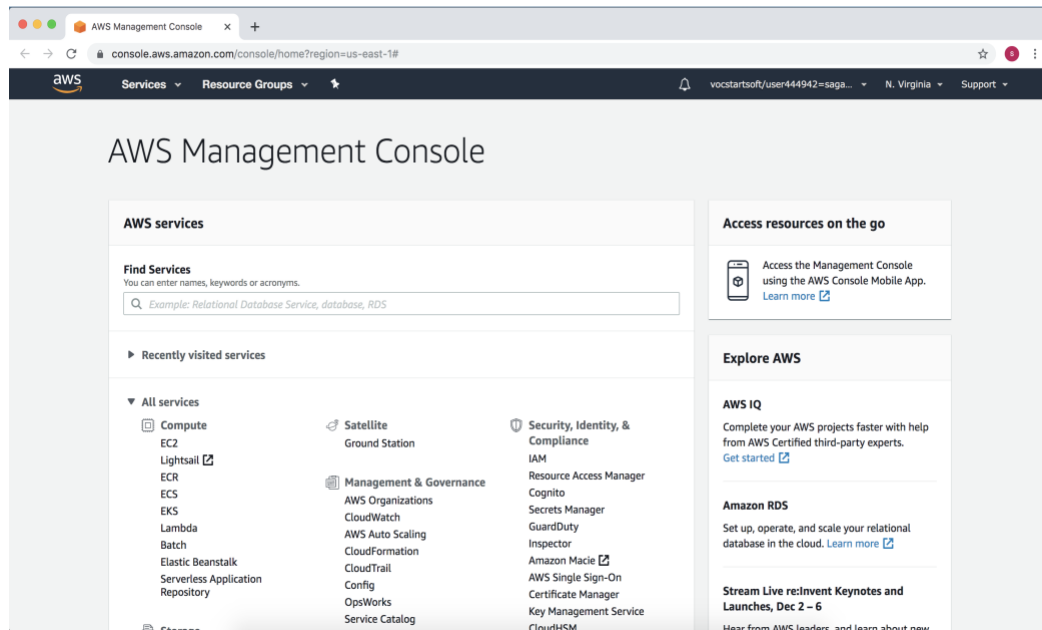


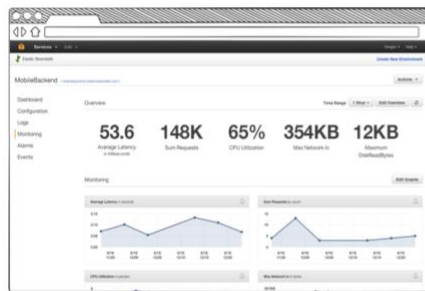
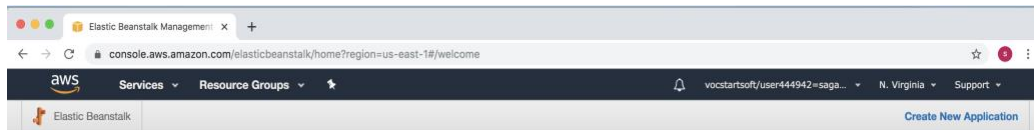
Steps to Deploy the Application on the Cloud:

- a. Create an AWS account if you don't already have one.
- b. Login to your account



- c. We will first create an Elastic beanstalk application. Follow the steps mentioned below to create the application and get it running.

After logging into your AWS console, click on Elastic Beanstalk under all services. You will see the following welcome page:



Welcome to AWS Elastic Beanstalk

With Elastic Beanstalk, you can **deploy**, **monitor**, and **scale** an application quickly and easily. Let us do the heavy lifting so you can focus on your business.

To deploy your **existing web application**, create an **application source bundle** and then create a new application. If you're using **Git** and would prefer to use it with our command line tool, please see [Getting Started with the EB CLI](#).

To deploy a **sample application**, click **Get started**, choose a name, select a platform and click **Create app**.

By launching the sample application, you allow AWS Elastic Beanstalk to administer AWS resources and necessary permissions on your behalf. [Learn more](#)

[Get started](#)

