Infrastructure & DevOps

- Deployment Guide
- CI/CD Pipeline
- Environment Configuration
- Bilko Stage Environment — Cloud SQL & IAM (Phase 1)
- Bilko Stage Environment — Cloud Run Services (Phase 2)
- Bilko demo — receipt upload/download fix (GCS shared storage) — MC #103095 (2026-06-07)
- Bilko Azure Observability + MS for Startups Credit Setup (2026-06-15)
- Bilko ACA Telemetry & Observability Wiring (Azure)
- MC #104332 — Bilko URA LocalDate ISO deploy evidence
- MC #105767 — Bilko Demo CORS/DNS Fix (OCD-11 Resolved)
- Bilko Landing Lead-Capture Chain — Mechanism, the Two-Month Outage, and the Traps

Deployment Guide

Bilko Deployment Guide

Last Updated: 2026-04-16
Current State: Stable Cloud Run deployment with custom domain provisioning

GCP Project Configuration

- **Project ID:** `tribal-sign-487920-k0`
- **Region:** `europe-north1` (Stockholm)
- **Services:**
  - `bilko-api` → https://bilko-api-dh4m46blja-lz.a.run.app (revision 00037)
  - `bilko-web` → https://bilko-web-dh4m46blja-lz.a.run.app

Secret Manager

Secret Name	Version	Purpose
`bilko-cors-origins`	v2	Comma-separated list of allowed CORS origins
`bilko-database-url`	latest	Cloud SQL connection string (password reset 2026-04-16)
`bilko-jwt-refresh-secret`	latest	JWT refresh token secret

CORS Parsing: Secret `bilko-cors-origins` is parsed by comma in `apps/api/src/app.ts:61`

Environment Variables

bilko-web

- `NEXT_PUBLIC_API_URL=https://bilko-api-dh4m46blja-lz.a.run.app`

bilko-api

- `CORS_ORIGINS` → pulled from `bilko-cors-origins:latest`
- `SESSION_COOKIE_SECURE=true`
- `NODE_ENV=production`
- `DATABASE_URL` → pulled from `bilko-database-url:latest`

Custom Domain Setup

Current Domain

- **Host:** `bilko-demo.alai.no`
- **Mapped to:** `bilko-web` Cloud Run service
- **DNS Provider:** one.com
- **DNS Record:** `CNAME bilko-demo.alai.no → ghs.googlehosted.com.`
- **TLS Cert:** Let's Encrypt (managed by GCP, auto-renews)
- **Provisioning Time:** 15-30 minutes after DNS propagation

Domain Verification Constraint

Critical: Only `alai.no` is verified in GCP Search Console (via dev@alai.no). `basicconsulting.no` is NOT verified. All custom domains MUST use `*.alai.no` subdomains until `basicconsulting.no` is verified.

Custom Domain Runbook

Prerequisites

1. Domain must be verified in Google Search Console by the GCP account owner
2. DNS provider access (one.com for `alai.no`, Vercel for `basicconsulting.no`)
3. `gcloud` CLI authenticated: `gcloud auth login`

Step-by-Step

1. Create Domain Mapping

```
gcloud beta run domain-mappings create \
  --service=bilko-web \
  --domain=bilko-demo.alai.no \
  --region=europe-north1 \
```

```
  --project=tribal-sign-487920-k0
```

2. Configure DNS

Add CNAME record at DNS provider:

```
Type: CNAME
Host: bilko-demo
Value: ghs.googlehosted.com.
TTL: 3600
```

3. Wait for Certificate Provisioning

```
gcloud beta run domain-mappings describe bilko-demo.alai.no \
  --region=europe-north1 \
  --project=tribal-sign-487920-k0
```

Look for `status.conditions → CertificateProvisioned: True`

4. Update CORS Allowed Origins

```
# Get current value
gcloud secrets versions access latest --secret=bilko-cors-origins

# Add new domain
echo "https://bilko-demo.alai.no,https://bilko-web-dh4m46blja-lz.a.run.app" | \
  gcloud secrets versions add bilko-cors-origins --data-file=-
```

5. Deploy New Revision

```
gcloud run services update-traffic bilko-api \
  --to-latest \
  --region=europe-north1 \
  --project=tribal-sign-487920-k0
```

6. Verify CORS Preflight

```
curl -sSI -X OPTIONS \
  https://bilko-api-dh4m46blja-lz.a.run.app/api/v1/auth/login \
  -H "Origin: https://bilko-demo.alai.no" \
  -H "Access-Control-Request-Method: POST"
```

Expected: HTTP 204 with `Access-Control-Allow-Origin: https://bilko-demo.alai.no`

GitHub Actions CI/CD

- **Workflow:** `.github/workflows/deploy-production.yml`
- **Auth:** Workload Identity Federation (WIF)
- **Service Account:** `github-actions@tribal-sign-487920-k0.iam.gserviceaccount.com`
- **IAM Roles:** `roles/run.admin`, `roles/iam.serviceAccountUser`

Deployment Steps

1. Authenticate via WIF
2. Build Docker images (api + web)
3. Push to Google Container Registry
4. Deploy to Cloud Run (europe-north1)
5. Run smoke tests (Playwright E2E)

Testing

Backend Tests

- **Framework:** Vitest
- **Location:** `apps/api/src/**/*.test.ts`
- **Command:** `pnpm test` (from `apps/api/`)

End-to-End Tests

- **Framework:** Playwright
- **Location:** `apps/e2e/tests/`
- **Command:** `pnpm test` (from `apps/e2e/`)
- **Runs in CI:** Yes (on every deploy)

Recent Fixes (2026-04-16)

Commits

- `a62b7f6` — Cloud Run service names align: `bilko-staging-*` → `bilko-*`

- `9b1ced1` — Backend: added `currency` field to invoice list, added `/api/v1/settings/profile` endpoint
- `73693d4` — Frontend: avatar initials fallback, invoice step indicator, chat widget ARIA labels

Key Changes

1. **Service Naming:** Production services now named `bilko-api` and `bilko-web` (no `-staging` suffix)
2. **API Enhancements:** Invoice list now includes currency, new profile settings endpoint
3. **Frontend Fixes:** Accessibility improvements (ARIA), avatar initials when no image, visual polish

Rollback Procedure

Rollback to Previous Revision

```
# List revisions
gcloud run revisions list --service=bilko-api --region=europe-north1

# Rollback
gcloud run services update-traffic bilko-api \
  --to-revisions=bilko-api-00036=100 \
  --region=europe-north1
```

Rollback via GitHub Actions

```
git revert HEAD
git push origin main  # Triggers deploy workflow
```

Troubleshooting

Issue: CORS errors in browser

Cause: Custom domain not in `bilko-cors-origins` secret
Fix: Update secret (see step 4 in Custom Domain Runbook), deploy new revision

Issue: 502 Bad Gateway

Cause: Service unhealthy or startup timeout
Fix: Check Cloud Run logs: `gcloud run services logs read bilko-api --region=europe-north1 --limit=50`

