AWS DNS Services Project 2

Contents

[Hosting a static website on Amazon EC2 2](#_Toc195017551)

[Project 2: Hosting on website on EC2 Apache Server 2](#_Toc195017552)

[Hosted zones 2](#_Toc195017553)

[Let’s get spin up EC2 instance 2](#_Toc195017554)

[SSH to EC2 and update packages 4](#_Toc195017555)

[Install httpd (Apache Server) 4](#_Toc195017556)

[Copy file to EC2 7](#_Toc195017557)

[Configure Route53 10](#_Toc195017558)

[Let’s configure apache. 13](#_Toc195017559)

[Setup Virtual Host 13](#_Toc195017560)

[Enable EPEL Repo (to access more packages) 15](#_Toc195017561)

[Install Certbot + Apache Plugin (Let’s Encrypt) 17](#_Toc195017562)

[Run Certbot to Configure SSL & HTTPS 18](#_Toc195017563)

[Add Auto-Renewal via Cron 19](#_Toc195017564)

[Configure Route 53 DNS Records 20](#_Toc195017565)

[(Optional) Check Public IP of Your EC2 22](#_Toc195017566)

[Verify Apache & SSL Status 22](#_Toc195017567)

[Test Your Site 22](#_Toc195017568)

[🔐 Cert File Locations (Auto-managed by Certbot) 22](#_Toc195017569)

[Final Result 23](#_Toc195017570)

# Hosting a static website on Amazon EC2

## Project 2: Hosting on website on EC2 Apache Server

Apache is a cross-platform web server designed to host web applications and websites. Websites can be either static or dynamic in nature. For this we need to spin-up an EC2 instance and install Apache server on it. Apache owns about a third of the total market share, sitting below nginx as of 2025. It is scalable, secure, cross-platform, easy to configure and best of all it is a Free and Open-source software. Companies like Cisco, IBM, Salesforce and Adobe leverage this web server’s power to host their website.

### Hosted zones

Specify how you want Route 53 to respond to DNS queries for a domain such as example.com.

### Let’s get spin up EC2 instance



Figure 1 Search EC2 on Search bar.



Figure 2 EC2 page. Now go to instances and create or spin up an already existing instance.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 3 I have started F\_Instance as I already created for other projects.

### SSH to EC2 and update packages

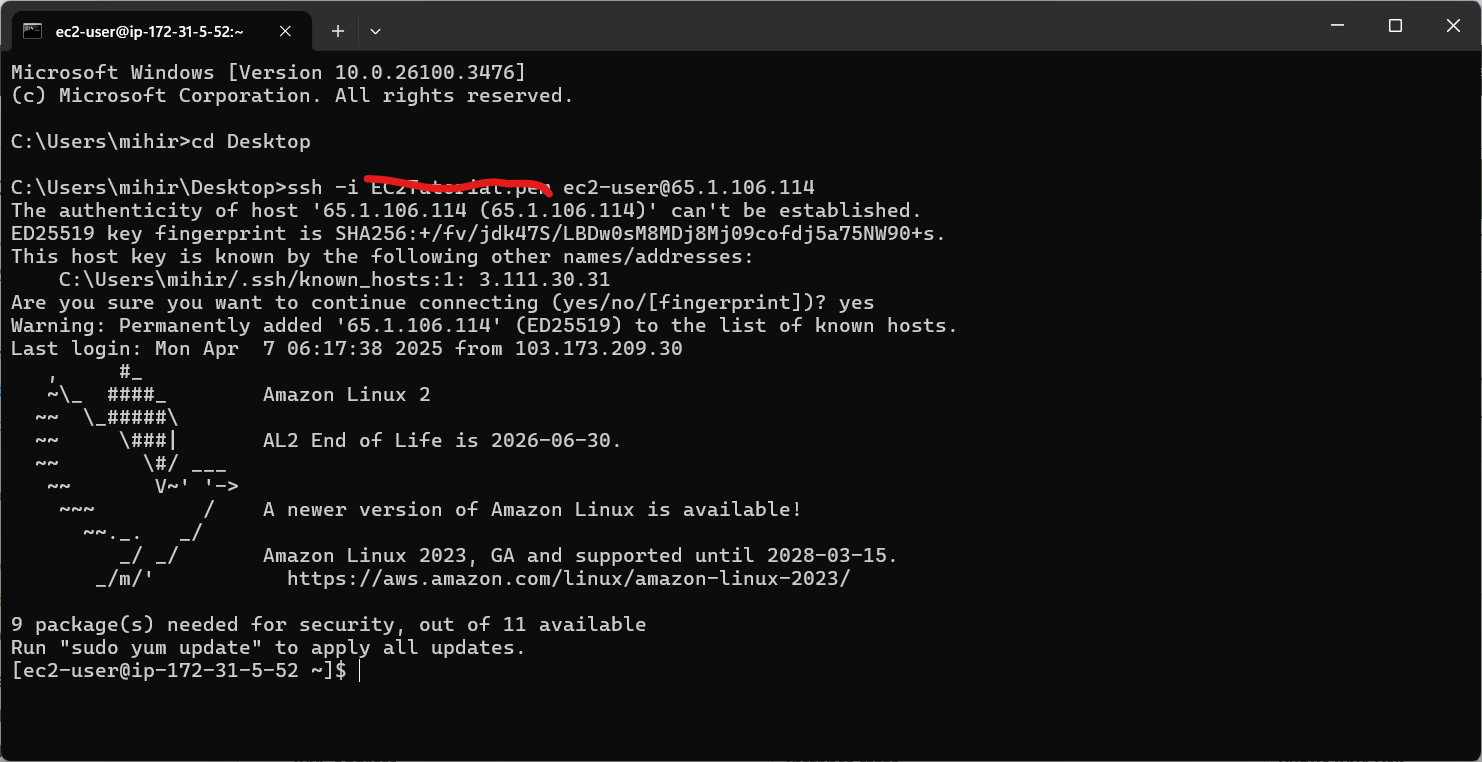


Figure 4 ssh into your instance. I have colored my key for security purpose.

Enter following commands:

ssh -i ‘Path/to/your/key.pem’ ec2-user@your-instance-public-ip 🡪This will connect to your EC2 instance using SSH

sudo yum update -y 🡪 This will update your AMI, I have used Amazon Linux2

### Install httpd (Apache Server)

sudo yum install -y httpd 🡪 this will install apache server

sudo systemctl start httpd 🡪 this will start server

sudo systemctl enable httpd 🡪 this will configure httpd to start automatically at boot

**Breakdown:**

* sudo: Run the command with superuser privileges.
* systemctl: The command-line tool to manage systemd services.
* enable: Configures the service to **start automatically** when the system boots.
* httpd: The name of the **Apache web server service**.

I have executed all the above commands and will stick final screenshot.

