TeleSec-AI

GCP Production Deployment Documentation

Cloud Run | Compute Engine | Docker Databases | Kafka | Pub/Sub | Cloud Build

PROJECT	TeleSec MVP / TeleSec-AI
GCP PROJECT ID	telesec-ai
PRIMARY REGION	us-central1
DOMAINS	telesecai.com / api.telesecai.com
DEPLOYMENT DATE	June 2026
DOCUMENT STATUS	Production documentation - credentials redacted

Document Control

This document captures the end-to-end production deployment flow for the TeleSec-AI project on Google Cloud Platform. It is structured as an operations-ready deployment record with architecture, infrastructure inventory, database configuration, application deployment, CI/CD, troubleshooting notes, and immediate next steps.

Sensitive Information Handling

Passwords, tokens, and long-lived secrets from the original deployment notes have been intentionally redacted in this PDF. Before sharing the document outside the core engineering team, rotate any credential that was previously pasted into chat, tickets, or informal notes, and migrate production secrets to a managed secrets store.

PRODUCTION READY	PRODUCTION READY	DEPLOYED	PENDING FIX
Core FE/BE	Admin	Agents	Kafka

Field	Value
Project	TeleSec MVP / TeleSec-AI
GCP Project ID	telesec-ai
Primary Region	us-central1
Frontend Domain	https://telesecai.com
Backend Domain	https://api.telesecai.com
Deployment Date	June 2026
Infrastructure Control Plane	PowerShell and Google Cloud SDK
Document Version	1.0

Table of Contents

1. Project Architecture Overview

TeleSec-AI is deployed using a serverless-first, event-driven architecture on Google Cloud Platform. The application layer runs on Cloud Run, while the primary operational data layer is hosted in Docker containers on a private Compute Engine VM. Agent services run independently on Cloud Run and communicate through Kafka/Pub/Sub style event pipelines.