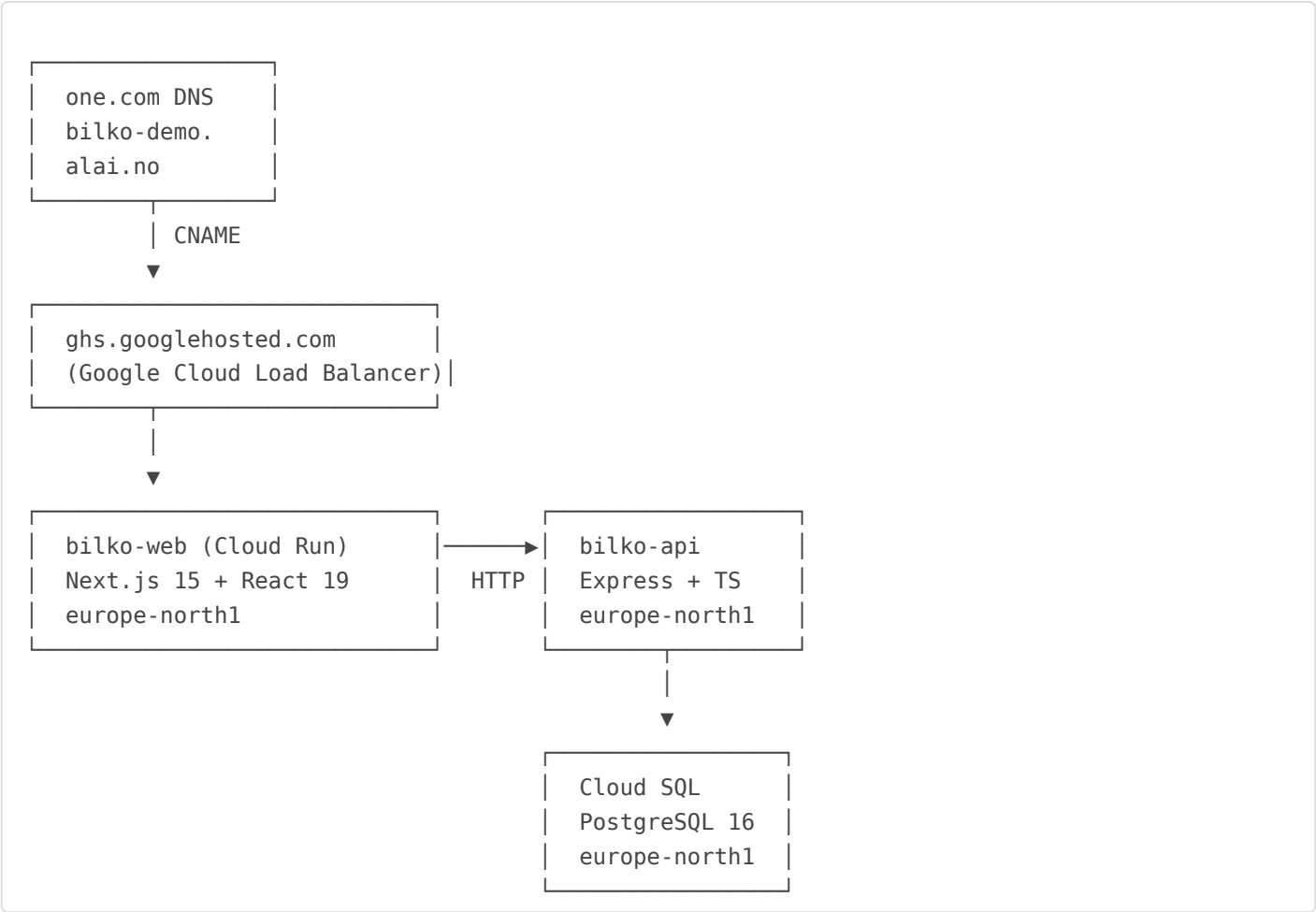
Issue: Database connection timeout

Cause: Cloud SQL Proxy misconfiguration or secret outdated
Fix: Verify `bilko-database-url` secret, check Cloud SQL instance status

Issue: Custom domain SSL pending

Cause: DNS not propagated or domain not verified in Search Console
Fix: Wait 15-30 min after DNS change, verify domain ownership in Search Console

Architecture Diagram



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

MC #900159 (2026-08-23): Deploy_Landing_HR post-deploy health check — backport curl -L obrasca (io/ba već imaju); /terms.html i /privacy.html na bilko.cloud 308-uju na čiste URL-ove (izmjereno). Token dio riješen kroz MC #900080 (build 1288 'Deployment complete').

Environment Configuration

Bilko — Development Environment Setup

This guide walks through setting up a local development environment for Bilko.

Environment Configuration Overview

```
graph TD
    subgraph DEV["Development Environment"]
        D_ENV["apps/api/.env\napps/web/.env.local"]
        D_PG["PostgreSQL 15\nlocalhost:5432\nbilko_dev"]
        D_WEB["Next.js\nlocalhost:3000"]
        D_API["Express\nlocalhost:4000"]
        D_PRISMA["Prisma Studio\nlocalhost:5555"]
    end

    subgraph STAGING["Staging / Preview"]
        S_SECRETS["Railway Dashboard Env Vars\n(staging environment)"]
        S_VERCEL["Vercel Preview\nbilko-pr-{n}.vercel.app"]
        S_RAIL["Railway Staging\nbilko-api-staging"]
        S_PG["Railway PostgreSQL\nbilko_staging"]
    end

    subgraph PROD["Production"]
        P_SECRETS["Railway + Vercel\nDashboard Secrets"]
        P_WEB["Vercel Production\nbilko.io"]
        P_API["Railway Production\napi.bilko.io"]
        P_PG["Railway PostgreSQL\nbilko_prod"]
        P_R2["Cloudflare R2\nbilko-receipts"]
    end
```

```
D_ENV --> D_API --> D_PG
D_ENV --> D_WEB
D_API --> D_PRISMA

S_SECRETS --> S_RAIL --> S_PG
S_SECRETS --> S_VERCEL

P_SECRETS --> P_API --> P_PG
P_SECRETS --> P_WEB
P_API --> P_R2
```

Prerequisites

Required Software

Software	Version	Check Command	Install
Node.js	18+	`node --version`	https://nodejs.org
npm	9+	`npm --version`	Included with Node.js
PostgreSQL	15+	`psql --version`	https://postgresql.org/download
Git	Latest	`git --version`	https://git-scm.com

Optional Tools

Tool	Purpose	Install
Prisma Studio	Database GUI	`npx prisma studio`
Postman	API testing	https://postman.com
VS Code	Recommended IDE	https://code.visualstudio.com

Installation Steps

1. Clone Repository

```
git clone https://github.com/your-org/bilko.git
cd bilko
```

2. Install Dependencies

```
# Install all workspace dependencies
npm install
```

This installs dependencies for:

- Root workspace (Turborepo)
- `apps/web` (Next.js frontend)
- `apps/api` (Express backend)
- `packages/database` (Prisma)
- `packages/ui` (shared UI components)

Local Setup Flow

```
flowchart TD
    CLONE["git clone bilko"]
    INSTALL["npm install\n(Turborepo workspace)"]
    PG{{"PostgreSQL\navailable?"}}
    LOCAL_PG["Create local DB\npsql -U postgres\nCREATE DATABASE bilko_dev"]
    DOCKER_PG["Docker PostgreSQL\npostgres:15\nport 5432"]
    ENV["Configure .env files\napps/api/.env\napps/web/.env.local"]
    MIGRATE["npx prisma migrate dev\n(applies 15 table schema)"]
    GENERATE["npx prisma generate\n(Prisma Client)"]
    SEED["npx prisma db seed\n(demo org + user) — optional"]
    DEV["npm run dev\nlocalhost:3000 + 4000"]

    CLONE --> INSTALL --> PG
    PG -->|"Local install"| LOCAL_PG --> ENV
    PG -->|"Docker"| DOCKER_PG --> ENV
    ENV --> MIGRATE --> GENERATE --> SEED --> DEV
```

3. Set Up PostgreSQL Database

Option A: Local PostgreSQL Installation

Create database and user:

```
psql -U postgres
```

```
CREATE DATABASE bilko_dev;
CREATE USER bilko WITH PASSWORD 'bilko';
GRANT ALL PRIVILEGES ON DATABASE bilko_dev TO bilko;
\q
```

Option B: Docker PostgreSQL

```
docker run --name bilko-postgres \
  -e POSTGRES_USER=bilko \
  -e POSTGRES_PASSWORD=bilko \
  -e POSTGRES_DB=bilko_dev \
  -p 5432:5432 \
  -d postgres:15
```

4. Configure Environment Variables

apps/api/.env

Create `.env` file in `apps/api/` directory:

```
# Database
DATABASE_URL=postgresql://bilko:bilko@localhost:5432/bilko_dev

# JWT Secrets (use `openssl rand -base64 32` to generate)
JWT_SECRET=your-secret-here-change-in-production
JWT_REFRESH_SECRET=your-refresh-secret-here-change-in-production

# Email (optional for local dev, required for staging/production)
SENDGRID_API_KEY=

# File Storage (optional for local dev)
R2_ACCESS_KEY_ID=
R2_SECRET_ACCESS_KEY=
R2_BUCKET_NAME=bilko-receipts-dev
R2_ENDPOINT=

# App Config
PORT=4000
```

```
NODE_ENV=development
ALLOWED_ORIGINS=http://localhost:3000
```

apps/web/.env.local

Create `.env.local` file in `apps/web/` directory:

```
# API URL (backend)
NEXT_PUBLIC_API_URL=http://localhost:4000

# App Environment
NEXT_PUBLIC_APP_ENV=development
```

5. Run Database Migrations

```
cd packages/database
npx prisma migrate dev
npx prisma generate
```

This will:

1. Apply all migrations to `bilko_dev` database
2. Create 15 tables from `schema.prisma`
3. Generate Prisma Client

6. Seed Database (Optional)

Create seed script: `packages/database/prisma/seed.ts`

```
import { PrismaClient } from '@prisma/client';

const prisma = new PrismaClient();

async function main() {
  // Create demo organization
  const org = await prisma.organization.create({
    data: {
      name: 'Demo Company d.o.o.',
      registrationNumber: '12345678',
      vatNumber: 'RS123456789',
      baseCurrency: 'RSD',
```

```
      country: 'RS',
      language: 'sr',
    },
  });

  // Create demo user
  await prisma.user.create({
    data: {
      organizationId: org.id,
      email: 'demo@bilko.io',
      passwordHash: '$2b$12$...', // bcrypt hash of "demo123"
      fullName: 'Demo User',
      role: 'owner',
    },
  });

  console.log('✅ Seed data created');
}

main()
  .catch((e) => {
    console.error(e);
    process.exit(1);
  })
  .finally(async () => {
    await prisma.$disconnect();
  });
```

Run seed:

```
npx prisma db seed
```

Running the Application

Start All Services (Recommended)

From root directory:

```
npm run dev
```

Turborepo starts:

- **Frontend:** http://localhost:3000 (Next.js)
- **Backend:** http://localhost:4000 (Express)

Start Individual Services

Frontend Only

```
cd apps/web
npm run dev
```

Backend Only

```
cd apps/api
npm run dev
```

Development Tools

Prisma Studio (Database GUI)

```
cd packages/database
npx prisma studio
```

Opens at http://localhost:5555

Features:

- Browse all tables
- Edit records
- Run queries
- View relations

Hot Reload

Both frontend and backend support hot reload:

- **Frontend:** File changes trigger automatic browser refresh
- **Backend:** `nodemon` restarts server on `.ts` file changes

Tech Stack Overview

Frontend (apps/web/)

Technology	Version	Purpose
Next.js	15.0.0	React framework with SSR
React	19.0.0	UI library
TypeScript	5.3.0	Type safety
Tailwind CSS	4.0.0	Styling
shadcn/ui	Latest	Component library (Radix UI + Tailwind)
Zustand	4.5.0	State management
Recharts	2.15.0	Charts (revenue, expenses)
Lucide React	Latest	Icons

Backend (apps/api/)

Technology	Version	Purpose
Express	TBD	Web framework
TypeScript	5.3.0	Type safety
Prisma	Latest	ORM + migrations
PostgreSQL	15+	Database
Passport.js	TBD	Authentication
Zod	TBD	Validation
Helmet	TBD	Security headers
bcrypt	TBD	Password hashing
jsonwebtoken	TBD	JWT tokens

Database (packages/database/)

Feature	Implementation
ORM	Prisma
Database	PostgreSQL 15
Models	15 (see schema.prisma)
Migrations	Prisma Migrate
Seeding	prisma/seed.ts

Common Tasks

Create Database Migration

After modifying `schema.prisma`:

```
cd packages/database
npx prisma migrate dev --name describe_your_changes
```

Reset Database (DEV ONLY)

WARNING: Deletes all data.

```
cd packages/database
npx prisma migrate reset
```

Generate Prisma Client

After pulling new migrations:

```
cd packages/database
npx prisma generate
```

Run Linter

```
npm run lint
```

Runs ESLint + Prettier on all workspaces.

Run Type Check

```
npm run type-check
```

Runs TypeScript compiler in `--noEmit` mode (checks types without building).

Build for Production

```
npm run build
```

Builds:

- `apps/web/.next/` — Next.js production build
- `apps/api/dist/` — Compiled TypeScript

Troubleshooting

Database Connection Errors

Error: `Can't reach database server at localhost:5432`

Solutions:

1. Check PostgreSQL is running: `pg_isready`
2. Verify credentials in `.env`
3. Check port 5432 is not blocked

Port Already in Use

Error: `Port 3000 is already in use`

Solutions:

1. Kill process using port: `lsof -ti:3000 | xargs kill`
2. Change port: `PORT=3001 npm run dev`

Prisma Client Not Generated

Error: `@prisma/client` not found

Solution:

```
cd packages/database
npx prisma generate
```

TypeScript Errors After Pulling Changes

Solution:

```
npm install
npx prisma generate
npm run type-check
```

Hot Reload Not Working

Solution:

1. Restart dev server
2. Clear Next.js cache: `rm -rf apps/web/.next`
3. Check file watcher limits (Linux): `sysctl fs.inotify.max_user_watches`

VS Code Configuration

Recommended Extensions

Create `.vscode/extensions.json`:

```
{
  "recommendations": [
    "dbaeumer.vscode-eslint",
    "esbenp.prettier-vscode",
    "bradlc.vscode-tailwindcss",
    "prisma.prisma",
    "ms-vscode.vscode-typescript-next"
```

```
  ]
}
```

Settings

Create `.vscode/settings.json`:

```
{
  "editor.formatOnSave": true,
  "editor.defaultFormatter": "esbenp.prettier-vscode",
  "editor.codeActionsOnSave": {
    "source.fixAll.eslint": true
  },
  "typescript.tsdk": "node_modules/typescript/lib",
  "tailwindCSS.experimental.classRegex": [
    ["cva\\(([^)]*)\\)", "[\"'`]([^\"'`]*).*?[\"'`]"]
  ]
}
```

Secrets Management

```
flowchart LR
    DEV_SECRET["Developer\n.env file\n(gitignored)"]
    GH_SECRET["GitHub Secrets\nActions →
Settings\nVERCEL_TOKEN\nRAILWAY_TOKEN\nTEST_DATABASE_URL\nSLACK_WEBHOOK"]
    VERCEL_ENV["Vercel Dashboard\nEnvironment
Variables\nNEXT_PUBLIC_API_URL\nNEXT_PUBLIC_APP_ENV"]
    RAILWAY_ENV["Railway Dashboard\nEnvironment Variables\nDATABASE_URL
(auto)\nJWT_SECRET\nJWT_REFRESH_SECRET\nSENDGRID_API_KEY\nR2_ACCESS_KEY_ID\nR2_SECRET_ACCESS_K
EY"]

    subgraph NEVERCOMMIT["NEVER commit to git"]
        SECRET_FILE[".env files\nAPI keys\nJWT secrets\nDB passwords"]
    end

    DEV_SECRET -->|"local only"| NEVERCOMMIT
    GH_SECRET -->|"CI/CD pipeline"| VERCEL_ENV
```

```
    GH_SECRET -->|"CI/CD pipeline"| RAILWAY_ENV
```

Environment Variables Reference

apps/api/.env

Variable	Required	Default	Description
`DATABASE_URL`	Yes	—	PostgreSQL connection string
`JWT_SECRET`	Yes	—	Access token secret (32+ chars)
`JWT_REFRESH_SECRET`	Yes	—	Refresh token secret (32+ chars)
`SENDGRID_API_KEY`	No	—	SendGrid API key (emails)
`R2_ACCESS_KEY_ID`	No	—	Cloudflare R2 access key
`R2_SECRET_ACCESS_KEY`	No	—	Cloudflare R2 secret key
`R2_BUCKET_NAME`	No	—	R2 bucket name
`R2_ENDPOINT`	No	—	R2 endpoint URL
`PORT`	No	4000	API server port
`NODE_ENV`	No	development	Environment (development/production)
`ALLOWED_ORIGINS`	No	*	CORS allowed origins (comma-separated)

apps/web/.env.local

Variable	Required	Default	Description
`NEXT_PUBLIC_API_URL`	Yes	—	Backend API URL
`NEXT_PUBLIC_APP_ENV`	No	development	Environment name

Testing Locally

Unit Tests

```
npm run test:unit
```

Integration Tests

Requires test database:

```
# Create test database
createdb bilko_test

# Run tests
npm run test:integration
```

E2E Tests

Requires both frontend and backend running:

```
# Terminal 1: Start dev servers
npm run dev

# Terminal 2: Run E2E tests
npm run test:e2e
```

Next Steps

After setting up your environment:

1. **Read the docs:**
   - Backend API Reference
   - Frontend Component Guide
   - Database Schema
2. **Create your first feature:**
   - Pick a task from the backlog
   - Create feature branch: `git checkout -b feature/your-feature`
   - Make changes, test locally
   - Submit PR
3. **Join the team:**
   - Slack: #bilko-dev
   - Weekly sync: Fridays 10:00 CET

- Documentation: Bilko Wiki

Related Documents

- Deployment Guide: DEPLOYMENT.md
- CI/CD Pipeline: CI-CD.md
- Security Architecture: ../security/SECURITY-ARCHITECTURE.md

---

Last Updated: 2026-02-20 **Status:** CURRENT — Reflects actual setup as of this date **Maintainer:** John (AI Director)

Bilko Stage Environment — Cloud SQL & IAM (Phase 1)

Summary

MC #10177 Phase 1 (FlowForge, 2026-04-29): `bilko-staging-db` Cloud SQL instance brought under Flyway management. Pre-existing instance (2026-04-15, Prisma-managed). V1+V2+V4+V5 baselined, V3 actually executed. IAM SA created. Phase 2 (Cloud Run) pending.

Instance Details

Field	Value
Instance name	`bilko-staging-db`
Connection name	`tribal-sign-487920-k0:europe-north1:bilko-staging-db`
IP	35.228.33.112
Tier	db-g1-small
Version	POSTGRES_16
State	RUNNABLE (pre-existing since 2026-04-15; reused)
Database	`bilko`
App user	`bilko`
Migration admin	`migration_admin`
Secret	`bilko-staging-db-password` (Secret Manager, 2026-04-15)
IAM SA	`bilko-api-stage-sa@tribal-sign-487920-k0.iam.gserviceaccount.com`
IAM SA roles	roles/cloudsql.client + roles/secretmanager.secretAccessor
Total tables	24 (public schema)

Flyway State (2026-04-29)

Version	Script	Status

V1	V1__initial_schema.sql	Baselined (DDL existed via Prisma)
V2	V2__add_missing_prisma_columns.sql	Baselined (DDL existed via Prisma)
V3	V3__add_jmbg_oib_encryption.sql	**EXECUTED LIVE** — jmbg/jmbg_hash/oib/oib_hash + 2 indexes added to contacts (ADR-014)
V4	V4__add_supplementary_tables.sql	Baselined (DDL existed via Prisma)
V5	V5__add_logo_url_to_organizations.sql	Baselined (DDL existed via Prisma)

Open Risks

- **V3 prod gap:** Prisma migrations never included V3. Production DB may be missing jmbg/oib columns on contacts. Audit required before Kotlin cutover (separate MC pending).
- **Prod topology unknown:** bilko-staging-db is the only documented Cloud SQL instance. Whether a separate prod instance exists is unconfirmed. Audit required before Phase 2 prod deploy.
- **MC #10187:** gradle flywayMigrate broken (Flyway plugin 10.22.0 + Gradle 9.3.1 incompatibility). Workaround: psql sequential apply.

