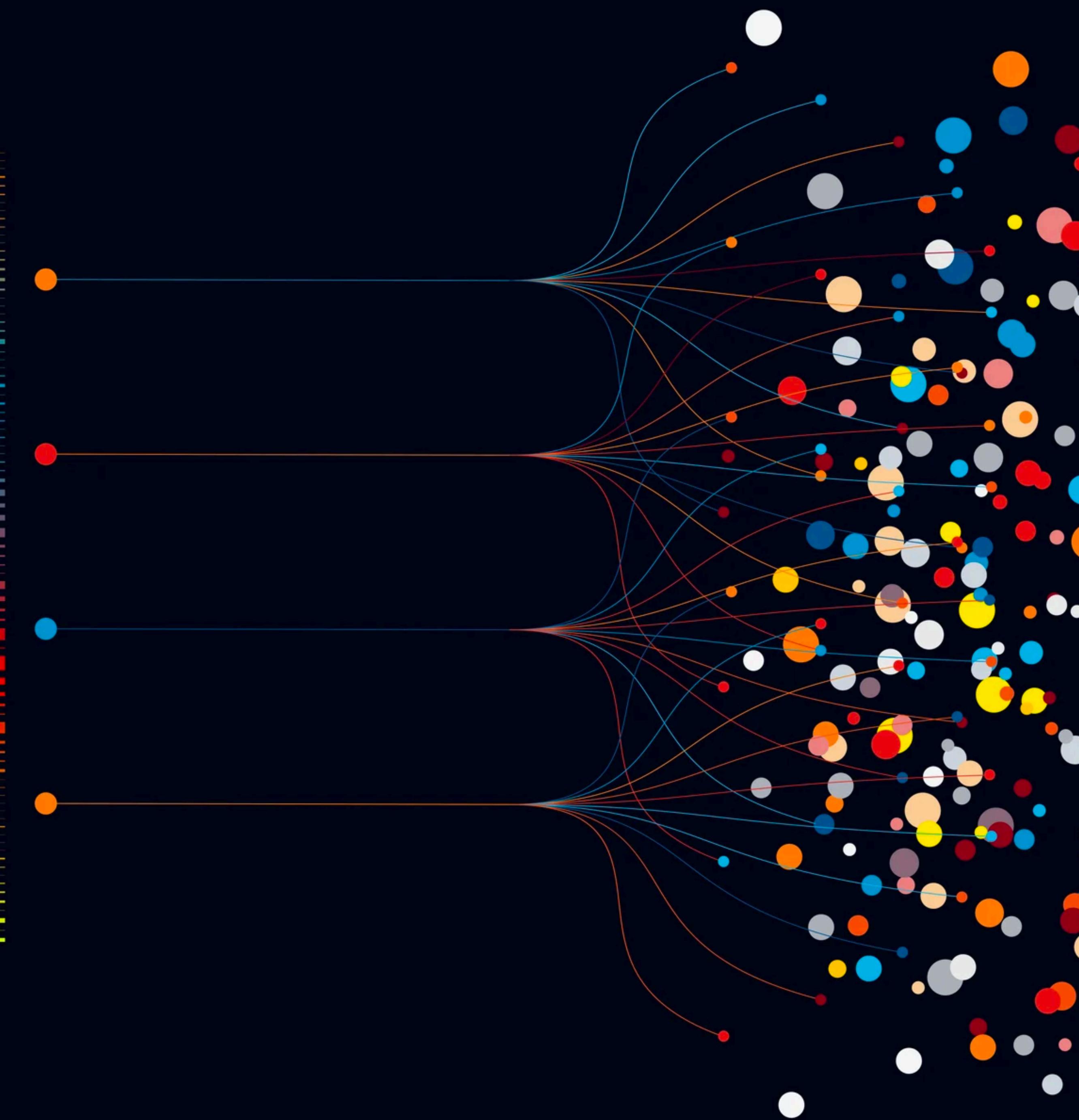


Module 4 - day 2

Reproducible data analysis and workflows

Antonio Fernandez-Guerra
antonio@metagenomics.eu

Fundamentals Data Analysis | March 7 2025 - Copenhagen



1. Understanding Workflow Management Systems (WfMS)

What they are and why they matter

2. Workflow Management Systems in Bioinformatics

A gentle introduction

4. Environments and containers

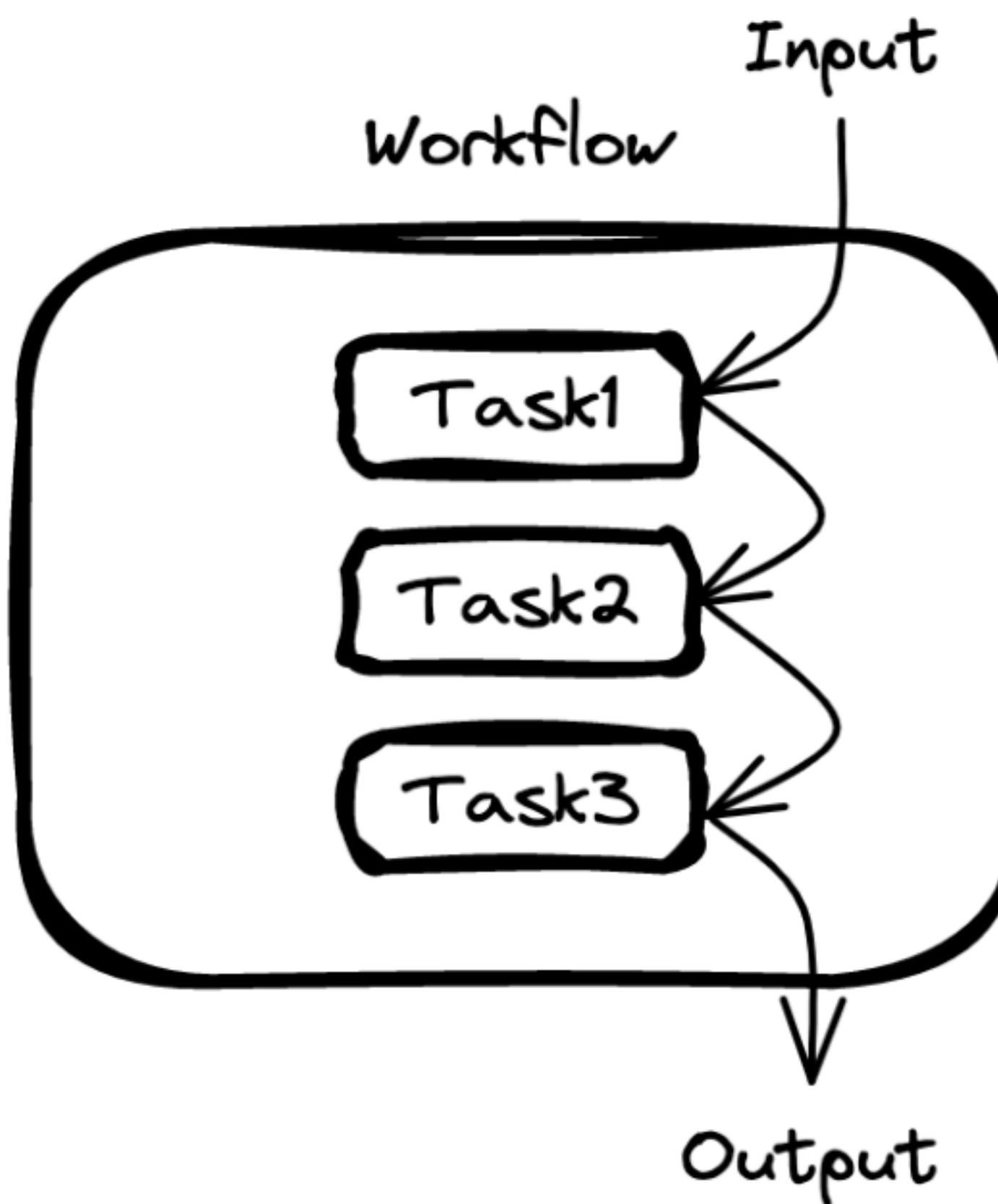
Creating and managing reproducible computational environments

5. An Introduction to Snakemake

A Python-based WfMS for reproducible research

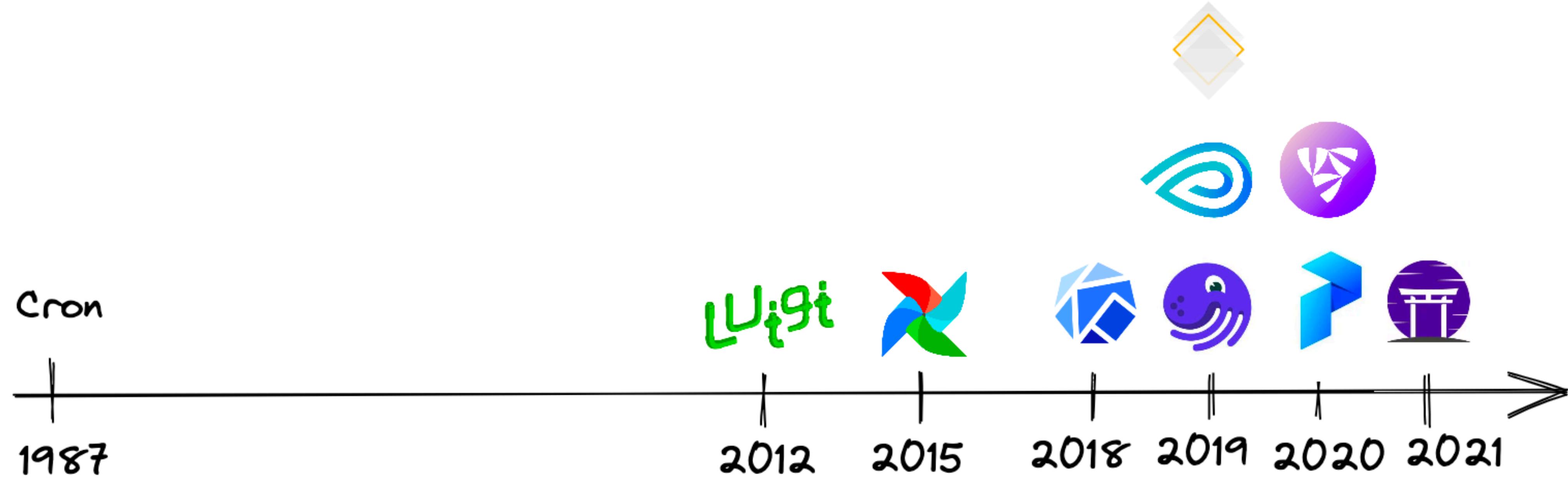
Understanding Workflow Management Systems

What They Are and Why They Matter



- Workflow management systems (WfMS) are software tools that help researchers and data scientists **automate** and **streamline complex data analysis workflows**
- WfMS enable users to **define, execute, and monitor tasks and dependencies** in a modular and scalable manner, making it **easier to manage and reproduce analyses**
- By using WfMS, researchers can **reduce the risk of errors** and inconsistencies in their analyses, while increasing the **efficiency and reproducibility** of their research.

