



Placement Empowerment Program

Cloud Computing and DevOps Centre

Implement DNS for Your Application: Set up a DNS record to map your web application's IP or load balancer to a domain name.

Name: Akshaya S

Department: CSE

Introduction

In cloud computing, establishing a proper **Domain Name System (DNS)** configuration is essential for ensuring that applications are accessible over the internet. AWS offers **Route 53**, a highly available and scalable DNS web service, which allows users to manage domain names and point them to AWS resources like EC2 instances. This Proof of Concept (PoC) demonstrates the process of creating and configuring a DNS record in **AWS Route 53**, pointing it to a web server hosted on an **EC2 instance**, thus making the web application accessible via a custom domain name.

Overview

This PoC involves:

1. **Launching an EC2 instance** to serve a web application.
2. **Setting up a web server** (Apache or Nginx) on the EC2 instance to host the application.
3. **Creating a hosted zone** in AWS Route 53 for a custom domain (e.g., myapp.local).
4. **Configuring an A record** in Route 53 to map the domain to the EC2 instance's public IP.
5. **Modifying the hosts file** on a local machine to test the custom domain name before making it publicly available.
6. **Testing the configuration** by accessing the application using the custom domain name.

Objective

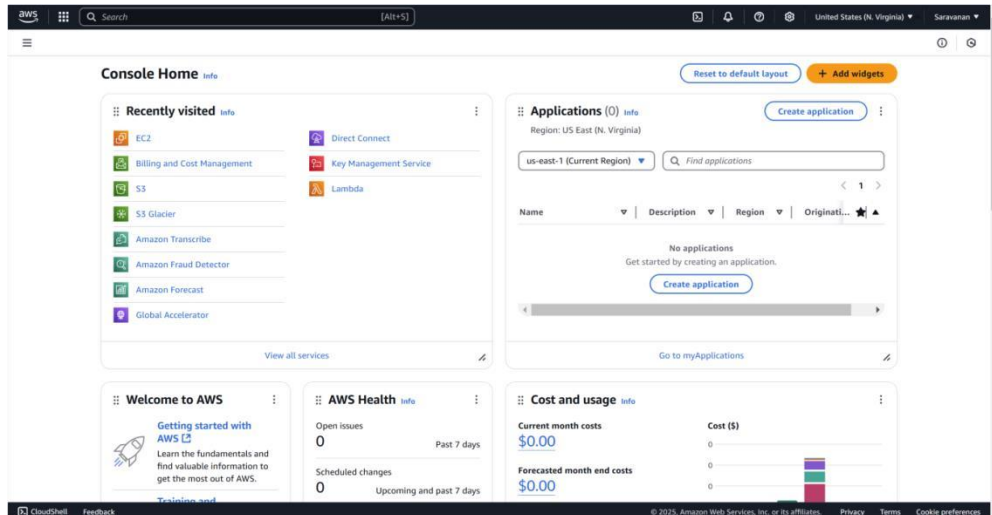
The main objectives of this PoC are:

- 1. Familiarize with Route 53 DNS configuration:** Understand how to use AWS Route 53 to manage domain names and map them to cloud resources.
- 2. Learn EC2 setup and configuration:** Gain hands-on experience in launching an EC2 instance and configuring a web server to serve a web application.
- 3. Enable custom domain access:** Configure a custom domain name to point to the EC2 instance, ensuring that the web application is easily accessible through the domain.
- 4. Test and verify the configuration:** Ensure that the domain correctly points to the EC2 instance by testing it in a browser and troubleshooting any issues.

Step-by-Step Overview

Step 1:

1. Go to [AWS Management Console](#).
2. Enter your username and password to log in.



Step 2:

Launch an instance named **route instance** .

Configure Security Group:

Add a **new security group** with a rule for **HTTP** (port 80) and **SSH** (port 22).

For HTTP, set the source to **Anywhere (0.0.0.0/0)** to allow access via the web.

For SSH, set the source to your **IP** (recommended for security), or use **Anywhere** for now.

Create a Key Pair (or use an existing one) and download the key file (.pem).

