

# Introduction & Setup

from the  
Health Data Science  
Sandbox

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

## 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

SERVICES

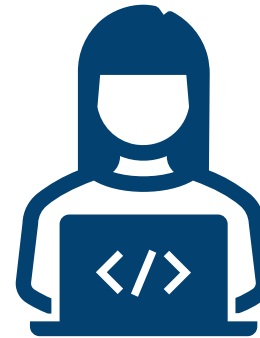
# WHO WE SUPPORT



**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

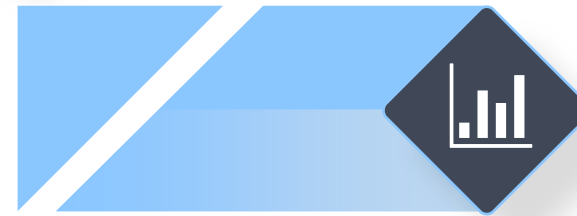


### CONSULTATIONS

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)

From  
Excel to R



Python  
Tsunami



Bash &  
Terminal



Git &  
Github



R for  
Data Science



Python for  
Data Science



HPC-Launch



GDPR for  
Biomed



Genomics  
Sandbox



RNA-seq  
Sandbox



Proteomics  
Sandbox



Health Records  
Sandbox



HPC-Pipes



ML in Biomed



HPC-ML



Practical AI for  
Biomed



**1st class on  
October 19th**

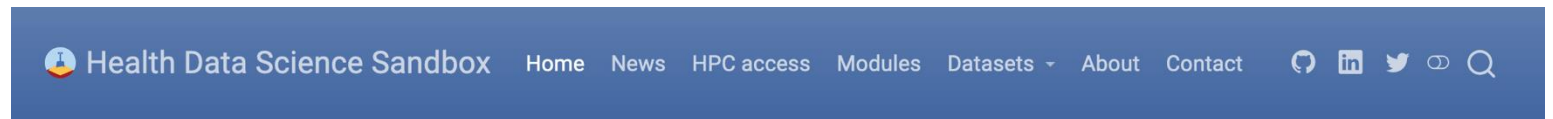
# HEALTH DATA SCIENCE SANDBOX



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>



## Welcome to the Health Data Science Sandbox

Access our training modules

<b>HPC Lab</b>	<b>Genomics</b>	<b>Transcriptomics</b>	<b>Proteomics</b>	<b>Health records</b>
Research Data Management HPC launch HPC pipes	NGS data analysis Population Genomics GWAS	Bulk RNAseq Single-cell RNAseq	Clinical Proteomics ColabFold	Synthetic data Personalized Medicine

## We are a collaborative project with team members spanning five Danish universities

The Health Data Science Sandbox is a national project coordinated by the [Center for Health Data Science](#) at the University of Copenhagen. We're working with a network of health data science experts to build training resources on academic supercomputers for students and researchers in Denmark. Our Sandbox contains [training modules](#) that pair topical datasets with recommended analysis tools, pipelines, and learning materials/tutorials in a portable, containerized format.



## course objectives

- Explain the purpose and structure of a bioinformatics pipeline
- Develop and manage reproducible data analysis pipelines using Snakemake (and briefly nextflow)
- Control the software environment of a workflow/pipeline using workspace management tools like conda and docker
- Manage data and computing using best practices (RDM) and appropriate compute provisioning (HPC)

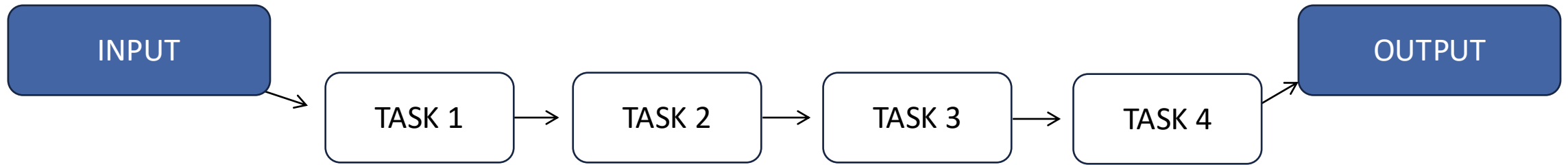


<https://hds-sandbox.github.io/HPC-lab/workshop/pipes-requirements.html>

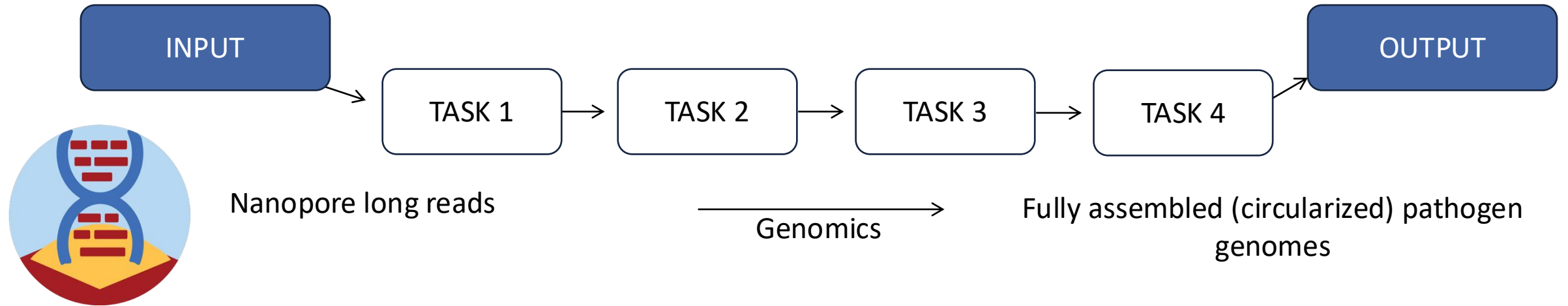
# Introduction to pipelines



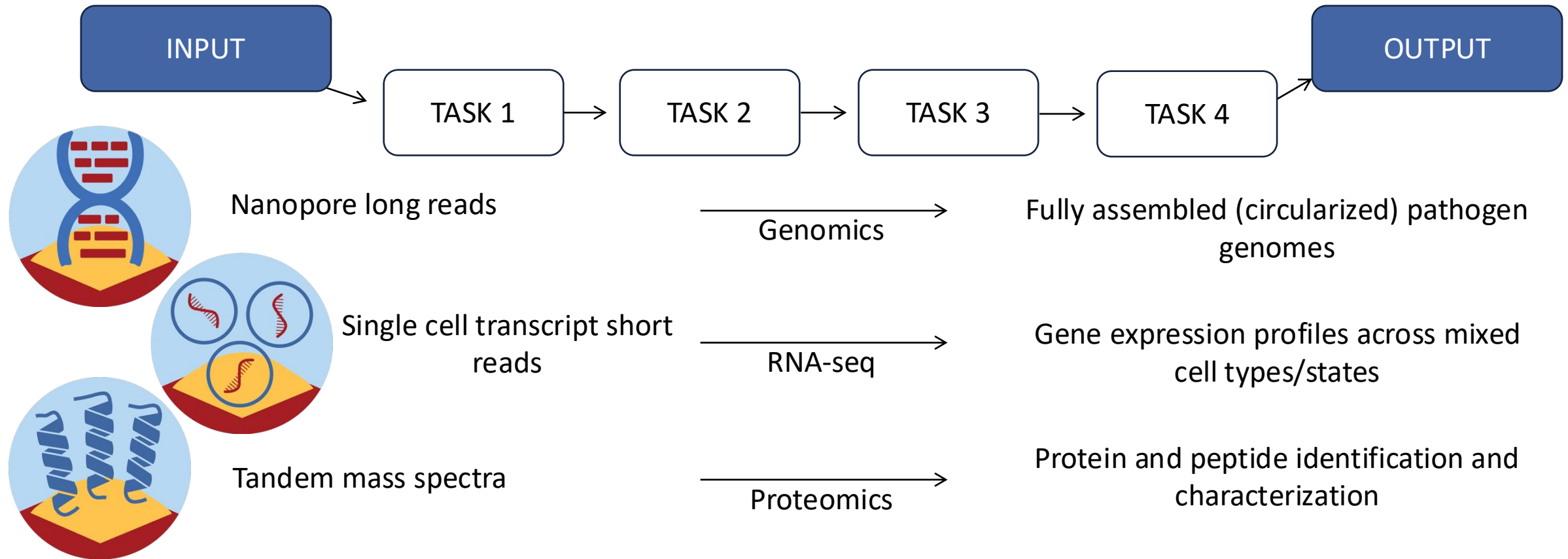
## purpose of a bioinformatics pipeline



# purpose of a bioinformatics pipeline



# purpose of a bioinformatics pipeline



# components of a pipeline

## Data

The raw experimental files going in (e.g. FASTQ reads) and the processed results coming out (e.g. aligned BAMs, count tables, figures)



