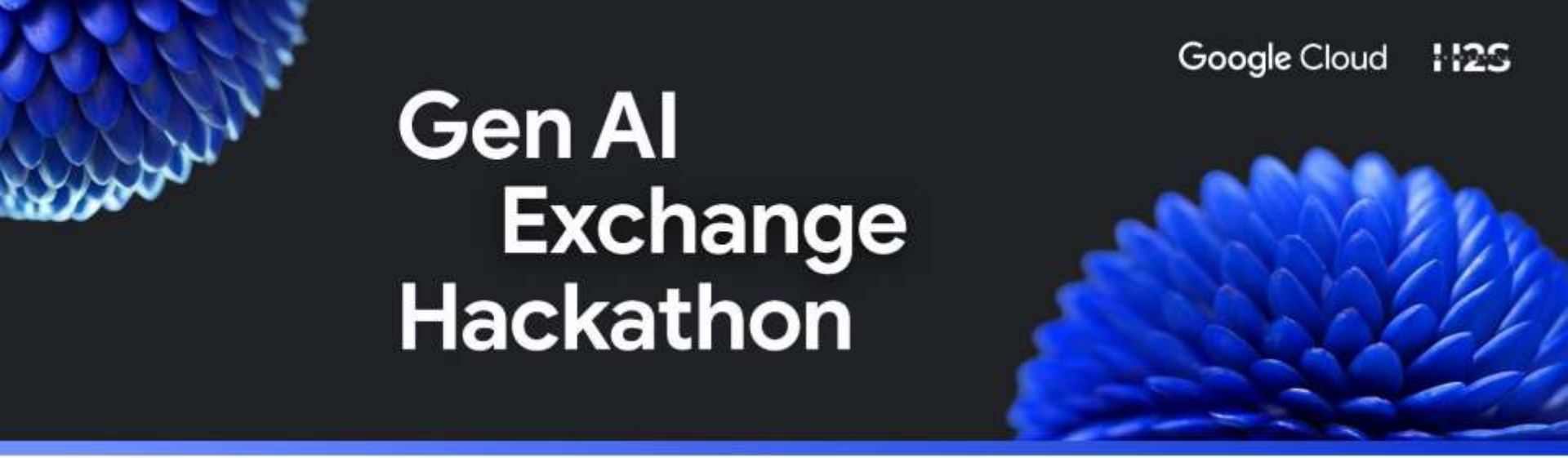


Team Name : Soumya

Team Leader Name : Soumya S Kadakol

Problem Statement : How to deploy a secure MCP server on cloud run

Prototype:

- The MCP server is implemented using Python with a simple HTTP transport for scalable, stateless client connections.
- The code is containerized via a Dockerfile. For example, use the official Python image and employ a tool like uv to run your server script.
- Project structure includes server.py for MCP logic, a Dockerfile for building the image, and optionally a test_server.py for connectivity tests.

Steps to Deploy Securely:

- Build and containerize the MCP server, either with Docker or directly from source.
- Deploy to Cloud Run using the gcloud run deploy mcp-server --no-allow-unauthenticated command, which requires authentication for all requests.
- Set IAM policy to bind the roles/run.invoker role to client or service accounts needing access, enforcing identity-based authorization.
- To test/demo, use the Cloud Run proxy locally for authenticated communication between MCP client and server.

