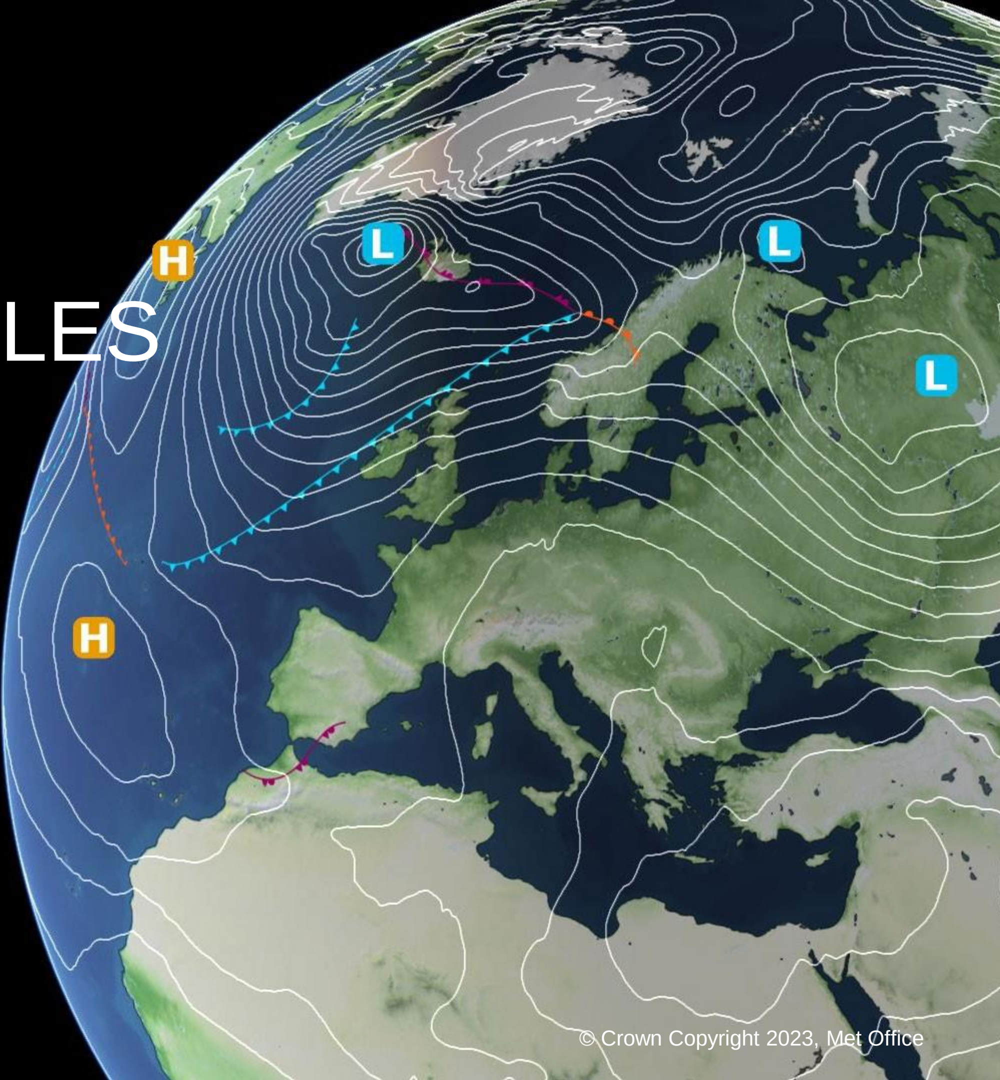
Met Office

Technical Future of JULES

Richard Gilham
Jenny Hickson
Giorgia Line
Lianne Parkhouse

Simulation Systems & Deployment (SSD) team
aka UM System Team



A team event…

- Top level overview- Rich
- Working Practices & LFRic Apps- Jenny
- Fab & GitHub- Giorgia
- JULES docs GitHub Migration- Lianne
- Conclusions- Rich
- Q&A

Top Level Overview

Rich

SSD (aka UM Sys) Team

Curators of JULES, UM etc

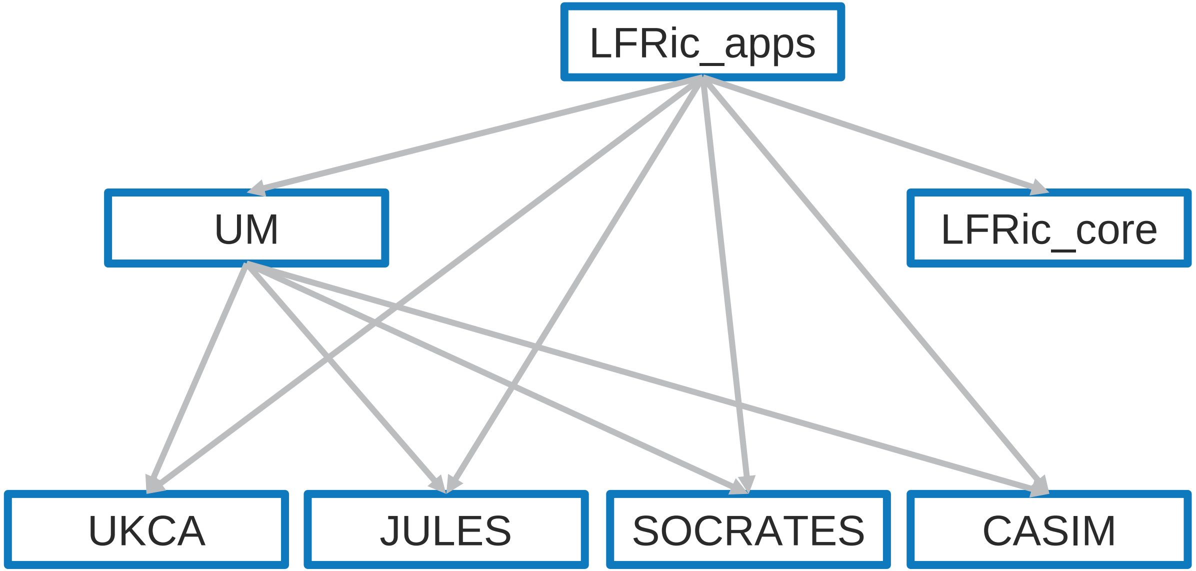
- Coordinating releases
- Code reviews
- Working Practices
- User/Developer experience
- Technical support

Development (planned/reactive)

- Tech debt payback
- New capability development
- Supercomputer porting

One big ecosystem; many different needs

- Millions of lines of code
  - JULES about 10%*
- 100s users & developers
- Atmosphere models numerically sensitive
- Important safeguards for one group a barrier to others
- Navigate a tricky network of compromises



Really difficult to please everyone!

*Very approx. measured in lines of code or tickets committed

Why this talk?

