TransitStream Master Workflow (Persistent Guide)

🎯 Project Vision

> Build **TransitStream**, a real-time transportation platform (bus/train fleet telemetry + customer-facing ETAs) using **Kafka, Spark Streaming, Postgres, Schema Registry, FastAPI, and Observability (Prometheus/Grafana)**.
> It should mimic industry best practices: **event sourcing, CQRS, exactly-once semantics, schema evolution, CI/CD, monitoring, multi-service orchestration**.

🏗️ System Architecture (static blueprint)

- **Data ingestion**: Simulated GPS → Kafka (`vehicle.position.v1` Avro).
- **Processing**: Spark Structured Streaming jobs (ETA compute, anomaly detection).
- **Storage**: Kafka (raw), Postgres (read models).
- **Serving**: FastAPI APIs + WebSocket feed.
- **Coordination**: Kafka topics link microservices.
- **Observability**: Prometheus, Grafana dashboards.
- **CI/CD**: GitHub Actions, Docker images, integration tests.

📍 Milestone Roadmap

Milestone 0 – Infra up

- Docker Compose with Kafka (KRaft), Schema Registry, Postgres, Prometheus, Grafana.
- Verify: topics, schema registry health, Grafana accessible.

Milestone 1 – Ingestion & Event Sourcing

- FastAPI GPS ingestor → Kafka with Avro (Schema Registry).
- Use **transactional/idempotent producers**.
- DLQ for bad events.

Milestone 2 – Stream Processing

- Spark jobs: enrich positions, compute ETAs, detect anomalies.
- Exactly-once Kafka→Kafka sink.

Milestone 3 – CQRS Read Models

- Consumers project data into Postgres.
- Upserts ensure idempotency.
- Read models: `vehicle_state`, `stop_eta`.

Milestone 4 – Serving Layer

- FastAPI API: `/eta`, `/vehicles/:id`.
- WebSocket: `/ws/live` for real-time feed.

Milestone 5 – CI/CD

- GitHub Actions pipeline: lint, test, build Docker images, run integration tests with Compose.

Milestone 6 – Observability

- Prometheus scrapes Kafka, Spark, FastAPI.
- Grafana dashboards (consumer lag, API latency, job health).

Milestone 7 – Advanced Patterns

- Event sourcing & CQRS fully applied.
- Schema evolution (backward compatible).

- Exactly-once semantics revisited.
- Stream joins (positions ⋈ routes).

🧭 Rules of Engagement (how I'll guide you)

- **Explain, then code**: I'll always tell you *what this block does, why we need it, and where it fits in the bigger puzzle*.
- **Progress markers**: At the end of each step, I'll mark "✅ Done, Next → …".
- **Debugging support**: If something breaks, I'll walk you through *what to check, where it might fail, and how to verify*.
- **Checkpoint reminders**: I'll remind you where we are in the roadmap (Milestone N, Step X).

📝 Checkpoints & Progress Markers

Keep track of your progress like this (just edit your copy as you go):

```
Progress Log:
- Milestone 0: Infra up ✅
- Milestone 1: Ingestor started ⏳
- Milestone 2: Spark jobs pending
- Milestone 3: ...
```

Whenever you come back after a break, just paste:

- The **Master Workflow** (this doc).
- Your **Progress Log**.

That instantly puts me back into your context.