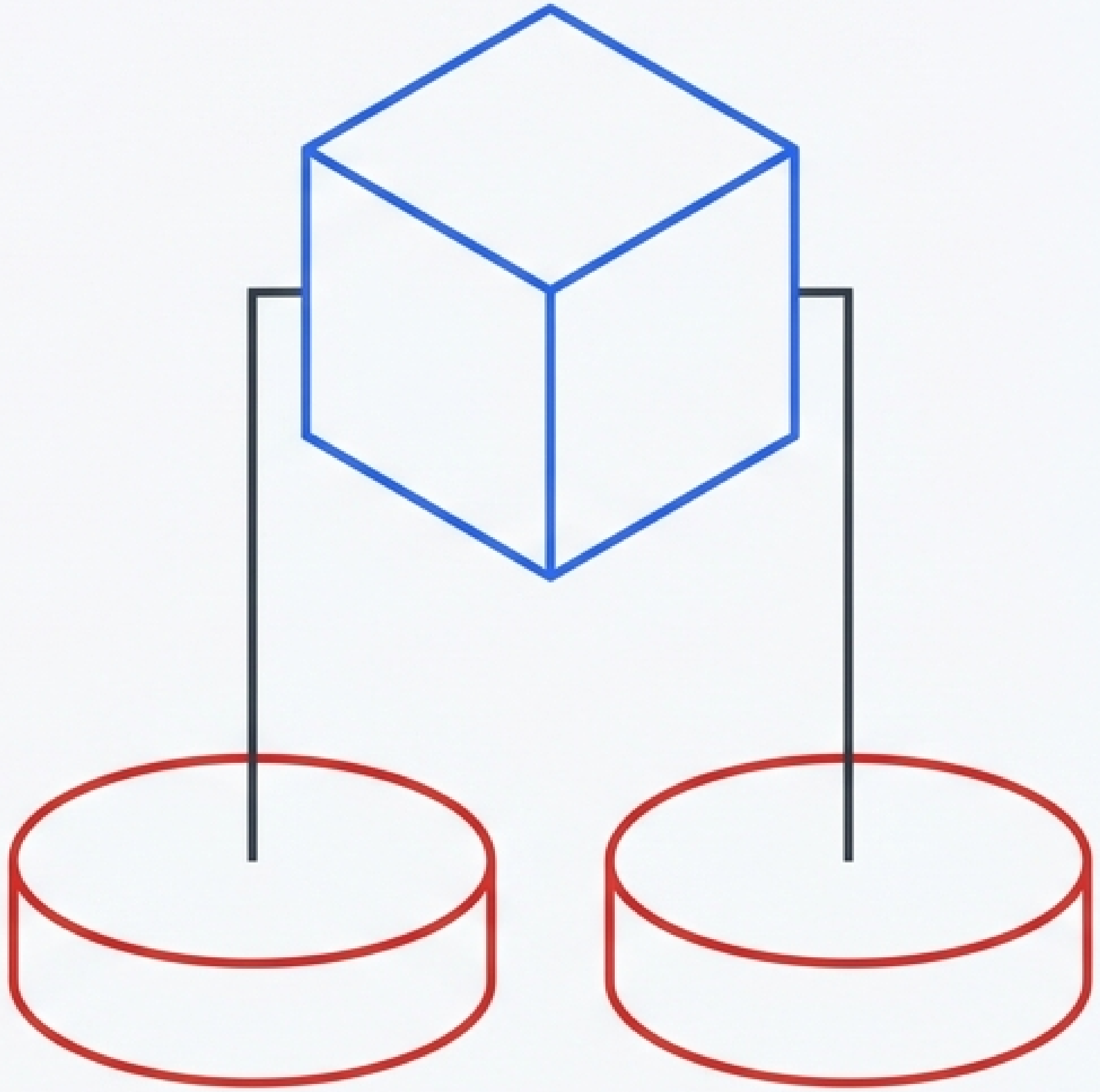
# Odoo on Google Cloud Platform

## RAD Platform Module Analysis: Architecture, Initialization, and Configuration

`modules/Odoo`

A technical breakdown of the implementation.



## Serverless Architecture for a Stateful ERP

The 'modules/Odoo' module is a specialized wrapper deployment. It leverages the 'modules/CloudRunApp' foundation to host Odoo Community Edition on Google Cloud Run.

### The Challenge

Odoo is inherently stateful. It requires specific filesystem structures and database preparations before it can boot. This conflicts with standard stateless container paradigms.