- Tech makes the science possible
  - Investment, not a Cost
- Should ‘just work’
- Kirk never called Scott to say everything’s fine
- As software, JULES either evolves or dies

The Engine Room wants to talk about some upgrades…

Next Generation Modelling System (NGMS)

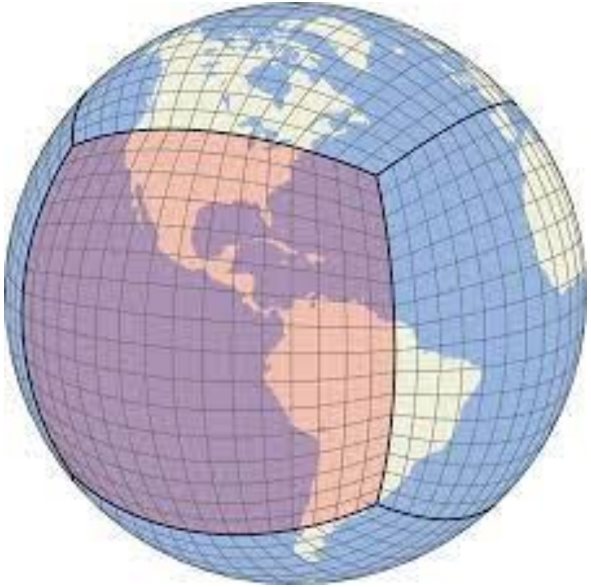
- Produce a framework of models fit for future HPCs
  - UM is running out of road
- Includes an array of LFRic Applications
  - Common core technical infrastructure, configuration etc
- LFRic_atm will succeed UM in the late 2020s
  - Linear & adjoint for Data Assimilation
  - Dynamics-only, Gravity Wave…
- Why not LFRic_JULES?
  - exaJULES project! (Emma)

Momentum

The Unified Earth Environment
Prediction Framework

LFRic_atm- things to know

- Separation of concerns- tech and science code better partitioned.
  - PSyClone to write parallel code for CPU, GPU etc
  - Requires new build system- Fab (Giorgia)
- Horizontally unstructured data
  - Global will use ‘cubed-sphere’
  - Concept of rows and columns is gone
- New dynamical core- GungHo
- Shares UM physics, JULES, UKCA, CASIM, SOCRATES
- NetCDF file IO (using XIOS)



Git Migration Project

- MOSRS will reach End of Life in 2/3 years
- Decision to migrate to git/GitHub
- Rich Gilham representing JULES’ interests

- Git as a version control system is easy(ish)…
- …Working Practices for our needs hard!

- JULES potentially during 2024

NGMS + Git = Opportunities + Challenges

- Transfer experience between applications
- Consolidate effort eg for WPs, Training, GPUs, cloud…
- Get science into multiple applications ‘for free’
- Get rid of niche/legacy eg fcm_make
- GitHub automation opportunities

- LFRic steeper initial learning curve to run and develop
- LFRic won’t be fully open source until the UM Physics is sorted
- GitHub’s powerful toolbox could ensnare the over-enthusiastic
- Git makes it easy for developers to fork and never come back, losing community benefit

What to expect

- **Nothing breaks today**

- GitHub- comms plan and change management during 2024
    - Longer-term build-up of automation etc
    - JULES integral to overall plan for all the ‘simulation models’

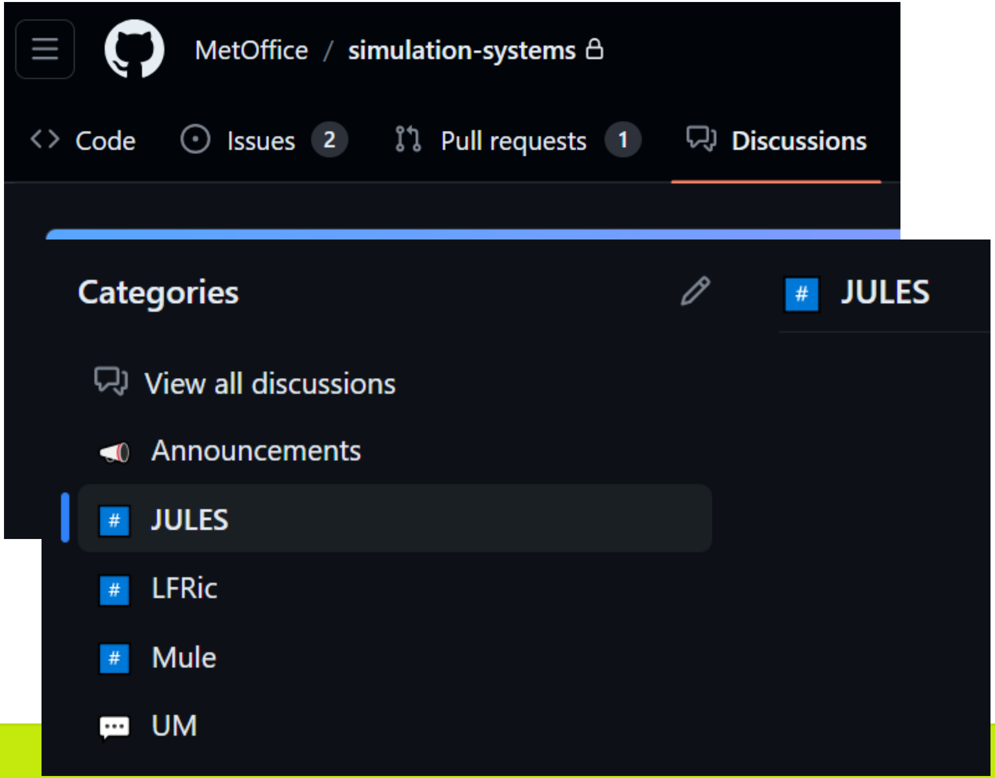
- Next Generation Modelling System & LFRic applications
    - Long-term decision around standalone JULES tech roadmap- role of LFRic
    - exaJULES a crucial project

Some smaller things…

Met Office

Announcing GitHub Forums

- Met Office is closing its external Yammer forums
- Accessibility for JULES always unsatisfactory
- Transition JULES, UM and LFRic user forums to GitHub
- Go-live in September



Open Sourcing JULES- an update

- Aim to transition to BSD-3 Open-Source licence
- Work to satisfy contributor completed; now waiting for legal sign-off
- JULES docs will move to BSD-3 imminently (Lianne will say more)

Benefits:

- Sharing code with referees will be easier (if you're already doing this but not following the process, stop!)
- Generate DOIs to credit releases and tech developments
- Aligns with broader vision

Code and Module Owners

- Owners compliment Science/Code reviewers by using their situational awareness of how their area is evolving
  - Breadth vs depth
  - Proven very robust

- **Giorgia Line is now lead Code Owner**
  - Rich Gilham now deputy
  - All tickets need Code Owner approval

Working Practices & LFRic Apps

Jenny

The Problem

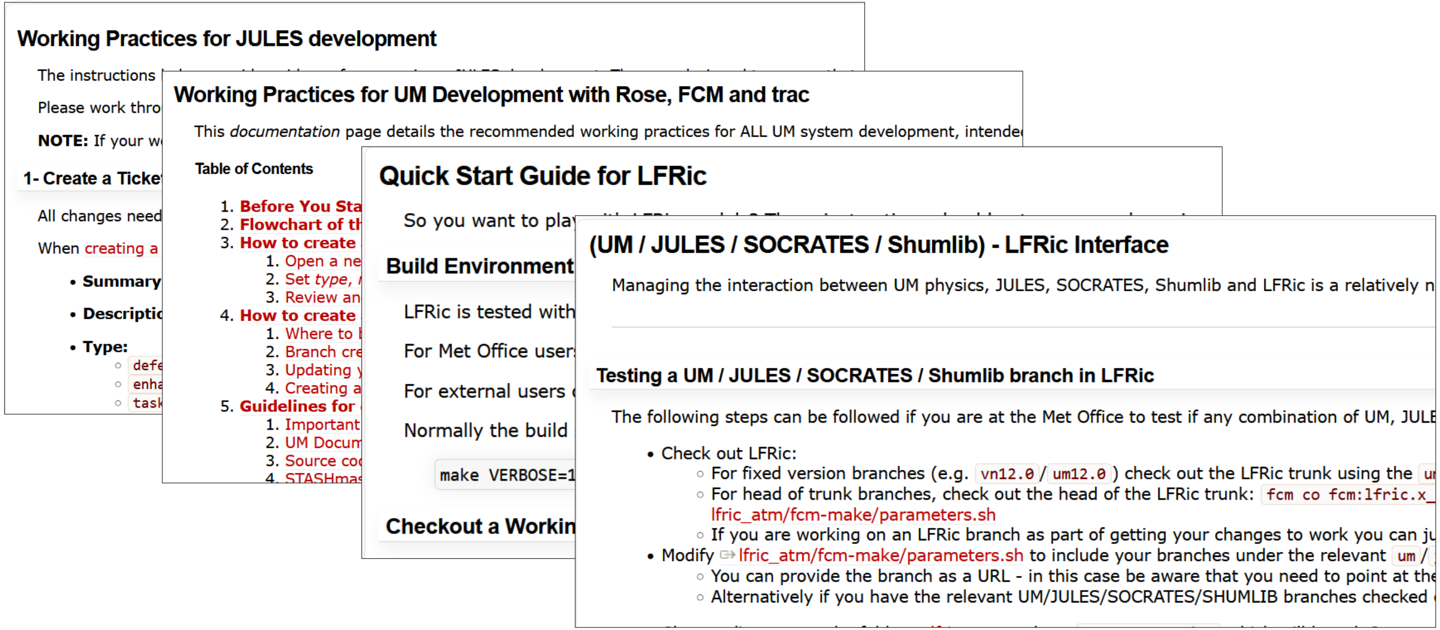
- All developers are working across many codebases
- Developer experience is disjointed
- Information is widely spread and hard to find

