Demo - Introduction to Containers

Nicolas El Khoury

Introduction

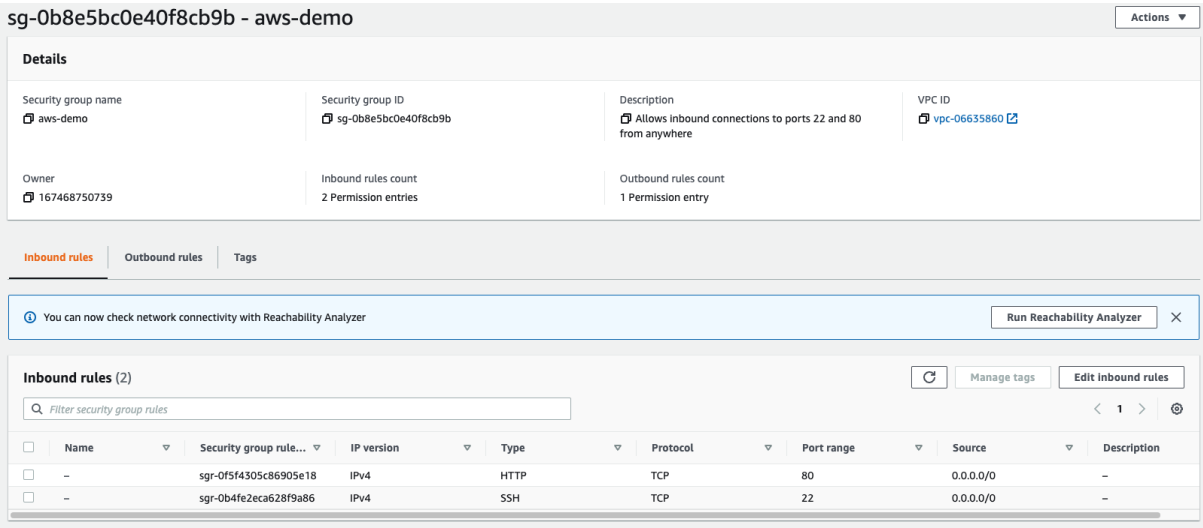
To better understand the difference between the concepts explained, we will attempt to deploy a simple HTML application on an Ubuntu EC2 machine. Then we will containerize and redeploy it. The following steps will be performed:

- Create the networking and compute resources on AWS.
- Deploy a simple HTML application on an AWS EC2 machine.
- Containerize the application.
- Store the image on AWS Elastic Container Registry.
- Deploy the containerized application.

AWS Infrastructure

Security Group

- Navigate to **AWS EC2** —> **Security Groups** —> **Create security group**, with the following parameters:
- **Security group name**: aws-demo
- **Description**: Allows inbound connections to ports 22 and 80 from anywhere
- **VPC**: default VPC
- **Inbound rules**:
  - **Rule1:**
    - **Type**: SSH
    - **Source**: Anywhere-IPv4
  - **Rule2:**
    - **Type**: HTTP
    - **Source**: Anywhere-IPv4



Key pair

- Navigate to **AWS EC2** —> **Key pairs** —> **Create key pair**, with the following parameters:
- Name: aws-demo
- Private key file format: .pem

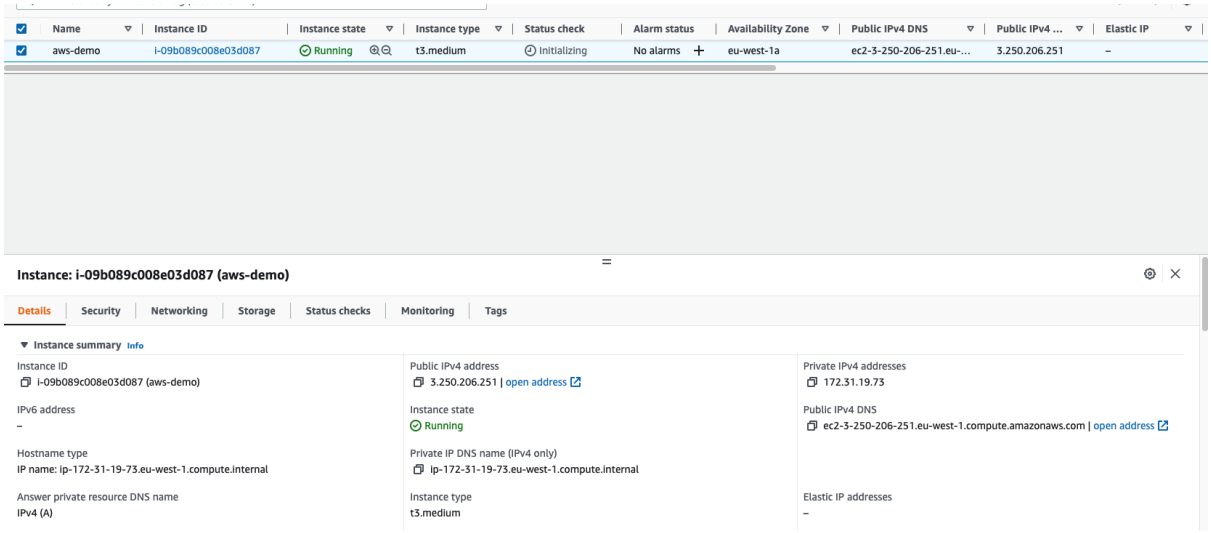
```
# Create a hidden directory
mkdir ~/.keypairs/aws-demo
# Move the key to the created directory
mv ~/Downloads/aws-demo.pem ~/.keypairs/aws-demo/
# Change the permissions of the key
sudo chmod 400 ~/.keypairs/aws-demo/aws-demo.pem
```

```
devops-beyond-limits@DBL aws-demo % ls -la
total 8
drwxr-xr-x@  3 devops-beyond-limits  staff    96 Dec 14 11:06 .
drwxr-xr-x  10 devops-beyond-limits  staff   320 Dec 14 11:05 ..
-r--------@  1 devops-beyond-limits  staff  1674 Dec 14 11:03 aws-demo.pem
```

