodes/128 cores
GPU nodes	32 NVIDIA H100s, some older A100s and L40s	24 NVIDIA L40S (expanding)	40 NVIDIA V100s, some NVIDIA A100s	1528 NVIDIA H100s	2978/4 AMD MI250x
Storage	3 PB	33 PB	50 PB	~8 PB	~120 PB
Security	ISO 27001	ISAE 3000 + ISO 27001	ISAE 3000 + ISO 27001	NA	ISAE 3000 + ISO 27001
Sensitive data	yes, ‘at own risk’	yes, ‘closed zones’	yes, private clouds	not yet	not yet
Env mgmt	conda	Singularity, Apptainer	conda (& Docker?)	NA	Singularity, Apptainer
OS	UCloud GUI	AlmaLinux 8	CentOS 7	Red Hat Enterprise Linux (RHEL)	SLES 15 / CRAY

"free" HPC resources

An obvious consideration when choosing a HPC provider is cost. Luckily, for most of the major platforms, free resources are available through **DeiC**. Resource allocation from this pool is done by KU-IT.

You can get >10,000 CPU hours, >100 GB of storage and a generous amount of GPU hours for free with minimal overhead. If you write a more detailed proposal, those limits can increase 10x or more.



access to HPC resources

External

Via KU IT
(DeiC membership)

free

UCloud
LUMI
GENOME | DK
GDP R

Via Grants

free

DeiC
UCloud
LUMI
GENOME | DK
GDP R

EuroHPC
LUMI
+ 10 other EuroHPC systems

Via direct contract
(you pay)

paid

UCloud
GENOME | DK
Computerome2
Whoever you want to pay
GDP R

In-house

GDP R

paid

Rubus

GDP R

paid

Local systems

Hendrix,
BRIC cluster, Bio Cluster,
...

Slide from Henrike Zschach (KU-IT)

access to HPC resources

	UCloud Type 1	GenomeDK Type 2	LUMI Type 5	RUBUS local KU
User base	Users **unfamiliar** with HPC	Users familiar with HPC	Users **very** familiar with HPC	Users familiar with HPC
Interface	Custom graphical UI (can get terminal via UI)	Terminal + virtual desktop	Terminal + UI via On Demand (Jupyter, Matlab, VSCode)	Terminal (+ UI via On Demand in the future)
Connected to KU Network Storage	✗	✗	✗	✓
GPUs	NVIDIA	✗	AMD	NVIDIA
Cost	Basic amount free Extreme scale use via: • DeiC HPC grant • contract w UCloud	Basic amount free Extreme scale use via: • DeiC HPC grant • contract w GenomeDK	Basic amount free Extreme scale use via: • DeiC HPC grant • EuroHPC grant	Three payment models: • Pay as you go • Own hardware (integrated) • Own hardware (separate)
Personal Data	✓ DPA + ISO cert	(✗) ISO cert but no DPA - but you can make one!	✗ Not built for sensitive data	✓ same organization

Slide from Henrike Zschach (KU-IT)

access to HPC resources

UCloud
Type 1

log in at https://cloud.sdu.dk/app/dashboard

↓

create project or select existing

↓

Click ‘Apply for resources’

GenomeDK
Type 2

write to KU-IT-datalab@adm.ku.dk

↓

We create a project and invite you

LUMI
Type 5

write to KU-IT-datalab@adm.ku.dk

↓

We create a project and invite you

↓

You receive a LUMI username and project ID to use

RUBUS
local KU

Request slurm billing account via Service Portal ticket

↓

Add members via identity.ku.dk

↓

Work inside compute project (order via SP if you don’t have)

Own Server
local KU

Fill “IT solutions” ticket in Service Portal

↓

Meeting to discuss needs and options

Slide from Henrike Zschach (KU-IT)

access to HPC resources

Service Portal

Research IT > Research Applications (generic KU IT Datalab ticket)

Research IT > DeiC HPC resources (for HPC topics)

Research IT > IT solutions (DLF IT solutions ticket)

Email

KU-IT-datalab@adm.ku.dk

Slide from Henrike Zschach (KU-IT)

HPCs at KU

Some departments at KU have local servers or clusters that they regularly use. They are often fine-tuned to their needs and contain protocols and workflows you can reuse. They should be the first option if you are considering using HPCs.

If your center or department does not have a server or cluster, you may still be able to access the existing one via collaborations. Remember to consider whether the platform is compatible with handling **sensitive data**, if that is your use case.

Alternatively, you can build a server/cluster of your own. However, depending on the budget, this requires a very slow (12-24 months) process of security and financial evaluations.

Mjølnir

Novo Nordisk Foundation
CENTER FOR
BASIC
METABOLIC
RESEARCH

UNIVERSITY OF
COPENHAGEN

Department of Drug Design and Pharmacology

summary

There are many HPC resources

There are multiple options in case you need to use HPCs. A few of them are free (up to a certain limit), and requesting it is straightforward.

Consider your use case

It’s very important that you understand which resources you need. Processing sensitive data limits your options considerably.

Pick a provider that fits your expertise

If you are just a beginner, you might want to pick options with easier access (virtual desktop, GUI). If you are familiar with the command-line and ssh, you have a lot more freedom.

Remember your responsibilities

You or your PI are directly responsible for the data you store and use. This means that skipping steps like not setting up a DPA could have serious consequences in case of a data breach.

UCloud Setup

what is UCloud?

UCloud is a unified research environment built by SDU. It has several types of compute resources available (CPU, GPU, fat nodes, etc). More importantly, it has a well designed, custom User Interface that you can access directly through your web browser.

It packages complex data science pipelines and analytical tools into ready to launch applications. This makes advanced computing highly accessible to beginners.

SDU
University of Southern Denmark

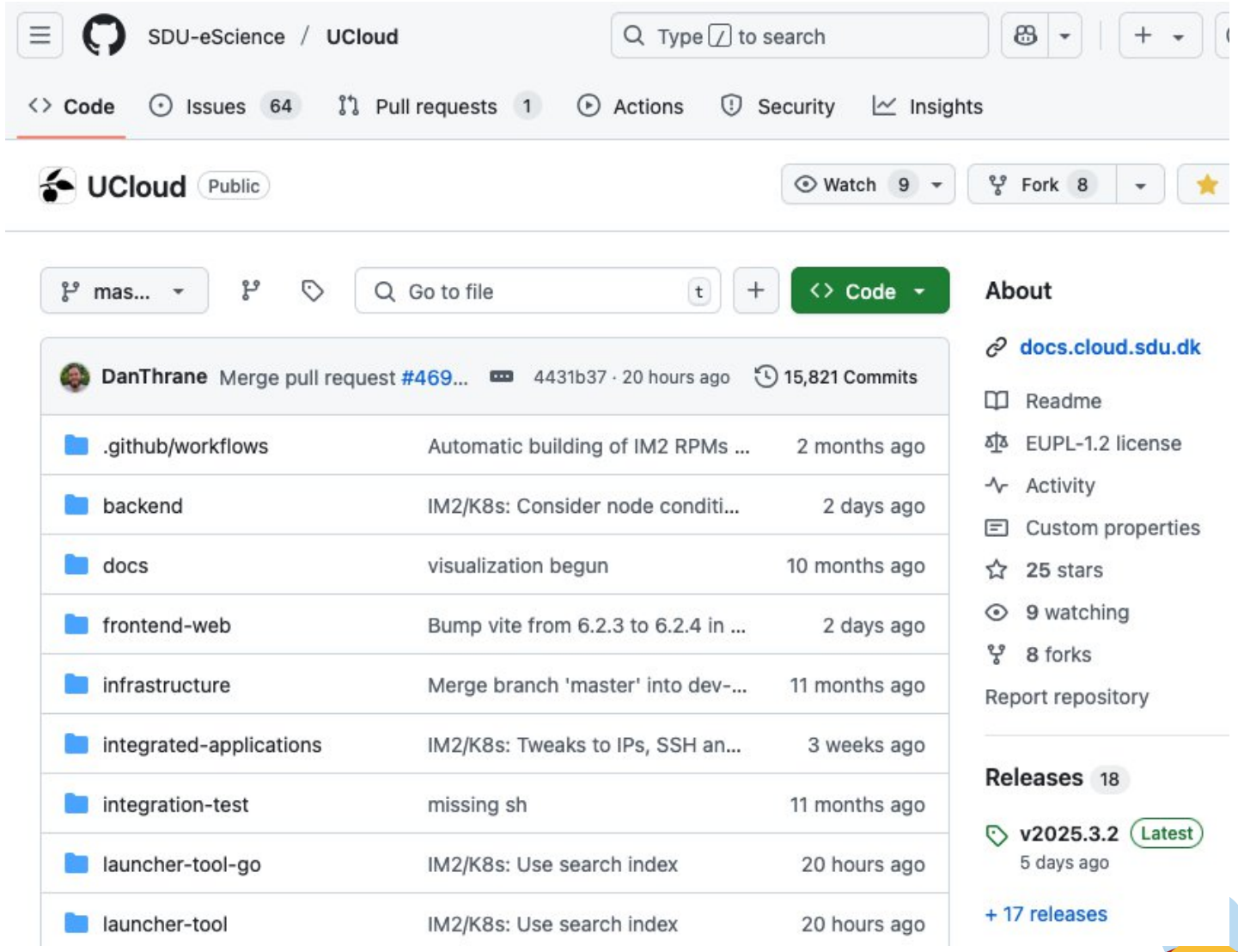
UCloud

UCloud vs regular HPC

UCloud is not a good example for the general experience of operating a HPC. It abstracts away some of the complexity involved in accessing HPC resources.

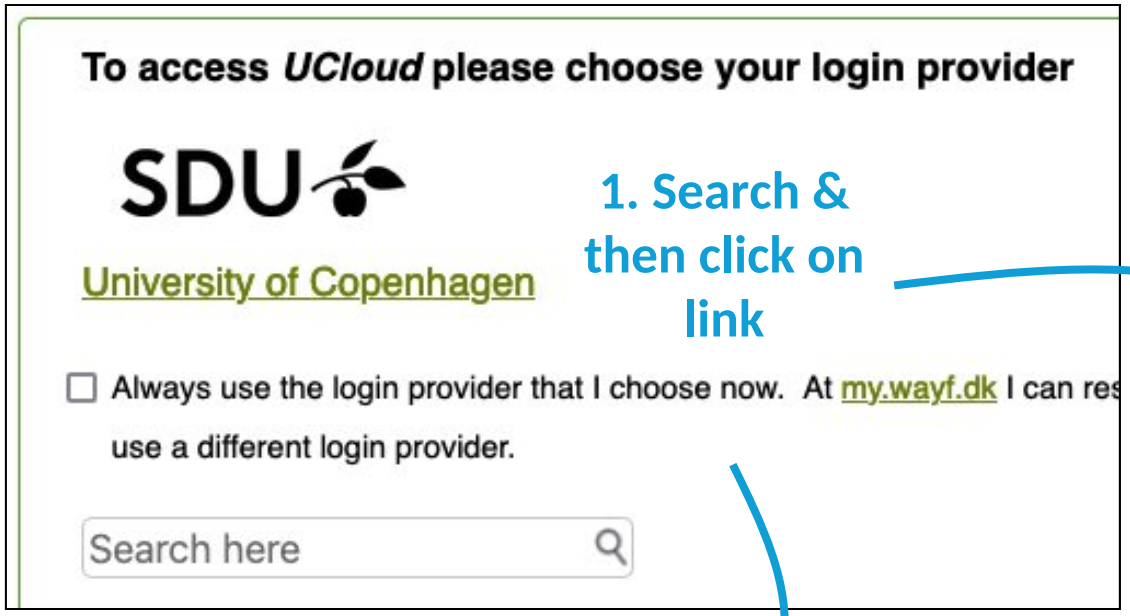
- Allows compute and storage resources from different providers to be accessed through the same interface (SDU/AU/AAU)

- GUI to help beginners, but can still support advanced work



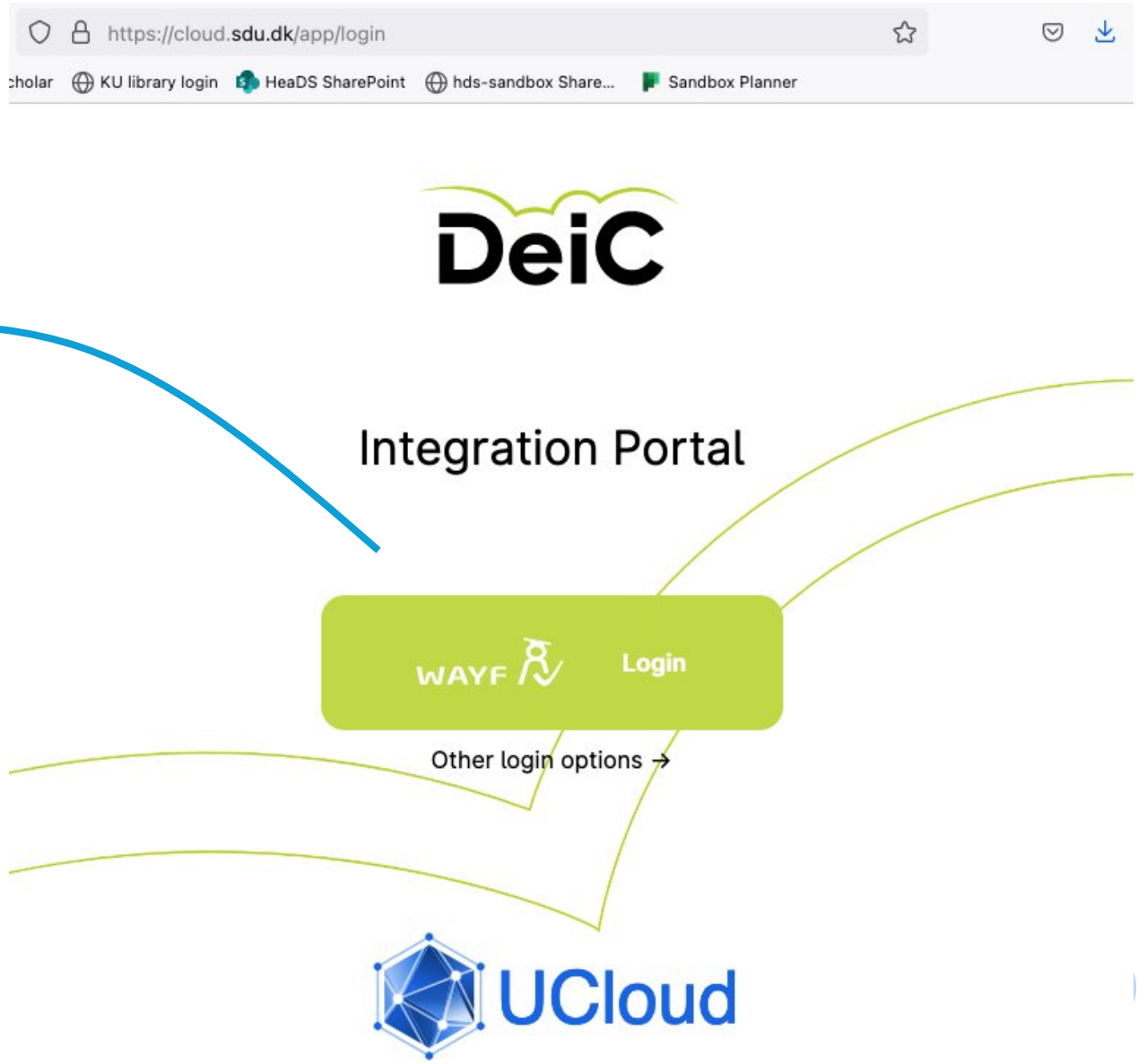
interactive HPC with UCloud

https://cloud.sdu.dk

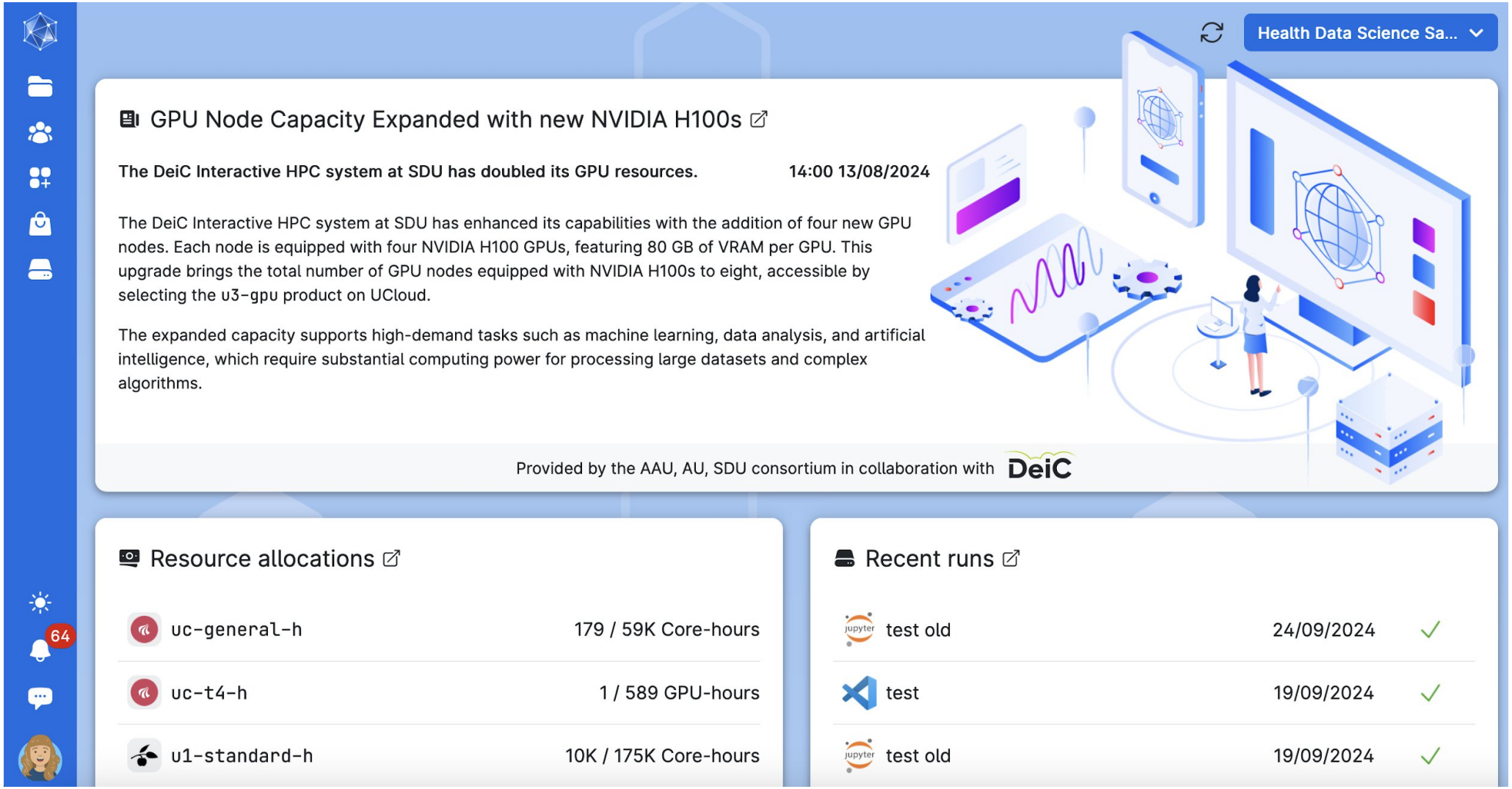


1. Search & then click on link

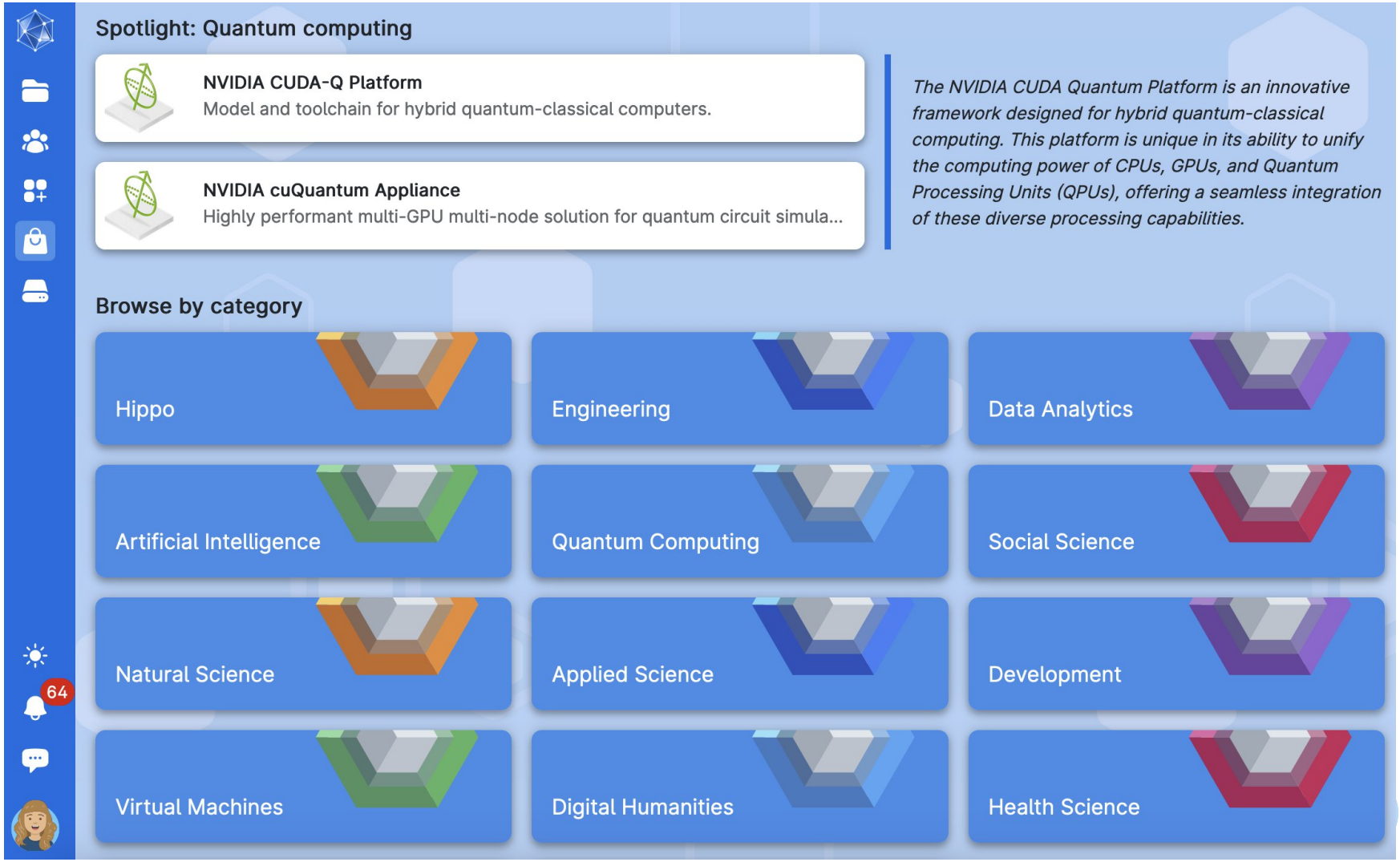
2. Sign-in via KU portal



interactive HPC with UCloud



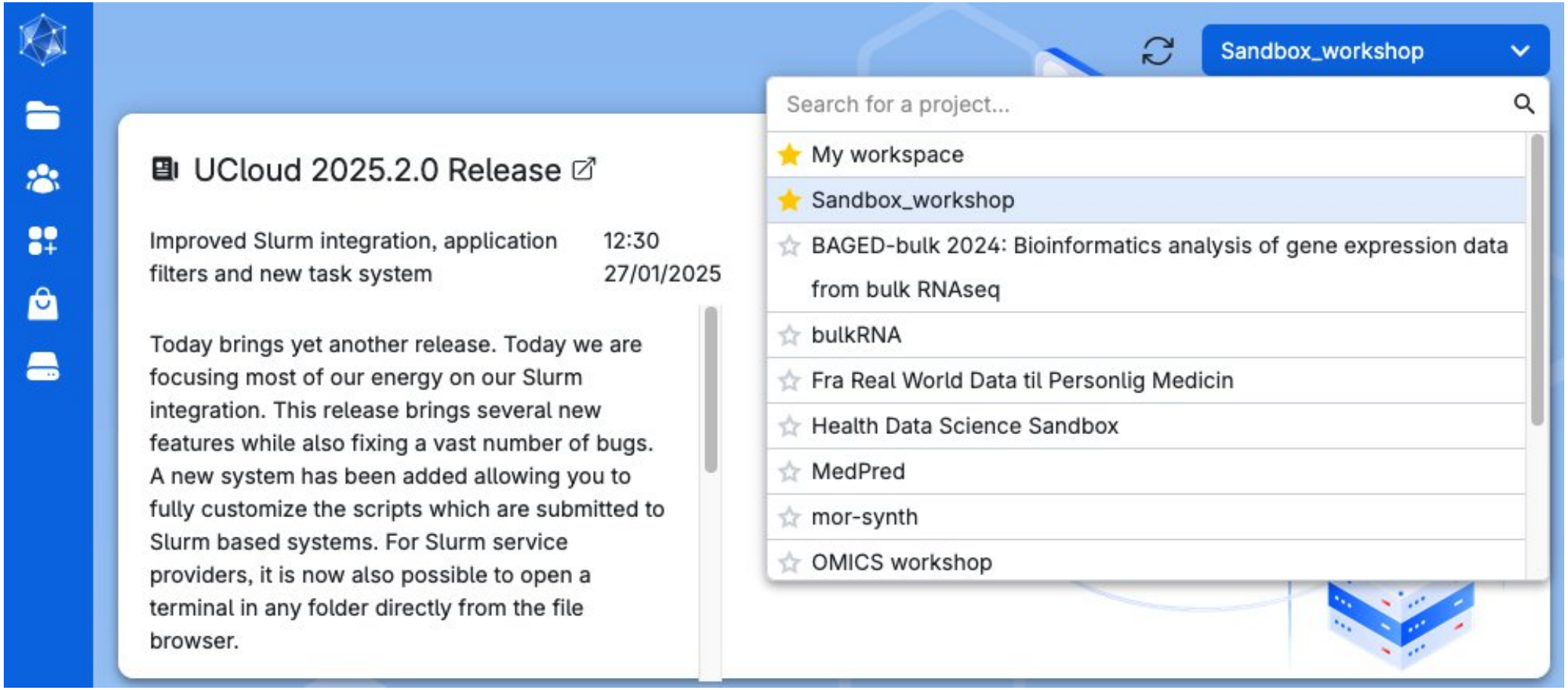
interactive HPC with UCloud



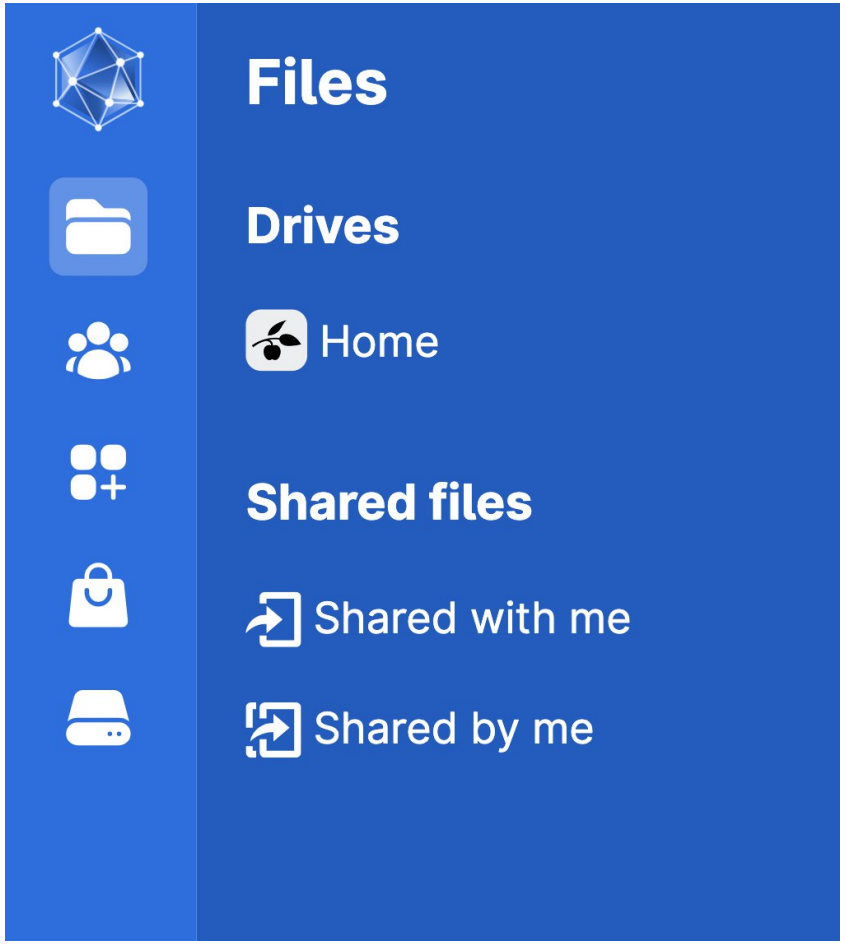
interactive HPC with UCloud

Apps can be starred for easy access

workspaces


Sandbox_workshop
Search for a project...
My workspace
Sandbox_workshop
BAGED-bulk 2024: Bioinformatics analysis of gene expression data from bulk RNAseq
bulkRNA
Fra Real World Data til Personlig Medicin
Health Data Science Sandbox
MedPred
mor-synth
OMICS workshop
UCloud 2025.2.0 Release
Improved Slurm integration, application filters and new task system
12:30
27/01/2025
Today brings yet another release. Today we are focusing most of our energy on our Slurm integration. This release brings several new features while also fixing a vast number of bugs. A new system has been added allowing you to fully customize the scripts which are submitted to Slurm based systems. For Slurm service providers, it is now also possible to open a terminal in any folder directly from the file browser.

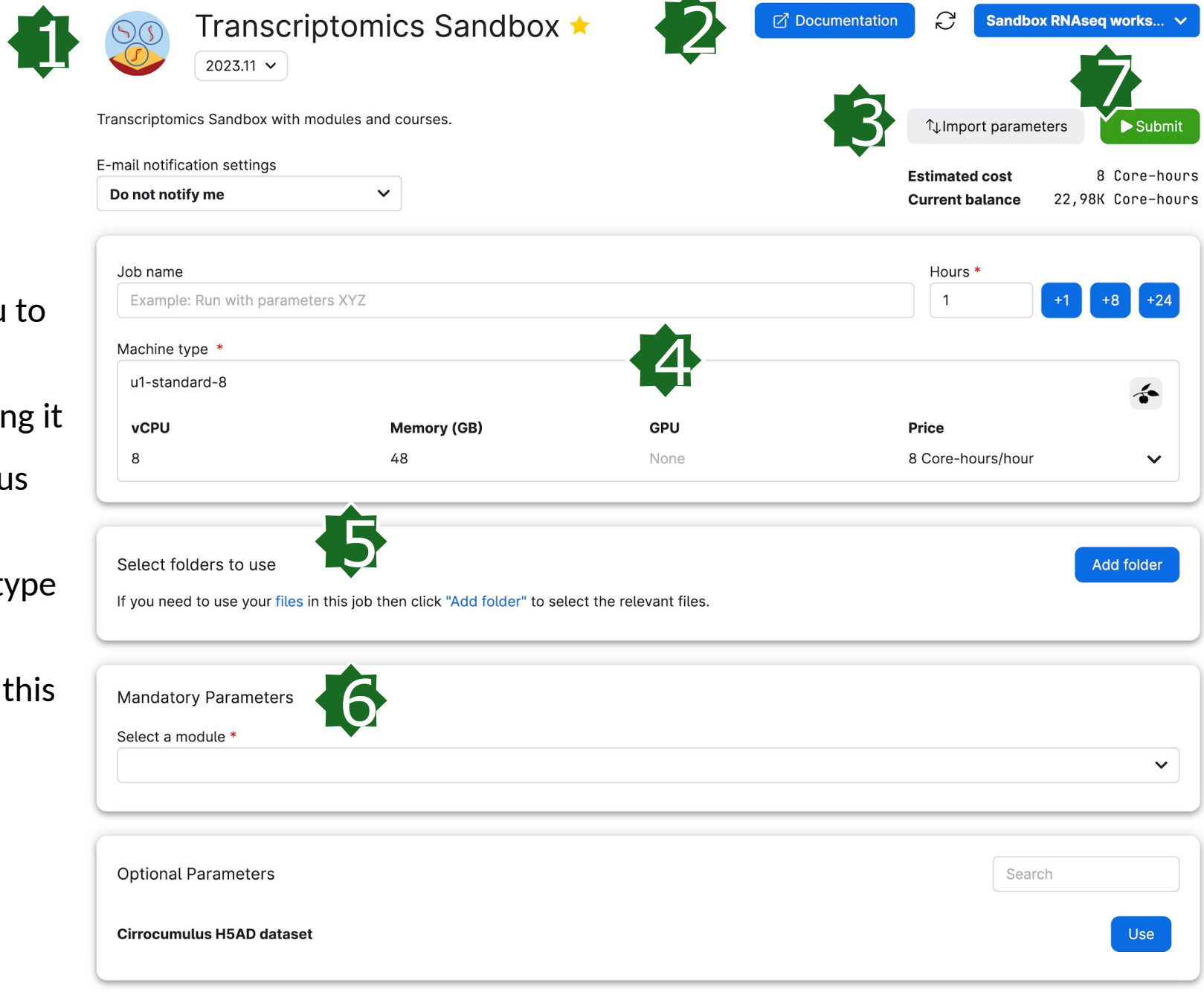
workspaces



Virtual workspaces allow you to share resources and work together with project collaborators

Project folders and files that only belong to the active workspace will be accessible from the menu at the left

UCloud demo



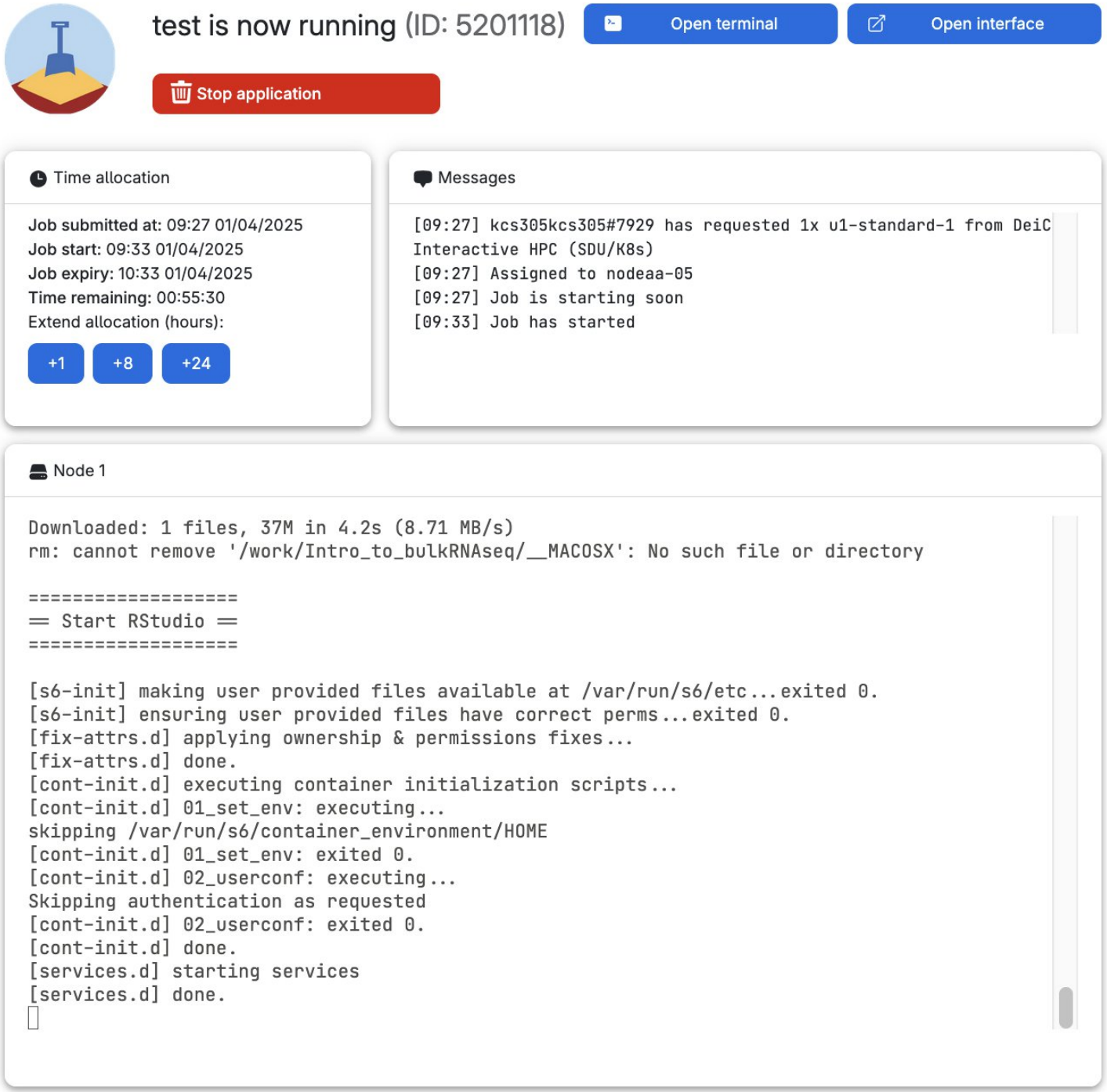
1. App & version (dropdown menu to change it)
2. Read documentation before using it
3. Import parameters from previous runs or JSON file
4. Job name, hours, and machine type (resources set-up)
5. Folders to access while running this particular job
6. Module to use (which includes Notebooks & Data)
7. Submit

UCloud demo

Choose ‘Open Interface’ once your job starts to get to your RStudio instance

Remember you can return to the job via the ‘Runs’ menu

- Check remaining time
- Top up time if you’re running out
- Stop job via ‘Stop application’ button to stop burning compute if you’re done

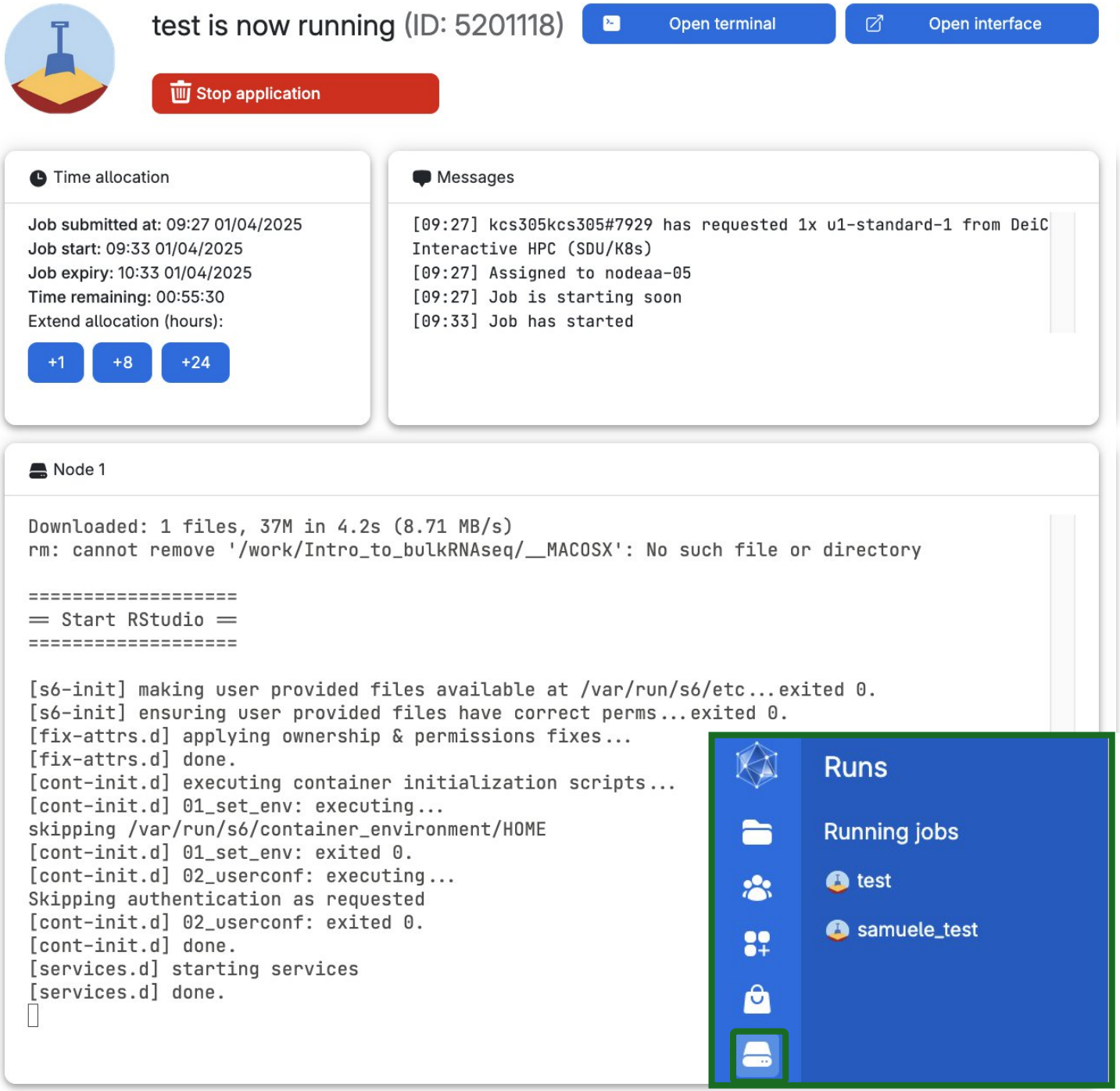


UCloud demo

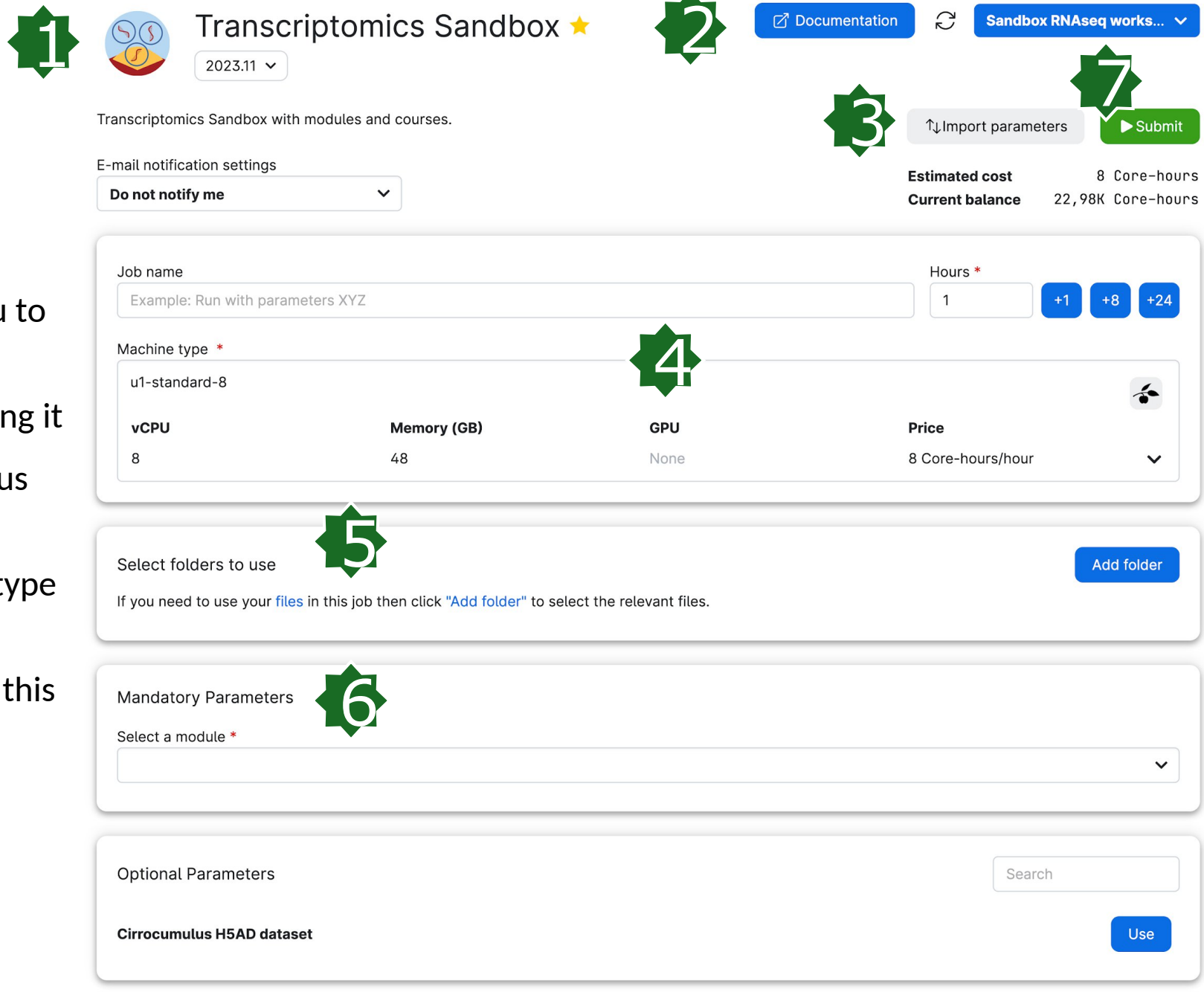
Choose ‘Open Interface’ once your job starts to get to your RStudio instance

