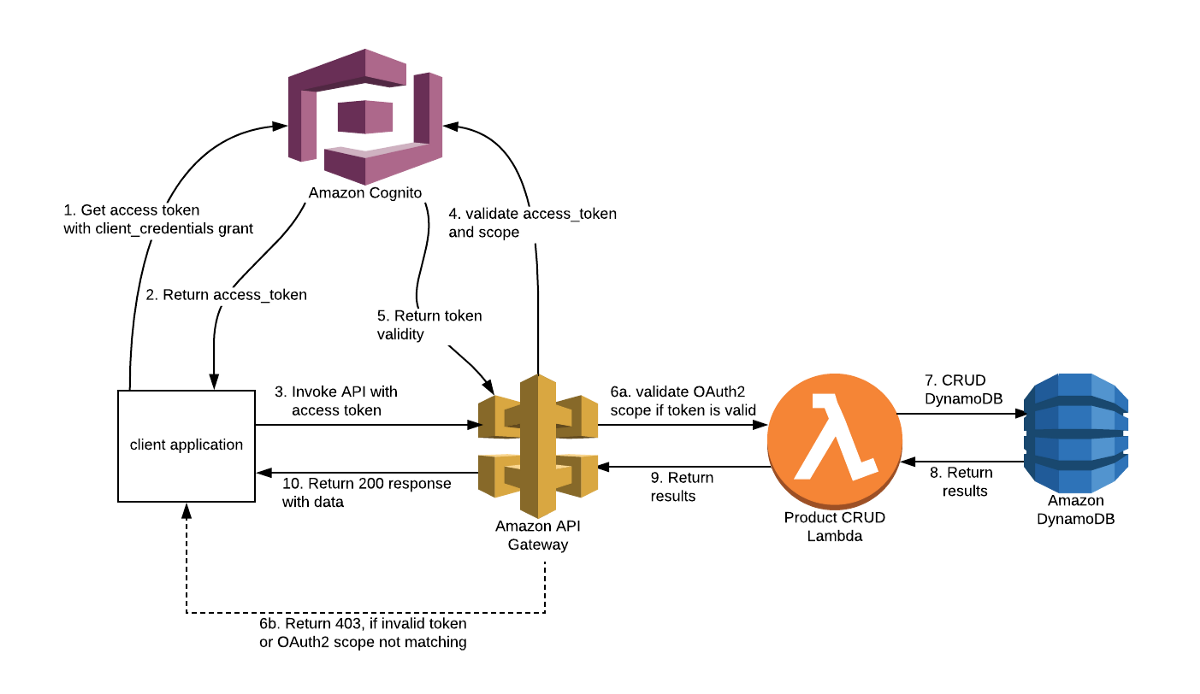
**AWS Cognito Webservice For ShopNow**

**Amazon Cognito :**

## What is Amazon Cognito?

Amazon Cognito is a simple user identity and data synchronization service that provides authentication, authorization and user management, helping us securely manage app data across applications for our users. Amazon Cognito allows us to control permissions for different users’ groups in our applications to ensure that they have appropriate access to back-end resources according to the group they belong to.

How Amazon Cognito Work-



See the Flow Diagram of Amazon Cognito . In Above Figure.The main Cognito Java classes we will be using in our Java application are:

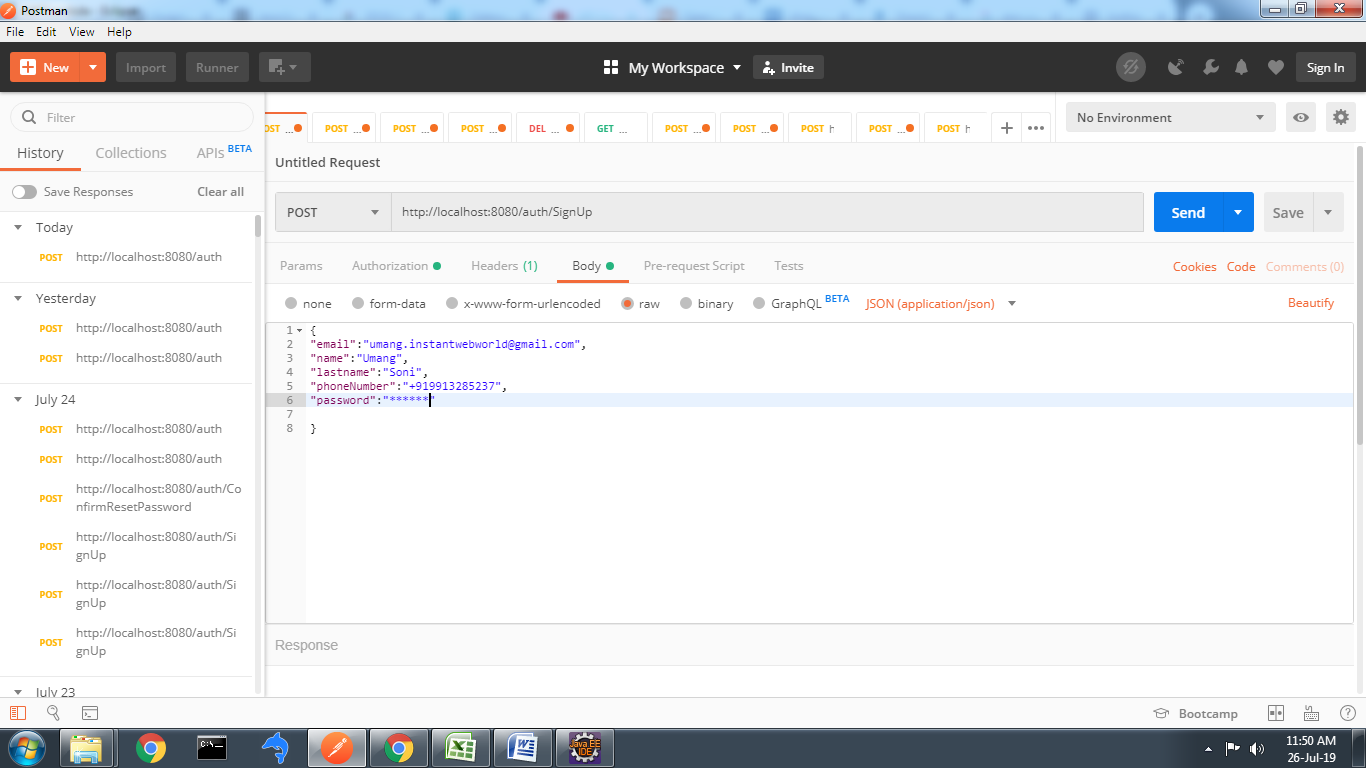
1. **AWSCognitoIdentityProvider:**The **AWSCognitoIdentityProvider** class allows us to execute a lot of actions, some of the most useful being:

* Add Custom Attributes
* Admin Add User to Group
* Admin Confirm Sign Up
* Admin Create User
* Admin Delete User
* Admin Reset User Password
* Admin Initiate Auth
* Admin Enable User
* Change Password
* Create Group
* Delete User
* Delete Group
* Delete User
* Forgot Password
* Get User
* Global Sign Out
* List Groups
* List Users
* Sign Up
* Update Group
* Update User Attributes

Step-1 If user want to use Shopnow webservice. Then we authentic him using AWS Cognito Inbuilt Webservice of Amazon. In this Step user sign-up if user new or if they alredy registed user then it login .

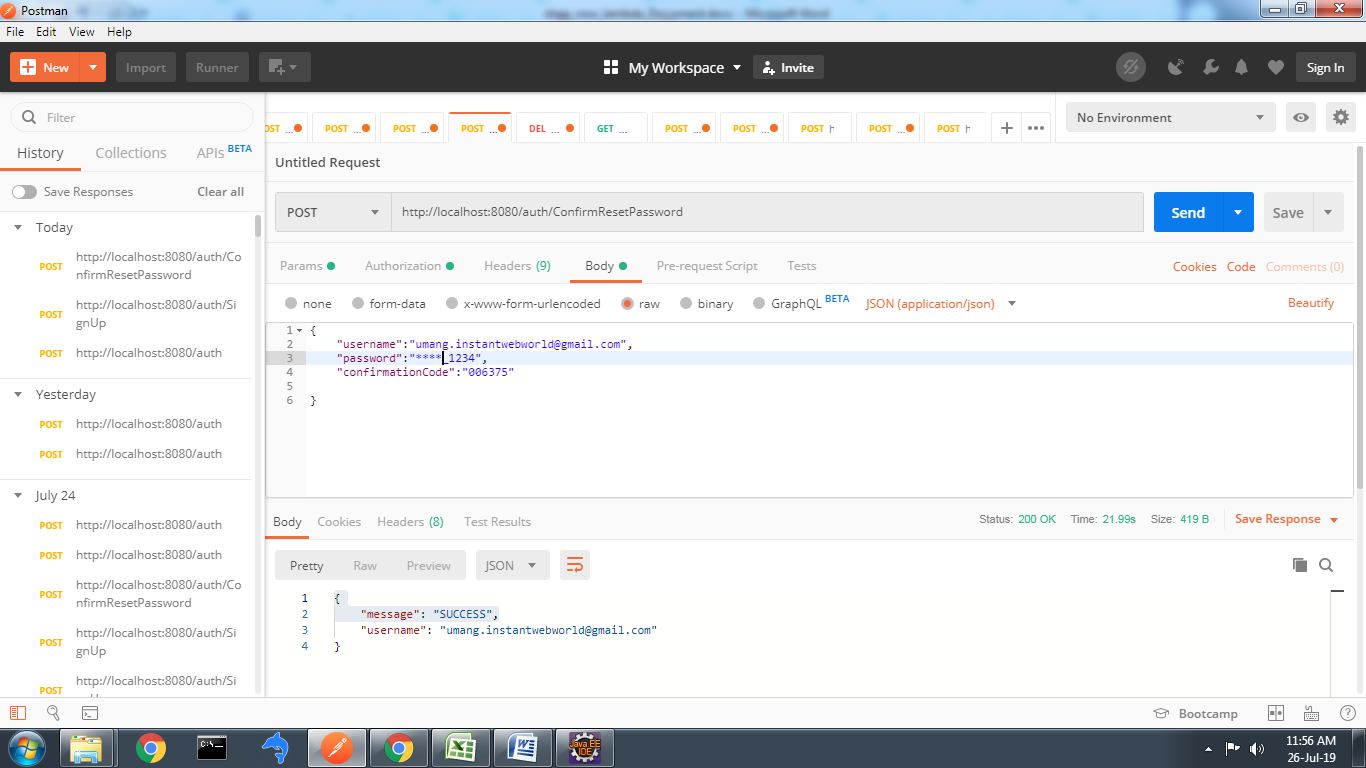
If user want to Login With Shopnow login and password then 1st they SignUp

**Registration**  - POST <http://localhost:8080/auth/SignUp>



After this step Conformation Code will send to LoginId Email .After then user call Conform Its Account using this step