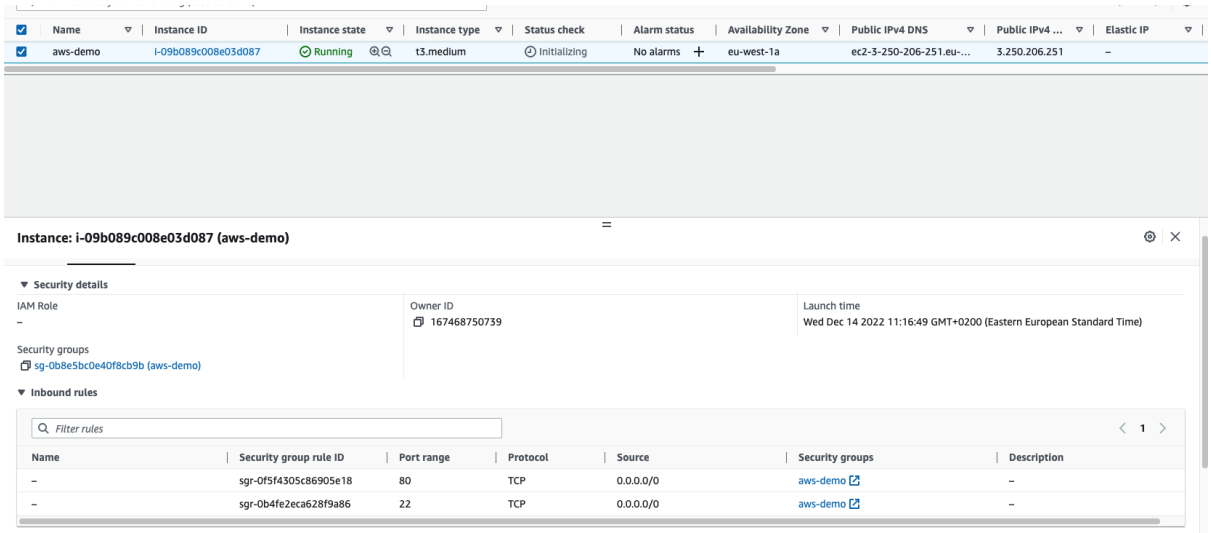
AWS EC2 Machine

- Navigate to **AWS EC2** —> **instances** —> **Launch instances**, with the following parameters:
- **Name**: aws-demo
- **AMI**: Ubuntu Server 20.04 LTS (HVM), SSD Volume Type
- **Instance Type**: t3.medium (t3.micro can be used for free tier, but may suffer from performance issues)
- **Key pair name**: aws-demo
- **Network Settings**:
  - **Select existing security group**: aws-demo
- **Configure storage**: 1 x 25 GiB gp2 Root volume

Leave the rest as defaults and launch the instance.



An EC2 VM is created, and is assigned both a private and a public IPv4 addresses.



