International Journal of Computer (IJC)
ISSN 2307-4523 (Print & Online)

http://ijcjournal.org/

Using Computing Containers and Continuous Integration to Improve Numerical Research Reproducibility

Alexey Vasyukov[a]*, Igor Petrov[b]

[a,b]*Moscow Institute of Physics and Technology, Institutsky lane 9, Dolgoprudny, 141701, Russia*
[a]*Email: a.vasyukov@phystech.edu*
[b]*Email: petrov@mipt.ru*

Abstract

Cloud computing has opened new options of collaboration between research teams in the field of high performance computing and numerical research. Running computational workloads in virtual machines became common in recent years. However, the use of computing containers provides many additional advantages besides just proving new possible runtime choice. One of the most important (and often underappreciated) is an option to improve the reproducibility of research results based on complex mathematical modeling. This paper provides an overview of architecture based on computing containers and continuous integration tools we used to achieve reproducible numerical results.

Keywords: linux containers; docker; cloud computing; continuous integration; research reproducibility.

1. Computing containers

Historically cloud computing was associated with virtual machines. So, in the field of high performance computing initial move towards cloud was also based on virtual computing environments. KVM and OpenStack can be considered as de facto standard at the moment for building a cloud infrastructure for scientific applications. Such environments allow the user to run virtual machines with preconfigured software packages, as well as create computational clusters from such machines [1]. However, overhead when running applications in virtual machines can be quite high, and forming a single image for different virtualization environments is quite a challenge [2]. In connection with this, the use of container virtualization technologies within the same model is actively developing [3].

---

* Corresponding author.

1.1. Constraints and limitations

Computing containers have some limitations, and it's worth providing a brief summary here. The major constraints related to numerical research are limited availability on Windows and restrictions related to distributed applications performance. The first limitation is that containers were developed initially as a part of Linux ecosystem, and containers support on Windows is still under active developments. It means, the approach discussed in this paper may be hard to implement for certain applications that are tightly bound to Windows ecosystem. However, recent versions of Windows Servers support Docker containers, so we expect that the situation will become much better in the nearest future. The second limitation is related to distributed applications that rely on specialized hardware – high speed network, GPU accelerators and so on. Using such hardware in containers is possible, but requires specific tuning for each new environment.

1.2. Achieving reproducibility of numerical results

However, containers also provide a lot of new promising features for the field of numerical research. Several published articles have already mentioned that containers can contribute to the major goal of achieving complete reproducibility of scientific results in this field [4, 5].

One of the traditional difficulties when using multiple complex computing packages to solve a coupled problem involving several areas of physics is their heterogeneous and often conflicting requirements for the assembly tools and the computing environment. As a result, installation of almost any complex software package on a cluster is a cumbersome and error prone procedure. It is necessary to pre-compile a large number of libraries of specific versions from the source code. Authors of this paper had an experience of dealing with the situation when, in order to install the application, it was first required to compile a new version of the compiler from the sources using the older compiler on the cluster.

Similar cases cause a lot of problems with research reproducibility – formally the same sources of numerical software built and run in different computing environments may show significantly different results when solving the same numerical problems. So, after successful build and test of complete software stack on local machine (see Figure 1) it is quite a complex and deterministic task to move this stack to high performance computing cluster. Building and running on a new system involves different assembly tools and different versions of system libraries (Figure 2). So, there is no certainty that the resulted binary really matches tested one.

