Operational Runbook

Operational Runbook

> **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. Service Overview

Service	URL	Platform	Health Check
Frontend	https://bilko.io	Vercel	`curl -I https://bilko.io` → 200
API	https://api.bilko.io	Railway EU West	`curl https://api.bilko.io/health`
Database	bilko_prod	Railway PostgreSQL 15	Health check via API
File storage	bilko-receipts	Cloudflare R2	API upload test
Email	noreply@bilko.io	SendGrid	Test email send

On-call: Alem Bašić (+47 40 47 42 51)

---

2. Routine Operations

2.1 Check System Health

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

# Frontend
curl -I https://bilko.io
# Expected: HTTP/2 200

# Railway logs (last 50 lines)
railway logs --tail 50

# Railway metrics (via dashboard)
# Railway Dashboard → Project → api → Metrics
```

2.2 Deploy New Version

Standard deploy (automatic):

1. Merge PR to `main` branch
2. GitHub Actions CI pipeline runs automatically
3. On pass: Vercel and Railway auto-deploy
4. Monitor: Railway logs + BetterStack for 15 min post-deploy

Manual deploy (emergency):

```
# Deploy frontend manually
cd apps/web && vercel --prod

# Deploy backend manually
railway up --service api --environment production

# Run migrations before backend deploy
railway run npx prisma migrate deploy
```

2.3 Database Migrations

Never run migrations directly in production without backup:

```
# Step 1: Take backup
railway run pg_dump $DATABASE_URL -f backup_$(date +%Y%m%d_%H%M).dump
```

```
# Step 2: Test migration on staging
railway run --environment staging npx prisma migrate deploy

# Step 3: Apply to production (after staging verification)
railway run npx prisma migrate deploy

# Step 4: Verify
railway run npx prisma db pull  # Confirm schema matches
```

2.4 View Application Logs

```
# Railway API logs (streaming)
railway logs --service api

# Railway API logs (last 100 lines)
railway logs --service api --tail 100

# Filter for errors
railway logs --service api | grep -i error

# Filter for specific organization
railway logs --service api | grep "organizationId=<uuid>"
```

2.5 Environment Variable Updates

```
# View current env vars (Railway)
railway variables list --service api

# Update a secret (Railway CLI)
railway variables set JWT_SECRET=<new-value> --service api --environment production
# Restart service after update
railway service restart --service api

# Vercel env var update
vercel env add NEXT_PUBLIC_API_URL production
```

---

3. Monitoring & Alerting

3.1 Normal Operating Ranges

Metric	Normal Range	Alert if
API CPU	5–30%	> 70% for 5 min
API Memory	200–800MB	> 1.5GB
DB connections	2–10 active	> 20 active
API P95 latency	< 200ms	> 1000ms
Error rate (5xx)	< 0.1%	> 1%
Uptime	100%	Any downtime > 2 min

3.2 BetterStack Dashboards (PLANNED)

- System status: status.bilko.io
- Internal metrics: BetterStack dashboard
- Uptime history: BetterStack → Monitors

3.3 Sentry Error Monitoring (PLANNED)

- Frontend errors: Sentry → bilko-frontend project
- Backend errors: Sentry → bilko-backend project
- Financial logic errors: tagged `financial-logic` in Sentry — P0 response required

---

4. Incident Response Procedures

4.1 API Down (all requests failing)

1. Check Railway Dashboard → api → Status
2. Check health endpoint: `curl https://api.bilko.io/health`
3. Check Railway logs: `railway logs --service api --tail 50`
4. Common causes and fixes:

Cause	Fix
Recent bad deploy	Railway → Deployments → Redeploy previous

Cause	Fix
Out of memory (OOM)	Restart service, investigate memory leak
Database connection exhausted	Restart service, check PgBouncer config
Database down	See 4.2

5. If not resolved in 5 min → post to #bilko-alerts, start incident response

4.2 Database Issues

Connection refused:

```
# Check Railway PostgreSQL status
# Railway Dashboard → Project → PostgreSQL → Status

# Test connection manually
railway run psql $DATABASE_URL -c "SELECT 1;"
```

High connection count:

```
# Check active connections
railway run psql $DATABASE_URL -c "
SELECT count(*), state, wait_event_type
FROM pg_stat_activity
GROUP BY state, wait_event_type
ORDER BY count DESC;"

