Day 10

Topics

1.Demo

- **Cloud Run**
- **Cloud Run Function**
- **App Engine**

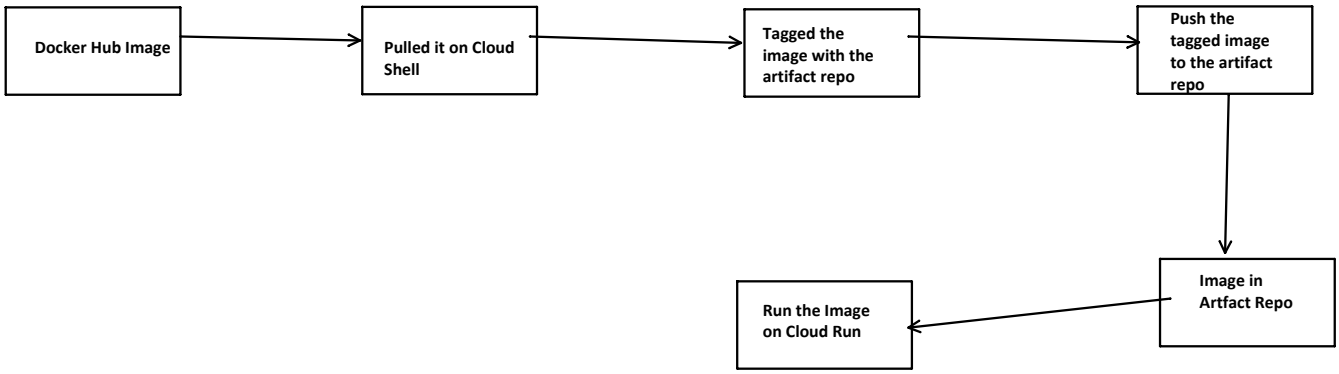
2. Security

a. **IAM**

b. **CMEK**

c. **IAP**

Demo - Cloud Run - Bring your containers and just run your containers.



Demo - Cloud Run Function

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

Main.py

```
import functions_framework
import os
from google.cloud import storage
from PIL import Image
import io
# Initialize Google Cloud Storage client
storage_client = storage.Client()
# Destination bucket for processed images
DESTINATION_BUCKET = "my-processed-bucket"
@functions_framework.cloud_event
def gcs_trigger(cloud_event):
    """Triggered by a change to a Cloud Storage bucket."""
    data = cloud_event.data
    file_name = data["name"]
    bucket_name = data["bucket"]

    print(f"New file uploaded: {file_name} in bucket: {bucket_name}")
    # Process only image files
    if file_name.lower().endswith(('.png', '.jpg', '.jpeg')):
        print(f"Processing image: {file_name}")
        convert_image_to_grayscale(bucket_name, file_name)
    else:
        print("Not an image file. No action taken.")
def convert_image_to_grayscale(bucket_name, file_name):
    """Convert an image to grayscale and upload to another bucket"""
    bucket = storage_client.bucket(bucket_name)
    blob = bucket.blob(file_name)

    # Download image into memory
    image_bytes = blob.download_as_bytes()
    image = Image.open(io.BytesIO(image_bytes)).convert("L")  # Convert to grayscale
    # Save to output bucket
    output_bucket = storage_client.bucket(DESTINATION_BUCKET)
    output_blob = output_bucket.blob(f"grayscale-{file_name}")
    with io.BytesIO() as output_bytes:
```

Requirements.txt

```
google-cloud-storage
functions-framework
pillow
```

```
    image.save(output_bytes, format="PNG")
    output_blob.upload_from_string(output_bytes.getvalue(), content_type="image/png")
print(f"Processed image saved to: gs://{DESTINATION_BUCKET}/grayscale-{file_name}")
```

App Engine

- Fully managed PAAS.
- It is going to help you to deploy,scale and manage web applications without managing the infrastructure.

