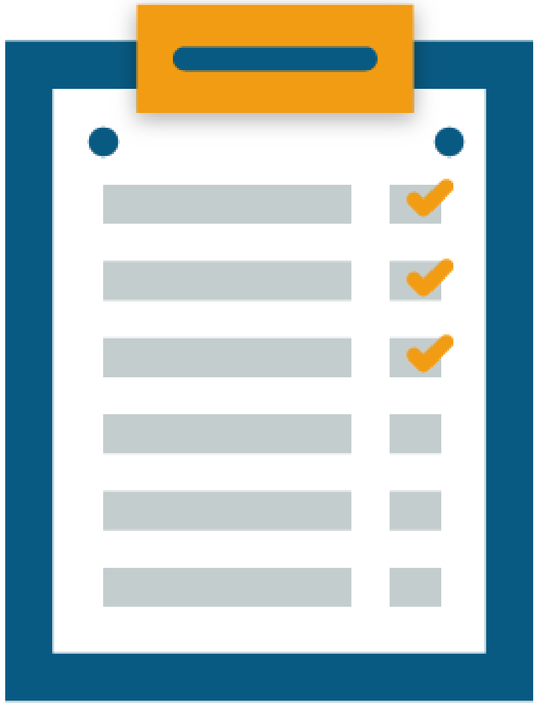
edureka!

Version Control with Git - I

Topics

Following are the topics covered in this module:

- Version Control
- Git Introduction
- Git Installation
- Commonly used commands in Git
- Demo: Basic Git Commands

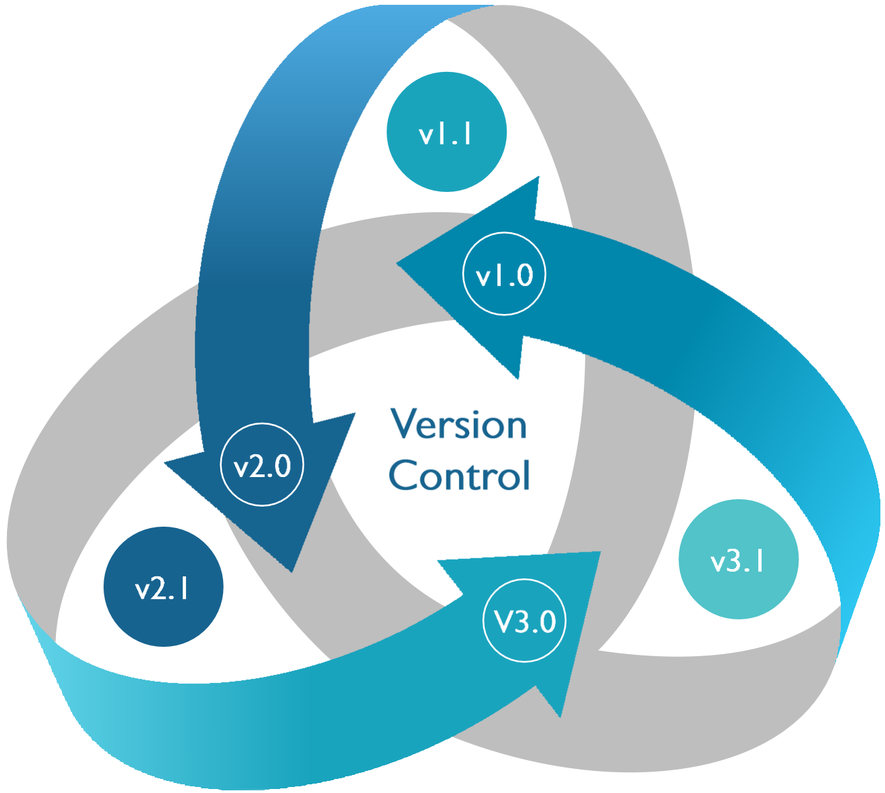


edureka!

Version Control

What Is Version Control?

Version Control is a system that documents changes made to a file or a set of files. It allows multiple users to manage multiple revisions of the same unit of information. It is a snapshot of your project over time.

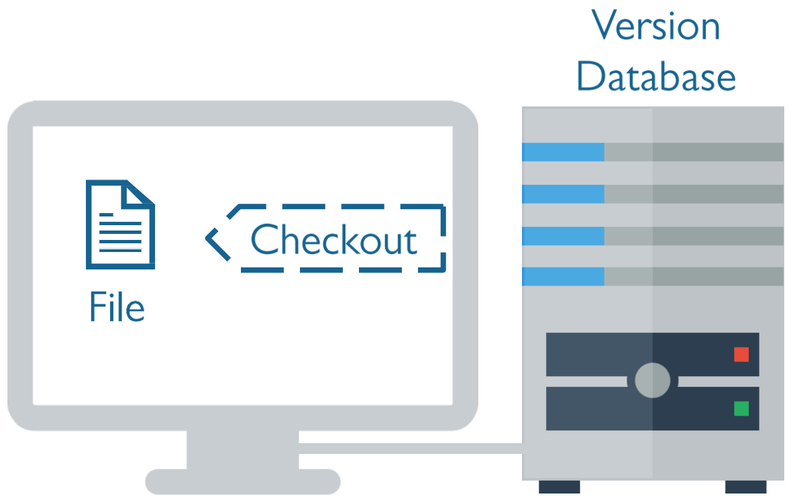


Version Control: Types

1. Local
2. Centralized
3. Distributed

Local Version Control (LVC)

- The practice of having the Version Database in the local computer
- Local database keeps a record of the changes made to files in version database



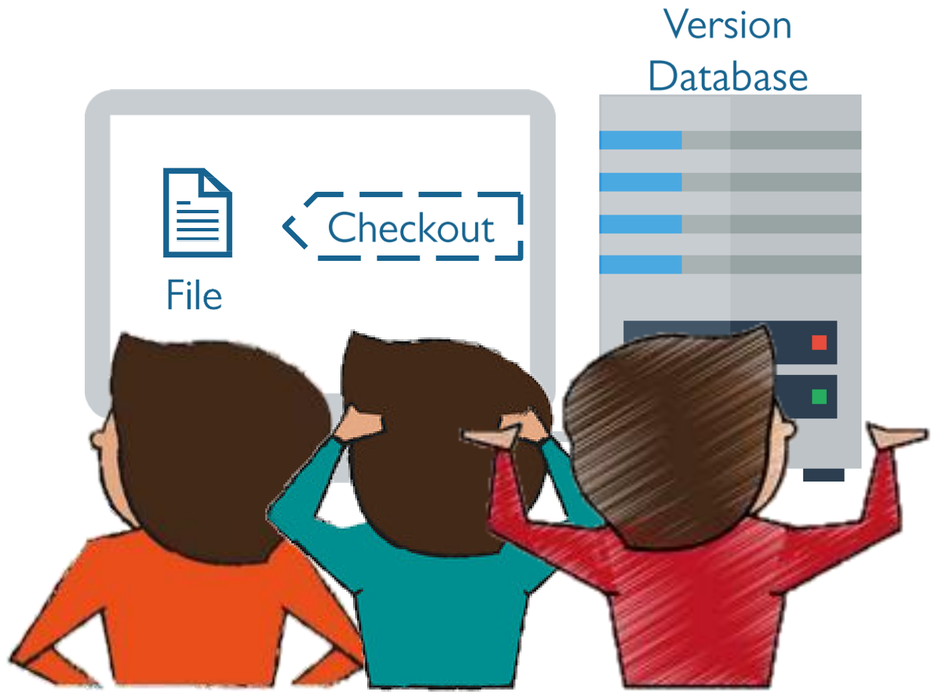
edureka!

Version Control: Types

1. Local
2. Centralized
3. Distributed

Local Version Control: Issue

- **Issue:** Multiple people parallelly working on the same project
- **Solution:** Centralized Version Control



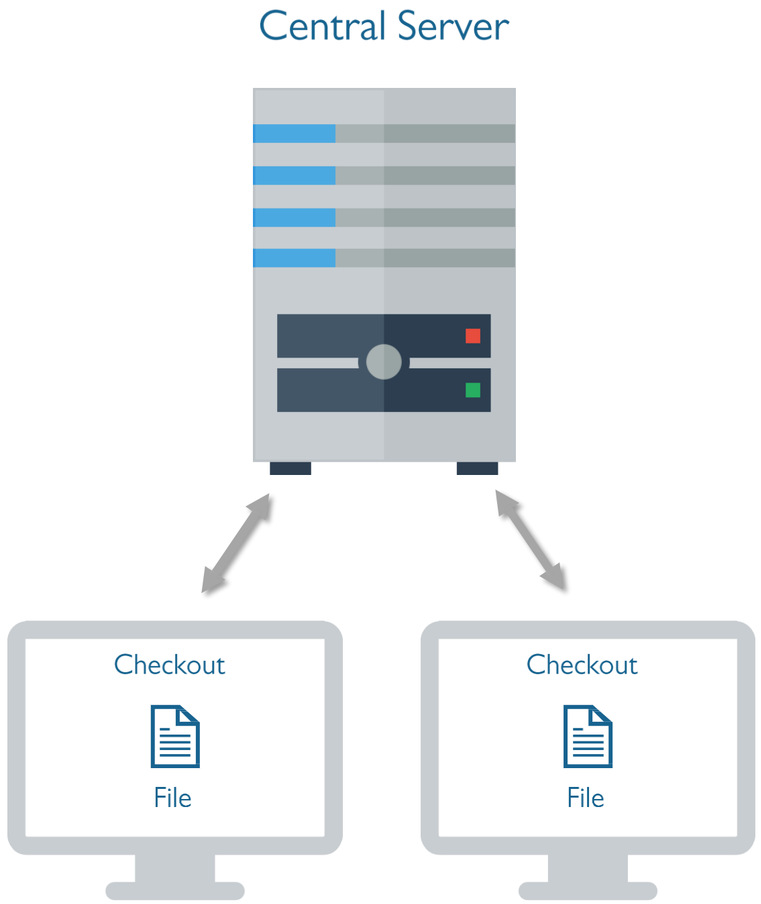
edureka!

