

# @push.rocks/smartscaf

# caf

A project aimed at quickly scaffolding projects with support for TypeScript, smart file handling, and template rendering.

- [readme.md for @push.rocks/smartscaf](#)
- [changelog.md for @push.rocks/smartscaf](#)

# readme.md for @push.rocks/smartscaf

**Lightning-fast project scaffolding with smart templates and variable interpolation**

## What It Does

SmartScaf is a powerful TypeScript scaffolding engine that transforms template directories into fully configured projects in seconds. Think of it as your project's DNA replicator - define once, deploy anywhere with custom variables, dependency merging, and automated post-setup scripts.

Perfect for:

- **☐ Bootstrapping microservices** with consistent structure
- **☐ Creating npm packages** with standard configurations
- **☐ Spinning up frontend projects** with your tech stack
- **☐ Standardizing team workflows** with company templates
- **⚡ Automating repetitive project setups**

## Installation

```
# Install globally for CLI usage
npm install -g @push.rocks/smartscaf

# Or add to your project
npm install @push.rocks/smartscaf --save-dev
```

## Quick Start

## Create Your First Template

## 1. Set up a template directory with your project structure:

```
my-template/
├── .smartscaf.yml      # Template configuration
├── package.json        # With {{projectName}} variables
├── src/
│   └── index.ts       # Your source files
├── readme.md          # With {{description}} placeholder
└── .gitignore         # Standard ignores
```

## 2. Configure your template in `.smartscaf.yml`:

```
# Template defaults and configuration
defaults:
  projectName: 'my-awesome-project'
  description: 'A fantastic new project'
  author: '{{gitName}}'
  license: 'MIT'
  nodeVersion: '20'

# Merge other templates (optional)
dependencies:
  merge:
    - ../shared-config-template
    - ../company-standards-template

# Post-scaffold automation
runafter:
  - 'npm install'
  - 'git init'
  - 'git add .'
  - 'git commit -m "Initial commit from SmartScaf"'
```

## 3. Use Handlebars syntax in your template files:

```
// src/index.ts
/**
 * {{description}}
 * @author {{author}}
 * @license {{license}}
```

```
*/  
  
export class {{className}} {  
  constructor() {  
    console.log('Welcome to {{projectName}}!');  
  }  
}
```

## Scaffold a New Project

```
import { ScafTemplate } from '@push.rocks/smartscaf';  
  
async function createProject() {  
  // Load your template  
  const template = await ScafTemplate.createTemplateFromDir('./my-template');  
  await template.readTemplateFromDir();  
  
  // Supply variables (overrides defaults)  
  await template.supplyVariables({  
    projectName: 'super-api',  
    description: 'Revolutionary API service',  
    className: 'SuperAPI',  
    author: 'John Doe'  
  });  
  
  // Interactive mode - asks for any missing variables  
  await template.askCliForMissingVariables();  
  
  // Generate the project  
  await template.writeToDisk('./projects/super-api');  
  
  console.log('☑ Project scaffolded successfully!');  
}  
  
createProject();
```

## Core Features

# ☐☐ Smart Variable System

SmartScaf intelligently manages template variables through multiple layers:

```
const template = await ScafTemplate.createTemplateFromDir('./template');
await template.readTemplateFromDir();

// Access variable collections
console.log(template.defaultVariables); // From .smartscaf.yml defaults
console.log(template.requiredVariables); // Found in template files
console.log(template.suppliedVariables); // What you've provided
console.log(template.missingVariables); // What still needs values
```

# ☐☐ Template Composition

Merge multiple templates to create complex project structures:

```
# .smartscaf.yml
dependencies:
  merge:
    - ../base-template      # Common structure
    - ../auth-template      # Authentication setup
    - ../docker-template    # Container configuration
```

Templates are merged in order, allowing you to build sophisticated scaffolds from modular components.

# ☐☐ Dynamic File Naming

Use frontmatter to dynamically rename files during scaffolding:

```
---
fileName: {{projectName}}.config.js
---

module.exports = {
  name: '{{projectName}}',
  version: '{{version}}'
```

```
};
```

## ☐☐ Interactive CLI Mode

Let SmartScaf prompt for missing variables:

```
// Automatically prompts for any variables not yet supplied  
await template.askCliForMissingVariables();
```

## ☐☐ Post-Scaffold Scripts

Automate setup tasks after scaffolding:

```
runafter:  
- 'npm install'           # Install dependencies  
- 'npm run build'        # Initial build  
- 'npm test'             # Verify setup  
- 'code .'               # Open in VS Code
```

Scripts run in order using an interactive shell, ensuring proper environment handling.

