Capgemini

CONSULTING.TECHNOLOGY.OUTSOURCING

Kubernetes on GCP

by GCP Tower

People matter, results count.

KUBERNETES

*The name **Kubernetes** originates from Greek, meaning helmsman or pilot, and is the root of governor and cybernetic. **K8S** is an abbreviation derived by replacing the 8 letters “ubernete” with “8”*

Kubernetes is an open-source platform for **automating deployment**, scaling, and operations of application containers across clusters of hosts, providing **container-centric infrastructure**.

Kubernetes is not a mere **orchestration system**. In fact, it eliminates the need for orchestration. The technical definition of orchestration is execution of a defined workflow: first do A, then B, then C. In contrast, Kubernetes is comprised of a set of independent, composable control processes that continuously drive the current state towards the provided desired state. It shouldn’t matter how you get from A to C. Centralized control is also not required; the approach is more akin to choreography.

DEPLOYING AN APPLICATION ON GCP CONTAINER

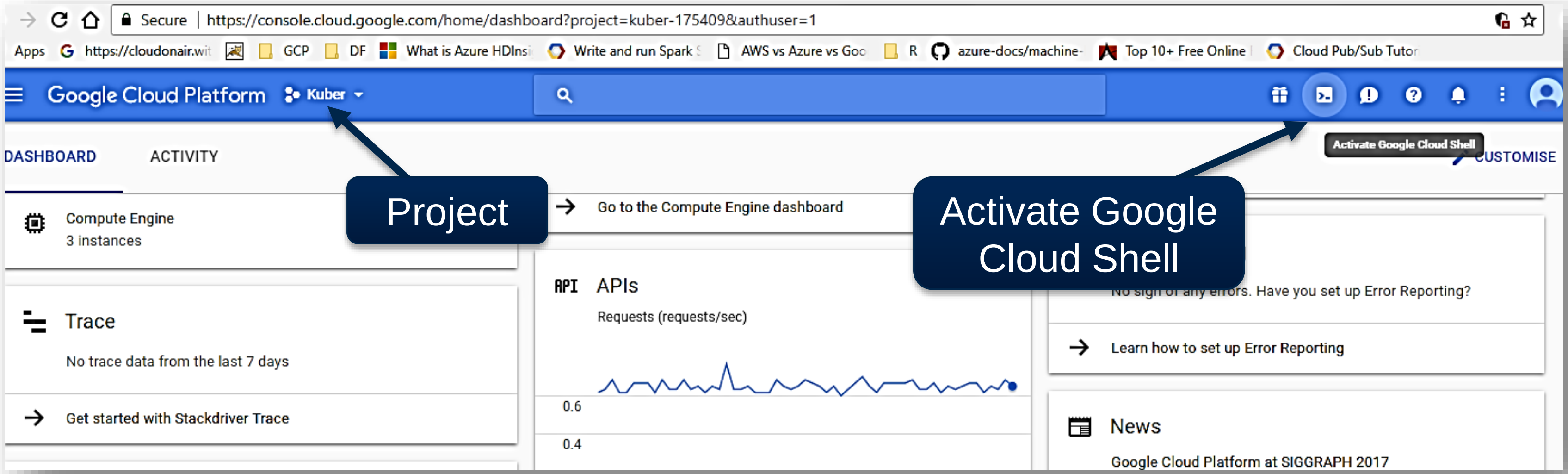
Step by Step packaging and deployment of application

- **Step 1: Build the container image**

  Google Container Engine accepts Docker images as the application deployment format.

