LAPORAN PRAKTIKUM

TEKNOLOGI CLOUD

PERTEMUAN KE – 12

Disusun Oleh :

NAMA	**: TARISA DWI SEPTIA**
NIM	**: 205410126**
JURUSAN	**: TEKNIK INFORMATIKA**
JENJANG	**: S1**

UNIVERSITAS TEKNOLOGI DIGITAL INDONSIA

YOGYAKARTA

2020

Docker Compose

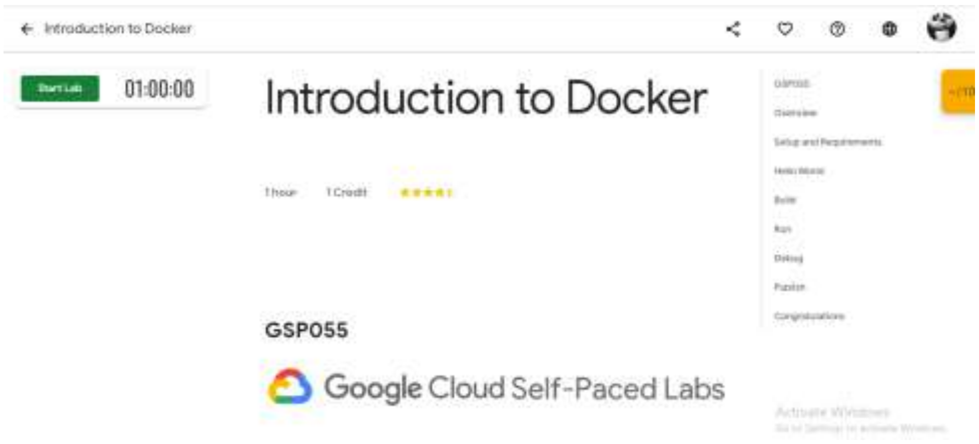
A. Tujuan

- Mahasiswa dapat mem-build, menjalankan, dan melakukan debug container Docker.
- Mahasiswa dapat mengambil image Docker dari Docker Hub dan Google Container Registry.
- Mahasiswa dapat menerapkan image Docker ke Google Container Registry.

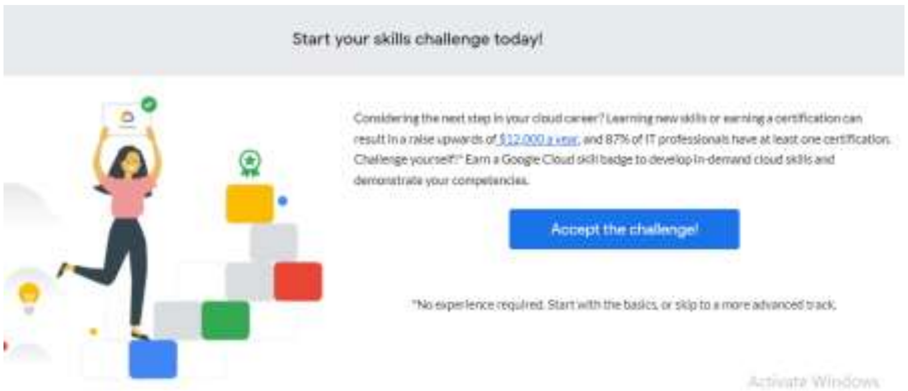
B. Praktikum

1. Persiapan

a. Masuk ke lab



b. Mengambil token untuk mengakses lab

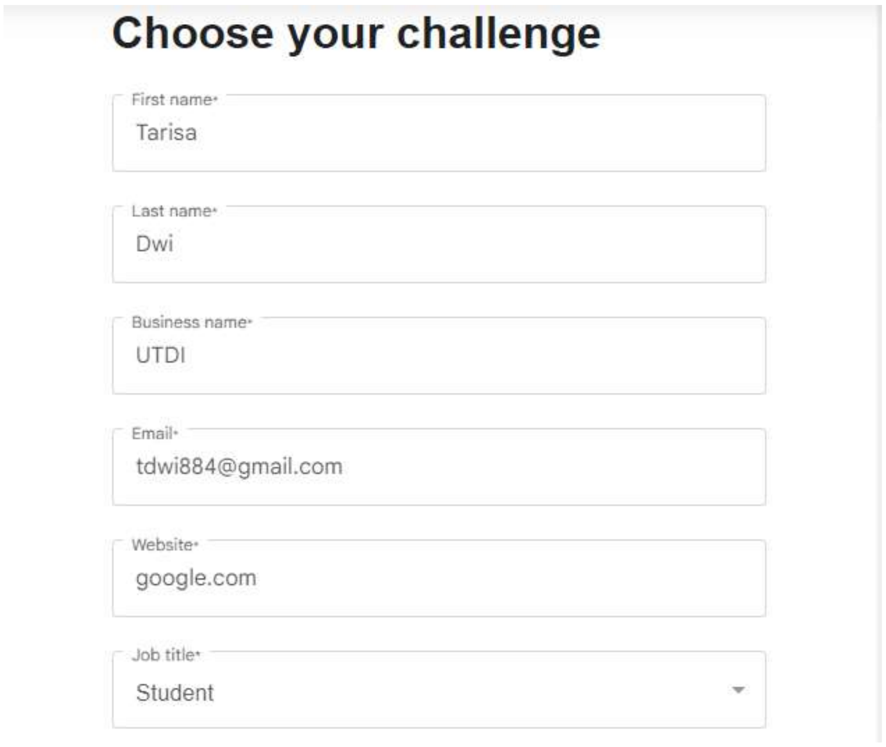


Start your skills challenge today!

