WHY SHOULD YOU DO OPEN SOURCE?

PRESENTED BY ANIRUDH PANDA

OUR MAIN TOPICS TODAY

What is Open Source?

Criteria for Open Source

How does Open Source work?

Pros and Cons of Open Source

Benefits of Doing Open Source

Examples

To Be Continued...

VERSION CONTROL

What is Git?

History of Git

How to install and configure Git?

Most used Git Commands

What are Merge Conflicts?

How useful git stash is!

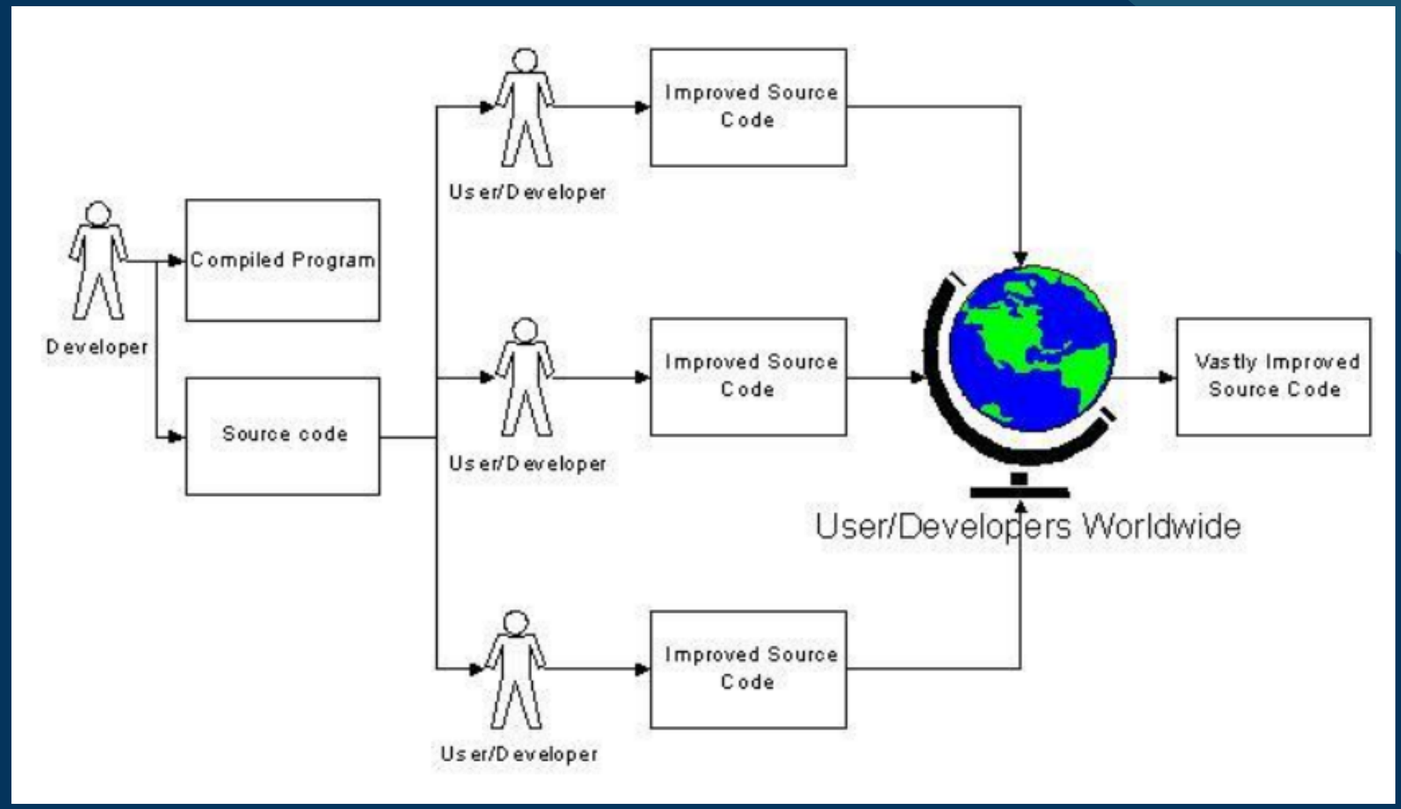
WHAT IS OPEN SOURCE?