Remember you can return to the job via the ‘Runs’ menu

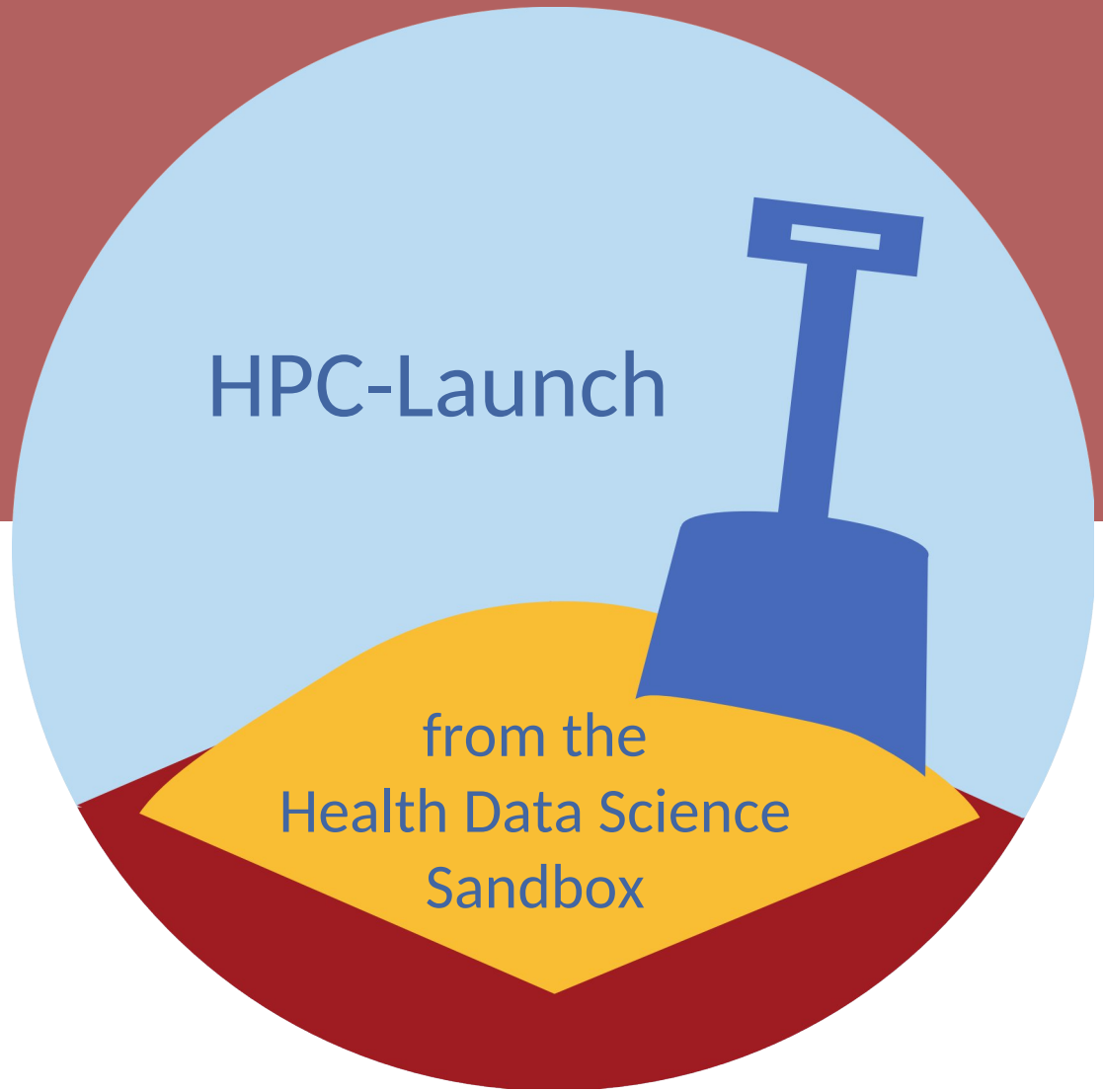
- Check remaining time
- Top up time if you’re running out
- Stop job via ‘Stop application’ button to stop burning compute if you’re done



UCloud demo



1. App & version (dropdown menu to change it)
2. Read documentation before using it
3. Import parameters from previous runs or JSON file
4. Job name, hours, and machine type (resources set-up)
5. Folders to access while running this particular job
6. Module to use (which includes Notebooks & Data)
7. Submit



Alba Refoyo Martinez, PhD
Center for Health Data Science (HeaDS)

Samuele Soraggi, PhD
Aarhus University

UNIVERSITY OF
COPENHAGEN

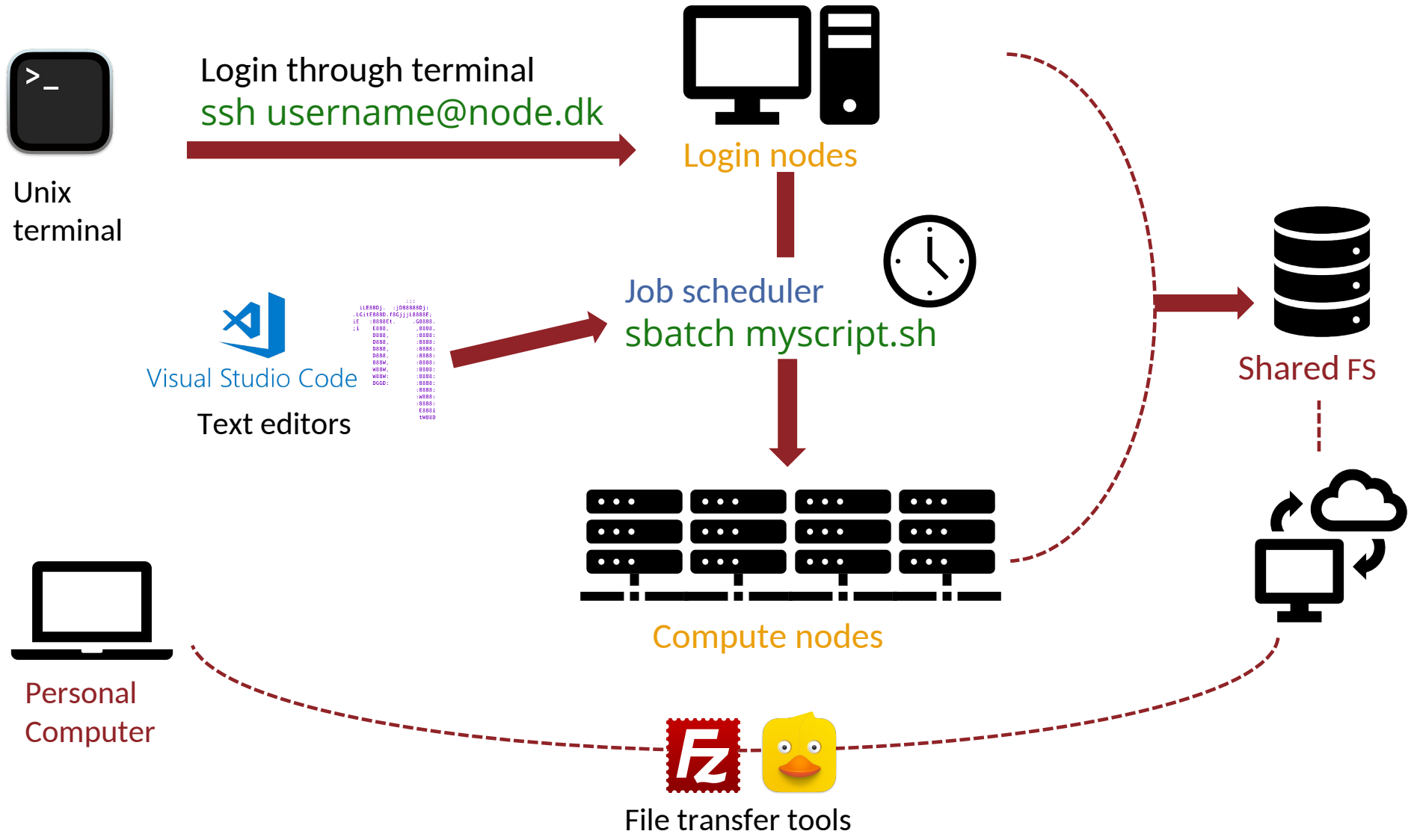
Content

☑ **Introduction to HPC systems**
- Understand HPC systems
- Assess high performance computing goals
- Secure compute and storage for your project

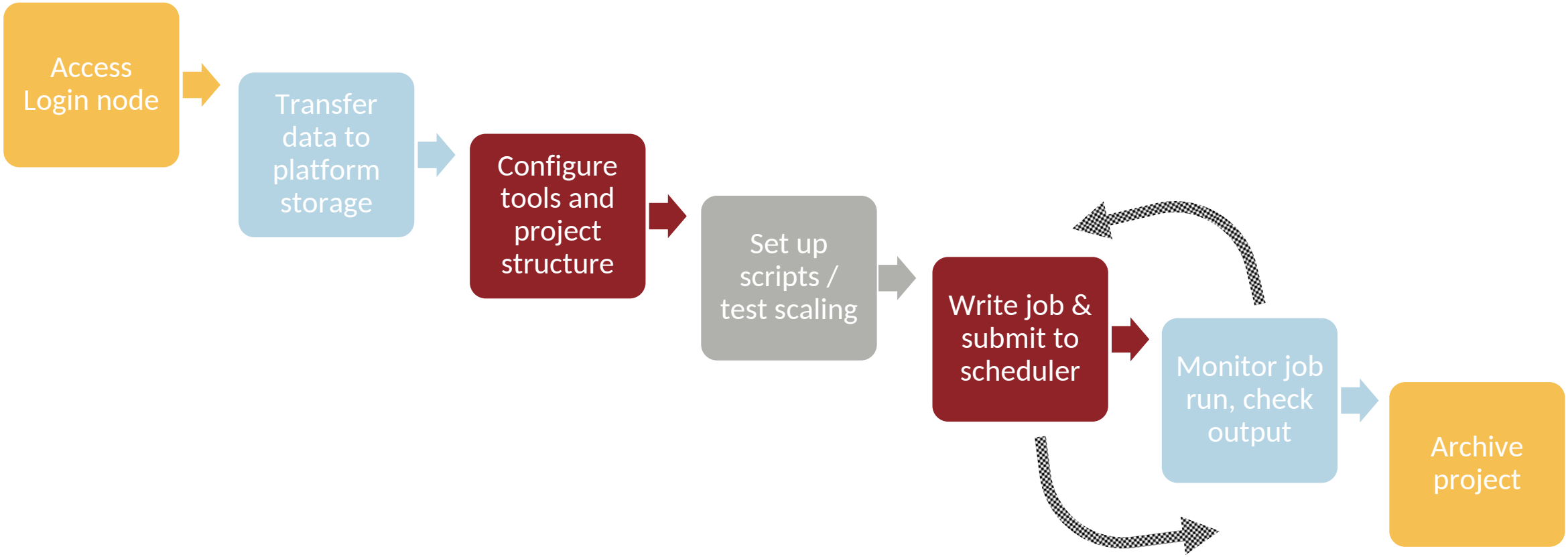
- **Set up computing environment essentials**
  - Access and authentication
  - Job scheduling basics
  - Data handling

- **Manage computational projects with a research data management (RDM) focus**
  - Data lifecycle
  - File system and storage structure
  - Reproducibility
  - Helpful tools to support RDM implementation

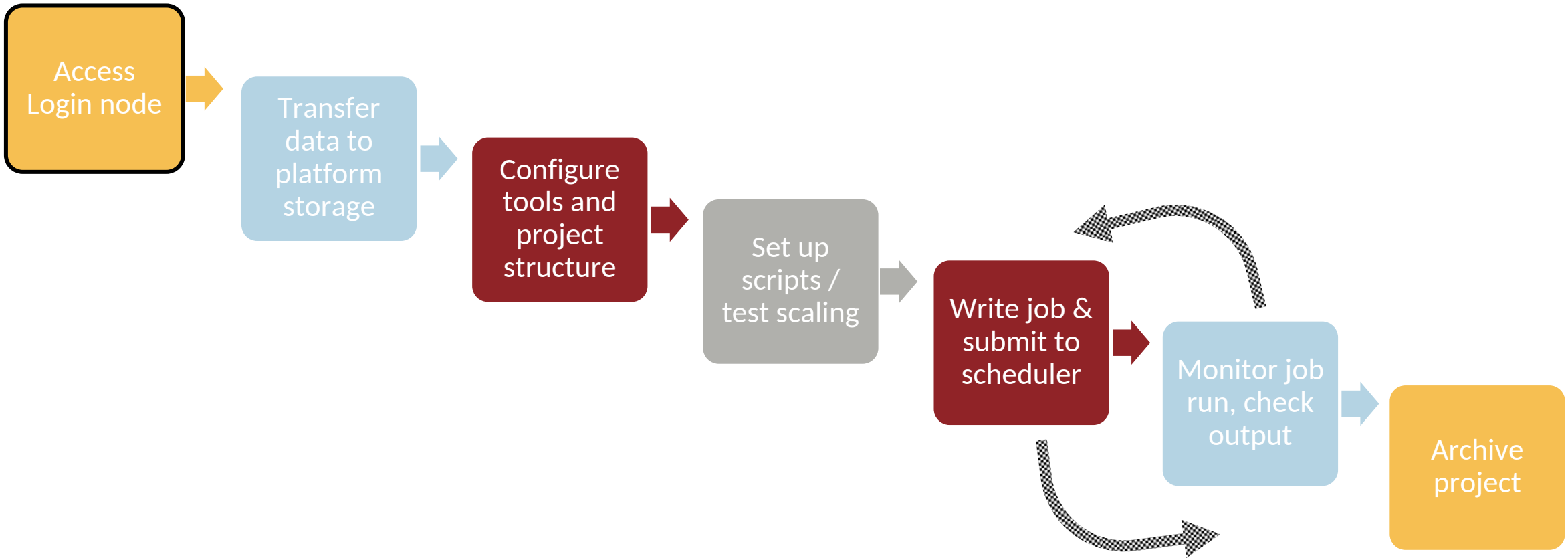
Overview: Working on remote servers



Standard HPC workflow



Standard HPC workflow



Accessing the HPC platform

Access to HPCs

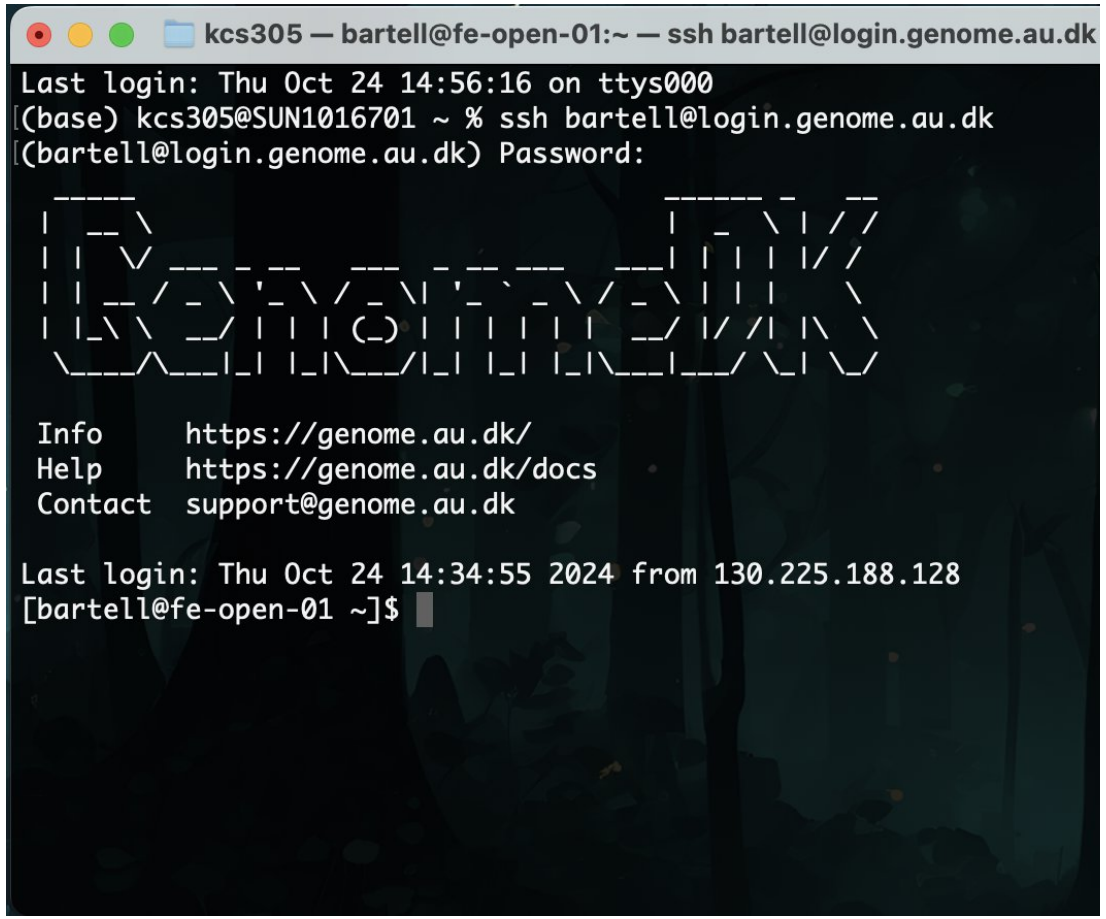
ssh, ssh keys & ssh agent

File system (FS)

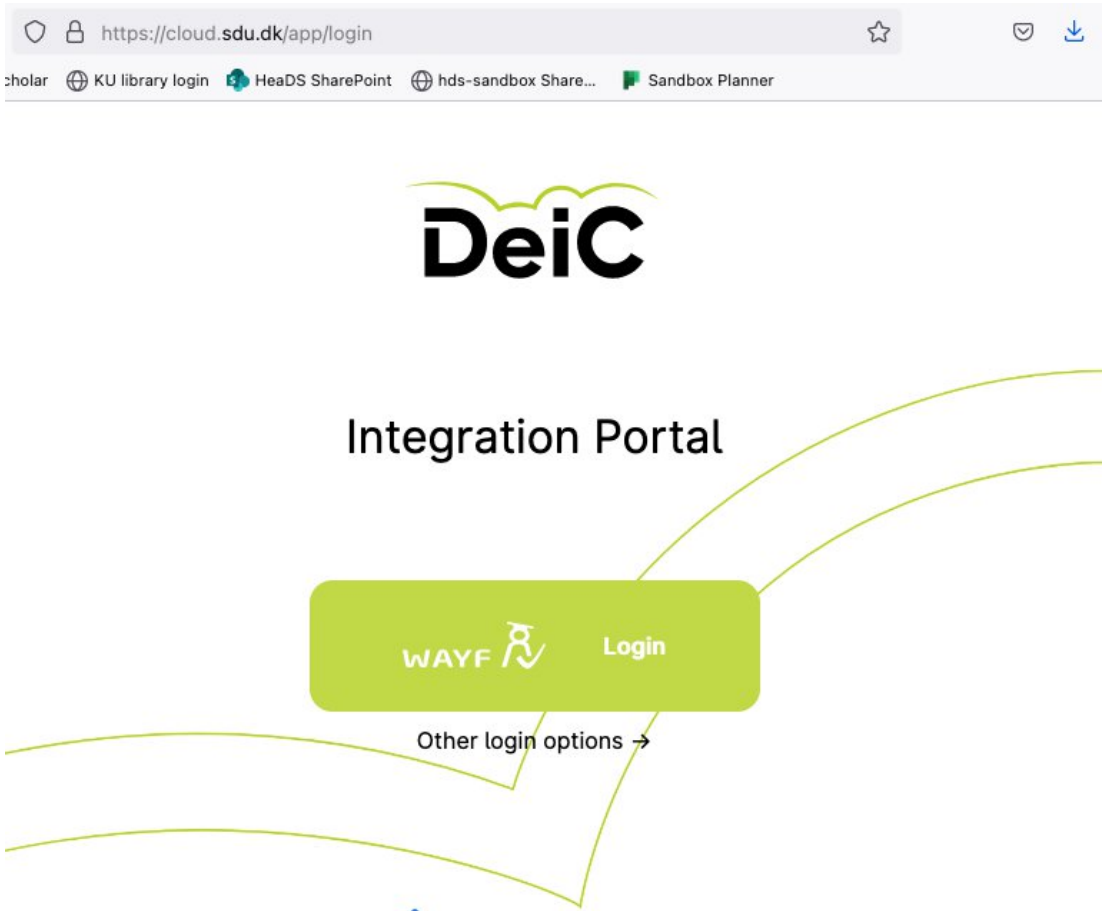
File editors and IDEs

Accessing the HPC platform

GENOME | DK
HIGH-PERFORMANCE COMPUTING



UCloud



ssh (Secure shell protocol)

Cryptographic network protocol for **remote login and command-line execution** using your terminal

Supported on **HPC** platforms (even GUI-based ones i.e. UCloud)

Commonly protected with a **2FA** (text or authenticator app)

Flexible communication

- Local to server
- server to local
- server to server!

```
> (base) ssh username@login.genome.dk
                         hostname
(username@login.genome.dk) 2FA token: 123456

> [username@fe-open-01 ~]$
```

Indicates you are on the remote server

SSH keys

ID verification via public/private keys: speed remote login and link to other services like GitHub

Multiple key algorithms

- **RSA** (Rivest–Shamir–Adleman, 1977) is a classic cryptography systems
- Algorithms evolve over time [-t ed25519].

Key mgmt best practices

- Use separate keys for servers, GitHub, etc.
- Protect each private key with a strong passphrase
- Store passphrases in a keychain or ssh-agent (with timeout or reset on restart)

Public keys live stored on the cluster in: .ssh/authorized_keys.

Config your ssh keys further: .ssh/config

name that describes the purpose

```
> ssh-keygen -t ed25519 -f ~/.ssh/id_UCloud –C user@sund.ku.dk


# to add a passphrase to an existing key
> ssh-keygen -p -f ~/.ssh/id_UCloud
```

.ssh directory must exist
(create otherwise)

-f: output_keyfile
-t: [-t ecdsa | ecdsa-sk | ed25519 | rsa]
-C user_host

SSH agent

- Default manager for ssh keys and passphrases
- Securely stores keys locally, avoiding repeated passphrase entry
- Supports agent forwarding (remote servers can use local SSH keys & passphrases stored on your local machine)

```
> eval ‘ssh-agent’ # shell inherits env variables and access to the agent

> ssh-add id_your_key
```

Why would you want to do this?

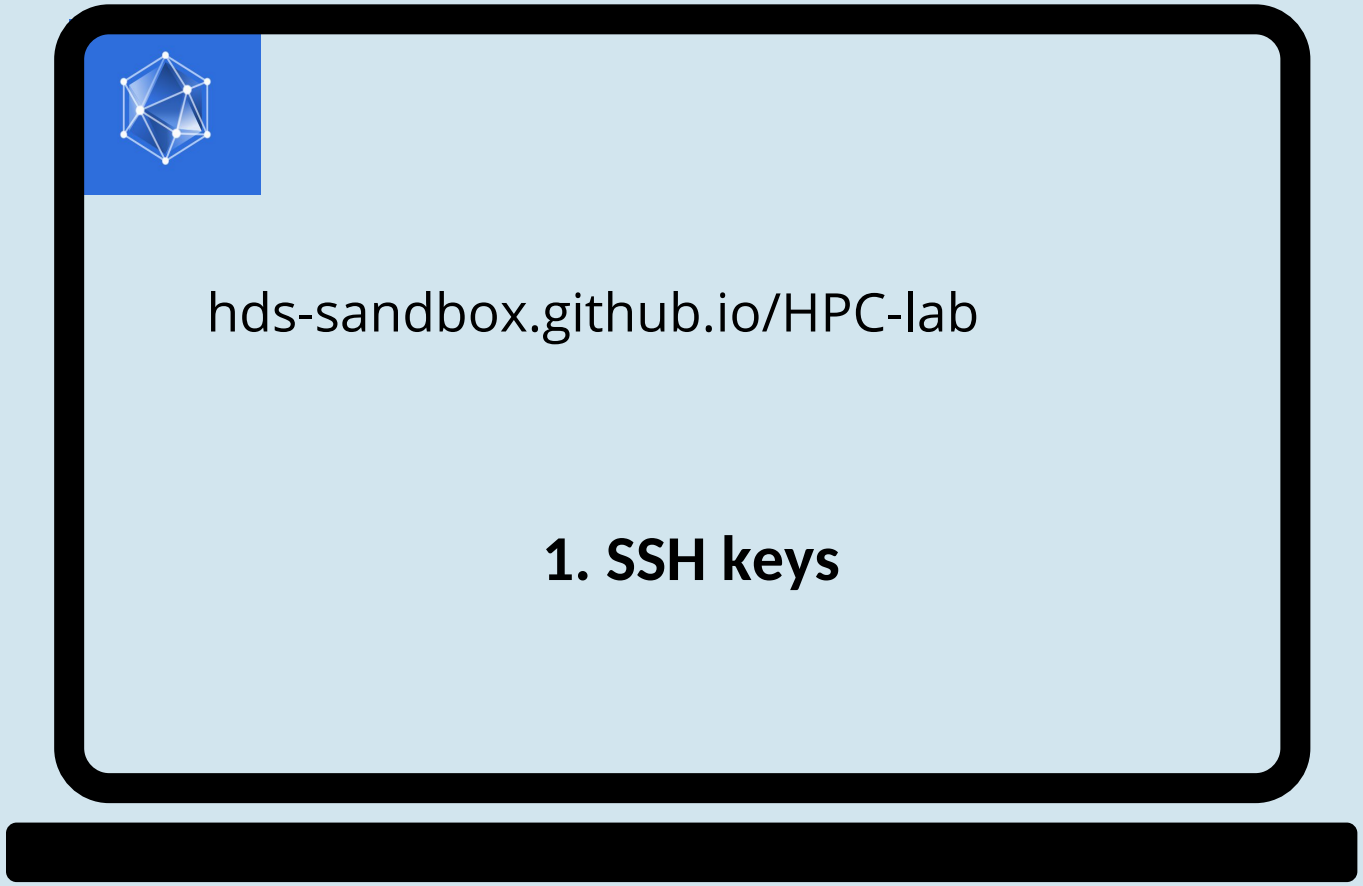
It’s the only way in on many servers!

It allows working in your local customized IDE (integrated development environment)

- Dev with secure files on server
- Dev with extra processors / RAM
- f.x. RStudio, Jupyter Lab, Visual Studio Code…

Visual Studio Code
- multi-language support, integrations like file explorer, git & terminal, many helper plug-ins!

Secure file access / transfer between local drive and server



~ 15 min

Now is YOUR time!

Day 1 > HPC setup

Eval your existing ssh keys and set one up for UCloud

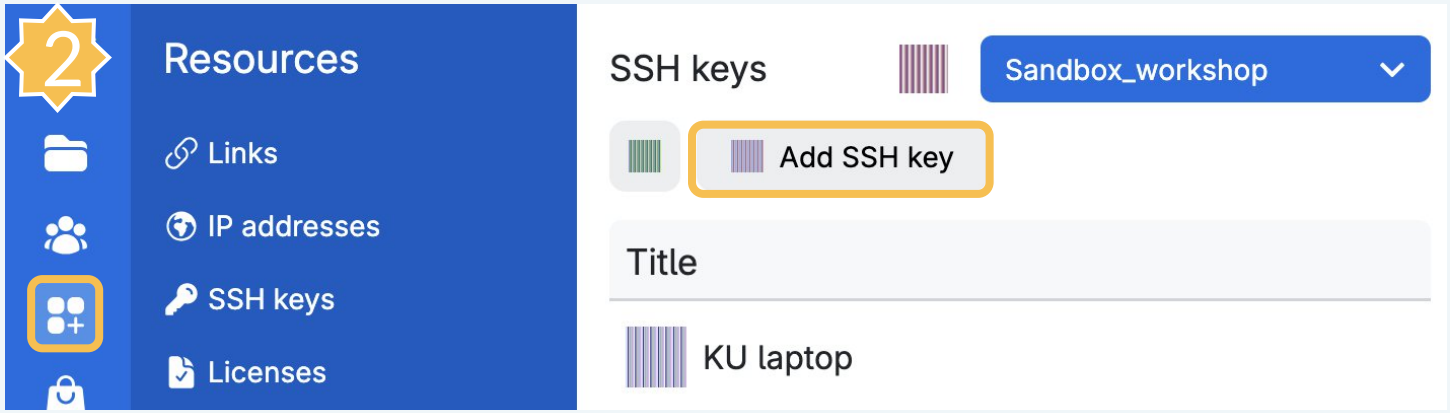
Start a job from the **terminal** app and then **ssh** in (remember to mount the folder you created!)

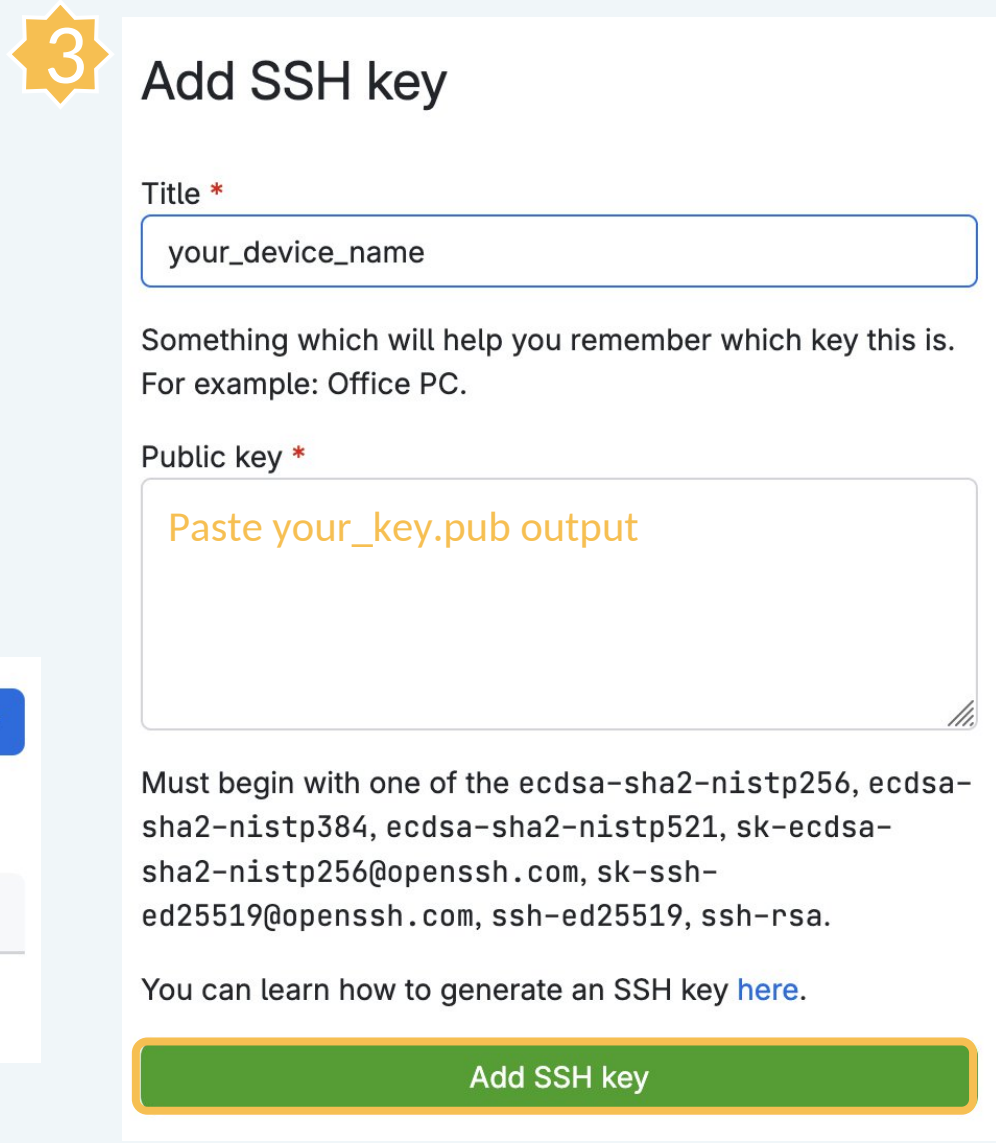
Remember: Add folder > create folder > **hpcLaunch**

Exercise: set ssh keys up for UCloud

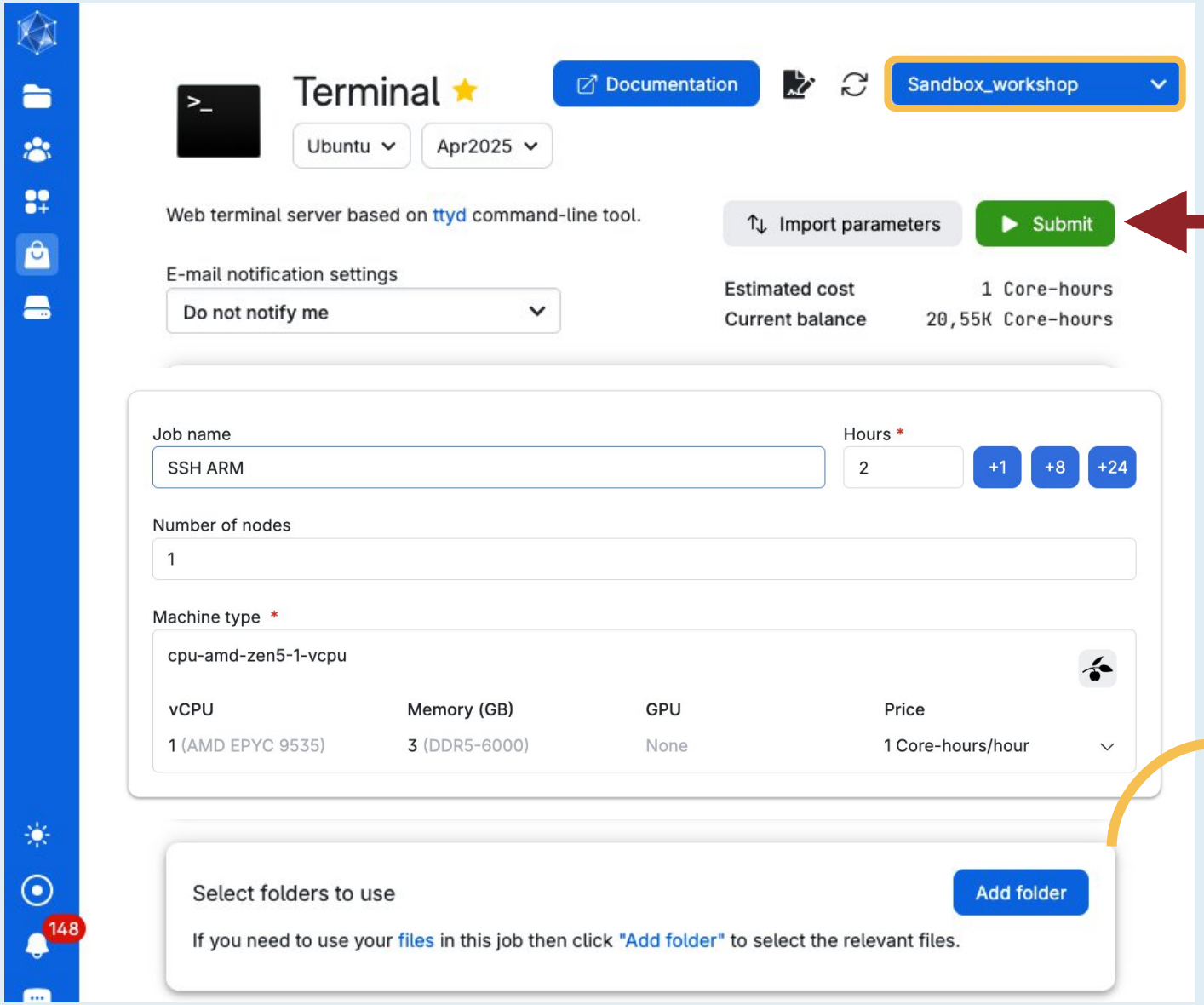
1 On your local device (Mac and Linux)

```
> ssh-keygen –t ed25519

> ssh-add your_key

> cat your_key.pub
```

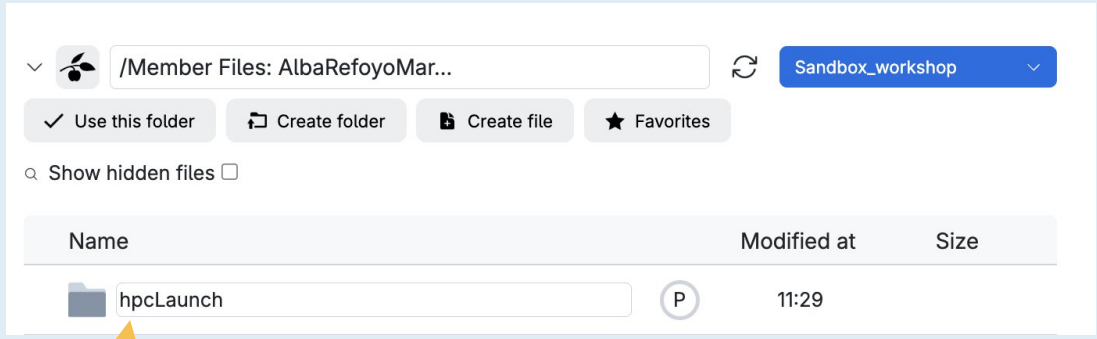


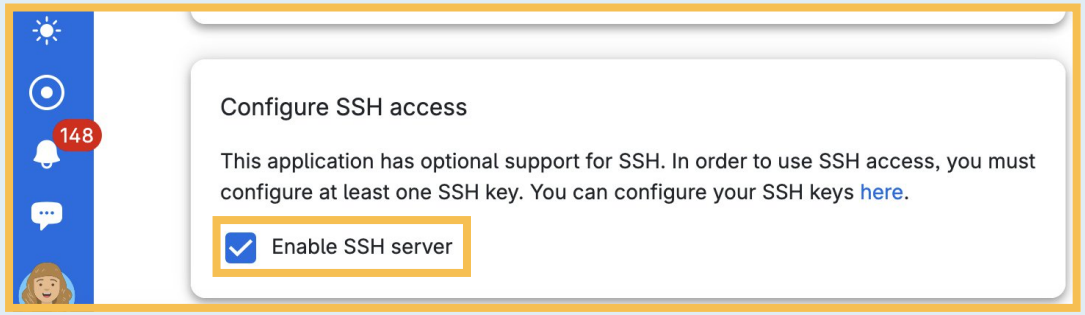


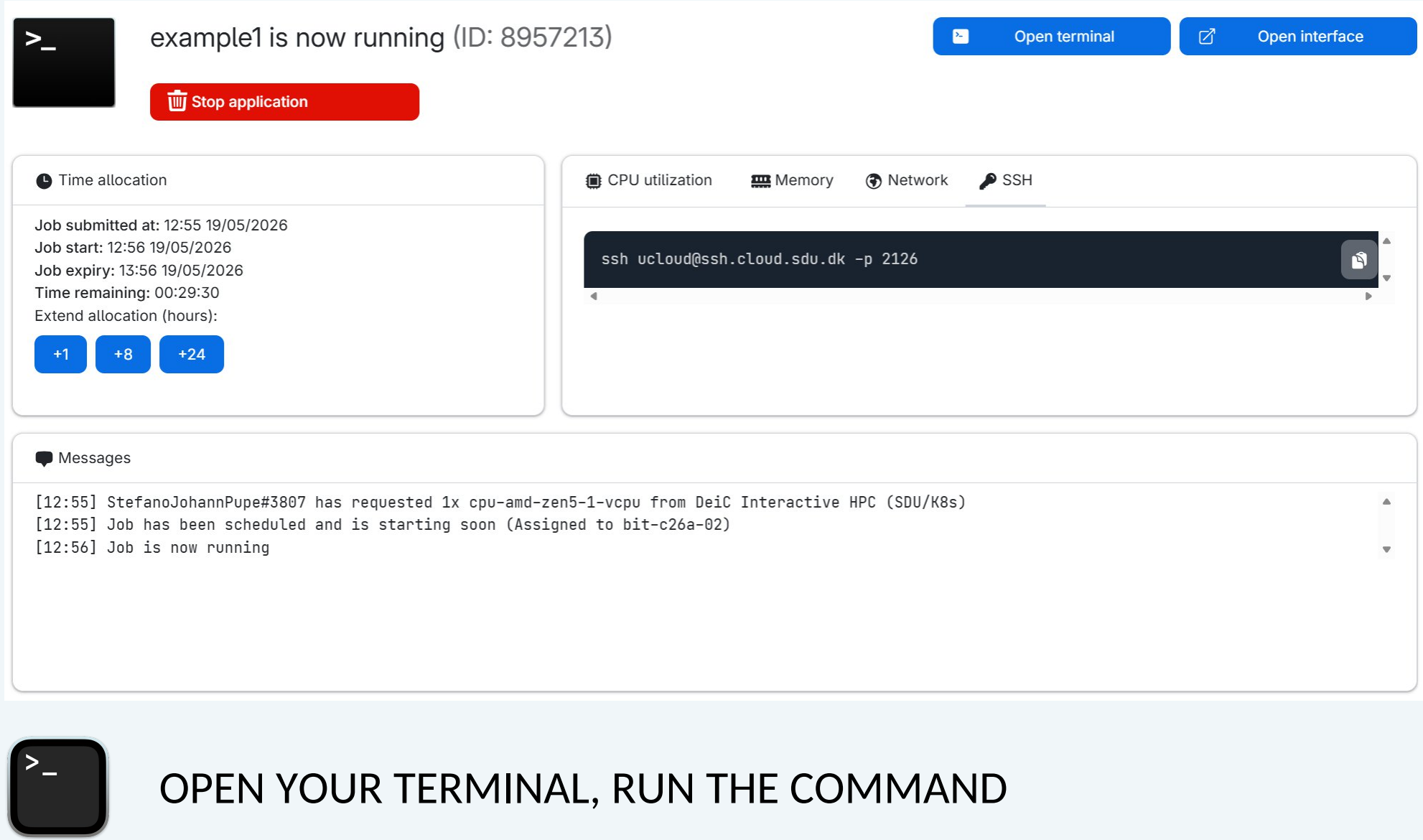
Exercise: Start a job from the terminal app and then ssh in



Start job w/ SSH server







OPEN YOUR TERMINAL, RUN THE COMMAND

Exercise: set one up for UCloud **Windows**

1. Start Windows PowerShell with Run as Administrator mode

```
> cd C:\Users\username\.ssh

> ssh-keygen

> Get-Service ssh-agent | Set-Service -StartupType Automatic

> Start-Service ssh-agent

> ssh-add your_key

> cat your_key.pub
```

Access – login node

Running your scripts on a login node is very tempting, but you must resist!