Considering the next step in your cloud career? Learning new skills or earning a certification can result in a raise upwards of $12,000 a year, and 87% of IT professionals have at least one certification. Challenge yourself!" Earn a Google Cloud skill badge to develop in-demand cloud skills and demonstrate your competencies.

*No experience required. Start with the basics, or skip to a more advanced track.

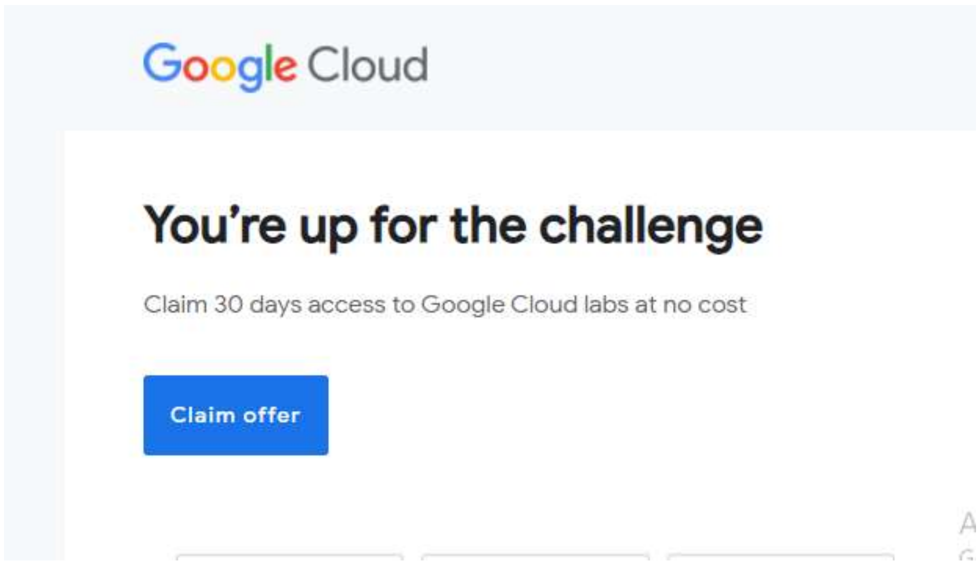
c. Mengisi biodata untuk mendapatkan token



Choose your challenge

First name*	Tarisa
Last name*	Dwi
Business name*	UTDI
Email*	tdwi884@gmail.com
Website*	google.com
Job title*	Student

d. Cek email dari qwicklab, setelah itu klik claim offer dan login akun google melalui link yang diberikan

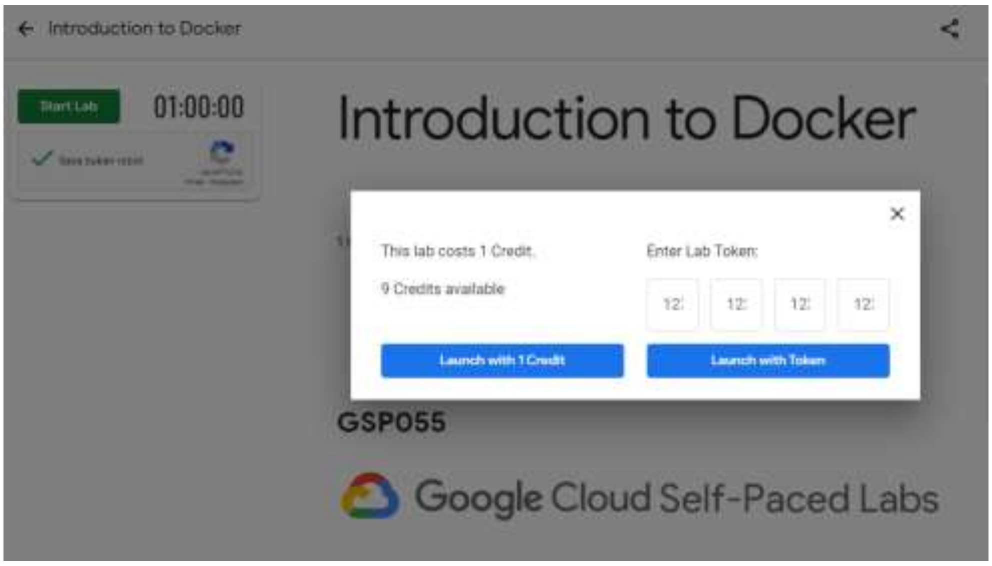


Google Cloud

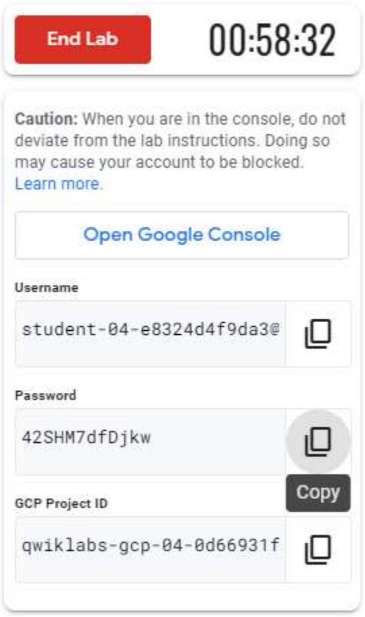
You're up for the challenge

Claim 30 days access to Google Cloud labs at no cost

e. Setelah itu mencari lab Introcution to Docker, kemudian start lab. Setelah di klik start lab akan muncul popup seperti di atas, klik saja lunch with 1 credit



f. Lab sudah mulai


End Lab 00:58:32

Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. Learn more.

