readme.md for @push.rocks/smartgit

> **Modern Git operations for Node.js** - A powerful TypeScript wrapper around isomorphic-git that makes repository management and analysis a breeze

npm version TypeScript

What is SmartGit?

SmartGit is a sophisticated, promise-based Git toolkit designed for Node.js applications. Built on top of `isomorphic-git`, it provides a clean, intuitive API for repository management, diff analysis, and commit history exploration. Perfect for automation tools, CI/CD pipelines, and any application that needs to interact with Git repositories programmatically.

Key Features

- **Repository Management** - Create, clone, and open repositories with ease
- **Remote Operations** - Manage remotes and push changes effortlessly
- **Diff Analysis** - Get detailed diffs of uncommitted changes
- **Commit History** - Extract commit messages with metadata and version tracking
- ⚡ **Async/Await** - Modern promise-based API throughout
- **TypeScript First** - Full type safety and IntelliSense support
- **Production Ready** - Battle-tested and actively maintained

Installation

```
# Using pnpm (recommended)
pnpm install @push.rocks/smartgit
```

```
# Using npm
npm install @push.rocks/smartgit

# Using yarn
yarn add @push.rocks/smartgit
```

□□ Quick Start

```
import { Smartgit } from '@push.rocks/smartgit';

// Initialize SmartGit
const smartgit = new Smartgit();
await smartgit.init();

// Clone a repository
const repo = await smartgit.createRepoByClone(
  'https://github.com/username/repo.git',
  './local-repo'
);

console.log('□□Repository cloned successfully!');
```

□□ Usage Guide

□□ Repository Operations

Creating a New Repository

```
const repo = await smartgit.createRepoByInit('./my-new-repo');
```

Cloning from Remote

```
const repo = await smartgit.createRepoByClone(
  'https://github.com/octocat/Hello-World.git',
```

```
  './hello-world'
);
```

Opening Existing Repository

```
const repo = await smartgit.createRepoByOpen('./existing-repo');
```

□□ Working with Remotes

List All Remotes

```
const remotes = await repo.listRemotes();
console.log(remotes);
// Output: [{ remote: 'origin', url: 'https://github.com/...' }]
```

Ensure Remote Exists

```
await repo.ensureRemote('origin', 'https://github.com/username/repo.git');
```

Get Remote URL

```
const originUrl = await repo.getUrlForRemote('origin');
console.log(`□□Origin URL: ${originUrl}`);
```

Push to Remote

```
await repo.pushBranchToRemote('main', 'origin');
console.log('□ Changes pushed successfully!');
```

□□ Diff Analysis

Get a detailed diff of uncommitted changes (perfect for code review automation):

```
const diffs = await repo.getUncommittedDiff(['node_modules/*', '*.log']);

diffs.forEach(diff => {
  console.log('□□File changes:');
  console.log(diff);
```

```
});
```

□□ Commit History Analysis

Extract commit history with package.json version tracking:

```
const history = await repo.getAllCommitMessages();

history.forEach(commit => {
  console.log(`□□${commit.date} | v${commit.version} | ${commit.message}`);
});

// Example output:
// □□2024-01-15 | v1.2.3 | feat: add new authentication method
// □□2024-01-14 | v1.2.2 | fix: resolve memory leak in parser
```

□□ Advanced Usage

Error Handling

```
try {
  const repo = await smartgit.createRepoByClone(invalidUrl, './test');
} catch (error) {
  console.error('□ Clone failed:', error.message);
}
```

Working with Multiple Repositories

```
const smartgit = new Smartgit();
await smartgit.init();

const repositories = [
  await smartgit.createRepoByOpen('./repo1'),
  await smartgit.createRepoByOpen('./repo2'),
```

```
  await smartgit.createRepoByOpen('./repo3')
];

// Analyze all repositories
for (const repo of repositories) {
  const remotes = await repo.listRemotes();
  console.log(`□□Repository has ${remotes.length} remotes`);
}
```

□□ Architecture

SmartGit is built with a clean, modular architecture:

- `Smartgit` - Main entry point and repository factory
- `GitRepo` - Individual repository operations and analysis
- `isomorphic-git` - Underlying Git implementation
- **Environment Detection** - Automatic Node.js environment setup

□□ Use Cases

- □□ **Automation Scripts** - Automate repository management and analysis
- □□ **CI/CD Pipelines** - Repository operations in build processes
- □□ **Code Analysis Tools** - Extract commit data and diff analysis
- □□ **Developer Tools** - Build Git-powered development utilities
- □□ **Release Management** - Track versions and generate changelogs

□□ Requirements

- **Node.js** 16+ (Browser support not available)
- **Git repository** access (for clone operations)
- **TypeScript** 4+ (recommended)

□□ API Reference

Smartgit Class

Method	Description	Returns
`init()`	Initialize the SmartGit instance	`Promise<void>`
`createRepoByInit(dir)`	Create new repository	`Promise<GitRepo>`
`createRepoByClone(url, dir)`	Clone repository	`Promise<GitRepo>`
`createRepoByOpen(dir)`	Open existing repository	`Promise<GitRepo>`

GitRepo Class

Method	Description	Returns
`listRemotes()`	List all remotes	`Promise<{remote: string, url: string}[]>`
`ensureRemote(name, url)`	Ensure remote exists	`Promise<void>`
`getUrlForRemote(name)`	Get URL for remote	`Promise<string>`
`pushBranchToRemote(branch, remote)`	Push branch to remote	`Promise<void>`
`getUncommittedDiff(excludeFiles?)`	Get uncommitted changes	`Promise<string[]>`
`getAllCommitMessages()`	Get commit history	`Promise<CommitInfo[]>`

□□ Pro Tips

- □□ **Always call** `init()` before using any repository operations
- □□ **Use absolute paths** for repository directories when possible
- □□ **Exclude large files** from diff analysis using the `excludeFiles` parameter
- ⚡ **Batch operations** when working with multiple repositories
- □□ **Handle errors gracefully** - network issues can cause clone operations to fail

License and Legal Information

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.

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 11:10:38 UTC by foss.global Team
Updated 2026-03-28 12:17:27 UTC by foss.global Team