1️⃣ Problem definition (1–2 minutes)

- We had telemetry systems independently deployed at each site
- Full stack replicated per site
- High cost, schema drift, fragmented data
- No global analytics or ML readiness

Goal

> Centralize telemetry ingestion and analytics while keeping sites lightweight and resilient.

2️⃣ Functional requirements (3–4 minutes)

- Ingest high-volume telemetry from hundreds of devices per site
- Support multiple global sites
- Support SOC queries (current, average, range)
- Retain data for 3 years

3️⃣ Non-functional requirements (2–3 minutes)

Highlight only what drives architecture:

- High availability
- Eventual consistency is acceptable
- Write-heavy workload
- Large data volume and long retention
- Geo-distribution
- Security (IDs only, encrypted)

This justifies **event-driven + TSDB**.

4️⃣ APIs (2 minutes)

You did this well. Keep it short.

- No ingestion APIs (internal pub-sub)
- Query APIs for dashboards
- Metadata service for joins

Mention Prometheus **only to reject it**. That shows judgment.

5️⃣ Data model + partitioning (5 minutes)

Explain:

- Time-series key: site, device, metric, timestamp
- Partition by site + minute bucket
- Why minute bucket is the right tradeoff
- How queries work within and across partitions

This naturally leads into HLD.

6️⃣ High-level design (5–7 minutes)

You already have a solid diagram.

Walk through **one clean write path** and **one read path**:

- Device service → Event Hub → Stream processor
- Stream processor → TSDB (hot) + Blob (cold)
- Query service → cache → TSDB / blob → dashboard

Avoid enumerating every box. Focus on flow.

7️⃣ Deep dive (10–15 minutes)

This is where you **slow down**.

You chose the *right* deep dive:

> End-to-end ingestion correctness under retries and late/out-of-order events

Cover only:

- At-least-once delivery
- Stable event identity
- Dedup
- Raw vs aggregates

- Hybrid correction approach

This is where interviewers engage.

8️⃣ Tradeoffs and closing (2 minutes)

End strong.

- Eventual consistency over strict real-time accuracy
- Slightly more backend complexity
- Much lower long-term operational cost
- Future-ready for ML and analytics