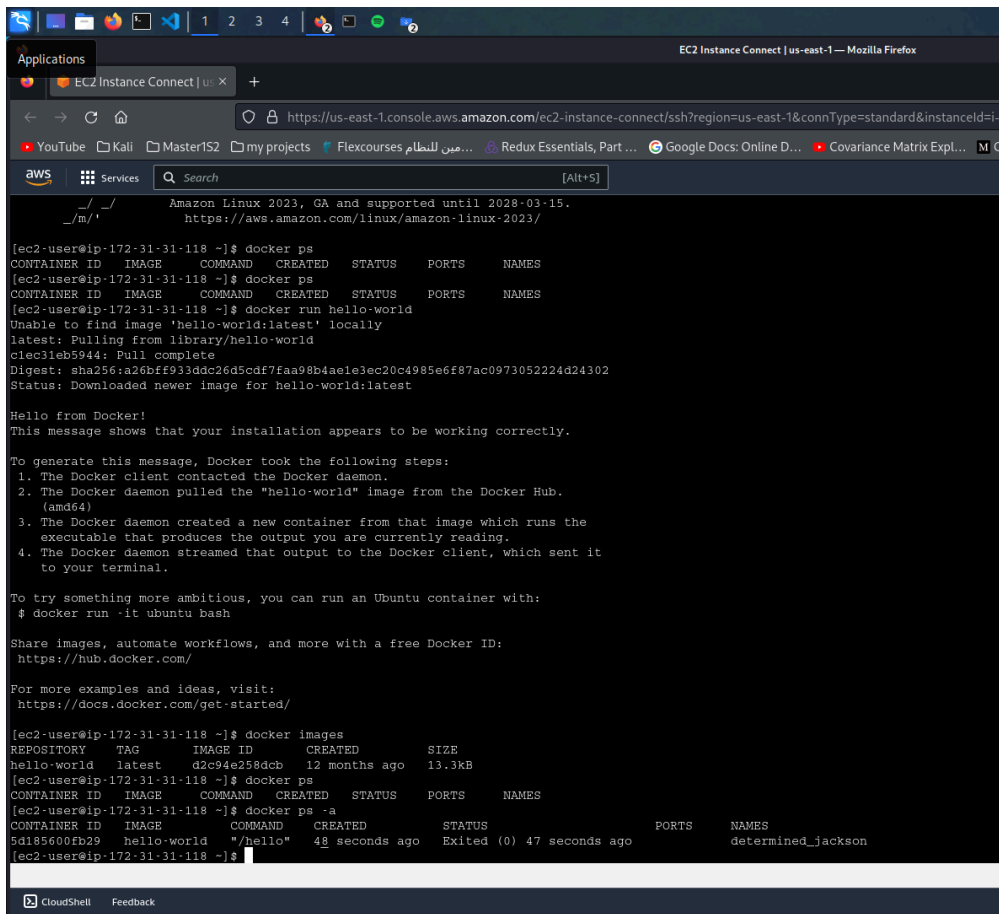


### Step 3: hello-world



The screenshot shows a terminal window connected to an Amazon Linux 2023 instance via EC2 Instance Connect. The user runs the command `docker ps`, which returns an empty table. Then, they run `docker run hello-world`. The terminal output shows that Docker pulled the `hello-world:latest` image from the Docker Hub and executed it, resulting in the message "Hello from Docker!". Below this, a list of steps explains how Docker generated this message. Finally, the user runs `docker images`, which shows the `hello-world:latest` image is now locally available.

```
Amazon Linux 2023, GA and supported until 2028-03-15.
https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@ip-172-31-31-118 ~]$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
[ec2-user@ip-172-31-31-118 ~]$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