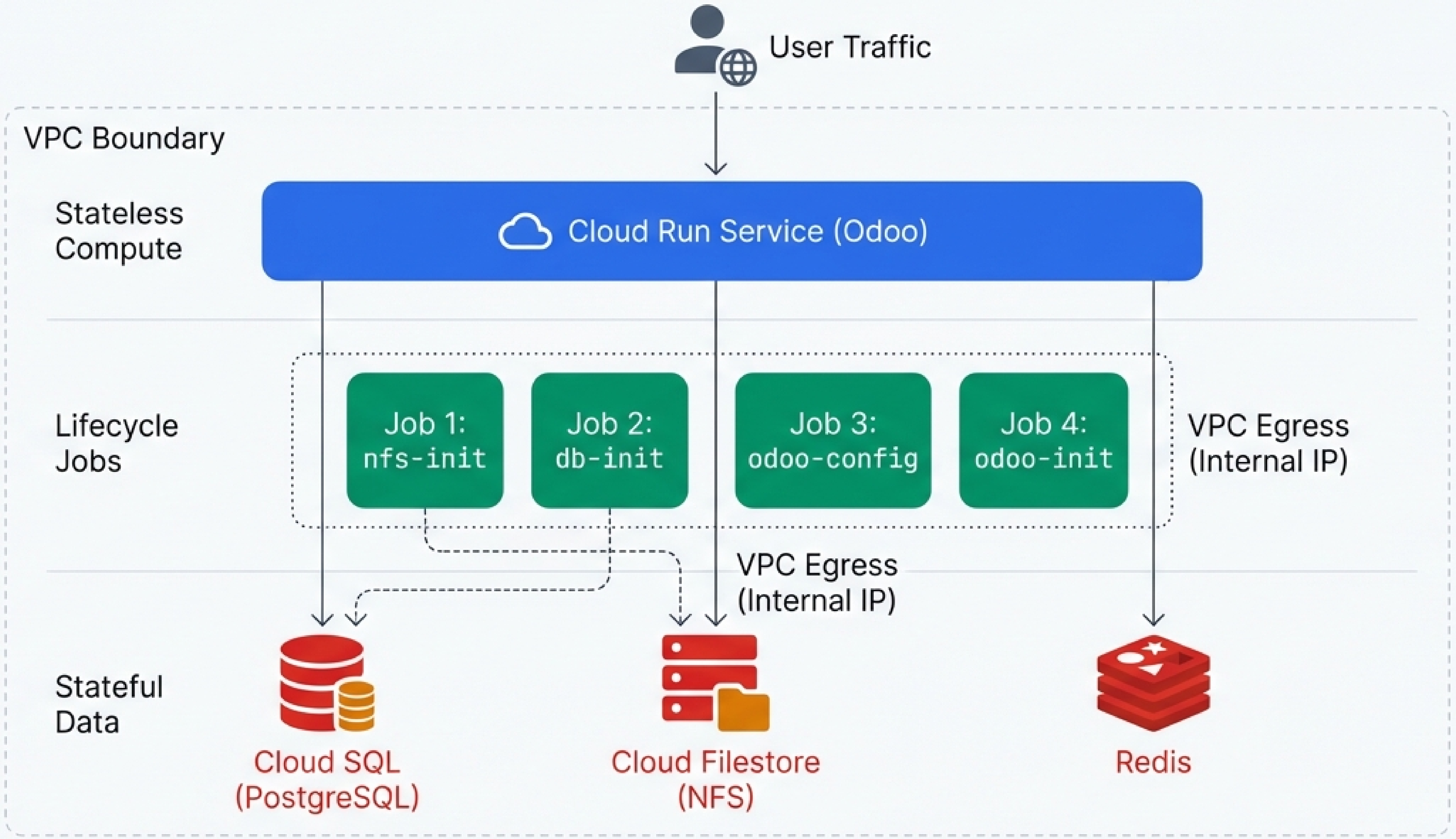
### The Solution

A novel implementation utilizing a 4-step Cloud Run Job initialization sequence to prepare the environment (NFS, DB, Config) prior to the application start.

## Key Components Checklist

- **Compute:** Cloud Run (Gen 2) `JetBrains Mono`
- **Database:** Cloud SQL (PostgreSQL 15)
- **Storage:** Cloud Filestore (NFS)
- **Optional:** Redis for session caching

