Giant Sloth Orchard Weather Server - Production Deployment Guide

🌺 Production Hosting Setup

Hosting Requirements

- **Node.js Support**: Version 14.0.0 or higher
- **NPM Package Installation**: Ability to run `npm install`
- **Port Access**: HTTP/HTTPS port availability
- **Process Management**: PM2 or similar (recommended)
- **SSL Certificate**: For HTTPS (recommended)

Supported Hosting Platforms

- **VPS/Dedicated Servers**: DigitalOcean, Linode, AWS EC2, Vultr
- **Platform as a Service**: Heroku, Railway, Render, Vercel (with serverless functions)
- **Shared Hosting**: Any provider with Node.js support
- **Cloud Platforms**: AWS, Google Cloud, Azure

🚀 Deployment Methods

Option 1: VPS/Cloud Server (Recommended)

Step 1: Server Preparation

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

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

# Install PM2 globally for process management
sudo npm install -g pm2

# Verify installations
node --version
npm --version
pm2 --version
```

Step 2: Deploy Application

bash

```
# Create application directory
sudo mkdir -p /var/www/giant-sloth-weather
cd /var/www/giant-sloth-weather

# Upload your files (via SCP, Git, or FTP)
# Required files:
# - proxy-server.js
# - package.json
# - public/ directory with all your website files

# Set proper permissions
sudo chown -R $USER:$USER /var/www/giant-sloth-weather

# Install dependencies
npm install --production

# Create required directories
mkdir -p logs data backups
```

Step 3: Configure Environment

bash

```
# Create production environment file
cat > .env << EOF
NODE_ENV=production
PORT=3000
CORS_ORIGIN=https://yourdomain.com
API_TIMEOUT=10000
LOG_LEVEL=info
EOF
```

Step 4: Start with PM2

bash

```
# Start the application
pm2 start proxy-server.js --name "giant-sloth-weather"

# Configure auto-restart on boot
pm2 startup
sudo env PATH=$PATH:/usr/bin /usr/lib/node_modules/pm2/bin/pm2 startup systemd -u $USER --hp /home/$USER
pm2 save

# View logs
pm2 logs giant-sloth-weather
```

Option 2: Heroku Deployment

Step 1: Prepare for Heroku

bash

```
# Install Heroku CLI
# Create Procfile
echo "web: node proxy-server.js" > Procfile

# Update package.json start script
# "start": "node proxy-server.js"
```

Step 2: Deploy to Heroku

bash

```
# Login and create app
heroku login
heroku create giant-sloth-weather-station

# Set environment variables
heroku config:set NODE_ENV=production
heroku config:set NPM_CONFIG_PRODUCTION=false

# Deploy
git add .
git commit -m "Deploy to Heroku"
git push heroku main

# Open app
heroku open
```

Option 3: Static Hosting + Serverless Functions

For Vercel/Netlify with API Routes

javascript

```javascript
// api/weather/[...path].js (Vercel)
export default async function handler(req, res) {
  // Proxy logic for weather APIs
  // Similar to proxy-server.js but as serverless function
}
```

🔧 Production Configuration

1. Environment Variables

bash

```bash
# Essential production variables
NODE_ENV=production
PORT=3000
CORS_ORIGIN=https://yourdomain.com
API_TIMEOUT=10000
WEATHERLINK_CLOUD_URL=https://api.weatherlink.com/v2
MAX_REQUEST_SIZE=1mb
RATE_LIMIT_MAX=100
RATE_LIMIT_WINDOW=900000
```

2. Reverse Proxy Configuration (Nginx)

nginx

```nginx
# /etc/nginx/sites-available/giant-sloth-weather
server {
    listen 80;
    server_name yourdomain.com www.yourdomain.com;
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl http2;
    server_name yourdomain.com www.yourdomain.com;

    ssl_certificate /path/to/your/certificate.crt;
    ssl_certificate_key /path/to/your/private.key;

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

    # Gzip compression
    gzip on;
    gzip_types text/plain text/css application/json application/javascript text/xml application/xml;

    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;
        proxy_read_timeout 300s;
        proxy_connect_timeout 75s;
    }

    # Cache static assets
    location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg)$ {
        expires 1y;
        add_header Cache-Control "public, immutable";
        proxy_pass http://localhost:3000;
    }
}
```

3. SSL Certificate (Let's Encrypt)

bash

```bash
# Install Certbot
sudo apt install certbot python3-certbot-nginx

# Get certificate
sudo certbot --nginx -d yourdomain.com -d www.yourdomain.com

# Auto-renewal
sudo crontab -e
# Add: 0 12 * * * /usr/bin/certbot renew --quiet
```

🌐 Weather API Configuration in Production

1. Cloud API Setup (WeatherLink v2)

javascript

```javascript
// Production proxy URLs for weather-manager.js
const apiSettings = {
    cloud: {
        apiKey: '', // Set in weather settings UI
        stationId: '', // Set in weather settings UI
        apiSecret: '', // Set in weather settings UI
        baseUrl: 'https://yourdomain.com/api/cloud' // Your production proxy
    },
    local: {
        ip: '192.168.1.100',
        port: '80',
        https: false,
        proxyUrl: 'https://yourdomain.com' // Your production domain
    }
};
```

2. Update Weather Manager for Production

javascript

