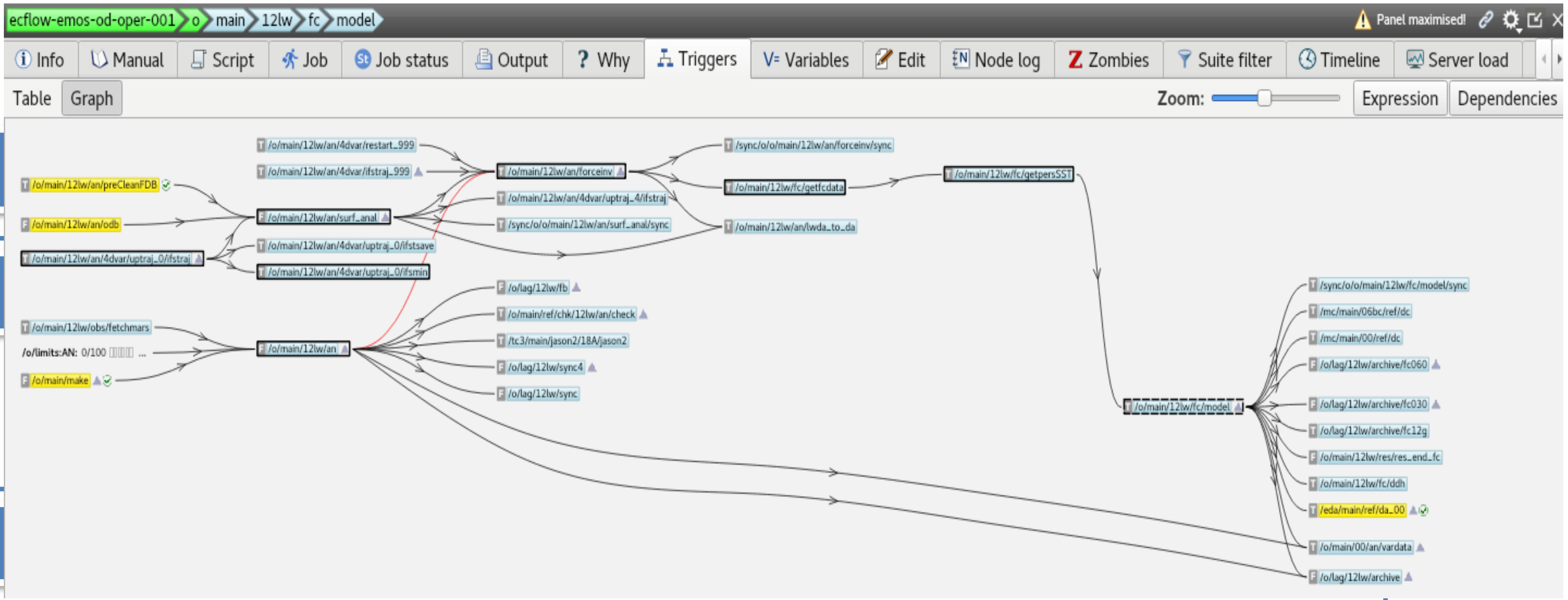
Online training course

ecFlow

map

ecFlow



Overview

Components

- ecflow_server
- ecflow_client
- ecflow_ui

Workflows

- Definition file
- Tasks wrappers, headers
- Python API

ecFlow in operation

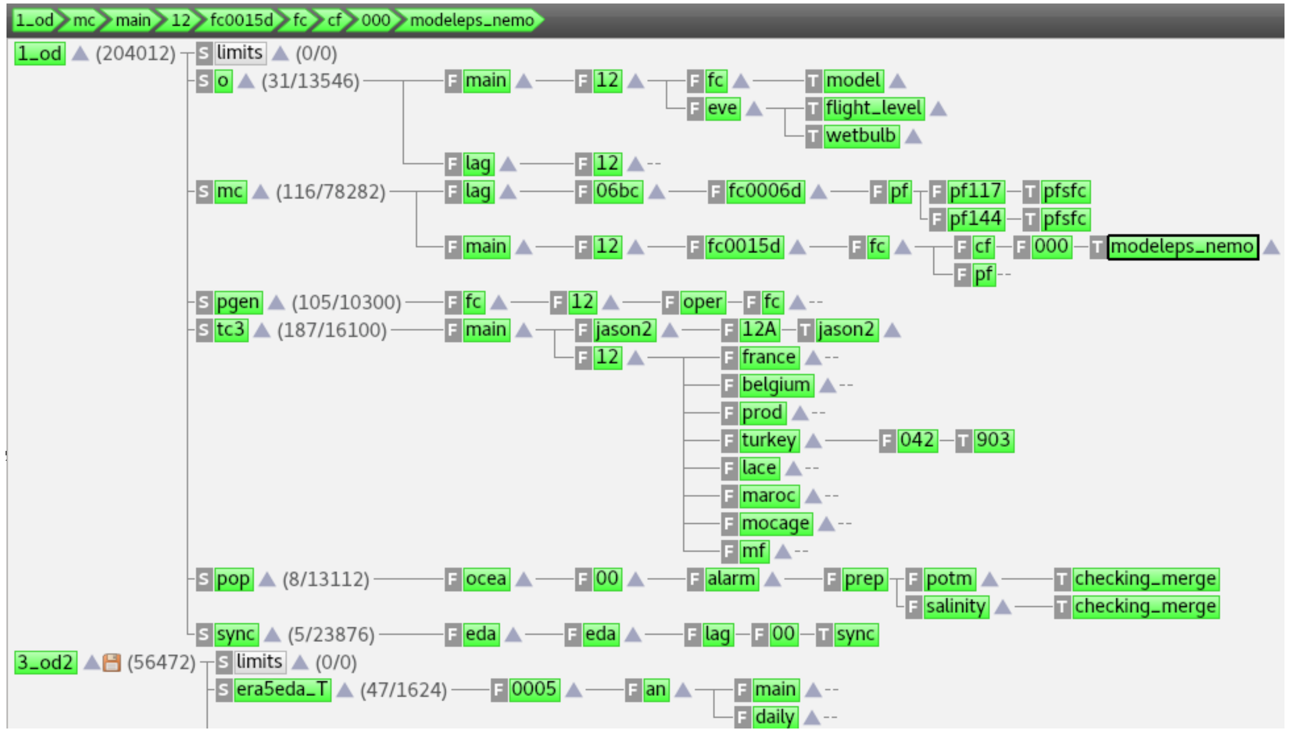
practical

Overview – What is ecFlow?

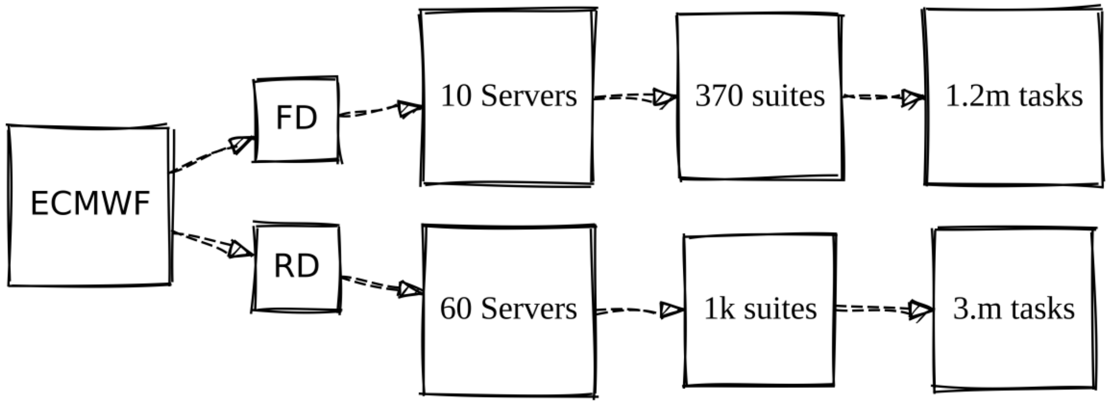
- **Distributed** workflows management,
- **Large** complex workflows,
- Tasks **scheduling**,
- Tasks **monitoring** and **supervision** (run, stop, check),
- Times, dates and **triggers-based** execution,
  - Advanced **dependencies** management,
  - With **hierarchical structure,**
  - With capability to **failover** and **recovery,**

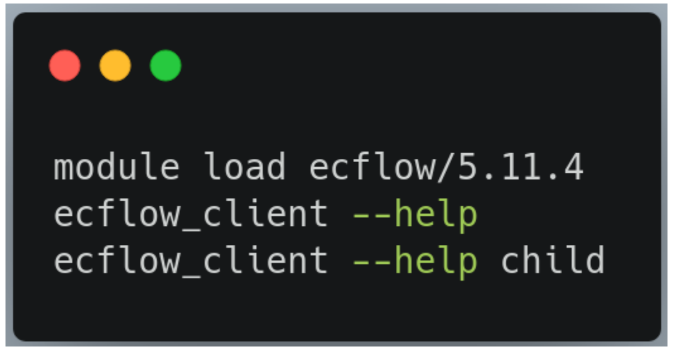
It is **Open-Source** apache license 2.0 and **extensible**,

can be integrated with HPC jobs schedulers (Slurm, …)



1_od > mc > main > 12 > fc0015d > fc > cf > 000 > modeleps_nemo
1_od (204012) S limits (0/0)
S o (31/13546) F main F 12 F fc T model
F eve T flight_level
T wetbulb
F lag F 12
S mc (116/78282) F lag F 06bc F fc0006d F pf F pf117 T pfsfc
F pf144 T pfsfc
F main F 12 F fc0015d F fc F cf F 000 T modeleps_nemo
F pf
S pgen (105/10300) F fc F 12 F oper F fc
S tc3 (187/16100) F main F jason2 F 12A T jason2
F 12 F france
F belgium
F prod
F turkey F 042 T 903
F lace
F maroc
F mocage
F mf
S pop (8/13112) F ocea F 00 F alarm F prep F potm T checking_merge
F salinity T checking_merge
S sync (5/23876) F eda F eda F lag F 00 T sync
3_od2 (56472) S limits (0/0)
S era5eda_T (47/1624) F 0005 F an F main
F daily

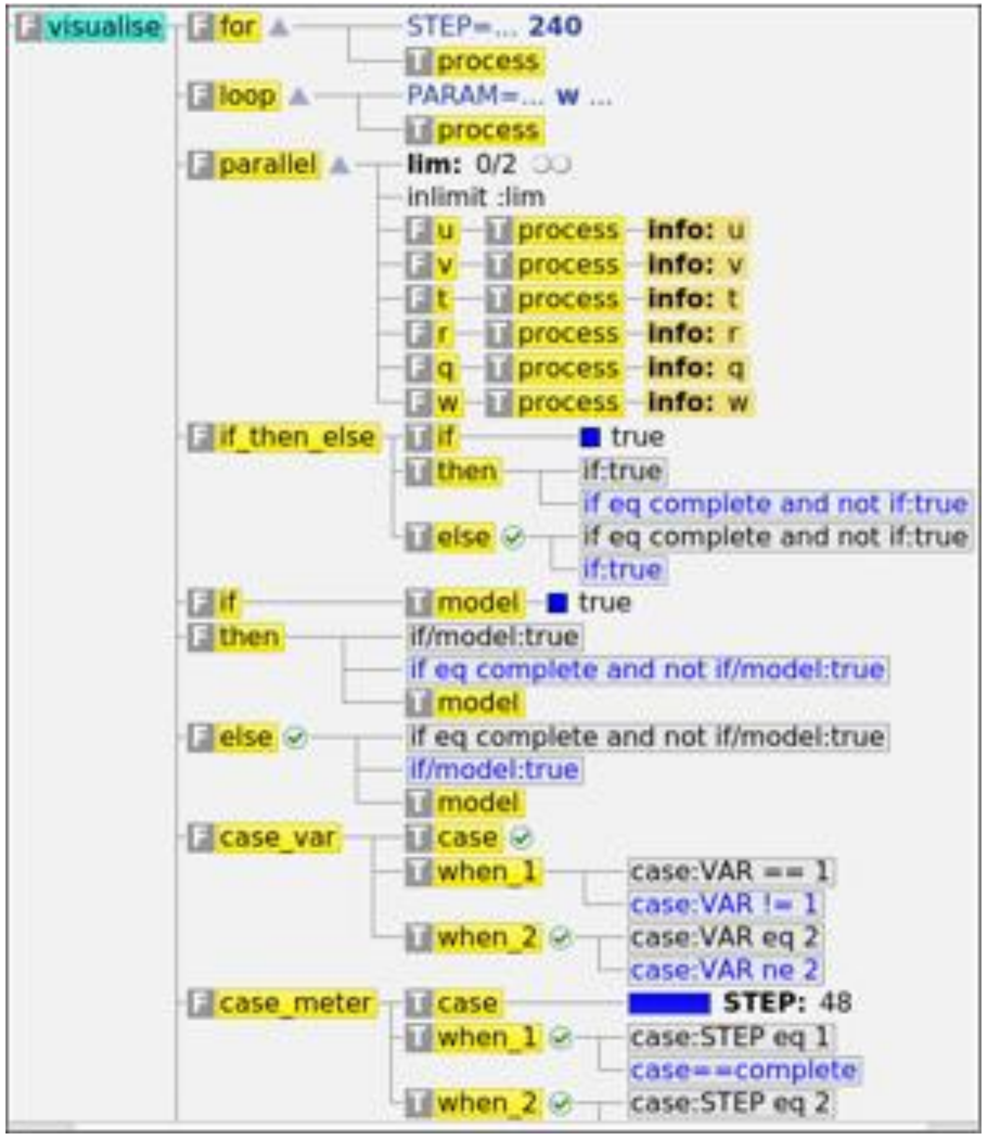




```
module load ecflow/5.11.4
ecflow_client --help
ecflow_client --help child
```

