

Project Report

Cloud Deployment Project Title: Sarran Dojo

Student Name: Aruljeyakumar Sarrangan

Student ID: 35603826

Project Description:

For this project, I built a static website in the cloud. The idea was to create a website for a martial arts club called **Sarran Dojo**, which offers karate and kickboxing classes. The site includes the class schedule, coach information, pricing, and a special “OSS” button that triggers a fun blood animation with CSS and JavaScript. This was inspired by my personal interest in martial arts and fighting sports.

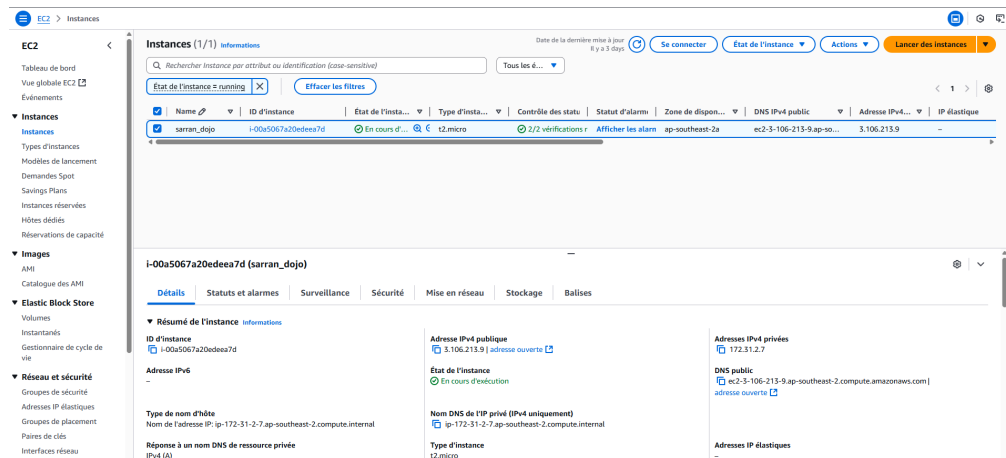
The website was made with HTML and CSS only. It's hosted on an AWS EC2 instance running Ubuntu 22.04, and I set up Apache2 as the web server. To get the website online, I used SCP to copy my files to the server's /var/www/html/ folder. Visitors can see the site using either the public IP address (**3.106.213.9**) or the DNS domain name (**dojosarran.xyz**).

I also created a simple Bash script called **check_ports.sh** to check which ports are open on the server and compare them with a list of expected ports, which helps to spot any suspicious activity.

Creating the server (AWS EC2):

To start this project, I first created an account on **Amazon Web Services** (AWS). Then, I went to the AWS EC2 (Elastic Compute Cloud) section. I clicked on **Launch Instance** to make a new server. For the operating system, I chose Ubuntu Server 22.04 LTS. During the setup, I also created a key pair (a **.pem** file) to securely connect to the server later. I downloaded it and saved it on my computer. After that I allowed port 22 (SSH), so I can connect to the server, and port (HTTP) so people can see my website. Finally, I clicked on **Launch Instance** and waited a few minutes for the server to be ready. AWS gave me a public IP (**3.106.213.9**) address for my server, for the access online

Screenshot:



Here, you can see my AWS EC2 instance details. This page shows that the instance is running and gives me the public IP (3.106.213.9) that I used to connect to it and to point to my domain name.

Connecting to the server (SSH):

After launching my EC2 instance, To do this, I used SSH, which lets me access the server securely from my computer. First, I downloaded the .pem key file when I created the instance on AWS. This key file is very important because it acts like a password to connect safely. I made sure to move it to a safe location on my computer.

Screenshot:

```
(thelatest@kali)~$ cd Documents
(thelatest@kali)~/Documents$ ls
aws/
(thelatest@kali)~/Documents$ cd aws
(thelatest@kali)~/Documents/aws$ ls
check_ports.sh  sarran_dojo_key.pem
```

This is my .pem file

Then, in my terminal (I used Kali Linux), I navigated to the folder where my .pem file was saved. I used the following command to connect to the server:

```
ssh -i ~/Documents/aws/sarran_dojo_key.pem ubuntu@3.106.213.9
```

Screenshot:

```
thelatest@kali: ~/Documents/aws
$ ssh -i ~/Documents/aws/sarran_dojo_key.pem ubuntu@3.106.213.9
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-1024-aws x86_64)
* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/pro

System information as of Sun Jun  8 13:30:58 UTC 2025

System load:  0.08      Processes:      108
Usage of /:   38.2% of 6.71GB   Users logged in:  0
Memory usage: 29%      IPv4 address for enX0: 172.31.2.7
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.

48 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

*** System restart required ***
Last login: Fri Jun  6 03:13:49 2025 from 203.132.71.102
ubuntu@ip-172-31-2-7:~$
```

I am connected to the server now

Installing Apache and Testing the Web Server:

After connecting to my server, I needed to install Apache2, which is the web server software.