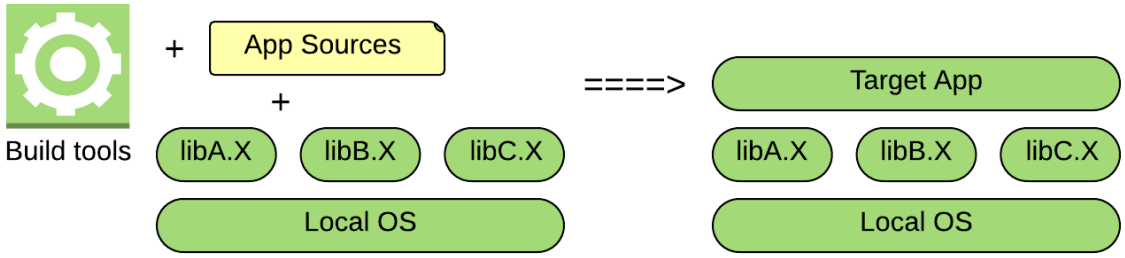

Figure 1. Building on local machine

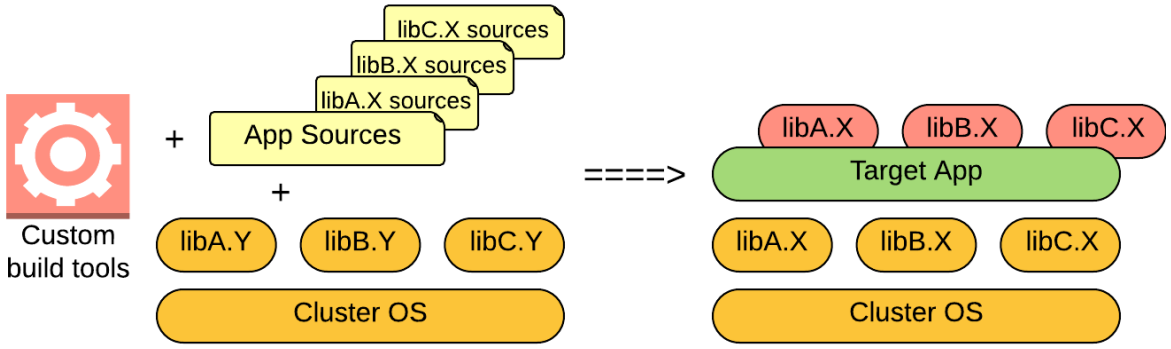

Figure 2: Building on remote cluster

The authors adopted Docker containers to solve this problem. The numerical applications were compiled from the sources and linked with system libraries in the most convenient environment (CentOS 7, Fedora 22, Ubuntu 15.10 - depending on the application). At the next step a computational Docker container was created with the application binary and the software environment required for its operation, including specific versions of system libraries and necessary environment variables (Figure 3). So, the container becomes a reference ready-to-use environment, the result of the build procedure is the solver plus the entire runtime.

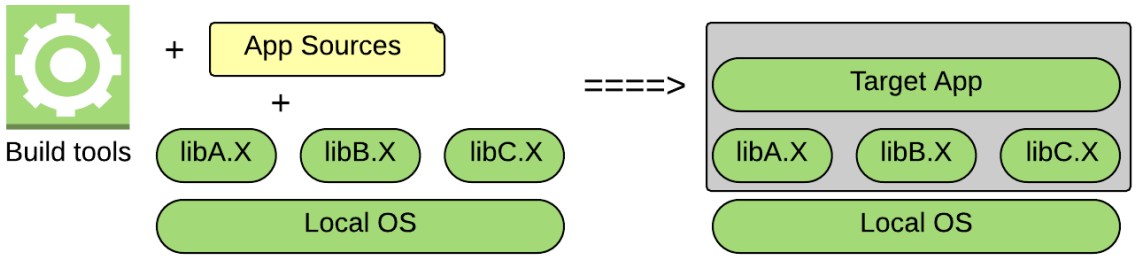

Figure 3: Building a container of local machine

After that the containers were transferred to a high performance cluster (based on CentOS 7) and launched in a cluster environment without the need to perform any build operations (Figure 4).

Without the use of containers, the build and run procedures for the cluster would be a nontrivial task that requires a high level system programmer. Thus, the use of containers made it possible to separate the environments for the assembly and execution of applications and to provide the user with the opportunity to quickly prepare and launch their own calculation modules without requiring either system programmer skills or cluster administrator access rights.

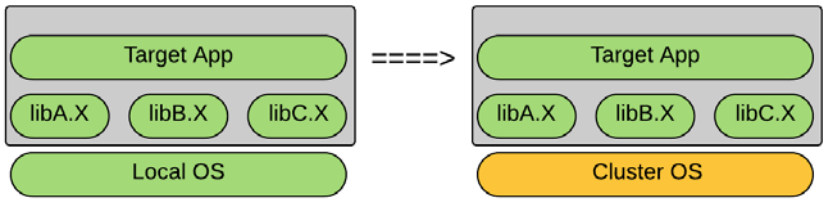

Figure 4: Running container on a remote cluster

1.3. Deployment options

We used two deployment options during our research – bare metal deployment (Figure 5) and KVM deployment (Figure 6).

In our previous work [6] we tested the performance of computational packages when working in containers. The main tests were performed with OpenFOAM and GCM-3D solvers. Due to the architecture of container virtualization there was no RAM overhead compared with bare metal deployment. Measurements of CPU time for solving model problems also showed quite a low level of overhead, in all measurements the performance degradation relative to the launch on physical equipment did not exceed 1-2%. So, running containers on bare metal looks very promising, and the architecture shown on Figure 5 looks like a natural choice.

