Evaluation of lightweight virtualization using Docker in the context of ASTERICS to enhance reproducibility of analyses

Stefan Geißelsöder
ECAP, Erlangen

March 20, 2017

1 Motivation

As analyses in astronomy, astrophysics and particle astrophysics tend to increase in complexity with the abilities available on modern computer systems, the basic requirement of (short and long term) reproducibility is becoming harder to achieve. The high number of dependencies on other software packages, which have implicitly been used to obtain a result, are non-trivial to be reproduced exactly and sometimes not all dependencies are recognized explicitly.

Techniques like Docker can help not only to achieve this reproducibility, but also in situations where checks or changes are desired after the know-how is not fully available anymore, e.g. when the original creator has left a collaboration, or when a new person is to become involved in a workflow.

2 Considerations that led to Docker

Why do we recommend Docker instead of another technology for experiments in ASTERICS? Here are a few considerations of alternatives.

2.1 Plain scripts

Plainly executing custom scripts on a machine in its currently available state risks not to be able to reproduce the computation and results with updated hard/software. This does not constitute an acceptable solution.

2.2 Virtual machines

In principle virtual machines (VMs) are a reasonable idea to keep analyses reproducible. While there are different competing standards, most of the technologies are compatible with each other. Without additional care, VMs can decrease the efficiency of computations [RM15], but as the effect is not too big for our application (and can be reduced by certain VM techniques [xen17, RM15]),

this is not considered a major drawback here. The main problem with VMs for the envisioned application is that the memory requirement doesn't scale good enough for a large number of analyses, since a whole virtual machine has to be stored for each analysis.

2.3 Docker

Docker [doc17] is based on layered containerization. It allows direct access to the underlying kernel while abstracting the software dependencies. Since only the changes to a common base system required by an analysis have to be stored for an analysis, the main drawback of virtual machines in the envisioned application scenario is dealt with, namely the large memory requirements to store analyses. A drawback of the current status of Docker is the danger of "privilege escalation", which potentially allows users to perform actions on a machine they otherwise wouldn't have the rights to. This is no relevant concern for the application to physics analyses, but it prevents several clusters from supporting Docker directly. Desktop and cloud systems on the other hand support Docker.

2.4 Singularity

Singularity [sin17] is another recent option that could help with reproducibility similar to Docker. While it even deals with privilege escalations, so far it is not available on Microsoft Windows operating systems which are used by some members of some collaborations. Furthermore the already established user basis is not as large as Dockers and therefore the risk is higher that it might not be supported in the long run[1].

2.5 Nix

Nix [nix17] also constitutes a development that could be suited to help keeping analyses reproducible. However it focuses on an application and its dependencies. While it offers an elegant way to deal with these dependencies, it does not (and is not intended to) deal with e.g. input files. Furthermore it doesn't support Microsoft Windows and, judging from the current size of the Nix community, is also less widespread than Docker.

2.6 Others

There are many other, similar approaches (e.g. rkt [cor17], Flockport [flo17]), but on the one hand, there is no essential feature lacking in Docker that would be required for our use-case. On the other hand, to achieve a long-term reproducibility, which is key to us, a solution should be supported by a large community and as many other cooperations or companies as possible. The current status is that this is fulfilled best by Docker. Furthermore, many alternatives

[1] Future developments can't be predicted, but a large existing user base tends to be correlated with a longer support and compatibility of following technologies.

also support Docker images or Dockerfiles, while their extensions usually are incompatible to others.

3 Summary

Table 3 is a summary of the comparison with perceived most relevant disadvantages in red. While it should be clear at this point that there is no single perfect solution, Docker seems to be the best choice. It does solve the problem of reproducibility of physics analyses, is relatively easy to use [Gei17b] and doesn't have major drawbacks as far as the investigated use-case is concerned.

As this lightweight virtualization promises significant benefits, first efforts have already begun to employ Docker for KM3NeT [Gei17a]. The same concepts for packaging an analysis with Docker that are being worked out with KM3NeT (e.g. a common base image and a default input/output interface) can also be applied for other experiments.

	Plain	VMs	Docker	Nix	Singularity
Reproducible	Maybe not	Yes	Yes	Yes	Yes
All OS	Yes	Yes	Yes	No	No
Widespread*	Yes	Yes	Yes	No	No
Efficient	Yes	Possible	Yes	Yes	Yes
Storage required	Low	High	Moderate	Low	Moderate

*Community for different lightweight techniques on github.com, state as of 2017-03-07

	Commits	Contributors
Docker	31219	1627
Nix	2048	34
Singularity	5075	110
rkt	5167	177

References

[cor17] coreos.com. rkt A security-minded, standards-based container engine. https://coreos.com/rkt, 2017. 2017-03.

[doc17] docker.com. What is Docker. https://www.docker.com/what-docker, 2017. 2017-03.

[flo17] flockport.com. Instant apps for everyone. https://www.flockport.com/learn, 2017. 2017-03.

[Gei17a] Stefan Geißelsöder. dockerProjects. https://github.com/sgeisselsoeder/dockerProjects, 2017.

[Gei17b] Stefan Geißelsöder. Using Docker for your analysis in KM3NeT, 2017.

[nix17] nixos.org. Nix The Purely Functional Package Manager. https://nixos.org/nix, 2017. 2017-03.

[RM15] M. Komu R. Morabito, J. Kjallman. Hypervisors vs. Lightweight Virtualization: a Performance Comparison. *2015 IEEE International Conference on Cloud Engineering (IC2E)*, 2015.

[sin17] singularity.lbl.gov. About Singularity. http://singularity.lbl.gov/about, 2017. 2017-03.

[xen17] xenproject.org. Why Xen Project? https://www.xenproject.org/users/why-the-xen-project.html, 2017. 2017-03.