Version Control: Types

1. Local
2. Centralized
3. Distributed

Centralized Version Control (CVC)

- Local Version Control's issues are resolved by Centralized Version Control
- In CVC, a central repository is maintained where all the versioned files are kept
- Now users can checkout, and check-in files from their different computers at any time



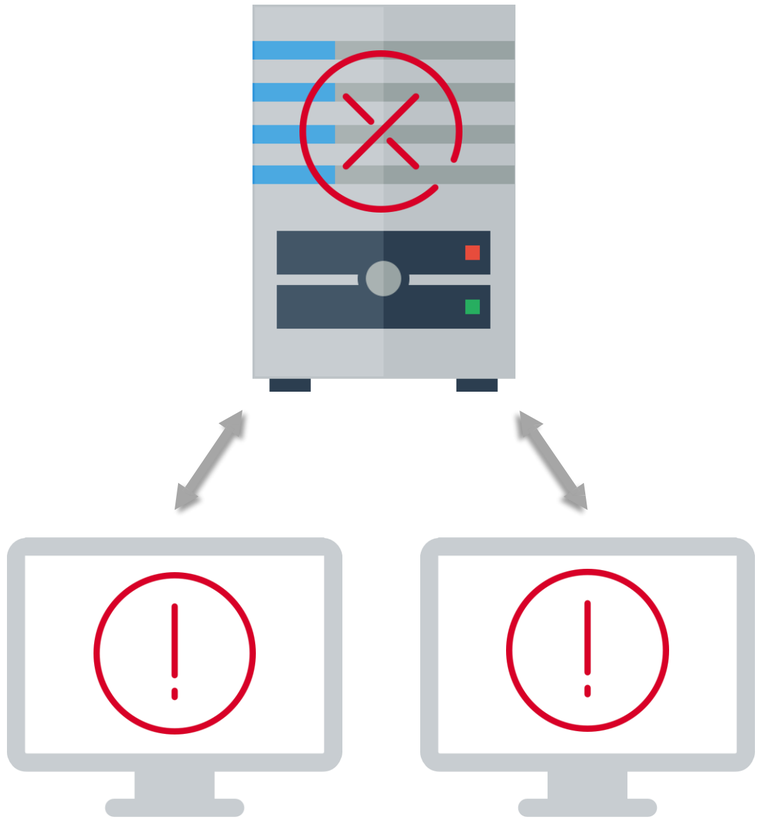
edureka!

Version Control: Types

1. Local
2. Centralized
3. Distributed

Centralized Version Control: Issue

- **Issue:** In case of central server failure whole system goes down
- **Solution:** Distributed Version Control

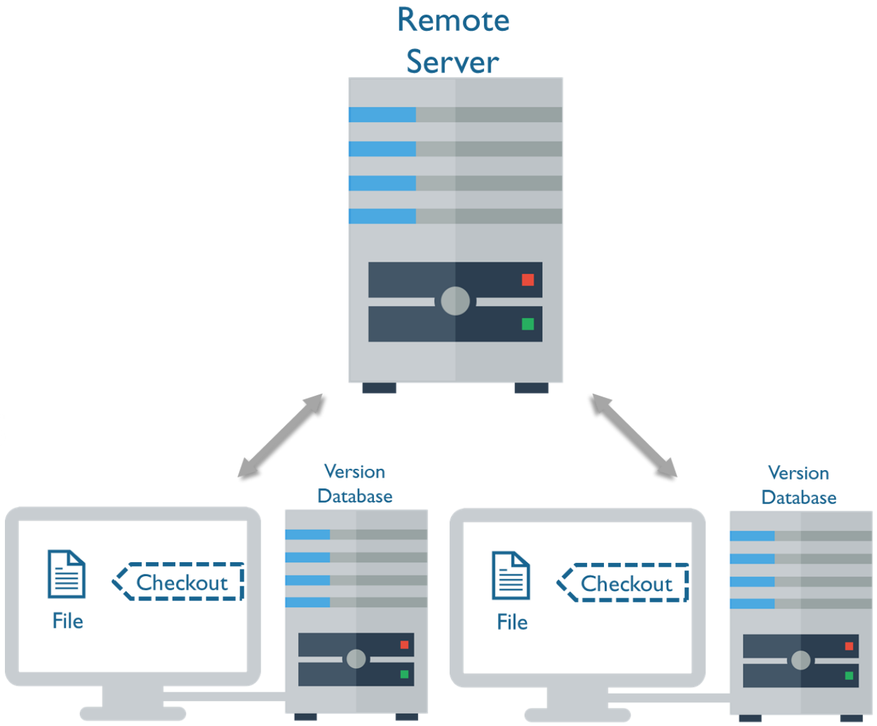


Version Control: Types

1. Local
2. Centralized
3. Distributed

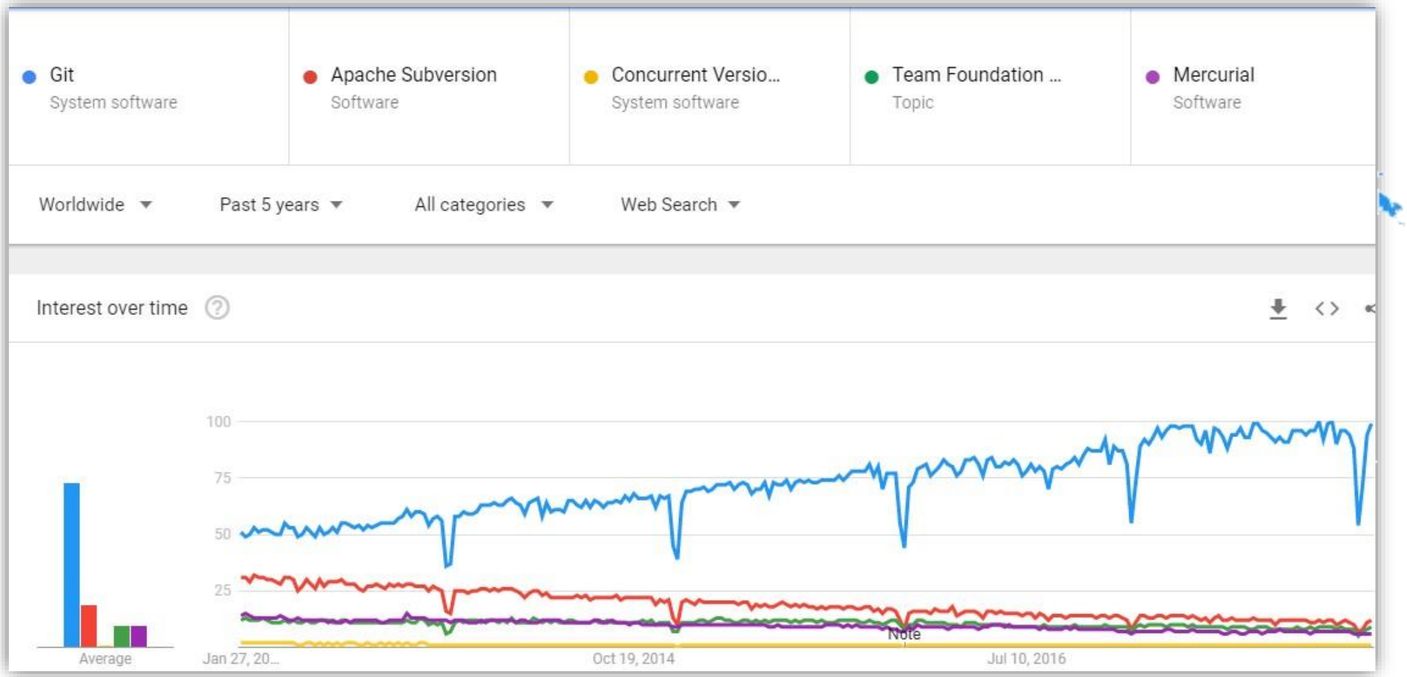
Distributed Version Control

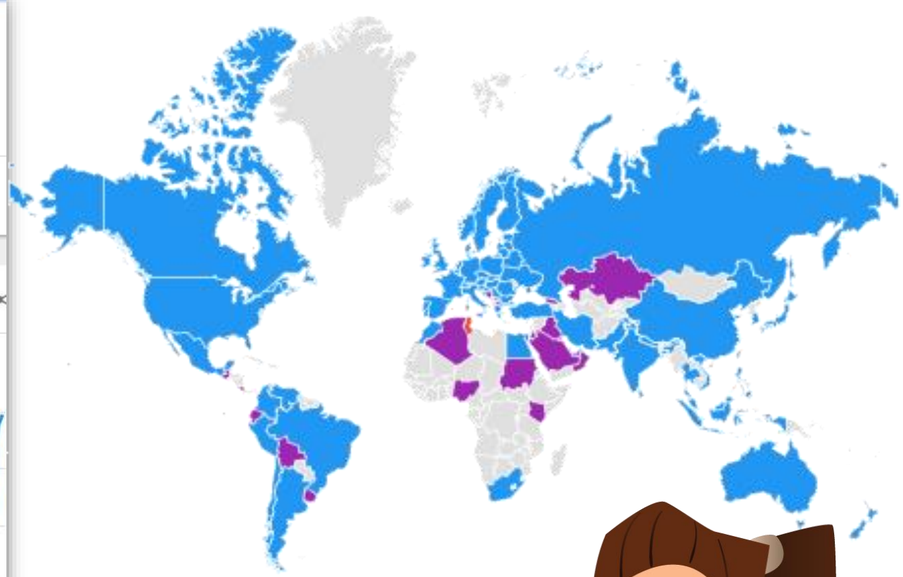
- Version Database is stored at every users' local system and at remote server
- Users manipulate the local files and then upload the changes to the remote server
- If any of the server dies, a client server can be used to restore



edureka!

Why Git?





Note: The above graph shows the usability of the popular VCS tools in the past 5 years

Why Git Is A Clear Winner?

Snapshots

Git records changes made to a file rather than file itself. That means if a file isn't changed it isn't stored again.

Fast Operations

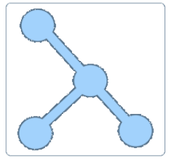
Almost every operation on git is local, hence the speed offered by Git is lightening fast compared to other VCS's.

Distributed

Every user has his own copy of the repository data stored locally allowing full functionality even on disconnection.

Branch Handling

Every collaborator's working directory is in itself a branch. Different branches can be merged with ease.



Integrity

Check-sum before storing ensures that you can't make any changes to anything without Git recording that change.

Robust

Nearly every task in Git is undo-able and it is really hard to loose any change or data in Git.