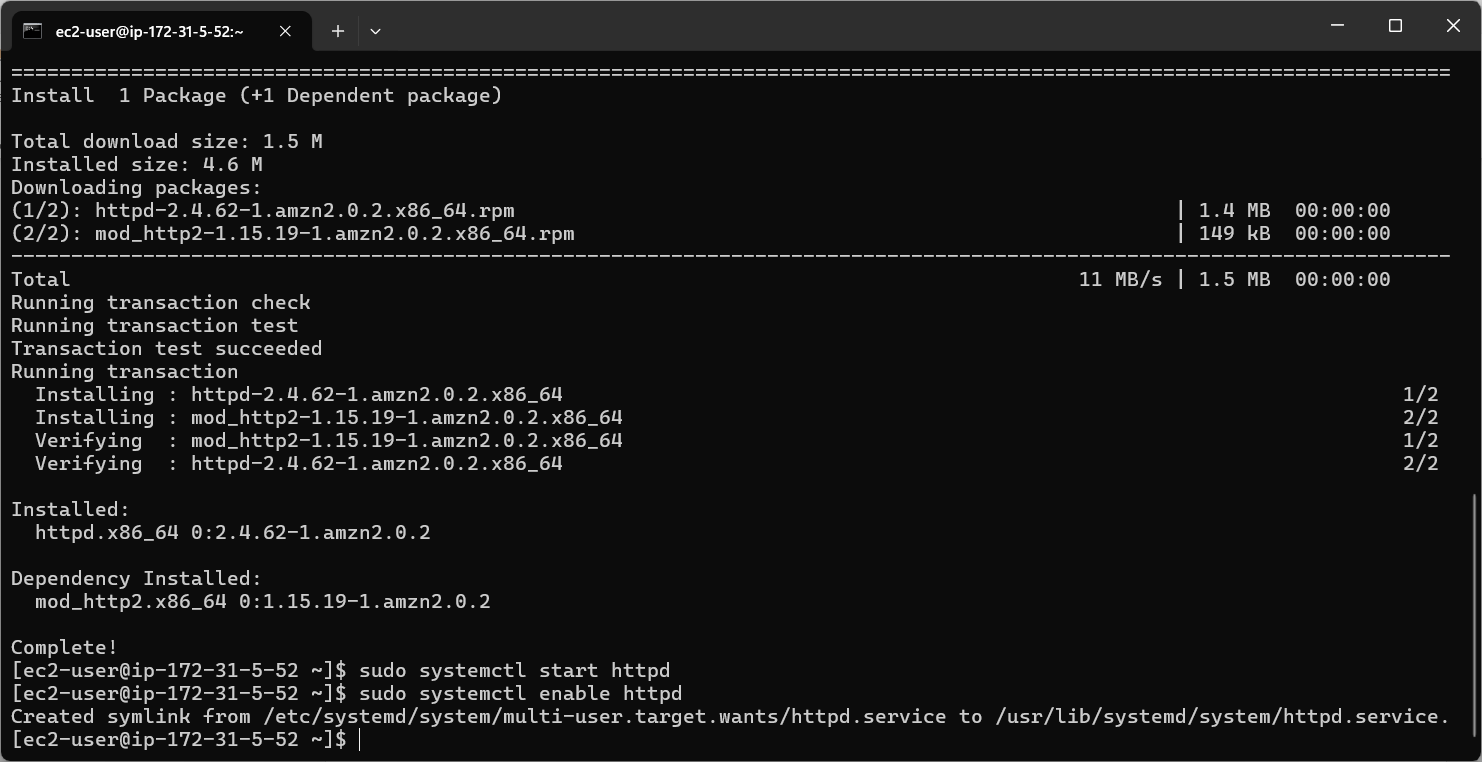


Figure 5 Installed and enabled httpd.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 6 If you copy and paste your public IP on another tab you might see an Apache webpage, or it will display it works. Since I already went ahead and uploaded my website its loading my web page.

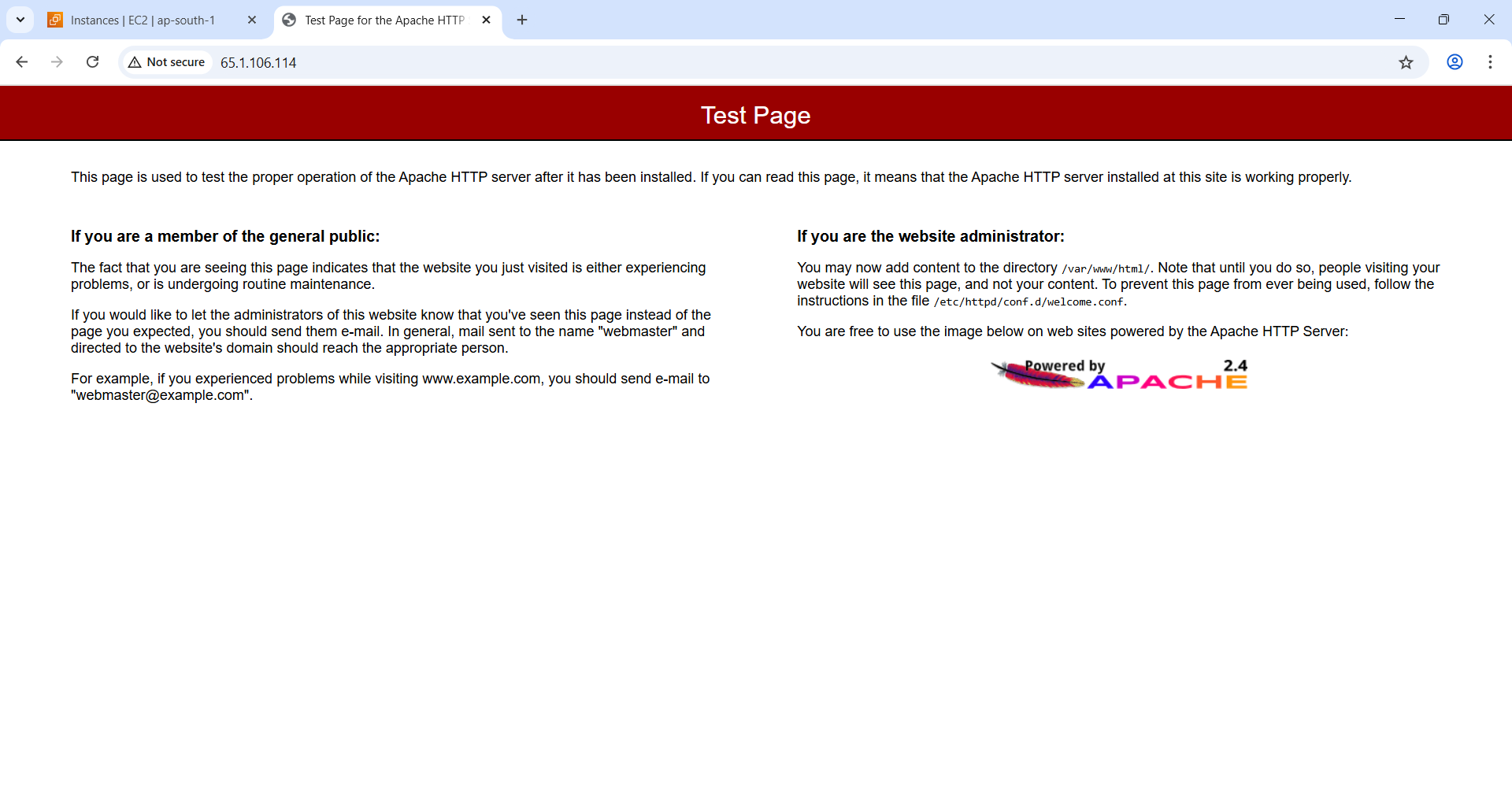


Figure 7 I move my index file to demonstrate what could you possibly see on first launch.

Copy your public IP from instance and paste it onto new tab.

### Copy file to EC2

In the next step we will leverage SCP or Secure Copy command that runs over SSH to copy files securely into your instance.

