ICT171 Assignment 2: Cloud Server Project & Video Explainer

Name: Anuj Mahato
Student ID: 35450857
Domain Name: https://anuj.fun
Server IP Address: 170.64.253.234
License Chosen: MIT License
Github Link
Video Link

Table of Contents

1. Project Scenario

This project involves deploying a cloud-based website specifically designed to provide free stock images for users to download and use. The website is aimed at students, bloggers, and anyone who needs high-quality images without worrying about copyright or licensing fees.. At first, I honestly had no idea how to even put a site on the cloud, but I learned by doing. The main thing was to show my ICT171 skills, so I set up a Linux server on DigitalOcean (never used it before), did all the DNS stuff to point my own domain there, and managed to get HTTPS running using Certbot. I built the whole website from scratch so it actually works for anyone on the internet.

While doing all this, I learned a lot about server admin, using Linux commands (sometimes messing it up), how DNS and SSL/TLS works, and how to write everything up proper for uni. Also figured out how to keep all my files and changes on GitHub which is good for backup. Now the website isn't just for this unit, it can actually help someone and it's there for other ICT171 students who wanna do something like this in future.

2. Server Setup

Provider: DigitalOcean (IaaS)
Operating System: Ubuntu 22.04 LTS

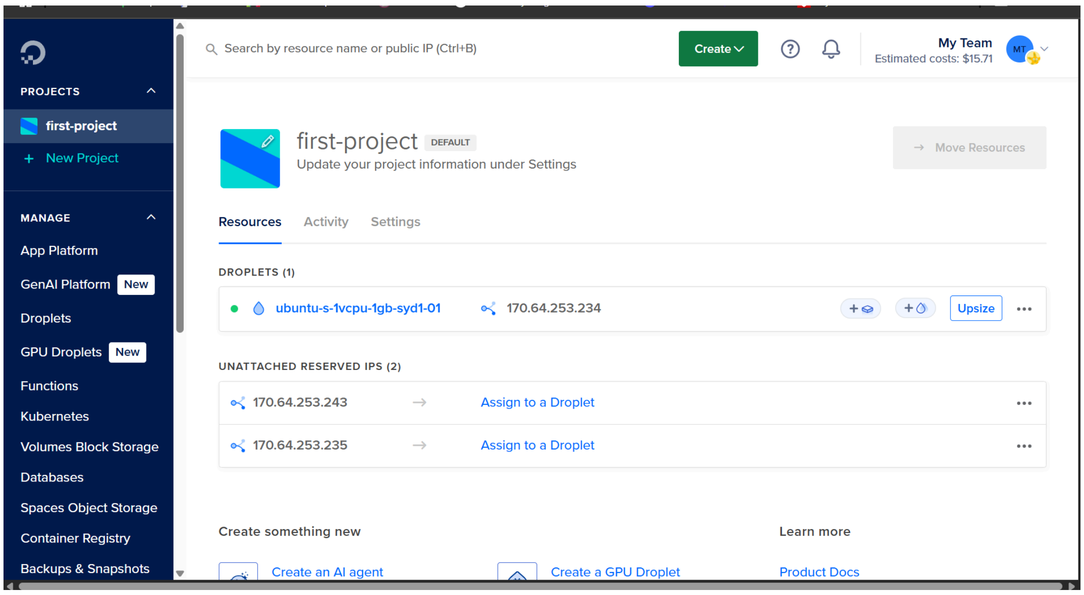
Steps Taken:

- First, I signed up on DigitalOcean and created what they call a "droplet", which is just their name for a mini server.
- I set it up so I had root access, so I could basically change whatever I want on it.
- DigitalOcean gave me a public IP (mine's 170.64.253.234), so my server's out there on the internet for anyone to find.

- After that, I registered my own domain – anuj.fun – and had to play around with the DNS settings until it pointed to my server's IP. Took a bit of trial and error honestly.

- Once all that was sorted, it meant I was ready to start installing stuff and make the site live for anyone to check out.

Screenshot: DigitalOcean droplet dashboard and DNS management page, confirming the correct server setup and domain linkage.



first-project DEFAULT
Update your project information under Settings

DROPLETS (1)

ubuntu-s-1vcpu-1gb-syd1-01	170.64.253.234

UNATTACHED RESERVED IPS (2)

170.64.253.243	Assign to a Droplet
170.64.253.235	Assign to a Droplet

3. SSH Connection

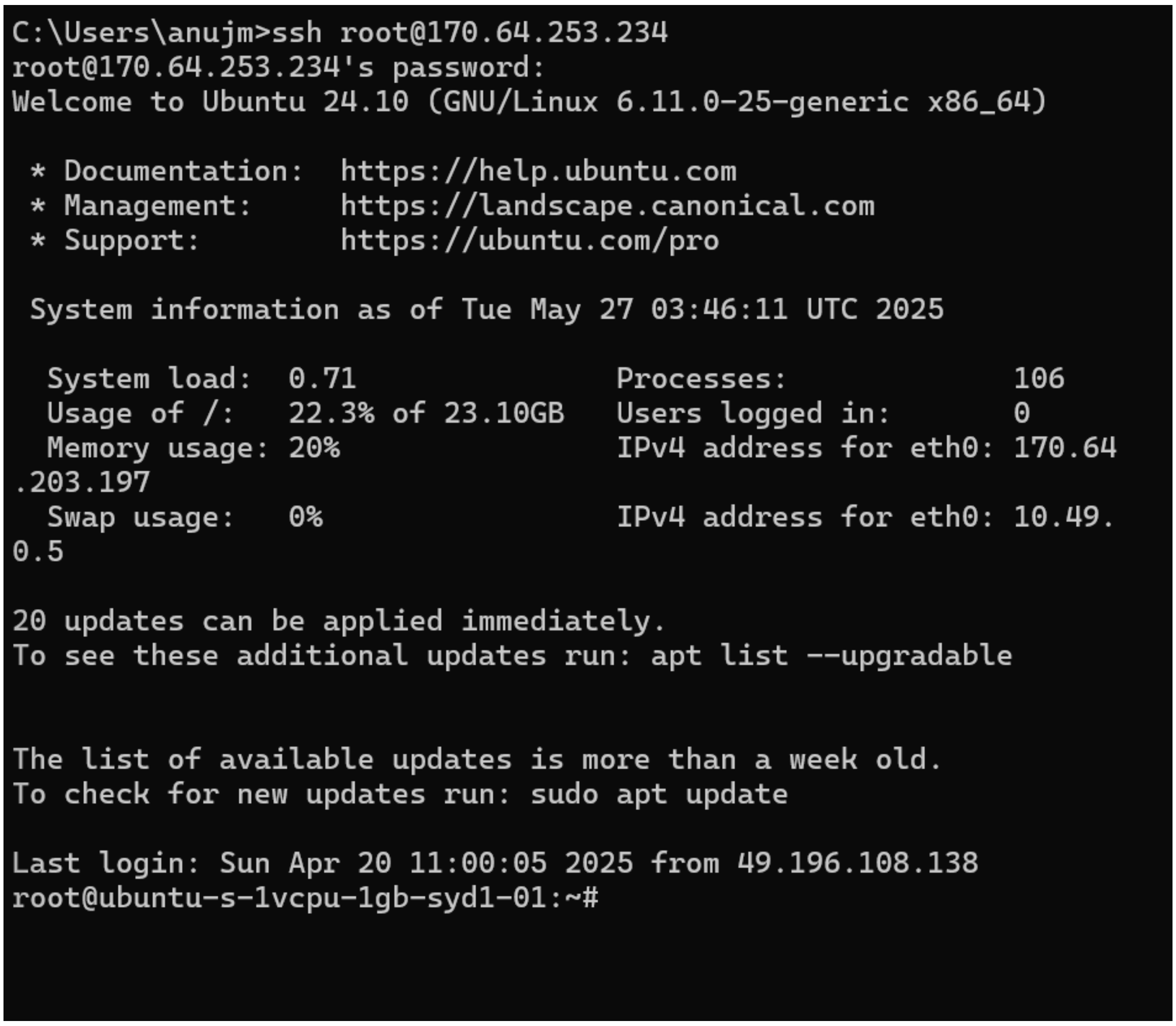
Secure Shell (SSH) is how I connected to my server from my own computer, and it's pretty much what everyone uses because it's safe. I used SSH to log into my DigitalOcean server straight from my laptop's terminal. Almost everything I did—like setting up, installing software, and making changes—was done through SSH. Without it, I really couldn't have finished the project.

- The main command I used for connecting was: `ssh root@170.64.253.234`

- SSH was used for all the setup, updates, and fixing stuff whenever I messed up.

- All my server management happened in the terminal using SSH.
-

Screenshot: Terminal window showing a successful SSH connection to the cloud server as the root user.



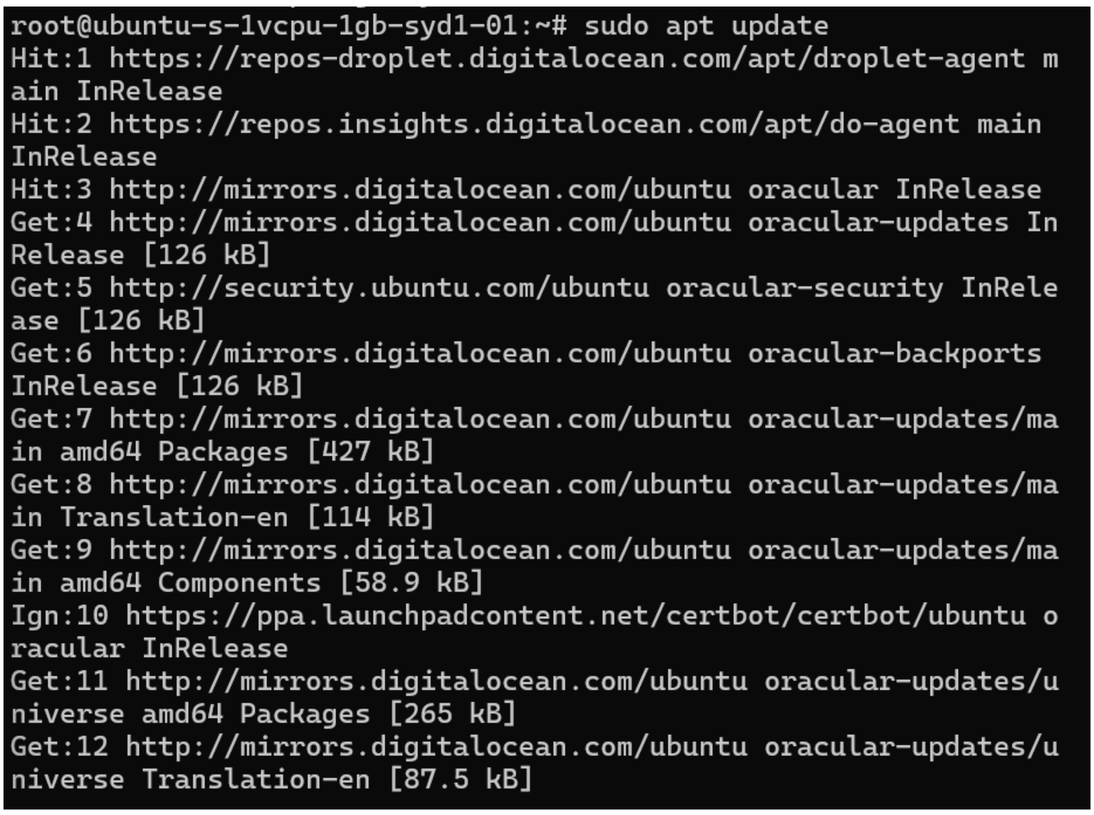
```
C:\Users\anujm>ssh root@170.64.253.234
root@170.64.253.234's password:
Welcome to Ubuntu 24.10 (GNU/Linux 6.11.0-25-generic x86_64)

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

 System information as of Tue May 27 03:46:11 UTC 2025

  System load:  0.71               Processes:             106
  Usage of /:   22.3% of 23.10GB   Users logged in:       0
  Memory usage: 20%                IPv4 address for eth0: 170.64
.203.197
  Swap usage:   0%                 IPv4 address for eth0: 10.49.
0.5

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


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Sun Apr 20 11:00:05 2025 from 49.196.108.138
root@ubuntu-s-1vcpu-1gb-syd1-01:~#
```