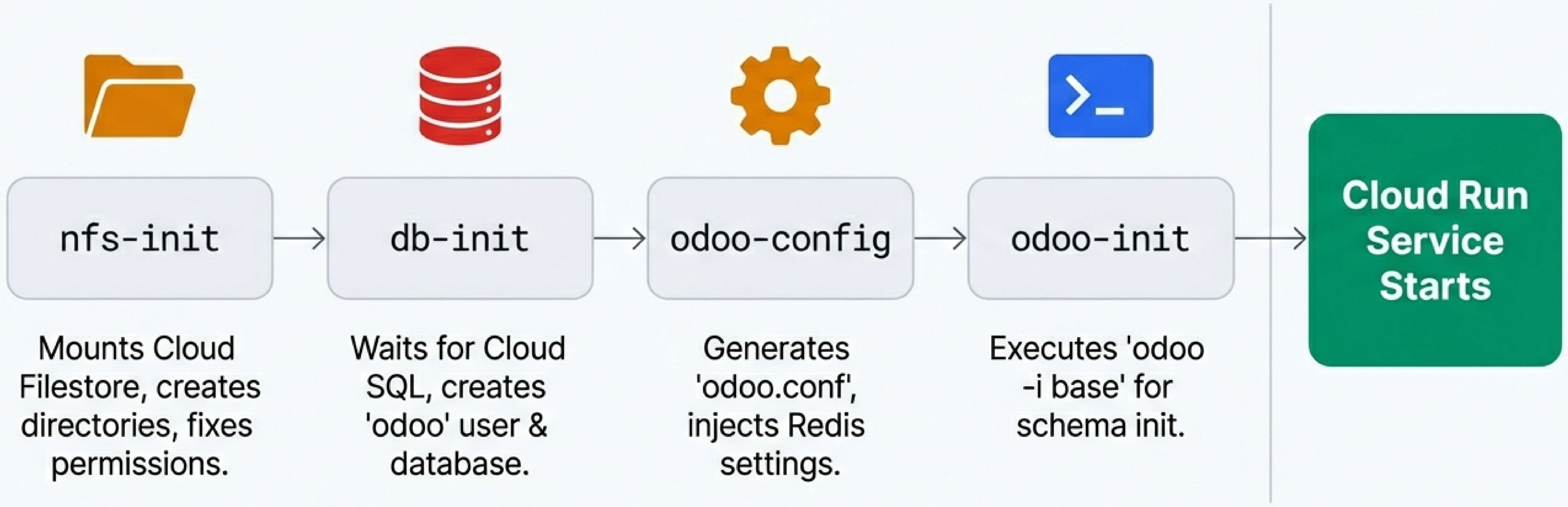
# High-Level System Architecture



Architectural separation of Stateless Compute from Stateful Persistence.

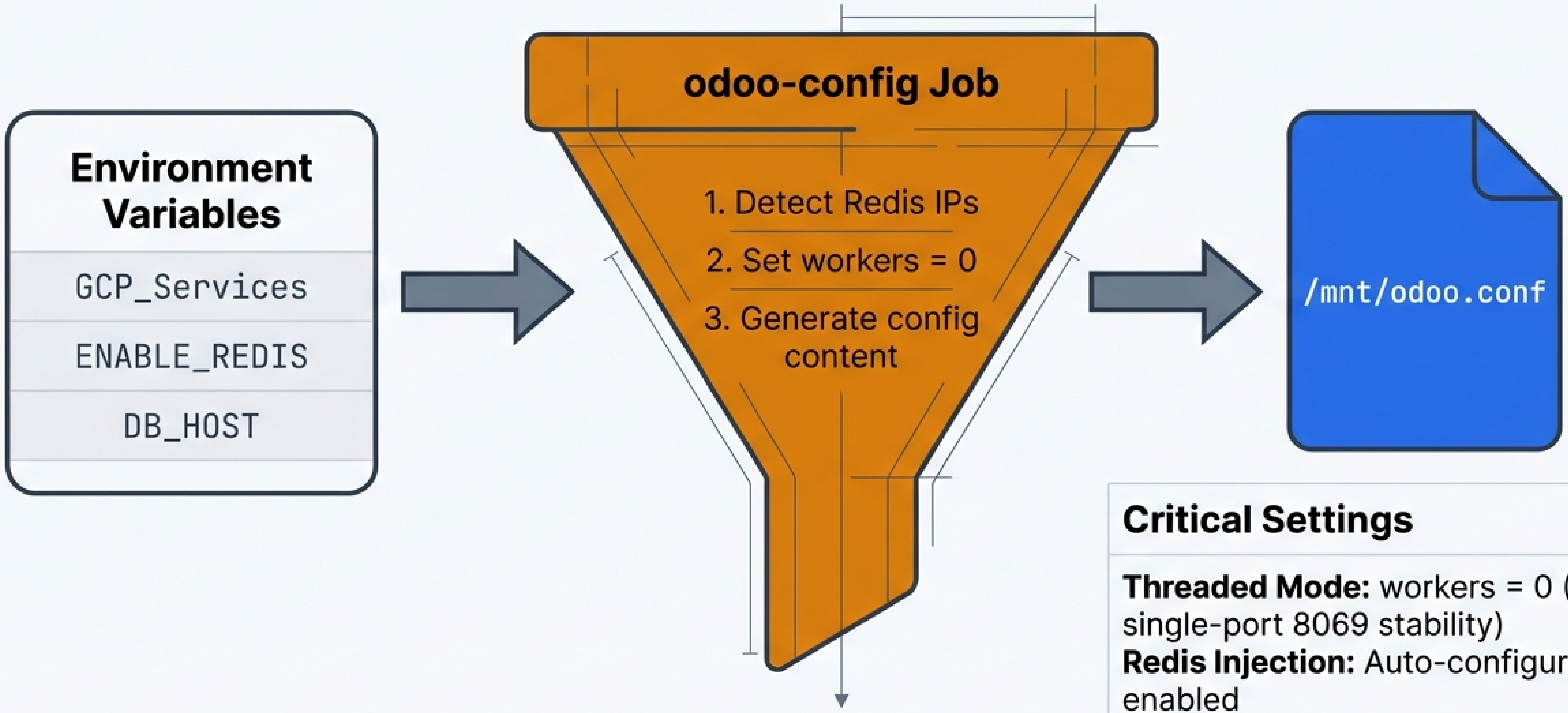
## Solving the Stateless Paradox: The Initialization Sequence

