Comprehensive Implementation Guide: Secure Data Lake with Cloud Storage and Dataproc (PRJ-GCP-DATA-078)

1. Project Overview

The **PRJ-GCP-DATA-078** project delivers a robust, secure, and governed data lake architecture on Google Cloud Platform (GCP). This solution is engineered to handle sensitive and regulated data, providing a foundation for advanced analytics while strictly adhering to enterprise security and compliance mandates. The core components of this architecture are **Cloud Storage** for scalable, durable, and cost-effective data storage, and **Dataproc** for managed, high-performance big data processing, specifically for Extract, Transform, Load (ETL) and Extract, Load, Transform (ELT) workloads.

What sets this project apart is its defense-in-depth security model, which integrates several key GCP security and governance services:

- **VPC Service Controls (VPC-SC):** Establishes a security perimeter to prevent unauthorized access and data exfiltration.
- **Cloud Data Loss Prevention (Cloud DLP):** Automatically discovers, classifies, and de-identifies sensitive data.
- **Cloud Key Management Service (Cloud KMS):** Enforces Customer-Managed Encryption Keys (CMEK) for all data at rest, providing full control over the encryption lifecycle.
- **Dataplex:** Provides a unified, centralized control plane for metadata management, data quality, and governance across the entire data lake.

This guide provides a detailed, step-by-step blueprint for deploying this production-ready, secure data lake environment.

2. Business Context: Value, ROI, and Efficiency Gains

In the modern data landscape, the ability to analyze vast amounts of data is crucial, but it must be balanced with the imperative to protect sensitive information. This project directly addresses this challenge, transforming a potential compliance liability into a secure, high-value asset.

The Challenge: Data Security and Governance at Scale

Organizations operating in regulated industries (e.g., finance, healthcare) face a constant struggle to maintain compliance while leveraging cloud-native data services. Traditional security measures often fail to scale, leading to:

1. **High Risk of Data Exfiltration:** Unsecured cloud storage and data processing services are prime targets for unauthorized access or accidental data leaks.
2. **Manual Compliance Overhead:** Data classification, access auditing, and encryption key management are often manual, time-consuming, and error-prone processes.
3. **Fragmented Data Silos:** Data spread across various services (Cloud Storage, BigQuery, Cloud SQL) lacks unified governance, complicating security and discovery.

The Solution: A Governed, Secure Data Perimeter

The PRJ-GCP-DATA-078 architecture implements a **secure data perimeter** using VPC Service Controls, ensuring that data never leaves the trusted network boundary. Furthermore, the integration of Dataplex and Cloud DLP automates critical governance functions, reducing manual effort and improving data quality.

Quantified Business Value and Return on Investment (ROI)

The investment in this secure data lake yields significant returns across three key areas: risk mitigation, operational efficiency, and business enablement.

Value Driver	Description	Quantified Benefit / ROI
Risk Mitigation (Security)	Prevents data breaches and unauthorized access through a VPC-SC perimeter and CMEK encryption.	**Avoided Cost of Breach:** Estimated average cost of a data breach is $4.45 million (IBM, 2023). This architecture significantly reduces the probability of such an event.
Operational Efficiency (Governance)	Automated data discovery, classification (Cloud DLP), and unified governance (Dataplex) replace manual processes.	**Efficiency Gain:** Up to 40% reduction in time spent on data governance, compliance auditing, and security reviews.
Compliance Automation	Provides clear, auditable evidence (Cloud Audit Logs, DLP reports) of compliance with major frameworks.	**Reduced Audit Costs:** Lower external audit fees and reduced internal staff time dedicated to preparing for compliance checks.
Data Monetization	Securely unlocks sensitive data for analytics and machine learning, enabling new business insights.	**Revenue Growth:** Faster time-to-insight and secure access to data drives new product development and revenue streams.

The architecture's focus on automation and security-by-design ensures that the initial setup cost is quickly offset by reduced operational expenses and the avoidance of catastrophic security incidents.

3. GRC Mapping: Compliance and Security Controls

Governance, Risk, and Compliance (GRC) are not afterthoughts but are foundational to this data lake deployment. The architecture is explicitly designed to align with major industry and regulatory frameworks, providing a clear path to audit readiness.