Photo by Xavi Cabrera on Unsplash

The Aim

- Cohesive experience
- Accessible documentation
- Unified working practices
  - doesn’t mean they have to be identical

Working Practices



Working Practices for JULES development

The instructions

Please work thro

NOTE: If your w

1- Create a Ticke

All changes need

When creating a

- **Summary**
- **Descriptio**
- **Type:**
  - defe
  - enha
  - task

Working Practices for UM Development with Rose, FCM and trac

This *documentation* page details the recommended working practices for ALL UM system development, intende

Quick Start Guide for LFRic

So you want to play

Build Environment

LFRic is tested with

For Met Office user

For external users

Normally the build

```
make VERBOSE=1
```

Checkout a Workin

(UM / JULES / SOCRATES / Shumlib) - LFRic Interface

Managing the interaction between UM physics, JULES, SOCRATES, Shumlib and LFRic is a relatively n

Testing a UM / JULES / SOCRATES / Shumlib branch in LFRic

The following steps can be followed if you are at the Met Office to test if any combination of UM, JULE

- Check out LFRic:
  - For fixed version branches (e.g. `vn12.0` / `um12.0`) check out the LFRic trunk using the `ur`
  - For head of trunk branches, check out the head of the LFRic trunk: `fcm co fcm:lfric.x_` lfric_atm/fcm-make/parameters.sh
  - If you are working on an LFRic branch as part of getting your changes to work you can ju
- Modify lfric_atm/fcm-make/parameters.sh to include your branches under the relevant `um` /
  - You can provide the branch as a URL - in this case be aware that you need to point at the
  - Alternatively if you have the relevant UM/JULES/SOCRATES/SHUMLIB branches checked

Working Practices

- JULES, UM, UKCA and LFRic instructions in one place
- Publically available
- As unified as possible

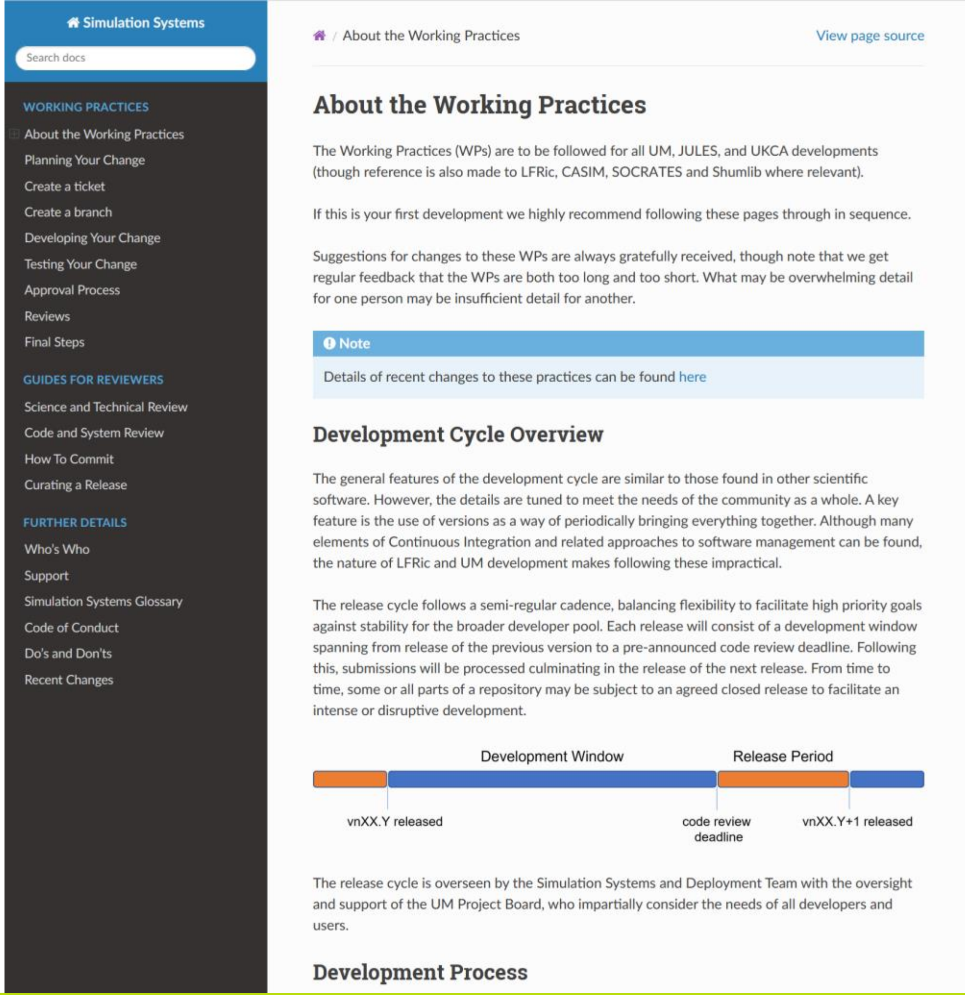
Met Office

Working Practices

- Written in Sphinx
- Stored in Github

- JULES, UM and UKCA live

https://metoffice.github.io/simulation-systems/



Simulation Systems

Search docs

WORKING PRACTICES

About the Working Practices

Planning Your Change

Create a ticket

Create a branch

Developing Your Change

Testing Your Change

Approval Process

Reviews

Final Steps

GUIDES FOR REVIEWERS

Science and Technical Review

Code and System Review

How To Commit

Curating a Release

FURTHER DETAILS

Who's Who

Support

Simulation Systems Glossary

Code of Conduct

Do's and Don'ts

Recent Changes

/ About the Working Practices

View page source

About the Working Practices

The Working Practices (WPs) are to be followed for all UM, JULES, and UKCA developments (though reference is also made to LFRic, CASIM, SOCRATES and Shumlib where relevant).

If this is your first development we highly recommend following these pages through in sequence.

Suggestions for changes to these WPs are always gratefully received, though note that we get regular feedback that the WPs are both too long and too short. What may be overwhelming detail for one person may be insufficient detail for another.

Note

Details of recent changes to these practices can be found here

Development Cycle Overview

The general features of the development cycle are similar to those found in other scientific software. However, the details are tuned to meet the needs of the community as a whole. A key feature is the use of versions as a way of periodically bringing everything together. Although many elements of Continuous Integration and related approaches to software management can be found, the nature of LFRic and UM development makes following these impractical.