4. System Update & Upgrade

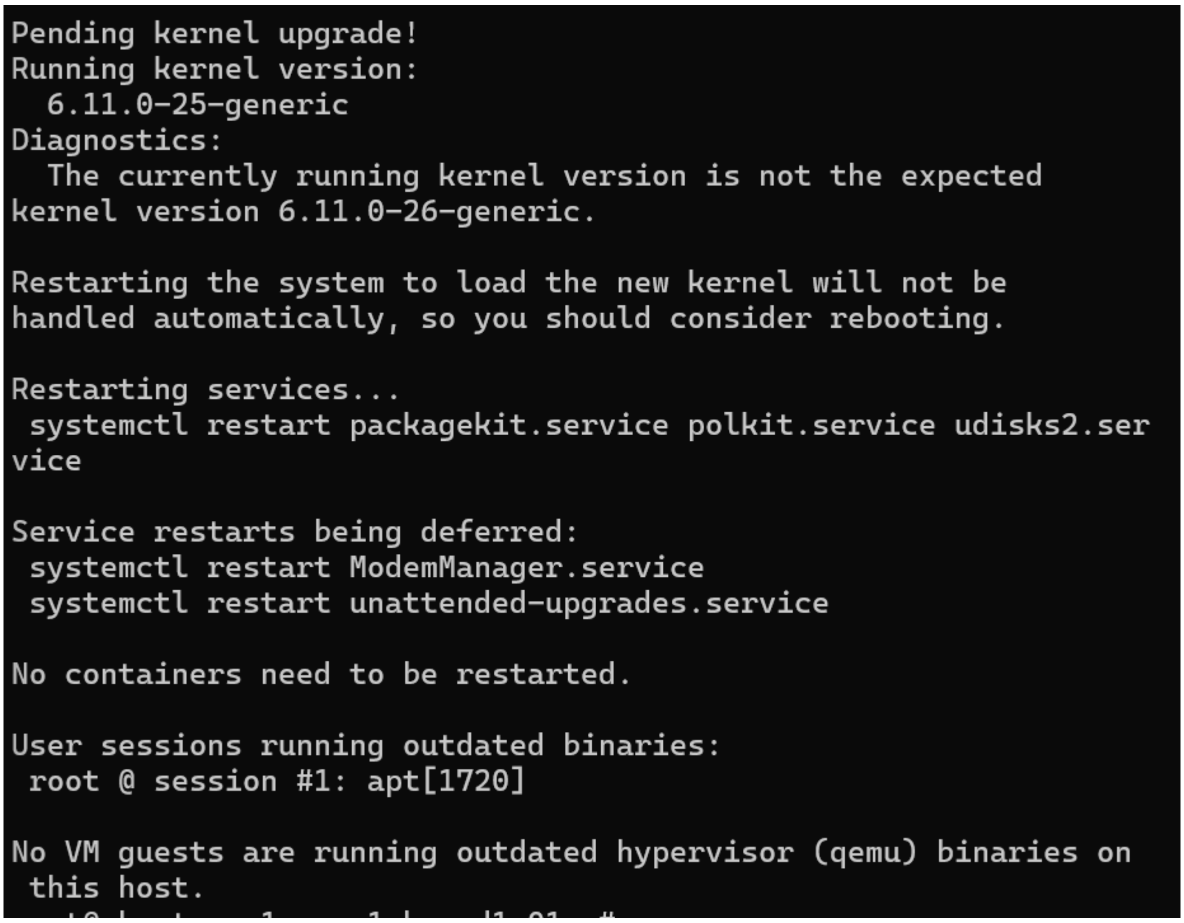
As soon as I managed to get into the server with SSH, the first thing I did was make sure everything was updated and safe. This just means you gotta run some commands to update the package list and then upgrade all the stuff that's installed, so you don't have any old software that could mess things up or be insecure.

- I used: `sudo apt update`
- Then: `sudo apt upgrade -y`
- Doing this right away stops random problems later and makes sure the server is running the newest versions of everything.

Screenshot: Terminal output confirming updates and upgrades were completed successfully.



```
root@ubuntu-s-1vcpu-1gb-syd1-01:~# sudo apt update
Hit:1 https://repos-droplet.digitalocean.com/apt/droplet-agent m
ain InRelease
Hit:2 https://repos.insights.digitalocean.com/apt/do-agent main
InRelease
Hit:3 http://mirrors.digitalocean.com/ubuntu oracular InRelease
Get:4 http://mirrors.digitalocean.com/ubuntu oracular-updates In
Release [126 kB]
Get:5 http://security.ubuntu.com/ubuntu oracular-security InRele
ase [126 kB]
Get:6 http://mirrors.digitalocean.com/ubuntu oracular-backports
InRelease [126 kB]
Get:7 http://mirrors.digitalocean.com/ubuntu oracular-updates/ma
in amd64 Packages [427 kB]
Get:8 http://mirrors.digitalocean.com/ubuntu oracular-updates/ma
in Translation-en [114 kB]
Get:9 http://mirrors.digitalocean.com/ubuntu oracular-updates/ma
in amd64 Components [58.9 kB]
Ign:10 https://ppa.launchpadcontent.net/certbot/certbot/ubuntu o
racular InRelease
Get:11 http://mirrors.digitalocean.com/ubuntu oracular-updates/u
niverse amd64 Packages [265 kB]
Get:12 http://mirrors.digitalocean.com/ubuntu oracular-updates/u
niverse Translation-en [87.5 kB]
```



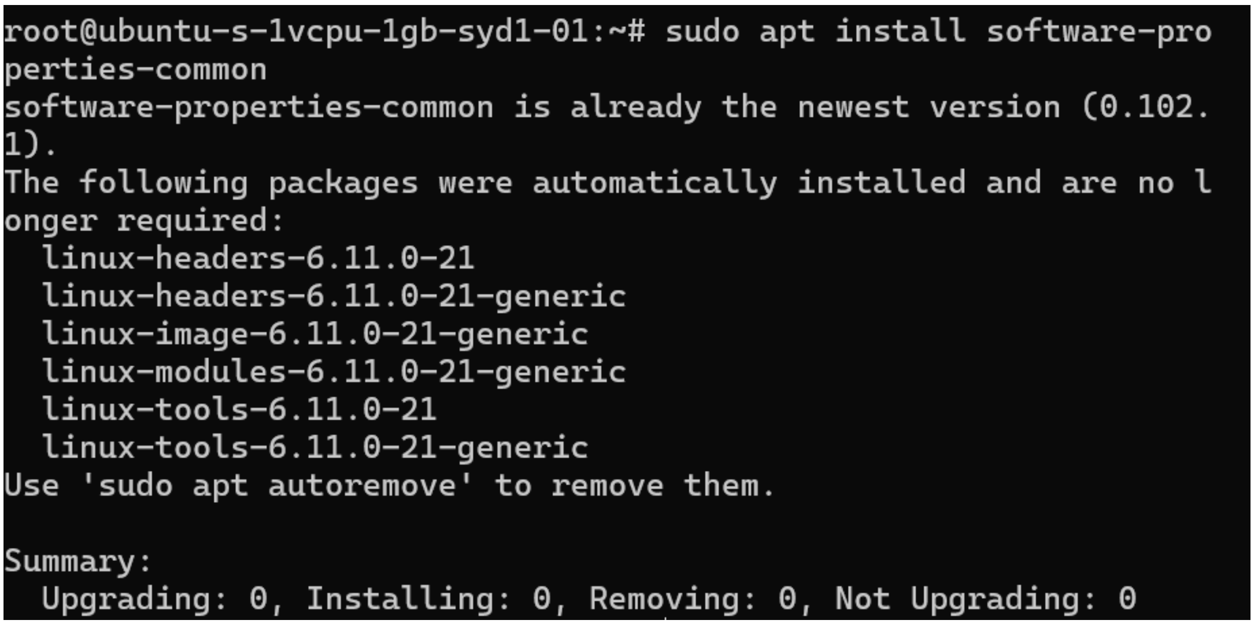
```
Pending kernel upgrade!
Running kernel version:
  6.11.0-25-generic
Diagnostics:
  The currently running kernel version is not the expected
kernel version 6.11.0-26-generic.

Restarting the system to load the new kernel will not be
handled automatically, so you should consider rebooting.

Restarting services...
 systemctl restart packagekit.service polkit.service udisks2.ser
vice

Service restarts being deferred:
 systemctl restart ModemManager.service
 systemctl restart unattended-upgrades.service

No containers need to be restarted.

User sessions running outdated binaries:
 root @ session #1: apt[1720]

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

5. Install Prerequisites

Before installing advanced packages, I installed `software-properties-common` which is required for managing additional repositories and installing third-party software on Ubuntu:

- `sudo apt install software-properties-common`

This package allows for the easy addition of repositories necessary for Certbot and other services later in the project.



```
root@ubuntu-s-1vcpu-1gb-syd1-01:~# sudo apt install software-pro
perties-common
software-properties-common is already the newest version (0.102.
1).
The following packages were automatically installed and are no l
onger required:
  linux-headers-6.11.0-21
  linux-headers-6.11.0-21-generic
  linux-image-6.11.0-21-generic
  linux-modules-6.11.0-21-generic
  linux-tools-6.11.0-21
  linux-tools-6.11.0-21-generic
Use 'sudo apt autoremove' to remove them.

Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
```

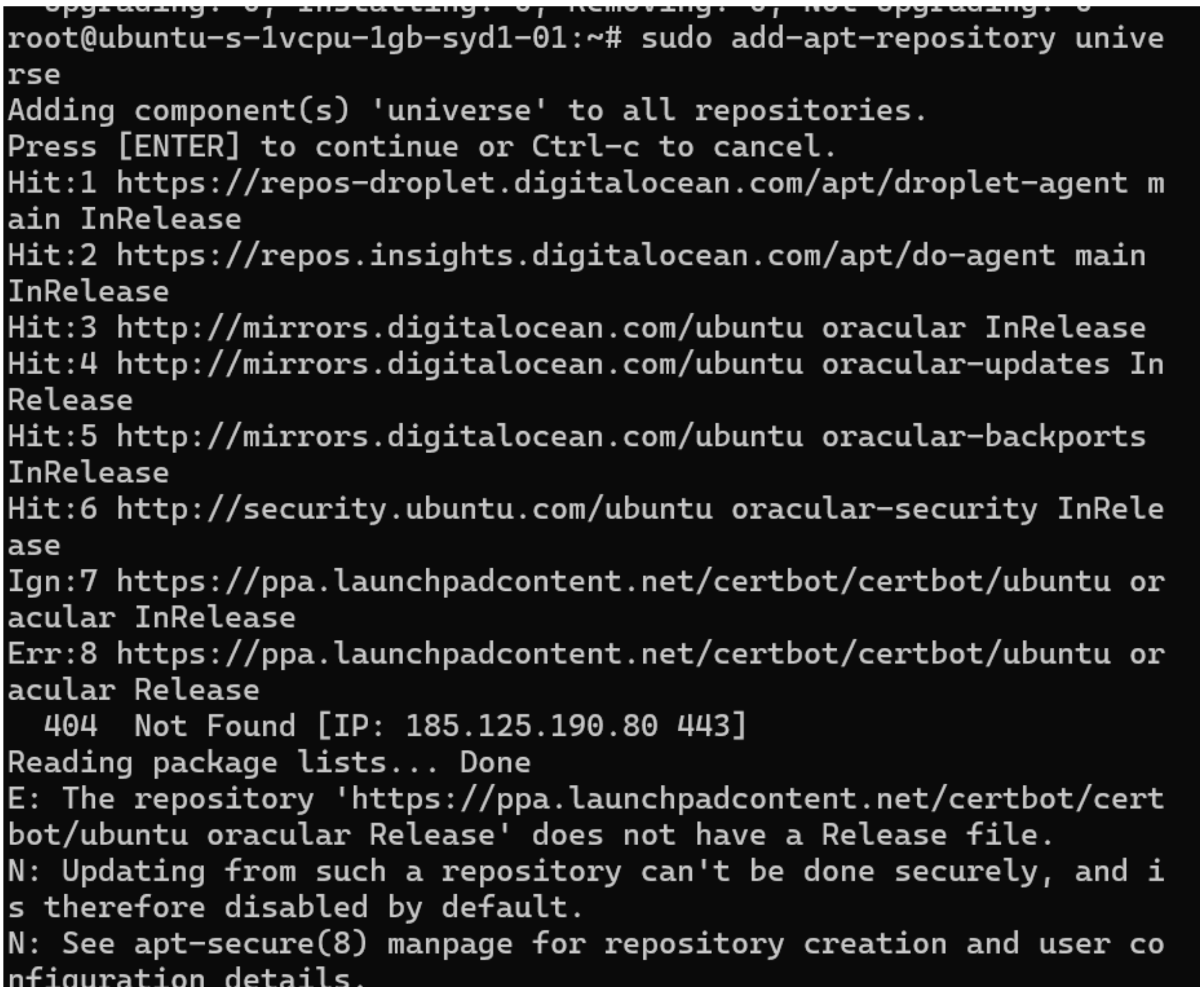
6. Add Universe Repository

One of the first weird things I had to do was enable this “universe” repository on Ubuntu. Apparently, the regular setup doesn’t give you every program you might need, so you have to unlock more stuff yourself. This step is pretty important, especially for things like Certbot that you need later for HTTPS.

- Command I used: `sudo add-apt-repository universe`

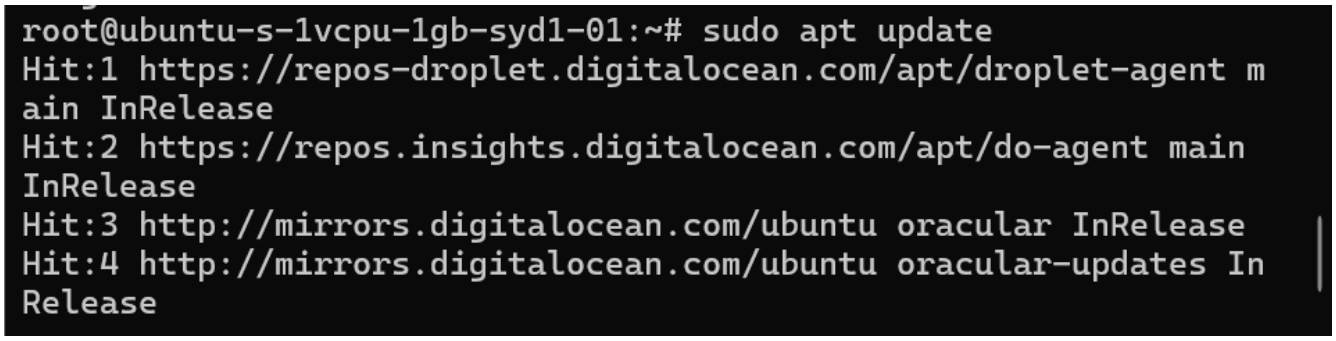
- Then I did: `sudo apt update`

- After this, my server could find and install a bunch of extra software I wouldn’t get otherwise.



```
root@ubuntu-s-1vcpu-1gb-syd1-01:~# sudo add-apt-repository unive
rse
Adding component(s) 'universe' to all repositories.
Press [ENTER] to continue or Ctrl-c to cancel.
Hit:1 https://repos-droplet.digitalocean.com/apt/droplet-agent m
ain InRelease
Hit:2 https://repos.insights.digitalocean.com/apt/do-agent main
InRelease
Hit:3 http://mirrors.digitalocean.com/ubuntu oracular InRelease
Hit:4 http://mirrors.digitalocean.com/ubuntu oracular-updates In
Release
Hit:5 http://mirrors.digitalocean.com/ubuntu oracular-backports
InRelease
Hit:6 http://security.ubuntu.com/ubuntu oracular-security InRele
ase
Ign:7 https://ppa.launchpadcontent.net/certbot/certbot/ubuntu or
acular InRelease
Err:8 https://ppa.launchpadcontent.net/certbot/certbot/ubuntu or
acular Release
  404  Not Found [IP: 185.125.190.80 443]
Reading package lists... Done
E: The repository 'https://ppa.launchpadcontent.net/certbot/cert
bot/ubuntu oracular Release' does not have a Release file.
N: Updating from such a repository can't be done securely, and i
s therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user co
nfiguration details.
```

sudo apt update



```
root@ubuntu-s-1vcpu-1gb-syd1-01:~# sudo apt update
Hit:1 https://repos-droplet.digitalocean.com/apt/droplet-agent m
ain InRelease
Hit:2 https://repos.insights.digitalocean.com/apt/do-agent main
InRelease
Hit:3 http://mirrors.digitalocean.com/ubuntu oracular InRelease
Hit:4 http://mirrors.digitalocean.com/ubuntu oracular-updates In
Release
```

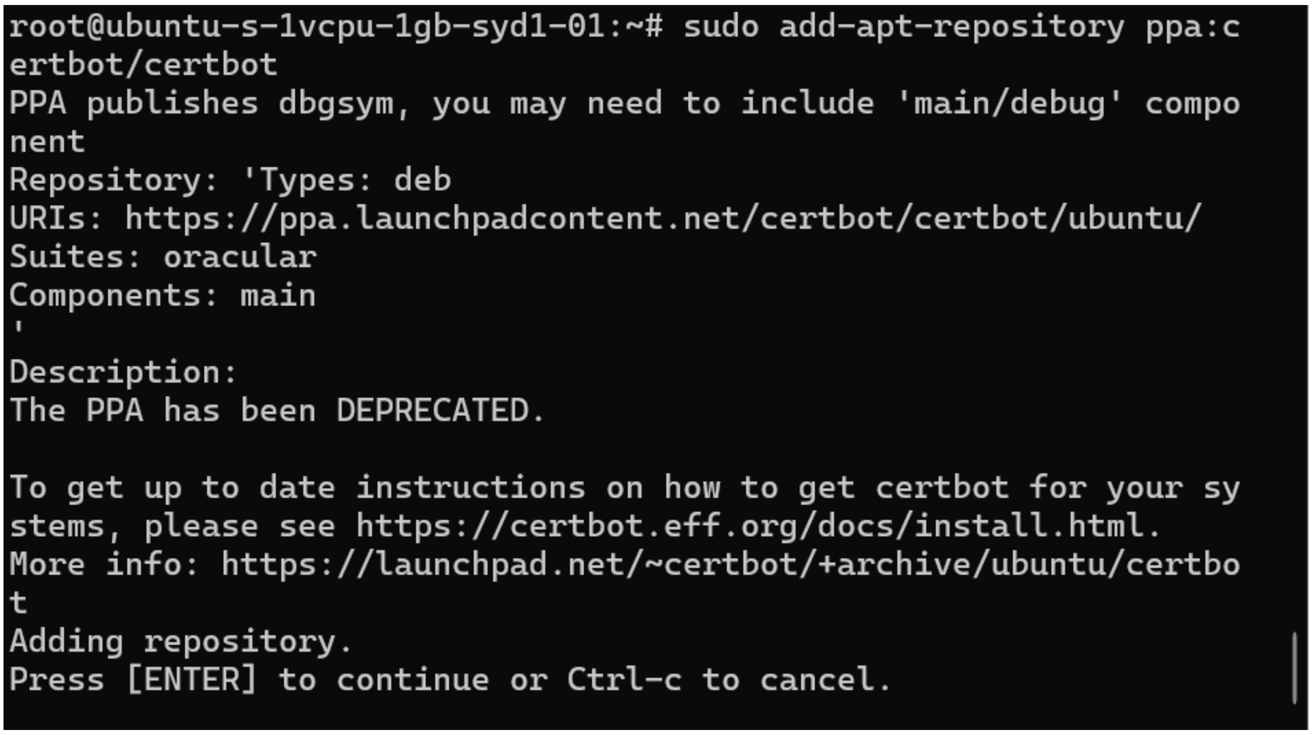
7. Add Certbot PPA

Certbot is what lets you get those free SSL certificates so your site is secure and has that padlock. But the version in Ubuntu can be pretty old, so I had to add a special source (they call it a PPA) to make sure I'd get the newest Certbot. Without this, you can't really keep Certbot updated properly, which could mess up your HTTPS later on.

- I ran: `sudo add-apt-repository ppa:certbot/certbot`

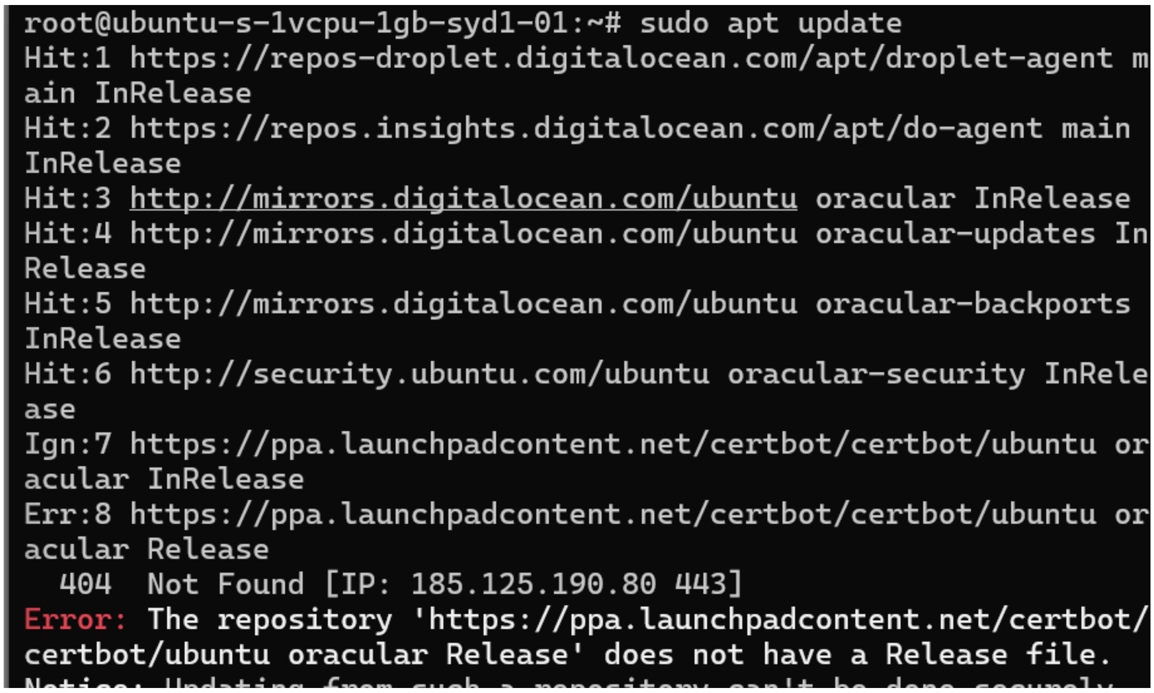
- And then: `sudo apt update`

- Doing this meant I could install Certbot without any issues and keep it working for the future.



```
root@ubuntu-s-1vcpu-1gb-syd1-01:~# sudo add-apt-repository ppa:c
ertbot/certbot
PPA publishes dbgsym, you may need to include 'main/debug' compo
nent
Repository: 'Types: deb
URIs: https://ppa.launchpadcontent.net/certbot/certbot/ubuntu/
Suites: oracular
Components: main
'
Description:
The PPA has been DEPRECATED.