Review and click **Launch**.

aws

Search

[Alt+S]

United States (N. Virginia)

Saravanan

Instances (1/1) Info

Last updated less than a minute ago

Connect

Instance state

Actions

Launch instances

Find instance by attribute or tag (case-sensitive)

All states

<input checked="" type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
<input checked="" type="checkbox"/>	route instance	i-004154969b120bb8c	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-52-91-79-69.con

i-004154969b120bb8c (route instance)

DetailsStatus and alarmsMonitoringSecurityNetworkingStorageTags

Security details

IAM Role

Owner ID

Launch time

Security groups

Inbound rules

CloudShell

Feedback

© 2025, Amazon Web Services, Inc. or its affiliates. PrivacyTermsCookie preferences

aws

Search

[Alt+S]

United States (N. Virginia)

Saravanan

EC2 > Security Groups > sg-091ff45e68e5f742f - launch-wizard-29

Actions

Details

Security group name

Security group ID

Description

VPC ID

Owner

Inbound rules count

Outbound rules count

Inbound rules

Outbound rules

Sharing - new

VPC associations - new

Tags

Inbound rules (2)

Manage tags

Edit inbound rules

<input type="checkbox"/>	Name	Security group rule ID	IP version	Type	Protocol	Port range	Source
<input type="checkbox"/>	-	sgr-08cd3a57f47836e47	IPv4	SSH	TCP	22	0.0.0.0/0
<input type="checkbox"/>	-	sgr-0fe112bc7cae9defb	IPv4	HTTP	TCP	80	0.0.0.0/0

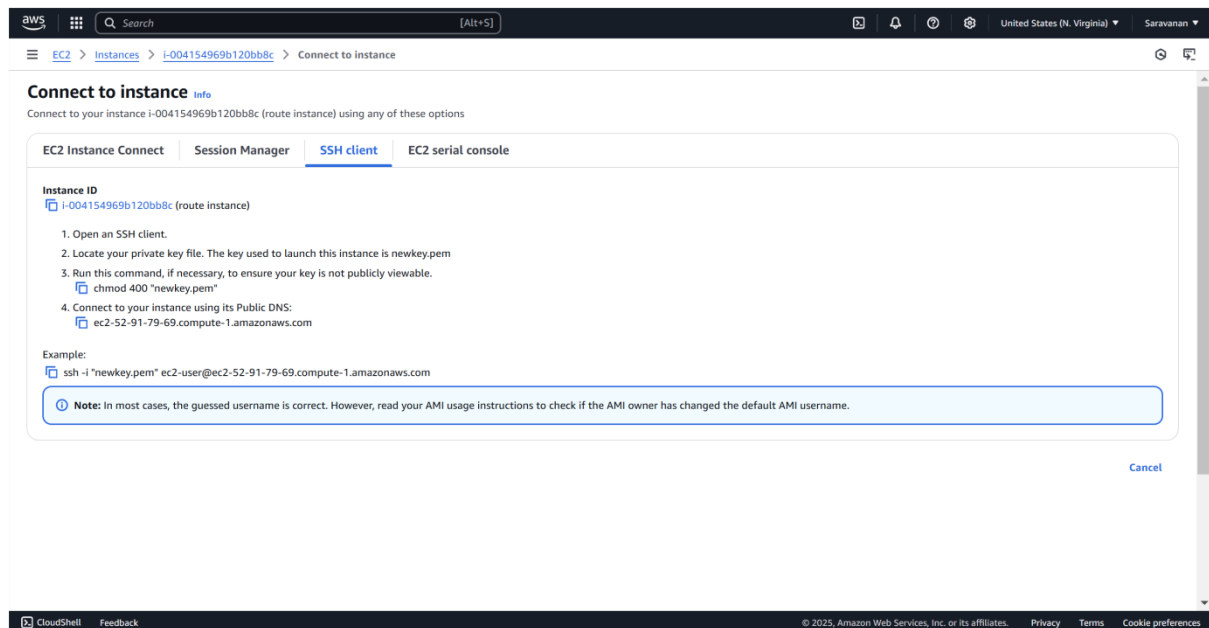
CloudShell

Feedback

© 2025, Amazon Web Services, Inc. or its affiliates. PrivacyTermsCookie preferences

Step 3:

Click the 'Connect' option on your launched instance, go to the SSH client section, and copy the command provided under the 'Example' section.



