Guía de Despliegue y Rollback - INMOVA

🚀 Proceso de Despliegue

Pre-Despliegue

1. **Verificar Tests**
   ```bash
   cd nextjs_space
   yarn test
   ```

2. **Build Local**
   ```bash
   yarn build
   ```

3. **Verificar Migraciones de Base de Datos**
   ```bash
   yarn prisma migrate status
   yarn prisma migrate deploy # Si hay migraciones pendientes
   ```

Despliegue Producción

1. **Crear Tag de Versión**
   ```bash
   git tag -a v1.0.x -m "Release v1.0.x: Descripción de cambios"
   git push origin v1.0.x
   ```

2. **Deploy**
   - El pipeline de GitHub Actions se ejecutará automáticamente
   - Monitorear logs en GitHub Actions
   - Verificar health check: `https://inmova.app/api/health`

3. **Post-Despliegue**
   - Verificar funcionamiento en producción
   - Monitorear logs de errores
   - Verificar métricas de rendimiento

🔄 Estrategia de Rollback

Nivel 1: Rollback Rápido (Frontend)

Cuando usar: Errores de UI, bugs visuales, problemas de JavaScript.

```
# Revertir a la última versión estable
git revert <commit-hash>
git push origin main

# O desplegar tag anterior
git checkout v1.0.x-1
git push origin main --force
```

Tiempo estimado: 2-5 minutos

Nivel 2: Rollback con Base de Datos

Cuando usar: Cambios en schema de BD, migraciones problemáticas.

Paso 1: Revertir Código

```
git revert <commit-hash>
git push origin main
```

Paso 2: Revertir Migraciones

```
# Listar migraciones
yarn prisma migrate status

# Revertir última migración (NO RECOMENDADO EN PRODUCCIÓN)
# yarn prisma migrate reset

# MEJOR: Crear migración de reversión
yarn prisma migrate dev --name rollback_feature_x
```

⚠️ **IMPORTANTE:**
- **NUNCA** usar `prisma migrate reset` en producción
- Siempre crear migraciones “forward” para revertir cambios
- Hacer backup de BD antes de cualquier revert

Tiempo estimado: 10-20 minutos

Nivel 3: Rollback Completo (Disaster Recovery)

Cuando usar: Corrupción de datos, pérdida de servicio crítica.

Paso 1: Backup de Base de Datos

```
# Crear backup antes de cualquier cambio
pg_dump $DATABASE_URL > backup_$(date +%Y%m%d_%H%M%S).sql
```

Paso 2: Restaurar Versión Completa

```
# 1. Revertir código a tag estable
git checkout v1.0.x-stable
git push origin main --force

# 2. Restaurar base de datos desde backup
psql $DATABASE_URL < backup_YYYYMMDD_HHMMSS.sql

# 3. Verificar integridad
yarn prisma migrate status
yarn prisma generate
```

Tiempo estimado: 30-60 minutos

📄 Checklist de Rollback

Antes del Rollback

- [ ] Identificar la causa raíz del problema
- [ ] Determinar el nivel de rollback necesario
- [ ] Notificar al equipo del rollback inminente
- [ ] Crear backup de base de datos actual
- [ ] Documentar el estado actual del sistema

Durante el Rollback

- [ ] Ejecutar rollback según el nivel correspondiente
- [ ] Monitorear logs y errores
- [ ] Verificar health check endpoint
- [ ] Probar funcionalidad crítica

Después del Rollback

- [ ] Verificar que el sistema funciona correctamente
- [ ] Documentar el incidente y causa
- [ ] Crear plan de corrección
- [ ] Actualizar documentación y runbooks
- [ ] Comunicar resolución al equipo

📊 Monitoreo Post-Rollback

Health Checks

```
# Verificar salud del sistema
curl -H "Authorization: Bearer $CRON_SECRET" https://inmova.app/api/health
```

Métricas Clave

- [ ] Tiempo de respuesta de APIs < 500ms

- [ ] Tasa de errores < 1%
- [ ] Conexiones de BD estables
- [ ] Memoria < 80% uso

Logs a Revisar

```
# Logs de aplicación
tail -f /var/log/inmova/app.log

# Logs de Prisma
tail -f /var/log/inmova/prisma.log

# Logs de Next.js
tail -f /var/log/inmova/nextjs.log
```

📞 Contactos de Emergencia

Equipo Técnico

- **DevOps Lead:** [Nombre] - [Email/Teléfono]
- **Backend Lead:** [Nombre] - [Email/Teléfono]
- **DBA:** [Nombre] - [Email/Teléfono]

Escalación

1. **Nivel 1 (5 min):** Equipo de desarrollo
2. **Nivel 2 (15 min):** Tech Lead
3. **Nivel 3 (30 min):** CTO/Director Técnico

📝 Plantilla de Incident Report

```
# Incident Report: [YYYY-MM-DD]

## Resumen
- **Fecha/Hora:**
- **Duración:**
- **Impacto:**
- **Severidad:** P1 / P2 / P3 / P4

## Descripción
[Qué sucedió]

## Causa Raíz
[Por qué sucedió]

## Acción Tomada
[Cómo se resolvió]

## Rollback Ejecutado
- **Tipo:** Nivel 1 / 2 / 3
- **Versión anterior:** v1.0.x
- **Resultado:** Éxito / Parcial / Fallo

## Prevención Futura
[Cómo evitar que vuelva a suceder]

## Action Items
- [ ] Item 1
- [ ] Item 2

## Timeline
| Hora | Evento |
|------|--------|
| 14:00 | Detección del problema |
| 14:05 | Inicio de investigación |
| 14:15 | Decisión de rollback |
| 14:20 | Rollback ejecutado |
| 14:30 | Verificación completada |
```

🔒 Backups Automáticos

Configuración de Backups

Frecuencia:
- **Diario:** 02:00 AM (retención: 7 días)
- **Semanal:** Domingo 03:00 AM (retención: 4 semanas)
- **Mensual:** Día 1 a las 04:00 AM (retención: 12 meses)

Ubicación: S3 / Cloud Storage

Verificación:

```
# Listar backups disponibles
ls -lh /backups/database/

# Verificar integridad
pg_restore --list backup_file.sql
```

✅ Testing de Rollback

Simulacro Trimestral

1. **Preparación** (15 min)
   - Crear entorno de staging idéntico a producción
   - Notificar al equipo del simulacro

2. **Ejecución** (30 min)
   - Ejecutar rollback completo en staging
   - Documentar tiempos y problemas

3. **Revisión** (15 min)
   - Analizar resultados
   - Actualizar procedimientos
   - Entrenar al equipo

Próximo Simulacro: [Fecha]

📖 Referencias

- Next.js Deployment Docs (https://nextjs.org/docs/deployment)
- Prisma Migration Docs (https://www.prisma.io/docs/concepts/components/prisma-migrate)
- PostgreSQL Backup & Recovery (https://www.postgresql.org/docs/current/backup.html)
- Git Revert vs Reset (https://git-scm.com/docs/git-revert)

Última actualización: Diciembre 2024
Versión: 1.0
Responsable: Equipo DevOps INMOVA