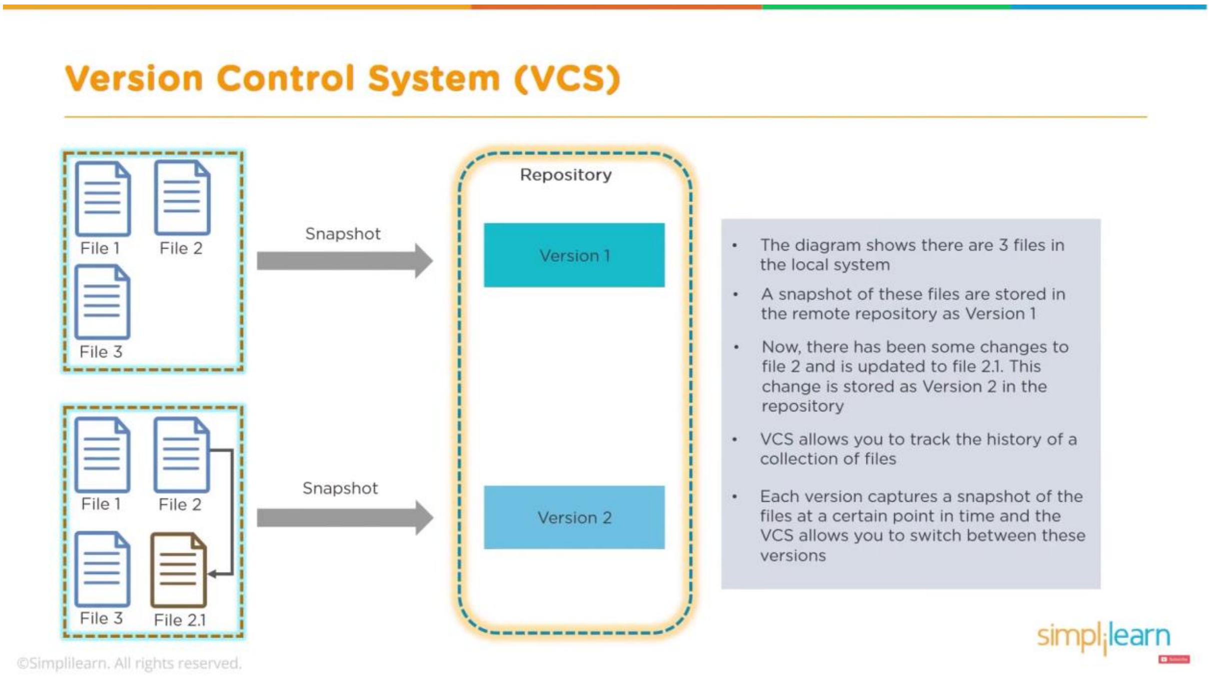
GIT – version control system

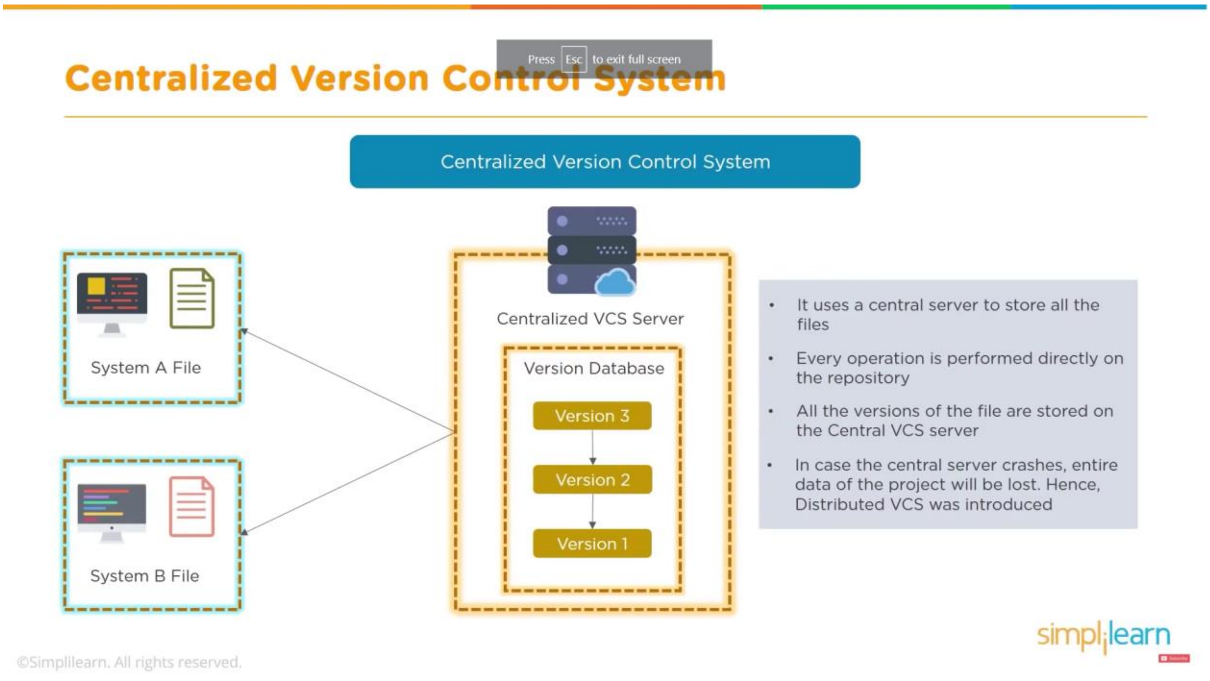
GIT and GITHUB allows us to maintain the history of the project

GITHUB is an online platform that hosts the GIT repository



Version Control System (VCS)

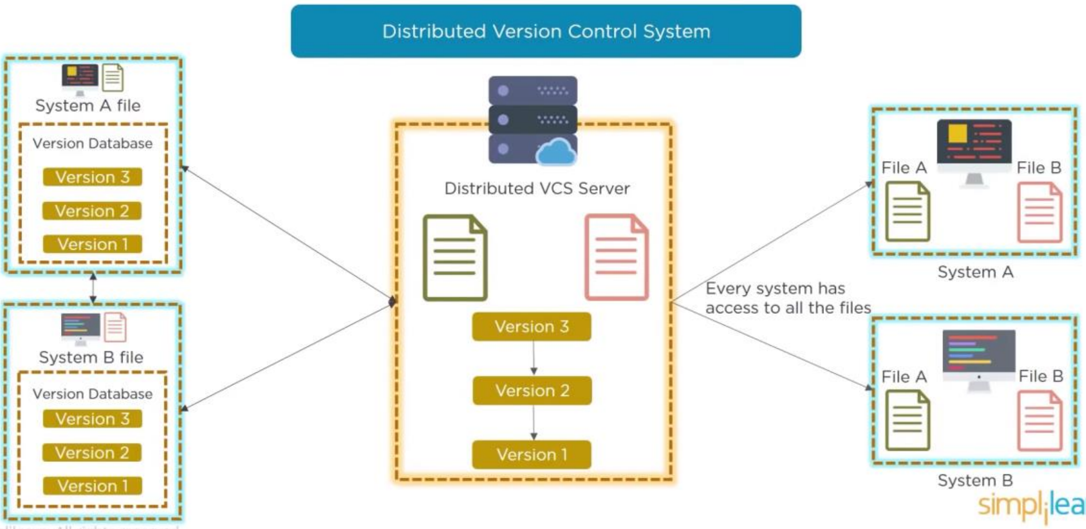
- The diagram shows there are 3 files in the local system
- A snapshot of these files are stored in the remote repository as Version 1
- Now, there has been some changes to file 2 and is updated to file 2.1. This change is stored as Version 2 in the repository
- VCS allows you to track the history of a collection of files
- Each version captures a snapshot of the files at a certain point in time and the VCS allows you to switch between these versions



Centralized Version Control System

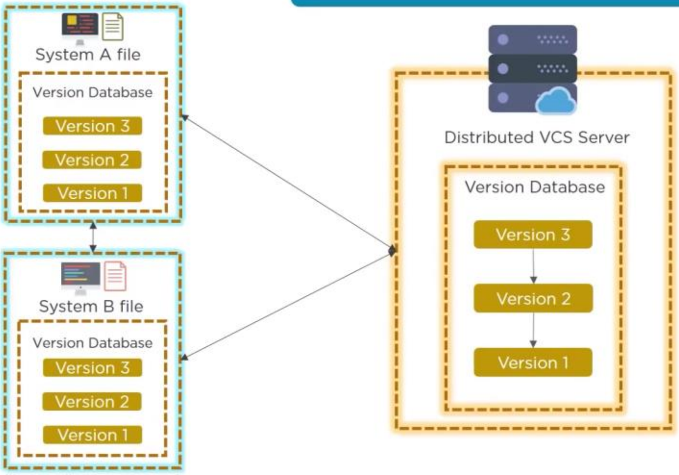
- It uses a central server to store all the files
- Every operation is performed directly on the repository
- All the versions of the file are stored on the Central VCS server
- In case the central server crashes, entire data of the project will be lost. Hence, Distributed VCS was introduced

