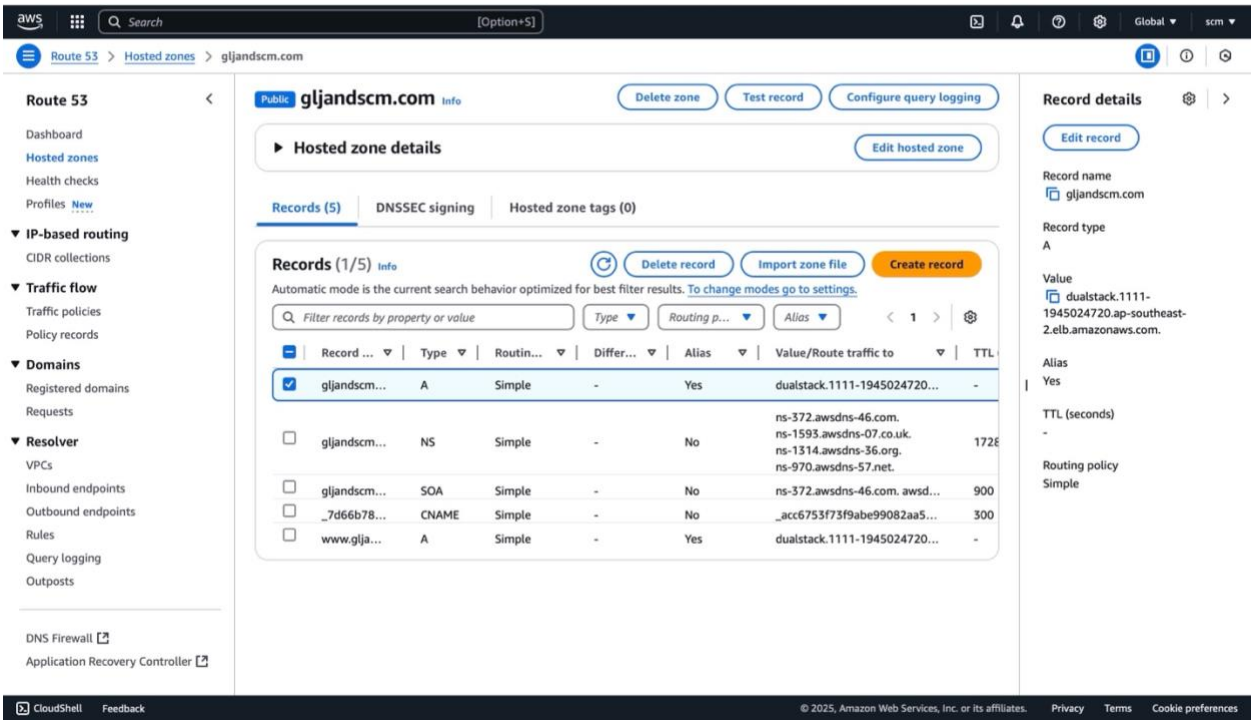
ICT 171 server configuration and Script Deployment Process of backup project

Name :Chenming Suo
Student number: 35014152
Ipv4 address: 13.237.110.154
Git hub link:
Web:gljandscm.com
video: https://youtu.be/D4nGLkzJtwQ



Part 1

1. Make the instances
    a. Login to the AWS Management Console
        i. Open AWS as first then open EC2. aws webside
        ii. Login by account and password.
    b. Make the E2 instance
        i. Launch instances:
            1. Search “EC2” and open it.
            2. Click the “Launch instances”
        ii. Select image:

1. Choose “Ubuntu server 20.04 LTS” (Or another Linux OS)

iii. Select Instance Type:
   1. Choose t3.micro (t2.micro)
      a. I suggest choosing t3.micro because it can use “EC2 serial console” when there is problem with ssh connection, it can use is to connect the instance.

iv. Configure Instance:
   1. Keep the default settings
   2. Storage: 8GB by default (can be changed based on the project)

v. Select key pair:
   1. Create a new key pair with ED25519 (more security) or RAS
   2. Choose .pem file format (AWS standard format)
   3. Move the key pair file to a security location. Such as (~/Downloads/)

vi. Configure security group
   1. Great a new security group (such as 171-project-backup)
   2. Add new rules:
      a. SSH port 22
      b. HTTP port 80(website serve)
      c. HTTPS port 443 (website serve)

      d. Click "Launch Instance"

vii. Elastic IP
   1. In the left-hand menu of EC2, select network & security then Elastic IP.

   2. Click Allocate Elastic IP Address
      a. Choose “Amazon's pool of IPv4 addresses”
      b. Click the Allocate button
   3. Find the newly allocated Elastic IP in the list.
      a. Click Operation then Associate Elastic IP Address
      b. Choose the instance already made
      c. Click relevance.

   After this, the instance public ipv4 address will change to the Elastic address.

