南昌大学实验报告

姓名：Qing Liu

学号：6130116184

邮箱地址：1119637652@qq.com

专业班级：Class 164 of Computer Science and Technology

实验日期：THU. April 4th, 2019

课程名称：Cloud Computing Technology Experiments

实验项目名称

Introduction to Cloud Computing -- Virtualized Services

实验目的

- **Successfully install a Linux system with all basic capabilities**
- **Build an IaaS service system - Docker**
- **Finish the Docker Service initialization**

实验基础

- **Hardware: Lenovo Ideapad 700 - 15ISK**
- **Software: Vmware Workstation Pro, Ubuntu 18.04 LTS and Docker CE**

实验步骤

Build an OS

Before this task, I have already installed the Ubuntu 18.04 LTS on VMware, which was required in the second homework. And there is an Ubuntu 14.04 KVM in it. So I do experiments directly on the basis of this virtual machine. But I didn't use the kvm. I do the experiment in the Ubuntu 18.04 LTS.

In Ubuntu OS, once you create an user and set the password successfully, it will be added into the sudoer list so that we can use `sudo command-sequence` to get the root privilege temporarily to execute the command or use `sudo su` and enter the user's password to change to user root instead of root's password. In Ubuntu of my virtual machine, there is only one user cleo.

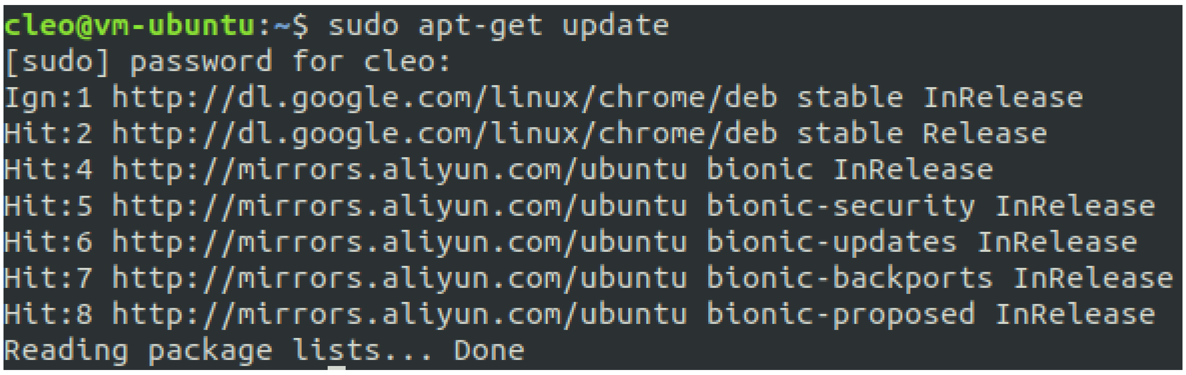
Initialize a Service

I installed Docker CE by setting up the repository firstly.

Set up the repository

1. Update the apt package index

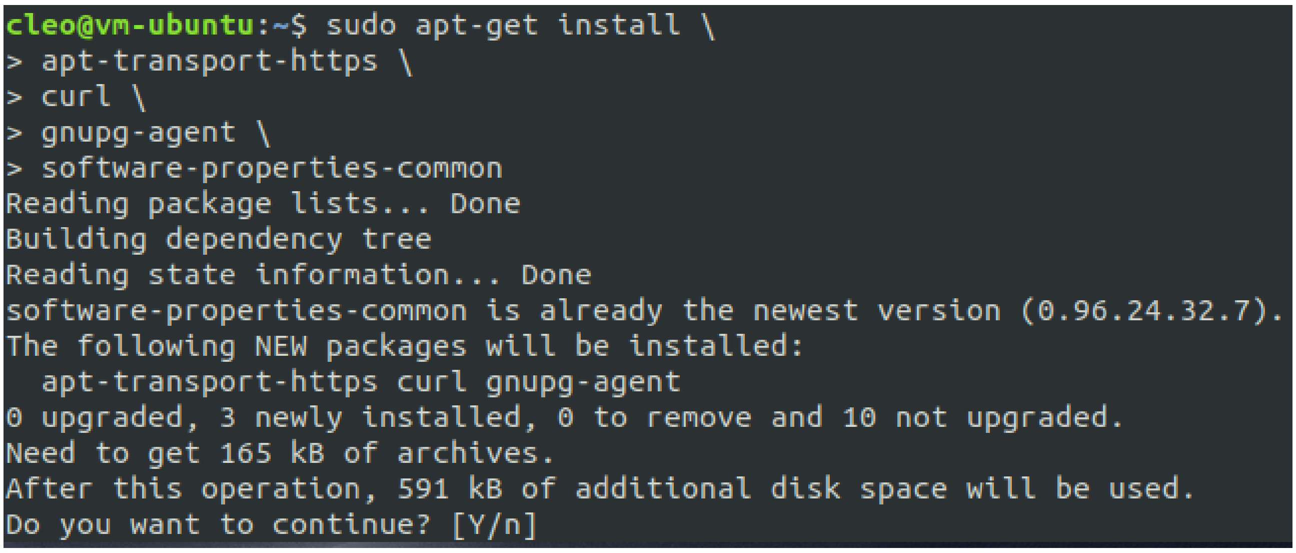
```
$ sudo apt-get update
```



```
cleo@vm-ubuntu:~$ sudo apt-get update
[sudo] password for cleo:
Ign:1 http://dl.google.com/linux/chrome/deb stable InRelease
Hit:2 http://dl.google.com/linux/chrome/deb stable Release
Hit:4 http://mirrors.aliyun.com/ubuntu bionic InRelease
Hit:5 http://mirrors.aliyun.com/ubuntu bionic-security InRelease
Hit:6 http://mirrors.aliyun.com/ubuntu bionic-updates InRelease
Hit:7 http://mirrors.aliyun.com/ubuntu bionic-backports InRelease
Hit:8 http://mirrors.aliyun.com/ubuntu bionic-proposed InRelease
Reading package lists... Done
```

