Open Access Library Journal
2019, Volume 6, e5519
ISSN Online: 2333-9721
ISSN Print: 2333-9705

Establishment of the Docker-Based Laboratory Environment

Fang Hu[1], Shijun Che[2]

[1]Sichuan Staff University of Science and Technology, Chengdu, China
[2]Chengdu No. 43 Middle School, Chengdu, China
Email: 635675317@qq.com

How to cite this paper: Hu, F. and Che, S.J. (2019) Establishment of the Docker-Based Laboratory Environment. *Open Access Library Journal*, **6**: e5519.
https://doi.org/10.4236/oalib.1105519

Received: June 3, 2019
Accepted: June 25, 2019
Published: June 28, 2019

Abstract

At present, part of the experimental environment of computer science courses and the software development environment of the school are built based on virtual machines. As the number of students rapidly increases, the demand for virtual machines goes up correspondingly. Virtual machines consume a lot of resources, and the shortage of resources becomes the bottleneck of laboratory construction. According to the current situation of school resources, this paper proposes to build a docker-based laboratory environment for computer course teaching as well as software development and test, so as to provide teachers and students with various lightweights and stable Linux system services.

Subject Areas

Computer Engineering

Keywords

Docker Container, Image, Private Repositories, Cluster

1. Docker Technology Introduction

Docker is an open-source project that automates the deployment of applications inside software containers, by providing an additional layer of abstraction and automation of operating-system-level virtualization on Linux. Docker implements a high-level API to provide lightweight containers that run processes in isolation, and could provide virtualization solutions [1]. Currently, PAAS and IAAS architecture patterns are gradually changed in cloud computing platform, instead, major cloud service providers have successively provided docker-based container cloud services [2].

1.1. Docker Container and Virtual Machine

Both Docker container and virtual machine are used for resource and environment isolation [3]. Virtual machine draws upon the Hypervisor to virtualize hardware resources, such as CPU, memory, I/O devices, etc. [4]. Its essence is to simulate plenty of virtual servers on a physical server, and these virtual servers can operate like a real server for a variety of application deployment. However, the price that comes with the convenience is that each virtual machine needs more hardware resources. Docker based on the host operating system kernel adopts Cgroup and Namespace to implement resource isolation, which does not need a complete operating system instance. Consequently, using Docker can greatly reduce the resource occupancy of server CPU, memory and so on [5]. The application that runs in the Docker container directly uses the hardware resource of physical machines, so that the Docker has certain advantages over the traditional virtual machine in terms of hardware utilization.

1.2. Docker Framework

Docker follows a Client-Server architecture and is divided into three main parts. Figure 1 shows the framework diagram of Docker. Docker_Host contains containers, images, and Docker daemons. It provides a comprehensive environment to run applications. Docker Client is the platform where the Docker Client interacts with the Docker daemon. The Docker Client sends a request command to the server, and the server performs the corresponding operations, such as building the image, running and distributing the container, and returning the result to the client. Docker hub/registry is responsible for sharing and managing Docker images. Users can upload or download the containing images. The official address is https://registry.hub.docker.com/, or you can set up your own private Docker registry. When pulling a mirror, if no warehouse address is specified, the default will stem from the public library maintained by Docker [6]. For better understanding Docker, some concepts are briefly introduced as follows. Docker image is a multi-level read-only application template with a complete runtime environment such as Linux, MySQL etc. These images are highly reusable. A

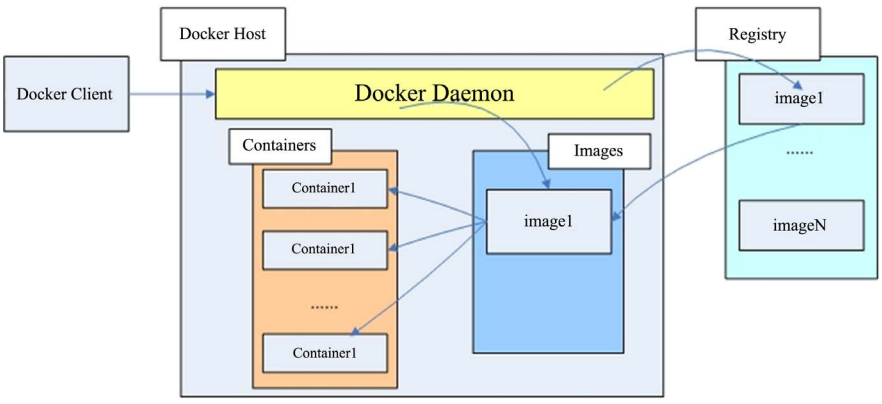

Figure 1. Docker framework.

