Google Cloud

Sergio Paniego Blanco
@sergiopaniego
sergiopaniegoblanco@gmail.com
https://sergiopaniego.github.io/

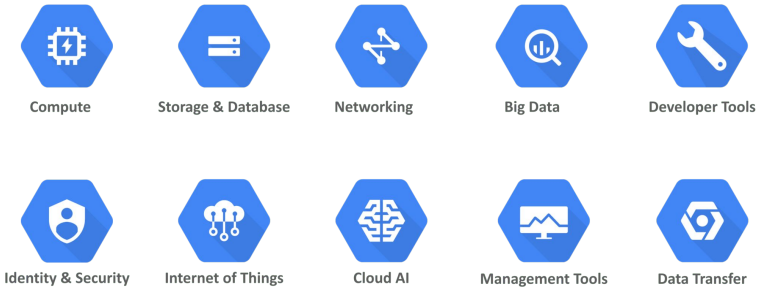
Google Cloud

- Fundamentals of Google Cloud Platform (GCP)
- Setting up a project in Google Cloud Platform
- Deploying Python applications on Google App Engine
- Data storage in Google Cloud Storage
- Using additional GCP Services (Firestore and BigQuery)

Google Cloud

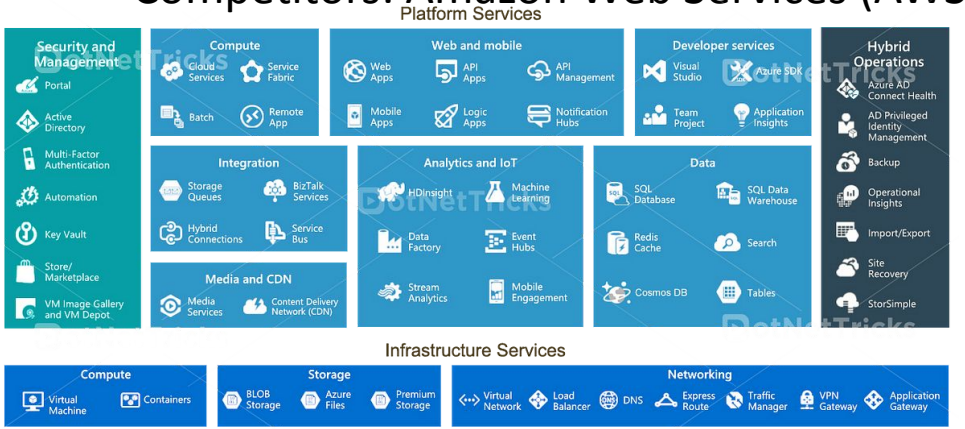
Fundamentals of GCP

- Cloud computing: delivery of computing services (servers, storage, databases, networking, software) over the internet (“the cloud”).
- Key features:
  - **On-Demand Self-Service**: Users can provision resources as needed.
  - **Broad Network Access**: Services are available over the network and accessed through standard mechanisms (e.g., web browsers).
  - **Resource Pooling**: Provider’s resources are pooled to serve multiple consumers.
  - **Rapid Elasticity**: Resources can be quickly scaled up or down.
  - **Measured Service**: Cloud systems automatically control and optimize resource use.



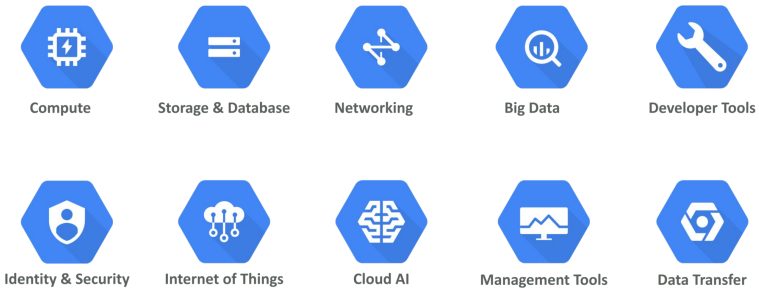
Fundamentals of GCP

- Specific Benefits of GCP:
  - **Scalability:** Easily scale applications based on demand.
  - **Cost Efficiency:** Pay-as-you-go pricing model allows for better cost management.
  - **Global Reach:** GCP provides a global network of data centers to enhance availability and performance.
  - **Advanced Technology:** Access to machine learning, big data analytics, and high-performance computing.
  - **Security:** Robust security measures and compliance certifications.
- Competitors: Amazon Web Services (AWS) and Microsoft Azure.