  We have created an application and a Dockerfile. We are cloning it from github: https://github.com/kamakshig20/Kubernetes-Test/tree/master/sampleApp

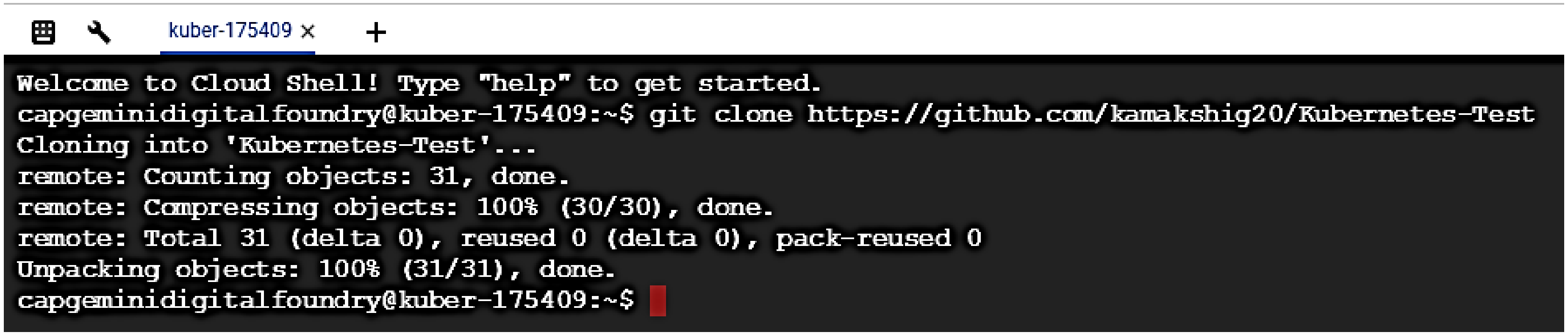
  Build a project and Open cloud shell in google cloud console:



Secure | https://console.cloud.google.com/home/dashboard?project=kuber-175409&authuser=1
Google Cloud Platform
Kuber
DASHBOARD
ACTIVITY
Activate Google Cloud Shell
CUSTOMISE
Project
Activate Google Cloud Shell
Compute Engine
3 instances
Go to the Compute Engine dashboard
Trace
No trace data from the last 7 days
Get started with Stackdriver Trace
APIs
Requests (requests/sec)
No sign of any errors. Have you set up Error Reporting?
Learn how to set up Error Reporting
News
Google Cloud Platform at SIGGRAPH 2017

Building container image

git clone https://github.com/kamakshig20/Kubernetes-Test



```
Welcome to Cloud Shell! Type "help" to get started.
capgeminidigitalfoundry@kuber-175409:~$ git clone https://github.com/kamakshig20/Kubernetes-Test
Cloning into 'Kubernetes-Test'...
remote: Counting objects: 31, done.
remote: Compressing objects: 100% (30/30), done.
remote: Total 31 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (31/31), done.
capgeminidigitalfoundry@kuber-175409:~$
```

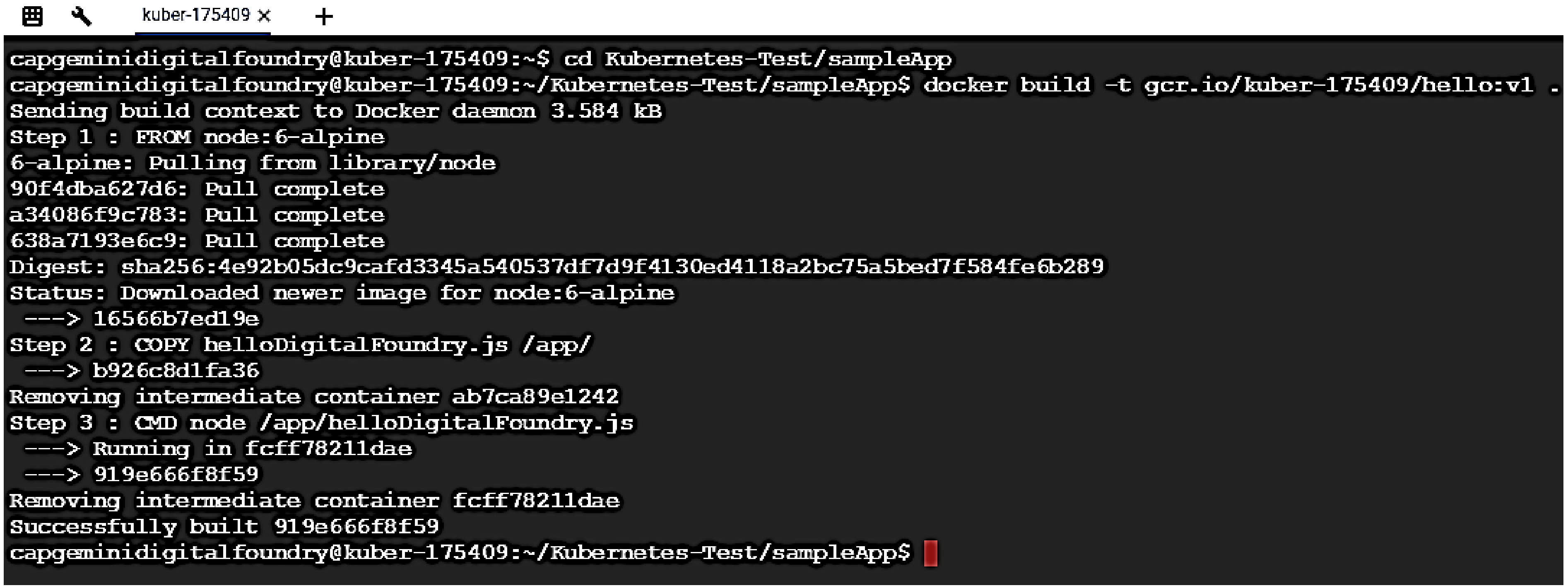
The application is a Node.js server application that responds to all requests with the message “Hello There, Have a Great Day! \n by Digital Foundry!” on port 8080. The repository also contains the Dockerfile which contains the instructions to build a container image of this application.

