**WakePour Production Deployment Guide**

**Overview**

This document outlines the planned production deployment strategy for WakePour, including database migration, infrastructure, and backup strategies.

**Current Development Setup**

* **Database**: SQLite (file-based, single user)
* **Server**: Node.js + Express
* **Frontend**: React + Vite
* **Email**: Gmail SMTP
* **Instance**: Development on local machine

**Planned Production Setup**

**Infrastructure**

* **Server**: AWS EC2 t3.small (2 vCPU, 2GB RAM) - ~$15-17/month
* **Database**: Self-managed PostgreSQL on same EC2 instance
* **Domain**: Custom domain with SSL certificate
* **Storage**: EBS volumes for data and backups

**Database Migration Strategy**

**Development vs Production**

* **Development**: Continue using SQLite (no changes to current workflow)
* **Production**: PostgreSQL for better concurrency and reliability

**Database Configuration (knexfile.js)**

export default {

development: {

client: 'sqlite3',

connection: { filename: './data/database.sqlite3' },

useNullAsDefault: true

},

production: {

client: 'postgresql',

connection: {

host: process.env.DB\_HOST || 'localhost',

port: process.env.DB\_PORT || 5432,

user: process.env.DB\_USER,

password: process.env.DB\_PASSWORD,

database: process.env.DB\_NAME,

ssl: process.env.NODE\_ENV === 'production' ? { rejectUnauthorized: false } : false

},

pool: { min: 2, max: 10 },

migrations: {

tableName: 'knex\_migrations',

directory: './migrations'

}

}

};

**Required Production Environment Variables**

# Application

NODE\_ENV=production

JWT\_SECRET=your-production-jwt-secret-64-chars-minimum

FRONTEND\_URL=https://wakepour.com

# Database

DB\_HOST=localhost

DB\_PORT=5432

DB\_NAME=wakepour\_production

DB\_USER=wakepour\_user

DB\_PASSWORD=your-secure-database-password

# Email (same as development)

EMAIL\_HOST=smtp.gmail.com

EMAIL\_PORT=587

EMAIL\_USER=your-wakepour-gmail@gmail.com

EMAIL\_PASS=your-app-password

EMAIL\_FROM\_NAME=WakePour

EMAIL\_FROM\_ADDRESS=your-wakepour-gmail@gmail.com

ADMIN\_EMAIL=your-admin-email@gmail.com

**Backup Strategy**

**Three-Tier Backup Approach**

**Tier 1: Daily Automated Backups (On-Server)**

#!/bin/bash

# /home/ec2-user/scripts/daily-backup.sh

DATE=$(date +%Y%m%d\_%H%M%S)

BACKUP\_DIR="/var/backups/wakepour"

DB\_NAME="wakepour\_production"

# Create backup directory if it doesn't exist

mkdir -p $BACKUP\_DIR

# Create PostgreSQL dump

pg\_dump $DB\_NAME | gzip > $BACKUP\_DIR/wakepour\_backup\_$DATE.sql.gz

# Keep only last 7 days of local backups

find $BACKUP\_DIR -name "wakepour\_backup\_\*.sql.gz" -mtime +7 -delete

# Copy to S3 (if configured)

if [ ! -z "$AWS\_S3\_BUCKET" ]; then

aws s3 cp $BACKUP\_DIR/wakepour\_backup\_$DATE.sql.gz s3://$AWS\_S3\_BUCKET/daily/

fi

**Cron job**: 0 2 \* \* \* /home/ec2-user/scripts/daily-backup.sh

**Tier 2: Cloud Storage (S3)**

* **Purpose**: Off-site daily backups
* **Retention**: 30 days of daily backups
* **Cost**: ~$1-2/month for storage
* **Setup**: AWS CLI + S3 bucket

**Tier 3: Local PC Weekly Backups**

* **Purpose**: Long-term retention + ultimate safety
* **Method**: SCP/rsync weekly download
* **Retention**: Keep monthly backups indefinitely
* **Script Example**:

# Run weekly from local PC

scp ec2-user@wakepour.com:/var/backups/wakepour/latest.sql.gz ./backups/wakepour\_$(date +%Y%m%d).sql.gz

**Pre-Deployment Checklist**

**Domain & SSL Setup**

* [ ] Domain name registered and DNS configured
* [ ] SSL certificate obtained (Let's Encrypt or CloudFlare)
* [ ] Email verification functionality ready to enable
* [ ] Password reset functionality ready to enable

**Database Preparation**

* [ ] Update knexfile.js with production configuration
* [ ] Test migrations on local PostgreSQL instance
* [ ] Prepare database user creation scripts
* [ ] Set up backup scripts and test restore process

**Security Configuration**

* [ ] Generate strong JWT\_SECRET for production
* [ ] Configure PostgreSQL authentication
* [ ] Set up firewall rules (SSH, HTTP, HTTPS only)
* [ ] Review all environment variables

**Monitoring & Maintenance**

* [ ] Set up log rotation
* [ ] Configure PostgreSQL logging
* [ ] Plan for application monitoring
* [ ] Document server maintenance procedures

**Migration Steps (When Ready)**

**1. Server Setup**

1. Launch t3.small EC2 instance (Amazon Linux 2)
2. Install Node.js, PostgreSQL, nginx
3. Configure security groups and firewall
4. Set up domain DNS records

**2. Database Setup**

1. Install and configure PostgreSQL
2. Create production database and user
3. Set up authentication and security
4. Test database connectivity

**3. Application Deployment**

1. Clone repository to server
2. Set up production environment variables
3. Install dependencies (npm install)
4. Run database migrations
5. Build frontend (npm run build)
6. Configure nginx reverse proxy
7. Set up SSL certificate

**4. Backup Configuration**

1. Set up backup scripts and cron jobs
2. Configure S3 bucket and AWS CLI
3. Test backup and restore procedures
4. Set up local PC backup routine

**5. Testing & Go-Live**

1. Thorough testing of all functionality
2. Test email sending with production domain
3. Verify SSL certificate and security
4. Monitor for 24-48 hours before announcing

**Cost Estimates (Monthly)**

* **EC2 t3.small**: $15-17
* **EBS storage**: $2-3 (for 20GB)
* **S3 backups**: $1-2
* **Domain**: $1-2 (annual cost divided by 12)
* **Total**: ~$20-25/month

**Current Status: Development Complete ✅**

* User registration and approval workflow
* Email notifications (Gmail SMTP)
* Profile management with phone formatting
* Admin panel for user management
* Clean database schema and API structure
* Professional UI/UX with WakePour branding

**Next Phase: Core Bourbon Tracking Features**

Before production deployment, plan to implement:

* Bourbon inventory tracking
* Store location management
* User notifications/alerts
* Search and filtering capabilities

**Last Updated**: [Current Date]  
**Status**: Planning Phase - Ready for Production Deployment when needed