Call - POST - <http://localhost:8080/auth/ConfirmResetPassword>

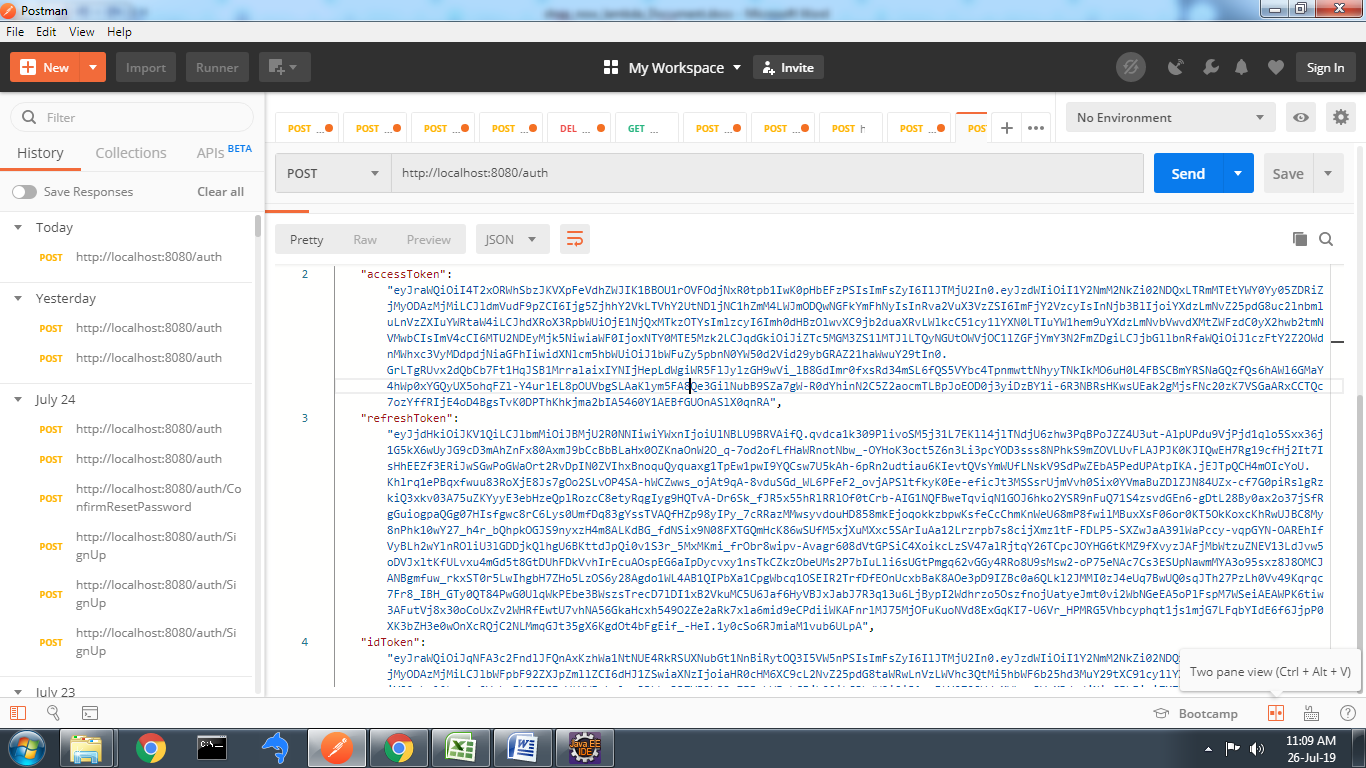


After Conform Account Successfully its Automatic Redirect Login Page .

**In Login** Step user can login with its signup loginId and Password OR They Login with different Social Account like google , FaceBook twitter etc. They Provide its Username and Password to Amazon cognito .if LoginId is Valid Domain then it’s give Token. see in figure

Request - POST http://localhost:8080/auth

Response -



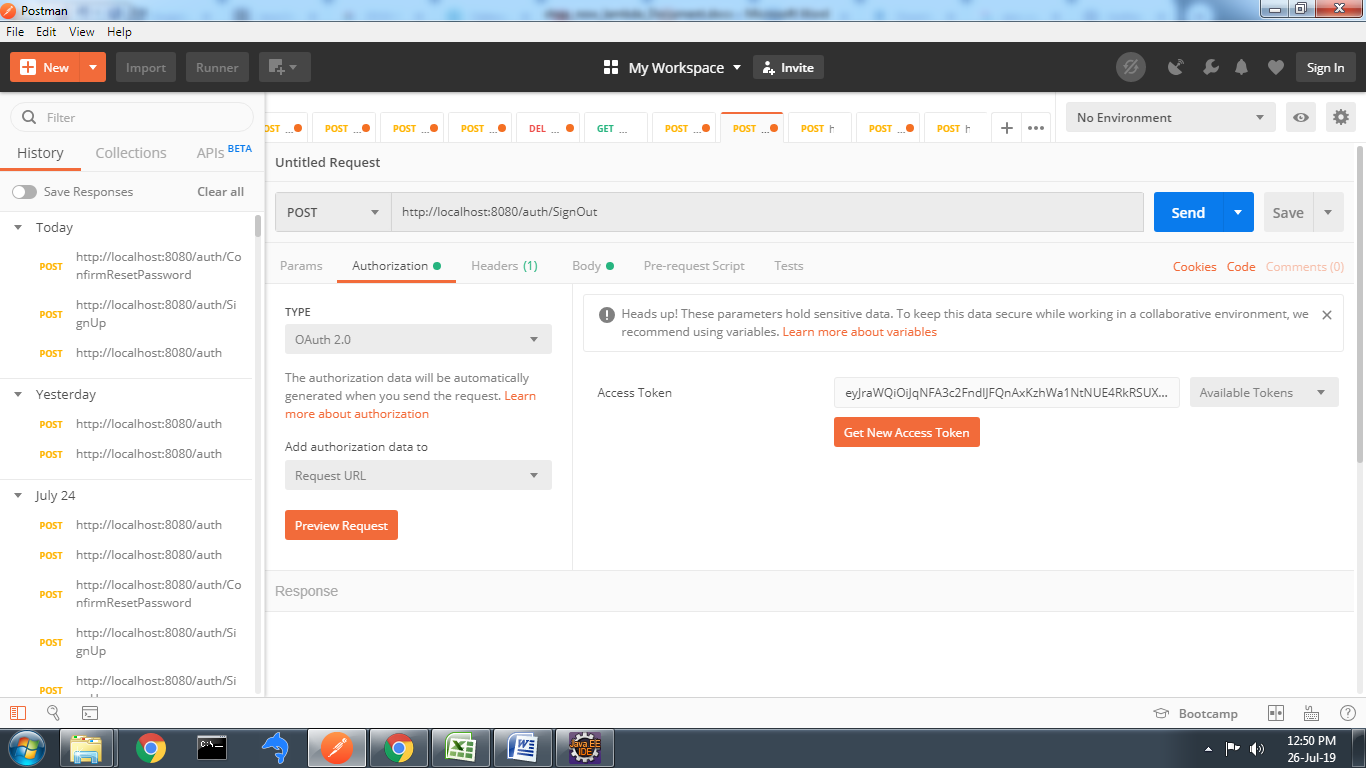
Amazon Cognito Provide Token it has Three Parts

1- accessToken 2 – refreshToken 3 – idToken

Using this token we can call others api like signOut , ChangePassword ,……

Here we give demo for SignOut API in which we pass accessToken and username.

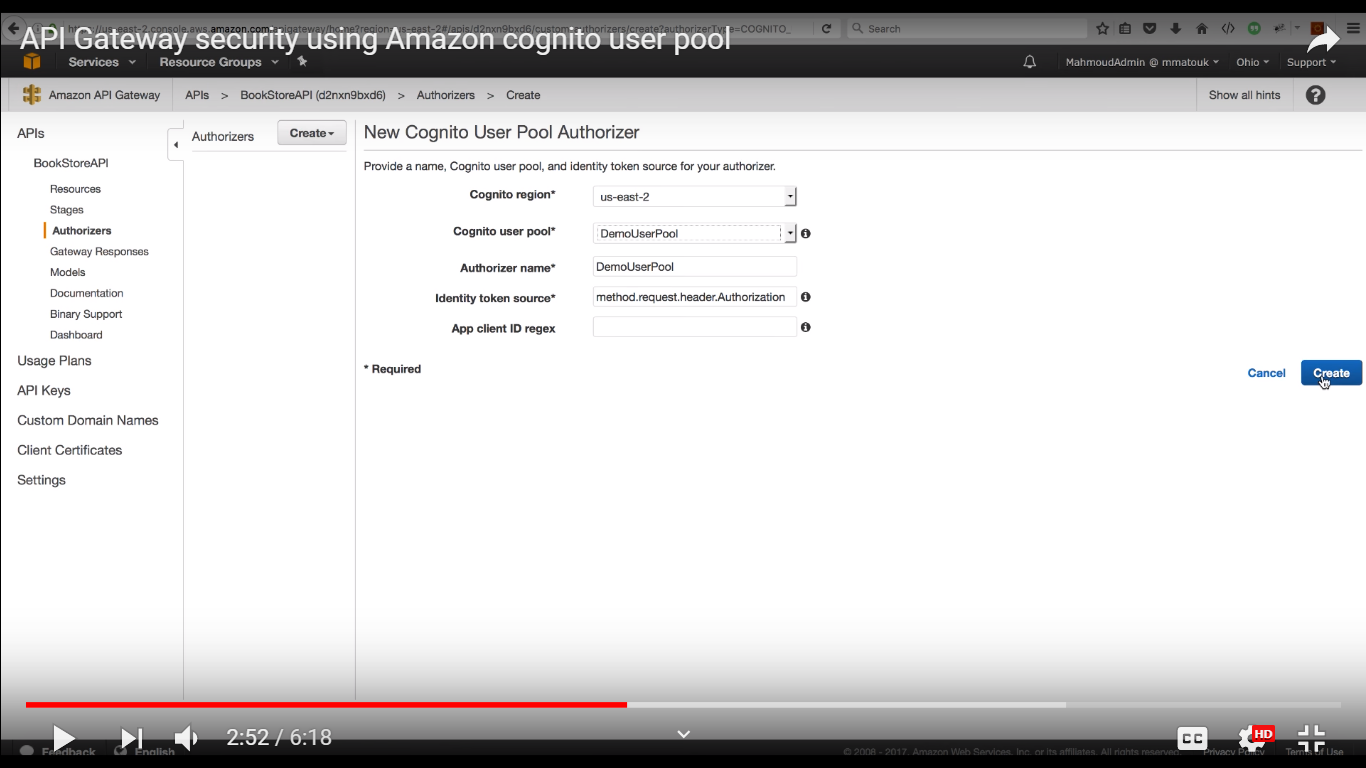
Authentication Menu select Oauth 2.0 and accessTokan . Body part pass username