Fundamentals of GCP

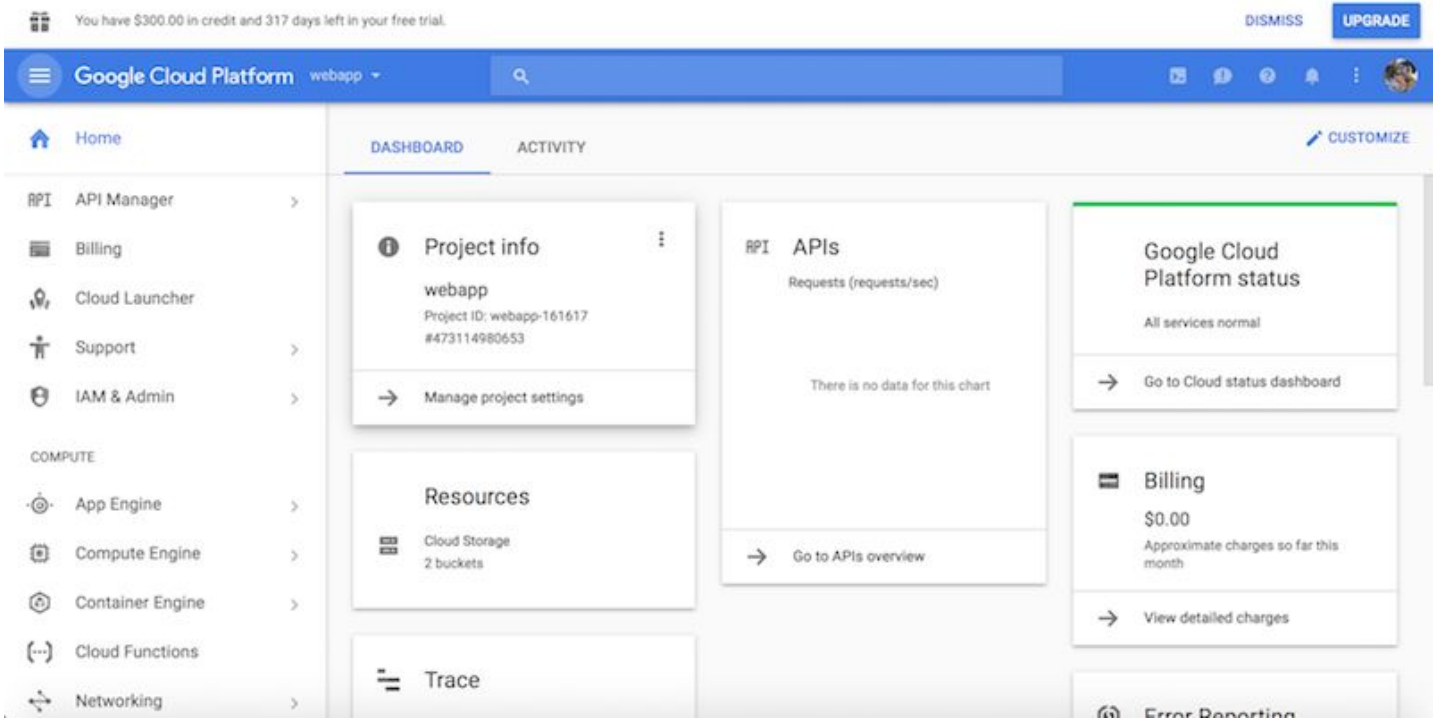
- Brief overview of Key Services offered by GCP:
  - **Compute Services:**
    - **Google Compute Engine:** Virtual machines for running applications.
    - **Google App Engine:** Platform for building and hosting web applications in Google-managed data centers.
    - **Google Kubernetes Engine:** Managed environment for deploying containerized applications.
  - **Storage Services**:
    - **Google Cloud Storage:** Object storage for unstructured data.
    - **Cloud SQL**: Managed relational database service for SQL databases.
    - **Firestore:** NoSQL document database for building web and mobile applications.
  - **Networking Services:**
    - **Virtual Private Cloud (VPC):** Isolated virtual network to host GCP resources.
    - **Cloud Load Balancing:** Distributes traffic across multiple instances to ensure reliability and performance.
  - **Big Data & Machine Learning:**
    - **BigQuery:** Fully-managed data warehouse for analytics.
    - **Cloud Machine Learning Engine:** Build, train, and deploy machine learning models.