Open Google Console

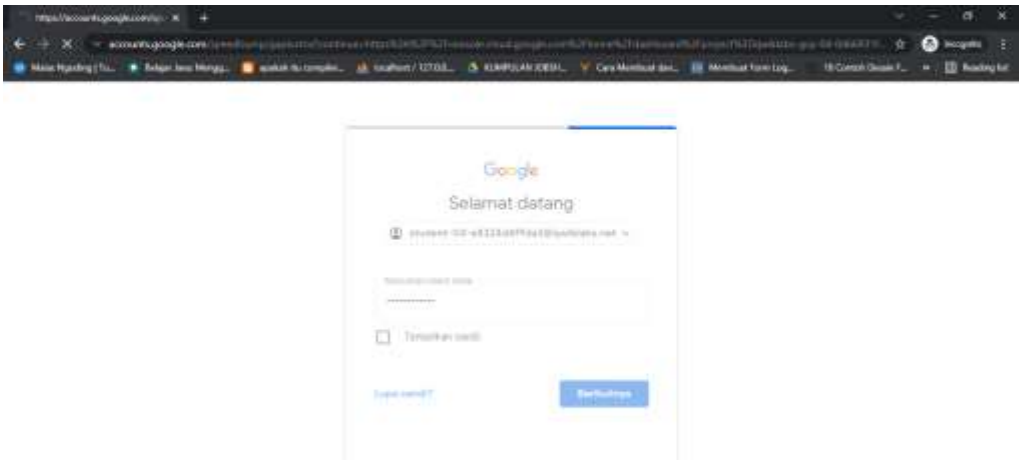
Username
student-04-e8324d4f9da3@

Password
42SHM7dfDjkw

Copy

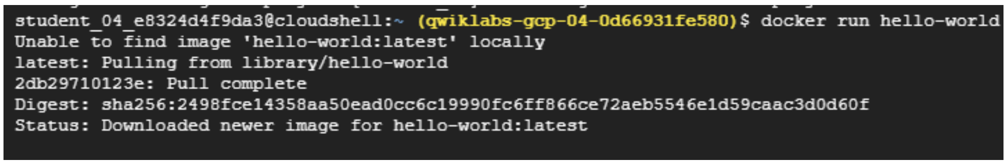
GCP Project ID
qwiklabs-gcp-04-0d66931f

g. Copy link ke tab samara agar akunya tidak tertindih dengan akun qwiklabs



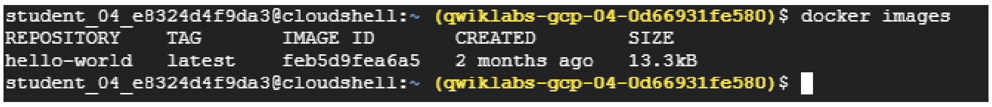
2. Hello Word

a. Docker run hello-world



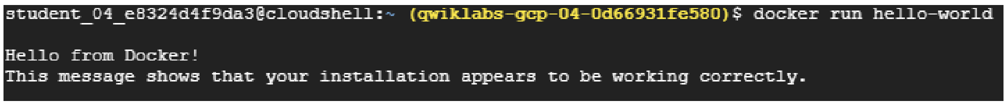
```
student_04_e8324d4f9da3@cloudshell:~ (qwiklabs-gcp-04-0d66931fe580)$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:2498fce14358aa50ead0cc6c19990fc6ff866ce72aeb5546e1d59caac3d0d60f
Status: Downloaded newer image for hello-world:latest
```

b. Docker – images



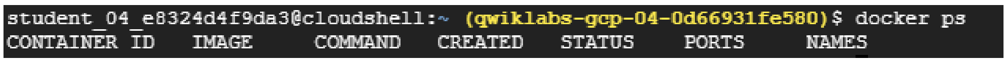
```
student_04_e8324d4f9da3@cloudshell:~ (qwiklabs-gcp-04-0d66931fe580)$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world   latest    feb5d9fea6a5   2 months ago   13.3kB
student_04_e8324d4f9da3@cloudshell:~ (qwiklabs-gcp-04-0d66931fe580)$
```

c. Docker run hello-world



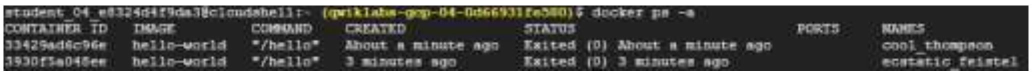
```
student_04_e8324d4f9da3@cloudshell:~ (qwiklabs-gcp-04-0d66931fe580)$ docker run hello-world

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

d. Docker ps



```
student_04_e8324d4f9da3@cloudshell:~ (qwiklabs-gcp-04-0d66931fe580)$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

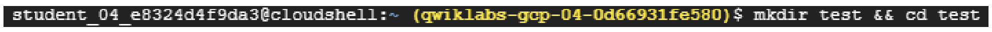
e. Docker ps –a



```
student_04_e8324d4f9da3@cloudshell:~ (qwiklabs-gcp-04-0d66931fe580)$ docker ps -a
CONTAINER ID   IMAGE         COMMAND    CREATED              STATUS                          PORTS     NAMES
33429ad6c96e   hello-world   "/hello"   About a minute ago   Exited (0) About a minute ago             cool_thompson
3930f5a048ee   hello-world   "/hello"   3 minutes ago        Exited (0) 3 minutes ago                  ecstatic_feistel
```

