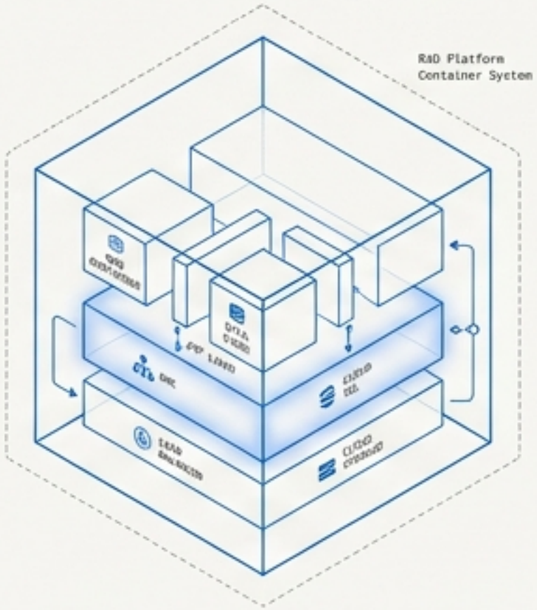
OpenEMR on Google Cloud Platform

Architecture, Implementation, and Optimization Strategy for the RAD Platform.



Module: modules/OpenEMR

Scope: Architecture Analysis & Roadmap

Version Focus: OpenEMR rel-704 / Alpine 3.20

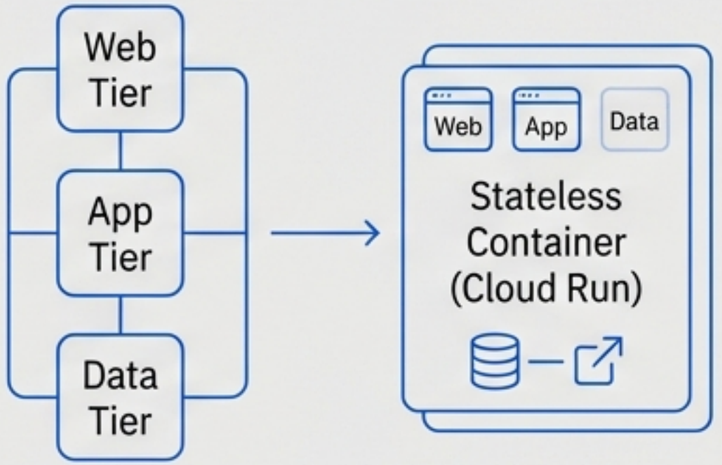
Modernizing the Monolith

The OpenEMR module deploys a scalable, containerized instance of OpenEMR on Google Cloud Run (Gen 2). This implementation bridges the gap between legacy requirements and modern serverless infrastructure.