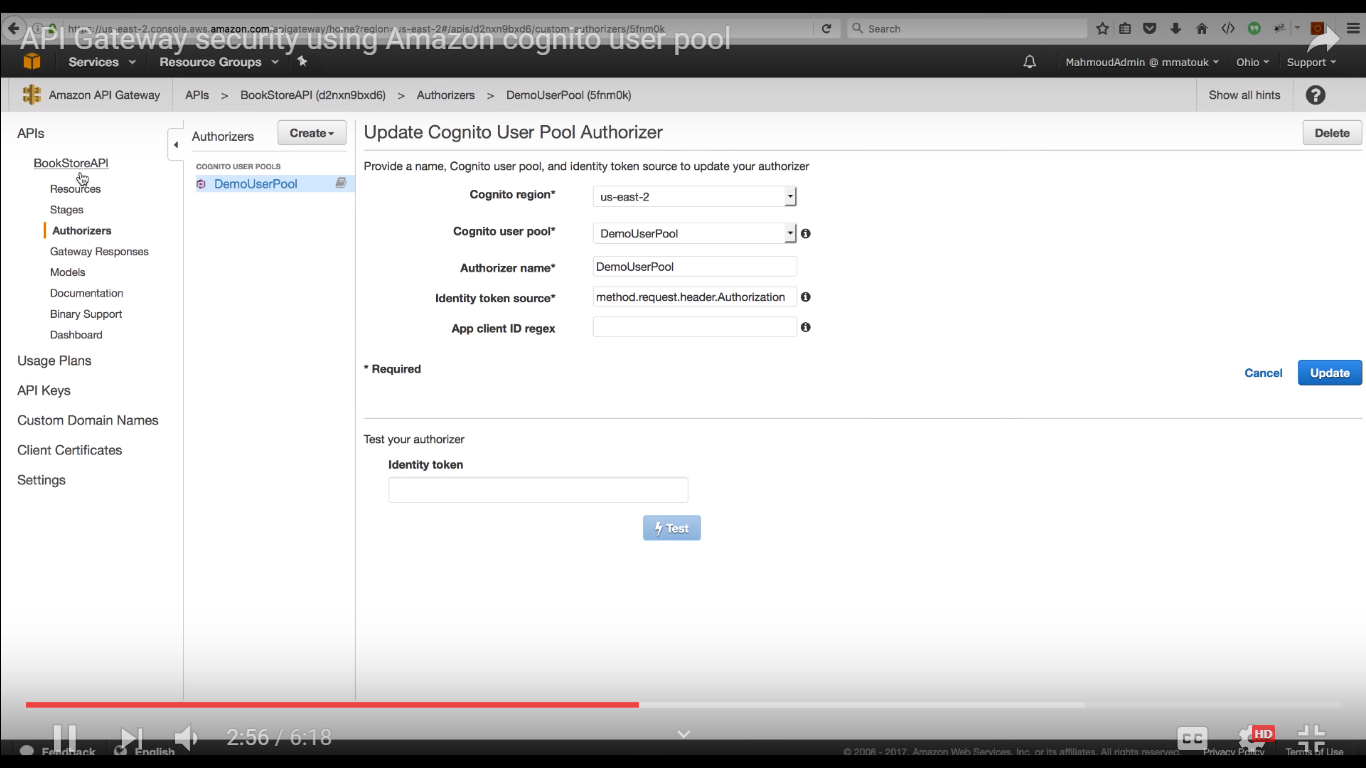
**Connection Between API Gateway and Cognito**.

First we create New API GateWay UserPool go Authorize Menu.

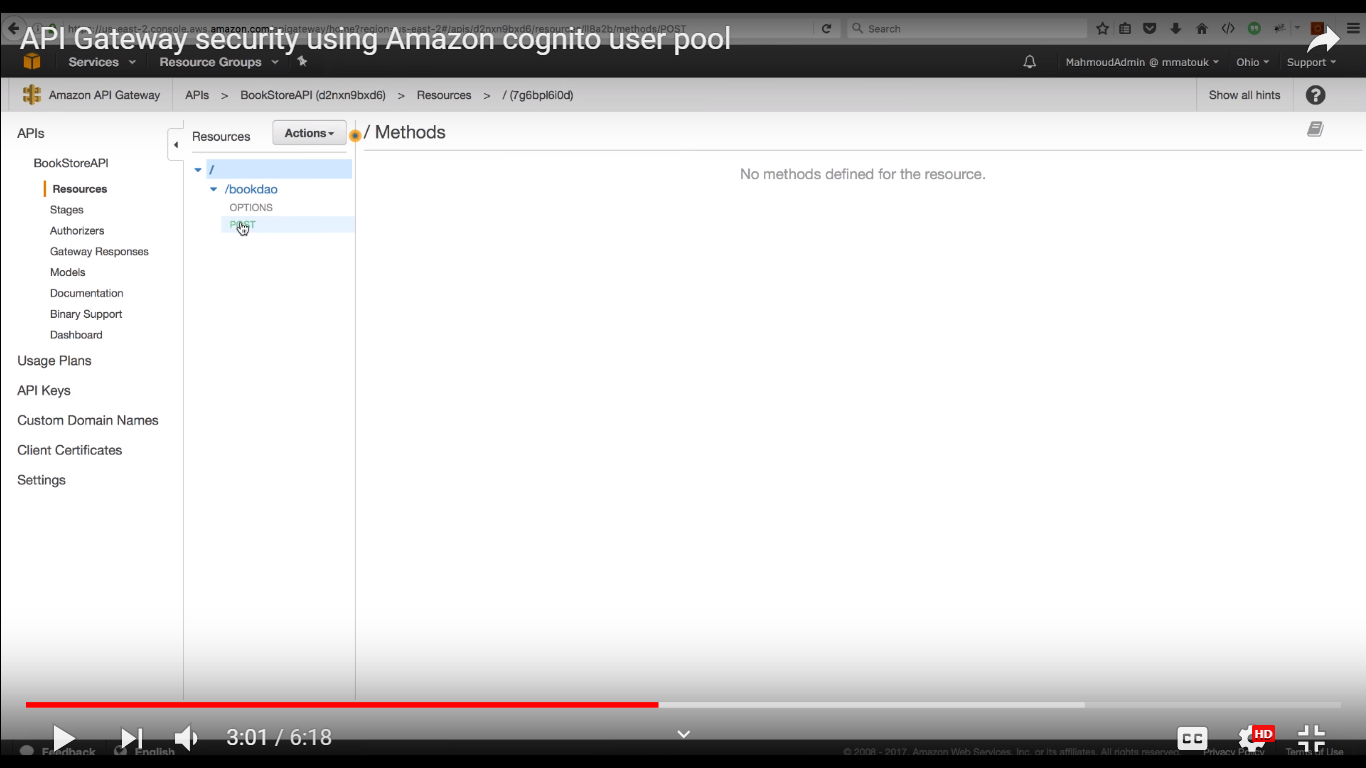
Create new Cognito User Pool Authorizer pass needed parameter and click Create button.



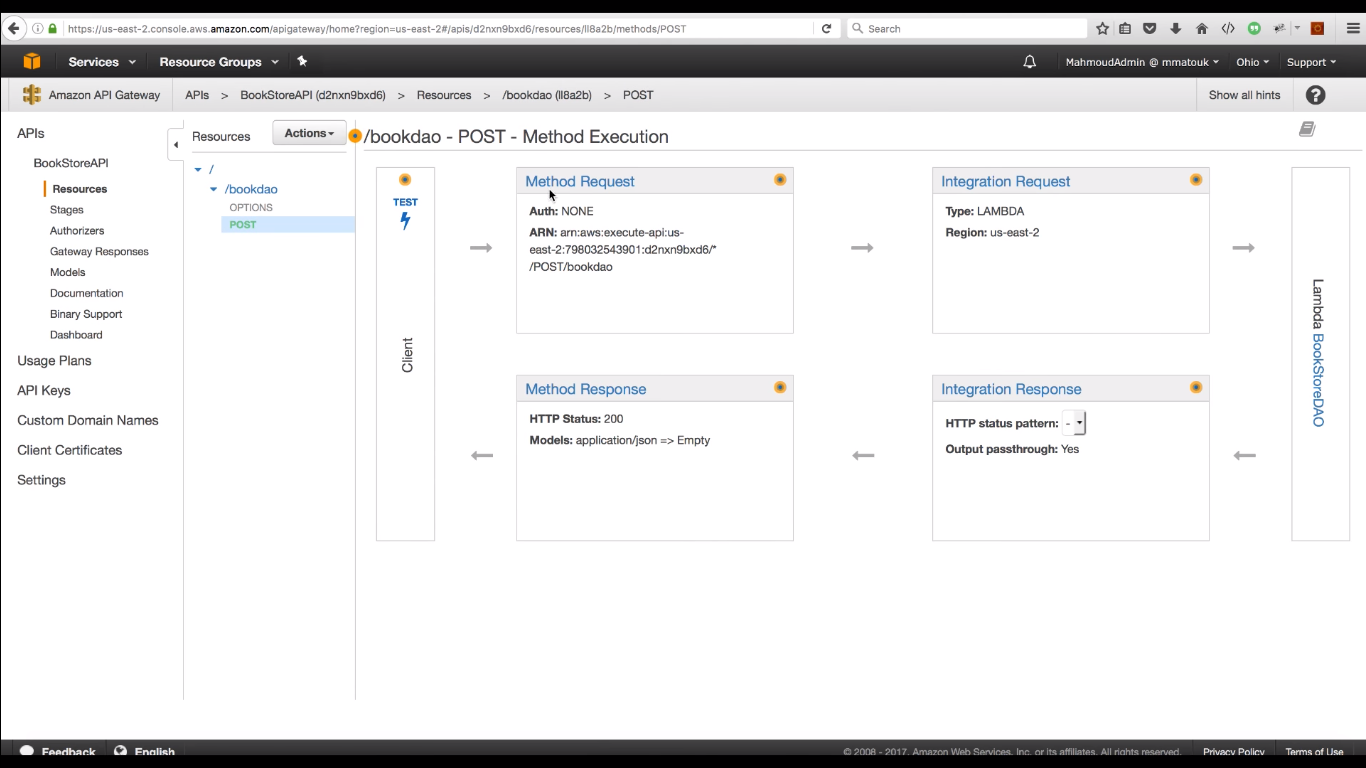
Pass the IdToken and upadate it



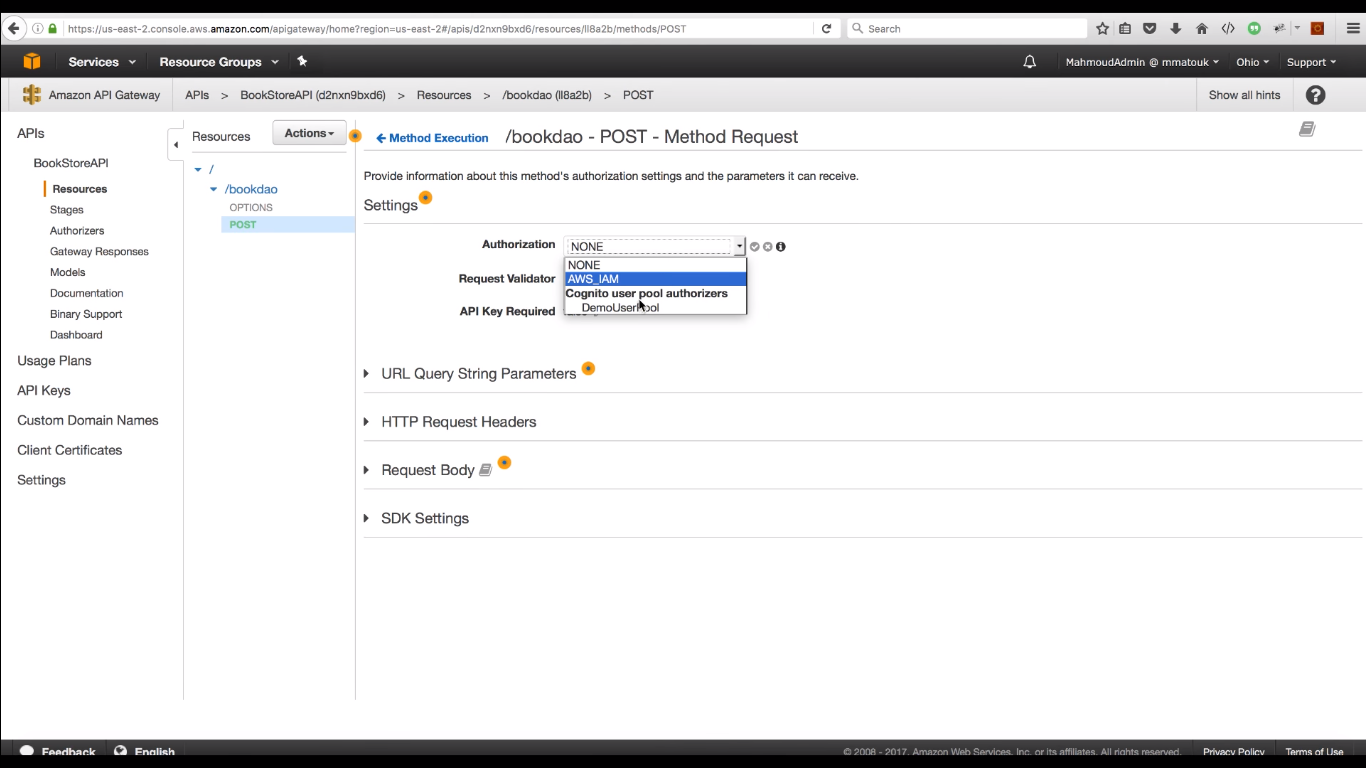
After That goto Resources link which is left corner of aws API Gateway menu. Here click on POST method



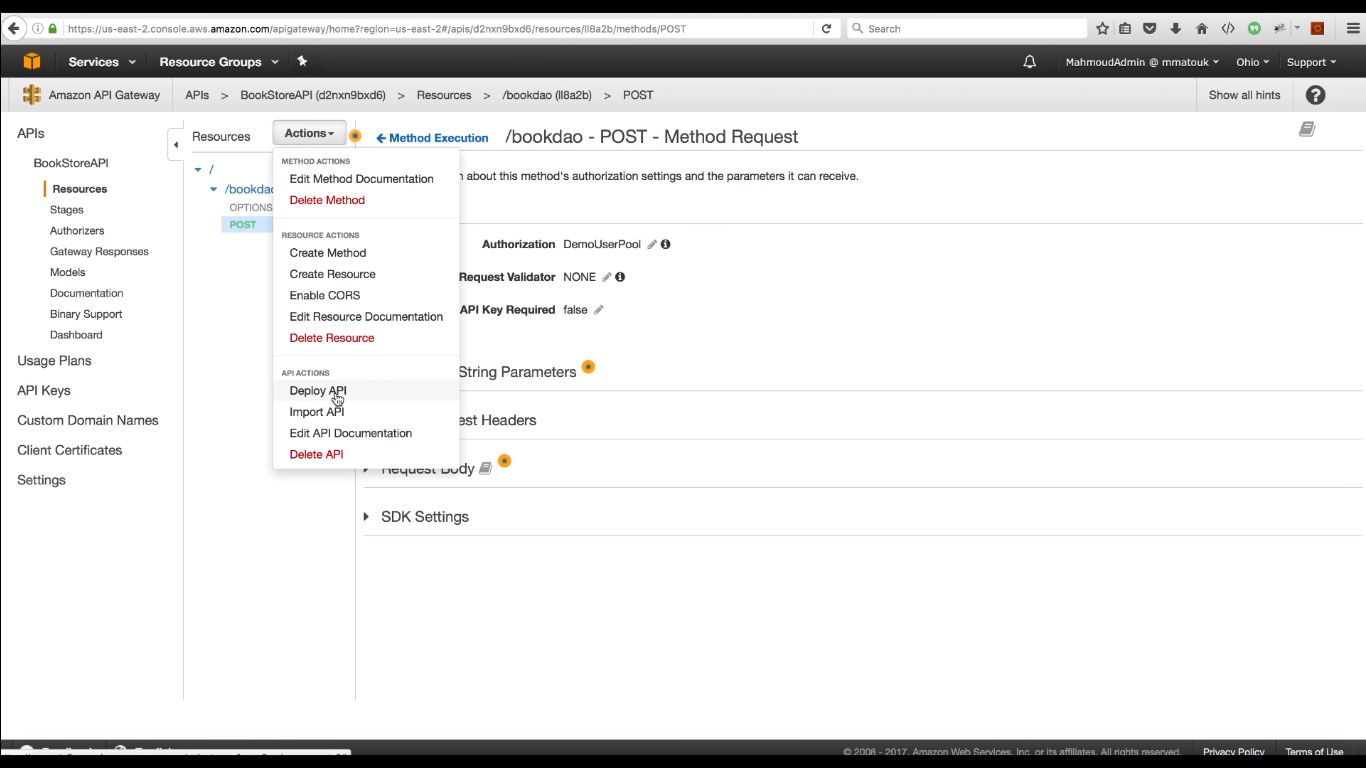
After Click on Method Request Link

****

Here set Authentication Name which we create

****

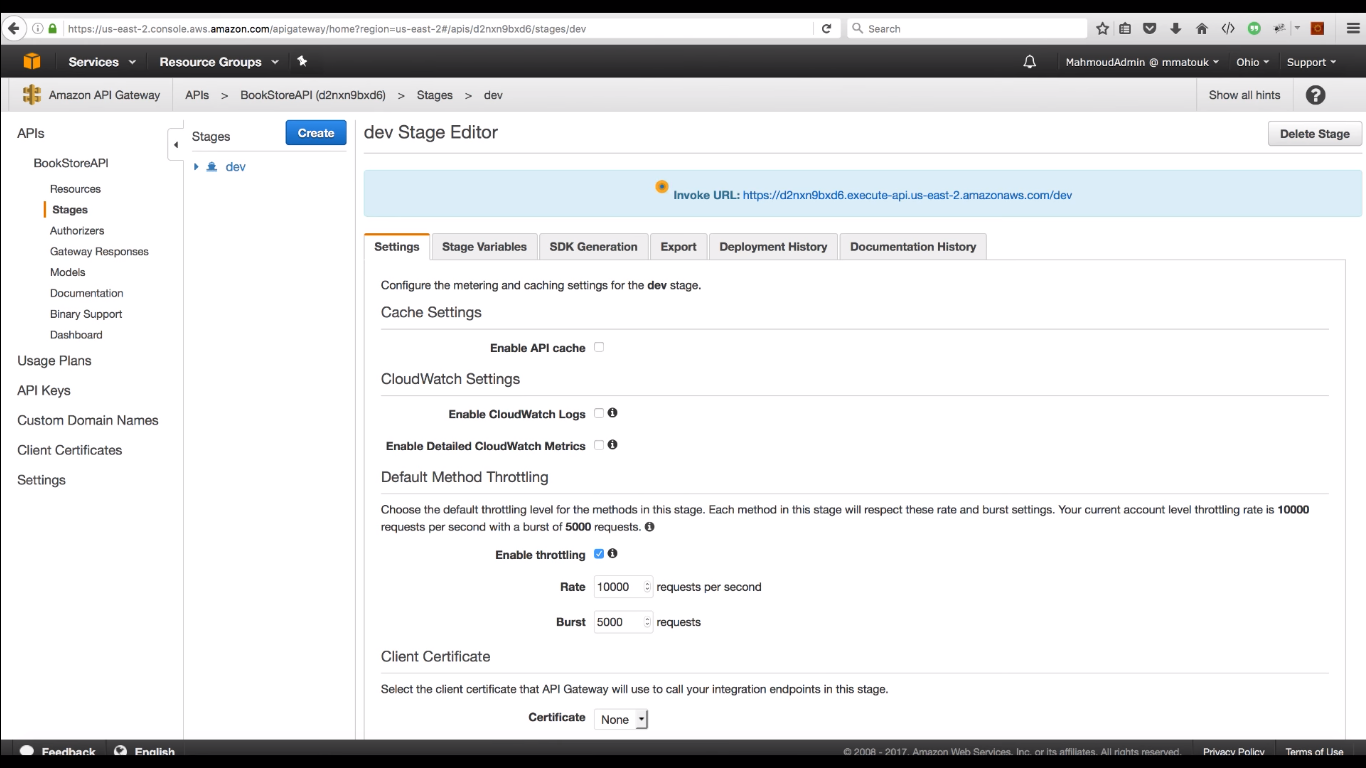
After goto Accetion Button and appy Deploye API

****

Set Deployment Stage and click Deploy Button

****

In this stage we get Invoke URL using this we can access Lambda function. Now Lambda Function call need Token Without authentication token we cannot access it.

