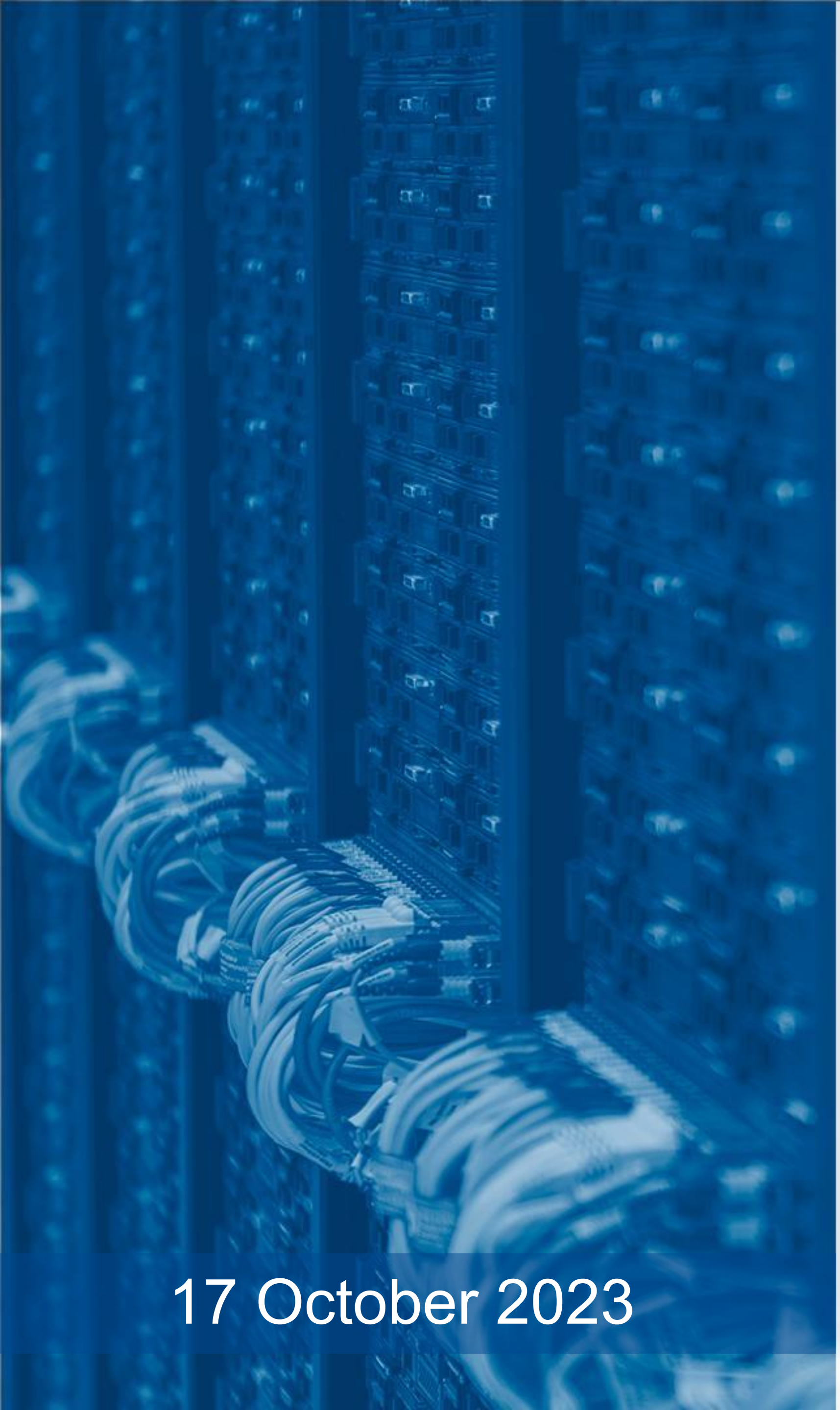


Overview of Autosubmit, Cylc, ecFlow and workflows in ESiWACE

Bruno P. Kinoshita, Miguel Castrillo

17 October 2023

Funded by the European Union

Outline

- Workflow managers, meta-schedulers, experiment managers
- An overview
  - Autosubmit
  - Cylc
  - ecFlow
- Final thoughts

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Workflow managers, meta schedulers, experiment managers

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Workflow managers

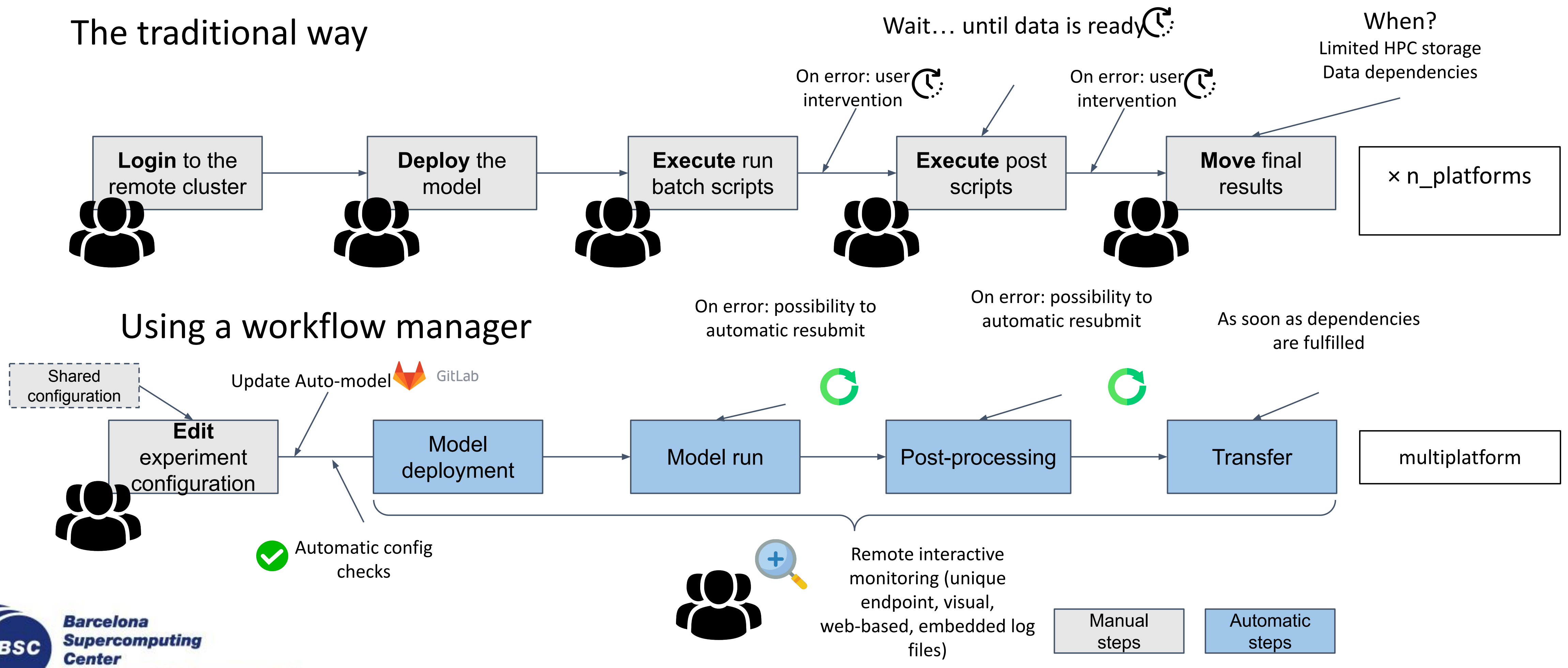
A workflow manager is a utility to run computational workflows.

A computational workflow is a series of steps in a certain sequence to complete a process (a graph of tasks). These steps can require running scripts and tools on a computer platform.

Besides the three workflow managers listed here, there are many other examples: Airflow, Jenkins, Nextflow, Luigi, Conductor, StreamFlow, Pegasus, COMPSs, WfExS, Dagster, cwltool, …

Why workflow managers?

What is a workflow manager good for?


The traditional way
Login to the remote cluster
Deploy the model
Execute run batch scripts
Execute post scripts
Move final results
× n_platforms
On error: user intervention
On error: user intervention
Wait… until data is ready
When?
Limited HPC storage
Data dependencies
Using a workflow manager
Shared configuration
Edit experiment configuration
Update Auto-model
GitLab
Automatic config checks
Model deployment
Model run
Post-processing
Transfer
multiplatform
On error: possibility to automatic resubmit
On error: possibility to automatic resubmit
As soon as dependencies are fulfilled
Remote interactive monitoring (unique endpoint, visual, web-based, embedded log files)
Manual steps
Automatic steps

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Meta schedulers

A meta scheduler is a utility that optimizes the scheduling of tasks by combining multiple job schedulers into a single unit.

You submit jobs to a meta scheduler, which in turn will organize these jobs and submit them to other job schedulers (PBS, Slurm, at, cloud, etc.) trying to optimize how resources are used.

Many workflow managers are also meta schedulers (but not all workflow managers).

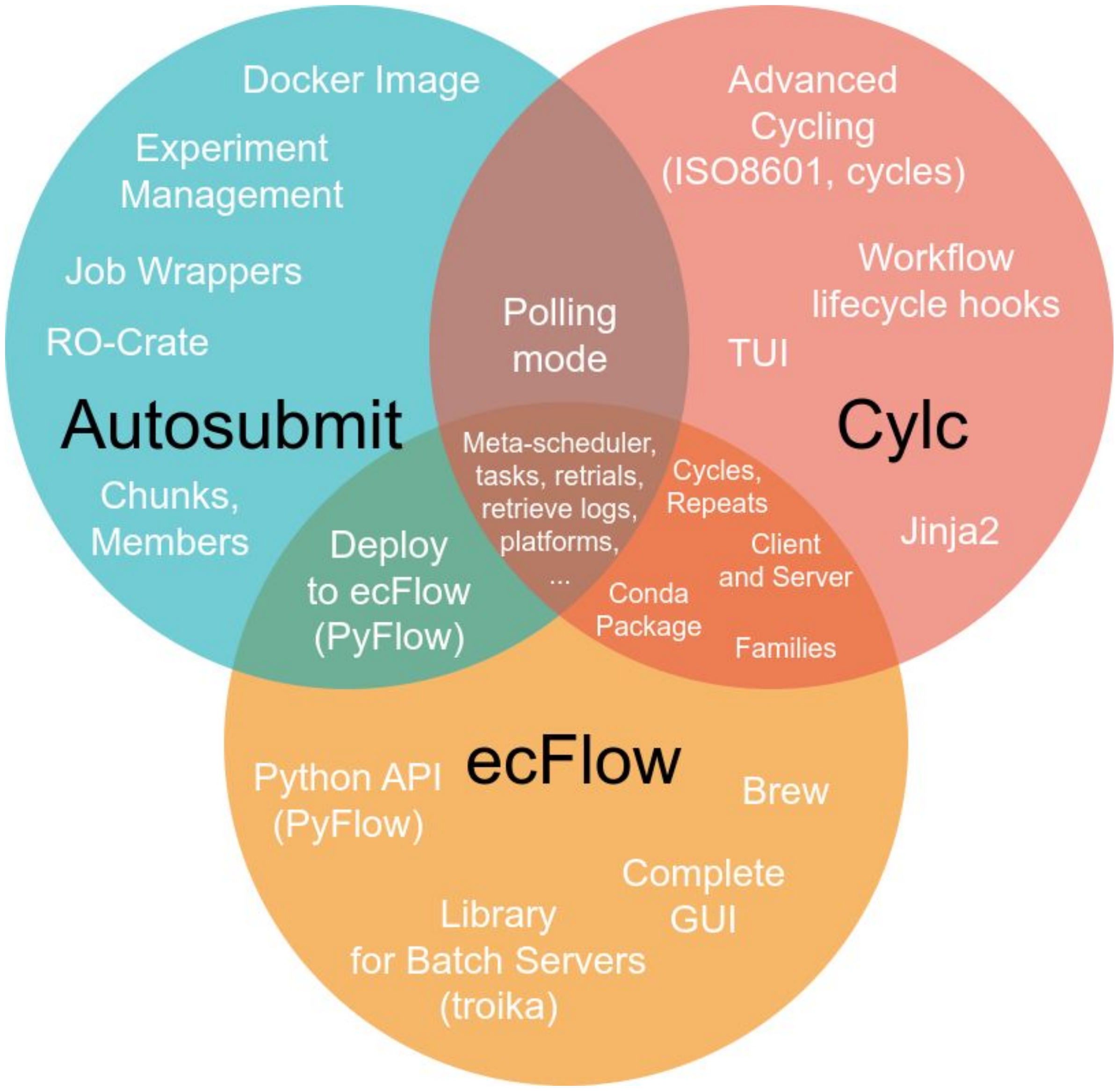
Experiment managers

An experiment manager is a utility that maintains scientific experiments.

