The Journal of Open Source Software

x11docker: Run GUI applications in Docker containers

Martin Viereck[1]

1 Martin Viereck, Germany

DOI: 10.21105/joss.00085

Software

- Review
- Repository
- Archive

Submitted: 28 September 2016
Published: 29 March 2019

Summary

`x11docker` allows to run graphical desktop applications in a GNU/Linux container using Docker.

About Containers in general

Containerisation in general has proven as a useful technology for packaging applications and their dependencies for deployment in cloud-based infrastructures. A container is similar in usage to a virtual machine, but needs less resources. The technical concept, however, is completely different.

The properties of containers such as portability, dependency packaging, minimal requirements of system environment (i.e. only the container runtime), isolation, and version management of complete application stacks make it a promising candidate to increase computational reproducibility and reusability of research analyses (Boettiger, 2015). Their use has been demonstrated in various disciplines, such as software engineering research (Cito & Gall, 2016), bioinformatics (Hosny, Vera-Licona, Laubenbacher, & Favre, 2016), and archeology (Marwick, 2017), and their preservation is an active field of research (Emsley & De Roure, 2018; Rechert et al., 2017).

Software and required libraries can be installed in a Docker image to run software that is difficult to install otherwise. It is possible to run outdated versions, specific versions, or latest development code side by side.

About x11docker

The most popular Linux container frontend, Docker, does not provide a display server that would allow to run applications with a graphical user interface (GUI), because Docker is originally built for server software. `x11docker` fills this gap.

`x11docker` allows to execute Desktop GUI applications in an isolated environment by running an X display server on the host system and providing it to applications in Docker containers.

`x11docker` simplifies container setup and access to host resources like shared files, GPU acceleration, audio, webcam and printer. Non-GUI applications can benefit from this, too.

Additionally, `x11docker` does some security setup to enhance container isolation from host system. It follows the principle of least privilege.

`x11docker` thereby facilitates quick creation, distribution, and evaluation of research prototypes without compromising on a researcher's skills (not imposing browser-based

GUI nor requiring command-line proficiency), domain (having e.g. established and widely-acknowledged GUI-based tools), security, computational reproducibility, or a scholarly review process.

The target audience of `x11docker` in general are users who want to run desktop applications in containers. Another target audience are developers of desktop applications who need an isolated environment. In scientific and academic context the target audience are researchers in the field of reproducible science.

`x11docker` has its own (optional) graphical frontend, `x11docker-gui`, and runs on GNU/Linux. Running in a Virtual Linux Machine on MS Windows and macOS is supported. With a few limitations it can run natively on MS Windows, too.

Alternatives to x11docker

A common way to allow GUI applications in containers is by providing a web server within the container and rendering an HTML-based GUI in a common web browser, e.g. as jupyter notebooks (Jupyter et al., 2018). Further possibilities are an xrdp server, VNC server, SSH server or xpra server within the container. These solutions require some specific setup and provide a rather slow interaction due to a lot of network data transfer.

References

Boettiger, C. (2015). An introduction to Docker for reproducible research, with examples from the R environment. *ACM SIGOPS Operating Systems Review*, *49*(1), 71–79. doi:10.1145/2723872.2723882

Cito, J., & Gall, H. C. (2016). Using Docker Containers to Improve Reproducibility in Software Engineering Research. In *Proceedings of the 38th International Conference on Software Engineering Companion*, ICSE '16 (pp. 906–907). ACM. doi:10.1145/2889160.2891057

Emsley, I., & De Roure, D. (2018). A Framework for the Preservation of a Docker Container International Journal of Digital Curation. *International Journal of Digital Curation*, *12*(2). doi:10.2218/ijdc.v12i2.509

Hosny, A., Vera-Licona, P., Laubenbacher, R., & Favre, T. (2016). AlgoRun: A Docker-based packaging system for platform-agnostic implemented algorithms. *Bioinformatics*, *32*(15), 2396–2398. doi:10.1093/bioinformatics/btw120

Jupyter, Bussonnier, Forde, Freeman, Granger, Head, Holdgraf, et al. (2018). Binder 2.0 - Reproducible, interactive, sharable environments for science at scale. In Fatih Akici, David Lippa, Dillon Niederhut, & M. Pacer (Eds.), *Proceedings of the 17th Python in Science Conference* (pp. 113–120). doi:10.25080/Majora-4af1f417-011

Marwick, B. (2017). Computational Reproducibility in Archaeological Research: Basic Principles and a Case Study of Their Implementation. *Journal of Archaeological Method and Theory*, *24*(2), 424–450. doi:10.1007/s10816-015-9272-9

Rechert, K., Liebetraut, T., Kombrink, S., Wehrle, D., Mocken, S., & Rohland, M. (2017). Preserving Containers. In J. Kratzke & V. Heuveline (Eds.), *Forschungsdaten managen* (pp. 143–151). Heidelberg. doi:10.11588/heibooks.285.377