

# Deploying a Web Application on AWS EC2 Using Apache Web Server

---

## Project Objective

The objective of this project is to gain hands-on experience with Amazon Web Services (AWS) by launching an EC2 instance, configuring secure access, installing and managing the Apache2 web server, and deploying a simple web application that can be accessed through a web browser.

By completing this project, students will develop foundational skills in: - Cloud infrastructure setup using AWS EC2 - Basic Linux server administration - Networking concepts such as ports and security groups - Web server installation and deployment

---

## Prerequisites

- Basic knowledge of Linux commands
  - An active AWS account
  - Basic understanding of networking concepts (ports, security groups)
  - SSH client:
    - Terminal (Linux/macOS)
    - PuTTY (Windows)
- 

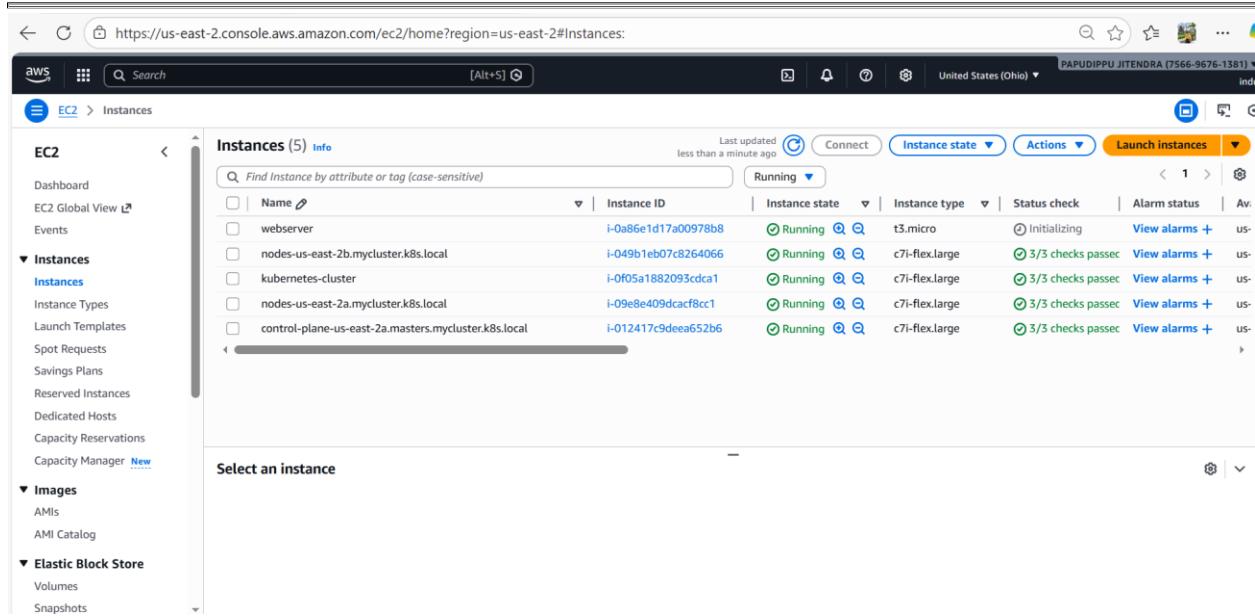
## Task 1: Create an EC2 Instance

### Steps Performed

1. Logged in to the AWS Management Console.
2. Navigated to **EC2 → Launch Instance**.
3. Selected an Amazon Machine Image (AMI):
  - Ubuntu Server 20.04 LTS or 22.04 LTS
4. Chose the instance type:
  - **t2.micro** (Free Tier eligible)
5. Created or selected an existing key pair for SSH access.
6. Launched the instance and verified that its state changed to **Running**.

## Deliverable

- Screenshot showing the EC2 instance in **Running** state.



