EBRAINS workflow
Workflow Components

Task 4.3

Florent Bonnier

Paris-Saclay Institute of Neuroscience
`florent.bonnier@cnrs.fr`

EBRAINS Context

- Various tools and services provided by EBRAINS
- Multiplication of tools/services increases dependencies concurrency and complexity

$\rightarrow$ Provide the healthiest environnement(s) able to run the tools together to maximize:

- stability
- flexibility
- reproducibility
- supported hardware
- large datasets support

$\rightarrow$ Containerization

$\rightarrow$ Workflows

Section 1

Containerization

Encapsulation

- A software is a singular system involving requirements, inputs, environment, hardware ...
- Encapsulating the construction of dedicated system

Computation is a social society

- Main goal is to compute on a maximum types of hardware (HPC, cloud, SoC, VM)
- Unbreakable relations between the machine and the encapsulated system
- Example: better use of on dedicated HPC compilers

Containerization

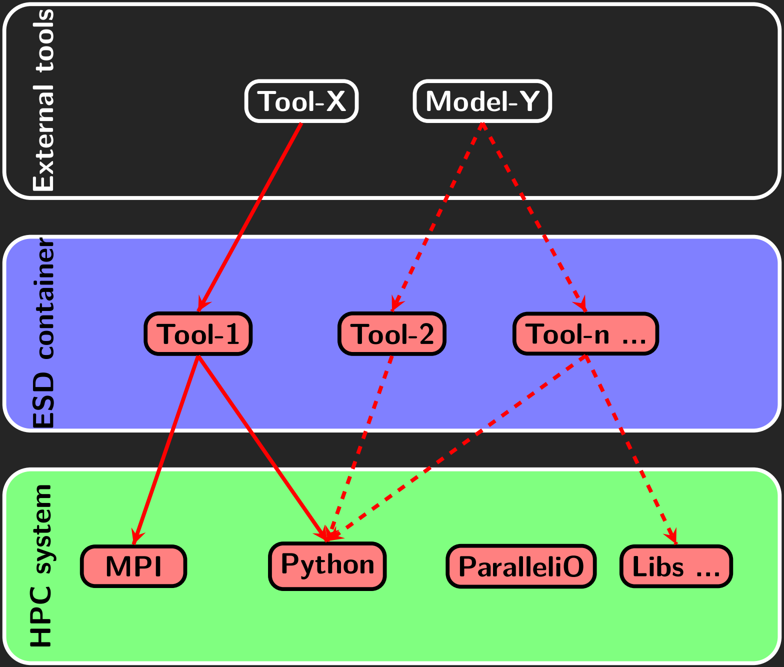
A **container** is a portable independent environment able to compute on it's own.

A **container** is a single instance of an **image**

A container can be used to encapsulate a single application or an entire operating system.

HPC systems have optimized libraries

$\rightarrow$ containers must be encapsulated **and** connected to local host environment


External tools
Tool-X
Model-Y
ESD container
Tool-1
Tool-2
Tool-n ...
HPC system
MPI
Python
ParalleliO
Libs ...

Docker containers

Docker containers are built from **Dockerfiles**

```
# Build stage with Spack pre-installed and ready to be used
FROM spack/ubuntu-jammy:develop as builder

# Copy manifest file (spack.yaml)
COPY ./spack.yaml /opt/spack-environment/spack.yaml

# Get the additional EBRAINS package definitions
# TODO: here we get the main branch, we may wish to use a release tag
RUN git clone https://gitlab.ebrains.eu/ri/tech-hub/platform/esd/ebrains-spack-
    ↪ builds.git /opt/ebrains-spack-builds

# Install the software, remove unnecessary deps
RUN cd /opt/spack-environment && spack repo add /opt/ebrains-spack-builds &&
    ↪ spack env activate . && spack install --fail-fast && spack gc -y
```

Docker containers can be built from multiple Docker containers

```
FROM ubuntu:22.04

RUN apt-get update && apt-get install -y gfortran

# Copy data from other container
COPY --from=builder /opt/spack-environment /opt/spack-environment
COPY --from=builder /opt/software /opt/software
```

Section 2

Workflows

Workflows

Definition[1]

A workflow is a well-defined, and possibly repeatable, pattern or systematic organization of activities designed to achieve a certain transformation of data.

