IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-5, May 2019 E-ISSN: 2347-2693

Continuous Integration and end-to-end Automation Framework Deployment using Docker

Sanskruti Shrawane[1*], Medha Shah[2]

[1,2]Computer Science and Engineering, Walchand College of Engineering, Sangli, India

Corresponding Author: sanskruti.shrawane@gmail.com

DOI: https://doi.org/10.26438/ijcse/v7i5.521525 | Available online at: www.ijcseonline.org

Accepted: 18/May/2019, Published: 31/May/2019

***Abstract*—** Docker is a platform for building, deploying and managing various application containers, which are virtualized, on a common operating system. Docker has been widely adopted in every sector. Also, it can be used for the distribution and management of Docker images. Images are nothing but the iso images ie. images of operating systems and enterprise applications. Docker registry is a container which can be used to store images of operating system and enterprise applications. This paper proposes building a private centralized registry for employees of any organization. It includes generating Docker images for Linux platforms and automating image creation with Dockerfiles that will work across platforms. The paper also proposes the development of effective search logic for searching Docker images to get faster and easier outcomes. Also integrating this framework with the existing framework of any organization to improve their work efficiency.

***Keywords*—**Docker, Docker images, Dockerfile, Docker registry, Continuous integration.

I. INTRODUCTION

The Docker is basically a computer program, which performs containerization i.e. operating system level virtualization. The first release of docker came out in 2013 and is developed by Docker, Inc. Docker is an open platform which is used to develop, ship and run any application. Docker allows its users to separate their applications from their infrastructure, so software delivery becomes faster. The working of Docker depends on containers. The containers are grouped together, considering their own tools, libraries, configuration files, etc. Properly defined channels are used as media for their internal communication. All these containers can be run by a single operating system kernel and these are more lightweight than virtual machines. Images are created by the combination and modification of different images from repositories. Fig. 1 shows an overview of the architecture of Docker.

The Docker is basically a computer program, which performs containerization i.e. operating system level virtualization. The first release of docker came out in 2013 and is developed by Docker, Inc. Docker is an open platform which is used to develop, ship and run any application. Docker allows its users to separate their applications from their infrastructure, so software delivery becomes faster. The working of Docker depends on containers. The containers are grouped together, considering their own tools, libraries, configuration files, etc. Properly defined channels are used as media for their internal communication. All these containers can be run by a single operating system kernel and these are more lightweight than virtual machines. Images are created by the combination and modification of different images from repositories. Fig. 1 shows an overview of the architecture of Docker.

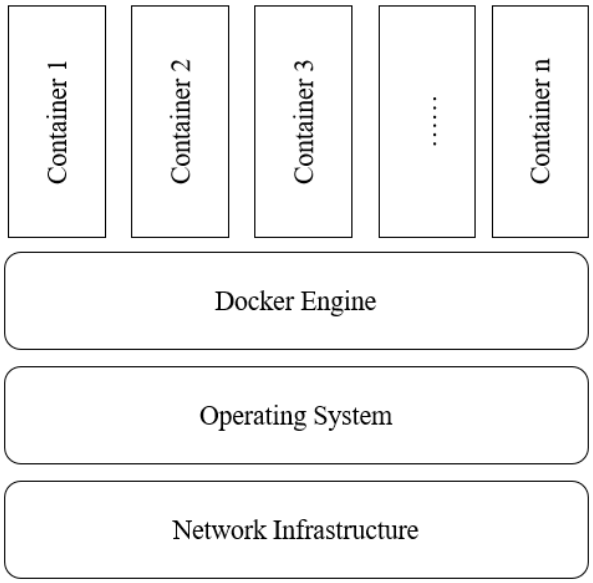

Fig. 1. Docker Architecture

As shown in Fig. 1, Docker Architecture contains infrastructure, host operating system, and Docker containers. Docker is a platform to develop, ship and run applications. Docker gives the provisions to sort the applications from the infrastructure so that the software can be delivered fast. First, it needs to pull the image from the public repository and then it can able to spawn a container out of that image. Also, from one image we can spawn many containers. A container is a runnable instance of an image and can be used to perform the tasks which the user wants to do.

