Snakemake Done Three Ways

HTCondor European Workshop 2025

Brian Bockelman

MORGRIDGE
INSTITUTE FOR RESEARCH

Often Overheard:

- I think __________ is the best workflow language out there, why doesn’t HTCondor support it?
  - Common Workflow Language

Often Overheard:

- I think __________ is the best workflow language out there, why doesn't HTCondor support it?
  - ~~Common Workflow Language~~
  - Workflow Description Language

Often Overheard:

- I think __________ is the best workflow language out there, why doesn't HTCondor support it?
  - ~~Common Workflow Language~~
  - ~~Workflow Description Language~~
  - Yet Another Workflow Language

Often Overheard:

- I think __________ is the best workflow language out there, why doesn't HTCondor support it?
    - ~~Common Workflow Language~~
    - ~~Workflow Description Language~~
    - ~~Yet Another Workflow Language~~
    - Apache Airivata

Often Overheard:

- I think __________ is the best workflow language out there, why doesn't HTCondor support it?
  - ~~Common Workflow Language~~
  - ~~Workflow Description Language~~
  - ~~Yet Another Workflow Language~~
  - ~~Apache Airivata~~
  - Kepler

Often Overheard:

- I think __________ is the best workflow language out there, why doesn’t HTCondor support it?
  - ~~Common Workflow Language~~
  - ~~Workflow Description Language~~
  - ~~Yet Another Workflow Language~~
  - ~~Apache Airivata~~
  - ~~Kepler~~
  - Swift

Often Overheard:

- I think __________ is the best workflow language out there, why doesn't HTCondor support it?
  - ~~Common Workflow Language~~
  - ~~Workflow Description Language~~
  - ~~Yet Another Workflow Language~~
  - ~~Apache Airivata~~
  - ~~Kepler~~
  - ~~Swift~~
  - Business Process Execution Language

Often Overheard:

- I think __________ is the best workflow language out there, why doesn't HTCondor support it?
  - ~~Common Workflow Language~~
  - ~~Workflow Description Language~~
  - ~~Yet Another Workflow Language~~
  - ~~Apache Airivata~~
  - ~~Kepler~~
  - ~~Swift~~
  - ~~Business Process Execution Language~~
  - Pegasus DAX

Often Overheard:

- I think __________ is the best workflow language out there, why doesn't HTCondor support it?
  - ~~Common Workflow Language~~
  - ~~Workflow Description Language~~
  - ~~Yet Another Workflow Language~~
  - ~~Apache Airivata~~
  - ~~Kepler~~
  - ~~Swift~~
  - ~~Business Process Execution Language~~
  - ~~Pegasus DAX~~
  - Galaxy

Often Overheard:

- I think __________ is the best workflow language out there, why doesn't HTCondor support it?
    - ~~Common Workflow Language~~
    - ~~Workflow Description Language~~
    - ~~Yet Another Workflow Language~~
    - ~~Apache Airivata~~
    - ~~Kepler~~
    - ~~Swift~~
    - ~~Business Process Execution Language~~
    - ~~Pegasus DAX~~
    - ~~Galaxy~~
    - Nextflow

Often Overheard:

- I think _________ is the best workflow language out there, why doesn't HTCondor support it?
  - ~~Common Workflow Language~~
  - ~~Workflow Description Language~~
  - ~~Yet Another Workflow Language~~
  - ~~Apache Airivata~~
  - ~~Kepler~~
  - ~~Swift~~
  - ~~Business Process Execution Language~~
  - ~~Pegasus DAX~~
  - ~~Galaxy~~
  - ~~Nextflow~~
  - SnakeMake

Often Overheard:

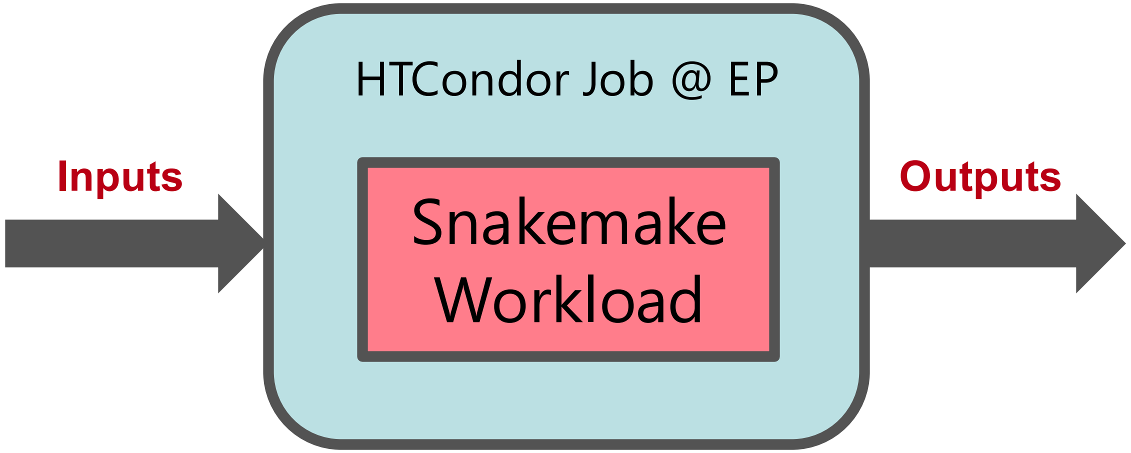
- I think __________ is the best workflow language out there, why doesn't HTCondor support it?
  - ~~Common Workflow Language~~
  - ~~Workflow Description Language~~
  - ~~Yet Another Workflow Language~~
  - ~~Apache Airivata~~
  - ~~Kepler~~
  - ~~Swift~~
  - ~~Business Process Execution Language~~
  - ~~Pegasus DAX~~
  - ~~Galaxy~~
  - ~~Nextflow~~
  - SnakeMake
- For as long as there are ~~physicists~~ biologists, there will be new workflow languages! **(arguably, this is a good thing!)**

This talk is about how CHTC approaches users that arrive with The Best workflow language.

Example: Snakemake

Phase I: “I just need to run this one pipeline”

- Never underestimate the power of running an entire pipeline as a single HTCondor job!
  - Pipeline is completely encapsulated in a single HTCondor job.
  - Pipeline has *no* knowledge of HTCondor whatsoever
- Advantages:
  - Simple! Works for nearly any workflow engine.
  - Quickest “time to science” for researchers wanting to run a single pipeline, once.
- Disadvantages:
  - Pipeline needs to fit in a single job!
  - Multiple steps in pipeline may need disparate resource requirements: the “512GB for 5 minutes” problem.



Snakemake invocation example

- For snakemake, this might involve creating a container image with a conda environment installed, the Snakefile, and a script that ultimately invokes

