Collaborative Container Modules with Singularity Registry HPC

Vanessa Sochat[1] **and Alec Scott**[2]

1 Lawrence Livermore National Lab, Livermore, CA, USA **2** University of Arizona Research Computing, Tuscon, AZ, USA

DOI: 10.21105/joss.03311

Software

- Review
- Repository
- Archive

Editor: Patrick Diehl

Reviewers:

- @rcannood
- @zbeekman
- @ArangoGutierrez

Submitted: 24 April 2021
Published: 20 July 2021

Summary

Portability and reproducibility of complex software stacks is essential for researchers to perform their work. High Performance Computing (HPC) environments add another level of complexity, where possibly conflicting dependencies must co-exist. Although container technologies like Singularity (Kurtzer et al., 2017) make it possible to "bring your own environment," without any form of central strategy to manage containers, researchers that seek reproducibility via using containers are tasked with managing their own container collection, often not taking care to ensure that a particular digest or version is used. The reproducibility of the work is at risk, as they cannot easily install and use containers, nor can they share their software with others.

Singularity Registry HPC (shpc) is the first of its kind to provide an easy means for a researcher to add their research software for sharing and collaboration with other researchers to an existing collection of over 200 popular scientific libraries Gorgolewski et al. (2017). The software installs containers as environment modules (McLay et al., 2011) that are easy to use and read documentation for, and exposes aliases for commands in the container that the researcher can add to his or her pipeline without thinking about complex interactions with a container. The simple addition of an entry to the registry maintained by shpc comes down to adding a yaml file, and after doing this, another researcher can easily install the same software, down to the digest, to reproduce the original work.

Statement of Need

Using environment modules (McLay et al., 2011) on HPC clusters is a common trend. Although writing the recipes can be complex, it's a fairly common practice for cluster administrators to provide a set of natively installed recipes for their users (*An Introduction to Writing Modulefiles — Lmod 8.4.30 Documentation*, 2021), or for researchers to develop and deploy their own software via containers. Even well-known package managers like Spack (*Spack Modules*, 2015) and EasyBuild (*EasyBuild Modules*, 2021) expose software as modules. However, these package manager approaches don't always ensure reproducibility, or ease of development for the researcher. They typically require relying on some subset of system software, the underlying operating system, or even making changes to the system, which is not under the researcher's control. Although using containers in this context has been discussed previously (*Community Collections*, 2020; *Software Containers*, 2021), the majority of these approaches and tools do not make the process of developing and installing container modules easy. The single researcher must either convince a cluster administrator to install dependencies needed for their software, or build a container and manually move and interact with it on the cluster. All of these small challenges come together to make it harder for a researcher to develop and manage their own software, and subsequently to share their approach to reproduce the work. Using Singularity, Podman, or other container technologies installed via Singularity Registry HPC offers a solution to this challenge. The only requirement is the container technology software, and writing a simple configuration file for the registry. By clearly defining commands, and pinning exact versions of scientific software, researchers on high performance computing clusters can have more confidence in the reproducibility of their work (Boettiger, 2014; Santana-Perez & Pérez-Hernández, 2015; Wandell et al., 2015).

Usage

Installing shpc is as easy as cloning the repository and installing in place:

```
$ git clone https://github.com/singularityhub/singularity-hpc
$ cd singularity-hpc
$ pip install -e .
```

While the defaults are suitable for most, the researcher can customize the location of registry metadata files, the module directory to which modules are installed, and the directory to which containers are installed. The user can then use `shpc show` to see readily available recipes, or browse the library for an easily searchable interface. Installation comes down to installing a chosen module, loading it, and using it:

```
$ shpc install biocontainers/samtools
$ module load biocontainers/samtools
$ samtools
```

The samtools command would be an alias for a much more complicated command that the researcher would not need to remember, including global options, command options, the full path to the container, and the path to the executable inside the container. The container pulled would also be a specific digest, making it easier to reproduce the researcher's workflow. Finally, although containers typically only provide one entrypoint, there is no limit to the number of aliases that can be exposed for easy usage.