- **NEVER** run computational jobs directly on the login node. This will slow down all other users on the front-end
- **Monitor** resources
- Lightweight tasks only
  - Creating folder structure
  - Managing folders and files (do not perform large file transfers)
  - Small software installations (if allowed)
  - Submitting batch jobs
  - Editing code (*Interactive jobs*)

File system (FS)

Directory structure

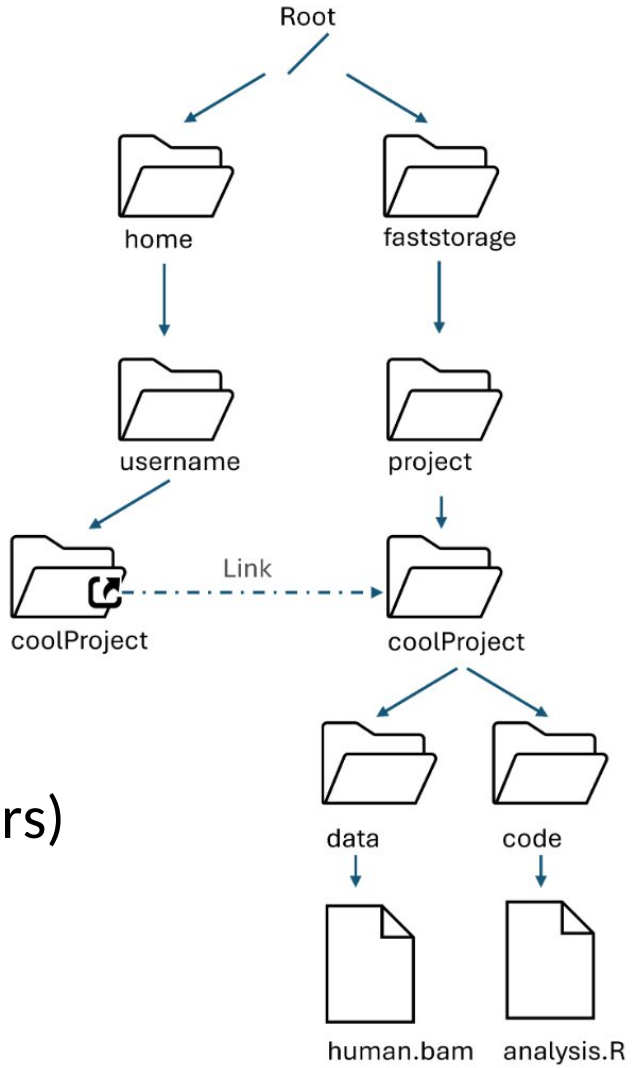
Absolute and relative paths

Navigation of the FS on the command line

FS – directory structure

- Folders and files follow a **hierarchy**
- **/ is the root** folder of the filesystem - nothing is above that
- The FS is shared across all machines and available to all users
- Commonly, there are two important folders in the root
  - home (or /home/username) and
  - faststorage (or shared, projects, etc.)
- /tmp or /scratch

Example FS GenomeDK

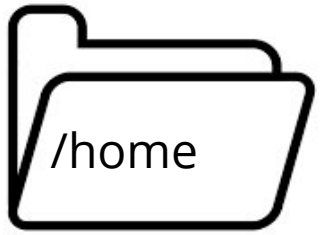


You can link projects to your home (shortcuts pointing to file/folders)

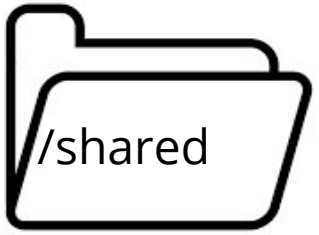
```
> ln -s /fastorage/project/coolProject
/home/myusername/coolProject
```

On UCloud, we mount the data

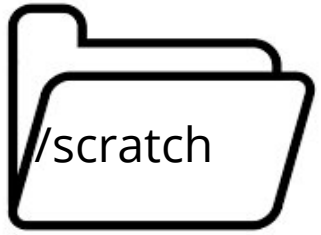
FS – directory structure



- Small
- Usually not backed-up and private to you
- Use for software and general scripts



- Large
- Backed-up!
- Can be accessed by all members of the project
- Data



- Large (1TB +)
- Not backed-up
- Data is deleted whenever the job in question finishes
- Use as “working directory” for processing data (e.g.: great for intermediate files or output files written in small chucks)

FS – directory structure

In general , do not fill up your home folder with data (or any large file)

Usually a limited amount of storage (less than shared or something equivalent).

Remember /home is private to you.

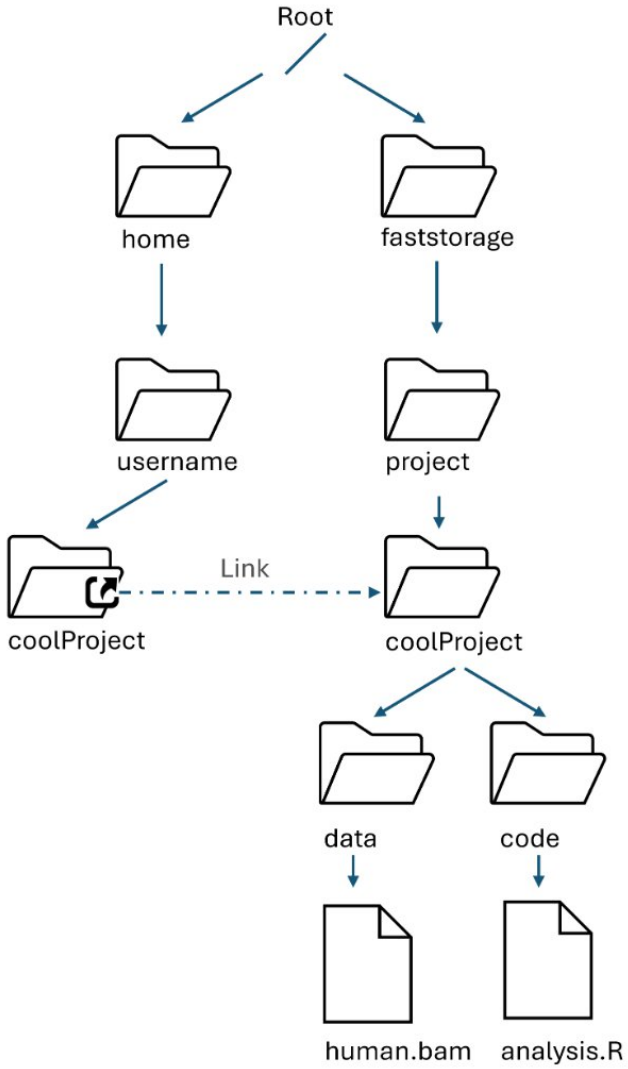
Command-line:

```
> pwd # print path of working directory (WD)
/home/username
```

Absolute path

Not the case on UCloud!

File system – absolute/relative paths



Remember: You can write relative paths from your WD (working directory)

Example

If your WD is coolProject

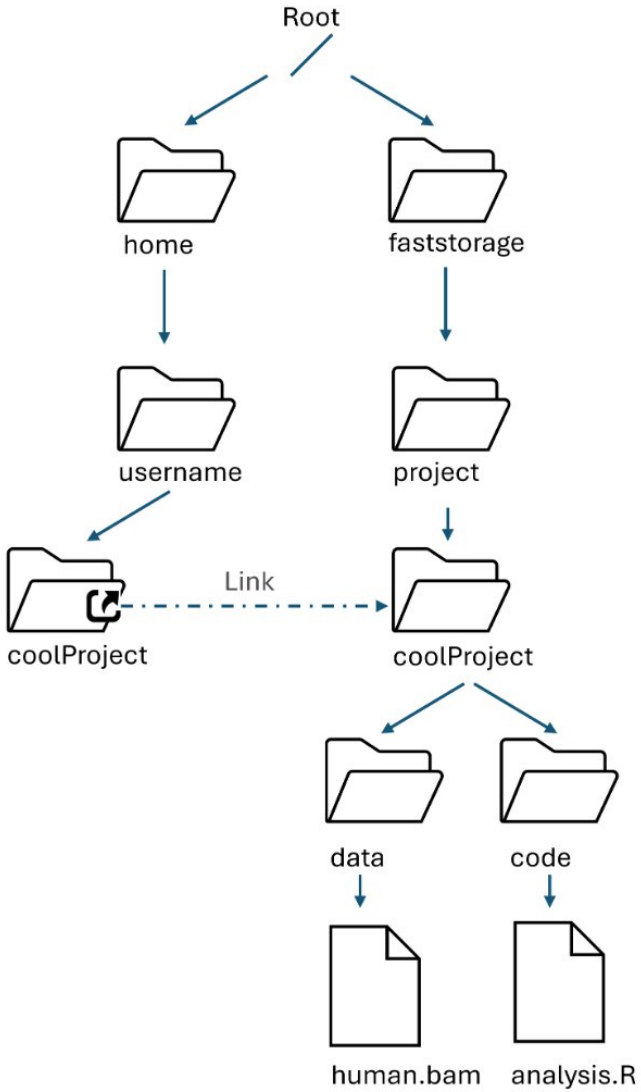
ls ./data

ls .. # see content in project

File system – absolute/relative paths

Which of the following statements are correct?

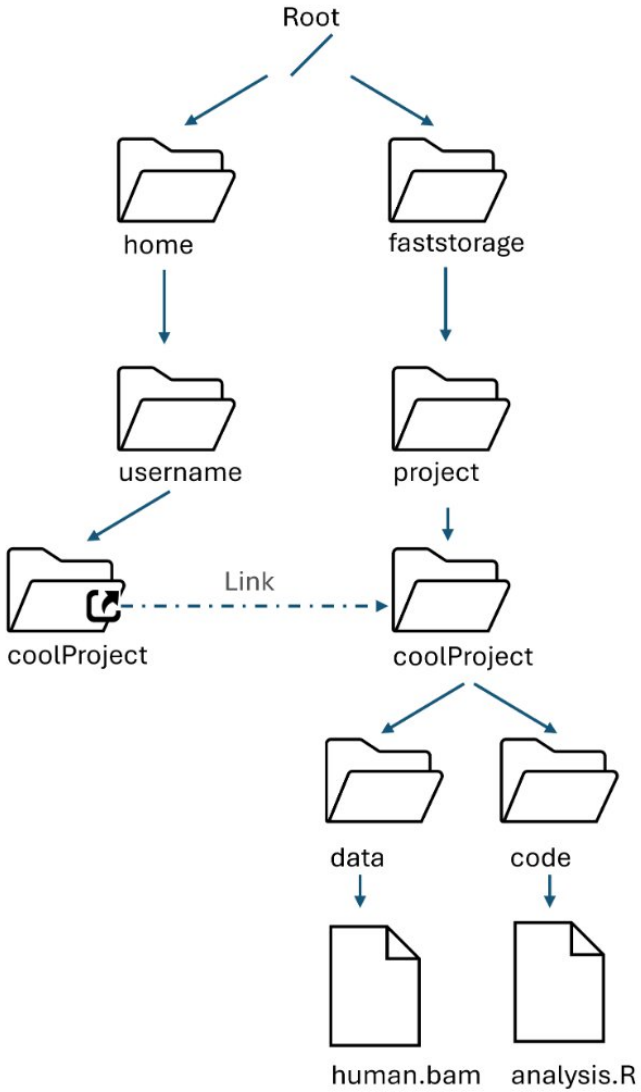
- Once a path goes through /home, it cannot go into /faststorage without going back along the file system tree
- /home/username/coolProject can be accessed by all users involved in the project coolProject
- There are two copies of the file human.bam and analysis.R in the file system
- /faststorage/project/coolProject is inside the storage nodes and should be used to save data



File system – absolute/relative paths

Which paths are correct in the file system in the fig.?

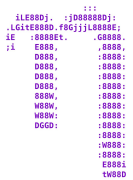
- /
- /home/username/coolProject
- /home/project
- /project/coolProject/data/human.bam



Files editors

VS Code

- GUI-based software
- Has the capability to connect to remote servers using the ssh protocol
- Many plug-ins



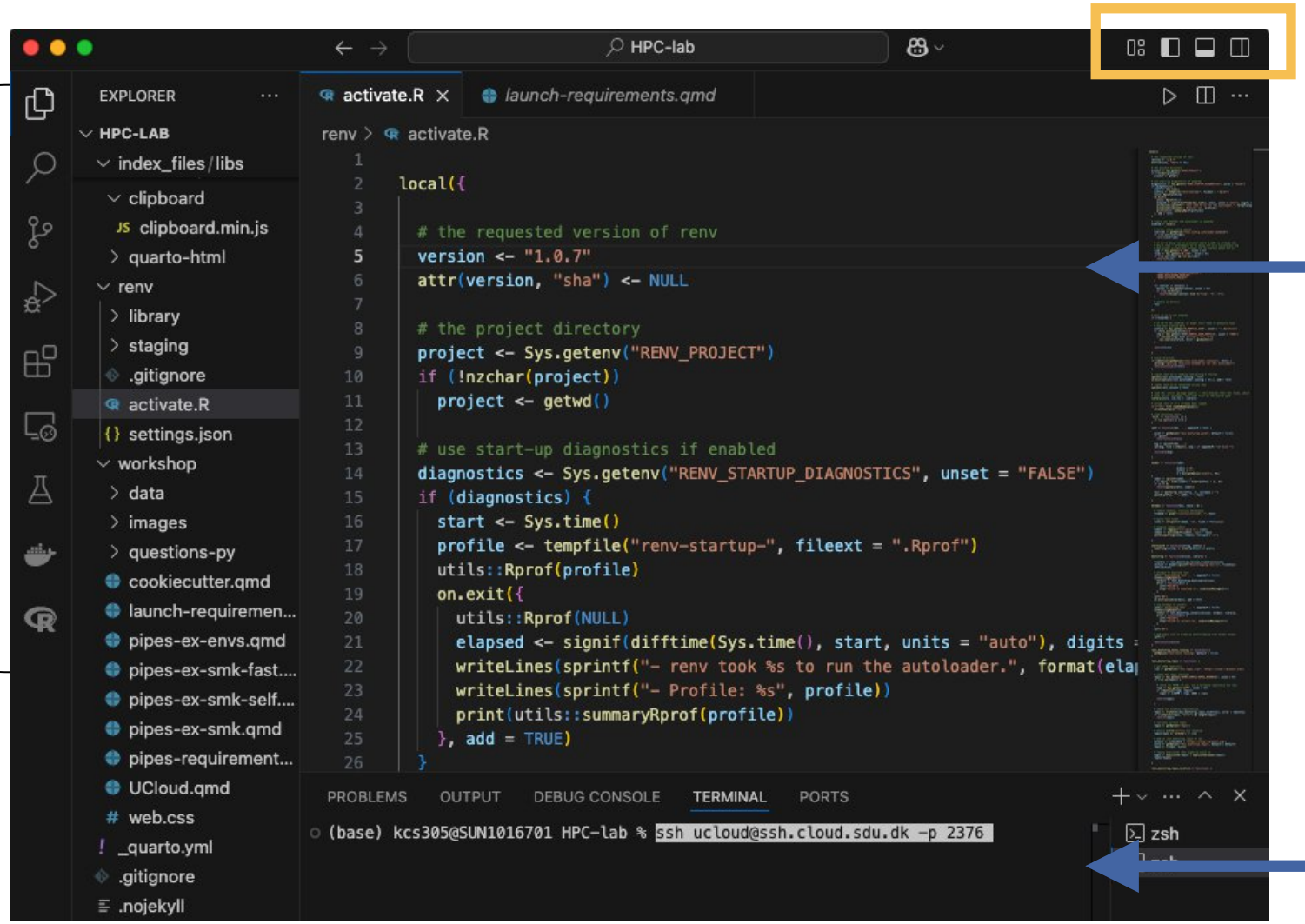
nano

- Command-line text editor available on most Linux distributions
- Ctrl + X to exit nano
  - It will ask if you want to save the file → type Y (yes)
  - It will ask to confirm the file name → press Enter

vim

- Command-line text editor available on most Linux distributions
- Very advanced (but steep learning curve)
- If you know what this is, you don’t need us to tell you about text editors

Integrated development environment (IDE)

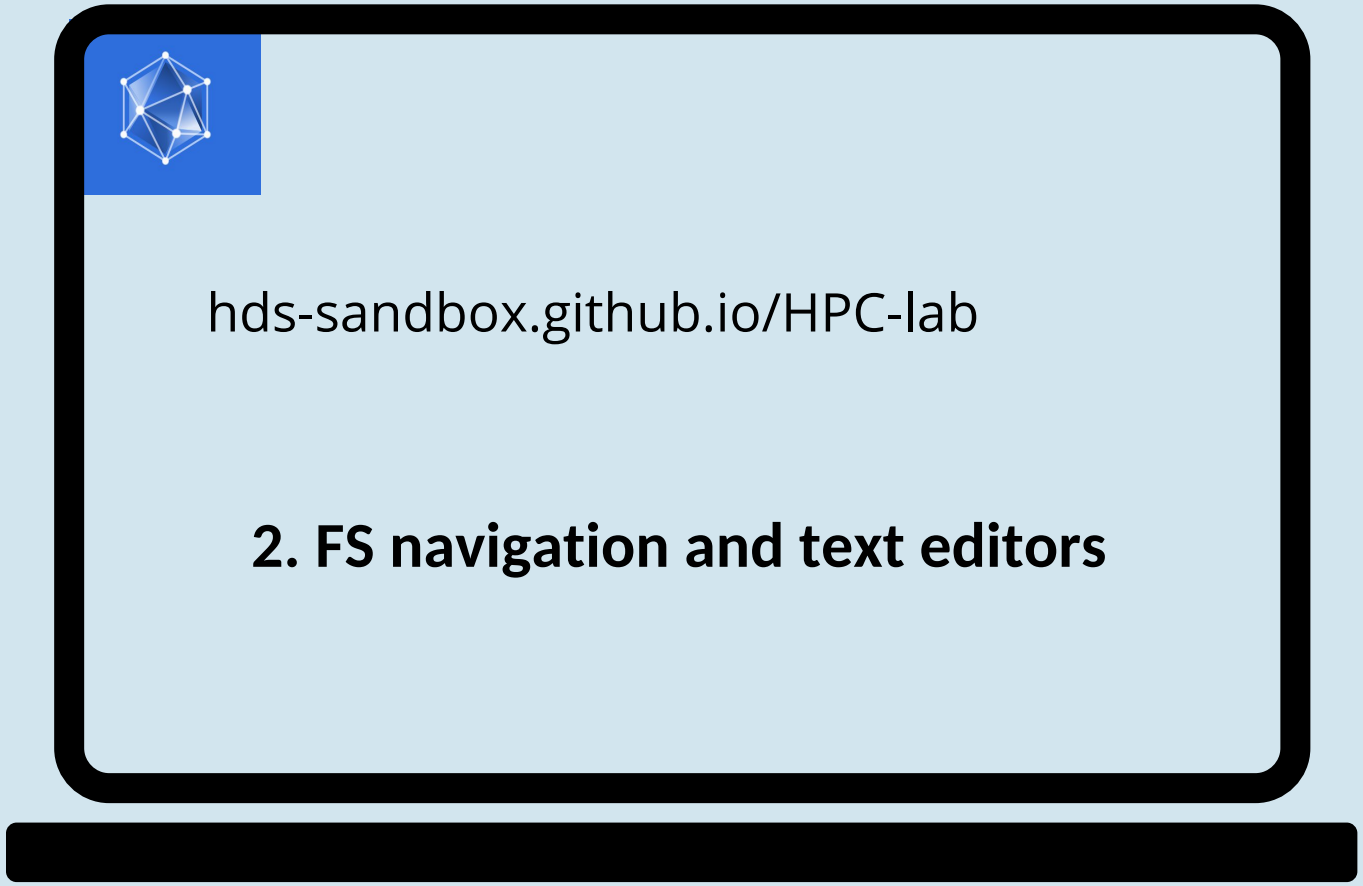


Side bar

Editor

Terminal

FS



~ 15 min

Now is YOUR time!

Day 1 > HPC setup

Working with files on HPC (FS, nano/vim)

Basic command-line familiarity

Use UCloud interface or SSH session

Stop the job when you are done!

Comments/questions

- ssh command - note **–p** for node port

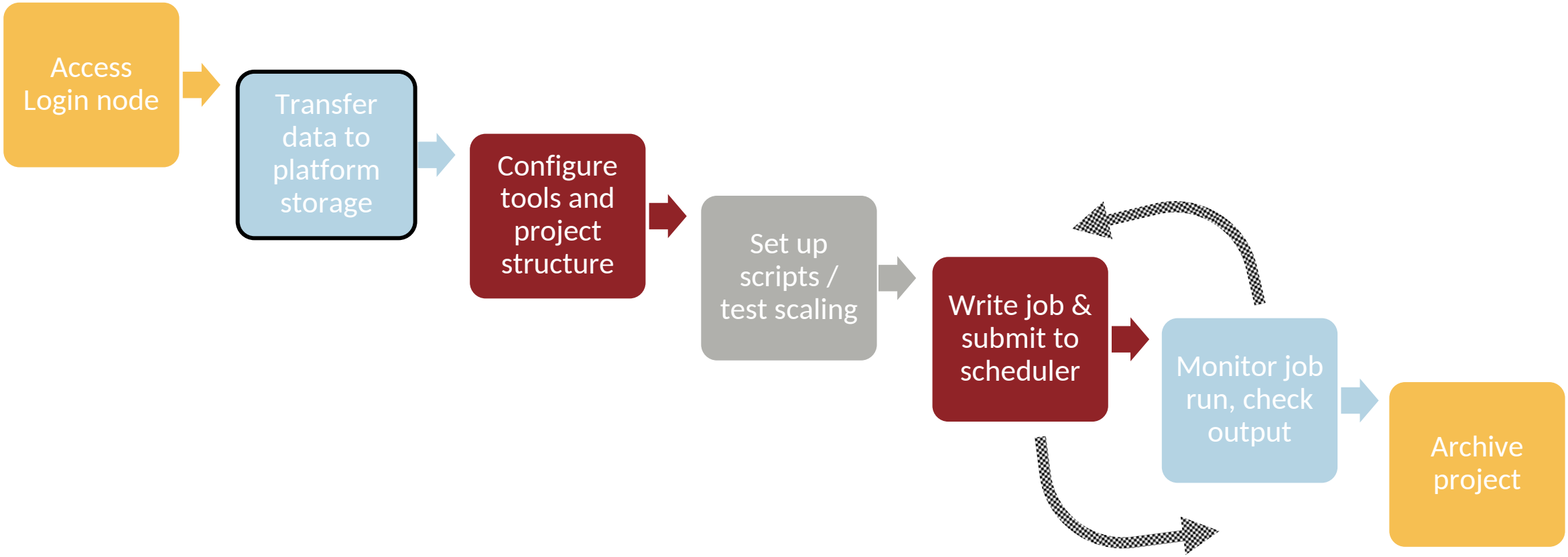
- pdw = /work (Open interface)
- pdw = /home/ucloud (SSH session)

- The symbol * is a wildcard for the file name

- HPC: no trash bin - removed files are lost forever - with no exception (rm)

- **Any questions/problems with the commands?**

Standard HPC workflow

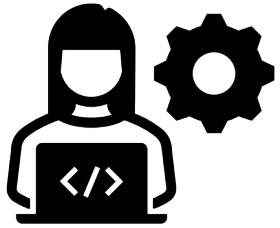


Collect & benchmark

- Availability of large public datasets, such the 1000 Genomes Project, TCGA, GEO… (promoting the open exchange of data and accelerating the pace of scientific discovery)
- Public datasets are particularly useful for **testing tools and methods**, allowing researchers to **validate** findings
- **Reduce costs and time** associated with primary data collection, thus focusing more on analysis and innovation.
- Public datasets help **increase sample size**, leading to more robust statistical power and improving the generalizability of results
- If collecting your own or collating data in a novel way…

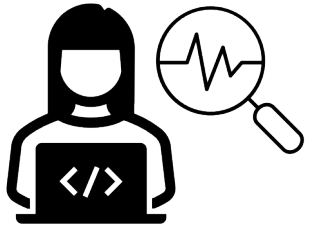
Public datasets

How do I select a dataset from existing resources?



Tool dev

- Scope that maps model strengths & weaknesses
- Common use / easy access
- Straightforward setup



Research

- Topical match
- High quality data
- Clear provenance
- Transferability

Teaching

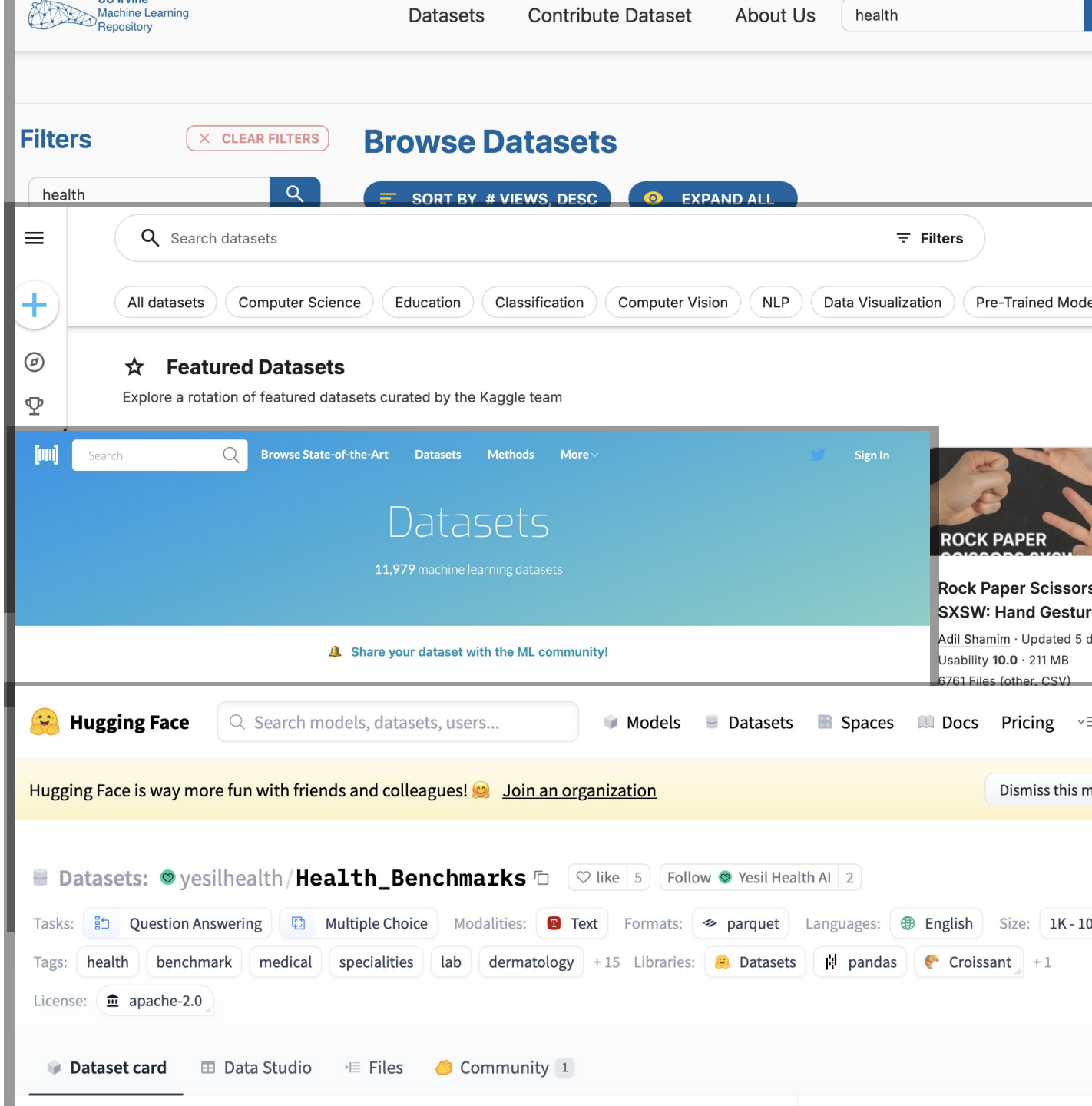
- Topical match
- Minimal corrections / preprocessing
- Easy OPEN access
- Multiple use cases

Data resources

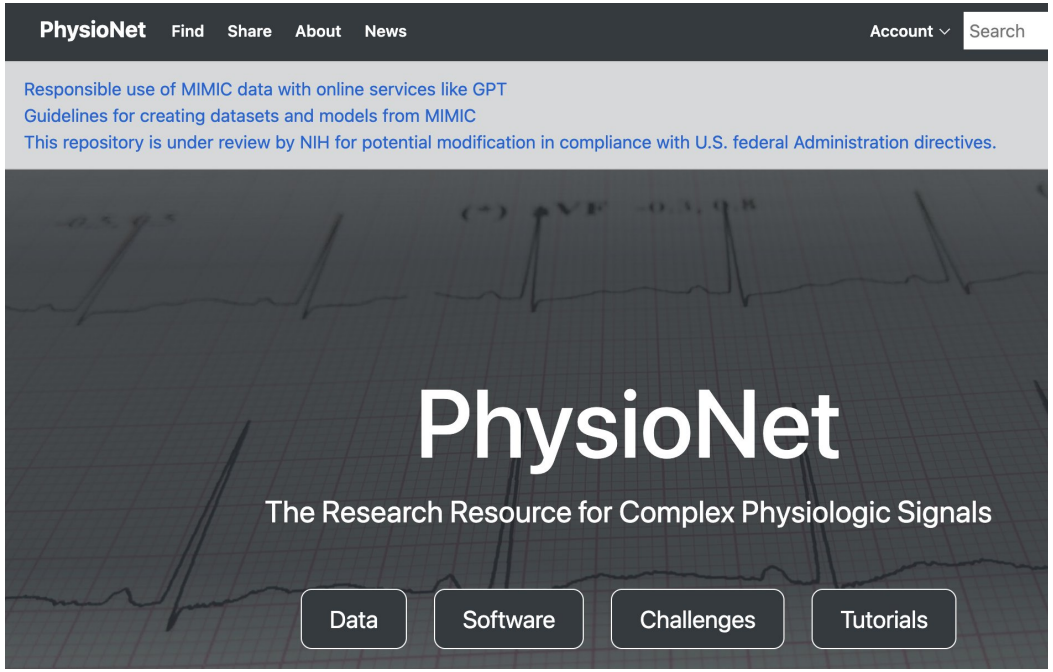
Free datasets (cross-disciplinary):

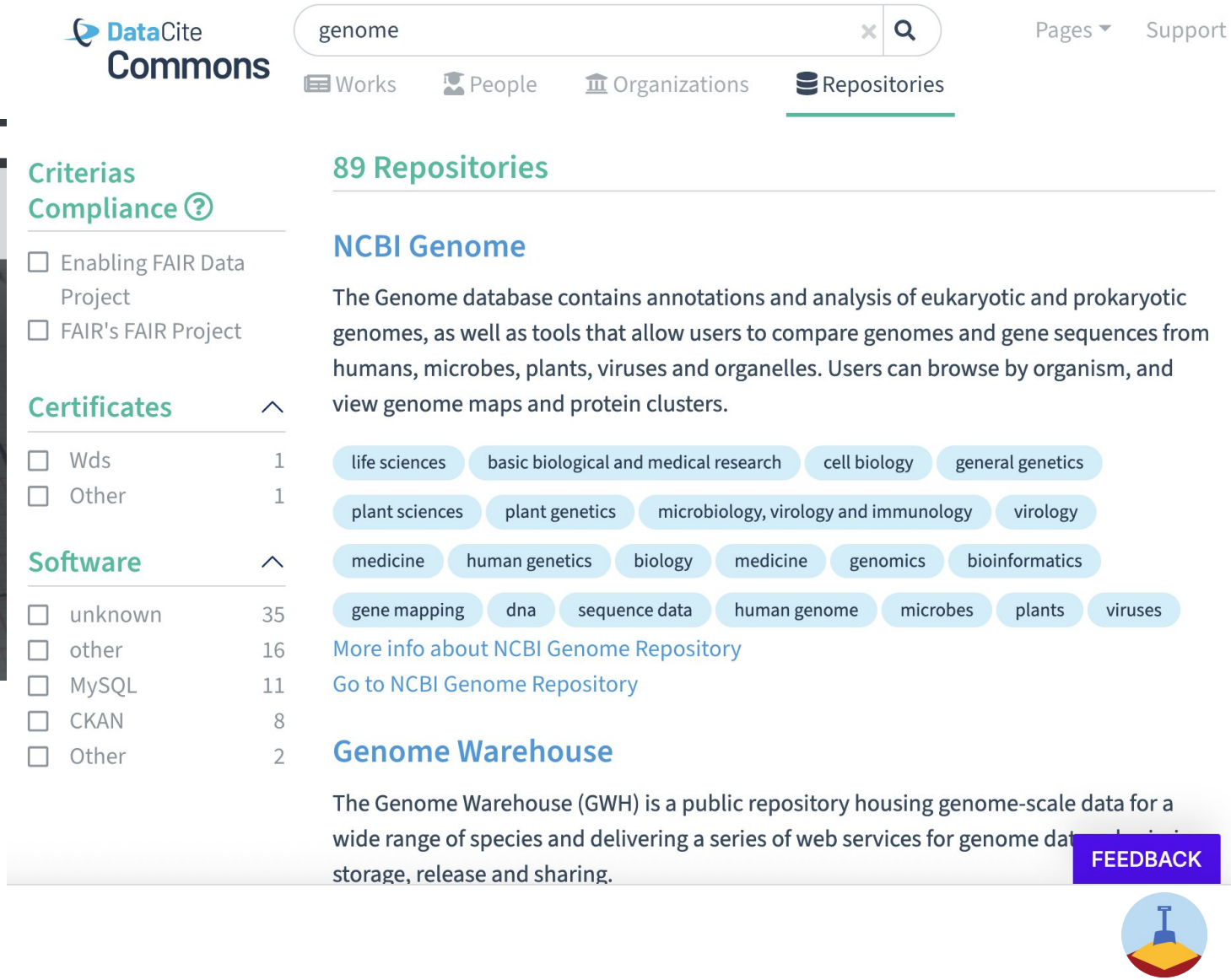
- UCI Machine Learning Repository
- Kaggle
- Papers with Code
- Hugging Face
- Zenodo

Many more open repositories for specific fields



Data resources





Downloading data

Quick wat to download data or source code.

Any file that can be downloaded in a web browser through a direct link can be downloaded using curl or wget.

Syntax:

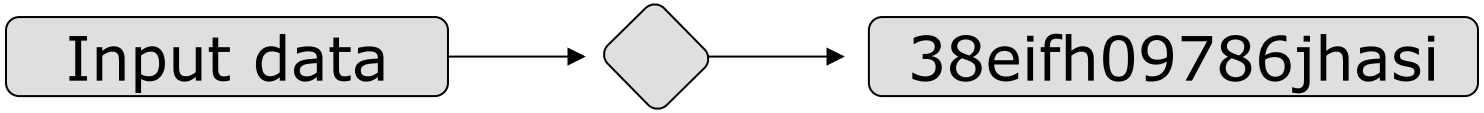
wget [-O new_name] https://some/link/to/a/file

curl [-o new_name] https://some/link/to/a/file

How do we ensure data integrity?

Checksums

A checksum is a unique value generated from a file's data to verify its integrity.



Cryptographic hashing algorithms: MD5, SHA1, SHA256

If you are **collecting** large files, ensure to calculate the checksum and compare to the original one.

If you are **generating and archiving** large datasets, store them with a checksum which will help others check if files are corrupted by checking their signature.

Go through the README (before downloading) & check for checksum file.

Checksums - example

1. Check README.md

- md5checksum.txt - a file detailing md5 checksums for all files in the top-level directory

```
MD5 (Alignments_Rel_3570.zip) = f5fa3ff180cc9a0bf292f1f452b9126d
MD5 (Allele_status.txt) = 3d6aef601f01b0e692c1c1d65a29c251
MD5 (Allelelist.txt) = 409841d4635f1d7af20d9a78003494d2
MD5 (hla_prot.fasta) = 7348fbef5ab204f3aca67e91f6c59ed2
```

2. Verify the file against the checksum

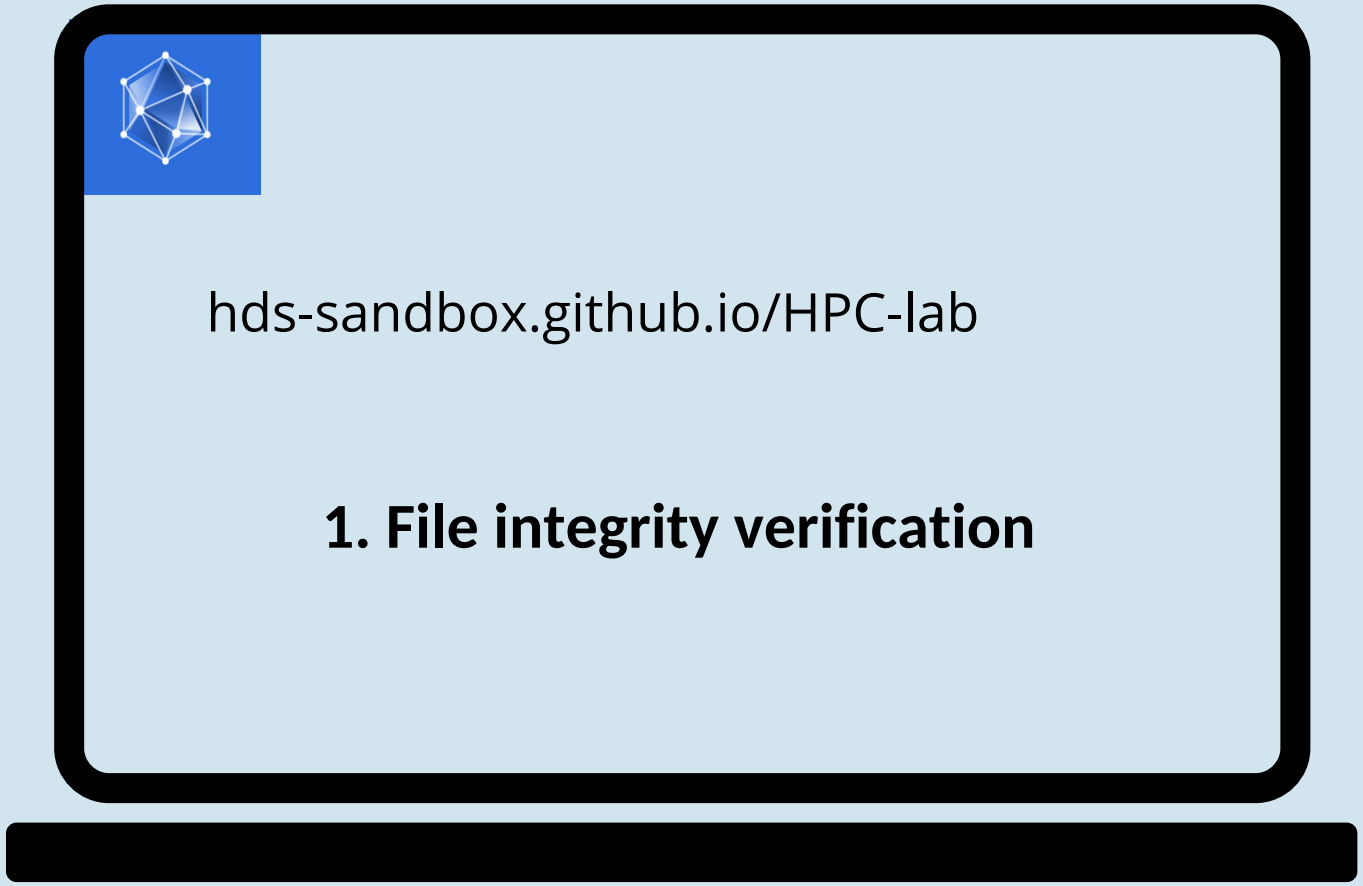
md5sum –-check mymd5checksums.txt

3. Generate the md5 hash & compare to the one you downloaded

md5sum <file>

Tip: ––quiet to not output anything when success

––status stop running is a job failed (useful for complex pipelines)



~ 15 min

Now is YOUR time!

Day 1 > HPC file transfers

How do you make sure files are not altered? md5sum, sha256sum

If you have extra time, do the Bonus exercise.

```
0438hasdf0283nsdf029nf9
```

Start a new job! **Please, don't forget to able tmux and SSH keys.**

Comments/questions

When are checksums important?

- Downloading large public datasets
- Moving files across unstable networks
- Ensuring reproducibility
- Validating long-term storage
- Archiving datasets (we will talk about this tomorrow!)