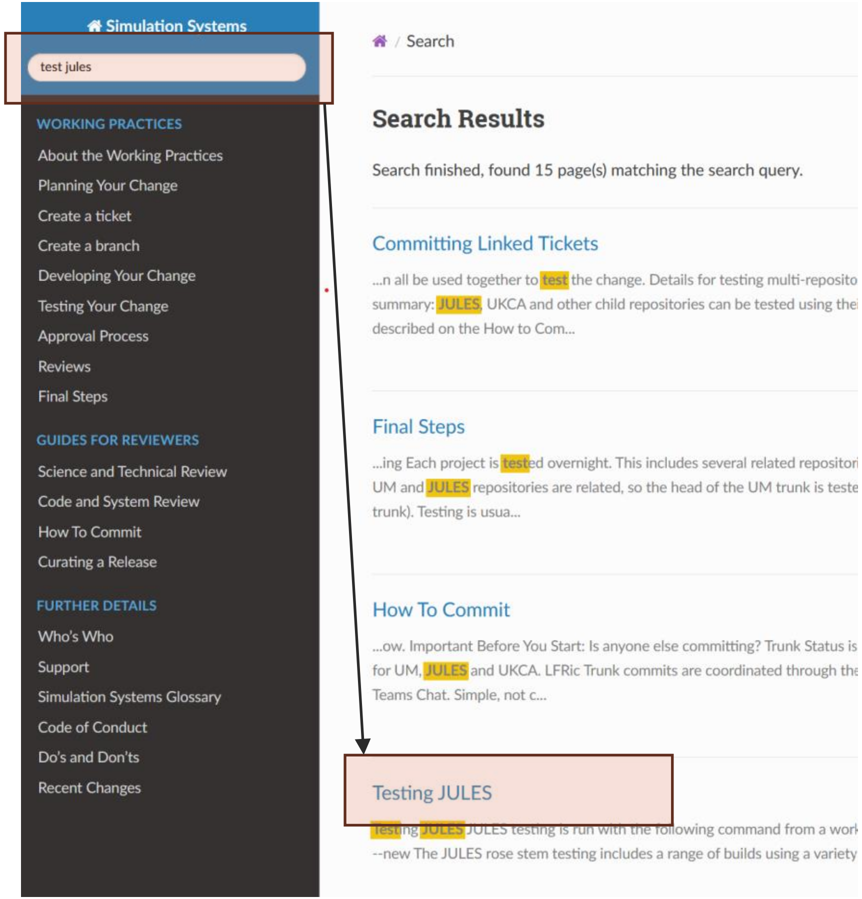
The release cycle follows a semi-regular cadence, balancing flexibility to facilitate high priority goals against stability for the broader developer pool. Each release will consist of a development window spanning from release of the previous version to a pre-announced code review deadline. Following this, submissions will be processed culminating in the release of the next release. From time to time, some or all parts of a repository may be subject to an agreed closed release to facilitate an intense or disruptive development.

The release cycle is overseen by the Simulation Systems and Deployment Team with the oversight and support of the UM Project Board, who impartially consider the needs of all developers and users.

Development Process

Working Practices

- Search facility



Simulation Systems

test jules

WORKING PRACTICES

- About the Working Practices
- Planning Your Change
- Create a ticket
- Create a branch
- Developing Your Change
- Testing Your Change
- Approval Process
- Reviews
- Final Steps

GUIDES FOR REVIEWERS

- Science and Technical Review
- Code and System Review
- How To Commit
- Curating a Release

FURTHER DETAILS

- Who's Who
- Support
- Simulation Systems Glossary
- Code of Conduct
- Do's and Don'ts
- Recent Changes

/ Search

Search Results

Search finished, found 15 page(s) matching the search query.

Committing Linked Tickets

...n all be used together to test the change. Details for testing multi-reposito summary: JULES, UKCA and other child repositories can be tested using thei described on the How to Com...

Final Steps

...ing Each project is tested overnight. This includes several related repositori UM and JULES repositories are related, so the head of the UM trunk is teste trunk). Testing is usua...

How To Commit

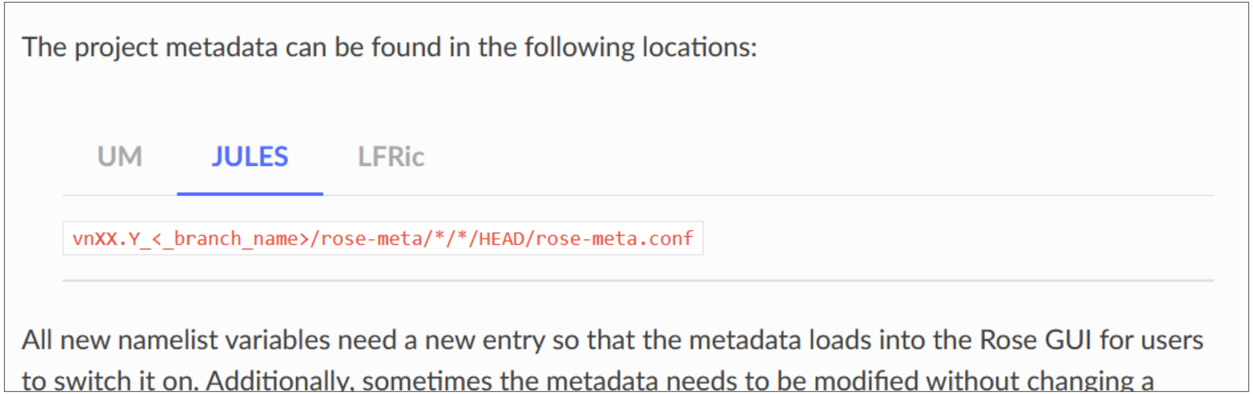
...ow. Important Before You Start: Is anyone else committing? Trunk Status is for UM, JULES and UKCA. LFRic Trunk commits are coordinated through the Teams Chat. Simple, not c...

Testing JULES

Testing JULES JULES testing is run with the following command from a work --new The JULES rose stem testing includes a range of builds using a variety

Working Practices

- Search facility
- Highlight project differences



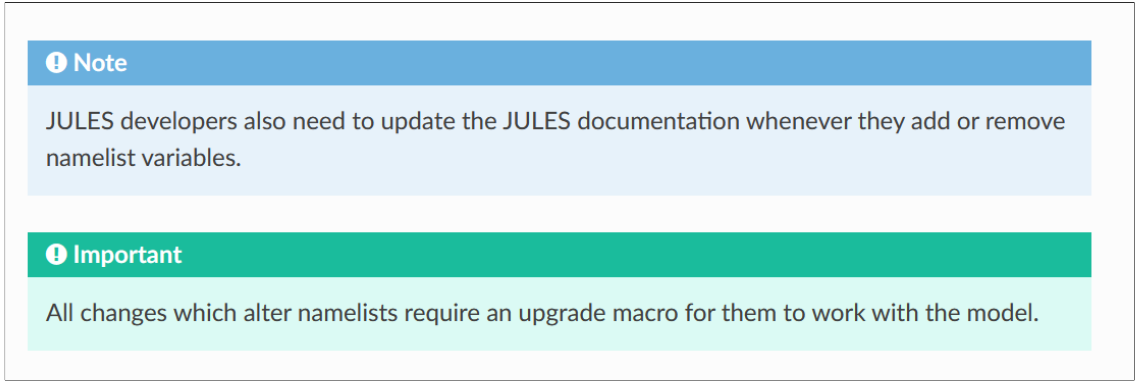
The project metadata can be found in the following locations:

```
vnXX.Y_<_branch_name>/rose-meta/*/*/HEAD/rose-meta.conf
```

All new namelist variables need a new entry so that the metadata loads into the Rose GUI for users to switch it on. Additionally, sometimes the metadata needs to be modified without changing a

