Docker Multi Stage Build And Deploying Stateless Go Application with Redis

[Edition 1]

[Last Update 201006]

For any issues/help contact : ***support@k21academy.com***

Contents

1 INTRODUCTION

Stateless applications are applications which do not store data or application state to the cluster or to persistent storage. Instead, data and application state stay with the client, which makes stateless applications more scalable.

Kubernetes uses the Deployment controller to deploy stateless applications as uniform, non-unique Pods. Deployments manage the *desired state* of your application: how many Pods should run your application, what version of the container image should run, what the Pods should be labelled, and so on.

This guide Covers:

Building Optimised Image for Docker Container – Multi Stage Dockerfile

- Git clone the Go App Code
- Building and Running the app locally
- Containerising the Go App - Building Docker Image
- Building Optimised Image – Multi Stage Builds

Deploying Stateless Go Application with Redis

- Git clone the Go App Code
- Containerising the Go App - Building Muilti Stage Docker Image
- Tag and Push image to Docker Private Registry
- Deploying the Go App and Redis to kubernetes Cluster
- Accessing the Go App

2 DOCUMENTATION

2.1 Kubernetes Documentation

1. Deploying WordPress and MySql
   https://kubernetes.io/docs/tutorials/stateful-application/mysql-wordpress-persistent-volume/
2. Single-Instance Stateful Application
   https://kubernetes.io/docs/tasks/run-application/run-single-instance-stateful-application/

2.2 Linux Commands and VIM Commands

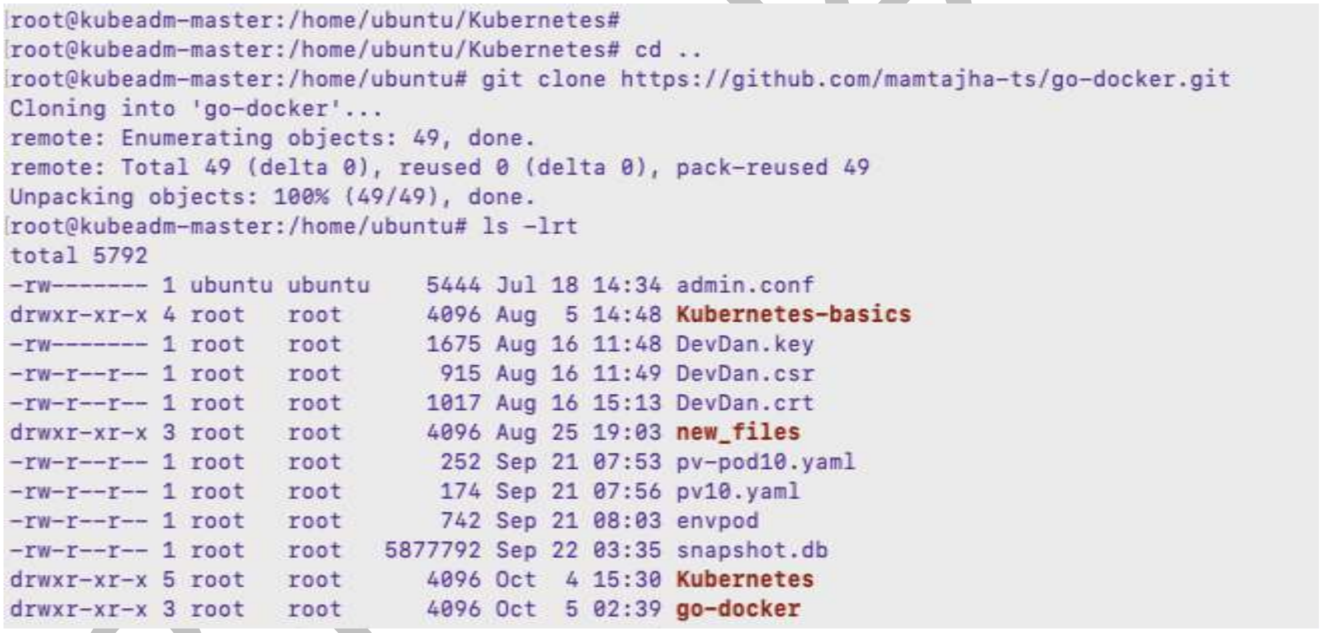
1. Basic Linux Commands
   https://maker.pro/linux/tutorial/basic-linux-commands-for-beginners
   https://www.hostinger.in/tutorials/linux-commands
2. Basic VIM Commands
   https://coderwall.com/p/adv71w/basic-vim-commands-for-getting-started
3. Popular VIM Commands
   https://www.keycdn.com/blog/vim-commands

3 BUILDING OPTIMISED IMAGE FOR DOCKER CONTAINER – MULTI STAGE DOCKERFILE

3.1 Git clone the Go App Code

1. Clone and get all the code files to the local server. Move out of the Kubernetes folder and execute the clone command

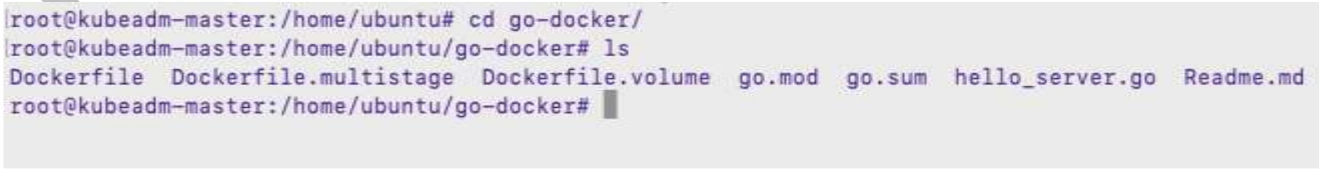
```
$ cd ..
$ git clone https://github.com/mamtajha-ts/go-docker.git
$ ls -lrt
```



```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# cd ..
root@kubeadm-master:/home/ubuntu# git clone https://github.com/mamtajha-ts/go-docker.git
Cloning into 'go-docker'...
remote: Enumerating objects: 49, done.
remote: Total 49 (delta 0), reused 0 (delta 0), pack-reused 49
Unpacking objects: 100% (49/49), done.
root@kubeadm-master:/home/ubuntu# ls -lrt
total 5792
-rw------- 1 ubuntu ubuntu    5444 Jul 18 14:34 admin.conf
drwxr-xr-x 4 root   root      4096 Aug  5 14:48 Kubernetes-basics
-rw------- 1 root   root      1675 Aug 16 11:48 DevDan.key
-rw-r--r-- 1 root   root       915 Aug 16 11:49 DevDan.csr
-rw-r--r-- 1 root   root      1017 Aug 16 15:13 DevDan.crt
drwxr-xr-x 3 root   root      4096 Aug 25 19:03 new_files
-rw-r--r-- 1 root   root       252 Sep 21 07:53 pv-pod10.yaml
-rw-r--r-- 1 root   root       174 Sep 21 07:56 pv10.yaml
-rw-r--r-- 1 root   root       742 Sep 21 08:03 envpod
-rw-r--r-- 1 root   root   5877792 Sep 22 03:35 snapshot.db
drwxr-xr-x 5 root   root      4096 Oct  4 15:30 Kubernetes
drwxr-xr-x 3 root   root      4096 Oct  5 02:39 go-docker
```

2. Move to the newly create code folder go-docker and list all the files

```
$ cd go-docker
$ ls
```

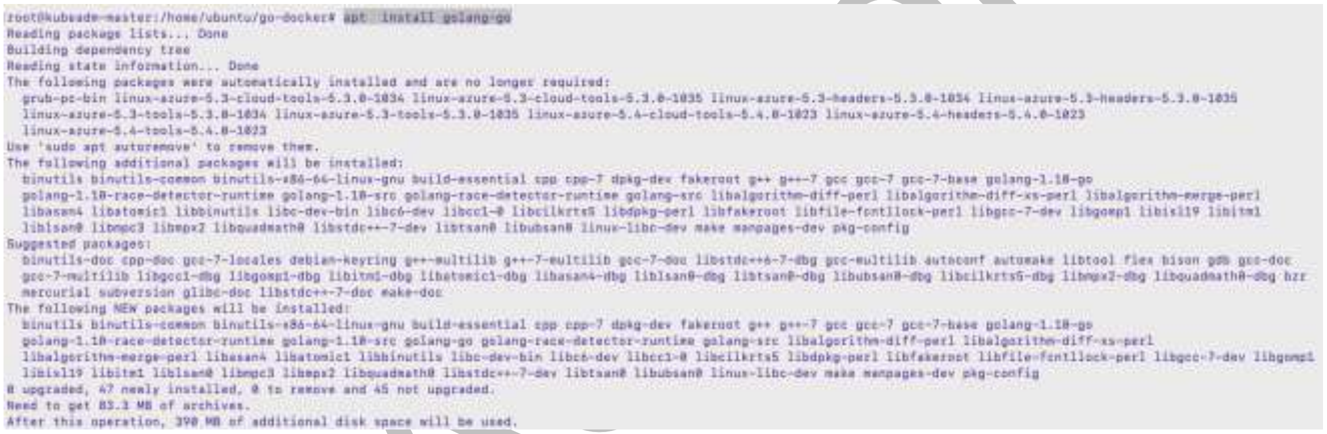


```
root@kubeadm-master:/home/ubuntu# cd go-docker/
root@kubeadm-master:/home/ubuntu/go-docker# ls
Dockerfile  Dockerfile.multistage  Dockerfile.volume  go.mod  go.sum  hello_server.go  Readme.md
root@kubeadm-master:/home/ubuntu/go-docker#
```

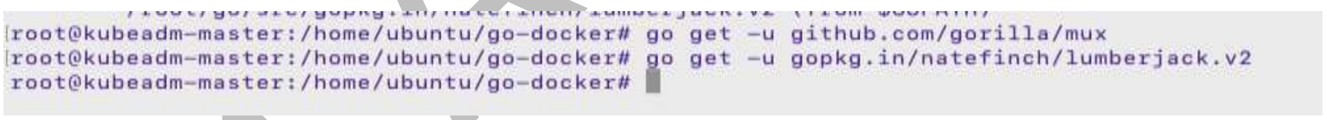
3.2 Building and Running the app locally

1. We will have to install go language and needed tools in the local server so that we can build and run our application locally

```
$ apt  install golang-go -y
$ go get -u github.com/gorilla/mux
$ go get -u gopkg.in/natefinch/lumberjack.v2
```