# Kill idle connections if needed (after investigation)
railway run psql $DATABASE_URL -c "
SELECT pg_terminate_backend(pid)
FROM pg_stat_activity
WHERE state = 'idle'
  AND state_change < NOW() - INTERVAL '10 minutes';"
```

Slow queries:

```
# Find long-running queries
railway run psql $DATABASE_URL -c "
SELECT pid, now() - query_start AS duration, query
FROM pg_stat_activity
WHERE state = 'active' AND query_start < NOW() - INTERVAL '5 seconds'
```

```
ORDER BY duration DESC;"
```

4.3 High Error Rate

1. Open Sentry → bilko-backend → Issues → Sort by date
2. Identify most frequent error in last 15 min
3. Check if error is from recent deploy: Railway → Deployments → check timestamp
4. If financial logic error (VAT/double-entry): **treat as P0, rollback immediately**
5. If non-financial error: assess impact, investigate root cause before rollback

4.4 Storage (R2) Issues

```
# Test R2 connectivity from API
railway run node -e "
const { S3Client, HeadBucketCommand } = require('@aws-sdk/client-s3');
const client = new S3Client({ endpoint: process.env.R2_ENDPOINT, ... });
client.send(new HeadBucketCommand({ Bucket: 'bilko-receipts' }))
  .then(() => console.log('R2 OK'))
  .catch(e => console.error('R2 Error:', e.message));"
```

R2 outage: file uploads fail but core accounting functionality works. Log errors, retry uploads. R2 outages are typically < 15 min.

4.5 Email Delivery Failure

```
# Check SendGrid activity
# SendGrid Dashboard → Activity → Filter by date → Look for bounces/blocks

# Test email sending manually
railway run node -e "
const sgMail = require('@sendgrid/mail');
sgMail.setApiKey(process.env.SENDGRID_API_KEY);
sgMail.send({ to: 'alem@alai.no', from: 'noreply@bilko.io', subject: 'Test', text: 'Test' })
  .then(() => console.log('Email sent'))
  .catch(e => console.error('Email error:', e.message));"
```

Email failure: invoices cannot be sent. Users can still create and download invoices. Not P0 but resolve within 2h.

---

5. Maintenance Operations

5.1 Routine Backup Verification (Monthly)

```
# Verify backup exists and restore to staging
# Railway Dashboard → PostgreSQL → Backups → Select latest

# Download backup
railway run pg_dump $DATABASE_URL -f monthly_verify_$(date +%Y%m).dump

# Restore to staging and verify counts
railway run psql $STAGING_DB_URL < monthly_verify_$(date +%Y%m).dump
railway run psql $STAGING_DB_URL -c "SELECT COUNT(*) FROM organizations;"
railway run psql $STAGING_DB_URL -c "SELECT COUNT(*) FROM invoices;"
```

5.2 Certificate Renewal

Certificates auto-renew via Vercel (Let's Encrypt) and Railway. Monitor expiry dates:

```
echo | openssl s_client -connect bilko.io:443 2>/dev/null | openssl x509 -noout -dates
echo | openssl s_client -connect api.bilko.io:443 2>/dev/null | openssl x509 -noout -dates
```

Alert if expiry < 30 days.

5.3 Secret Rotation (Annual / On Compromise)

```
# Generate new JWT secrets
openssl rand -base64 32  # JWT_SECRET
openssl rand -base64 32  # JWT_REFRESH_SECRET

# Update in Railway (this will invalidate ALL existing sessions)
railway variables set JWT_SECRET=<new> --service api --environment production
railway variables set JWT_REFRESH_SECRET=<new> --service api --environment production
railway service restart --service api

# Notify users: all sessions invalidated, need to log in again
```

---

6. Useful Commands Reference

```
# Railway CLI quick reference
railway login                          # Authenticate
railway status                         # Project status
railway logs --service api             # Stream logs
railway run <command>                  # Run in Railway context
railway variables list --service api   # Show env vars
railway service restart --service api  # Restart service

# Database quick reference
railway run psql $DATABASE_URL         # Connect to database
railway run npx prisma studio          # Database GUI (opens on localhost:5555)
railway run npx prisma migrate deploy  # Apply pending migrations

# Vercel quick reference
vercel ls                              # List deployments
vercel --prod                          # Deploy to production
vercel rollback                        # Rollback to previous
```

Related Documents

- Monitoring & Observability
- Disaster Recovery Plan
- Incident Report
- Go-Live Runbook

Approval

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

Revision #3
Created 2026-02-24 23:11:22 UTC by John
Updated 2026-05-31 20:04:08 UTC by John