# components of a pipeline

## Data

The raw experimental files going in (e.g. FASTQ reads) and the processed results coming out (e.g. aligned BAMs, count tables, figures)

## Procedures

The ordered series of tasks/tools that do the actual work (QC, filtering, mapping, annotation, output), with each step's output feeding the next



# components of a pipeline



## Data

The raw experimental files going in (e.g. FASTQ reads) and the processed results coming out (e.g. aligned BAMs, count tables, figures)



## Procedures

The ordered series of tasks/tools that do the actual work (QC, filtering, mapping, annotation, output), with each step's output feeding the next



## Environment

Specific software, tool versions, and dependencies each step needs. Often more than one, since different steps may need conflicting versions





# components of a pipeline

## Data

The raw experimental files going in (e.g. FASTQ reads) and the processed results coming out (e.g. aligned BAMs, count tables, figures)

## Procedures

The ordered series of tasks/tools that do the actual work (QC, filtering, mapping, annotation, output), with each step's output feeding the next

## Environment

Specific software, tool versions, and dependencies each step needs. Often more than one, since different steps may need conflicting versions

## Platform

*Where it all runs:*  
the platform providing the CPUs, memory, and storage, plus the scheduler that allocates them (your laptop, an HPC cluster or the cloud)



# components of a pipeline



## Data

The raw experimental files going in (e.g. FASTQ reads) and the processed results coming out (e.g. aligned BAMs, count tables, figures)



## Procedures

The ordered series of tasks/tools that do the actual work (QC, filtering, mapping, annotation, output), with each step's output feeding the next



## Environment

Specific software, tool versions, and dependencies each step needs. Often more than one, since different steps may need conflicting versions



## Platform

*Where* it all runs: the platform providing the CPUs, memory, and storage, plus the scheduler that allocates them (your laptop, an HPC cluster or the cloud)



## Version control

The record of *how* the pipeline is defined and how it changed over time, so the workflow itself is reproducible and shareable (eg Git)



# components of a pipeline



Data



Procedures



Environment



Platform



Version control



# why use an HPC?

## **Reproducibility**

Same tool versions and dependencies, same master files, same compute resources



## why use an HPC?

### **Reproducibility**

Same tool versions and dependencies, same master files, same compute resources

### **Scalability**

Easily upgrade a job's resources beyond laptop power without needing to overspend



## why use an HPC?

### **Reproducibility**

Same tool versions and dependencies, same master files, same compute resources

### **Scalability**

Easily upgrade a job's resources beyond laptop power without needing to overspend

### **Collaboration**

Shared data, environments and code lets more people access and reproduce your code



## why use an HPC?

### **Reproducibility**

Same tool versions and dependencies, same master files, same compute resources

### **Scalability**

Easily upgrade a job's resources beyond laptop power without needing to overspend

### **Collaboration**

Shared data, environments and code lets more people access and reproduce your code

### **Safety**

HPC providers have higher security standards and adequate backup procedures



## why use an HPC?

### **Reproducibility**

Same tool versions and dependencies, same master files, same compute resources

### **Scalability**

Easily upgrade a job's resources beyond laptop power without needing to overspend

### **Collaboration**

Shared data, environments and code lets more people access and reproduce your code

### **Safety**

HPC providers have higher security standards and adequate backup procedures

Be careful when working with sensitive data on someone else's computer!





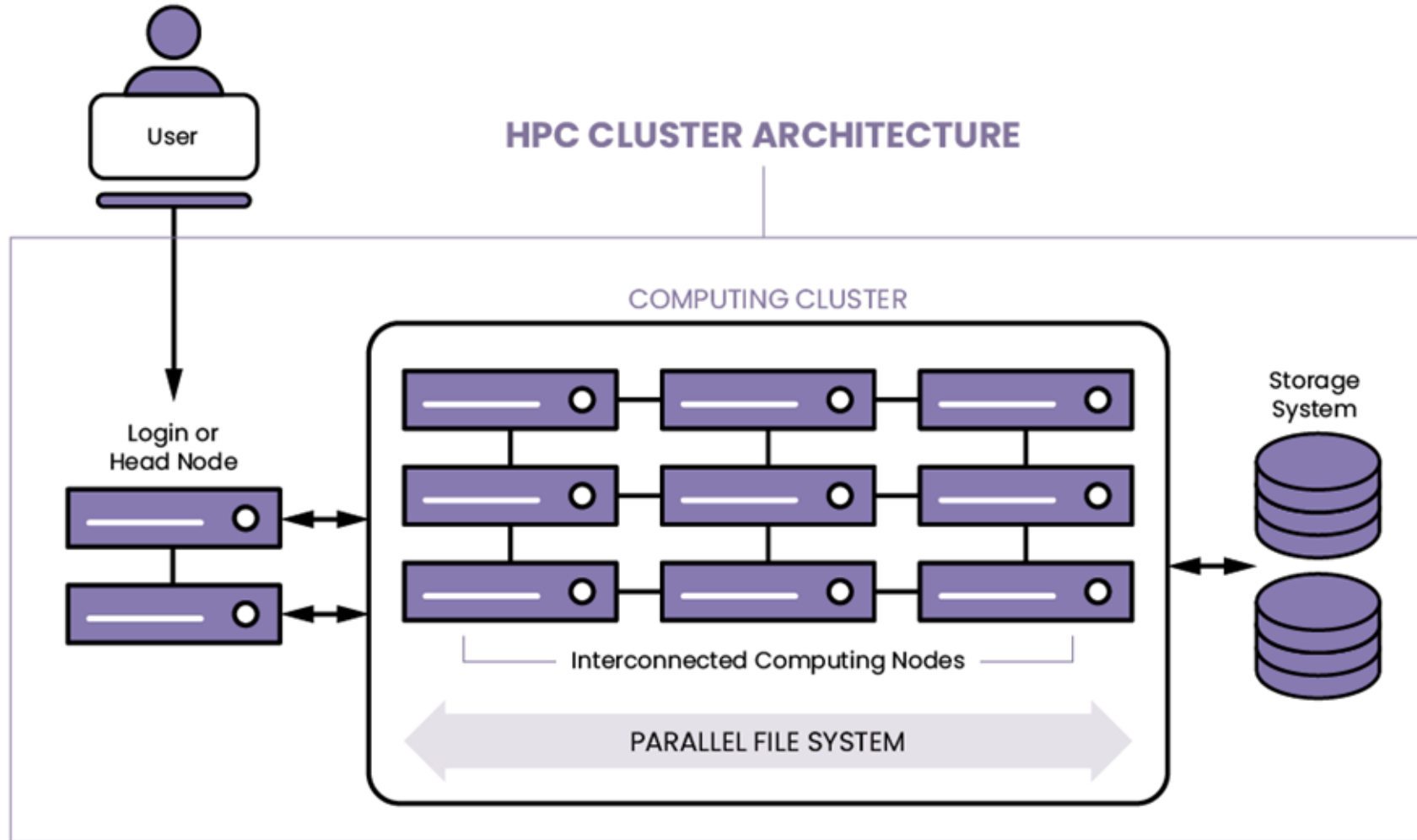
 Have you used an HPC before? If so, which one(s)?

 Have you used an HPC before? If so, which one(s)?

 Do you prefer working on the terminal or with a different setup?

- Have you used an HPC before? If so, which one(s)?
- Do you prefer working on the terminal or with a different setup?
- Do you work with sensitive data?