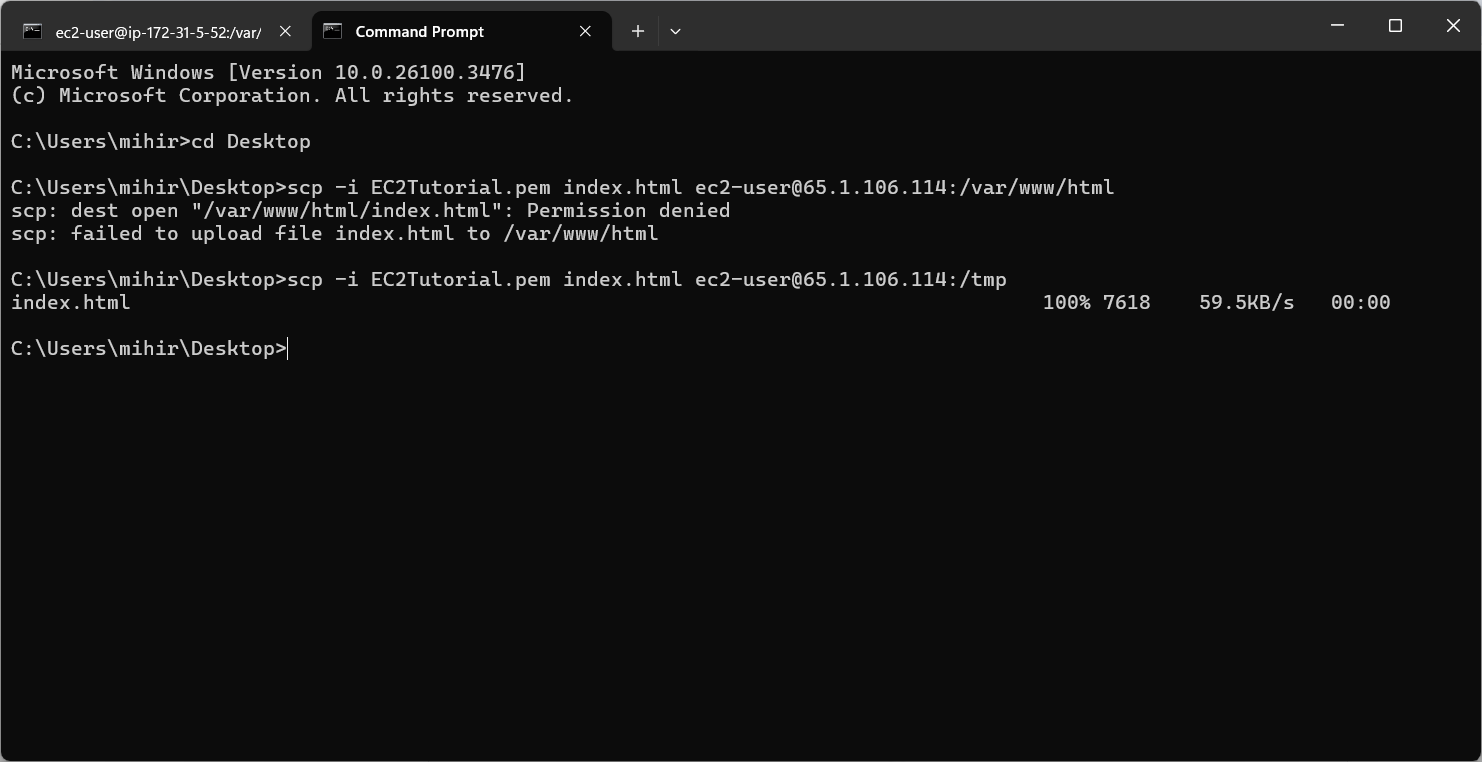


Figure 8. Copied index page to tmp folder.

If you look closely why it failed once, because I want to highlight a point that is if you try top copy directly into a /var/www/html you don’t have permission. The directory might be set to read only. So instead we copy the file to /tmp directory.

**Why /tmp?**

* **Writable by all users**: Most Linux systems allow all users to write to /tmp, so you don’t run into permission issues.
* **Temporary use**: It’s a good place for temporary files you don’t need to persist long-term.
* **Cleans up on reboot**: /tmp often gets cleared when the system reboots, keeping it clean.

**Example using /tmp:**

bash

CopyEdit

# Copy to /tmp instead of home directory

scp -i SSHKEY.pem index.html ec2-user@IP:/tmp/

# SSH into instance and move with sudo

ssh -i SSHKEY.pem ec2-user@IP

sudo mv /tmp/index.html /var/www/html/

**🆚 Home Directory vs /tmp**

| **Path** | **Use Case** | **Persistence** | **Permissions** |
| --- | --- | --- | --- |
| /home/ec2-user | Personal files, scripts, long-term use | Persistent | Owned by user |
| /tmp | Quick transfers, one-off operations | Temporary | Writable by everyone |

So: copying to /home/ec2-user or /tmp is just about where you have permission — both are fine. /tmp is just more neutral and not tied to the user's home.

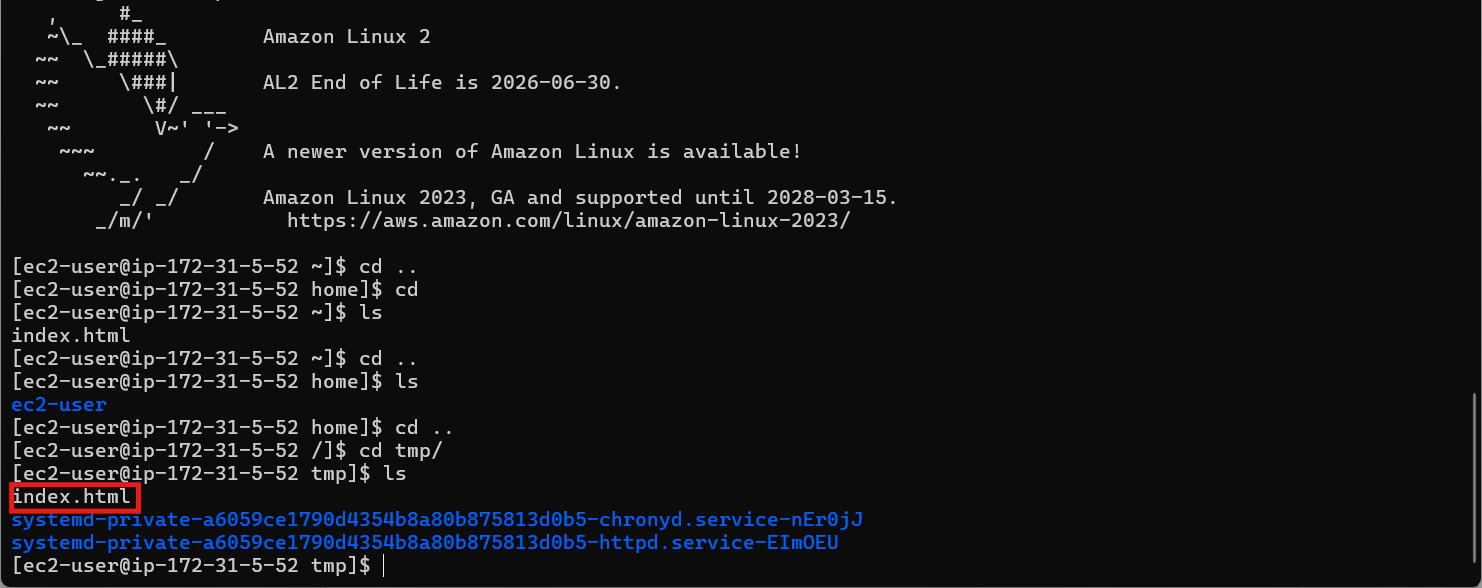


Figure 9 It is copied to tmp file. We will now move the file to /var/www/html directory

Command:sudo mv index.html /var/www/html

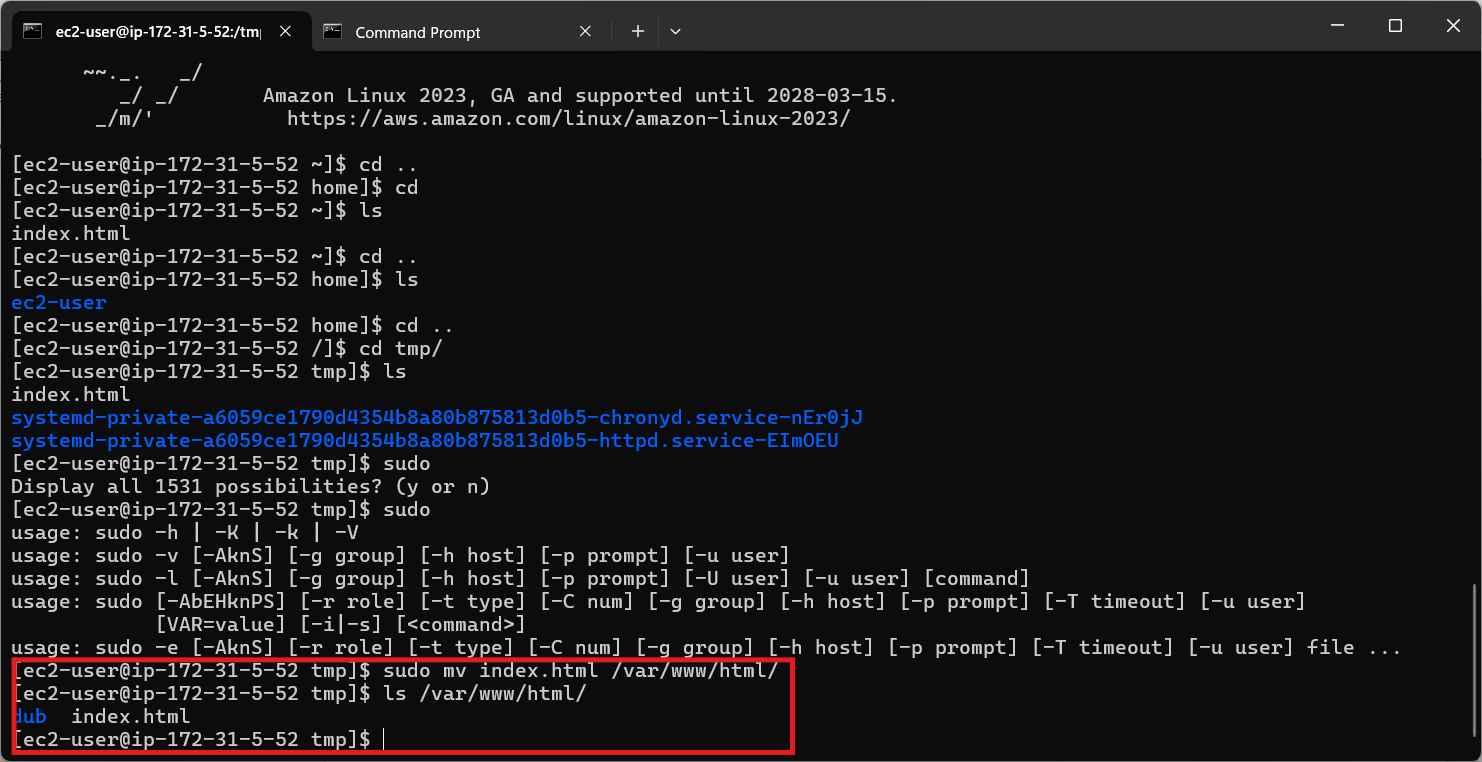


Figure 10 If you see it is already moved to /var/www/html directory where we place our index file which will load when EC2 is working.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 11 If you reload your website should be available.