Data transfer

SFTP / SCP file transfer

Direct file transfers: (wget, rsync, specific API, …)

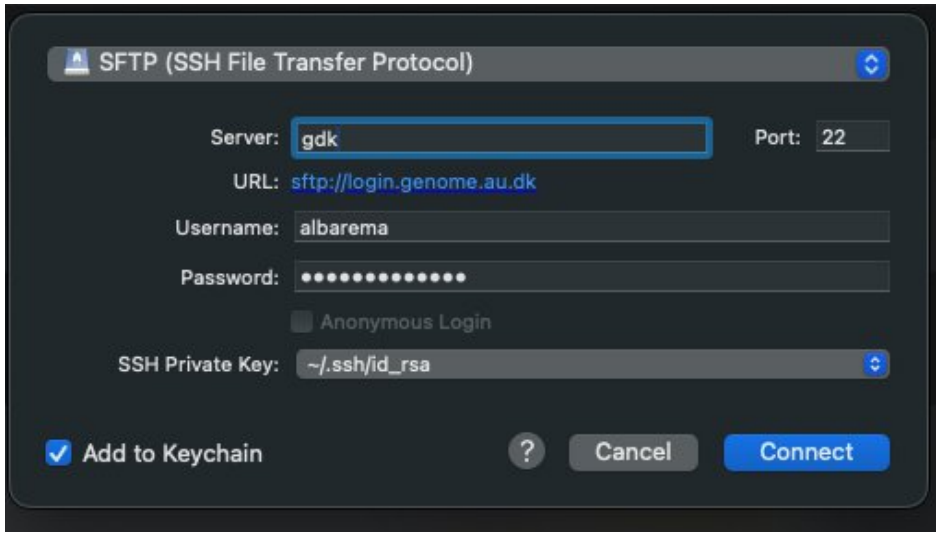
Backup your core data and scripts (on and off platform)

Direct file transfer to remote servers

	Filezilla	SCP	Rsync
Interface	GUI	Command Line	Command Line
Data synchronisation	yes	no	yes

Download/transfer interactively

How to establish a secure connection to an HPC?



Host: login.genome.au.dk
Username, Password
Port: 22

You might be required to provide additional login credentials (2FA)

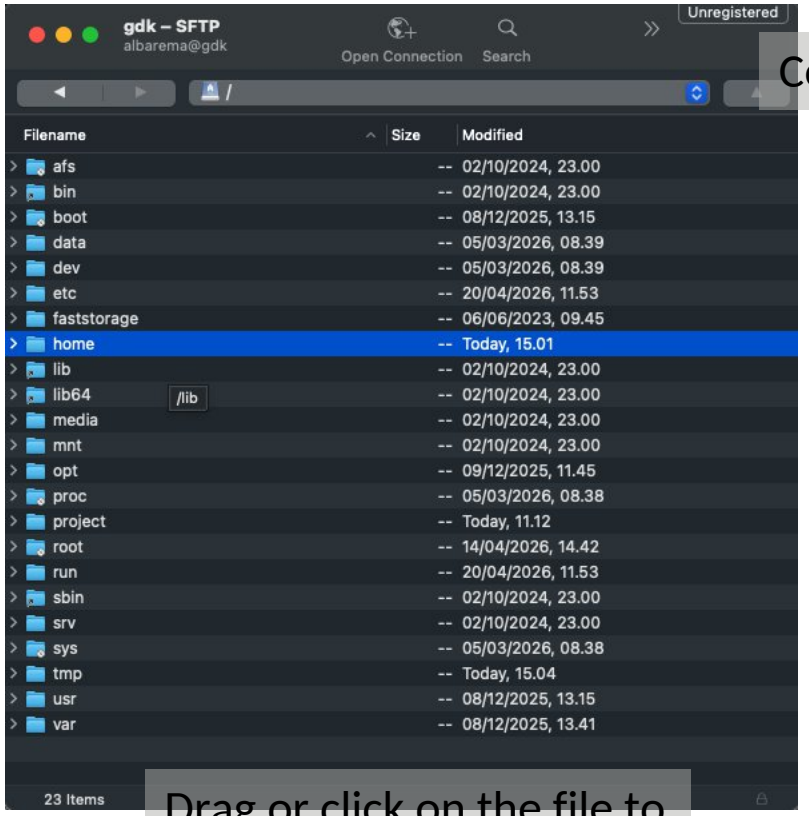
Pros:

- Easy to use (GUI-based)
- Data synchronization
- Can pause/restart transfer if needed

Cons:

- Not suitable if you want to transfer data between two remote servers (e.g. two HPC servers)

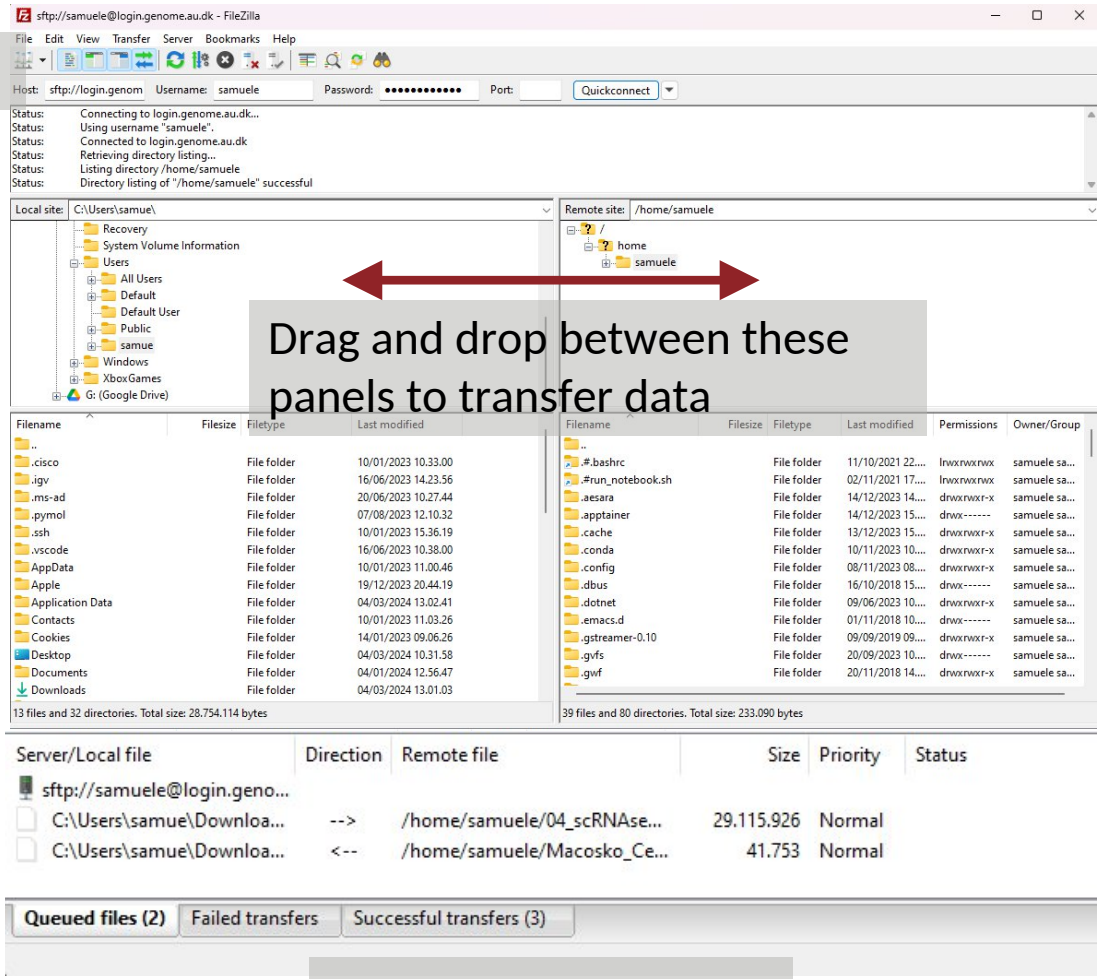
Cyberduck



Drag or click on the file to transfer data

FileZilla

Connection panel



Drag and drop between these panels to transfer data

Transfer progress

scp - local to HPC platform

You will need to specify your login, and the paths to the original file and the download destination.

scp – file/folder (-r) transfer (scp -> scp.exe on Windows)

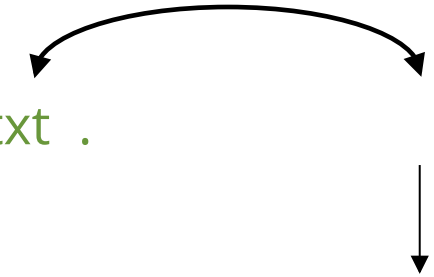
```
[local]$ scp –r proj/data login.genome.au.dk:/faststorage/project/HDSSandbox
```

- copy directories recursively (like cp)
- The **source** dir/file I want to copy (relative to my current dir)
- Credentials to access the HPC (same as with ssh)
- A separator
- The **destination** I want to copy it into

scp - HPC platform to local

Transfer a file to the current dir:

```
[local]$ cd ~/Documents
[local]$ scp login.genome.au.dk:/faststorage/project/HDSSandbox/file.txt  .
```



The credentials to access the HPC (same as with ssh)

The source file I want to copy

The destination I want to copy it into (relative to my current dir)

You can also use absolute paths!

scp

Advantages:

- Works from the command line
- Works similarly to standard cp command
- Can be used to move data between two remote servers:
  - First login to one of the servers with ssh
  - Then use scp from that server to the other (remote) server

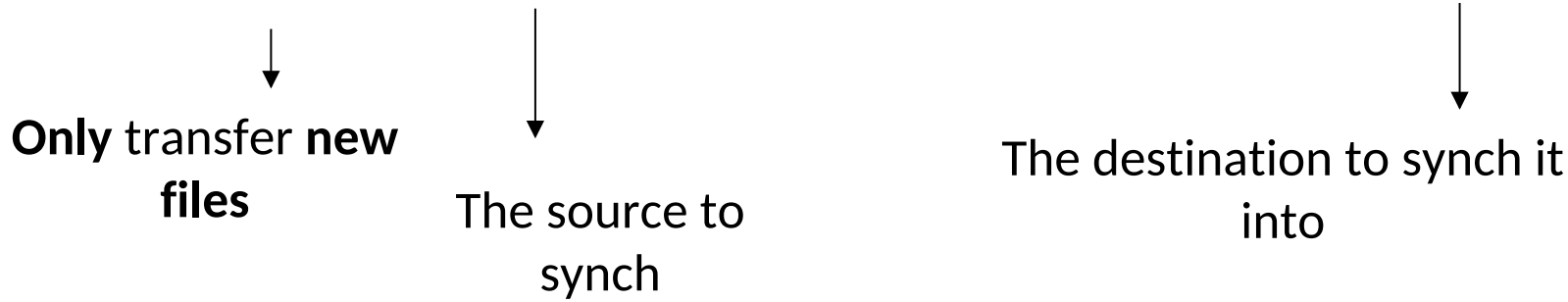
Disadvantages:

- Always copies all the files and overwrites them if they already exist (no sync ability)

Rsync - transfer and sync

Content sync (Linux/Mac)

[local]$ rsync –e ssh **–avhu** ./data/ user_name@login.genome.au.dk:data/

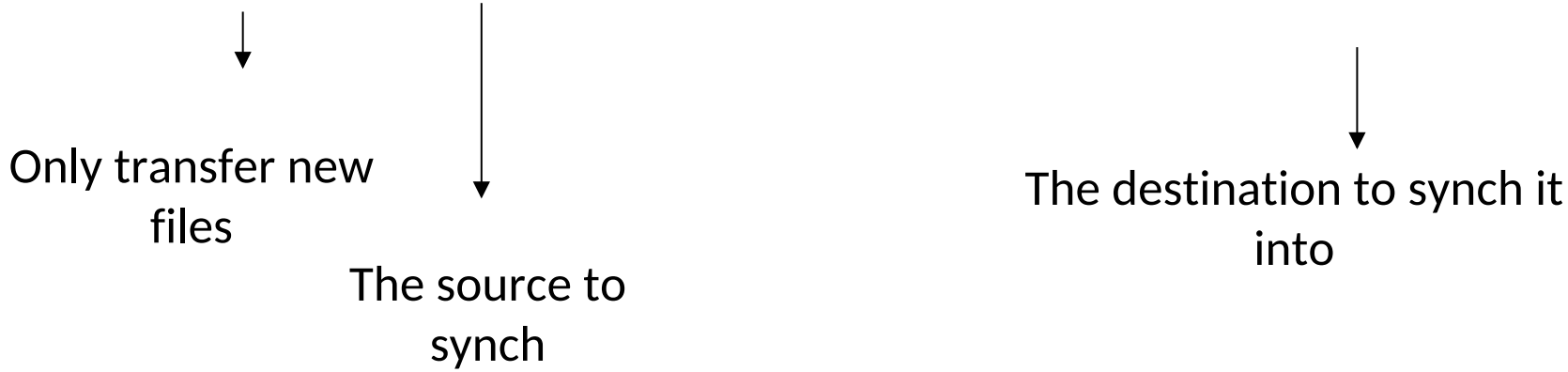


-a: archive
-v: verbose
-P: progress
-z: compress
-h: human-readable (numbers)
-r: specify the remote shell to use
-u: update

Rsync

Content sync (Linux/Mac)

[local]$ rsync –e ssh –avhu ./data/ user_name@login.genome.au.dk:data/



The / in the source path is important!

- If you include / at the end → transfer the **contents** inside the folder to the destination
- If you don’t → transfer the actual **entire folder** to the destination

Rsync (transfer and sync)

Content sync (Linux/Mac)

[local]$ rsync –e ssh –avhu ./data user_name@login.genome.au.dk:data/

Only transfer new files

The source to synch

The destination to synch it into

My computer:
/Users
|_gsd818
|_Documents
|_data

The HPC filesystem:
/data
|_ data

This would happen if you didn't include the / in source path

Rsync - Exercise

[local]$ cd Documents

[local]$ rsync –e ssh –avhu user_name@login.genome.au.dk:data/test data/

The HPC filesystem:
/data/
|_ test/
|_ file1.txt

The destination

After the transfer…?

A)

My computer:
/Users
|_gsd818
|_Documents/
|_data/
|_ file1.txt

B)

My computer:
/Users
|_gsd818
|_Documents/
|_data/
|_test/

Rsync - Exercise

[local]$ rsync –e ssh –avhu user_name@login.genome.au.dk:data/test data/

By excluding the / we are saying: “transfer the entire test folder to data”

After the transfer…?

A)

My computer:
/Users
|_gsd818/
|_Documents/ (WD)
|_data/
|_ file1.txt

B)

My computer:
/Users
|_gsd818/
|_Documents/ (WD)
|_data/
|_test/

Rsync - Exercise

The HPC filesystem:
/data/
|_ test/
|_ file1.txt

[local]$ **cd Documents**

[local]$ rsync –e ssh –avhu user_name@login.genome.au.dk:data/test/ ./

After the transfer…?

A)
My computer:
/Users
|_gsd818
|_Documents
|_data
|_test

B)
My computer:
/Users
|_gsd818
|_Documents
|_data/
|_test

C)
My computer:
/Users
|_gsd818
|_Documents
|_data/
|_file1.txt

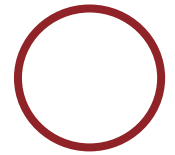
D)
My computer:
/Users
|_gsd818
|_Documents
|_data
|_file1.txt

Rsync - Exercise

The HPC filesystem:
/data/
|_ test/
|_ file1.txt

[local]$ cd Documents

[local]$ rsync –e ssh –avhu user_name@login.genome.au.dk:data/test/ ./



This would happen if you included
the /in the source path

After the transfer…?

A)
My computer:
/Users
|_gsd818
|_Documents
|_data
|_test

B)
My computer:
/Users
|_gsd818
|_Documents
|_data/
|_test

C)
My computer:
/Users
|_gsd818
|_Documents
|_data/
|_file1.txt

D)
My computer:
/Users
|_gsd818
|_Documents
|_data
|_file1.txt

Rsync - transfer and sync

Folder transfer on genomeDK

[local]$ rsync –e ssh –avz ./myfolder user_name@login.genome.au.dk:data/

Folder transfer on UCloud

[local]$ rsync -avP -e "ssh -i ~/.ssh/id_rsa -p <port>" ./myfolder ucloud@ssh.cloud.sdu.dk:/work/data/

The destination (/work IS THE WD)

Rsync - options

-a → only transfer files that have changed and preserve their timestamps and other properties (ownership, symbolic links, etc.)

-u → in addition, only transfer files if they are newer compared to the destination (avoids overwriting newer files with older versions by accident)

-h → print file sizes in human format like Mb, Gb, etc.

-v → verbose mode, print some information about what was transferred

--progress → show the progress of file transfer, useful for transferring larger files

-z → **compress** the data in transit (can save some bandwidth)

Tip:

--dry-run → shows you what files/folders rsync will transfer with your command, but not actually do the transfer.

Great to make sure you specified your paths and options correctly!

Rsync - summary

rsync versatile tool to download/update but also to create backups and versioning!

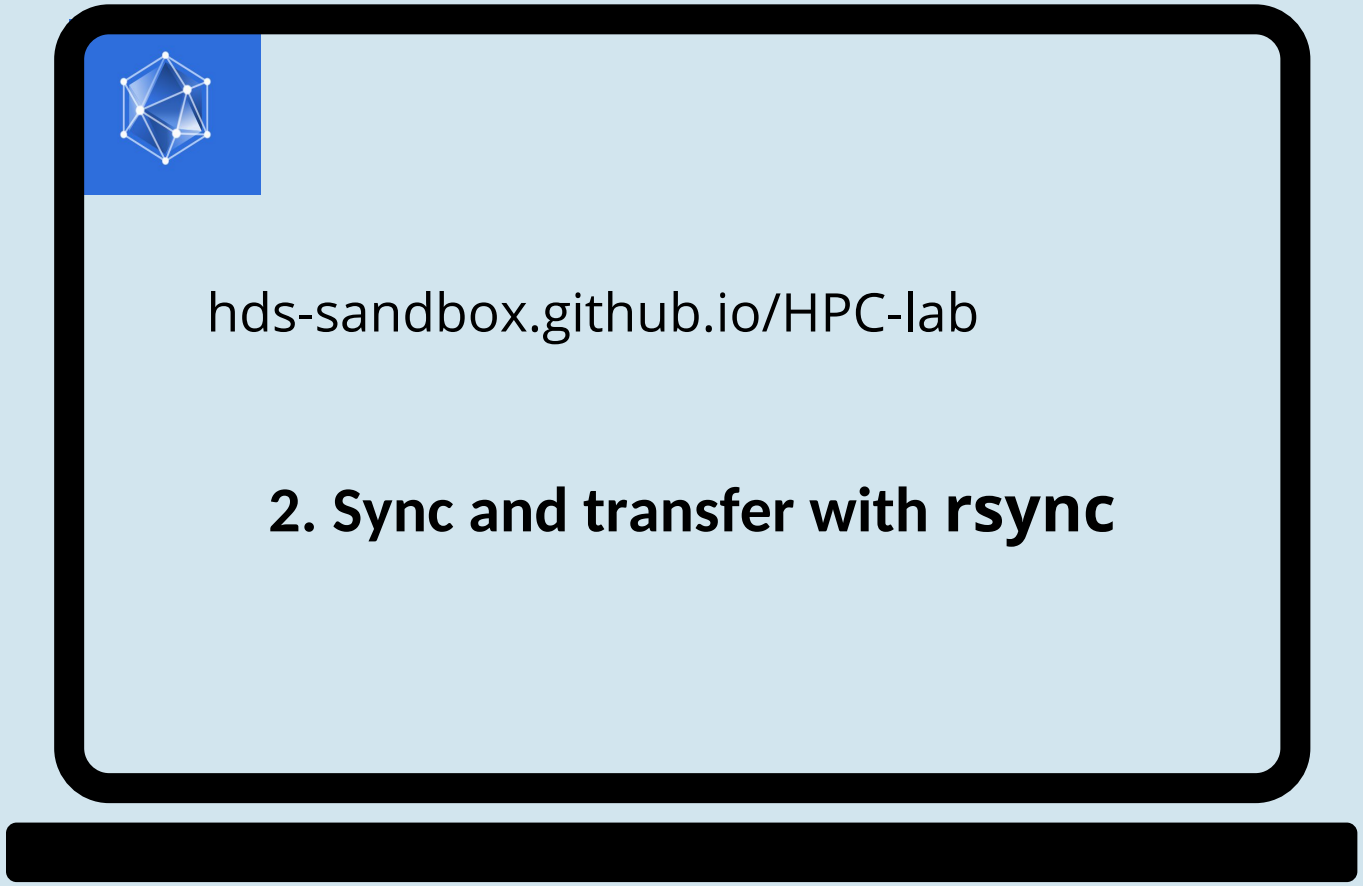
Advantages:

- Works from the command line
- Can be used to move data between two remote servers (sometimes)
- Much more flexible than scp → can synchronize files saving time and bandwidth
- The option --dry-run allows testing what the command would do before actually running

Disadvantages:

- Many more options can make it a steeper learning curve
- The subtle / at the end of the source path can cause confusion

https://linux.die.net/man/1/rsync



~ 20 min

Now is YOUR time!

Day 1 > HPC file transfers

1. Synchronising
2. Transferring data

And learn some more basic bash commands! Make sure you understand what each command does

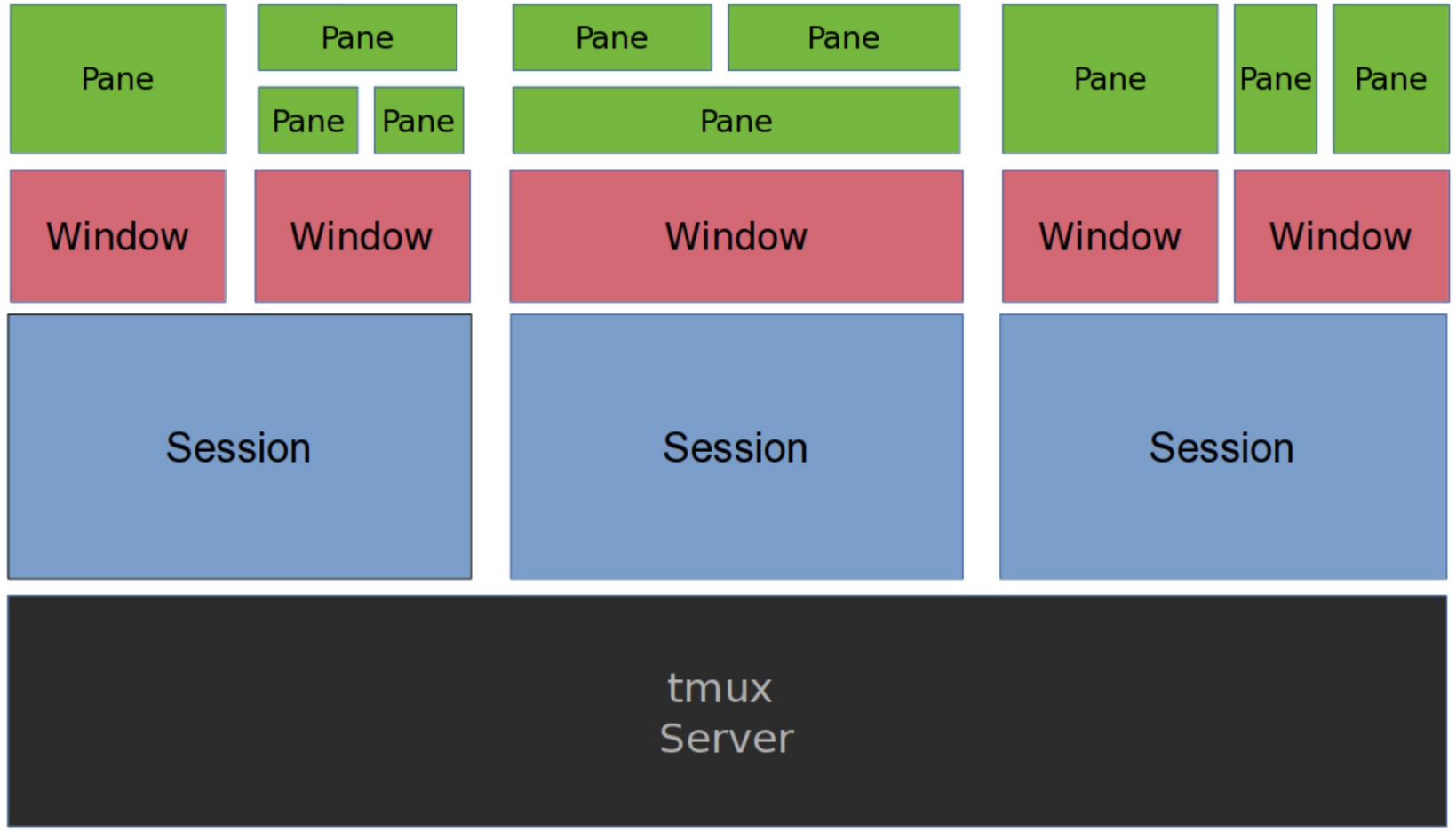
Everyone has transferred all fastq files to UCloud?

Everyone has now a copy of the day1 folder locally?

Terminal multiplexers - tmux

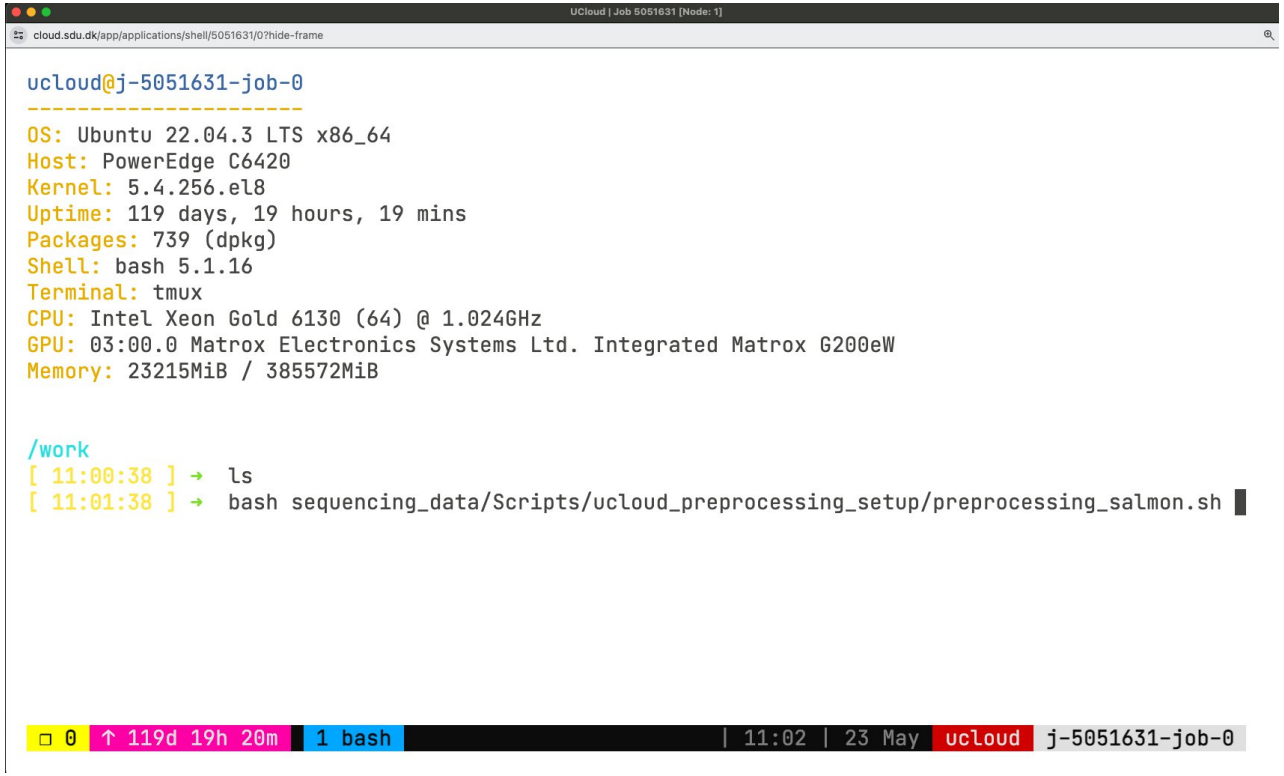
tmux is a very versatile tool for

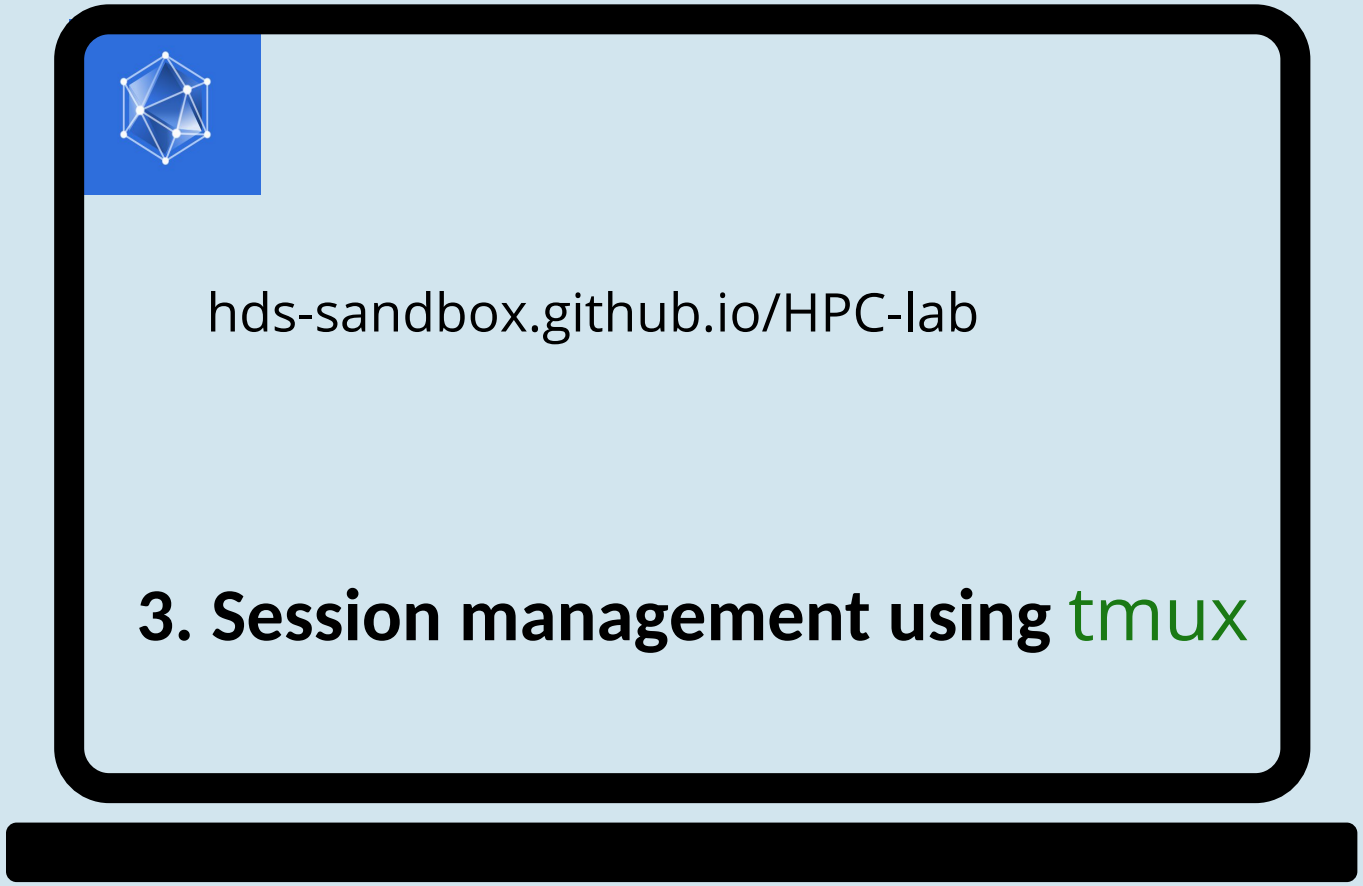
- creating and navigating multiple sessions



tmux

- Start a server with multiple sessions
- Each session containing one or more windows with multiple terminals (panes)
- Each terminal run simultaneously and be accessed (attached) or exited from (detached)
- The tmux server keeps running without a logged user





~ 10 min

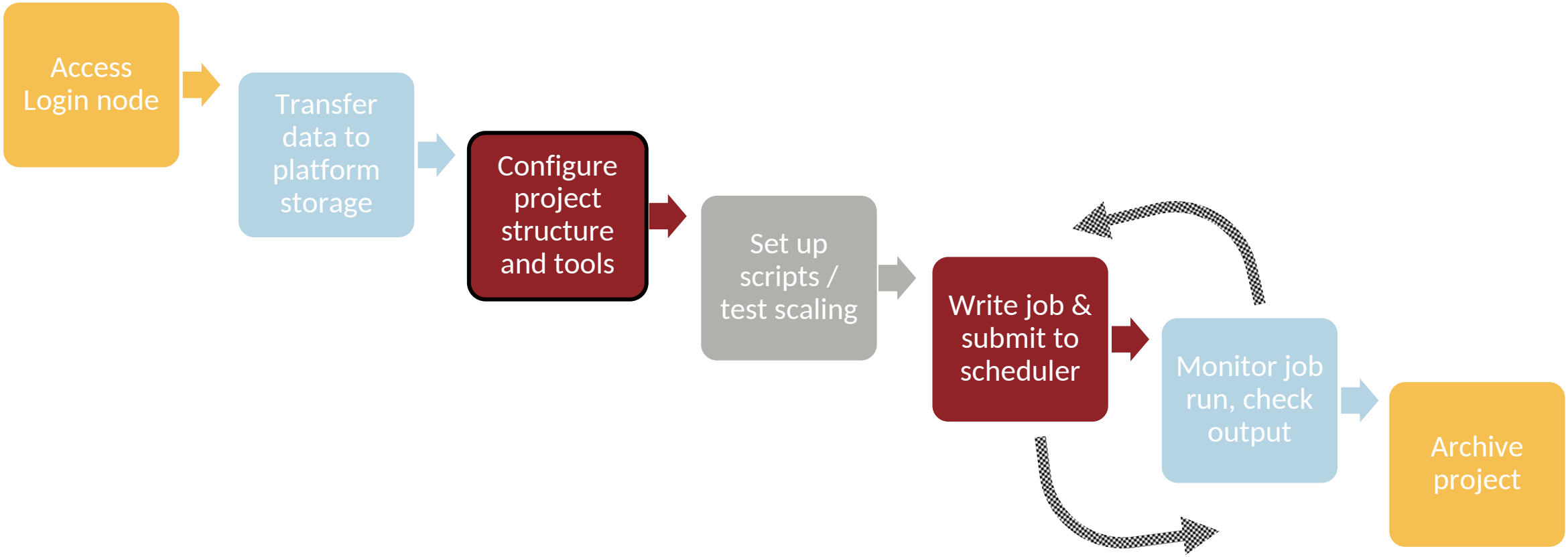
Now is YOUR time!

Day 1 > HPC file transfers

Let’s create a tmux session.

Problems/Issues/Comments

Standard HPC workflow



Configure project and tools

Version control

Folder templates & naming conventions

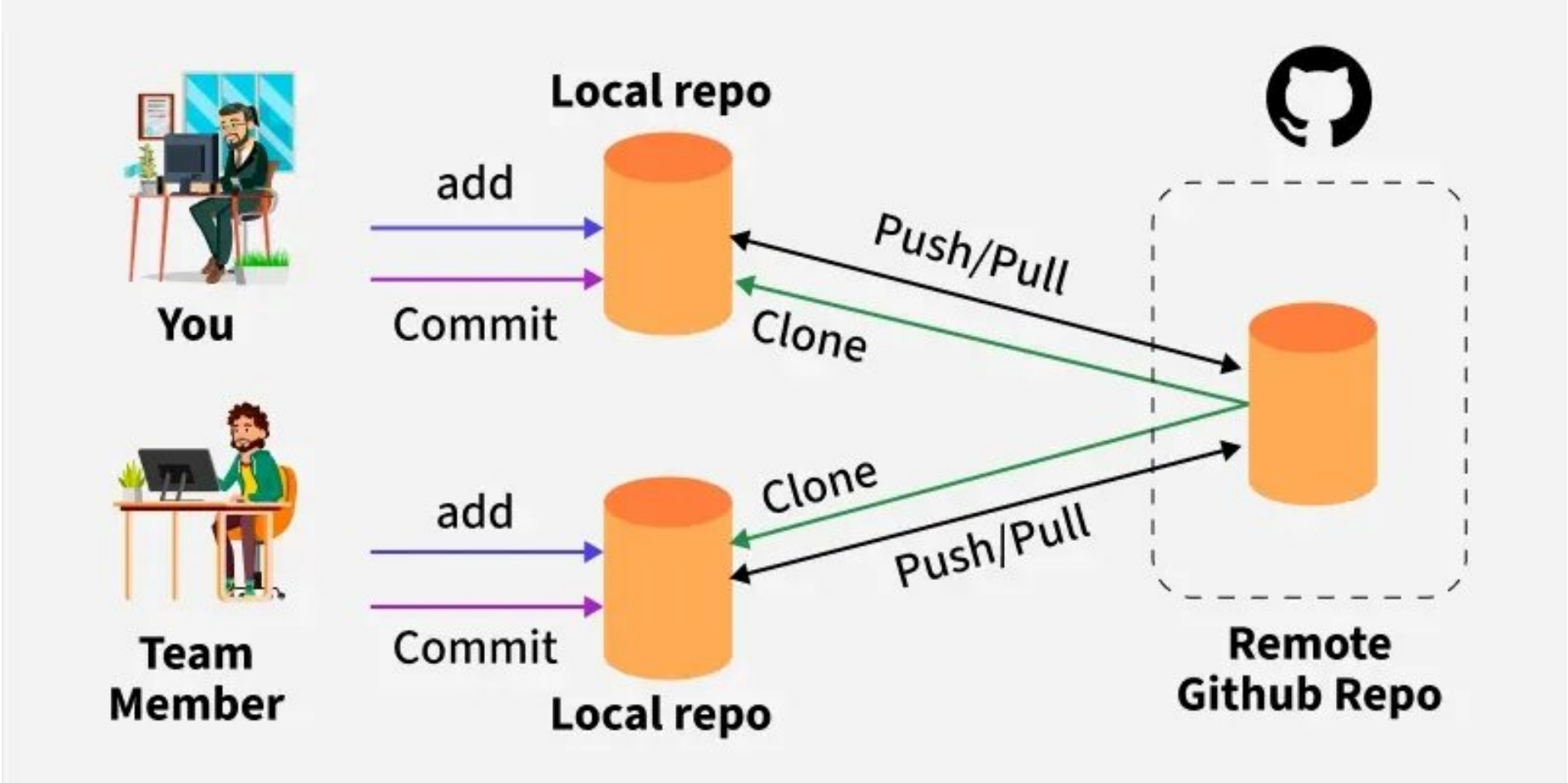
Managing software (conda, modules, containers…)

Version control

Git &
Github

- Stores files, branches, commits, and history of a project
  - Tracking file versions over time (who/what/when change)
- Manage multiple versions (possible to retrieve any previous state of your project)
- Supports collaboration, enabling multiple developers to work together without overwriting each other’s changes
- Can be cloned to create multiple copies on different machines

Version control



https://www.geeksforgeeks.org/git/what-is-a-git-repository/

Version control

Git

- Version control system (software)
- Command-line tool
- Manages different versions of edits

GitHub/GitLab (etc.)

