

Bitcot - DevOps - Jr Machine Task

- Mahesh Gore



Github Repo - <https://github.com/mahigore/php-rds.git>

1. Create a PHP Web Application

- Blow is php application.

```
<!DOCTYPE html>
<html>
<head>
  <title>Feedback Form</title>
</head>
<body>
  <h2>Submit Feedback</h2>
  <form method="POST" action="">
    Name: <input type="text" name="name"><br><br>
    Feedback: <textarea name="feedback"></textarea><br><br>
    <input type="submit" value="Submit">
  </form>

  <?php
  $servername = getenv('DB_HOST');
  $username = getenv('DB_USER');
  $password = getenv('DB_PASS');
  $dbname = getenv('DB_NAME');

  $conn = new mysqli($servername, $username, $password, $dbname);

  if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
  }
```

```

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $name = $_POST["name"];
    $feedback = $_POST["feedback"];

    $sql = "INSERT INTO feedback (name, feedback) VALUES ('$name', '$feedback')";
    if ($conn->query($sql) === TRUE) {
        echo "Feedback submitted successfully!";
    } else {
        echo "Error: " . $conn->error;
    }
}

echo "<h3>All Feedback:</h3>";
$result = $conn->query("SELECT * FROM feedback");
while($row = $result->fetch_assoc()) {
    echo "<b>" . $row["name"] . ":</b> " . $row["feedback"] . "<br>";
}

$conn->close();
?>
</body>
</html>

```

2. Dockerize the PHP Application

- Creation of a Dockerfile for the application. This Dockerfile should use an official PHP Docker image and should copy your application files into the appropriate directory within the Docker image.

```

# Use official PHP image with Apache web server
FROM php:8.2-apache

# Install mysqli extension
RUN docker-php-ext-install mysqli

```

```
# Copy your PHP app files into the web server directory
COPY . /var/www/html/

# Expose port 80 to access the app in the browser
EXPOSE 80

# Start Apache server when the container runs
CMD ["apache2-foreground"]
```

3. Set Up MySQL Database on AWS RDS

3.1 Create a MySQL database instance on AWS RDS. Connect your PHP application to this database and ensure it works as expected.

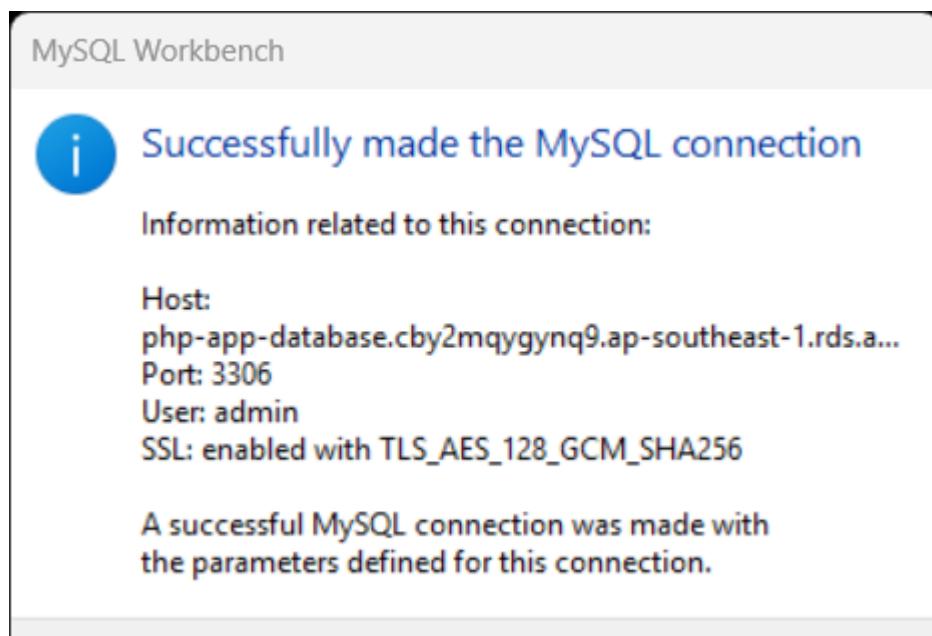
Configuration:

- Creation method: Standard create
- Engine: MySQL
- Engine version: 8.0.43
- Templates: Free tier
- Availability: Single-AZ DB instance
- DB instance identifier: `php-app-database`
- Master username: `admin`
- Credentials management: Self managed
- Master password: `[Your Password]` - Mahi7888
- Instance class: `db.t2g.micro`
- Storage: 5 GB
- VPC: default
- DB subnet group: default
- Public access: Yes
- VPC security group: default { port 3306 }
- Availability Zone: 1a
- Enhanced monitoring: Enable audit logs, error log, slow query log