### Configure Route53

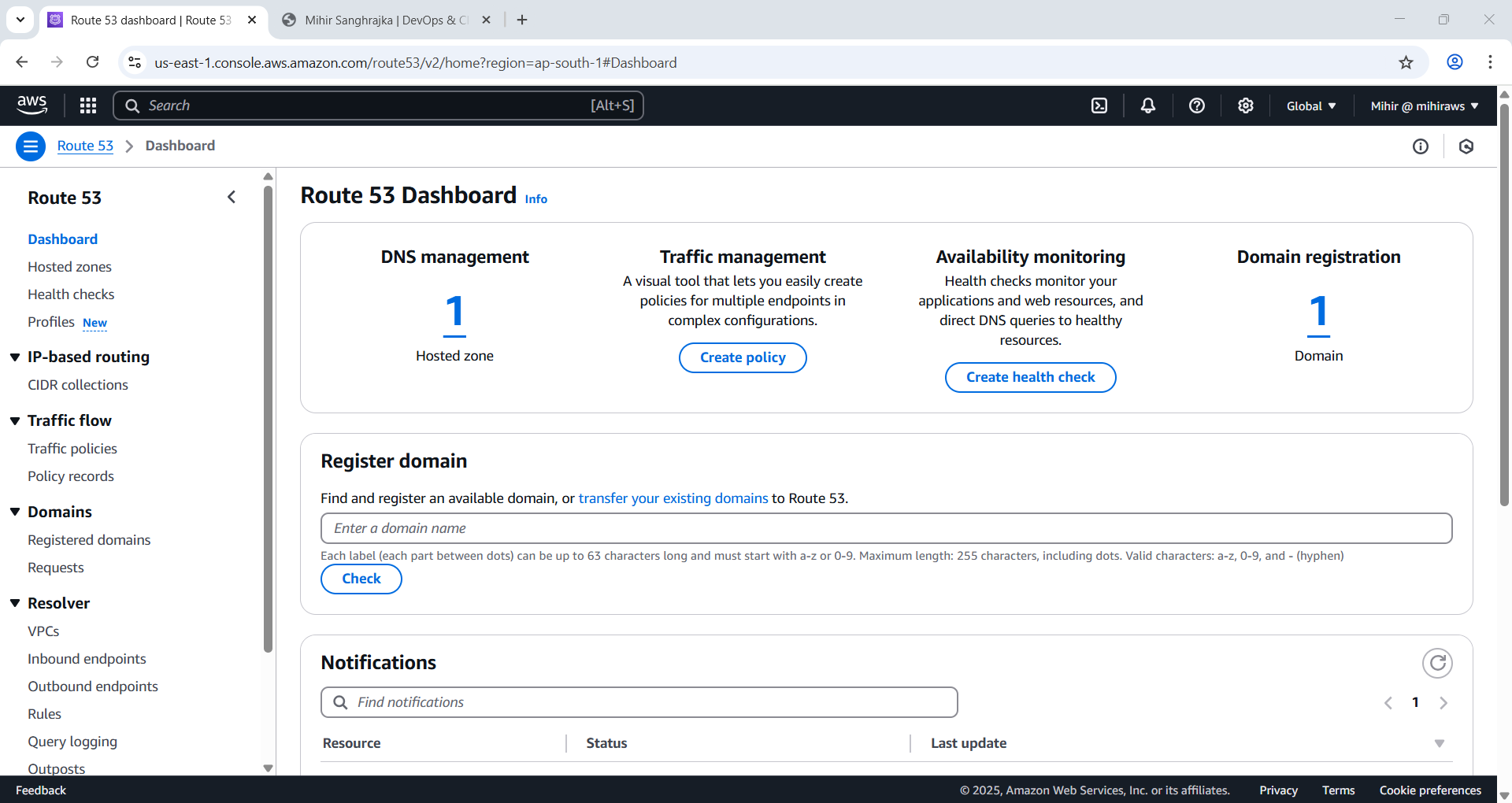


Figure 12 Go to Route53 page.