[ec2-user@ip-172-31-31-118 ~]$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:a26bff933ddc26d5cdf7faa98b4ae1e3ec20c4985e6f87ac0973052224d24302
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.

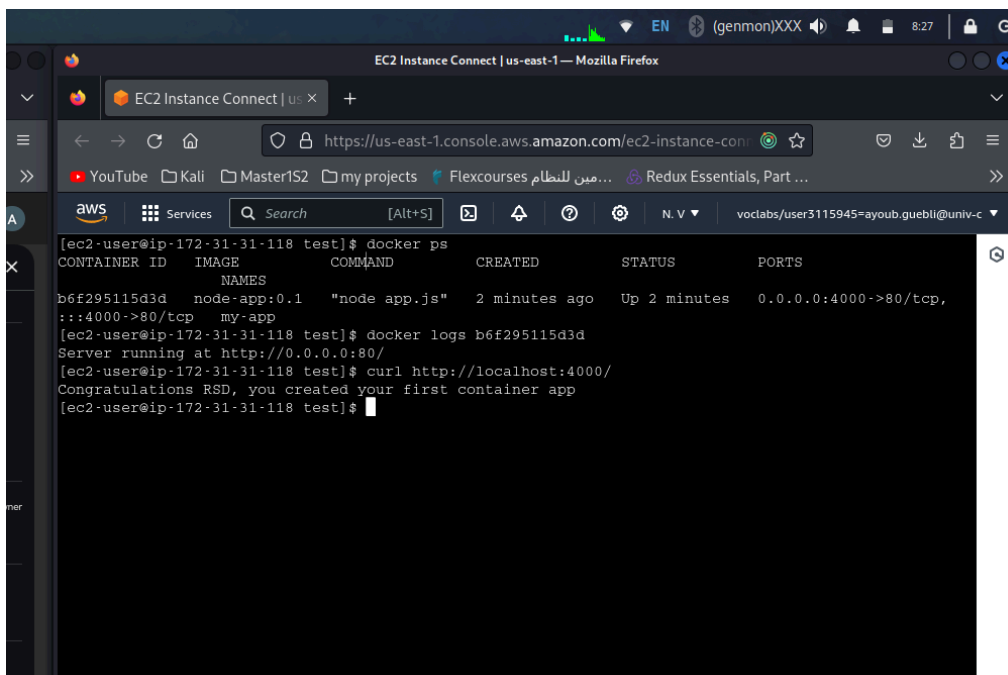
To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

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

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

[ec2-user@ip-172-31-31-118 ~]$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world    latest    42c94e258dc6   12 months ago  13.3kB
[ec2-user@ip-172-31-31-118 ~]$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
[ec2-user@ip-172-31-31-118 ~]$ docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED        STATUS      PORTS   NAMES
5d18560fb29    hello-world  "/hello"    48 seconds ago  Exited (0)  47 seconds ago  determined_jackson
[ec2-user@ip-172-31-31-118 ~]$
```