Alignment with Major Compliance Frameworks

The project's security controls map directly to the control families of leading global standards.

Framework	Control ID / Area	Project Implementation
NIST SP 800-53 (Rev. 5)	**PR.DS-1** (Data-at-Rest), **PR.DS-5** (Data-in-Transit)	Enforced CMEK (Cloud KMS) for encryption at rest; VPC Service Controls for network security.
ISO/IEC 27001:2022	**A.8.2** (Information Classification), **A.18.1** (Compliance)	Cloud DLP for automated data classification; Dataplex for unified governance and policy enforcement.
SOC 2 (Trust Services Criteria)	**CC6.1** (Logical Access), **CC6.7** (Data Classification)	IAM Least Privilege, BigQuery Column-Level Security, and Cloud DLP.
CIS Controls (v8)	**Control 3** (Data Protection), **Control 13** (Network Monitoring)	CMEK, VPC Service Controls, and Cloud Audit Logs for continuous monitoring.

Detailed Security Controls Implemented

The defense-in-depth strategy is realized through the following GCP service integrations:

Control	GCP Service	Purpose and Implementation Detail
Data Classification & Discovery	**Dataplex, Cloud DLP**	Dataplex inventories data assets, while Cloud DLP scans data in the raw stage to identify PII, PCI, and other sensitive information, enabling automated policy application.
Encryption at Rest	**Cloud KMS (CMEK)**	Customer-Managed Encryption Keys are mandatory for all Cloud Storage buckets and BigQuery datasets, ensuring the customer retains full control over the cryptographic keys.
Perimeter Security	**VPC Service Controls**	Creates a secure, network-level boundary around the core data services (Storage, BigQuery, Dataproc, DLP, KMS), preventing data from being moved to external, unauthorized projects.
Fine-Grained Access Control	**BigQuery, IAM**	Implements column-level security in BigQuery to restrict access to sensitive fields, complementing IAM roles for broader resource access.
Audit and Monitoring	**Cloud Audit Logs**	Captures all administrative and data access events, providing an immutable record for compliance reporting and forensic analysis.

Regulatory Alignment

The security controls implemented are essential for meeting key regulatory requirements globally:

Regulation	Requirement	Project Alignment
GDPR (General Data Protection Regulation)	Article 32 (Security of Processing), Article 25 (Privacy by Design)	Encryption (CMEK), access control (IAM, VPC-SC), and data minimization (Cloud DLP de-identification).
HIPAA (Health Insurance Portability and Accountability Act)	§ 164.312(a)(2)(iv) (Encryption), § 164.312(e) (Transmission Security)	CMEK for ePHI at rest and VPC-SC to secure data transmission paths.
PCI DSS (Payment Card Industry Data Security Standard)	Requirement 3 (Protect Stored Cardholder Data), Requirement 4 (Encrypt Transmission)	CMEK for cardholder data storage and VPC-SC to ensure secure network transmission.

4. Prerequisites

Successful deployment requires the following accounts, tools, and permissions to be configured prior to execution.

4.1. GCP Account and Project Setup

1. **GCP Project:** A Google Cloud Project must be created with an active billing account linked.
2. **Project ID:** Note the Project ID, as it will be used extensively in environment variables.
3. **Billing:** Ensure the project has billing enabled to use the required services (Cloud Storage, Dataproc, etc.).

4.2. Local Environment Setup

1. **Google Cloud SDK (gcloud CLI):** The `gcloud` command-line tool must be installed and configured on your local machine or a cloud shell environment.
2. **Authentication:** Authenticate the CLI and set the target project:

```
gcloud auth login
gcloud config set project [YOUR_PROJECT_ID]
```

3. **APIs:** Ensure the following APIs are enabled (this will be handled in Step 6.2, but pre-checking is recommended): Compute Engine, Cloud Storage, Dataproc, BigQuery, Cloud KMS, Cloud DLP, Dataplex, and Service Control.

