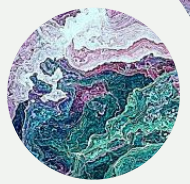
Tutorial on Git

TUSK Machine Learning Operations Course

Keerat Kaur Guliani



What we expect to cover

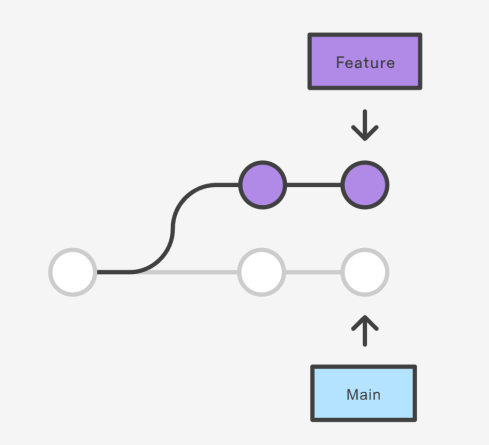
- Version control system
- What is git
- Important git commands
- Undoing git commits and changes
- Collaborating using git
- Making pull requests
- Using branches
- Git Workflow – Gitflow Workflow
- References

Building a version control system

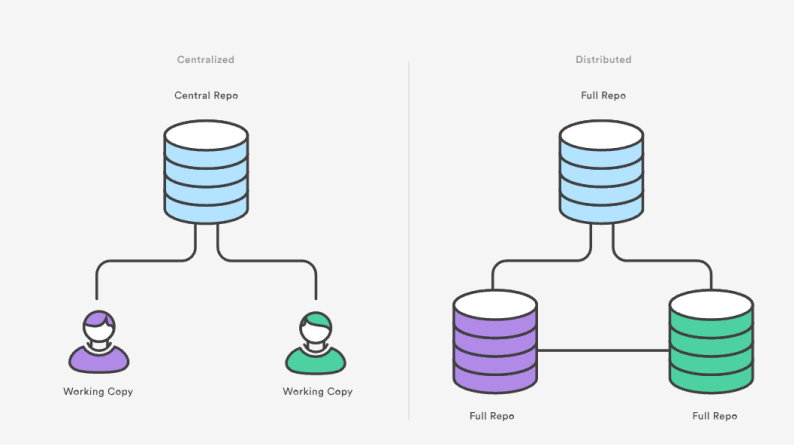
- What is version control
- What kind of features would you want in a version control software?
  - Keep track of all features
  - Should allow you to backtrack to earlier versions of the code while causing minimum disruption to the team
  - Team members can work on different parts of the source code at the same time – check for conflicts before merging
  - Supports a developer's preferred workflow without imposing one particular way of working
  - Should work on all platforms and operating systems

What is git

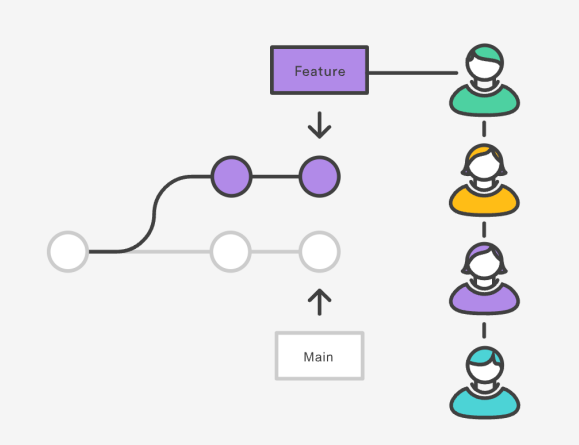
- Version control system that offers performance, security and flexibility