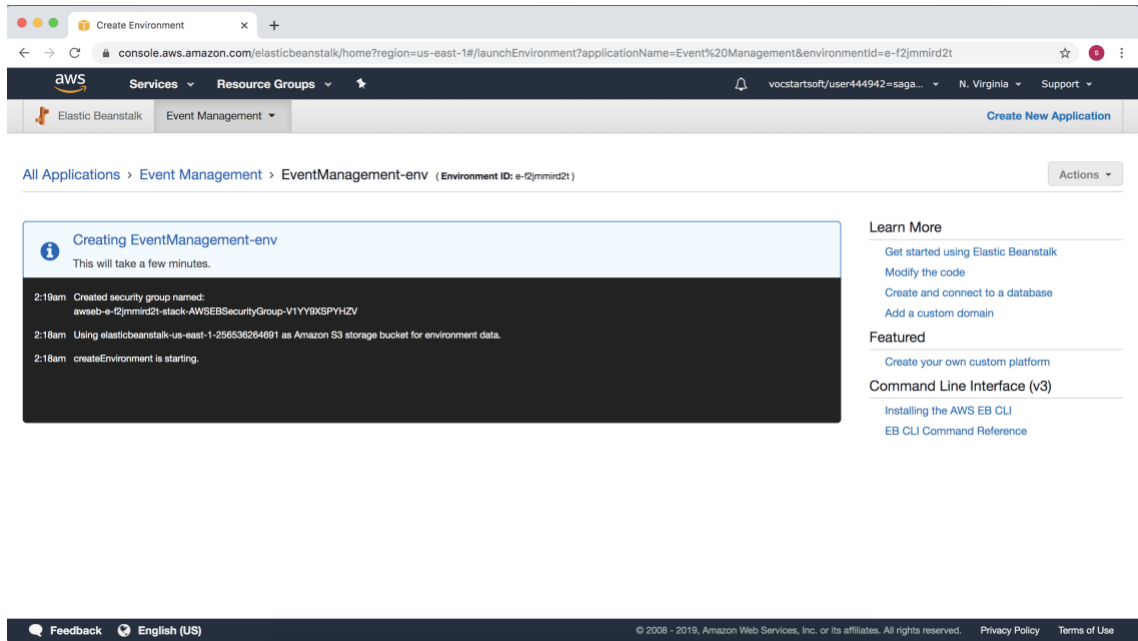
Get Started in Three Easy Steps



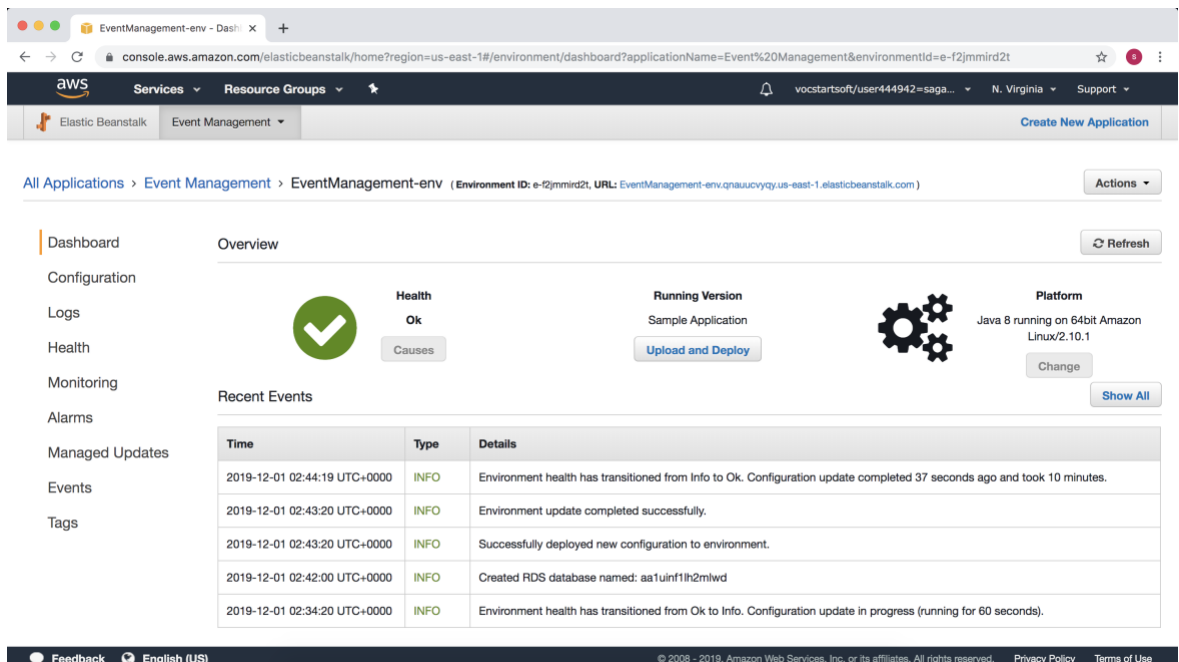
Click on Get Started to create an application on the cloud.

- d. On the next page, enter Application name as 'Event Management'. Choose platform as 'Java'. Leave all other configurations as they are. Click on 'Create application'.

- e. The following screen appears. It creates the environment, creates EC2 instance, Elastic IP, security groups and other configurations required to run the application on the cloud.



Once the environment is created and all the required configurations are set up, you will see the following screen:



- f. Now, we need to configure the application port.
Spring boot runs the application on port 8080 while Elastic Beanstalk on the port 5000. Thus, we need to configure the same so that our application starts listening to port 5000 instead of port 8080.

Go to Configurations tab, search Software and click on Modify option.

The screenshot shows the AWS Elastic Beanstalk console. The breadcrumb navigation is 'All Applications > Event Management > EventManagement-env'. The environment ID is 'e-f2jmmird2t'. The left sidebar contains links to Dashboard, Configuration, Logs, Health, Monitoring, Alarms, Managed Updates, Events, and Tags. The 'Configuration overview' section is active, showing a table of configuration options. The table has three columns: Category, Options, and Actions. There are three rows: Software, Instances, and Capacity. Each row has a 'Modify' button in the Actions column. The 'Software' row shows environment properties like GRADLE_HOME, JAVA_HOME, M2, M2_HOME, XRAY_ENABLED, and log streaming settings. The 'Instances' row shows monitoring interval, IOPS, size, root volume type, and EC2 security groups. The 'Capacity' row shows AMI ID, scaling cooldown, environment type, instance type, and time-based scaling.

Category	Options	Actions
Software	Environment properties: GRADLE_HOME, JAVA_HOME, M2, M2_HOME, XRAY_ENABLED Rotate logs: disabled Log streaming: disabled X-Ray daemon: enabled	<button>Modify</button>
Instances	Monitoring interval: 5 minute IOPS: container default Size: container default Root volume type: container default EC2 security groups: awseb-e-f2jmmird2t-stack-AWSEBSecurityGroup-V1YY9XSPYHZV	<button>Modify</button>
Capacity	AMI ID: ami-0cc4e505d334ac1db Scaling cooldown: 360 seconds Environment type: single instance Instance type: t2.micro Time-based Scaling	<button>Modify</button>

Look for environment properties and add the following value to the list:
SERVER_PORT: 5000

Retention 7 days

Lifecycle Keep logs after terminating environmen

Environment properties

The following properties are passed in the application as environment properties. [Learn more](#)

Name	Value
GRADLE_HOME	/usr/local/gradle ✕
JAVA_HOME	/usr/lib/jvm/java ✕
M2	/usr/local/apache-maven/bin ✕
M2_HOME	/usr/local/apache-maven ✕
XRAY_ENABLED	{ "Fn::GetOptionSetting" : { "Ni ✕
SERVER_PORT	5000 ✕

Cancel Continue Apply

Click on apply button.

