Reproducible, Replicable, Reusable Research (RRRR) in Computational Geodynamics

Louis Moresi (and the Underworld development team)

ProfDiablo@underworldcode.org ; http://www.underworldcode.org ;

DI33C-0047

Abstract

Reproducibility (the ability to re-run an experiment with the same outcome), and replicability (that the experiment can be independently verified) are core concepts in science and they ought to be straightforward to implement in computational disciplines.

It can be surprisingly hard to implement workflows that make results reproducible, particularly after a number of years have elapsed. Replicability contains a component of subjectivity: what does it take to have an independent result ? Do the algorithms need to be independent or only the implementation ? What does it mean to get the same answer in a numerical environment where errors are not derived from experimental "noise" ?

Going beyond each of these concepts is the idea that numerical models should be re-usable too. This means that it is possible to build new models and results from existing ones. This extends the expectations of publishing results to include the fact that people should be able to validate the numerical analysis and extend it with their own idea and data.

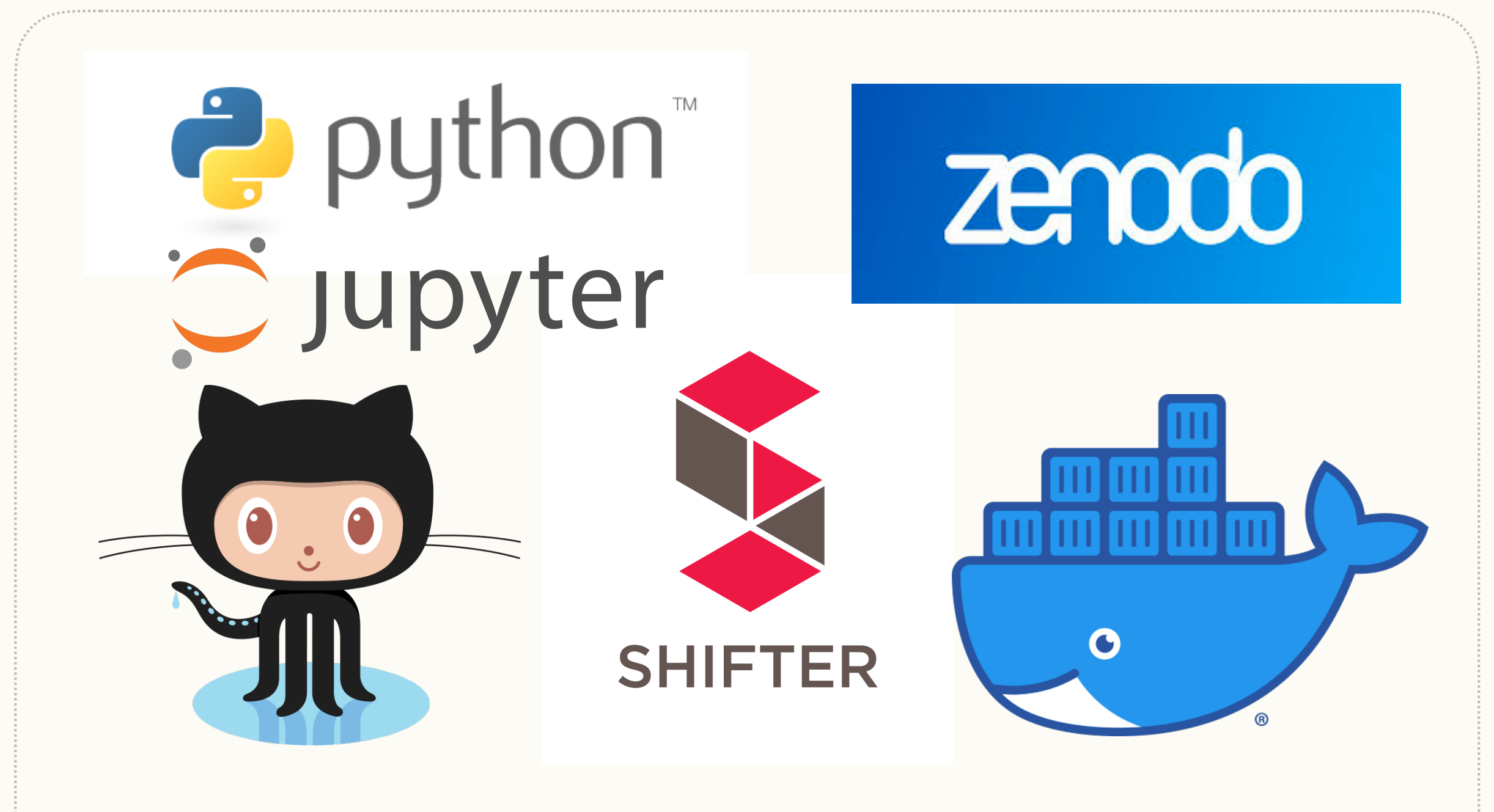
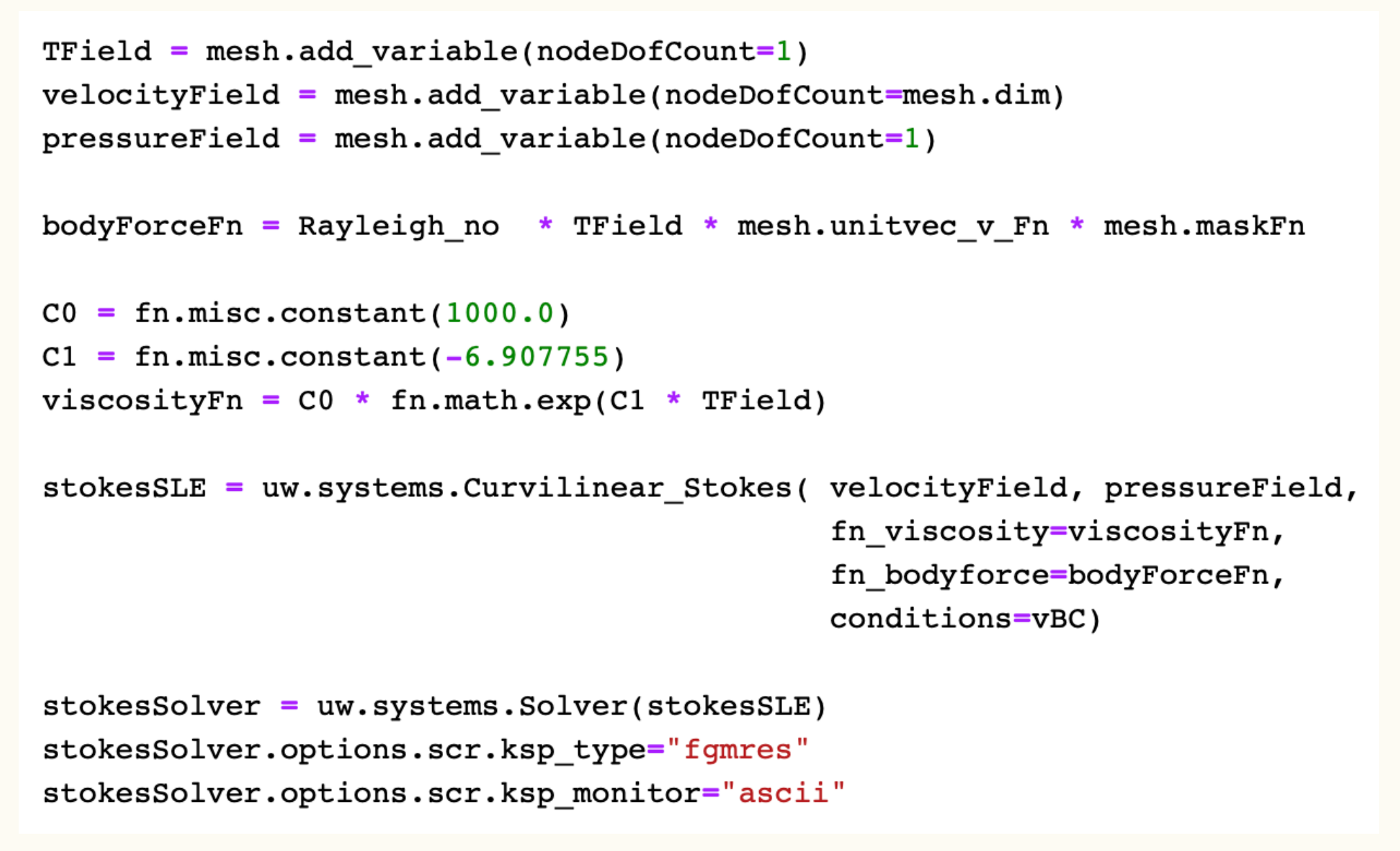