Building container image

docker build -t gcr.io/kuber-175409/hello:v1 .

This command builds the container image of this application and tags it for uploading.

kuber-175409 is the Project id.

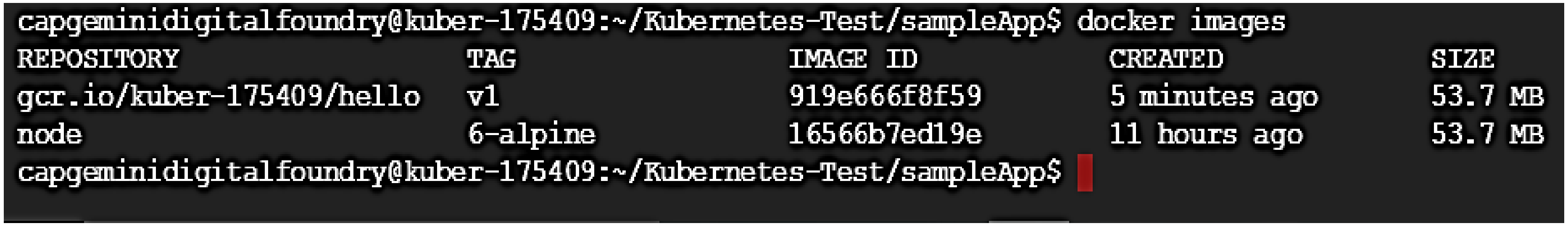


```
capgeminidigitalfoundry@kuber-175409:~$ cd Kubernetes-Test/sampleApp
capgeminidigitalfoundry@kuber-175409:~/Kubernetes-Test/sampleApp$ docker build -t gcr.io/kuber-175409/hello:v1 .
Sending build context to Docker daemon 3.584 kB
Step 1 : FROM node:6-alpine
6-alpine: Pulling from library/node
90f4dba627d6: Pull complete
a34086f9c783: Pull complete
638a7193e6c9: Pull complete
Digest: sha256:4e92b05dc9cafd3345a540537df7d9f4130ed4118a2bc75a5bed7f584fe6b289
Status: Downloaded newer image for node:6-alpine
 ---> 16566b7ed19e
Step 2 : COPY helloDigitalFoundry.js /app/
 ---> b926c8d1fa36
Removing intermediate container ab7ca89e1242
Step 3 : CMD node /app/helloDigitalFoundry.js
 ---> Running in fcff78211dae
 ---> 919e666f8f59
Removing intermediate container fcff78211dae
Successfully built 919e666f8f59
capgeminidigitalfoundry@kuber-175409:~/Kubernetes-Test/sampleApp$
```