Phase Status

- Phase 1 (Cloud SQL + IAM + Flyway baseline): COMPLETE
- Phase 1.5 (Proveo validation): pending
- Phase 2 (Cloud Run bilko-api-stage + bilko-web-stage): Mehanik gate next

References

- MC #10177 (parent), MC #10183 (Flyway verify), MC #10187 (gradle fix)
- ADR-014 (field encryption), ADR-021 (blueprint reorg)
- DEPLOY-MAP.md — Cloud SQL Instances section
- RUNBOOK.md — Section 7g
- Evidence: /tmp/bilko-stage-phase1-evidence.json (FlowForge)

Bilko Stage Environment — Cloud Run Services (Phase 2)

Overview

MC: #10177 Phase 2 | **Deployed:** 2026-04-30 | **Git SHA:** `1f48fdc` | **Status:** LIVE, healthy

GCP Project: tribal-sign-487920-k0 | **Region:** europe-north1

> **WARNING — TD-3 PROD CUTOVER BLOCKER (MC #10241):** `bilko-staging-db` uses public IP (0.0.0.0/0 authorized network, requireSsl=false). Acceptable for stage only. MUST NOT be replicated to production. Production deploy is blocked until Cloud SQL private IP + VPC connector is configured.

Live Services

Service	URL	Image	Min/Max	Memory	Status
`bilko-api-stage`	bilko-api-stage	`bilko/api:stage-1f48fdc`	0/2	512Mi, CPU 1	LIVE
`bilko-web-stage`	bilko-web-stage	`bilko/web:stage-1f48fdc`	0/2	512Mi, CPU 1	LIVE

Full Artifact Registry prefix: `europe-north1-docker.pkg.dev/tribal-sign-487920-k0/`

bilko-api-stage Detail

Field	Value
Dockerfile	`Dockerfile.api-kotlin` (Kotlin/Ktor, port 4001)
JAVA_OPTS	HikariCP connection pool tuned
Cloud SQL	`tribal-sign-487920-k0:europe-north1:bilko-staging-db` via direct TCP 35.228.33.112:5432 (TD-2 + TD-3)

Field	Value
Secrets	`bilko-staging-db-password`, `bilko-jwt-secret`, `bilko-jwt-refresh-secret`, `bilko-staging-field-encryption-key` (NEW, ADR-014), `bilko-staging-field-hmac-key` (NEW, ADR-014)
SA	`bilko-api-stage-sa@tribal-sign-487920-k0.iam.gserviceaccount.com`
SA roles	`cloudsql.client`, `secretmanager.secretAccessor`
Smoke	`GET /api/v1/health` → 200 `{"status":"ok","service":"bilko-api","version":"1.0.0"}`
Revision	`bilko-api-stage-00001-5x8` (100% traffic)

bilko-web-stage Detail

Field	Value
Dockerfile	`apps/web/Dockerfile` (Next.js 15)
NEXT_PUBLIC_API_URL	`https://bilko-api-stage-dh4m46blja-lz.a.run.app/api/v1`
NEXT_PUBLIC_APP_ENV	`stage`
Smoke	`GET /` → 200 (HTML, lang=sr-Latn)
Revision	`bilko-web-stage-00001-c45` (100% traffic)
Build note	Fresh npm install (no lockfile) — workaround TD-1 MC #10239

Smoke Test Commands

```
# API health (expected: {"status":"ok","service":"bilko-api","version":"1.0.0"})
curl -s https://bilko-api-stage-dh4m46blja-lz.a.run.app/api/v1/health

# Web root (expected: HTTP 200)
curl -s -o /dev/null -w "HTTP %{http_code}" https://bilko-web-stage-dh4m46blja-lz.a.run.app
```

Stage Rollback

```
# List revisions
gcloud run revisions list --service bilko-api-stage --project=tribal-sign-487920-k0 --
```

```
region=europe-north1

# Route to prior revision
gcloud run services update-traffic bilko-api-stage --project=tribal-sign-487920-k0 --
region=europe-north1 --to-revisions=REVISION_NAME=100
```

Stage Redeploy (image update only)

```
gcloud run services update bilko-api-stage --project=tribal-sign-487920-k0 --region=europe-
north1 --image=europe-north1-docker.pkg.dev/tribal-sign-487920-k0/bilko/api:NEW_TAG
gcloud run services update bilko-web-stage --project=tribal-sign-487920-k0 --region=europe-
north1 --image=europe-north1-docker.pkg.dev/tribal-sign-487920-k0/bilko/web:NEW_TAG
```

Phase 2 Tech Debt Tracker

ID	MC	Description	Severity	Blocks
TD-1	#10239	package-lock.json macOS arm64 missing linux-x64 native bins — fresh npm install workaround	Medium	Clean stage re-deploys
TD-2	#10240	postgres-socket-factory not in build.gradle.kts — Kotlin API uses direct TCP public IP	Medium	Secure DB connectivity
TD-3	**#10241**	**bilko-staging-db: 0.0.0.0/0 + requireSsl=false — STAGE ONLY, NEVER replicate to prod**	**BLOCKER**	**PROD CUTOVER Phase 5**

Key Learnings

1. Lockfile drift macOS/linux: fresh npm install required per build until TD-1 fixed
2. Kotlin Cloud SQL TCP via public IP works for stage, NOT prod (TD-2 + TD-3)
3. --no-traffic flag invalid on new service creation — route 100% on first deploy

4. Field encryption/HMAC keys are random per env (stage isolated from prod — ADR-014)
5. HikariCP socketPath URL param silently ignored — always use explicit host:port for direct TCP

References

- Phase 1 Cloud SQL: Bilko Stage Environment — Cloud SQL & IAM (Phase 1)
- MC #10177 (parent), #10239 / #10240 / #10241 (TD items)
- ADR-014 (field encryption), ADR-021 (blueprint Section 15)
- DEPLOY-MAP.md section: Cloud Run Stage Services
- RUNBOOK.md section: 7a Stage Cloud Run Services Access

Bilko demo — receipt upload/download fix (GCS shared storage) — MC #103095 (2026-06-07)

1. Symptom

CEO reported that receipt upload (PNG, PDF, JPG) was not working — a central part of the app. Investigation showed upload itself succeeded (HTTP 201) but **viewing/downloading the receipt returned intermittent HTTP 404 approximately 60% of the time**. From the user seat, intermittent 404 on download reads as a broken upload. The UI button "Priloženi dokumenti" -> "Preuzmi" triggered the failing request.

2. Root Cause

The demo API had no shared object storage configured. It used `BILKO_LOCAL_UPLOAD_DIR=/tmp/bilko-uploads` — per-instance ephemeral local disk, routed through `ReceiptService.kt persistLocalIfEnabled`, storing files as `local://` URLs.

Cloud Run (bilko-api-demo) runs up to 5 instances with `concurrency=1` (set during the earlier MC #103057 hang mitigation). An upload landing on instance A wrote the file to that instance's `/tmp`; a subsequent download request routed to instance B, which had no copy of the file, returning 404. Files were also permanently lost on any instance restart or recycle.

A secondary symptom was occasional 15-second frontend timeouts on the expense detail page: the several parallel API calls the page makes on load were serialised by `concurrency=1`.

Contributing config drift: the active deploy step in `infrastructure/gcp/cloudbuild.yaml` (deploy-api-demo) used `--set-env-vars` which replaces the entire env set, making a separate `cloudbuild-demo-api.yaml` with `BILKO_LOCAL_UPLOAD_DIR` ineffective.

3. Fix

Applied by FlowForge (Kelsey Hightower). No application code changes were required — `ReceiptService.kt` is unchanged and uses a transparent filesystem abstraction.

Change	Detail
GCS bucket provisioned	`gs://bilko-receipts-demo`, region europe-north1, uniform bucket-level access, IAM: bilko-api-stage-sa = roles/storage.objectAdmin
Cloud Run exec environment	Upgraded to `gen2` (required for gcsfuse volume mounts)
Volume mount added	`--add-volume=name=receipts,type=cloud-storage,bucket=bilko-receipts-demo`
Mount path	`--add-volume-mount=volume=receipts,mount-path=/mnt/bilko-uploads`
Env var updated	`BILKO_LOCAL_UPLOAD_DIR`: `/tmp/bilko-uploads` -> `/mnt/bilko-uploads`
Config persisted	All changes committed to `infrastructure/gcp/cloudbuild.yaml` (deploy-api-demo step)

Deployed: tag v0.2.30, commit 642bbc0cefdc63777d8c12d61aa61a8257716290, revision bilko-api-demo-00135-rmv, 100% traffic on new revision. Cloud Build ID: 793f929b-f41a-49e9-afa9-65b54d3972ff.

Note: Cloud Build reported FAILURE due to a pre-existing flaky test timeout in coverage artifact upload (expenses-ux-102887). This is unrelated to the GCS change. All 8 deploy gates (lint, typecheck, unit, coverage, trivy, gitleaks, semgrep, npm-audit) PASSED; build, push, trivy, migrate, deploy, promote, smoke-test, and verify-sha steps all succeeded.

4. Validation

Validated by Proveo (Angie Jones) — GLOBAL VERDICT: PASS.