It assigns unique & standardised IDs, keeps track of experiment configuration and metadata, and allow users to safely manage and share experiments.

Examples: prepIFS/IFShub, Autosubmit, rosie, mkexp, …

Autosubmit, Cylc, ecFlow



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

The common parts

All three are **Open Source** workflow managers that work as **meta-schedulers** with **platforms** such as PBS and Slurm.

They also support **job retrials**, **user management**, and **log retrieval** from remote platforms. There are many more commonalities amongst the three (and many differences too).

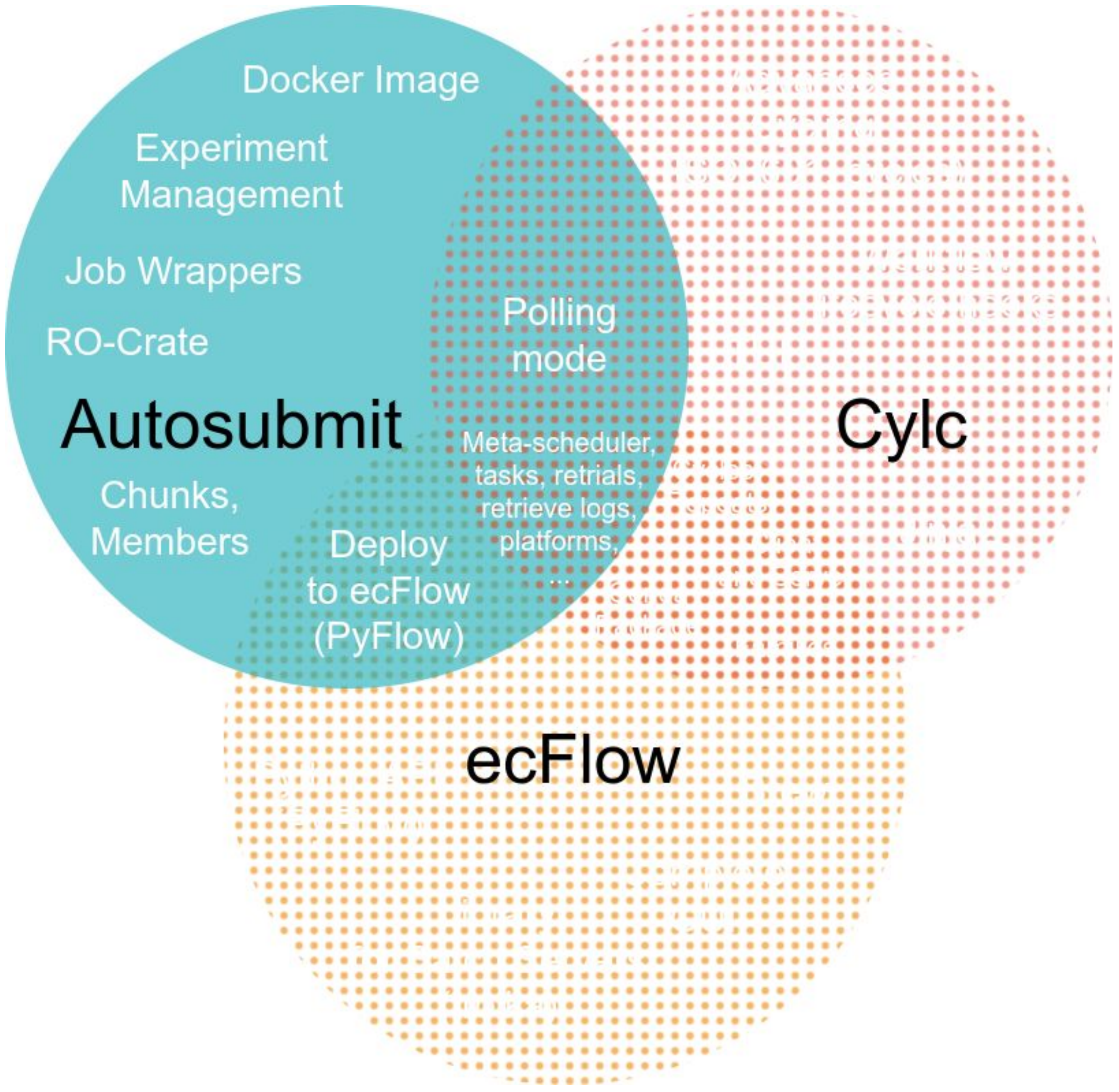
We present just a few in this overview.

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Autosubmit

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Autosubmit


Docker Image
Experiment Management
Job Wrappers
RO-Crate
Autosubmit
Chunks, Members
Polling mode
Meta-scheduler, tasks, retrials, retrieve logs, platforms, ...
Deploy to ecFlow (PyFlow)
Cylc
ecFlow

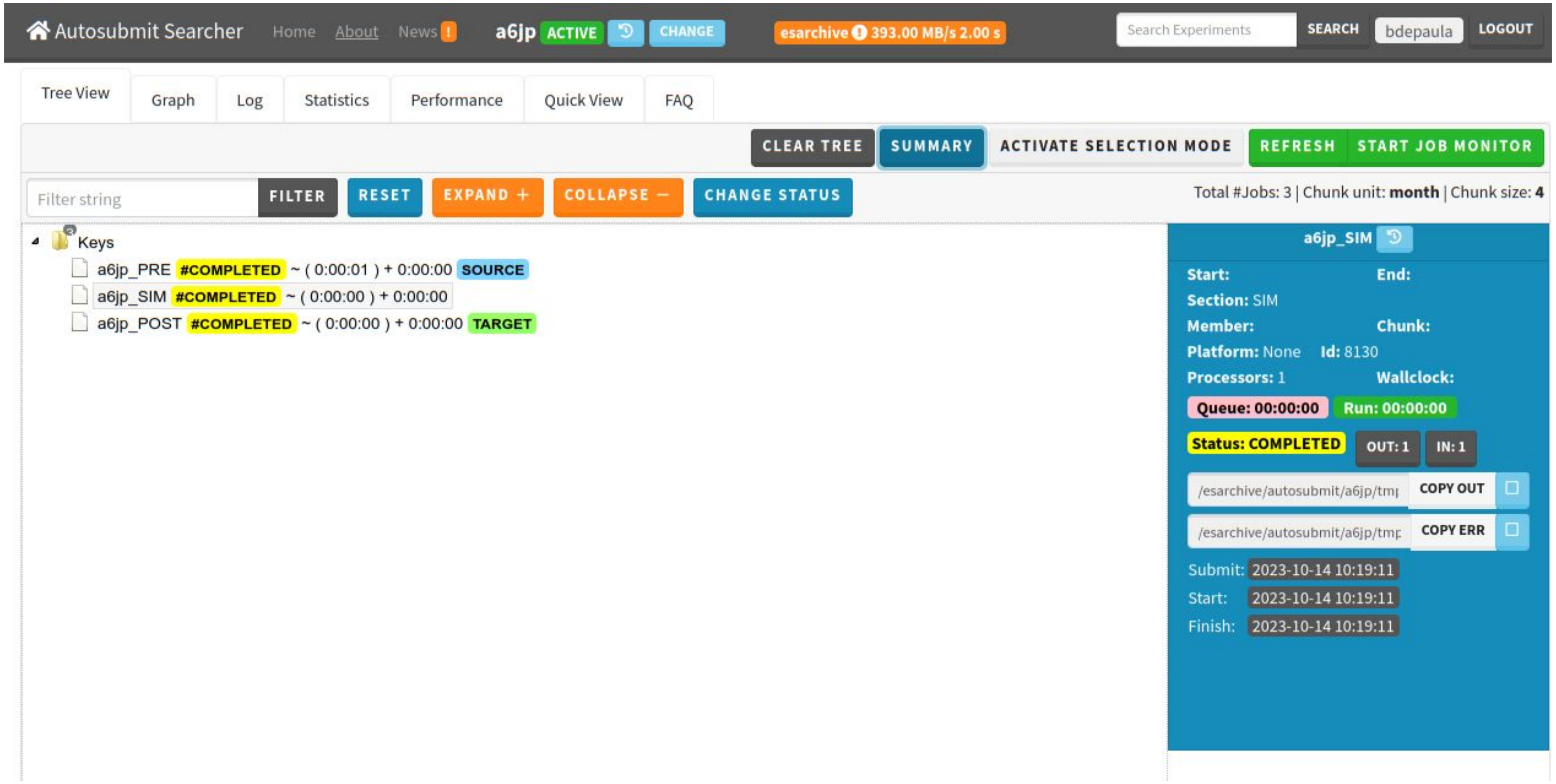
Autosubmit

Autosubmit is a Python **experiment** and workflow manager. Users create, configure, and share experiments (with unique & standardised IDs).

These experiments contain a workflow that can be scheduled to run on local and remote platforms (e.g. HPC).

It was created to manage climate experiments at the BSC.

GUI Screenshot



Total #Jobs: 3 | Chunk unit: **month** | Chunk size: **4**

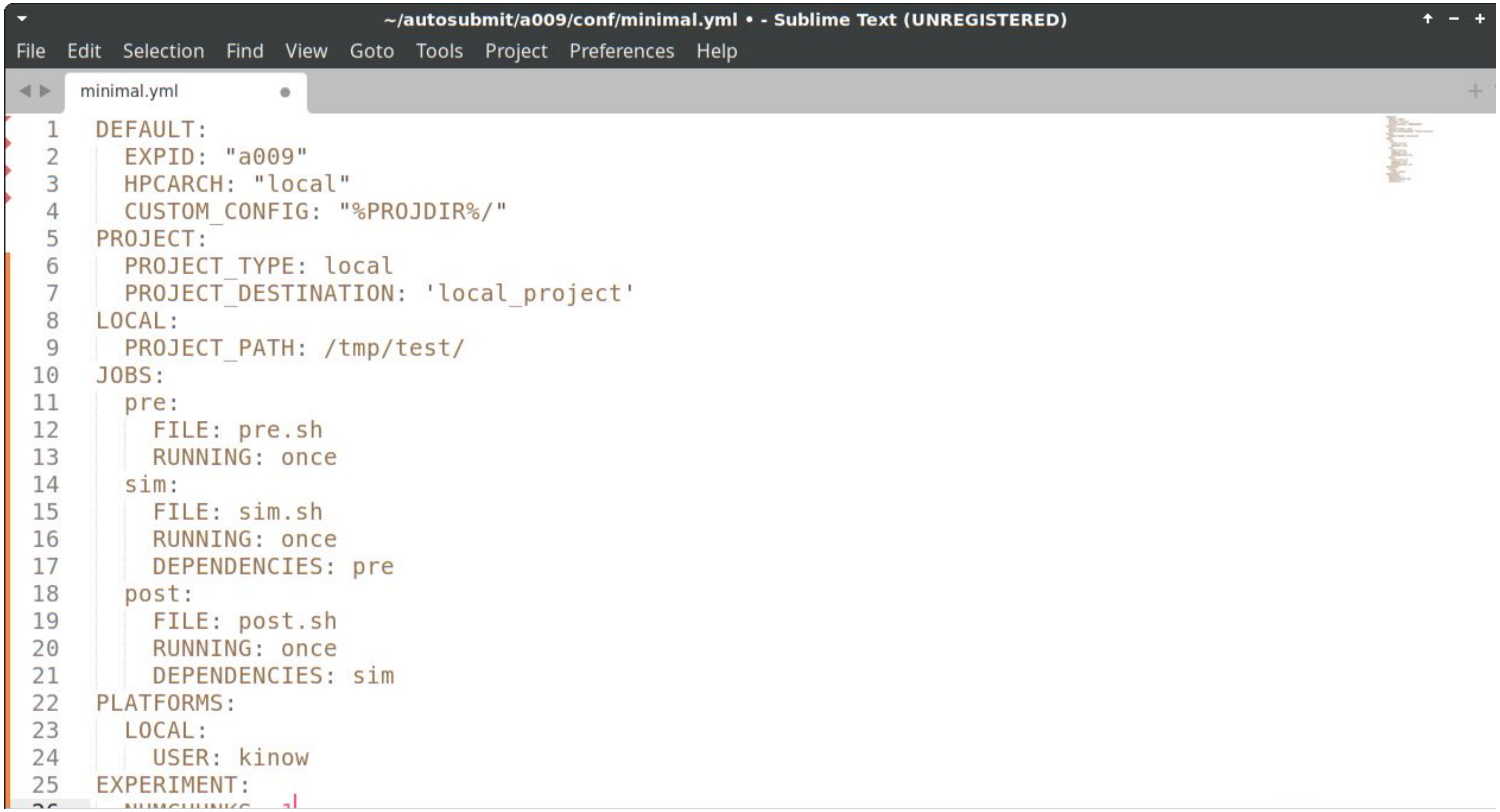
Keys
a6jp_PRE #COMPLETED ~ (0:00:01) + 0:00:00 SOURCE
a6jp_SIM #COMPLETED ~ (0:00:00) + 0:00:00
a6jp_POST #COMPLETED ~ (0:00:00) + 0:00:00 TARGET

a6jp_SIM

Start: **End:**
Section: SIM
Member: **Chunk:**
Platform: None **Id:** 8130
Processors: 1 **Wallclock:**
Queue: 00:00:00 **Run: 00:00:00**
Status: COMPLETED OUT: 1 IN: 1
/esarchive/autosubmit/a6jp/tmp
/esarchive/autosubmit/a6jp/tmp
Submit: 2023-10-14 10:19:11
Start: 2023-10-14 10:19:11
Finish: 2023-10-14 10:19:11