I installed Apache2 with this command: **sudo apt install apache2 -y**

I enabled the Apache service so that it starts automatically if the server restarts: **sudo systemctl enable apache2**

Finally, I started the Apache service: **sudo systemctl start apache2**

After confirming that Apache was working, I was ready to upload my website files. My website was created using HTML and CSS on my computer. I used the SCP command to copy the files to the **/var/www/html/** folder on the server

ScreenShot:

```
the1atest@kali: ~/Documents/aws
$ ssh -i ~/Documents/aws/sarran_dojo_key.pem ubuntu@3.106.213.9
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-1024-aws x86_64)
RUDDISH
 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Sun Jun  8 13:48:29 UTC 2025
Kernel: 6.8.0-1024-aws
System load:  0.0          Processes:            111
Usage of /:   38.2% of 6.71GB   Users logged in:      0
Memory usage: 29%          IPv4 address for enx0: 172.31.2.7
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.

48 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

*** System restart required ***
Last login: Sun Jun  8 13:48:30 2025 from 203.132.71.102
ubuntu@ip-172-31-2-7:~$ cd /var/www/html/
ubuntu@ip-172-31-2-7:/var/www/html$ ls
index.html
```

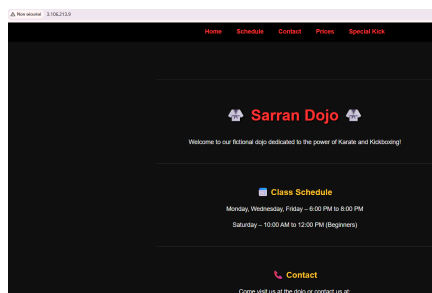
We saw the index.html file ready to be seen !

Accessing the Website in the Browser:

Once I uploaded all my files, I opened my web browser and typed the public IP address of my server (3.106.213.9). I saw my website appear.

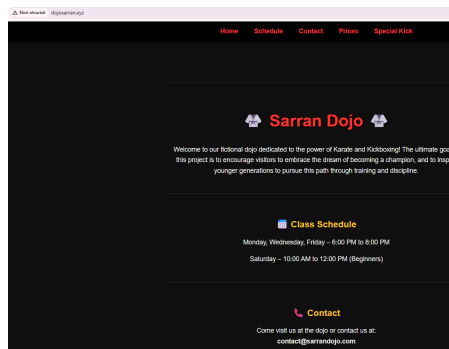
After I saw that my website was working with the public IP address (3.106.213.9), I wanted to make it easier for people to visit by using a custom domain name. So I bought the domain dojosarran.xyz from a domain registrar on Porkbun. Then, I logged into my domain registrar's website and went to the DNS management section. There, I created an A record that points dojosarran.xyz to my server's public IP address (3.106.213.9). it take a few minutes for the DNS propagation to complete .Once that was done, I tested my domain in the browser and saw my website appear at dojosarran.xyz

Screenshot:



I tested my website in the browser using 3.106.213.9, it's worked

Screenshot:



I tested my website in the browser using dojosarran.xyz, it's worked

The Script: `check_ports.sh`:

For my script, I chose to make a simple security check that focuses on the open ports of my server. My idea was to add a basic layer of protection to make sure that no unexpected or suspicious ports were open.

I created a Bash script called `check_ports.sh`. The script lists all the open ports on the server using the `netstat` command. Then, it compares the open ports to a list of expected ports (22 for SSH, 80 for HTTP, 443 for HTTPS). If it finds other ports, it marks them as suspicious. This helps to keep my server safe from unexpected services that might be open by accident.