- g. Our application will need a database to store the event and user related data. We will configure an RDS instance and create a database instance for the application to use.

Under Configurations tab, search database and click on modify option.

EventManagement-env - Conf

console.aws.amazon.com/elasticbeanstalk/home?region=us-east-1#/environment/configuration?applicationName=Event%20Management&environmentId=e-f2jmmird2t

Capacity	EC2 security groups: awseb-e-f2jmmird2t-stack-AWSEBSecurityGroup-V1YY9XSPYHZV AMI ID: ami-0cc4e505d334ac1db Scaling cooldown: 360 seconds Environment type: single instance Instance type: t2.micro Time-based Scaling:	Modify
Load balancer	This configuration does not contain a load balancer.	
Rolling updates and deployments	Command timeout: 600 Deployment policy: All at once Ignore health check: disabled Healthy threshold: Single instance Rolling updates: disabled	Modify
Security	IAM instance profile: aws-elasticbeanstalk-ec2-role EC2 key pair: -- Service role: arn:aws:iam::256536264691:role/aws-elasticbeanstalk-service-role	Modify
Monitoring	Health event log streaming: disabled CloudWatch Custom Metrics-Environment: CloudWatch Custom Metrics-Instance: Ignore HTTP 4xx: disabled System: Enhanced	Modify
Managed updates	Instance replacement: disabled Managed updates: enabled Update level: Minor and patch Weekly update window: Sun:08:00	Modify
Notifications	Email: --	Modify
Network	This environment is not part of a VPC.	
Database		Modify

Feedback English (US) © 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Now create an RDS instance by specifying the following details:

- a. Engine: postgres
- b. Username: postgres
- c. Password: welcome123

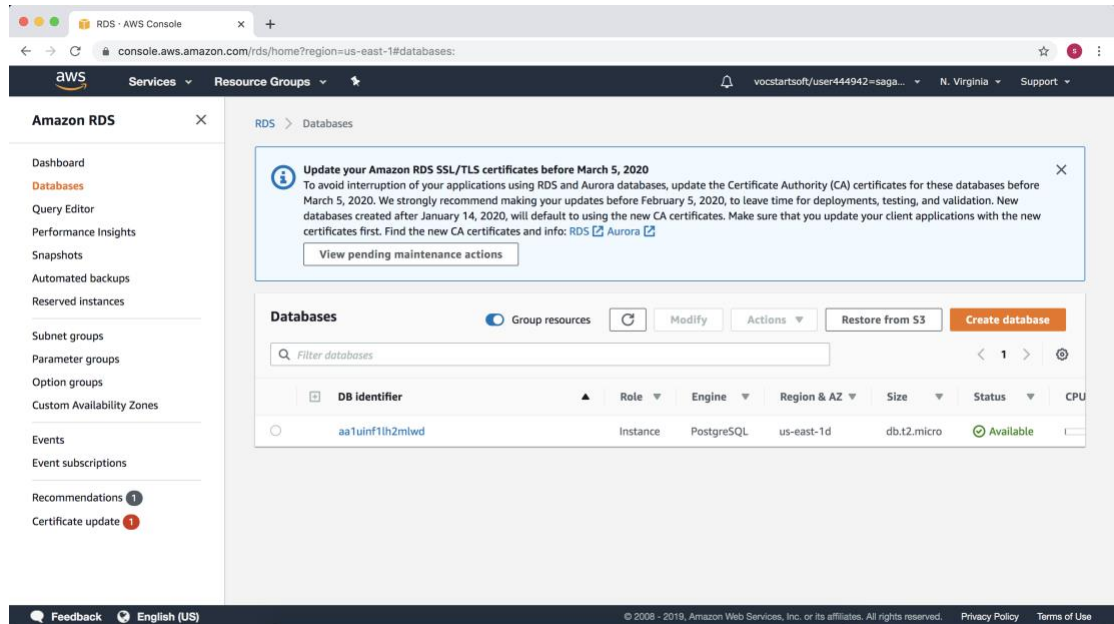
The screenshot shows the AWS Management Console interface for configuring an Elastic Beanstalk environment. The 'Database settings' section is active, displaying various configuration options for the database instance. The settings are as follows:

- Engine:** postgres
- Engine version:** 11.5
- Instance class:** db.t2.micro
- Storage:** 5 GB (with a note: 'Choose a number between 5 GB and 1024 GB.')
- Username:** postgres
- Password:** (masked with asterisks)
- Retention:** Create snapshot (with a note: 'When you terminate your environment, your database instance is also terminated. Choose **Create snapshot** to save a snapshot of the database prior to termination. Snapshots incur standard storage charges.')
- Availability:** Low (one AZ)

At the bottom right of the configuration area, there are three buttons: 'Cancel', 'Continue', and 'Apply'. The 'Apply' button is highlighted in blue, indicating it is the next step to create the RDS instance.