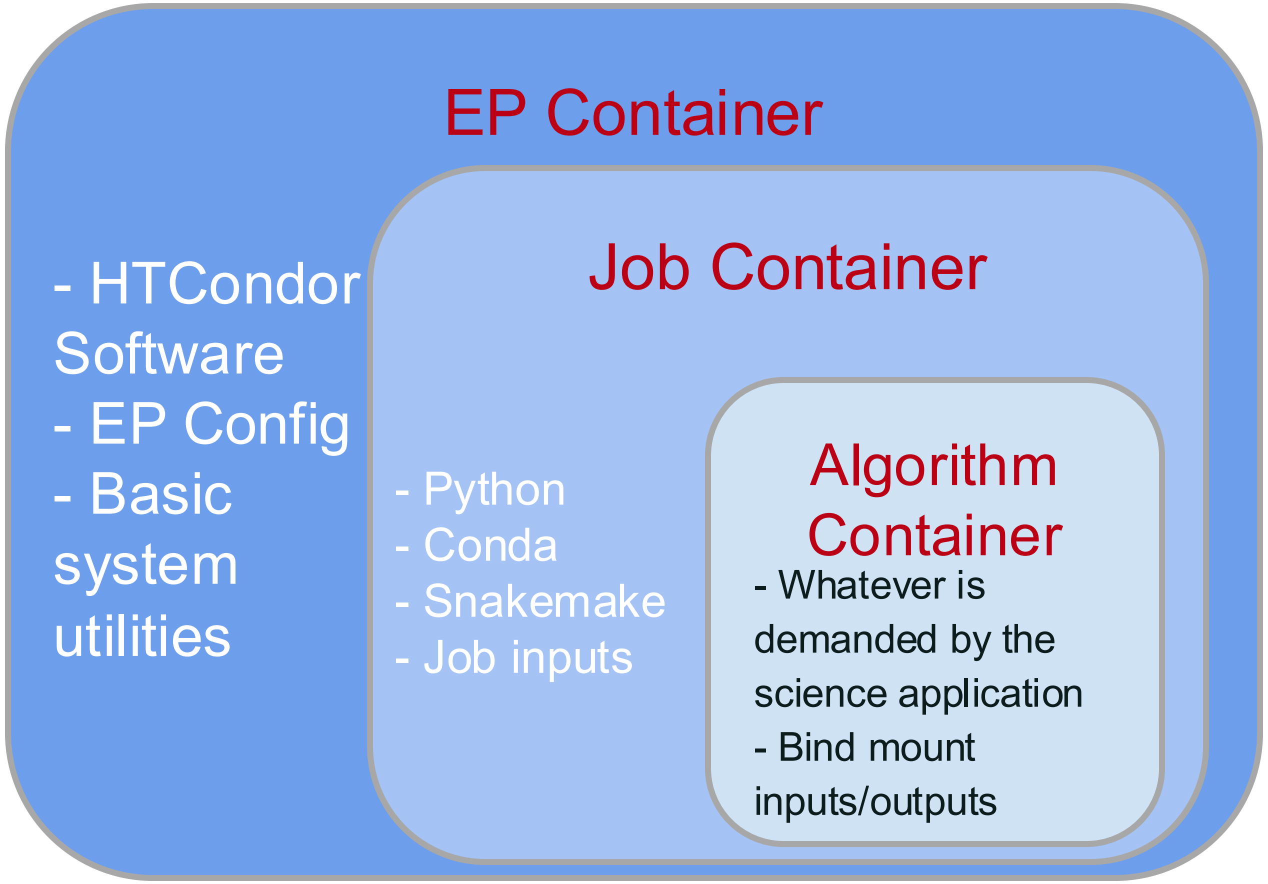
snakemake <output file>

- Snakemake will construct a DAG and, ideally, create the prescribed output file that HTCondor will send back to the AP.
- What can possibly go wrong?

Nested

Containers!

Example challenge: Nested Containers in the OSPool



EP Container

- HTCondor Software
- EP Config
- Basic system utilities

Job Container

- Python
- Conda
- Snakemake
- Job inputs

Algorithm Container

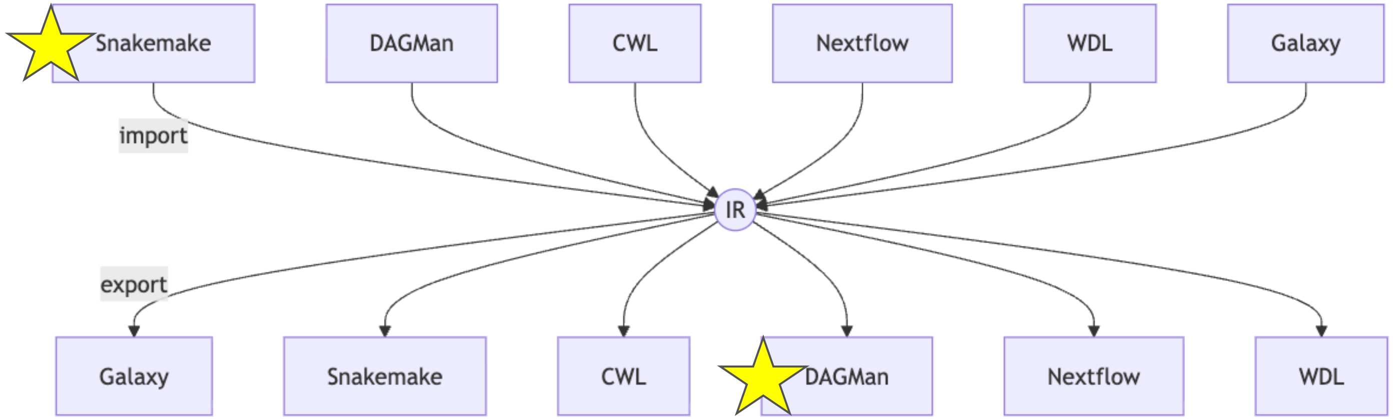
- Whatever is demanded by the science application
- Bind mount inputs/outputs

When running unprivileged (in a glidein), the Snakemake community’s frequent use of containers can cause triply-nested containers!

- Docker is not your friend, but Apptainer is!
- Users may need to convert their Docker images to an Apptainer .sif file.
  - N.b.: A very opaque, difficult environment to develop in!
- **The secret**: starting apptainer from "unpacked" .sif images (`--sandbox`)
- Fewer issues when the Snakemake workload is not itself using containers – but we don’t get to control that!

