AWS project 3

AWS 3 tier project

Overview of the Lab

In this lab you will learn how to create 3 tier architecture using aws services

Scaling

Adjusting the number or size of instances based on demand.

Instance Type

It is the size, power, and capacity of an instance (CPU, Memory and Storage)

Apache

It is a web server software developed by Apache Software Foundation

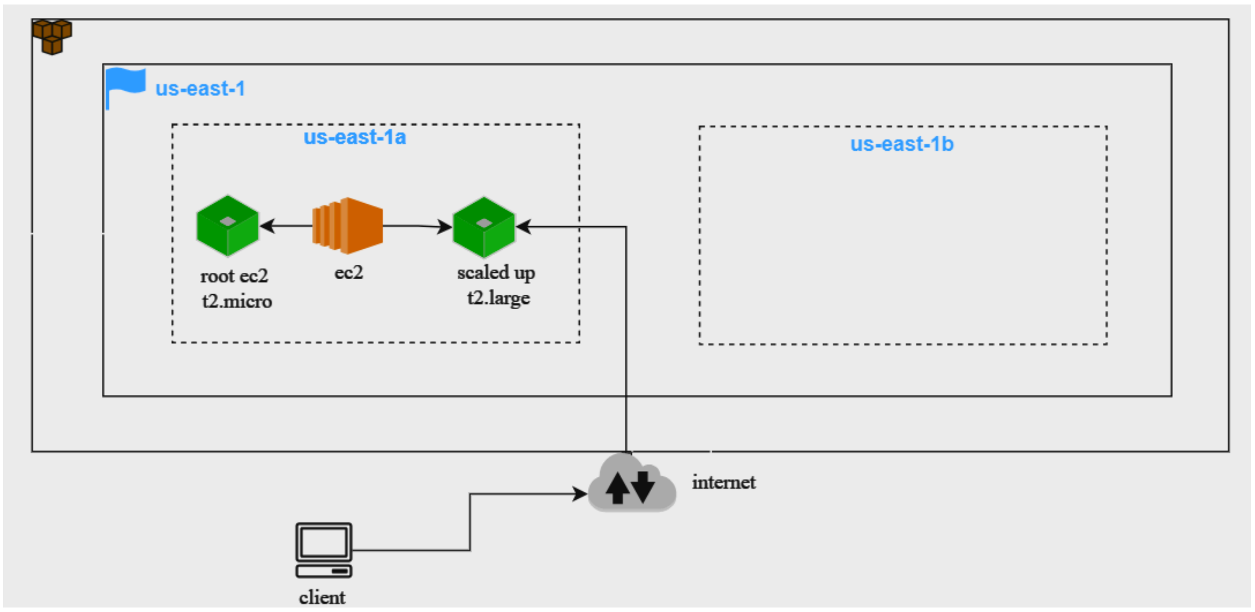
EC2 Instance Connect

It is a browser based CLI login for Linux servers (not all Linux OS are supported)

User Data

Aws userdata is the set of commendable/data you can provide to an instance at launch time.

Architecture of the Lab



Step by Step Lab

Go to vpc service to create custom vpc

1. Select vpc and more
2. Name - demo-vpc
3. IPV4 CIDR block - 192.168.0.0/22
4. Tenancy - default
5. Number of availability zone - 2
6. Number of public subnet - 2
7. Number of private subnet - 4
8. Nat gateway - 1
9. Vpc endpoint - None
10. Click - create vpc

Edit the subnet names

1. Rename the(demo vpc subnet private 1 - ap-south-1a) to (demo vpc subnet app 1 - ap-south-1a)
2. Rename the(demo vpc subnet private 2 - ap-south-1a) to (demo vpc subnet app 2 - ap-south-1a)
3. Rename the(demo vpc subnet private 3 - ap-south-1a) to (demo vpc subnet app 3 - ap-south-1a)
4. Rename the(demo vpc subnet private 4 - ap-south-1a) to (demo vpc subnet app 4 - ap-south-1a)

Now go the security group (5)

Create security group for web alb

1. Security group name - Web-Alb-sg
2. Description - demo-vpc
3. Vpc - demo vpc
4. Inbound rules - type - http source - anywhere 0.0.0.0/0
5. Outbound rules - all traffic (leave it default)

Web tier security group

6. Security group name - Web-sg
7. Description - demo-vpc
8. Vpc - demo vpc
9. Inbound rules - type - http source - custom - 192.168.0.0/22
10. Inbound rules - type - http source - custom - Web alb sg
11. Outbound rules - all traffic (leave it default)

12. Click create security group

Data base security group - private

13. Security group name - App-sg
14. Description - demo-vpc
15. Vpc - demo vpc
16. Inbound rules - type - custom tcp port - 4000 source - custom - 192.168.0.0/22
17. Outbound rules - all traffic (leave it default)
18. Click - create security group

Database sg

19. Security group name - RDS-sg
20. Description - demo-vpc
21. Vpc - demo vpc
22. Inbound rules - type - mysql aurora source - custom - 192.168.0.0/22
23. Outbound rules - all traffic (leave it default)
24. Click - create security group.

Internal load balancer security group

25. Security group name - Internal -Alb-sg
26. Description - demo-vpc
27. Vpc - demo vpc
28. Inbound rules - type - Http source - custom - 192.168.0.0/22
29. Outbound rules - all traffic (leave it default)
30. Click - create security group.

Create the s3 bucket

1. Name - Demo 3 tier project
2. Rest leave it as default
3. Click create bucket
4. Now click on the name of the bucket.
5. Now upload the application folder to your s3 bucket.

Note

From the 3 tier architecture git hub fork the 3 tier architecture to git hub and create the repository and download the application folder to your local computer.

Create a Iam role

1. Click on create role
2. Entity type
   2.1. AWS service
3. Use case
   3.1. Ec2
4. Add permission
   4.1. Amazon ec2 role for SSM.
5. Click - next
6. Role name - demo-ec2-role
7. Create = role

Create Relational database subnet group

1. Name - DB-sngp
2. Vpc - demo-vpc
3. Add subnet

   3.1. Availability zone
   3.2. Ap-south-1a
   3.3. ap -south-1b
4. Subnets -
   4.1. demo-vpc-subnet-DB1-ap-south-1a
   4.2. demo=vpc=subnet-DB2-ap-south-ib
5. Click on create

Create the relational database database tier setup

1. Click on the database
2. Select the standard create.
3. Engine option - Mysql
   3.1. Edition - mysql community
   3.2. Engine version - mysql 8.0.3.5
4. Template
   4.1. Free tier
5. Settings
6. DB instance identifier - database-1
7. Credential settings
   7.1. Master username - admin
8. Credential management
   8.1. Self managed
   8.2. Master password - admin12345##
   8.3. Confirm password - admin12345##
9. Burstable class - DBt4g.micro
10. Storage - general purpose ssd gp2
11. Connectivity type
    11.1. Dont connect to an computer resource
12. Network type - ipv4
13. Vpc - demo-vpc

14. DB subnet group : db-sngp
15. Public access - no
16. Vpc security group
    16.1. Choose existing - RDS- sg
17. AZ - no preference
18. Database authentication
    18.1. Password authentication
19. Back up - uncheck
20. Click - database

Application tier setup

Launch an instance

1. Login to aws cloud account via the aws management console
2. Select us-east-1 region

   (you can choose any region of your choice)
3. Search for EC2 and in EC2 management console, launch instance
    3.1. Name and tags – App-tier
    3.2. Application and OS Images – Amazon-linux2
    1.1. Instance type - t2.micro
    1.2. Key pair – Create new keypair
        1.2.1. Key pair name - proceed with no key pair - Click on Create key pair

    (download and save for later use)
    1.3. Edit Network settings
        1.3.1. Subnet – demo-vpc-app-ap-south-1

1.3.2. Firewall – choose the existing security group

1.3.3. Security group name - app-sg

1.4. Click drop down advance details

1.4.1. Iam role - demo-role

2. Number of instances - 1

(Leave all other settings as default and launch instance)

1. Once the instance is launched
   1.1. Wait for instance state – running
   1.2. Wait for status check – 2/2

After connecting to linux ssm copy paste the below commands.

1. Navigate to root.
2. Sudo su
3. Cd ..
4. Cd /home/ec2-user
5. Check the internet connection
6. Ping 8.8.8.8

1. In this instance we will do the App Server Setup and DB Server Configuration. Execute the below commands;
2. Install MySQL
3. sudo yum install mysql -y

1. Configure MySQL Database
2. Connect to the database and perform basic configuration: Replace below info with your DB information
3. mysql -h <DB EndPoint> -u admin -p ----> Enter the Password i.e kastro2025 (this is DB password). If you couldn't connect, there is a problem with the SG of the DB.

4. Ex: mysql -h database-1.c380a08uukyc.ap-south-1.rds.amazonaws.com -u admin -p

   1. Lets create a database. The database name i'm creating is "webappdb" (This is same name that you should give in DvConfig.js file);
   2. CREATE DATABASE webappdb;
   3.
   4. SHOW DATABASES;
   5.
   6. USE webappdb; ----> You will see 'Database changed'

1. Execute the below code as a single code. Here we are creating a table with the name 'transactions'
2. CREATE TABLE IF NOT EXISTS transactions(
3. id INT NOT NULL AUTO_INCREMENT,
4. amount DECIMAL(10,2),
5. description VARCHAR(100),
6. PRIMARY KEY(id)
7.);

1. To verify whether table got created or not;
2. SHOW TABLES;

1. Lets insert some info into the table
2. INSERT INTO transactions (amount, description) VALUES ('400', 'groceries');

1. To verify whether the entry is really created or not
2. SELECT * FROM transactions;
3. You will see the info you have written

4. To come out of the DB;
5. exit (You will see 'ec2-user' at the end of command line and at the beginning of command line you will see 'root')

1. Update Application Configuration to with DB information
2. Update the **application-code/app-tier/DbConfig.js** file with your database credentials.

1. nstall and Configure Node.js and PM2
2. curl -o- https://raw.githubusercontent.com/avizway1/aws_3tier_architecture/main/install.sh | bash
3. source ~/.bashrc

1. nvm install 16
2. nvm use 16 (You will see 'Now using node v16.20.2)
3. NVM means Node Version Manager

1. To run node as a service, we will install pm2
2. npm install -g pm2 (You will see 'found 0 vulnerabilities)

1. Download application code from S3 and start the application
2. cd ~/
3. sudo aws s3 cp s3://<S3BucketName>/application-code/app-tier/ app-tier --recursive
4.
5. Ex: sudo aws s3 cp s3://demo-3tier-project/application-code/app-tier/ app-tier --recursive

1. ls ---> You will see 'app-tier' folder
2.
3. cd app-tier/
4. npm install

ls ----> You will see 'index.js' file. We have to start that.

pm2 start index.js (You will see the status as 'online')

To verify;
pm2 list (or) pm2 status
pm2 logs (You will not see anything in red colour, everything in white colour you should see)

At the end you will see something like; http://localhost:4000

pm2 startup
pm2 save ---> To save the configuration

Verify that the application is running by executing
curl http://localhost:4000/health
It should return: This is the health check.

With this we have completed the application configuration.

4.2. Creation of Internal Load Balancer for App Tier

Goto the downloaded code folder in local system ----> Open nginx.conf file and in the end of the file you will see something like below;

```
    #proxy for internal lb
    location /api/{
            proxy_pass http://[REPLACE-WITH-INTERNAL-LB-DNS]:80/;
    }
```

Replace the LB DNS in the above

Upload the updated nginx.conf file to the S3 bucket

This one we are going to copy to the webserver in sometime.

Create the web tier application instance

1. Login to aws cloud account via the aws management console
2. Select us-east-1 region (you can choose any region of your choice)
3. Search for EC2 and in EC2 management console, launch instance
   - 3.1. Name and tag – linux-webserver
   - 3.2. Application and OS Images – Amazon Linux
   - 3.3. Instance type - t2.micro
4. Key pair – select the existing keypair
5. Edit Network settings
   - a. Subnet – subnet in us-east-1a (even no preference is fine)
   - b. Firewall – select existing security group

4. Number of instances - 1

   (Leave all other settings as default and launch instance)

5. Once the instance is launched
   - 5.1 Wait for instance state – running

After connecting to linux ssm copy paste the below commands.

Creation of Web tier resources including External Load Balancer

```
sudo -su ec2-user (To work as an ec2-user)
```

```
cd /home/ec2-user
```

sudo amazon-linux-extras install nginx1 -y

Update Nginx configuration:

cd /etc/nginx (Your are in nginx path)

ls ----> You will see 'nginx.conf' file

sudo rm nginx.conf

sudo aws s3 cp s3://<S3 Bucker Name>/application-code/nginx.conf .

Ex: sudo aws s3 cp s3://demo-3tier-project/application-code/nginx.conf .

sudo service nginx restart

chmod -R 755 /home/ec2-user

sudo chkconfig nginx on

To check the output of the App, we can check using the Web-Tier-Instance public IP. But before checking lets open port no 80 with http, Anywhere IPv4, 0.0.0.0/0 ---> Save rules ----> Now paste the pubic ip of Web-Tier-Instance in new tab of browser ----> You will see the app ----> Enter the data in the app

```
curl -o- https://raw.githubusercontent.com/avizway1/aws_3tier_architecture/main/install.sh | bash

source ~/.bashrc

nvm install 16

nvm use 16


aws s3 cp s3://<S3 Bucker Name>/application-code/web-tier/ web-tier --recursive

Ex: aws s3 cp s3://demo-3tier-project/application-code/web-tier/ web-tier --recursive


ls ----> You will see 'web-tier'


cd web-tier

npm install

npm run build
```