==
DEMO INSTALLATION GUIDE
docs/install-demo-detailed.md
==

IDQE DEMO INSTALLATION GUIDE (DETAILED)

This guide installs the full demo stack on macOS using Docker Desktop.

1. Prerequisites

* macOS with Docker Desktop running
* Docker Compose v2
* Free ports:
  - ‘8080‘ workflow client
  - ‘8001‘ dq-engine
  - ‘8002‘ mcp-server-a
  - ‘8003‘ mcp-server-b
  - ‘5432‘ postgres-a
  - ‘5433‘ postgres-b

Verify:

```
docker --version
docker compose version
```

2. Project Setup

```
cd .
```

Demo uses:

* ‘docker-compose.demo.yml‘
* ‘config/rules.demo.a.yaml‘
* ‘config/rules.demo.b.yaml‘
* ‘config/llm.demo.yaml‘

3. Start Demo Stack

```
docker compose -f docker-compose.demo.yml up --build
```

Wait until all containers are healthy.

4. Open Endpoints

* Workflow UI: http://localhost:8080
* MCP A: http://localhost:8002/docs
* MCP B: http://localhost:8003/docs
* DQ Engine: http://localhost:8001/docs

5. Demo Login

The app shows Login/Register first.

Demo admin default:

* Email: ‘admin@idqe.local‘
* Password: ‘Admin123!‘

You can also register normal users from the Login/Register page.

6. MCP Session Setup

In ‘MCP Session‘ tab:

1. Connect ‘http://localhost:8002‘.
2. Optional: connect ‘http://localhost:8003‘.
3. Check ‘Connected MCP Servers‘.
4. Use ‘My MCP Servers‘ to save personal endpoints.
5. Use ‘Shared MCP Servers‘ to connect to shared endpoints.

7. Run First Workflow

In ‘Run Workflow‘:

1. Select provider and dataset type.
2. Select one or more target MCP servers.
3. Click ‘Preview Dataset‘.
4. Click ‘Run Assessment‘.
5. Click ‘Run Correction‘.
6. Open ‘Workflow History‘ and view run details.

8. Rules Management

In ‘DQ Rules Editor‘:

1. Click ‘Load Rules‘.
2. Edit assessment/correction rules in table editor.
3. Click ‘Save Rules‘.
4. Review accepted/declined suggestion decisions below.

9. LLM in Demo

Demo UI includes ‘LLM Config‘ tab.

If using OpenAI:

```
export OPENAI_API_KEY=your_key
docker compose -f docker-compose.demo.yml up --build
```

If no key is provided, keep provider ‘mock‘ or ‘none‘.

10. Smoke Test

Run API smoke test:

```
    python3 scripts/test_mcp_flow.py
```

11. Stop / Reset

Stop:

```
    docker compose -f docker-compose.demo.yml down
```

Full reset (containers + volumes):

```
    docker compose -f docker-compose.demo.yml down -v
```

12. Troubleshooting

* Port conflict: stop local process/container using same port.
* Login/session issues: logout and login again.
* Missing data: confirm target server and dataset type.
* Service error: inspect logs:

```
    docker compose -f docker-compose.demo.yml logs -f
```

==
PRODUCTION INSTALLATION GUIDE
docs/install-production-detailed.md
==

IDQE PRODUCTION INSTALLATION GUIDE (DETAILED)

This guide describes a production setup pattern for IDQE.

1. Production Architecture

Recommended services:

* ‘workflow-client‘ (web UI)
* ‘mcp-server‘ (API/orchestration)
* ‘dq-engine‘ (assessment/correction)
* Managed PostgreSQL (HA, backup, PITR)
* Ingress/API Gateway + TLS

Important production behavior:

* Login is required.
* ‘LLM Config‘ tab is hidden in UI.
* LLM is configured via file only.

2. Security Requirements

* Strong ‘AUTH_SECRET‘
* ‘AUTH_MODE=production‘
* ‘AUTH_REQUIRED=true‘
* Restricted ‘CORS_ALLOW_ORIGINS‘

* Secrets in secret manager (not Git)
* Network policies and least privilege

3. Build and Publish Images

```
    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
```

4. Prepare Configuration Files

* Rules: ‘config/rules.prod.yaml‘
* LLM: ‘config/llm.prod.yaml‘

In production, update only the YAML files and redeploy/restart relevant services.

5. Required Environment Variables

For ‘mcp-server‘:

* ‘AUTH_MODE=production‘
* ‘AUTH_REQUIRED=true‘
* ‘AUTH_SECRET=<strong secret>‘
* ‘ACCESS_TOKEN_TTL_MINUTES=120‘ (or stricter)
* ‘UI_ALLOW_LLM_TAB=false‘
* ‘CORS_ALLOW_ORIGINS=https://<your-ui-domain>‘
* ‘DATABASE_URL=postgresql+psycopg2://...‘
* ‘DQ_ENGINE_URL=http://dq-engine:8001‘

6. Deploy with Compose (Production Profile Template)

Update image names in ‘docker-compose.prod.yml‘, then run:

```
    cd .
    export AUTH_SECRET=’<strong secret>’
    export CORS_ALLOW_ORIGINS=’https://idqe.company.com’
    docker compose -f docker-compose.prod.yml up -d
```

Use this as a baseline. For enterprise, prefer Kubernetes manifests in ‘deploy/k8s‘.