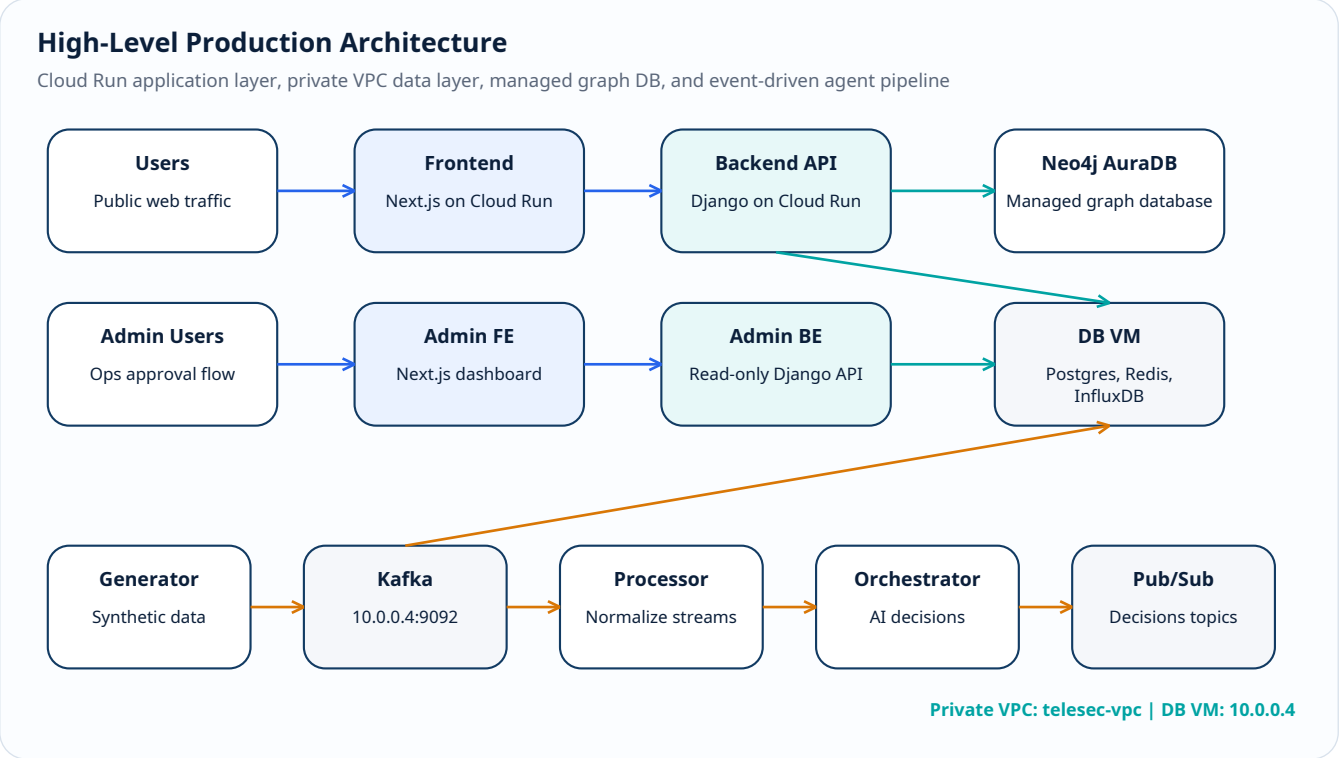

Figure 1. Logical architecture of the TeleSec-AI production deployment.

1.1 Component Inventory

Component	Technology	GCP / Service	URL / Location
Frontend	Next.js / TypeScript	Cloud Run	https://telesecai.com
Backend	Django / Python	Cloud Run	https://api.telesecai.com
Admin Frontend	Next.js	Cloud Run	telesec-admin-fe
Admin Backend	Django read-only API	Cloud Run	telesec-admin-be
PostgreSQL	postgres:15-alpine Docker container	Compute Engine VM	db-instance / 10.0.0.4:5432
Redis	redis:7-alpine Docker container	Compute Engine VM	db-instance / 10.0.0.4:6379
InfluxDB	influxdb:2.7-alpine Docker container	Compute Engine VM	db-instance / 10.0.0.4:8086
Graph Database	Neo4j AuraDB	Managed Neo4j service	94d8846f.databases.neo4j.io
Agents	Python / async / Kafka	Cloud Run	synthetic-generator, stream-processor, orchestrator

Architecture Note

The deployment is event-driven and serverless at the application layer. The database layer currently uses a Compute Engine VM with Dockerized services, which should be treated as a stateful infrastructure component with backup, monitoring, and firewall controls.

2. Networking and Infrastructure Setup

2.1 VPC and Subnet

A custom Virtual Private Cloud was created to secure internal communication between Cloud Run services and the database VM. The private subnet is small and controlled, making it suitable for the current MVP footprint.

Resource	Configuration	Purpose / Notes
VPC	telesec-vpc	Private network boundary for backend, agents, and DB access.
Subnet	telesec-subnet / 10.0.0.0/28	Internal subnet used by private resources and serverless connector traffic.
Serverless VPC Access Connector	telesec-connector	Allows Cloud Run services to reach private IP resources.
Connector Machine Type	e2-micro	Cost-effective connector sizing for MVP workload.
Connector State	READY	Connector is operational and available for Cloud Run routing.

2.2 Database VM

A Compute Engine e2-micro VM named db-instance was provisioned to run Dockerized PostgreSQL, Redis, InfluxDB, and planned Kafka services. The VM is addressed by the internal private IP 10.0.0.4.

Property	Value
Instance Name	db-instance
Machine Type	e2-micro
Private IP	10.0.0.4
Temporary Public IP	35.223.171.249 - assigned for setup and later removed
Primary Role	Stateful database and cache host for the MVP environment

2.3 Required Firewall Posture

The current open networking action is Kafka connectivity. Cloud Run agent services are unable to connect to 10.0.0.4:9092 until the VPC firewall permits TCP/9092 from the serverless connector subnet and Kafka is listening on the VM interface.

