Towards Continuous Preservation

Preserving Computational Workflows

PV2023

Jurek Oberhauser, Rafael Gieschke, Klaus Rechert, Dirk von Suchodoletz

Problem

How can we preserve scientific computational workflows?

Basic terms

Workflow: Executable process, describes order and dependencies of *Tools*

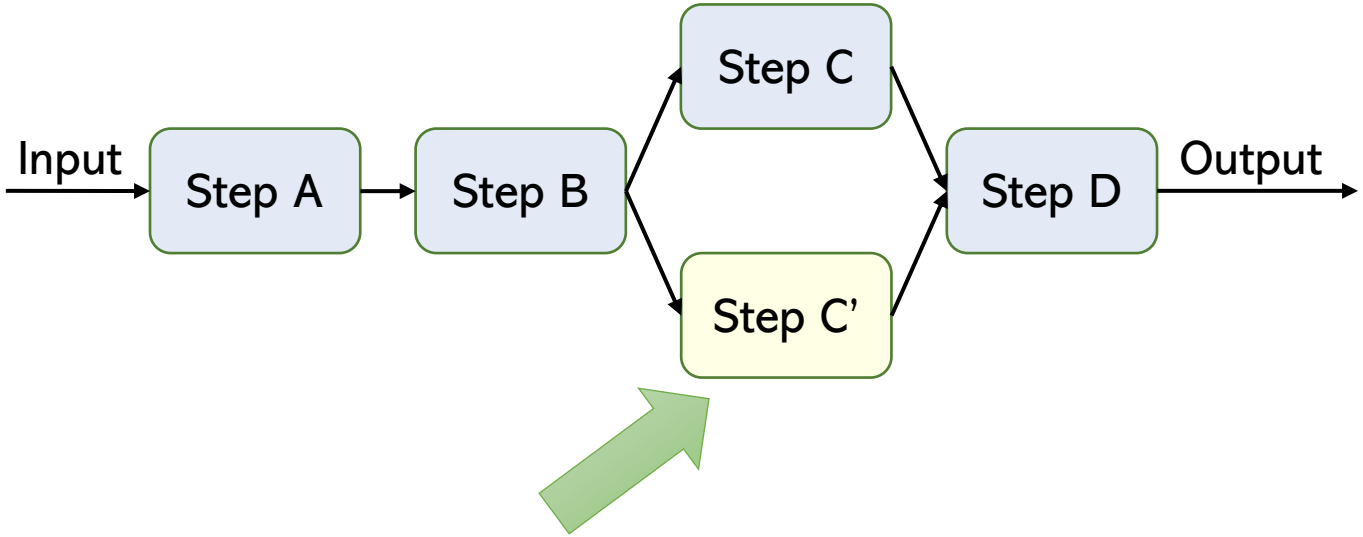
Tool: Component of a *Workflow*. Usually Program/Script that can be executed on the command line

Workflow Preservation

- Workflow Specifications, e.g., CWL, WDL, … → „Recipe“
- Problem: Preservation of workflow components
  - Components often Container-Images

Preservation of Container-Images → **Integration in Workflows**

Workflow Preservation



Preserved Workflow component

CWL

- Workflow Specification:
  Order + Dependencies
- **Workflow** and **CommandLineTool**
- Workflow: Steps -> Tools/Workflows
- Executed by CWL *Runner*
- Execution : Input File (Job File) + CWL File

input.yml

```yaml
map_otu_table:
  class: File
  path: test-input/test-otu

map_query:
  class: File
  path: test-input/test-mapseq
map_label: 'test'

return_dirname: returnDir
```

biom-convert.cwl

```yaml
class: CommandLineTool

hints:
  DockerRequirement:
    dockerPull: quay.io/biocontainers/biom-format:2.1.6--py36_0

baseCommand: [ biom, convert ]

inputs:
  biom:
    type: File?
    format: edam:format_3746  # BIOM
    inputBinding:
      prefix: --input-fp

  ...

outputs:
  result:
    type: File
    outputBinding:
      glob: _hdf5.biom
```