An image is a template with a set of read-only instructions to create a Docker container. Generally, the images are dependent on each other by little or more customization. It is possible to create your own images by using the base image or you can use those already created images by others and published in an accessible registry. A Dockerfile must be created, in order to create your own image, which includes all the detailed commands needed to create an image and run it. While building an image using Dockerfile, it creates many containers at every intermediate step. But, at last, it creates only one container for an image using all those intermediate containers.

II. RELATED WORK

Antonio Brogi et.al. [5] introduced DOCKERFINDER for the enhancement in discovering the Docker images. They proposed the attribute-based way of searching the Docker images. The various attributes are the name and size of the image, distribution of supported software, etc. They also explained the working of DOCKERFINDER i.e., dragging remote Docker registry stored images. The local registry is used to stores the analyzed images and multi-attribute descriptions, which resulted from the analysis carried out by DOCKERFINDER.

Wang Kangjin Yang Yong et. al [6] explained the idea of Faster Image Distribution (FID). It is basically an image distribution system on a large scale, based on P2P. This is so helpful in increasing the Docker images distribution speed. And for the same, it uses the full bandwidth of Docker registry as well as the cluster nodes. Generally, FID reduces network traffic and distribution time up in a considerable amount.

Christopher B. Hauser et. al. [7] presented the concept that describes the images as a virtual environment. This concept consists of architecture and a metadata field set. The execution environment runs the virtual environment using declarative and deployable image registry.

Senthil Nathan et. al. [8] emphasized on a system which is used for management of various Docker images in between the different node sets. The system is known as CoMICon. It has come up with several features, by using which, they managed the images and made provisions for the management of applications related to distributed microservices.

Jeeva Chelladhurai et. al. [9] introduced various issues related to the Docker containers security. They also told the security matter related to different work. They worked on different algorithms and methods for security. The tests and experimentation carried out are very useful.

Qiang Liu, Wei Zheng et. al. [10], illustrated the uses Docker system, method of containerization for various lightweight software platforms. They pushed and pulled images to the docker repository and scaled it out. The deployment is also possible on various cloud services automatically.

Abdulrahman Azab [11], presented a security wrapper called as Socker. It is used to execute Docker containers on various queuing systems. They explained the difference between Docker and Socker applications. Socker can use Docker engine without replacements. Socker system has tested for different tasks on linked clusters.

Bin Xie et. al. [12] illustrated the use of network mode. Different network modes can be used for different applications. Authors used a specific application and explained the use of network modes. They also focused on the study of different modes and concluded the best mode out of all available.

Orest Lavriv et. al. [13] studied the process of continuous integration (CI) for various services related to the user. The CI is based on free modern software. They explained the various parameters by which the CI pipeline is comprised of. To handle the security and quality of the services CI is important. They proposed the infrastructure design and pipeline design for CI in this paper. Also, they used the server for local repository and Docker for deploying services.

Enrico Bacis et. al. [14] emphasized on adoption and retrieval of Docker images sticking to the security issues. The Docker has different security facilities for the Linux kernel. They illustrated the extension to the Dockerfile formats for reasons of security. The motive of the research was to enhance docker container security, by adopting different ways.

III. PROPOSED WORK

Docker is a very emerging technology now-a-days. It tries to acquire almost all the market of virtual machines. It is an open platform for building, deploying and managing virtualized application containers on a common operating

system. Today, it has been used on a large scale, in a higher level environment. It is provided with different services like storage, distribution, and management of Docker images. These services are important to execute Docker containers. A Docker image is a file which consists of multiple layers, used for the execution of code in a Docker container. And a Docker registry is a centralized repository which can store images of operating system and enterprise applications.