To get up to date instructions on how to get certbot for your sy
stems, please see https://certbot.eff.org/docs/install.html.
More info: https://launchpad.net/~certbot/+archive/ubuntu/certbo
t
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
```

sudo apt update



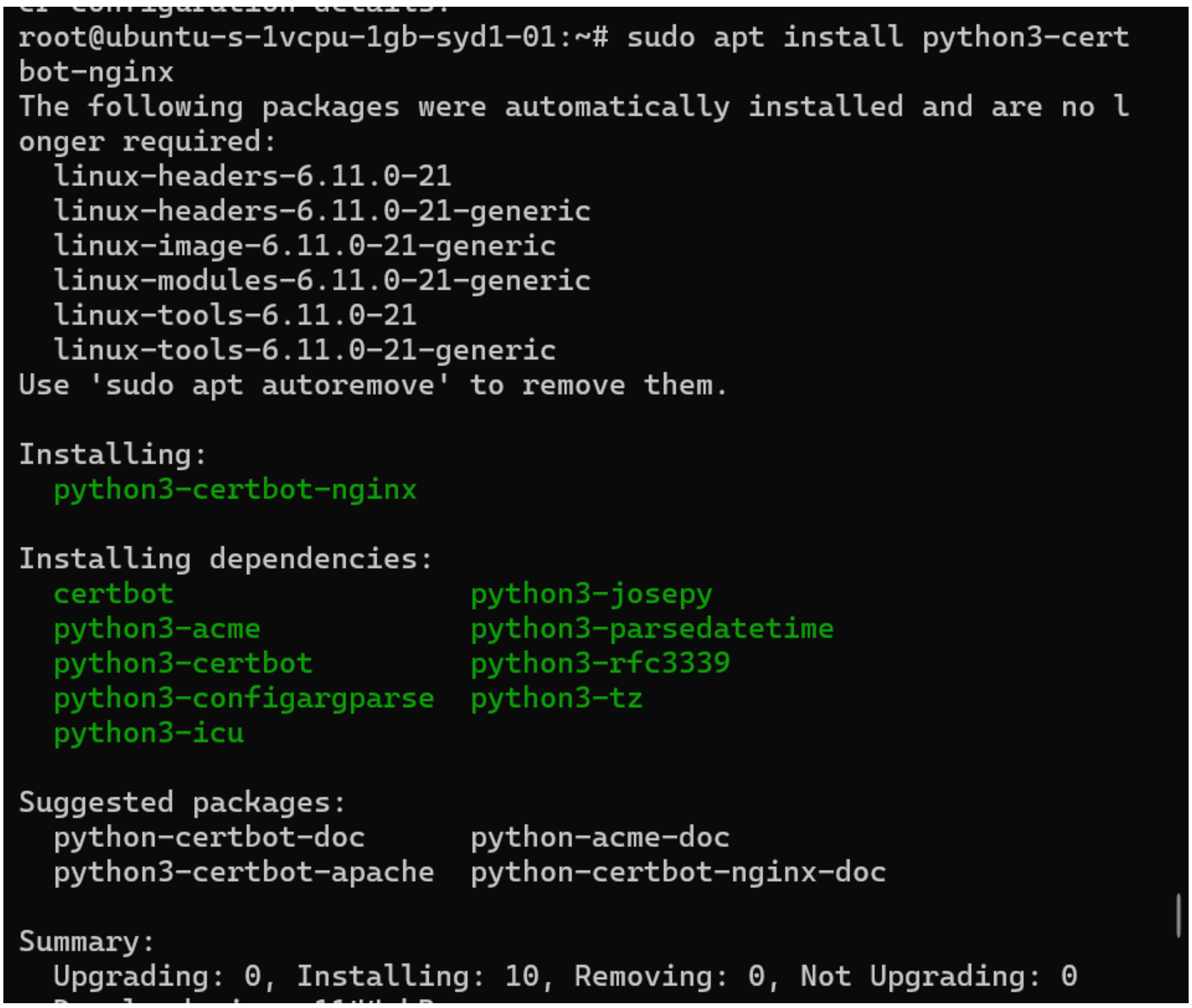
```
root@ubuntu-s-1vcpu-1gb-syd1-01:~# sudo apt update
Hit:1 https://repos-droplet.digitalocean.com/apt/droplet-agent m
ain InRelease
Hit:2 https://repos.insights.digitalocean.com/apt/do-agent main
InRelease
Hit:3 http://mirrors.digitalocean.com/ubuntu oracular InRelease
Hit:4 http://mirrors.digitalocean.com/ubuntu oracular-updates In
Release
Hit:5 http://mirrors.digitalocean.com/ubuntu oracular-backports
InRelease
Hit:6 http://security.ubuntu.com/ubuntu oracular-security InRele
ase
Ign:7 https://ppa.launchpadcontent.net/certbot/certbot/ubuntu or
acular InRelease
Err:8 https://ppa.launchpadcontent.net/certbot/certbot/ubuntu or
acular Release
  404  Not Found [IP: 185.125.190.80 443]
Error: The repository 'https://ppa.launchpadcontent.net/certbot/
certbot/ubuntu oracular Release' does not have a Release file.
Notice: Updating from such a repository can't be done securely
```

8. Install Certbot’s Nginx Package

To make Certbot actually work with my Nginx server, I had to install its special Nginx package. Without this, Certbot doesn’t really know how to set up HTTPS on Nginx automatically. After running this, it just kind of connects the two and makes it way easier to set up SSL or renew certificates.

- I used: `sudo apt install python3-certbot-nginx`
- This made it so Certbot could talk to Nginx and do all the HTTPS setup for me, which saved a ton of time.

sudo apt install python3-certbot-nginx



```
root@ubuntu-s-1vcpu-1gb-syd1-01:~# sudo apt install python3-cert
bot-nginx
The following packages were automatically installed and are no l
onger required:
  linux-headers-6.11.0-21
  linux-headers-6.11.0-21-generic
  linux-image-6.11.0-21-generic
  linux-modules-6.11.0-21-generic
  linux-tools-6.11.0-21
  linux-tools-6.11.0-21-generic
Use 'sudo apt autoremove' to remove them.

Installing:
  python3-certbot-nginx

Installing dependencies:
  certbot                   python3-josepy
  python3-acme              python3-parsedatetime
  python3-certbot           python3-rfc3339
  python3-configargparse    python3-tz
  python3-icu

Suggested packages:
  python-certbot-doc        python-acme-doc
  python3-certbot-apache    python-certbot-nginx-doc

Summary:
  Upgrading: 0, Installing: 10, Removing: 0, Not Upgrading: 0
```

9. Setup SSL/TLS with Certbot for Nginx

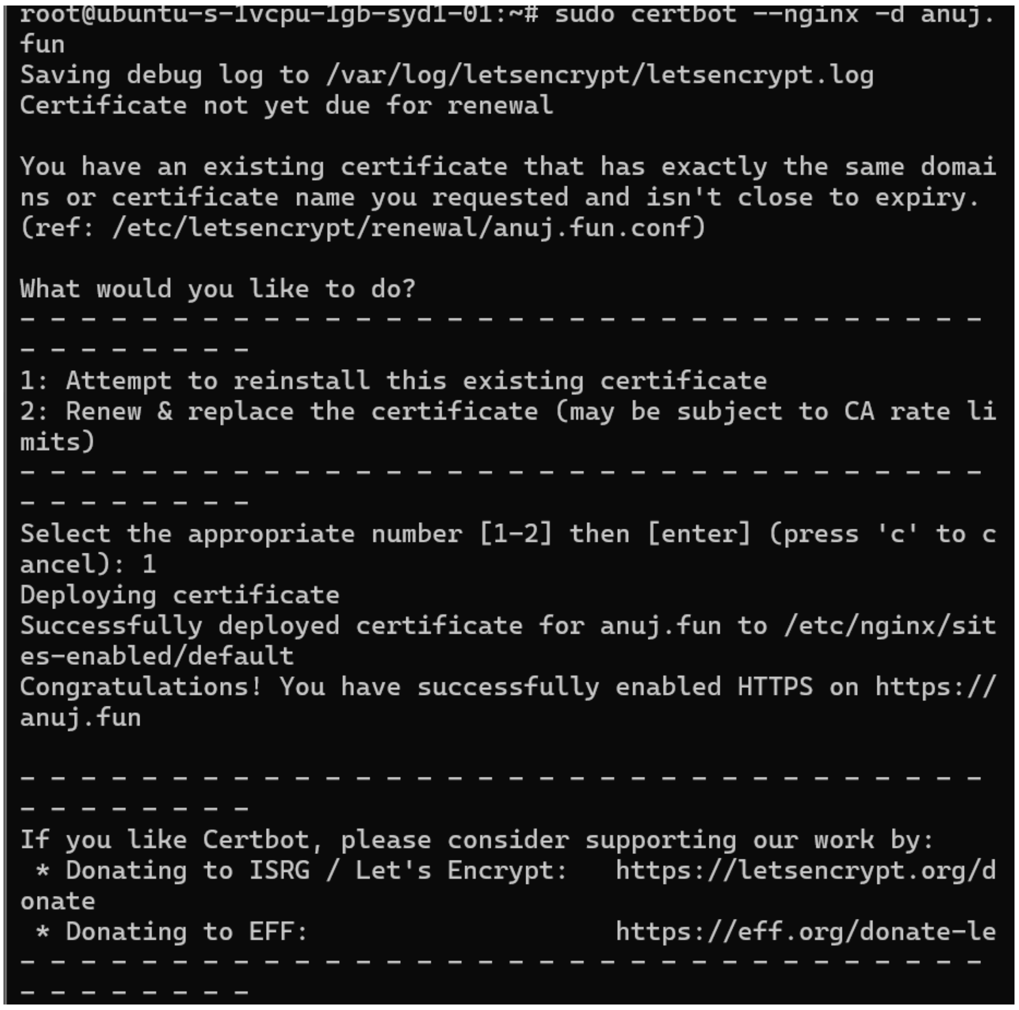
Getting SSL/TLS working is super important these days so people’s info is safe and your site shows that padlock. I used Certbot with Nginx to grab a free SSL cert from Let’s Encrypt and set everything up. The command was actually pretty simple, and Certbot did all the tricky stuff for me.

- The main command: `sudo certbot --nginx -d anuj.fun`

- Certbot went and fixed my Nginx config, so now anyone who goes to http just gets sent straight to https.

- It even set up the certs to renew automatically, so I don’t have to stress about it expiring. This not only makes the site secure but also looks more legit in Google and for anyone visiting.

sudo certbot --nginx -d anuj.fun



```
root@ubuntu-s-1vcpu-1gb-syd1-01:~# sudo certbot --nginx -d anuj.
fun
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Certificate not yet due for renewal

You have an existing certificate that has exactly the same domai
ns or certificate name you requested and isn't close to expiry.
(ref: /etc/letsencrypt/renewal/anuj.fun.conf)

What would you like to do?
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - -
1: Attempt to reinstall this existing certificate
2: Renew & replace the certificate (may be subject to CA rate li
mits)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - -
Select the appropriate number [1-2] then [enter] (press 'c' to c
ancel): 1
Deploying certificate
Successfully deployed certificate for anuj.fun to /etc/nginx/sit
es-enabled/default
Congratulations! You have successfully enabled HTTPS on https://
anuj.fun