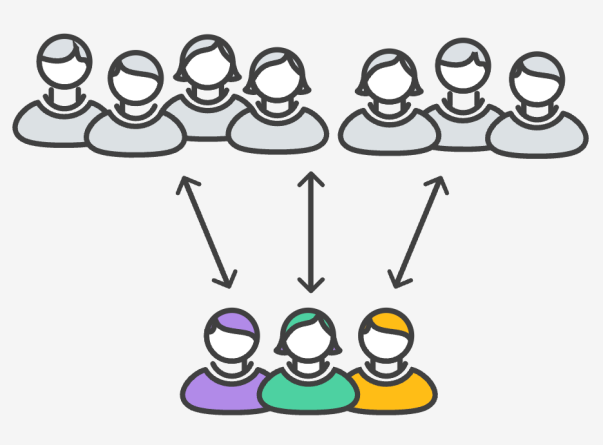
Feature branch workflow



Distributed development



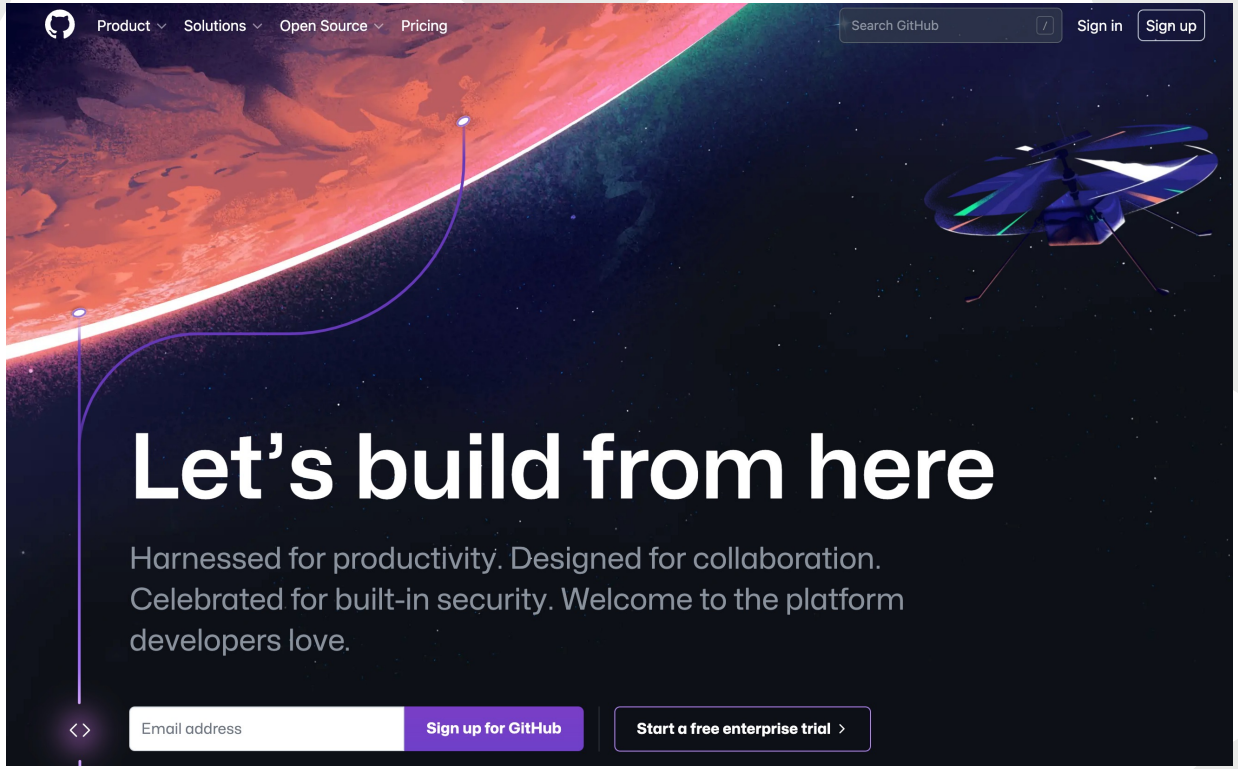
Pull Requests



Community

Output: faster release cycle facilitating an agile workflow

Getting Started – Creating an account on github + git Installation

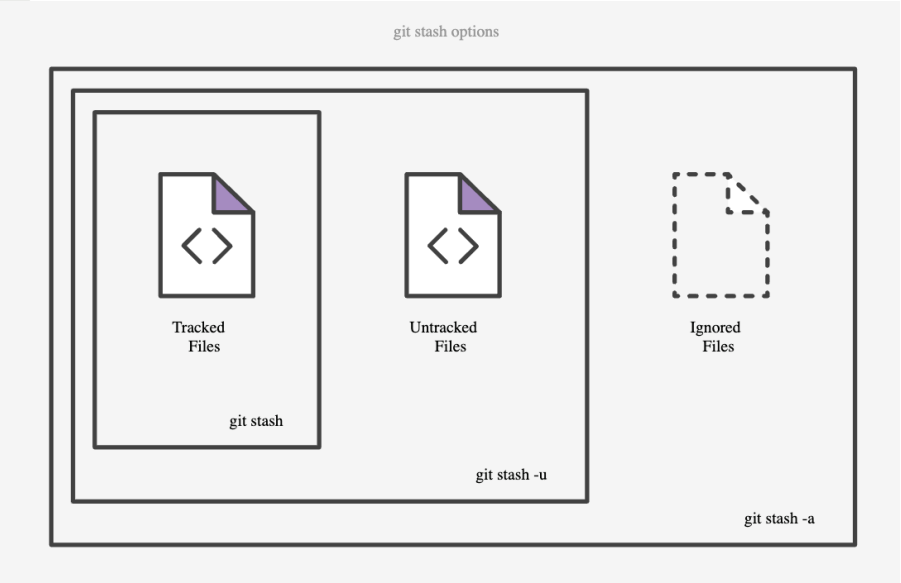


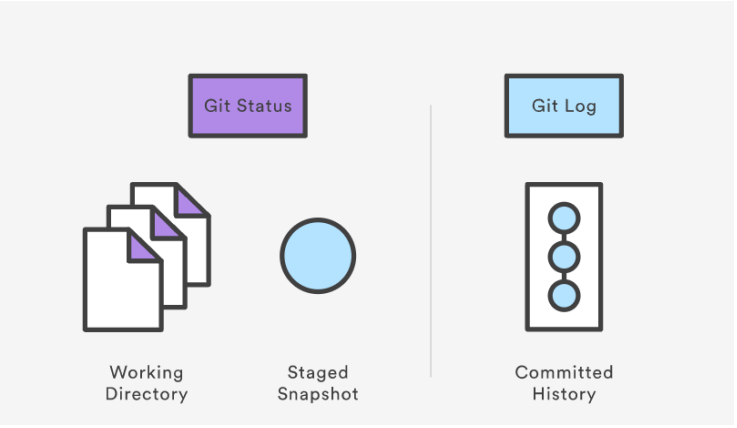
If you want to use git from the command line -> Installation procedure dependent on your operating system:

Refer https://www.atlassian.com/git/tutorials/install-git

Important git commands

- A git repo is a virtual storage of your project. It allows you to save versions of your code, which you can access when needed.
- Initializing a new repo: git init
- Cloning an existing repo: git clone <repo url>
- Make changes and add to staging area - git add
- Save changes to integrate in future – git commit –m “commit-message”
- Save changes to return to them in the future – git stash
- Examine the result of any command, state of working and staging area – git status
- Push changes to remote repo for collaboration – git push