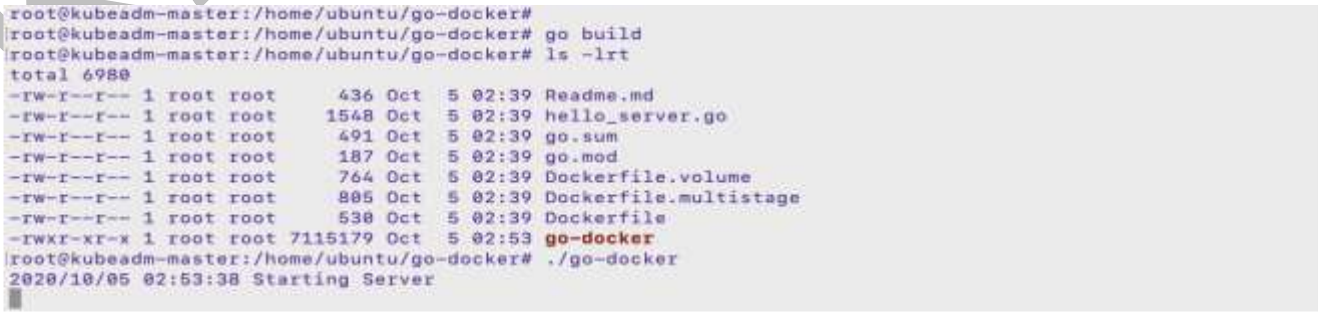
```
root@kubeadm-master:/home/ubuntu/go-docker# apt  install golang-go
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  grub-pc-bin linux-azure-5.3-cloud-tools-5.3.0-1034 linux-azure-5.3-cloud-tools-5.3.0-1035 linux-azure-5.3-headers-5.3.0-1034 linux-azure-5.3-headers-5.3.0-1035
  linux-azure-5.3-tools-5.3.0-1034 linux-azure-5.3-tools-5.3.0-1035 linux-azure-5.4-cloud-tools-5.4.0-1023 linux-azure-5.4-headers-5.4.0-1023
  linux-azure-5.4-tools-5.4.0-1023
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential cpp cpp-7 dpkg-dev fakeroot g++ g++-7 gcc gcc-7 gcc-7-base golang-1.10-go
  golang-1.10-race-detector-runtime golang-1.10-src golang-race-detector-runtime golang-src libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl
  libasan4 libatomic1 libbinutils libc-dev-bin libc6-dev libcc1-0 libcilkrts5 libdpkg-perl libfakeroot libfile-fcntllock-perl libgcc-7-dev libgomp1 libisl19 libitm1
  liblsan0 libmpc3 libmpx2 libquadmath0 libstdc++-7-dev libtsan0 libubsan0 linux-libc-dev make manpages-dev pkg-config
Suggested packages:
  binutils-doc cpp-doc gcc-7-locales debian-keyring g++-multilib g++-7-multilib gcc-7-doc libstdc++6-7-dbg gcc-multilib autoconf automake libtool flex bison gdb gcc-doc
  gcc-7-multilib libgcc1-dbg libgomp1-dbg libitm1-dbg libatomic1-dbg libasan4-dbg liblsan0-dbg libtsan0-dbg libubsan0-dbg libcilkrts5-dbg libmpx2-dbg libquadmath0-dbg bzr
  mercurial subversion glibc-doc libstdc++-7-doc make-doc
The following NEW packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential cpp cpp-7 dpkg-dev fakeroot g++ g++-7 gcc gcc-7 gcc-7-base golang-1.10-go
  golang-1.10-race-detector-runtime golang-1.10-src golang-go golang-race-detector-runtime golang-src libalgorithm-diff-perl libalgorithm-diff-xs-perl
  libalgorithm-merge-perl libasan4 libatomic1 libbinutils libc-dev-bin libc6-dev libcc1-0 libcilkrts5 libdpkg-perl libfakeroot libfile-fcntllock-perl libgcc-7-dev libgomp1
  libisl19 libitm1 liblsan0 libmpc3 libmpx2 libquadmath0 libstdc++-7-dev libtsan0 libubsan0 linux-libc-dev make manpages-dev pkg-config
0 upgraded, 47 newly installed, 0 to remove and 45 not upgraded.
Need to get 83.3 MB of archives.
After this operation, 390 MB of additional disk space will be used.
```



```
root@kubeadm-master:/home/ubuntu/go-docker# go get -u github.com/gorilla/mux
root@kubeadm-master:/home/ubuntu/go-docker# go get -u gopkg.in/natefinch/lumberjack.v2
root@kubeadm-master:/home/ubuntu/go-docker#
```

2. Let's first build and run our application locally. We will build the app, it will produce an executable file named go-docker. We can run the binary executable like so -

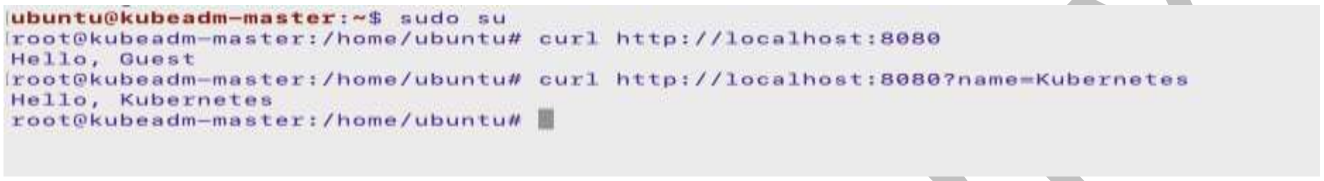
```
$ go build
$ ./go-docker
```



```
root@kubeadm-master:/home/ubuntu/go-docker#
root@kubeadm-master:/home/ubuntu/go-docker# go build
root@kubeadm-master:/home/ubuntu/go-docker# ls -lrt
total 6980
-rw-r--r-- 1 root root     436 Oct  5 02:39 Readme.md
-rw-r--r-- 1 root root    1548 Oct  5 02:39 hello_server.go
-rw-r--r-- 1 root root     491 Oct  5 02:39 go.sum
-rw-r--r-- 1 root root     187 Oct  5 02:39 go.mod
-rw-r--r-- 1 root root     764 Oct  5 02:39 Dockerfile.volume
-rw-r--r-- 1 root root     805 Oct  5 02:39 Dockerfile.multistage
-rw-r--r-- 1 root root     530 Oct  5 02:39 Dockerfile
-rwxr-xr-x 1 root root 7115179 Oct  5 02:53 go-docker
root@kubeadm-master:/home/ubuntu/go-docker# ./go-docker
2020/10/05 02:53:38 Starting Server
```

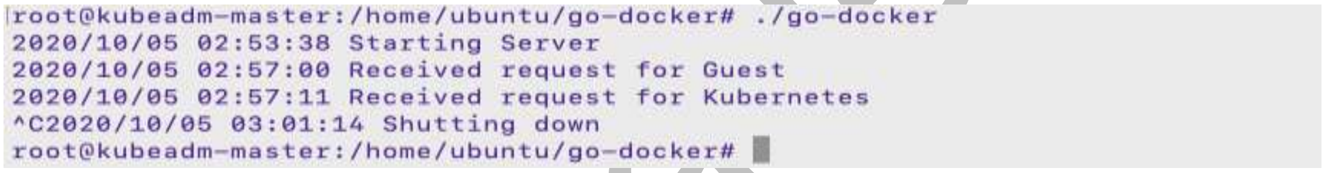
3. Our go hello server is now running. Try interacting with the hello server using curl – duplicate the putty session and execute curl command

```
$ curl http://localhost:8080

$ curl http://localhost:8080?name=Kubernetes
```



```
ubuntu@kubeadm-master:~$ sudo su
root@kubeadm-master:/home/ubuntu# curl http://localhost:8080
Hello, Guest
root@kubeadm-master:/home/ubuntu# curl http://localhost:8080?name=Kubernetes
Hello, Kubernetes
root@kubeadm-master:/home/ubuntu#
```

4. Go back to the previous session in which we have started the hello server and exit the process by executing ctrl c

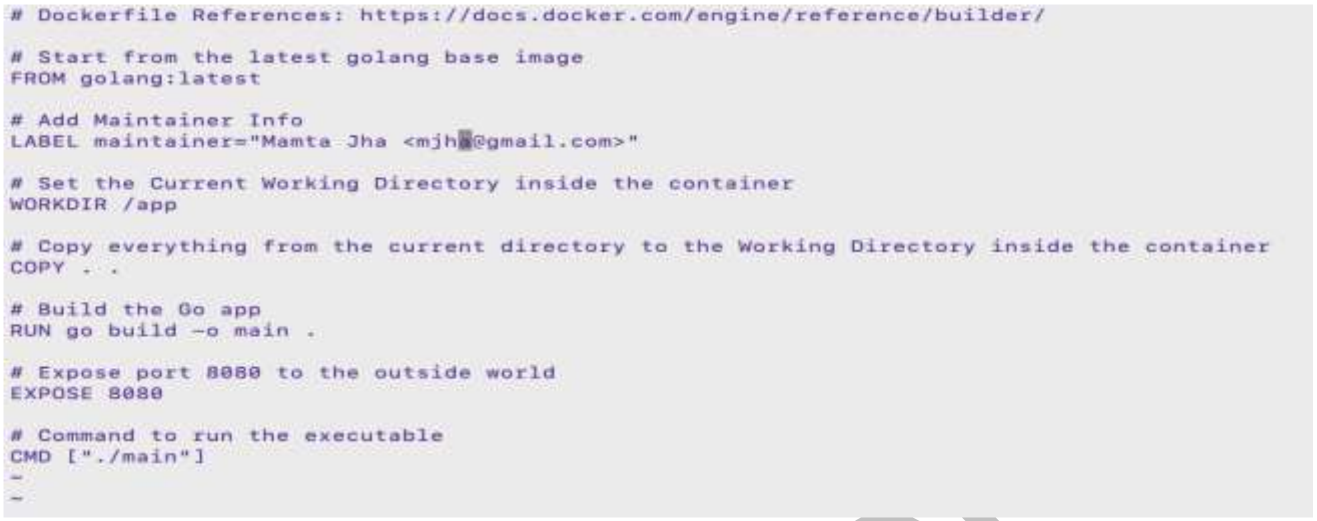


```
root@kubeadm-master:/home/ubuntu/go-docker# ./go-docker
2020/10/05 02:53:38 Starting Server
2020/10/05 02:57:00 Received request for Guest
2020/10/05 02:57:11 Received request for Kubernetes
^C2020/10/05 03:01:14 Shutting down
root@kubeadm-master:/home/ubuntu/go-docker#
```