7. Deploy with Kubernetes (Recommended)

1. Create namespace:

```
    kubectl apply -f deploy/k8s/namespace.yaml
```

2. Create DB and app secrets:

```
kubectl -n idqe create secret generic idqe-db
--from-literal=database_url=’postgresql+psycopg2://<user>:<pass>@<host>:5432/dq’
```

```
kubectl -n idqe create secret generic idqe-auth --from-literal=auth_secret=<strong secret>
```

3. Create config maps:

```
kubectl -n idqe create configmap idqe-rules --from-file=rules.yaml=config/rules.prod.yaml -o
yaml --dry-run=client | kubectl apply -f -
kubectl -n idqe create configmap idqe-llm --from-file=llm.yaml=config/llm.prod.yaml -o yaml
--dry-run=client | kubectl apply -f -
```

4. Update image references in:

* ‘deploy/k8s/dq-engine.yaml‘
* ‘deploy/k8s/mcp-server.yaml‘
* ‘deploy/k8s/workflow-client.yaml‘

5. Apply manifests:

```
    kubectl apply -f deploy/k8s/dq-engine.yaml
    kubectl apply -f deploy/k8s/mcp-server.yaml
    kubectl apply -f deploy/k8s/workflow-client.yaml
```

8. Post-Deployment Validation

* Health endpoints return ‘ok‘.
* Login works.
* Non-admin users cannot access admin features.
* Workflow runs are user-scoped.
* Shared MCP servers managed by admin only.
* LLM tab is not visible in production UI.

9. Operations Checklist

* Backup DB and test restore.
* Monitor API latency/error rates.
* Audit admin changes (users/shared MCP servers).
* Rotate secrets regularly.
* Pin image tags and use controlled rollout.

10. Rollback

* Roll back to previous image tags.
* Restore prior rules/LLM config.
* Restart ‘mcp-server‘ and ‘dq-engine‘.
* Re-run smoke tests and critical user flows.

==
USER MANUAL
docs/user-manual.md
==

IDQE USER MANUAL

This manual explains how business users, data stewards, and admins use IDQE.

1. Login and Roles

When you open the app, start from Login/Register.

* ‘User‘ role:
  - Run workflows
  - Edit own DQ rules
  - Manage own MCP server list
  - Use shared MCP servers
* ‘Admin‘ role:
  - All user capabilities
  - Manage users (role/active)
  - Manage shared MCP server catalog

2. Main Tabs

* ‘Run Workflow‘
* ‘DQ Rules Editor‘
* ‘MCP Session‘
* ‘LLM Config‘ (demo/non-production only)
* ‘Help‘
* ‘Admin‘ (admin only)

3. Run Workflow

Purpose: assess and correct data quality for one or more MCP servers.

Steps:

1. Select provider and dataset type.
2. Enter dataset ID.
3. Select target MCP server(s).
4. Optional: enable auto-correction when data source changes.
5. Click:
   - ‘Preview Dataset‘
   - ‘Run Assessment‘
   - ‘Run Correction‘
   - ‘Workflow History‘

Outputs include:

* quality index
* failed checks
* violations
* corrections applied

4. Workflow Analytics

Click ‘Workflow Analytics‘ in ‘Run Workflow‘.

Available views:

* Quality Index Trend (line)
* Violation Severity Trend (line)
* Rule Application Trend (existing vs suggested approved vs declined)
* Correction Rule Usage split

- DQ Workflow Progress Timeline

Filters:

* Date range: all, 7, 30, 90 days
* Analytics server selection

Hover on chart points to see detailed popup context.

5. DQ Rules Editor

Purpose: maintain assessment and correction rules in editable form.

Steps:

1. Click ‘Load Rules‘.
2. Select or add dataset type.
3. Edit assessment rules:
   - id, field, type, severity, params
4. Edit correction rules:
   - field, type, default
5. Click ‘Save Rules‘.

Lower section shows:

* accepted suggested rules
* declined suggested rules

Use ‘Refresh Suggestion Decisions‘ to reload.

6. MCP Session

Purpose: manage active MCP connectivity and server catalogs.

Functions:

* Connect MCP server by URL
* Initialize active server
* List tools/data sources
* Show data model/dataset/FK relations

Server catalogs:

* ‘My MCP Servers‘: user-owned list
* ‘Shared MCP Servers‘: admin-managed shared list

Admin users can edit shared catalog in the same tab.

7. LLM Configuration (Demo / Non-Production)

If visible:

1. Set provider, model, endpoint, key env, key, temperature.
2. Save config.
3. Run assessment to generate rule suggestions (if enabled).

Production note:

* LLM configuration is file-based only.
* LLM tab is hidden in production UI.

8. Suggestion Queue

After assessment, suggested rules may appear in a review queue.

Actions:

* ‘Approve Suggestion‘: stores rule and applies in later runs
* ‘Decline Suggestion‘: rejects suggestion
* ‘Next Suggestion‘: move through queue

9. Admin Page

Admins can:

* list users
* change role (‘user‘ / ‘admin‘)
* enable/disable users

Changes apply immediately.

10. Troubleshooting

* If actions fail with auth error:
  - login again
* If no data appears:
  - verify active MCP server and dataset type
* If run history is empty:
  - execute at least one assessment/correction first
* If shared servers missing:
  - ask admin to configure shared MCP servers