
JULES From Scratch

Updated for the JULES Annual Meeting Sep 2023

Dr Toby Marthews



The JULES Land Surface Model

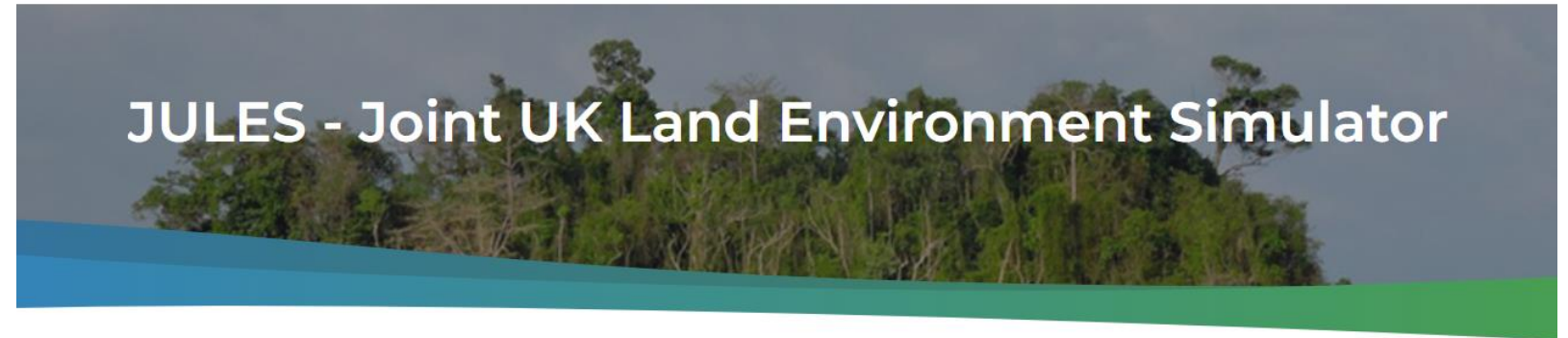
The main JULES website is:

<https://jules.jchmr.org/>

(maintained by myself since 2017).

The JULES website contains a wide range of information including how to get set up with JULES, what data you need and much other information.

The PDFs of all presentations given at previous Annual JULES Science meetings are on <https://jules.jchmr.org/meetings>, which is the easiest way to get a sense of what is happening in the wide international JULES Community of users and developers.

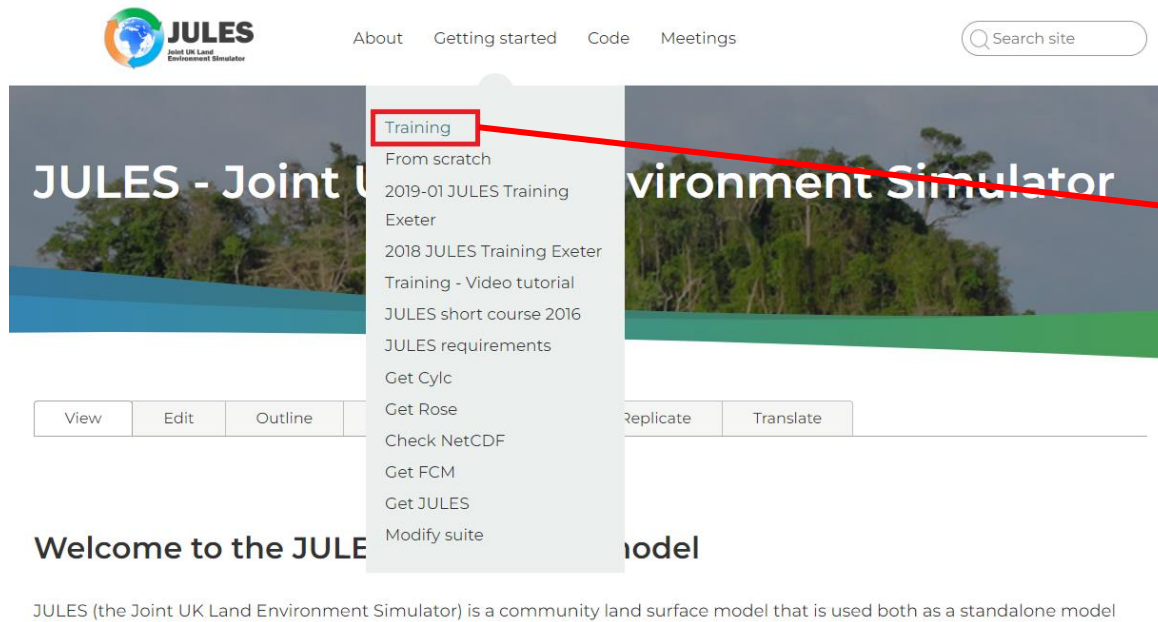


| | | | | | | |
|----------------------|----------------------|-------------------------|------------------------|---------------------------|---------------------------|---------------------------|
| View | Edit | Outline | Delete | Revisions | Replicate | Translate |
|----------------------|----------------------|-------------------------|------------------------|---------------------------|---------------------------|---------------------------|

Welcome to the JULES land surface model

JULES (the Joint UK Land Environment Simulator) is a community land surface model that is used both as a standalone model and as the land surface component in the Met Office Unified Model. JULES is a core component of both the Met Office's modelling infrastructure and NERC's Earth System Modelling Strategy. JULES is a major part of the UK's contribution to global model intercomparison projects (e.g. CMIP6) and is placed firmly at the cutting edge of international land surface modelling because of continual science development and improved accessibility.

The JULES Land Surface Model



The screenshot shows the JULES website home page. The navigation bar includes 'About', 'Getting started', 'Code', and 'Meetings', along with a search box. A dropdown menu is open over the 'Getting started' link, listing various training and documentation options. A red arrow points from the 'Training' option in the dropdown to the 'Training' page shown in the next screenshot.

JULES - Joint UK Land Environment Simulator

About Getting started Code Meetings

Search site

Training

From scratch

2019-01 JULES Training Exeter

2018 JULES Training Exeter

Training - Video tutorial

JULES short course 2016

JULES requirements

Get Cylc

Get Rose

Check NetCDF

Get FCM

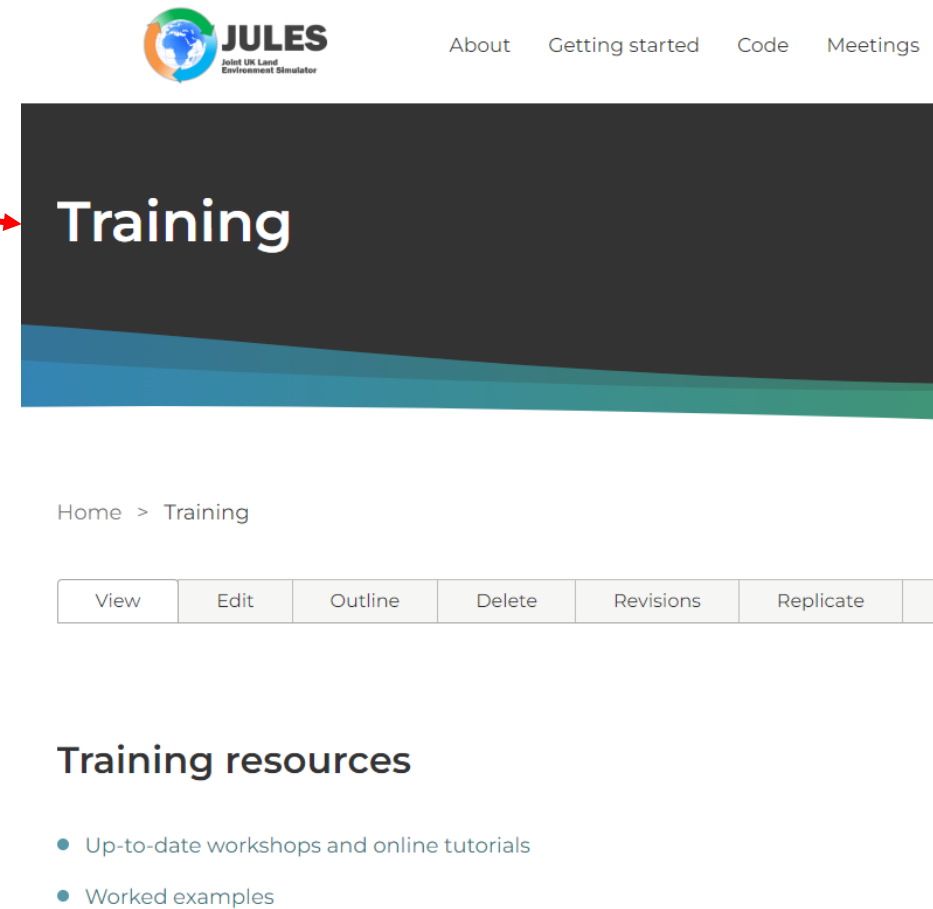
Get JULES

Modify suite

View Edit Outline Replicate Translate

Welcome to the JULES model

JULES (the Joint UK Land Environment Simulator) is a community land surface model that is used both as a standalone model



The screenshot shows the JULES website training page. The navigation bar is the same as the home page. The page title is 'Training'. Below the title, there is a breadcrumb trail 'Home > Training' and a set of navigation buttons: 'View', 'Edit', 'Outline', 'Delete', 'Revisions', and 'Replicate'. The main content area is titled 'Training resources' and contains a list of two items: 'Up-to-date workshops and online tutorials' and 'Worked examples'.