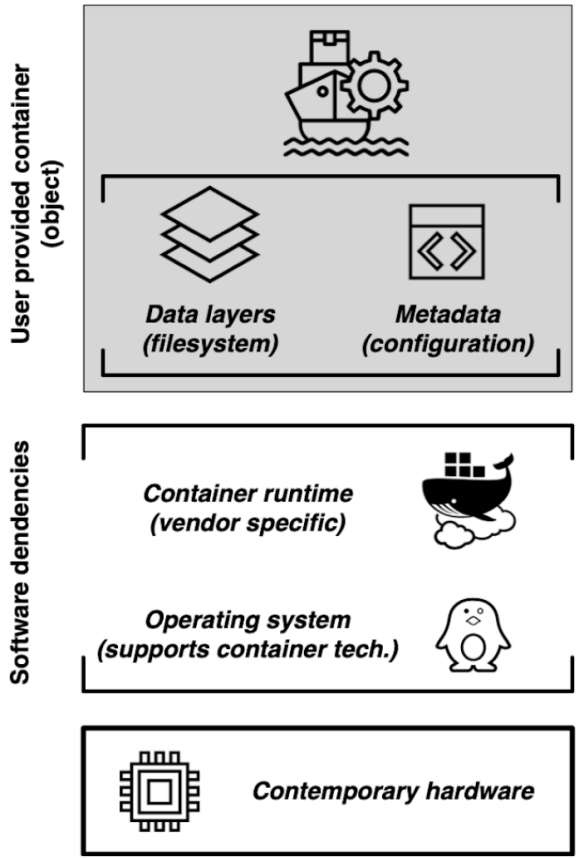
workflow.cwl

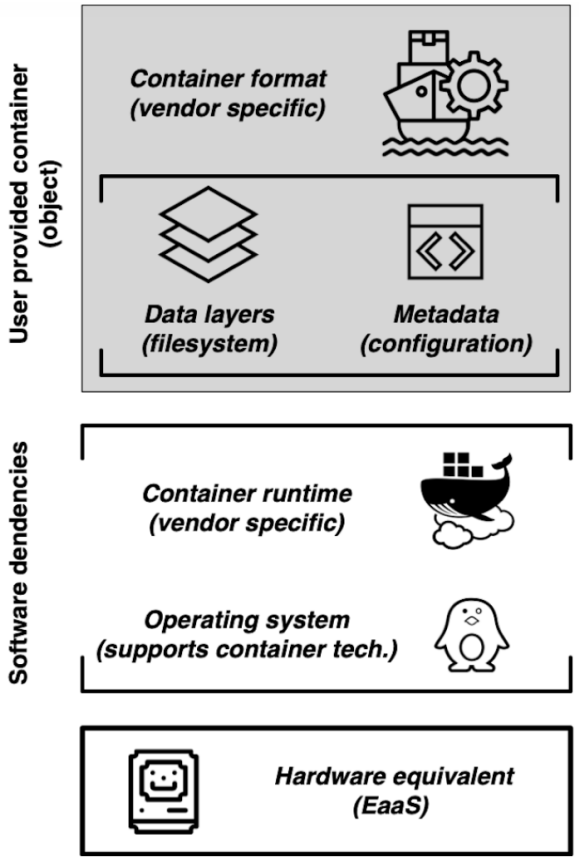
```yaml
inputs:
  map_otu_table: File
  map_query: File
  map_label: string
  return_dirname: string

outputs:
  out_dir:
    type: Directory?
    outputSource: return_output_dir/out

steps:

  mapseq2biom:
    run: ../mapseq2biom/mapseq2biom.cwl
    in:
      otu_table: map_otu_table
      label: map_label
      query: map_query
    out: [ otu_tsv, otu_txt, otu_tsv_notaxid ]

  counts_to_hdf5:
    run: ../biom-convert/biom-convert.cwl
    in:
      biom: mapseq2biom/otu_tsv
      hdf5: { default: true }
      table_type: { default: 'OTU table' }
    out: [ result ]
```

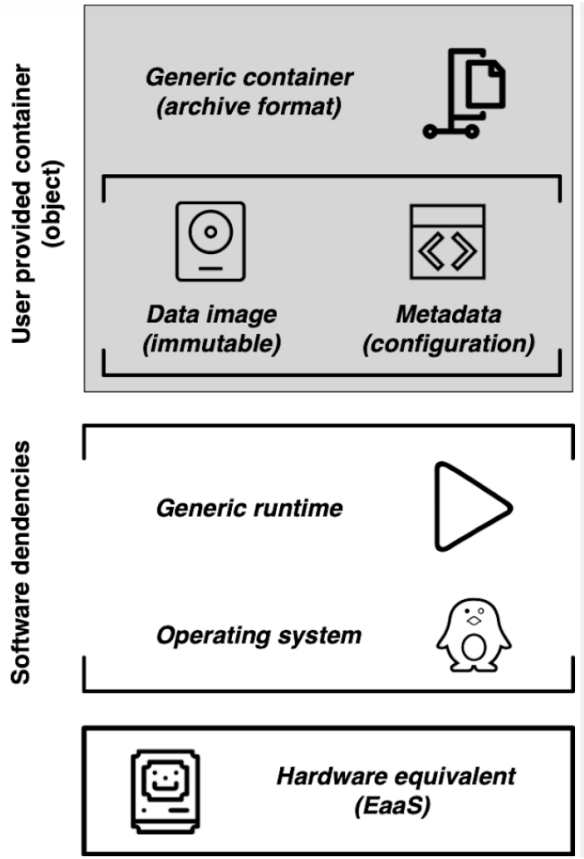
Workflow preservation in four steps

1. Preserve container-based tools: ?
2. Use preserved tools without user interface: ?
3. Use preserved tools in workflows: ?
4. Automated Integration of preserved tools: ?