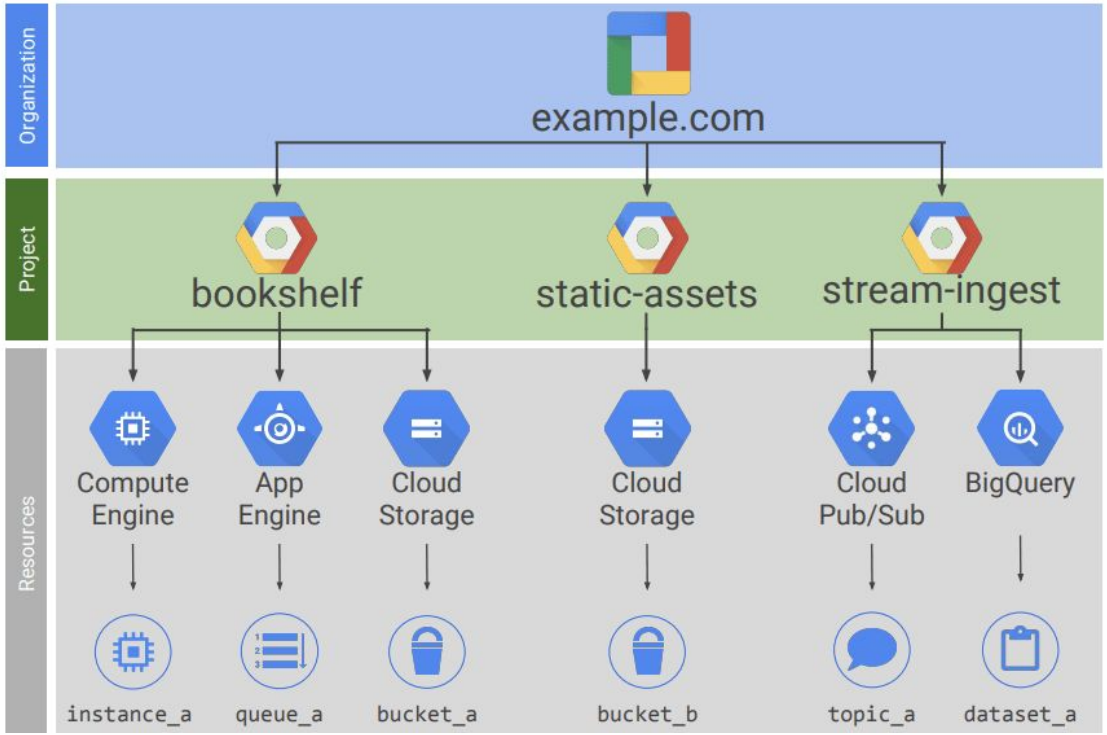
Fundamentals of GCP

- Navigating the GCP Console (https://console.cloud.google.com/):
  - **GCP Interface:**
    - **Overview of the GCP Console:** web-based interface for managing Google CLoud resources and services. It allows users to create, manage, and configure projects, view billing information, and monitor services.
      - **Top Navigation Bar:** Contains the Google Cloud logo, project selection, notifications, and account settings.
      - **Left Navigation Menu:** Provides access to various services (Compute, Storage, Networking, etc.) and quick links to common actions.
      - **Main Workspace**: Displays the selected service or dashboard, where you can interact with the resources and settings.
    - **Dashboard:** centralized location for monitoring your cloud environment and managing your resources effectively.
      - **Project Overview:** Displays key metrics, resource usage, and service status for the selected project.
      - **Quick Access Cards:** Shortcuts to frequently used services and tools, such as Cloud Functions, App Engine, and Compute Engine.
      - **Alerts and Notifications:** Important notifications related to your resources, such as billing alerts or service outages.