2. Install packages to allow apt to use a repository over HTTPS

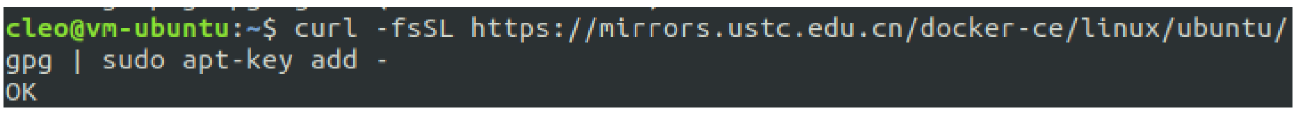
```
$ sudo apt-get install \
  apt-transport-https \
  curl \
  gnupg-agent \
  software-properties-common
```



```
cleo@vm-ubuntu:~$ sudo apt-get install \
> apt-transport-https \
> curl \
> gnupg-agent \
> software-properties-common
Reading package lists... Done
Building dependency tree
Reading state information... Done
software-properties-common is already the newest version (0.96.24.32.7).
The following NEW packages will be installed:
  apt-transport-https curl gnupg-agent
0 upgraded, 3 newly installed, 0 to remove and 10 not upgraded.
Need to get 165 kB of archives.
After this operation, 591 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

3. Add Docker's official GPG key (remember to change the mirror)

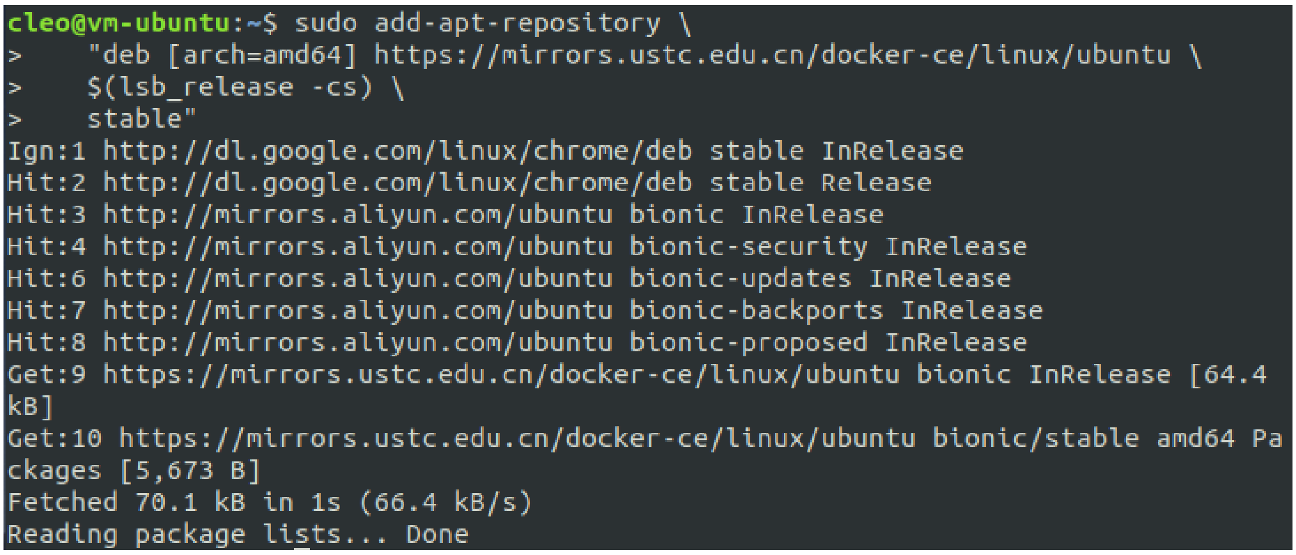
```
$ curl -fsSL  https://mirrors.ustc.edu.cn/docker-ce/linux/ubuntu/gpg | sudo apt-
```



```
cleo@vm-ubuntu:~$ curl -fsSL https://mirrors.ustc.edu.cn/docker-ce/linux/ubuntu/
gpg | sudo apt-key add -
OK
```

4. Use the following command to set up the stable repository (remember to change the mirrror)

```
$ sudo add-apt-repository \
  "deb [arch=amd64] https://mirrors.ustc.edu.cn/docker-ce/linux/ubuntu \
  $(lsb_release -cs) \
  stable"
```



