IRIS-HEP final presentation

Snakemake backend for RECAST

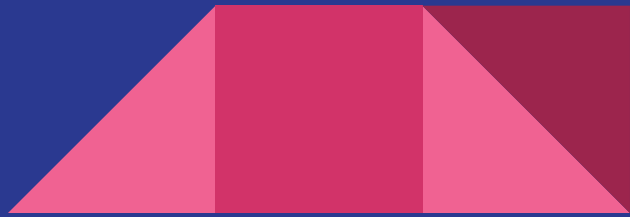
Andrii Povsten
Mentored by Matthew Feickert (University Wisconsin-Madison)
Lukas Heinrich (Technical University of Munich)
GitHub repository: AndriiPovsten/Snakemake-backend-for-RECAST

Motivation

Reproducibility ensures the integrity of your research. When your analysis is reproducible, it's easier for others to validate your findings and build upon your work.

Reproducible workflows save time. You won't need to redo your analysis from scratch every time you make a change or need to re-run an experiment. This efficiency allows you to focus on advancing your research instead of repetitive tasks.

Your work becomes a valuable resource. Reproducible research can be referenced, reused, and built upon for years to come, contributing to the collective knowledge of physics.



RECAST is a framework for extending the impact of existing analyses performed by high-energy physics experiments. This framework allows scientists to access and reuse workflows that have already been performed.

What is the workflow languages?

Workflow language is a way to describe and organize a series of tasks or steps that need to be performed in a logical order.

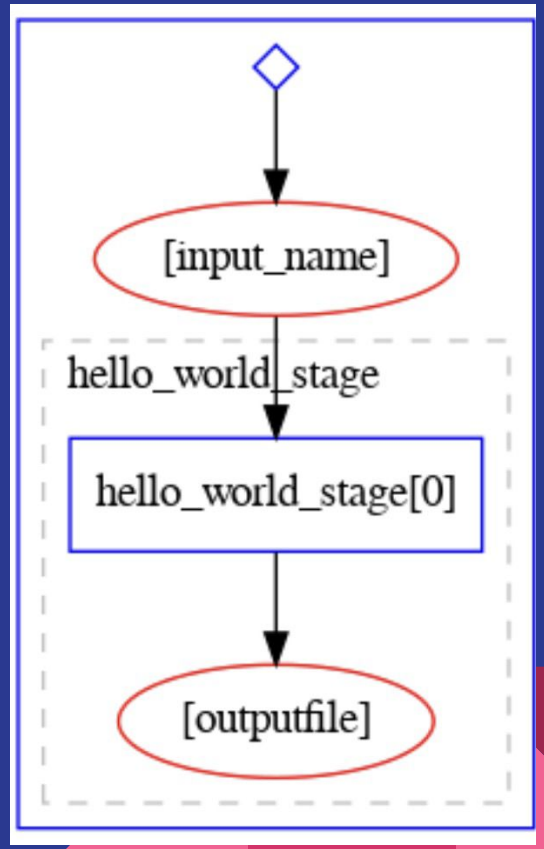
Help define what tasks to do, when to do them, and how they depend on each other.

These languages make it easier for people or computer systems to understand and execute a sequence of actions efficiently.

Yadage is current RECAST workflow language

Example Workflow

```
cat << 'EOF' > workflow.yml
stages:
- name: hello_world
  dependencies: [init]
  scheduler:
    scheduler_type: singlestep-stage
    parameters:
      name: {step: init, output: name}
      outputfile: '{workdir}/hello_world.txt'
    step:
      process:
        process_type: 'string-interpolated-cmd'
        cmd: 'echo Hello my Name is {name} | tee {outputfile}'
      publisher:
        publisher_type: 'frompar-pub'
        outputmap:
          outputfile: outputfile
      environment:
        environment_type: 'docker-encapsulated'
        image: busybox
EOF
```



Motivation of using Snakemake

Recreating the same analysis pipeline makes work easily shareable.

Widely used and robustly supported through contributions from all communities.

Python based workflow management system.

Currently supports containers and seems to offer a better supported alternative to Yadage.

