SolarFactsNW Production Deployment Guide

🚀 **Complete deployment guide for SolarFactsNW Production Powerhouse v3.0**

Pre-Deployment Checklist

System Requirements

- [ ] Ubuntu 20.04+ or CentOS 8+
- [ ] Node.js 18+ installed
- [ ] MySQL 8.0+ running
- [ ] Nginx installed (for reverse proxy)
- [ ] SSL certificates configured
- [ ] Domain name configured
- [ ] Firewall configured (ports 80, 443, 1880)

API Keys & Credentials

- [ ] OpenAI API key obtained
- [ ] Telnyx API key and phone number configured
- [ ] Retell API key obtained
- [ ] MySQL database and user created
- [ ] Strong passwords generated for all services

Step-by-Step Deployment

1. Server Preparation

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

# Install required packages
sudo apt install -y curl wget git nginx mysql-client redis-tools

# Install Node.js 18
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt install -y nodejs

# Create application user
sudo useradd -m -s /bin/bash solarfacts
sudo usermod -aG sudo solarfacts
```

2. Application Deployment

```
# Clone repository
cd /opt
sudo git clone <repository-url> solarfactsnw
sudo chown -R solarfacts:solarfacts solarfactsnw

# Switch to application user
sudo su - solarfacts
cd /opt/solarfactsnw

# Install dependencies
npm ci --production

# Configure environment
cp .env.example .env.local
node scripts/generate-secrets.js

# Edit .env.local with your credentials
nano .env.local
```

3. Environment Configuration

Edit `.env.local` with your production values:

```
# Production Environment Configuration
NODE_RED_CREDENTIAL_SECRET=your_generated_secret_here
NODE_ENV=production

# API Configuration
OPENAI_API_KEY=sk-your_openai_key_here
TELNYX_API_KEY=your_telnyx_key_here
RETELL_API_KEY=your_retell_key_here

# Database Configuration
MYSQL_HOST=localhost
MYSQL_PORT=3306
MYSQL_DATABASE=solarfacts_production
MYSQL_USER=solarfacts_user
MYSQL_PASSWORD=your_secure_password_here

# Application Configuration
APP_ENV=production
APP_PORT=1880
APP_HOST=127.0.0.1

# Security Configuration
ADMIN_USERNAME=admin
ADMIN_PASSWORD=your_secure_admin_password_here
SESSION_SECRET=your_session_secret_here
JWT_SECRET=your_jwt_secret_here

# External URLs
WEBHOOK_BASE_URL=https://your-domain.com
CALLBACK_BASE_URL=https://your-domain.com/callback
```

4. Database Setup

```
# Connect to MySQL
mysql -u root -p

# Create database and user
CREATE DATABASE solarfacts_production;
CREATE USER 'solarfacts_user'@'localhost' IDENTIFIED BY 'your_secure_password_here';
GRANT ALL PRIVILEGES ON solarfacts_production.* TO 'solarfacts_user'@'localhost';
FLUSH PRIVILEGES;
EXIT;
```

5. Security Hardening

```
# Migrate credentials
node scripts/migrate-credentials.js

# Install health system
node scripts/install-health-system.js

# Set proper permissions
chmod 600 .env.local
chmod +x scripts/*.sh
chmod +x scripts/*.js
```

6. Service Configuration

```
# Install systemd service
sudo cp dist/solarfactsnw.service /etc/systemd/system/
sudo systemctl daemon-reload
sudo systemctl enable solarfactsnw
```

7. Nginx Configuration

```
# Copy nginx configuration
sudo cp nginx/nginx.conf /etc/nginx/sites-available/solarfactsnw
sudo ln -sf /etc/nginx/sites-available/solarfactsnw /etc/nginx/sites-enabled/
sudo rm -f /etc/nginx/sites-enabled/default

