# CPC357 ASSIGNMENT 2

# IOT ARCHITECTURE AND SMART APPLICATIONS

SCHOOL OF COMPUTER SCIENCE

UNIVERSITI SAINS MALAYSIA

SEMESTER 1 2024/2025

ASSIGNMENT 2

**Lecturer: Dr. Mohammad Nadhir Ab Wahab**

**Group Members:**

| Name | Matric No |
|---|---|
| BRYON SAVERO A/L MICHAEL LEO | 159351 |
| SUVARNAA MUGUNTHAN A/L RAMAN | 159208 |

**Submission: 26th January 2025**

## Table Of Contents

## 1. Introduction

The **weather monitoring station** is an **Internet of Things (IoT)**-based application designed to measure and report **environmental conditions** in **real time**. This system is built to provide **accurate** and **reliable data** on key environmental parameters, specifically **temperature**, **humidity**, and **rainfall**. Such data is crucial for a wide range of applications, from **agricultural planning** to **urban development**. Leveraging the widely used **DHT11 sensor** for temperature and humidity measurements and a **rain sensor** for detecting precipitation, the station offers a **cost-effective** and **efficient solution** for continuous environmental monitoring.

By integrating these sensors with **Google Cloud Platform (GCP)**, the system achieves several critical objectives: **scalability**, **real-time data processing**, and **secure data storage**. The integration allows the collected data to be transmitted seamlessly from **physical sensors** to the **cloud**, enabling efficient **storage**, **analysis**, and **visualization**. The use of **GCP** ensures the system can handle **increased data volumes** and provide reliable access to the **information** it generates.

This project addresses the growing need for **efficient** and **reliable environmental monitoring solutions**. As the world faces challenges such as **climate change** and **urban expansion**, monitoring systems like this play a critical role in supporting **data-driven decision-making**. For instance, in **agriculture**, accurate weather data helps farmers optimize **planting schedules**, **irrigation**, and **harvests**. In **urban planning**, such systems can support **drainage design** and **flood prevention** measures. Moreover, in **disaster management**, **real-time weather data** can improve **preparedness** and **response strategies**, potentially saving **lives** and **resources**.

The project emphasizes the interplay between **hardware** and **software components**, ensuring **seamless communication** and **robust data handling**. The **hardware layer** collects environmental data from **sensors**, while the **software layer** processes, stores, and analyzes this data using **cloud-based tools**. This **dual-layered approach** ensures that the system operates **efficiently** and **reliably**. The design also focuses on **flexibility** and **scalability**, enabling the addition of **new sensors** or functionalities without disrupting the existing setup. This **forward-looking architecture** ensures the weather monitoring station can adapt to **future technological advancements** and **emerging use cases**.

By focusing on these principles, the **weather monitoring station** exemplifies how **IoT technologies** can bridge the gap between **physical** and **digital systems**. It provides a foundation for building **smarter**, more **sustainable environments**. The system's ability to deliver **real-time**, **actionable insights** makes it a valuable tool across various domains, reinforcing the importance of **environmental monitoring** in today's **data-driven world**.

## 2. High-Level System Architecture Overview

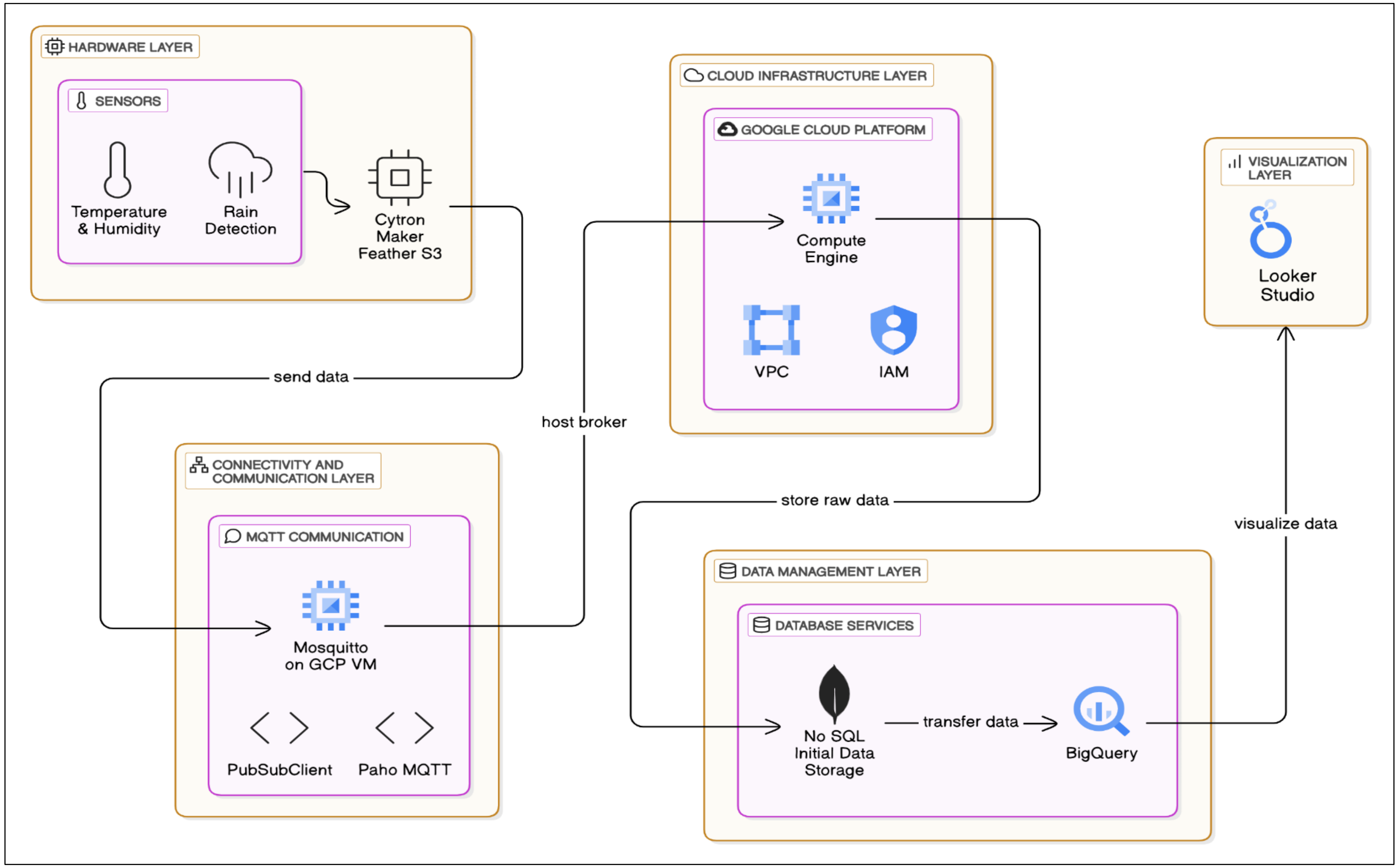


*Figure 1: High-level architecture diagram of the system*

## 2.1 Hardware Layer

The hardware layer forms the foundational component of the weather monitoring system and comprises the physical IoT devices responsible for environmental data collection. At the core of this layer is the **Cytron Maker Feather S3 microcontroller**, which functions as the primary data acquisition and transmission unit. This microcontroller is equipped with two sensors to measure environmental parameters:

- **DHT11 Sensor:** This sensor is responsible for measuring temperature and humidity levels in the environment. It provides digital output and is known for its simplicity and cost-effectiveness, making it suitable for IoT applications.
- **Rain Sensor:** This sensor detects precipitation levels and provides real-time data regarding rainfall. It can determine the presence and intensity of rain, enabling the system to monitor weather conditions effectively.

The microcontroller collects the raw environmental data from these sensors and transmits it to the cloud for further processing and analysis. The selection of reliable and efficient hardware components ensures accurate data collection, forming the basis for the system's functionality.

## 2.2 Connectivity and Communication Layer

The connectivity and communication layer is a critical component responsible for the secure and efficient transmission of sensor data from the IoT devices to the cloud infrastructure. This layer leverages the following key technologies:

- **Mosquitto MQTT Broker:** The Mosquitto MQTT broker is hosted on a Google Cloud Platform (GCP) Compute Engine virtual machine. It serves as the central messaging hub, managing the publication and subscription of data between IoT devices and the server.
- **MQTT Protocol:** The **MQTT (Message Queuing Telemetry Transport)** protocol is used for lightweight and efficient data transmission. This protocol is particularly well-suited for IoT applications due to its low bandwidth requirements. For secure communication, **TLS (Transport Layer Security)** is employed on port **8883**, ensuring data integrity and confidentiality during transmission.
- **PubSubClient Library:** On the microcontroller side, the **PubSubClient library** is utilized to enable MQTT communication. It allows the microcontroller to publish the collected sensor data to the broker.

- **Paho MQTT Library:** On the server side, the **Paho MQTT library** facilitates interaction with the MQTT broker, enabling the reception of incoming sensor data for further processing.

This robust communication framework ensures seamless and secure data transfer between the IoT devices and the cloud infrastructure, forming the backbone of the system's operation.

### 2.3 Cloud Infrastructure Layer

The cloud infrastructure layer is designed to host and manage the essential services required for the weather monitoring solution. It leverages several Google Cloud Platform (GCP) services to ensure scalability, security, and reliability:

- **Compute Engine:** The Mosquitto MQTT broker runs on a **Compute Engine virtual machine**, which provides a high-performance and customizable computing environment.
- **Virtual Private Cloud (VPC):** The **VPC service** is used to configure a secure and isolated network for the system. It manages the network communication and enforces **firewall rules** to protect the MQTT broker and other cloud components from unauthorized access.
- **Identity and Access Management (IAM):** The **IAM service** ensures appropriate access control and permissions for the various system components. A service account has been specifically created for **BigQuery** with the role of **BigQuery Admin**, ensuring proper permissions for managing and querying the data warehouse. This enhances security by restricting access to only authorized users and applications.

The infrastructure layer provides a scalable and secure environment to support the system's requirements by leveraging these cloud services, ensuring uninterrupted data processing and storage.

### 2.4 Data Management Layer

The data management layer focuses on the storage and processing of sensor data collected from the IoT devices. This layer incorporates the following components:

- **MongoDB:** The raw sensor data is initially stored in a **MongoDB database** hosted on the Compute Engine virtual machine. MongoDB, being a NoSQL database, offers flexibility in handling unstructured or semi-structured data. It serves as a temporary storage solution, enabling efficient ingestion of real-time data.
- **BigQuery:** The collected data is subsequently transferred from MongoDB to **BigQuery**, a fully managed, highly scalable data warehouse service provided by GCP. BigQuery enables fast and efficient analysis of large datasets, making it ideal for processing and analysing the high volume of sensor data generated by the system.