Distributed Version Control System

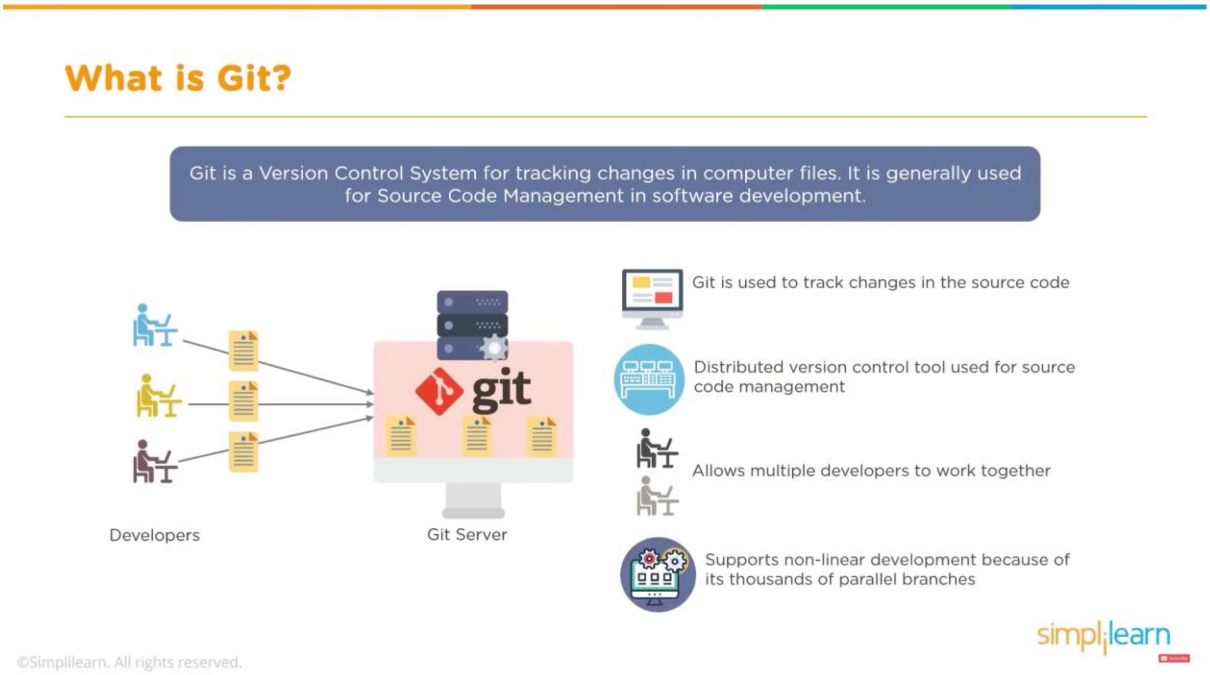


Distributed Version Control System

Distributed Version Control System



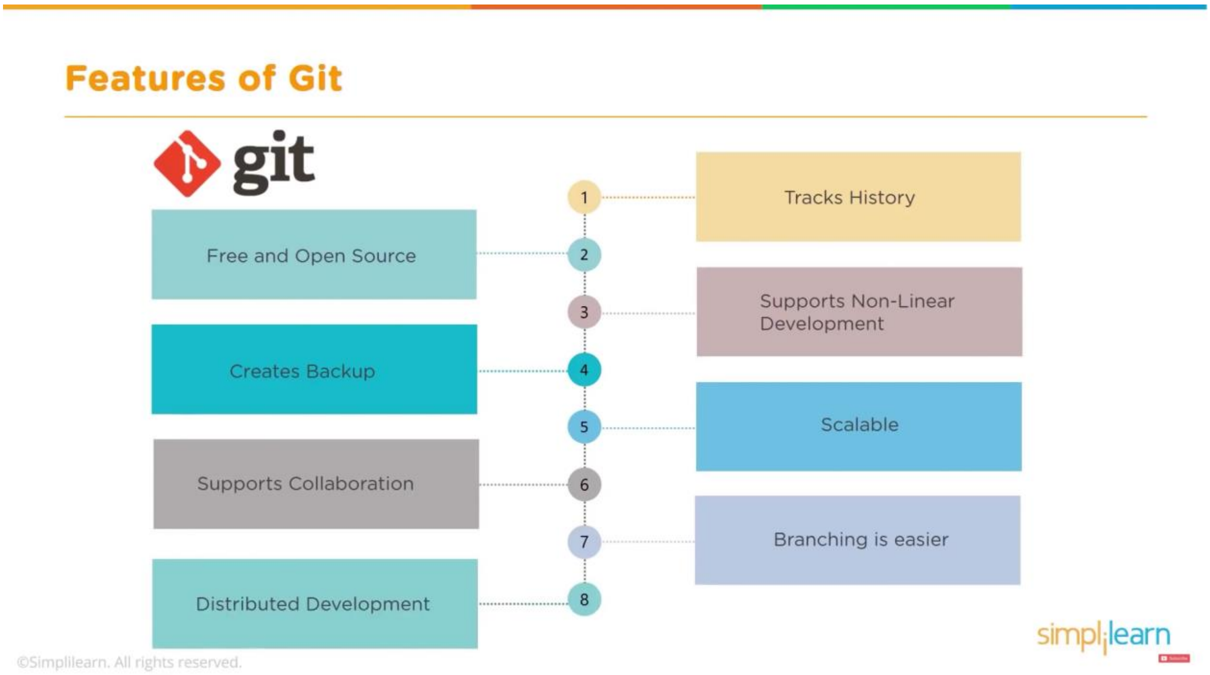
- Every programmer has a copy of all the versions of the code on their local system
- Distributed VCS moves from the client-server approach of Central VCS to peer-to-peer approach
- They can update their local repositories with new data from the central server and changes gets reflected to the main repository
- Git is one such Distributed VCS tool



What is Git?

Git is a Version Control System for tracking changes in computer files. It is generally used for Source Code Management in software development.

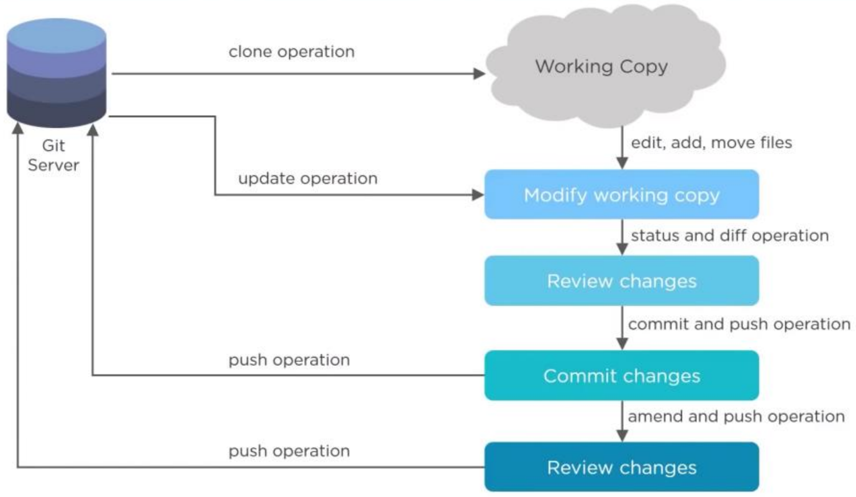
- Git is used to track changes in the source code
- Distributed version control tool used for source code management
- Allows multiple developers to work together
- Supports non-linear development because of its thousands of parallel branches



Features of Git

1. Tracks History
2. Free and Open Source
3. Supports Non-Linear Development
4. Creates Backup
5. Scalable
6. Supports Collaboration
7. Branching is easier
8. Distributed Development

Git Workflow



Git Server

clone operation

Working Copy

edit, add, move files

update operation

Modify working copy

status and diff operation

Review changes

commit and push operation

push operation

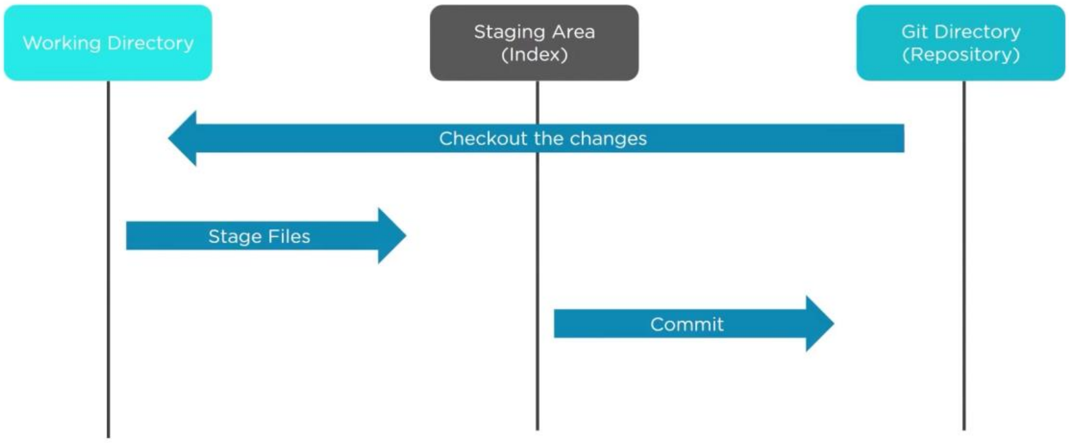
Commit changes

amend and push operation

push operation

Review changes

Git Workflow – 3 States



Working Directory

Staging Area (Index)

Git Directory (Repository)

Checkout the changes

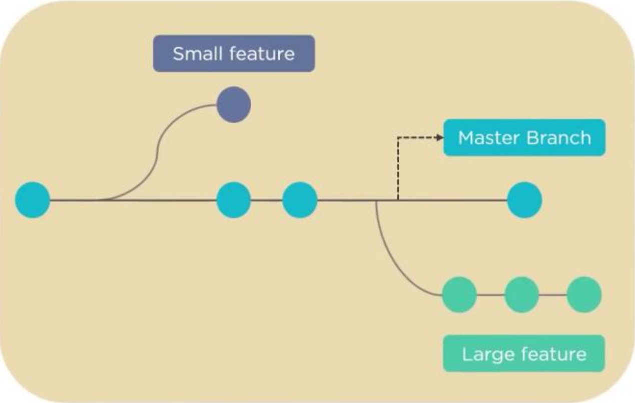
Stage Files

Commit

Checkout any existing version, make changes, stage them and commit

Branch in Git

- ❑ Branch in Git is used to keep your changes until they are ready
- ❑ You can do your work on a branch while the main branch (master) remains stable. After you are done with your work, you can merge it to the main branch