- Telnet is one way to ensure the machine is accessible on ports **22** and **80:**

```
# Make sure to replace the machine's IP with the one attributed to your machine
telnet 3.250.206.251 22
telnet 3.250.206.251 80
```

```
devops-beyond-limits@DBL aws-demo % telnet 3.250.206.251 22
Trying 3.250.206.251...
Connected to ec2-3-250-206-251.eu-west-1.compute.amazonaws.com.
Escape character is '^]'.
SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.5

Invalid SSH identification string.
Connection closed by foreign host.
devops-beyond-limits@DBL aws-demo % telnet 3.250.206.251 80
Trying 3.250.206.251...
Connected to ec2-3-250-206-251.eu-west-1.compute.amazonaws.com.
Escape character is '^]'.
Connection closed by foreign host.
devops-beyond-limits@DBL aws-demo %
```

- SSH to the machine, using the key pair created: `ssh ubuntu@3.250.206.251 -i ~/.keypairs/aws-demo/aws-demo.pem`

```
devops-beyond-limits@DBL aws-demo % ssh ubuntu@3.250.206.251 -i ~/.keypairs/aws-demo/aws-demo.pem
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.15.0-1026-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed Dec 14 09:23:08 UTC 2022

  System load:  0.07               Processes:             100
  Usage of /:   6.5% of 24.05GB    Users logged in:       0
  Memory usage: 5%                 IPv4 address for ens5: 172.31.19.73
  Swap usage:   0%


0 updates can be applied immediately.


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
New release '22.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Wed Dec 14 09:23:00 2022 from 178.135.1.7
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-19-73:~$
```

Application Deployment on the EC2 machine

Application Code

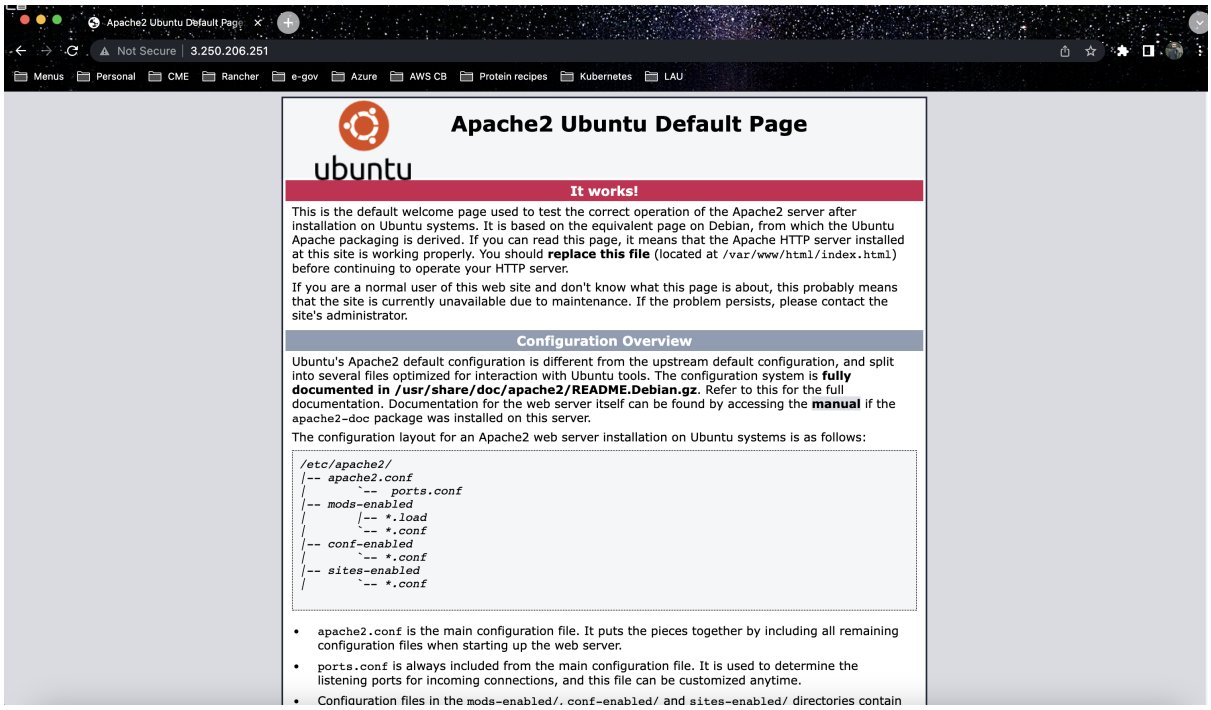
The application to be deployed is a simple HTML document:

```
<!DOCTYPE html>
<html>
    <head>
        <title>My First Application</title>
    </head>
    <body>
```

```
        <p>I have no idea what I'm doing.</p>
    </body>
</html>
```

Apache2 Installation

- Update the local package index to reflect the latest upstream changes: `sudo apt-get update`
- Install the **Apache2** package: `sudo apt-get install -y apache2`
- Check if the service is running: `sudo service apache2 status`
- Verify that the deployment worked by hitting the public IP of the machine:



Application Deployment

```
# Create a directory
sudo mkdir /var/www/myfirstapp
# Change the ownership to www-data
sudo chown -R www-data:www-data /var/www/myfirstapp
# Change the directory permissions
sudo chmod -R 755 /var/www/myfirstapp
# Create the index.html file and paste the code in it
sudo nano /var/www/myfirstapp/index.html
# Change the owership to www-data
sudo chown -R www-data:www-data /var/www/myfirstapp/index.html
# Create the log directory
sudo mkdir /var/log/myfirstapp
```

```
# Change the ownership of the directory
sudo chown -R www-data:www-data /var/log/myfirstapp/
```

