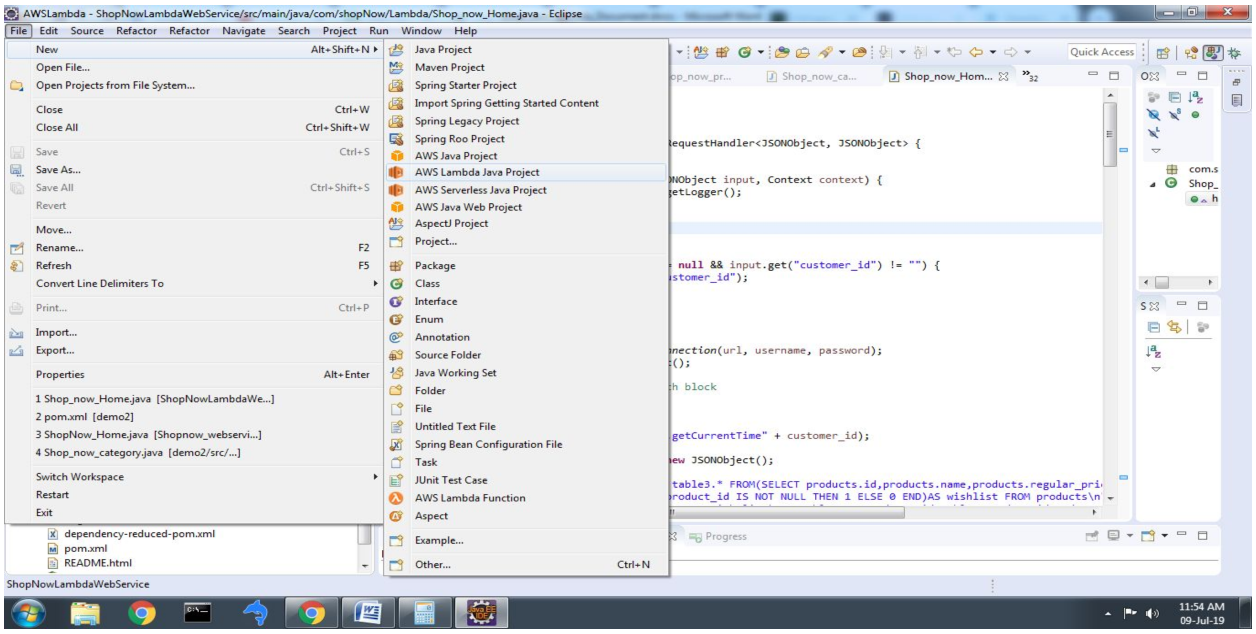
Shop_now AWS Lambda Web Service

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 and choose **Create an AWS Account**.

Step-2 Install the AWS Toolkit for Eclipse

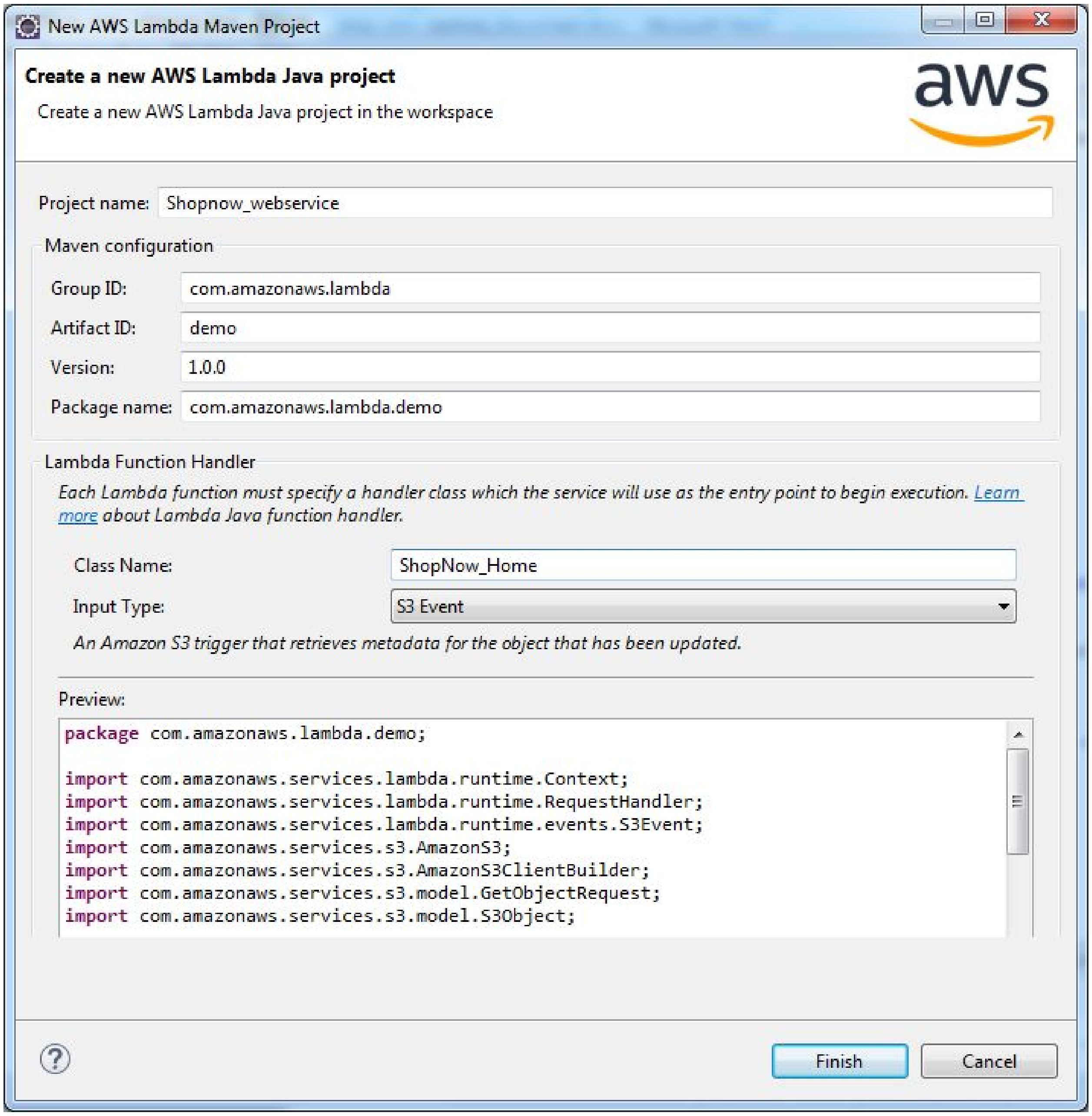
1. Within Eclipse, click **Help** and then click **Install New Software**.
2. In the **Work with** box, type https://aws.amazon.com/eclipse and then press Enter.
3. Choose the components of the AWS Toolkit for Eclipse that you want to install. Click **Select All** to install all components at once.

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



```
equestHandler<JSONObject, JSONObject> {

NObject input, Context context) {
etLogger();

 null && input.get("customer_id") != "") {
stomer_id");

nection(url, username, password);
();
h block

getCurrentTime" + customer_id);
ew JSONObject();
table3.* FROM(SELECT products.id,products.name,products.regular_pri
roduct_id IS NOT NULL THEN 1 ELSE 0 END)AS wishlist FROM products\n
```

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



Create a new AWS Lambda Java project

Create a new AWS Lambda Java project in the workspace

Project name: Shopnow_webservice

Maven configuration

Group ID: com.amazonaws.lambda

Artifact ID: demo

Version: 1.0.0

Package name: com.amazonaws.lambda.demo

Lambda Function Handler

Each Lambda function must specify a handler class which the service will use as the entry point to begin execution. Learn more about Lambda Java function handler.

Class Name: ShopNow_Home

Input Type: S3 Event

An Amazon S3 trigger that retrieves metadata for the object that has been updated.