Container Preservation







Workflow preservation in four steps

1. Preserve container-based tools: EaaS ✓
2. Use preserved tools without user interface: ?
3. Use preserved tools in workflows: ?
4. Automated Integration of preserved tools: ?

Environments

Virtual machines | Object Environments | Containers

Private | Public | Remote

Search...

+ Import Container

Name ↑	ID	Owner	Actions
Alpine Test	bf951b83-6574-4b...	shared	Choose action
CWL_auto_import_aeolic/mapseq2biom:latest	930e4481-d60f-4e...	shared	Choose action
CWL_auto_import_quay.io/biocontainers/biom-format:2.1.6--py36_0	6cb04e56-4bab-45...	shared	Choose action
Test Container	53bcfd27-23f2-498...	shared	Choose action

Page Size: 25

[1] to [4] of [4] Page [1] of [1]

Input/Output folders

Input folder

/input

Output folder

/output

Processes and Variables

Environment variables

PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

YOURVAR=value

Process

echo

"HELLO PV2023"

cmdl / args

Container Runtime

Networking

☐ **Enable networking**

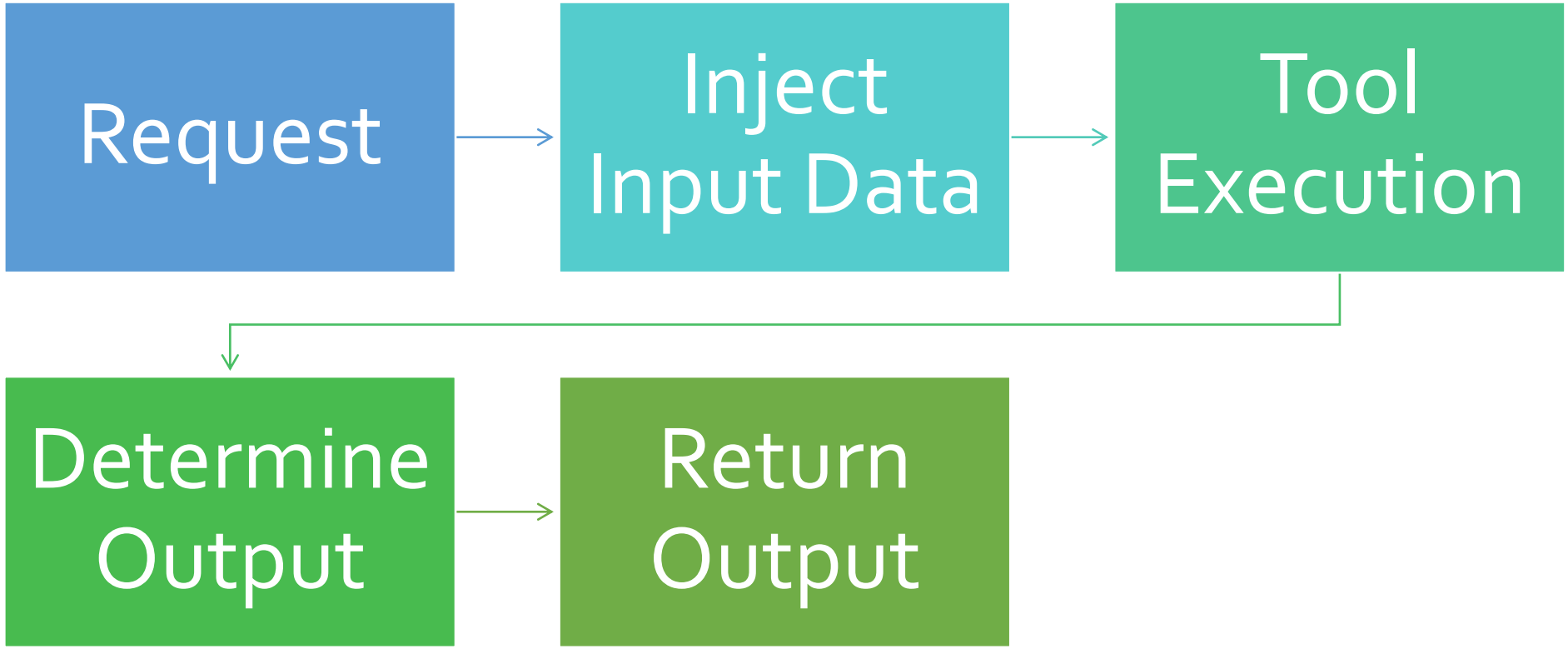
Cancel Save

```
SeaBIOS (version rel-1.12.0-0-ga698c8995f-prebuilt.qemu.org)


iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+1FF914A0+1FEF14A0 C980



Booting from Hard Disk...
GRUB loading...
Welcome to GRUB!
```

```
[    0.000000] percpu: Embedded 52 pages/cpu s172632 r8192 d32168 u2097152
[    0.000000] Built 1 zonelists, mobility grouping on.  Total pages: 128873
[    0.000000] Policy zone: DMA32
[    0.000000] Kernel command line: BOOT_IMAGE=/boot/bzImage root=/dev/sda1 root
wait console=tty1 console=ttyS0
[    0.000000] Dentry cache hash table entries: 65536 (order: 7, 524288 bytes, l
inear)
[    0.000000] Inode-cache hash table entries: 32768 (order: 6, 262144 bytes, li
near)
[    0.000000] mem auto-init: stack:off, heap alloc:off, heap free:off
[    0.000000] Memory: 485728K/523768K available (14340K kernel code, 1559K rwda
ta, 3424K rodata, 1172K init, 976K bss, 38040K reserved, 0K cma-reserved)
[    0.000000] SLUB: HWalign=64, Order=0-3, MinObjects=0, CPUs=1, Nodes=1