****

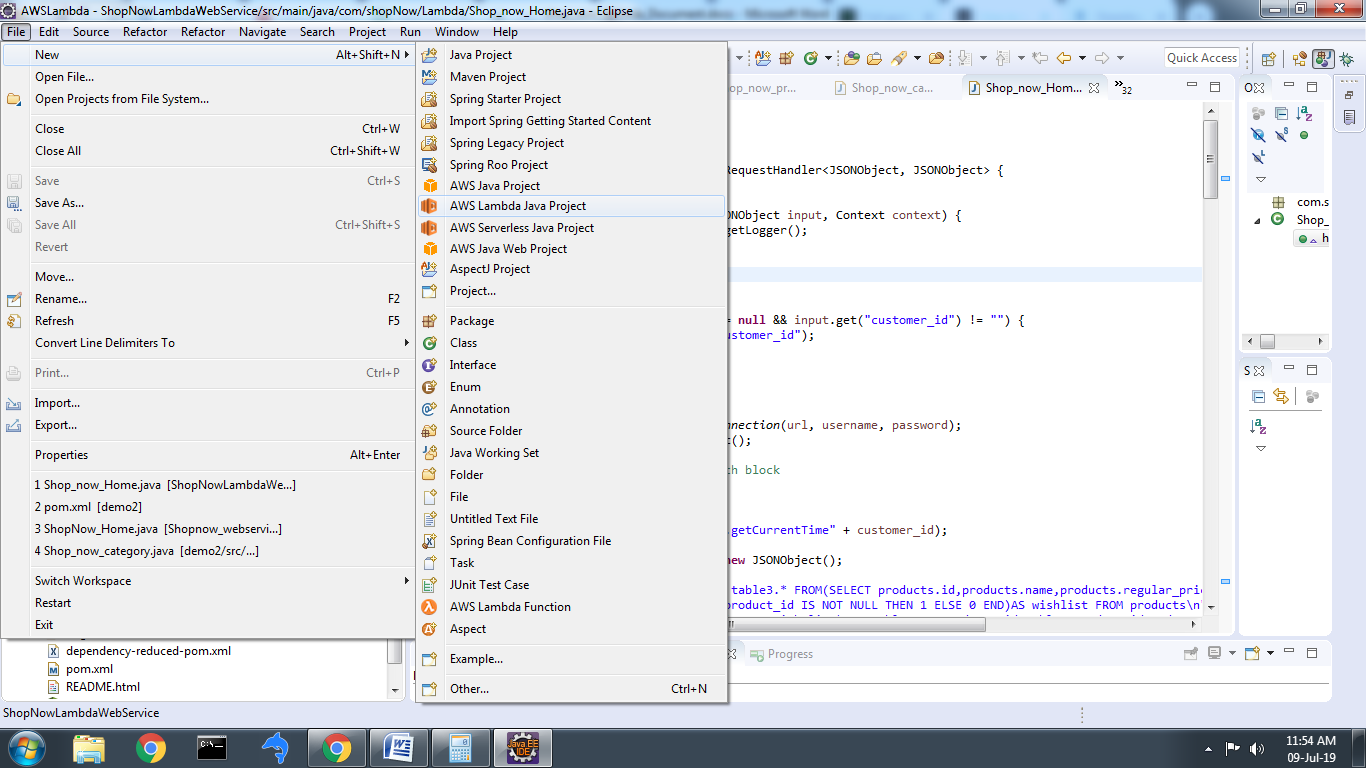
**Shop now AWS Lambda Web Service**

Step-1 to use Lambda and other AWS services, you need an AWS account. If you don't have an account, visit [aws.amazon.com](https://aws.amazon.com/) and choose **Create an AWS Account**.

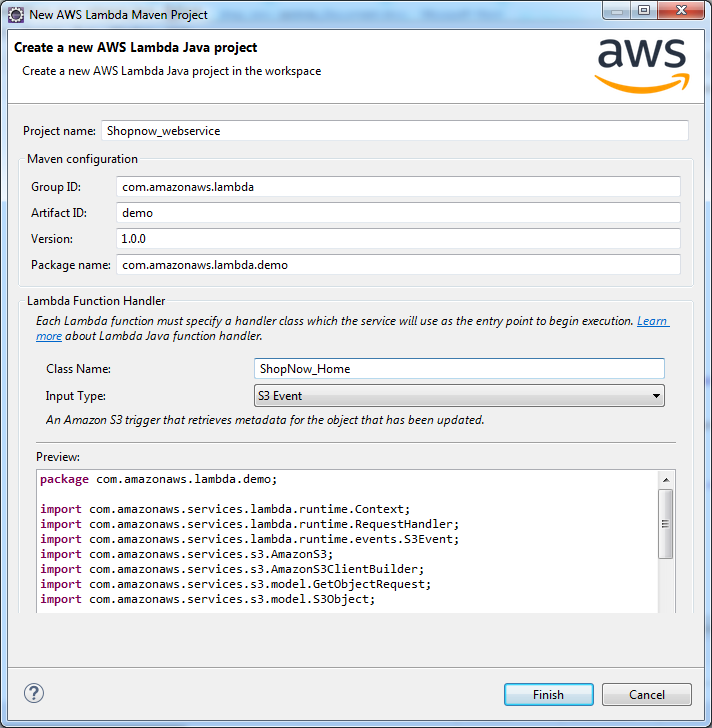
### Step-2 Install the AWS Toolkit for Eclipse

1. Within Eclipse, click **Help** and then click **Install New Software**.
2. In the **Work with** box, type https://aws.amazon.com/eclipse and then press Enter.
3. Choose the components of the AWS Toolkit for Eclipse that you want to install. Click **Select All** to install all components at once.

Step-3 **Create** AWS Lambda Java Project from AWS Toolkit.



Step-4 Add Project name, package name and Class name



### Step – 5 Add AWS lambda Function Logic and Save it.

### Step – 6 Add dependency in pomp.xml file and also add external jar files

### Dependencies

<Dependencies>

This dependency for amzone –Lambda web service

<dependency>

<groupId>com.amazonaws</groupId>

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

<version>1.0.0</version>

</dependency>

This dependency for database connectivity service

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.35</version>

</dependency>

This dependency use for create shade jar file which can be upload aws lambda function web service

<dependency>

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

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

<version>2.3</version>

</dependency>

This dependency for output in json formate web service

<dependency>

<groupId>com.googlecode.json-simple</groupId>

<artifactId>json-simple</artifactId>

<version>1.1</version>

### </dependency>

<dependencies>

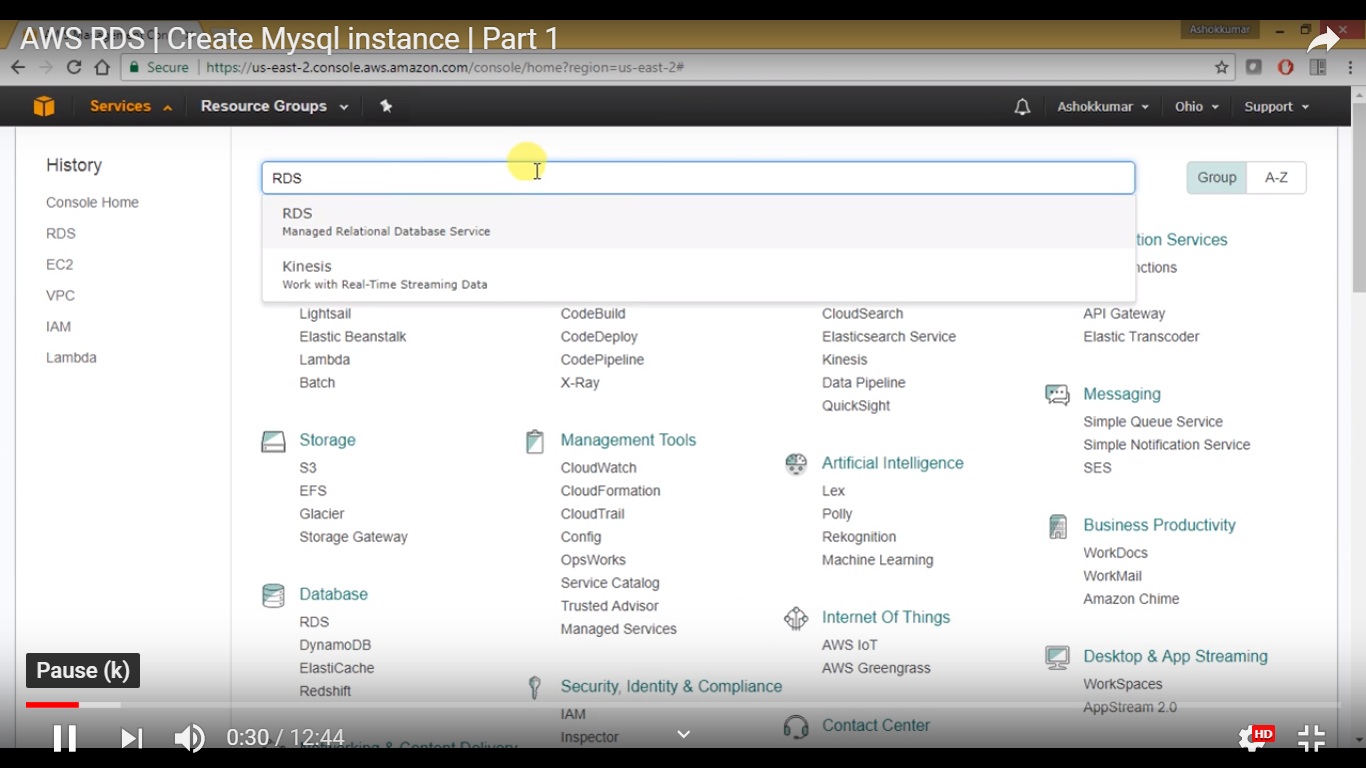
# Step- 7 Database connectivity

Create AWS RDS Instant

In this step you create an Amazon RDS MySQL DB instance that maintains the data used by a web application.

**To launch a MySQL DB instance**

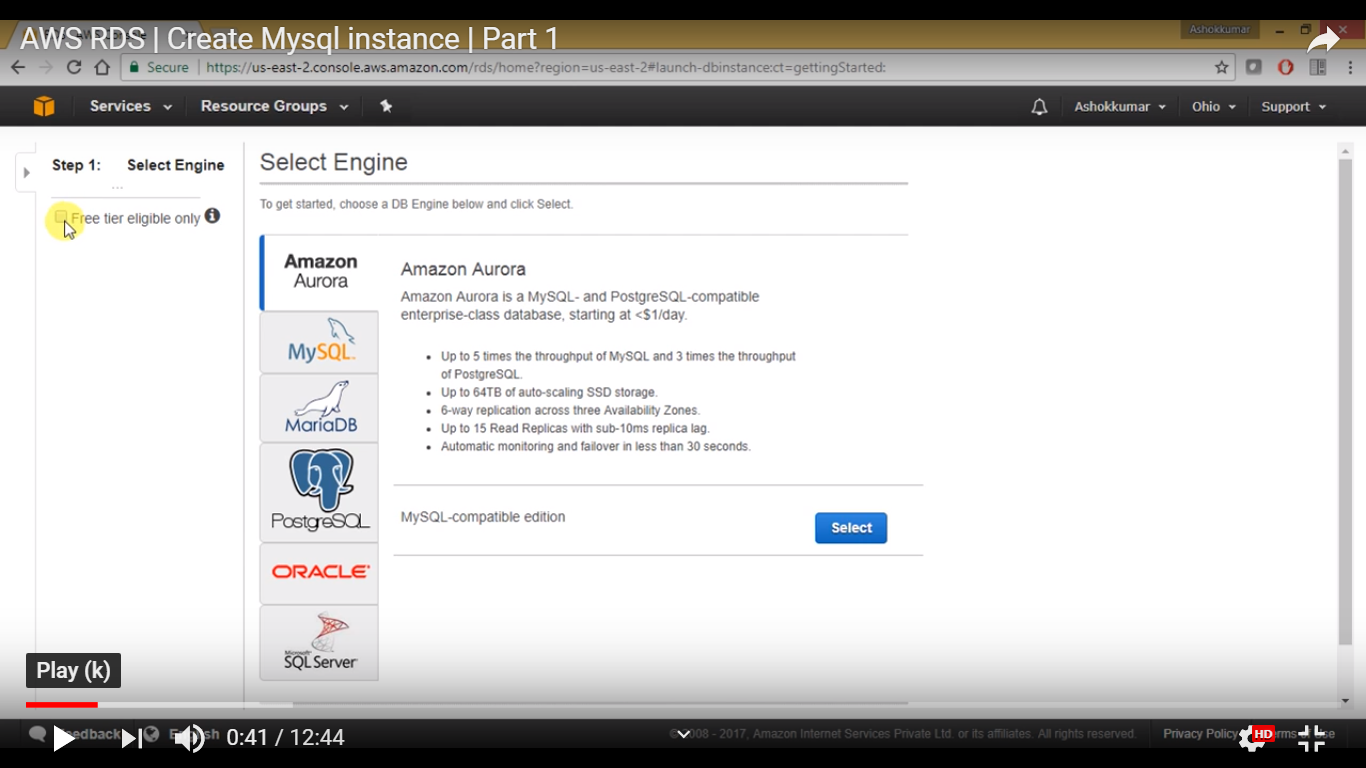
1. Sign in to the AWS Management Console and open the Amazon RDS console athttps://console.aws.amazon.com/ search service RDS



