1 Bioinformatics Essentials

1.1 Introduction

In this tutorial we will cover two important and essential concepts for bioinformatics analyses.

- Management and installation of bionformatics software applications
- Automating the process of data analysis with workflow managers

1.2 Learning outcomes

On completion of the tutorial, you can expect to be able to:

- Describe what a software package manager is and why it is useful

- Install conda and use it to manage bioinformatics software applications
- Describe what a workflow manager is and why it is useful
- Install Nextflow and run existing Nextflow pipelines to automate NGS data analysis

1.3 Tutorial sections

This tutorial comprises the following sections:
1. Conda
2. Nextflow
3. Nextflow Pipelines

1.4 Authors and License

This tutorial was written by Jacqui Keane.

1.5 Running the commands in this tutorial

To get started, open a terminal window and type the command below followed by the `Enter` key:

```
[ ]: su bioinf
```

When prompted for a password enter the text below followed by the `Enter` key:

```
[ ]: password
```

This will switch the Linux user from `manager` to `bioinf` to allow us to start from a empty non-configured set up without conda installed.

1.6 Prerequisites

This tutorial does not assume that you have any software installed on your computer. Instead it will take you through the process of installing `conda`, some bioinformatics software (`bwa`, `mafft` and `samtools`), `nextflow` and some nextflow pipelines for processing NGS data.

To get started with the tutorial, go to the first section: Conda

2 Conda

There are many bioinformatics software applications available for Linux and we have used many of these in this course (e.g. `samtools`, `bcftools`, `iqtree`, `roary` etc.). Many of these applications are complex and difficult to install. This is because often they dependend on and require other software applications and packages. This is complicated further when you have software applications that require different versions of the same software dependency. Consider `srst2` which has a dependency on python 2.7 and `ARIBA` which requires python 3.X or higher. How can you install both `srst2` and `ARIBA` on your computer and manage the conflicting python dependencies?

One solution is to use a software package manager to help manage software installations and their dependencies. A package manager is a tool that automates the process of installing, updating, and removing software packages on your computer. One of the most common package managers is `conda`. Here we will demonstrate how to install `conda`, create `conda` environments and install bioinformatics software with `conda`.

2.1 Installing conda

A software distribution is a collection of software packages that are pre-built and pre-configured for use on a specific computer system. There are two available distributions of conda:

- **miniconda**: a basic, lightweight installation (mostly just conda and Python)
- **anaconda**: a larger distribution with more pre-installed packages (includes conda, Python and 250+ automatically installed, open-source scientific packages and their dependencies)

Let's check if conda is installed:

[ ]:
```
conda --version
```

In previous tutorials conda was installed for the Linux user `manager`. As we have switched to a different user on the system (`bioinf`) you should see a message indicating that conda is not installed.

[ ]:
```
Command 'conda' not found.
```

Now let's install *miniconda*, a lightweight distribution of conda. Detailed instructions for installing conda on a Linux system can be found at https://docs.conda.io/projects/conda/en/latest/user-guide/install/linux.html.

In summary you need to run the following commands:

1. Set up a directory in your home directory for miniconda:

[ ]:
```
mkdir -p ~/miniconda3
```

2. Download the miniconda installation script and store in a file called `miniconda.sh` in the `~/miniconda3` directory:

[ ]:
```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh \\
-O ~/miniconda3/miniconda.sh
```

3. Run the installation script:

[ ]:
```
bash ~/miniconda3/miniconda.sh -b -p ~/miniconda3
```

Here `-b` and `-p` are options passed to the script `miniconda.sh`, `-b` means run the install in batch mode (without manual intervention) and `-p` specifies where on the computer to put the miniconda installation and in this case it will be stored in ~/mininconda3.

[ ]:
```
ls ~/miniconda3
```

4. Remove the installation script `~/miniconda3/miniconda.sh` as it is no longer required:

[ ]:
```
rm -rf ~/miniconda3/miniconda.sh
```

5. Initialize your newly-installed Miniconda:

[ ]:
```
~/miniconda3/bin/conda init bash
```

Now check if conda has installed:

[ ]:
```
conda --version
```

If installed correctly, you should also see (*base*) at the start of your prompt in your terminal.

