Deployment & Environment

Environment Setup

Drop Environment Configuration

Last updated: 2026-02-13 **Source:** `src/drop-app/package.json`, `next.config.ts`, `Dockerfile`, `docker-compose.yml`, `fly.toml`

Technology Stack

Layer	Technology	Version	Source
Runtime	Node.js	22 (Alpine)	`Dockerfile:2`
Framework	Next.js	16.1.6	`package.json:14`
UI	React	19.2.3	`package.json:15-16`
Database (all environments)	PostgreSQL 16 via Drizzle ORM	drizzle-orm	`src/shared/db/schema.ts`
Auth	JWT via jose	^6.1.3	`package.json:8`
Password hashing	bcryptjs	^3.0.3	`package.json:5`
Styling	Tailwind CSS	^4	`package.json:33`
UI Components	Radix UI	^1.4.3	`package.json:13`
Icons	Lucide React	^0.563.0	`package.json:9`
Theme	next-themes	^0.4.6	`package.json:10`
Toasts	Sonner	^2.0.7	`package.json:17`

Dev Dependencies

Tool	Version	Purpose	Source
Vitest	^4.0.18	Unit/integration testing	`package.json:36`
Playwright	^1.58.2	E2E testing	`package.json:21`
TypeScript	^5	Type checking	`package.json:35`
ESLint	^9	Linting	`package.json:29`
shadcn	^3.8.4	UI component generation	`package.json:32`

NPM Scripts

Source: `src/drop-app/package.json:5-12`

Script	Command	Description
`dev`	`next dev`	Start development server (port 3000)
`build`	`next build`	Build for production (standalone output)
`start`	`next start`	Start production server
`lint`	`eslint`	Run ESLint
`test`	`vitest run`	Run unit/integration tests (single run)
`test:watch`	`vitest`	Run tests in watch mode

Next.js Configuration

Source: `src/drop-app/next.config.ts:1-49`

Setting	Value	Purpose
`output`	`"standalone"`	Self-contained server for Docker (`next.config.ts:4`)
`devIndicators`	`false`	Disable dev indicators (`next.config.ts:5`)

Security Headers

All responses include these headers (configured in `next.config.ts:6-58`):

Header	Value (Production)	Value (Development)	Purpose
Content-Security-Policy	`default-src 'self'; script-src 'self'; style-src 'self' 'unsafe-inline'; font-src 'self'; img-src 'self' data: blob:; connect-src 'self'; frame-ancestors 'none'`	`default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; style-src 'self' 'unsafe-inline'; font-src 'self'; img-src 'self' data: blob:; connect-src 'self'; frame-ancestors 'none'`	XSS and injection protection
X-Frame-Options	`DENY`	`DENY`	Clickjacking prevention
X-Content-Type-Options	`nosniff`	`nosniff`	MIME sniffing prevention

Header	Value (Production)	Value (Development)	Purpose
Referrer-Policy	`strict-origin-when-cross-origin`	`strict-origin-when-cross-origin`	Referrer leakage prevention
Permissions-Policy	`camera=(self), microphone=(), geolocation=(self)`	`camera=(self), microphone=(), geolocation=(self)`	Feature restriction
Strict-Transport-Security	`max-age=63072000; includeSubDomains; preload`	`max-age=63072000; includeSubDomains; preload`	Force HTTPS

Note: CSP is stricter in production (no `unsafe-eval` for scripts). Development mode allows `unsafe-inline` and `unsafe-eval` for HMR (Hot Module Replacement) to work.

Environment Modes

Development

- `NODE_ENV=development` (default)
- Demo user seeded automatically
- Login page shows demo credentials hint
- In-memory rate limiting fallback
- PostgreSQL 16 via Docker (`docker compose up -d`), port 5433

Production

- `NODE_ENV=production`
- Demo seed data disabled
- `JWT_SECRET` required (fatal error if missing)
- Cookies set with `secure: true`
- PostgreSQL 16 on AWS RDS via `DATABASE_URL`

Test

- `NODE_ENV=test`
- PostgreSQL 16 test database (`drop_test`), created via `pg-test-db.ts` helper
- Tables truncated between tests; schema pushed via Drizzle before suite runs
- Mocked Next.js modules (server, headers)

Port Mapping

Service	Internal Port	External Port	Protocol
Drop App	3000	3000	HTTP
PostgreSQL (local dev)	5432	5433	TCP
PostgreSQL (production RDS)	5432	5432	TCP

Docker Image Details

Base: `node:22-alpine` **User:** `nextjs` (UID 1001) **Working dir:** `/app` **Exposed port:** 3000
Entrypoint: `node server.js` **Build context:** `src/drop-app/`

Image contents (runner stage):

- `/app/public/` -- Static assets
- `/app/.next/standalone/` -- Next.js standalone server
- `/app/.next/static/` -- Static build output

Secrets Management

Secrets Management

Last updated: 2026-02-17 **Source:** `src/drop-app/src/lib/secrets.ts`

Overview

Drop uses an abstracted secrets management system with pluggable providers. The system is backward compatible -- if no secrets provider is configured, it reads directly from environment variables (existing behavior).