Preview:

```
package com.amazonaws.lambda.demo;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.GetObjectRequest;
import com.amazonaws.services.s3.model.S3Object;
```

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>
```

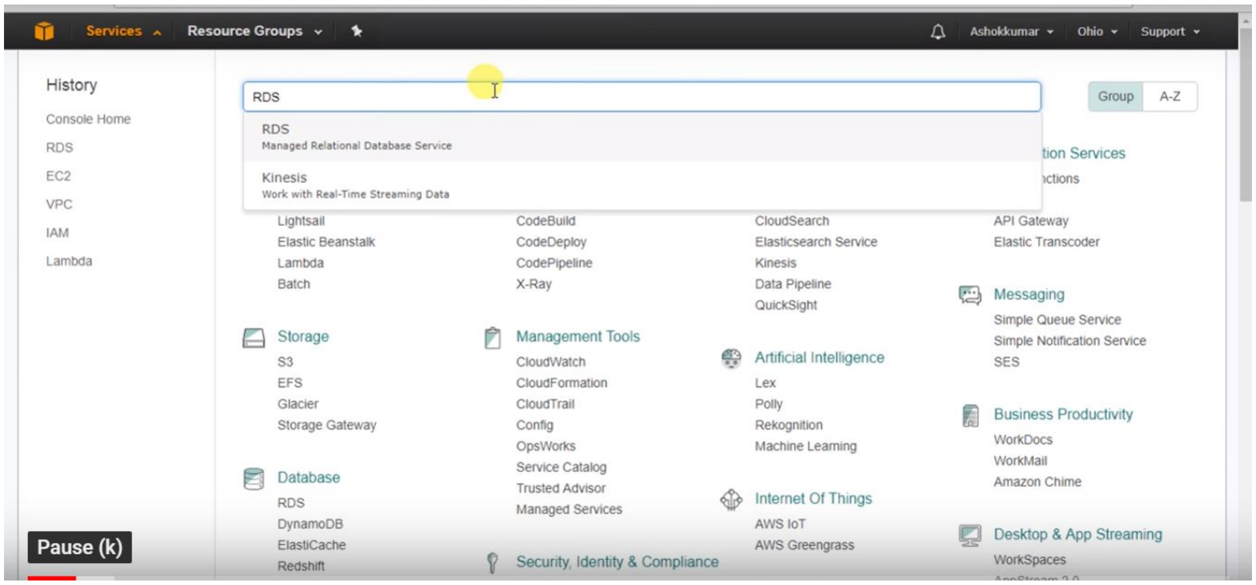
Step- 7 Database connectivity

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

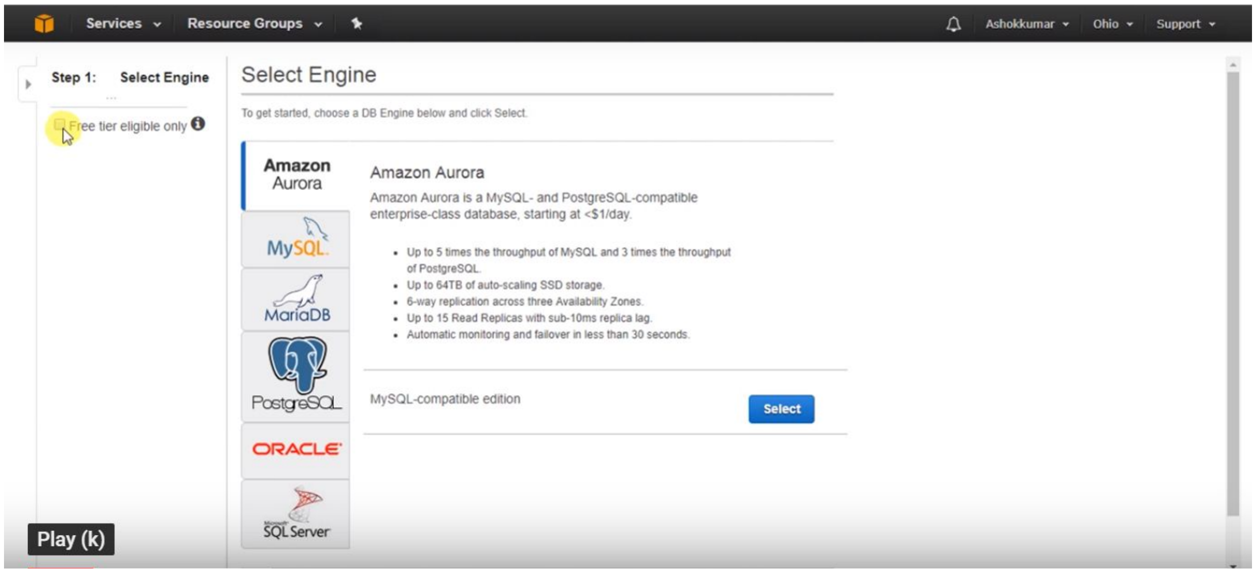
1. Sign in to the AWS Management Console and open the Amazon RDS console athttps://console.aws.amazon.com/ search service RDS



2. 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.
3. In the navigation pane, choose **Databases**.

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

4. Choose **Create database** to open the **Select engine** page.
5. 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.



Select Engine

To get started, choose a DB Engine below and click Select.

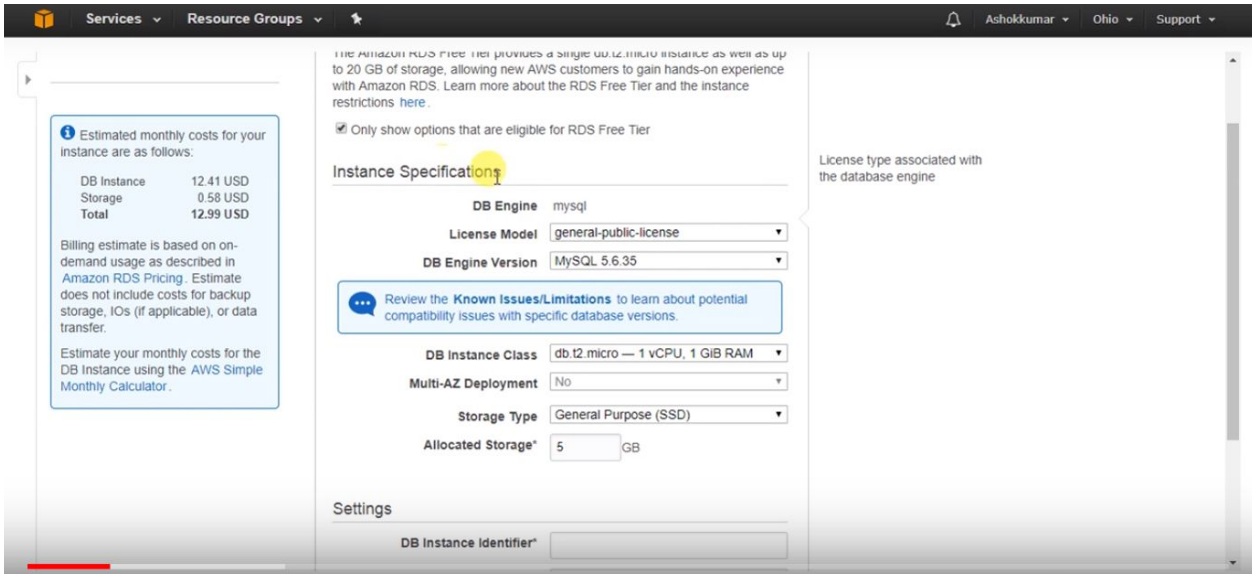
Amazon Aurora

Amazon Aurora is a MySQL- and PostgreSQL-compatible enterprise-class database, starting at <$1/day.

- Up to 5 times the throughput of MySQL and 3 times the throughput of PostgreSQL.
- Up to 64TB of auto-scaling SSD storage.
- 6-way replication across three Availability Zones.
- Up to 15 Read Replicas with sub-10ms replica lag.
- Automatic monitoring and failover in less than 30 seconds.

MySQL-compatible edition

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



The Amazon RDS Free Tier provides a single db.t2.micro instance as well as up to 20 GB of storage, allowing new AWS customers to gain hands-on experience with Amazon RDS. Learn more about the RDS Free Tier and the instance restrictions here.

☑ Only show options that are eligible for RDS Free Tier