This layered data management approach ensures that the system can handle both the real-time ingestion of sensor data and the long-term storage and analysis of large datasets.

## 2.5 Visualization Layer

The visualization layer is responsible for presenting the processed sensor data in an intuitive and insightful manner. This is achieved using Looker Studio, a powerful business intelligence tool offered by GCP:

- **Interactive Dashboards:** Looker Studio is utilized to create interactive dashboards that allow users to visualize and analyze the weather-related data effectively. These dashboards provide a user-friendly interface to explore trends, patterns, and insights derived from the sensor data.
- **Seamless Integration with BigQuery:** Looker Studio integrates seamlessly with BigQuery, enabling real-time access to the processed data stored in the data warehouse. This integration facilitates the creation of dynamic visualizations that update automatically as new data becomes available.

The system provides stakeholders with actionable insights and a comprehensive understanding of weather conditions, enhancing decision-making capabilities by utilizing Looker Studio.

This weather monitoring station architecture demonstrates a comprehensive and well-structured IoT solution that leverages various Google Cloud Platform (GCP) services to collect, store, process, and visualize environmental data. By adopting a layered approach—consisting of distinct hardware, connectivity, cloud infrastructure, data management, and visualization components—the system ensures scalability, security, and flexibility. This architecture can provide valuable weather-related insights, making it a robust and efficient solution for monitoring environmental conditions.

# 3. Design Considerations

## 3.1 Factors Influencing Design Decisions

### 1. Sensor Selection

- The **DHT11 sensor** was chosen for its **affordability**, **simplicity**, and **adequate precision** for measuring temperature and humidity. It provides **digital output**, making it ideal for IoT applications where low-cost and reliable data collection is prioritized. While the DHT11 may lack the accuracy of advanced sensors like the DHT22, it offers sufficient performance for general weather monitoring tasks, ensuring a cost-effective solution.
- The **rain sensor** was selected for its ability to provide both **digital** and **analog outputs**, enabling it to detect the **presence** and **intensity** of rainfall. While it may not offer the precision of advanced rain gauges, its versatility and cost-effectiveness make it suitable for the system's real-time monitoring requirements.

### 2. Hardware Selection

- The **Cytron Maker Feather S3 microcontroller** was chosen due to its **Wi-Fi** and **Bluetooth capabilities**, which facilitate seamless connectivity with cloud services. Its **compatibility with MQTT** protocols and support for lightweight data processing ensures efficient operation. Compared to less advanced microcontrollers, this choice balances performance and energy efficiency, making it ideal for IoT applications like this weather monitoring station.

### 3. Scalability

- The system's architecture supports **future expansion**, allowing for the integration of additional sensors, such as **wind speed detectors** or **air quality monitors**, without requiring significant modifications. The use of **Google Cloud Platform (GCP)** services, such as **BigQuery** and **Looker Studio**, ensures the system can handle **large data volumes** and **dynamic visualizations**. This scalability makes the design flexible for future requirements or changing application needs.

### 4. Cost-Effectiveness

- The integration of **low-cost sensors** like the DHT11 and rain sensor, combined with GCP's **pay-as-you-go model**, ensures the system remains affordable for a wide range of users. While components like MongoDB offer efficient real-time data ingestion, and BigQuery handles large-scale analytics, the system avoids unnecessary expenditure by leveraging managed cloud services and cost-effective hardware. While more advanced sensors and cloud features might improve performance, the chosen components and configurations provide a **balanced approach** to maintaining affordability.

**5. Data Security**

- The system implements several **security measures** to protect data integrity and confidentiality. **Transport Layer Security (TLS)** is used for MQTT communication, ensuring secure data transmission. The **Virtual Private Cloud (VPC)** enforces **firewall rules**, while **Identity and Access Management (IAM)** provides **role-based access control (RBAC)**, restricting access to only authorized users and applications. These measures ensure compliance with IoT security best practices, even though they introduce additional setup complexity.

**6. Connectivity and Communication**

- The use of **Mosquitto MQTT Broker** on a **GCP Compute Engine VM** provides a reliable and centralized messaging hub. The lightweight **MQTT protocol**, combined with libraries like **PubSubClient** (microcontroller-side) and **Paho MQTT** (server-side), ensures **efficient communication** between hardware and cloud infrastructure. This robust framework minimizes latency and bandwidth usage, which are critical for IoT systems.

## 3.2 Trade-Offs

**Cost vs. Performance**

- A deliberate decision was made to use **cost-effective sensors** like the DHT11 and rain sensor, even though they may lack the precision of more expensive alternatives. For instance, while the DHT11 is less accurate than the DHT22, it provides sufficient data accuracy for **general monitoring purposes**. Similarly, the rain sensor offers basic precipitation detection but cannot provide **quantitative rainfall measurements** like advanced rain gauges. These trade-offs were necessary to achieve the project's objective of creating a **budget-friendly solution** without compromising the system's primary functionality.

**Ease of Use vs. Security**

- Implementing **advanced security configurations**, such as **strict firewall rules** and **encrypted MQTT communication**, added complexity to the system's deployment and maintenance. These measures were essential for ensuring **robust data protection**, but they required additional setup and expertise, potentially increasing the learning curve for users. The decision to prioritize security over ease of use reflects the importance of **data integrity** and **user trust** in IoT applications. This trade-off ensures that the system remains **secure and reliable**, even if it slightly reduces accessibility for non-technical users.

**Real-Time Data vs. Long-Term Analysis**

- The architecture prioritizes real-time data ingestion using **MongoDB** while leveraging **BigQuery** for long-term storage and analytics. While this layered approach ensures efficiency and scalability, it introduces an additional step in the data pipeline, slightly increasing system complexity. This trade-off is justified by the system's ability to handle both **real-time operations** and **historical data analysis** effectively.

**Flexibility vs. Specialization**

- The decision to use general-purpose components like the **Maker Feather S3** and **rain sensor** emphasizes flexibility but sacrifices some precision and specialization. For example, using a **dedicated weather station kit** could simplify integration but reduce the ability to customize the system for unique use cases.

## 4. Development Process

### 4.1 Setting Up Google Cloud Platform Services

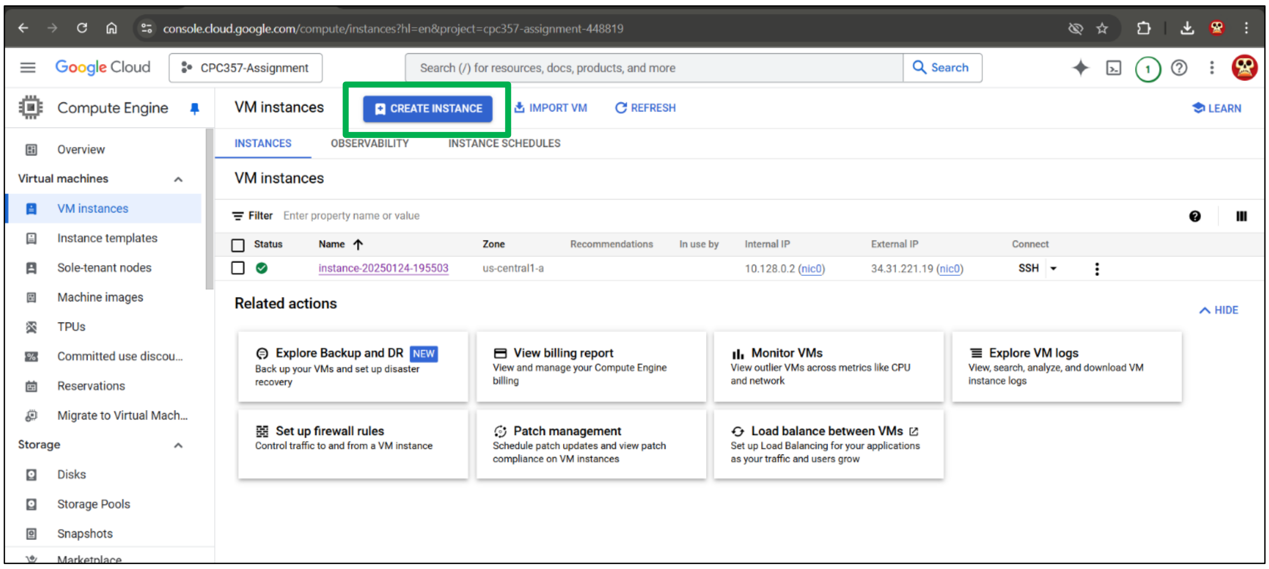
Services that are used for the development:

- Compute Engine (VM Instances)
- VPC Network (Firewall Rule)
- IAM & Admin (Service Accounts)
- BigQuery
- Looker Studio

**Compute Engine**

Compute Engine is a customizable compute service that allows you to create and run virtual machines on Google Cloud. We can opt in for pre-built and ready-to-go configurations of VMs for a quick start, or can go for custom machine types and create VMs with the optimal amount of processing power and memory based on our needs. Go to this link to navigate to the Google Cloud Console. If you are using Compute Engine for the first time, you need to complete a few additional steps before you can proceed with the rest of the session. Please create a project in the console and enable the Compute Engine API. Note that you also need to set up a billing account in order to use the service.