Building container image

docker images

To verify that the image is built:



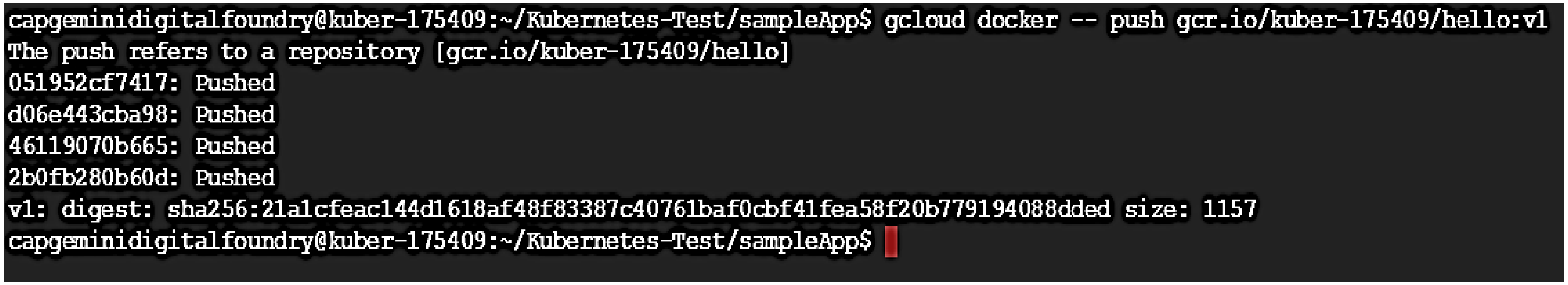
```
capgeminidigitalfoundry@kuber-175409:~/Kubernetes-Test/sampleApp$ docker images
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
gcr.io/kuber-175409/hello   v1                  919e666f8f59        5 minutes ago       53.7 MB
node                        6-alpine            16566b7ed19e        11 hours ago        53.7 MB
capgeminidigitalfoundry@kuber-175409:~/Kubernetes-Test/sampleApp$
```

- Step 2: Upload the container image

Next step is to upload your container image to a container registry

Upload the container image

gcloud docker -- push gcr.io/kuber-175409/hello:v1

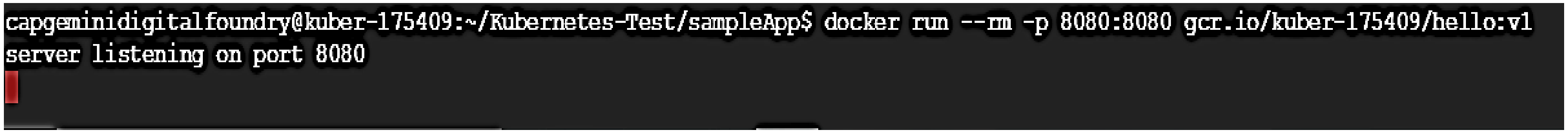


```
capgeminidigitalfoundry@kuber-175409:~/Kubernetes-Test/sampleApp$ gcloud docker -- push gcr.io/kuber-175409/hello:v1
The push refers to a repository [gcr.io/kuber-175409/hello]
051952cf7417: Pushed
d06e443cba98: Pushed
46119070b665: Pushed
2b0fb280b60d: Pushed
v1: digest: sha256:21a1cfeac144d1618af48f83387c40761baf0cbf41fea58f20b779194088dded size: 1157
capgeminidigitalfoundry@kuber-175409:~/Kubernetes-Test/sampleApp$
```