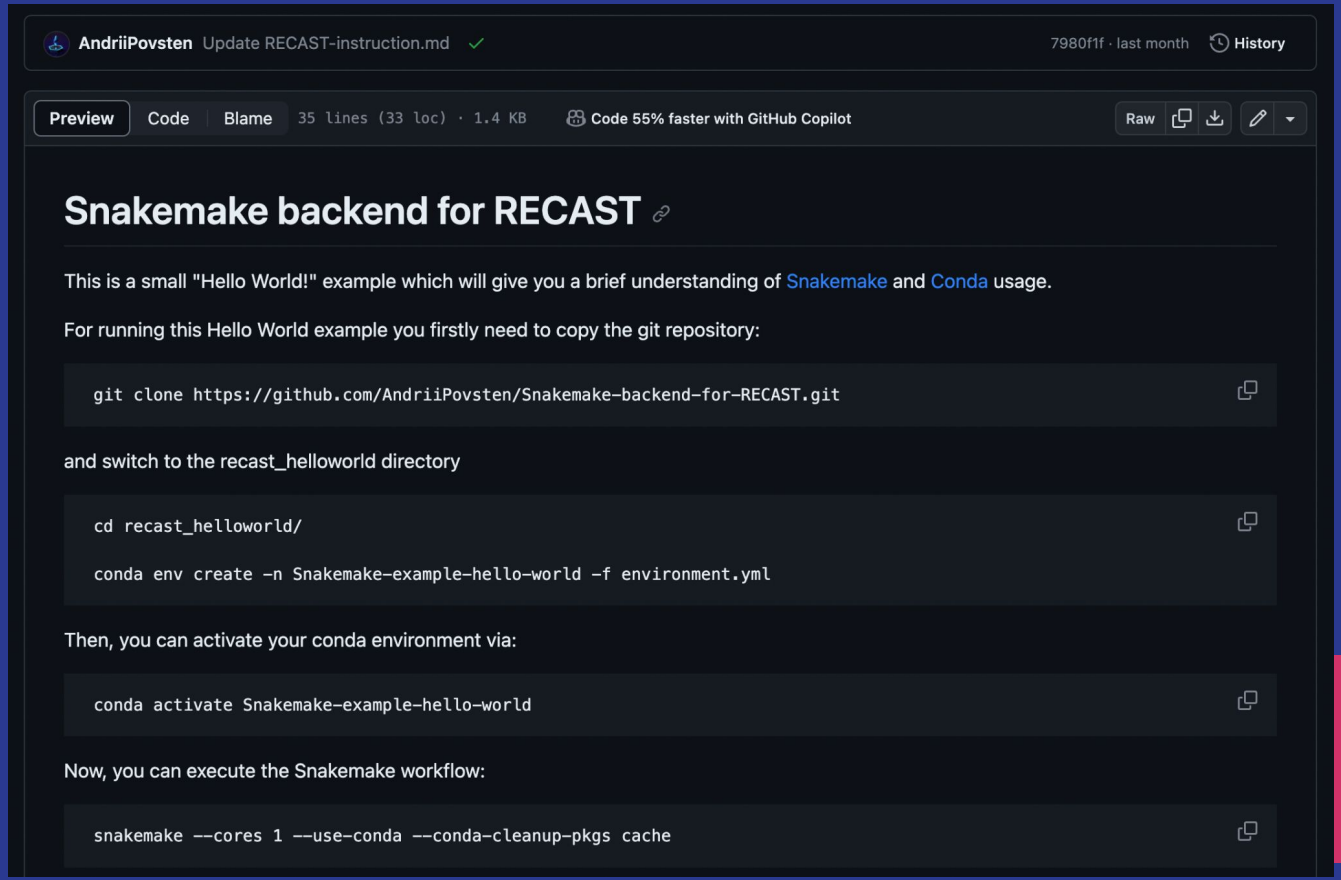
Environment file containing the conda virtual environment, which provides all the necessary packages to run the workflow.

environment.yml	Adding conda environments to Snakefile	last month
reana-environment.yml	Instructions for using REANA-client + singularity	last month

```yaml
name: Snakemake-example-hello-world
channels:
  - conda-forge
dependencies:
  - python=3.11
  - bioconda::snakemake=7.25.0
  - pip
  - pytest
  - pandas
  - samtools
  - matplotlib
  - ruff
```

```yaml
name: Snakemake-reana-helloworld-example
channels:
  - conda-forge
dependencies:
  - python=3.10
  - bioconda::snakemake=7.25.0
  - pip
  - pytest
  - pandas
  - samtools
  - matplotlib
  - ruff
  - singularity
  - pip:
    - reana-client
```

Instructions on how to run Snakemake:



AndriiPovsten Update RECAST-instruction.md ✓ 7980f1f · last month History

Preview | Code | Blame | 35 lines (33 loc) · 1.4 KB | Code 55% faster with GitHub Copilot | Raw

Snakemake backend for RECAST

This is a small "Hello World!" example which will give you a brief understanding of Snakemake and Conda usage.