[    0.000000] rcu: Hierarchical RCU implementation.
[    0.000000] rcu: ∘RCU event tracing is enabled.
[    0.000000] rcu: ∘RCU restricting CPUs from NR_CPUS=64 to nr_cpu_ids=1.
[    0.000000] rcu: RCU calculated value of scheduler-enlistment delay is 100 ji
ffies.
[    0.000000] rcu: Adjusting geometry for rcu_fanout_leaf=16, nr_cpu_ids=1
[    0.000000] NR_IRQS: 4352, nr_irqs: 256, preallocated irqs: 16
[    0.000000] random: get_random_bytes called from start_kernel+0x357/0x522 wit
h crng_init=0
[    0.000000] Console: colour VGA+ 80x25
[    0.000000] printk: console [tty1] enabled
```

```
==> metadata.json <==
{"dhcp":false,"telnet":true,"process":"/bin/sh","args":["-c","mkdir container-ou
tput && emucon-cgen --enable-extensive-caps --disable-network-namespace \"$@\";
runc run eaas-job | tee container-output/container-log-fa13c4e5-73dd-4687-9a03-f
e99c4333c2e.log","","--output","config.json","--mount","container-output:/output
:bind:rw","--env","PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin
/bin","--","echo","\"HELLO PV2023\""]}false
true
RUNNING: '/bin/sh' '-c' 'mkdir container-output && emucon-cgen --enable-extensiv
e-caps --disable-network-namespace "$@"; runc run eaas-job | tee container-outpu
t/container-log-fa13c4e5-73dd-4687-9a03-fe99c4333c2e.log' '' '--output' 'config
json' '--mount' 'container-output:/output:bind:rw' '--env' 'PATH=/usr/local/sbin
:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin' '--' 'echo' '"HELLO PV2023"'
Removing network-namespace...
Adding bind-mounts...
Disable seccomp support...
Adding masked paths...
Adding readonly paths...
Adding container's command...
Running config generator...
"HELLO PV2023"
EXIT STATUS: 0
[    4.607976] EXT4-fs (sdc): re-mounted. Opts: (null)
[    4.632420] EXT4-fs (sda1): re-mounted. Opts: (null)
```

Workflow API: Preserved Tool Execution



Workflow preservation in four steps

1. Preserve container-based tools: EaaS ✓
2. Use preserved tools without user interface: Workflow API ✓
3. Use preserved tools in workflows: ?
4. Automated Integration of preserved tools: ?