Note: If you are installing conda on your own computer, the instructions may vary depending on what operating system you are using. Detailed installation instructions can be found at https://docs.conda.io/projects/conda/en/latest/user-guide/install/index.html

2.2 Conda channels

There are many software packages available via conda and they are made available to users via channels. Conda channels are the online locations where packages are stored and serve as the base for hosting and managing packages. Examples of conda channels are `default` and `conda-forge` and many bioinformatics software packages are available in the `bioconda` channel.

By default, conda will search the `default` channel for a software package.

[ ]:
```
conda install bwa
```

Conda fails to install `bwa` as it is not available in the `default` channel.

If you want to install a software package from a specific channel you can specify the channel with the `-c` option of the `conda install` command. For example, to install bwa from the `bioconda` channel:

[ ]:
```
conda install -c bioconda bwa
```

Alternatively you can configure and modify which channels are automatically searched by conda using the commands:

[ ]:
```
conda config --add channels default
conda config --add channels bioconda
conda config --add channels conda-forge
```

When you add a channel to your conda channel list it is automatically added to the top of the list. So with the commands above the order to be searched will be cond-forge, then bioconda and then default.

Try installing another bioinformatics software:

[ ]:
```
conda install mafft
```

More information about Bioconda can be found at https://bioconda.github.io/

2.3 Installing software

Some software packages will have dependencies in common where others will have dependencies that may conflict with each other. One approach to manage this is to install workflows or even individual software versions in their own environments.

You can create a conda environment with:

[ ]:
```
conda create -n samtools-1.17 samtools=1.17
```

This will create a conda environment (think of it as a box) and put `samtools` version 1.17 and all it's dependencies in the environment called `samtools-1.17`. You can name the environment anything but it is good practice to name it using a combination of the software name and version.

Once the environment is created you can access the software by activating the environment that contains the software:

[ ]:
```
conda activate samtools-1.17
```

The start of your terminal prompt will change from (base) to (samtools-1.17).

Check that `samtools` is installed:

[ ]:
```
samtools -h
```

To deactivate the environment and go back to the base environment:

[ ]:
```
conda deactivate
```

To see the list of available environments:

[ ]:
```
conda info --envs
```

To search for a particular environment:

[ ]:
```
conda info --envs | grep -i samtools
```

To list all the software packages installed in a specific environment:

[ ]:
```
conda list -n samtools-1.17
```

2.4 Keeping conda up to date

Conda and related software are often updated regularly so it is good to keep everything up to date. To update all software in the base conda environment you can use:

[ ]:
```
conda update --all
```

2.5 Cheatsheet

A list of useful conda commands can be found at https://docs.conda.io/projects/conda/en/stable/_downloads/843d9e0198f2a193a3484886fa28163c/conda-cheatsheet.pdf

2.6 Exercises

1. List all the software packages installed in the base environment
2. List all the software packages installed in the samtools-1.17 environment
3. Install the ARIBA software package in a new environment
4. What version of Python was installed with ARIBA? Is this different to the version installed in the base environment?
5. How many software packages were installed with ARIBA?
6. In the ARIBA environment, what channel was ARIBA installed from?
7. In the ARIBA environment, what channel was Python installed from?
8. How many conda environments have been created?

Now move on to the next part of the tutorial Nextflow

3 Nextflow

Nextflow is a workflow management system (consisting of a domain specific language and workflow engine) that can be used to write and run data-intensive bioinformatics workflows (i.e. pipelines).

One of the main advantages of Nextflow is that it provides an abstraction between the workflow's functional logic (what the workflow does) and the underlying execution system (how the workflow runs). Thus, it is possible to write a workflow that runs seamlessly on your computer, a cluster, or the cloud, without being modified. You simply define the target execution platform (local, lsf, aws etc.) in a configuration file.

With Nextflow you can write a pipeline once and run it everywhere!

Let's change back to the user `manager`. Type the commands below in the terminal window:

```
[ ]: su manager
```

When prompted for a password enter the text below followed by the `Enter` key:

```
[ ]: manager
```

3.1 Installing nextflow

We will use `conda` to install `nextflow`. Let's create a `conda` environment called *nextflow* and put `nextflow` version 23.04.1 and all it's dependencies in that environment.

```
[ ]: conda create -n nextflow nextflow=23.04.1
```

Activate the *nextflow* environment:

```
[ ]: conda activate nextflow
```

3.2 Nextflow and Containers

Bioinformatics workflows are rarely composed of a single script or software tool. More often, they depend on many software applications and packages. As noted in the previous section, installing and maintaining such dependencies is a challenging task.