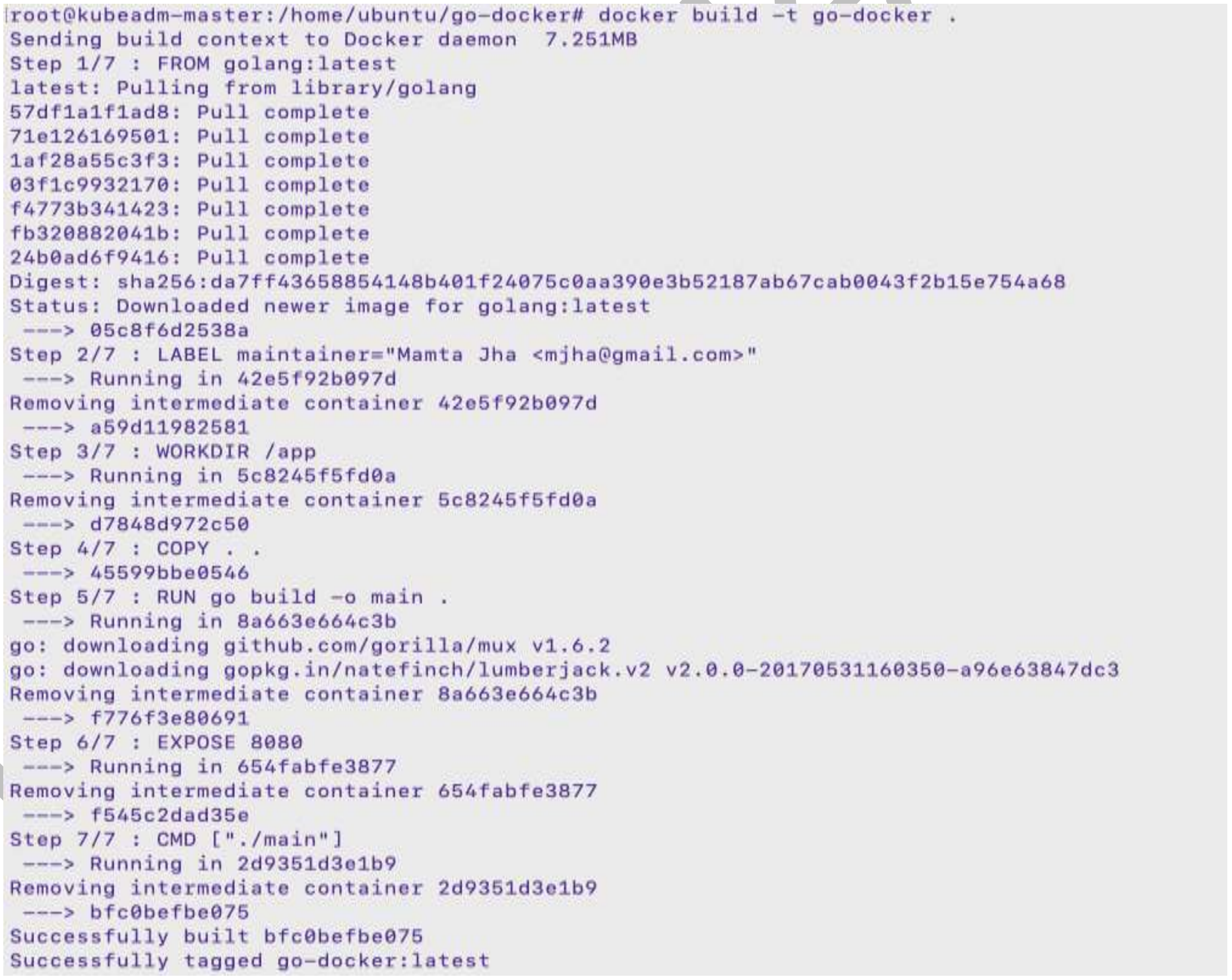
3.3 Containerising the Go App - Building Docker Image

1. Look into the content of the Dockerfile and use the Dockerfile to build the image

```
$ vi Dockerfile

$ docker build -t go-docker .
```



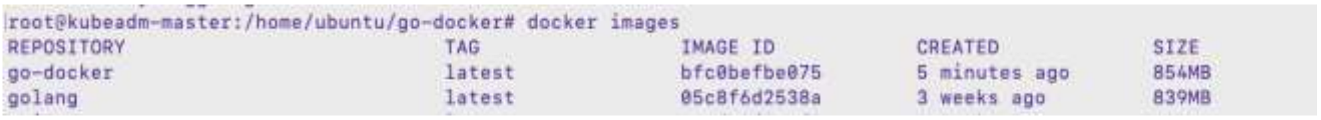
```
# Dockerfile References: https://docs.docker.com/engine/reference/builder/

# Start from the latest golang base image
FROM golang:latest

# Add Maintainer Info
LABEL maintainer="Mamta Jha <mjha@gmail.com>"

# Set the Current Working Directory inside the container
WORKDIR /app

# Copy everything from the current directory to the Working Directory inside the container
COPY . .

# Build the Go app
RUN go build -o main .

# Expose port 8080 to the outside world
EXPOSE 8080

# Command to run the executable
CMD ["./main"]
~
~
```



```
root@kubeadm-master:/home/ubuntu/go-docker# docker build -t go-docker .
Sending build context to Docker daemon  7.251MB
Step 1/7 : FROM golang:latest
latest: Pulling from library/golang
57df1a1f1ad8: Pull complete
71e126169501: Pull complete
1af28a55c3f3: Pull complete
03f1c9932170: Pull complete
f4773b341423: Pull complete
fb320882041b: Pull complete
24b0ad6f9416: Pull complete
Digest: sha256:da7ff43658854148b401f24075c0aa390e3b52187ab67cab0043f2b15e754a68
Status: Downloaded newer image for golang:latest
 ---> 05c8f6d2538a
Step 2/7 : LABEL maintainer="Mamta Jha <mjha@gmail.com>"
 ---> Running in 42e5f92b097d
Removing intermediate container 42e5f92b097d
 ---> a59d11982581
Step 3/7 : WORKDIR /app
 ---> Running in 5c8245f5fd0a
Removing intermediate container 5c8245f5fd0a
 ---> d7848d972c50
Step 4/7 : COPY . .
 ---> 45599bbe0546
Step 5/7 : RUN go build -o main .
 ---> Running in 8a663e664c3b
go: downloading github.com/gorilla/mux v1.6.2
go: downloading gopkg.in/natefinch/lumberjack.v2 v2.0.0-20170531160350-a96e63847dc3
Removing intermediate container 8a663e664c3b
 ---> f776f3e80691
Step 6/7 : EXPOSE 8080
 ---> Running in 654fabfe3877
Removing intermediate container 654fabfe3877
 ---> f545c2dad35e
Step 7/7 : CMD ["./main"]
 ---> Running in 2d9351d3e1b9
Removing intermediate container 2d9351d3e1b9
 ---> bfc0befbe075
Successfully built bfc0befbe075
Successfully tagged go-docker:latest
```

2. List and verify the newly created image

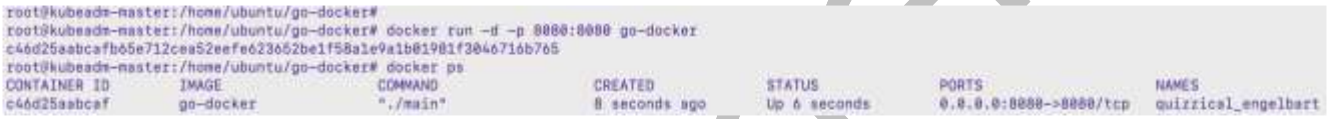
```
$ docker images
```



```
root@kubeadm-master:/home/ubuntu/go-docker# docker images
REPOSITORY                              TAG                 IMAGE ID            CREATED             SIZE
go-docker                               latest              bfc0befbe075        5 minutes ago       854MB
golang                                  latest              05c8f6d2538a        3 weeks ago         839MB
```

3. Use the above created image to create container using docker run command to verify the application is containerised properly

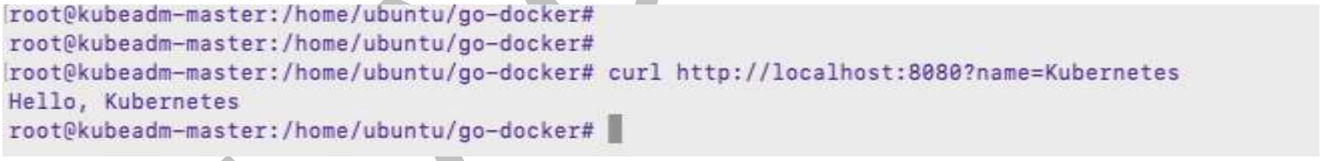
```
$ docker run -d -p 8080:8080 go-docker
$ docker ps
```



```
root@kubeadm-master:/home/ubuntu/go-docker#
root@kubeadm-master:/home/ubuntu/go-docker# docker run -d -p 8080:8080 go-docker
c46d25aabcafb65e712cea52eefe623652be1f58a1e9a1b01901f3046716b765
root@kubeadm-master:/home/ubuntu/go-docker# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS                    NAMES
c46d25aabcaf        go-docker           "./main"            8 seconds ago       Up 6 seconds        0.0.0.0:8080->8080/tcp   quizzical_engelbart
```

4. List the container to verify the status and if its in running state try Interacting with the app running inside the container

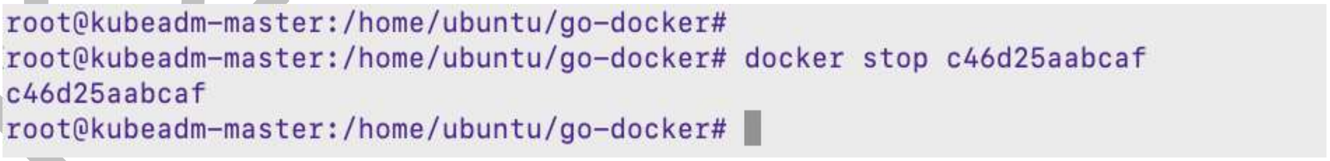
```
$ curl http://localhost:8080?name=Kubernetes
```



```
root@kubeadm-master:/home/ubuntu/go-docker#
root@kubeadm-master:/home/ubuntu/go-docker#
root@kubeadm-master:/home/ubuntu/go-docker# curl http://localhost:8080?name=Kubernetes
Hello, Kubernetes
root@kubeadm-master:/home/ubuntu/go-docker#
```

5. Stop the container using the container id as mentioned in your server

```
$ docker stop <container id>
```

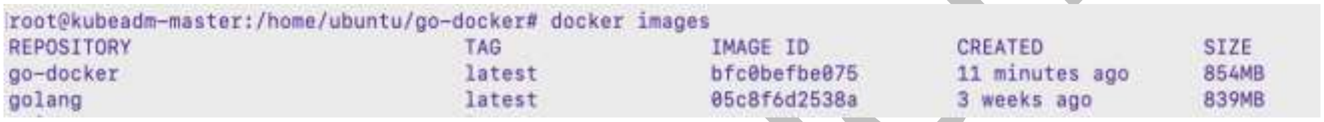


```
root@kubeadm-master:/home/ubuntu/go-docker#
root@kubeadm-master:/home/ubuntu/go-docker# docker stop c46d25aabcaf
c46d25aabcaf
root@kubeadm-master:/home/ubuntu/go-docker#
```

3.4 Building Optimised Image – Multi Stage Builds

1. Look into the size of the docker image created in the above task. The golang:latest image that we're using as our base is **839MB** in size, and our application images are **854MB** in size

```
$ docker images
```



```
root@kubeadm-master:/home/ubuntu/go-docker# docker images
REPOSITORY                          TAG         IMAGE ID        CREATED           SIZE
go-docker                           latest      bfc0befbe075    11 minutes ago    854MB
golang                              latest      05c8f6d2538a    3 weeks ago       839MB
```

