Project Report

Cloud Deployment Project Title: Sarran Dojo
Student Name: Aruljeyakumar Sarrangan
Student ID: 35603826

1. Project Description

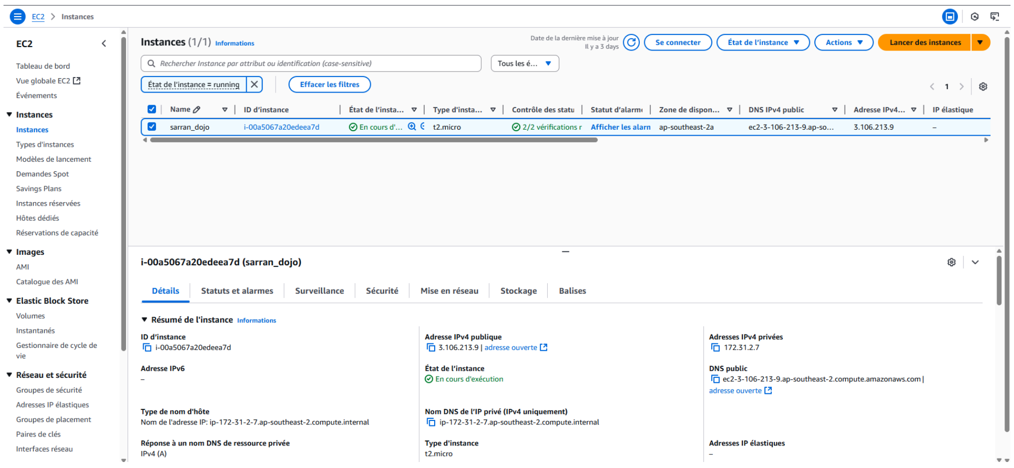
For this project, I built a static website in the cloud. The idea was to create a website for a martial arts club called Sarran Dojo, which offers karate and kickboxing classes. The site includes the class schedule, coach information, pricing, and a special “OSS” button that triggers a fun blood animation 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 (sarran-dojo.duckdns.org).

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.

2. 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:



Instances (1/1) Informations

Name	ID d'instance	État de l'insta…	Type d'insta…	Contrôle des statu	Statut d'alarm	Zone de dispon…	DNS IPv4 public	Adresse IPv4…	IP élastique
sarran_dojo	i-00a5067a20edeea7d	En cours d'…	t2.micro	2/2 vérifications r	Afficher les alarn	ap-southeast-2a	ec2-3-106-213-9.ap-so…	3.106.213.9	–

i-00a5067a20edeea7d (sarran_dojo)

Résumé de l'instance Informations

ID d'instance
i-00a5067a20edeea7d

Adresse IPv4 publique
3.106.213.9 | adresse ouverte

Adresses IPv4 privées
172.31.2.7

Adresse IPv6
–

État de l'instance
En cours d'exécution

DNS public
ec2-3-106-213-9.ap-southeast-2.compute.amazonaws.com | adresse ouverte

Type de nom d'hôte
Nom de l'adresse IP: ip-172-31-2-7.ap-southeast-2.compute.internal

Nom DNS de l'IP privé (IPv4 uniquement)
ip-172-31-2-7.ap-southeast-2.compute.internal

Réponse à un nom DNS de ressource privée
IPv4 (A)

Type d'instance
t2.micro

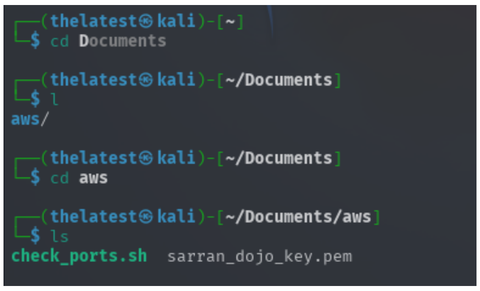
Adresses IP élastiques
–

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.