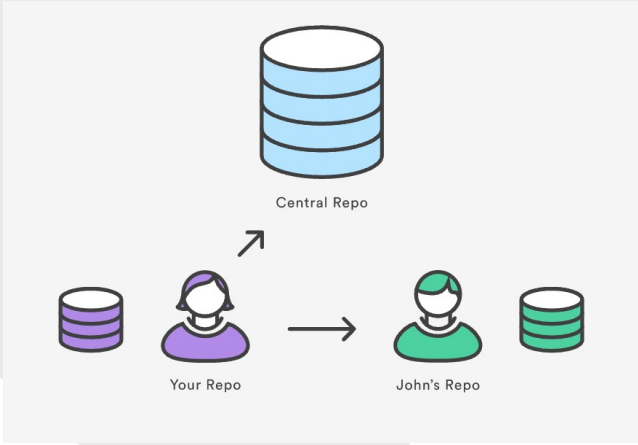
Undoing Commits and Changes

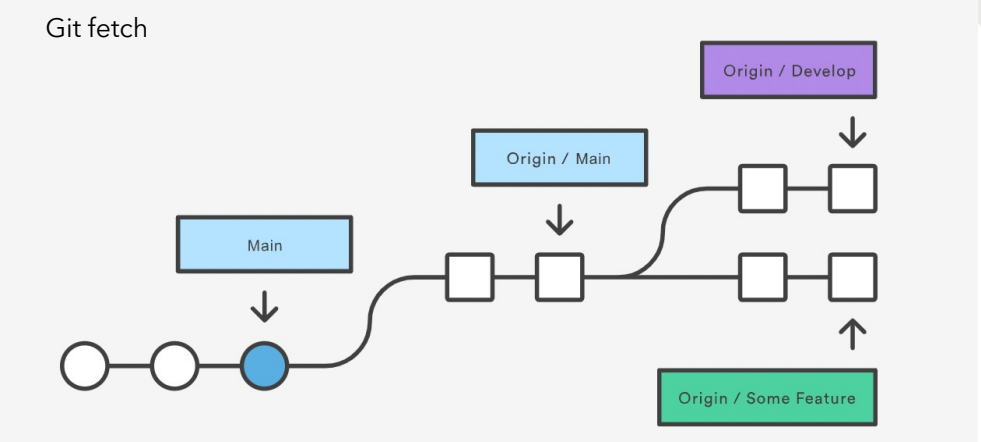
- To examine a previous commit - git checkout <commit name>
- To undo a commit with git checkout – git checkout -b <new_branch_without_crazy_commit>
- To undo a commit with git revert – git revert HEAD
- Hard reset to a specific commit – git reset <commit-id>
- To undo changes in the working directory – git clean/reset
- To undo changes in the staging area – git reset
- To remove files from a git repo – git rm (inverse of git add)

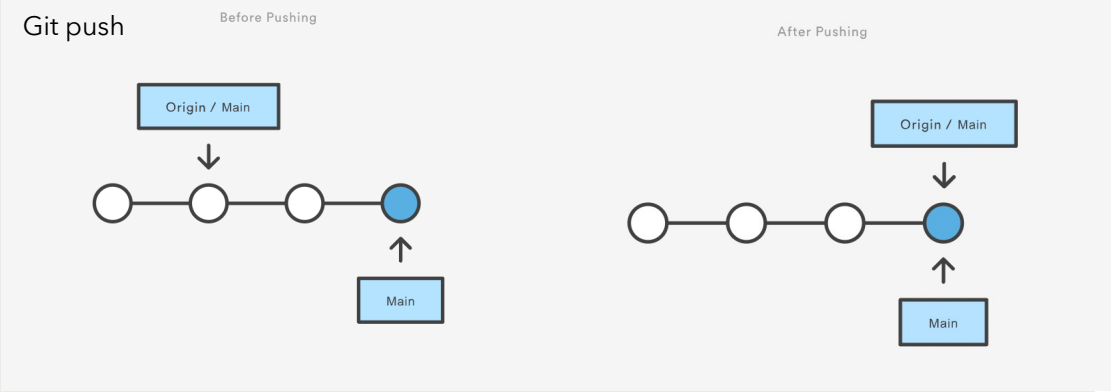
NOTE: The preferred method of undoing shared history is git revert. It is safer than a reset because it will not remove any commits from a shared history. Git reset should generally be considered a 'local' undo method, for example, when undoing changes to a private branch.

