

Dockerization (Replacement of VMs)

Lakshay Swani¹, Prakita Tyagi²

¹ Software Engineer, Globallogic Pvt. Ltd, Sector-144, Noida, UP, INDIA

² Software Engineer, Infosys, Bengaluru, INDIA

Abstract – With the present trend in the IT industry, the computational work is becoming more and more complex and integral to various scientific and mathematical researches, there arises a need for reproducibility of extensive computational systems. Though it seems possibly easy to replicate the computational systems as is done with physical systems, but the rapid change and evolution the software industry go through, replicating the computational systems is a serious challenge. Through this paper, we provide a scenarios where the physical replication becomes unviable and a need for computational replication comes up. The standard approaches to the problems raised in the above scenarios fail to provide solution and thus to tackle those issues, we explore the option of an emerging technology, Docker, that provides us with computational replication by taking in consideration through the aspects of versioning, virtualization, portability and modularity.

Key Words: Computational Replication, Containers, Docker, Docker Hub, Ubuntu

1. INTRODUCTION

Computation is the base of all the mathematical and scientific researches [5]. At each step of research, from the collection of data to analysis, conceptualization and visualization, computational work is involved [24]. Thus the need for replication of computation has been a top priority to all the researchers [17, 18]. As the complexity involved in analysis & processing of data increased in the past few years, the computational replication has gained attention [17]. Reproducible research has received an increasing level of attention throughout the scientific community [18, 21] and the public at large [24].

1.1 The Problem

It is worth observing from the outset that the primary barrier to computational reproducibility in many domain sciences has nothing to do with the technological approaches discussed here, but stems rather from a reluctance to publish the code used in generating the results in the first place [2]. To process the ever increasing demand of computation, multiple processors with extensive hardware is required with proper support to handle the changes. Multiple processor came into play, Virtual Machines took the load for providing extensive computational capabilities, high-end

computers were designed which when interconnected within a network could perform the computational task effectively. As easy it seemed, the more difficult was the task to achieve the viable computational capabilities [6]. The demands were ever increasing for higher computational replication which was to be fulfilled through the above discussed technologies. But the problem that was faced was to how much extensibility is to be provided [23]. The physical replication or the replication through virtual machines was limited to an extent beyond which, being able to manage those hardware equipment wouldn't be viable for any researcher. With the increased hardware equipment, the increased resource utilization such as RAM, Hard Disks and others were to increase significantly. Also with the changing trend of the technologies, being able to manage the vast number of equipment or VMs and to update them as per the requirement was hectic and proved to be time consuming and complex. Thus to eradicate all these issues, Docker came into play.

The problems that were being tackled can be categorized into 4 technical challenges- Dependency Issues, Software dynamicity, Limited Documentation & Barriers to adoption [3, 7, 9, 15 and 26].

Dependency Issues

To account for the computational replication, multiple solutions have been provided. But for a particular code to be run over the provided solution to provide the computational replication, various dependencies of the code being run needs to be resolved [13]. The dependencies involve installation and configuration of various tools required to execute the piece of code. Taking the case of VMs to solve the issue of computational complexity, multiple VMs running the particular piece of code needs to be refreshed and installed with the required tools and software which therein needs to be configured as per the requirements [11]. This issue occurring due to the overhead involved in deploying and configuring the dependencies [3], a well-structured solution to computational replication was required [4].

Software Dynamicity

The code being deployed is non-static, i.e. it is updated frequently. The updating could be possibly due to inclusion of new features or solving of the previously occurring bugs. To be able to solve this issue through the traditional

approach of VMs, each VM will have to be updated and reconfigured to support the new code so that the newly written code becomes compatible with the configured VM [16]. This generally is an overhead and required careful examination and work-force to implement any change of code that too frequently [19].

Limited Documentation

With changing workforce within the organization, deploying the pre-requisites and configuration of VMs through the provided documentation could be limited [15]. For a novice, it becomes a barrier understanding the documentation [7]. Also there is a barrier in understanding and executing the documentation provided for executing the piece of code. Since the deployment and configuration is a critical part, any complications during this stage could prove to be harmful for the organization [9]. Thus limited documentation is also one of the challenges being faced.

