

# Combining Tools for Reproducible Research with Snakemake

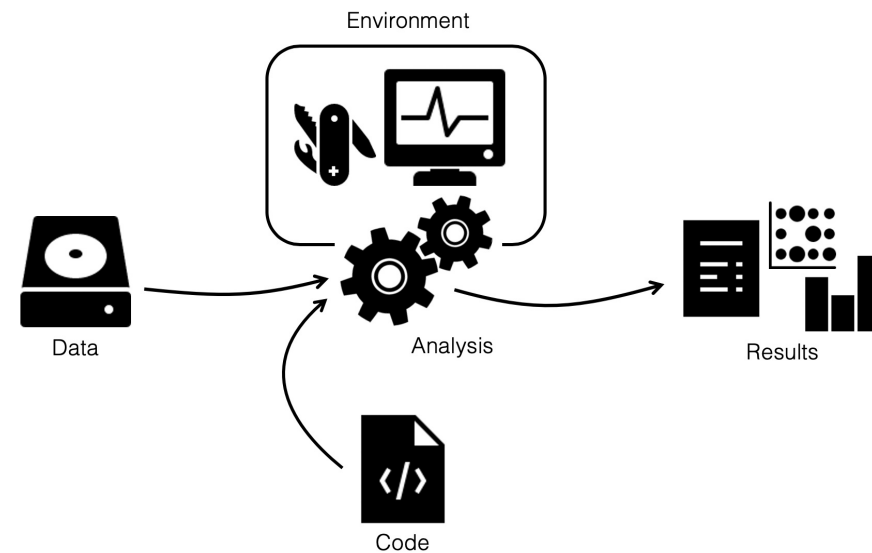
# Reproducibility is rarer than you think

The results of only 26% out of 204 randomly selected papers in the journal Science could be reproduced.<sup>1</sup>

<sup>1</sup> Stodden et. al (2018). "An empirical analysis of journal policy effectiveness for computational reproducibility". PNAS. 115 (11): 2584-2589

Many journals are revising author guidelines to include data and code availability.  
(...) an improvement over no policy, but currently insufficient for reproducibility.

