Challenge 1 :

We want you to automate the deployment of a simple web application (e.g. WordPress site) in a cloud provider of your choice. We would like to see the infrastructure choices you have provisioned to host such application using IaC of your choice and the way you automate the deployment of both your infrastructure and application. Make some notes on how you would further "Productionize" such a solution.

Brownie points if the application is running.

Stages:

1) **Choosing a web application:** For the project, a python web application is chosen which will connect to dynamo DB and expose some APIs.
2) **Choosing a cloud:** For the project, AWS is used for infrastructure choice.
3) **Infrastructure as Code:** For IaC, terraform is used to deploy the infrastructure and application in AWS Cloud.
4) **Automation:** For automation, Jenkins is used to deploy the infrastructure.
5) **Notes on "Productionize":** A detailed note is written about how to productionize the application.

Steps:

1) A terraform template is prepared to deploy the application and the infrastructure. (template.tf)
   The terraform script (template.tf) will create below resources in AWS:

    1) AWS VPC with two public subnets in different availability zone
        a) VPC
        b) Subnets
        c) Route Tables
    2) One AWS IAM Role with AWS IAM policy having access of Amazon DynamoDB
        a) Policy
        b) Role
        c) Instance Profile
        d) User Data for Server
    3) Two EC2 instances with above-mentioned role
        a) Instances
        b) Security Group
        c) Role attachment
    4) One DynamoDB Instance
    5) One Network Load Balancer to access the application using Load balancer
        a) Listener
    6) Target group containing the two instances
        a) Health checks

2) Create an AWS access key and secret key in the target AWS Account (Admin Access). The access key and secret key will be used during terraform deployment.

3) Configure AWS CLI in workstation. Execute below command to setup AWS CLI in the workstation.

```
aws configure
```

Use the appropriate Access key and Secret Key.

4) Install Terraform in workstation

5) Download/Copy template.tf terraform script in the workstation to create resources from the template

6) Use below command in the workstation to initialize a working directory containing Terraform configuration files.

```
terraform init
```

The terraform init command initializes a working directory containing Terraform configuration files. This is the first command that should be run after writing a new Terraform configuration or cloning an existing one from version control. It is safe to run this command multiple times.

7) Preview the action by using following command

```
terraform plan
```

The terraform plan command lets you preview the actions Terraform would take to modify your infrastructure or save a speculative plan which you can apply later. The function of terraform plan is speculative: you cannot apply it unless you save its contents and pass them to a terraform apply command.

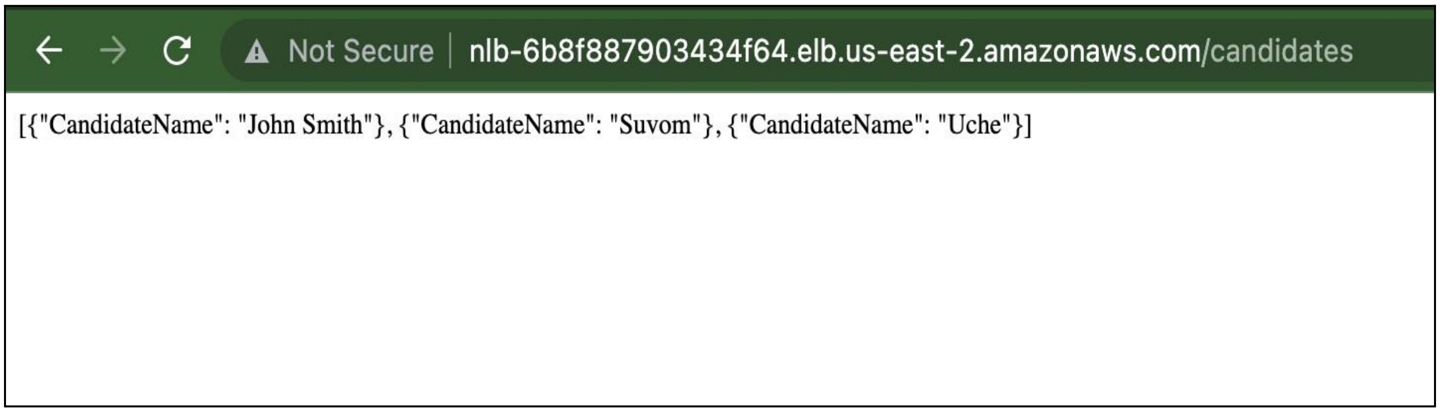
8) Deploy the infrastructure

```
terraform apply
```

terraform apply — Creates or updates infrastructure depending on the configuration files. By default, a plan will be generated first and will need to be approved before it is applied. terraform apply -auto-approve — Apply changes without having to interactively type 'yes' to the plan.