For running this Hello World example you firstly need to copy the git repository:

```
git clone https://github.com/AndriiPovsten/Snakemake-backend-for-RECAST.git
```

and switch to the recast_helloworld directory

```
cd recast_helloworld/

conda env create -n Snakemake-example-hello-world -f environment.yml
```

Then, you can activate your conda environment via:

```
conda activate Snakemake-example-hello-world
```

Now, you can execute the Snakemake workflow:

```
snakemake --cores 1 --use-conda --conda-cleanup-pkgs cache
```

RECAST “Hello World” example implementing using the Snakemake

recast_helloworld > Snakefile

```
configfile: "steps.yml"
rule all:
    input:
        "results/helloworld_output.txt"  #Final output

rule prepare_data:
    output:
        "data/input.txt"
    conda:
        "environment.yml"
    shell:
        """
        mkdir -p data
        echo "Hello World!" > {output}
        """
```

```
rule run_example:
    input:
        "data/input.txt"
    conda:
        "environment.yml"
    output:
        "results/example_output.txt"
    shell:
        """
        mkdir -p results
        cat {input} > {output}
        """

rule output_results:
    input:
        "results/example_output.txt"
    output:
        "results/helloworld_output.txt"
    conda:
        "environment.yml"
    shell:
        """
        cp {input} {output}
        """
```

Log files

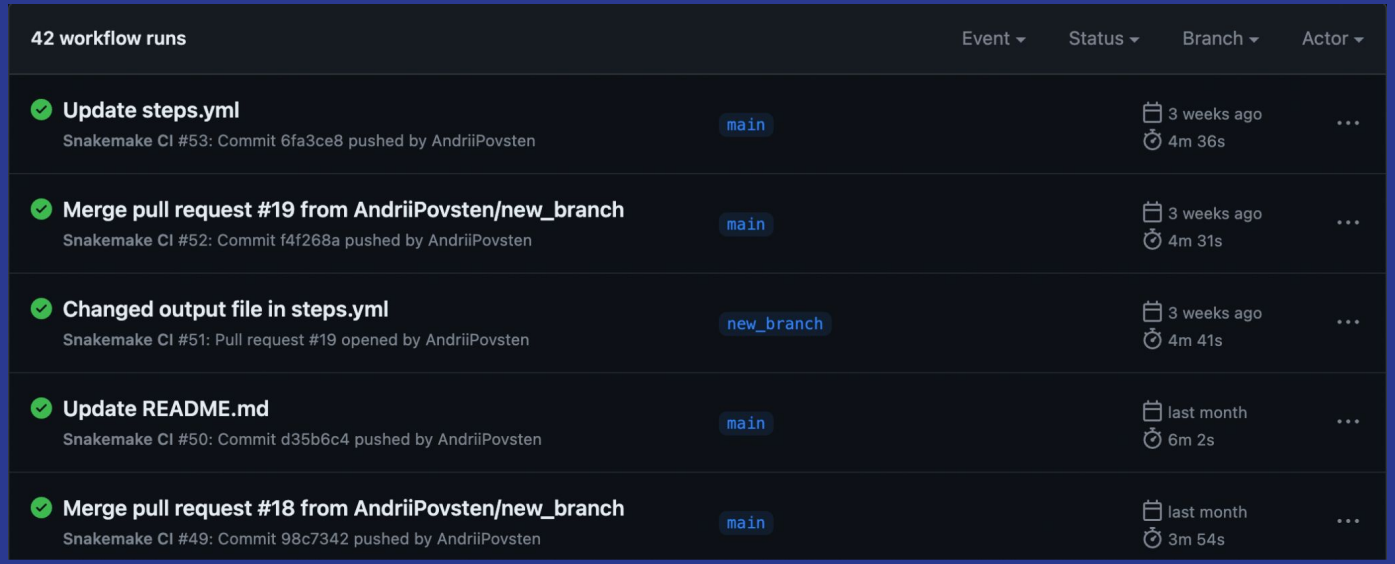
.snakemake/log/2023-09-07T151352.075031.snakemake.log