Figure 1: Assorted cyber-gadgetry that promotes RRRR

Reproducibility

It is relatively simple to ensure that a self-contained program can be re-run exactly if all the components are deterministic. We can make use of container systems such as docker that guarantee reproducibility of the execution environment provided that the hardware virtualisation is persistent through time. Containers can run in high performance (parallel) environments using systems such as shifter. If a model is initiated using external data, has an inherent sensitivity to (for example) the parallel decomposition or with some randomness is used to seed the simulation, then the notion of reproducibility needs some further development. A model is not reproducible if the data it depends upon cannot be marshalled at any point in the future exactly as it was at the first run time. Clearly this poses interesting problems for accessing continually evolving and very large datasets that cannot simply be bundled with the code. Randomness and non-deterministic algorithms mean that exact reproducibility is not possible. From run to run, we may instead consider 'replicating' a result.



```
TField = mesh.add_variable(nodeDofCount=1)
velocityField = mesh.add_variable(nodeDofCount=mesh.dim)
pressureField = mesh.add_variable(nodeDofCount=1)

bodyForceFn = Rayleigh_no  * TField * mesh.unitvec_v_Fn * mesh.maskFn

C0 = fn.misc.constant(1000.0)
C1 = fn.misc.constant(-6.907755)
viscosityFn = C0 * fn.math.exp(C1 * TField)

stokesSLE = uw.systems.Curvilinear_Stokes( velocityField, pressureField,
                                           fn_viscosity=viscosityFn,
                                           fn_bodyforce=bodyForceFn,
                                           conditions=vBC)


stokesSolver = uw.systems.Solver(stokesSLE)
stokesSolver.options.scr.ksp_type="fgmres"
stokesSolver.options.scr.ksp_monitor="ascii"
```

Figure 2: Code snippet showing use of uw.function to create an abstraction of the model away from the implementation

Replicability

A scientific result can be replicated if the experiments result in the same interpreted outcome each time it is conducted even if that is in a different laboratory or with a different implementation of an algorithm or with two algorithms that purport to solve the same equations.

Replicability is about understanding the uncertainty inherent in a scientific experiment and realising that a question cannot be considered to have been answered until we understand the scope of alternative interpretations of the same experiment or model. Benchmarking is an example of how we develop this knowledge.

One computational approach to assisting in replicability of numerical experiments is to create abstract model descriptions that can switch between algorithmic implementations or even physical representations of the problem at run-time or within an ensemble of models. See Figure 2 for an example of high-level model description.

Re-use

"... if I have seen further, it is by standing on the shoulders of giants." Isaac Newton (1675).

Reproducibility and Replicability allow us to trust a model and to interpret it in the context of its inherent uncertainty.

