ICT171 Assignment 2 – Cloud Web Server Deployment Documentation

Student Name: Nasrin Sadiqi

Student Number: 34898644

Global IP Address: 18.139.162.207

DNS Entry: livinginbetweenblog.space

GitHub Repository: github.com/N23897/Personal-blog

Live Website: https://livinginbetweenblog.space/

Table of Contents

Introduction

This report documents the process of deploying a modern, secure personal blog website to the cloud using AWS EC2 and Apache2, linked to a custom DNS entry with HTTPS enabled.
All commands, screenshots, and explanations are provided to ensure anyone can reproduce this server from scratch.
The deployment end-to-end server setup, custom scripting, domain management, and security best practices.
The project demonstrates skills in cloud infrastructure, Linux server management, web hosting, DNS configuration, and SSL/TLS implementation.

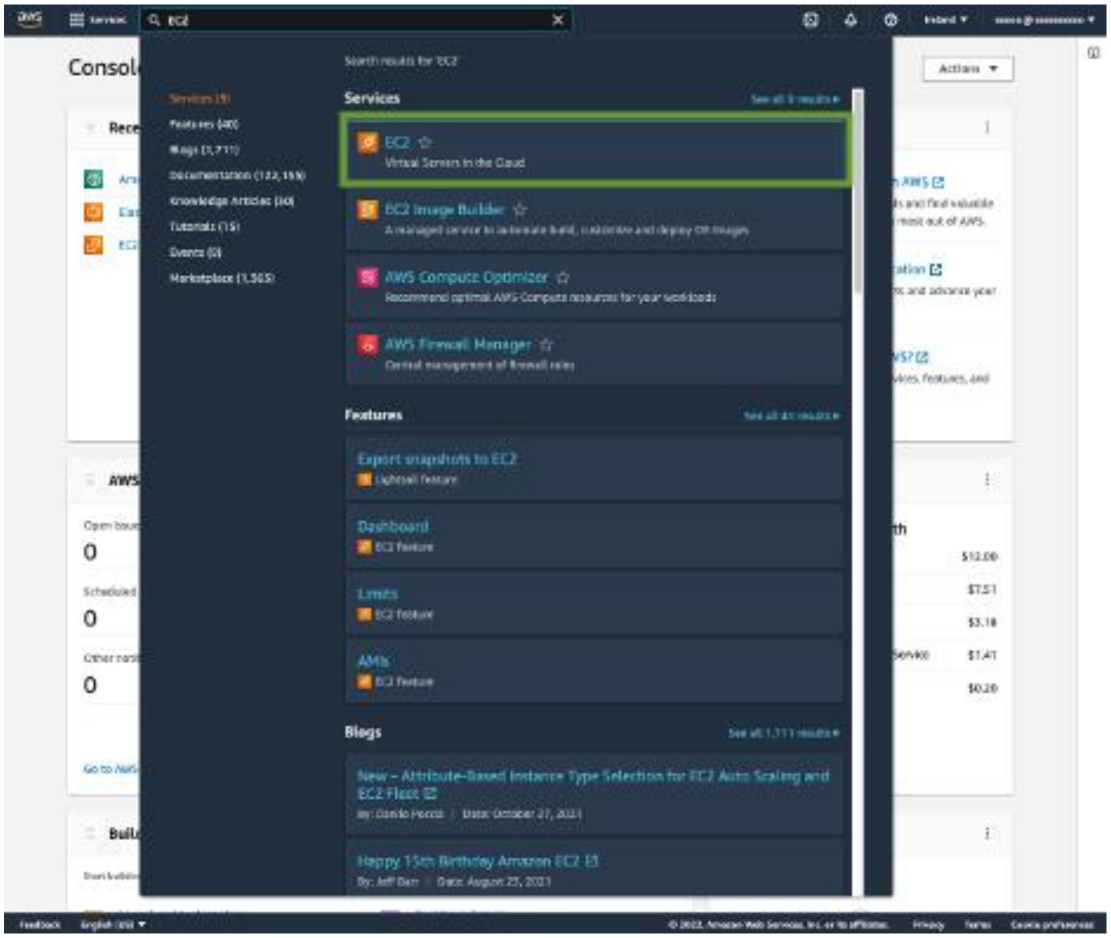
Step 1 – Launching an AWS EC2 Instance

1.1. Create an AWS Account and Log In

- Go to https://aws.amazon.com/
- Click Create an AWS Account if you don't have one.
- Enter your details and payment method (AWS offers a Free Tier for new accounts).
- If you already have an account, just sign in.

1.2. Access the EC2 Dashboard

- After logging in, find the Search bar at the top of the AWS console.
- Type EC2 and select EC2 from the dropdown.



1.3. Launch a New EC2 Instance

a) Click Launch Instance (top right button).

---

b) Fill in Instance Details

Instance Name:

- Type a name (e.g. personal-blog-server).

Amazon Machine Image (AMI):

- Under "Application and OS Images (Amazon Machine Image)",
    - select Ubuntu Server 22.04 LTS (HVM), SSD Volume Type – 64-bit x86 (should say "Free tier eligible")

Instance Type:

- Choose t2.micro (Free Tier eligible).