Orchestration



keep software processes, workflows, systems and services running at the right time and in the right order

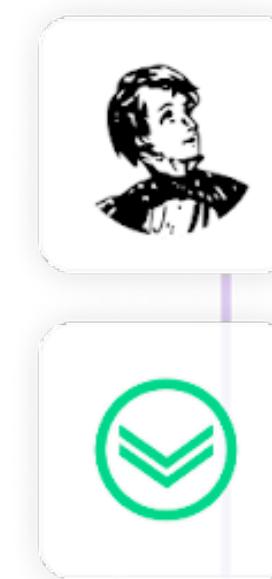
Data Orchestration

the process of gathering, cleansing, organizing and analyzing data
to make informed decisions

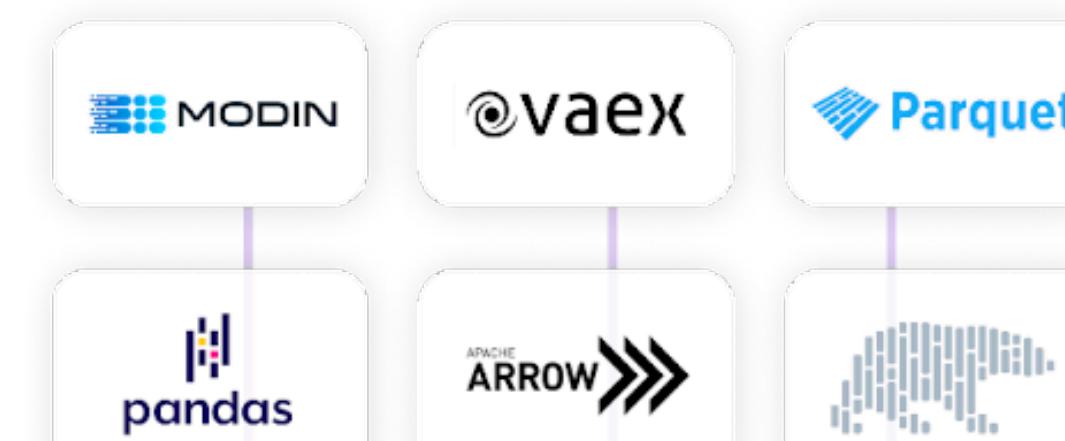
Machine Learning Orchestration

effective implementation, management and deployment of
production-grade ML pipelines to simplify the creation of building
ML-powered products

Data Quality



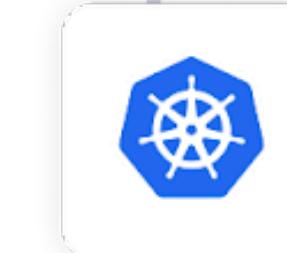
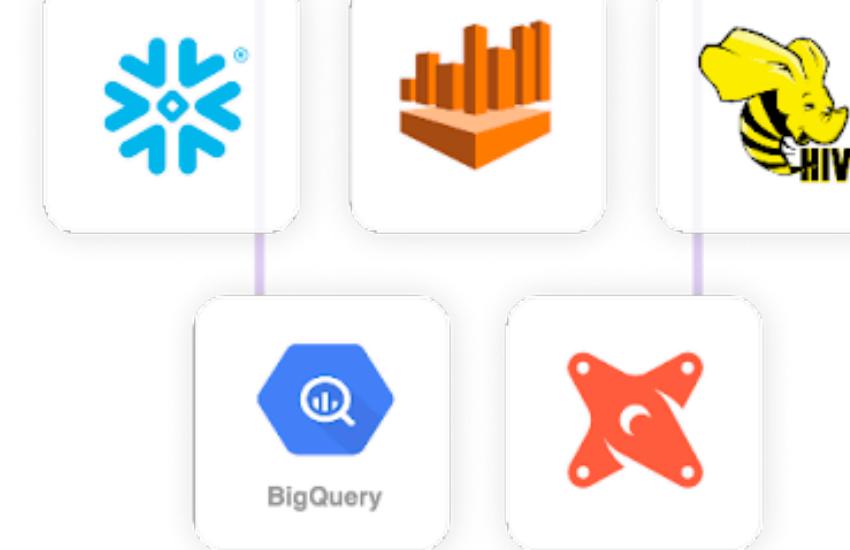
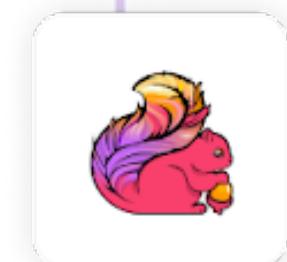
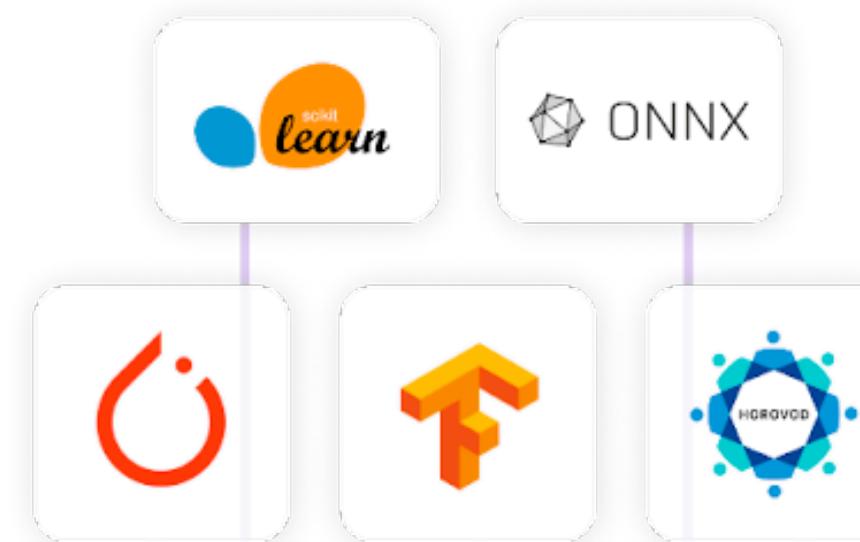
Data Frameworks