Explanation of the `check_ports.sh` script:

The script starts by printing a title and then using the `netstat` command to list all the open ports on the server. It uses the options `-tulnp` to display TCP and UDP ports that are currently open and in listening state:

```
sudo netstat -tulnp
```

Then, it defines the expected ports (22 for SSH, 80 for HTTP, 443 for HTTPS). These are the ports that should be open normally:

```
EXPECTED_PORTS="22 80 443"
```

Next, it extracts the list of actual open ports from the output of `netstat`:

```
OPEN_PORTS=$(sudo netstat -tuln | grep LISTEN | awk '{print $4}' | sed 's/.*://')
```

This command: `grep LISTEN`: filters only lines showing listening ports, `awk '{print $4}'`: extracts the fourth column, which is the local port and IP, `sed 's/.*://'`: removes everything before the colon, so we only get the port number.

```
echo
for port in $OPEN_PORTS; do
    if [[ "$EXPECTED_PORTS" =~ "$port" ]]; then
        echo "✓ Port $port : OK (expected)"
    else
        echo "✗ Port $port : suspicious or unexpected!"
    fi
done
```

Here in the screenshot, we see the output of my Bash script. First, it lists all the open ports on my server. We can see the protocol (TCP or UDP) and the program that is using it. After that, the script checks which ports are expected to be open and which ones are suspicious. I expected port 80 (for the website), and 443 (for HTTPS). The script finds the actual open ports and compares them to this list. In this example, the script found that the ports 80 and 443 are **OK** because they are expected. But it also found port 53, which it says is **suspicious or unexpected**.

License:

I used the **MIT License** so that anyone can reuse and share my project code. I also wrote that if someone reuses it, they should mention the original website (dojosarran.xyz).

Adding the Project to GitHub:

I created a public GitHub repository to store my website's code and related files. I used the Git commands to initialize my local repository and upload my files:

Screenshot:

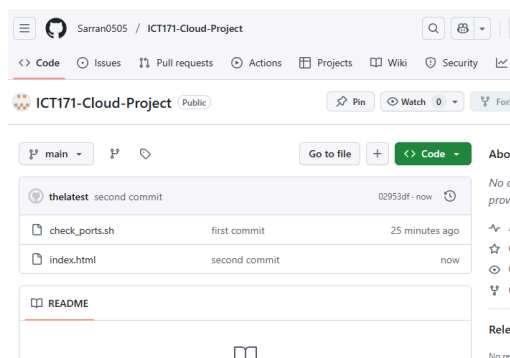
```
thelatest@kali: ~/Documents/aws
$ git push origin main
Username for 'https://github.com': Sarran0505
Password for 'https://Sarran0505@github.com':
thelatest@kali: ~/Documents/aws
$ git push origin main
Username for 'https://github.com': Sarran0505
Password for 'https://Sarran0505@github.com':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Waiting for objects: 100% (3/3), 278 bytes | 278.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com:Sarran0505/ICT171-Cloud-Project.git
+ [new branch] main -> main
branch 'main' set up to track 'origin/main'.
thelatest@kali: ~/Documents/aws
$ ls
check_ports.sh  sarran_dojo_key.pem
thelatest@kali: ~/Documents/aws
$ git add check_ports.sh
thelatest@kali: ~/Documents/aws
$ git commit -m "first commit"
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    sarran_dojo_key.pem

nothing added to commit but untracked files present (use "git add" to track)
thelatest@kali: ~/Documents/aws
$ git push
Everything up-to-date
```

I show how I connected to my GitHub and push my .sh file with git commands

ScreenShot:



I successfully uploaded my project files to my GitHub repository using git commands

GitHub Repository : All the project files are publicly available on my GitHub repository: <https://github.com/Sarran0505/ICT171-Cloud-Project>

Conclusion:

Overall, this project provided an opportunity to practice cloud concepts, Linux command line, Apache configuration, basic web design, and deployment in a real-world cloud scenario. It was an informative and engaging experience.

Through this project, I learned a lot about how to create and host a website, how to work with a real cloud server, and how to write a simple Bash script to add a security layer. Specifically, I chose to create a script to check for open ports on my server. This taught me more about server security and how to protect my site from unexpected activity. I also learned how to use Git and GitHub to manage my project files.

In the future, I plan to improve this website and turn it into a real project for my own dojo when I open one. This experience gave me a good experience

References:

AWS documentation: <https://docs.aws.amazon.com>

Apache2 documentation: <https://httpd.apache.org/docs>

Porkbun documentation (for DNS setup): <https://porkbun.com>

Bash scripting basics: <https://linuxize.com/post/bash-for-loop>

Netstat command (for port scanning): <https://linux.die.net/man/8/netstat>

LMS MyMurdoch resources