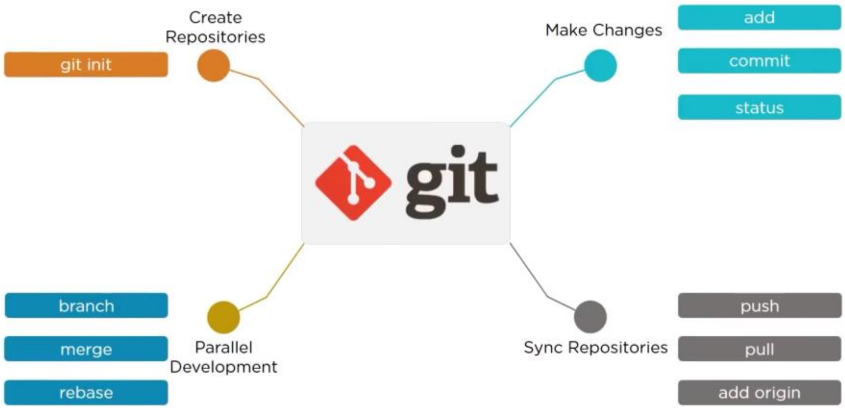
Small feature

Master Branch

Large feature

- ❑ The diagram shows there is a Master branch
- ❑ There are 2 more branches Small feature and Large feature working separately
- ❑ Once the work is complete for the two separate branches, you can merge it to the master branch

Commands in Git



Create Repositories

git init

Make Changes

add

commit

status

Parallel Development

branch

merge

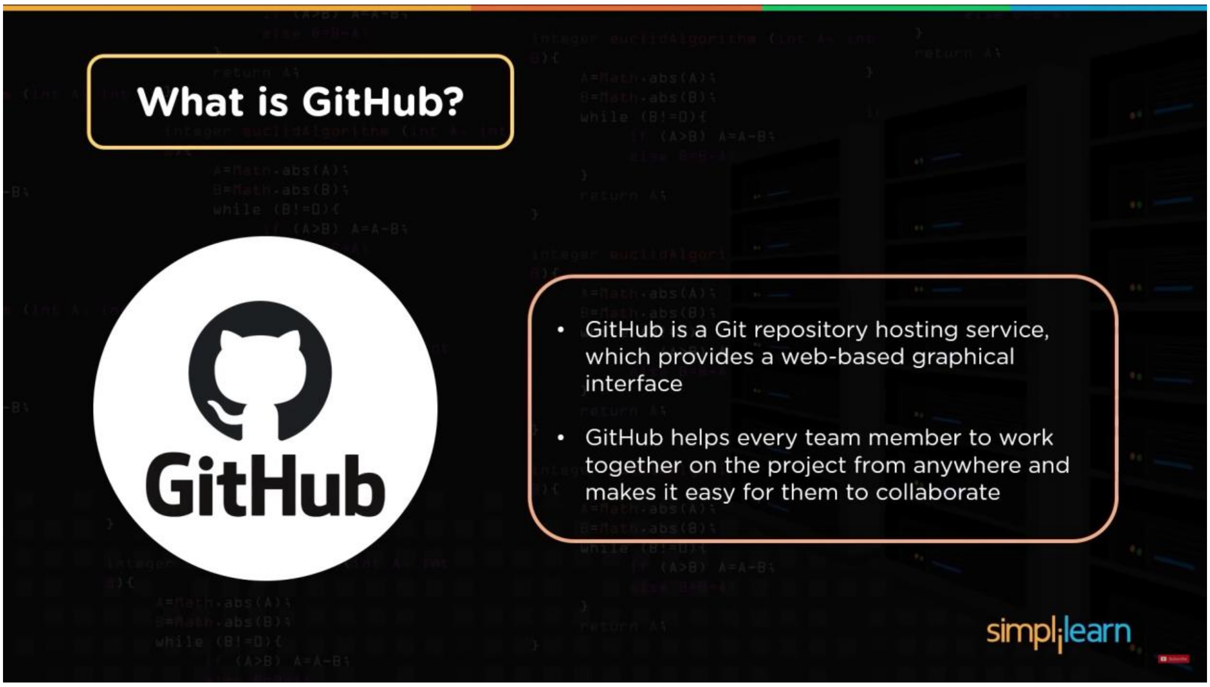
rebase

Sync Repositories

push

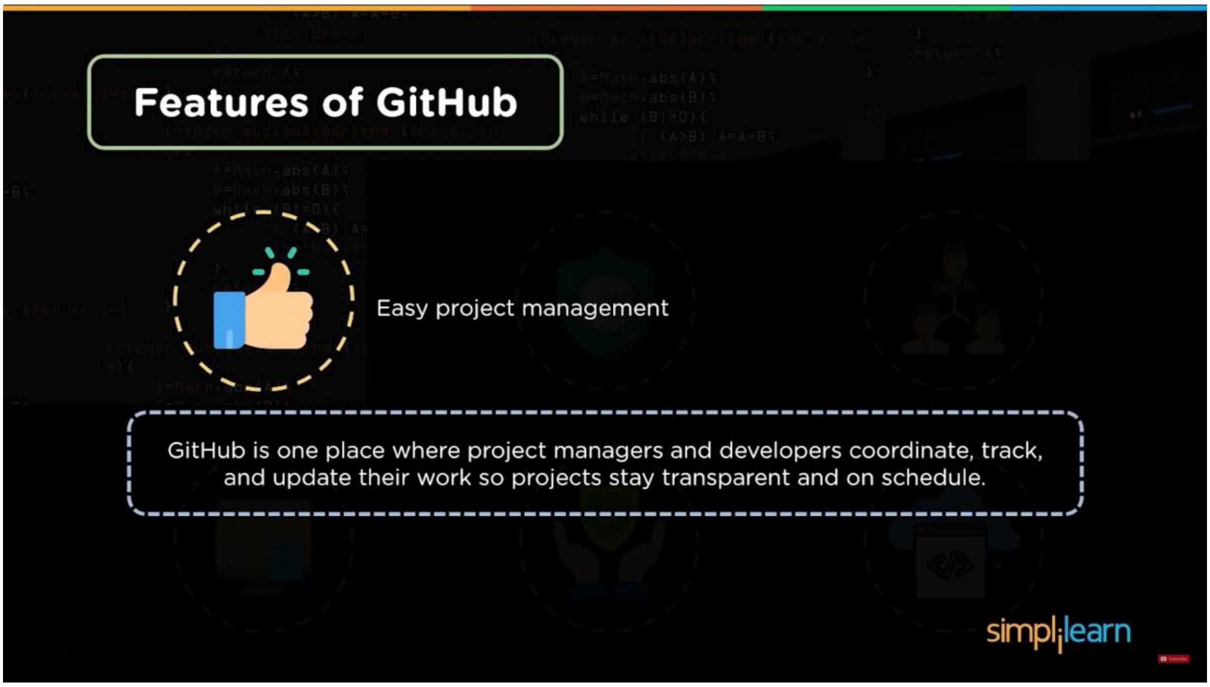
pull

add origin



What is GitHub?

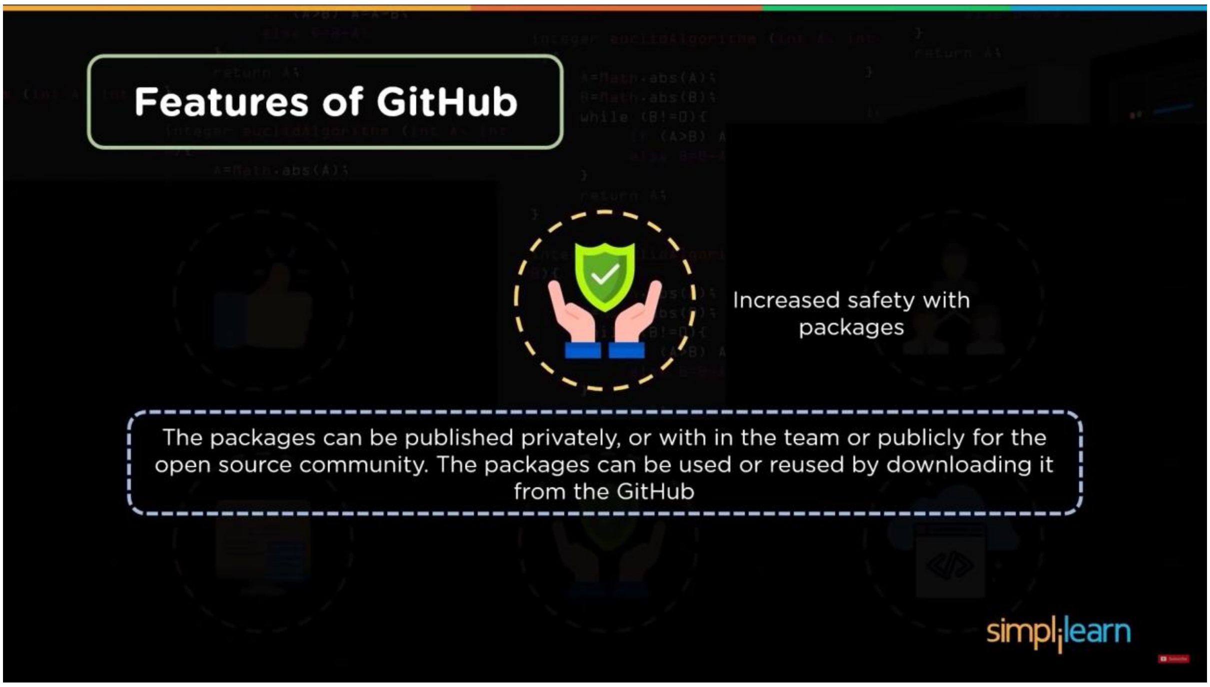
- GitHub is a Git repository hosting service, which provides a web-based graphical interface
- GitHub helps every team member to work together on the project from anywhere and makes it easy for them to collaborate