# HPC architecture

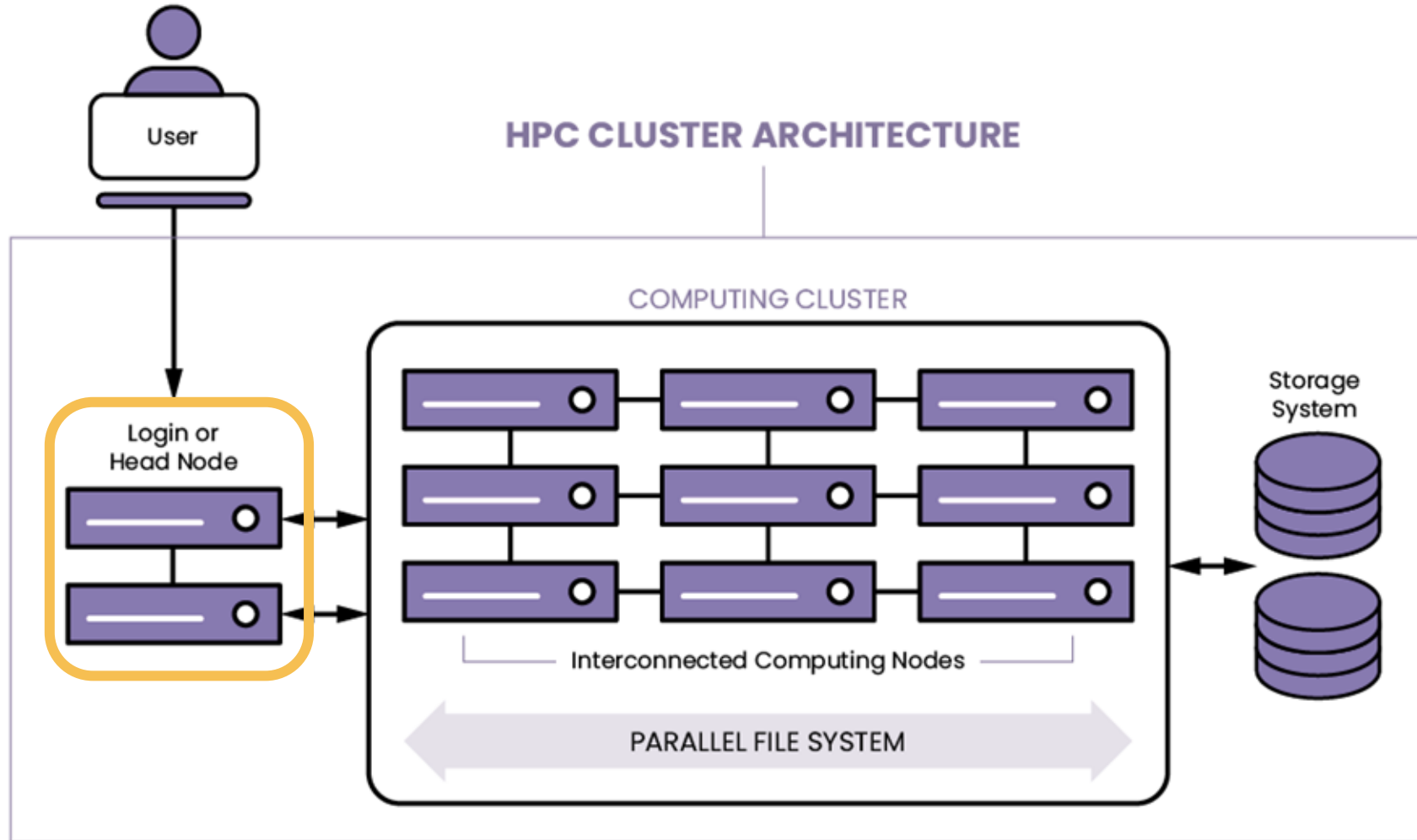


## Node

- 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

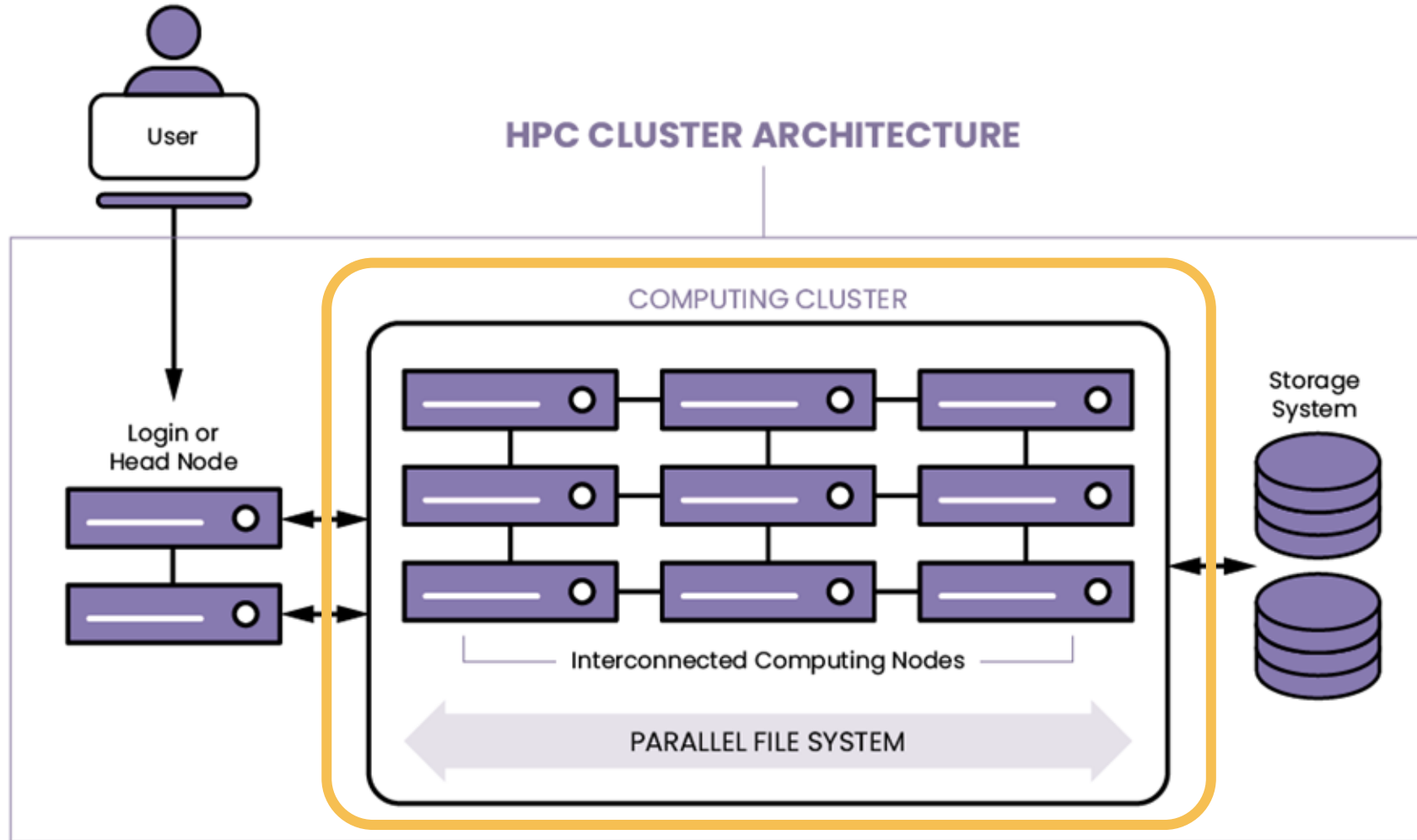


## Login Node

- 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

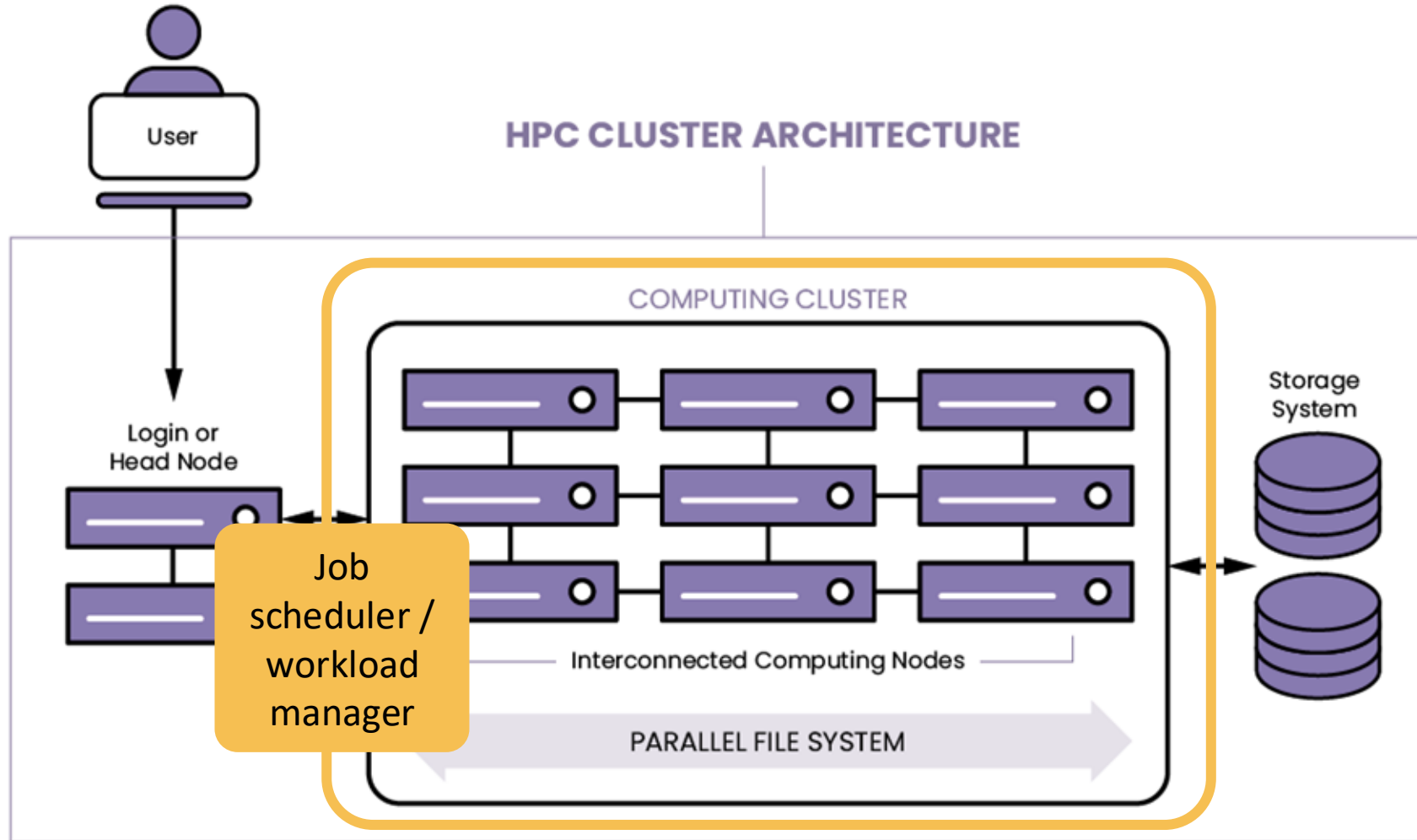


## Computing Cluster

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



# HPC architecture

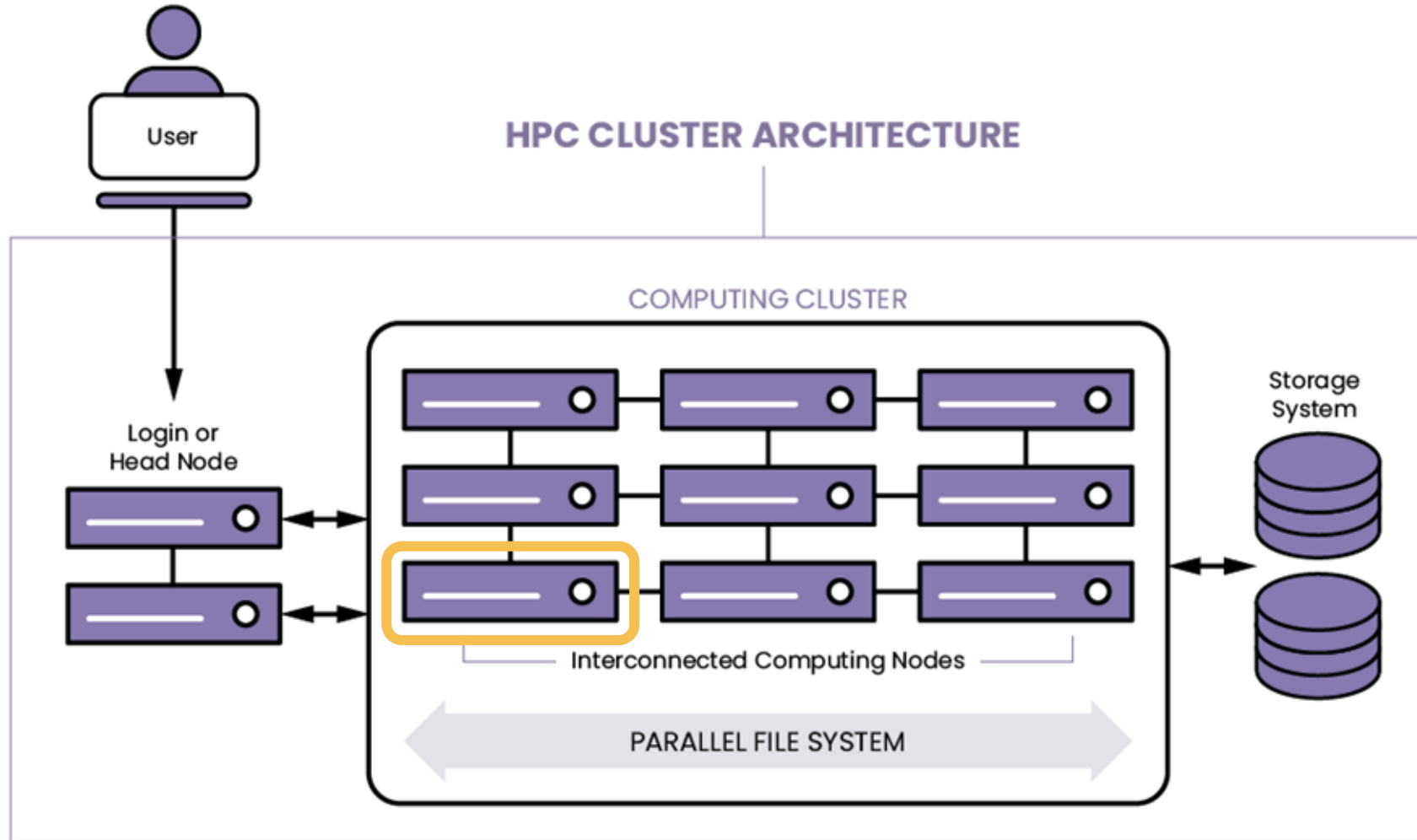


## Job scheduler

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



# HPC architecture



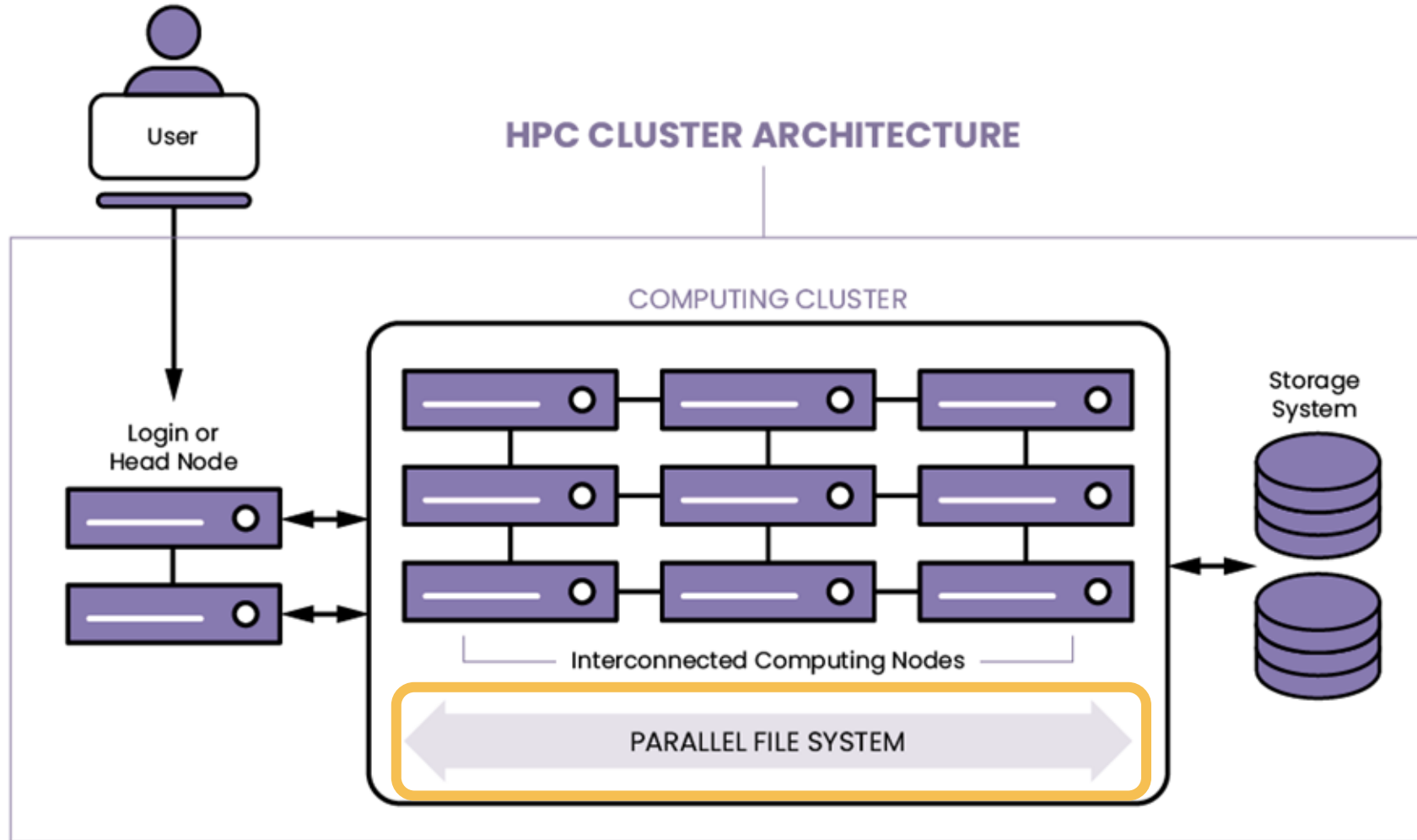