Collaborative Registry for Reproducible Science

Creating a registry entry for a scientific container comes down to writing a simple `container .yaml` file with basic metadata and description, definition any and all important entrypoints, and the digests to pull. As soon as a researcher puts their container in an online registry and adds the entry, new versions of the container are automatically discovered by shpc, and can be installed by the researcher when he or she chooses. The user does not need to look in advance for a version if they want the latest provided by the registry. Software is easy to search for, and quickly see complete documentation and commands available:

```
$ module spider samtools
```

Every module includes a help section with a description, a complete list of commands that map to interactions with the container, and a list of environment variables also available. The module system also provides command line completion, so the user can use tabs to autocomplete the module names. Along with the expected commands to use the container primary software (e.g., samtools) commands to shell, the software automatically generates alias to inspect, run, shell, or inspect container metadata. E.g., here is the shell command:

```
$ samtools-shell
```

Another compelling example is using a notebook provided by Jupyter Stacks (Cook, 2017). Running notebooks that can be exposed via networking ports tends to be very complicated. A full interaction might look like the following:

```
# pull the latest container, a moving target
$ singularity pull docker://jupyter/minimal-notebook

$ singularity exec --home ${HOME} --bind ${HOME}/.local:/home/joyvan/.local \
   minimal-notebook_latest.sif \
   jupyter notebook --no-browser --port=12345 --ip 0.0.0.0
```

With Singularity Registry HPC, the interaction to run the notebook can be figured out and written down once, and provided as a community recipe. In this case, it's exposed as the command "run-notebook":

```
$ run-notebook
```

which automatically selects a random port, binds the expected directories, and starts the notebook. The registry recipes are collaborative in nature because anyone can open a pull request with a new recipe, or request a container be added by opening an issue. Automation also ensures that adding and testing new containers, or working on the code base is easy. Once a container is added, no further work is needed to update versions for it. By way of a GitHub bot (Scott, 2021) both the latest version and newly available tags are updated automatically, following any filters that the recipe creator has provided for which tags should be added. Finally, on merge to the main branch, the documentation and library are also automatically updated.

Conclusion

Singularity Registry HPC is the first local container registry that supports reproducibility, easy of use, and portability of research software. You can read more about it on the GitHub repository (https://github.com/singularityhub/singularity-hpc) or the main documentation site (https://singularity-hpc.readthedocs.io).

References

An introduction to writing modulefiles — lmod 8.4.30 documentation. (2021). https://lmod.readthedocs.io/en/latest/015_writing_modules.html.

Boettiger, C. (2014). *An introduction to docker for reproducible research, with examples from the R environment*. https://doi.org/10.1145/2723872.2723882

Community collections. (2020). https://community-collections.github.io/.

Cook, J. (2017). The opinionated jupyter stacks. In *Docker for Data Science* (pp. 119–135). Springer. https://doi.org/10.1007/978-1-4842-3012-1_7

EasyBuild modules. (2021). https://wiki.fysik.dtu.dk/niflheim/EasyBuild_modules.

Gamblin, T., LeGendre, M., Collette, M. R., Lee, G. L., Moody, A., Supinski, B. R. de, & Futral, S. (2015). The spack package manager: Bringing order to HPC software chaos. *SC'15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 1–12.

Gorgolewski, K. J., Alfaro-Almagro, F., Auer, T., Bellec, P., Capotă, M., Chakravarty, M. M., Churchill, N. W., Cohen, A. L., Craddock, R. C., Devenyi, G. A., & others. (2017). BIDS apps: Improving ease of use, accessibility, and reproducibility of neuroimaging data analysis methods. *PLoS Computational Biology*, *13*(3), e1005209. https://doi.org/10.1371/journal.pcbi.1005209

Kurtzer, G. M., Sochat, V., & Bauer, M. W. (2017). Singularity: Scientific containers for mobility of compute. *PLoS One*, *12*(5), e0177459. https://doi.org/10.1371/journal.pone.0177459

McLay, R., Schulz, K. W., Barth, W. L., & Minyard, T. (2011). Best practices for the deployment and management of production HPC clusters. *State of the Practice Reports*, 1–11. https://doi.org/10.1145/2063348.2063360

NVIDIA NGC. (2018). https://ngc.nvidia.com/catalog.

Santana-Perez, I., & Pérez-Hernández, M. S. (2015). Towards reproducibility in scientific workflows: An Infrastructure-Based approach. *Sci. Program.*, *2015*. https://doi.org/10.1155/2015/243180

Scott, A. (2021). *Binoc*. Github; https://github.com/autamus/binoc.

Software containers. (2021). https://www.rc.virginia.edu/userinfo/rivanna/software/containers/.

Spack modules. (2015). https://spack.readthedocs.io/en/latest/module_file_support.html.

The autamus registry. (2021).

Veiga Leprevost, F. da, Grüning, B. A., Alves Aflitos, S., Röst, H. L., Uszkoreit, J., Barsnes, H., Vaudel, M., Moreno, P., Gatto, L., Weber, J., & others. (2017). BioContainers: An open-source and community-driven framework for software standardization. *Bioinformatics*, *33*(16), 2580–2582.

Wandell, B. A., Rokem, A., Perry, L. M., Schaefer, G., & Dougherty, R. F. (2015). *Data management to support reproducible research*. http://arxiv.org/abs/1502.06900