Programming languages



ML Frameworks



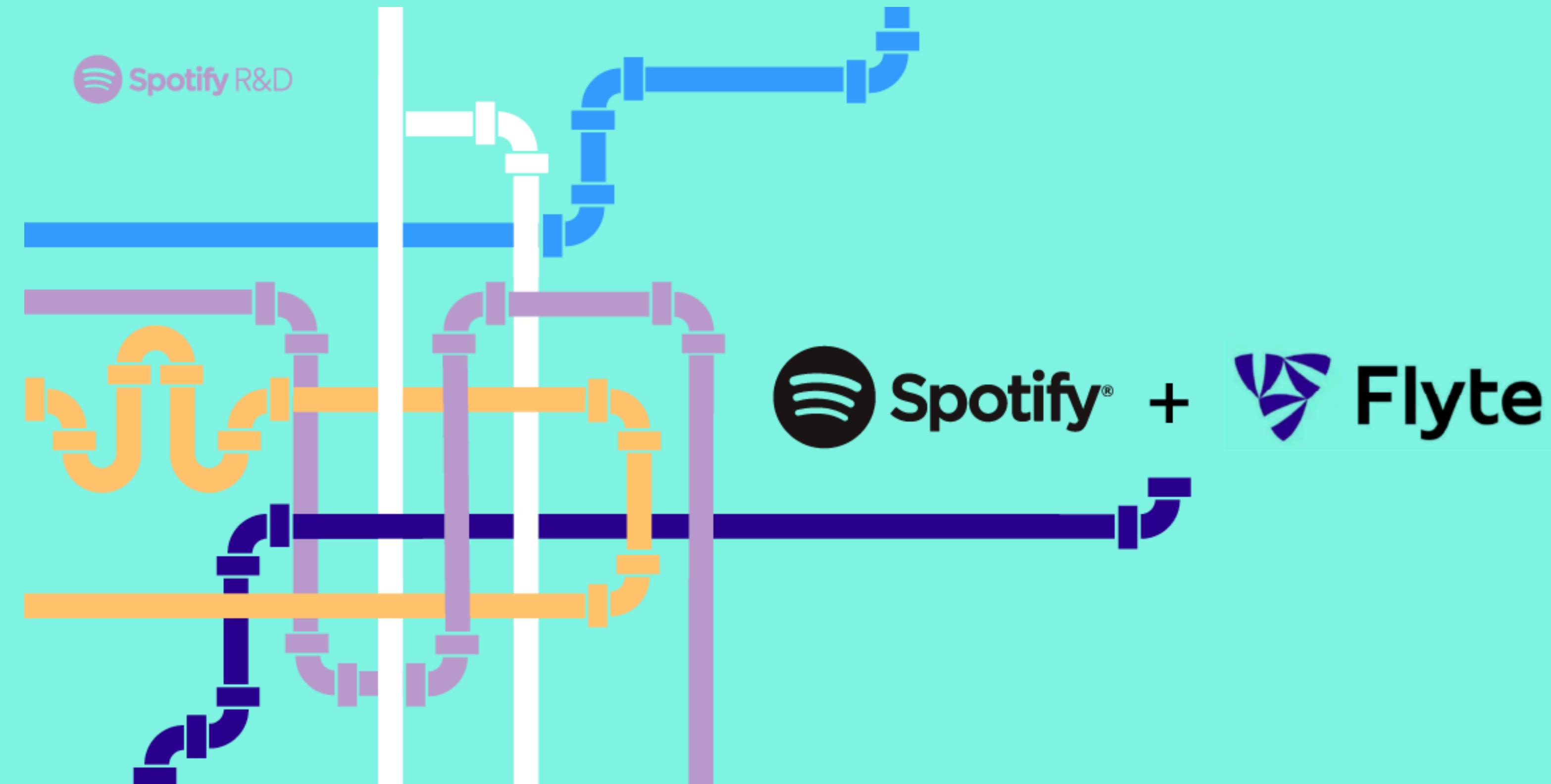
Data processing

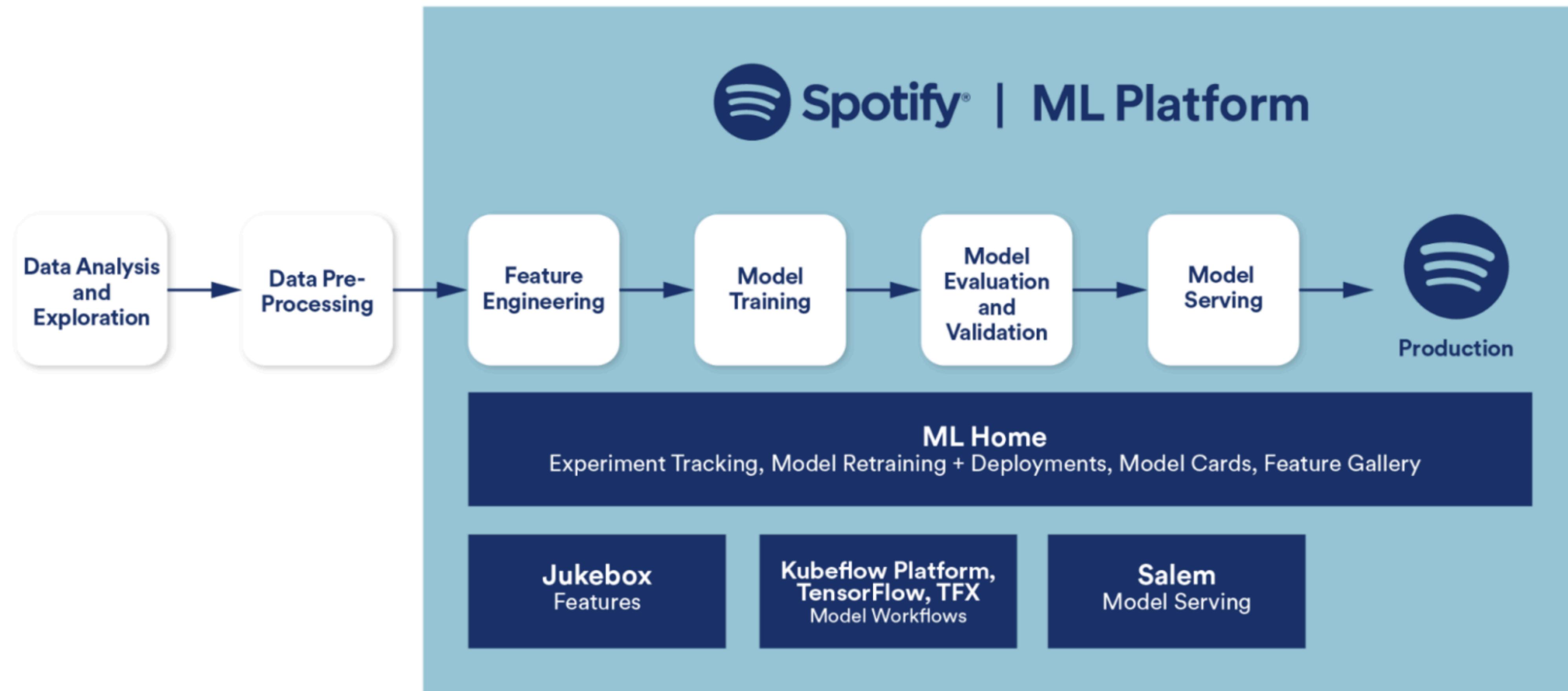
Data Warehouses & SQL Transforms

Training orchestration

Data Logging

Feature Store







```
@workflow_class
class WorkflowDemo:
    endpoint = Input(Types.String, 'endpoint')
    partition = Input(Types.Datetime, 'partition')

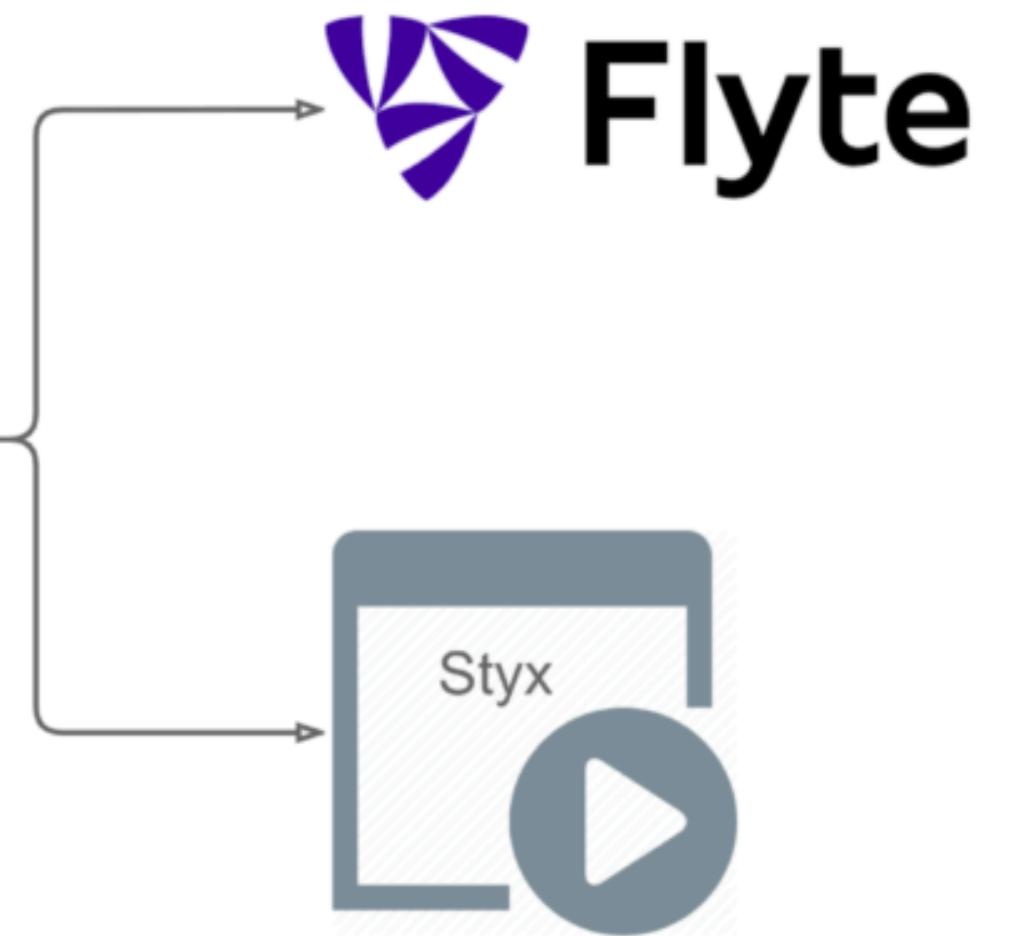
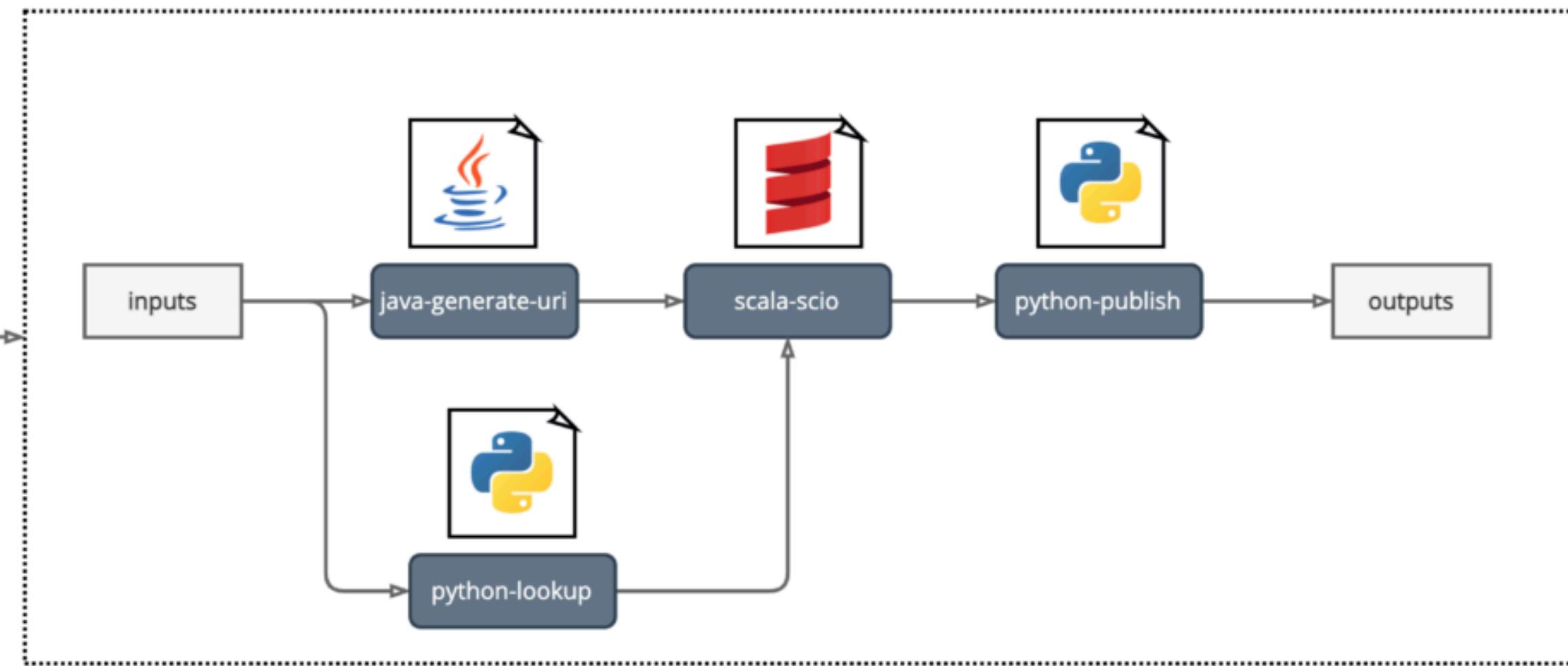
    python_lookup = python_LookupData(
        partition=partition,
        endpoint=endpoint)

    java_generate_uri = java_GenerateURI(
        partition=partition,
        endpoint=endpoint)

    scala_scio = scala_ScioTask(inputs=[python_lookup],
                                 outputs=[java_generate_uri])

    python_publish = python_Publish(
        partition=partition,
        endpoint=endpoint)

    Output(python_publish.outputs.out)
```



