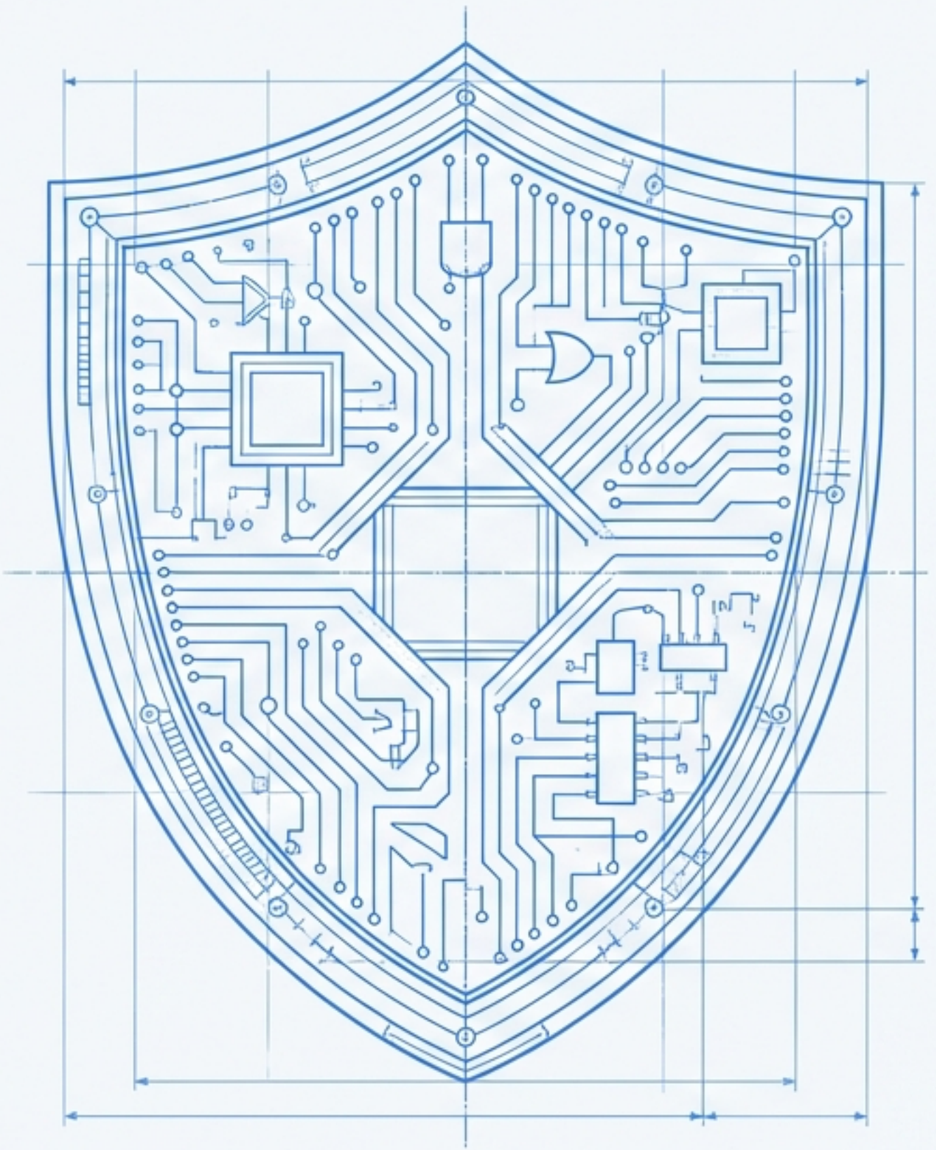
AIRS

AI Incident Readiness Score

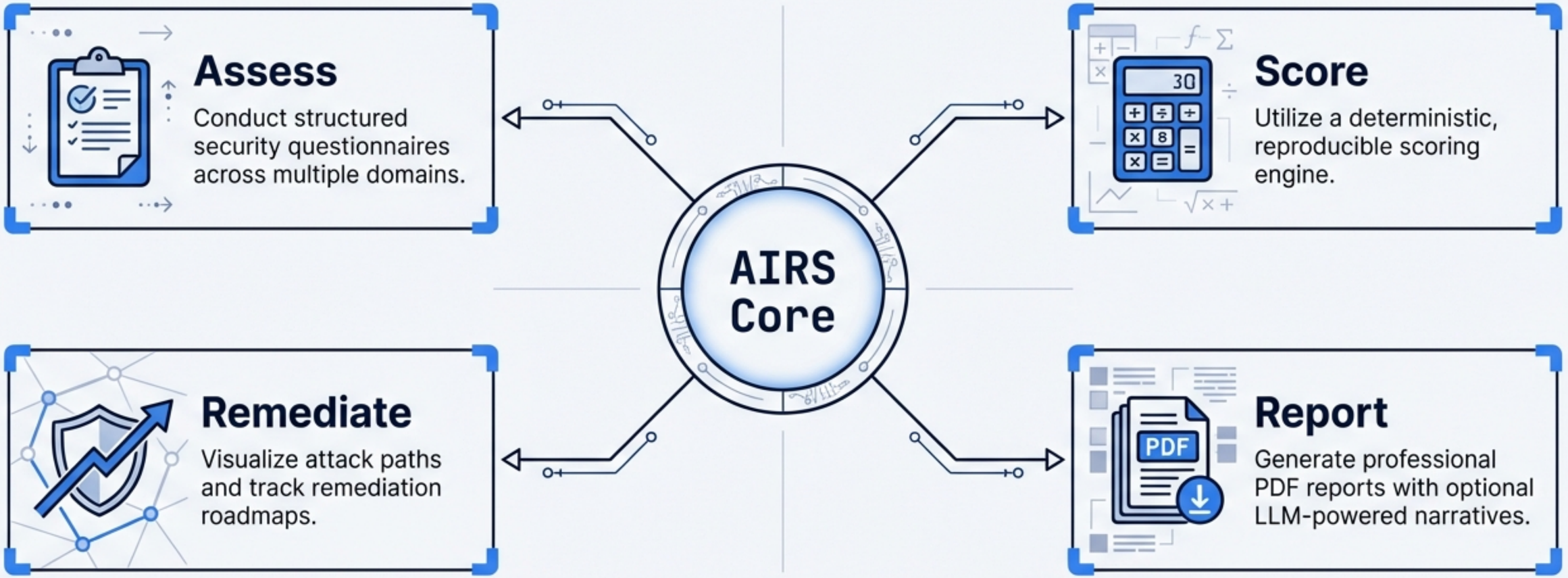
A FastAPI Application for Security Readiness Assessments, Scoring, and Reporting.



OPEN SOURCE | MIT LICENSE | PYTHON 3.11+

A Standardized Platform for AI Security Posture

AIRS is an open-source platform designed to assess, score, and improve the security readiness of AI systems through structured workflows.



The Feature Suite

Security Assessment

Domain-specific questioning to baseline security posture.

Analytics & Attack Paths

Visual analysis of top risks and potential vulnerabilities.

Automated Findings

Rule-based generation of findings with specific recommendations.

Edit & Re-score

Dynamic updates—change answers and recalculate scores instantly.