3. 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:

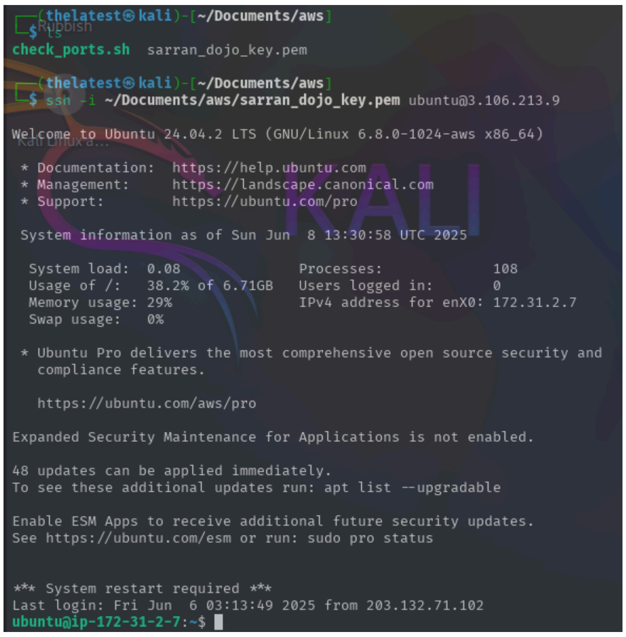


```
┌──(thelatest㉿kali)-[~]
└─$ cd Documents

┌──(thelatest㉿kali)-[~/Documents]
└─$ l
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]
└─$ ls
check_ports.sh  sarran_dojo_key.pem

┌──(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:~$ █
```

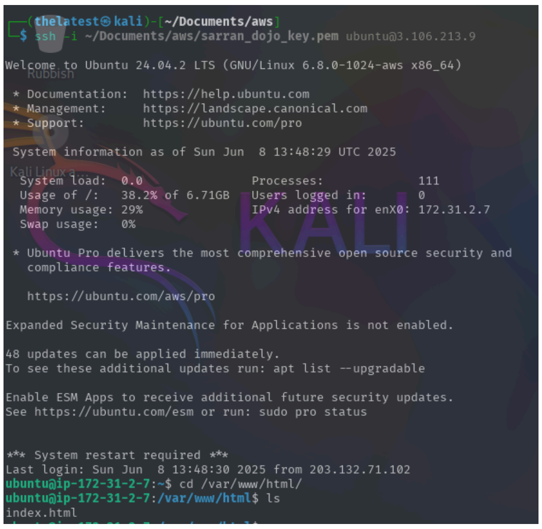
I am connected to the serveur now

4. 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:



```
┌──(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 13:48:29 UTC 2025

  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
```

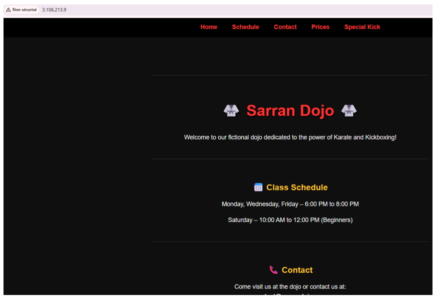
We saw the index.html file ready to be seen !

5. 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 confirming 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 took the domain sarran-dojo.duckdns.org from the Duck DNS service. Then, I logged into my Duck DNS account and went to the DNS management section. There, I pointed sarran-dojo.duckdns.org to my server's public IP address (3.106.213.9). It took 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 sarran-dojo.duckdns.org.

Screenshot:



Sarran Dojo

Welcome to our fictional dojo dedicated to the power of Karate and Kickboxing!

Class Schedule

Monday, Wednesday, Friday – 6:00 PM to 8:00 PM

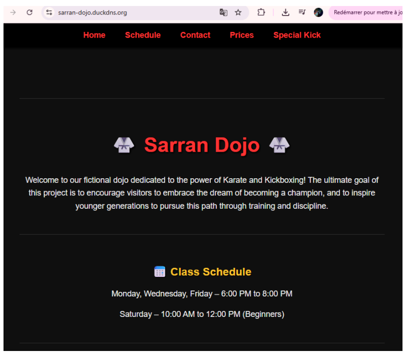
Saturday – 10:00 AM to 12:00 PM (Beginners)

Contact

Come visit us at the dojo or contact us at:

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

Screenshot:



Sarran Dojo

Welcome to our fictional dojo dedicated to the power of Karate and Kickboxing! The ultimate goal of this project is to encourage visitors to embrace the dream of becoming a champion, and to inspire younger generations to pursue this path through training and discipline.

