Troika: Submit, monitor, and interrupt jobs on any HPC system with the same interface

Axel Bonet, Olivier Iffrig

About ECMWF

Established in 1975, Intergovernmental Organisation
- 23 Member States | 12 Cooperating States
- 350+ staff

24/7 operational service
- Operational NWP – 4x HRES+ENS forecasts / day
- Supporting NWS (coupled models) and businesses
- Largest meteorological archive (400PiB+)

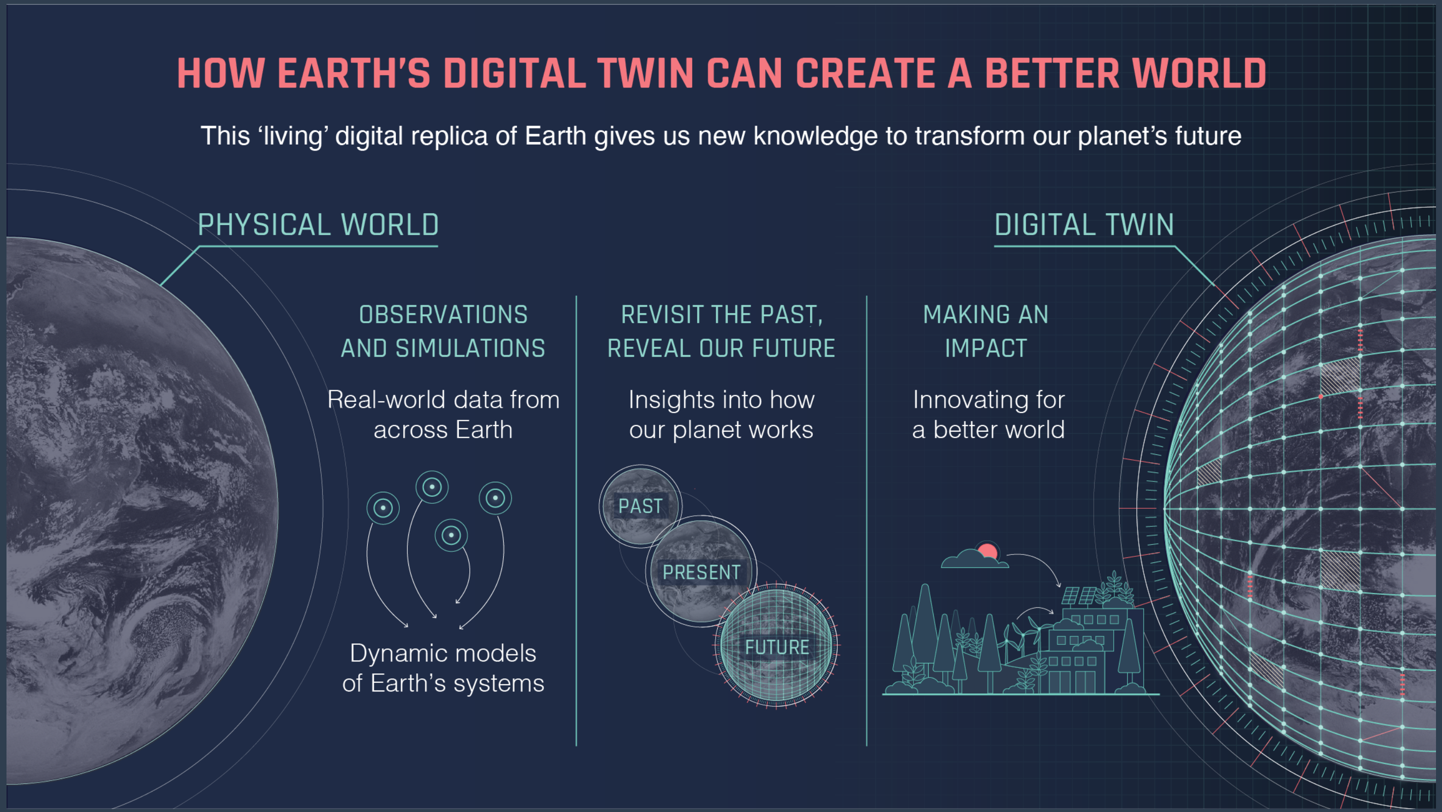
Research institution
- Experiments to continuously improve our models
- Reforecasts and Climate Reanalysis (ERA5)