Instance Specifications

DB Engine	mysql
License Model	general-public-license
DB Engine Version	MySQL 5.6.35

Review the Known Issues/Limitations to learn about potential compatibility issues with specific database versions.

DB Instance Class	db.t2.micro — 1 vCPU, 1 GiB RAM
Multi-AZ Deployment	No
Storage Type	General Purpose (SSD)
Allocated Storage*	5 GB

Settings

DB Instance Identifier*

Estimated monthly costs for your instance are as follows:

DB Instance	12.41 USD
Storage	0.58 USD
Total	12.99 USD

Billing estimate is based on on-demand usage as described in Amazon RDS Pricing. Estimate does not include costs for backup storage, IOs (if applicable), or data transfer.

Estimate your monthly costs for the DB Instance using the AWS Simple Monthly Calculator.

License type associated with the database engine

7. 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.

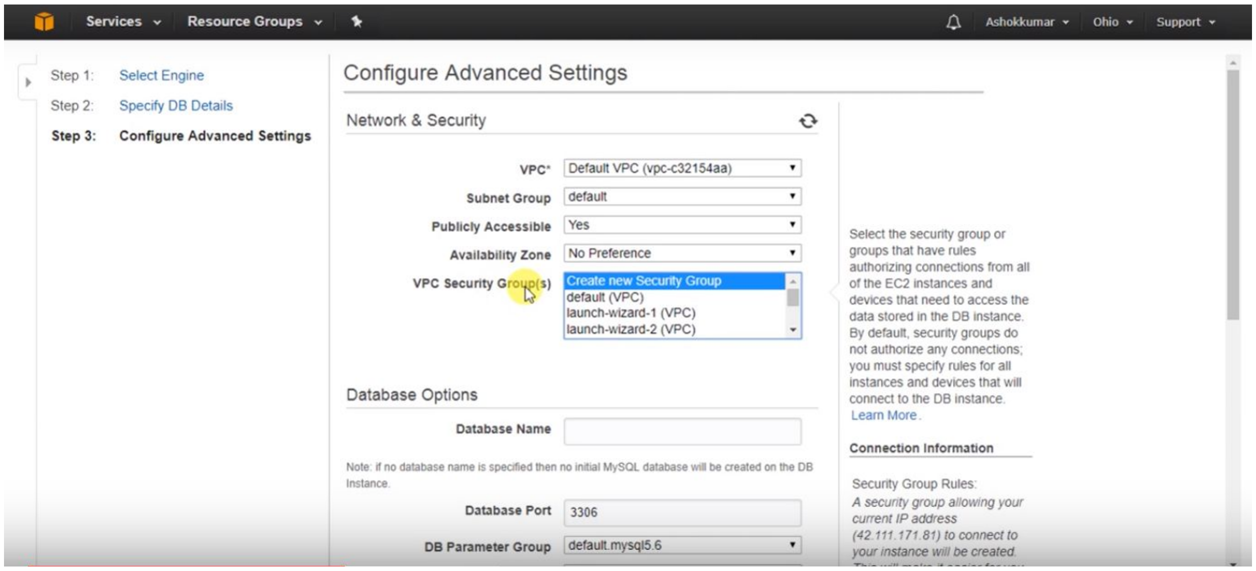
8. Choose **Next** and set the following values in the **Configure advanced settings** page:
   - **Virtual Private Cloud (VPC):** Choose an existing VPC with both public and private subnets, such as the tutorial-vpc (vpc-*identifier*) created in Create a VPC with Private and Public Subnets

     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

- **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.
  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.



Step 1: Select Engine
Step 2: Specify DB Details
Step 3: Configure Advanced Settings

Configure Advanced Settings

Network & Security

VPC* Default VPC (vpc-c32154aa)
Subnet Group default
Publicly Accessible Yes
Availability Zone No Preference
VPC Security Group(s) Create new Security Group / default (VPC) / launch-wizard-1 (VPC) / launch-wizard-2 (VPC)

