Technical Audit Report: g-france-vm-project

DATE: 2026-01-29 **AUTHOR:** Senior Cloud Architect and Systems Forensic Expert **SUBJECT:** Systematic Technical Audit of the "Gemini CLI" Infrastructure

Executive Summary

This report provides a forensic analysis of the `g-france-vm-project` cloud infrastructure. The audit has uncovered a system with a clearly defined architectural intent that is undermined by significant **environment drift**, manual configuration, and violations of the principle of least privilege. While the dual-hub coordination model for orchestration and failover is sound in theory, its implementation is disjointed and relies on manual processes, posing a risk to operational continuity. The investigation has identified critical discrepancies between the declared state in the primary Terraform configuration and the actual deployed infrastructure, particularly in compute resources and IAM policies.

---

1. COMPUTE & ORCHESTRATION HIERARCHY

The project employs a dual-hub model for compute and orchestration, centered around a primary orchestrator (`vm1-orch`) and a persistent secondary failover node (`vm4-seed`).

1.1. Dual-Hub Coordination Model

- **VM1-Orch (Primary):** This instance is intended to be the primary "brain" of the operation, responsible for orchestrating data processing workflows. The Terraform configuration specifies a powerful `n2-standard-8` machine type, indicating a resource-intensive role.
- **VM4-Seed (Persistent Secondary/Failover):** As documented in `GEMINI.md` and confirmed by the reverse-engineered Terraform files, `vm4-seed` is a minimal `e2-micro` instance designed for persistence and disaster recovery. It holds the "DNA" of the infrastructure and is the designated jump host for the `mobile_admin` user.

1.2. Failover Logic Analysis

The failover mechanism from `vm1-orch` to `vm4-seed` appears to be **entirely manual**. No automated failover logic, health checks, or scripts to trigger a failover were discovered in the scanned files. The recovery process, as described in `GEMINI.md`, relies on a manual `terraform apply` from `vm4-seed` to resurrect the infrastructure. This introduces a significant recovery time objective (RTO) and a dependency on human intervention.

1.3. VM2 "Workhorse" and Environment Drift

`vm2-worker` is the primary "Workhorse" for data processing, with a startup script that installs PDF processing tools. However, this audit discovered significant **environment drift** between the intended and actual state of the compute resources:

VM Name	Declared Machine Type (`g-france-vm-project/terraform`)	Actual Machine Type (`reverse_engineering`)	Discrepancy
`vm1-orch`	`n2-standard-8`	`e2-standard-4`	**Major**
`vm2-worker`	`n2-standard-8`	`e2-standard-8`	**None**

The discrepancy in `vm1-orch`'s machine type is a major concern. The system is running on a less powerful instance than intended, which could lead to performance bottlenecks. This drift indicates that changes are being made to the infrastructure outside of the primary Terraform workflow.

---

2. DATA & STATE MANAGEMENT

The project's data and state are managed through a combination of Google Cloud Storage (GCS) and Firestore.

2.1. GCS Bucket Analysis

Three GCS buckets are used for a data processing pipeline and Terraform state storage:

Bucket Name	Purpose	Key Configuration
`g-france-vm-project-input`	Ingestion	30-day lifecycle rule, versioning enabled
`g-france-vm-project-output`	Long-term storage	7-year lifecycle rule, versioning enabled
`g-france-vm-project-terraform-files`	Terraform state	Standard, no lifecycle rules

This setup is logical and well-structured for a data pipeline.

2.2. State-File Accuracy and Environment Drift

The Terraform state is correctly configured to use the `g-france-vm-project-terraform-files` bucket. However, the presence of the `reverse_engineering` directory, which contains a `terraformer` dump of the project, reveals a critical disconnect between the declared state and the actual environment. The discrepancies in

machine types and IAM policies prove that the main Terraform configuration is not an accurate representation of the deployed infrastructure. This is a significant operational risk.

2.3. Firestore Schema

The Firestore API is enabled, but no Firestore schema definitions, queries, or application code were found in the scanned files. The "litigation support" collections for deadlines, discovery, and audit logs mentioned in the user's request are not defined at the infrastructure level and must be managed at the application level.

---

3. WORKSPACE & API PERMISSIONS

The security posture of the project is a major concern. The principle of least privilege is not consistently applied, and key security components are managed manually.

3.1. Service Account and OAuth Scope Review

- **Manual Management:** The service accounts `orchestrator-sa` and `pdf-worker-sa` are **not defined** in any Terraform file, indicating they were created manually. This is a significant gap in infrastructure-as-code practices.
- **Undocumented Service Account:** An undocumented service account, `image-worker-sa`, was discovered with GCS permissions.
- **Overly Broad Scopes:** All VMs are configured with the `cloud-platform` OAuth scope, granting them broad access to all Google Cloud services, which is a major security risk.
- **`vm4-seed` Vulnerability:** `vm4-seed` uses the default Compute Engine service account, which has project-wide editor permissions by default. This is a critical vulnerability that should be remediated immediately.

3.2. Specific Access Levels

While the VM scopes are broad, the investigation into the GCS IAM policies revealed more granular permissions on the storage buckets:

Service Account	GCS Roles
`orchestrator-sa`	`roles/storage.objectViewer`
`pdf-worker-sa`	`roles/storage.objectCreator`, `roles/storage.objectViewer`, `roles/storage.objectAdmin`

Service Account	GCS Roles
`image-worker-sa`	`roles/storage.objectCreator`, `roles/storage.objectViewer`

These granular permissions are a positive finding, but they are undermined by the broad `cloud-platform` scope at the VM level.

3.3. Domain-Wide Delegation and Token Storage

No evidence of Domain-Wide Delegation, Google Workspace API usage (Gmail, Drive, Calendar), or token storage protocols was found in the infrastructure files. This functionality, if it exists, is likely managed at the application level, and its security could not be audited.

---

4. FUNCTIONAL SUMMARY

The `g-france-vm-project` is a data processing pipeline that ingests files from an `input` GCS bucket, processes them using a "workhorse" VM (`vm2-worker`), and stores the results in an `output` bucket. The workflow is likely coordinated by `vm1-orch`.

However, the project's operational continuity is at risk due to a combination of factors: * **Configuration Drift:** The infrastructure's actual state does not match its declared state in Terraform. * **Manual Processes:** Critical components like service accounts and the failover mechanism are manually managed. * **Security Vulnerabilities:** The use of broad OAuth scopes and the default service account on `vm4-seed` creates significant security risks.

Recommendations:

1. **Remediate Environment Drift:** Use `terraformer` to import the existing infrastructure into the main Terraform configuration and create a single source of truth.
2. **Implement Least Privilege:** Replace the `cloud-platform` scope on all VMs with more granular scopes. Create dedicated, least-privilege service accounts for each VM, including `vm4-seed`.
3. **Automate Failover:** Implement an automated failover mechanism for `vm1-orch` to reduce the recovery time objective.
4. **Manage all Resources with Terraform:** Bring the service accounts and any other manually created resources under Terraform management.
5. **Audit Application-Level Permissions:** Conduct a separate audit of the application code to assess the security of any Google Workspace integrations and Firestore data access.