The Core Strategy

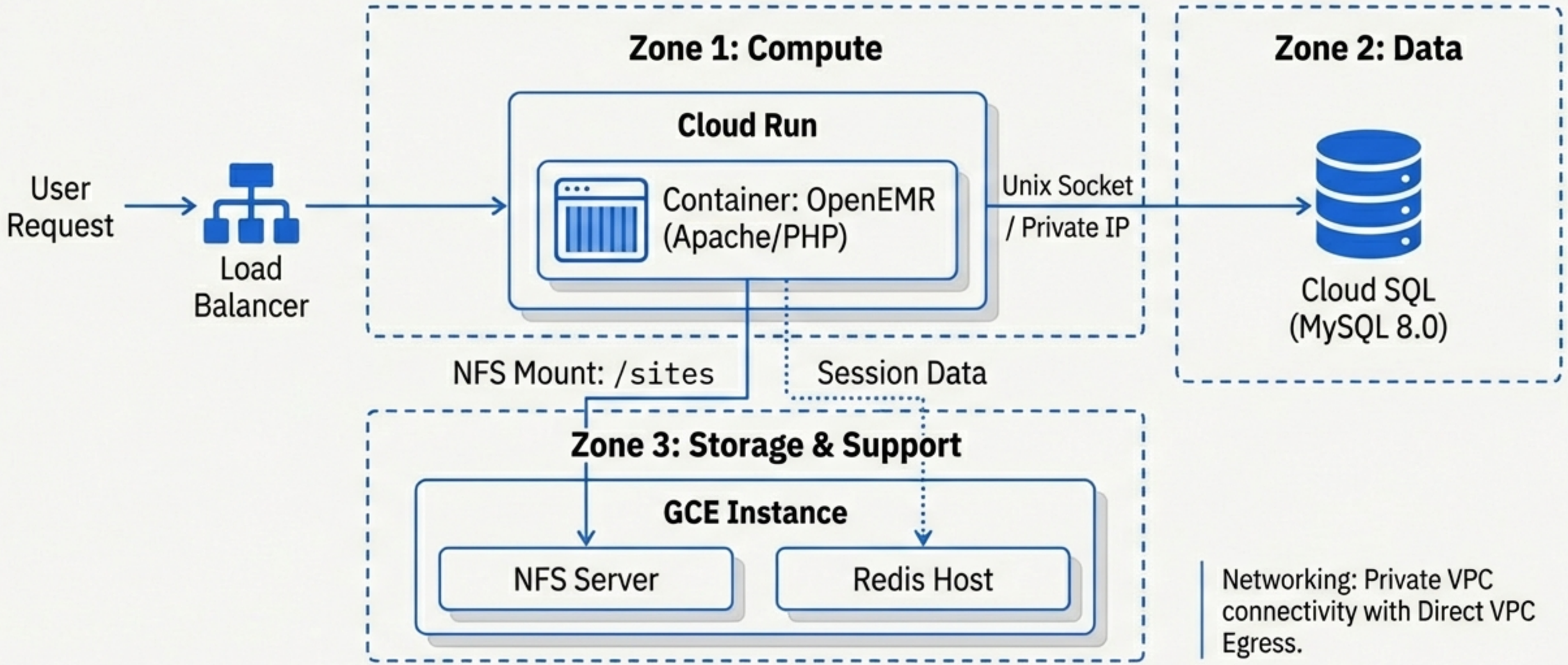
- **Wrapper Architecture**: Utilizes "`CloudRunApp`" to provision standard infrastructure while wrapping OpenEMR-specific configurations.
- **State Management**: Solves the persistence challenge by offloading the database to Cloud SQL and the "sites" directory to an external NFS server (GCE).
- **Session Handling**: Leverages Redis (hosted on NFS infrastructure) to manage session stickiness in a containerized environment.

Key Outcome



A functional 3-tier web application architecture adapted for stateless containers.

System Architecture



Compute Layer: Cloud Run Configuration

Optimizing container runtime for a heavy PHP application.

Container Specs

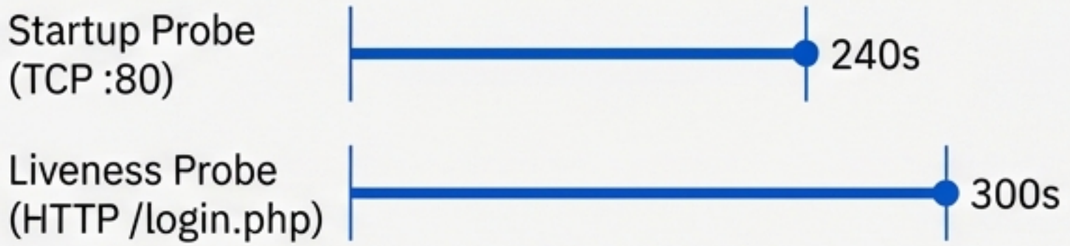
Base Image	`alpine:3.20 / PHP 8.3 / Apache`
vCPU	`2 (2000m)`
Memory	`4Gi`
Concurrency	`max_instances: 1`
Scaling	`min_instances: 1`

Architectural Decisions

Concurrency Control: "`max_instances`" is restricted to 1. This addresses legacy concurrency concerns and potential session stickiness issues.