Next, go to the navigation menu and select ***Compute Engine > Virtual Machines > VM Instances***. Click the ***Create Instance*** button to create a new virtual machine.



| Status | Name | Zone | Recommendations | In use by | Internal IP | External IP | Connect |
|---|---|---|---|---|---|---|---|
| ✓ | instance-20250124-195503 | us-central1-a | | | 10.128.0.2 (nic0) | 34.31.221.19 (nic0) | SSH |

Explore Backup and DR NEW — Back up your VMs and set up disaster recovery

View billing report — View and manage your Compute Engine billing

Monitor VMs — View outlier VMs across metrics like CPU and network

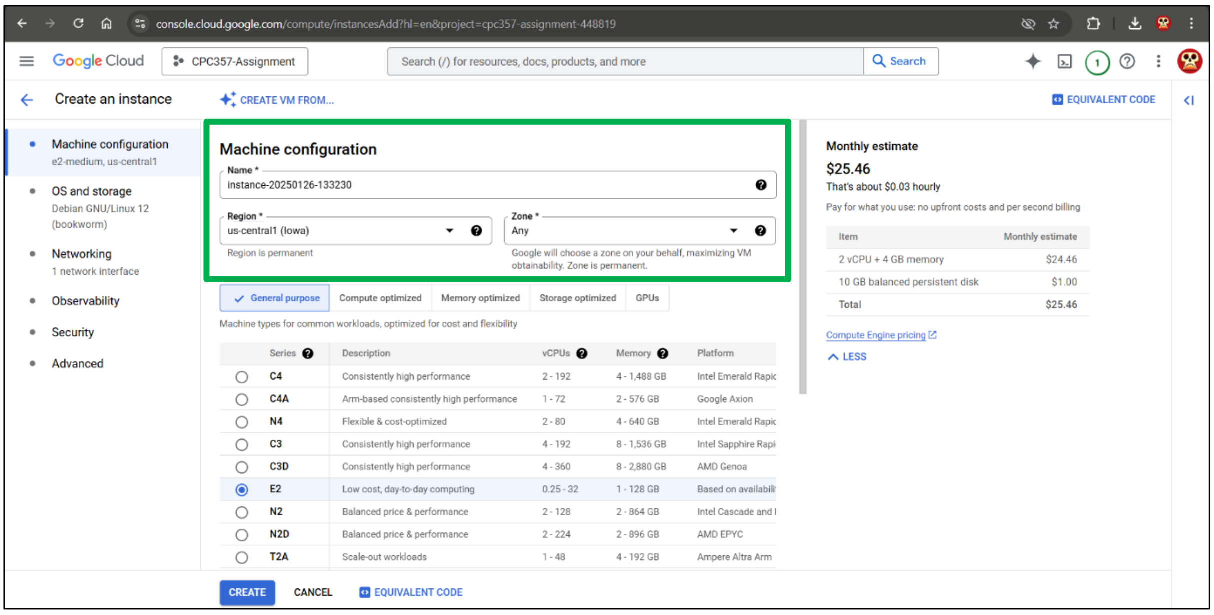
Explore VM logs — View, search, analyze, and download VM instance logs

Set up firewall rules — Control traffic to and from a VM instance

Patch management — Schedule patch updates and view patch compliance on VM instances

Load balance between VMs — Set up Load Balancing for your applications as your traffic and users grow

Under ***Machine configuration***, you can set the region and zone that runs your virtual machine. Next, go to ***OS and storage*** in the navigation menu to choose the operating system. Click the Change button and select ***Ubuntu*** for the operating system and ***Ubuntu 20.04 LTS*** for the version.



console.cloud.google.com/compute/instancesAdd?hl=en&project=cpc357-assignment-448819

Google Cloud · CPC357-Assignment · Search (/) for resources, docs, products, and more · Search

Create an instance · CREATE VM FROM... · EQUIVALENT CODE

- Machine configuration
  e2-medium, us-central1
- OS and storage
  Debian GNU/Linux 12 (bookworm)
- Networking
  1 network interface
- Observability
- Security
- Advanced

**Machine configuration**

Name *: instance-20250126-133230

Region *: us-central1 (Iowa) — Region is permanent

Zone *: Any — Google will choose a zone on your behalf, maximizing VM obtainability. Zone is permanent.

General purpose | Compute optimized | Memory optimized | Storage optimized | GPUs

Machine types for common workloads, optimized for cost and flexibility

| | Series | Description | vCPUs | Memory | Platform |
|---|---|---|---|---|---|
| ○ | C4 | Consistently high performance | 2 - 192 | 4 - 1,488 GB | Intel Emerald Rapic |
| ○ | C4A | Arm-based consistently high performance | 1 - 72 | 2 - 576 GB | Google Axion |
| ○ | N4 | Flexible & cost-optimized | 2 - 80 | 4 - 640 GB | Intel Emerald Rapic |
| ○ | C3 | Consistently high performance | 4 - 192 | 8 - 1,536 GB | Intel Sapphire Rapi |
| ○ | C3D | Consistently high performance | 4 - 360 | 8 - 2,880 GB | AMD Genoa |
| ● | E2 | Low cost, day-to-day computing | 0.25 - 32 | 1 - 128 GB | Based on availabili |
| ○ | N2 | Balanced price & performance | 2 - 128 | 2 - 864 GB | Intel Cascade and I |
| ○ | N2D | Balanced price & performance | 2 - 224 | 2 - 896 GB | AMD EPYC |
| ○ | T2A | Scale-out workloads | 1 - 48 | 4 - 192 GB | Ampere Altra Arm |

**Monthly estimate**

$25.46

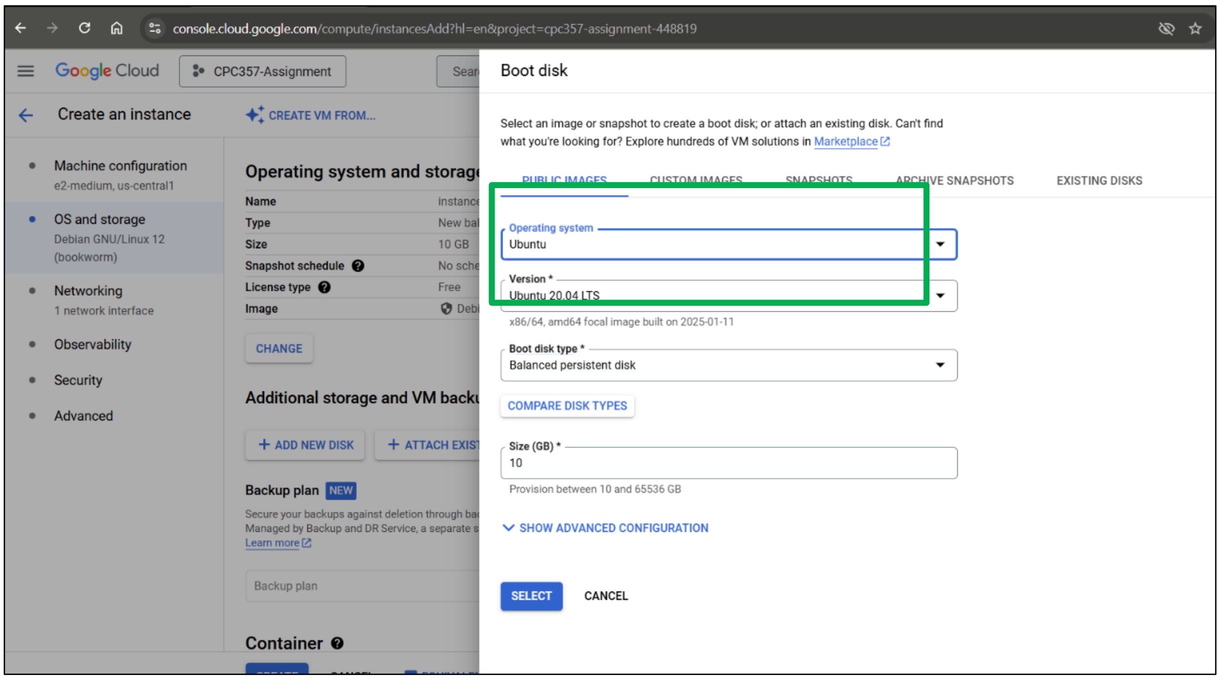
That's about $0.03 hourly

Pay for what you use: no upfront costs and per second billing

| Item | Monthly estimate |
|---|---|
| 2 vCPU + 4 GB memory | $24.46 |
| 10 GB balanced persistent disk | $1.00 |
| Total | $25.46 |

Compute Engine pricing · LESS

CREATE · CANCEL · EQUIVALENT CODE



console.cloud.google.com/compute/instancesAdd?hl=en&project=cpc357-assignment-448819

Google Cloud · CPC357-Assignment

Create an instance · CREATE VM FROM...

- Machine configuration
  e2-medium, us-central1
- OS and storage
  Debian GNU/Linux 12 (bookworm)
- Networking
  1 network interface
- Observability
- Security
- Advanced

**Operating system and storag**

| | |
|---|---|
| Name | instanc |
| Type | New bal |
| Size | 10 GB |
| Snapshot schedule | No sche |
| License type | Free |
| Image | Debi |

CHANGE

**Additional storage and VM backu**

ADD NEW DISK · ATTACH EXIS

Backup plan NEW

Secure your backups against deletion through ba
Managed by Backup and DR Service, a separate s
Learn more

Backup plan

Container

**Boot disk**

Select an image or snapshot to create a boot disk; or attach an existing disk. Can't find what you're looking for? Explore hundreds of VM solutions in Marketplace

