DeiC Interactive HPC
A Guide

DANISH eINFRASTRUCTURE COOPERATION

EURO

Rasmus Berg Jensen

EuroCC

February 14, 2022

Preface

The intended purpose of this guide is two-fold:

1. Provide a practical guide to get started as simply as possible, which is the content of Chapter 1.
2. Provide some useful background information for new users of High-Performance Computing (HPC) systems, that will enhance their experience and proficiency with HPC. That is the content of Chapter 2.

Following the guide new user's will hopefully obtain a proficiency with the system such that HPC becomes a comfortable and useful part of their research toolkit. Experienced users will in this guide hopefully find the information necessary to get started quickly. For this reason the guide is designed such that one can skip large parts and find the needed information in a concise and efficient manner.

Good luck on your adventures in the world of high-performance computing.

Contents

Chapter 1

DeiC Interactive HPC

This chapter is designed such that, in many cases is sufficient to look at the figures and the figure texts while skipping the main text. The purpose of the main text is to provide clarification and additional information. As most steps are pretty trivial, feel free to skip as much as you like.

If you do not feel this guide answers all of your questions please consult the UCloud Docs.

1.1 Logging in

To access DeiC Interactive HPC go to the website `cloud.sdu.dk` where your will find the login screen displayed in Fig. 1.1. From this front page, a green button redirects you to a screen like that shown in Fig. 1.2 where DeiC's system Where Are You From (WAYF) enables you to log in through your organization. As indicated in Fig. 1.2 use the search bar indicated by the red arrow to find your organization. By clicking the link of your organization, you will be redirected to the access portal of that organization, where you can log in in the same way as for all other services that use the portal. If you have used WAYF before, previous selections will be shown in the position indicated by the blue arrow.

The first time a new user logs in, the user will be presented with the terms of service that must be accepted before proceeding. Once accepted, one will arrive at the UCloud home screen, indicating a successful login.

1.2 Home screen

Having logged in, one will see the ones home screen like the one displayed in Fig. 1.3. Several tabs display different sorts of information on the home screen, e.g., in the "Recent Runs" tab; one can see the most recent jobs one has run and navigate directly to them. In the figure, arrows point to the system documentation and search bar. Most of the user's time will be spent in the directories highlighted with a green box. The "Files" directory shows all the user's files, including job results, "Apps" shows all available applications from where jobs can be started, and "Runs" lists currently running and previous jobs. More on that in Sections 1.3 and 1.4..

The "Resources" and "Usage" tabs are essential to keep an eye on as these indicate the amounts of resources available and used last 30 days. This is seen in Fig. 1.4 with colored boxes indicating the amount of computing and storage resources available within the storage quota. With DeiC Interactive

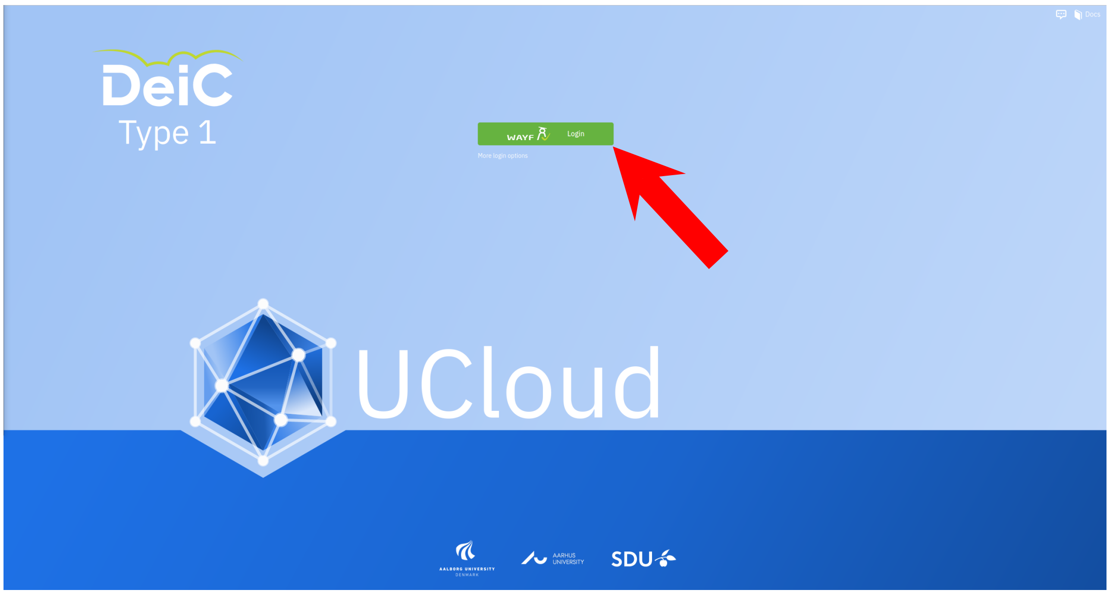
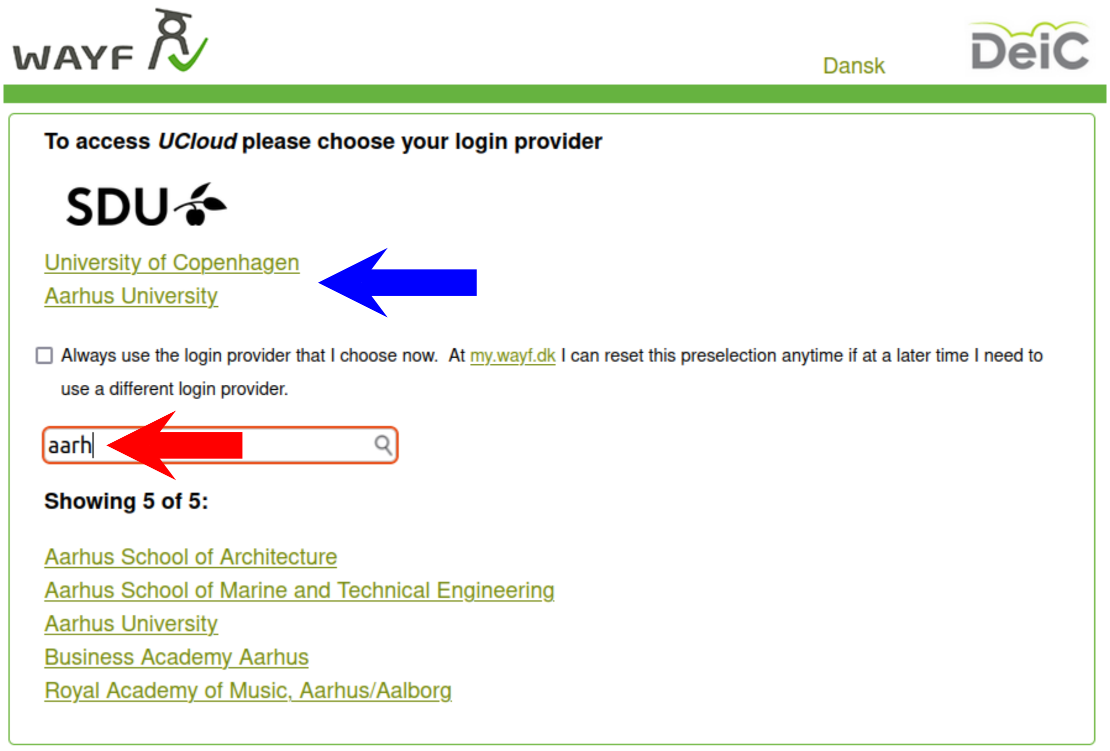


FIGURE 1.1: Login screen for DeiC Interactive HPC. Click on the green button indicated by the red arrow.