```
// In weather-manager.js, update proxy URLs for production
function fetchCloudData() {
    // Production URL will be: https://yourdomain.com/api/cloud/current/{stationId}
}

function fetchLocalData() {
    // Production URL will be: https://yourdomain.com/api/weather/current_conditions
}
```

3. CORS Configuration

javascript

```
// proxy-server.js production CORS
app.use(cors({
    origin: [
        'https://yourdomain.com',
        'https://www.yourdomain.com',
        process.env.NODE_ENV === 'development' ? 'http://localhost:3000' : false
    ].filter(Boolean),
    credentials: true
}));
```

📊 Production Monitoring

Health Checks

bash

```
# Setup monitoring endpoints
curl https://yourdomain.com/api/health
curl https://yourdomain.com/api/status

# Monitor with external services
# - UptimeRobot
# - Pingdom
# - StatusCake
```

Log Management

bash

```
# PM2 log management
pm2 logs giant-sloth-weather --lines 100
pm2 flush giant-sloth-weather

# Rotate logs (add to crontab)
0 0 * * * pm2 reloadLogs
```

Performance Monitoring

bash

```
# PM2 monitoring
pm2 monit

# System resources
htop
df -h
free -m
```

🔒 Production Security

1. Firewall Configuration

bash

```
# UFW firewall setup
sudo ufw enable
sudo ufw allow ssh
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
sudo ufw deny 3000/tcp  # Block direct access to Node.js
```

2. Security Headers

javascript

```
// Add to proxy-server.js
app.use((req, res, next) => {
    res.setHeader('X-Frame-Options', 'SAMEORIGIN');
    res.setHeader('X-Content-Type-Options', 'nosniff');
    res.setHeader('X-XSS-Protection', '1; mode=block');
    res.setHeader('Referrer-Policy', 'strict-origin-when-cross-origin');
    next();
});
```

3. Rate Limiting

bash

```
# Install rate limiting
npm install express-rate-limit

# Add to proxy-server.js
const rateLimit = require('express-rate-limit');
app.use(rateLimit({
  windowMs: 15 * 60 * 1000, // 15 minutes
  max: 100 // limit each IP to 100 requests per windowMs
}));
```

🎯 Production Testing Checklist

Pre-Deployment Tests

- [ ] Server starts without errors
- [ ] All static files serve correctly
- [ ] Demo mode works with green LED
- [ ] Weather settings panel loads and saves
- [ ] API proxy endpoints respond correctly
- [ ] HTTPS redirects work properly
- [ ] Charts and visualizations load
- [ ] Mobile responsive design works

Post-Deployment Tests

bash

```
# Test all endpoints
curl -f https://yourdomain.com/
curl -f https://yourdomain.com/api/health
curl -f https://yourdomain.com/api/status
curl -f https://yourdomain.com/api/demo/current

# Test weather functionality
# 1. Open https://yourdomain.com
# 2. Navigate to Weather page
# 3. Verify demo mode works (green LED)
# 4. Test settings panel
# 5. Configure real API if available
# 6. Test all weather modes
```

Performance Tests

```bash
# Load testing with Apache Bench
ab -n 100 -c 10 https://yourdomain.com/
ab -n 50 -c 5 https://yourdomain.com/api/demo/current
```

🔧 Production Maintenance

Regular Tasks

```bash
# Weekly: Update dependencies
npm audit
npm update

# Monthly: Clean logs
pm2 flush giant-sloth-weather

# Quarterly: Full system update
sudo apt update && sudo apt upgrade -y
```

Backup Strategy

```bash
# Daily backup script
#!/bin/bash
DATE=$(date +%Y%m%d)
tar -czf /backups/weather-$DATE.tar.gz /var/www/giant-sloth-weather
```

Troubleshooting Production Issues

1. **Service Won't Start**

   ```bash
   pm2 logs giant-sloth-weather
   pm2 restart giant-sloth-weather
   ```

2. **High Memory Usage**

   ```bash
   pm2 reload giant-sloth-weather
   ```

3. **API Connection Issues**

```bash
curl https://yourdomain.com/api/status
# Check network connectivity to WeatherLink APIs
```

✅ Production Success Metrics

Your production deployment is successful when:

1. ✅ Website loads quickly over HTTPS
2. ✅ Weather dashboard shows live data with stable connection
3. ✅ All weather modes work (demo, local, cloud, auto)
4. ✅ Health checks return successful responses
5. ✅ SSL certificate is valid and auto-renewing
6. ✅ Server monitoring shows stable performance
7. ✅ Logs show no critical errors
8. ✅ Mobile responsive design works perfectly

🎉 Production Optimization

Performance Enhancements

- Enable Gzip compression
- Set proper cache headers for static assets
- Use CDN for static files (optional)
- Optimize images and assets
- Monitor and tune PM2 settings

Advanced Features

- Implement Redis for session management
- Add database for historical data storage
- Set up automated backups
- Configure alerting for service disruptions
- Add API analytics and usage tracking

🦥 **Your Giant Sloth Orchard Weather Station is now ready for production!**

For support or issues, check the server logs and health endpoints. The system is designed to be robust and self-healing, with automatic fallbacks between weather data sources.