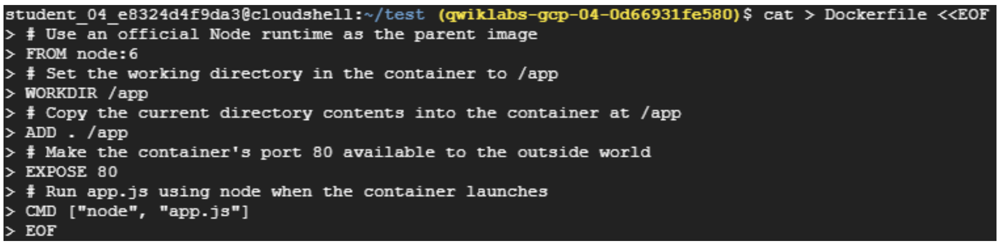
3. Build

a. mkdir test && cd test



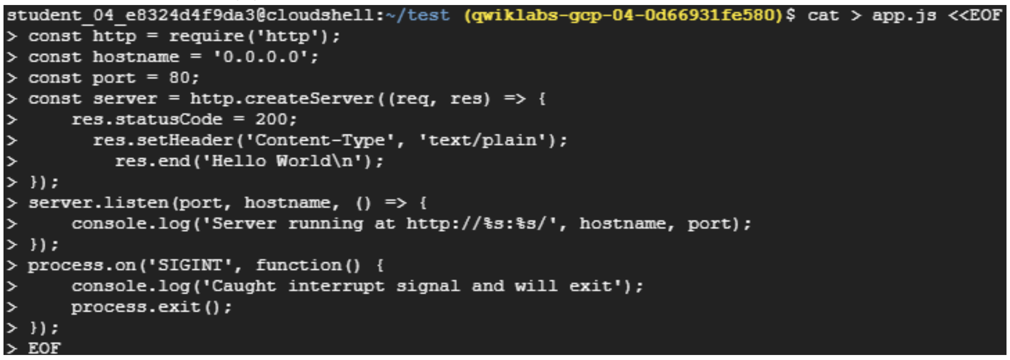
```
student_04_e8324d4f9da3@cloudshell:~ (qwiklabs-gcp-04-0d66931fe580)$ mkdir test && cd test
```

b. Create a Dockerfile



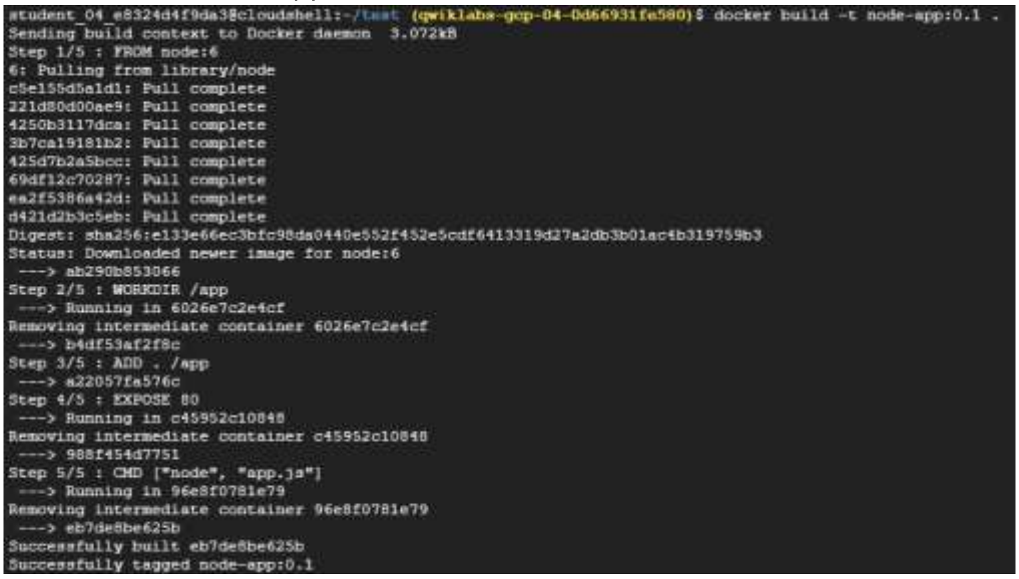
```
student_04_e8324d4f9da3@cloudshell:~/test (qwiklabs-gcp-04-0d66931fe580)$ cat > Dockerfile <<EOF
> # Use an official Node runtime as the parent image
> FROM node:6
> # Set the working directory in the container to /app
> WORKDIR /app
> # Copy the current directory contents into the container at /app
> ADD . /app
> # Make the container's port 80 available to the outside world
> EXPOSE 80
> # Run app.js using node when the container launches
> CMD ["node", "app.js"]
> EOF
```

c. Create the node application



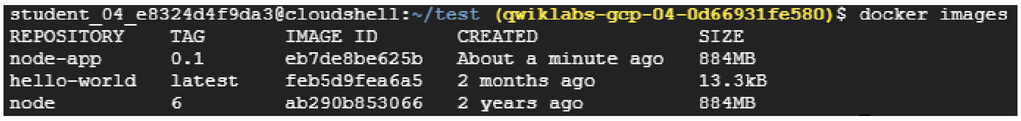
```
student_04_e8324d4f9da3@cloudshell:~/test (qwiklabs-gcp-04-0d66931fe580)$ cat > app.js <<EOF
> const http = require('http');
> const hostname = '0.0.0.0';
> const port = 80;
> const server = http.createServer((req, res) => {
>     res.statusCode = 200;
>       res.setHeader('Content-Type', 'text/plain');
>         res.end('Hello World\n');
> });
> server.listen(port, hostname, () => {
>     console.log('Server running at http://%s:%s/', hostname, port);
> });
> process.on('SIGINT', function() {
>     console.log('Caught interrupt signal and will exit');
>     process.exit();
> });
> EOF
```