Step 4:

Open PowerShell, navigate to the 'Downloads' directory where the downloaded key pair is located using the **cd Downloads** command

Paste the command copied from the EC2 Connect's SSH client section, replace the key pair name with your downloaded key (e.g., new.pem), press Enter, and type 'yes' when prompted.

```
PS C:\Users\Hi> cd downloads
PS C:\Users\Hi\downloads> ssh -i "newkey.pem" ec2-user@ec2-52-91-79-69.compute-1.amazonaws.com
The authenticity of host 'ec2-52-91-79-69.compute-1.amazonaws.com (52.91.79.69)' can't be established.
ED25519 key fingerprint is SHA256:V4SUI93qczBAzHQQxXWzd0m5epjEoS0SfhgKbDx0kw.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-52-91-79-69.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
#_
~_##### Amazon Linux 2023
NN\#####
NN\###|
NN\#/
NNV_!_>
NN_
NN_/_/!_/_/
NN_/_/!_/_/
```

Step 5:

Install Apache :

```
sudo yum install httpd -y
```

```
[ec2-user@ip-172-31-82-55 ~]$ sudo yum install httpd -y
```

Step 6:

Start Apache:

sudo systemctl start httpd Make

Apache start on boot:

```
sudo systemctl enable httpd
```

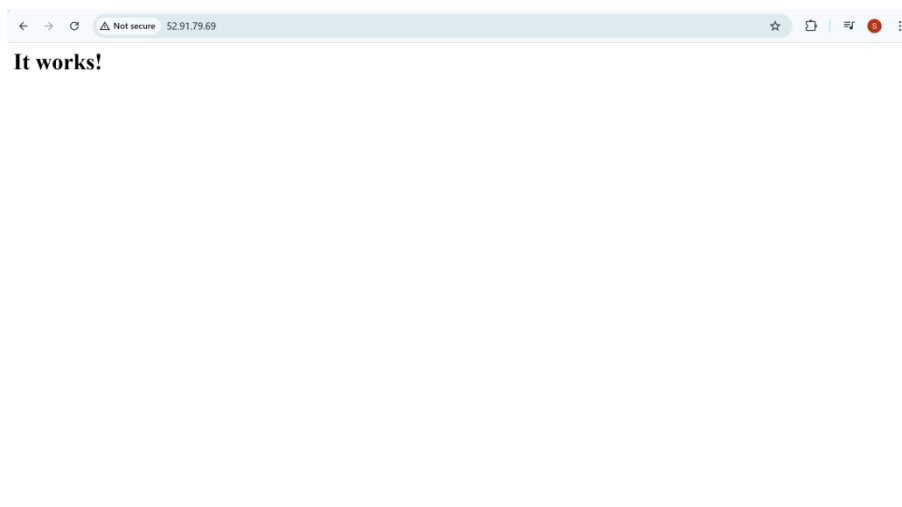
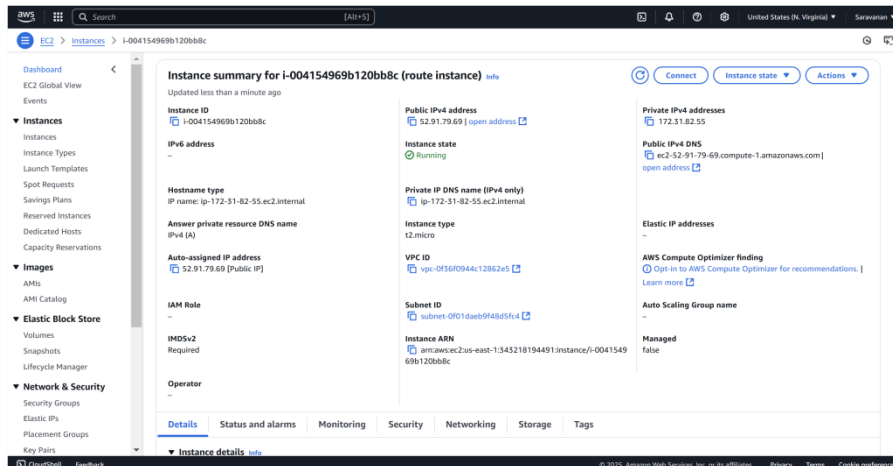
```
[ec2-user@ip-172-31-82-55 ~]$ sudo systemctl start httpd
[ec2-user@ip-172-31-82-55 ~]$ sudo systemctl enable httpd
```