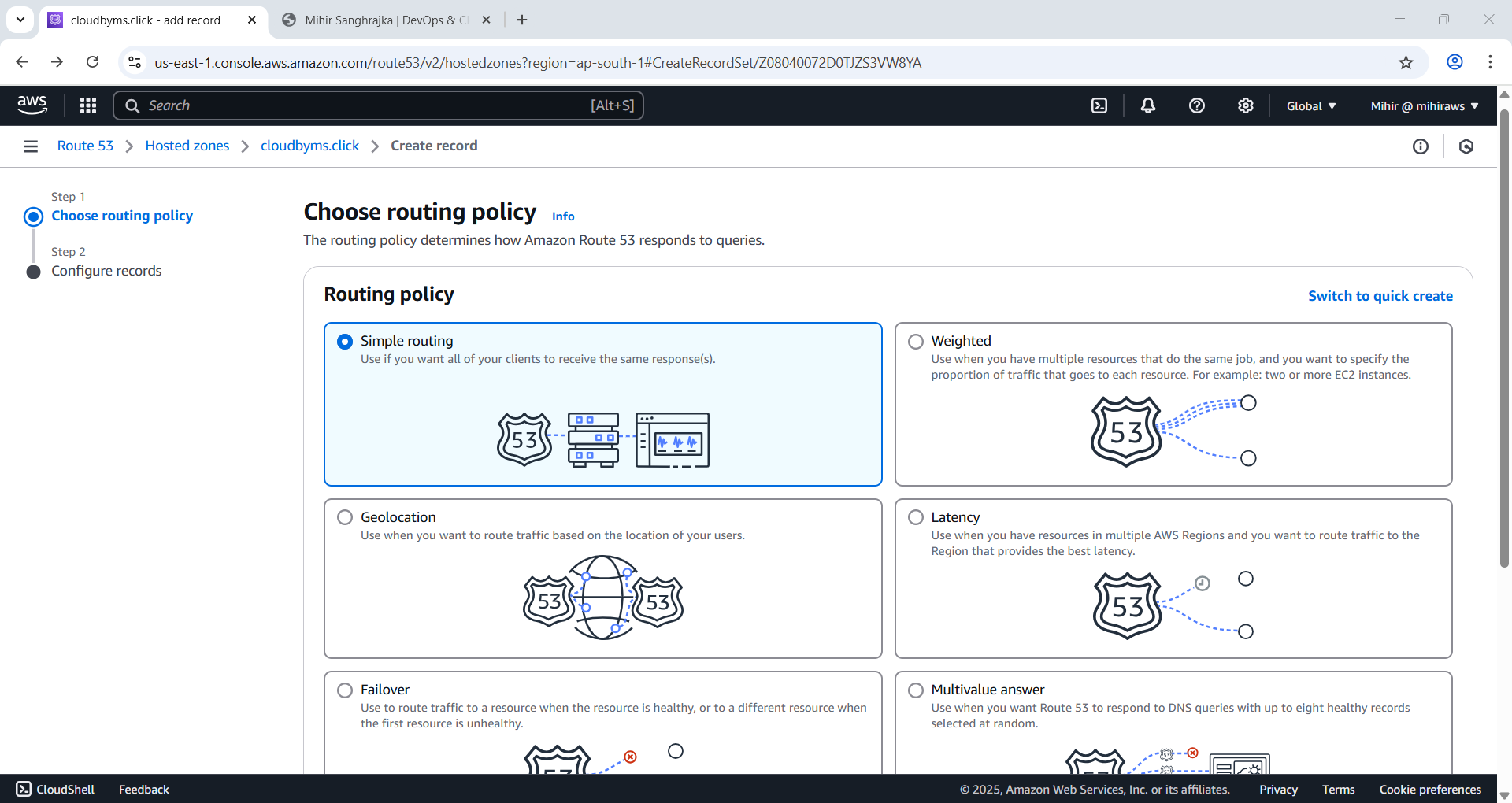


Figure 13 Select Simple Routing

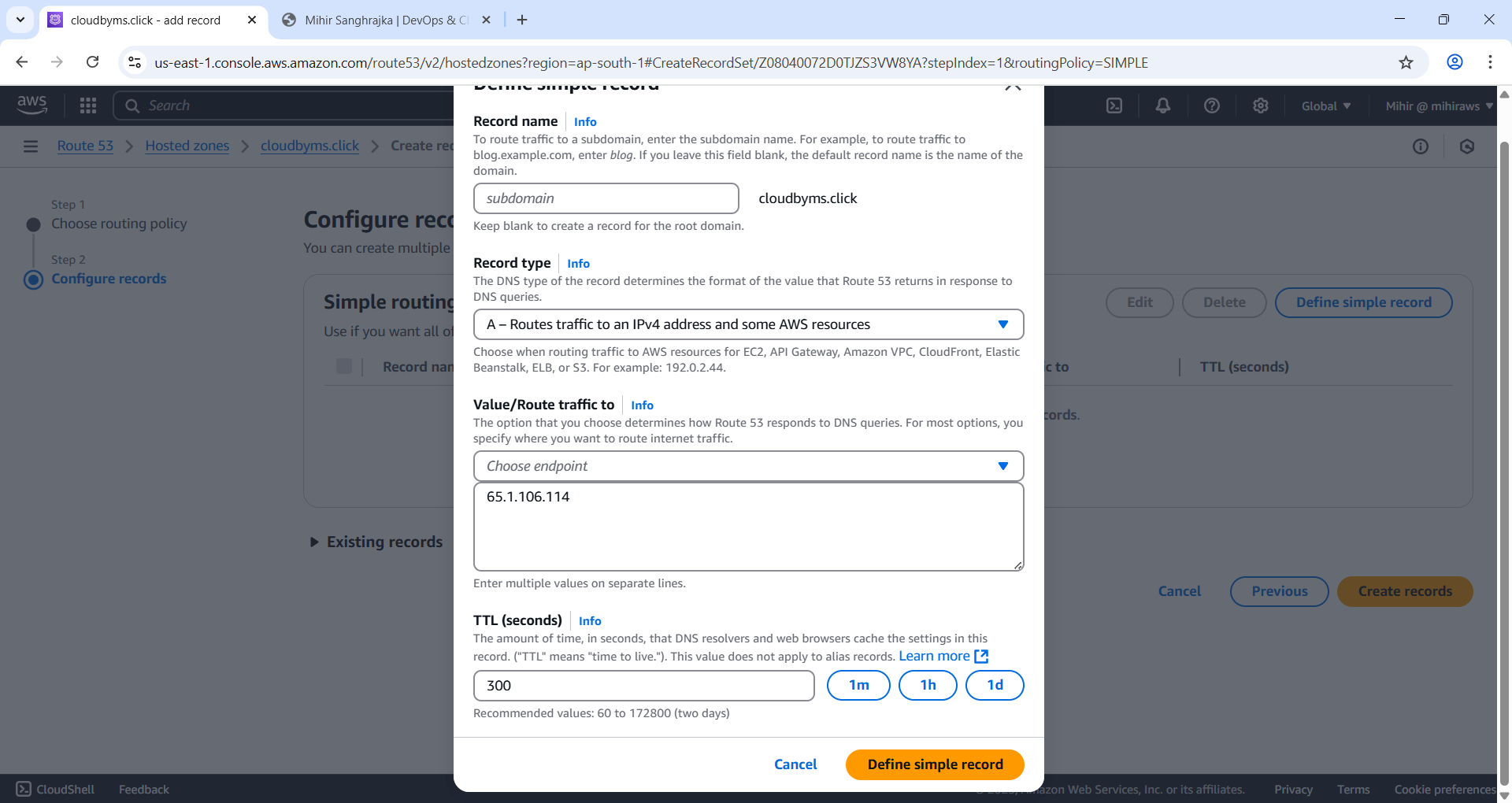


Figure 14 Select the displayed options

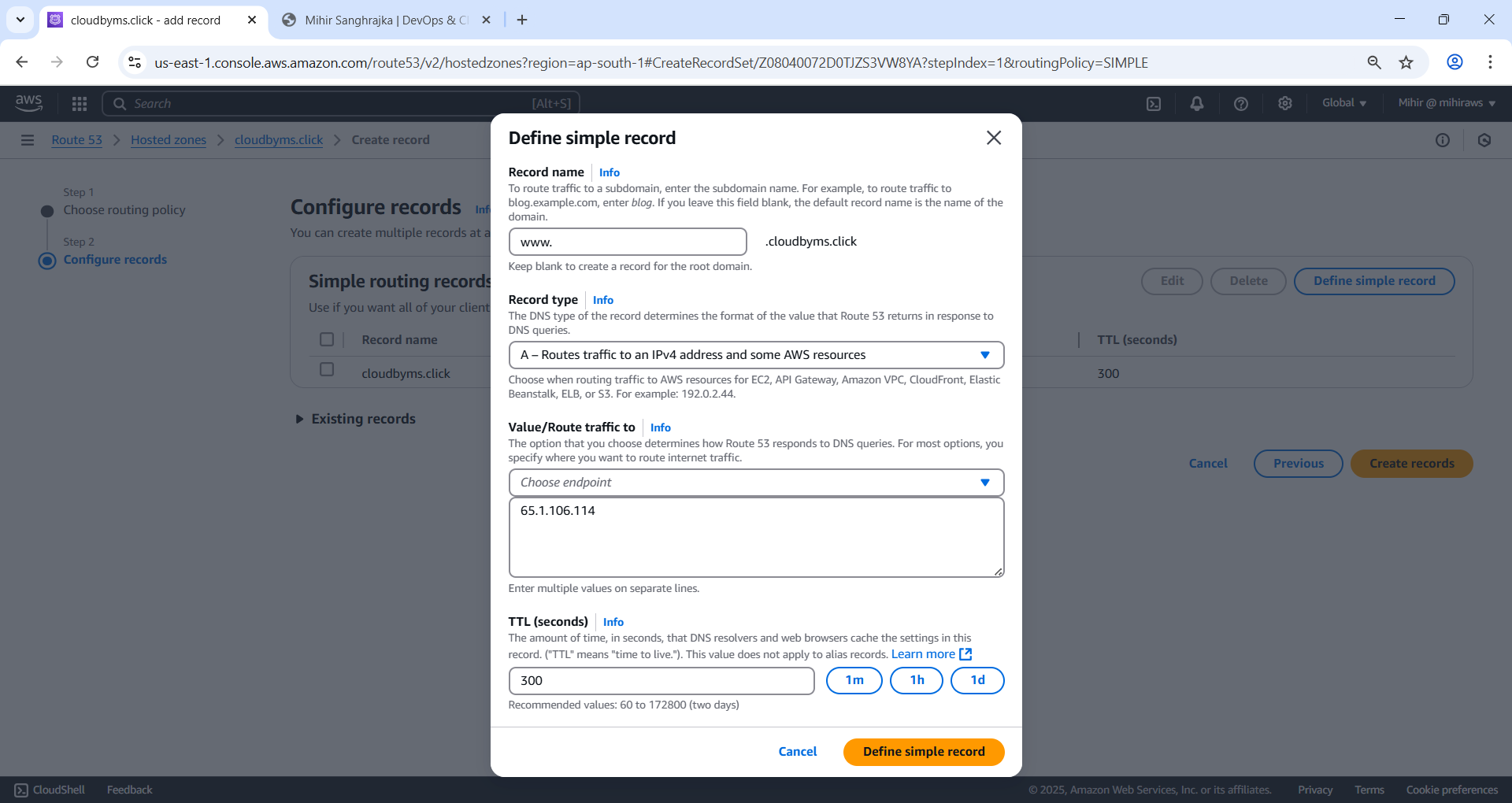


Figure 15 Repeat for www.