To access *UCloud* please choose your login provider

University of Copenhagen
Aarhus University

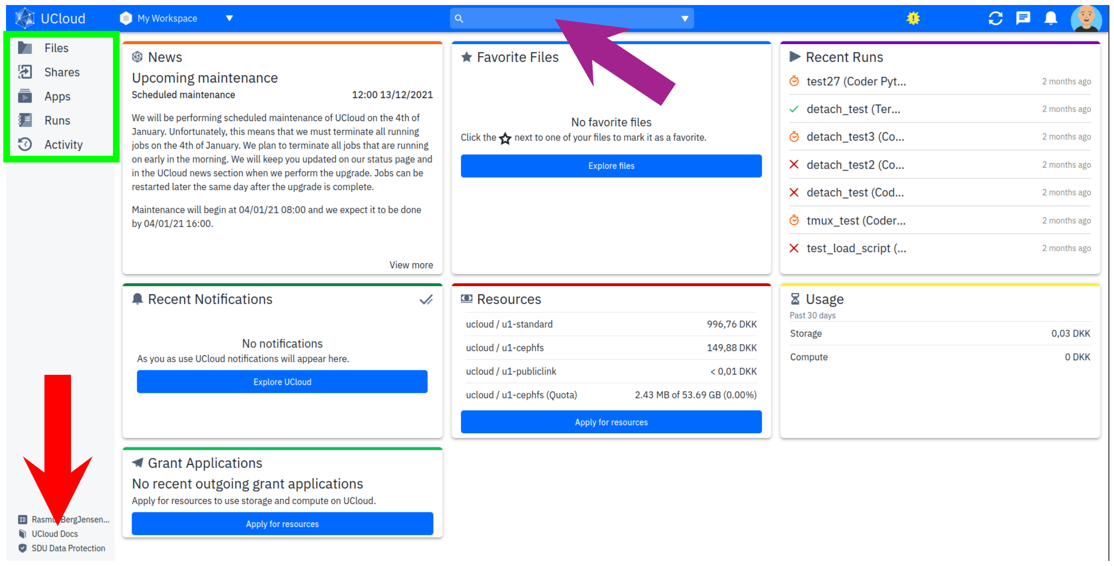
☐ Always use the login provider that I choose now. At my.wayf.dk I can reset this preselection anytime if at a later time I need to use a different login provider.

aarh

Showing 5 of 5:

Aarhus School of Architecture
Aarhus School of Marine and Technical Engineering
Aarhus University
Business Academy Aarhus
Royal Academy of Music, Aarhus/Aalborg

FIGURE 1.2: Login provider screen the button in Fig. 1.1 redirects to. Type a part of the organization's name through which you have access in the field indicated by the red arrow. In the case of this figure, "aarh" for Aarhus University has been typed. The blue arrow indicates the position where previous selections are shown – here, University of Copenhagen and Aarhus University.



News

Upcoming maintenance

Scheduled maintenance 12:00 13/12/2021

We will be performing scheduled maintenance of UCloud on the 4th of January. Unfortunately, this means that we must terminate all running jobs on the 4th of January. We plan to terminate all jobs that are running on early in the morning. We will keep you updated on our status page and in the UCloud news section when we perform the upgrade. Jobs can be restarted later the same day after the upgrade is complete.

Maintenance will begin at 04/01/21 08:00 and we expect it to be done by 04/01/21 16:00.

Favorite Files

No favorite files

Click the ☆ next to one of your files to mark it as a favorite.

Recent Runs

test27 (Coder Pyt...	2 months ago
detach_test (Ter...	2 months ago
detach_test3 (Co...	2 months ago
detach_test2 (Co...	2 months ago
detach_test (Cod...	2 months ago
tmux_test (Coder...	2 months ago
test_load_script (...	2 months ago

Recent Notifications

No notifications

As you as use UCloud notifications will appear here.

Resources

ucloud / u1-standard	996,76 DKK
ucloud / u1-cephfs	149,88 DKK
ucloud / u1-publiclink	< 0,01 DKK
ucloud / u1-cephfs (Quota)	2.43 MB of 53.69 GB (0.00%)

Usage

Past 30 days

Storage	0,03 DKK
Compute	0 DKK

Grant Applications

No recent outgoing grant applications

Apply for resources to use storage and compute on UCloud.

FIGURE 1.3: Home screen at UCloud. The red arrow points to a link to the system's documentation, and the purple indicates the search bar. A green box highlights the directories where most of the user's time is usually spent. Since this screenshot was taken the Resources and Usage tabs have changed slightly. Figure 1.4 shows an updated version as of early Februrary 2022.

Type	Name	Amount
Standard compute node time	`u1-standard`	1000 DKK
Storage	`u1-cephfs`	50 GB

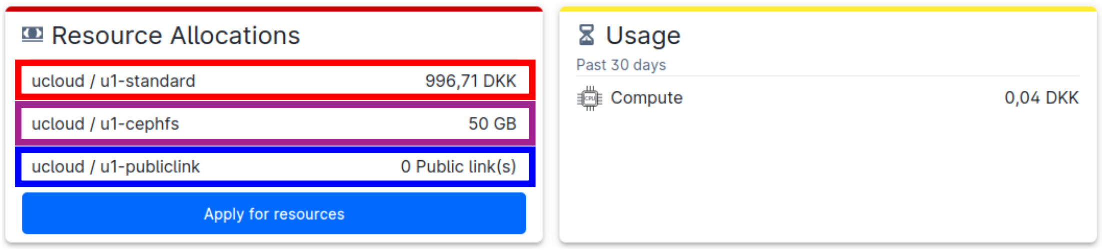
TABLE 1.1: Resources available to researchers and students at the Danish universities.

HPC, compute time and storage space determine the cost of use with prices depending on the hardware used. Students and researchers at the Danish universities have the resource amounts displayed in Table 1.1 available. The other types of compute nodes available can be seen in the documentation. The documentation also contain some video tutorial which supplement the material covered here. These can be seen here.

1.3 File system

Clicking the Files tab in the green box in Fig. 1.3 one can see the files stored on that account. One can have multiple drives, e.g., for multiple projects, but we stick to the Home directory in this guide. An example of how the Home directory might look is seen in Fig. 1.5.

The tools to create and upload files and folders are seen in the red box, while the purple arrows point out the Jobs directory. This contains the files generated during all jobs sorted by the app used. An example of a subdirectory of Jobs, in this case for the Python app, is seen in Fig. 1.6. The figure shows a list of folders, one for each completed job, folders that have been given the name of the job followed by a job ID. For efficient navigation, it is thus essential to use indicative and distinct job names, such that


Resource Allocations

ucloud / u1-standard	996,71 DKK
ucloud / u1-cephfs	50 GB
ucloud / u1-publiclink	0 Public link(s)

Apply for resources

Usage
Past 30 days

Compute	0,04 DKK

FIGURE 1.4: Overview of the resource tabs. The yellow tab on the right displays the resources one has used in the last 30 days. The red tab on the left displays several things. The red box shows the remaining computing resources on UCloud standard nodes (`u1-standard`). The remaining amount of storage in the Ceph File System (`u1-cephfs`) is displayed in the purple box, whereas the blue box shows the number of public links in use. If it is the first time login in into the system, this will not be present.