Introduction To Git

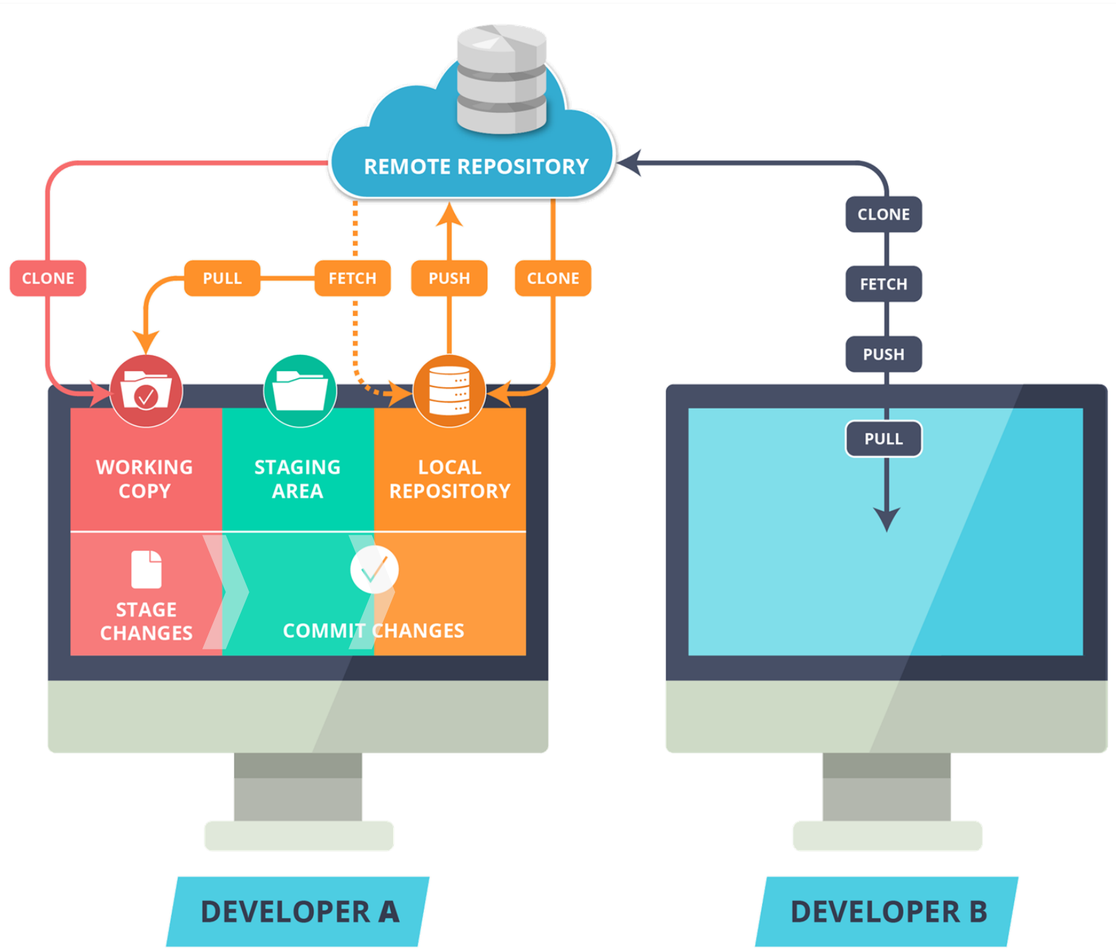
edureka!

What Is Git?

Git is an open source Distributed Version Control System(DVCS) which records changes made to the files laying emphasis on **speed**, **data integrity** and **distributed**, **non-linear workflows**

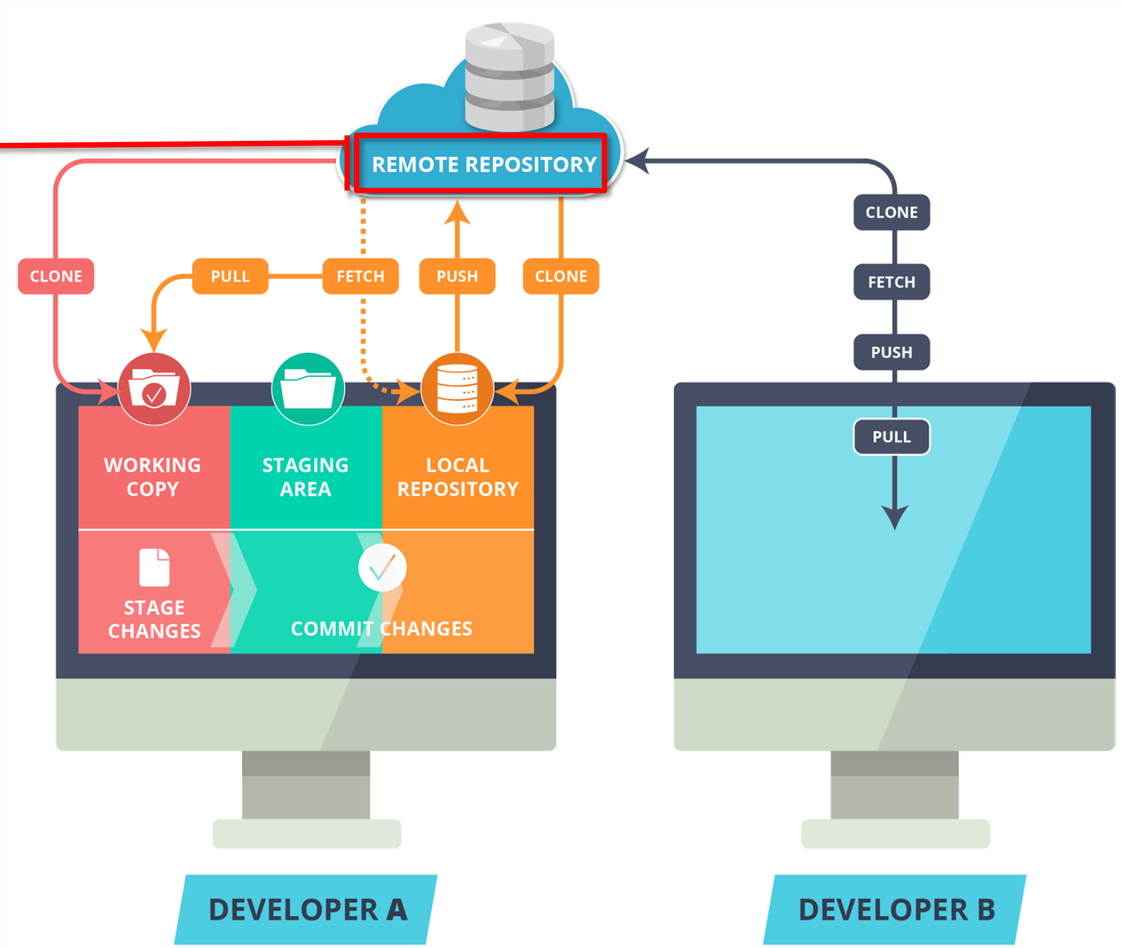
The Git File Workflow

- Use Git workflow to manage your project effectively
- Working with set of guidelines increases Git's consistency and productivity



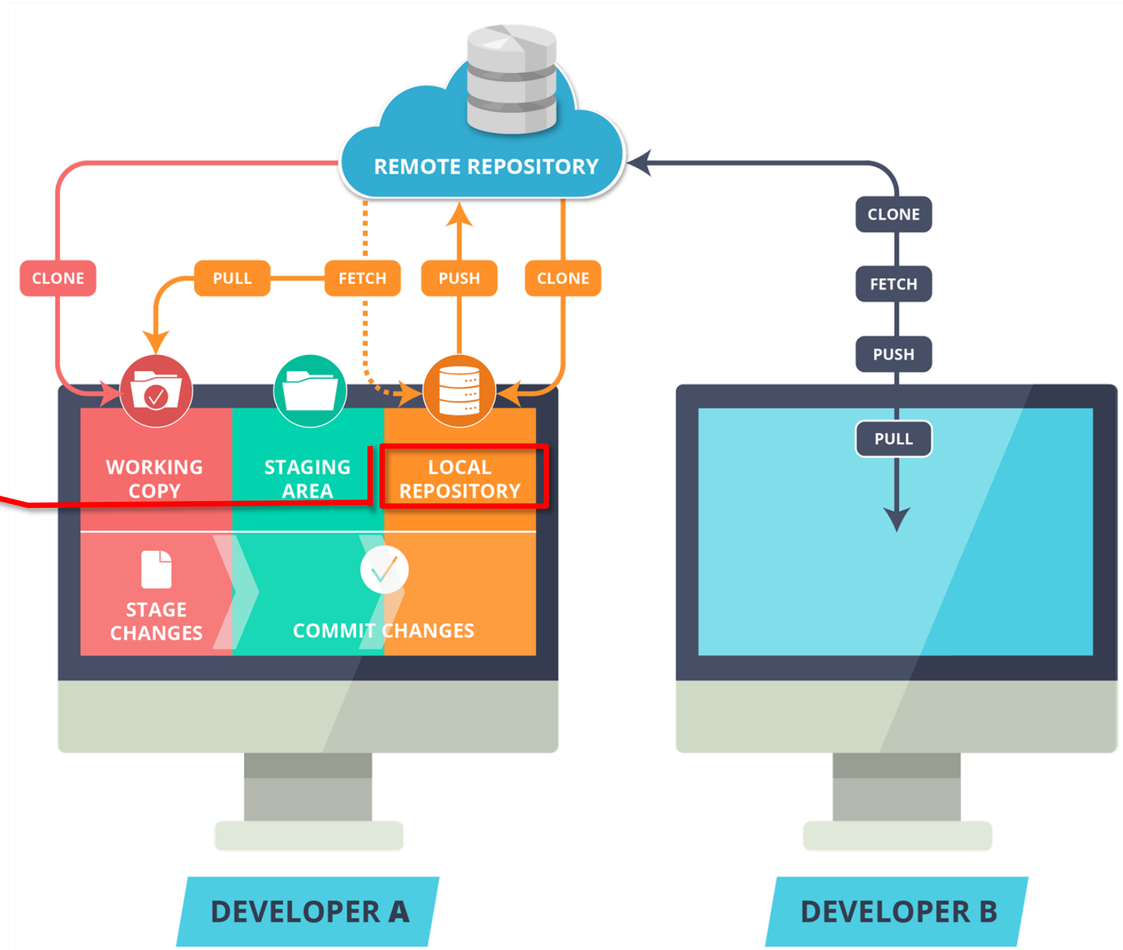
The Git File Workflow

The Remote Repository is the server where all the collaborators upload changes made to the files



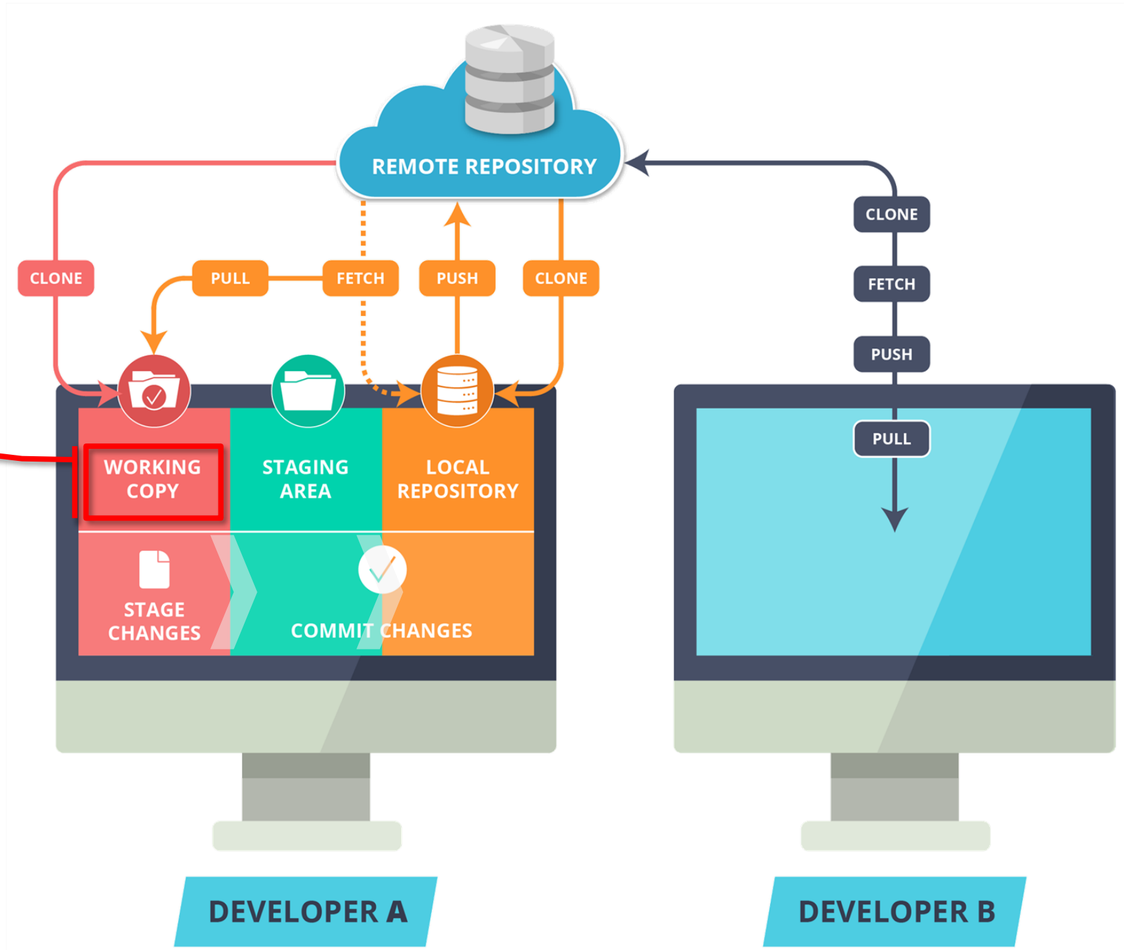
The Git File Workflow

- "Local Repository" is user's copy of the Version Database
- The user accesses all the files through local repository and then push the change made to the "Remote Repository"