Step 7:

Verify Apache is running:

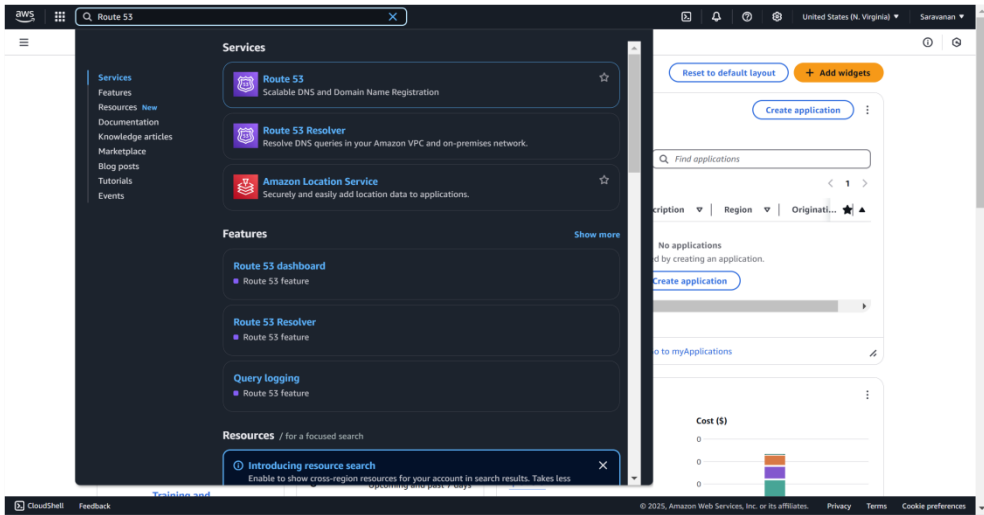
In your browser, enter the **EC2 public IP** (e.g., <http://<your-ec2public-ip>>).

You should see the **Apache default page**. This means your EC2 instance is set up to serve websites.



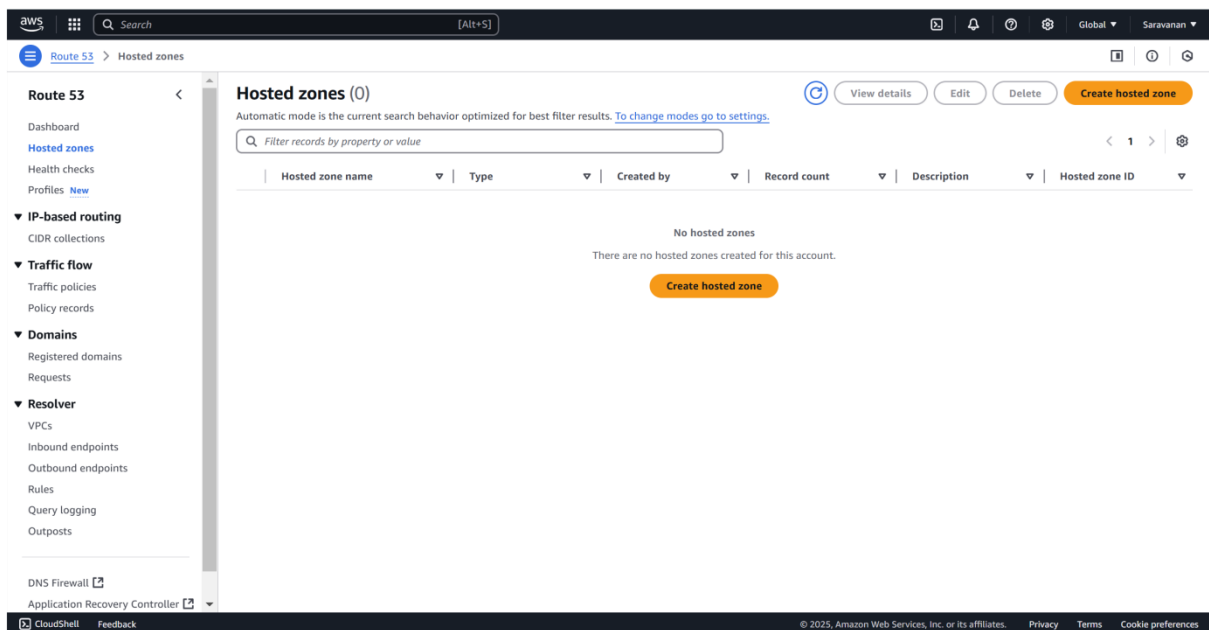
Step 8:

In the AWS Console, search for **Route 53** and select it.