To address this challenge, Nextflow uses containers to manage software dependencies. Containers are similar to a virtual machine in that they have their own copy of the file system, processing space, memory management, and software installations etc. They can be run on any computer that supports containers in such a way that they are isolated from the host machine.

Nextflow requires software applications used in a workflow to be encapsulated in one or more self-contained, ready-to-run containers. Nextflow supports both Docker and Singularity containers. We will demonstrate how to use Singularity with Nextflow as the usage of Docker is generally not allowed on compute clusters due to security constraints.

Therefore if using Nextflow you will also need Singulaity installed. To check that it is installed type:

```
[ ]: singularity -h
```

Installing `singularity` may require root or admin privileges so the instructions will vary depending on your operating system. It is already installed on the computer you are using and here are the commands that were used to install it (do not run these commands here).

```
wget https://github.com/sylabs/singularity/releases/download/v3.10.2/singularity-ce_3.10.2-jammy_amd64.deb
sudo dpkg -i singularity-ce_3.10.2-jammy_amd64.deb
rm singularity-ce_3.10.2-jammy_amd64.deb
```

3.3 The nf-core project

The nf-core project is a community effort to collect a curated set of best-practice analysis pipelines built using Nextflow. It provides Nextflow implementations of modules/subworkflows for common bioinformatics analysis tasks (e.g. running bwa) and pipelines for common bioinformatics workflows (e.g. map and snp call a set of bacterial isolates) along with a set of guidelines that these implementations must adhere to.

You can benefit from nf-core by using existing well-established and tested pipelines rather than having to implement your own Nextflow pipeline. The pipelines offered by nf-core are standardized, portable, well documented and user-friendly and guarantee reproducibility of results. You can also become a developer and write your own pipelines in Nextflow using ready-made modules available in nf-core. However this is out of the scope of this tutorial.

Now install `nf-core` with:

[ ]:
```
conda install nf-core
```

Once installed, you can check that everything is working by printing the help:

[ ]:
```
nf-core --help
```

As you can see from the `--help` output, the `nf-core` has a range of sub-commands. The simplest is `nf-core list`, which lists all available nf-core pipelines. The output shows the latest version number, when that was released. If the pipeline has been installed locally using Nextflow, it tells you when that was and whether you have the latest version.

[ ]:
```
nf-core list
```

To browse the available nf-core pipelines online visit https://nf-co.re/pipelines

3.4 Exercises

1. Use the nf-core help flag to print the list command usage
2. List all available nf-core pipelines
3. How many nf-core pipelines are there?
4. Sort the pipelines by popularity (stars)
5. Filter pipelines for those that work with RNA

To download and run your first nf-core pipeline, continue to the next section section: Running Nextflow Pipelines

4 Nextflow Pipelines

In this section we will download the `nf-core/bactmap` pipeline. We will apply this `nf-core/bactmap` pipeline to the data from the SNP Phylogeny tutorial and demonstrate how to use Nextflow Tower to track the progress of the pipeline run.

But first, let's check that you're in the right place. Type the commands below in the terminal window:

```
[ ]: cd ~/course_data/bioinformatics_essentials/data
pwd
```

It should display something like:

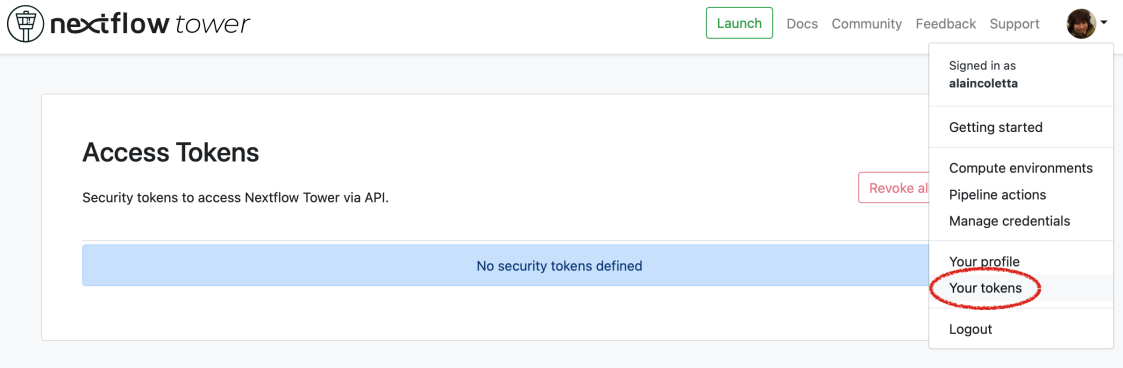
`/home/manager/course_data/bioinformatics_essentials/data`