WfMS in Bioinformatics

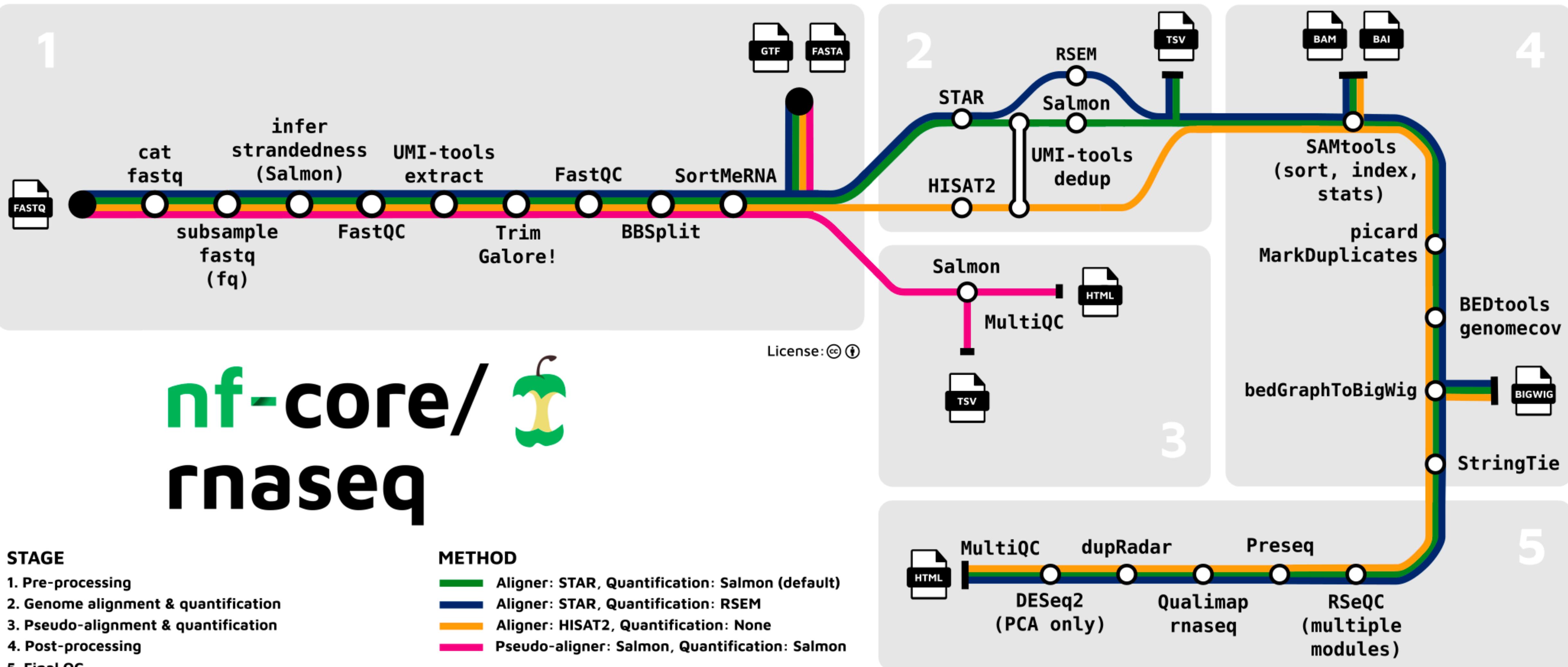
A gentle introduction

- Bioinformatics researchers often deal with **large and complex datasets** that require **advanced computational methods** and tools
- WfMS enable researchers to **automate and streamline** analyses, making it easier to manage and reproduce results
- We should design workflows that are **modular, scalable, and reproducible**, and using tools like **containerization** and **version control** to ensure **consistency and transparency** in the analysis pipeline

	Bioinformatics	Data Engineering
Size of data files	Large	Small
Data type	Compressed text, proprietary formats	Common formats (text, images, etc)
Number of data files	Small	Large
Compute Intensity per step	Medium to large	Small to medium
Store results in databases?	No	Yes

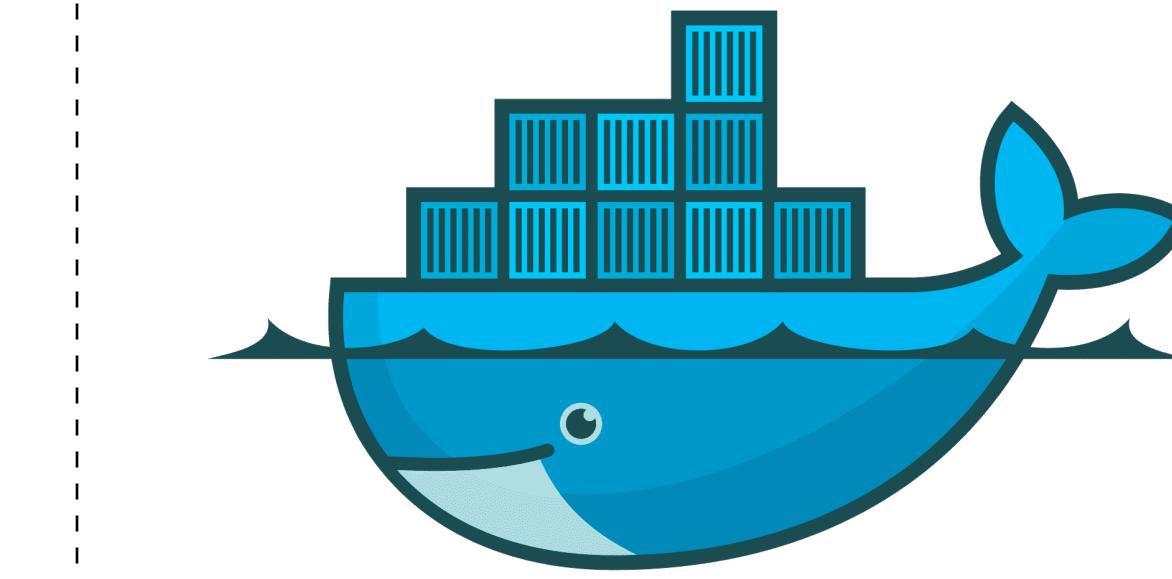
Tool	Class	Ease of use ^a	Expressiveness ^b	Portability ^c	Scalability ^d	Learning resources ^e	Pipeline initiatives ^f
Galaxy	Graphical	●●●	●○○	●●●	●●●	●●●	●●○
KNIME	Graphical	●●●	●○○	○○○	●●○	●●●	●●○
Nextflow	DSL	●●○	●●●	●●●	●●●	●●●	●●●
Snakemake	DSL	●●○	●●●	●●○	●●●	●●○	●●●
GenPipes	DSL	●●○	●●●	●●○	●●○	●●○	●●○
bPipe	DSL	●●○	●●●	●●○	●●○	●●○	●○○
Pachyderm	DSL	●●○	●●●	●○○	●●○	●●●	○○○
SciPipe	Library	●●○	●●●	○○○	○○○	●●○	○○○
Luigi	Library	●●○	●●●	●○○	●●○	●●○	○○○
Cromwell + WDL	Execution + workflow specification	●○○	●●○	●●●	●●○	●●○	●●○
cwltool + CWL	Execution + workflow specification	●○○	●●○	●●○	○○○	●●●	●●○
Toil + CWL/WDL/Python	Execution + workflow specification	●○○	●●●	●○○	●●●	●●○	●●○