Step 10:

1. Click on **Create hosted zone**.
2. Enter a **Domain Name** (e.g., myapp.local).
3. Set the **Type** to **Public Hosted Zone**.
4. Click **Create hosted zone**.

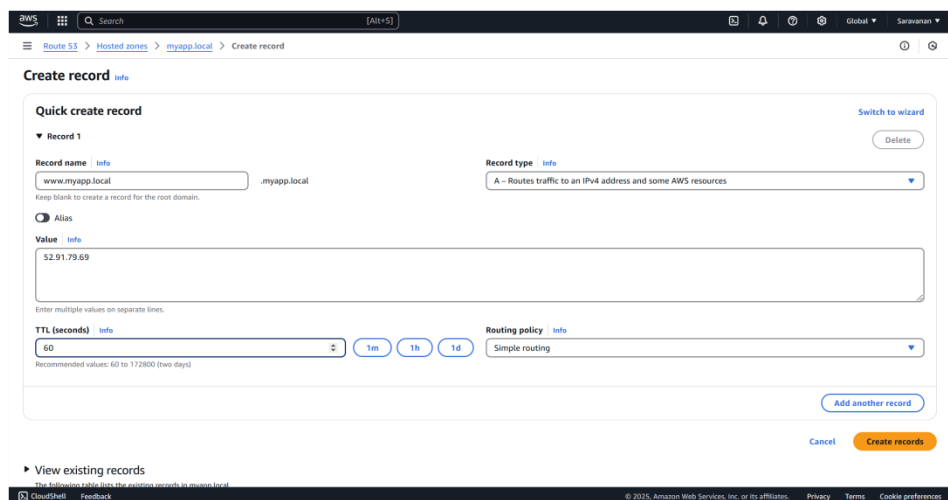
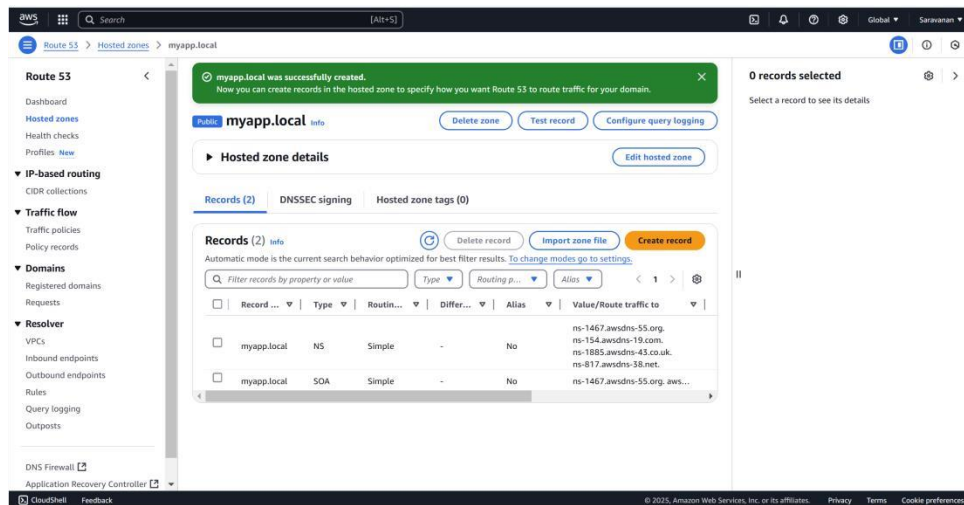


Step 11:

The screenshot shows the AWS Management Console interface for creating a hosted zone. The breadcrumb navigation at the top indicates the path: Route 53 > Hosted zones > Create hosted zone. The main heading is 'Create hosted zone' with an 'info' link. Below this is the 'Hosted zone configuration' section, which includes a description of a hosted zone and a 'Domain name' field containing 'myapp.local'. A 'Description - optional' field is also present, containing 'POC for DNS setup'. The 'Type' section shows two radio buttons: 'Public hosted zone' (selected) and 'Private hosted zone'. The 'Tags' section at the bottom is empty, with an 'Add tag' button. The footer of the console shows 'CloudShell', 'Feedback', and copyright information for Amazon Web Services, Inc.