CWL Wrapper

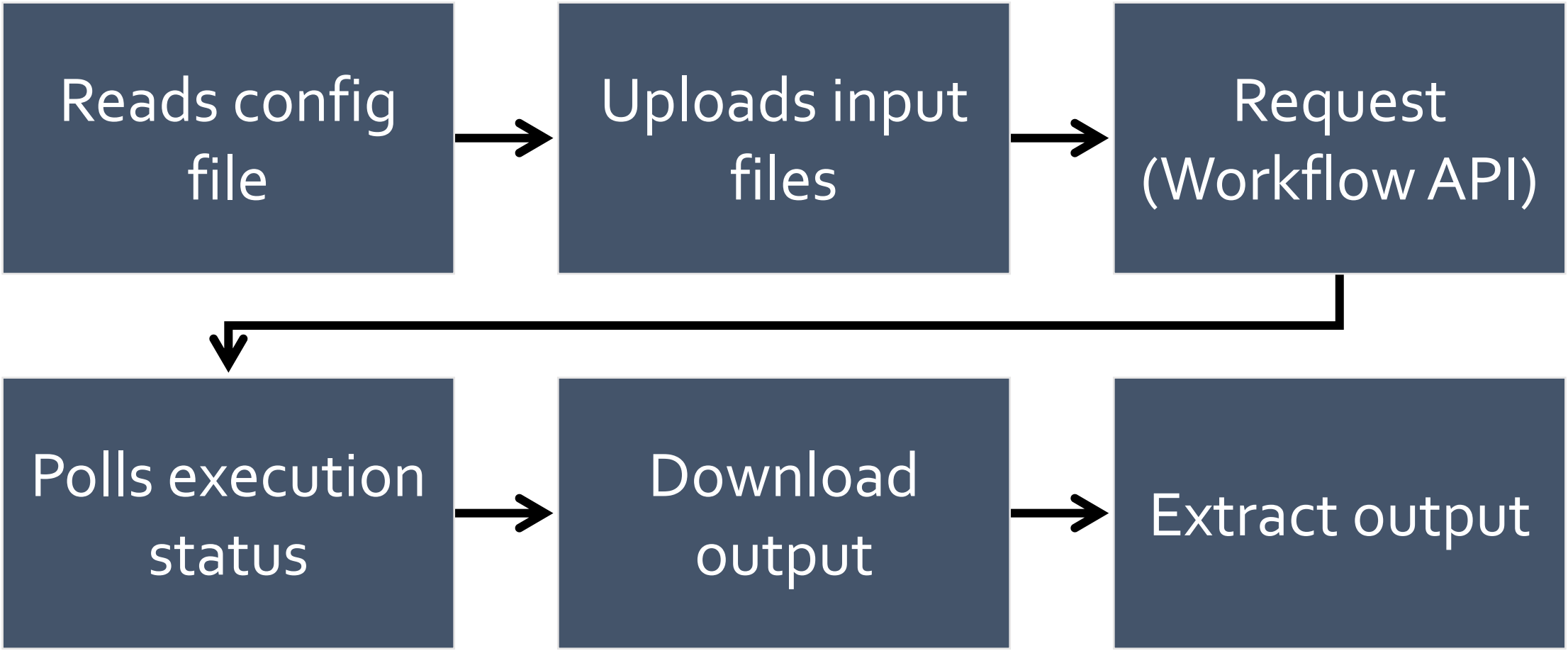
How can we use the Workflow API in actual workflows?

„Wrapper“ Container-Image

Replaces original tool while providing same functionality

Communicates with Workflow API

CWL Wrapper


Reads config file
Uploads input files
Request (Workflow API)
Polls execution status
Download output
Extract output

Accessing the Wrapper in CWL

```yaml
class: CommandLineTool
baseCommand: [myTool.sh]
requirements:
  DockerRequirement:
    dockerPull: myContainer:1.0
inputs:
  inputFile:
    type: File
    inputBinding:
      prefix: --file
```

```yaml
class: CommandLineTool
baseCommand: [python3, /app/wrapper.py,
                myTool.sh]
requirements:
 DockerRequirement:
  dockerPull: .../cwl-wrapper:latest
  dockerOutputDirectory: /app/output
inputs:
 inputFile:
  type: File
  inputBinding:
   prefix: --file
InitialWorkDirRequirement:
listing:
 - entryname: config.json
   entry: |-
     {
      "environmentId": "5fe4bd..."
     }
```

Workflow preservation in four steps

1. Preserve container-based tools: EaaS ✓
2. Use preserved tools without user interface: Workflow API ✓
3. Use preserved tools in workflows: CWL Wrapper ✓
4. Automated Integration of preserved tools: ?