Features of GitHub

Easy project management

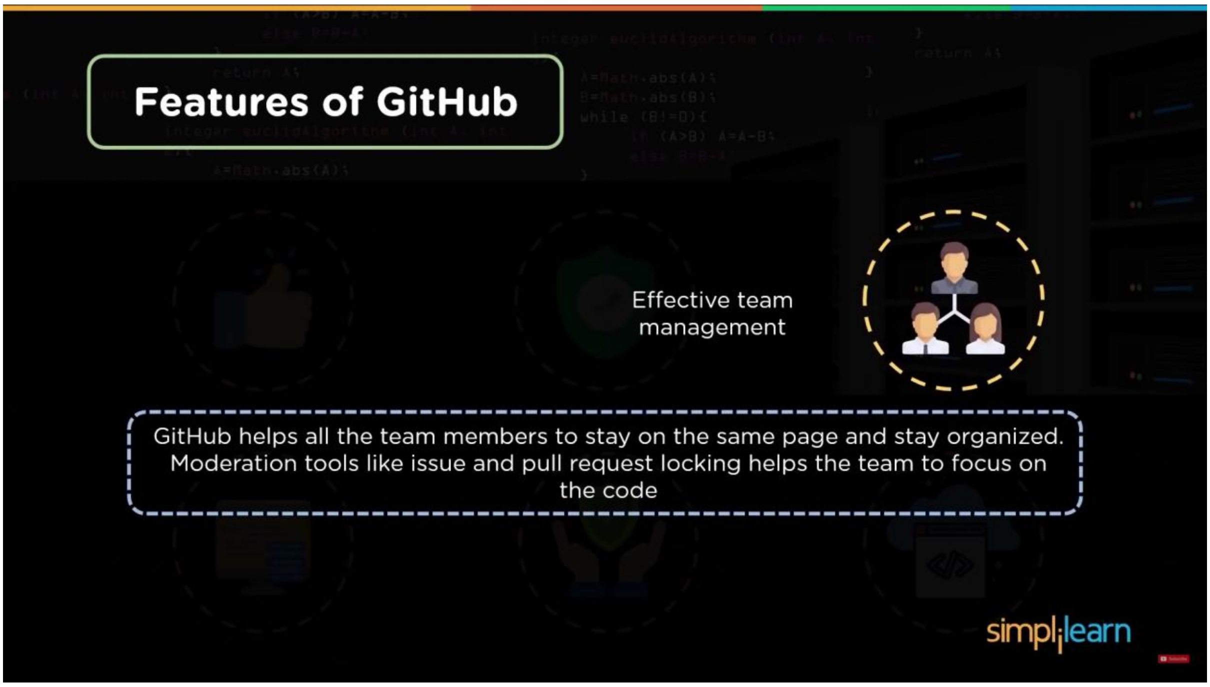
GitHub is one place where project managers and developers coordinate, track, and update their work so projects stay transparent and on schedule.



Features of GitHub

Increased safety with packages

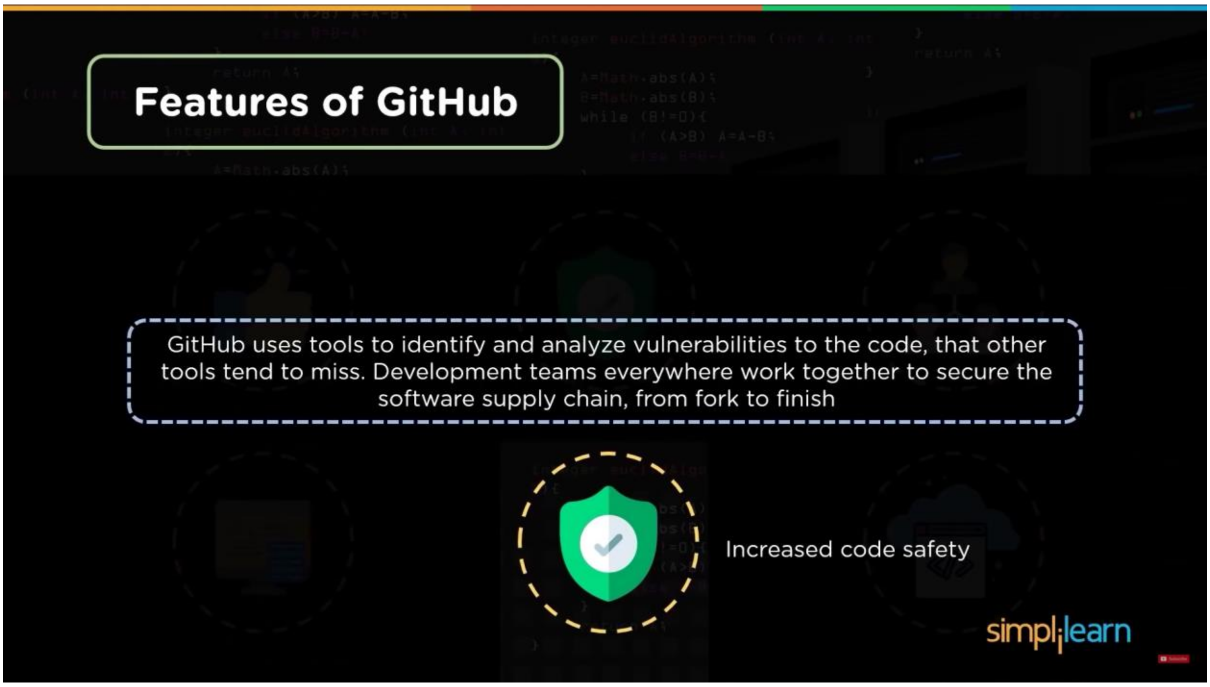
The packages can be published privately, or with in the team or publicly for the open source community. The packages can be used or reused by downloading it from the GitHub



Features of GitHub

Effective team management

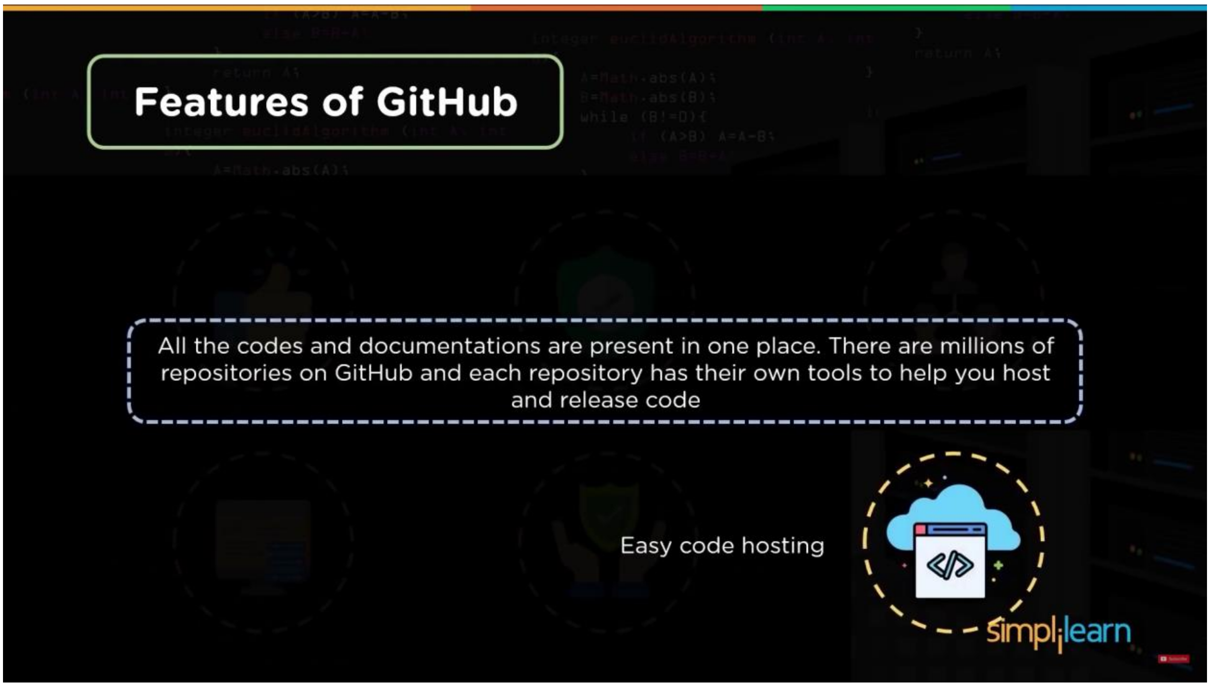
GitHub helps all the team members to stay on the same page and stay organized. Moderation tools like issue and pull request locking helps the team to focus on the code



Features of GitHub

GitHub uses tools to identify and analyze vulnerabilities to the code, that other tools tend to miss. Development teams everywhere work together to secure the software supply chain, from fork to finish

