ecflow - Python

Axel Bonet

Production Section – Integration Team

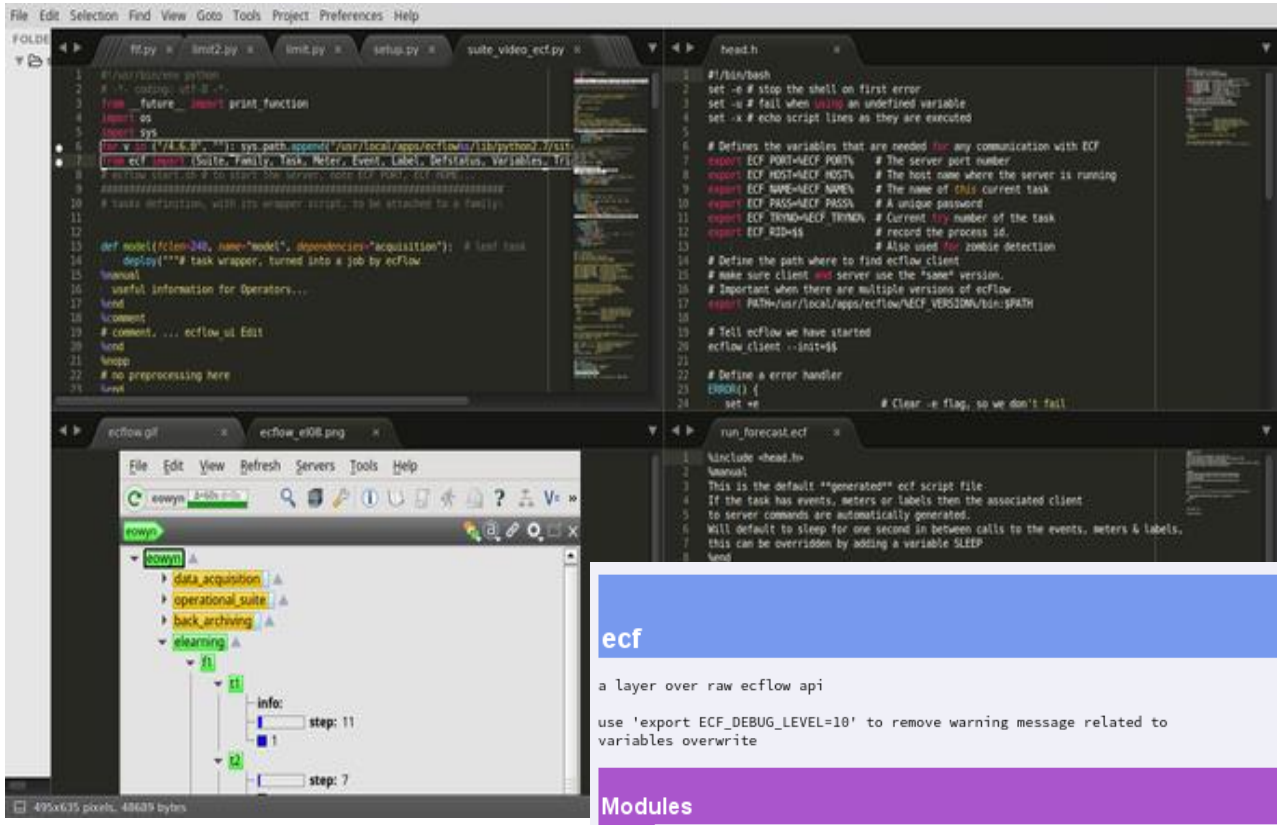
ECMWF

Python suites

- Python for the **definition file**
- Python for a passive **client**: monitor, download server content
- Python for an **active** client: monitor, upload, alter
- Python for a **Job**
  - 'pure python job' v. 'calling python from a job', KISS, robust, reproducible!
  - Job environment: exceptions, classes, scope by blocks
  - Python for **child commands:** init, complete, abort, event, meter
  - `Edit({"ECF_EXTN":".py", "ECF_MICRO":'$'})`
  - `edit micro ^`

Python suites

- Definition file
  - Pep8 style guide (here), python docs (here), ipython (here)
  - **Pylint**, PyCheck, PyFlake, coverage, **pydoc**
  - No need to print expanded suite definition
  - Handle/raise **exceptions**: detect issues earlier, ex. missing key in a dictionary
  - **Typed** variables, Local/Block scope for variables (module, def, class) unless global is specified
  - **Functional Programming**: reduce temporary variables
  - **List comprehension** for loops, Lambda expression for anonymous functions



ecf

a layer over raw ecflow api

use 'export ECF_DEBUG_LEVEL=10' to remove warning message related to variables overwrite

Modules

ecflow pwd
os sys

Classes

__builtin__.object

Attribute

Autocancel
Clock
Defcomplete

DefcompleteIf
Defstatus

Event(Attribute, ecflow.ecflow.Event)
InLimit

Inlimit

Attributes

Attributes	Behaviour	Context	script
Repeat	Requeue on complete, one per node	Problem with Cron! Used in trigger	%YMD%
Edit	hide Repeat	Used in trigger	%VAR:default%
Event	Set by Child, user, batch	Used in trigger Used to monitor	ecflow_client --event
Meter	Set by Child, user, batch	Used in trigger Used to monitor	--meter
Label		Monitor only	--label
Trigger	One per node		--wait
Limit		Used in trigger	
Late	Flag, pop	Used in trigger	

Attributes	Behaviour
Cron	Requeue immediately at complete
Complete	One per node, set complete asap
Time	00:00 23:59 01:00
Date	1.*.* good to associate with Time
Today	The time of node replace
Day	good to associate with Time
Limit	Global, local, inherited, “hidden” Like suspend when set to 0
Inlimit	One token for each task, or One token on active (family inlimit)
Defstatus	After begin, requeue, (repeat)
Clock	Real, hybrid

Python suites

- One script “do-it-all”? **mirror.py** is an example
  - Suite definition, load, replace node
  - Create task templates and headers
  - It can be a wrapper itself for job definition: pre-process, submit
  - Still, it can be called from command line: test, standalone
  - Child communication by a dedicated class
  - Something missing? YES! ^include <header.py>
  - Doc-string for blocs ^manual … ^end ^comment … ^end
- Native API: `import ecflow`
- Functional Programming API, polymorphism: `import ecf`
  - Trigger/Complete/Late/Inlimit: activate, inhibit with one variable change
  - A chance for “No Hardcoded Trigger expressions”: expression generated from python objects
  - When playing a suite on a SMS server, variables are translated “on the fly”

Python suites - classes – inheritance

```
#!/usr/bin/env python
# inherit
class Parent(object):
    def sing(self):
        print("singing")

class Kid(Parent):
    pass

mother = Parent()
kid = Kid()

mother.sing()
kid.sing()

"""
singing
singing
"""
```

```
#!/usr/bin/env python
# overwrite
class Parent(object):
    def sing(self):
        print("singing")

class Kid(Parent):
    def sing(self):
        print("stammering")

mother = Parent()
kid = Kid()

mother.sing()
kid.sing()
"""
singing
stammering
"""
```

Python suites - classes – inheritance (inhibit, enhance)

```
#!/usr/bin/env python
# inhibit
class Parent(object):
    def sing(self): print("singing")

class Kid(object):
    def sing(self): pass

mother = Parent()
kid = Kid()
mother.sing()
kid.sing()
"""
singing
"""
```

```
#!/usr/bin/env python
# extend
class Parent(object):
    def sing(self): print("singing")

class Kid(Parent):
    def sing(self):
        print("listen")
        super(Kid, self).sing()
        print("stammering")

mother = Parent()
kid = Kid()
mother.sing()
kid.sing()
"""singing
listen
singing
stammering"""
```

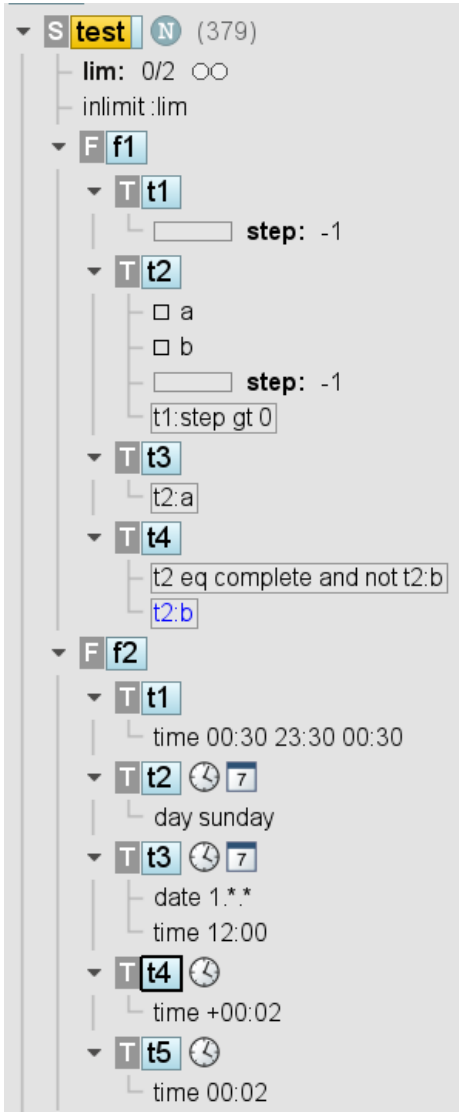
Python suites - classes – Composition

```
#!/usr/bin/env python
class OPER(object):
  def implicit(self): print "implicit"
  def overwrite(self): print "overwrite"
  def inhibit(self): print "inhibit"
  def extend(self): print "extend"

class Suite(object):
    def __init__(self): self.stream = OPER()
    def implicit(self): self.stream.implicit()
    def overwrite(self): print "Suite overwrite"
    def inhibit(self): pass
    def extend(self):
        print "Suite before",
        self.stream.extend()
        print "Suite after"

if __name__ == '__main__':
  item = Suite(); item.implicit(); item.overwrite(); item.inhibit()
  item.extend()
"""implicit
Suite overwrite
Suite before extend
Suite after"""
```

```
#!/usr/bin/env python
from compose import OPER, Suite
class ENFO(OPER):
  def __init__(self): self.id = "ENFO"
  def ident(): print self.id
  def implicit(self): print "implicit " + self.id
  def overwrite(self): print "overwrite " + self.id
  def inhibit(self): print "inhibit " + self.id
  def extend(self): print "extend " + self.id

class Test(Suite):
    def __init__(self):
        super(Test, self).__init__()
        self.stream = ENFO()

if __name__ == '__main__':
  item = Test(); item.implicit();  item.overwrite()
  item.inhibit(); item.extend()
"""implicit ENFO
Suite overwrite
Suite before extend ENFO
Suite after
"""
```

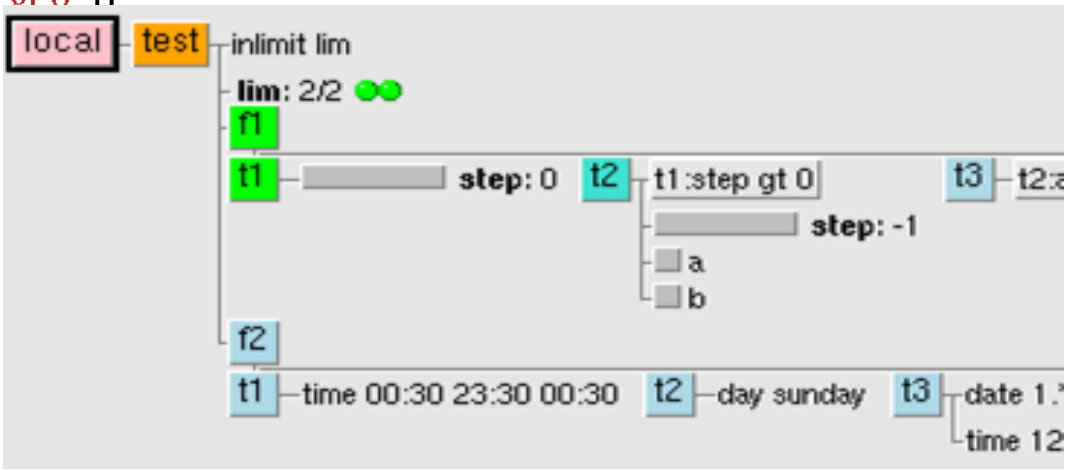
Python suites – Suite, Family definition

- With a shell suite:
  - attribute are added in place
  - sequential suite definition!
- With a Python suite: navigate anytime
  - Verify/fix a suite before loading a node
  - Check job creation, Simulate the suite, Add/Replace/Delete Attributes
- **Module** as provider/decorator
  - **OOP**: no “if” anymore, use Class?
  - **FP**: no “temporary objects”?
- iterative family addition with replace
- What’ s wrong with this suite?



