**MCPX‑KendoBridge — CI/CD Plan (GitHub‑first; Alpha → Beta → RTM → Prod)**

**Document:** docs/10\_ci\_cd.docx  
**Project Code:** MCPX‑KendoBridge  
**Version:** 2.0.0 (Kendo Migration)  
**Last Updated:** 2025‑09‑27  
**Owner:** DoSE (Accountable) — SRE Lead (Responsible) — DocFactory (Author)

**Purpose.** Define a **merge‑queue‑aware**, **audit‑ready** CI/CD path that builds, verifies, and promotes MCPX‑KendoBridge across **Alpha → Beta → RTM → Prod**, with **RTM validating against the Prod DB (read‑only)**. The plan embeds required gates: **Build/Tests**, **OpenAPI lint/diff**, **CodeQL**, **Dependency Review**, **Secret Scanning** (repo setting), **SBOM**, and **Evidence Pack** assembly.

**DB & Secrets Compliance (applies to all pipelines):**  
**Add‑only** schema; **Stored‑procedure‑only** data access; **No‑Hard‑Coding** of dynamic values (child cmd/args/cwd, allowed origins, timeouts/keep‑alive). All dynamic values come from **SQL Server** via **AppConfig/FeatureFlag** through **sp\_Config\_\***, **sp\_Feature\_IsEnabled**, **sp\_Lookup\_Get**. **Secrets** (SQL connection string, Telerik license) **never** appear in code/docs/DB and are configured only in **GitHub Environments**.

**Table of Contents**

1. Repository Standards & Protections
2. CI Overview (Required Gates)
3. CD Overview (Promotion Flow & Approvals)
4. Secrets & Environment Configuration
5. Evidence Pack (What to Collect)
6. Pipeline Definitions (Embedded YAML)
7. Quality Gates & Release Criteria
8. Rollback/Incident Integration
9. Compliance Mapping
10. Assumptions
11. Next Steps

**1) Repository Standards & Protections**

* **Default branch:** main, protected with **merge queue** (merge\_group events).
* **Required checks** for merge:
  + **Build & Tests** (.NET 8; optional UI build),
  + **OpenAPI lint/diff**,
  + **CodeQL** (C#/JS),
  + **Dependency Review** (fail on **high**),
  + **SBOM** generated,
  + **Secret Scanning** (enabled at repo/org level).
* **CODEOWNERS** enforced for API, DB, workflows, and docs paths.
* **PR template** includes **No‑Hard‑Coding** & **SP‑only** checklist and Evidence hooks.

**2) CI Overview (Required Gates)**

**CI triggers:** push to main, pull\_request, merge\_group, manual (workflow\_dispatch).  
**Jobs (summary):**

1. **Build & Test (.NET 8)** — compile with warnings‑as‑errors; run unit/integration tests; upload results.
2. **UI Build (optional)** — if KendoReact UI exists, inject **Telerik license** at build time from secrets; run **axe** a11y smoke; upload UI artifacts.
3. **OpenAPI Governance** — lint 3.1 spec and diff against last tag; archive results.
4. **Dependency Review** — fail on **high** vulns.
5. **SBOM** — generate CycloneDX and upload.
6. **CodeQL** — language matrix: csharp and javascript.
7. **Gates Summary** — aggregate status for a single required check.

**Outcomes:** Artifacts (tests, OpenAPI lint/diff, SBOM) are stored and referenced by CD and the Evidence Pack.

**3) CD Overview (Promotion Flow & Approvals)**

**Flow:** **Alpha → Beta → RTM → Prod** (each as a **GitHub Environment** with approvals).

* **Alpha:** Deploy latest image; readiness smoke.
* **Beta:** Promote on approval; run E2E + perf smoke (**/mcp** p50≤300 ms / p95≤800 ms; **SSE TTFB ≤ 200 ms**).
* **RTM:** Deploy with **Prod DB (read‑only)** secret; run parity checks (/config/effective and contract tests).
* **Prod:** Canary (5–10%) → full rollout; start **24‑h post‑release checks**; attach monitoring snapshot to Release.

**Safety:** Ingress must **not buffer text/event-stream**; runbooks define **graceful drain** of SSE and child processes. Promotion blocks on **RTM parity** mismatches.