JULES - Joint UK Land Environment Simulator

About Getting started Code Meetings

Training

Home > Training

View Edit Outline Delete Revisions Replicate

Training resources

- Up-to-date workshops and online tutorials
- Worked examples

The JULES website has lots of training tutorials (many of them are actually hosted on the **JULES TRAC** <https://code.metoffice.gov.uk/trac/jules/> , which is a 'sister website' containing more developer-oriented material).

JULES From Scratch



The *JULES from Scratch* tutorial is my attempt to lead you through all you need to get started on JULES.

I (Toby) take you through setting up a JULES run right from the basics, including all required installation steps on a new UNIX/Linux platform (the only thing I assume is that your platform has NetCDF libraries already installed).

I wrote the first version of this when I took over maintaining the jules.jchmr.org website in May 2017 and it has been updated several times since (most recently Sep 2023).

Please note you may need a bit of familiarity with UNIX commands to accomplish the steps below. If you need a quick refresh of this, please see my 2-page **UNIX Basics** doc on <https://www.tobymarthews.com/resources.html> .

My instructions in this tutorial are designed to be general in the sense that they should be applicable to any UNIX/Linux platform.

- If you are using the **Jasmin.ac.uk Superdata cluster** (as I do in my example below), then I have some extra tips to help you get set up at <https://www.tobymarthews.com/jules-on-jasmin.html> .
- However, if you are using any sort of **Virtual Machine (VM)**, then I probably can't help you with set-up, so please instead see the UK Met Office help page at <https://code.metoffice.gov.uk/trac/jules/wiki/JULESVirtualMachine> .

From scratch



JULES From Scratch is an alternative how-to for those who don't work on a system where Cylc, Rose and FCM have already been installed and/or can't use the Met Office Virtual Machine (e.g. for security reasons). I'm also assuming little or no familiarity with any recent version of JULES.

In order to follow this tutorial you need:

- A **MOSRS login** (see [here](#)) (this is free to everyone, whether in the UK or not). You should also subscribe to the **email support** lists (see [here](#)), by the way.

JULES From Scratch



but a slightly modified version of JULES called vn6.0_hj, the fcm co command that you will need to get hold of that version of JULES I've given you there

I should mention that there is also a **video version of this tutorial** on <https://jules.jchmr.org/scratch>, however it's a bit out of date now (I did it in July 2021 for an earlier version of the *JULES From Scratch* slides). Also, I promised a video version of the grid-based run in there, which I have still not got around to doing (...).

Where this video differs from the updated slides, please follow these updated slides (e.g. suite `u-cg242` has been replaced now by `u-cz674`). At some point I hope to redo the video, but until then this version is still very useful.

JULES From Scratch: Cylc, Rose, FCM and JULES



From <https://jules.jchmr.org/scratch> :

- Access your command prompt (either locally or on a server) using e.g. MobaXTerm (Windows) or XQuartz (Mac).
- You will need to install Cylc, Rose, FCM and, of course, a version of the JULES model itself.
- If they are not there, you will need to install local versions of Cylc, Rose and FCM ON JASMIN (it's not as hard as it sounds):
- These steps also set up your user profile for JULES.
- Start with Cylc, then do Rose, FCM and JULES in that order.

YOU WILL NEED CYLC, ROSE and FCM:

Ideally, these three will be installed globally by your system administrator, but if that isn't possible (or you are the system administrator) then you will need to go through some/all of these steps for a local installation. First check what is on your system by putting these commands into a UNIX shell:

```
cd ~
echo $SHELL
#You need to be in a bash shell, so if this returns anything except "/bin/bash" ask your system administrator how you can get into bash
xmessage -center hello!
#You need to have X11 forwarding activated. If this opens a little window to say Hello then you are OK, but if it says "Can't open display" then check (i) you did use "-X" in the options to log in on your ssh command and (ii) your terminal program (e.g. MobaXTerm, XQuartz) has been set up to accept X11 Forwarding.
xclock
xeyes
#xclock and xeyes are alternative tests for X11 forwarding: these both try to open separate windows and will fail if X11 is not activated (depending on how fun your IT Support people are, you may have more of these things installed http://cyber.dabamos.de/unix/x11/ and can try them too).
cat ~/.bashrc
#Check your .bashrc file and make sure that it has been set up correctly. For example, on JASMIN you may need to have a PATH= command in there (described here) to get Cylc, Rose and FCM working.
#The best way to sort out your .bashrc is to copy someone else's on the system. On JASMIN, you are welcome to copy any commands you see in my one: look at geany ~tmarthews/.bashrc & (access is not restricted).
```

NOW **GO THROUGH ALL FOUR SUBPAGES** BELOW (even if some/all of them are already installed, your personal profile settings may not be correct for using JULES so please do anyway go through the STEPs in these pages).

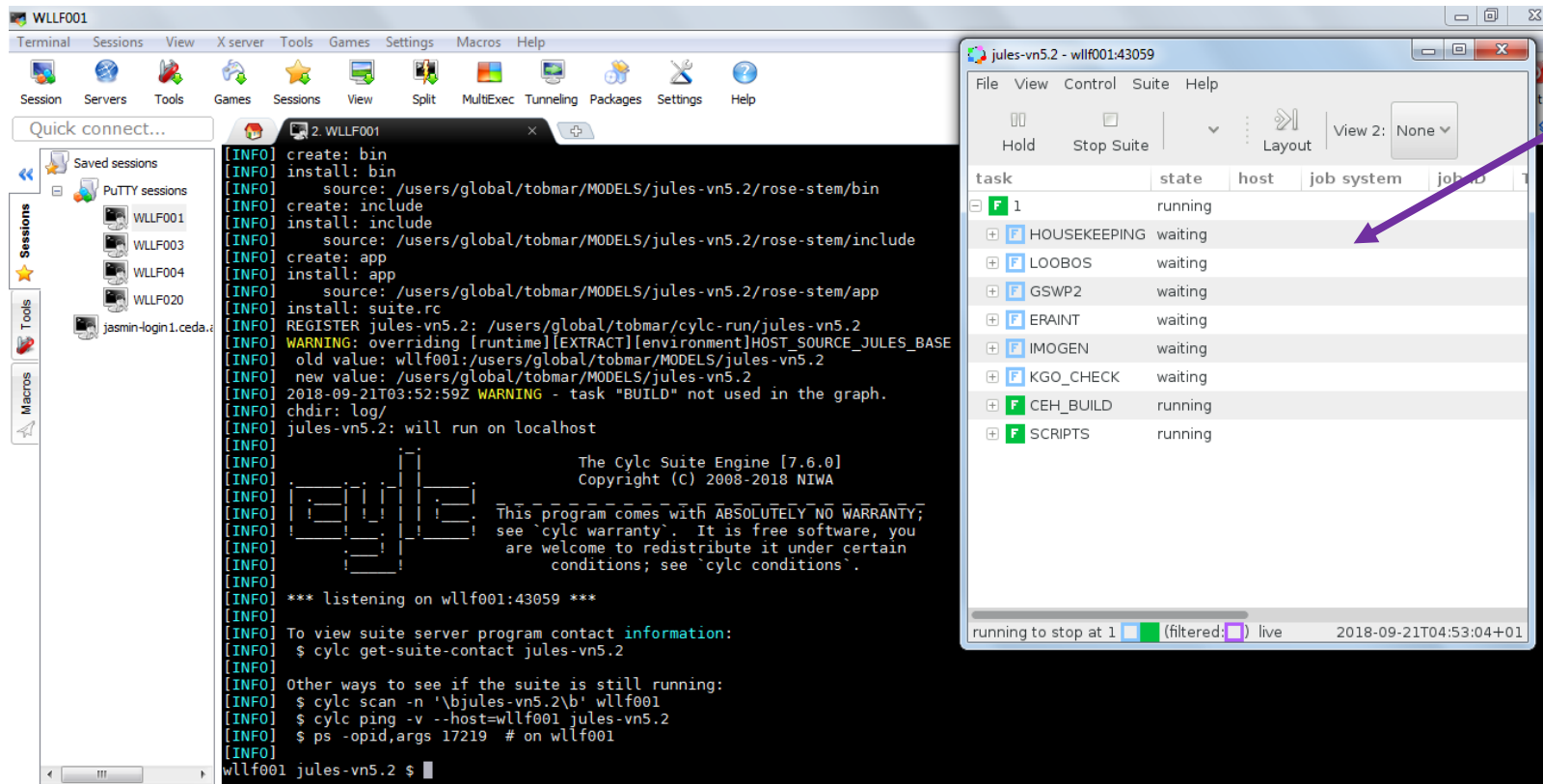
- Install Cylc from GitHub following these steps
- Install Rose from GitHub following these steps (inc. what goes in your ~/.metomi/rose.conf file, **which each user needs to set**)

JULES From Scratch: Check installation



We can't yet do a run of JULES, but I can show you some screenshots of what running it looks like:

- First, some text comes up on the command line including an ASCII-art "cylc" (see black background text below)
- Next, a new window appears called the *Cylc GUI* window



```
[INFO] create: bin
[INFO] install: bin
[INFO] source: /users/global/tobmar/MODELS/jules-vn5.2/rose-stem/bin
[INFO] create: include
[INFO] install: include
[INFO] source: /users/global/tobmar/MODELS/jules-vn5.2/rose-stem/include
[INFO] create: app
[INFO] install: app
[INFO] source: /users/global/tobmar/MODELS/jules-vn5.2/rose-stem/app
[INFO] install: suite.rc
[INFO] REGISTER jules-vn5.2: /users/global/tobmar/cylc-run/jules-vn5.2
[INFO] WARNING: overriding [runtime][EXTRACT][environment]HOST_SOURCE_JULES_BASE
[INFO] old value: wllf001:/users/global/tobmar/MODELS/jules-vn5.2
[INFO] new value: /users/global/tobmar/MODELS/jules-vn5.2
[INFO] 2018-09-21T03:52:59Z WARNING - task "BUILD" not used in the graph.
[INFO] chdir: log/
[INFO] jules-vn5.2: will run on localhost
[INFO]
[INFO] The Cylc Suite Engine [7.6.0]
[INFO] Copyright (C) 2008-2018 NIWA
[INFO]
[INFO] This program comes with ABSOLUTELY NO WARRANTY;
[INFO] see 'cylc warranty'. It is free software, you
[INFO] are welcome to redistribute it under certain
[INFO] conditions; see 'cylc conditions'.
[INFO]
[INFO] *** listening on wllf001:43059 ***
[INFO]
[INFO] To view suite server program contact information:
[INFO] $ cylc get-suite-contact jules-vn5.2
[INFO]
[INFO] Other ways to see if the suite is still running:
[INFO] $ cylc scan -n '\bjules-vn5.2\b' wllf001
[INFO] $ cylc ping -v --host=wllf001 jules-vn5.2
[INFO] $ ps -opid,args 17219 # on wllf001
[INFO]
wllf001 jules-vn5.2 $
```

| task | state | host | job system | job id |
|--------------|---------|------|------------|--------|
| 1 | running | | | |
| HOUSEKEEPING | waiting | | | |
| LOOBOS | waiting | | | |
| GSWP2 | waiting | | | |
| ERAINT | waiting | | | |
| IMOGEN | waiting | | | |
| KGO_CHECK | waiting | | | |
| CEH_BUILD | running | | | |
| SCRIPTS | running | | | |

New 'Cylc GUI' window showing the progress of each app in the JULES run.

If you close this window by accident before they complete, type:

```
cylc gscan &
```

and double-click on the right job to find it again.

Terminology:

JOB = Your JULES run/suite

APP = a 'subtask' of the job

JULES From Scratch: Check installation



A run then progresses to completion something like this (each line is an 'app', which is a subtask / component of the run):

| task | state | host | job system | job ID |
|--------------|---------|------|------------|--------|
| task 1 | running | | | |
| HOUSEKEEPING | waiting | | | |
| LOOBOS | waiting | | | |
| GSWP2 | waiting | | | |
| ERAIN | waiting | | | |
| IMOGEN | waiting | | | |
| KGO_CHECK | waiting | | | |
| CEH_BUILD | running | | | |
| SCRIPTS | running | | | |

| task | state | host | job system | job ID |
|--------------|-----------|------|------------|--------|
| task 1 | running | | | |
| HOUSEKEEPING | queued | | | |
| LOOBOS | queued | | | |
| GSWP2 | running | | | |
| ERAIN | running | | | |
| IMOGEN | running | | | |
| KGO_CHECK | queued | | | |
| CEH_BUILD | succeeded | | | |
| SCRIPTS | running | | | |

| task | state | host | job system | job ID | T-submit | T-sta |
|--------------|--------------------------|------|------------|--------|----------|-------|
| task 1 | stopped with 'succeeded' | | | | | |
| HOUSEKEEPING | succeeded | | | | | |
| LOOBOS | succeeded | | | | | |
| GSWP2 | running | | | | | |
| ERAIN | running | | | | | |
| IMOGEN | running | | | | | |
| KGO_CHECK | running | | | | | |
| CEH_BUILD | running | | | | | |
| SCRIPTS | running | | | | | |



Note 1: The “Stop Suite” button doesn’t actually stop the whole suite (!!) - see ‘Troubleshooting’ slide below

Note 2: The app lines will all disappear when the last one completes (there’s no option for changing that).

Note 3: This is a JULES run with lots of apps just intended as an example: most usually you will only have TWO apps: *fcm_make* and *jules*.

JULES From Scratch: Doing your own simulation



OK: Now we should all be set up and we know what a run should look like. How can we actually use JULES for something?

In common with any model of this type, JULES needs THREE elements to do a simulation:

*Driving data,
Ancillary/prescribed data and
Control files*

(together these are called the **model configuration**). I've put advice on where to get these from on <https://jules.jchmr.org/getting-started> .

The control files usually come in the form of a *Rose suite*, which is what we need to look at next.

Here's how to get started as a JULES user:

- Get started
- Email/support lists
- Other files you need (configuration files):
 1. Driving data
 2. Ancillary data
 3. Control files
- Analysis tools

JULES From Scratch: Rose suites



A *Rose suite* is a 'container' for *apps*. JULES Rose suites usually have only two apps that run sequentially: *fcm_make* (the compilation step) and *jules* (actually running the model) (see my FAQ about Rose suites on <https://jules.jchmr.org/rose-suites>).

For example, I have a suite `u-cm322`, stored on my system at `~/roses/u-cm322/` (which from now on I'm going to refer to as `$RSUITE`; `u-cm322` is the *IDX* code for this suite):

```
export RSUITE=$HOME/roses/u-cm322
echo $RSUITE
```

...which e.g. looks like this on JASMIN:

```
[tmarthews@cylc1 ~]$ export RSUITE=$HOME/roses/u-cm322
[tmarthews@cylc1 ~]$ echo $RSUITE
/home/users/tmarthews/roses/u-cm322
[tmarthews@cylc1 ~]$
```

I can open this suite in two different ways:

1. On UNIX using the Rose Edit GUI: `rose edit -C $RSUITE &`
2. On UNIX opening the four most important individual parts in a text editor:
`$RSUITE/app/fcm_make/rose-app.conf $RSUITE/app/jules/rose-app.conf &` `nedit $RSUITE/rose-suite.info $RSUITE/suite.rc`



JULES From Scratch: Rose suites



As mentioned above, you can edit a Rose suite in TWO different ways:

1. On UNIX using the Rose Edit GUI: `rose edit -C $RSUITE &`

Tip: Always change your settings in the Editor so that View -> View All Ignored Variables is selected. This is because the search box on the main screen only searches visible parameters (e.g. try searching for `L_veg_compete`).

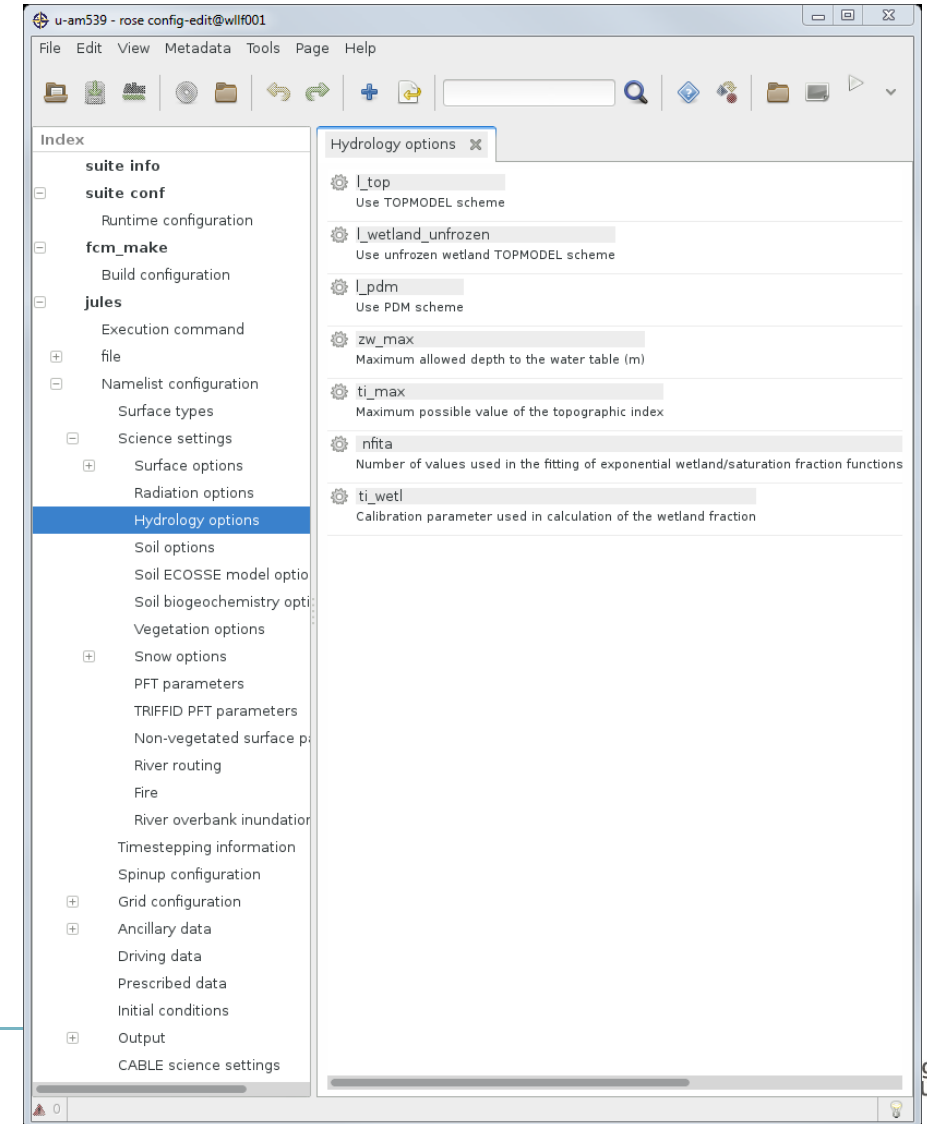
2. On UNIX, open the four most important textfiles in the suite using a text editor:

```
nedit $RSUITE/rose-suite.info $RSUITE/suite.rc $RSUITE/app/fcm_make/rose-app.conf  
$RSUITE/app/jules/rose-app.conf &
```

Tip: it's the last of these four textfiles that contains all the parameters and options for the JULES run.