Python suites – example – suite.py

```
#!/usr/bin/env python2.7
import sys, os; path = "/home/ma/emos/def/o/def"; sys.path.append(path)
import ecf; from ecf import (Autocancel, Client, Clock, Complete, Cron, Date, Day, Defs,
Defstatus, Edit, Event, Extern, Family, Inlimit, Label, Late, Limit, Meter, Node, Repeat, Suite,
Task, Time, Today, Trigger)
home = os.environ['HOME'] + '/course'
def create(): return Suite("test").add(
   Edit(ECF_INCLUDE=home, ECF_FILES=home, ECF_HOME=home),
   Defstatus("suspended"), Limit("lim", 2), Inlimit("lim"),
   Family("f1").add(
      Task("t1").add(Meter("step", -1, 100)),
      Task("t2").add(Meter("step", -1, 100), Event("a"), Event("b"), Trigger("t1:step gt 0")),
      Task("t3").add(Trigger("t2:a")),
      Task("t4").add(Complete("t2:b"), Trigger("t2 eq complete and not t2:b"))),
   Family("f2").add(
      Task("t1").add(Time("00:30 23:30 00:30")),
      Task("t2").add(Day("sunday")),
      Task("t3").add(Time("12:00"), Date("1.*.*")),
      Task("t4").add(Time("+00:02")),
      Task("t5").add(Time("00:02"))))
if __name__ == "__main__":
      client = Client("localhost@%s" % os.getenv('ECF_PORT', '31415'));
      defs = Defs(); suite = create(); defs.add_suite(suite); client.replace("/test", defs)
```

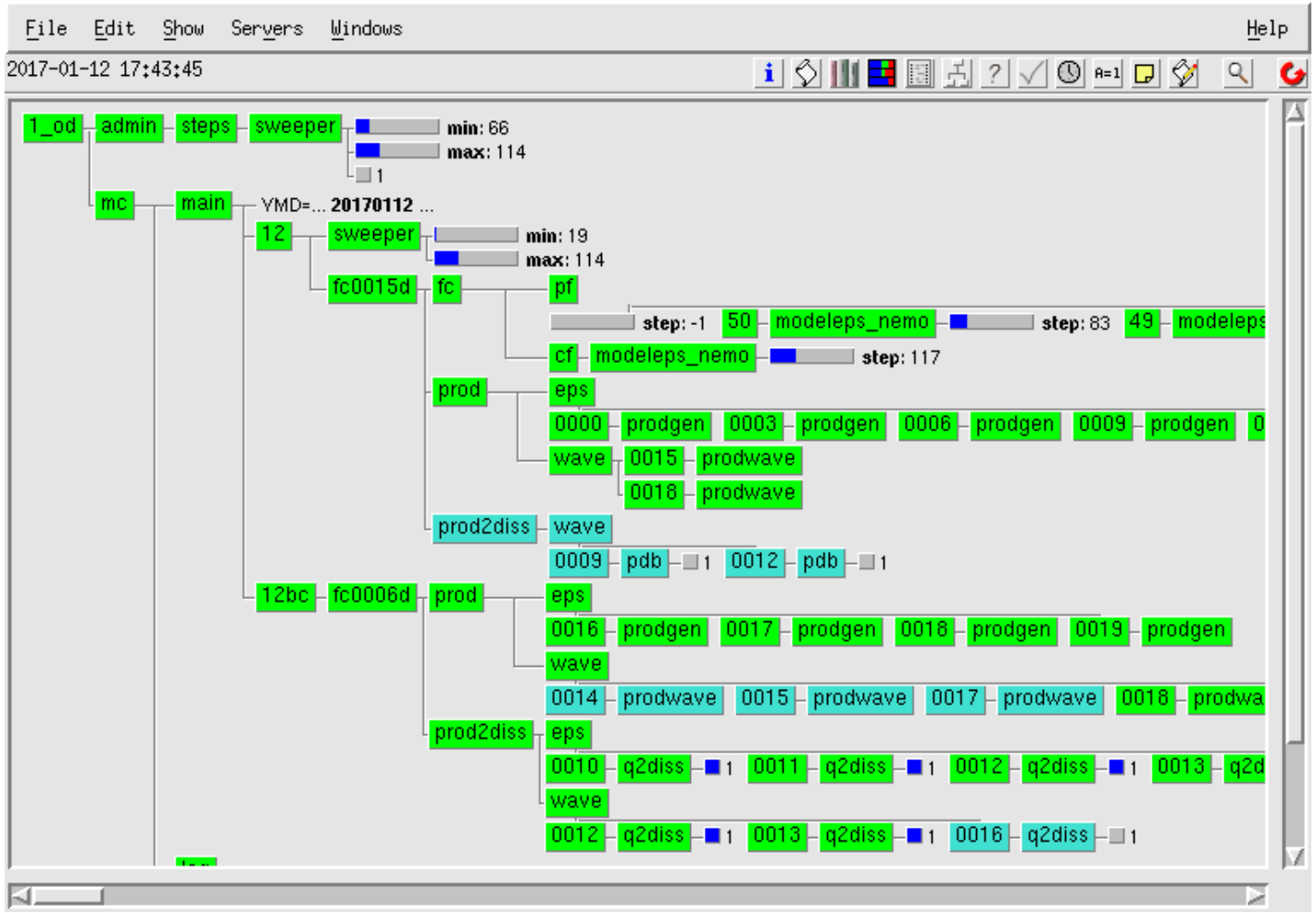


Python script – task template may be executable

- Operational example: **sweeper.py**

```
#!/usr/bin/env python
"""
^manual
  a python script can be used as task wrapper...
^end
^comment
  # add into def-file
  edit ECF_MICRO '^'  # '^' ... balance
  edit ECF_EXTN '.py'
^end
"""
MICRO = "^^" # double for micro character balance with preprocess
host = os.getenv("ECF_HOST", "^ECF_HOST^")
if MICRO[0] in host: print("command line call")
else: print("processed into a job by ecFlow")
```

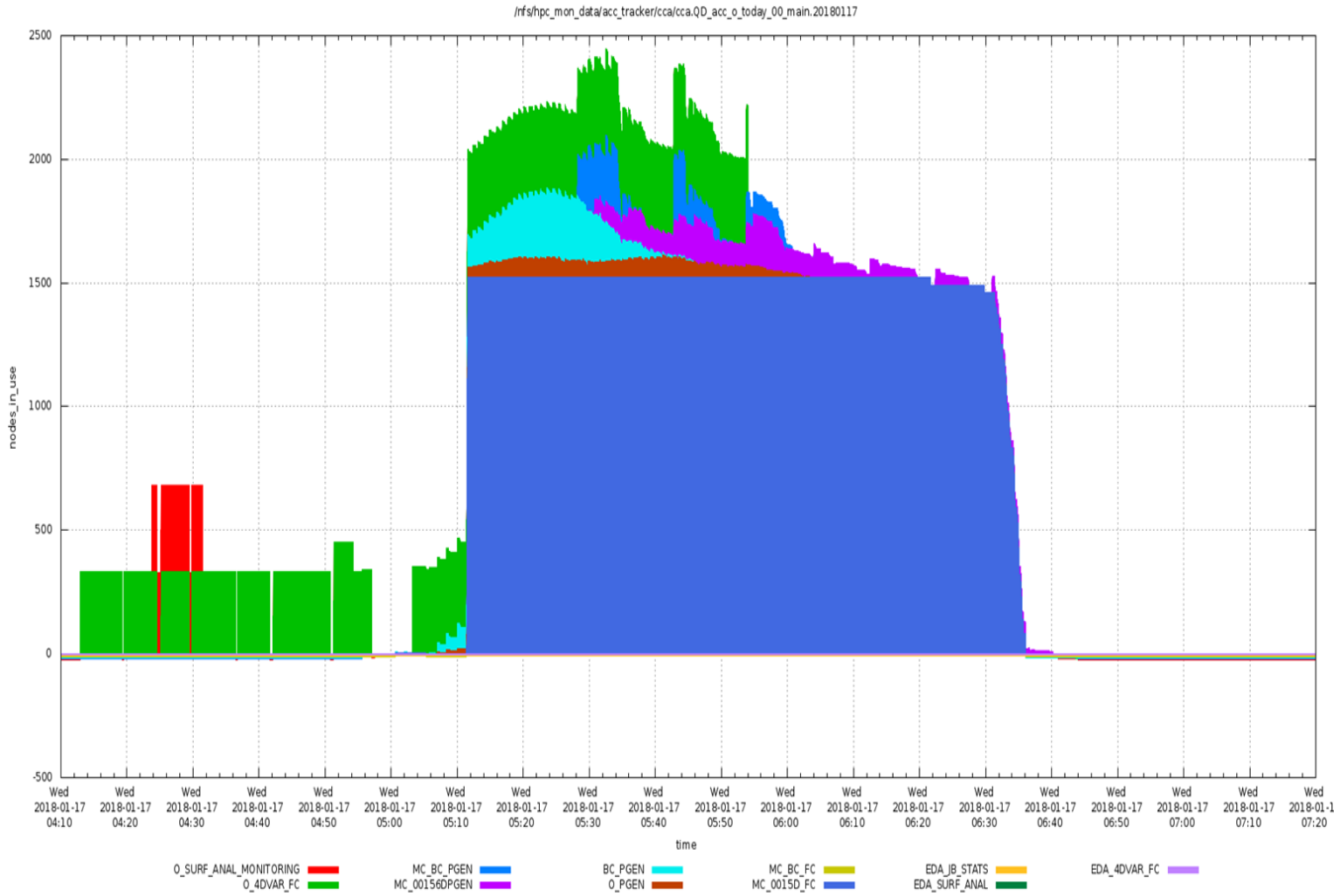
Python script – simple client v. complex triggers



Python script - sweeper - the elephant chart

ENS models checkpoint

by step 48, 96, 144, 192, 236



Python script – task template – Child class

- mirror.py: communication with server handled by one class, Child

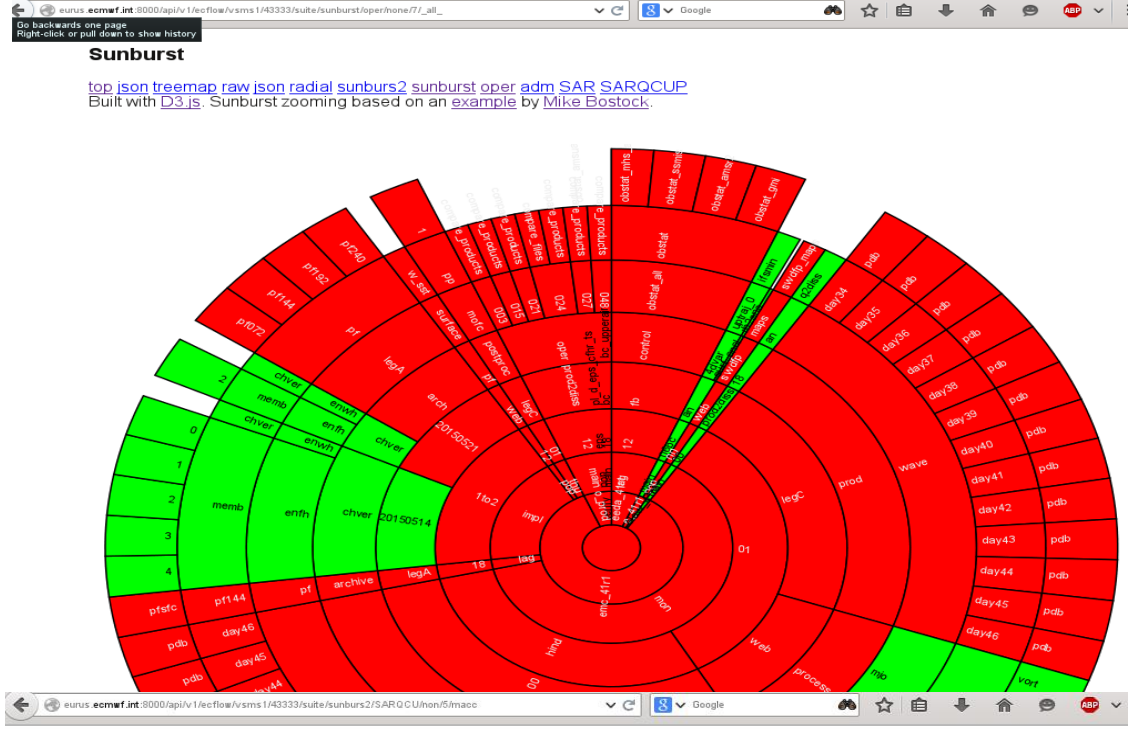
```
#!/usr/bin/python
class Child(object):
   """ kid = Child(); kid.report("complete")
      this does nothing when script is called from command line    """
   def __init__(self):
      env = { "ECF_NODE": "$ECF_NODE$", "ECF_PASS": "$ECF_PASS$",
            "ECF_NAME": "$ECF_NAME$", "ECF_PORT": "$ECF_PORT$", "ECF_TRYNO":
"$ECF_TRYNO$", }
      if MICRO[0] in env["ECF_PORT"]: self.client = None; return
      self.client = ec.Client();        self.client.set_child_timeout(20)
      self.client.set_host_port(env["ECF_NODE"], int(env["ECF_PORT"]))
      self.client.set_child_pid(os.getpid())
      self.client.set_child_path(env["ECF_NAME"])
      self.client.set_child_password(env["ECF_PASS"])
      self.client.set_child_try_no(int(env["ECF_TRYNO"]))
      self.report("init"); # ...
kid = Child()
```