- Business definition adapted to scientific processes
- **Processes and activities**: set of tasks

- Reusability and reproducibility
- High level of abstraction

- Higlhy depends on data movement mechanisms
- Highly depends on tools interface

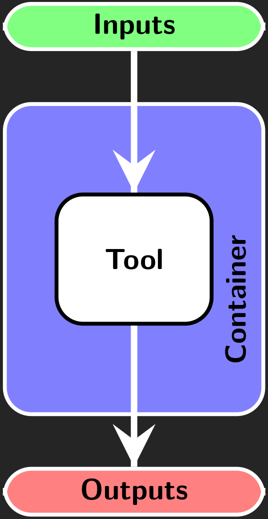
[1] *Workflow Systems for Science: Concepts and Tools* - Domenico Talia - `http://dx.doi.org/10.1155/2013/404525`

Common Workflow Language

- Language dedicated to workflows description
- Based on 2 files
  - $\boxed{\textit{.cwl}}$ script file
  - $\boxed{\textit{.yaml}}$ / $\boxed{\textit{.json}}$ inputs file
- CLI based instructions
- Use of Docker/Singularity/Apptainer environments, including userspace/rootless containers: **Podman**, **uDocker**
- Automatic image pull

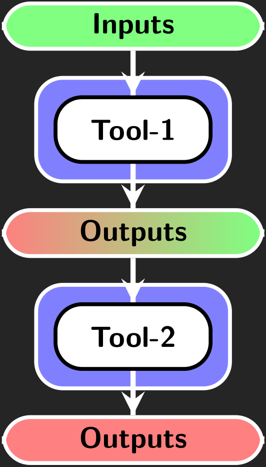
```
cwltool my-script.cwl my-inputs.yaml
```

1 tool → 1 CWL script



```
class: CommandLineTool
cwlVersion: v1.0
id: my-tool-id
label: My Tool is awesome

baseCommand: ["CallOfTool.exe"]

requirements:
  - class: DockerRequirement
    dockerPull: docker-registry.ebrains.eu/my-docker

inputs:
  input-1:
    type: string
    inputBinding:
      position: 1
      prefix: --input-1

outputs:
  output-A:
    type: int
```

1 workflow → 1 CWL script



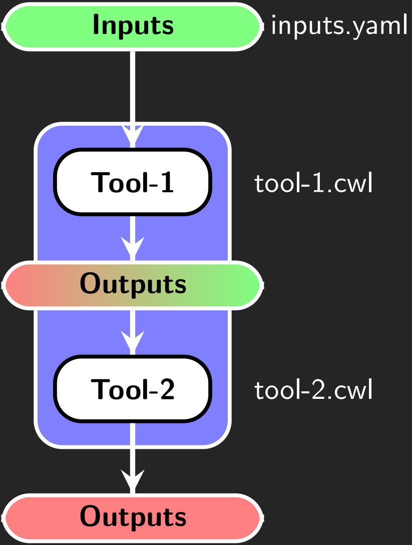
inputs.yaml

tool-1.cwl

tool-2.cwl

```
class: Workflow
cwlVersion: v1.0
id: my-workflow

inputs:
  input-tool-1: <type>

outputs:
  output-tool-2:
    type: <type>
    outputSource: tool-2/output-tool-2

steps:

  tool-1:
    run: tool-1.cwl
    in:
      input-tool-1: input-tool-1
    out: [output-tool-1]

  tool-2:
    run: tool-2.cwl
    in:
      input-tool-2: tool-1/output-tool-1
    out: [output-tool-2]
```

1 workflow → 1 CWL script



```
class: Workflow
cwlVersion: v1.0
id: my—workflow
requirements:
  — class: DockerRequirement
    dockerPull: docker—registry.ebrains.eu
          ↳ /my—docker

inputs:
  input—tool—1: <type>

outputs:
  output—tool—2:
    type: <type>
    outputSource: tool—2/output—tool—2

steps:

  tool—1:
    run: tool—1.cwl
    in:
      input—tool—1: input—tool—1
    out: [output—tool—1]

  tool—2:
    run: tool—2.cwl
    in:
      input—tool—2: tool—1/output—tool—1
    out: [output—tool—2]
```