4.1 Getting started with Nextflow Tower

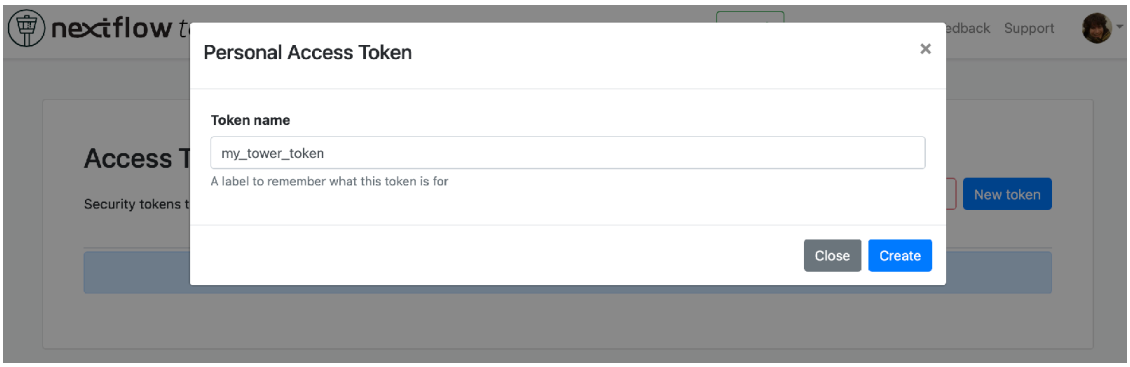
Nextflow Tower is the centralized command post for managing and tracking Nextflow workflows. It has many features including the functionality to monitor the progress of your Nextflow pipeline runs via a web interface.

Don't worry if this does not make sense initially, all will become clear once you run your first nextflow pipeline! For now follow the instructions below to set up and configure Nextflow Tower.

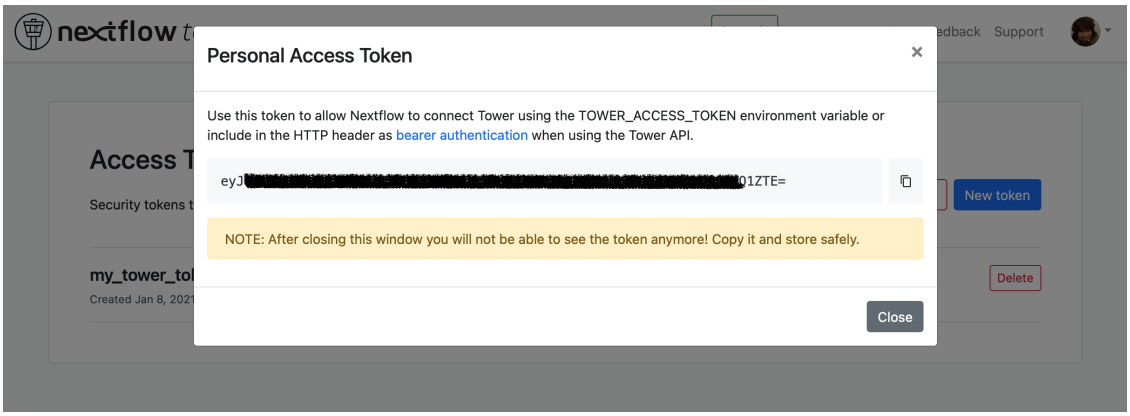
1. Go to `https://cloud.tower.nf/` and create an account and login into Tower.
2. Create a new token by selecting `Your tokens` from the Settings drop-down menu:



3. Name your token, this can be anything you like.



4. Copy your new token into a text editor and put in a safe place.



5. Once your token has been created, open a terminal and type:

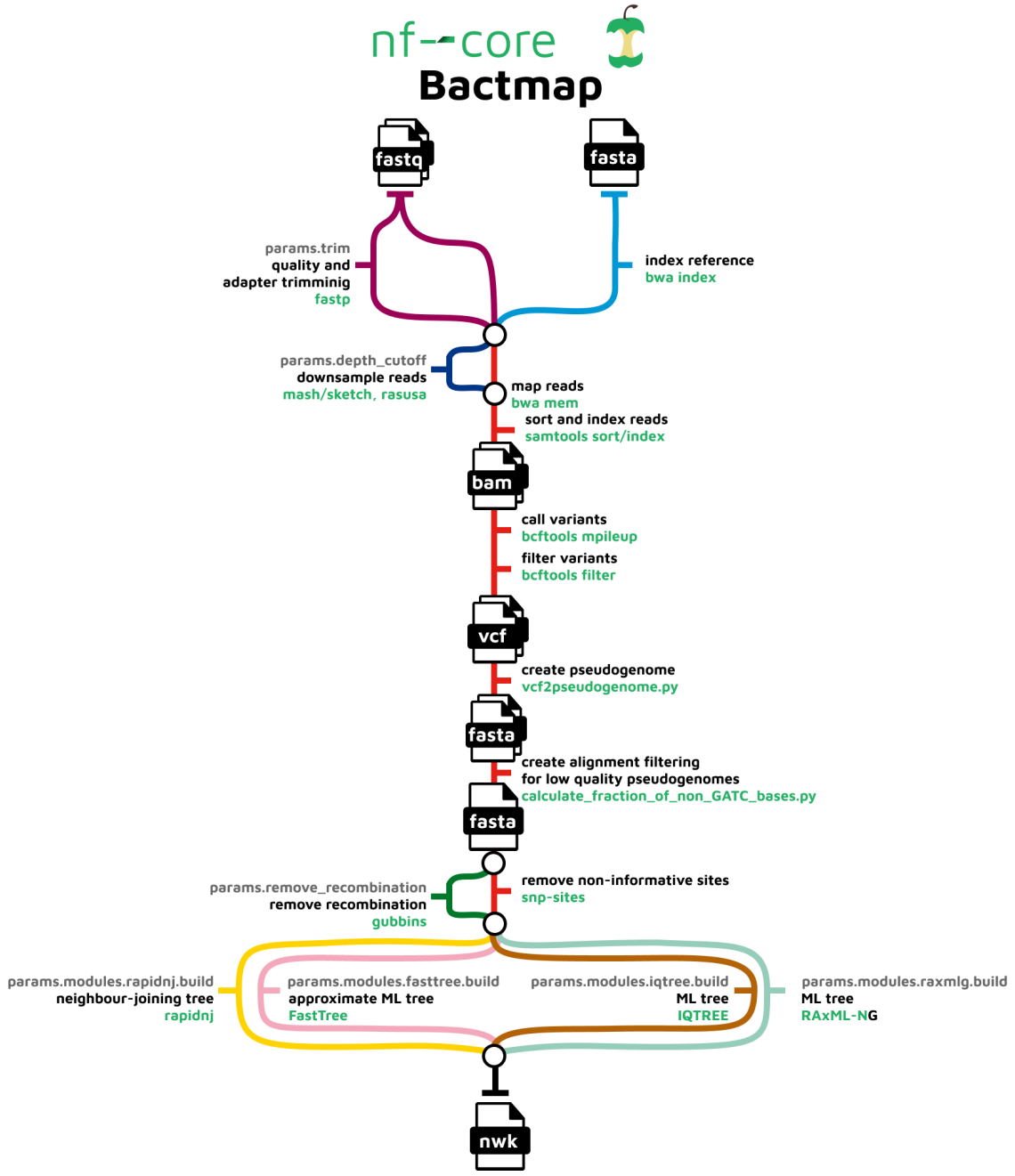
```
[ ]: export TOWER_ACCESS_TOKEN=eyxxxxxxxxxxxxxxxQ1ZTE
```

Where eyxxxxxxxxxxxxxxxQ1ZTE= is the token you have just created.

Just a note to say that if you want this TOWER_ACCESS_TOKEN to be set permanently then you can edit a file called `.bashrc` in your home directory and add the above line to the end of the file.

4.2 Installing Nextflow Pipelines

The `nf-core/bactmap` pipeline is a bioinformatics best-practice analysis pipeline for mapping short reads from bacterial WGS to a reference sequence, creating filtered VCF files, making pseudogenomes based on high quality positions in the VCF files and optionally creating a phylogeny from an alignment of the pseudogenomes.



