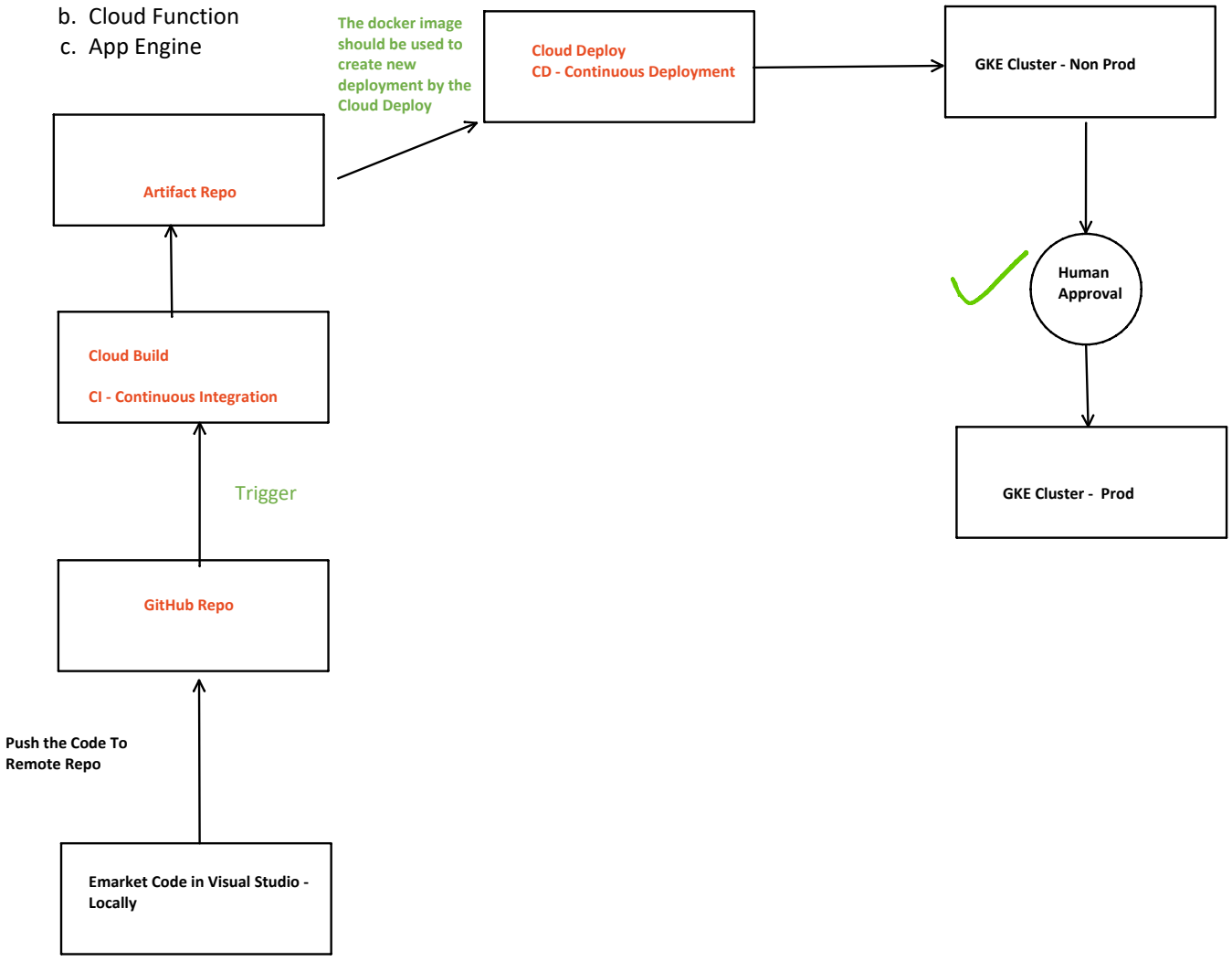
Day 9

Topics

1. CI/CD Continued
   a. GKE Deployment
   b. Cloud Build - Continuous Integration
   c. Cloud Deploy - Continuous Deployment
2. **Serverless**
   a. Cloud Run
   b. Cloud Function
   c. App Engine



- Inside a GKE Cluster:
  - Service [Load balancer Public Ip]
  - Deployment.yaml

Demo GCP CI/CD

Part 1 - Create the app

1. Created a emarket application in local machine on visual studio.
2. Tested the application on local host.

Part 2 - Containerize the app locally

1. We created the docker file for the application code.
2. Containerized the application

Part 3 - Push your Code to GitHub

1. Created a new repository on GitHub.
2. Add the remote repo to our local IDE.
3. Pushed my code to the main branch in GitHub.

Part 4 - Create infrastructure of GKE

1. Created:
    a. GKE - Non prod
    b. GKE - Prod Clusters

```
gcloud container clusters create prod-cluster \
 --zone us-central1-c \
 --machine-type g1-small \
 --disk-size 32 \
 --release-channel rapid \
 --num-nodes 1 \
 --no-enable-autoscaling \
 --logging=NONE \
 --monitoring=NONE
```

Part 5 - Create Artifact Registry

- To Save my Docker files on GCP and to be used by Cloud Build

Part 6 - Create Cloud Build

- Cloud build will use your deployment file from the github and create a new docker image and push it to the Artifact registry.
- **Trigger** - This will automatically start build a new image the moment we push a code to github.