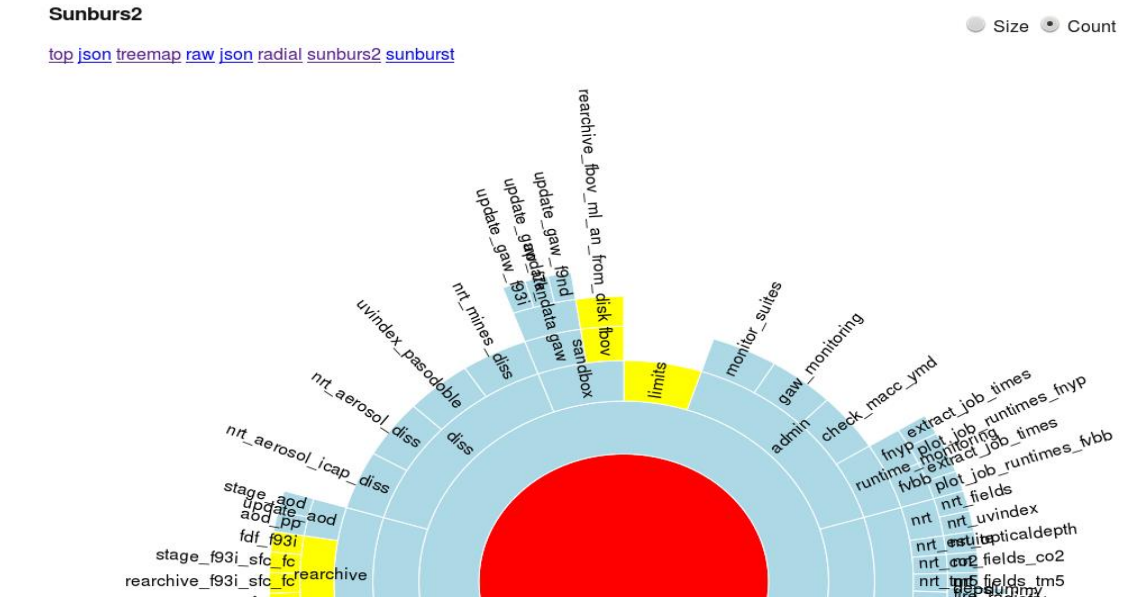
Python client – example - suite navigator

```
#!/usr/bin/env python
import sys, os
from ecflow import *
#path = "/home/ma/emos/def/o/def"; sys.path.append(path); from ecf import *
client = Client(os.getenv('ECF_HOST', "localhost"), os.getenv('ECF_PORT',
"1630"))
client.ch_register(False, [ "test", "suite"])
client.sync_local()
defs = client.get_defs()

def process(node):
    if isinstance(node, ecflow.Task): print "task",
    elif isinstance(node, ecflow.Family): print "family",
    elif isinstance(node, ecflow.Suite): print "suite",
    print node.get_abs_node_path(), node.get_state(), "T:", node.get_trigger(), "C:",
node.get_complete()
    for item in node.nodes: process(item)

for item in defs.suites: process(item)
```

Python for visualisation: Django

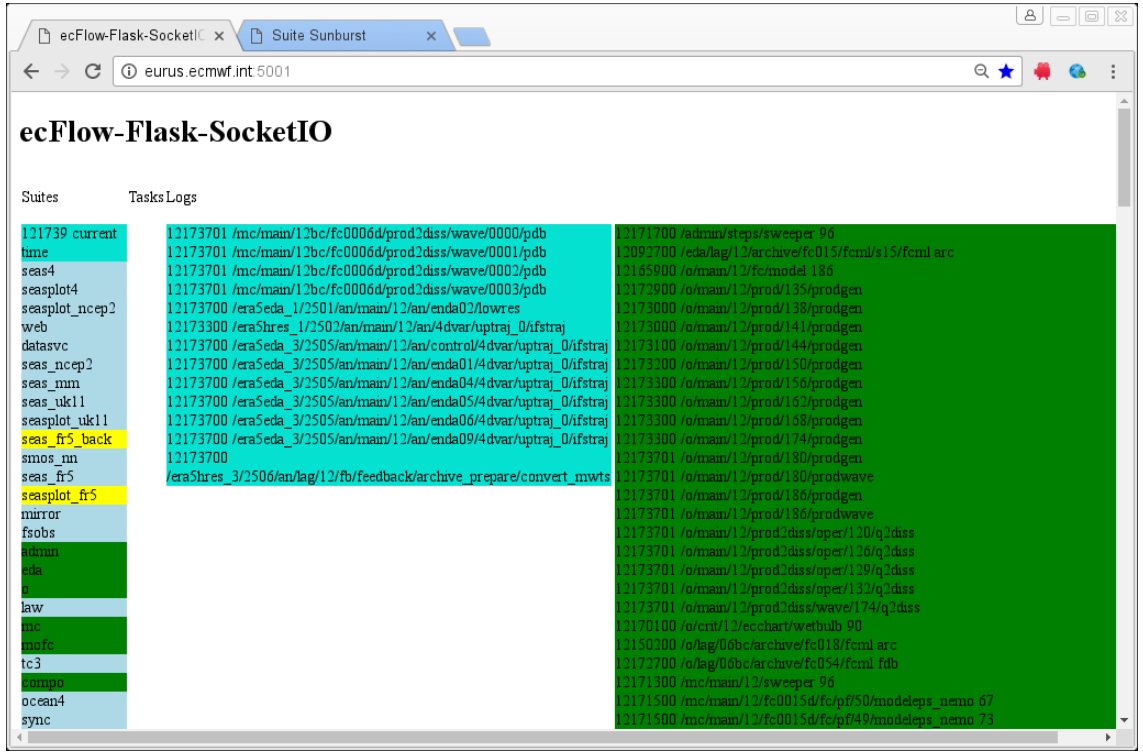
- ecflow python client together with
- https://d3js.org/Data-Driven Document
- Django https://www.djangoproject.com

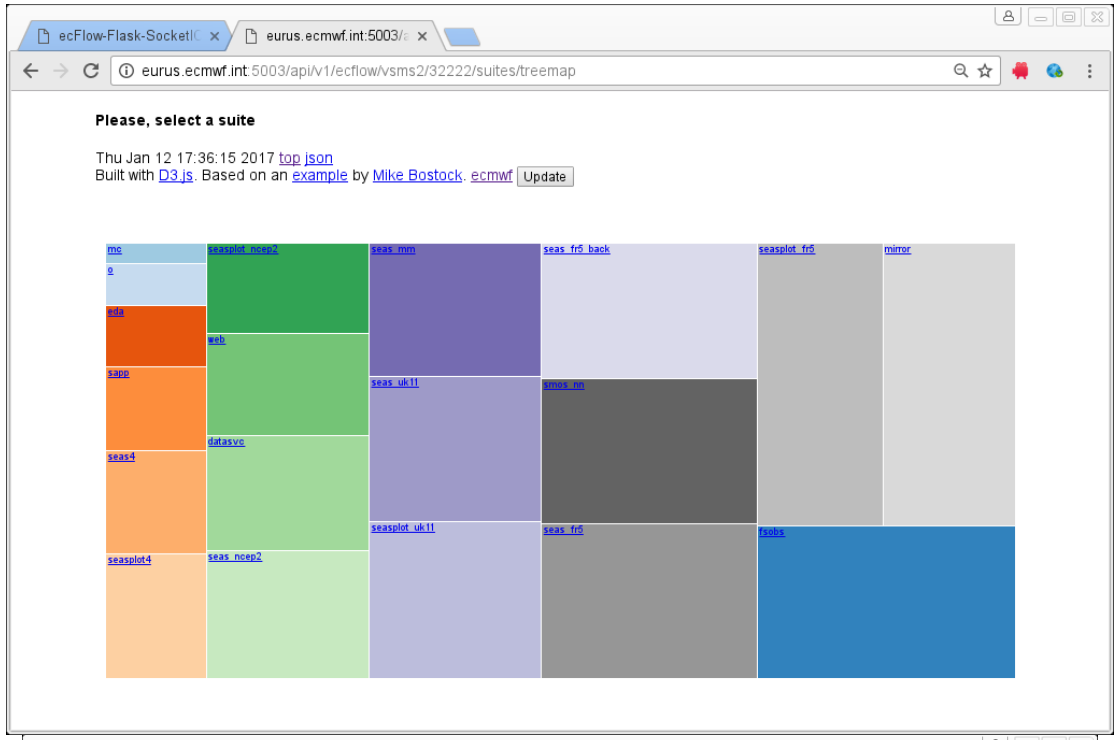


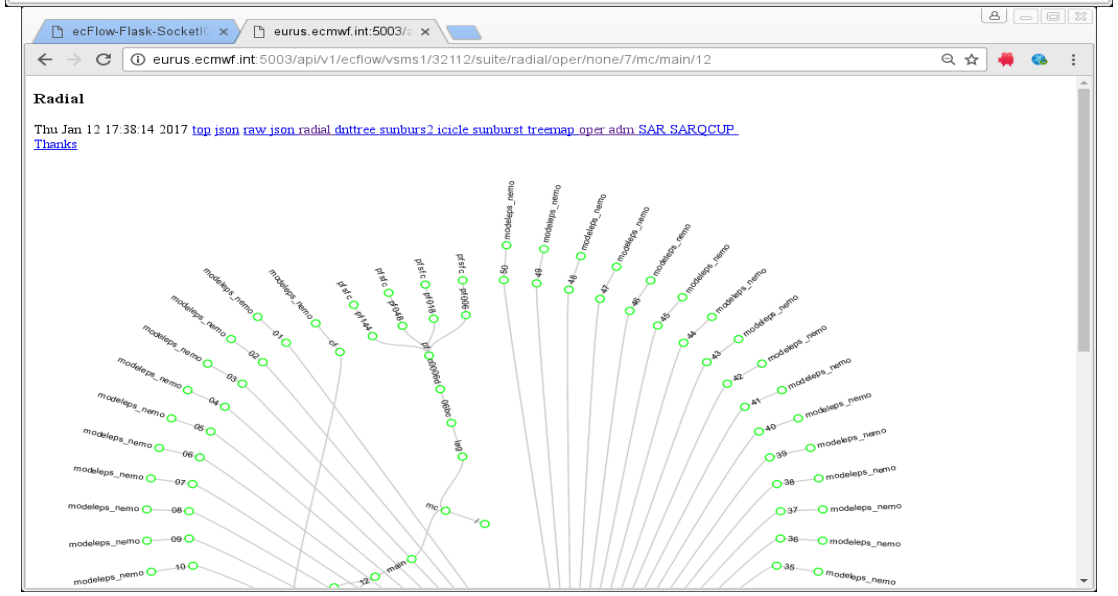


Python for visualisation: Flask

- Flask web server is lightweight (here)

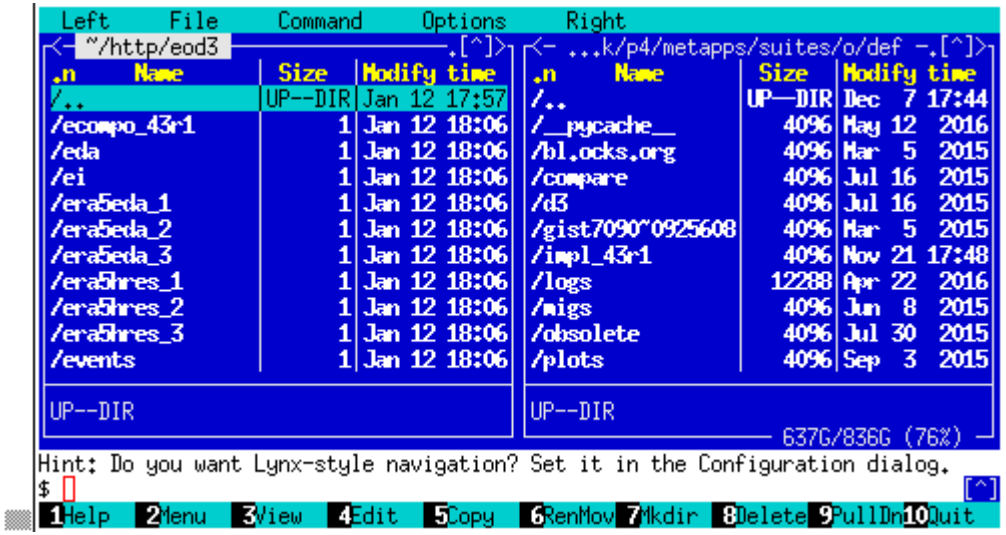


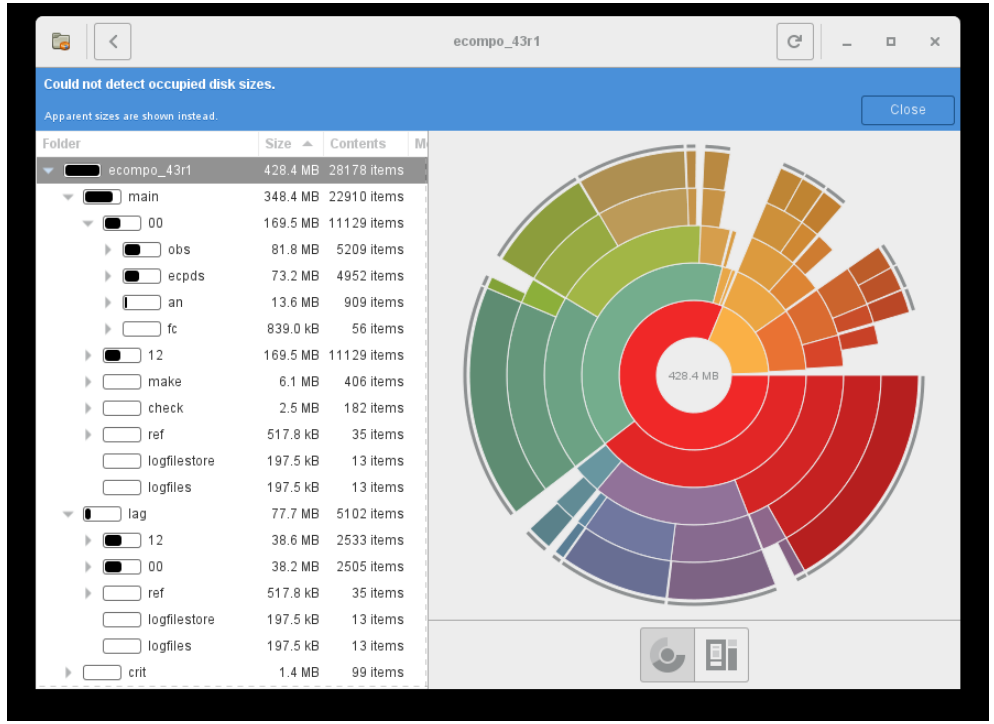




Python for visualisation: Fuse

- ecflow python client may mount suite content as a UNIX file system
- ls, find, mc, baobab for suite navigation
- Passive and/or active: suspend resume
- Fuse file access leads to server