PUBLIC IMAGES | CUSTOM IMAGES | SNAPSHOTS | ARCHIVE SNAPSHOTS | EXISTING DISKS

Operating system: Ubuntu

Version *: Ubuntu 20.04 LTS

x86/64, amd64 focal image built on 2025-01-11

Boot disk type *: Balanced persistent disk

COMPARE DISK TYPES

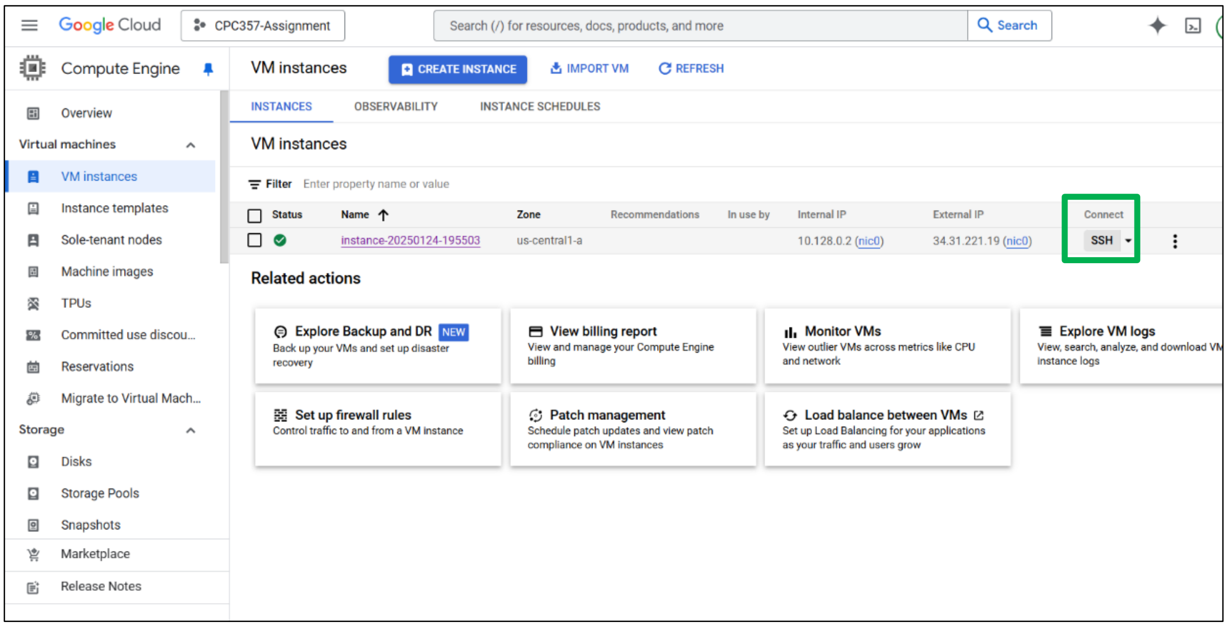
Size (GB) *: 10

Provision between 10 and 65536 GB

SHOW ADVANCED CONFIGURATION

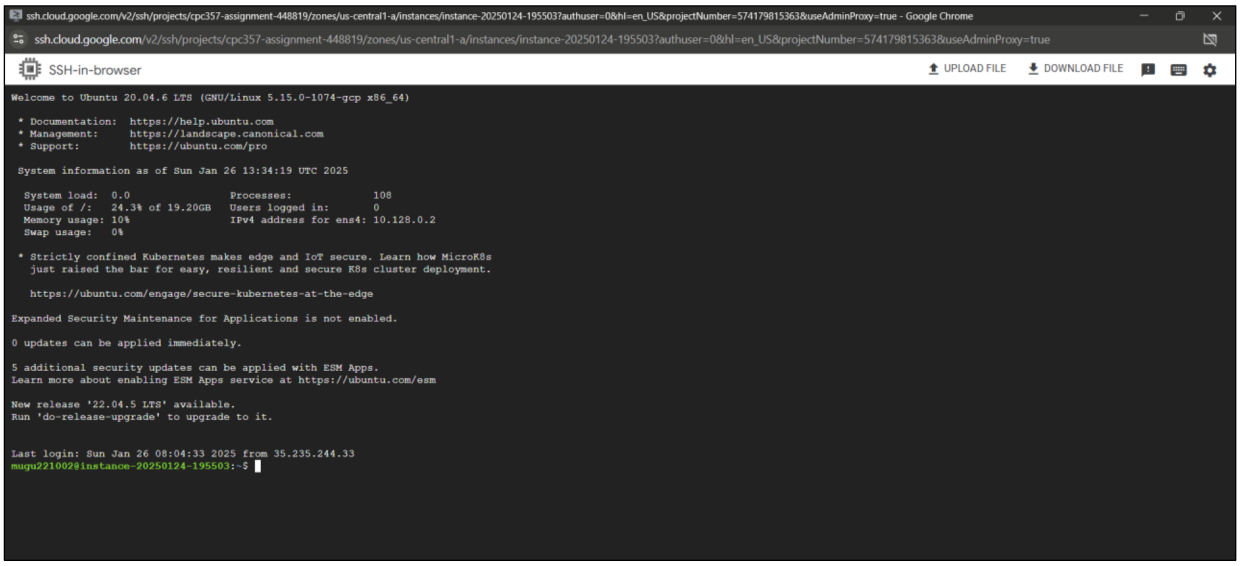
SELECT · CANCEL

Click the ***Create*** button to finish up the instance creation. Upon completion, you should be able to find your newly created VM instance in the list. Next, click the ***SSH*** button next to the VM instance to open up the SSH-in-browser.


Google Cloud
CPC357-Assignment
Search (/) for resources, docs, products, and more
Search
Compute Engine
VM instances
CREATE INSTANCE
IMPORT VM
REFRESH
INSTANCES
OBSERVABILITY
INSTANCE SCHEDULES
Overview
Virtual machines
VM instances
Instance templates
Sole-tenant nodes
Machine images
TPUs
Committed use discou...
Reservations
Migrate to Virtual Mach...
Storage
Disks
Storage Pools
Snapshots
Marketplace
Release Notes
Filter
Enter property name or value
Related actions
Explore Backup and DR
NEW
Back up your VMs and set up disaster recovery
View billing report
View and manage your Compute Engine billing
Monitor VMs
View outlier VMs across metrics like CPU and network
Explore VM logs
View, search, analyze, and download VM instance logs
Set up firewall rules
Control traffic to and from a VM instance
Patch management
Schedule patch updates and view patch compliance on VM instances
Load balance between VMs
Set up Load Balancing for your applications as your traffic and users grow

| Status | Name | Zone | Recommendations | In use by | Internal IP | External IP | Connect |
|---|---|---|---|---|---|---|---|
| ✓ | instance-20250124-195503 | us-central1-a | | | 10.128.0.2 (nic0) | 34.31.221.19 (nic0) | SSH |

A pop-up window will appear and click the ***Authorize*** button to give permission for the SSH-in-browser to connect to the VM.


ssh.cloud.google.com/v2/ssh/projects/cpc357-assignment-448819/zones/us-central1-a/instances/instance-20250124-195503?authuser=0&hl=en_US&projectNumber=574179815363&useAdminProxy=true - Google Chrome
ssh.cloud.google.com/v2/ssh/projects/cpc357-assignment-448819/zones/us-central1-a/instances/instance-20250124-195503?authuser=0&hl=en_US&projectNumber=574179815363&useAdminProxy=true
SSH-in-browser
UPLOAD FILE
DOWNLOAD FILE

```
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1074-gcp x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sun Jan 26 13:34:19 UTC 2025

  System load:  0.0               Processes:             108
  Usage of /:   24.3% of 19.20GB   Users logged in:       0
  Memory usage: 10%               IPv4 address for ens4: 10.128.0.2
  Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

5 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

New release '22.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Sun Jan 26 08:04:33 2025 from 35.235.244.33
mugu221002@instance-20250124-195503:~$
```

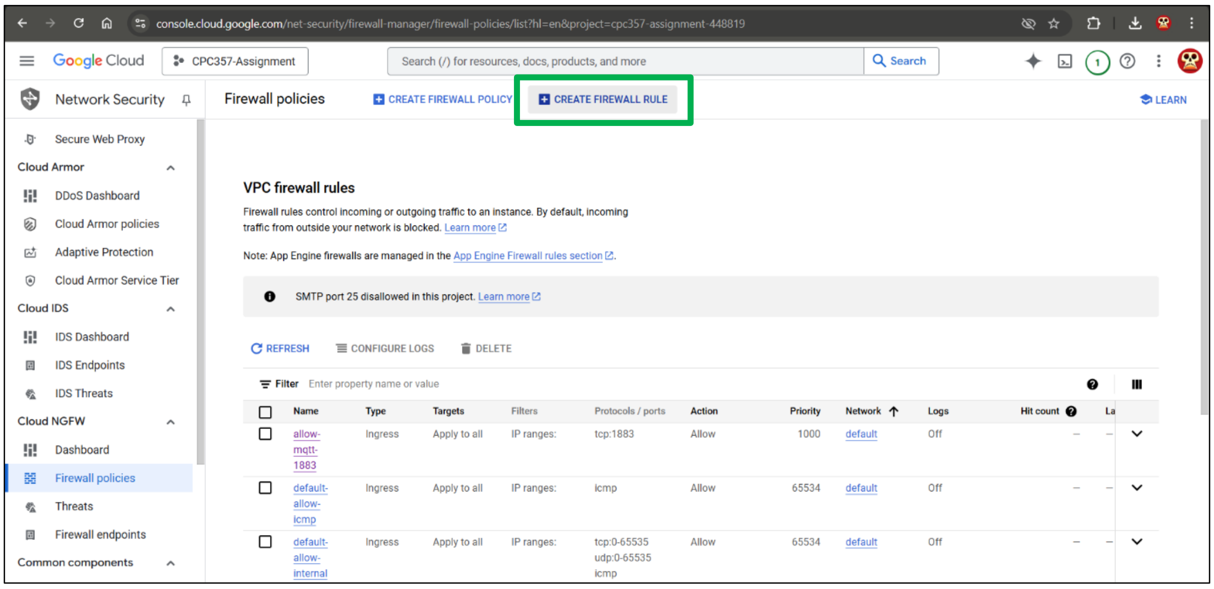
After that, run the following commands in the SSH-in-browser to setup the VM:

```
$ sudo apt-get update
```

```
$ sudo apt-get upgrade
```

## Firewall Rule

To provide connectivity for your Compute Engine VM instances, you need to configure the Virtual Private Cloud (VPC) network and set up the VPC firewall rules accordingly. To put it simply, VPC firewall rules let you allow or deny traffic to or from VM instances in a VPC network based on port number, tag, or protocol. First, go to the navigation menu and select ***VPC Network* > *Firewall***. Then, click the ***Create Firewall Rule*** button at the top.



VPC firewall rules

Firewall rules control incoming or outgoing traffic to an instance. By default, incoming traffic from outside your network is blocked. Learn more

Note: App Engine firewalls are managed in the App Engine Firewall rules section.

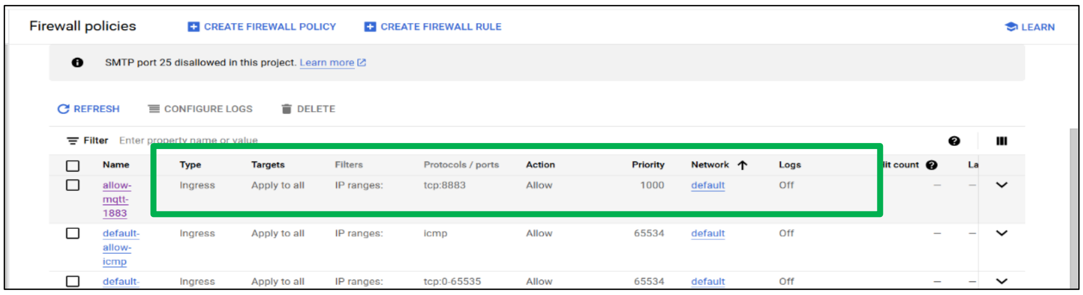
SMTP port 25 disallowed in this project. Learn more

| Name | Type | Targets | Filters | Protocols / ports | Action | Priority | Network | Logs | Hit count | La |
|---|---|---|---|---|---|---|---|---|---|---|
| allow-mqtt-1883 | Ingress | Apply to all | IP ranges: | tcp:1883 | Allow | 1000 | default | Off | – | – |
| default-allow-icmp | Ingress | Apply to all | IP ranges: | icmp | Allow | 65534 | default | Off | – | – |
| default-allow-internal | Ingress | Apply to all | IP ranges: | tcp:0-65535 udp:0-65535 icmp | Allow | 65534 | default | Off | – | – |

Configure the VPC firewall rule according to the specifications below:

- Name: allow-mqtt-8883
- Description: Allow traffic on port 8883
- Logs: off
- Network: default
- Priority: 1000
- Direction of traffic: Ingress
- Action on match: Allow
- Targets: All instances in the network
- Source filter: IPv4 ranges
- Source IPv4 ranges: 0.0.0.0/0
- Second source filter: None
- Destination filter: None
- Protocols and ports: Specified protocols and ports (TCP Port 8883)

Once you're done, click on the Create button to save and create the VPC firewall rule and upon completion, the newly created VPC firewall rule will be displayed in the list.

