Implementing a CI/CD Pipeline with Jenkins, GitHub, and Docker on AWS EC2 for Automated Application Deployment

Steps: Implementing an Enhanced CI/CD Pipeline with Jenkins, GitHub, Docker, and AWS EC2

---

1. Create EC2 Instance:

- **Launch EC2 Instance:**

  Go to the **AWS Management Console** and navigate to the EC2 Dashboard.

  Click on **Launch Instance** and select **Ubuntu Server 22.04 LTS (HVM), SSD Volume Type**.

  Choose the instance type as **t2.micro** (free tier eligible).

  Click **Next: Configure Instance Details**, ensuring that you select a VPC and Subnet (use the default options).

  Ensure **Auto-assign Public IP** is enabled so the instance can be accessed externally.

- **Configure Network Settings:**

  In **Step 6: Configure Security Group**, create a new security group and add the following inbound rules:

  - **SSH (Port 22)** from your IP address (or anywhere if needed).
  - **HTTP (Port 80)** from anywhere (for your application).
  - **Custom TCP Rule for Jenkins (Port 8080)** from anywhere (for Jenkins UI).

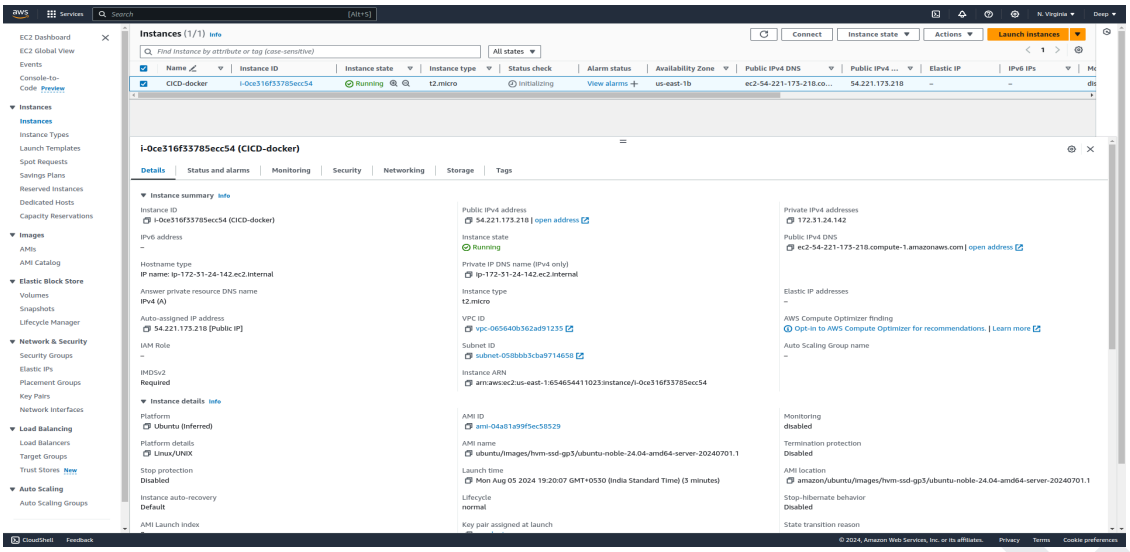
- **Key Pair:**

  Create or select an existing key pair. This is important for SSH access.

  Download the private key file (`.pem`) and keep it safe. You'll need it to SSH into the instance.

- **Launch the Instance:**

  Review the details and click **Launch**. Wait for the instance to be ready.



---

2. Install Jenkins:

- **SSH into the EC2 Instance:**

Open a terminal and SSH into the EC2 instance using the key pair:

```
ssh -i /path/to/your-key.pem ubuntu@<EC2-Public-IP>
```

- **Update System:**

Once logged in, update the system packages:

```
sudo apt update && sudo apt upgrade -y
```

- **Install Java:**

Jenkins requires Java to run. Install OpenJDK 11:

```
sudo apt install openjdk-11-jdk -y
```

- **Verify the Java installation by checking the version:**

```
java -version
```

- **You should see output similar to:**

```
openjdk version "11.0.X" ...
```

- **Install Jenkins:**

Add the Jenkins repository and import the GPG key:

```
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee \
/usr/share/keyrings/jenkins-keyring.asc > /dev/null
sudo sh -c 'echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
https://pkg.jenkins.io/debian-stable binary/ > \
/etc/apt/sources.list.d/jenkins.list'
sudo apt update
sudo apt install jenkins -y
```

- **Start Jenkins and Enable on Boot:**

Start Jenkins and enable it to run on boot:

```
sudo systemctl start jenkins
sudo systemctl enable jenkins
```

- **Access Jenkins:**
  - Open your browser and navigate to http://<EC2-Public-IP>:8080.

