**# phpapp**

PHP based web app to perform CRUD operations on MYSQL.

This web-based app will perform the CREATE, READ, UPDATE & DELETE operations on MySQL installed on the same machine. We can configure it to perform the CRUD operations on MySQL installed on any other machine.

Deploy 2-Tier PHP based Web App in AWS

In this project, we will be deploying a PHP based web application (employee-php-app) in a 2-tier architecture provisioned in the AWS. The source code for the web application is available on GitHub. First we have to provision the 2-tier architecture in AWS. The 2-tier architecture includes a web-tier for deploying the web application. It is done in an Ubuntu based LAMP stack installed in the EC2 instance. This should be exposed to the internet. Then we have to provision the data-tier in which we will launch a MySQL based Amazon RDS instance which should not get exposed to the internet. It will get updated through a NAT gateway. For both the tiers, we will create an isolated network with 2 subnets, 1 public for web-tier and 1 private for data-tier. NAT gateway is created in public subnet to let RDS instance get minor updates as required.

Follow the steps to deploy PHP based Web application in a 2-tier architecture provisioned in AWS:

Isolated Network Setup:

1. Create a VPC with name "myVPC" & CIDR notation - 10.0.0.0/16
2. Create 2 Subnets with given details: a. Name: Public-1a, AZ: us-east-1a, CIDR: 10.0.1.0/24 b. Name: Private-1a, AZ: us-east-1a, CIDR: 10.0.2.0/24
3. Edit the public subnet to enable "Auto-assign Public IPv4 address".
4. Create Internet Gateway with name "myIGW" and attach it to "myVPC".
5. Create a NAT Gateway in Public-1a subnet with name: "myNGW".
6. Create 2 Route Tables with given details: a. Name: Public-RT, Routes: (add dest-0.0.0.0/0 & target-"myIGW"), Subnets: Public-1a b. Name: Private-RT, Routes: (add dest-0.0.0.0/0 & target-"myNGW"), Subnets: Private-1a
7. Create 2 Security Groups with given details in "myVPC": a. Name: employee-DB-SG, Inbound rules: [MySQL/Aurora@3306@employee-Web-SG] b. Name: employee-Web-SG, Inbound rules: [SSH@22@My IP, HTTP@80@Anywhere, MySQL/Aurora@3306@employee-DB-SG]

MySQL Based Amazon RDS Instance setup for Data Tier (Private Subnet)

Launch MySQL based RDS Instance in the Private-1a subnet with given details:

1. DB Engine: employee-DBServer
2. Templates: Free Tier
3. Availability and durability: Deployment options: Single DB instance (default selection) (uneditable due to Free Tier template)
4. DB instance identifier: myDBServer
5. Under Credetials Settings: Master username: root
6. Master Password: give a password
7. Confirm master password: same as Master Password
8. Instance Configuration: DB instance class: Burstable Classes (includes t classes): db.t3.micro
9. Storage: Storage type: gp2, Allocated storage: 20GB, Storage autoscaling: Enable storage autoscaling (checked), Maximum storage threshold: 1000GB
10. Connectivity:Compute resource: Don't connect to an EC2 compute resource
11. Virtual private cloud (VPC): myVPC, DB subnet group: Public-1a & Private-1a
12. Public access: No, VPC security group (firewall): Choose Existing: employee-DB-SG
13. Availability Zone: Private-1a
14. Database authentication: Password authentication
15. Monitoring: Enable Enhanced monitoring: check to enable & choose all default options.
16. Additional configuration: For Database options: Initial database name: demo & choose all default options.
17. For Backup: Enable automated backups: choose all default options.
18. For Encryption: choose all default options.
19. For Maintenance: choose all default options.
20. For Deletion protection: choose all default options.

Ubuntu based EC2 instance set up for Web Tier

Launch an Ubuntu based EC2 instance with Apache Web Server, PHP and MySQL client as Web Tier:

1. Name: employee-Web
2. AMI: Ubuntu 22.04 LTS
3. Instance type: t2.micro
4. Key Pair: Create new and give any name.
5. Network Settings: Click Edit button
6. VPC: myVPC, Subnet: Public-1a
7. Firewall: Select existing security group: employee-Web-SG
8. Storage choose defaults
9. User data: Type below script:

"#!/bin/bash

sudo su –

apt-get update

apt-get install apache2 mysql-client -y

apt-get install php libapache2-mod-php php-mysql -y

service apache2 start"

Connect to EC2 instance and deploy the application:

1. Connect to the instance using Git Bash/Putty/any SSH based tool
2. Switch to root user:

$ sudo su –

1. Move the PHP index file to the begining in the Apache's dir.conf file

vi /etc/apache2/mods-enabled/dir.conf

1. Modify the content by moving index.php before index.html like given below:

<IfModule mod\_dir.c>

DirectoryIndex index.php index.html index.cgi index.pl index.xhtml index.htm

</IfModule>

1. Restart Apache Service

service apache2 restart

1. Clone this git repository

git clone <https://github.com/devopstraining611-282/phpapp.git>

1. Go to the phpapp directory

cd employees\_php\_app

1. Copy the employee-DBServer endpoint and paste in below command and run it connect to the instance, create a table "employees" and dump sample data in it.

mysql -h -u root -p < employees.sql

1. Remove the default index.html page from apache web servers default deployment path

rm /var/www/html/index.html

1. Edit the config.php file to add the RDS instance endpoint and master password.
2. Open the config.php file in vi editor and replace the RDS\_INSTANCE\_ENDPOINT with the endpoint of the above launched MySQL based RDS instance.
3. Also replace the RDS\_MASTER\_PASSWORD with the master password specified while launching the above MySQL based RDS instance.
4. Now copy all the .php file into the apache web servers default deployment path

cp \*.php /var/www/html

1. Copy the public ipv4 ip of the instance and paste in the browser to open the application. This will access the application using HTTP protocol.

You have now successfully deployed PHP based Web App for CRUD operation on MySQL in AWS 2-tier architecture