# Test nginx configuration
sudo nginx -t

# Create SSL certificate directory
sudo mkdir -p /etc/nginx/ssl

# Copy your SSL certificates
sudo cp your-cert.pem /etc/nginx/ssl/cert.pem
sudo cp your-key.pem /etc/nginx/ssl/key.pem
sudo chmod 600 /etc/nginx/ssl/*
```

8. SSL Certificate Setup

Option A: Let's Encrypt (Recommended)

```
# Install certbot
sudo apt install -y certbot python3-certbot-nginx

# Obtain certificate
sudo certbot --nginx -d your-domain.com

# Test auto-renewal
sudo certbot renew --dry-run
```

Option B: Self-Signed (Development Only)

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
  -keyout /etc/nginx/ssl/key.pem \
  -out /etc/nginx/ssl/cert.pem \
  -subj "/C=US/ST=State/L=City/O=Organization/CN=your-domain.com"
```

9. Firewall Configuration

```
# Configure UFW firewall
sudo ufw allow ssh
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
sudo ufw --force enable

# Check status
sudo ufw status
```

10. Start Services

```
# Start SolarFactsNW
sudo systemctl start solarfactsnw

# Start Nginx
sudo systemctl restart nginx

# Check service status
sudo systemctl status solarfactsnw
sudo systemctl status nginx
```

11. Health Check Verification

```
# Run startup health check
cd /opt/solarfactsnw
node scripts/startup-health-check.js

# Check health endpoint
curl -k https://your-domain.com/health

# Access health dashboard
# Open browser: https://your-domain.com/health/dashboard
```

Docker Deployment (Alternative)

1. Docker Compose Deployment

```
# Clone repository
git clone <repository-url> solarfactsnw
cd solarfactsnw

# Configure environment
cp .env.example .env.local
# Edit .env.local with your values

# Deploy with Docker Compose
docker-compose up -d

# Check container status
docker-compose ps

# View logs
docker-compose logs -f solarfactsnw
```

2. Production Docker Deployment

```
# Deploy with monitoring stack
docker-compose --profile monitoring up -d

# Scale application (if needed)
docker-compose up -d --scale solarfactsnw=3
```

Post-Deployment Verification

1. Service Health Checks

```
# Check all services
curl -k https://your-domain.com/health

# Check individual services
curl -k https://your-domain.com/health/openai
curl -k https://your-domain.com/health/telnyx
curl -k https://your-domain.com/health/retell
curl -k https://your-domain.com/health/mysql
```

2. Access Verification

- [ ] Main interface: `https://your-domain.com`
- [ ] Health dashboard: `https://your-domain.com/health/dashboard`
- [ ] System dashboard: `https://your-domain.com/dashboard`
- [ ] Admin interface: `https://your-domain.com/admin` (requires auth in production)

3. Performance Testing

```
# Test response times
curl -w "@curl-format.txt" -o /dev/null -s https://your-domain.com/health

# Load testing (install apache2-utils first)
ab -n 100 -c 10 https://your-domain.com/health
```

Monitoring & Maintenance

1. Log Monitoring

```
# Application logs
tail -f /opt/solarfactsnw/logs/info.log
tail -f /opt/solarfactsnw/logs/error.log

# System logs
sudo journalctl -u solarfactsnw -f
sudo journalctl -u nginx -f
```

2. Health Monitoring

```
# Automated health monitoring script
cat > /opt/solarfactsnw/scripts/monitor.sh << 'EOF'
#!/bin/bash
HEALTH_URL="https://your-domain.com/health"
ALERT_EMAIL="admin@your-domain.com"

if ! curl -f -s "$HEALTH_URL" > /dev/null; then
    echo "SolarFactsNW health check failed" | mail -s "Alert: SolarFactsNW Down"
"$ALERT_EMAIL"
fi
EOF

chmod +x /opt/solarfactsnw/scripts/monitor.sh

# Add to crontab
echo "*/5 * * * * /opt/solarfactsnw/scripts/monitor.sh" | crontab -
```

