Managing Academic Software Development

Date: November 9, 2022

Presented by: Sam Mangham (University of Southampton)

(The slides are available at https://slides.com/sammangham/2022-11-09-hpc-bp.)

Q. Any guidance regarding how to change version numbers? For example, what does version 1.0 usually mean?

A. A common strategy is (as mentioned in the comments) Semantic Versioning - 1.0.0 is your first stable build you share with users (e.g. the first time you use it in a paper), then you increment the third digit when you patch a bug (e.g. 1.0.0 -> 1.0.1), the second when you add a new feature that doesn't break backwards compatibility (e.g. 1.0.1 -> 1.1.0), and finally the first digit when you *do* break backwards compatibility (e.g. 1.1.0 -> 2.0.0). Academic software often also does a major release for each paper, as changing algorithms etc. might not break backwards compatibility in a technical sense, but it does mean that different versions will not give the same scientific results.

C. I think you can include the DOI since Zenodo lets you pre-allocate a DOI.

A. You can pre-allocate a DOI for an *upload*, but not for an automatic upload for a linked GitHub repository (or couldn't last I checked!)

C. For compiler, libs etc version: I think this is where EasyBuild is quite good as it is really prescriptive. So quoting: the software was built with EasyBuild version 4.6.2, using the supplied EasyConfig file Foo-foss-2021a.eb

C. Semantic versioning is one way to determine version #'s: https://semver.org

Q. Outside the US DOE, how popular is spack for managing compiler libraries, etc dependencies? We have had success using spack for these issues but I am not sure how widely adopted it is outside the DOE…

A. I don't use Spack so I'm afraid I can't comment! I normally use Docker or Singularity for system-level dependency management, but I see there are Spack/Singularity integrations.

C. I think Spack is more US based, EasyBuild more European based. Broadly speaking. The 'downside' of Spack is its mix-and-match approach. Great for some things, maybe not so great for others like simple reproducibility.

C. The Spack developer gives numerous talks at the SC conferences, which are normally very well attended, and not many DOE system admins go to SC because of the DOE travel restrictions

Q. I've seen a few researchers consult with softwarecarpentry.org. Can you comment on that approach?

A. This was clarified in chat as being in reference to the training Software Carpentries offer. I'd strongly recommend using Software Carpentry as an intro to scientific programming - either attending a course with a qualified instructor, or self-learning from the materials online. They cover the basics well, and there's also a range of specialist topics are available in the Lesson Incubator - including the Intermediate lessons I've been involved in developing and delivering, that go into more depth on things like design patterns, programming paradigms and continuous integration!

Chat transcript (edited and anonymized, California times):

10:42:53 From participant-1 To Everyone:
This is a great talk. These are all ideas I have heard before but it is great to see them all collected and spun with a focus on the unique needs of research software. Sorry I have no questions, but the talk is great to hear.

10:44:46 From participant-2 To Everyone:
HA. Also I agree with participant-1 — I'm familiar with PEP8/Sphinx/pylint/etc, but it's really handy to have everything pulled together into one place. An endorsement for something like Flake8 from someone working in research software also goes a long way.