Deployment Guide

This guide covers different deployment options for the CapitalFlow Portal.

🚀 Deployment Options

1. Vercel (Recommended)

Vercel provides the best experience for Next.js applications with automatic deployments and serverless functions.

Prerequisites

- GitHub account
- Vercel account
- PostgreSQL database (Vercel Postgres or external)

Steps

1. **Prepare your repository**
   ```
   bash
      git add .
      git commit -m "feat: prepare for deployment"
      git push origin main
   ```

2. **Connect to Vercel**
   - Visit vercel.com (https://vercel.com)
   - Import your GitHub repository
   - Configure project settings

3. **Environment Variables**
   Add these environment variables in Vercel dashboard:
   ```
   DATABASE_URL=postgresql://username:password@host:5432/database
      NEXTAUTH_URL=https://your-domain.vercel.app
      NEXTAUTH_SECRET=your-secret-key
   ```

4. **Deploy**
   - Vercel will automatically build and deploy
   - Run database migrations in Vercel dashboard or via CLI

Vercel CLI Deployment

```
# Install Vercel CLI
npm install -g vercel

# Deploy
vercel --prod
```

2. Docker Deployment

Deploy using Docker containers for full control over the environment.

Prerequisites

- Docker and Docker Compose installed
- PostgreSQL database

Steps

1. **Build and run with Docker Compose**

```bash
# For production
docker-compose up -d

# For development
docker-compose -f docker-compose.dev.yml up -d
```

1. **Environment Configuration**
   Update `docker-compose.yml` with your environment variables:

```yaml
  environment:
    DATABASE_URL: "postgresql://user:password@db:5432/capitalflow"
    NEXTAUTH_URL: "https://your-domain.com"
    NEXTAUTH_SECRET: "your-secret-key"
```

2. **Database Setup**

```bash
# Access the app container
docker exec -it capitalflow_app bash

# Run migrations
npx prisma db push
npx prisma db seed
```

3. Manual Server Deployment

Deploy on your own server with PM2 or similar process manager.

Prerequisites

- Node.js 18+
- PostgreSQL database
- PM2 (optional but recommended)

Steps

1. **Server Setup**

```bash
# Update system
sudo apt update && sudo apt upgrade -y

# Install Node.js
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt-get install -y nodejs
```

```
# Install PM2
sudo npm install -g pm2
```

1. **Deploy Application**

```bash
# Clone repository
git clone https://github.com/your-username/capitalflow-portal.git
cd capitalflow-portal/app

# Install dependencies
yarn install

# Set up environment
cp .env.example .env
# Edit .env with your configuration

# Build application
yarn build

# Start with PM2
pm2 start npm –name “capitalflow” – start
pm2 save
pm2 startup
```

1. **Nginx Configuration**

```nginx
server {
listen 80;
server_name your-domain.com;

location / {
proxy_pass http://localhost:3000;
proxy_http_version 1.1;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection ‘upgrade’;
proxy_set_header Host $host;
proxy_cache_bypass $http_upgrade;
}
}
```

4. AWS Deployment

Deploy on AWS using various services.

Option A: AWS App Runner

1. **Create App Runner Service**
   - Connect to your GitHub repository
   - Configure build settings
   - Set environment variables

2. **Build Configuration**
   ```yaml
   # apprunner.yaml
   version: 1.0
   runtime: nodejs18
   build:
     commands:
       build:
         - cd app
         - npm install
         - npm run build
   run:
     runtime-version: 18
     command: cd app && npm start
     network:
       port: 3000
   ```

Option B: AWS ECS with Fargate

1. **Create ECS Cluster**
2. **Build and push Docker image to ECR**
3. **Create ECS Service with Fargate**
4. **Configure Load Balancer**

5. Railway Deployment

Railway provides an easy deployment experience with automatic HTTPS.

Steps

1. **Connect Repository**
   - Visit railway.app (https://railway.app)
   - Connect your GitHub repository

2. **Add Database**
   - Add PostgreSQL plugin
   - Copy connection string

3. **Environment Variables**
   ```
   DATABASE_URL=${{Postgres.DATABASE_URL}}
   NEXTAUTH_URL=${{RAILWAY_STATIC_URL}}
   NEXTAUTH_SECRET=your-secret-key
   ```

4. **Deploy**
   - Railway will automatically deploy on git push

🔧 Production Configuration

Database Setup

PostgreSQL Configuration

```sql
-- Create database
CREATE DATABASE capitalflow;

-- Create user
CREATE USER capitalflow_user WITH PASSWORD 'secure_password';

-- Grant permissions
GRANT ALL PRIVILEGES ON DATABASE capitalflow TO capitalflow_user;
```

Connection Pooling

For production, use connection pooling:

```
# Example with PgBouncer
DATABASE_URL="postgresql://user:password@pgbouncer:5432/capitalflow"
```

Security Considerations

1. **Environment Variables**
   - Use strong, unique secrets
   - Never commit secrets to version control
   - Use environment-specific configurations

2. **Database Security**
   - Use SSL connections
   - Implement proper user permissions
   - Regular security updates

3. **Application Security**
   - Enable HTTPS
   - Configure security headers
   - Implement rate limiting

Performance Optimization

1. **Database Optimization**

```
sql
   -- Add indexes for frequently queried columns
   CREATE INDEX idx_user_email ON users(email);
   CREATE INDEX idx_transaction_date ON transactions(transaction_date);
```

2. **Application Optimization**

```
javascript
   // Enable compression
   const nextConfig = {
     compress: true,
     images: {
       domains: ['your-domain.com'],
    },
  };
```

3. **CDN Configuration**
   - Use CDN for static assets
   - Configure proper cache headers
   - Optimize images

Monitoring and Logging

1. **Application Monitoring**
   ```bash
   # PM2 monitoring
   pm2 monit

# View logs
pm2 logs capitalflow
```

1. **Database Monitoring**
   ```sql
   – Monitor active connections
   SELECT * FROM pg_stat_activity;

– Check query performance
SELECT * FROM pg_stat_statements;
```

Backup Strategy

1. **Database Backups**

```
bash
  # Daily backup script
  #!/bin/bash
  pg_dump -h localhost -U capitalflow_user capitalflow > backup_$(date +%Y%m%d).sql
```

2. **Application Backups**

```
bash
  # Backup uploaded files
  tar -czf files_backup_$(date +%Y%m%d).tar.gz /app/uploads/
```

🚨 Troubleshooting

Common Issues

1. **Database Connection Issues**
   ```bash
   # Check connection
   psql -h hostname -U username -d database

# Test from application
npx prisma db pull
```

1. **Build Failures**
   ```bash
   ```

    # Clear cache
    rm -rf .next
    yarn build

# Check dependencies
yarn install –frozen-lockfile

1. **Performance Issues**
   ```bash
   # Check memory usage
   free -h

# Check disk space
df -h

# Monitor processes
top
```

Health Checks

The application includes a health check endpoint:

```
GET /api/health
```

Response:

```
{
  "status": "healthy",
  "timestamp": "2024-01-01T12:00:00Z",
  "database": "connected",
  "version": "1.0.0"
}
```

📚 Additional Resources

- Next.js Deployment Documentation (https://nextjs.org/docs/deployment)
- Vercel Deployment Guide (https://vercel.com/docs)
- Docker Best Practices (https://docs.docker.com/develop/dev-best-practices/)
- PostgreSQL Performance Tuning (https://wiki.postgresql.org/wiki/Performance_Optimization)

For more specific deployment questions, please refer to the CONTRIBUTING.md (../CONTRIBUTING.md) file or open an issue.