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - -
If you like Certbot, please consider supporting our work by:
 * Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/d
onate
 * Donating to EFF:                    https://eff.org/donate-le
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - -
```

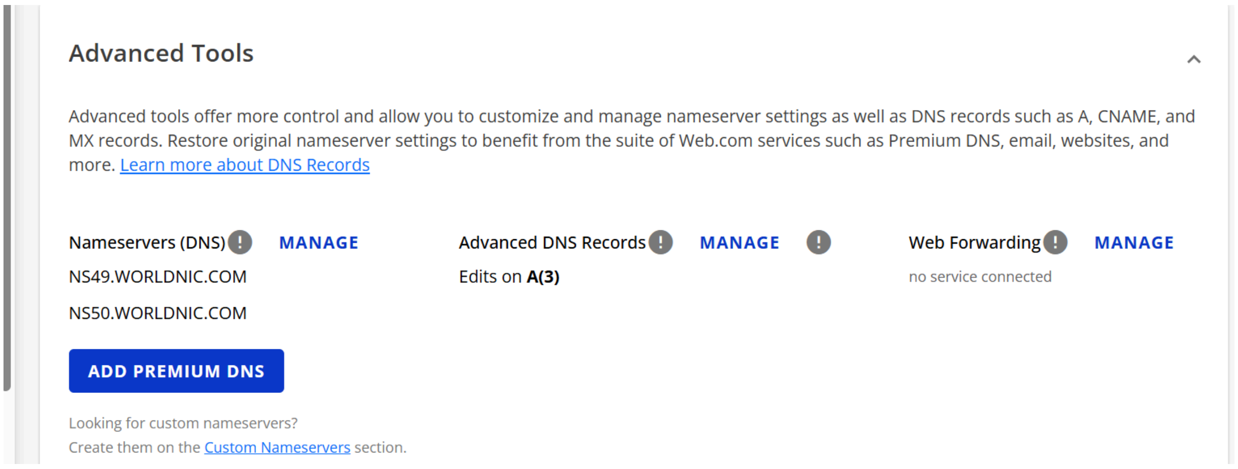
Screenshot: Successful Certbot output, browser showing secure HTTPS.

10. DNS Functionality and Accessibility

For the domain part, I had to go into where I bought my domain(web.com) and set up what they call an "A" record. Basically, I pointed "anuj.fun" at my server's IP (170.64.253.234). It didn't work right away—sometimes DNS just takes a while to catch up, which was kind of annoying.

- I kept checking with `nslookup` and `ping` in the terminal, plus used some online DNS checker sites.
- After a bit of waiting, my domain finally started sending people to my server, and I knew it was all set up right.

- DNS Provider settings verification
- DNS propagation check
  Screenshot: Verified DNS resolution with ping and DNS lookup.



Advanced Tools

Advanced tools offer more control and allow you to customize and manage nameserver settings as well as DNS records such as A, CNAME, and MX records. Restore original nameserver settings to benefit from the suite of Web.com services such as Premium DNS, email, websites, and more. Learn more about DNS Records

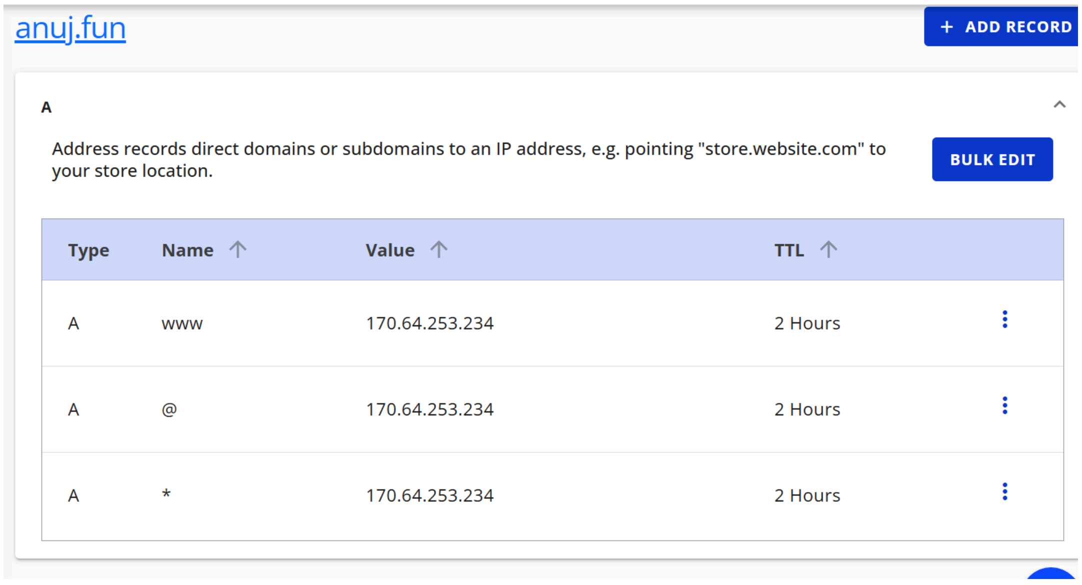
Nameservers (DNS) MANAGE
NS49.WORLDNIC.COM
NS50.WORLDNIC.COM

Advanced DNS Records MANAGE
Edits on **A(3)**

Web Forwarding MANAGE
no service connected

ADD PREMIUM DNS

Looking for custom nameservers?
Create them on the Custom Nameservers section.

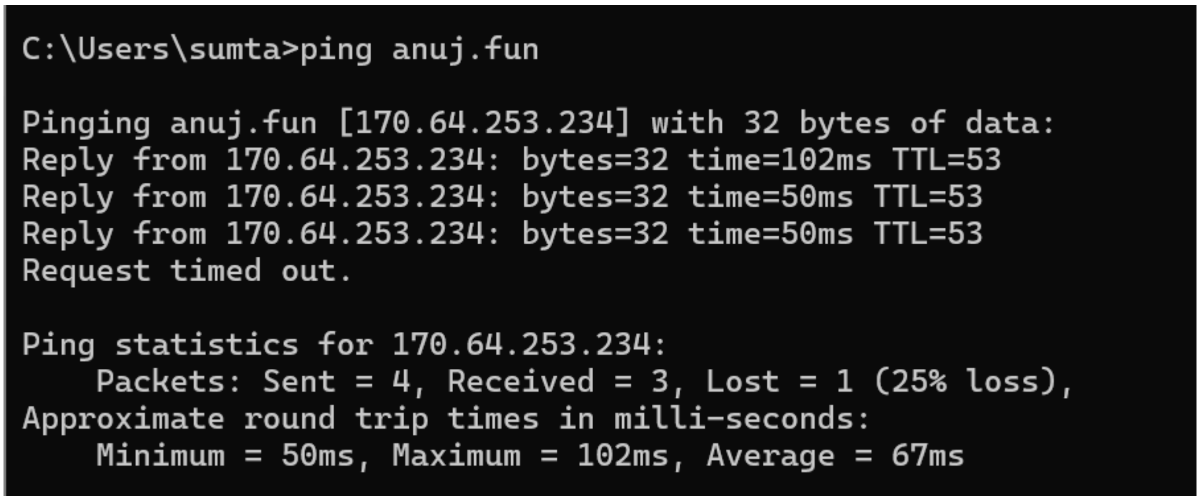
- 

anuj.fun

A

Address records direct domains or subdomains to an IP address, e.g. pointing "store.website.com" to your store location.

Type	Name	Value	TTL
A	www	170.64.253.234	2 Hours
A	@	170.64.253.234	2 Hours
A	*	170.64.253.234	2 Hours



```
C:\Users\sumta>ping anuj.fun

Pinging anuj.fun [170.64.253.234] with 32 bytes of data:
Reply from 170.64.253.234: bytes=32 time=102ms TTL=53
Reply from 170.64.253.234: bytes=32 time=50ms TTL=53
Reply from 170.64.253.234: bytes=32 time=50ms TTL=53
Request timed out.

Ping statistics for 170.64.253.234:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 50ms, Maximum = 102ms, Average = 67ms
```

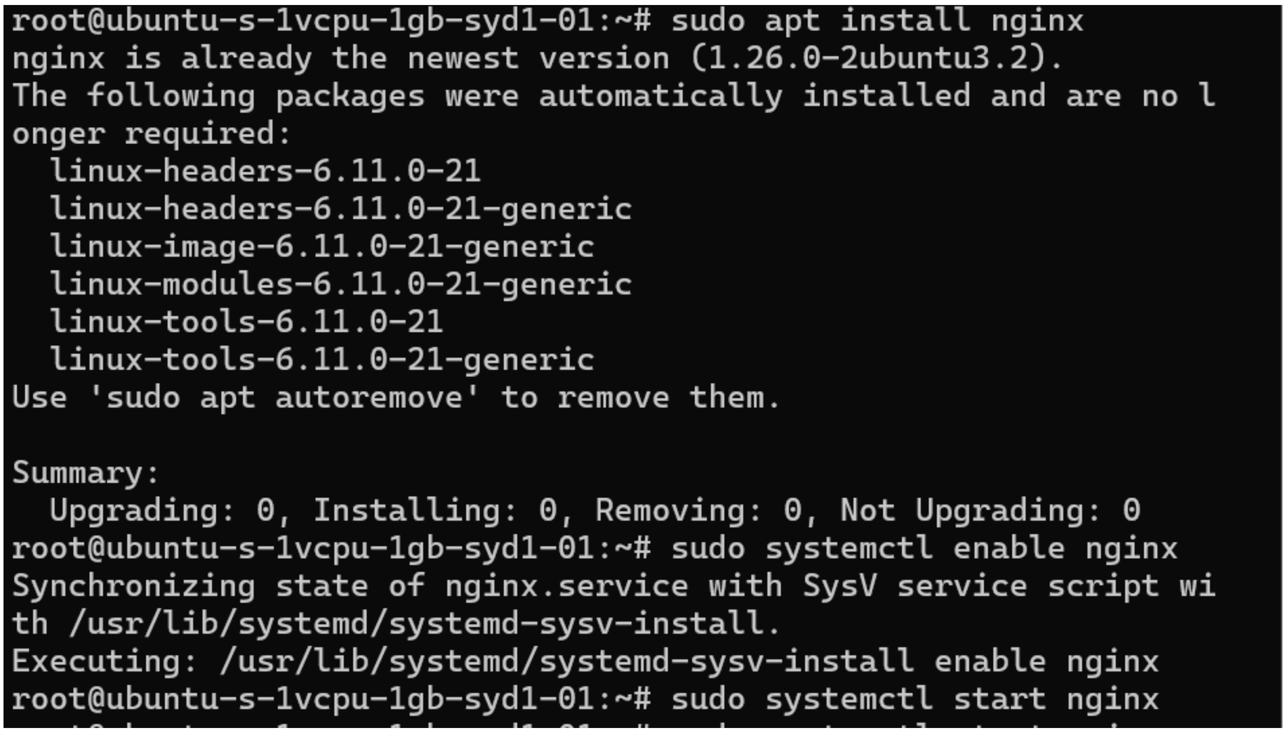
11. Web Server Configuration (Nginx)

I set up Nginx as my main web server to run the website. The install was pretty easy—just a couple commands—and I made sure it always started up if the server rebooted. Didn't really change much in the config files at first, just wanted it to work for both http and https.

- Commands I used:
  - `sudo apt install nginx`
  - `sudo systemctl enable nginx`
  - `sudo systemctl start nginx`

- To check it all worked, I just went to my domain and saw the Nginx welcome page. Later I swapped that out for my own index.html with my project info. After that, my site was running and ready for people to visit.

Screenshot: Nginx configuration and running status.

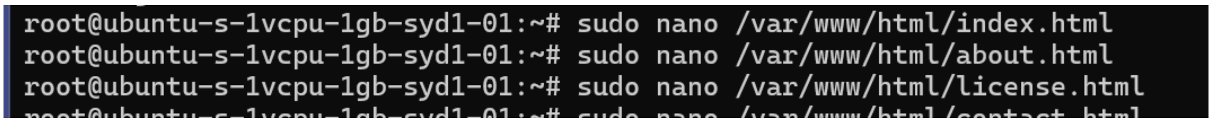


```
root@ubuntu-s-1vcpu-1gb-syd1-01:~# sudo apt install nginx
nginx is already the newest version (1.26.0-2ubuntu3.2).
The following packages were automatically installed and are no l
onger required:
  linux-headers-6.11.0-21
  linux-headers-6.11.0-21-generic
  linux-image-6.11.0-21-generic
  linux-modules-6.11.0-21-generic
  linux-tools-6.11.0-21
  linux-tools-6.11.0-21-generic
Use 'sudo apt autoremove' to remove them.

Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
root@ubuntu-s-1vcpu-1gb-syd1-01:~# sudo systemctl enable nginx
Synchronizing state of nginx.service with SysV service script wi
th /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable nginx
root@ubuntu-s-1vcpu-1gb-syd1-01:~# sudo systemctl start nginx
```

12. Editing HTML Page

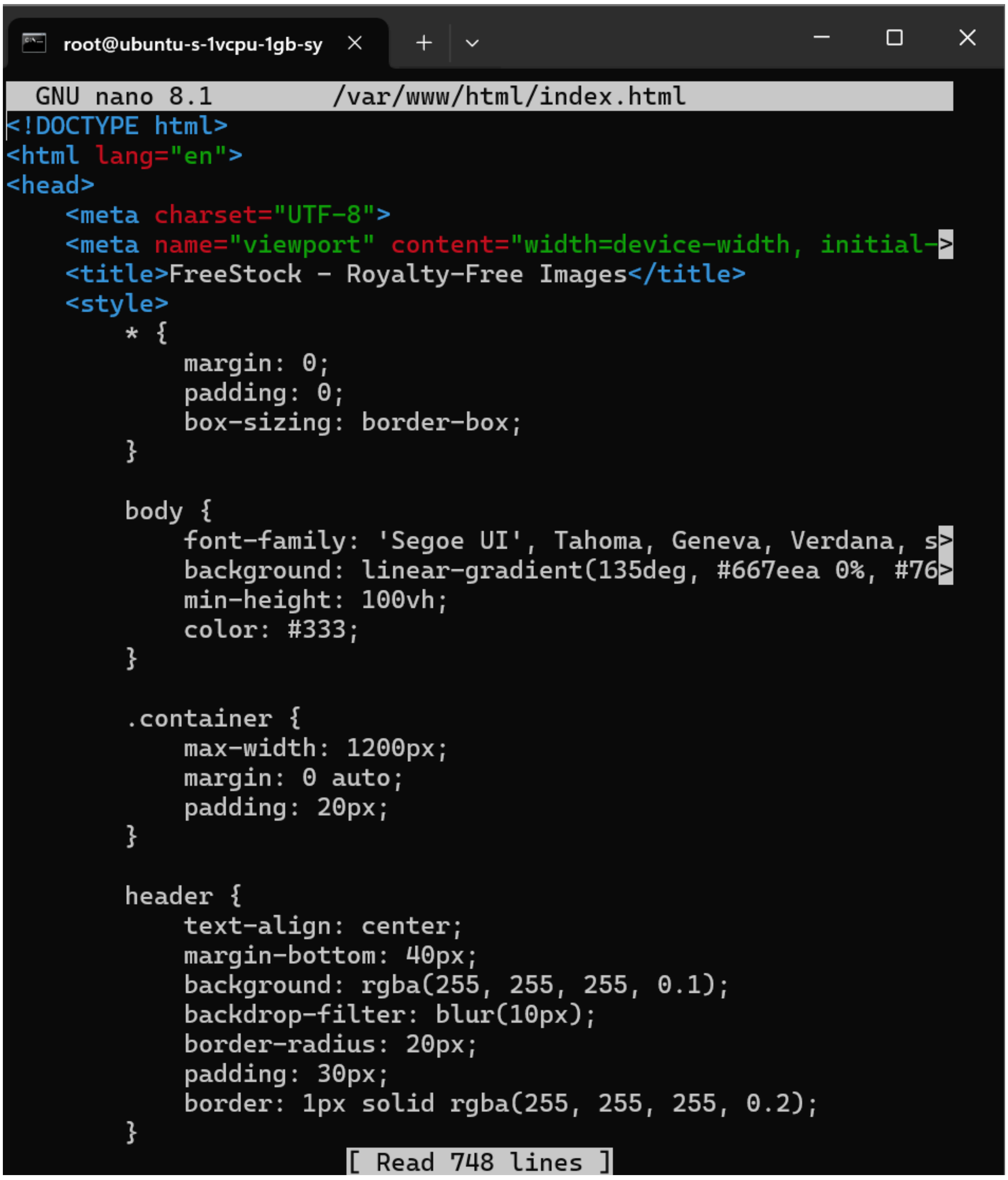
Customized default HTML content to include project description, student ID, and essential information:

- File path: `/var/www/html/index.html`

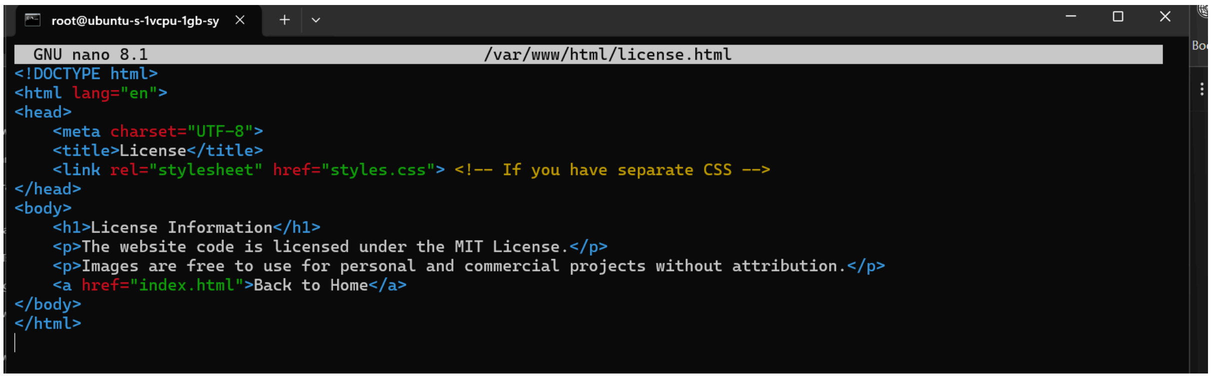


```
root@ubuntu-s-1vcpu-1gb-syd1-01:~# sudo nano /var/www/html/index.html
root@ubuntu-s-1vcpu-1gb-syd1-01:~# sudo nano /var/www/html/about.html
root@ubuntu-s-1vcpu-1gb-syd1-01:~# sudo nano /var/www/html/license.html
root@ubuntu-s-1vcpu-1gb-syd1-01:~# sudo nano /var/www/html/contact.html
```

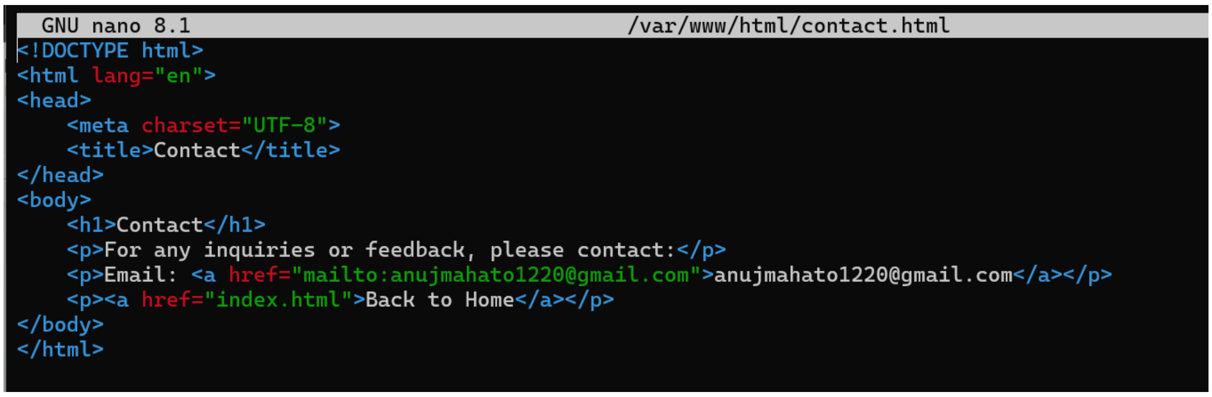
- HTML structure explained and modified
  Screenshot: Code snippet showing edits.



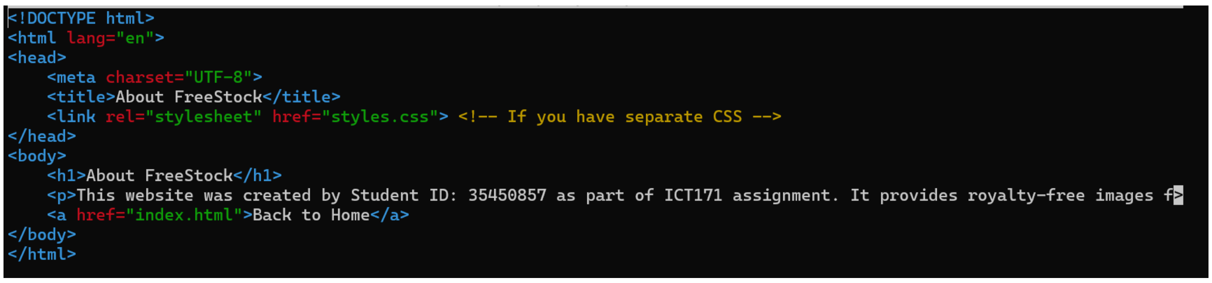
```
  GNU nano 8.1              /var/www/html/index.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial->
    <title>FreeStock - Royalty-Free Images</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, s>
            background: linear-gradient(135deg, #667eea 0%, #76>
            min-height: 100vh;
            color: #333;
        }

        .container {
            max-width: 1200px;
            margin: 0 auto;
            padding: 20px;
        }

        header {
            text-align: center;
            margin-bottom: 40px;
            background: rgba(255, 255, 255, 0.1);
            backdrop-filter: blur(10px);
            border-radius: 20px;
            padding: 30px;
            border: 1px solid rgba(255, 255, 255, 0.2);
        }
                       [ Read 748 lines ]
```



```
  GNU nano 8.1                                   /var/www/html/license.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>License</title>
    <link rel="stylesheet" href="styles.css"> <!-- If you have separate CSS -->
</head>
<body>
    <h1>License Information</h1>
    <p>The website code is licensed under the MIT License.</p>
    <p>Images are free to use for personal and commercial projects without attribution.</p>
    <a href="index.html">Back to Home</a>
</body>
</html>
```



```
  GNU nano 8.1                                          /var/www/html/contact.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Contact</title>
</head>
<body>
    <h1>Contact</h1>
    <p>For any inquiries or feedback, please contact:</p>
    <p>Email: <a href="mailto:anujmahato1220@gmail.com">anujmahato1220@gmail.com</a></p>
    <p><a href="index.html">Back to Home</a></p>
</body>
</html>
```

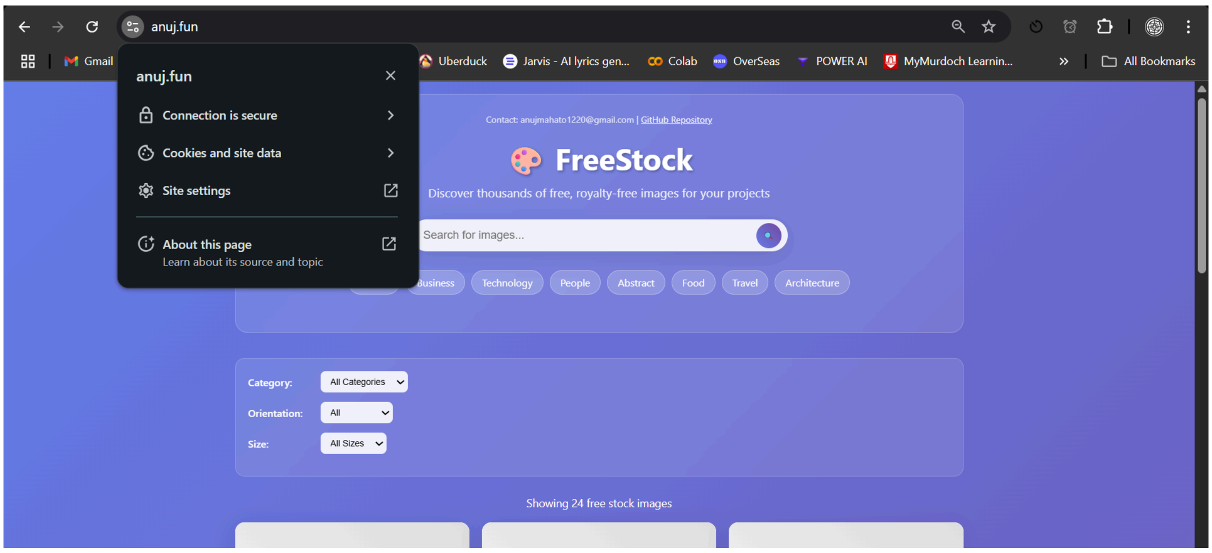