3. Backup Strategy

```
# Database backup script
cat > /opt/solarfactsnw/scripts/backup.sh << 'EOF'
#!/bin/bash
BACKUP_DIR="/opt/backups/solarfactsnw"
DATE=$(date +%Y%m%d_%H%M%S)

mkdir -p "$BACKUP_DIR"

# Backup database
mysqldump -u solarfacts_user -p solarfacts_production > "$BACKUP_DIR/db_$DATE.sql"

# Backup application data
tar -czf "$BACKUP_DIR/app_$DATE.tar.gz" /opt/solarfactsnw --exclude=node_modules

# Keep only last 7 days of backups
find "$BACKUP_DIR" -name "*.sql" -mtime +7 -delete
find "$BACKUP_DIR" -name "*.tar.gz" -mtime +7 -delete
EOF

chmod +x /opt/solarfactsnw/scripts/backup.sh

# Schedule daily backups
echo "0 2 * * * /opt/solarfactsnw/scripts/backup.sh" | crontab -
```

Troubleshooting

Common Issues

Service Won't Start

```
# Check logs
sudo journalctl -u solarfactsnw -n 50

# Check configuration
node -c settings.js

# Verify permissions
ls -la /opt/solarfactsnw/.env.local
```

Health Checks Failing

```
# Run manual health check
cd /opt/solarfactsnw
node scripts/startup-health-check.js

# Check API connectivity
curl -v https://api.openai.com/v1/models -H "Authorization: Bearer $OPENAI_API_KEY"
```

SSL Certificate Issues

```
# Check certificate validity
openssl x509 -in /etc/nginx/ssl/cert.pem -text -noout

# Test SSL configuration
openssl s_client -connect your-domain.com:443
```

Database Connection Issues

```
# Test database connection
mysql -u solarfacts_user -p -h localhost solarfacts_production

# Check MySQL status
sudo systemctl status mysql
```

Performance Optimization

Nginx Optimization

```
# Edit nginx configuration
sudo nano /etc/nginx/nginx.conf

# Add to http block:
worker_processes auto;
worker_connections 2048;
keepalive_timeout 30;
client_max_body_size 100M;
```

Node.js Optimization

```
# Set Node.js memory limit
export NODE_OPTIONS="--max-old-space-size=2048"

# Add to systemd service file
sudo systemctl edit solarfactsnw
```

Security Hardening

1. System Security

```
# Disable root login
sudo sed -i 's/PermitRootLogin yes/PermitRootLogin no/' /etc/ssh/sshd_config
sudo systemctl restart ssh

# Install fail2ban
sudo apt install -y fail2ban
sudo systemctl enable fail2ban
```

2. Application Security

```
# Set secure file permissions
chmod 600 /opt/solarfactsnw/.env.local
chmod 700 /opt/solarfactsnw/logs
chown -R solarfacts:solarfacts /opt/solarfactsnw
```

3. Database Security

```
# Run MySQL security script
sudo mysql_secure_installation

# Remove test databases and users
mysql -u root -p -e "DROP DATABASE IF EXISTS test;"
mysql -u root -p -e "DELETE FROM mysql.user WHERE User='';"
mysql -u root -p -e "FLUSH PRIVILEGES;"
```

Scaling Considerations

Horizontal Scaling

- Use load balancer (HAProxy/Nginx)
- Shared database and Redis
- Session store in Redis
- File storage on shared filesystem

Vertical Scaling

- Increase server resources
- Optimize Node.js memory settings
- Database performance tuning
- Enable caching layers

🎉 **Deployment Complete!**

Your SolarFactsNW Production Powerhouse v3.0 is now running securely in production mode with comprehensive health monitoring and auto-healing capabilities.

Access Points:
- Main Application: `https://your-domain.com`
- Health Dashboard: `https://your-domain.com/health/dashboard`
- System Dashboard: `https://your-domain.com/dashboard`
- Admin Interface: `https://your-domain.com/admin`

Next Steps:
1. Configure your solar lead generation flows
2. Set up monitoring alerts
3. Test auto-healing functionality
4. Configure backup procedures
5. Plan scaling strategy