Virtual Host

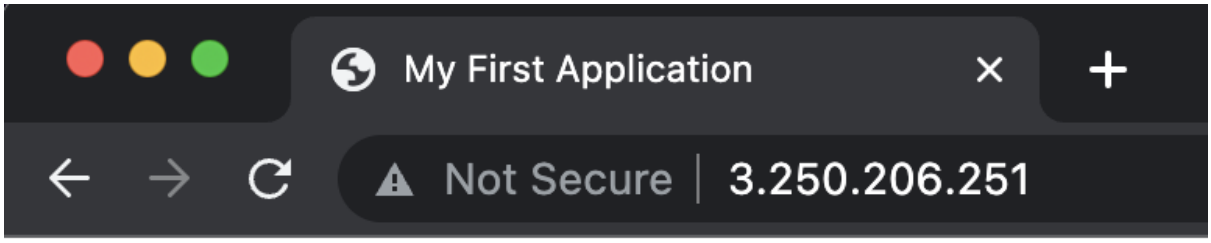
- Create the virtual host file: `sudo nano /etc/apache2/sites-available/myfirstapp.conf`
- Paste the following:

```
<VirtualHost *:80>
    DocumentRoot /var/www/myfirstapp
    ErrorLog /var/log/myfirstapp/error.log
    CustomLog /var/log/myfirstapp/requests.log combined
</VirtualHost>
```

- Enable the configuration:

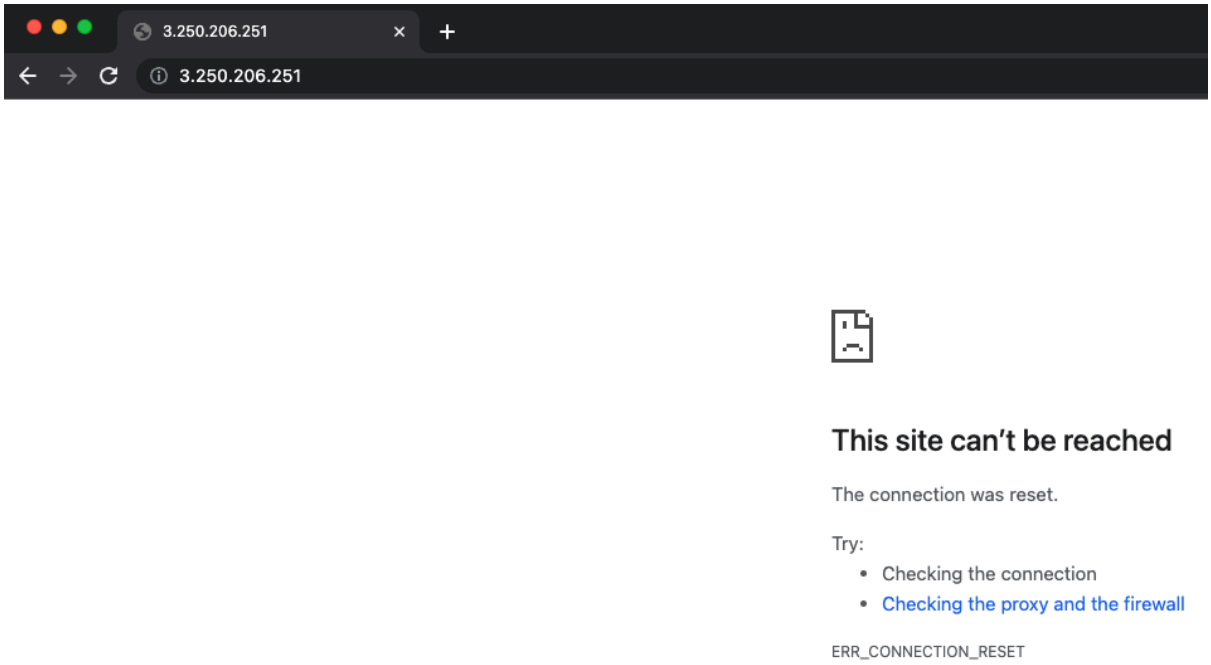
```
# Enable the site configuration
sudo a2ensite myfirstapp.conf
# Disable the default configuration
sudo a2dissite 000-default.conf
# Test the configuration
sudo apache2ctl configtest
# Restart apache
sudo systemctl restart apache2
```

- Perform a request on the server. The response will now return the HTML document created:



I have no idea what I'm doing.