The screenshot shows the AWS RDS Databases page. At the top, there's a navigation bar with 'Aurora and RDS' and 'Databases'. Below it is a search bar labeled 'Filter by databases'. The main area displays a table titled 'Databases (1)'. The table has columns: DB identifier, Status, Role, Engine, Upgrade rollout or..., Region & AZ, Size, Recom..., and CPU. A single row is shown for 'php-app-database', which is 'Available', an 'Instance', using 'MySQL Co...', 'SECOND' engine, located in 'ap-southeast-1a', with a size of 'db.t4g.micro'. At the bottom of the page, there are links for 'CloudShell', 'Feedback', 'Console Mobile App', and copyright information: '© 2025, Amazon Web Services, Inc. or its affiliates.' followed by 'Privacy', 'Terms', and 'Cookie preferences'.

3.2. Connect to MySQL in Workbench

1. Open MySQL Workbench
2. Click "+" to create a new connection
3. Enter connection details:
 - **Connection Name:** PHP Feedback App
 - **Hostname:** RDS-endpoint
 - **Port:** 3306
 - **Username:** admin
 - **Password:** Click "Store in Vault" and enter your password
4. Click "Test Connection" to verify
5. Click "OK" to save



3.3. Create Database and Table

Once connected, open a new SQL tab and run this script:

```
# Create the database
CREATE DATABASE IF NOT EXISTS `php-app-database`;
# Use the database
USE `php-app-database`;
# Create the feedback table
CREATE TABLE IF NOT EXISTS `feedback` ( `id` INT AUTO_INCREMENT PRIMARY KEY, `name` VARCHAR(255) NOT NULL, `feedback` TEXT NOT NULL, `created_at` TIMESTAMP DEFAULT CURRENT_TIMESTAMP )
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
# Verify table was created
SHOW TABLES;
# View table structure
DESCRIBE feedback;
# Optional: Insert some test data
INSERT INTO `feedback` (`name`, `feedback`) VALUES
('John Doe', 'This is a test feedback!'),
('Jane Smith', 'Great application, very easy to use.');
# View all records
SELECT * FROM feedback;
```

The screenshot shows the MySQL Workbench interface. In the top-left, the Navigator pane displays the schema 'php-app-database' with its tables ('feedback'), views, stored procedures, and functions. The main area contains a query editor with the following SQL code:

```

9 • SHOW TABLES;
10 # View table structure
11 • DESCRIBE feedback;
12 # Optional: Insert some test data
13 • INSERT INTO `feedback` (`name`, `feedback`) VALUES
14 ('John Doe', 'This is a test feedback!'),
15 ('Jane Smith', 'Great application, very easy to use.');
16 # View all records
17 • SELECT * FROM feedback;

```

The Result Grid shows two rows of data from the 'feedback' table:

ID	Name	Feedback	Created At
1	John Doe	This is a test feedback!	2025-12-08 04:23:40
2	Jane Smith	Great application, very easy to use.	2025-12-08 04:23:40

The bottom pane, titled 'Output', displays the execution log:

Action	Time	Message	Duration / Fetch
6 09:53:40 USE `php-app-database`		0 row(s) affected	0.047 sec
7 09:53:40 CREATE TABLE IF NOT EXISTS `feedback` (`id` INT AUTO_INCREMENT PRIMARY KEY, `name` VARCHAR... 0 row(s) affected		0.059 sec	
8 09:53:40 SHOW TABLES		1 row(s) returned	0.047 sec / 0.000 sec
9 09:53:40 DESCRIBE feedback		4 row(s) returned	0.062 sec / 0.000 sec
10 09:53:40 INSERT INTO `feedback`(`name`, `feedback`) VALUES ('John Doe', 'This is a test feedback!'), ('Jane Smith', 'Great application, very easy to use.')		2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0	0.047 sec

4. Manual Deployment to EC2

4.1 Create an AWS EC2 instance

- Basic Configuration:
 - Name: `php-app`
 - AMI: Amazon Linux 2
 - Instance type: `t2.micro`
 - Key pair: [your key pair]
 - VPC: default
 - Public IP: Enable
 - Security groups: default [ports → 22, 80]
 - VPC: default
- Security group