Remediation Roadmap

Track and manage tasks over time.

React Frontend

Interactive interface for seamless user assessment.

The Hybrical Swiss Blueprint

The Hybrid Engine: Deterministic Logic + Generative Insight

The Generative Layer

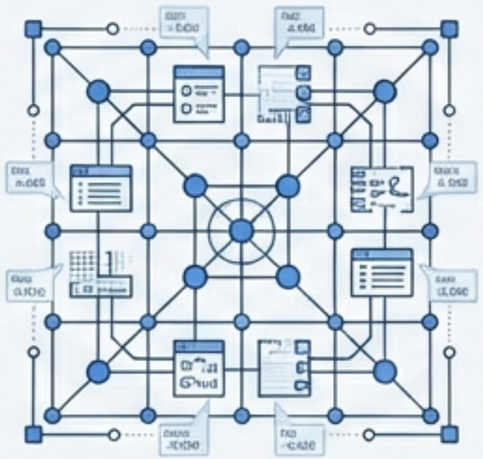
What AI Does

- Executive summary narratives
- Roadmap narrative text
- Natural language insights

Powered by Google Gemini

The Deterministic Core

What AI Does NOT Modify



- Assessment scores
- Finding severity/priority
- Compliance mappings

Computed by code to ensure reproducibility

Technical Architecture & Stack

Frontend Layer

React (TypeScript) • Interactive Assessment Interface

Backend Layer

Python 3.11+ • FastAPI (API Framework) • SQLAlchemy (ORM) • Pydantic (Schemas)