Working Practices

- Search facility
- Highlight project differences
- Highlight key information

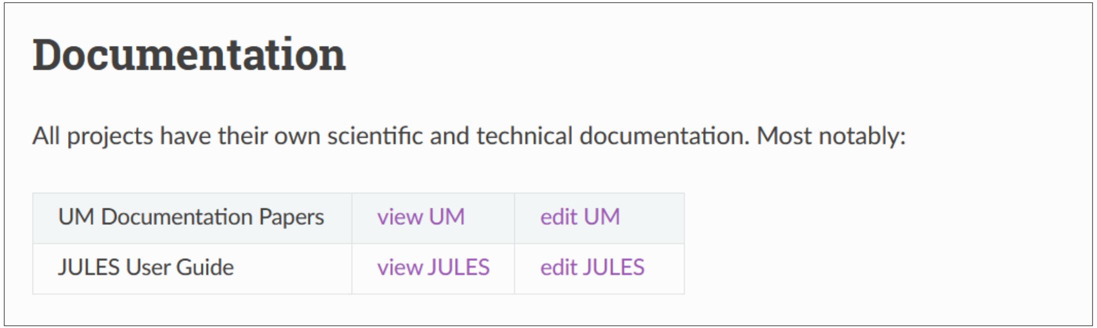


JULES developers also need to update the JULES documentation whenever they add or remove namelist variables.

All changes which alter namelists require an upgrade macro for them to work with the model.

Working Practices

- Search facility
- Highlight project differences
- Highlight key information
- Wiki links



Documentation

All projects have their own scientific and technical documentation. Most notably:

UM Documentation Papers	view UM	edit UM
JULES User Guide	view JULES	edit JULES

LFRic

- LFRic Apps and LFRic Core
  - Splitting repository into two
  - JULES interface code to be in LFRic Apps
- Rose Stem
  - Unified rose-stem testing for all LFRic Applications
- Upgrade Macros
  - Easier to pull through science from one model to another
- Releases
  - LFRic Apps release synchronised with UM and JULES

https://metoffice.github.io/simulation-systems/

Fab & GitHub

Giorgia

Met Office

What is Fab?

Next generation build system

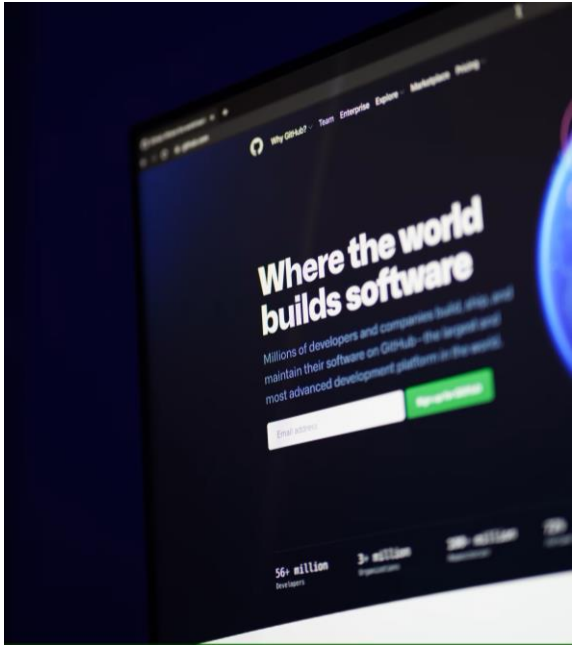
Designed for scientific software and science developers!

Grab Tool	For extracting source code from a repo or working copy
Build Tool	To compile the source

Why Fab?

Eventually, you’re going to have to for;

- Git migration
  - Migration to Git can be made as simple as a single line change
- The move to NGMS
  - Fab is a requirement for leveraging optimisations from LFRic applications



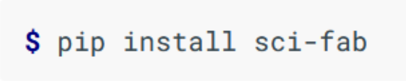
Where the world builds software

Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world.

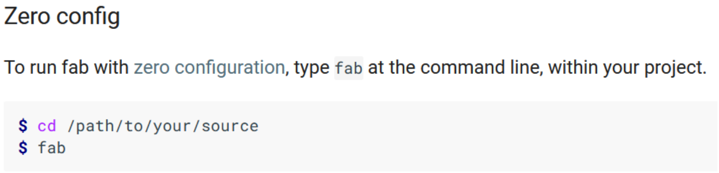
Why Fab?

But it also has many benefits:

- Consistent approach across projects
- Flexibility
  - Python scripts make custom steps simple
- Quick and easy to install and use
  - Can be installed via pip install (as **sci-fab**!)
  - Available on Conda Forge
- Zero Configuration option available
  - Just run ‘fab’!



```
$ pip install sci-fab
```



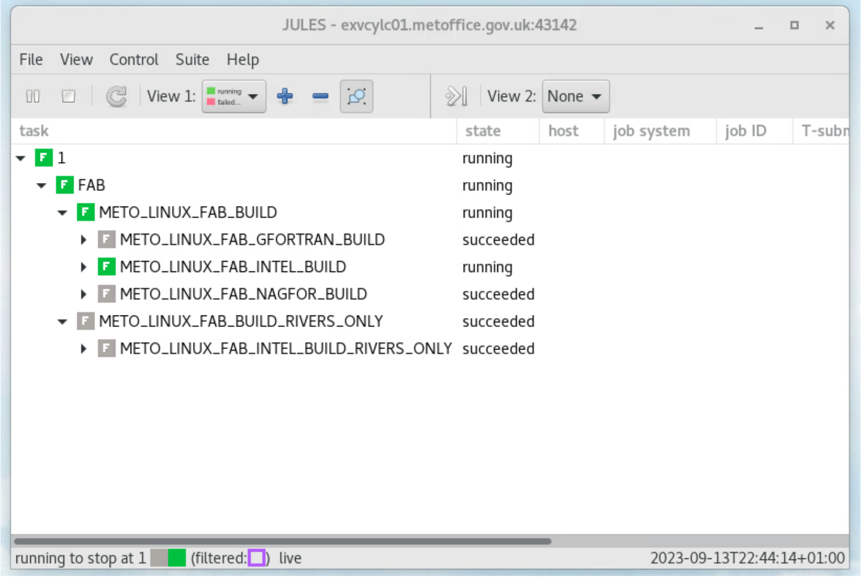
Zero config

To run fab with zero configuration, type `fab` at the command line, within your project.

```
$ cd /path/to/your/source
$ fab
```

Where are we up to?

- JULES beta configuration available in rose stem
    - Currently runs as part of our regular testing
- Fab vn1.0 JULES configuration in progress
    - Using the UM-Fab vn1.0 configuration as a template



task	state	host	job system	job ID	T-subn
1	running				
FAB	running				
METO_LINUX_FAB_BUILD	running				
METO_LINUX_FAB_GFORTRAN_BUILD	succeeded				
METO_LINUX_FAB_INTEL_BUILD	running				
METO_LINUX_FAB_NAGFOR_BUILD	succeeded				
METO_LINUX_FAB_BUILD_RIVERS_ONLY	succeeded				
METO_LINUX_FAB_INTEL_BUILD_RIVERS_ONLY	succeeded				

running to stop at 1 (filtered:) live 2023-09-13T22:44:14+01:00

Help us help you

- We need users and early adopters!
- The more use cases we test now, the better prepared we'll be.
  - Find and fix problems while we have alternatives.
  - Make Git migration smoother