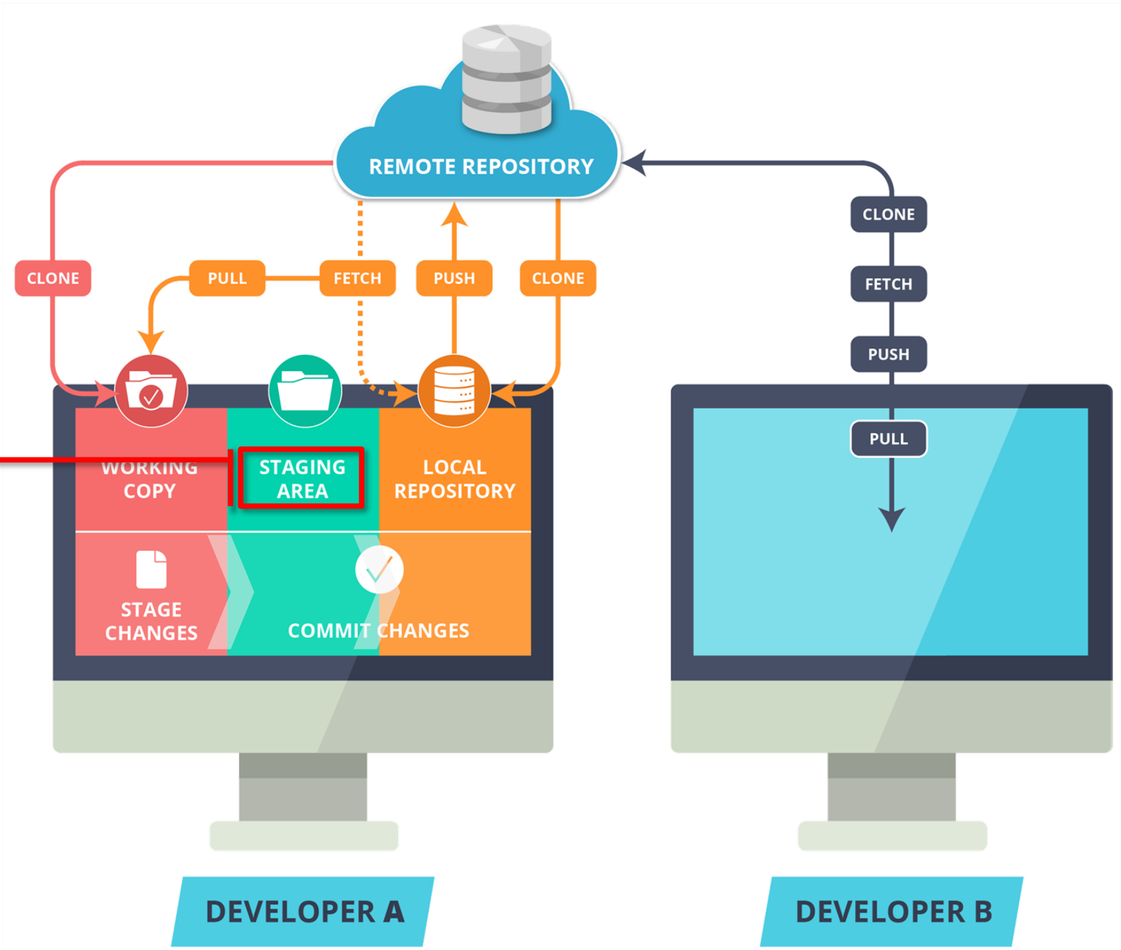
The Git File Workflow

- **"Workspace"** is user's active directory
- The user modifies existing files and creates new files in this space. Git tracks these changes compared to your Local Repository



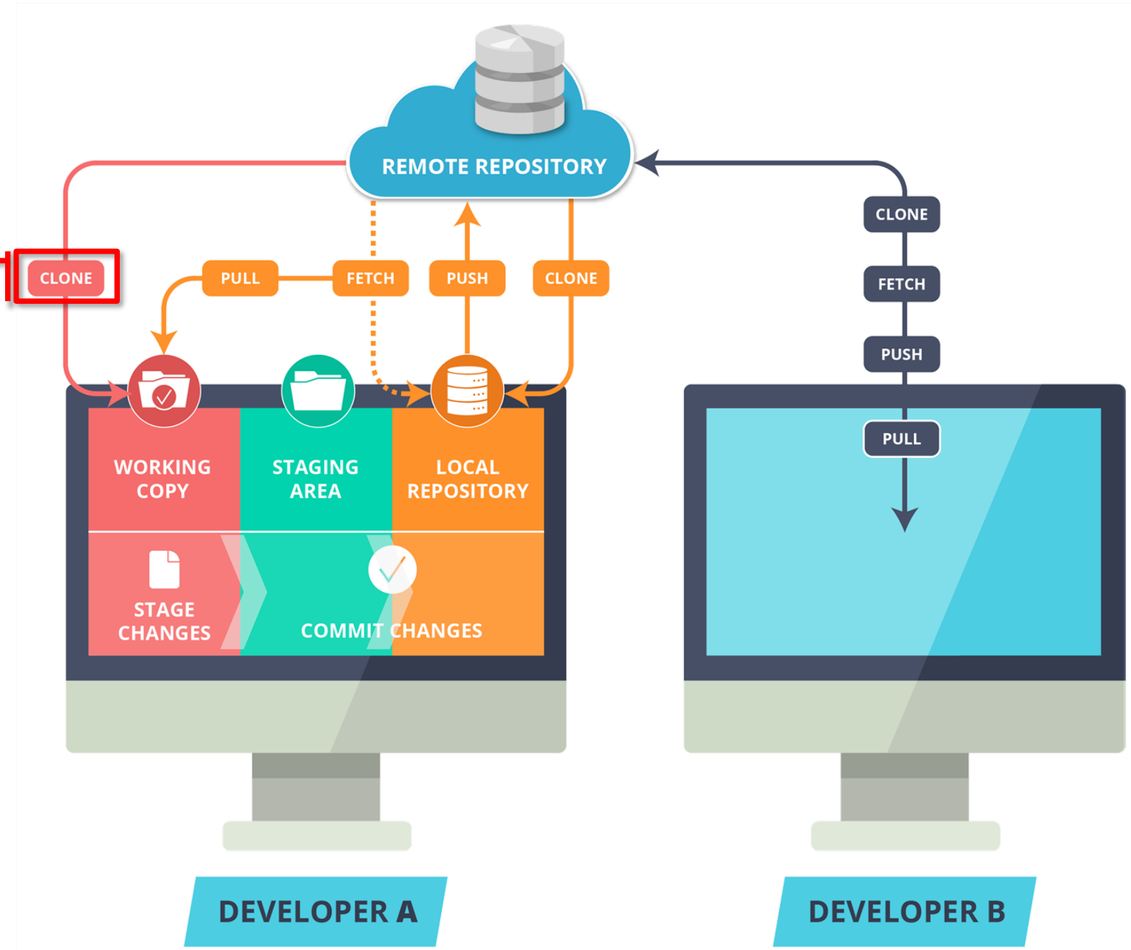
The Git File Workflow

Stage is a place where all the modified files marked to be committed are placed.



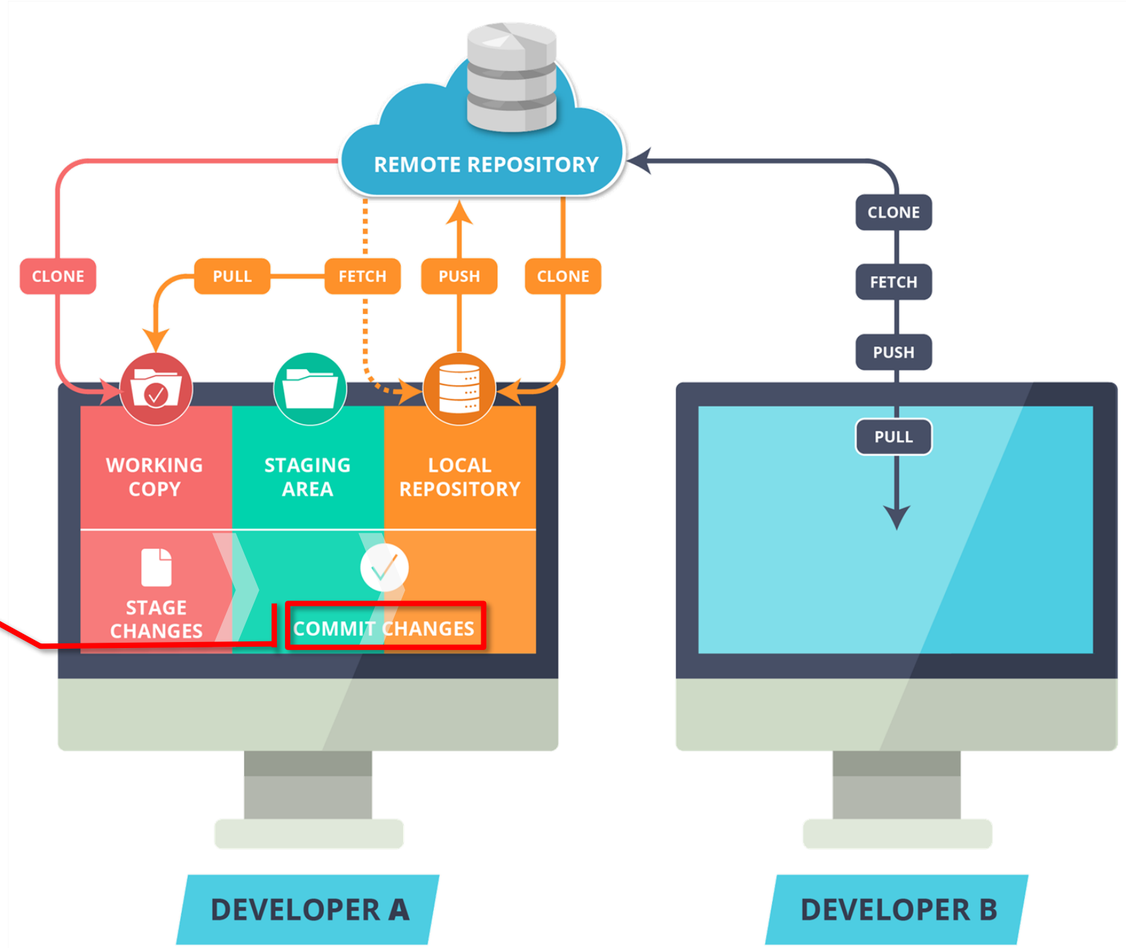
The Git File Workflow

Clone command creates a copy of an existing Remote Repository inside the Local Repository.