## Computing Node

- 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

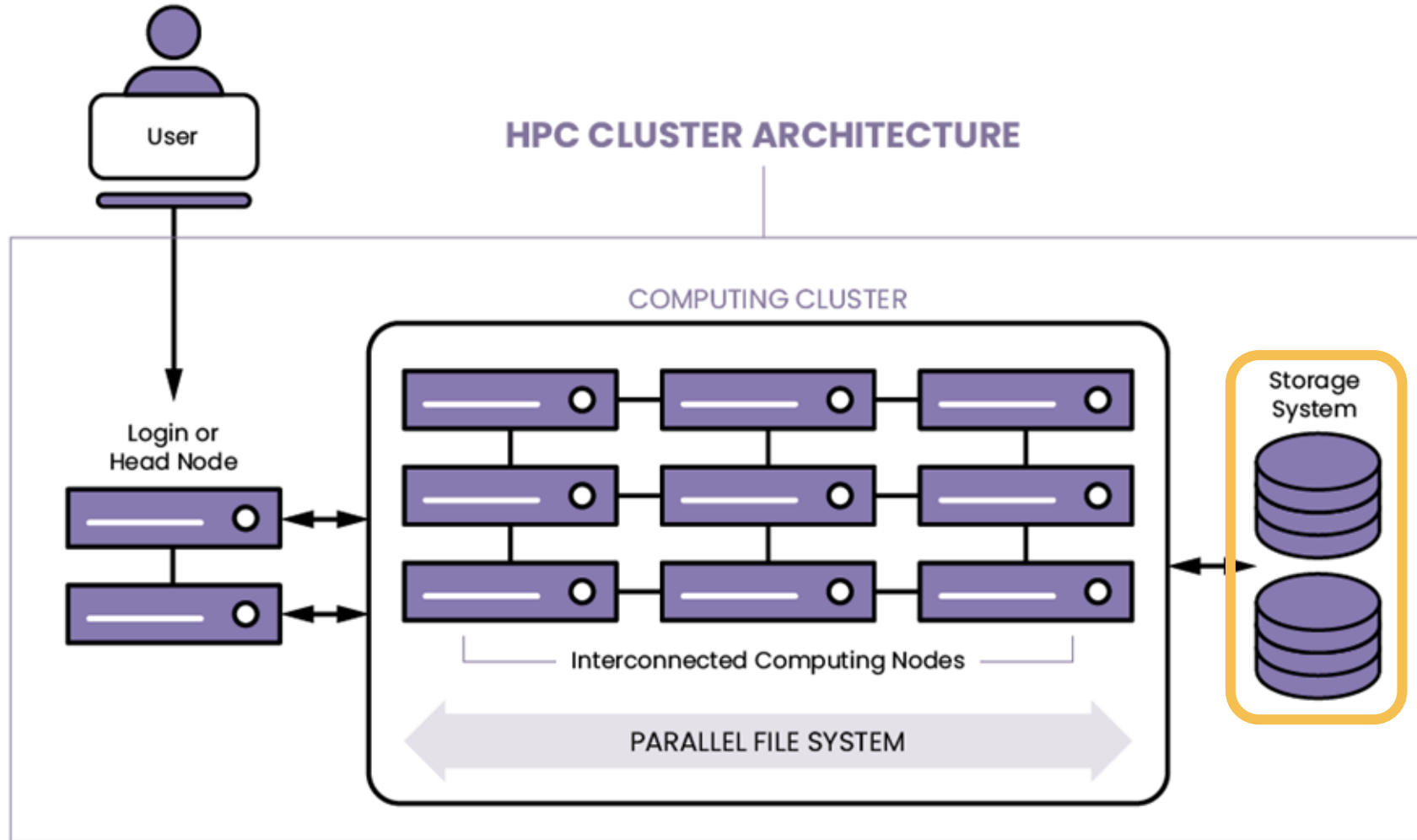


## Parallel File System

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



# HPC architecture



## 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



## standard HPC workflow



access login node

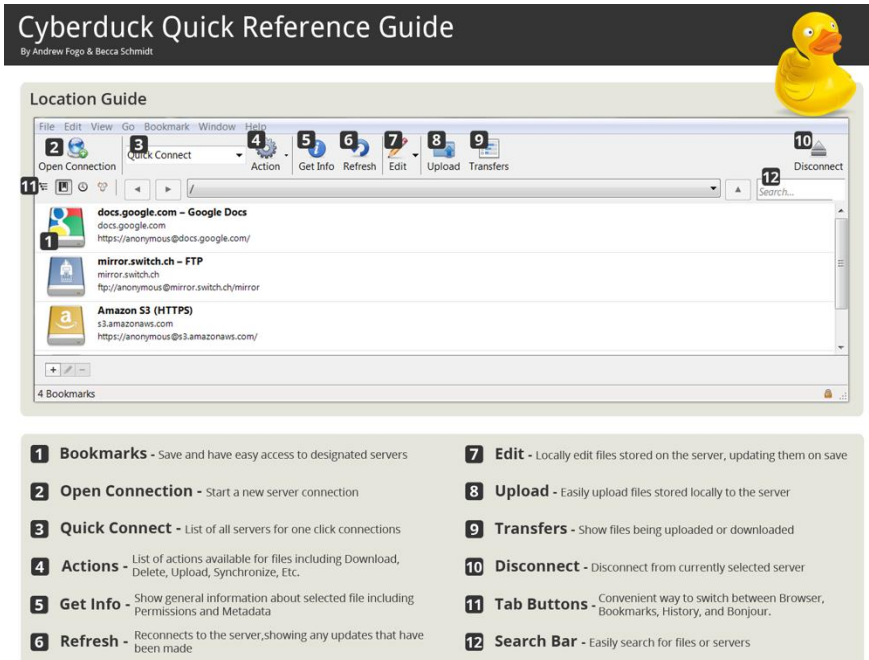
**GENOME | DK**  
HIGH-PERFORMANCE COMPUTING

[illegible]

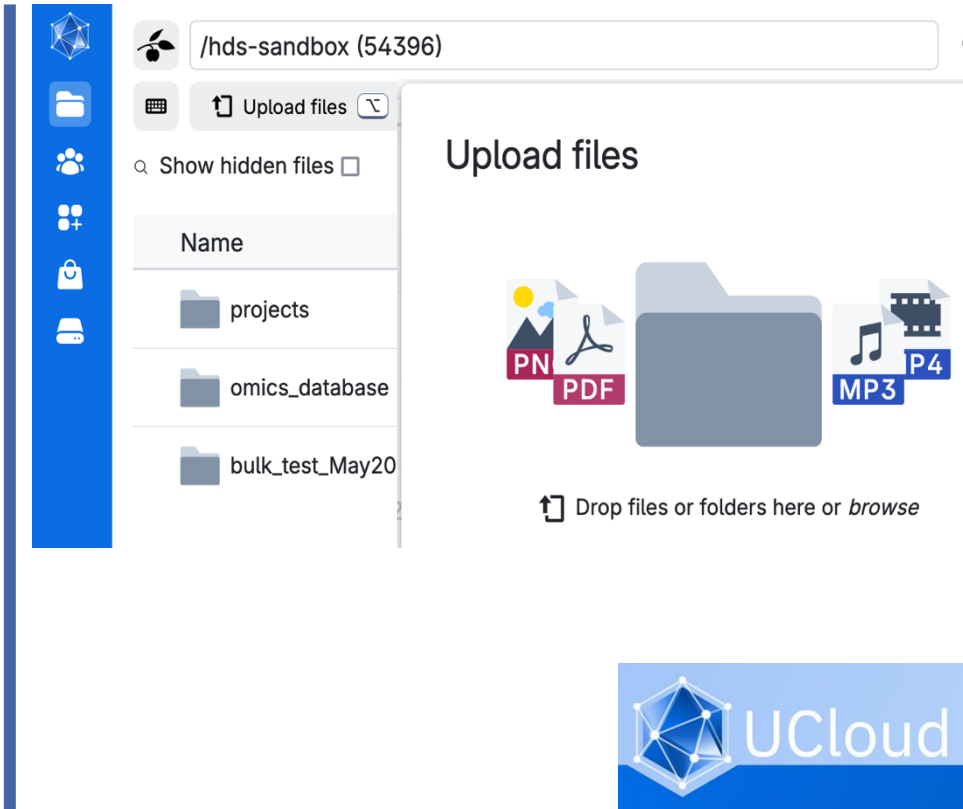
The screenshot shows the UCloud dashboard interface. On the left is a blue sidebar with the 'Applications' section expanded, listing various tools like ColabFold, Proteomics Sandbox, Genomics Sandbox, JupyterLab, and 'Coder' (which is highlighted). Below this is a 'Store' section with items like Hippo, Engineering, Data Analytics, Artificial Intelligence, and Quantum Computing. The main content area on the right features a news article titled 'City Expanded with new NVIDIA H100s' dated 13/08/2024, which describes the addition of four NVIDIA H100 GPUs to the system. At the bottom right, a summary shows '10,03K / 174,58K Core-hours'.