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Configuration Screenshot


~/autosubmit/a009/conf/minimal.yml • - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
minimal.yml

```yaml
DEFAULT:
  EXPID: "a009"
  HPCARCH: "local"
  CUSTOM_CONFIG: "%PROJDIR%/"
PROJECT:
  PROJECT_TYPE: local
  PROJECT_DESTINATION: 'local_project'
LOCAL:
  PROJECT_PATH: /tmp/test/
JOBS:
  pre:
    FILE: pre.sh
    RUNNING: once
  sim:
    FILE: sim.sh
    RUNNING: once
    DEPENDENCIES: pre
  post:
    FILE: post.sh
    RUNNING: once
    DEPENDENCIES: sim
PLATFORMS:
  LOCAL:
    USER: kinow
EXPERIMENT:
```

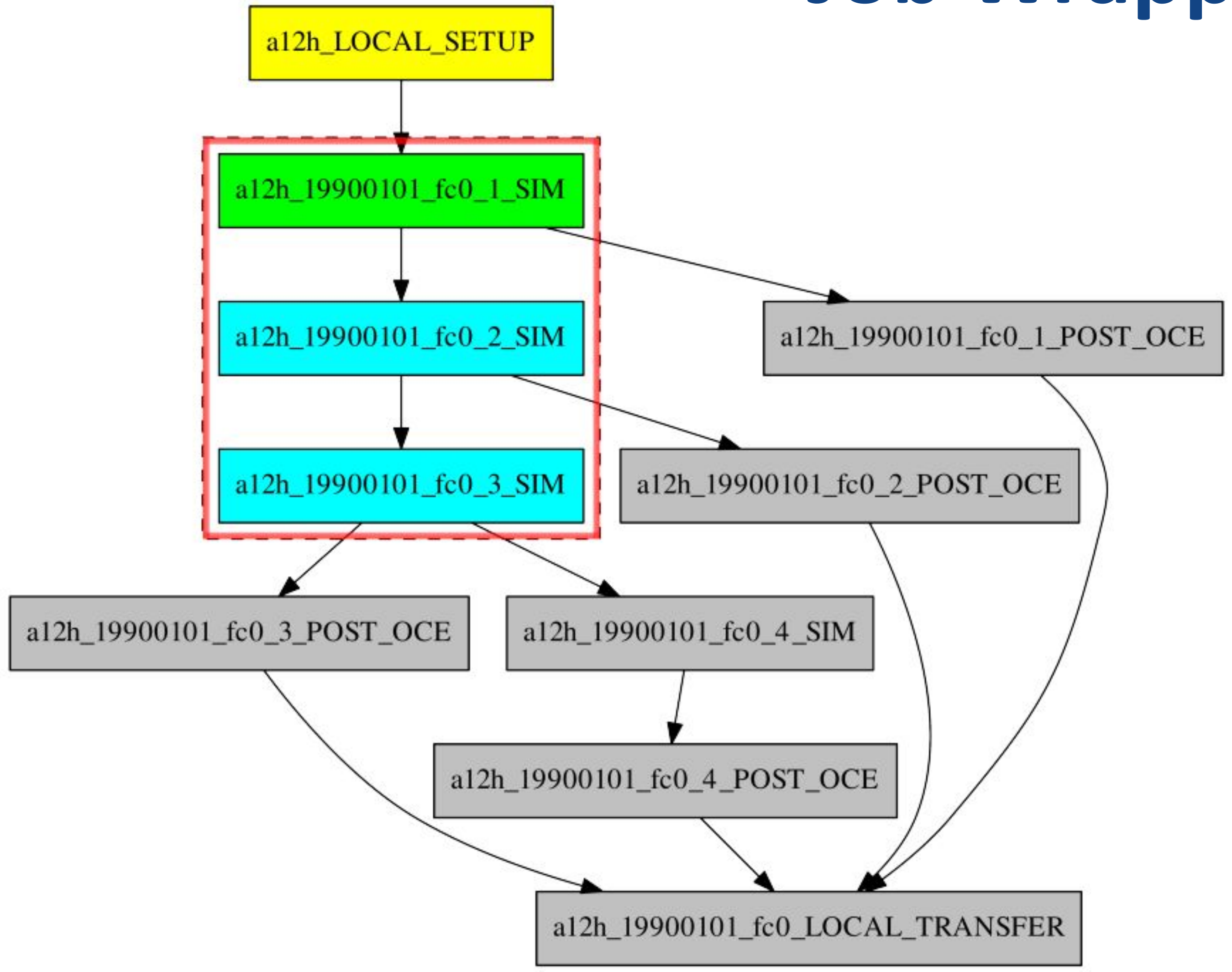
Job Wrappers

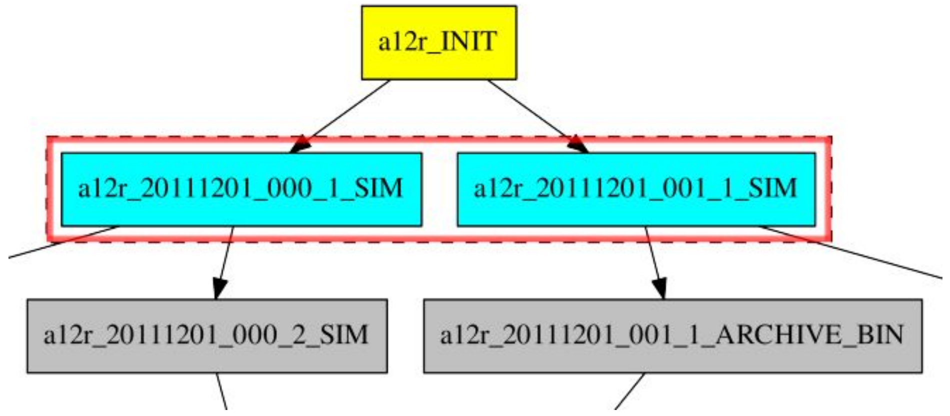
Autosubmit is able to run workflows in environments where multiple users compete for resources to schedule jobs, by "**wrapping**" multiple jobs and submitting as a single job.

This is essential for scheduling in HPC environments like MareNostrum 4, with limited resources shared by many groups.

Job Wrappers

Vertical Wrapper


a12h_LOCAL_SETUP
a12h_19900101_fc0_1_SIM
a12h_19900101_fc0_2_SIM
a12h_19900101_fc0_1_POST_OCE
a12h_19900101_fc0_3_SIM
a12h_19900101_fc0_2_POST_OCE
a12h_19900101_fc0_3_POST_OCE
a12h_19900101_fc0_4_SIM
a12h_19900101_fc0_4_POST_OCE
a12h_19900101_fc0_LOCAL_TRANSFER



Horizontal Wrapper

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Members, Chunks

Autosubmit configuration contains concepts familiar to climate researchers, such as **start dates**, **members**, and **chunks**.

They are useful for configuring experiments for **ensemble climate simulations**.

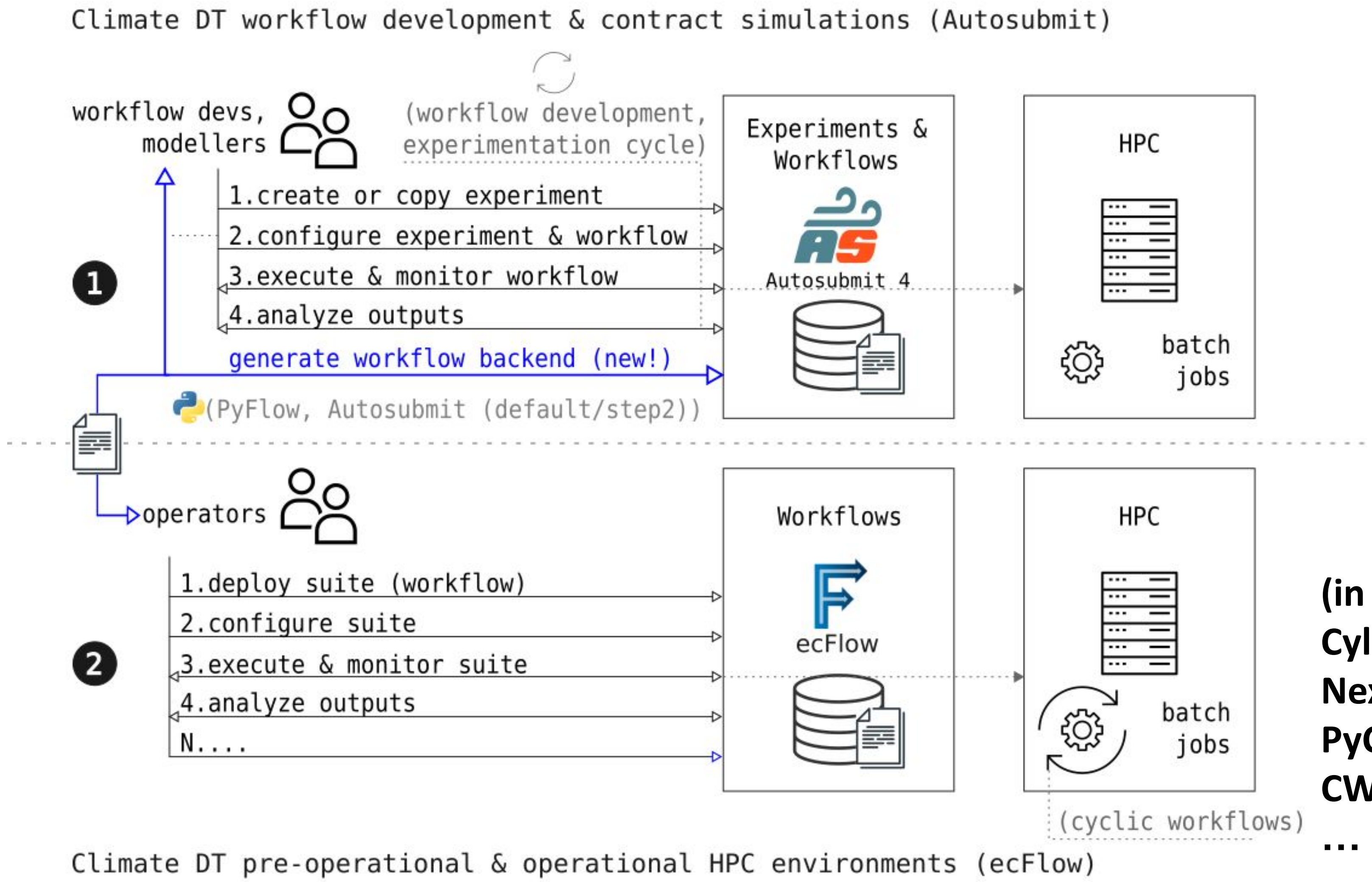
Other

Autosubmit provides an official **Docker** image, and is also the only that conforms to the **RO-Crate** standard (for metadata, provenance, FAIR).