When editing JULES suites, all parameters are explained on the JULES manual pages so I recommend to keep that open at the same time too (there is a search box there for you to find any parameter you are unfamiliar with):


<http://jules-lsm.github.io/latest/namelist/contents.html>



JULES From Scratch: Rose suites



See information on <https://jules.jchmr.org/getting-started#control-files> for how to get hold of Rose suites. There are basically two ways:

1. Download a suite from one of the (many) standard configurations on <https://jules.jchmr.org/configurations>
2. On UNIX using the *Rose Suite Discovery Engine*: `rosie go` 

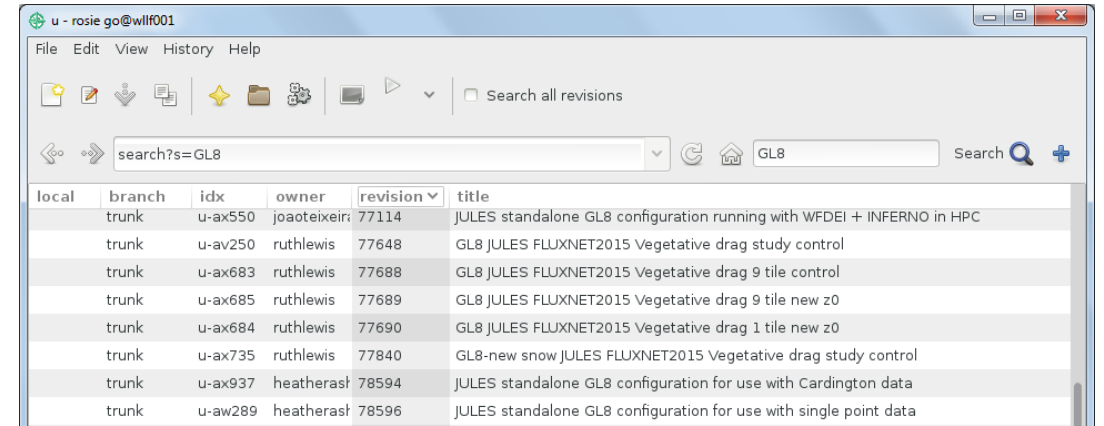
(select Edit → the ‘u’ data source, then use the search box). To download, choose:

- **Checkout Suite** to take a duplicate of it
- or **Copy Suite** to use it for your own work (you’ll get a new suite ID and this will belong to you).
- Every suite has its own URL based on its IDX code (e.g. *u-cm322*) and you can use this to see the changes to the suite over time, e.g.

<https://code.metoffice.gov.uk/trac/roses-u/browser/c/m/3/2/2>

(to see a changeset, click on the red number (not the cog) in the **Rev** column → View changes). Or, equivalently, go to <https://code.metoffice.gov.uk/rosie/u/>, put an idx code in the search box, find the right suite and then click “../” once to go up to ‘trunk’.

- Please DO NOT be tempted to rename the suite on your local system (e.g. renaming the directory in `~/roses/` to `rivers_suite_!`: this will confuse the FCM system for sharing your suite with the repository. If you need to keep track of them, make yourself a textfile to remind you what all the IDX codes mean (mine is called `~/roses/what_are_these_suites.txt`).



| local | branch | idx | owner | revision | title |
|-------|--------|---------|--------------|----------|------------------------------------------------------------------------|
| | trunk | u-ax550 | joaoteixeira | 77114 | JULES standalone GL8 configuration running with WFDEI + INFERNO in HPC |
| | trunk | u-av250 | ruthlewis | 77648 | GL8 JULES FLUXNET2015 Vegetative drag study control |
| | trunk | u-ax683 | ruthlewis | 77688 | GL8 JULES FLUXNET2015 Vegetative drag 9 tile control |
| | trunk | u-ax685 | ruthlewis | 77689 | GL8 JULES FLUXNET2015 Vegetative drag 9 tile new z0 |
| | trunk | u-ax684 | ruthlewis | 77690 | GL8 JULES FLUXNET2015 Vegetative drag 1 tile new z0 |
| | trunk | u-ax735 | ruthlewis | 77840 | GL8-new snow JULES FLUXNET2015 Vegetative drag study control |
| | trunk | u-ax937 | heatherasl | 78594 | JULES standalone GL8 configuration for use with Cardington data |
| | trunk | u-aw289 | heatherasl | 78596 | JULES standalone GL8 configuration for use with single point data |

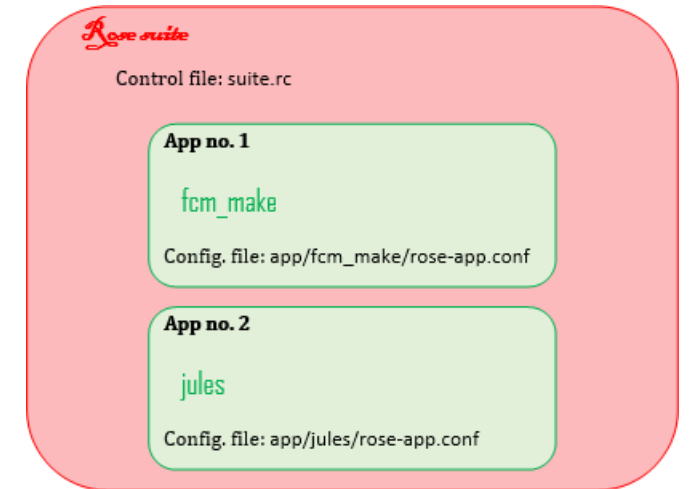
The *Rosie Go* window. *Rosie Go* expects a directory `~/roses/` to be on your system, which it uses as a store for downloaded Rose suites.

JULES From Scratch: Rose suites



It's important to remember that Rose suites are NOT version-independent and are also NOT platform-independent. For example, if you have downloaded a suite suitable for JULESv6.1 that ran on a Univ. Durham's computer, you will need to modify it to have it run for JULESv6.2 on a Univ. Exeter's computer (or JASMIN, or ARCHER, etc.) even if all the science settings are identical.

Apart from very tiny suites (e.g. my examples in this tutorial), suites also do NOT contain the driving and ancillary files they need to run (they just have pointers to their locations), so if you want to use that suite from Durham you will also generally have to download these files separately (usually involving emailing the suite authors and asking for copies and permission to use them).



Details of the modification steps are on this page:

<https://jules.jchmr.org/modify-suite>

For suite.rc syntax, see: <https://cylc.github.io/cylc-doc/7.8.8/html/appendices/suiterc-config-ref.html>



We're about halfway through *JULES From Scratch* now, so let's recap:

- We started with **installation**, starting at the page <https://jules.jchmr.org/scratch> , which got you set up with *Cylc*, *Rose* and *FCM* installed and also JULES itself (each has their own page describing the installation steps).
- **Rose suites** came next <https://jules.jchmr.org/rose-suites> , what they are and where to get them. These are the control files for a JULES run and details of the essential modification steps you need to go through to get a downloaded suite working on your system are here: <https://jules.jchmr.org/modify-suite>

See my 'getting started' page <https://jules.jchmr.org/getting-started> for the other files you need like driving data, ancillary files (and links to analysis tools are there too because you will need to plot out the results of your runs too).

Let's go through a **worked example** ...

JULES From Scratch: Worked example of a point run



JULES
Joint UK Land
Environment Simulator



Here's an example **case**: a point run using the 1997 Loobos example that used to be included in the JULES model download (Loobos is the well-known eddy covariance tower in the Netherlands; see 'Example configurations' on <https://jules.jchmr.org/configurations>).

I have prepared a Rose suite for you that is on the *Rosie Go* database:

- `rosie go`

Download suite `u-cz674` (see above) and modify the paths inside it to point to your own profile rather than mine (i.e. list any occurrence of "tmarthews" using `grep`, open the relevant files in a text editor and replace "tmarthews" with something different).

- `export RSUITE=$HOME/roses/u-cz674`
- `grep -ir "tmarthews" $RSUITE/*`
- #Edit the files that come up appropriately.