d. Docker build -t node-app:0.1 .



```
student_04_e8324d4f9da3@cloudshell:~/test (qwiklabs-gcp-04-0d66931fe580)$ docker build -t node-app:0.1 .
Sending build context to Docker daemon  3.072kB
Step 1/5 : FROM node:6
6: Pulling from library/node
c5e155d5a1d1: Pull complete
221d80d00ae9: Pull complete
4250b3117dca: Pull complete
3b7ca19181b2: Pull complete
425d7b2a5bcc: Pull complete
69df12c70287: Pull complete
ea2f5386a42d: Pull complete
d421d2b3c5eb: Pull complete
Digest: sha256:e133e66ec3bfc98da0440e552f452e5cdf6413319d27a2db3b01ac4b319759b3
Status: Downloaded newer image for node:6
 ---> ab290b853066
Step 2/5 : WORKDIR /app
 ---> Running in 6026e7c2e4cf
Removing intermediate container 6026e7c2e4cf
 ---> b4df53af2f8c
Step 3/5 : ADD . /app
 ---> a22057fa576c
Step 4/5 : EXPOSE 80
 ---> Running in c45952c10848
Removing intermediate container c45952c10848
 ---> 988f454d7751
Step 5/5 : CMD ["node", "app.js"]
 ---> Running in 96e8f0781e79
Removing intermediate container 96e8f0781e79
 ---> eb7de8be625b
Successfully built eb7de8be625b
Successfully tagged node-app:0.1
```

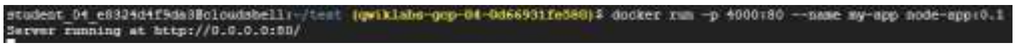
e. Run the following command to look at the images you built



```
student_04_e8324d4f9da3@cloudshell:~/test (qwiklabs-gcp-04-0d66931fe580)$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED              SIZE
node-app      0.1       eb7de8be625b   About a minute ago   884MB
hello-world   latest    feb5d9fea6a5   2 months ago         13.3kB
node          6         ab290b853066   2 years ago          884MB
```

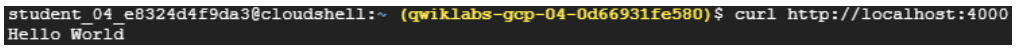
4. Run

a. use this code to run containers based on the image you built



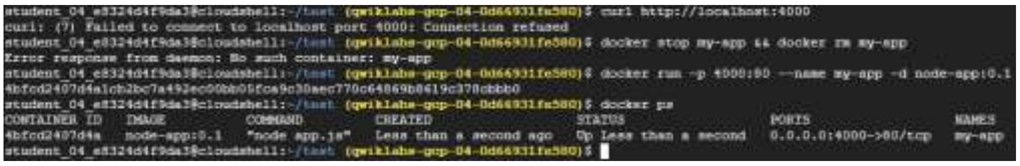
```
student_04_e8324d4f9da3@cloudshell:~/test (qwiklabs-gcp-04-0d66931fe580)$ docker run -p 4000:80 --name my-app node-app:0.1
Server running at http://0.0.0.0:80/
```

b. Open another terminal (in Cloud Shell, click the + icon), and test the server



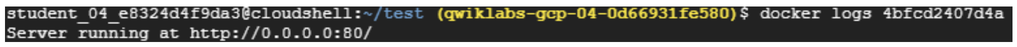
```
student_04_e8324d4f9da3@cloudshell:~ (qwiklabs-gcp-04-0d66931fe580)$ curl http://localhost:4000
Hello World
```

c. Close the initial terminal and then run the following command to stop and remove the container:



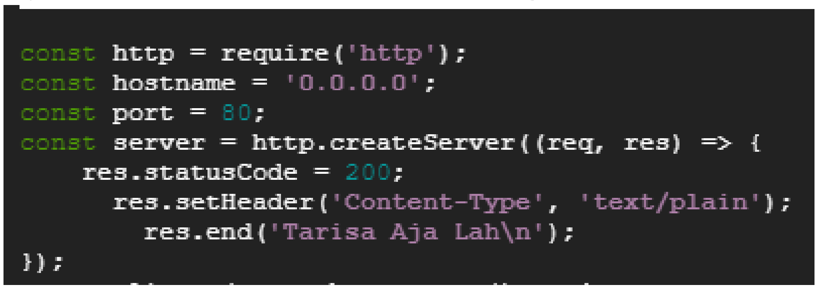
```
student_04_e8324d4f9da3@cloudshell:~/test (qwiklabs-gcp-04-0d66931fe580)$ curl http://localhost:4000
curl: (7) Failed to connect to localhost port 4000: Connection refused
student_04_e8324d4f9da3@cloudshell:~/test (qwiklabs-gcp-04-0d66931fe580)$ docker stop my-app && docker rm my-app
Error response from daemon: No such container: my-app
student_04_e8324d4f9da3@cloudshell:~/test (qwiklabs-gcp-04-0d66931fe580)$ docker run -p 4000:80 --name my-app -d node-app:0.1
4bfcd2407d4a1cb2bc7a492ec00bb05fca9c30aec770c64869b0619c378cbbb0
student_04_e8324d4f9da3@cloudshell:~/test (qwiklabs-gcp-04-0d66931fe580)$ docker ps
CONTAINER ID   IMAGE          COMMAND         CREATED                   STATUS                   PORTS                    NAMES
4bfcd2407d4a   node-app:0.1   "node app.js"   Less than a second ago    Up Less than a second    0.0.0.0:4000->80/tcp     my-app
student_04_e8324d4f9da3@cloudshell:~/test (qwiklabs-gcp-04-0d66931fe580)$
```

d. You can look at the logs by executing docker logs [container_id].