Open Source refers to the source code that is available for modification, enhancement and anyone can use freely. Open-source software is usually developed as a public collaboration and made freely available. Anyone can also contribute in the source code to make it better.

CRITERIA FOR OPEN SOURCE

- Source Code
- Free Redistribution
- Distribution of License
- License Must Not Restrict Other Software
- No Discrimination Against Fields of Endeavor
- No Discrimination Against Persons or Groups
- Derived Integrity of The Author's Source Code Works

HOW DOES OPEN SOURCE WORK?



Pros and Cons of Open Source software

PROS:

- Highly Reliable, if maintained!
- Availability of Source Code
- Build your Networking with like-minded people
- Free and/or Cheaper than Commercial Products.

CONS:

- No Guaranteed Support
- Security becomes a major issue
- Not generally straightforward to use and requires a certain learning curve to get accustomed.

BENEFITS OF DOING OPEN SOURCE CONTRIBUTIONS

1. HELPS IN WRITING CLEANER CODE
2. YOUR OWN CODE CAN BE USED GLOBALLY
3. YOU WILL LEARN A LOT BOTH SOFT & HARD SKILLS
4. YOU WILL WORK WITH THE SMARTEST PEOPLE
5. A BETTER UNDERSTANDING OF TECHNOLOGY
6. HELPS IN BEING PREPARED FOR PROJECTS
7. GAIN RECOGNITION

TOP OPEN SOURCE SOFTWARE EXAMPLES

- MOZILLA FIREFOX
- VS CODE
- LIBREOFFICE
- VLC MEDIA PLAYER
- LINUX
- BLENDER
- PYTHON
- DJANGO
- REACT/ANGULAR/VUE
- WORDPRESS
- KUBERNETES

EXAMPLES OF SOME OSS

GIT

VERSION CONTROL SYSTEM

Git is an **Open Source Distributed Version Control System**

- GIT WAS DEVELOPED BY LINUS TORVALDS IN 2005
- THE SAME PERSON WHO HAS DEVELOPED THE LINUX KERNEL

HISTORY OF GIT
INNOVATION THROUGH CONTROVERSY

As a result of the **Linux Kernel**'s unusual development situation, the developers struggled to find VCSs(version control systems) that fit their needs. They settled for a mix of **BitKeeper and Concurrent Revisions System** (CVS), with a group of core developers working on each system to manage the development of the kernel.