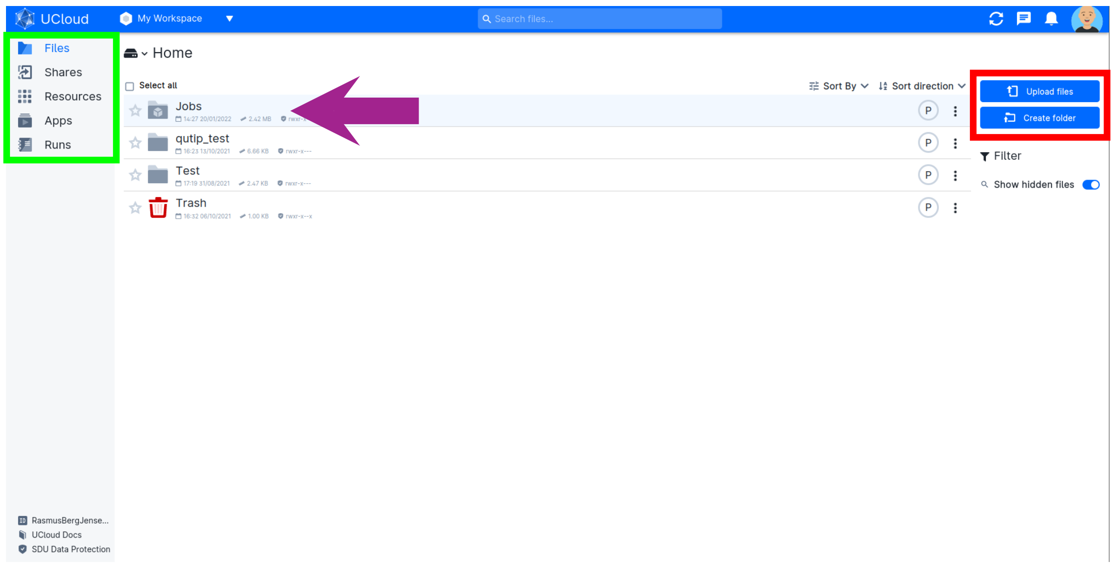


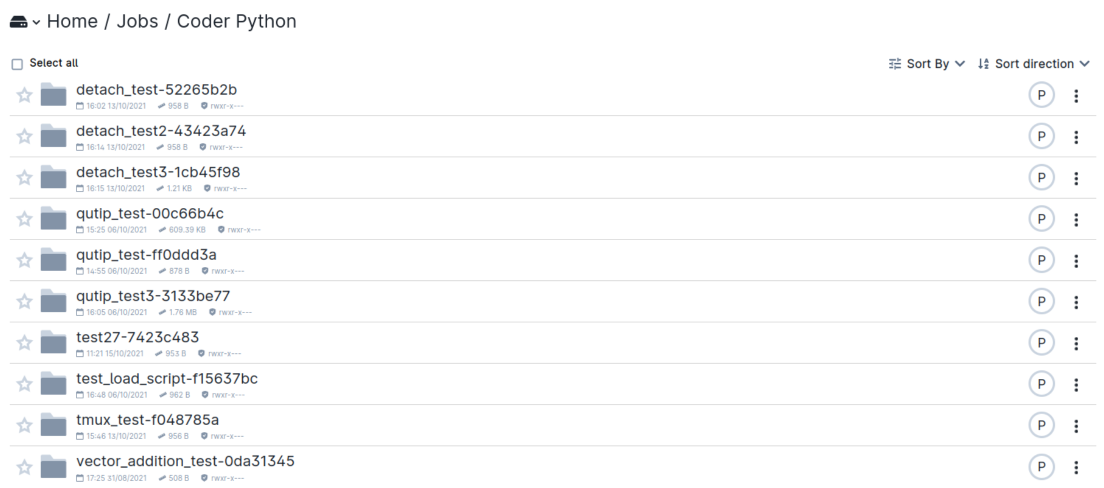
FIGURE 1.5: Example of how the Home directory in Files might look. Files can be uploaded in a drag-and-drop fashion using the button in the red box. The purple arrow indicates the Jobs directory which contains the output from finished jobs.

these folders are distinct contrary to the figure. Completed jobs can also be seen in the Runs tab found in the green box of Fig. 1.5.

Notice that the Files tab is not designed for the user to edit files – it is view only. Editing code and files is done locally on the user's personal computer or in a job.

1.4 Running jobs

To do anything with the system, one needs to create a job to access computing resources. This is done in the Apps menu by selecting the desired software. This guide will use Python and MATLAB as open source and proprietary software examples, respectively. Figures 1.7 and 1.8 shows how this is done. If you cannot find the desired software see Section 1.4.3. Notice that the popular programming languages C++, Java, and Python come with the Visual Studio Code editor environment, and the apps therefore



Home / Jobs / Coder Python

Select all

Sort By · Sort direction

detach_test-52265b2b
16:02 13/10/2021 · 958 B · rwxr-x---

detach_test2-43423a74
16:14 13/10/2021 · 958 B · rwxr-x---

detach_test3-1cb45f98
16:15 13/10/2021 · 1.21 KB · rwxr-x---

qutip_test-00c66b4c
15:25 06/10/2021 · 609.39 KB · rwxr-x---

qutip_test-ff0ddd3a
14:55 06/10/2021 · 878 B · rwxr-x---

qutip_test3-3133be77
16:05 06/10/2021 · 1.76 MB · rwxr-x---

test27-7423c483
11:21 15/10/2021 · 953 B · rwxr-x---

test_load_script-f15637bc
16:48 06/10/2021 · 962 B · rwxr-x---

tmux_test-f048785a
15:46 13/10/2021 · 956 B · rwxr-x---

vector_addition_test-0da31345
17:25 31/08/2021 · 508 B · rwxr-x---

FIGURE 1.6: Example of how a folder in the Jobs directory might look. Notice the structure of the folder names: job name followed by a job ID. Since this user has been sloppy and given multiple jobs the same indistinct name, it is challenging to navigate the jobs and find specific files. *Do not make the same mistake!*