Class Schedule

Monday, Wednesday, Friday – 6:00 PM to 8:00 PM

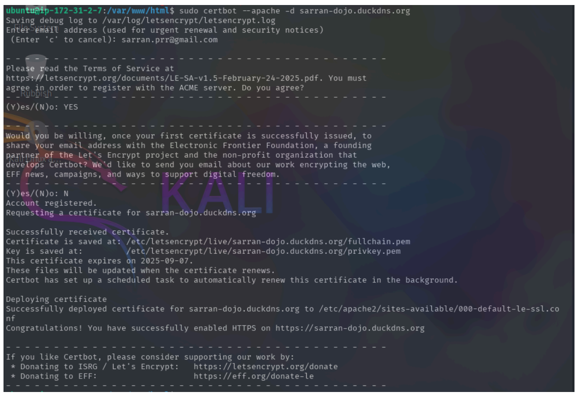
Saturday – 10:00 AM to 12:00 PM (Beginners)

I tested my website in the browser using sarran-dojo.duckdns.org, *it's worked*

6. HTTPS with Let's Encrypt

After making sure my website was working with the domain name, I wanted to secure it using HTTPS. I used Let's Encrypt with Certbot to generate a free SSL/TLS certificate

Screenshot:



```
ubuntu@ip-172-31-2-7:/var/www/html$ sudo certbot --apache -d sarran-dojo.duckdns.org
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Enter email address (used for urgent renewal and security notices)
 (Enter 'c' to cancel): sarran.prr@gmail.com

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.5-February-24-2025.pdf. You must
agree in order to register with the ACME server. Do you agree?
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: YES

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Would you be willing, once your first certificate is successfully issued, to
share your email address with the Electronic Frontier Foundation, a founding
partner of the Let's Encrypt project and the non-profit organization that
develops Certbot? We'd like to send you email about our work encrypting the web,
EFF news, campaigns, and ways to support digital freedom.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: N
Account registered.
Requesting a certificate for sarran-dojo.duckdns.org

Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/sarran-dojo.duckdns.org/fullchain.pem
Key is saved at:         /etc/letsencrypt/live/sarran-dojo.duckdns.org/privkey.pem
This certificate expires on 2025-09-07.
These files will be updated when the certificate renews.
Certbot has set up a scheduled task to automatically renew this certificate in the background.

Deploying certificate
Successfully deployed certificate for sarran-dojo.duckdns.org to /etc/apache2/sites-available/000-default-le-ssl.co
nf
Congratulations! You have successfully enabled HTTPS on https://sarran-dojo.duckdns.org

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

I *used Certbot to generate an HTTPS certificate for my domain. Certbot configured Apache automatically and confirmed that HTTPS is now active for my website.*

My website is now accessible securely at [https://sarran-dojo.duckdns.org/]

7. 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:

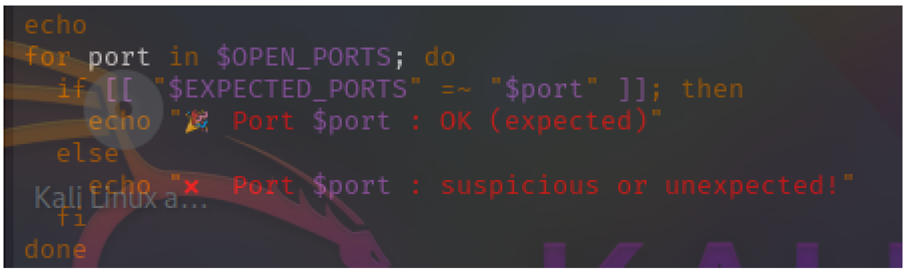
```
EXPECTED_PORTS="22 80 443"
```

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.

Finally, the script compares the list of open ports to the list of expected ports. For each open port, it checks if it's expected or not:

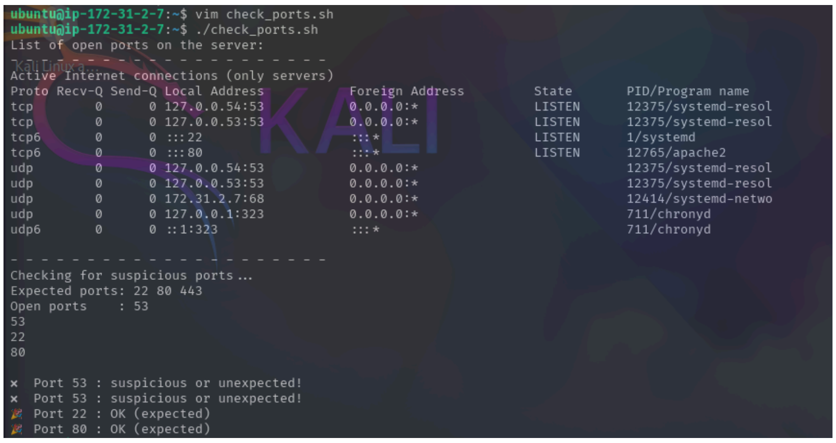