Provider Selection

The provider is selected automatically based on which environment variables are set:

Priority	Condition	Provider	Description
1	`DOPPLER_TOKEN` set	Doppler	Cloud secrets manager via Doppler API
2	`AWS_SECRET_ARN` set	AWS	AWS Secrets Manager (requires AWS SDK)
3	(default)	env	Reads from `process.env`

Initialization (call once at app startup):

```
import { initSecrets } from '@/lib/secrets';

// Auto-detect provider based on env vars
initSecrets();

// Optional: custom cache TTL (default 5 minutes)
initSecrets({ ttlMs: 10 * 60 * 1000 }); // 10 minutes
```

Usage:

```
import { getSecret } from '@/lib/secrets';

const jwtSecret = await getSecret('JWT_SECRET');
const dbUrl = await getSecret('DATABASE_URL');
```

Caching

All secret values are cached in memory with a configurable TTL (default: 5 minutes). This reduces API calls to external providers while ensuring secrets are refreshed periodically.

- Cache is cleared on `initSecrets()` call
- Cache entries expire individually based on TTL
- If a provider returns `undefined`, the system falls back to `process.env`

Rotation Procedures

JWT_SECRET

Impact: All active user sessions will be invalidated.

1. Generate new secret: `openssl rand -base64 48`
2. Update in secrets provider (Doppler/AWS/env)
3. Call `rotateSecret('JWT_SECRET', newValue)` or restart the app
4. Users will need to log in again

Recommended frequency: Every 90 days or after a suspected compromise.

DATABASE_URL (PostgreSQL credentials)

Impact: Application loses DB connectivity until updated.

1. Create new PostgreSQL credentials
2. Update PostgreSQL user: `ALTER USER drop WITH PASSWORD 'new_value';`
3. Update `DATABASE_URL` in secrets provider with new credentials
4. Restart the application (or call `rotateSecret`)

Recommended frequency: Every 90 days.

SENTRY_DSN

Status: REMOVED (MC #1271 — Sentry deinstalled)

SLACK_WEBHOOK_URL

Impact: Alerts stop sending to Slack until updated.

1. Create new incoming webhook in Slack workspace
2. Update `SLACK_WEBHOOK_URL` in secrets provider
3. Restart the application

Recommended frequency: Only on suspected compromise.

Open Banking API Keys

Impact: Bank connectivity (AISP/PISP) stops working.

1. Regenerate keys in the Open Banking provider dashboard
2. Update the relevant env vars in secrets provider
3. Restart the application
4. Verify bank account connectivity via `/api/health`

Recommended frequency: Per provider policy or every 180 days.

Environment Setup per Provider

Environment Variables (Default)

No setup required. Set secrets as environment variables:

```
# .env.local (development)
JWT_SECRET=dev-secret-do-not-use-in-production

# Production (Fly.io)
fly secrets set JWT_SECRET="$(openssl rand -base64 48)"
fly secrets set DATABASE_URL="postgresql://..."

# Production (Docker)
```

```
# Pass via -e flags or docker-compose environment section
```

Doppler

1. Create account at doppler.com
2. Create project "drop" with environments (dev, staging, production)
3. Add all secrets in the Doppler dashboard
4. Generate a service token for each environment
5. Set `DOPPLER_TOKEN` in your deployment:

```
# Fly.io
fly secrets set DOPPLER_TOKEN="dp.st.production.xxxxx"


# Docker (pass as environment variable)
```

AWS Secrets Manager

1. Create a secret in AWS Secrets Manager (JSON format):

   ```
   {
     "JWT_SECRET": "your-jwt-secret",
     "DATABASE_URL": "postgresql://...",
     "SLACK_WEBHOOK_URL": "https://..."
   }
   ```

2. Note the secret ARN
3. Ensure the application has IAM permissions for `secretsmanager:GetSecretValue`
4. Install the AWS SDK: `npm install @aws-sdk/client-secrets-manager`
5. Set `AWS_SECRET_ARN` in your deployment

Audit Trail

All secret rotation events are logged to the `audit_log` table:

Field	Value
action	`secret_rotated`
resource_type	`secret`
resource_id	Secret key name (e.g., `JWT_SECRET`)
details	JSON with provider name and rotation timestamp

