Conda

conda info	Verify Conda is installed, check version number.
conda update conda	Update conda package and environment manager to current version.
conda list	List linked packages in a conda environment.
conda create --name ENVNAME python=3.6 "PKG1>7.6" PKG2	Create a new environment named ENVNAME with specific version of Python and packages installed.
conda activate ENVNAME	Activate the conda environment named ENVNAME.
conda list --name ENVNAME	List all packages and versions in the environment named ENVNAME.
conda install PKGNAME==3.1.4	Install a package by exact version number (3.1.4).
conda uninstall PKGNAME --name ENVNAME	Remove a package from an environment named ENVNAME.
conda env export --name ENVNAME > envname.yml	Export an environment named ENVNAME to a YAML file.
conda env create --file envname.yml	Create an environment from a YAML file.
conda clean	Remove unused packages and caches.
conda deactivate	Deactivate the current active conda environment.

DOCKER

docker search <myApp>	Search for an image of <myApp> in dockerhub
docker pull <imageDocker:1.0>	Install the image “imageDocker” with version “1.0”
`docker images`	List all images available
docker inspect <imageDocker:1.0>	Obtain a description of the image in JSON format
docker run <imageDocker:1.0> <myCommand>	Run the command of the application
docker run -v <localPath:imagePath> <imageDocker:1.0> <myCommand>	Map a local volume to a volume in the image while running a command
docker build -t <myDockerfile:1.0> <path/to/dockerfile>	Create an image named “myDockerfile:1.0” from a file named “dockerfile”
docker ps -l	List the containers (-l for the last container)
docker start <myContainer>	Restart a stopped container (it will be detached)
...	

DOCKER

...	
docker stop myContainer	Stop the container “myContainer”
docker exec myContainer myCommand	Run command in a running container
docker commit myContainer myImage:1.0	Create an image from a container
docker save myImage:1.0 > path/savedImage.tar	Save an image into a file
docker load < path/savedImage.tar	Load an image from a saved image file
docker rm containerid	Removing one local container
docker rmi myImage:1.0	Removing one local image
docker system prune	Clean stopped containers, residual images, cache...

Snakemake

`docker pull snakemake/snakemake` `conda create -n smk-env -c bioconda snakemake`	Downloads and installs snakemake respectively within docker or conda (using bioconda)
`docker run -v ${PWD}:/data -w /data snakemake/snakemake` `conda activate smk-env` `module load snakemake`	Activates snakemake environment within docker/conda/an HPC cluster respectively
`snakemake --cores 1 --snakefile ex1_o1.smk --use-conda/envmodule/singularity`	Runs the pipeline described in ex1_o1.smk on 1 core within a conda/SLURM module/singularity environment
`snakemake -c1 -s ex1_o1.smk --configfile myConfig.yml`	Runs the pipeline described in ex1_o1.smk on 1 core using a YAML config file containing parameters and paths to files
`snakemake --cores all --snakefile ex1_o1.smk`	Runs the pipeline described in ex1_o1.smk on all cores available
`snakemake --cores 8 --snakefile ex1_o1.smk -R fastqc`	Redoes rule/step fastqc within the pipeline described in ex1_o1.smk on 8 cores

.smk files describe the pipeline with “rules”: named descriptions of each step, including names of input and output files, and command lines to be launched. The first rule must describe the end files as input and the others can be written in any order provided the output files are exactly the same as the input files of the next rule/step in the pipeline. Rules will be recursively examined until the latest files are found, and the pipeline will be redone from there.

```
rule fastqc:
  output:
    "FastQC/SRR3099585_chr18_fastqc.zip",
    "FastQC/SRR3099585_chr18_fastqc.html"
  input:
    "Data/SRR3099585_chr18.fastq.gz"
  shell: "fastqc --outdir FastQC/ {input}"
```

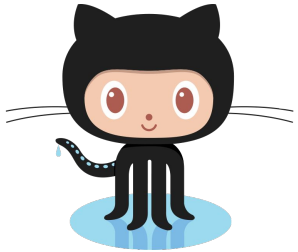
Rmarkdown Notebook

Rmarkdown documentation File: my_Rmarkdown_Notebook.Rmd	
Header: yml format	--- key:value ---
Text: explanation (markdown formatted)	# title ## subtitle
Chunk: place for code (specify the language between {}) Add {r, eval=FALSE} to present code and not the result of its execution	```{r} i <- i+1 ```

Jupyter Notebook

Jupyter documentation File: my_Jupyter_Notebook.ipynb	Nbviewer : static renderer for notebooks Binder : Jupyter + Docker
Header: ??	
Text:	
Cells: place for code	

GitHub Cheat sheet



Pages

- Choose/create a folder for the content of the website (can be the root of the repository)
- Go to GitHub Settings -> Options -> paragraph GitHub Pages
- Choose the source (branch / folder) and save
- Choose the theme and select
- Create HTML/MD files for the content in the repository

Fork Pull request

- Fork the repository
- Create a branch in the forked repository
- Clone/edit the forked repository
- Push changes to the forked repository
- Pull request to the initial repository

Release

- “Create a new release”
- Choose a tag, i.e. vMAJOR.MINOR.PATCH
- Choose a branch
- Provide a title and description
- (optional) attach binaries
- Click on Publish the release

Issues

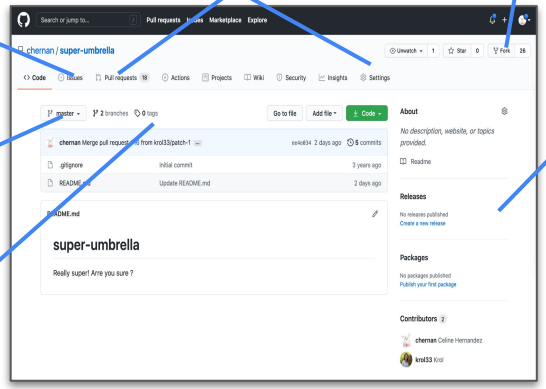
To keep track of problems/bugs
- Created by users
- Created by yourself to communicate with users

Branch

- New branch : provide a new name
- Switch branch : select an existing branch

Tag

Add a label to the repository in a specific status at a given time



Zenodo

- Choose a Licence on GitHub
- Create a Zenodo account with GitHub identifiers
- Zenodo Settings -> GitHub tab
- Flip the switch for the repository to connect (adds a webhook on GitHub)
- Integrate the badge on GitHub (README)
- Any new release on GitHub will generate a DOI on Zenodo