```
cleo@vm-ubuntu:~$ sudo add-apt-repository \
>    "deb [arch=amd64] https://mirrors.ustc.edu.cn/docker-ce/linux/ubuntu \
>    $(lsb_release -cs) \
>    stable"
Ign:1 http://dl.google.com/linux/chrome/deb stable InRelease
Hit:2 http://dl.google.com/linux/chrome/deb stable Release
Hit:3 http://mirrors.aliyun.com/ubuntu bionic InRelease
Hit:4 http://mirrors.aliyun.com/ubuntu bionic-security InRelease
Hit:6 http://mirrors.aliyun.com/ubuntu bionic-updates InRelease
Hit:7 http://mirrors.aliyun.com/ubuntu bionic-backports InRelease
Hit:8 http://mirrors.aliyun.com/ubuntu bionic-proposed InRelease
Get:9 https://mirrors.ustc.edu.cn/docker-ce/linux/ubuntu bionic InRelease [64.4
kB]
Get:10 https://mirrors.ustc.edu.cn/docker-ce/linux/ubuntu bionic/stable amd64 Pa
ckages [5,673 B]
Fetched 70.1 kB in 1s (66.4 kB/s)
Reading package lists... Done
```

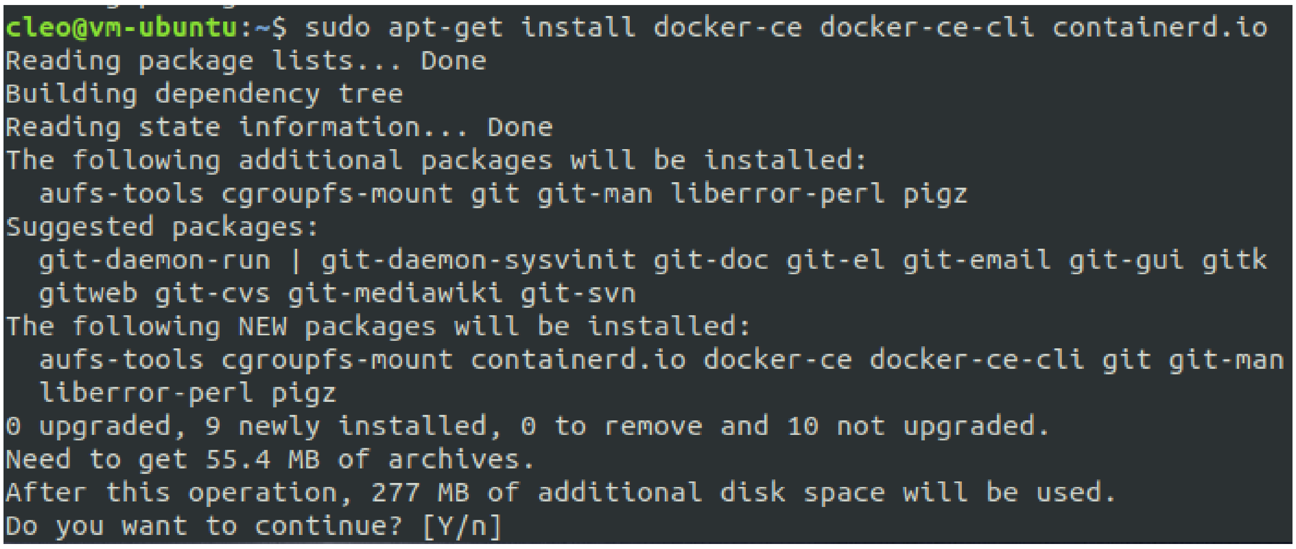
Install Docker CE

1. Update the `apt` package index.

```
$ sudo apt-get update
```

2. Install the latest version of Docker CE and containerd. If you change the mirror to internal mirror, it will be very fast.

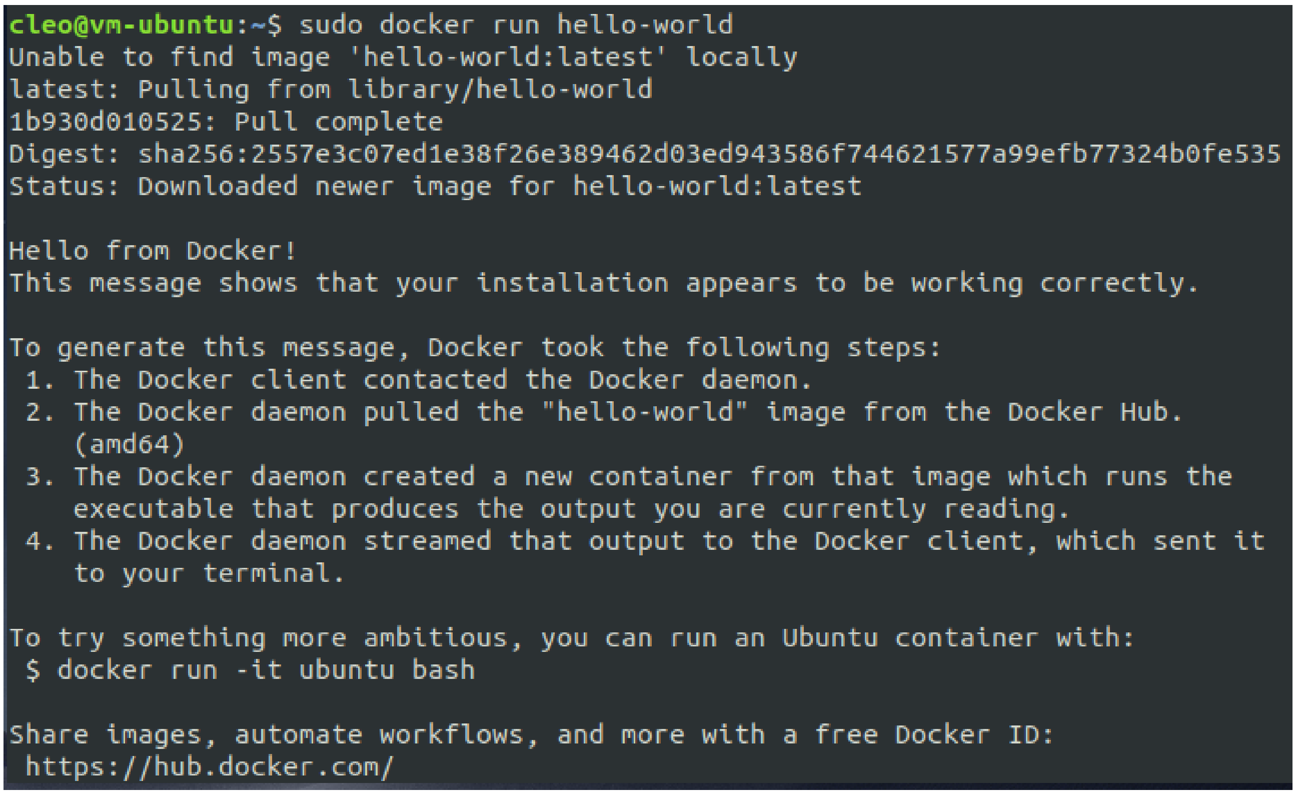
```
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```



```
cleo@vm-ubuntu:~$ sudo apt-get install docker-ce docker-ce-cli containerd.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  aufs-tools cgroupfs-mount git git-man liberror-perl pigz
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk
  gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  aufs-tools cgroupfs-mount containerd.io docker-ce docker-ce-cli git git-man
  liberror-perl pigz
0 upgraded, 9 newly installed, 0 to remove and 10 not upgraded.
Need to get 55.4 MB of archives.
After this operation, 277 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

3. Verify that Docker CE is installed correctly by running the hello-world image.