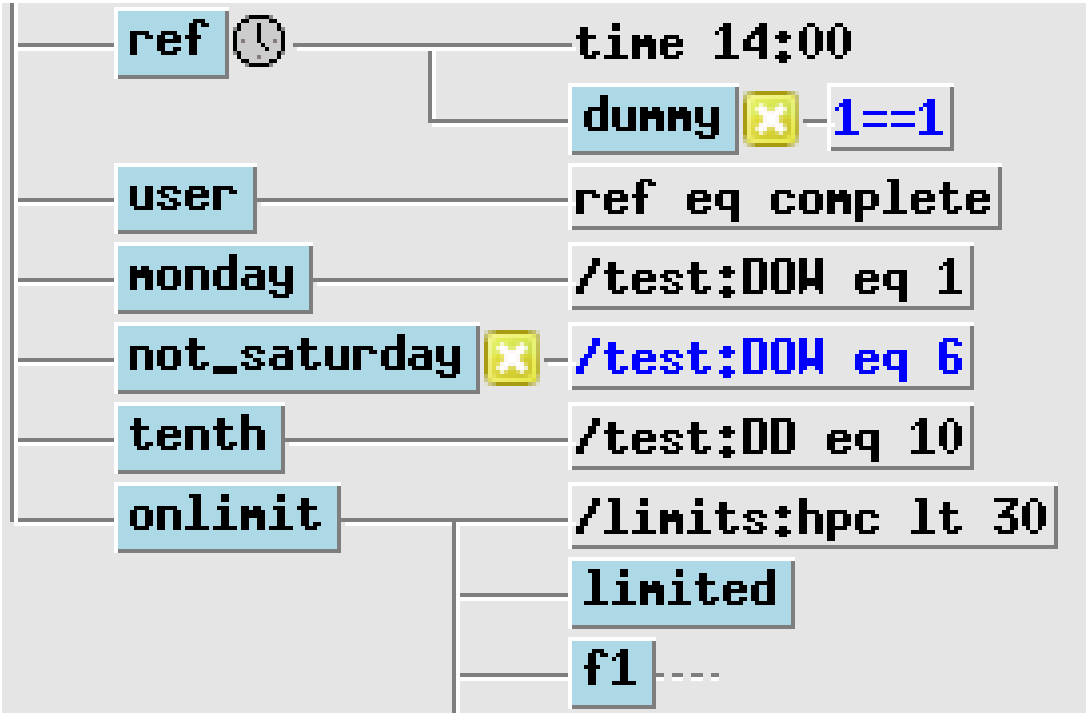



```
MSG:[17:58:12 12.1.2017] --log=get 100  :map

eurus:http/eod3 $ ls
ecflow_client.exe  era5hres_1.act   mofc.com.sus        verify.act
ecompo_43r1        era5hres_2       monit               vsms2@43333.RUNNING
ecompo_43r1.que    era5hres_2.act   monit.que           vsms2@43333.att
eda                era5hres_3       naj                 vsms2@43333.history
eda.que.sus        era5hres_3.act   naj.com.sus         vsms2@43333.log
ei                 events           o                   vsms2@43333.news
ei.que             events.com       o.que.sus           vsms2@43333.ping
era5eda_1          limits           paf                 vsms2@43333.png
era5eda_1.act      limits.unk       paf.que.sus         vsms2@43333.stats
era5eda_2          mc               sapp                vsms2@43333.url
era5eda_2.act      mc.que.sus       sapp.que.sus        vsms2@43333.zombies
era5eda_3          metops_test      tigge_lam_prod      wmolcdnv
era5eda_3.act      metops_test.que  tigge_lam_prod.que  wmolcdnv.que
era5hres_1         mofc             verify
eurus:http/eod3 $ cat vsms2@43333.ping
ping server(vsms2:43333) succeeded in 00:00:00.001908  ~1 millisecondseurus:http
/eod3 $ cat vsms2@43333.history
```

ecflow – Elements for suites

Attributes: Trigger, Complete with Date,Time, Limit, Cron

- Limit: `Limit({'mutex':1,'semaphore':5})`
- Date, Time: attached
  - To the same node ('and then') or not ('and')
  - To dummy task, referred by a Trigger
  - Goes down under suspended node!
- Trigger expression can refer to
  - State, Event, Meter, Variable
  - ECF_DATE, TIME, DOW, DOY, DD, MM, YY
    - real-time suites, it holds under suspend!
  - Limit: use eq, ne, lt, le, gt, ge operators!
  - Beware: Trigger to a Cron Task
  - Livelock? `ecflow_client --wait %CONDITION%`



Attributes: Trigger, Complete with Date, Time, Cron

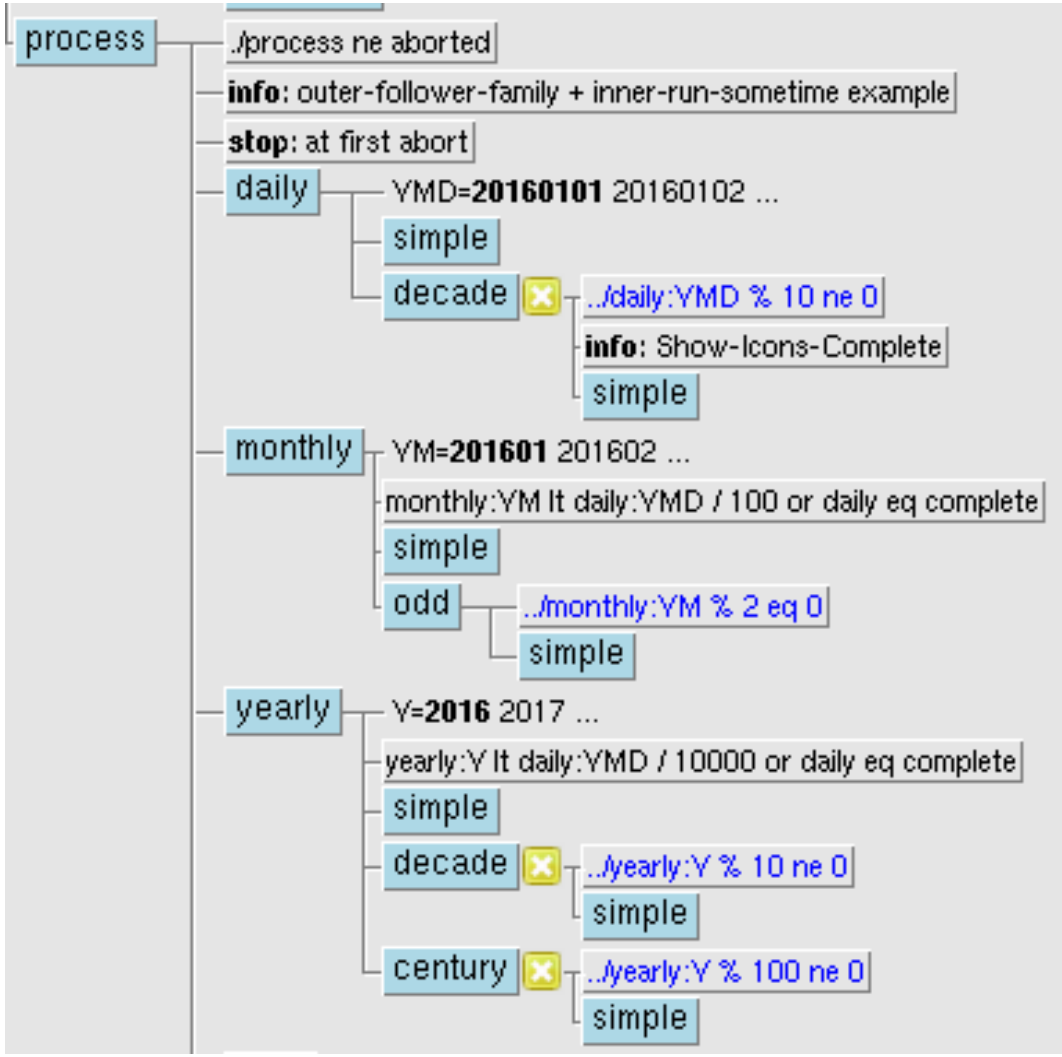
- **group time dependencies** in dedicated families + triggers
  - easy replacement when schedule changes
  - defstatus complete in not-real-time-mode

- **group external trigger** dependencies in dedicated families (dummy tasks)
  - easily replaced if reference suite changes
  - can be set defstatus complete in standalone-mode

- **'umbrella triggers'** to prevent evaluating multiple triggers all day long
  - 80-90 triggers for products generation depending on model meter

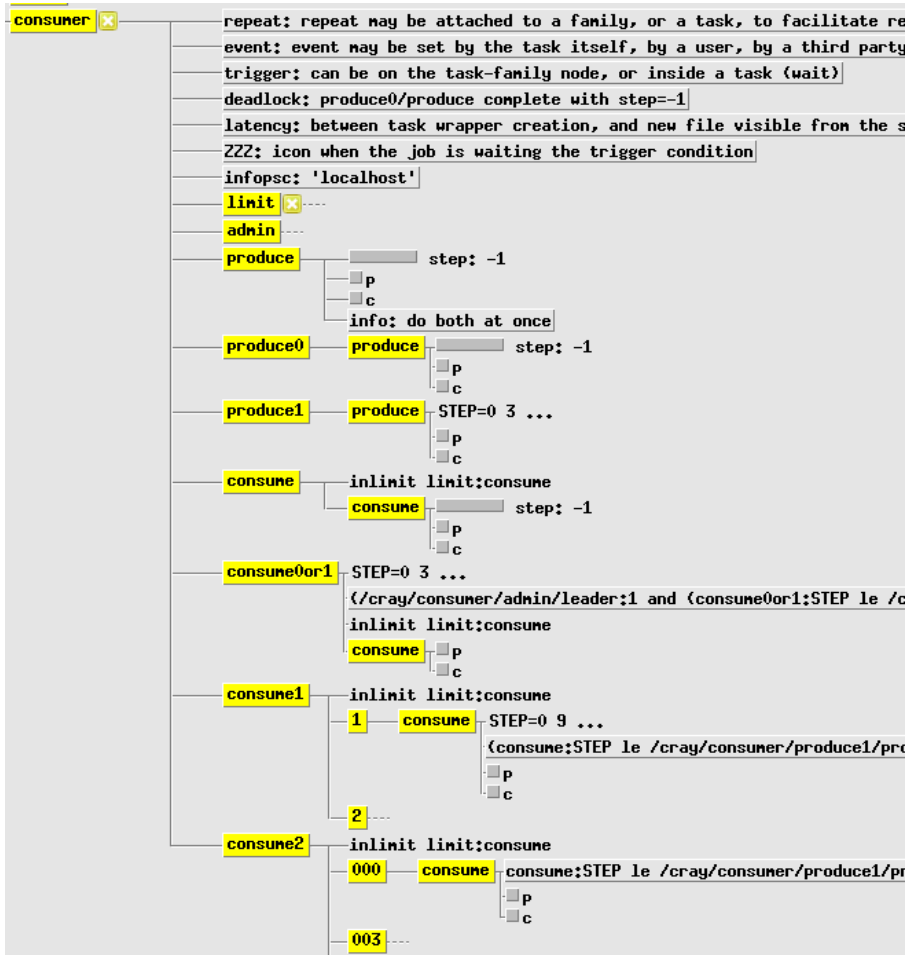
Attributes: Trigger, Complete with Repeat

- Mixing Repeat, Trigger, Date,Time
  - Can lead to a **deadlock**!
  - A chance to use **Simulate**!
- Multiple Repeat following each other
  - Not to delay the lead Family
  - Leads to long Trigger expressions!
    - It can be simplified using dummy node memorising “ok to go”
  - mars stage? Data size? IO? Swap?