```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>About FreeStock</title>
    <link rel="stylesheet" href="styles.css"> <!-- If you have separate CSS -->
</head>
<body>
    <h1>About FreeStock</h1>
    <p>This website was created by Student ID: 35450857 as part of ICT171 assignment. It provides royalty-free images f>
    <a href="index.html">Back to Home</a>
</body>
</html>
```

13. Final Website and Functionality

The finished website is up and running at https://anuj.fun for anyone to check out. The home page has a short welcome, my project info, my student ID, and I made sure to put the MIT license down the bottom so it's clear anyone can use the code. All those setup steps finally led to a proper website that's live with https (and the padlock shows it's secure) on my own domain name. I even tested it on my phone and some friends' laptops just to make sure it loads everywhere and looks right.

Screenshot: Just take a screenshot of the homepage in a browser, showing the domain, padlock, and all the info on the page.



14. GitHub Repository Submission & Commits

I put all my code, scripts, and files in a public GitHub repo so it's easy for anyone to see what I did or even copy it if they want. Tried to remember to make a commit after every big change, like when I set stuff up, changed the html, or worked on the scripts. Each commit has a short note so I remember what I changed. My repo's got everything: index.html, server.sh, backup_script.sh, plus a README that explains how it works and how to set it up yourself.

GitHub Link:
Screenshot: Screenshot of the repo's commit history and file structure.

15. Bash Scripting (server.sh)

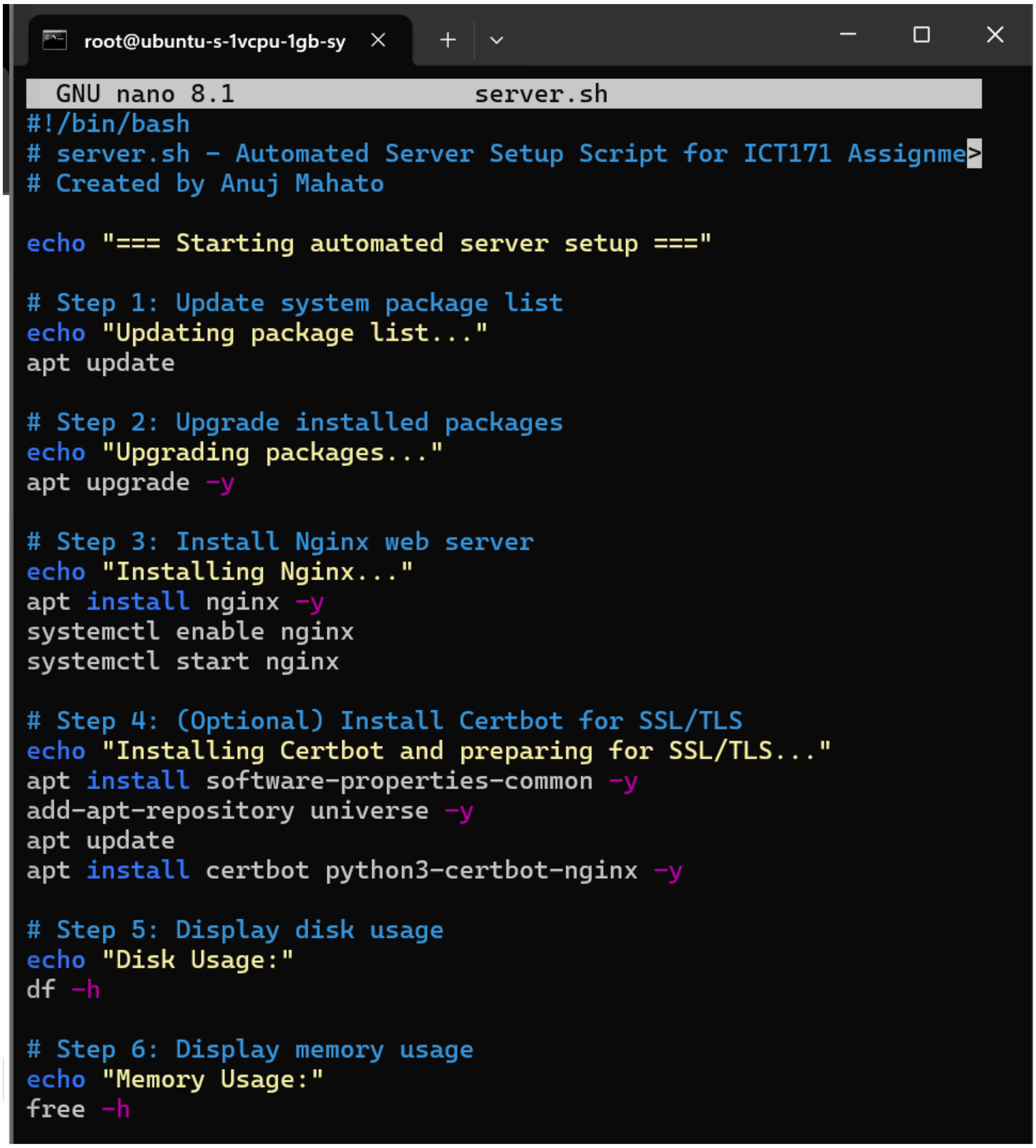
Automated routine server setup tasks with a clearly commented script:

To automate server setup, I wrote a bash script (`server.sh`) that updates system packages, installs and configures Nginx, and prepares the environment for further deployment. This script reduces manual steps and increases repeatability. Additionally, I created a backup script that compresses the website's documents into a date-stamped archive, safeguarding project data.

Both scripts are thoroughly commented to explain each command and are included in the GitHub repo for review. Testing and verification confirmed that running the scripts resulted in a working server and reliable automated backups.

```
#!/bin/bash
# Updates and installs necessary software
sudo apt update
sudo apt upgrade -y
sudo apt install nginx -y
sudo systemctl enable nginx
```

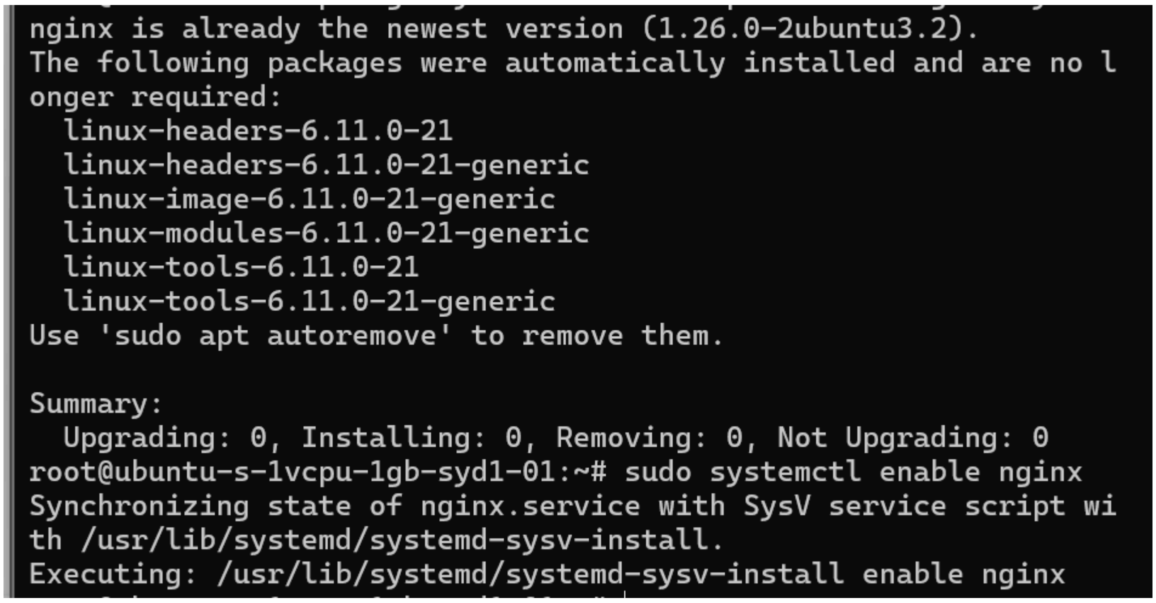
Screenshot: Execution of server.sh in terminal.



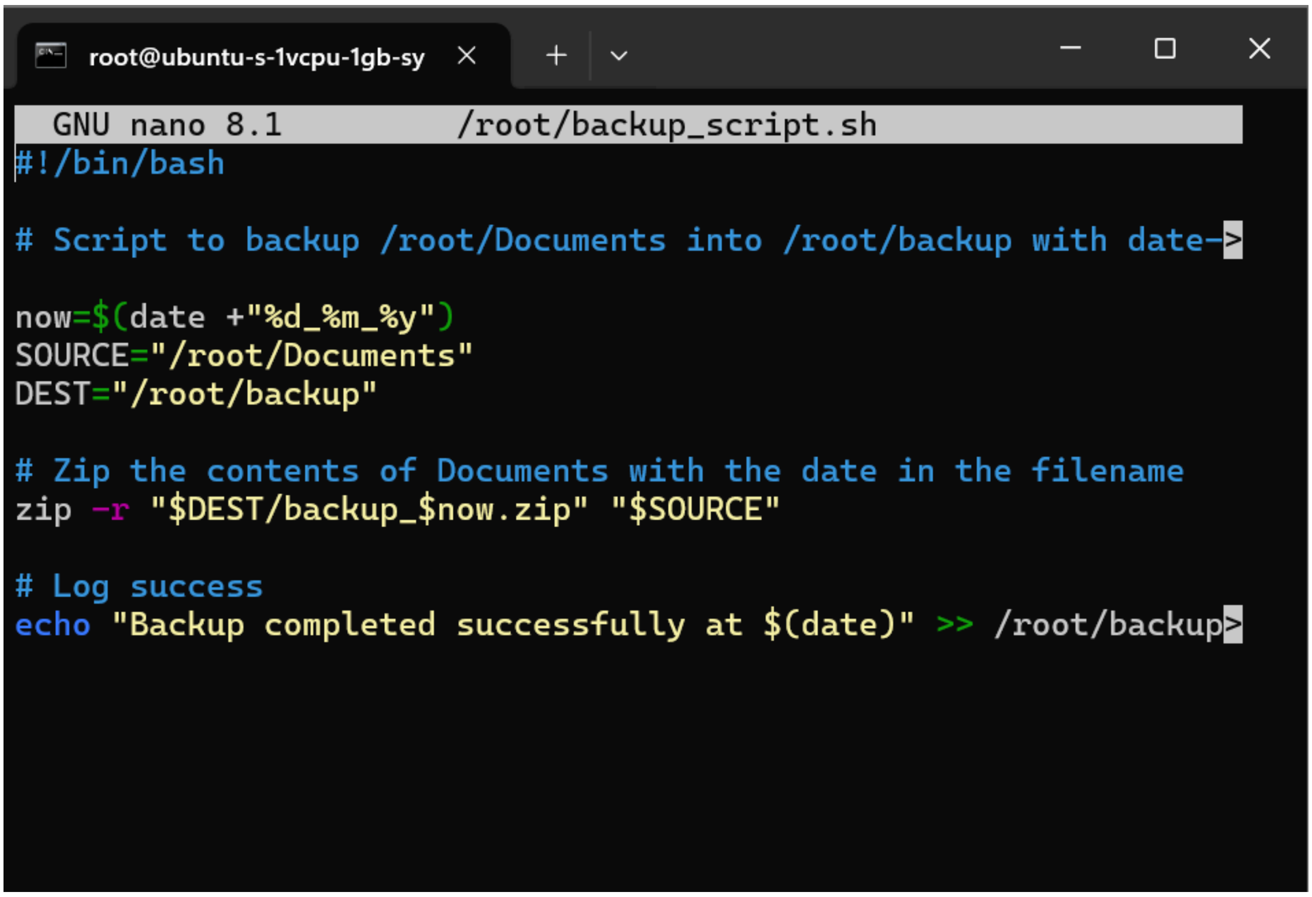
root@ubuntu-s-1vcpu-1gb-sy

```
  GNU nano 8.1                    server.sh
#!/bin/bash
# server.sh - Automated Server Setup Script for ICT171 Assignme>
# Created by Anuj Mahato

echo "=== Starting automated server setup ==="