Mirror is used to create a container and one image can run multiple containers. Docker images can be customized, and the corresponding content is recorded by the Dockerfile. The advantage is that the operation and maintenance personnel only need the Dockerfile to create the specified container, which will greatly facilitate the propagation and storage of Docker images, achieving the follow-up [7]. A Docker container is a running instance of a mirror that can be switched on, started, stopped, or deleted. The container provides an isolated environment so that there is no interface between any containers. It is featured with the secure access to resources, which ensures that the programs in the container run in a relatively secure environment. Containers consume low resources and can be easily run on machines and data centers. Swarm is currently the only Docker officially specified cluster management tool. Starting from Docker 1.12, the swarm mode is inserted. It converts a system of multiple Docker hosts into a single virtual Docker host, allowing containers to form a subnet network across hosts.

2. Lab Information

All students in the Department of Information Engineering in our school are required to take the "Network Operating System" course in the third semester. Based on two ThinkServer QT940 servers, we have achieved virtualized cluster of servers and built 46 virtual machines, among which 22 are installed with Windows server2008 R2 while 26 are installed with CentOS7 using XenServer. 20 Windows-based virtual machines and 20 Linux-based virtual machines are provided for students with their Network Operating System course experiments. The remaining 6 are designated to software development and testing. As the demand gradually changes recently, at least 80 Windows virtual machines and 80 Linux virtual machines are simultaneously needed to support the teaching, and the existing virtual machine cluster resources cannot meet the demand. When students are studying web design, website development and other courses, they need to deploy databases, web services and other applications; also, the research teams of our department have put forward the need to build a more robust integration environment.

3. Countermeasures

As we have mentioned above, Docker is featured with lightweight, fast startup, and low resource consumption. Therefore, we use the original virtualized cluster to build 82 virtual machines based on Windows. And we select two ThinkServer RD 630 to build an experimental environment for the Linux platform based on Docker. Details are shown in Table 1.

Table 1. Linux experimental environment establishment.

Server model	Host OS	Host Name	Host IP	Private Repositories	Cluster Node Role
Thinkserver RD 630	CentOS7	Swarm01	192.168.3.2	Swarm01	Leader
		Swarm02	192.168.3.3		Worker

4. Procedure [8]

1) Install CentOS7, Docker on two servers respectively, and set the host name, IP address, and so on. OS and Docker versions are as follows:

CentOS7 version: 3.10.0-957.10.1.el7.x86_64

Docker version:

[root@swarm01 ~] # docker version

Client:

Version: 1.13.1

API version: 1.26

Package version: docker-1.13.1-94.gitb2f74b2.el7.centos.x86_64

Go version: go1.10.3

Git commit: b2f74b2/1.13.1

Built: Tue Mar 12 10:27:24 2019

OS/Arch: Linux/amd64

Server:

Version: 1.13.1

API version: 1.26 (minimum version 1.12)

Package version: docker-1.13.1-94.gitb2f74b2.el7.centos.x86_64

Go version: go1.10.3

Git commit: b2f74b2/1.13.1

Built: Tue Mar 12 10:27:24 2019

OS/Arch: linux/amd64

Experimental: false

2) To build a local mirror repository on Swarm01. First, pull registry and run. Then add a private image repository source by adding the following to the /etc/docker/daemon.json file on the server Swarm01 followed by a restart of Docker:

{

"insecure-registries": ["192.168.3.2:5000"]

}

(3) Create a CentOS7 image that supports the SSH service on Swarm01 and push it to the local repository. Then, start the base image, create a new container named centos7ssh, and start a Bash terminal to interact with it. The centos7ssh image is shown in Figure 2.

[root@swarm01 /] # docker run -it --name centos7ssh centos:7 /bin/bash

Container ID is: 276490788fca.

Add the SSH service to the container centos7ssh, then submit the container and convert it to a new image named sshd_centos7.

```
[root@swarm01 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
     PORTS                    NAMES
276490788fca        centos:7            "/bin/bash"              2 minutes ago       Up 2 minutes
                                 centos7ssh
0126ea3bd647        registry            "/entrypoint.sh /e..."   13 hours ago        Up About an hou
r    0.0.0.0:5000->5000/tcp   friendly_newton
```

Figure 2. The centos7ssh image list.

[root@swarm01 /] # docker commit 276490788fca sshd_centos7

sha256:1a89847f75ee631a89da101993e46ba743ab375ef8b175a874e0c4150e10156c

Tag it and push it to the local repository.

#docker tag 192.168.3.2:5000/sshd_centos7

#dockerpush 192.168.3.2:5000/sshd_centos7.

The result is shown in Figure 3.

Figure 4 shows the procedure of how to pull 192.168.3.2:5000/sshd_centos7 from the local repository in host Swarm02.

The following steps verify remote login to the container af674da76d8d. First, start the container that mirrors sshd_centos7 on the host Swarm01, and map the local port 15,000 to the port 22 of the container to start the SSHD service of the container.

docker run -d -p 15000:22 sshd_centos7 /usr/sbin/sshd –D

[root@swarm01 /] # docker run -d -p 15000:22 sshd_centos7 /usr/sbin/sshd -D

af674da76d8d1d68152b5de34e88655bd20928128ac2c87839e83883a2eb0858

The result is shown in Figure 5.

Then, open a new terminal and enter the host IP address 192.168.3.2, port: 15000.

The result is shown in Figure 6.