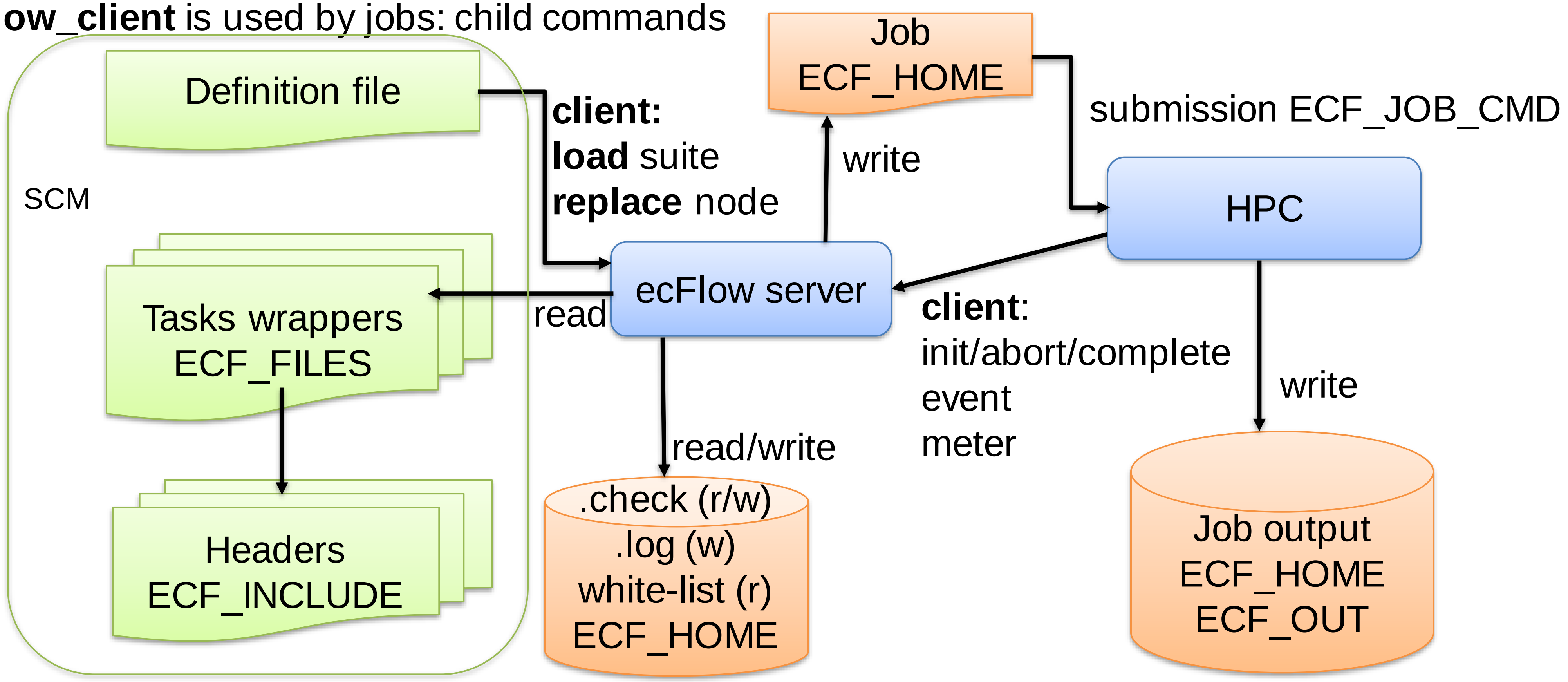
Overview – What is ecFlow?

- ecFlow is **job language agnostic**: bash, ksh, python, …
- ecFlow is **target agnostic** (HPC, cluster, localhost)
- Can use **Troika** submitter
- ecFlow is a **template engine (**JIT jobs creation)
- ecFlow used in **pure monitoring mode**
- **Sources** http://github.com/ecmwf/ecFlow
- https://ecflow.readthedocs.io/en/latest/index.html
  - **Documentation** and **tutorial**
- A Server, a client, a GUI, python API, REST API, UDP
- A **visual** programming language
- **Collaboration** between developer, analyst, operators



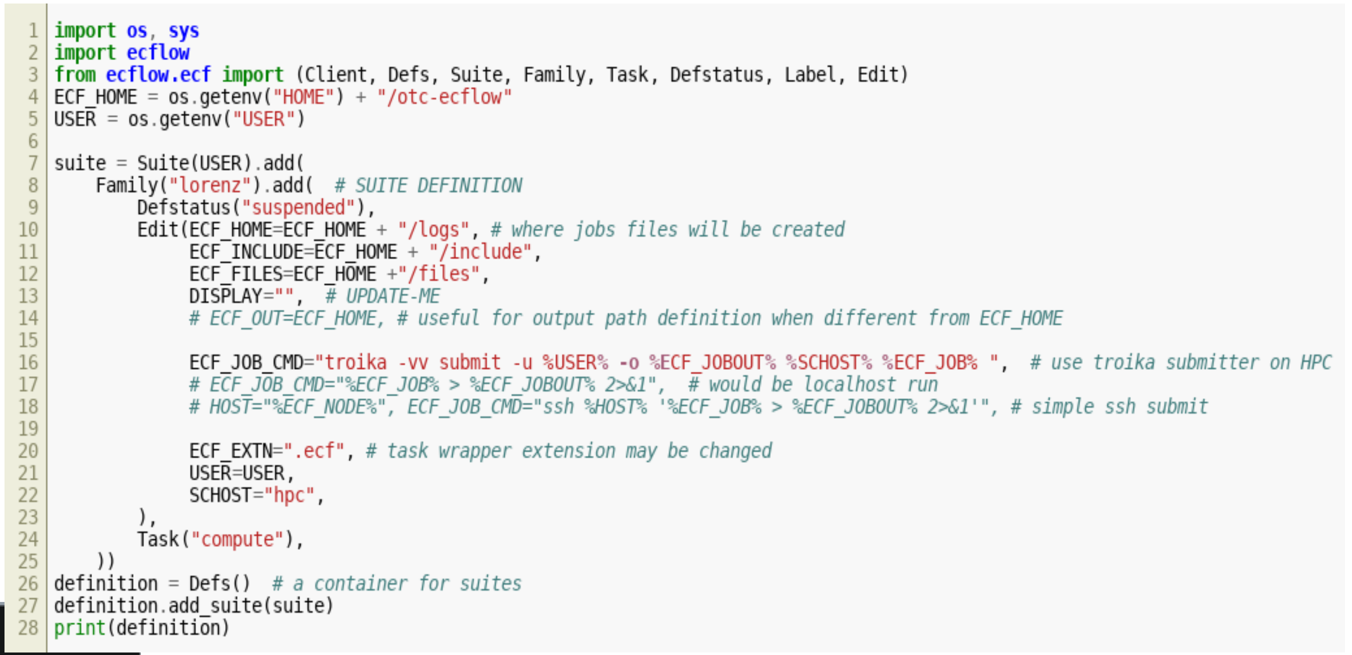
ecFlow components

- **definition file**, **tasks wrappers**, **headers**
- **ecflow_server**
- **ecflow_client** is used by users
- **ecflow_client** is used by jobs: child commands
- **REST-API**
- **ecflow_udp, ecflow_udp_client**
- **Python-API**

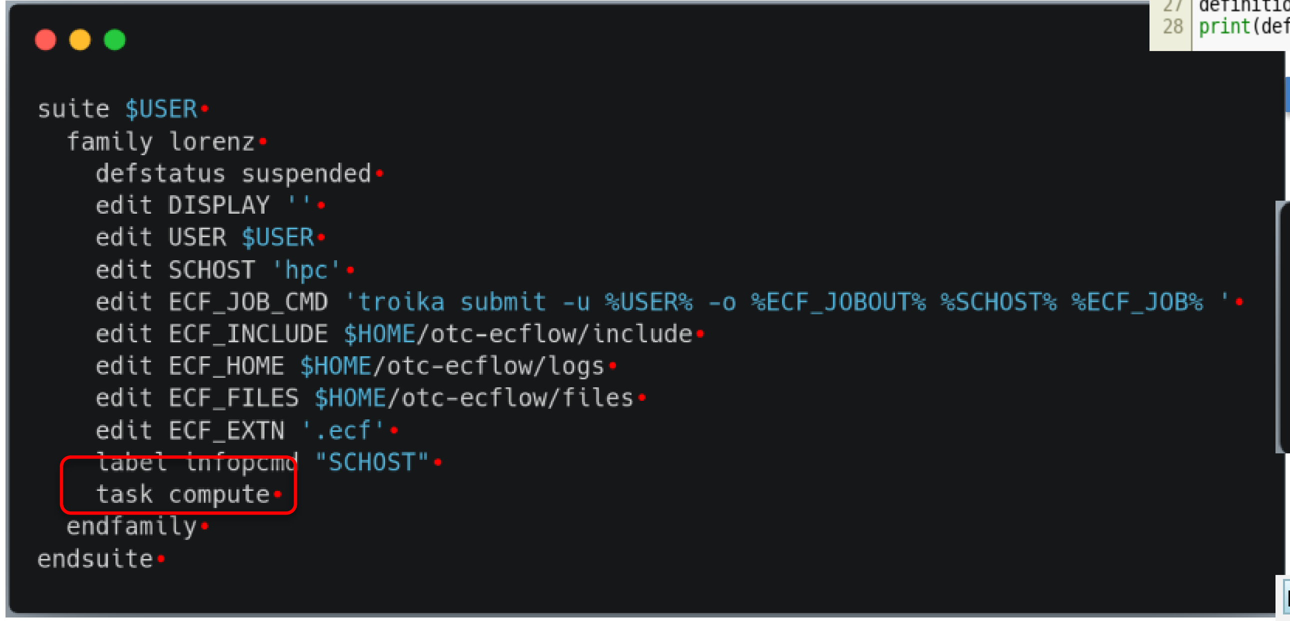


ecFlow definition file

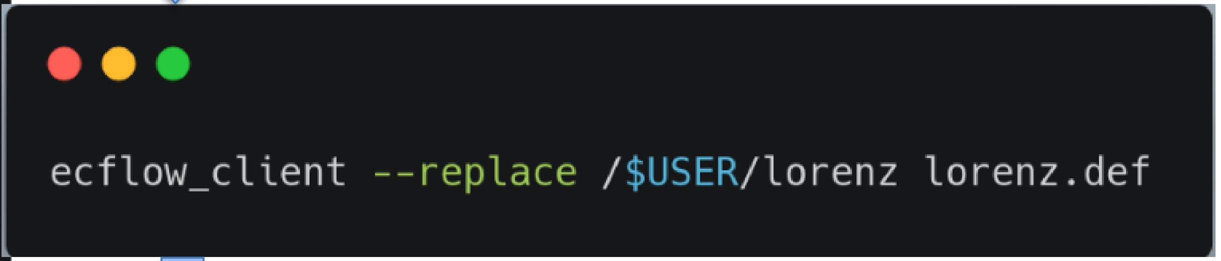
- A text file to describe the tasks and their relations
- **Nodes**: suite, family, task
- **Attributes**: event, meter, label: to receive update
  - clock complete cron date day defs_state defstatus **edit** inlimit late limit repeat time today trigger zombie



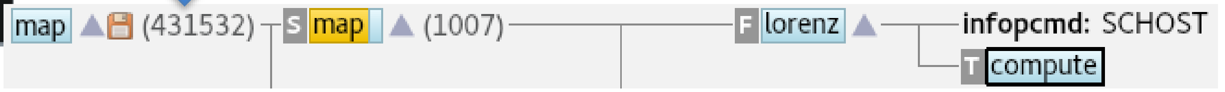
```python
import os, sys
import ecflow
from ecflow.ecf import (Client, Defs, Suite, Family, Task, Defstatus, Label, Edit)
ECF_HOME = os.getenv("HOME") + "/otc-ecflow"
USER = os.getenv("USER")

suite = Suite(USER).add(
    Family("lorenz").add(  # SUITE DEFINITION
        Defstatus("suspended"),
        Edit(ECF_HOME=ECF_HOME + "/logs", # where jobs files will be created
             ECF_INCLUDE=ECF_HOME + "/include",
             ECF_FILES=ECF_HOME +"/files",
             DISPLAY="",  # UPDATE-ME
             # ECF_OUT=ECF_HOME, # useful for output path definition when different from ECF_HOME

             ECF_JOB_CMD="troika -vv submit -u %USER% -o %ECF_JOBOUT% %SCHOST% %ECF_JOB% ",  # use troika submitter on HPC
             # ECF_JOB_CMD="%ECF_JOB% > %ECF_JOBOUT% 2>&1",  # would be localhost run
             # HOST="%ECF_NODE%", ECF_JOB_CMD="ssh %HOST% '%ECF_JOB% > %ECF_JOBOUT% 2>&1'", # simple ssh submit

             ECF_EXTN=".ecf", # task wrapper extension may be changed
             USER=USER,
             SCHOST="hpc",
        ),
        Task("compute"),
    ))
definition = Defs()  # a container for suites
definition.add_suite(suite)
print(definition)
```



```
suite $USER
  family lorenz
    defstatus suspended
    edit DISPLAY ''
    edit USER $USER
    edit SCHOST 'hpc'
    edit ECF_JOB_CMD 'troika submit -u %USER% -o %ECF_JOBOUT% %SCHOST% %ECF_JOB% '
    edit ECF_INCLUDE $HOME/otc-ecflow/include
    edit ECF_HOME $HOME/otc-ecflow/logs
    edit ECF_FILES $HOME/otc-ecflow/files
    edit ECF_EXTN '.ecf'
    label infopcmd "SCHOST"
    task compute
  endfamily
endsuite
```