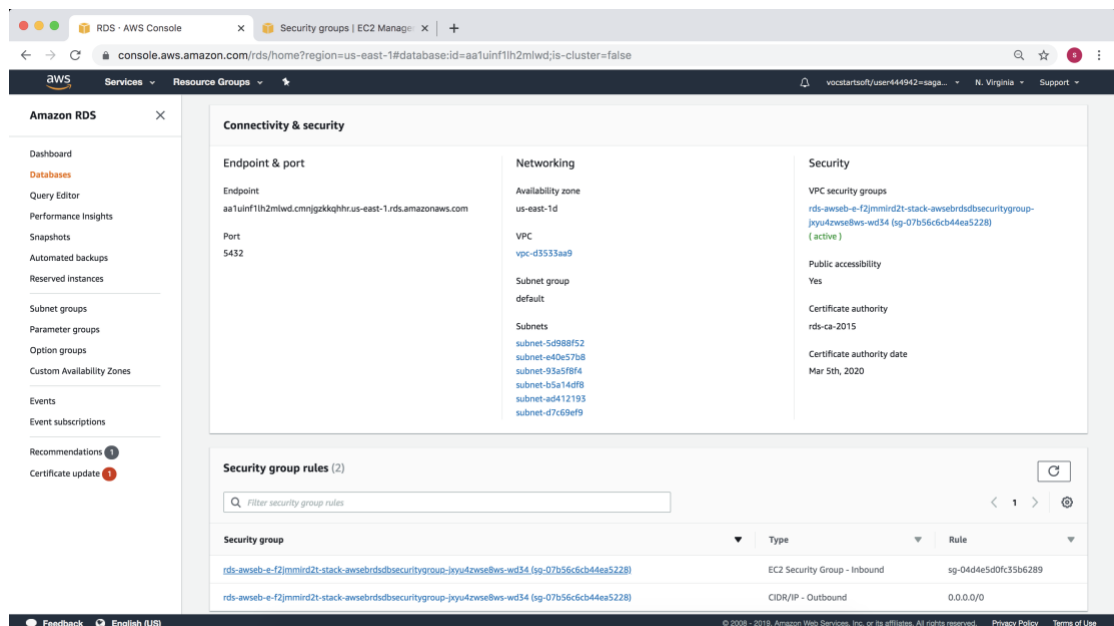
Click on Apply button. An RDS instance gets created.

On the AWS management console, click on RDS under database and then click on DB Instances under databases on the next screen. You should see an instance of RDS created as follows:

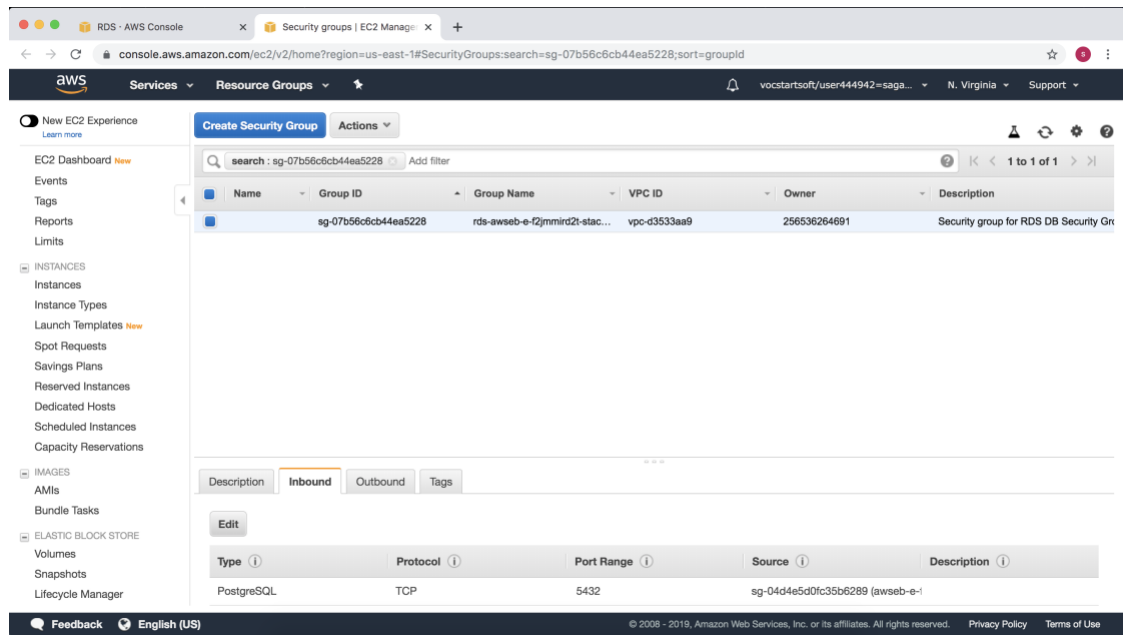


- h. Next we need to allow inbound connections to the database so that we can connect our PostgreSQL client to the database and create tables.