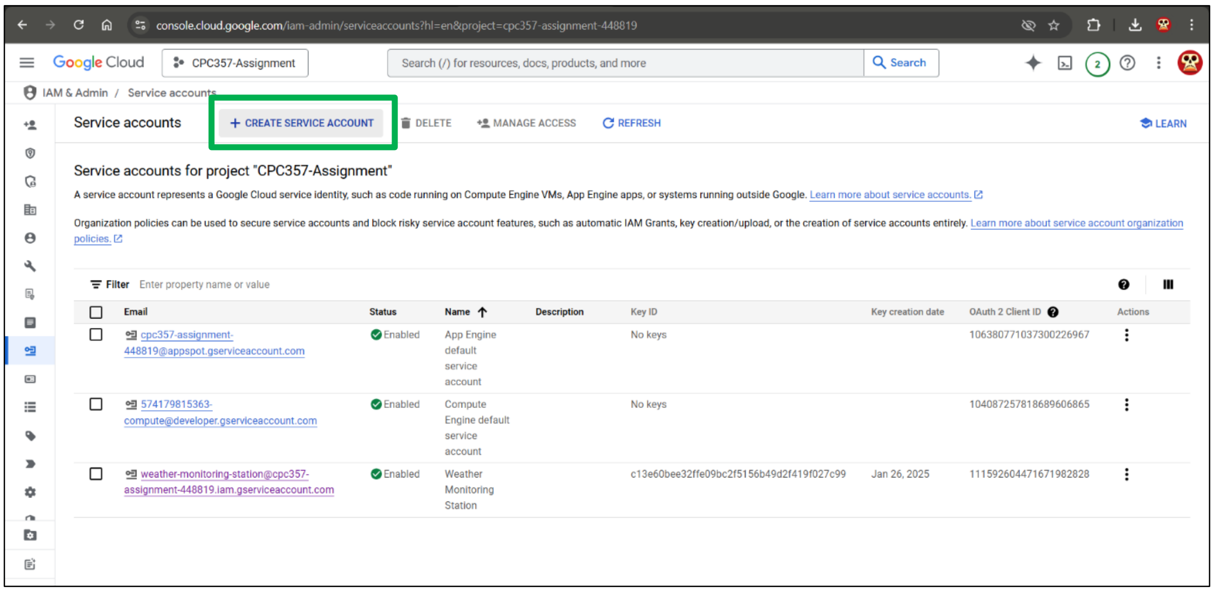


SMTP port 25 disallowed in this project. Learn more

| Name | Type | Targets | Filters | Protocols / ports | Action | Priority | Network | Logs | it count | La |
|---|---|---|---|---|---|---|---|---|---|---|
| allow-mqtt-1883 | Ingress | Apply to all | IP ranges: | tcp:8883 | Allow | 1000 | default | Off | — | — |
| default-allow-icmp | Ingress | Apply to all | IP ranges: | icmp | Allow | 65534 | default | Off | — | — |
| default- | Ingress | Apply to all | IP ranges: | tcp:0-65535 | Allow | 65534 | default | Off | — | — |

## Service Account

A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outside Google. Organization policies can be used to secure service accounts and block risky service account features, such as automatic IAM Grants, key creation/upload, or the creation of service accounts entirely. Go to the navigation menu and select ***IAM & Admin* > *Service Accounts***. Then, click the ***Create Service Account*** button at the top.


console.cloud.google.com/iam-admin/serviceaccounts?hl=en&project=cpc357-assignment-448819
Google Cloud
CPC357-Assignment
Search (/) for resources, docs, products, and more
Search
IAM & Admin / Service accounts
Service accounts
CREATE SERVICE ACCOUNT
DELETE
MANAGE ACCESS
REFRESH
LEARN
Filter Enter property name or value

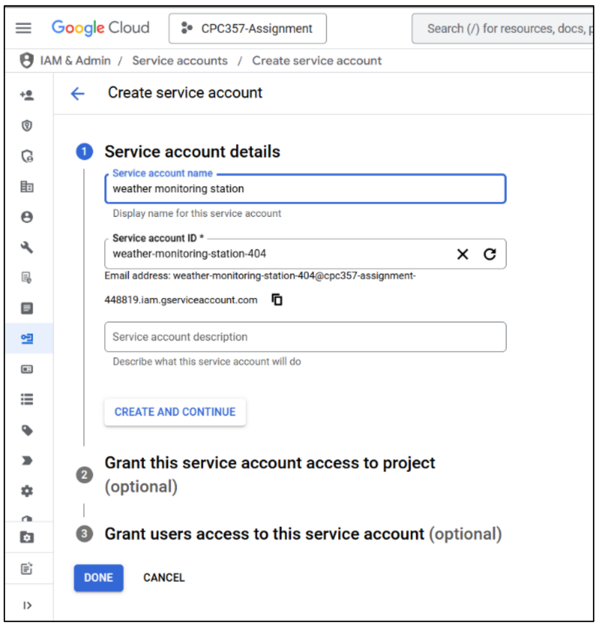
Service accounts for project "CPC357-Assignment"

A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outside Google. Learn more about service accounts.

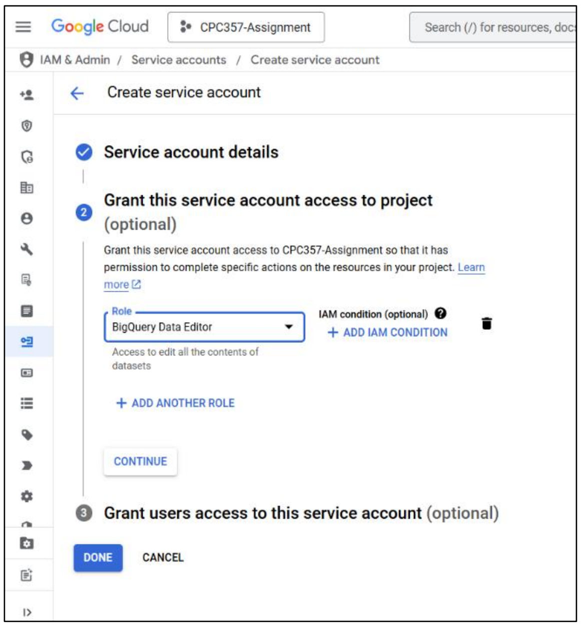
Organization policies can be used to secure service accounts and block risky service account features, such as automatic IAM Grants, key creation/upload, or the creation of service accounts entirely. Learn more about service account organization policies.

| Email | Status | Name | Description | Key ID | Key creation date | OAuth 2 Client ID | Actions |
|---|---|---|---|---|---|---|---|
| cpc357-assignment-448819@appspot.gserviceaccount.com | Enabled | App Engine default service account | | No keys | | 106380771037300226967 | ⋮ |
| 574179815363-compute@developer.gserviceaccount.com | Enabled | Compute Engine default service account | | No keys | | 104087257818689606865 | ⋮ |
| weather-monitoring-station@cpc357-assignment-448819.iam.gserviceaccount.com | Enabled | Weather Monitoring Station | | c13e60bee32ffe09bc2f5156b49d2f419f027c99 | Jan 26, 2025 | 111592604471671982828 | ⋮ |

Under ***Service account details***, give your account a name and click ***Create and Continue***. Then, grant a role to your service account which is ***BigQuery Data Editor*** (access to edit all the contents of datasets. After that, create the service account by clicking ***Done***.

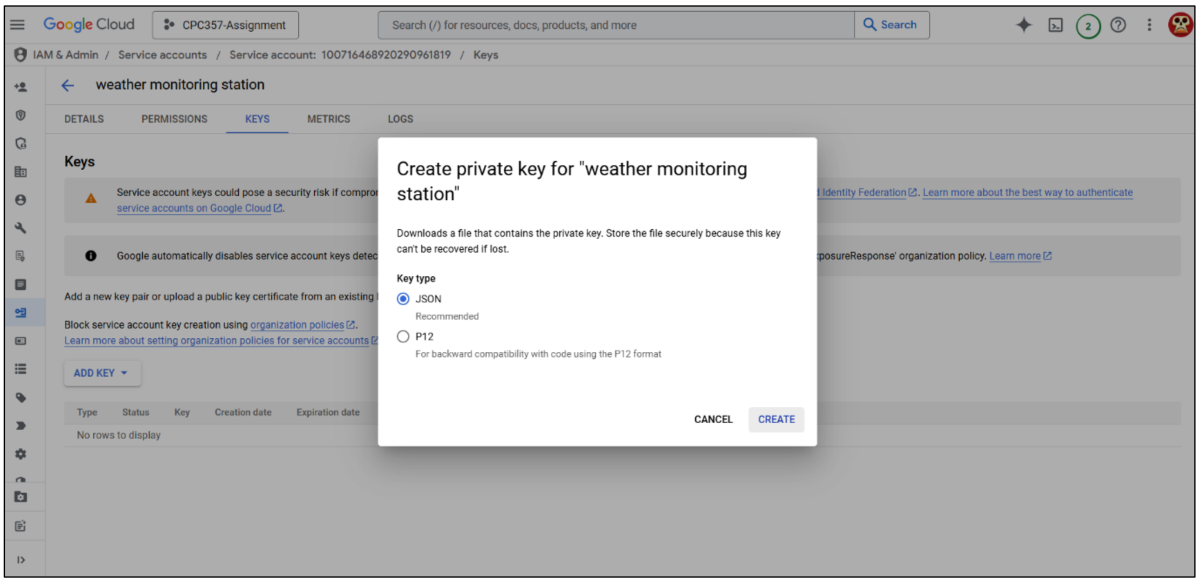

Google Cloud
CPC357-Assignment
Search (/) for resources, docs,
IAM & Admin / Service accounts / Create service account
Create service account
1 Service account details
CREATE AND CONTINUE
2 Grant this service account access to project (optional)
3 Grant users access to this service account (optional)
DONE
CANCEL

Service account name: weather monitoring station
Display name for this service account
Service account ID *: weather-monitoring-station-404
Email address: weather-monitoring-station-404@cpc357-assignment-448819.iam.gserviceaccount.com
Service account description
Describe what this service account will do


Google Cloud
CPC357-Assignment
Search (/) for resources, doc
IAM & Admin / Service accounts / Create service account
Create service account
Service account details
2 Grant this service account access to project (optional)
IAM condition (optional)
+ ADD IAM CONDITION
+ ADD ANOTHER ROLE
CONTINUE
3 Grant users access to this service account (optional)
DONE
CANCEL

Grant this service account access to CPC357-Assignment so that it has permission to complete specific actions on the resources in your project. Learn more
Role: BigQuery Data Editor
Access to edit all the contents of datasets

After creating the service account, the account will be displayed in the list. Click the account and navigate to ***Keys*** and click ***Add Key > Create New Key***. Then, in the pop-up window, choose ***JSON*** as the key type and click ***Create***. After clicking, a JSON file containing the service account's key will be downloaded automatically. Keep the file safe because we will use that key in our python script later.