**4) Secrets & Environment Configuration**

**Where secrets live:** **GitHub Environments** only (never in code/DB/logs).  
**Per‑environment expectations:**

| **Env** | **Required Secrets** | **Notes** |
| --- | --- | --- |
| alpha | SQL\_CONNECTION\_STRING (Alpha DB), optional TELERIK\_LICENSE/\_PATH for UI build | UI license used **only** at build time |
| beta | SQL\_CONNECTION\_STRING (Beta DB) | — |
| rtm | SQL\_CONNECTION\_STRING\_PROD\_RO (**Prod DB read‑only**) | Enforces parity; **no writes** |
| prod | SQL\_CONNECTION\_STRING\_PROD | Canary + 24‑h checks |

**Policy reminders:** No secrets in DB (AppConfig/FeatureFlag are **non‑secret**); runtime dynamic values are DB‑sourced via SPs; license is not persisted in images.

**5) Evidence Pack (What to Collect)**

Attach to the GitHub Release (Prod) and retain **≥ 1 year**:

* Test reports (unit/integration/E2E), coverage summary.
* **CodeQL SARIF**, **Dependency Review** report, **Secret‑scan** summary.
* **SBOM** artifact and attestation (if used).
* **OpenAPI** file + **lint/diff** output.
* Readiness and config snapshots; **monitoring snapshot** (latency p50/p95, **SSE TTFB**, error rate, availability).
* Approvals for promotions (Alpha → Beta → RTM → Prod).

**6) Pipeline Definitions (Embedded YAML)**

These definitions are authoritative for CI/CD behavior. Copy to /.github/workflows/\* and adapt only via PR.

**6.1 CI workflow — .github/workflows/ci.yml (merge‑queue aware)**

name: CI

on:

push: { branches: [ "main" ] }

pull\_request: {}

merge\_group: {}

workflow\_dispatch: {}

permissions:

contents: read

packages: read

security-events: write

actions: read

concurrency:

group: ci-${{ github.ref }}

cancel-in-progress: true

jobs:

build\_test\_dotnet:

name: Build & Test (.NET 8)

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v4

- uses: actions/setup-dotnet@v4

with: { dotnet-version: '8.0.x' }

- run: dotnet restore

- run: dotnet build --no-restore -warnaserror

- run: dotnet test --no-build -p:CollectCoverage=true -p:CoverletOutputFormat=lcov

- uses: actions/upload-artifact@v4

if: always()

with:

name: dotnet-tests

path: |

\*\*/TestResults/\*\*/\*.trx

\*\*/coverage\*.info

build\_ui\_optional:

name: Build UI (KendoReact if present)

runs-on: ubuntu-latest

needs: build\_test\_dotnet

steps:

- uses: actions/checkout@v4

- name: Detect UI dir

id: detect

run: |

if [ -f "web/package.json" ]; then echo "dir=web" >> $GITHUB\_OUTPUT

elif [ -f "web/admin-portal/package.json" ]; then echo "dir=web/admin-portal" >> $GITHUB\_OUTPUT

elif [ -f "package.json" ]; then echo "dir=." >> $GITHUB\_OUTPUT

else echo "dir=" >> $GITHUB\_OUTPUT; fi

- name: Skip if no UI

if: ${{ steps.detect.outputs.dir == '' }}

run: echo "No UI portal detected."

- uses: actions/setup-node@v4

if: ${{ steps.detect.outputs.dir != '' }}

with: { node-version: '20.x' }

- name: Check for Kendo deps

id: kendo

if: ${{ steps.detect.outputs.dir != '' }}

run: |

if grep -E '"@progress/kendo-(react|theme)-' -q ${{ steps.detect.outputs.dir }}/package.json; then

echo "kendo=true" >> $GITHUB\_OUTPUT

else

echo "kendo=false" >> $GITHUB\_OUTPUT

fi

- name: Prepare Telerik license file

if: ${{ steps.kendo.outputs.kendo == 'true' }}

env:

TELERIK\_LICENSE: ${{ secrets.TELERIK\_LICENSE }}

TELERIK\_LICENSE\_PATH: ${{ secrets.TELERIK\_LICENSE\_PATH }}

run: |

set -euo pipefail

if [ -z "${TELERIK\_LICENSE:-}${TELERIK\_LICENSE\_PATH:-}" ]; then

echo "Missing Telerik license secret for Kendo build." >&2; exit 1; fi

if [ -n "${TELERIK\_LICENSE:-}" ]; then

echo "$TELERIK\_LICENSE" > "$RUNNER\_TEMP/telerik-license.txt"

echo "TELERIK\_LICENSE\_PATH=$RUNNER\_TEMP/telerik-license.txt" >> "$GITHUB\_ENV"

fi

- name: Install & Build UI

if: ${{ steps.detect.outputs.dir != '' }}

working-directory: ${{ steps.detect.outputs.dir }}

run: |

npm ci || npm i

npm run build

- name: A11y smoke (axe)

if: ${{ steps.detect.outputs.dir != '' }}

working-directory: ${{ steps.detect.outputs.dir }}

run: npm run test:a11y || echo "A11y smoke not configured"