Barriers to adoption

The trivial solutions provided for computational replication such as Virtual Machines, integration services and others involve learning and becoming apt to the technology is one of the main concern. The researchers face the barrier to learn and understand the tools and approaches that will further work towards achieving computational replication [14].

1.2 The Solution

Docker is a new, yet an excellent piece of technology that can facilitate the computational replicability and has been identified by many [8]. A hypervisor is a software, hardware or firmware over the top of which, the VMs run. The hypervisors are themselves ran on physical machines, which are referred to as the “host machines”. The host machine is responsible for providing the VMs with hardware resources such as RAM and CPU [22]. These resources provided to the host machine are then divided between VMs running over the host machine and can be distributed as it see fit [27]. So if one of the VM executing on the host machine is running a heavy application, the host machine allocates more resources to that particular VM than the other VMs running on that host machine [11]. Docker is similar to a hypervisor in the terms that it supports building of containers on the top of the operating system Docker is running over. Docker, based on Linux Containers, is an open-source project. It uses Linux Kernel features like namespaces and control groups to create containers on top of an operating system. Docker brings to the table a whole new set of capabilities that the other technologies couldn't bring. They include Ease of Use, speed, Docker Hub, Modularity and Scalability.

DOCKER

Docker is an open source project that builds on many long familiar technologies from operating systems research: LXC containers, virtualization of the OS, and a hash-based or Git-like versioning and differencing system, among others. Docker automates the repetitive tasks of setting up and configuring development environments so that developers can focus on what matters: building great software. When an app is dockerized, that complexity is pushed into containers that are easily built, shared and run. Onboarding a co-worker to a new codebase no longer means hours spent installing software and explaining setup procedures. Code that ships with Docker files is simpler to work on: Dependencies are pulled as neatly packaged Docker images and anyone with Docker and an editor installed can build and debug the app in minutes.

Docker is capable of overcoming the technical problems that the earlier technical solutions provided to computational replication.

Docker is capable of handling the dependency issue very well. Instead of installation and configuration of required tools over the containers is done manually, the installations and configurations need to be done once and then the Docker has the capabilities to create a binary file that contains the requirements and configuration over which the code needs to be run.

This binary file is referred to as “Docker Images”. The key difference between the Docker images to the other Virtual Machine images is that the Docker image provides along with itself, the Linux kernel instead of the complete virtual machine host. This enables the system to run the running instance of Docker container with ease and perfection. Since the kernel being present instead of the whole VM, one can on a typical computer can run over a 100 Docker container instances.

As discussed earlier, the code gets refreshed and updated frequently which could be due to any bug fix or annual maintenance or new feature that needs to add to the code. The challenge raised through this can be significantly reduced through the use of Docker has Docker images defines the environment to a particular operating system.

The Docker has the ease of use capabilities through which the images that need to be created and lets the user create the Docker image easily and interactively.

2. INSTALLATION & DEPLOYMENT

Docker is portable and has the ability to run over various OS present out there. Now we will go through a step by step procedure for installation and configuration of Docker onto a commonly used OS – Ubuntu. Before we proceed, we should

be aware of the common terminology being used in Docker. It involves a bit of knowledge around Docker Images, Docker Container & Docker Registry.

Docker Images

A Docker image is basically a template that is used over Docker containers. This image includes data and configuration of the various tools that are installed over the OS required and also contains the OS. This image is used to run the container. Docker has provided us with pre-built images of various OS and basic software that have been installed on to them. These images are ready to deploy images which are deployed over the container in Docker.

Docker Container

Docker Container is itself a type of image that can be read and written to that runs on top of the Docker image available to the Docker. So as to provide the functionality of versioning, Docker uses the union-file system as a backend for the Docker Container. This allows the Docker to save any changes made to the container as a new layer above the original image hence maintaining the versioning of the Docker images and container. The container is the referred layer over which different applications are installed and each Docker container is run independent of each other such as a VM, thus providing a secure application platform.

