Paytm

Version Control System

Yash Bedi
Software Engineer-iOS

Outline

- Definition
- Available Softwares for Source Control
- What is the need of Version Control/Source Control ?
  - Benefits
- Basic Commands
- Leveraging the power of git .diff files.
  - Benefits
- Intermediate Commands
  - Extending rebase with an example.
- Git tags + Benefits
- Best Practices
  - While doing a commit
  - DO NOT commit
- Tools and Apps
- Miscellaneous
- Appendix

Paytm

The Definition

A system that records changes to
a file
over time,
so you can recall
specific versions later.

Available Softwares for Source Control

- Git (Linus Torvalds - 2005)
- Apache Subversion
- Mercurial
- Azure DevOps Server
- Concurrent Versions Systems
- Microsoft Visual SourceSafe
- IBM's Rational ClearCase
- Adobe Version Cue

What’s the need of Version Control aka Source Control ?

Discussions.!!!

Benefits :

- Backup and Restore
- Sync with multiple computers
- Working in a team
- Safely create and test new features
- Ownership + Credits + Blame

Basic Commands of Git

- `git init <project name>` – Initialises a new repo.
- `git status` – Gives you list of modified/added files locally.
- `git add <filename>` – Adds the file to the staging area.
- `git commit -m "message"` – Commits all files added to staging area.
- `git diff <filename>` – Lists the changes made to the file when compared to the HEAD version.
- `git checkout [-b] <branch name>` – Switches the branch. The `-b` parameter will create a new branch.
- `git stash` – Backs up any local changes made.
- `git stash apply` – Restores the most recently backed up files.
- `git pull` – Pulls the latest changes from the remote repo.
- `git reset HARD` – Removes all the local changes made to file.
- `git push origin` – Pushes local changes to remote repo.
- `git rm <file name>` – Used to remove a file from a Git repo.

Discussions.!!!

Leveraging the power of git .diff files !!!

Benefits :

- A patch is a small file that indicates the changes made in a repository.
- It's generally used when someone from outside your team has read-only access but had a good code change available. He then creates a patch and sends it to you.
- You apply it and push it to the git repository.
- Everyone then benefits from the updated version, and the author of the patch didn't need read/write access.
- Also to setup a testing environment with just couple of clicks.

Demo (Sort off..)!!!

Intermediate Commands:

- `git tag <tagname>` – Tags are ref's that point to specific points in git history.
- `git merge` – Join two or more histories together.
- `git cherry-pick` – Picking arbitrary commits by reference and appended to current working head.
- `git rebase -i ~<number of commits/commit hash>` – Combining commits to make `git log` more concise. `-i` is to make the rebase interactive
  below are the commands:
  - `p , pick` - use commit
  - `e , edit` - use commit but stop for amending
  - `d , drop` - remove commit
  - `s , squash` - use commit, but meld into previous commit.

Example for `git rebase`:

```
* 44d2565 (HEAD -> home_page, master) Create nav border
* 1bd999c Styling user data WIP. Add margin
* a135e04 Implement search input for user display
* 76e5811 Add padding to nav list items
* 97178bd User data display WIP
* c289adc Create div for user data
* 14a64ff Add background color to nav and body
* 83c9bbb Add list items to nav
* 60042c8 Add nav to application
* c0fa78d Initial commit
```

Example for `git rebase`:

```
git rebase -i HEAD~9
```

or

```
git rebase -i HEAD 60042c8
```

Example for `git rebase` contd.:

After using `git rebase` the new commit history looks like this :

```
* ca22552 (HEAD -> home_page) Add home page to website
* c0fa78d Initial commit
```

Discussions.!!!

Benefits of using `git tags`:

Git tags are really useful to archive code base related to a specific release version.

Suppose, you are the delivery manager to finally push the code to Heroku Continuous Integration (Heroku CI) server using `git push heroku master` and finally deploy that using `heroku open`.

Congratulations!!🤝🤝 your have successfully deployed `v0.0.1`.

Wait🧐🧐, do you need a backup of this code base?

Because next time when you will publish `v0.0.2` and so on, it will be hard to hop from one commit to another to get the production code base for each version. Using `tag`s can help you a lot.

So, when you deploy a version of your application, you can archive that codebase by creating a new tag.

Discussions.!!!

Paytm

Best Practices !!!

While doing a `commit` :

- Fixing two bugs = Two separate commits.
- Small commits = Easy to [understand, roll back], less conflict.
- Write a good commit message, think twice before hitting the commit button.
- Imperative present tense: “added” -> ❌ , “adding” -> ❌, “add” -> ✔️.
- Limit commit messages to ~50 words.
- Always think these 3 before commiting: **Why ? How ? What ?**
- Follow **YAGNI** principle.
- Always review your code before the commit.

While doing a `commit` contd. with an eg.:

If applied this commit will _____

- Refactor subsystem X for readability
- Update getting started documentation
- Remove deprecated methods
- Release version 1.0.0
- Merge pull request #123 from user/branch

DO NOT `commit` ❌:

- Broken code
- Something that's half done
- Untested work
- A piece of code which is to be sent in a different PR or commit, do not club it with the current one, if your team is not going to need it now.

A diff will tell you
***what* changed,**
but only
commit message can properly tell you
***why*.**

- Peter Hutterer

Tools & Apps

- gityapp.com
- git-tower.com
- sourcetreeapp.com
- gitboxapp.com
- Github Desktop
- TortoiseGit

Miscellaneous

- Never push a keyphrase, password, key, secret etc. to Remote. ❌
- Close all comments BEFORE getting your PR merged
- Prefer merging branch1 to branch2 or vice-versa. Taking pull from a different branch to your current branch will attempt to merge remote's master to yours.
- Use `git log --oneline` instead of `git log`.
- You can filter commits even by date, author, time, file content etc.
- Deleting multiple branches: `git branch -d branch1 branch2 branch3`
- Discussing Size issues, solving Merge conflicts.

Appendix:

https://blog.sourcetreeapp.com/2013/03/28/sourcetree-for-mac-1-6-beta/
https://www.atlassian.com/git/tutorials/inspecting-a-repository/git-tag
https://medium.com/@slamflipstrom/a-beginners-guide-to-squashing-commits-with-git-rebase-8185cf6e62ec
https://en.wikipedia.org/wiki/Version_control
https://www.atlassian.com/git/tutorials/cherry-pick
https://stackoverflow.com/questions/8279602/what-is-a-patch-in-git-version-control
https://www.atlassian.com/git/tutorials/what-is-git
https://superuser.com/questions/163033/pull-for-another-git-branch-without-switching
https://hackernoon.com/ten-useful-git-log-tricks-7nt3yxy
https://medium.com/@sauvik_dolui/a-few-git-tricks-tips-b680c3968a9b

Thank you!

Yash Bedi
yash.bedi@paytm.com