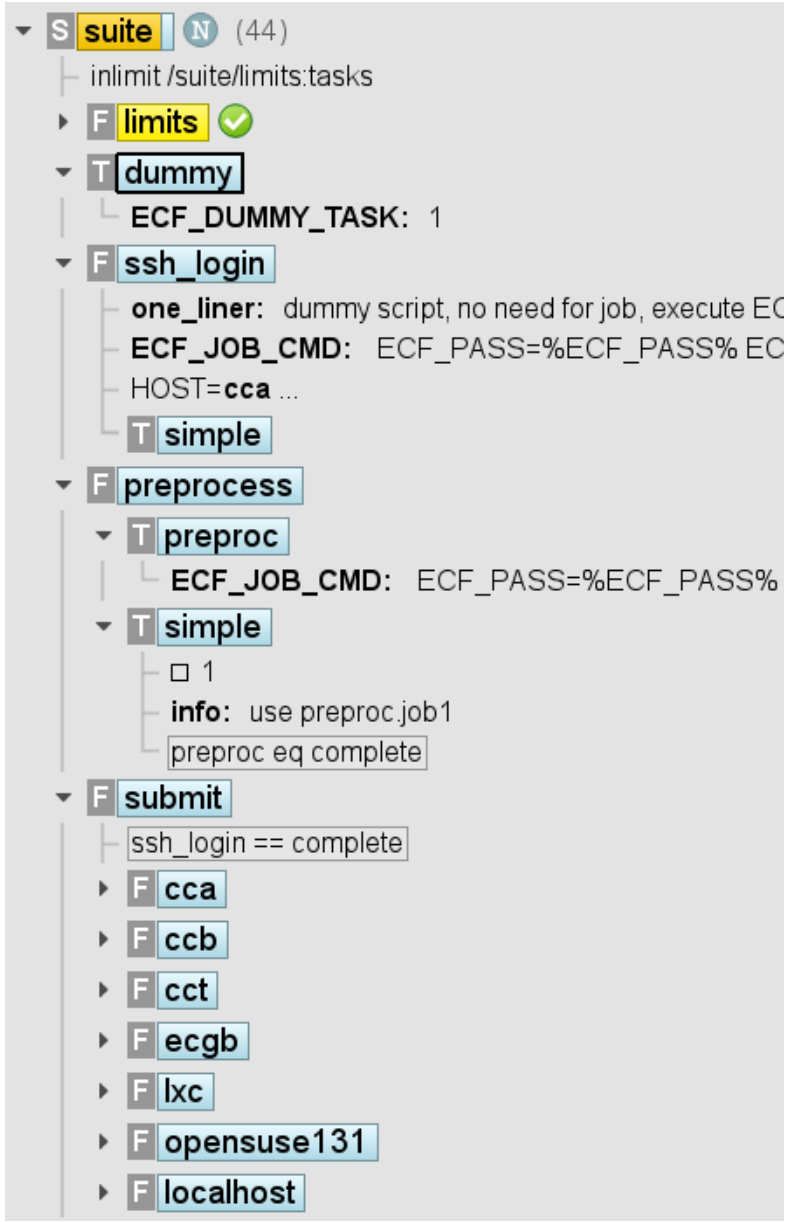
Attributes: Trigger with Repeat and Limit, Producer Consumer

- Producer-Consumer
  - from one to multiple tasks
  - Balance number of jobs with Limit
  - Time constraints
  - Available CPUs



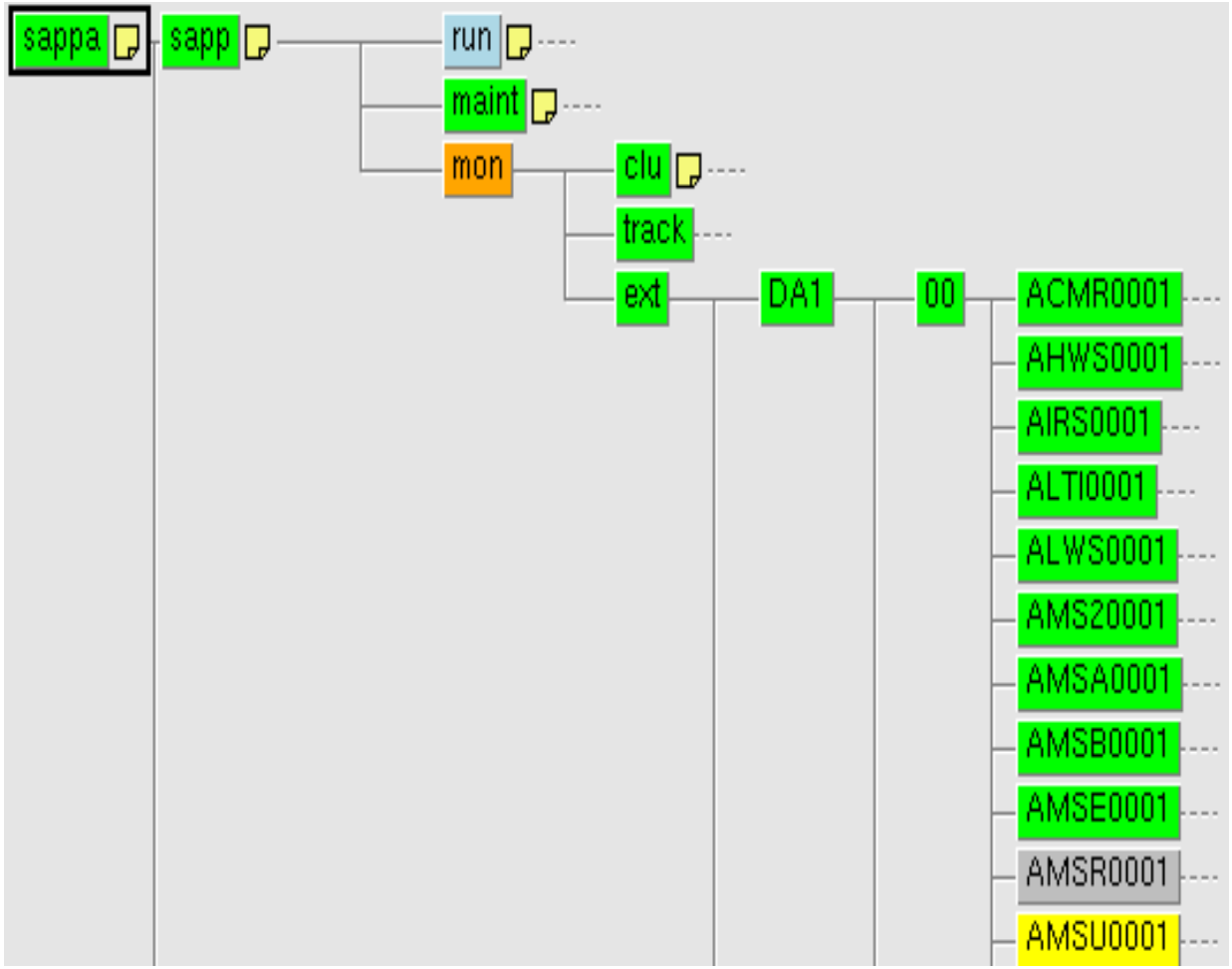
Jobs - use cases - 1

- Classic: wrapper, header, job, submit
- Dummy tasks: no wrapper, no job
  - Handle complex triggers, external dependencies, facilitate date/time update
  - Edit(ECF_DUMMY_TASK=1), Trigger("1==0")
- No task wrapper?
  - edit ECF_NO_SCRIPT 1
- No job, only submit, one-liners:
  - edit ECF_JOB_CMD "$do"
  - ECF_HOST ECF_PORT ECF_NAME, ECF_PASS
- Preprocessing only, no submit



Jobs - use cases - 2

- Same wrapper, headers for all tasks?
  - Suite variables make the difference
  - Variables for Job, manual, output
- Too many jobs to submit at once?
  - **Monitor** mode
  - Jobs started independently
  - child commands for status change: ECF_HOST, ECF_PORT, ECF_NAME, ECF_PASS=FREE
  - defstatus **suspended** # sh-def
  - **edit ECF_PASS FREE** # sh-def



Jobs – use case - 3

- ecflow as a **central** point
  - Collect-Share information
  - Reporting status
  - Re-Routing
  - Retrieving job information
  - Allows profiling with timeline
- ecflow as distributed fleet: Inter-server cooperation
  - Maintaining work during server and network outages
  - Handling of priorities, systems, tests
  - Sharing load
  - Sync suite: client to mirror status/variables
- Works in “soft” real-time (ECF_INTERVAL is 60 seconds): sleep %SLEEP:60%

Shell environment for suites?

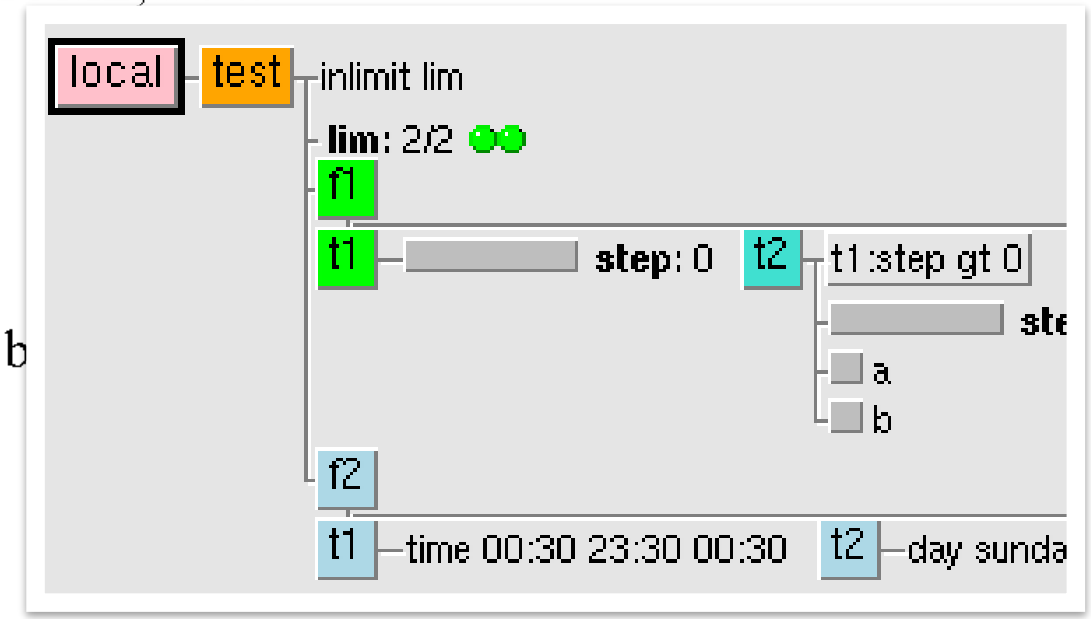
- Suite definition
  - ecflowrc: suite definition keywords as shell commands
  - suite.sh: expanded suite definition file .exp is generated (for node replace)
  - small **standalone** suite (maintenance!), **global scope**!
- Client-server interaction: ecflowrc for useful aliases,
  - `export ECF_PORT ECF_HOST`
- Shell as job environment: Ksh/Bash usually
  - trap, set –eux, PS4 variable for time stamps
- One script is enough? command line call/test, suite definition, task wrapper
  - `edit ECF_EXTN .sh; edit ECF_MICRO ^;  # sh-def`
  - `^include <file.man> ^manual …^end ^comment …^end (?) cat > /dev/null <<@@ … @@ # .ecf`
  - operational example: **monitor.sh**

Shell suites definition - ecflowrc

```
#!/bin/bash
# aka $HOME/.ecflowrc/ecflowrc
export ECF_PORT=$(($(id -u)+1500)) ECF_HOST=$(uname -n)
client="ecflow_client --port=$ECF_PORT --host=$ECF_HOST"
alias replace="$client --replace"; alias load="$client --load"

commands="autocancel clock complete cron date day defstatus edit event
family inlimit label late limit meter repeat suite task today
trigger endfamily endsuite endtask extern " # keywords
exec 3> ${SUITE:=test}.exp # expended def-file will be created
for fname in $commands; do source /dev/stdin <<@@
$fname() { echo $fname \${*} >&3; }
@@
done
alias time="echo time \${*} >&3"
```

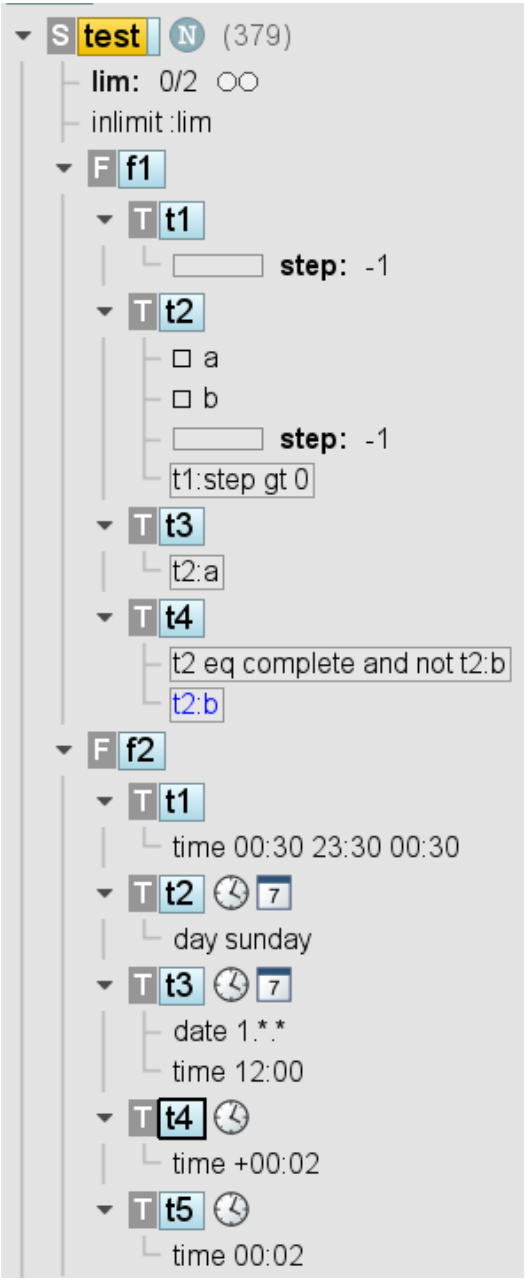
Shell Suite – example – suite.def

```
#!/bin/bash
. ecflowrc; ECF_HOME=$HOME/course
suite ${SUITE:=test}
edit ECF_INCLUDE $ECF_HOME; edit ECF_FILES $ECF_HOME; # r-x
edit ECF_HOME $ECF_HOME  # rwx
  defstatus suspended; limit lim 2; inlimit lim; edit SLEEP 20
  . for.def; . if.def ; . case.def # external definition families
  family f1
    task t1; meter step -1 100; # endtask # optional
    task t2; trigger t1:step gt 0; meter step -1 100; event a; event b
    task t3; trigger t2:a
    task t4; complete t2:b; trigger "t2 eq complete and not t2:b"
  endfamily;
  family f2
    for i in $(seq 1 5); do task t$i; case $i in 1)time 00:30 23:30 00:30;;
    2)day sunday;; 3)time 12:00; date 1.*.*;; 4)time +00:02;; 5)time 00:02;;
    esac; done
#endfamily # f2 #endsuite # SUITE # not necessary at the end!
```