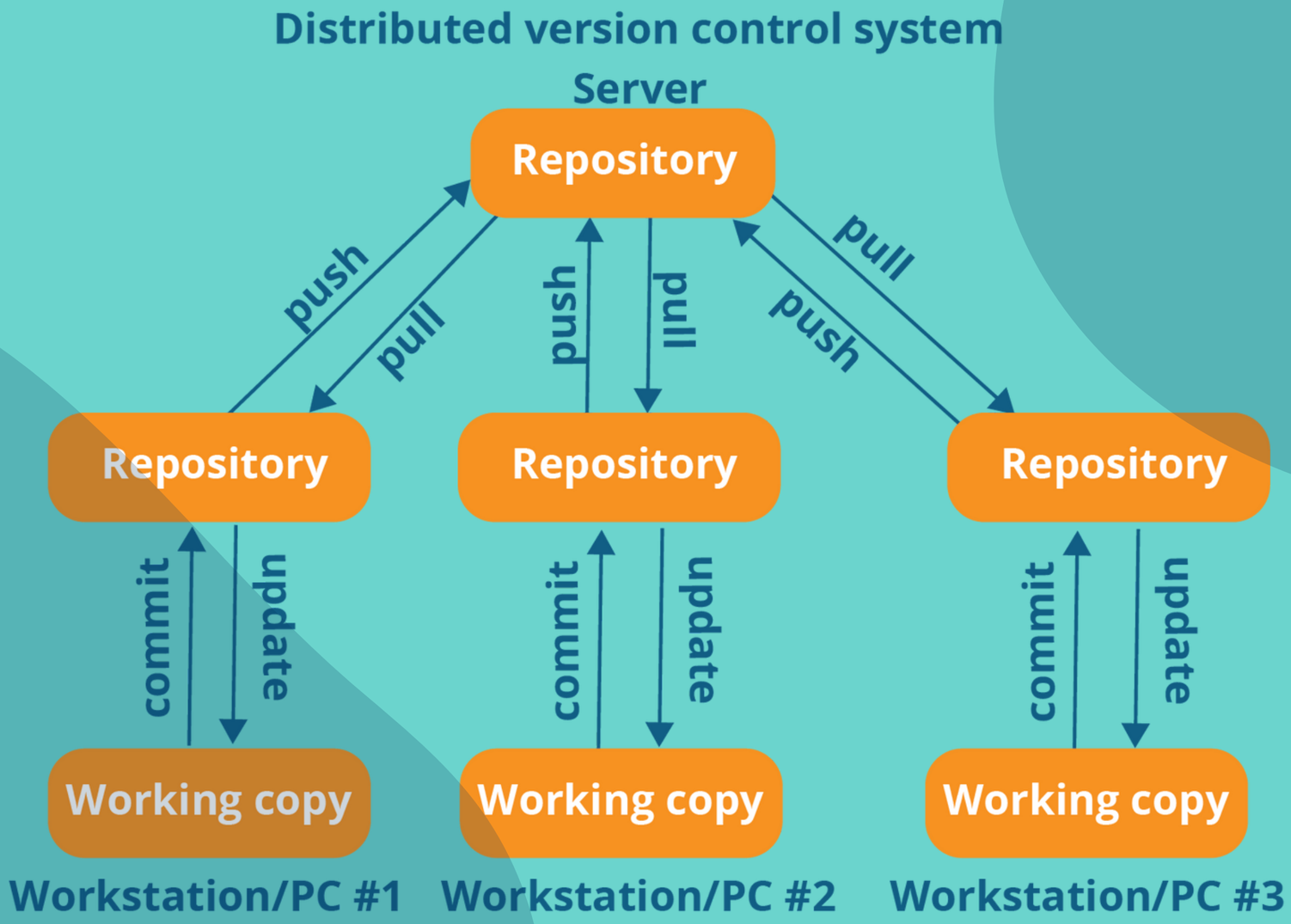
In early 2005, the relationship between the community that developed the Linux kernel and the commercial company that developed BitKeeper broke down, and the tool's free-of-charge status was revoked. Many Linux core developers that relied on BitKeeper's free software to develop the Linux kernel were now locked out from using it.

This prompted the Linux development community to develop their own tool based on some of the lessons they learned while using BitKeeper. Linus Torvalds, the principal developer of the Linux kernel, began work on a new VCS after seeing no other free options that met their needs.

WHAT IS A VERSION CONTROL SYSTEM?

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

- For example, You have created an **App** that is on the 1.0 version. Suppose you want to add a feature and so the **App** is upgraded to the 1.1 version.
- But after a few days, the **App** is crashing because of the feature. So now you want to roll back to the 1.0 version.
- So here **VCS** comes into the picture. You can track all the changes to file made from a particular checkpoint (**commit**)


Distributed version control system
Server
Repository
push
pull
push
pull
pull
push
Repository
Repository
Repository
commit
update
commit
update
commit
update
Working copy
Working copy
Working copy
Workstation/PC #1
Workstation/PC #2
Workstation/PC #3

WHAT IS GITHUB?

GitHub is a code hosting platform for Version Control and Collaboration. It lets you and others work together on projects from anywhere.

OTHER ALTERNATIVES

GitLab and BitBucket

COMMON GIT COMMANDS

- git --version
- git init
- git status
- git add <filename>
- git commit -m "Your message"
- git diff
- git log
- git clone
- git remote
- git push
- git pull
- git branch
- git merge
- git stash

git

https://git-scm.com/

CONFIGURATION OF GLOBAL VARIABLES

git config --global user.name “[firstname lastname]”

git config --global user.email “[valid-email]”