Tool	Class	Ease of use ^a	Expressiveness ^b	Portability ^c	Scalability ^d	Learning resources ^e	Pipeline initiatives ^f
Galaxy	Graphical	●●●	●○○	●●●	●●●	●●●	●●○
KNIME	Graphical	●●●	●○○	○○○	●●○	●●●	●●○
Nextflow 	DSL	●●○	●●●	●●●	●●●	●●●	●●●
Snakemake 	DSL	●●○	●●●	●●○	●●●	●●○	●●●
GenPipes	DSL	●●○	●●●	●●○	●●○	●●○	●●○
bPipe	DSL	●●○	●●●	●●○	●●○	●●○	●○○
Pachyderm	DSL	●●○	●●●	●○○	●●○	●●●	○○○
SciPipe	Library	●●○	●●●	○○○	○○○	●●○	○○○
Luigi	Library	●●○	●●●	●○○	●●○	●●○	○○○
Cromwell + WDL	Execution + workflow specification	●○○	●●○	●●●	●●○	●●○	●●○
cwltool + CWL	Execution + workflow specification	●○○	●●○	●●○	○○○	●●●	●●○
Toil + CWL/WDL/Python	Execution + workflow specification	●○○	●●●	●○○	●●●	●●○	●●○



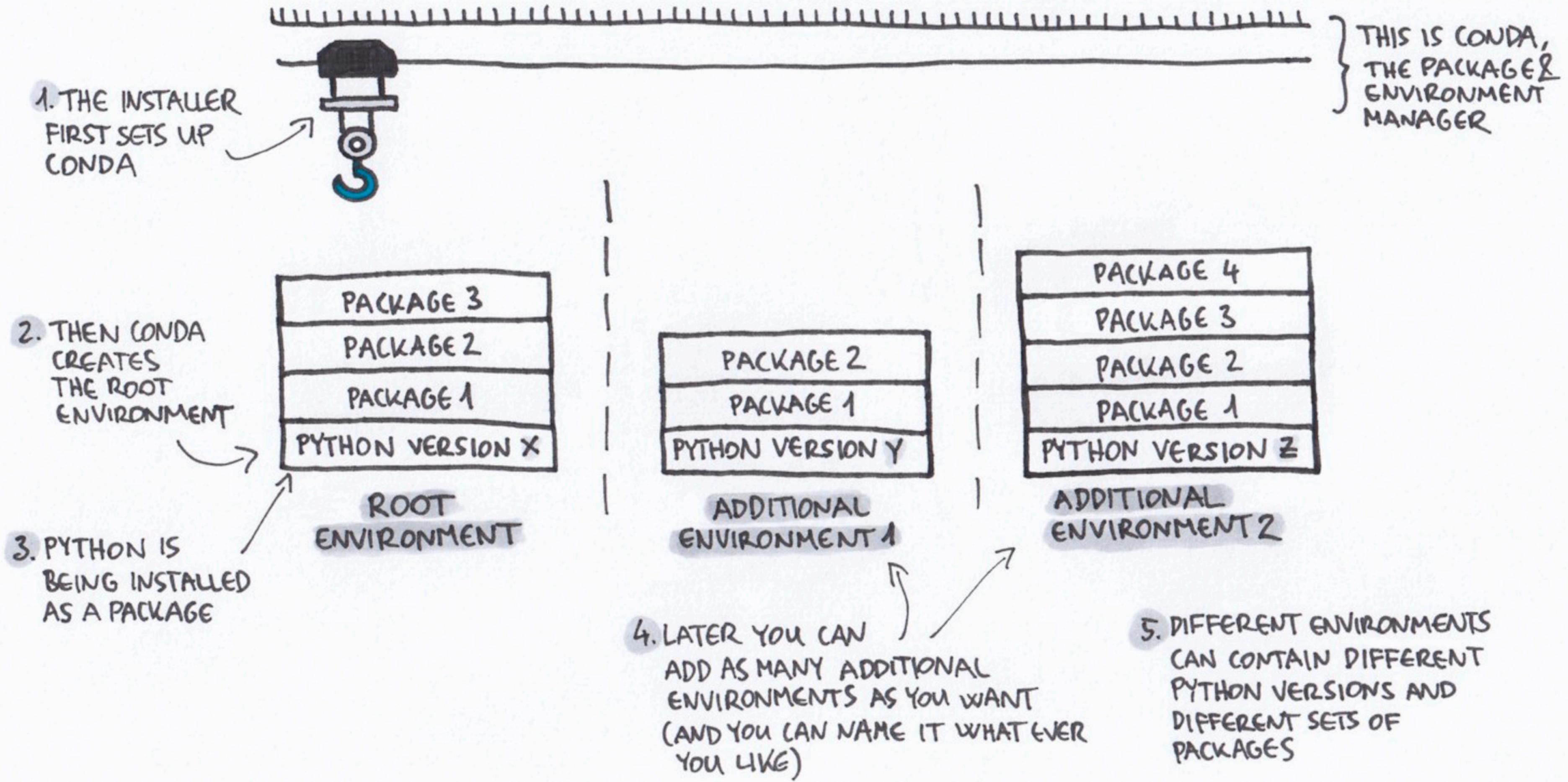
Environments and containers

Creating and managing reproducible computational environments



docker

- Conda is an open-source package manager that allows users to **easily create, manage, and share reproducible software environments** for data analysis and scientific computing
- Conda can **create isolated environments for different projects**, allowing users to install different versions of software and packages without conflicts
- Conda **environments can be easily shared and reproduced across different systems and platforms**, making it a useful tool for **collaboration and reproducibility**

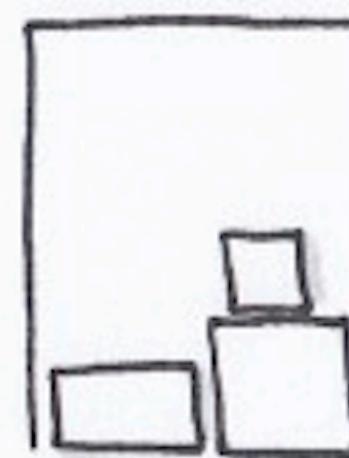




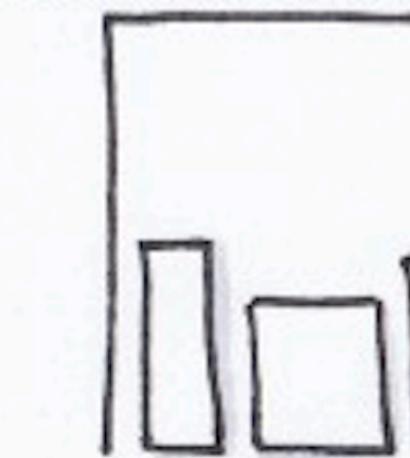
CONDA IS

LOOKING FOR
THIS PACKAGE

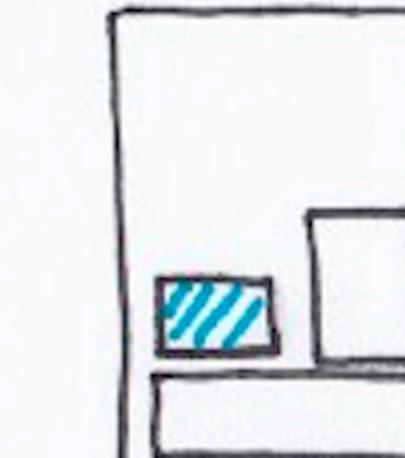
CHANNELS ARE LIKE STORAGES...



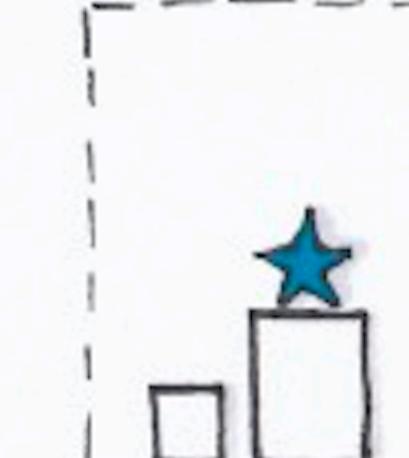
CHANNEL 1



CHANNEL 2



CHANNEL 3



CHANNEL 4

WHAT HAPPENS IF CONDA DID NOT FIND THE PACKAGE?

BY DEFAULT, CONDA LOOKS FOR PACKAGES IN THE OFFICIAL STORAGES OF CONTINUUM.

WHY? BECAUSE BY DEFAULT THESE HAVE THE HIGHEST PRIORITIES.

HOWEVER, YOU HAVE THE POWER TO

ADD NEW CHANNELS (STORAGES) THAT CONTAIN THE PACKAGES YOU NEED!

