İSG Denetim Sistemi - Deployment Dokümantasyonu

📋 İçindekiler

🎯 Genel Bakış

Bu dokümantasyon, İSG Denetim Sistemi'nin production ortamına deployment sürecini açıklar. Sistem iki ana bileşenden oluşur:

- **Backend**: NestJS tabanlı REST API (Port: 3000)
- **Frontend**: React + Vite tabanlı SPA (Port: 5173)

Sistem Gereksinimleri

- Node.js v18+
- npm v9+
- PostgreSQL 14+
- Systemd (Linux)
- Git (opsiyonel ama önerilen)
- sudo yetkileri

Proje Yapısı

```
/home/ubuntu/isg_denetim_sistemi/
├── backend/                    # Backend uygulaması
│   ├── src/                    # Kaynak kodlar
│   ├── prisma/                 # Database schema
│   ├── dist/                   # Build çıktıları
│   └── package.json
├── frontend/                   # Frontend uygulaması
│   ├── src/                    # Kaynak kodlar
│   ├── dist/                   # Build çıktıları
│   └── package.json
├── systemd/                    # Systemd servis dosyaları
│   ├── isg-backend.service
│   └── isg-frontend.service
├── deploy.sh                   # Otomatik deployment script'i
└── DEPLOYMENT.md               # Bu dosya
```

🚀 Otomatik Deployment

1. Deployment Script'ini Kullanma

Otomatik deployment için `deploy.sh` script'ini kullanabilirsiniz. Bu script tüm deployment adımlarını otomatik olarak gerçekleştirir.

Hızlı Başlangıç

```
cd /home/ubuntu/isg_denetim_sistemi
sudo ./deploy.sh
```

> **Not**: Script sudo yetkisi gerektirir çünkü systemd servislerini yönetir.

2. Script'in Yaptığı İşlemler

Script şu adımları sırayla gerçekleştirir:

Adım 1: Proje Dizinine Geçiş

- Proje dizininin varlığını kontrol eder
- Çalışma dizinini proje dizinine değiştirir

Adım 2: Git Repository Kontrolü

- Git repository olup olmadığını kontrol eder
- Varsa `git pull` ile son değişiklikleri çeker
- Yoksa uyarı verir ve devam eder

Adım 3: Backend Dependencies Kontrolü

- `package.json` dosyasında değişiklik olup olmadığını kontrol eder
- Değişiklik varsa veya `node_modules` yoksa `npm install --legacy-peer-deps` çalıştırır
- Değişiklik yoksa bu adımı atlar (performans için)

Adım 4: Frontend Dependencies Kontrolü

- Backend ile aynı mantıkla çalışır
- `package.json` hash'ini kontrol eder
- Gerekirse dependencies yükler

Adım 5: Prisma Database Migration

- Prisma schema değişikliklerini database'e uygular
- `npx prisma db push` komutunu çalıştırır
- Yeni tablolar veya alanlar varsa ekler

Adım 6: Backend Build

- `npm run build` ile backend'i derler
- TypeScript kodlarını JavaScript'e çevirir
- Çıktılar `backend/dist/` dizinine yazılır

Adım 7: Frontend Build

- `npm run build` ile frontend'i derler
- React kodlarını optimize eder
- Çıktılar `frontend/dist/` dizinine yazılır

Adım 8: Systemd Servisleri Restart

- Backend servisini yeniden başlatır (`isg-backend.service`)
- Frontend servisini yeniden başlatır (`isg-frontend.service`)
- Servislerin çalışıp çalışmadığını kontrol eder

Adım 9: Health Check

- Backend'in port 3000'de yanıt verip vermediğini kontrol eder
- Frontend'in port 5173'te yanıt verip vermediğini kontrol eder
- Maksimum 10 deneme yapar (20 saniye timeout)

3. Script Çıktıları

Script çalışırken renkli çıktılar verir:

- 🔵 **Mavi (i)**: Bilgi mesajları
- 🟢 **Yeşil (✓)**: Başarılı işlemler
- 🟡 **Sarı (⚠)**: Uyarılar
- 🔴 **Kırmızı (✗)**: Hatalar

4. Log Dosyaları

Tüm deployment işlemleri log dosyasına kaydedilir:

```
# Deployment logları
tail -f /var/log/isg-deployment.log

# Backend servis logları
tail -f /var/log/isg-backend.log
tail -f /var/log/isg-backend-error.log

# Frontend servis logları
tail -f /var/log/isg-frontend.log
tail -f /var/log/isg-frontend-error.log
```

5. Script Örnek Çıktısı

```
╔═══════════════════════════════════════════════════════════╗
║          İSG Denetim Sistemi - Otomatik Deployment        ║
╚═══════════════════════════════════════════════════════════╝

ℹ Proje Dizini: /home/ubuntu/isg_denetim_sistemi
ℹ Log Dosyası: /var/log/isg-deployment.log

═══════════════════════════════════════════════════════════
▶ 1/9 - Proje Dizinine Geçiliyor
═══════════════════════════════════════════════════════════

✓ Proje dizinine geçildi

═══════════════════════════════════════════════════════════
▶ 2/9 - Git Repository Kontrolü
═══════════════════════════════════════════════════════════

ℹ Git repository bulundu, güncelleme çekiliyor...
✓ Git güncelleme başarılı

[... diğer adımlar ...]

╔═══════════════════════════════════════════════════════════╗
║              🎉 DEPLOYMENT BAŞARIYLA TAMAMLANDI! 🎉       ║
╚═══════════════════════════════════════════════════════════╝

✓ Deployment süresi: 45 saniye
✓ Backend: http://localhost:3000
✓ Frontend: http://localhost:5173
✓ Log dosyası: /var/log/isg-deployment.log
```

🛠️ Manuel Deployment

Script çalışmadığında veya özel durumlar için manuel deployment yapabilirsiniz.

1. Git Güncelleme (Opsiyonel)

```
cd /home/ubuntu/isg_denetim_sistemi
git pull origin main
```

2. Backend Deployment

```
cd /home/ubuntu/isg_denetim_sistemi/backend

# Dependencies yükle
npm install --legacy-peer-deps

# Prisma migration
npx prisma db push

# Build
npm run build

# Servisi restart et
sudo systemctl restart isg-backend.service

# Servis durumunu kontrol et
sudo systemctl status isg-backend.service
```

3. Frontend Deployment

```
cd /home/ubuntu/isg_denetim_sistemi/frontend

# Dependencies yükle
npm install --legacy-peer-deps

# Build
npm run build

# Servisi restart et
sudo systemctl restart isg-frontend.service

# Servis durumunu kontrol et
sudo systemctl status isg-frontend.service
```

4. Servisleri Kontrol Etme

```
# Servis durumları
sudo systemctl status isg-backend.service
sudo systemctl status isg-frontend.service

# Servislerin logları
sudo journalctl -u isg-backend.service -f
sudo journalctl -u isg-frontend.service -f

# Port kontrolü
netstat -tulpn | grep :3000
netstat -tulpn | grep :5173

# HTTP kontrolü
curl http://localhost:3000
curl http://localhost:5173
```

🔄 Rollback İşlemleri

Deployment sonrası problem çıkarsa önceki versiyona dönebilirsiniz.