4) Use the swarm mode to build a cluster to manage various types of containers. The management node is Swarm01 and the worker node is Swarm02. We should open the relevant firewall port of the host Swarm01#:

firewall-cmd --zone=public --add-port=2377/tcp --permanent

firewall-cmd --zone=public --add-port=7946/tcp --permanent

firewall-cmd --zone=public --add-port=7946/udp --permanent

firewall-cmd --zone=public --add-port=4789/udp --permanent

Initialize the cluster. The IP address of the nodes communicating with each other is 192.168.3.2, and the default port is 2377.

The result is shown in Figure 7.

Then, add the host Swarm02 to the cluster. The list of nodes is shown in Figure 8.

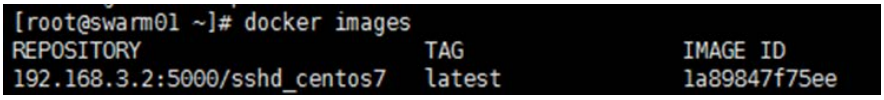

Figure 3. The mirror in the local repository 192.168.3.2:5000/sshd_centos7.

Figure 4. Host swarm02 pulls mirror 192.168.3.2:5000/sshd_centos7.

```
[root@swarm01 /]# docker ps -a
CONTAINER ID        IMAGE               COMMAND               CREATED             STATUS                      PORTS
                    NAMES
af674da76d8d        sshd_centos7        "/usr/sbin/sshd -D"   39 seconds ago      Up 38 seconds               0.0.0.0:
15000->22/tcp       romantic_pare
276490788fca        centos:7            "/bin/bash"           About an hour ago   Up 41 minutes
                    centos7ssh
```

Figure 5. Host swarm01 container list.

```
Connecting to 192.168.3.2:15000...
Connection established.
To escape to local shell, press 'Ctrl+Alt+]'.

Last login: Sun Apr 21 03:58:03 2019 from localhost
[root@af674da76d8d ~]#
```

Figure 6. A success login of remote connection to the container af674da76d8d.

```
[root@swarm01 ~]# docker swarm init --advertise-addr 192.168.3.2
Swarm initialized: current node (geplj3j6gyltztro6270wknnp) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join \
    --token SWMTKN-1-1pgcqylhze7e71q8qo6d5cbe3t98z1ztkbwsaf4q3iew4zgsuy-29ojpgtgoz6xh41en5s1cuzfi \
    192.168.3.2:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

Figure 7. Cluster initialization.

```
[root@swarm01 ~]# docker node ls
ID                            HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
b52qdryg8ky6n85ivve0thkfa     swarm02   Ready   Active
geplj3j6gyltztro6270wknnp *   swarm01   Ready   Active        Leader
[root@swarm01 ~]#
```

Figure 8. Cluster nodes list.

Up to now, the laboratory environment has been built and tested.

Teachers in each teaching could build a mirror image of students' practice environment and create services according to the content of the course. Each teaching and research section could build a continuous integration environment according to their own needs, and optimizes program development, testing, system operation and maintenance.

5. Concluding Remarks

In summary, Docker can quickly build and deploy applications, as well as build a highly flexible distributed system, making full use of hardware resources to reduce corresponding costs. After two months of operation, this experimental program solved the problem of insufficient server resources, and met the teaching needs of relevant courses and the research needs of the teaching and research section. Further research topic could be studying Docker's security, resource isolation, and exploring its deep application in the actual development environment.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Bernstein, D. (2014) Containers and Cloud: From Lxc to Docker to Kubernetes. *IEEE Cloud Computing*, **1**, 81-84. https://doi.org/10.1109/MCC.2014.51

[2] Boettiger, C. (2015) An Introduction to Docker for Reproducible Research. *ACM SIGOPS Operating Systems Review*, **49**, 71-79. https://doi.org/10.1145/2723872.2723882

[3] Fink, J. (2014) Docker: A Software as a Service, Operating System-Level Virtualization Framework. *Code4Lib Journal*, **25**, 29.

[4] Combe, T., Martin, A. and Di Pietro, R. (2016) To Docker or Not to Docker: A Security Perspective. *IEEE Cloud Computing*, **3**, 54-62. https://doi.org/10.1109/MCC.2016.100

[5] Mouat, A. (2015) Using Docker: Developing and Deploying Software with Containers. O'Reilly Media, Inc., Sebastopol.

[6] Matthias, K. and Kane, S.P. (2015) Docker: Up & Running: Shipping Reliable Containers in Production. O'Reilly Media, Inc., Sebastopol.

[7] Peinl, R., Holzschuher, F. and Pfitzer, F. (2016) Docker Cluster Management for the Cloud-Survey Results and Own Solution. *Journal of Grid Computing*, **14**, 265-282. https://doi.org/10.1007/s10723-016-9366-y

[8] Seo, K.-T., *et al.* (2014) Performance Comparison Analysis of Linux Container and Virtual machine for Building cloud. *Advanced Science and Technology Letters*, **66**, 105-111. https://doi.org/10.14257/astl.2014.66.25