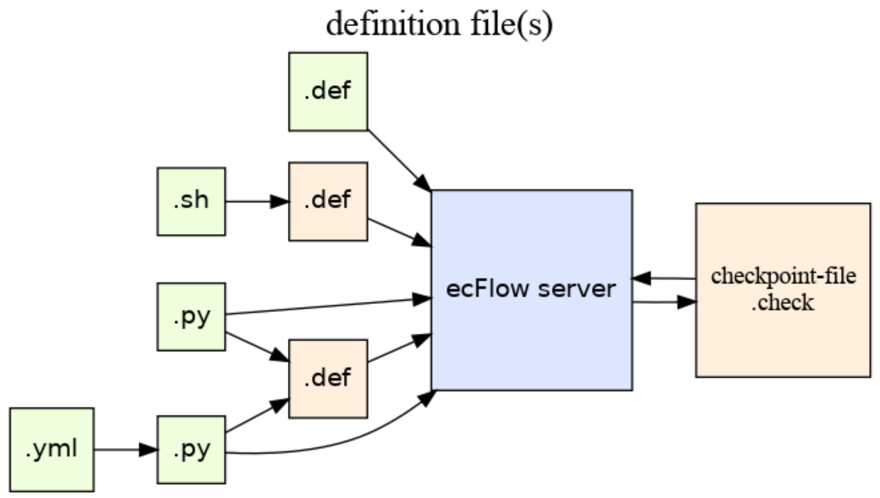
```
ecflow_client --replace /$USER/lorenz lorenz.def
```

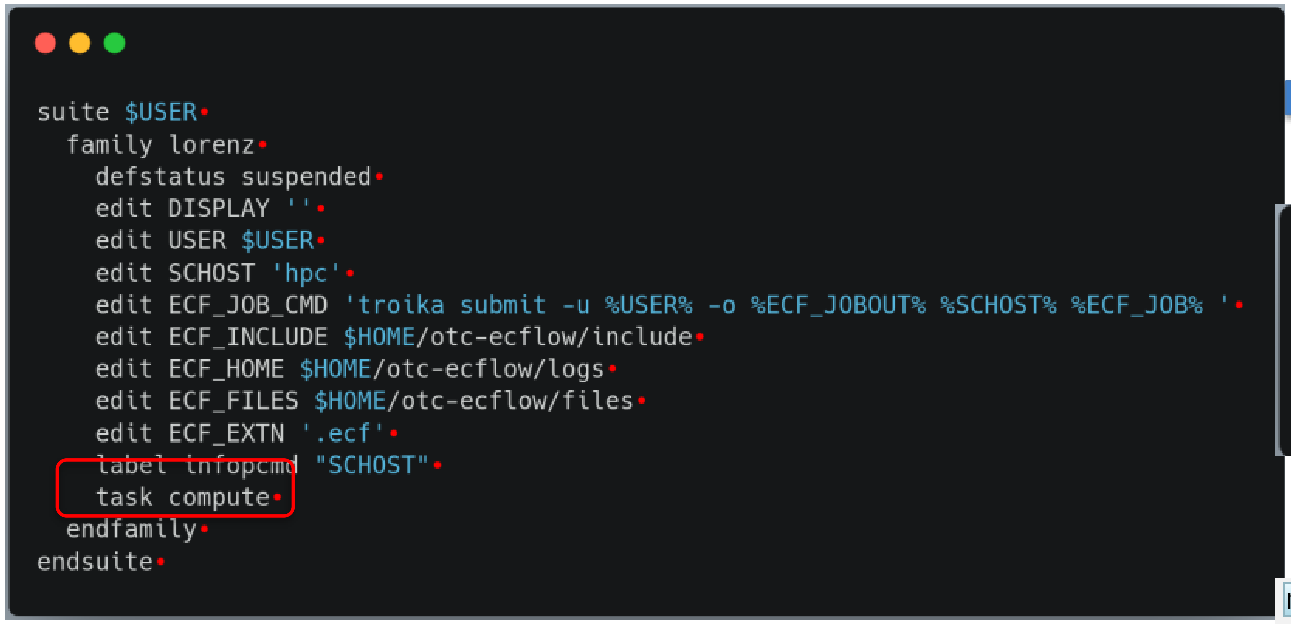


```
map (431532) ─ S map (1007) ──── F lorenz ── infopcmd: SCHOST
                                          └─ T compute
```

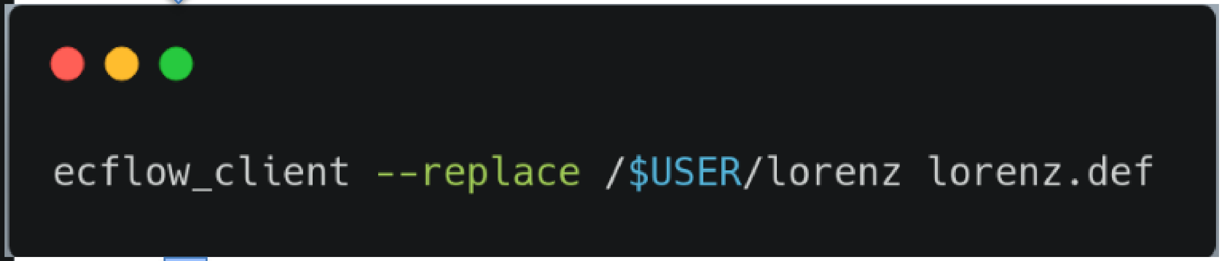
ecFlow definition file

- Only **consistency** is required in the definition file
- A suite can be defined from multiple definition files
- A Suite can be defined **incrementally**
- Once loaded, a node **can be moved** with the GUI
- Keep it simple ☺

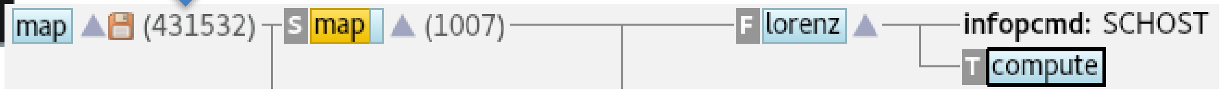




```
suite $USER
  family lorenz
    defstatus suspended
    edit DISPLAY ''
    edit USER $USER
    edit SCHOST 'hpc'
    edit ECF_JOB_CMD 'troika submit -u %USER% -o %ECF_JOBOUT% %SCHOST% %ECF_JOB% '
    edit ECF_INCLUDE $HOME/otc-ecflow/include
    edit ECF_HOME $HOME/otc-ecflow/logs
    edit ECF_FILES $HOME/otc-ecflow/files
    edit ECF_EXTN '.ecf'
    label infopcmd "SCHOST"
    task compute
  endfamily
endsuite
```



```
ecflow_client --replace /$USER/lorenz lorenz.def
```



```
map (431532) ─ S map (1007) ─── F lorenz ─ infopcmd: SCHOST
                                          └ T compute
```

ECMWF EUROPEAN CENTRE FOR MEDIUM-RANGE WEATHER FORECASTS

ecFlow: definition-file, checkpoint-file, nodes + attributes

Checkpoint-file written by ecflow_server

- a definition file
- defs, enddef, history additional keywords
- recent values for states and variables, next run time in comment

Nodes:

- suite, family, task
- (endsuite, endfamily, endtask)

Attributes can be classified in multiple ways:

- Active/passive (task requeued)
- Related with child command or not
- Behavioural: defstatus, complete

Looping

- repeat, cron, time, today, date, day, defstatus, autocancel

Scheduling attributes

- trigger, complete, limit, inlimit

Informational attributes

- label, zombie

used in jobs

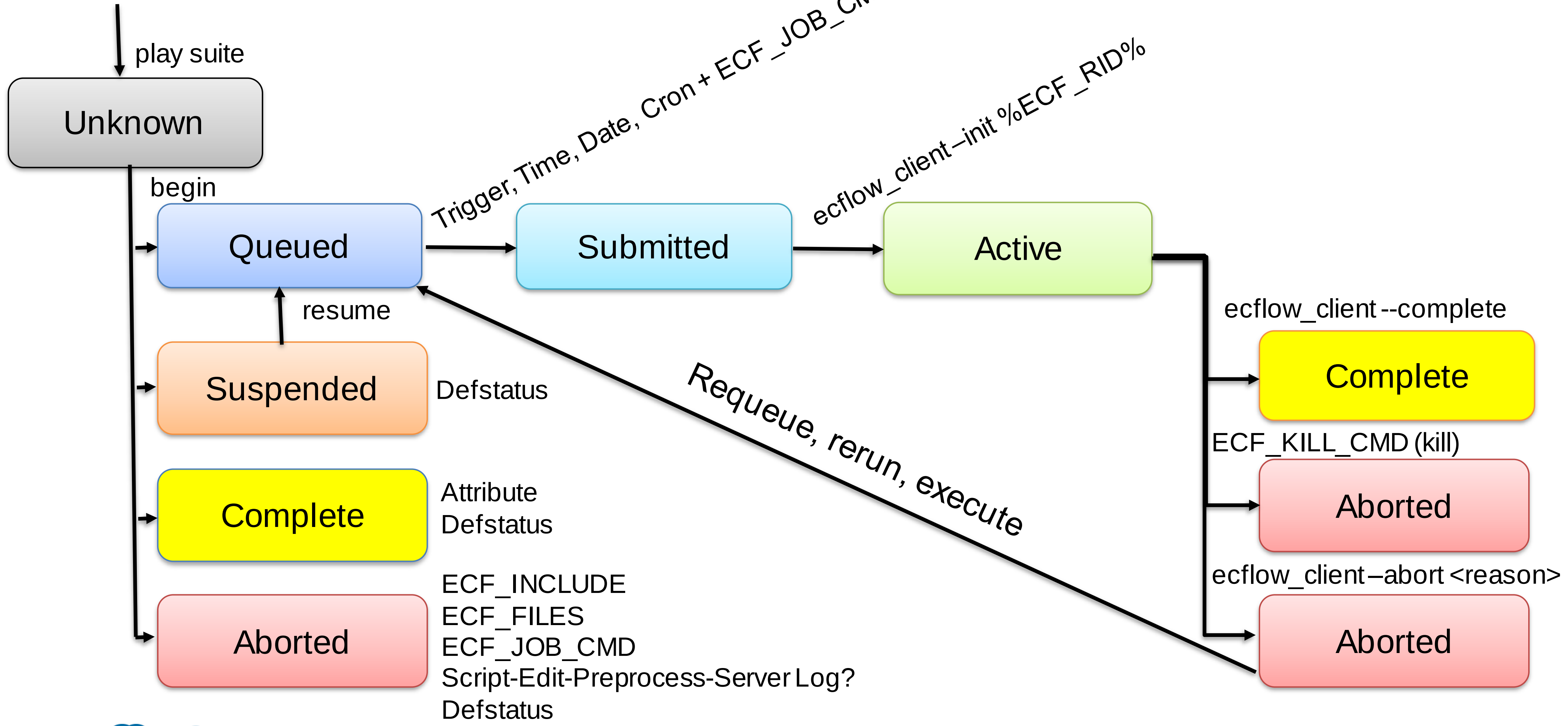
- edit (variable)

used in trigger

- Node status, variable, event, meter, limit, late

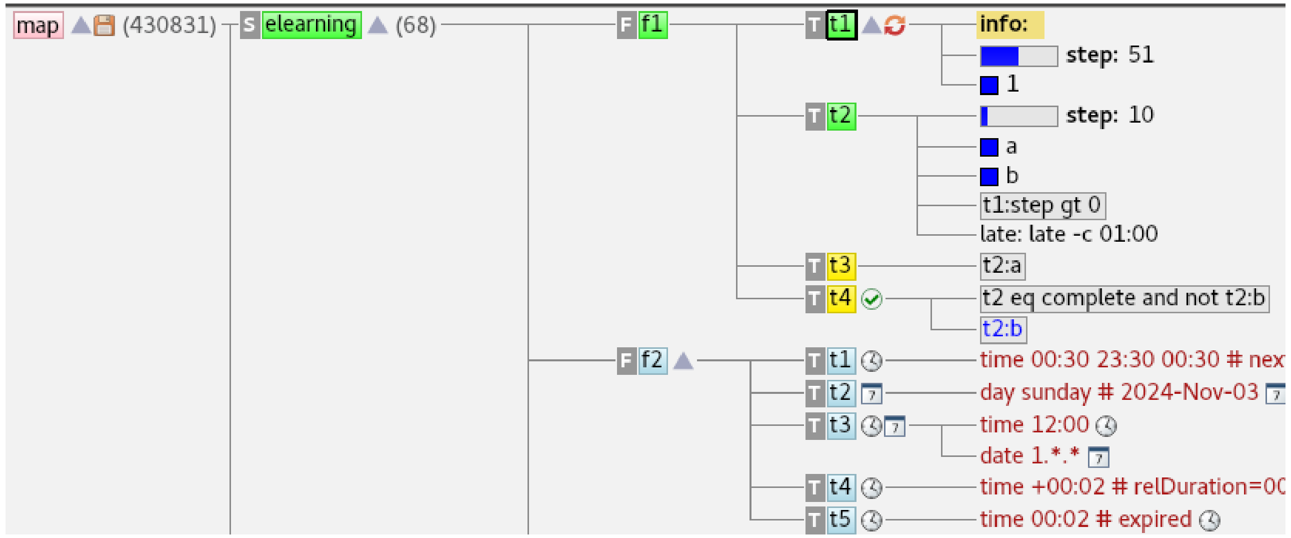
ecFlow: Status

- Task v Node: ecflow_client v inheritance



ecFlow: Inheritance status v variables v dependencies

- Status inheritance is bubbling up
- A suite or family node reflects most important status
- server node status can be
  - Halted: accept only user commands
  - Shutdown: accept user and child commands
  - Running: additionally, jobs can be submitted



```
map (430831) ─ S elearning (68) ─ F f1 ─ T t1 ─ info:
                                               ├ step: 51
                                               └ 1
                                   ├ T t2 ─ step: 10
                                          ├ a
                                          ├ b
                                          ├ t1:step gt 0
                                          └ late: late -c 01:00
                                   ├ T t3 ─ t2:a
                                   └ T t4 ─ t2 eq complete and not t2:b
                                          └ t2:b
                              └ F f2 ─ T t1 ─ time 00:30 23:30 00:30 # nex
                                     ├ T t2 ─ day sunday # 2024-Nov-03
                                     ├ T t3 ─ time 12:00
                                            └ date 1.*.*
                                     ├ T t4 ─ time +00:02 # relDuration=00
                                     └ T t5 ─ time 00:02 # expired
```