# Advanced Usage

## Programmatic Control

```
import { ScafTemplate } from '@push.rocks/smartscaf';  
  
class ProjectGenerator {  
  private template: ScafTemplate;  
  
  async initialize(templatePath: string) {  
    this.template = await ScafTemplate.createTemplateFromDir(templatePath);  
    await this.template.readTemplateFromDir();  
  }  
  
  async generate(config: any, outputPath: string) {
```

```

// Supply configuration
await this.template.supplyVariables(config);

// Check what's missing
if (this.template.missingVariables.length > 0) {
  console.log('Missing:', this.template.missingVariables);
  // Handle missing variables programmatically
  // or use: await this.template.askCliForMissingVariables();
}

// Generate project
await this.template.writeToDisk(outputPath);
}
}

```

## Template Validation

```

// Validate template before using
const template = await ScafTemplate.createTemplateFromDir('./template');
await template.readTemplateFromDir();

// Check template structure
if (template.templateSmartfileArray.length === 0) {
  throw new Error('Template is empty!');
}

// Verify required variables
const required = template.requiredVariables;
const defaults = Object.keys(template.defaultVariables);
const missing = required.filter(v => !defaults.includes(v));

if (missing.length > 0) {
  console.warn(`Template needs: ${missing.join(', ')}\`);
}

```

## Complex Variable Structures

SmartScaf supports nested object variables:

```
# .smartscaf.yml
defaults:
  app.name: 'MyApp'
  app.version: '1.0.0'
  database.host: 'localhost'
  database.port: 5432
  api.endpoints.users: '/api/users'
  api.endpoints.auth: '/api/auth'
```

Use in templates:

```
// config.js
export default {
  app: {
    name: '{{app.name}}',
    version: '{{app.version}}'
  },
  database: {
    host: '{{database.host}}',
    port: {{database.port}}
  },
  api: {
    users: '{{api.endpoints.users}}',
    auth: '{{api.endpoints.auth}}'
  }
};
```

## CI/CD Integration

```
// ci-scaffold.ts
import { ScaffoldTemplate } from '@push.rocks/smartscaf';

async function scaffoldForCI() {
  const template = await ScaffoldTemplate.createTemplateFromDir(
    process.env.TEMPLATE_PATH
  );
  await template.readTemplateFromDir();
}
```

```
// Get variables from environment
const vars = {
  projectName: process.env.PROJECT_NAME,
  environment: process.env.DEPLOY_ENV,
  apiKey: process.env.API_KEY,
  region: process.env.AWS_REGION
};

await template.supplyVariables(vars);
await template.writeToDisk(process.env.OUTPUT_PATH);
}

// Run in CI pipeline
scaffoldForCI().catch(console.error);
```

# Real-World Examples

## Microservice Template

```
# .smartscaf.yml for microservice template
defaults:
  serviceName: 'my-service'
  port: 3000
  dockerImage: 'node:20-alpine'
  healthPath: '/health'

dependencies:
  merge:
    - ../shared/base-service
    - ../shared/monitoring
    - ../shared/logging

runafter:
  - 'npm install'
  - 'docker build -t {{serviceName}}:latest .'
  - 'npm run test:integration'
```

# React Component Library

```
# .smartscaf.yml for React library
defaults:
  libraryName: 'ui-components'
  componentPrefix: 'UI'
  useTypeScript: true
  useStorybook: true
  testRunner: 'jest'

runafter:
  - 'pnpm install'
  - 'pnpm build'
  - 'pnpm storybook:build'
  - 'pnpm test'
```

## API Reference

### ScafTemplate Class

```
class ScafTemplate {
  // Factory method
  static createTemplateFromDir(dirPath: string): Promise<ScafTemplate>

  // Core methods
  readTemplateFromDir(): Promise<void>
  supplyVariables(variables: object): Promise<void>
  askCliForMissingVariables(): Promise<void>
  writeToDisk(destinationPath: string): Promise<void>

  // Properties
  name: string // Template name
  description: string // Template description
  templateSmartfileArray: SmartFile[] // Loaded template files
  defaultVariables: object // Default values from .smartscaf.yml
```

```
requiredVariables: string[]      // Variables found in templates
suppliedVariables: object        // Variables you've provided
missingVariables: string[]       // Variables still needed
}
```

# Best Practices

## ☐☐ Template Organization

```
company-templates/
├─ base/                # Shared foundations
|  ├─ typescript/       # TS configuration
|  ├─ eslint/           # Linting setup
|  └─ ci-cd/            # CI/CD pipelines
├─ services/           # Service templates
|  ├─ rest-api/
|  ├─ graphql-api/
|  └─ worker-service/
└─ frontends/          # Frontend templates
   ├─ react-app/
   ├─ vue-app/
   └─ static-site/
```

## ☐☐ Security Considerations

- Never commit sensitive data in templates
- Use environment variables for secrets
- Add `.smartscaf.yml` to `.gitignore` if it contains private defaults
- Validate user input when using in automation

## ☐☐ Template Design Tips

1. **Start small** - Begin with minimal templates and grow them
2. **Use composition** - Build complex templates from simple ones
3. **Document variables** - Add comments in `.smartscaf.yml`
4. **Test templates** - Create test scaffolds before deploying

5. **Version templates** - Use git tags for template versions

# License and Legal Information

This repository contains open-source code that is licensed under the MIT License. A copy of the MIT License can be found in the [license](#) file within this repository.