Features Of App Engine/Cloud Run/Cloud Function

1. Fully managed
2. Automatic scaling
3. Built in monitoring - Cloud logging and monitoring
4. Secured by default

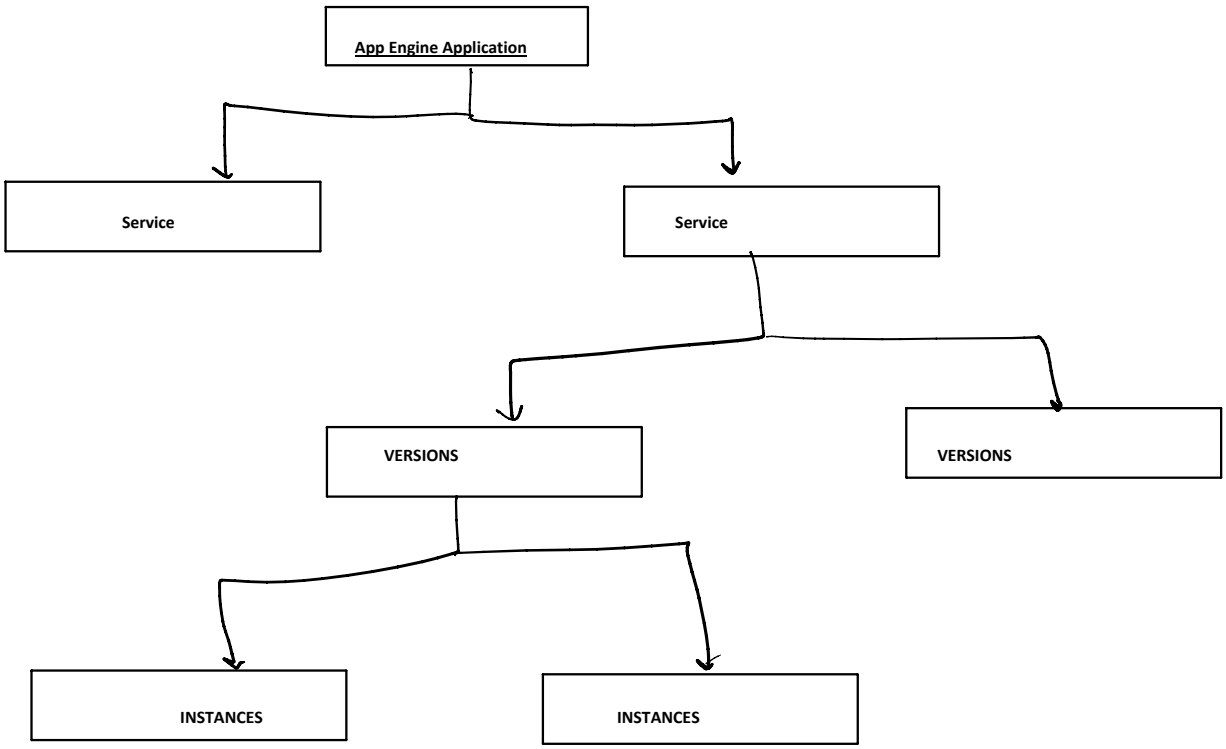
Any app engine service you create in a project. You cannot delete it.
You can only disable app engine or delete the project.

You have 2 options that are available with App Engine:

Standard Environment	Flexible Environment
Code in few languages/version only.	Code in far more languages.
No background processes support.	Background processes supported.
No SSH Debugging	SSH debugging is supported
Scale to zero.	No scaling to zero.
No installation of third party binaries.	Installation of third party binaries is supported.

Choose the flexible env for application that require custom runtimes or third party libraries that are not supported in standard env.

App Engine Components



Service - This is like a microservice, which can scale up/down independently.
Every application will have atleast one service - The **default service.**
Versions - Multiple versions of each service can be deployed. Every new deployment to a particular service creates a new version.
Instances - Versions of your service, run on one or more instance.