• I have a number of suites in my `~/roses/` directory on JASMIN (the "cylc1" shows I am on the *JASMIN Cylc server* *):

```
[tmarthews@cylc1 roses]$ pwd
/home/users/tmarthews/roses
[tmarthews@cylc1 roses]$ ls
nlists u-cz674 u-cf638 u-cl832 u-cm340 u-cm790 u-cm795 u-cm913 u-cp589 u-cq784 u-cr234 u-cr235
nlists u-cz674 u-cf682 u-cm022 u-cm641 u-cm791 u-cm796 u-cn118 u-cp592 u-cq785
nlists u-cz674 u-ae544 u-cg242 u-cm131 u-cm787 u-cm792 u-cm797 u-cn119 u-cp678 u-cq915 what_are_these_suites.txt
nlists u-cz674 u-bq564 u-cg421 u-cm322 u-cm788 u-cm793 u-cm798 u-co126 u-cp679 u-cr079
nlists u-cz674 u-bu747 u-ck387 u-cm323 u-cm789 u-cm794 u-cm809 u-cp588 u-cq783 u-cr083
[tmarthews@cylc1 roses]$
```



See <http://www.climateexchange.nl/sites/loobos/index.htm>



* JASMIN recommends that all *JULES* users log into the Cylc server cylc.jasmin.ac.uk (see <https://help.jasmin.ac.uk/article/147-cylc-rose-on-jasmin>) for their work.

JULES From Scratch: Worked example of a point run



This suite uses [JASMIN](#), a UK superdata cluster, and a particular branch of JULES that is close to JULESvn7.3 (I modified vn7.3 to have a few extra checks on the surface land cover fractions). Extract the version of JULES this suite needs to use like this:

- `#Use some UNIX string operations to set $JULES_ROOT to be the JULES source set in $RSUITE (i.e. equal to $JULES_SOURCE as used inside the suite)`
- `export tmp1=`grep -ir "JULES_SOURCE" $RSUITE/app/fcm_make/rose-app.conf``
- `export JULES_ROOT=`echo ${tmp1##*=}``
- `unset tmp1`
- `echo $JULES_ROOT`



For bash built-in string operations, see <https://linuxhandbook.com/bash-strings/>. The “##” means “find the last occurrence of the following character” (one # would be the first occurrence) and the {} means “evaluate”.

You should find that \$JULES_ROOT is set to `./vn7.3_elevq`. If you don't already have it, you can find this branch at the URL https://code.metoffice.gov.uk/trac/jules/browser/main/branches/dev/tobymarthews/vn7.3_elevq.

- If we were going to download JULESvn7.3 we would do this (see STEP 2 of <https://jules.jchmr.org/get-jules>):
 - `# cd ~/MODELS`
 - `# fcm co fcm:jules.x_tr@vn7.0 jules-vn7.3`
- However, for this example we need instead to download a slightly modified branch of the model (pls note the URL is not the same as the one above):
 - `cd ~/MODELS`
 - `fcm co https://code.metoffice.gov.uk/svn/jules/main/branches/dev/tobymarthews/vn7.3_elevq`
 - `#Equivalently, with FCM keywords set up then you can do the slightly-shorter: fcm co fcm:jules.x_br/dev/tobymarthews/vn7.3_elevq`

JASMIN

JULES From Scratch: Worked example of a point run

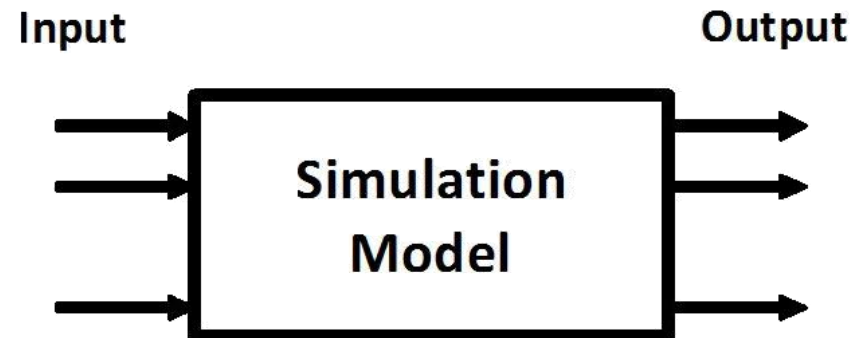


Now it's a good idea to check where the output of the model run is going:

- `#Extract the output directory specified in $RSUITE into environment variable $OUTPUT_DIR`
- `#` (for u-cz674 this location is actually a subdirectory of \$RSUITE, which is a bit unusual: usually \$OUTPUT_DIR will be a separate location, e.g. on your scratch space)
- `export tmp1=`grep -ir "output_dir" $RSUITE/app/jules/rose-app.conf``
- `export OUTPUT_DIR=`echo ${tmp1##*=} | sed "s///g"``
- `unset tmp1`

- `#Check that the output directory exists and that we have permission to write to it (you might also want to delete its contents using rm $OUTPUT_DIR/*):`
- `echo $OUTPUT_DIR`
- `chmod -R 755 $OUTPUT_DIR`

- `#Check all your environment variables are correct:`
- `echo $RSUITE`
- `echo $JULES_ROOT`
- `echo $OUTPUT_DIR`



JASMIN

JULES From Scratch: Worked example of a point run



...and more:

- #Optionally, I recommend to run these quick checks on your suite too:
- `export tmp1=`grep -ir "meta=" $RSUITE/app/fcm_make/rose-app.conf``
- `export JMETAI=`echo $tmp1 | awk '{split($0,a,"/rose-meta");print a[1]}' | sed "s/meta=//g"``
- `unset tmp1`
- `export tmp1=`grep -ir "meta=" $RSUITE/app/jules/rose-app.conf``
- `export JMETAZ=`echo $tmp1 | awk '{split($0,a,"/rose-meta");print a[1]}' | sed "s/meta=//g"``
- `unset tmp1`
- `echo -n 'Metadata path in "$RSUITE/app/fcm_make/rose-app.conf same as "$JULES_ROOT? '; if ["$JMETAI" == "$JULES_ROOT"]; then echo 'Yes'; else echo 'No (this may not be incorrect, but please check it)'; fi`
- `echo -n 'Metadata path in "$RSUITE/app/jules/rose-app.conf same as "$JULES_ROOT? '; if ["$JMETAZ" == "$JULES_ROOT"]; then echo 'Yes'; else echo 'No (this may not be incorrect, but please check it)'; fi`

A point about environment variables: I try to make it so that my suites use as few environment variables as possible because they cause problems if the profile you are calling the suite from differs from the profile where the suite is running.

Some suites require you to define a number of environment variables in order to be able to run them outside Rose. Most of these should be listed by `cat $RSUITE/rose-suite.conf` (for this example, you'll see I only have the three obligatory ones defined there). Note: the variables defined above like `$RSUITE` are deliberately not used anywhere inside the suite, but just make it easier to present what comes next in my slides here.

JULES From Scratch: Worked example of a point run



Next, you need to choose your Fortran COMPILER. Options described here are (1) **gcc/Gfortran** (in the JasPy module on Jasmin) or (2) **intel/ifort** and you need to tell JULES which one you are using. I inserted platform files for Jasmin into JULES in 2021 with this information (`$JULES_ROOT/etc/fcm-make/platform/jasmin*.cfg`):



- #FOR OPTION (1), Gfortran:
 - `sed -i 's/custom/jasmin-gcc-nompi/g' $JULES_ROOT/etc/fcm-make/make.cfg`
 - `sed -i 's/jasmin-intel-nompi/jasmin-gcc-nompi/g' $JULES_ROOT/etc/fcm-make/make.cfg`
 - `sed -i 's/custom/jasmin-gcc-nompi/g' $RSUITE/app/fcm_make/rose-app.conf`
 - `sed -i 's/jasmin-intel-nompi/jasmin-gcc-nompi/g' $RSUITE/app/fcm_make/rose-app.conf`
- #If you have Gfortran version \geq vn10.0 (check using `gfortran --version` after the appropriate module load command), do this:
 - `sed -i 's/= gfortran$/= gfortran_10_plus/g' $JULES_ROOT/etc/fcm-make/platform/jasmin-gcc-nompi.cfg`
- #If you have Gfortran version \geq vn4.9 and \leq vn11.2 (check using `gfortran --version`), do this*:
 - `sed -i 's/= -std=f2003 -/= -/g' $JULES_ROOT/etc/fcm-make/compiler/gfortran.cfg`
 - `sed -i 's/= -std=f2003 -/= -/g' $JULES_ROOT/etc/fcm-make/compiler/gfortran_10_plus.cfg`
- #Check these changes have gone through using: `nedit $RSUITE/suite.rc $JULES_ROOT/etc/fcm-make/make.cfg $RSUITE/app/fcm_make/rose-app.conf $JULES_ROOT/etc/fcm-make/platform/jasmin-gcc-nompi.cfg $JULES_ROOT/etc/fcm-make/compiler/gfortran.cfg $JULES_ROOT/etc/fcm-make/compiler/gfortran_10_plus.cfg &`



- #FOR OPTION (2), ifort:
 - #n.b. *** to use this option on Jasmin you need access to the jules GWS ***
 - `sed -i 's/custom/jasmin-intel-nompi/g' $JULES_ROOT/etc/fcm-make/make.cfg`
 - `sed -i 's/jasmin-gcc-nompi/jasmin-intel-nompi/g' $JULES_ROOT/etc/fcm-make/make.cfg`
 - `sed -i 's/custom/jasmin-intel-nompi/g' $RSUITE/app/fcm_make/rose-app.conf`
 - `sed -i 's/jasmin-gcc-nompi/jasmin-intel-nompi/g' $RSUITE/app/fcm_make/rose-app.conf`
- #My suites are set up for Gfortran by default, so you need to change the `suite.rc` file slightly here. Open it in the `nedit` text editor and change the `module load` commands according to the comments I have left in the `suite.rc` file:
 - `nedit $RSUITE/suite.rc &`
- #If using JULES version \leq 7.2 please do this too (see <https://code.metoffice.gov.uk/trac/jules/ticket/1351>):
 - `sed -i 's/= intel$/= intel_15_plus/g' $JULES_ROOT/etc/fcm-make/platform/jasmin-intel-nompi.cfg`
- #Check these changes have gone through using: `nedit $RSUITE/suite.rc $JULES_ROOT/etc/fcm-make/make.cfg $RSUITE/app/fcm_make/rose-app.conf $JULES_ROOT/etc/fcm-make/platform/jasmin-intel-nompi.cfg &`

* The “std=f2003” compiler flag triggers an error message (see <https://code.metoffice.gov.uk/trac/jules/wiki/ticket/711/TicketDetails>).