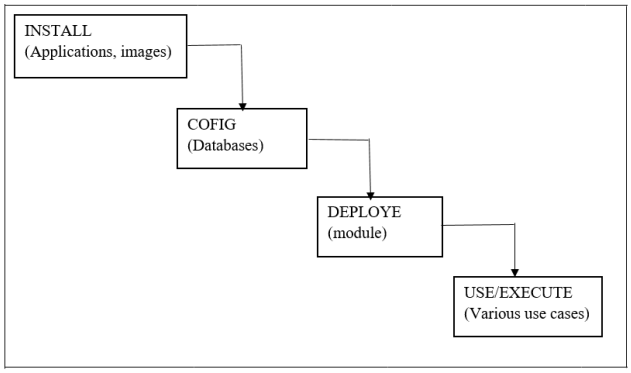

Fig. 2. Block Diagram

In the current scenario, the time consumption for installation of any operating system on any platform takes hours to install and run it properly. Also, users need to spend hours of their work in just installing the applications. But the Docker containers takes only a few seconds to install the operating system on any platform. So, it is more convenient to use Docker containers instead of the traditional way of installing any operating system or any application. Also, the existing system does not have the facility of automatic creation of images in Dockerfiles. But, through Docker, it's very easy to create Dockerfiles to automate the installation process. Also, an effective search logic for searching Docker images helps a lot in finding the correct images from the repository. And finally, all this work is integrated with the existing python automation framework for better utilization. Fig. 2 illustrates the overall process of development. The Docker Registry or Repository is a stateless, highly scalable server-side application that stores and lets user to distribute Docker images. Docker Engine is a Client-Server application. The 3 main components of Docker engine are; Server (daemon), Rest API (Interface), CLI (Client).

The first aim is to build a private centralized repository for employees of any organization. This goal plays a very important role in my project. In this step, the author proposes to build a private centralized repository for internal use of any organization. Sometimes, we don't want to publish our images publicly. So, in such cases, this Docker private registry can help you to keep your images safe at a single place called repository or registry and these images can be used by only the people of that organization only.

Here, three CentOS virtual machines were used for the registry, push, pull as a server, client1, client2, respectively. As shown in Fig. 3, the block at the root level shows the Sever for Docker repository. And the below blocks are the clients which able to perform the pull and push operations on the repository. Also, the TLS certificates are created at every machine for security purpose. And Authentication is done at the server for the user. If the user is authorized, then and only then one can pull or push the images to the repository.

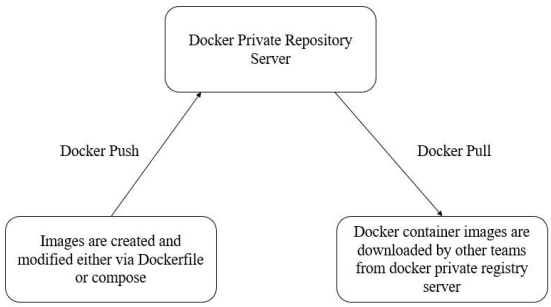

Fig. 3. Block diagram for Docker repository

Then generate Docker images for various platforms and automate image creation with Dockerfiles that works across platforms. The images are generated automatically by using just a single command. This magic happens due to Dockerfiles. Dockerfiles contains all the necessary commands which are required for the creation and installation on any image.

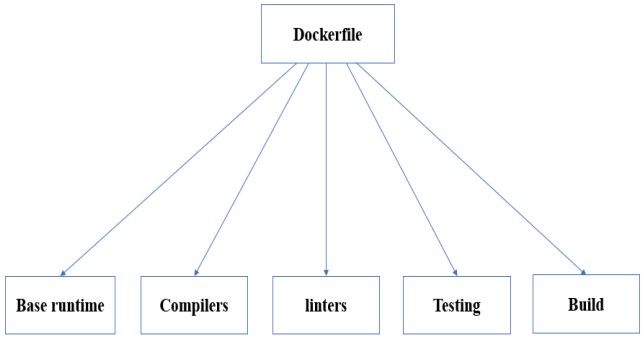

Fig. 4 Block Diagram for Dockerfile

Then the last aim is to develop effective search logic for searching Docker images and, going to find some new searching tool for better results. As the search logic plays a vital role in any search engine, here also good search logic is very import for getting better results of searching. So, here the author is going to find a good search logic which will be helpful for getting better search results. The author also proposes to build a new tool for searching.