Cold Boot Mitigation: "`min_instances`" set to 1 ensures a warm start, preventing latency spikes during initial access.

Resiliency (Probes)



Persistence Strategy: Solving the State Problem

Bridging stateless containers with stateful requirements.

Database State

Google Cloud SQL (MySQL 8.0)

Connection: Primary via Unix Socket (`/cloudsql`); , **Secondary** via Internal IP (`DB_HOST`).

File Storage (The Critical Path)

NFS Volume Mount

Path: `/var/www/localhost/htdocs/openemr/sites`

Infrastructure: External GCE Instance identified via `get-nfsserver-info.sh`

Note

Why NFS?

OpenEMR requires persistent local file write access for patient documents. The variable `nfs_enabled` defaults to `true` to support this requirement across container restarts.

IAM & Security Model

Runtime Identity

cloudrun-sa

Used by OpenEMR container.

- `roles/secretmanager.secretAccessor` (Access: `OE_PASS`, `DB_PASSWORD`)
- `roles/storage.objectAdmin` (Manage GCS buckets)

CI/CD Identity

cloudbuild-sa

Used by Cloud Build triggers.

- `roles/run.developer` (Deploy revisions)
- `roles/iam.serviceAccountUser`

Network Access

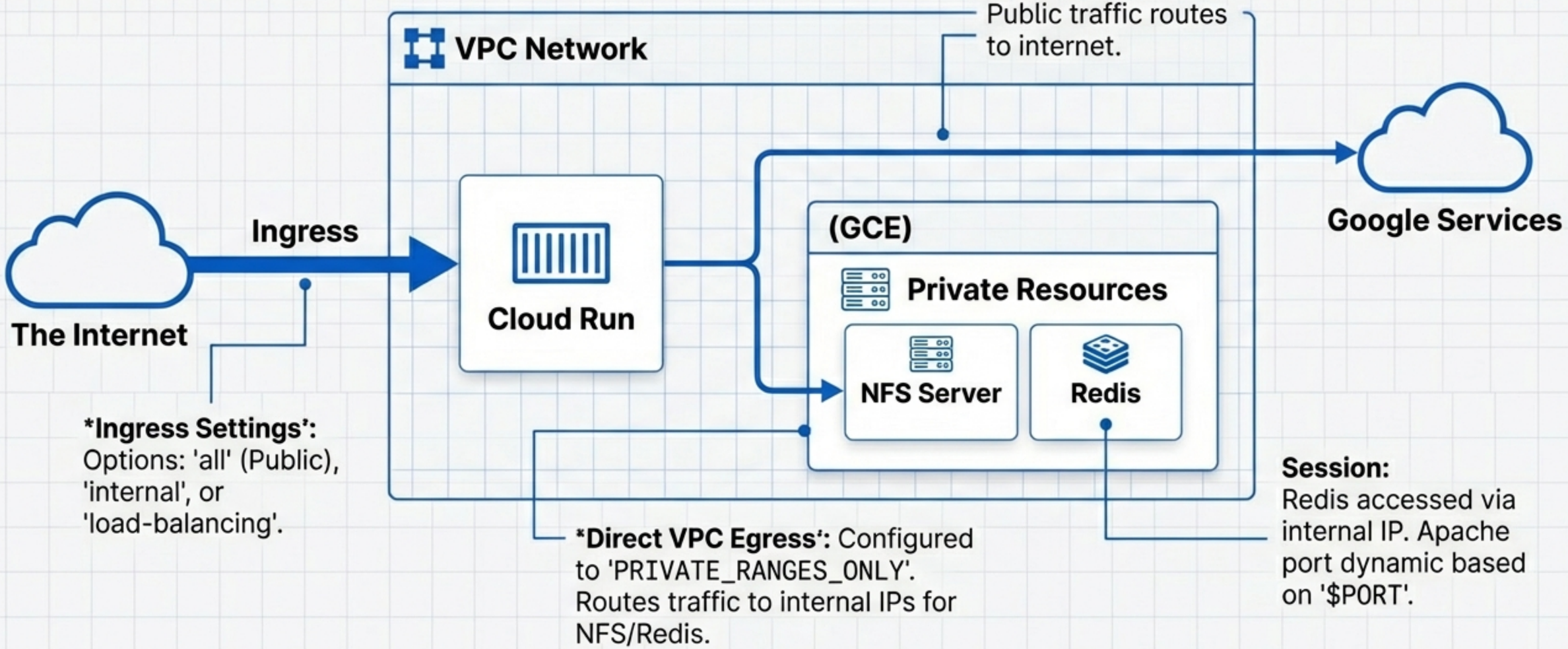
Invoker

Defaults to 'allUsers' (Public).

ⓘ Note: Controlled via 'public_access' variable.

Networking & Connectivity

Simplified network topology and traffic flow configuration.



