**Setting Up AWS Ubuntu Instance with Jenkins, Docker, and MySQL RDS**

**1. Create an AWS Ubuntu Instance and RDS MySQL Connection**

**1.1 Launch an AWS EC2 Ubuntu Instance**

1. Log in to the AWS Management Console.
2. Navigate to **EC2** under the **Compute** section.
3. Click **Launch Instance**.
4. Select an **Ubuntu** Amazon Machine Image (AMI).
5. Choose an instance type (e.g., t2.micro for testing, or a higher instance type for production).
6. Configure instance details:
   * Set up a security group to allow necessary ports (e.g., SSH - 22, Jenkins - 8080, MySQL - 3306, Docker - custom if needed).
   * Attach an IAM role with permissions to access RDS if necessary.
7. Add storage as required.
8. Configure key pair for SSH access.
9. Click **Launch**.

**1.2 Set Up MySQL RDS Connection**

1. Navigate to **RDS** under the **Database** section in AWS.
2. Click **Create Database** and choose **MySQL**.
3. Select the instance type and configure storage.
4. Set up a database username and password.
5. Enable public access if you want external connections.
6. Add the EC2 instance’s security group to the **RDS security group** for access.
7. Note down the **endpoint URL** for connection.

**2. Install Java, Maven, Jenkins, and Docker on Ubuntu Instance**

**2.1 Update System Packages**

sudo apt update && sudo apt upgrade -y

**2.2 Install Java (OpenJDK 17 recommended)**

sudo apt install openjdk-17-jdk -y

java -version

**2.3 Install Maven**

sudo apt install maven -y

mvn -version

**2.4 Install Jenkins**

wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null

echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian binary/" | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt update

sudo apt install jenkins -y

**2.5 Install Docker**

sudo apt install docker.io -y

sudo systemctl enable --now docker

**3. Run Docker Without Sudo**

**3.1 Add Docker Group (If Not Exists)**

sudo groupadd docker

**3.2 Add User to Docker Group**

sudo usermod -aG docker $USER

**3.3 Log Out and Log Back In**

exit

**3.4 Verify Docker Group Assignment**

groups

**4. Start Jenkins**

sudo systemctl restart jenkins

**5. Install Jenkins Plugins**

In Jenkins UI:

1. Go to **Manage Jenkins** > **Manage Plugins**.
2. Install the following plugins:
   * Docker Pipeline
   * SSH Agent
   * SSH Pipeline
   * Git Client
   * Git

**6. Add Credentials in Jenkins**

1. Navigate to **Manage Jenkins** > **Manage Credentials**.
2. Add the following credentials:
   * **Git Personal Access Token (PAT)** for repository access.
   * **EC2 Private Key (PEM file)** for SSH access.
   * **Docker Hub Password** for image push/pull access.

**7. Configure Jenkins to Use Docker**

**7.1 Check Jenkins Running User**

ps aux | grep jenkins

**7.2 If Jenkins Runs as jenkins User, Add to Docker Group**

sudo usermod -aG docker jenkins

sudo systemctl restart jenkins

**7.3 Verify Docker Access for Jenkins**

sudo -u jenkins docker ps

**8. Running Jenkins and Docker to Consume Services and Run Builds**

**8.1 Start Jenkins Service**

sudo systemctl start jenkins

**8.2 Enable Jenkins to Start on Boot**

sudo systemctl enable jenkins

**8.3 Verify Jenkins is Running**

sudo systemctl status jenkins

**8.4 Start Docker Service**

sudo systemctl start docker

**8.5 Enable Docker to Start on Boot**

sudo systemctl enable docker

**8.6 Verify Docker is Running**

sudo systemctl status docker

**8.7 Run a Jenkins Build**

1. Open Jenkins UI (http://<your-ec2-public-ip>:8080).
2. Create a **New Item** and select **Pipeline**.
3. Configure the pipeline to pull the project from **Git**.
4. Set up a **Jenkinsfile** in the repository with build and Docker steps.
5. Click **Build Now** to trigger the build.
6. Monitor logs and ensure successful execution.

**Conclusion**

After completing these steps, your AWS Ubuntu instance should be ready with Jenkins, Docker, and MySQL RDS integration. You can now create CI/CD pipelines leveraging these tools and run automated builds successfully.