JULES From Scratch: Worked example of a point run



Next, be aware that there are TWO WAYS to run any particular JULES Suite (use one of **left** or **right** below):

- ```
>_
```
- **#OUTSIDE ROSE** (i.e. from the command line: this is good to try first because it will work even if you don't have Rose set up):
  - #If using Gfortran, do this (see advice I put at the top of `$JULES_ROOT/etc/fcm-make/platform/jasmin-gcc-nompi.cfg`):
  - `module load jasper`
  - #If using ifort, do this (see advice I put at the top of `$JULES_ROOT/etc/fcm-make/platform/jasmin-intel-nompi.cfg`):
  - `module load intel/19.0.0`
  - `module load eb/OpenMPI/intel/3.1.1`
  - #Now compile\* the JULES model (as recommended on <http://jules-lsm.github.io/latest/building-and-running/fcm.html>; the "-j 2" is optional so if this command fails, just try it without that option):
  - #n.b. \*\*\* for the VM at UKCEH, no need for any of the module load commands above or anything on the last slide \*\*\*
  - `cd $JULES_ROOT`
  - `fcm make -j 2 -f etc/fcm-make/make.cfg --new`
  - `echo -e "\a"`
  - #After a few minutes of [info] lines you should see "[done] make" which indicates it's finished compiling (or you can do the echo command which sounds a 'beep' when it is done \*\*). This is, by the way, just about enough time to listen to the ShowHawk Duo playing an acoustic version of *Miserlou* [https://www.youtube.com/watch?v=zVUjGyBu\\_WY](https://www.youtube.com/watch?v=zVUjGyBu_WY). 😊
  - #Now extract the namelists from the Rose suite and put them in `$NAMELIST` (I use a subdirectory of `~/roses/` for this):
  - `export NAMELIST=$HOME/roses/nlists_${RSUITE##*/}; mkdir -p $NAMELIST; cd $NAMELIST`
  - `rose app-run -i -C $RSUITE/app/jules; cd ~`
  - #Finally, initiate the run (I prefer running from `~/` but you can run from anywhere):
  - `$JULES_ROOT/build/bin/jules.exe $NAMELIST`
  - #You can also define an alias to save typing (`alias jrun="$JULES_ROOT/build/bin/jules.exe $NAMELIST"`)
  - #and then do e.g. `"jrun >job.out"` to divert screen output to a file called `job.out`



- **#USING ROSE** (for the Rose system, see <https://code.metoffice.gov.uk/trac/jules/wiki/JULESWithRose>; this is better because you can log out and leave it running e.g. overnight):
- #Now open the suite:
- `rose edit -C $RSUITE &`
- Ignore any red warning triangles ⚠ and click the Play button in Rose Edit (triangle, top-right: ▶ ).
- #n.b. outside Rose the executable appears at `$JULES_ROOT/build/bin/jules.exe` (see left), while here the executable appears instead at `$HOME/cylc-run/${RSUITE##*/}/share/fcm_make/build/bin/jules.exe` (and the namelists are extracted to `$HOME/cylc-run/${RSUITE##*/}/work/1/jules/`).

\* The JULES model consists of approximately 600 source files (see `tree $JULES_ROOT/src | tail -n 1`) totalling around 150,000 lines of Fortran code (see `wc -l `find $JULES_ROOT/src/ -type f -iname "*.f90"``).

\*\* By default, MobaXTerm does not sound beeps: if you want that, you need to update the settings to enable this.

# JULES From Scratch: Worked example of a point run



A stream of [INFO] lines like this (right) indicates that JULES is running (if you are running outside Rose they will come up straight away; if running through Rose you can find them by right-clicking on the 'jules running' line in the Cylc GUI → View → [job.out or job stdout]):

```
[INFO] next_time: Timestep: 743; Started at: 1997-01-16 10:00:00
[INFO] next_time: Timestep: 744; Started at: 1997-01-16 10:30:00
[INFO] next_time: Timestep: 745; Started at: 1997-01-16 11:00:00
[INFO] next_time: Timestep: 746; Started at: 1997-01-16 11:30:00
[INFO] next_time: Timestep: 747; Started at: 1997-01-16 12:00:00
[INFO] next_time: Timestep: 748; Started at: 1997-01-16 12:30:00
[INFO] next_time: Timestep: 749; Started at: 1997-01-16 13:00:00
[INFO] next_time: Timestep: 750; Started at: 1997-01-16 13:30:00
[INFO] next_time: Timestep: 751; Started at: 1997-01-16 14:00:00
[INFO] next_time: Timestep: 752; Started at: 1997-01-16 14:30:00
[INFO] next_time: Timestep: 753; Started at: 1997-01-16 15:00:00
[INFO] next_time: Timestep: 754; Started at: 1997-01-16 15:30:00
[INFO] next_time: Timestep: 755; Started at: 1997-01-16 16:00:00
```

- #When you see "[INFO] jules: Run completed successfully", the output files will be here:
- `cd $OUTPUT_DIR`
- `ls -alh`
- #...and you can delete this directory if you wish:
- `rm -r $NAMELIST`



If that all works, then you have just run the JULES model (!).

- Next, try a modified run. Change the duration of the run (see "you can edit a Rose suite in TWO different ways" above and find the *Timestepping* tab) so that it is just the first 15 days of June 1997 (midnight is always at the start of the day, so this means *main\_run\_end* should be 1997-06-16 00:00:00). Does it re-run?
- **PLEASE NOTE that** this Loobos suite is an **EXAMPLE**, not a **RECOMMENDED set of parameters** for a JULES point run (see **Example configurations** <https://jules.jchmr.org/configurations> ). If you use this setup to undertake a JULES run at a different site, you **MUST change the parameters inside**. For example, the soil parameter and vegetation values in `u-cz674` are appropriate for the Loobos site, but will almost certainly NOT be appropriate at a different site.

## JULES From Scratch: Worked example of a point run



For your convenience (i.e. saving you copying out the commands from multiple slides above), here's a summary of the commands you need (if you are using Rose, stop just before the module load commands, go to Rose Edit and press Play):

```
#Select the suite for your run (= "case") and check all settings are right in Rose Edit:
export RSUITE=$HOME/roses/u-cz674
rose edit -C $RSUITE &
```

```
#Identify the JULES code used by this suite and its output folder:
export tmp1=`grep -ir "JULES_SOURCE" $RSUITE/app/fcm_make/rose-app.conf`
export JULES_ROOT=`echo ${tmp1##*=}`
unset tmp1
export tmp1=`grep -ir "output_dir" $RSUITE/app/jules/rose-app.conf`
export OUTPUT_DIR=`echo ${tmp1##*=}| sed "s//g"`
unset tmp1
chmod -R 755 $OUTPUT_DIR
echo $RSUITE;echo $JULES_ROOT;echo $OUTPUT_DIR
```

```
#Check the compiler you are using at this point with:
nedit $RSUITE/suite.rc $JULES_ROOT/etc/fcm-make/make.cfg &
```

```
#If it's gfortran, do:
module load jaspys
```

```
#If it's intel, do:
module load intel/19.0.0;module load eb/OpenMPI/intel/3.1.1
```

```
#Compile (if outside Rose)
cd $JULES_ROOT
fcm make -j 2 -f etc/fcm-make/make.cfg --new
```

```
#Generate namelists (if outside Rose)
export NAMELIST=$HOME/roses/nlists_${RSUITE##*/}; mkdir -p $NAMELIST; cd $NAMELIST
rose app-run -i -C $RSUITE/app/jules; cd ~
#Run (if outside Rose)
nohup nice $JULES_ROOT/build/bin/jules.exe $NAMELIST >~/out_${RSUITE##*/}.txt &
#(or just $JULES_ROOT/build/bin/jules.exe $NAMELIST if you want output to screen)
```

```
#After the run:
rm -r $NAMELIST
cd $OUTPUT_DIR
ls -alh
```

All done!