Firewall rule required for Kafka access

```
gcloud compute firewall-rules create allow-kafka-from-vpc \
  --network=telesec-vpc \
  --allow=tcp:9092 \
  --source-ranges=10.0.0.0/28 \
  --target-tags=db-server \
  --project=telesec-ai
```

Important Firewall Dependency

The firewall rule depends on the db-instance VM having the network tag db-server. If the tag is missing, either add the tag to the VM or update the firewall rule target to match the VM's actual tag strategy.

3. Database Configuration and Management

The TeleSec-AI data layer currently consists of PostgreSQL for relational application data, Redis for caching, InfluxDB for time-series telemetry, and Neo4j AuraDB for graph-based topology and relationship data.

Database	Container / Service	Endpoint	Configuration
PostgreSQL	postgres:15-alpine	10.0.0.4:5432	Database: telesec; primary user: postgres; password redacted.

Database	Container / Service	Endpoint	Configuration
Redis	redis:7-alpine	10.0.0.4:6379	maxmemory 100mb; policy allkeys-lru.
InfluxDB	influxdb:2.7-alpine	10.0.0.4:8086	Org: telesec-org; bucket: telemetry; token redacted.
Neo4j AuraDB	Managed service	neo4j+s://94d8846f.databases.neo4j.io	User: neo4j; password redacted.

3.1 PostgreSQL

Container image: postgres:15-alpine.

Network mapping: private VM IP 10.0.0.4 on port 5432.

Database name: telesec.

Primary administrative user: postgres; password redacted in this document.

PgAdmin setup used an SSH tunnel through localhost:5433 during setup.

Admin app read-only user: telesec_admin_ro with SELECT-only permissions.

3.2 Redis

Container image: redis:7-alpine.

Network mapping: private VM IP 10.0.0.4 on port 6379.

Configured with maxmemory 100mb.

Eviction policy: allkeys-lru.

3.3 InfluxDB

Container image: influxdb:2.7-alpine.

Network mapping: private VM IP 10.0.0.4 on port 8086.

Organization: telesec-org.

Bucket: telemetry.

Token stored in Cloud Run environment variables; recommended final state is Secret Manager-backed injection.

3.4 Neo4j AuraDB

Connection URI: neo4j+s://94d8846f.databases.neo4j.io.

Database user: neo4j.

Password redacted and should be rotated if ever exposed in plain text notes.

4. Main Application Deployment - Core TeleSec

4.1 Cloud Run Services

Service	Repository	Runtime	Public Entry Point
telesec-backend	telesec-be	Django / Python / Gunicorn	https://api.telesecai.com
telesec-frontend	telesec-fe	Next.js / TypeScript	https://telesecai.com

4.2 Environment Variables

The following key variables were configured through cloudbuild.yaml and Cloud Run environment settings. Secret values are redacted.

Variable	Used By	Value / Example	Notes
DB_HOST	Backend / Admin BE	10.0.0.4	Private DB VM endpoint.
DB_PASSWORD	Backend	<REDACTED>	Move to Secret Manager and rotate.
NEO4J_URI	Backend / Agents	neo4j+s://94d8846f.databases.neo4j.io	Managed graph database URI.
NEO4J_PASSWORD	Backend / Agents	<REDACTED>	Move to Secret Manager and rotate.
PUBSUB_PROJECT	Agents / Backend	telesec-ai	GCP project for Pub/Sub topics.
NEXT_PUBLIC_API_URL	Frontend / Admin FE	https://api.telesecai.com	Must be present at build time for Next.js.
KAFKA_BOOTSTRAP_SERVERS	Agents	10.0.0.4:9092	Pending Kafka connectivity fix.

