readme.md for @ship.zone/szci

A lightweight CI/CD orchestrator built with Deno that unifies Node.js, Docker, NPM, SSH, and Git workflows into a single CLI. Compiles to standalone binaries — no runtime required.

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.

□□ Architecture

szci is a **thin orchestrator** — it doesn't reinvent the wheel. Instead, it wires up best-in-class tools and handles the CI-specific glue:

Domain	What szci does	Delegates to
□□ Docker	Bridges `SZCI_LOGIN_DOCKER*` env vars, auto-detects GitLab CI tokens	`@git.zone/tsdocker`
□□ NPM	Generates `.npmrc` from `SZCI_TOKEN_NPM*` env vars, handles multi-registry publish	`pnpm` + `npm publish`
□□ Node.js	Manages Node versions via NVM with named aliases	`nvm`
□□ SSH	Deploys SSH keys from env vars to `~/.ssh`	`@push.rocks/smartssh`
□□ Git	Mirrors repos to GitHub	`git remote`

```
CLI Input (Deno.args)
    ↓
```

```
SmartCLI Router
    ↓
├─ szci docker *   → env var bridging → npx @git.zone/tsdocker
├─ szci npm *      → .npmrc generation → pnpm / npm publish
├─ szci node *     → version aliasing  → nvm install
├─ szci git *      → token injection   → git push --mirror
└─ szci ssh *      → key parsing       → write to ~/.ssh
```

□□ Installation

Automated Installer (Recommended)

The easiest way to install szci is using the automated installer script. It handles platform detection, binary download, and global setup:

```
curl -sSL https://code.foss.global/ship.zone/szci/raw/branch/main/install.sh | sudo bash
```

To install a specific version:

```
curl -sSL https://code.foss.global/ship.zone/szci/raw/branch/main/install.sh | sudo bash -s --
--version v7.0.0
```

Via NPM (downloads pre-compiled binary)

```
npm install -g @ship.zone/szci
```

From Source

```
git clone https://code.foss.global/ship.zone/szci.git
cd szci
deno task compile
```

The compiled binaries land in `dist/binaries/` for Linux (x64/arm64), macOS (x64/arm64), and Windows (x64).

□□ Quick Start

```
# Setup Node.js environment
szci node install stable

# Install project dependencies
szci npm install

# Build & push Docker images
szci docker build
szci docker push registry.example.com

# Run tests
szci npm test
```

□□ CLI Reference

`szci docker` — Docker Operations

All Docker commands delegate to `@git.zone/tsdocker` after bridging environment variables.

```
szci docker build                       # Build all Dockerfiles in cwd
szci docker login                       # Login to all configured registries
szci docker prepare                     # Alias for login
szci docker push registry.example.com   # Push images to a registry
szci docker pull registry.example.com   # Pull images from a registry
szci docker test                        # Test Dockerfiles
```

Env var bridging: Before delegating, szci converts its own env var format to tsdocker's expected format:

```
SZCI_LOGIN_DOCKER_1  →  DOCKER_REGISTRY_1
SZCI_LOGIN_DOCKER_2  →  DOCKER_REGISTRY_2
...
```

In GitLab CI, `CI_JOB_TOKEN` is automatically bridged as `DOCKER_REGISTRY_0` for `registry.gitlab.com`.

`szci npm` — NPM/pnpm Workflows

```
szci npm install     # Runs pnpm install
szci npm build       # Runs pnpm run build
szci npm test        # Runs pnpm test
szci npm prepare     # Generates ~/.npmrc from SZCI_TOKEN_NPM* env vars
szci npm publish     # Full workflow: prepare → install → build → clean → npm publish
```

The `publish` command supports multi-registry publishing. If `npmAccessLevel` is `public` and a Verdaccio registry is configured, it publishes to both npm and Verdaccio automatically.

`szci node` — Node.js Version Management

```
szci node install stable    # Node.js 22
szci node install lts       # Node.js 20
szci node install legacy    # Node.js 18
szci node install 21        # Any specific version
```

Uses NVM under the hood (auto-detected at `/usr/local/nvm/nvm.sh` or `~/.nvm/nvm.sh`). After installing, it:

1. Sets the installed version as `nvm alias default`
2. Upgrades npm to latest
3. Installs any global tools listed in `npmextra.json` → `npmGlobalTools`

`szci ssh` — SSH Key Deployment

```
szci ssh prepare    # Deploy SSH keys from env vars to ~/.ssh
```

Reads all `SZCI_SSHKEY_*` env vars and writes the keys to disk.

`szci git` — Git Mirroring

```
szci git mirror    # Mirror repository to GitHub
```

Pushes all branches and tags to a GitHub mirror. Requires `SZCI_GIT_GITHUBTOKEN`. Refuses to mirror packages marked as `private` in `package.json`.

⚙ Configuration

`npmextra.json`

Place this in your project root to configure szci behavior:

```
{
  "@ship.zone/szci": {
    "npmGlobalTools": ["typescript", "pnpm"],
    "npmAccessLevel": "public",
    "npmRegistryUrl": "registry.npmjs.org",
    "urlCloudly": "https://cloudly.example.com"
  }
}
```

Option	Type	Default	Description
`npmGlobalTools`	`string[]`	`[]`	Global npm packages to install during `node install`
`npmAccessLevel`	`"public" | "private"`	`"private"`	Access level for `npm publish`
`npmRegistryUrl`	`string`	`"registry.npmjs.org"`	Default npm registry
`urlCloudly`	`string?`	—	Cloudly endpoint URL (can also be set via `SZCI_URL_CLOUDLY`)

□□ Environment Variables

Docker Registry Authentication

Pipe-delimited format: `registry|username|password`

```
SZCI_LOGIN_DOCKER_1="registry.example.com|myuser|mypass"
SZCI_LOGIN_DOCKER_2="ghcr.io|token|ghp_xxxx"
```

In **GitLab CI**, the `CI_JOB_TOKEN` is automatically used for `registry.gitlab.com` — no manual config needed.

