GIT TUTORIAL

simplilearn

What's in it for you?

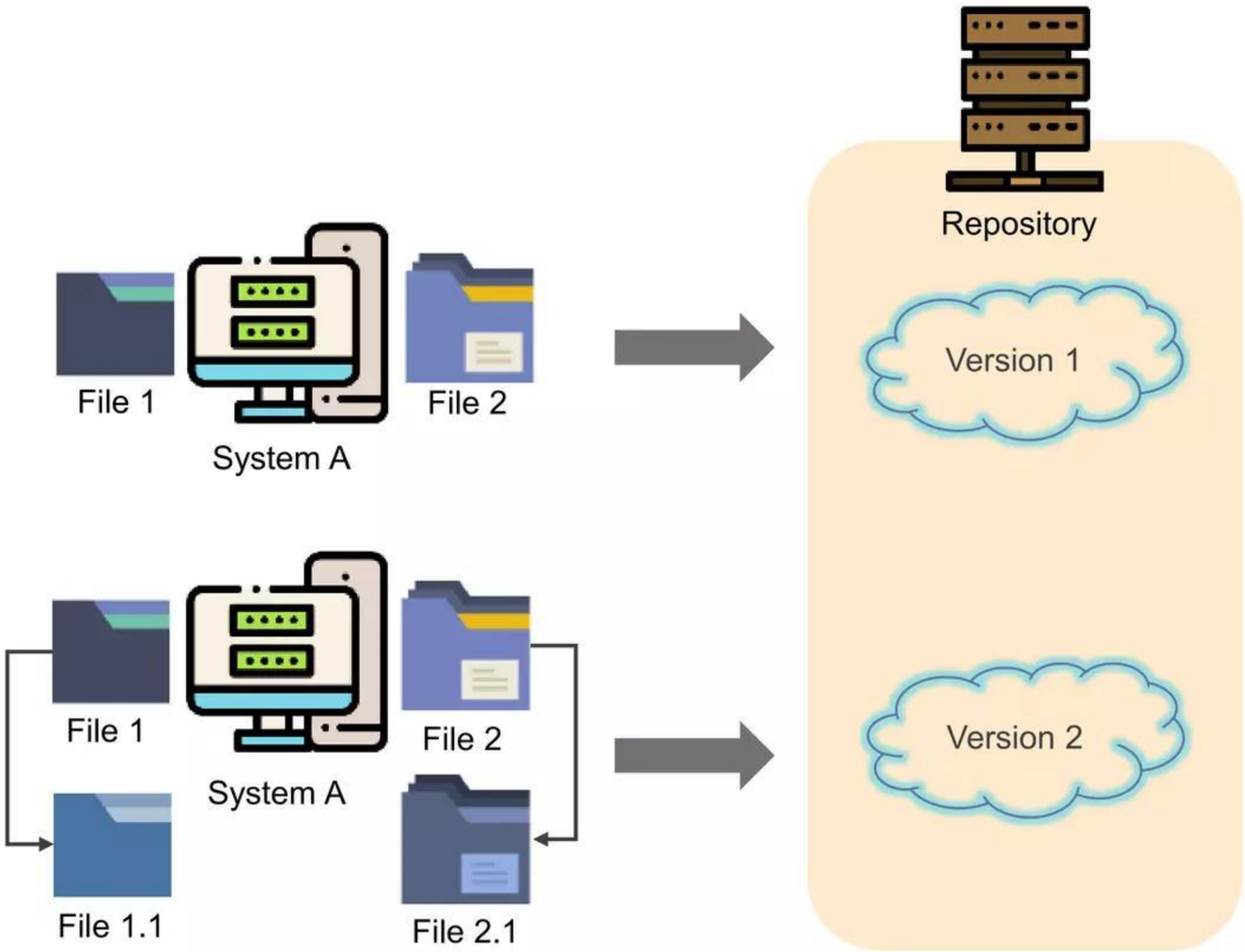
- Version Control System
- Distributed Version Control System
- What is Git?
- Git vs GitHub
- Git Architecture