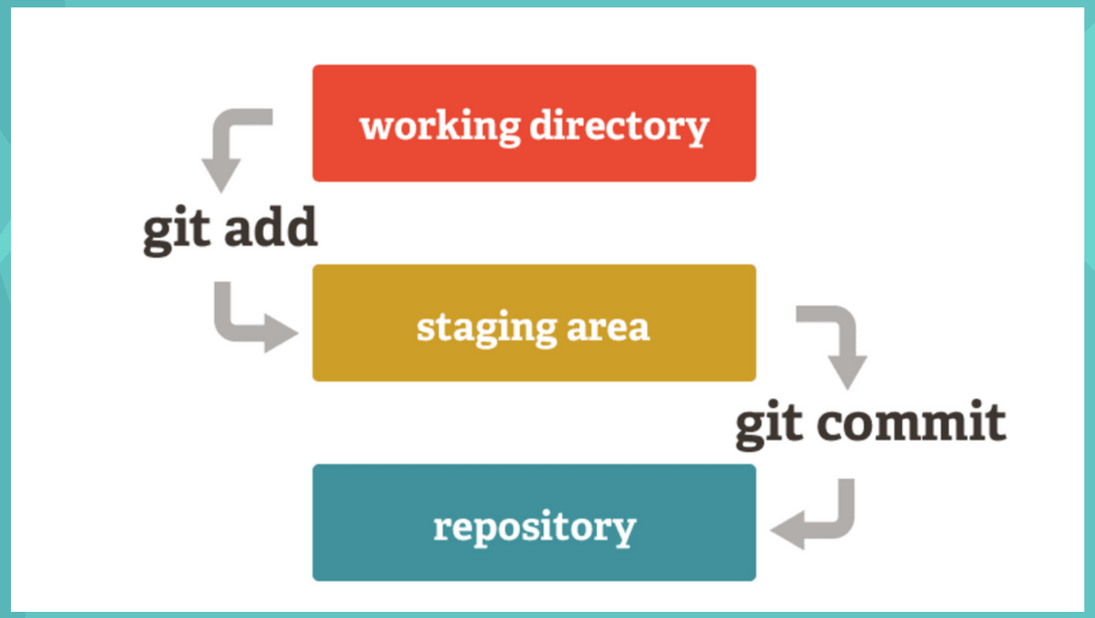
- **git --version:** Returns the version of the git installed in the machine.
- **git init:** Creates an empty git repository or reinitializes the existing one.
- **git status:** Showing the working tree status and gives the list of files changed.

git add

git add <filename>: Add file contents to the staging area.

git add . :- Add all the changed files to the staging area.

- **Staging Area:** Git knows what files are changed, but doesn't know why changed.



git commit

git commit -m "Your Message"

Creates a new commit containing the current contents of the index and the given log message describing the changes.

MOVING FILES FROM STAGED AREA TO THE GIT FOLDER IS CALLED A COMMIT.

git diff

git diff: If you want to know exactly what you changed, not just which files were changed – you can use the command

git diff filename:- Shows the content changed in a file after the last commit.

git log.

git log

You can see all the commits/versions of your git project.

git clone

git clone

Clone a repository into a new directory.

git remote

used to add a new remote

git remote add origin <url>: The git remote add command takes two arguments:

- A remote name, for example, **origin**
- A remote URL, for example, **https://github.com/user/repo.git**

git push

git push

- Update remote refs along with associated objects
- **git push origin main:-** update local branch with remote branch

git pull

git pull: The git pull command is used to fetch and download content from a remote repository and immediately update the local repository to match that content.

git pull origin main:- Fetch from the main branch of the remote repository and integrate with local branch