Query rotation history:

```
SELECT * FROM audit_log
WHERE action = 'secret_rotated'
ORDER BY timestamp DESC;
```

Deployment Checklist

Deployment Checklist: [PROJECT NAME]

Release: v[X.Y.Z] **Date:** YYYY-MM-DD **Deploy Lead:** DevOps **Approved by:** Tech Lead + John
Environment: Staging → Production

Pre-Deployment (T-1 Day)

Verification

- ☐ All tests passing in CI
- ☐ Code review approved and merged
- ☐ UAT sign-off received
- ☐ Release notes prepared
- ☐ No critical/high open bugs

Preparation

- ☐ Database backup completed
- ☐ Staging environment matches production config
- ☐ Rollback procedure tested
- ☐ Stakeholders notified of deployment window
- ☐ On-call person confirmed and available

Configuration

- ☐ Environment variables verified
- ☐ API keys / secrets rotated if needed

- [ ] DNS changes prepared (if applicable)
- [ ] SSL certificates valid (expiry > 30 days)
- [ ] Third-party service limits adequate

Deployment (T-0)

Window

- **Allowed:** Tue-Thu, 10:00-16:00
- **Never:** Fridays
- **Hotfix:** Anytime business hours (Tech Lead + John approval)
- **Emergency:** Anytime (John + Alem approval)

Execution

- [ ] Announce deployment start in channel
- [ ] Deploy to staging — verify
- [ ] Run staging smoke tests
- [ ] Manual approval gate — Tech Lead confirms
- [ ] Deploy to production
- [ ] Monitor deployment logs for errors

Post-Deployment (T+0)

Smoke Tests

- [ ] Homepage loads correctly
- [ ] Authentication works (login/logout)
- [ ] Core user flow #1 works
- [ ] Core user flow #2 works
- [ ] API health endpoint returns 200
- [ ] No errors in error tracking (Sentry)

Monitoring (First 30 Minutes)

- [ ] Error rate normal (< 1%)
- [ ] Response times normal (p95 < 500ms)
- [ ] No 5xx errors in logs
- [ ] Database connections stable
- [ ] Memory/CPU usage normal

Communication

- [ ] Announce deployment complete
- [ ] Send release notes to stakeholders
- [ ] Update project status

Rollback Plan

Rollback Triggers

- Critical functionality broken
- Data integrity issues
- Security vulnerability discovered
- Error rate > 5%
- Response time > 3x normal

Rollback Procedure

- [ ] Announce rollback in channel
- [ ] Revert to previous version
- [ ] Restore database backup (if schema changed)
- [ ] Verify rollback successful
- [ ] Announce rollback complete
- [ ] Create incident report

Rollback Time Targets

- Application rollback: < 15 minutes
- Database rollback: < 30 minutes
- Full rollback: < 1 hour

Sign-off

Role	Name	Pre-Deploy	Post-Deploy
DevOps		☐	☐
Tech Lead		☐	☐
John		☐	☐

DR Runbook

Drop — Disaster Recovery Runbook

Infrastructure Overview

Production Environment

- **Service:** AWS App Runner
- **Region:** eu-west-1 (Ireland)
- **Service ARN:** `arn:aws:apprunner:eu-west-1:324480209768:service/drop-web/8e45b0d335304487a1880f4e32d6aeec`
- **Service URL:** https://9ef3szvvsb.eu-west-1.awsapprunner.com
- **ECR Repository:** `324480209768.dkr.ecr.eu-west-1.amazonaws.com/drop-web`

Database

- **RDS Instance:** drop-db
- **Endpoint:** `drop-db.czu2qe4quy4v.eu-west-1.rds.amazonaws.com:5432`
- **Database Name:** dropapp
- **Username:** dropuser
- **Backup Strategy:** Automated snapshots, 7-day retention
- **Backup Window:** 23:24-23:54 UTC daily

Staging Environment

- **Platform:** Fly.io
- **App Name:** drop-staging
- **Region:** arn (Stockholm)
- **Database:** PostgreSQL 16 (RDS, eu-north-1, or Docker in CI)

Domain

- **Production:** getdrop.no (future)

- **Current:** App Runner subdomain

Backup Strategy

RDS PostgreSQL (Production)

- **Automated Snapshots:** Daily at 23:24 UTC
- **Retention Period:** 7 days
- **Point-in-Time Recovery:** Enabled (5-minute granularity)
- **Manual Snapshots:** Created before major changes
- **Storage:** Same region (eu-west-1)

Staging PostgreSQL (RDS)