Key Pair (login):

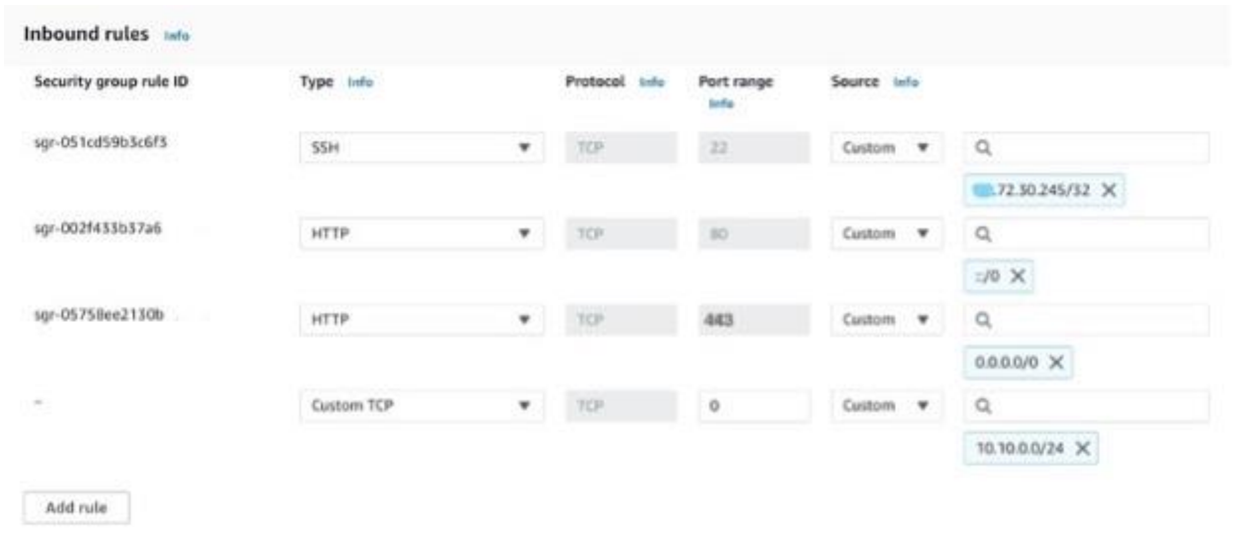
- In the "Key pair (login)" section, choose Create new key pair.
    - Name: myblog-key
    - Key pair type: RSA
    - Private key file format: PEM
- Click Create key pair.
- Download the .pem file and keep it somewhere safe (e.g. ~/Downloads/myblog-key.pem).

---

c) Configure Network Settings (Security Group)

- Under "Network settings," click Edit.
- Add (or make sure you have) the following inbound rules:

Type	**Protocol**	**Port Range**	**Source**	**Description**
SSH	TCP	22	0.0.0.0/0	SSH from anywhere
HTTP	TCP	80	0.0.0.0/0	Web access
HTTPS	TCP	443	0.0.0.0/0	Secure web access



Inbound rules Info

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info	
sgr-051cd59b3c6f3	SSH	TCP	22	Custom	[redacted].72.30.245/32
sgr-002f433b37a6	HTTP	TCP	80	Custom	::/0
sgr-05758ee2130b	HTTP	TCP	443	Custom	0.0.0.0/0
–	Custom TCP	TCP	0	Custom	10.10.0.0/24

Add rule

---

d) Launch the Instance

- Scroll down and click Launch Instance (orange button).

1.4. Get Your Public IPv4 Address

- In the EC2 Dashboard, click Instances in the left menu.
- Find your new instance in the list.
- Copy the Public IPv4 address (e.g., 3.9.142.21).

Step 2 – Connecting to Your EC2 Instance via SSH

2.1. Prerequisites

- Your EC2 instance is running (see Step 1).
- You have downloaded your private key file (e.g., myblog-key.pem).
- You have your Public IPv4 address from AWS EC2.

2.2. Option A: Connect Using AWS EC2 Instance Connect (Browser)

This is the easiest method!

1. Go to the AWS EC2 Console.
2. Click Instances on the left sidebar.
3. Select your instance by ticking the checkbox.
4. Click Connect at the top.
5. Make sure the EC2 Instance Connect tab is selected.
6. Click the orange Connect button.

A browser terminal will open, and you will be logged in as the ubuntu user.

2.3. Option B: Connect Using Your Computer’s Terminal (with SSH)

a) Set permissions on your key file (first time only):

In your terminal (Mac/Linux/WSL/Git Bash), run:

```
chmod 400 ~/Downloads/myblog-key.pem
```

(Make sure the path matches where you saved your key file.)

---

b) Connect to your instance:

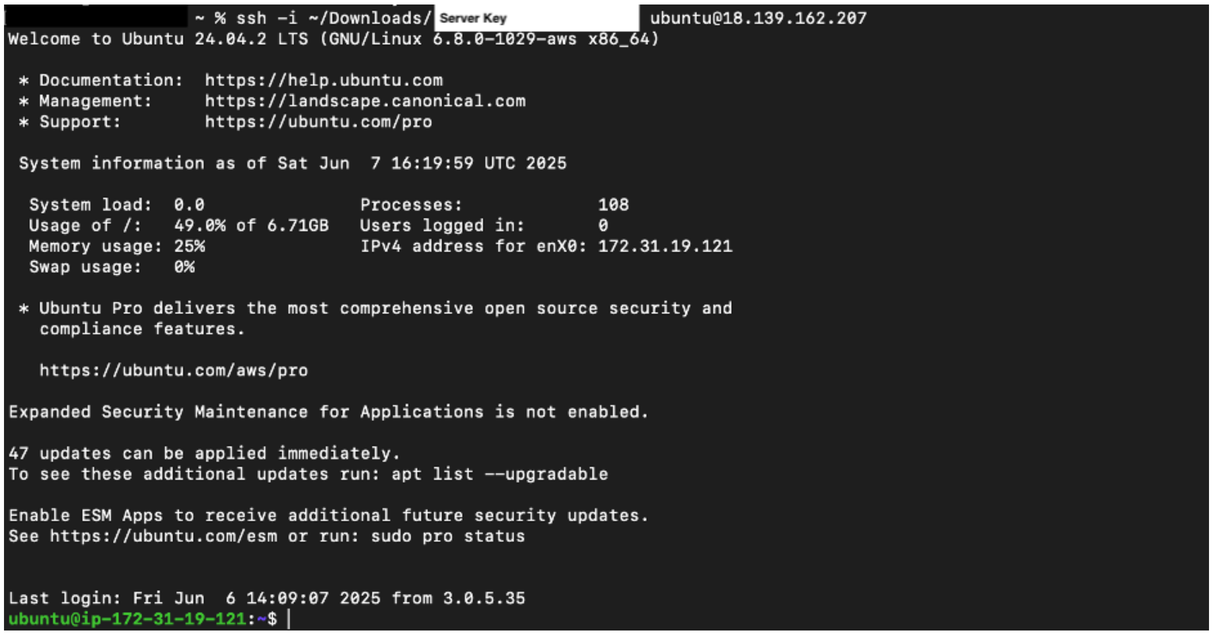
Use the SSH command:

```
ssh -i ~/Downloads/myblog-key.pem ubuntu@<YOUR_PUBLIC_IP>
```

Example:

```
ssh -i ~/Downloads/myblog-key.pem ubuntu@3.9.142.21
```

- If it's your first time connecting, type yes when prompted to trust the host.



```
[redacted] ~ % ssh -i ~/Downloads/Server Key ubuntu@18.139.162.207
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-1029-aws x86_64)

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

 System information as of Sat Jun  7 16:19:59 UTC 2025

  System load:  0.0               Processes:             108
  Usage of /:   49.0% of 6.71GB   Users logged in:       0
  Memory usage: 25%               IPv4 address for enX0: 172.31.19.121
  Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

47 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

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


Last login: Fri Jun  6 14:09:07 2025 from 3.0.5.35
ubuntu@ip-172-31-19-121:~$ |
```

2.4. Troubleshooting

- **Permission denied (publickey):**
    - Double-check you used chmod 400 on your PEM file.
    - Make sure you are using the correct username (ubuntu) and your current Public IP.
- **Connection timed out:**
    - Make sure your Security Group allows SSH (port 22) from anywhere.

2.5. Success

You are now logged in to your cloud server as the ubuntu user!
You can install software and manage your web server from this terminal.

Step 3 – Installing Apache2 Web Server

3.1. Update Your Server

Before installing any new software, always update your server to get the latest security patches and package lists.

In your SSH terminal, run:

```
sudo apt update

sudo apt upgrade -y
```

- sudo apt update refreshes the list of available software.
- sudo apt upgrade -y installs the newest versions of all packages.

3.2. Install Apache2

Install Apache2 using:

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

- This command downloads and installs the Apache2 web server software.
- The process should take less than a minute.

3.3. Check Apache2 Status

Check that Apache2 is running:

```
sudo systemctl status apache2
```

- You should see Active: active (running) in green in the output.
- Press q to exit the status window.