# Combining Tools for Reproducible Research with Snakemake

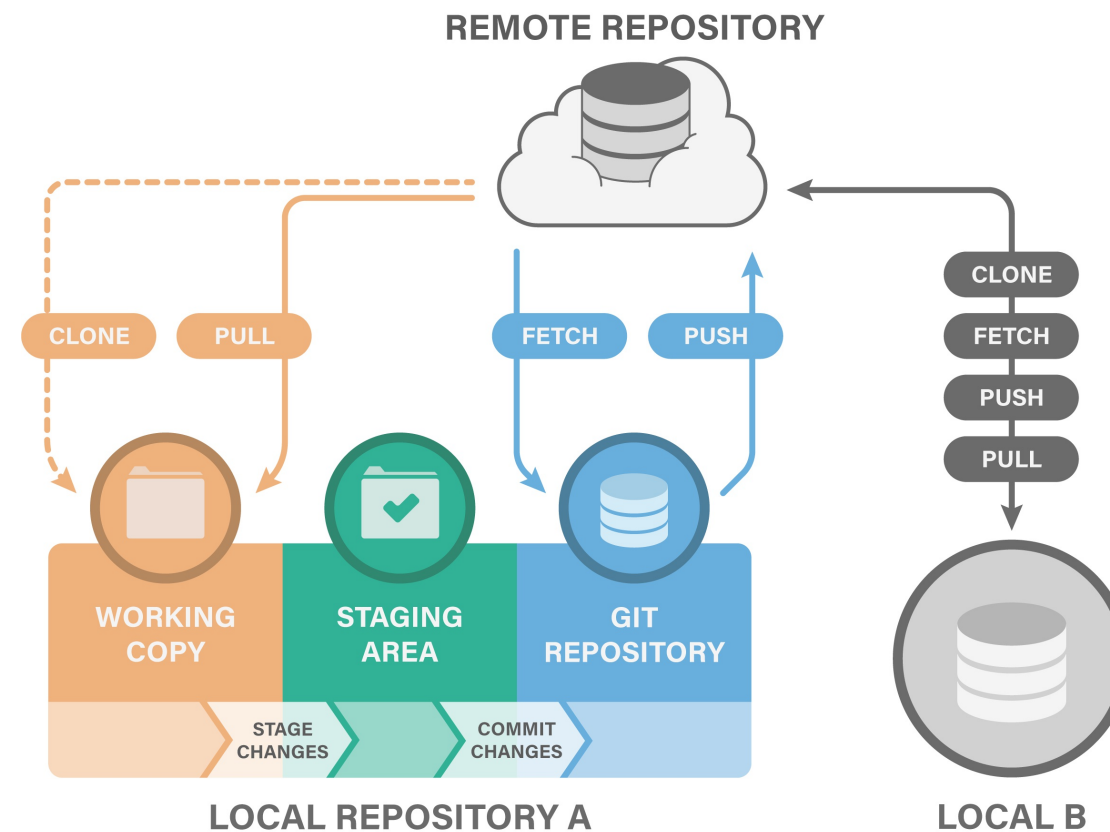


- Track your Snakemake code with [Git](#) and push it to a remote [repository](#) on GitHub or BitBucket to ensure that the different code versions are tracked and available
- Combine Snakemake with [Conda](#) and/or [containers](#) to make the compute environment and the code reproducible
- Integrate foreign workflow management systems such as [Nextflow](#) pipelines into your Snakemake workflow

# Git

- A widely used system for distributed version control to **version, backup and share** your code and documents
- Keeps a complete **history** of the changes you make to your files that can be re-visited & compared
- Git tracks **who contributed what** to your code
- Git is mainly used for **text files**, not large or binary files, so ideal for Snakemake workflows
- You can publish your code along with your research paper by sharing it in a remote repository (e.g. on **GitHub** or **BitBucket**)

# Git



1. Do some **coding** (i.e. add or change contents of files)
2. **Stage** the changes (i.e. specify which changes should be stored)
3. **Commit** the changes (storing them in the repository's history)
4. **Push** and **pull** regularly to/from your remote repository (on GitHub or Bitbucket) to collaborate, backup and share your code

# Conda

- Is a **package, dependency, and environment** manager

packages: any type of program (e.g. bowtie2, snakemake etc.)

dependency: other software required by a package

environment: a distinct collection of packages

- Keeps track of the dependencies between packages in each environment

# Conda

## Running a Snakemake rule with a Conda environment

- Make sure you have Conda **installed** (Miniconda or Anaconda)
- Find your Conda **package** on <http://anaconda.org>
- Create a Conda **environment file** (e.g. `bwa.yaml` )

```
channels:  
- conda-forge  
- bioconda  
- defaults  
dependencies:  
- bwa=0.7.17
```

(from best practice snakemake workflow: [https://github.com/NBISweden/snakemake\\_best\\_practice](https://github.com/NBISweden/snakemake_best_practice))

- Store your `yaml` files in a directory for environments
- For reproducibility, it is important to keep include package **versions** in your environment file
- **Git** is ideal to track changes in your Conda environment files

# Conda

## Running a Snakemake rule with a Conda environment

- Add the **path** to the Conda environment **yaml** file to your rule using the **conda** directive

```
rule map_bwa_index:  
  output: expand("{{ref}}{ext}", ext=[".amb", ".ann", ".bwt", ".pac", ".sa"])  
  input: config["ref"]  
  log: "logs/bwa/index/{ref}.log"  
  conda: "../envs/bwa.yaml"  
  shell:  
    "bwa index {input}"
```

(modified from best practice snakemake workflow: [https://github.com/NBISweden/snakemake\\_best\\_practice](https://github.com/NBISweden/snakemake_best_practice))

- Start your workflow on the command line with **--use-conda**

```
$ snakemake --use-conda
```

- This doesn't work if you use **run** instead of **shell** (or other directives)



# Conda

## Using a Conda environment for the entire workflow

- Write a Conda **environment file** that includes all tools used by the workflow, or those used by rules with `run` directives (save it as e.g. `environment.yaml` )

```
name: best-practice-smk
channels:
- conda-forge
- bioconda
- default
dependencies:
- snakemake=6.8.0
- python=3.8
- pandas=1.3.3
- jupyter=1.0
- jupyter_contrib_nbextensions=0.5.1
- jupyterlab_code_formatter=1.4
- bwa=0.7.17
- multiqc=1.11
- r-ggplot2=3.3.5
- samtools=1.13
```

(from best practice snakemake workflow: [https://github.com/NBISweden/snakemake\\_best\\_practice](https://github.com/NBISweden/snakemake_best_practice))

# Conda

## Using a Conda environment for the entire workflow

- **Create** the environment

```
$ conda env create -f environment.yml
```

- Use a terminal multiplexer to run the workflow in a shell instance in the background, e.g. **tmux** or **screen**
- **Activate** your Conda environment in the tmux or screen session

```
$ conda activate best-practice-smk
```

- Start your Snakemake workflow

```
(best-practice-smk) [...] $ snakemake
```

# Docker & Singularity

What can I use Docker for?