### Step 12 :

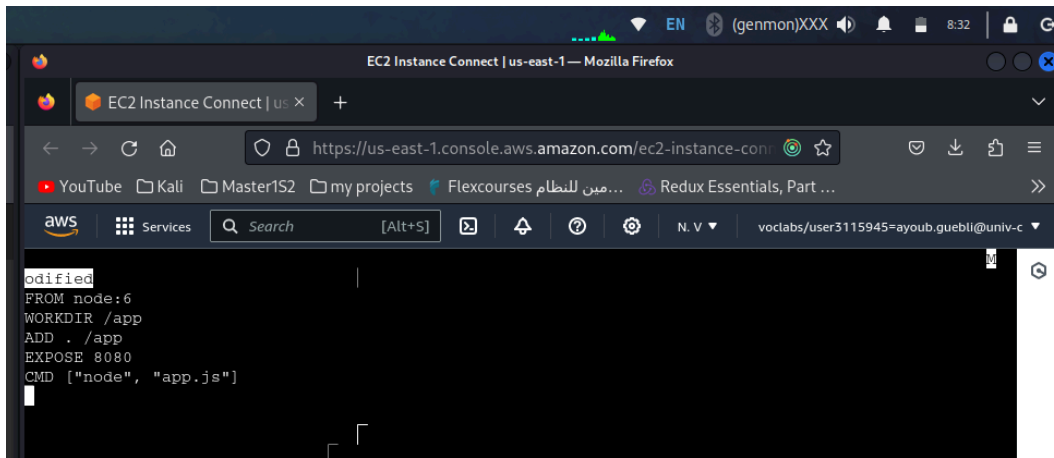


The screenshot shows a terminal window connected to an Amazon Linux 2023 instance via EC2 Instance Connect. The user runs `docker ps`, showing a container named `my-app` with ID `b6f295115d3d`. Then, they run `docker logs b6f295115d3d`, which shows the server running at `http://0.0.0.0:80/`. Finally, they run `curl http://localhost:4000/`, which returns a "Congratulations RSD, you created your first container app" message.

```
[ec2-user@ip-172-31-31-118 test]$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
b6f295115d3d   node-app:0.1  "node app.js"  2 minutes ago  Up 2 minutes  0.0.0.0:4000->80/tcp, :::4000->80/tcp  my-app
[ec2-user@ip-172-31-31-118 test]$ docker logs b6f295115d3d
Server running at http://0.0.0.0:80/
[ec2-user@ip-172-31-31-118 test]$ curl http://localhost:4000/
Congratulations RSD, you created your first container app
[ec2-user@ip-172-31-31-118 test]$
```

## Challenge:

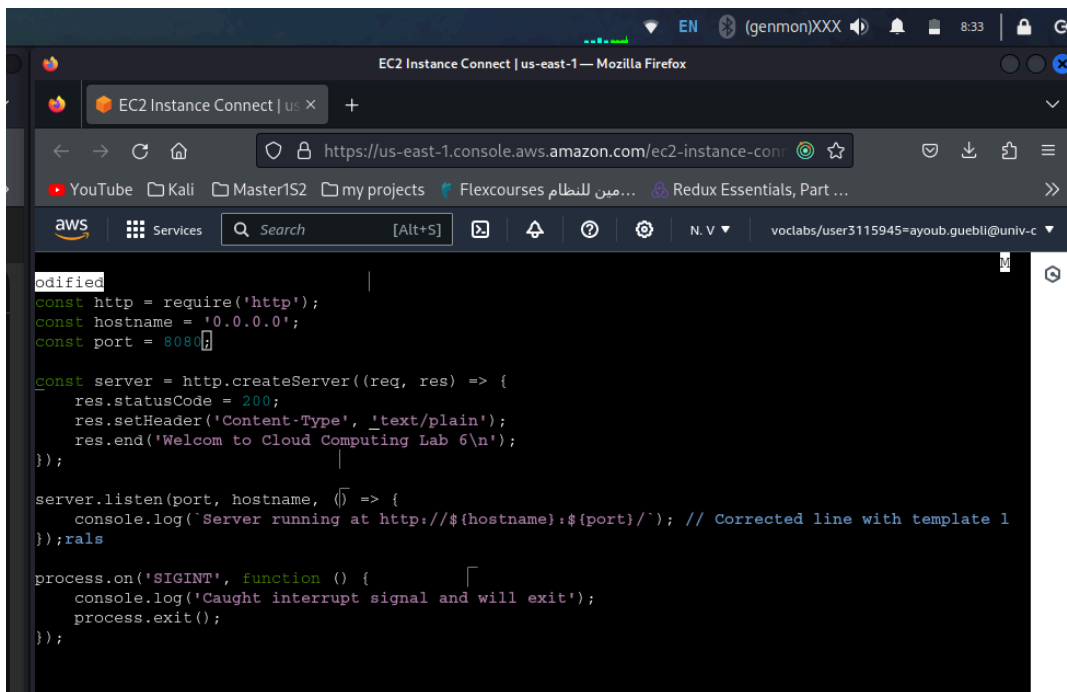
Dockerfile:



A screenshot of a code editor showing a Dockerfile. The editor has a dark theme and a sidebar on the left. The Dockerfile content is as follows:

```
FROM node:6
WORKDIR /app
ADD . /app
EXPOSE 8080
CMD ["node", "app.js"]
```

App.js:



A screenshot of a code editor showing the App.js file. The editor has a dark theme and a sidebar on the left. The App.js content is as follows:

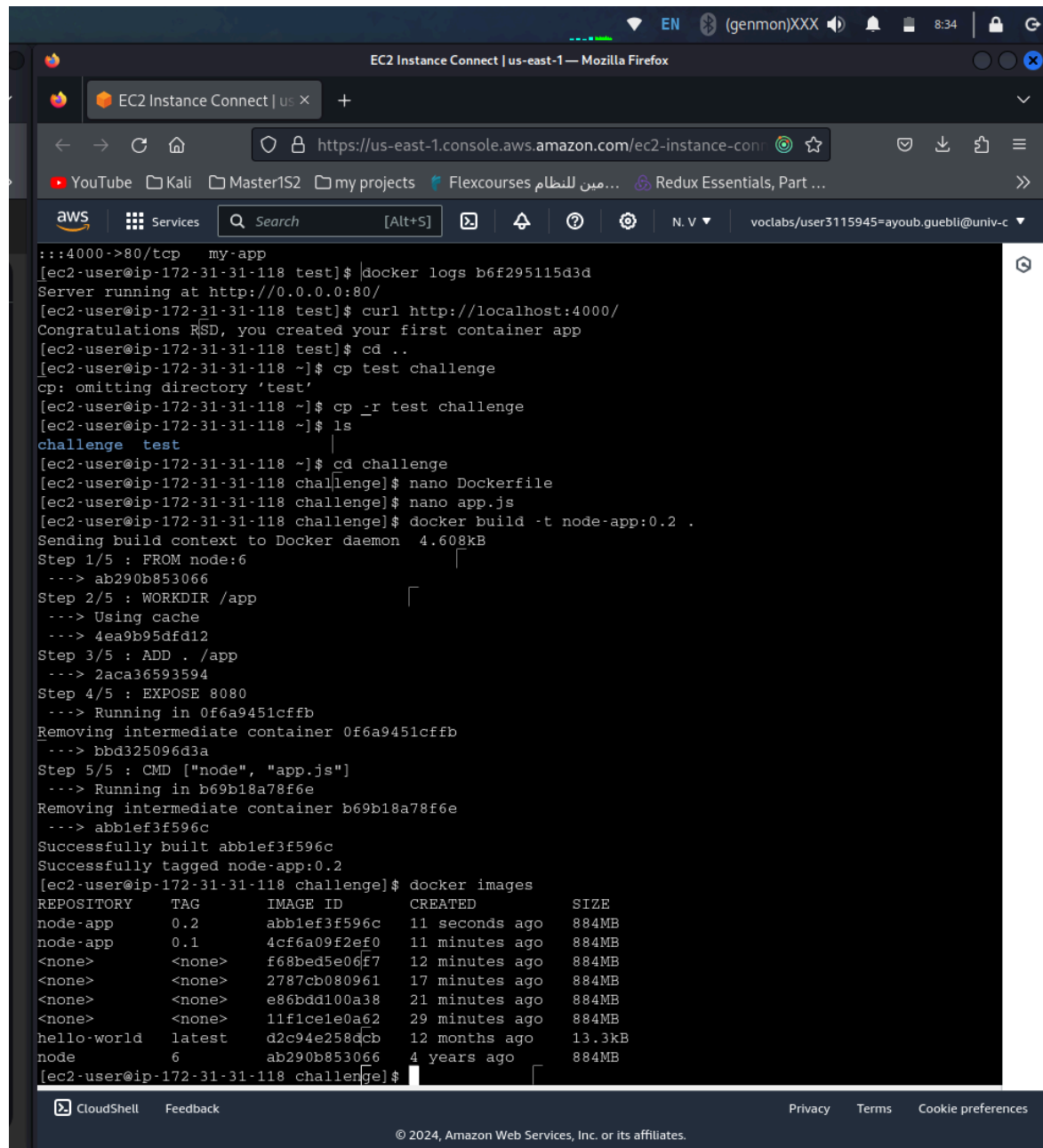
```
const http = require('http');
const hostname = '0.0.0.0';
const port = 8080;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Welcom to Cloud Computing Lab 6\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`); // Corrected line with template 1
});

process.on('SIGINT', function () {
  console.log('Caught interrupt signal and will exit');
  process.exit();
});
```