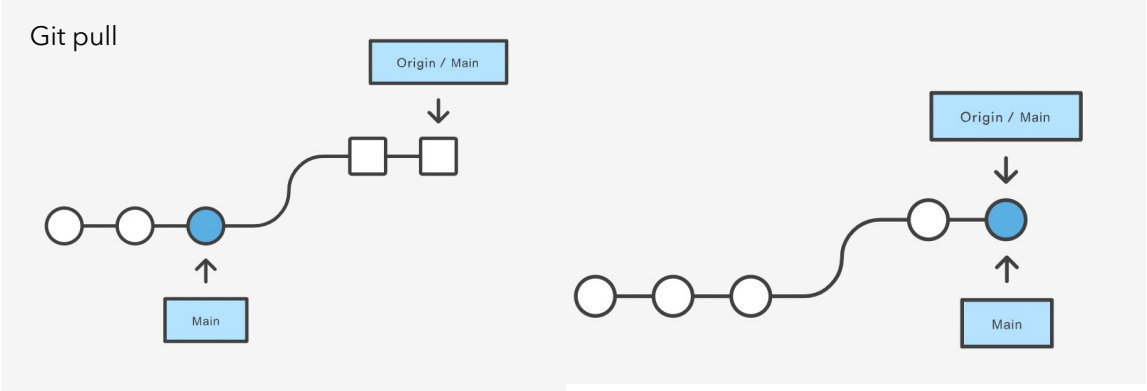
Collaborating using git

- First you may need to view all your connections to remote repositories – git remote
- To make a working copy of the remote repo to your local machine – git clone <URL-to-repo>
- To download contents from a remote repository – git fetch <remote>
- (More aggressive because automatically performs git merge after git fetch) – git pull
- To contribute (make changes) to a repository - git push <remote-name> <branch-name>





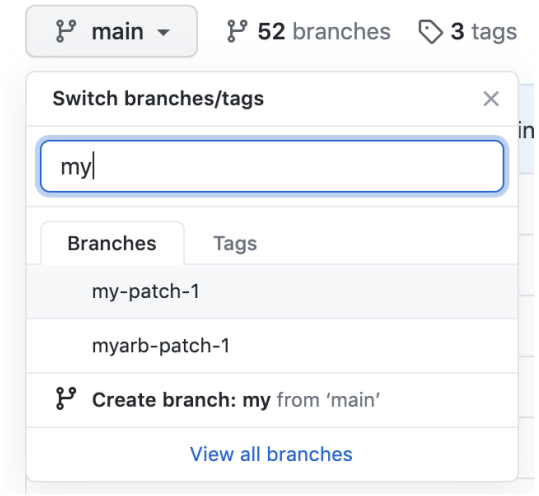




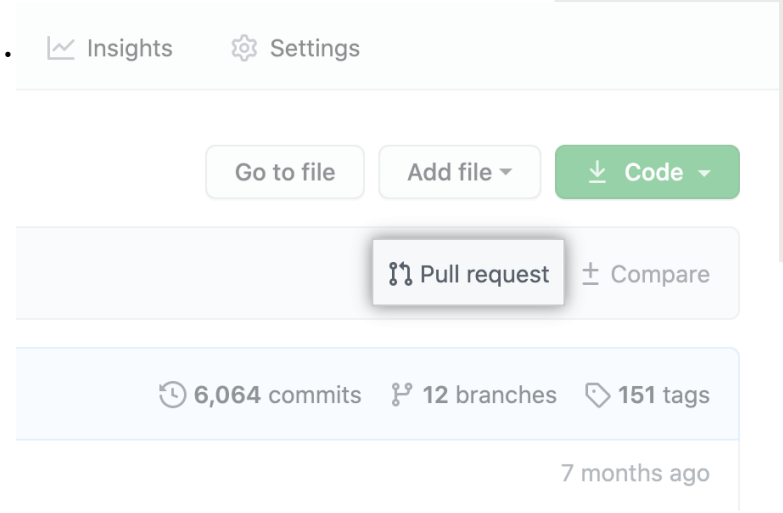
Making Pull Requests (on Github)

- Pull requests are a mechanism for a developer to notify team members that they have completed a feature.

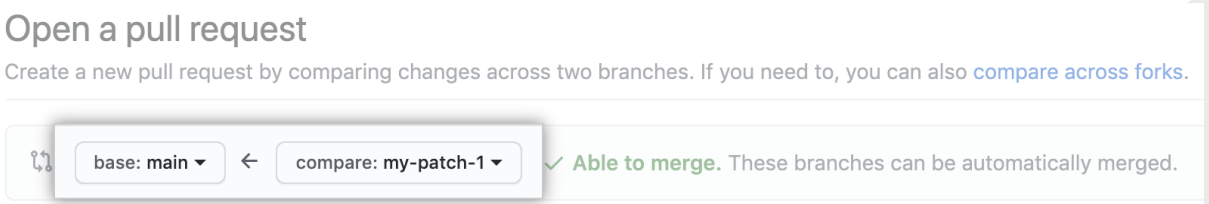
1.