Sequential Cloud Run Jobs executed prior to service start


nfs-init
Mounts Cloud Filestore, creates directories, fixes permissions.
db-init
Waits for Cloud SQL, creates 'odoo' user & database.
odoo-config
Generates 'odoo.conf', injects Redis settings.
odoo-init
Executes 'odoo -i base' for schema init.
Cloud Run Service Starts

# Dynamic Configuration Generation

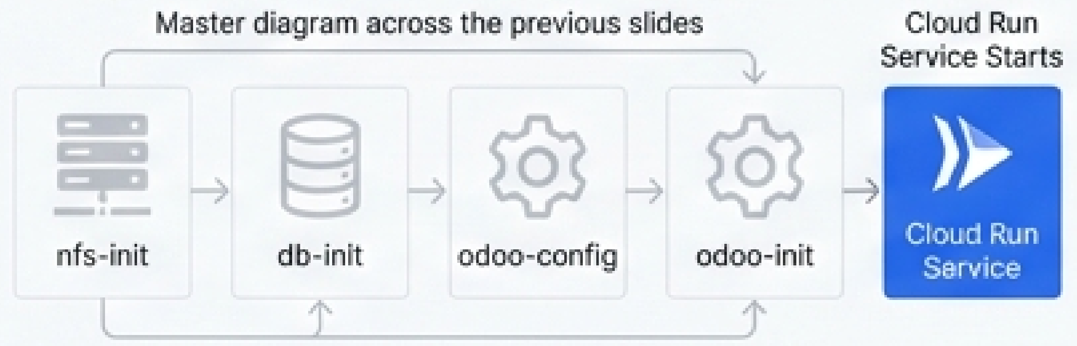
Deep Dive: The 'odoo-config' Job



**Critical Settings**

**Threaded Mode:** workers = 0 (Ensures single-port 8069 stability)
**Redis Injection:** Auto-configured if enabled

# Compute Layer: Cloud Run (Gen 2)



## Resource Specifications

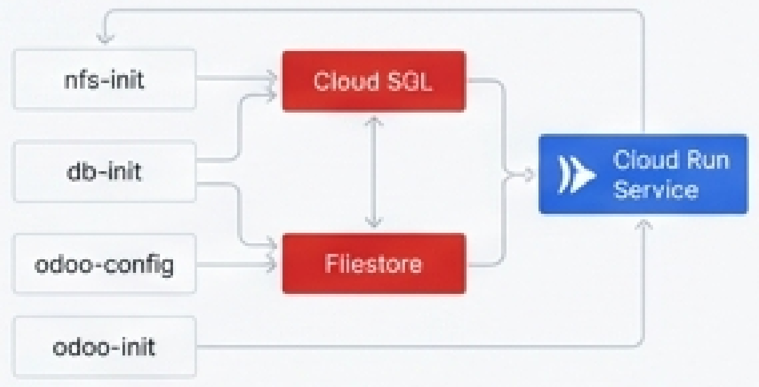
- **Resources:** 2 vCPU / 4Gi Memory (Default)
- **Scaling:** 0-3 instances (**Scale-to-zero** enabled)
- **Networking:** Direct VPC Egress enabled

## Command Override Logic

```
Command: /bin/bash -c
Args: check_config && odoo -c
/mnt/odoo.conf
```

The container checks for the existence of the configuration file generated by the init jobs before attempting to boot the application.

# Persistence Layer: SQL & NFS



Master diagram across the previous slides

## Cloud SQL

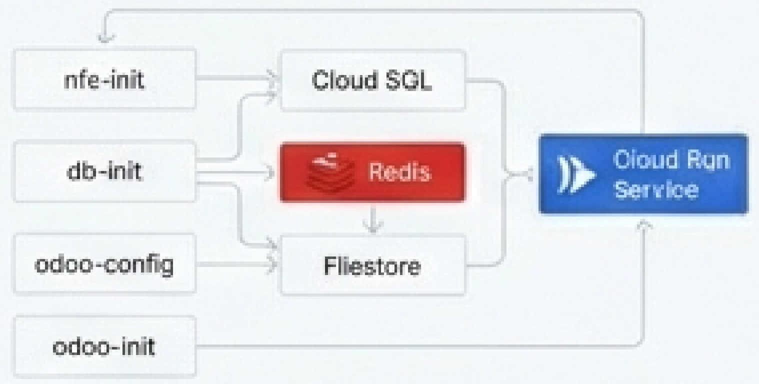
- **Engine:** PostgreSQL 15
- **Connection:** Unix Socket via Volume Mount (/cloudsql)
- **Discovery:** External script identifies existing SQL instances