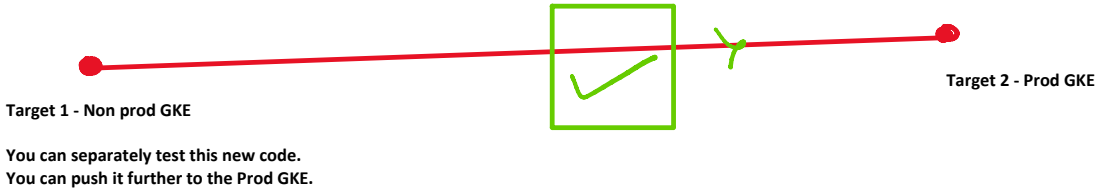
CloudBuild.yaml

```yaml
steps:
# Step 0
#docker build the image from httpd public dockerhub Apache image
  - name: 'gcr.io/cloud-builders/docker'
    args: [ 'build', '-t', 'us-central1-docker.pkg.dev/project-alpha12/emarket-repo/emarket-image',
'./App' ]

# Step 1
#push the previously built image to the docker repo created in Step 0
  - name: 'gcr.io/cloud-builders/docker'
    args: ['push', 'us-central1-docker.pkg.dev/project-alpha12/emarket-repo/emarket-image']

# Step 2
# Publish the release
  - name: 'gcr.io/google.com/cloudsdktool/cloud-sdk'
    entrypoint: 'bash'
    args:
    - '-c'
    - |
      gcloud deploy releases create r-$BUILD_ID \
      --delivery-pipeline="emarketpipeline" \
      --region="us-central1" \
      --images="emarket-image=us-central1-docker.pkg.dev/project-alpha12/emarket-repo/emarket-
image:latest" \
      --source="./"
options:
  logging: CLOUD_LOGGING_ONLY  # Use Cloud Logging for logs
```

```
  defaultLogsBucketBehavior: REGIONAL_USER_OWNED_BUCKET  # (Alternative) Use regional bucket for logs

# **Fix: Ensure Service Account is Configured**
serviceAccount: "projects/project-alpha12/serviceAccounts/896205892070-
compute@developer.gserviceaccount.com"
```

Part 7 - Create Cloud Deploy

- Cloud deploy will use the 'latest' docker image created by Cloud Build and the deploy it to the GKE infrastructure.
- **Pipeline**
  - **Target -** The Deployment infrastructure. In our case GKE.



Skaffold.yaml

```
apiVersion: skaffold/v2beta16
kind: Config
deploy:
  kubectl:
    manifests: ["k8s/*.yaml"]
profiles:
- name: non-prod
  deploy:
    kubectl:
      manifests: ["k8s/nonprod/*.yaml"]
- name: gke-prod
  deploy:
    kubectl:
      manifests: ["k8s/prod/*.yaml"]
```

Serverless [PAAS - Platform As A Service]

1.Cloud Run
2.Cloud Function
3.App Engine

- Build and run software without provisioning, managing or maintaining physical or virtual machines.
- Remove everything that distracts you from writing/improving or adding new features to your app.

Benefits

- No maintenance of infrastructure.
- Elastic
- Portable
- Pay only for what you use.
- Event-driven

Managed Services vs Serverless

Having a third party manage the installation, maintenance, updating and uptime of a software packaged on behalf of the user.
You still pay for infrastructure/networking/storage regardless you use it or not !

- Serverless - You only pay for the memory used/CPU used and network used.

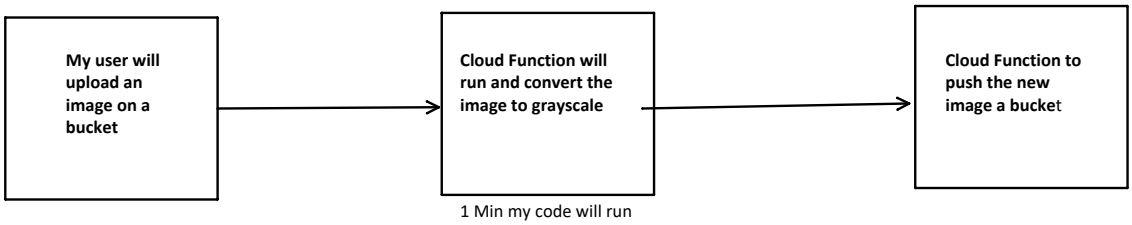
- App Engine - Managed app platform [2008]
- Cloud Function - Event Driven serverless [2017]
- Cloud Run - Serverless setup to run your containers. [2019]

Cloud Run

- Serverless setup to run your containers.

Cloud Function

- Lets say I have an app to convert an image to grayscale.



Main.py

Requirements.txt