ADR-003: Kubernetes Platform

Submitters

- Luke Doyle (D00255656)
- Hafsa Moin (D00256764)

Change Log

- approved 2026-02-10

Referenced Use Case(s)

- UVote Operational Platform Requirements - Container orchestration, environment separation, network policy enforcement, and secret management for the UVote election system.

Context

The UVote system requires a container orchestration platform that supports multi-replica deployments, health checking, rolling updates, network policy enforcement, namespace-based environment separation, and secret management. The platform must also satisfy the PROJ I8009 module requirement for an "operational platform on cloud-native technologies (containers, Kubernetes, or serverless)" and demonstrate DevOps competencies including Infrastructure as Code, resilience, and automated deployment.

Docker Compose was used during initial development for local multi-service orchestration. While adequate for starting containers together, Compose lacks the production-grade features required by the module brief: network policies for service isolation, health-based traffic routing, rolling update strategies, and namespace-based environment separation. The module assessment rubric specifically allocates marks for provisioned infrastructure using Infrastructure as Code and resilience evidence, both of which require an orchestration platform beyond Docker Compose.

The selected approach is Kubernetes, deployed locally using the Kind (Kubernetes IN Docker) distribution. All cluster configuration is version-controlled as YAML manifests, and the Calico CNI is used for network policy enforcement.

Proposed Design

Services and modules impacted:

All six UVote microservices are deployed as Kubernetes Deployments within the `uvote-dev`, `uvote-test`, and `uvote-prod` namespaces. Each service is configured with liveness and readiness probes, resource limits, and a minimum of 2 replicas for high availability.

Cluster topology:

- Distribution: Kind (see ADR-012 for distribution selection rationale)
- Nodes: 1 control-plane, 2 workers
- CNI: Calico v3.26.1 (default CNI disabled)
- Namespaces: `uvote-dev`, `uvote-test`, `uvote-prod`
- Storage: PersistentVolumeClaim (5Gi) for PostgreSQL
- Ingress: Nginx Ingress Controller via Helm

Kind cluster configuration (`kind-config.yaml`):

```
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
  - role: control-plane
  - role: worker
```

```
  - role: worker
networking:
  disableDefaultCNI: true  # Required for Calico
  podSubnet: "192.168.0.0/16"  # Calico's default CIDR

```

API and configuration impact:

All cluster resources are defined as version-controlled YAML manifests. Deployment is automated via Python scripts. Kubernetes Secrets store database credentials and JWT keys (see ADR-011).

DevOps impact:

Docker Compose remains in the repository for rapid local development without Kubernetes. Developers can use Compose for code iteration and Kubernetes for integration testing and platform demonstrations.

Considerations

Docker Swarm (rejected): Docker Swarm provides basic clustering and service replication using familiar Docker CLI commands. It was rejected because it has no NetworkPolicy support, which is a hard requirement for the zero-trust security model. Swarm also lacks readiness probes, namespace isolation, and the rolling update strategies required by the module brief. Its industry adoption has declined as Docker Inc. shifted focus to Docker Desktop.

Docker Compose only (rejected): Docker Compose is already in use for local development and has near-zero learning curve. It was rejected as a production platform because it provides no replicas, no health probes, no rolling updates, no network policies, no namespace isolation, and no secret management beyond plaintext values in YAML or `.env` files. It does not satisfy the module requirement for a cloud-native operational platform.

HashiCorp Nomad (rejected): Nomad is a lighter-weight orchestrator that can schedule containers alongside other workload types. It was rejected because it has no native NetworkPolicy equivalent (service mesh via Consul Connect would be required), it has significantly less industry adoption than Kubernetes, and it requires learning HCL configuration in addition to its own concepts. The smaller ecosystem means fewer ready-made solutions for monitoring, logging, and CI/CD integration.

Learning curve: Kubernetes concepts including pods, services, deployments, network policies, RBAC, and persistent volumes require a significant learning investment. This is mitigated by starting with Kind as the simplest local distribution, automating setup with Python scripts, and documenting every configuration decision. The learning curve is also considered a feature in this context, as the module explicitly requires demonstrating Kubernetes competency.

Resource overhead: Running a 3-node Kind cluster with Calico, 6 services, and PostgreSQL requires approximately 4 to 6 GB of RAM. This is within the 16 GB constraint of the development machine. Resource limits are set on all pods, and 2 replicas per service are used as the minimum for high availability rather than 3.

Debugging complexity: The multi-layer abstraction of Kubernetes makes troubleshooting more involved than Docker Compose. This is mitigated by health endpoints on all services, comprehensive logging, and troubleshooting documentation in `PLATFORM.md`.

Decision

Kubernetes was selected as the container orchestration platform for UVote.

Kubernetes is the only option evaluated that satisfies all requirements. The primary driver is module alignment: the PROJ I8009 brief explicitly requires an operational platform on cloud-native technologies, and Kubernetes is the canonical implementation of this requirement. The module assessment rubric allocates marks for Infrastructure as Code and resilience evidence, both of which are first-class Kubernetes features.

The second driver is network policy enforcement. The zero-trust security model (ADR-010) requires NetworkPolicies enforced by a CNI plugin. Of the options considered, only Kubernetes provides this capability natively. Calico is selected as the CNI for its mature NetworkPolicy support and compatibility with Kind.

The third driver is industry relevance. According to the CNCF Survey 2023, 96% of organisations adopting container orchestration use Kubernetes. Demonstrating Kubernetes competency is directly relevant for employment after graduation.

Replicas, health probes, rolling updates, Secrets, and namespaces are all first-class Kubernetes features that require no workarounds or third-party additions to satisfy the project requirements.

The trade-off of increased operational complexity and higher resource usage over Docker Compose is accepted. Compose is a development tool and is fundamentally inadequate as an operational platform for the project's requirements. The same Kubernetes manifests, with minor modifications, can be used to deploy UVote to a cloud Kubernetes service such as AKS or EKS.

A review is scheduled for the end of Stage 2 (April 2026) to evaluate migration to a cloud Kubernetes service for production.

Other Related ADRs

- ADR-001: Python FastAPI Backend - services deployed to the cluster
- ADR-002: PostgreSQL Database - PersistentVolumeClaim and database pod
- ADR-004: Calico CNI - CNI plugin installed on this cluster
- ADR-010: Zero-Trust Network Security - NetworkPolicy resources deployed to this cluster
- ADR-011: Secrets Management - Kubernetes Secrets for credentials and keys
- ADR-012: Kind Distribution - local Kubernetes distribution selection

References

- Kubernetes Documentation
- Kind Documentation
- CNCF Survey 2023 - Kubernetes adoption statistics
- Calico Documentation
- EdgeX Foundry ADR Template