- variables inheritance is top down
  - A Vvriable can be redefined lower in the tree
  - Lowest value prevail for jobs creation

- dependencies can be defined on any node
  - Trigger, complete, time, date, cron attribute
  - All conditions must be true to create a job
  - High dependency will hide the lower
  - Trigger, complete attribute are instantaneous
  - Date, time, cron attribute have memory

ecFlow: Tasks wrappers / Tasks headers

key variables

- ECF_EXTN: wrapper extension .ecf .sh .py
- ECF_FILES: wrappers location (r)
- ECF_INCLUDE : headers location (r)
- ECF_HOME: where .job are created (w)

Tasks wrappers

- a template script
- describe generic or specific work to do

Tasks headers

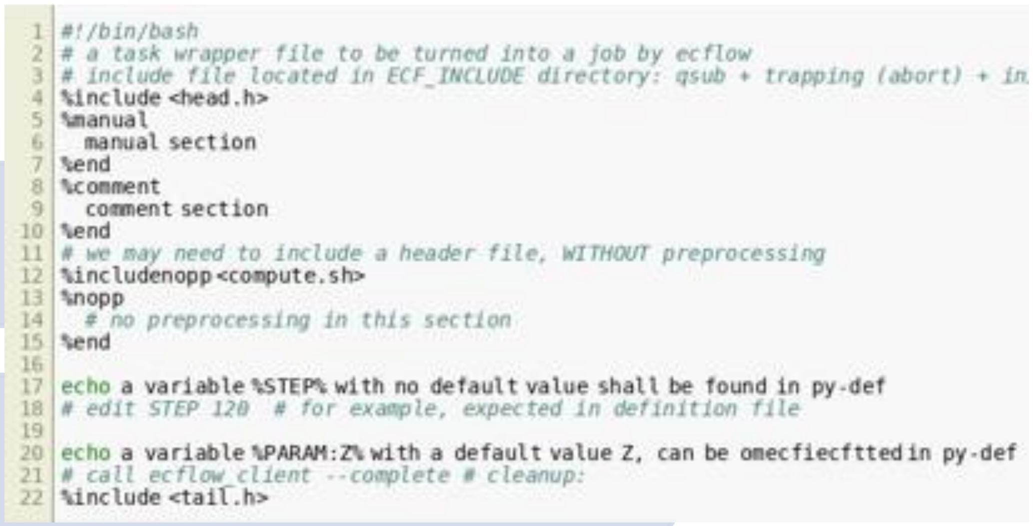
- head.h / qsub.h / tail.h
- %include <%QSUB_H:qsub.h%>

ECF_MICRO % character: variable/block/keyword

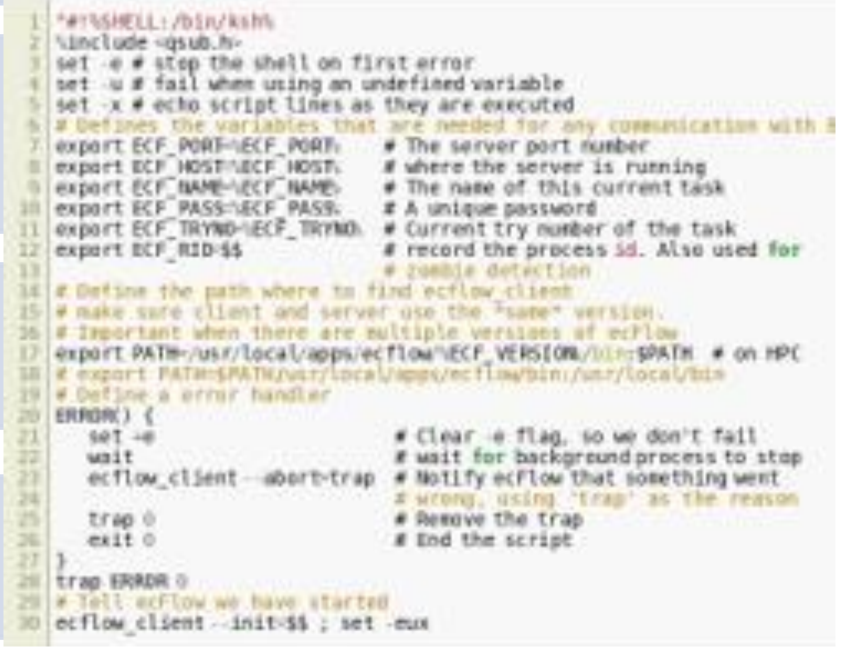
- %VARIABLE:default_value%
- manual, nopp, comment,
- include, includenopp
- global scale or locally in the template script: %ecf_micro $

Tolerance for failures (hardware and software):

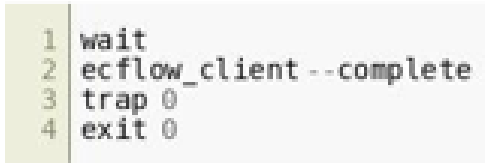
- ECF_TRIES: number of automatic rerun
- ECF_TRYNO: job instance number
- Watchdog task to handle known issues



```
#!/bin/bash
# a task wrapper file to be turned into a job by ecflow
# include file located in ECF_INCLUDE directory: qsub + trapping (abort) + in
%include <head.h>
%manual
  manual section
%end
%comment
  comment section
%end
# we may need to include a header file, WITHOUT preprocessing
%includenopp <compute.sh>
%nopp
  # no preprocessing in this section
%end

echo a variable %STEP% with no default value shall be found in py-def
# edit STEP 120  # for example, expected in definition file

echo a variable %PARAM:Z% with a default value Z, can be omecfiecftted in py-def
# call ecflow_client --complete # cleanup:
%include <tail.h>
```



```
#!%SHELL:/bin/ksh%
%include <qsub.h>
set -e # stop the shell on first error
set -u # fail when using an undefined variable
set -x # echo script lines as they are executed
# Defines the variables that are needed for any communication with E
export ECF_PORT=%ECF_PORT%    # The server port number
export ECF_HOST=%ECF_HOST%    # where the server is running
export ECF_NAME=%ECF_NAME%    # The name of this current task
export ECF_PASS=%ECF_PASS%    # A unique password
export ECF_TRYNO=%ECF_TRYNO%  # Current try number of the task
export ECF_RID=$$             # record the process id. Also used for
                              # zombie detection
# Define the path where to find ecflow_client
# make sure client and server use the *same* version.
# Important when there are multiple versions of ecFlow
export PATH=/usr/local/apps/ecflow/%ECF_VERSION%/bin:$PATH  # on HPC
# export PATH=$PATH/usr/local/apps/ecflow/bin:/usr/local/bin
# Define a error handler
ERROR() {
   set +e                       # Clear -e flag, so we don't fail
   wait                         # wait for background process to stop
   ecflow_client --abort=trap   # Notify ecFlow that something went
                                # wrong, using 'trap' as the reason
   trap 0                       # Remove the trap
   exit 0                       # End the script
}
trap ERROR 0
# Tell ecFlow we have started
ecflow_client --init=$$ ; set -eux
```

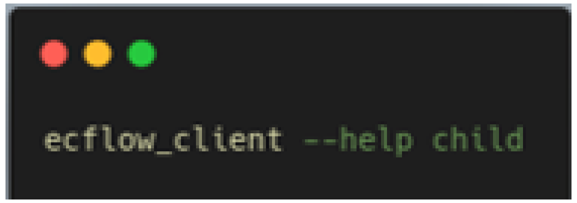


```
wait
ecflow_client --complete
trap 0
exit 0
```

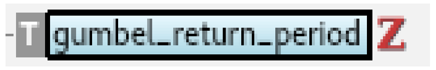
ecFlow: child commands

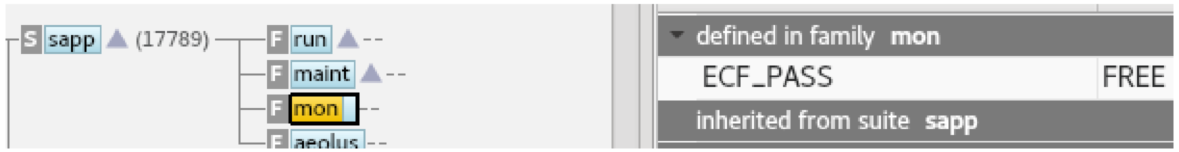
- ecflow_client called from a job

- 4 variables:
  - ECF_NAME: path for the node in the definition tree
  - ECF_HOST,
  - ECF_PORT,
  - ECF_PASS:
    - unique pseudorandom key for current job.
    - Zombie flag is raised when incorrect.
    - set to FREE to rescue a child, or in monitoring mode



```
ecflow_client --help child
```





defined in family mon

ECF_PASS	FREE

inherited from suite sapp

Update status:
- init <jid>
- complete
- abort <reason>

Update attribute:
- meter <name> <value>
- event <name>
- label <name> <msg>

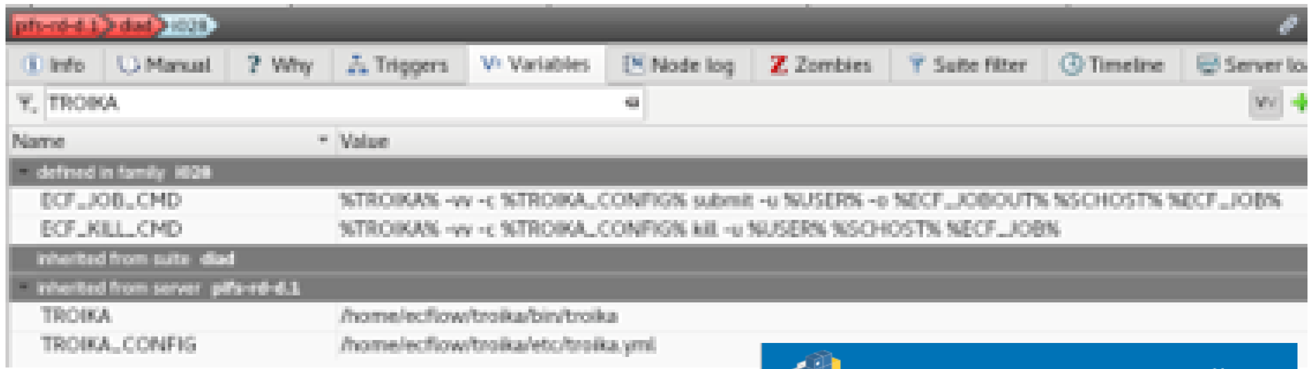
Embedded trigger:
- wait <expression>

Write into server log:
- log <msg>

Get an item from a list: queue
- queue <name> <list> # def-file

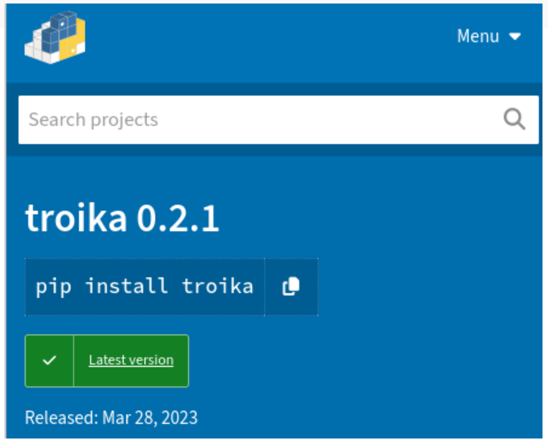
ecFlow: troika, a dedicated jobs submitter

