**GCP DEMO – CI/CD Pipeline**

**#pushing an image to GCR & Pulling an image from GCR**

-->create a project with name cicd-demo-1234

-->enable GCR API

goto->API&Services->+enable API & Services->Container Registry ->Enable

-->Activate Cloud Shell from GCP Console

$ gcloude auth list

--To check cred are active

$ docker -v

--To check version available

$ gcloud auth configure-docker

--To configure docker to use gcloud cred with GCR

--Has produced config.json in ./docker directory

**#lets specify this location in path environmental variable**

$ export PATH=$PATH:~/.docker

$docker images -a

--to list the images

**#Create a simple docker image**

$ mkdir -p cicd/demo01

--create a directory with name cicd/demo01

$vi welcome.sh

--create a file

--add a message

echo "Welcome to the CICD Pipeline GCP demo"

--press esc key, :wq - to save and quit

$chmod +x welcome.sh

--give execute permission

$ vi Dockerfile

--create a docker file and add the following

From alpine

copy welcome.sh /

CMD ["sh","welcome.sh"]

**#Now we will build docker image locally by running a command**

$ docker build .

--output

--sucessfully build <image\_id>

**#Let us name the image in certian formate -->refer PPT**

$ docker tag <image\_id> eu.gcr.io/<project\_id>/welcome

--ex: docker tag b2e0c9117e65 eu.gcr.io/cicd-demo-1234/welcome

--Check container registry--refresh

**#We will perform pull operation from docker, before that, remove all docker images stored locally**

$ docker images -a

--list all the images available locally

$ docker rmi $(docker images -a -q)

(Or)

$ docker rmi –f [images] [image]…

$ docker images -a

--empty list

**#Now let us pull the welcome image from repositry**

$ docker pull eu.gcr.io/cicd-demo-1234/welcome

$ docker images -a

**#Cloud Source Repositories**

goto->source repositories->+Add repositories-->create new repositories

Rep Name: pipeline-demo

project: <project\_ID>

**#Google Cloud Build**

**#Create CI/CD Pipeline, that automatically builds the docker image for your application.**

Enable Cloud Buid API

--search Cloud Build API in GCP console search bar and enable it

Open Cloud shell

**#setting up environment variable**

$ export PROJECT\_ID=<project-id>

--ex: cicd-demo-1234

$ ZONE=europe-west2-a

--we can give any region/zone

**#Enable GKE API**

$ gcloud services enable container.googleapis.com

**#Create a GKE cluster**

$ export GKE\_CLUSTER\_NAME=<cluster\_name>

--ex: export GKE\_CLUSTER-NAME=pipeline-demo

$ gcloud container clusters create ${GKE\_CLUSTER\_NAME} --project=${PROJECT\_ID} --zone=${ZONE} --scopes "https://www.googleapis.com/auth/cloud-platform"

goto->GKE Engine @ GCP console-->check the cluster is creating

Now the GKE cluster is available for you to deploy the Application

**#Enable kubernetesEngineDeveloperRole**

goto->IAM->select @cloudbuild.gserviceaccount-->click on pincile symbol-->+add kubernetesEngineDeveloper role-->save

**#Let us clone our cloud source repository to cloud shell instance**

goto->cloud shell

$ gcloud source repos clone <repos name> --project=<project-id>

--ex: gcloud source repos clone pipeline-demo --project=cicd-demo-1234

**#Change the directory to cloned repos name**

$cd <repos-name>

--ex: cd pipeline-demo

**# Branch out of master and create a feature branch as TICKET-001**

$ git checkout -b <feature-branch-name>

--ex: git checkout -b TICKET-001

**#Refere source\_code\_pipeline-demo-repos.zip file -->available at lms**

goto->cloud shell editor-->select <repos\_name> ex: pipeline-demo

step1: Create 2 sub folders under **pipeline-demo**

--app

--deploy

Step2: Create 2 file under **app** sub folder

--app.py - Copy and paste flask code from .zip file

--test\_app.py - Copy and paste content from .zip file

goto->cloud shell

**# Execute both app.py and test\_app.py to check end points are working properly**

$ python3 app/test\_app.py

$ python3 app/app.py

So far, we have done in our local file station, i.e cloud shell.

**# Now, Let us containerized this application by creating Dockers file.**

Step1: Create a new file under app sub folder.

--Dockerfile – copy and paste the content from .zip file.

- this file uses a python3 as a base image and install required packages.

Step2: Create a new file under pipeline-demo directory.

--cloudbuild.yaml – copy and paste first 3 steps from .zip file.

- cloudbuild.yaml has 3 steps

To run the test

Build the docker image

Push docker image to GCR

We are done with CI part of CI/CD pipeline…

**#Create a cloud build trigger**

-To run these steps as soon as push some code to the repository

-Once configured cloud build trigger successfully, It automatically build once the code is pushed to source repository based on rules defined in the configuration.

Goto->cloud build🡪triggers🡪click new trigger

Name: flaskapp

Desc: Flask App Trigger

Event: Select push to a branch

Source: select repository name

Pipeline-demo(cloud source repository)

Branch: select .\* //regular expression

Build config: file type 🡪select cloud build config file(yaml or json)

cloud build conf file location

/cloudfile.yaml

Click 🡪create

Trigger:-

🡪Automatically triggers – build automatically when code is pushed to any branch of source code repository.

🡪Manual trigger – We can click on manual trigger

**#Let us test the trigger**

Goto🡪cloud shell🡪editor window

Create a file under pipeline-demo

* .gitignore – copy and paste the content from .zip file.

**# Run a git command to commit and push the local code to the cloud source repository**

$ git add . && git commit –m “add code”

$ git push origin TICKET-001

Goto🡪cloud buid🡪dashboard

View the history page

Goto🡪container registry // to see docker images

**# Continuous Delivery Pipeline**

Input: immutable artifacts

Deploy: GKE cluster & Cloud build

Goto🡪 cloud shell🡪editor window

Create a new file under deploy sub folder.

--kube-app.yaml.tpl – copy and paste the content from .zip file.

Go back to cloudebuild.yaml file

--Initially, we have added 3 steps, now add remaining steps from .zip file.

**# Lets commit and push the code**

$ git add . && git commit –m “Add Kubernetes template”

$ git push origin TICKET-001

Goto🡪cloud build🡪history🡪 we can see 2 additional steps

**# after successful build, we will verify by connecting to GKE cluster**

Goto🡪cloud shell

$ gcloud container clusters get-credentials pipeline-demo --zone europe-west2-a --project <project\_id>

**# We can run kubectl command and interact with the cluster**

$ kubectl get deployments //to list the deployments

$ kubectl get pods // to list the pods and status

**# we can check the logs of the pods**

$ kubectl logs pod/<podsname>

**#We can expose our application using kubernetes service**

$ kubectl get svc

--copy the external IP and paste in new tab

We could see the message from our application

**# to see another route**

External\_IP/today

We could see the current date

\*\*\*\*END\*\*\*