# JULES From Scratch: Worked example of a point run



| task       | state   | host      | job system | job ID | T-submit  | T-start   | T-finish | dT-mean | latest me  |
|------------|---------|-----------|------------|--------|-----------|-----------|----------|---------|------------|
| 1          | running |           |            |        |           |           |          |         |            |
| ■ fcm_make | running | localhost | background | 13198  | 11:52:22Z | 11:52:23Z | *        | *       | job(01) st |
| □ jules    | waiting | *         | *          | *      | *         | *         | *        | *       | *          |

running to stop at 1 ■ □ (filtered: □) live 2022-10-10T12:52:35+01:00

## JULES From Scratch: Worked example of a gridded run



**JULES**  
Joint UK Land  
Environment Simulator



Here's a grid run using *earthH2Observe* (E2O) data that you should be able to follow if you can download the ancillary and driving data files from <https://www.tobymarthews.com/jules-short-course-2016.html> :



- Download the ancil file, unzip it and save it in a directory called **ancils/** on your system.
- Download the driving data files (all the other .zip files), unzip and put them all in a directory called **2013/** (i.e. you should end up with several files each in their own directory like `/.../2013/Tair/Tair_EI_025_noheight_201301.nc`). Note that you only have 16 days of data from Jan 2013, but this will do fine for this example (if you want to use these data for a longer run, the complete files are on [the LSPCEH workspace on JASMIN](#) but you will need to ask permission to use those files). Don't forget to set up the soft links described on <https://www.tobymarthews.com/jules-short-course-2016.html> after you have downloaded the driving data files.
- Take a copy of my suite `u-cz675` from Rosie Go and modify the paths inside it to point to equivalent locations on your system (see above).



## JULES From Scratch: Worked example of a gridded run



**JULES**  
Joint UK Land  
Environment Simulator



- Next, follow the instructions for the point run above, with the difference that your suite number will be u-cz675 not u-cz674 so you need to start with:
  - `export RSUITE=$HOME/roses/u-cz675`
- The JULES version used by u-cz674 and u-cz675 is the same, and I have made no changes to the suite.rc (the files u-cz674/suite.rc and u-cz675/suite.rc are actually identical because they are two copies of my generalised 'suite.rc for JASMIN'). You can just change the \$RSUITE at the top of 'Worked example of a point run' above and follow *all subsequent steps exactly as for the point example above*. For me, it takes 18 min to run.
- You should find yourself running JULES for a grid containing 111536 soil points.
- Success?
- Next, try a modified run. Try turning OFF the spin-up and rerunning (it should now take only 9 min to run). Success?
- **PLEASE NOTE** that if you want to use *earthH2Observe* data 'for real' then I need to give another **health warning**: **My u-cz675 suite is NOT one of the recommended configurations of JULES** (although it's not too different from **WRR2** on <https://jules.jchmr.org/configurations> ). Many other global configurations are available, so do not just copy my parameter settings - they may very well NOT be appropriate for your work. Also, of course, any use of the *earthH2Observe* data outside my example here will require **requesting permission** (see <http://www.earth2observe.eu/> ).



While you have that example running:

- Check that you can right-click on jules → View → job stdout to check on progress (this file holds the [ INFO ] lines JULES produces during execution).
- If you close the Cylc GUI accidentally, type `cylc gscan &` and double-click on your Rose suite job to reopen it. In the same way, you can reopen a suite you've left running e.g. overnight.
- A new directory `~/cylc-run/` will appear on your system when the run starts (if it is not there already). When you run a rose suite, a copy of the whole suite is put in this directory at location `$CSUITE` defined like this (`$CSUITE` is what I call the suite's 'run directory' described on <https://metomi.github.io/rose/doc/html/tutorial/rose/suites.html#suite-directory-vs-run-directory>):

```
export CSUITE=$HOME/cylc-run/${RSUITE##*/}
echo $CSUITE
```

- Note that Rose suites run independently of the session you're in (Cylc suites run as daemons) so you gain nothing by opening three separate shells and initiating three runs in each rather than running three in the same session.
- If you find that the `fcmake` step works, but it hangs when it moves on to the `jules` app, check that you have correctly added the JULES command set to `$PATH` (see <https://jules.jchmr.org/get-jules> ).

# JULES From Scratch: Troubleshooting



If you get an error like this:

```
ModuleNotFoundError: No module named 'Queue'
```

...then I am afraid you are using a version of Rose that was put together before late 2018 (-ish) (most likely Rose version 53 if you were following the July 2021 version of my JULES From Scratch tutorial).

You will need to go back, delete the entire contents of ~/.local on JASMIN and install a later version of Cylc, Rose and FCM (I have now updated all my installation pages on <https://jules.jchmr.org/scratch> to use Rose version >53).

If you get an error like this:


```
Bad name-space
```

...then there could be nothing wrong with your suite: I have found that if I abort a run too quickly/often then sometimes the system gets confused and produces this error. Logging out and logging back in to Jasmin usually clears it, but sometimes not. If not, you may need to copy the suite to a new idx number and run that instead (discarding the old idx number).

# JULES From Scratch: Troubleshooting



If you want to abort/stop a suite:

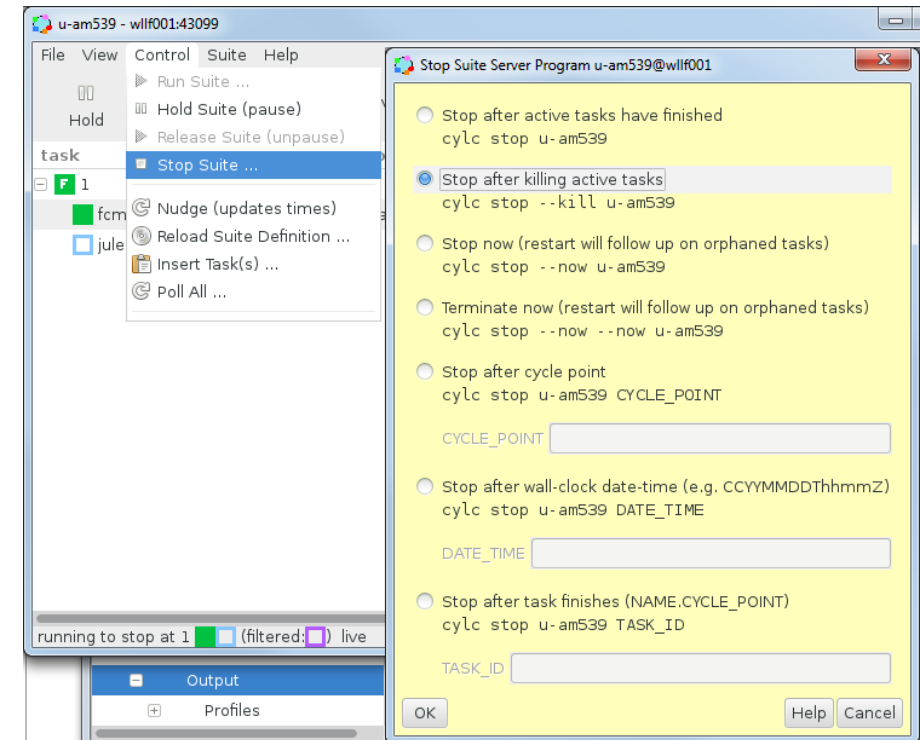
DON'T JUST CLOSE OFF THE CYLC GUI WINDOW (the one with the  ): the suite will still continue in the background and will prevent you starting another run of the same suite.

DON'T CLICK THE BIG STOP SUITE BUTTON: Surprisingly, this *doesn't* stop the suite \*\*: Cylc will actually wait for all submitted / running apps to complete (during which it says "stopping"), and only when they've completed will it stop the suite. I feel it really should be called a 'Complete submitted apps & exit' button (but that's just my opinion)

If you want to actually kill/stop your suite completely, you need to go into the menus like I'm showing in the screenshot right and choose 'Stop after killing active tasks', which will do the equivalent of the command:

```
cylc stop ${RSUITE##*/} --kill
```

It's never happened to me, but if that still doesn't work try NCAS's advice at <http://cms.ncas.ac.uk/wiki/RoseCylc/Hints#Problemsshuttingdownsuites> .



\*\* In *Cylc* terminology, the suite is just the calling structure for a set of apps, so this button does 'stop the suite' in the sense that no further apps will be initiated, but it doesn't stop any apps that have already been called (see <http://metomi.github.io/rose/doc/html/cheat-sheet.html>).

## JULES From Scratch: Viewing the log files



Rather than printing progress information to the screen (e.g. as [ INFO ] lines) and any execution errors, when JULES is run through Rose/Cylc these two are diverted into two textfiles called 'log files' stored in the directory `~/cylc-run/`. Progress information goes into a file called **job.out** and any errors go to a file called **job.err**, which you can open from the command line like this:

```
export CSUITE=$HOME/cylc-run/${RSUITE##*/}
more $CSUITE/log/job/1/jules/NN/job.out
more $CSUITE/log/job/1/jules/NN/job.err
```

Or, as mentioned above, you can also open these two log files through the Cylc GUI by right-clicking on the 'jules running' line → View → [job stdout or job stderr]

If your job succeeds, note that the Cylc GUI will always clear the job away and you won't be able to open these files with the mouse any more, however, which I find a bit annoying (the files are still accessible from the command line, though).