Section 3

Workflow-components

Workflow components repository

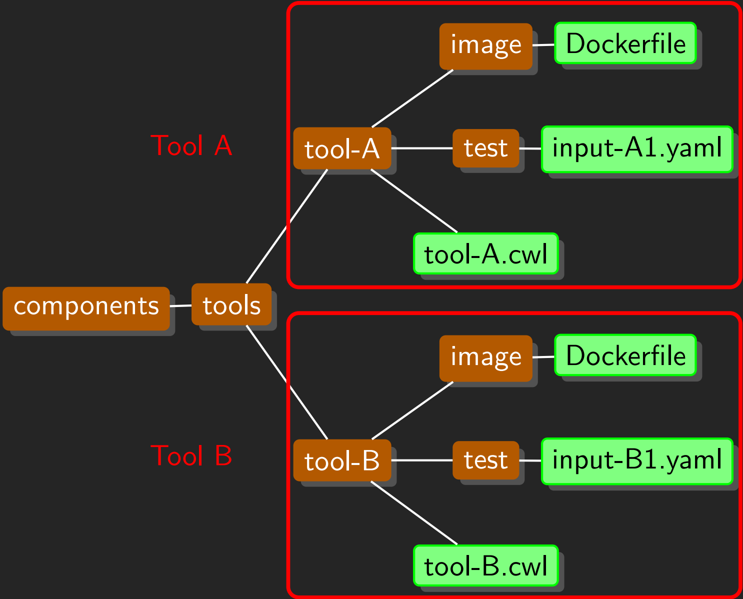
`https://gitlab.ebrains.eu/workflows/components`

Internal

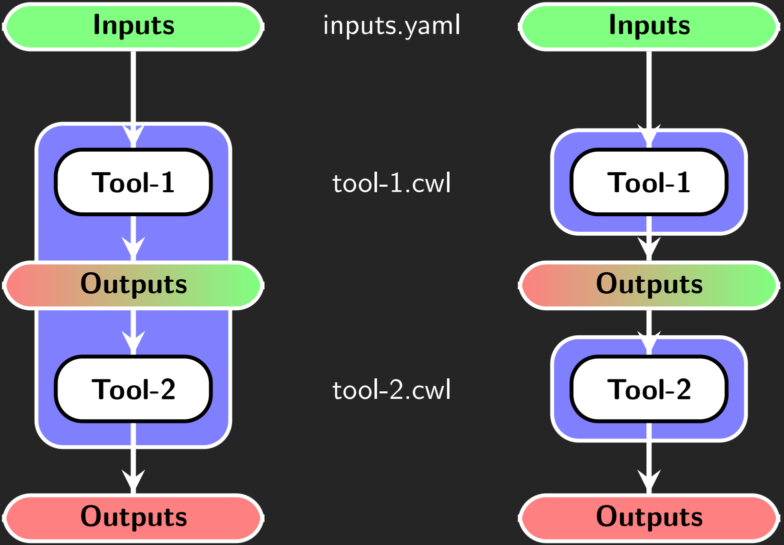
- Adapt EBRAINS tools for complex workflows
- Identify requirements and environments
- Implement CWL interfaces
- Test the tools
- Integrate tools to ESD

200 EBRAINS tools identifed	
55 candidates	98 not candidate
Arbor	Neural Activity Browser
Elephant	OpenMINDS
fMRIPrep	Synaptome.db