Demo Code For App Engine

Main.py

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def home():
    return "Hello, Google App Engine!"
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080)
```

Requirements.txt

```
Flask
gunicorn
```

App.yaml

```
runtime: python39  # Specify Python runtime
entrypoint: gunicorn -b :$PORT main:app  # Use Gunicorn as WSGI server
handlers:
- url: /.*
  script: auto
```

	Cloud Run	Cloud Function	App Engine
Use Case	Deploy containerized application	Deploy event-driven functions	Deploy web app or a services
Execution Model	Fully managed container service	Serverless function execution	PAAS for web applications Run your code here.
Supported Runtimes	Any runtime - inside a Docker container	Node.js, Python, Ruby,Java,PHP, .NET Go	Supports most of the languages.
Deployment Unit	Docker container	Function code	Application code
Pricing	Pay per request	Pay per execution	Pay per instance runtime
Best For	Microservices, APIs	Event driven architecture	Web Apps Monolithic apps

GCP Security

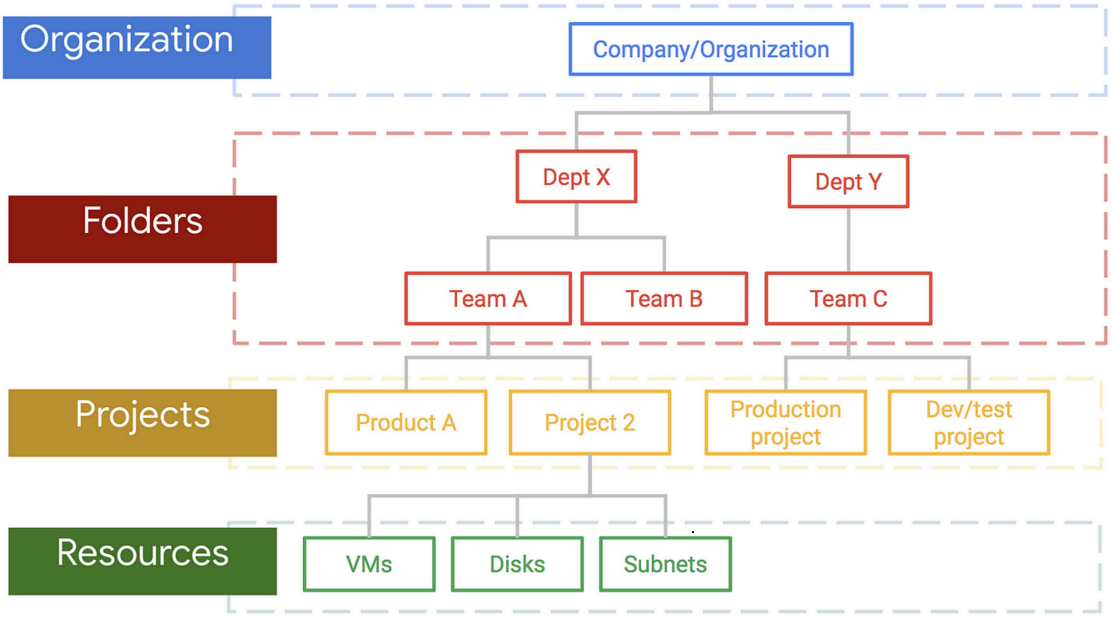
a. **IAM - Identity and Management Service**
b. **CMEK - Customer Managed Encryption Keys**