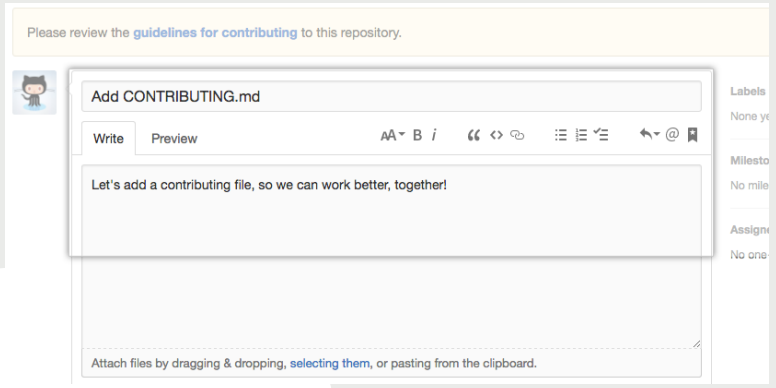
2.



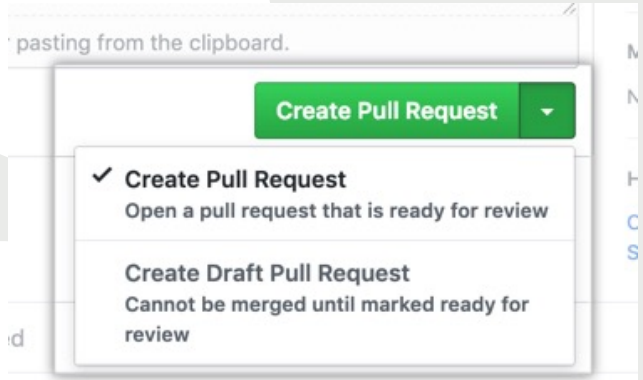
3.



4.

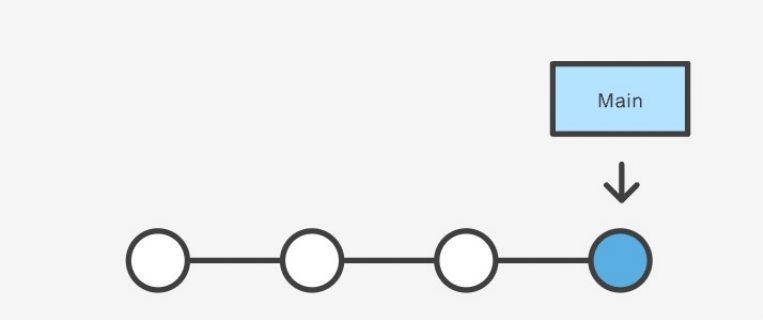


5.



Using branches

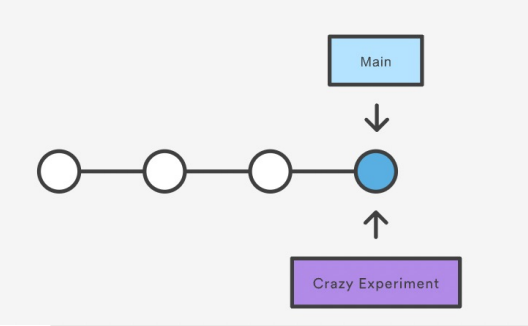
- A branch represents an independent line of development. But really, they are just pointers to commits!