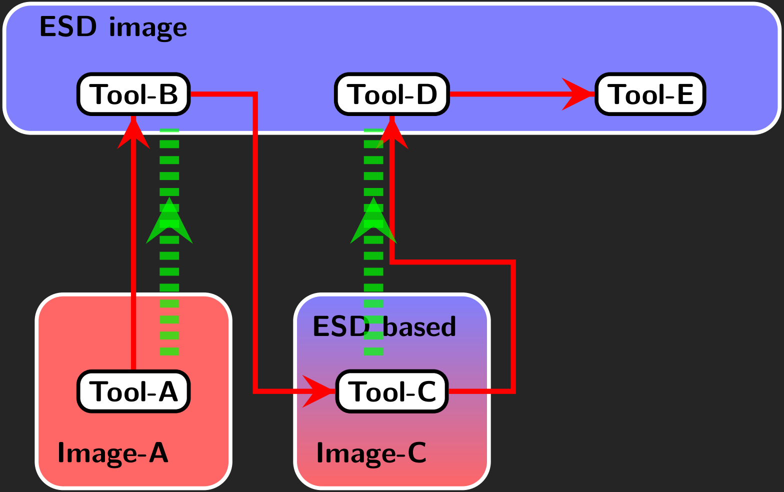
Workflow components repository



Workflow component's container



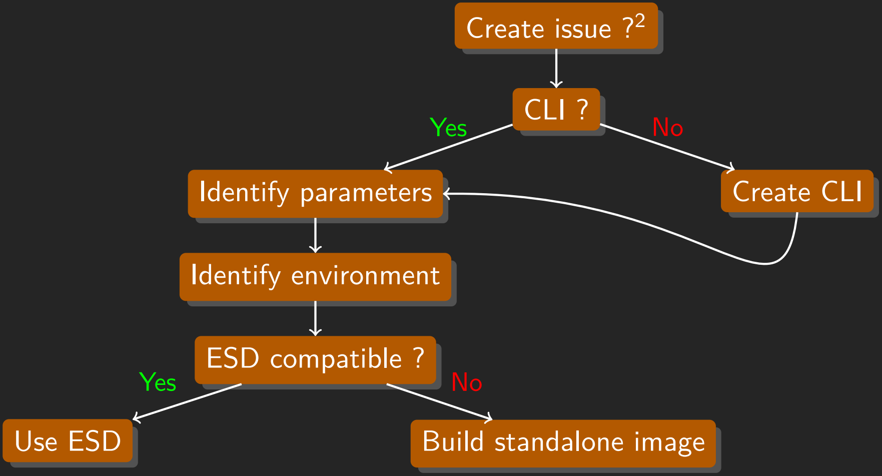
Workflow component's container



Section 4

Create a workflow component

Create a workflow component



[2]`https://gitlab.ebrains.eu/workflows/components/-/issues`

Section 5

Elephant Demonstrator

Elephant demonstrator

Generic analysis functions for spike train data and time series.
2 functions have been implemented yet:

- **wavelet_transform_cli.py**: Compute the wavelet transform of a given signalwith Morlet mother wavelet,
- **butterworth_filter_cli.py**: Butterworth filtering function.

Implemented by:

- Michael Denker
- Moritz Kern
- Cristiano Köhler

Elephant wavelet-transform

```
cwlVersion: v1.2

class: CommandLineTool
baseCommand: wavelet_transform_cli.py

hints:
  DockerRequirement:
    dockerImageId: docker-registry.ebrains.eu/workflow-components/elephant

    label: elephant-wavelet-transform

inputs:
  input_file:
    type: File
    label: "A file, containing sampled signals, that can be read by Neo"
    inputBinding:
      prefix: --input_file
  input_format:
    type: string?
    label: "Format of the input data, as a Neo IO class name (optional; TODO: use
        ↳ openMINDS content-types instead?)"
    inputBinding:
      prefix: --input_format
[...]

outputs:
  output_file:
    type: File
    outputBinding:
      glob: "$(inputs.output_file)"
```

Elephant butterworth-filter

```
cwlVersion: v1.2

class: CommandLineTool
baseCommand: butterworth_filter_cli.py

hints:
    DockerRequirement:
        dockerImageId: docker-registry.ebrains.eu/workflow-components/elephant

label: elephant-butterworth-filter

inputs:
  input_file:
    type: File
    label: "A file, containing sampled signals, that can be read by Neo"
    inputBinding:
      prefix: --input_file
  input_format:
    type: string?
    label: "Format of the input data, as a Neo IO class name (optional; TODO: use
        ↳ openMINDS content-types instead?)"
    inputBinding:
      prefix: --input_format
[...]
outputs:
  output_file:
    type: File
    outputBinding:
      glob: "$(inputs.output_file)"
```

Elephant butterworth-filter

```
cwlVersion: v1.2
class: Workflow

inputs:
  input_file:[...]
  highpass_frequency:[...]
  lowpass_frequency: [...]
[...]
outputs:
  filtered_output_file: [...]
  wavelet_output_file: [...]

steps:
  step_butterworth_filter:
    run: ./butterworth_filter.cwl
    in:
      input_file: input_file
      input_format: input_format
      [...]
    out: [output_file]

  step_wavelet_transform:
    run: ./wavelet_transform.cwl
    in:
              input_file: step_butterworth_filter/output_file
              input_format: input_format
              output_file: wavelet_output_file
              frequency: frequency
              n_cycles: n_cycles
              sampling_frequency: sampling_frequency
    out: [output_file]
```