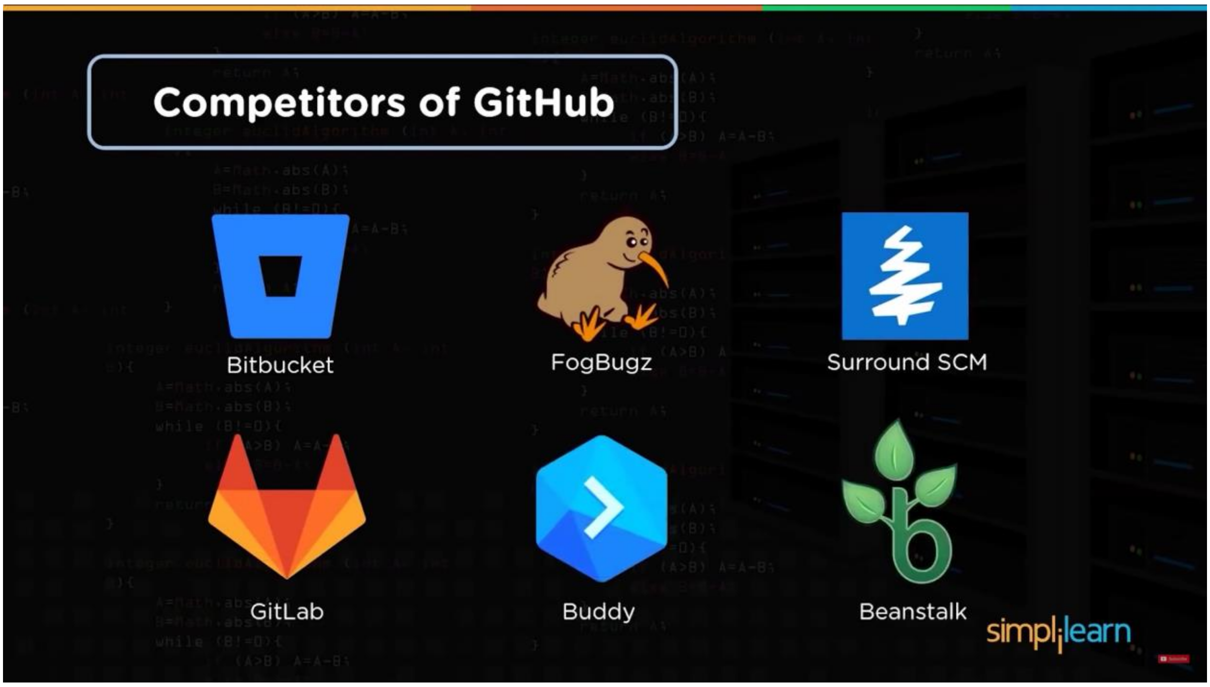
Increased code safety



Features of GitHub

All the codes and documentations are present in one place. There are millions of repositories on GitHub and each repository has their own tools to help you host and release code

Easy code hosting



Competitors of GitHub

Bitbucket

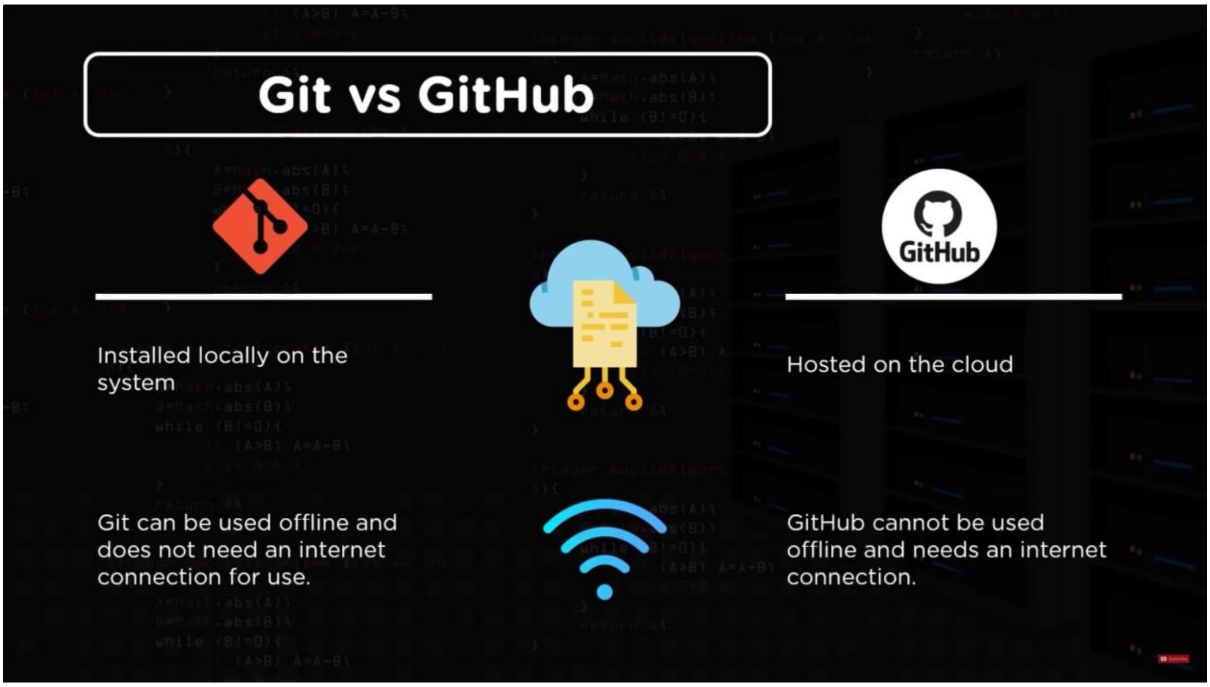
FogBugz

Surround SCM

GitLab

Buddy

Beanstalk



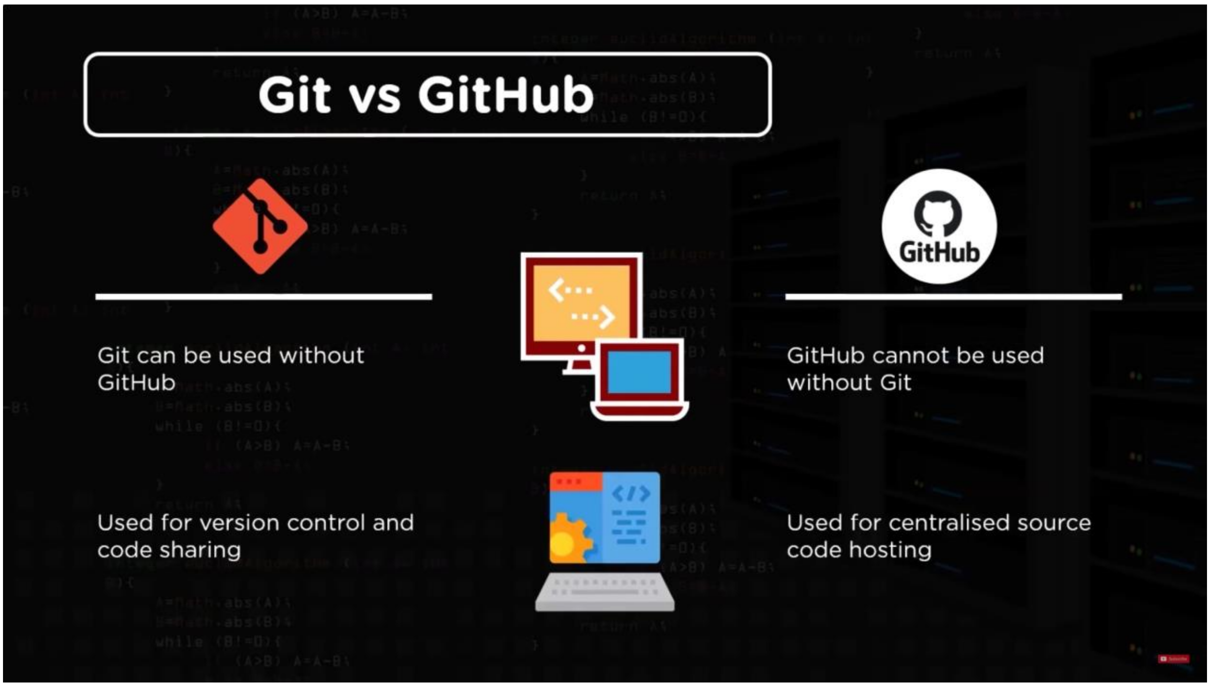
Git vs GitHub

Installed locally on the system

Hosted on the cloud

Git can be used offline and does not need an internet connection for use.

GitHub cannot be used offline and needs an internet connection.