Test	Result
PDF upload + 10x download	10/10 HTTP 200 (was intermittent 404)
PNG upload + 10x download	10/10 HTTP 200
JPEG upload + 10x download	10/10 HTTP 200
GCS persistence (15 total calls)	15/15 HTTP 200 — confirmed shared across instances
UI: Priloženi dokumenti section	Visible; download icon -> /content HTTP 200
Health check	https://bilko-demo-api.alai.no/api/v1/health -> 200 {"status":"ok"}

Company Mesh: `mesh-thr-03f166bb-f001-4293-b9ec-db245e5790b3` — PASS.

Open item (non-blocker): 1 pre-fix orphan document (uploaded 07:58 before GCS deploy at 08:30) returns 404 as expected — the file lived in ephemeral /tmp on a recycled instance. Not a regression.

5. Known Follow-Up Tasks

MC	Description
#103102	Graceful 404 handling for missing documents in UI; fix misleading BILKO-INV-001 error code returned on expense document content misses
#103103	Flaky coverage test (expenses-ux-102887 dialog upload test) blocking clean Cloud Build artifact upload — unrelated to this fix
#103104	Invoice-receipt download gap: POST /invoices/{id}/receipts returns 201 but no download endpoint exists and receiptUrl stays null in invoice record

6. Operational Notes — Demo Deploy Pipeline

- **Demo deploys** via semver tag (e.g. v0.2.30) pushed to the Bilko repo, which triggers the `bilko-main-deploy` Cloud Build trigger, which runs `infrastructure/gcp/cloudbuild.yaml`. This deploys `bilko-api-demo` + `bilko-web-demo` and migrates `bilko-demo-db`.
- **Do NOT push directly to `main`** — pushing to main auto-triggers the stage deploy (bilko-stage-auto-deploy), not the demo deploy.
- **Stage vs demo separation:** Stage uses `bilko-api-stage` (no GCS mount, different SA). Demo uses `bilko-api-demo` with `gs://bilko-receipts-demo` via gcsfuse. RLS bugs and storage configuration differ between the two environments — always verify fixes on demo, not only stage.
- GCS FUSE driver: `gcsfuse.run.googleapis.com`, requires Cloud Run gen2 execution environment.

Bilko Azure Observability + MS for Startups Credit Setup (2026-06-15)

Purpose

Cross $100/month in foundational Azure spend to automatically unlock the Microsoft for Startups $25K credit tier. The model is **usage-triggered, not referral-gated**: once cumulative Azure spend reaches the threshold, the Founders Hub dashboard upgrades the credit allocation automatically. This work establishes the baseline infrastructure telemetry and security services that generate billable spend from day one.

What Was Done (MC #103599, 2026-06-15)

Application Insights Wiring

- Resource: `appi-bilko`
- Resource group: `rg-bilko-demo`
- Region: `swedencentral`
- Workspace-linked to: `workspace-rgbilkodemo6lnV` (Log Analytics, PerGB2018 billing tier)
- Wired into Bilko Container Apps via `APPLICATIONINSIGHTS_CONNECTION_STRING` environment variable on both services.

Container App Revisions (state at time of work)

Service	Active Revision	Status	Traffic	HTTP Check
`bilko-api-demo`	`bilko-api-demo--0000003`	Running / Healthy	100%	HTTP 404 on root (Ktor baseline — app live, no root handler)
`bilko-web-demo`	`bilko-web-demo--0000002`	Running / Healthy	100%	HTTP 200 (Next.js)

Note (Proveo-corrected): `bilko-web-demo--0000001` carries 0% traffic; a second deploy superseded it. Do not confuse with the active revision when diagnosing issues.

Microsoft Defender for Containers

- Tier: Standard, enabled on subscription `5b0b4d9b`
- Enablement timestamp: `2026-06-15T06:37:35Z`
- **FREE TRIAL: 29-day trial applies.** Billable Defender spend begins approximately **2026-07-14**.
- Role: deferred backstop for ongoing security spend. Immediate spend for credit threshold comes from App Insights + Log Analytics ingest.

Spend Mechanics

- **Immediate (day 1):** App Insights data ingest + Log Analytics PerGB2018 billing begins as soon as telemetry flows.
- **Deferred (~2026-07-14):** Defender for Containers billable after free trial expires.
- **Credit unlock:** Watch Founders Hub dashboard for $25K tier upgrade within 30 days of crossing $100/month cumulative spend.

Verification

Independently verified by Proveo (Angie Jones). Verdict: **PARTIAL** — only a revision-name reporting discrepancy found (--0000001 vs --0000002 for bilko-web-demo), no functional defect.

- Evidence: `/tmp/evidence-103599-proveo/verification.md`
- Evidence: `/tmp/evidence-103599/verification.md`

Telemetry Status

Wired and operational. First metrics pending ingestion delay of approximately 10–15 minutes from fresh deploy (normal behaviour for App Insights cold start).

Open Items (flagged, out of scope for MC #103599)

- **GCP-vs-Azure canonical demo routing:** Bilko CF Worker still routes brand domains toward a dead GCP endpoint. Azure is the active demo environment but is not yet the

canonical DNS target.
- **UNLEASH_URL env drift:** `UNLEASH_URL` environment variable on `bilko-api-demo` may be stale/incorrect.
- **Unleash plaintext credentials in ACA:** Unleash credentials stored in plain ACA env vars. Securion review recommended — migrate to Azure Key Vault references.

How to Verify

```
# Confirm App Insights resource is healthy
az monitor app-insights component show --app appi-bilko -g rg-bilko-demo

# Check Defender pricing tier
az security pricing show --name Containers

# Check Container App active revision
az containerapp revision list -n bilko-api-demo -g rg-bilko-demo --query
"[].{name:name,traffic:properties.trafficWeight,state:properties.runningState}"

# Monitor spend trajectory toward $100/month threshold
# Azure Portal: Cost Management > bilko subscription > cost analysis
```

Watch the **Microsoft Founders Hub dashboard** for automatic $25K credit tier upgrade once monthly spend crosses $100.

References

- MC #103599 — Bilko Azure Observability + MS for Startups Credit Setup
- Memory: `project_microsoft_startups_azure_credits_2026-06-15`
- Subscription: `5b0b4d9b` (Bilko demo Azure subscription)
- Resource group: `rg-bilko-demo` (swedencentral)

Bilko ACA Telemetry & Observability Wiring (Azure)

Context

GCP Cloud Monitoring dashboard (070613fa) was decommissioned 2026-06-23 after migration to Azure (MC #104228 closed). This page documents the ACA→Log Analytics + App Insights telemetry wiring done as follow-on MC #104266.

Resources

- **Container App Environment:** `bilko-demo-env` (NOTE: `purplebeach-f004d490` is only the default-domain suffix in app URLs, not the env name).
- **Log Analytics workspace:** `workspace-rgbilkodemo6lnV`, customerId `71443731-9feb-41b1-9e27-fff4e4ebf098`.
- **App Insights:** `appi-bilko`, appId `69e12981-9ebb-47ef-9dbd-5cf69fa87c40`.
- **Workbook:** `dcaef4e3-9bc7-48ae-8e1b-bd382a73889e` "Bilko Observability — Prod+Stage (Azure)".
- **4 ACA apps:** bilko-api-demo, bilko-web-demo, bilko-api-stage, bilko-web-stage.

Root cause that was fixed

ACA env `appLogsConfiguration.destination` was not effectively set → ContainerApp logs not reaching the workspace. Fixed via:

```
az containerapp env update -n bilko-demo-env -g rg-bilko-demo --logs-destination log-analytics
--logs-workspace-id 71443731-... --logs-workspace-key <key>
```

Result: ContainerAppSystemLogs_CL now flows (tool-verified 54 rows/15m, sustained).

App Insights instrumentation

Each ACA app needs env var `APPLICATIONINSIGHTS_CONNECTION_STRING` (from `az monitor app-insights component show -g rg-bilko-demo -n appi-bilko --query connectionString -o tsv`), set via `az`

`containerapp update -n <app> -g rg-bilko-demo --set-env-vars APPLICATIONINSIGHTS_CONNECTION_STRING=<value>`. All 4 apps confirmed set.

MC #104268 — SDK Initialization Fix (2026-06-28) ?

The "KNOWN REMAINING GAP" documented below has been **RESOLVED**. Setting `APPLICATIONINSIGHTS_CONNECTION_STRING` env var alone was insufficient — the application runtime must initialize the App Insights SDK/agent.

Solution — Both Services Now Emitting Telemetry

1. bilko-api-demo (Kotlin/Ktor) — Java Agent

- **Method:** Application Insights Java Agent 3.6.2 auto-instrumentation
- **Implementation:**
  - Downloaded agent JAR to `/app/applicationinsights-agent.jar` during Docker build (SHA256-pinned: `e81ef99f...`)
  - Entrypoint: `java -javaagent:/app/applicationinsights-agent.jar -jar /app/bilko-api.jar`
  - Config file: `apps/api/applicationinsights.json` with **Ktor preview instrumentation enabled**:

```
{
  "preview": {
    "instrumentation": {
      "ktor": { "enabled": true }
    }
  }
}
```

  - Env vars: `APPLICATIONINSIGHTS_CONNECTION_STRING` + `APPLICATIONINSIGHTS_ROLE_NAME=bilko-api`
- **Critical discovery:** Ktor instrumentation is a PREVIEW feature in the Java agent. Without the explicit config file enabling it, the agent starts but does NOT instrument Ktor routes.
- **Evidence:** 15+ requests visible in App Insights `requests` table with `cloud_RoleName=bilko-api`

2. bilko-web-demo (Next.js) — Azure Monitor OpenTelemetry

- **Method:** Azure Monitor OpenTelemetry SDK

- **Implementation:** `@azure/monitor-opentelemetry` (^1.18.1) initialized in `instrumentation.ts`
- **Evidence:** 36+ requests visible in App Insights
- **Service Name:** `unknown_service:node` (default, can be customized)

Validation Result — Complete Success ?