```
$ sudo docker run hello-world
```

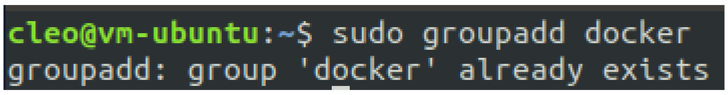


```
cleo@vm-ubuntu:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:2557e3c07ed1e38f26e389462d03ed943586f744621577a99efb77324b0fe535
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/
```

Add the docker group to avoid using docker with `sudo`

1. Create a new user group

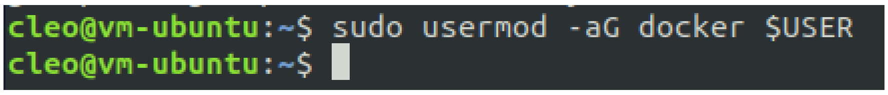
```
$ sudo groupadd docker
```



```
cleo@vm-ubuntu:~$ sudo groupadd docker
groupadd: group 'docker' already exists
```

2. Add user to group docker

```
$ sudo usermod -aG docker $USER
```



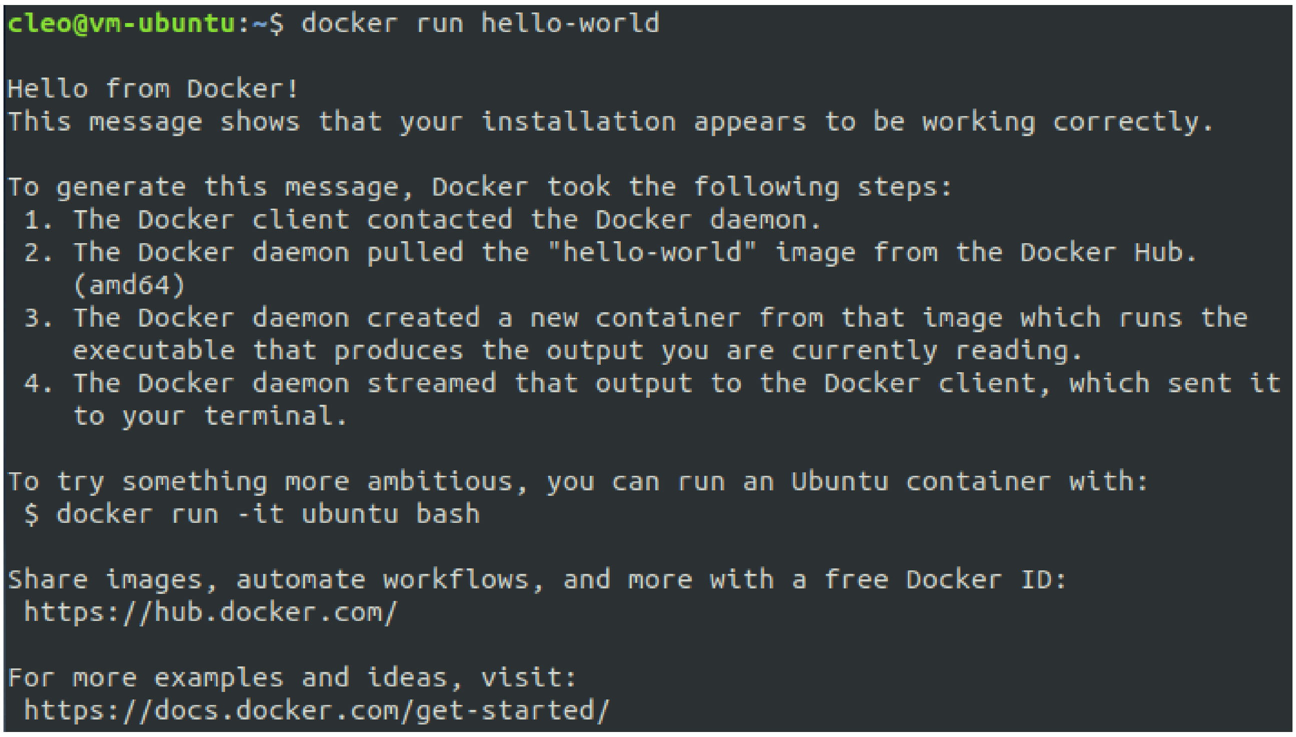
```
cleo@vm-ubuntu:~$ sudo usermod -aG docker $USER
cleo@vm-ubuntu:~$
```

3. Restart the virtual machine

```
$ reboot
```

4. Verify that you can run docker commands without sudo.

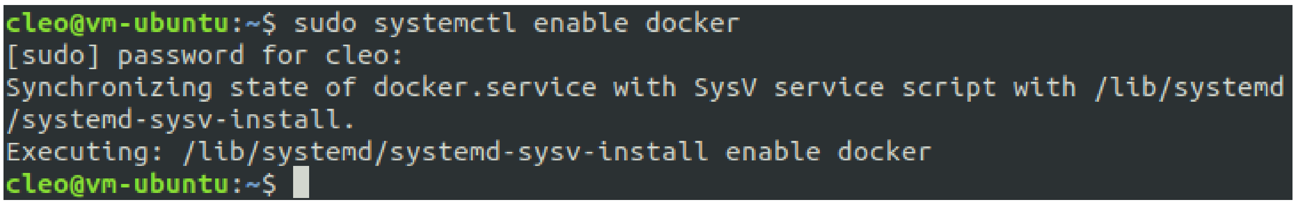
```
$ docker run hello-world
```