The screenshot shows the AWS EC2 Instances page with the URL <https://us-east-2.console.aws.amazon.com/ec2/home?region=us-east-2#Instances>. The page displays five EC2 instances in the 'Running' state. The instances are:

Name	Instance ID	Instance State	Instance Type	Status Check	Alarm Status
webserver	i-0a86e1d17a00978b8	Running	t3.micro	Initializing	View alarms +
nodes-us-east-2b.mycluster.k8s.local	i-049b1eb07c8264066	Running	c7i-flex.large	3/3 checks passed	View alarms +
kubernetes-cluster	i-0f05a1882093cdca1	Running	c7i-flex.large	3/3 checks passed	View alarms +
nodes-us-east-2a.mycluster.k8s.local	i-09e8e409dcacf8cc1	Running	c7i-flex.large	3/3 checks passed	View alarms +
control-plane-us-east-2a.masters.mycluster.k8s.local	i-012417c9dea652b6	Running	c7i-flex.large	3/3 checks passed	View alarms +

## Task 2: Configure Security Groups (Firewall Rules)

### Steps Performed

1. Opened the EC2 instance details page.
2. Edited the inbound rules of the associated security group.
3. Added the following inbound rules:

Protocol	Port	Source
SSH	22	My IP
HTTP	80	0.0.0.0/0

4. Saved the security group configuration.

## Deliverable

- Screenshot of the security group inbound rules configuration.

**Edit inbound rules** Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

**Inbound rules** Info

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0623f2a15ff7c9dd4	SSH	TCP	22	Custom	0.0.0.0/0 <small>X</small>
=	Custom TCP	TCP	80	Anywh...	0.0.0.0/0 <small>X</small>

**Add rule**

⚠ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel Preview changes Save rules