Git ile Rollback

1. Son Commit'e Dön

```
cd /home/ubuntu/isg_denetim_sistemi

# Son commit'i gör
git log -1

# Bir önceki commit'e dön
git reset --hard HEAD~1

# Veya belirli bir commit'e dön
git reset --hard <commit-hash>

# Sonra deploy script'ini çalıştır
sudo ./deploy.sh
```

2. Belirli Bir Branch/Tag'e Dön

```
# Branch'e dön
git checkout production-stable
sudo ./deploy.sh

# Tag'e dön
git checkout v1.0.0
sudo ./deploy.sh
```

Manuel Rollback (Git olmadan)

1. Backup'tan Dön

```
# Önceden backup aldıysanız
cd /home/ubuntu
rm -rf isg_denetim_sistemi
mv isg_denetim_sistemi.backup isg_denetim_sistemi

# Servisleri restart et
sudo systemctl restart isg-backend.service
sudo systemctl restart isg-frontend.service
```

2. Önceki Build'i Kullan

```
# Eğer dist dizinlerini yedeklediyseniz
cd /home/ubuntu/isg_denetim_sistemi/backend
rm -rf dist
mv dist.backup dist

cd /home/ubuntu/isg_denetim_sistemi/frontend
rm -rf dist
mv dist.backup dist

# Servisleri restart et
sudo systemctl restart isg-backend.service
sudo systemctl restart isg-frontend.service
```

Database Rollback

```
cd /home/ubuntu/isg_denetim_sistemi/backend

# Prisma migration geri al (dikkatli kullanın!)
npx prisma migrate resolve --rolled-back <migration-name>

# Veya database backup'tan restore et
# (PostgreSQL backup'ınız varsa)
```

🔍 Troubleshooting

Problem 1: Script Çalışmıyor

Hata: `Permission denied` veya `command not found`

Çözüm:

```
# Script'i executable yap
chmod +x /home/ubuntu/isg_denetim_sistemi/deploy.sh

# Sudo ile çalıştır
sudo /home/ubuntu/isg_denetim_sistemi/deploy.sh
```

Problem 2: npm install Başarısız

Hata: `npm ERR! ERESOLVE unable to resolve dependency tree`

Çözüm:

```
# --legacy-peer-deps kullan
npm install --legacy-peer-deps

# Veya node_modules'u sil ve tekrar dene
rm -rf node_modules package-lock.json
npm install --legacy-peer-deps
```

Problem 3: Prisma Migration Hatası

Hata: `prisma db push failed`

Çözüm:

```
# Prisma client'ı yeniden oluştur
npx prisma generate

# Database bağlantısını kontrol et
npx prisma db pull

# Migration'ı force et (dikkatli!)
npx prisma db push --force-reset
```

Problem 4: Build Hatası

Hata: `Build failed` veya `TypeScript errors`

Çözüm:

```
# TypeScript hatalarını kontrol et
npm run build

# node_modules'u temizle
rm -rf node_modules package-lock.json dist
npm install --legacy-peer-deps
npm run build
```

Problem 5: Servis Başlamıyor

Hata: `systemctl restart failed`

Çözüm:

```
# Servis durumunu kontrol et
sudo systemctl status isg-backend.service
sudo journalctl -u isg-backend.service -n 50

# Servis dosyasını kontrol et
sudo systemctl daemon-reload
sudo systemctl restart isg-backend.service

# Port kullanımda mı kontrol et
sudo lsof -i :3000
sudo lsof -i :5173

# Gerekirse process'i öldür
sudo kill -9 <PID>
```

Problem 6: Health Check Başarısız

Hata: `Health check failed`

Çözüm:

```
# Manuel port kontrolü
curl -v http://localhost:3000
curl -v http://localhost:5173

# Firewall kontrolü
sudo ufw status
sudo ufw allow 3000
sudo ufw allow 5173

# Logları kontrol et
tail -f /var/log/isg-backend.log
tail -f /var/log/isg-frontend.log
```

Problem 7: Database Bağlantı Hatası

Hata: `Can't reach database server`

Çözüm:

```
# PostgreSQL çalışıyor mu?
sudo systemctl status postgresql

# PostgreSQL'i başlat
sudo systemctl start postgresql

# Bağlantı bilgilerini kontrol et
cat /home/ubuntu/isg_denetim_sistemi/backend/.env

# PostgreSQL'e bağlan
psql -U admin -d postgres -h localhost -p 5432
```