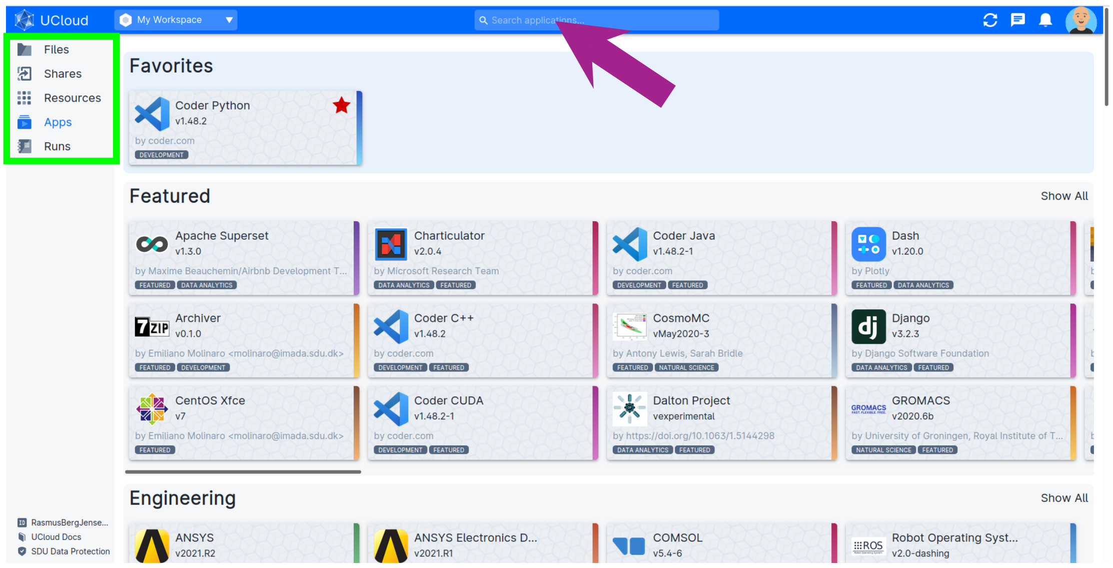

UCloud
My Workspace
Search applications...
Files
Shares
Resources
Apps
Runs
Favorites
Coder Python v1.48.2 by coder.com DEVELOPMENT
Featured
Show All
Apache Superset v1.3.0 by Maxime Beauchemin/Airbnb Development T... FEATURED DATA ANALYTICS
Charticulator v2.0.4 by Microsoft Research Team DATA ANALYTICS FEATURED
Coder Java v1.48.2-1 by coder.com DEVELOPMENT FEATURED
Dash v1.20.0 by Plotly FEATURED DATA ANALYTICS
Archiver v0.1.0 by Emiliano Molinaro <molinaro@imada.sdu.dk> FEATURED DEVELOPMENT
Coder C++ v1.48.2 by coder.com DEVELOPMENT FEATURED
CosmoMC vMay2020-3 by Antony Lewis, Sarah Bridle FEATURED NATURAL SCIENCE
Django v3.2.3 by Django Software Foundation DATA ANALYTICS FEATURED
CentOS Xfce v7 by Emiliano Molinaro <molinaro@imada.sdu.dk> FEATURED
Coder CUDA v1.48.2-1 by coder.com DEVELOPMENT FEATURED
Dalton Project vexperimental by https://doi.org/10.1063/1.5144298 DATA ANALYTICS FEATURED
GROMACS v2020.6b by University of Groningen, Royal Institute of T... NATURAL SCIENCE FEATURED
Engineering
Show All
ANSYS v2021.R2
ANSYS Electronics D... v2021.R1
COMSOL v5.4-6
Robot Operating Syst... v2.0-dashing
RasmusBergJense...
UCloud Docs
SDU Data Protection

FIGURE 1.7: The Apps tab contains all the applications installed on UCloud sorted by research fields. The top apps marked as favorites are displayed, and the purple arrow indicates a search bar.

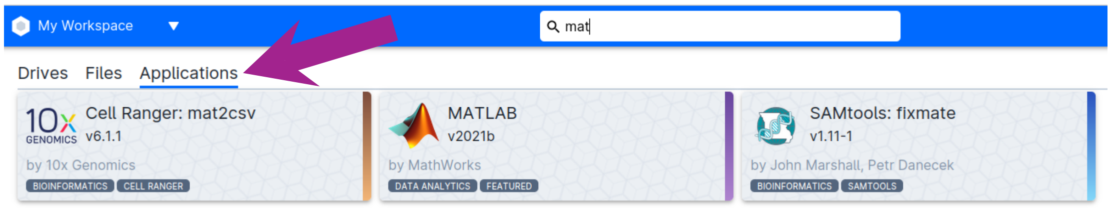


FIGURE 1.8: Searching for an apps will yield a result like this. Notice that you might need to manually select the Applications search target as indicated by the purple arrow.

have "Coder" prepended to their name. This can be seen in Fig. 1.7. Selecting the desired software the takes the user to the job creation screen, where there are three main things to do:

1. Naming the job in an indicative way such that by the name alone, one knows what the purpose of that job is. This is important because the name identifies running and completed jobs. A good name makes it easy to find the output of specific jobs upon completion.

2. Selecting an appropriate amount of resources. Keep in mind that the user pays for what they ask for – not what they actually use! Therefore one should only request the number of CPUs that the program one intends to run can use. Concerning job time, one should select enough time to comfortably do what one intends to do, but not so much as to waste valuable compute time. One can always manually cancel a job.

3. Select the files to be transferred to the job. Inside the job, one only has access to the requested software and the directories specified in the job request. In most cases, one already has a working program that one intends to run during the job. If the program files and dependencies are not listed to be transferred to the job, they will not be available.

Figure 1.9 shows the job creation screen for the Coder Python app. Once The green box has been filled, the estimated cost of the job and the current balance will be shown where the purple arrow is in Fig. 1.9.

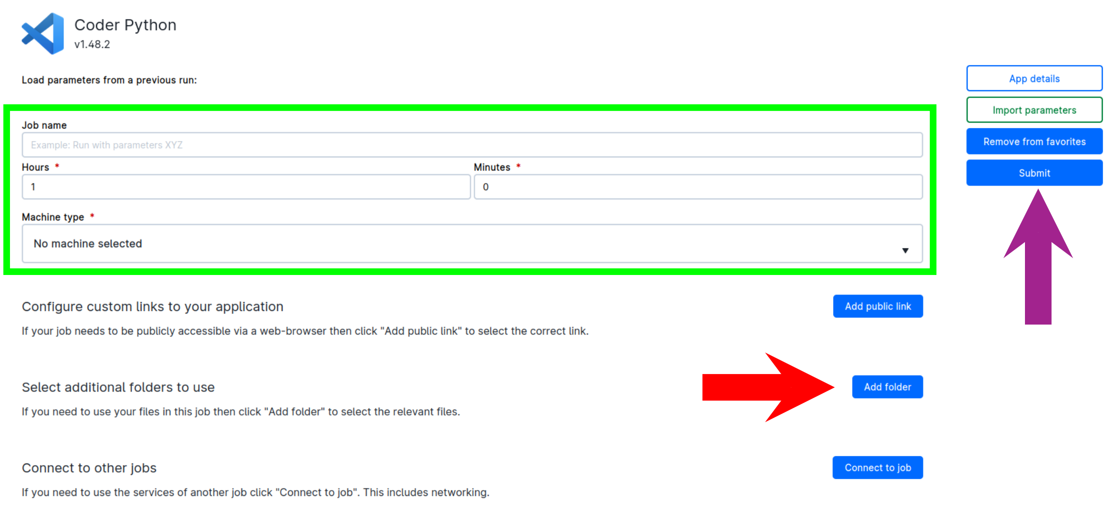
1.4.1 Open source programs

When a job request is sent, it is placed in a queue waiting for the requested resources. In the experience of the author, this is usually a matter of seconds[1] and thus not a practical inconvenience. When the requested resources are granted and the job starts, the user will, in most cases, have two options to run the requested software: a standard Linux terminal or an interactive interface. This is seen in Fig. 1.10.

In the case of Python, the interface is Visual Studio Code, as Fig. 1.11 shows. This runs just like it would on a personal computer. The terminal works like any Linux terminal and can be seen in Fig. 1.13a.

As programs like Python have a vast catalog of libraries for various tasks, not all libraries are installed by default. Missing libraries can be installed through the terminal, e.g.,

1: With the trial access, one has access to 1 u1-standard node, i.e., 64 CPU-cores, per job, and no GPU partitions. As this is the access type the author has used, no claim is made with regards to the waiting time for multinode jobs.



Coder Python
v1.48.2

Load parameters from a previous run:

Job name: Example: Run with parameters XYZ

Hours *: 1

Minutes *: 0

Machine type *: No machine selected

Configure custom links to your application

If your job needs to be publicly accessible via a web-browser then click "Add public link" to select the correct link.

Select additional folders to use

If you need to use your files in this job then click "Add folder" to select the relevant files.

Connect to other jobs

If you need to use the services of another job click "Connect to job". This includes networking.