- uses: actions/upload-artifact@v4

if: ${{ steps.detect.outputs.dir != '' }}

with:

name: ui-dist

path: ${{ steps.detect.outputs.dir }}/dist

openapi\_governance:

name: OpenAPI Lint & Diff

runs-on: ubuntu-latest

needs: build\_test\_dotnet

steps:

- uses: actions/checkout@v4

- name: Lint OpenAPI

run: npx --yes @redocly/cli@latest lint api/openapi/mcp-proxy.yaml

- name: Diff OpenAPI vs last tag

run: |

set -e

LAST\_TAG=$(git describe --tags --abbrev=0 2>/dev/null || echo "")

if [ -n "$LAST\_TAG" ]; then

npx --yes @redocly/cli@latest diff api/openapi/mcp-proxy.yaml $LAST\_TAG:api/openapi/mcp-proxy.yaml || true

else

echo "No previous tag to diff against."

fi

- name: Save OpenAPI outputs

if: always()

run: |

mkdir -p evidence/api

npx --yes @redocly/cli@latest lint api/openapi/mcp-proxy.yaml > evidence/api/openapi.lint.txt || true

LAST\_TAG=$(git describe --tags --abbrev=0 2>/dev/null || echo "")

if [ -n "$LAST\_TAG" ]; then

npx --yes @redocly/cli@latest diff api/openapi/mcp-proxy.yaml $LAST\_TAG:api/openapi/mcp-proxy.yaml > evidence/api/openapi.diff.txt || true

fi

cp api/openapi/mcp-proxy.yaml evidence/api/openapi.yaml

- uses: actions/upload-artifact@v4

if: always()

with:

name: openapi-governance

path: evidence/api/

dependency\_review:

name: Dependency Review

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v4

- uses: actions/dependency-review-action@v4

with: { fail-on-severity: high }

sbom:

name: Generate SBOM

runs-on: ubuntu-latest

needs: build\_test\_dotnet

steps:

- uses: actions/checkout@v4

- uses: anchore/sbom-action@v0

with:

format: cyclonedx-json

artifact-name: sbom.cdx.json

- uses: actions/upload-artifact@v4

with:

name: sbom

path: sbom.cdx.json

codeql:

name: CodeQL (C#/JS)

uses: ./.github/workflows/codeql.yml

permissions:

actions: read

contents: read

security-events: write

gates:

name: Required Gates Summary

runs-on: ubuntu-latest

needs: [ build\_test\_dotnet, build\_ui\_optional, openapi\_governance, dependency\_review, sbom, codeql ]

steps:

- run: echo "All CI gates completed — see job details for failures."

**6.2 Deploy workflow — .github/workflows/deploy.yml**

name: Deploy

on:

workflow\_dispatch:

inputs:

image: { description: "Container image (tag or digest) to deploy", required: true }

environment: { description: "alpha|beta|rtm|prod", required: true, default: "alpha" }

permissions: { contents: read, packages: read, deployments: write }

env: { APP\_NAME: mcp-proxy }

jobs:

alpha:

if: ${{ github.event.inputs.environment == 'alpha' }}

name: Deploy Alpha

runs-on: ubuntu-latest

environment: alpha

steps:

- uses: actions/checkout@v4

- run: ./scripts/deploy-alpha.sh "${{ github.event.inputs.image }}"

- run: curl -fsS https://alpha.example.com/api/ready | jq .

- uses: actions/upload-artifact@v4

if: always()

with: { name: evidence-alpha, path: evidence/ }

beta:

if: ${{ github.event.inputs.environment == 'beta' }}

name: Deploy Beta

runs-on: ubuntu-latest

environment: beta

steps:

- uses: actions/checkout@v4

- run: ./scripts/deploy-beta.sh "${{ github.event.inputs.image }}"

- run: ./scripts/perf-smoke.sh https://beta.example.com/api

- uses: actions/upload-artifact@v4

if: always()

with: { name: evidence-beta, path: evidence/ }

rtm:

if: ${{ github.event.inputs.environment == 'rtm' }}

name: Deploy RTM (Prod DB read-only)

runs-on: ubuntu-latest

environment: rtm

steps:

- uses: actions/checkout@v4

- run: ./scripts/deploy-rtm.sh "${{ github.event.inputs.image }}"

- run: ./scripts/rtm-parity.sh https://rtm.example.com/api

- uses: actions/upload-artifact@v4

if: always()

with: { name: evidence-rtm, path: evidence/ }

prod:

if: ${{ github.event.inputs.environment == 'prod' }}

name: Deploy Prod (canary -> rollout)

runs-on: ubuntu-latest

environment: prod

steps:

- uses: actions/checkout@v4

- run: ./scripts/deploy-prod.sh "${{ github.event.inputs.image }}" --strategy canary --percent 10

- run: ./scripts/observe.sh https://prod.example.com/api 30m

- run: ./scripts/deploy-prod.sh "${{ github.event.inputs.image }}" --strategy rollout

- run: echo "Queue 24‑hour post‑release checks and snapshot export"

- uses: actions/upload-artifact@v4

if: always()

with: { name: evidence-prod, path: evidence/ }

**6.3 CodeQL workflow — .github/workflows/codeql.yml**

name: CodeQL

on:

push: { branches: [ "main" ] }

pull\_request: {}

merge\_group: {}

schedule: [ { cron: "24 2 \* \* 1" } ]

workflow\_dispatch: {}

permissions: { actions: read, contents: read, security-events: write }

jobs:

analyze:

name: CodeQL Analyze

runs-on: ubuntu-latest

strategy:

fail-fast: false

matrix:

language: [ 'csharp', 'javascript' ]

steps:

- uses: actions/checkout@v4

- uses: github/codeql-action/init@v3

with: { languages: ${{ matrix.language }} }

- uses: actions/setup-dotnet@v4

if: ${{ matrix.language == 'csharp' }}

with: { dotnet-version: '8.0.x' }

- uses: github/codeql-action/autobuild@v3

- uses: github/codeql-action/analyze@v3

with: { category: "/language:${{ matrix.language }}" }

**7) Quality Gates & Release Criteria**

A release is **eligible** for promotion when:

* **CI**: All jobs green; **OpenAPI** lint/diff reviewed; **CodeQL**/Dep Review green; **SBOM** attached; Secret‑scan has no new leaks.
* **Beta**: Perf smoke within budgets (JSON p50≤300 ms/p95≤800 ms; **SSE TTFB≤200 ms**).
* **RTM**: **Prod DB (RO)** parity verified; /config/effective snapshot matches expectations.
* **Prod**: Canary stable; 24‑h checks finish with availability ≥99.9% and error rate <1%.

**8) Rollback/Incident Integration**

* **Rollback:** Image flip to last good; **graceful SSE drain**; ingress must not buffer; see runbooks/rollback.docx.
* **Incident:** Follow runbooks/incident.docx — declare severity, collect TTFB/latency/error evidence, decide rollback vs forward‑fix vs config change.
* **License rotation:** runbooks/rotate\_telerik\_license.docx (build‑time only, never stored).
* **Scale‑out:** runbooks/scale\_out.docx (HPA, PDB, sticky routing on Mcp‑Session‑Id).

**9) Compliance Mapping**

* **No‑Hard‑Coding / SP‑only / Add‑only** reiterated in PR template, DB contracts, and CI checks.
* **Secrets** only in GitHub Environments; never in DB (AppConfig/FeatureFlag are **non‑secret**).
* **Evidence Pack** meets retention **≥ 1 year** with artifacts from CI/CD and monitoring.

**10) Assumptions**

1. GitHub Environments exist (alpha, beta, rtm, prod) with required secrets and approval rules.
2. Ingress supports **SSE** pass‑through; timeouts align with DB config (Network:SseKeepAliveSeconds, Network:RequestTimeoutSeconds).
3. RTM always points to **Prod DB (read‑only)** and never performs writes.

**11) Next Steps**

* Roll these workflows into the repo; enable **Secret Scanning** and **branch protections + merge queue**.
* Hook **Evidence Pack** collection into release tagging; add an internal checklist issue template.
* Schedule a **game day** to rehearse SSE drain, rollback, and license rotation.

**Footer (optional for Word header/footer):**  
*MCPX‑KendoBridge • CI/CD Plan • v2.0.0 • 2025‑09‑27 • Confidential — Technijian Internal*