Retrieve the Jenkins initial admin password by running this command:

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

  - Complete the Jenkins setup by entering the password and following the setup instructions.

---

3. Integrate Jenkins with GitHub:

- **Create Jenkins Project:**
  - In Jenkins, click on **New Item** and create a **Freestyle Project**.
  - Under the **Source Code Management** section, select **Git** and provide your GitHub repository URL.

    - Ensure **GitHub hook trigger for GITScm polling** is selected under **Build Triggers**.
- **Generate SSH Key Pair:**

Generate a new SSH key pair on the EC2 instance to allow Jenkins to access GitHub:

```
ssh-keygen -t rsa -b 4096 -C "your-email@example.com"
```

Copy the public key:

```
cat ~/.ssh/id_rsa.pub
```

- **Add SSH Key to GitHub:**
    - Go to your GitHub repository, navigate to **Settings** > **Deploy Keys**, and add the public key.
- **Configure Jenkins with Private Key:**
    - In Jenkins, go to **Manage Jenkins** > **Manage Credentials**, and add the private key for SSH access to GitHub.
- **Set Up GitHub Webhook:**

In GitHub, go to **Settings** > **Webhooks** and add a new webhook with the URL:

```
http://<EC2-Public-IP>:8080/github-webhook/
```

    - Select the **Just the push event** option to trigger Jenkins builds on code changes.

---

4. Set Up Docker:

- **Install Docker:**

Install Docker on the EC2 instance:

```
sudo apt install docker.io -y
```

- **Start Docker and Enable on Boot:**

Start Docker and ensure it runs on boot:

```
sudo systemctl start docker
```

```
sudo systemctl enable docker
```

- **Add Jenkins User to Docker Group:**

To allow Jenkins to run Docker commands, add the Jenkins user to the Docker group:

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

  - Log out and back in or restart Jenkins for the changes to take effect.

---

5. Containerize the Application:

- **Create a Dockerfile:**

In your GitHub repository, create a `Dockerfile` to define how your application should be containerized. Example:

```
# Use official Node.js image as base
FROM node:14-alpine as build

# Set the working directory inside the container
WORKDIR /app

# Copy package.json and package-lock.json to the working directory
COPY package*.json ./

# Install dependencies3.83.23.224
RUN npm install

# Copy the entire project directory into the container
COPY . .

# Build the React app
RUN npm run build

# Stage 2: Use a lightweight web server to serve the static content
FROM nginx:alpine

# Copy the build output from the 'build environment' stage to the nginx web
root directory
COPY --from=build /app/build /usr/share/nginx/html

# Expose port 80 to the outside world
```

```
EXPOSE 80

# Start nginx server when the container starts
CMD ["nginx", "-g", "daemon off;"]
```

- **Build Docker Image via Jenkins Pipeline:**

Add a build step in the Jenkins pipeline to build the Docker image:

```
docker build -t <image-name> .
```

- **Push Docker Image to Registry:**

Push the built image to Docker Hub or Amazon ECR:

```
docker tag <image-name> <your-dockerhub-username>/<image-name>
docker push <your-dockerhub-username>/<image-name>
```

- **Run Docker Container:**

Run the Docker container on your EC2 instance:

```
docker run -d -p 80:80 <your-dockerhub-username>/<image-name>
```

    -

---

6. Automate Builds and Deployment:

- **Configure Jenkins for Automated Builds:**
    - Set up Jenkins to automatically build Docker images on every GitHub commit by configuring the **Build Triggers** in Jenkins (e.g., **GitHub hook trigger**).
- **Automate Deployment:**

Add post-build actions in Jenkins to deploy the built Docker image to the EC2 instance. This can be done by running a shell script as part of the Jenkins job to pull the new image and restart the container:

```
docker pull <your-dockerhub-username>/<image-name>:latest
docker stop <container-id>
docker rm <container-id>
docker run -d -p 80:80 <your-dockerhub-username>/<image-name>:latest
```

- **Verify Deployment:**
    - Access your application via the EC2 instance's public IP (e.g., `http://<EC2-Public-IP>`), confirming that the latest version has been deployed.