## Cloud Filestore (NFS)

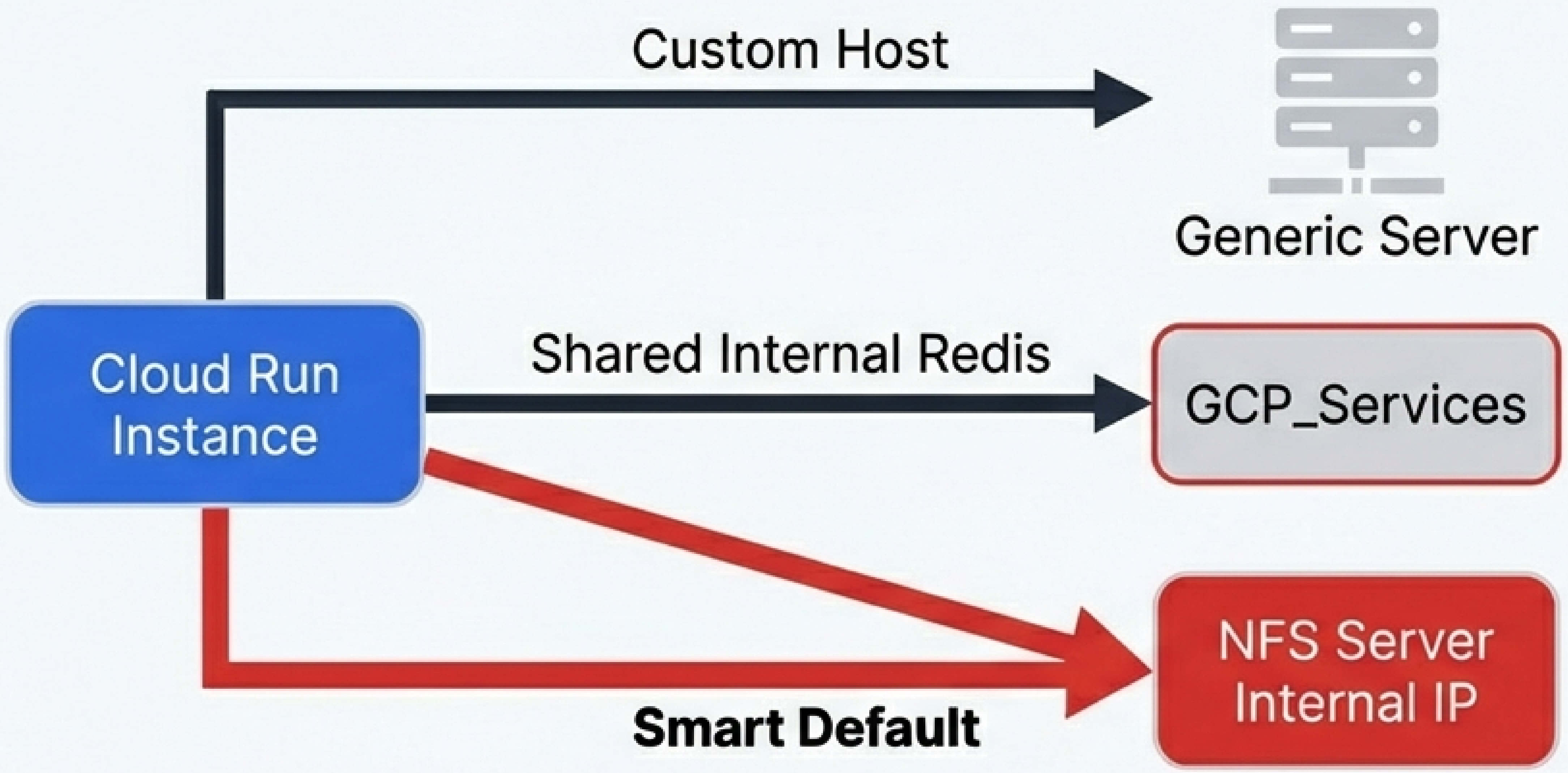
- **Mount Point:** /mnt (Mapped to /var/lib/odoo)
- **Criticality:** HIGH
- **Why?** Shared 'filestore' (attachments) and 'sessions' are mandatory for running multiple Odoo replicas. Without NFS, data consistency breaks across instances.