```
cleo@vm-ubuntu:~$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Enable Docker service to start while the system boot.

```
$ sudo systemctl enable docker
```



```
cleo@vm-ubuntu:~$ sudo systemctl enable docker
[sudo] password for cleo:
Synchronizing state of docker.service with SysV service script with /lib/systemd
/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable docker
cleo@vm-ubuntu:~$
```

Configuration of Docker Accelerator

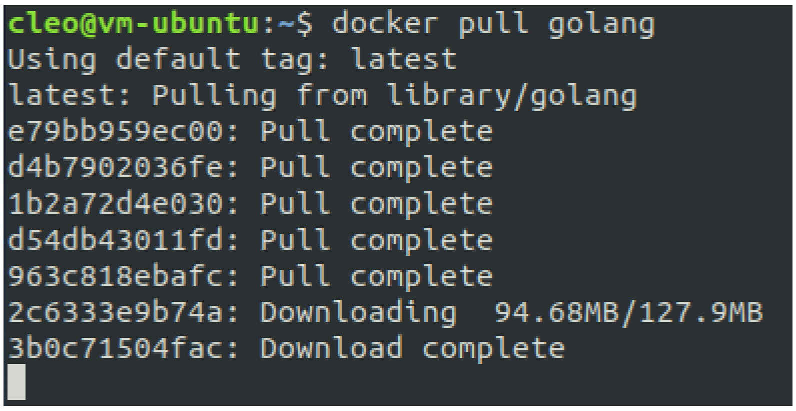
```
$ curl -sSL https://get.daocloud.io/daotools/set_mirror.sh | sh -s http://f1361d
```

Docker Image Deployment

Deploy golang image from Docker Hub

1. Pull from Docker Hub

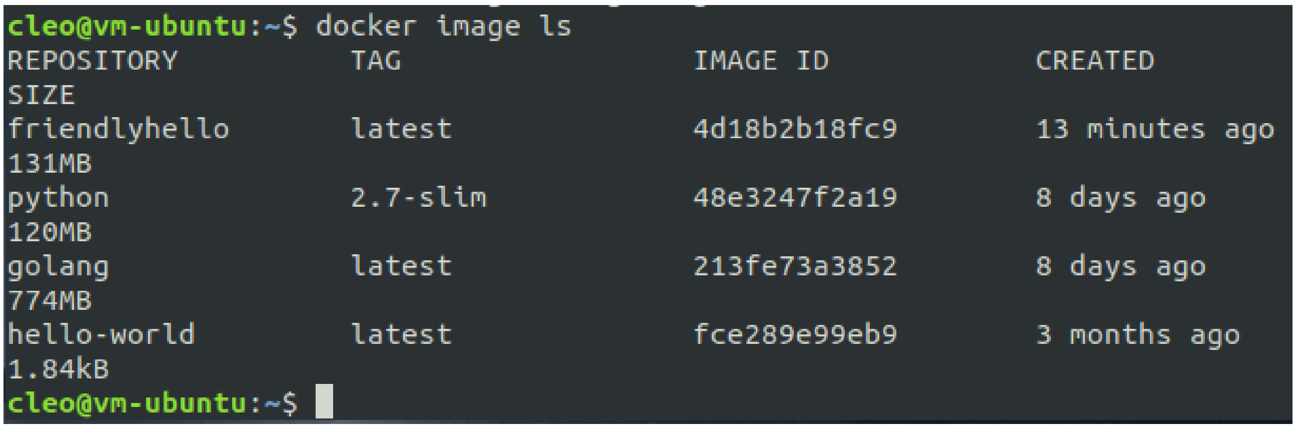
```
$ docker pull golang
```



```
cleo@vm-ubuntu:~$ docker pull golang
Using default tag: latest
latest: Pulling from library/golang
e79bb959ec00: Pull complete
d4b7902036fe: Pull complete
1b2a72d4e030: Pull complete
d54db43011fd: Pull complete
963c818ebafc: Pull complete
2c6333e9b74a: Downloading  94.68MB/127.9MB
3b0c71504fac: Download complete
```

2. List the imges which was downloaded

```
$ docker image ls
```



```
cleo@vm-ubuntu:~$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
friendlyhello       latest              4d18b2b18fc9        13 minutes ago
131MB
python              2.7-slim            48e3247f2a19        8 days ago
120MB
golang              latest              213fe73a3852        8 days ago
774MB
hello-world         latest              fce289e99eb9        3 months ago
1.84kB
cleo@vm-ubuntu:~$
```

Code for Hello World

1. Create a new project directory named gohello and enter into it.

```
$ mkdir gohello && cd gohello
```

2. Create a new file named hello.go, and the comtent is the code of the `helloworld` program

```
$ vim hello.go
```

```
package main

import (
  "fmt"
  "net/http"
)

func indexHandler(w http.ResponseWriter, r *http.Request) {
  fmt.Fprintf(w, "Hello World")
}

func main() {
  http.HandleFunc("/", indexHandler)
  http.ListenAndServe(":8000", nil)
}
```

