Master class

git clone https://github.com/amitopenwriteup/cicdproject.git

cd cicdproject

Explain about repo

**Step 1: Enable Required APIs**

Before you start, enable the necessary Google Cloud APIs:

gcloud services enable cloudbuild.googleapis.com \

artifactregistry.googleapis.com \

container.googleapis.com

**Step 2: Create an Artifact Registry Repository**

Artifact Registry is used to store container images.

1. Create a repository in **Artifact Registry**:

gcloud artifacts repositories create my-repo \

--repository-format=docker \

--location=us-central1

1. Authenticate Docker to use **Artifact Registry**:

gcloud auth configure-docker us-central1-docker.pkg.dev

**Step 3: Configure Cloud Build**

Cloud Build will be used to build and push the Docker image.

1. **Create a cloudbuild.yaml file** in your project root:

substitutions:

\_TAG: "latest"

steps:

- name: 'gcr.io/cloud-builders/docker'

args:

- 'build'

- '-t'

- 'us-central1-docker.pkg.dev/$PROJECT\_ID/my-repo/my-app:$\_TAG'

- '.'

- name: 'gcr.io/cloud-builders/docker'

args:

- 'push'

- 'us-central1-docker.pkg.dev/$PROJECT\_ID/my-repo/my-app:$\_TAG'

images:

- 'us-central1-docker.pkg.dev/$PROJECT\_ID/my-repo/my-app:$\_TAG'

Get the projects, and provide your projects list as per access [highlighted in red]

gcloud projects list

gcloud builds submit --config=cloudbuild.yaml --project=upgradlabs-1738334790345

Automatic trigger

**Automatic Build Trigger from GitHub**

1. Go to **Cloud Build > Triggers** in the GCP Console.
2. Click **Create Trigger**.
3. Select **GitHub** as the source and link your repository.
4. Set trigger conditions (e.g., push to main branch).
5. Use cloudbuild.yaml as the build configuration.
6. Save and enable the trigger.

Check your repo

gcloud artifacts docker images list us-central1-docker.pkg.dev/upgradlabs-1738334790345/my-repo

**Next Steps: Deploy to Kubernetes (Optional)**

If you have a **GKE cluster**, deploy the built image:

1. Authenticate **kubectl**:

gcloud container clusters get-credentials my-cluster --zone us-central1-c

1. Deploy to Kubernetes:

kubectl create deployment my-app --image=us-central1-docker.pkg.dev/$PROJECT\_ID/my-repo/my-app:latest

kubectl expose deployment my-app --type=LoadBalancer --port=80 --target-port=8080