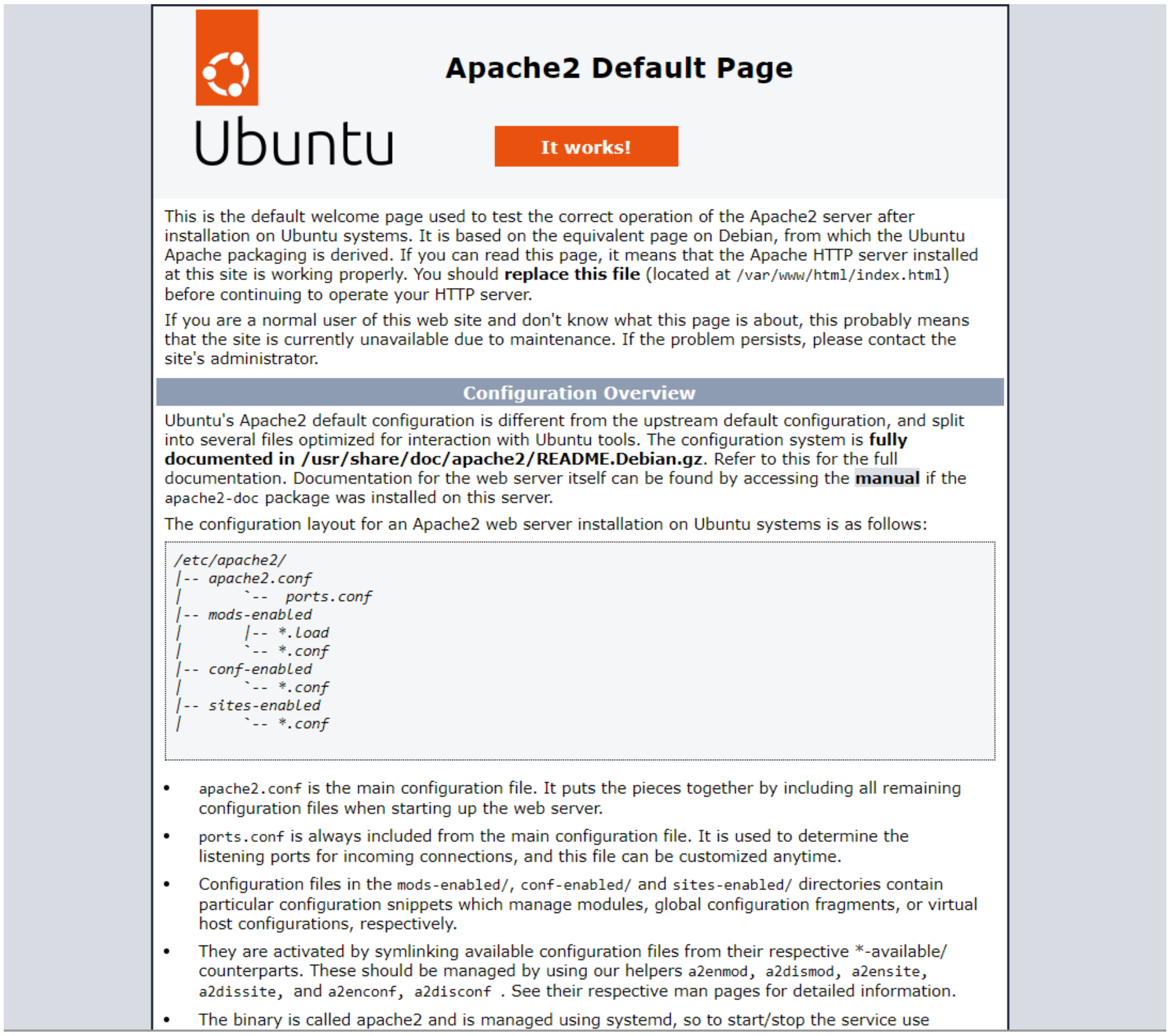
3.4. Test Apache2 in Your Browser

1. Find your instance's Public IPv4 address (from Step 1).
2. In your browser, go to:

```
http://<YOUR_PUBLIC_IP>
```

(Example: http://3.9.142.21)

- You should see the default **Apache2 Ubuntu Default Page** with the message:
  "It works!"



This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective *-available/ counterparts. These should be managed by using our helpers `a2enmod`, `a2dismod`, `a2ensite`, `a2dissite`, and `a2enconf`, `a2disconf` . See their respective man pages for detailed information.
- The binary is called apache2 and is managed using systemd, so to start/stop the service use

---

3.5. Where to Place Website Files

By default, Apache2 serves files from:

```
/var/www/html/
```

- The default homepage is /var/www/html/index.html.
- You will upload your website files here in the next step.

Step 4 – Uploading Your Website Files

4.1. Prepare Your Website Files

- On your computer, gather all files for your site:
    - index.html
    - custom.css
    - custom.js
    - /images folder (if you have images)
    - Any other supporting files

4.2. Use SCP to Upload Files to Your Server

SCP ("secure copy") allows you to securely transfer files from your computer to your EC2 instance using your PEM key.

a) Change to your project directory (where your website files are):

```
cd /path/to/your/website/files
```

(Replace with your actual directory)

---

b) Upload your main files (replace with your actual PEM and IP):

```
scp -i ~/Downloads/myblog-key.pem index.html ubuntu@<YOUR_PUBLIC_IP>:/var/www/html/

scp -i ~/Downloads/myblog-key.pem custom.css ubuntu@<YOUR_PUBLIC_IP>:/var/www/html/

scp -i ~/Downloads/myblog-key.pem custom.js ubuntu@<YOUR_PUBLIC_IP>:/var/www/html/
```

c) Upload your images folder (if you have one):

```
scp -i ~/Downloads/myblog-key.pem -r images ubuntu@<YOUR_PUBLIC_IP>:/var/www/html/
```

- Overwrite the default index.html if prompted.

4.3. Test Your Website

- Open your browser and go to:

```
http://<YOUR_PUBLIC_IP>
```

- You should now see your custom homepage (not the default Apache “It works!” page).

4.4. Troubleshooting

- **Permission Denied?**
  - Make sure your .pem file permissions are set correctly:

```
chmod 400 ~/Downloads/myblog-key.pem
```

- **Page Not Updating?**
  - Hard refresh your browser (Ctrl+Shift+R or Cmd+Shift+R).
  - Double-check you uploaded to /var/www/html/.
- **Missing Images or Styles?**
  - Make sure you uploaded all required files and folders.

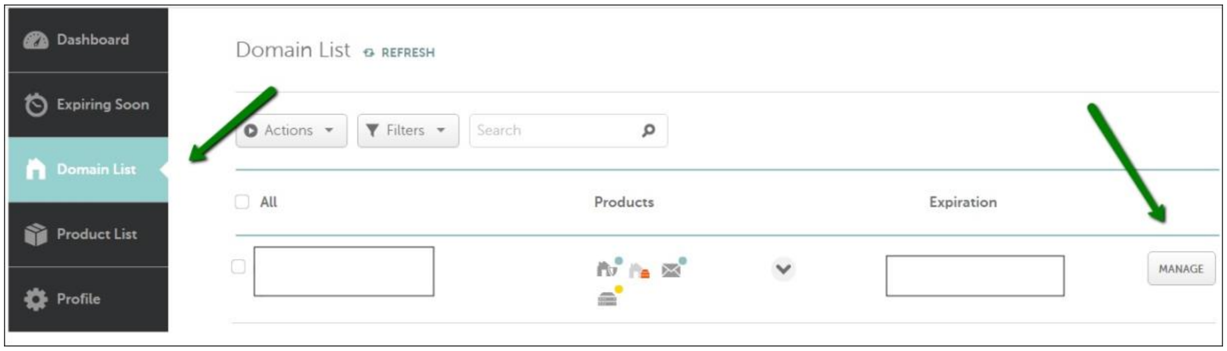
Step 5 – Setting Up Your Domain Name (DNS)

