ADR-012: Kind for Local Development

Submitters

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

Change Log

- approved 2026-02-14

Referenced Use Case(s)

- UVote Local Development Platform Requirements - Local Kubernetes distribution for development and Stage 1 demonstration of the UVote system.

Context

Having selected Kubernetes as the orchestration platform (ADR-003), a local Kubernetes distribution must be chosen for development and Stage 1 demonstrations. Cloud-managed Kubernetes such as AKS, EKS, or GKE is not feasible at this stage due to budget constraints. The development environment is a single machine with 16GB RAM and 12 CPU cores, Luke's laptop.

Kind (Kubernetes IN Docker) is a tool that runs Kubernetes clusters locally using Docker containers as nodes. It is primarily designed for testing and development purposes, providing a lightweight and efficient way to simulate a Kubernetes environment without the overhead of virtual machines or cloud infrastructure.

Kind was introduced to the team through the DkIT DevOps module, where the lecturer used it as the local Kubernetes environment for coursework. This direct teaching exposure meant the team already had working familiarity with Kind's cluster configuration, image loading workflow, and lifecycle commands before the project began. Given the machine constraints and the need to keep local tooling lightweight, Kind was the natural starting point for evaluation.

Proposed Design

Cluster topology:

- Cluster name: `evote`
- Nodes: 3 (1 control-plane, 2 workers)
- Pod subnet: `192.168.0.0/16` (Calico default)
- Port mappings: 80 to 80, 443 to 443 (for Nginx Ingress)
- CNI: disabled (Calico installed separately, see ADR-004)

Cluster configuration (`uvote-platform/kind-config.yaml`):

```yaml
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
  - role: control-plane
    extraPortMappings:
      - containerPort: 80
        hostPort: 80
      - containerPort: 443
        hostPort: 443
  - role: worker
  - role: worker
networking:
  disableDefaultCNI: true
  podSubnet: "192.168.0.0/16"

```

Cluster lifecycle commands:

```
# Create
kind create cluster --config kind-config.yaml --name evote

# Load locally-built images into cluster nodes
kind load docker-image auth-service:latest --name evote
kind load docker-image voting-service:latest --name evote

# Delete
kind delete cluster --name evote

```

Local image loading:

`kind load docker-image` loads locally-built Docker images directly into Kind nodes without requiring a container registry. This keeps the build-deploy cycle self-contained on the development machine.

Automated setup:

The `plat_scripts/setup_k8s_platform.py` script encapsulates the full cluster setup: Kind creation, Calico installation, namespace creation, secret deployment, database deployment, and network policy application. This reduces manual cluster setup to a single script invocation.

Integration points:

- ADR-003 (Kubernetes): Kind is the local Kubernetes distribution for Stage 1
- ADR-004 (Calico): Calico CNI installed on the Kind cluster after creation
- ADR-008 (Microservices): all six services deployed to the Kind cluster as Deployments
- ADR-011 (Secrets): secrets created during cluster bootstrapping via the setup script

Considerations

Minikube (not selected): Minikube is a widely documented local Kubernetes tool with a built-in add-on manager, dashboard, and support for multiple drivers including VM and Docker. It was not selected because its multi-node mode is experimental rather than stable, and single-node operation limits the realism of network policy enforcement and pod scheduling across worker nodes. Custom CNI configuration also requires specific driver configuration that is less straightforward than Kind's `disableDefaultCNI` flag.

K3s (not selected): K3s is a lightweight, production-capable Kubernetes distribution that replaces etcd with SQLite and bundles all components into a single binary. It uses less RAM than a standard Kubernetes installation. It was not selected because its default CNI is Flannel, which must be manually removed before Calico can be installed, adding setup complexity. K3s also uses containerd directly rather than Docker, which means locally-built images must be imported differently from Kind's `kind load` command. The team's existing familiarity with Kind and Docker made K3s a less practical fit.

Docker Desktop Kubernetes (not selected): Docker Desktop includes a built-in single-node Kubernetes cluster that can be enabled with a single click. It was not selected because it is single-node only and does not support disabling the default CNI, making it incompatible with both the multi-node requirement and Calico installation.

Not production-grade: Kind is designed for development and testing, not production workloads. The same Kubernetes manifests used in the Kind cluster will deploy to AKS or EKS for production in Stage 2 without modification, so this is not a long-term limitation.

Docker container nodes vs VMs: Kind nodes are Docker containers rather than full virtual machines, meaning some kernel-level features differ from a production environment. For the purposes of NetworkPolicy enforcement, application development, and platform demonstration, this distinction is not significant.

Decision

Kind was selected as the local Kubernetes distribution for UVote Stage 1.

The primary driver is the combination of direct teaching exposure and lightweight resource requirements. Kind was taught in the DkIT DevOps module, giving the team prior familiarity with its configuration and workflow before the project began. This removed the setup and learning overhead that Minikube or K3s would have introduced. The DevOps module context also means Kind is a well-understood and endorsed choice for this type of academic project.

Kind is also the only evaluated option that provides stable multi-node clusters with first-class support for disabling the default CNI, both of which are required for the Calico network policy setup in ADR-004. The `disableDefaultCNI: true` flag is a documented, stable Kind feature rather than a workaround, and the Kind documentation includes a dedicated Calico quickstart guide.

The resource footprint is well within the machine constraints. Three Kind nodes consume approximately 1.5GB of RAM, leaving the remainder available for application services and development tooling. The ephemeral cluster lifecycle (create, develop, delete) fits naturally with the `deploy_platform.py` automation approach and with the secret generation strategy in ADR-011.

Kind is not production-grade, and this is accepted. The project will evaluate migration to a cloud Kubernetes service at the end of Stage 2. The manifests, policies, and configuration developed against Kind will transfer directly to that environment.

A review is scheduled for the end of Stage 2 (April 2026) to evaluate migration to cloud Kubernetes (AKS or EKS).

Other Related ADRs

- ADR-003: Kubernetes Platform - platform decision for which Kind is the local distribution
- ADR-004: Calico CNI - CNI installed on the Kind cluster
- ADR-008: Microservices Architecture - services deployed to the Kind cluster
- ADR-011: Kubernetes Secrets - secrets provisioned during Kind cluster bootstrapping

References

- Kind Documentation
- EdgeX Foundry ADR Template