4.3. IAM Permissions

The user or service account executing the deployment script must possess sufficient permissions. The `roles/owner` role is the simplest but least-privileged approach is recommended for production:

- `roles/editor` or equivalent for resource creation (Storage, BigQuery, Dataproc).
- `roles/iam.serviceAccountAdmin` for creating the Dataproc service account.
- `roles/cloudkms.admin` for managing the KMS key and key ring.
- `roles/accesscontextmanager.policyAdmin` for configuring VPC Service Controls (if managed by the deploying user).

5. Architecture Overview: Multi-Stage Secure Data Lake

The data lake follows a standard, multi-stage pattern (Raw, Curated) but is overlaid with a comprehensive security and governance fabric.

Data Flow and Stages

1. **Raw Stage (Cloud Storage):** Data from external sources lands here. This bucket is CMEK-encrypted and subject to continuous Cloud DLP scanning for sensitive data discovery. It is registered as a `RAW` zone in Dataplex.
2. **Processing Layer (Dataproc):** A managed Hadoop/Spark cluster executes ETL/ELT jobs. The cluster uses a dedicated Service Account with least-privilege access to read from the Raw bucket and write to the Curated bucket and BigQuery.
3. **Curated Stage (Cloud Storage & BigQuery):**

- **Curated Bucket:** Stores processed, cleaned, and transformed data, typically in optimized formats like Parquet or Avro. It is also CMEK-encrypted and registered as a `CURATED` zone in Dataplex.
- **BigQuery:** The final destination for structured, high-value data, enabling high-performance analytics. The dataset is CMEK-encrypted and subject to column-level security policies.

Security and Governance Fabric

- **VPC Service Controls (VPC-SC):** This is the outermost layer, creating a security perimeter that encompasses all core services (Cloud Storage, BigQuery, Dataproc, DLP, KMS). This perimeter ensures that data operations can only occur within the trusted boundary, effectively mitigating data exfiltration risks.
- **Cloud KMS (CMEK):** Ensures that all data at rest is encrypted using keys managed by the customer, fulfilling regulatory requirements for key control.
- **Dataplex:** Acts as the unified governance layer, providing a single pane of glass for data discovery, metadata management, and data quality checks across the Cloud Storage and BigQuery assets.

6. Step-by-Step Implementation

The deployment is broken down into logical, sequential steps. **It is critical to execute the commands in the order presented.**

Step 6.1: Initialize Environment Variables

Before running any commands, set the project-specific variables. The script uses `date +%s` to ensure globally unique bucket names, a requirement for Cloud Storage. **You must replace `PRJ-GCP-DATA-078` with your actual Project ID.**

```
# --- Configuration Variables ---
export PROJECT_ID="PRJ-GCP-DATA-078" # <<< REPLACE WITH YOUR ACTUAL PROJECT
ID
export REGION="us-central1"
export ZONE="${REGION}-a"
export RAW_BUCKET_NAME="${PROJECT_ID}-raw-data-$(date +%s)"
export CURATED_BUCKET_NAME="${PROJECT_ID}-curated-data-$(date +%s)"
export DATAPROC_CLUSTER_NAME="dataproc-etl-cluster"
export BQ_DATASET_NAME="analytics_data_lake"
export KMS_KEYRING_NAME="data-lake-keyring"
export KMS_KEY_NAME="data-lake-cmek"
export SERVICE_ACCOUNT_ID="dataproc-sa"
export
SERVICE_ACCOUNT_EMAIL="${SERVICE_ACCOUNT_ID}@${PROJECT_ID}.iam.gserviceaccount.c

# Set the active project for gcloud
echo "Setting active project to $PROJECT_ID..."
gcloud config set project $PROJECT_ID
```

Step 6.2: Enable Required APIs

This step ensures all necessary GCP services are activated within your project.

```
echo "Enabling required GCP APIs..."
gcloud services enable \
    compute.googleapis.com \
    storage.googleapis.com \
    dataproc.googleapis.com \
    bigquery.googleapis.com \
    cloudkms.googleapis.com \
    dlp.googleapis.com \
    dataplex.googleapis.com \
    servicecontrol.googleapis.com
```