# transfer data

On the terminal you can use scp or rsync instead



- [Filezilla](#) [linux/mac/win]
- [Cyberduck](#) [mac]
- [MobaXterm](#) [win]
- [WinSCP](#) [win]



## tool configuration and testing

Pipelines are not born ready: you need to prototype first.

Since many of the issues you will find are platform-dependent, it makes sense to prototype in the platform your pipeline will be run in later (when feasible).

## tool configuration and testing

Pipelines are not born ready: you need to prototype first.

Since many of the issues you will find are platform-dependent, it makes sense to prototype in the platform your pipeline will be run in later (when feasible).

HPC platforms often manage dependencies via modules. Otherwise, manager your own installs with tools like conda or Docker.

Look for community standards: protocol and benchmarking papers, curated pipelines or senior advice.

## tool configuration and testing

Pipelines are not born ready: you need to prototype first.

Since many of the issues you will find are platform-dependent, it makes sense to prototype in the platform your pipeline will be run in later (when feasible).

HPC platforms often manage dependencies via modules. Otherwise, manager your own installs with tools like conda or Docker.

Look for community standards: protocol and benchmarking papers, curated pipelines or senior advice.

After installs, test with a benchmark dataset; try your best and worst exemplars to surface bugs. In some sense, your task is to find what will break (ie edge cases, data formatting issues, e tc.)



## tool configuration and testing

Pipelines are not born ready: you need to prototype first.

Since many of the issues you will find are platform-dependent, it makes sense to prototype in the platform your pipeline will be run in later (when feasible).

HPC platforms often manage dependencies via modules. Otherwise, manager your own installs with tools like conda or Docker.

Look for community standards: protocol and benchmarking papers, curated pipelines or senior advice.

After installs, test with a benchmark dataset; try your best and worst examplers to surface bugs. In some sense, your task is to find what will break (ie edge cases, data formatting issues,e tc.)

### **Rule of thumb**

If your analysis runs 5x without changes, start formalizing it into a pipeline

# job schedulers

## **SLURM**

Simple Linux Utility for  
Resource Management

Used at GenomeDK and  
many other servers.  
Supports parallel job  
arrays and cores-per-  
task

Job schedulers have  
resource monitoring  
tools that you can use in  
writing intermediate  
pipeline data.

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

You must specify cores,  
RAM & runtime up front,  
so expect some trial and  
error.

Start small on resources  
when protoyping, you can  
scale easily later. That  
said, your estimates are  
often wrong, so don't try  
to be too conservative.

## common SLURM comands

**sbatch** submit a batch job

**srun** interactive / run a step



**squeue** view running vs pending jobs



**jobinfo** status of a specific job




**scancel** kill a specific job

**Sinfo** stats on nodes and partitions

# UCloud is... different

**Coder** 

Base  1.93.1 

[Documentation](#)   Sandbox\_workshop 

E-mail notification settings  
Do not notify me 

Estimated cost 6 Core-hours  
Current balance 23,16K Core-hours

Job name  
test\_JAB

Hours   
3  +1 +8 +24

Machine type   
u1-standard-2

vCPU	Memory (GB)	GPU	Price
2 (Intel Xeon Gold 6130)	12	None	2 Core-hours/hour 

Select folders to use  Add folder

Your files will be available at /work/.

/Member Files: kcs305kcs305#7929/work\_JAB 

/shared/HPCLab\_workshop 

Additional Parameters

Initialization   
/shared/HPCLab\_workshop/setup.sh

Run a Bash script (\*sh) for initialization.

Optional Parameters

Search 

Modules path 

Configure SSH access

This application has optional support for SSH. In order to use SSH access, you must configure at least one SSH key. You can configure your SSH keys [here](#).

☒ Enable SSH server

## interactive vs batch

**Interactive jobs** let you request compute resources / access a node from an active terminal.

- Realtime output as you run commands (like working on your local machine)
- You must remain logged in / active while you are running an interactive job, or your computations will be cancelled
- Great for debugging, but not great for running lengthy jobs, or running multiple jobs in parallel

**Batch jobs** let you request compute resources or access a node using a job script.

- No realtime output or ability to respond to errors / fix bugs
- Includes commands that load necessary tools and data directories for the entire set of computations
- Great for running lengthy/parallel jobs independently (without being logged in / active monitoring), bad for debugging/troubleshooting



## tips for setting up jobs

Use commands for getting info on the different nodes available (ie sinfo for SLURM)

Always check platform documentation to see how you will be billed for using different resources

Some platforms may charge you for using an entire node regardless of core number requested (sensitive data compartmentalization)  
Fancier nodes (GPU / fat nodes) cost more

Don't request the max RAM for a node, because slightly less is available in realtime operation

Optimize your job accordingly and use usage monitoring tools

Remember to design your pipelines to benefit from parallelization when possible, this will save you time and money in the long run



# UCloud & GenomeDK Setup



## HPC alternatives

	UCloud	GenomeDK	Computerome	DCAI/Gefion	LUMI
<b>CPU nodes</b>	6592 vCPUs	52 thin/60 fat (~11k cores) (expanding)	692 thin/55 fat (50k cores)	191 nodes/112 cores each	2048 nodes/128 cores
<b>GPU nodes</b>	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
<b>Storage</b>	3 PB	33 PB	50 PB	~8 PB	~120 PB
<b>Security</b>	ISO 27001	ISAE 3000 + ISO 27001	ISAE 3000 + ISO 27001	NA	ISAE 3000 + ISO 27001
<b>Sensitive data</b>	yes, 'at own risk'	yes, 'closed zones'	yes, private clouds	not yet	not yet
<b>Env mgmt</b>	conda	Singularity, Apptainer	conda (& Docker?)	NA	Singularity, Apptainer
<b>OS</b>	UCloud GUI	AlmaLinux 8	CentOS 7	Red Hat Enterprise Linux (RHEL)	SLES 15 / CRAY





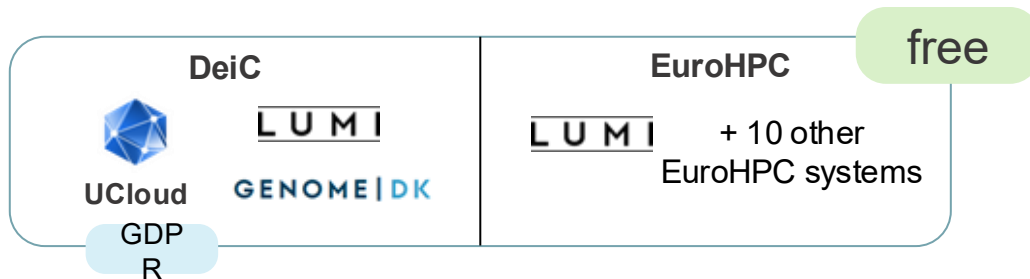
# access to HPC resources

## External

**Via KU IT**  
(DeiC membership)



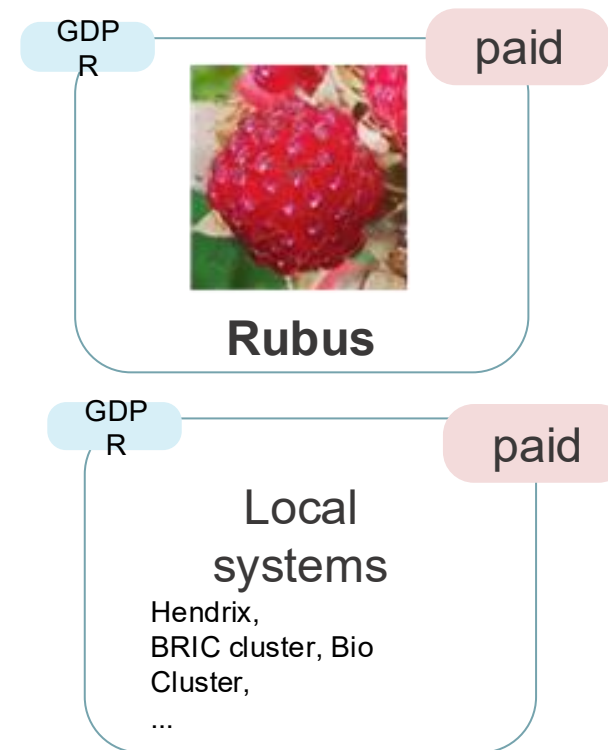
**Via Grants**



**Via direct contract**  
(you pay)



## In-house



# access to HPC resources

	UCloud Type 1	GenomeDK Type 2	LUMI Type 5	RUBUS local KU
User base	Users <b>unfamiliar</b> with HPC	Users familiar with HPC	Users <b>very</b> 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