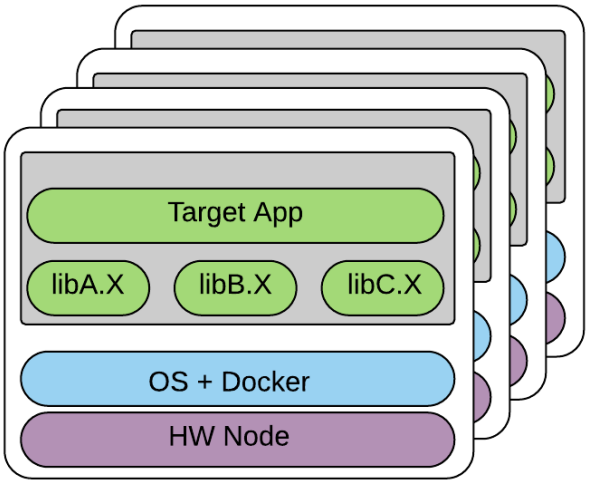

Figure 5: Bare metal deployment

However, running containers requires some level of support from operating system side. It is not that complex, but existing large scale computing clusters mostly were not build for containers. On the other hand, as mentioned above, many clusters have already adopted KVM based virtualization. So, using the architecture shown on Figure 6 may be a good compromise when running on such systems.

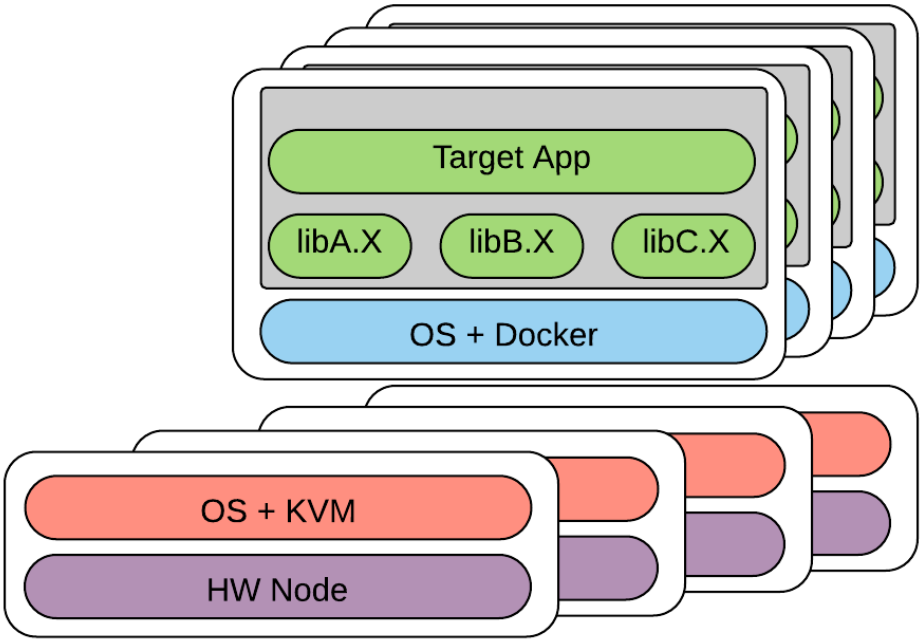

Figure 6: KVM deployment

2. Continuous integration

Containers may contribute the most to the reproducibility of numerical research when they are integrated directly into the development of numerical solvers. This approach was tested with GCM-3D calculation module. The source texts of GCM-3D are stored in Github repository. The repository infrastructure was configured in such a way that when the developer makes a change to the source code, the binary modules are automatically build and the container containing GCM-3D and the required software environment is updated. This container is used later as a golden image to run GCM-3D solver in all environments, from local machines of developers to high performance clusters. The workflow used while developing, building and running the solver is shown on the Figure 7.

The containers with GCM-3D are also used to automatically test the correctness of the application after the changes in solver sources. Cloud continuous integration service Travis is used for testing. An essential nuance of testing scientific applications is the resource capacity of tests - to verify the correctness it is required to perform calculations of a large number of test cases for which there are analytical solutions or physical experimental data, and then to compare the resulting numerical solution with a reference one. For a reasonable coverage of the functional tests of the application package, it is required to conduct tests for a lot of test cases

with different input parameters, so complete test package run may take several hours or even few days. The use of cloud services, such as Travis CI, makes it much easier to achieve continuous integration by offloading basic test runs to the cloud