- **Automated Snapshots:** Daily, 7-day retention (same config as production)
- **Backup Method:** Manual export via `flyctl ssh console` and `pg_dump` (PostgreSQL 16 — `sqlite3` no longer applies; see ADR-014)
- **Recommended:** Export before major changes

Recovery Procedures

Scenario 1: App Runner Service Down

Symptoms

- Service health checks failing
- 5xx errors from App Runner URL
- CloudWatch alarms triggered

Investigation Steps

```
# 1. Check service status
aws apprunner describe-service \
  --service-arn arn:aws:apprunner:eu-west-1:324480209768:service/drop-
web/8e45b0d335304487a1880f4e32d6aeec \
  --region eu-west-1
```

```
# 2. View recent logs (last 10 minutes)
aws logs tail /aws/apprunner/drop-web/8e45b0d335304487a1880f4e32d6aeec/application \
  --follow \
  --since 10m \
  --region eu-west-1

# 3. Check deployment history
aws apprunner list-operations \
  --service-arn arn:aws:apprunner:eu-west-1:324480209768:service/drop-
web/8e45b0d335304487a1880f4e32d6aeec \
  --region eu-west-1
```

Recovery Actions

Option A: Restart Service

```
# Trigger new deployment (no code change)
aws apprunner start-deployment \
  --service-arn arn:aws:apprunner:eu-west-1:324480209768:service/drop-
web/8e45b0d335304487a1880f4e32d6aeec \
  --region eu-west-1

# Monitor deployment status
aws apprunner describe-service \
  --service-arn arn:aws:apprunner:eu-west-1:324480209768:service/drop-
web/8e45b0d335304487a1880f4e32d6aeec \
  --query 'Service.Status' \
  --region eu-west-1
```

Option B: Rollback to Previous Image

```
# 1. List recent ECR images
aws ecr describe-images \
  --repository-name drop-web \
  --region eu-west-1 \
  --query 'sort_by(imageDetails,& imagePushedAt)[-5:]'

# 2. Update service to use previous image tag
# (Manual step: Update .github/workflows/deploy-aws.yml with previous tag and push)

# 3. Or update directly via App Runner console (rollback to previous deployment)
```

RTO: 5-10 minutes (restart) / 15-20 minutes (rollback)

Scenario 2: RDS Database Failure

Symptoms

- Connection timeouts to `drop-db.czu2qe4quy4v.eu-west-1.rds.amazonaws.com`
- Database errors in App Runner logs
- RDS CloudWatch metrics show instance down

Investigation Steps

```
# 1. Check RDS instance status
aws rds describe-db-instances \
  --db-instance-identifier drop-db \
  --region eu-west-1 \
  --query 'DBInstances[0].DBInstanceStatus'

# 2. Check for automated snapshots
aws rds describe-db-snapshots \
  --db-instance-identifier drop-db \
  --region eu-west-1 \
  --query 'DBSnapshots[?SnapshotType==`automated`] | sort_by(@, &SnapshotCreateTime)[-5:]'

# 3. Review recent events
aws rds describe-events \
  --source-identifier drop-db \
  --source-type db-instance \
  --region eu-west-1 \
  --duration 60
```

Recovery Actions

Option A: Restore from Latest Automated Snapshot

```
# 1. Identify latest snapshot
LATEST_SNAPSHOT=$(aws rds describe-db-snapshots \
  --db-instance-identifier drop-db \
  --region eu-west-1 \
  --query 'DBSnapshots[?SnapshotType==`automated`] | sort_by(@, &SnapshotCreateTime)[-
1].DBSnapshotIdentifier' \
```

```
  --output text)

echo "Latest snapshot: $LATEST_SNAPSHOT"

# 2. Restore to new instance
aws rds restore-db-instance-from-db-snapshot \
  --db-instance-identifier drop-db-restored \
  --db-snapshot-identifier $LATEST_SNAPSHOT \
  --db-instance-class db.t4g.micro \
  --vpc-security-group-ids sg-XXXXX \
  --db-subnet-group-name default \
  --region eu-west-1

# 3. Wait for restore to complete (10-20 minutes)
aws rds wait db-instance-available \
  --db-instance-identifier drop-db-restored \
  --region eu-west-1

# 4. Update DATABASE_URL in App Runner
# (Manual step: Update environment variable via AWS Console or CLI)

# 5. Verify connection
NEW_ENDPOINT=$(aws rds describe-db-instances \
  --db-instance-identifier drop-db-restored \
  --query 'DBInstances[0].Endpoint.Address' \
  --output text \
  --region eu-west-1)

echo "New endpoint: $NEW_ENDPOINT"
```