3. Edit the Dockerfile

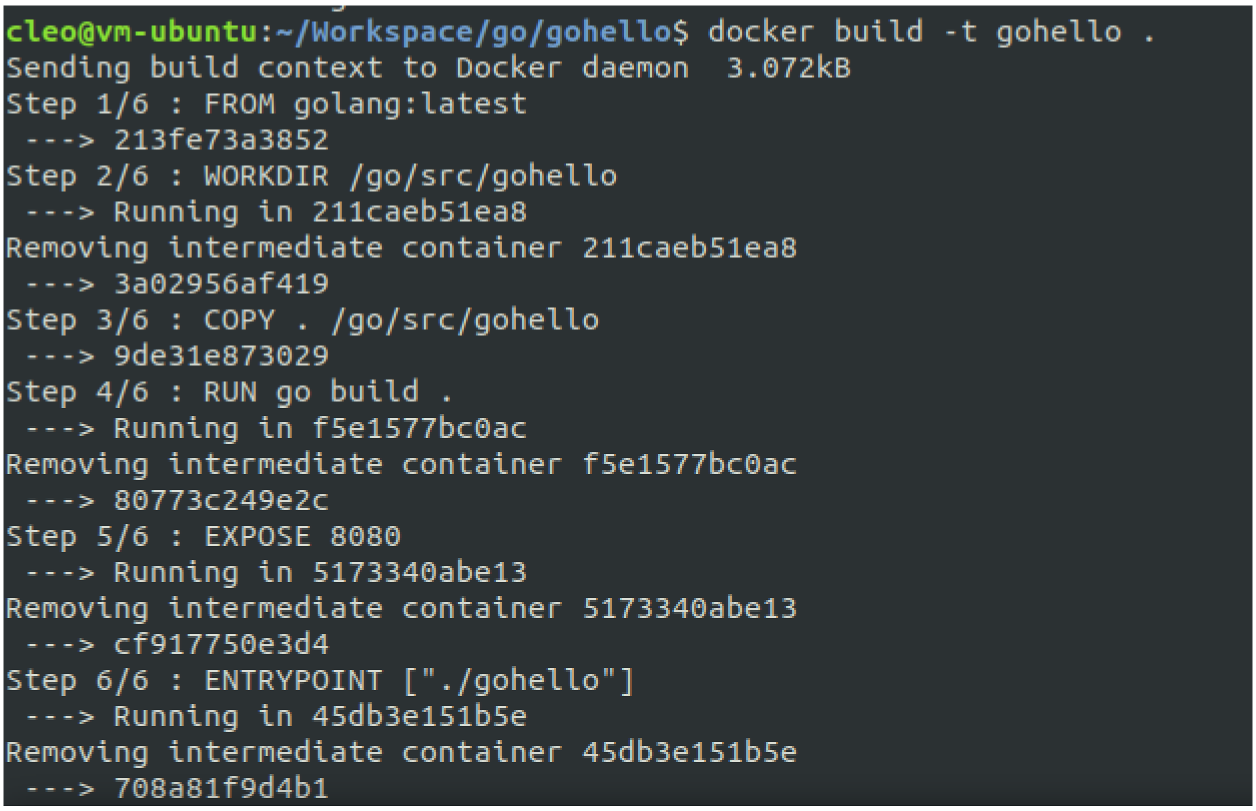
```
$ vim Dockerfile
```

```
FROM golang:latest

WORKDIR /go/src/gohello
COPY . /go/src/gohello

RUN go build .

EXPOSE 8080

ENTRYPOINT ["./gohello"]
```

Build and Run

1. Build the docker image

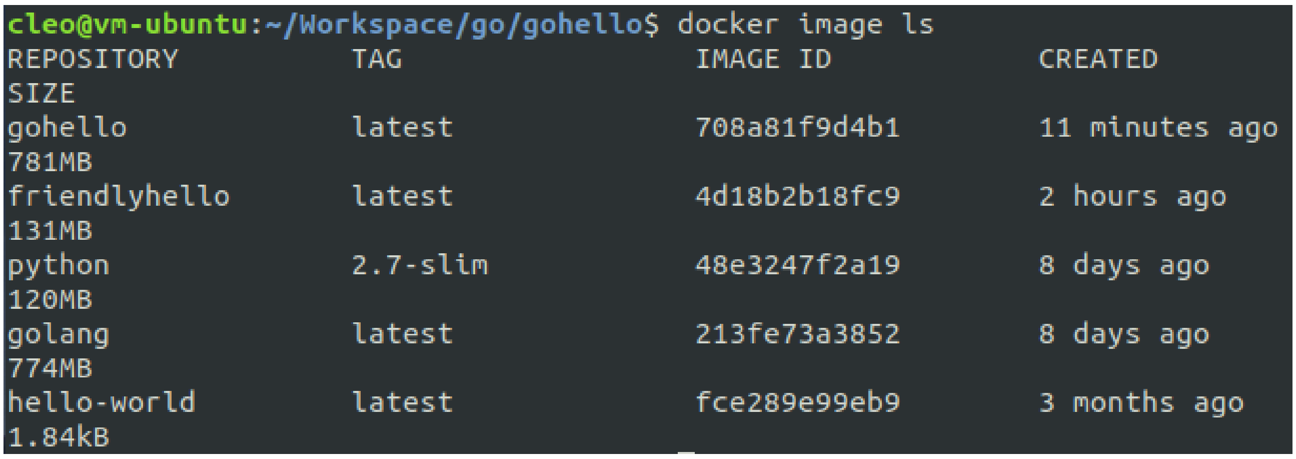
```
$ docker build -t gohello .
```



```
cleo@vm-ubuntu:~/Workspace/go/gohello$ docker build -t gohello .
Sending build context to Docker daemon  3.072kB
Step 1/6 : FROM golang:latest
 ---> 213fe73a3852
Step 2/6 : WORKDIR /go/src/gohello
 ---> Running in 211caeb51ea8
Removing intermediate container 211caeb51ea8
 ---> 3a02956af419
Step 3/6 : COPY . /go/src/gohello
 ---> 9de31e873029
Step 4/6 : RUN go build .
 ---> Running in f5e1577bc0ac
Removing intermediate container f5e1577bc0ac
 ---> 80773c249e2c
Step 5/6 : EXPOSE 8080
 ---> Running in 5173340abe13
Removing intermediate container 5173340abe13
 ---> cf917750e3d4
Step 6/6 : ENTRYPOINT ["./gohello"]
 ---> Running in 45db3e151b5e
Removing intermediate container 45db3e151b5e
 ---> 708a81f9d4b1
```