Select the security group or groups that have rules authorizing connections from all of the EC2 instances and devices that need to access the data stored in the DB instance. By default, security groups do not authorize any connections; you must specify rules for all instances and devices that will connect to the DB instance. Learn More.

Database Options

Database Name

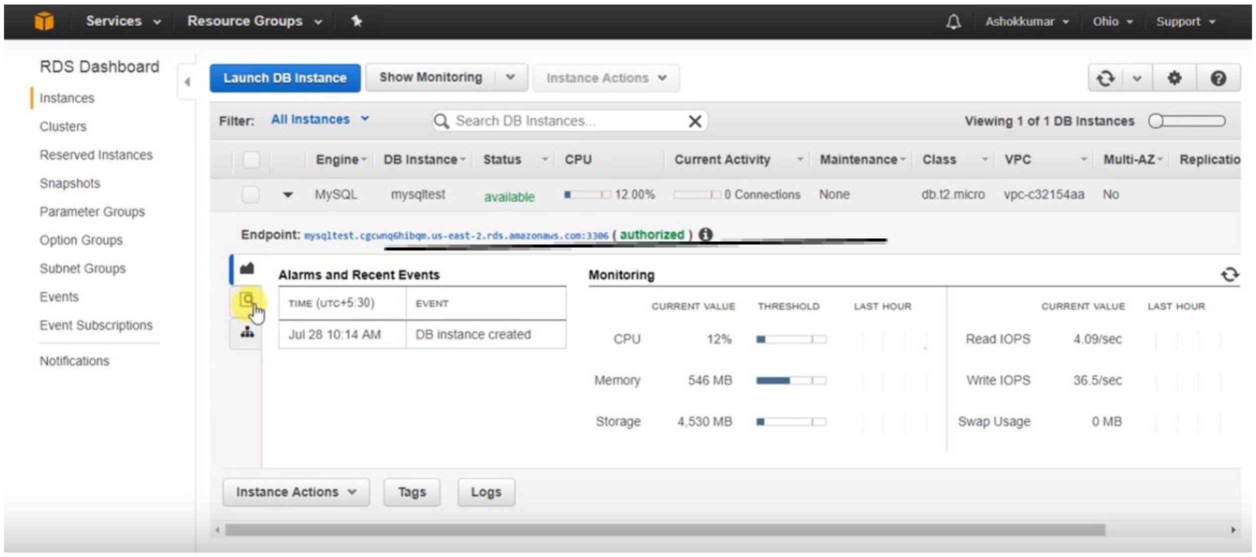
Note: if no database name is specified then no initial MySQL database will be created on the DB Instance.

Database Port 3306
DB Parameter Group default.mysql5.6

Connection Information

Security Group Rules:
A security group allowing your current IP address (42.111.171.81) to connect to your instance will be created.

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



Engine	DB Instance	Status	CPU	Current Activity	Maintenance	Class	VPC	Multi-AZ	Replicatio
MySQL	mysqltest	available	12.00%	0 Connections	None	db.t2.micro	vpc-c32154aa	No	

Endpoint: mysqltest.cgcwnq6hibqm.us-east-2.rds.amazonaws.com:3306 (authorized)

Alarms and Recent Events

TIME (UTC+5:30)	EVENT
Jul 28 10:14 AM	DB instance created

Monitoring

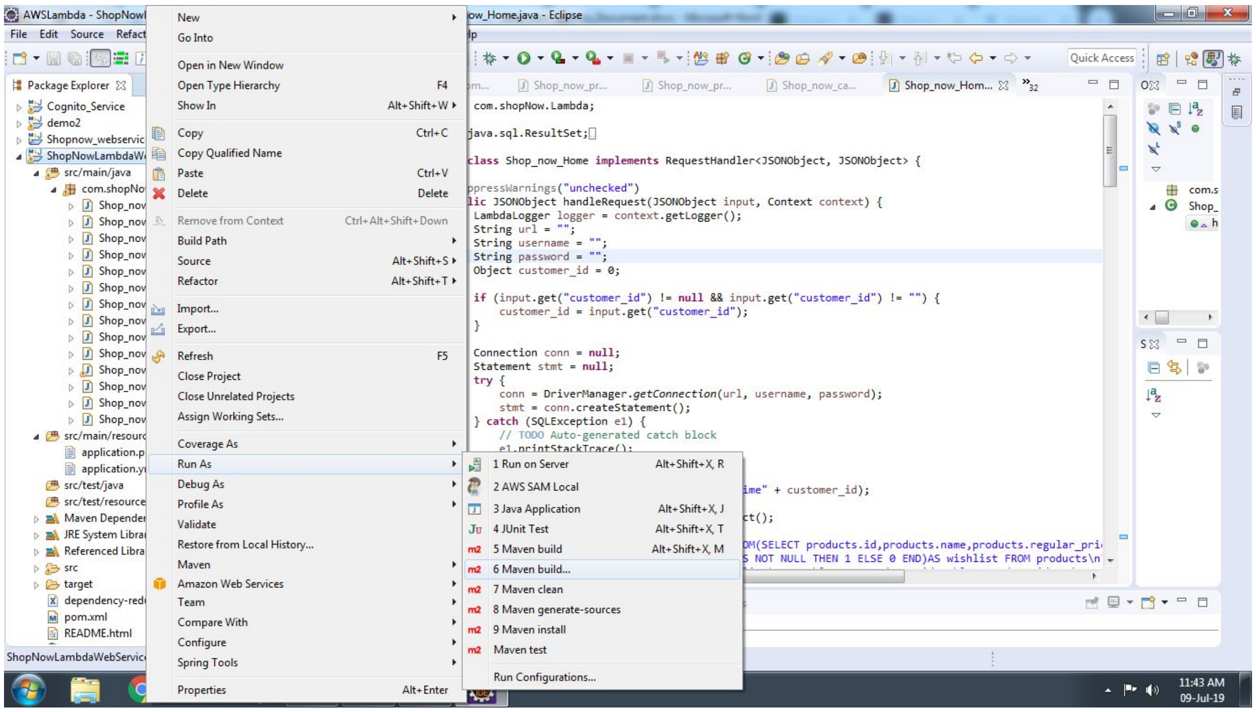
	CURRENT VALUE	THRESHOLD	LAST HOUR		CURRENT VALUE	LAST HOUR
CPU	12%			Read IOPS	4.09/sec	
Memory	546 MB			Write IOPS	36.5/sec	
Storage	4,530 MB			Swap Usage	0 MB	

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.

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

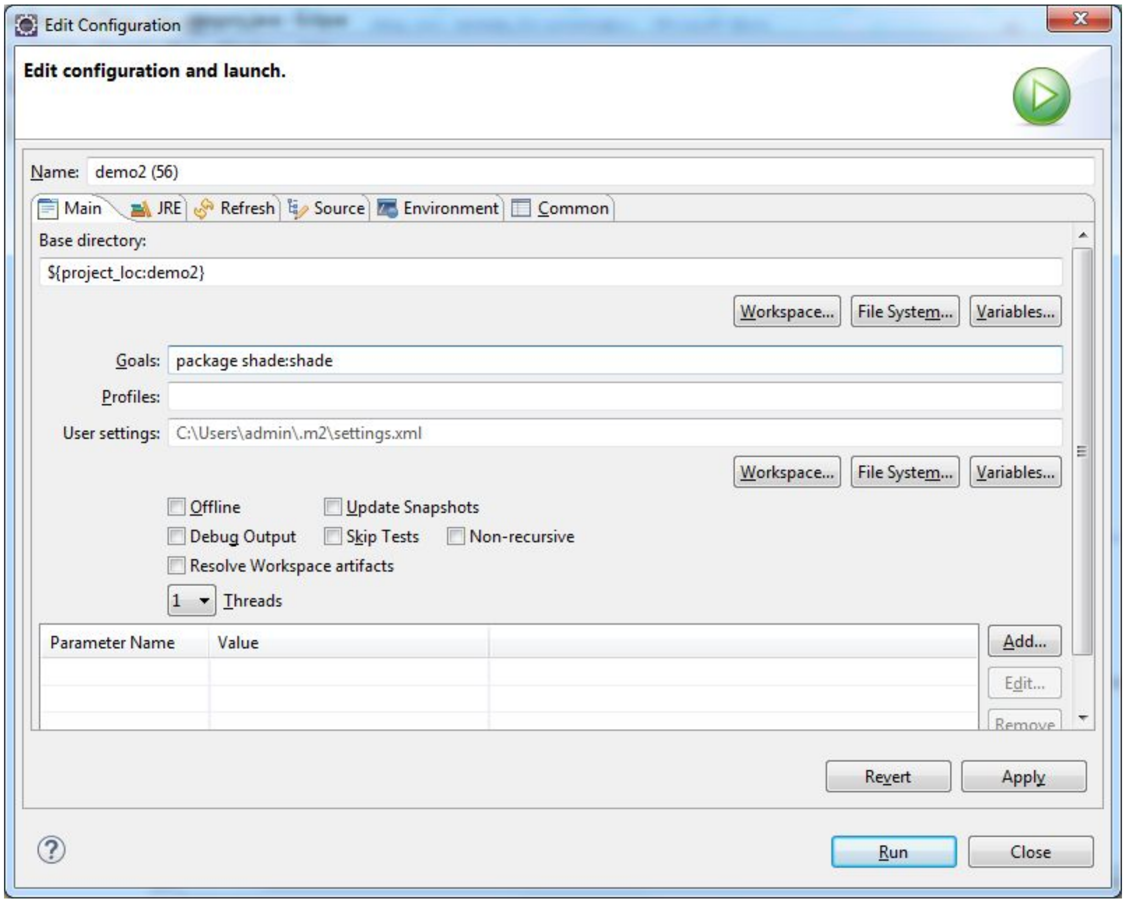
a) Configuration Mysql/ AWS RDS Database JDBC property configuration in **src\main\application.properties** file.

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