Option B: Point-in-Time Recovery

```
# Restore to specific timestamp (e.g., 1 hour ago)
aws rds restore-db-instance-to-point-in-time \
  --source-db-instance-identifier drop-db \
  --target-db-instance-identifier drop-db-pitr \
  --restore-time $(date -u -d '1 hour ago' '+%Y-%m-%dT%H:%M:%SZ') \
  --db-instance-class db.t4g.micro \
  --region eu-west-1
```

```
# Wait for restore
aws rds wait db-instance-available \
  --db-instance-identifier drop-db-pitr \
  --region eu-west-1
```

RPO: 24 hours (snapshot) / 5 minutes (PITR) **RTO:** 30 minutes (snapshot) / 30 minutes (PITR)

Scenario 3: Data Corruption

Symptoms

- Application reports data inconsistencies
- Missing or incorrect records in database
- User reports of lost data

Investigation Steps

```
# 1. Connect to RDS and inspect data
psql -h drop-db.czu2qe4quy4v.eu-west-1.rds.amazonaws.com \
     -U dropuser \
     -d dropapp \
     -c "SELECT COUNT(*) FROM users WHERE deleted_at IS NOT NULL;"

# 2. Check audit_log table for suspicious activity
psql -h drop-db.czu2qe4quy4v.eu-west-1.rds.amazonaws.com \
     -U dropuser \
     -d dropapp \
     -c "SELECT * FROM audit_log WHERE action IN ('DELETE', 'UPDATE') ORDER BY timestamp DESC
LIMIT 50;"

# 3. Identify time of corruption
# Review application logs and database query logs
```

Recovery Actions

Option A: Selective Data Restore (if corruption is isolated)

```
# 1. Create temporary snapshot of current state
aws rds create-db-snapshot \
  --db-instance-identifier drop-db \
  --db-snapshot-identifier drop-db-before-restore-$(date +%Y%m%d-%H%M) \
```

```
  --region eu-west-1

# 2. Restore clean snapshot to temporary instance
CLEAN_SNAPSHOT=<snapshot-before-corruption>

aws rds restore-db-instance-from-db-snapshot \
  --db-instance-identifier drop-db-temp \
  --db-snapshot-identifier $CLEAN_SNAPSHOT \
  --db-instance-class db.t4g.micro \
  --region eu-west-1

# 3. Export affected tables from clean instance
pg_dump -h <temp-endpoint> \
        -U dropuser \
        -d dropapp \
        -t users \
        -t transactions \
        --data-only \
        > clean_data.sql

# 4. Selectively import into production (after verification)
psql -h drop-db.czu2qe4quy4v.eu-west-1.rds.amazonaws.com \
     -U dropuser \
     -d dropapp \
     < clean_data.sql

# 5. Terminate temporary instance
aws rds delete-db-instance \
  --db-instance-identifier drop-db-temp \
  --skip-final-snapshot \
  --region eu-west-1
```

Option B: Full Database Restore (see Scenario 2)

RTO: 1-2 hours (selective) / 30 minutes (full restore) **RPO:** Depends on snapshot age

---

Scenario 4: Full Region Outage (eu-west-1)

Current State

- **No automated cross-region failover**
- **No replica in secondary region**
- **Manual failover required**

Investigation Steps

```
# 1. Check AWS Service Health Dashboard
# https://health.aws.amazon.com/health/status

# 2. Verify RDS snapshots are accessible
aws rds describe-db-snapshots \
  --db-instance-identifier drop-db \
  --region eu-west-1

# 3. Check ECR images (may need to copy to secondary region)
aws ecr describe-images \
  --repository-name drop-web \
  --region eu-west-1
```

Recovery Actions (Manual Failover to eu-north-1)

```
# 1. Copy latest RDS snapshot to eu-north-1
LATEST_SNAPSHOT=$(aws rds describe-db-snapshots \
  --db-instance-identifier drop-db \
  --region eu-west-1 \
  --query 'DBSnapshots[?SnapshotType==`automated`] | sort_by(@, &SnapshotCreateTime)[-
1].DBSnapshotIdentifier' \
  --output text)

aws rds copy-db-snapshot \
  --source-db-snapshot-identifier arn:aws:rds:eu-west-1:324480209768:snapshot:$LATEST_SNAPSHOT
\
  --target-db-snapshot-identifier drop-db-failover-$(date +%Y%m%d) \
  --region eu-north-1

# 2. Restore RDS in eu-north-1
aws rds restore-db-instance-from-db-snapshot \
  --db-instance-identifier drop-db-failover \
  --db-snapshot-identifier drop-db-failover-$(date +%Y%m%d) \
  --db-instance-class db.t4g.micro \
  --region eu-north-1
```

```
# 3. Copy ECR image to eu-north-1
# (Manual: create ECR repo in eu-north-1, retag and push latest image)

# 4. Deploy App Runner in eu-north-1
# (Manual: create new App Runner service via console with failover database endpoint)

# 5. Update DNS (when getdrop.no is active)
# Point getdrop.no to new App Runner URL
```