Step 6.3: Create Service Account and Grant Permissions

A dedicated service account is created for the Dataproc cluster to adhere to the principle of least privilege. This account requires permissions to manage storage, edit BigQuery data, and use the KMS key.

```
echo "Creating Dataproc Service Account and granting roles..."
# 1. Create a dedicated service account for Dataproc
gcloud iam service-accounts create $SERVICE_ACCOUNT_ID \
    --display-name "Dataproc ETL Service Account"

# 2. Grant necessary roles for data processing and storage
gcloud projects add-iam-policy-binding $PROJECT_ID \
    --member="serviceAccount:$SERVICE_ACCOUNT_EMAIL" \
    --role="roles/dataproc.worker" # Required for Dataproc cluster operations

gcloud projects add-iam-policy-binding $PROJECT_ID \
    --member="serviceAccount:$SERVICE_ACCOUNT_EMAIL" \
    --role="roles/storage.admin" # Required to read/write to Cloud Storage
buckets

gcloud projects add-iam-policy-binding $PROJECT_ID \
    --member="serviceAccount:$SERVICE_ACCOUNT_EMAIL" \
    --role="roles/bigquery.dataEditor" # Required to write data to BigQuery

gcloud projects add-iam-policy-binding $PROJECT_ID \
    --member="serviceAccount:$SERVICE_ACCOUNT_EMAIL" \
    --role="roles/cloudkms.cryptoKeyEncrypterDecrypter" # Required to use the
CMEK key
```

Step 6.4: Configure Cloud KMS (CMEK)

Customer-Managed Encryption Keys are configured to provide cryptographic control. This involves creating a Key Ring and a Key, and crucially, granting the Cloud Storage Service Agent permission to use this key for bucket encryption.

```
echo "Configuring Cloud KMS for CMEK..."
# 1. Create KMS Key Ring
gcloud kms keyrings create $KMS_KEYRING_NAME \
    --location $REGION

# 2. Create KMS Key
gcloud kms keys create $KMS_KEY_NAME \
    --keyring $KMS_KEYRING_NAME \
    --location $REGION \
    --purpose "encryption"

# 3. Grant the Cloud Storage Service Agent permission to use the key
# The GCS service agent is a Google-managed service account unique to your
project.
PROJECT_NUMBER=$(gcloud projects describe $PROJECT_ID --
format='value(projectNumber)')
GCS_SERVICE_AGENT="service-${PROJECT_NUMBER}@gs-project-
accounts.iam.gserviceaccount.com"

echo "Granting KMS permissions to GCS Service Agent: $GCS_SERVICE_AGENT"
gcloud kms keys add-iam-policy-binding $KMS_KEY_NAME \
    --keyring $KMS_KEYRING_NAME \
    --location $REGION \
    --member "serviceAccount:$GCS_SERVICE_AGENT" \
    --role "roles/cloudkms.cryptoKeyEncrypterDecrypter"
```

Step 6.5: Create Cloud Storage Buckets (CMEK Enabled)

The Raw and Curated buckets are created in the specified region and immediately configured to use the CMEK key created in the previous step. This ensures all data written to these buckets is encrypted with the customer-controlled key.

```
echo "Creating CMEK-enabled Cloud Storage Buckets..."
# 1. Get the full KMS key resource name
KMS_KEY_RESOURCE=$(gcloud kms keys describe $KMS_KEY_NAME \
    --keyring $KMS_KEYRING_NAME \
    --location $REGION \
    --format='value(name)')

# 2. Create Raw Bucket with CMEK
echo "Creating Raw Bucket: gs://$RAW_BUCKET_NAME"
gsutil mb -l $REGION gs://$RAW_BUCKET_NAME
gsutil kms set $KMS_KEY_RESOURCE gs://$RAW_BUCKET_NAME

# 3. Create Curated Bucket with CMEK
echo "Creating Curated Bucket: gs://$CURATED_BUCKET_NAME"
gsutil mb -l $REGION gs://$CURATED_BUCKET_NAME
gsutil kms set $KMS_KEY_RESOURCE gs://$CURATED_BUCKET_NAME
```

Step 6.6: Create BigQuery Dataset (CMEK Enabled)

The BigQuery dataset, which will hold the final analytical data, is also created with the same CMEK key as its default encryption key.

```
echo "Creating CMEK-enabled BigQuery Dataset: $BQ_DATASET_NAME"
# Create BigQuery Dataset
bq mk --dataset \
    --default_kms_key $KMS_KEY_RESOURCE \
    --location $REGION \
    $BQ_DATASET_NAME
```