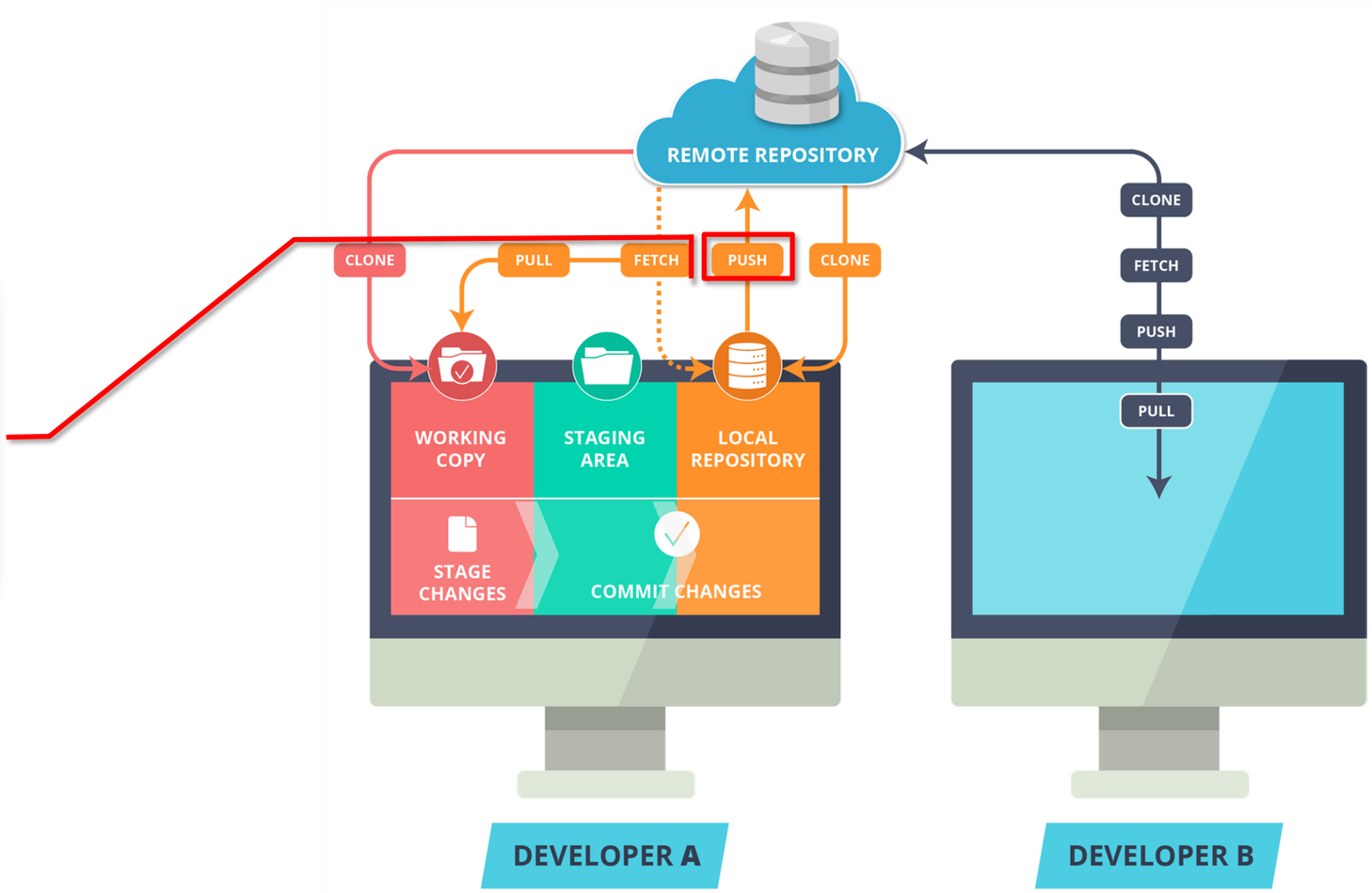
The Git File Workflow

Commit command commits all the files in the staging area to the local repository.



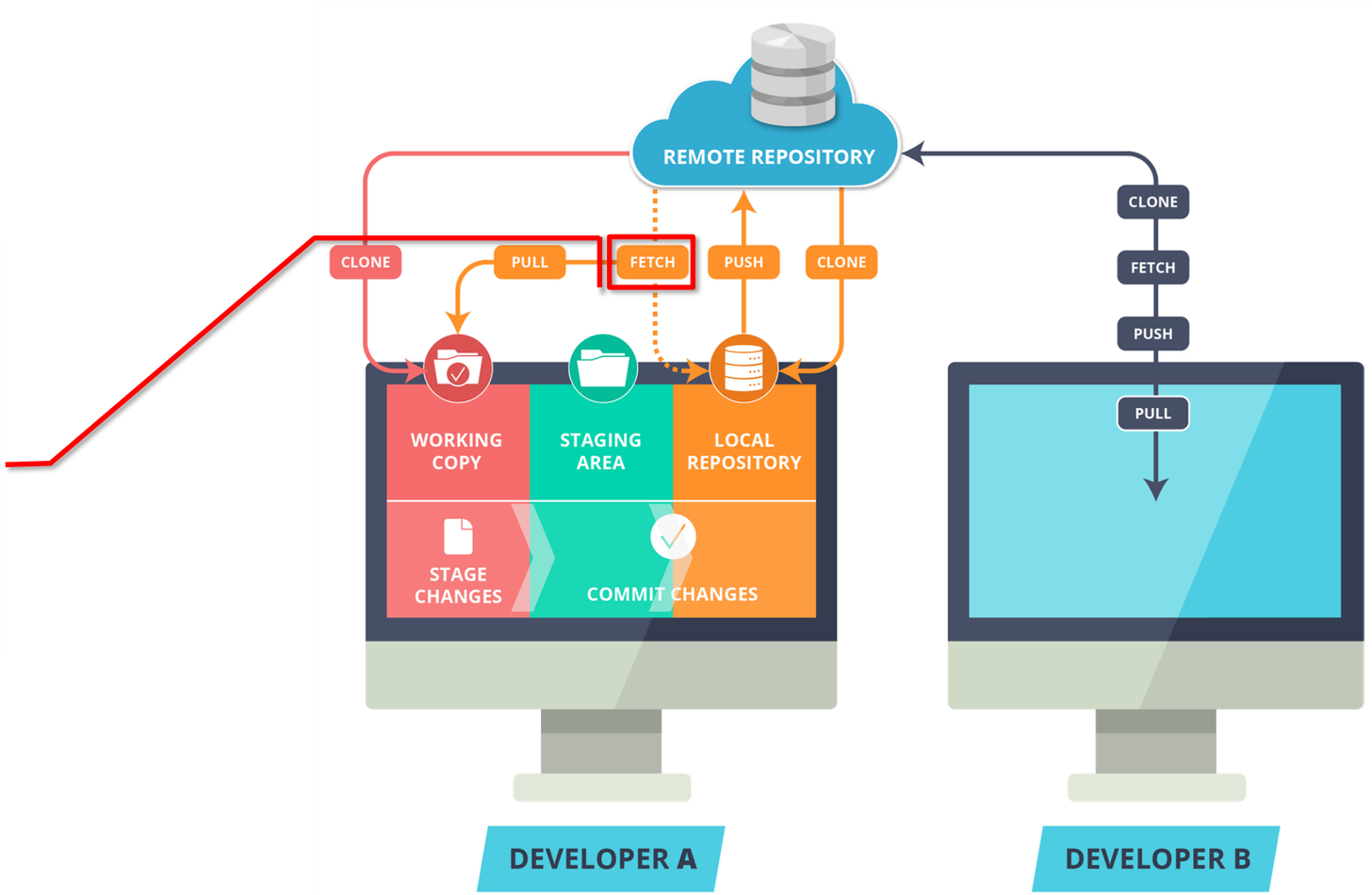
The Git File Workflow

Push command pushes all the changes made in the Local Repository to the Remote Repository