5.1. Purchase or Use a Domain Name

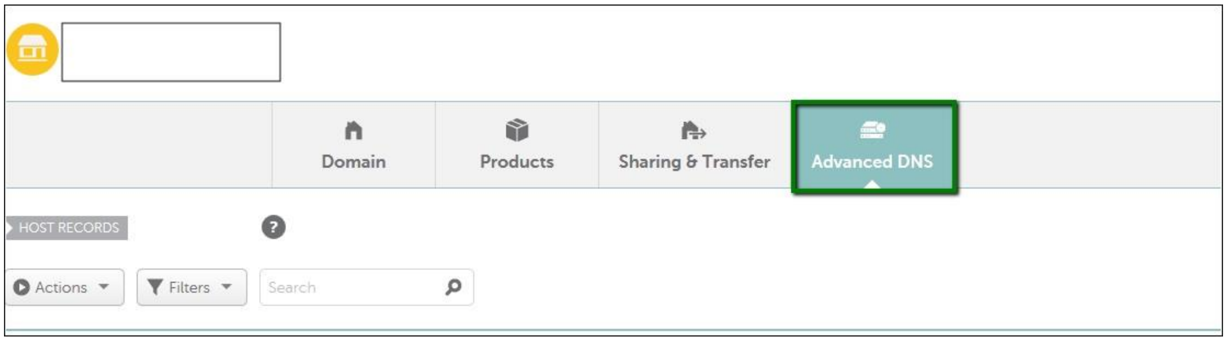
- If you don't already have a domain, purchase one from a registrar such as Namecheap or GoDaddy.
- After purchase, your domain will appear in your registrar's dashboard.

5.2. Access Your Domain's DNS Settings

- Log in to your domain registrar.
- Go to the Domain List or My Domains section.



- Click Manage next to your domain.
- Find and open the Advanced DNS or DNS Settings tab.

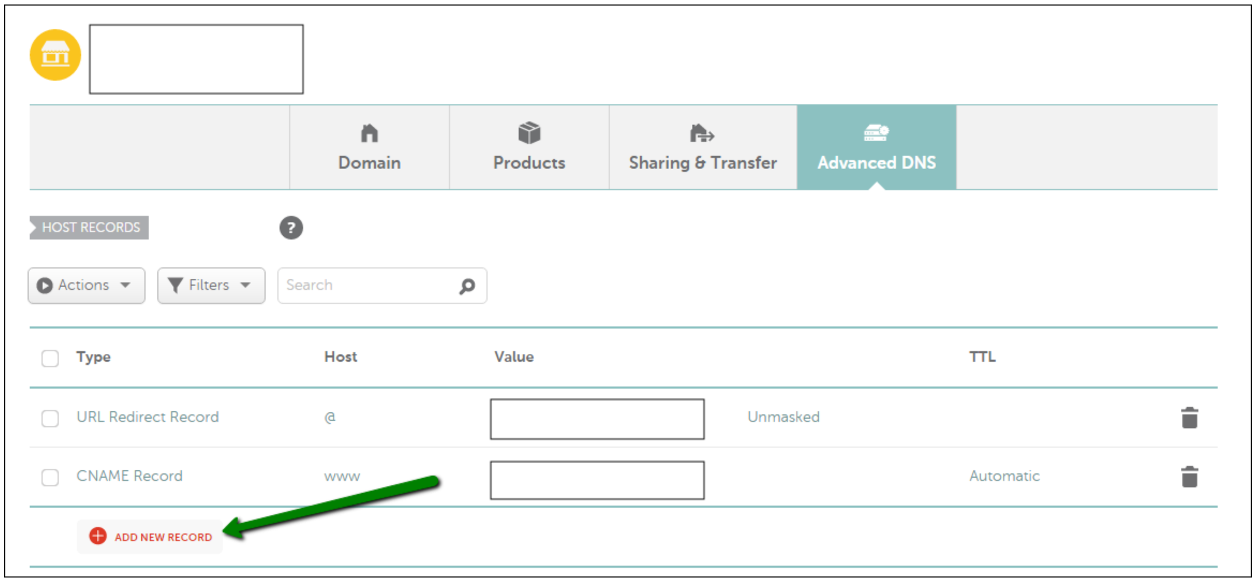


5.3. Add or Edit the A Record

- In the DNS settings, find the A Record for your root domain (@).
- Edit the A Record (or add a new one if needed):

Host	Type	Value (Destination)	TTL
@	A	<YOUR_PUBLIC_IP>	Automatic

- *(Replace <YOUR_PUBLIC_IP> with your EC2 instance's public IPv4 address)*



Type	Host	Value		TTL
URL Redirect Record	@		Unmasked	
CNAME Record	www			Automatic

- Save or confirm your changes.