Operate 2 EU Copernicus Services
- Climate Change Service (C3S)
- Atmosphere Monitoring Service (CAMS)
- Support Copernicus Emergency Management Service (CEMS)

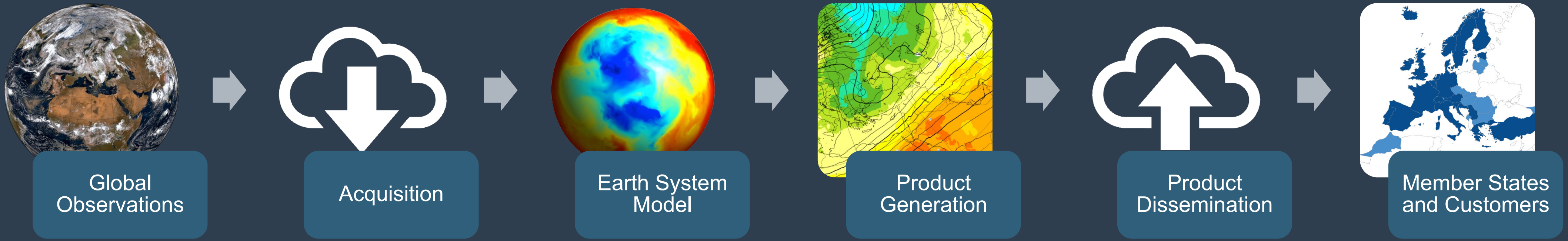
Destination Earth
- Operates two Digital Twins
- Operates the DestinE Digital Twin Engine (DTE)

A project: Destination Earth

- EU programme for weather and climate
- Large collaboration driven by ECMWF, ESA and EUMETSAT
- Simulations of the Earth at 1km resolution (digital twins)
- Running on multiple HPC centres across Europe
- Will require flexibility and adaptability of the workflows



Think workflow, not job



- A workflow consists of multiple tasks with dependencies
- Our workflows can have thousands of tasks each
- Multiple types of workflows: operational (time-critical), research, support…
- We run about half a million tasks per day on our HPC system
- Some tasks are big parallel jobs, but most are small

How can we handle these workflows?

Our workflow manager: ecFlow

- Manages a task graph as a tree with additional dependencies
- Runs a script for every task (leaf node in the tree)
- Stores variables to pre-process the scripts
- Keeps track of the task status
- Fetches log files on demand