- Hosting service and collaboration tools built around git tool
- Hosted on the web
- Provides a graphical interface
- The space to upload a copy of the git repository

Git-annex: tool for version control large files

Git commands: git [command] [options] [args]

git init

#Initialize a repo in your project folder

git add filename

#Add filenames or an entire folder to the stagging area (to track changes on files)

git commit –m "Add some files"

#Record changes (of any tracked file) with a message to remember

git push –m "Add some files"

#Transfers/uploads all the commits of the current repository to the remote repository (local -> remote)

git pull -u origin master

#Downloads changes from others and merges them into your current repository (remote -> local)

.gitignore, git merge, git status, git log

Types of Git repositories

Local repository

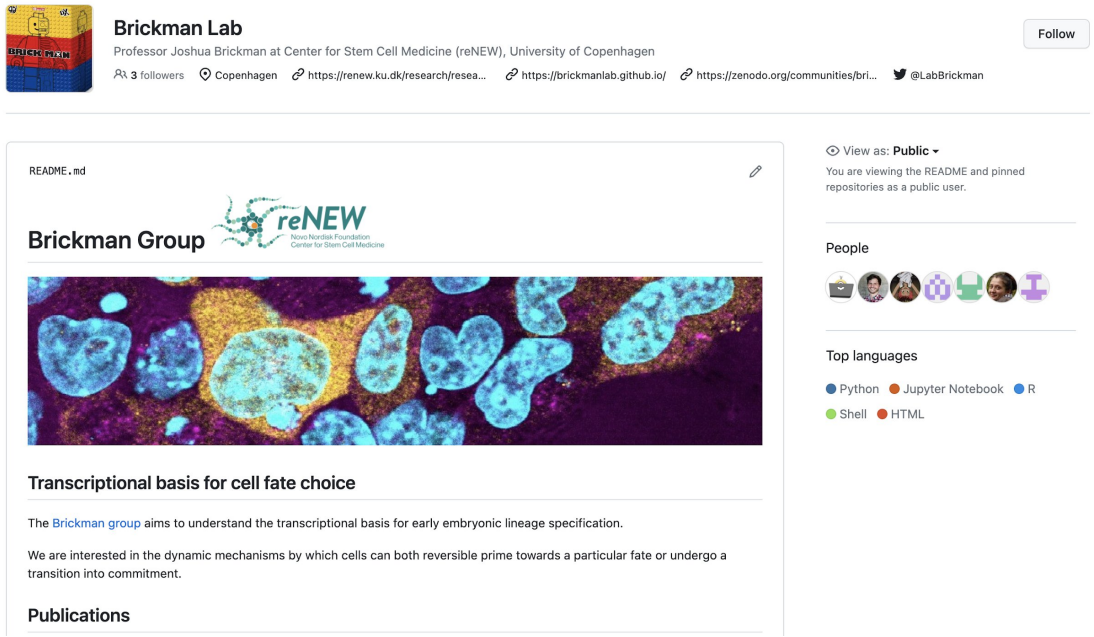
- Stored on your own computer or HPC
  - There might be multiple copies of the same repository
- Allows you to make changes, commit them and review your project history (no internet connection needed)

Remote repository

- Hosted on a server like GitHub, GitLab, or Bitbucket
- Enables multiple developers to collaborate on the same project
- Supports operations like push, pull and fetch to synchronize changes with the local repos
- They can be public (access by anyone), open source licenses or private (only members have access)

Version control – best practices

Make an organization page & start version controlling code, large files…



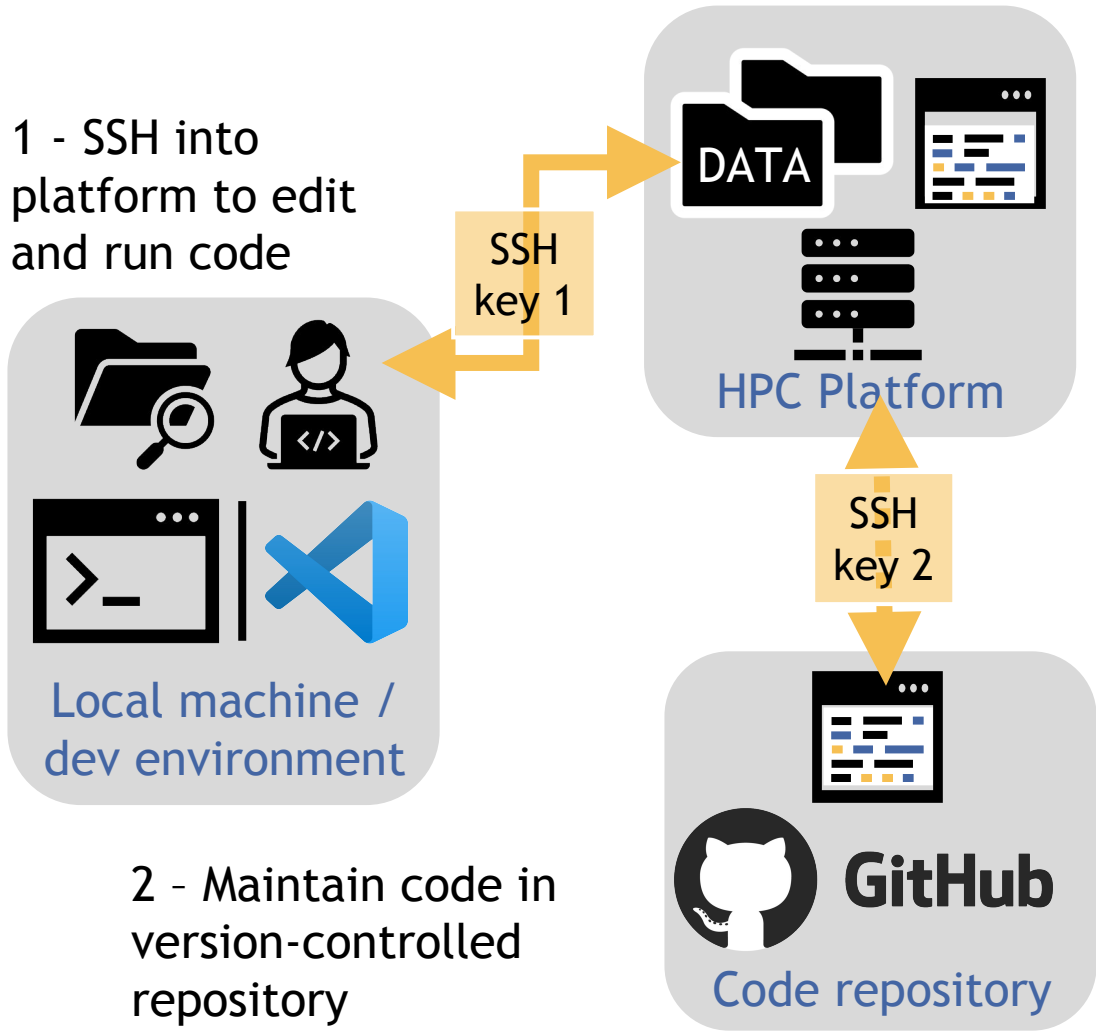
- ✓ Tracking changes
- ✓ Reproducibility
- ✓ Collaboration
- ✓ Documentation and metadata
- ✓ Accessibility
- ✓ Data integrity
- ✓ Backup

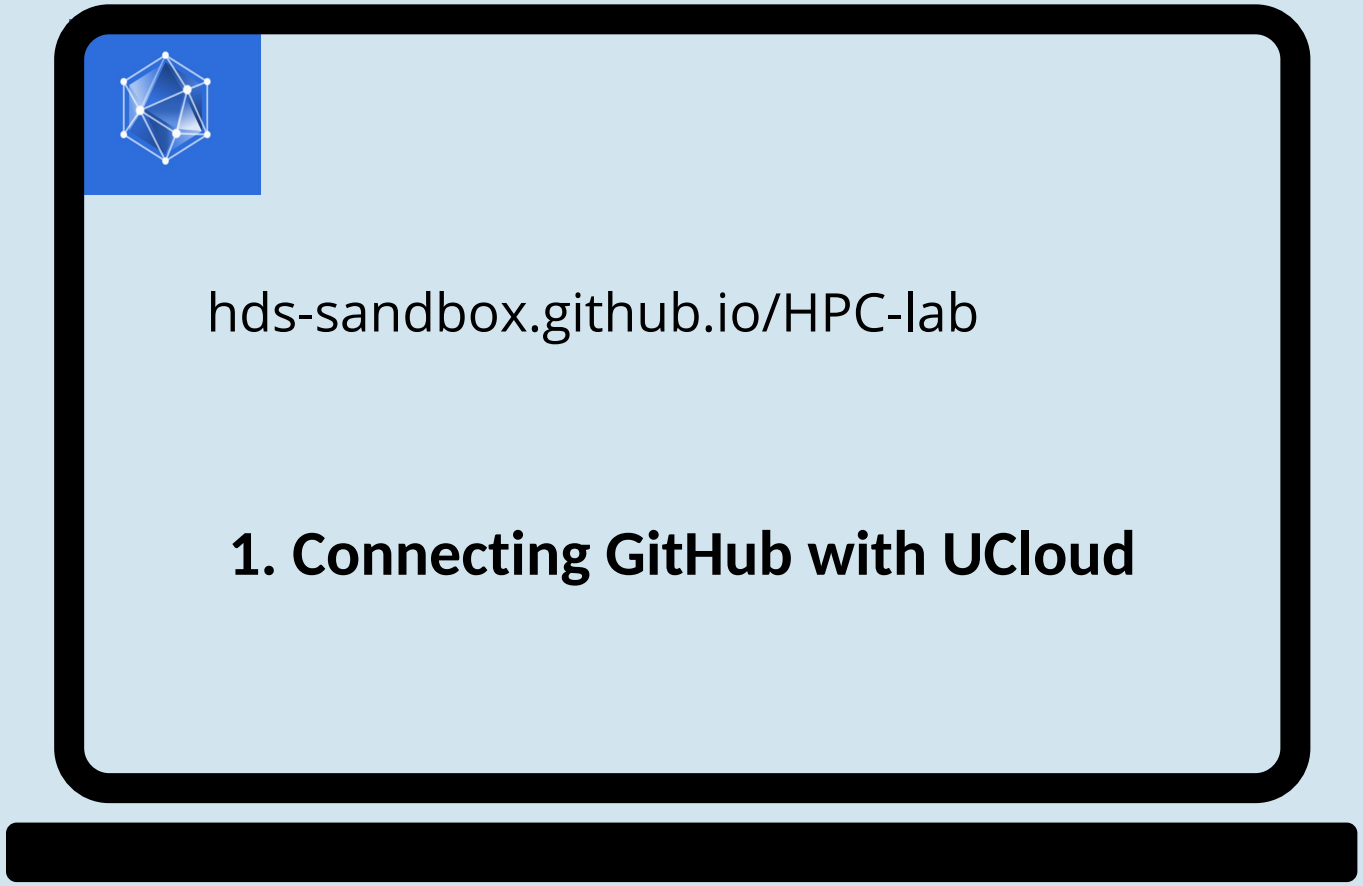
GitHub on UCloud

You can use GitHub on most HPC systems.

You can configure ssh keys to avoid writing your name and password every time that you push and pull changes.

Schematic for HPC platform operations





~ 10 min

Now is YOUR time!

Day 1 – Git & Github

Configure GitHub on UCloud

Add/commit/push some changes

You can create a repo or fork one of ours!

Stop the job! We will be using a different app for the next exercise.

Problems/Issues/Comments

Project structure – folder templates

- Create custom templates using Cookiecutter and share them on GitHub!
- Command line utility
  - Very flexible
  - Simple, but can do complex things
- Use the same structure across projects

- Good practice: two templates
  - Data folders: assays, datasets…
  - Project folders: data analyses

INPUT

1. Define a directory structure (from scratch or using a template)

```
cookiecutter-template/
├── {{ cookiecutter.project_name }}/ <--------- Project template
│       ├── scripts/
│       └── ...
├── readme.txt <--------- Non-templated files/dirs
│
└── cookiecutter.json <--------- Prompts & default values
```

2. cookiecutter.json to define variables (e.g., project_name) that can be referenced in the template

```
{
  "project_name": "MyProject",
  "author_name": "Your Name",
  "description": "A short description of your project"
}
```

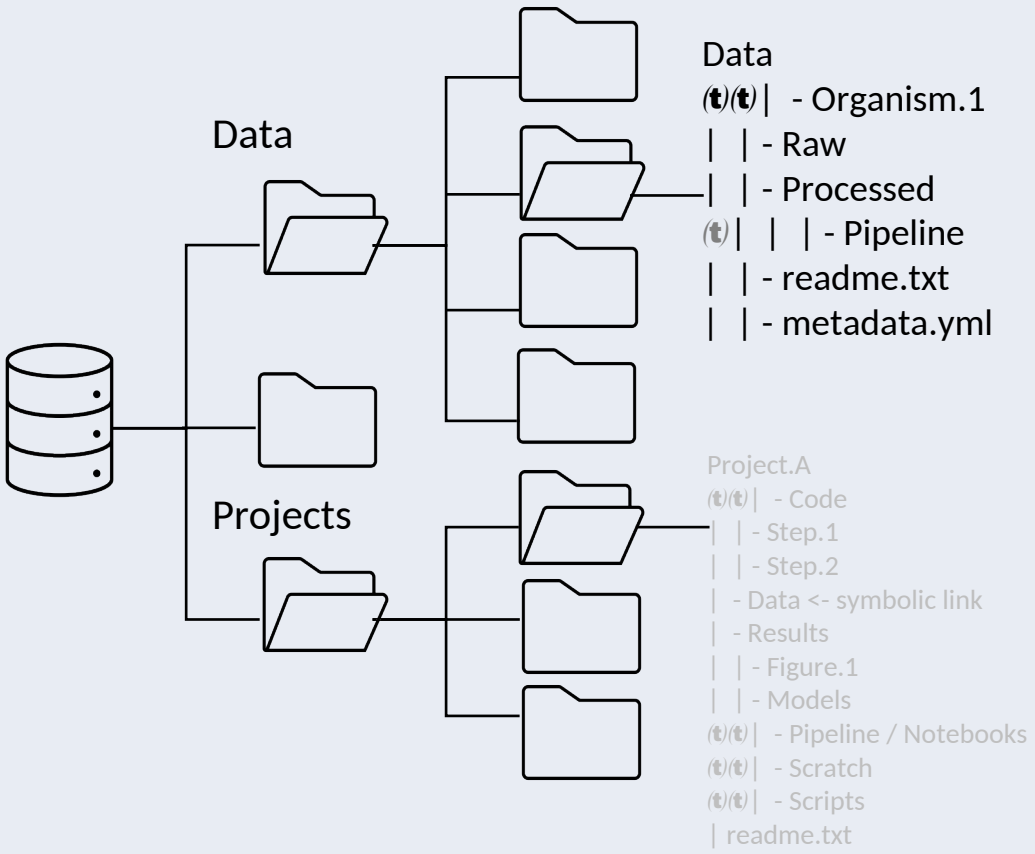
Cookiecutter will read the settings file, **prompt** the user interactively to enter the value for the variables (name of the project, author, date, etc.) and generate an output **directory structure**

OUTPUT

```
mytemplate/ <---------- Value corresponding to what you enter at the
│                                  project_name prompt
│  ├── scripts/
│  └── ... <-------- Files corresponding to those in your cookiecutter's
`{{ cookiecutter.project_name }}/` dir
```

It’s DEMO time!

FS – data dir example



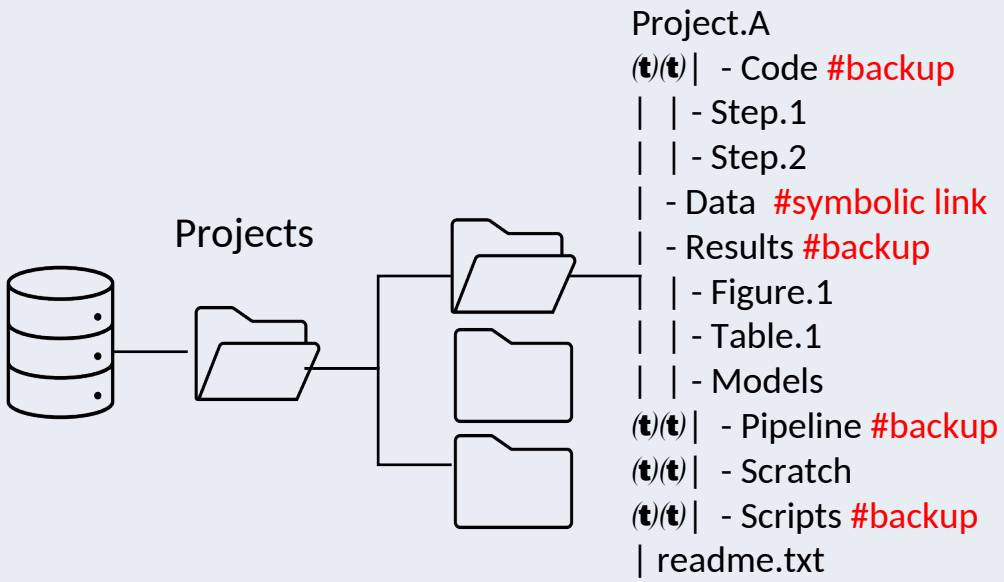
COOKIECUTTER

regular expressions 101

<project><method><description><DDMMYY><vX>.ext

- Separate data and data analyses
- Data folder content (raw, processed, etc.)
- Single copy, read-only (symbolic link)
- Downloading datasets using checksum file
- Document the data (non-proprietary formats)
- Use naming conventions, dates, and a folder structure! -> consistency, automation, querying…

FS – project dir example



COOKIECUTTER

regular expressions 101

<project><method><description><DDMMYY><vX>.ext

- Save all scripts/pipeline from input (raw/primary data) to results
- Results: subdirs by file types or analyses?
- Documentation for the project (readme.txt)
- Scratch or temporary folder
- Logs (warnings, errors, resource usage, configuration settings)

Extra tips for using HPCs efficiently

- Remove unused intermediate files (it is filling up the storage!)
- Backup only the established truth of your analysis (initial data and scripts used in the analysis)
- Restrict access to specific data (only authorised users can access it): chmod -R go-rwx <file or folder>
- Large folders containing unused output files should be zipped (save storage)
- Explore the different HPC storage types available on the server, such as /home, /scratch, /project, and /tmp, each designed for specific use cases.

Backup cost >> storage cost >> computation cost

Common HPC storage types

/home: personal storage space. Slower read/write performance. Good for source code, small params files and job submission scripts.

/scratch: short-term storage suited to intensive read/write operation on large files (> 100MB per file). Nor backed up and periodically purged. Good for checkpoint files, log output from jobs, and other data that can be easily recreated.

/project: larger storage quota (usually in PI's name). Good for sharing. Suited for large, static datasets and files that can be difficult or costly to reacquired or re-generated.

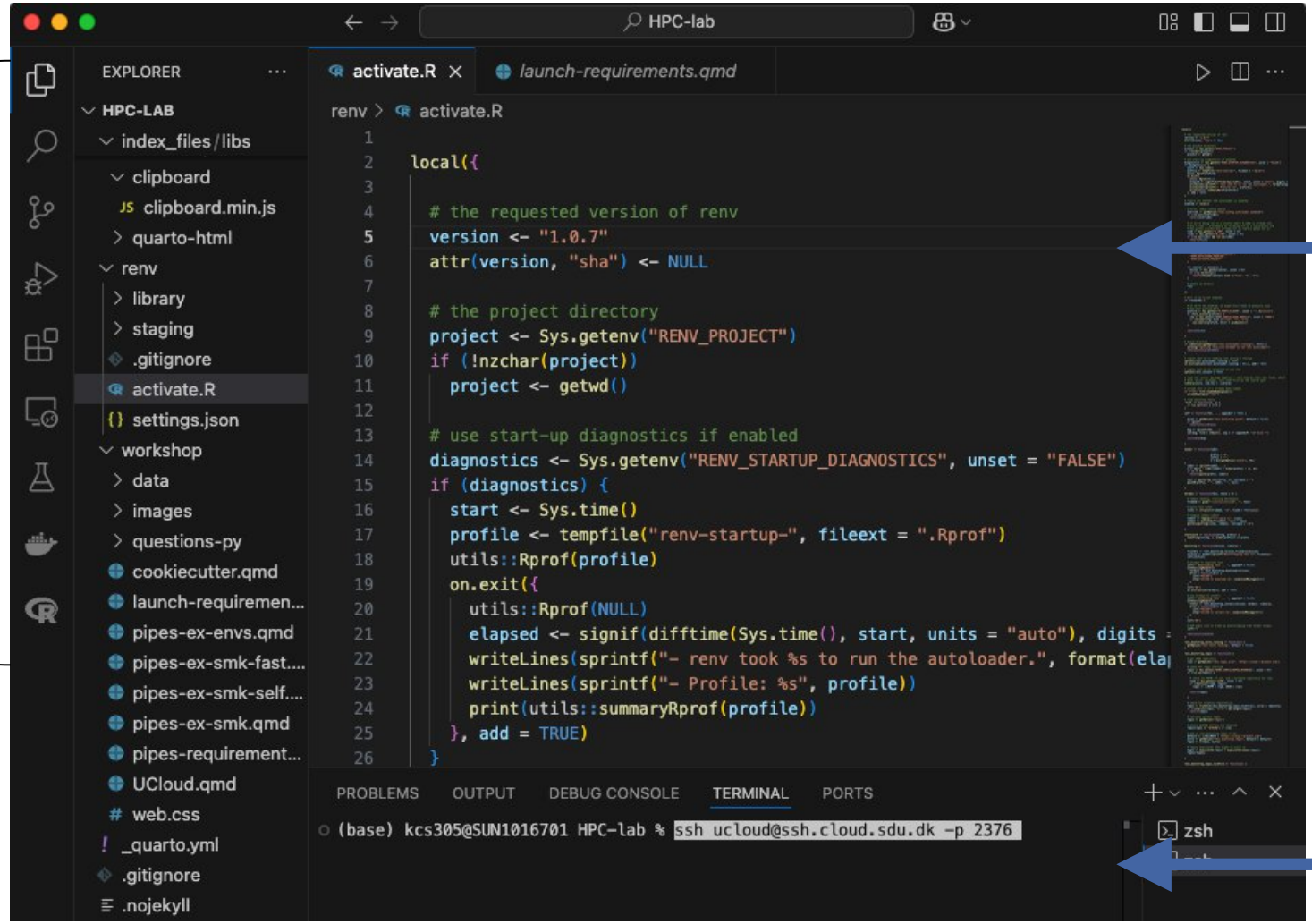
/tmp: temporary, fast, local storage used while a job is running. Delete when job ends.

Backup cost >> storage cost >> computation cost

Integrated development environment (IDE)

Visual Studio Code
- multi-language support, integrations like file explorer, git & terminal, many helper plug-ins!

Side bar



Editor

Terminal

FS

~ 15 min

Now is YOUR time!

Day 1 – Project Structure

Start a new job (VS code)

Create your tailored project structure!

If you have extra time, do the Bonus exercise.

Problems/Comments

- How does your folder structure look like now?
- Did you successfully create the reports/figures folder?

```
/work/hpcLaunch/day2 via hpclab-env
[ 15:22:56 ] →  cookiecutter https://github.com/hds-sandbox/cookiecutter-template
  [1/7] Project directory name [Example: project_short_description_202X] (): test_template_hpcLaunch
  [2/7] project_slug (test_template_hpclaunch): test
  [3/7] Author of the project (): ARM
  [4/7] Date of project creation, default is today's date (20260518):
  [5/7] Provide a detailed description of the project (context/content) (): testing cookiecutter template
  [6/7] version (0.1.0):
  [7/7] create_author_file (y):
```

Naming conventions

Use consistent order – Decide on a logical sequence for file details (e.g., date, project, version).

Plan for sorting & searching – Choose a filename structure that makes files easy to find.

Avoid spaces & special characters – Use underscores (file_name), dashes (file-name), or camel case (FileName).

Use leading zeros for numbers – Ensures proper sorting (001, 002, … 100).

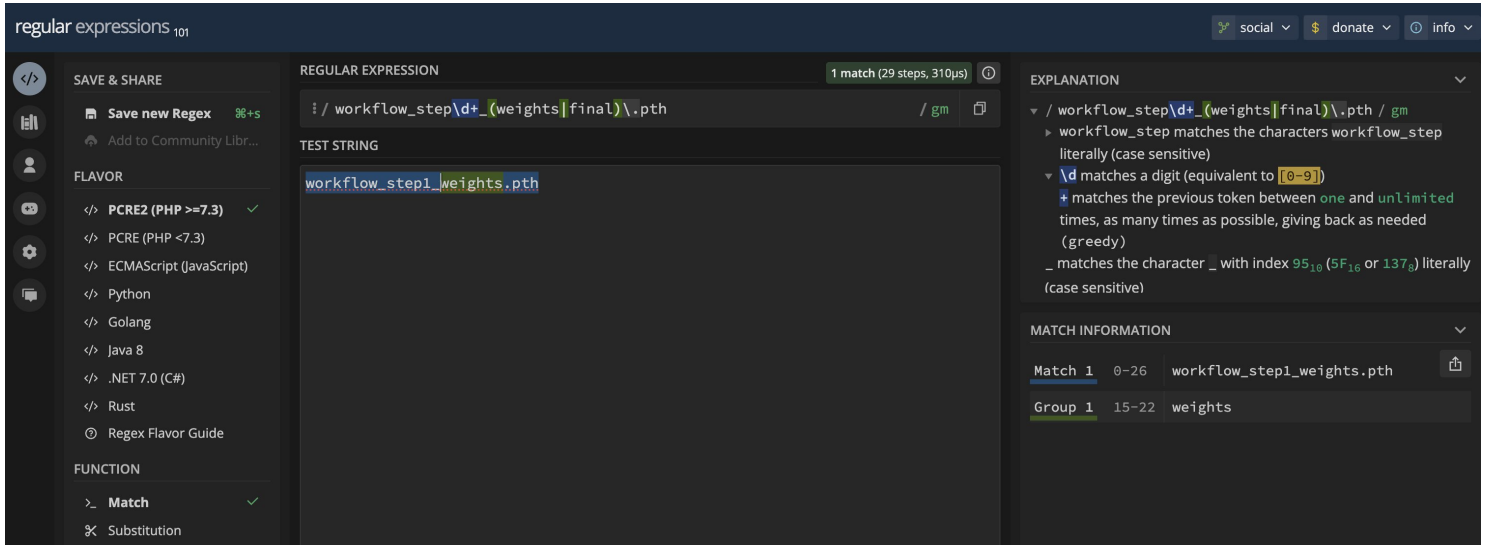
Document naming conventions – Record standards in a README.txt so others can follow them.

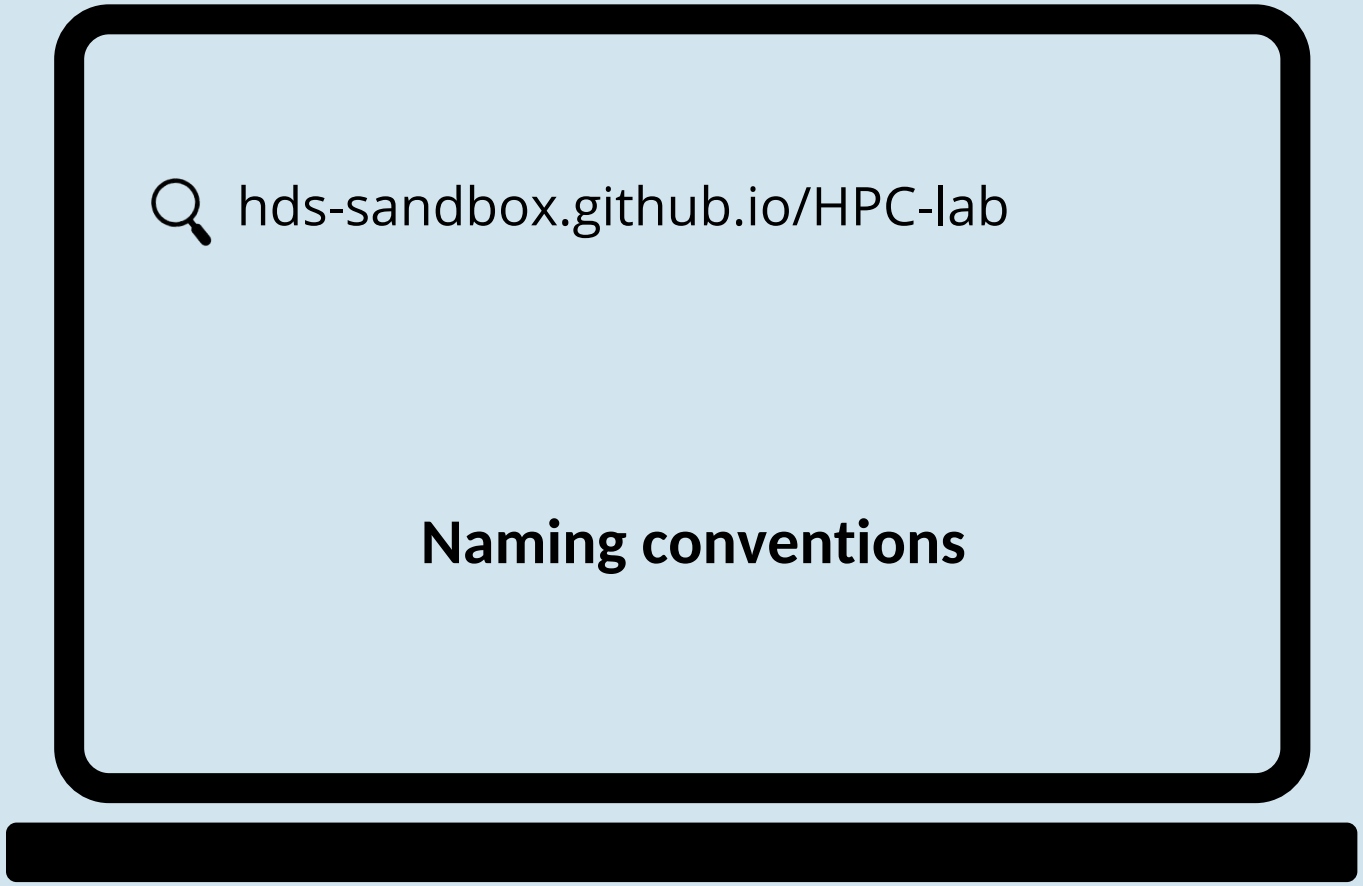
Filenames must be human-readable and easily parsed by scripts

Regular expressions help enforce structured naming

- Ensure compatibility with workflow managers like Snakemake or Nextflow
- Renaming multiple files: convert S1_R1.fastq, S1_R2.fastq to S001_R1.fastq, S001_R2.fastq
- Validating filenames before workflow execution: ensure all input files match sample_[A-Za-z0-9]+.fastq.gz
- Standardizing workflow intermediate files: match output patterns like step_\d+_result.txt

regex101.com





<project>_<analysisMethod>_<description>_<YYMMDD>_<vX>.ext

~ 5 min

Now is YOUR time!

Day 1 – Project structure

Why? Choosing the right naming conventions…

- Improves searchability & organization
- Reduces errors & enhances reproducibility
- Enables automation & consistency
- Batch-process & standardize files

Exercise 2, Q1: Which file names should not be used and why?

Grant proposal final.doc

differential_expression_results_clara.csv

sequence_alignment$v1.py

scripts/data_processing_carlo's.py

data/raw_sequences_V#20241111.fasta

data/gene_annotations_20201107.gff

alpha~1.0/beta~2.0/reg_2024-05-98.tsv

alpha=1.0/beta=2.0/reg_2024-05-98.tsv

run_pipeline:20241203.sh

Solution 2, Q1: Which file names should not be used and why?

A. Grant proposal final.doc
B. differential_expression_results_clara.csv
C. sequence_alignment$v1.py
D. scripts/data_processing_carlo's.py
E. data/raw_sequences_V#20241111.fasta
F. data/gene_annotations_20201107.gff
G. alpha~1.0/beta~2.0/reg_2024-05-98.tsv
H. alpha=1.0/beta=2.0/reg_2024-05-98.tsv
I. run_pipeline:20241203.sh

Exercise 2, Q2: Which file names are more readable?

1a. forecast2000122420240724.tsv

1b. forecast_2000-12-24_2024-07-24.tsv

1c. forecast_2000_12_24_2024_07_24.tsv

2a. 01_data_preprocessing.R

2b. 1_data_preProcessing.R

2c. 01_d4t4_pr3processing.R

3a. B1_2024-12-12_cond~pH7_rep2.fastq

3b. B1.20241212.pH7.rep2.fastq

3c. b1_2024-12-12_c0nd~pH7_r3p2.fastq

Solution 2, Q2: Which file names are more readable?

1a. forecast200012242024072 4.tsv

1b. forecast_2000-12-24_2024-07-24.tsv: easier for human & machine, “_” separates dates, “-” separates within time information (year/month/day)

1c. forecast_2000_12_24_2024_07_24.tsv

2a. 01_data_preprocessing.R: start with 0 for sorting, consistently with upper/lower case and the use of separators (“_”)!!

2b. 1_data_preProcessing.R

2c. 01_d4t4_pr3processing.R

3a. B1_2024-12-12_temp~37C_rep2.fastq: indicates variable temperature is set to 37 Celsius (temperature could be negative ”-” and is better used to separate values in time)

3b. B1.20241212.37C.rep2.fastq

3c. b1_2024-12-12_t3mp~37C_r3p2.fastq

Managing software

HPC-pipes

Containerization

Virtual environments

Package & environment managers

Managing software

Software installation is heterogeneous

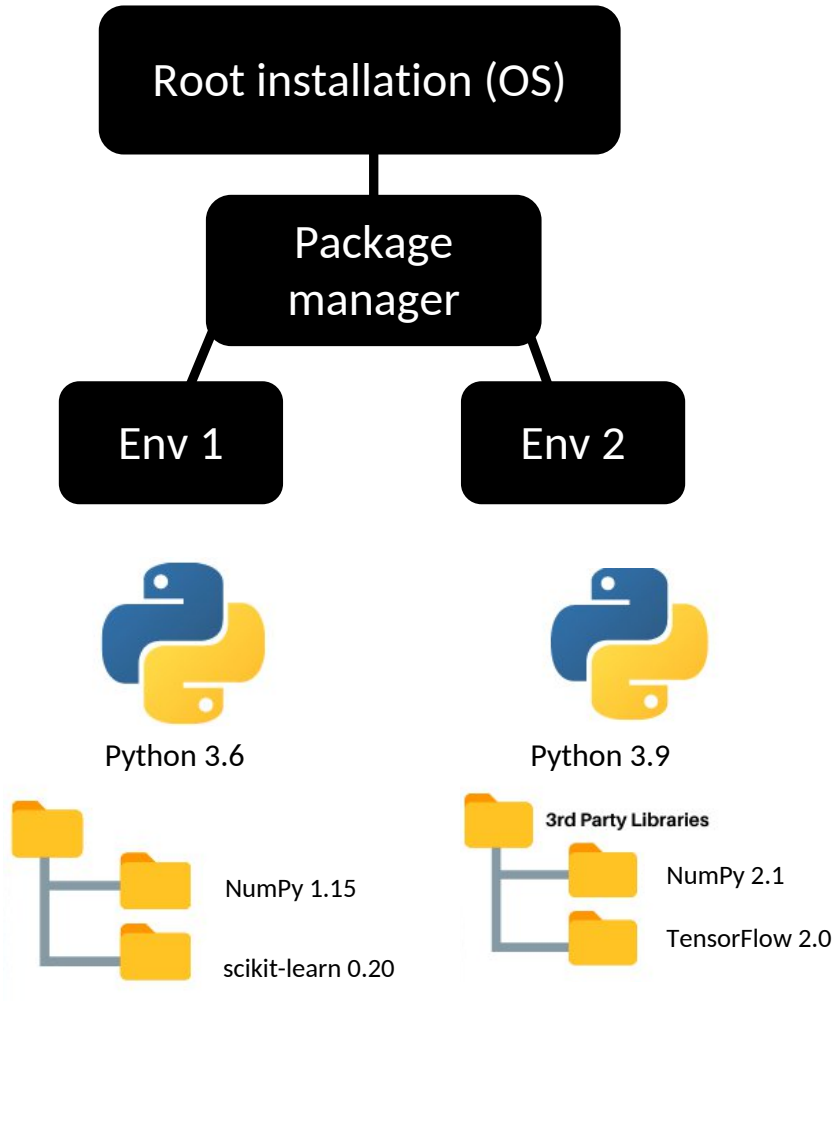
```
> install.packages("BiocManager")
> BiocManager::install("DESeq2")

> pip install cookiecutter

> apt-get install bwa

> cpan –i bioperl

> yum install python-h5py

> ./configure –-prefix=/usr/local
> make
> make install
```

Managing software – main concepts

A **virtual environment** isolates packages/software so different projects can use different versions without conflicts.

Package managers handle installation, dependency resolution, updates, and removal.

Dependencies are additional material (software/library) that a software requires for installation and operation.



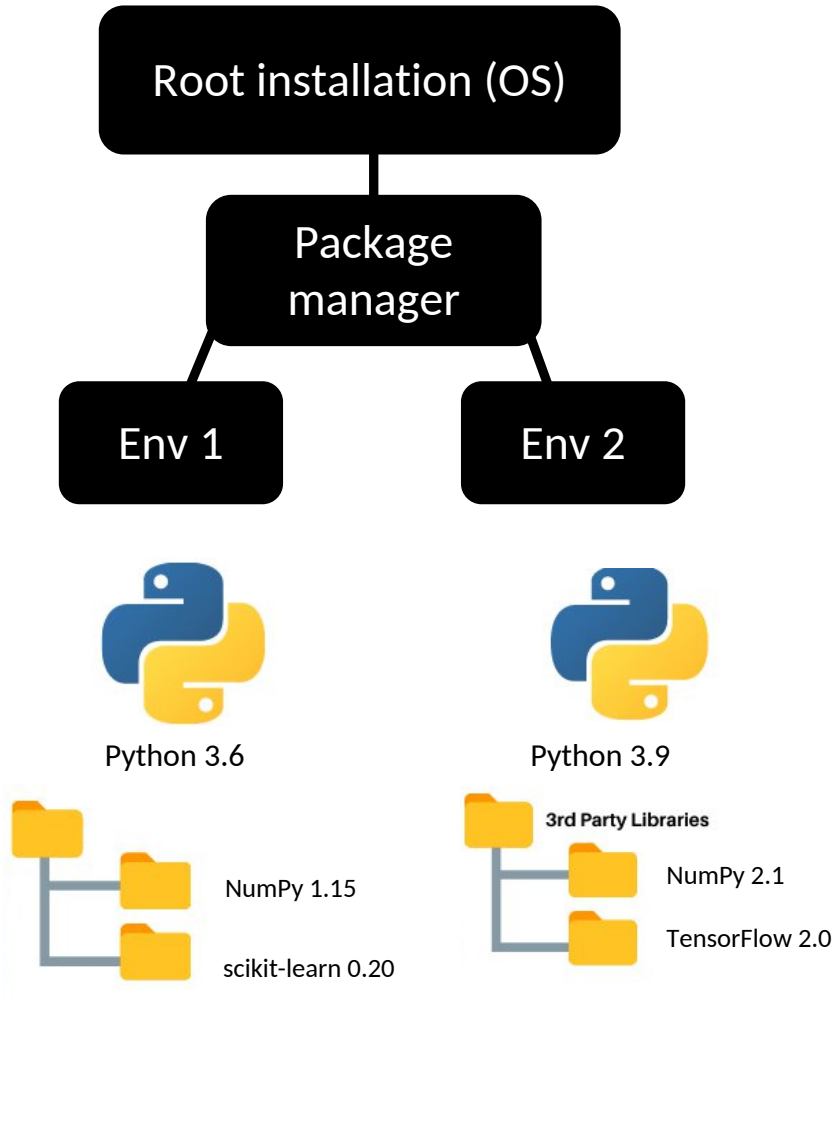
Managing software - environments

An **environment** is a directory with a specific set of installed packages and tools.

Changes made in one environment won't impact the others.

The user can manually build separate environments with **version conflict and/or circular dependency conflicts**

You can **easily activate or deactivate** environments to switch between them.



Isolated software environments

Local installations

↓

User

Packaging

automates the process of downloading, installing and configuring software tools and their dependencies

↓

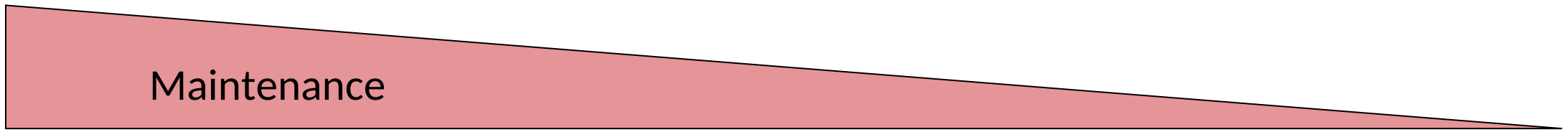
Package manager

Containerization

allows software tools, dependencies, system tools, system libraries and settings to be distributed and used in a platform-independent manner

↓

Container



Isolated software environments

Local installations

↓

User

Packaging

automates the process of downloading, installing and configuring software tools and their dependencies

↓

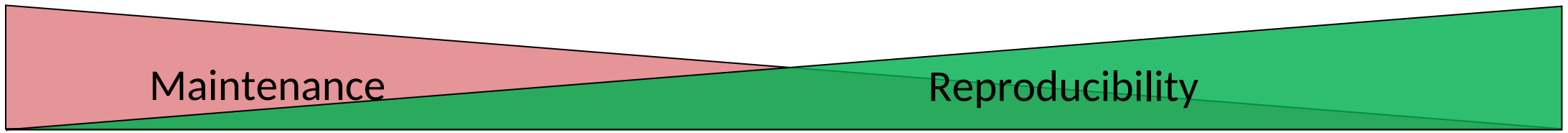
Package manager

Containerization

allows software tools, dependencies, system tools, system libraries and settings to be distributed and used in a platform-independent manner

↓

Container



Addressing the challenges using isolated software environments

Local installations

RDM

What happens if the program versions aren't available in 2/5/10 years? (Containerization!)

Packaging

↓

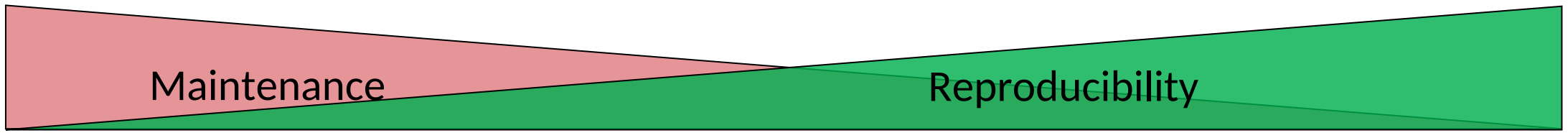
Package manager

Containerization

allows software tools, dependencies, system tools, system libraries and settings to be distributed and used in a platform-independent manner

↓

Container



Best practices - reproducible envs

Check platform documentation for where admin prefer you to install tools within your user directories / file system or if they'll install for you!

- Choose virtual environments, package managers or containers to manage software, package and dependencies
- One for each project! Isolate the project dependencies
- Document your environment setup
- Test the environment reproducibility
- Version control your environment configuration files
- Share reproducible environments with collaborators

Best practices - reproducible envs

	Interaction style		
What is reproduced?	Graphical	Command-line	
		Multi-language	Language-specific
Software & versions	ANACONDA NAVIGATOR binder	CONDA	renv
Entire computational environment	VirtualBox	Singularity	docker

Reproducibility

Interoperability

Package managers come in many flavours...

Package managers automate the process of installing, updating, configuring, and removing software packages

- **Language-Specific:** Some package managers are tailored to a specific programming language (e.g., pip for Python, CRAN for R).
- **Multi-Language Support:** Others, like Conda, support multiple languages and simplify dependency management across different platforms (Windows, macOS, Linux).

Name	Initial release	No. of tools	Type	Administrator permissions	Licensing	System requirements
CRAN (https://cran.r-project.org/index.html)	1997	18,660	Centralized	No	GNU GPL	CRAN is exclusively an R package manager, it requires R to be installed on the user's system
APT (https://wiki.debian.org/Apt)	1998	60,000	Centralized	Yes	GNU GPL 2+	For Debian based Linux systems
YUM (http://yum.baseurl.org/)	1999	—	Centralized	Yes	GNU GPL 2	For Red Hat based Linux systems
Bioconductor (https://www.bioconductor.or)	2001	2,083	Centralized	No	Open source	Bioconductor is exclusively an R package manager for genomic analysis
pip (https://pypi.org/project/pip/)	2008	348,149	Centralized	No	MIT License	pip is exclusively a python package manager
Homebrew (https://docs.brew.sh/Homebrew-on-Linux)	2009	5,916	Centralized	No	BSD 2-Clause Simplified License	Used mainly for macOS systems, but also works for Linux systems
EasyBuild (https://easybuilders.github.io/easybuild/)	2012	2,575	Both	No	GNU GPL 2	HPC environment
Conda (https://conda.io)	2012	2,587	Centralized	No	3-clause BSD license	For Windows, macOS and Linux, and for any programming languages
GNU Guix (https://www.gnu.org/software/guix/)	2013	19,729	Distributed	Yes	GNU AGPL	HPC environment
Spack (https://spack.io)	2013	6,139	Distributed	No	MIT License or Apache License	HPC environment

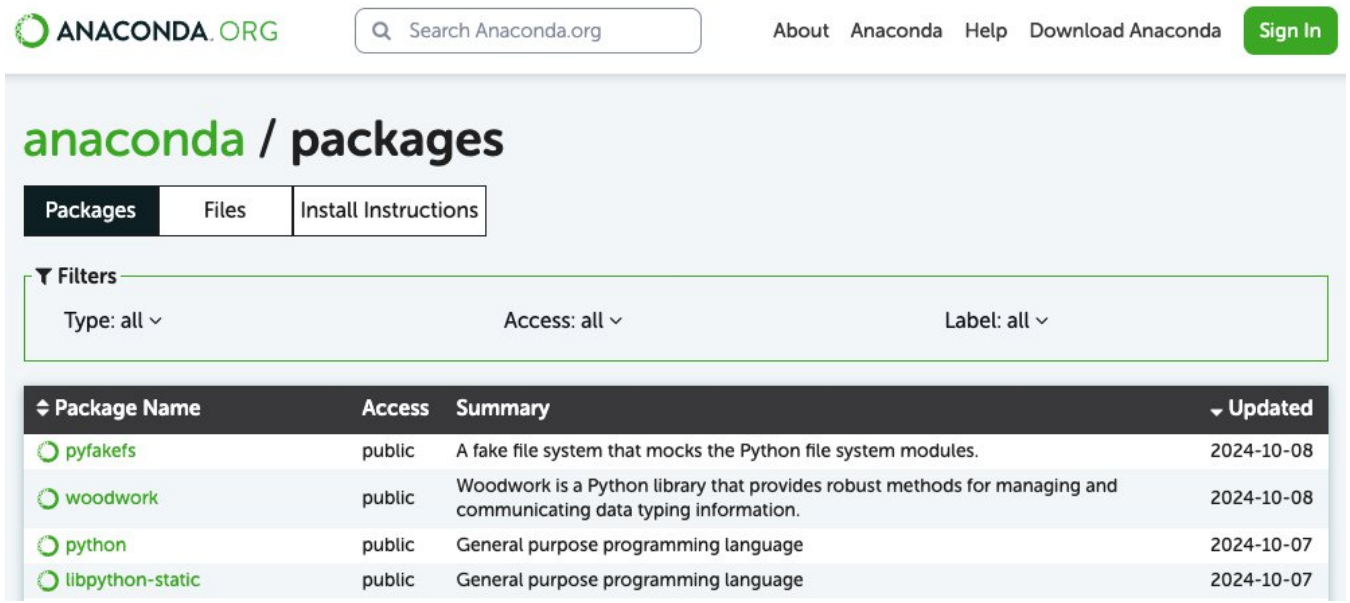
https://nature.com/articles/s41596-024-00986-0

Managing software – conda. WHY?