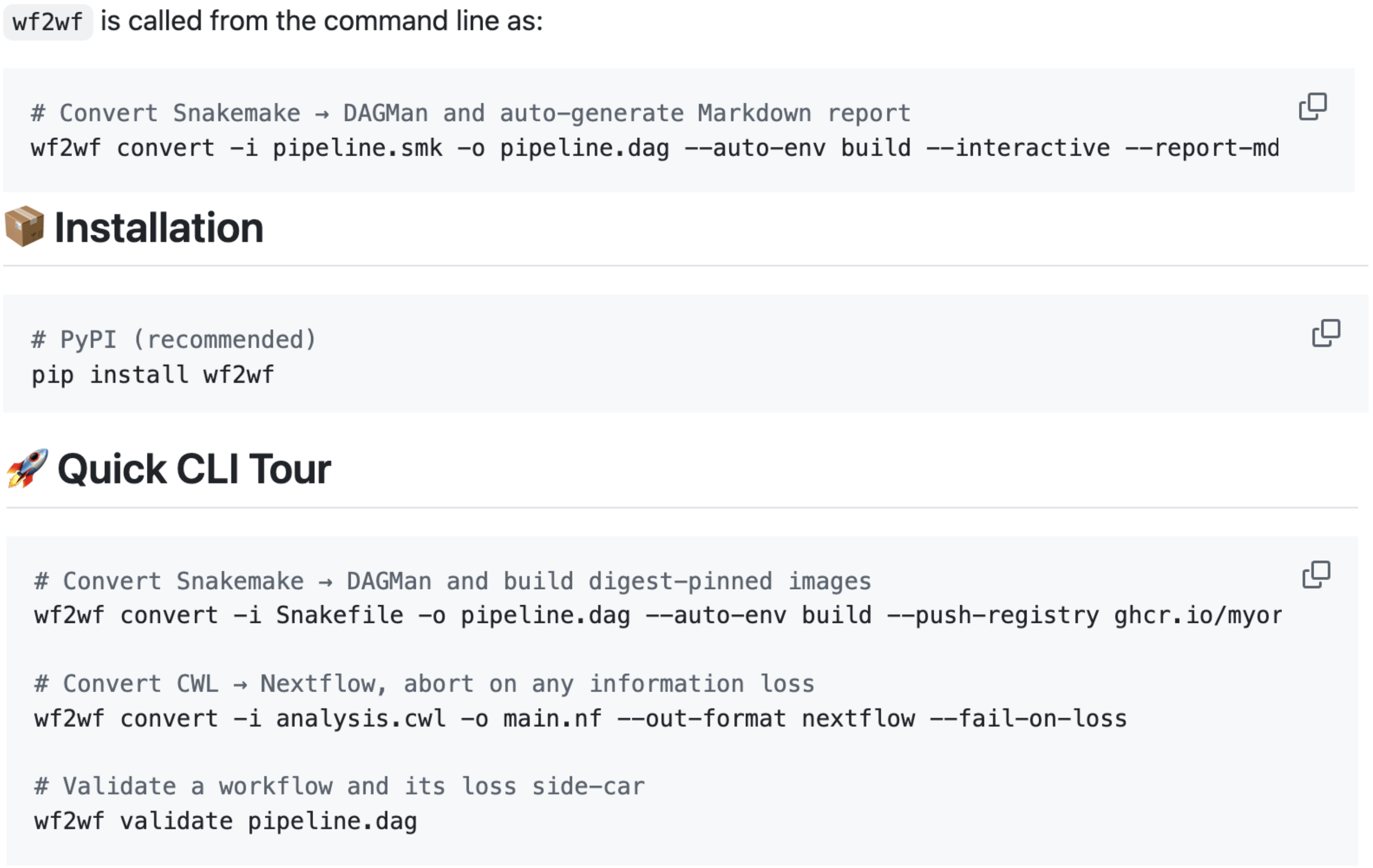
Phase II: “This postdoc joined my lab…”

- The “all-in-one” is great for the “I wanted to re-run this one workload I read in a paper”.
  - Not great for producing science of your own (what happens when reviewers ask you to re-run multiple results quickly?).
- What about the “postdoc using Snakemake joined my lab” case? Assumptions:
  - Locally, you use a native HTCondor workflow such as DAGman.
  - Postdoc used Snakemake because that’s what the former lab used, not because they are a Snakemake enthusiast.
  - Approach: Conversion!
- Case study: wf2wf utility. See https://github.com/csmcal/wf2wf
  - Goal: convert between workflow description formats