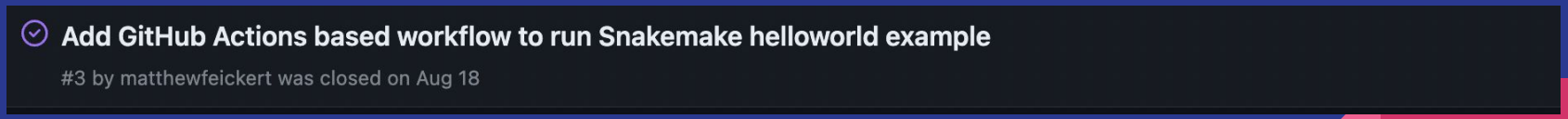
```
Building DAG of jobs...
Using shell: /usr/bin/bash
Provided cluster nodes: 300
Conda environments: ignored
Singularity containers: ignored
Job stats:
job               count    min threads    max threads
--------------  -------  -------------  -------------
all                   1              1              1
output_results        1              1              1
run_example           1              1              1
total                 3              1              1

Select jobs to execute...
```

```
[Thu Sep  7 15:13:52 2023]
rule run_example:
    input: data/input.yaml
    output: results/example_output.txt
    log: logs/run_example_output.log
    jobid: 2
    resources: tmpdir=/tmp

[Thu Sep  7 15:14:02 2023]
Finished job 2.
1 of 3 steps (33%) done
Select jobs to execute...

[Thu Sep  7 15:14:02 2023]
rule output_results:
    input: results/example_output.txt
    output: results/helloworld_output.txt
    log: logs/output_results_logging.log
    jobid: 1
    resources: tmpdir=/tmp
```

```
[Thu Sep  7 15:14:12 2023]
Finished job 1.
2 of 3 steps (67%) done
Select jobs to execute...

[Thu Sep  7 15:14:12 2023]
localrule all:
    input: results/helloworld_output.txt
    jobid: 0
    resources: tmpdir=/tmp

[Thu Sep  7 15:14:12 2023]
Finished job 0.
3 of 3 steps (100%) done
Complete log: /var/reana/users/361c88b1-9389-4164-8c71-f1d55a304701/workflows/8e925c76-4ab4-4403-8c08-0
Building DAG of jobs...
Creating report...
Downloading resources and rendering HTML.
Report created: report.html.
```

GitHub


42 workflow runs
Event
Status
Branch
Actor
Update steps.yml
Snakemake CI #53: Commit 6fa3ce8 pushed by AndriiPovsten
main
3 weeks ago
4m 36s
Merge pull request #19 from AndriiPovsten/new_branch
Snakemake CI #52: Commit f4f268a pushed by AndriiPovsten
main
3 weeks ago
4m 31s
Changed output file in steps.yml
Snakemake CI #51: Pull request #19 opened by AndriiPovsten
new_branch
3 weeks ago
4m 41s
Update README.md
Snakemake CI #50: Commit d35b6c4 pushed by AndriiPovsten
main
last month
6m 2s
Merge pull request #18 from AndriiPovsten/new_branch
Snakemake CI #49: Commit 98c7342 pushed by AndriiPovsten
main
last month
3m 54s


Add GitHub Actions based workflow to run Snakemake helloworld example
#3 by matthewfeickert was closed on Aug 18

CI using GitHub actions

```yaml
- name: Testing
  uses: snakemake/snakemake-github-action@v1
  with:
    directory: 'recast_helloworld'
    snakefile: 'recast_helloworld/Snakefile'
    args: '--cores 1 --use-conda --conda-cleanup-pkgs cache'

- name: Create container file
  uses: snakemake/snakemake-github-action@v1
  with:
    snakefile: 'recast_helloworld/Snakefile'
    task: 'containerize'
    directory: 'recast_helloworld'
```