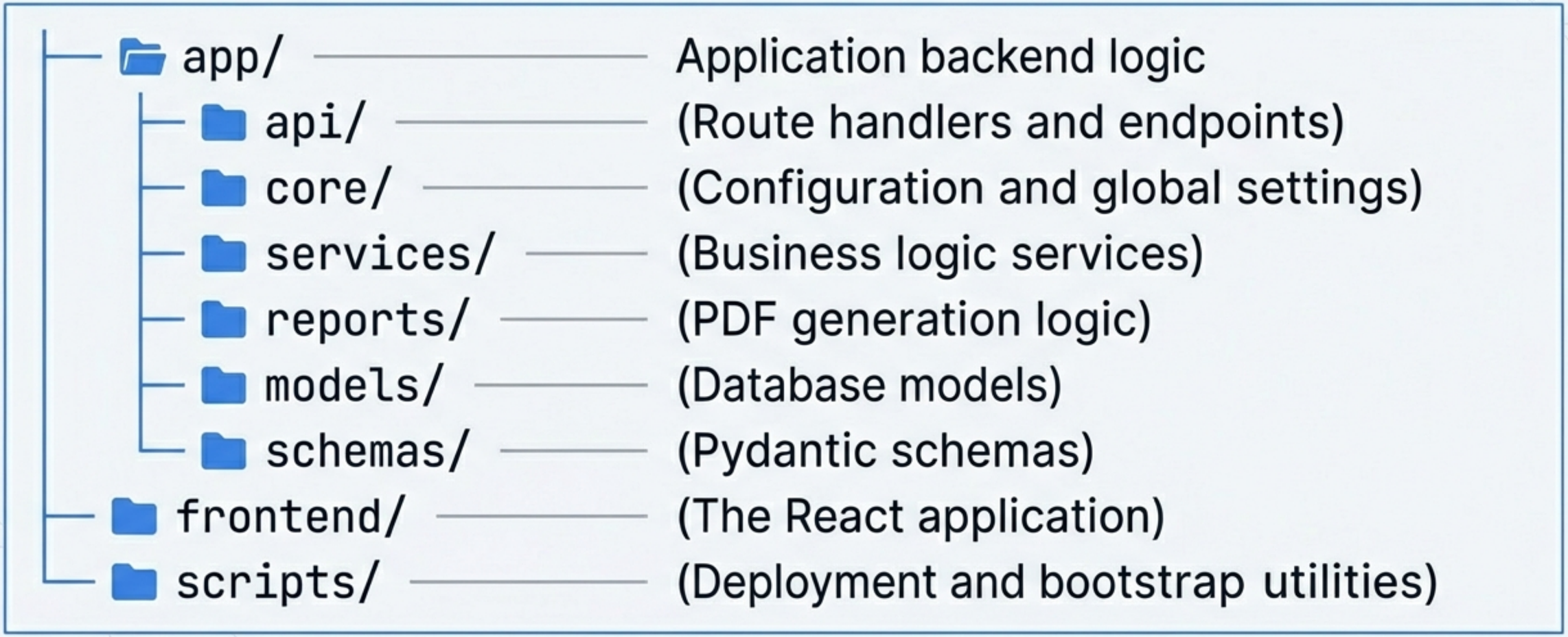
Infrastructure Layer

Docker • Google Cloud Run • Gunicorn + Uvicorn Workers

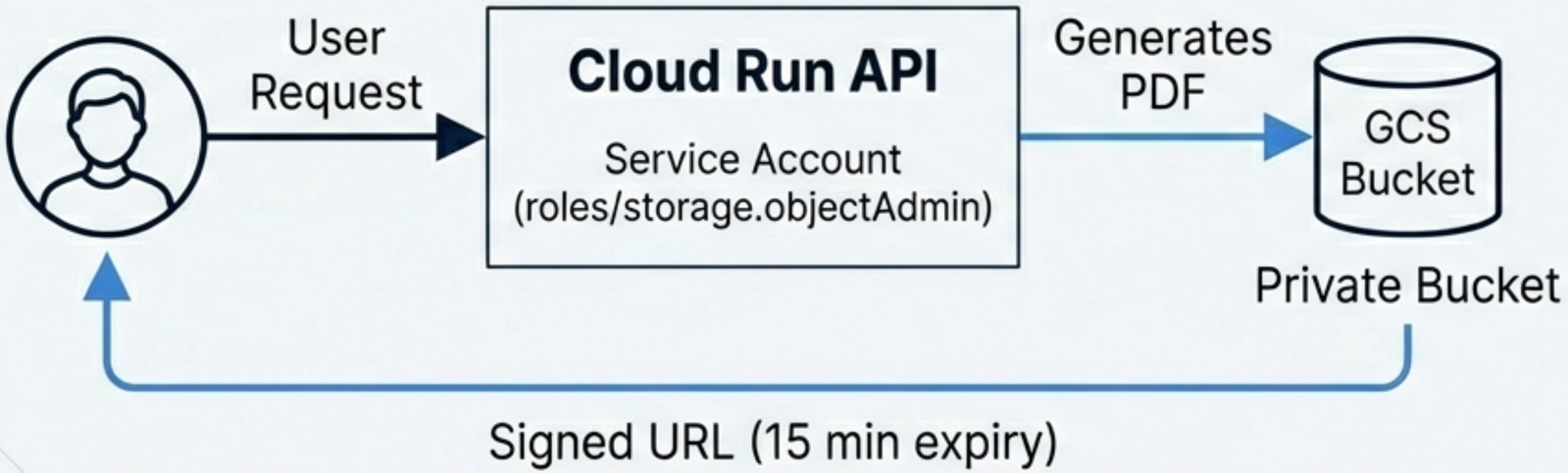
Data Layer

SQLite (Local Dev) • PostgreSQL (Production/Cloud SQL)

Repository Structure



Security Architecture & Data Flow



Safety Controls

- **Least Privilege**: Cloud Run only has Object Admin access.
- **Lifecycle**: Auto-delete rules (e.g., > 365 days).
- **Fallback:** On-the-fly PDF generation if storage fails.

Quick Start: Local Development

```
 # 1. Setup
$ git clone <repo_url>
