Edge, Centric Service Orchestration and Management

Modern operations for instance in both Industry 4.0 (factory lines, machine vision, OPC-UA/MQTT telemetry) and energy & utilities (solar farms, batteries, EV charging, pumping stations) demand decisions on site, with tight latency, data locality, and the ability to keep running through back-haul outages. This shifts analytics, control logic, and buffering to edge systems, fleets of heterogeneous gateways and small servers, where the core challenge is orchestration at scale: getting the right services to the right site, updating them safely, and proving they're healthy. Key risks include fragile rollouts, supply-chain tampering, secrets sprawl, and privacy leakage across sites. The solution is a disciplined edge platform: declarative deployments (GitOps), progressive delivery (canary/blue-green with health gates and automatic rollback), and a secure CI/CD lane with signed images, provenance/SBOMs, vulnerability scans, and per-site secrets. Runtime controls, mTLS between services, least-privilege identities, network policies, encrypted storage, and observability (SLOs, update success rates, drift and anomaly alerts) turn security and reliability into day-to-day practice. Finally, resilience patterns such as local queues and rules engines, offline cache/sync, and site quarantine ensure factories stay productive and energy assets stay safe, even when the cloud is unreachable.

1.1 What a student will learn

Students learn to evaluate and implement container orchestration on resource, constrained, heterogeneous edge devices, comparing centralized (cloud, managed) with distributed (edge, native) control planes. They design benchmarks for performance and resilience on varied hardware and build judgment around cloud-fog-edge trade, offs for use cases like video analytics and industrial IoT.

On top of that, students practice edge, appropriate CI/CD: artifact versioning and image signing; environment, aware manifests; progressive delivery (canary, blue/green, waves by site/region); health checks and automatic rollback; SBOM generation and vulnerability scanning; secrets management; and attestation of builds and nodes. They also develop a threat, informed mindset, applying lightweight threat modelling (e.g., STRIDE/attack, trees) to edge scenarios, covering supply, chain risks, exposed local services, credential misuse, data, at, rest/on, wire protection, and safety impacts, and then mapping mitigations into the pipeline and runtime controls.

1.2 Example study materials

- Edge orchestration & tools: K3s or MicroK8s docs; container runtime and registries; GitOps with Argo CD or Flux (progressive delivery, health checks, rollback).
- CI/CD & supply chain security: Docker/OCI hardening guides; SBOM basics (CycloneDX, SPDX); SLSA/Sigstore (cosign/fulcio/rekor) for signing and provenance; secrets management patterns (e.g., sealed secrets, vault, backed).
- Cybersecurity foundations for edge/OT: Short primers on STRIDE and attack trees; zero, trust at the edge (device identity, mutual TLS); overviews of relevant controls from NIST/IEC 62443 (lightweight, practice, oriented selections).
- Surveys/case studies: Energy, site edge deployments (utilities, EV charging) focusing on outage tolerance, local buffering, and privacy constraints.

1.3 Capstone project proposal

Design and implement a working orchestration approach for a small fleet (8–15 nodes) of edge gateways representing energy sites (e.g., PV inverters, battery rooms, EV chargers, a pumping station). Package three to four services as containers, telemetry ingestion (MQTT/OPC, UA → time, series DB), on, site anomaly detection or forecast inference, a rules/controls microservice, and a minimal dashboard/API.

Build two control, plane variants: (A) centralized (one cluster managing remote nodes) and (B) distributed (per, site lightweight clusters with remote coordination). Demonstrate declarative rollout/rollback and safe behaviour during back, haul outages (local buffering, continued control).

Add a secure CI/CD lane:

- Build artifacts in a reproducible pipeline; generate an SBOM, run image and dependency vulnerability scans, and fail the build on critical findings.
- Sign container images and manifests; verify signatures at deploy time; annotate rollouts with provenance.
- Use progressive delivery (canary → waves by site) with automated health gates (latency, error rate, control, loop timeliness) and automatic rollback.
- Manage secrets (per, site credentials, broker keys) without embedding them in images; rotate them during the project.

Include a threat model & security tests:

- Produce a concise threat model for the fleet: assets, trust boundaries, attack paths (e.g., poisoned image, stolen site key, exposed admin API, rogue node), and prioritized risks.
- Implement at least three concrete mitigations tied to those risks (e.g., network policy isolation, mTLS between services, signed updates with policy, enforced verification, least, privilege service accounts, encrypted storage).
- Validate with adversarial tests: unsigned image blocked; tampered manifest rejected; revoked node cannot join; compromised pod contained by policies.

Benchmark and compare the two variants on deployment latency, service recovery time, resource footprint (CPU/RAM/IO), and update success under induced faults (packet loss, node reboot, partial partitions). Deliver a ~12–15 page engineering report that ties CI/CD and security controls to resilience and privacy requirements, plus a short runbook for “emergency patch,” “credential rotation,” and “site quarantine.”

1.4 Prerequisite knowledge

Comfort with Linux and the command line, Docker/ Kubernetes containerised apps, and basic distributed, systems and networking concepts (TLS/mTLS, service discovery, ingress). Familiarity with Git and CI/CD is helpful; curiosity about security and observability will make the project smoother.

1.5 Contact information

Ambient Intelligence Research Group, Saxion University of Applied Sciences, Saxion. **Website:** http://www.saxion.nl/ami. **Contact:** Javier Ferreira Gonzalez, j.ferreiragonzalez@saxion.nl