git branch

list all the branches

git branch <branch name>

Creates a branch with name as <branch name>

git checkout

switch branches or restore working tree files

git checkout <branch name>

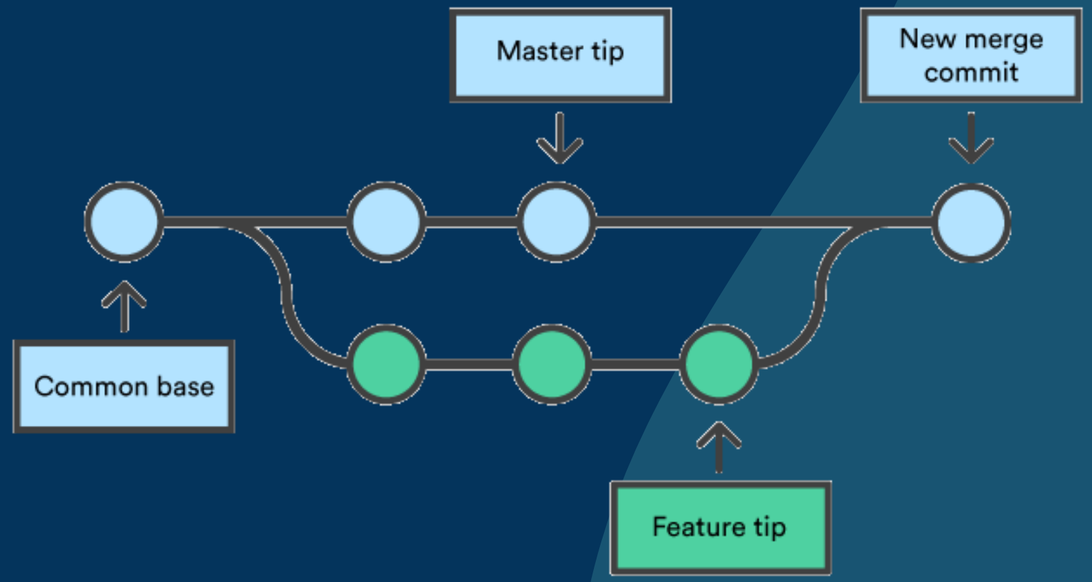
Checkout to <branch name>

git checkout -b <branch name>

Creates and Checkout to <branch name>

git merge

Join two or more development histories together.

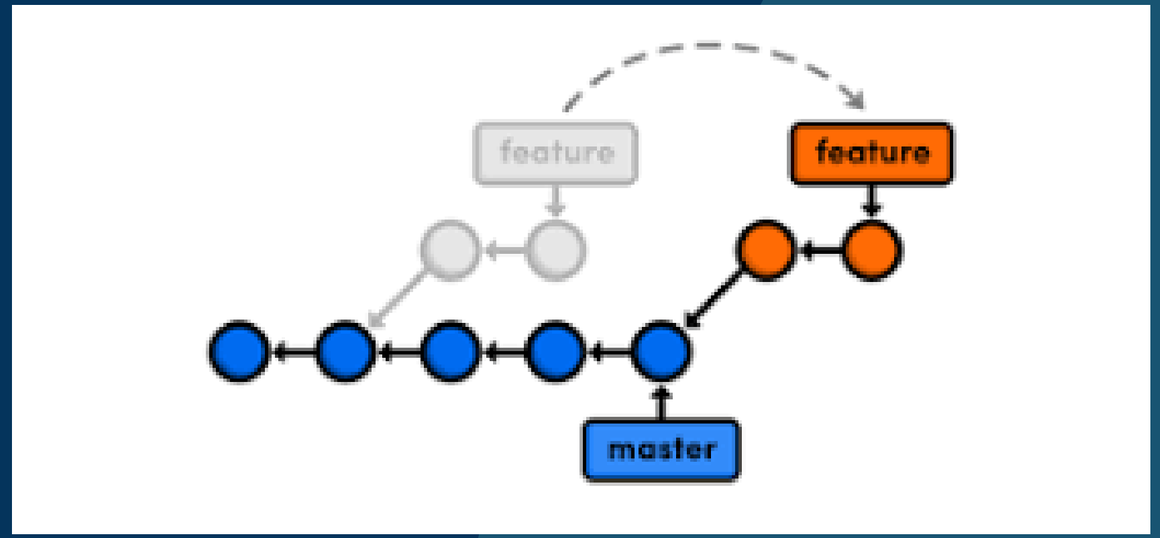


git merge <branch name>

Merge a <branch_name> with the current branch

git rebase

Rebasing is the process of moving or combining a sequence of commits to a new base commit.