Step 6.7: Deploy Dataproc Cluster

The Dataproc cluster is deployed, configured to use the dedicated service account, and initialized with necessary Python libraries (pandas, pyarrow) for ETL processing.

```
echo "Deploying Dataproc Cluster: $DATAPROC_CLUSTER_NAME"
gcloud dataproc clusters create $DATAPROC_CLUSTER_NAME \
    --region $REGION \
    --zone $ZONE \
    --service-account $SERVICE_ACCOUNT_EMAIL \
    --bucket $RAW_BUCKET_NAME \
    --image-version 2.1-debian11 \
    --master-machine-type n2-standard-4 \
    --worker-machine-type n2-standard-4 \
    --num-workers 2 \
    --enable-component-gateway \
    --optional-components JUPYTER \
    --metadata "PIP_PACKAGES=pandas,pyarrow" \
    --initialization-actions gs://dataproc-initialization-actions/python/pip-
install.sh
```

Step 6.8: Configure Dataplex Lake (Unified Governance)

Dataplex is configured to provide a unified view and governance layer over the data assets. A Lake is created, and the Raw Bucket and BigQuery Dataset are registered as Zones within the Lake.

```
echo "Configuring Dataplex Lake and Zones..."
# 1. Create a Dataplex Lake
gcloud dataplex lakes create data-lake-governance \
    --location $REGION \
    --display-name "Data Lake Governance"

# 2. Attach the Raw Bucket as a Dataplex Zone
gcloud dataplex zones create raw-zone \
    --location $REGION \
    --lake data-lake-governance \
    --display-name "Raw Data Zone" \
    --type RAW \
    --resource-location-type SINGLE_REGION \
    --discovery-enabled \
    --discovery-schedule "0 * * * *" \
    --resource-containers "projects/$PROJECT_ID/buckets/$RAW_BUCKET_NAME"

# 3. Attach the BigQuery Dataset as a Dataplex Zone
gcloud dataplex zones create curated-zone \
    --location $REGION \
    --lake data-lake-governance \
    --display-name "Curated Data Zone" \
    --type CURATED \
    --resource-location-type SINGLE_REGION \
    --discovery-enabled \
    --discovery-schedule "0 * * * *" \
    --resource-containers
"projects/$PROJECT_ID/locations/$REGION/datasets/$BQ_DATASET_NAME"
```

Step 6.9: Configure VPC Service Controls (Security Perimeter)

CRITICAL SECURITY STEP: VPC Service Controls (VPC-SC) is the cornerstone of the security perimeter. This configuration is typically managed at the Organization or Folder level and requires the `Access Context Manager API`. The following is a conceptual guide and must be executed by an administrator with the appropriate permissions.

1. **Create an Access Policy:** Ensure an Access Policy exists for your organization.
2. **Define Access Levels:** Define an Access Level that specifies the trusted identities (e.g., the Dataproc Service Account) and/or trusted IP ranges that are allowed to access the perimeter.

3. **Create a Service Perimeter:** Create a perimeter that includes the project `$PROJECT_ID` and restricts the following services:
    - `storage.googleapis.com`
    - `bigquery.googleapis.com`
    - `dataproc.googleapis.com`
    - `dlp.googleapis.com`
    - `cloudkms.googleapis.com`

This step is highly sensitive and is often done via the GCP Console or Terraform/Deployment Manager. The conceptual command below serves as a reminder of the required action:

```
echo "VPC Service Controls perimeter setup is a a critical security step."
echo "This must be configured via the Access Context Manager API or GCP
Console."
echo "Ensure the perimeter includes the project and restricts: storage,
bigquery, dataproc, dlp, and cloudkms."
```

7. Configuration Files and Data Preparation

This section details the sample ETL job and the necessary data preparation steps to test the deployed infrastructure.