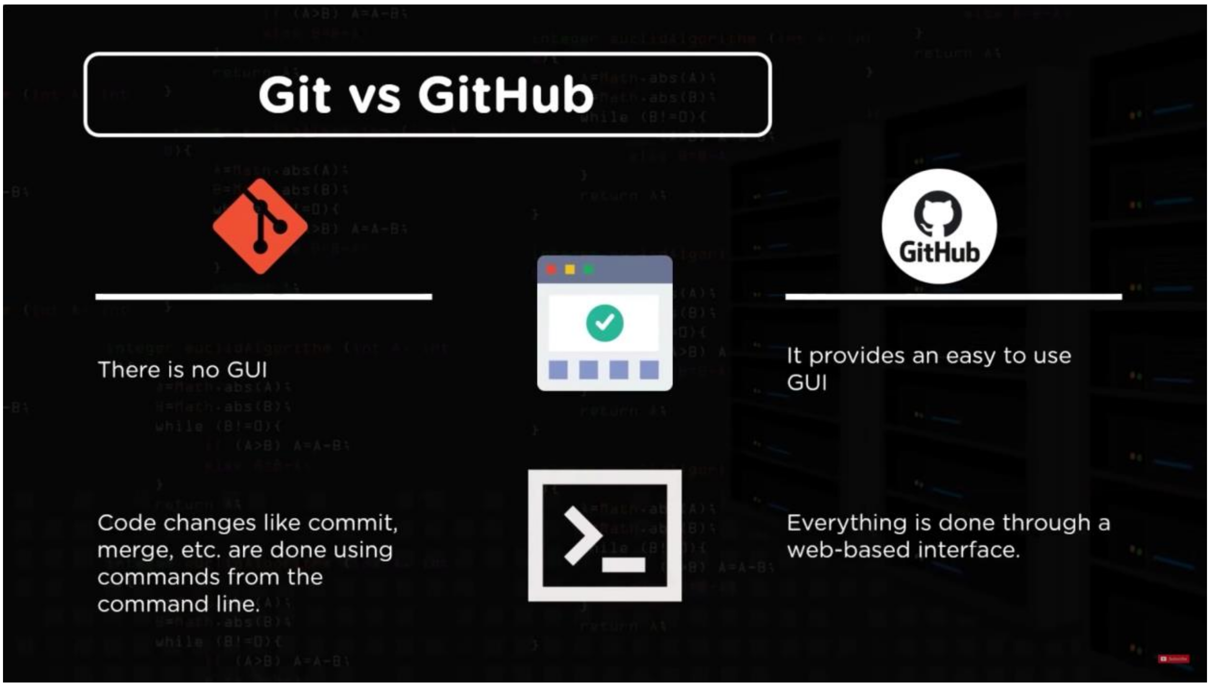
Git vs GitHub

Git can be used without GitHub

Used for version control and code sharing

GitHub cannot be used without Git

Used for centralised source code hosting



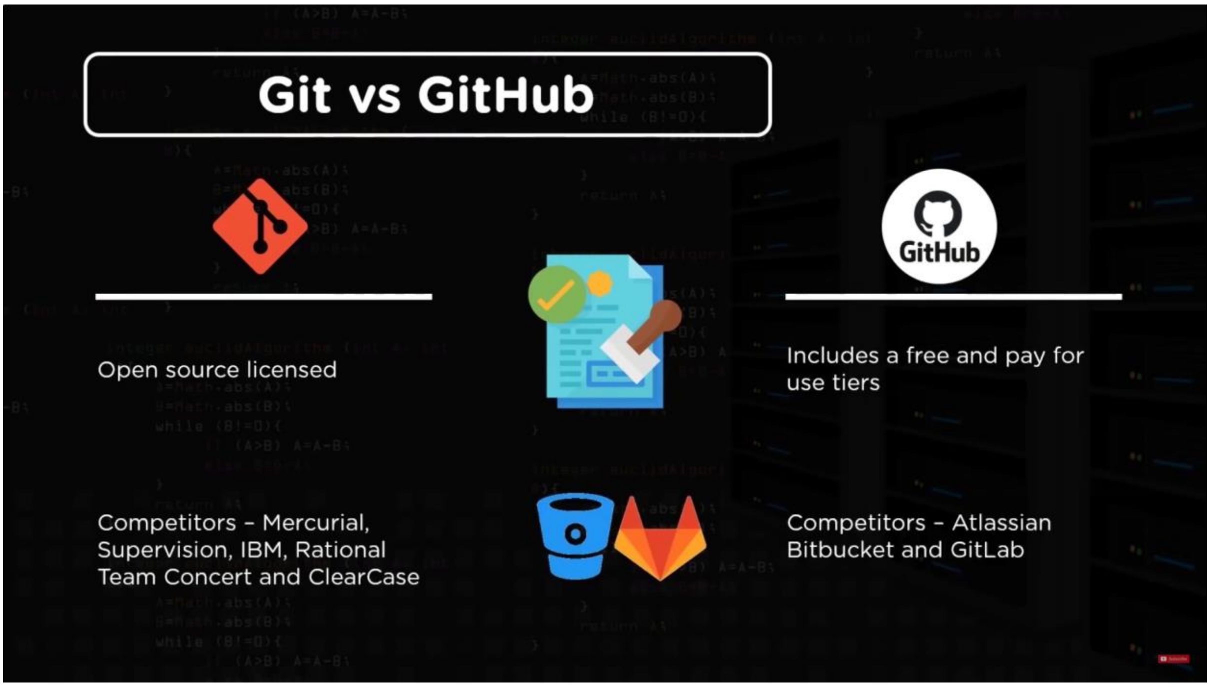
Git vs GitHub

There is no GUI

Code changes like commit, merge, etc. are done using commands from the command line.

It provides an easy to use GUI

Everything is done through a web-based interface.



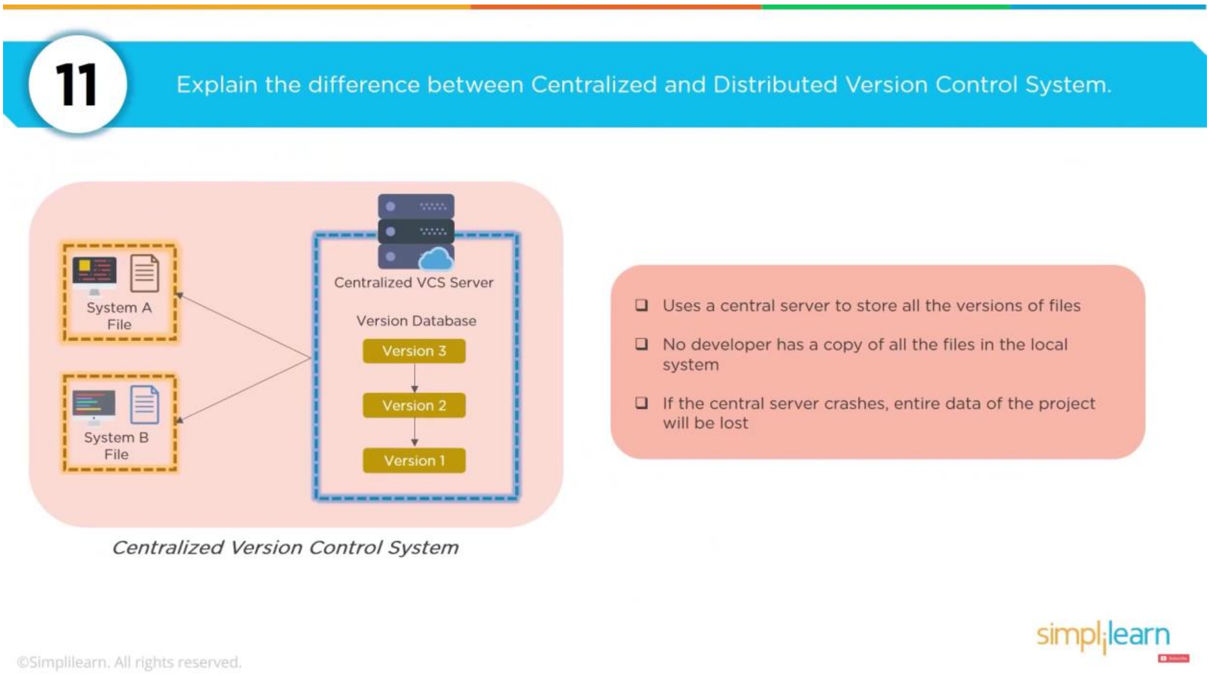
Git vs GitHub

Open source licensed

Competitors - Mercurial, Supervision, IBM, Rational Team Concert and ClearCase

Includes a free and pay for use tiers

Competitors - Atlassian Bitbucket and GitLab



11 Explain the difference between Centralized and Distributed Version Control System.

System A File

System B File

Centralized VCS Server

Version Database

Version 3

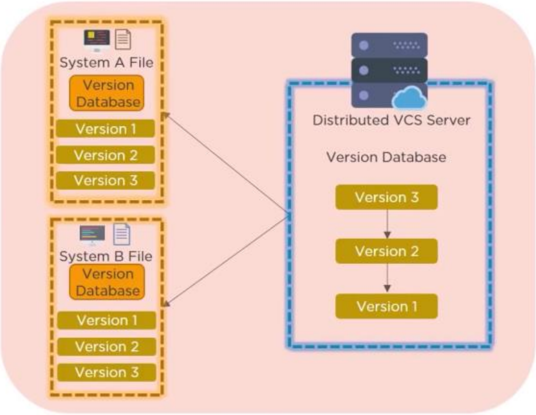
Version 2

Version 1

Centralized Version Control System

- ❑ Uses a central server to store all the versions of files
- ❑ No developer has a copy of all the files in the local system
- ❑ If the central server crashes, entire data of the project will be lost

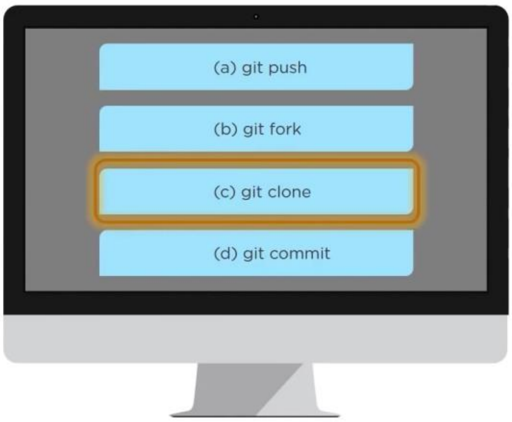
11 Explain the difference between Centralized and Distributed Version Control System.