Docker Registry

The Docker registry is basically a repository for Docker images. Docker provides us with private and public repositories. The public repository is referred to as Docker Hub where the Docker images can be pushed and pulled and Docker provides us with a mechanism to automate the building and deployment process of these Docker images.

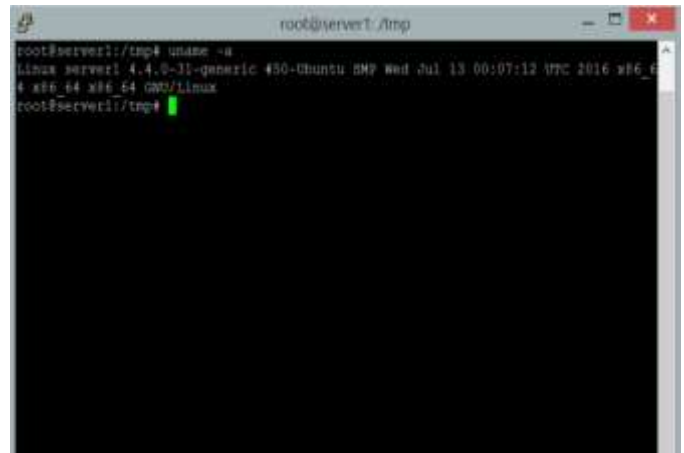
2.1 Installation of Docker

To be able to install the Docker software on Ubuntu, we need to run the commands as a root user, hence we need to run the following command on the Ubuntu System.

`sudo -s`

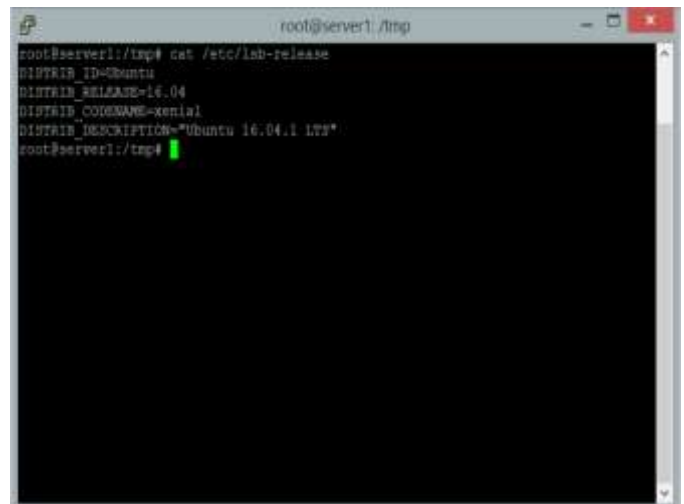
Check the version of the currently running Kernel version.

`uname -a`



```
root@server1:/tmp# uname -a
Linux server1 4.4.0-31-generic #50-Ubuntu SMP Wed Jul 13 00:07:12 UTC 2016 x86_64
x86_64 x86_64 GNU/Linux
root@server1:/tmp#
```

The image shows the presence of x86_64 bit OS being run. Check the version of the Ubuntu installed over the system.
`cat /etc/lsb-release`



```
root@server1:/tmp# cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=16.04
DISTRIB_CODENAME=xenial
DISTRIB_DESCRIPTION='Ubuntu 16.04.1 LTS'
root@server1:/tmp#
```

The image depicts the version of Ubuntu on which the Docker is going to be installed.

Fetch the latest updates of Ubuntu before installing Docker on the system.

`apt-get update`

`apt-get upgrade`

Install Docker with the following apt command.