- Troika is open-source, developed at ECMWF
- A system description with a yml file
- To Interact with remote queueing system
- Extra jobs tuning (MEM, THREADS, NPES)
- Run hooks (pre / post action)
- Allow deterministic + load balancing submit
- Troika is used in FD/RD/CD/MS workflows
- Extensible: connections (ssh, local), queuing system (Slurm, PBS, …), hooks
- https://github.com/ecmwf/troika



TROIKA

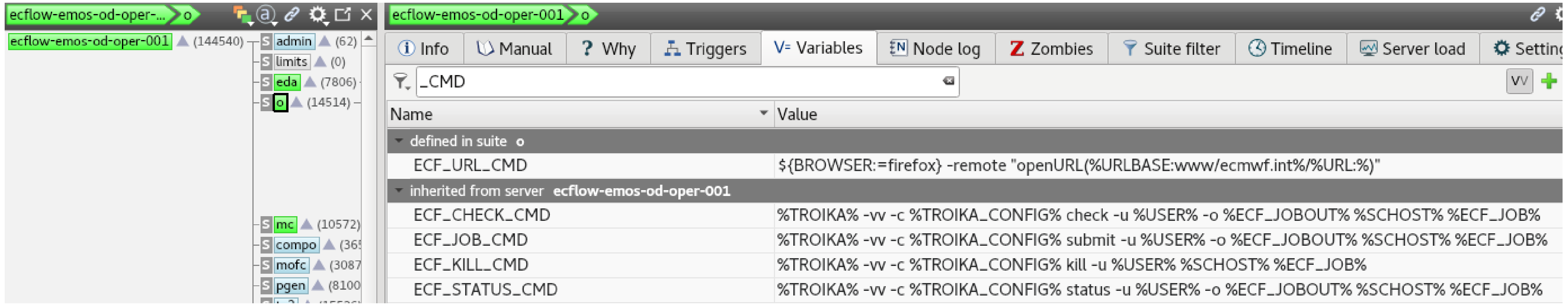
Name	Value
defined in family i028	
ECF_JOB_CMD	%TROIKA% -vv -c %TROIKA_CONFIG% submit -u %USER% -o %ECF_JOBOUT% %SCHOST% %ECF_JOB%
ECF_KILL_CMD	%TROIKA% -vv -c %TROIKA_CONFIG% kill -u %USER% %SCHOST% %ECF_JOB%
inherited from suite dad	
inherited from server pifs-rd-d.1	
TROIKA	/home/ecflow/troika/bin/troika
TROIKA_CONFIG	/home/ecflow/troika/etc/troika.yml



troika 0.2.1

```
pip install troika
```

Released: Mar 28, 2023

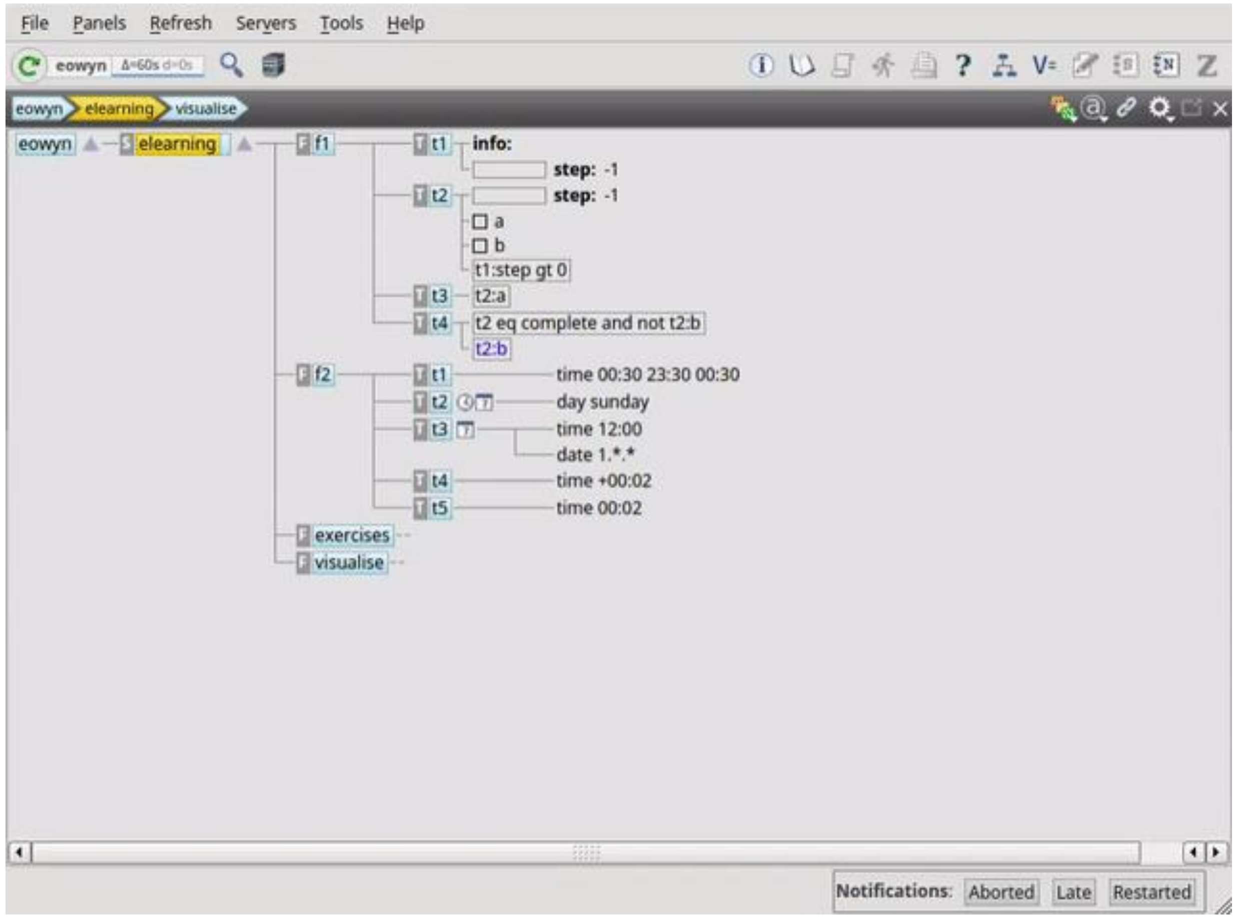


_CMD

Name	Value
defined in suite o	
ECF_URL_CMD	${BROWSER:=firefox} -remote "openURL(%URLBASE:www/ecmwf.int%/%URL:%)"
inherited from server ecflow-emos-od-oper-001	
ECF_CHECK_CMD	%TROIKA% -vv -c %TROIKA_CONFIG% check -u %USER% -o %ECF_JOBOUT% %SCHOST% %ECF_JOB%
ECF_JOB_CMD	%TROIKA% -vv -c %TROIKA_CONFIG% submit -u %USER% -o %ECF_JOBOUT% %SCHOST% %ECF_JOB%
ECF_KILL_CMD	%TROIKA% -vv -c %TROIKA_CONFIG% kill -u %USER% %SCHOST% %ECF_JOB%
ECF_STATUS_CMD	%TROIKA% -vv -c %TROIKA_CONFIG% status -u %USER% -o %ECF_JOBOUT% %SCHOST% %ECF_JOB%

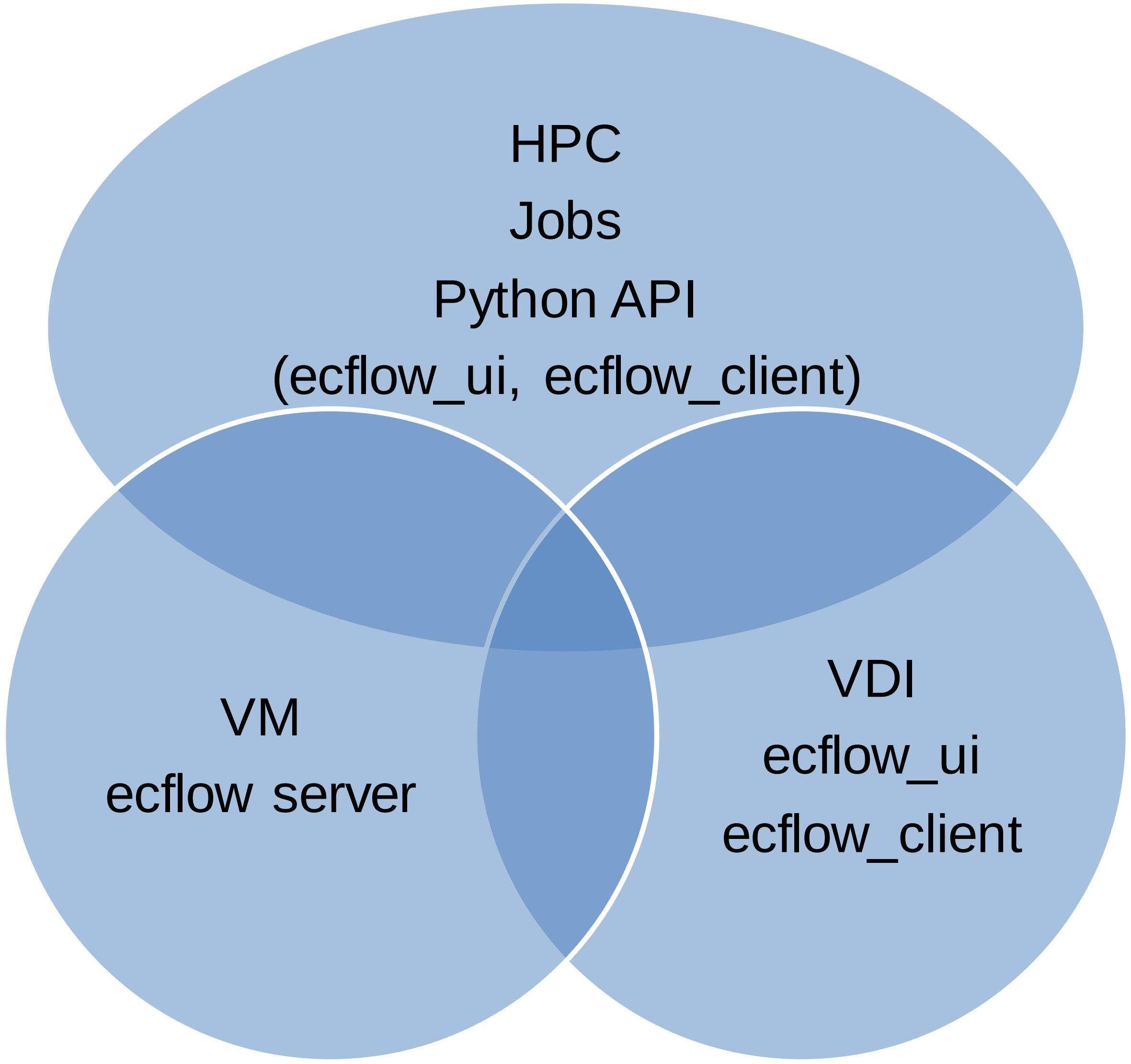
ecFlow: Alias

- Interactive way to fix, test, debug
- A node created dynamically from ecflow_ui
  - Edit, click "Submit as alias", Submit
  - A file is created near the task job files (ECF_HOME):
    - AliasN.usrM, alias number, job occurrence number
    - It can be run multiple times
  - It can be deleted directly from ecflow_ui menu



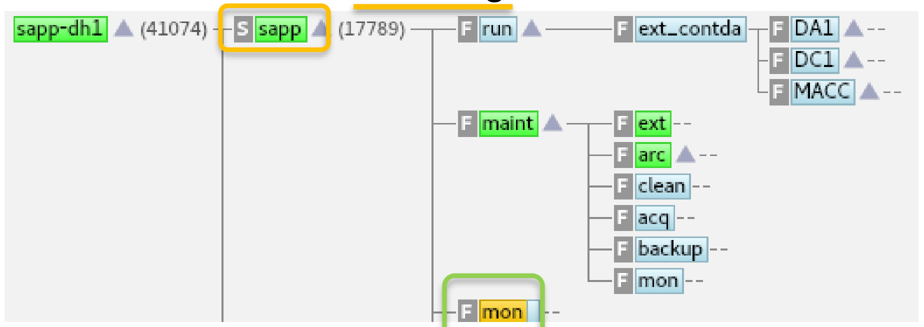
ecFlow: users use case

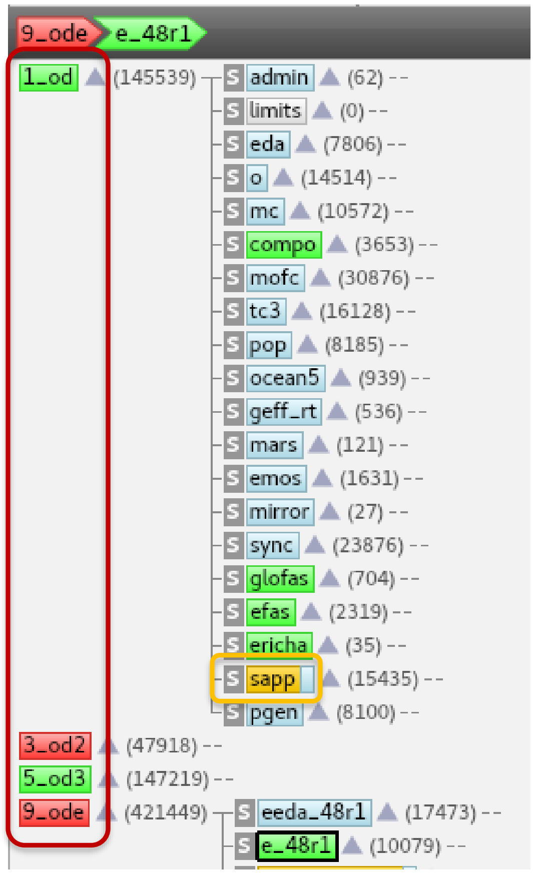
- ecFlow server is hosted in a dedicated VM
  - ping ecflow-gen-${USER}-001
- ecflow_ui is run on VDI (or laptop, or HPC)
- Jobs are submitted on HPC
- $HOME is common between VM, HPC, VDI
  - .check, .log under $HOME/ecflow_servers
  - File ecf.lists to grant or refrain access (rw/r/none)

