Day 5
Introduction to Git and GitHub

Presented By Kanjal Dalal

Topics covered today

- Breeze through the topics covered yesterday - git log, git tag, branching, merging, rebasing
- Issues
- Forking a repository
- PR’s - pull requests
- Git Aliases
- .gitignore file
- Good practices & mistakes

Git log

- Used to get information about the previous commits.
- git log --all
- git log --oneline

Tagging

- They are essentially bookmarks.
- They are used to mark important points so that the users of the repository can easily visit and navigate through important parts of the history.
- Most importantly used to keep track of releases/versioning.
- git tag -a v1.7.0 -m “this tag is for version 1.7.0”
- To visit tag

  git checkout v.1.7.0

  Refer the doc i shared yesterday for more common usage.

Branching

- Non-linear Development
- To create a branch

1) git branch temp
2) git checkout temp

Or you can do the above two commands using one command itself

- git checkout -b temp

Merging

- git merge branch_name (branch_name is the branch that needs to be merged into the current branch you are in)
- Eg -
  1) git status (verify you are in master)
  2) git checkout -b temp (checkout and create a new branch temp, change the name if already present)
  3) make some change - touch temp.txt (create a temp file)
  4) add and commit -
  git add .
  git commit -m “temp file created”

Merging

5) go back to master - git checkout master

6) view the commit history of master -

git log --oneline

you would notice the temp commit you made is not visib, you can aso verify by now searching for that temp file, it won't be present.

7) merge the temp branch into master -

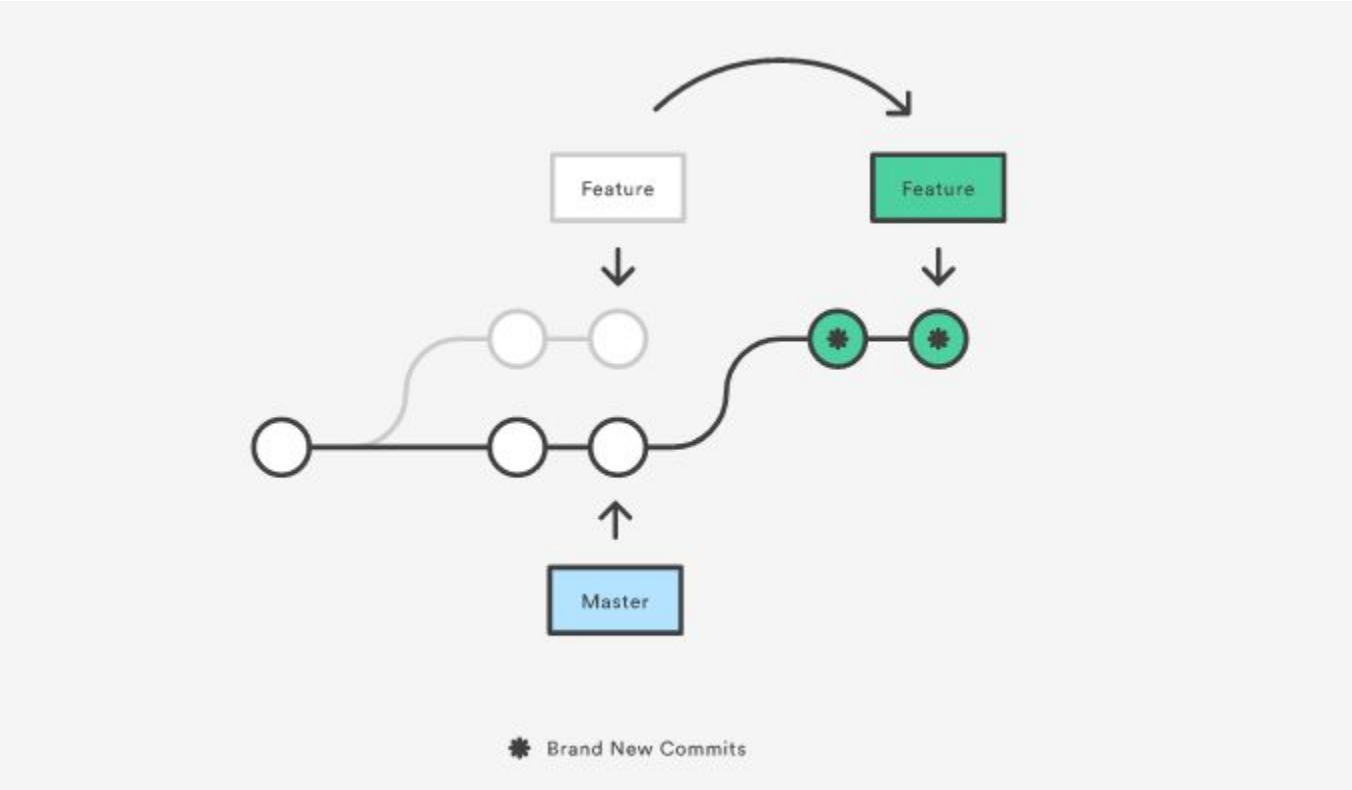
git merge temp

8) delete the temp branch after merge if its not required any more -

git branch -d temp

Note - Now incase when making changes in the temp branch the master branch too has some changes in that case merge conflicts could arise and you would need to resolve them.

Rebasing



QUIZ!

Issues

- Issues can be created on GitHub for a repository to manage your work
- It is generally used to track ideas, bugs, enhancements or tasks
- If you're a project maintainer, you can assign the issue to someone, add labels to help people better search for issues they want to work on, etc.
- Lets create one

Forking a repository

- Fork is a copy of a repository
- Mostly used to propose changes in someone else’s repository or use someone else’s repository as a starting point.
- Lets fork the trial repository that was just created.

Pull Requests

- It is used to let others working on the project to know you have made some change / added a feature and would now like to merge the change in the main repository
- It allows the maintainer/owner of the repository to review you change and ask you to make changes after which it would be merged.
- Pull request have different states mainy -> open, draft, ready for review, closed, merged.
- Lets create a pull request.

Git Aliases

- It is used to make life simpler
- They are just shorter or other names that you can map to the commands to use instead.
- You have to ensure that the alias you use isn’t already a command in git.
- Eg -
  git config --global alias.st status
  Now you can use
  git st instead of git status

  git config --global alias.listco ‘log --oneline’
  Now you can use git listco

.gitignore file

- It is a plain text file that species the files or folders or their pattern which is to be ignored by git.
- This is there so you do not add and commit files which are not to be present in git. Like compiled code, dependencies, system files, file containing api keys, etc.
- How to create one? Simply create a file named .gitignore in your repository.
- It won’t ignore files which you have already staged or commited, you have to add the file names before staging.
- Lets do this for the repository we just created.

Good Practices and mistakes

- Protect master branch from direct commits - create branch add a feature and then merge it.
- Avoid making commits with the wrong email id (being a unrecognized committer).
- Aways separate secret credentials like api keys etc from the source code and make sure not to push them.
- Avoid committing dependencies into your project.
- Create meaningful .gitignore files.
- Keep commit messages meaningful but short.

Good Practices and mistakes

- Made an incorrect commit message. Use -
  git commit --amend
- Created a branch with incorrect name. Use -
  git branch -m old_name new_name
- Always test once you are done with the feature/change to check to see nothing broke in the code. Ie - no bug got introduced
- Undo changes made in a commit
  git log --oneline
  Note down the hash of the commit you want to undo
  git revert {hash_of_that_commit}

QUIZ TIME

Thank You