Fundamentals of GCP



Fundamentals of GCP



Fundamentals of GCP

- Project configuration options:
  - Creating a project
    - Access the console https://console.cloud.google.com/
    - Select or Create a project
      - Click on the project dropdown in the top navigation bar.
      - Select “New Project.”
    - Enter Project Details:
      - Provide a project name and a billing account (if required).
      - Optionally, select an organization if applicable.
    - Create the Project:
      - Click the “Create” button to finalize the process.
    - Verify Creation:
      - Once created, you’ll be redirected to the new project’s dashboard.
  - Importance of projects in GCP:
    - Resource isolation: Each project acts as an isolated environment, allowing different teams or applications to work independently without resource interference.
    - Management and Organization: Projects help organize resources based on business needs, allowing for easier monitoring and management of resources (e.g., billing, access control).
    - Security and Permissions: Projects serve as boundaries for IAM policies, making it easier to assign roles and permissions at the project level.

Fundamentals of GCP

- How to Enable and Manage APIs for Your Project:
  - Access the APIs & Services:
    - From the left navigation menu, select “APIs & Services.”
  - Enable APIs:
    - Click on “Library.”
    - Search for the desired API (e.g., Cloud Storage, BigQuery).
    - Click on the API, then click “Enable.”
  - Manage APIs:
    - Access the “Dashboard” under “APIs & Services” to view enabled APIs.
    - Monitor usage and quotas.
    - Manage credentials (API keys, OAuth 2.0) by selecting “Credentials”.

ACTIVITY

- Explore the console!

Setting up a project in GCP

- How to Create a Project from the GCP Console:
  - Accessing the GCP Console:
    - Open the Google Cloud Console.
  - Creating a New Project:
    - Click on the project dropdown in the top navigation bar. Select “New Project.”
    - Fill in the required details:
      - Project Name: Choose a descriptive name for your project.
      - Billing Account: If applicable, associate a billing account with the project.
      - Location: (optional) Select an organization or folder to place the project.
    - Click “Create.”
    - Wait for confirmation that the project has been created and select it from the project dropdown.

Setting up a project in GCP

- Setting Up APIs and Authentication
  - Managing API Keys:
    - Accessing APIs & Services:
      - Click on “APIs & Services” in the left navigation menu.
    - Enabling APIs:
      - Select “Library.”
      - Search for the required APIs (e.g., Google Cloud Storage, Compute Engine).
      - Click on the API and select “Enable.”
    - Creating API Keys:
      - Go to “Credentials.”
      - Click on “Create Credentials” and choose “API Key.”
      - Copy the generated API key and configure restrictions if needed (HTTP referrer, IP address, etc.).

Setting up a project in GCP

- Setting Up APIs and Authentication
  - Configuring the Google Cloud SDK in the Local Environment:
    - Installing the Google Cloud SDK:
      - The Google Cloud SDK (Software Development Kit) is a set of command-line tools that enable you to manage Google Cloud resources and services directly from your local terminal or command prompt. The primary component is the `gcloud` command-line tool, which allows you to perform a wide range of tasks, including:
        - Creating and managing projects.
        - Deploying applications.
        - Managing compute instances.
        - Configuring networking and security settings.
      - Other components include `gsutil` for managing Google Cloud Storage and `bq` for BigQuery operations.
      - https://cloud.google.com/sdk/
    - Authenticating with the SDK:
      - Open a terminal (or command prompt).
      - Run `gcloud init` to initialize the SDK and log in to your Google account.
      - Select the project you created from the list during the initialization process.
    - Verifying the Configuration:
      - Use the command `gcloud config list` to display the current configuration and ensure it is set up correctly.

Setting up a project in GCP

- Configuring Billing and Cost Management
  - Accessing Billing Information:
    - Navigating to Billing:
      - From the left navigation menu, select “Billing.”
    - Viewing Billing Reports:
      - Access the billing overview, including spending reports and forecasts.
      - Discuss the importance of understanding billing details and cost trends to manage budgets effectively.
    - Setting Up Budgets and Alerts:
      - Creating a Budget:
        - In the billing section, select “Budgets & Alerts.”
        - Click on “Create Budget.”
        - Specify budget details such as the budget amount, time period, and project.
        - Set up notifications to alert you when spending approaches the budget limit.
      - Importance of Budgets:
        - Budgets help control spending and prevent unexpected charges, especially for projects that may scale up quickly.

