Setup

Learning outcomes

> **Note**
>
> You might already be able to do some or all of these learning outcomes. If so, you can go through the corresponding exercises quickly. The general aim of this chapter is to work comfortably on a remote server by using the command line.

After having completed this chapter you will be able to:

- Use the command line to:
  - Make a directory
  - Change file permissions to 'executable'
  - Run a `bash` script
  - Pipe data from and to a file or other executable
- Program a loop in `bash`

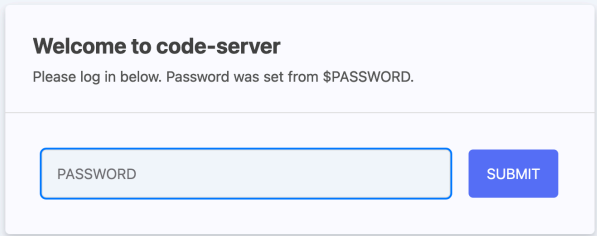
> **Choose your platform**
>
> In this part we will show you how to access the cloud server, or setup your computer to do the exercises with conda or with Docker.
>
> If you are doing the course **with a teacher**, you will have to login to the remote server. Therefore choose:
>
> - Cloud notebook
>
> If you are doing this course **independently** (i.e. without a teacher) choose either:
>
> - conda
> - Docker

Cloud notebook

Exercises

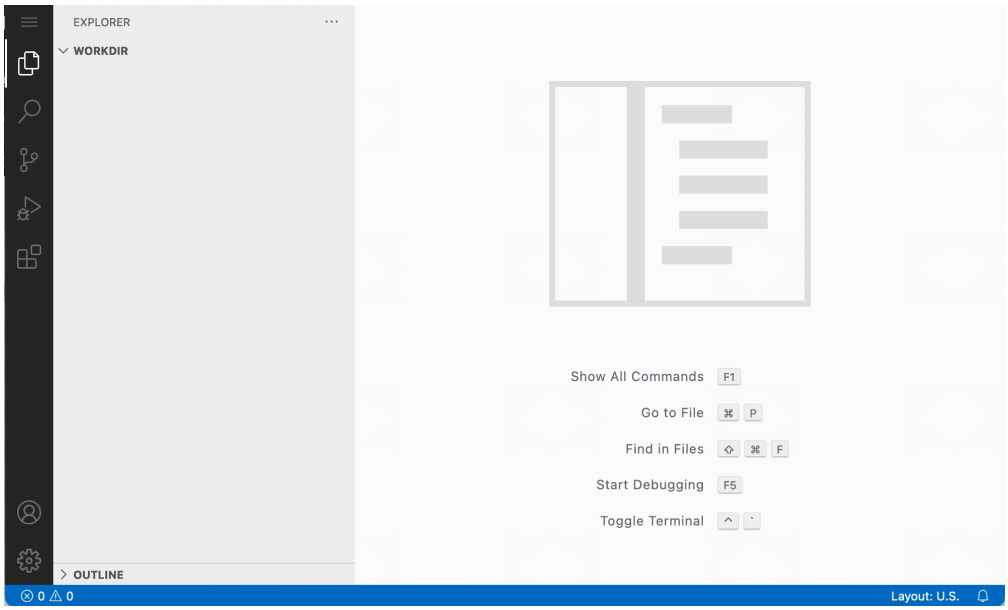
First login

If you are participating in this course with a teacher, you have received a link and a password. Copy-paste the link (including the port, e.g.: `http://12.345.678.91:10002`) in your browser. This should result in the following page:

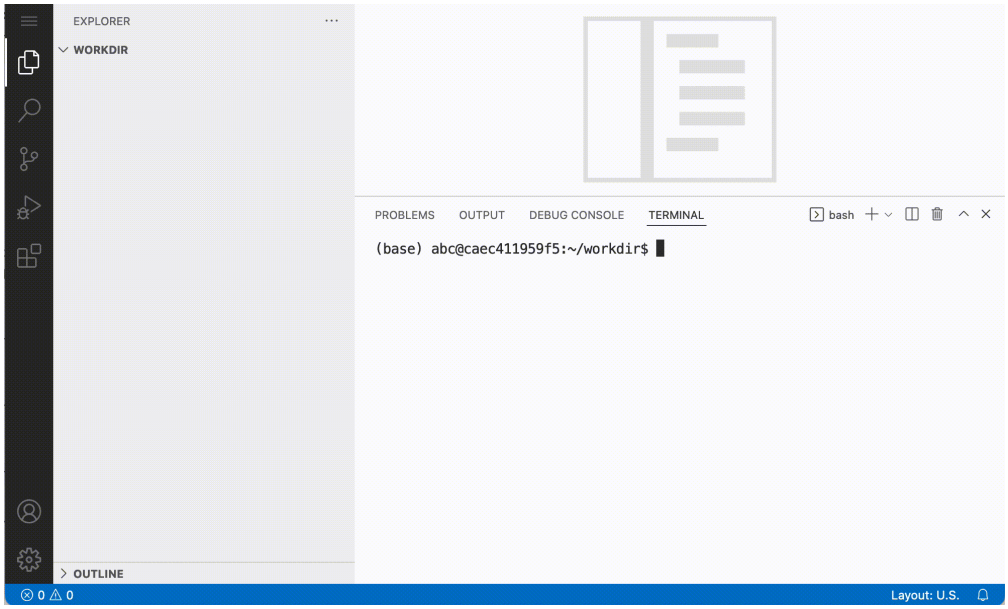


> **Info**
>
> The link gives you access to a web version of Visual Studio Code. This is a powerful code editor that you can also use as a local application on your computer.

Type in the password that was provided to you by the teacher. Now let's open the terminal. You can do that with `^ Ctrl` + `` ` ``. Or by clicking **Application menu** > **Terminal** > **New Terminal**:



For a.o. efficiency and reproducibility it makes sense to execute your commands from a script. With use of the 'new file' button:



Docker

Material

- Instructions to install docker
- Instructions to set up to container

Exercises

First login

Docker can be used to run an entire isolated environment in a container. This means that we can run the software with all its dependencies required for this course locally in your computer. Independent of your operating system.

In the video below there's a tutorial on how to set up a docker container for this course. Note that you will need administrator rights, and that if you are using Windows, you need the latest version of Windows 10.

The command to run the environment required for this course looks like this (in a terminal):

> **Modify the script**
>
> Modify the path after `-v` to the working directory on your computer before running it.

```
docker run \
--rm \
-p 8443:8443 \
-e PUID=1000 \
-e PGID=1000 \
-e DEFAULT_WORKSPACE=/config/project \
-v $PWD:/config/project \
geertvangeest/ngs-introduction-vscode:latest
```

If this command has run successfully, navigate in your browser to http://localhost:8443.

The option `-v` mounts a local directory in your computer to the directory `/config/project` in the docker container. In that way, you have files available both in the container and on your computer. Use this directory on your computer to e.g. visualise data with IGV. Change the first path to a path on your computer that you want to use as a working directory.

> **Don't mount directly in the home dir**
>
> Don't directly mount your local directory to the home directory (`/root`). This will lead to unexpected behaviour.

The part `geertvangeest/ngs-introduction-vscode:latest` is the image we are going to load into the container. The image contains all the information about software and dependencies needed for this course. When you run this command for the first time it will download the image. Once it's on your computer, it will start immediately.

conda

If you have a conda installation on your local computer, you can install the required software using conda. If not, you can install Miniconda like this:

Windows

- Get the `.exe` file here
- Double click the file
- Follow the instructions on the screen (defaults are usually fine)
- Run the command `conda list` in the Ananconda prompt or terminal to check whether your installation has succeeded.