4.3 Key Code Fixes Applied

Fix	Reason	Implementation
Gunicorn startup	Backend failed because gunicorn was missing or not invoked correctly.	Updated requirements.txt and Dockerfile to run python -m gunicorn.
Allowed hosts and CSRF	Django rejected requests from production domains and Cloud Run URLs.	Added ALLOWED_HOSTS and CSRF_TRUSTED_ORIGINS for api.telesecai.com and Cloud Run service URLs.
billing_plan column	Database schema was out of sync with Django models.	SSH into db-instance and ran makemigrations authentication followed by migrate.
Next.js API URL	Frontend could not connect because NEXT_PUBLIC_API_URL was not baked at build time.	Added ARG NEXT_PUBLIC_API_URL in Dockerfile and --build-arg in cloudbuild.yaml.

Database migration applied for missing billing_plan column

```
# Django migration fix performed on the VM-backed backend environment
python manage.py makemigrations authentication
python manage.py migrate
```

4.4 Domain Mapping and SSL

Domain	Mapped Service	DNS Type	Target / Status
telesecai.com	telesec-frontend	A records	216.239.32.21 and related Google domain mapping IPs
api.telesecai.com	telesec-backend	CNAME	ghs.googlehosted.com.
SSL	Frontend and Backend	Managed certificate	Google Cloud provisioned certificate status: READY

5. Admin Application Deployment

The Admin application is deployed as a separate read-only operational interface for user approval and management workflows. It uses a dedicated read-only database user to reduce the blast radius of admin-facing operations.

Layer	Service	Purpose	Important Controls
Admin Backend	telesec-admin-be	Django API for admin workflows and approvals	Uses telesec_admin_ro with SELECT-only permissions; migrations disabled for authentication app.
Admin Frontend	telesec-admin-fe	Next.js dashboard	Connects to Admin BE through NEXT_PUBLIC_API_URL.

5.1 Read-Only Backend Safety

The Admin Backend was configured with a read-only database user. Migration safety was added by disabling migrations for the authentication module so the admin service cannot accidentally alter the production schema.

Admin migration safety setting

```
# settings.py safety control used by Admin Backend
MIGRATION_MODULES = {
    'authentication': None
}
```

6. Agent Architecture and Event-Driven Pipeline

The agent layer implements the hand-drawn agent architecture using separate Cloud Run microservices. The initial pipeline includes synthetic generation, stream processing, and orchestration services.

Repository / Service	Role	Status
telesec-synthetic-generator	Generates synthetic raw telecom/security events.	Cloud Build green; Cloud Run live; health check serving on port 8080.
telesec-stream-processor	Consumes raw data, normalizes streams, and prepares processed events.	Cloud Build green; Cloud Run live; health check serving on port 8080.
telesec-orchestrator-ai-system-	Consumes normalized events and produces AI orchestration decisions.	Cloud Build green; Cloud Run live; health check serving on port 8080.

6.1 Pub/Sub Topics

Topic	Purpose
synthetic-raw-data	Raw generated telemetry/event data.
synthetic-normalized-data	Normalized stream data produced by processors.
orchestrator-decisions	Decision outputs from the orchestrator service.

6.2 Kafka Implementation

The agent architecture was implemented with Apache Kafka as the asynchronous event backbone. Agent services use aiokafka for producer and consumer behavior. The shared Kafka bootstrap endpoint is 10.0.0.4:9092.

Agent Kafka environment variable

```
KAFKA_BOOTSTRAP_SERVERS=10.0.0.4:9092
```

6.3 Cloud Run Health Requirement and Threading Fix

Cloud Run requires a container to listen on the port defined by PORT, typically 8080. Long-running Kafka consumers were adjusted to run alongside a lightweight web server using threading so the service remains healthy while the async pipeline continues running.

main.py threading pattern for Cloud Run

```
web_thread = threading.Thread(target=start_web_server, daemon=True)
web_thread.start()
asyncio.run(produce_messages())
```

6.4 Current Agent Pipeline Status

Area	Status	Details
Cloud Build	READY	All three initial agent services build successfully.
Cloud Run	READY	All three initial agent services are live and serving health checks on port 8080.
Kafka Connectivity	PENDING	Services currently fail to reach 10.0.0.4:9092 with Connection refused. Firewall and Kafka listener binding must be corrected.

