U-Vote: Platform Scripts

Scope: Full platform test coverage across cluster setup, network policy validation, database security, and service deployment

---

Overview

Three scripts must be run in order. Each validates a different layer of the platform and is a prerequisite for the next.

```
setup_k8s_platform.py  ->  tests/test_db.py  ->  deploy_platform.py
```

Why Python?

All three scripts are written in Python rather than Bash or PowerShell. The team is split across Linux and macOS, and Python runs without modification on both. This also matches the recommendation from our main DevOps lecturer roughly three years ago -- write once, run anywhere, and avoid the shell compatibility issues that come with maintaining separate scripts per OS.

Script 1: `setup_k8s_platform.py`

Provisions the U-Vote Kubernetes infrastructure from scratch on a local Kind cluster. This script must run before either of the other two -- `test_db.py` depends on a live PostgreSQL pod, and `deploy_platform.py` depends on the cluster, namespaces, and network policies all being in place.

Steps

Step	Function	What it does

0	`check_prerequisites`	Verifies docker, kubectl, kind, helm are on PATH
1	`create_kind_cluster`	Creates Kind cluster `uvote` from `kind-config.yaml`
2	`install_calico`	Applies Calico operator and CNI manifests; waits up to 300s
3	`apply_namespaces`	Applies `namespaces.yaml`
4	`deploy_database`	Applies db-secret, db-pvc, db-deployment manifests
5	`apply_database_schema`	Pipes `schema.sql` into pod via `kubectl exec psql`
6	`apply_network_policies`	Applies all `*.yaml` in `network-policies/` in sorted order
7	`install_ingress_controller`	Installs `ingress-nginx` via Helm
8	`verify_setup`	Checks nodes, Calico pods, PostgreSQL pod, namespaces

Known Limitations

Script ordering is not enforced. Running `test_db.py` or `deploy_platform.py` against an incomplete cluster produces failures that look like test failures rather than setup failures.

Calico pulls from Docker Hub at runtime. A restricted network or no internet access will cause Step 2 to fail or stall. There is no offline fallback.

If Calico times out, the script continues anyway. The 300-second wait in `install_calico` is a soft timeout. If Calico is not ready when it expires, the script logs a warning and moves on. Worker nodes remain NotReady, the PostgreSQL pod cannot be scheduled, the database schema is never applied, and the network policies have nothing to enforce against. If you see unexpected failures in `test_db.py` or Phase 7 of `deploy_platform.py`, check `kubectl get pods -n calico-system` before investigating anything else.

`verify_setup` is informational only. It reports failures but does not exit non-zero. Treat a failing verify as a hard blocker before running `test_db.py`.

`apply_database_schema` masks SQL errors. `psql` exits 0 on partial failure, so this step returns True either way. Schema errors only surface in `test_db.py` Test 3.

Script 2: `tests/test_db.py`

Validates the deployed PostgreSQL database across five categories: infrastructure health, schema correctness, seed data presence, security controls, and performance indexes. All tests execute SQL via `kubectl exec` into the PostgreSQL pod. Test 11 additionally validates network-path connectivity from a separate pod.

Test 1 is a gate-keeper -- if the PostgreSQL pod is not Running, the suite exits without running anything further.

Test Cases

#	Test	Pass Recommended CI Ordercriteria
1	`test_pod_running`	Pod is Running with 1/1 containers ready -- exits suite on failure
2	`test_connection`	`SELECT version()` returns a row containing `PostgreSQL`
3	`test_tables_exist`	All 7 tables present: `admins`, `elections`, `candidates`, `voters`, `voting_tokens`, `votes`, `audit_logs`
4	`test_sample_data`	`admins`, `elections`, `candidates` each have at least 1 row
5	`test_vote_immutability`	UPDATE and DELETE on votes both raise trigger error; INSERT succeeds
6	`test_hash_generation`	`vote_hash` is non-NULL in at least one of the last 5 votes after INSERT
7	`test_user_permissions`	`auth_service` can SELECT admins, denied on votes; `results_service` can SELECT votes, denied on INSERT

8	`test_complex_queries`	Multi-table JOIN returns rows with a `candidate` column
9	`test_indexes`	At least 1 of: `idx_votes_election`, `idx_votes_candidate`, `idx_tokens_token`
10	`test_foreign_keys`	FK constraint count > 0 in `information_schema` -- WARN not FAIL if inconclusive
11	`test_connection_from_pod`	Temp pod with label `app=auth-service` connects to `postgresql:5432` via ClusterIP -- skipped with `--quick`
12	`test_load_performance`	Bulk INSERT of N votes succeeds; distribution query returns rows -- off by default

Known Limitations

Test 12 crashed a development laptop. Do not run it on a machine in active use. A dedicated VM or CI runner with at least 16GB RAM is recommended. Votes inserted by this test are permanent -- the immutability trigger blocks any cleanup afterward.

Test 11 requires Calico to have initialised fully during setup. If Calico timed out in Script 1, Test 11 will fail.

Test 7 covers only two of six database roles. `election_service`, `voting_service`, and `admin_service` permissions are not verified.

Script 3: `plat_scripts/deploy_platform.py`

Builds Docker images for the six U-Vote microservices, loads them into the Kind cluster, manages Kubernetes secrets, deploys service manifests, and runs network connectivity and health endpoint tests across 9 sequential phases.

Phases

Phase	What it does
1	Pre-flight checks -- cluster, Docker, namespace, PostgreSQL pod

2	Builds all 6 service images (600s timeout per build)
3	Loads images into Kind via `kind load docker-image`
4	Creates or updates `db-credentials`, `jwt-secret`, `flask-secret`; syncs PostgreSQL password
5	Applies service manifests from `uvote-platform/k8s/services/`
6	Polls pod readiness every 5s up to `--timeout` (default 300s)
7	Tests TCP connectivity from each backend to `postgresql:5432`; frontend connection expected to be blocked
8	Opens port-forward tunnel per service and checks health endpoint for HTTP 200
9	Prints pass/fail counts and declares overall result

Execution Order

```
# Step 1: Provision infrastructure
python3 plat_scripts/setup_k8s_platform.py

# Step 2: Validate database (skip slow network test)
python3 tests/test_db.py --quick

# Step 3: Deploy services and run smoke tests
python3 plat_scripts/deploy_platform.py --verbose

# Step 4: Full database test including pod-to-pod network path
python3 tests/test_db.py
```