CI/CD Overview (Current State)

This document reflects the CI/CD that actually exists in the repo today: per-service GitHub Actions pipelines that lint/test/build/publish images to GHCR, plus local/manual deploy paths via Docker Compose and Argo CD on the local minikube cluster.

---

1) GitHub Actions Workflows

- **Lint** (`.github/workflows/lint.yml`)
  - Trigger: PRs and pushes to `main`.
  - Runs `golangci-lint` in a matrix across all Go modules (`services/auth`, `gateway`, `crypto-core`, `keys`, `messages`).
- **Service CI/CD** (one per service: `auth.yaml`, `gateway.yaml`, `keys.yaml`, `messages.yaml`, `crypto-core.yaml`)
  - Trigger: PRs and pushes to `main` affecting the service (and related k8s overlay paths).
  - Jobs:
    - `build-and-test` : `go test ./...` for the service.
    - `sonarqube-analysis` : coverage + scan on a self-hosted runner (requires `SONAR_TOKEN`, `SONAR_HOST_URL`).
    - `build-and-push` (push-only): builds and pushes images to GHCR with `latest` and `${{ github.sha }}` tags.
  - Image names: `ghcr.io/klickk/secumsg-server/<service>[:latest|:<sha>]` (e.g., `auth`, `gateway`, `keys`, `messages`). Migrations images are built where present (auth/keys/messages).

Not present: a central monolithic CI workflow or an automated deploy workflow; deployments are manual (Compose) or handled by Argo CD on the local cluster.

---

2) Build & Publish Flow

1. Commit/PR triggers lint + service-specific CI.
2. On pushes to `main`, each service workflow builds its Docker image and publishes to GHCR with two tags: `latest` and `${{ github.sha }}`.
3. SonarQube runs on a self-hosted runner after tests to upload coverage and perform static analysis.

---

3) Environments & Deployments

- **Local Dev (Docker Compose)**
  - `.docker/docker-compose.dev.yml` spins up Postgres, migration jobs, and the services (auth, keys, messages, gateway).
  - `.docker/docker-compose.observability.yml` optionally adds Prometheus, Loki/Promtail, and Grafana.
  - `Makefile` shortcuts: `up`, `logs`, `down`, `migrate-*`.
- **Local/Manual Prod (Docker Compose)**
  - `.docker/docker-compose.prod.yml` consumes an env file for secrets (`POSTGRES_PASSWORD`, `SIGNING_KEY`, etc.) and pulls GHCR images.
  - No automated GH Actions deploy job; run Compose manually on the target host.
- **Local Minikube via Argo CD**
  - Application manifest: `infra/argocd/sem7-platform-minikube.yaml`.
  - Deploys `k8s/overlays/minikube` (auth, keys, messages, gateway) into the minikube cluster.
  - Argo CD sync is automated with prune/self-heal; Argo CD Image Updater tracks GHCR images (`auth`, `keys`, `messages`, `gateway`) using the `latest` strategy and writes back to `main`.
  - Requires Argo CD + Image Updater running in the cluster.

---

4) Secrets, Variables, and Credentials

- **GitHub Secrets**
  - `SONAR_TOKEN`, `SONAR_HOST_URL` for SonarQube scans.
  - Workflows use `GITHUB_TOKEN` to push images to GHCR.
- **Compose env (manual)**
  - Provide `POSTGRES_PASSWORD`, `SIGNING_KEY`, CORS origins, etc., via `.docker/.env.prod` when running the prod stack.
- **Argo CD**
  - Uses GHCR public auth via repo visibility; if images are private, configure image pull secrets in the cluster.

---

5) How to Run Locally

```
 # Dev stack with observability
make up
docker compose -f .docker/docker-compose.observability.yml up -d

# Tail logs
make logs

# Tear down
make down
```

For CI parity, run `golangci-lint` per module and `go test ./...` inside each service directory.