Autosubmit and Cylc support connecting to remote platforms in an unidirectional way (via **polling**).

Both Autosubmit and ecFlow are able to deploy to **ecFlow** servers. Autosubmit uses PyFlow to generate an ecFlow suite.

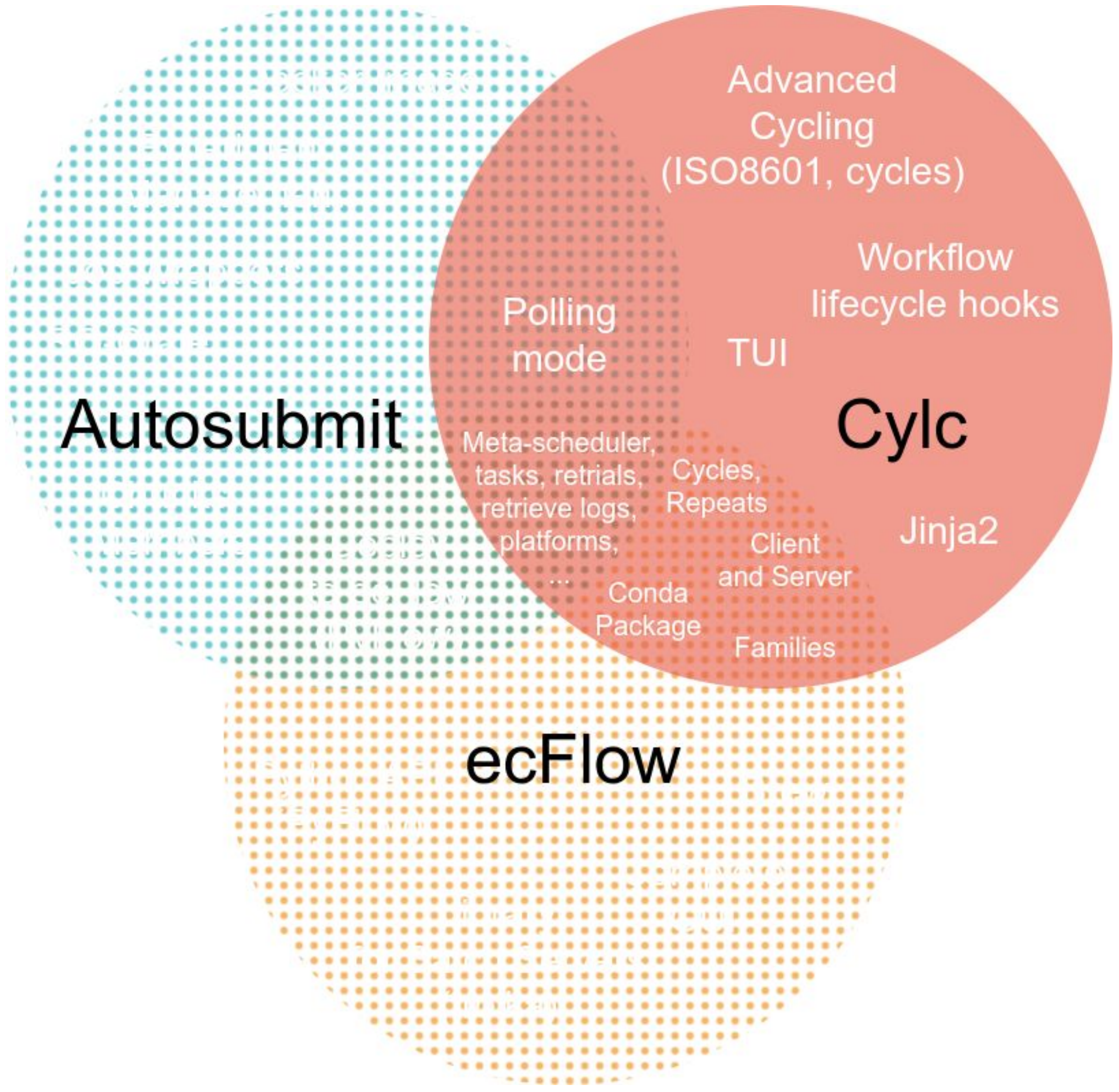
Autosubmit + PyFlow (ecFlow)


Climate DT workflow development & contract simulations (Autosubmit)
workflow devs, modellers
(workflow development, experimentation cycle)
Experiments & Workflows
Autosubmit 4
HPC
1
1.create or copy experiment
2.configure experiment & workflow
3.execute & monitor workflow
4.analyze outputs
generate workflow backend (new!)
(PyFlow, Autosubmit (default/step2))
batch jobs
operators
Workflows
ecFlow
HPC
2
1.deploy suite (workflow)
2.configure suite
3.execute & monitor suite
4.analyze outputs
N....
batch jobs
(cyclic workflows)
Climate DT pre-operational & operational HPC environments (ecFlow)

(in theory)
Cylc
Nextflow
PyCOMPSs
CWL
...

Cylc

Cylc


Advanced Cycling (ISO8601, cycles)
Workflow lifecycle hooks
Polling mode
TUI
Autosubmit
Cylc
Meta-scheduler, tasks, retrials, retrieve logs, platforms, ...
Cycles, Repeats
Client and Server
Jinja2
Conda Package
Families
ecFlow

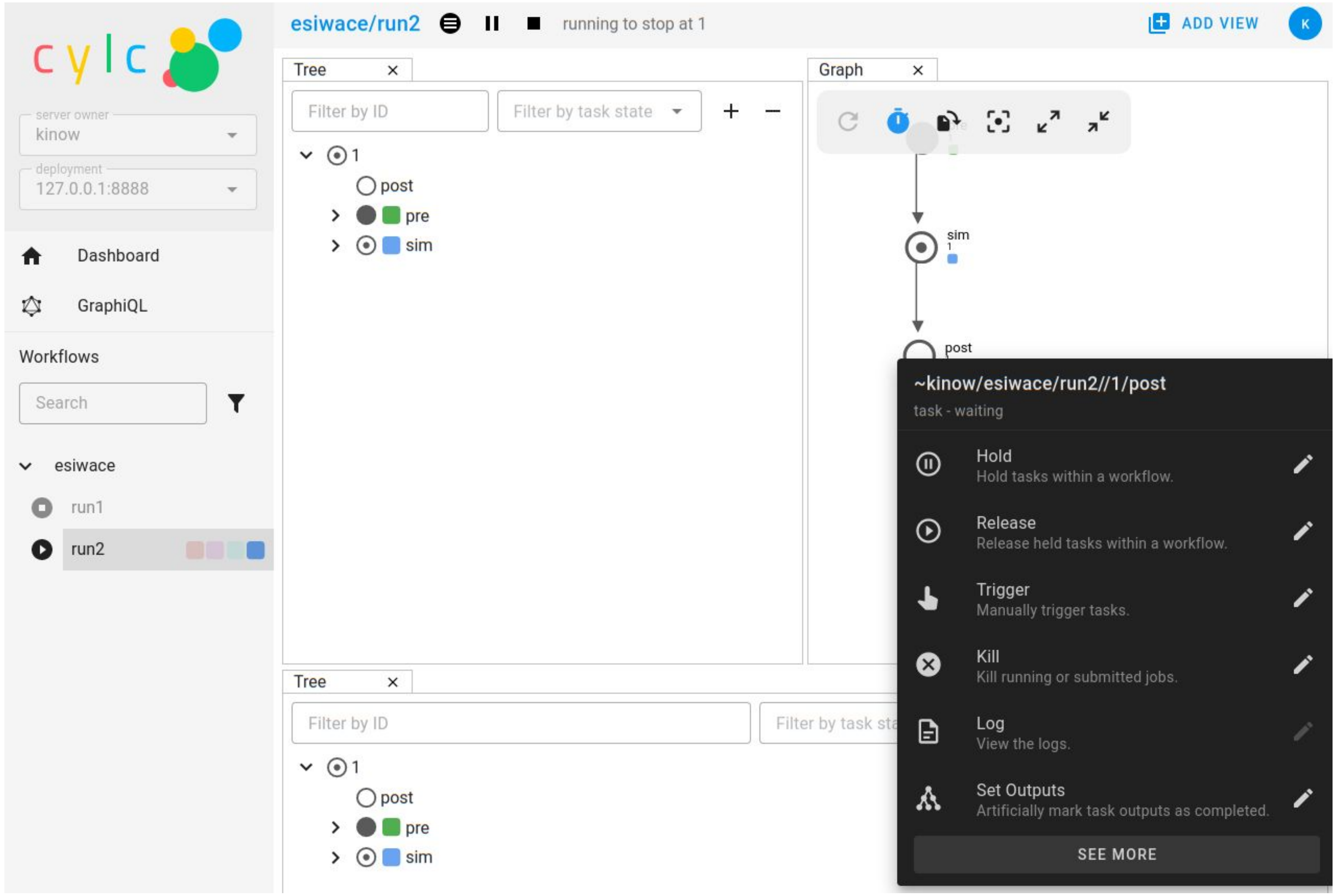
Cylc

Cylc (now Cylc Flow) is a workflow manager written in Python, created at NIWA, New Zealand, to manage NWP workflows.

NIWA and MetOffice use it to manage a large number of HPC jobs every year. Cylc 8 was redesigned to use Python 3 with a new Web interface.

It is the option with more features, and most modern UI. However, it is also the one with the steepest learning curve.

GUI Screenshot



~kinow/esiwace/run2//1/post

task - waiting

Hold
Hold tasks within a workflow.

Release
Release held tasks within a workflow.

Trigger
Manually trigger tasks.

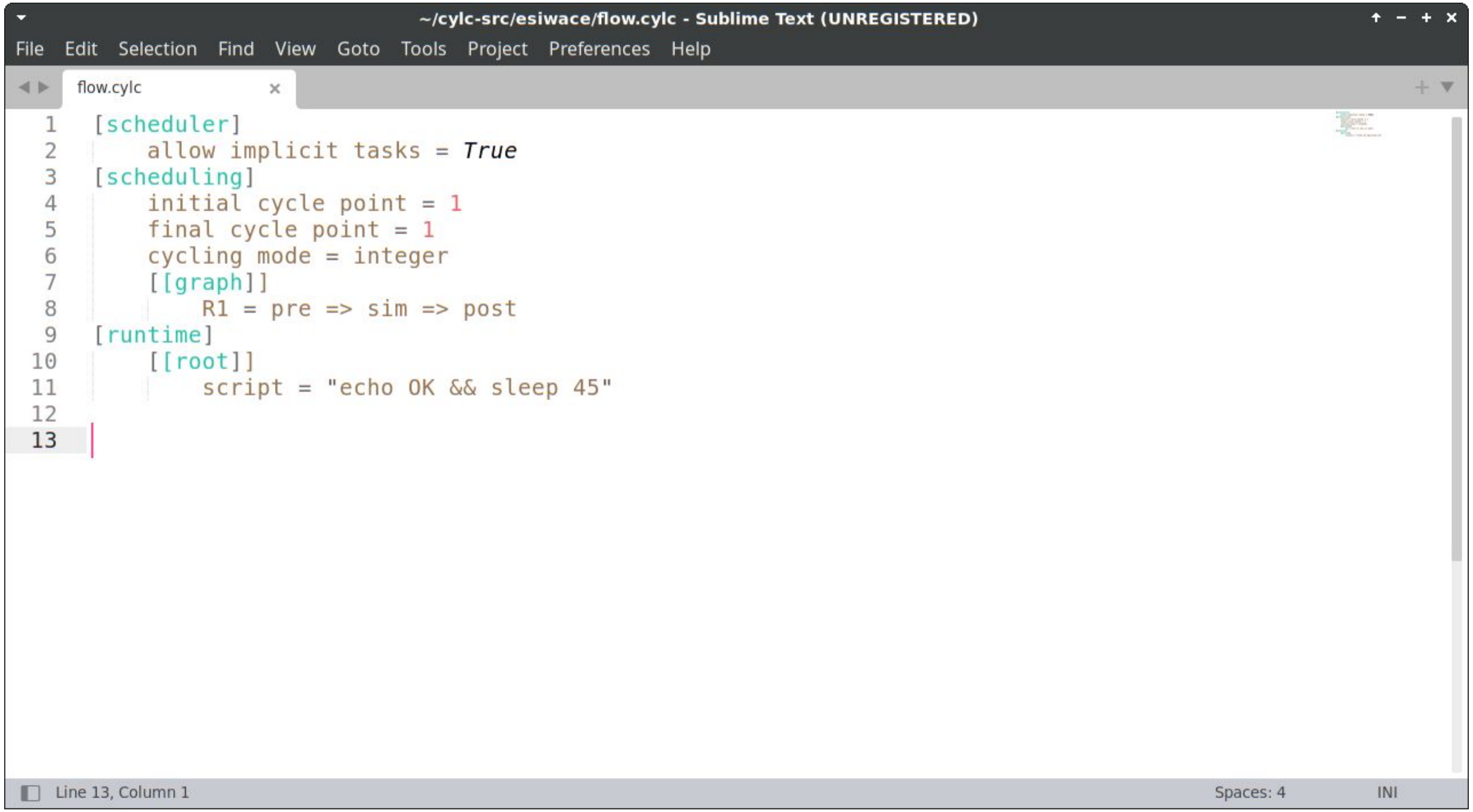
Kill
Kill running or submitted jobs.