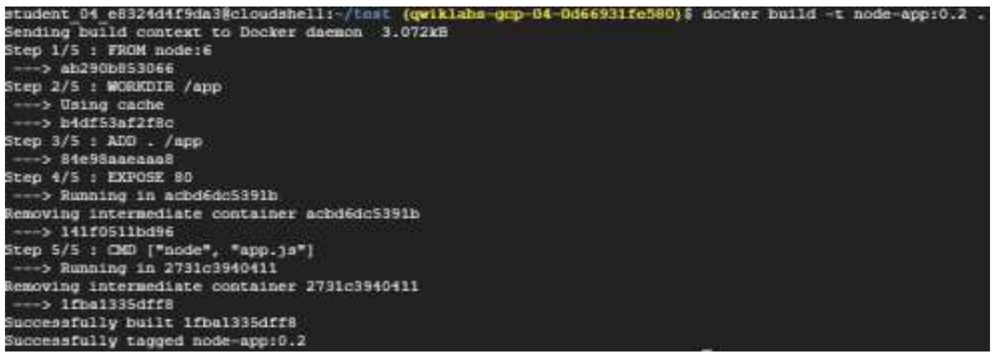
```
student_04_e8324d4f9da3@cloudshell:~/test (qwiklabs-gcp-04-0d66931fe580)$ docker logs 4bfcd2407d4a
Server running at http://0.0.0.0:80/
```

e. Edit app.js with a text editor of your choice (for example nano or vim) and replace "Hello World" with another string:



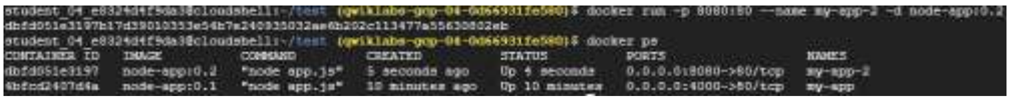
```
const http = require('http');
const hostname = '0.0.0.0';
const port = 80;
const server = http.createServer((req, res) => {
    res.statusCode = 200;
      res.setHeader('Content-Type', 'text/plain');
        res.end('Tarisa Aja Lah\n');
});
```

f. Build this new image and tag it with 0.2:



```
student_04_e8324d4f9da3@cloudshell:~/test (qwiklabs-gcp-04-0d66931fe580)$ docker build -t node-app:0.2 .
Sending build context to Docker daemon  3.072kB
Step 1/5 : FROM node:6
 ---> ab290b853066
Step 2/5 : WORKDIR /app
 ---> Using cache
 ---> b4df53af2f8c
Step 3/5 : ADD . /app
 ---> 84e98aaeaaa8
Step 4/5 : EXPOSE 80
 ---> Running in acbd6dc5391b
Removing intermediate container acbd6dc5391b
 ---> 141f0511bd96
Step 5/5 : CMD ["node", "app.js"]
 ---> Running in 2731c3940411
Removing intermediate container 2731c3940411
 ---> 1fba1335dff8
Successfully built 1fba1335dff8
Successfully tagged node-app:0.2
```

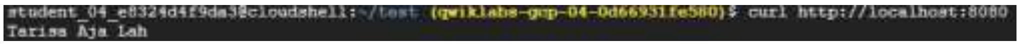
g. Run another container with the new image version. Notice how we map the host's port 8080 instead of 80. We can't use host port 4000 because it's



```
student_04_e8324d4f9da3@cloudshell:~/test (qwiklabs-gcp-04-0d66931fe580)$ docker run -p 8080:80 --name my-app-2 -d node-app:0.2
dbfd051e3197b17d39010353e54b7e240935032ae6b202c113477a55630802eb
student_04_e8324d4f9da3@cloudshell:~/test (qwiklabs-gcp-04-0d66931fe580)$ docker ps
CONTAINER ID   IMAGE          COMMAND          CREATED          STATUS          PORTS                  NAMES
dbfd051e3197   node-app:0.2   "node app.js"    5 seconds ago    Up 4 seconds    0.0.0.0:8080->80/tcp   my-app-2
4bfcd2407d4a   node-app:0.1   "node app.js"    10 minutes ago   Up 10 minutes   0.0.0.0:4000->80/tcp   my-app
```

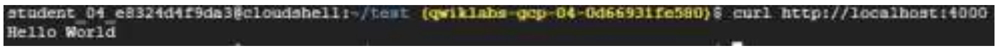
already in use.

h. Test the containers:



```
student_04_e8324d4f9da3@cloudshell:~/test (qwiklabs-gcp-04-0d66931fe580)$ curl http://localhost:8080
Tarisa Aja Lah
```

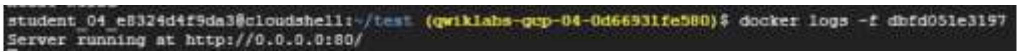
i. And now test the first container you made:



```
student_04_e8324d4f9da3@cloudshell:~/test (qwiklabs-gcp-04-0d66931fe580)$ curl http://localhost:4000
Hello World
```

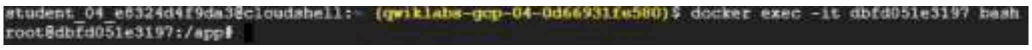
5. Debug

a. You can look at the logs of a container using docker logs [container_id]. If you want to follow the log's output as the container is running, use the -f option.



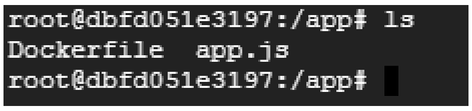
```
student_04_e8324d4f9da3@cloudshell:~/test (qwiklabs-gcp-04-0d66931fe580)$ docker logs -f dbfd051e3197
Server running at http://0.0.0.0:80/
```

b. Open another terminal (in Cloud Shell, click the + icon) and enter the following command:



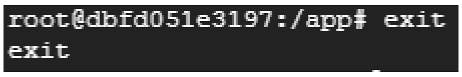
```
student_04_e8324d4f9da3@cloudshell:~ (qwiklabs-gcp-04-0d66931fe580)$ docker exec -it dbfd051e3197 bash
root@dbfd051e3197:/app#
```

c. Look at the directory.