1. Under **Routing policy**, choose:  
**✅ Simple routing**

2. You’ll now see the form for record creation:

* **Record name**: Leave it **blank** for root domain (e.g., example.com)  
  Or type www if you're creating a record for www.example.com
* **Record type**: A – IPv4 address
* **Value**: Paste your **EC2 public IP address**
* **TTL (seconds)**: Leave as **300** (default)
* **Routing policy**: Confirm it's still on **Simple routing**
* **Evaluate target health**: Leave it **unchecked**

3. Click **Create records**

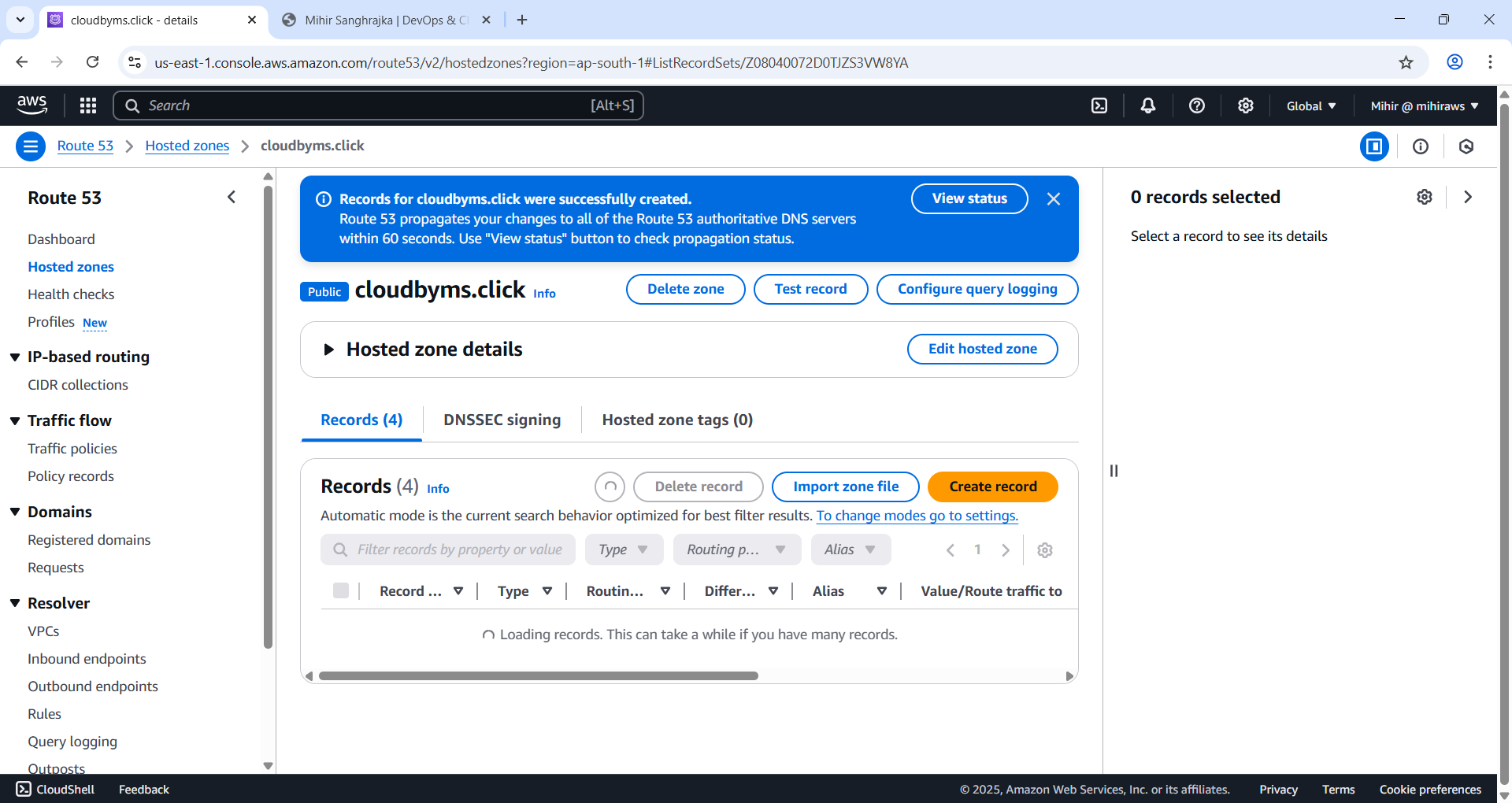


Figure 16. Successfully added routing information. Wait for 60 seconds.

Test the website is up?

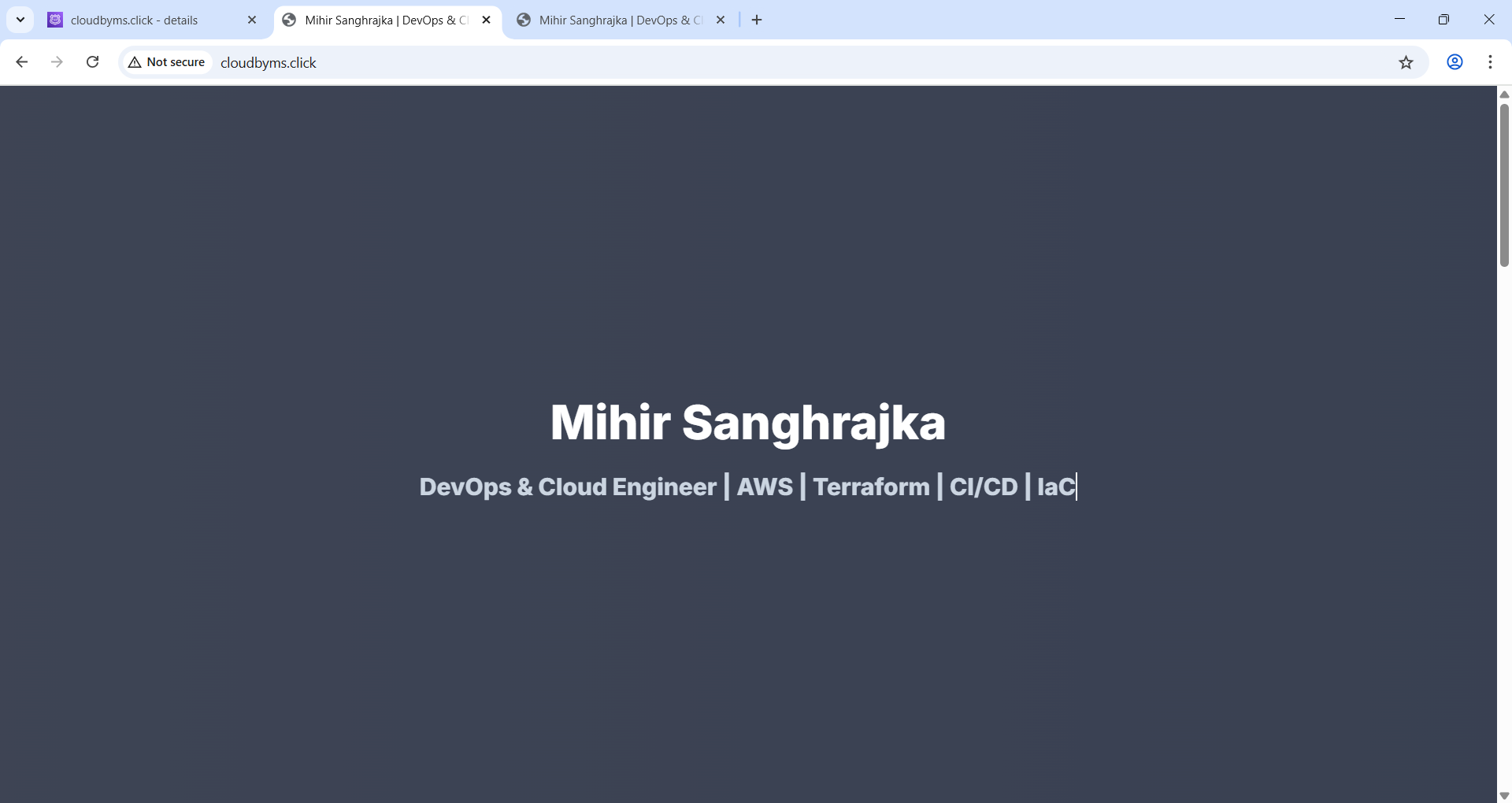


Figure 17 Hit [www.cloudbyms.click](http://www.cloudbyms.click)

### Let’s configure apache.

**Note: Amazon Linux doesn’t use apache2 directory and instead use httpd.**

### Setup Virtual Host

**A virtual host allows a single web server to host multiple websites or applications, each with its own domain name and configuration, while sharing the same underlying resources.**

Command: sudo nano /etc/httpd/conf.d/cloudbyms.click.conf

Paste:

<VirtualHost \*:80>

ServerName cloudbyms.click

ServerAlias www.cloudbyms.click

DocumentRoot /var/www/html

<Directory /var/www/html>

AllowOverride All

Require all granted

</Directory>

ErrorLog /var/log/httpd/cloudbyms.click-error.log

CustomLog /var/log/httpd/cloudbyms.click-access.log combined

</VirtualHost>

Command: sudo systemctl restart httpd

**A step-by-step recap of all the commands we executed to host your domain cloudbyms.click on Apache with HTTPS on an Amazon EC2 instance running Amazon Linux 2, along with what each command does.**