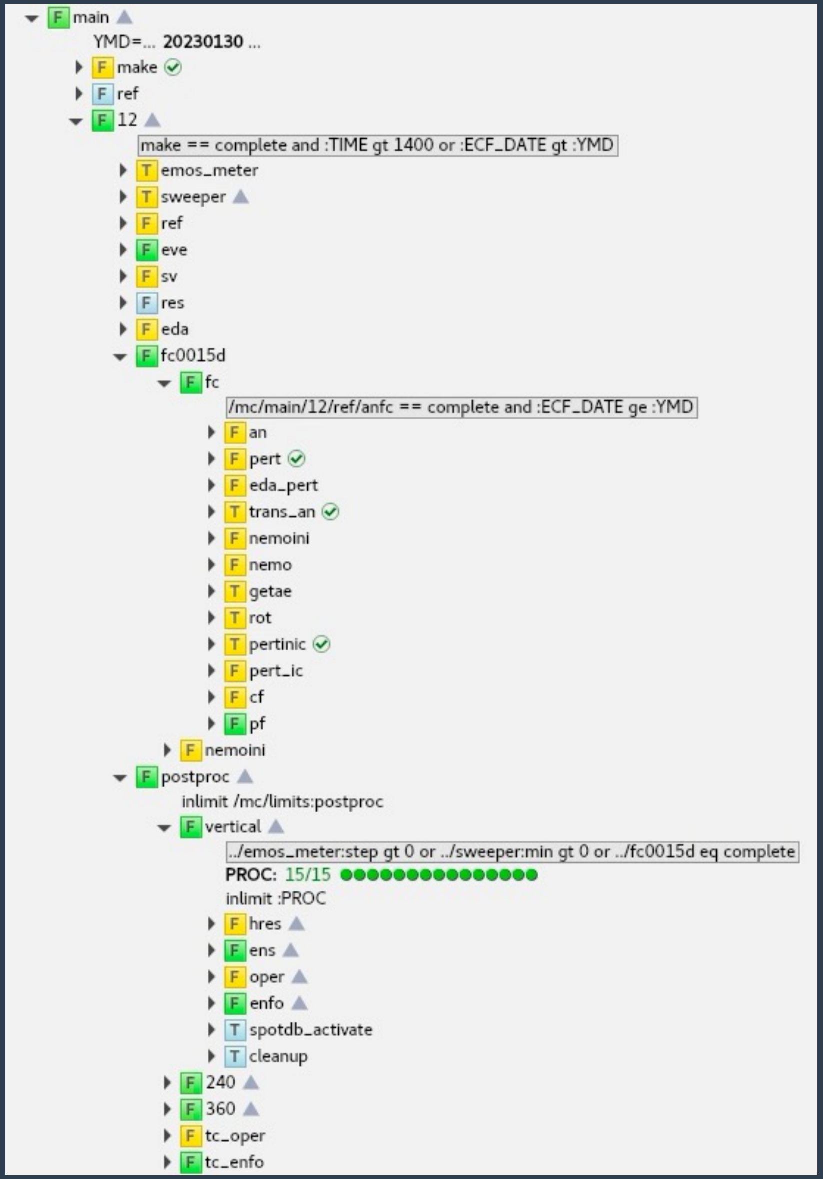
What it doesn't do

- Connect to remote systems
- Talk to specific queueing systems

What it does instead

- Run commands on the server host
- 3 main entry points: **submit**, **monitor**, **kill**

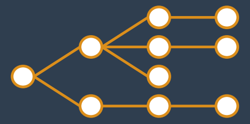
https://github.com/ecmwf/ecflow



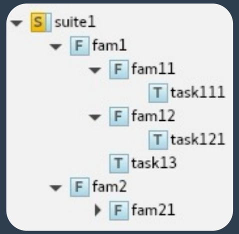
Actually running jobs on real target systems

- Simplest solution: `ssh $cluster sbatch $job`
  - Hard to make generic
  - Very limited in what it can do
- Write a shell script
  - Can do multiple actions
  - Hard to maintain when combinations accumulate
  - Everyone has their own
- Delegate to a submitter software
  - Can be made generic
  - Lots of flexibility
  - Software lifecycle (versioning, testing)

Workflow



ecFlow



???