Converting between Workflow Languages

- The wf2wf utility *converts* a given workflow between formats.
- Once done, we assume you tweak/change/tinker with the destination format (DAGMan!).
- Challenges:
  - **Lossy translation**. Anyone who uses Google Translate understands that automatic translation is not perfect!
    - A <u>human</u> is needed to review the result.
  - **One-time-nature**. Good for moving into a new ecosystem.



`wf2wf` is called from the command line as:

```
# Convert Snakemake → DAGMan and auto-generate Markdown report
wf2wf convert -i pipeline.smk -o pipeline.dag --auto-env build --interactive --report-md
```

📦 Installation

```
# PyPI (recommended)
pip install wf2wf
```

🚀 Quick CLI Tour

```
# Convert Snakemake → DAGMan and build digest-pinned images
wf2wf convert -i Snakefile -o pipeline.dag --auto-env build --push-registry ghcr.io/myor

# Convert CWL → Nextflow, abort on any information loss
wf2wf convert -i analysis.cwl -o main.nf --out-format nextflow --fail-on-loss

# Validate a workflow and its loss side-car
wf2wf validate pipeline.dag
```

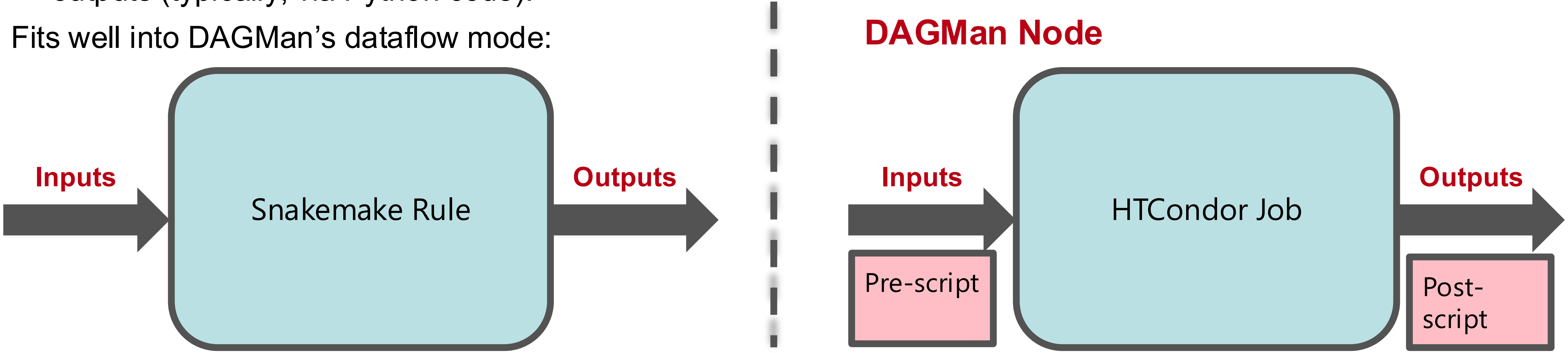
Phase III: The True Believer

- What happens when the ~~user~~ postdoc is not migrating between ecosystems but wants to work in Snakemake?
  - Valid! This may be due to expertise (“this is what I know”), passion (“I am a core Snakemake developer”), or the community (“Journal reviewers requested we publish workloads using Snakemake”).
  - We are not interested in a conversion requiring human touch-up but core HTCondor/Snakemake integration.

- Lesson learned from prior projects: HTCondor’s best analogy is *not* SLURM but AWS. That is:
  - Each workflow engine has a set of plugins for different “execution environments”.
  - The “batch system-like” plugin interface (i.e., SLURM) almost always assumes a shared filesystem.
  - HTCondor prefers to have **no** shared filesystem.
  - Thus: look for executions environments that work with AWS which rarely has a shared filesystem between worker (nodes / VMs / containers).