```
com.shopNow.Lambda;

java.sql.ResultSet;

class Shop_now_Home implements RequestHandler<JSONObject, JSONObject> {

ppressWarnings("unchecked")
lic JSONObject handleRequest(JSONObject input, Context context) {
 LambdaLogger logger = context.getLogger();
 String url = "";
 String username = "";
 String password = "";
 Object customer_id = 0;

 if (input.get("customer_id") != null && input.get("customer_id") != "") {
     customer_id = input.get("customer_id");
 }

 Connection conn = null;
 Statement stmt = null;
 try {
     conn = DriverManager.getConnection(url, username, password);
     stmt = conn.createStatement();
 } catch (SQLException e1) {
     // TODO Auto-generated catch block
     e1.printStackTrace();

ime" + customer_id);

ct();

OM(SELECT products.id,products.name,products.regular_pri
S NOT NULL THEN 1 ELSE 0 END)AS wishlist FROM products\n
```

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


Edit Configuration

Edit configuration and launch.

Name: demo2 (56)

Base directory:

${project_loc:demo2}

Goals: package shade:shade

Profiles:

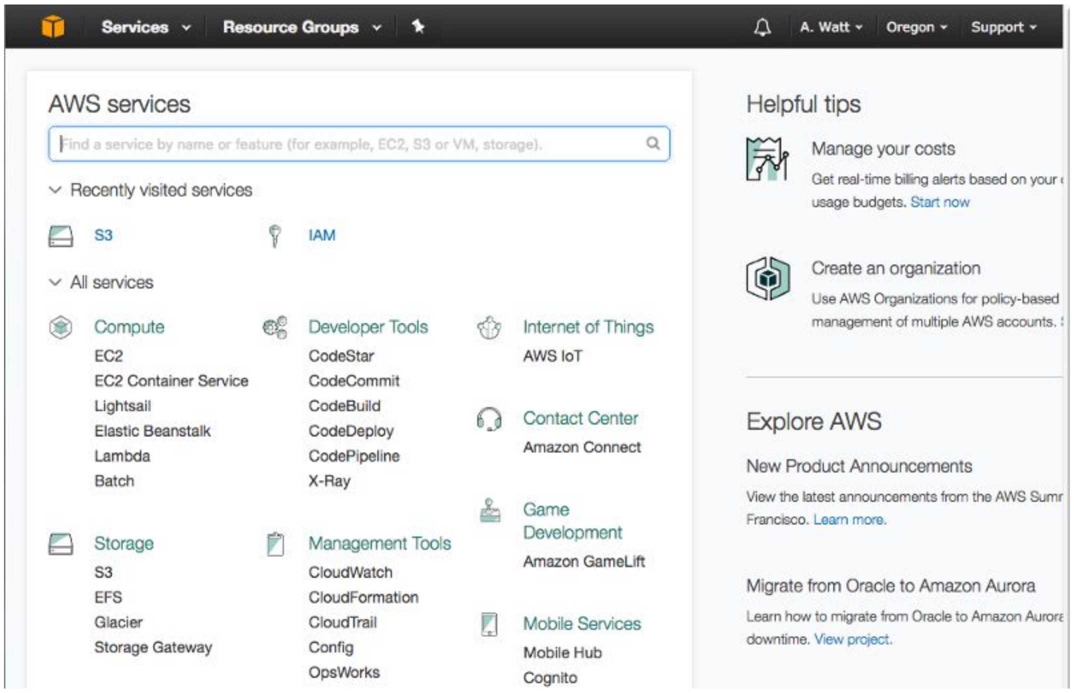
User settings: C:\Users\admin\.m2\settings.xml

Step -10 This Shop_now_lambda_webservices-1.0.0-shaded.jar file upload on Lambda Function

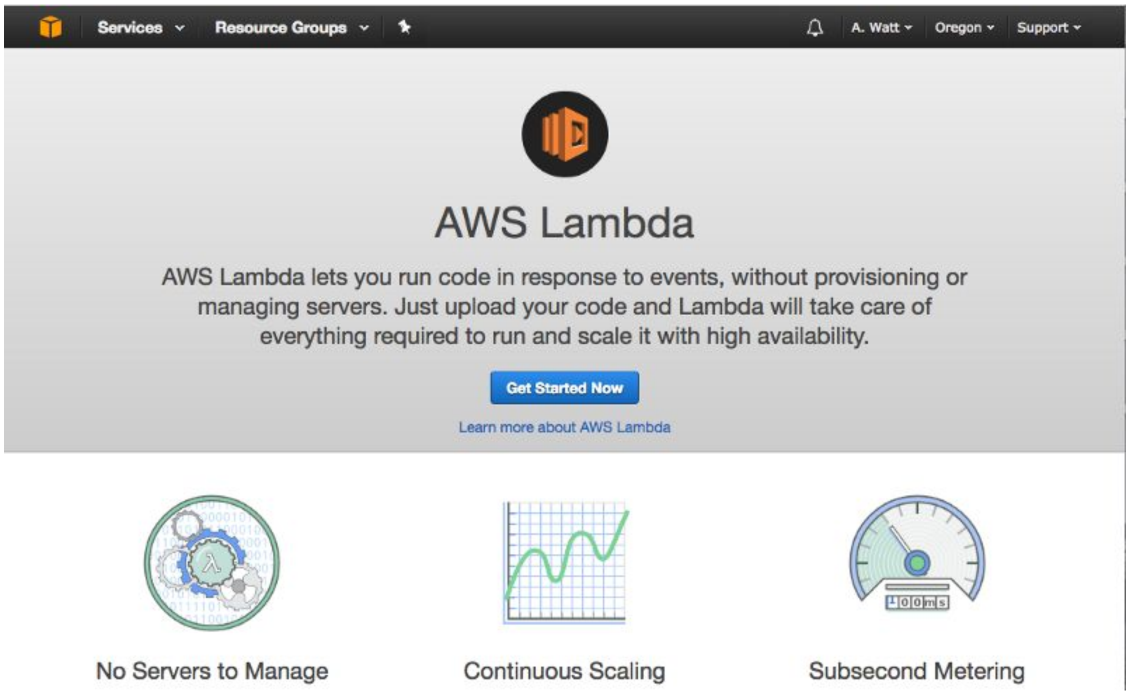
In this step we will be heading to the AWS Console to create the Lambda Function:

Step-11 Create Lambda Function Via Management Console.

Here are the steps required to create our lambda

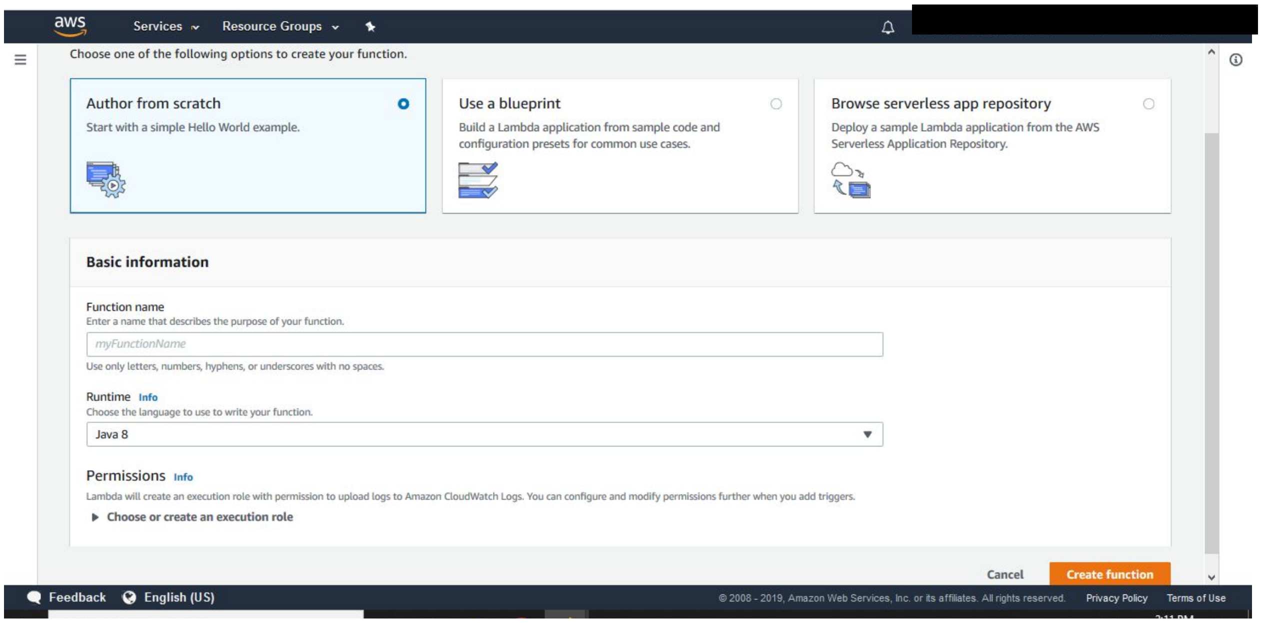


- And then press “Get Started Now.”



AWS Lambda lets you run code in response to events, without provisioning or managing servers. Just upload your code and Lambda will take care of everything required to run and scale it with high availability.

- For runtime select **java8,** Give Function name and then press "Create Function."