**Please note:** The MIT License does not grant permission to use the trade names, trademarks, service marks, or product names of the project, except as required for reasonable and customary use in describing the origin of the work and reproducing the content of the NOTICE file.

## 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](mailto: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.

# changelog.md for @push.rocks/smartscaf

## 2026-03-24 - 4.0.21 - fix(smartsconfig)

correct smartsconfig package key and add npmjs registry for public releases

- Rename the CLI configuration key from git.zone/cli to @git.zone/cli.
- Add <https://registry.npmjs.org> to the release registries alongside the existing Verdaccio registry.
- Register @ship.zone/szci in the project smart configuration.

## 2025-08-17 - 4.0.19 - fix(readme)

Update README with comprehensive quick start, usage examples, API reference and advanced guides

- Expanded README with a detailed Quick Start guide and installation instructions
- Added examples for creating templates, supplying variables, and scaffolding projects programmatically
- Documented core features: smart variable system, template composition, dynamic file naming, interactive CLI, and post-scaffold scripts
- Included advanced usage, CI/CD integration, template validation, and real-world example templates
- Improved formatting and added code snippets for API reference and ScafTemplate usage

## 2025-08-17 - 4.0.18 - fix(ScafTemplate)

Use interactive shell for post-scaffold scripts; update CI workflows and package metadata

- Switched ScafTemplate.runScripts to use smartshell.execInteractive so post-scaffold 'runafter' commands can run interactively
- Updated Gitea workflow environment: replaced CI image with code.foss.global/host.today/ht-docker-node:npmci and adjusted NPMCI\_COMPUTED\_REPOURL
- Changed CI install target from @shipzone/npmci to @ship.zone/npmci
- Updated package.json metadata (bugs URL and homepage now point to code.foss.global) and added a pnpm.overrides entry
- Minor TypeScript code cleanups and formatting (consistent trailing commas, safer string handling when parsing templates/frontmatter, improved concatenation when merging template files)
- tsconfig adjustments: enabled emitDecoratorMetadata and added baseUrl/paths entries
- Normalized README and template/test output files; updated .gitignore to ignore AI agent directories
- Updated changelog and commitinfo metadata

## 2025-08-17 - 4.0.17 - fix(ScafTemplate)

Use interactive shell for post-scaffold scripts; update test/build config and dependency versions

- Switched from smartshell.exec to smartshell.execInteractive when running smartscf 'runafter' commands to allow interactive commands during post-scaffold execution (ScafTemplate.runScripts).
- Updated package.json scripts: test now runs with additional flags (--verbose --logfile --timeout 60). Added typings entry pointing to dist\_ts/index.d.ts.
- Bumped devDependencies: @git.zone/tsbuild -> ^2.6.4, @git.zone/tstest -> ^2.3.4, @push.rocks/tapbundle -> ^6.0.3.
- Updated runtime dependency versions: @push.rocks/lik -> ^6.2.2, @push.rocks/smartfile -> ^11.2.5, @push.rocks/smartshell -> ^3.3.0 (other deps kept or minor bumps).
- Added pnpm-workspace.yaml with onlyBuiltDependencies configuration (esbuild, mongodb-memory-server, puppeteer).

## 2025-04-15 - 4.0.16 - fix(dependencies)

Update dependency references and bump version numbers; adjust workflow and template commands

- Bump versions for devDependencies (@git.zone/tsbuild, @git.zone/tsrun, @git.zone/tstest, @push.rocks/tapbundle, and @types/node) and dependencies (@push.rocks/lik, @push.rocks/smartfile, @push.rocks/smartfm, @push.rocks/smarthbs, @push.rocks/smartinteract, @push.rocks/smartobject, @push.rocks/smartpromise, @push.rocks/smartshell, and @push.rocks/smartyaml)
- Replace smartparam with smartobject in both plugins and core modules
- Fix workflow install command by renaming '@gitzone/tsdoc' to '@git.zone/tsdoc'
- Update template runafter command from 'npm install' to 'echo "runafter"'

## 2024-05-29 - 4.0.15 - configuration updates

Updates to the project's configuration files, description and build settings.

- Updated description.
- Updated tsconfig (applied in multiple commits).
- Updated npmextra.json (github).