## 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.



# 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

SDU-eScience / UCloud

Code Issues 64 Pull requests 1 Actions Security Insights

UCloud Public Watch 9 Fork 8

mas... Go to file Code

**DanThrane** Merge pull request #469... 4431b37 · 20 hours ago 15,821 Commits

.github/workflows	Automatic building of IM2 RPMs ...	2 months ago
backend	IM2/K8s: Consider node conditi...	2 days ago
docs	visualization begun	10 months ago
frontend-web	Bump vite from 6.2.3 to 6.2.4 in ...	2 days ago
infrastructure	Merge branch 'master' into dev-...	11 months ago
integrated-applications	IM2/K8s: Tweaks to IPs, SSH an...	3 weeks ago
integration-test	missing sh	11 months ago
launcher-tool-go	IM2/K8s: Use search index	20 hours ago
launcher-tool	IM2/K8s: Use search index	20 hours ago

**About**

[docs.cloud.sdu.dk](#)

Readme

EUPL-1.2 license

Activity

Custom properties

25 stars

9 watching

8 forks

Report repository

**Releases** 18


v2025.3.2 Latest 5 days ago

+ 17 releases

# interactive HPC with UCloud

<https://cloud.sdu.dk>

To access *UCloud* please choose your login provider

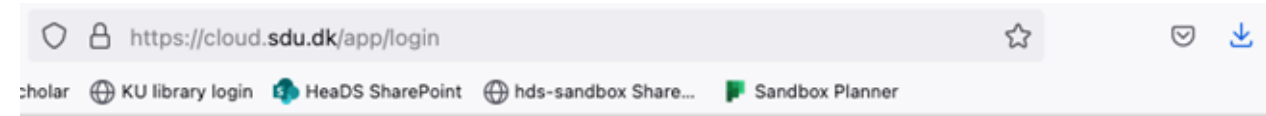
**SDU**   
University of Copenhagen

☐ Always use the login provider that I choose now. At [my.wayf.dk](https://my.wayf.dk) I can res  
use a different login provider.

Search here

1. Search &  
then click on  
link

2. Sign-in via  
KU portal



**DeiC**

Integration Portal

WAYF  Login

Other login options →



# interactive HPC with UCloud

Health Data Science Sa... ▾

## GPU Node Capacity Expanded with new NVIDIA H100s

The DeiC Interactive HPC system at SDU has doubled its GPU resources. 14:00 13/08/2024

The DeiC Interactive HPC system at SDU has enhanced its capabilities with the addition of four new GPU nodes. Each node is equipped with four NVIDIA H100 GPUs, featuring 80 GB of VRAM per GPU. This upgrade brings the total number of GPU nodes equipped with NVIDIA H100s to eight, accessible by selecting the u3-gpu product on UCloud.

The expanded capacity supports high-demand tasks such as machine learning, data analysis, and artificial intelligence, which require substantial computing power for processing large datasets and complex algorithms.

Provided by the AAU, AU, SDU consortium in collaboration with DeiC

### Resource allocations

	uc-general-h	179 / 59K Core-hours
	uc-t4-h	1 / 589 GPU-hours
	u1-standard-h	10K / 175K Core-hours

### Recent runs

test old	24/09/2024	✓
test	19/09/2024	✓
test old	19/09/2024	✓

64



# interactive HPC with UCloud

The screenshot displays the UCloud interactive HPC dashboard. On the left is a vertical sidebar with icons for navigation: a cube, a folder, a group of people, a plus sign, a shopping bag, and a server rack. At the bottom of the sidebar are icons for a sun, a notification bell with a red '64' badge, a speech bubble, and a user profile.

**Spotlight: Quantum computing**

- NVIDIA CUDA-Q Platform**  
Model and toolchain for hybrid quantum-classical computers.
- NVIDIA cuQuantum Appliance**  
Highly performant multi-GPU multi-node solution for quantum circuit simula...

*The NVIDIA CUDA Quantum Platform is an innovative framework designed for hybrid quantum-classical computing. This platform is unique in its ability to unify the computing power of CPUs, GPUs, and Quantum Processing Units (QPUs), offering a seamless integration of these diverse processing capabilities.*

**Browse by category**

Hippo	Engineering	Data Analytics
Artificial Intelligence	Quantum Computing	Social Science
Natural Science	Applied Science	Development
Virtual Machines	Digital Humanities	Health Science



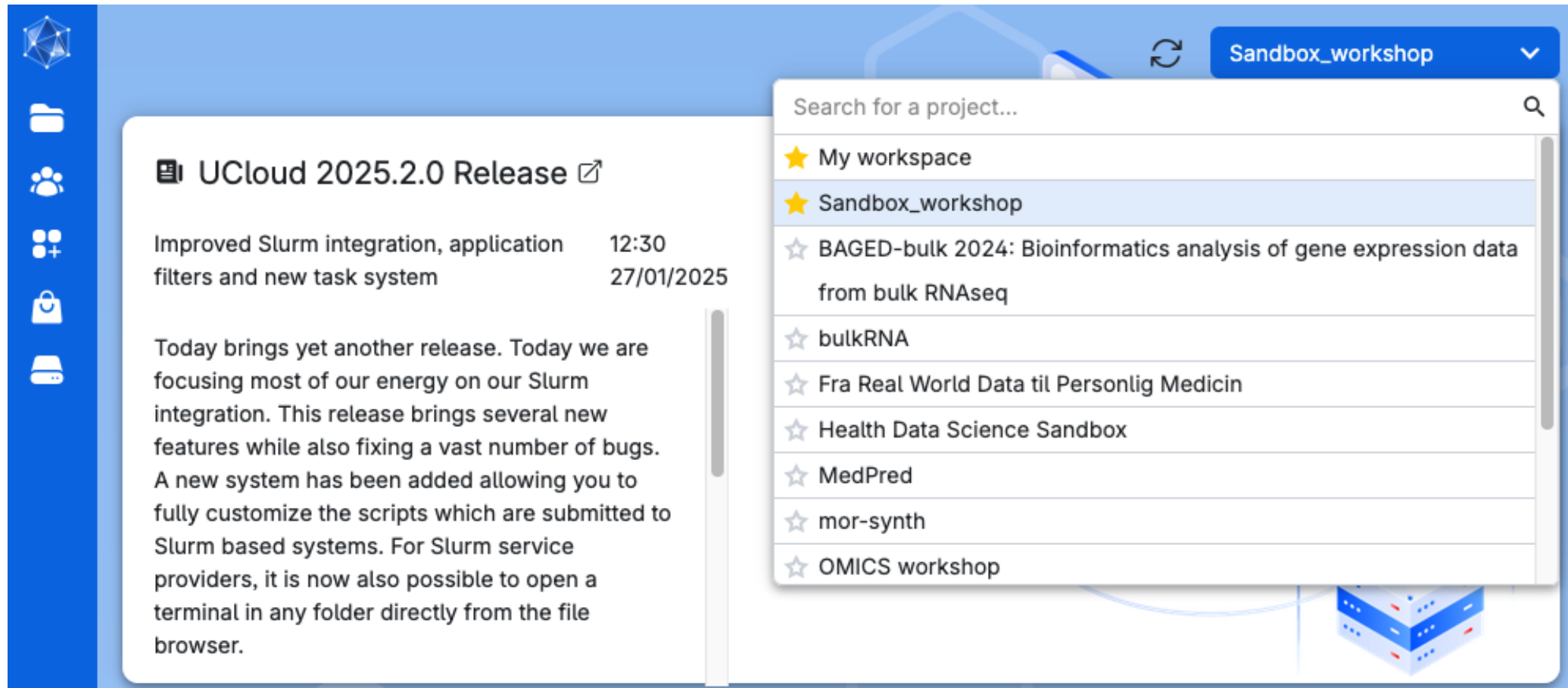
## interactive HPC with UCloud

Apps can be starred  
for easy access





# workspaces



The screenshot displays the UCloud 2025.2.0 Release announcement on the left and a workspace search dropdown on the right. The dropdown lists several projects, with 'Sandbox\_workshop' highlighted.