NotebookLM

# Caching & Performance: Redis Integration



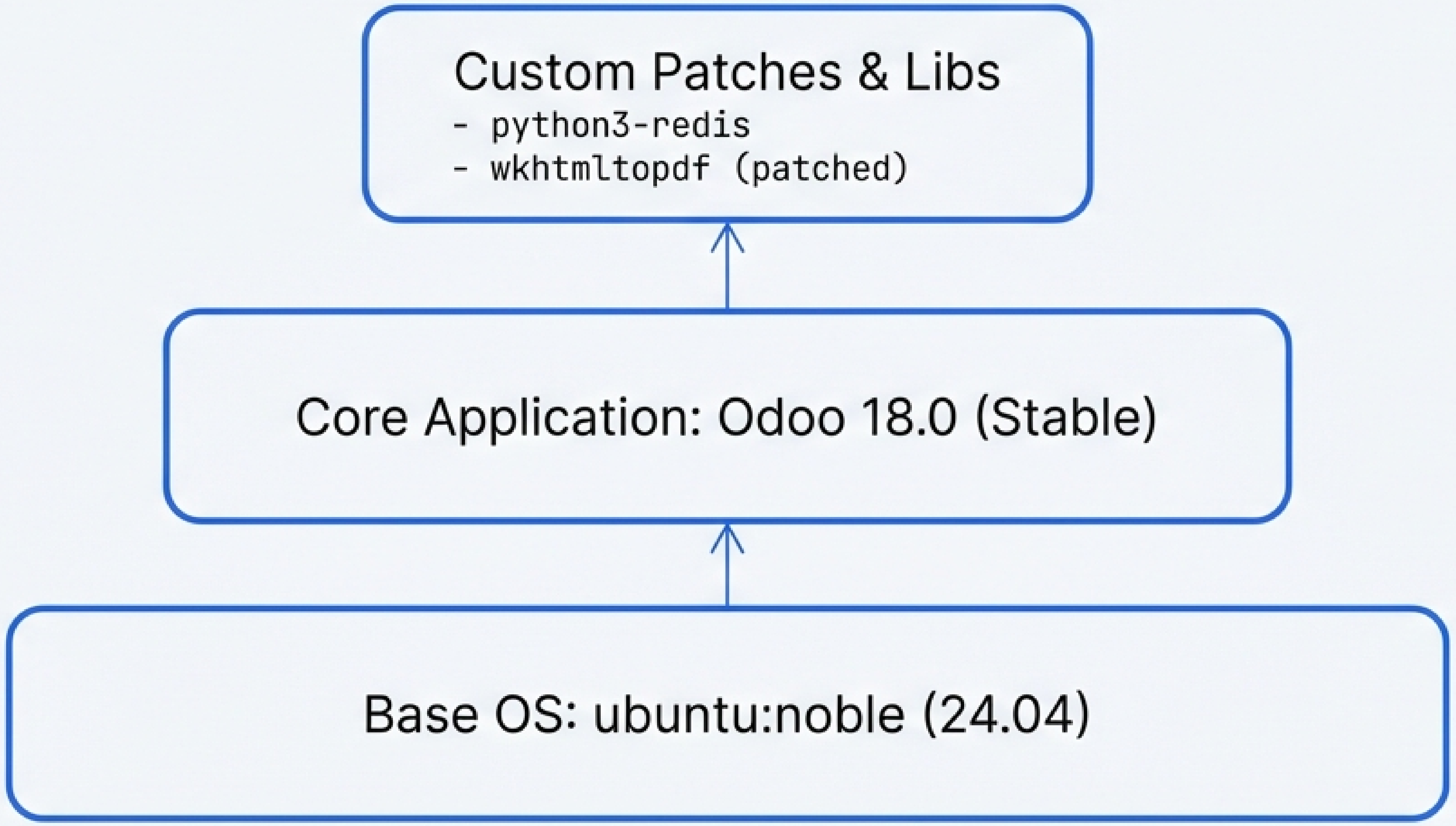
Master diagram across the previous slides



If 'enable_redis' is true but host is unset, defaults to co-located NFS server.