Python Suite – example – suite.py

```
#!/usr/bin/env python2.7
import sys, os; path = "/home/ma/emos/def/o/def"; sys.path.append(path)
import ecf; from ecf import (Autocancel, Client, Clock, Complete, Cron, Date, Day, Defs,
Defstatus, Edit, Event, Extern, Family, Inlimit, Label, Late, Limit, Meter, Node, Repeat, Suite,
Task, Time, Today, Trigger)
home = os.environ['HOME'] + '/course'
def create(): return Suite("test").add(
   Edit(ECF_INCLUDE=home, ECF_FILES=home, ECF_HOME=home),
   Defstatus("suspended"), Limit("lim", 2), Inlimit("lim"),
   Family("f1").add(
      Task("t1").add(Meter("step", -1, 100)),
      Task("t2").add(Meter("step", -1, 100), Event("a"), Event("b"), Trigger("t1:step gt 0")),
      Task("t3").add(Trigger("t2:a")),
      Task("t4").add(Complete("t2:b"), Trigger("t2 eq complete and not t2:b"))),
   Family("f2").add(
      Task("t1").add(Time("00:30 23:30 00:30")),
      Task("t2").add(Day("sunday")),
      Task("t3").add(Time("12:00"), Date("1.*.*")),
      Task("t4").add(Time("+00:02")),
      Task("t5").add(Time("00:02"))))
if __name__ == "__main__":
      client = Client("localhost@%s" % os.getenv('ECF_PORT', '31415'));
      defs = Defs(); suite = create(); defs.add_suite(suite); client.replace("/test", defs)
```

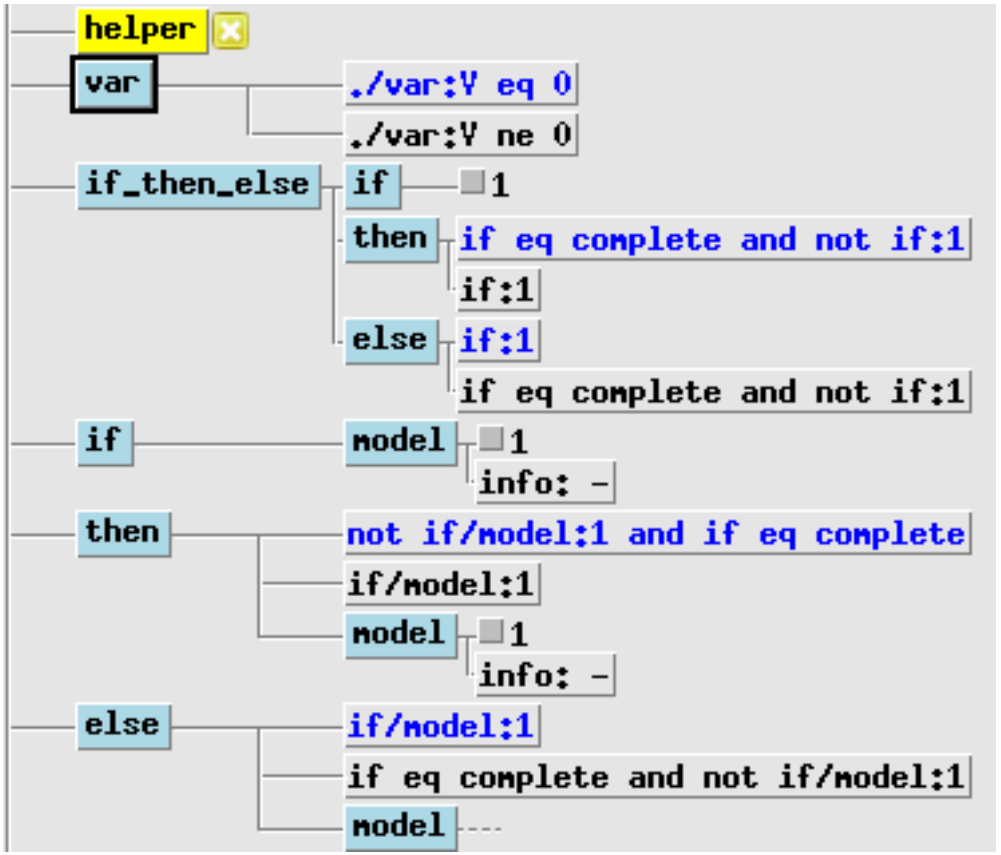


Suite loader – Shell - Python

```
#!/bin/bash
# bash ./suite-load.sh /test test.exp # syntax
# module unload ecflow || : ; module load ecflow/dev
# ecflow_client --ping || ecflow_start.sh
export ECF_PORT=$((1500+$(id -u))) ECF_HOST="$(uname -n)"
node=/${SUITE:=test} ; [[ "$1" != "" ]] && node=$1
file=$SUITE.exp      ; [[ "$2" != "" ]] && file=$2
ecflow_client --replace $node $file
[[ "$?" == "0" ]] && echo "# replaced $node from $file on $ECF_HOST@$ECF_PORT"
```

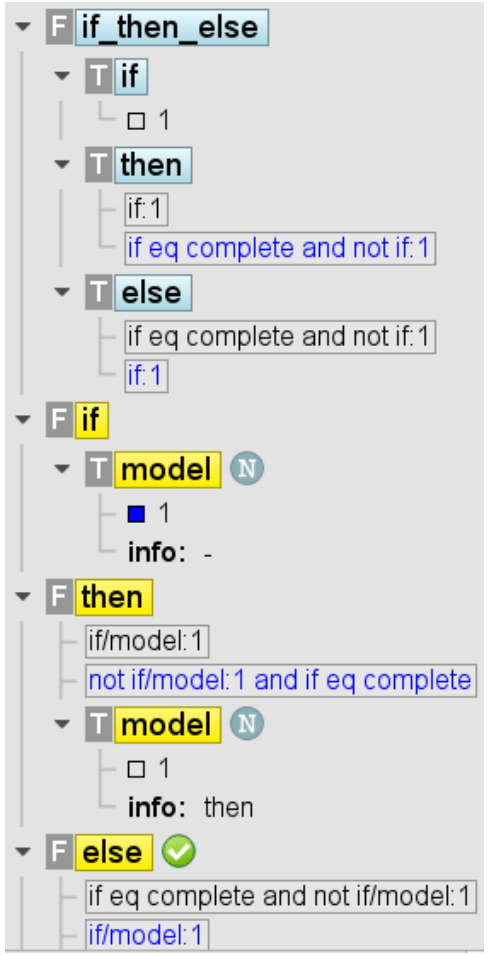
```
#!/usr/bin/env python2.7
import sys, os; path = "/home/ma/emos/def/o/def"; sys.path.append(path)
import ecf; from ecf import (Autocancel, Client, Clock, Complete, Cron, Date, Day, Defs,
Defstatus, Edit, Event, Extern, Family, Inlimit, Label, Late, Limit, Meter, Node, Repeat, Suite,
Task, Time, Today, Trigger)
home = os.environ['HOME'] + '/course'
def create(): return Suite("test").add()

if __name__ == "__main__":
    client = Client("localhost@%s" % os.getenv('ECF_PORT', '31415'));
    defs = Defs(); suite = create(); defs.add_suite(suite); client.replace("/test", defs)
```

Suite - Visualise – If block – reuse wrapper?



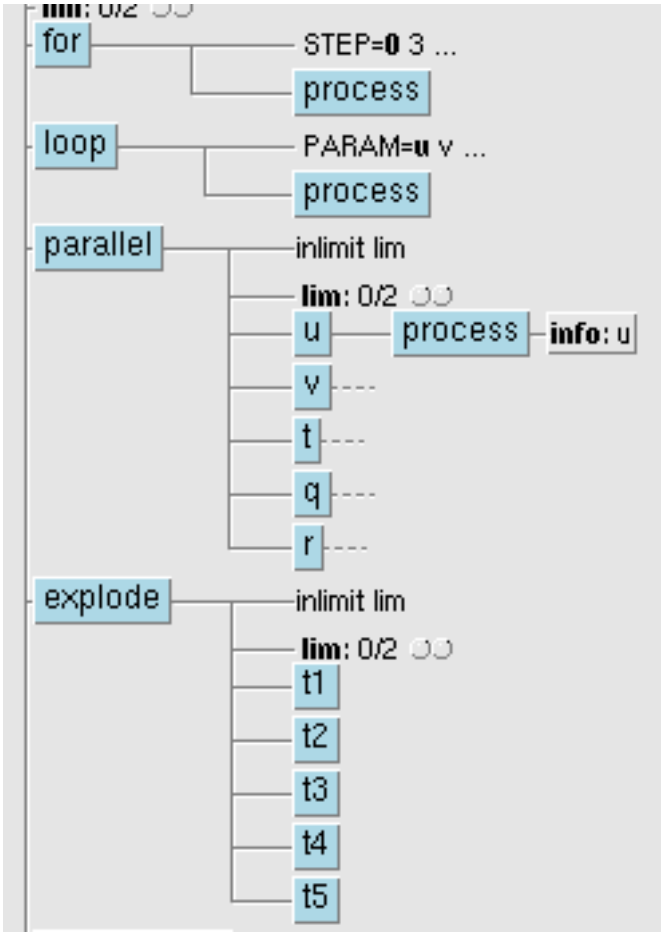

```
#!/bin/bash
task helper; defstatus complete

task var; edit V 1;
  complete ./var:V eq 0; trigger ./var:V ne 0

alias model="task model;event 1;label info -"
family if_then_else
task if; event 1
task then; trigger "if:1"
  complete "if eq complete and not if:1"
task else; complete "if:1"
  trigger "if eq complete and not if:1"
endfamily

family if # one script:
  model; endfamily
family then; trigger "if/model:1"
  complete "not if/model:1 and if eq complete"
  model; endfamily
family else; complete "if/model:1"
  trigger "if eq complete and not if/model:1"
  model; endfamily
```

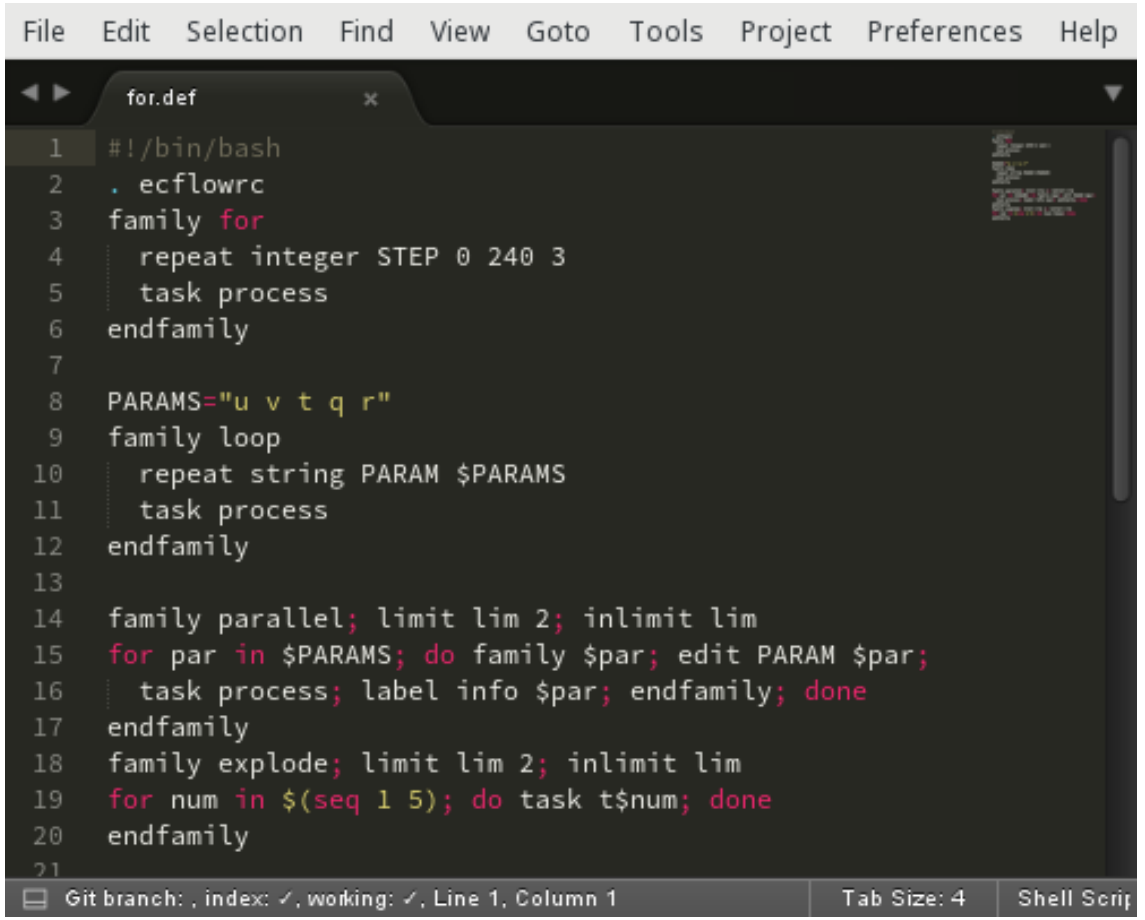
Suite - Visualise – If block



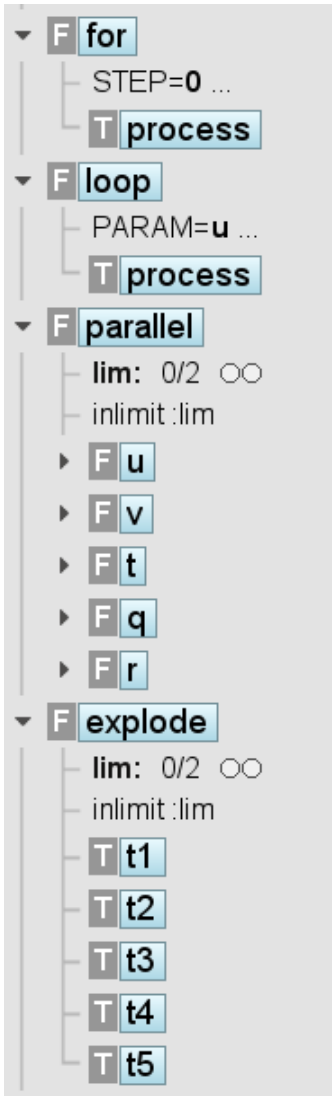

```
#!/bin/python
from ecf import *
def family_if(): return(
  Family("if_then_else").add(
    Task("if").add(Event(1)),
    Task("then").add(Trigger("if:1"),
     Complete("if==complete")), # FIXME
    Task("else").add(Complete("if:1"),
     Trigger("if eq complete and not if:1"))),

  Family("if").add(    # one script
    Task("model").add(Event(1))),
  Family("then").add(Trigger("if/model:1"),
    Complete("if eq complete and not if/model:1"),
    Task("model")),
  Family("else").add(Complete("if/model:1"),
    Trigger("if eq complete and not if/model:1"),
    Task("model")))
```