- Step 3: Run your container locally

    To test your container image using your local Docker engine

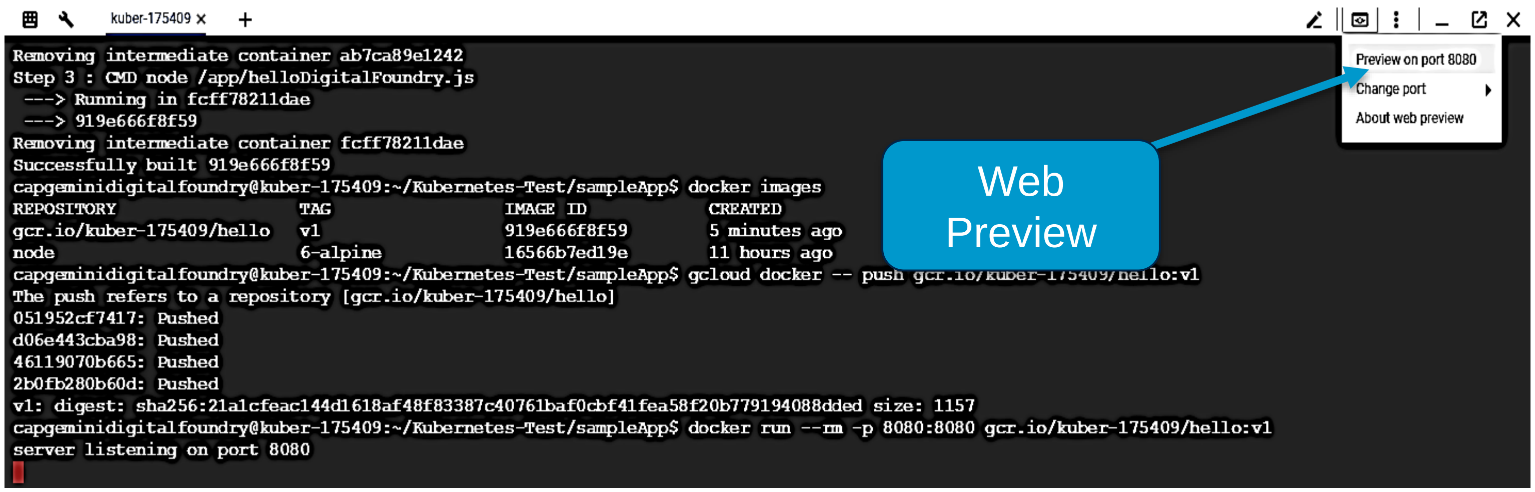
docker run --rm -p 8080:8080 gcr.io/kuber-175409/hello:v1



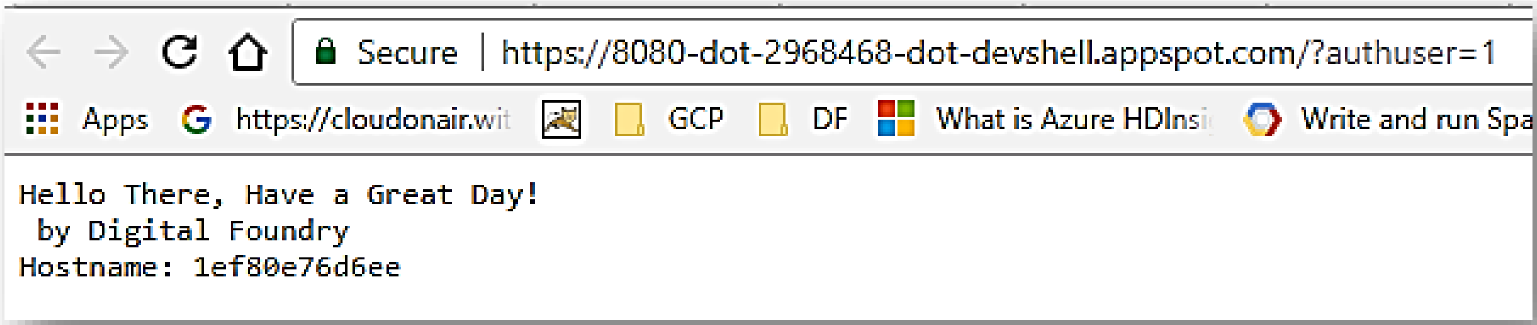
```
capgeminidigitalfoundry@kuber-175409:~/Kubernetes-Test/sampleApp$ docker run --rm -p 8080:8080 gcr.io/kuber-175409/hello:v1
server listening on port 8080
```