- Fork and Clone
- Collaborators
- Branch, Merge and Rebase
- Commands in Git
- Git Demo

simplilearn

Version Control System

Version Control System (VCS)



All the files in System A are stored as Version 1 in the remote repository

Now we make some changes to the files in System A

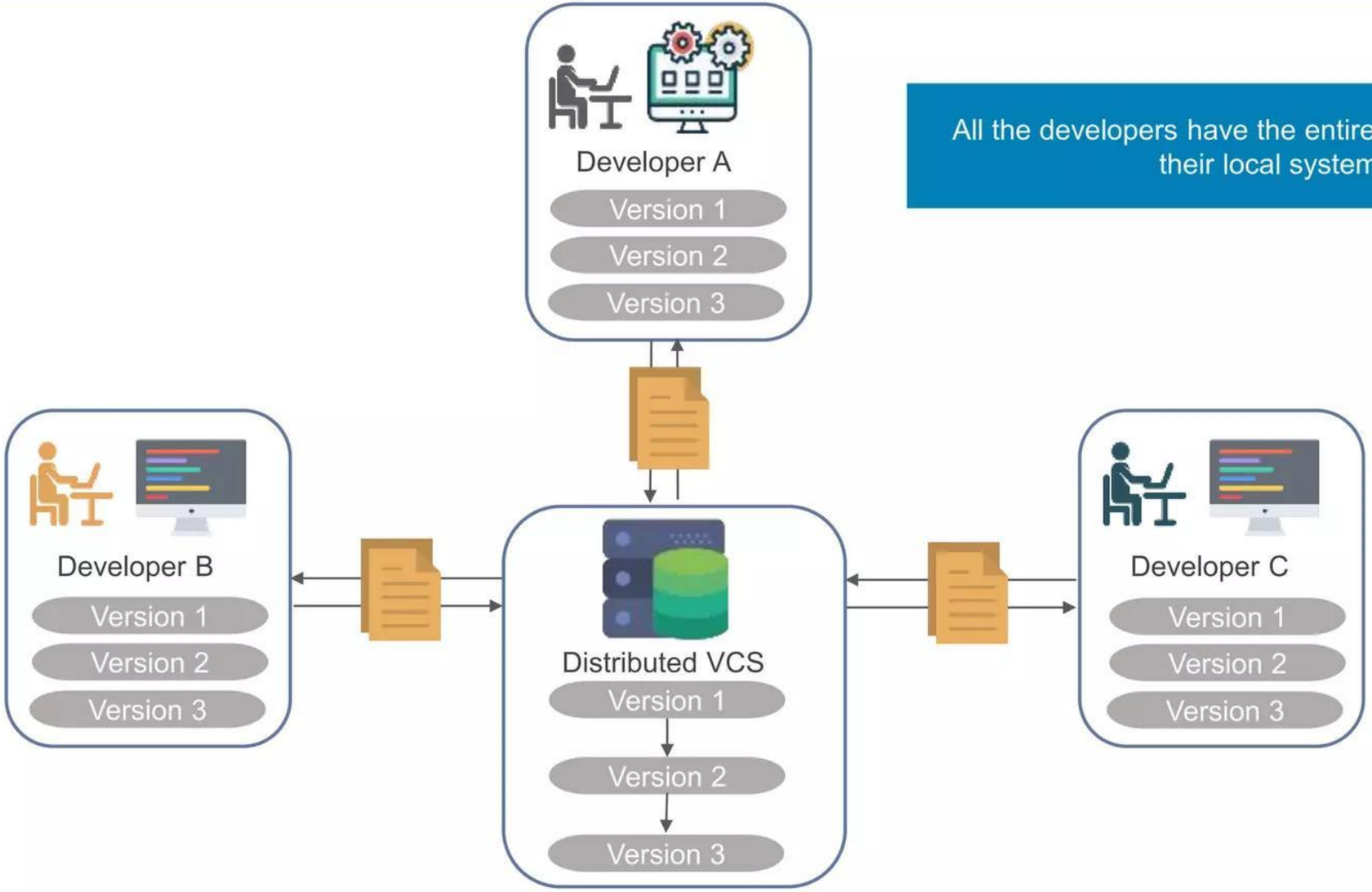
File 1 ------> saved as File 1.1
File 2 ------> saved as File 2.1