Problem 8: Frontend CORS Hatası

Hata: `CORS policy blocked`

Çözüm:

```
# Backend .env dosyasını kontrol et
cd /home/ubuntu/isg_denetim_sistemi/backend
cat .env | grep CORS

# CORS ayarlarını düzenle
nano .env

# Örnek:
# CORS_ORIGIN=http://localhost:5173,http://77.42.22.226:5173

# Backend'i restart et
sudo systemctl restart isg-backend.service
```

Problem 9: Disk Dolu

Hata: `No space left on device`

Çözüm:

```
# Disk kullanımını kontrol et
df -h

# Büyük dosyaları bul
du -sh /home/ubuntu/isg_denetim_sistemi/*

# Log dosyalarını temizle
sudo truncate -s 0 /var/log/isg-backend.log
sudo truncate -s 0 /var/log/isg-frontend.log
sudo truncate -s 0 /var/log/isg-deployment.log

# npm cache'i temizle
npm cache clean --force

# Eski build'leri sil
cd /home/ubuntu/isg_denetim_sistemi/backend
rm -rf dist

cd /home/ubuntu/isg_denetim_sistemi/frontend
rm -rf dist
```

Problem 10: Port Çakışması

Hata: `Port 3000 is already in use`

Çözüm:

```
# Portu kullanan process'i bul
sudo lsof -i :3000
sudo lsof -i :5173

# Process'i öldür
sudo kill -9 <PID>

# Servisi restart et
sudo systemctl restart isg-backend.service
```

📝 Best Practices

1. Deployment Öncesi

✅ **Yapılması Gerekenler**:

- [ ] Kod değişikliklerini local'de test edin
- [ ] Git commit ve push yapın
- [ ] Database backup alın
- [ ] Proje dosyalarını yedekleyin
- [ ] Servislerin çalıştığını kontrol edin
- [ ] Deployment zamanını planlayın (düşük trafik saatlerinde)

```
# Backup alma örneği
cd /home/ubuntu
tar -czf isg_denetim_sistemi.backup.$(date +%Y%m%d_%H%M%S).tar.gz isg_denetim_sistemi/

# Database backup
pg_dump -U admin -d postgres > postgres_backup_$(date +%Y%m%d_%H%M%S).sql
```

2. Deployment Sırası

✅ **Önerilen Sıra**:

1. Kullanıcıları bilgilendirin (bakım modu)
2. Database backup alın
3. Deployment script'ini çalıştırın
4. Health check yapın
5. Manuel test edin
6. Kullanıcılara duyurun

3. Deployment Sonrası

✅ **Kontrol Listesi**:

- [ ] Servislerin çalıştığını kontrol edin
- [ ] API endpoint'lerini test edin
- [ ] Frontend'in açıldığını kontrol edin
- [ ] Login işlemini test edin
- [ ] Kritik fonksiyonları test edin
- [ ] Log dosyalarını kontrol edin
- [ ] Error loglarını inceleyin

```
# Hızlı health check
curl http://localhost:3000/health
curl http://localhost:5173

# Servis durumları
sudo systemctl status isg-backend.service
sudo systemctl status isg-frontend.service

# Son loglar
sudo tail -20 /var/log/isg-backend.log
sudo tail -20 /var/log/isg-frontend.log
```

4. Güvenlik

✅ **Güvenlik İpuçları**:

- `.env` dosyalarını git'e eklemeyin
- Hassas bilgileri loglamayın
- Production'da debug mode'u kapatın
- Düzenli security update yapın
- Firewall kurallarını doğru ayarlayın

```
# .env dosyasını koru
chmod 600 /home/ubuntu/isg_denetim_sistemi/backend/.env
chmod 600 /home/ubuntu/isg_denetim_sistemi/frontend/.env
```

5. Monitoring

✅ **İzleme Önerileri**:

- Log dosyalarını düzenli kontrol edin
- Disk kullanımını izleyin
- Memory kullanımını izleyin
- Response time'ları takip edin
- Error rate'leri izleyin

```
# Sistem kaynaklarını izle
htop

# Servis loglarını canlı izle
sudo journalctl -u isg-backend.service -f
sudo journalctl -u isg-frontend.service -f

# Disk kullanımı
df -h
du -sh /home/ubuntu/isg_denetim_sistemi/*
```

6. Git Workflow

✅ **Önerilen Git Workflow**:

```
# Development branch'te çalış
git checkout -b feature/yeni-ozellik

# Değişiklikleri commit et
git add .
git commit -m "feat: yeni özellik eklendi"

# Main branch'e merge et
git checkout main
git merge feature/yeni-ozellik

# Push et
git push origin main

# Production'a deploy et
sudo /home/ubuntu/isg_denetim_sistemi/deploy.sh
```

7. Automated Deployment

✅ **Cron ile Otomatik Deployment** (Dikkatli kullanın!):

```
# Crontab düzenle
crontab -e

# Her gece 3'te deployment (önermiyoruz, sadece örnek)
0 3 * * * cd /home/ubuntu/isg_denetim_sistemi && sudo ./deploy.sh >> /var/log/cron-deployment.log 2>&1
```

> **Uyarı**: Otomatik deployment'ı production'da kullanırken çok dikkatli olun!

❓ Sık Sorulan Sorular

S1: Deployment ne kadar sürer?

C: Normal şartlarda 30-60 saniye arası. İlk deployment veya büyük değişikliklerde 2-3 dakika sürebilir.

S2: Deployment sırasında sistem erişilebilir mi?

C: Hayır, servisler restart edildiği için kısa bir downtime olur (5-10 saniye).

S3: Her deployment'ta npm install gerekir mi?

C: Hayır, script `package.json` değişikliğini kontrol eder. Değişiklik yoksa npm install atlanır.

S4: Prisma migration ne yapar?

C: Database schema değişikliklerini uygular. Yeni tablolar, alanlar veya ilişkiler ekler.

S5: Rollback nasıl yapılır?

C: Git ile önceki commit'e dönüp deploy script'ini tekrar çalıştırın. Detaylar için Rollback İşlemleri bölümüne bakın.

S6: Log dosyaları çok büyüyorsa ne yapmalı?

C: Log rotation yapın veya manuel olarak temizleyin:

```
# Log dosyalarını sıfırla
sudo truncate -s 0 /var/log/isg-*.log

# Veya logrotate kullanın
sudo logrotate -f /etc/logrotate.conf
```

S7: Development ortamında da aynı script kullanılır mı?

C: Hayır, development'ta `npm run dev` kullanın. Bu script production içindir.

```
# Development - Backend
cd /home/ubuntu/isg_denetim_sistemi/backend
npm run start:dev

# Development - Frontend
cd /home/ubuntu/isg_denetim_sistemi/frontend
npm run dev
```

S8: Script'i webhook ile otomatik çalıştırabilir miyim?

C: Evet, GitHub webhooks veya GitLab CI/CD kullanabilirsiniz. Ancak güvenlik için dikkatli olun.

S9: Production'da hangi environment kullanılıyor?

C: Backend `NODE_ENV=production` kullanıyor. `.env` dosyasında ayarlanmış.

S10: Backup ne sıklıkla alınmalı?

C: Her deployment öncesi. Ayrıca günlük otomatik backup önerilir.

📞 Destek

Sorun yaşarsanız:

1. Log dosyalarını kontrol edin
2. Troubleshooting bölümüne bakın
3. Sistem yöneticinize ulaşın

📚 Ek Kaynaklar

- NestJS Deployment (https://docs.nestjs.com/deployment)
- Vite Production Build (https://vitejs.dev/guide/build.html)
- Prisma Deployment (https://www.prisma.io/docs/guides/deployment)
- Systemd Service (https://www.freedesktop.org/software/systemd/man/systemd.service.html)

Son Güncelleme: 1 Aralık 2025
Versiyon: 1.0.0
Yazar: İSG Denetim Sistemi Ekibi