**📁 .husky/ – Git Hooks Automation in Detail**

**🔍 What is .husky/?**

.husky/ is a directory where **Git hooks** (scripts that run at various stages of Git operations) are stored in a **project-managed** way. It’s a **tool that automates running scripts** like linters, formatters, tests, or commit message validations **before committing or pushing code**.

🧠 Git hooks exist in .git/hooks/, but they aren’t tracked by Git. Husky solves that by allowing version-controlled hooks in .husky/.

**✅ Common Use Cases**

| **Hook** | **Purpose** | **Example Command** |
| --- | --- | --- |
| pre-commit | Validate code before commit | Linting, formatting, type-check |
| pre-push | Prevent broken code from being pushed to remote | Run tests, build |
| commit-msg | Validate commit messages (e.g., Conventional Commits) | Check format, reject invalid commits |

**🧱 Detailed Folder Structure**

.husky/

├── pre-commit # Hook triggered before a commit

├── pre-push # Hook triggered before a push

├── commit-msg # Hook triggered on commit message input

└── \_/

└── husky.sh # Common shell bootstrapper (auto-generated)

**🔧 Step-by-Step Explanation**

**1. \_ (Underscore) Folder**

* **File: .husky/\_/husky.sh**
* Auto-generated script for hook initialization
* Loads the proper environment for Git hook execution

bash

CopyEdit

# .husky/\_/husky.sh

#!/bin/sh

# Husky common shell setup

if [ -z "$husky\_skip\_init" ]; then

debug () {

[ "$HUSKY\_DEBUG" = "1" ] && echo "husky (debug) - $1"

}

readonly hook\_name="$(basename "$0")"

debug "starting $hook\_name..."

if [ -f ~/.huskyrc ]; then

debug "sourcing ~/.huskyrc"

. ~/.huskyrc

fi

fi

⚠️ Don’t modify this manually unless you know what you’re doing.

**🔍 Why it's important:**

* Ensures **reusability** for all hooks.
* Common bootstrapping logic.
* Avoids repeating shell setup logic in each hook file.

**2. .husky/pre-commit**

Runs **before** a git commit is made. Usually used to:

* Run **linters** (eslint)
* Check **formatting** (prettier)
* Perform **type checks** or custom validations

**✅ Real Example:**

#!/bin/sh

. "$(dirname "$0")/\_/husky.sh"

echo "✅ Running Pre-Commit Checks..."

# Format code

npx prettier --check "src/\*\*/\*.{ts,js,json}"

# Lint code

npx eslint "src/\*\*/\*.{ts,js}" --fix

**3. .husky/pre-push**

Runs **before** git push. Commonly used to:

* Run **tests**
* Prevent pushing failing builds
* Run **type-checking** with tsc

**✅ Real Example:**

#!/bin/sh

. "$(dirname "$0")/\_/husky.sh"

echo "🚀 Running Pre-Push Checks..."

# Run tests

npm run test

# Optional: Validate build

npm run build

This prevents broken code or failing tests from being pushed accidentally.

**4. .husky/commit-msg**

Runs after commit message is entered. You can enforce rules like:

* Must follow [Conventional Commit](https://www.conventionalcommits.org/) style
* Validate commit message with commitlint

**✅ Real Example:**

#!/bin/sh

. "$(dirname "$0")/\_/husky.sh"

echo "📝 Validating Commit Message..."

npx commitlint --edit "$1"

📦 Required dev dependencies:

npm install --save-dev @commitlint/{cli,config-conventional}

Then add commitlint.config.js:

module.exports = {

extends: ['@commitlint/config-conventional'],

};

**🛠️ How to Set It Up in a NestJS Project**

**🛠️ Setup Commands (Once)**

1. **Install Husky**:

bash

CopyEdit

# Install husky

npm install husky --save-dev

# Initialize Git hooks

npx husky install

**2. Enable Git Hooks**

npx husky install

Add to your package.json:

"scripts": {

"prepare": "husky install"

}

**3. Add Hooks**

npx husky add .husky/pre-commit "npm run lint"

npx husky add .husky/pre-push "npm run test"

npx husky add .husky/commit-msg "npx commitlint --edit \$1"

**🧠 Best Practices**

| **Best Practice** | **Why** |
| --- | --- |
| Keep hooks fast | Developers hate slow commits |
| Use lint-staged | Only lint files that are staged for commit |
| Version-control hooks | Everyone in the team uses the same rules |
| Use --no-verify for emergency bypass | For rare, urgent pushes/commits |

**🧪 Optional Enhancement: lint-staged**

This speeds up pre-commit by only checking staged files:

npm install lint-staged --save-dev

Add to package.json:

"lint-staged": {

"\*.ts": [

"eslint --fix",

"prettier --write"

]

}

Modify .husky/pre-commit:

#!/bin/sh

. "$(dirname "$0")/\_/husky.sh"

npx lint-staged

**📌 Summary**

* .husky/ brings your Git hooks under version control.
* You can automate code checks to ensure quality and consistency.
* Hooks improve productivity and prevent silly mistakes from reaching production.