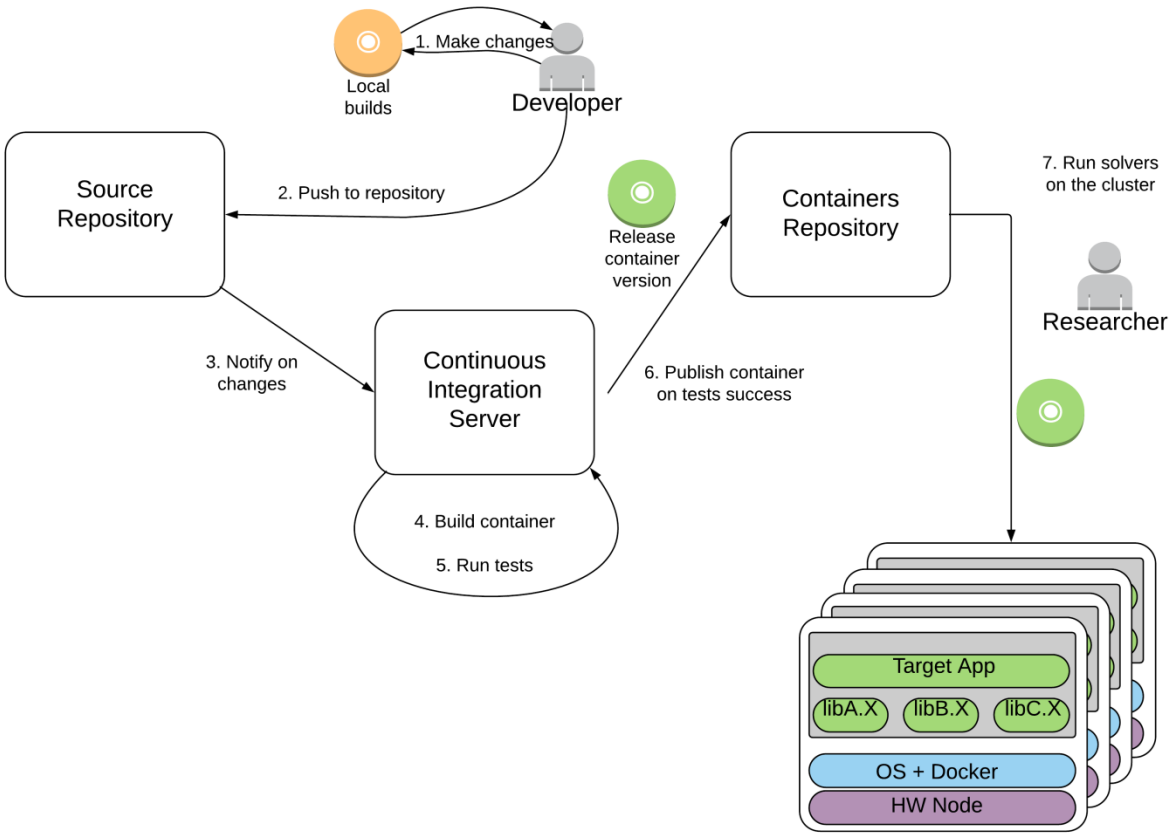

Figure 7: GCM-3D workflow

3. Conclusion

Using computing containers for scientific applications provides a lot of new tools to achieve research reproducibility. If containers are integrated into the development workflow of the solver, then the result of the build is not just a binary, but a binary with the complete computing environment (system libraries of certain versions, environment variables). This container can be transferred to any system and provide the same calculation results.

4. Recommendations

Taking into account all the benefits that computing containers provide for the reproducibility of the numerical research, it's worth recommending publishing computing container images as a part of research supplementary materials. This will allow independent researchers from other laboratories to reproduce author's results reliable, and this will contribute a lot to the trust into mathematical modeling methods in general.

Acknowledgments

The research was supported by Russian Foundation for Basic Research grant 15-29-07096.

References

[1]. U. Markwardt. “Running Virtual Machines in a Slurm Batch System”, presented at Slurm User Group Meeting, Washington DC, USA, 2015. [On-line]. Available: http://slurm.schedmd.com/SLUG15/SlurmVM.pdf [June 01, 2018].

[2]. W. Felter, A. Ferreira, R. Rajamony, J. Rubio. "IBM Research Report. Performance Comparison of Virtual Machines and Linux Containers." [On-line]. Available: http://domino.research.ibm.com/library/cyberdig.nsf/papers/0929052195DD819C85257D2300681E7B/$File/rc25482.pdf [June 01, 2018].

[3]. C. Kniep. "Containerization of High Performance Compute Workloads using Docker." [On-line]. Available: http://doc.qnib.org/2014-11-05_Whitepaper_Docker-MPI-workload.pdf [June 01, 2018].

[4] J. Cito, V. Ferme, H.C. Gall. "Using Docker Containers to Improve Reproducibility in Software and Web Engineering Research." in Web Engineering. ICWE 2016. Lecture Notes in Computer Science, vol. 9671. A. Bozzon, P. Cudre-Maroux, C. Pautasso (eds). Springer, 2016.

[5] L.H. Hung, D. Kristiyanto, S.B. Lee, K.Y. Yeung. "GUIdock: Using Docker containers with a common graphics user interface to address the reproducibility of research." PLoS One, 11(4):e0152686, 2016.

[6] A. Ermakov, A. Vasyukov “Testing Docker Performance for HPC Applications” IOSR Journal of Computer Engineering, vol. 20, pp. 36-43, 2018.