- Run applications securely **isolated** in a container, packaged with **all dependencies and libraries**
- As advanced **environment manager**
- To package your **code** with the environment it needs
- To package a whole **workflow** (e.g. to accompany a manuscript)
- And much more

Singularity

- Is an open source container platform suitable for **HPC clusters**

# Docker & Singularity

## Docker nomenclature

- A Docker **file** is a recipe used to build a Docker **image**
- A Docker **image** is a standalone executable package of software
- A Docker **container** is a standard unit of software run on the Docker Engine
- **DockerHub** is an online service for sharing docker images
- Docker images can be converted into Singularity images

# Docker & Singularity

## Running Snakemake rules with Singularity

- Snakemake can run a rule **isolated** in a container, using Singularity
- Each Conda package is also available as Docker and Singularity images (e.g. check <http://biocontainers.pro> for Conda packages from the bioconda channel)
- Many other Docker images are also available on DockerHub (<https://hub.docker.com/>)
  - But be aware that Docker images in free accounts are automatically deleted after a certain time of inactivity

# Docker & Singularity

## Running Snakemake rules with Singularity

- Make sure your system has Singularity **installed**
- Find your Docker or Singularity **image**, e.g. on <http://biocontainers.pro>
- Add the **link** to the container image (or the path to a Singularity `*.sif` file) to your rule using the `container` directive

```
rule NAME:
  input:
    "table.txt"
  output:
    "plots/myplot.pdf"
  container:
    "docker://joseespinoza/docker-r-ggplot2"
  script:
    "scripts/plot-stuff.R"
```

(example from Snakemake documentation)

- Start your workflow on the command line with `--use-singularity`

```
$ snakemake --use-singularity
```

# Docker & Singularity

## Packaging your Snakemake workflow in a Docker container

- Write a **Docker file** (`my_workflow`), e.g.
  - Start with the official Miniconda `base` image
  - Install the core packages of the workflow (e.g. Snakemake and dependencies such as pandas)
  - Include all rule-specific environments as separate Conda files (running your rules with Conda)
  - Include your workflow with `COPY <local-src> <container-destination>` into the Docker file
  - Include the required input data, e.g.
    - Mount the path with the data inside the container
    - Mount a sample list, specifying their data paths

# Docker & Singularity

## Packaging your Snakemake workflow in a Docker container

- Create a Docker **image** from your Docker file (e.g. called `my_workflow` )

```
$ docker build -t my_workflow .
```

- **Run** your container

```
$ docker run --name my_first_workflow_instance -it my_workflow
```

- **Share** your Docker file on GitHub or BitBucket, or your Docker image on DockerHub



# Docker & Singularity

## Containerization of Conda based workflows

- Snakemake can **automatically** generate a Docker container image that contains all Conda environments specified with the `conda` directive in the rules of the workflow
- Generate a **Docker file** with `--containerize`

```
snakemake --containerize > Dockerfile
```

- The Docker **image** from this Docker file can then be used via the directive `containerized` (globally or per rule):

```
containerized: "docker://username/myworkflow:1.0.0"

rule NAME:
  input:
    "table.txt"
  output:
    "plots/myplot.pdf"
  conda:
    "envs/ggplot.yaml"
  script:
    "scripts/plot-stuff.R"
```

(example from Snakemake documentation)

# Integrating foreign workflow management systems

- From version 6.2 on, Snakemake can run workflows written in other workflow management systems such as **Nextflow**
- The workflow is run in Snakemake until a **rule** to run the foreign workflow is reached
- In this rule, Snakemake **hands over** to the other workflow management system indicated by the directive `handover`

```
rule chipseq_pipeline:
  input:
    input="design.csv",
    fasta="data/genome.fasta",
    gtf="data/genome.gtf",
  output:
    "multiqc/broadPeaks/multiqc_report.html",
  params:
    pipeline="nf-core/chipseq",
    revision="1.2.1",
    profile=["conda"],
  handover: True
  wrapper:
    "0.74.0/utils/nextflow"
```

(example from Snakemake documentation)

# Integrating foreign workflow management systems

- The handover rule is run **locally**
- **Job submissions** to the cluster or cloud system are handled by the other workflow management system
- Afterwards, **Snakemake** continues running rules that use the output files of the foreign workflow

# Summary

There are many ways to use other **tools for reproducible research** together with Snakemake:

- Use **Git** to version control, backup and share your code
- Run rules or your entire workflow in **Conda** environments
- Run your rules in isolated Docker or Singularity **containers** using Singularity
- Package your entire workflow in a **Docker container**
- Run pipelines written in **other workflow management systems** in your Snakemake workflow

Questions?