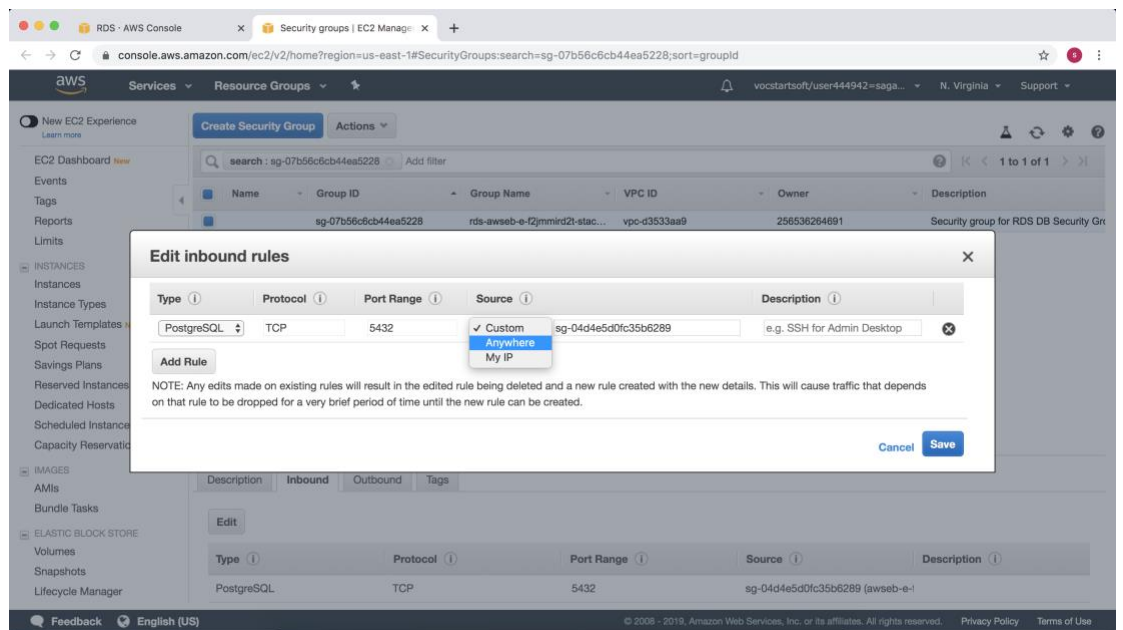
Click on the created instance ("aa1uinf1h2mlwd" in the above image). Under Connectivity and Security tab on the next page, scroll down to Security group rules. Click on the security group that has Type as "EC2 Security Group – Inbound".



Click on the Inbound tab on the next page. Click on the edit button to edit the inbound rules.

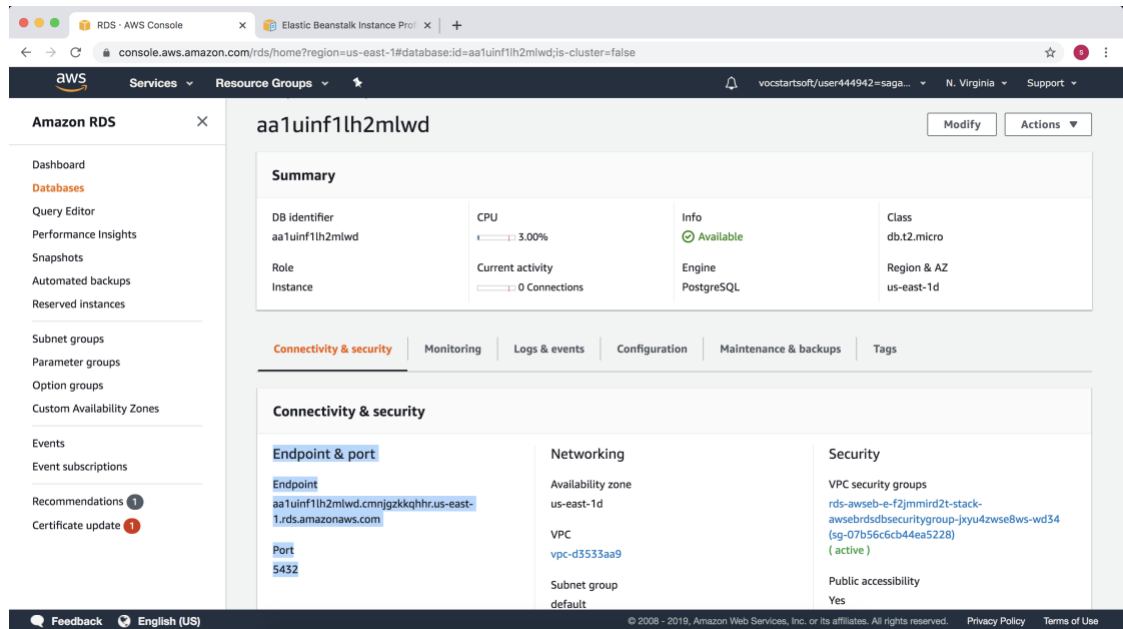


Change the source to 'Anywhere'. Click on save button.