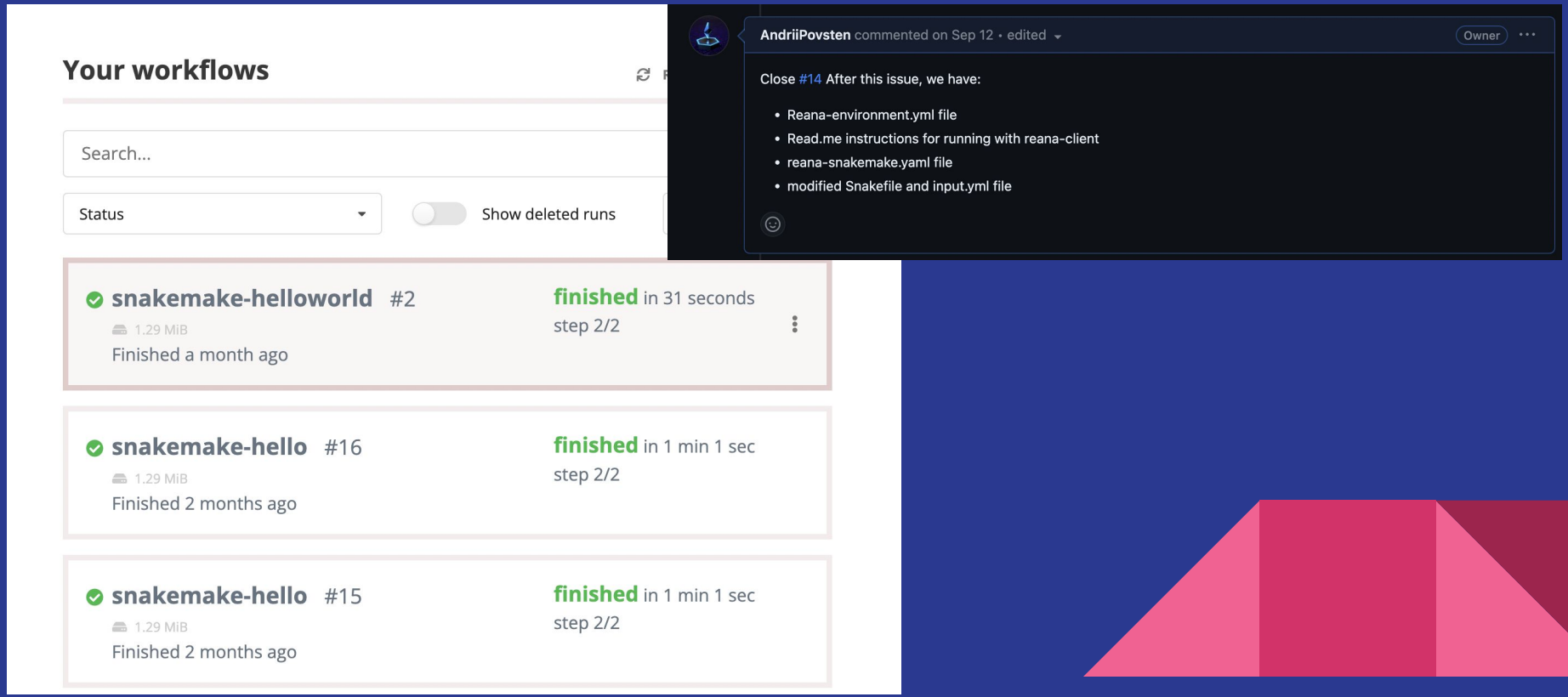
Creates Snakemake logs/linting Snakefile

```yaml
- name: Create container file
  uses: snakemake/snakemake-github-action@v1
  with:
    snakefile: 'recast_helloworld/Snakefile'
    task: 'containerize'
    directory: 'recast_helloworld'
```

Create Dockerfile

```yaml
- name: Upload a Dockerfile
  uses: actions/upload-artifact@v3
  with:
    name: Dockerfile
    path: Dockerfile

- name: Linting
  uses: snakemake/snakemake-github-action@v1
  with:
    directory: 'recast_helloworld'
    snakefile: 'reana_helloworld/Snakefile'
    args: '--lint'
```

Implementation of RECAST based with REANA

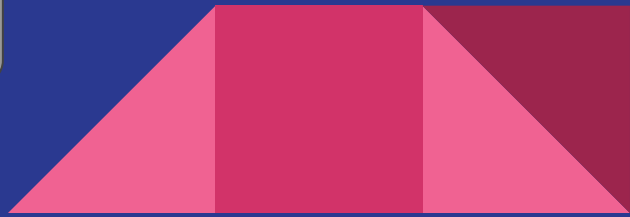


Conclusions:

The Snakemake is a simple to use/debug workflow language

Provides the conda environments and containerisation

Snakemake can be implemented for RECAST for more complex analyses.



Thank you for your attention!