Event link

https://hpc.guix.info/events/2023/workshop/

Title

Reproducible Computational Environments with Binder

Abstract

Reproducible research is necessary to ensure that scientific work can be trusted. Funders and publishers are beginning to require that publications include access to the underlying data and the analysis code. The goal is to ensure that all results can be independently verified and built upon in future work. This is sometimes easier said than done! Sharing these research outputs means understanding data management, library sciences, software development, and continuous integration techniques: skills that are not widely taught or expected of academic researchers. A particularly steep barrier to working with codebases is setting up computational environments, and getting the combination of package versions just right can influence the reproducibility of code: from outright failures, to subtle changes in generated outputs. There are many tools available to manage your computational environment; but in this talk, we'll explore Project Binder and its subproject repo2docker, which aims to automate reproducibility best practices across a number of ecosystems. Binder can build portable computational environments, when requested, with all the information encoded in a single, clickable URL, which greases the wheels of collaborative research while reducing the toil involved. We will discuss how these concepts can apply to the HPC community.

Notes/Things to Remember

- repo2docker itself is not a package manager/solver. It can recognise specific file types and invoke the correct, *native* solver (e.g., it uses pip, conda, RStudio Package Manager, etc.)
    - Not duplicating effort
- repo2docker automates best practices from a number of communities - this can include Guix/HPC!
- Audience is users - make a pitch that this helps them be more productive, and they can ask sysadmins for that feature
- HPC is a shared env, wanna run a software stack, dependent on modules being loaded, some incantation before the batch job. Once that's setup, it's done for X time.
- Advantage for Binder, sharing workflow with someone, or trying someone else's, don't want to set it up. Components of stack may or may not be supported. Portability. "Can I run it here, can I run it there?" Containers bundle the stack for the application, independent of what's been deployed. Cross-institutional/multi-institutional training events, demo events.

- How do we get workflows to run on multiple HPC centres where stack will be different, workshops, demos, training events, the one off stacks. How to bring my settings for my project to another system.
- JM's experience: installing a tool yourself with large amount of files inside your own directory that you have to self-maintain. Which means every user has a copy of their self-installed stack.
- Something like repo2<container> can be rebuilt on the fly, ephemeral, they go away after a while, cleanup
- Portability between institutions
- How to get HPC workflow to run in the Cloud
    - Why? Because of where data lives
    - Dataset produced by another institution - they're big!
    - Package your workflow, bring compute to the data - Binder!
    - Need to develop in the env you're currently in - need to test, otherwise you need a cloud instance
- Archiving (repo2docker)
    - DMPs state data has to go somewhere
    - But there's an analysis software stack that needs to be packaged with that data
    - Repo2docker isn't foolproof, archive the container instead
    - Simplicity of taking a software definition and reliably generating an environment
- Launch with just a URL
    - Entire info in encoded within it
    - HPC application? Accessing HPC env from a browser directly (instead of going in and creating a tunnel out)
    - Launch JupyterLab automatically - ease of getting into environment with their tooling
    - Friction: HPC centre that already hosts a JupyterHub environment
    - Busy PIs
- Call to Action
    - How can concepts from Binder API/repo2<container> be used in a HPC centre
    - Acknowledging that the Cloud and HPC are all just server rooms
    - We migrate to whatever resources are available

Project Binder offers two great advantages to researchers which aid in reproducibility and collaborative efforts: portability and ease of entering the environment. By leveraging container technology, an entire software stack can be packaged in a way that it becomes portable across systems, which makes the setup process on a new system much easier and facilitates trying someone else's code without needing to install their environment. repo2docker provides a consistent interface for generating such portable environments. Then by exposing the container over a URL

Talk Outline - 40mins + 5mins Q&A

- Who am I? Establish credentials
    - JupyterHub/Binder core team member

    - *The Turing Way* core team member
- What does reproducible mean? Specifically, *computational* reproducibility
    - The code vs. data matrix
    - Computational reproducibility: What dependencies do you need to run the code? What command do you need to run?
- Why is reproducibility important?
    - Mistakes in research have real world effects
- Why is reproducibility so difficult?
    - We are not incentivised to learn or implement these skills
    - Software packaging ecosystems are a mess, dude
- What things go into reproducibility?
    - Version control - snapshots in time
    - Testing - what changed? Be explicit about it!
        - For big HPC calculations, lean towards unit testing, fail-fast methods, end-to-end/integration tests that can run with small amount of data and for short runtimes
    - Continuous Integration - take the toil out of seeing what changed
    - Software environment management - yikes
- Project Binder/mybinder.org
    - What is it? Vocab
    - Binder/mybinder.org UX
        - Clicking a link is an incredibly simple and frictionless way to enter an environment
        - The full environment specification is encapsulated within the URL
        - Benefits busy PIs and sharing work without needing to setup a whole new environment
    - BinderHub tech stack/repo2docker
        - Containerisation tends towards Portability, move stacks between systems without setup
        - Reduce toil of one-off setups: events, workshops, training, visiting other systems
    - ~~Federation~~
    - ~~Repo2docker~~
- Call to Action
    - How can concepts from the Binder API (URL generation)/repo2$CONTAINER be applied to a HPC centre?
    - Acknowledge that “the cloud” and a “HPC centre” are all just “someone else’s computers”
    - We migrate to the resources that are available (and meet the needs of our workflows)

—

- How is this relevant in HPC?

    - The Binder team learned a lot about the kinds of expectations users have and mistakes they make when trying to properly implement reproducible workflows
    - If you are developing best practices around using containers in HPC, then it's totally in scope to upstream these to repo2docker!
        - https://github.com/jupyterhub/repo2docker/pull/1048
- Parting message/questions
    - Is the HPC community open to using containers at all? Do you see a need for them? If not, repo2docker may not be the right tool
    - If yes, there are other tools that circumvent the privileges concerns that arise specifically with Docker, e.g., podman, singularity/apptainer.
        - https://github.com/ncar-xdev/repo2apptainer
    - Can also build the images outside the HPC system in an isolated VM and only run images inside the HPC
        - Reference 2i2c/GESIS work to allow a JupyterHub to use the Binder API to build an image, but not launch it