The screenshot shows the AWS EC2 Security Groups console. A security group named 'sg-0883de8f7cbad2f81 - default' is selected. The 'Inbound rules' tab is active, showing four entries:

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source
-	sgr-0a72a35b173013291	IPv4	SSH	TCP	22	0.0.0.0/0
-	sgr-0517c7784190f4d0b	IPv4	HTTP	TCP	80	0.0.0.0/0
-	sgr-015dd006bef018f46	IPv4	MySQL/Aurora	TCP	3306	0.0.0.0/0
-	sgr-0668a514156458c02	IPv4	Custom TCP	TCP	8080	0.0.0.0/0

- Install Git, Docker

```
yum install git
yum install docker
systemctl enable docker
systemctl start docker
systemctl status docker
sudo usermod -aG docker ec2-user
sudo systemctl restart docker
```

4.2 Manually transfer your PHP application files and Dockerfile to the EC2 instance.

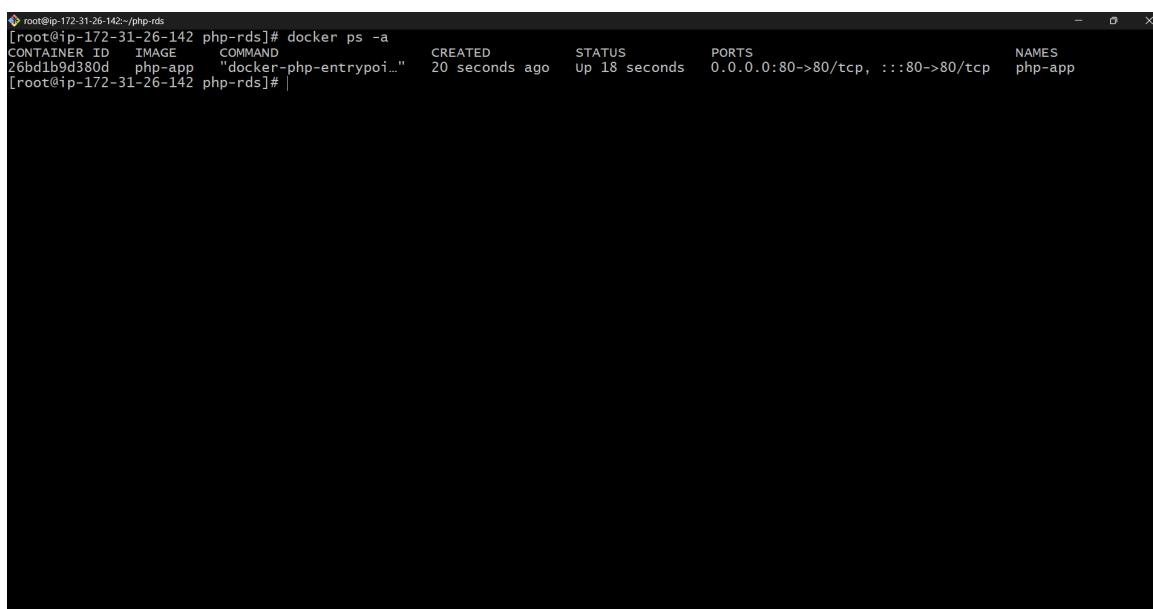
1. `git clone https://github.com/mahigore/php-rds.git`
2. `cd php-rds`

4.3 On the EC2 instance, build your Docker image using the docker build command and run it using the docker run command. Ensure the PHP application is able to connect and interact with the MySQL database hosted on AWS RDS.

1. `docker build -t php-app .`
2. `docker run`

```
docker run -d \
-p 80:80 \
```

```
-e DB_HOST=php-app-database.cby2mqygynq9.ap-southeast-1.rds.amazonaws.com \
-e DB_NAME=php-app-database \
-e DB_USER=admin \
-e DB_PASS=Mahi7888 \
--name php-app \
--restart unless-stopped \
php-app
```



A terminal window showing the output of the command `docker ps -a`. The output lists one container named `php-app` with the following details:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
26bd1b9d380d	php-app	"docker-php-entrypoi..."	20 seconds ago	Up 18 seconds	0.0.0.0:80->80/tcp, :::80->80/tcp	php-app

5. Testing

- Test your application by accessing the public IP of your EC2 instance. Verify that your application can read and write to the database.

Submit Feedback

Name:

Feedback:

Feedback submitted successfully!