## 2023-08-18 - 4.0.14 - core

Improved core functionality.

- Fixed core update.

## 2023-07-25 - 4.0.13 - core & organization

Enhancements in core behavior and organizational structure.

- Fixed core update.
- Switched to new organization scheme.

# 2023-06-25 - 4.0.12 - core

Core improvements.

- Fixed core update.

# 2023-06-25 - 4.0.11 - core

Core improvements.

- Fixed core update.

# 2023-06-25 - 4.0.10 - core

Core improvements.

- Fixed core update.

# 2023-06-25 - 4.0.9 - core

Core improvements.

- Fixed core update.

# 2023-06-25 - 4.0.8 - core

Core improvements.

- Fixed core update.

# 2023-06-25 - 4.0.7 - core

Core improvements.

- Fixed core update.

## 2023-06-25 - 4.0.6 - core

Core improvements.

- Fixed core update.

## 2023-06-25 - 4.0.5 - core

Core improvements.

- Fixed core update.

## 2023-06-24 - 4.0.4 - core

Core improvements.

- Fixed core update.

## 2023-06-24 - 4.0.3 - core

Core improvements.

- Fixed core update.

## 2022-06-25 - 4.0.2 - core

Core improvements.

- Fixed core update.

## 2022-06-25 - 4.0.1 - core

Core improvements.

- Fixed core update.

## 2022-06-25 - 4.0.0 - core

Core improvements.

- Fixed core update.

## 2022-06-25 - 3.0.10 - core (breaking)

A breaking change was introduced affecting the module format.

- BREAKING CHANGE (core): switched to ESM.

## 2020-01-31 - 3.0.9 - core

Routine core update.

- Fixed core update.

## 2019-10-11 - 3.0.8 - core

Routine core update.

- Fixed core update.

## 2019-10-02 - 3.0.7 - core

Routine core update.

- Fixed core update.

## 2019-09-10 - 3.0.6 - core

Routine core update.

- Fixed core update.

## 2019-09-10 - 3.0.5 - core

Routine core update.

- Fixed core update.

## 2019-09-10 - 3.0.4 - core

Routine core update.

- Fixed core update.

## 2019-09-10 - 3.0.3 - general updates

Updates to project dependencies and general code improvements.

- Performed update.
- Updated dependencies.

## 2019-02-17 - 3.0.2 - core

Routine core update.

- Fixed core update.

## 2019-01-27 - 3.0.1 - core

Routine core update.

- Fixed core update.

## 2018-10-04 - 3.0.0 - core

Changes from merging and core improvements.

- Fixed core update.
- Merged master branch changes and performed additional updates.

## 2018-08-30 - 2.0.2 - structure (breaking)

A breaking change in the project's structure.

- BREAKING CHANGE (structure): templates now take their path within the constructor.

## 2018-08-27 - 2.0.1 - minor

Versions in this range involved only version bump commits with no significant changes.

- No significant changes.

## 2017-08-09 - 1.0.14 - scope (breaking)

A breaking change in the package scope was applied.

- BREAKING CHANGE (scope): switched to new @pushrocks scope.

## 2017-08-09 - 1.0.13 - core

Improvements to variable handling.

- Fixed variable distribution.

## 2017-07-28 - 1.0.12 - dependencies

Dependency updates.

- Updated dependencies.

## 2017-06-01 - 1.0.11 - dependencies

Dependency updates.

- Updated dependencies.

## 2017-06-01 - 1.0.10 - CLI improvements

Enhanced CLI error prevention.

- Prevented error due to empty defaults.yml.

# 2017-05-27 - 1.0.9 - dependencies

Dependency updates.

- Updated dependencies.

# 2017-05-27 - 1.0.8 - templating

Improved file templating support.

- Added support for frontmatter for advanced file templating.

# 2017-05-27 - 1.0.7 - documentation

Documentation improvements.

- Added docs.

# 2017-05-27 - 1.0.6 - fixes

Minor fixes.

- Fixed deep add.

# 2017-05-26 - 1.0.5 - smartfile updates

Smartfile updates.

- Updated smartfile.

# 2017-05-26 - 1.0.4 - tests

Test fixes.

- Fixed tests.

# 2017-05-26 - 1.0.3 - smartfile updates

Smartfile updates.

- Updated smartfile.

# 2017-05-25 - 1.0.2 - functionality

Minor functionality improvements.

- Ensured proper functionality.

# 2017-05-25 - 1.0.1 - core

Core fixes.

- Fixed working issues.

# 2017-05-06 - 1.0.0 - initial release and CLI integration

The initial release introducing CLI support and project setup.

- Added CLI to prompt for missing variables.
- Started CLI integration.
- Added readme and CI tslint configuration.

- Performed initial setup and various update tasks.