- Stop the apache webserver: `sudo service apache2 stop`



Application Deployment using containers

IAM Role

- Navigate to **IAM** —> **Roles** —> **Create Role**, with the following parameters:
- **Trusted entity type:** AWS Service
- **Common use cases**: EC2
- **Permissions policies:** AdministratorAccess
- **Role Name**: aws-demo

Attach this role to the EC2 Machine: **Actions —> Security —> Modify IAM Role**

AWS CLI

```
# Update the package repository
sudo apt-get update
# Install unzip on the machine
sudo apt-get install -y unzip
# Download the zipped package
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" \
-o "awscliv2.zip"
# unzip the package
unzip awscliv2.zip
# Run the installer
sudo ./aws/install
```

- Ensure the AWS CLI is installed by checking the version: `aws --version`

```
ubuntu@ip-172-31-19-73:~$ aws --version
aws-cli/2.9.6 Python/3.9.11 Linux/5.15.0-1026-aws exe/x86_64.ubuntu.20 prompt/off
ubuntu@ip-172-31-19-73:~$
```

Docker Installation

```
# Update the package index and install the required packages
sudo apt-get update
sudo apt-get install -y ca-certificates curl gnupg lsb-release

# Add Docker's official GPG key:
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg \
| sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

# Set up the repository
echo "deb [arch=$(dpkg --print-architecture) \
signed-by=/etc/apt/keyrings/docker.gpg] \
https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list \
> /dev/null

# Update the package index again
sudo apt-get update

# Install the latest version of docker
sudo apt-get install -y docker-ce docker-ce-cli containerd.io \
docker-compose-plugin

# Add the Docker user to the existing User's group
#(to run Docker commands without sudo)
sudo usermod -aG docker $USER
```

To validate that Docker is installed and the changes are all applied, restart the SSH session, and query the docker containers: `docker ps -a`. A response similar to the one below indicates the success of the installation.

```
ubuntu@ip-172-31-19-73:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
ubuntu@ip-172-31-19-73:~$
```

Base Image

- Pull the Apache2 Docker Image: `docker pull httpd:2.4-alpine`.
- List the available images `docker images`.

```
ubuntu@ip-172-31-19-73:~$ docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
ubuntu@ip-172-31-19-73:~$ docker pull httpd:2.4-alpine
2.4-alpine: Pulling from library/httpd
c158987b0551: Pull complete
af0dc97e3e7a: Pull complete
7a4f45a5d61d: Pull complete
3b6adf47f20b: Pull complete
a7c0837131ea: Pull complete
82cf25aabba6: Pull complete
Digest: sha256:86ed18b4670b3be349e62f05c34bf0c28f3e0a73732969c417fd53e04af807f4
Status: Downloaded newer image for httpd:2.4-alpine
docker.io/library/httpd:2.4-alpine
ubuntu@ip-172-31-19-73:~$ docker images
REPOSITORY   TAG          IMAGE ID       CREATED       SIZE
httpd        2.4-alpine   4e7c9ee81ce6   13 days ago   56.9MB
ubuntu@ip-172-31-19-73:~$
```

- Create a Docker container: `docker run -d --name myfirstcontainer -p 80:80 httpd:2.4-alpine`
- Ensure that the container is successfully running: `docker ps -a`
- Monitor the container logs: `docker logs -f myfirstcontainer`

```
ubuntu@ip-172-31-21-19:~$ docker run -d --name myfirstcontainer -p 81:80 httpd:2.4-alpine
ddb4bcef66cf1b3a426ee141dc1c64dc8782f27e24c9372a450520ea567f1028
ubuntu@ip-172-31-21-19:~$ docker ps -a
CONTAINER ID   IMAGE              COMMAND              CREATED         STATUS         PORTS                                   NAMES
ddb4bcef66cf   httpd:2.4-alpine   "httpd-foreground"   7 seconds ago   Up 6 seconds   0.0.0.0:81->80/tcp, :::81->80/tcp   myfirstcontainer
ubuntu@ip-172-31-21-19:~$ docker logs -f myfirstcontainer
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message
[Tue Nov 08 16:24:28.914558 2022] [mpm_event:notice] [pid 1:tid 140713108110152] AH00489: Apache/2.4.54 (Unix) configured -- resuming normal operations
[Tue Nov 08 16:24:28.914693 2022] [core:notice] [pid 1:tid 140713108110152] AH00094: Command line: 'httpd -D FOREGROUND'
```

- Attempt to make a request to the container, using the machine's public IP and port 80: `http://<MACHINE IP>:80`

Customize the Base Image

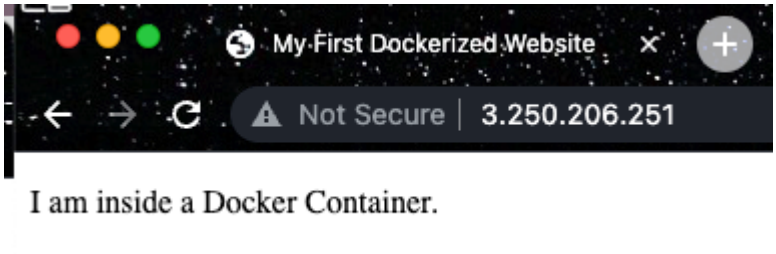
In this example, the following simple HTML page representing a website will be added, thus creating a custom image.

```
<!DOCTYPE html>
<html>
    <head>
        <title>My First Dockerized Website</title>
    </head>
```

```
    <body>
        <p>I am inside a Docker Container.</p>
    </body>
</html>
```

