

Project Title: Web Application Update & Deployment- Mini Capstone 1

Project Tasks:

1. Version Control & Code Management:

- I created my GitHub account and created a new repository.
- I used git clone <https://github.com/babaJay23/capstone-aws.git> command to clone the project codebase on my laptop.
- I removed the origin, set my repository as origin and pushed the code to the main branch on my repository. Commands are listed below.

```
git clone https://github.com/babaJay23/capstone-aws.git
git remote remove origin
git remote -v
git remote add origin https://github.com/priyajanani10/capstone-aws.git
git remote -v
git push
git push --set-upstream origin main
git config --global user.email "priyajanani10@gmail.com"
git config --global user.name "Priya Janani Sreenivasan"
git status
git checkout -b website_updates
```

2. Web Application Update:

Using three separate commits on branch website_updates I,

- Updated the page and section titles.
- Deleted the image wr-home-top.jpg
- Added a new unicorn image wr-home-top.jpg
- Pushed the branch website updates to my repository and created a pull request to merge the commits into main. I merged the commits using a merge commit.

The screenshot shows the GitHub repository interface for the 'main' branch. It displays a list of commits grouped by date:

- Commits on Jul 10, 2023:**
 - Adding Dockerfile (by priyajanani10, committed 1 hour ago)
- Commits on Jul 9, 2023:**
 - update section title for CD test. (by priyajanani10, committed yesterday)
 - Merge pull request #1 from priyajanani10/website_updates ... (by priyajanani10, committed yesterday) - This commit is marked as Verified.
 - Add new wr-home-top.jpg (by priyajanani10, committed yesterday)
 - remove wr-home-top.jpg image (by priyajanani10, committed yesterday)
 - Updated index.html with new page and section titles. (by priyajanani10, committed yesterday)
- Commits on Jun 26, 2023:**
 - deletions (by Jamal Ali authored and Jamal Ali committed 2 weeks ago)
 - deletion (by Jamal Ali authored and Jamal Ali committed 2 weeks ago)

3. Containerization:

- I created a Dockerfile.

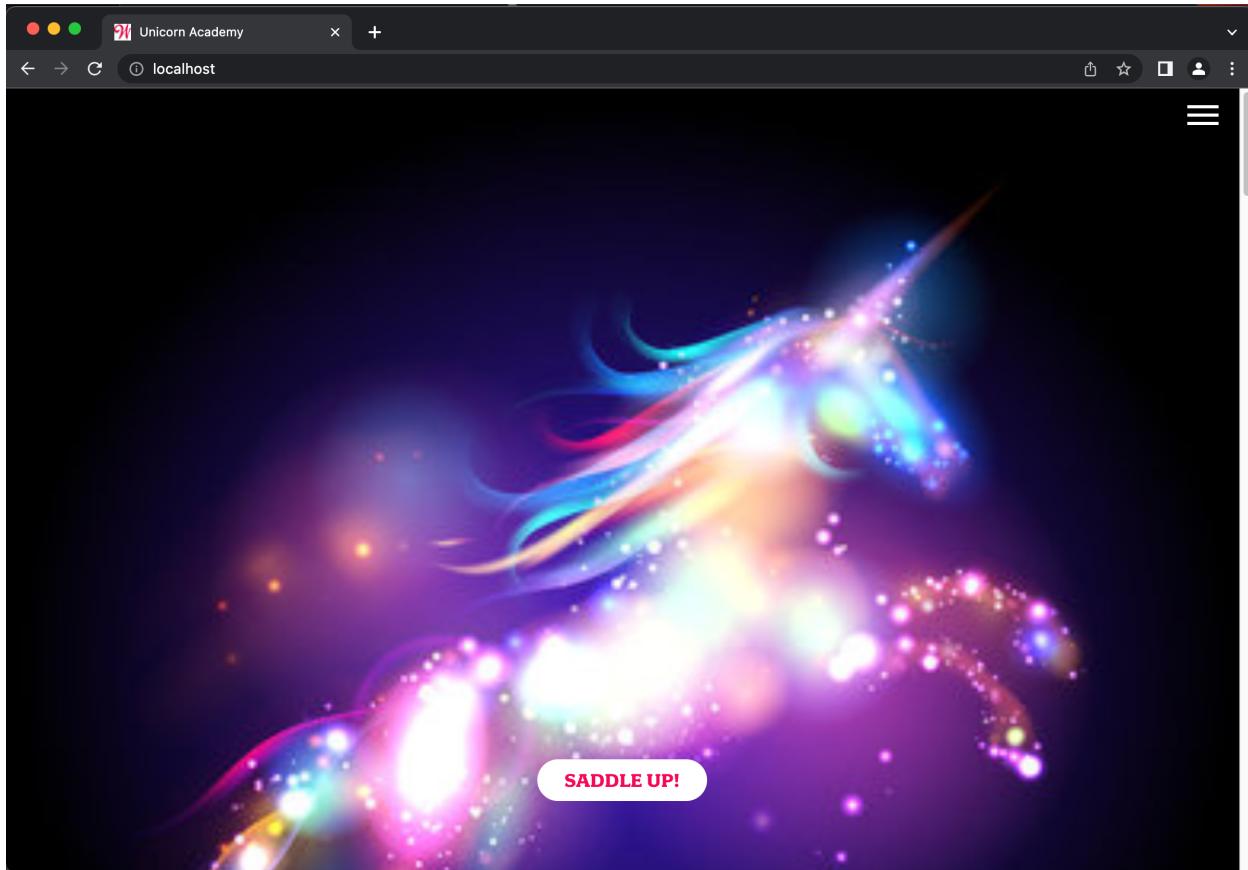
```
1 FROM nginx
2 COPY . /usr/share/nginx/html
3 EXPOSE 80
4 CMD ["nginx", "-g", "daemon off;"]
```

