AWS Deployment

#springboot #aws

Using EC2 Instance and S3 bucket

We are going to use two services of AWS, EC2 and S3 Buckets
EC2 stands for **Elastic Compute Cloud** this is where we create out VM that we need for our applications
S3 bucket stands for **simple storage service** is a simple storage service where we can store/copy our artefacts like jar and war files or it can be anything that is required for our application. Spring boot jar will go into the S3 bucket and on the EC2 instance will set up mySQL which is required for the application.

1. Create AWS account
2. Search for EC2
3. Create an Instance
4. Install Java&mySQL in it
5. Create a S3 bucket
6. Upload required files (JAR)

EC2 allows you to create instances or machines where you can deploy our applications.

Steps:

1. Launch a Ec2 Instance
2. Setup Java and MySQL DB
3. Copy the JAR to S3 Bucket
4. Deploy and Test

Launch a EC2 Instance :

Login to AWS Console
Search for EC2
Create new EC2 Instance
It Includes Steps for selecting OS, RAM, Storage, Security etc.,

1. AMI (Amazon Machine Image) : Choose which is eligible for free tire and supports most of your requirements.

2. Instance Type : We can choose number of CPU's, Processors etc here, choose which is free tire eligible.

3. Configure Instance Details : We can choose number of instances to be created,

which is number of VM should be created, one instance is enough and leave rest of the options as default.

4. Storage Details : We can choose size/volume of the instance, default size is 8GB and free tire is eligible for upto 30GB, leave it to 8GB (type = EBS (Elastic block storage))

5. Add Tag : Here we can add tags to out instances, which can be alter used to group instances later on

6. Security Group : We can configure how instance can be accessed from different places (local/Remote), By default Security group has certain rules, It has one, Allow SSH Connections, so that we can connect to this instance from local machine or remote through SSH, it has port details which we can access. We can create rules to open up other ports and IP Addresses as well

Default rule will only open up port 22 from anywhere as ip address is specified as 0.0.0.0/0 through SSH

Add custom rule, Choose TCP, Port number 80, Address as Anywhere, Custom, Current IP

7. Review : we can review all the configuration we have setup in all the steps and launch the instance

NOTE : When you launch an instance it will ask for Selecting **Key Pair,** it can be existing or we can create new one. The SSH communication happens through these keys instead of user names and passwords, it has public and private keys, AWS creates a public and private key where public key is stored in was and a private key is used /can be stored on any device that wish to connect to this instance.

For the first time you'll create a new key pair, give it a name and click on download key pair to your machine and store it anywhere in your machine(required later)

CLICK ON LAUNCH INSTANCE

Once the creation of instances is done, you can connect to it and start working with that.
Select that instance in instance list menu, here you can see all the details of the instance, the public name, IP Address , have status checks.
Select an instance and click on connect button to connect to that instance, there are various methods that you can connect to an instance (SSH Client, Session Manager, EC2 Instance Connect), there are instructions for every method

For windows, SSH client is PUTTY,
For Mac, there is already a terminal which has inbuilt support for ssh communication.

Use that ssh command to connect to the instance through terminal, run the command for the first time it will download few certificates and keys that are used to connect to the instance.

Then you are connected to your AWS Instance

But by default we are under **home/ec2-user**, we need to become root user which is important to have all the permissions
Do a **sudo -i**
To become root user, this must be your first command every time you connects to your instance.

For Windows use mobaXterm, SSH Communication, use host name(public IP of instance), use downloaded private key, select port and enter
Then use **ec2-user** in terminal

If you are not able to use mobaXTerm you can use putty, it is an another ssh client.

Setup Java and MySQL DB on the Instance

Connect to the instance through terminal/SSH Client

```
sudo -i     // to become root user

yum install java-1.8.0-openjdk

alternatives --config java

//selects the latest java

yum install mysql-server

chkconfig mysqld on

service mysqld start
service mysqld stop
service mysqld restart

mysqladmin -u root password test1234

mysql -u root -p
```

Once you connect to the mysql you can run the commands to create databases and populate them with the data, and you can do whatever you want to do with sql, basically it is an mySql server.

Copy the JAR to S3 Bucket

Create a JAR file of the project
Eclipse - run as maven install - creates a new JAR file in the target folder

Maven command - we need few plugins and there are various way to create a JAR file **`./mvnw clean package`**

Login to AWS console, go to S3 buckets, Create a new bucket

Give it a name, leave remaining options as it is,

NOTE: There is permission setting where it blocks the public access, usually in realtime applications this setting is turned on and there are policies through which they grant permission to access these buckets, but for our purpose we can turn it off.

Create bucket and wait for it to be created
Select the bucket and upload the JAR file into the bucket, click on next, Do not grant public read access, upload.

Wait for it to upload, after uploading make the jar object as public so that you can access it publicly

Deploy and Test

Use the object URL of the jar file which is in the bucket.
Connect to the instance through terminal
Use wget object url to download that object into instance machine.

```
wget <objectURL>
```

Now we have JAVA, SQL, and JAR file in our instance machine, we should now able to run the Spring boot application in the instance.
Use

```
java –jar <jarName>
```

This runs the application and this application runs on the port 8080 usually in our machine
Go to EC2 AWS console
Select instance which is running
Select Properties, by default only the SSH port will be enabled
We can see that in security group assigned to the instance
Select the security group and it has inbound rules
Security group is like a firewall by default only ssh port is enabled so that we connect to the instance
Edit the inbound rules, select custom TCP/IP, Port 8080, and anywhere and save the changes.
Once the changes are made, select the instance and copy the public DNS/Public IP Address

Paste the copied url into the browser and append the port number and our application RESTFUL end point to access the endpoint which returns the data from the database.