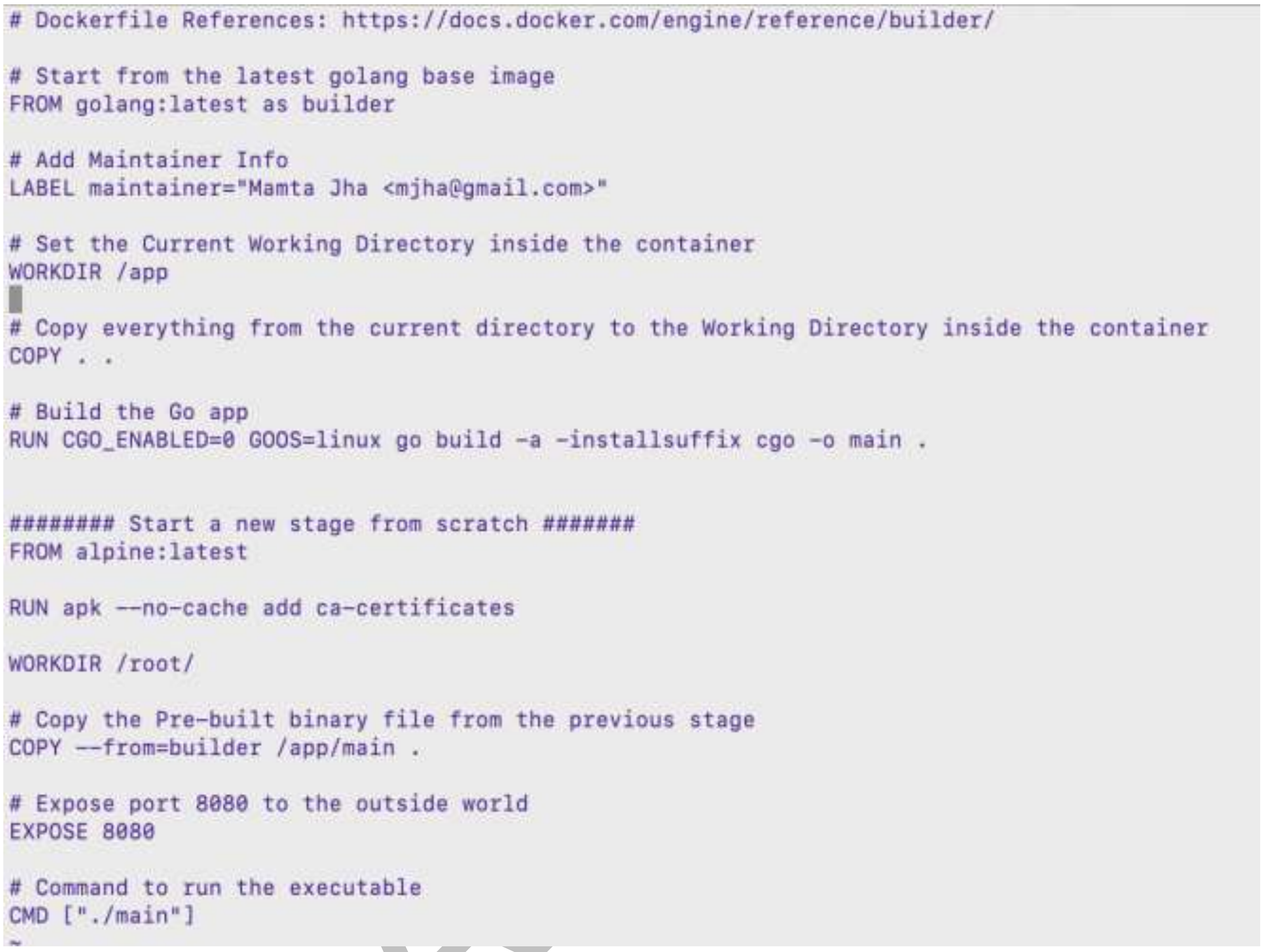
2. To reduce the size of the docker image, we can use a multi-stage build. The first stage of the multi-stage build will use the golang:latest image and build our application.

3. The second stage will use a very lightweight Alpine linux image and will only contain the binary executable built by the first stage

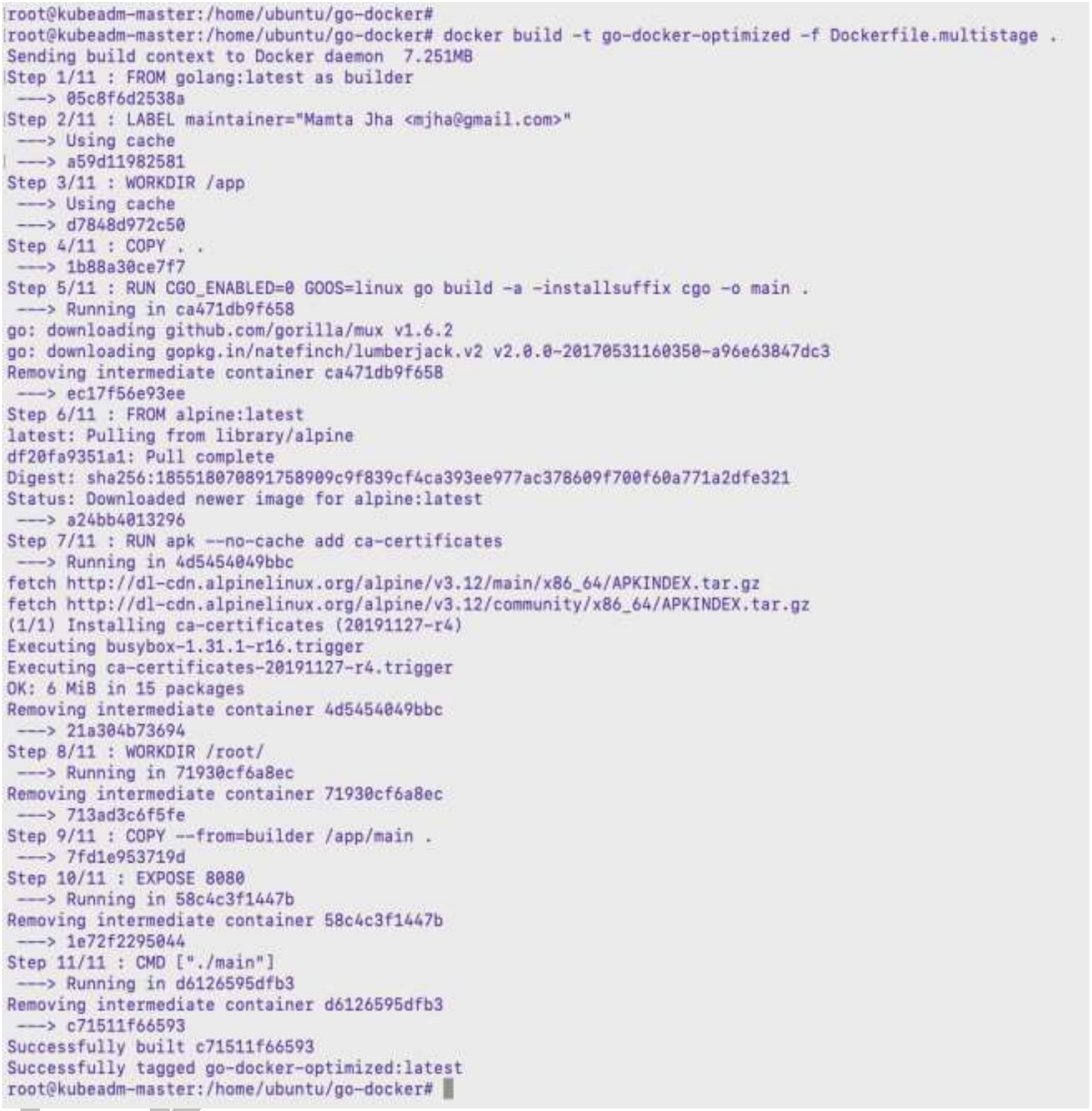
4. This way, our final image will be very small because It won't have all the Golang runtime. It will only contain the things needed to run the binary executable

5. Look into the content of the Dockerfile.multistage and build new image using the multi stage Dockerfile

```
$ vi Dockerfile.multistage

$ docker build -t go-docker-optimized -f Dockerfile.multistage .
```



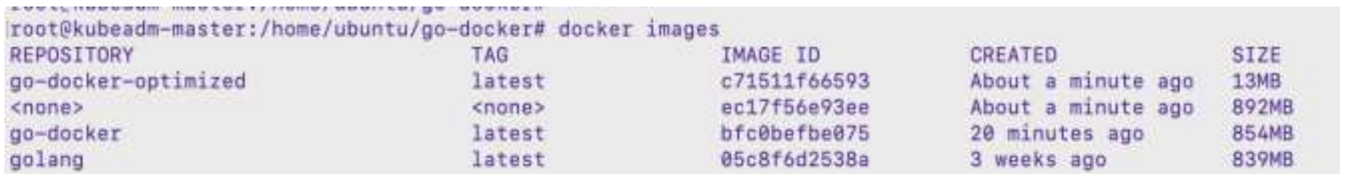
```
# Dockerfile References: https://docs.docker.com/engine/reference/builder/

# Start from the latest golang base image
FROM golang:latest as builder

# Add Maintainer Info
LABEL maintainer="Mamta Jha <mjha@gmail.com>"

# Set the Current Working Directory inside the container
WORKDIR /app

# Copy everything from the current directory to the Working Directory inside the container
COPY . .

# Build the Go app
RUN CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o main .


######## Start a new stage from scratch #######
FROM alpine:latest

RUN apk --no-cache add ca-certificates

WORKDIR /root/

# Copy the Pre-built binary file from the previous stage
COPY --from=builder /app/main .

# Expose port 8080 to the outside world
EXPOSE 8080

# Command to run the executable
CMD ["./main"]
```



```
root@kubeadm-master:/home/ubuntu/go-docker#
root@kubeadm-master:/home/ubuntu/go-docker# docker build -t go-docker-optimized -f Dockerfile.multistage .
Sending build context to Docker daemon  7.251MB
Step 1/11 : FROM golang:latest as builder
 ---> 05c8f6d2538a
Step 2/11 : LABEL maintainer="Mamta Jha <mjha@gmail.com>"
 ---> Using cache
 ---> a59d11982581
Step 3/11 : WORKDIR /app
 ---> Using cache
 ---> d7848d972c50
Step 4/11 : COPY . .
 ---> 1b88a30ce7f7
Step 5/11 : RUN CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o main .
 ---> Running in ca471db9f658
go: downloading github.com/gorilla/mux v1.6.2
go: downloading gopkg.in/natefinch/lumberjack.v2 v2.0.0-20170531160350-a96e63847dc3
Removing intermediate container ca471db9f658
 ---> ec17f56e93ee
Step 6/11 : FROM alpine:latest
latest: Pulling from library/alpine
df20fa9351a1: Pull complete
Digest: sha256:185518070891758909c9f839cf4ca393ee977ac378609f700f60a771a2dfe321
Status: Downloaded newer image for alpine:latest
 ---> a24bb4013296
Step 7/11 : RUN apk --no-cache add ca-certificates
 ---> Running in 4d5454049bbc
fetch http://dl-cdn.alpinelinux.org/alpine/v3.12/main/x86_64/APKINDEX.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.12/community/x86_64/APKINDEX.tar.gz
(1/1) Installing ca-certificates (20191127-r4)
Executing busybox-1.31.1-r16.trigger
Executing ca-certificates-20191127-r4.trigger
OK: 6 MiB in 15 packages
Removing intermediate container 4d5454049bbc
 ---> 21a304b73694
Step 8/11 : WORKDIR /root/
 ---> Running in 71930cf6a8ec
Removing intermediate container 71930cf6a8ec
 ---> 713ad3c6f5fe
Step 9/11 : COPY --from=builder /app/main .
 ---> 7fd1e953719d
Step 10/11 : EXPOSE 8080
 ---> Running in 58c4c3f1447b
Removing intermediate container 58c4c3f1447b
 ---> 1e72f2295044
Step 11/11 : CMD ["./main"]
 ---> Running in d6126595dfb3
Removing intermediate container d6126595dfb3
 ---> c71511f66593
Successfully built c71511f66593
Successfully tagged go-docker-optimized:latest
root@kubeadm-master:/home/ubuntu/go-docker#
```