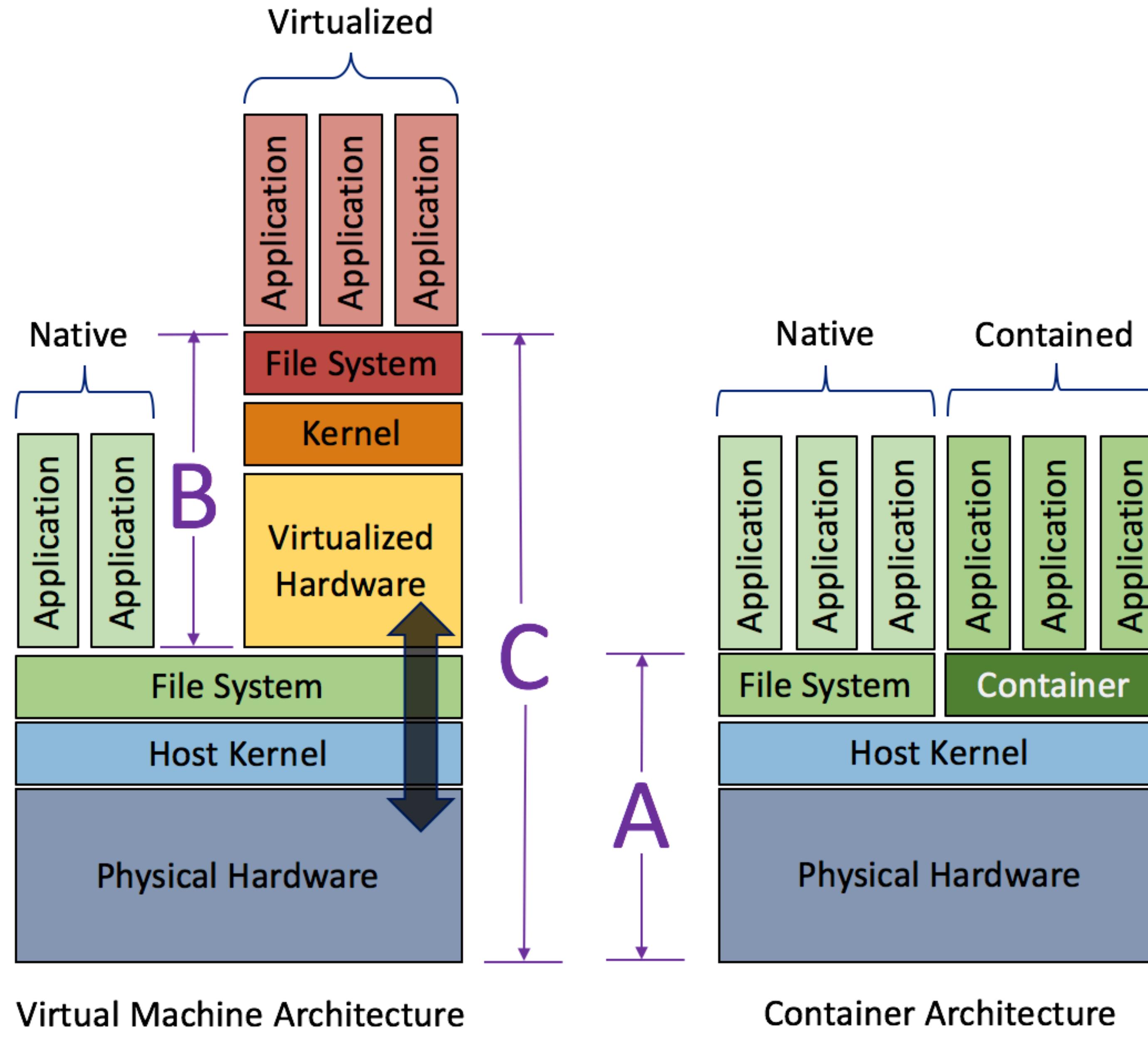
SO LET'S SAY YOU WANT TO INSTALL THIS PACKAGE:

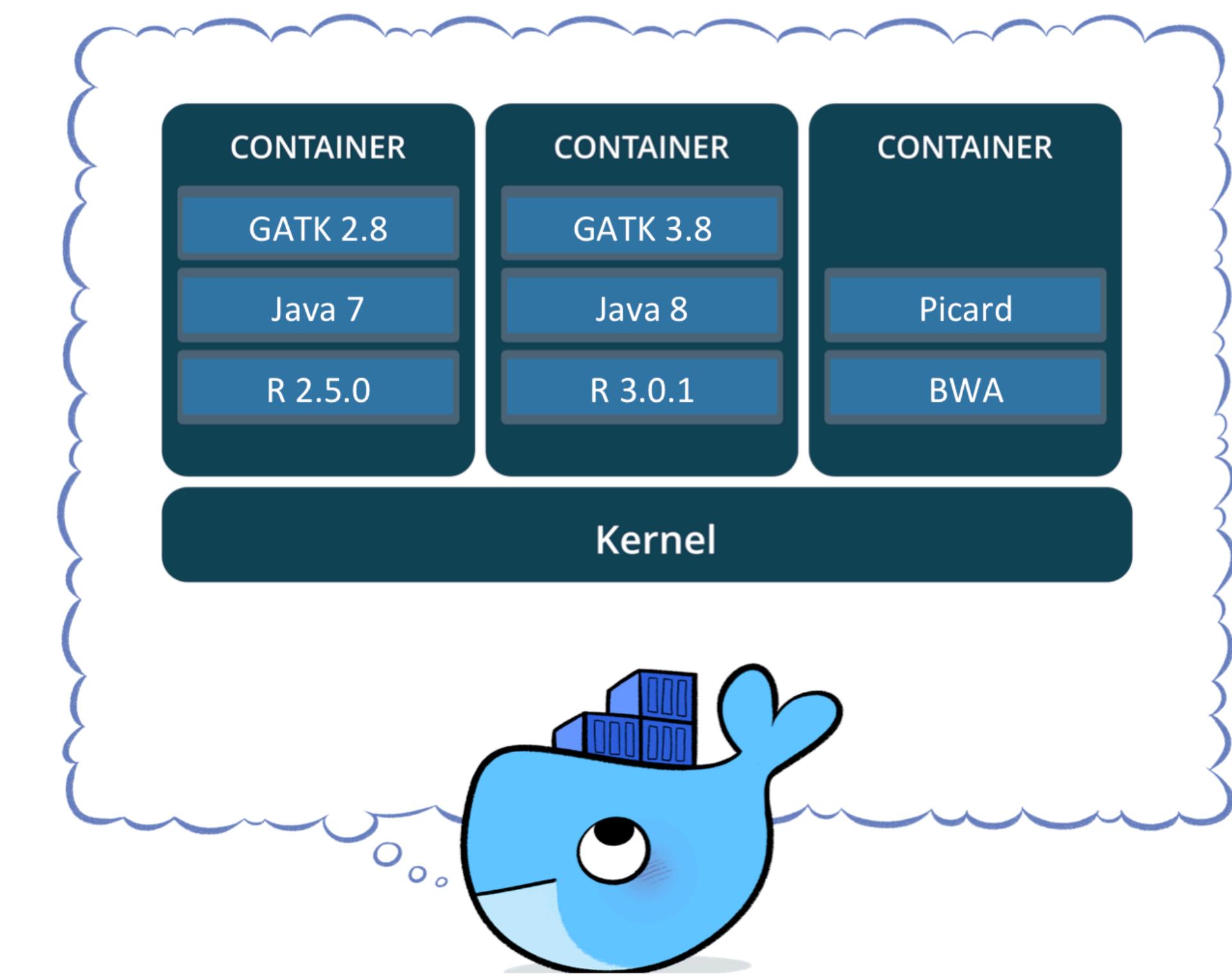
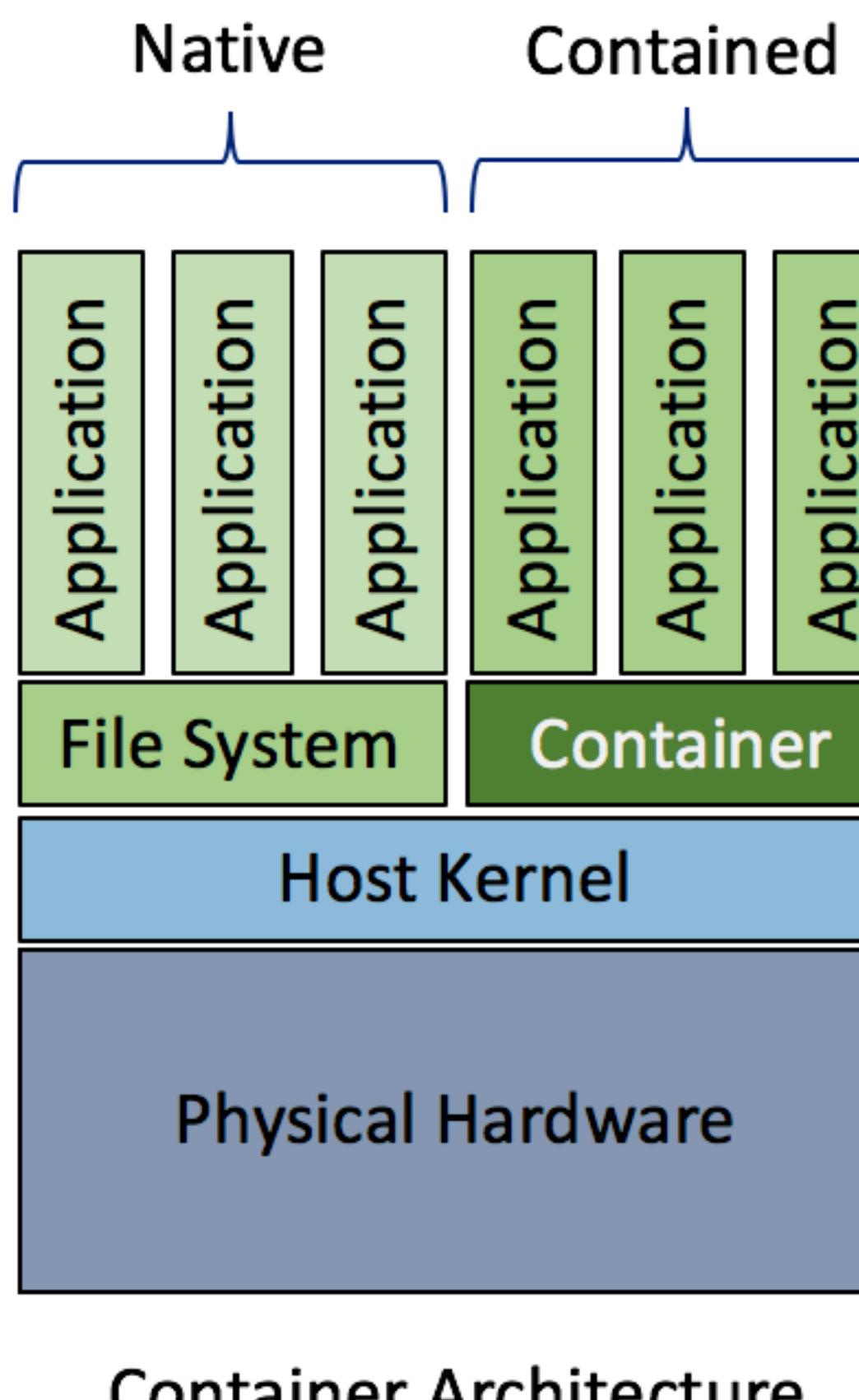
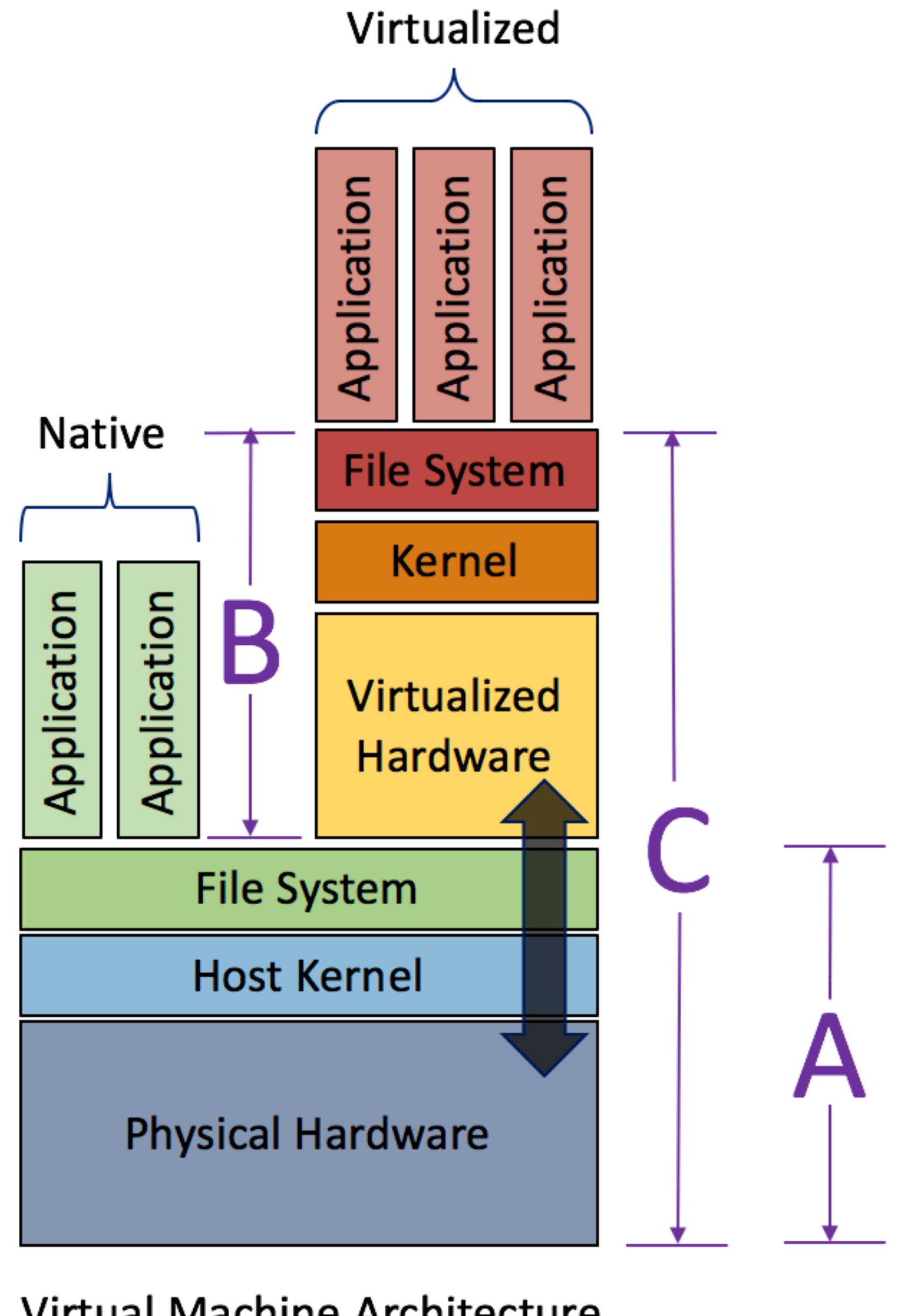
→ CONDA DOES NOT FIND IT IN THE FIRST 3 CHANNELS