- I committed the Docker file to my git repository.
 - I built and tested my docker image locally.

```
[janani@Priyas-Air capstone-aws % docker build -t capstone-aws .
[+] Building 1.9s (8/8) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 159B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/nginx:latest
=> [auth] library/nginx:pull token for registry-1.docker.io
=> [internal] load build context
=> => transferring context: 25.30kB
=> CACHED [1/2] FROM docker.io/library/nginx@sha256:08bc36ad52474e528cc1ea3426b5e3f4bad8a130318e3140d6cfe29c8892c7ef
=> [2/2] COPY . /usr/share/nginx/html
=> exporting to image
=> => exporting layers
=> => writing image sha256:3085b7e624e7e84898f1767d23bb4f7a675060c9052dc9ee969fff08a862b8f
=> => naming to docker.io/library/capstone-aws

What's Next?
  View summary of image vulnerabilities and recommendations → docker scout quickview
[janani@Priyas-Air capstone-aws % docker run -dp 80:80 capstone-aws
b618266226b8c91ca3090ac14d43985b249f7eb7aa78ec248aa51bdcb8022b736
```

- I checked to see if my container is running using localhost:80 and my updated website showed up.



4. Cloud Deployment:

- I launched a new EC2 instance, created a new key pair and selected the check box to allow anyone from the internet to access port 80.

| Instance summary for i-0591628d07608c891 (capstone-aws) Info | | |
|---|---|---|
| Updated less than a minute ago | | |
| Instance ID i-0591628d07608c891 (capstone-aws) | Public IPv4 address 44.201.216.229 open address | Private IPv4 addresses 172.31.88.86 |
| IPv6 address - | Instance state Running | Public IPv4 DNS ec2-44-201-216-229.compute-1.amazonaws.com open address |
| Hostname type IP name: ip-172-31-88-86.ec2.internal | Private IP DNS name (IPv4 only) ip-172-31-88-86.ec2.internal | Elastic IP addresses - |
| Answer private resource DNS name IPv4 (A) | Instance type t2.micro | AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. Learn more |
| Auto-assigned IP address 44.201.216.229 [Public IP] | VPC ID vpc-0c5cf0e44a50a41d6 | Auto Scaling Group name - |
| IAM Role - | Subnet ID subnet-0c7bdaa43cc0d89f4 | |
| IMDSv2 Required | | |
| Details Security Networking Storage Status checks Monitoring Tags | | |
| ▼ Instance details Info | | |
| Platform Amazon Linux (Inferred) | AMI ID ami-06ca3ca175f37dd66 | Monitoring disabled |
| Platform details Linux/UNIX | AMI name al2023-ami-2023.1.20230705.0-kernel-6.1-x86_64 | Termination protection Disabled |
| Stop protection Disabled | Launch time Mon Jul 10 2023 21:38:59 GMT-0700 (Pacific) | AMI location amazon/al2023-ami-2023.1.20230705.0-kernel-6.1- |