$ python -m venv venv && source venv/bin/activate

# 2. Install Dependencies
 $ make install

# 3. Configure Env
$ cp .env.example .env

# 4. Run Development Server
 $ make dev
```

Starts Uvicorn with hot-reload at localhost:8000

Production Deployment: Google Cloud Run

Bootstrap

Run `scripts/bootstrap_gcp.sh` to configure project & APIs.

The Runtime Engine

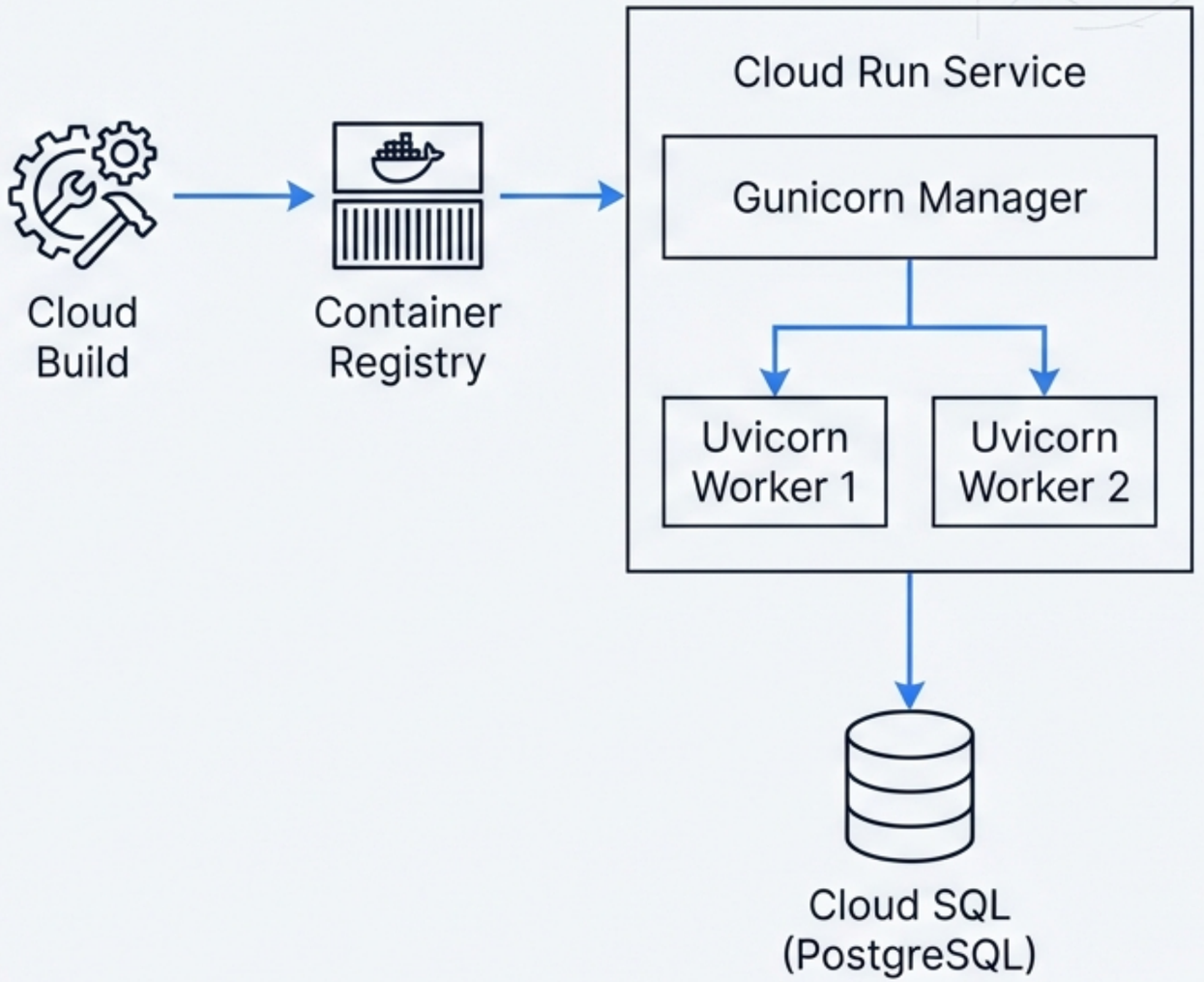
Uses Gunicorn with Uvicorn workers for reliability.

```
gunicorn -k uvicorn.workers.UvicornWorker
app.main:app --workers 2
```