[redacted]

Choose one of the following options to create your function.

Author from scratch ●
Start with a simple Hello World example.

Use a blueprint ○
Build a Lambda application from sample code and configuration presets for common use cases.

Browse serverless app repository ○
Deploy a sample Lambda application from the AWS Serverless Application Repository.

Basic information

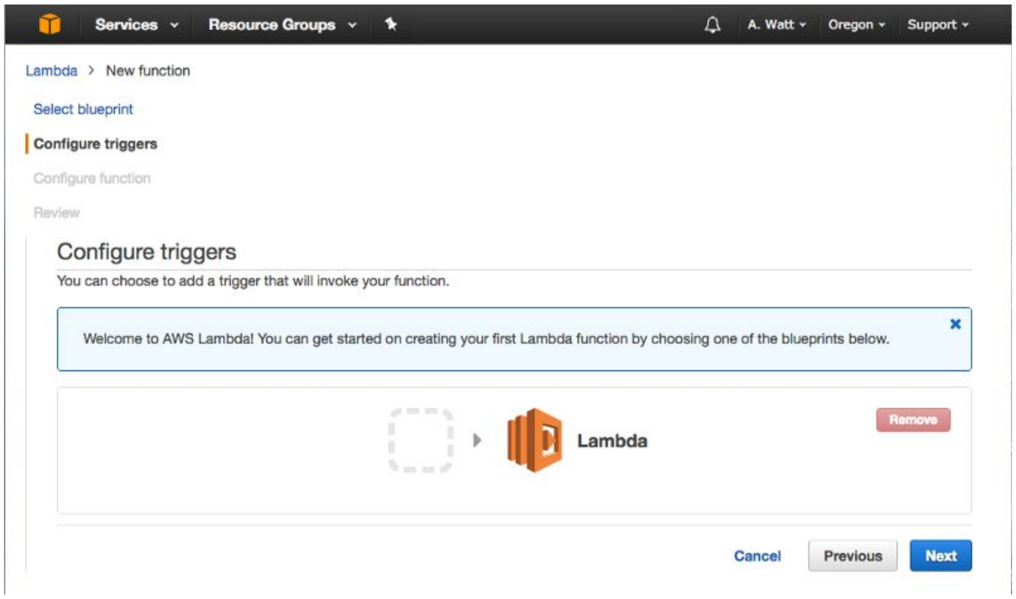
Function name
Enter a name that describes the purpose of your function.
myFunctionName
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function.
Java 8

Permissions Info
Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can configure and modify permissions further when you add triggers.
▶ Choose or create an execution role

© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

- Skip this step and press "Next."



Configure triggers

You can choose to add a trigger that will invoke your function.

Welcome to AWS Lambda! You can get started on creating your first Lambda function by choosing one of the blueprints below.

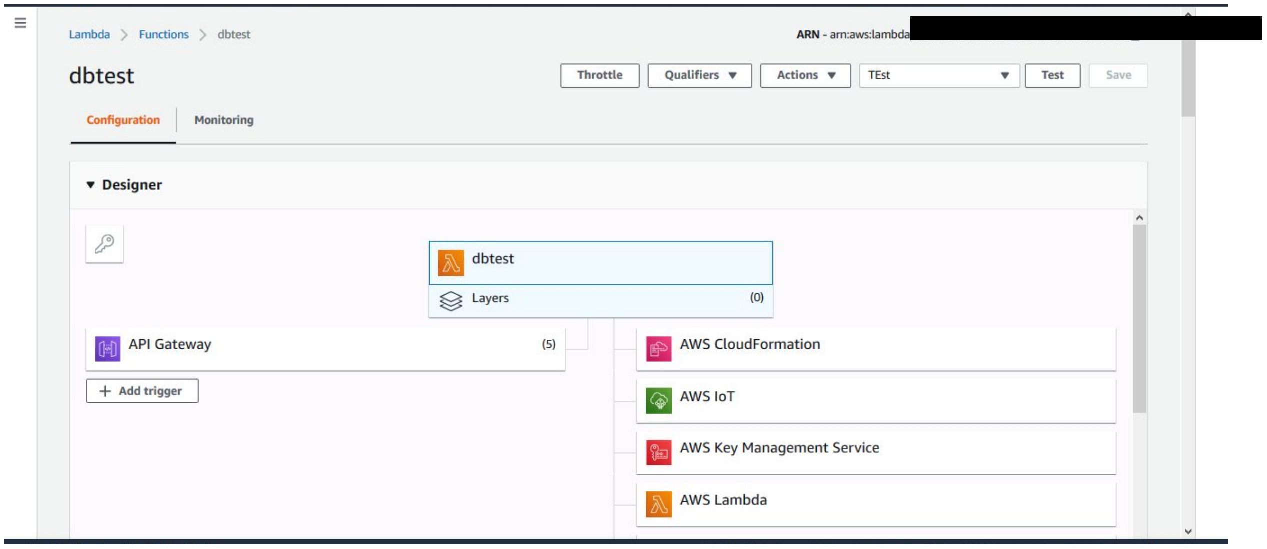
“Configure function”:

- Name: Provide **MethodHandlerLambda**,
- Description: Anything that describes our lambda function
- Runtime: Select **java8**

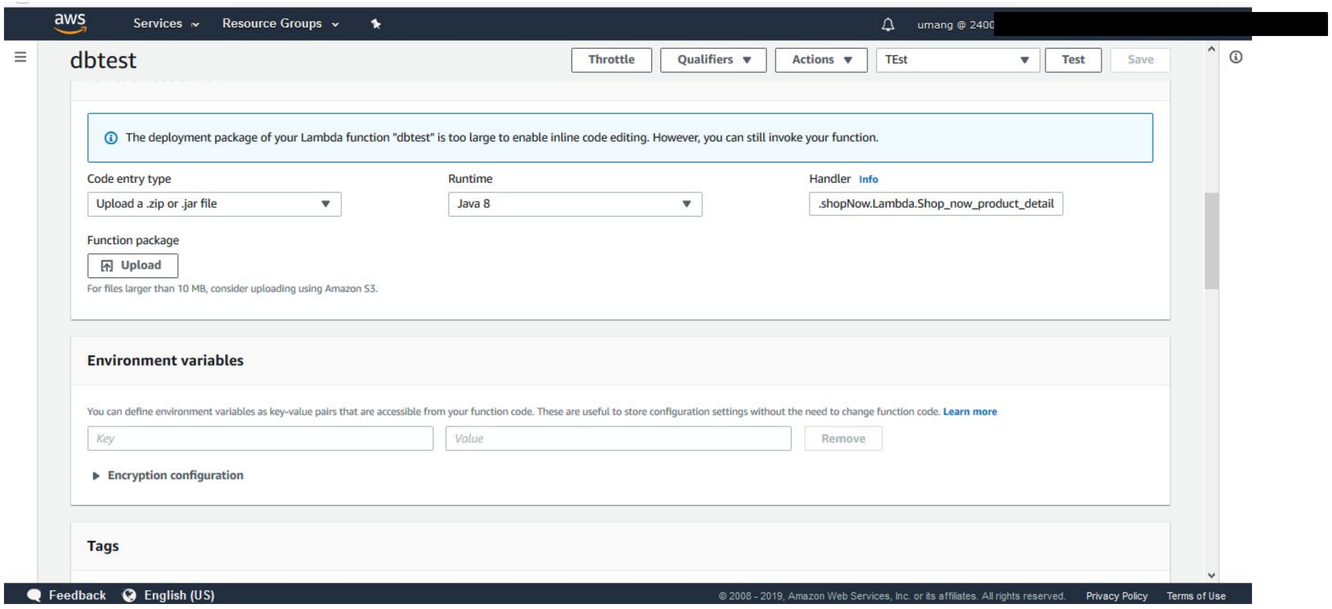
Under **Lambda function handler and role**:

- 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.**



The deployment package of your Lambda function "dbtest" is too large to enable inline code editing. However, you can still invoke your function.

Code entry type: Upload a .zip or .jar file

Runtime: Java 8

Handler Info: .shopNow.Lambda.Shop_now_product_detail

Function package

For files larger than 10 MB, consider uploading using Amazon S3.

Environment variables

