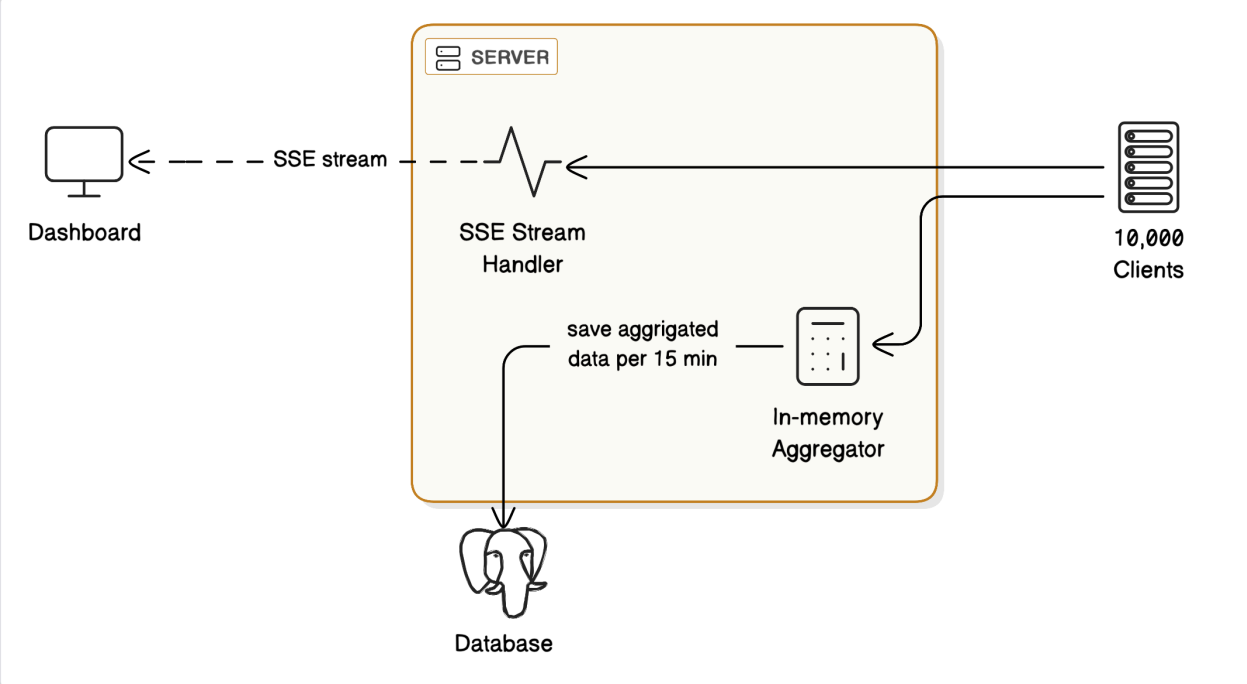
ARCHITCTUR1

System architecture diagram



OSI model layer breakdown of your protocols

Network Layer: IP (IPv4/IPv6) handles routing packets between client, server, and dashboards, ensuring data reaches the correct destination.