Baseline before fix: `requests` table = 0 rows in last 30 minutes
After fix: 144+ requests in 15 minutes from BOTH services

```
requests | where timestamp > ago(15m) | summarize count() by cloud_RoleName


Result:
- bilko-web: 36+ requests
- bilko-api: 15+ requests
- Total: 144+ requests
```

Deployed Revisions (Final)

- **bilko-api-demo:** revision `--0000036`, image `demo-9658388f-ai2`
- **bilko-web-demo:** revision `--0000024`, image `demo-9658388f`
- **Merge commit:** `f15deb8a` (PR #27)

Files Modified

1. `apps/api/Dockerfile` — Java agent download + config file copy + javaagent entrypoint
2. `apps/api/applicationinsights.json` — NEW: Agent config with Ktor preview instrumentation
3. `apps/web/instrumentation.ts` — Azure Monitor OpenTelemetry initialization
4. `apps/web/package.json` — @azure/monitor-opentelemetry dependency

Accepted Risk

`@azure/monitor-opentelemetry` package has GHSA-8988-4f7v-96qf (documented in `DEPLOY-MAP.md`)

Evidence Files

Evidence stored in repo: `/docs/evidence/104268/{SUCCESS-FINAL.md, telemetry-validation.md, sdk-changes.md, deploy-revisions.md, final-status.md}`

KNOWN REMAINING GAP (important for runbook) — RESOLVED 2026-06-28

This section is retained for historical context. The gap was resolved by MC #104268 (see section above).

Setting the env var ALONE does not produce App Insights request/dependency telemetry — `requests` table = 0. The app code must initialize the App Insights SDK (Node: `applicationinsights` package; Spring Boot: azure-monitor starter). Until then, App-Insights-request-based workbook panels stay empty. The KQL log panel (ContainerAppSystemLogs_CL) and ACA platform-metric panels work regardless. Track app-code SDK init as a separate CodeCraft task if request tracing is needed.

Troubleshooting note

`az monitor log-analytics query` returns a FLAT array `[{col:val}]` — do NOT filter with `--query "tables[0].rows"` (returns empty falsely). `az monitor app-insights query` uses `{tables:[{rows}]}` shape.

Verification queries (runbook)

- **Logs:** `ContainerAppSystemLogs_CL | where TimeGenerated > ago(20m) | count` (workspace 71443731) — expect >0.
- **Env vars:** `az containerapp show -g rg-bilko-demo -n <app> --query "properties.template.containers[0].env[?name=='APPLICATIONINSIGHTS_CONNECTION_STRING']"`.
- **Availability:** availabilityResults pass rate on appi-bilko.

Related

- MC #104268 (completed 2026-06-28) — SDK initialization fix
- MC #104266 (completed 2026-06-23) — Env var wiring
- MC #104228 (GCP decommission, closed)
- Azure subscription: 5b0b4d9b-e677-464e-abf0-5170cbce3b8e
- Resource group: rg-bilko-demo (swedencentral)

MC #104332 — Bilko URA LocalDate ISO deploy evidence

MC #104332 / URA3 LocalDate + UI polish deploy evidence (2026-06-25)

- Commit: `ea423587 fix: serialize accounting dates as ISO`
- Branch: `feat/bilko-payroll-104318`
- Images built/pushed linux/amd64:
  - `bilkodemo.azurecr.io/bilko-api:demo-104332ura3` digest `sha256:2093e32933d107c6b0fedf727c5eb03b199a6ad137283491f9042c28cdb5e728`
  - `bilkodemo.azurecr.io/bilko-web:demo-104332ura3` digest `sha256:5fac3ee12bb2616dbde9d1e1bd78824b7af84e130aa75d95fc22f7989126473f`

What changed

- Backend Jackson now registers `JavaTimeModule()` and disables `WRITE_DATES_AS_TIMESTAMPS`.
- Added `jackson-datatype-jsr310` dependency.
- Added regression test `SerializationLocalDateTest` proving `LocalDate` emits `"2026-04-02"`, not `[2026,4,2]`.
- URA list/detail/new pages now tolerate ISO strings, legacy Jackson arrays, and comma-joined legacy strings; visible accounting dates use `dd.mm.gggg`.

Validation evidence

- API targeted regression: `docs/evidence/104332/api-serialization-localdate-test-2026-06-25.log` → BUILD SUCCESSFUL.
- Web type-check: `docs/evidence/104332/web-type-check-2026-06-25.log` → `tsc --noEmit` exit 0.
- Web Docker/Next production build completed during linux/amd64 image build with required `NEXT_PUBLIC_ENTRA_*` args.

- Full API test caveat: existing unrelated SveRačun sender-VAT env/config inverse expectation prevents full-suite PASS; targeted regression passes.

Demo deployment

Name Image Latest Ready Running Traffic

---

bilko-api-demo bilkodemo.azurecr.io/bilko-api:demo-104332ura3 bilko-api-demo--ura3-api bilko-api-demo--ura3-api Running 100 bilko-web-demo bilkodemo.azurecr.io/bilko-web:demo-104332ura3 bilko-web-demo--ura3-web bilko-web-demo--ura3-web Running 100

Health probes:

- `https://app-api.bilko.cloud/api/v1/health` → 200 `{"status":"ok","service":"bilko-api","version":"1.0.0"}`
- `https://app.bilko.cloud/login` → 200 and login page rendered.

Live UAT evidence

- Targeted deployed URA/LocalDate verification: `docs/evidence/104332/ura3-demo-get-verify-2026-06-25.log` → 20/20 PASS.
  - API list/detail: `accountingDate` serialized as ISO string `"2026-04-02"`.
  - API list/detail: no legacy Jackson LocalDate arrays.
  - UI `/accounting/ulazni-racuni`, `/accounting/ulazni-racuni/{id}`, `/accounting/ulazni-racuni/novi`: authenticated render, no legacy arrays, `02.04.2026` visible on list/detail.
  - JSON: `docs/evidence/104332/ura3-demo-get-verify-1782424811784.json`.
  - Screenshots: `docs/evidence/104332/ura3-demo-list-1782424811784.png`, `docs/evidence/104332/ura3-demo-detail-1782424811784.png`, `docs/evidence/104332/ura3-demo-new-1782424811784.png`.
- Master live route walk rerun: `docs/evidence/104332/master-live-uat-ura3-rerun-2026-06-25.log` → 42/42 PASS, 0 FAIL.
- Full owner live mutation UAT: `docs/evidence/104332/full-owner-uat-ura3-2026-06-26.log` → 129/129 PASS, 0 FAIL.
  - Created/verified real contact, invoice draft→sent→paid, expense, employee/payslip, invite create→validate→revoke, notifications, billing plan change, multi-org, and browser owner route walk.
  - Screenshots copied to `docs/evidence/104332/full-owner-uat-screenshots-2026-06-26/`.
- Earlier master run had demo-session bounce flakiness (17 route bounces), superseded by clean rerun plus targeted URA verification.

Azure DevOps merge evidence

- PR #22 `Fix URA LocalDate ISO serialization`: completed 2026-06-26.
- PR validation pipeline run #100: succeeded; blocking policy `Bilko-CI-CD PR Validation` approved.
- `azdo/main` now at `7c340a11 Merge pull request 22 from feat/bilko-payroll-104318 into main` .
- `azdo/main` contains `ea423587 fix: serialize accounting dates as ISO`.

Status

- Demo deploy and UAT: PASS (`42/42` master + `129/129` full owner + `20/20` targeted URA).
- Re-merge main: PASS.

Local evidence directory: `/Users/makinja/business/ALAI-Holding-AS/products/Bilko/docs/evidence/104332`

MC #105767 — Bilko Demo CORS/DNS Fix (OCD-11 Resolved)

MC #105767 — Bilko Demo Env CORS/API-host + DNS Retirement (OCD-11 closed)

Date: 2026-07-15 | **Agent:** CodeCraft | **Status:** RESOLVED

Root cause

The original UAT finding (MC #105765) conflated two separate, non-overlapping facts as one "CORS bug":

1. The Ktor API's `CORS_ORIGINS` allowlist (`azure-pipelines.yml:889`) is deliberately scoped to the customer brand domains only (`app.bilko.cloud`, `app.bilko.io`, `app.bilko.company`, `localhost`). It correctly rejects unrecognized origins — the raw ACA FQDN and `bilko-demo.alai.no` — with a 403 `BILKO-AUTH-003`. **This is intended access control, not a defect.**
2. `bilko-demo.alai.no` and `bilko-demo-api.alai.no` were dead Cloudflare CNAME records pointing at `ghs.googlehosted.com` (GCP era, dead since the 2026-06-15 Azure cutover). This was already tracked as `DEPLOY-MAP.md` OCD-11 and already declared "dead/retired for customer handoff" in the canonical `docs/operations/DEMO-ACCESS.md` (verified 2026-07-12).