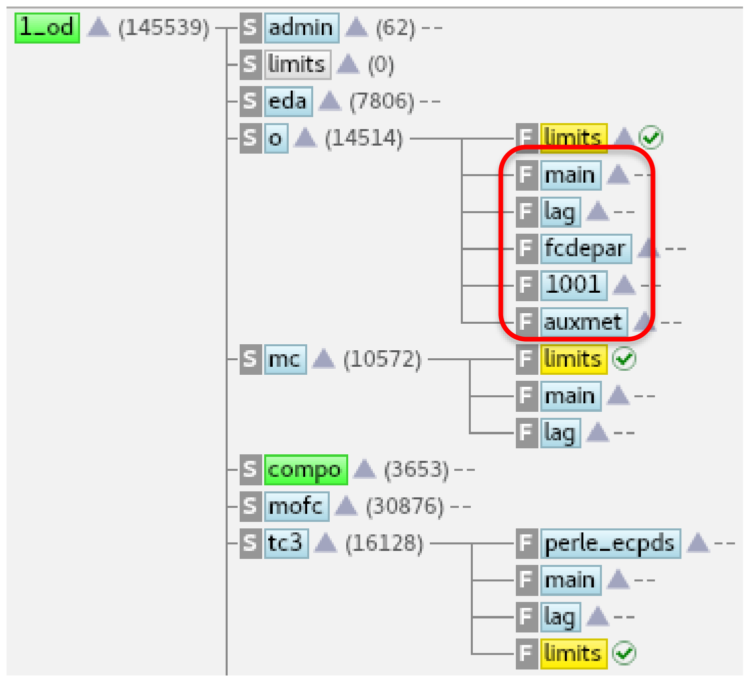


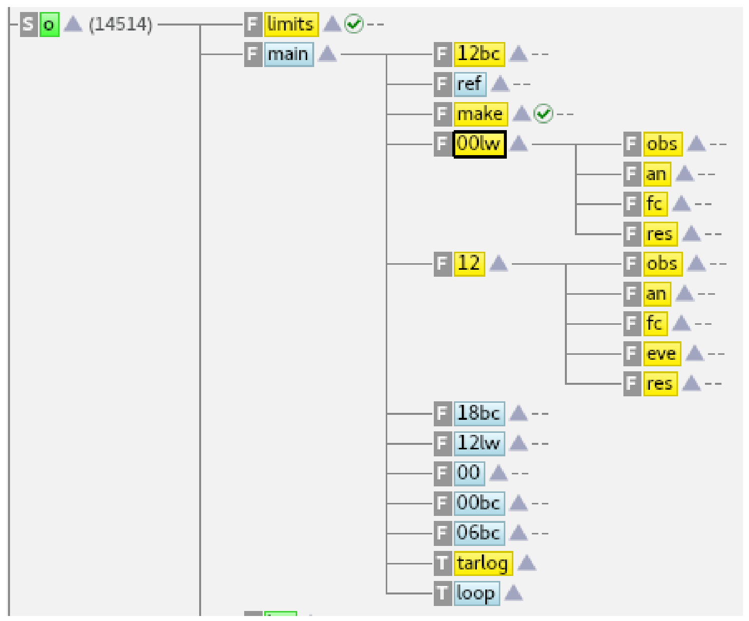
ecFlow in Operation: EMOS operational servers

- EMOS servers:
  - Display criticality: 1_od, 3_od2, 5_od3, 9_ode
- Suite structure reflects on-call criticality:
  - Main: critical path
  - Lag: archive, slow postprocessing
  - Other postprocessing families
  - Inner vs outer watchdog
- Suspended suites may be lively entities:
  - ECF_PASS: FREE, monitoring mode
  - Mirror suite: reflecting a suite on another server

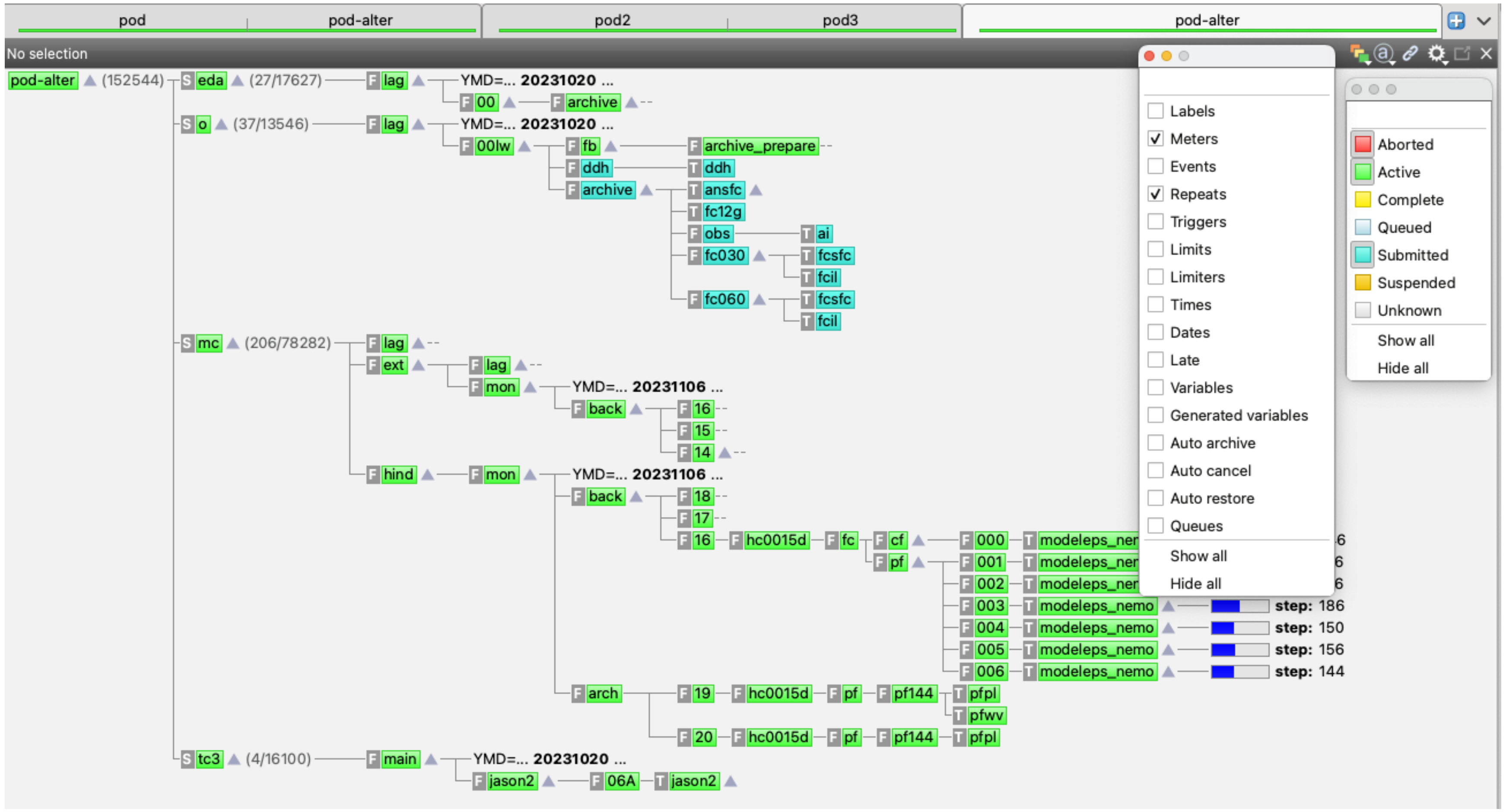








ecFlow in Operations: operators' view



- [ ] Labels
- [✓] Meters
- [ ] Events
- [✓] Repeats
- [ ] Triggers
- [ ] Limits
- [ ] Limiters
- [ ] Times
- [ ] Dates
- [ ] Late
- [ ] Variables
- [ ] Generated variables
- [ ] Auto archive
- [ ] Auto cancel
- [ ] Auto restore
- [ ] Queues

Show all
Hide all

- 🟥 Aborted
- 🟩 Active
- 🟨 Complete
- ⬜ Queued
- 🟦 Submitted
- 🟧 Suspended
- ⬜ Unknown

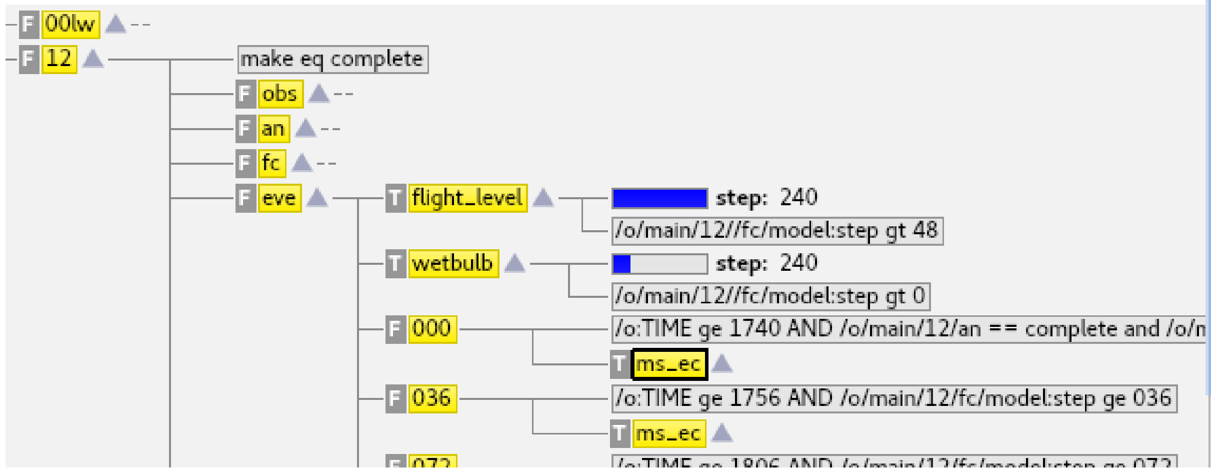
Show all
Hide all

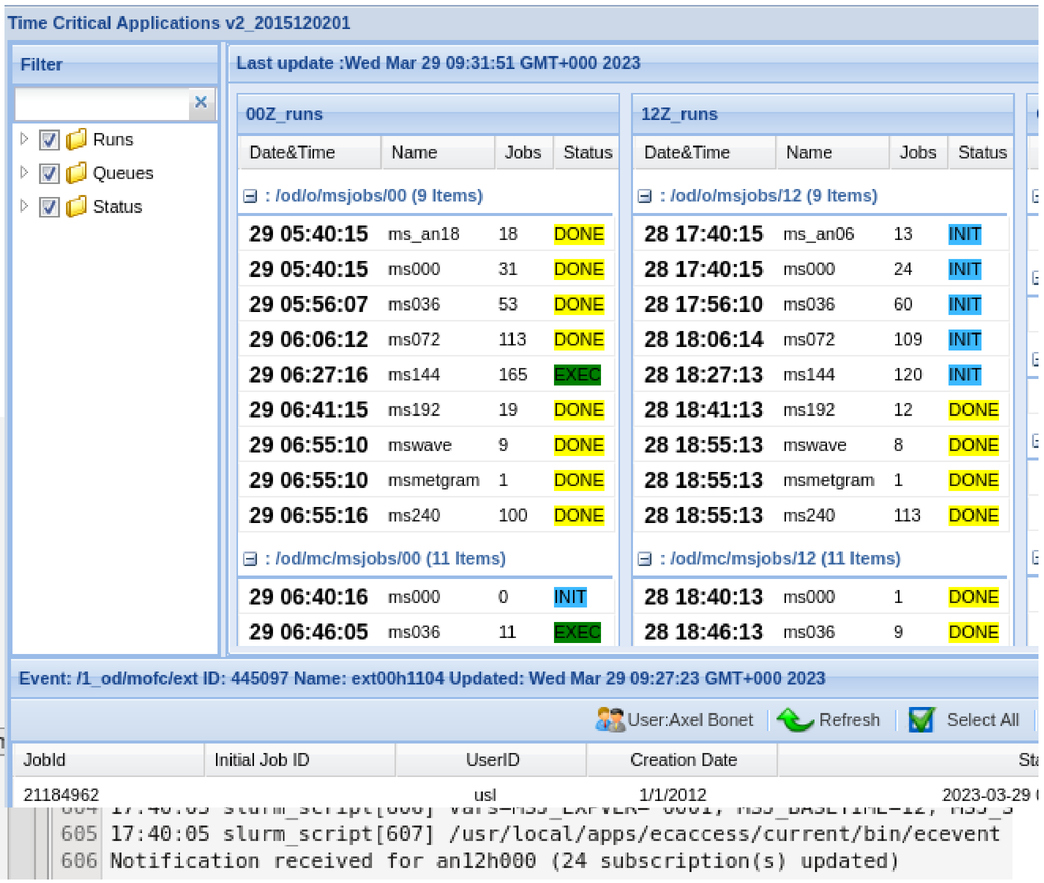
ecFlow: Time critical TC1

- Operational suites send events to ecaccess
- subscribe to events for simple jobs to run

```
module load ecaccess
ecaccess-event-list
ecaccess-job-submit --help
```

