About Git subtree merges

In this article

If you need to manage multiple projects within a single repository, you can use a *subtree merge* to handle all the references.

Mac Windows Linux

About subtree merges

Typically, a subtree merge is used to contain a repository within a repository. The "subrepository" is stored in a folder of the main repository.

The best way to explain subtree merges is to show by example. We will:

- Make an empty repository called `test` that represents our project
- Merge another repository into it as a subtree called `Spoon-Knife`.
- The `test` project will use that subproject as if it were part of the same repository.
- Fetch updates from `Spoon-Knife` into our `test` project.

Setting up the empty repository for a subtree merge

1. Open TerminalTerminalGit Bash.
2. Create a new directory and navigate to it.

```
mkdir test
cd test
```

3. Initialize a new Git repository.

```
$ git init
> Initialized empty Git repository in /Users/octocat/tmp/test/.git/
```

4. Create and commit a new file.

```
$ touch .gitignore
$ git add .gitignore
```

```
$ git commit -m "initial commit"
> [main (root-commit) 3146c2a] initial commit
>  0 files changed, 0 insertions(+), 0 deletions(-)
>  create mode 100644 .gitignore
```

Adding a new repository as a subtree

1. Add a new remote URL pointing to the separate project that we're interested in.

```
$ git remote add -f spoon-knife https://github.com/octocat/Spoon-Knife.git
> Updating spoon-knife
> warning: no common commits
> remote: Counting objects: 1732, done.
> remote: Compressing objects: 100% (750/750), done.
> remote: Total 1732 (delta 1086), reused 1558 (delta 967)
> Receiving objects: 100% (1732/1732), 528.19 KiB | 621 KiB/s, done.
> Resolving deltas: 100% (1086/1086), done.
> From https://github.com/octocat/Spoon-Knife
>  * [new branch]      main     -> Spoon-Knife/main
```

2. Merge the `Spoon-Knife` project into the local Git project. This doesn't change any of your files locally, but it does prepare Git for the next step.

   If you're using Git 2.9 or above:

```
$ git merge -s ours --no-commit --allow-unrelated-histories spoon-knife/main
> Automatic merge went well; stopped before committing as requested
```

   If you're using Git 2.8 or below:

```
$ git merge -s ours --no-commit spoon-knife/main
> Automatic merge went well; stopped before committing as requested
```

3. Create a new directory called **spoon-knife**, and copy the Git history of the `Spoon-Knife` project into it.

```
$ git read-tree --prefix=spoon-knife/ -u spoon-knife/main
> fatal: refusing to merge unrelated histories
```

4. Commit the changes to keep them safe.

```
$ git commit -m "Subtree merged in spoon-knife"
> [main fe0ca25] Subtree merged in spoon-knife
```

Although we've only added one subproject, any number of subprojects can be incorporated into a Git repository.

> **Tip**: If you create a fresh clone of the repository in the future, the remotes you've added will not be created for you. You will have to add them again using the `git remote add` command.

Synchronizing with updates and changes

When a subproject is added, it is not automatically kept in sync with the upstream changes. You will need to update the subproject with the following command:

```
git pull -s subtree REMOTE-NAME BRANCH-NAME
```

For the example above, this would be:

```
git pull -s subtree spoon-knife main
```

Further reading

- The "Advanced Merging" chapter from the *Pro Git* book
- "How to use the subtree merge strategy"