⇒ YOU NEED TO ADD A 4TH ONE!

THEY ARE
CONDA'S
DEVELOPERS.

- Docker is a **containerization platform** that allows users to **package an entire software environment**, including dependencies and configuration settings, into a **portable container** that can be run on any system with Docker installed.
- Docker containers can be used **to ensure that a software environment is exactly the same across different machines**, making it easier to reproduce computational analyses and experiments.

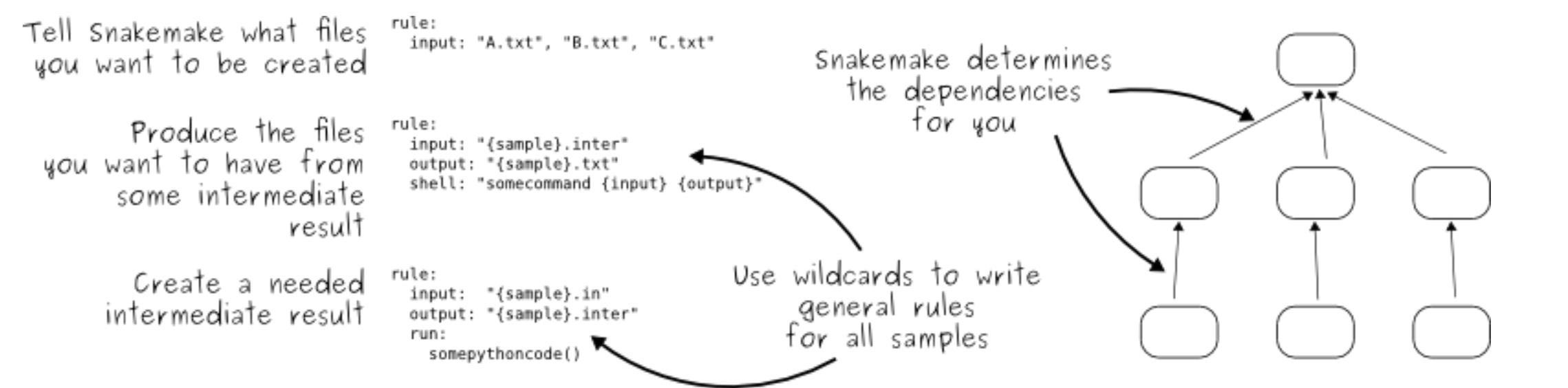


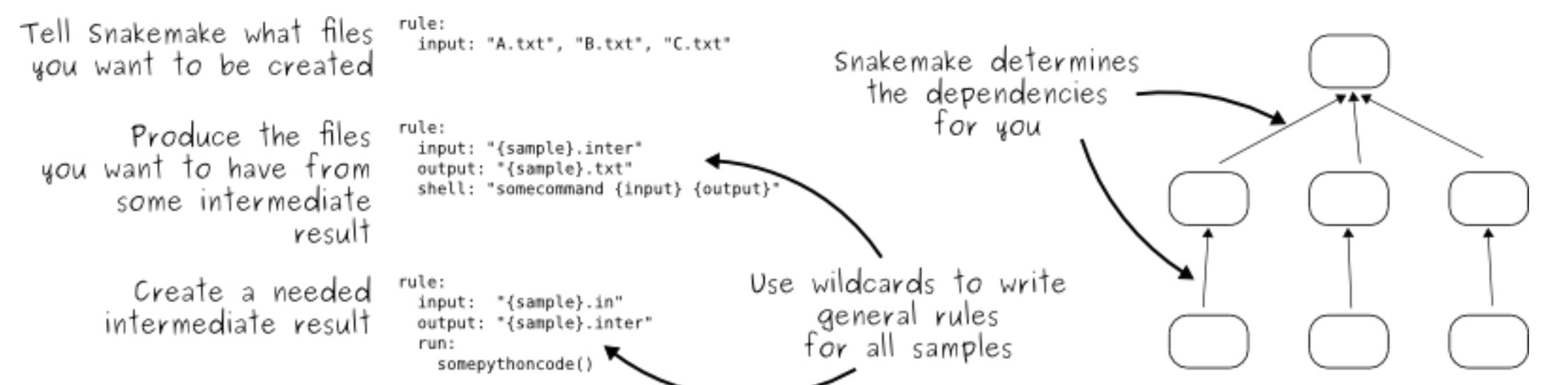


An introduction to Snakemake

A Python-based WfMS for reproducible research

- Snakemake is a WfMS that uses a **Python-based DSL** to define and execute workflows. It is particularly well-suited for data analysis pipelines that require **complex dependencies and parallelization**
- Snakemake enables researchers to **build and manage workflows in a modular and scalable manner**, and provides a range of features and tools for reproducible research, including **version control, containerization, and automated testing**
- Snakemake is widely used in bioinformatics for data processing and analysis, and has **a large and active user community**.





```

1 configfile: "config.yaml"
2
3 module some_workflow:
4     snakefile: "https://github.com/some/raw/v1.0.0/Snakefile"
5     config: config["some"]
6
7 use rule * from some_workflow
8
9 use rule simulate_data from some_workflow with:
10    params:
11        some_threshold=1.e-7
12
13 module other_workflow:
14     snakefile: "https://github.com/other/raw/v1.0.0/Snakefile"
15     config: config["other"]
16
17 use rule * from other_workflow as other_*
18
19 rule some_plot:
20     input:
21         "results/tables/all.csv"
22     output:
23         "results/plots/all.svg"
24     conda:
25         "envs/stats.yaml"
26     notebook:
27         "notebooks/some-plot.py.ipynb"