Suite - Visualise – for block



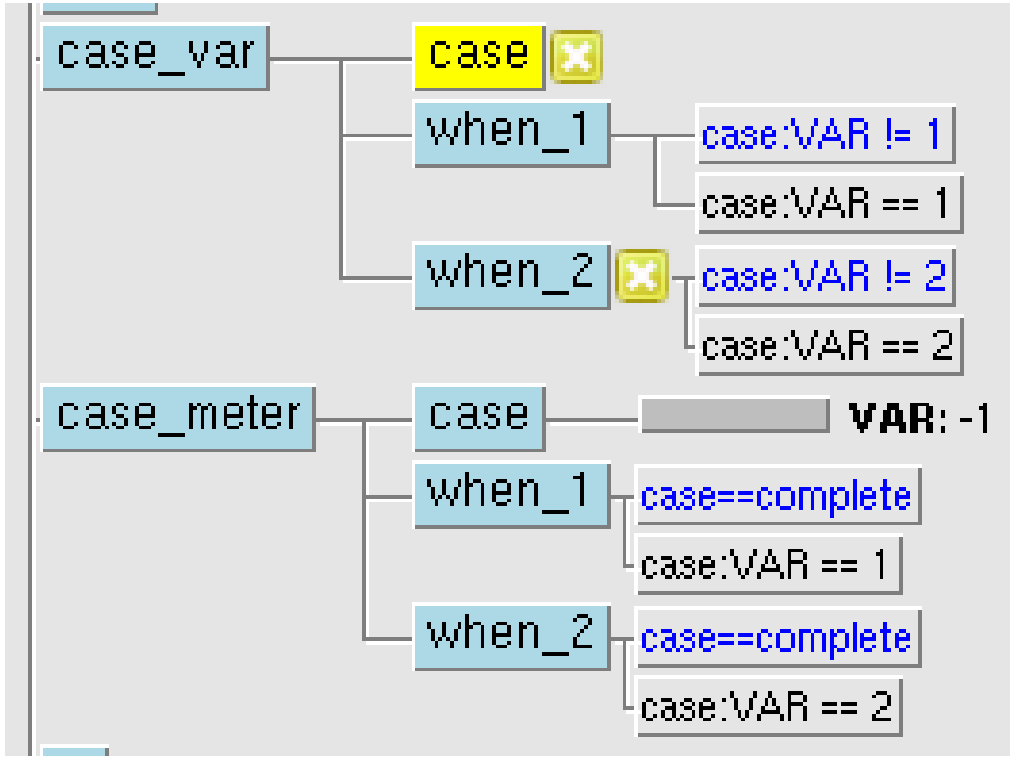


Suite - Visualise – for block



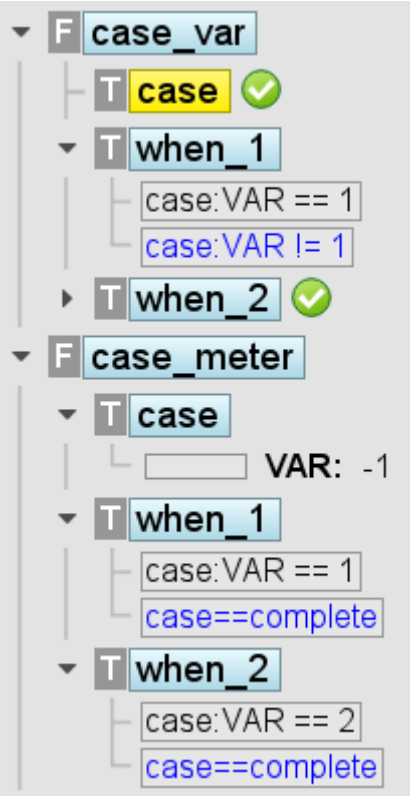

```
#!/usr/bin/env python
PARAMS = ["u", "v", "t", "r", ]
def process(): return Task("process")

def family_for(): return (
 Family("for").add(process(),
  Repeat(kind="integer", name="STEP",
       start=1, stop=240, step=3)),

 Family("loop").add(process(),
  Repeat("PARAM", PARAMS, kind="string")),

 Family("parallel").add(
    Limit("lim", 2), Inlimit("lim"),
    [Family(par).add(Edit(PARAM=par),
       process().add(Label("info", par)))
     for par in PARAMS]),
 Family("explode").add(Limit("lim", 2), Inlimit("lim"),
  [Task("t%d" % num) for num in xrange(1, 5+1)]))
```

Suite - Visualise – case block – exclusive call?



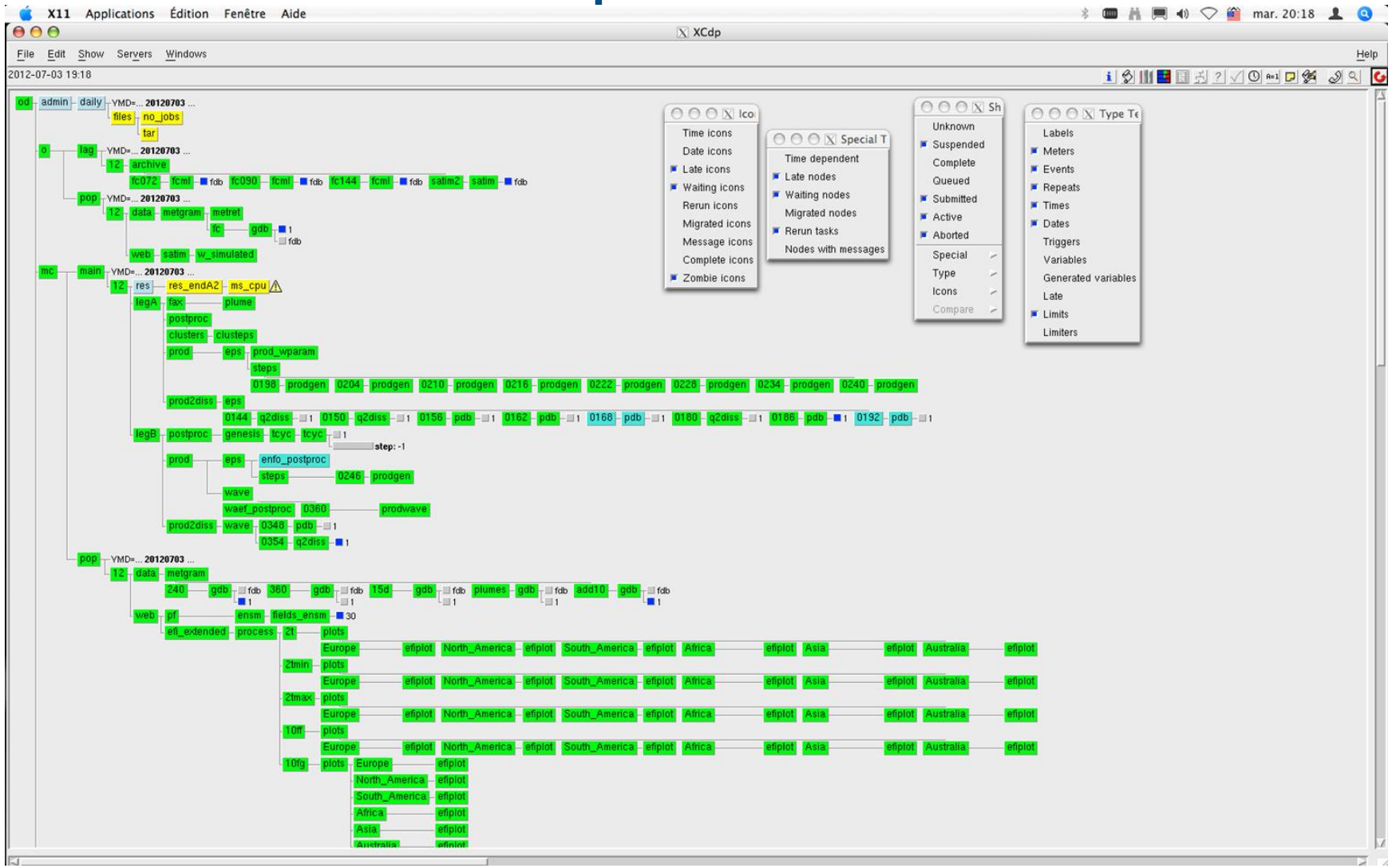

```
File  Edit  Selection  Find  View  Goto  Tools  Project  Preferences  Help
case.def
1  #!/bin/bash
2
3  family case_var
4    task case
5      defstatus complete
6      edit VAR 1
7
8    task when_1; trigger case:VAR == 1;
9      complete case:VAR != 1
10
11   task when_2; trigger case:VAR eq 2;
12     complete case:VAR ne 2
13 endfamily # case_var
14
15 family case_meter
16   task case; meter VAR -1 100
17
18   task when_1; trigger case:VAR == 1;
19     complete case==complete
20
21   task when_2; trigger case:VAR eq 2;
22     complete case eq complete
23 endfamily # case_meter
Git branch: , index: ✓, working: ✓, Line 17, Column 1   Spaces: 2   Shell Scrip
```

Suite - Visualise – case block




```
#!/bin/python
from ecf import *
def family_case(): return (
  Family("case_var").add(
    Task("case").add(Defstatus("complete"),
      Edit(VAR=1)),
    Task("when_1").add(Trigger("case:VAR==1"),
      Complete("case:VAR != 1")),
    Task("when_2").add(Trigger("case:VAR eq 2"),
      Complete("case:VAR ne 2"))),
  Family("case_meter").add(
    Task("case").add(Meter("VAR", -1, 100)),
    Task("when_1").add(Trigger("case:VAR==1"),
      Complete("case==complete")),
    Task("when_2").add(Trigger("case:VAR eq 2"),
      Complete("case eq complete"))))
```

Suite - Visualise – Operators



ECMWF Projects: Background

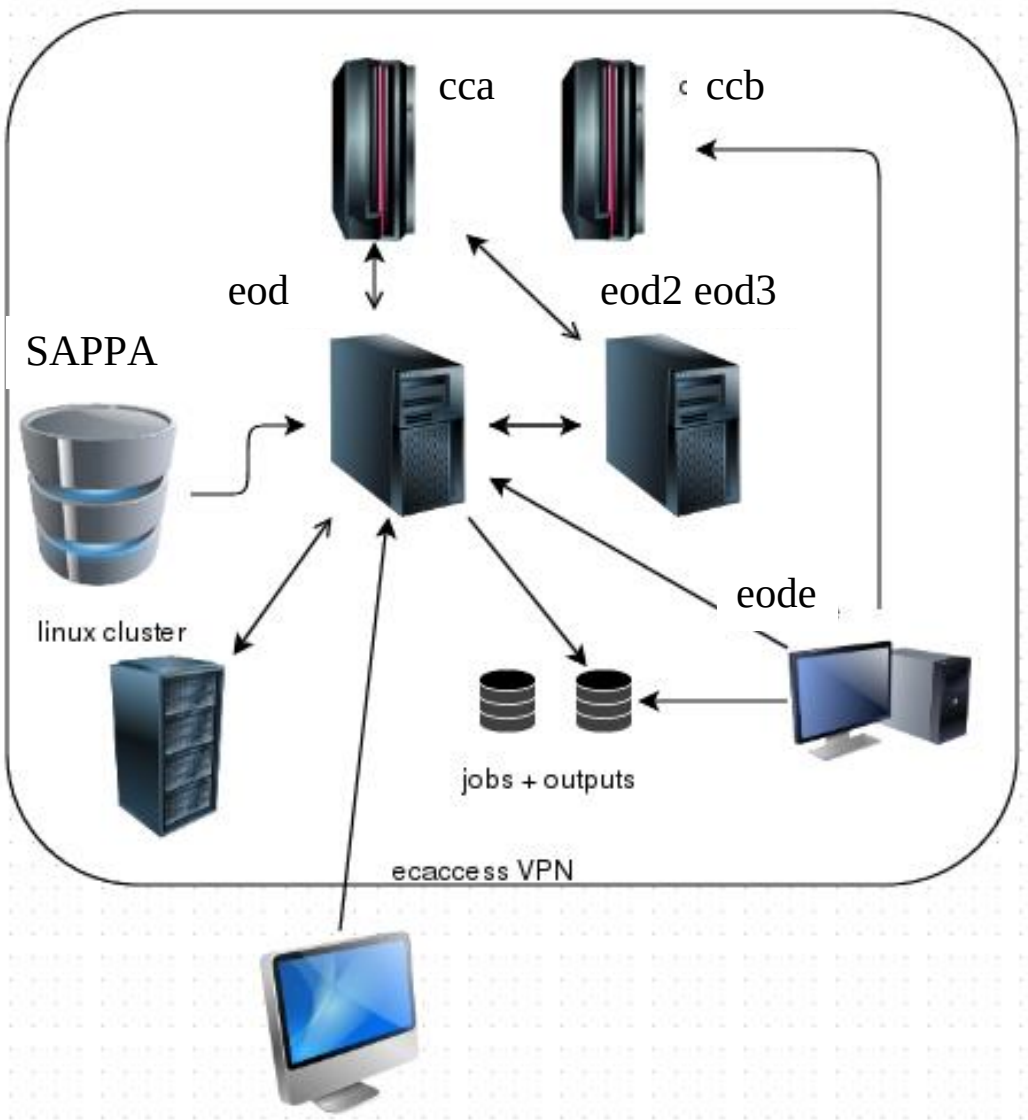
- ECMWF code runs on multiple platforms
- Software installation should be simultaneous across them all
  - Need ability to quickly revert changes if problems
- Need automated routine maintenance
- Need to handle both operational and non-operational tasks
- Numerous housekeeping tasks

