What's the best practice for putting multiple projects in a git repository?

Solution 3

This is for using a single directory for multiple projects. I use this technique for some closely related projects where I often need to pull changes from one project into another. It's similar to the orphaned branches idea but the branches don't need to be orphaned. Simply start all the projects from the same empty directory state.

Start all projects from one committed empty directory

Don't expect wonders from this solution. As I see it, you are always going to have annoyances with untracked files. Git doesn't really have a clue what to do with them and so if there are intermediate files generated by a compiler and ignored by your .gitignore file, it is likely that they will be left hanging some of the time if you try rapidly swapping between - for example - your software project and a PH.D thesis project.

However here is the plan. Start as you ought to start any git projects, by committing the empty repository, and then start all your projects from the same empty directory state. That way you are certain that the two lots of files are fairly independent. Also, give your branches a proper name and don't lazily just use "master". Your projects need to be separate so give them appropriate names.

Git commits (and hence tags and branches) basically store the state of a directory and its subdirectories and **Git has no idea whether these are parts of the same or different projects** so really there is no problem for git storing different projects in the same repository. The problem is then for you clearing up the untracked files from one project when using another, or separating the projects later.

Create an empty repository

```
cd some_empty_directory
git init
touch .gitignore
git add .gitignore
git commit -m empty
git tag EMPTY
```

Start your projects from empty.

Work on one project.

```
git branch software EMPTY
git checkout software
echo "array board[8,8] of piece" >
chess.prog

git add chess.prog
git commit -m "chess program"
```

Start another project

whenever you like.

```
git branch thesis EMPTY
git checkout thesis
echo "the meaning of meaning" >
philosophy_doctorate.txt
git add philosophy_doctorate.txt
git commit -m "Ph.D"
```

Switch back and forth

Go back and forwards between projects whenever you like. This example goes back to the chess software project.

```
git checkout software
echo "while not end_of_game do make_move()" >>
chess.prog
git add chess.prog
git commit -m "improved chess program"
```

Untracked files are annoying

You will however be annoyed by untracked files when swapping between projects/branches.

```
touch untracked_software_file.prog
git checkout thesis
ls
    philosophy_doctorate.txt
untracked_software_file.prog
```

It's not an insurmountable problem

Sort of by definition, git doesn't really know what to do with untracked files and it's up to you to deal with them. You can stop untracked files from being carried around from one branch to another as follows.

```
git checkout EMPTY
ls
    untracked_software_file.prog
rm -r *
    (direcory is now really empty, apart from the repository
stuff!)
git checkout thesis
ls
    philosophy_doctorate.txt
```

By ensuring that the directory was empty before checking out our new project we made sure there were no hanging untracked files from another project.

A refinement

```
$ GIT_AUTHOR_DATE='2001-01-01:T01:01:01' GIT_COMMITTER_DATE='2001-01-01T01:01:01' git commit -m empty
```

If the date is specified when committing the empty repository, then the commits can have the same SHA1 code. This allows two repositories to be created independently and then merged together into a single tree with a common root in one repository later.

Example

```
mkdir single_repo_for_thesis_and_chess
cd single_repo_for_thesis_and_chess
git init
touch .gitignore
git add .gitignore
GIT_AUTHOR_DATE='2001-01-01:T01:01:01' GIT_COMMITTER_DATE='2001-01-01:T01:01:01' git commit -m empty
git tag EMPTY
echo "the meaning of meaning" > thesis.txt
git add thesis.txt
git commit -m "Wrote my PH.D"
git branch -m master thesis
git remote add chess ../chessrepository/.git
git fetch chess chess:chess
```

Result