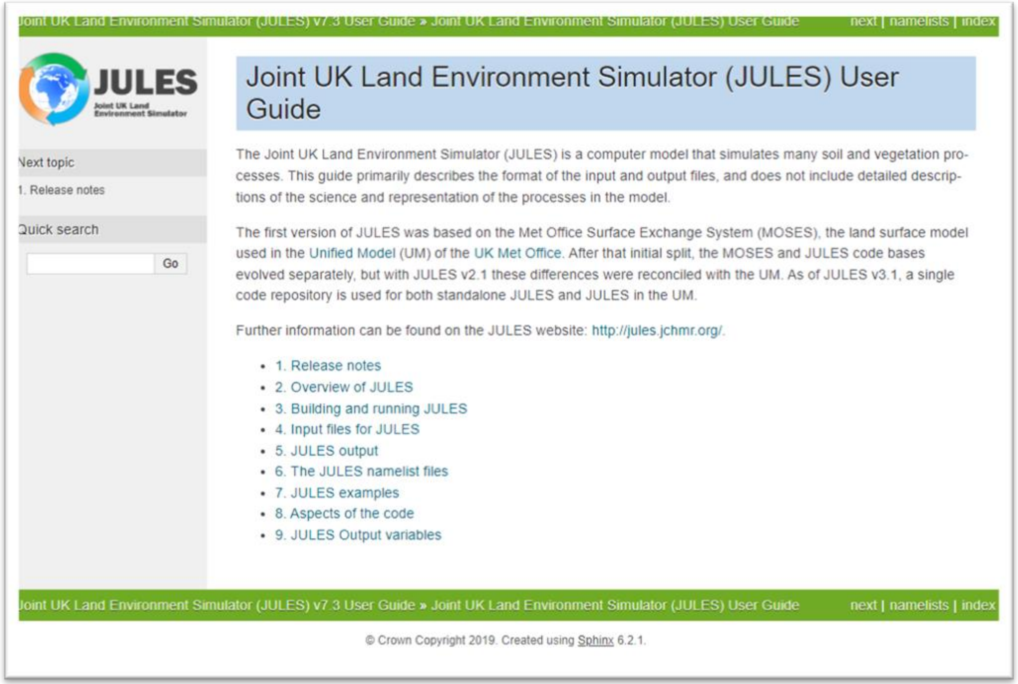
JULES docs GitHub Migration

Lianne

Met Office

Overview

- The JULES User Guide is built using Sphinx
- Met Office is undergoing a major Git Migration project
- As a result, the JULES documentation will be moving from Subversion/Trac to Git/GitHub



Joint UK Land Environment Simulator (JULES) v7.3 User Guide » Joint UK Land Environment Simulator (JULES) User Guide
next | namelists | index
JULES
Joint UK Land Environment Simulator
Next topic
1. Release notes
Quick search
Go

Joint UK Land Environment Simulator (JULES) User Guide

The Joint UK Land Environment Simulator (JULES) is a computer model that simulates many soil and vegetation processes. This guide primarily describes the format of the input and output files, and does not include detailed descriptions of the science and representation of the processes in the model.

The first version of JULES was based on the Met Office Surface Exchange System (MOSES), the land surface model used in the Unified Model (UM) of the UK Met Office. After that initial split, the MOSES and JULES code bases evolved separately, but with JULES v2.1 these differences were reconciled with the UM. As of JULES v3.1, a single code repository is used for both standalone JULES and JULES in the UM.

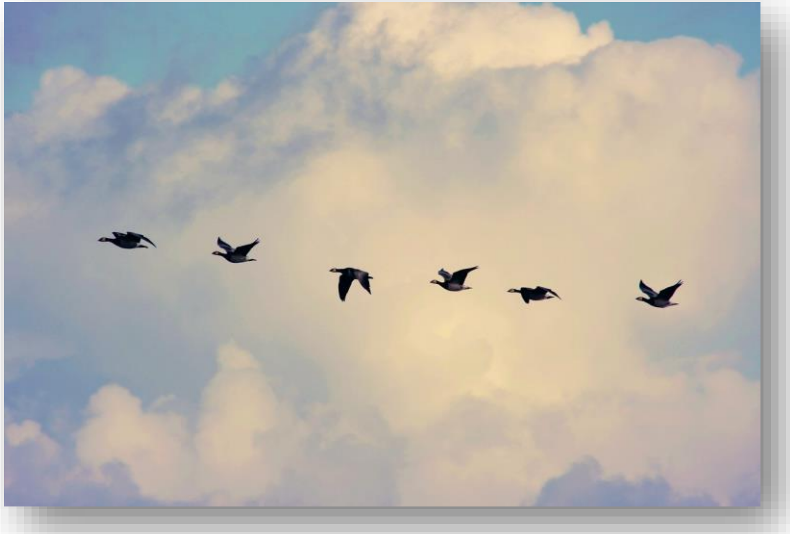
Further information can be found on the JULES website: http://jules.jchmr.org/.

- 1. Release notes
- 2. Overview of JULES
- 3. Building and running JULES
- 4. Input files for JULES
- 5. JULES output
- 6. The JULES namelist files
- 7. JULES examples
- 8. Aspects of the code
- 9. JULES Output variables

Joint UK Land Environment Simulator (JULES) v7.3 User Guide » Joint UK Land Environment Simulator (JULES) User Guide
next | namelists | index

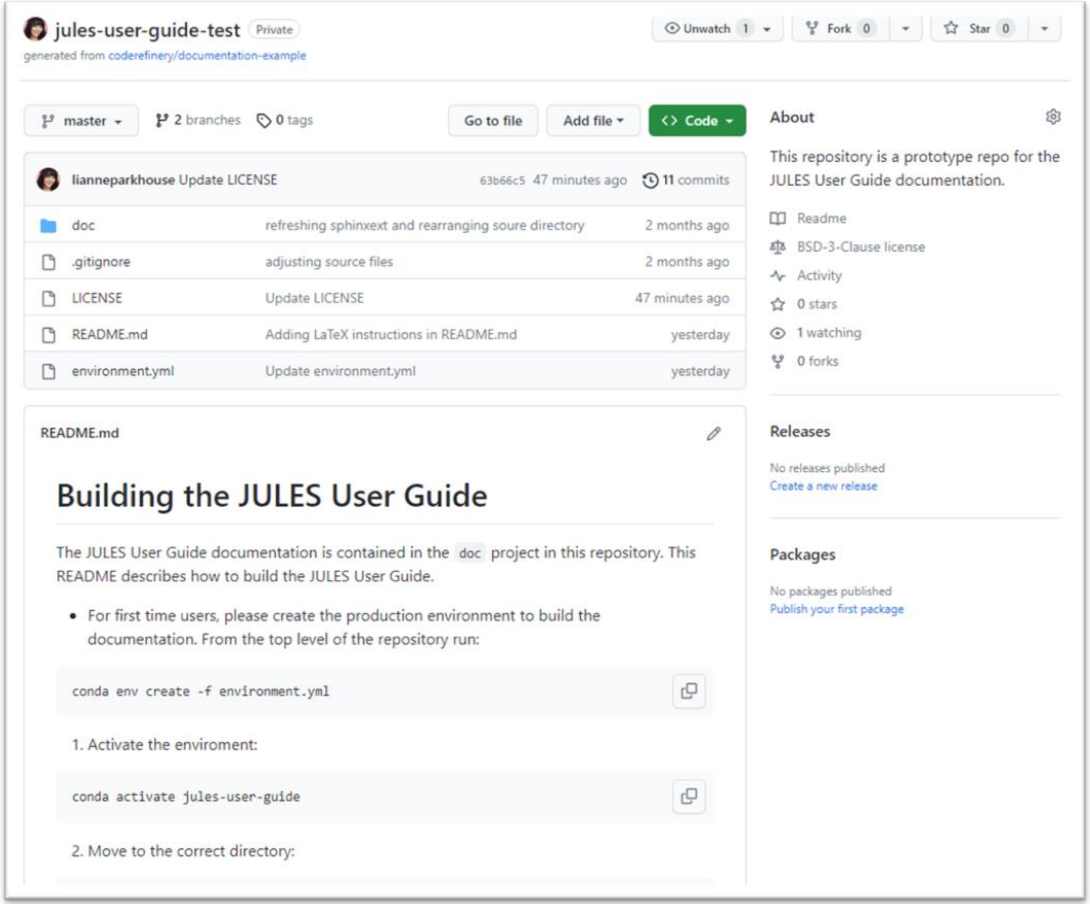
What are the benefits?