7. CI/CD Pipelines - Cloud Build

All repositories are linked to GCP using Developer Connect connection telesecai-github-conn. Deployment is triggered on pushes to the main branch using the branch pattern ^main$.

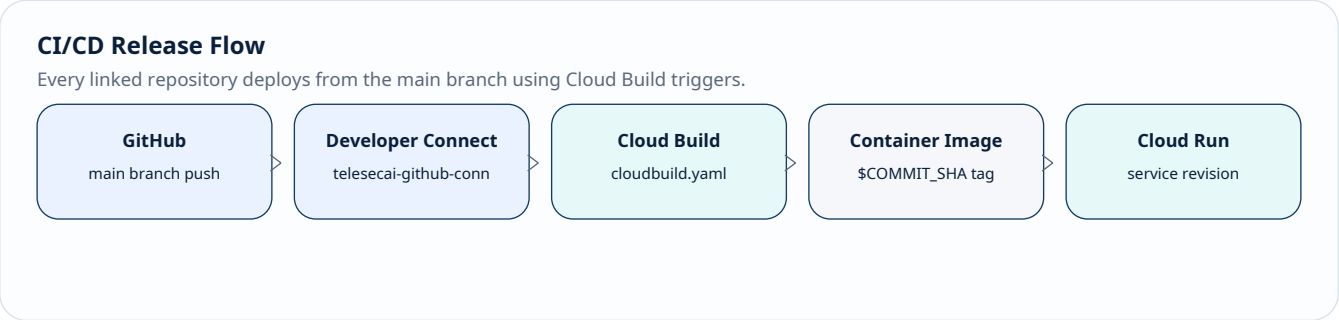

Figure 2. CI/CD flow from GitHub source to Cloud Run revision.

Repository	Cloud Build Trigger Name	Service Account
telesec-be	telesec-be-trigger	telesec-build-sa@...
telesec-fe	telesec-fe-trigger	telesec-build-sa@...
admin-be	telesec-admin-be-trigger	telesec-build-sa@...
admin-fe	telesec-admin-fe-trigger	telesec-build-sa@...
synthetic-generator	telesec-synthetic-generator-trigger	telesec-build-sa@...
stream-processor	telesecai-telesec-stream-processor	telesec-build-sa@...
orchestrator	telesecai-telesec-orchestrator-ai-system	telesec-build-sa@...

7.1 Standard Trigger Condition

Cloud Build trigger configuration

```
Branch pattern: ^main$
Trigger event: Push to main
Build file: cloudbuild.yaml
Image tagging pattern: image:$COMMIT_SHA
```

8. Troubleshooting Summary and Applied Fixes

The following table records the major deployment issues encountered and the fixes applied. This section should be used as the first reference when recreating the environment or onboarding another engineer.

Issue	Root Cause	Applied Fix
Build fails immediately under 20 seconds	cloudbuild.yaml image tag did not include $COMMIT_SHA.	Replaced plain image reference with image:$COMMIT_SHA.
403 Forbidden on Admin Login	Cloud Run URL missing from CSRF_TRUSTED_ORIGINS.	Added the live Cloud Run URL to Django settings.py.
column billing_plan does not exist	Database schema was out of sync with Django models.	SSH into db-instance and ran makemigrations and migrate.
Frontend fails to connect to Backend	NEXT_PUBLIC_API_URL was not baked at build time.	Added ARG NEXT_PUBLIC_API_URL and --build-arg in Dockerfile/cloudbuild.yaml.
Git nothing to commit issues	Local Git state was out of sync or corrupted.	Force pushed corrected cloudbuild.yaml or edited directly through GitHub UI.
Cloud Run container crashes after build	main.py lacked an HTTP server or had a blocking async thread.	Implemented a BaseHTTPRequestHandler health server in a background thread while async consumers run.