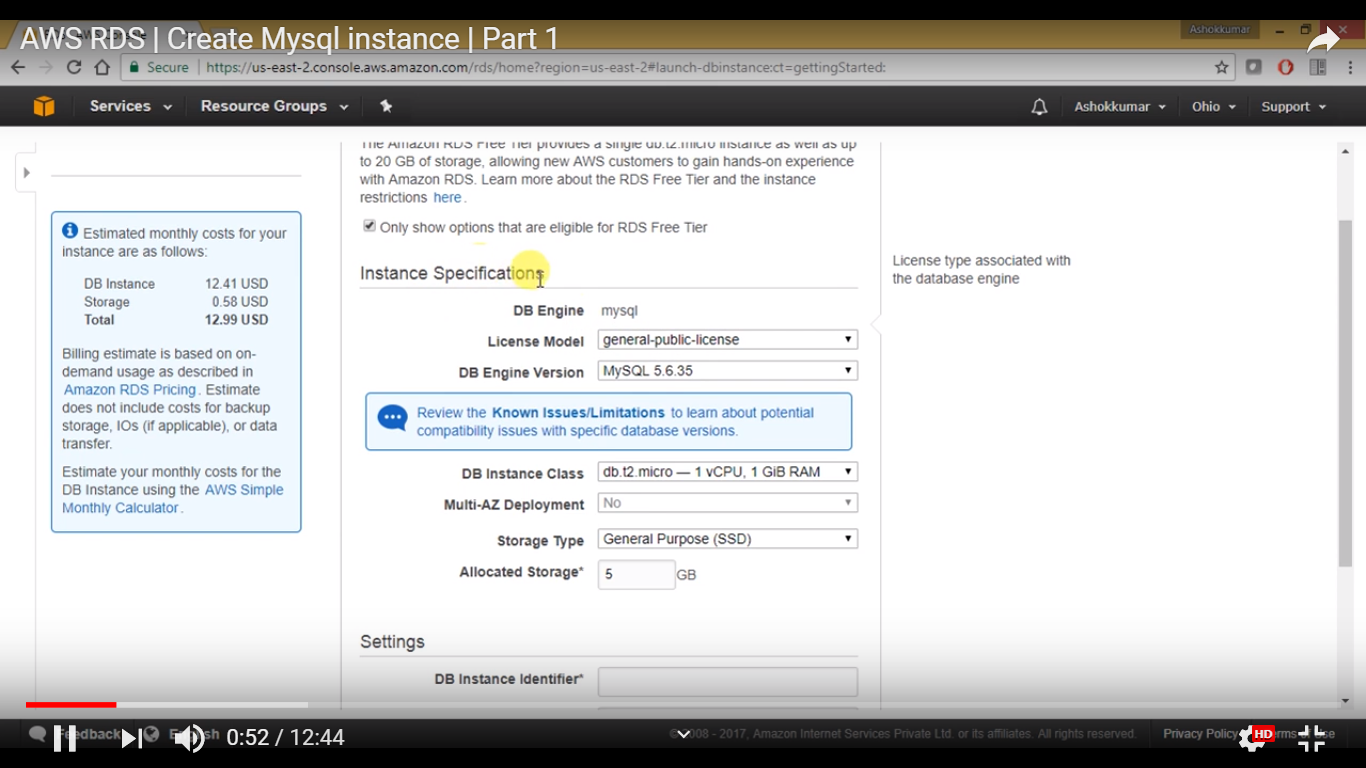
1. In the top-right corner of the AWS Management Console, choose the AWS Region in which you want to create the DB instance. This example uses the US West (Oregon) region.
2. In the navigation pane, choose **Databases**.

If the navigation pane is closed, choose the menu icon at the top left to open it.

1. Choose **Create database** to open the **Select engine** page.
2. On the **Select engine** page, shown following, You can also find which is free tier eligble Database mark on checkbox. Here we use free mySql and press select.



1. On the **Choose use case** page, choose **Dev/Test – MySQL**, and then choose **Next**.



1. On the Instance Specifications page, shown following, set these values:
   * **License model:** Use the default value.
   * **DB engine version:** Use the default value.
   * **DB instance class:** db.t2.small
   * **Multi-AZ deployment:** No
   * **Storage type:** General Purpose (SSD)
   * **Allocated storage :** you can give 5 to 20 GB
   * **DB instance identifier:** tutorial-db-instance
   * **Master username:** user\_name
   * **Master password:** Choose a password.
   * **Confirm password:** Retype the password.
2. Choose **Next** and set the following values in the **Configure advanced settings** page:
   * **Virtual Private Cloud (VPC):** Choose an existing VPC with both public and private subnets, such as the tutorial-vpc (vpc-*identifier*) created in [Create a VPC with Private and Public Subnets](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Tutorials.WebServerDB.CreateVPC.html#CHAP_Tutorials.WebServerDB.CreateVPC.VPCAndSubnets)

**Note**

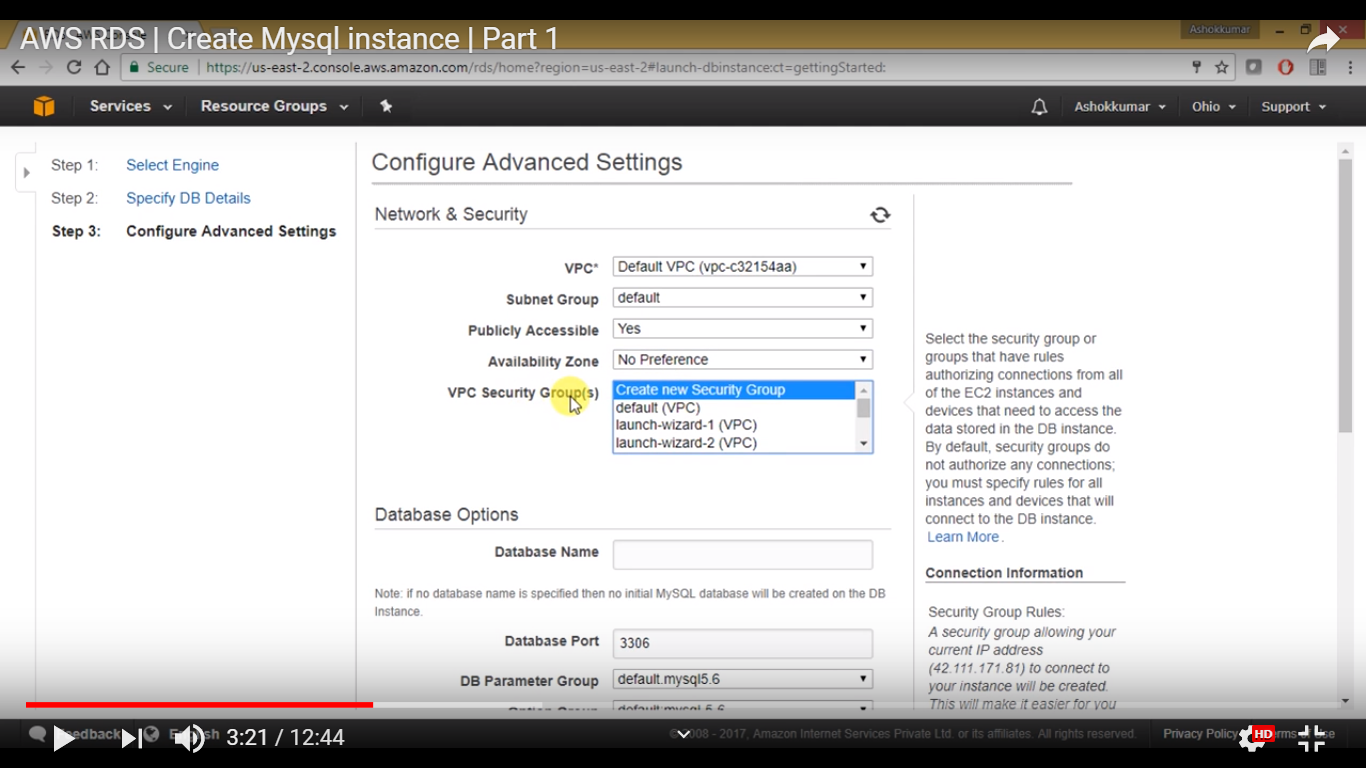
The VPC must have subnets in different Availability Zones.

* + **Subnet group:** The DB subnet group for the VPC, such as the tutorial-db-subnet-group created in [Create a DB Subnet Group](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Tutorials.WebServerDB.CreateVPC.html#CHAP_Tutorials.WebServerDB.CreateVPC.DBSubnetGroup)
  + **Public accessibility:** **No**
  + **Availability zone:** **No Preference**
  + **VPC security groups:** Choose an existing VPC security group that is configured for private access, such as the tutorial-db-securitygroup created in [Create a VPC Security Group for a Private DB Instance](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Tutorials.WebServerDB.CreateVPC.html#CHAP_Tutorials.WebServerDB.CreateVPC.SecurityGroupDB).

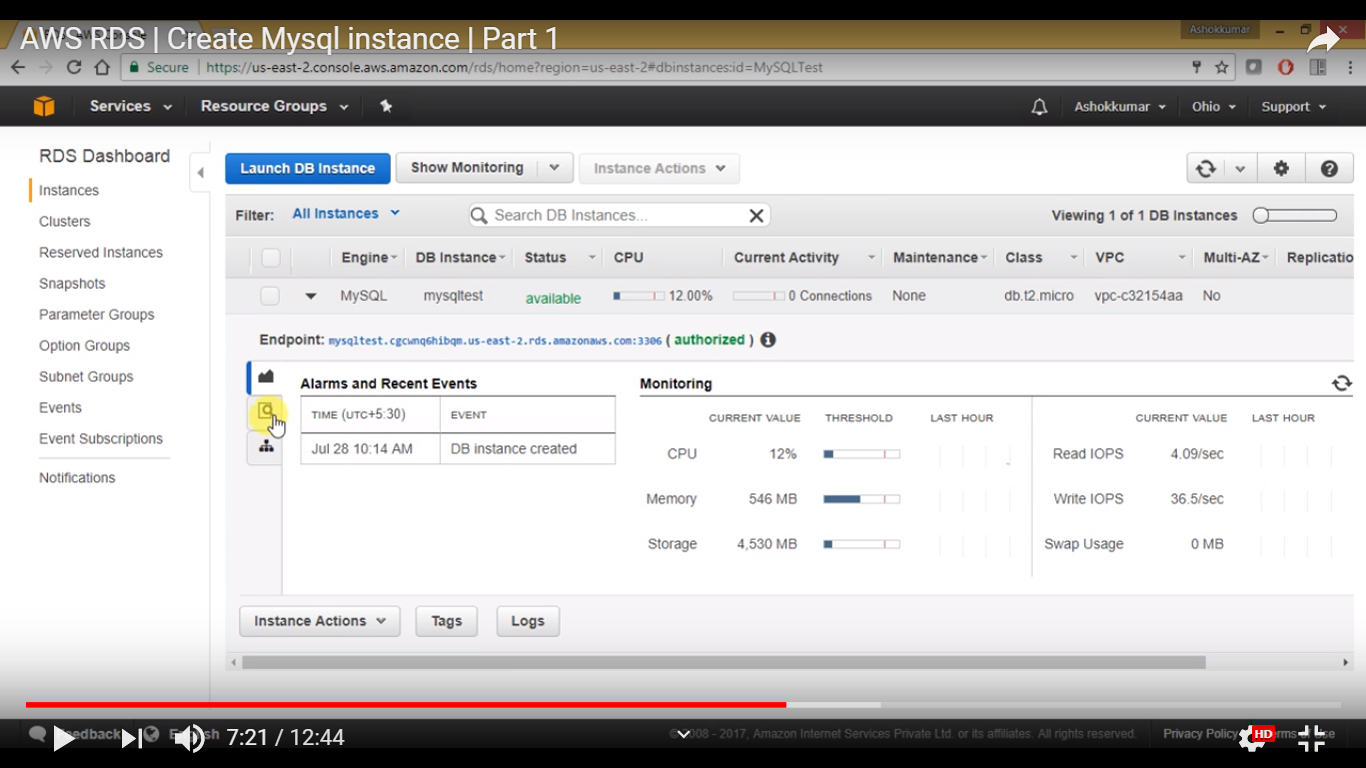
Remove other security groups, such as the default security group, by choosing the **X**associated with each.