- Create an interactive `sh` shell on the container: `docker exec -it myfirstcontainer sh`.
- Navigate to the designated directory `cd /usr/local/apache2/htdocs/`. The directory already has a file named `index.html` which contains the default Apache page loaded above. Modify it to include the custom HTML page above, and hit the container again: `http://MACHINE IP>:80`

Clearly, the image shows that the changes have been reflected.



Create a Custom Image

- The changes performed will not persist, especially when the container crashes. As a matter of fact, by default, containers are ephemeral. To verify it, remove the container and start it again:

```
docker rm -f myfirstcontainer
docker ps -a
docker run -d --name myfirstcontainer -p 80:80 httpd:2.4-alpine
```

- Now hit the container again `http://<MACHINE IP>:80`.
- The changes performed disappeared. To persist the changes, a custom image must be built. The custom image is a snapshot of the container after adding the custom website. Repeat the steps above to add the HTML page, and ensure the container is returning the new page again.
- Create a Docker Image from the running container: `docker commit myfirstcontainer`.
- Name and tag the image: `docker tag <image ID> custom-httpd:v1`.

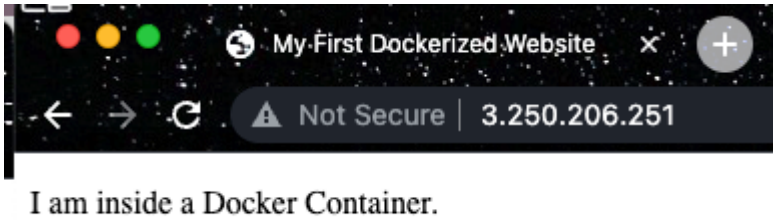
```
ubuntu@ip-172-31-19-73:~$ docker images
REPOSITORY   TAG          IMAGE ID       CREATED          SIZE
<none>       <none>       66a3cb4fd72f   7 seconds ago    56.9MB
httpd        2.4-alpine   4e7c9ee81ce6   13 days ago      56.9MB
ubuntu@ip-172-31-19-73:~$ docker tag 66a3cb4fd72f custom-httpd:v2
ubuntu@ip-172-31-19-73:~$ docker images
REPOSITORY     TAG          IMAGE ID       CREATED               SIZE
custom-httpd   v2           66a3cb4fd72f   About a minute ago    56.9MB
httpd          2.4-alpine   4e7c9ee81ce6   13 days ago           56.9MB
ubuntu@ip-172-31-19-73:~$
```

- Remove the old container, and create a new one using the new image:

```
docker rm -f myfirstcontainer
docker run -d --name mysecondcontainer -p 80:80 custom-httpd:v1
```

```
ubuntu@ip-172-31-19-73:~$ docker ps
CONTAINER ID   IMAGE            COMMAND              CREATED          STATUS         PORTS                                   NAMES
08fddb907c98   custom-httpd:v2  "httpd-foreground"   5 seconds ago    Up 5 seconds   0.0.0.0:80->80/tcp, :::80->80/tcp       mysecondcontainer
ubuntu@ip-172-31-19-73:~$
```

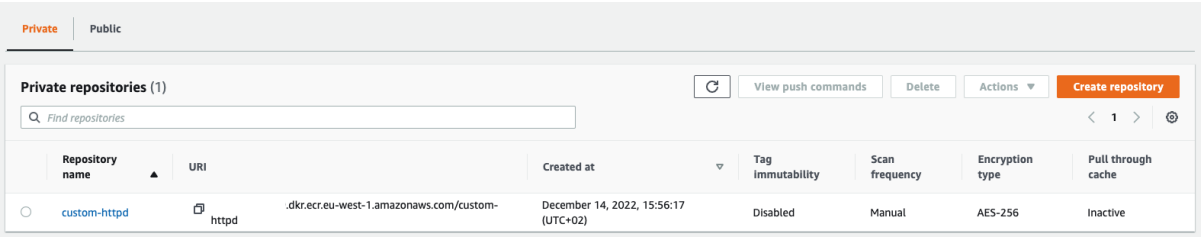
- Hitting the machine on port 80 should return the new HTML page now no matter how many times the container is destroyed and created.



