Louis Poustis Medou Beigbeder
n¨35609412

Link to github :
https://github.com/LouisPoustis21/Assigment2_ICT171_35609412.git

Link to the video :

https://youtu.be/1RG65QCbERE

1. Cloud Server Creation (IaaS)

Provider

- Chosen platform: AWS EC2 (Amazon Web Services)

Instance configuration

- Image (AMI): Ubuntu Server 22.04 LTS
- Instance type: t2.micro (Free Tier eligible)
- Region: Asia Pacific (Sydney)
- Key pair: Cleassig1.pem (downloaded locally)

Security Group

- Inbound Rules:
    - TCP port 22 (SSH) — only for my IP
    - TCP port 80 (HTTP) — for public access to the site
    - TCP port 443 (HTTPS) — for secure web access
- Outbound Rules: allow all (default)

Storage

- 8 GB EBS volume

Initial setup steps

1. Connected to the server using:

ssh -i Cleassig1.pem ubuntu@54.253.220.10

2. Performed system update:

sudo apt update && sudo apt upgrade

3. Installed Apache web server:

sudo apt install apache2

Apache Server Monitoring and Auto-Restart

Objective:

Set up a system that automatically monitors the Apache2 service on my cloud server. If the service crashes or stops unexpectedly, a Bash script detects the issue and restarts the service every 30 seconds.

1. Monitoring Script: watchdog.sh

```
#!/bin/bash


SERVICE="apache2"   # Name of the service to monitor
DELAY=30            # Delay between each check


while true; do
    if ! systemctl is-active --quiet $SERVICE; then
        echo "$(date) - $SERVICE is down. Restarting..." >>
/var/log/service_watchdog.log
        systemctl restart $SERVICE
    fi
    sleep $DELAY
done
```

The script is located at:

/home/ubuntu/watchdog.sh

It checks the status of the Apache service every 30 seconds. If Apache is found to be inactive, it is automatically restarted, and a timestamped log entry is written to:

/var/log/service_watchdog.log

2. Make the Script Executable

```
chmod +x /home/ubuntu/watchdog.sh
```

This command makes the script executable so it can be launched by systemd.

3. Create the systemd Service

Create the service file:

sudo nano /etc/systemd/system/watchdog.service

Content of the file:

```
[Unit]
Description=Watchdog - Restart Apache if down
After=network.target

[Service]
ExecStart=/home/ubuntu/watchdog.sh
Restart=always
User=root
StandardOutput=syslog
StandardError=syslog

[Install]
WantedBy=multi-user.target
```

Explanation:

- ExecStart: Path to the script to execute
- Restart=always: Automatically restarts the service if it fails
- User=root: Runs the script with root privileges (needed to restart Apache)
- WantedBy=multi-user.target: Ensures the service starts automatically on boot

4. Enable and Start the Service

Reload systemd after adding the service:

sudo systemctl daemon-reexec

sudo systemctl daemon-reload

Enable the service to run at startup:

sudo systemctl enable watchdog.service

Start the service immediately:

sudo systemctl start watchdog.service

5. Test the Service

Check the service status:

sudo systemctl status watchdog.service

Expected result:

Active: active (running)

To simulate a crash:

sudo systemctl stop apache2

The script will detect the stopped service and restart it automatically within 30 seconds.

6. Log File

All restart events are logged in:

/var/log/service_watchdog.log

Example output:

Thu Jun 5 18:31:42 UTC 2025 - apache2 is down. Restarting...

Free Domain Name Configuration with DuckDNS

Objective:

Associate a free domain name (assigment2ict17135609412.duckdns.org) with my web server so it can be accessed via a human-readable URL, and prepare for the addition of HTTPS support.

1. DuckDNS Domain Creation

- Website used: https://www.duckdns.org
- Authentication method: GitHub account login
- Domain name selected: assigment2ict17135609412
- Full domain: http://assigment2ict17135609412.duckdns.org

2. Linking the Domain to My Public IP Address

Retrieving the server's public IP:

curl ifconfig.me

Result: 54.253.220.10

Manually updating the domain from the server:

Curl "https://www.duckdns.org/update?domains=assigment2ict17135609412&token=YOUR_DUCKDNS_TOKEN&ip=54.253.220.10"

Replace YOUR_DUCKDNS_TOKEN with the token shown on your DuckDNS account page.

Expected result: OK

3. Verification

Open a browser and access:

http://assigment2ict17135609412.duckdns.org

- The site loads correctly and matches the content served at the raw IP address.
- DNS resolution is confirmed to be working.

4. Preparing for HTTPS Activation (Let's Encrypt)

Once the domain is active, I prepared to install a free SSL certificate using Certbot.

Installing Certbot for Apache:

sudo apt update

sudo apt install certbot python3-certbot-apache

Command to request the certificate:

sudo certbot --apache -d assigment2ict17135609412.duckdns.org

Expected Result

- Website accessible at:
  http://assigment2ict17135609412.duckdns.org
  https://assigment2ict17135609412.duckdns.org (after certificate is installed)
- No manual DNS intervention required unless the server's IP changes.
- Domain is free, valid, and suitable for use in the assignment.