IV. METHODOLOGY

A. Building of Private Centralized Repository

The following are the steps to create a centralized repository for the employees of any organization:
1) Creating a registry container on a private registry server virtual machine.
2) Edited /etc/hostname and /etc/hosts file with a combination of host name and domain name.
3) See whether all machines are pingable to each other or not.
4) Create a Docker private registry on the main server where we install registry container in first step.
5) Edit /usr/lib/systemd/system/docker.service file
6) Pull the image from Docker hub.
7) Tag the image to a private repository.
8) Push the image to a private repository.
9) Pull the image from the private repository.
10) Create TLS certificates on both sides client as well as sever.
11) While the creation of the certificates does not forget to give your CN (i.e. Common Name) as your registry URL.
12) Save TLS certificates and keys as a secret.
13) Update the node where you want to run the registry.
14) Create service, granting it access to the two secrets and constraining it to only run on nodes with the label.
15) Login to your registry through root login.
16) Able to pull and push images to the private registry.
17) These images created on one VM can be pulled to another VM also.

In this way, the creation of the private registry is done and solved the issue of login to the private repository.

B. Generating Docker images for enterprise applications for Linux platforms

The below steps are showing the creation of Docker image using Dockerfile for NetBackup on RedHat0 7.4:
1) Created client VM of Rhel7.2
2) mkdir/mnt/aedepot
3) mount the desired point from domain box
4) Unzip tar.gz file using; tar xvzf filename.tar.gz at /tmp.
5) Created script for installation on NBU.
6) Created Linux question answer yaml file which contains all the questions step by step which is required for installation.
7) Call the existing Linux question answer yaml in the above script.
8) Made the changes in nbu Linux yaml file as per my client details like name, IP, pwd.
9) At the client side, created a group using; groupadd nbwebgrp
10) Change the semaphore tuning value.
11) Run the script All the above steps are done in the Dockerfile and when you do the Docker build command the image will be created automatically with all the above requirements needed for installation. And for the usage of this image users just need to spawn a container using Docker run command.

C. Development of an effective Search logic for Searching Docker images

The next aim is to develop effective search logic for searching Docker images. So, here the good search logic which will be helpful for getting better search results is very important. So, it's very easy for users to search a particular image with its version from a large number of images in the repository. The following is the algorithm for search logic:
1) Checks for the format of the URL i.e. http or https.
2) Check the input format i.e. it should be in the proper format, otherwise, it will give the error as Cannot connect to Registry.
3) Check the command line argument i.e. search or list.
4) Parse the command line argument i.e. search or list.
5) Get a registry catalog request.
6) Get a registry tag request.
7) Extract URL: Check the delimiters, if none delimiter is present then show an error. Else, return it.
8) Get the list of all repos/images using repo array and parse functions.
9) Search for repo/images with the name or tag of the image using repo array and parse functions.
10) Get tags for the repo and get all the repo dictionary to get the final result.
11) Print the all available options.
Input: python search.py reponame.hostname.com:5000 search/list imagename
Output: Available options:
Name: imagename
Tag: imageversion n image found!

D. Enable existing framework automation on Docker images

In this step, the exiting framework of automation has been enabled using Dockerfiles. For this, the existing framework is enabled by adding deploy.sh, file which contains all the test cases, pytest module and all the files required in Dockerfile. The framework is enabled by using some of the commands in Dockerfile like RUN, ADD, COPY, etc

E. End to end integration of Docker module with existing automation framework: continuous integration

The work done around the Docker is integrated with the existing framework of automation by adding it to their

framework and allow it to the users for their use. The scripts for installing NetBackup and search logic is added to the existing framework. And the Docker repository is also added to the framework so that the employees of the organization can store their images securely in the repository.