6. Now let's see the size of the image - Wow! Our optimized image is only **13MB** in size. That's awesome!

```
$ docker images
```



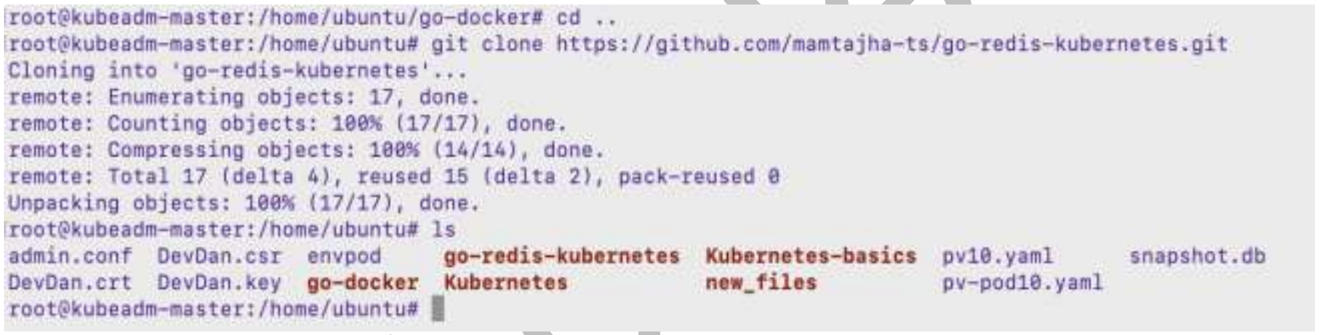
```
root@kubeadm-master:/home/ubuntu/go-docker# docker images
REPOSITORY                    TAG          IMAGE ID       CREATED              SIZE
go-docker-optimized           latest       c71511f66593   About a minute ago   13MB
<none>                        <none>       ec17f56e93ee   About a minute ago   892MB
go-docker                     latest       bfc0befbe075   20 minutes ago       854MB
golang                        latest       05c8f6d2538a   3 weeks ago          839MB
```

4 DEPLOYING STATELESS GO APPLICATION WITH REDIS

4.1 Git clone the Go App Code

1. We'll create a simple web application in Go that contains an API to display the "Quote of the day"
2. Clone and get all the code files to the local server. Move out of the Kubernetes folder and execute the clone command

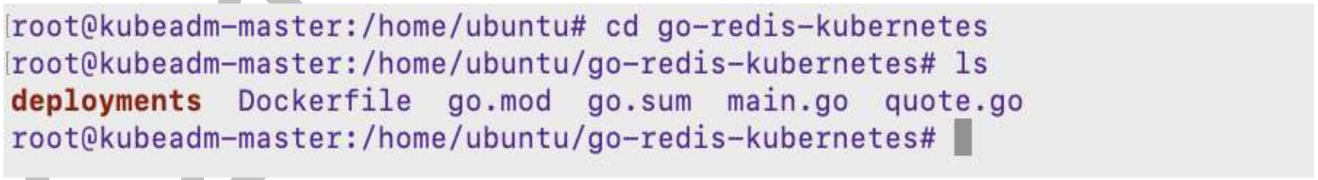
```
$ cd ..
$ git clone https://github.com/mamtajha-ts/go-redis-kubernetes.git
$ ls -lrt
```



```
root@kubeadm-master:/home/ubuntu/go-docker# cd ..
root@kubeadm-master:/home/ubuntu# git clone https://github.com/mamtajha-ts/go-redis-kubernetes.git
Cloning into 'go-redis-kubernetes'...
remote: Enumerating objects: 17, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 17 (delta 4), reused 15 (delta 2), pack-reused 0
Unpacking objects: 100% (17/17), done.
root@kubeadm-master:/home/ubuntu# ls
admin.conf  DevDan.csr  envpod     go-redis-kubernetes  Kubernetes-basics  pv10.yaml     snapshot.db
DevDan.crt  DevDan.key  go-docker  Kubernetes           new_files          pv-pod10.yaml
root@kubeadm-master:/home/ubuntu#
```

3. Move to the newly create code folder go-redis-kubernetes and view all the files

```
$ cd go-redis-kubernetes
$ ls
```

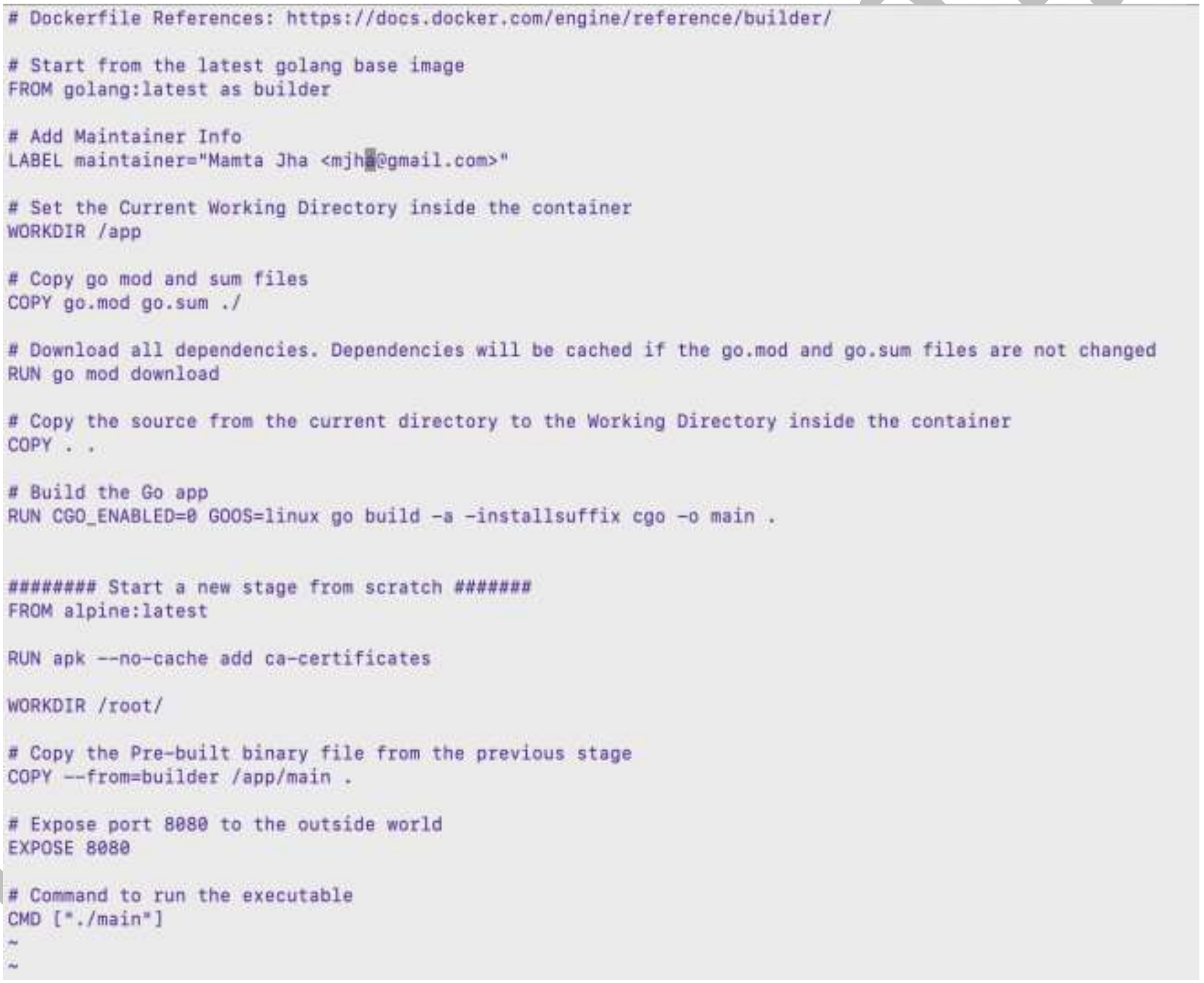


```
root@kubeadm-master:/home/ubuntu# cd go-redis-kubernetes
root@kubeadm-master:/home/ubuntu/go-redis-kubernetes# ls
deployments  Dockerfile  go.mod  go.sum  main.go  quote.go
root@kubeadm-master:/home/ubuntu/go-redis-kubernetes#
```

