Username - 1032231291@tcetmumbai.in

Password - Nakhru@123

1.On AWC ,create EC 2 instance and java/Python/ C program

Launch an EC2 Instance:

In the search bar, type EC2, and click EC2 under "Services."

Click Launch Instance.

Configure Your Instance:

Choose an Amazon Machine Image (AMI): Select an Ubuntu.

Instance Type: Choose a free-tier eligible type like t2.micro if you're just testing or starting.

Key Pair: no need

Configure Network Settings: Keep the default VPC and subnet or configure as needed. Make sure SSH (port 22) is open in the security group.

Launch the Instance: Once configured, click Launch.

Connect to Your EC2 Instance:

From the EC2 dashboard, select your instance, and click Connect.

Follow the provided SSH connection commands (if using your terminal or Cloud9).

sudo apt update && sudo apt install default-jdk

java -version

vi HelloWorld.java

public class HelloWorld {

public static void main(String[] args) {

System.out.println("Hello, World!");

}

}

javac HelloWorld.java

java HelloWorld

2. Create S3 bucket n upload java program

click on S3

click on create bucket

enter bucket name

directly click on create bucket button

click on your bucket name

click on upload , create a java file , add file ,

3. Terraform- build, plan n destroy instance

Search user navigate to its console

click on create user , enter user name , go on security credentials , create access key , click on application running option , click next , in description write Akshit,

click on create

Test.tf

provider "aws" {

region = "ap-south-1"

access\_key = "AKIAZI2LE3RMVMDUKV5X"

secret\_key = "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"

}

resource "aws\_instance" "ubuntu" {

ami = "ami-0ad21ae1d0696ad58"

instance\_type = "t2.micro"

}

1. Open Command Prompt or PowerShell and run the following commands:

setx AWS\_ACCESS\_KEY\_ID "AKIAZI2LE3RMVMDUKV5X"

setx AWS\_SECRET\_ACCESS\_KEY "b//wcNcwKkqo4o1WERrsVTOAV8fwbzI1Y5Wkc+AZ"

setx AWS\_DEFAULT\_REGION "ap-south-1"

**Step 3: Create Your Terraform Configuration File**

1. **Create a Directory**:
   * Create a new directory for your Terraform project. For example:

mkdir C:\Terraform\my-aws-instance

cd C:\Terraform\my-aws-instance

1. **Create a Terraform File**:
   * Create a new file named main.tf in your project directory and add the following configuration:

provider "aws" {

region = "ap-south-1"

access\_key = "AKIAZI2LE3RMVMDUKV5X"

secret\_key = "b//wcNcwKkqo4o1WERrsVTOAV8fwbzI1Y5Wkc+AZ"

}

resource "aws\_instance" "ubuntu" {

ami = "ami-0ad21ae1d0696ad58"

instance\_type = "t2.micro"

}

**Step 4: Initialize Terraform**

1. **Open Command Prompt or PowerShell** and navigate to your Terraform directory:

cd C:\Terraform\my-aws-instance

1. **Run the Initialization Command**:

terraform init

* + This command downloads the necessary provider plugins required for your configuration.

**Step 5: Plan Your Terraform Deployment**

1. **Create an Execution Plan**:

terraform plan

* + This command shows what Terraform will do when you apply your configuration. Review the output to ensure everything looks correct.

**Step 6: Apply Your Terraform Configuration**

1. **Apply the Changes**:

terraform apply

* + Terraform will prompt you to confirm the action. Type yes to proceed with the deployment.

**Step 7: Verify Your EC2 Instance**

* After the terraform apply command completes, you can verify that your EC2 instance has been created:
  + Go to the [AWS Management Console](https://aws.amazon.com/console/) and navigate to the EC2 dashboard.
  + Check if the instance appears in the instances list.

**Step 8: Clean Up**

To destroy the resources created by Terraform (e.g., the EC2 instance), run the following command:

terraform destroy

* This command will prompt you to confirm the action. Type yes to proceed.

1. **Create Lambda function in AWS n java program upload**

Open visual studio

Run Command Prompt or PowerShell as Administrator

Download maven by choco install maven

mvn archetype:generate -DgroupId=com.example -DartifactId=my-lambda-function -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

cd my-lambda-function

open pom.xml

<project xmlns="http://maven.apache.org/POM/4.0.0"

         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>

    <artifactId>my-lambda-function</artifactId>

    <version>1.0-SNAPSHOT</version>

    <packaging>jar</packaging>

    <dependencies>

        <dependency>

            <groupId>com.amazonaws</groupId>

            <artifactId>aws-lambda-java-core</artifactId>

            <version>1.2.1</version>

        </dependency>

        <dependency>

            <groupId>junit</groupId>

            <artifactId>junit</artifactId>

            <version>4.12</version> <!-- or any compatible version -->

            <scope>test</scope>

        </dependency>

        <!-- Add other dependencies if required -->

    </dependencies>

    <build>

        <plugins>

            <plugin>

                <groupId>org.apache.maven.plugins</groupId>

                <artifactId>maven-compiler-plugin</artifactId>

                <version>3.8.1</version>

                <configuration>

                    <source>1.8</source>

                    <target>1.8</target>

                </configuration>

            </plugin>

            <plugin>

                <groupId>org.apache.maven.plugins</groupId>

                <artifactId>maven-shade-plugin</artifactId>

                <version>3.2.4</version>

                <executions>

                    <execution>

                        <phase>package</phase>

                        <goals>

                            <goal>shade</goal>

                        </goals>

                    </execution>

                </executions>

            </plugin>

        </plugins>

    </build>

</project>

Make new java file name LambdaHandler.java

package com.example;

import com.amazonaws.services.lambda.runtime.Context;

import com.amazonaws.services.lambda.runtime.RequestHandler;

public class LambdaHandler implements RequestHandler<String, String> {

@Override

public String handleRequest(String input, Context context) {

context.getLogger().log("Input: " + input);

return "Hello, " + input;

}

}

Aptest.class

// Source code is decompiled from a .class file using FernFlower decompiler.

package com.example;

import junit.framework.Test;

import junit.framework.TestCase;

import junit.framework.TestSuite;

public class AppTest extends TestCase {

   public AppTest(String testName) {

      super(testName);

   }

   public static Test suite() {

      return new TestSuite(AppTest.class);

   }

   public void testApp() {

      assertTrue(true);

   }

}

App.java

package com.example;

import com.amazonaws.services.lambda.runtime.Context;

import com.amazonaws.services.lambda.runtime.RequestHandler;

// This class represents the expected input JSON structure

class InputData {

    private String message; // Field corresponding to the JSON input

    // Getter and Setter

    public String getMessage() {

        return message;

    }

    public void setMessage(String message) {

        this.message = message;

    }

}

// Your main Lambda handler class

public class App implements RequestHandler<InputData, String> {

    @Override

    public String handleRequest(InputData input, Context context) {

        // Access the input data

        String message = input.getMessage();

        // Process the input and return a response

        return "Hello, " + message; // This will return "Hello, <message from input>"

    }

}

mvn clean package

iam -> role -> create role -> services -> lambda ->next ->AmazonS3FullAccess ->CloudWatchFullAccess->create

1. **Navigate to AWS Lambda**: From the AWS dashboard, find and select **Lambda** under "Compute" or simply search for **Lambda** in the search bar.
2. **Create a New Function**:
   * Click **Create function**.
   * Select **Author from scratch**.
   * Fill in the basic details:
     + **Function name**: Name your function (e.g., myLambdaFunction).
     + **Runtime**: Select Java 11 or Java 17 (whichever your function is compatible with).
     + Execution role

Choose a role that defines the permissions of your function->Use an existing role->Existing role->khushi

* + Click **Create function**.

**Step 3: Upload the JAR File to AWS Lambda**

1. **Upload the JAR**:
   * On the function configuration page, under the **Code** section, you'll see an option to **Upload a .zip or .jar file**.
   * Choose **Upload from** → **.zip or .jar file**.
   * Click **Upload** and select the JAR file from your target directory (e.g., my-lambda-function-1.0-SNAPSHOT.jar).

[Lambda](https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/discover)->[Functions](https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions)->[javaproject](https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/javaproject)->Edit runtime settings->handler-> com.example.App::handleRequest

Now go in test->test event->test->test1-> {

"message": "khushi"

} ->click on test

1. Create Lambda function using S3 bucket

Create bucket ->add helloworld.java (just like above)

Create lambda->add trigger->S3(source)->name of bucket->add

Test->testname->test like above

1. Sonar cube n sonar scanner installation

7. Study expt- NAGIOS, cloud 9,kubernetes