Original repository

```
git branch crazy-experiment
```

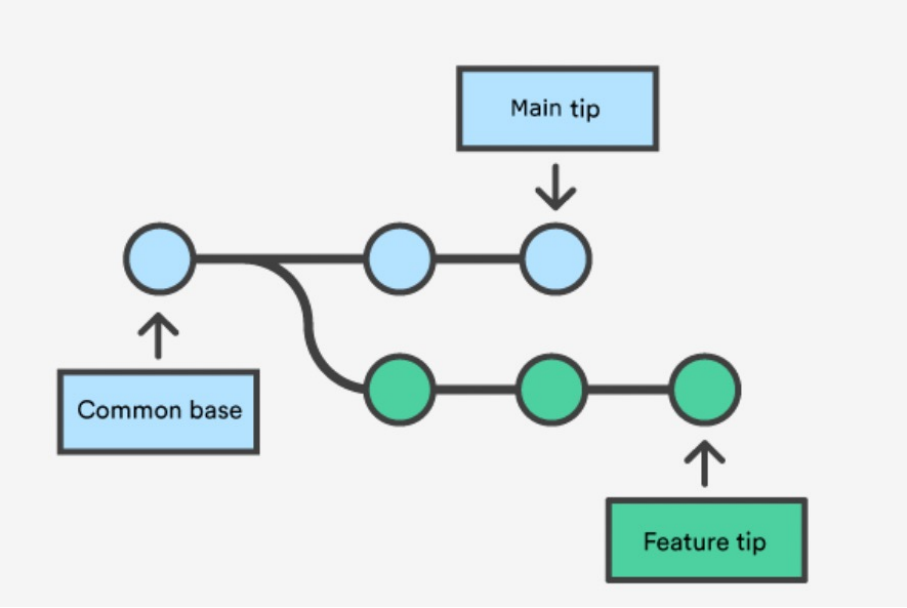
Command to create a new branch



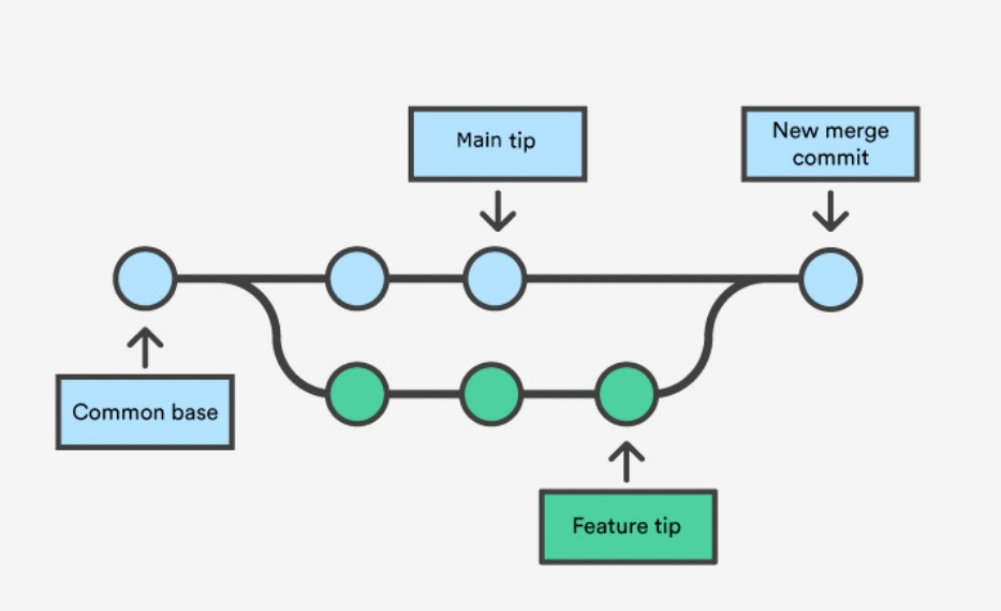
Modified repository with new branch

Using branches – Merging

The git merge command lets you take the independent lines of development created by git branch and integrate them into a single branch.



Before



After

Merge Conflicts!

2 reasons a merge conflict can occur:

- Changes in your working directory/staging area may be overwritten
- Changes by another developer may be overwritten

How to handle?

```
$ git status
On branch main
You have unmerged paths.
(fix conflicts and run "git commit")
(use "git merge --abort" to abort the merge)

Unmerged paths:
(use "git add <file>..." to mark resolution)

both modified:   merge.txt
```

```
here is some content not affected by the conflict
<<<<<<< main
this is conflicted text from main
=======
this is conflicted text from feature branch
>>>>>>> feature branch;
```

Git Workflow

A Git workflow is a recipe or recommendation for how to use Git to accomplish work in a consistent and productive manner.