Inherited variables	
MSJ_EVENT	MSJ_STEP
MSJ_EXPVER	MSJ_MEMBERS
MSJ_BASETIME	…


F 00lw
F 12
make eq complete
F obs
F an
F fc
F eve
T flight_level
step: 240
/o/main/12//fc/model:step gt 48
T wetbulb
step: 240
/o/main/12//fc/model:step gt 0
F 000
/o:TIME ge 1740 AND /o/main/12/an == complete and /o/n
T ms_ec
F 036
/o:TIME ge 1756 AND /o/main/12/fc/model:step ge 036
T ms_ec


Time Critical Applications v2_2015120201

Filter

Runs
Queues
Status

Last update :Wed Mar 29 09:31:51 GMT+000 2023

00Z_runs			
Date&Time	Name	Jobs	Status
: /od/o/msjobs/00 (9 Items)			
29 05:40:15	ms_an18	18	DONE
29 05:40:15	ms000	31	DONE
29 05:56:07	ms036	53	DONE
29 06:06:12	ms072	113	DONE
29 06:27:16	ms144	165	EXEC
29 06:41:15	ms192	19	DONE
29 06:55:10	mswave	9	DONE
29 06:55:10	msmetgram	1	DONE
29 06:55:16	ms240	100	DONE
: /od/mc/msjobs/00 (11 Items)			
29 06:40:16	ms000	0	INIT
29 06:46:05	ms036	11	EXEC

12Z_runs			
Date&Time	Name	Jobs	Status
: /od/o/msjobs/12 (9 Items)			
28 17:40:15	ms_an06	13	INIT
28 17:40:15	ms000	24	INIT
28 17:56:10	ms036	60	INIT
28 18:06:14	ms072	109	INIT
28 18:27:13	ms144	120	INIT
28 18:41:13	ms192	12	DONE
28 18:55:13	mswave	8	DONE
28 18:55:13	msmetgram	1	DONE
28 18:55:13	ms240	113	DONE
: /od/mc/msjobs/12 (11 Items)			
28 18:40:13	ms000	1	DONE
28 18:46:13	ms036	9	DONE

Event: /1_od/mofc/ext ID: 445097 Name: ext00h1104 Updated: Wed Mar 29 09:27:23 GMT+000 2023

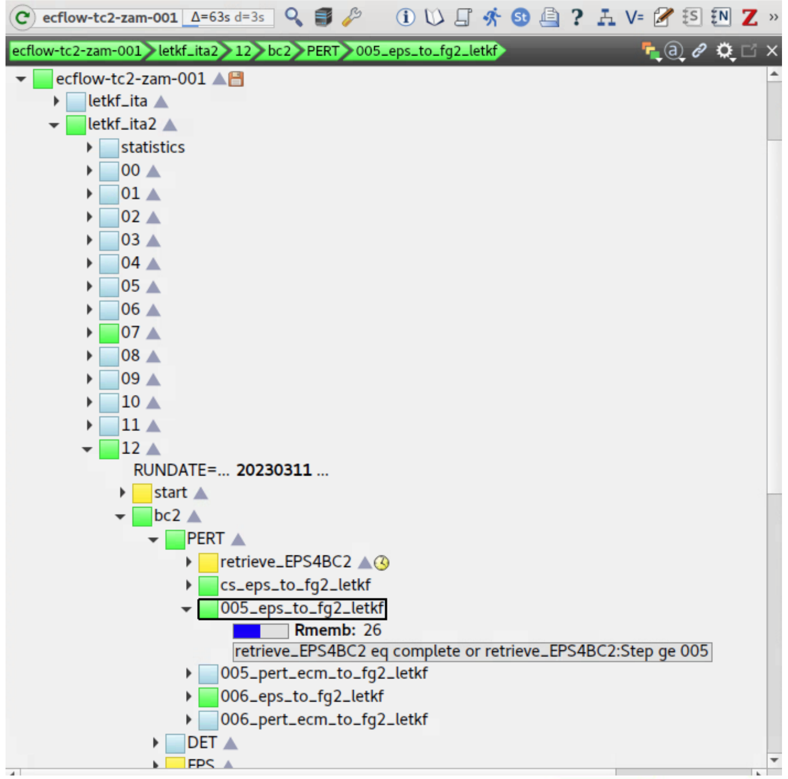
User:Axel Bonet
Refresh
Select All

JobId	Initial Job ID	UserID	Creation Date	Sta
21184962		usl	1/1/2012	2023-03-29

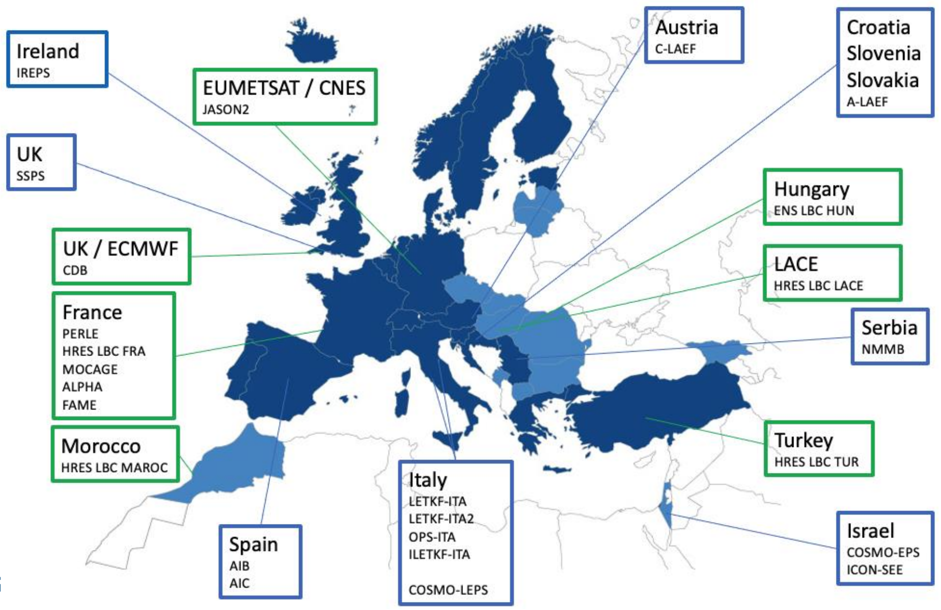
```
17:40:05 slurm_script[607] /usr/local/apps/ecaccess/current/bin/ecevent
Notification received for an12h000 (24 subscription(s) updated)
```

ecFlow: Time critical TC2

- Member State ecFlow suites monitored by ECMWF
- Run with special user accounts on HPCF
  - Enhanced priority
  - Access to redundant computing and storage backends
- Use of ECMWF Dissemination system for data transfers
- Shift staff monitor jobs 24/7
  - Rerun tasks if failed. follow manual page if present
  - Notification to MS responsible team if problem persists

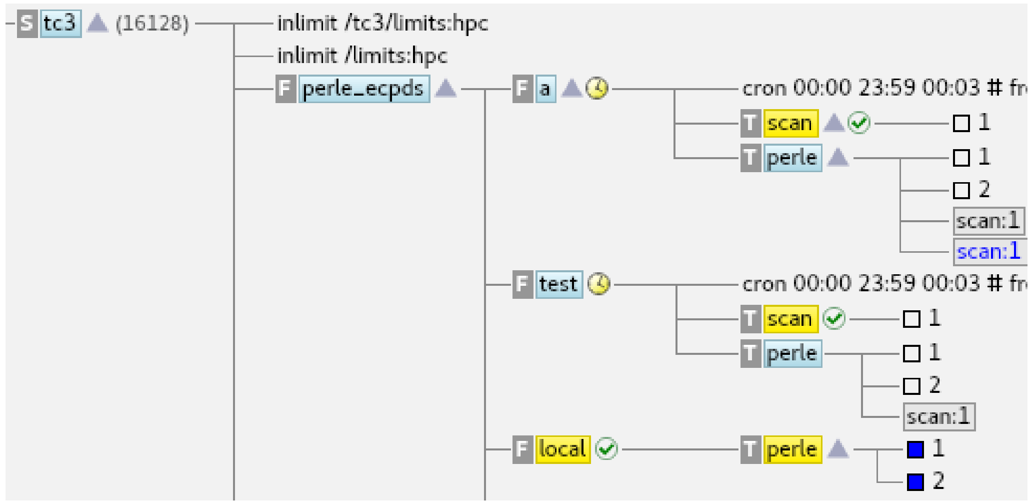


```
ecflow-tc2-zam-001  Δ=63s d=3s
ecflow-tc2-zam-001 > letkf_ita2 > 12 > bc2 > PERT > 005_eps_to_fg2_letkf
ecflow-tc2-zam-001
  letkf_ita
  letkf_ita2
    statistics
    00
    01
    02
    03
    04
    05
    06
    07
    08
    09
    10
    11
    12
      RUNDATE=... 20230311 ...
      start
      bc2
        PERT
          retrieve_EPS4BC2
          cs_eps_to_fg2_letkf
          005_eps_to_fg2_letkf
            Rmemb: 26
            retrieve_EPS4BC2 eq complete or retrieve_EPS4BC2:Step ge 005
          005_pert_ecm_to_fg2_letkf
          006_eps_to_fg2_letkf
          006_pert_ecm_to_fg2_letkf
        DET
        EPS
```