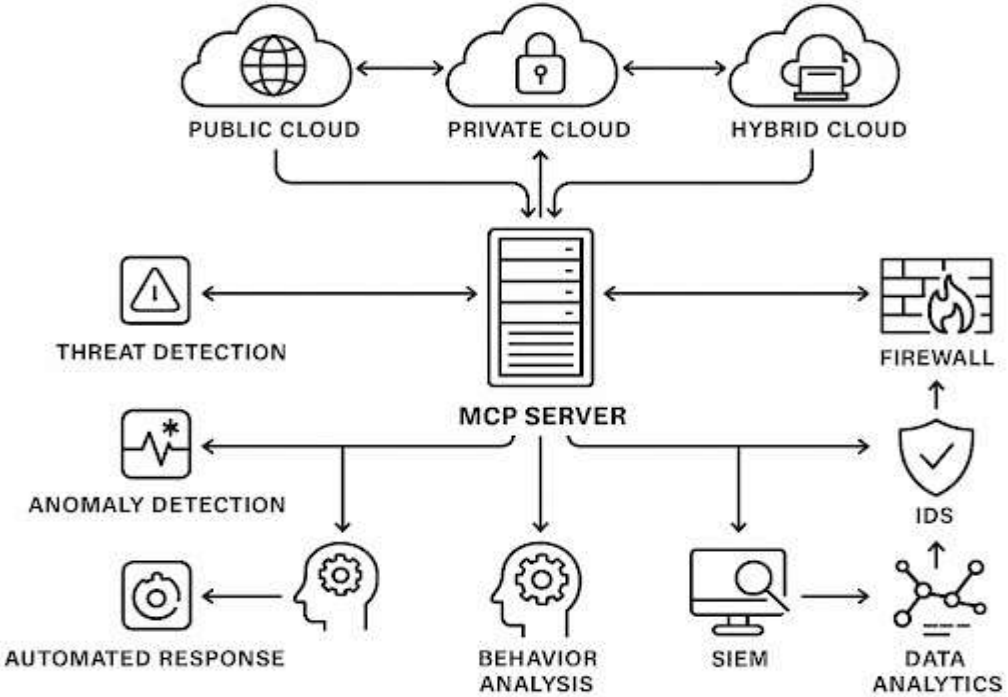
Opportunities MCP server solution :

Difference: Uses the standardized Model Context Protocol with modern streamable HTTP transport deployed on Google Cloud Run, offering automatic scalability and strict IAM-based authentication for security. This contrasts with other local or less secure MCP implementations.

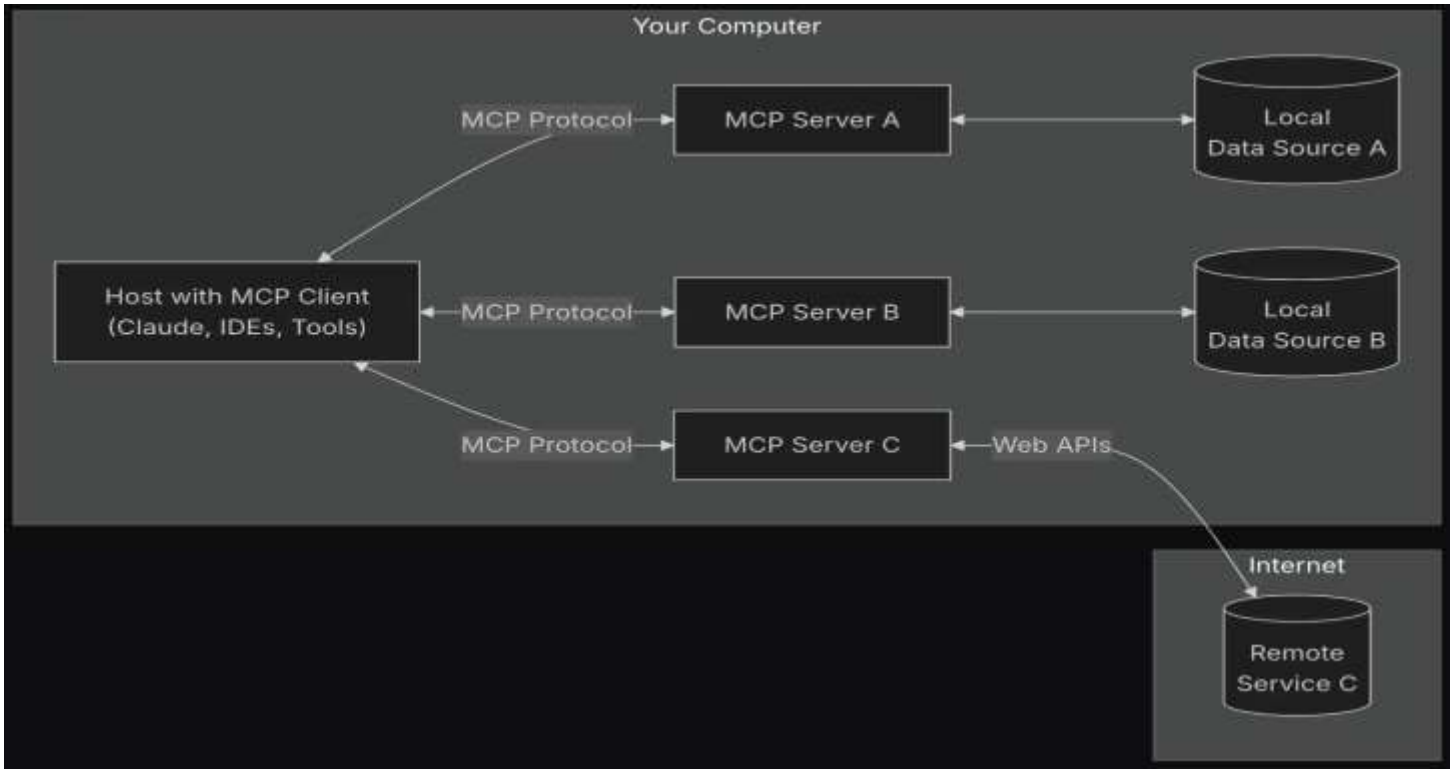
Problem Solving: Enables secure, low-latency, centralized context sharing between AI tools and LLMs, simplifying integration and collaboration across distributed systems while managing authentication and access control natively in the cloud.