The new files are stored as Version 2 in the repository

VCS allows you to store multiple versions of a system file in the remote repository

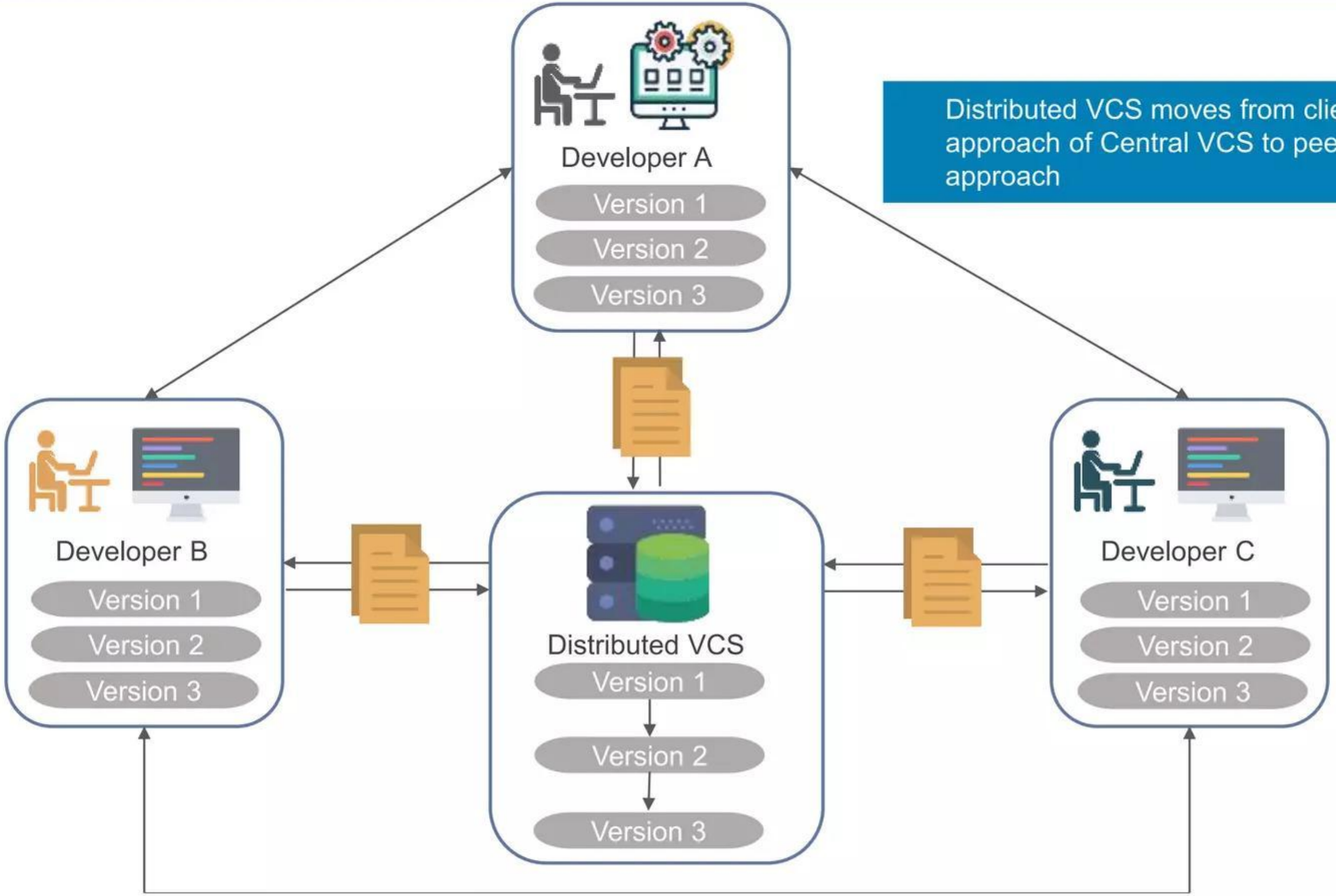