Fundamentals of GCP

- Basic permission management:
  - IAM (Identity and Access Management):
    - Definition: IAM is a framework for managing access to GCP resources by defining who (identity) has what access (roles) to which resources.
    - Importance: IAM ensures that only authorized users can access specific resources, providing a secure and controlled environment.
  - Assigning Roles and Permissions:
    - Access IAM:
      - From the left navigation menu, select “IAM & Admin” and then “IAM.”
    - Add a Member:
      - Click on “Grant access” at the top of the IAM page.
      - Enter the email address of the user or service account you want to grant access to.
    - Select a Role:
      - Choose a role from the predefined list (e.g., Viewer, Editor, Owner) or create a custom role.
    - Save Changes:
      - Click “Save” to apply the permissions.

ACTIVITY

- Create a new project!
  - `gcp-course-example`

Deploy a Python app on Google App Engine

- What is Google App Engine?
  - It is a Platform as a Service (PaaS) offering that allows developers to build and host applications in Google-managed data centers.
  - Key Features:
    - **Serverless:** Automatically handles infrastructure management, scaling, and load balancing.
    - **Language Support:** Supports multiple languages, including Python, Java, Node.js, and Go.
    - **Integrated Services:** Offers integration with other GCP services (like Cloud Storage, Firestore, etc.) for building robust applications.

Deploy a Python app on Google App Engine

- Serverless Application Model
  - Developers focus on writing code without managing servers or worrying about scaling.
  - Benefits:
    - Automatic scaling based on traffic.
    - Reduced operational overhead.
    - Cost-efficient, as you pay only for the resources you use.

Deploy a Python app on Google App Engine

Let’s give the correct permission:

- Go to the Google Cloud Console:
  - Visit the Google Cloud Console.
- Navigate to the IAM & Admin Page:
  - From the left sidebar, click on IAM & Admin > IAM.
- Locate Your Service Account:
  - Find the service account named gcp-course-example@appspot.gserviceaccount.com in the list of members. If you don’t see it, look for the App Engine default service account for your project.
- Edit Permissions:
  - Click the pencil icon (edit) next to the service account.
- Add Permissions:
  - Add the Storage Admin role (or at least the Storage Object Admin role) to allow the service account to access Cloud Storage buckets.
    - Click on Add Another Role.
    - In the Role dropdown, select Storage > Storage Admin (or Storage Object Admin).
  - Click Save.

Deploy a Python app on Google App Engine

- Preparing and deploying a Python (Django) application.
  - Project structure: it includes an `app.yaml` file
  - Let's create the project

```
django-admin startproject your_django_project .
pip install gunicorn
```

  - Let's change the settings (update ALLOWED_HOSTS based on our project name)

```
# settings.py
ALLOWED_HOSTS = ['your-project-id.appspot.com']

STATIC_URL = '/static/'
STATIC_ROOT = os.path.join(BASE_DIR, 'static')
```

  - Collect Django app statics for deployment.

```
python manage.py collectstatic
```

Deploy a Python app on Google App Engine

- Preparing and deploying a Python (Django) application.
  - Project structure: it includes an `app.yaml` file

```
my-django-app/
├── app.yaml
├── requirements.txt
├── manage.py
├── your_django_project/
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
└── static/
```

Deploy a Python app on Google App Engine

- Preparing and deploying a Python (Django) application.
  - `app.yaml` configuration: needs to specify the python runtime and the entry point for the Django application.

```
runtime: python312
entrypoint: gunicorn -b :$PORT your_django_project.wsgi

handlers:
- url: /static
  static_dir: static/
- url: /.*
  script: auto
```

Deploy a Python app on Google App Engine

- Preparing and deploying a Python (Django) application.
  - Let's init the console
  - Then, deploy (in the case of an error, review in the Google Cloud Console web App Engine and IAM permissions.
  - Finally, browse the deployed project.

```
gcloud init

gcloud app deploy

# select the created project!

# select the location

gcloud app browse
```

Deploy a Python app on Google App Engine

- Useful parts:
  - Go to the Console and App Engine > Versions to see the versions
  - Go to the Console and Logging to see the logs for debugging or `gcloud app logs tail -s default`
  - `gcloud app versions list`: lists app versions

Deploy a Python app on Google App Engine

- Deleting the created project (if needed)

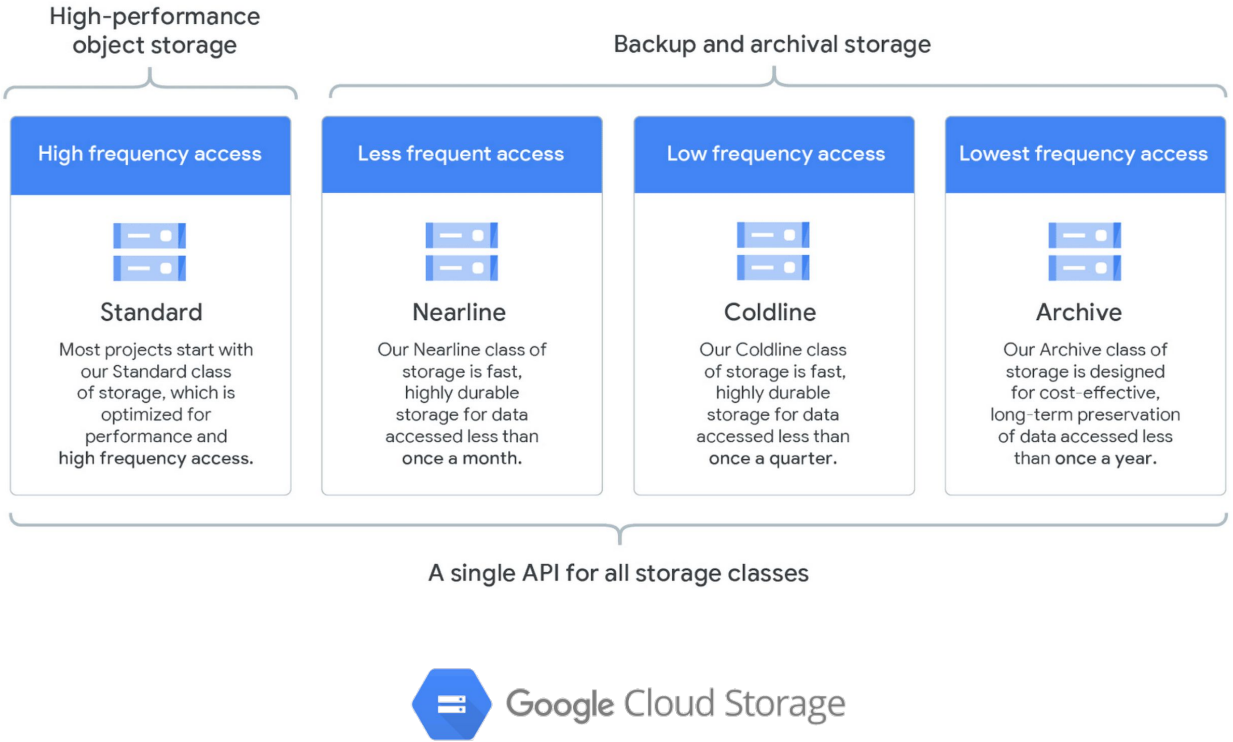
```
gcloud projects delete gcp-course-example

gcloud projects undelete gcp-course-example
```

Data Storage in Google Cloud Storage

- Google Cloud Storage (GCS) is a scalable and secure object storage service offered by Google Cloud Platform (GCP) that allows users to store and retrieve any amount of data at any time from anywhere on the web.
- GCS is designed for high durability, availability, and performance, making it suitable for various use cases, including data backups, archival storage, serving static website content, and supporting big data analytics and machine learning applications.
- Key Features:
  - **Scalability:** GCS can handle an extensive amount of data and traffic without requiring users to provision or manage infrastructure.
  - **Global Accessibility:** Data can be accessed from anywhere, making it ideal for distributed applications.
  - **Security:** GCS provides robust security features, including encryption at rest and in transit, IAM for access control, and audit logging.

Data Storage in google Cloud Storage

- Storage Classes in Google Cloud Storage



Data Storage in google Cloud Storage

- Creating and configuring a storage Bucket:
- Step-by-Step Bucket Creation
  - Creating a Bucket in the Google Cloud Console:
    - Access the Google Cloud Console:
      - Navigate to the Google Cloud Console at console.cloud.google.com.
    - Select Your Project:
      - Choose the project where you want to create the storage bucket (e.g., `gcp-course-example`).
    - Open Cloud Storage:
      - In the left sidebar, click on "Cloud Storage" and then "Overview."
    - Create a Bucket:
      - Click the "Create Bucket" button.
    - Bucket Name:
      - Enter a unique name for the bucket. Note that bucket names must be globally unique across all Google Cloud projects.
    - Choose a Bucket Location:
      - Select either "Region" (data is stored in a specific geographic location) or "Multi-region" (data is stored in multiple locations for redundancy).
        - Regional Storage: Lower latency for users in a specific area, better control over data locality.
        - Multi-regional Storage: Higher availability and redundancy, ideal for serving content globally.
    - Set Storage Class:
      - Choose the default storage class (e.g., Standard, Nearline, Coldline, Archive) based on the intended usage of the bucket.
    - Review and Create:
      - Review the settings and click "Create" to finalize the bucket creation.

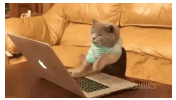
Data Storage in google Cloud Storage

- Bucket configurations:
  - Access control:
    - Uniform Access Control: All objects in the bucket use the same permissions; simplifies access management.
    - Fine-Grained Access Control: Individual objects can have different permissions, offering more flexibility.
  - Lifecycle management:
    - Automatically delete old versions or objects that have not been accessed for a specified time.
    - Transition objects to cheaper storage classes based on their age.
  - Versioning:
    - Helps in recovering from accidental deletions or overwrites.
    - Each version of an object is stored with a unique identifier.

Data Storage in google Cloud Storage

- Setting Permissions
  - IAM Policies for the Bucket:
    - Select the Bucket:
      - Return to "Cloud Storage" and select the bucket created earlier.
    - Open Permissions:
      - Click on the "Permissions" tab for the bucket.
    - Add Members:
      - Click the "Grant access" button to grant access to new users or service accounts.
    - Set Roles:
      - Assign appropriate roles (e.g., Storage Object Viewer, Storage Object Admin) to manage permissions based on the principle of least privilege.
      - Common roles:
        - Storage Admin: Full control over buckets and objects.
        - Storage Object Creator: Allows users to upload objects.
        - Storage Object Viewer: Grants permission to view objects without modifying them.
    - Review Policies: regularly review bucket permissions to ensure they align with access needs and security practices.

Data Storage in google Cloud Storage



- Configure authentication:
  - Using a Service Account Key (For Local Development)
    - Create a Service Account:
      - Go to IAM & Admin > Service Accounts in the Google Cloud Console.
      - Create a service account with Storage Admin or Storage Object Viewer roles, depending on what actions you need.
      - Go to Keys tab > Add Key.
      - Download the credentials as a JSON file.
    - Set the `GOOGLE_APPLICATION_CREDENTIALS` Environment Variable: Point this environment

```
export GOOGLE_APPLICATION_CREDENTIALS="path/to/your/service-account-key.json"
```

Data Storage in google Cloud Storage

- Uploading and Downloading Files:
  - Using the Google Cloud Console
  - Using the Python SDK.

```
pip install google-cloud-storage
```

  - We can upload and download files

```
# upload
blob = bucket.blob(destination_blob_name)
blob.upload_from_filename(source_file_name)

# download
blob = bucket.blob(source_blob_name)
blob.download_to_filename(destination_file_name)
```

ACTIVITY

- Think about how we may include this in our Django application.