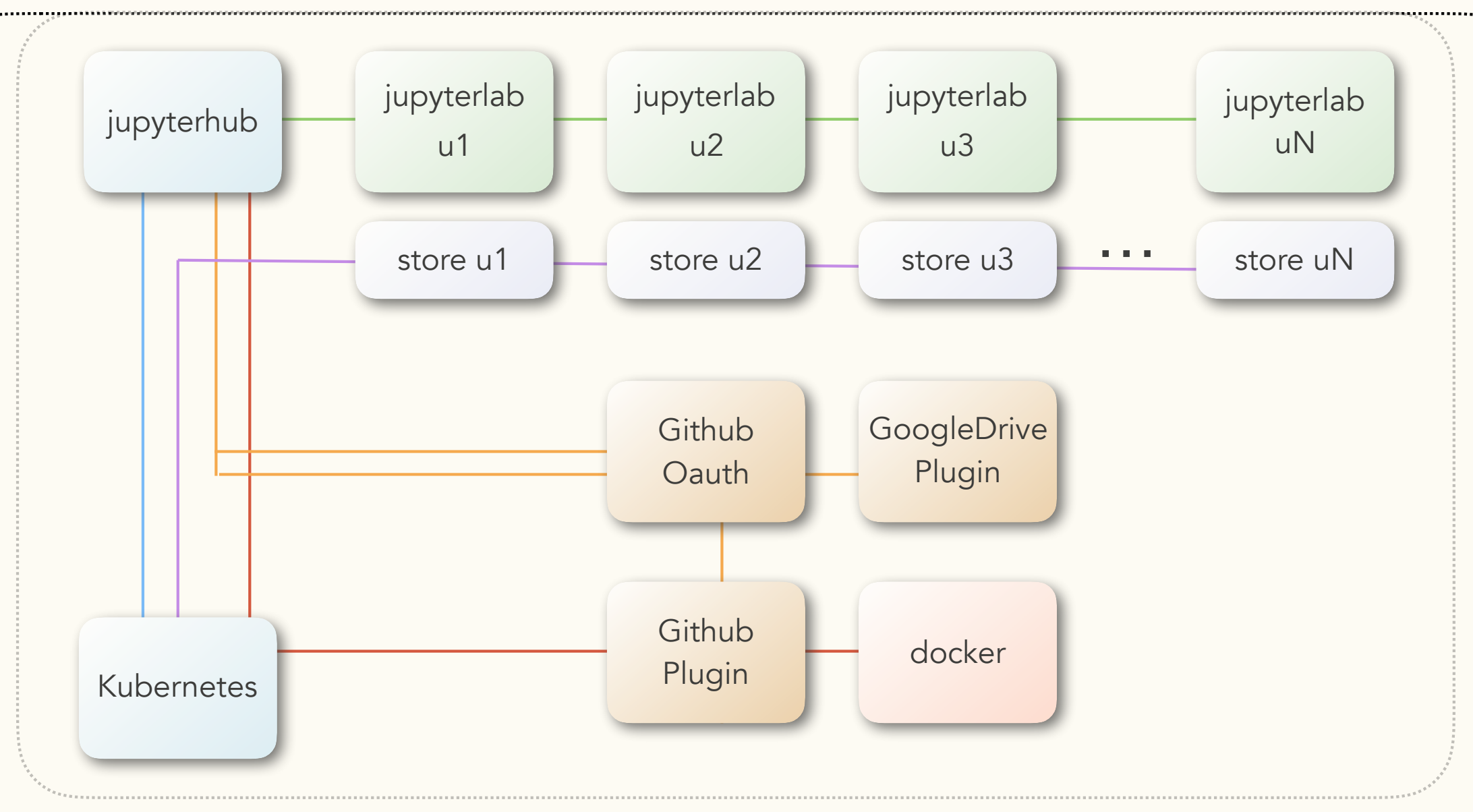


Figure 3: Architecture of the Underworld Cloud

If we actually wish to build on a model, then we also need to be able to modify and extend the work. This is very different from the slavish reproduction of a previous model but many of the technical issues are related.

To build upon a model, we must first be able to run that model (i.e. it should be **reproducible** for us). To understand the model as it runs, we ought to be able to explore its sensitivity and **replicate** some of the scientific results even if the model we wish to run has a different choice of parameters or updated data.

Both of these can be satisfied if we build code, workflows and scripts into a container that can be re-run repeatedly and consistiently.

The Underworld approach to model **re-use** is to supply simplified examples of published results in the form of jupyter notebooks that run, reliably in the cloud. Figure 3 shows the architecture of the Underworld cloud which allows models to be developed and run based on *cookbook* examples.

One goal is to transport those cloud-cookbooks examples directly to parallel HPC on demand and to provide automated mechanisms for exploring how implementation details impact the details of a given experiment.

References

Hughes, Ted. *The Iron Man*. London: Faber and Faber, 2005.

Newton, Isaac. "Letter to Robert Hooke," February 5, 1675.

Acknowledgements

On a personal note, these ideas were refined as a result of many conversations with Louise Kellogg at CIG and CIDER workshops. Illustration: Tom Gault, from Hughes (2005)