🌐 One.com Hosting Deployment Guide - SOaC Framework

Complete guide for deploying the SOaC Framework Phase 2 to one.com hosting for public testing and production.

📋 Table of Contents

🎯 Prerequisites

Required

- ✅ Active one.com hosting account
- ✅ FTP/SFTP access credentials
- ✅ SSH access (if available with your plan)
- ✅ Domain or subdomain configured
- ✅ Node.js support (check with one.com)

Recommended

- ✅ FileZilla or similar FTP client
- ✅ Terminal/SSH client
- ✅ Postman for API testing
- ✅ Local copy of project ready for deployment

🏢 One.com Requirements

Hosting Plan Requirements

Check that your one.com hosting plan includes:

Feature	Required	Notes
Node.js Support	✅ Yes	Version 18+ preferred
SSH Access	⭐ Recommended	Makes deployment easier
FTP/SFTP Access	✅ Yes	For file uploads
Custom Domains	✅ Yes	For production URL
SSL Certificate	⭐ Recommended	For HTTPS
Database	✅ Yes	MySQL or PostgreSQL

Important Limitations

⚠️ **Be aware of one.com limitations:**

1. **Node.js**: Not all one.com plans support Node.js applications
2. **Port Restrictions**: May not allow custom ports (like 5000)
3. **Process Management**: May require specific startup methods
4. **Memory Limits**: Shared hosting has limited RAM
5. **Database**: Usually MySQL (not PostgreSQL by default)

Recommendation: Consider upgrading to VPS hosting if standard shared hosting doesn't support Node.js.

🛠️ Pre-Deployment Steps

Step 1: Verify One.com Node.js Support

Contact One.com Support or check your control panel:

```
Login to One.com Control Panel
→ Hosting Settings
→ Check for "Node.js" or "Application Hosting"
```

If Node.js is **not supported**, consider:
- Upgrading to VPS plan
- Using alternative hosting (Heroku, DigitalOcean, AWS, Vercel)

Step 2: Gather One.com Credentials

You'll need:
- **FTP/SFTP Host**: Usually `ftp.yourdomain.com` or provided by one.com
- **FTP Username**: Your one.com username
- **FTP Password**: Your FTP password
- **SSH Host**: If available
- **Database Host**: Usually provided in control panel
- **Database Credentials**: Username, password, database name

Step 3: Build Production Version Locally

```
cd soac-framework

# Run the build script
./scripts/build-production.sh

# Or manually:
# Build frontend
cd frontend
npm install
npm run build

# This creates frontend/dist directory with production files
```

🚀 Deployment Methods

Method 1: FTP/SFTP Upload (Standard)

Best for: Shared hosting without SSH access

Method 2: SSH Deployment (Recommended)

Best for: VPS or hosting plans with SSH access

Method 3: Git Deployment

Best for: Advanced setups with Git support

💾 Database Setup

Option A: MySQL Database (Most Common on One.com)

1. Create Database via One.com Control Panel:

```
Login to One.com Control Panel
→ Databases
→ MySQL Databases
→ Create New Database

Database Name: soac_db
Username: soac_user
Password: [generate strong password]
```

2. Note Database Credentials:

```
Host: localhost (or provided hostname)
Port: 3306
Database: soac_db
Username: soac_user
Password: [your password]
```

3. Update Backend Code for MySQL:

Since the project currently uses in-memory storage, you'll need to add MySQL support:

```
# Install MySQL driver (locally first)
cd backend
npm install mysql2
```

4. Create Database Tables:

Access phpMyAdmin (usually in one.com control panel) and run:

```
-- Users table
CREATE TABLE users (
  id INT PRIMARY KEY AUTO_INCREMENT,
  username VARCHAR(255) UNIQUE NOT NULL,
  password VARCHAR(255) NOT NULL,
  role VARCHAR(50) DEFAULT 'user',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Devices table
CREATE TABLE devices (
  id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(255) NOT NULL,
  ip_address VARCHAR(50) NOT NULL,
  type VARCHAR(100) NOT NULL,
  status VARCHAR(50) DEFAULT 'active',
  location VARCHAR(255),
  last_seen TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Rules table
CREATE TABLE rules (
  id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(255) NOT NULL,
  description TEXT,
  severity VARCHAR(50) DEFAULT 'medium',
  enabled BOOLEAN DEFAULT TRUE,
  category VARCHAR(100),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);

-- Alerts table
CREATE TABLE alerts (
  id INT PRIMARY KEY AUTO_INCREMENT,
  title VARCHAR(255) NOT NULL,
  description TEXT,
  severity VARCHAR(50) DEFAULT 'medium',
  status VARCHAR(50) DEFAULT 'open',
  source VARCHAR(255),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  resolved_at TIMESTAMP NULL
);

-- Insert default admin user (password: admin123)
INSERT INTO users (username, password, role)
VALUES ('admin', '$2a$10$YourHashedPasswordHere', 'admin');
```

🔧 Backend Deployment

Method 1: FTP/SFTP Upload

1. Connect via FTP Client (FileZilla):

```
Host: ftp.yourdomain.com
Username: your_username
Password: your_password
Port: 21 (FTP) or 22 (SFTP)
```

2. Upload Backend Files:

```
Local: soac-framework/backend/
Remote: /public_html/api/

Upload these files/folders:
✅ src/
✅ package.json
✅ .env (create on server)
❌ node_modules (install on server)
❌ tests/
❌ .env.example
```

3. Create Production .env File:

Create `/public_html/api/.env` on the server:

```
# Server Configuration
PORT=5000
NODE_ENV=production

# JWT Configuration
JWT_SECRET=your_super_secure_production_secret_key_CHANGE_THIS
JWT_EXPIRES_IN=24h

# Admin Credentials
ADMIN_USERNAME=admin
ADMIN_PASSWORD=YourSecurePassword123!

# Database Configuration
DB_HOST=localhost
DB_PORT=3306
DB_NAME=soac_db
DB_USER=soac_user
DB_PASSWORD=your_database_password

# API Rate Limiting
RATE_LIMIT_WINDOW_MS=900000
RATE_LIMIT_MAX_REQUESTS=100

# CORS Configuration
CORS_ORIGIN=https://yourdomain.com

# URLs
API_URL=https://yourdomain.com/api
FRONTEND_URL=https://yourdomain.com
```

4. Install Dependencies via SSH (if available):

```
ssh your_username@yourdomain.com
cd public_html/api
npm install --production
```

If SSH not available:
- Upload node_modules folder (slower)
- Or request SSH access from one.com support

Method 2: SSH Deployment

1. Connect via SSH:

```
ssh your_username@yourdomain.com
```

2. Navigate to web root:

```
cd public_html
mkdir api
cd api
```

3. Clone repository (if Git is available):

```
git clone https://github.com/ge0mant1s/soac-framework.git temp
mv temp/backend/* .
rm -rf temp
```

4. Install dependencies:

```
npm install --production
```

5. Create .env file:

```
nano .env
# Paste production environment variables
```

6. Start the application:

```
# Using node directly
node src/server.js &

# Or using PM2 (if available)
npm install -g pm2
pm2 start src/server.js --name soac-backend
pm2 save
pm2 startup
```

🎨 Frontend Deployment

Step 1: Build Frontend Locally

```
cd soac-framework/frontend

# Update .env for production
cat > .env << EOF
VITE_API_BASE_URL=https://yourdomain.com/api
VITE_APP_NAME=SOaC Framework
VITE_APP_VERSION=1.0.0
EOF

# Build production files
npm run build
```

This creates `frontend/dist/` directory with:
- `index.html`
- `assets/` folder with JS/CSS

Step 2: Upload Frontend Files via FTP

1. Connect via FTP:

```
Host: ftp.yourdomain.com
Username: your_username
Password: your_password
```

2. Upload Built Files:

```
Local: soac-framework/frontend/dist/
Remote: /public_html/

Upload:
✅ index.html
✅ assets/ (entire folder)
✅ vite.svg (favicon)

Upload to: /public_html/ (root of your domain)
```

3. Verify Files:

After upload, your server should have:

```
/public_html/
  ├── index.html
  ├── assets/
  │   ├── index-[hash].js
  │   └── index-[hash].css
  ├── api/
  │   ├── src/
  │   ├── package.json
  │   └── .env
  └── .htaccess
```

Step 3: Configure .htaccess for React Router

Create `/public_html/.htaccess`:

```
# Enable Rewrite Engine
RewriteEngine On

# If file or directory exists, serve it directly
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d

# Don't rewrite API requests
RewriteCond %{REQUEST_URI} !^/api/

# Redirect all other requests to index.html
RewriteRule . /index.html [L]

# Security Headers
<IfModule mod_headers.c>
    Header set X-Content-Type-Options "nosniff"
    Header set X-Frame-Options "SAMEORIGIN"
    Header set X-XSS-Protection "1; mode=block"
</IfModule>

# Enable compression
<IfModule mod_deflate.c>
    AddOutputFilterByType DEFLATE text/html text/plain text/xml text/css text/javas-
cript application/javascript
</IfModule>

# Browser caching
<IfModule mod_expires.c>
    ExpiresActive On
    ExpiresByType image/jpg "access plus 1 year"
    ExpiresByType image/jpeg "access plus 1 year"
    ExpiresByType image/gif "access plus 1 year"
    ExpiresByType image/png "access plus 1 year"
    ExpiresByType text/css "access plus 1 month"
    ExpiresByType application/javascript "access plus 1 month"
    ExpiresByType text/html "access plus 1 hour"
</IfModule>
```

🌍 Domain Configuration

Option 1: Main Domain

Deploy to your main domain:

```
Frontend: https://yourdomain.com
Backend: https://yourdomain.com/api
```

Option 2: Subdomain

Create a subdomain for SOaC:

In One.com Control Panel:

```
Domains → Subdomains → Create Subdomain
Subdomain: soac.yourdomain.com
Document Root: /public_html/soac/
```

Then upload files to:

```
Frontend: /public_html/soac/
Backend: /public_html/soac/api/
```

Option 3: Separate API Subdomain

Advanced setup:

```
Frontend: https://soac.yourdomain.com
Backend: https://api.yourdomain.com
```

Create two subdomains and deploy separately.

🔒 SSL/HTTPS Setup

One.com SSL Certificate

1. Enable SSL in Control Panel:

```
Login to One.com Control Panel
→ SSL Certificates
→ Enable SSL for your domain
→ Select "Let's Encrypt" (free) or purchase SSL
```

2. Force HTTPS via .htaccess:

Add to `/public_html/.htaccess`:

```
# Force HTTPS
RewriteEngine On
RewriteCond %{HTTPS} off
RewriteRule ^(.*)$ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]
```

3. Update Environment Variables:

```
# In backend/.env
CORS_ORIGIN=https://yourdomain.com
API_URL=https://yourdomain.com/api

# In frontend/.env (before build)
VITE_API_BASE_URL=https://yourdomain.com/api
```

4. Rebuild and Redeploy Frontend:

```
cd frontend
npm run build
# Upload new dist/ contents to server
```

✅ Post-Deployment

Step 1: Start Backend Service

Via SSH:

```
ssh your_username@yourdomain.com
cd public_html/api

# Start with PM2
pm2 start src/server.js --name soac-backend
pm2 save

# Or with Node.js directly
nohup node src/server.js > app.log 2>&1 &
```

Step 2: Test Backend API

```
# Health check
curl https://yourdomain.com/api/health

# Login test
curl -X POST https://yourdomain.com/api/auth/login \
  -H "Content-Type: application/json" \
  -d '{"username":"admin","password":"admin123"}'
```

Step 3: Test Frontend

1. Open browser: `https://yourdomain.com`
2. Verify login page loads
3. Test login with admin credentials
4. Verify all pages work:
   - Dashboard
   - Devices
   - Rules
   - Alerts
5. Check browser console for errors

Step 4: Verify Database Connectivity

```
# Via SSH
mysql -u soac_user -p soac_db

# Run test query
SELECT * FROM users;
SELECT * FROM devices;
```

Step 5: Monitor Logs

```
# Backend logs
tail -f public_html/api/app.log

# Or PM2 logs
pm2 logs soac-backend

# Apache/nginx error logs
tail -f /var/log/apache2/error.log  # Path may vary
```

🔧 Troubleshooting

Issue: “Node.js not available”

Solution: One.com shared hosting may not support Node.js

Options:
1. Upgrade to VPS plan
2. Use alternative hosting (Heroku, Vercel, DigitalOcean)
3. Contact one.com support to enable Node.js

Issue: “Cannot connect to database”

Solution: Verify database credentials

```
# Test MySQL connection
mysql -h localhost -u soac_user -p soac_db

# Check MySQL is running
service mysql status

# Verify credentials in backend/.env
DB_HOST=localhost
DB_USER=soac_user
DB_PASSWORD=correct_password
DB_NAME=soac_db
```

Issue: “502 Bad Gateway” or “Cannot connect to backend”

Solutions:

1. **Check if backend is running:**
   ```
   bash
   ps aux | grep node
   ```
2. **Restart backend:**
   ```
   bash
   pm2 restart soac-backend
   # Or
   killall node
   cd public_html/api
   node src/server.js &
   ```

3. **Check firewall/port:**
   - Ensure port 5000 is open (or use port 80/443)
   - Configure reverse proxy if needed

Issue: “CORS errors in browser”

Solution: Update CORS configuration

```
# In backend/.env
CORS_ORIGIN=https://yourdomain.com

# Restart backend after change
```

Issue: “404 on page refresh”

Solution: React Router needs .htaccess configuration

Upload the .htaccess file from Frontend Deployment

Issue: “Mixed Content” warnings

Solution: Ensure all URLs use HTTPS

```
# In frontend/.env
VITE_API_BASE_URL=https://yourdomain.com/api

# Rebuild frontend
npm run build
```

Issue: “Out of memory” errors

Solution: One.com shared hosting has memory limits

Options:
1. Optimize application (reduce memory usage)
2. Upgrade to higher-tier plan
3. Use VPS with more RAM

Issue: “Permission denied” when uploading files

Solution: Check FTP permissions

```
# Via SSH
chmod -R 755 public_html/
chmod 644 public_html/.htaccess
```

🔄 Updating the Application

Update Backend

```
# Via SSH
cd public_html/api

# Pull latest code
git pull origin main

# Install dependencies
npm install --production

# Restart
pm2 restart soac-backend
```

Update Frontend

```
# Locally
cd frontend
git pull origin main
npm install
npm run build

# Upload new dist/ contents via FTP
# Upload to: /public_html/
```

📊 Performance Optimization

1. Enable Caching

Already included in .htaccess, but verify:

```
<IfModule mod_expires.c>
    ExpiresActive On
    ExpiresByType application/javascript "access plus 1 month"
    ExpiresByType text/css "access plus 1 month"
</IfModule>
```

2. Enable Compression

```
<IfModule mod_deflate.c>
    AddOutputFilterByType DEFLATE text/html text/css application/javascript
</IfModule>
```

3. Optimize Frontend Build

```
cd frontend

# Build with optimizations
npm run build

# Check bundle size
ls -lh dist/assets/
```

4. Use CDN (Optional)

For better performance, consider using a CDN:
- Cloudflare (free tier available)
- AWS CloudFront
- Google Cloud CDN

🛡️ Security Checklist

Before going live:

- [ ] Change default admin password
- [ ] Generate unique JWT_SECRET
- [ ] Enable HTTPS/SSL
- [ ] Update CORS origins
- [ ] Enable rate limiting
- [ ] Remove .env.example from production
- [ ] Disable directory listing
- [ ] Set secure file permissions
- [ ] Enable security headers
- [ ] Implement backup strategy
- [ ] Set up monitoring/logging

🔗 Alternative Hosting Options

If one.com doesn't meet your needs:

Recommended Alternatives

1. **Heroku** (Easy, Free tier available)
   - Node.js support ✅
   - PostgreSQL support ✅
   - Easy deployment ✅

2. **Vercel** (Frontend) + **Railway** (Backend)
   - Free tiers ✅
   - Excellent performance ✅
   - CI/CD integration ✅

3. **DigitalOcean** (VPS)
   - Full control ✅
   - Starting at $5/month ✅
   - Scalable ✅
4. **AWS** / **Google Cloud** / **Azure**
   - Enterprise-grade ✅
   - Complex setup ⚠️
   - Pay-as-you-go ✅

📚 Additional Resources

- One.com Documentation (https://www.one.com/en/support)
- Node.js Deployment Guide (https://nodejs.org/en/docs/guides/nodejs-docker-webapp/)
- PM2 Process Manager (https://pm2.keymetrics.io/)
- Let's Encrypt SSL (https://letsencrypt.org/)

🎉 Success!

If deployment is successful, you should be able to:

✅ Access SOaC Framework at https://yourdomain.com
✅ Login with admin credentials
✅ View dashboard and statistics
✅ Manage devices and rules
✅ Backend API responding at https://yourdomain.com/api
✅ HTTPS enabled and working
✅ All features functioning correctly

Need Help? Contact one.com support or check the main deployment guide (../DEPLOYMENT.md).

Last Updated: November 13, 2024 - Phase 2

Important Note: This localhost refers to the server's localhost, not your local machine. The application will be accessible publicly via your domain name.