- An “easy win” in terms of Git Migration
- More streamlined docs change and build process
- Enhances collaborative development
- Will align with the BSD-3 relicense
- Allows assignment of persistent identifiers/DOIs



First look

- The structure of the `docs/user_guide` is the same
- Easier to checkout and build the JULES documentation
- MO users no longer need to rely on SciTools



jules-user-guide-test Private

generated from coderefinery/documentation-example

lianneparkhouse Update LICENSE 63b66c5 47 minutes ago 11 commits

doc	refreshing sphinxext and rearranging soure directory	2 months ago
.gitignore	adjusting source files	2 months ago
LICENSE	Update LICENSE	47 minutes ago
README.md	Adding LaTeX instructions in README.md	yesterday
environment.yml	Update environment.yml	yesterday

About

This repository is a prototype repo for the JULES User Guide documentation.

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

README.md

Building the JULES User Guide

The JULES User Guide documentation is contained in the `doc` project in this repository. This README describes how to build the JULES User Guide.

- For first time users, please create the production environment to build the documentation. From the top level of the repository run:

```
conda env create -f environment.yml
```

1. Activate the enviroment:

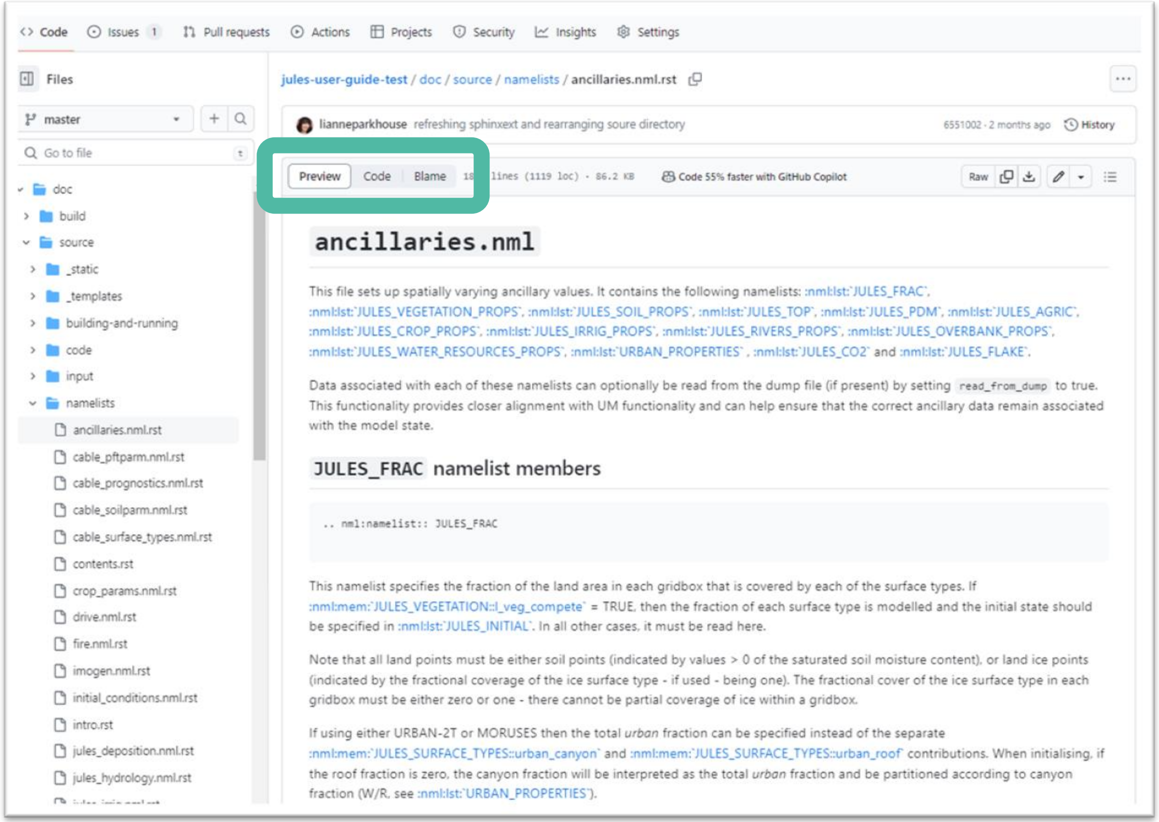
```
conda activate jules-user-guide
```

2. Move to the correct directory:

Met Office

First look

- More user-friendly browsing
- Automated testing/deployment
- Option to preview the full HTML of the .rst files
- Can easily view the commit history



ancillaries.nml

This file sets up spatially varying ancillary values. It contains the following namelists: :nml:lst:`JULES_FRAC`, :nml:lst:`JULES_VEGETATION_PROPS`, :nml:lst:`JULES_SOIL_PROPS`, :nml:lst:`JULES_TOP`, :nml:lst:`JULES_PDM`, :nml:lst:`JULES_AGRIC`, :nml:lst:`JULES_CROP_PROPS`, :nml:lst:`JULES_IRRIG_PROPS`, :nml:lst:`JULES_RIVERS_PROPS`, :nml:lst:`JULES_OVERBANK_PROPS`, :nml:lst:`JULES_WATER_RESOURCES_PROPS`, :nml:lst:`URBAN_PROPERTIES` , :nml:lst:`JULES_CO2` and :nml:lst:`JULES_FLAKE`.

Data associated with each of these namelists can optionally be read from the dump file (if present) by setting `read_from_dump` to true. This functionality provides closer alignment with UM functionality and can help ensure that the correct ancillary data remain associated with the model state.

JULES_FRAC namelist members

```
.. nml:namelist:: JULES_FRAC
```

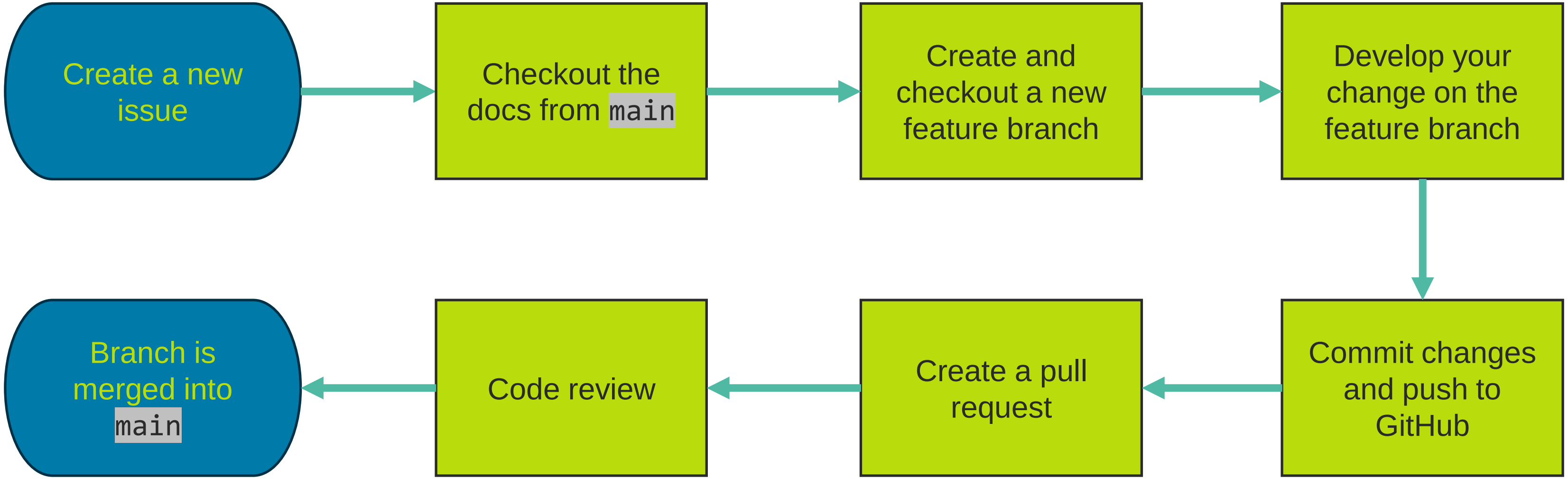
This namelist specifies the fraction of the land area in each gridbox that is covered by each of the surface types. If :nml:mem:`JULES_VEGETATION::l_veg_compete` = TRUE, then the fraction of each surface type is modelled and the initial state should be specified in :nml:lst:`JULES_INITIAL`. In all other cases, it must be read here.