**Benefit:** Offloads session management from the filesystem to memory for improved performance.

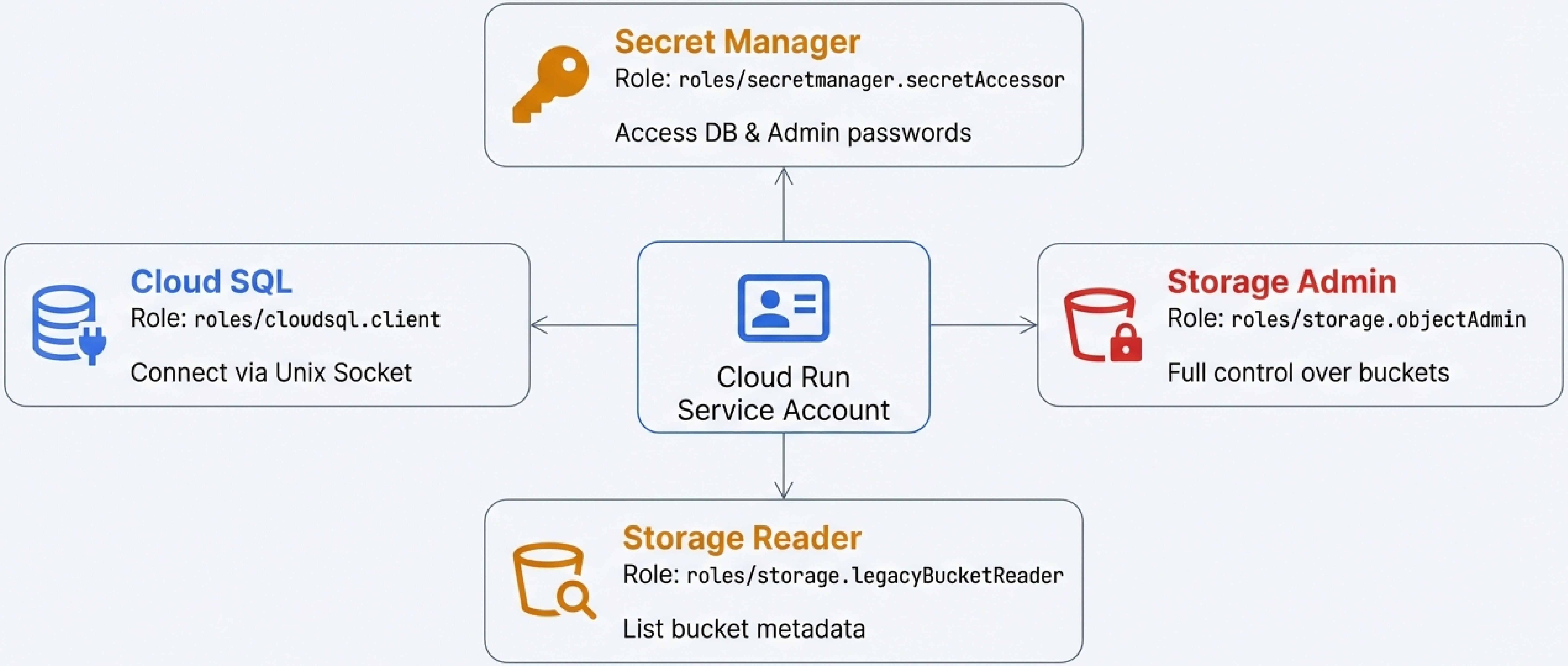
# The Artifact: Custom Container Image


Custom Patches & Libs
- python3-redis
- wkhtmltopdf (patched)
Core Application: Odoo 18.0 (Stable)
Base OS: ubuntu:noble (24.04)

## Build Details

- **Strategy:** `image_source = 'custom'`
- **Verification:** Flexible SHA verification for nightly builds.
- **Dockerfile Source:** `scripts/odoo/Dockerfile`

## IAM & Least Privilege Security

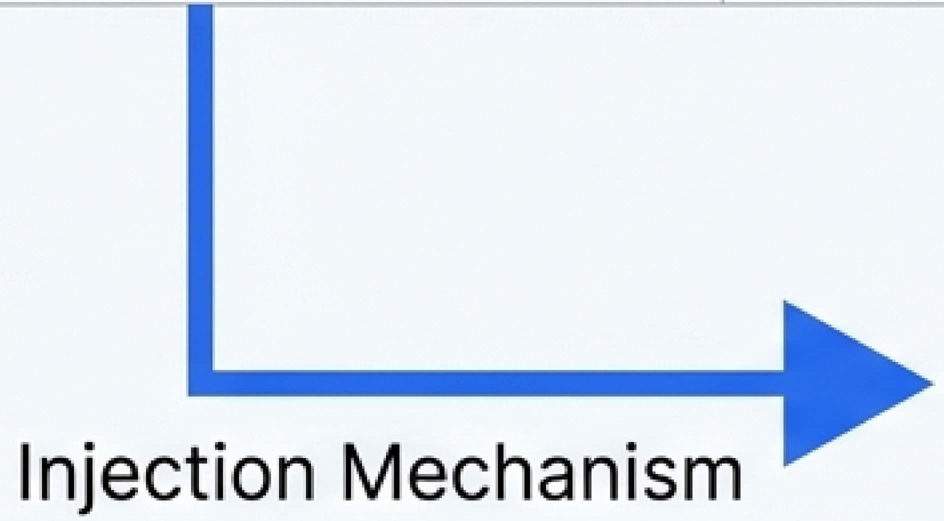

Secret Manager
Role: roles/secretmanager.secretAccessor
Access DB & Admin passwords
Cloud SQL
Role: roles/cloudsql.client
Connect via Unix Socket
Cloud Run Service Account
Storage Admin
Role: roles/storage.objectAdmin
Full control over buckets
Storage Reader
Role: roles/storage.legacyBucketReader
List bucket metadata

Strict adherence to Least Privilege principles.

# Secrets Management

Decoupling credentials from codebase

| | |
|---|---|
| `ODOO_MASTER_PASS` | The master administrator password for Odoo. Critical for database management. |
| `DB_PASSWORD` | Credentials for the PostgreSQL database connection. |



Accessed at runtime via Service Account Identity (Google Secret Manager). No hardcoded values in Terraform.