Deployment

Command: `` `make deploy-gcp` ``

Buildpacks auto-detect Python via `requirements.txt`



Configuration & Environment Variables

Variable	Description	Values
`APP_NAME`	Application identifier	Default: `"AIRS"`
`DEBUG`	Toggle debug mode	`true` / `false`
`DATABASE_URL`	Connection string	SQLite (local) / Postgres (prod)
`AIRS_USE_LLM`	Enable Gemini narratives	`true` / `false`
`GEMINI_API_KEY`	Required for LLM features	[Secret Key]
`REPORTS_STORAGE_MODE`	PDF storage strategy	`"local"` / `"gcs"`
`GCS_BUCKET_NAME`	Target bucket for reports	[Bucket Name]

Enabling Generative Narratives

Optional integration with Google Gemini for executive summaries.

Create Secret

```
>_
gcloud secrets create
gemini-api-key
```

Grant Access

Grant IAM role `roles/secretmanager.secretAccessor` to the Cloud Run service account.

Deploy

```
>_
make deploy-gcp-with-
llm
```

Verify functionality via endpoint: `GET /health/llm`

Database Configuration & Migrations

Local Strategy (MVP)

SQLite

`sqlite:///./airs.db`

Note: Data does not persist across Cloud Run instances.

Production Strategy

Cloud SQL (PostgreSQL)

Uses Unix socket connection.

Connection Pool: Size 5, Overflow 10, Recycle 30m.

Migrations (Alembic)

```
$ alembic upgrade head                          // Apply changes
$ alembic revision --autogenerate -m "desc"  // Create revision
```

API & Troubleshooting

Key Endpoints

`GET` `/health` : Load balancer check

`GET` `/health/cors` : Debug CORS headers

`POST` `/api/assessments` : Initiate audit

`GET` `/api/assessments/{id}/report` : Retrieve PDF

Common Issues

CORS Errors:
Run `` `scripts/get_deployment_urls.sh` `` to find valid origins.

Data Loss:
SQLite resets on Cloud Run deploy. Use Postgres for persistence.

Build Fail:
Check `requirements.txt` includes `gunicorn`.

Ready to Assess

AIRS combines deterministic security scoring with generative AI insights.

Live Web App: `airs-demo.web.app`
API Health: `airs-api-[id].us-central1.run.app/health`

- 1. Star the Repository.
- 2. Fork & Contribute.
- 3. Deploy your own instance.

github.com/purvanshbhatt/AIRS