Before we install any nf-core pipelines we must first set up a directory to hold the singularity containers for the software applications that used in the pipelines.

[ ]:
```
mkdir ~/singularity
export NXF_SINGULARITY_CACHEDIR="~/singularity"
```

To install this `nf-core/bactmap` pipeline, type

[ ]:
```
nf-core download --compress none --container-system singularity \\
--revision 1.0.0 bactmap
```

This may take some time to install. This command will download the nf-core pipeline called `bactmap`, the options `--compress none` indicate how to compress the output, `--container singularity` specifies to download singularity container images for the required software and `--revision 1.0.0` specifies which version of bactmap to install.

Check that the pipeline has installed:

[ ]:
```
ls nf-core-bactmap_1.0.0/1_0_0/
```

4.3 Running Nextflow Pipelines

Now that we have installed our first nextflow pipeline, let's run it on some data. First we need to consult the pipeline documentation to see how to run it. This can be found at https://nf-co.re/bactmap/1.0.0/docs/usage/. In summary, we need to create a samplesheet file with information about the samples in our experiment. This file has to be a comma-separated file with 3 columns, and a header row as shown in the example below.

```
sample,fastq_1,fastq_2
G18582004,fastqs/G18582004_1.fastq.gz,fastqs/G18582004_2.fastq.gz
G18756254,fastqs/G18756254_1.fastq.gz,fastqs/G18756254_2.fastq.gz
G18582006,fastqs/G18582006_1.fastq.gz,fastqs/G18582006_2.fastq.gz
```

Let's run the bactmap pipeline on our samples from the snp-phylogeny tutorial. To get you started we have already created a samplesheet file for this data. Take a look at it:

[ ]:
```
cat samplesheet.csv
```

Now run the `nf-core/bactmap` pipeline with

[ ]:
```
nextflow run nf-core-bactmap_1.0.0/1_0_0/main.nf \
--input samplesheet.csv \
--outdir my_results \
--reference chromosome.fasta \
--iqtree \
-w my_results/work \
-profile singularity \
-resume \
-with-tower
```

The pipeline steps are written in the nextflow language and are found in the file `nf-core-bactmap_1.0.0/1_0_0/main.nf`. Let's look at the options, these can be seperated into two types.

The first set of options (starting with `--`) are specific to the `nf-core/bactmap` pipeline:

- `--input` specifies the samplesheet.csv file that contains the fastq file information for our samples
- `--output` specifies the output where the results from the pipeline will be saved
- `--reference` specifies a fasta file of the reference sequence

- `--iqtree` specifies to build a tree using the IQ-TREE

The full set of parameters for the bactmap pipeline can be found at https://nf-co.re/bactmap/1.0.0/parameters/.

The second set of options (starting with -) are specific to the nextflow engine:

- `-w` option tells nextflow where to store the intermediate files produced by the pipeline
- `-profile` option tells nextflow to use the singularity profile
- `-resume` option tells nextflow to run the script using the cached results
- `-with-tower` option tells nextflow to send information about the pipeline run to Nextflow Tower using the TOWER_ACCESS_TOKEN set earlier

4.3.1 Nextflow profiles

A profile is a set of configuration attributes that can be activated/chosen when running a pipeline. One use of a profile is to define a set of attributes specific to where you are running the pipeline e.g. singularity here tells Nextflow to run the pipeline locally using Singularity containers.