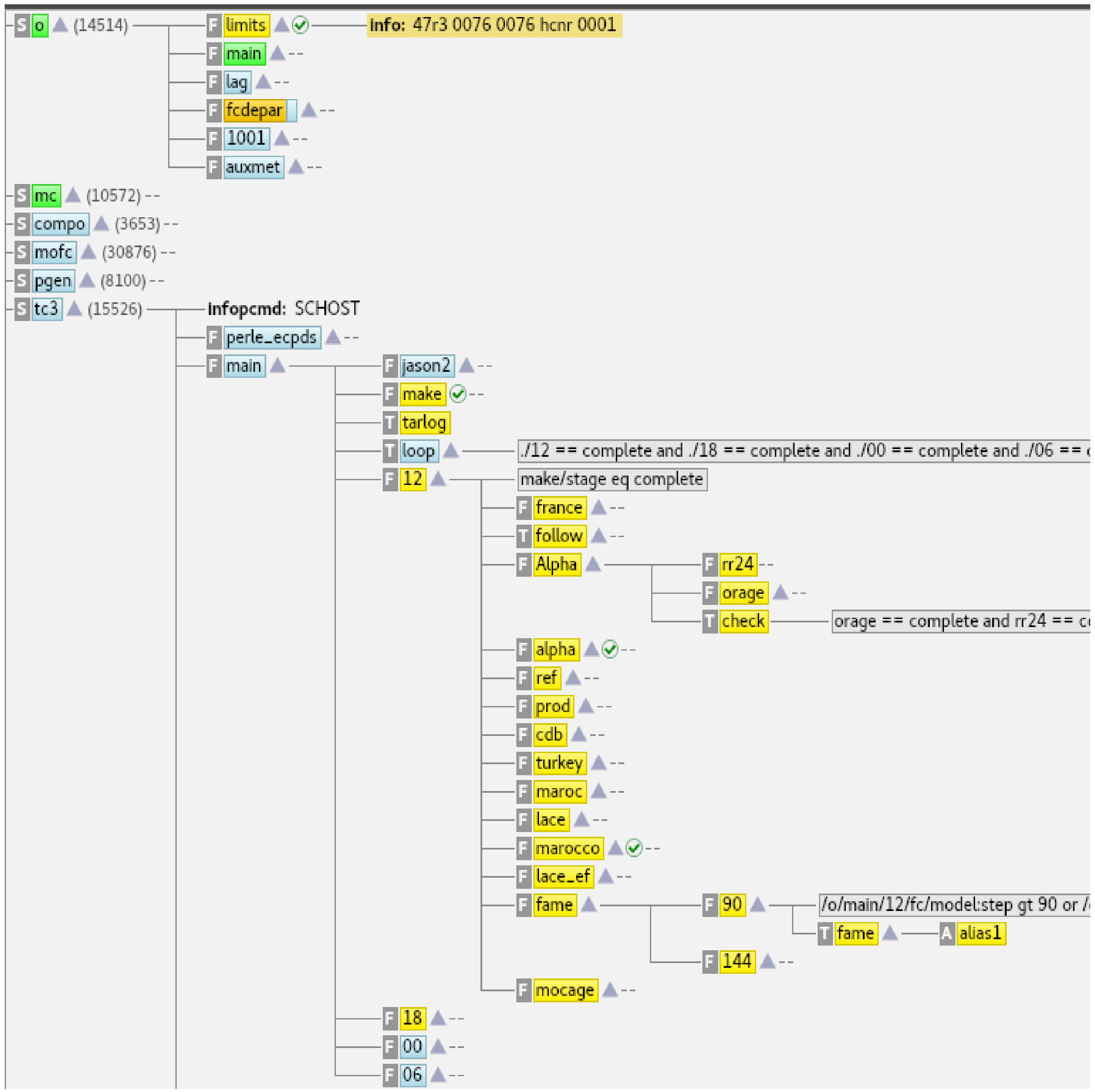


ecFlow: Time critical TC3

- A dedicated suite run as EMOS
- Hosted on EMOS operational server 1_od
- "extern" triggers to o and mc suite
- Ecpds is used to disseminate products
- Tested with new cycles as an esuite
- Ecpds acq can set event to start tasks



```
S tc3 (16128) ── inlimit /tc3/limits:hpc
               ├ inlimit /limits:hpc
               └ F perle_ecpds ─┬ F a ── cron 00:00 23:59 00:03 # fr
                                │   ├ T scan ── □ 1
                                │   └ T perle ─┬ □ 1
                                │              ├ □ 2
                                │              ├ scan:1
                                │              └ scan:1
                                ├ F test ── cron 00:00 23:59 00:03 # fr
                                │   ├ T scan ── □ 1
                                │   └ T perle ─┬ □ 1
                                │              ├ □ 2
                                │              └ scan:1
                                └ F local ── T perle ─┬ ■ 1
                                                      └ ■ 2
```



```
S o (14514) ──┬ F limits ── Info: 47r3 0076 0076 hcnr 0001
              ├ F main --
              ├ F lag --
              ├ F fcdepar --
              ├ F 1001 --
              └ F auxmet --
S mc (10572) --
S compo (3653) --
S mofc (30876) --
S pgen (8100) --
S tc3 (15526) ── Infopcmd: SCHOST
              ├ F perle_ecpds --
              └ F main ─┬ F jason2 --
                        ├ F make --
                        ├ T tarlog
                        ├ T loop ── ./12 == complete and ./18 == complete and ./00 == complete and ./06 == c
                        ├ F 12 ── make/stage eq complete
                        │   ├ F france --
                        │   ├ T follow --
                        │   ├ F Alpha ─┬ F rr24 --
                        │   │          ├ F orage --
                        │   │          └ T check ── orage == complete and rr24 == c
                        │   ├ F alpha --
                        │   ├ F ref --
                        │   ├ F prod --
                        │   ├ F cdb --
                        │   ├ F turkey --
                        │   ├ F maroc --
                        │   ├ F lace --
                        │   ├ F marocco --
                        │   ├ F lace_ef --
                        │   ├ F fame ─┬ F 90 ── /o/main/12/fc/model:step gt 90 or /
                        │   │         │   └ T fame ── A alias1
                        │   │         └ F 144 --
                        │   └ F mocage --
                        ├ F 18 --
                        ├ F 00 --
                        └ F 06 --
```

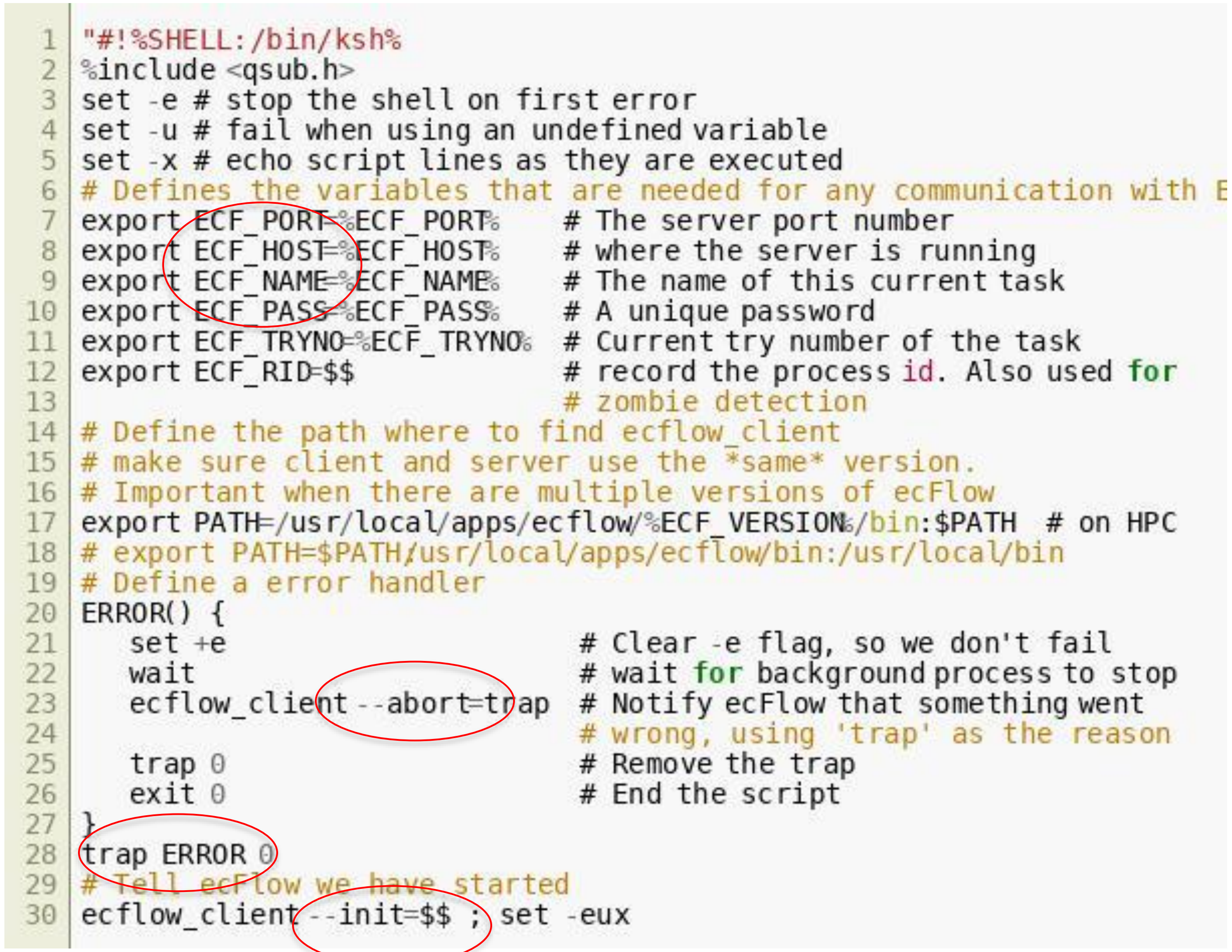
Questions + Practical

ecFlow: wrap up

- You learnt:
  - Using few ecflow components, server, client, ecflow_ui
  - How to start with a suite definition
  - How to run one or few tasks with ecflow on HPC

- ecFlow is fun: enjoy ☺

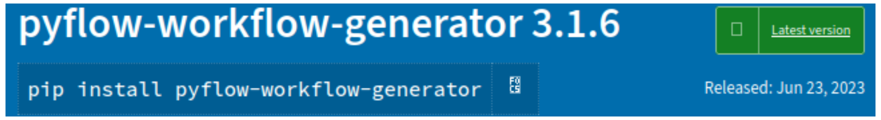
Head.h



```
"#!%SHELL:/bin/ksh%
%include <qsub.h>
set -e # stop the shell on first error
set -u # fail when using an undefined variable
set -x # echo script lines as they are executed
# Defines the variables that are needed for any communication with E
export ECF_PORT=%ECF_PORT%    # The server port number
export ECF_HOST=%ECF_HOST%    # where the server is running
export ECF_NAME=%ECF_NAME%    # The name of this current task
export ECF_PASS=%ECF_PASS%    # A unique password
export ECF_TRYNO=%ECF_TRYNO%  # Current try number of the task
export ECF_RID=$$             # record the process id. Also used for
                              # zombie detection
# Define the path where to find ecflow_client
# make sure client and server use the *same* version.
# Important when there are multiple versions of ecFlow
export PATH=/usr/local/apps/ecflow/%ECF_VERSION%/bin:$PATH  # on HPC
# export PATH=$PATH/usr/local/apps/ecflow/bin:/usr/local/bin
# Define a error handler
ERROR() {
   set +e                      # Clear -e flag, so we don't fail
   wait                        # wait for background process to stop
   ecflow_client --abort=trap  # Notify ecFlow that something went
                               # wrong, using 'trap' as the reason
   trap 0                      # Remove the trap
   exit 0                      # End the script
}
trap ERROR 0
# Tell ecFlow we have started
ecflow_client --init=$$ ; set -eux
```

ecFlow: pyflow

- Closer integration of the suite definition and tasks wrappers creation with python language:
  - Families and Tasks can be defined through derivation / composition
  - Possibility to define task wrappers in the definition (Script attribute)
  - Meta definition: the tasks wrappers are created dynamically in the "natural" file tree structure
  - Optimisation like expressions template with C++, compute in advance
  - Reduce the need for %include header
  - Trigger/complete expression naturally expressed with python language and objects


pyflow-workflow-generator 3.1.6

```
pip install pyflow-workflow-generator
```

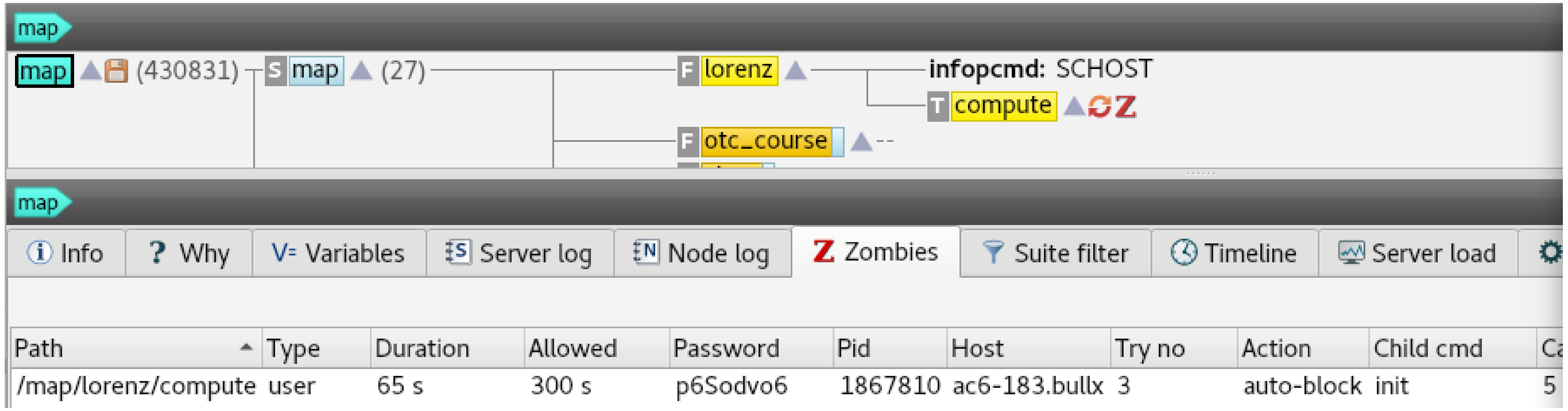
Released: Jun 23, 2023

Glossary

- API: Application Programming Interface
- CLI: Command line interface
- GUI: Graphical User Interface
- LLM: Large Language Model
- Proxy chain: to run an application through a proxy server
- REST: Representational State Transfer
- SCM: Source Control manager
- UDP: User Datagram protocol
- VM: Virtual Machine
- Workflow: set of tasks and their dependencies

ecFlow: important concepts, Zombies

- Jobs are submitted with variable **ECF_PASS** set to pseudo-random value by ecflow server
- Jobs are defined with unique identifiers **ECF_HOST-ECF_PORT-ECF_NAME-ECF_PASS**
  - A zombie arises when a **child** command is received and ECF_PASS does not match
  - set ECF_PASS FREE # allow communication with zombie
  - Clear Flag
  - Rescue?
  - Fob off?
  - Kill?
  - Terminate?
  - Delete?



Path	Type	Duration	Allowed	Password	Pid	Host	Try no	Action	Child cmd	Ca
/map/lorenz/compute	user	65 s	300 s	p6Sodvo6	1867810	ac6-183.bullx	3	auto-block	init	5

ecFlow: security

- Designed for collaborative working, in absence of ecf.lists file, access is **open**
- ecflow server is protected with **white list** file: ecf.lists
  - restricted set of users with **read** (Script, Output) or **read-write access** (Edit, Submit)
- We use specific accounts for operations and research
- Communication on fixed port: **ECF_PORT**
- **black** list file for user **authentification** to access server, suite, node
- Communication may be **encrypted**: compile with option ENABLE_SSL
- Some jobs are submitted for another user: careful with
  - job-file owner, output file owner, ssh settings, queueing system permissions

Components

- **Server** on port 3141 (default at ECMWF)
- Status inheritance is bubbling up
  - User VM ecflow-gen-${USER}-001
  - multiple servers can run for one and multiple users on same CPU in general
  - Log file: ECF_LOG
  - Checkpoint file: ECF_CHECK
  - White-list: ECF_LIST
- **Client** / shell CLI + Python API
  - Can be used by users and jobs (child commands)
  - Use class for multiple connections at once
- **GUI:** ecflow_ui
- **UDP** (light) client
- REST-API

- Definition file: a simple DSL
- Tasks wrappers: a simple template language
- Tasks headers pure or template language
- Supervision
  - ECF_JOB_CMD
  - ECF_KILL_CMD
  - ECF_STATUS_CMD
  - ECF_CHECK_CMD
  - ECF_URL_CMD