7.1. PySpark ETL Job (`etl_job.py`)

This sample PySpark job demonstrates the core ETL process: reading data from the raw bucket, performing a simple transformation, and writing the result to both the curated bucket (Parquet) and BigQuery.

```python
# etl_job.py
from pyspark.sql import SparkSession
from pyspark.sql.functions import current_timestamp

# Initialize Spark Session
spark = SparkSession.builder \
    .appName("DataLakeETL") \
    .getOrCreate()

# Configuration variables are dynamically replaced by the Dataproc job
submission
# For local testing, replace the placeholders with actual values
PROJECT_ID = "${PROJECT_ID}"
RAW_BUCKET = "${RAW_BUCKET_NAME}"
CURATED_BUCKET = "${CURATED_BUCKET_NAME}"
BQ_DATASET = "${BQ_DATASET_NAME}"
BQ_TABLE = "processed_data"

# 1. Read data from Raw Bucket (assuming a sample CSV)
# The raw data is expected to be in the 'input' subdirectory
raw_path = f"gs://{RAW_BUCKET}/input/sample_data.csv"
print(f"Reading raw data from: {raw_path}")
df = spark.read.csv(raw_path, header=True, inferSchema=True)

# 2. Transformation: Add a processing timestamp
df_processed = df.withColumn("processing_timestamp", current_timestamp())

# 3. Write to Curated Bucket (Parquet format for efficiency)
# Parquet is the recommended format for data lakes due to its columnar
storage
curated_path = f"gs://{CURATED_BUCKET}/processed/data/"
print(f"Writing processed data to Curated Bucket: {curated_path}")
df_processed.write.mode("overwrite").parquet(curated_path)

# 4. Write to BigQuery
# The temporaryGcsBucket option is required for the BigQuery connector
print(f"Writing processed data to BigQuery table: {BQ_DATASET}.{BQ_TABLE}")
df_processed.write.format("bigquery") \
    .option("table", f"{BQ_DATASET}.{BQ_TABLE}") \
    .option("temporaryGcsBucket", CURATED_BUCKET) \
    .mode("overwrite") \
    .save()

print(f"ETL job completed. Data written to {curated_path} and BigQuery table
{BQ_DATASET}.{BQ_TABLE}")
```

```
spark.stop()
```

7.2. Sample Data and Job File Upload

Create the ETL job file and a small sample data file locally, then upload them to the Raw bucket.

```
echo "Creating etl_job.py locally..."
# Create the PySpark job file
cat << EOF > etl_job.py
from pyspark.sql import SparkSession
from pyspark.sql.functions import current_timestamp

spark = SparkSession.builder \
    .appName("DataLakeETL") \
    .getOrCreate()

PROJECT_ID = "\${PROJECT_ID}"
RAW_BUCKET = "\${RAW_BUCKET_NAME}"
CURATED_BUCKET = "\${CURATED_BUCKET_NAME}"
BQ_DATASET = "\${BQ_DATASET_NAME}"
BQ_TABLE = "processed_data"

raw_path = f"gs://{RAW_BUCKET}/input/sample_data.csv"
df = spark.read.csv(raw_path, header=True, inferSchema=True)

df_processed = df.withColumn("processing_timestamp", current_timestamp())

curated_path = f"gs://{CURATED_BUCKET}/processed/data/"
df_processed.write.mode("overwrite").parquet(curated_path)

df_processed.write.format("bigquery") \
    .option("table", f"{BQ_DATASET}.{BQ_TABLE}") \
    .option("temporaryGcsBucket", CURATED_BUCKET) \
    .mode("overwrite") \
    .save()

print(f"ETL job completed. Data written to {curated_path} and BigQuery table
{BQ_DATASET}.{BQ_TABLE}")

spark.stop()
EOF

echo "Creating sample_data.csv locally..."
# Create a dummy CSV file
echo "id,name,value" > sample_data.csv
echo "1,alpha,100" >> sample_data.csv
echo "2,beta,200" >> sample_data.csv
echo "3,gamma,300" >> sample_data.csv
echo "4,delta,400" >> sample_data.csv
echo "5,epsilon,500" >> sample_data.csv

echo "Uploading job file and sample data to the Raw Bucket..."
```

```
# Upload the job file and sample data to the raw bucket
gsutil cp etl_job.py gs://$RAW_BUCKET_NAME/jobs/
gsutil cp sample_data.csv gs://$RAW_BUCKET_NAME/input/
```

8. Validation & Testing

This section outlines the steps to verify that the data lake infrastructure is correctly deployed and the ETL pipeline is functioning securely.