Push the Image to AWS ECR

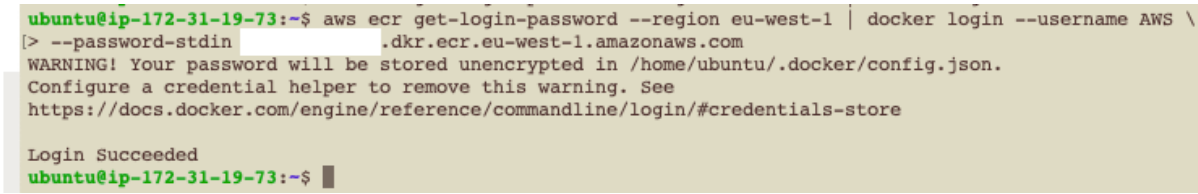
- Create a Container repository on AWS ECR, navigate to **Amazon ECR** —> **Repositories** —> **Private** —> **Create repository**, with the following parameters:
- **Visibility Settings**: Private
- **Repository name:** custom-httpd

Leave the rest as defaults and create the repository.



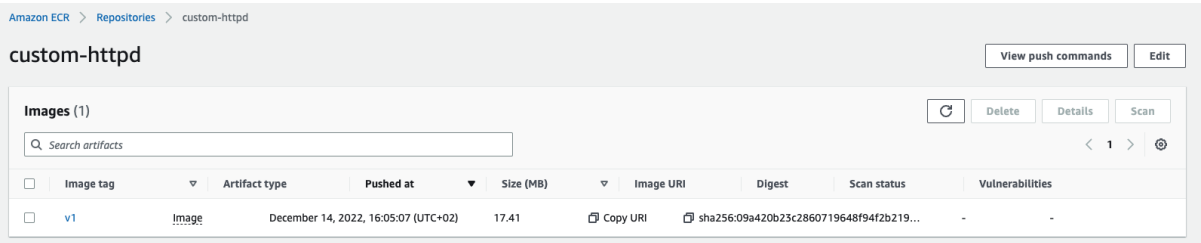
- First, login to the ECR from the VM:

```
aws ecr get-login-password --region <REGION ID> | docker login --username AWS \
--password-stdin <ACCOUNT ID>.dkr.ecr.<REGION ID>.amazonaws.com
```



- Tag the image with that found in the ECR repository:

```
# Tag the image with the correct repository name
docker tag custom-httpd:v1 \
<ACCOUNT ID>.dkr.ecr.<REGION ID>.amazonaws.com/custom-httpd:v1
# Push the image
docker push <ACCOUNT ID>.dkr.ecr.<REGION ID>.amazonaws.com/custom-httpd:v1
```



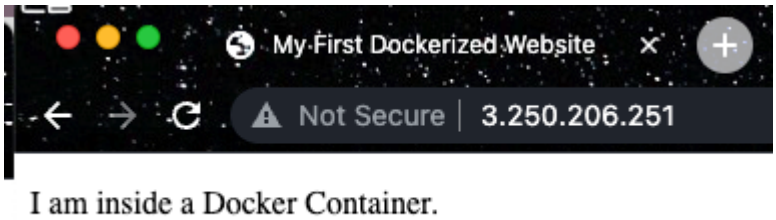
- Remove all the images and containers from the VM.

```
# Delete all the containers from the server
docker rm -f $(docker ps -a)
# Delete all the images from the server
docker rmi -f $(docker images)
# List all the available images and containers (should return empty)
docker images
docker ps -a
```

- create a third container, but this time, reference the image located in the ECR:

  `docker run -d --name mythirdcontainer -p 80:80 <ACCOUNT ID>.dkr.ecr.<REGION ID>.amazonaws.com/custom-httpd:v1`

- Finally, hit the server again `http://3.250.206.251:80`



- Remove the images and containers from the VM:

```
# Delete all the containers from the server
docker rm -f $(docker ps -a -q)
# Delete all the images from the server
docker rmi -f $(docker images)
# List all the available images and containers (should return empty)
docker images
docker ps -a
```

Create a Docker image using Dockerfiles

- Create a temporary directory: `mkdir ~/tempDir`
- Place the application code inside the directory in a file called `index.html`

```
<!DOCTYPE html>
<html>
    <head>
        <title>My Final Dockerized Website</title>
    </head>
    <body>
        <p>I am Dockerized using a Dockerfile.</p>
    </body>
</html>
```

- Create a Dockerfile next to the **index.html** file, with the following content:

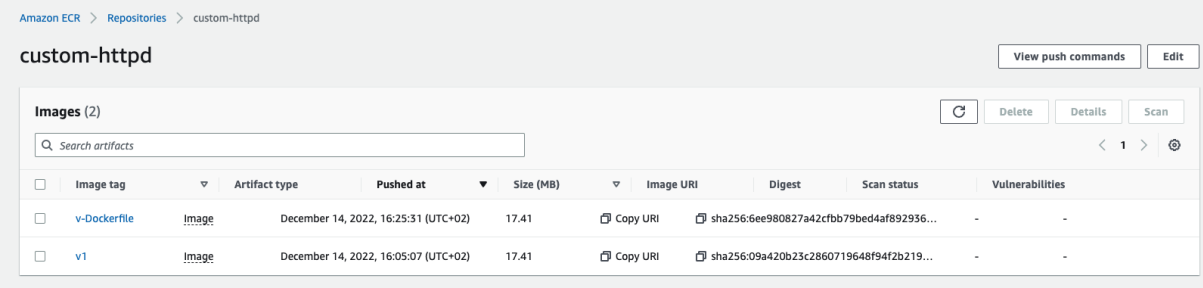
```
FROM httpd:2.4-alpine
COPY index.html /usr/local/apache2/htdocs/
```