Log
View the logs.

Set Outputs
Artificially mark task outputs as completed.

SEE MORE

Configuration Screenshot

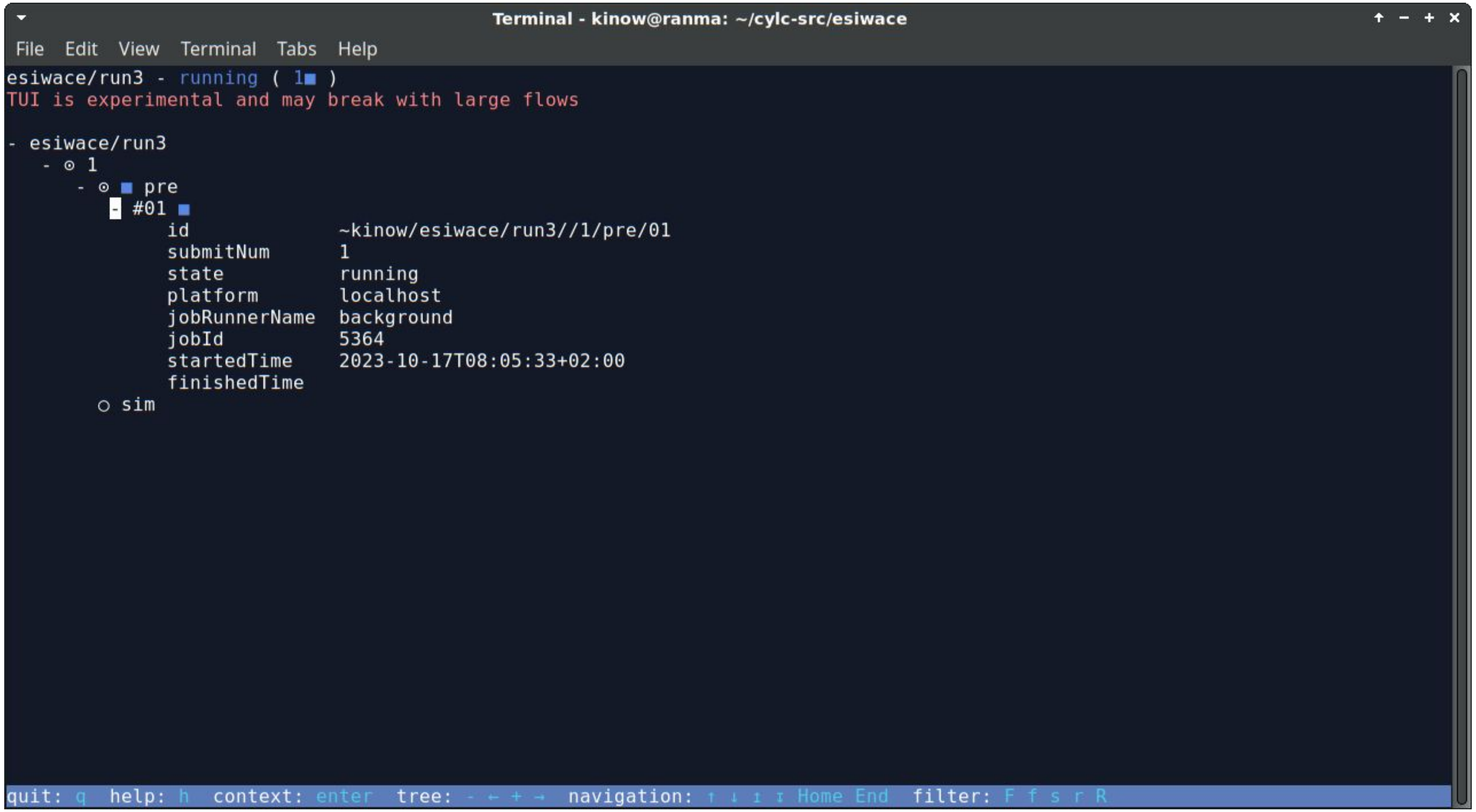

~/cylc-src/esiwace/flow.cylc - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
flow.cylc

```
[scheduler]
    allow implicit tasks = True
[scheduling]
    initial cycle point = 1
    final cycle point = 1
    cycling mode = integer
    [[graph]]
        R1 = pre => sim => post
[runtime]
    [[root]]
        script = "echo OK && sleep 45"

```

Line 13, Column 1
Spaces: 4
INI

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

TUI


Terminal - kinow@ranma: ~/cylc-src/esiwace
File Edit View Terminal Tabs Help

```
esiwace/run3 - running ( 1■ )
TUI is experimental and may break with large flows

- esiwace/run3
   - ⊙ 1
      - ⊙ ■ pre
         - #01 ■
              id             ~kinow/esiwace/run3//1/pre/01
              submitNum      1
              state          running
              platform       localhost
              jobRunnerName  background
              jobId          5364
              startedTime    2023-10-17T08:05:33+02:00
              finishedTime
        ○ sim

quit: q  help: h  context: enter  tree: - ← + →  navigation: ↑ ↓ ↥ ↧ Home End  filter: F f s r R
```

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Jinja2

Used in Cylc template scripts, Jinja allows users to customize their workflows using Python and importing Python modules.

Users are able to add conditionals to their scripts, and control the execution of tasks, as well as modify the workflow graph.

Workflow lifecycle hooks

Users are able to execute actions based on certain **workflow lifecycle** stages, through event handlers.

- Startup
- Shutdown
- Abort
- Workflow timeout
- Stall
- Stall timeout
- Inactivity timeout

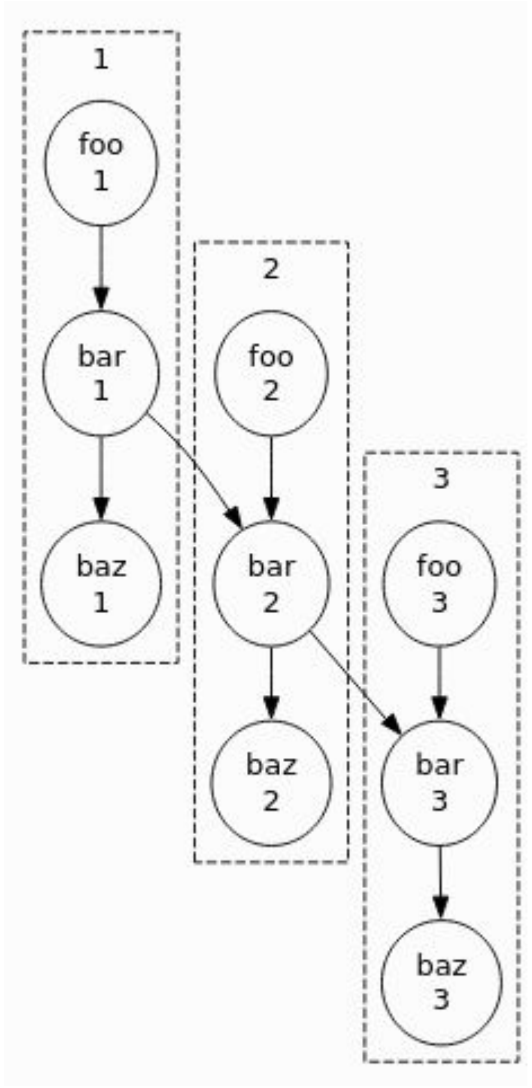
It also supports **task** lifecycle event handlers (with other stages for Tasks).

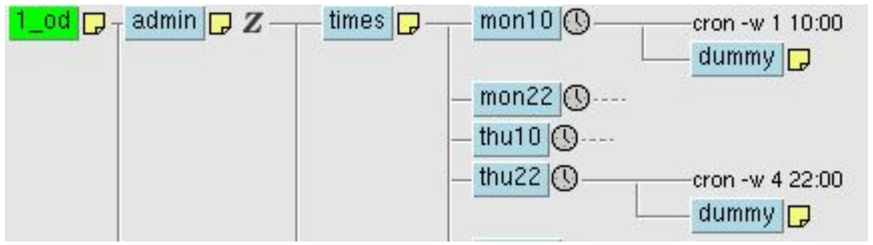
Cycles, Repeats

As both Cylc and ecFlow were developed for running operational NWP workflows, both support **repeats, or cycles**.

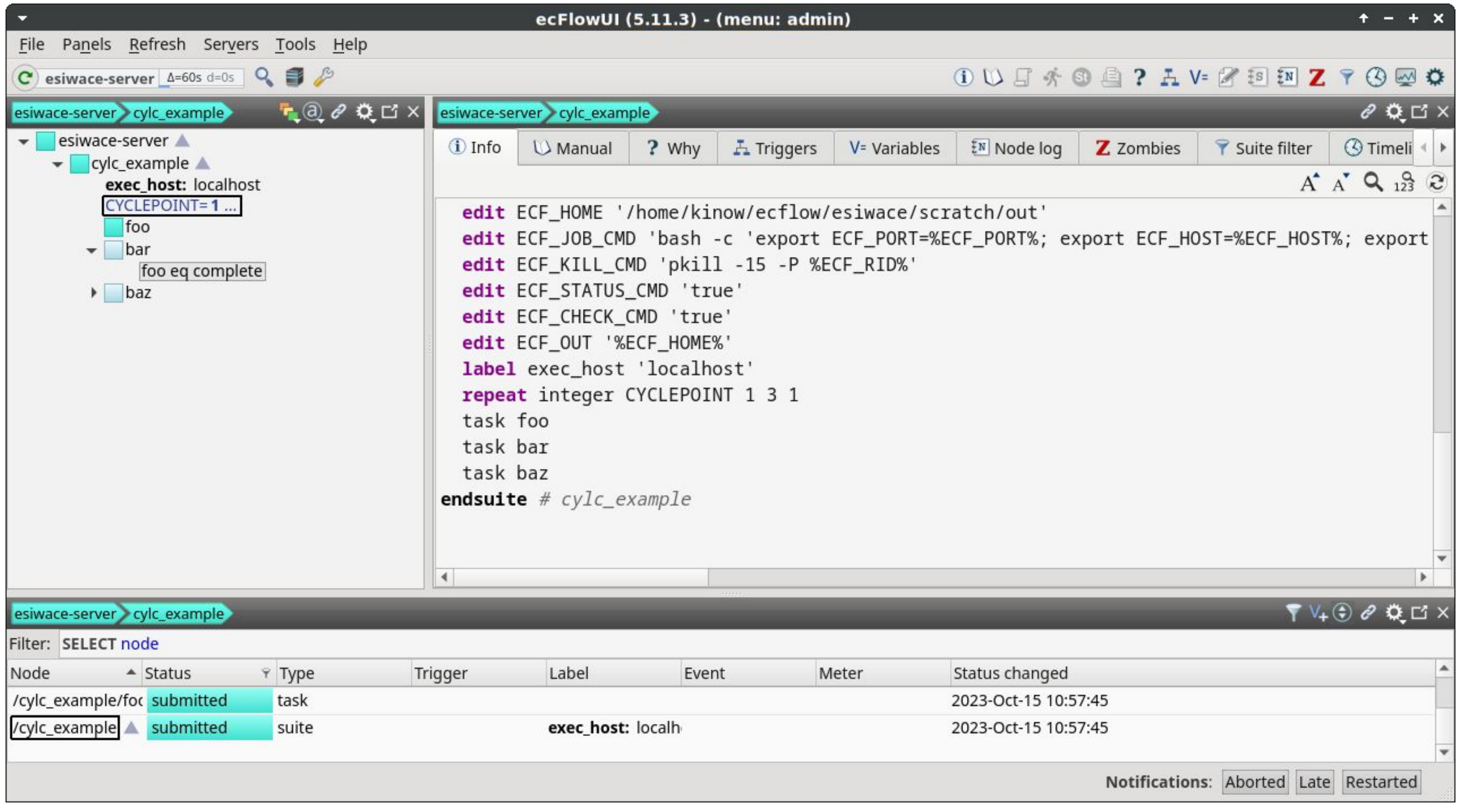
This way you are able to execute the same workflow multiple times, scheduling as many tasks as soon as possible (e.g. you can start tasks of the second cycle before the first has completed).