Run Container locally

Click "Web preview" on the top left to see your application running in a browser tab.



```
Removing intermediate container ab7ca89e1242
Step 3 : CMD node /app/helloDigitalFoundry.js
 ---> Running in fcff78211dae
 ---> 919e666f8f59
Removing intermediate container fcff78211dae
Successfully built 919e666f8f59
capgeminidigitalfoundry@kuber-175409:~/Kubernetes-Test/sampleApp$ docker images
REPOSITORY                  TAG                 IMAGE ID            CREATED
gcr.io/kuber-175409/hello   v1                  919e666f8f59        5 minutes ago
node                        6-alpine            16566b7ed19e        11 hours ago
capgeminidigitalfoundry@kuber-175409:~/Kubernetes-Test/sampleApp$ gcloud docker -- push gcr.io/kuber-175409/hello:v1
The push refers to a repository [gcr.io/kuber-175409/hello]
051952cf7417: Pushed
d06e443cba98: Pushed
46119070b665: Pushed
2b0fb280b60d: Pushed
v1: digest: sha256:21a1cfeac144d1618af48f83387c40761baf0cbf41fea58f20b779194088dded size: 1157
capgeminidigitalfoundry@kuber-175409:~/Kubernetes-Test/sampleApp$ docker run --rm -p 8080:8080 gcr.io/kuber-175409/hello:v1
server listening on port 8080
```



```
Hello There, Have a Great Day!
 by Digital Foundry
Hostname: 1ef80e76d6ee
```

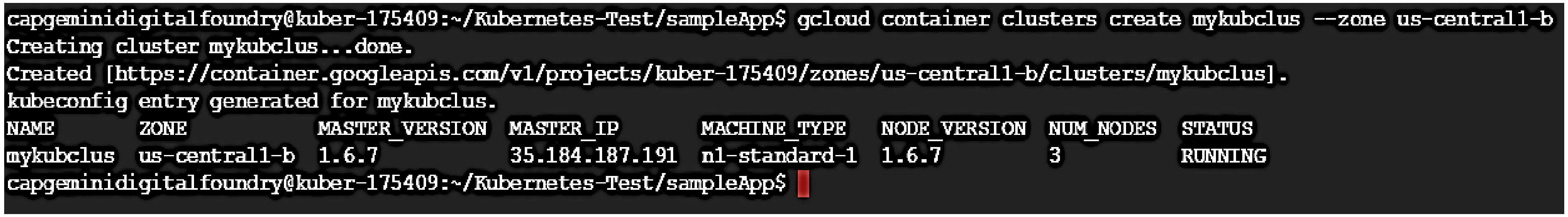
Shut down the container by pressing Ctrl+C in the tab where docker run command is running.

Create a container cluster

- Step 4: Create your container cluster

Create a container cluster to run the container image. A cluster consists of a pool of compute instances running Kubernetes.

gcloud container clusters create mykubclus --zone us-central1-b



```
capgeminidigitalfoundry@kuber-175409:~/Kubernetes-Test/sampleApp$ gcloud container clusters create mykubclus --zone us-central1-b
Creating cluster mykubclus...done.
Created [https://container.googleapis.com/v1/projects/kuber-175409/zones/us-central1-b/clusters/mykubclus].
kubeconfig entry generated for mykubclus.
NAME       ZONE           MASTER_VERSION  MASTER_IP       MACHINE_TYPE   NODE_VERSION  NUM_NODES  STATUS
mykubclus  us-central1-b  1.6.7           35.184.187.191  n1-standard-1  1.6.7         3          RUNNING
capgeminidigitalfoundry@kuber-175409:~/Kubernetes-Test/sampleApp$
```

We have created a container cluster named mykubclus with default 3 nodes.