All Feedback:

John Doe: This is a test feedback!
 Jane Smith: Great application, very easy to use.
 Mahesh: good

6. Set Up CI\CD (Jenkins):

Jenkins Server - Create an AWS EC2 instance

- **Basic Configuration:**
 - Name: Jenkins
 - AMI: Amazon Linux 2
 - Instance type: `t2.medium`
 - Key pair: [your key pair]
 - Security groups: default [ports → 22, 8080]
 - VPC: default

- **Install Git:**

```
yum install git
```

- **Install Docker:**

```
yum install docker
systemctl enable docker
systemctl start docker
systemctl status docker
```

- **Install Java**

```
# Become a root
sudo su -
sudo yum update
sudo yum install java-17-amazon-corretto-devel -y
java -version
```

- **Install Jenkins**

```
# Become a root
sudo su -

# Jenkins repo is added to yum.repos.d
sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo

# Import key from Jenkins
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key

# Install Jenkins
yum install jenkins -y

# You can enable the Jenkins service to start at boot with the command:
systemctl enable jenkins

# You can start the Jenkins service with the command:
systemctl start jenkins

# You can check the status of the Jenkins service using the command:
systemctl status jenkins
```

- Adding users to Jenkins group

```
sudo usermod -aG sudo jenkins
sudo usermod -aG docker jenkins
```

- Restart Jenkins: `http://<Your_Jenkins_URL>/restart`
- Access Jenkins Dashboard → `http://public-ip:8080`
- Copy password from CLI → `cat /var/lib/jenkins/secrets/initialAdminPassword`
- Install plugins
- **Create First Admin User**
- Start Jenkins

The screenshot shows the Jenkins dashboard with the following interface elements:

- Header:** Jenkins
- Left sidebar:**
 - + New Item
 - Build History
- Middle section:**
 - Welcome to Jenkins!**: A main heading with a subtext: "This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project."
 - Start building your software project**: A button labeled "Create a job" with a "+" icon.
 - Set up a distributed build**: Three buttons: "Set up an agent", "Configure a cloud", and "Learn more about distributed builds".
- Bottom right:** REST API, Jenkins 2.528.2

Create ECR Repo:

- Name = **project/php-app**

The screenshot shows the AWS ECR console with the following interface elements:

- Header:** AWS, Search [Alt+S], Account ID: 8649-8173-5502, mahi, Asia Pacific (Singapore)
- Breadcrumbs:** Amazon ECR > Private registry > Repositories
- Section title:** Private repositories (1)
- Table:**| Repository name | URI | Created at | Tag immutability | Encryption type |
| --- | --- | --- | --- | --- |
| project/php-app | 864981735502.dkr.ecr.ap-southeast-1.amazonaws.com/project/php-app | December 06, 2025, 12:28:17 (UTC+05:5) | Mutable | AES-256 |
- Actions:** View push commands, Delete, Actions ▾, Create repository
- Bottom:** CloudShell, Feedback, Console Mobile App, © 2025, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, Cookie preferences

Iam Role for EC2 to access ECR Repo:

- Name = ec2-ecr
- trust entity = ec2
- policy name = AmazonElasticContainerRegistryPublicFullAccess
- AmazonEC2ContainerRegistryPowerUser
- Create role
- Attach to Jenkins, Php-app ec2 instances

Jenkins Plugins to add:

- stage view
- blueocean
- click install
- Restart

Create a Jenkins Pipeline:

- Name - php-app
- Create pipeline
- string - App_Version
- Choose - pipeline script form SCM
- Add git repo url = <https://github.com/mahigore/php-rds.git>
- Branch = */main
- Script Path = Jenkinsfile
- Apply, Save

Jenkins Credentials to add:

- SSH key pair
 - ID = ssh-key
 - Description = ssh-key
 - Username = ec2-user
 - Private key = add ssh key here
- Database pass
 - ID = db-pass

- Description = db-pass
- secret = add password here

Jenkins Pipeline

```

pipeline {
    agent any

    environment {
        // AWS Configuration
        AWS_REGION = 'ap-southeast-1' // Your region
        AWS_ACCOUNT_ID = '864981735502'
        ECR_REGISTRY = "${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com"
        ECR_REPOSITORY = 'project/php-app'

        // EC2 Configuration
        EC2_HOST = '54.255.144.223' // Application EC2 (2nd EC2)
        EC2_USER = 'ec2-user'

        // Database Configuration (store only non-sensitive info here)
        DB_HOST = 'php-app-database.cby2mqyqynq9.ap-southeast-1.rds.amazonaws.com'
        DB_NAME = 'php-app-database'
        DB_USER = 'admin'
    }

    stages{
        stage("Repo Clone") {
            steps{
                echo "checkout github code"
                checkout scmGit(branches: [[name: '/main']], extensions: [], useRemoteConfigs: [[url: 'https://github.com/mahigore/php-rds.git']])
            }
        }
        stage("Docker Image Build") {
            steps {
                sh """