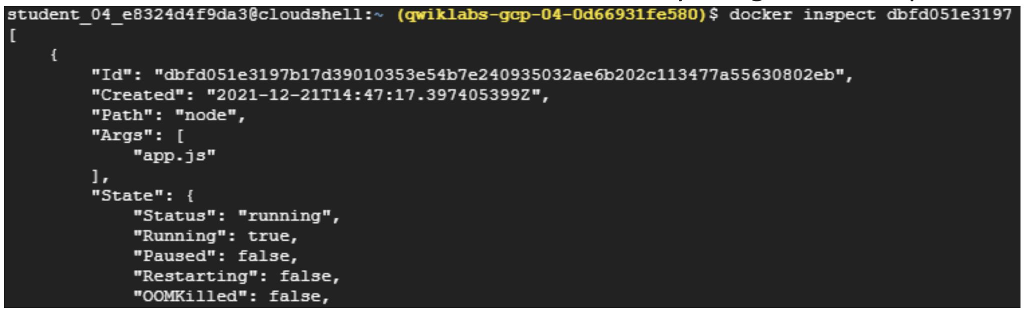
```
root@dbfd051e3197:/app# ls
Dockerfile  app.js
root@dbfd051e3197:/app#
```

d. Exit the Bash session:



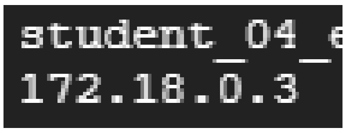
```
root@dbfd051e3197:/app# exit
exit
```

e. You can examine a container's metadata in Docker by using Docker inspect:



```
student_04_e8324d4f9da3@cloudshell:~ (qwiklabs-gcp-04-0d66931fe580)$ docker inspect dbfd051e3197
[
    {
        "Id": "dbfd051e3197b17d39010353e54b7e240935032ae6b202c113477a55630802eb",
        "Created": "2021-12-21T14:47:17.397405399Z",
        "Path": "node",
        "Args": [
            "app.js"
        ],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
```

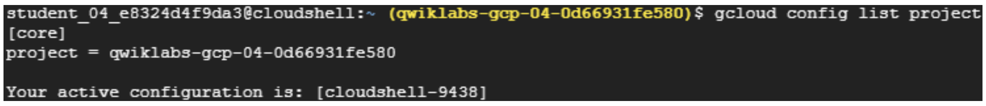
f. Use --format to inspect specific fields from the returned JSON. For example:



```
student_04_e
172.18.0.3
```

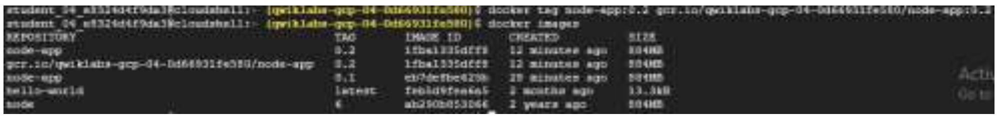
6. Publish

a. You can find your project ID by running:



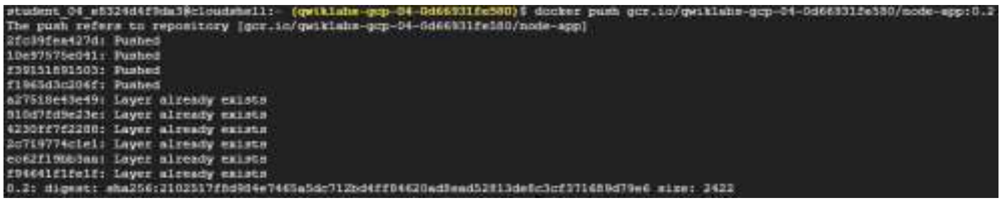
```
student_04_e8324d4f9da3@cloudshell:~ (qwiklabs-gcp-04-0d66931fe580)$ gcloud config list project
[core]
project = qwiklabs-gcp-04-0d66931fe580

Your active configuration is: [cloudshell-9438]
```

b. Tag node-app:0.2. Replace [project-id] with your configuration..



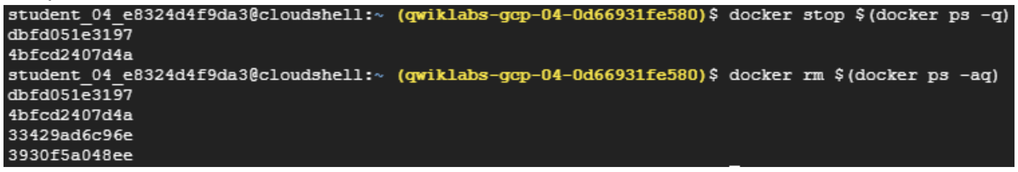
```
student_04_e8324d4f9da3@cloudshell:~ (qwiklabs-gcp-04-0d66931fe580)$ docker tag node-app:0.2 gcr.io/qwiklabs-gcp-04-0d66931fe580/node-app:0.2
student_04_e8324d4f9da3@cloudshell:~ (qwiklabs-gcp-04-0d66931fe580)$ docker images
REPOSITORY                                        TAG       IMAGE ID       CREATED          SIZE
node-app                                          0.2       1fba1335dff9   12 minutes ago   884MB
gcr.io/qwiklabs-gcp-04-0d66931fe580/node-app      0.2       1fba1335dff9   12 minutes ago   884MB
node-app                                          0.1       eb7defbe4290   28 minutes ago   884MB
hello-world                                       latest    feb5d9fea6a5   2 months ago     13.3kB
node                                              6         ab290b853066   2 years ago      884MB
```

c. Push this image to gcr. Remember to replace [project-id].