Cycles, Repeats





Left: Cylc cycling integer cycle points
Middle: ecFlow cron triggers
Bottom: ecFlow Repeats



ecFlowUI (5.11.3) - (menu: admin)
File Panels Refresh Servers Tools Help
esiwace-server
esiwace-server > cyclc_example
Info Manual Why Triggers Variables Node log Zombies Suite filter Timeli

esiwace-server
cylc_example
exec_host: localhost
CYCLEPOINT=1 ...
foo
bar
foo eq complete
baz

```
  edit ECF_HOME '/home/kinow/ecflow/esiwace/scratch/out'
  edit ECF_JOB_CMD 'bash -c 'export ECF_PORT=%ECF_PORT%; export ECF_HOST=%ECF_HOST%; export
  edit ECF_KILL_CMD 'pkill -15 -P %ECF_RID%'
  edit ECF_STATUS_CMD 'true'
  edit ECF_CHECK_CMD 'true'
  edit ECF_OUT '%ECF_HOME%'
  label exec_host 'localhost'
  repeat integer CYCLEPOINT 1 3 1
  task foo
  task bar
  task baz
endsuite # cylc_example
```

Filter: SELECT node

Node	Status	Type	Trigger	Label	Event	Meter	Status changed
/cylc_example/foo	submitted	task					2023-Oct-15 10:57:45
/cylc_example	submitted	suite		exec_host: localh			2023-Oct-15 10:57:45

Notifications: Aborted Late Restarted

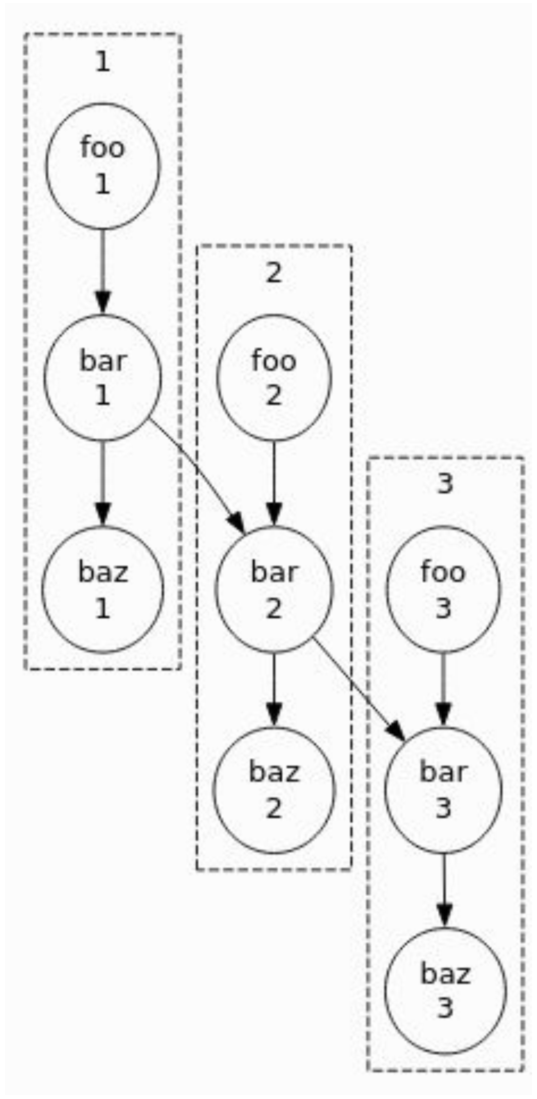
Advanced Cycling

Cycles in Cylc can be based on **ISO 8601** dates and periods (with isodatetime library), or integers.

Cylc unrolls the cycle loop to create a non-cycling workflow composed of repeating tasks - **no barrier between cycles**.

It is also the only one that handles **advanced cycling**, e.g. a -> a (actually a.1 -> a.2, or with dates), and multiple & merging “flows”.

Advanced Cycling



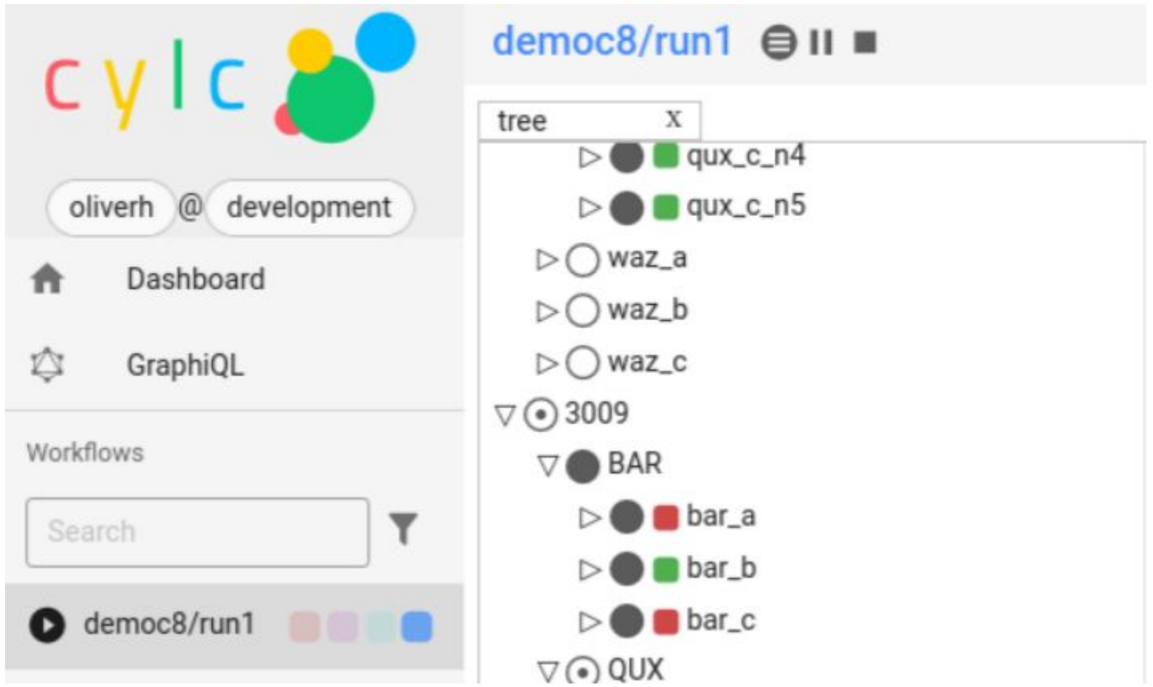
A flow is a single logical run through the graph. Cylc supports multiple concurrent flows over the same graph.

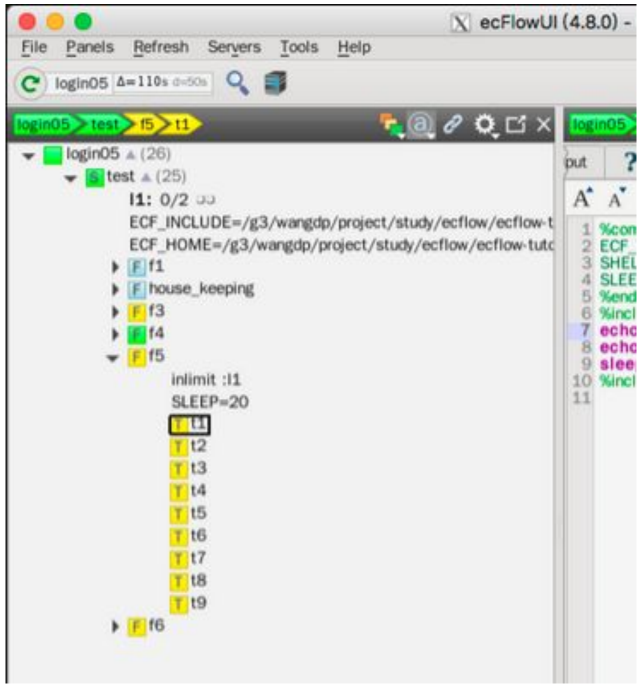
- In a single flow
    - foo.1 triggers bar.1
    - bar.1 triggers baz.1 and bar.2
    - **bar.2** may start before/at the same time baz.1 is started/submitted/running
    - You can have **multiple cycles** running **in parallel**
- You can start flows to re-run tasks or cycles, and they can be **merged**

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Families

Cylc and ecFlow both support grouping workflow tasks under "**families**". This is useful as you can use a family in a similar way to a task, in the graph dependency.




ecFlowUI (4.8.0) -
File Panels Refresh Servers Tools Help
login05 Δ=110s
login05 test f5 t1
login05 (26)
test (25)
l1: 0/2
ECF_INCLUDE=/g3/wangdp/project/study/ecflow/ecflow-t
ECF_HOME=/g3/wangdp/project/study/ecflow/ecflow-tutc
f1
house_keeping
f3
f4
f5
inlimit :l1
SLEEP=20
t1
t2
t3
t4
t5
t6
t7
t8
t9
f6

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

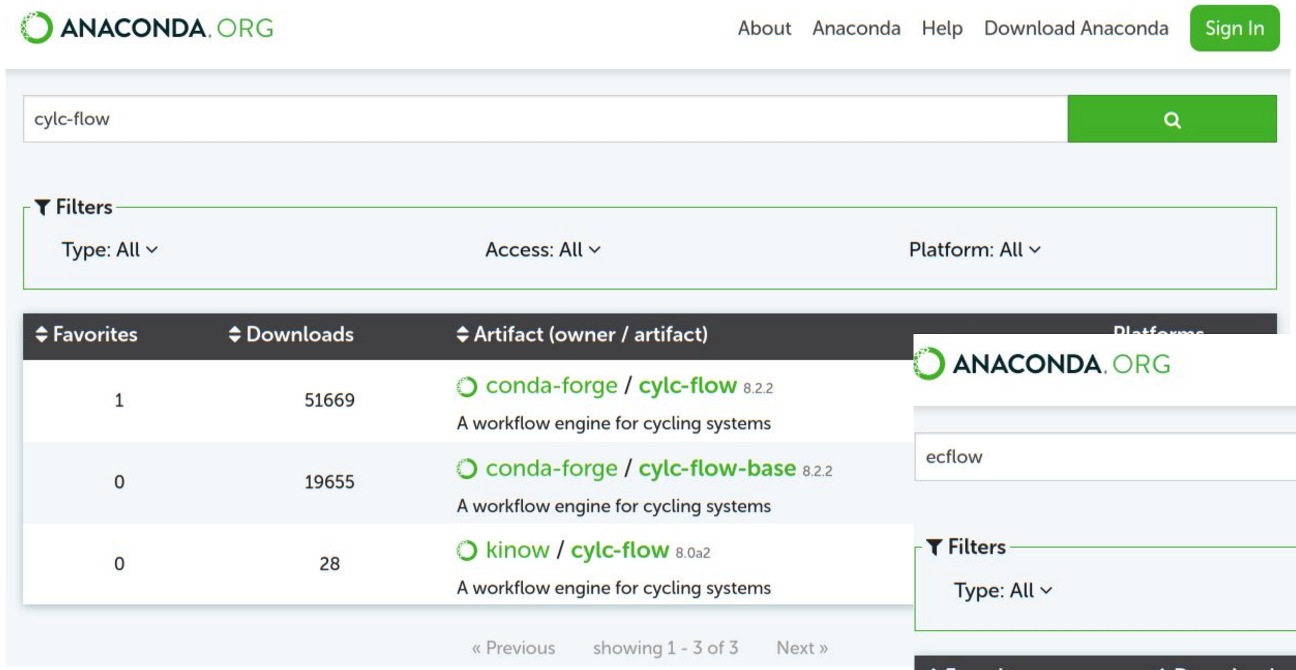
Client and Server

Cylc and ecFlow work with **client-server** architectures. In ecFlow you have the ecFlow server, and clients such as Python, ecFlow command-line, and the ecFlow GUI.