`apt-get install -y docker.io`

Once the installation is completed, proceed with starting the docker using the systemctl command

`systemctl start docker`

To enable the Docker to be run each time the system boot, run the following command

`systemctl enable docker`

To check the version of Docker installed on the system, run the following command:

`docker version`

```
root@server1:/tmp# docker version
Client:
Version:      1.10.3
API version:  1.22
Go version:   gol.1.1
Git commit:   20f81dd
Built:        Wed, 20 Apr 2016 14:19:16 -0700
OS/Arch:      linux/amd64

Server:
Version:      1.10.3
API version:  1.22
Go version:   gol.1.1
Git commit:   20f81dd
Built:        Wed, 20 Apr 2016 14:19:16 -0700
OS/Arch:      linux/amd64
```

The image shows that Docker is now installed on the Ubuntu System. The next step involves creation of container using either of the pre-built Docker images provided by Docker in the Docker public registry i.e. the Docker hub.

2.2 Deployment

Docker provides in the Docker hub, pre-built Docker images of various OS. We can pull the Docker image from the Docker hub and have it installed on the Docker.

To search for a base image for an OS from the Docker hub, run the following command:

```
docker search Ubuntu
```

```
root@server1:/tmp# docker search ubuntu
NAME                DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
ubuntu              Ubuntu is a Debian-based Linux operating s... 4377      [OK]
ubuntu-upstart      Upstart is an event-based replacement for ... 83        [OK]
kubernetes/ubuntu-... Dockerized k8s exec:iv, built on top of of... 39        [OK]
kubernetes/ubuntu-... Always updated official Ubuntu docker ima... 34        [OK]
ubuntu:dockerfile   Dockerfile --variant:ubuntu --component... 23        [OK]
s1d1c3r/ubuntu-lamp LAMP server on Ubuntu                          8         [OK]
m3ap3w/ubuntu       Simple always updated Ubuntu docker image... 8         [OK]
s1d1c3r/ubuntu-lamp-wordpress LAMP on Ubuntu with wp-cli installed         8         [OK]
k8s/ubuntu          This is a docker image of the official kube... 5         [OK]
k8s/ubuntu         Docker base image built on Ubuntu with apt... 2         [OK]
s1d1c3r/ubuntu     Base ubuntu image based on the official s... 1         [OK]
d4rkh0w/ubuntu     Base Ubuntu image -- updated hourly         1         [OK]
jindi/ubuntu       Ubuntu base image                             1         [OK]
rootpr/ubuntu      https://github.com/rootpr/docker-ubuntu     0         [OK]
kubernetes/ubuntu custom flavor of the official ubuntu base... 0         [OK]
s1d1c3r/ubuntu     Ubuntu is a Debian-based Linux operating s... 0         [OK]
cyp0r/ubuntu       Ubuntu 17.04                                  0         [OK]
w1d3r1an/ubuntu   Our basic Ubuntu images                       0         [OK]
mar1n3ry/ubuntu   Ubuntu with mariadb                          0         [OK]
c1r0v1n1/ubuntu   Customized Ubuntu                            0         [OK]
w1d3r1an/ubuntu   Docker images for ubuntu                     0         [OK]
t3am00c/ubuntu    Team00c's Ubuntu image configured with M... 0         [OK]
c1t101a/ubuntu    ubuntu image for docker with npm mirror    0         [OK]
k3m3r1n3s/ubuntu  Ubuntu base image                           0         [OK]
s1r4p1n0/ubuntu   Ubuntu image                                  0         [OK]
```

The above command will provide us with all the Ubuntu images present on the Docker Hub. The next step includes downloading the Docker image from the Docker Hub to your Ubuntu system.

```
docker pull Ubuntu
```

This command will download the image file from the server to your Docker registry/Docker Hub.

```
root@server1:/tmp# docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
433b9d0dcb30: Pull complete
2dc64e8f8d4f: Pull complete
670a593e1b50: Pull complete
183b0bf0d10e: Pull complete
Digest: sha256:c6674c44c6439673bf56536c1a15916639c47ea04c3d6296c5df938add67b54b
Status: Downloaded newer image for ubuntu:latest
```