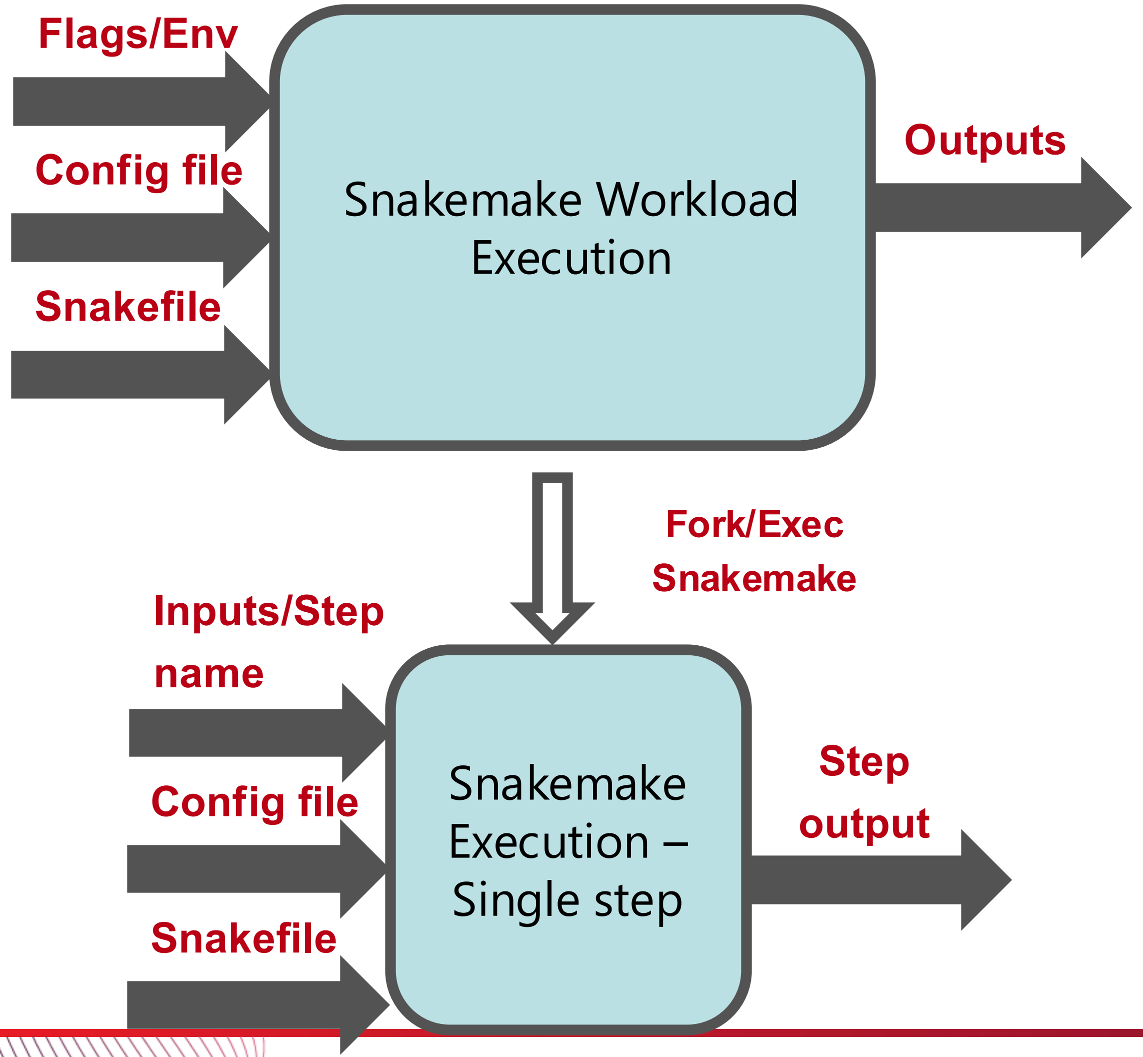
The Snakemake example

- It's all in the name: Snakemake.
  - Snake: Workflow description language is largely in Python, interspersed with YAML. Familiar to a "python native".
  - **Make**: Core concept is that of a Unix Makefile: each rule describes the **inputs**, **outputs**, and how to create the outputs (typically, via Python code).
- Fits well into DAGMan's dataflow mode:



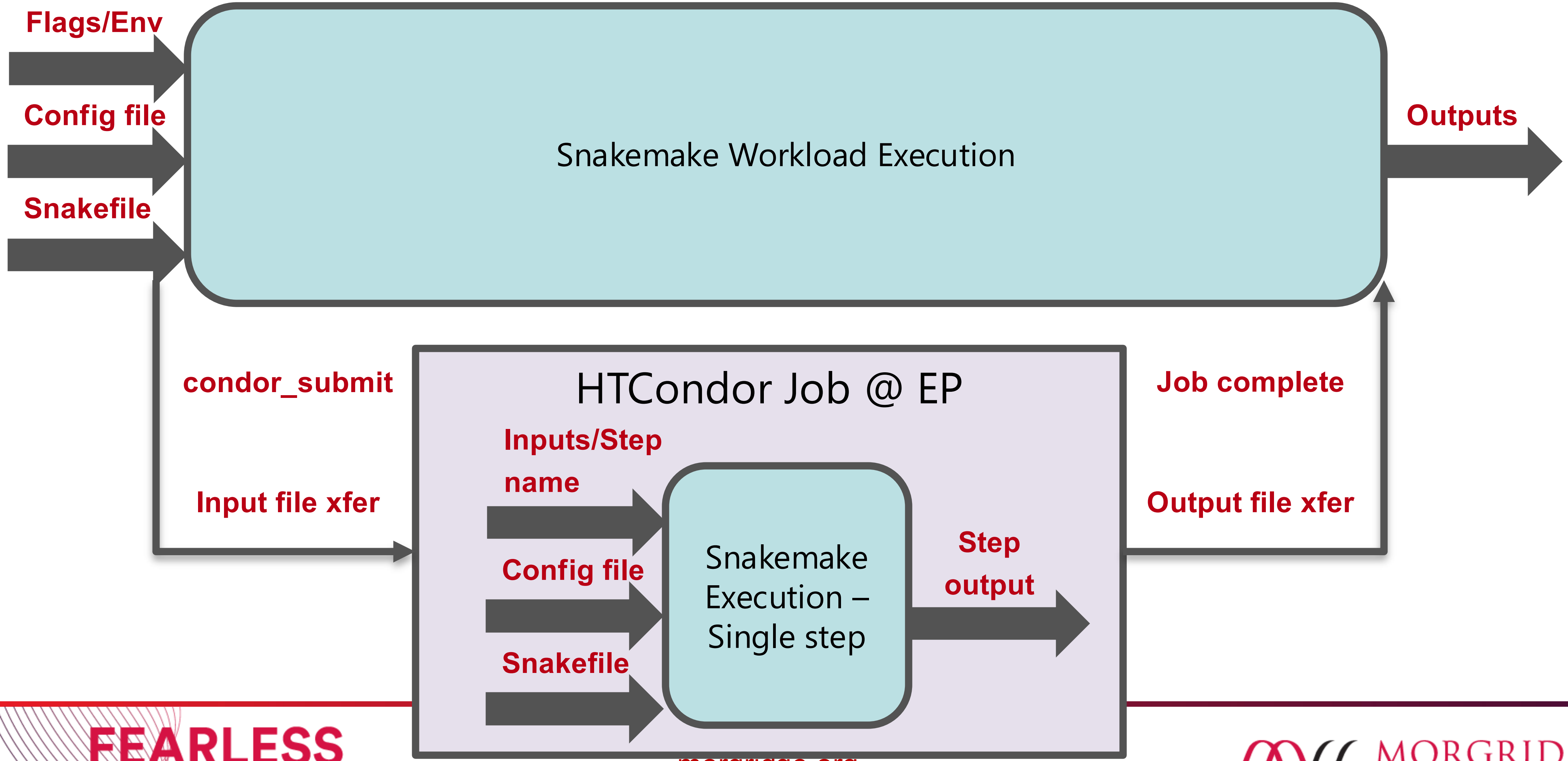
- **Note**: Same issues as any dataflow language. How do you know the "outputs" were completed?!?

A bit on Snakemake



- Snakemake will calculate the missing files and fork/exec itself to execute itself to create the missing data.
  - This "single step" mode is the same as workload mode except it works on a single rule, creating the desired output.
  - Assumptions:
    - At beginning of step execution, the input is at "the right place".
    - At end of execution, the output is at "the right place".
    - **Assumption**: This is all a shared filesystem!
- Via plugin we can replace fork/exec with any arbitrary Python code (such as submitting a HTCondor job).