Operational Systems

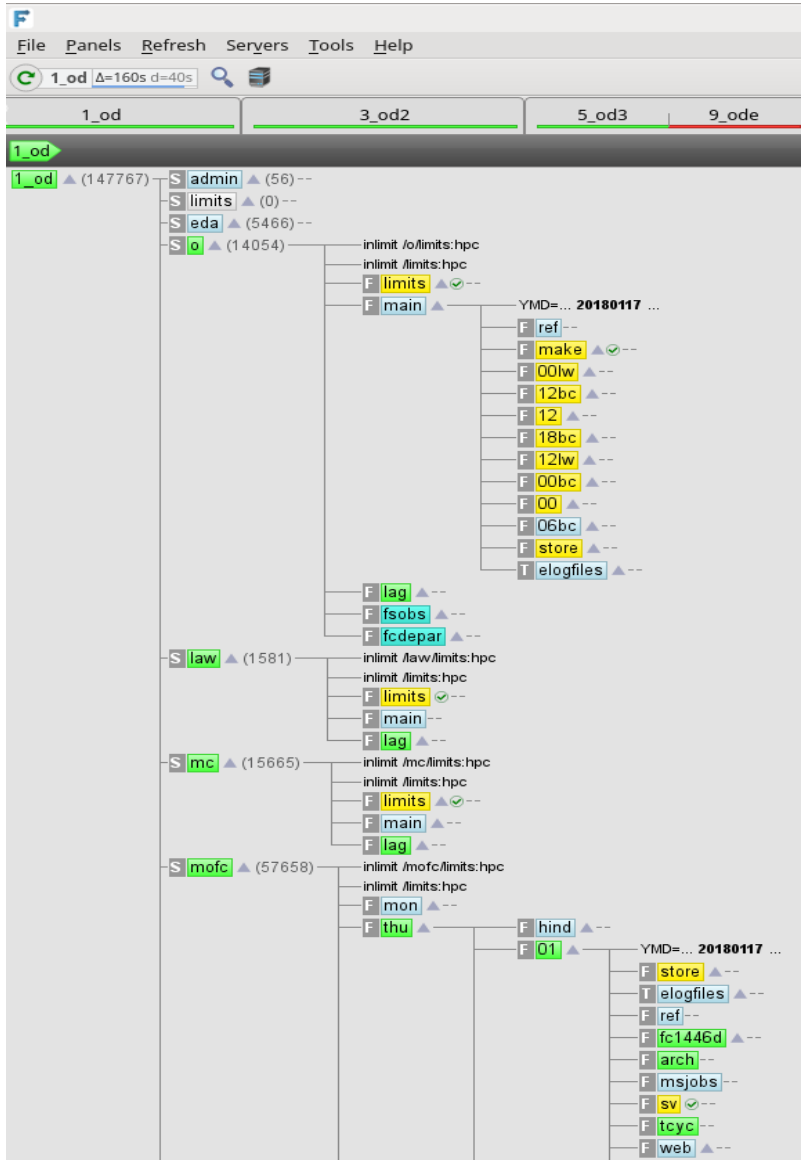
- Operationally we will run dozens of suites, tens of thousands of tasks
- Number of servers reflecting criticality
  - ode: tests and design mode
  - eod3: official e-suites monitored by Operators, special projects
  - eod2: higher criticality, seasonal suites
  - od: operational suites looping daily
- Servers hosted on linux workstation in Ops-room (with UPS), VM, or WS
- Access controlled
- Heterogeneous: tasks run on HPC, Linux Clusters, locally
- Suite structure separated by criticality: main-crit-lag-pop families
- Operators monitoring
- Watchdog tasks both internal and external to suites

Operation System: ecflow server

- Server is target agnostic:
  - ECF_JOB_CMD (submit)
  - ECF_KILL_CMD (kill)
  - ECF_STATUS_CMD (query)
- Variable to locate wrapper files:
  - ECF_FILES
- Variable to locate header files:
  - ECF_INCLUDE
- Checkpoint files:
  - Written /2min, back /4min
  - Duplicated /10min, stored /30min
- Cluster switch, host switch, storage host

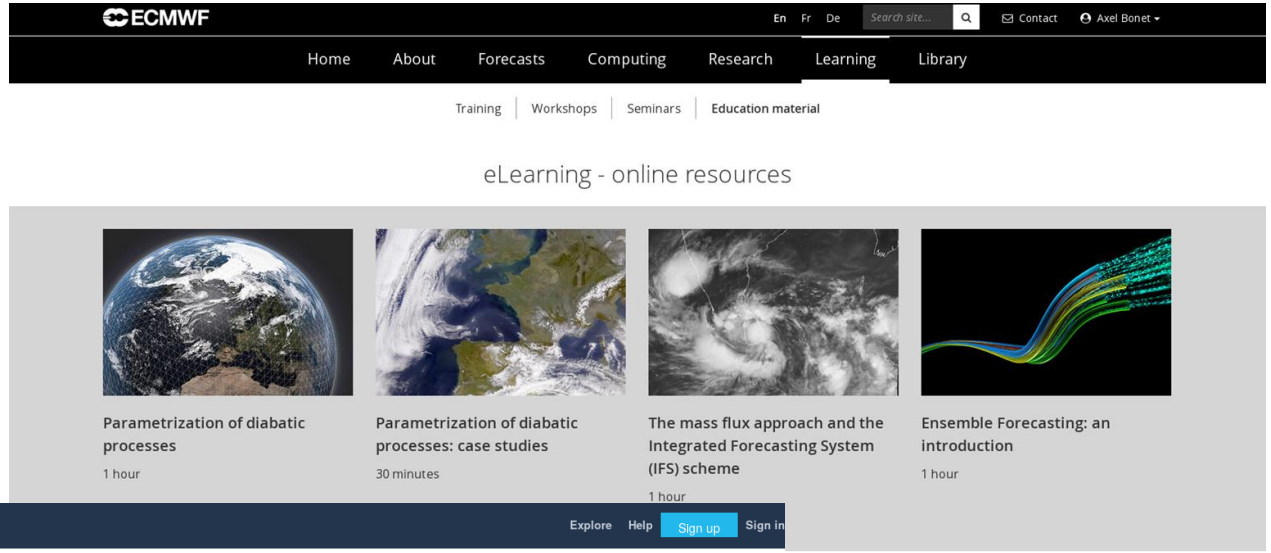


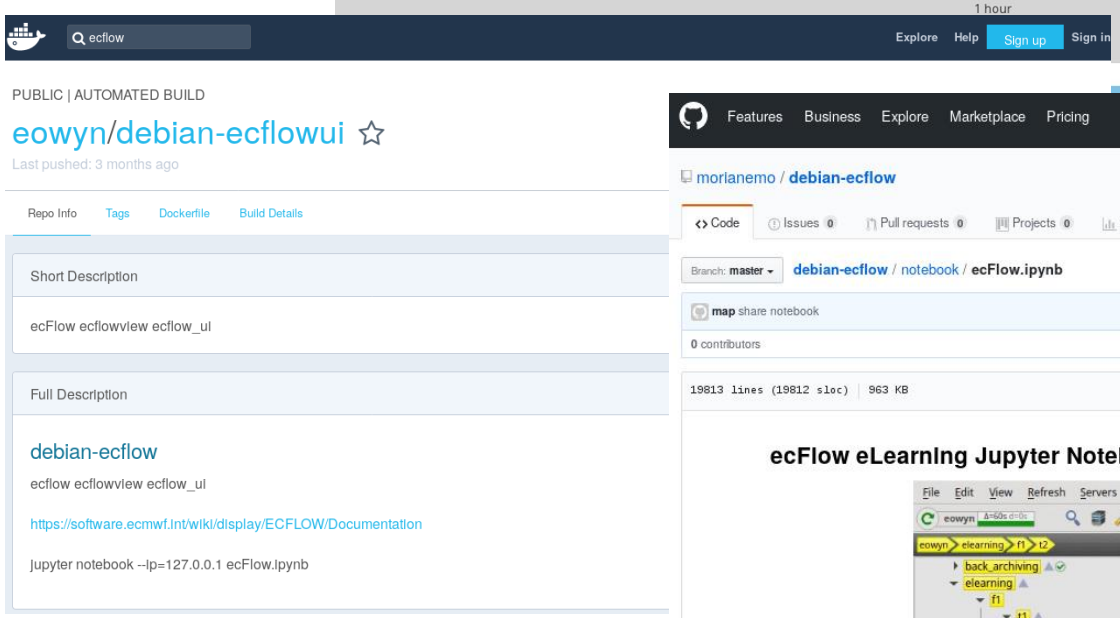
Operation System: servers - suites

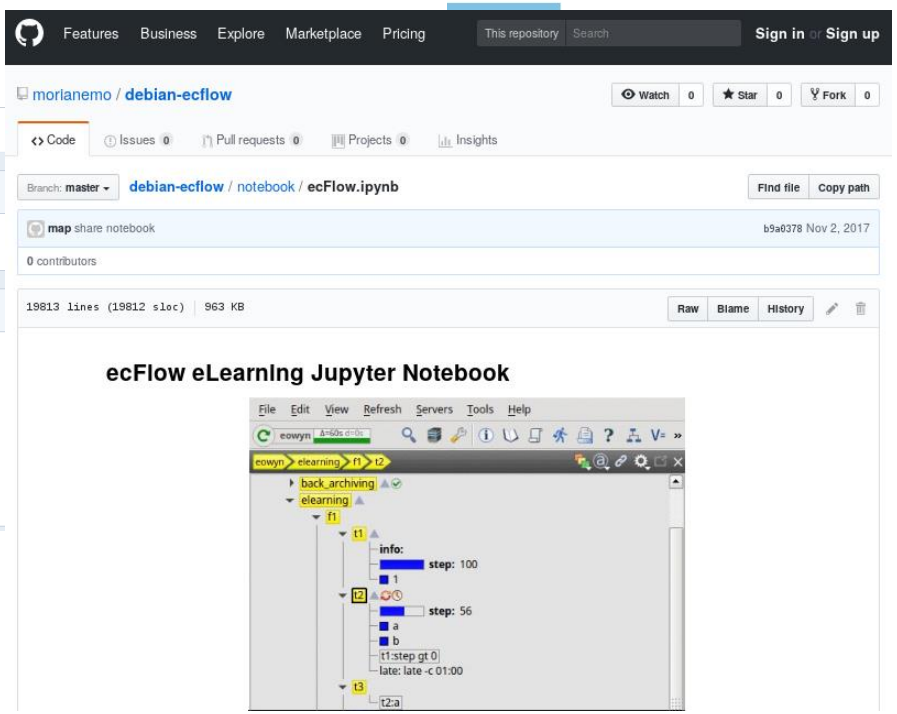


Additional content

- Elearning
  - elearning git repository
- Github, Docker, GUI with docker
- notebook







End Section

Server Configuration

- Server configuration variables:

```
ECF_HOME            # server admin directory
ECF_PORT            # port number
ECF_CHECK           # checkpoint file name
ECF_CHECKOLD        # backup checkpoint file name
ECF_LOG             # server log file name
ECF_CHECKINTERVAL   # [120], 600 sec
ECF_LISTS           # white list file name
ECF_DEBUG_SERVER # turn on debug mode
```

- Server log file:
  - Can be handled by client command
  - `ecflow_client --port 3141 --log=new  # [new|clear|flush]`

Key ecflow variables

- ECF_HOME, ECF_FILES, ECF_INCLUDE: input scripts
- ECF_HOME, ECF_OUT: job files, output
- Mandatory variables for jobs

```
ECF_HOST        # server hostname
ECF_PORT        # server port
ECF_NAME        # task path
ECF_PASS        # pseudo-random unique identifier
```

- Useful variable for jobs

```
ECF_TRYNO       # job occurrence number
ECF_HOSTFILE    # alternative host server list (server recovery)
ECF_RID         # job remote id (queuing id)
ECF_TIMEOUT     # interval between two attempts
ECF_DENIED      # to enable job exit with error before 24h
NO_ECF          # standalone mode (set to use)
```