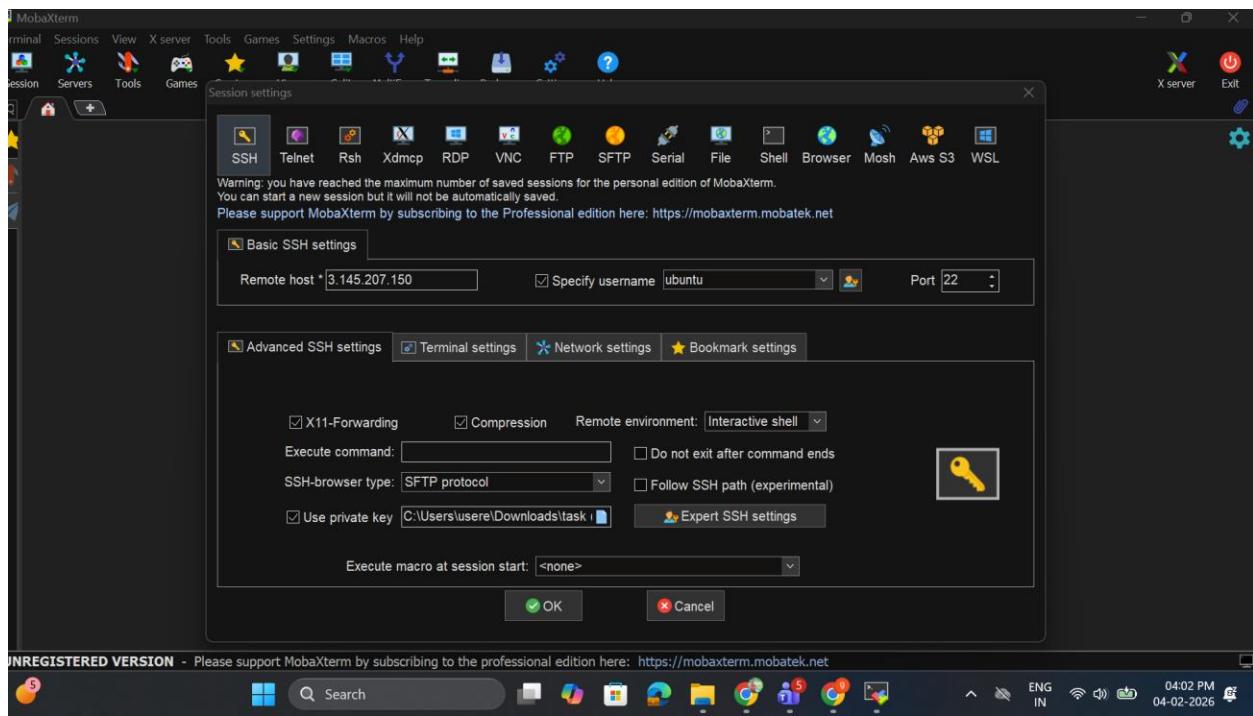
## Task 3: Connect to EC2 Using SSH

### Steps Performed

1. Copied the public IPv4 address of the EC2 instance.
2. Opened a terminal (or PuTTY on Windows).
3. Connected to the EC2 instance using the SSH command:  
`ssh -i your-key.pem ubuntu@<public-ip>`
4. Verified successful login to the Ubuntu server.

### Deliverable

- Screenshot showing successful SSH login to the EC2 instance.



## Task 4: Install and Start Apache2 Web Server

### Steps Performed

1. Updated the package list:

```
sudo apt update
```

2. Installed Apache2 web server:

```
sudo apt install apache2 -y
```

3. Started and enabled Apache to run on boot:

```
sudo systemctl start apache2  
sudo systemctl enable apache2
```

4. Verified Apache service status:

```
sudo systemctl status apache2
```

### Deliverable

- Screenshot showing Apache2 service running successfully.

```
[created symlink /etc/systemd/system/multi-user.target.wants/apache-htcacheclean.service → /usr/lib/systemd/system/apache-htcacheclean.service]
Processing triggers for ufw (0.36.2-6) ...
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.6) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.

ubuntu@ip-172-31-16-179:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
    Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset: enabled)
    Active: active (running) since Wed 2026-02-04 10:30:27 UTC; 15s ago
      Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 2175 (apache2)
     Tasks: 55 (limit: 1008)
    Memory: 5.3M (peak: 5.8M)
       CPU: 41ms
      CGroup: /system.slice/apache2.service
              ├─2175 /usr/sbin/apache2 -k start
              ├─2178 /usr/sbin/apache2 -k start
              └─2179 /usr/sbin/apache2 -k start

Feb 04 10:30:27 ip-172-31-16-179 systemd[1]: Starting apache2.service - The Apache HTTP Server...
Feb 04 10:30:27 ip-172-31-16-179 systemd[1]: Started apache2.service - The Apache HTTP Server.
ubuntu@ip-172-31-16-179:~$ █
```