We can see all the downloaded Docker images on our Docker registry through the following command:

```
root@server1:/tmp# docker images
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
ubuntu              latest      43119e3df429     11 days ago     124.8 MB
root@server1:/tmp#
```

In the next step, we have to create a container to run an instance of Docker image that has been downloaded to the Docker registry.

```
docker create ubuntu:16.04
```

```
root@server1:/tmp# docker create ubuntu:16.04
Unable to find image 'ubuntu:16.04' locally
16.04: Pulling from library/ubuntu
Digest: sha256:c6674c44c6439673bf56536c1a15916639c47ea04c3d6296c5df938add67b54b
Status: Downloaded newer image for ubuntu:16.04
d5df265ef0cf161d8f43335686942f1c2c800289afa5a85f9104dbf707cd8873
```

Once the Docker container has been created, it is time to run the instance that has been created. To run the Docker container created, execute the command:

```
docker run -i -t ubuntu:16.04 /bin/bash
```

```
root@server1:/tmp# docker run -i -t ubuntu:16.04 /bin/bash
root@8f64bd2b0547:/# exit
exit
root@server1:/tmp#
```

The above command will run the container created and will automatically redirect us directly into the container that is created and run the command bin/bash/ inside the container. To view all the containers running in the background, we can execute the following command:

```
docker ps
```

To stop a running container, we need to execute the stop command

```
docker stop Name/ContainerId
```

To remove a created container from the Docker, execute the rm command

```
docker rm Name/ContainerId
```

3. FEATURES

In response to the requirement of computational replicability, Docker provides a full-fledged solution to all the problems that arose in the trivial solutions and provides us with extensive features that are required throughout the desired computational replication. We highlight few of the features below.

Development over Local Environment

This is one of the most important feature that is provided by Docker. The trivial solutions to computational reproducibility did not provide the space to learn and fit in a locally available system and be integrated seamlessly into the existing workflow patterns. Docker, being a new and unfamiliar tool is immune from the above specified problem as it can seamlessly integrate into the existing workflow and can be learned and developed locally providing the researchers with a tool that can be easily adopted.

Effective Configuration

This is a key feature of Docker that provides us with effective configuration of the system easy and quick. Our code can be deployed in lesser time requiring low effort. Since Docker can be used in a wide variety of environments, the infrastructure requirements are no longer managed and linked with the environment the application is being run on [20].

Enhanced productivity

Through ease in the technical configuration and rapid deployment of application on the Docker, there is no doubt in the increased productivity of the code that is being deployed. Docker not only provides us with functionality to execute the application in an isolated environment, but also it has reduced the resources required for the deployment of the code.

Application Isolation

Docker provides us with containers that are used to run applications in isolated environments. Each of the container is independent to one another and allows us to execute any kind of application over a number of isolated containers with different configuration parameters.

Services

Services provide us a list of tasks that lets us specify the state of container inside the cluster. Each of the task represents one instance of a container that should be running and Swarm schedules them across nodes.

Swarm

It is a scheduling and clustering tool for Docker containers. Swarm uses the Docker API as its front end, which provides us with a way to use various tools to control it. It also helps in controlling a cluster of Docker hosts as a single virtual host. It is a self-organized group of engines used to enable pluggable backend.

4. CONCLUSIONS

Docker is an open source container virtualization platform which helps developers to deploy their applications and system administrators to manage applications in a safe virtual container environment. Docker runs on the Intel / AMD 64-bit architecture and the kernel should be higher 3.10 version. With Docker, you can build and run your application inside a container and then move your containers to other machines running Docker without any worries.