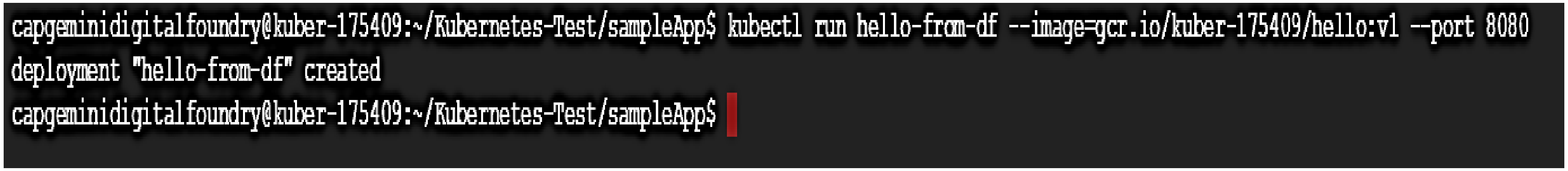
Deploy your application

- Step 5: Deploy your application

To deploy and manage applications on a Container Engine cluster, we communicate with the Kubernetes cluster management system by using the **kubectl** command-line tool.

Kubernetes represents applications as **Pods**, which are units that represent a **container (or group of tightly-coupled containers).** The Pod is the smallest deployable unit in Kubernetes. Here each Pod contains only our Node.js container image.

kubectl run hello-from-df --image=gcr.io/kuber-175409/hello:v1 --port 8080



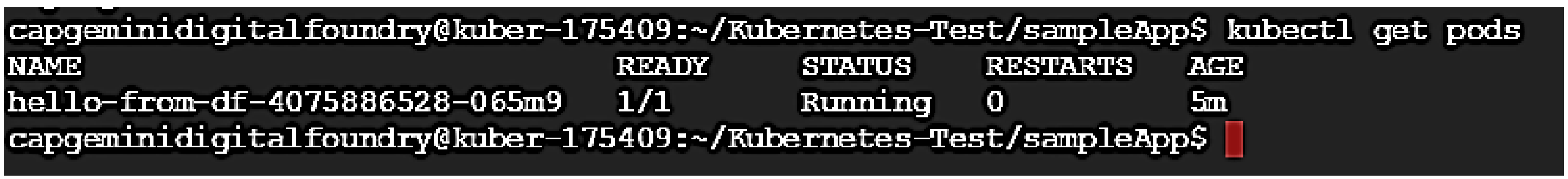
```
capgeminidigitalfoundry@kuber-175409:~/Kubernetes-Test/sampleApp$ kubectl run hello-from-df --image=gcr.io/kuber-175409/hello:v1 --port 8080
deployment "hello-from-df" created
capgeminidigitalfoundry@kuber-175409:~/Kubernetes-Test/sampleApp$
```

Deploy your application

This creates a single Pod listening on port 8080. The kubectl run command causes Kubernetes to create a Deployment resource on cluster.

kubectl get pods

To view the Pod that got created by the Deployment:



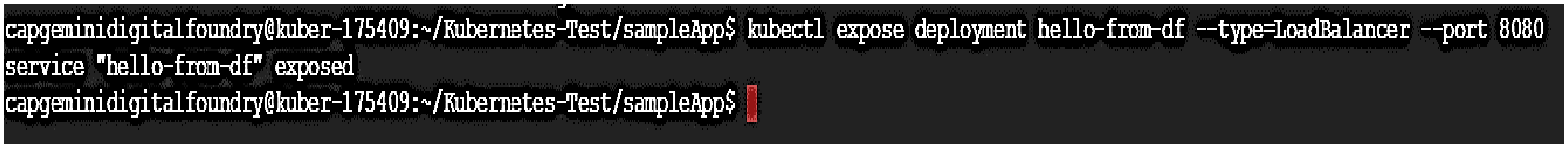
```
capgeminidigitalfoundry@kuber-175409:~/Kubernetes-Test/sampleApp$ kubectl get pods
NAME                                 READY     STATUS    RESTARTS   AGE
hello-from-df-4075886528-065m9       1/1       Running   0          5m
capgeminidigitalfoundry@kuber-175409:~/Kubernetes-Test/sampleApp$
```

Expose application

- Step 6: Expose application

  The containers you run on Container Engine are not accessible from the internet, since they do not have external IP addresses. We expose the application to traffic from the internet by using the kubectl expose command.