* + **Database name:** sample

Leave the default settings for the other options.



1. To create your Amazon RDS MySQL DB instance, choose **Create database**.
2. On the next page, choose **View DB instances details** to view your RDS MySQL DB instance.
3. Wait for the **DB instance status** of your new DB instance to show as **available**. Then scroll to the **Connect** section, shown following.



Make note of the endpoint and port for your DB instance. We will use this information to connect our web server to our RDS DB instance.

1. Create Database and Tables which we use in Our web-Service.

# a) Configuration Mysql/ AWS RDS Database JDBC property configuration in src\main\application.properties file.

# Step – 8 Create shared jar file –project/new/Run As/Maven build

# 

### Step- 9 set Goals -> package shade:shade in Edit Configuration and click run.

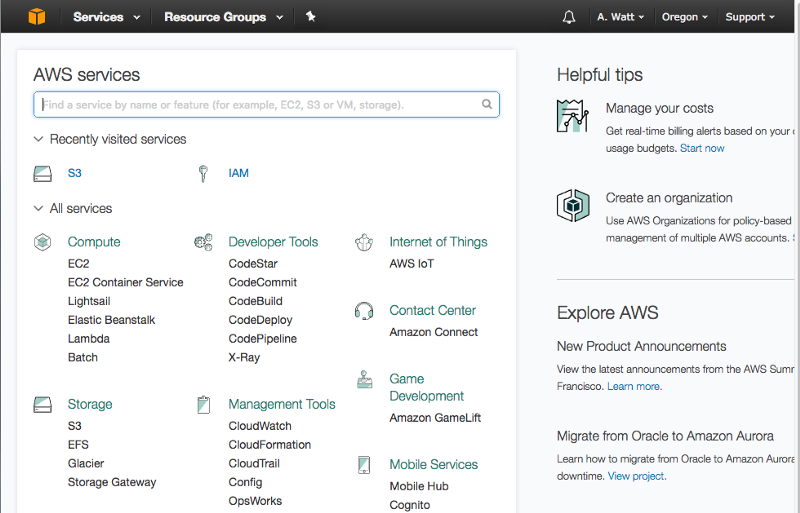
### 

### Step -10 This Shop\_now\_lambda\_webservices-1.0.0-shaded.jar file upload on Lambda Function

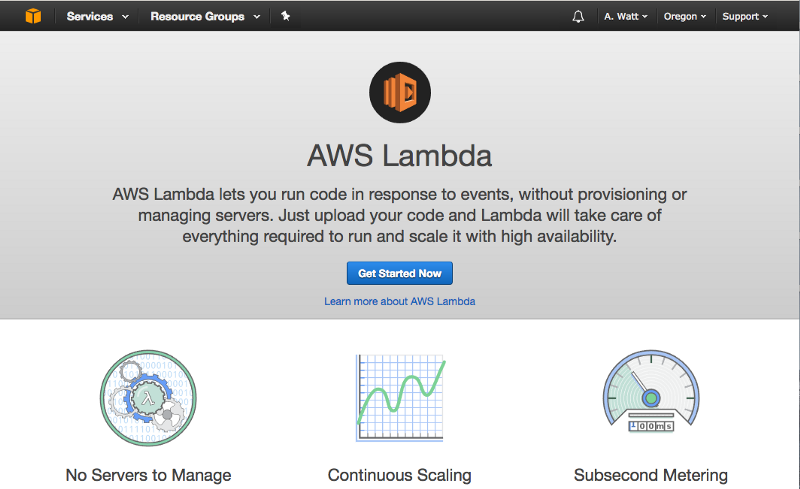
In this step we will be heading to the AWS Console to create the Lambda Function:

## Step-11 Create Lambda Function Via Management Console.

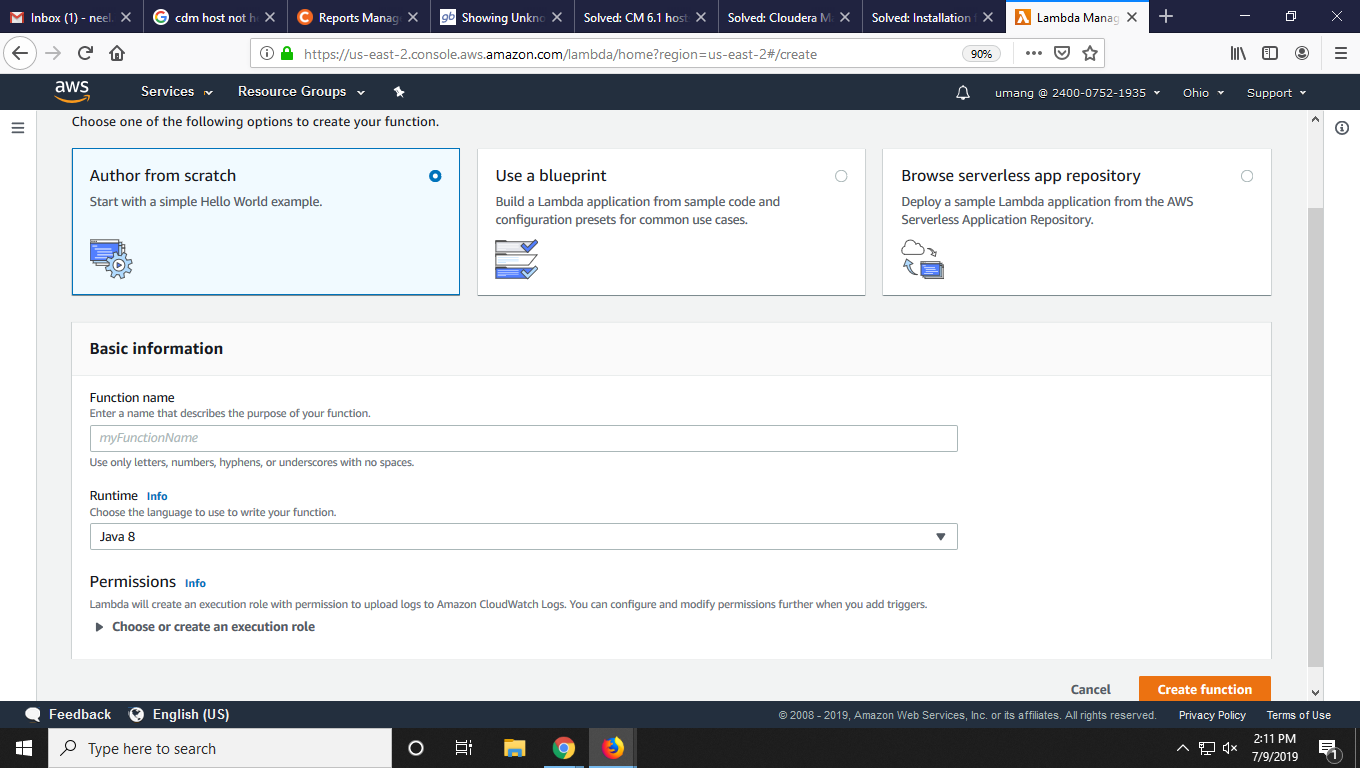
Here are the steps required to create our lambda



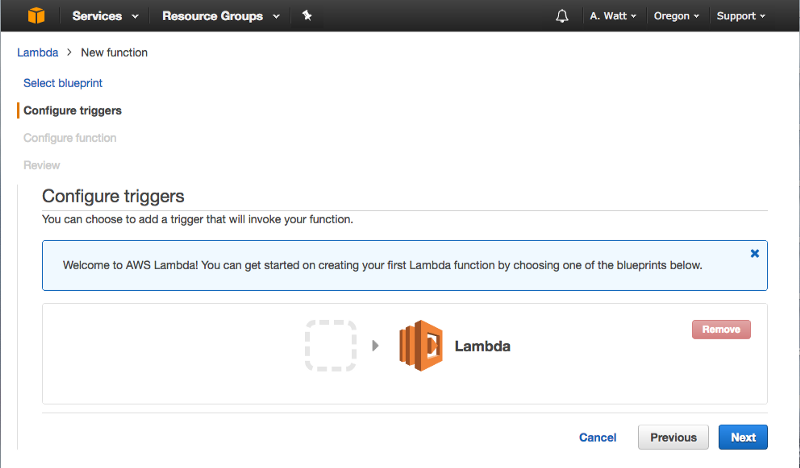
* And then press “Get Started Now.”



* For runtime select **java8,** Give Function nameand then press “Create Function.”