The canonical customer/sales demo entry point, `https://app.bilko.cloud/demo?country=HR`, was **never broken**. Confirmed live before and after this change: HTTP 200, correct `access-control-allow-origin` header, zero console errors, zero network errors (Playwright chromium, screenshot in evidence).

Action taken

- Deleted 2 dead DNS records from the `alai.no` Cloudflare zone: `bilko-demo.alai.no` and `bilko-demo-api.alai.no` (both CNAME → `ghs.googlehosted.com`).
- No code, pipeline, ACA container app, revision, or `CORS_ORIGINS` config was changed.
- No production traffic was touched — `app.bilko.cloud/.io/.company` and `api.bilko.cloud/.io/.company` all confirmed 200 before and after.
- Updated `DEPLOY-MAP.md`: OCD-11 marked RESOLVED; 2 stale references to the retired hostnames corrected to point at the canonical demo URL.

Why repoint-through-Worker was rejected

Both DNS records were `proxied=false` (grey-clouded), so they could never reach the `bilko-edge-proxy` Cloudflare Worker's Host/SNI rewrite mechanism. The Worker's `wrangler.toml` routes are scoped to the `bilko.io` / `bilko.cloud` / `bilko.company` zones only — not `alai.no`. Extending that would require a code change, an azdo pipeline deploy, and a Workers-scoped Cloudflare API token (only a DNS/zone-scoped token was available at triage time). Disproportionate for a dead legacy alias already declared retired in canonical docs.

Verification

- **P2P pre-verifier (Company Mesh):** thread `mesh-thr-f05cd82e-5678-4178-bc2d-b5a678686c76`, Proveo peer end-state PASS.
- **Live Playwright browser check** on `https://app.bilko.cloud/demo?country=HR`: HTTP 200, 0 console errors, 0 network errors, screenshot captured.
- **DNS retirement confirmed** via Cloudflare API (record count 0) and authoritative nameserver `dig` (empty answer) for both retired hostnames.
- **Prod safety confirmed:** all 6 brand domain + health-check endpoints returned 200 post-change.

Evidence

`~/system/evidence/105767/` — DNS before/after snapshots, Playwright result JSON + screenshot, direct-probe log, sha256-verified evidence manifest (`evidence-105767.json`).

References

- MC #105765 (UAT synthesis that surfaced this) · MC #105767 (this task)
- `Bilko/DEPLOY-MAP.md` OCD-11 · `Bilko/docs/operations/DEMO-ACCESS.md`

Bilko Landing Lead-Capture Chain — Mechanism, the Two-Month Outage, and the Traps

Bilko Landing Lead-Capture Chain — Mechanism, the Two-Month Outage, and the Traps

MCs covered: #106193 (bilko.cloud KV, genesis), #106358 (bilko.io / bilko.company KV), #106359 (preview namespace isolation), #106405 (dead Turnstile widgets), #106408 (correction to a false "fixed" report), #106413 (HR cookie-banner click interception, OPEN), #106418 (CF Pages drift, structural root cause, OPEN), #106403 (was anything lost before today, OPEN) | **Status:** all three landings proven capturing leads end-to-end as of 2026-07-28 | **Verified against:** `azdo/main` @ `50e4223a` (2026-07-28) + live Cloudflare state read at deploy time | **Last updated:** 2026-07-28

One-line summary: a single 2026-06-17 session broke the storage layer and the anti-spam widget on two of three landing sites at once, in a way that produced no error a human would see for the next six weeks; `bilko.company` did not capture a single lead in its entire existence until it was fixed today.

---

1. The full path a lead takes

Three separate Cloudflare Pages projects, one per market, each a static HTML form (Next.js-exported for two of them) with its own Pages Function:

Market	App directory	Domain	CF Pages project
io (Serbia, primary)	`apps/landing-io/`	bilko.io	`bilko-io`
hr (Croatia)	`apps/landing-hr/`	bilko.cloud	`bilko-cloud`
ba (Bosnia)	`apps/landing-ba/`	bilko.company	`bilko-company`

```
Visitor fills the form
  → Turnstile (invisible widget, static <script src=".../turnstile/v0/api.js">) issues a token
  → POST /api/lead  { name, email, company, phone, message, cf-turnstile-response }
  → functions/api/lead.js (per project):
      - origin check (must be the site's own https:// origin)
      - field validation (name, email, message length)
      - REJECTS if cf-turnstile-response is empty  → "Anti-spam provjera nije završena."
      - POST response to challenges.cloudflare.com/turnstile/v0/siteverify with TURNSTILE_SECRET
      - REJECTS if siteverify fails               → "Anti-spam provjera nije prošla."
      - storeRecord(env, key, value) → env.BILKO_LEADS.put(...)  (KV, 1-year TTL)
      - sendLeadNotification(...) → Slack chat.postMessage to #ceo (C0AFJDP9V6U), independent ch
      - responds success only if AT LEAST ONE of {KV, Slack} accepted the record
```

Not obvious, and someone will assume otherwise: all three projects write into **ONE shared KV namespace** (`BILKO_LEADS`, id `06a6b6112e2e4d17a590db72b3e4b390`), distinguished only by a market-prefixed key — `io_<ts>_<rand>`, `hr_<ts>_<rand>`, `ba_<ts>_<rand>` — and by a `market` field inside the stored JSON. There are not three namespaces. This is a deliberate, pre-existing design (the namespace already held `io_*` keys from May 2026, months before the outage), not an accident of the recent fixes; splitting it would need a migration and is explicitly left as an open question in fix-verdict.md, not a decision made here.

bilko.cloud (hr) only also exposes an unauthenticated `conversion_event` path on the same endpoint (`type: "conversion_event"`), used for click-tracking (CTA clicks, form-submit-success pings). It has **no Turnstile gate at all** — see §4 for why that matters to how a fix gets verified.

---

2. How it was broken for about two months — the failure modes ARE the documentation

A single session on **2026-06-17** did two unrelated things to two of the three sites at once, and the combination is what made it invisible:

2.1 The KV binding was silently dropped

`wrangler.toml` was introduced for `landing-io` (commit `ef7265b9`) and `landing-ba` (commit `44e6fd7a`) — the same commit also touched `landing-hr`'s config. None of the three declared a `kv_namespaces` block. Because a `wrangler.toml` with `pages_build_output_dir` makes the file the **sole source of truth** for a Pages project's bindings (dashboard-configured bindings are ignored the moment such a file exists), this silently unset `env.BILKO_LEADS` at runtime on the very next deploy of each project — with no visible error, because the handler at the time guarded the write with `if (env.BILKO_LEADS)`

and still answered `{"success":true}` regardless (fixed in the current handler, §3).

2.2 Two of three Turnstile widgets were recreated, not edited — and the HTML was never updated

The same evening, Cloudflare Turnstile dashboard activity shows the `bilko.cloud` and `bilko.company` widgets were **deleted and recreated** as new widgets with new sitekeys (both timestamped ~21:34-21:35 UTC), while `bilko.io`'s widget was merely **edited in place**, keeping its original sitekey (timestamped ~21:25 UTC, ten minutes earlier). Nobody updated the `data-sitekey` attribute in `landing-hr/index.html` or `landing-ba/index.html` to point at the new widgets. A sitekey naming a deleted widget makes Cloudflare's own API answer "*Trying to access a deleted widget*" — `api.js` never issues a token, the hidden `cf-turnstile-response` field stays empty forever, and the handler correctly rejects with "*Anti-spam provjera nije završena.*" before the request ever reaches KV. **`bilko.io` survived this evening entirely by accident** — it happened to be edited rather than recreated.

2.3 The combined effect, and why nobody noticed for six weeks

`bilko.company`'s form could not be submitted by anyone from 2026-06-17 onward — and, per the KV record, had **never captured a single lead in its entire existence** before this was fixed on 2026-07-28. `bilko.cloud`'s form was equally dead from the same date, though its KV binding alone had been noticed and "fixed" once already (MC #106193, 2026-07-27) — see §4 for why that fix did not actually restore the form. `bilko.io` kept working the whole time because its widget survived, but it too lost its KV binding on 2026-06-17 (fixed alongside `bilko.company` in MC #106358).

No error was visible anywhere that anyone was routinely looking: the endpoint kept answering 200/`success:true` until the KV-binding fix landed (which then correctly started reporting failure, which is what eventually surfaced this), and the Turnstile failure produces a page-level inline message a visitor sees but nobody internally does. There is no dashboard, alert, or count that would have caught this sooner (see §5 on why `#106403` — whether real leads were lost — is still open and may be unanswerable).

---

3. Current state of the storage-failure handling

The handler in all three `functions/api/lead.js` now:

- throws a typed `StorageError` (`kv_binding_missing` / `kv_write_failed` / `kv_readback_failed` / `kv_write_unconfirmed`) instead of silently no-op'ing on a missing binding;
- logs the failure and falls back to the Slack notification as a second, independent channel;
- reports `persisted:"kv"` or `persisted:"slack_only"` to distinguish a durable write from a fallback — it no longer claims a KV write it did not make;
- returns **503**, not a fabricated success, only if *neither* KV nor Slack accepted the record — a visitor is never thanked for a submission that was completely dropped.