kubectl expose deployment hello-from-df --type=LoadBalancer --port 8080

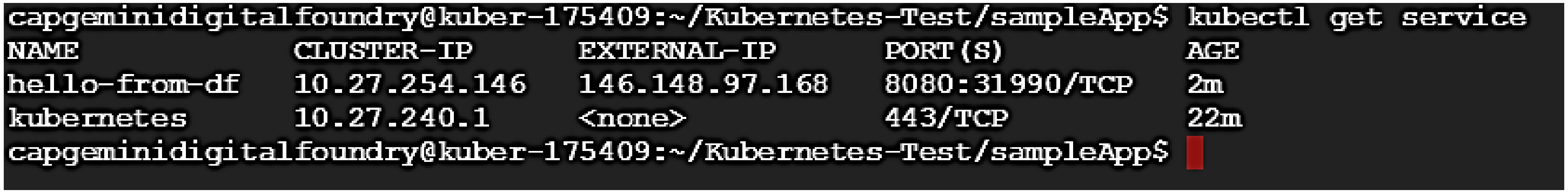


```
capgeminidigitalfoundry@kuber-175409:~/Kubernetes-Test/sampleApp$ kubectl expose deployment hello-from-df --type=LoadBalancer --port 8080
service "hello-from-df" exposed
capgeminidigitalfoundry@kuber-175409:~/Kubernetes-Test/sampleApp$
```

The kubectl expose command causes Kubernetes to create a Service resource, which provides networking and IP support to your application's Pods. Container Engine creates an external IP and a Load Balancer for application.

Expose application

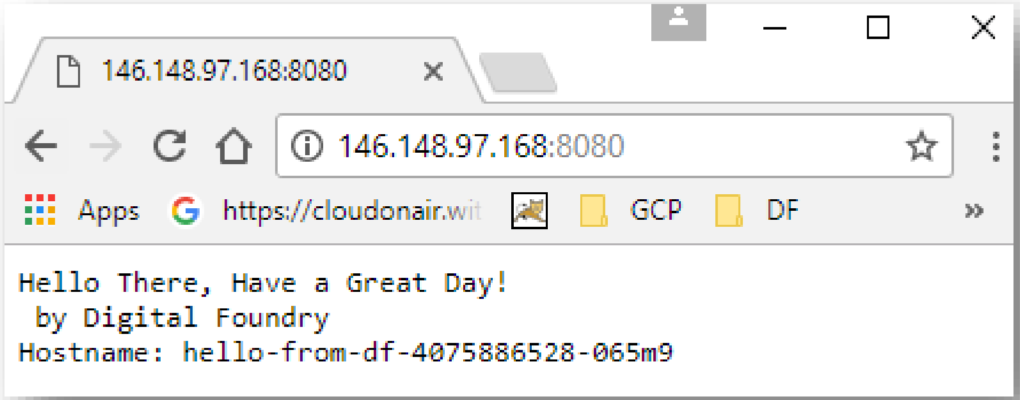
kubectl get service



```
capgeminidigitalfoundry@kuber-175409:~/Kubernetes-Test/sampleApp$ kubectl get service
NAME            CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
hello-from-df   10.27.254.146   146.148.97.168   8080:31990/TCP   2m
kubernetes      10.27.240.1     <none>           443/TCP          22m
capgeminidigitalfoundry@kuber-175409:~/Kubernetes-Test/sampleApp$
```

From here we determine the external IP address for application, copy the IP address and add the port number.

Point browser to this URL (e.g. http://146.148.97.168:8080) to access the application.



```
Hello There, Have a Great Day!
 by Digital Foundry
Hostname: hello-from-df-4075886528-065m9
```

People matter, results count.

About Capgemini

With almost 140,000 people in over 40 countries, Capgemini is one of the world's foremost providers of consulting, technology and outsourcing services. The Group reported 2013 global revenues of EUR 10.1 billion.
Together with its clients, Capgemini creates and delivers business and technology solutions that fit their needs and drive the results they want. A deeply multicultural organization, Capgemini has developed its own way of working, the Collaborative Business Experience^{TM}, and draws on Rightshore®, its worldwide delivery model.

www.capgemini.com