The HTCondor Executor



Invoking at a technical level

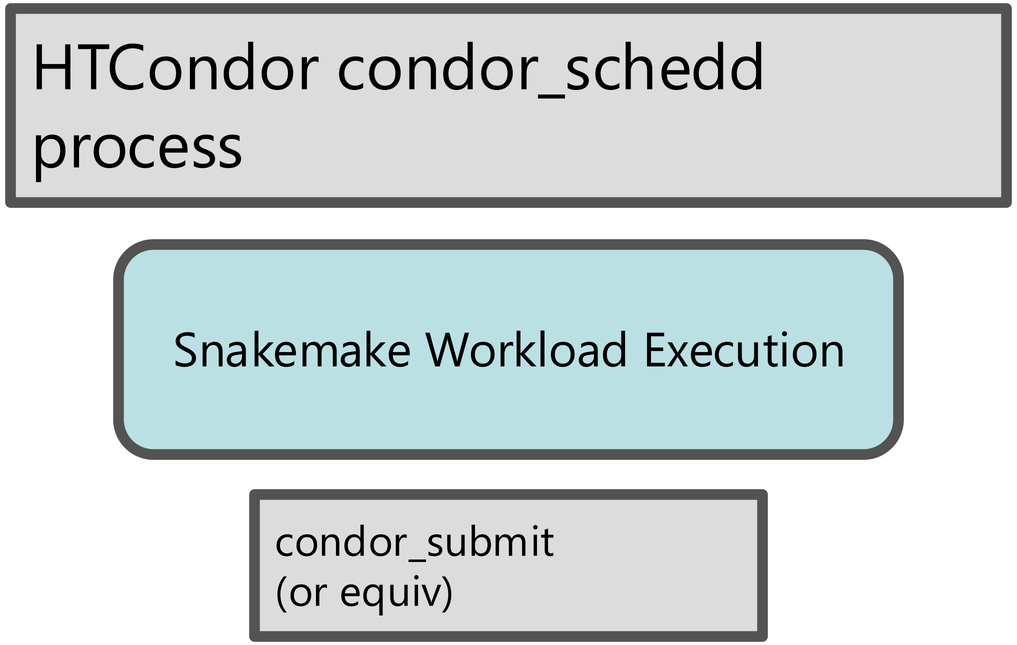
- At the core, it’s as simple as adding the following arguments:

```
snakemake --executor htcondor --shared-fs-usage none
```

- That is:
  - ‘**--executor htcondor**’: When executing a rule, use the HTCondor executor plugin instead of fork/exec.
  - ‘**--shared-fs-usage none**’: Request HTCondor executor to handle all input/output file movement.
- Additionally, the following are supported as user-defined resources in the configuration YAML:
  - request_{gpus,memory,disk}
  - requirements, rank, max_retries, retry_until
  - environment, job_wrapper

Next steps – “The True Believer” (full integration)

- In the prior slide, the “parent” snakemake process was running throughout the workload’s lifetime.
  - Something must monitor for completion, submit next steps, etc.
  - Equivalent: **condor_dagman** process.
- **Downside**: *Has anyone here actually executed* ***condor_dagman*** *directly*?
  - In fact: Snakemake will be killed if you close the terminal.
- Instead, we need the equivalent of **condor_submit_dag**:
  - Have the AP run the snakemake workflow instead of terminal.
  - Work-in-progress: submit the snakemake executable as a “scheduler universe” job running underneath the condor_schedd process in the AP.



Get it for yourself!

Check out the HTCondor-maintained Snakemake executor from https://github.com/htcondor/snakemake-executor-plugin-htcondor.

Check out the externally-maintained wf2wf package: https://github.com/csail/wf2wf

Parting Thoughts

- It’s Good to have engaged users enthusiastic about their favorite workload engine.
  - However, it’s tough to pick winners: not all will have the same lifetime at your site, not all are created equal
- There are three basic approaches to facilitating users:
  - “All in one”, embedding the engine inside a single job.
  - “Translation”, providing tools to (automatically) migrate users to HTCondor-native workloads.
  - “Integration”, providing plugins for the engine to HTCondor.
- Don’t assume “integration” is the best option: may be overkill for your local needs!
  - Also, by far, the most <u>expensive</u> option.
- For the Snakemake engine, at CHTC, we’ve seen all three in action!
  - Hopefully our Snakemake experience specifically …
  - … and workflow integration generically …
  - **Will benefit your site well!**
- **What’s the <u>Next Best Workload Engine</u> on your radar?**

Questions?

This project is supported by the National Science Foundation under Cooperative Agreements OAC-2030508. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

MORGRIDGE
INSTITUTE FOR RESEARCH