5.4. (Optional) Add an A Record for "www"

- To support both yourdomain.com and www.yourdomain.com, add:

Host	Type	Value (Destination)	TTL
www	A	<YOUR_PUBLIC_IP>	Automatic

- Save your changes.

5.5. Wait for DNS Propagation

- DNS changes usually take 5–30 minutes but can be up to 1 hour for new domains.

5.6. Test Your Domain

- In your browser, visit:

```
http://yourdomain.com
```

- You should see your website (the same as when you used your public IP).

5.7. Troubleshooting

- **Domain does not load your site.**
  - Double-check your A Record points to your EC2 public IP.
  - Wait for DNS propagation.
  - Test with whatsmydns.net to check worldwide.
- **Site loads at IP but not at domain?**
  - Try a hard refresh or check for typos in your domain entry.

Step 6 – Securing Your Website with HTTPS (SSL)

6.1. What is HTTPS and Why is it Important?

HTTPS (Hypertext Transfer Protocol Secure) protects the information sent between your website and its visitors.
It's required for privacy, trust, and a padlock icon in the browser, and Google also prefers secure sites.
You'll use Let's Encrypt (a free certificate authority) and Certbot **(**the setup tool) to make your site secure.

6.2. Prerequisites

- Your website is running on your EC2 Ubuntu instance.
- Apache2 is installed and serving your site.
- Your DNS A record is set up and points to your EC2's public IP.
- You can access your site using your domain name (e.g., http://yourdomain.com).

6.3. Install Snap and Core Utilities (if needed)

Certbot is easiest to install and keep updated with snap**.**

First, make sure snap is installed and up to date:

```
sudo apt update
sudo apt install snapd -y
sudo snap install core; sudo snap refresh core
```

6.4. Remove Any Old Certbot Packages (Optional, but recommended)

If you've tried Certbot before, remove older versions to avoid conflicts:

```
sudo apt remove certbot
```

6.5. Install Certbot with Snap

Install Certbot using snap:

```
sudo snap install --classic certbot
```

Link Certbot so it runs as a command

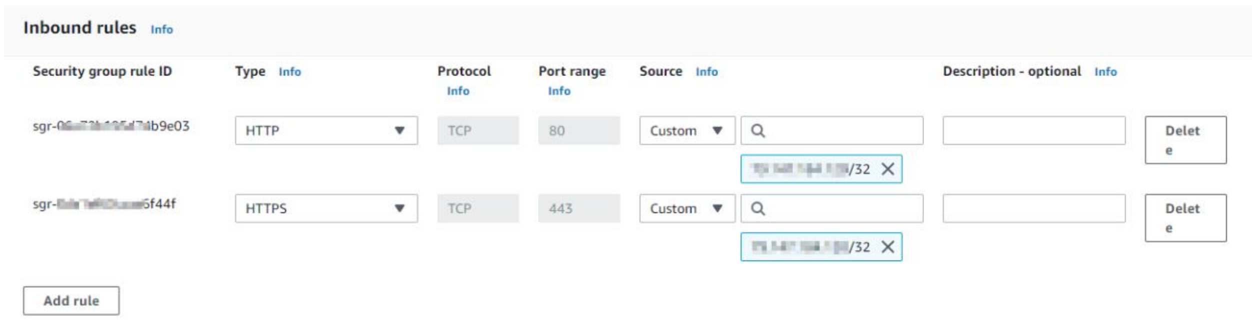
```
sudo ln -s /snap/bin/certbot /usr/bin/certbot
```

6.6. Ensure HTTPS (Port 443) Is Open in AWS

- Go to your AWS EC2 dashboard, select your instance's Security Group, and check Inbound Rules.
- There should be a rule for:

Type	Protocol	Port	Source
HTTPS	TCP	443	0.0.0.0/0

- If missing, click Edit inbound rules and add it.



Inbound rules Info

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info		Description - optional Info	
sgr-0[redacted]b9e03	HTTP	TCP	80	Custom	[redacted]/32		Delete
sgr-0[redacted]5f44f	HTTPS	TCP	443	Custom	[redacted]/32		Delete

Add rule

6.7. Run Certbot to Get and Install Your SSL Certificate

Use Certbot's Apache plugin to automatically configure your server:

```
sudo certbot --apache
```

What happens next:

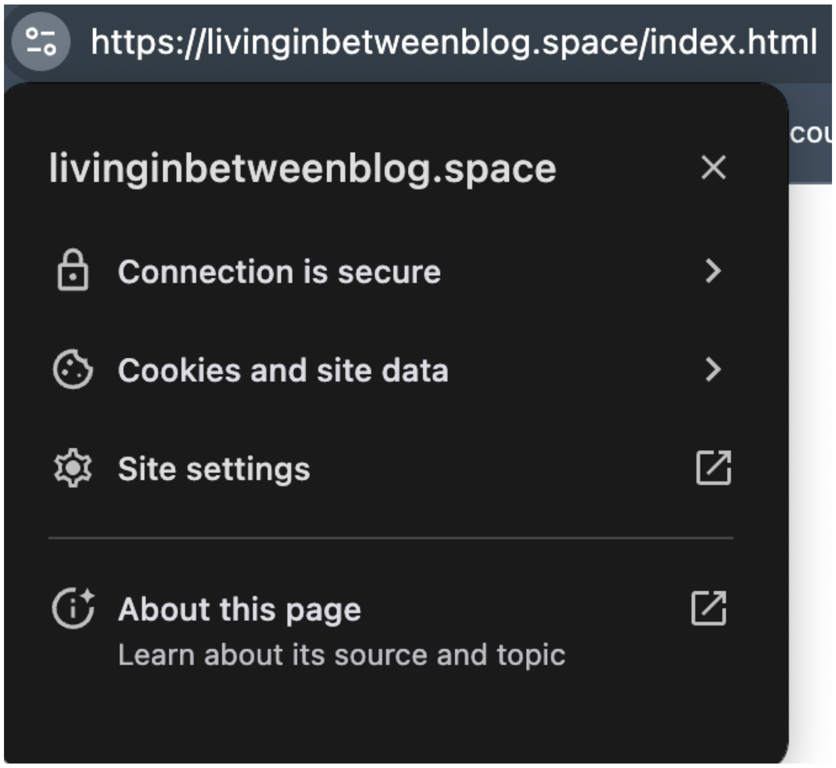
- Certbot will prompt you for your email (for important expiry notices).
- You'll be asked to agree to the terms of service.
- Certbot will detect your domain(s). If not, enter them when prompted (e.g., *yourdomain.com* and *www.yourdomain.com*).
- Certbot will run a challenge to verify you own the domain.
- Certbot will update your Apache settings automatically.
- You'll be asked if you want to redirect all traffic to HTTPS (choose "redirect" for best security).

6.8. Test Your HTTPS Website

Visit your site with https:// in front of your domain (e.g., *https://yourdomain.com*).

- You should see a padlock icon in the browser address bar.
- The site is now secure!

If you get any certificate warning, check your DNS and wait up to 15 minutes for DNS to fully update.



https://livinginbetweenblog.space/index.html

livinginbetweenblog.space

Connection is secure

Cookies and site data

Site settings

About this page

Learn about its source and topic

6.9. How to Renew SSL Certificates (Let’s Encrypt)

Let’s Encrypt certificates last for 90 days. Certbot installs a timer to auto-renew for you.

You can check this timer is active by running:

```
sudo systemctl status snap.certbot.renew.service
sudo systemctl status snap.certbot.renew.timer
```

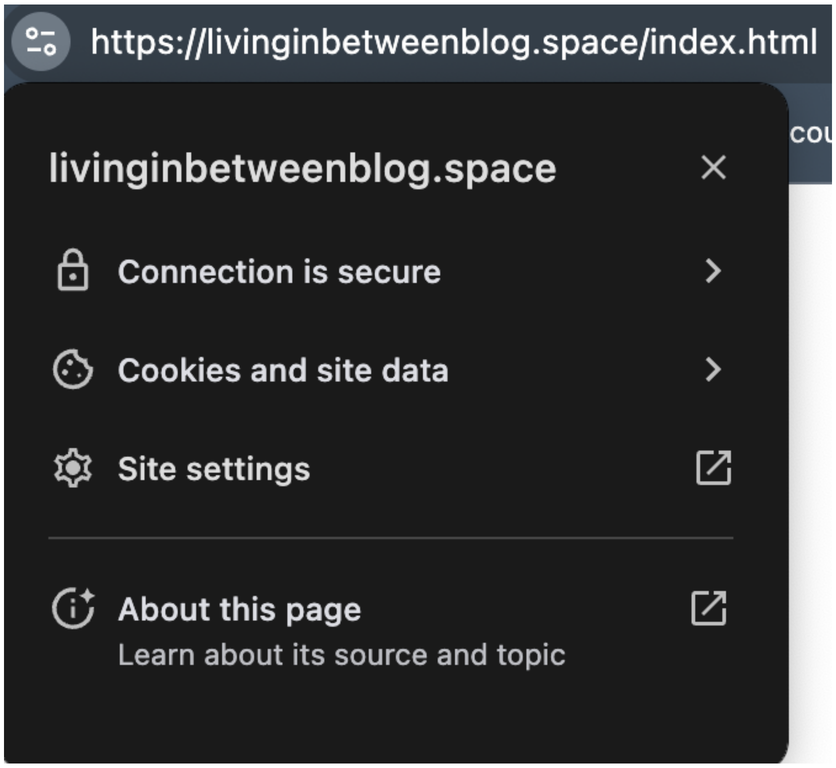
To manually renew at any time:

```
sudo certbot renew
```

6.10. Troubleshooting Tips

- "Domain not found" or challenge fails:
  Make sure your DNS A record is correct and your website loads at your domain.
- Still "Not Secure":
  Try opening your site in a private/incognito window.
  Clear your browser cache.
  Make sure you selected the redirect option during Certbot setup.
- Port 443 blocked:
  Double-check AWS Security Group inbound rules include port 443.
- Renewal errors:
  Make sure your EC2 is always running, and DNS is still correct.

- You should see your homepage with a padlock icon in the address bar.


https://livinginbetweenblog.space/index.html

Step 7 – Final Checks, Script Documentation, and References

7.1. Final Checks

A. Verify Your Website is Live and Secure

- Visit your website using your custom domain and HTTPS:

```
https://yourdomain.com
```

(Example: https://livinginbetweenblog.space)

- Confirm the following:
    - Your homepage loads (not Apache “It works!” page)
    - Padlock icon is present (connection is secure)
    - All images, styles, and scripts work
    - Every link, filter, button, and interactive feature functions

---

B. Check on Multiple Devices and Networks

- Test your site on your phone and other devices.
- Check using different networks (home WIFI, mobile data) to ensure global DNS/SSL works.