Mac/Linux

- Get the installation (`.sh`) script here
- Run in your terminal:

```
bash Miniconda3-latest-Linux-x86_64.sh
```

- Follow the prompts
- Close and reopen your terminal for changes to have effect
- Run the command `conda list` in the Ananconda prompt or terminal to check whether your installation has succeeded.

After installation, you can install the required software:

Windows/MacOS

```
conda create -n ngs-tools

conda activate ngs-tools

conda install -y -c bioconda \
    samtools \
    bwa \
    fastqc \
    sra-tools \
```

```
    bowtie2=2.4.2 \
    hisat2=2.2.1 \
    subread=2.0.1 \
    entrez-direct \
    minimap2 \
    gatk4 \
    freebayes \
    multiqc \
    fastp
```

Linux

Download ngs-tools.yml, and generate the conda environment like this:

```
conda env create --name ngs-tools -f ngs-tools.yml
```

Note

If that did not succeed, follow the instructions for Windows/MacOS.

This will create the conda environment `ngs-tools`

Activate it like so:

```
conda activate ngs-tools
```

After successful installation and activating the environment all the software required to do the exercises should be available.

If you are doing project 2 (long reads)

If you are doing the project 2 as part of the course, you will need to install `NanoPlot` as well, using `pip`:

```
pip install NanoPlot
```

A UNIX command line interface (CLI) refresher

Most bioinformatics software are UNIX based and are executed through the CLI. When working with NGS data, it is therefore convenient to improve your knowledge on UNIX. For this course, we need basic understanding of UNIX CLI, so here are some exercises to refresh your memory.

If you need some reminders of the commands, here's a link to a UNIX command line cheat sheet:

UNIX cheat sheet

Make a new directory

Make a directory `scripts` within `~/project` and make it your current directory.

Answer

```
cd ~/project
mkdir scripts
cd scripts
```

File permissions

Generate an empty script in your newly made directory `~/project/scripts` like this:

```
touch new_script.sh
```

Add a command to this script that writes "SIB courses are great!" (or something you can better relate to.. 😉) to stdout, and try to run it.

Answer

Generate a script as described above. The script should look like this:

```
#!/usr/bin/env bash

echo "SIB courses are great!"
```

Usually, you can run it like this:

```
./new_script.sh
```

But there's an error:

```
bash: ./new_script.sh: Permission denied
```

Why is there an error?

Hint

Use `ls -lh new_script.sh` to check the permissions.

Answer

```
ls -lh new_script.sh
```

gives:

```
-rw-r--r--  1 user  group    51B Nov 11 16:21 new_script.sh
```

There's no `x` in the permissions string. You should change at least the permissions of the user.

Make the script executable for yourself, and run it.

Answer

Change permissions:

```
chmod u+x new_script.sh
```

`ls -lh new_script.sh` now gives:

```
-rwxr--r--  1 user  group    51B Nov 11 16:21 new_script.sh
```

So it should be executable:

```
./new_script.sh
```

More on `chmod` and file permissions here.

Redirection: `>` and `|`

In the root directory (go there like this: `cd /`) there are a range of system directories and files. Write the names of all directories and files to a file called `system_dirs.txt` in your working directory.

Answer

```
ls / > ~/project/system_dirs.txt
```

The command `wc -l` counts the number of lines, and can read from stdin. Make a one-liner with a pipe `|` symbol to find out how many system directories and files there are.

Answer

```
ls / | wc -l
```

Variables

Store `system_dirs.txt` as variable (like this: `VAR=variable`), and use `wc -l` on that variable to count the number of lines in the file.

Answer

```
FILE=~/project/system_dirs.txt
wc -l $FILE
```

shell scripts

Make a shell script that automatically counts the number of system directories and files.

Answer

Make a script called e.g. `current_system_dirs.sh`:

```
#!/usr/bin/env bash
cd /
ls | wc -l
```