Adding a New Subproject

Submodule

```
git submodule add https://github.com/githubtraining/example-submodule

git commit -m "adding new submodule"
```

The `submodule add` command adds a new file called `.gitmodules` along with a subdirectory containing the files from `example-submodule`. Both are added to your index (staging area) and you simply need to commit them. The submodule's history remains independent of the parent project.

Subtree

```
git subtree add --prefix=example-submodule https://github.com/githubtraining/example-submodule master --squash
```

The subtree command adds a subdirectory containing the files from `example-submodule`. The most common practice is to use the `--squash` option to combine the subproject's history into a single commit, which is then grafted onto the existing tree of the parent project. You can omit the `--squash` option to maintain all of the history from the designated branch of the subproject.

Viewing a Diff of the Subproject

Submodule

To view a diff of the submodule:

```
# show changes to the submodule commit
git diff example-submodule
# show oneline log of new commits in the submodule
git diff --submodule example-submodule
# show changes to the files in the submodule
git diff --submodule=diff
```

Subtree

No special command required

Cloning a Repository with a Subproject

Submodule

To clone a repository along with its submodules:

```
git clone --recurse-submodules URL
```

If you forgot `--recurse-submodules`, you can clone and initialize all submodules:

Subtree

No special command required

```
git submodule update --init --recursive
```

Adding `--recursive` is only required if any submodule itself has submodules.

Pulling in Superproject Updates

Submodule

By default, the submodule repository is fetched, but not updated when you run `git pull` in the superproject. You need to use `git submodule update`, or add the `--recurse-submodules` flag to `pull`:

```
git pull
git submodule update --init --recursive
# or, in one step (Git >= 2.14)
git pull --recurse-submodules
```

`--init` is required if the superproject added new submodules, and `--recursive` is needed if any submodule itself has submodules.

If ever the superproject changes the URL of the submodule, a separate command is required:

```
# copy the new URL to your local config
git submodule sync --recursive
# update the submodule from the new URL
git submodule update --init --recursive
```

`--recursive` is only needed if any submodule itself has submodules.

Subtree

No special command required

Changing branches

Submodule

By default, the submodule working tree is not updated to match the commit recorded in the superproject when changing branches. You need to use `git submodule update`, or add the `--recurse-submodules` flag to `checkout`:

```
git checkout <branch>
git submodule update --recursive
# or, in one step (Git >= 2.13)
git checkout --recurse-submodules <branch>
```

Subtree

No special command required

Pulling in Subproject Updates

Submodule

```
# Update the submodule repository
git submodule update --remote
# Record the changes in the superproject
git commit -am "Update submodule"
```

If you have more than one submodule, you can add the path to the submodule at the end of the `git submodule update --remote` command to specify which subproject to update.

By default, `git submodule update --remote` will update the submodule to the latest commit on the `master` branch of the submodule remote.

You can change the default branch for future calls with:

```
# Git >= 2.22
git submodule set-branch other-branch
# or
```

Subtree

```
git subtree pull --prefix=example-submodule https://github.com/githubtraining/example-submodule master --squash
```

You can shorten the command by adding the subtree URL as a remote:

```
git remote add sub-remote https://github.com/githubtraining/example-submodule.git
```

You can add/pull from other refs by replacing `master` with the desired ref (e.g. `stable`, `v1.0`).

Making Changes to a Subproject

In most cases, it is considered best practice to make changes in a separate clone of the subproject repository and pull them in to the parent project. When this is not practical, follow these instructions:

Submodule

Access the submodule directory and create a branch:

```
cd example-submodule
git checkout -b branch-name master
```

Changes require two commits, one in the subproject repository and one in the parent repository. Don't forget to push in both the submodule and the superproject!

Subtree

No special command required, changes will be committed on the parent project branch.

It is possible to create commits mixing changes to the subproject and the parent project, but this is generally discouraged.

Pushing Changes to the Subproject Repository

Submodule

While in the submodule directory:

```
git push
```

Or while in the parent directory:

```
git push --recurse-submodules=on-demand
```

Subtree

```
git subtree push --prefix=example-submodule https://github.com/githubtraining/example-submodule master
```

Helpful Configs for Submodules

Always show the submodule log when you diff:

```
git config --global diff.submodule log
```

Show a short summary of submodule changes in your `git status` message:

```
git config --global status.submoduleSummary true
```

Make `push` default to `--recurse-submodules=on-demand` :

```
git config --global push.recurseSubmodules on-demand
```

Make all commands (except `clone`) default to `--recurse-submodules` if they support the flag (this works for `git pull` since Git 2.15):

```
git config --global submodule.recurse true
```