Distributed Version Control System

Distributed Version Control System



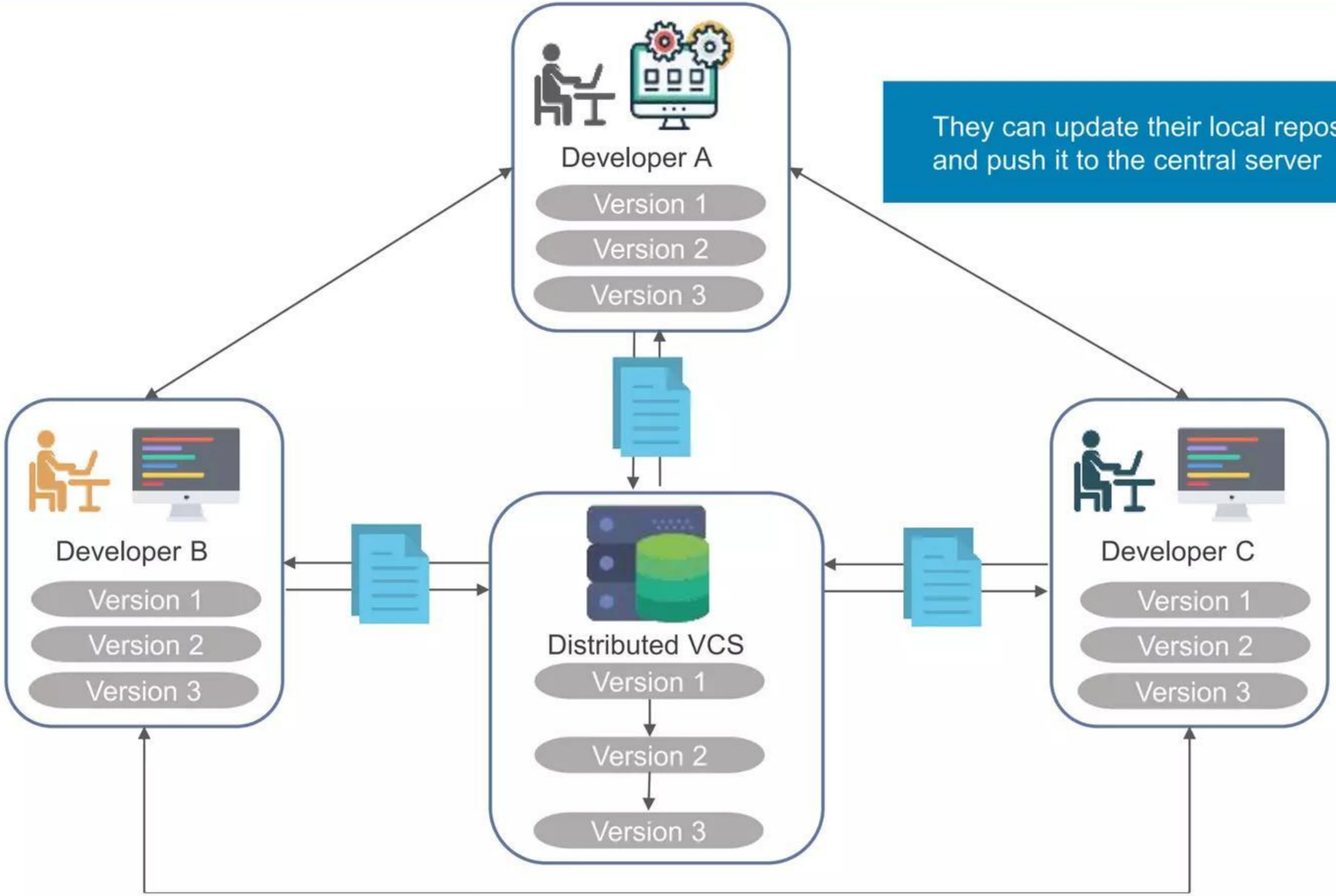
All the developers have the entire copy of code on their local system

