INTELLIGENT DATA QUALITY ENGINE (IDQE)

Enterprise Installation Guide (Test and Production)

Date: February 7, 2026

## 1. Scope

This guide describes how to install and operate IDQE in:
* Enterprise Test environment
* Enterprise Production environment

## 2. Solution Components

* ‘workflow-client‘: web UI (workflow execution, DQ rules editor, MCP session, LLM config)
* ‘mcp-server‘: MCP orchestration/API, data-source access, workflow persistence, suggestion approvals
* ‘dq-engine‘: assessment/correction engine
* ‘postgres‘: workflow metadata + source data

## 3. Deployment Topology

Recommended enterprise topology:
* Kubernetes namespace ‘idqe‘
* Ingress/API gateway in front of ‘workflow-client‘ and ‘mcp-server‘
* Managed PostgreSQL (HA, backups, encryption)
* Central secrets management for DB and API keys
* Observability stack (logs, metrics, traces)

Manifests provided:
* ‘./deploy/k8s/namespace.yaml‘
* ‘./deploy/k8s/dq-engine.yaml‘
* ‘./deploy/k8s/mcp-server.yaml‘
* ‘./deploy/k8s/workflow-client.yaml‘
* ‘./deploy/k8s/configmap-llm.yaml‘

## 4. Prerequisites

### 4.1 Test Environment
* Non-production Kubernetes cluster
* Registry access for container images
* PostgreSQL test database
* DNS + TLS for test ingress

### 4.2 Production Environment
* HA Kubernetes cluster
* Managed PostgreSQL with backup/restore and PITR
* Secret manager (Vault/KMS/cloud-native)
* WAF/API gateway, network policies, RBAC
* SIEM/SOC integration for logs and alerts

## 5. Build and Publish Images

From repository root:

```
    cd .
    docker build -f services/dq-engine/Dockerfile -t <registry>/idqe-dq-engine:1.0.0 .
    docker build -f services/mcp-server/Dockerfile -t <registry>/idqe-mcp-server:1.0.0 .
docker build -f services/workflow-client/Dockerfile -t <registry>/idqe-workflow-client:1.0.0
```

services/workflow-client
docker push <registry>/idqe-dq-engine:1.0.0
docker push <registry>/idqe-mcp-server:1.0.0
docker push <registry>/idqe-workflow-client:1.0.0

6. Configure Environment Values

6.1 DQ Rules
* Test baseline: ‘./config/rules.demo.yaml‘
* Production baseline: ‘./config/rules.prod.yaml‘

6.2 LLM
* Test: use ‘mock‘ or sandbox API key
* Production: use ‘openai‘ and managed key rotation
* Config file: ‘./config/llm.prod.yaml‘

6.3 Secrets
Store outside Git:
* Database URL
* ‘OPENAI_API_KEY‘ (if enabled)

7. Install to Enterprise Test

kubectl apply -f deploy/k8s/namespace.yaml
kubectl -n idqe create secret generic idqe-db
--from-literal=database_url=’postgresql+psycopg2://<user>:<pass>@<host>:5432/dq’
kubectl -n idqe create configmap idqe-llm-config --from-file=llm.yaml=config/llm.prod.yaml -o
yaml --dry-run=client | kubectl apply -f -
kubectl apply -f deploy/k8s/dq-engine.yaml
kubectl apply -f deploy/k8s/mcp-server.yaml
kubectl apply -f deploy/k8s/workflow-client.yaml

Before apply, replace image names in manifests with your registry images.

8. Test Validation Checklist

* UI reachable via ingress
* ‘mcp-server‘ and ‘dq-engine‘ health endpoints OK
* MCP session can connect/list tools
* Assessment and correction run successfully
* Workflow history rows open run details
* DQ rules and LLM config save/load work
* LLM suggestions appear (if enabled), approve/decline persists behavior

9. Install to Production

Recommended rollout:
1. Validate in pre-prod with production-like data
2. Deploy with rolling strategy
3. Canary release (10%, 50%, 100%)
4. Monitor latency/error/throughput and DQ run success rates
5. Roll back immediately if SLO thresholds are breached

Mandatory controls:
* TLS everywhere
* RBAC for rules/LLM edits
* Audit logs for suggestion approvals/declines
* Backup and restore tests
* Alerting on failed workflows and service degradation

## 10. Operations

Daily tasks:
* Review failed workflow runs
* Review pending/approved LLM suggestions
* Track DQ quality index trends

Suggested SLOs:
* API uptime >= 99.9%
* Assessment p95 latency < 3s
* Failed workflow run ratio < 1%

## 11. Rollback Plan

* Revert deployments to previous image tags
* Restore previous rules and LLM config from backup
* Re-run smoke tests and health checks

## 12. Quick Ops Commands

```
kubectl -n idqe get pods
kubectl -n idqe get svc
kubectl -n idqe logs deploy/mcp-server -f
kubectl -n idqe logs deploy/dq-engine -f
kubectl -n idqe rollout restart deploy/mcp-server
```