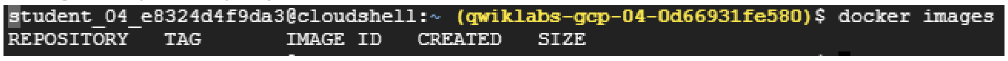
```
student_04_e8324d4f9da3@cloudshell:~ (qwiklabs-gcp-04-0d66931fe580)$ docker push gcr.io/qwiklabs-gcp-04-0d66931fe580/node-app:0.2
The push refers to repository [gcr.io/qwiklabs-gcp-04-0d66931fe580/node-app]
2fc39fea427d: Pushed
10e97575e041: Pushed
f39151891503: Pushed
f1963d3c206f: Pushed
a27518e43e49: Layer already exists
910d7fd9e23e: Layer already exists
4230ff7f2288: Layer already exists
2c719774c1e1: Layer already exists
ec62f19bb3aa: Layer already exists
f94641f1fe1f: Layer already exists
0.2: digest: sha256:2102517f8d984e7465a5dc712bd4ff84620ad8ead52813de8c3cf371689d79e6 size: 2422
```

d. Stop and remove all containers:



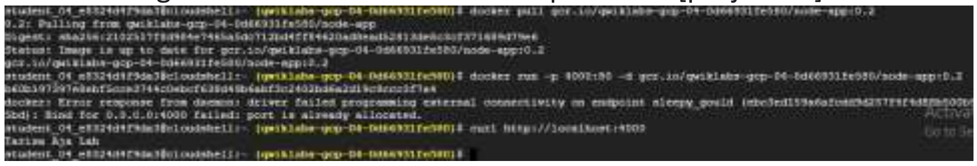
```
student_04_e8324d4f9da3@cloudshell:~ (qwiklabs-gcp-04-0d66931fe580)$ docker stop $(docker ps -q)
dbfd051e3197
4bfcd2407d4a
student_04_e8324d4f9da3@cloudshell:~ (qwiklabs-gcp-04-0d66931fe580)$ docker rm $(docker ps -aq)
dbfd051e3197
4bfcd2407d4a
33429ad6c96e
3930f5a048ee
```

e. You have to remove the child images (of node:6) before you remove the node image. Replace [project-id].



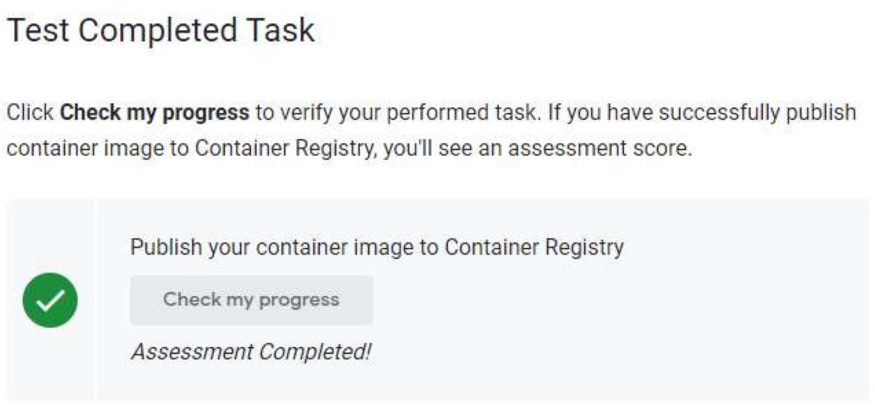
```
student_04_e8324d4f9da3@cloudshell:~ (qwiklabs-gcp-04-0d66931fe580)$ docker images
REPOSITORY    TAG       IMAGE ID    CREATED    SIZE
```

g. Pull the image and run it. Remember to replace the [project-id].



```
student_04_e8324d4f9da3@cloudshell:~ (qwiklabs-gcp-04-0d66931fe580)$ docker pull gcr.io/qwiklabs-gcp-04-0d66931fe580/node-app:0.2
0.2: Pulling from qwiklabs-gcp-04-0d66931fe580/node-app
Digest: sha256:2102517f8d984e7465a5dc712bd4ff84620ad8ead52813de8c3cf371689d79e6
Status: Image is up to date for gcr.io/qwiklabs-gcp-04-0d66931fe580/node-app:0.2
gcr.io/qwiklabs-gcp-04-0d66931fe580/node-app:0.2
student_04_e8324d4f9da3@cloudshell:~ (qwiklabs-gcp-04-0d66931fe580)$ docker run -p 4000:80 -d gcr.io/qwiklabs-gcp-04-0d66931fe580/node-app:0.2
b60b197297ebbf5cca3744c0ebcf638d4 8b6abf3c2402bd6a2d19c8ccc3f7a4
docker: Error response from daemon: driver failed programming external connectivity on endpoint sleepy_gould (ebc3ed159a6afcd69d237f9f4d[illegible]
5bd): Bind for 0.0.0.0:4000 failed: port is already allocated.
student_04_e8324d4f9da3@cloudshell:~ (qwiklabs-gcp-04-0d66931fe580)$ curl http://localhost:4000
Tarima Aja Lah
student_04_e8324d4f9da3@cloudshell:~ (qwiklabs-gcp-04-0d66931fe580)$
```

7. **Test Completed Task**


Test Completed Task

Click **Check my progress** to verify your performed task. If you have successfully publish container image to Container Registry, you'll see an assessment score.

Publish your container image to Container Registry

Assessment Completed!

C. Kesimpulan

Setelah melakukan praktik seperti diatas, dapat disimpulkan bahwa mahasiswa dapat mem-build, menjalankan, dan melakukan debug container Docker. Mahasiswa juga dapat mengambil image Docker dari Docker Hub dan Google Container Registry dan juga dapat menerapkan image Docker ke Google Container Registry.