**🚀 FULL SETUP: Host Website on EC2 with HTTPS (Amazon Linux 2 + Apache + Certbot + Route 53)**

**✅ 1. Update & Install Apache (httpd)**

*sudo yum update -y*

*sudo yum install -y httpd*

**Updates all packages and installs Apache web server.**

**✅ 2. Start and Enable Apache**

*sudo systemctl start httpd*

*sudo systemctl enable httpd*

**Starts Apache now and ensures it runs on every reboot.**

### Enable EPEL Repo (to access more packages)

**Commands:**

*sudo amazon-linux-extras enable epel*

**🔧 What It Does:**

**This command enables the EPEL (Extra Packages for Enterprise Linux) repository on an Amazon Linux instance using the amazon-linux-extras tool.**

**🧩 Breakdown:**

1. **sudo  
   Runs the command as superuser (root), which is required because you're modifying system-level configurations.**
2. **amazon-linux-extras  
   This is a tool provided in Amazon Linux 2 (not the original Amazon Linux) that allows you to enable and manage extra software repositories or features that are not available by default.**
3. **enable epel  
   You're telling amazon-linux-extras to enable the EPEL repository (Extra Packages for Enterprise Linux).**
   * **EPEL is maintained by the Fedora Project and provides extra, high-quality packages not available in the base Amazon Linux repo.**

**✅ Why Use This?**

**By enabling EPEL, you get access to a much wider range of packages, such as:**

* **htop**
* **nginx**
* **fail2ban**
* **ansible**
* **Many Python/Ruby/NodeJS libraries and tools**

****

Figure 18 Enabling EPEL(Extra Packages for Enterprise Linux)

**A screenshot of a computer

AI-generated content may be incorrect.**

Figure 19 Install Epel Package Manager

**Command:** *sudo yum install -y epel-release*

**📖 Explanation:**

* **sudo**: Again, runs the command with root privileges.
* **yum**: This is the **Yellowdog Updater, Modified**, Amazon Linux’s package manager (like apt in Ubuntu).
* **install epel-release**: You're asking yum to install the **epel-release** package.

**🧠 What is epel-release?**

It's a **meta-package** that:

* Sets up the EPEL repository.
* Installs necessary GPG keys (used to verify the authenticity of packages).
* Adds repo configuration files to your system.

Basically, it **makes EPEL packages available for installation via yum**.

**Command:** *sudo yum clean metadata*

**EPEL = Extra Packages for Enterprise Linux. Required for certbot.**

### Install Certbot + Apache Plugin (Let’s Encrypt)

**Commands:** *sudo yum install -y certbot python-certbot-apache*

Installs Let's Encrypt's Certbot tool and the Apache plugin so it can automatically set up SSL certs for your virtual host.

**A screenshot of a computer

AI-generated content may be incorrect.**

Figure 20 Install Certbot and Apache Plugin to set up SSL.

**Installs Let's Encrypt's Certbot tool and the Apache plugin so it can automatically set up SSL certs for your virtual host.**

### Run Certbot to Configure SSL & HTTPS

**Commands:**

*sudo certbot --apache*

**Walks you through:**

* **Entering your email**
* **Agreeing to terms**
* **Selecting domain names (e.g., cloudbyms.click, www.cloudbyms.click)**
* **Auto-generates SSL certs**
* **Auto-creates Apache SSL config files**
* **Redirects HTTP → HTTPS**

*Follow the prompts — it’ll auto-configure HTTPS for you and even redirect HTTP to HTTPS.*

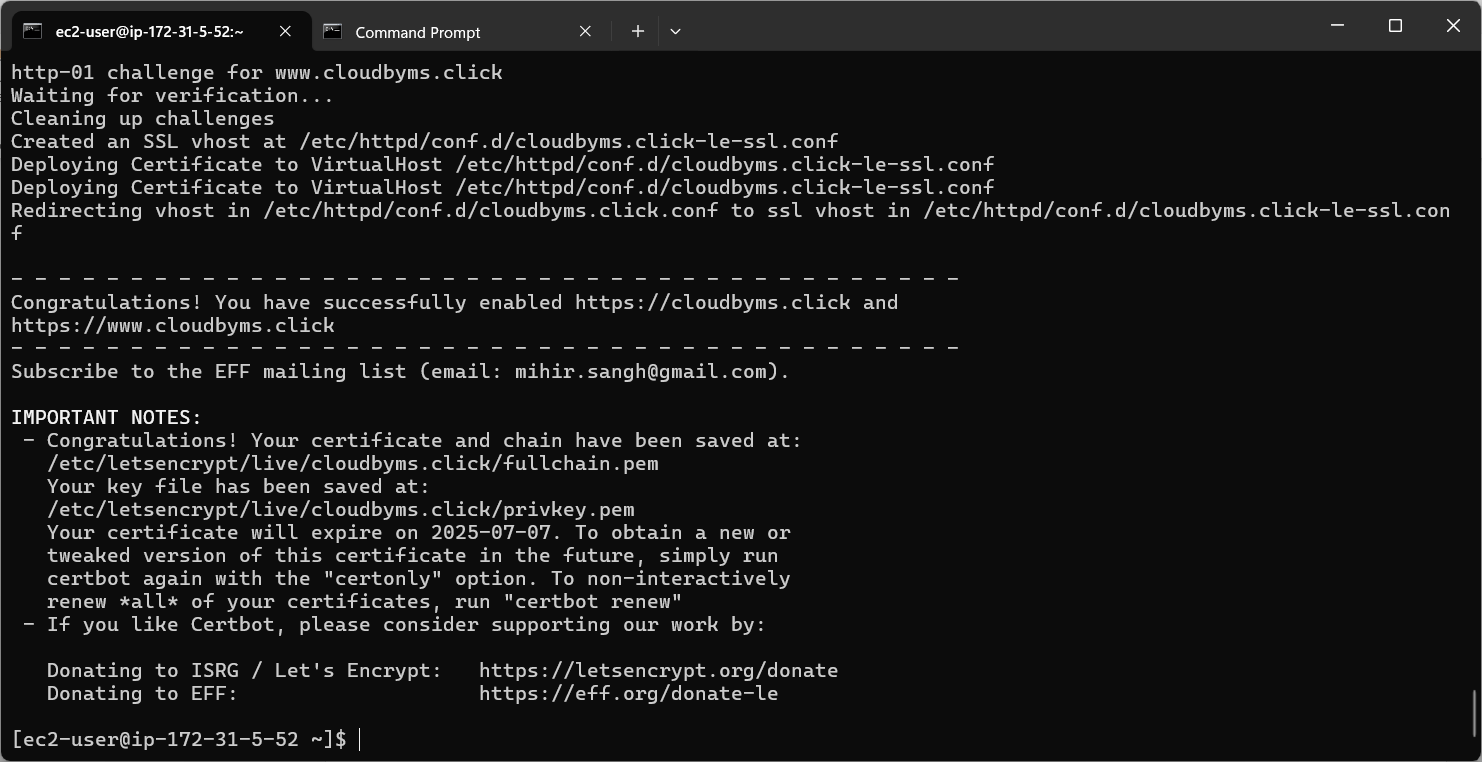
****

Figure 21 SSL Certificates installed

### Add Auto-Renewal via Cron

**Commands:**

*sudo crontab -e*

**🧠 sudo crontab -e Explained**

It’s a command to **edit the cron jobs for the root user**.

* cron is a time-based job scheduler in Linux. It runs scripts or commands automatically at set times.
* crontab stands for **"cron table"**, where scheduled jobs live.
* -e means **"edit"** the crontab file.
* sudo means you’re editing the crontab for the **root (admin) user**, not your current user.

**🤔 Why use sudo here?**

Because certbot renew often needs **root-level access** to:

* Read cert files from /etc/letsencrypt
* Restart the Apache server after renewal

So scheduling the command as **root** ensures it has the proper permissions.

**Inside the editor (press i to insert, then paste):**

0 2 \* \* \* /usr/bin/certbot renew –quiet

📌 **Breakdown**:

* 0 0 \* \* \* → Run at **midnight every day**
* /usr/bin/certbot → Path to the certbot binary
* --quiet → Runs silently unless there’s an error

**Saves with Esc, then :wq**

**This tells your server to renew the SSL certificate every day at 2:00 AM quietly (auto-renews before the 90-day expiry).**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

Figure 22 Auto-renewal via cron.