Distributed Version Control System

- ❑ Every developer has a copy of all the versions of the code on their system
- ❑ Improves the ability to work offline and does not rely on single location for backups
- ❑ There is no threat even if the server crashes

12 What is the git command that downloads any repository from GitHub to your computer?



(a) git push

(b) git fork

(c) git clone

(d) git commit

13 How to push a file from your local system on to GitHub repository using Git?

First, connect the local repository to your remote repository:

```
git remote add origin [copied web address]
```

→ *git remote add origin https://github.com/Simplilearn-github/test.git*

Second, push your file to the remote repository:

```
git push origin master
```

14 How is bare repository different from the standard way of initializing a Git repository?

Using the standard way

```
git init
```

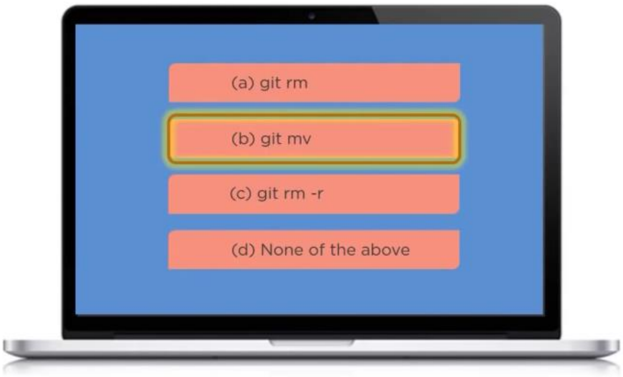
- You create a working directory with git init
- A .git subfolder is created with all the git related revision history

Using the bare way

```
git init --bare
```

- Does not contain any working or checked out copy of source files
- Bare repositories store git revision history in the root folder of your repository instead of .git subfolder

15 Which of the following CLI command can be used to rename files?



(a) git rm

[✓ (b) git mv]

(c) git rm -r

(d) None of the above

16 What is the process to revert a commit that has already been pushed and made public?

There are two processes through which you can revert a commit:

1 Remove or fix the bad file in a new commit and push it to the remote repository. Then commit it to the remote repository using:

git commit –m “commit message”

2 Create a new commit that undoes all the changes that were made in the bad commit. Use the following command:

git revert <commit id>

Example: git revert 56de0938f

17 Explain the difference between git fetch and git pull.

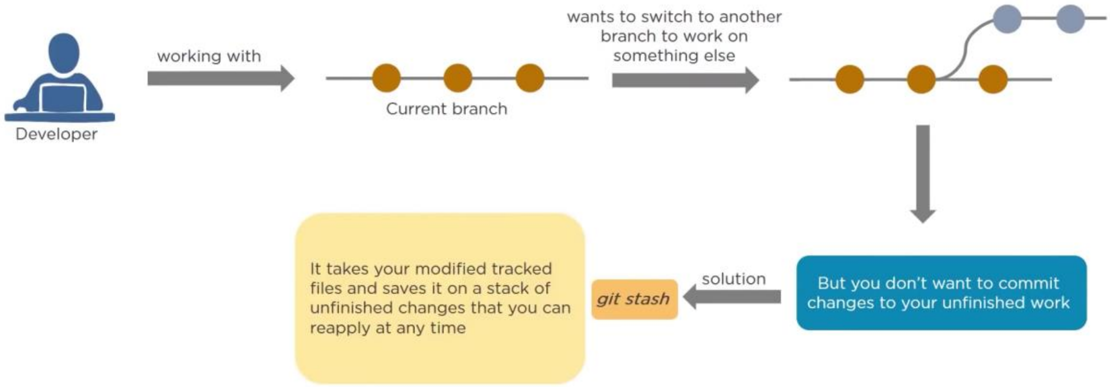
Git fetch

- Git fetch only downloads new data from a remote repository
- Does not integrate any of this new data into your working files
- Git fetch can be done any time to update the remote-tracking branches
- *Command - git fetch origin*
  git fetch --all

Git pull

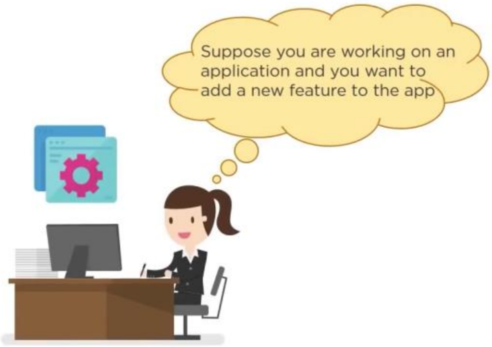
- Git pull updates the current HEAD branch with the latest changes from the remote server
- It downloads new data and integrates it with the current working files
- It tries to merge remote changes with your local ones
- *Command - git pull origin master*

18 What is Git stash?

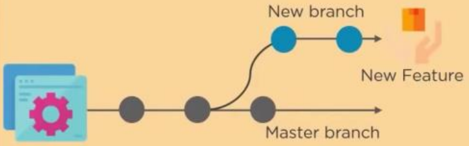

Developer
working with
Current branch
wants to switch to another branch to work on something else
But you don't want to commit changes to your unfinished work
solution
git stash
It takes your modified tracked files and saves it on a stack of unfinished changes that you can reapply at any time

19 Explain the concept of branching in Git.

Suppose you are working on an application and you want to add a new feature to the app



You can create a new branch and build the new feature on that branch

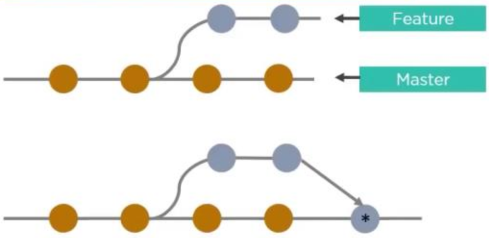


- ❑ By default, you always work on the master branch
- ❑ The circles on the branch represent various commits made on the branch
- ❑ After you are done with all the changes, you can merge it to the master branch

20 What is the difference between git merge and git rebase.

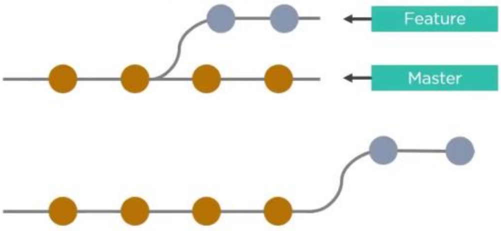
Suppose you are working on a new feature in a dedicated branch and another team member updates the master branch with new commits

To incorporate the new commits into your feature branch, you use ***merge***



- Creates an extra merge commit every time you need to incorporate changes
- Pollutes your feature branch history

As an alternative to merging, you can rebase the feature branch onto master



- Incorporates all the new commits in master branch
- Re-writes the project history by creating brand new commits for each commit in the original branch

21 How do you find a list of files that has been changed in a particular commit?

The command to get a list of files that has been changed in a particular commit is:

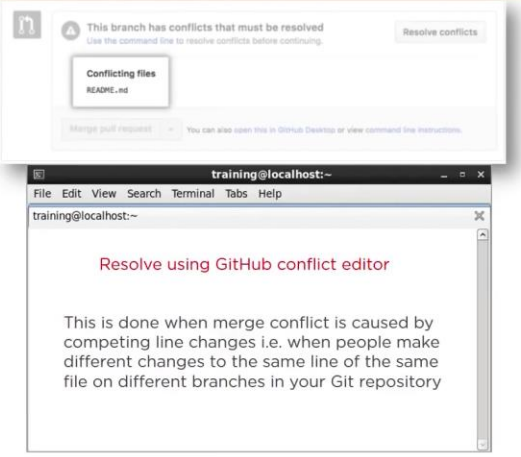
git diff-tree –r {commit hash}

Example: git diff-tree –r 87e673f21b

- *-r* flag allows the command to list individual files
- *commit hash* will list all the files that were changed or added in that commit

22 What is a merge conflict in Git and how can it be resolved?

Git merge conflict It arises when you have merge branches that have competing commits and Git needs your help to decide which changes to incorporate in the final merge


This branch has conflicts that must be resolved
Use the command line to resolve conflicts before continuing.
Resolve conflicts
Conflicting files
README.md
Merge pull request
You can also open this in GitHub Desktop or view command line instructions.
training@localhost:~
File Edit View Search Terminal Tabs Help
training@localhost:~

Resolve using GitHub conflict editor

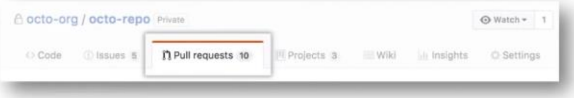
This is done when merge conflict is caused by competing line changes i.e. when people make different changes to the same line of the same file on different branches in your Git repository

Manually edit the conflicted file to select the changes that you want to keep in the final merge

22 What is a merge conflict in Git and how can it be resolved?

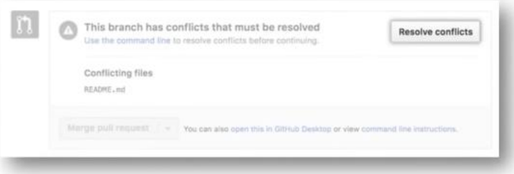
Resolving a merge conflict using conflict editor

1 Under your repository name, click Pull requests.



2 In "**Pull Requests**" list, click the pull request with a merge conflict that you'd like to resolve

3 Near the bottom of your pull request, click Resolve conflicts


This branch has conflicts that must be resolved

Use the command line to resolve conflicts before continuing.

Conflicting files

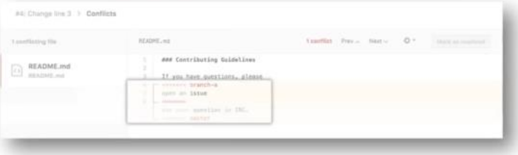
README.md

You can also open this in GitHub Desktop or view command line instructions.