RTO: 2-4 hours (manual process) **RPO:** Last snapshot before outage (24 hours worst case, 5 minutes with PITR if available)

Scenario 5: Security Incident

Symptoms

- Suspicious database activity
- Unauthorized access attempts
- AML alerts triggered
- STR report filed

Investigation Steps

```
# 1. Check audit logs for suspicious activity
psql -h drop-db.czu2qe4quy4v.eu-west-1.rds.amazonaws.com \
     -U dropuser \
     -d dropapp \
     -c "SELECT * FROM audit_log WHERE timestamp > NOW() - INTERVAL '24 hours' ORDER BY
timestamp DESC;"

# 2. Review AML alerts
psql -h drop-db.czu2qe4quy4v.eu-west-1.rds.amazonaws.com \
     -U dropuser \
     -d dropapp \
     -c "SELECT * FROM aml_alerts WHERE status = 'open' OR created_at > NOW() - INTERVAL '24
hours';"

# 3. Check AWS CloudTrail for API activity
aws cloudtrail lookup-events \
```

```
  --lookup-attributes AttributeKey=ResourceName,AttributeValue=drop-db \
  --region eu-west-1 \
  --max-results 50

# 4. Review App Runner access logs
aws logs filter-log-events \
  --log-group-name /aws/apprunner/drop-web/8e45b0d335304487a1880f4e32d6aeec/application \
  --start-time $(date -u -d '24 hours ago' +%s)000 \
  --region eu-west-1
```

Containment Actions

```
# 1. Revoke compromised sessions
psql -h drop-db.czu2qe4quy4v.eu-west-1.rds.amazonaws.com \
     -U dropuser \
     -d dropapp \
     -c "UPDATE sessions SET revoked = 1 WHERE user_id IN (SELECT user_id FROM aml_alerts
WHERE status = 'open');"

# 2. Temporarily disable affected users
psql -h drop-db.czu2qe4quy4v.eu-west-1.rds.amazonaws.com \
     -U dropuser \
     -d dropapp \
     -c "UPDATE users SET kyc_status = 'rejected' WHERE id IN (SELECT user_id FROM aml_alerts
WHERE severity = 'critical');"

# 3. Rotate database credentials
aws rds modify-db-instance \
  --db-instance-identifier drop-db \
  --master-user-password <new-password> \
  --apply-immediately \
  --region eu-west-1

# Update DATABASE_URL in App Runner with new password

# 4. Enable enhanced monitoring
aws rds modify-db-instance \
  --db-instance-identifier drop-db \
  --monitoring-interval 1 \
  --monitoring-role-arn arn:aws:iam::324480209768:role/rds-monitoring-role \
```

```
  --region eu-west-1

# 5. Take forensic snapshot
aws rds create-db-snapshot \
  --db-instance-identifier drop-db \
  --db-snapshot-identifier drop-db-incident-$(date +%Y%m%d-%H%M) \
  --region eu-west-1
```

Investigation & Remediation

1. **Analyze audit logs** — identify scope of breach
2. **File STR reports** — if financial crime suspected (via `str_reports` table)
3. **Notify Finanstilsynet** — if user data compromised (GDPR requirement)
4. **Update security policies** — patch vulnerabilities
5. **User communication** — notify affected users if required by GDPR

RTO: Immediate containment (revoke sessions) / 24-48 hours full investigation

---

RTO/RPO Targets

Scenario	RTO	RPO
App Runner restart	5-10 minutes	0 (no data loss)
App Runner rollback	15-20 minutes	0 (no data loss)
RDS snapshot restore	30 minutes	24 hours (last snapshot)
RDS PITR restore	30 minutes	5 minutes (PITR granularity)
Full region failover	2-4 hours	24 hours (manual process)
Security incident containment	Immediate	0 (logs preserved)

Contacts

Primary

- **Alem Bašić (CEO):** +47 40 47 42 51
- **Email:** alem@alai.no

AI Operations

- **John (AI Director):** Slack #drop-alerts channel

External Support

- **AWS Support:** Premium support via AWS Console
- **Fly.io Support:** Email support@fly.io

Runbook Maintenance

Review Schedule

- **Quarterly review** — verify all ARNs, endpoints, and procedures
- **After incidents** — update based on lessons learned
- **Before major releases** — verify backup and rollback procedures

Test Schedule

- **Annually** — full DR drill (restore from snapshot to temporary instance)
- **Quarterly** — App Runner restart and rollback tests
- **Monthly** — verify snapshot creation and retention