### Configure Route 53 DNS Records

**In the AWS Console → Route 53 → Hosted Zones → cloudbyms.click, create the following:**

| **Name** | **Type** | **Value (Target)** | **TTL** |
| --- | --- | --- | --- |
| **cloudbyms.click** | **A** | **<your EC2 Public IP>** | **300** |
| **www** | **A** | **<your EC2 Public IP>** | **300** |

**This points your domain and subdomain to your EC2 instance.**

**Open Required Ports in EC2 Security Group**

**Go to AWS Console → EC2 → Your Instance → Security Groups → Inbound Rules**

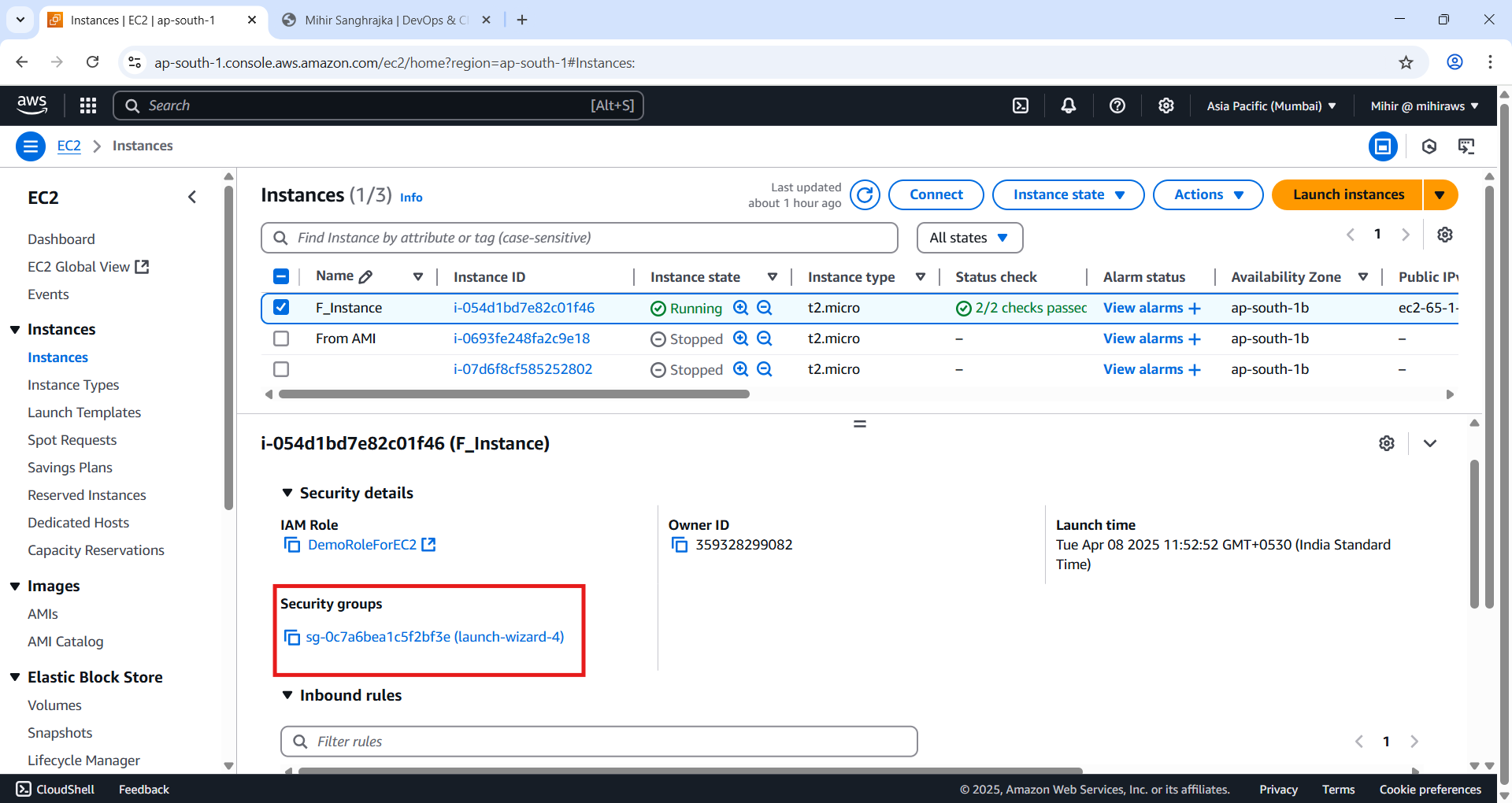
****

Figure 23 Select Security Groups, a new page will open go to find Inbound rules and hit ‘Edit inbound Rules’.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 24 Add 'HTTPS' as Type and 'IPv4 Anywhere' under Source.

**Add rules:**

| **Type** | **Protocol** | **Port** | **Source** |
| --- | --- | --- | --- |
| **HTTP** | **TCP** | **80** | **0.0.0.0/0** |
| **HTTPS** | **TCP** | **443** | **0.0.0.0/0** |

**Ensures your site is reachable on both HTTP and HTTPS.**

### (Optional) Check Public IP of Your EC2

*curl ifconfig.me*

**Confirms what public IP your domain should point to.**

### Verify Apache & SSL Status

*sudo systemctl status httpd*

**Should show active (running)**

### Test Your Site

**In browser:**

* **https://cloudbyms.click**
* **https://www.cloudbyms.click**

**Should show your Apache website with a secure padlock in the browser.**

### 🔐 Cert File Locations (Auto-managed by Certbot)

**Just FYI, your SSL certs are stored at:**

**/etc/letsencrypt/live/cloudbyms.click/**

**Includes:**

* **fullchain.pem — Full SSL cert chain**
* **privkey.pem — Your private key**

### Final Result

**You now have:**

* **Your domain live on EC2**
* **Apache serving your content**
* **HTTPS with Let's Encrypt SSL**
* **Auto-renewal configured**
* **Fully managed DNS via Route 53**

### Learnings

**📘 Learnings**

Through this project, I gained hands-on experience in deploying a static website using **Apache Web Server on an Amazon EC2 instance** running **Amazon Linux 2**. Here are the key concepts and skills I learned and applied:

**🔧 EC2 & Apache Web Server**

* Learned to **launch and connect to an EC2 instance** using SSH and manage the system using yum.
* Installed and configured the **Apache (httpd)** web server and verified it using the public IP.
* Understood the importance of using systemctl to **start, enable, and check service status** for Apache.

**📂 File Management on EC2**

* Used **SCP (Secure Copy Protocol)** to securely transfer files to the instance.
* Learned best practices like using the /tmp directory for temporary transfers to avoid permission issues.
* Gained understanding of **Linux file permissions** and ownership when moving files to /var/www/html.

**🌐 Route 53 & DNS**

* Configured **AWS Route 53 Hosted Zones** and added **A records** to map a custom domain (cloudbyms.click) and its subdomain (www.cloudbyms.click) to the EC2 instance.
* Understood how DNS propagation works and verified domain routing success.

**🌍 Virtual Hosting**

* Created and configured a **virtual host file** in Apache to support domain-based hosting using cloudbyms.click and www.cloudbyms.click.
* Understood the structure of Apache config files and how to define ServerName, DocumentRoot, and custom logs.

**🔐 SSL with Let's Encrypt (Certbot)**

* Enabled **EPEL repository** using amazon-linux-extras and installed **Certbot** with the Apache plugin.
* Ran certbot interactively to request and install an SSL certificate from **Let’s Encrypt**.
* Learned how Certbot automatically configures SSL in Apache and redirects HTTP to HTTPS for secure access.

**🔄 SSL Auto-Renewal via Cron**

* Configured a **cron job using sudo crontab -e** to automatically renew SSL certificates daily.
* Understood how Certbot stores certificates in /etc/letsencrypt/live/<domain>/ and handles renewal lifecycle.

**🔐 EC2 Security Groups**

* Modified **EC2 Security Group inbound rules** to allow traffic on ports **80 (HTTP)** and **443 (HTTPS)**, making the site accessible via secure connection globally.

**🧠 Additional Takeaways**

* Learned about the difference between **Amazon Linux 1 and Amazon Linux 2**, particularly in terms of package installation and repository management.
* Discovered how **EPEL (Extra Packages for Enterprise Linux)** enables installation of extended tools not available in the base Amazon Linux repo.
* Gained a strong understanding of how multiple AWS services—**EC2, Route 53, and IAM**—work together in a real-world hosting scenario.
* Learned how to **troubleshoot permission errors, service status issues, and DNS delays**, which are common in live deployments.
* Developed confidence in using terminal tools like nano, scp, curl, systemctl, and crontab.