Such as my instance elastic address is 13.237.110.154

2. Local environment preparation (mac OS)
    a. Save the key pair
        i. Open the local terminal
            1. mkdir -p ~/.ssh
            2. chmod 700 ~/.ssh// Create the .ssh directory

        ii. Move the key pair to .ssh directory

```
mv~/"your_key_pair_location"/"your_key_pair_name ~/.ssh/
```

such as

```
mv ~/Downloads/ec2-ubuntu-key.pem ~/.ssh/
```

```
chmod 400 ~/.ssh/your_key_pair_name.pem
chmod 400 ~/.ssh/ec2-ubuntu-key.pem
(set correct key permissions)
```

Try use ssh to connect:

```
ssh -i ~/.ssh/your_key_pair_name.pem ubuntu@"your_address"

ssh -i ~/.ssh/ec2-ubuntu-key.pem ubuntu@13.237.110.154
```

if it is the first time to connect it will show that:

```
Are you sure you want to continue connecting (yes/no/[fingerprint])? Answer: yes
```

3. script description (the script link)
    a. this documentation focus on the Bash script, designed to backing up the data from a local directory to a cloud serve.
    b. Configuration section

```
SOURCE_DIR="/var/www/html"   # Test data directory
SOURCE_DIR="/var/www/html"
BACKUP_DIR="$HOME/Documents/ICT171_Backup_Project/backups"
LOG_FILE="$BACKUP_DIR/backup.log"
ERROR_LOG="$BACKUP_DIR/error.log"
```

        1. SOURCE_DIR, define the path to the local directory containing the data to be back up.
        2. BACKUP_DIR, the location of the backup files
        3. CLOUD_USER, the username used to log in to the cloud serve.
        4. LOG_FILE and ERROR_LOG, define the paths to the log files.
    c. Creating the backup file name

```
mkdir -p "$BACKUP_DIR"
```

d. Check the source source directory

```
if [ ! -d "$SOURCE_DIR" ]; then
   echo "[$(date)] ERROR: can not find file: $SOURCE_DIR" >> "$ERROR_LOG"  #
    exit 1
fi
```

e. Get the time

```
now=$(date +%Y%m%d%H%M%S)
ZIP_FILE="$BACKUP_DIR/${now}_backup.zip"
```

date +%Y%m%d%H%M%S used to create the file name with year, month, day, hour, minute, and second.

f. Performing the Backup

```
# Fix: Directly zip the directory instead of its contents
echo "[$(date)] START TO BACKUP" >> "$LOG_FILE"
zip -rq "$ZIP_FILE" "$SOURCE_DIR" 2>> "$ERROR_LOG"
```

first use zip -rq command to compress the $SOURCE_DIR directory into the $ZIP_FILE archive.

g. checks the backup result and upload

```
if [ $? -eq 0 ]; then
    echo "[$(date)] SUCCESSFUL: Backup saved to $ZIP_FILE" >> "$LOG_FILE"
else
    echo "[$(date)] ZIP COMMAND FAILED: CHECK $ERROR_LOG" >> "$LOG_FILE"
fi
set +x
```

4. Use SCP command to upload the script

```
scp -i ~/.ssh/your_key_pair.pem ~your_script_location serve_username@ip:~/scripts/
```

such as :

```
scp -i ~/.ssh/ec2-ubuntu-key.pem ~/Documents/ICT171_Backup_Project/backup_script.sh
ubuntu@13.237.110.154:~/scripts/
```

5. Check the script

```
ls -l ~/scripts/backup_script.sh
```

6. Create the deploy user on the serve
   a. Login your serve by username “ubuntu”

```
sudo adduser deploy
sudo usermod -aG sudo deploy
```

use those two commend to create a user “deploy”

b. copy the ssh key pair to the deploy

```
sudo mkdir -p /home/deploy/.ssh
sudo cp ~/.ssh/authorized_keys /home/deploy/.ssh/
sudo chown -R deploy:deploy /home/deploy/.ssh
sudo chmod 700 /home/deploy/.ssh
sudo chmod 600 /home/deploy/.ssh/authorized_keys
```

c. use the same commend to login user deploy

```
ssh -i ~/.ssh/ec2-ubuntu-key.pem deploy@13.237.110.154
```

7. move the script to the executable directory

```
sudo mv ~/scripts/backup_script.sh /usr/local/bin/backup_script.sh
chmod +x ~/scripts/backup_script.sh
```