Using Additional GCP Services

- Google Cloud Platform offers additional services like Firestore or BigQuery

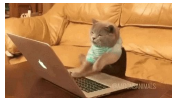
Using Additional GCP Services

- Introduction to Firestore
  - Google Firestore is a NoSQL document database that allows you to store, sync, and query data for your mobile, web, and server development.
  - It organizes data into documents, which are grouped into collections, allowing for flexible and hierarchical data storage.

Using Additional GCP Services

- Creating a Firestore Database:
  - Access Firestore:
    - Navigate to the Google Cloud Console.
    - In the top bar, search "Firestore.
    - Click on "Create Database".
    - Choose between "Start in Production Mode" or "Start in Test Mode". (For development, Test Mode is recommended for easier access.)
    - Select the location for your Firestore database and click "Done".
- Understanding Documents and Collections:
  - Collections: Groups of related documents. For example, a collection of users.
  - Documents: Individual records within a collection. Each document can contain fields (key-value pairs) that hold various types of data.

Using Additional GCP Services



- Configure authentication:
  - Using a Service Account Key (For Local Development)
    - Create a Service Account:
      - Go to IAM & Admin > Service Accounts in the Google Cloud Console.
      - Create a service account with **Cloud Datastore for Firebase Admin** roles.
      - Download the credentials as a JSON file.
    - Set the `GOOGLE_APPLICATION_CREDENTIALS` Environment Variable: Point this environment variable to the downloaded credentials file:

```
export GOOGLE_APPLICATION_CREDENTIALS="path/to/your/service-account-key.json"
```

Using Additional GCP Services

- Connecting a Python Application to Firestore

```
pip install google-cloud-firestore
```

  - Writing data to Firestore

```
from google.cloud import firestore
# Initialize Firestore client

db = firestore.Client()
# Create a new document in the "users" collection
doc_ref = db.collection('users').document('user_1')
doc_ref.set({
    'name': 'Alice',
    'age': 30,
    'email': 'alice@example.com'
})
print('User data written to Firestore.')
```

Using Additional GCP Services

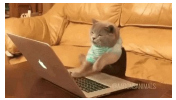
- Connecting a Python Application to Firestore
  - Reading data from Firestore

```
from google.cloud import firestore
# Initialize Firestore client
db = firestore.Client()
# Retrieve the document
user_ref = db.collection('users').document('user_1')
user_data = user_ref.get()
if user_data.exists:
    print(f'User data: {user_data.to_dict()}')
else:
    print('No such user found.')
```

Using Additional GCP Services

- Introduction to BigQuery:
  - Google BigQuery is a fully managed, serverless data warehouse that enables super-fast SQL queries using the processing power of Google's infrastructure.
- Access BigQuery:
  - Navigate to the Google Cloud Console.
  - In the left sidebar, select "BigQuery".
- Reviewing public datasets:
  - https://cloud.google.com/datasets

Using Additional GCP Services



- Configure authentication:
  - Using a Service Account Key (For Local Development)
    - Create a Service Account:
      - Go to IAM & Admin > Service Accounts in the Google Cloud Console.
      - Create a service account with BigQuery Admin, BigQuery Data Viewer, BigQuery Data Editor roles, depending on what actions you need.
      - Download the credentials as a JSON file.
    - Set the `GOOGLE_APPLICATION_CREDENTIALS` Environment Variable: Point this environment variable to the downloaded credentials file:

```
export GOOGLE_APPLICATION_CREDENTIALS="path/to/your/service-account-key.json"
```

Using Additional GCP Services

- Using the BigQuery Client Library

```
pip install google-cloud-bigquery
```

  - Running a Query

```
from google.cloud import bigquery
# Initialize BigQuery client
client = bigquery.Client()
# Define a query
query = """
    SELECT name, age FROM `your-project-id.your_dataset.users`
    WHERE age > 25
"""
# Run the query
query_job = client.query(query)
# Print the results
for row in query_job:
    print(f'Name: {row.name}, Age: {row.age}')
```

ACTIVITY

- Think about how we may include one of these services in our Django application.

Using Additional GCP Services

- Other services:
  - Google Cloud Functions: serverless compute service to run code in response to events.
    - Use Case: Offload specific processes from Django, like processing images, handling webhooks, or sending notifications.
    - Integration: Use Cloud Functions to perform these tasks independently and trigger them based on HTTP requests or events in other GCP services.