4.2 Containerising the Go App - Building Muilti Stage Docker Image

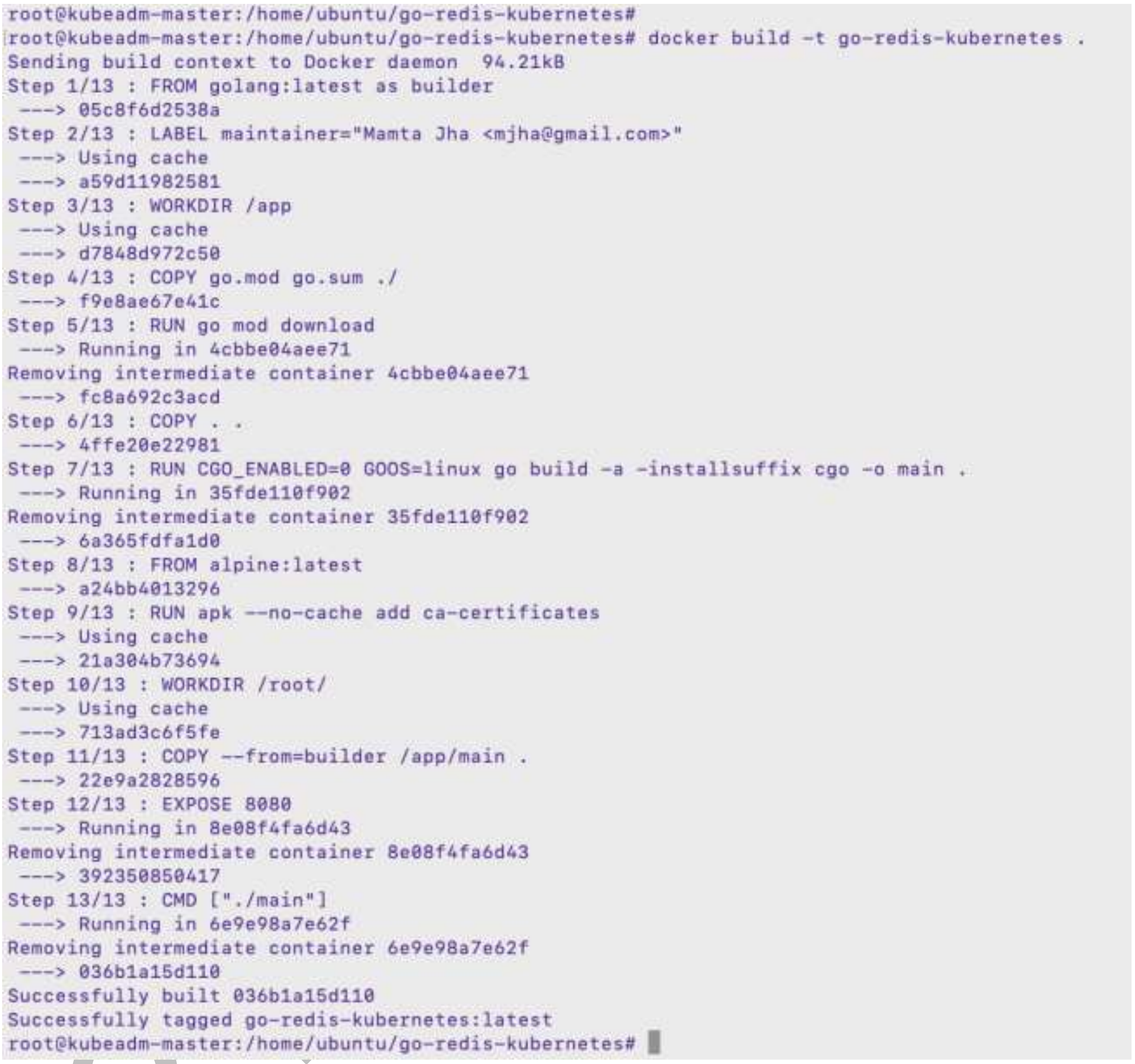
1. Look into the content of the Dockerfile and use the Dockerfile to build the image

```
$ vi Dockerfile
```



```
# Dockerfile References: https://docs.docker.com/engine/reference/builder/

# Start from the latest golang base image
FROM golang:latest as builder

# Add Maintainer Info
LABEL maintainer="Mamta Jha <mjha@gmail.com>"

# Set the Current Working Directory inside the container
WORKDIR /app

# Copy go mod and sum files
COPY go.mod go.sum ./

# Download all dependencies. Dependencies will be cached if the go.mod and go.sum files are not changed
RUN go mod download

# Copy the source from the current directory to the Working Directory inside the container
COPY . .

# Build the Go app
RUN CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o main .


######## Start a new stage from scratch #######
FROM alpine:latest

RUN apk --no-cache add ca-certificates

WORKDIR /root/

# Copy the Pre-built binary file from the previous stage
COPY --from=builder /app/main .

# Expose port 8080 to the outside world
EXPOSE 8080

# Command to run the executable
CMD ["./main"]
~
~
```

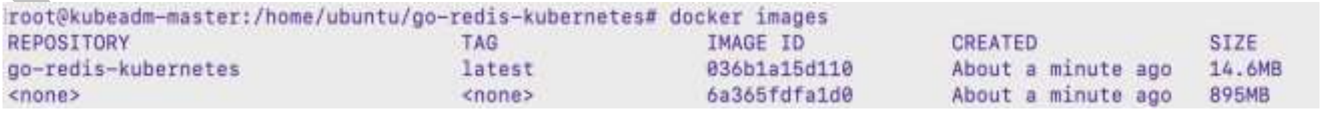
```
$ docker build -t go-redis-kubernetes .
```



```
root@kubeadm-master:/home/ubuntu/go-redis-kubernetes#
root@kubeadm-master:/home/ubuntu/go-redis-kubernetes# docker build -t go-redis-kubernetes .
Sending build context to Docker daemon  94.21kB
Step 1/13 : FROM golang:latest as builder
 ---> 05c8f6d2538a
Step 2/13 : LABEL maintainer="Mamta Jha <mjha@gmail.com>"
 ---> Using cache
 ---> a59d11982581
Step 3/13 : WORKDIR /app
 ---> Using cache
 ---> d7848d972c50
Step 4/13 : COPY go.mod go.sum ./
 ---> f9e8ae67e41c
Step 5/13 : RUN go mod download
 ---> Running in 4cbbe04aee71
Removing intermediate container 4cbbe04aee71
 ---> fc8a692c3acd
Step 6/13 : COPY . .
 ---> 4ffe20e22981
Step 7/13 : RUN CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o main .
 ---> Running in 35fde110f902
Removing intermediate container 35fde110f902
 ---> 6a365fdfa1d0
Step 8/13 : FROM alpine:latest
 ---> a24bb4013296
Step 9/13 : RUN apk --no-cache add ca-certificates
 ---> Using cache
 ---> 21a304b73694
Step 10/13 : WORKDIR /root/
 ---> Using cache
 ---> 713ad3c6f5fe
Step 11/13 : COPY --from=builder /app/main .
 ---> 22e9a2828596
Step 12/13 : EXPOSE 8080
 ---> Running in 8e08f4fa6d43
Removing intermediate container 8e08f4fa6d43
 ---> 392350850417
Step 13/13 : CMD ["./main"]
 ---> Running in 6e9e98a7e62f
Removing intermediate container 6e9e98a7e62f
 ---> 036b1a15d110
Successfully built 036b1a15d110
Successfully tagged go-redis-kubernetes:latest
root@kubeadm-master:/home/ubuntu/go-redis-kubernetes#
```

6. List and verify the newly created image

```
$ docker images
```

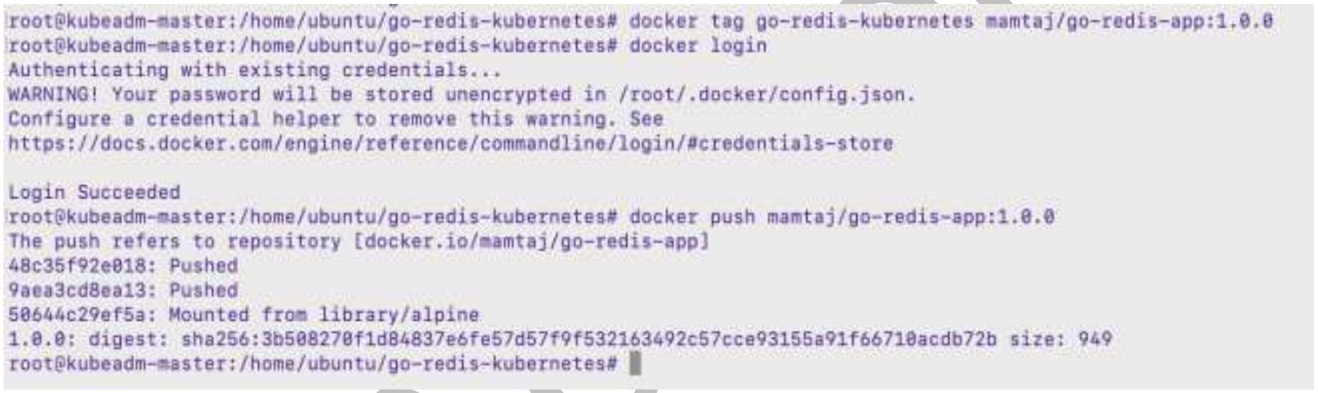


```
root@kubeadm-master:/home/ubuntu/go-redis-kubernetes# docker images
REPOSITORY                          TAG                 IMAGE ID            CREATED              SIZE
go-redis-kubernetes                 latest              036b1a15d110        About a minute ago   14.6MB
<none>                              <none>              6a365fdfa1d0        About a minute ago   895MB
```

4.3 Tag and Push image to Docker Private Registry

1. Tag the image to push it to the Docker Private Registry. Remember to tag the image as per your Docker ID and push it to your Docker private registry

```
$ docker tag go-redis-kubernetes mamtaj/go-redis-app:1.0.0
$ docker login
$ docker push mamtaj/go-redis-app:1.0.0
```

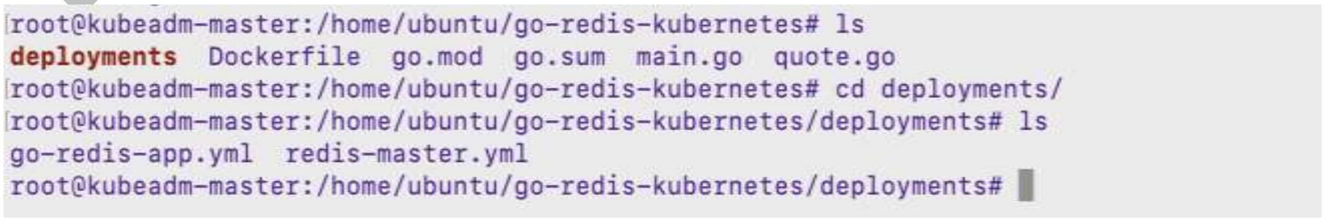


```
root@kubeadm-master:/home/ubuntu/go-redis-kubernetes# docker tag go-redis-kubernetes mamtaj/go-redis-app:1.0.0
root@kubeadm-master:/home/ubuntu/go-redis-kubernetes# docker login
Authenticating with existing credentials...
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@kubeadm-master:/home/ubuntu/go-redis-kubernetes# docker push mamtaj/go-redis-app:1.0.0
The push refers to repository [docker.io/mamtaj/go-redis-app]
48c35f92e018: Pushed
9aea3cd8ea13: Pushed
50644c29ef5a: Mounted from library/alpine
1.0.0: digest: sha256:3b508270f1d84837e6fe57d57f9f532163492c57cce93155a91f66710acdb72b size: 949
root@kubeadm-master:/home/ubuntu/go-redis-kubernetes# 
```