9) The application can be accessed via load balancer once deployment is done.
    1) Application is deployed in linux system (mentioned in template.tf)
    2) Both application server is in different zone to maintain high availability
    3) All the routes mentioned in the application repository is available
    4) The load balancer server the app over standard HTTP (80)

       protocol Refer to the screenshot below:



```
[{"CandidateName": "John Smith"}, {"CandidateName": "Suvom"}, {"CandidateName": "Uche"}]
```

CI/CD

A CI/CD pipeline is created using Jenkins to create the infrastructure:

Following is the instruction to setup a Jenkins server:

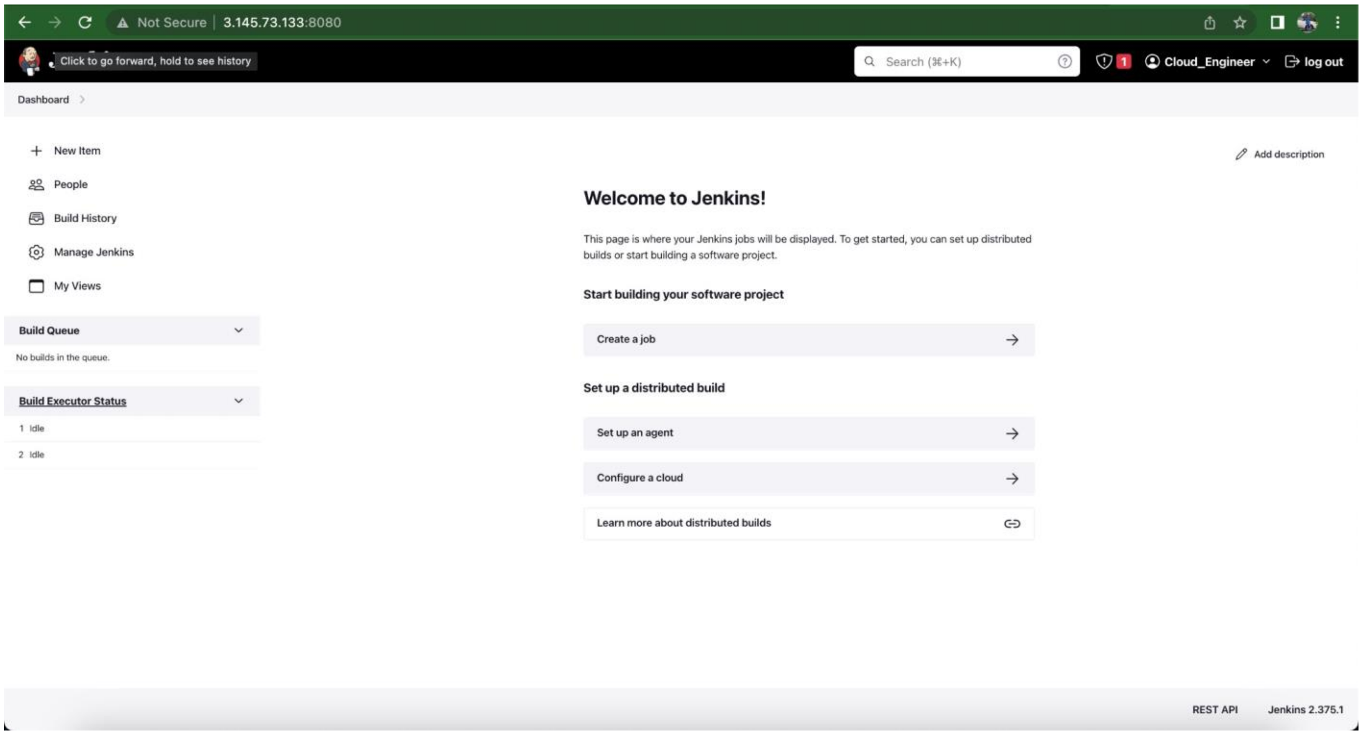
```
1) mkdir ~/install_jenkins
2) cd ~/install_jenkins
3) sudo apt install default-jdk
4) sudo apt update
5) wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
6) sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > \
7)     /etc/apt/sources.list.d/jenkins.list'
8) sudo apt-get update
9) sudo apt install jenkins -y
```

After performing above-mentioned steps Jenkins will be running on the server.

You can access Jenkins using below information:

```
Jenkins URL: http://3.145.73.133:8080/
Username: Cloud_Engineer
Password: **************
```

Refer to the screenshot below:



Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Build Queue

No builds in the queue.

Build Executor Status

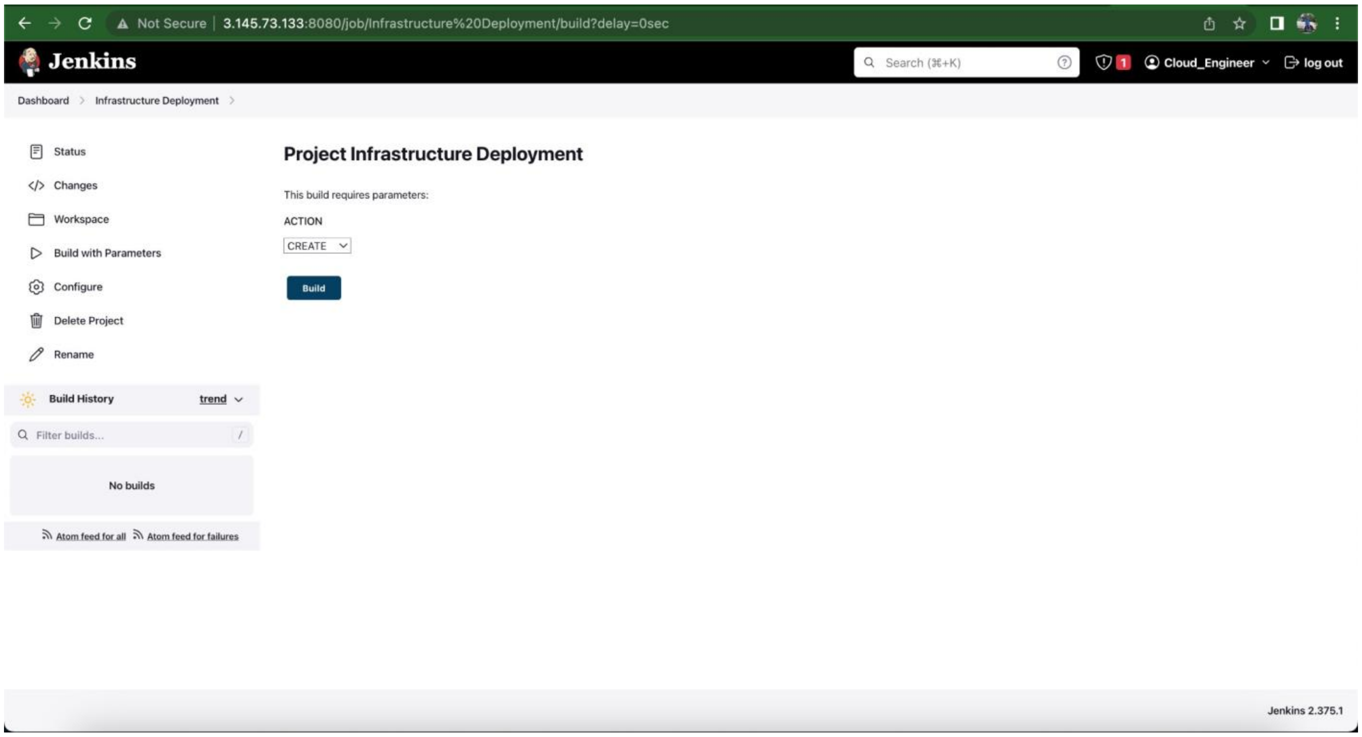
1 Idle

2 Idle

A Jenkins Job is created to deploy the Terraform Script in AWS.

This is a parameterized job which will create or destroy the infrastructure as per user input.

Refer to the screenshot below:



Project Infrastructure Deployment

This build requires parameters:

ACTION

CREATE

Build

Creation of Jenkins Job for Infrastructure Deployment:

Following are the steps to create a Jenkins Job:

1) Click on **New Item**
2) Create a **Freestyle Project** with an **Item Name**
3) **Click on This project is parameterized** option to provide parameter for the project (ACTION)
4) In the **Build Step,** choose **Execute Shell** to create a shell script for the deployment.
5) Add code in shell (bash)

```
cp /var/lib/jenkins/TF/template.tf .
if [ "$ACTION" = "CREATE" ]
then
terraform init
terraform plan
terraform apply -auto-approve
else
terraform destroy -auto-approve
fi
```

Above script will copy the terraform script from a location specified in Jenkins Server.

Then, as per the parameter of **ACTION** in the job, the script will create or delete the infrastructure.

Optional (Productionize)

- **The application should start when the instances start**

To start the application on instance, start up, we need to add a script in the systemd file in the linux server.

Following are the steps for that:

1) Navigate to **/etc/systemd/system/** folder in the Linux machine.
2) Create a new file named **flask.service**
3) Add below mentioned script in the flask.service file

```
[Unit]
Description=flask_service

[Service]
ExecStart=/bin/bash -c 'cd /home/ubuntu/flask/2022-Challenge-/ &&
gunicorn -b 0.0.0.0 app:candidates_app'
ExecStartPost=/bin/bash -c 'cd /home/ubuntu/flask/2022-Challenge-/ &&
python3 test_candidates.py'
ExecStop=
Type=forking
TimeoutStartSec=300
TimeoutStopSec=300

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

4) After that, we have to enable the script by using following command:

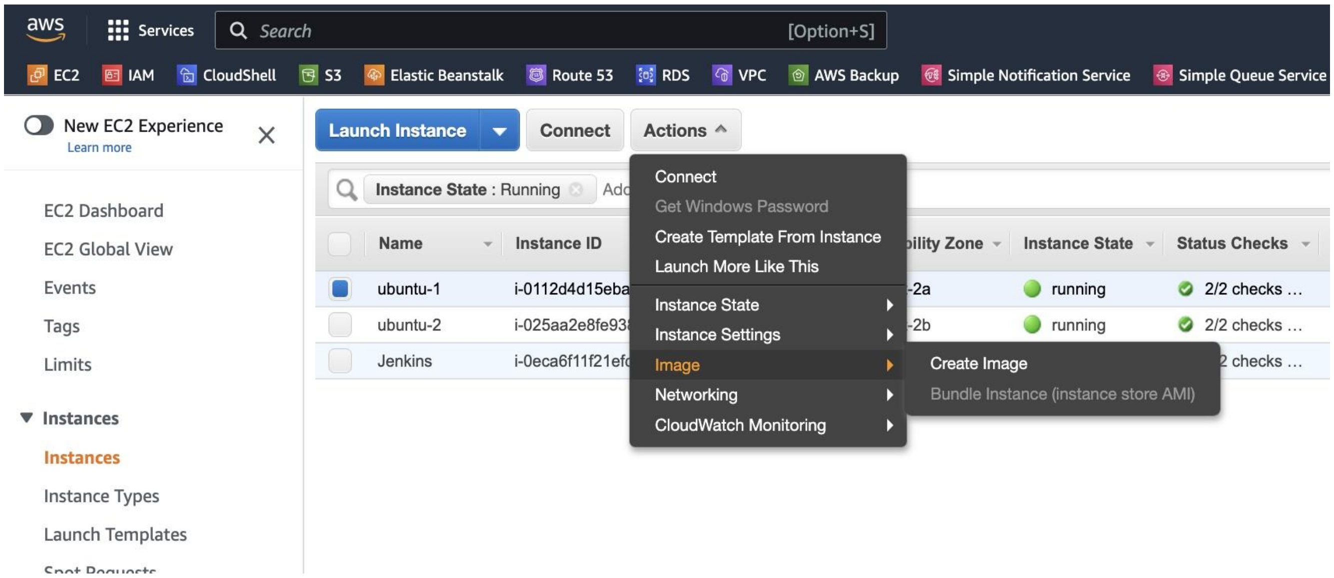
```
sudo systemctl enable flask.service
```

5) After that, the service will automatically start once an instance is started.
6) We can add the above lines using terraform template as well, for that we must add a few extra lines in the user-data section of instances.

```
  touch /etc/systemd/system/flask.service

  echo "[Unit]" >> /etc/systemd/system/flask.service
  echo "Description=flask_service" >> /etc/systemd/system/flask.service
  echo "[Service]" >> /etc/systemd/system/flask.service
  echo "ExecStart=/bin/bash -c 'cd /home/ubuntu/flask/2022-Challenge-/ &&
gunicorn -b 0.0.0.0 app:candidates_app'" >>
/etc/systemd/system/flask.service
  echo "ExecStartPost=/bin/bash -c 'cd /home/ubuntu/flask/2022-Challenge-/
&& python3 test_candidates.py'" >> /etc/systemd/system/flask.service
  echo "ExecStop=" >> /etc/systemd/system/flask.service
  echo "Type=forking" >> /etc/systemd/system/flask.service
  echo "TimeoutStartSec=300" >> /etc/systemd/system/flask.service
  echo "TimeoutStopSec=300" >> /etc/systemd/system/flask.service

  echo "[Install]" >> /etc/systemd/system/flask.service
  echo "WantedBy=multi-user.target" >> /etc/systemd/system/flask.service

  sudo systemctl enable flask.service
```

- **Pre-build AMIs with the application already loaded**
  We can create a pre-build ami from the existing deployment. The benefit of creating a pre-build AMI is to launch the application whenever we want. This will also help to autoscaling the application.

  Step to create an AMI from the existing infrastructure:
  1) We need to create the infrastructure first to have the source instance running.
  2) Once the instance is up, we can create an image from the console as well as using AWS CLI.
  3) Select the instance, choose Actions -> Image -> Create Image
     Refer to the screenshot below:



Name	Instance ID	ility Zone	Instance State	Status Checks
ubuntu-1	i-0112d4d15eba	-2a	running	2/2 checks ...
ubuntu-2	i-025aa2e8fe93	-2b	running	2/2 checks ...
Jenkins	i-0eca6f11f21efc			2 checks ...

4) Provide a Name for the image and click on create.
5) The process will take around 5 minutes to create the image.
6) Once the image is created you can launch an instance from the image in future.
7) You can also use AWS CLI to create an image

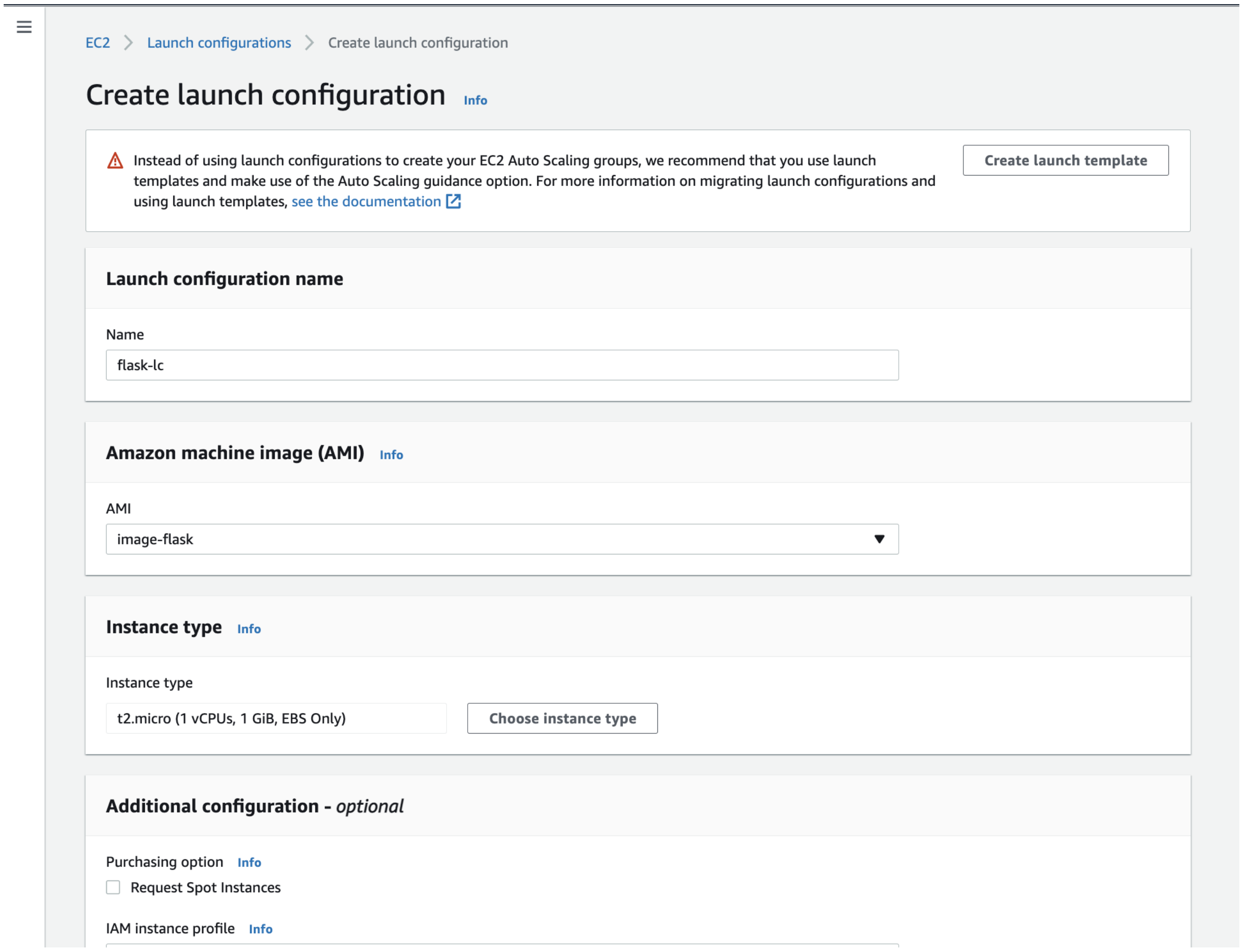
```
aws ec2 create-image --instance-id <instance_id> --name <image_name>
--description <image_deccription>
```

- **Use Autoscaling Groups to help with application deployment**

To deploy the application with autoscaling, following are the steps:

1) Navigate to **Launch Configuration** in EC2 Console.
2) Click on **Create Launch Configuration**
3) Provide appropriate name and AMI which is created in the last step.
4) Provide server configuration as well.

Refer to the screenshot below:



Create launch configuration

Instead of using launch configurations to create your EC2 Auto Scaling groups, we recommend that you use launch templates and make use of the Auto Scaling guidance option. For more information on migrating launch configurations and using launch templates, see the documentation

Launch configuration name

Name: flask-lc

Amazon machine image (AMI)

AMI: image-flask

Instance type

Instance type: t2.micro (1 vCPUs, 1 GiB, EBS Only)

Additional configuration - *optional*

Purchasing option

☐ Request Spot Instances

IAM instance profile

5) Click on **Create Launch Configuration.**
6) Once Launch Configuration is created, navigate to **Auto Scaling Group** in the EC2 console.
7) Provide a name for the group and choose above created Launch Configuration for the autoscaling group.

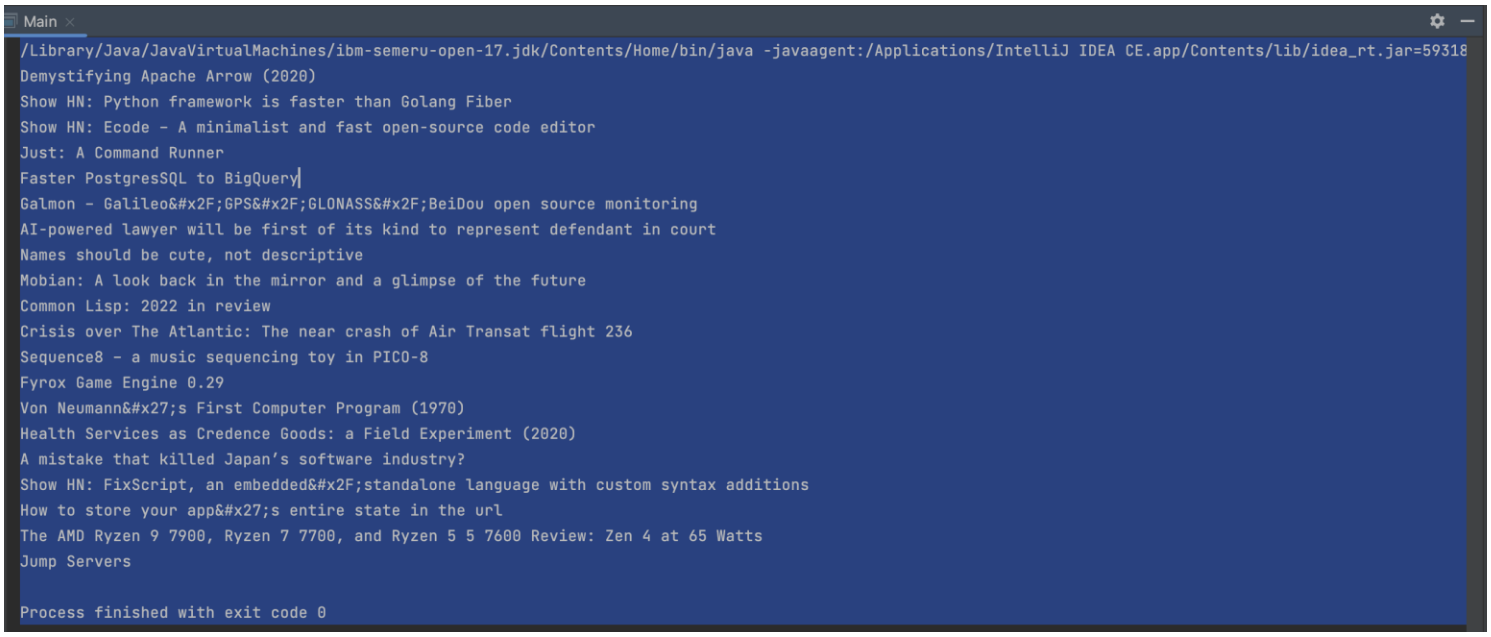
8) Attach the target group created from terraform script to the autoscaling group.
9) Now, we can use an auto scaling group for the application, we can increase the count of servers as per our need. Also, we can provide auto scaling configuration based on CPU load as well.

Challenge 2:
List out the top 20 articles from the Hacker News homepage (https://news.ycombinator.com). The choice of language and implementation is up to you.

A Java Program is written to find top 20 articles from the Hacker News homepage.

```java
import java.util.ArrayList;
import java.util.List;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;

public class Main {
    public static void main(String[] args) {

        String html = getHTML("https://news.ycombinator.com");
        //System.out.println(html);

        //String html = "<html><body><p>Hello, world! <a
href='https://www.example.com'>Link</a></p><a
href='https://www.example.com'>Link2</a></body></html>";

        int count = 0;

        List<String> links = getLinks(html);
        for (String link : links) {
            if(link.contains("https") &&
!link.contains("news.ycombinator.com") && count < 20){
                System.out.println(link.replaceAll("<[^>]*>", ""));
                count++;
            }
        }
    }

    public static String getHTML(String urlString) {
        StringBuilder html = new StringBuilder();
        try {
            URL url = new URL(urlString);
            BufferedReader in = new BufferedReader(new
InputStreamReader(url.openStream()));

            String inputLine;
            while ((inputLine = in.readLine()) != null) {
                html.append(inputLine);
            }
            in.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return html.toString();
    }

    public static List<String> getLinks(String html) {
        List<String> links = new ArrayList<>();
        int startIndex = 0;
        while (startIndex != -1) {
            startIndex = html.indexOf("<a", startIndex);
            if (startIndex != -1) {
                int endIndex = html.indexOf("</a>", startIndex) + 4;
                links.add(html.substring(startIndex, endIndex));
                startIndex = endIndex;
            }
        }
        return links;
    }
}
```

Output :



```
/Library/Java/JavaVirtualMachines/ibm-semeru-open-17.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=59318
Demystifying Apache Arrow (2020)
Show HN: Python framework is faster than Golang Fiber
Show HN: Ecode – A minimalist and fast open-source code editor
Just: A Command Runner
Faster PostgresSQL to BigQuery
Galmon – Galileo&#x2F;GPS&#x2F;GLONASS&#x2F;BeiDou open source monitoring
AI-powered lawyer will be first of its kind to represent defendant in court
Names should be cute, not descriptive
Mobian: A look back in the mirror and a glimpse of the future
Common Lisp: 2022 in review
Crisis over The Atlantic: The near crash of Air Transat flight 236
Sequence8 – a music sequencing toy in PICO-8
Fyrox Game Engine 0.29
Von Neumann&#x27;s First Computer Program (1970)
Health Services as Credence Goods: a Field Experiment (2020)
A mistake that killed Japan’s software industry?
Show HN: FixScript, an embedded&#x2F;standalone language with custom syntax additions
How to store your app&#x27;s entire state in the url
The AMD Ryzen 9 7900, Ryzen 7 7700, and Ryzen 5 5 7600 Review: Zen 4 at 65 Watts
Jump Servers

Process finished with exit code 0
```

Challenge 3:
You have been dropped into a large ecommerce website (which takes payments) and they suspect they are suffering from an ongoing system attack.
Can you please help them by analyzing the attached log file to help the client identify the attack and make recommendations on strengthening their website.
There are potentially many issues in the file, discuss with us your top 3.

Here is a explanation of three issues from the file:

1) 111.76.171.207 - Justin - [2023-01-05 13:25:56 +00:00] "PUT /admin/remove_product HTTP/1.0" 500 4964 "http://gomez.biz/terms.htm" "Mozilla/5.0 (X11; Linux i686 on x86_64; rv:49.0) Gecko/20100101 Firefox/49.0" 1338

   The potential issue in the line provided is the status code 500. This status code indicates an internal server error, which means that there was a problem with the server while processing the request. This could be due to a variety of factors, such as a bug in the server-side code, a problem with the server's configuration, or an issue with the server's hardware. It would be necessary to investigate further to determine the specific cause of the issue.

2) "Mozilla/5.0 (Windows NT 10.0; WOW64; rv:50.0) Gecko/20100101 Firefox/50.0" 994

```
101.14.81.197 - Lowri - [2023-01-05 13:25:56 +00:00] "DELETE /admin HTTP/1.0" 200
4937 "http://gomez.biz/terms.htm" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:51.0)
Gecko/20100101 Firefox/51.0" 3852
```

There doesn't appear to be any issues in the line you provided. The status code 200 indicates that the request was successful, and the rest of the line appears to be properly formatted. It contains information about the user's browser and IP address, the time the request was made, the type of request (DELETE), the path of the requested resource (/admin), and the HTTP version used (HTTP/1.0). The user agent string provides information about the user's browser and operating system. It is worth noting that HTTP/1.0 is an older version of the HTTP protocol and is no longer widely used. It may be worth considering upgrading to a newer version of HTTP if this is relevant to your situation.

3)
```
60.234.181.42 -  - [2023-01-05 13:25:56 +00:00] "PUT /api HTTP/1.0" 403 5099
"http://gomez.biz/terms.htm" "Mozilla/5.0 (Windows NT 6.2; rv:47.0) Gecko/20100101
Firefox/47.0" 3653
```

The potential issue in the line provided is the status code 403. This status code indicates that the server understood the request, but it refuses to authorize it. This means that the server is denying access to the resource that was requested. This could be due to a variety of factors, such as the user not having the necessary permissions to access the resource, the user not being properly authenticated, or the resource being temporarily unavailable. It would be necessary to investigate further to determine the specific cause of the issue. Additionally, the user is missing the name field, this could be missing due to some error in logging or some other reason.