2. List the images and we can find that there is a new image called gohello

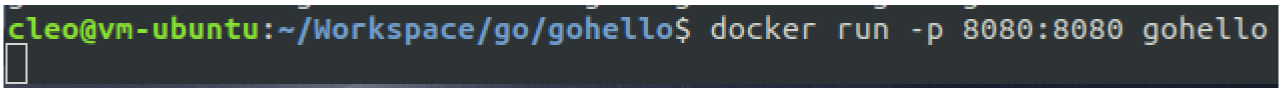
```
$ docker image ls
```



```
cleo@vm-ubuntu:~/Workspace/go/gohello$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
gohello             latest              708a81f9d4b1        11 minutes ago
781MB
friendlyhello       latest              4d18b2b18fc9        2 hours ago
131MB
python              2.7-slim            48e3247f2a19        8 days ago
120MB
golang              latest              213fe73a3852        8 days ago
774MB
hello-world         latest              fce289e99eb9        3 months ago
1.84kB
```

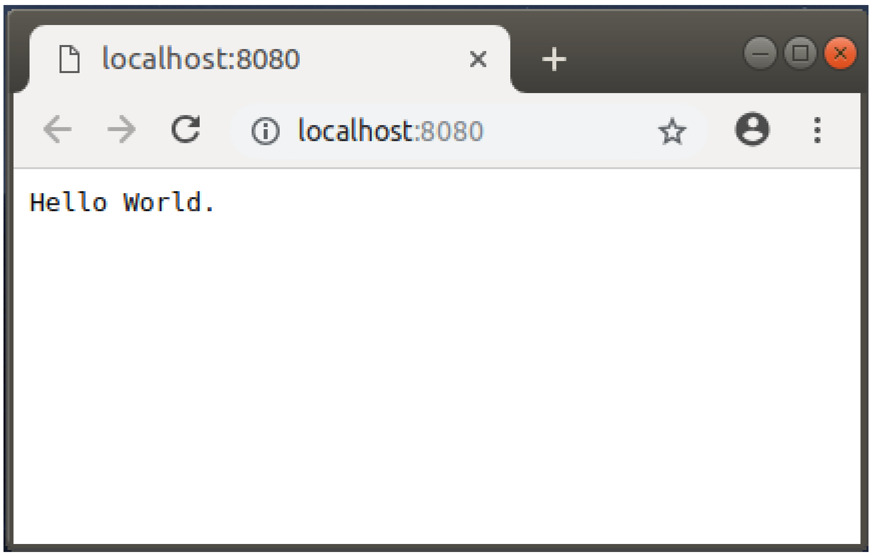
3. Run the gohello image

```
$ docker run -p 8080:8080 gohello
```



```
cleo@vm-ubuntu:~/Workspace/go/gohello$ docker run -p 8080:8080 gohello
```

4. Open the browser and enter `localhost:8080', then we can see the "Hello World"



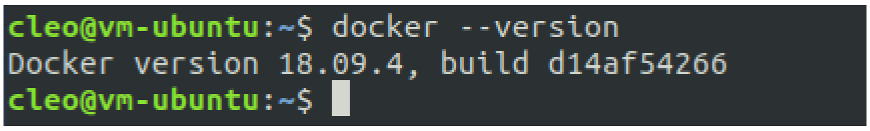
```
Hello World.
```

5. Stop the runing image. If it is running as foreground mode, we can use `ctrl + C` to stop it. If it is running as background mode, we can use the command `docker stop id` to stop it.