ECF_JOB_CMD	%ECF_JOB% 1> %ECF_JOB
ECF_KILL_CMD	kill -15 %ECF_RID%
ECF_STATUS_CMD	ps --pid %ECF_RID% -f > %

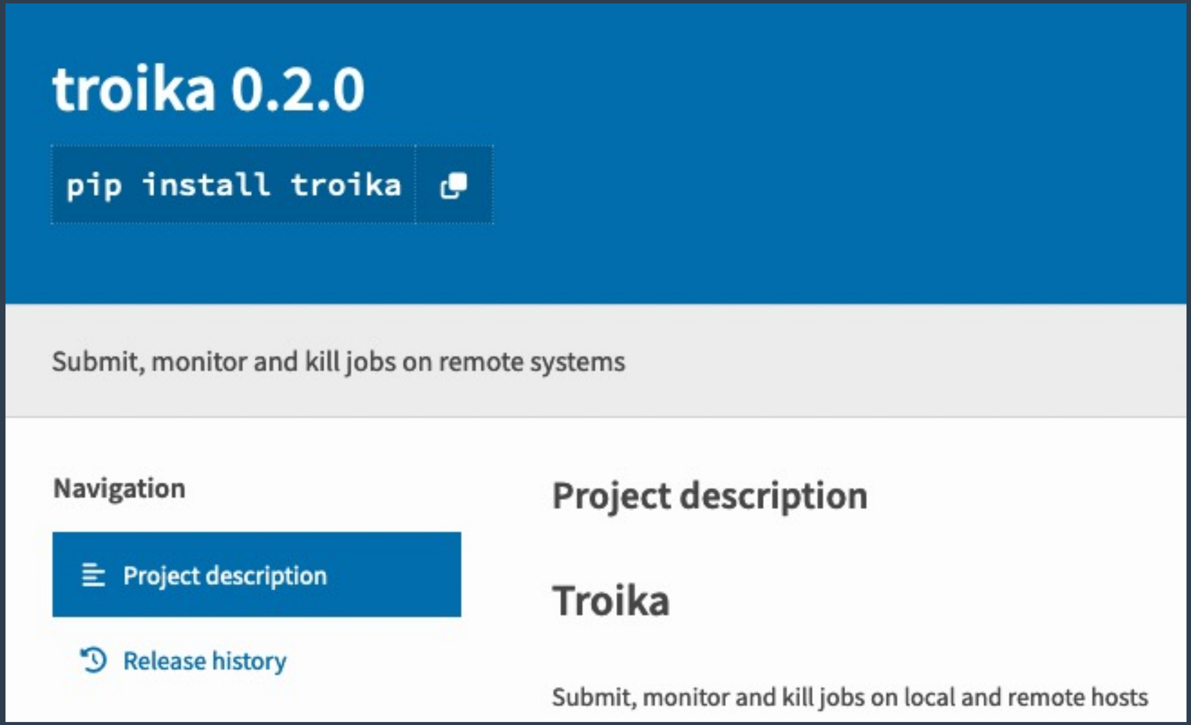
HPC

Enter Troika

- Troika fulfils the 3 actions: submit, monitor, kill
- Handles the connection to a remote system if needed
- Prepares the job script for submission to the queueing system
- Interacts with the queueing system
- Optionally, runs hooks at various points

Features

- Written in Python
- Fully configurable
- Extensible
  - Connection methods (local, SSH)
  - Queueing systems (direct, Slurm, PBS)
  - Hooks (create directories, copy files, …)



https://github.com/ecmwf/troika

Usage example

```
$ troika
  -v                                # verbose
  -n                                # dry run (don’t do anything)
  -c config.yml                     # default in $PREFIX/etc/troika.yml
  submit                            # could have been monitor, kill
  mycluster                         # host, as defined in the config
  -u user001                        # default: current user
  -o /scratch/user001/test.log      # output file (remote)
  test.job                          # path to the script (local)
```

```
INFO; Execute: 'ssh' 'user001@mycluster' 'mkdir' '-p' '/scratch/user001'
INFO; Execute: 'scp' 'test.job' 'user001@mycluster:/scratch/user001/test.job'
INFO; Execute: 'ssh' 'user001@mycluster' 'sbatch' '/scratch/user001/test.job'
INFO; Execute: 'ssh' 'user001@mycluster' 'mkdir' '-p' '/scratch/user001'
INFO; Execute: 'scp' 'test.job.submitlog' 'user001@mycluster:/scratch/user001/test.job.submitlog'
```

Configuration

- Each site is defined by
  - A name (`localhost`, `remote`, `slurm_cluster`, `pbs_cluster`)
  - A connection (`local`, `ssh`)
  - A type (`direct`, `slurm`, `pbs`)
  - Optional definitions, e.g. hooks
- Everything is configurable
- Simple command-line interface

```
troika submit -o myoutput.log slurm_cluster myjob.sh
troika monitor slurm_cluster myjob.sh
troika kill slurm_cluster myjob.sh
```