FIGURE 1.9: Creating a job is done by filling out the information in the green box adding the required files by pressing the button pointed to by the red arrow before submitting the job with the button indicated by the purple arrow. Remember to choose a unique and indicative job name to avoid the situation in Fig. 1.6. The time should be large enough to complete the job comfortably, but not much longer to avoid wasting compute time. The machine type should be selected based on the hardware needs of the job. See Section 2.2 for help with hardware selection.

```
pip install qutip
```

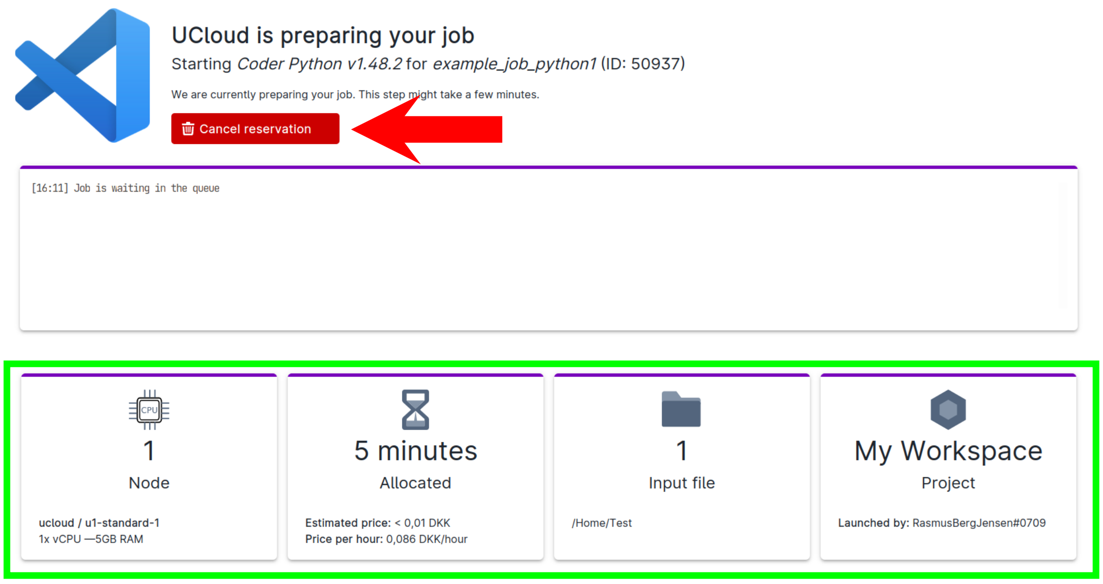
for the Python library QuTiP.

If you have many dependencies for your program it can be an advantage to create a Conda environment on your local computer and make it as an .yml file and upload it to Ucloud in the same directory as you program. See Anaconda docs for details on how to create an environment `https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html`. Conda is preinstalled on the PyTorch and JupyterLab applications, however, on the Coder Python application you have to install it yourself.

1.4.2 Proprietary software

The system is generally set up to allow the user to register a license through a project application. It is, however, possible to access programs through a university-owned license in some cases. To illustrate how this is done, we now proceed with MATLAB as an example; see Fig. 1.12. Initially, one creates a job just like the open-source case. Please notice that there is a license server option in the job creation menu. *This will not be used in this guide.* Once the job is running, the user needs to activate their license. For MATLAB, this is done through the user's Mathworks account, either through the terminal or the interactive interface.

Terminal: When the user executes a MATLAB program, the user name of a Mathworks account will be asked for and subsequently the password connected to that account. Once the user credentials have been accepted, the program will execute as called. This is seen in Fig. 1.13.



UCloud is preparing your job

Starting *Coder Python v1.48.2* for *example_job_python1* (ID: 50937)

We are currently preparing your job. This step might take a few minutes.

```
[16:11] Job is waiting in the queue
```

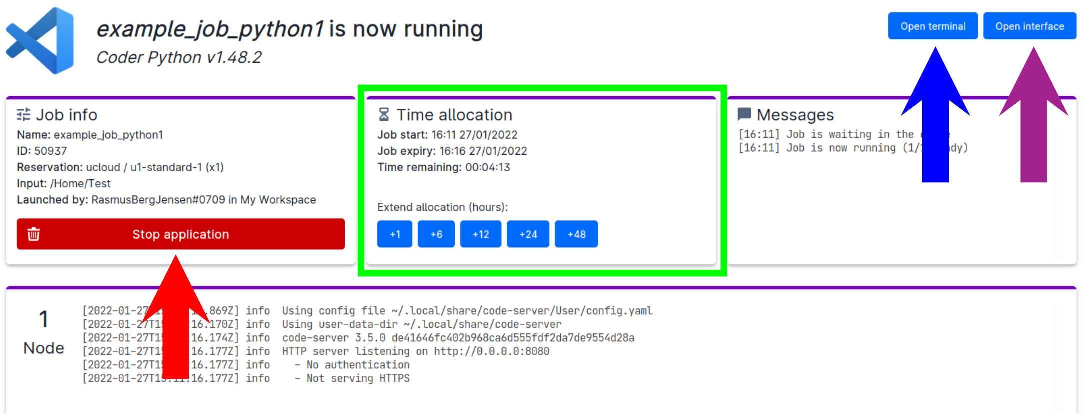
1 Node — ucloud / u1-standard-1, 1x vCPU —5GB RAM

5 minutes Allocated — **Estimated price:** < 0,01 DKK, **Price per hour:** 0,086 DKK/hour

1 Input file — /Home/Test

My Workspace Project — **Launched by:** RasmusBergJensen#0709

(A) Job in queue waiting for resources. The green box highlights the job settings. If these contain a mistake, one can cancel the job with the button pointed to by the red arrow.



***example_job_python1* is now running**

Coder Python v1.48.2

Job info

Name: example_job_python1
ID: 50937
Reservation: ucloud / u1-standard-1 (x1)
Input: /Home/Test
Launched by: RasmusBergJensen#0709 in My Workspace

Time allocation

Job start: 16:11 27/01/2022
Job expiry: 16:16 27/01/2022
Time remaining: 00:04:13

Extend allocation (hours):

Messages

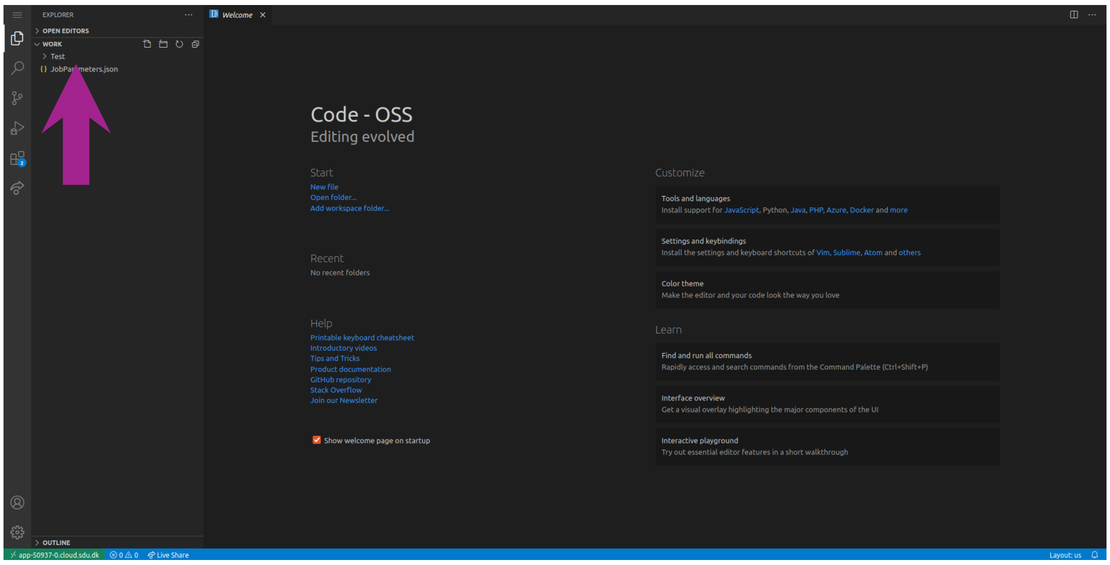
```
[16:11] Job is waiting in the
[16:11] Job is now running (1/    dy)
```

