# Rollback Plan

# Rollback Plan

> **Project:** Bilko **Version:** 0.1 **Date:** 2026-02-23 **Author:** Ops Architect **Status:** Draft **Reviewers:** Tech Lead, Alem Bašić

# Document History

| Version | Date | Author | Changes |
|---|---|---|---|
| 0.1 | 2026-02-23 | Ops Architect | Initial draft |

---

# 1. Overview

This document defines the rollback procedure for Bilko deployments. A rollback restores the production environment to the previous working state when a deployment causes critical issues.

**Rollback authority:** Alem Bašić (primary). Any engineer can initiate rollback for P0 issues. **Target rollback time:** < 5 minutes for frontend, < 10 minutes for backend, < 60 minutes for database

---

# 2. Rollback Decision Criteria

## Automatic Rollback Triggers

Initiate rollback immediately without waiting:

- Health check `https://api.bilko.io/health` returns non-200 for > 3 consecutive minutes
- Error rate > 5% of all API requests (Sentry)
- Any financial calculation producing provably incorrect results (VAT, double-entry)

- Authentication completely broken (no user can log in)
- Database migrations caused data corruption

## Manual Rollback Triggers (Alem Baši? decision)

- P99 latency > 5s sustained for > 5 minutes
- Critical feature broken with no quick fix available
- Security vulnerability discovered in new release
- User-reported data loss

## Do NOT Roll Back For

- Performance degradation < 20% (investigate first)
- Non-critical feature broken with workaround available
- Minor UI regressions
- Single user reporting an issue (investigate first)

---

# 3. Rollback Procedures by Component

## 3.1 Frontend Rollback (Vercel) — < 5 minutes

Vercel keeps all previous deployments. Rollback is instant (no rebuild).

**Via Vercel Dashboard (recommended):**

1. Open https://vercel.com/alai/bilko/deployments
2. Find the last successful deployment (before current broken one)
3. Click "..." → "Promote to Production"
4. Wait 30 seconds for propagation
5. Verify: `curl -I https://bilko.io` → HTTP/2 200

**Via Vercel CLI:**

```
# List recent deployments
vercel ls --prod

# Promote specific deployment
vercel rollback <deployment-url>
```

**Verification:**

```
curl -I https://bilko.io
# Open bilko.io in browser — should show previous version
```

**Estimated time:** < 2 minutes

---

# 3.2 Backend Rollback (Railway) — < 5 minutes

Railway keeps the last 10 deployments.

**Via Railway Dashboard (recommended):**

1. Open Railway Dashboard → Project → api service
2. Click "Deployments" tab
3. Find last successful deployment (look at deployment timestamp + status)
4. Click "..." → "Redeploy"
5. Wait for deployment to complete (~2 min)
6. Verify health check

**Via Railway CLI:**

```
# List recent deployments
railway deployments list --service api

# Note the previous deployment ID
# Redeploy via dashboard (CLI redeploy not yet supported)
```

**Pre-rollback — if migration was included:**

If the broken release included database migrations, you must decide:

- **Option A (preferred):** Write a forward-fix migration and deploy instead of rolling back
- **Option B:** Database rollback (see 3.3) — use only if Option A is not possible

**Verification:**

```
curl https://api.bilko.io/health
# Expected: {"status":"ok","db":"ok","timestamp":"..."}

# Test auth
curl -X POST https://api.bilko.io/api/v1/auth/login \
  -H "Content-Type: application/json" \
  -d '{"email":"test@bilko.io","password":"test123"}'
```

**Estimated time:** < 5 minutes

---

# 3.3 Database Rollback — < 60 minutes

**WARNING: Database rollbacks are destructive. Any data written since the bad migration WILL BE LOST.**

**Before rolling back database:**

1. Export all data created since the bad migration (if any):

   ```
   railway run psql $DATABASE_URL -c "
   COPY (SELECT * FROM invoices WHERE created_at > '[migration_time]') TO STDOUT;"
   ```

2. Assess data loss: is losing this data acceptable?
3. Prefer forward-fix migration if at all possible

**If rollback is necessary:**

```
# Step 1: Stop API traffic (put up maintenance page)
# Railway → api → Suspend

# Step 2: Restore from pre-deploy backup
# Railway Dashboard → PostgreSQL → Backups → Select backup taken before deploy

# OR restore from manual backup:
railway run psql $DATABASE_URL < pre_deploy_YYYYMMDD_HHMM.dump

# Step 3: Verify backup integrity
railway run psql $DATABASE_URL -c "SELECT COUNT(*) FROM invoices;"
railway run psql $DATABASE_URL -c "SELECT COUNT(*) FROM organizations;"

# Step 4: Redeploy previous backend version (without the bad migration)
# Railway → api → Deployments → Redeploy previous

# Step 5: Verify
railway run npx prisma db pull  # Should match backup schema
curl https://api.bilko.io/health

# Step 6: Resume API traffic
# Railway → api → Resume
```

```
# Step 7: If any data was lost, manually re-enter from exported data
```

**Estimated time:** 30–60 minutes

---

# 4. Rollback Verification Checklist

After any rollback, verify ALL of these before declaring rollback successful:

- ☐ API health: `curl https://api.bilko.io/health` → `{"status":"ok","db":"ok"}`
- ☐ Frontend loads: https://bilko.io opens without errors
- ☐ Login works: test account can authenticate
- ☐ Invoice creation works: create test invoice, verify totals
- ☐ VAT calculation correct: verify Serbia 20% on 1000 RSD = 200 RSD VAT
- ☐ BetterStack: all monitors green
- ☐ Sentry: no new error types
- ☐ Database counts: record counts match pre-deploy snapshot (if DB rollback)

---

# 5. Rollback Communication

## During Rollback

1. Post in Slack #bilko-alerts immediately: "Initiating rollback — [reason]"
2. Update status.bilko.io: "We are investigating an issue and reverting recent changes"
3. Do not provide ETAs until rollback is verified successful

## After Successful Rollback

1. Post in Slack #bilko-alerts: "Rollback complete — previous version restored — investigating root cause"
2. Update status.bilko.io: "Service restored — all systems operational"
3. If any user impact: send email to affected organizations within 2 hours
4. Create incident report within 24 hours

---

# 6. Version-Specific Rollback Notes

| Release | Frontend Tag | Backend Tag | DB Migration Included | Rollback Notes |
|---|---|---|---|---|
| v1.0.0 | Initial | Initial | `0001_initial_schema`, `0002_indexes` | First release — no rollback possible |

*(Update this table with each release)*

---

# 7. Rollback Testing

Each major release must include a rollback test on staging before production deploy:

```
# Deploy to staging
# ... (standard deploy)

# Test rollback on staging
vercel rollback  # Frontend
railway deployments redeploy <previous-id>  # Backend

# Verify staging is back on previous version
curl https://staging-api.bilko.io/health
```

Document rollback test results in deployment checklist.

---

# Related Documents

- Deployment Checklist
- Disaster Recovery Plan
- Operational Runbook
- Incident Report
- DEPLOYMENT.md

---

# Approval

| Role | Name | Date | Signature |
|---|---|---|---|
| Author | Ops Architect | 2026-02-23 | |
| Reviewer | Tech Lead | | |
| Approver | Alem Bašić | | |

Revision #4
Created 2026-02-24 23:11:26 UTC by John
Updated 2026-07-05 20:03:55 UTC by John