---

4. The traps — each is the reason this outage is worth a page, not just a fix

4.1 Deploying from outside the app directory ships static assets with NO Functions

`wrangler pages deploy <path>` run from anywhere other than inside the project directory (e.g. `wrangler pages deploy apps/landing-io` from the repo root) uploads the static files but does **not** bundle `functions/`. The result is not an error — the site loads fine and `/api/lead` just returns **405**, which looks like a routing problem, not a deploy problem. Always `cd` into the app directory and deploy `.`:

```
cd apps/landing-io
wrangler pages deploy . --project-name=bilko-io --branch=main
```

The log must contain **both** `"✨ Compiled Worker successfully"` **and** `"✨ Uploading Functions bundle"`. Either line missing means the Function did not ship, regardless of what the site's homepage looks like.

4.2 `wrangler kv key delete --force` is not a valid flag in wrangler 4.83

It prints usage help and deletes nothing, with no error exit code that would make a script notice. Use the Cloudflare REST API for deletes during test cleanup (`DELETE /accounts/:acct/storage/kv/namespaces/:ns/values/:key`), and always **re-list the namespace afterward** to confirm the count actually dropped — a delete that silently no-op'd looks identical to one that worked if you only check the HTTP status of the delete call itself.

4.3 Preview inherits the production KV binding by default — this is the normal state, not a misconfiguration

`kv_namespaces` is a non-inheritable key in Cloudflare's own terms, which — counter-intuitively — means the opposite of what "non-inheritable" suggests: with **no** explicit `[env.preview]` override, a `wrangler pages deploy` from any non-production branch applies the **top-level** `kv_namespaces` block to preview too. So by default, every preview deployment of every one of these three projects writes preview traffic straight into the **production** leads namespace. This is exactly what happened to `bilko-cloud` on 2026-07-27 (MC #106359) from an ordinary preview deploy — proven, not theorized: a real preview write was shown landing in the production namespace before the fix, and in a dedicated preview namespace (`BILKO_LEADS_PREVIEW`, id `dcd1f732b2d24850b6c81179beef9909`) after it.

Check this on every project before relying on preview being isolated — it is not the default, and the fact that `landing-io`/`landing-ba` declare `[env.preview] kv_namespaces = []` (no preview binding at all, falls back to Slack-only) while `landing-hr` gets a real preview namespace is a deliberate, stated inconsistency (HR is the only landing with the unauthenticated `conversion_event` path, which made a real isolation test possible there) — not an oversight to "fix" toward uniformity without a reason.

4.4 The HR cookie banner does not block Turnstile — it intercepts the CLICK (MC #106413, OPEN)

Only `landing-hr` has a cookie-consent banner; it is entirely absent from `landing-ba` and `landing-io` (confirmed: zero matches for `cookie-banner` in either file). The first diagnosis of the resulting symptom was wrong and is worth recording as a lesson in itself: it is **not** that the banner blocks Turnstile from loading — Turnstile is not consent-gated at all, and loads and runs its challenge with no consent choice made. What actually happens: the banner is `position:fixed; bottom:0; z-index:999` and occupies the bottom ~84px of the viewport. When a visitor scrolls the form to its natural end position, the submit button's rect lands entirely inside that bottom strip, and the click goes to the banner element, not the button. **The submit handler never runs, so there is no error, no spinner, and no message of any kind** — the single worst failure shape, because the visitor has no reason to try again and there is no signal on our side either. Dismissing the banner (either choice) removes the overlay and the exact same click then works. Status: open, not yet fixed — proposed remedy is bottom padding equal to the banner height while visible, or restricting the banner's clickable area to its own buttons.

4.5 CF Pages projects are not reliably git-integrated — merging to main can deploy nothing, and manual deploys bypass the merge gate entirely (MC #106418, OPEN)

This is the structural root cause that allowed the whole chain in §2 to happen and go unnoticed — not just one more fact about this system. GitHub Actions workflow files nominally exist for auto-deploy on push to each project's `main`, but the GitHub mirror of this repository (`gh-mirror-stale`/`gh-ssh`) is itself frozen roughly 300 commits behind `azdo/main` (documented separately, MC #106042) — so those workflows, even where they still fire, are not deploying current content. The **actual** mechanism in practice is a person running `wrangler pages deploy` by hand from a freshly synced `azdo/main` checkout after a PR merges. That means:

- Merging a PR to `azdo/main` deploys **nothing** by itself — confirmed live: PR 201 (a cosmetic Turnstile attribute cleanup) has been in `azdo/main` since before this outage was fixed, and was still not live on any of the three sites at time of writing, purely because nobody had run the manual deploy step since it merged.
- `wrangler pages deploy` run manually bypasses the azdo PR/CI merge gate entirely — it is possible to ship code to these three production domains that never went through review.
- The only reliable way to know what commit is actually live is `deployment_trigger.metadata.commit_hash` on the Pages deployment record (via the Cloudflare API), **not** `git log azdo/main` and not the GitHub Actions run history.

Open, undecided as of this writing: either give these three projects real git integration (merge → deploy, matching the rest of Bilko's pipeline discipline) or add an explicit deploy stage to the azdo pipeline; failing either of those, at minimum build a drift detector that compares the commit each Pages project is actually serving against `azdo/main`'s tip and alerts on divergence. A deliberate decision was made NOT to deploy PR 201 immediately after these fixes landed, specifically to avoid re-risking freshly-proven-working forms for a cosmetic change — that is a one-time judgment call, not a substitute for the structural fix.

---

5. How to verify a fix here — and what does NOT count

Two specific mistakes already happened in this chain and are worth naming so a third does not:

- **Automated browsers cannot pass Turnstile** (Playwright, headless or headed, real Chrome via CDP — all return `TurnstileError 600010`, the automation refusal). An automated probe of the lead form therefore proves nothing about whether a real visitor

can submit it — it can only ever demonstrate that the anti-spam check is doing its job.
- The `conversion_event` path on `bilko.cloud` has **no Turnstile gate at all**. Verifying a fix through that path proves only that KV writes work in general — exactly the part that was **not** broken. This is precisely the mistake that produced a false "HR lead capture is fixed" report (MC #106193 → corrected the next day as MC #106408): the verification exercised the unprotected event-tracking path and never touched the actual, still-dead lead form.

A real fix is verified by a human driving the actual form in a real browser, then reading the stored record back out of KV and matching a discriminator field (e.g. a company name like `MC106405-HR-DELETE-ME` chosen for the test) against what was typed — followed by deleting the test key and re-listing the namespace to confirm both the write and the cleanup actually happened. This was the exact standard applied on 2026-07-28: a human (team-lead, through the CEO's real Chrome) submitted all three forms for real; every record was read back verbatim; all test keys were removed and the namespace count returned to its pre-test value.

---

6. Related open tickets

MC #	Status	What it is
#106413	OPEN, H	HR cookie-banner intercepts the submit click with zero user-visible feedback — §4.4. Not fixed; a proposed remedy exists but is not implemented.
#106418	OPEN, H	CF Pages drift is structural, not a one-off — §4.5. Two remedies proposed (git integration or a drift detector), neither implemented; PR 201 deliberately not yet deployed.
#106403	OPEN, H	Whether any real customer enquiries arrived during the outage and were lost is **currently unanswerable**: the Slack channel (#ceo, C0AFJDP9V6U) shows zero lead-shaped messages in its last 200, and the newest pre-fix `io_*` KV key predates the outage by three weeks (2026-05-26) — but there is no third, independent source (e.g. Cloudflare Web Analytics request counts) to distinguish "no enquiries arrived" from "enquiries arrived and both capture channels silently dropped them". Whether the Slack bot token even has write permission to that channel had not been separately verified as of this writing.

---

7. Source

Verified against `azdo/main` @ `50e4223a` (2026-07-28) plus live Cloudflare account/API state read at fix and deploy time (account `d0ac2afb6bb5b298723b85a114151a04`).

- `apps/landing-io/functions/api/lead.js`, `apps/landing-io/wrangler.toml`, `apps/landing-io/index.html`
- `apps/landing-hr/functions/api/lead.js`, `apps/landing-hr/wrangler.toml`, `apps/landing-hr/index.html`
- `apps/landing-ba/functions/api/lead.js`, `apps/landing-ba/wrangler.toml`, `apps/landing-ba/index.html`
- `DEPLOY-MAP.md` — "CF Pages projects" section carries the operational recipe and the per-project binding/Turnstile tables this page summarizes; that file is the source for exact command syntax, this page is the narrative and the trap history. Note: at time of writing its "Status per project" table still reads bilko-io/bilko-company as "pending deploy" — that is stale; both were deployed and proven live the same day (§5), and this page reflects the current, corrected state.

Evidence: `~/system/evidence/106193/`, `~/system/evidence/106358/fix-verdict.md` + `deploy-verdict.md`, `~/system/evidence/106359/fix-verdict.md`, `~/system/evidence/106405/fix-verdict.md`.

Personal-data note: any stored lead record quoted in the evidence above has its submitter IP redacted; test submissions used internal `flowforge+<mc>@alai.no` addresses, not real customer data.