1 Node

```
[2022-01-27      .869Z] info  Using config file ~/.local/share/code-server/User/config.yaml
[2022-01-27T1     16.170Z] info  Using user-data-dir ~/.local/share/code-server
[2022-01-27T1     16.174Z] info  code-server 3.5.0 de41646fc402b968ca6d555fdf2da7de9554d28a
[2022-01-27T1     16.177Z] info  HTTP server listening on http://0.0.0.0:8080
[2022-01-27T1     16.177Z] info    - No authentication
[2022-01-27T1 .  .16.177Z] info    - Not serving HTTPS
```

(B) Running job that has been granted the requested resources. The blue and purple arrows indicate how to open a terminal and the interactive interface, respectively. If one finishes the job before the job time runs out, the job can be stopped with the red button pointed to by a red arrow. In the green box, one can see the remaining time and extend the job if necessary.

FIGURE 1.10: After submitting a job one will see the screen in Fig. 1.10a which turns into Fig. 1.10b ones resources are granted.



Code - OSS
Editing evolved

Start
New file
Open folder...
Add workspace folder...

Recent
No recent folders

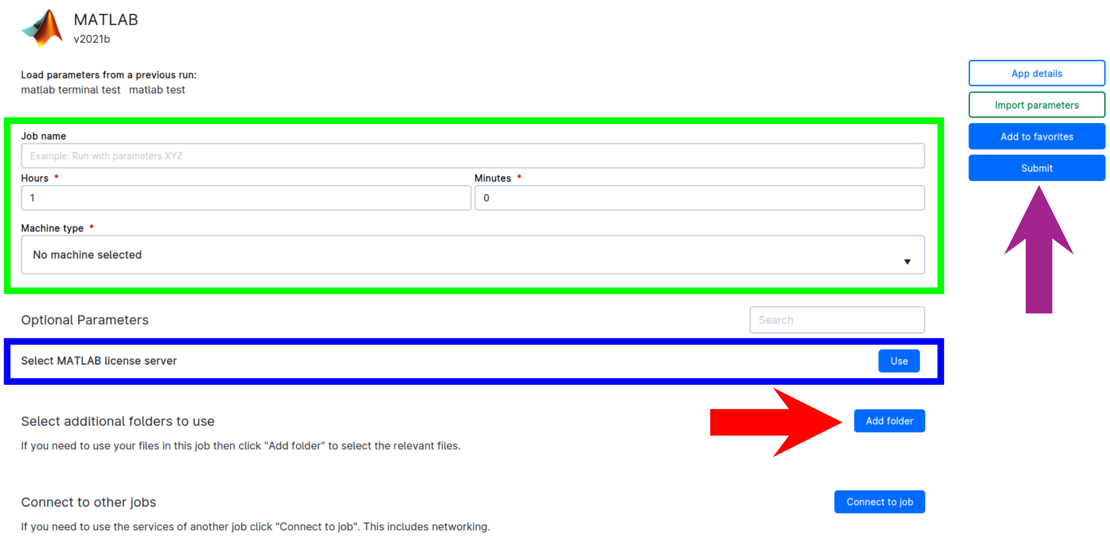
Help
Printable keyboard cheatsheet
Introductory videos
Tips and Tricks
Product documentation
GitHub repository
Stack Overflow
Join our Newsletter

Show welcome page on startup

Customize
Tools and languages
Install support for JavaScript, Python, Java, PHP, Azure, Docker and more
Settings and keybindings
Install the settings and keyboard shortcuts of Vim, Sublime, Atom and others
Color theme
Make the editor and your code look the way you love

Learn
Find and run all commands
Rapidly access and search commands from the Command Palette (Ctrl+Shift+P)
Interface overview
Get a visual overlay highlighting the major components of the UI
Interactive playground
Try out essential editor features in a short walkthrough

FIGURE 1.11: Choosing the interactive interface in Fig. 1.10b one sees this screen. The folders imported to the job will be seen at the location indicated by the purple arrow. From here, it runs just like Visual Studio Code on a personal computer.



MATLAB
v2021b

Load parameters from a previous run:
matlab terminal test   matlab test

Job name
Example: Run with parameters XYZ

Hours * 1
Minutes * 0

Machine type *
No machine selected

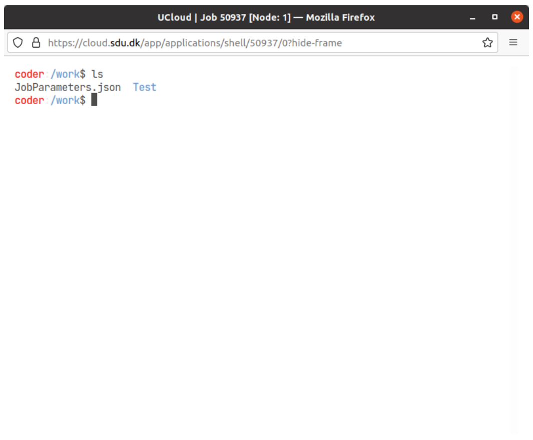
Optional Parameters
Search

Select MATLAB license server

Select additional folders to use
If you need to use your files in this job then click "Add folder" to select the relevant files.

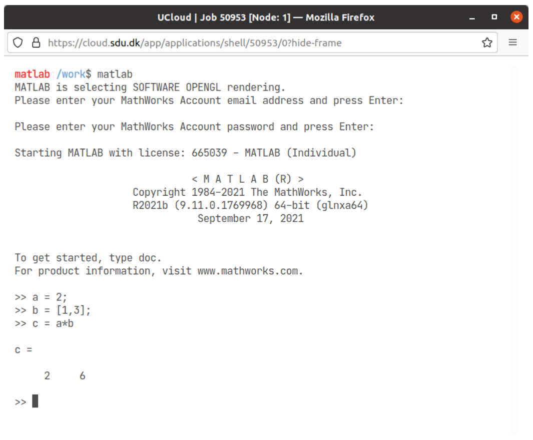
Connect to other jobs
If you need to use the services of another job click "Connect to job". This includes networking.

FIGURE 1.12: Creating a job using proprietary software is done much like in Fig. 1.9 by filling out the information in the green box, adding the required files by pressing the button pointed to by the red arrow before submitting the job with the button indicated by the purple arrow. Remember to choose a unique and indicative job name to avoid the situation in Fig. 1.6. The time should be set so that the job can be completed comfortably, but not much longer to avoid wasting compute time. The machine type should be selected based on the hardware needs of the job. See Section 2.2 for help with hardware selection. The blue box indicates the option of a license server, which we will not use.


UCloud | Job 50937 [Node: 1] — Mozilla Firefox
https://cloud.sdu.dk/app/applications/shell/50937/0?hide-frame

```
coder /work$ ls
JobParameters.json  Test
coder /work$ 
```

(A) The terminal as one sees when starting it. Here, it is a Python job.