- Same commands regardless of the system

That's all good, but what about the script contents?

```yaml
sites:
    localhost:
        type: direct
        connection: local
    remote:
        type: direct
        connection: ssh
        host: remotebox
        copy_script: true
        at_startup: ["check_connection"]
    slurm_cluster:
        type: slurm
        connection: ssh
        host: remotecluster
        copy_script: true
        at_startup: ["check_connection"]
        pre_submit: ["create_output_dir"]
        at_exit: ["copy_submit_logfile"]
    pbs_cluster:
        type: pbs
        connection: ssh
        host: othercluster
        copy_script: true
        at_startup: ["check_connection"]
        pre_submit: ["create_output_dir"]
        at_exit: ["copy_submit_logfile"]
```

Directive translation

- Queueing systems usually understand directives to set options
- They are usually not interoperable
- We need some kind of system to translate them

Input

- Generic directives in the script
- Site-specific directives in the configuration
- Translation for complex mappings (plug-ins)

Output

- Site-specific generator
- Make the last few translations required (names of parameters, etc.)
- Add code if required (e.g. define environment variables)

Input:

```
#!/bin/bash
#TROIKA name=testjob
#TROIKA queue=fractional
#TROIKA mail_type=begin,end,fail
#TROIKA walltime=01:00:00
#TROIKA export_vars=all
#TROIKA threads_per_core=1
```

Slurm:

```
#!/bin/bash
#SBATCH --job-name=testjob
#SBATCH --qos=fractional
#SBATCH --mail-type=BEGIN,END,FAIL
#SBATCH --time=01:00:00
#SBATCH --export=ALL
#SBATCH --threads-per-core=1
#SBATCH --output=playground/test.log
#SBATCH --hint=nomultithread
```

PBS:

```
#!/bin/bash
#PBS -N testjob
#PBS -q fractional
#PBS -m bea
#PBS -L walltime=01:00:00
#PBS -V
#PBS -o playground/test.log
#PBS -j oe
```

Main components (extendable as plugins)

- Interaction with the queueing system
  - Parses its native directives if needed
  - Generates a job script
  - Runs the appropriate commands (could also use different APIs)
  - Keeps track of the submission (job ID) for monitor and kill
- Connection
  - Runs commands on the remote system
  - Copies files over if needed
- Hooks
  - At various points: at startup, pre-submit, post-kill, at exit
  - Perform extra actions: create directories, copy files, notify ecFlow…
- Translators
  - Controls the set of directives more finely (computed resources, conditional flags, etc)

Success story

- We've just switched to a new HPC system, with a new set of ecFlow server VMs
  - Much easier to rewrite a config file than a whole shell script with complex logic!
- Different users have different needs and different ways of working
  - We managed to bring them all together within a single tool
  - Operational workflows: tight control over the submitted scripts
  - Research workflows: lots of flexibility, with some complex logic to set directives
  - General purpose use: easy-to-use interface
- Troika now handles most of the jobs submitted to our HPC system
  - About half a million jobs per day!

What it will help us with

- Support our software development
  - Troika is not tied to ecFlow!
  - Control the CI/CD pipelines from GitHub workflows
  - Run specific tasks on our HPC, from GitHub runners
- Run our in-house workflows
  - Operational forecast
  - Research, support and general-purpose
  - Adapt to future HPC systems as they come
- Destination Earth
  - Support multiple HPC systems with minimal changes required

Where do we want to go from here?

- Features
  - Support more queueing systems
  - Enquire info from the queueing system (queues, partitions, etc)
  - Generic resource computation routines
- Improvements
  - Improve script generation
  - Widen test coverage
  - Provide packages (deb, RPM, etc)

Contributions welcome!

https://github.com/ecmwf/troika