Cylc has an extra player, the UI Server, but also command-line and GUI clients.

Conda

Cylc and ecFlow provide official **Conda** packages.



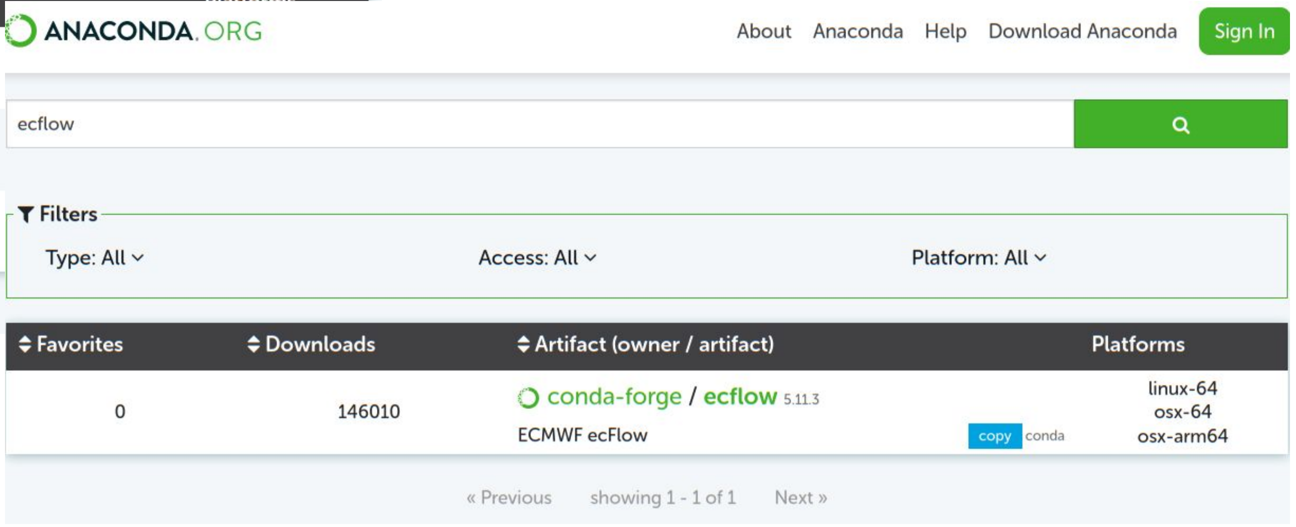
ANACONDA.ORG — About · Anaconda · Help · Download Anaconda · Sign In

Search: cylc-flow

Filters — Type: All · Access: All · Platform: All

Favorites	Downloads	Artifact (owner / artifact)
1	51669	conda-forge / cylc-flow 8.2.2 — A workflow engine for cycling systems
0	19655	conda-forge / cylc-flow-base 8.2.2 — A workflow engine for cycling systems
0	28	kinow / cylc-flow 8.0a2 — A workflow engine for cycling systems

« Previous showing 1 - 3 of 3 Next »



ANACONDA.ORG — About · Anaconda · Help · Download Anaconda · Sign In

Search: ecflow

Filters — Type: All · Access: All · Platform: All

Favorites	Downloads	Artifact (owner / artifact)	Platforms
0	146010	conda-forge / ecflow 5.11.3 — ECMWF ecFlow	linux-64 osx-64 osx-arm64

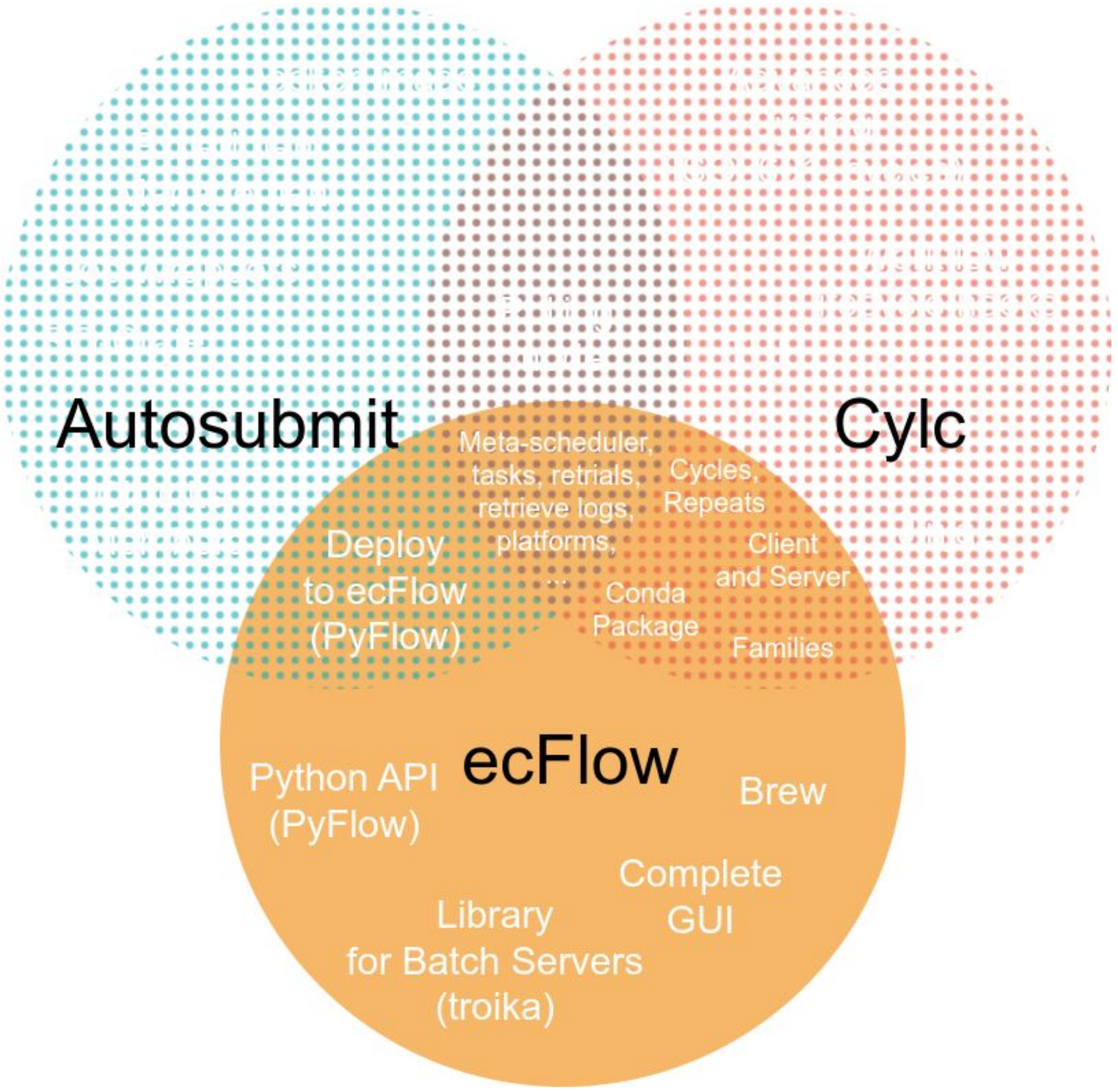
« Previous showing 1 - 1 of 1 Next »

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

ecFlow

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

ecFlow


Autosubmit
Cylc
Meta-scheduler, tasks, retrials, retrieve logs, platforms, ...
Cycles, Repeats
Client and Server
Deploy to ecFlow (PyFlow)
Conda Package
Families
ecFlow
Python API (PyFlow)
Brew
Complete GUI
Library for Batch Servers (troika)

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

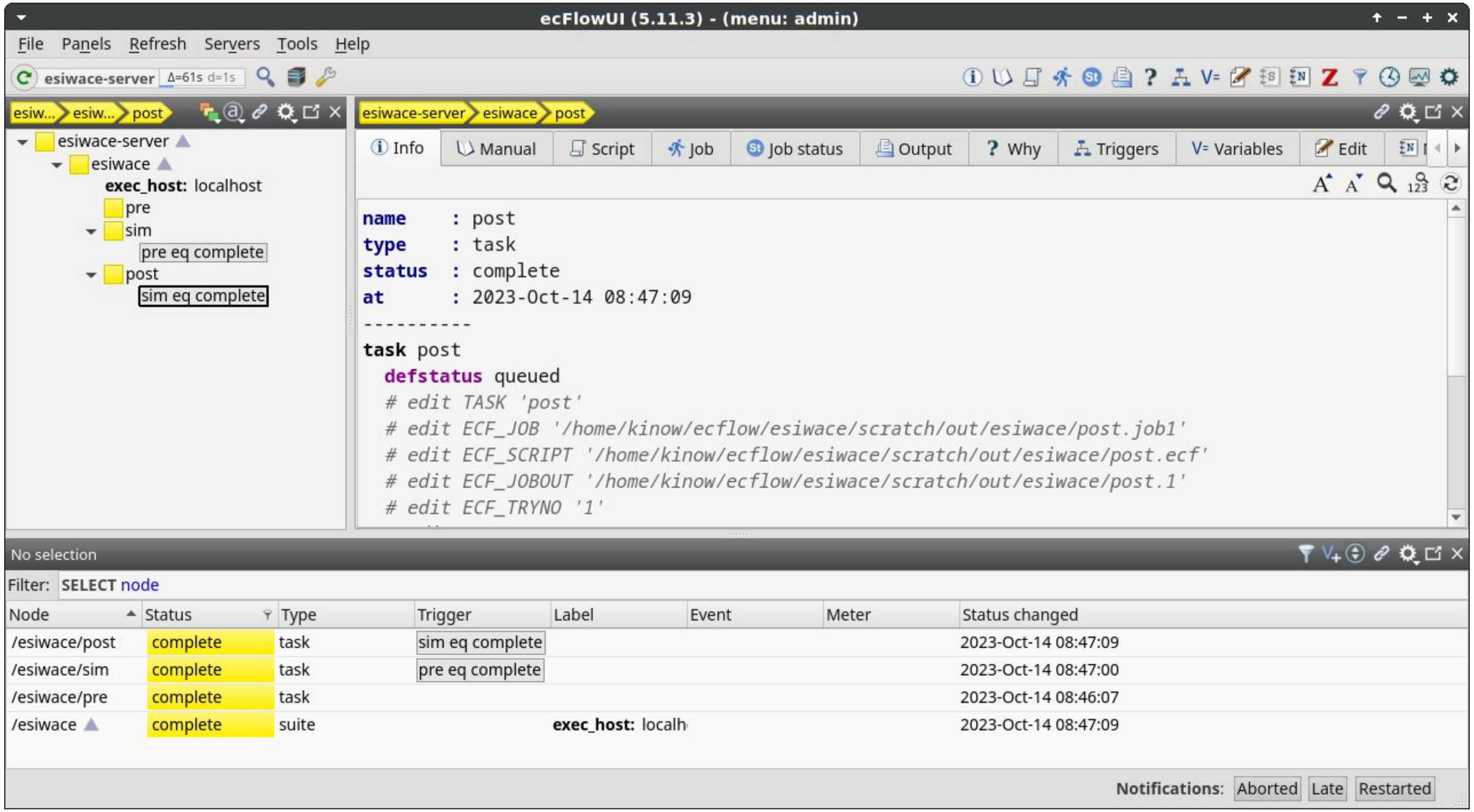
ecFlow

ecFlow was created by the ECMWF, as an evolution of SMS. It has been used over several years to run NWP workflows.

As it is written in C++, it has excellent performance when managing multiple workflows (suites). Its GUI is also the most complete.

While its cyclic workflows are not as powerful as Cylc's, it can repeat parts of the workflow and also use cron and repeats to trigger tasks and families.