# JULES From Scratch: Viewing the log files



A more user-friendly way of viewing these logs that works on most systems is to use **Rose Bush** (n.b. not installed on JASMIN):

```
rose slv --name=${RSUITE##*/}
```

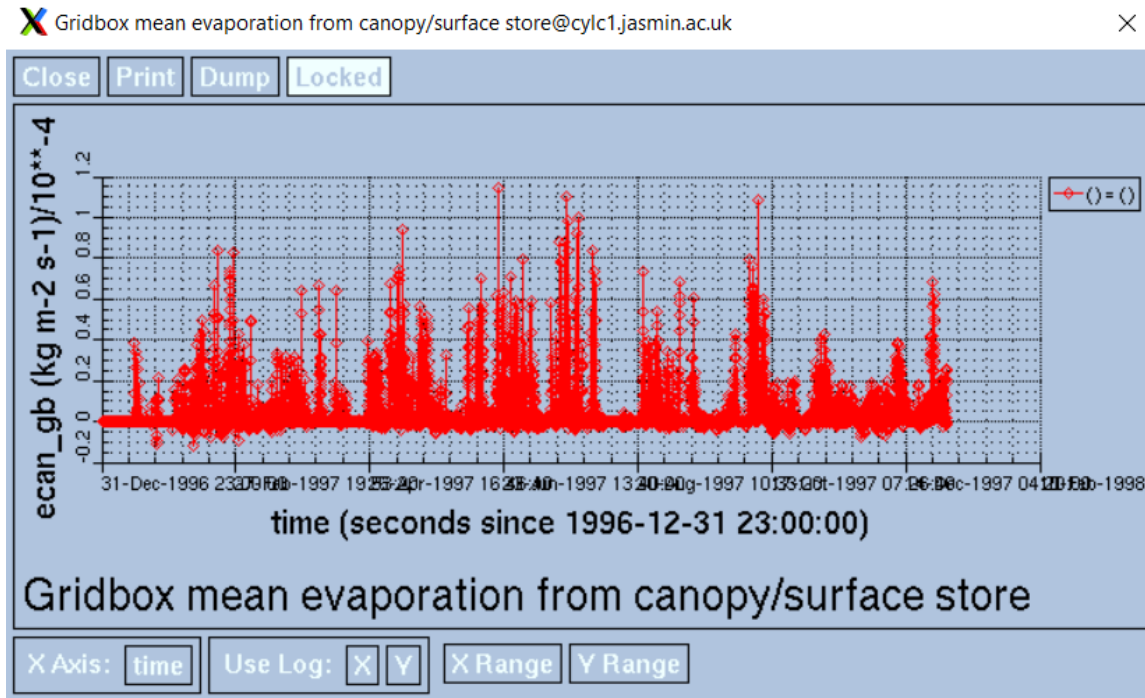
- On JASMIN note that Rose Bush is not installed.
- On some systems this command can take 10-15 sec to open: on Monsoon, a workaround is to launch firefox on xcslcX and go to <http://localhost/rose-bush/> (see <http://cms.ncas.ac.uk/wiki/RoseCylc/Hints#CantviewoutputinRosebush>)
- Rose Bush also has a 'history' feature and will show you the logs of previous runs. This is useful, but note that if you ever delete the ~/cylc-run/ directory from your system then you will wipe this history too (I sometimes clear ~/cylc-run/ because it's really just a scratch directory and it sometimes gets very large).

# JULES From Scratch: Analysis

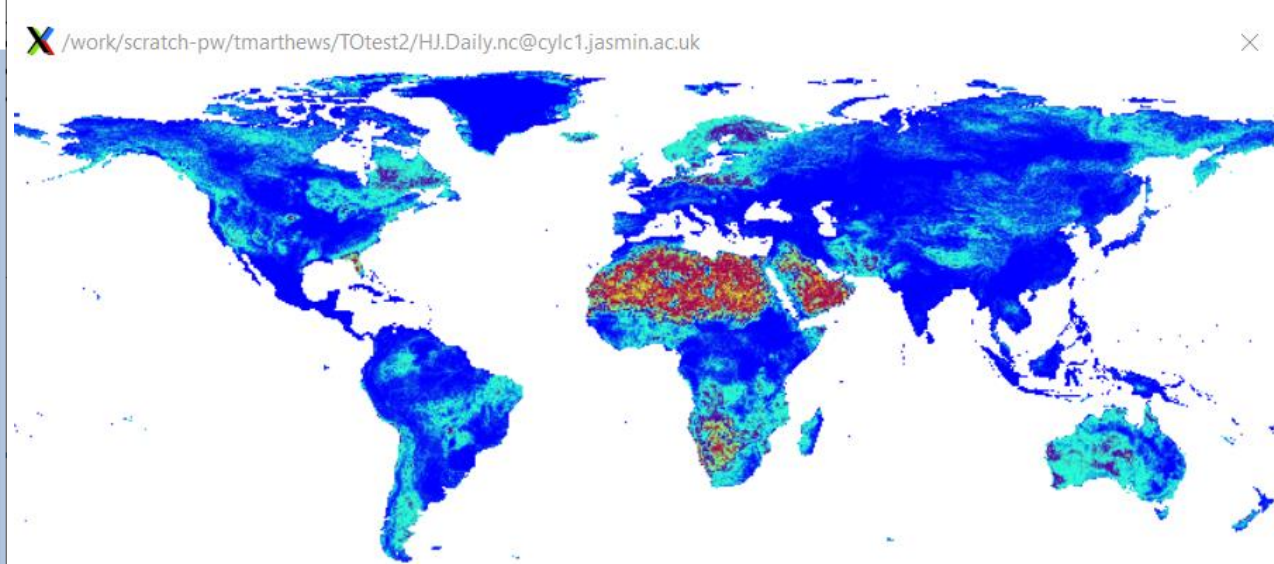


When your run(s) have finished, you will want to look at the outputs with commands like this:

```
ncview -extra -noautoflip -no_auto_overlay -minmax all $OUTPUT_DIR/JFScratch.Base.nc &
ncview -extra -noautoflip -no_auto_overlay -minmax all $OUTPUT_DIR/JFScratch.Daily.nc &
```



Output from `u-cz674`



Output `fwetl` from `u-cz675`



- For some basic links about the analysis stage, please see <https://jules.jchmr.org/getting-started#analysis-tools>
- One option I encourage you to look into is to upload your JULES outputs to the JASMIN Cloud and produce some plots using Datalabs <https://datalab.datalabs.ceh.ac.uk/> .

## Analysis tools for JULES output data

Gridded runs from Land Surface Models (LSMs) in general can output data in 2D or 1D NetCDF files and you usually need to use different software to visualise/analyse these two types of gridded output. However, please be aware that **JULES will (almost) always give you 1D output from gridded runs, whether the driving data are 1D or 2D** (only in a few specific cases will the output be 2D: details are in the [documentation section for model\\_grid.nml](#)). This means that you either need to use analysis software that can deal with 1D files or you need to anticipate a *post-processing* step to convert your 1D JULES outputs into 2D files.

- For converting 1D JULES output to 2D, see [here](#).
- For JASMIN users, be aware of the [JASMIN Analysis Platform](#).
- Finally, see [here](#) for evaluation / benchmarking.

- For ensemble runs, I also recommend to have a look at iLAMB: see <https://www.ilamb.org/doc/tutorial.html> .



## JULES From Scratch: Help resources



I suggest to bookmark the following webpages if you are ever dealing with JULES:

JULES website: <https://jules.jchmr.org/>

JULES TRAC: <https://code.metoffice.gov.uk/trac/jules/>

JULES online manual: <http://jules-lsm.github.io/latest/namelists/contents.html>

Example Rose suite: <https://code.metoffice.gov.uk/trac/roses-u/browser/b/p/6/9/6>

JULES Tickets: <https://code.metoffice.gov.uk/trac/jules/query>

JULES Code Branches: <https://code.metoffice.gov.uk/trac/jules/browser/main/branches/dev>

Much more information about Rose: <https://code.metoffice.gov.uk/doc/um/latest/um-training/index.html>

Remember: **YOU ARE NOT ALONE** in your labours and trials with JULES, Rose, Cylc and FCM: Please do use the support email lists/groups. Alternatively, if you don't feel these are fit for your purposes (too high-brow, too technical, unfriendly, simply scary ...) then go ahead and create your own (!) and I'll add it to the list on <https://jules.jchmr.org/getting-started#email-and-support-lists> .

# JULES From Scratch



- In this short course we have only looked at *using* JULES, but of course there is a huge amount of ongoing work about how to *develop and improve* this model, not least because JULES is the UK's national land surface model and forms a large part of our contribution to the global climate change debate and international collaborations such as CMIP and the IPCC reports.
- **If you are attending the JULES meeting starting tomorrow then you will hear very much more about this (including my talk tomorrow at 11:45).**



Website: <https://jules.ichmr.org/news-and-media/news/annual-jules-meeting-13-15th-september-2023>

For travel and accommodation advice for Exeter see bottom of document.

Contact: [arthur.argles@metoffice.gov.uk](mailto:arthur.argles@metoffice.gov.uk)

This year we are hosting the JULES Annual Science Meeting down at the University of Exeter between the 14th-15th of September in the Newman building. There is a training day with JULES at the Innovation Centre, University of Exeter, before the start of the official Science Meeting on the afternoon of Wednesday 13th of September.

Main meeting MS Teams link to join the on (days 1-2):

|                                                |
|------------------------------------------------|
| JULES Annual Science Meeting - Main            |
| <a href="#">Click here to join the meeting</a> |
| Meeting ID: 354 995 118 391                    |
| Passcode: pwxwpr                               |

Breakout group MS Teams (16:00-17:00, day 1) links are as follows:

If you wish to find out more about JULES and how it is used (and a lot of the work being done at UKCEH), please subscribe to [@CEHScienceNews](#) and maybe come and visit us?



# **JULES**

**Joint UK Land  
Environment Simulator**

**Thank you very  
much!**