4.4 Deploying the Go App and Redis to kubernetes Cluster

1. All Kubernetes manifest files are available in deployments folder. Look into the content of the folder

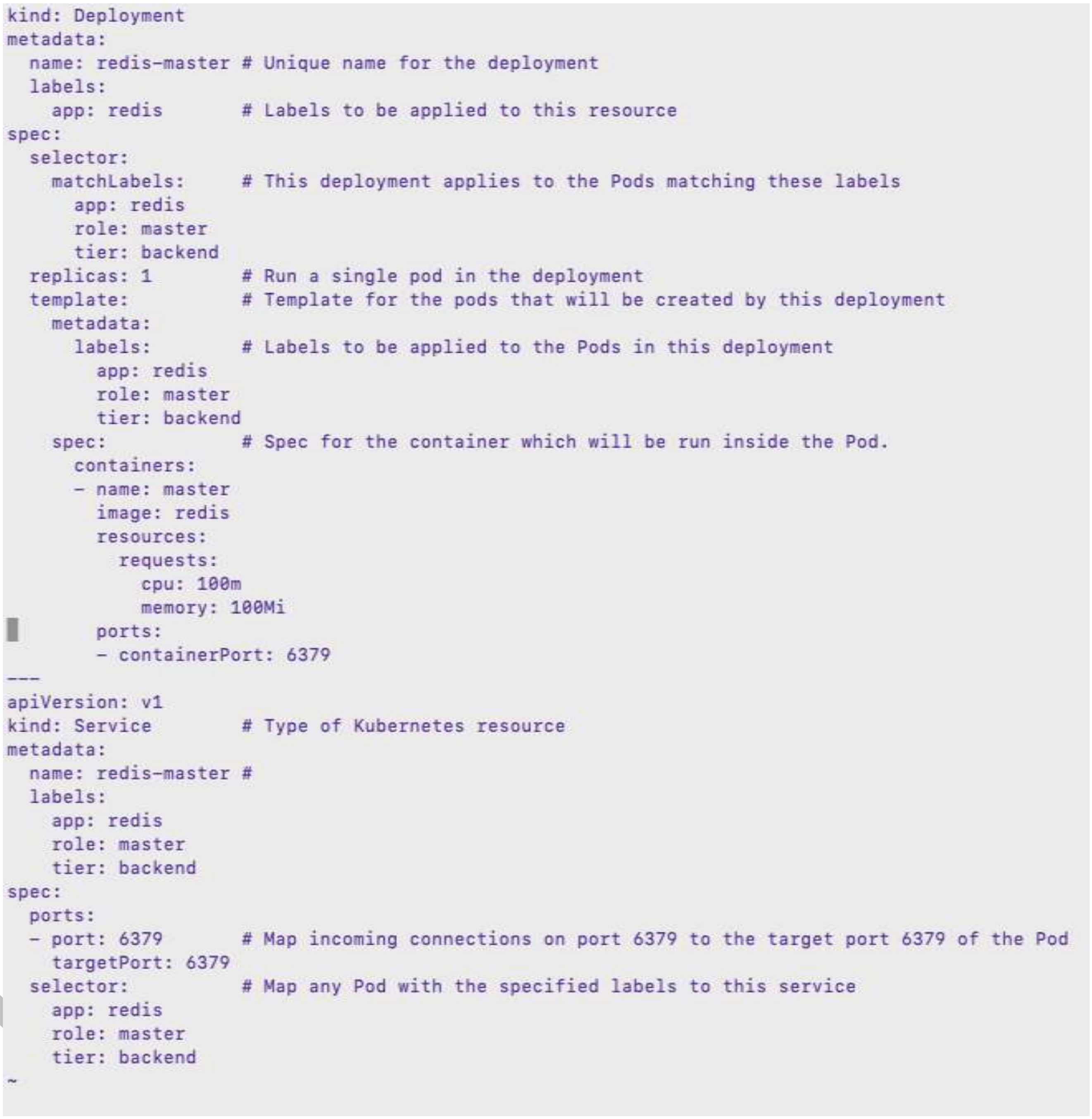
```
$ cd deployments
$ ls
```



```
root@kubeadm-master:/home/ubuntu/go-redis-kubernetes# ls
deployments  Dockerfile  go.mod  go.sum  main.go  quote.go
root@kubeadm-master:/home/ubuntu/go-redis-kubernetes# cd deployments/
root@kubeadm-master:/home/ubuntu/go-redis-kubernetes/deployments# ls
go-redis-app.yml  redis-master.yml
root@kubeadm-master:/home/ubuntu/go-redis-kubernetes/deployments# 
```

2. Use redis-master.yml file to create redis deployment and service

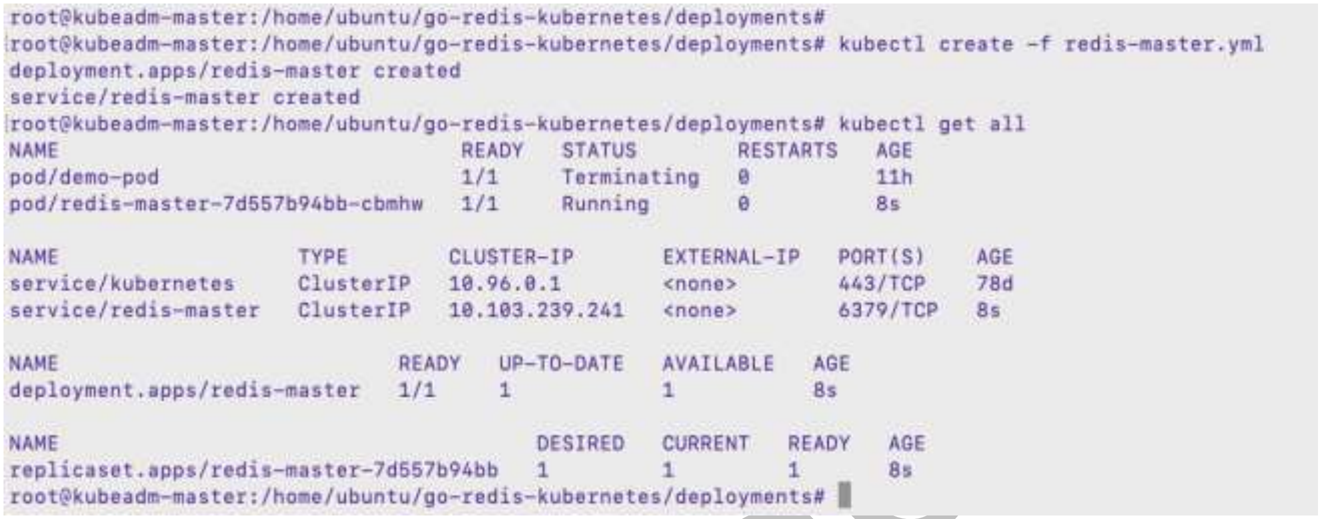
```
$ vi redis-master.yml
```



```
kind: Deployment
metadata:
  name: redis-master # Unique name for the deployment
  labels:
    app: redis       # Labels to be applied to this resource
spec:
  selector:
    matchLabels:     # This deployment applies to the Pods matching these labels
      app: redis
      role: master
      tier: backend
  replicas: 1        # Run a single pod in the deployment
  template:          # Template for the pods that will be created by this deployment
    metadata:
      labels:        # Labels to be applied to the Pods in this deployment
        app: redis
        role: master
        tier: backend
    spec:            # Spec for the container which will be run inside the Pod.
      containers:
      - name: master
        image: redis
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
        ports:
        - containerPort: 6379
---
apiVersion: v1
kind: Service        # Type of Kubernetes resource
metadata:
  name: redis-master #
  labels:
    app: redis
    role: master
    tier: backend
spec:
  ports:
  - port: 6379       # Map incoming connections on port 6379 to the target port 6379 of the Pod
    targetPort: 6379
  selector:          # Map any Pod with the specified labels to this service
    app: redis
    role: master
    tier: backend
~
```

3. Create and view the resources created using the redis-master.yml manifest file

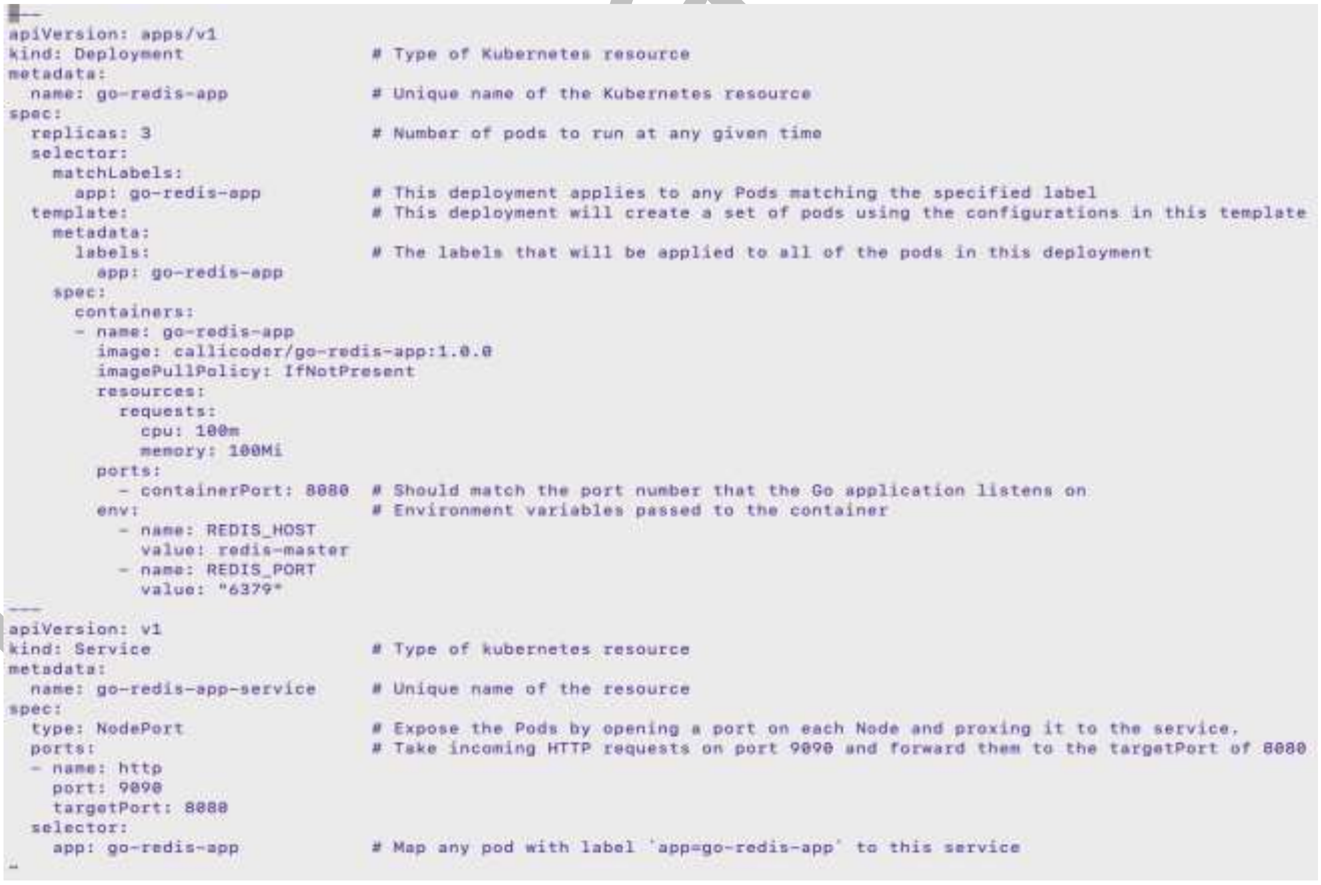
```
$ kubectl create -f redis-master.yml
$ kubectl get all
```



```
root@kubeadm-master:/home/ubuntu/go-redis-kubernetes/deployments#
root@kubeadm-master:/home/ubuntu/go-redis-kubernetes/deployments# kubectl create -f redis-master.yml
deployment.apps/redis-master created
service/redis-master created
root@kubeadm-master:/home/ubuntu/go-redis-kubernetes/deployments# kubectl get all
NAME                                READY   STATUS        RESTARTS   AGE
pod/demo-pod                        1/1     Terminating   0          11h
pod/redis-master-7d557b94bb-cbmhw   1/1     Running       0          8s

NAME                   TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
service/kubernetes     ClusterIP   10.96.0.1        <none>        443/TCP    78d
service/redis-master   ClusterIP   10.103.239.241   <none>        6379/TCP   8s

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/redis-master   1/1     1            1           8s

NAME                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/redis-master-7d557b94bb   1         1         1       8s
root@kubeadm-master:/home/ubuntu/go-redis-kubernetes/deployments#
```