USP: Combines production-ready security, cloud-native auto-scaling, team-wide centralized access, and protocol flexibility in a turnkey serverless deployment that reduces operational complexity and enhances AI model effectiveness.

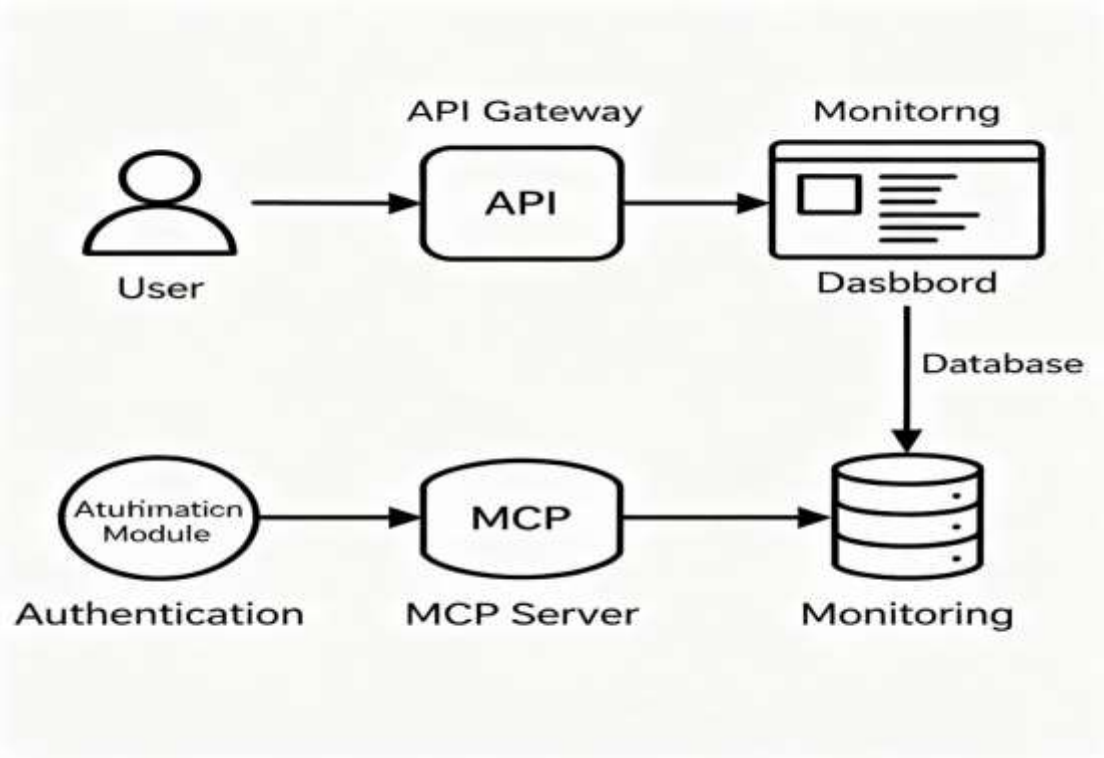
List of features offered by the solution in visual representation :