Issue	Root Cause	Applied Fix
Kafka connection refused	Kafka service unreachable at 10.0.0.4:9092.	Create firewall rule, tag DB VM, and verify Kafka listens on 0.0.0.0:9092.

9. Operations Runbook and Validation Checklist

Use this checklist after deployments, infrastructure changes, or incident recovery. It is designed to verify public endpoints, private connectivity, service health, data layer availability, and agent pipeline readiness.

Check	Expected Result	Action / Command
Frontend domain	telesecai.com loads successfully over HTTPS.	Open https://telesecai.com and validate login/user flows.
Backend domain	api.telesecai.com responds over HTTPS.	Call health/API endpoint used by the Django backend.
Admin FE	Dashboard loads successfully.	Open admin frontend service URL or mapped admin domain if added.
Admin BE	Read-only API is reachable.	Validate admin login/approval endpoints and ensure no write migrations run.
DB VM private IP	Cloud Run can reach 10.0.0.4 through VPC connector.	Review Cloud Run VPC connector settings and DB logs.
PostgreSQL	Port 5432 available from backend services.	Check backend Cloud Run logs for successful DB connection.
Redis	Port 6379 available from services that use cache.	Check application logs and Redis container status.
InfluxDB	Port 8086 available for telemetry writes.	Validate bucket telemetry receives data.
Kafka	Port 9092 reachable and listener bound.	Run container/listener checks on db-instance and test from Cloud Run logs.

9.1 Useful Verification Commands

Operational validation commands

```
# SSH into database VM
gcloud compute ssh db-instance --project=telesec-ai --zone=<ZONE>

# Check running Docker containers
sudo docker ps

# Confirm Kafka is listening on the VM
sudo ss -lntp | grep 9092

# Inspect service logs
gcloud run services logs read telesec-synthetic-generator   --region=us-central1
    --project=telesec-ai
```

10. Security, Secrets, and Production Hardening

The current MVP is production-ready for the deployed core application and admin interface, but the following hardening actions are recommended before wider production usage, external audit, or high-volume onboarding.

Area	Current State	Recommended Action
Secrets	Some values were configured through environment variables during deployment.	Move secrets to Secret Manager and rotate all exposed credentials.
Database VM	Stateful Docker containers run on Compute Engine e2-micro.	Implement backups, disk monitoring, restart policies, and upgrade plan.
Public IP	Temporary setup IP was removed after initial provisioning.	Keep public IP disabled unless a controlled maintenance window requires it.

Area	Current State	Recommended Action
Network access	Private subnet and VPC connector are configured.	Use least-privilege firewall rules and document every open port.
Admin Backend	Read-only DB user and migration safety enabled.	Keep read-only permissions enforced; audit any permission changes.
CI/CD	Cloud Build triggers deploy from main.	Enforce branch protection, code review, and limited service account IAM.
Observability	Deployment logs and service health available through Cloud Run.	Add uptime checks, alerting, log-based metrics, and error budgets.

10.1 Redacted Secret Inventory

Secret / Credential	Referenced By	Document Handling	Recommended Storage
PostgreSQL primary password	Backend service	Redacted	Secret Manager
PostgreSQL admin read-only password	Admin backend	Redacted	Secret Manager
InfluxDB token	Telemetry services	Redacted	Secret Manager
Neo4j AuraDB password	Backend and graph/agent code	Redacted	Secret Manager

Credential Rotation Priority

Because credentials were included in the original raw deployment notes, rotate PostgreSQL, Neo4j, and InfluxDB secrets before distributing this document beyond trusted administrators.

11. Immediate Next Steps

The TeleSec Core application and Admin application are operational. The next focus is completing Kafka connectivity and expanding the agent microservice fleet.