- I connected to the EC2 instance using ssh, I changed the user permissions using the following commands

```
chmod 400 vockey2-5.pem
ssh -i "vokey2-5.pem" euser@ec2-44-201-216-229.compute-1.amazonaws.com
```

```
[janani@Priyas-Air Desktop % ssh -i "vokey2-5.pem" ec2-user@ec2-44-201-216-229.compute-1.amazonaws.com
'`#_
~\_\_ #####_      Amazon Linux 2023
~~ \_\#\#\#\\
~~ \#\#\#
~~ #/ /-- https://aws.amazon.com/linux/amazon-linux-2023
~~ V~ ,`-->
~~ ._. /`/
~~ /_ /`/
~/m/'`_
Last login: Tue Jul 11 04:42:33 2023 from 98.33.83.15
```

- I installed git, docker and started the docker service. The commands I used are,

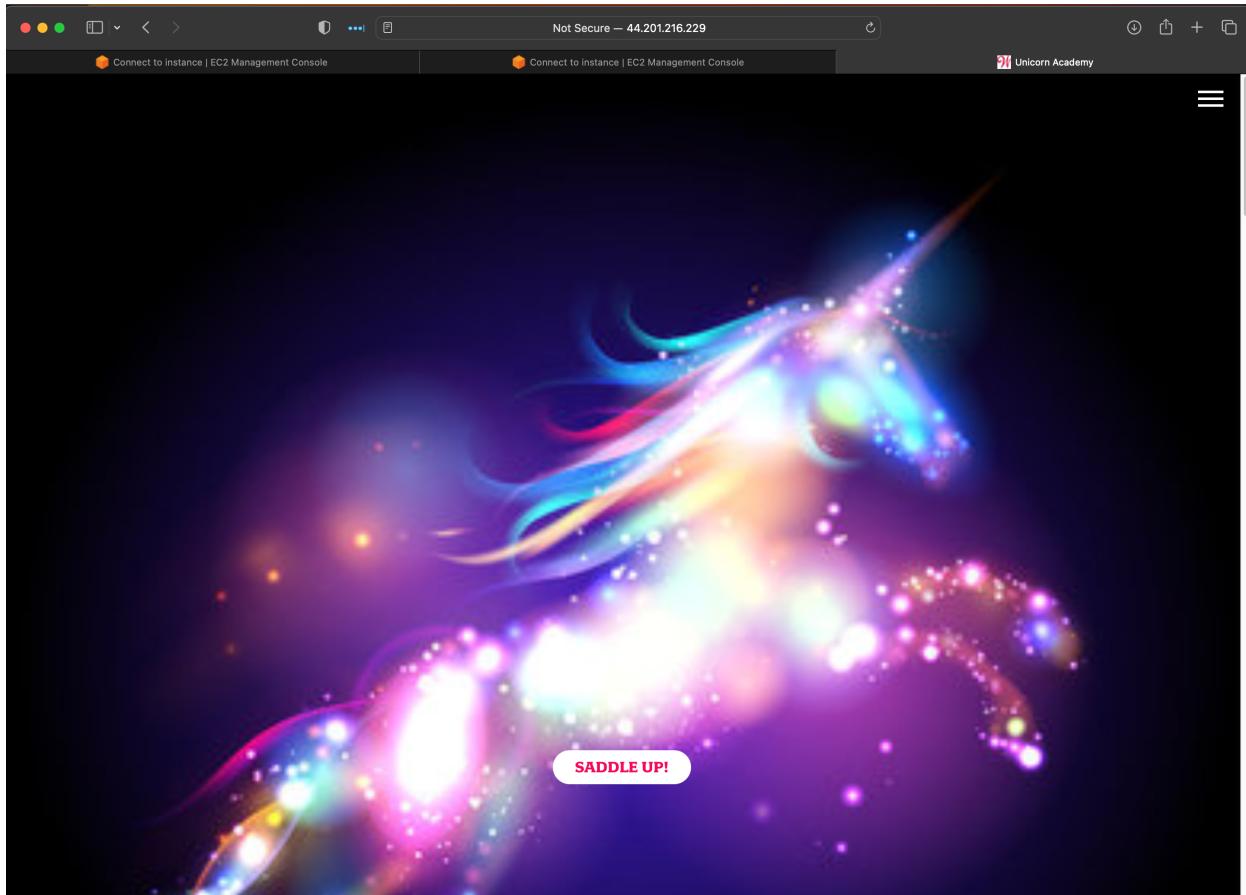
```
sudo yum install git -y
sudo yum install docker -y
sudo service docker start
sudo usermod -a -G docker ec2-user
```

5 of 9

- I built and ran my docker image.