## Orchestration Logic: odoo.tf

```
resource 'google_cloud_run_v2_service' 'odoo' {
  template {
    containers {
      command = ['/bin/bash', '-c']
      args    = ['check_config && run_odoo']
      volume_mounts {
        name = 'cloudsql'
        path = '/cloudsql'
      }
    }
  }
}
```

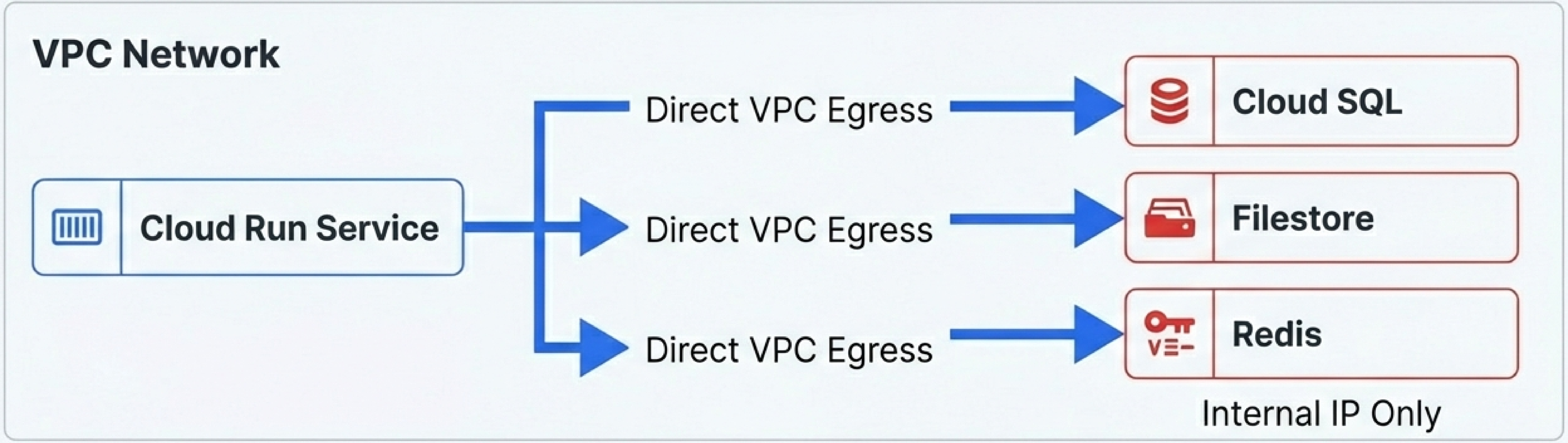
**Entrypoint Override**

Intercepts boot process to ensure config exists.

**Infrastructure Mapping**

Binds the SQL socket and NFS share.

# Networking & Connectivity



## Port Logic

`Service Port: 8069`

Threaded Mode (workers = 0) allows handling XML-RPC and Longpolling on a single port. This simplifies Load Balancer configuration by removing the need for separate routing rules.

# Technical Specifications Summary

**Architecture**

Serverless (Cloud Run) + Stateful Backends (SQL/NFS)

**Version**

Odoo 18.0 on Ubuntu 24.04

**Scaling**

0 to N instances (Auto-scaling)

**Initialization**

4-Stage Job Sequence (nfs, db, config, app)

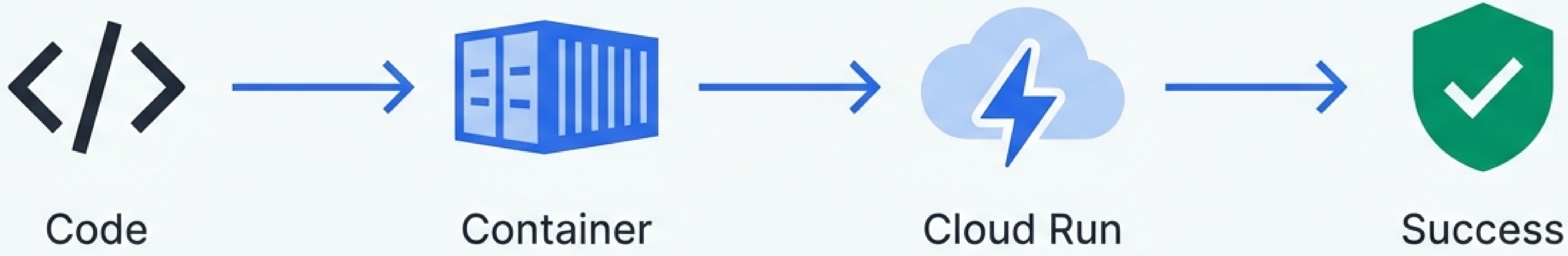
**Security**

IAM-based Auth, Secret Manager, VPC Egress

**Status**

Production-ready implementation of modules/Odoo

# The RAD Platform Advantage



By decoupling application state (SQL/NFS) from runtime (Cloud Run) and bridging them with intelligent initiallization jobs, we achieve a highly scalale, maintenance-free ERP environment.