**Secure GCP Chatbot Platform**

Client Proposal & Delivery Overview

Prepared by GCP Delivery Team | October 04, 2025

# Executive Summary

We designed a secure, production-ready Vertex AI chatbot platform on Google Cloud that blends high-availability infrastructure, defense-in-depth controls, and automated delivery pipelines. The stack is fully codified in Terraform and backed by tested application components, giving your organisation a fast path to pilot and scale conversational AI with confidence.

# Business Outcomes

* Accelerate time-to-value for AI assistants by pairing managed Vertex AI services with reusable platform modules.
* Reduce operational risk through automated guardrails, continuous security checks, and auditable infrastructure as code.
* Enable data protection and compliance from day one via private networking, curated IAM roles, and immutable logging exports.

# Solution Overview

The solution packages a landing zone, ML operations toolkit, and client-facing FastAPI service into a cohesive programme. Environment-specific Terraform stacks (`infra/envs/dev`, `infra/envs/prod`) assemble composable modules for networking, identity, observability, security operations, GKE, Vertex AI, Cloud Run, and Cloud Build. The Python application in `app/src/chatbot\_service` provides a zero-trust gateway to Vertex AI with rate limiting, token verification, and offline-safe fallbacks.

# Architecture Highlights

* Shared VPC with dedicated app and data subnets, secondary IP ranges for GKE, optional NAT, and Cloud Armor security policy baselines for SQLi/XSS protection (see `infra/modules/networking/main.tf`).
* Autopilot GKE deployment with private nodes, Workload Identity integration, managed Prometheus, and Binary Authorization toggle for container provenance (`infra/modules/gke/main.tf`).
* Vertex AI foundation including dataset bucket, Feature Store, Tensorboard, Artifact Registry, and private endpoint bindings to the VPC (`infra/modules/vertex/main.tf`).
* Hardended IAM footprint using purpose-built service accounts and GitHub workload identity federation so CI/CD runs without long-lived keys (`infra/modules/iam/main.tf`).
* Security operations pipeline creating logging sinks to BigQuery, GCS archive, Pub/Sub incident topics, plus SCC high-severity alerts and Web Security Scanner automation (`infra/modules/logging/main.tf`, `infra/modules/security/main.tf`).
* Edge services through Cloud Run with explicit ingress, VPC access, autoscaling hints, and optional NEG integration for load balancing (`infra/modules/cloudrun/main.tf`).

# Infrastructure Modules At A Glance

|  |  |  |
| --- | --- | --- |
| Module | Purpose | Key Highlights |
| networking | Shared VPC & subnet orchestration | Custom subnets, Cloud NAT, firewall rules for IAP/LB, optional Cloud Armor policy. |
| iam | Least-privilege identities | Service accounts for CI/CD, runtime, Vertex; GitHub OIDC federation; scoped role bindings. |
| logging | Telemetry & retention | Security log bucket, BQ dataset, GCS archive with retention policies, Pub/Sub for IR notifications. |
| security | Operational security | Security Command Center streaming, IAM change metrics & alerts, weekly DAST scanner configuration. |
| gke | Core runtime cluster | Autopilot cluster with private nodes, Workload Identity, managed Prometheus, mesh certificates. |
| vertex | ML operations backbone | Dataset storage, Feature Store, endpoint provisioning, Artifact Registry for model images. |
| cloudrun | Edge inference API | Gen2 runtime, declarative autoscaling, VPC connectors, authenticated-only by default. |
| cloudbuild | CI/CD trigger | GitHub-triggered builds, Docker build/push automation, substitutions for environment promotion. |
| env stacks | Environment composition | Parameterised `dev`/`prod` stacks wiring modules together with consistent labels and CIDRs. |

Modules are version-controlled and reusable, enabling phased rollout or scoped adoption depending on the target environment.

# Security & Compliance Posture

* High-severity Security Command Center findings stream to Pub/Sub for real-time notification while IAM policy changes raise Cloud Monitoring alerts (`infra/modules/security/main.tf`).
* Immutable log retention across BigQuery (analytics) and Cloud Storage (cold archive) with CMEK-ready buckets and 365-day retention (`infra/modules/logging/main.tf`).
* Policy-as-code guardrails using OPA/Conftest and Config Validator rules (`policies/opa`, `policies/terraform-validator`) executed via `scripts/security\_checks.sh`.
* Service perimeter ready design: private Vertex AI endpoints, Cloud Run VPC connectors, and workload identity enforce zero standing credentials.

# Operations & Automation

* One-touch `scripts/security\_checks.sh` pipeline installs app dependencies, runs lint/type/test suites, validates Terraform, and executes Checkov/OPA checks before deployment.
* GitHub-triggered Cloud Build pipeline builds and pushes hardened container images with environment substitutions for promotion control (`infra/modules/cloudbuild/main.tf`).
* Terraform plans structured per environment with remote state, formatting, validation, and policy enforcement baked in (`infra/envs/\*`).
* Documentation already seeded with architecture and pipeline runbooks (`docs/architecture.md`, `docs/pipeline.md`) to accelerate onboarding.

# Application Experience

* FastAPI service exposes `/chat` and `/health` endpoints with Pydantic schemas, contextual prompts, and offline fallbacks (`app/src/chatbot\_service/main.py`).
* Google-signed token verification, configurable audiences, and rate limiting defend the API against abuse (`app/src/chatbot\_service/security.py`, `app/src/chatbot\_service/rate\_limiter.py`).
* Vertex AI client wrapper initialises managed models when available and gracefully downgrades to offline mode for local testing (`app/src/chatbot\_service/vertex\_client.py`).
* Automated unit tests validate core behaviours and offline handling to keep CI fast and deterministic (`app/tests/test\_main.py`).

# Implementation Roadmap

1. Initiation (Week 0-1): confirm project scope, landing zone prerequisites, and networking alignment; activate required Google Cloud APIs.
2. Foundations (Week 2-4): provision shared VPC, IAM federation, logging/security modules; deliver Connectivity & Security sign-off.
3. Application Enablement (Week 5-6): deploy GKE Autopilot, Vertex AI assets, Cloud Run edge; integrate CI/CD and policy automation.
4. Validation (Week 7-8): execute security checks, penetration tests, and performance baselines; document runbooks and handover guides.
5. Launch & Optimisation (Week 9+): production cutover with monitoring dashboards, iterative tuning of model endpoints, and optional SRE support package.

# Engagement Model & Next Steps

* Joint steering cadence with weekly demos across infrastructure, ML ops, and application workstreams.
* Client enablement sessions covering Terraform workflows, security operations, and Vertex AI best practices.
* Optional managed services add-ons: 24/7 SRE coverage, finops dashboards, or model evaluation accelerators.
* Next step: schedule a 60-minute solution walkthrough and align on pilot success criteria to kick off mobilisation.

**We look forward to partnering on a secure, scalable AI assistant platform that meets your governance standards and accelerates business impact.**