4 Decide if you want to keep only your branch's changes, keep only the other branch's changes or make a brand new change, which may incorporate changes from both branches

22 What is a merge conflict in Git and how can it be resolved?

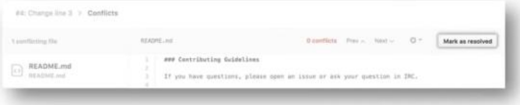
5 Delete the conflict markers `<<<<<<<`, `=======`, `>>>>>>>` and make changes you want in the final merge



```
### Contributing Guidelines

If you have questions, please
<<<<<<< branch-a
open an issue
=======
ask your question in IRC.
>>>>>>> master
```

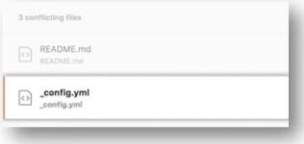
6 If you have more than one merge conflict in your file, scroll down to the next set of conflict markers and repeat steps four and five to resolve your merge conflict

7 Once you have resolved all the conflicts in the file, click **Mark as resolved**



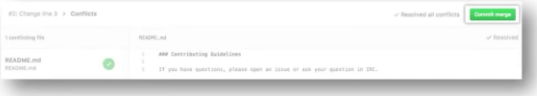
```
### Contributing Guidelines

If you have questions, please open an issue or ask your question in IRC.
```

8 If you have more than one file with a conflict, select the next file you want to edit on the left side of the page under "**conflicting files**" and repeat steps 4 to 7 until you've resolved all of your pull request's merge conflicts



22 What is a merge conflict in Git and how can it be resolved?

9 Once you've resolved all your merge conflicts, click **Commit merge**. This merges the entire base branch into your head branch



```
1   ### Contributing Guidelines
2
3   If you have questions, please open an issue or ask your question in IRC.
```

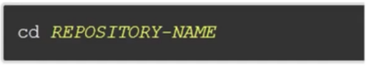
10 To merge your pull request, click Merge pull request

22 What is a merge conflict in Git and how can it be resolved?

Resolving a merge conflict using the command line

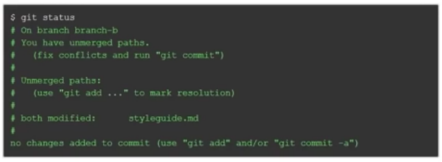
1 Open Git Bash

2 Navigate into the local Git repository that has the merge conflict

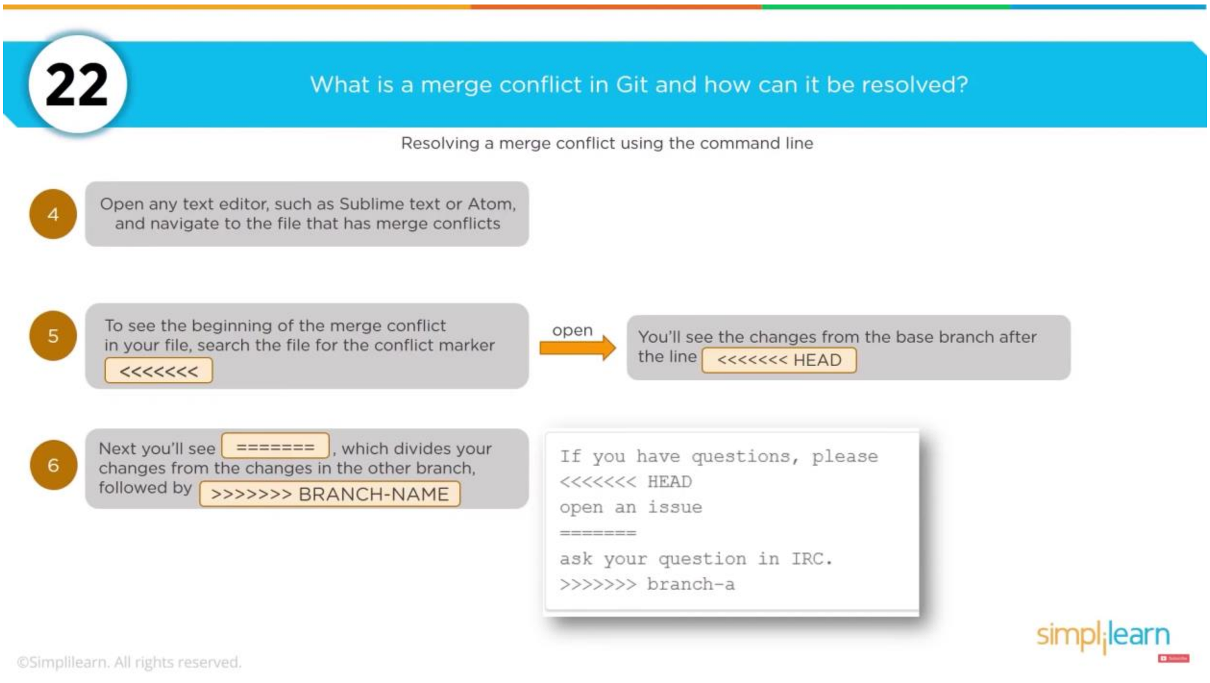


```
cd REPOSITORY-NAME
```

3 Generate a list of the files affected by the merge conflict. In this example, the file **styleguide.md** has a merge conflict



```
$ git status
# On branch branch-b
# You have unmerged paths.
#   (fix conflicts and run "git commit")
#
# Unmerged paths:
#   (use "git add ..." to mark resolution)
#
# both modified:      styleguide.md
#
no changes added to commit (use "git add" and/or "git commit -a")
```



22

What is a merge conflict in Git and how can it be resolved?

Resolving a merge conflict using the command line

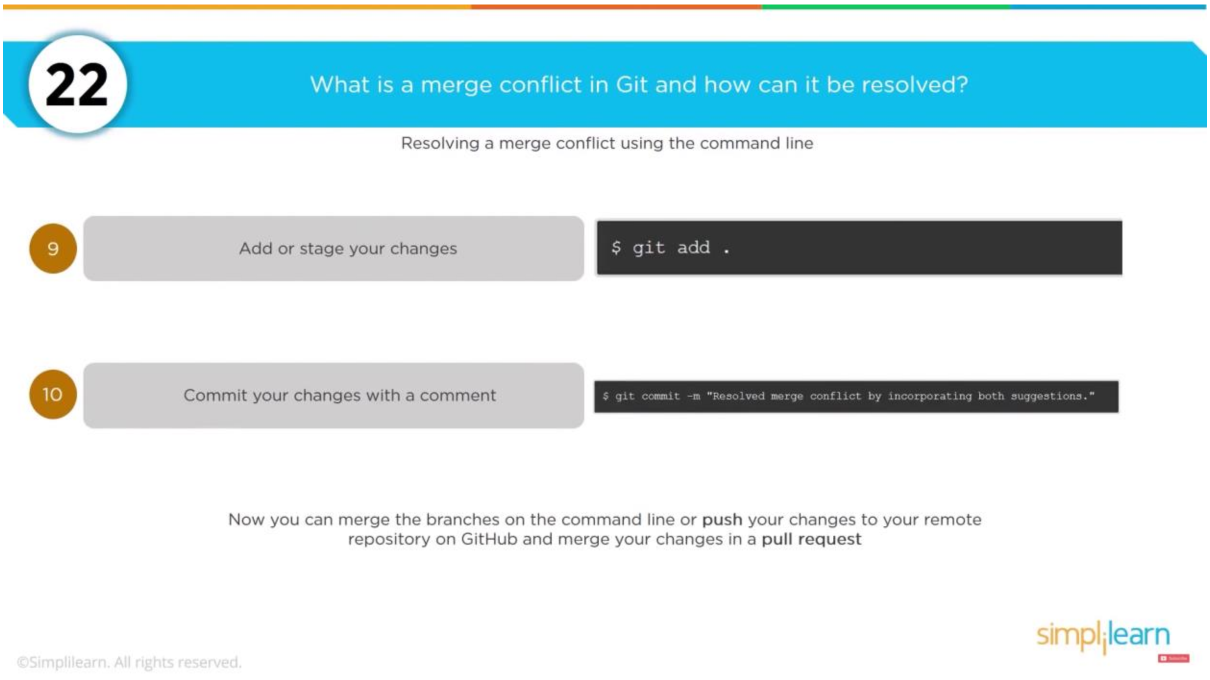
4 Open any text editor, such as Sublime text or Atom, and navigate to the file that has merge conflicts

5 To see the beginning of the merge conflict in your file, search the file for the conflict marker `<<<<<<<`

open → You'll see the changes from the base branch after the line `<<<<<<< HEAD`

6 Next you'll see `=======`, which divides your changes from the changes in the other branch, followed by `>>>>>>> BRANCH-NAME`

```
If you have questions, please
<<<<<<< HEAD
open an issue
=======
ask your question in IRC.
>>>>>>> branch-a
```



22

What is a merge conflict in Git and how can it be resolved?

Resolving a merge conflict using the command line

9 Add or stage your changes

```
$ git add .
```

10 Commit your changes with a comment

```
$ git commit -m "Resolved merge conflict by incorporating both suggestions."
```

Now you can merge the branches on the command line or **push** your changes to your remote repository on GitHub and merge your changes in a **pull request**

Why it is preferred to commit a feature in a separate branch and not on main/master branch?

It is preferred because, the main/master branch will not get affected by whatever change made by the feature. Till the feature is reviewed & verified it won't be merged with main branch.

After review with the pull request, the feature will be merged with the main branch

Can you make changes to anyone account on github?

No, to use anyone code, use forking option in github which will clone the entire project to your account, from where we can modify and play with it by cloning the project to our local system.

The url from which the project is forked is called upstream, whereas the URL of my github repo will be known as origin.

Can one branch have more than one pull requests?

No, for each branch there can be only one pull requests and whenever a new feature or bug is worked on, It is always recommended to create a new branch and create a pull request. The old branch will not create PR and it will be difficult when there are more than one change in one branch to review.