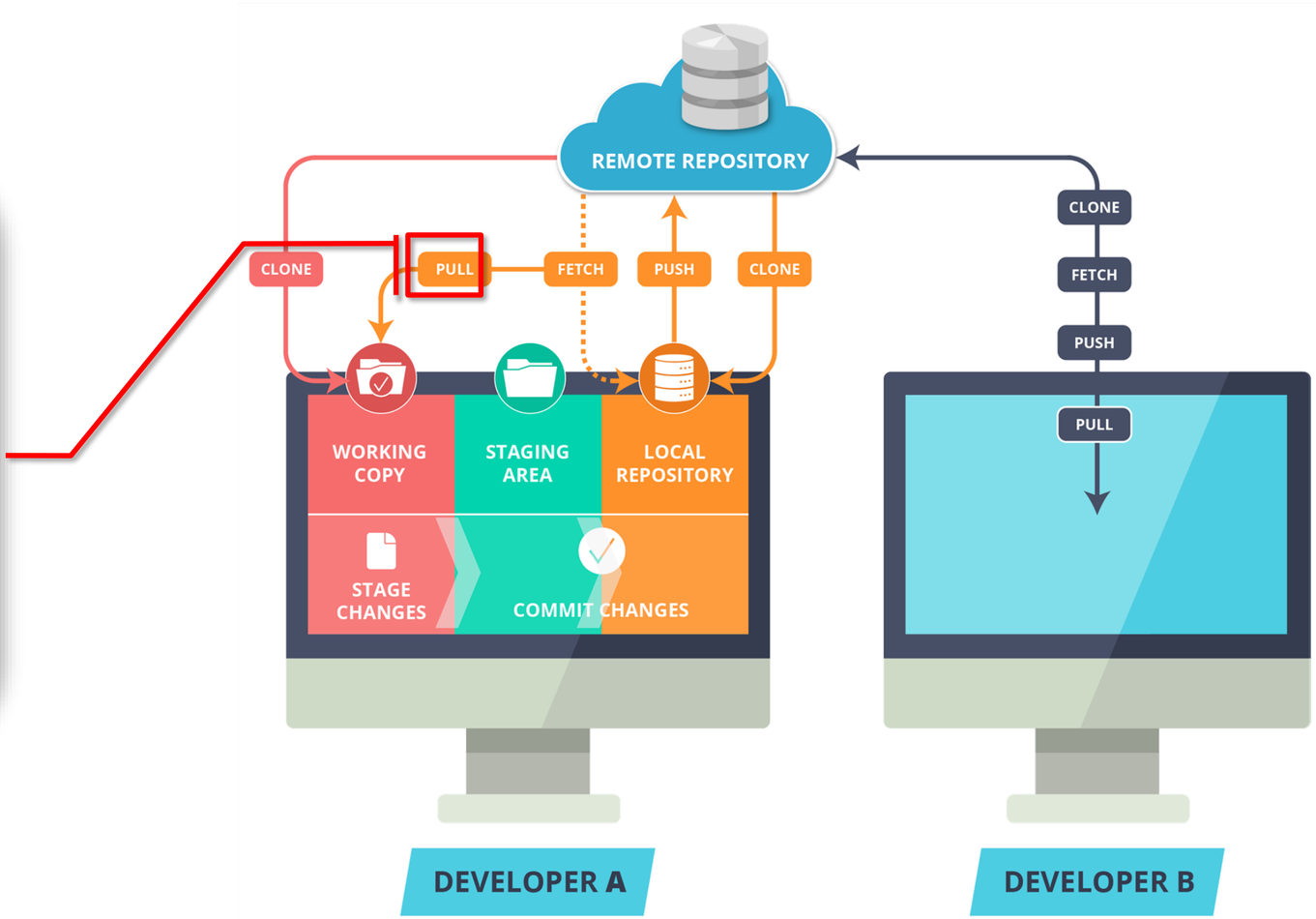
The Git File Workflow

Fetch command collects the changes made in the Remote repository and copies them to the Local Repository. This command doesn't affect our Workspace.



The Git File Workflow

- Pull like Fetch, gets all the changes from the remote repository and copies them to the Local Repository
- Pull merges those changes to the current working directory



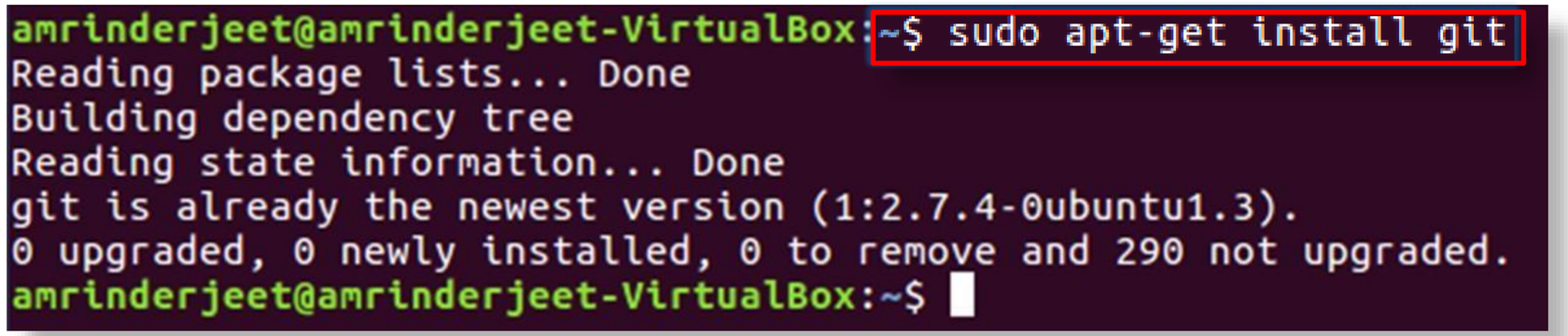
Git Common Commands

edureka!

Installing Git On Ubuntu

- To install Git on your Linux Machine you can type in the following command in Terminal:

```
Syntax: sudo apt-get install git
```



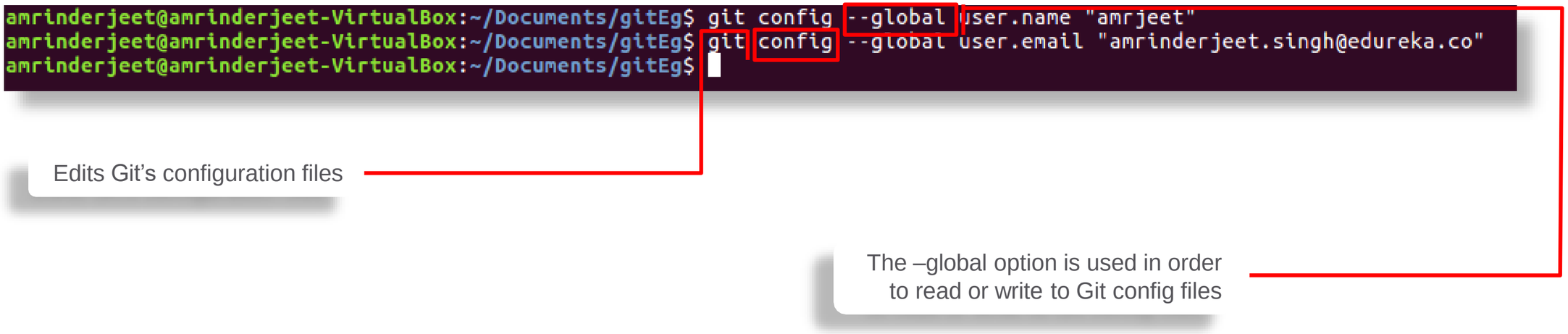
Setting Up User

- Set up username for your repository

```
Syntax: git config --global user.name “username”
```

- Set up user-email for your repository

```
Syntax: git config --global user.email “useremail@example.com”
```



Create Your Local Repository

- Select or create the Directory where you want to initialize Git
- **Initialize Git** in the Directory

```
Syntax: git init
```

```
amrinderjeet@amrinderjeet-VirtualBox:~/Documents$ mkdir demo
amrinderjeet@amrinderjeet-VirtualBox:~/Documents$ cd demo
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git init
Initialized empty Git repository in /home/amrinderjeet/Documents/demo/.git/
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$
```

Initializes Git in the selected Directory

Files in the Directory

Git folder initialized

```
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ ls -a
.  ..  Demo.class  Demo.java  .git
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$
```

Adding Files And Checking Status

- To **add** a file to the **staging area**

```
Syntax: git add <filename>
```

- To check the working tree **status**

```
Syntax: git status
```

```
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git add Demo.java
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   Demo.java

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        Demo.class

amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$
```

Committing Changes

To **commit** the **staged** files to you **local repository**:

```
Syntax: git commit
```

```
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git commit
```

```
 GNU nano 2.5.3                        File: /home/amrinderjeet/Documents

Added the first file to the repo
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
#
# Initial commit
#
# Changes to be committed:
#       new file:   Demo.java
#
# Untracked files:
#       Demo.class
#
```

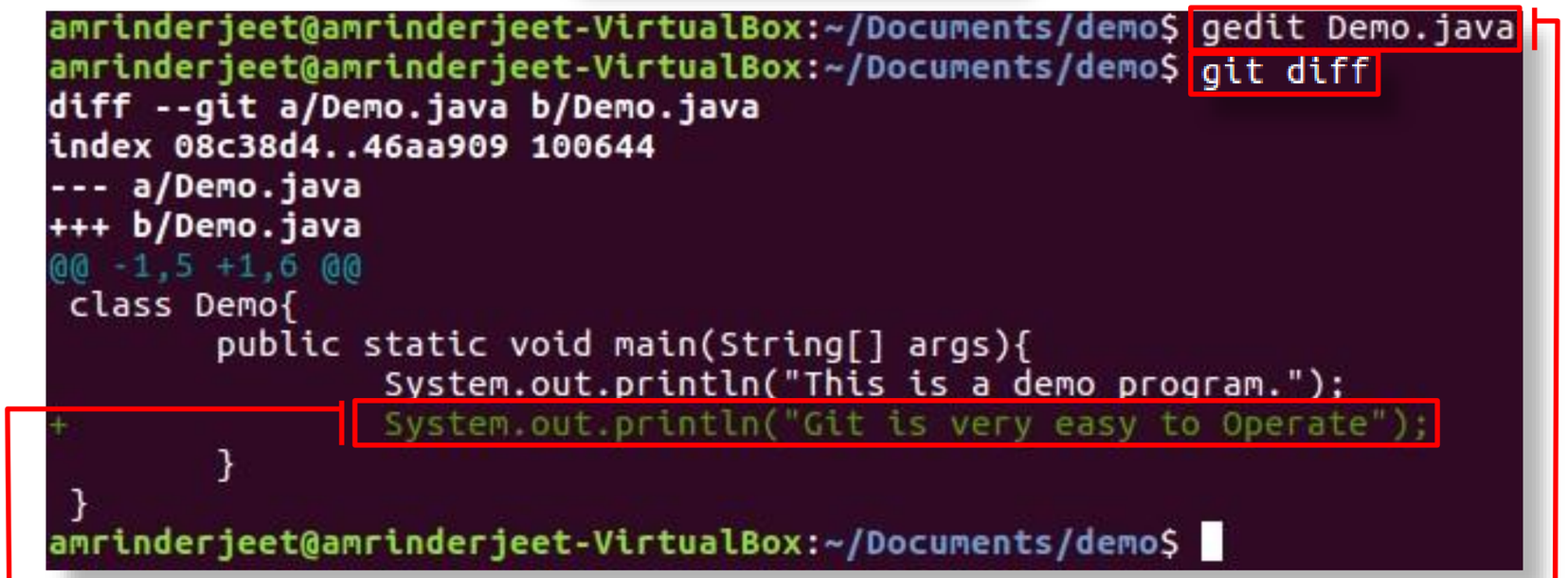
Commit message and description can be added here

```
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git commit
[master (root-commit) 1a79bb3] Added the first file to the repo
 1 file changed, 5 insertions(+)
 create mode 100644 Demo.java
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$
```

Tracking Changes

The git **diff** command displays all the changes made to the **tracked** files

```
Syntax: git diff
```