8. check the script

```
ls -l ~/scripts/backup_script.sh
```

9. run it

```
~/scripts/backup_script.sh
```

10. Scheduled task configuration

```
crontab -e
```

add the script to the crontab

```
0 * * * * ~/scripts/backup_script.sh
```

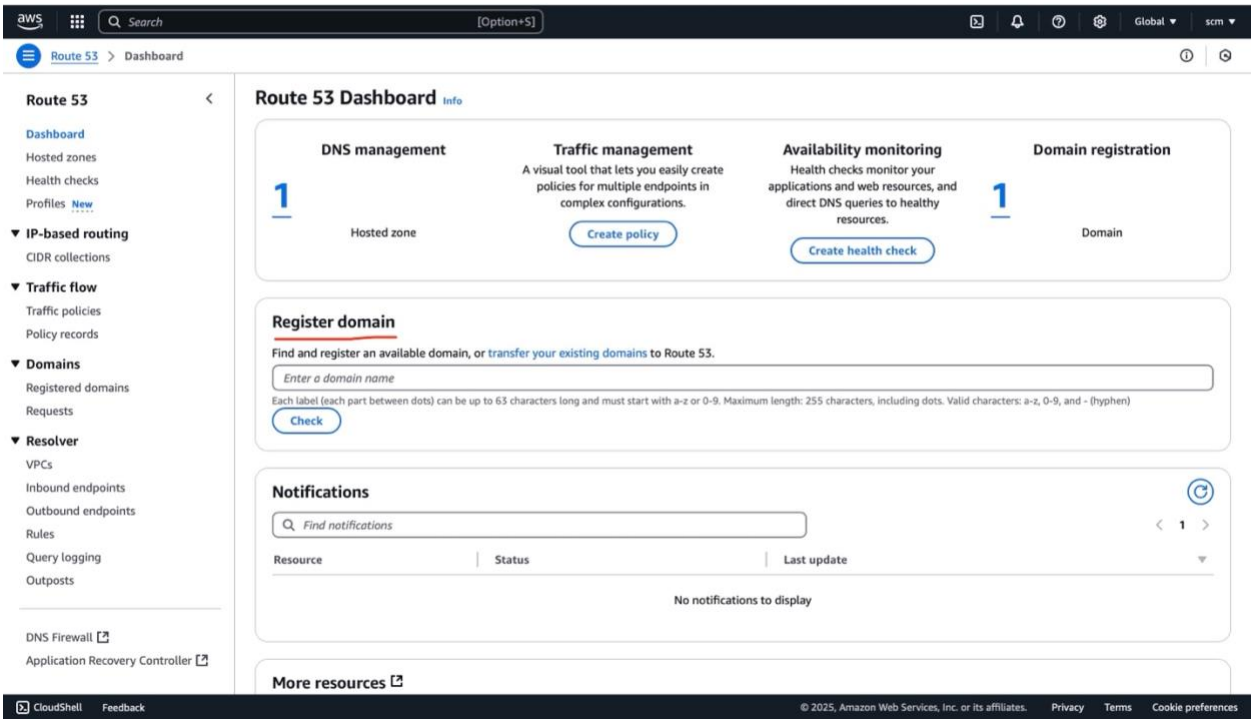
11. Set the firewall

```
sudo ufw allow 22/tcp  # SSH
sudo ufw allow 80/tcp  # HTTP
sudo ufw allow 443/tcp # https
sudo ufw enable
sudo ufw status
```

tips: if you close the port 22 which is ssh, you may not can connect your instance, so you can go to instances page right click your instance choose “Monitor and troubleshoot” → “EC2 serial console “then you can use your username and password to change the setting without ssh connection.

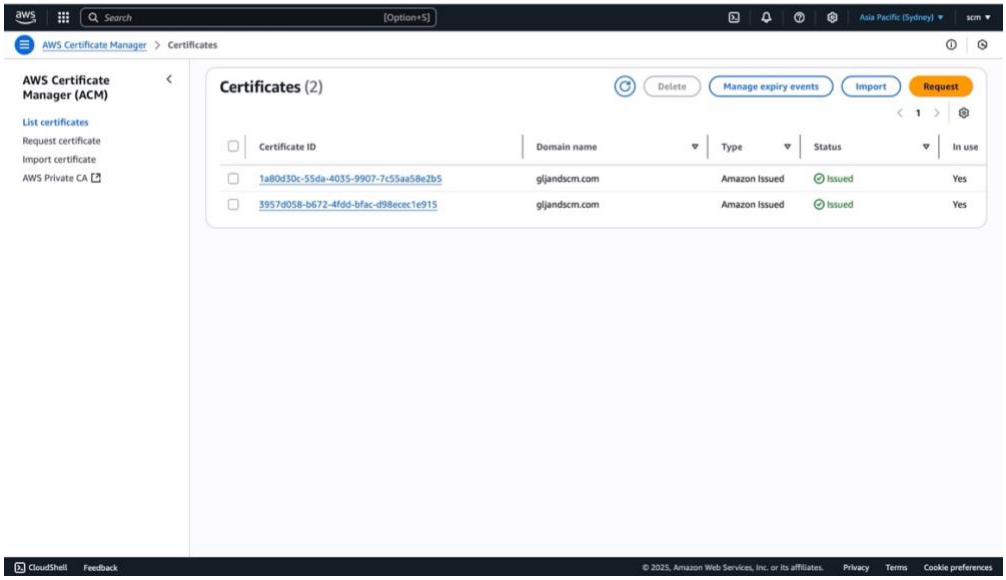
Part 2 – DNS, HTTPS

1. Search and go to Route 53, click “Dashboard” on the left hand, search the domain name after “Register domain” then check it, then you can find anyone you want to use then buy it.