Step 8.1: Run the ETL Job

Submit the PySpark job to the Dataproc cluster. This job will test the entire data path: reading from CMEK-encrypted raw storage, processing on Dataproc (using the dedicated SA), and writing to CMEK-encrypted curated storage and BigQuery.

```
echo "Submitting PySpark ETL job to Dataproc cluster..."
gcloud dataproc jobs submit pyspark \
    --cluster $DATAPROC_CLUSTER_NAME \
    --region $REGION \
    gs://$RAW_BUCKET_NAME/jobs/etl_job.py \
    --project $PROJECT_ID
```

Step 8.2: Validate Data in Curated Storage

After the job completes successfully, verify that the processed data (in Parquet format) exists in the Curated Cloud Storage bucket.

```
echo "Validating processed data in Curated Storage..."
# Check for Parquet files in the curated bucket
gsutil ls gs://$CURATED_BUCKET_NAME/processed/data/
```

Step 8.3: Validate Data in BigQuery

Query the BigQuery table to confirm that the data was successfully ingested and is accessible.

```
echo "Validating data ingestion in BigQuery..."
# Query the BigQuery table to confirm data ingestion
bq query --use_legacy_sql=false \
    "SELECT * FROM \`${PROJECT_ID}.${BQ_DATASET_NAME}.processed_data\` LIMIT
5"
```

Step 8.4: Validate Security Controls (DLP and CMEK)

- **CMEK Validation:** In the GCP Console, navigate to the Cloud Storage and BigQuery resources. Verify that the encryption key listed for the buckets and dataset is the KMS key `$KMS_KEY_RESOURCE`.
- **Cloud DLP Validation:** Configure a DLP inspection job to scan the Raw Bucket. The successful identification and classification of the `sample_data.csv` (or a more realistic dataset containing PII) confirms the governance layer is active.

```
echo "Manually verify Cloud DLP scan results for the raw bucket to ensure
sensitive data is correctly identified and classified."
echo "Verify CMEK status in the GCP Console for all storage and BigQuery
assets."
```

9. Troubleshooting: Common Issues and Resolutions

This table provides solutions for common deployment and runtime errors, particularly those related to the complex security components (IAM, KMS, VPC-SC).

Issue	Potential Cause	Resolution
Dataproc Job Fails	IAM permissions missing for the Dataproc Service Account (`$SERVICE_ACCOUNT_EMAIL`).	Verify the service account has `storage.admin` , `bigquery.dataEditor` , and `cloudkms.cryptoKeyEncrypterDecrypter` roles. Check Dataproc cluster logs for specific permission denied errors.
Access Denied (403)	VPC Service Controls perimeter is blocking access.	Check the VPC-SC audit logs in Cloud Audit Logs. Ensure the source IP or the service account (`$SERVICE_ACCOUNT_EMAIL`) is included in the perimeter's Access Level.
Encryption Error	Cloud Storage Service Agent lacks KMS Decrypt/Encrypt permission.	Re-run Step 6.4. The GCS service agent (`service-<PROJECT_NUMBER>@gs-project-accounts.iam.gserviceaccount.com`) must be bound to the `roles/cloudkms.cryptoKeyEncrypterDecrypter` role.
Dataplex Discovery Fails	Dataplex Service Agent lacks necessary IAM roles on the bucket/dataset.	Grant the Dataplex service account (`service-<PROJECT_NUMBER>@gcp-sa-dataplex.iam.gserviceaccount.com`) the `roles/storage.viewer` and `roles/bigquery.metadataViewer` roles.
Bucket Creation Fails (409)	The bucket name is not globally unique.	Re-run Step 6.1 to generate new, unique bucket names.

10. Cost Optimization

Managing costs is a critical aspect of cloud operations. This architecture is designed with several cost-saving mechanisms.

10.1. Compute Optimization (Dataproc)

- **Dataproc Auto-scaling:** Configure the Dataproc cluster with auto-scaling policies to dynamically adjust the number of worker nodes based on the workload. This ensures you only pay for the compute resources actively being used.