Change Log

Date	Change	Author
2026-02-18	Initial version created	Builder 3 (AI)

Appendix: Useful Commands

Quick Health Check

```
# Check App Runner status
aws apprunner describe-service \
```

```
  --service-arn arn:aws:apprunner:eu-west-1:324480209768:service/drop-
web/8e45b0d335304487a1880f4e32d6aeec \
  --query 'Service.Status' \
  --output text \
  --region eu-west-1

# Check RDS status
aws rds describe-db-instances \
  --db-instance-identifier drop-db \
  --query 'DBInstances[0].DBInstanceStatus' \
  --output text \
  --region eu-west-1

# Check latest snapshot age
aws rds describe-db-snapshots \
  --db-instance-identifier drop-db \
  --region eu-west-1 \
  --query 'DBSnapshots[?SnapshotType==`automated`] | sort_by(@, &SnapshotCreateTime)[-
1].SnapshotCreateTime' \
  --output text
```

Database Connection Test

```
# Test connection from local machine
psql -h drop-db.czu2qe4quy4v.eu-west-1.rds.amazonaws.com \
     -U dropuser \
     -d dropapp \
     -c "SELECT 1;"
```

Log Streaming

```
# Stream App Runner application logs
aws logs tail /aws/apprunner/drop-web/8e45b0d335304487a1880f4e32d6aeec/application \
  --follow \
  --region eu-west-1

# Stream RDS error logs
aws rds download-db-log-file-portion \
  --db-instance-identifier drop-db \
```

```
--log-file-name error/postgresql.log \
--region eu-west-1
```

Deployment Guide

Drop Deployment Guide

Last updated: 2026-03-03 **Source:** `src/drop-app/Dockerfile`, `docker-compose.yml`, `DOCKER.md`

> **NOTE (2026-03-03):** This document was updated for ADR-014 (PostgreSQL-only). The SQLite single-container deployment and `better-sqlite3` native dependency have been removed. Current deployment: Docker + PostgreSQL 16 (dev), AWS App Runner + RDS (production).

Architecture Overview

Drop uses a **multi-stage Docker build** producing a minimal Node.js 22 Alpine production image. The application is a Next.js 16 standalone server.

Build stages (from `Dockerfile:1-41`):

Stage	Base	Purpose
`deps`	`node:22-alpine`	Install `node_modules` via `npm ci`.
`builder`	`node:22-alpine`	Copy deps + source, run `npm run build` (Next.js standalone output).
`runner`	`node:22-alpine`	Minimal production image. Copies only `public/`, `.next/standalone/`, `.next/static/`.

Security features in the runner stage (`Dockerfile:25-26`):

- Non-root user: `nextjs` (UID 1001, GID 1001)
- Data directory `/app/data` owned by `nextjs:nodejs`
- No build tools or source code in production image

Deployment Configurations

1. Local Development -- `docker-compose.yml`

PostgreSQL 16 + Drop app (ADR-014).

File: `src/drop-app/docker-compose.yml:1-22`

```
services:
  drop-app:
    build: .
    ports:
      - "3000:3000"
    environment:
      - JWT_SECRET=${JWT_SECRET:?JWT_SECRET is required}
      - NODE_ENV=production
      - NEXT_PUBLIC_SERVICE_MODE=mock
    volumes:
      - drop_data:/app/data
    healthcheck:
      test: ["CMD", "wget", "--no-verbose", "--tries=1", "--spider",
"http://localhost:3000/api/health"]
      interval: 30s
      timeout: 10s
      retries: 3
      start_period: 10s
    restart: unless-stopped
```

Quick start:

```
export JWT_SECRET="your-secure-random-string-min-32-chars"
docker compose up -d
```

Data persistence: PostgreSQL data stored in Docker volume `drop_pgdata`.

2. Production (PostgreSQL) -- `docker-compose.production.yml`

Multi-container setup with separate PostgreSQL 16 database.

File: `src/drop-app/docker-compose.production.yml:1-38`

```
services:
  drop-app:
    build: .
    ports:
      - "3000:3000"
    depends_on:
      postgres:
        condition: service_healthy
    restart: unless-stopped

  postgres:
    image: postgres:16-alpine
    environment:
      - POSTGRES_DB=drop
      - POSTGRES_USER=drop
      - POSTGRES_PASSWORD=${POSTGRES_PASSWORD:-drop_local_dev}
    volumes:
      - postgres_data:/var/lib/postgresql/data
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U drop"]
      interval: 10s
      timeout: 5s
      retries: 5
```

Quick start:

```
export JWT_SECRET="your-secure-random-string-min-32-chars"
export POSTGRES_PASSWORD="secure-postgres-password"
docker compose -f docker-compose.production.yml up -d
```

3. Fly.io Staging -- `fly.toml`

File: `src/drop-app/fly.toml:1-28`

Setting	Value
App name	`drop-staging`
Region	`arn` (Stockholm -- closest to Norway)
Internal port	3000
Force HTTPS	`true`

Setting	Value
Auto-stop machines	`stop` (scales to zero)
Auto-start machines	`true`
Min machines	0
Persistent storage	Volume `drop_data` mounted at `/app/data`

Health check: `GET /api/health` every 30s, 5s timeout, 10s grace period.