NPM Registry Authentication

Pipe-delimited format: `registry|token[|plain]`

```
# Base64-encoded token (default)
SZCI_TOKEN_NPM_1="registry.npmjs.org|dGhlLXRva2VuLWhlcmU="

# Plain text token
SZCI_TOKEN_NPM_2="verdaccio.example.com|the-token-here|plain"
```

SSH Keys

Pipe-delimited format: `host|privKeyBase64|pubKeyBase64`

```
SZCI_SSHKEY_1="github.com|BASE64_PRIVATE_KEY|BASE64_PUBLIC_KEY"
SZCI_SSHKEY_2="gitlab.com|BASE64_PRIVATE_KEY|##"   # Use ## to skip a field
```

Git Mirroring

```
SZCI_GIT_GITHUBTOKEN="ghp_your_personal_access_token"
SZCI_GIT_GITHUBGROUP="your-org"       # Defaults to repo owner
SZCI_GIT_GITHUB="your-repo"           # Defaults to repo name
```

Debugging & Testing

```
DEBUG_SZCI="true"      # Log all shell commands before execution
SZCI_TEST="true"       # Test mode: mocks bash execution, skips SSH disk writes
```

Full Environment Variable Reference

Variable	Purpose
`SZCI_LOGIN_DOCKER_*`	Docker registry credentials (pipe-delimited)
`SZCI_TOKEN_NPM_*`	NPM registry auth tokens (pipe-delimited)
`SZCI_SSHKEY_*`	SSH key pairs (pipe-delimited)

Variable	Purpose
`SZCI_GIT_GITHUBTOKEN`	GitHub personal access token for mirroring
`SZCI_GIT_GITHUBGROUP`	GitHub org/user for mirror target
`SZCI_GIT_GITHUB`	GitHub repo name for mirror target
`SZCI_URL_CLOUDLY`	Cloudly endpoint URL
`SZCI_COMPUTED_REPOURL`	Override auto-detected repo URL
`DEBUG_SZCI`	Enable verbose shell command logging
`SZCI_TEST`	Enable test mode (mock execution)

□□ CI/CD Integration Examples

GitLab CI

```
image: node:22

stages:
  - prepare
  - build
  - test
  - deploy

before_script:
  - curl -sSL https://code.foss.global/ship.zone/szci/raw/branch/main/install.sh | bash

prepare:
  stage: prepare
  script:
    - szci node install stable
    - szci npm install

build:
  stage: build
  script:
    - szci docker build
```

```
test:
  stage: test
  script:
    - szci npm test

deploy:
  stage: deploy
  script:
    - szci docker push $CI_REGISTRY
  only:
    - master
```

> □□ In GitLab CI, `CI_JOB_TOKEN` is automatically detected — szci will login to `registry.gitlab.com` without any extra config.

GitHub Actions

```
name: CI/CD

on:
  push:
    branches: [main]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4

      - name: Install szci
        run: curl -sSL https://code.foss.global/ship.zone/szci/raw/branch/main/install.sh |
sudo bash

      - name: Setup Node.js
        run: szci node install stable

      - name: Install & Build
```

```
        run: |
          szci npm install
          szci npm build

      - name: Test
        run: szci npm test

      - name: Push Docker
        if: github.ref == 'refs/heads/main'
        run: szci docker push ghcr.io
        env:
          SZCI_LOGIN_DOCKER_1: "ghcr.io|${{ github.actor }}|${{ secrets.GITHUB_TOKEN }}"
```

Gitea CI (Woodpecker)

```
steps:
  - name: ci
    image: node:22
    commands:
      - curl -sSL https://code.foss.global/ship.zone/szci/raw/branch/main/install.sh | bash
      - szci node install stable
      - szci npm install
      - szci docker build
      - szci npm test
```

□□ Migration from npmci

Upgrading from `@ship.zone/npmci`? Three steps:

1. **Rename the binary** — replace `npmci` with `szci` in all CI scripts
2. **Rename env vars** — `NPMCI_*` → `SZCI_*` (same formats, just the prefix changed)
3. **Docker ops** — now delegate to `@git.zone/tsdocker` (installed automatically via `npx`)

Old	New
`NPMCI_LOGIN_DOCKER*`	`SZCI_LOGIN_DOCKER*`
`NPMCI_TOKEN_NPM*`	`SZCI_TOKEN_NPM*`
`NPMCI_SSHKEY_*`	`SZCI_SSHKEY_*`

Old	New
`NPMCI_GIT_GITHUBTOKEN`	`SZCI_GIT_GITHUBTOKEN`
`NPMCI_URL_CLOUDLY`	`SZCI_URL_CLOUDLY`
`DEBUG_NPMCI`	`DEBUG_SZCI`
`NPMTS_TEST`	`SZCI_TEST`

□□ Development

```
# Type check
deno task check

# Run tests
deno task test

# Watch tests
deno task test:watch

# Dev mode
deno task dev -- --help

# Compile binaries for all platforms
deno task compile

# Format code
deno task fmt

# Lint
deno task lint
```

License and Legal Information

This repository contains open-source code licensed under the MIT License. A copy of the license can be found in the LICENSE file.

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 or third parties, 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 or the guidelines of the respective third-party owners, and any usage must be approved in writing. Third-party trademarks used herein are the property of their respective owners and used only in a descriptive manner, e.g. for an implementation of an API or similar.

Company Information

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

For any legal inquiries or 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 11:14:45 UTC by foss.global Team
Updated 2026-03-28 12:21:32 UTC by foss.global Team