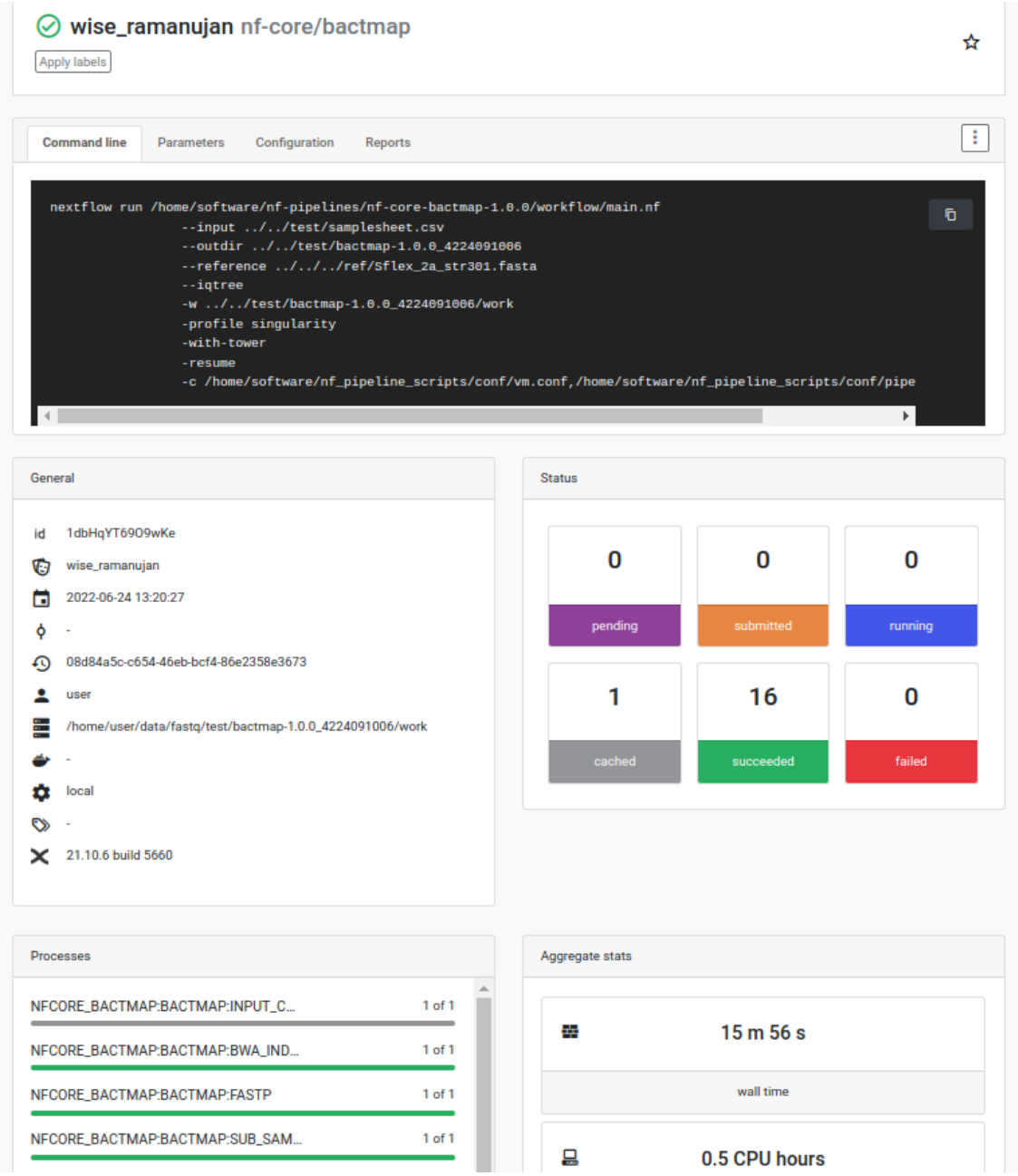
There are a range of pre-configured profiles to use with nf-core pipelines. If you were running the pipeline on a commercial cloud platform (e.g. google or amazon) you could use the specific profile for that platform (e.g. google or aws). If you were running the pipeline on a compute cluster at your institute (e.g. Sanger or Cambridge University) you could use an institute specific profile provided someone has implemented one in nf-core (e.g. sanger or cambridge). A list of available institute profiles can be found at https://github.com/nf-core/configs/

4.3.2 Nextflow resume

A very useful feature of Nextflow is the `--resume` option that can be used to re-start a pipeline if it fails. This will cache any steps of the pipeline that run succesfully. When you fix the problem with your data and re-run the pipeline it will not run all of the steps from the beginning but instead it will start from the last failed step.

4.3.3 Nextflow tower

You should be able to monitor your Nextflow jobs in Nextflow Tower. So go back to the Nextflow Tower website and see if you can find your pipeline run on the website. Nextflow Tower is extremely powerful, if you have time explore the interface for your pipeline run. Notice it lists all the steps of a pipeline run, the commands involved in each step and tracks the compute resources (cpu time and memory) for the pipeline.



4.4 Nextflow Tips

- Always take note of where you launced a nextflow pipeline from, pipeline progress will be stored in a hidden file called `.nextflow` in that directory
- If running more than one nextflow pipeline at a time make sure to launch or start them from different directories
- Remember to delete the work directory if the pipeline was successful

Congratulations you have just successfully installed and run a nextflow pipeline!