Environment Variables

Variable	Required	Default	Description
`JWT_SECRET`	**Yes** (production)	Dev: `process.cwd()` hash	JWT signing secret. Minimum 32 characters. Fatal error if missing in production.
`NODE_ENV`	No	`development`	Set to `production` in containers. Controls seed data gating.
`NEXT_PUBLIC_SERVICE_MODE`	No	-	Set to `mock` for MVP mode (no external API calls).
`DATABASE_URL`	**Yes**	-	PostgreSQL 16 connection string. Required in all environments. Local dev: `postgresql://drop:dev_only_not_a_secret@localhost:5433/drop_dev`
`POSTGRES_PASSWORD`	Production only	`drop_local_dev`	PostgreSQL password (production compose).
`PORT`	No	`3000`	HTTP server port.
`HOSTNAME`	No	`0.0.0.0`	Server bind address.

Database: PostgreSQL 16 is required in all environments. There is no SQLite fallback (ADR-014).

Health Check

Endpoint: `GET /api/health` **Source:** `src/drop-app/src/app/api/health/route.ts:1-35`

The health check performs a real database query (`SELECT 1 as ok`) and reports latency.

Success response (200):

```
{
  "status": "ok",
  "version": "0.1.0",
  "uptime": 123,
  "db": "connected",
  "dbLatencyMs": 5,
  "timestamp": "2026-02-13T12:00:00.000Z"
}
```

Failure response (503):

```
{
  "status": "error",
  "db": "disconnected",
  "timestamp": "..."
}
```

Building from Source

```
# Build Docker image
docker build -t drop-app .

# Run standalone container
docker run -d \
  -p 3000:3000 \
  -e JWT_SECRET="your-secret-min-32-chars" \
  -v drop_data:/app/data \
  --name drop-app \
  drop-app
```

Data Backup and Restore

Production Backups (AWS RDS)

Production database is PostgreSQL 16 on AWS RDS. Backups are managed by AWS:

- **Automated backups:** Daily snapshots, 7-day retention (configured in RDS)
- **Point-in-time recovery:** Available within the 7-day retention window
- **Manual snapshot:** Via AWS Console or CLI before major deployments

Create a manual RDS snapshot before deployments:

```
aws rds create-db-snapshot \
  --db-instance-identifier drop-production \
  --db-snapshot-identifier drop-pre-deploy-$(date +%Y%m%d-%H%M%S)
```

Restore from snapshot: Via AWS Console → RDS → Snapshots → Restore.

Local Dev Backups (Docker)

Local development data in the `drop_pgdata` Docker volume is disposable. Recreate with:

```
docker compose down -v   # Remove volume (deletes local data)
docker compose up -d
make db-push && npm run db:seed
```

Backup Verification

Verify production database connectivity and integrity:

```
# Check health endpoint
curl https://your-app-runner-url/api/health

# Connect to RDS (requires VPN or bastion)
psql $DATABASE_URL -c "SELECT COUNT(*) FROM users;"
```

Demo User

In non-production mode (`NODE_ENV !== 'production'`), a demo user is seeded:

Field	Value
Email	`amir@example.com`
Password	`demo1234`

Field	Value
Role	merchant

Source: Drizzle seed script in `src/shared/db/seed.ts`. Gated behind `NODE_ENV !== 'production'`.

Troubleshooting

Container won't start:

```
docker compose logs
docker compose exec drop-app env | grep JWT_SECRET
```

Database connection issues:

```
# Check PostgreSQL container is running
docker compose ps

# Test connection
docker compose exec db psql -U drop -d drop_dev -c "SELECT COUNT(*) FROM users;"

# Check app DATABASE_URL is set correctly
docker compose exec drop-app env | grep DATABASE_URL
```

Permission denied:

```
docker compose down -v   # Remove volumes
docker compose up -d     # Recreate with correct permissions
```

Cleanup:

```
docker compose down      # Stop containers
docker compose down -v   # Stop + remove volumes (WARNING: deletes data)
docker rmi drop-app      # Remove image
```