The resultant directory should look as follows:

```
ubuntu@ip-172-31-19-73:~/tempDir$ ls -lah
total 16K
drwxrwxr-x 2 ubuntu ubuntu 4.0K Dec 14 14:20 .
drwxr-xr-x 8 ubuntu ubuntu 4.0K Dec 14 14:20 ..
-rw-rw-r-- 1 ubuntu ubuntu   65 Dec 14 14:20 Dockerfile
-rw-rw-r-- 1 ubuntu ubuntu  179 Dec 14 14:20 index.html
ubuntu@ip-172-31-19-73:~/tempDir$ cat Dockerfile
FROM httpd:2.4-alpine
COPY index.html /usr/local/apache2/htdocs/
ubuntu@ip-172-31-19-73:~/tempDir$ cat index.html
<!DOCTYPE html>
<html>
    <head>
        <title>My Final Dockerized Website</title>
    </head>
    <body>
        <p>I am Dockerized using a Dockerfile.</p>
    </body>
</html>
ubuntu@ip-172-31-19-73:~/tempDir$
```

- Build the Docker Image: `docker build -f Dockerfile -t <ACCOUNT ID>.dkr.ecr.<REGION ID>.amazonaws.com/custom-httpd:v-Dockerfile .`
- Push the image to the ECR: `docker push <ACCOUNT ID>.dkr.ecr.<RGION ID>.amazonaws.com/custom-httpd:v-Dockerfile`

```
ubuntu@ip-172-31-19-73:~/tempDir$ docker build -f Dockerfile -t [redacted].dkr.ecr.eu-west-1.amazonaws.com/custom-httpd:v-Dockerfile .
Sending build context to Docker daemon  3.072kB
Step 1/2 : FROM httpd:2.4-alpine
2.4-alpine: Pulling from library/httpd
c158987b0551: Pull complete
af0dc97e3e7a: Pull complete
7a4f45a5d61d: Pull complete
3b6adf47f20b: Pull complete
a7c0837131ea: Pull complete
82cf25aabba6: Pull complete
Digest: sha256:86ed18b4670b3be349e62f05c34bf0c28f3e0a73732969c417fd53e04af807f4
Status: Downloaded newer image for httpd:2.4-alpine
 ---> 4e7c9ee81ce6
Step 2/2 : COPY index.html /usr/local/apache2/htdocs/
 ---> ed95b86e218e
Successfully built ed95b86e218e
Successfully tagged [redacted].dkr.ecr.eu-west-1.amazonaws.com/custom-httpd:v-Dockerfile
ubuntu@ip-172-31-19-73:~/tempDir$ docker images
REPOSITORY                                          TAG            IMAGE ID       CREATED          SIZE
[redacted].dkr.ecr.eu-west-1.amazonaws.com/custom-httpd   v-Dockerfile   ed95b86e218e   7 seconds ago    56.9MB
httpd                                               2.4-alpine     4e7c9ee81ce6   13 days ago      56.9MB
ubuntu@ip-172-31-19-73:~/tempDir$ docker push [redacted].dkr.ecr.eu-west-1.amazonaws.com/custom-httpd:v-Dockerfile
The push refers to repository [[redacted].dkr.ecr.eu-west-1.amazonaws.com/custom-httpd]
ad64793e9b48: Pushed
c9d552c45814: Layer already exists
f327819305ff: Layer already exists
74756a75a40e: Layer already exists
9a6eb5d4ab65: Layer already exists
1d9520684849: Layer already exists
ded7a220bb05: Layer already exists
v-Dockerfile: digest: sha256:6ee980827a42cfbb79bed4af892936b58119d763794b7aa0cb2d43a269bcc902 size: 1779
ubuntu@ip-172-31-19-73:~/tempDir$
```



- Simulate a fresh installation of the image, remove all the containers and images from the server, and create a final container from the newly pushed image:

```
# Remove existing containers
docker rm -f $(docker ps -a -q)
# Remove the images
docker rmi -f $(docker images)
# Create the final container
docker run -d --name myfinalcontainer -p 80:80 <ACCOUNT ID>.dkr.ecr.<REGION ID>.amazon
aws.com/custom-httpd:v-Dockerfile
```

Hit the machine via its IP