Building:



The screenshot shows a terminal window titled "EC2 Instance Connect | us-east-1 — Mozilla Firefox". The terminal output shows the following commands and results:

```
...:4000->80/tcp my-app
[ec2-user@ip-172-31-31-118 test]$ docker logs b6f295115d3d
Server running at http://0.0.0.0:80/
[ec2-user@ip-172-31-31-118 test]$ curl http://localhost:4000/
Congratulations RSD, you created your first container app
[ec2-user@ip-172-31-31-118 test]$ cd ..
[ec2-user@ip-172-31-31-118 ~]$ cp test challenge
cp: omitting directory 'test'
[ec2-user@ip-172-31-31-118 ~]$ cp -r test challenge
[ec2-user@ip-172-31-31-118 ~]$ ls
challenge test
[ec2-user@ip-172-31-31-118 ~]$ cd challenge
[ec2-user@ip-172-31-31-118 challenge]$ nano Dockerfile
[ec2-user@ip-172-31-31-118 challenge]$ nano app.js
[ec2-user@ip-172-31-31-118 challenge]$ docker build -t node-app:0.2 .
Sending build context to Docker daemon 4.608kB
Step 1/5 : FROM node:6
--> ab290b853066
Step 2/5 : WORKDIR /app
--> Using cache
--> 4ea9b95dfd12
Step 3/5 : ADD . /app
--> 2aca36593594
Step 4/5 : EXPOSE 8080
--> Running in 0f6a9451cffb
Removing intermediate container 0f6a9451cffb
--> bbd325096d3a
Step 5/5 : CMD ["node", "app.js"]
--> Running in b69b18a78f6e
Removing intermediate container b69b18a78f6e
--> abb1ef3f596c
Successfully built abb1ef3f596c
Successfully tagged node-app:0.2
[ec2-user@ip-172-31-31-118 challenge]$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
node-app	0.2	abb1ef3f596c	11 seconds ago	884MB
node-app	0.1	4cf6a09f2ef0	11 minutes ago	884MB
<none>	<none>	f68bed5e06f7	12 minutes ago	884MB
<none>	<none>	2787cb080961	17 minutes ago	884MB
<none>	<none>	e86bdd100a38	21 minutes ago	884MB
<none>	<none>	11f1ce1e0a62	29 minutes ago	884MB
hello-world	latest	d2c94e258dcb	12 months ago	13.3kB
node	6	ab290b853066	4 years ago	884MB

[ec2-user@ip-172-31-31-118 challenge]\$


At the bottom of the terminal window, there are links for "CloudShell", "Feedback", "Privacy", "Terms", and "Cookie preferences", along with a copyright notice: "© 2024, Amazon Web Services, Inc. or its affiliates."

Access to server:

```
[ec2-user@ip-172-31-31-118 challenge]$ docker run -p 4040:8080 --name my-app-2 -d node-app:0.2
5d180d7e636b03ab5a6329c06f118154bfff98fb55a3ab6ba0048070740fbfc92
[ec2-user@ip-172-31-31-118 challenge]$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
5d180d7e636b	node-app:0.2	"node app.js"	15 seconds ago	Up 14 seconds	0.0.0.0:4040->8080/tcp
b6f295115d3d	node-app:0.1	"node app.js"	11 minutes ago	Up 11 minutes	0.0.0.0:4000->80/tcp

```
[ec2-user@ip-172-31-31-118 challenge]$ docker logs 5d180d7e636b
Server running at http://0.0.0.0:8080/
[ec2-user@ip-172-31-31-118 challenge]$ curl http://localhost:4000/
Congratulations RSD, you created your first container app
[ec2-user@ip-172-31-31-118 challenge]$ curl http://localhost:4040/
Welcom to Cloud Computing Lab 6
[ec2-user@ip-172-31-31-118 challenge]$
```

 CloudShell [Feedback](#) [Privacy](#) [Terms](#) [Cookie preferences](#)  
© 2024, Amazon Web Services, Inc. or its affiliates.

By Guebli Ayoub Abdessami G2