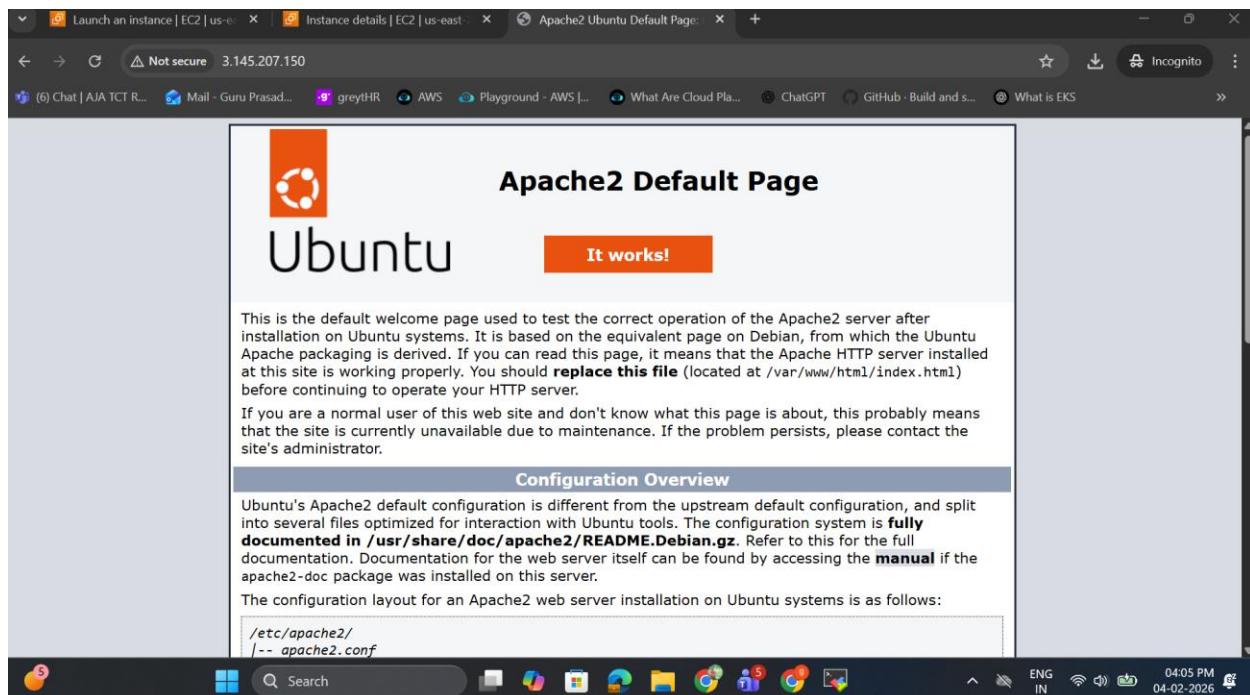
## Task 5: Test HTTP Access

## Steps Performed

1. Opened a web browser.
  2. Entered the EC2 public IPv4 address in the address bar.
  3. Confirmed that the Apache2 default welcome page was displayed, indicating successful HTTP access.

## Deliverable

- Screenshot of the Apache2 default web page displayed in the browser.



## Task 6: Deploy a Sample Web Application

### Steps Performed

1. Navigated to the Apache web root directory:  
`cd /var/www/html`

2. Created and edited the `index.html` file:  
`sudo nano index.html`

3. Added the following HTML content:

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Hanshik Restaurant | Food Menu</title>

<style>
```

```
body {  
    font-family: Arial, sans-serif;  
  
    background: #fff3e0;  
  
    margin: 0;  
}
```

```
header {  
background: linear-gradient(135deg, #ff5722, #ff9800);  
  
color: white;  
  
padding: 25px;  
  
text-align: center;  
}
```

```
header h1 {  
    margin: 0;  
    font-size: 2em;  
}
```

```
header p {  
    margin: 5px 0 0;  
    font-size: 16px;
```

```
}

.container {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(220px, 1fr));
    gap: 20px;
    padding: 20px;
}

.food-card {
    background: white;
    border-radius: 12px;
    box-shadow: 0 5px 15px rgba(0,0,0,0.15);
    padding: 15px;
    text-align: center;
}

.food-card h3 {
    margin: 10px 0;
    color: #ff5722;
}
```

```
.price {  
    font-weight: bold;  
    color: #2e7d32;  
    margin-bottom: 10px;  
}  
  
button {  
    background: #4caf50;  
    border: none;  
    color: white;  
    padding: 10px 15px;  
    border-radius: 6px;  
    cursor: pointer;  
}  
  