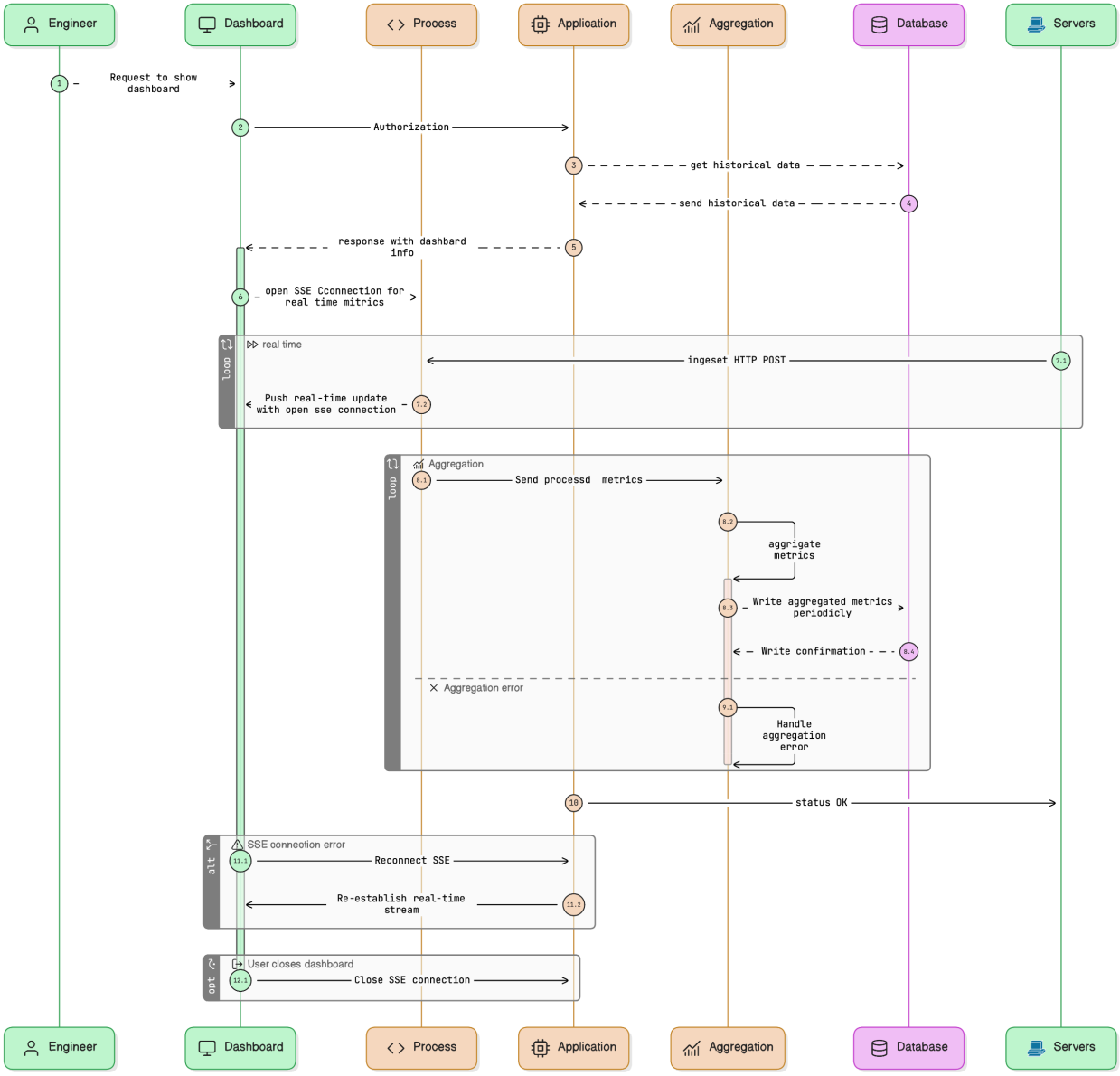
Transport Layer: TCP for HTTP/1 connections between client ↔ server and server ↔ dashboards.

Session Layer: HTTP/1 manages sessions and connection lifetimes for requests and real-time dashboard updates.

Presentation Layer: JSON formatting for request and response payloads.

Application Layer: HTTP/1 POST requests from client to server, Server-side aggregation in memory, HTTP/1 responses / SSE to dashboards for real-time updates

Sequence diagrams for key flows



Why you chose specific patterns

Data Ingestion HTTP POST:

HTTP POST is preferred over GET for metric ingestion because it can handle larger payloads and reduces unnecessary network and server overhead, making it more efficient for for submitting data from clients to the server.

Server → Dashboard (HTTP/1 or SSE):

- Reason: Provides real-time updates to clients in a unidirectional, lightweight way without the complexity.

Aggregation in Memory:

- **Reason:** enabling **lower latency** and faster responsiveness for real-time dashboards.

Single thread (node JS):

- efficiently without the overhead of thread management.
- This ensures non-blocking I/O for clients and dashboards, keeping updates low-latency and responsive.

Design decisions and trade-offs

- **HTTP POST from Client to Server:** it is less efficient for very high-frequency updates because each request incurs network and handshake overhead, including headers and connection setup. this repeated overhead can hit system and network resource limits, whereas streaming protocols amortize connection cost and reduce load.
- **In-Memory Aggregation:**
  - data lost on server crash and scales less well than a dedicated external store.
  - may increasing p95/p99 latency significantly.