Important considerations when deciding on a git workflow:

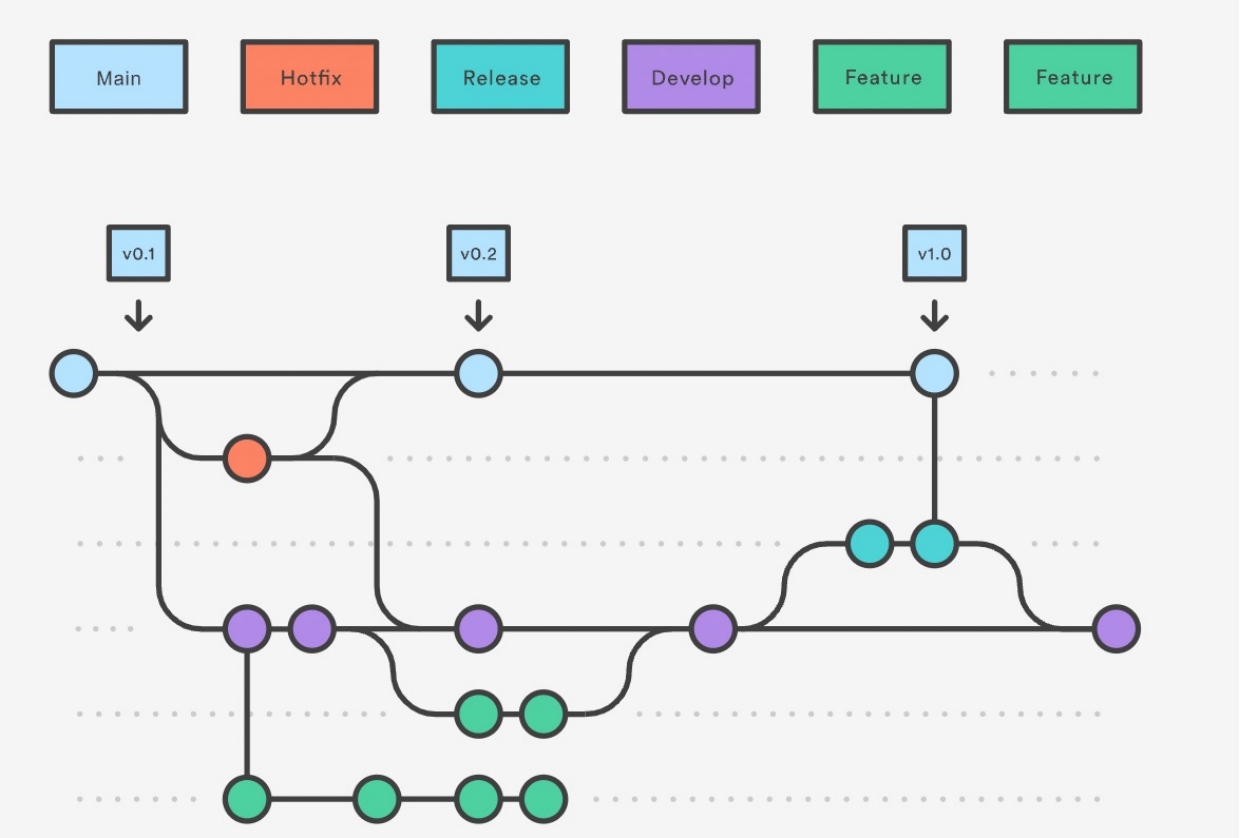
1. Should scale with team size.
2. Minimize unnecessary cognitive overhead.
3. Should be easy to undo error within this workflow.

Many examples: Centralized workflow, Feature Branch workflow, Forking workflow, Gitflow workflow.

Gitflow Workflow

The overall flow of Gitflow is:

- A develop branch is created from main
- A release branch is created from develop
- Feature branches are created from develop
- When a feature is complete it is merged into the develop branch
- When the release branch is done it is merged into develop and main
- If an issue in main is detected a hotfix branch is created from main
- Once the hotfix is complete it is merged to both develop and main



Source Code Management best practices

1. Commit often
2. Ensure you're working from the latest version
3. Make detailed notes
4. Review changes before committing
5. Use branches
6. Agree on a workflow

References

Atlassian Bitbucket's git tutorial (series of webpages):

https://www.atlassian.com/git/tutorials/what-is-version-control

Game to learn how git branching works:

https://learngitbranching.js.org/

Cheatsheet:

https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet