readme.md for @push.rocks/smartnginx

Control Nginx programmatically from Node.js with full TypeScript support □□

Issue Reporting and Security

For reporting bugs, issues, or security vulnerabilities, please visit community.foss.global/. This is the central community hub for all issue reporting. Developers who sign and comply with our contribution agreement and go through identification can also get a code.foss.global/ account to submit Pull Requests directly.

Features

- □□ **Dynamic Configuration** - Generate and manage Nginx configs on the fly
- □□ **SSL/TLS Ready** - Built-in support for SSL certificates with automatic HTTP→HTTPS redirects
- □□ **Hot Reload** - Apply configuration changes without downtime
- □□ **Zero-Config Defaults** - Self-signed certificates auto-generated for immediate testing
- □□ **Reverse Proxy Made Easy** - Set up proxy hosts with a single method call
- □□ **Smart Diffing** - Only reloads Nginx when configurations actually change
- □□ **TypeScript First** - Full type definitions included

Install

```
pnpm add @push.rocks/smartnginx
# or
npm install @push.rocks/smartnginx
```

> **Prerequisites**: Nginx must be installed and available in your system PATH.

Quick Start

```
import { SmartNginx } from '@push.rocks/smartnginx';

// Create a SmartNginx instance with a default fallback URL
const nginx = new SmartNginx({
  defaultProxyUrl: 'https://your-default-site.com'
});

// Add a reverse proxy host
nginx.addHostCandidate({
  hostName: 'api.example.com',
  destination: 'localhost',
  destinationPort: 3000,
  privateKey: '<your-ssl-private-key>',
  publicKey: '<your-ssl-certificate>'
});

// Deploy and start Nginx
await nginx.deploy();
```

That's it! Your reverse proxy is now running □□

Usage

Creating the SmartNginx Instance

The `SmartNginx` class is your main interface for managing Nginx:

```
import { SmartNginx } from '@push.rocks/smartnginx';

const nginx = new SmartNginx({
```

```
  defaultProxyUrl: 'https://fallback.example.com',  // Where unmatched requests go
  logger: myCustomLogger  // Optional: pass a @push.rocks/smartlog instance
});
```

Adding Host Configurations

Each host represents a domain/subdomain with its proxy rules and SSL certificates:

```
// Add a host using addHostCandidate()
const myHost = nginx.addHostCandidate({
  hostName: 'app.example.com',        // Domain name
  destination: '127.0.0.1',           // Backend server address
  destinationPort: 8080,              // Backend port
  privateKey: sslPrivateKeyPem,       // SSL private key (PEM format)
  publicKey: sslCertificatePem        // SSL certificate (PEM format)
});
```

Multi-Host Setup

Run multiple sites through a single Nginx instance:

```
// Production API
nginx.addHostCandidate({
  hostName: 'api.myapp.com',
  destination: 'localhost',
  destinationPort: 3000,
  privateKey: apiPrivateKey,
  publicKey: apiCertificate
});

// Admin panel
nginx.addHostCandidate({
  hostName: 'admin.myapp.com',
  destination: 'localhost',
  destinationPort: 4000,
  privateKey: adminPrivateKey,
  publicKey: adminCertificate
});
```

```
// Staging environment
nginx.addHostCandidate({
  hostName: 'staging.myapp.com',
  destination: '192.168.1.100',
  destinationPort: 8080,
  privateKey: stagingPrivateKey,
  publicKey: stagingCertificate
});

// Deploy all at once
await nginx.deploy();
```

Deploying Configurations

The `deploy()` method is smart about changes:

```
// First deploy - writes configs and starts Nginx
await nginx.deploy();

// Add more hosts dynamically
nginx.addHostCandidate({
  hostName: 'newsite.example.com',
  destination: 'localhost',
  destinationPort: 5000,
  privateKey: newPrivateKey,
  publicKey: newCertificate
});

// Second deploy - detects changes, updates configs, hot-reloads Nginx
await nginx.deploy();

// If you call deploy() with no changes, it skips the reload (efficient!)
await nginx.deploy(); // → "hosts have not diverged, skipping nginx reload"
```

Querying Deployed Hosts

```
// Get all deployed hosts
const hosts = await nginx.listDeployedHosts();
console.log(`Running ${hosts.length} hosts`);

// Find a specific host by domain
const apiHost = nginx.getDeployedNginxHostByHostName('api.example.com');
if (apiHost) {
  console.log(`API proxying to ${apiHost.destination}:${apiHost.destinationPort}`);
}
```

Removing Hosts

```
const hostToRemove = nginx.getDeployedNginxHostByHostName('staging.myapp.com');
if (hostToRemove) {
  await nginx.removeDeployedHost(hostToRemove);
  // Nginx automatically reloaded with updated config
}
```

Stopping Nginx

```
// Gracefully stop the Nginx process
await nginx.stop();
```

Generated Configuration

SmartNginx generates production-ready Nginx configurations:

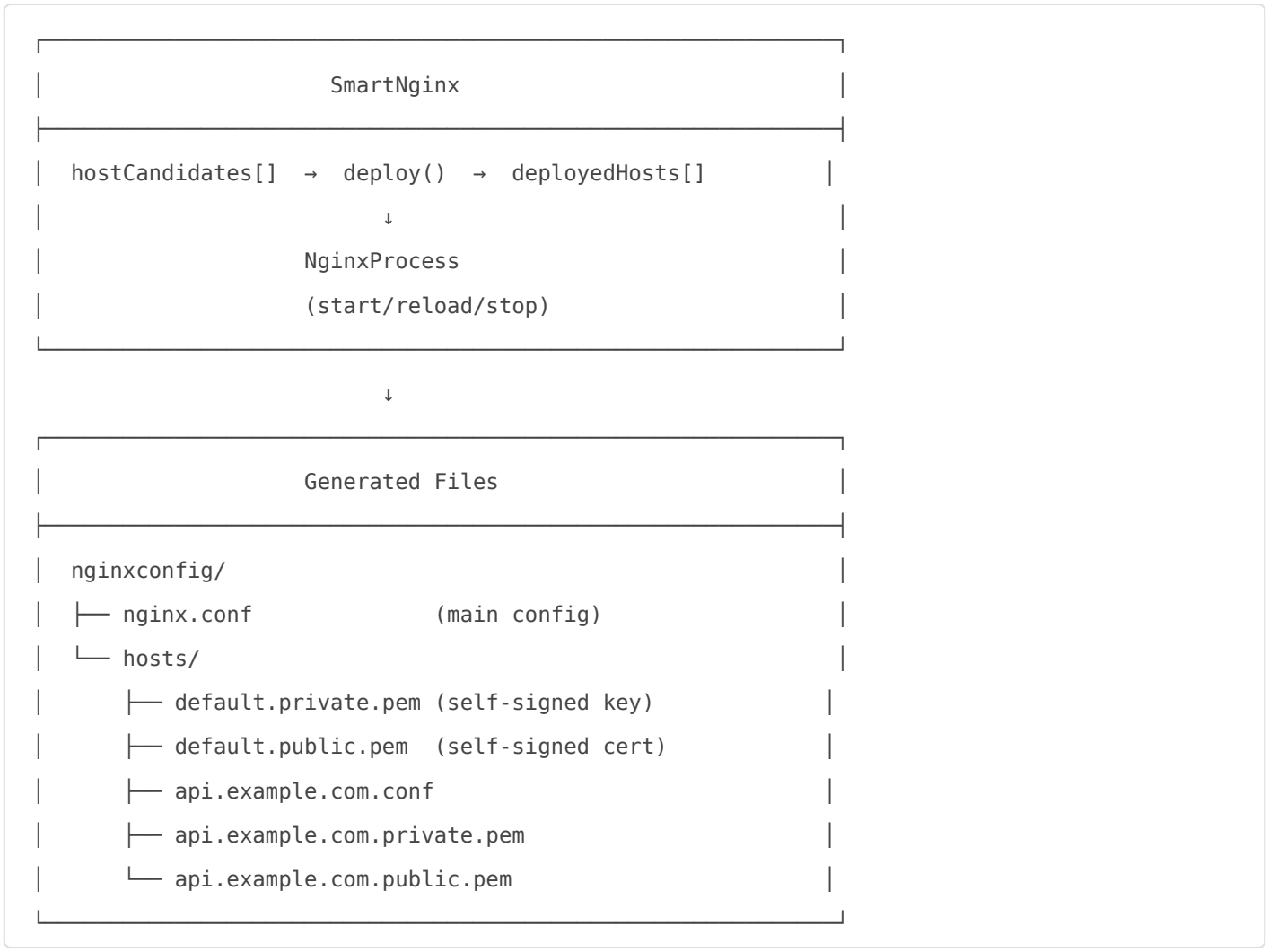
For each host, you get:

- HTTP (port 80) → HTTPS (port 443) automatic redirect
- Upstream with keepalive connections (100 idle connections)
- WebSocket-friendly proxy settings (HTTP/1.1, no buffering)
- Proper proxy headers (`X-Real-IP`, `X-Forwarded-For`, `X-Forwarded-Proto`)
- Smart failover (`proxy_next_upstream` on errors, timeouts, 404/429/500/502)

Default server:

- Self-signed certificate for unmatched domains
- Redirects to your configured `defaultProxyUrl`

Architecture



API Reference

SmartNginx

Method	Description
`addHostCandidate(config)`	Add a new host configuration
`deploy()`	Write configs and start/reload Nginx

Method	Description
`listDeployedHosts()`	Get all currently deployed hosts
`getDeployedNginxHostByHostName(hostname)`	Find a host by domain name
`removeDeployedHost(host)`	Remove a host and reload Nginx
`stop()`	Gracefully stop Nginx

IHostConfig

```
interface IHostConfig {
  hostName: string;        // Domain name (e.g., 'api.example.com')
  destination: string;     // Backend server IP/hostname
  destinationPort: number; // Backend port
  privateKey: string;      // SSL private key (PEM)
  publicKey: string;       // SSL certificate (PEM)
}
```

NginxHost

Each host instance exposes:

- `hostName` - The configured domain
- `destination` - Backend address
- `destinationPort` - Backend port
- `configString` - The generated Nginx config (after deploy)
- `deploy()` - Write this host's config files

License and Legal Information

Trademarks

This project is owned and maintained by Task Venture Capital GmbH. The names and logos associated with Task Venture Capital GmbH and any related products or services are trademarks of Task Venture Capital GmbH and are not included within the scope of the MIT license granted herein. Use of these trademarks must comply with Task Venture Capital GmbH's Trademark Guidelines, and any usage must be approved in writing by Task Venture Capital GmbH.

Company Information

Task Venture Capital GmbH Registered at District court Bremen HRB 35230 HB, Germany

For any legal inquiries or if you require further information, please contact us via email at hello@task.vc.

By using this repository, you acknowledge that you have read this section, agree to comply with its terms, and understand that the licensing of the code does not imply endorsement by Task Venture Capital GmbH of any derivative works.

Revision #3
Created 2026-03-28 11:11:33 UTC by foss.global Team
Updated 2026-03-28 12:18:24 UTC by foss.global Team