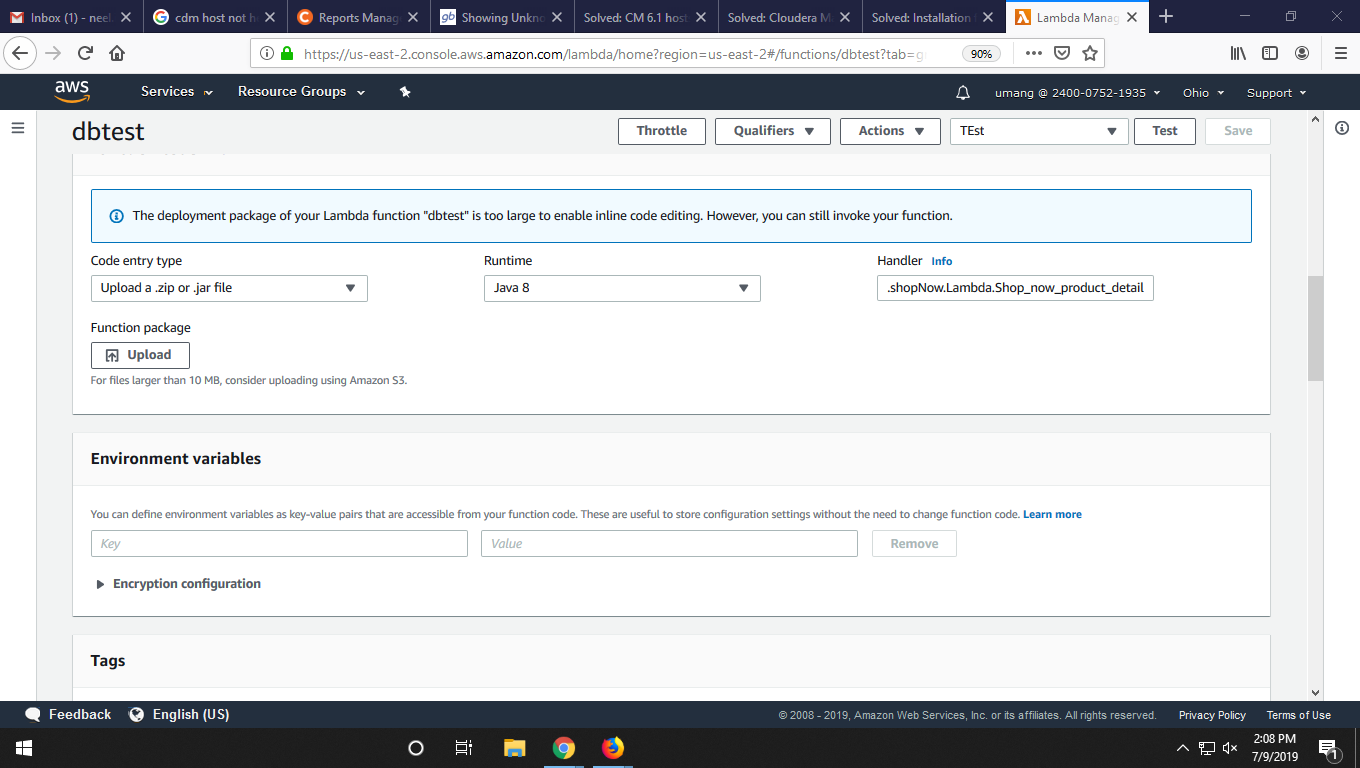
* Skip this step and press “Next.”



* **“Configure function”:**
* Name: Provide **MethodHandlerLambda**,
* Description: Anything that describes our lambda function
* Runtime: Select **java8**
* Under **Lambda function handler and role**:
* Role name: If any other AWS resources are used in lambda function, then provide access by creating/using existing role and also define the policy template.
  + Under**Advanced settings:**
    - Memory: Provide memory that will be used by our lambda function.
    - Timeout: Select a time for execution of lambda function for each request.
* Once you are done with all inputs, click “**Next”** which will show you to review the configuration.
* Once a review is completed, click on “**Create Function”**.



* Code Entry Type and Function Package: Select “**Upload a .ZIP and Jar file”** and click on “**Upload”** button. Select the file which contains lambda code.
* Runtime select **Java8**
* Handler name: Provide lambda function handler name  **Package\_Name.Class\_Name.**



## Step-11. Invoke the Function

Once the AWS lambda function is created, we’ll test it by passing in some data:

* Click on your lambda function from lists and then click on “**Test”** button
* A popup window will appear which contains dummy value for sending data. Override the data with {**“id”:113”}**
* Click on “**Save and test”** button

On the screen, you can see the **Execution result** section with successfully returned output.

**AWS Lambda function call using the AWS API Gateway.**

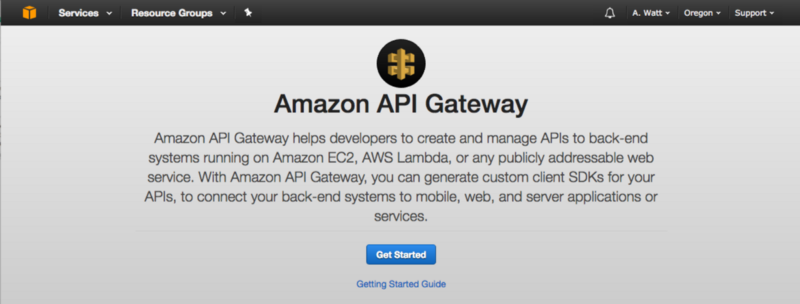
We need to setup an API Gateway instance first that handles those verbs.

In this section

### Creating the API Gateway

### Go to your AWS Console and press “API Gateway”. And then press “Get Started.”

### https://cdn-images-1.medium.com/max/800/1*ZclBoLSJm0mOx1-U_46J6w.png



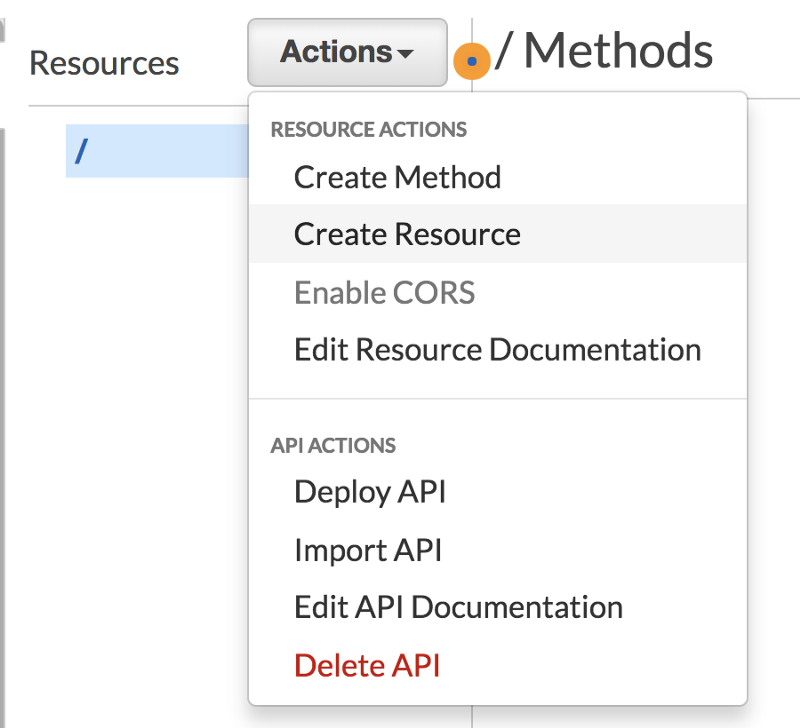
### Then click on *Create API*, and enter a name

### C:\Users\admin\Desktop\AWS7.png

Clicking “Create API”will get us into the configuration page for the API.

The first thing we need to do is to add a resource onto the API. Using resources allows us to group similar API calls together using nested slashes.

Click the “Actions” dropdown and select “Create Resource”*.* Name your resources, making sure that they are both in the “/” path.

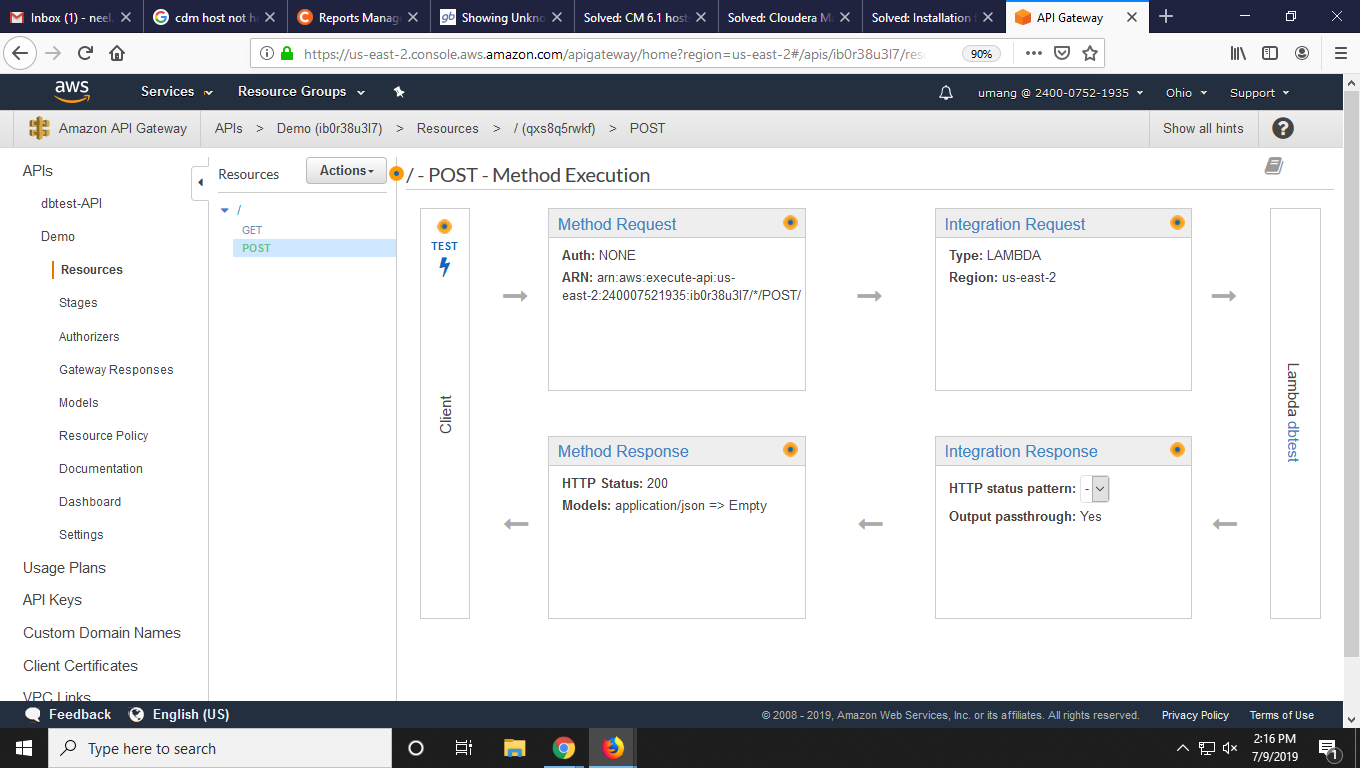


### Create New Child Resource

### C:\Users\admin\Desktop\AWS8.png

#### Connecting the Lambdas to API Gateway

Back in API Gateway, we can add our new Lambdas to the methods we created earlier. We need to make sure that “Use Lambda Proxy integration”is selected and that we are pointing at the correct Lambda. Clicking “Save” will ask you for permissions to access this Lambda, to which we can give the “OK”



### Step-12 Save and Test AWS Lambda Function.

### C:\Users\admin\Desktop\AWS5.png

### Do this for the POSTmethods on both resources and we can start to test. Select Stage and function name –POST -/ method we get invoke –URL copy this url and test on ARC give needed parameter and we get output of lambda function.

### C:\Users\admin\Desktop\AWS6.png

### Reference Link:

### <https://docs.aws.amazon.com/lambda/latest/dg/java-programming-model.html>

### <https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-getting-started-with-rest-apis.html>