- **Preemptible VMs:** Utilize preemptible worker VMs for non-critical, fault-tolerant workloads (like batch ETL). Preemptible VMs offer significant discounts (up to 80%) compared to standard VMs.
- **Cluster Deletion:** Use the cleanup script (Section 12) or set an idle time-out policy to automatically delete the Dataproc cluster when it is not in use.

10.2. Storage Optimization (Cloud Storage)

- **Cloud Storage Lifecycle Policies:** Implement lifecycle management rules on the Raw Bucket to automatically transition older, less-frequently accessed data to cheaper storage classes (e.g., Nearline or Coldline) after 30 or 60 days.
- **Data Format:** Store curated data in columnar formats like Parquet or Avro, which are highly compressed and reduce storage footprint and I/O costs for processing.

10.3. BigQuery Optimization

- **Partitioning and Clustering:** Ensure BigQuery tables are properly partitioned (e.g., by date) and clustered (e.g., by a high-cardinality column) to minimize the amount of data scanned per query, directly reducing query costs under on-demand pricing.
- **Slot Reservations:** For predictable, high-volume workloads, consider purchasing BigQuery flat-rate slots instead of on-demand pricing for more predictable and potentially lower costs.

11. Security Best Practices

Beyond the core deployment, continuous security posture management is essential.

11.1. Continuous Security Monitoring

- **Cloud DLP Integration:** Configure automated, scheduled DLP inspection jobs to continuously scan new data landing in the raw bucket for PII and other sensitive information. Use DLP to de-identify data before it moves to the curated stage.
- **Audit Log Analysis:** Regularly analyze Cloud Audit Logs for suspicious activity, especially failed access attempts related to the VPC Service Controls perimeter.

11.2. IAM and Access Control

- **Least Privilege:** Continuously review and refine the IAM roles granted to the Dataproc Service Account and any human users. Use custom roles instead of broad primitive roles where possible.
- **BigQuery Column-Level Security:** Implement fine-grained access policies on sensitive columns within BigQuery tables to restrict access to authorized users/groups, even if they have general access to the table.

11.3. Key Management

- **Key Rotation:** Implement an automated key rotation schedule for the Cloud KMS key to comply with security policies.
- **Key Access Policy:** Strictly limit who can manage, view, or use the KMS key. The key should only be accessible by the service agents and service accounts that absolutely require it.

12. Cleanup

To avoid incurring future charges, execute the following commands to delete all resources created by this deployment. **Ensure you have the correct environment variables set from Step 6.1 before running this.**

```
echo "Starting cleanup process for project $PROJECT_ID..."

# 1. Delete the Dataproc Cluster
echo "Deleting Dataproc Cluster: $DATAPROC_CLUSTER_NAME"
gcloud dataproc clusters delete $DATAPROC_CLUSTER_NAME --region $REGION --
quiet

# 2. Delete the Dataplex Lake (This will also delete the zones)
echo "Deleting Dataplex Lake: data-lake-governance"
gcloud dataplex lakes delete data-lake-governance --location $REGION --quiet

# 3. Delete the BigQuery Dataset
echo "Deleting BigQuery Dataset: $BQ_DATASET_NAME"
bq rm -r -f $BQ_DATASET_NAME

# 4. Delete the Cloud Storage Buckets (requires -r for recursive deletion)
echo "Deleting Raw Bucket: gs://$RAW_BUCKET_NAME"
gsutil rm -r gs://$RAW_BUCKET_NAME
echo "Deleting Curated Bucket: gs://$CURATED_BUCKET_NAME"
gsutil rm -r gs://$CURATED_BUCKET_NAME

# 5. Delete the KMS Key and Key Ring
echo "Deleting KMS Key: $KMS_KEY_NAME and Key Ring: $KMS_KEYRING_NAME"
gcloud kms keys delete $KMS_KEY_NAME --keyring $KMS_KEYRING_NAME --location
$REGION --quiet
gcloud kms keyrings delete $KMS_KEYRING_NAME --location $REGION --quiet

# 6. Delete the Service Account
echo "Deleting Service Account: $SERVICE_ACCOUNT_EMAIL"
gcloud iam service-accounts delete $SERVICE_ACCOUNT_EMAIL --quiet

echo "Cleanup complete. All resources have been deleted."
```