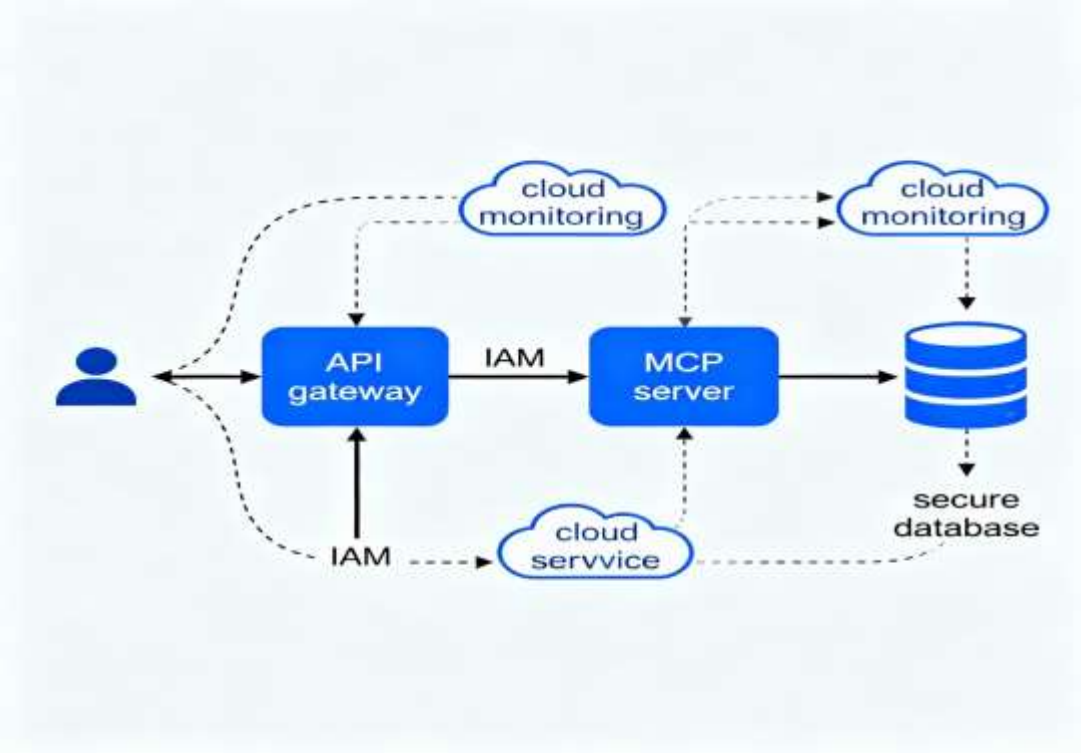
Proccss flow diagram or Usc-casc diagram



Mock diagrams of the proposed solution :



Architecture diagram of the proposed for solution :



Technologies used in the solution :

- **Google Cloud Run:** Serverless container platform for hosting MCP server.
- **Docker:** Containerizes MCP server application.
- **Google Cloud IAM:** Manages authentication and access control.
- **Artifact Registry:** Stores container images securely.
- **Cloud Build:** Automates building and pushing container images.
- **Cloud Monitoring & Logging:** Tracks service health and logs.
- **Optional databases:** Cloud SQL or Firestore for storing persistent data.

Estimated implementation cost :

Cloud Run compute (1 vCPU, 1 GB RAM): approximately $52 (always-on) to $69 (on-demand) per instance.
Artifact Registry storage: minimal, around $0.10 per GB stored monthly.
Cloud Build for container image builds: low cost, around a few cents per build minute.
Cloud Monitoring and Logging: costs vary with usage; generally modest.
Optional database (e.g., Cloud SQL) pricing varies by instance size.

Add as per the requirements for the hackathon :

- Use minimal resource allocation (e.g., 1 vCPU, 512 MB RAM) to reduce Cloud Run costs.
- Limit continuous runtime; use on-demand scaling to pay only for actual usage during the hackathon.
- Artifact Registry and Cloud Build costs remain low due to limited container builds and image storage.
- Minimize or use free tiers of Cloud Monitoring and Logging within usage limits.
- Use serverless or free-tier databases like Firestore or minimal Cloud SQL instances.
- Estimate overall cost to stay below $20–$50 for the hackathon duration by efficient scaling and minimal resources.

GenAI Exchange Hackathon

Thank you