C. Check DNS Propagation

- Use https://www.whatsmydns.net/
- Enter your domain and confirm the A Record points to your EC2 public IP in all regions.

D. Test SSL/HTTPS Grade

- Use https://www.ssllabs.com/ssltest/
- Enter your domain and verify you score an “A” or “A+”.

7.2. Script Documentation and Output

What the Script Does

Your custom JavaScript file (custom.js) enables dynamic and interactive features on your website, including:

•Mood toggle button (random pastel background)

•Countdown timer to next trip

•Animated “flower shower” (frangipani falling effect)

•Typing welcome message

•Random fun fact generator

•Back-to-top button

•Modal/lightbox for gallery images

•Filter buttons for photo gallery

All these features are visible live on your website.

Script Purpose

The interactive scripts in this project are not just decorative—they are central to the function, engagement, and user experience of the website.
They make the site feel modern, personalized, and memorable. By blending interactivity and animation with storytelling and navigation, these scripts were added because they help:

- Encourage user engagement (e.g., mood toggle and fun facts invite exploration)
- Guide navigation (e.g., “Back to Top” button improves usability)
- Deliver timely information (e.g., countdown timer for events/trips)
- Create a unique brand feel (e.g., flower shower and polaroid carousel visually reinforce the travel/blog theme)
- Support accessibility and fun (e.g., typing animation and modal gallery keep users interested and make the site more enjoyable)

Script Examples

Example 1: Mood Toggle Button

```
// Returns a random pastel HSL color string for the background
function getRandomPastel() {
  const h = Math.floor(Math.random() * 360);
  const s = 70 + Math.random() * 10;
  const l = 80 + Math.random() * 10;
  return `hsl(${h},${s}%,${l}%)`;
}
// Changes the website background when the button is clicked
function toggleMood() {
  document.body.style.background = getRandomPastel();
}
document.getElementById('toggle-mood-btn').onclick = toggleMood;
```

What this code does:

This feature allows users to change the entire website's background to a new, randomly generated pastel colour every time they click a button.

This makes the site feel fresh and dynamic, adds an element of playful discovery, and encourages repeat interactions.

Example 2: Animated Countdown Timer

```
const endDate = new Date("2025-09-20T00:00:00"); // Target event date

function pad(n) { return n < 10 ? '0' + n : n; } // Pads numbers to two digits

function updateUnit(id, value) {
  const el = document.getElementById(id);
  if (el.textContent !== value) {
    el.classList.remove('flip');
    void el.offsetWidth; // Restarts flip animation
    el.textContent = value;
    el.classList.add('flip');
  }
}

function updateCountdown() {
  const now = new Date();
  const diff = endDate - now;
  if (diff > 0) {
    updateUnit("days", pad(Math.floor(diff / (1000 * 60 * 60 * 24))));
    updateUnit("hours", pad(Math.floor((diff / (1000 * 60 * 60)) % 24)));
    updateUnit("minutes", pad(Math.floor((diff / (1000 * 60)) % 60)));
    updateUnit("seconds", pad(Math.floor((diff / 1000) % 60)));
  } else {
    document.getElementById("countdown").textContent = "🎉 The trip has started!";
  }
}
setInterval(updateCountdown, 1000);
updateCountdown();
```

What this code does:

This animated flip-clock countdown shows the time remaining until a specific date (like your next trip or major blog milestone).

It updates every second and uses animations to make each digit flip, grabbing attention and building anticipation.

Live Output:

You can see this in action at https://livinginbetweenblog.space by pressing the "Press me" button at the top of the homepage.

7.3. Troubleshooting

- **Site not loading?**
  - Ensure EC2 is running, and security groups allow HTTP/HTTPS.
  - Double-check DNS A Record is correct.
  - Wait up to 60 minutes for DNS propagation.
- **No padlock or browser says, "Not Secure"?**
  - Visit with https://
  - Check Certbot output; renew if needed.
- **Styles/scripts/images missing?**
  - Re-upload all files to /var/www/html/.

Conclusion

This documentation demonstrates the full development of deploying a personal website on a cloud server using best-practice methods.
From launching an AWS EC2 instance to configuring Apache2, uploading content, managing DNS, and enabling SSL security, every phase is thoroughly explained and supported by screenshots and code samples.
Custom JavaScript brings the site to life with interactive features, while clear script documentation and live links enable independent verification of all work.

By following these instructions, anyone can successfully rebuild this server, ensuring both accessibility and security.

References

Certbot. (n.d.). *Certbot instructions for Apache on Ubuntu (snap)*. Electronic Frontier Foundation. https://certbot.eff.org/instructions?ws=apache&os=snap

GitHub. (n.d.). *Obtaining a digital certificate from Let's Encrypt (Murdoch University Guide)*. https://github.com/SCH-IT-MurdochUni/NetworkingLabs/blob/main/Server_Environments_and_Architectures/obtaining_a_digital_certificate_from_lets_encrypt.md

Let’s Encrypt. (n.d.). *Getting started*. https://letsencrypt.org/getting-started/

Namecheap. (n.d.). *How to manage DNS records*. https://www.namecheap.com/support/knowledgebase/article.aspx/319/2237/how-can-i-set-up-an-a-record-for-my-domain/

Ubuntu. (n.d.). *Official Ubuntu Documentation*. https://ubuntu.com/server/docs

Amazon Web Services. (n.d.). *Amazon EC2 user guide for Linux instances*. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html