c. **IAP - Identity Aware Proxy**

IAM - Identity and Access Management Service

- WHO can do WHAT on WHICH Resource.

WHO - Person/Group/Application
WHAT - Specific rights/ACTIONS
WHICH - Any GCP Service

CLOUD IAM HIERARCHY



Moving a project into different GCP organization is possible.
The moment you move a project to new org. All the policies from the new org get applied to the project.
Policies applied at the org level will take precedence.

IAM Roles
3 Types:

1. **Primitive Roles** - Applied across all the GCP services inside a project.
2. **Predefined Roles** - Applied to a particular GCP service
3. **Custom Roles** - Precise set of permission.

Always Follow The Principle Of Least Privilege !!

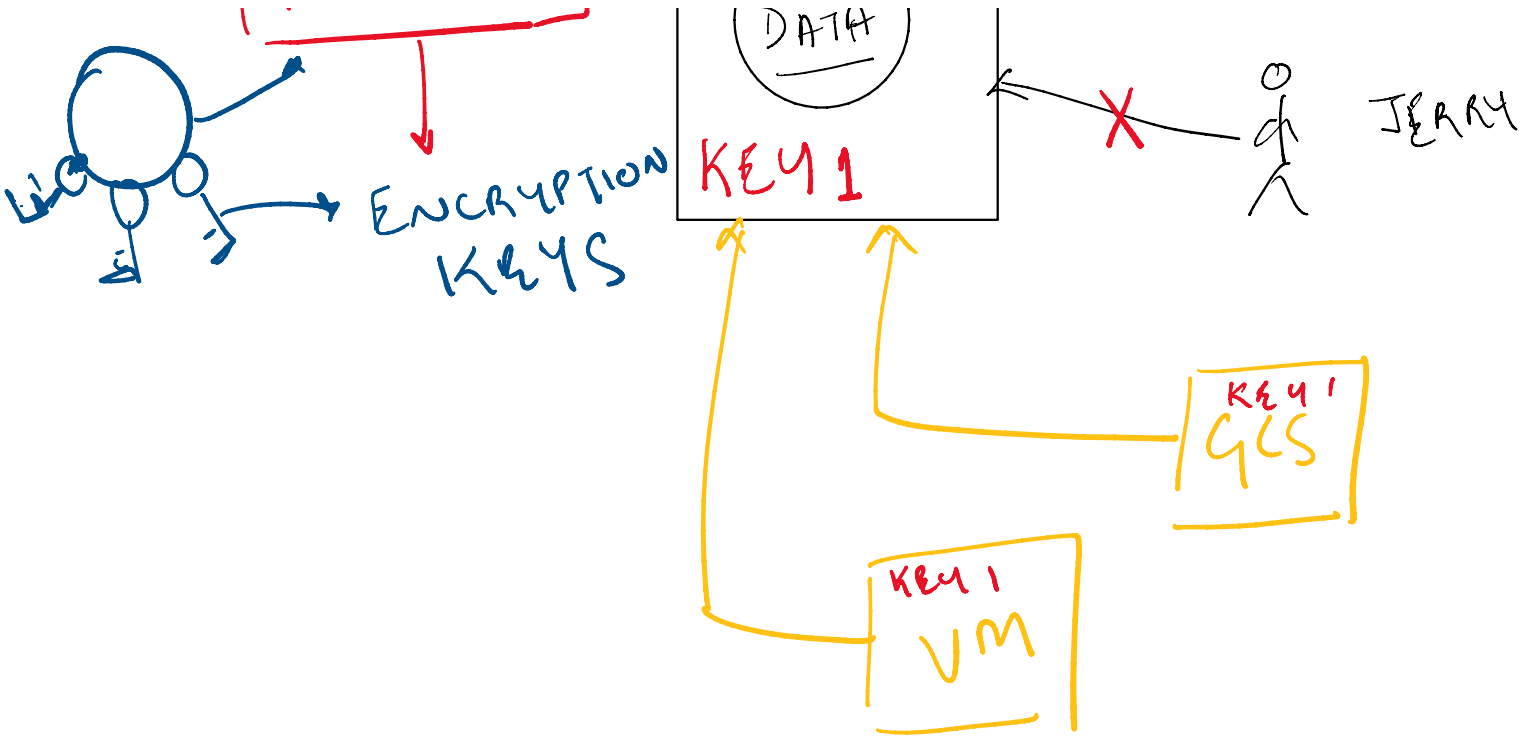
IAP - Identity Aware Proxy

- To control access to web applications and cloud resources based on **user's identity** and context [device,location]

Traditional - Firewall and VPNs
Cloud IAP - Google's IAM roles

Google Cloud KMS

- Cloud Key Management Service
- Allow you to create, manage and use encryption keys to protect your data.
- GCS, BigQuery,Compute Engine and Cloud SQL.


DATA
KEY 1
ENCRYPTION
KEYS
JERRY
KEY 1
GCS
KEY 1
VM