**Configure and Connect a MySQL Database Instance with a Web Server**

Project 1

**DESCRIPTION**

You are working as a database administrator for an IT firm. You have been asked to create a new database instance on AWS cloud and connect it with the employee management portal hosted on a web server.

**Background of the problem statement:**

Your organization wants to deploy a new multi-tier application. The application will take live inputs from the employees and it will be hosted on a web server running on the AWS cloud.

The development team has asked you to set up the web server and configure it to scale automatically in cases of a traffic surge, to make the application highly available. They have also asked you to take the inputs from the employees and store them securely in the database.

**You must use the following:**

* Create a Database Instance with the following specifications:
  + Database creation method: Standard Create
  + Engine: MySQL
  + Database Instance size: db.t2.micro

* Create an EC2 Instance with the following specifications:
  + AMI: Amazon Linux
  + Region: Use only US East (N Virginia), us-east-1, and us-east-2
  + Instance types: t2.micro and t3.micro
  + Allowed EBS types: GP2 and Standard

**Following requirements should be met:**

* Follow the above-mentioned specifications
* Make sure that the Availability Zone is similar throughout the instances and volumes
* Ensure that the server scales automatically and the traffic is optimally routed among the scaled servers
* Document the step-by-step process involved in completing this task

**Proposed Solution – Architecture Diagram**

**Note:** Security Group and Route tables are available in screenshots of project documents

Graphical user interface, diagram

Description automatically generated with medium confidence

**Project Code (Backend)**

New Web application project ‘**MyProject’** is created using following technologies.

Java, HTML, Bootstrap, jQuery, Maven, Jersey RESTful Web Service, MySQL.

The application exposes following REST API which helps to insert data for new Employee.

[**http://localhost:8080**/myproject/rest/employee/add](http://localhost:8080/myproject/rest/employee/add)

This application will be deployed on the EC2 instance where the hostname (highlighted in URL) will be replaced by the URL given by load balancer.

***Note :*** Please find the project code uploaded

Following are the important files in the web application.

**pom.xml**

This file has dependency added for mysql version 8.0.23 jar file.

<dependency>

<groupId>mysql</groupId>

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

<version> EmployeeDAO </version>

</dependency>

**EmployeeDAO.java**

This file has method to create a connection to RDS (MySQL DB) created on AWS and insert the data into Database send from UI.

**public** **class** EmployeeDAO {

**public** String addInEmployee(String firstName, String lastName, String email) **throws** SQLException, ClassNotFoundException {

**final** String dbClassName = "com.mysql.jdbc.Driver";

// Endpoint URL from the RDS (MySQL DB)

**final** String CONNECTION = "jdbc:mysql://myprojectdbinstance1.c7n28mz1nzf6.us-east-1.rds.amazonaws.com";

//Load the class with the specified className.

Class.*forName*(dbClassName);

Properties p = **new** Properties();

p.put("user","admin");

p.put("password","adminpassword");

Connection connect = DriverManager.*getConnection*(CONNECTION,p);

PreparedStatement ps =connect.prepareStatement("INSERT INTO Employee VALUES (?,?,?)");

ps.setString(1, firstName);

ps.setString(2, lastName);

ps.setString(3, email);

ps.executeUpdate();

connect.close();

**return** "success";

}

}

**EmployeeRestAPI.java**

Following file is used to expose the REST API.

@Path("/employee")

**public** **class** EmployeeRestAPI {

@POST

@Path("/add")

@Consumes(MediaType.***APPLICATION\_JSON***)

@Produces(MediaType.***APPLICATION\_JSON***)

**public** String addEmployee(String empData) **throws** ClassNotFoundException, SQLException, ParseException {

JSONParser jp = **new** JSONParser();

JSONObject jo1 = (JSONObject) jp.parse(empData);

AddEntry add = **new** AddEntry();

String msg = add.addInEmployee((String)jo1.get("firstname"),(String)jo1.get("lastname"),(String)jo1.get("email"));

System.***out***.println(msg);

**return** empData;

}

}

**DB queries:**

Database name : EmployeeManagement

Table Name : Employee

use EmployeeManagement;

**drop** **table** Employee;

**CREATE** **TABLE** IF **NOT** **EXISTS** Employee (

emp\_id **INT** AUTO\_INCREMENT,

first\_name **VARCHAR**(255) **NOT** **NULL**,

last\_name **VARCHAR**(255) **NOT** **NULL**,

email **VARCHAR**(255) **NOT** **NULL**,

**PRIMARY** **KEY** (emp\_id)

);

**INSERT** **INTO** Employee (first\_name,last\_name,email) **VALUES** ('Sachin','Tendulkar','st@abc.com');

**INSERT** **INTO** Employee (first\_name,last\_name,email) **VALUES** ('Rahul','Dravid','rd@abc.com');

**select** \* **from** Employee;

**Project Code (Frontend)**

Following is index.html file which will be available on frontend

User will enter the Employee details (first\_name, last\_name and email) in the form. After clicking ‘Insert into Database’ button the record will be inserted in the database.