The changes made are displayed on the console

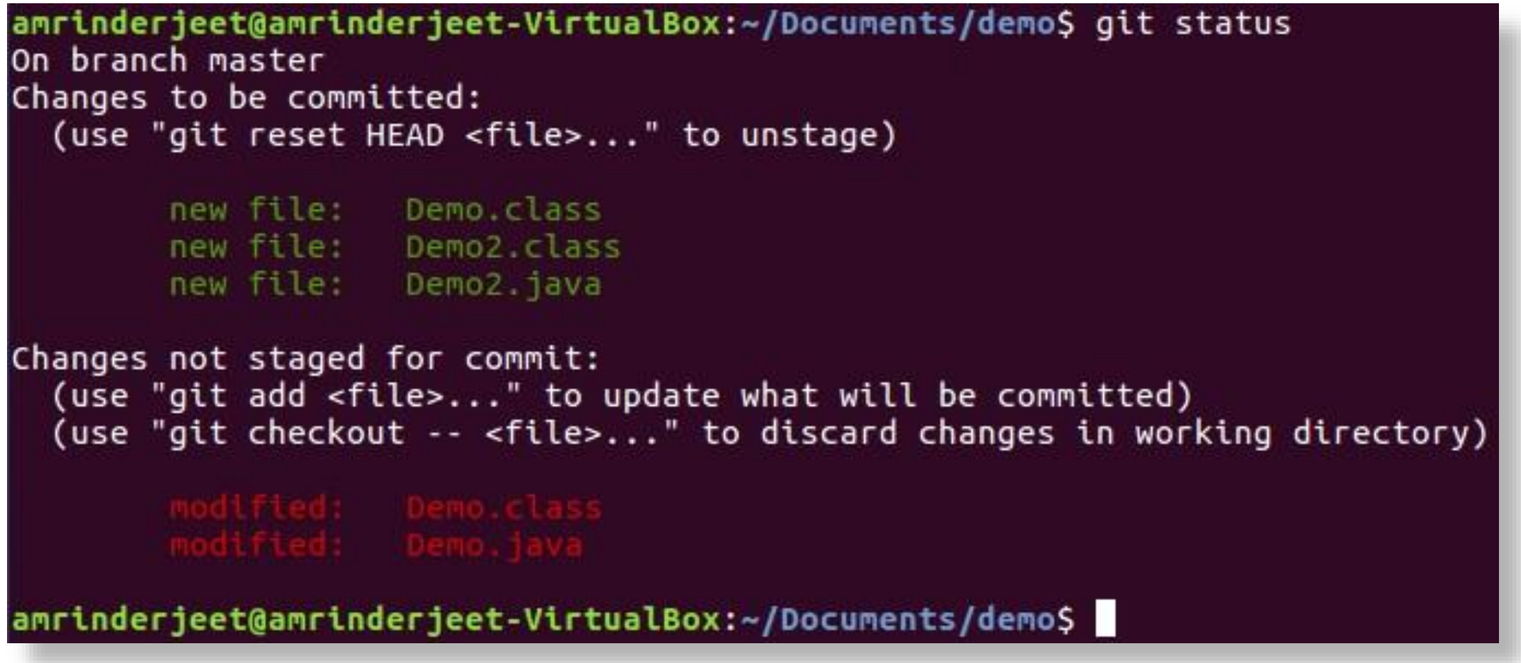
Changes made to the **Demo.java** file

Staging And Committing Multiple Files

- To **stage and commit** multiple files at once we use -a flag with the commit command

```
Syntax: git commit -a -m ‘message’
```

- Commit with -a flag automatically stages all the modified files and commits changes to the local repository

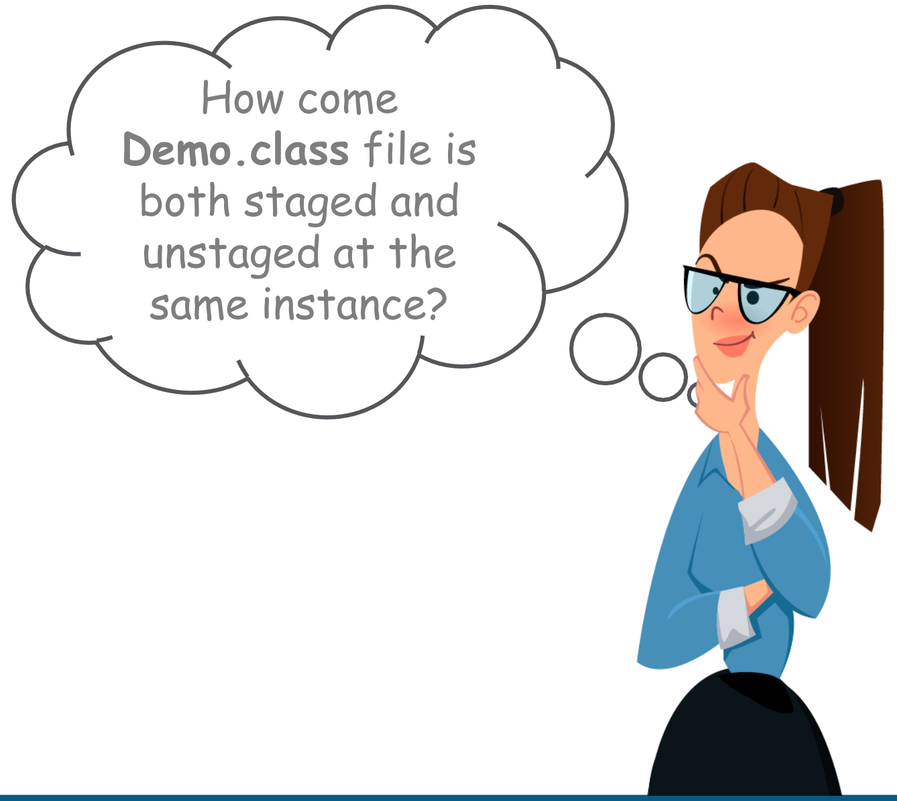


Staged and unstaged files before **commit -a**

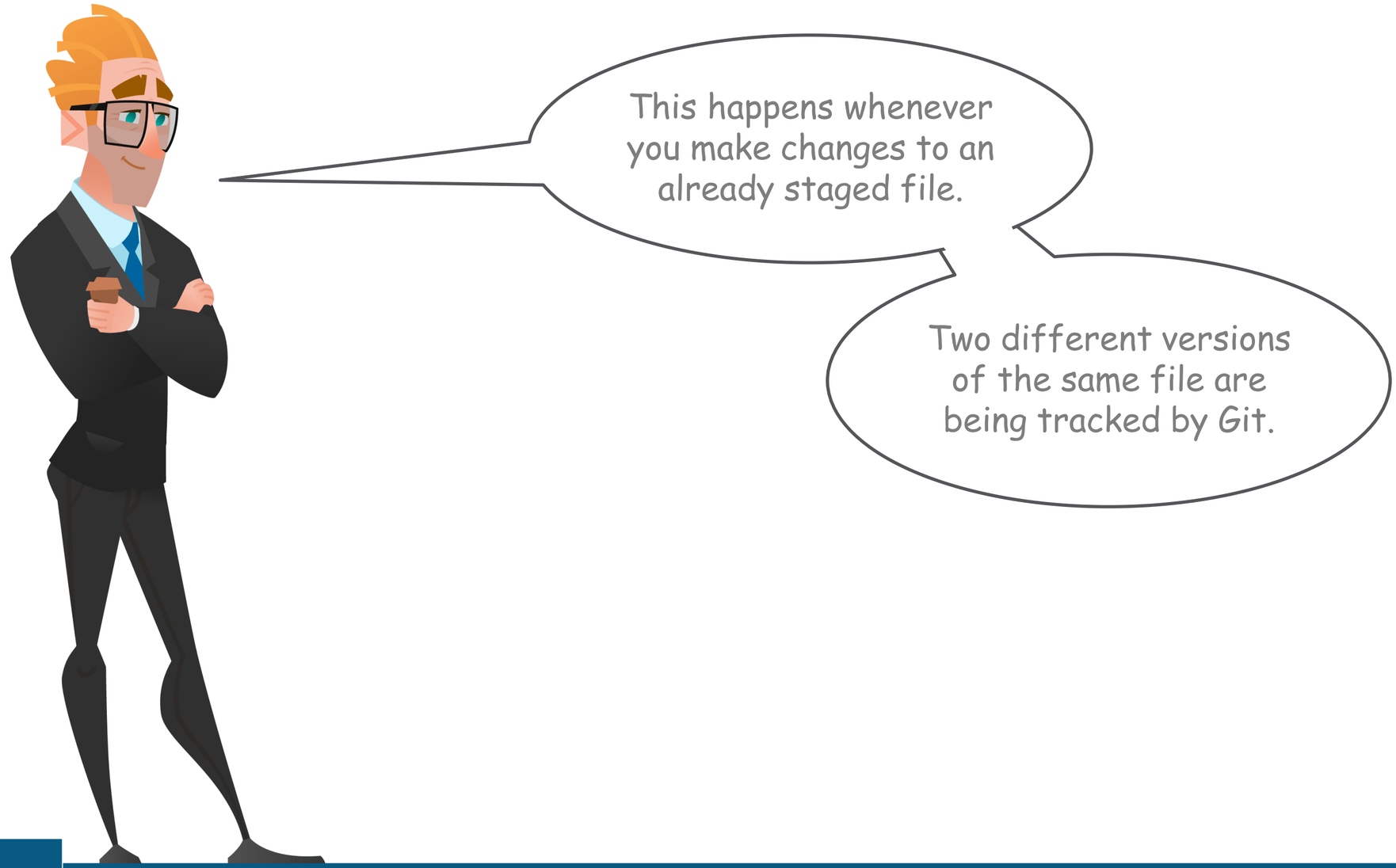
Staging And Committing Multiple Files

```
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git commit -a -m 'New files added and Demo.java modified'
[master dd756e6] New files added and Demo.java modified
 4 files changed, 6 insertions(+)
 create mode 100644 Demo.class
 create mode 100644 Demo2.class
 create mode 100644 Demo2.java
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$
```

```
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   Demo.class
        new file:   Demo2.class
        new file:   Demo2.java

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   Demo.class
        modified:   Demo.java

amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$
```



Why A File Is Both Staged And Unstaged?



Removing Files From Repository

- The **git rm** command deletes the file from git repository as well as users system

```
Syntax: git rm <filename>
```

```
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git rm Demo.class
```

- To remove the file from git repository but not from the system --**cached** option

```
Syntax: git rm --cached <filename>
```