UCloud | Job 50953 [Node: 1] — Mozilla Firefox
https://cloud.sdu.dk/app/applications/shell/50953/0?hide-frame

```
matlab /work$ matlab
MATLAB is selecting SOFTWARE OPENGL rendering.
Please enter your MathWorks Account email address and press Enter:

Please enter your MathWorks Account password and press Enter:

Starting MATLAB with license: 665039 - MATLAB (Individual)

                              < M A T L A B (R) >
                    Copyright 1984-2021 The MathWorks, Inc.
                    R2021b (9.11.0.1769968) 64-bit (glnxa64)
                               September 17, 2021


To get started, type doc.
For product information, visit www.mathworks.com.

>> a = 2;
>> b = [1,3];
>> c = a*b

c =

     2     6

>> 
```

(B) Authentication of the user's MATLAB license through the terminal. This is done through a registered Mathworks account.

FIGURE 1.13: The terminal available during jobs.

Interactive interface: Running the interactive interface, a new window with the home screen of a virtual machine will open. This virtual machine contains the requested version of MATLAB and the files imported for the job. This is seen in Fig. 1.14. MATLAB is started by double-clicking on the icon. Once the program has started, the user will be asked for credentials exactly like the terminal case, Fig. 1.15. When the credentials are accepted, MATLAB runs as it does with a local installation, Fig. 1.16.

1.4.3 Missing software

If you need software that is not already installed on the system, you should contact your local HPC Front Office or the Danish division of EuroCC (email: eurocc@listserv.deic.dk). As software is installed on UCloud through a Docker container, it is easier to install if a Docker image exists. If you can find such an image suitable for your needs, it will be easier to help you.

1.5 Getting more resources

There are two options should you run out of computing time and/or storage. If your project is small or you need a bit more to finish, e.g., a Master's project, write an email to either eurocc@listserv.deic.dk or your local HPC front office with a brief description of your project, the resources your need and why you need if. They will be able to allocate the amounts seen in Table 1.1.

For larger projects spanning multiple years and/or involving multiple people, the resources in Table 1.1 will probably be insufficient. In this case, you will need to apply for a project through the UCloud system, or the DeiC calls, which are open twice per year. See the DeiC website.

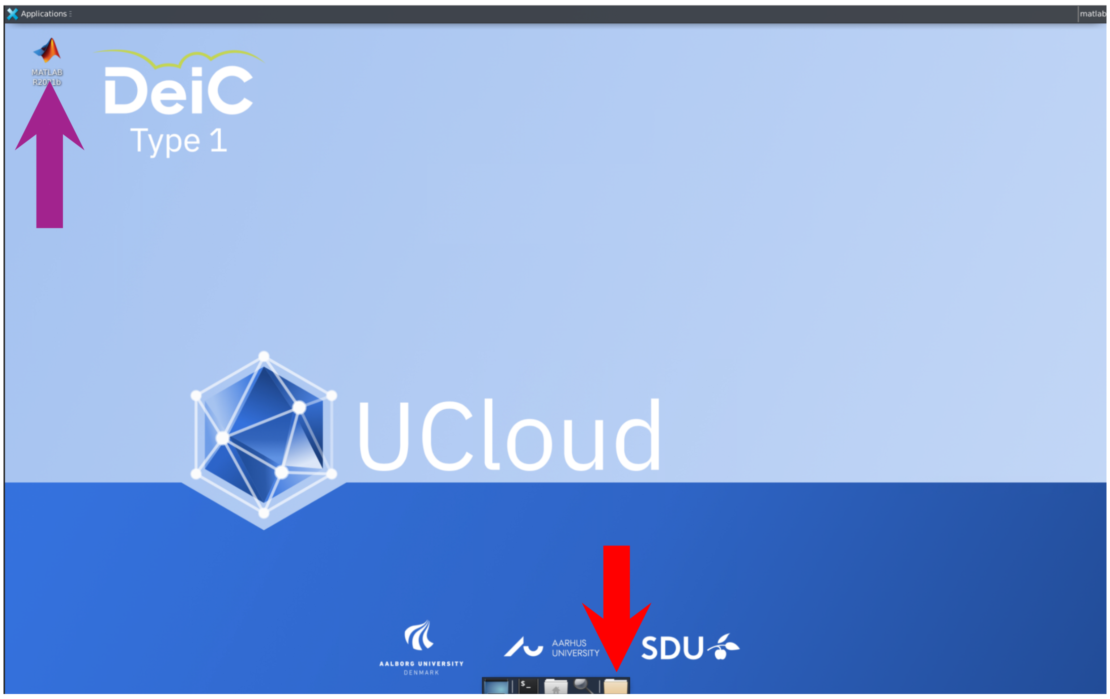
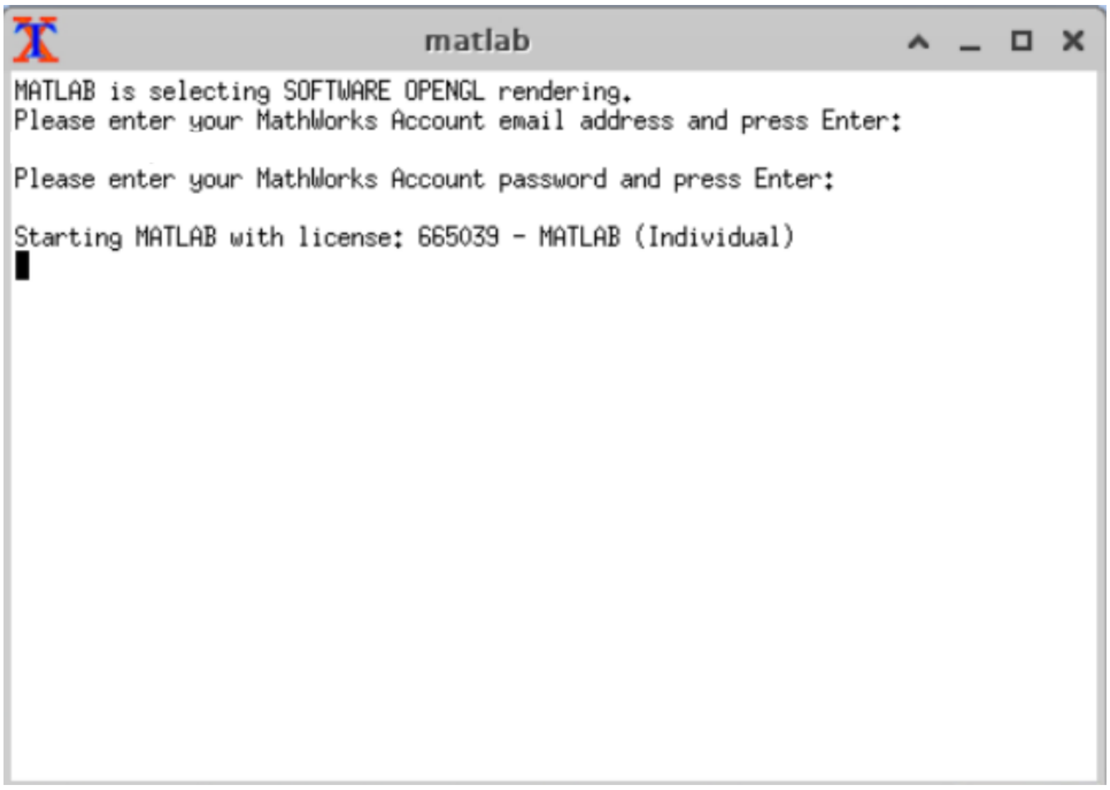


FIGURE 1.14: The home screen of the interactive interface in a MATLAB job. Files uploaded to the job can be seen through the folder icon indicated by the red arrow. MATLAB is started by double-clicking the icon pointed to by the purple arrow.