4. Use go-redis-app.yml file to create redis deployment and service

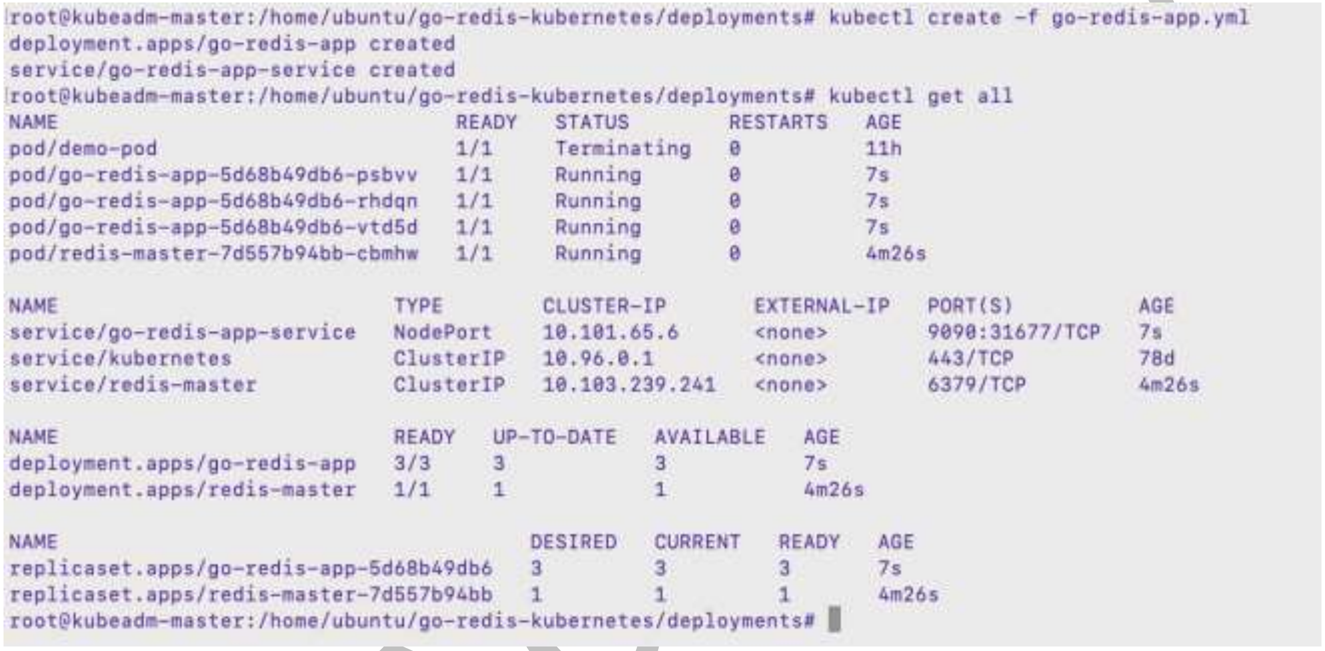
```
$ vi go-redis-app.yml
```



```
---
apiVersion: apps/v1
kind: Deployment                 # Type of Kubernetes resource
metadata:
  name: go-redis-app             # Unique name of the Kubernetes resource
spec:
  replicas: 3                    # Number of pods to run at any given time
  selector:
    matchLabels:
      app: go-redis-app          # This deployment applies to any Pods matching the specified label
  template:                      # This deployment will create a set of pods using the configurations in this template
    metadata:
      labels:                    # The labels that will be applied to all of the pods in this deployment
        app: go-redis-app
    spec:
      containers:
      - name: go-redis-app
        image: callicoder/go-redis-app:1.0.0
        imagePullPolicy: IfNotPresent
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
        ports:
          - containerPort: 8080  # Should match the port number that the Go application listens on
        env:                     # Environment variables passed to the container
          - name: REDIS_HOST
            value: redis-master
          - name: REDIS_PORT
            value: "6379"
---
apiVersion: v1
kind: Service                    # Type of kubernetes resource
metadata:
  name: go-redis-app-service     # Unique name of the resource
spec:
  type: NodePort                 # Expose the Pods by opening a port on each Node and proxing it to the service.
  ports:                         # Take incoming HTTP requests on port 9090 and forward them to the targetPort of 8080
  - name: http
    port: 9090
    targetPort: 8080
  selector:
    app: go-redis-app            # Map any pod with label `app=go-redis-app` to this service
~
```

5. Create and view the resources created using the go-redis-app.yml manifest file

```
$ kubectl create -f go-redis-app.yml
$ kubectl get all
```

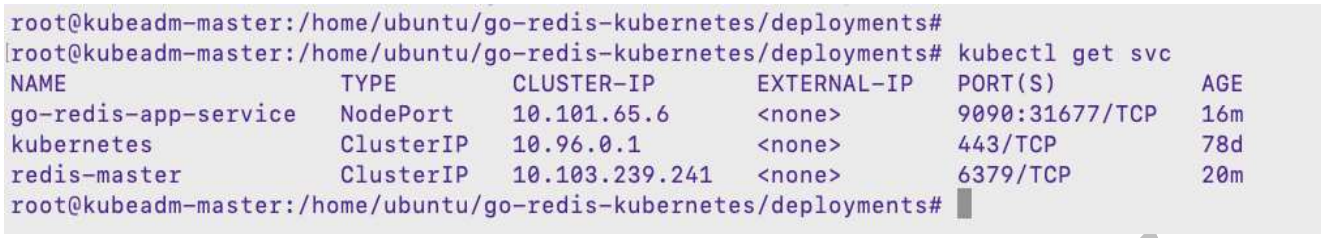


```
root@kubeadm-master:/home/ubuntu/go-redis-kubernetes/deployments# kubectl create -f go-redis-app.yml
deployment.apps/go-redis-app created
service/go-redis-app-service created
root@kubeadm-master:/home/ubuntu/go-redis-kubernetes/deployments# kubectl get all
NAME                                READY   STATUS        RESTARTS   AGE
pod/demo-pod                        1/1     Terminating   0          11h
pod/go-redis-app-5d68b49db6-psbvv   1/1     Running       0          7s
pod/go-redis-app-5d68b49db6-rhdqn   1/1     Running       0          7s
pod/go-redis-app-5d68b49db6-vtd5d   1/1     Running       0          7s
pod/redis-master-7d557b94bb-cbmhw   1/1     Running       0          4m26s

NAME                           TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
service/go-redis-app-service   NodePort    10.101.65.6      <none>        9090:31677/TCP   7s
service/kubernetes             ClusterIP   10.96.0.1        <none>        443/TCP          78d
service/redis-master           ClusterIP   10.103.239.241   <none>        6379/TCP         4m26s

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/go-redis-app   3/3     3            3           7s
deployment.apps/redis-master   1/1     1            1           4m26s

NAME                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/go-redis-app-5d68b49db6   3         3         3       7s
replicaset.apps/redis-master-7d557b94bb   1         1         1       4m26s
root@kubeadm-master:/home/ubuntu/go-redis-kubernetes/deployments#
```

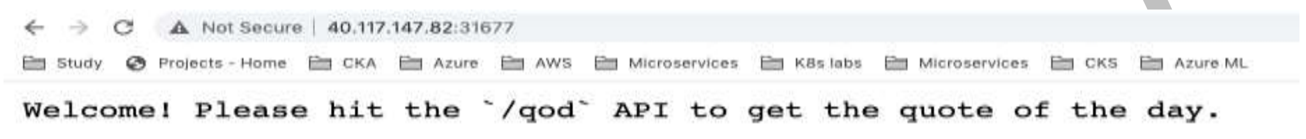
4.5 Accessing the Go App

1. The Go app is exposed as NodePort via the service. Get the nodeport details and access the application in web browser using Master Node's Public Ip address with Nodeport. Note: Make sure you use your master node's ip and nodeport allocated in your cluster

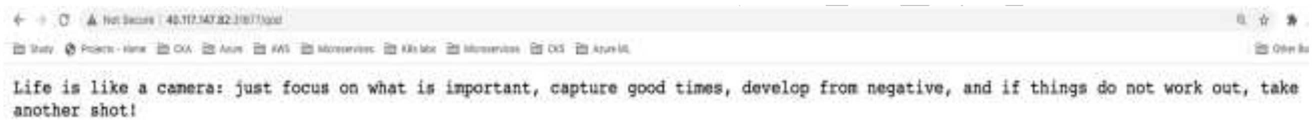
```
$ kubectl get svc
$ http://40.117.147.82:31677
$ http://40.117.147.82:31677/qod
```



```
root@kubeadm-master:/home/ubuntu/go-redis-kubernetes/deployments#
root@kubeadm-master:/home/ubuntu/go-redis-kubernetes/deployments# kubectl get svc
NAME                   TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
go-redis-app-service   NodePort    10.101.65.6      <none>        9090:31677/TCP   16m
kubernetes             ClusterIP   10.96.0.1        <none>        443/TCP          78d
redis-master           ClusterIP   10.103.239.241   <none>        6379/TCP         20m
root@kubeadm-master:/home/ubuntu/go-redis-kubernetes/deployments#
```



```
Welcome! Please hit the `/qod` API to get the quote of the day.
```



```
Life is like a camera: just focus on what is important, capture good times, develop from negative, and if things do not work out, take
another shot!
```

4.6 Cleaning up the resources

```
$ kubectl delete -f redis-master.yml
$ kubectl delete -f go-redis-app.yml
```

5 SUMMARY

This guide Covers:

Building Optimised Image for Docker Container – Multi Stage Dockerfile

- Git clone the Go App Code
- Building and Running the app locally
- Containerising the Go App - Building Docker Image
- Building Optimised Image – Multi Stage Builds

Deploying Stateless Go Application with Redis

- Git clone the Go App Code
- Containerising the Go App - Building Muilti Stage Docker Image
- Tag and Push image to Docker Private Registry
- Deploying the Go App and Redis to kubernetes Cluster
- Accessing the Go App