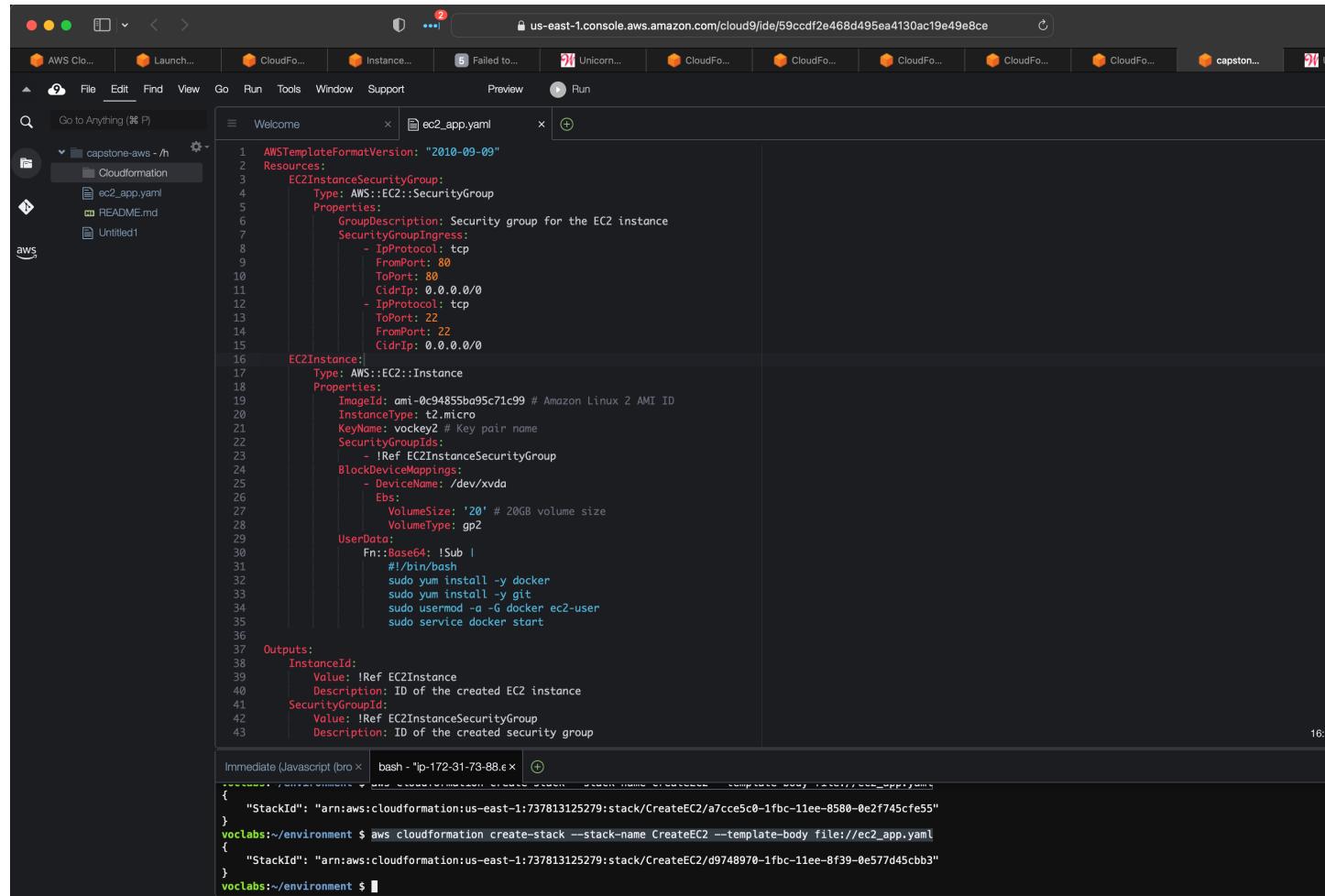
```
[ec2-user@ip-172-31-88-86 ~]$ cd capstone-aws/
[ec2-user@ip-172-31-88-86 capstone-aws]$ ls
Dockerfile apply.html faq.html fonts index.html js ride.html signin.html verify.html
README.md css favicon.ico images investors.html register.html robots.txt unicorns.html
[ec2-user@ip-172-31-88-86 capstone-aws]$ docker build -t capstone-aws .
Sending build context to Docker daemon 22.62MB
Step 1/4 : FROM nginx
latest: Pulling from library/nginx
faef57ae888: Pull complete
76579e9ed380: Pull complete
cf707e233955: Pull complete
91bb7937700d: Pull complete
4b962717ba55: Pull complete
f46d7b05649a: Pull complete
103501419a0a: Pull complete
Digest: sha256:08bc36ad52474e528cc1ea3426b5e3f4bad8a130318e3140d6cfe29c8892c7ef
Status: Downloaded newer image for nginx:latest
--> 021283c8e695
Step 2/4 : COPY ./usr/share/nginx/html
--> 67c4a9261316
Step 3/4 : EXPOSE 80
--> Running in a889c1a7587a
Removing intermediate container a889c1a7587a
--> 08eeffcd0a0
Step 4/4 : CMD ["nginx", "-g", "daemon off;"]
--> Running in 5cae21723548
Removing intermediate container 5cae21723548
--> 4f80a31aa7e7
Successfully built 4f80a31aa7e7
Successfully tagged capstone-aws:latest
[ec2-user@ip-172-31-88-86 capstone-aws]$ docker run -dp 80:80 capstone-aws
339cabbe2a4cfa769d813eb787d265b97495bd21c4498689b90efaf9067fbdd6
[ec2-user@ip-172-31-88-86 capstone-aws]$
```