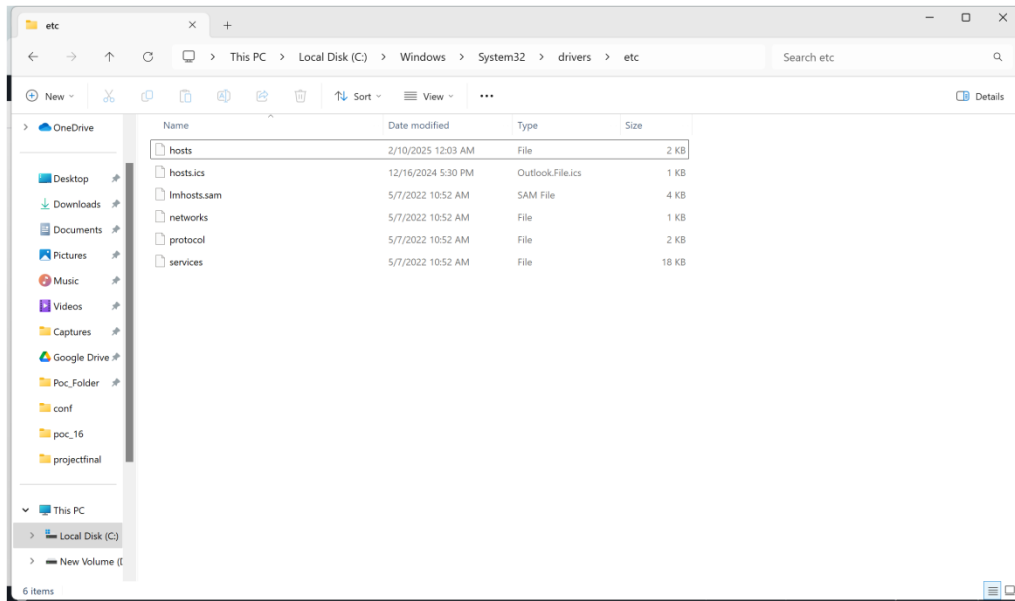
1. In your hosted zone, click **Create Record**.
2. **Record Name**: Leave it empty for the root domain (myapp.local),
3. **Record Type**: Select **A – IPv4 address**.
4. **Value**: Enter the **Public IP** of your EC2 instance.
5. **TTL**: Set to 60 seconds.
6. Click **Create records**.

Step 12:



1. Go to **FileExplorer > Open**.
2. Navigate to: **C:\Windows\System32\drivers\etc**.
3. In the file name field, type **hosts** and press **Enter**.

Step 13:



Step 12:

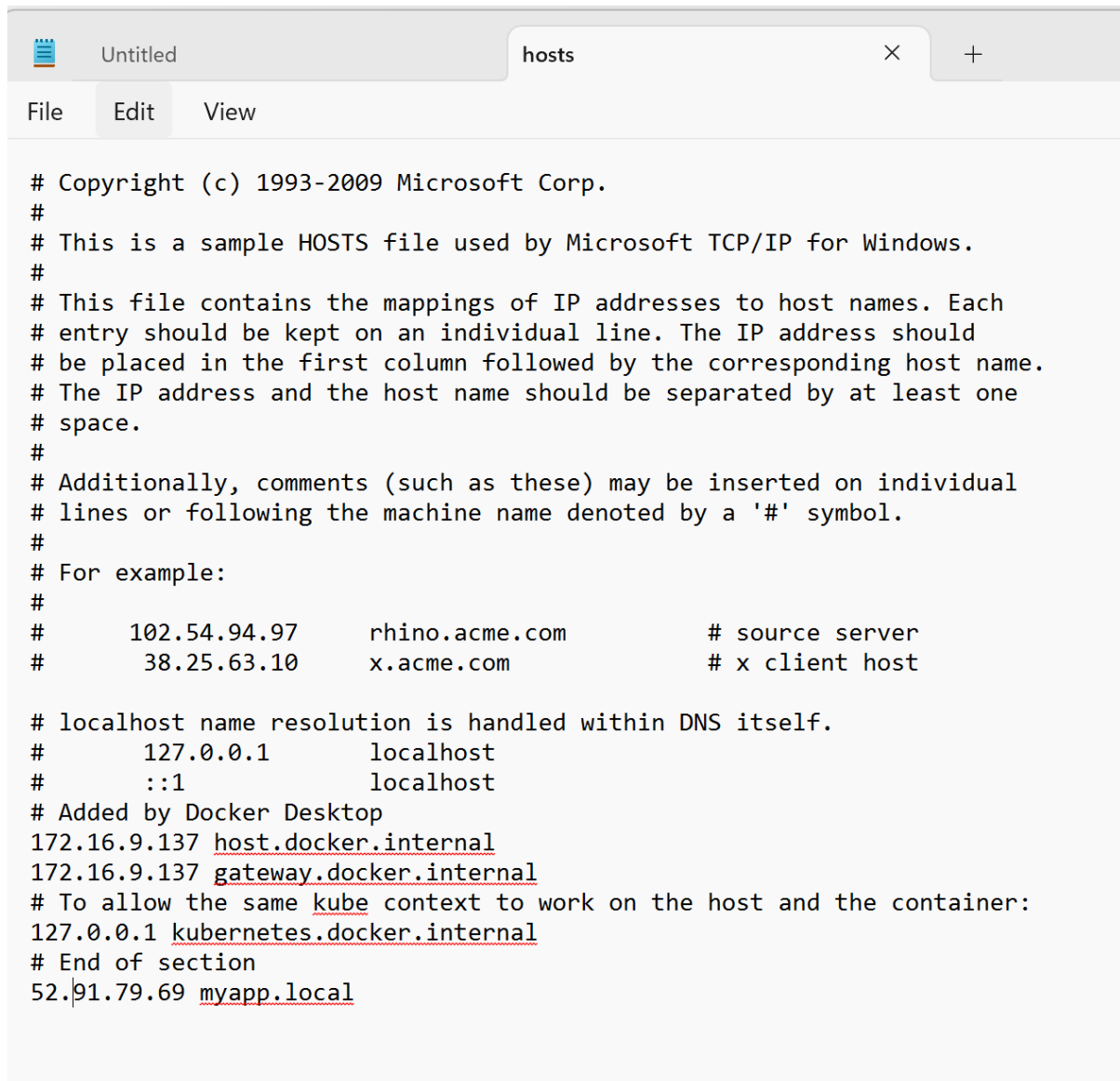
1. At the bottom of the file, add:

<Your EC2 Public IP> myapp.local

Replace <Your EC2 Public IP> with the public IP you copied.

(Eg: 52.91.79.69 myapp.local)

2. Save the file and close Notepad.

A screenshot of a text editor window with a tab titled 'hosts'. The window has a menu bar with 'File', 'Edit', and 'View'. The text inside the editor is the standard Windows hosts file content, including copyright information, usage instructions, and several IP-to-hostname mappings. The mappings include '102.54.94.97 rhino.acme.com', '38.25.63.10 x.acme.com', '127.0.0.1 localhost', '::1 localhost', '172.16.9.137 host.docker.internal', '172.16.9.137 gateway.docker.internal', '127.0.0.1 kubernetes.docker.internal', and '52.91.79.69 myapp.local'.

```
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#       102.54.94.97       rhino.acme.com           # source server
#       38.25.63.10       x.acme.com               # x client host


# localhost name resolution is handled within DNS itself.
#       127.0.0.1        localhost
#       ::1              localhost
# Added by Docker Desktop
172.16.9.137 host.docker.internal
172.16.9.137 gateway.docker.internal
# To allow the same kube context to work on the host and the container:
127.0.0.1 kubernetes.docker.internal
# End of section
52.91.79.69 myapp.local
```

Step 12:

Open your **web browser**.

Type `myapp.local` in the address bar and press **Enter**.

You should see the Apache default page