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

</head>

<link rel=*"stylesheet"* href=*"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"*>

<!-- Scripts here -->

<body>

<div id=*"includedContent"*></div>

<div style="padding-left: *70px*;" class=*"container-fluid"*>

<br>

<div class=*"row"*>

<div>

<h2 style="padding-left: *100px*;">Employee Management System </h2>

<h3 style="padding-left: *250px*;color:*green*;">

Add Employee Record

</h3>

</div>

</div>

<div style="padding-left: *150px*;" class=*"row"*>

<span class=*"col-md-offset-2 col-md-2"*>

<b> First Name :</b> <input id=*"firstname"* type=*"text"*>

</span>

<span class=*"col-md-2"*>

<b> Last Name : </b><input id=*"lastname"* type=*"text"*>

</span>

<span class=*"col-md-2"*>

<b> Email : </b><input id=*"email"* type=*"text"*>

</span>

</div>

<br>

<div style="padding-left: *150px*;" class=*"row"*>

<div class=*"col-md-offset-3 col-md-4"*>

<button id=*"btn"* type=*"button"* class=*"btn btn-default"*> Add Employee </button>

</div>

</div>

<br>

<div style="padding-left: *120px*;" class=*"row"*>

<div class=*"col-md-offset-3 col-md-4"*>

<h4><span id=*"1"* hidden=*"hidden"*>ID </span>

<span style="color:*green*" id=*"2"*></span>

<span id=*"3"* hidden=*"hidden"*>is successfully added.</span></h4>

</div>

</div>

</div>

</body>

</html>

**Following are the scripts (functions written) in html file.**

The Ajax function call at the top is to collect employee data (first\_name, last\_name and email) from the form and call REST API to add employee data in could DB.

Second function is used to display IP address of host which is available in hostname\_file.html.

This file gets created from user data section while creating EC2 instance

<script src=*"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"*></script>

<script src=*"https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"*></script>

<script type=*"text/javascript"*>

$(document).ready(**function**(){

$('#btn').click(**function**(){

**var** jsonfile = JSON.stringify({

"firstname" : $("#firstname").val(),

"lastname" : $("#lastname").val(),

"email" : $("#email").val()

});

console.log(jsonfile);

**var** ans = $.ajax({

type : 'POST',

url : "http://localhost:8080/myproject/rest/employee/add",

dataType : 'json',

contentType : 'application/json',

data : jsonfile

});

ans.done(**function**(data){

$("#1").show();

$("#2").show();

$("#3").show();

});

ans.fail(**function**() {

alert( "Request failed: " );

$("#1").show();

$("#2").show();

$("#3").show();

});

})

});

</script>

<script>

$(**function**(){

$("#includedContent").load("hostname\_file.html");

});

</script>

**Project Execution Steps in AWS**

Make sure first you install the Backend application. All the below steps with screenshots are available in the ppt file uploaded in the project

1. **Network setup (VPC, Subnet, Internet Gateway, Route tables)**

In this step, new VPC is created for MyProject and two public subnets are created in AZ us-east-1a, us-east-1b. New internet Gateway is created and attached to VPC. In the route table created for VPC, inbound traffic from internet is linked to internet Gateway.

1. **RDS – MySQL DB (Subnet group, DB setup and configure)**

In this, first create subnet group and attach VPC and its subnets. Then create MySQL RDS with standard, free-tier instance setup with auto-scaling, backup and minor version upgrade enabled.

This DB should be created to VPC and the new subnet group created for the VPC.

.

1. **EC2 instance creation (Configure MySQL in Security Group)**

Create EC 2 instance (in free-tier) with the user data to configure Apache HTTP server and MySQL client. Add Entry for MySQL/Aurora at port 3306 in security group.

1. **Configure RDS Security Group to EC2 Security Group**

In RDS instance, add inbound rule for type MySQL/Aurora (port 3306) pointing to source as security group created for EC2 instances

1. **Login to EC2 and test DB from MYSQL client, create tables**

From EC2 connect, login to DB using MySQL client, create table and add some records to verify the connectivity from EC2

1. **Create new AMI and test creating new instances with the new AMI**

From the created EC2 instance (which is already tested with MySQL client), create new AMI. Later create and test new EC2 instance with new AMI

1. **Create Load Balancer for EC2 (Load Balancer, Target Group)**

Create load balancer with Target group for monitoring all the instances with health check. Also test the new domain link created by load balancer

1. **Create Scaling for EC2 (Auto Scaling Group, Launch Configuration)**

Create a new launch template for EC2 instance from the already created AMI. In Auto Scaling Group configure launch template, VPC, subnets. Select health check from already created Load Balancer. Also, mention min, max and designed CPU instances based on auto scaling policy