Deploying an Apache2 HTTP Application on Ubuntu in AWS & Troubleshooting Downtime (Step-by-Step Blog)

Introduction

In real-world cloud environments, deploying an application is only half the job. The real challenge begins when the server goes down or the application becomes inaccessible.

This blog is written for beginners and freshers who want to understand: - How to deploy an Apache2 web server using HTTP on Ubuntu - How AWS VPC, Subnet, Route Table, NACL, and Security Groups affect application access - How to troubleshoot downtime step by step (no guessing)

Architecture Flow

User → Internet → VPC → Public Subnet → Route Table → NACL → Security Group → EC2 (Ubuntu + Apache2)

Step 1: Create AWS VPC

- VPC CIDR: 10.0.0.0/16
- Purpose: Private network for your application

Step 2: Create Public Subnet

- Subnet CIDR: 10.0.1.0/24
- Enable **Auto-assign Public IPv4**

A public subnet allows the EC2 instance to communicate with the internet.

Step 3: Create and Attach Internet Gateway (IGW)

- Create an Internet Gateway
- Attach it to the VPC

Without an IGW, the application cannot be accessed from the browser.

Step 4: Configure Route Table

Add the following route:

Destination: 0.0.0.0/0
Target: Internet Gateway

Associate this route table with the public subnet.

Step 5: Configure Network ACL (NACL)

Inbound Rules

- Allow HTTP (80)
- Allow SSH (22)

Outbound Rules

- Allow ALL traffic

NACL is **stateless**, so both inbound and outbound rules are mandatory.

Step 6: Launch Ubuntu EC2 Instance

- AMI: Ubuntu
- Subnet: Public Subnet
- Auto-assign Public IP: Enabled

Security Group Rules

- Allow SSH (22)
- Allow HTTP (80)

Step 7: Install Apache2 on Ubuntu

Update packages:

```
sudo apt update -y
```

Install Apache2:

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

Verify Apache2 service:

```
systemctl status apache2
```

Step 8: Deploy Web Application

```
cd /var/www/html
sudo vi index.html
```

Step 9: Access Application Using HTTP

From browser:

18.221.234.66

If everything is configured correctly, the web page will load successfully.

Downtime Scenario: Website Is Not Loading

Problem Statement

Users report: > “The website is not opening”

This is an unplanned downtime situation.

Step-by-Step Troubleshooting (No Guessing)

Step 1: Check Route Table

Ensure the subnet has:

0.0.0.0/0 → Internet Gateway

If missing, internet traffic will not reach the instance.

Step 2: Check NACL Rules

Verify: - Inbound port 80 → ALLOW - Outbound traffic → ALLOW

Incorrect NACL rules can silently block traffic.

Step 3: Check Security Group

Inbound rules must allow: - SSH (22) - HTTP (80)

Security Groups are **stateful**, so return traffic is allowed automatically.

Step 4: Check OS Firewall (UFW)

sudo ufw status

If active, ensure HTTP is allowed:

Sudo ufw disable
sudo ufw enable

Step 5: Check Server Health

Uptime
df -h
free -m

Look for: - High CPU usage - Disk full issues - Memory exhaustion

Step 6: Check Apache2 Service

Systemctl status apache2

Restart if stopped:

sudo systemctl restart apache2

Step 7: Verify Port Listening

ss -ntpl

Apache2 must be listening on port 80.

Step 8: Check Apache Logs (Most Important Step)

tail -f /var/log/apache2/error.log

Logs show the exact root cause of the issue.

Step 9: Test Locally

IP ADDRESS

- Works locally → Network issue
- Fails locally → Apache or application issue

Root Cause Analysis (RCA)

After fixing the issue, always document: - What caused the downtime? - How was it fixed? - How can it be prevented?

Best Practices

- Enable monitoring and alerts
- Regularly review NACL and firewall rules
- Monitor Apache logs
- Follow layered troubleshooting: Network → OS → Application

Conclusion

Deploying an Apache2 HTTP application on Ubuntu in AWS requires a strong understanding of both AWS networking and OS-level troubleshooting. By following a structured, step-by-step approach, downtime can be resolved quickly and confidently.

This approach reflects real production troubleshooting, not guesswork.