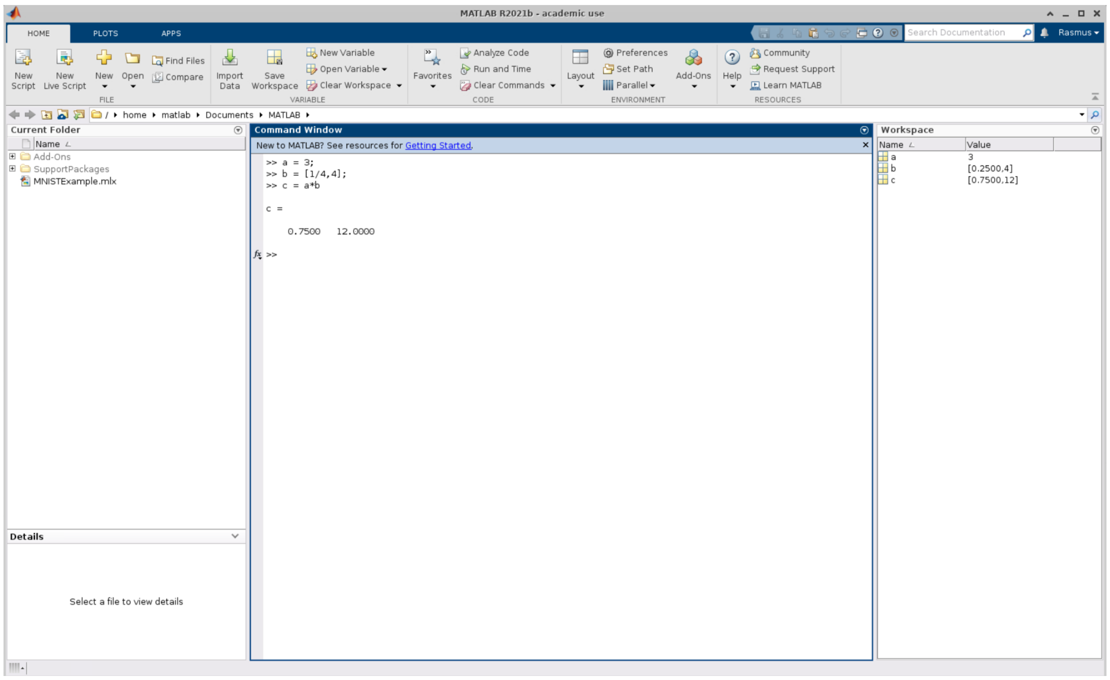


```
MATLAB is selecting SOFTWARE OPENGL rendering.
Please enter your MathWorks Account email address and press Enter:

Please enter your MathWorks Account password and press Enter:

Starting MATLAB with license: 665039 - MATLAB (Individual)
```

Figure 1.15: Once started MATLAB will ask for authentication just like in the terminal case, see Fig. 1.13b.



```
>> a = 3;
>> b = [1/4,4];
>> c = a*b

c =

    0.7500   12.0000

>>
```

Name	Value
a	3
b	[0.2500,4]
c	[0.7500,12]

Figure 1.16: Once started the familiar MATLAB interface is seen.

Chapter 2

High-Performance Computing

This chapter is designed to provide information for new users that can enhance their experience and proficiency with HPC systems. Experienced users will likely be familiar with the contents of this chapter

2.1 What is High-Performance Computing?

High-Performance is a general term for cloud computing with a focus on performance. It works by offloading computations to a central facility from the user's personal computer, with several advantages.

- **Computing power**: A central facility offers more computing power than is available in most personal computers. The number of CPUs (or, more accurately, CPU-cores) and the amount of memory is far greater than on most personal computers.
- **Dedicated hardware**: The hardware one is allocated during a job has the sole purpose of running your job. It does not have to handle several running programs simultaneously like on most personal computers.
- **Convenience**: A great perk of the dedicated hardware is that the user's personal computer does not have to run the code and can thus be used for other tasks or turned off completely.
- **Multiple jobs**: One can run multiple concurrent jobs, each with its dedicated hardware.

HPC facilities are generally designed for medium-sized jobs, which in this context means run times on the scale of hours. It is not particularly suited for jobs that are complete in seconds or frequent small runs during one job. For example, consider a generic project consisting of developing a code for a simulation and running that simulation in several cases before the data is processed into one or several figures. HPC systems are designed for the simulation phase of the project, where the code can run autonomously for long periods. The tinkering one does when creating the perfect figure to showcase the results of the simulations with several runs of slightly tweaked code is something for which HPC systems are ill-suited.

A good workflow can thus be to submit jobs that perform the bulk of computational work while working with other tasks, e.g., creating figures from previously obtained data.

2.2 Hardware selection

The power of HPC-systems lies in the availability of a large number of CPU-cores[1]. A modern laptop CPU typically consists of two or four cores which can each execute code independently. HPC systems often have 32 or 64 cores per CPU and can thus perform 32 or 64 independent tasks simultaneously or in parallel, as it is said in HPC lingo.

The availability of multiple CPU cores poses an important question: "how many cores should one use for a given job?" The simple answer is "as many as one benefits from using". Although simple, it is not very practical advice for the beginner. There are generally two bottlenecks in modern scientific programs: memory and compute time.

Memory: If a job runs out of memory, the program will crash, so one needs to allocate sufficient memory. There is no simple, bulletproof strategy to estimate the amount of memory a program requires, so the best advice is to experiment. If you run out of memory, your program will throw an out-of-memory error like Python's MemoryError or MATLAB's out of memory error; otherwise, you should have enough memory for your program.

CPU-cores: Using multiple CPU cores simultaneously can alleviate problems with long run times if the code can use the cores. This is the case if certain parts of the code are explicitly parallelized. Many libraries and commercial software contain some degree of parallelization. So even without parallelizing the code yourself, there might be parallelism hidden in the libraries. To test this, try running your code in a job with one CPU core and one with two CPU cores. In both cases, time the runs and output the numbers. If the two CPU-core run is significantly faster, your code has some parallelism.

Like with memory, there is no simple answer to how many CPU cores a parallel code should use. Due to Amdahl's law, one will eventually experience that added more CPU-cores yield only diminishing improvements in run time. A good rule of thumb is thus to experiment to find out when this point of diminishing returns occurs for your code. To find this point, here are a few guidelines:

- *Human time is far more important than machine time.* Long runtime only truly becomes an issue when it impacts our productivity as users.
- *Convenience.* Use the system in a manner that is convenient to you. It should be a tool for better research, not a hassle.
- *When will you check the results?* If you do not plan to use the results before the following day, use only the number of cores required for the job to finish in the morning.
- *How many resources do you have left on the system?* Keep an eye on the tables in Fig. 1.4 and how fast you are using them. There is no reason to worry about resource use unless it is necessary.

2.3 Programming for HPC

Writing code for performance is difficult and time-consuming. There are many things to consider and only limited time to do so. The best advice the author can give is that in most cases, the results the

1: GPUs are typically also available on HPC systems, but that is beyond the scope of this tutorial.

code produces matter – not the code itself. Therefore only worry about performance when it limits your research and go for the easy solutions[2]. In general, HPC will give you the biggest boost if you can parallelize the parts of the code that occupy the majority of the run time. Many programming languages have tools to parallelize functions (Python, MATLAB) – use these.

2: Unless you are working on a long-term project designing code to be used by others for many years to come.