After this you can find the domain name in the “Hosted zones”.

2. Request a public SSL/TLS certificate
    a. Search AWS Certificate Manager, click “List certificates” on the left hand.
    b. Choose Request batten, then input the “domain names” you just bought, after this you can find the SSL certificate in “List certificates”.
    c. Check if the CNAME record is automatically added in Route 53.

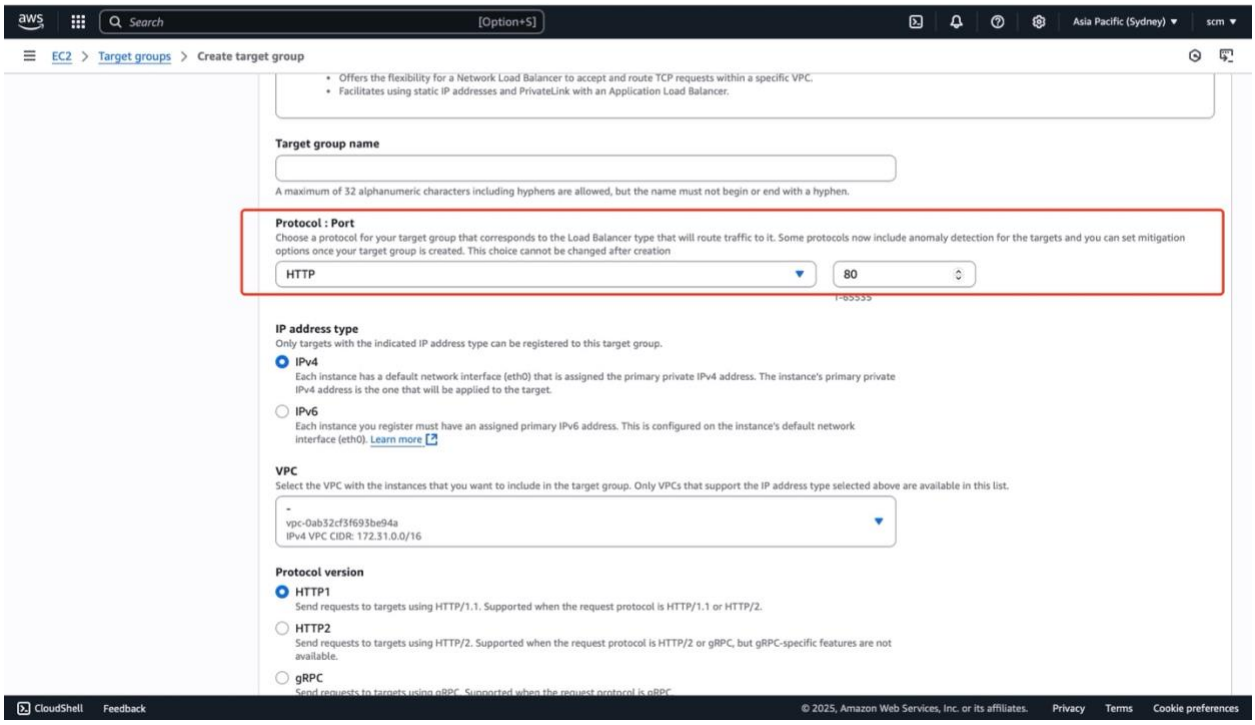


3. Use the load balancers to implement HTTPS connection for the website

   Find the "Load Balancers" in EC2 page, then chose "create load balancer"→ chose to create Application Load Balancer, in the setting page,

   1. make sure subnet include your instance subnet.
   2. The security group need to include the allow port 443 with inbound rules and allow port 80 with outbound.
   3. The Listeners and routing, the protocol should be HTTPS, the target group need to make with the instances type, then choose your instance (make sure the protocol port is http).
   4. For Default SSL/TLS server certificate part, choose the certificate you just applied.
   5. Great the load balance.
   6. Check the load balance to make sure the state is active and

      Tips: check the port in Target groups is http, this is connect with instance.



4. Add the records
   a. Search Route 53 then choose Hosted zones, choose the Hosted zone name, then chose create record with A type, then turn on the alias then choose

“alias to application and classic load balancer” then choose the load balancer you just made.

5. After finished these things you can use https to visit your website.