Cross-platform compatibility and programming **language agnostic**.

Easily install packages from a vast archive with a single command, eliminating concerns about compilers, dependencies, and binary locations.

Search for the package you need (**thousands of packages available**)



https://anaconda.org/anaconda/repo

Managing software - UCloud

- Every app comes with its pre-defined installed software on UCloud
- The Terminal app has no preinstalled (bio)software 
- Use **modules** (lmod) package and eb (easybuild)
- You can install and manage your software and its dependencies using virtual environments (conda)

Managing software - modules

List all software available:

```
[user@frontend ~]$ module list
```

Search for a particular package

```
[user@frontend ~]$ module avail 2>&1  | grep -i bowtie
```

Use a particular version of the software

```
[user@frontend ~]$ module load GCCcore/10.2.0
[user@frontend ~]$ module load bowtie2-2.3.1
```

Unload software

```
[user@frontend ~]$ module unload bowtie2-2.3.1
[user@frontend ~]$ module purge  #unloads everything
```

The module packages adds software to the user's PATH

Modules

```
#!/bin/bash
#SBATCH -N 1 -n 64
#SBATCH -p fat
#SBATCH -t 01:00:00

module purge
module load <list_of_modules>

# Add below some commands depending on the modules
```

Remember to include the module load command in your SLURM submission script for each package you want to use

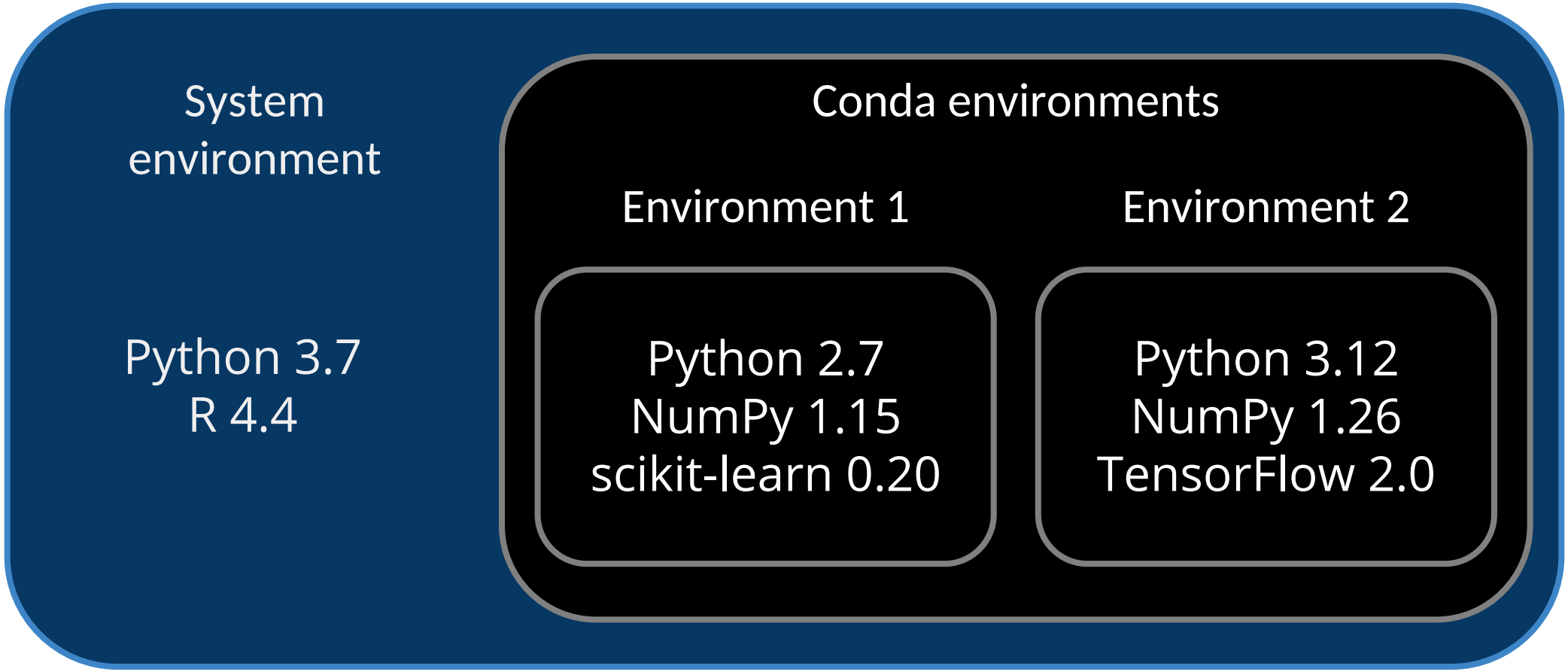
Advantages

- Ready and easy to use - no setup time
- No need to worry about installation dependencies, etc. HPC admins take care of this for you
- Software is compiled in an optimal way for the specific hardware of the HPC (may often increase speed and efficiency of the software)

Limitations

- If a software/version is not available, need to request it from the HPC admins, which may take some time.

Managing software - UCloud



Managing software - conda

Conda is a package manager with virtual environment functionality!

- easy to use and understand
- can handle quite big environments
- environments are easily shareable
- a large archive (Anaconda) of packages
- active community of people archiving their packages on Anaconda

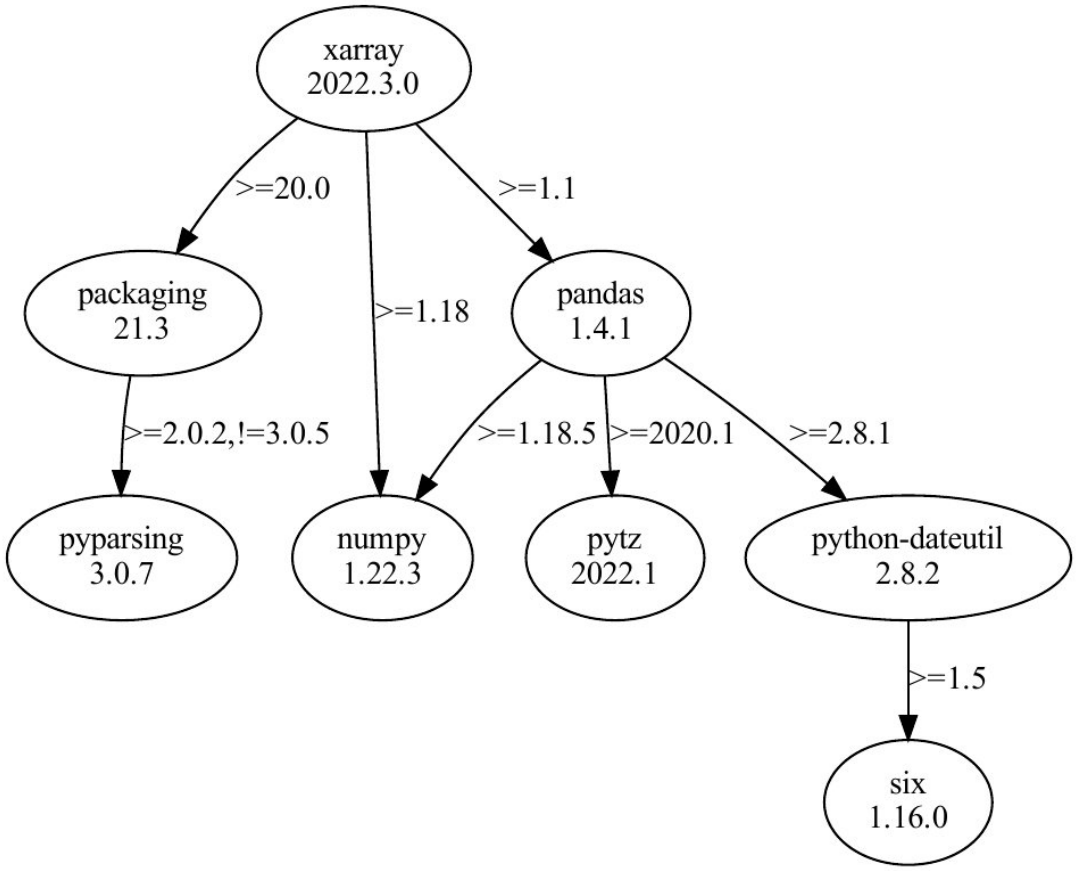
conda

Package management

- Install packages on a host without admin privileges.
- Each environment is installed locally and isolated from others.

Package installation

- conda assembles the dependency trees of requested packages to find all compatible dependencies versions (on the edges).



How does conda work?

Project reproducibility

- Ensure projects are self-contained and reproducible by capturing all package dependencies in a single requirements file.
- Resolve dependency issues by allowing different versions of a package for different projects.

myenv_1.yml

```
channels:
  - conda-forge
  - nodefaults
dependencies:
  - pandas==0.20.3
  - statsmodels==0.8.0
  - r-dplyr==0.7.0
  - r-base==3.4.1
  - python==3.6.0
```

How can I add & use a new environment?

Create new environment

```
> conda create --name mynewenv
> conda create --prefix /path/to/env/mynewenv
```

Activate my environment

```
> conda activate mynewenv
```

Install software

```
> mamba install -c conda-forge -c bioconda snakemake
```

Leave an environment

```
> conda deactivate
```

Remove environment perm

```
> conda env remove –n mynewenv
```

Dirs of the active environment's executable files are added to the system path (== easy access).

Summary of my environments

An environment is a folder, which contains all installed packages and other configurations and utilities.

List all available environments

```
gsd818@SUN1016678 ~ %  conda env list
> # conda environments
> #
> Base                  /home/username/miniconda3/
> myenv  *      /home/username/miniconda3/envs/myenv
```

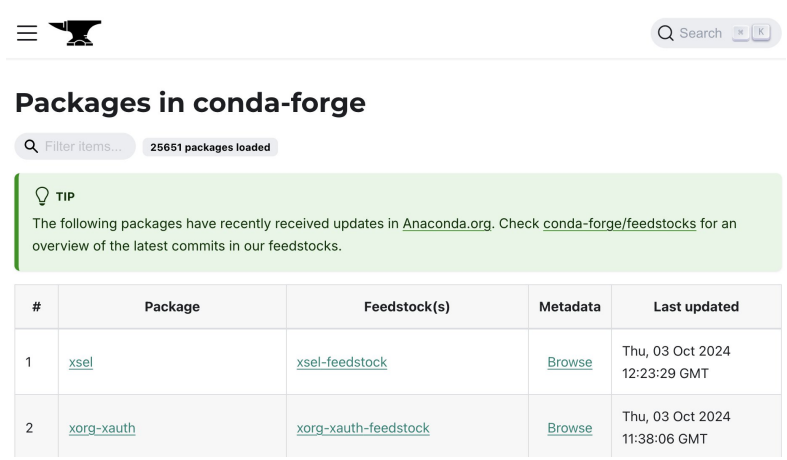
Active env

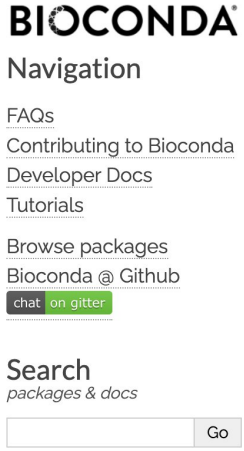
```
gsd818@SUN1016678 ~ %  conda list <PKG>
```

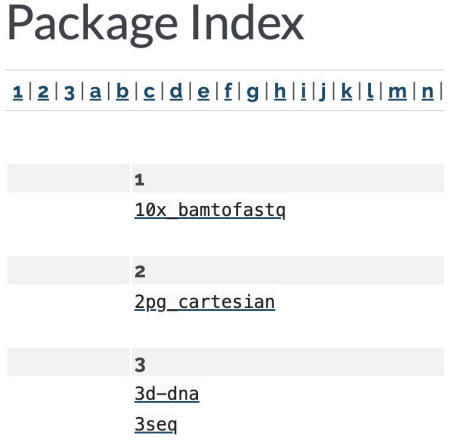
Popular channels

The software installation “recipes” used by conda are maintained by large communities of software developers. These communities are organised by channels, i.e. software repositories.

- **Conda-forge**: for scientific computing software
- **Bioconda**: for bioinformatics software (>8000 packages)







Append relevant channels

Bioconda is a community-enabled repository of 3,000+ bioinformatics packages, installable via the conda package manager.

Note: bioconda is not available for windows systems

Add channels:

```
> conda config --append channels bioconda
> conda config --append channels conda-forge
> conda config --append channels r
> conda config --append channels genomedk
```

conda config --add

checks if the value already exists and only adds it if it’s not present.

Avoids duplication

Faster for regular updates

conda config --append

simply adds the value at the end of the list without checking for duplicates.

Prioritizing channels: channel is added at the end of the list, giving it a lower priority (regardless if it exists)

Debugging

Install specific package's version

List out all the installed packages in the currently active environment:

(t)> conda list

Search for all available versions of a certain package

> conda search samtools

```
Loading channels: done
# Name                  Version           Build  Channel
samtools                0.1.12                0  bioconda
samtools                0.1.12                1  bioconda
samtools                0.1.12                2  bioconda
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
samtools                1.9           h91753b0_3  bioconda
samtools                1.9           h91753b0_4  bioconda
samtools                1.9           h91753b0_5  bioconda
```

Specify the version to install

> conda install samtools=1.9

> conda update

> conda remove

When to use a Package Manager?

- Same HPC platform (sharing OS and architecture)
- Quickly install and manage software
- Projects are relatively simple and do not require complex isolation
- To conduct repetitive data analyses using a pipeline

Conda cheat sheet

Advantages

Easily search for packages in a centralised repository/registry (e.g., Bioconda)

Do not require administrative privileges to install software & manage dependencies automatically

Can encapsulate different software version in environments

Allow recreation of the environment on other systems

Limitations

Not every software is available through Conda

Cannot address complex dependency conflicts (circular dependency and version conflict)

Can take a lot of space (tip: run conda clean to remove unused and cached packages)

Reproducibility: some packages may have different versions or may not be available on other operating systems and architectures.

Pixi

Pixi

next-gen package manager for reproducible development setups

New virtual environment and package manager

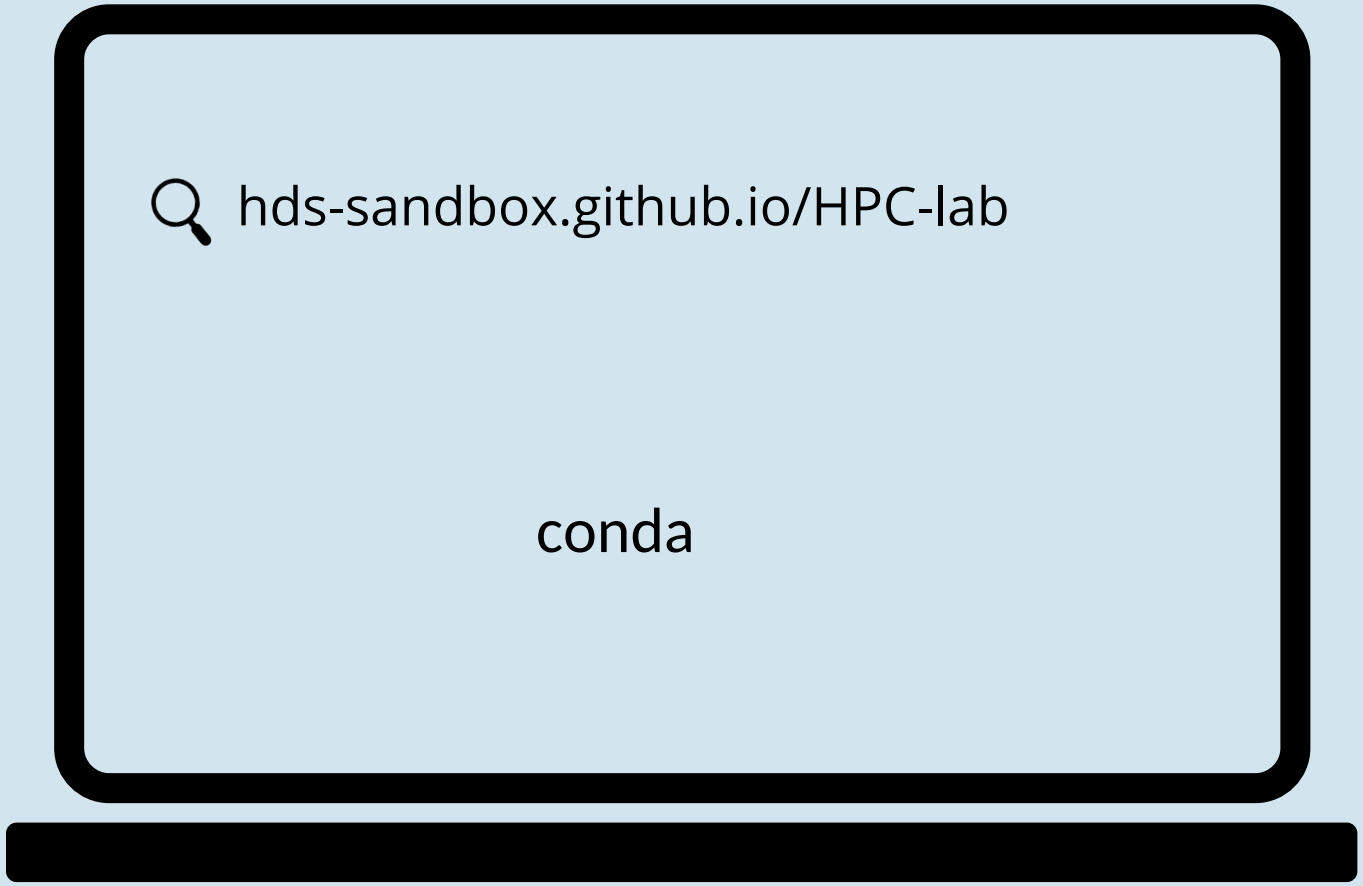
An upgrade of Conda in speed and stability

Can install the same packages as conda

Easily switch between Conda and Pixi using YAML files

> pixi add rstudio-desktop jupyterlab r-ggplot2 r-dplyr pandas

pixi.toml contains info pixi uses to create the environment



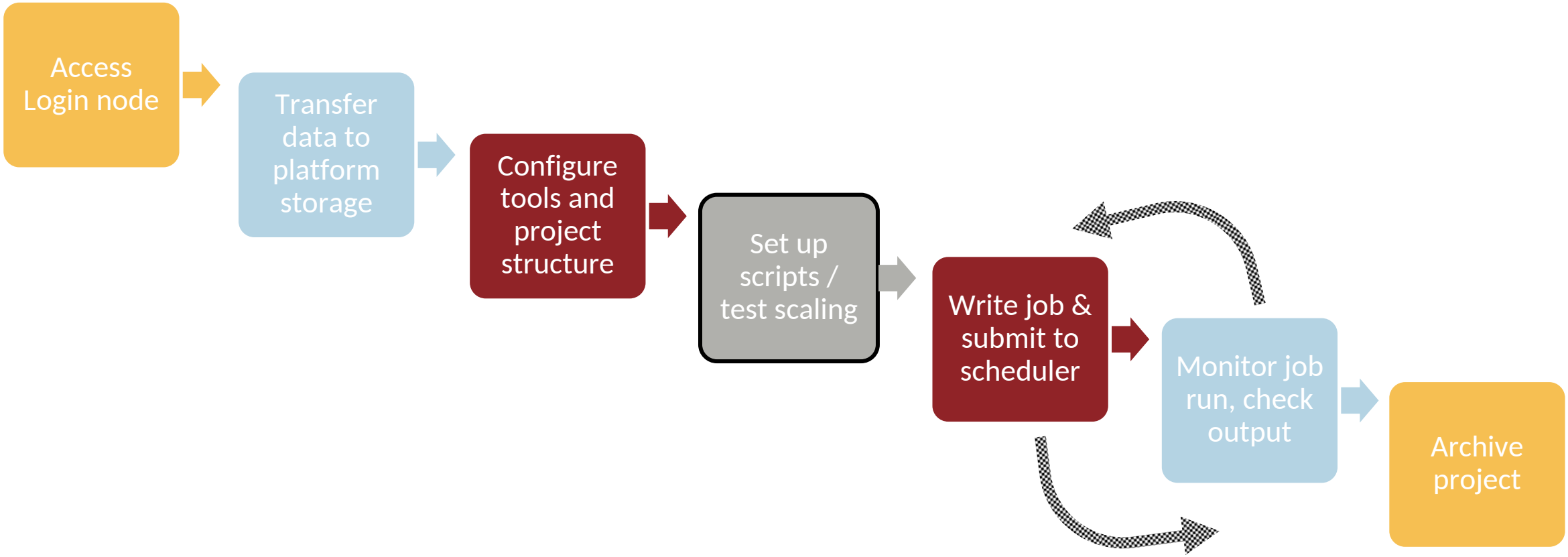
<project>_<analysisMethod>_<description>_<YYMMDD>_<vX>.ext

~ 15 min

Now is YOUR time!

Day 2 – Managing software

Standard HPC workflow



Managing computation (1)

- **Always test before scaling.** Run small, uncoupled tests before a big job
- **Benchmark cores, memory, and runtime (optimize!)**
  - Identify scaling limits
    - Plot runtime vs. number of cores to identify the saturation point
- **Make scripts reproducible and parameterized**
  - No hardcoding paths: inputs, outputs, and resources (e.g., paths, threads, memory) should be passed as arguments
  - Use package managers and/or containers to import software

Managing computation (2)

- **Validate output** on small data before scaling. Submit one job as a test and make sure everything works as expected before submitted a large number of jobs.
- **Log resource usage** inside the job (e.g., time, memory snapshots)
- Make your **code style and formatting consistent**
- Test failure scenarios (e.g., what happens if job stops halfway?)

Write programs/code for people, not computers

Scalable | Reproducibility | Interoperability

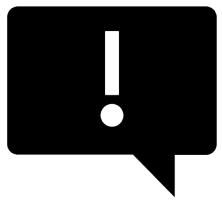
Use resources efficiently

What can go wrong?

A small bug in a script can waste tons of computational resources. Imagine processing 1000 samples where each sample requires 4 hours on a single CPU core. Even with 4 cores running in parallel, the analysis could still take more than a month.

Now imagine waiting days in the queue only for the job to fail within seconds because of a typo, incorrect parameter, a wrong reference file or a broken path.

This can be extremely frustrating and waste both time and resources.



Test before scaling

Remember you (or someone else) are generally charged for usage on HPC

Best practices: reproducible data analyses

HPC-pipes

- Annotate your pipelines and comment your code
- Version control your code and pipelines
- Make your source code accessible (GitHub, GitLab…)
- Parameterize your pipelines (flexible and reusable)
- Write pipelines using workflow mgmt. systems to streamline and automate various steps in your data analyses (Snakemake, Nextflow, etc.)

Scalable | Reproducibility | Interoperability

Best practices: community-curated workflows

Community-curated analysis pipelines (sequences of benchmarked tools)

Common languages & styles

Well-documented, validated, portable and highly optimised

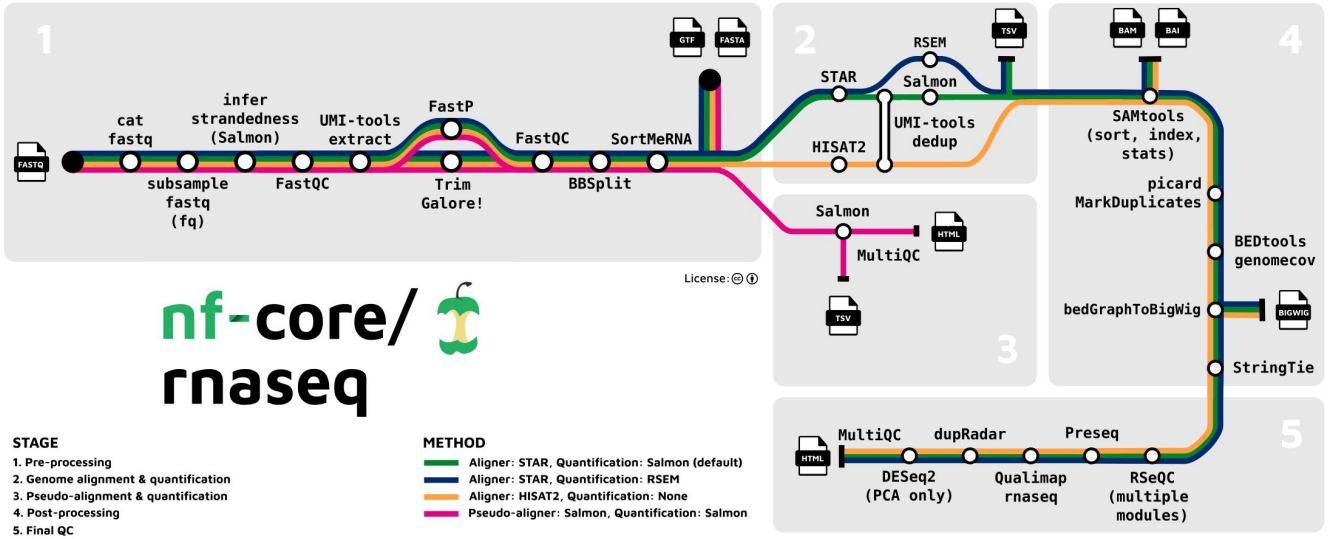
Work on any computational infrastructure

Open source

Starter template

Reproducibility

Parallelisation



Best practices: analysis reports

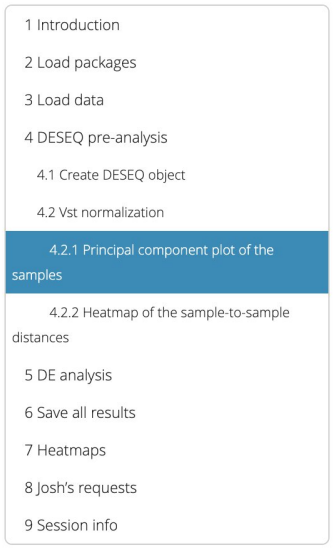
Notebooks

GitHub pages for docs
(i.e. Quarto/MkDocs)

Integrate:

- Text
- Code (including R Markdown & Jupyter Notebooks)
- Visualizations

Collaboration



4.2.1 Principal component plot of the samples

PCA plot using the first two components

```
pcaData <- plotPCA(vsd, intgroup=c("ShortLabel"), returnData=TRUE)
percentVar <- round(100 * attr(pcaData, "percentVar"))
```

```
ggplot(pcaData, aes(PC1, PC2, color=ShortLabel, label=ShortLabel)) +
  geom_point(size=3) + geom_text_repel() +
  xlab(paste0("PC1: ",percentVar[1],"% variance")) +
  ylab(paste0("PC2: ",percentVar[2],"% variance")) +
  coord_fixed() + theme_bw()
```

Documentation

READMEs & metadata

- Include all necessary information to reproduce an experiment/analysis
- Cookiecutter template compels consistent metadata recording == standardization (variables)
- Package the metadata with the data itself
- Use ontology online services

```
# README.md

## General info
This folder contains…

## Aims

## Organization
Folder structure,
abbreviations,
file formats

 Data
(t)(t)|  - Organism.1
|  | - Raw
|  | - Processed
|  | - readme.txt

File naming convention:
<AAA>_<BBB>_<YYMMDD>.ext

## Data
Variables and content
Data collection or re-use
```

Metadata field	Convention	Example
assay_type	-	ChIP-seq
owner	<Initials>	JARH
creation_date	<YYYYMMDD>	20231108
platform	-	Illumina
organism	<Genus species>	Homo sapiens
nsamples	<integer>	9

RDMkit

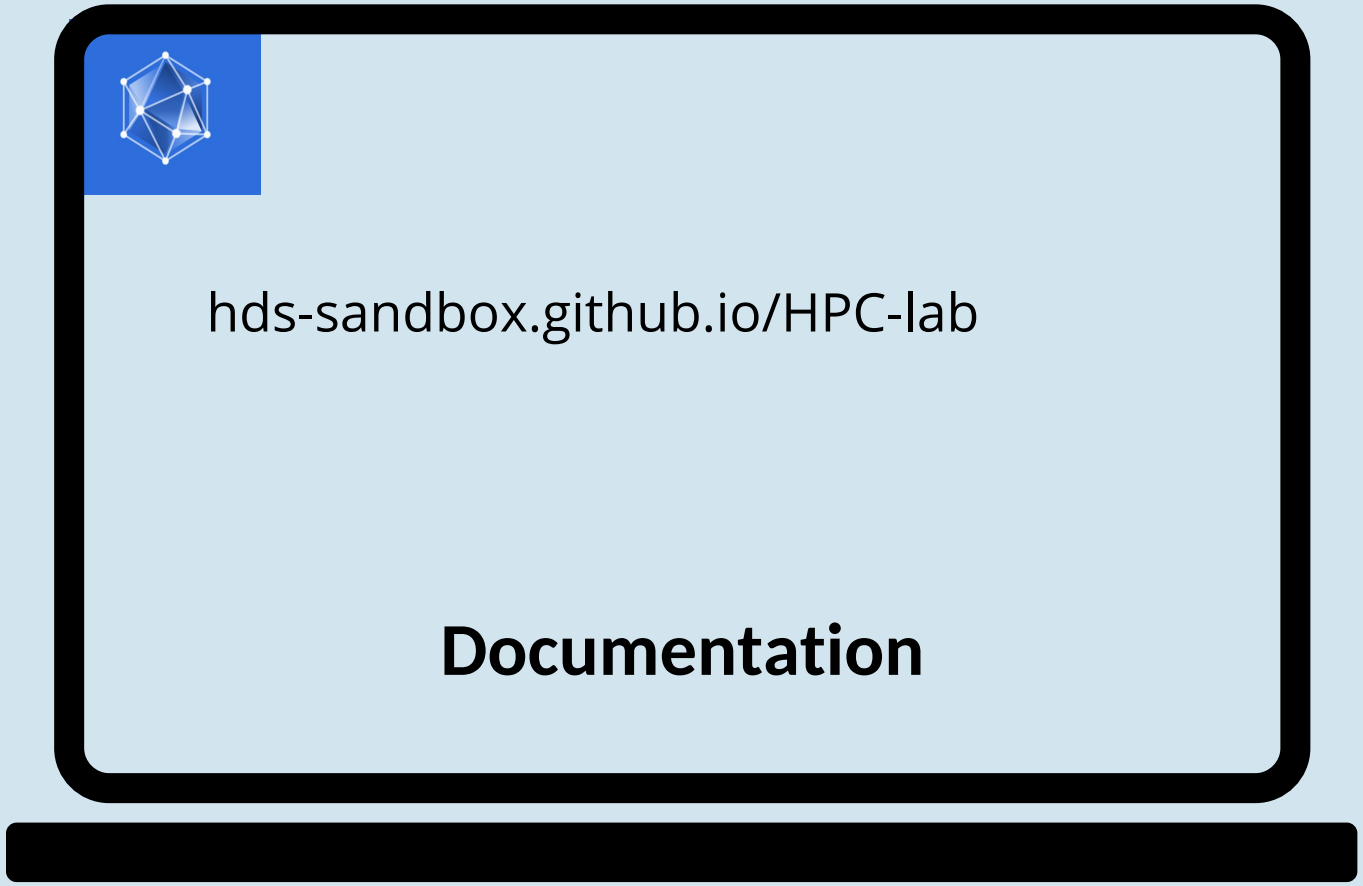
FAIRsharing.org
standards, databases, policies

Data context and content

Without metadata

With metadata

Document for future use. This applies to code, software, data…

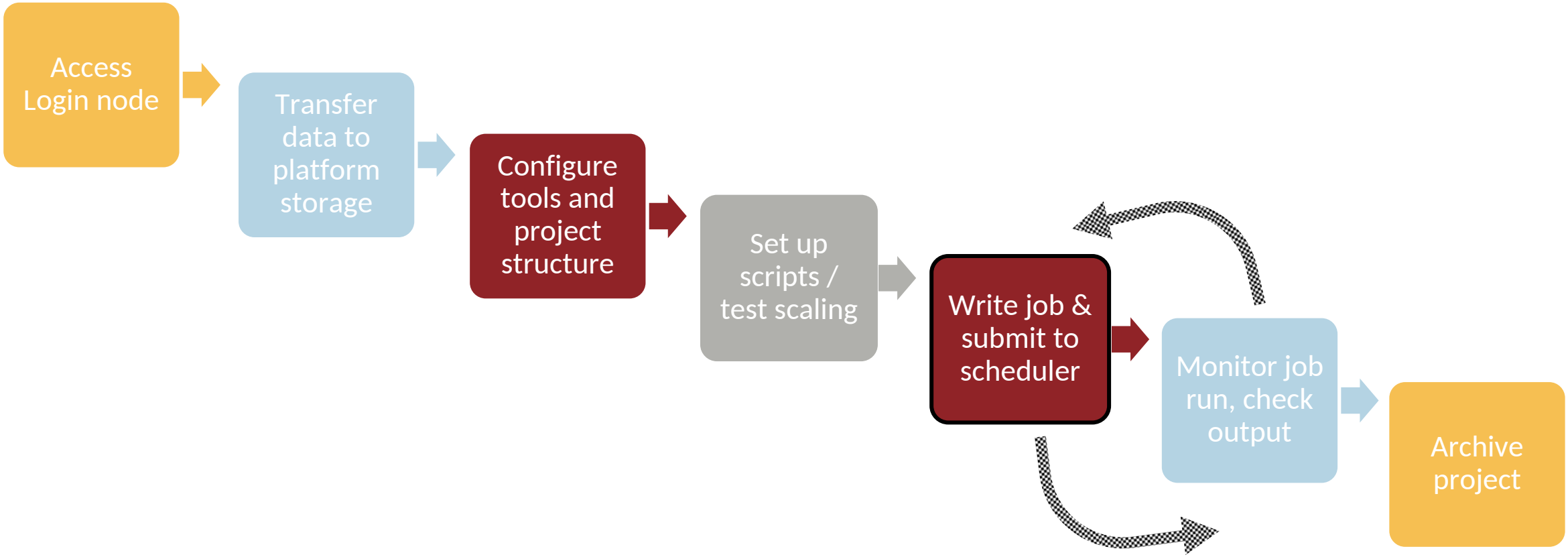


~ 5 min

Now is YOUR time!

Discuss how a README.md conveys key details about the project and how would you structure your own

Standard HPC workflow



Job submission

SLURM is the software that manages the job queue:

- decides **when each jobs runs**, depending on the availability of compute nodes and the resources requested for that job
- Queue position depends on account priority and **requested resources** (the more CPUs/memory/time you ask for, the further back in the queue you start)
  - Benchmark – request what you need!
- Your “queue position” improves the longer you are in the queue (so that it eventually gets scheduled)

Job submission workflow

1. Write your commands in a shell script
2. Configure your job options using #SBATCH directives
3. Use $SLURM_* environment variables to further customise your commands
4. Test your code by requesting a terminal on a compute node using srun/salloc (interactive jobs)
    - salloc: allocated resources, gives you a shell to run commands
    - srun: simultaneously requests the allocation and subsequent launch of tasks, processes, or parallel applications

Running jobs on an HPC

Single computer environment (interactive use):

```
$ bowtie2 –x ref_index -1 reads_1.fastq -2 reads_2.fastq
```

HPC (using scheduler):

```
$ sbatch RESOURCE_REQUEST mapping.sh
$ squeue
 JOBID PARTITION  NAME     USER   ST   TIME  NODES NODELIST
 2048  highmem    mapping  ht123  R    0:02  1     highmem-node01
```

Best practices: HPC usage

How do I avoid over-allocation?

How do I find how much memory my job has used?

- Request only the resources you actually need
- Avoid submitting large numbers of jobs at once without throttling

Best practices: HPC usage (1)

Optimize your jobs for CPU and time usage

- Analyze resource usage: sacct <jobid>
  - Increase #nodes while keeping the same problem size (if/when it doesn't have a very significant effect, reduce again)
  - Start with a small memory allocation, checking the actual usage, then adjusting the memory allocation for subsequent runs

Best practices : HPC usage (2)

- Job priority – be nice to your fellow users (nice)
- Choose the node type or cluster queue (if applicable)

```
#SBATCH --constraint=hm1
```

UCloud
hm1: huge memory, I/O-heavy
hm3: CPU-heavy tasks, fastest runtime

- Create universal and software-specific submission scripts (but never sample specific)
- Use scratch directories for temporary files (or cleanup after job completion)

Best practices: HPC usage (3)

Run multiple jobs (e.g. samples) in **parallel**

- Build up dependency chains

```
job1=$sbatch step1.sh | awk '{print $4}')
sbatch --dependency=afterok:$job1 step2.sh
```

Print jobID
--parsable

- Control number of running jobs via job array

```
sbatch --array=1-10 step1.sh
```

- Using workflow systems

Logging: specify the location for standard output and error

```
#SBATCH --output=logs/%x_%j.out
#SBATCH --error=logs/%x_%j.err
```

%x → job name
%j → job ID

Best practice: logging (in general)

Terminal output is stored in log files **std.err** & or **std.out** depending on your setup

Log file stores info from your run:

- tool/job exit status
- faults, errors, failures, and abnormal events
- job parameters, allocated resources and resource utilization >> ‘benchmark.txt’

Tools may also provide output logs that can be written to text files (bash scripting)

```
python analysis.py > output.log 2> error.log
```

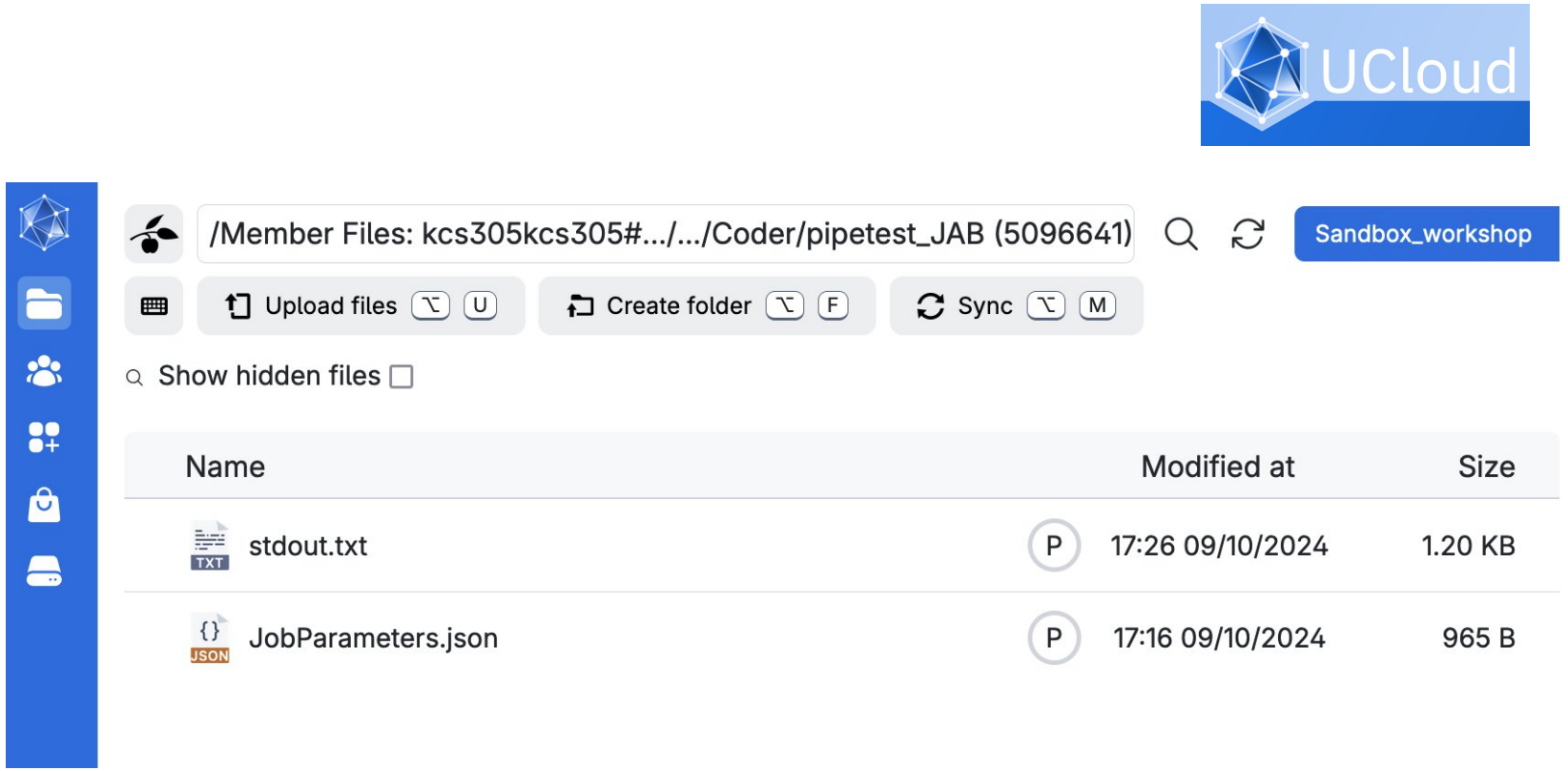
&>: saves all output

Logging - example

```
> cat align.sh-45668367.out

Looking to launch executable "/home/samuele/miniconda3/envs/gwf_workshop/bin/bwa-mem2.avx2", simd = .avx2
Launching executable "/home/samuele/miniconda3/envs/gwf_workshop/bin/bwa-mem2.avx2"
[bwa_index] Pack FASTA... 1.02 sec
* Entering FMI_search
init ticks = 17879756192
ref seq len = 486398746
binary seq ticks = 7924387248
build suffix-array ticks = 125574135569
ref_seq_len = 486398746
count = 0, 144839395, 243199373, 341559351, 486398746
BWT[207461114] = 4
CP_SHIFT = 6, CP_MASK = 63
sizeof CP_OCC = 64
pos: 60799844, ref_seq_len__: 60799843
max_occ_ind = 7599980
build fm-index ticks = 32653281302
Total time taken: 79.5864
Looking to launch executable "/home/samuele/miniconda3/envs/gwf_workshop/bin/bwa-mem2.avx2", simd = .avx2
Launching executable "/home/samuele/miniconda3/envs/gwf_workshop/bin/bwa-mem2.avx2"
ERROR: unknown command '-t'
[W::hts_set_opt] Cannot change block size for this format
samtools sort: failed to read header from "-"
```

Logging on UCloud apps

- UCloud automatically outputs stdout.txt for each job run through an app
- It also provides a .json of input job parameters



~ 1 min

Start a Terminal app job, following the configs on the website!