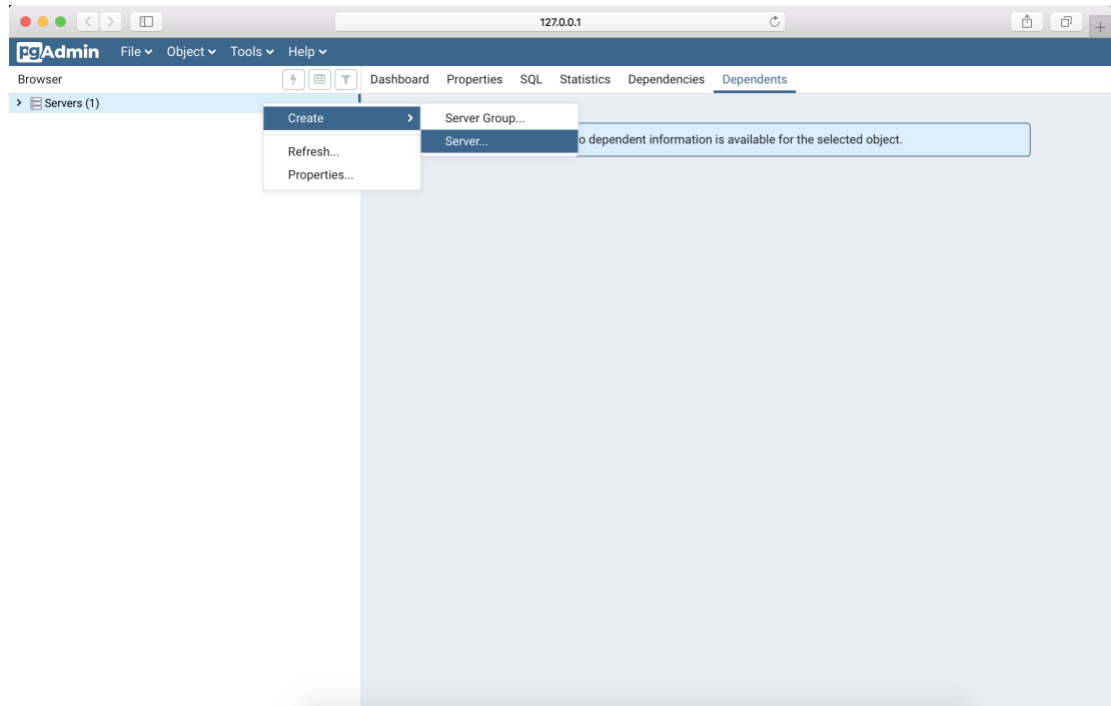


Also ensure Public Accessibility is set to 'Yes' for the database instance.

- i. Go back to the RDS instance screen and **copy** the **endpoint** and **port** available under the Connectivity & Security tab.



Open PostgreSQL on local machine, right click on Servers -> Create -> New Server



Under General tab, give a name to the server. Under Connection tab, use the following credentials to connect to the RDS server.

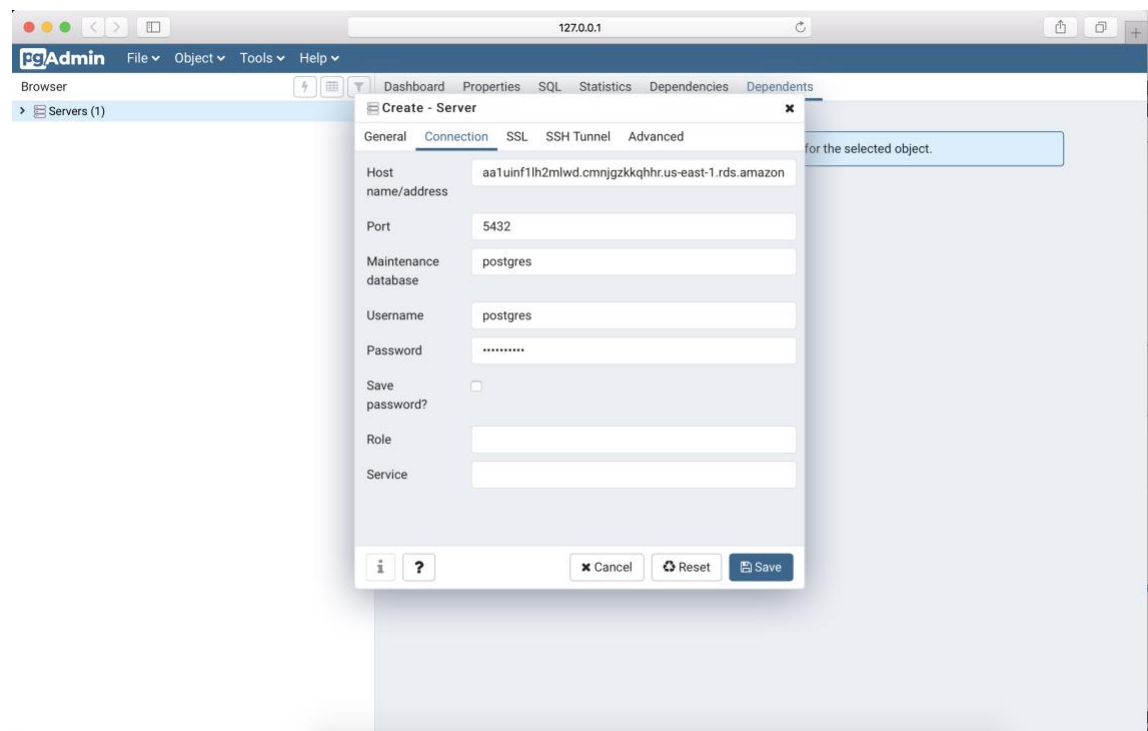
Host name/ address : Endpoint (Copied above)

Port: Port (Copied above)