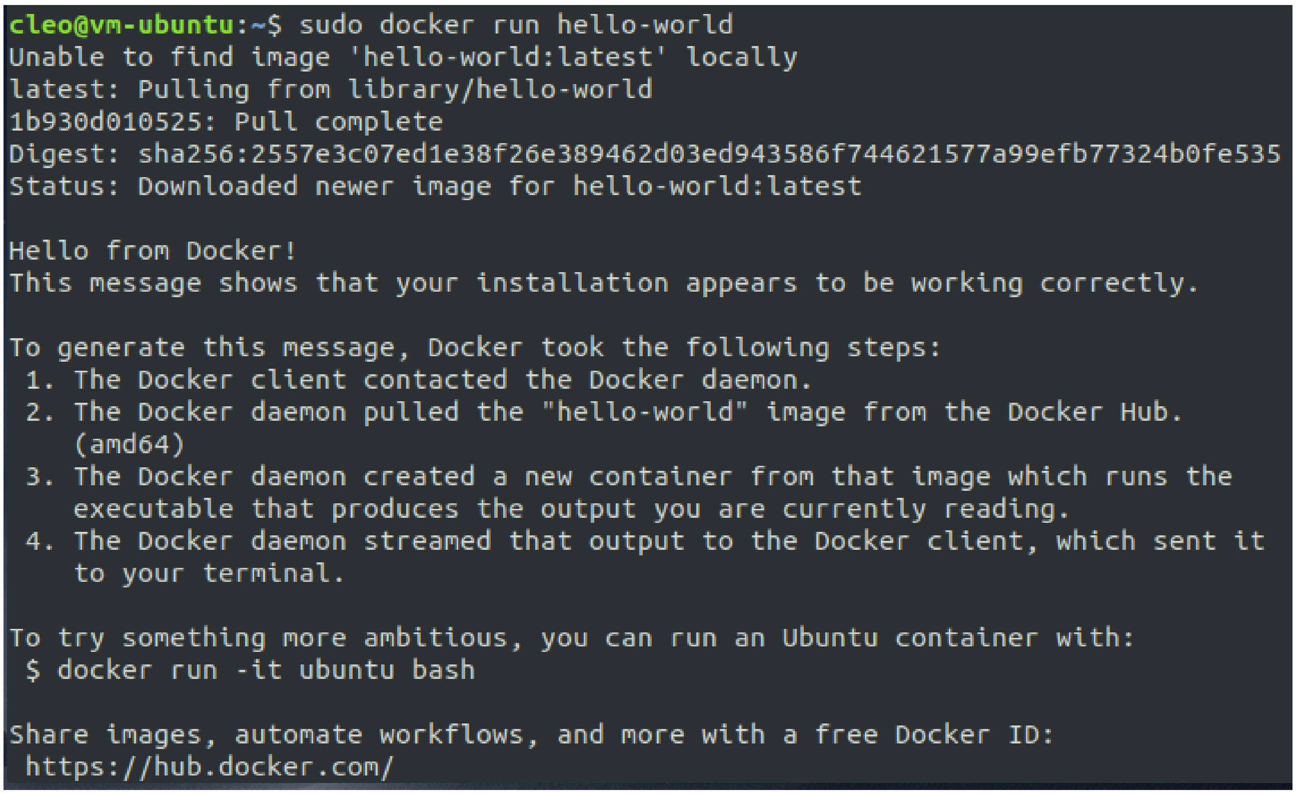
实验数据或结果

Ubuntu 18.04 LTS

Docker CE

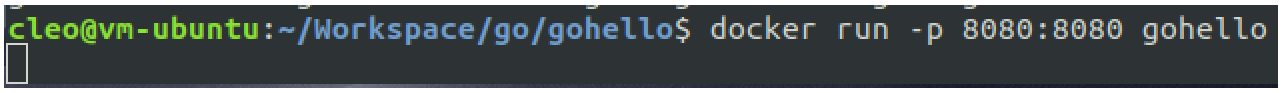


```
cleo@vm-ubuntu:~$ docker --version
Docker version 18.09.4, build d14af54266
cleo@vm-ubuntu:~$ 
```

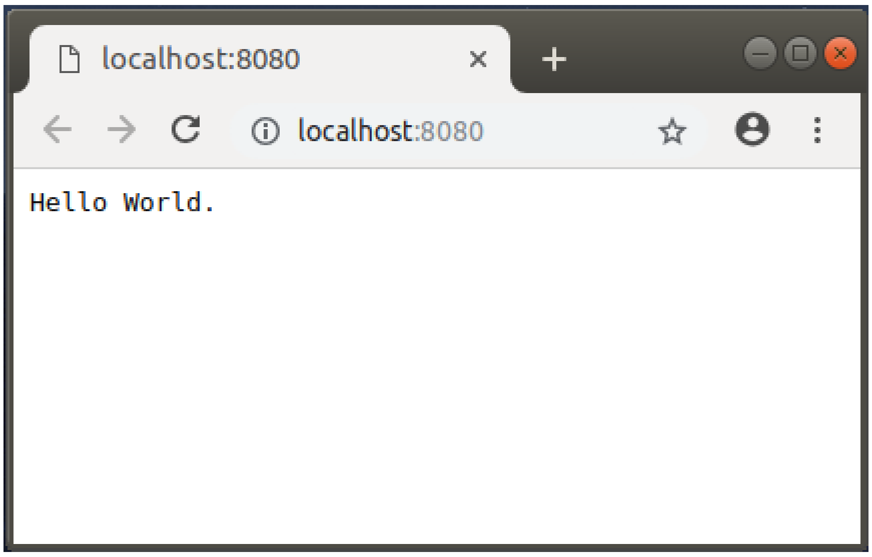


```
cleo@vm-ubuntu:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:2557e3c07ed1e38f26e389462d03ed943586f744621577a99efb77324b0fe535
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/
```

Hello World running in Docker by Go Programming Language



```
cleo@vm-ubuntu:~/Workspace/go/gohello$ docker run -p 8080:8080 gohello
```



```
Hello World.
```

实验思考

Docker is a computer program that performs operating-system-level virtualization. It is an instance of IaaS. Docker is used to run software packages called containers. Containers are isolated from each other and bundle their own application, tools, libraries and configuration

files; they can communicate with each other through well-defined channels. All containers are run by a single operating-system kernel and are thus more lightweight than virtual machines. Containers are created from images that specify their precise contents. Images are often created by combining and modifying standard images downloaded from public repositories. Docker is a tool that can package an application and its dependencies in a virtual container that can run on any Linux server. This helps enable flexibility and portability on where the application can run, whether on premises, public cloud, private cloud, bare metal, etc.

Here are some usual commands for docker:

```
# List Docker CLI commands
docker
docker container --help

# Display Docker version and info
docker --version
docker version
docker info

# Execute Docker image
docker run hello-world

# List Docker images
docker image ls

# List Docker containers (running, all, all in quiet mode)
docker container ls
docker container ls --all
docker container ls -aq

# Create image using this directory's Dockerfile
docker build -t friendlyhello .

# Run "friendlyname" mapping port 4000 to 80
docker run -p 4000:80 friendlyhello

# Same thing, but in detached mode
docker run -d -p 4000:80 friendlyhello

# Gracefully stop the specified container
docker container stop <hash>

# Force shutdown of the specified container
docker container kill <hash>

# Remove specified container from this machine
docker container rm <hash>

# Remove all containers
docker container rm $(docker container ls -a -q)

# List all images on this machine
```

```
docker image ls -a

# Remove specified image from this machine
docker image rm <image id>

# Remove all images from this machine
docker image rm $(docker image ls -a -q)

# Log in this CLI session using your Docker credentials
docker login

# Tag <image> for upload to registry
docker tag <image> username/repository:tag

# Upload tagged image to registry
docker push username/repository:tag

# Run image from a registry
docker run username/repository:tag
```

参考资料

- https://hub.docker.com
- https://docs.docker.com/
- https://en.wikipedia.org/wiki/Docker_(software)
- *The Go Programming Language*, Alan A. A. Donovan, Brian W. Kernighan, 2017