```

```

        echo "----- Building Docker Image -----"
        docker build -t php-app:latest .
        echo "----- Image Successfully Built -----"
        ...
    }
}

stage("Docker Image Tag") {
    steps{
        sh """
        echo "----- Tagging Docker Image -----"
        docker tag php-app:latest "${ECR_REGISTRY}"/"${ECR_REPOSITORY}":latest
        echo "----- Tagging Docker Image Completed."
        ...
    }
}

stage("Loggingin & Pushing Docker image to ECR") {
    steps {
        sh """
        echo "----- Logging To ECR -----"
        aws ecr get-login-password --region ${AWS_REGION} | docker login --username AWS --password-stdin ${ECR_REGISTRY}
        echo "----- Login Successful -----"

        echo "----- Pushing Docker Image To ECR -----"
        docker push "${ECR_REGISTRY}"/"${ECR_REPOSITORY}":latest
        echo "----- Docker Image Pushed Successfully -----"
        ...
    }
}

stage("cleanup") {
    steps {
        sh """
        echo "----- Cleaning Up Jenkins Machine -----"
        docker image prune -a -f
        echo "----- Clean Up Successful -----"
        ...
    }
}

```

```

    }

stage("Deployment Acceptance") {
    steps {
        input 'Trigger Deployment'
    }
}

stage('Deploy to EC2') {
    steps {
        script{
            echo 'Deploying to EC2 instance...'
            withCredentials([
                sshUserPrivateKey(credentialsId: 'ssh-key', keyFileVariable: 'SSH_KEY'),
                string(credentialsId: 'db-pass', variable: 'DB_PASS')
            ]) {
                sh """
                    # Deploy using SSH
                    ssh -o StrictHostKeyChecking=no -i \${SSH_KEY} \${EC2_USER}@\${EC2_HOST} << 'ENDSSH'
                    set -e

                    # Login to ECR
                    echo "Logging into ECR on EC2..."
                    aws ecr get-login-password --region \${AWS_REGION} | docker login --username AWS --password-stdin \${ECR_REGISTRY}

                    # Stop existing container
                    echo "Stopping existing container..."
                    docker stop php-app 2>/dev/null || true
                    docker rm php-app 2>/dev/null || true
                    docker rmi -f \$(docker images -aq) 2>/dev/null || true

                    # Pull latest image
                    echo "Pulling latest image..."
                    docker pull \${ECR_REGISTRY}/\${ECR_REPOSITORY}:latest

                    # Run new container
                    echo "Starting new container..."
                """
            }
        }
    }
}

```

```
docker run -d \
-p 80:80 \
-e DB_HOST=${DB_HOST} \
-e DB_NAME=${DB_NAME} \
-e DB_USER=${DB_USER} \
-e DB_PASS=${DB_PASS} \
--name php-app \
--restart unless-stopped \
${ECR_REGISTRY}/${ECR_REPOSITORY}:latest

# Verify container is running
echo "Verifying deployment..."
sleep 10
docker ps | grep php-app
```

Click Build Now:

- new image usng Jenkins CICD is build and deployed.

Feedback Form

Not secure 54.255.144.223

Apps Schedule Checks Setu Manish Batheja FinOps AWS Devops Linux Shopping AWS 3 tier project Arohi Software Workshop on AWS...

Submit your Feedback

Name:

Feedback:

Feedback submitted successfully!

All Feedback:

John Doe: This is a test feedback!
Jane Smith: Great application, very easy to use.
Mahesh: good
Maheshgore:
Maheshgore:
Maheshgore:
:
Maheshgore:

Video Links:

- Bitcot Task Part 1 →
[https://drive.google.com/file/d/1qjZCcccLrf6zMJm9WqZ8vkda7wr6mdqG/view?
usp=drive_link](https://drive.google.com/file/d/1qjZCcccLrf6zMJm9WqZ8vkda7wr6mdqG/view?usp=drive_link)
 - Bitcot Task Part 2 →
[https://drive.google.com/file/d/19EbT9neO4LOhJnp_3MxJgY7h77mNwjDR/view?
usp=drive_link](https://drive.google.com/file/d/19EbT9neO4LOhJnp_3MxJgY7h77mNwjDR/view?usp=drive_link)