Implementation: The Build Pipeline

Source: `scripts/openemr/Dockerfile`

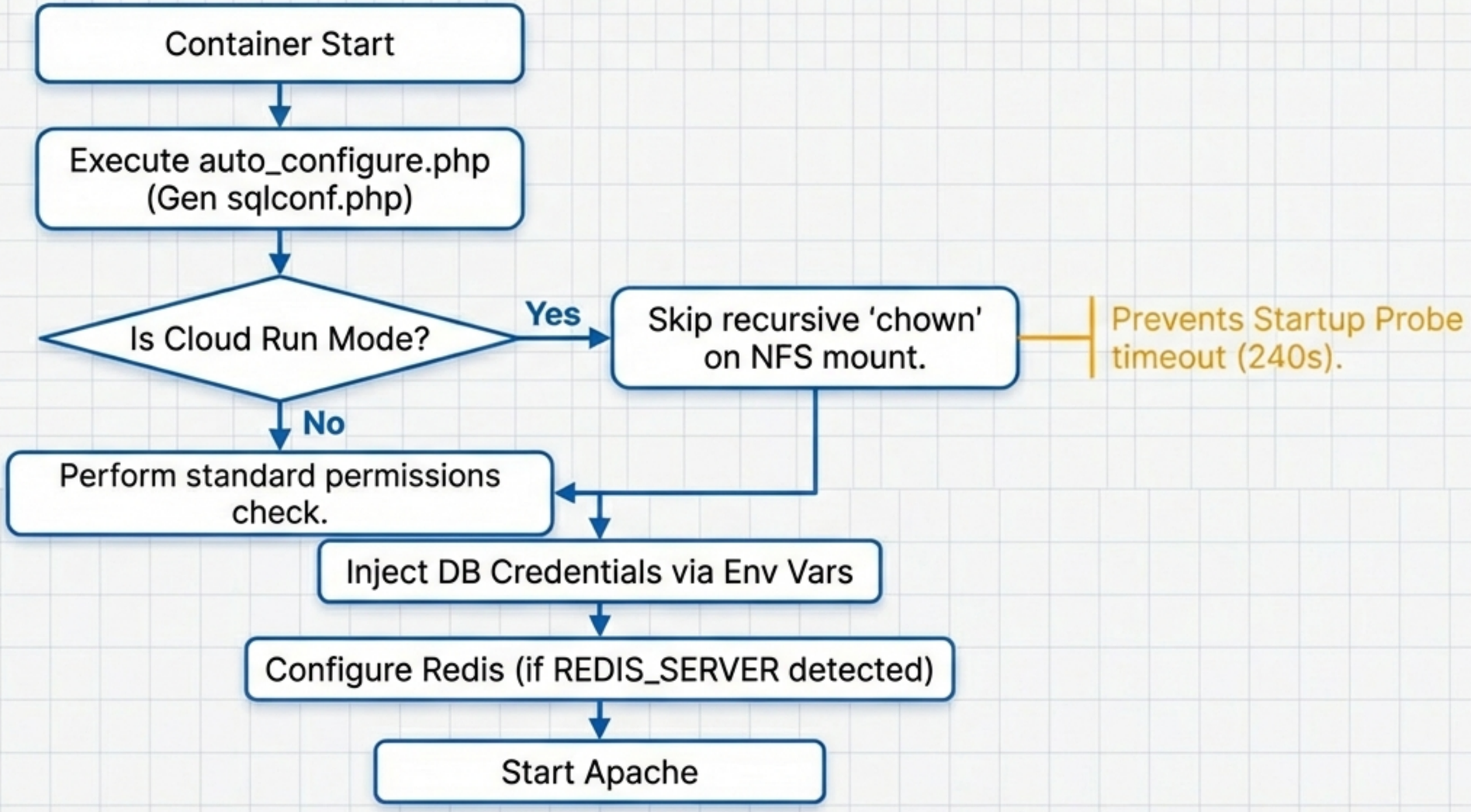
Security Posture

Process runs as non-root user **apache** (UID 1000).

Logs redirected to **/dev/stderr** and **/dev/stdout** for Cloud Logging.

Runtime Logic & Wrapper Architecture

Script: `scripts/openemr/openemr.sh`



Infrastructure Initialization (nfs-init)

Prepping the environment before traffic hits.

1

Ownership

Set `'sites'` directory to UID 1000 (User apache).

2

Backup Retrieval

Download backup from GCS or GDrive (if `BACKUP_FILEID` exists).

3

Restore

Unzip and restore data to NFS volume.

4

Re-Config

Regenerate `'sqlconf.php'` to match current DB creds.

Significance: This job decouples heavy storage operations from the critical path of the application startup, solving the timeout issue.

Configuration & Observability

Key Terraform Variables

Variable	Default	Description
`nfs_enabled`	`true`	Critical toggle for NFS volume mounting.
`database_type`	`MYSQL_8_0`	Enforces compatibility version.
`backup_source`	`gcs`	Restore source (gcs or gdrive).
`ingress_settings`	`all`	Controls network visibility.

Observability

****Logging****

Apache/PHP logs piped to standard streams.

```
[01-Jan-2024 09:00:01 UTC] PHP Notice:  Undefined index: user
in /var/www/html/openemr/library/globals.inc.php on line 234
127.0.0.1 - - [01/Jan/2024:09:00:01 +0000] "GET /openemr/i
nterface/login/login.php HTTP/1.1" 200 4523 "-" "Mozilla/5.0
(X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/120.0.0.0 Safari/537.36"
```

****Secrets****

Managed via Secret Manager.

- `OE_PASS`
- `DB_PASSWORD`

Currently static after creation.

Roadmap: Architectural Enhancements

Migration to Managed Services (PaaS) in IBM Plex Sans

Current State

GCE Instance

NFS Server

Redis Process

Self-managed VM.
High maintenance.

Enhance Reliability

Future State

Cloud Filestore (Basic Tier)

Replaces NFS. Managed service, better SLA, easy scaling.

Cloud Memorystore

Replaces Redis. Automated patching, high availability.