- I copied the public IPV4 address of the EC2 instance and pasted it into my browser and my application came up.



5. Infrastructure as code using AWS Cloud Formation:

- I created a cloud9 environment and created a new cloud formation folder and a new yaml template file called ec2_app.yaml



The screenshot shows the AWS Cloud9 IDE interface. The left sidebar displays a file tree for a project named 'capstone-aws - /h' containing 'Cloudformation', 'ec2_app.yaml', 'README.md', and 'Untitled1'. The main editor window shows the content of 'ec2_app.yaml' (shown below), and the bottom terminal window shows the execution of the command to create a CloudFormation stack.

```

AWSTemplateFormatVersion: "2010-09-09"
Resources:
  EC2InstanceSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Security group for the EC2 instance
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          ToPort: 22
          FromPort: 22
          CidrIp: 0.0.0.0/0
  EC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: ami-0c94855ba95c71c99 # Amazon Linux 2 AMI ID
      InstanceType: t2.micro
      KeyName: vockey2 # Key pair name
      SecurityGroupIds:
        - !Ref EC2InstanceSecurityGroup
      BlockDeviceMappings:
        - DeviceName: /dev/xvda
          Ebs:
            VolumeSize: '20' # 20GB volume size
            VolumeType: gp2
      UserData:
        Fn::Base64: |Sub |
          #!/bin/bash
          sudo yum install -y docker
          sudo yum install -y git
          sudo usermod -a -G docker ec2-user
          sudo service docker start
Outputs:
  InstanceId:
    Value: !Ref EC2Instance
    Description: ID of the created EC2 instance
  SecurityGroupId:
    Value: !Ref EC2InstanceSecurityGroup
    Description: ID of the created security group
}
  "StackId": "arn:aws:cloudformation:us-east-1:737813125279:stack/CreateEC2/a7cce5c0-1fbc-11ee-8580-0e27745cfef55"
}
voclabs:~/environment $ aws cloudformation create-stack --stack-name CreateEC2 --template-body file://ec2_app.yaml
{
  "StackId": "arn:aws:cloudformation:us-east-1:737813125279:stack/CreateEC2/d9748970-1fbc-11ee-8f39-0e577d45cbb3"
}
voclabs:~/environment $ 

```

- I created the stack using the following command,

```
aws cloudformation create-stack --stack-name CreateEC2 --template-body file://ec2_app.yaml
```

- My created stack is called CreateEC2
- I went to the EC2 instances page and saw that the new EC2 instance had been created.

The screenshot shows the AWS CloudFormation console interface. On the left, the navigation pane includes 'CloudFormation', 'Services', and a search bar. Under 'CloudFormation', the 'Stacks' section is selected, showing a list of stacks. One stack, 'CreateEC2', is highlighted and expanded, revealing its creation details: it was created on 2023-07-11 at 00:30:45 UTC-0700, status is 'CREATE_COMPLETE', and its ARN is arnaws:cloudformation:us-east-1:737813125279:stack/CreateEC2/d9748970-1fbc-11ee-8f39-0e577d45cbb3. The right side of the screen displays the detailed view for the 'CreateEC2' stack, with tabs for 'Stack info', 'Events', 'Resources', and 'Outputs'. The 'Stack info' tab is active, showing the overview and stack ID.

- This is my EC2 instance that was created.

The screenshot shows the AWS EC2 Instance Details page for an instance named i-0359abaaebf125a1c. The page is titled "Instance summary for i-0359abaaebf125a1c Info". The instance is currently running. Key details include:

- Instance ID: i-0359abaaebf125a1c
- Public IPv4 address: 18.215.146.196
- Private IP DNS name (IPv4 only): ip-172-31-25-118.ec2.internal
- Instance type: t2.micro
- VPC ID: vpc-0c5cf0e44a50a41d6
- Subnet ID: subnet-036d9c84f1bedb7b3
- Platform: Amazon Linux (Inferred)
- AMI ID: ami-0c94855ba95c71c99
- AMI name: amzn2-ami-hvm-2.0.20200904.0-x86_64-gp2
- Launch time: Tue Jul 11 2023 00:30:57 GMT-0700 (Pacific Daylight Time) (7 minutes)
- Monitoring: disabled
- Termination protection: Disabled
- AMI location: amazon/amzn2-ami-hvm-2.0.20200904.0-x86_64-gp2

- I connected to the instance using ssh.
- I built and ran my docker image.

9 of 9

```
janani@Priyas-Air Desktop % ssh -i "vockey2-5.pem" ec2-user@ec2-18-215-146-196.compute-1.amazonaws.com
Last login: Tue Jul 11 07:32:45 2023 from c-98-33-83-15.hsd1.ca.comcast.net

      _\ _ ) _/   Amazon Linux 2 AMI
     _\_\_|_||

https://aws.amazon.com/amazon-linux-2/
54 package(s) needed for security, out of 113 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-25-118 ~]$ cd capstone-aws/
[ec2-user@ip-172-31-25-118 capstone-aws]$ docker build -t test_app .
Sending build context to Docker daemon 22.62MB
Step 1/4 : FROM nginx
latest: Pulling from library/nginx
faef57eae888: Pull complete
76579e9ed380: Pull complete
cf707e233955: Pull complete
91bb7937700d: Pull complete
4b962717ba55: Pull complete
f46d7b05649a: Pull complete
103501419a0a: Pull complete
Digest: sha256:08bc36ad52474e528cc1ea3426b5e3f4bad8a130318e3140d6cfe29c8892c7ef
Status: Downloaded newer image for nginx:latest
--> 021283c8eb95
Step 2/4 : COPY . /usr/share/nginx/html
--> 804adb4259f2
Step 3/4 : EXPOSE 80
--> Running in 8f54446ec24e
Removing intermediate container 8f54446ec24e
--> 6e691987a497
Step 4/4 : CMD ["nginx", "-g", "daemon off;"]
--> Running in bc0ef7125239
Removing intermediate container bc0ef7125239
--> 92fc5f88f5a7
Successfully built 92fc5f88f5a7
Successfully tagged test_app:latest
[ec2-user@ip-172-31-25-118 capstone-aws]$ docker run -dp 80:80 test_app
f7ba88103793667d9c0a9ad6b05faa95bcd1ea934d8aa71e819b875ef2f12507
[ec2-user@ip-172-31-25-118 capstone-aws]$ █
```

- I launched my application using the public IPv4 address of the EC2 instance and it worked.
- I committed and pushed my ec2_app.yaml file to my repository.