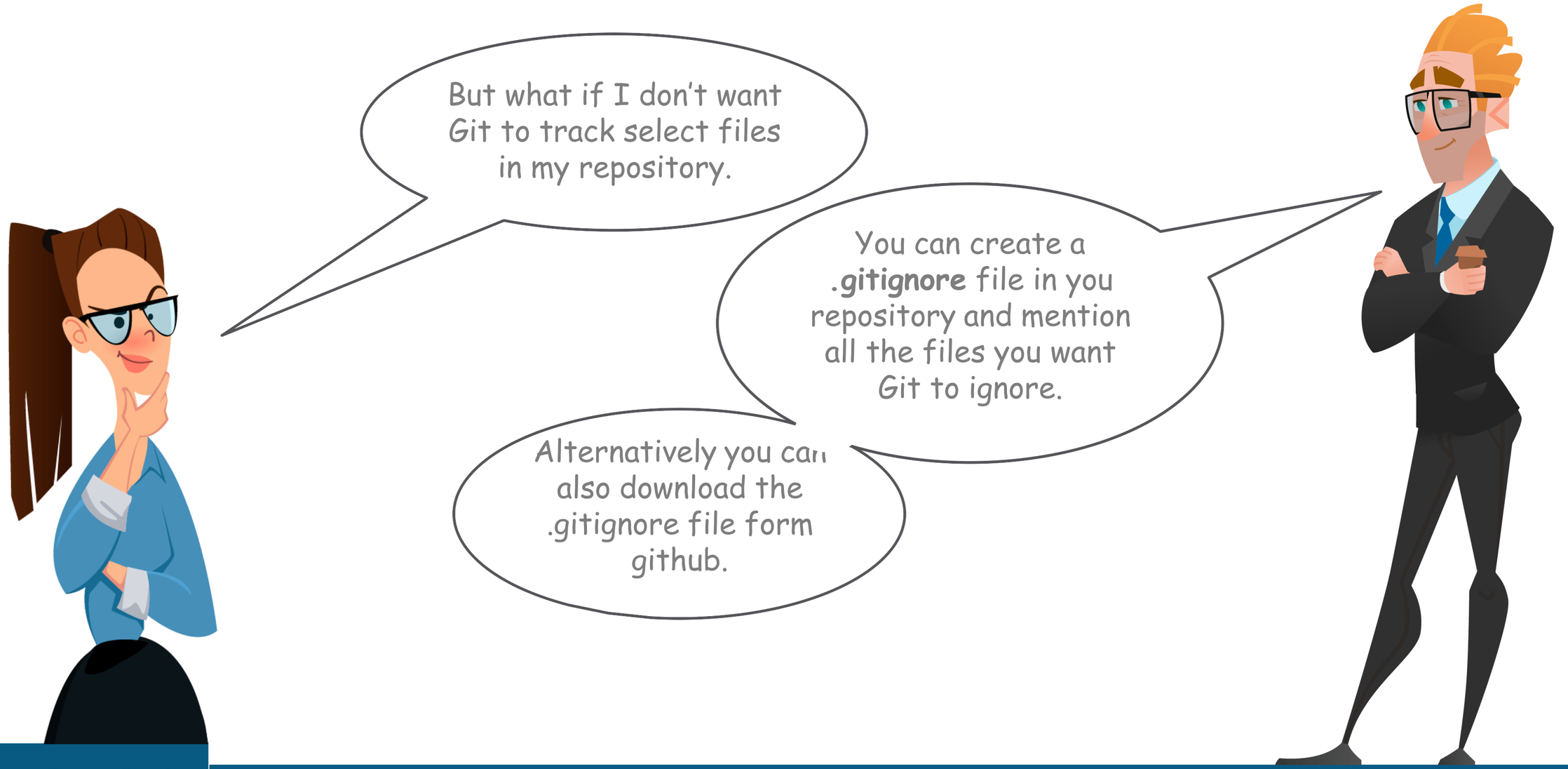
```
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git rm --cached Demo.class
rm 'Demo.class'
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$
```

- An error shows up if you try to delete a staged file

- You can force remove a staged file by using –**f** flag

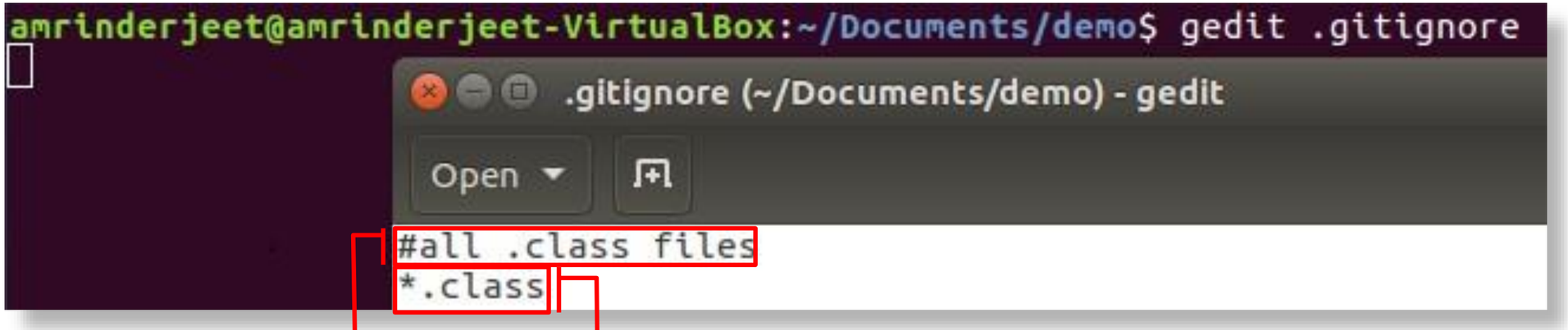
```
Syntax: git rm -f <filename>
```

Ignoring Files



Creating A Gitignore File

You can create a **.gitignore** file and add all the untracked files you want Git to ignore



Comments can be made using #

You can add file names or ignore a specific type of file as shown in the example

Adding A Gitignore File In The Repository

```
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        Demo.class
        Demo2.class

nothing added to commit but untracked files present (use "git add" to track)
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$
```

Before adding **.gitignore** file

```
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ gedit .gitignore
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gitignore

nothing added to commit but untracked files present (use "git add" to track)
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$
```

After adding **.gitignore** file

Commit: Git Log

The **git log** command shows all the commits so far on the current branch

```
Syntax: git log
```

```
amrinderjeet@amrinderjeet-VirtualBox:~$ cd Documents/demo
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git log
commit d07dc4d9c79b78ed72fc0ac782da394f6e21d495
Author: amrjeet <amrinderjeet.singh@edureka.co>
Date:   Tue Jan 30 15:30:56 2018 +0530

    removed class etension files

commit dd756e61d7b30b9aebb5c76d808eda49874e6bff
Author: amrjeet <amrinderjeet.singh@edureka.co>
Date:   Tue Jan 30 14:40:02 2018 +0530

    New files added and Demo.java modified

commit 1a79bb3a625707116dc8fddd7f37677348f7bf2d
Author: amrjeet <amrinderjeet.singh@edureka.co>
Date:   Tue Jan 30 12:44:27 2018 +0530

    Added the first file to the repo
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$
```

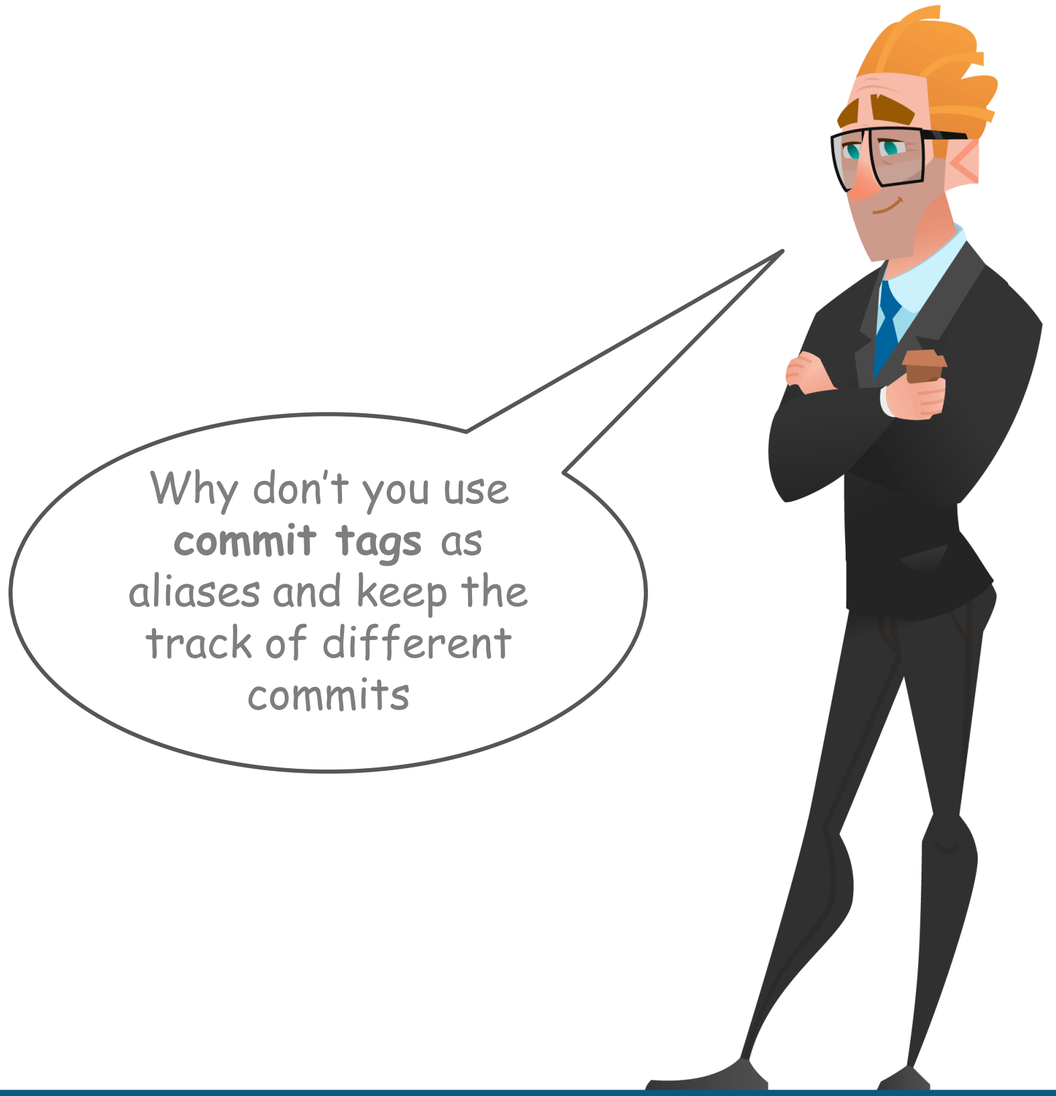
Commit: Amend

Don't worry. You can change the last commit message using **--amend** option

```
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git commit --amend
[master 11cf392] removed .class extension files
 Date: Tue Jan 30 15:30:56 2018 +0530
 2 files changed, 0 insertions(+), 0 deletions(-)
 delete mode 100644 Demo.class
 delete mode 100644 Demo2.class
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$
```

edureka!

Commit Tags



Commit: Git Tag

- Commit tags provide an alias for commitID

```
Syntax: git tag --a <annotation> --m <message>
```

- You can also view all the tags you have created

```
Syntax: git tag
```

```
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git tag -a v1.3 -m 'Version 1.3'
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git tag
v1.3
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$
```

Commit Tags

- Adding a tag to one of the previous commits

```
Syntax: git tag --a <annotation> <commit id> --m <message>
```

- Commit id can be obtained from git logs

- All these tags can be viewed in git

```
Syntax: git show <tag-name>
```

```
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git tag -a v0.3 11cf392d -m 'Added tag to a previous commit'
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git show v0.3
tag v0.3
Tagger: amrjeet <amrinderjeet.singh@edureka.co>
Date:   Wed Jan 31 11:51:32 2018 +0530

Added tag to a previous commit

commit 11cf392dc4307d8446fdc4419e8d9c3674549385
Author: amrjeet <amrinderjeet.singh@edureka.co>
Date:   Tue Jan 30 15:30:56 2018 +0530

    removed .class extension files

diff --git a/Demo.class b/Demo.class
deleted file mode 100644
index fa53ecb..0000000
Binary files a/Demo.class and /dev/null differ
diff --git a/Demo2.class b/Demo2.class
deleted file mode 100644
index 81c13a3..0000000
Binary files a/Demo2.class and /dev/null differ
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$
```

Thank You

For more information please visit our website
www.edureka.co