CWL Rewriter

AUTOMATED REWRITING OF CWL FILES

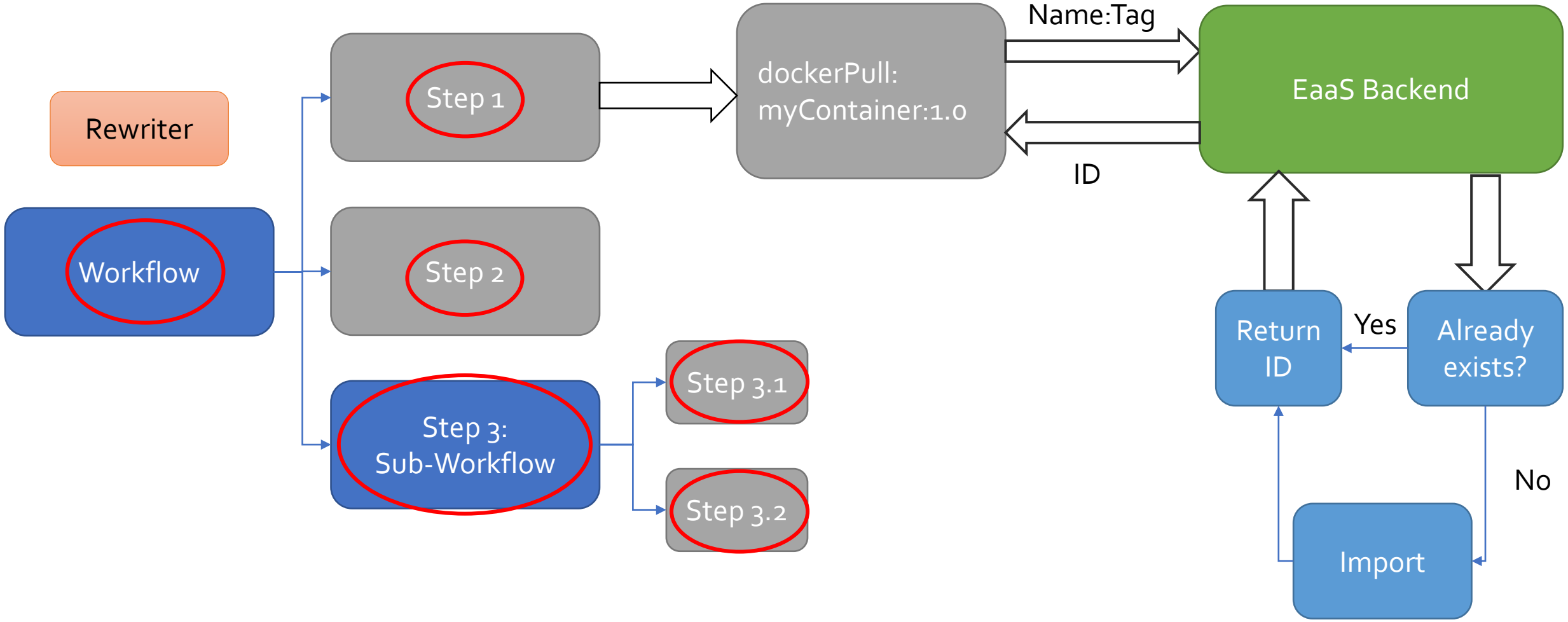
AUTOMATICALLY IMPORTS IMAGES TO EAAS

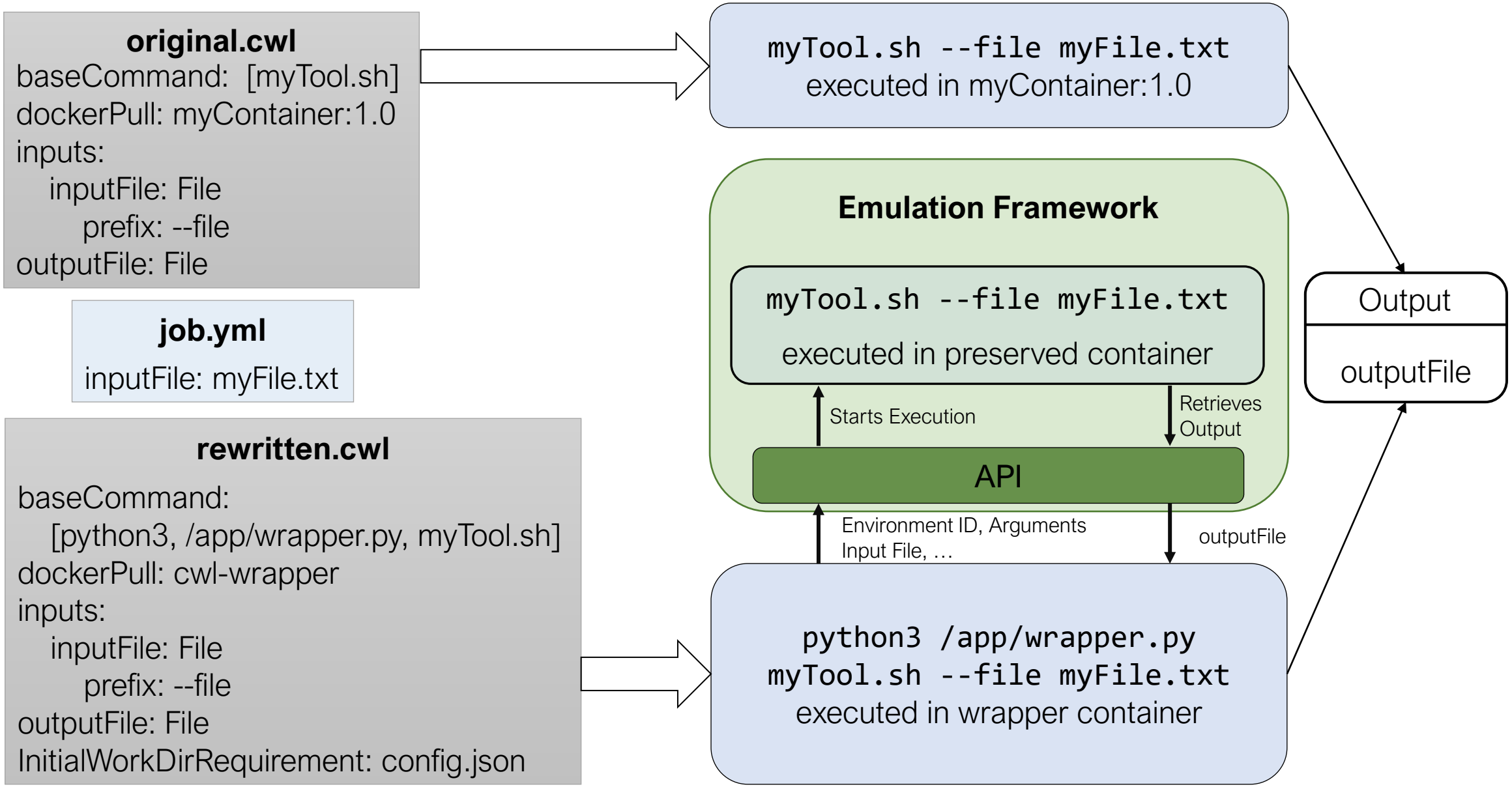
RECURSIVE

ONLY HANDLES CWL FILES WITH *DOCKERPULL*

INTEGRATED IN EAAS

CWL Rewriter




original.cwl
baseCommand: [myTool.sh]
dockerPull: myContainer:1.0
inputs:
inputFile: File
prefix: --file
outputFile: File
job.yml
inputFile: myFile.txt
rewritten.cwl
baseCommand:
[python3, /app/wrapper.py, myTool.sh]
dockerPull: cwl-wrapper
inputs:
inputFile: File
prefix: --file
outputFile: File
InitialWorkDirRequirement: config.json