**UCloud 2025.2.0 Release**

Improved Slurm integration, application filters and new task system 12:30 27/01/2025

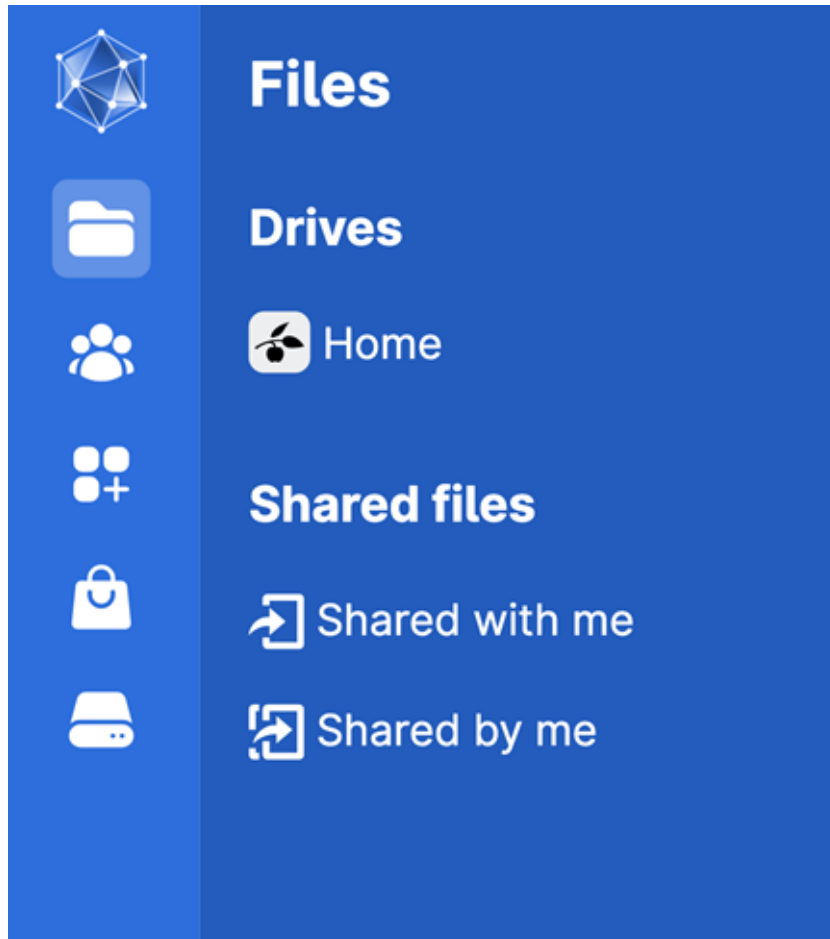
Today brings yet another release. Today we are focusing most of our energy on our Slurm integration. This release brings several new features while also fixing a vast number of bugs. A new system has been added allowing you to fully customize the scripts which are submitted to Slurm based systems. For Slurm service providers, it is now also possible to open a terminal in any folder directly from the file browser.

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



# 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 

1

 Transcriptomics Sandbox ★  
2023.11 ▼

2

[Documentation](#)  [Sandbox RNAseq works...](#) ▼

Transcriptomics Sandbox with modules and courses.

E-mail notification settings  
**Do not notify me** ▼

**Estimated cost**  
**Current balance**

8 Core-hours  
22,98K Core-hours

Job name  
Example: Run with parameters XYZ

Hours \*  
1 +1 +8 +24

Machine type \*  
u1-standard-8

4

vCPU	Memory (GB)	GPU	Price
8	48	None	8 Core-hours/hour

5

Select folders to use  
If you need to use your [files](#) in this job then click "Add folder" to select the relevant files.

Add folder

6

Mandatory Parameters  
Select a module \*

Optional Parameters  
Cirrocumulus H5AD dataset

Use

# 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



test is now running (ID: 5201118)



Open terminal



Open interface



Stop application

## Time allocation

Job submitted at: 09:27 01/04/2025

Job start: 09:33 01/04/2025

Job expiry: 10:33 01/04/2025

Time remaining: 00:55:30

Extend allocation (hours):

+1

+8

+24

## Messages

[09:27] kcs305kcs305#7929 has requested 1x u1-standard-1 from Deic Interactive HPC (SDU/K8s)

[09:27] Assigned to nodeaa-05

[09:27] Job is starting soon

[09:33] Job has started

## Node 1

Downloaded: 1 files, 37M in 4.2s (8.71 MB/s)

rm: cannot remove '/work/Intro\_to\_bulkRNAseq/\_\_MACOSX': No such file or directory

```
=====
= Start RStudio =
=====
```

[s6-init] making user provided files available at /var/run/s6/etc...exited 0.

[s6-init] ensuring user provided files have correct perms...exited 0.

[fix-attrs.d] applying ownership & permissions fixes...

[fix-attrs.d] done.

[cont-init.d] executing container initialization scripts...

[cont-init.d] 01\_set\_env: executing...

skipping /var/run/s6/container\_environment/HOME

[cont-init.d] 01\_set\_env: exited 0.

[cont-init.d] 02\_userconf: executing...

Skipping authentication as requested

[cont-init.d] 02\_userconf: exited 0.

[cont-init.d] done.

[services.d] starting services

[services.d] 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



test is now running (ID: 5201118)



Open terminal



Open interface

Stop application

## Time allocation

Job submitted at: 09:27 01/04/2025  
Job start: 09:33 01/04/2025  
Job expiry: 10:33 01/04/2025  
Time remaining: 00:55:30  
Extend allocation (hours):

+1

+8

+24

## Messages

[09:27] kcs305kcs305#7929 has requested 1x u1-standard-1 from Deic Interactive HPC (SDU/K8s)  
[09:27] Assigned to nodeaa-05  
[09:27] Job is starting soon  
[09:33] Job has started

## Node 1

Downloaded: 1 files, 37M in 4.2s (8.71 MB/s)  
rm: cannot remove '/work/Intro\_to\_bulkRNAseq/\_\_MACOSX': No such file or directory

=====  
= Start RStudio =  
=====

```
[s6-init] making user provided files available at /var/run/s6/etc...exited 0.  
[s6-init] ensuring user provided files have correct perms...exited 0.  
[fix-attrs.d] applying ownership & permissions fixes...  
[fix-attrs.d] done.  
[cont-init.d] executing container initialization scripts...  
[cont-init.d] 01_set_env: executing...  
skipping /var/run/s6/container_environment/HOME  
[cont-init.d] 01_set_env: exited 0.  
[cont-init.d] 02_userconf: executing...  
Skipping authentication as requested  
[cont-init.d] 02_userconf: exited 0.  
[cont-init.d] done.  
[services.d] starting services  
[services.d] done.  
[
```



Runs



Running jobs




test




samuele\_test




# 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 

1

 Transcriptomics Sandbox ★  
2023.11 ▼

2

[Documentation](#)  [Sandbox RNAseq works...](#) ▼

Transcriptomics Sandbox with modules and courses.

E-mail notification settings  
**Do not notify me** ▼

Estimated cost  
Current balance

8 Core-hours  
22,98K Core-hours

Job name

Example: Run with parameters XYZ

Hours \*  
1 +1 +8 +24

Machine type \*

u1-standard-8

4

vCPU	Memory (GB)	GPU	Price
8	48	None	8 Core-hours/hour

5

Select folders to use

If you need to use your [files](#) in this job then click "[Add folder](#)" to select the relevant files.

[Add folder](#)

6

Mandatory Parameters

Select a module \*

Optional Parameters

Cirrocumulus H5AD dataset

[Use](#)

# what is GenomeDK?

GenomeDK is Aarhus University's high-performance computing cluster, especially strong for genomics and health/life-science workloads. It is open to researchers across Danish universities, the public sector, and SMEs.

It is ISO 27001 certified and approved to store sensitive data in compliance with GDPR and the Danish Data Protection Act, which is what makes it suitable for real patient/health data.

Access is organised into zones. Most users belong to the "open" zone, while sensitive projects run in restricted ("closed") zones.



## connecting to GenomeDK?

You interact with the cluster over **SSH** (Secure Shell), an encrypted terminal session on a remote machine. There's no screen or keyboard to walk up to; you connect from your own computer. First you need an account (request it via the user-request form) and you must set up two-factor authentication before your first login.

### Connect from an IDE:

In VS Code, install the Remote–SSH extension, choose "Remote-SSH: Connect to Host...", and enter [<username>@login.genome.au.dk](ssh://<username>@login.genome.au.dk). You then edit files and run code on GenomeDK as if they were local, with a real editor, terminal, and extensions.

### Connect from a terminal:

```
# this drops you onto the login  
node, which should be your staging  
area, not where heavy work runs
```

```
ssh <username>@login.genome.au.dk
```





# GenomeDK configuration

GenomeDK uses Slurm to organize compute resources. You ask Slurm for cores/memory/time, and it places your work on a compute node. Remember to always charge exercises to our project account: **DeiC-KU-L65**

```
# interactive session (for debugging)
```

```
srun -account=DeiC-KU-L65 -mem=8g -cores=2 -time 02:00:00
```

```
# batch job (for real, long-running work). Write an sbatch script and submit it:
```

```
sbatch my job.sh #should contain #SBATCH lines and your commands
```