You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. Learn more

Key | Value

▸ Encryption configuration

Tags

© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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

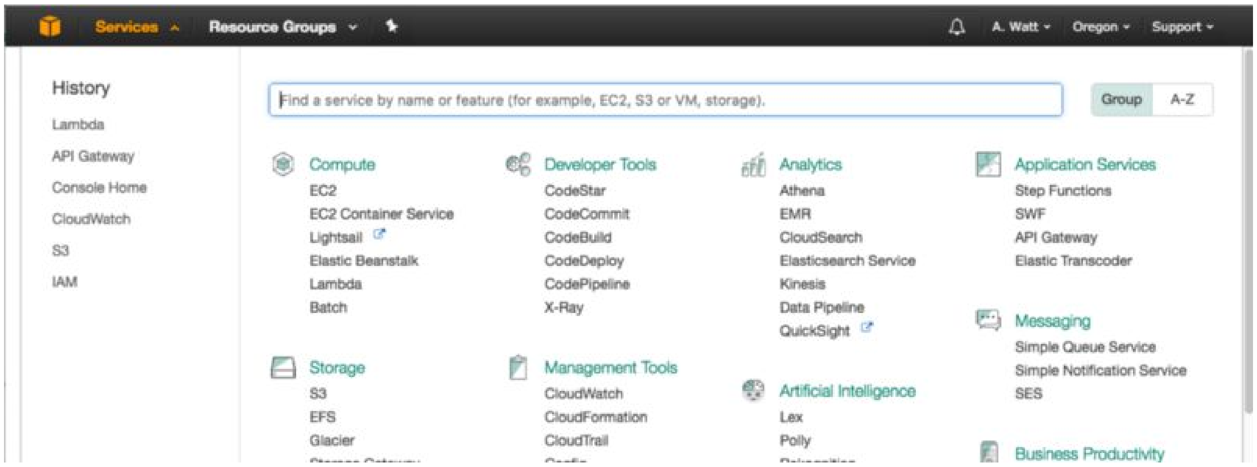
On the screen, you can see the **Execution result** section with successfully returned output.

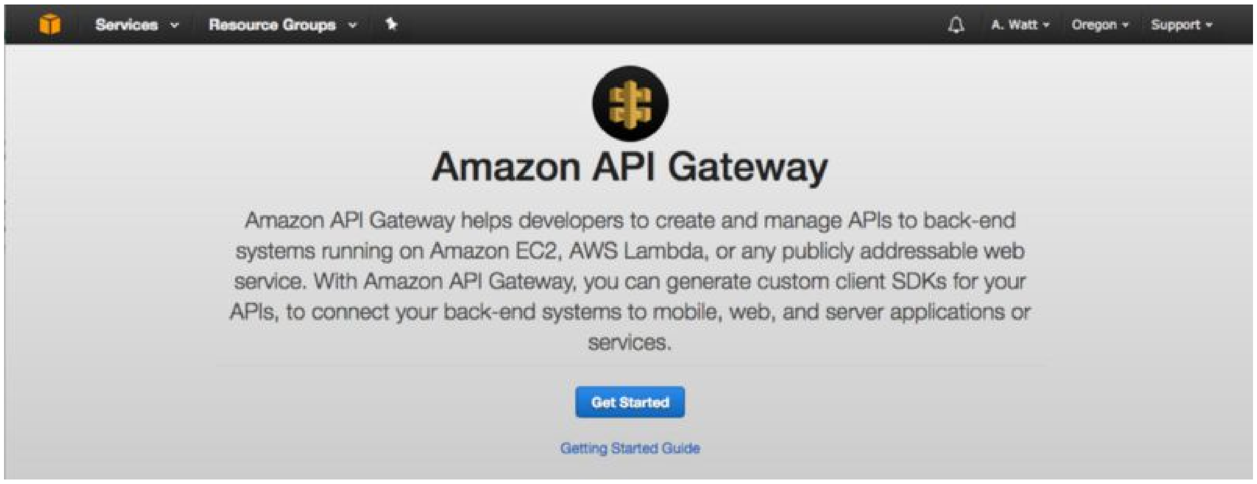
AWS Lambda function call using the AWS API Gateway.

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."

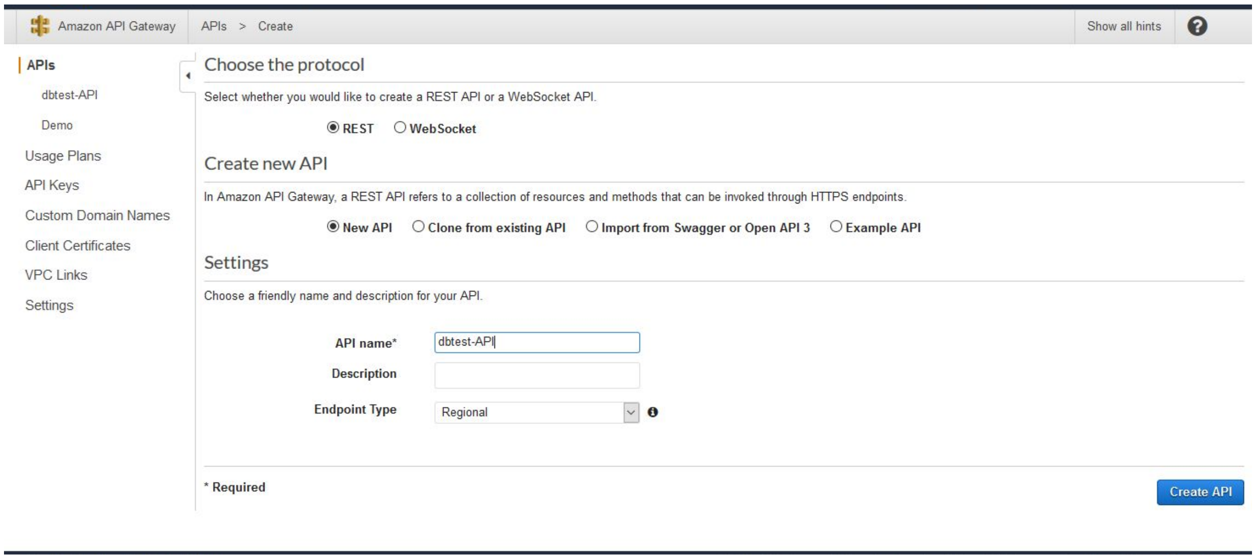




Amazon API Gateway

Amazon API Gateway helps developers to create and manage APIs to back-end systems running on Amazon EC2, AWS Lambda, or any publicly addressable web service. With Amazon API Gateway, you can generate custom client SDKs for your APIs, to connect your back-end systems to mobile, web, and server applications or services.

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


Amazon API Gateway | APIs > Create | Show all hints

APIs
- dbtest-API
- Demo

Usage Plans
API Keys
Custom Domain Names
Client Certificates
VPC Links
Settings

Choose the protocol

Select whether you would like to create a REST API or a WebSocket API.

[✓ REST] WebSocket

Create new API

In Amazon API Gateway, a REST API refers to a collection of resources and methods that can be invoked through HTTPS endpoints.

[✓ New API] Clone from existing API | Import from Swagger or Open API 3 | Example API

Settings

Choose a friendly name and description for your API.

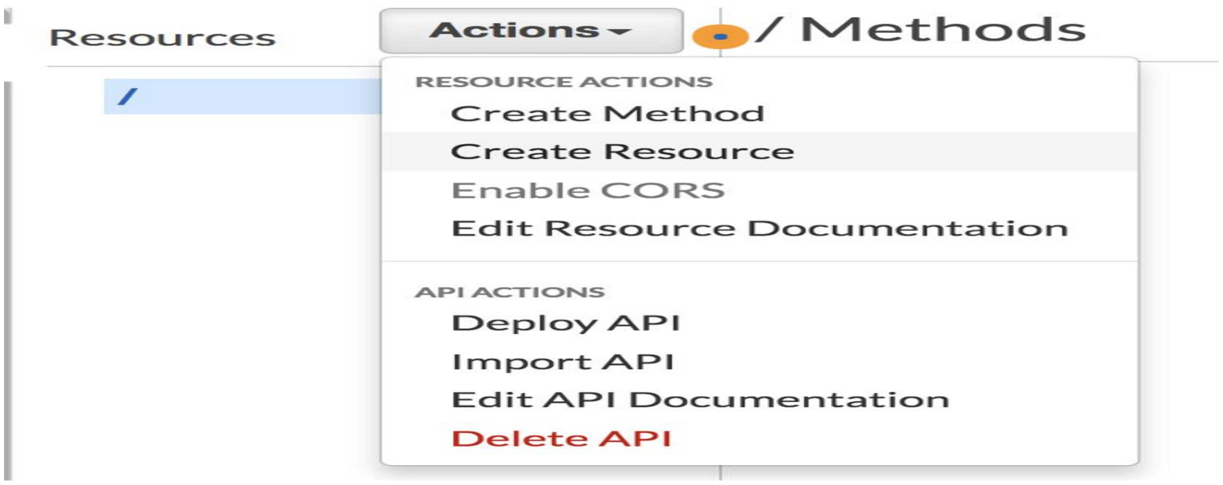
API name*	dbtest-API
Description	
Endpoint Type	Regional

* Required

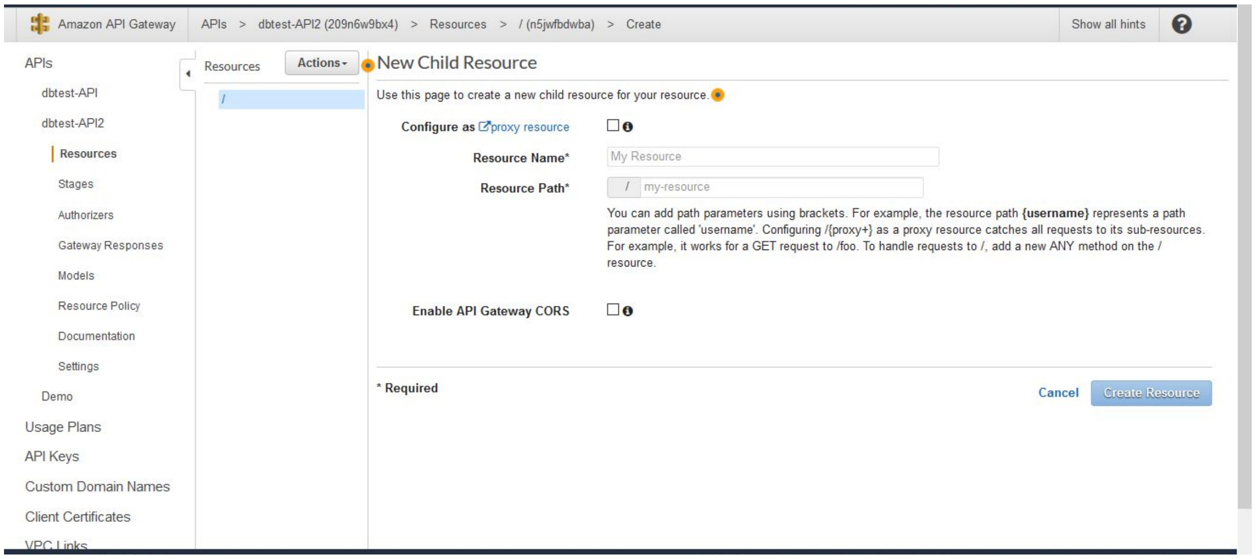
Create API

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



New Child Resource

Use this page to create a new child resource for your resource.

Configure as proxy resource ☐