REFERENCES

- [1] Altintas, I. et al. 2004. Kepler: an extensible system for design and execution of scientific workflows. Proceedings. 16th international conference on scientific and statistical database management, 2004. (2004).
- [2] Barnes, N. 2010. Publish your computer code: it is good enough. *Nature*. 467, 7317 (Oct. 2010), 753–753.
- [3] Collberg, C. et al. 2014. Measuring Reproducibility in Computer Systems Research.
- [4] Dudley, J.T. and Butte, A.J. 2010. In silicon research in the era of cloud computing. *Nat Biotechnol*. 28, 11 (Nov. 2010), 1181–1185.
- [5] Eide, E. 2010. Toward Replayable Research in Networking and Systems. Archive '10, the nSF workshop on archiving experiments to raise scientific standards (2010).
- [6] FitzJohn, R. et al. 2014. Reproducible research is still a challenge. <http://ropensci.org/blog/2014/06/09/reproducibility/>.
- [7] Garijo, D. et al. 2013. Quantifying reproducibility in computational biology: The case of the tuberculosis drugome. *{PLoS} {ONE}*. 8, 11 (Nov. 2013), e80278.
- [8] Gil, Y. et al. 2007. Examining the challenges of scientific workflows. *Computer*. 40, 12 (2007), 24–32.
- [9] Gilbert, K.J. et al. 2012. Recommendations for utilizing and reporting population genetic analyses: the reproducibility of genetic clustering using the program structure. *Mol Ecol*. 21, 20 (Sep. 2012), 4925–4930.
- [10] Harji, A.S. et al. 2013. Our Troubles with Linux Kernel Upgrades and Why You Should Care. *ACM SIGOPS Operating Systems Review*. 47, 2 (2013), 66–72.
- [11] Howe, B. 2012. Virtual appliances, cloud computing, and reproducible research. *Computing in Science & Engineering*. 14, 4 (Jul. 2012), 36–41.

- [12]Hull, D. et al. 2006. Taverna: a tool for building and running workflows of services. *Nucleic Acids Research*. 34, Web Server (Jul. 2006), W729–W732.
- [13]Ince, D.C. et al. 2012. The case for open computer programs. *Nature*. 482, 7386 (Feb. 2012), 485–488.
- [14]Joppa, L.N. et al. 2013. Troubling Trends in Scientific Software Use. *Science (New York, N.Y.)*. 340, 6134 (May 2013), 814–815.
- [15]Lapp, Hilmar 2014. Reproducibility / repeatability bigThink (with tweets) @hlapp. Storify. <http://storify.com/hlapp/reproducibility-repeatability-bigthink>.
- [16]Leisch, F. 2002. Sweave: Dynamic Generation of Statistical Reports Using Literate Data Analysis. *Compstat. W. Härdle and B. Rönz, eds. Physica-Verlag HD*.
- [17]Merali, Z. 2010. Computational science: ...Error. *Nature*. 467, 7317 (Oct. 2010), 775–777.
- [18]Nature Editors 2012. Must try harder. *Nature*. 483, 7391 (Mar. 2012), 509–509.
- [19]Ooms, J. 2013. Possible directions for improving dependency versioning in r. *arXiv.org*. <http://arxiv.org/abs/1303.2140v2>.
- [20]Ooms, J. 2014. The openCPU system: Towards a universal interface for scientific computing through separation of concerns. *arXiv.org*. <http://arxiv.org/abs/1406.4806>.
- [21]Peng, R.D. 2011. Reproducible research in computational science. *Science*. 334, 6060 (Dec. 2011), 1226–1227.
- [22]Stodden, V. 2010. The scientific method in practice: Reproducibility in the computational sciences. *SSRN Journal*. (2010).
- [23]Stodden, V. et al. 2013. Setting the Default to Reproducible. (2013), 1–19.
- [24]The Economist 2013. How science goes wrong. *The Economist*. <http://www.economist.com/news/leaders/21588069-scientific-research-has-changed-world-now-it-need-change-itself-how-science->
- [25]Xie, Y. 2013. *Dynamic documents with R and knitr*. Chapman; Hall/CRC.
- [26]2014. Examining reproducibility in computer science. <http://cs.brown.edu/~sk/Memos/ExaminingReproducibility/>.
- [27]2012. Mick Watson on Twitter: @ewanbirney @pathogenomenick @ctitusbrown you can't install an image for every pipeline you want... <https://twitter.com/BioMickWatson/status/265037994526928896>.