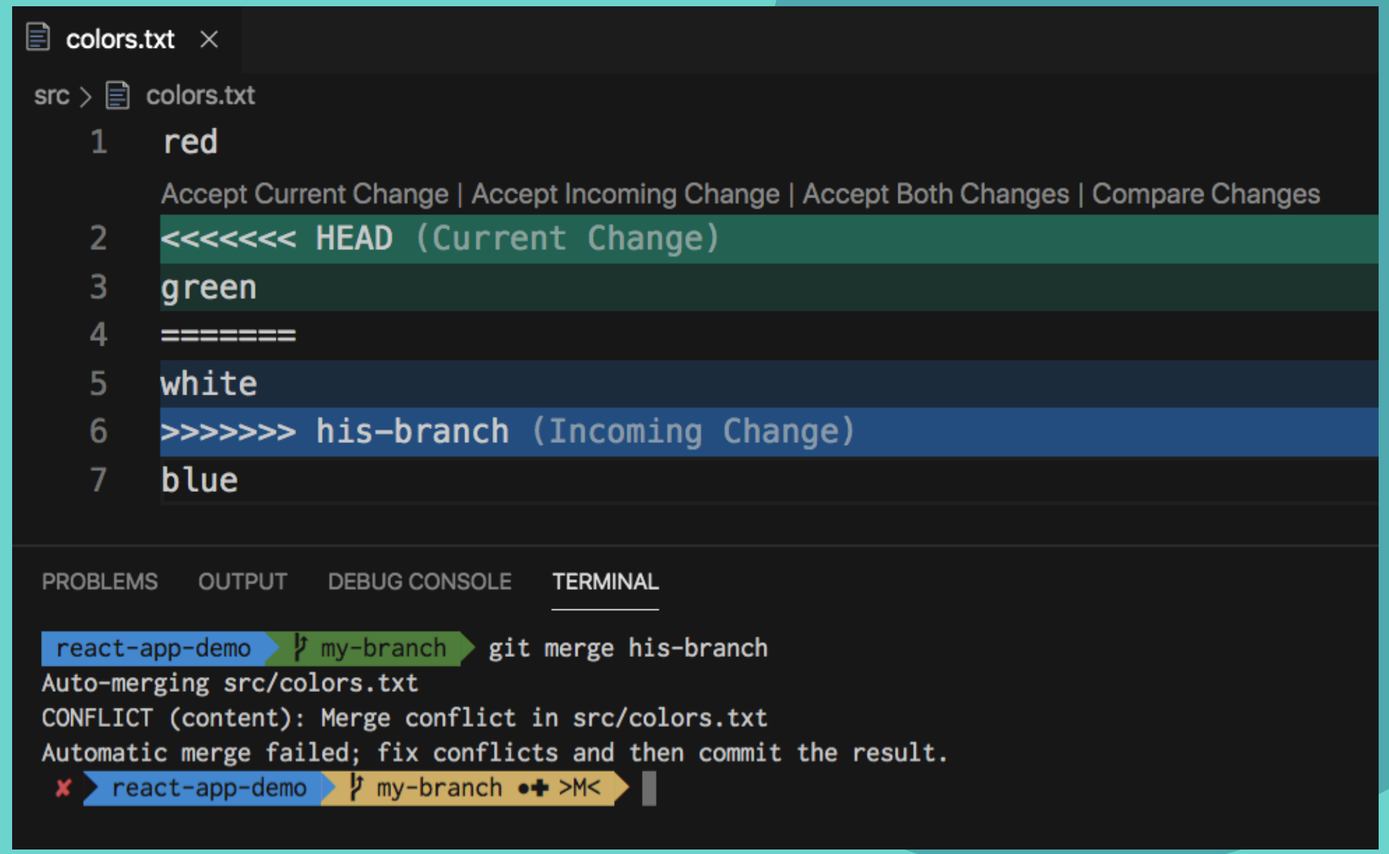
git rebase <branch name>

Merge a <branch_name> with the current branch

MERGE CONFLICTS

When two or more people are working on the same file in a repo and are making changes to it, there are chances that a merge conflict will occur

These conflicts need to be solved manually by deciding which code to keep and which code to discard.



git stash

Why is git stash important?

By default, **git stash** stores (or "stashes") the uncommitted changes (staged and unstaged files) and overlooks untracked and ignored files.

You can view your stashes with the command git stash list. Stashes are saved in a last-in-first-out (LIFO) approach: $ **git stash list**

git stash pop stash@{1}

A stash reapplies the changes while pop removes the changes from the stash and reapplies them to the working copy.

It is good practice to remove stashes that are no longer needed. You must do this manually with the following commands:

- **git stash clear** empties the stash list by removing all the stashes.
- **git stash drop <stash_id>** deletes a particular stash from the stash list.

HOW COLLABORATION WORKS AT SOFTWARE COMPANIES

- Suppose the code for an application is hosted on a platform like **GitHub**.
- The repository will have 2 branches that is **main** and **develop**.

- **Main:** This contains the code of the application that is currently on Production.
- **Develop:** This branch has the latest stable code for the application.

ROUGH AGENDA FOR TOMORROW'S SESSION:

- HACKTOBERFEST in DETAIL
- How to use Github and Github Student Developer pack
- How to participate in various Open Source programs?
- How to find organizations to contribute?
- Hands-on session on successfully creating a PR

And a LOT MOREEEEE!!!

Thankyou!! Questions??