“Expert Cloud Consulting”

SOP | Canary Deployment using Docker and NGINX

22 july 2025

—

Contributed by: Akshata
Approved by: Akshay (In Review)
Expert Cloud Consulting
Office #811, Gera Imperium Rise,
Hinjewadi Phase-II Rd, Pune, India – 411057

Canary Deployment using Docker and NGINX

Expert Cloud Consulting
Enhance Optimise & Scale
ASCP GPUonCLOUD Pvt Ltd

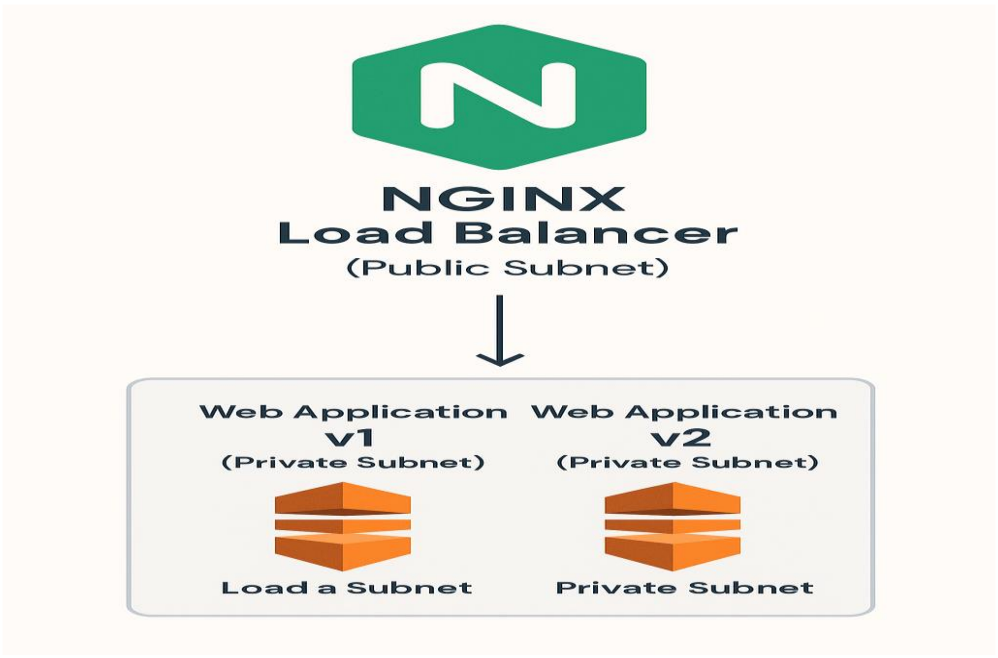
Advanced IaC and Deployment Strategies

Topics :

- Advanced Terraform: modules, workspaces.
- Deployment strategies: Blue-Green, Canary, Rolling updates.

Assignments:

1. **Build a Terraform project with:**
   - ➢ Modules for reusable VPC and EC2 resources.
   - ➢ Separate workspaces for development and production environments.
2. **Simulate a Canary deployment:**
   - ➢ Create two Dockerized versions of a web app.
   - ➢ Gradually route traffic to the new version using an Nginx load balancer.



Objective

To simulate a real-world Canary deployment strategy by:

- Creating two Dockerized versions of a web application:
  - v1: Stable and currently live version
  - v2: New version with potential changes or improvements
- Using NGINX as a reverse proxy and load balancer to:
  - Gradually shift traffic from the stable version (v1) to the new version (v2)
  - Minimize risk during deployment by allowing controlled exposure of new features
  - Enable early detection of issues with the new version before full rollout

This setup mimics production-grade deployments where continuous delivery and high availability are crucial, allowing for safe testing of new releases in live environments with real user traffic.

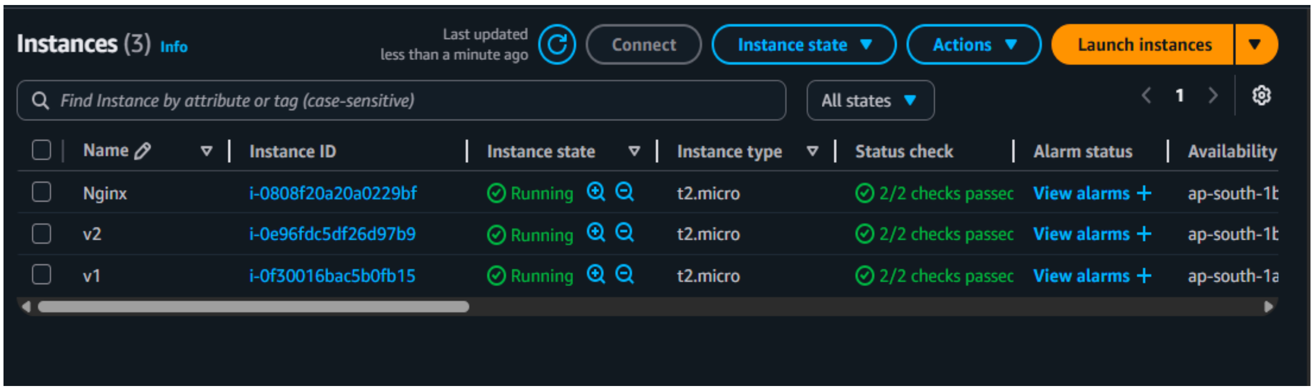
Document References

The following resources were referred to during the creation and execution of this Terraform-based infrastructure setup

Date	Document	Filename / Url
17 July	AWS Canary Deployment	https://youtu.be/TzShhyhSpHc?si=U96lHlWVfraMuMWa
18 July	Blue-Green/Canary deployment with NGINX	https://techannotation.wordpress.com/2019/12/13/blue-green-canary-deployment-with-nginx/

To simulate a Canary deployment, I created the following AWS infrastructure using EC2 instances

Total EC2 Instances: 3


Instances (3) Info
Last updated less than a minute ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
Nginx	i-0808f20a20a0229bf	Running	t2.micro	2/2 checks passec	View alarms	ap-south-1b
v2	i-0e96fdc5df26d97b9	Running	t2.micro	2/2 checks passec	View alarms	ap-south-1b
v1	i-0f30016bac5b0fb15	Running	t2.micro	2/2 checks passec	View alarms	ap-south-1a

1 Public EC2 Instance (Nginx Load Balancer)

Acts as the public-facing load balancer.

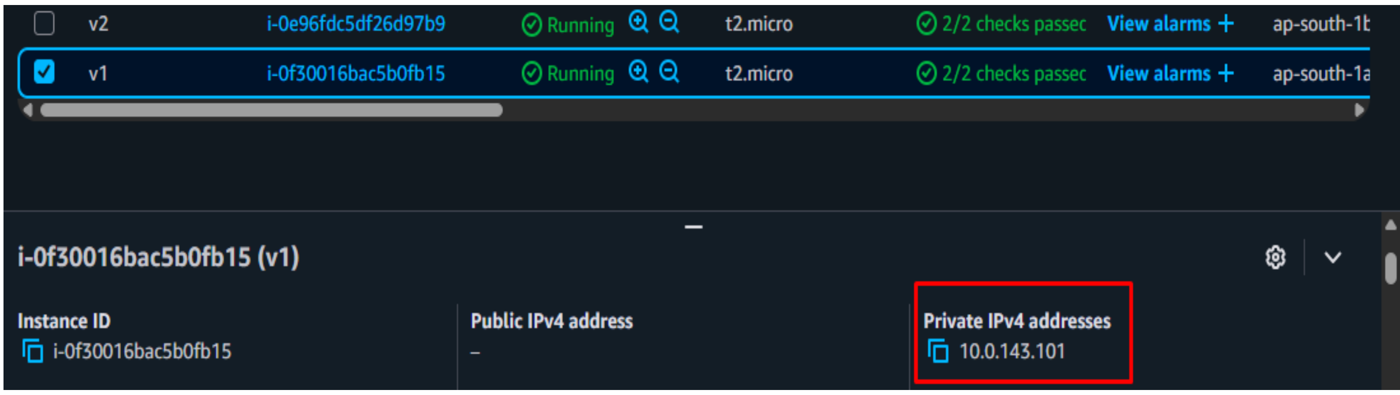
Installed with Nginx to route traffic to backend instances.

Assigned a public IP.

2 Private EC2 Instances (Application Servers):

Web App v1 Instance

Runs the version 1 of the Dockerized web application.
Accessible only within the VPC via private IP (e.g., 10.0.143.101)



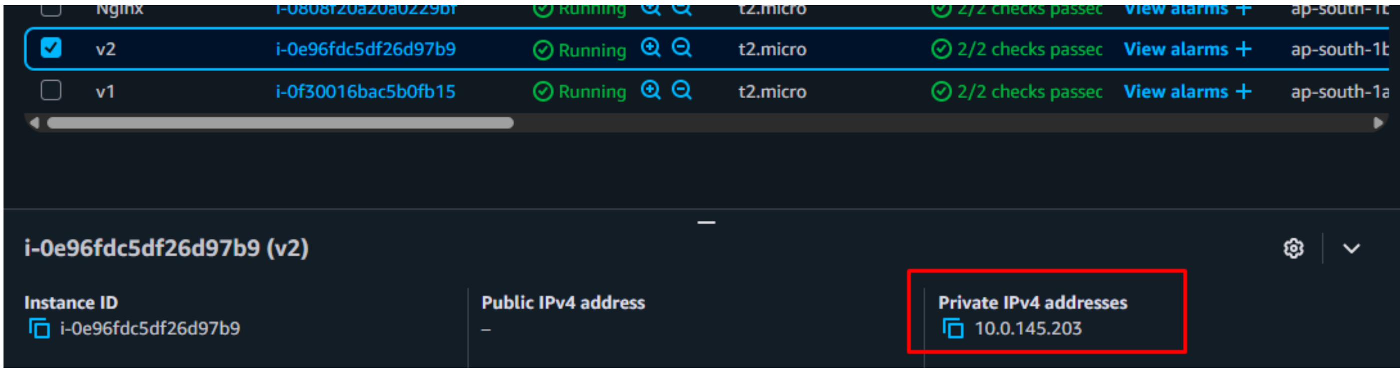
v2	i-0e96fdc5df26d97b9	Running	t2.micro	2/2 checks passec	View alarms	ap-south-1b
✓ v1	i-0f30016bac5b0fb15	Running	t2.micro	2/2 checks passec	View alarms	ap-south-1a

i-0f30016bac5b0fb15 (v1)

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0f30016bac5b0fb15	–	10.0.143.101

Runs the version 2 of the Dockerized web application.
Also private with IP (e.g., 10.0.145.203)

Web App v2 Instance



	Name	Instance ID	State	Type	Status check	Alarms	AZ
☐	Nginx	i-0808f20a20a0229bf	Running	t2.micro	2/2 checks passec	View alarms +	ap-south-1b
✓	v2	i-0e96fdc5df26d97b9	Running	t2.micro	2/2 checks passec	View alarms +	ap-south-1b
☐	v1	i-0f30016bac5b0fb15	Running	t2.micro	2/2 checks passec	View alarms +	ap-south-1a

i-0e96fdc5df26d97b9 (v2)

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0e96fdc5df26d97b9	–	10.0.145.203

VPC Configuration:

A custom VPC was created with:

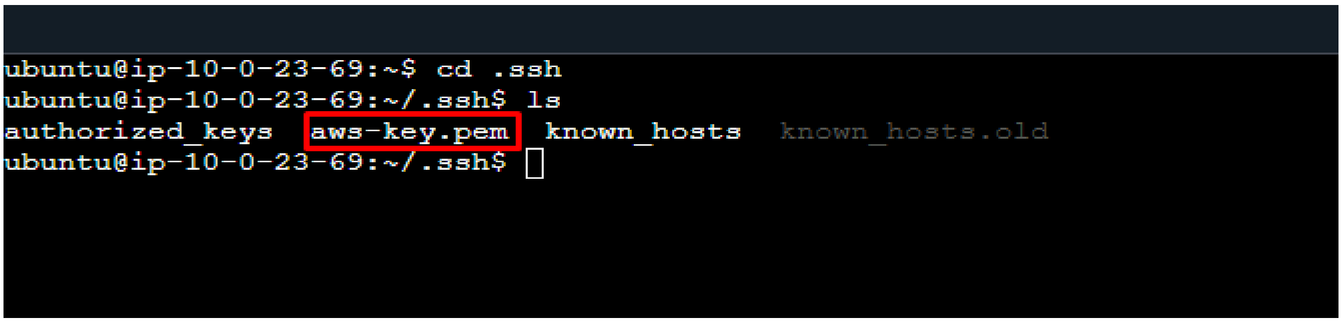
2 private subnets (for v1 and v2 app servers)

1 public subnet (for Nginx load balancer)

Route tables and NAT gateway properly configured to allow internet access where needed.

Connecting to Private EC2s (v1 & v2) from NGINX

To securely configure and deploy web applications on private EC2 instances (v1 and v2), SSH access is achieved through the public EC2 instance using its .pem key

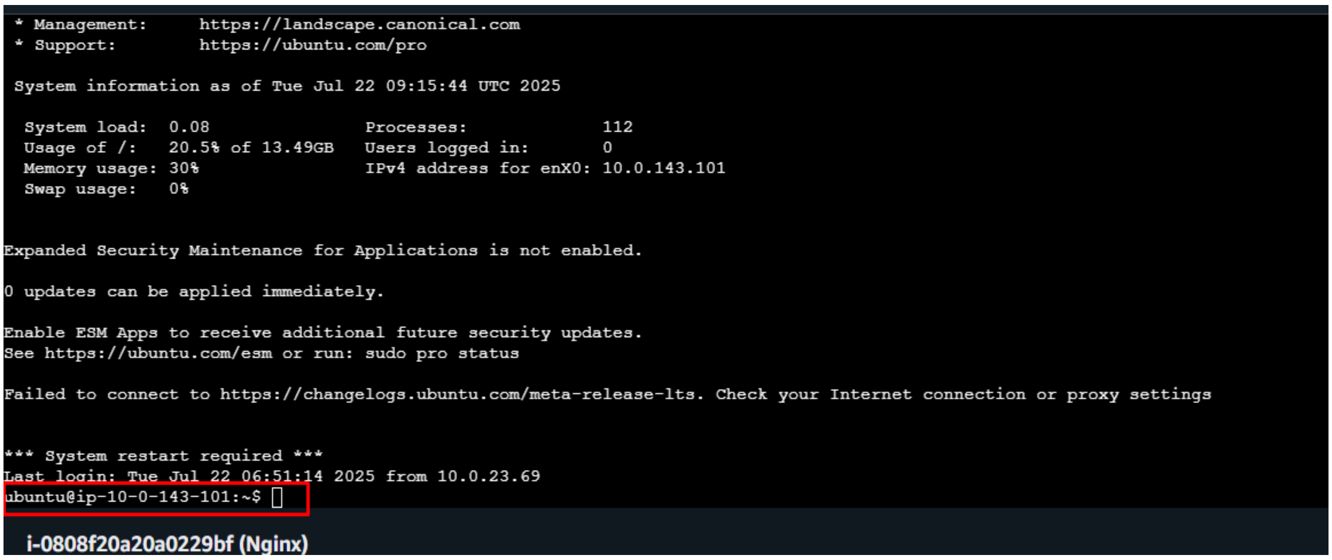


```
ubuntu@ip-10-0-23-69:~$ cd .ssh
ubuntu@ip-10-0-23-69:~/.ssh$ ls
authorized_keys  aws-key.pem  known_hosts  known_hosts.old
ubuntu@ip-10-0-23-69:~/.ssh$ 
```

chmod 400aws-key.pem

ssh -i "your-key.pem" ubuntu@<v1-private-ip>

Once connected to the v1 EC2 instance



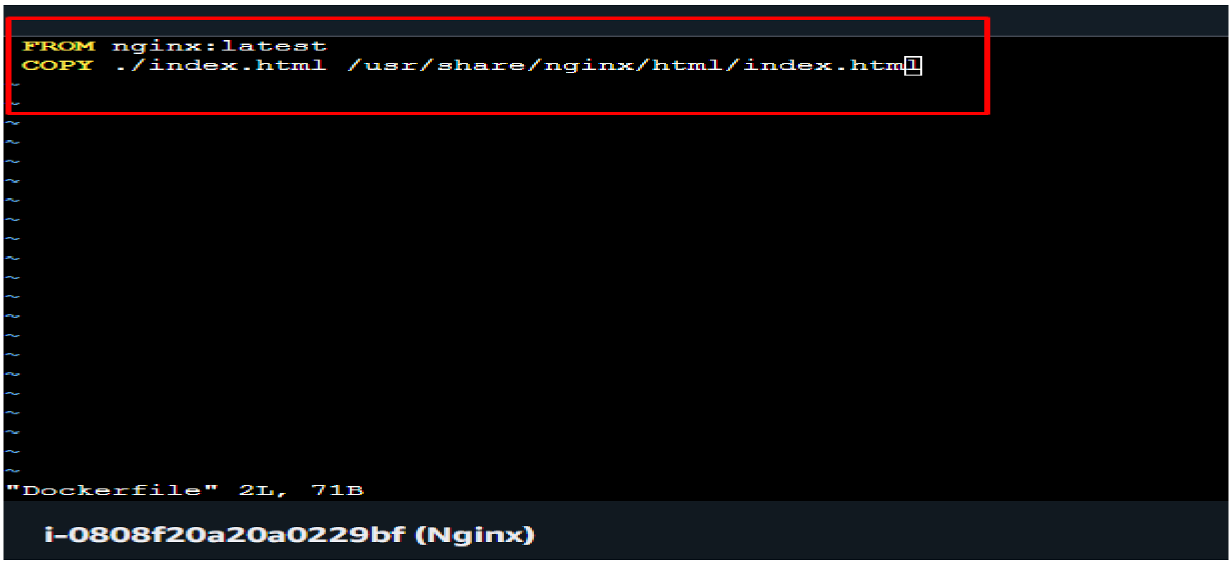
```
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Tue Jul 22 09:15:44 UTC 2025

  System load:  0.08               Processes:             112
  Usage of /:   20.5% of 13.49GB   Users logged in:       0
  Memory usage: 30%                IPv4 address for enX0: 10.0.143.101
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings


*** System restart required ***
Last login: Tue Jul 22 06:51:14 2025 from 10.0.23.69
ubuntu@ip-10-0-143-101:~$
```

i-0808f20a20a0229bf (Nginx)

Update the system and install Docker:

```
sudo apt update
sudo apt install docker.io -y
sudo systemctl start docker
sudo systemctl enable docker
```

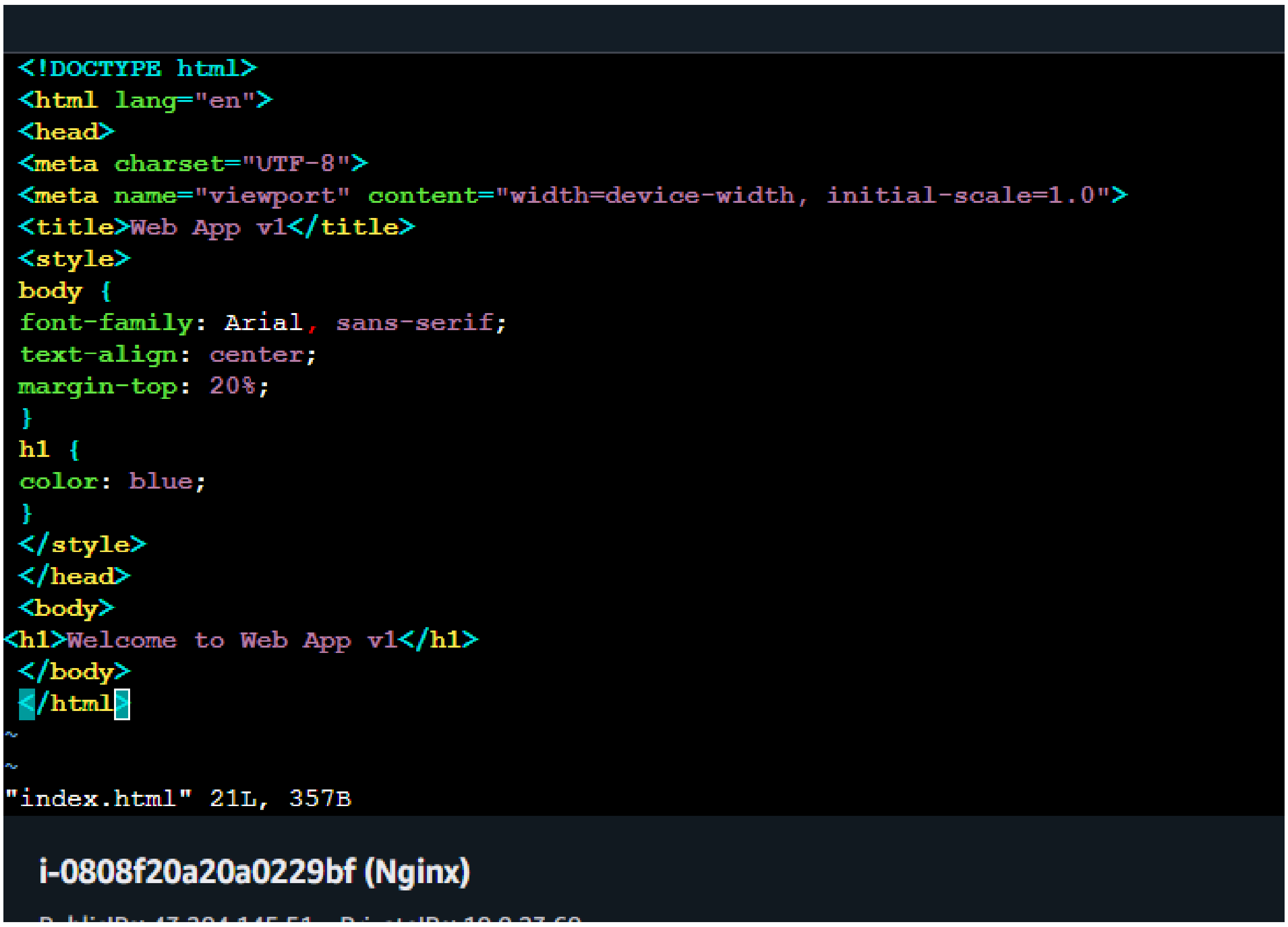
Create a Dockerfile:



```
FROM nginx:latest
COPY ./index.html /usr/share/nginx/html/index.html
"Dockerfile" 2L, 71B
```

i-0808f20a20a0229bf (Nginx)

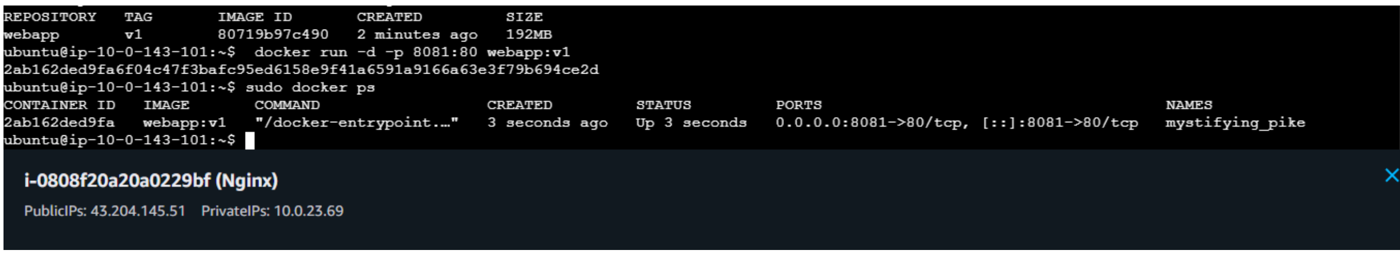
Create a simple index.html for v1:



```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Web App v1</title>
<style>
body {
font-family: Arial, sans-serif;
text-align: center;
margin-top: 20%;
}
h1 {
color: blue;
}
</style>
</head>
<body>
<h1>Welcome to Web App v1</h1>
</body>
</html>
~
~
"index.html" 21L, 357B
```

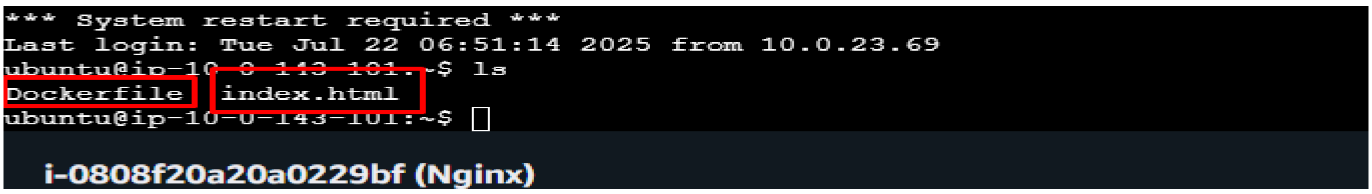
i-0808f20a20a0229bf (Nginx)

Build and run the Docker container:
docker build -t webapp-v1 .
docker run -d -p 8081:80 --name webapp-v1-container webapp-v1
sudo docker ps



```
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
webapp       v1        80719b97c490   2 minutes ago   192MB
ubuntu@ip-10-0-143-101:~$  docker run -d -p 8081:80 webapp:v1
2ab162ded9fa6f04c47f3bafc95ed6158e9f41a6591a9166a63e3f79b694ce2d
ubuntu@ip-10-0-143-101:~$ sudo docker ps
CONTAINER ID   IMAGE       COMMAND                  CREATED         STATUS         PORTS                                   NAMES
2ab162ded9fa   webapp:v1   "/docker-entrypoint.…"   3 seconds ago   Up 3 seconds   0.0.0.0:8081->80/tcp, [::]:8081->80/tcp   mystifying_pike
ubuntu@ip-10-0-143-101:~$
```

i-0808f20a20a0229bf (Nginx)
PublicIPs: 43.204.145.51   PrivateIPs: 10.0.23.69



```
*** System restart required ***
Last login: Tue Jul 22 06:51:14 2025 from 10.0.23.69
ubuntu@ip-10-0-143-101:~$ ls
Dockerfile  index.html
ubuntu@ip-10-0-143-101:~$
```

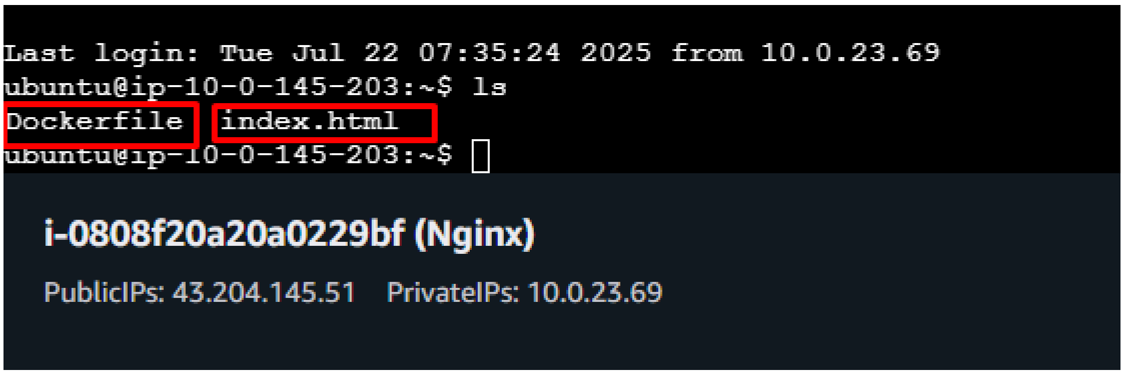
i-0808f20a20a0229bf (Nginx)

Expert Cloud Consulting
Enhance Optimise & Scale

connected to the v2 EC2 instance

chmod 400aws-key.pem

ssh -i "your-key.pem" ubuntu@<v2-private-ip>



```
Last login: Tue Jul 22 07:35:24 2025 from 10.0.23.69
ubuntu@ip-10-0-145-203:~$ ls
Dockerfile  index.html
ubuntu@ip-10-0-145-203:~$ 
```

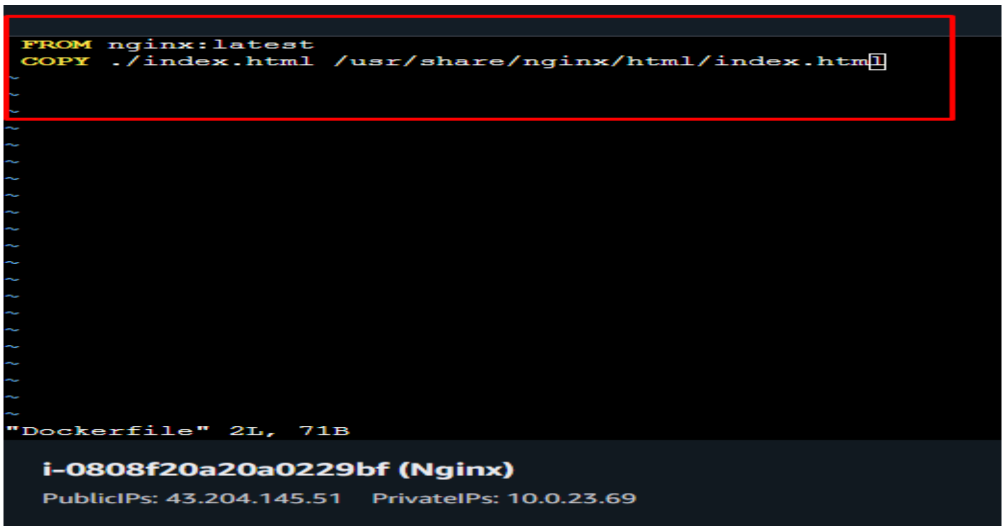
i-0808f20a20a0229bf (Nginx)

PublicIPs: 43.204.145.51   PrivateIPs: 10.0.23.69

Update the system and install Docker:

```
sudo apt update
sudo apt install docker.io -y
sudo systemctl start docker
sudo systemctl enable docker
```

Create a Dockerfile:

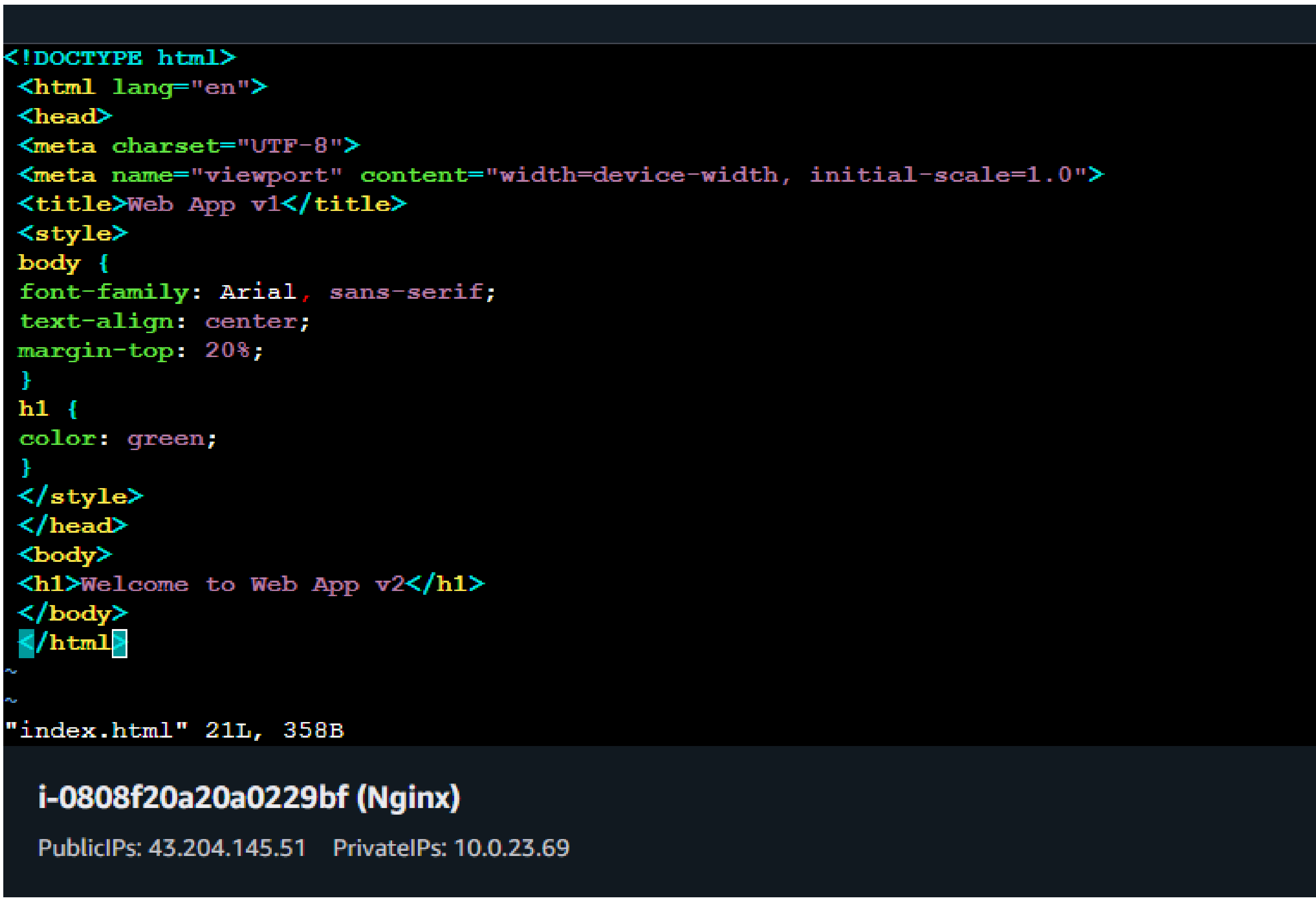


```
FROM nginx:latest
COPY ./index.html /usr/share/nginx/html/index.html
"Dockerfile" 2L, 71B
```

i-0808f20a20a0229bf (Nginx)

PublicIPs: 43.204.145.51   PrivateIPs: 10.0.23.69

Create a simple index.html for v2:



```
<!DOCTYPE html>
 <html lang="en">
 <head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Web App v1</title>
 <style>
 body {
 font-family: Arial, sans-serif;
 text-align: center;
 margin-top: 20%;
 }
 h1 {
 color: green;
 }
 </style>
 </head>
 <body>
 <h1>Welcome to Web App v2</h1>
 </body>
 </html>
~
~
"index.html" 21L, 358B
```

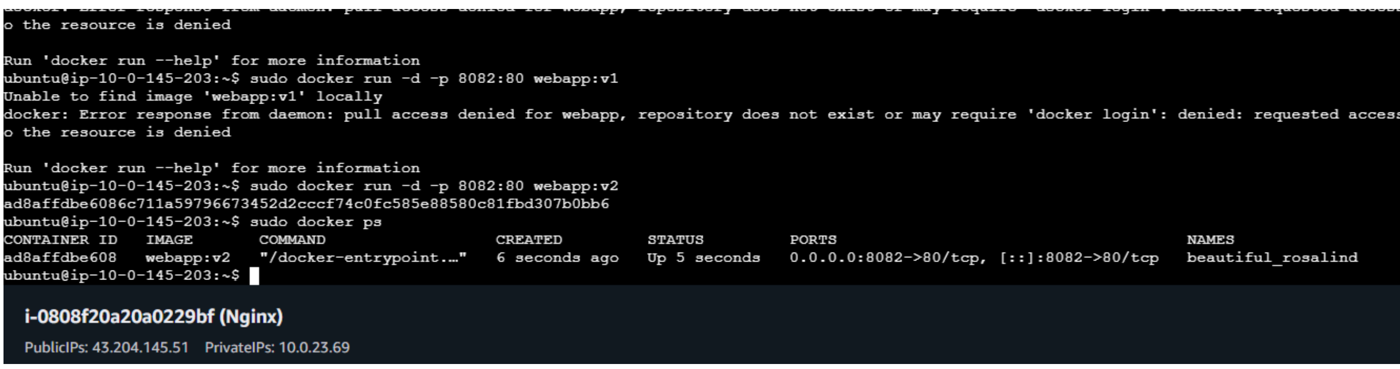
i-0808f20a20a0229bf (Nginx)

PublicIPs: 43.204.145.51   PrivateIPs: 10.0.23.69

Build & Run Docker container

docker build -t webapp-v2 .
docker run -d -p 8081:80 webapp-v2

sudo docker ps



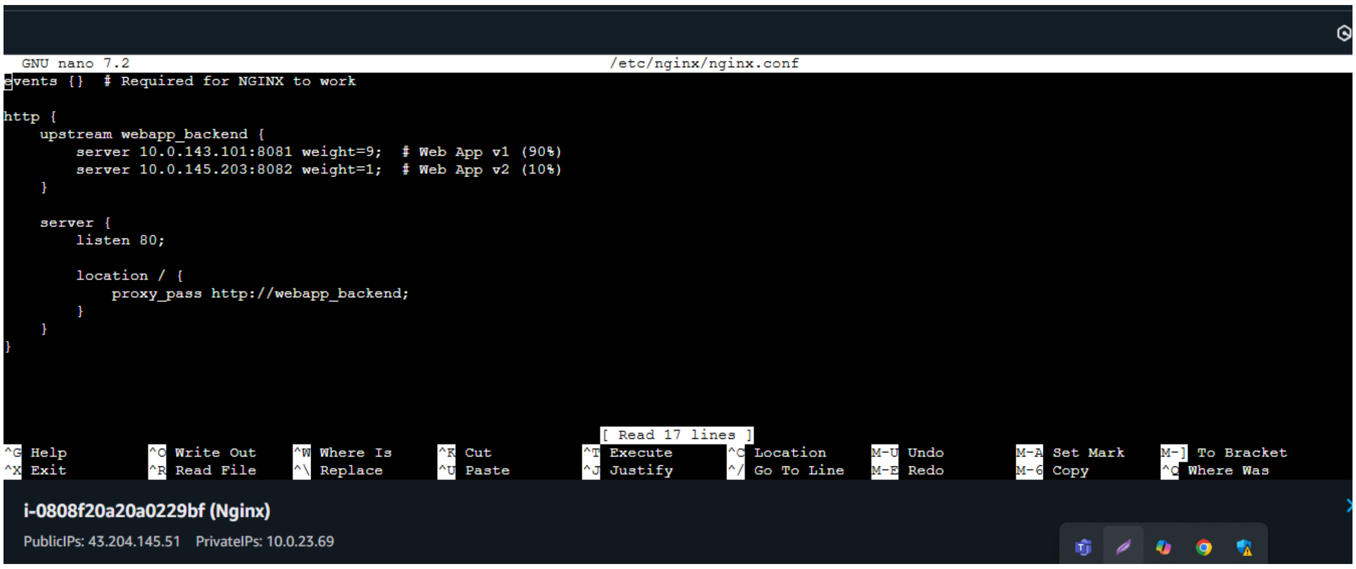
```
o the resource is denied

Run 'docker run --help' for more information
ubuntu@ip-10-0-145-203:~$ sudo docker run -d -p 8082:80 webapp:v1
Unable to find image 'webapp:v1' locally
docker: Error response from daemon: pull access denied for webapp, repository does not exist or may require 'docker login': denied: requested access
o the resource is denied

Run 'docker run --help' for more information
ubuntu@ip-10-0-145-203:~$ sudo docker run -d -p 8082:80 webapp:v2
ad8affdbe6086c711a59796673452d2cccf74c0fc585e88580c81fbd307b0bb6
ubuntu@ip-10-0-145-203:~$ sudo docker ps
CONTAINER ID   IMAGE       COMMAND                  CREATED         STATUS         PORTS                                   NAMES
ad8affdbe608   webapp:v2   "/docker-entrypoint.…"   6 seconds ago   Up 5 seconds   0.0.0.0:8082->80/tcp, [::]:8082->80/tcp   beautiful_rosalind
ubuntu@ip-10-0-145-203:~$
```

i-0808f20a20a0229bf (Nginx)

PublicIPs: 43.204.145.51   PrivateIPs: 10.0.23.69

Nginx Configuration on Public EC2

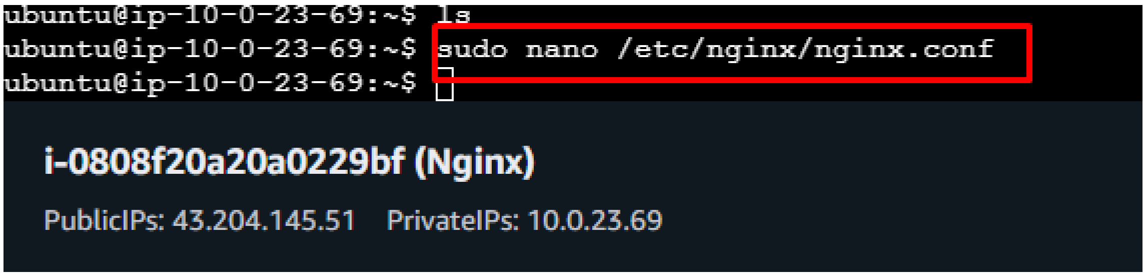
/etc/nginx/nginx.conf:



```
  GNU nano 7.2                                 /etc/nginx/nginx.conf
events {}  # Required for NGINX to work

http {
    upstream webapp_backend {
        server 10.0.143.101:8081 weight=9;  # Web App v1 (90%)
        server 10.0.145.203:8082 weight=1;  # Web App v2 (10%)
    }

    server {
        listen 80;

        location / {
            proxy_pass http://webapp_backend;
        }
    }
}

                                            [ Read 17 lines ]
^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark   M-] To Bracket
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy       ^Q Where Was
```

i-0808f20a20a0229bf (Nginx)
PublicIPs: 43.204.145.51   PrivateIPs: 10.0.23.69

ls



```
ubuntu@ip-10-0-23-69:~$ ls
ubuntu@ip-10-0-23-69:~$ sudo nano /etc/nginx/nginx.conf
ubuntu@ip-10-0-23-69:~$
```

i-0808f20a20a0229bf (Nginx)
PublicIPs: 43.204.145.51   PrivateIPs: 10.0.23.69

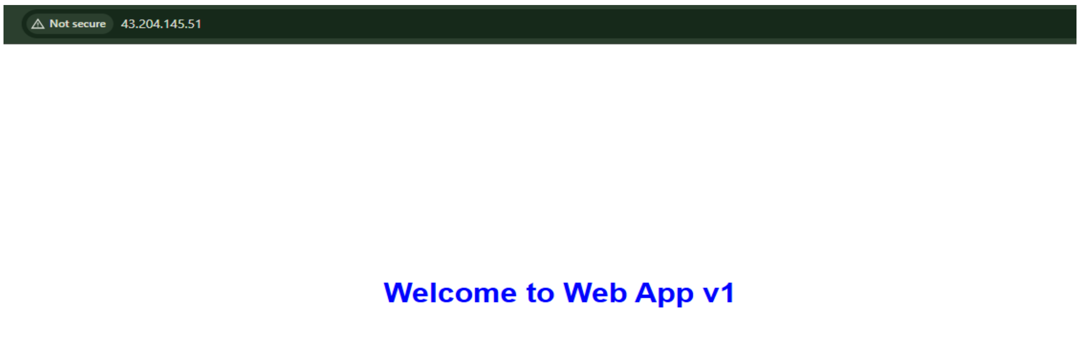
Restart nginx:

sudo nginx -t
sudo systemctl restart nginx

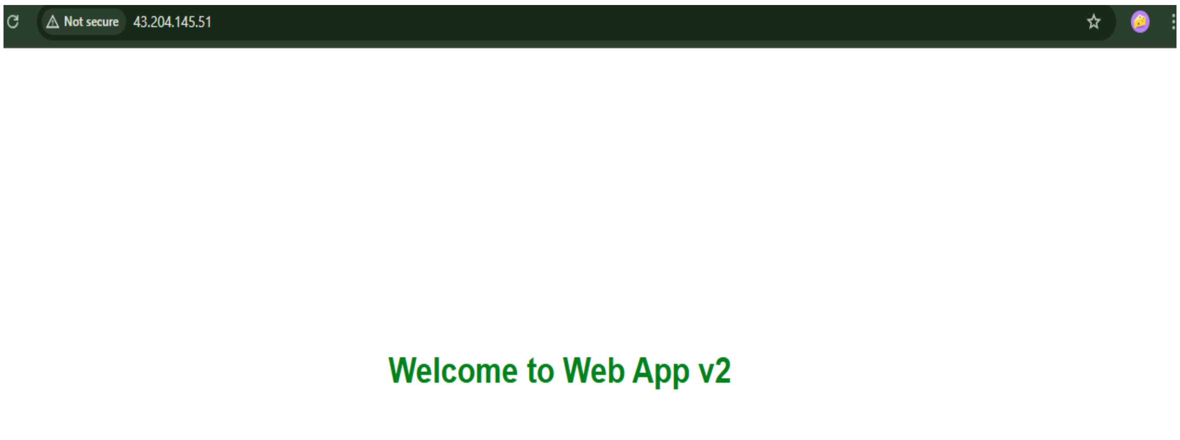
Open Browser and Paste Public IP

Now, go to your browser and open:

http://43.204.145.51


Not secure 43.204.145.51

Welcome to Web App v1


Not secure 43.204.145.51

Welcome to Web App v2