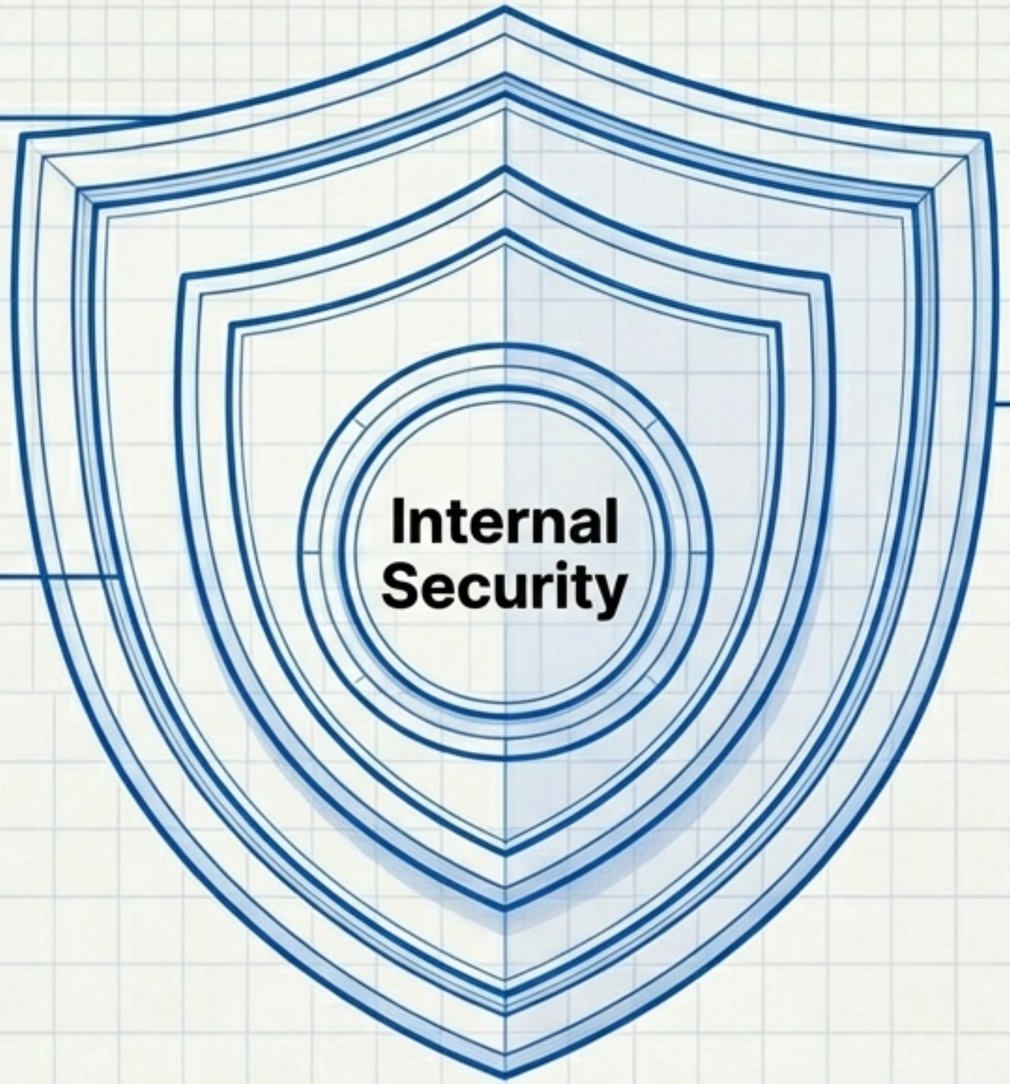
Roadmap: Security Hardening

Perimeter Defense

Cloud Armor (WAF): Mitigate OWASP Top 10 attacks (Critical for public ingress).

Access Control

Identity-Aware Proxy (IAP): Enforce Zero-Trust access to login screen.



Internal Security

- Automated Secret Rotation (`OE_PASS`, `DB_PASSWORD`).
- Least Privilege: Downgrade `'objectAdmin'` to `'objectCreator'` for storage.

Roadmap: Operational Excellence

Structured Logging

Update `'openemr.conf' LogFormat` to output JSON. Enables advanced filtering/parsing in Cloud Logging.

Automated Backups

Implement `'Cloud Scheduler + Cloud Run Job'`. Periodically dump MySQL & `'sites'` to GCS (vs current restore-only).

Performance Tuning

Optimize `'auto_configure'` to reduce the `240s` startup probe delay. Faster rollouts.

Summary & Strategic Recommendation

Assessment

The current module successfully modernizes OpenEMR by wrapping a legacy application in a containerized, serverless shell. The **'Wrapper Architecture'** effectively handles the stateful **requirements via NFS** and **Cloud SQL**.

The Verdict

Functional Baseline Achieved.

- **Priority**: Migrate to **Cloud Filestore** for enterprise reliability.
- → **Security**: Enable **Cloud Armor** for public deployments.