```

- Legend**
- declare module
 - use rules
 - modify rule
 - extend workflow

Tell Snakemake what files you want to be created

Produce the files you want to have from some intermediate result

Create a needed intermediate result

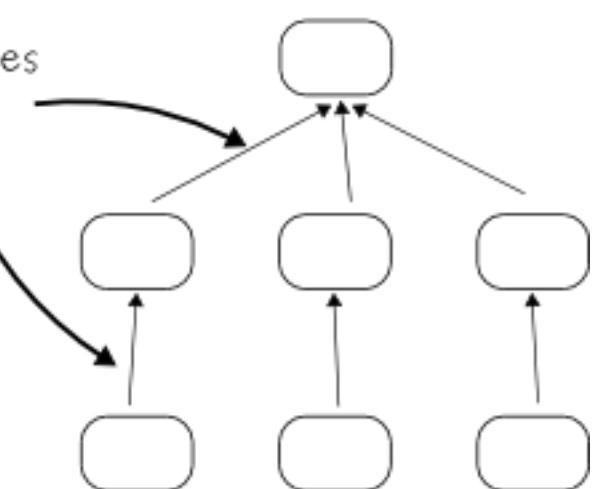
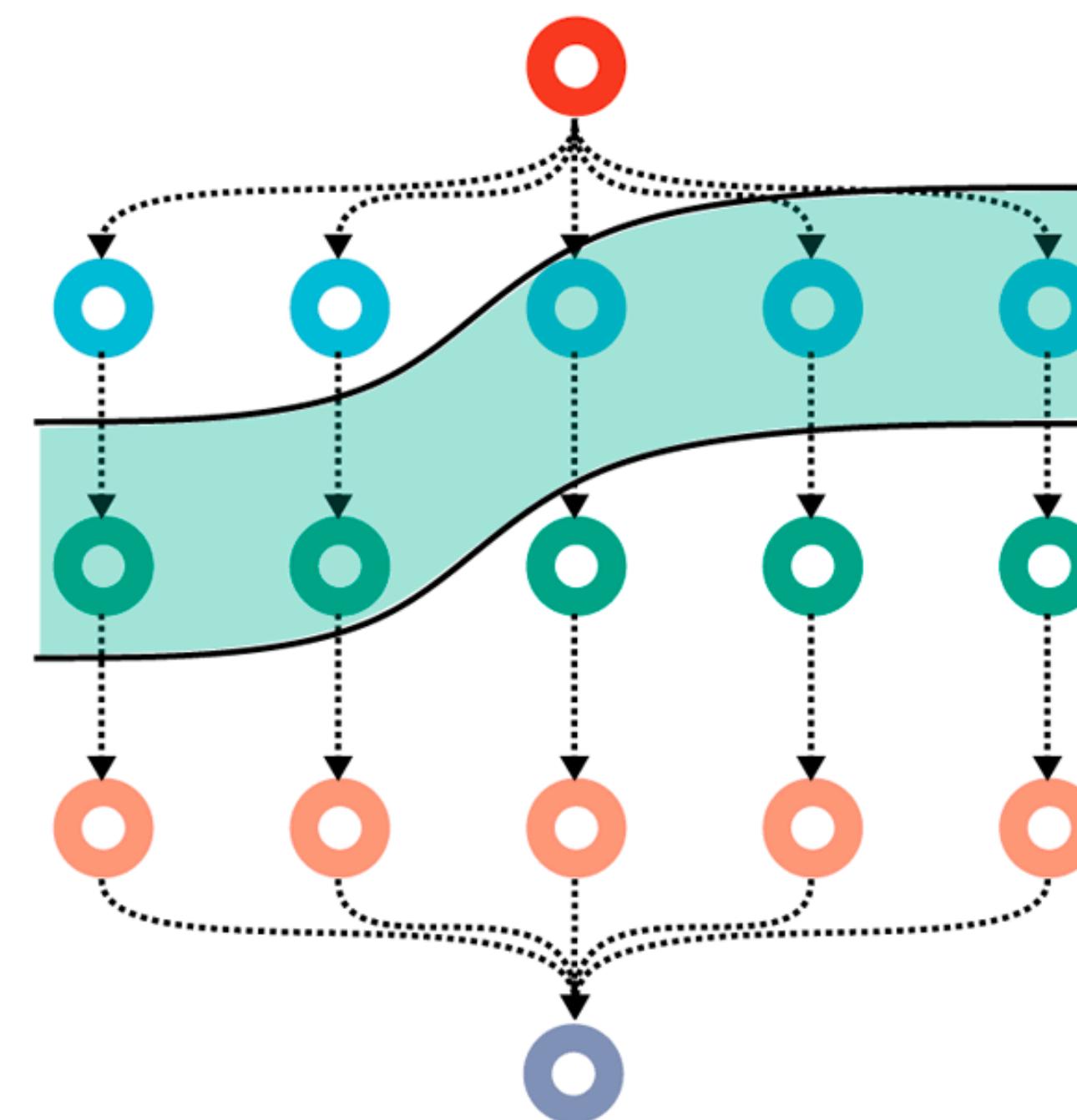
```
rule:  
    input: "A.txt", "B.txt", "C.txt"
```

```
rule:  
    input: "{sample}.inter"  
    output: "{sample}.txt"  
    shell: "somecommand {input} {output}"
```

```
rule:  
    input: "{sample}.in"  
    output: "{sample}.inter"  
    run:  
        somepythoncode()
```

Snakemake determines the dependencies for you

Use wildcards to write general rules for all samples

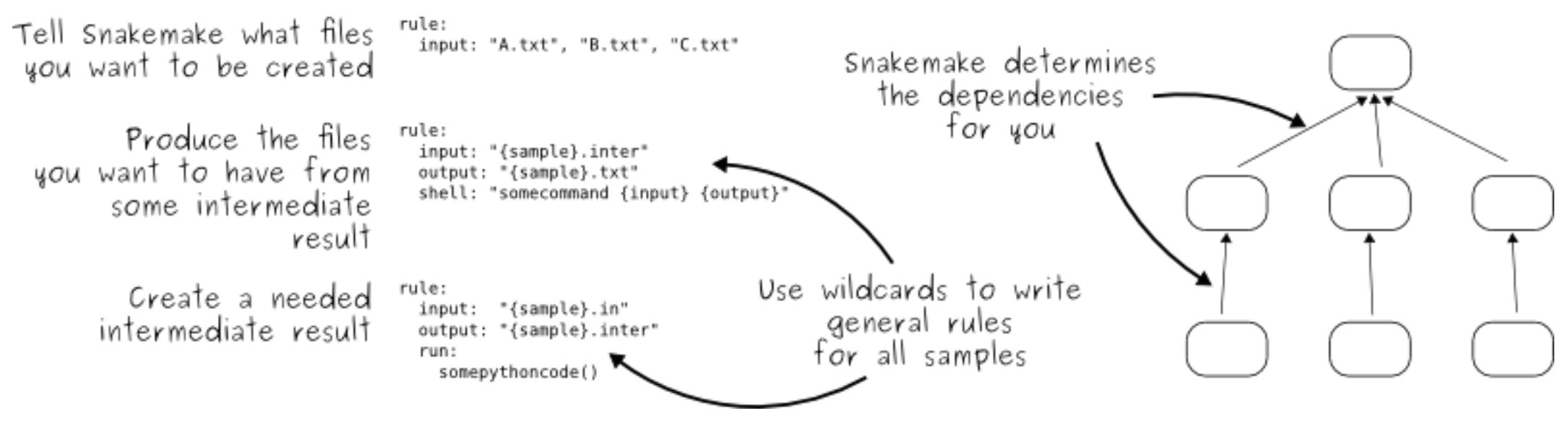
**a****b****cores****cores****c**

a

```

1 configfile: "config.yaml"
2
3 rule all:
4     input:
5         expand(
6             "results/plots/{country}.hist.pdf",
7             country=config["countries"]
8         )
9
10 rule download_data:
11     output:
12         "data/worldcitiespop.csv"
13     log:
14         "logs/download.log"
15     conda:
16         "envs/curl.yaml"
17     shell:
18         "curl -L https://burntsushi.net/stuff/worldcitiespop.csv > {output} 2> {log}"
19
20 rule select_by_country:
21     input:
22         "data/worldcitiespop.csv"
23     output:
24         "results/by-country/{country}.csv"
25     log:
26         "logs/select-by-country/{country}.log"
27     conda:
28         "envs/xsv.yaml"
29     shell:
30         "xsv search -s Country '{wildcards.country}' "
31             "{input} > {output} 2> {log}"
32
33 rule plot_histogram:
34     input:
35         "results/by-country/{country}.csv"
36     output:
37         "results/plots/{country}.hist.svg"
38     container:
39         "docker://faizanbashir/python-datasience:3.6"
40     log:
41         "logs/plot-hist/{country}.log"
42     script:
43         "scripts/plot-hist.py"
44
45 rule convert_to_pdf:
46     input:
47         "{prefix}.svg"
48     output:
49         "{prefix}.pdf"
50     log:
51         "logs/convert-to-pdf/{prefix}.log"
52     wrapper:
53         "0.47.0/utils/cairosvg"

```



Hands-on session

<https://github.com/GeoGenetics/data-analysis-2025/reproducible-data-analysis>