Secure GCP AI Chatbot Platform – Client Briefing

*Prepared for: <Client Name>*

*Prepared by: <Your Team / Company>*

Date: October 04, 2025

# 1. Executive Overview

We engineered a production-ready Google Cloud Platform (GCP) foundation that combines secure networking, managed AI services, and automated delivery workflows for an enterprise chatbot workload. The repository already contains working Terraform modules, a secured FastAPI gateway, and a CI/CD pipeline that prove our ability to deliver high-assurance AI platforms rapidly.

* Accelerates time-to-value with repeatable Terraform blueprints that codify networking, security, data, and runtime services.
* Pairs a FastAPI-based inference gateway with Vertex AI to deliver auditable, rate-limited chatbot experiences.
* Bakes in DevSecOps guardrails—policy-as-code, static analysis, automated testing—ensuring every change ships with confidence.

# 2. Business Outcomes and Success Metrics

Partnering on this implementation positions your team to achieve the following measurable outcomes:

* Launch a governed AI assistant with 99.9% uptime targets while maintaining private data boundaries.
* Cut infrastructure provisioning lead times from weeks to hours through environment codification under `infra/envs/\*`.
* Reduce security review cycles by enforcing OPA and Terraform Validator controls (`policies/opa`, `policies/terraform-validator`).
* Enable auditable deployments via workload identity federation and GitHub Actions pipelines in `.github/workflows/`.

# 3. Solution Snapshot

The table below summarizes the pillars already implemented in this repository and the value they drive.

|  |  |  |
| --- | --- | --- |
| Pillar | Key Components | Client Value |
| Infrastructure Foundation | Custom VPC, subnets, Cloud NAT, firewall and Cloud Armor rules in `infra/modules/networking`. | Creates segmented, internet-facing yet protected landing zones ready for regulated workloads. |
| Identity & Access | Service accounts and GitHub OIDC federation in `infra/modules/iam`. | Removes shared secrets while enforcing least-privilege automation and runtime access. |
| Security Operations | Security Command Center configuration, DAST scanner, API enablement in `infra/modules/security`. | Delivers continuous monitoring, high-severity alert routing, and proactive vulnerability management. |
| Observability & Compliance | Centralized logging, BigQuery analytics, Pub/Sub IR topics in `infra/modules/logging`. | Guarantees immutable log retention, accelerates incident investigations, and supports compliance reporting. |
| Application & AI Runtime | Cloud Run edge, GKE Autopilot, Vertex AI Feature Store/Endpoint modules (`infra/modules/cloudrun`, `infra/modules/gke`, `infra/modules/vertex`). | Provides scalable execution for chatbot APIs and model lifecycles with secure private networking. |
| DevSecOps Automation | GitHub Actions pipelines (`ci.yml`, `deploy.yml`) plus `scripts/security\_checks.sh` mirror. | Enables consistent testing, policy enforcement, and push-button promotions across environments. |

# 4. Architecture Highlights

## 4.1 Networking & Perimeter Security

* Dedicated VPC with explicitly defined application and data subnets, plus secondary ranges for pods/services (`infra/modules/networking`).
* Private GKE control planes secured by master CIDR blocks and optional authorized networks (`infra/modules/gke`).
* Cloud Armor baseline WAF rules with room for bespoke expressions to mitigate geo/IP threats.
* Ingress policies for load balancer health checks and Identity-Aware Proxy ranges aligned to Google best practices.

## 4.2 Identity & Access Management

* Separate service accounts for CI/CD, runtime, and Vertex pipelines with curated role sets (`infra/modules/iam`).
* GitHub Actions authentication via Workload Identity Federation eliminates long-lived JSON keys.
* Environment modules (`infra/envs/dev`, `infra/envs/prod`) inherit IAM outputs and propagate least privilege to workloads.

## 4.3 Application Delivery & AI Integration

* FastAPI gateway (`app/src/chatbot\_service/main.py`) wraps Vertex AI text generation with rate limiting and OAuth token verification.
* Offline development mode safeguards local iteration while preserving production-grade behaviors (`app/src/chatbot\_service/vertex\_client.py`).
* Cloud Run services provide authenticated HTTPS ingress, optionally connected to serverless NEG for load balancing (`infra/modules/cloudrun`).
* GKE Autopilot clusters manage container workloads with Binary Authorization toggles and Workload Identity integration (`infra/modules/gke`).