GUI Screenshot


ecFlowUI (5.11.3) - (menu: admin)
File Panels Refresh Servers Tools Help
esiwace-server Δ=61s d=1s
esiw... esiw... post
esiwace-server esiwace post
Info Manual Script Job Job status Output Why Triggers Variables Edit

esiwace-server
esiwace
exec_host: localhost
pre
sim
pre eq complete
post
sim eq complete

```
name     : post
type     : task
status   : complete
at       : 2023-Oct-14 08:47:09
----------
task post
  defstatus queued
  # edit TASK 'post'
  # edit ECF_JOB '/home/kinow/ecflow/esiwace/scratch/out/esiwace/post.job1'
  # edit ECF_SCRIPT '/home/kinow/ecflow/esiwace/scratch/out/esiwace/post.ecf'
  # edit ECF_JOBOUT '/home/kinow/ecflow/esiwace/scratch/out/esiwace/post.1'
  # edit ECF_TRYNO '1'
```

No selection

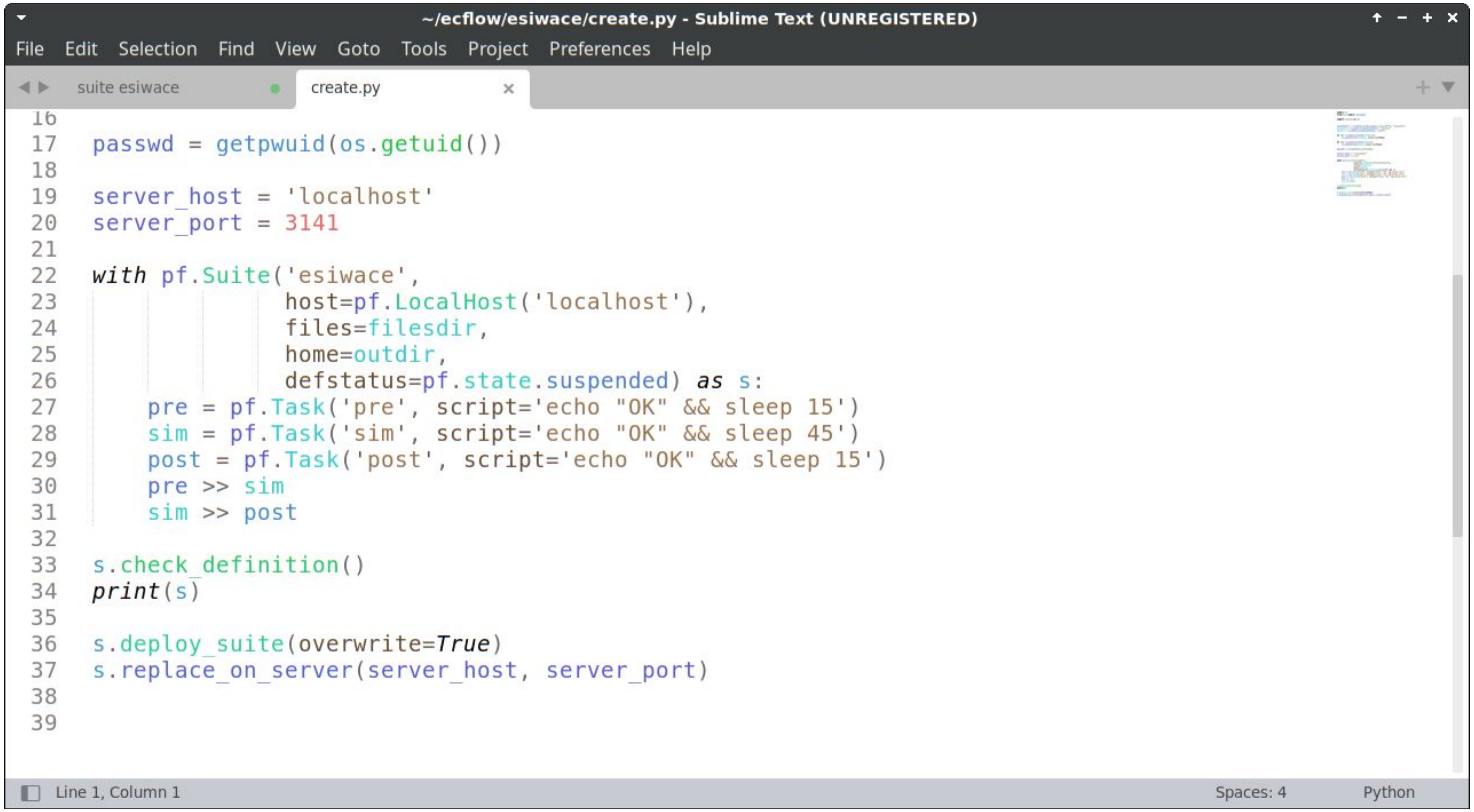
Filter: SELECT node

Node	Status	Type	Trigger	Label	Event	Meter	Status changed
/esiwace/post	complete	task	sim eq complete				2023-Oct-14 08:47:09
/esiwace/sim	complete	task	pre eq complete				2023-Oct-14 08:47:00
/esiwace/pre	complete	task					2023-Oct-14 08:46:07
/esiwace	complete	suite		exec_host: localh			2023-Oct-14 08:47:09

Notifications: Aborted Late Restarted

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

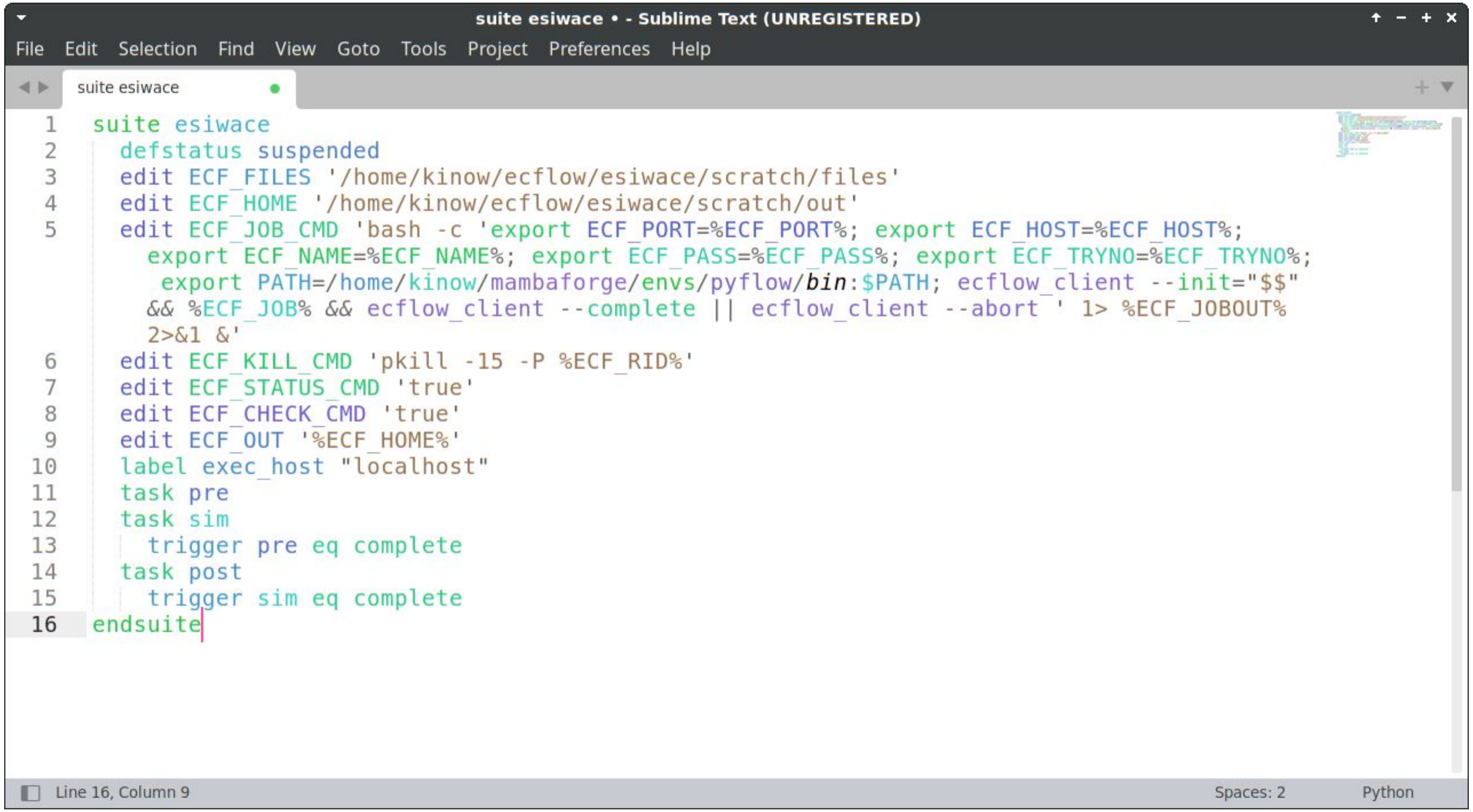
Configuration Screenshot


~/ecflow/esiwace/create.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
suite esiwace
create.py
Line 1, Column 1
Spaces: 4
Python

```python

passwd = getpwuid(os.getuid())

server_host = 'localhost'
server_port = 3141

with pf.Suite('esiwace',
              host=pf.LocalHost('localhost'),
              files=filesdir,
              home=outdir,
              defstatus=pf.state.suspended) as s:
    pre = pf.Task('pre', script='echo "OK" && sleep 15')
    sim = pf.Task('sim', script='echo "OK" && sleep 45')
    post = pf.Task('post', script='echo "OK" && sleep 15')
    pre >> sim
    sim >> post

s.check_definition()
print(s)

s.deploy_suite(overwrite=True)
s.replace_on_server(server_host, server_port)


```

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Configuration Screenshot


suite esiwace • - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
suite esiwace

```
suite esiwace
  defstatus suspended
  edit ECF_FILES '/home/kinow/ecflow/esiwace/scratch/files'
  edit ECF_HOME '/home/kinow/ecflow/esiwace/scratch/out'
  edit ECF_JOB_CMD 'bash -c 'export ECF_PORT=%ECF_PORT%; export ECF_HOST=%ECF_HOST%;
    export ECF_NAME=%ECF_NAME%; export ECF_PASS=%ECF_PASS%; export ECF_TRYNO=%ECF_TRYNO%;
     export PATH=/home/kinow/mambaforge/envs/pyflow/bin:$PATH; ecflow_client --init="$$"
    && %ECF_JOB% && ecflow_client --complete || ecflow_client --abort ' 1> %ECF_JOBOUT%
    2>&1 &'
  edit ECF_KILL_CMD 'pkill -15 -P %ECF_RID%'
  edit ECF_STATUS_CMD 'true'
  edit ECF_CHECK_CMD 'true'
  edit ECF_OUT '%ECF_HOME%'
  label exec_host "localhost"
  task pre
  task sim
    trigger pre eq complete
  task post
    trigger sim eq complete
endsuite
```

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

PyFlow

ecFlow has had a Python API for a long time. ECMWF released now a Python library called **PyFlow**, that is able to generate ecFlow workflows with a simple Python API.

troika

While all three workflow managers support scheduling jobs using remote platforms, ecFlow is the only of the three that uses a **dedicated library** for that: **troika**.

It provides a simple configuration model, and allows users to add custom platforms (called sites).

FOSDEM 23, “Troika: Submit, monitor, and interrupt jobs on any HPC system with the same interface”
https://archive.fosdem.org/2023/schedule/event/troika_hpc_jobs/

Complete GUI

ecFlow users are able to manage the complete workflow (suite) using only the GUI (although the command-line client is useful in some cases too).

The Autosubmit GUI is read-only, and the Cylc 8 UI still has features that are being migrated from Cylc 7, or that have not been implemented yet.

Brew

ecFlow is the only of the three that provides a **brew** installer for MacOS.

Final thoughts

This is a general overview

The best workflow manager **depends** on the use case.

Some features might help you to decide which workflow manager to use (installation method, networking security limitations, maintenance, etc.).

This is a general overview, and it may be unfair as there are many other features included in each of these workflow managers. Check out their websites for more before making a decision on which one to use.

Personal take on this

I hope for **more integration** between workflow managers (like what is happening in Destination Earth with Autosubmit & ecFlow).

Also for more open **standards** to be adopted, like CWL, WDL, RO-Crate, FDO, DRMAA, or even closed standards like ISO-8601 (or its newer versions).

Finally, it would be great to have more “building block” shared among workflow managers. e.g. have Autosubmit Jobs Wrappers available in other workflow managers, or Cylc’s date cycles (isodatetime), or ECMWF’s Troika, or DRMAA used by more tools.

Work in ESiWACE3

These workflow managers are used by ESiWACE members to run weather and climate workflows — Cylc and Autosubmit have received funding.

There is a task in ESiWACE3 to **containerize** EC-Earth 4, a community ESM, and to orchestrate it with Autosubmit. Containerized models improve portability across workflow managers, and HPC platforms.

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Questions?

- https://autosubmit.readthedocs.io/
- https://cylc.github.io/
- https://ecflow.readthedocs.io/

Thank you

bruno.depaulakinoshita@bsc.es