```
echo
for port in $OPEN_PORTS; do
  if [[ "$EXPECTED_PORTS" =~ "$port" ]]; then
    echo "🎉 Port $port : OK (expected)"
  else
    echo "❌ Port $port : suspicious or unexpected!"
  fi
done
```

If the port is in the expected list, it prints that it's OK. Otherwise, it prints a warning message.

Output of the script:

Screenshot:



```
ubuntu@ip-172-31-2-7:~$ vim check_ports.sh
ubuntu@ip-172-31-2-7:~$ ./check_ports.sh
List of open ports on the server:
- - - - - - - - - - - - - - - - - - - - -
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.54:53           0.0.0.0:*               LISTEN      12375/systemd-resol
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      12375/systemd-resol
tcp6       0      0 :::22                   :::*                    LISTEN      1/systemd
tcp6       0      0 :::80                   :::*                    LISTEN      12765/apache2
udp        0      0 127.0.0.54:53           0.0.0.0:*                           12375/systemd-resol
udp        0      0 127.0.0.53:53           0.0.0.0:*                           12375/systemd-resol
udp        0      0 172.31.2.7:68           0.0.0.0:*                           12414/systemd-netwo
udp        0      0 127.0.0.1:323           0.0.0.0:*                           711/chronyd
udp6       0      0 ::1:323                 :::*                                711/chronyd

- - - - - - - - - - - - - - - - - - - - -
Checking for suspicious ports ...
Expected ports: 22 80 443
Open ports    : 53
53
22
80

× Port 53 : suspicious or unexpected!
× Port 53 : suspicious or unexpected!
🎉 Port 22 : OK (expected)
🎉 Port 80 : OK (expected)
```

Here in the screenshot, we see the output of my Bash script. First, it shows all the open ports on my server. We can see the protocol (TCP or UDP), the port number, 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 ports 22 (for SSH), 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 ports 22 and 80 are OK because they are expected. But it also found port 53, which is not in my list, so it says suspicious or unexpected.

This simple script is useful to make sure my server is secure and doesn't have any surprise open ports.

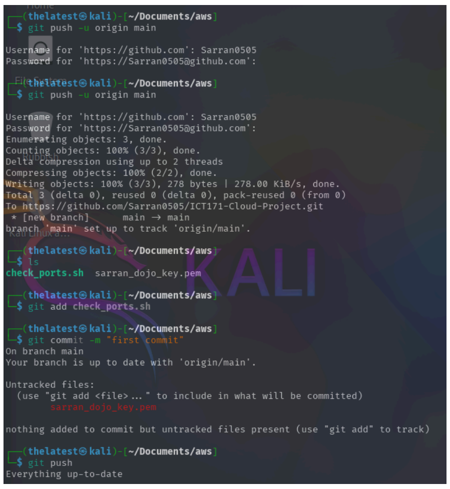
8. License

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 (sarran-dojo.duckdns.org).

9. 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:

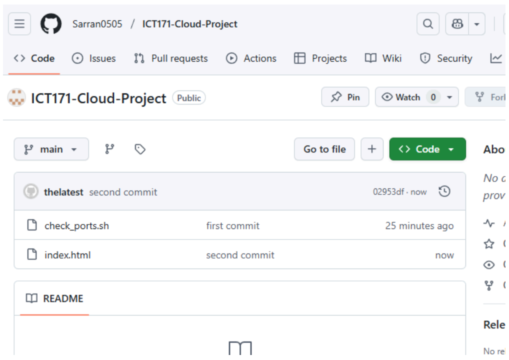
Screenshot:



```
┌──(thelatest㉿kali)-[~/Documents/aws]
└─$ git push -u origin main

Username for 'https://github.com': Sarran0505
Password for 'https://Sarran0505@github.com':

┌──(thelatest㉿kali)-[~/Documents/aws]
└─$ git push -u 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.
Writing 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:



Sarran0505 / ICT171-Cloud-Project

ICT171-Cloud-Project Public

main

thelatest second commit		02953df · now
check_ports.sh	first commit	25 minutes ago
index.html	second commit	now

README

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]

10. Video link :

https://drive.google.com/file/d/1m9SwzuJXxsSBJHoogHwHYZjeVbZgHT4a/view?usp=drivesdk

11. 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

12. References

AWS documentation: https://docs.aws.amazon.com
Apache2 documentation: https://httpd.apache.org/docs
Duck DNS (for DNS setup): https://www.duckdns.org/domains
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