SLURM - key commands

sbatch: submit a shell script to the queue

squeue: see all the jobs in the queue

squeue -u USERNAME: see your jobs only

scancel JOBID: cancel the job with the specified ID

scancel -u USERNAME: cancel all your jobs

Commands on job monitoring in a few slides

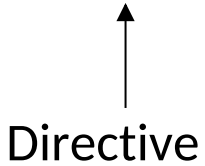
SLURM – job submission example

Submit job

```
[user@frontend ~]$ sbatch mapping.sh
```

You can add additional options, or to override options specified in the job script using directives

```
[user@frontend ~]$ sbatch --time 4:00:00 mapping.sh
```




```
> man sbatch
```

SLURM - directives

--time, -t: Set wall time (keep it low → faster scheduling)

--nodes,N: Number of nodes (use ranges like 2-4 for flexibility)

--ntasks-per-node: MPI processes per node (e.g., 1 per core)

--mem-per-cpu: Memory per CPU (request only what you need)

--account, -A: Billing account (only needed if multiple projects)

--exclusive: Reserve full node (charged for entire node)

--output, -o: batch script's standard error

--error, -e: batch script's standard error

--requeue: Auto-restart if preempted or fails

--job-name, -J: Custom job name (visible in queue)

SLURM – example mapping.sh

mapping.sh

```
#!/bin/bash

#SBATCH –-job-name mapping
#SBATCH --nodes 1                # nodes
#SBATCH --ntasks-per-node 1      # cores per node
#SBATCH --time 2:00:00           # max time (HH:MM:SS)
#SBATCH --output=logs/%x_%j.out
#SBATCH --error=logs/%x_%j.err



module load bowtie2          # Load required modules (or activate env)

bowtie2 –x ref_index -1 reads_1.fastq -2 reads_2.fastq
```

Include your job submission details as #SBATCH

Command/script you want to run

SLURM – example myscript.sh

```
#!/bin/bash
#SBATCH --job-name=myjob
#SBATCH --output=myjob.out
#SBATCH --error=myjob.err
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=8G    # memory per cpu-core
#SBATCH --time=01:00:00
#SBATCH --mail-type=begin       # send email when job begins
#SBATCH --mail-type=end         # send email when job ends
#SBATCH --mail-type=fail        # send email if job fails
#SBATCH --mail-user=your mail address

export TMPDIR=/scratch/     # Set up your temp dir (if necessary)

mycommand
```

SLURM – example myscript.sh

Submit job as is

```
sbatch myscript.sh
```

Override the options in the job script. Use the gpuqueue queue instead and request 1GPU resource

```
sbatch --partition=gpuqueue --gres=gpu:1  myscript.sh
```

Portable Batch System - Computerome

Supports the same functions as Slurm, just with different commands!

- **Slurm:** job starts in submission directory by default (sbatch)
- **Torque/PBS:** often starts in /home/login directory unless changed manually (qsub)

```
#PBS -W group_list=<group>
#PBS -A <account>
#PBS -l nodes=1:ppn=40,mem=120g,walltime=01:00:00
#PBS -j myjob
#PBS -o job.out
#PBS -e job.err

mycommand
```

Use qstat, checkjob, and showq to monitor job state.

https://computerome.dk/wiki/user-guide#submit-jobs-to-hpc-cluster

Best practices: Interactive jobs

- Useful to run **non-repetitive** tasks interactively
- Jobs **limited in time**
- Once you exit the job, anything running inside will stop

Examples:

- Testing/writing code
- Compress/decrompressing files
- Opening Rstudio/Jupyterlab (e.g. plotting)

You will need to specify the memory and cores usage, and the time. Wait in the queue until your resources have been allocated.

Jobs comparison

Interactive jobs

- Run commands directly on allocated compute resources from an active terminal
- Provide **realtime output and immediate interaction**
- Best for **testing** and **debugging**
- Usually require you to stay connected/logged in (tmux)

Batch jobs

- Run computations through a submitted job script
- Execute independently without active user interaction
- Well suited for **long-running or parallel workloads**
- Output is typically written to log files, making debugging less interactive

Interactive jobs – How to?

Request 2 CPUs, 4 GB and 1h walltime, use the following command:

```
srun --cpus-per-task=2 --mem=4G --time=1:00:00 --pty bash
```

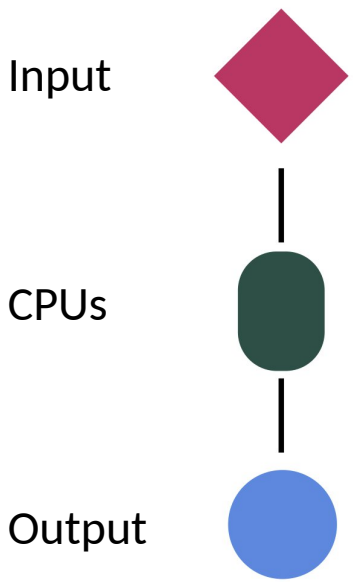
When the compute node is allocated, you will be logged in to the node with a bash shell. You can run your commands here as you would in a terminal.

DO kill interactive jobs when you've finished with them (exit)

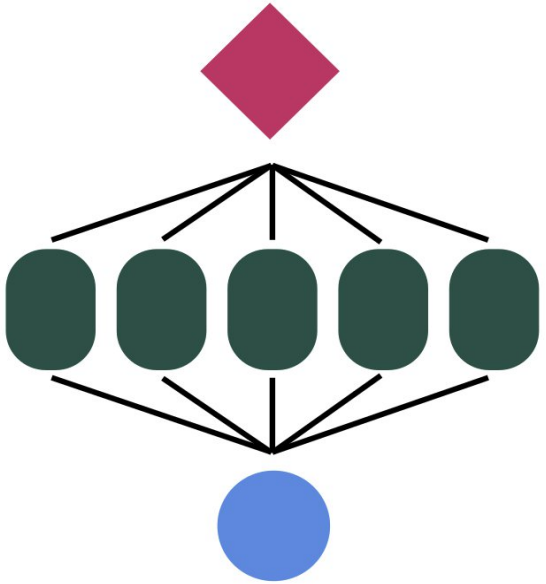
Running jobs in parallel

Submitting large quantities of jobs and/or submitting jobs with long runtimes may occupy resources for hours and days. Be mindful of other users in the cluster!

Running multiple jobs (e.g. samples):



1 input file, 5h
100 files, ~21 days



1 input file, 5h
100 files, ~4 days running serially

Does the software support multi-threading?

Parallel programming

- Parallel programming allows applications to take advantage of parallel hardware
- The queuing system facilitates executing parallel tasks
- Performance improvements from parallel execution do not scale linearly

Not all the work can be performed in parallel,
some of the processing/calculations will be serial

- One way of predicting improvements in execution time of a fixed workload: Amdahl's Law

Speedup factor (S) $$S(t_n) = \frac{t_1}{t_n}$$

https://carpentries-incubator.github.io/hpc-intro/17-parallel.html

Parallel programming

In practice, it's common to evaluate the parallelism of an MPI program by

- running the program across a range of CPU counts,
- recording the execution time on each run,
- comparing each **execution time to the time when using a single CPU**.

> The **Message Passing Interface (MPI)** is a set of tools which allow multiple tasks running simultaneously to communicate with each other

If you want to learn more, I would recommend to read "HPC parallelism for novices"

https://www.hpc-carpentry.org/hpc-parallel-novice/

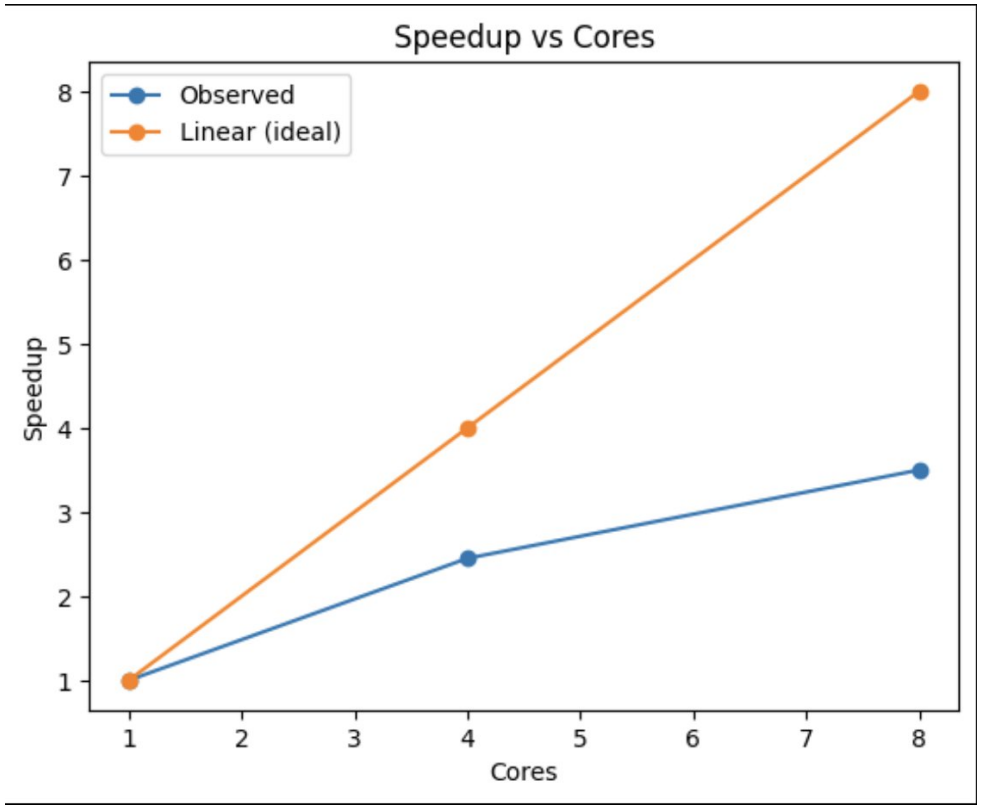
https://carpentries-incubator.github.io/hpc-intro/17-parallel.html

Parallel programming

Speedup factor (S)

$$S(t_n) = \frac{t_1}{t_n}$$

When would S = n?

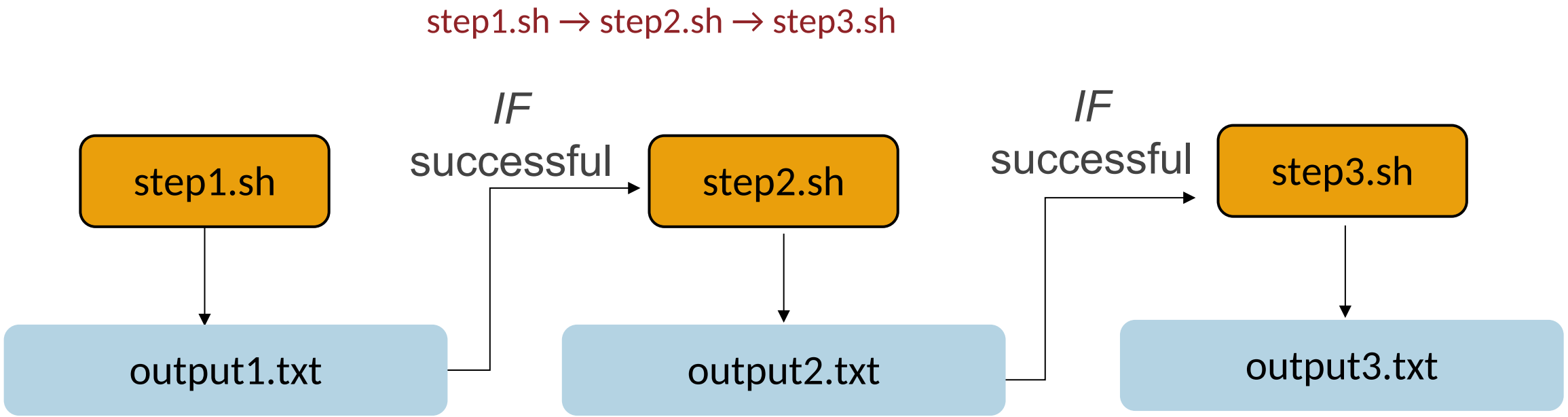


# of CPUs	Runtime (sec)	Speedup
1	33.326056	1
4	13.579746	2.45
8	9.514393	3.5

https://carpentries-incubator.github.io/hpc-intro/17-parallel.html

Job dependencies - afterok

Linear pipeline where each script depends on the output of a previous script:



Job dependencies - afterok

Linear pipeline where each script depends on the output of a previous script:

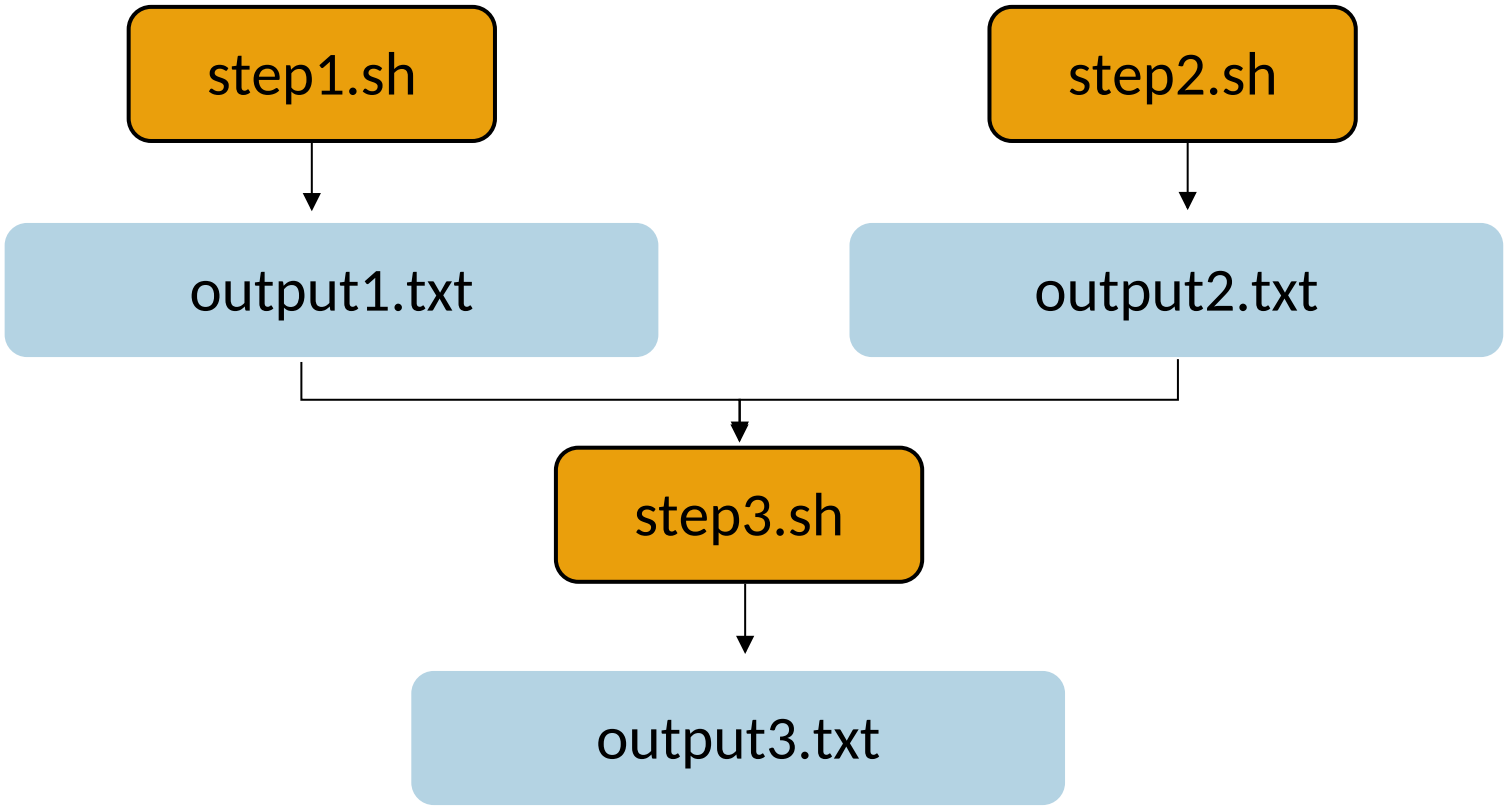
mypipeline.sh

```
job1=$(sbatch --parsable step1.sh)          # capture JOBID into a variable
job2=$(sbatch --dependency=afterok:$job1 --parsable step2.sh)
sbatch --dependency=afterok:$job2 step3.sh
```

> bash mypipeline.sh

Job dependencies

Other types of job dependencies (e.g.: afternotok starts the job only if another job fails or singleton use for tasks with multiple dependencies)



Job arrays

Job submission is quite fast and the job array is handled by SLURM as one, while each individual job is handled independently

All parameters (–cpus-per-task, –mem-per-cpu, –time, etc.) will be set for every individual job running (limitation)

Monitor the progress of the batch array:

```
sacct -j <jobid> (t)\
        --format=JobID,JobName,Partition,AllocCPUs,State,ExitCode,Elapsed
```

Job arrays - job submission example 1

```
#!/bin/bash
#SBATCH --job-name=myjob
#SBATCH --output=myjob.%A.%a.out
#SBATCH --error=myjob.%A.%a.err
#SBATCH --array=1-10%3     # a range of indices (1-10, max 3 parallel)
#SBATCH --time=00:10:00
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=10

echo "Processing input file input_${SLURM_ARRAY_TASK_ID}.txt"
python process.py input_${SLURM_ARRAY_TASK_ID}.txt
```

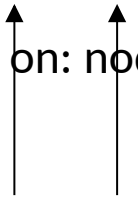
%A: array job ID (for the whole array)
%a: array task ID

environment variable

Job arrays - job submission (2)

3 jobs in parallel

SLURM_ARRAY_TASK_ID = 1	SLURM_ARRAY_TASK_ID = 2	SLURM_ARRAY_TASK_ID = 3
4 cores	4 cores	4 cores
Processing input file input_1.txt…	Processing input file input_2.txt…	Processing input file input_3.txt…
myjob.12345.1.out	myjob.12345.2.out	myjob.12345.3.out
Running on: node-hm1	Running on: node-hm2	Running on: node-hm1



%A: array job ID (for the whole array)
%a: array task ID

Job arrays – example 2

Input sample sheet (e.g. CSV)

```
sample,input
patient1,data/XYZ10231.fq
patient2,data/XYZ19381.fq
```

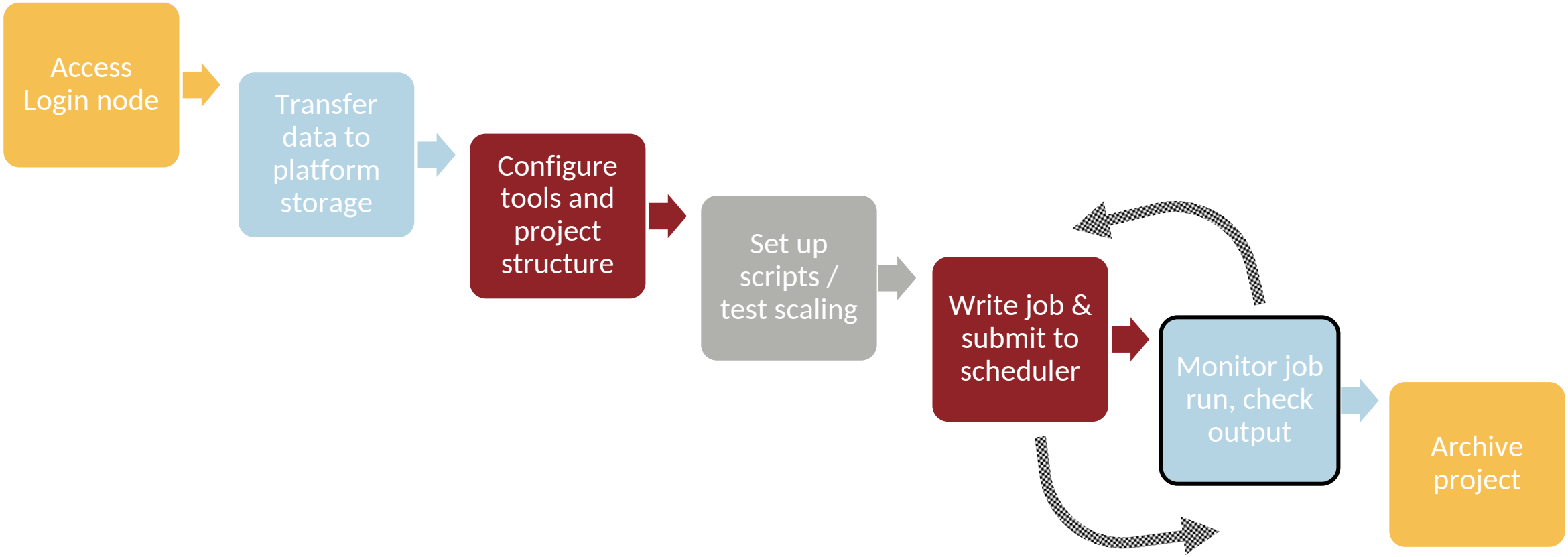
```
#SBATCH -a 2-3
#SBATCH
#...
```

Include job submission details

```
SAMPLE=$(head -n $SLURM_ARRAY_TASK_ID samplesheet.csv | tail -n1 | cut -d "," -f1)
INPUT=$(head -n $SLURM_ARRAY_TASK_ID samplesheet.csv | tail -n1 | cut -d "," -f2)
```

```
mycommand --input ${INPUT} --output results/${SAMPLE}.out
```

Standard HPC workflow



Monitor using scheduler - jobs

```
$ sbatch RESOURCE_REQUEST mapping.sh
$ squeue
 JOBID PARTITION  NAME     USER   ST   TIME  NODES NODELIST
  2048  highmem    mapping  ht123  R    0:02  1     highmem-node01
```

$ squeue -u your_username # all jobs submitted by you

$ squeue -j job_id # status of one job

$ jobinfo job_id

$ scontrol show job # detailed job info

$ sacct –u your_username # summary of completed jobs

$ sacct -j <id> --format=JobID,JobName,Partition,AllocCPUs,State,ExitCode,Elapsed # job arrays

Monitor using scheduler – node

sinfo –p <partition-name>

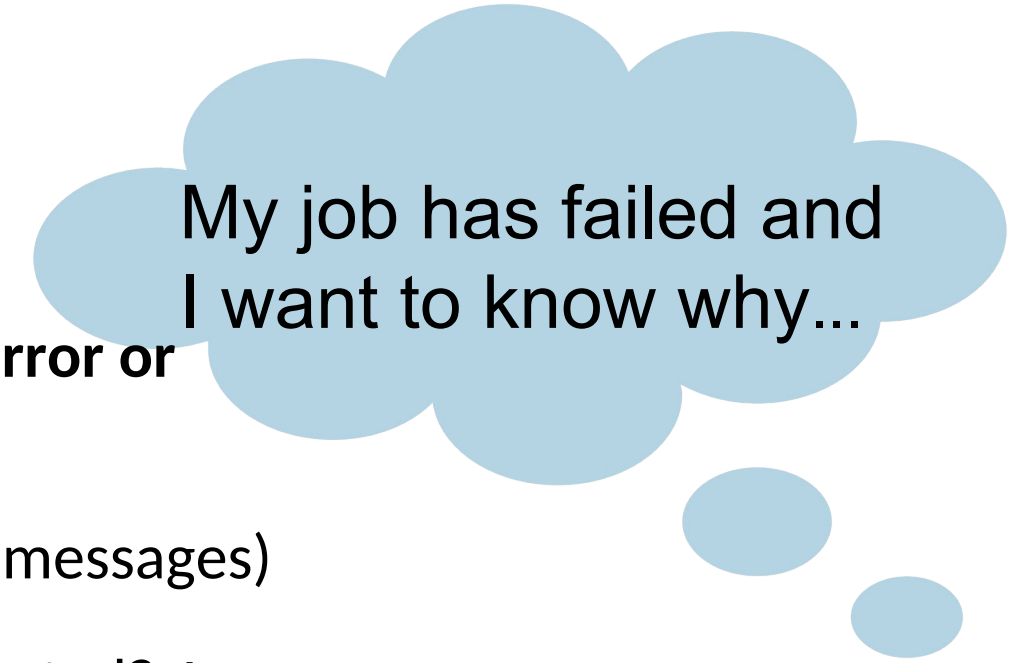
sinfo -N <node_name>

```
[vaduaka@discovery-l1 test]$ sbatch script.sh
Submitted batch job 8859
[vaduaka@discovery-l1 test]$ srun script.sh &
[1] 6840
[vaduaka@discovery-l1 test]$ squeue -u vaduaka
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
              8860    normal script.s  vaduaka  R       0:08      1 discovery-c4
              8859    normal     Test  vaduaka  R       0:17      1 discovery-c4
[vaduaka@discovery-l1 test]$ scancel 8859
[vaduaka@discovery-l1 test]$ squeue -u vaduaka
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
              8860    normal script.s  vaduaka  R       0:42      1 discovery-c4
[vaduaka@discovery-l1 test]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
normal*      up 7-01:00:00      2    mix discovery-c[3-4]
normal*      up 7-01:00:00     10  alloc discovery-c[1-2,8-15]
normal*      up 7-01:00:00      3   idle discovery-c[5-7]
gpu          up 7-01:00:00      1   idle discovery-g1
debug        up    1:00:00      2    mix discovery-c[3-4]
debug        up    1:00:00     10  alloc discovery-c[1-2,8-15]
debug        up    1:00:00      4   idle discovery-c[5-7],discovery-g1
backfill     up 14-02:00:0      5    mix discovery-c[3-4,19-20],discovery-hhm1
backfill     up 14-02:00:0     10  alloc discovery-c[1-2,8-15]
backfill     up 14-02:00:0     23   idle discovery-c[5-7,16-18,21-25],discovery-g[1-11],discovery-hm1
[vaduaka@discovery-l1 test]$
```

Cancelling jobs

```
scancel JOBID1,JOBID2,JOBID3 # multiple

scancel -u USERNAME # all submitted by user

scancel 123456 # cancel jobID of the array (entire array)

scancel --partition=PARTITION_NAME # all running on specific partition

scancel --nodelist=NODE_NAME
```

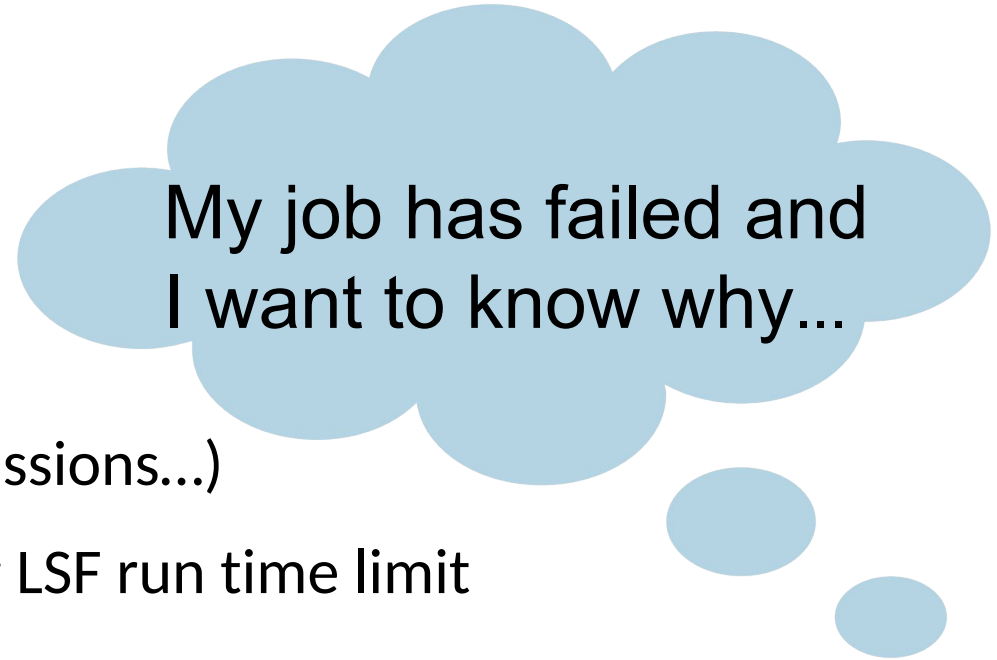
Checking output



Make sure you run your job using a **standard error or standard output**

- Check logging or debug print (look for error messages)
- Is the number of files written what you expected? Are you missing any results?
- Do their type/extension, size, timestamp all make sense?
- Do the first few lines of output files look sensible?

Common errors



- **Library/module errors**, command not found
- **File/path issues** (no such file/dir (wd), permissions…)
- **Walltime exceeded**: job killed after reaching LSF run time limit (TIMEOUT, TIMELIMIT).
- **Our of memory**: job killed after reaching LSF memory usage limit (MEMLIMIT). You have not requested sufficient memory for your job

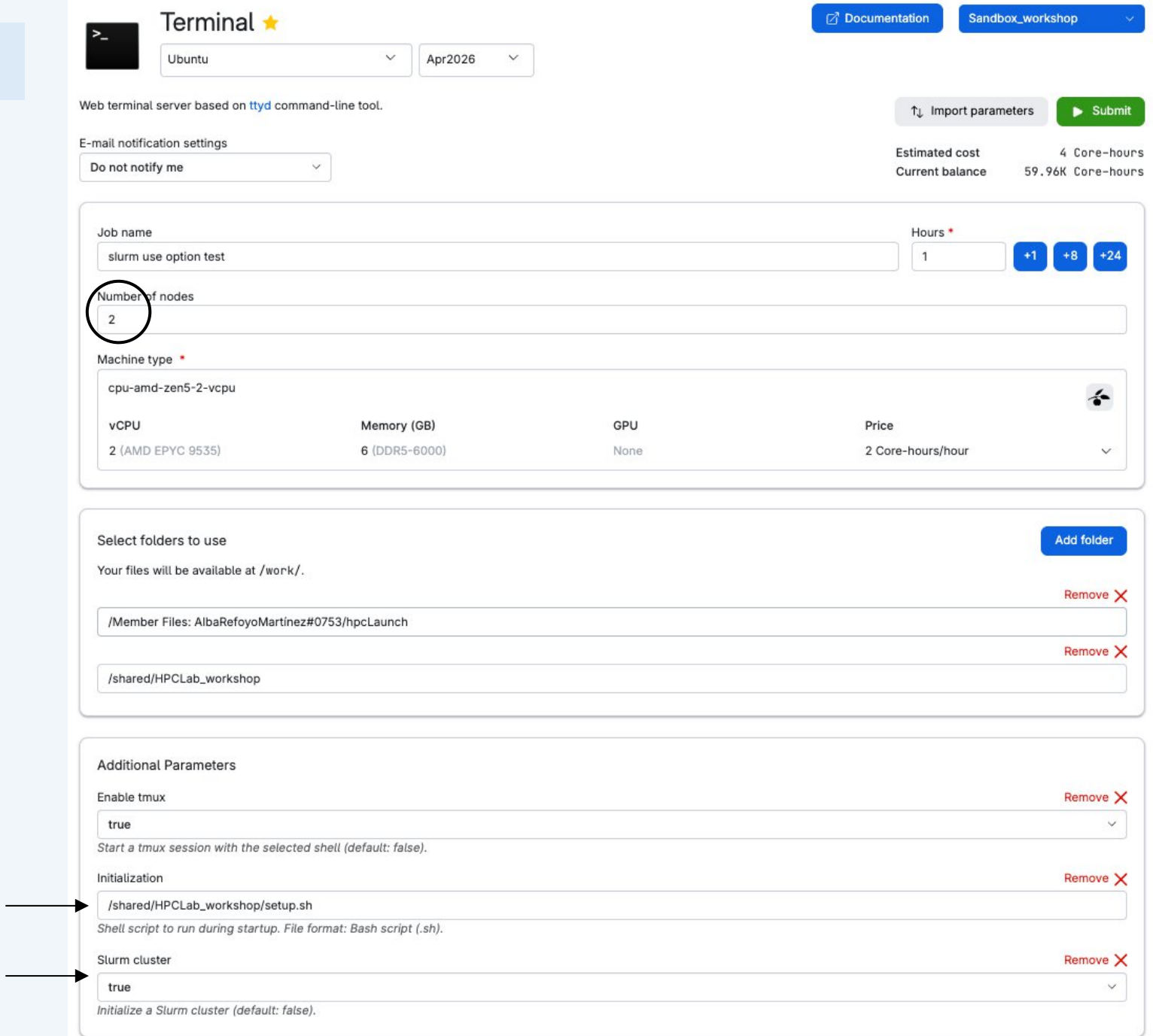
Rerun on small dataset

~ 5 min

Now is YOUR time!

Time for a quiz!

Job config settigns



Problems/Issues/Comments

- Any FAILED jobs?

srun --cpus-per-task=1 --mem=1G --time=00:01:00 --pty bash

```
hpcLaunch/day2/batchUCloud via ⓒ hpclab-env
[ 10:52:38 ] ➜  srun --cpus-per-task=1 --mem=1G --time=00:02:00 --pty bash
ucloud@j-8957816-job-1
----------------------
OS: Ubuntu 24.04.4 LTS (Noble Numbat) x86_64
Host: HPE Cray XD225v (00)
Kernel: Linux 6.12.0-124.52.3.el10_1.x86_64
Uptime: 17 days, 3 hours, 8 mins
Packages: 1100 (dpkg)
Shell: bash 5.2.21
CPU: 2 x AMD EPYC 9535 64-Core (256) @ 4.31 GHz
GPU: ASPEED Technology, Inc. ASPEED Graphics Family
Memory: 190.34 GiB / 754.90 GiB (25%)

hpcLaunch/day2/batchUCloud via ⓒ hpclab-env
[ 10:53:27 ] ➜  ls
align.sh        data.fastq   data4.fastq     logs            ref.fasta.amb          ref.fasta.pac
align_array.sh  data2.fastq  fastq_list.txt  ref.fasta       ref.fasta.ann          report_job.sh
data.bam        data3.fastq  index_array.sh  ref.fasta.0123  ref.fasta.bwt.2bit.64  results

hpcLaunch/day2/batchUCloud via ⓒ hpclab-env
[ 10:53:33 ] ➜
 ▢ 2  ↑ 13d 2h 1m   1 srun                                  | 10:53 | 20 May  ucloud  j-8957816-job-0
```

```
hpcLaunch/day2/batchUCloud via ⓒ hpclab-env
[ 10:55:40 ] ➜  srun: Job step aborted: Waiting up to 32 seconds for job step to finish.
slurmstepd-node1: error: *** STEP 52.0 ON node1 CANCELLED AT 2026-05-20T10:55:40 DUE TO TIME LIMIT ***
```

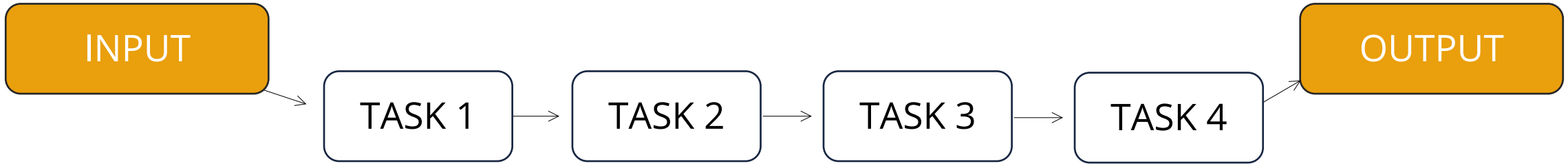
Bioinformatics workflows

For complex pipelines using dedicated workflow management systems such as Snakemake or Nextflow may be more suitable.

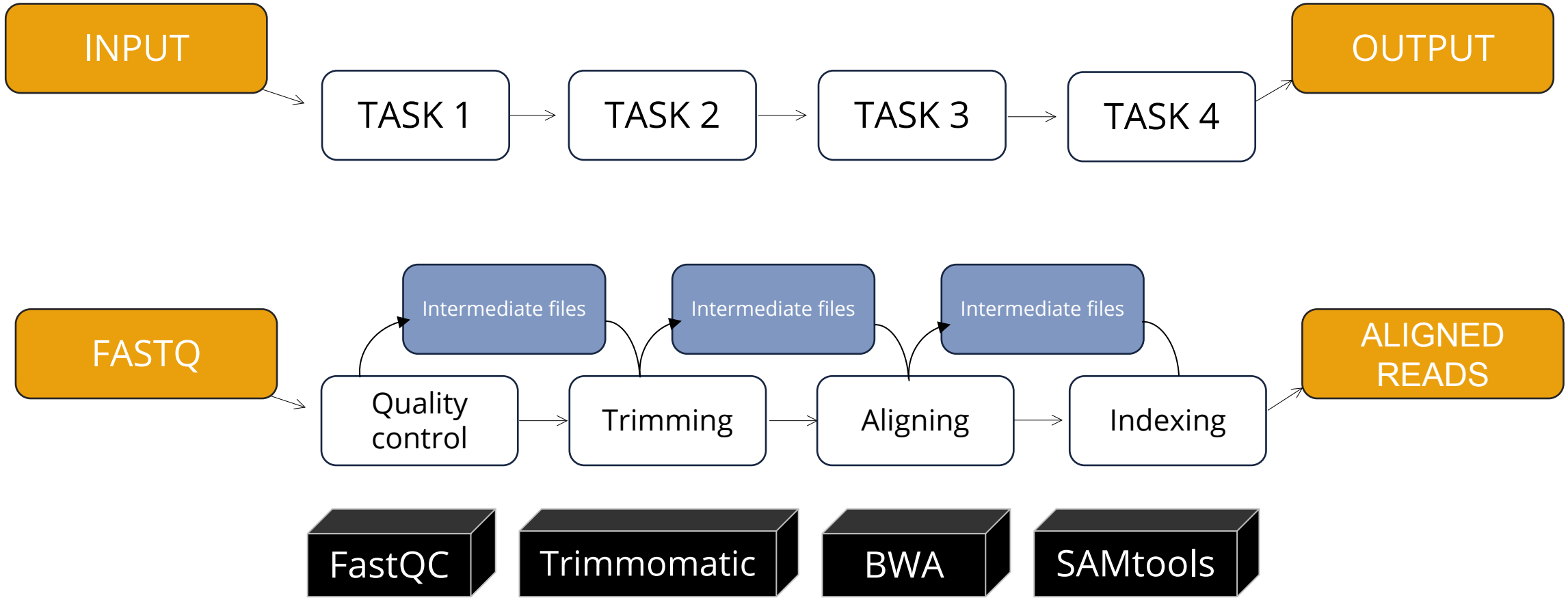
Workflow/pipeline? A series of programmatic steps to transform raw data into processed results, figures, and insights.

Workflow managers are the tools that orchestrate the processes, dependencies, deployment and tracking of large bioinformatics pipelines

Bioinformatics workflows



Bioinformatics workflows



Several software are needed to generate the desired output

Bioinformatics workflows

Let me do this by hand…

01_trim.sbatch

```
#!/bin/bash
#SBATCH --job-name=trim
#SBATCH --output=logs/trim_%A_%a.out
#SBATCH --error=logs/trim_%A_%a.err
#SBATCH --time=04:00:00
#SBATCH --cpus-per-task=8
#SBATCH --mem=16G
#SBATCH --array=1-100%10 # adjust after counting samples


module load fastqc trimmomatic

INPUT_DIR="/path/to/fastq"
INT="/path/to/intermediate"
mkdir -p "$ INT" logs

SAMPLES=$(sed -n "${SLURM_ARRAY_TASK_ID}p" \
samples.tsv)
echo "Processing sample: $SAMPLES"

# FastQC
fastqc ${INPUT_DIR}/${SAMPLES}_R1.fastq.gz \
${INPUT_DIR}/${SAMPLES}_R2.fastq.gz \
-o ${INT}/fastqc

# Trimming
trimmomatic PE -threads 8 \
${INPUT_DIR}/${SAMPLES}_R1.fastq.gz \
${INPUT_DIR}/${SAMPLES}_R2.fastq.gz \
${INT}/${SAMPLES}_R1_paired.fq.gz \
${INT}/${SAMPLES}_R1_unpaired.fq.gz \
${INT}/${SAMPLES}_R2_paired.fq.gz \
${INT}/${SAMPLES}_R2_unpaired.fq.gz \
SLIDINGWINDOW:4:20 MINLEN:50
```

02_align.sbatch

```
#!/bin/bash
#SBATCH --job-name=align
#SBATCH --output=logs/align_%A_%a.out
#SBATCH --error=logs/align_%A_%a.err
#SBATCH --time=06:00:00
#SBATCH --cpus-per-task=8
#SBATCH --mem=32G
#SBATCH --array=1-100%10

set -euo pipefail

module load bwa samtools

INPUTDIR="/path/to/intermediate"
REF="/path/to/reference/genome.fa"
OUT="/path/to/intermediate"

SAMPLE=$(sed -n "${SLURM_ARRAY_TASK_ID}p" \
samples.tsv)

echo "Aligning $SAMPLE"

bwa mem -t 8 $REF \
${INPUTDIR}/${SAMPLE}_R1_paired.fq.gz \
${INPUTDIR}/${SAMPLE}_R2_paired.fq.gz \
> ${OUT}/${SAMPLE}.sam
```

03_bam.sbatch

```
#!/bin/bash
#SBATCH --job-name=bam
#SBATCH --output=logs/bam_%A_%a.out
#SBATCH --error=logs/bam_%A_%a.err
#SBATCH --time=04:00:00
#SBATCH --cpus-per-task=4
#SBATCH --mem=16G
#SBATCH --array=1-100%10

set -euo pipefail

module load samtools

INT="/path/to/intermediate"
ODIR="/path/to/final"

mkdir -p "$FINAL"

SAMPLE=$(sed -n "${SLURM_ARRAY_TASK_ID}p" \
samples.tsv)

echo "Processing BAM for $SAMPLE"

samtools view -bS ${INT}/${SAMPLE}.sam | \
samtools sort -o ${ODIR}/${SAMPLE}.sorted.bam

samtools index ${ODIR}/${SAMPLE}.sorted.bam
```

Bioinformatics workflows

Let me do this by hand...