myTool.sh --file myFile.txt
executed in myContainer:1.0
Emulation Framework
myTool.sh --file myFile.txt
executed in preserved container
Starts Execution
Retrieves Output
API
Environment ID, Arguments
Input File, …
outputFile
python3 /app/wrapper.py
myTool.sh --file myFile.txt
executed in wrapper container
Output
outputFile

Workflow preservation in four steps

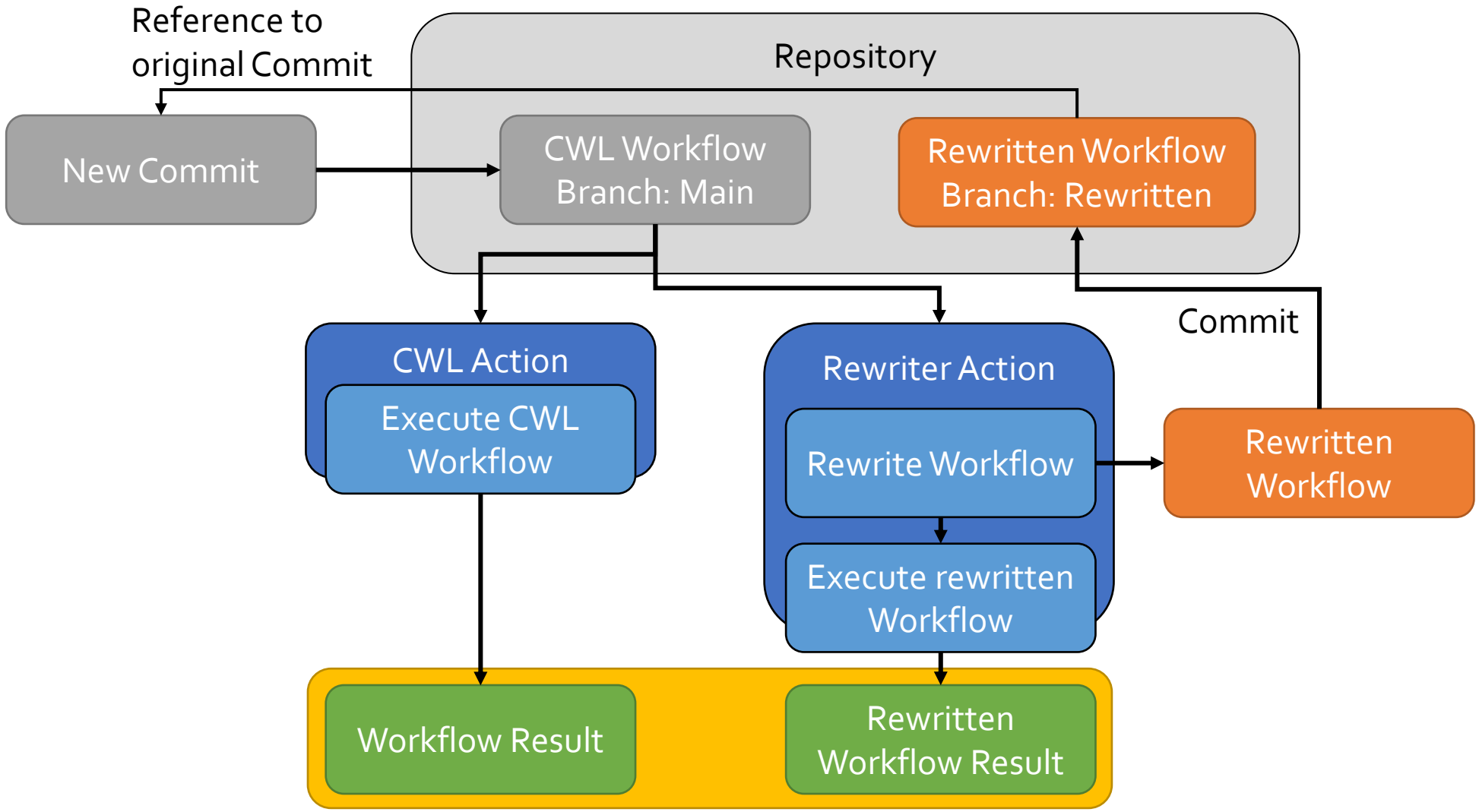
1. Preserve container-based tools: EaaS ✓
2. Use preserved tools without user interface: Workflow API ✓
3. Use preserved tools in workflows: CWL Wrapper ✓
4. Automated Integration of preserved tools: CWL Rewriter ✓