## 4.4 Security & Compliance Guardrails

* Curated API enablement, SCC high-severity notifications, and weekly DAST scans defined in `infra/modules/security`.
* Policy-as-code checks in `policies/opa/storage.rego` and `policies/terraform-validator/disallow\_public\_sql.yaml` block misconfigurations.
* CI pipeline enforces Ruff, MyPy, pytest, Terraform validate, Checkov, and OPA tests before promotion (`.github/workflows/ci.yml`).

## 4.5 Observability & Incident Response

* Log sinks stream to BigQuery and Cloud Storage with immutable retention settings (`infra/modules/logging`).
* Incident response Pub/Sub topics feed downstream automation or SIEM tooling.
* Health and chat endpoints covered by automated tests (`app/tests/test\_main.py`) ensuring early detection of regressions.

# 5. Delivery Approach and Timeline

Our recommended delivery cadence ensures rapid value while maintaining rigorous governance. Durations are illustrative and can be adjusted to match your internal cadence.

|  |  |  |  |
| --- | --- | --- | --- |
| Phase | Focus | Key Activities | Timeline |
| 1. Discovery & Alignment | Stakeholder alignment | Workshops, success criteria, compliance inputs, shared responsibility matrix. | Week 1 |
| 2. Foundation Deployment | Landing zone & IAM | Instantiate networking, IAM federation, logging baselines via Terraform in dev. | Weeks 2-3 |
| 3. Application & Data Enablement | Runtime + data plane | Deploy Cloud Run, GKE Autopilot, Vertex dataset buckets, integrate CI/CD pipelines. | Weeks 4-5 |
| 4. Hardening & Go-Live | Operational readiness | Policy tuning, performance testing, runbook sign-off, production apply, training. | Week 6 |

# 6. Collaboration Model and Next Steps

Engagement highlights that differentiate our team:

* Transparent delivery—Terraform plans, test results, and policy reports shared on every pull request.
* Knowledge transfer embedded through documentation (`docs/architecture.md`, `docs/pipeline.md`) and pair-working sessions.
* Operational readiness via reproducible runbooks (`scripts/security\_checks.sh`) and automated smoke tests in `deploy.yml`.
* Post-launch support options ranging from hypercare to co-managed operations.

Recommended immediate next steps:

1. Schedule a 90-minute architecture review to walk stakeholders through the Terraform stacks and deployment pipeline.
2. Agree on production landing zone parameters (project IDs, CIDR ranges, secrets management) to populate Terraform variables.
3. Finalize success metrics and onboarding plan so we can initiate Phase 1 within the next sprint.

# Appendix A – Technical Inventory

* `infra/modules/networking` – VPC, subnet, NAT, firewall, and Cloud Armor automation.
* `infra/modules/iam` – Service account provisioning plus GitHub OIDC federation for CI/CD.
* `infra/modules/security` – Security Command Center notifications and Cloud Security Scanner scheduling.
* `infra/modules/logging` – Central log retention via BigQuery, Cloud Storage, and Pub/Sub sinks.
* `infra/modules/gke` – Autopilot GKE clusters with private control planes, binary authorization toggles, and workload identity.
* `infra/modules/cloudrun` – Cloud Run service with configurable ingress, scaling, VPC connectors, and NEG integration.
* `infra/modules/vertex` – Vertex AI feature store, endpoint, artifact registry, and dataset buckets.
* `infra/modules/cloudbuild` – Cloud Build trigger for container builds tied to GitHub pushes.
* `infra/envs/dev` & `infra/envs/prod` – Environment compositions referencing all modules with environment-specific defaults.
* `app/src/chatbot\_service/` – FastAPI application with rate limiting, token verification, Vertex AI adapter, and typed schemas.
* `app/tests/test\_main.py` – Automated coverage of health and chat endpoints ensuring regression protection.
* `policies/opa` & `policies/terraform-validator` – Policy-as-code guardrails executed locally and in CI.
* `.github/workflows/ci.yml` & `deploy.yml` – CI/CD definitions for scanning, planning, applying, and smoke testing.

# Appendix B – Quality & Security Automation

* `scripts/security\_checks.sh` replicates the GitHub Actions pipeline locally, running Ruff, MyPy, pytest, Terraform validation, Checkov, and OPA.
* CI coverage reports are persisted as artifacts, enabling governance teams to inspect evidence of testing.
* Deployment workflow performs cloud-native container builds, Terraform applies, and endpoint smoke tests before marking success.
* OPA policies enforce uniform bucket-level access while Terraform Validator blocks public SQL instances, ensuring compliance baselines.