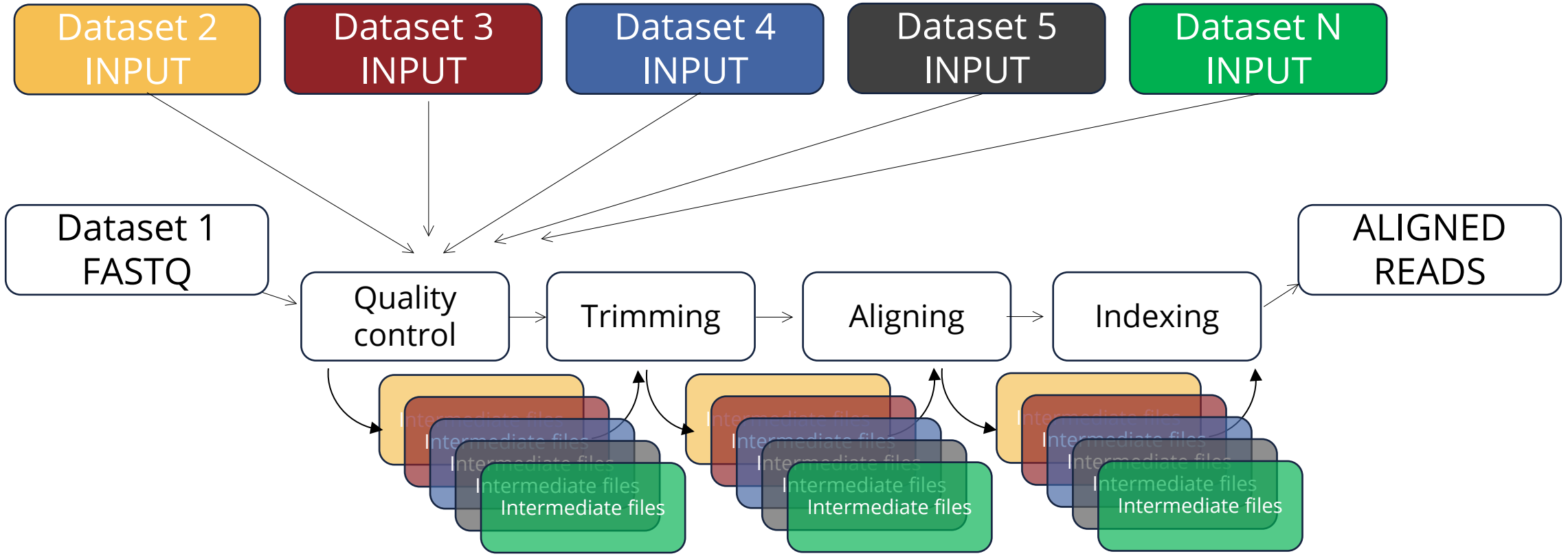
```
# Make file executable
chmod a+x <filename>.sh
```

```
./alignPipeline.sh
```

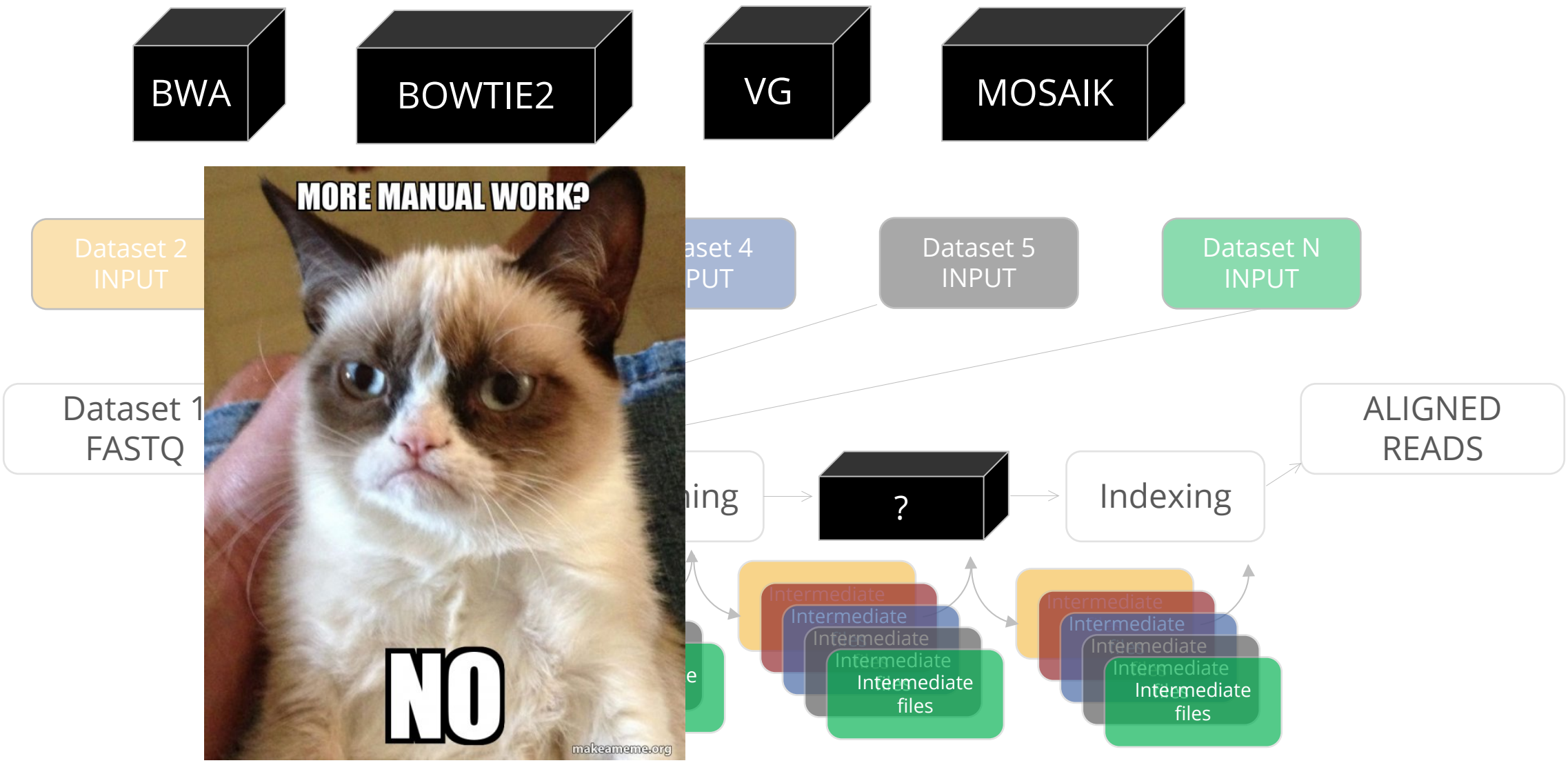
alignPipeline.sh

```
#!/bin/bash

j1=$(sbatch --parsable 01_trim.sbatch)
j2=$(sbatch --parsable \
        --dependency=afterok:$j1 02_align.sbatch)
sbatch --dependency=afterok:$j2 03_bam.sbatch
```

Now apply the same analysis to new data…



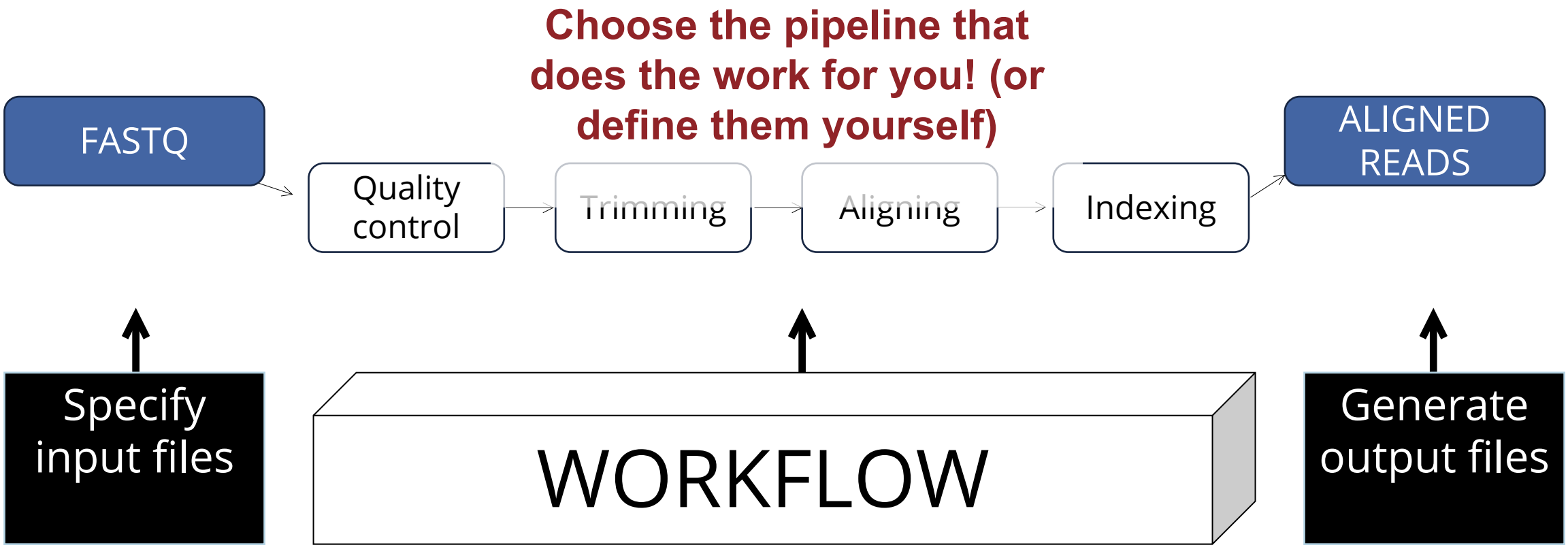
Wait, why don't we test different software mapping tools?

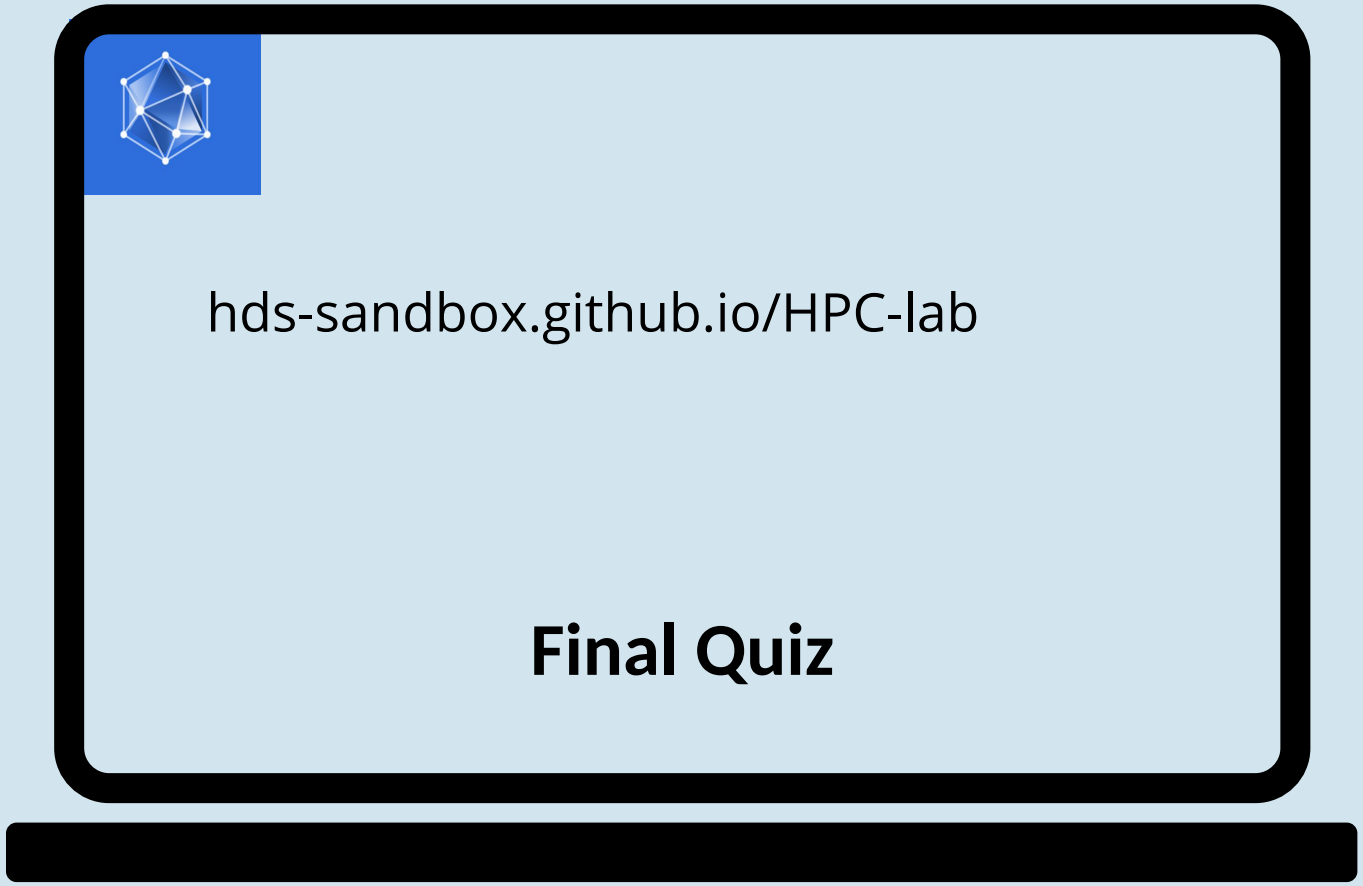


Workflow Management Systems

HPC-pipes

Execution without manual intervention





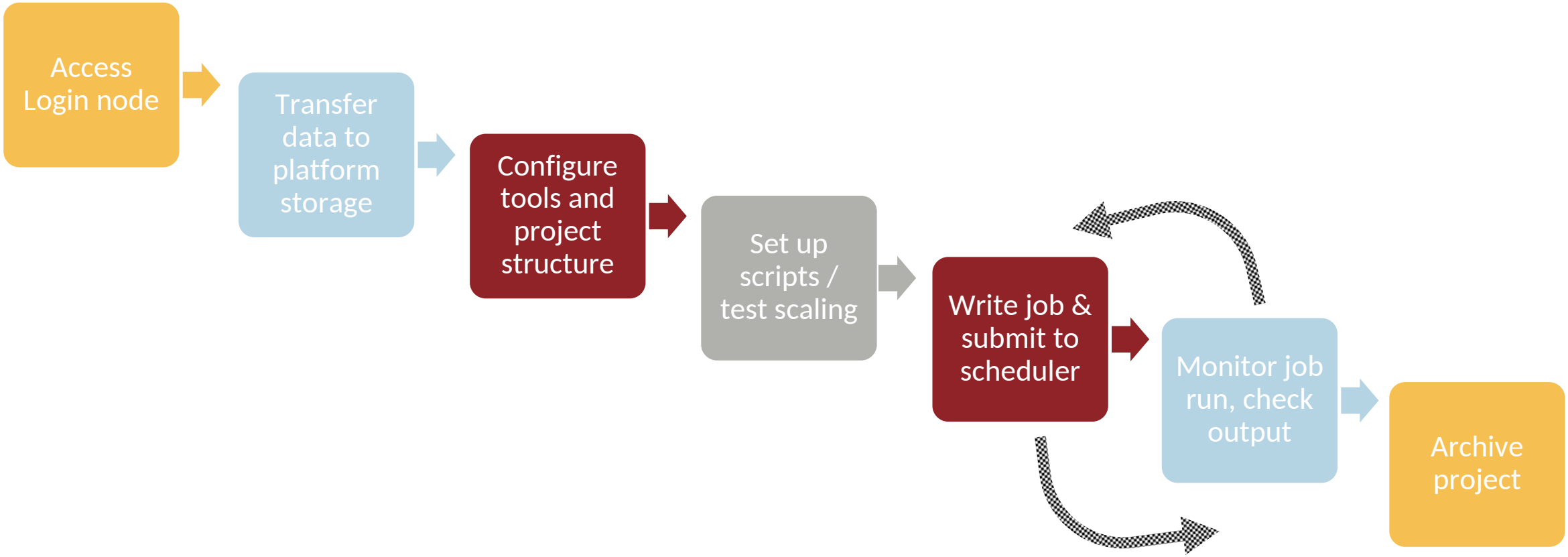
~ 5 min

Now is YOUR time!

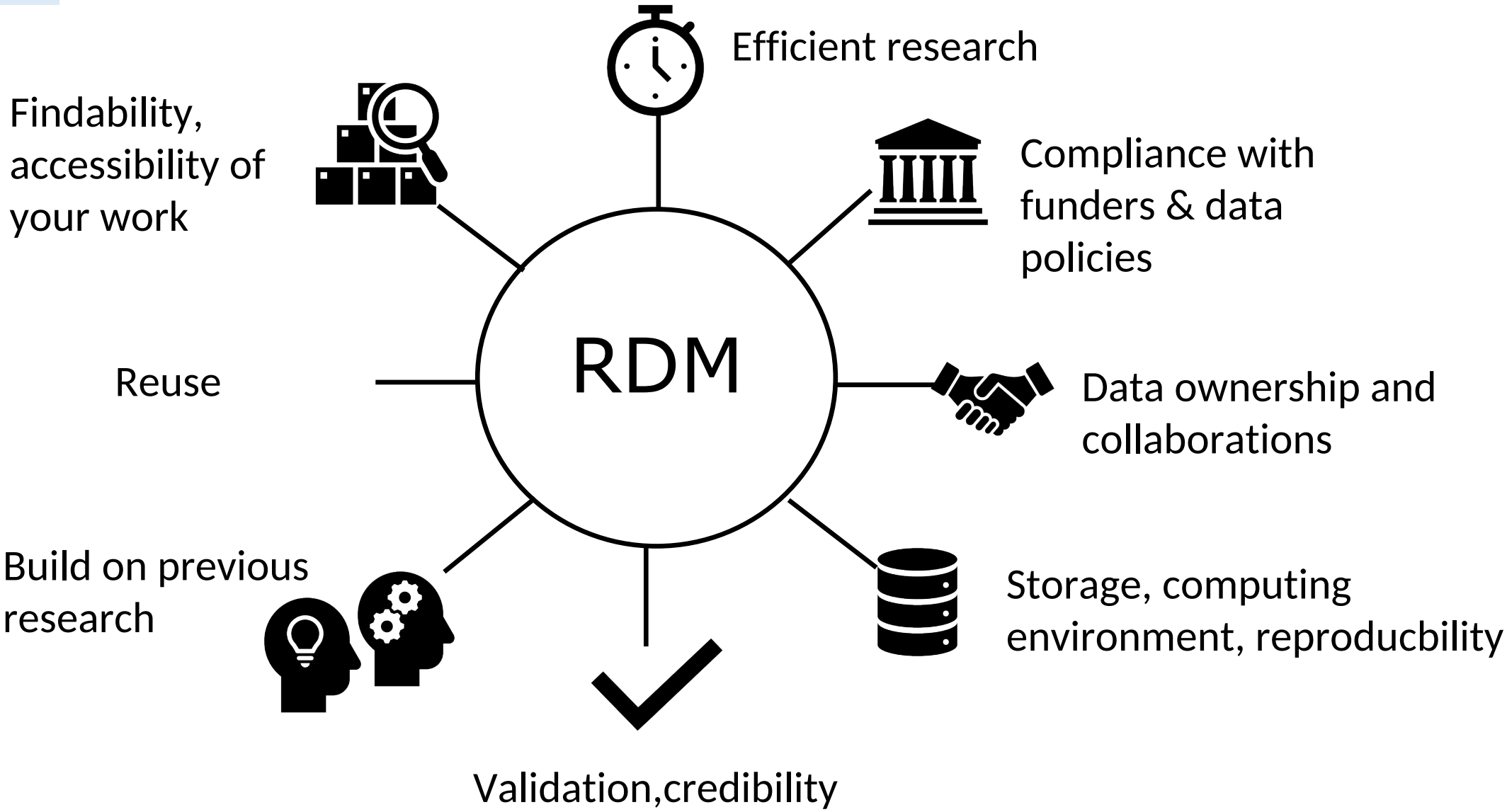
HPC, transfers, login nodes, and more!

Problems/Issues/Comments

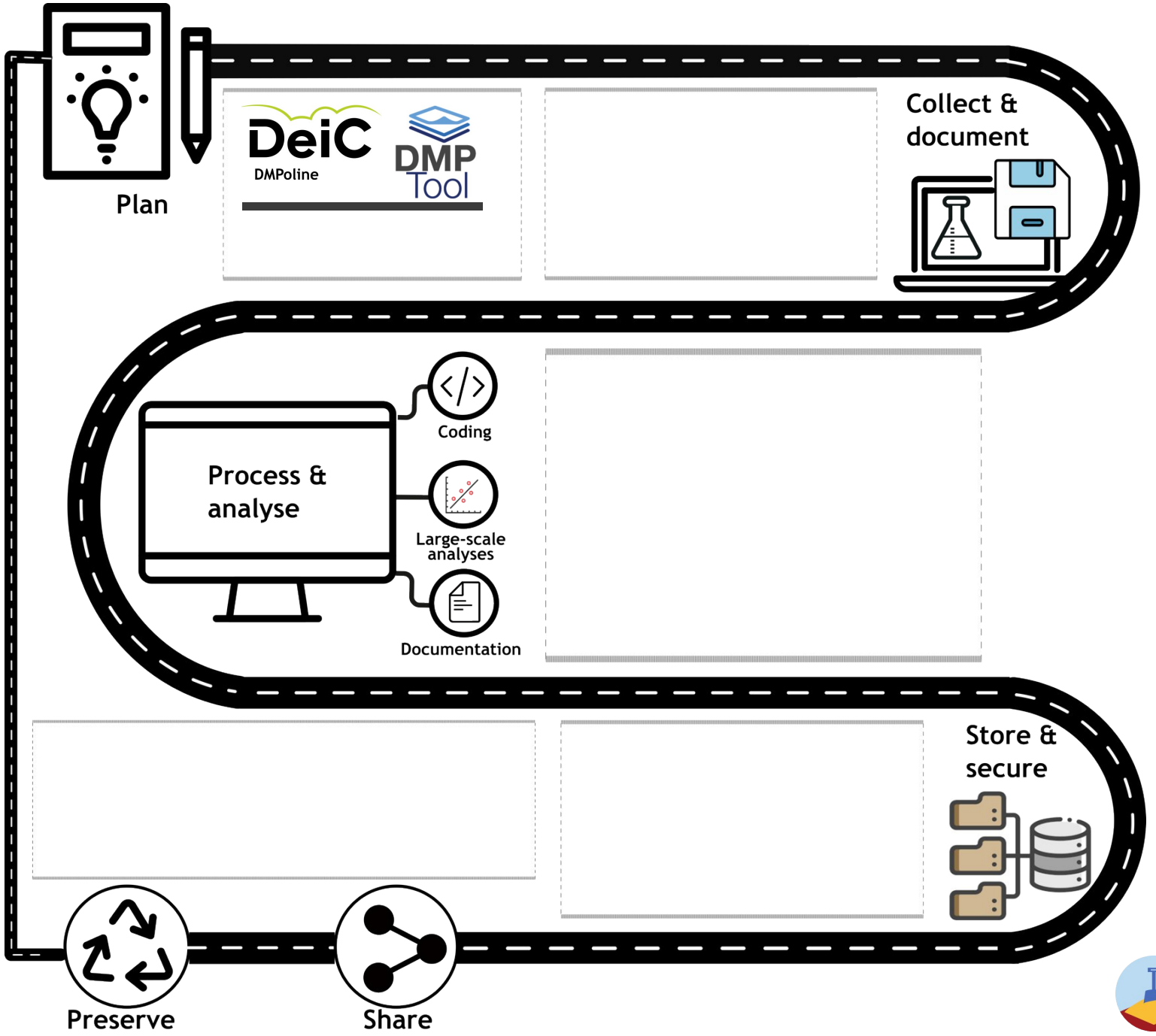
Standard HPC workflow



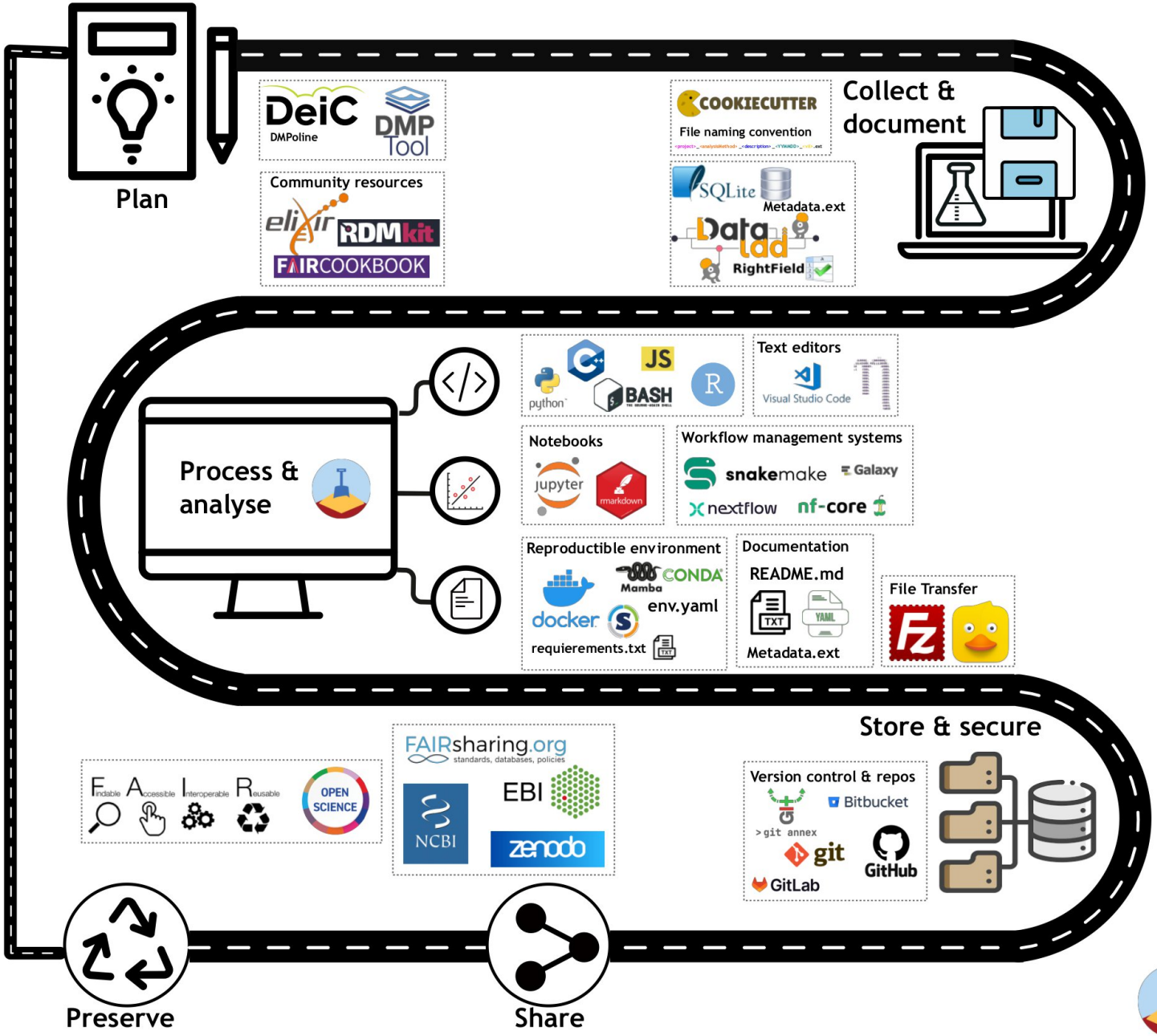
Wrap up


Efficient research
Findability, accessibility of your work
Compliance with funders & data policies
RDM
Reuse
Data ownership and collaborations
Build on previous research
Storage, computing environment, reproducbility
Validation,credibility

What tool/software/library/unix command that you have learnt in this course will help you in each of the data lifecycle phases?



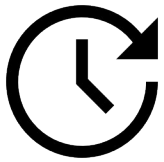
Tip sheet


Plan
DeiC
DMPoline
DMP Tool
Community resources
elixir
RDMkit
FAIRCOOKBOOK
COOKIECUTTER
File naming convention
Collect & document
SQLite
Metadata.ext
Datalad
RightField
Process & analyse
python
JS
BASH
R
Text editors
Visual Studio Code
Notebooks
jupyter
rmarkdown
Workflow management systems
snakemake
Galaxy
nextflow
nf-core
Reproductible environment
CONDA
Mamba
docker
env.yaml
requierements.txt
Documentation
README.md
TXT
YAML
Metadata.ext
File Transfer
Findable Accessible Interoperable Reusable
OPEN SCIENCE
FAIRsharing.org
standards, databases, policies
NCBI
EBI
zenodo
Store & secure
Version control & repos
Bitbucket
git annex
git
GitHub
GitLab
Preserve
Share

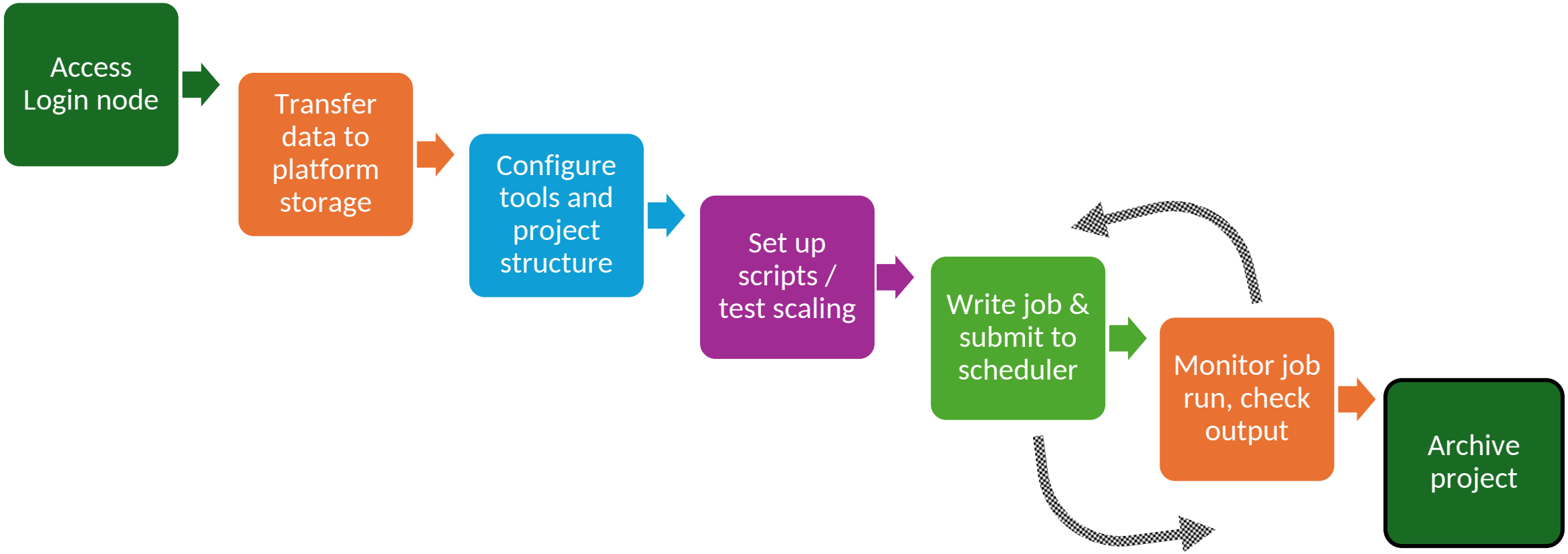
Questions

Thank you for your attention!

Your primary collaborator is yourself six months from now, and your past self doesn't answer ermails.



Data Preservation


Access Login node
Transfer data to platform storage
Configure tools and project structure
Set up scripts / test scaling
Write job & submit to scheduler
Monitor job run, check output
Archive project

rationale

rationale

- Keeping a permanent record for regulatory reasons
- Storing an accessible version of your results for replication/reuse
- Freeing up space for active projects
- COSTS

what to archive?

Archive:

Essential data and metadata

Processed data needed for reuse

Scripts, workflows and envs needed to reproduce your work

Don’t archive:

Data that is unusable

Data without any metadata

Temporary files or code that is not needed for reproducing or supporting your results

where to archive?

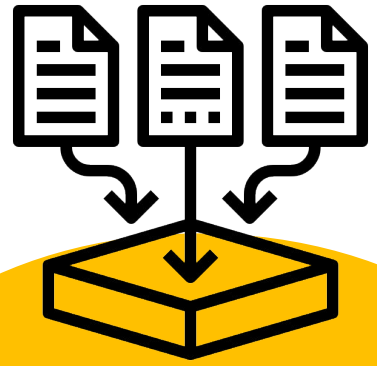
Repositories

There are public and commercial repositories which offer the ability to permanently store your data. They usually require you to make the data, or at least metadata, publicly available.

Cold storage

Many forms of long-term storage exist. They are usually using tape storage or another method where data is not actively available. Data in cold storage is usually not reused or shared.

repositories

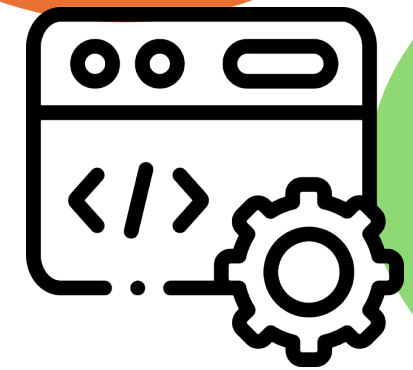


Experiment input & raw data

Domain-specific repository
Zenodo
Figshare
Dyrad

Pipelines & workflows

Zenodo
WorkflowHub
Snakemake workflow
Nextflow



Software and code

GitHub
Gitlab
Codeberg

cold storage

At present, **KU does not have any cold storage** options available in its infrastructure. This means that if your group or department does not have a specific provision for this, you would need to seek an external provider.

There are very few external providers which have a DPA with KU and can store sensitive data (Microsoft and DCAI). Even with a DPA, it is recommended that you **encrypt** your data before uploading to make it more difficult for a data leak to occur.

Encryption

Many forms of long-term storage exist. They are usually using tape storage or another method where data is not actively available. Data in cold storage is usually not reused or shared.

licensing

Licenses define **how datasets and software can be used**, including **copyright** ownership and, for human data, subject rights and usage restrictions.

The license:

- defines degree of publicity and rights to use your data
- ensures data/software is correctly attributed

Under GDPR, the data controller is responsible for ensuring lawful use and sharing of human subject data, often through signed agreements such as Data Transfer or Data Processing Agreements. It is an important aspect of the principle of reusability (R in FAIR) in data management.

Licenses are very important when using repositories, as they will determine which restrictions and use cases are available when downloading your data.

HPC checklist

Data storage is costly

- Zenodo 50GB per publication
- Figshare 20GB per publication, more costs e.g. 1TB $3500
- Dryad 50GB $150, $50 extra per 10GB extra 50$

- **Key implications**
- It can be cheaper to redo calculations instead of storage. Is it always feasible? No. Computations may be expensive and time-consuming. Evaluate your case!

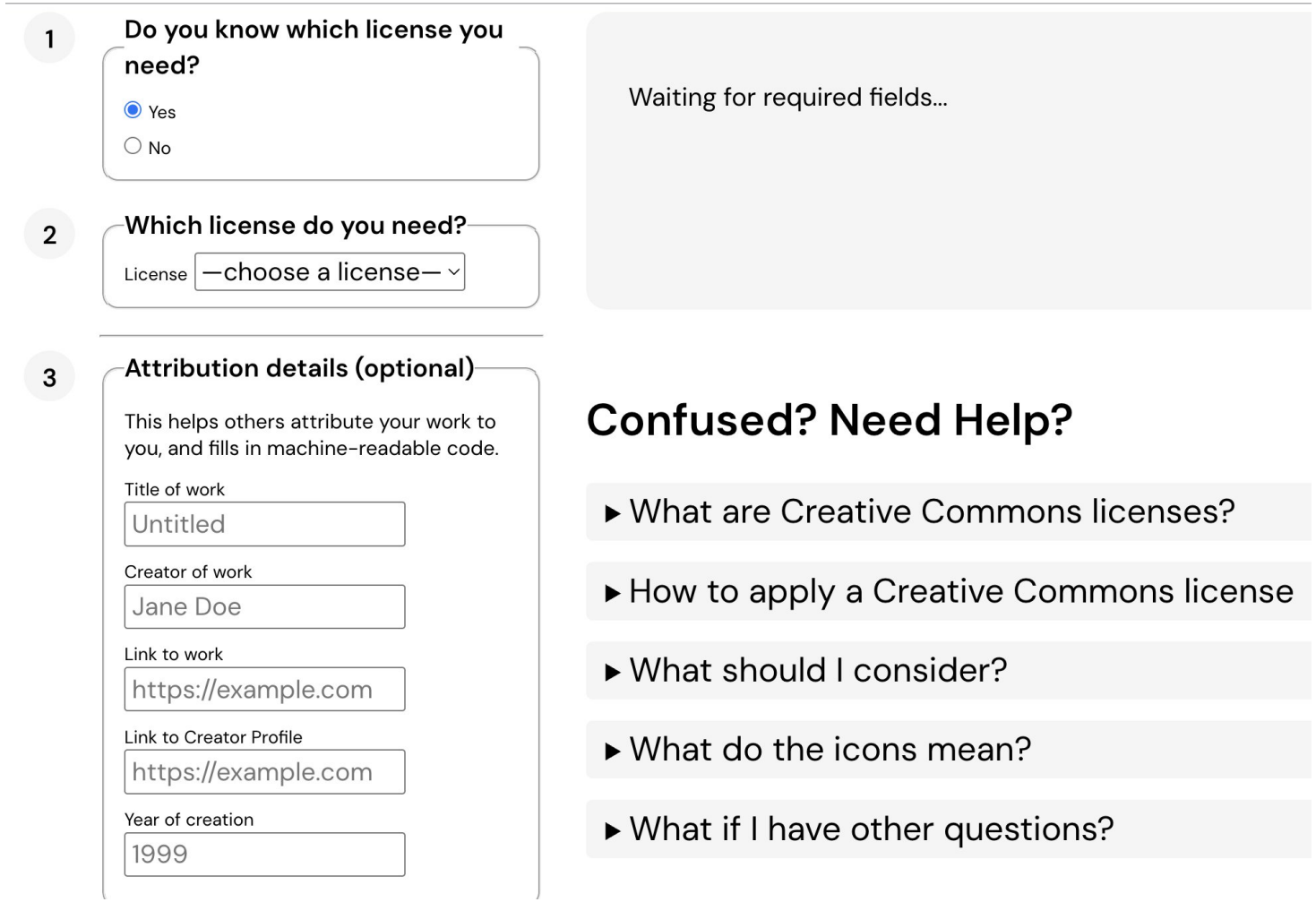
- **Best practices**
- Make sure all results are reproducible. Store?
  - Essential raw data
  - Processed data needed for reuse
  - Scripts, workflows and envs

Licenses and tools

- For data and software:
- **Creative Commons License Chooser:** https://creativecommons.org/chooser/ (CCL specific)
- **EUDAT licence selector wizard.** https://ufal.github.io/public-license-selector/
- **Choose a license.** https://choosealicense.com/ provided by GitHub, open source licenses.
- For datasets:
- data world license list: https://docs.data.world/
- For databases:
- Open Data Commons (ODC): https://opendatacommons.org/licenses/
-

Creative Commons License Chooser

Creative commons licenses widely used open data licences with different permission levels (in human-readable and machine-readable forms).



File compression

- tar can combine multiple files and dirs into a single archive file and optionally, compress the result
- The compression reduces the total data size
- Fever bytes will result in speed up data transfer
- Fewer files will also reduced metadata overhead
- Saves storage space on HPC systems (lower cost)

•tar -czf data.tar.gz data/

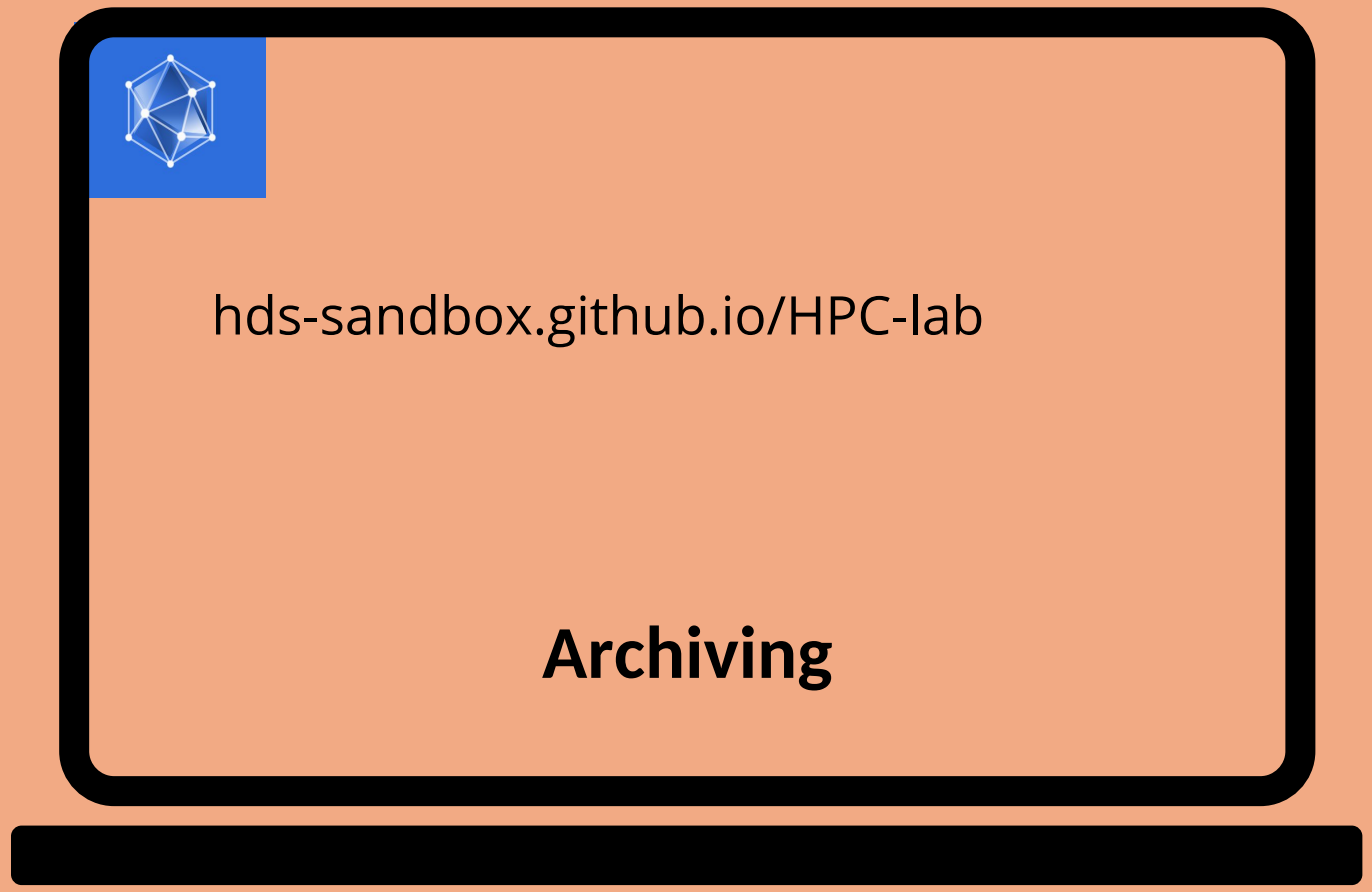
•At our university, approximately **1 PB (petabyte) of data is generated every month**

Tar - flags

- -x: to extract the archive
- -v: for verbose output
- -z: for gzip compression
- -f: «tarball» filename

#compressed archive format commonly used on Linux.

- Flags can be concatenated and interchangeable but –f needs to be followed by a filename.
- *.tar.gz: uses a two-step process: files are first archived with tar (files or folders merged), then compressed with gzip.



~ 5 min

Now is YOUR time!

- Time for a quiz!