Note that all land points must be either soil points (indicated by values > 0 of the saturated soil moisture content), or land ice points (indicated by the fractional coverage of the ice surface type - if used - being one). The fractional cover of the ice surface type in each gridbox must be either zero or one - there cannot be partial coverage of ice within a gridbox.

If using either URBAN-2T or MORUSES then the total *urban* fraction can be specified instead of the separate :nml:mem:`JULES_SURFACE_TYPES::urban_canyon` and :nml:mem:`JULES_SURFACE_TYPES::urban_roof` contributions. When initialising, if the roof fraction is zero, the canyon fraction will be interpreted as the total *urban* fraction and be partitioned according to canyon fraction (W/R, see :nml:lst:`URBAN_PROPERTIES`).

Met Office

Development process



Create a new issue

Checkout the docs from `main`

Create and checkout a new feature branch

Develop your change on the feature branch

Commit changes and push to GitHub

Create a pull request

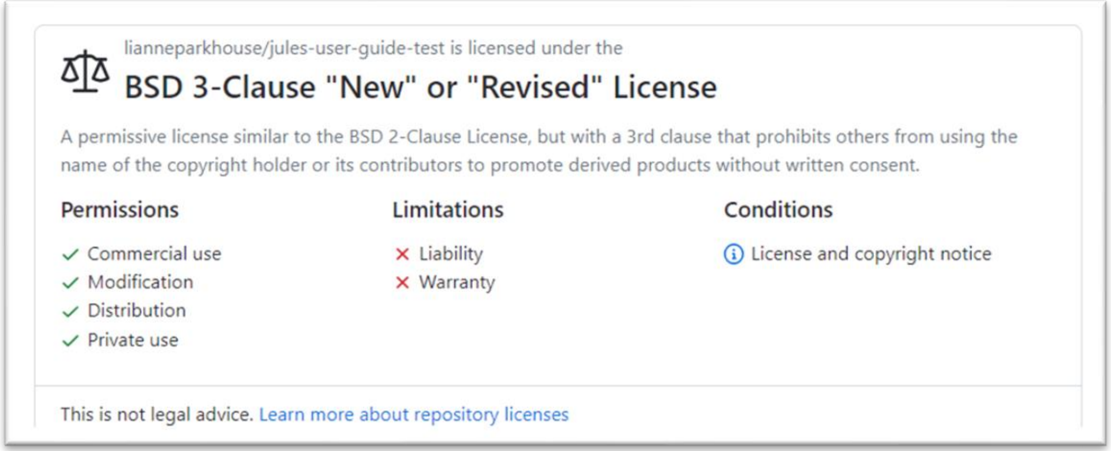
Code review

Branch is merged into `main`

BSD 3 relicense

- JULES docs will be in GitHub (thus relicensed) in time for the next release.

  October 2023: github.com/jules-lsm



lianneparkhouse/jules-user-guide-test is licensed under the

BSD 3-Clause "New" or "Revised" License

A permissive license similar to the BSD 2-Clause License, but with a 3rd clause that prohibits others from using the name of the copyright holder or its contributors to promote derived products without written consent.

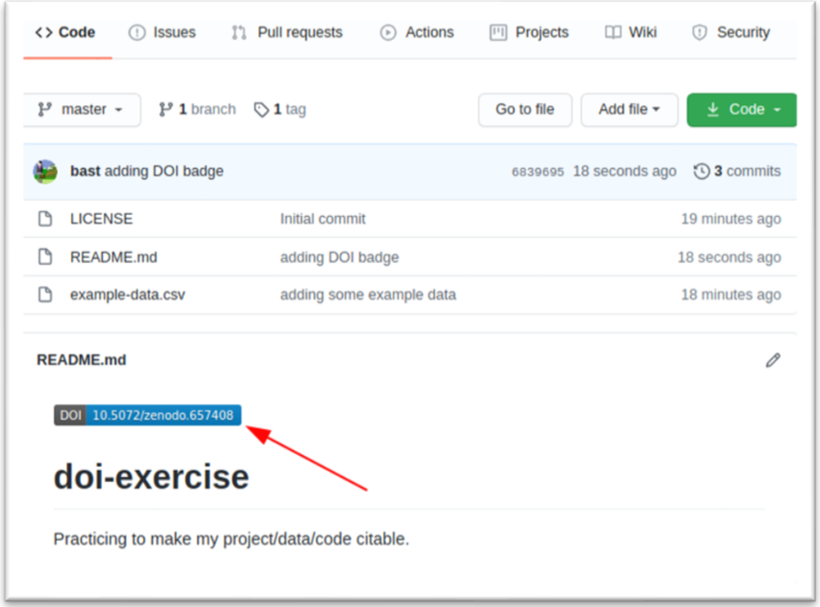
Permissions	Limitations	Conditions
✓ Commercial use	✗ Liability	ⓘ License and copyright notice
✓ Modification	✗ Warranty	
✓ Distribution		
✓ Private use		

This is not legal advice. Learn more about repository licenses

BSD 3 licence allows unlimited redistribution for **any** purpose, provided the copyright notices and the licence's disclaimers of warranty are maintained: https://opensource.org/license/bsd-3-clause

Archiving GitHub code in Zenodo

- Zenodo is an open repository maintained by CERN
- Will allow us to assign DOIs to any version of the JULES docs
- Ensures that the docs for JULES can be cited easily in papers
- Research outputs and resources discoverable and citable for the long term



bast adding DOI badge 6839695 18 seconds ago 3 commits

LICENSE	Initial commit	19 minutes ago
README.md	adding DOI badge	18 seconds ago
example-data.csv	adding some example data	18 minutes ago

README.md

DOI 10.5072/zenodo.657408

doi-exercise

Practicing to make my project/data/code citable.

Source: CodeRefinery guide on "Making your project citable": https://coderefinery.github.io/github-without-command-line/doi/

Conclusions

Rich

Take home messages

- Upcoming technical changes offer JULES opportunities
- GitHub transition is responding to a risk but opens doors to user experience improvements
- NGMS offers new routes for exploiting JULES science via the LFRic applications
- Fab is a key enabler for both, and will improve the present JULES-standalone easier
- Early steps towards GitHub are in progress

Timeline

- Working Practices- live
- Fab first steps- try it now!
- GitHub forums- this month
- JULES doc to GitHub- October release
- Fab transition- 2024
- GitHub transition & open source(?) (2024)
- LFRic_atm operational ~2027

Technical Surgery Breakout Session

- Feedback?
  - Tell us why ‘JULES is hard to use’
- Questions?
- Problems?
- Helping hand?

Safe space for all

Thank you… Q&A

richard.gilham@metoffice.gov.uk

jennifer.hickson@metoffice.gov.uk

giorgia.line@metoffice.gov.uk

lianne.parkhouse@metoffice.gov.uk