Deployment Guide

Overview

This guide covers deployment options for the AI RFP Risk Scanner application across different environments.

Prerequisites

- Node.js 18+ and yarn
- PostgreSQL database (local or cloud)
- Domain name (for production)
- SSL certificate (for production)

Development Deployment

Local Development

```
# 1. Clone and setup
git clone <repository-url>
cd ai_rfp_risk_scanner/app
yarn install

# 2. Configure environment
cp ../.env.example .env.local
# Edit .env.local with your configuration

# 3. Setup database
npx prisma generate
npx prisma db push
npm run seed

# 4. Start development server
yarn dev
```

Access at: `http://localhost:3000`

Production Deployments

1. Vercel Deployment (Recommended)

Prerequisites

- Vercel account
- GitHub repository
- PostgreSQL database (Supabase, PlanetScale, or AWS RDS recommended)

Steps

1. **Connect Repository**
   - Connect your GitHub repo to Vercel
   - Import the project

2. **Configure Environment Variables**
   `env`
   `DATABASE_URL=postgresql://user:pass@host:5432/db`
   `NEXTAUTH_URL=https://your-domain.vercel.app`
   `NEXTAUTH_SECRET=your-production-secret`
   `ABACUSAI_API_KEY=your-api-key`

3. **Build Settings**
   - Build Command: `cd app && yarn build`
   - Output Directory: `app/.next`
   - Install Command: `cd app && yarn install`

4. **Deploy**
   - Vercel will automatically deploy on push to main branch

2. Docker Deployment

Dockerfile

```
FROM node:18-alpine

WORKDIR /app

# Copy package files
COPY app/package*.json ./
COPY app/yarn.lock ./

# Install dependencies
RUN yarn install --frozen-lockfile

# Copy source code
COPY app/ ./

# Generate Prisma client
RUN npx prisma generate

# Build application
RUN yarn build

# Expose port
EXPOSE 3000

# Start application
CMD ["yarn", "start"]
```

Docker Compose

```yaml
version: '3.8'

services:
  app:
    build: .
    ports:
      - "3000:3000"
    environment:
      - DATABASE_URL=postgresql://postgres:password@db:5432/ai_rfp_scanner
      - NEXTAUTH_URL=http://localhost:3000
      - NEXTAUTH_SECRET=your-secret
      - ABACUSAI_API_KEY=your-api-key
    depends_on:
      - db

  db:
    image: postgres:15
    environment:
      - POSTGRES_DB=ai_rfp_scanner
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=password
    volumes:
      - postgres_data:/var/lib/postgresql/data
    ports:
      - "5432:5432"

volumes:
  postgres_data:
```

3. VPS/Cloud Server Deployment

Using PM2 (Process Manager)

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
npm install -g pm2 yarn

# Install PostgreSQL
sudo apt install postgresql postgresql-contrib
```

1. **Application Setup**
   ```bash
   # Clone repository
   git clone
   cd ai_rfp_risk_scanner/app

# Install dependencies
yarn install
   ```

```
# Setup environment
cp ../.env.example .env.production
# Edit .env.production

# Setup database
npx prisma generate
npx prisma db push

# Build application
yarn build
```

1. **PM2 Configuration**
   ```javascript
   // ecosystem.config.js
   module.exports = {
     apps: [{
       name: 'ai-rfp-scanner',
       script: 'npm',
       args: 'start',
       cwd: './app',
       instances: 'max',
       exec_mode: 'cluster',
       env: {
         NODE_ENV: 'production',
         PORT: 3000
       },
       env_production: {
         NODE_ENV: 'production',
         PORT: 3000
       }
     }]
   }
   ```

2. **Start Application**
   ```bash
   # Start with PM2
   pm2 start ecosystem.config.js –env production

   # Save PM2 configuration
   pm2 save
   pm2 startup
   ```

4. Nginx Configuration

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

    # Redirect HTTP to HTTPS
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl http2;
    server_name your-domain.com;

    # SSL Configuration
    ssl_certificate /path/to/ssl/certificate.crt;
    ssl_certificate_key /path/to/ssl/private.key;

    # Security headers
    add_header X-Frame-Options "SAMEORIGIN" always;
    add_header X-XSS-Protection "1; mode=block" always;
    add_header X-Content-Type-Options "nosniff" always;

    # File upload size
    client_max_body_size 50M;

    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_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_cache_bypass $http_upgrade;
    }

    # Static file serving
    location /_next/static/ {
        proxy_pass http://localhost:3000;
        add_header Cache-Control "public, max-age=31536000, immutable";
    }
}
```

Database Deployment Options

1. Supabase (Recommended for Vercel)

- Managed PostgreSQL with built-in auth
- Free tier available
- Global CDN and edge functions

2. PlanetScale

- Serverless MySQL with branching
- Schema changes without downtime
- Built-in analytics

3. AWS RDS

- Fully managed PostgreSQL
- Multi-AZ deployment for high availability
- Automated backups and monitoring

4. Self-hosted PostgreSQL

```
# Install PostgreSQL
sudo apt install postgresql postgresql-contrib

# Create database and user
sudo -u postgres psql
CREATE DATABASE ai_rfp_scanner;
CREATE USER app_user WITH PASSWORD 'secure_password';
GRANT ALL PRIVILEGES ON DATABASE ai_rfp_scanner TO app_user;
```

Security Checklist

Environment Security

- [ ] Use strong, unique NEXTAUTH_SECRET
- [ ] Configure proper CORS settings
- [ ] Use HTTPS in production
- [ ] Secure database connections
- [ ] Implement rate limiting
- [ ] Configure CSP headers

File Upload Security

- [ ] Validate file types and sizes
- [ ] Scan uploads for malware
- [ ] Store uploads outside web root
- [ ] Implement access controls

Database Security

- [ ] Use connection pooling
- [ ] Enable query logging
- [ ] Regular security updates
- [ ] Backup encryption

Monitoring and Maintenance

Application Monitoring

```
# PM2 monitoring
pm2 monit

# View logs
pm2 logs ai-rfp-scanner

# Restart application
pm2 restart ai-rfp-scanner
```

Database Monitoring

```
-- Check active connections
SELECT count(*) FROM pg_stat_activity;

-- Monitor query performance
SELECT query, mean_time, calls
FROM pg_stat_statements
ORDER BY mean_time DESC
LIMIT 10;
```

Backup Strategy

```
# Database backup
pg_dump -h localhost -U app_user ai_rfp_scanner > backup.sql

# File backup
tar -czf uploads_backup.tar.gz uploads/
```

Troubleshooting

Common Issues

1. **Build Failures**
   - Check Node.js version compatibility
   - Verify all environment variables
   - Clear node_modules and reinstall

2. **Database Connection Issues**
   - Verify connection string format
   - Check firewall settings
   - Ensure database is running

3. **File Upload Problems**
   - Check file permissions
   - Verify upload directory exists
   - Review file size limits

4. **Memory Issues**
   - Monitor memory usage with `htop`

- Adjust PM2 instance count
- Consider server upgrades

Performance Optimization

1. **Database Optimization**
   - Add proper indexes
   - Use connection pooling
   - Optimize queries

2. **Application Optimization**
   - Enable Next.js caching
   - Use CDN for static assets
   - Implement Redis for sessions

3. **Server Optimization**
   - Use HTTP/2
   - Enable gzip compression
   - Configure proper caching headers

Scaling Considerations

Horizontal Scaling

- Load balancer configuration
- Database read replicas
- Microservices architecture

Vertical Scaling

- Server resource monitoring
- Performance bottleneck identification
- Capacity planning

For additional support, refer to the main README.md or create an issue in the repository.