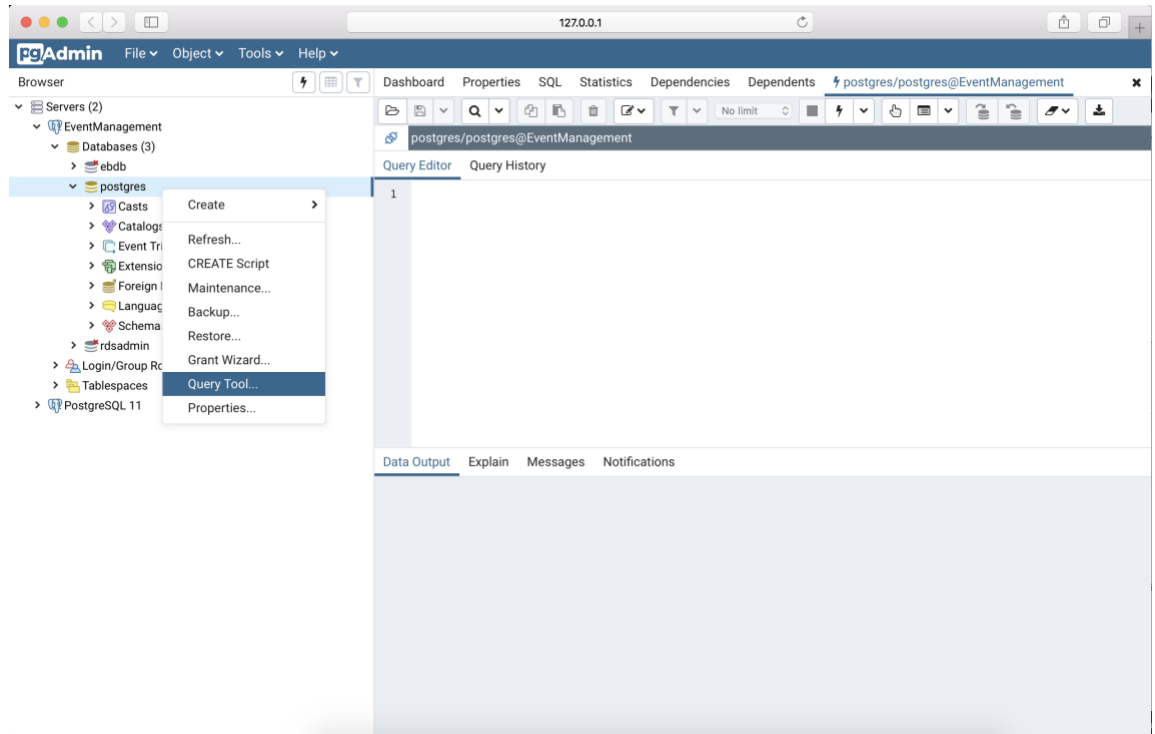
Username: postgres

Password: welcome123

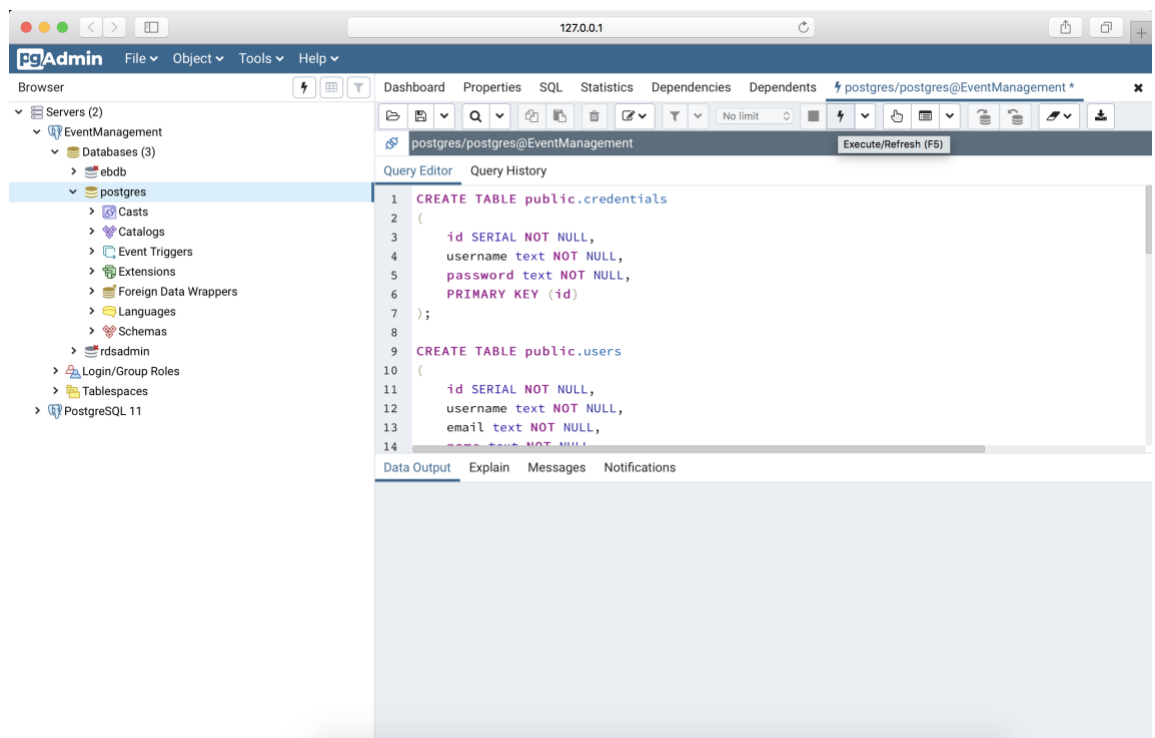
Note: Username and Password were specified while creating the database server in step (g).



Click on save button. Once the connection to the server is successful and database is created on the server, open the Query Tool.



Copy all scripts from the script.sql file present in the project folder and paste it here. Execute them. All the required tables get created.



- j. Now, we will be updating the database connections and the application url in the code.

Go to the following location in the project folder:

`com.service.main/src/main/java/model/connection/pooling/`

Open the `ConnectionPool.java` file and replace the value of endpoint variable with the **endpoint** copied earlier in step (i) and the value of port variable with the **port** copied earlier in step (i).

Go to AWS Management Console -> Elastic Beanstalk -> Click on the application created. Copy the URL shown below.

The screenshot shows the AWS Elastic Beanstalk console. The breadcrumb navigation is 'All Applications > Event Management > EventManagement-env'. The environment ID is 'e-f2jmmird2t' and the URL is 'EventManagement-env.qnauucvqj.us-east-1.elasticbeanstalk.com'. The left sidebar contains links for Dashboard, Configuration, Logs, Health, Monitoring, Alarms, Managed Updates, Events, and Tags. The main content area is titled 'Overview' and includes a 'Refresh' button. It displays three key metrics: Health (Ok), Running Version (Sample Application-1), and Platform (Java 8 running on 64bit Amazon Linux/2.10.1). Below these is a 'Recent Events' table with columns for Time, Type, and Details.