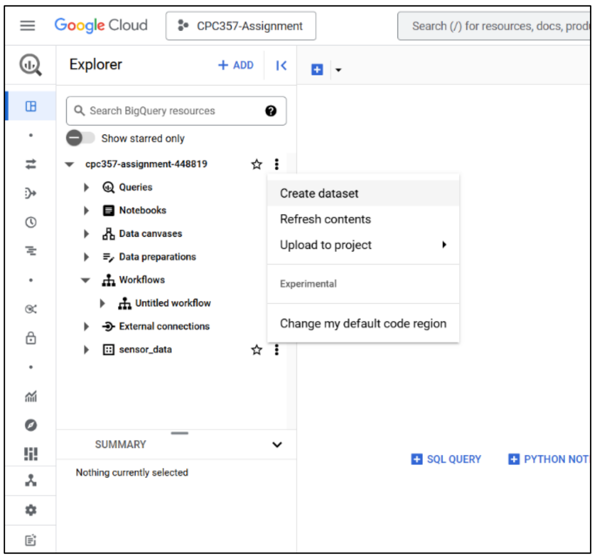

Google Cloud
CPC357-Assignment
Search (/) for resources, docs, products, and more
Search
IAM & Admin / Service accounts / Service account: 100716468920290961819 / Keys
weather monitoring station
DETAILS
PERMISSIONS
KEYS
METRICS
LOGS
Keys
ADD KEY
Type
Status
Key
Creation date
Expiration date
No rows to display
CANCEL
CREATE

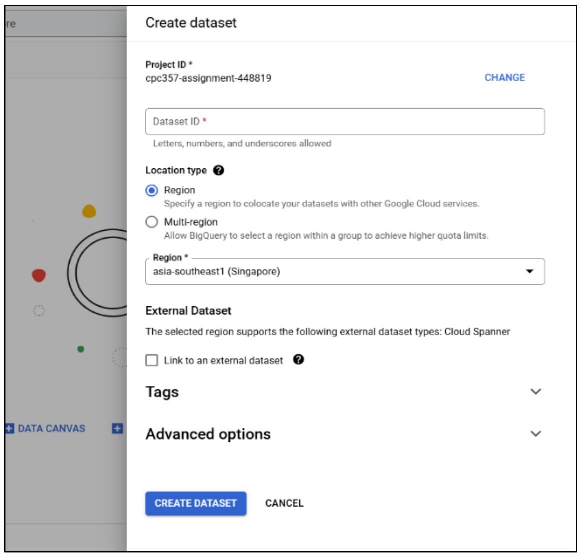
Service account keys could pose a security risk if compro… Identity Federation. Learn more about the best way to authenticate service accounts on Google Cloud.
Google automatically disables service account keys detec… posureResponse' organization policy. Learn more
Add a new key pair or upload a public key certificate from an existing
Block service account key creation using organization policies.
Learn more about setting organization policies for service accounts

Create private key for "weather monitoring station"
Downloads a file that contains the private key. Store the file securely because this key can't be recovered if lost.
Key type
[✓ JSON] Recommended
P12 For backward compatibility with code using the P12 format

## BigQuery

**BigQuery** is a serverless, cost-effective, and multicloud data warehouse designed to help you turn big data into valuable business insights. Navigate to BigQuery and click ***Create dataset*** under your project-id. Then, give enter your ***Dataset ID*** and specify the location type that you want to use to store your data. After that, click ***Create***.





Project ID *
cpc357-assignment-448819

Dataset ID *
Letters, numbers, and underscores allowed

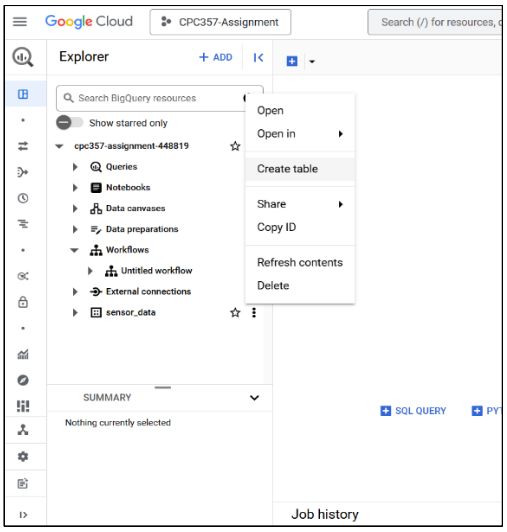
Location type
[✓ Region] Specify a region to colocate your datasets with other Google Cloud services.
Multi-region Allow BigQuery to select a region within a group to achieve higher quota limits.

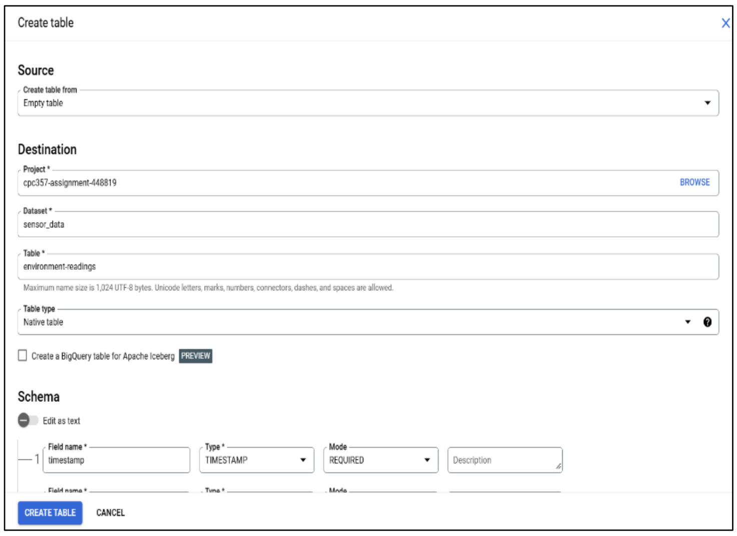
Region *
asia-southeast1 (Singapore)

External Dataset
The selected region supports the following external dataset types: Cloud Spanner
Link to an external dataset

Upon completion, ***Create table*** under the newly created dataset and give table a name. After that, create four fields as below table under the ***Schema*** section:

| **Field name** | **Type** | **Mode** |
|---|---|---|
| timestamp | TIMESTAMP | REQUIRED |
| temperature | FLOAT | NULLABLE |
| humidity | FLOAT | NULLABLE |
| rain | INTEGER | NULLABLE |




Create table

Source

Create table from: Empty table

Destination

Project *: cpc357-assignment-448819

Dataset *: sensor_data

Table *: environment-readings

Maximum name size is 1,024 UTF-8 bytes. Unicode letters, marks, numbers, connectors, dashes, and spaces are allowed.

Table type: Native table

Create a BigQuery table for Apache Iceberg PREVIEW

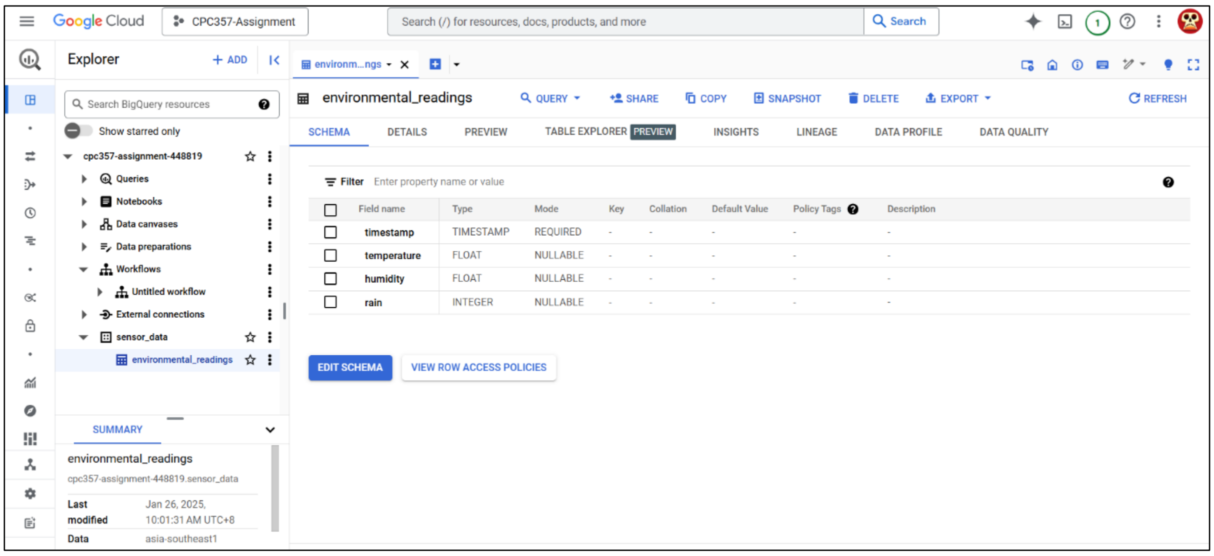
Schema

Edit as text

| | Field name * | Type * | Mode | Description |
|---|---|---|---|---|
| 1 | timestamp | TIMESTAMP | REQUIRED | |

CREATE TABLE CANCEL

Then click ***Create Table.*** Now you can see the created table with the column names and their types.


environmental_readings

SCHEMA DETAILS PREVIEW TABLE EXPLORER PREVIEW INSIGHTS LINEAGE DATA PROFILE DATA QUALITY

| Field name | Type | Mode | Key | Collation | Default Value | Policy Tags | Description |
|---|---|---|---|---|---|---|---|
| timestamp | TIMESTAMP | REQUIRED | - | - | - | - | - |
| temperature | FLOAT | NULLABLE | - | - | - | - | - |
| humidity | FLOAT | NULLABLE | - | - | - | - | - |
| rain | INTEGER | NULLABLE | - | - | - | - | - |

EDIT SCHEMA VIEW ROW ACCESS POLICIES

SUMMARY

environmental_readings

cpc357-assignment-448819.sensor_data

Last modified: Jan 26, 2025, 10:01:31 AM UTC+8

Data: asia-southeast1

## 4.2 Setting Up Hardware

### Arduino Code

Copy the code from the GitHub Repo: https://github.com/mugunthantypical/CPC357-Assignment-2/blob/main/arduino-code/arduino-code.ino and paste it in your Arduino IDE. Before uploading into your microcontroller, make sure you download and enable ***PubSubClient library*** in Arduino IDE.

### 4.3 Setting Up Environment in Google VM

Run all the following commands before creating the python script:

```
$ sudo apt-get install mosquitto
```

```
$ sudo apt-get install mosquitto-clients
```

```
$ sudo apt install python3-pip
```

```
$ pip install paho-mqtt
```

```
$ sudo apt-get install -y mongodb
```

```
$ pip install pymongo
```

```
$ pip3 install pymongo google-cloud-bigquery paho-mqtt
```

### 4.4 Creating the python script and running it

Use the python script from GitHub repo: https://github.com/mugunthantypical/CPC357-Assignment-2/blob/main/mongo-bq.py.

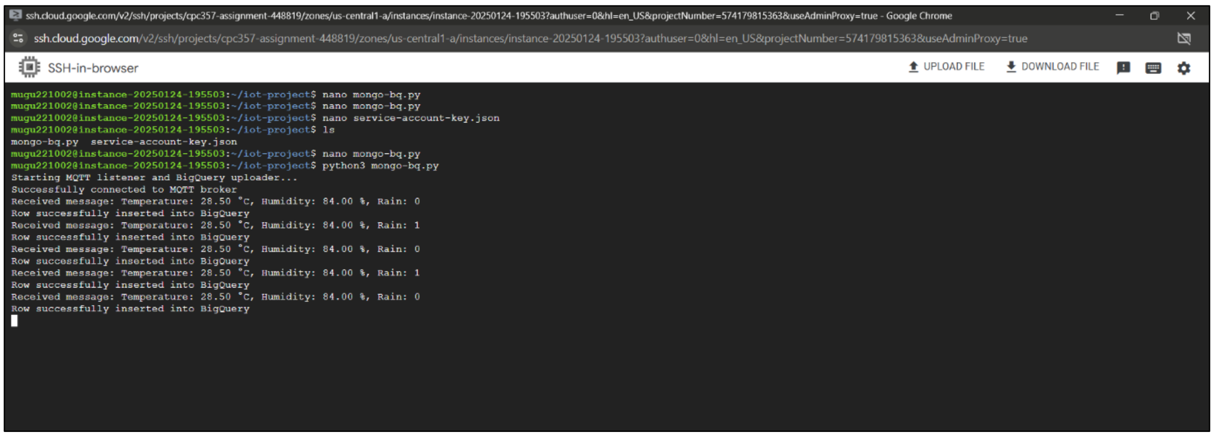
Simple follow the below command to create and run the script.

```
$ nano mongo-bq.py
```

Then copy the code and paste into the nano editor. After that, click ***Ctrl+O*** to save and ***Ctrl+X*** to exit the nano editor. You have to include your service account key in the same directory too. Copy the content from the JSON file , create a ***service-account-key.json*** file using the same command that used to create the python script, and paste it into the file. Save and close the file then run the below command to execute python script.

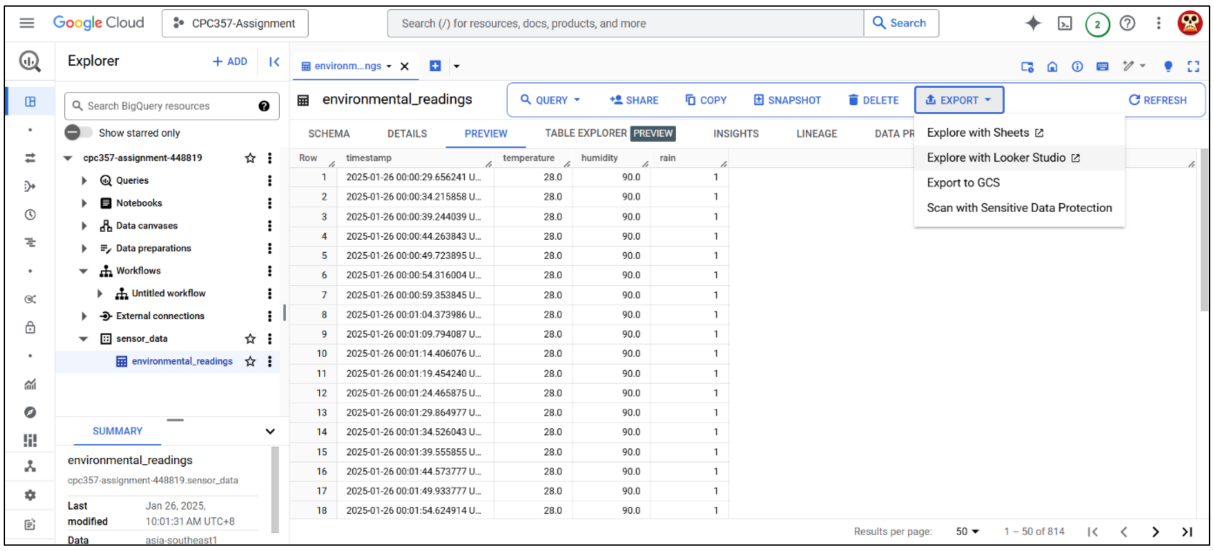
```
$ python3 mongo-bq.py
```

You will see the below output indicating the sensor readings are successfully inserted into BigQuery table.



```
mugu221002@instance-20250124-195503:~/iot-project$ nano mongo-bq.py
mugu221002@instance-20250124-195503:~/iot-project$ nano mongo-bq.py
mugu221002@instance-20250124-195503:~/iot-project$ nano service-account-key.json
mugu221002@instance-20250124-195503:~/iot-project$ ls
mongo-bq.py  service-account-key.json
mugu221002@instance-20250124-195503:~/iot-project$ nano mongo-bq.py
mugu221002@instance-20250124-195503:~/iot-project$ python3 mongo-bq.py
Starting MQTT listener and BigQuery uploader...
Successfully connected to MQTT broker
Received message: Temperature: 28.50 °C, Humidity: 84.00 %, Rain: 0
Row successfully inserted into BigQuery
Received message: Temperature: 28.50 °C, Humidity: 84.00 %, Rain: 1
Row successfully inserted into BigQuery
Received message: Temperature: 28.50 °C, Humidity: 84.00 %, Rain: 0
Row successfully inserted into BigQuery
Received message: Temperature: 28.50 °C, Humidity: 84.00 %, Rain: 1
Row successfully inserted into BigQuery
Received message: Temperature: 28.50 °C, Humidity: 84.00 %, Rain: 0
Row successfully inserted into BigQuery
```

Now navigate to the BigQuery table and click the ***Export > Explore with Looker Studio***. You will be redirect to Looker Studio where you can create charts according to your preference.

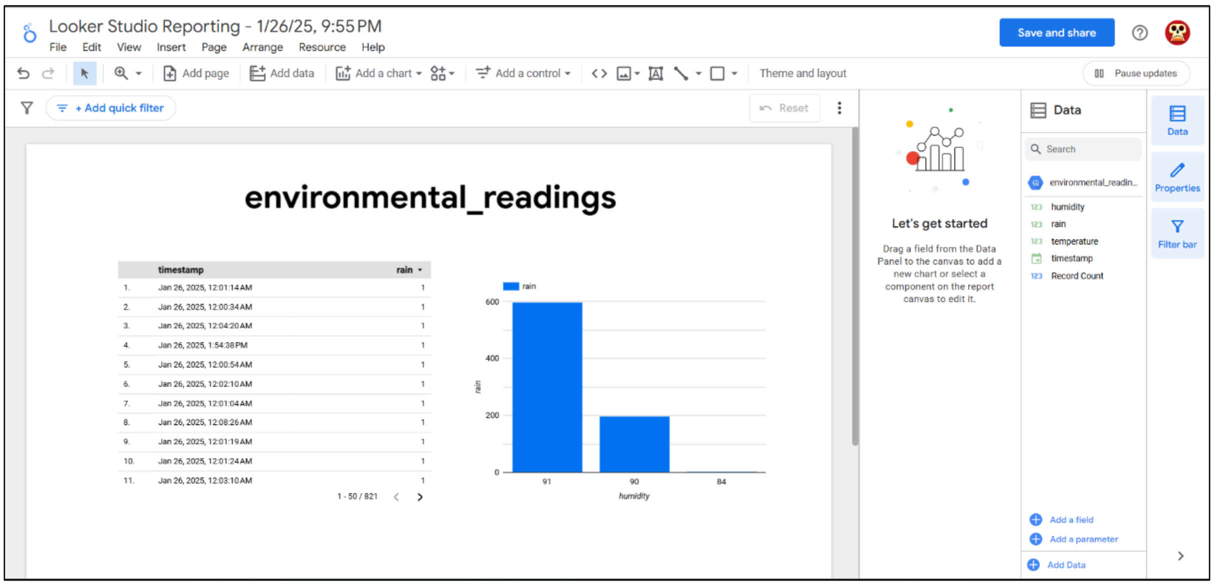


EXPORT
Explore with Sheets
Explore with Looker Studio
Export to GCS
Scan with Sensitive Data Protection

environmental_readings

| Row | timestamp | temperature | humidity | rain |
|---|---|---|---|---|
| 1 | 2025-01-26 00:00:29.656241 U... | 28.0 | 90.0 | 1 |
| 2 | 2025-01-26 00:00:34.215858 U... | 28.0 | 90.0 | 1 |
| 3 | 2025-01-26 00:00:39.244039 U... | 28.0 | 90.0 | 1 |
| 4 | 2025-01-26 00:00:44.263843 U... | 28.0 | 90.0 | 1 |
| 5 | 2025-01-26 00:00:49.723895 U... | 28.0 | 90.0 | 1 |
| 6 | 2025-01-26 00:00:54.316004 U... | 28.0 | 90.0 | 1 |
| 7 | 2025-01-26 00:00:59.353845 U... | 28.0 | 90.0 | 1 |
| 8 | 2025-01-26 00:01:04.373986 U... | 28.0 | 90.0 | 1 |
| 9 | 2025-01-26 00:01:09.794087 U... | 28.0 | 90.0 | 1 |
| 10 | 2025-01-26 00:01:14.406076 U... | 28.0 | 90.0 | 1 |
| 11 | 2025-01-26 00:01:19.454240 U... | 28.0 | 90.0 | 1 |
| 12 | 2025-01-26 00:01:24.465875 U... | 28.0 | 90.0 | 1 |
| 13 | 2025-01-26 00:01:29.864977 U... | 28.0 | 90.0 | 1 |
| 14 | 2025-01-26 00:01:34.526043 U... | 28.0 | 90.0 | 1 |
| 15 | 2025-01-26 00:01:39.555855 U... | 28.0 | 90.0 | 1 |
| 16 | 2025-01-26 00:01:44.573777 U... | 28.0 | 90.0 | 1 |
| 17 | 2025-01-26 00:01:49.933777 U... | 28.0 | 90.0 | 1 |
| 18 | 2025-01-26 00:01:54.624914 U... | 28.0 | 90.0 | 1 |

Results per page: 50   1 – 50 of 814



environmental_readings

| | timestamp | rain |
|---|---|---|
| 1. | Jan 26, 2025, 12:01:14 AM | 1 |
| 2. | Jan 26, 2025, 12:00:34 AM | 1 |
| 3. | Jan 26, 2025, 12:04:20 AM | 1 |
| 4. | Jan 26, 2025, 1:54:38 PM | 1 |
| 5. | Jan 26, 2025, 12:00:54 AM | 1 |
| 6. | Jan 26, 2025, 12:02:10 AM | 1 |
| 7. | Jan 26, 2025, 12:01:04 AM | 1 |
| 8. | Jan 26, 2025, 12:08:26 AM | 1 |
| 9. | Jan 26, 2025, 12:01:19 AM | 1 |
| 10. | Jan 26, 2025, 12:01:24 AM | 1 |
| 11. | Jan 26, 2025, 12:03:10 AM | 1 |

1 - 50 / 821

# 5. Security Considerations

## 5.1 Security Measures Implemented

1. **Transport Layer Security (TLS) for MQTT Communication**:

   All communications between the **microcontroller** and the **MQTT broker** are encrypted using **TLS** over port **8883**, ensuring that data remains **confidential** and protected from **eavesdropping** or **man-in-the-middle attacks** during transmission. This encryption mechanism guarantees that even if data packets are intercepted, they cannot be read or altered without proper decryption keys. Additionally, the use of **TLS certificates** validates both the client and server, establishing a trusted connection and preventing communication with unauthorized or rogue servers.

2. **Firewall Rules in Virtual Private Cloud (VPC)**:

   **Firewall rules** within the **Virtual Private Cloud (VPC)** are configured to regulate **inbound** and **outbound traffic**, ensuring that only **authorized devices** and **IP ranges** can interact with the **MQTT broker** and other system resources. Specific ports are selectively opened to limit exposure to vulnerabilities:\n - **Port 8883** is enabled exclusively for encrypted MQTT traffic.

- **Port 22**, used for SSH access, is further secured by restricting access to specific **IP addresses. This** strict control minimizes the system's attack surface and prevents unauthorized attempts to access cloud components, ensuring **network-level security**.

3. **Identity and Access Management (IAM)**:

   The system leverages **Google Cloud IAM** to implement **role-based access control (RBAC)**, ensuring that each service and user has the **minimum necessary permissions** required to perform their functions. This adherence to the principle of **least privilege** minimizes the risk of accidental or malicious misuse of resources. Examples include:

   - Assigning the **BigQuery Admin role** exclusively to service accounts responsible for **managing data storage** and **running queries**.

   - Restricting **end-user accounts** to **read-only permissions** for dashboards in **Looker Studio**, ensuring they cannot modify or delete data. IAM policies are reviewed regularly to align with changing system requirements, enhancing security over time.

4. **Authentication and Authorization**:

   Secure **API keys** and **OAuth tokens** are used to authenticate connections between the microcontroller, MQTT broker, and cloud services. These tokens ensure that only **verified devices** and **users** can access the system's endpoints. Furthermore, API keys are periodically rotated, and token lifetimes are limited to prevent misuse in case of a compromise. All **API endpoints** are configured to reject unauthenticated requests, providing a robust framework to prevent **unauthorized access**.

5. **Data Encryption in Transit and at Rest**:

   Data is encrypted **in transit** using **TLS** for MQTT communication and **HTTPS** for accessing dashboards. This encryption prevents the exposure of sensitive data as it moves between system components. Additionally, all data stored in **BigQuery** and **MongoDB** is encrypted **at rest** using Google Cloud's default **AES-256 encryption**. This ensures that even if storage devices are physically compromised, the data remains secure and unreadable without the decryption keys. Google Cloud's encryption mechanisms are compliant with industry standards, providing **trust and reliability** to users.

## 5.2 Strategies for Data Protection and Access Control

### Regular Key Rotation

To minimize the risk of compromised credentials, the system implements **regular key rotation** for all **encryption keys** and **API keys**. This practice ensures that even if a key is exposed, its usage window is limited. By automating this process, the system reduces the administrative burden and enhances the overall **security posture**. This proactive approach helps maintain **data confidentiality** and limits exposure to potential threats.

**Access Control Policies**

The system enforces **granular access policies** to ensure that only **authorized users** and devices have access to specific resources. **Role-based access control (RBAC)** is implemented using Google Cloud **Identity and Access Management (IAM)**, assigning roles with **minimum permissions** required for tasks. For example, the **BigQuery Admin role** allows managing data storage, while **end-user accounts** have read-only access to visualizations in **Looker Studio**. These policies are crucial for maintaining **system integrity** and preventing unauthorized access.

**Monitoring and Alerts**

**Real-time monitoring** is set up through **Google Cloud Monitoring**, which tracks system activity and identifies unusual patterns. Alerts are configured to notify administrators about **potential breaches**, such as unauthorized access attempts or anomalies in data traffic. By reviewing logs from **Cloud Audit Logs**, administrators can quickly identify and respond to suspicious activities. This continuous monitoring ensures **timely interventions** to mitigate risks and maintain **data security**.

**Backup and Disaster Recovery Plan**

The system employs a **robust backup strategy**, regularly saving critical data to secure locations. This ensures data can be restored quickly in case of a breach or system failure. Additionally, a **disaster recovery plan** is in place to restore services within **minutes**, ensuring minimal downtime. These measures protect the system against **data loss** and provide operational resilience during **unexpected events**.

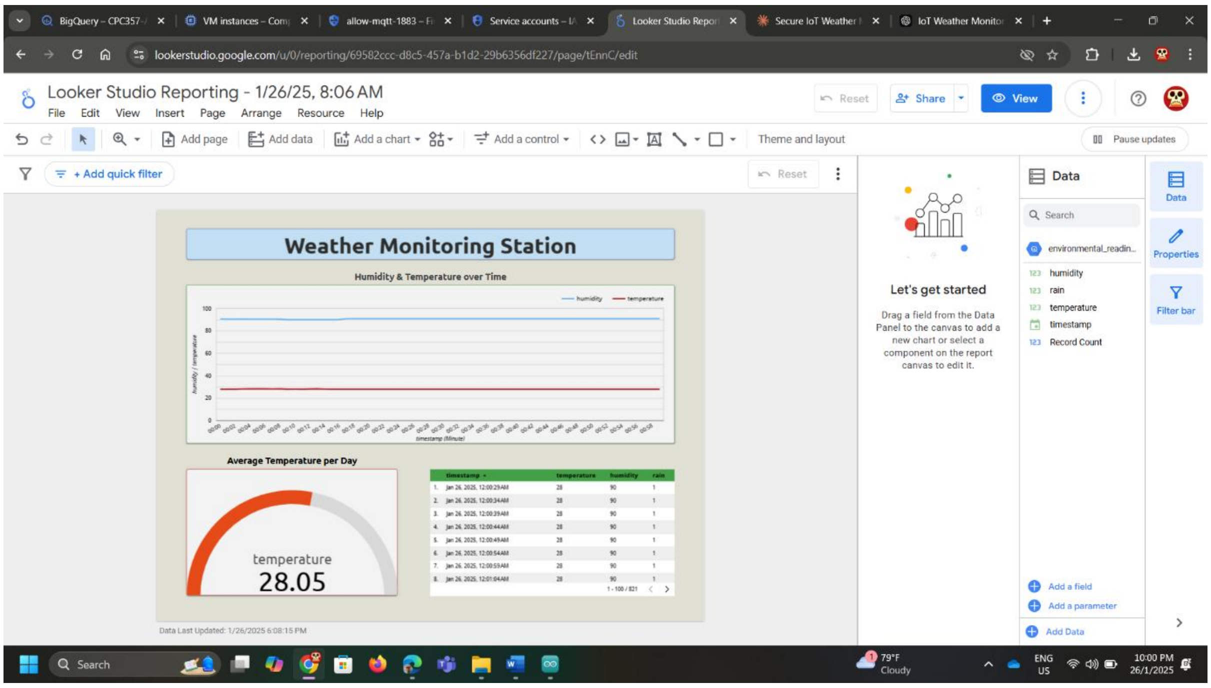
**Penetration Testing and Vulnerability Scanning**

The system undergoes periodic **penetration testing** to identify and fix vulnerabilities that could be exploited by attackers. Automated **vulnerability scans** are also conducted to detect **misconfigurations**, outdated libraries, or exposed endpoints. By addressing these weaknesses proactively, the system ensures **data integrity** and reduces the likelihood of successful cyberattacks.

## Token Expiration and Session Management

**Authentication tokens** are configured with short expiration times, requiring users and devices to regularly **reauthenticate**. This reduces the risk of unauthorized access if tokens are intercepted. Additionally, session timeouts are enforced, automatically logging out inactive users to prevent **session hijacking**. These measures enhance **access control** while maintaining a balance between **usability** and **security**.

## 6. Appendix



| | timestamp | temperature | humidity | rain |
|---|---|---|---|---|
| 1. | Jan 26, 2025, 12:00:29AM | 28 | 90 | 1 |
| 2. | Jan 26, 2025, 12:00:34AM | 28 | 90 | 1 |
| 3. | Jan 26, 2025, 12:00:39AM | 28 | 90 | 1 |
| 4. | Jan 26, 2025, 12:00:44AM | 28 | 90 | 1 |
| 5. | Jan 26, 2025, 12:00:49AM | 28 | 90 | 1 |
| 6. | Jan 26, 2025, 12:00:54AM | 28 | 90 | 1 |
| 7. | Jan 26, 2025, 12:00:59AM | 28 | 90 | 1 |
| 8. | Jan 26, 2025, 12:01:04AM | 28 | 90 | 1 |

1 - 100 / 821

*Figure 1A: Dashboard of the Weather Monitoring Station*

GitHub Repo: https://github.com/mugunthantypical/CPC357-Assignment-2.git