**GCP Secret Manager: Complete Beginner Guide**

Google Cloud **Secret Manager** is a secure and convenient service that allows you to store, manage, and access **secrets** (such as passwords, API keys, and certificates) centrally. It integrates with GCP IAM for fine-grained access control and logs all access attempts using Cloud Audit Logs.

**🔐 02. What is Secret Manager?**

**Secret Manager** is a fully managed service that allows you to store and access secrets securely. Secrets are versioned and encrypted with Google-managed or customer-managed encryption keys.

**🔧 Use Cases**

* Store API keys, DB passwords, OAuth tokens.
* Access secrets securely from GCP services like Cloud Run, Cloud Functions, or GKE.
* Enforce access control via IAM roles.
* Audit secret access for compliance.

**🛡️ 03. IAM Roles in Secret Manager**

GCP provides different roles to manage access:

| **Role** | **Description** | **Permissions** |
| --- | --- | --- |
| roles/secretmanager.admin | Full access to manage secrets | Create, update, delete, manage versions |
| roles/secretmanager.editor | Update secrets, cannot manage permissions | Add/update secrets |
| roles/secretmanager.viewer | View secret metadata only | No access to payload |
| roles/secretmanager.secretAccessor | Read secret payloads | access permission only |
| roles/owner, roles/editor | Full project access | Includes Secret Manager |

**🧪 04. Hands-On Example**

Let’s go through an example of **creating a secret**, **setting permissions**, and **accessing it** from a **Cloud Function**.

**✅ 4.1 Create a Secret via CLI**

gcloud secrets create db-password \

--replication-policy="automatic"

Add a value to the secret:

echo -n "my-db-pass-123" | gcloud secrets versions add db-password --data-file=-

**👤 4.2 Grant IAM Permissions**

Allow Cloud Function's service account to access the secret:

gcloud secrets add-iam-policy-binding db-password \

--member="[serviceAccount:MY-FUNCTION@PROJECT-ID.iam.gserviceaccount.com](mailto:serviceAccount%3AMY-FUNCTION@PROJECT-ID.iam.gserviceaccount.com)" \

--role="roles/secretmanager.secretAccessor"

Replace [MY-FUNCTION@PROJECT-ID.iam.gserviceaccount.com](mailto:MY-FUNCTION@PROJECT-ID.iam.gserviceaccount.com) with your function’s service account email.

**☁️ 4.3 Accessing Secret in Cloud Function (Python Example)**

import base64

from google.cloud import secretmanager

def access\_secret(request):

project\_id = "your-project-id"

secret\_id = "db-password"

version\_id = "latest"

client = secretmanager.SecretManagerServiceClient()

secret\_name = f"projects/{project\_id}/secrets/{secret\_id}/versions/{version\_id}"

response = client.access\_secret\_version(name=secret\_name)

payload = response.payload.data.decode("UTF-8")

return f"The secret is: {payload}"

**🌐 4.4 Accessing Secret from GCP Console**

1. Go to **Secret Manager** in GCP Console.
2. Click on your secret (e.g., db-password).
3. Click on **"View Secret Value"** to access it.
4. You’ll see a versioned history with creation date and access logs.

**🌐 4.5 Accessing Secret by terraform**

provider "google" {

project = "your-project-id"

region = "your-region"

}

# 🔍 Data source to fetch the secret version

data "google\_secret\_manager\_secret\_version" "db\_password" {

secret = "db-password"

version = "latest"

}

# Decode the base64-encoded secret

locals {

db\_password = data.google\_secret\_manager\_secret\_version.db\_password.secret\_data

}

# 🧪 Example: Use the secret in a resource

resource "google\_sql\_user" "default" {

name = "myuser"

instance = "my-sql-instance"

password = local.db\_password

}

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.