Time	Type	Details
2019-12-01 14:29:40 UTC+0000	INFO	Environment health has transitioned from Info to Ok. Configuration update completed 65 seconds ago and took 84 seconds.
2019-12-01 14:28:00 UTC+0000	INFO	Environment update completed successfully.
2019-12-01 14:28:00 UTC+0000	INFO	Successfully deployed new configuration to environment.
2019-12-01 14:27:40 UTC+0000	INFO	Environment health has transitioned from Ok to Info. Configuration update in progress (running for 10 seconds).
2019-12-01 14:26:37 UTC+0000	INFO	Updating environment EventManagement-env's configuration settings.

Next, go to the following location in the project folder:
com.apex.service.rest/src/main/resources/static/



```
1 var base_url='EventManager-env.qnauucvyqy.us-east-1.elasticbeanstalk.com';
2 base_url='http://'+base_url+'/';
3 function post_(url_path, object){
4   var deferred = $.Deferred();
5   var url = base_url+url_path;
6
7   if(object!=undefined)
8     var json = JSON.stringify(object);
9
10  var xhr = new XMLHttpRequest();
11  xhr.open("POST", url, true);
12  xhr.setRequestHeader('Content-type','application/json; charset=utf-8');
13  xhr.onload = function () {
14    var data = JSON.parse(xhr.responseText);
15    if (xhr.readyState == 4 && (xhr.status == "200")||(xhr.status == "201")) {
16      deferred.resolve(xhr.response);
17    } else{
18      deferred.reject("HTTP error: " + xhr.status);
19    }
20  }
21  if(json!=undefined)
22    xhr.send(json);
23  else
24    xhr.send();
25  return deferred.promise();
26 };
27
```

Open the base-model.js file and replace the base_url with the URL copied earlier.

- k. Next, we will compile and build the project using maven. A deployable jar file will be created which we will deploy on the Elastic Beanstalk.

Open terminal at the project folder and run the following command to execute the shell file: “sh ./build.sh”

The shell file compiles and installs all the maven dependencies. It creates a deployable jar file named “rest-0.1.0.jar” at the following location:

com.apex.service.rest/ target/

Copy the “rest-0.1.0.jar” jar file as we will deploy this jar file on the cloud.

- l. Now we will deploy the jar file on Elastic Beanstalk. Following are the steps to do so:

Go to the AWS Management Console, click on Elastic Beanstalk under Compute. Click on the application we created earlier. On the next page, click on the Upload and Deploy button.

The screenshot shows the AWS Elastic Beanstalk console. The top navigation bar includes the AWS logo, 'Services', 'Resource Groups', and a user profile. The main content area is titled 'Event Management' and shows the 'EventManagement-env' environment. The 'Overview' tab is active, displaying a 'Health' status of 'Ok' with a green checkmark icon. Below this, the 'Recent Events' table lists several events, including health transitions, successful updates, and configuration deployments. The 'Platform' section indicates 'Java 8 running on 64bit Amazon Linux/2.10.1'. A sidebar on the left contains links to various management tools like Dashboard, Configuration, Logs, and Health.

Time	Type	Details
2019-12-01 02:44:19 UTC+0000	INFO	Environment health has transitioned from Info to Ok. Configuration update completed 37 seconds ago and took 10 minutes.
2019-12-01 02:43:20 UTC+0000	INFO	Environment update completed successfully.
2019-12-01 02:43:20 UTC+0000	INFO	Successfully deployed new configuration to environment.
2019-12-01 02:42:00 UTC+0000	INFO	Created RDS database named: aa1uin1fh2mlwd
2019-12-01 02:34:20 UTC+0000	INFO	Environment health has transitioned from Ok to Info. Configuration update in progress (running for 60 seconds).

Choose the location of the file “rest-0.1.0.jar” mentioned in step (k) and give a version label or leave it to default.

This screenshot shows the same AWS Elastic Beanstalk console as the previous one, but with the 'Upload and Deploy' dialog box open in the center. The dialog box has a title bar and a close button. It contains a link to the 'Application Versions' page, a file selection button labeled 'Choose File' with 'rest-0.1.0.jar' selected, and a text input field for the 'Version label' containing 'Sample Application-1'. There are 'Cancel' and 'Deploy' buttons at the bottom right of the dialog. The background console interface is dimmed.

Click on the deploy button. The jar file gets uploaded and the elastic beanstalk updates the environment.

Once the health of the environment becomes Healthy, the application is deployed on the cloud and can be accessed using the URL mentioned on the Elastic Beanstalk application.

The screenshot shows the AWS Elastic Beanstalk console for an application named 'EventManagement-env'. The environment ID is 'e-f2jmmird2t' and the URL is 'EventManagement-env.qnauucvyqy.us-east-1.elasticbeanstalk.com'. The dashboard includes a sidebar with navigation links: Dashboard, Configuration, Logs, Health, Monitoring, Alarms, Managed Updates, Events, and Tags. The main content area shows the 'Overview' tab with a 'Health' status of 'Ok' (indicated by a green checkmark), a 'Running Version' of 'Sample Application-1', and a 'Platform' of 'Java 8 running on 64bit Amazon Linux/2.10.1'. A 'Recent Events' table lists several events, including health transitions and configuration updates.

Time	Type	Details
2019-12-01 14:29:40 UTC+0000	INFO	Environment health has transitioned from Info to Ok. Configuration update completed 65 seconds ago and took 84 seconds.
2019-12-01 14:28:00 UTC+0000	INFO	Environment update completed successfully.
2019-12-01 14:28:00 UTC+0000	INFO	Successfully deployed new configuration to environment.
2019-12-01 14:27:40 UTC+0000	INFO	Environment health has transitioned from Ok to Info. Configuration update in progress (running for 10 seconds).
2019-12-01 14:26:37 UTC+0000	INFO	Updating environment EventManagement-env's configuration settings.

Thus, the application is deployed:

