CI/CD Pipeline

Bilko — CI/CD Pipeline

Status: PLANNED (GitHub Actions workflows not yet configured)

This document describes the target continuous integration and deployment pipeline for Bilko.

Overview

Bilko uses **GitHub Actions** for CI/CD automation:

- **Trigger:** Every push to `main` and on pull requests
- **Stages:** Lint → Type Check → Unit Tests → Integration Tests → Build → E2E Tests → Deploy
- **Duration Target:** <10 minutes from commit to production

Why GitHub Actions?

- Free for public repos
- Native GitHub integration
- Easy to configure (YAML)
- Matrix builds for parallel testing
- Secret management built-in

Pipeline Overview

```
flowchart TD
    PUSH(["git push / PR opened"])
    TRIGGER{{"Branch?"}}

    subgraph PARALLEL["Stage 1 — Parallel Quality Checks"]
        LINT["Lint\nESLint + Prettier\n<2 min"]
        TC["Type Check\nTypeScript strict\n<2 min"]
        UT["Unit Tests\nVitest + coverage\n<3 min"]
        IT["Integration Tests\nSupertest + real PG\n<5 min"]
```

```
    end

    BUILD["Build (Turborepo)\napps/web → .next\napps/api → dist\n<4 min"]

    subgraph E2E_BLOCK["Stage 3 — E2E Tests"]
        VP["Wait for Vercel\nPreview URL"]
        E2E["Playwright E2E\nChromium + Firefox + WebKit\n<8 min"]
    end

    subgraph DEPLOY["Stage 4 — Deploy (main only)"]
        DF["Deploy Frontend\nVercel Production\nbilko.io"]
        DB["Deploy Backend\nRailway Production\napi.bilko.io"]
        MIGRATE["DB Migrations\nnpx prisma migrate deploy"]
    end

    NOTIFY["Slack Notification\n#bilko-deploys"]

    PUSH --> TRIGGER
    TRIGGER -->|"PR"| PARALLEL
    TRIGGER -->|"main"| PARALLEL
    PARALLEL --> BUILD
    BUILD --> E2E_BLOCK
    VP --> E2E
    E2E_BLOCK --> DEPLOY
    DEPLOY --> NOTIFY

    LINT & TC & UT & IT -->|"All pass"| BUILD
```

Pipeline Stages

1. Code Quality (Parallel)

ESLint + Prettier

```
- name: Lint
  run: npm run lint
```

Checks:

- ESLint rules for TypeScript
- Prettier formatting
- Import order
- Unused variables

Fail Conditions:

- Any ESLint errors
- Prettier format violations

TypeScript Type Check

```
- name: Type Check
  run: npm run type-check
```

Checks:

- TypeScript strict mode compliance
- No `any` types without justification
- Correct Prisma types
- React prop types

Fail Conditions:

- Any TypeScript errors
- Type inference failures

---

2. Unit Tests (Vitest)

```
- name: Unit Tests
  run: npm run test:unit
```

Coverage Requirements:

- Overall: >80%
- Financial logic (invoices, VAT, double-entry): >95%
- Utility functions: >90%

Test Types:

- Business logic (invoice calculations, VAT rates)
- Currency conversion
- Double-entry validation

- Date utilities
- Number formatting

Fail Conditions:

- Any test failures
- Coverage below threshold
- Test timeout (>30s per test)

---

3. Integration Tests (Supertest)

```
- name: Integration Tests
  run: npm run test:integration
  env:
    DATABASE_URL: ${{ secrets.TEST_DATABASE_URL }}
```

Setup:

- Provision test PostgreSQL database
- Run migrations: `npx prisma migrate deploy`
- Seed test data
- Run tests against real database
- Cleanup after tests

Test Types:

- API endpoint tests (all routes)
- Auth flow (register, login, refresh, logout)
- CRUD operations (invoices, expenses, contacts)
- Database transactions
- Error handling

Fail Conditions:

- Any test failures
- Database connection errors
- Memory leaks (heap growth >100MB)

Job Dependency Graph

```
graph LR
    LINT["lint"]
    TC["type-check"]
    UT["unit-tests"]
    IT["integration-tests"]
    BUILD["build\nneeds: lint, type-check,\nunit-tests, integration-tests"]
    E2E["e2e-tests\nneeds: build"]
    DF["deploy-frontend\nneeds: build, e2e-tests\nif: main branch"]
    DB_JOB["deploy-backend\nneeds: build, e2e-tests\nif: main branch"]

    LINT --> BUILD
    TC --> BUILD
    UT --> BUILD
    IT --> BUILD
    BUILD --> E2E
    E2E --> DF
    E2E --> DB_JOB
```

4. Build (Turborepo)

```
- name: Build
  run: npm run build
```

Build Targets:

- `apps/web` — Next.js production build
- `apps/api` — TypeScript compilation to `dist/`
- `packages/database` — Prisma Client generation

Fail Conditions:

- Build errors
- TypeScript compilation errors
- Missing environment variables (fail-fast)

Artifacts:

- `apps/web/.next/` — Next.js build output
- `apps/api/dist/` — Compiled JavaScript
- Build logs for debugging

---

5. E2E Tests (Playwright)

```
- name: E2E Tests
  run: npm run test:e2e
  env:
    PLAYWRIGHT_BASE_URL: ${{ env.PREVIEW_URL }}
```

Setup:

- Wait for Vercel preview deployment (for PRs)
- Install Playwright browsers
- Run tests against preview URL

Test Scenarios:

- **Invoice Flow:** Create → Preview → Send → Mark Paid
- **Expense Flow:** Add → Upload Receipt → Approve → Pay
- **Report Flow:** Generate P&L → Export PDF
- **Auth Flow:** Register → Login → 2FA → Logout

Browsers:

- Chromium (primary)
- Firefox (secondary)
- Safari/WebKit (mobile)

Fail Conditions:

- Any test failures
- Screenshot diffs (visual regression)
- Timeout (>60s per test)

---

6. Deploy

Frontend (Vercel)

```
- name: Deploy Frontend
  uses: vercel/action@v1
  with:
    vercel-token: ${{ secrets.VERCEL_TOKEN }}
    vercel-project-id: ${{ secrets.VERCEL_PROJECT_ID }}
    vercel-org-id: ${{ secrets.VERCEL_ORG_ID }}
```

```
    production: ${{ github.ref == 'refs/heads/main' }}
```

Deployment Strategy:

- **PR:** Deploy to preview URL (automatic)
- **main branch:** Deploy to production (automatic)

Rollback:

- Automatic if deployment fails health check
- Manual via Vercel Dashboard

Backend (Railway)

```
- name: Deploy Backend
  uses: railway-app/action@v1
  with:
    railway-token: ${{ secrets.RAILWAY_TOKEN }}
    service: api
    environment: ${{ github.ref == 'refs/heads/main' && 'production' || 'staging' }}
```

Pre-Deploy:

1. Run database migrations: `npx prisma migrate deploy`
2. Health check on current deployment

Deployment Strategy:

- **PR:** Deploy to staging Railway environment
- **main branch:** Deploy to production Railway environment

Rollback:

- Railway keeps last 10 deployments
- Rollback via Railway Dashboard or CLI

Workflow Files

Main Workflow (.github/workflows/main.yml)

```yaml
name: CI/CD Pipeline

on:
  push:
    branches: [main]
  pull_request:
    branches: [main]

jobs:
  lint:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: 18
          cache: npm
      - run: npm ci
      - run: npm run lint

  type-check:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: 18
          cache: npm
      - run: npm ci
      - run: npm run type-check

  unit-tests:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: 18
          cache: npm
      - run: npm ci
```

```
    - run: npm run test:unit -- --coverage
    - uses: codecov/codecov-action@v3
      with:
        files: ./coverage/coverage-final.json

  integration-tests:
    runs-on: ubuntu-latest
    services:
      postgres:
        image: postgres:15
        env:
          POSTGRES_USER: bilko_test
          POSTGRES_PASSWORD: bilko_test
          POSTGRES_DB: bilko_test
        options: >-
          --health-cmd pg_isready
          --health-interval 10s
          --health-timeout 5s
          --health-retries 5
        ports:
          - 5432:5432
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: 18
          cache: npm
      - run: npm ci
      - run: npx prisma migrate deploy
        env:
          DATABASE_URL: postgresql://bilko_test:bilko_test@localhost:5432/bilko_test
      - run: npm run test:integration
        env:
          DATABASE_URL: postgresql://bilko_test:bilko_test@localhost:5432/bilko_test

  build:
    needs: [lint, type-check, unit-tests, integration-tests]
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
```

```yaml
    - uses: actions/setup-node@v4
      with:
        node-version: 18
        cache: npm
    - run: npm ci
    - run: npm run build
    - uses: actions/upload-artifact@v3
      with:
        name: build-artifacts
        path: |
          apps/web/.next
          apps/api/dist

e2e-tests:
  needs: build
  runs-on: ubuntu-latest
  steps:
    - uses: actions/checkout@v4
    - uses: actions/setup-node@v4
      with:
        node-version: 18
        cache: npm
    - run: npm ci
    - run: npx playwright install --with-deps
    - name: Wait for Vercel Preview
      uses: patrickedqvist/wait-for-vercel-preview@v1.3.1
      id: vercel-preview
      with:
        token: ${{ secrets.GITHUB_TOKEN }}
        max_timeout: 300
    - run: npm run test:e2e
      env:
        PLAYWRIGHT_BASE_URL: ${{ steps.vercel-preview.outputs.url }}
    - uses: actions/upload-artifact@v3
      if: failure()
      with:
        name: playwright-screenshots
        path: test-results/

deploy-frontend:
```

```
    needs: [build, e2e-tests]
    if: github.ref == 'refs/heads/main'
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: vercel/action@v1
        with:
          vercel-token: ${{ secrets.VERCEL_TOKEN }}
          vercel-project-id: ${{ secrets.VERCEL_PROJECT_ID }}
          vercel-org-id: ${{ secrets.VERCEL_ORG_ID }}
          production: true

  deploy-backend:
    needs: [build, e2e-tests]
    if: github.ref == 'refs/heads/main'
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: railway-app/action@v1
        with:
          railway-token: ${{ secrets.RAILWAY_TOKEN }}
          service: api
          environment: production
```

Hotfix Workflow (.github/workflows/hotfix.yml)

Fast-track workflow for urgent production fixes (bypasses full pipeline):

```
name: Hotfix Deploy

on:
  workflow_dispatch:
    inputs:
      reason:
        description: 'Reason for hotfix'
        required: true

jobs:
  hotfix:
```

```
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: 18
          cache: npm
      - run: npm ci
      - run: npm run lint
      - run: npm run type-check
      - run: npm run build
      - uses: vercel/action@v1
        with:
          vercel-token: ${{ secrets.VERCEL_TOKEN }}
          vercel-project-id: ${{ secrets.VERCEL_PROJECT_ID }}
          vercel-org-id: ${{ secrets.VERCEL_ORG_ID }}
          production: true
      - uses: railway-app/action@v1
        with:
          railway-token: ${{ secrets.RAILWAY_TOKEN }}
          service: api
          environment: production
      - name: Notify Team
        uses: slackapi/slack-github-action@v1.24.0
        with:
          webhook-url: ${{ secrets.SLACK_WEBHOOK }}
          payload: |
            {
              "text": "□□Hotfix deployed: ${{ github.event.inputs.reason }}"
            }
```

Secrets Configuration

GitHub repository secrets (Settings → Secrets and variables → Actions):

Secret Name	Description	How to Generate
`VERCEL_TOKEN`	Vercel deployment token	Vercel Dashboard → Settings → Tokens

Secret Name	Description	How to Generate
`VERCEL_PROJECT_ID`	Vercel project ID	`vercel link` output
`VERCEL_ORG_ID`	Vercel organization ID	`vercel link` output
`RAILWAY_TOKEN`	Railway deployment token	Railway Dashboard → Settings → Tokens
`TEST_DATABASE_URL`	PostgreSQL test DB URL	Use GitHub Actions service
`SLACK_WEBHOOK`	Slack notification webhook	Slack → Apps → Incoming Webhooks

---

Branch Protection Rules

Configure on GitHub (Settings → Branches → Branch protection rules for `main`):

- □ Require status checks to pass before merging
  - `lint`
  - `type-check`
  - `unit-tests`
  - `integration-tests`
  - `build`
  - `e2e-tests`
- □ Require branches to be up to date before merging
- □ Require pull request reviews (1 approver minimum)
- □ Dismiss stale pull request approvals when new commits are pushed
- □ Require linear history (no merge commits, rebase/squash only)
- □ Do NOT allow force pushes (protect history)

Performance Targets

Pipeline Duration

- **Lint + Type Check:** <2 minutes
- **Unit Tests:** <3 minutes
- **Integration Tests:** <5 minutes
- **Build:** <4 minutes
- **E2E Tests:** <8 minutes
- **Deploy:** <3 minutes
- **TOTAL:** <10 minutes

Optimization Strategies

- Parallel jobs where possible
- Cache `node_modules` (GitHub Actions cache)
- Matrix builds for multi-browser E2E tests
- Incremental builds with Turborepo

Failure Handling

Failure & Rollback Flow

```
flowchart TD
    FAIL(["Pipeline Failure"])
    WHERE{{"Failed\nStage?"}}

    BLOCK_PR["PR Blocked\nLogs in GitHub Actions UI\nArtifacts uploaded"]
    FIX_CODE["Fix code\nPush new commit"]

    HEALTH{{"Health check\npasses?"}}
    AUTO_ROLL["Automatic Rollback\nPrevious deployment promoted"]
    SLACK_ALERT["Slack Alert\n#bilko-deploys"]
    MANUAL["Manual Investigation\nRailway / Vercel logs"]

    FLAKY{{"Flaky\nE2E test?"}}
    RETRY["Playwright retry\n(retries: 1)"]
    CRITICAL["Mark critical\nCreate GitHub issue"]

    FAIL --> WHERE
    WHERE -->|"Quality / Tests"| BLOCK_PR --> FIX_CODE
    WHERE -->|"Deploy"| HEALTH
    WHERE -->|"E2E"| FLAKY

    HEALTH -->|No| AUTO_ROLL --> SLACK_ALERT
    HEALTH -->|Yes| MANUAL

    FLAKY -->|Yes| RETRY
    RETRY -->|Still fails| CRITICAL
```

```
    FLAKY -->|No| BLOCK_PR
```

Test Failures

1. Pipeline stops immediately (fail-fast)
2. Logs available in GitHub Actions UI
3. Artifacts uploaded (screenshots, coverage reports)
4. PR blocked until fixed

Deployment Failures

1. Automatic rollback to previous version
2. Slack notification to team
3. Health check endpoint monitored
4. Manual intervention if health check fails

Flaky Tests

- Retry failed E2E tests once (Playwright config: `retries: 1`)
- If still fails, mark as critical and investigate
- Track flaky tests in issue tracker

Monitoring & Notifications

Slack Notifications

Notify on:

- Production deployment success/failure
- Critical test failures (E2E)
- Hotfix deployments
- Security vulnerabilities detected

Email Notifications

GitHub Actions built-in:

- Pipeline failures (to commit author)
- Deploy status (to repository admins)

Local Testing

Developers can run the full pipeline locally before pushing:

```
# Lint
npm run lint

# Type check
npm run type-check

# Unit tests with coverage
npm run test:unit -- --coverage

# Integration tests (requires local PostgreSQL)
npm run test:integration

# Build
npm run build

# E2E tests (requires build)
npm run test:e2e
```

Pre-commit Hook (Recommended): Install Husky to run lint + type-check before every commit:

```
npx husky install
npx husky add .husky/pre-commit "npm run lint && npm run type-check"
```

Future Enhancements

Security Scanning

- **Snyk:** Dependency vulnerability scanning
- **SonarQube:** Code quality and security analysis
- **OWASP Dependency-Check:** Known vulnerabilities

Performance Testing

- **Lighthouse CI:** Core Web Vitals on every PR
- **k6:** Load testing API endpoints (1K concurrent users)

Database Migration Testing

- Test migrations on copy of production database
- Validate data integrity post-migration
- Measure migration duration

Related Documents

- Deployment Guide: DEPLOYMENT.md
- Environment Setup: ENVIRONMENT.md
- Testing Guide: ../testing/TESTING-GUIDE.md

Last Updated: 2026-02-20 **Status:** PLANNED — No GitHub Actions workflows configured yet **Next Steps:** Create `.github/workflows/main.yml`, configure secrets, test on staging branch

---

Revision #4
Created 2026-02-23 10:48:10 UTC by John
Updated 2026-05-31 20:02:46 UTC by John