Resource Name* My Resource

Resource Path* / my-resource

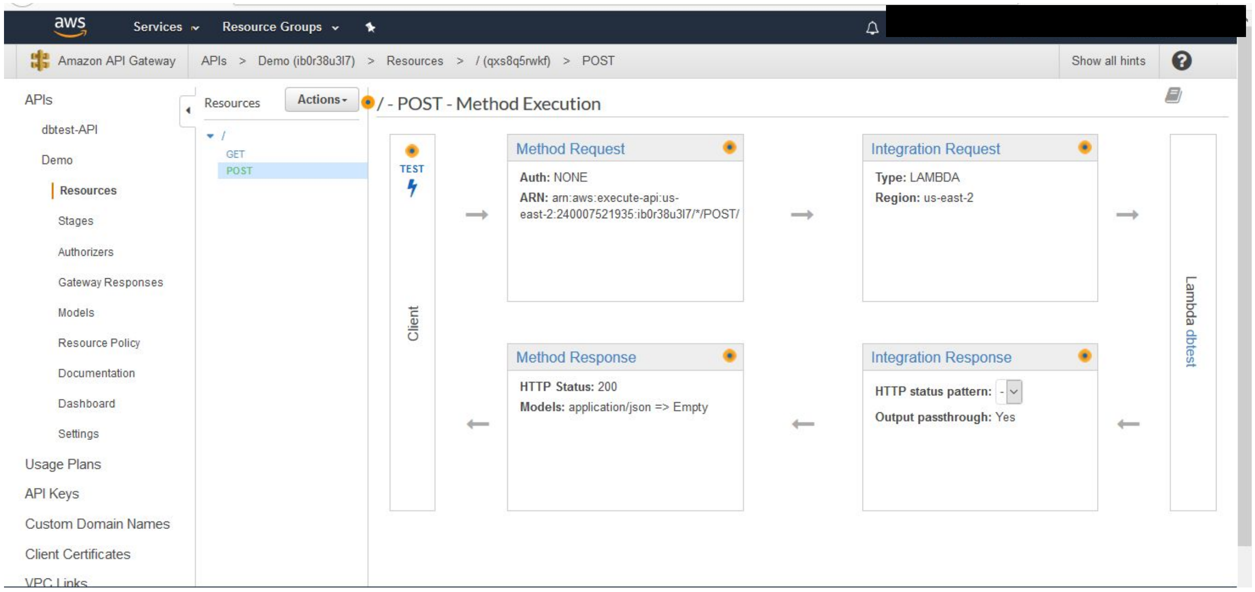
You can add path parameters using brackets. For example, the resource path **{username}** represents a path parameter called 'username'. Configuring /{proxy+} as a proxy resource catches all requests to its sub-resources. For example, it works for a GET request to /foo. To handle requests to /, add a new ANY method on the / resource.

Enable API Gateway CORS ☐

* Required

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”



/ - POST - Method Execution

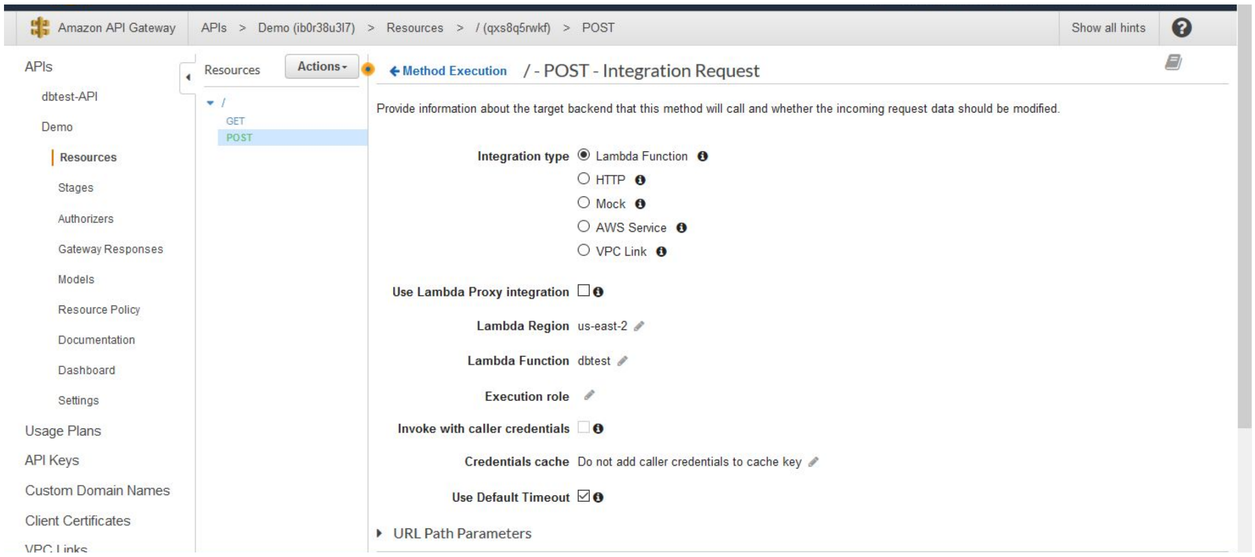
Method Request
Auth: NONE
ARN: arn:aws:execute-api:us-east-2:240007521935:ib0r38u3l7/*/POST/

Integration Request
Type: LAMBDA
Region: us-east-2

Method Response
HTTP Status: 200
Models: application/json => Empty

Integration Response
HTTP status pattern: -
Output passthrough: Yes

Step-12 Save and Test AWS Lambda Function.



/ - POST - Integration Request

Provide information about the target backend that this method will call and whether the incoming request data should be modified.

Integration type: [✓ Lambda Function] HTTP Mock AWS Service VPC Link

Use Lambda Proxy integration ☐

Lambda Region us-east-2

Lambda Function dbtest

Execution role

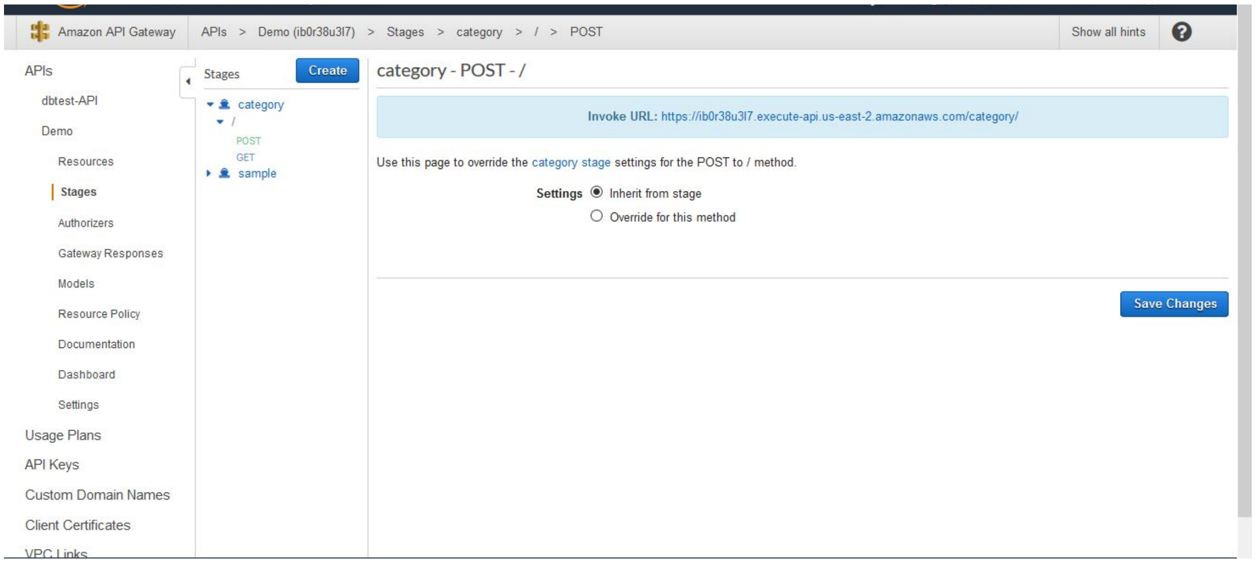
Invoke with caller credentials ☐

Credentials cache Do not add caller credentials to cache key

Use Default Timeout ☑

URL Path Parameters

Do this for the POST methods 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.



category - POST - /

Invoke URL: https://ib0r38u3l7.execute-api.us-east-2.amazonaws.com/category/

Use this page to override the category stage settings for the POST to / method.

Settings [✓ Inherit from stage] Override for this method

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