V. CONCLUSION

This paper gives us the brief idea about how to use Docker in an efficient way to generate the Docker images for various platforms and to make a private centralized repository which can help the people of any organization to keep their images in a safe place. These images can be used by the people of that organization only. Also, good search logic helps them to search a particular image very quickly and easily. Also, the integration will help them to improve their work efficiency.

In the future, it is also possible to develop a better search tool for finding docker images. Also creating Dockerfiles for various application made easy for the individuals to install and use the application more efficiently.

REFERENCES

[1] "Docker Hub" https://hub.docker.com/
[2] "Docker documentation" https://docs.docker.com/
[3] "Docker Registry" https://docs.docker.com/registry/
[4] "Dockerfiles" https://docs.docker.com/engine/reference/builder/#usage
[5] Antonio Brogi; Davide Neri; Jacopo Soldani: Docker Finder: Multi-attribute Search of Docker Images. In: IEEE International Conference on Cloud Engineering (IC2E), Canada, pp. 273 – 278. (2017).
[6] Wang Kangjin, Yang Yong; Li Ying; Luo Hanmei, Ma Lin: FID: A Faster Image Distribution System for Docker Platform. In: 2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS*W), Tucson, AZ, USA, pp. 191-198. (2017)
[7] Christopher B. Hauser; Jörg Domaschka: ViCE Registry: An image Registry for virtual Collaborative Environment. In: 2017 IEEE 9th International Conference on Cloud Computing Technology and Science, Hong-kong, China, pp. 82 – 89. (2017)
[8] Senthil Nathan; Rahul Ghosh; Tridib Mukherjee; Krishnaprasad Narayanan. CoMICon: A Co-operative Management System for Docker Container Images. In: 2017 IEEE International Conference on Cloud Engineering, Vancouver, BC, Canada, pp. 116 – 126. (2017)
[9] Jeeva Chelladhurai; Pethuru Raj Chelliah; Sathish Alampalayam Kumar. Securing Docker Containers from Denial of Service (DoS) Attacks. In: 2016 IEEE International Conference on Services Computing, San Francisco, CA, USA, pp. 856 – 859. (2016)
[10] Qiang Liu; Wei Zheng; Ming Zhang; Yuxing Wang; Kexun Yu. In: Docker-based Automatic Deployment for Nuclear Fusion Experimental Data Archive Cluster. *IEEE Transactions On Plasma Science*. pp. 1281 – 1284. (2018)
[11] Abdulrahman Azab. Enabling Docker Containers for High-Performance and Many-Task Computing. In: 2017 IEEE International Conference on Cloud Engineering, Vancouver, BC, Canada, pp. 279 – 285. (2017)
[12] Bin Xie, Guanyi Sun, Guo Ma. Docker Based Overlay Network Performance Evaluation in Large Scale Streaming System. In: 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), Xi'an, China, pp. 366 – 369. (2016)
[13] Orest Lavriv, Bohdan Buhyl, Mykhailo Klymash, Ganna Grynkevych. Services continuous integration based on modern free infrastructure. In: 2nd International Conference on Advanced Information and Communication Technologies (AICT), Ukraine, pp. 150 – 153. (2017)
[14] Enrico Bacis, Simone Mutti, Steven Capelli, Stefano Paraboschi. DockerPolicyModules: Mandatory Access Control for Docker containers. In: IEEE Conference on Communications and Network Security (CNS), Italy, pp. 749-750. (2015)

Authors Profile

Miss. Sanskruti M. Shrawane pursued a Bachelor of Engineering in Computer Engineering from Bapurao Deshmukh College of Engineering, Wardha in 2017. She is currently pursuing a Master of Technology in Computer Science and Engineering from Walchand College of Engineering, Sangli.

Mrs. Medha A. Shah pursued a Bachelor of Technology from Walchand College of Engineering Sangli in 1997. She has completed her Masters in Computer Engineering from Walchand College of Engineering Sangli in 2008. Currently, she is working as a professor in the same college as well as pursuing her Ph.D.