Towards *Continuous Preservation*

Manually using the rewriter? Second workflow to maintain?

- CWL Rewriter as GitHub Action
- Automatically create rewritten workflow on commit
- Run sample workflow on commit

Towards *Continuous Preservation*



```yaml
inputs:
  workflowPath:
    description: Path to CWL workflow to preserve in EaaS
    required: true
  eaasBackendUrl:
     description: EaaS backend base URL (usually ends with "/emil")
     required: true
  runtimeId:
    description: Container runtime ID in EaaS instance
    required: true
  repoUrl:
    description: Git URL of the original CWL Repo
    required: false
  branch:
    description: Target Branch for the Rewriter Results
    required: false

name: CWL Rewriter
description: Rewriter Action for CWL Files
runs:
  using: "composite"
  steps:
  - run: |
     ...
```

```yaml

  - run: |
      pip install -r "$GITHUB_ACTION_PATH"/requirements.txt
      "$GITHUB_ACTION_PATH"/rewriter.py --runtime-id "$runtimeId" "$workflowPath"
    shell: bash
    env:
      runtimeId: ${{inputs.runtimeId}}
      workflowPath: ${{inputs.workflowPath}}
      EMIL_BASE_URL: ${{inputs.eaasBackendUrl}}

  - run: |
      ...
      git checkout ${{inputs.branch}}
      ...
      git commit -m "Automated commit of rewritten files from branch ..."
      git push
    shell: bash
  - uses: actions/upload-artifact@v3
    with:
      name: Preserved CWL workflows for ${{inputs.workflowPath}}
      path: |
        **/wrapped_*.cwl

```

Rewriter Action

Towards *Continuous Preservation*

```yaml
on:
  push:
    branches: [ main ]

jobs:
  preserve:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: emulation-as-a-service/cwl-rewriter@main
        with:
          workflowPath: example_workflow.cwl
          eaasBackendUrl: https://c6564661-6070-42cc-b6e0-ad1277a1ca7e.fr.bw-cloud-instance.org/emil
          runtimeId: 2f49bdda-3f9d-47c6-84f3-611646b86828
      - uses: emulation-as-a-service/cwl-action@main
        with:
          workflowPath: wrapped_workflow_example_workflow.cwl
          jobFilePath: workflow-test.yml
```

Towards *Continuous Preservation*

rewritten

Commits on Apr 12, 2023

Automated commit of rewritten files from branch workflow-only, commit... ... github-actions[bot] committed 2 weeks ago	8a5e81b
Automated commit of rewritten files from branch workflow-only, commit... ... github-actions[bot] committed 2 weeks ago	8281ddb
Automated commit of rewritten files from branch workflow-only, commit... ... github-actions[bot] committed 2 weeks ago	10bc65d

Code Blame 3 lines (3 loc) · 123 Bytes

```
Original Branch: workflow-only
Commit SHA: 880196f9e2517bd50063149e4e145a3b38423d6b
Triggered:Wed Apr 12 10:32:33 UTC 2023
```

Thanks for you attention!