Priority	Action	Owner / Notes	Expected Outcome
P0	Create Kafka firewall rule for TCP/9092 from 10.0.0.0/28.	Infrastructure / GCP admin	Cloud Run agents can reach Kafka endpoint.
P0	Verify Kafka container is running and bound to 0.0.0.0:9092.	Infrastructure / VM owner	Connection refused error resolved.
P1	Add db-server network tag to db-instance if missing.	Infrastructure	Firewall target applies correctly.
P1	Deploy A1-A4 agent microservices: fiber, fraud, assurance, incident agents.	Agent engineering	Expanded agent pipeline aligned with full architecture.
P1	Move redacted secrets to Secret Manager and rotate old values.	Security / DevOps	Safer secret lifecycle and reduced exposure risk.
P2	Add automated backups and uptime/error alerts.	SRE / DevOps	Improved recoverability and incident response.

11.1 Kafka Completion Procedure

1. Confirm db-instance has the expected network tag used by firewall rules.
2. Create or update the firewall rule allowing TCP/9092 from the serverless connector subnet 10.0.0.0/28.
3. SSH into db-instance and verify the Kafka container is running.
4. Confirm Kafka advertises a listener reachable from Cloud Run, not only localhost.
5. Redeploy or restart agent services after connectivity is corrected.
6. Validate logs for successful producer/consumer startup and message flow.

12. End-to-End Deployment Flow

The following implementation sequence captures the deployment flow from infrastructure creation through application release and agent pipeline setup.

Step	Deployment Activity	Result
1	Create GCP project context and set primary region to us-central1.	All resources deployed under project telesec-ai.
2	Create custom VPC telesec-vpc and subnet telesec-subnet 10.0.0.0/28.	Private network foundation created.
3	Create Serverless VPC Access connector telesec-connector.	Cloud Run can reach private IP services.
4	Provision Compute Engine VM db-instance.	Stateful Docker host available at 10.0.0.4.
5	Run PostgreSQL, Redis, and InfluxDB Docker containers.	Application data, cache, and telemetry storage available.
6	Configure Neo4j AuraDB credentials and URI.	Graph database connectivity prepared.
7	Deploy backend Cloud Run service from telesec-be.	API service available behind api.telesecai.com.
8	Deploy frontend Cloud Run service from telesec-fe.	Web application available behind telesecai.com.
9	Apply DNS mapping and managed SSL certificates.	Production HTTPS endpoints active.
10	Deploy Admin Backend with read-only DB user and migration protection.	Admin API isolated from schema mutations.
11	Deploy Admin Frontend with Admin BE API URL.	Admin dashboard available.
12	Create Pub/Sub topics for raw, normalized, and decision events.	Event pipeline topics ready.
13	Deploy synthetic-generator, stream-processor, and orchestrator services.	Agent Cloud Run services live with health checks.
14	Resolve Kafka firewall/listener issue.	Final event stream connectivity to complete the initial pipeline.

13. Conclusion

The TeleSec-AI project has been successfully deployed on Google Cloud Platform using Cloud Run, Cloud Build, Compute Engine, Dockerized data services, Pub/Sub topics, and an initial Kafka-based agent framework.

Area	Production Status	Notes
Frontend and Backend	Production ready	telesecai.com and api.telesecai.com are mapped with managed SSL.
Admin Interface	Production ready	Uses strict read-only database access and migration safety controls.
Agent Infrastructure	Deployed	Initial three agent services are live; Kafka connectivity is pending.
CI/CD	Operational	Cloud Build triggers linked to main branch deployments.
Security	Requires hardening	Rotate exposed secrets and migrate to Secret Manager.

Final Deployment Position

TeleSec Core and Admin applications are operational. Completing the Kafka firewall/listener configuration will finalize the first agent pipeline, after which additional specialized micro-agents can be deployed using the same Cloud Run and Cloud Build pattern.