button:hover {  
    background: #388e3c;  
}  
</style>  
</head>
```

```
<body>

<header>

  <h1>🍴 Hanshik Restaurant</h1>

  <p>Delicious Food, Fresh Taste</p>

</header>

<div class="container">

  <div class="food-card">

    <h3>🍕 Pizza</h3>

    <p class="price">$10</p>

    <button>Add to Cart</button>

  </div>

  <div class="food-card">

    <h3>🍔 Burger</h3>

    <p class="price">$7</p>

    <button>Add to Cart</button>

  </div>

</div>
```

```
<div class="food-card">  
  
<h3>  Tacos</h3>  
  
<p class="price">$6</p>  
  
<button>Add to Cart</button>  
  
</div>
```

```
<div class="food-card">  
  
<h3>  Pasta</h3>  
  
<p class="price">$8</p>  
  
<button>Add to Cart</button>  
  
</div>
```

```
<div class="food-card">  
  
<h3>  Fried Chicken</h3>  
  
<p class="price">$9</p>  
  
<button>Add to Cart</button>  
  
</div>
```

```
<div class="food-card">  
  
<h3>  Salad</h3>  
  
<p class="price">$5</p>
```

```
<button>Add to Cart</button>

</div>
```

```
<div class="food-card">

  <h3> 🍩 Donut</h3>

  <p class="price">$3</p>

  <button>Add to Cart</button>

</div>
```

```
<div class="food-card">

  <h3> 🥤 Cold Drink</h3>

  <p class="price">$2</p>

  <button>Add to Cart</button>

</div>
```

```
</div>

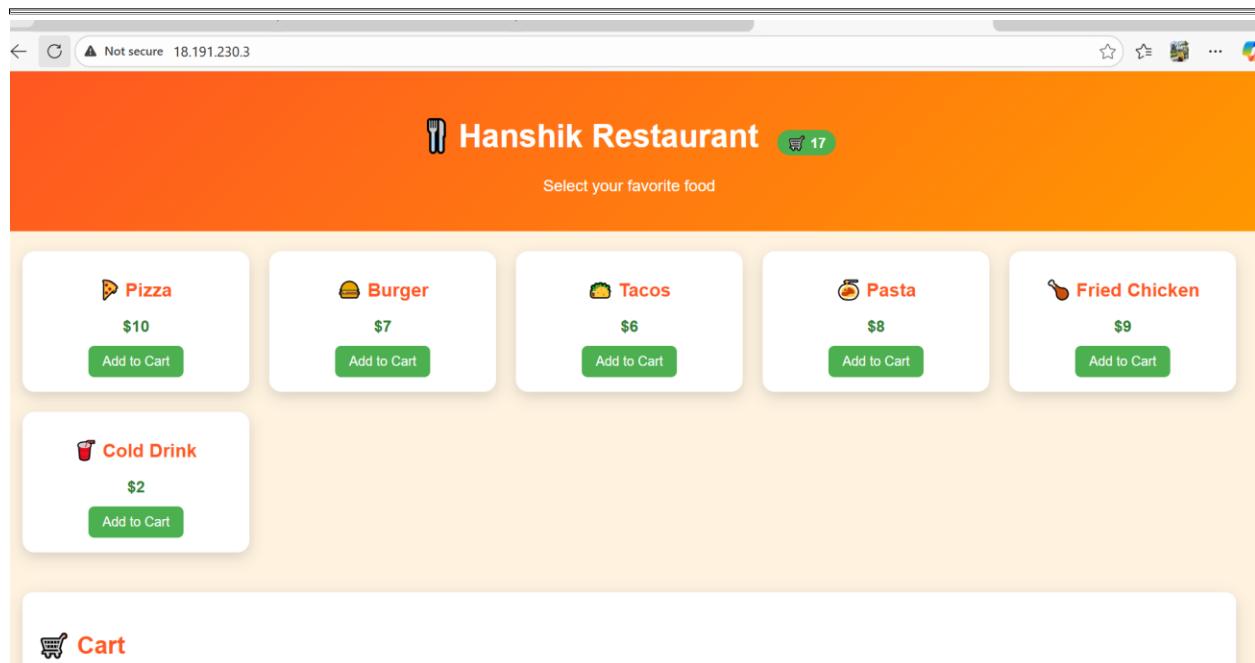
</body>

</html>
```

4. Saved and exited the file.
5. Refreshed the browser to verify that the custom web application was displayed.

## Deliverable

- Screenshot of the custom web application page running on the EC2 instance.



## Conclusion

This project successfully demonstrated how to deploy a basic web application on AWS using an EC2 instance and the Apache2 web server. Through this exercise, core concepts of cloud computing, Linux server management, security configuration, and web hosting were applied in a practical environment.

The deployed application confirms that the EC2 instance, security groups, Apache service, and web content are all functioning correctly.