# Step 1: Update system package list
echo "Updating package list..."
apt update

# Step 2: Upgrade installed packages
echo "Upgrading packages..."
apt upgrade -y

# Step 3: Install Nginx web server
echo "Installing Nginx..."
apt install nginx -y
systemctl enable nginx
systemctl start nginx

# Step 4: (Optional) Install Certbot for SSL/TLS
echo "Installing Certbot and preparing for SSL/TLS..."
apt install software-properties-common -y
add-apt-repository universe -y
apt update
apt install certbot python3-certbot-nginx -y

# Step 5: Display disk usage
echo "Disk Usage:"
df -h

# Step 6: Display memory usage
echo "Memory Usage:"
free -h
```

For Backup.sh

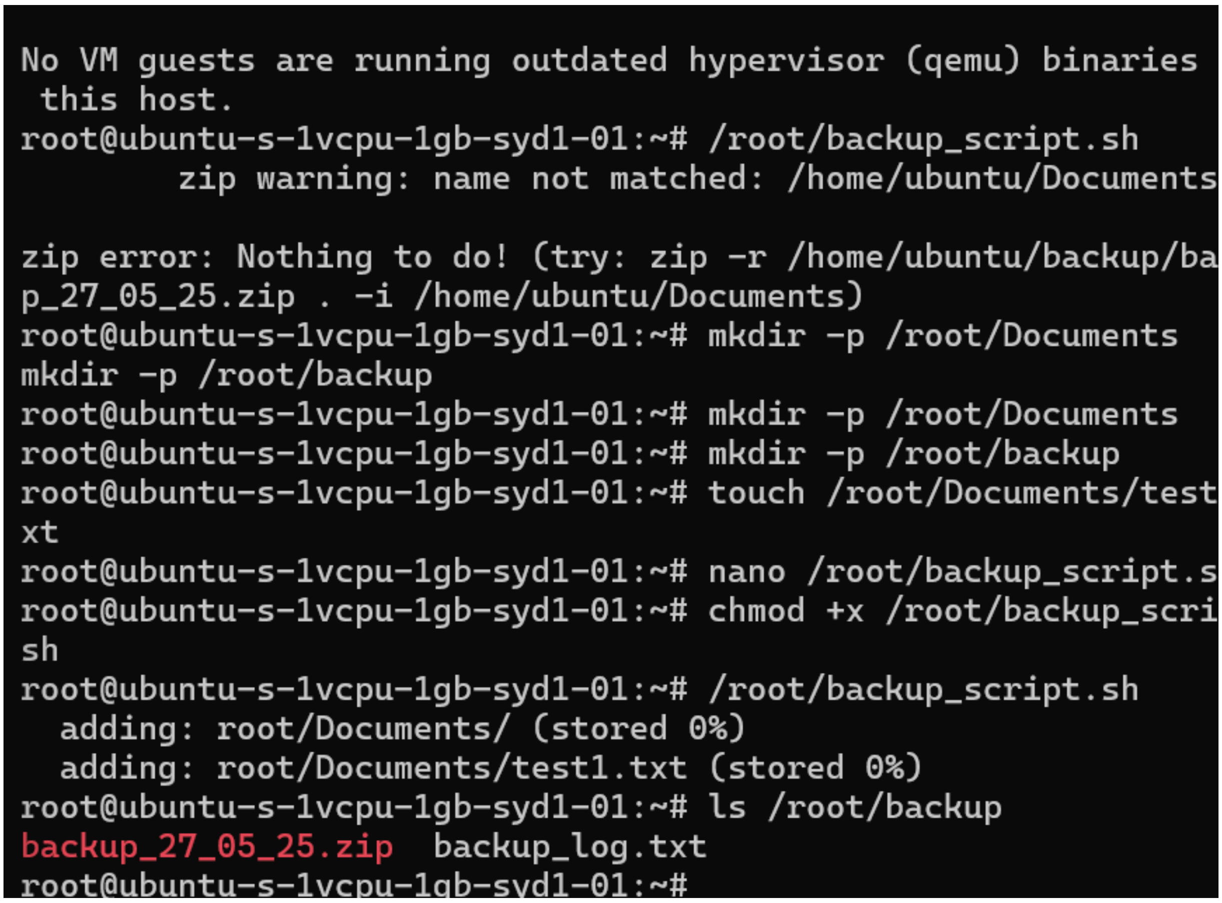


```
nginx is already the newest version (1.26.0-2ubuntu3.2).
The following packages were automatically installed and are no l
onger required:
  linux-headers-6.11.0-21
  linux-headers-6.11.0-21-generic
  linux-image-6.11.0-21-generic
  linux-modules-6.11.0-21-generic
  linux-tools-6.11.0-21
  linux-tools-6.11.0-21-generic
Use 'sudo apt autoremove' to remove them.

Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
root@ubuntu-s-1vcpu-1gb-syd1-01:~# sudo systemctl enable nginx
Synchronizing state of nginx.service with SysV service script wi
th /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable nginx
```



root@ubuntu-s-1vcpu-1gb-sy

```
  GNU nano 8.1           /root/backup_script.sh
#!/bin/bash

# Script to backup /root/Documents into /root/backup with date->

now=$(date +"%d_%m_%y")
SOURCE="/root/Documents"
DEST="/root/backup"

# Zip the contents of Documents with the date in the filename
zip -r "$DEST/backup_$now.zip" "$SOURCE"

# Log success
echo "Backup completed successfully at $(date)" >> /root/backup>
```



```
No VM guests are running outdated hypervisor (qemu) binaries
 this host.
root@ubuntu-s-1vcpu-1gb-syd1-01:~# /root/backup_script.sh
        zip warning: name not matched: /home/ubuntu/Documents

zip error: Nothing to do! (try: zip -r /home/ubuntu/backup/ba
p_27_05_25.zip . -i /home/ubuntu/Documents)
root@ubuntu-s-1vcpu-1gb-syd1-01:~# mkdir -p /root/Documents
mkdir -p /root/backup
root@ubuntu-s-1vcpu-1gb-syd1-01:~# mkdir -p /root/Documents
root@ubuntu-s-1vcpu-1gb-syd1-01:~# mkdir -p /root/backup
root@ubuntu-s-1vcpu-1gb-syd1-01:~# touch /root/Documents/test
xt
root@ubuntu-s-1vcpu-1gb-syd1-01:~# nano /root/backup_script.s
root@ubuntu-s-1vcpu-1gb-syd1-01:~# chmod +x /root/backup_scri
sh
root@ubuntu-s-1vcpu-1gb-syd1-01:~# /root/backup_script.sh
  adding: root/Documents/ (stored 0%)
  adding: root/Documents/test1.txt (stored 0%)
root@ubuntu-s-1vcpu-1gb-syd1-01:~# ls /root/backup
backup_27_05_25.zip  backup_log.txt
root@ubuntu-s-1vcpu-1gb-syd1-01:~#
```

16. Windows Scripting (File Transfer & Server Access)

I needed to move files between my Windows laptop and my cloud server, so I used Command Prompt and sometimes PowerShell to connect to the server. I mostly used the scp command (or WinSCP if I wanted to just drag and drop) to send files like index.html straight into the web folder on my server.

After putting in my server password, I could upload or even download files between Windows and Linux without any hassle. Doing this showed I can handle remote server stuff and keep my project up to date from my own computer, not just in the Linux terminal.

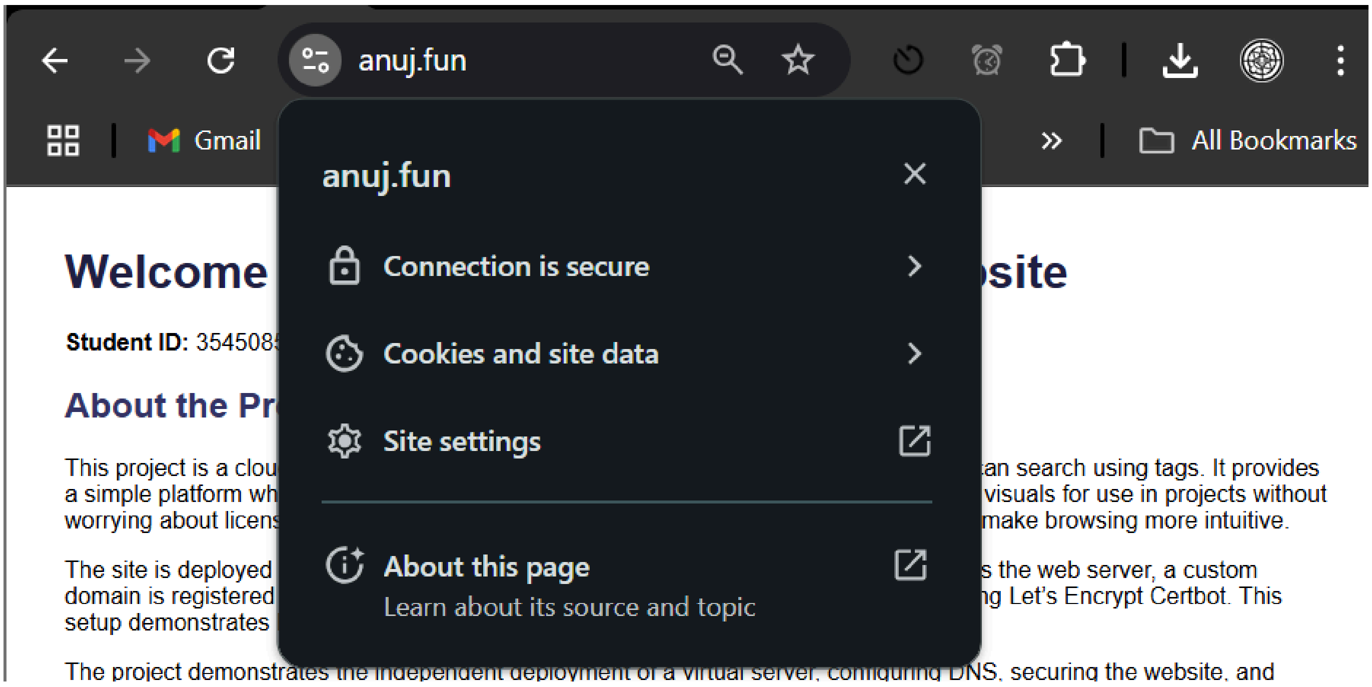
17. Script Usefulness and Output

Making scripts for setting up the server and backups honestly saves heaps of time. Instead of typing out all the steps every single time, I just run the script and it does everything for me in a few seconds.

- Less chance for mistakes too, since the script always does it the same way every time.
- If I (or someone else) ever needs to rebuild the site or get the files back, they just run the script and it works—no special tricks needed.
- Right after the scripts finish, my website is up and backups are sitting there in the folder.

I checked everything after running them, and both the site and backup files were exactly where they should be, so I know the scripts worked.

Screenshot: Resulting site functionality after running scripts.



Welcome … site

Student ID: 354508

About the Pr

This project is a clou … an search using tags. It provides
a simple platform wh … visuals for use in projects without
worrying about licen … make browsing more intuitive.

The site is deployed … s the web server, a custom
domain is registered … ng Let's Encrypt Certbot. This
setup demonstrates

The project demonstrates the independent deployment of a virtual server, configuring DNS, securing the website, and

18. License Explanation

I picked the MIT License for my project ’cause it’s simple and super open. Anyone can use my code, change it up, or share it, as long as they give me credit and keep the license bit in there. It just fits with the point of the project—making it easy for other students or anyone to use what I made without worrying about legal stuff. So, anyone can build on this for their own uni work or just for learning.

The full license text is available in both the website footer and the GitHub repository.

19. Video Demonstration & Script

A concise, engaging video clearly demonstrates:

https://youtu.be/HqxH3ka98Qo

- Project introduction and goals
- Live demonstration of website features and secure connection
- Explanation and demonstration of SSH and scripting automation
- GitHub repository review
- Concluding statements
  Script: Clearly written and aligned with spoken presentation.

References

1. **DigitalOcean documentation:**
   https://docs.digitalocean.com/
2. **Certbot user guide:**
   https://certbot.eff.org/docs/
3. **Nginx official documentation:**
   https://nginx.org/en/docs/