Distributed Version Control System

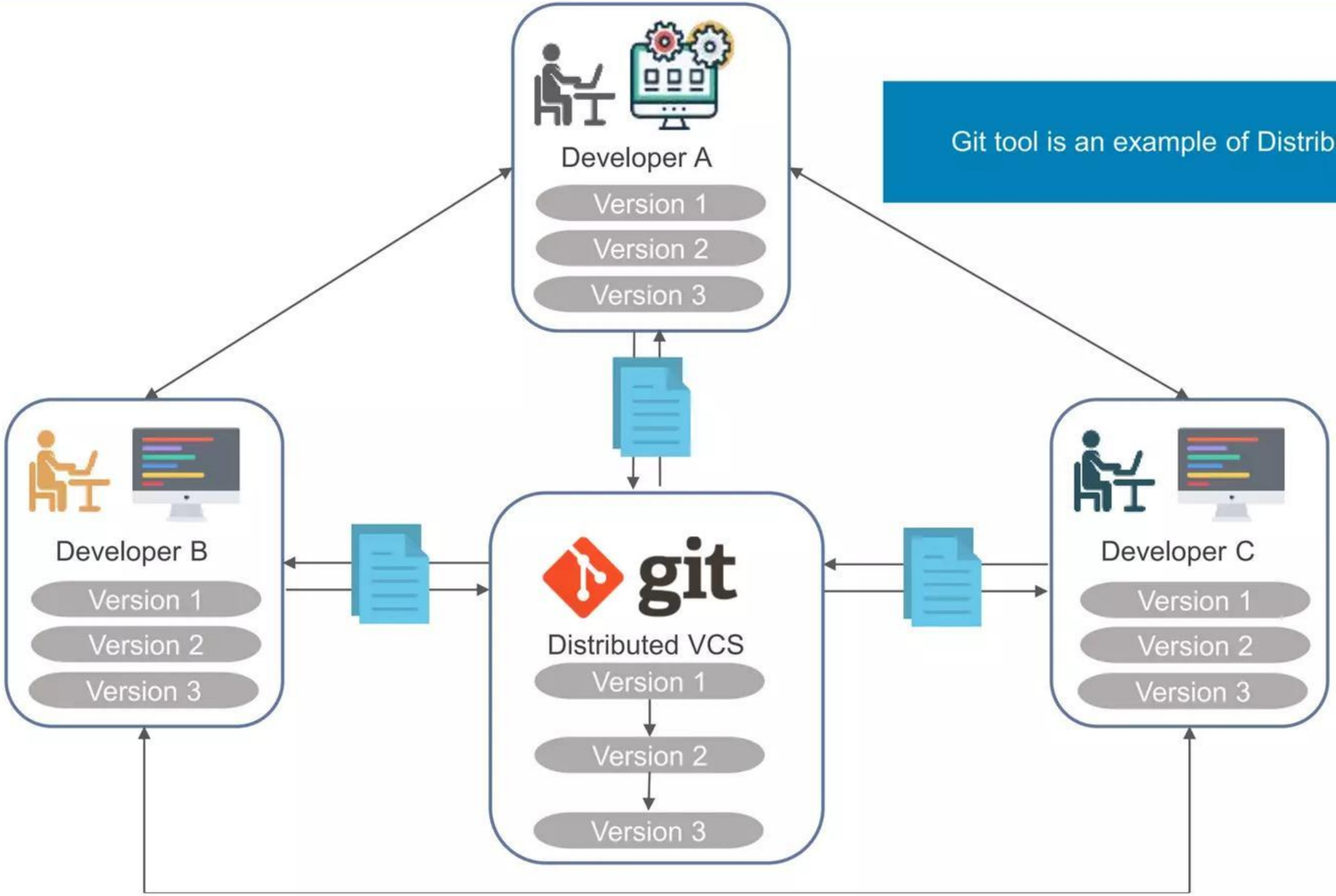


Distributed VCS moves from client-server approach of Central VCS to peer-to-peer approach

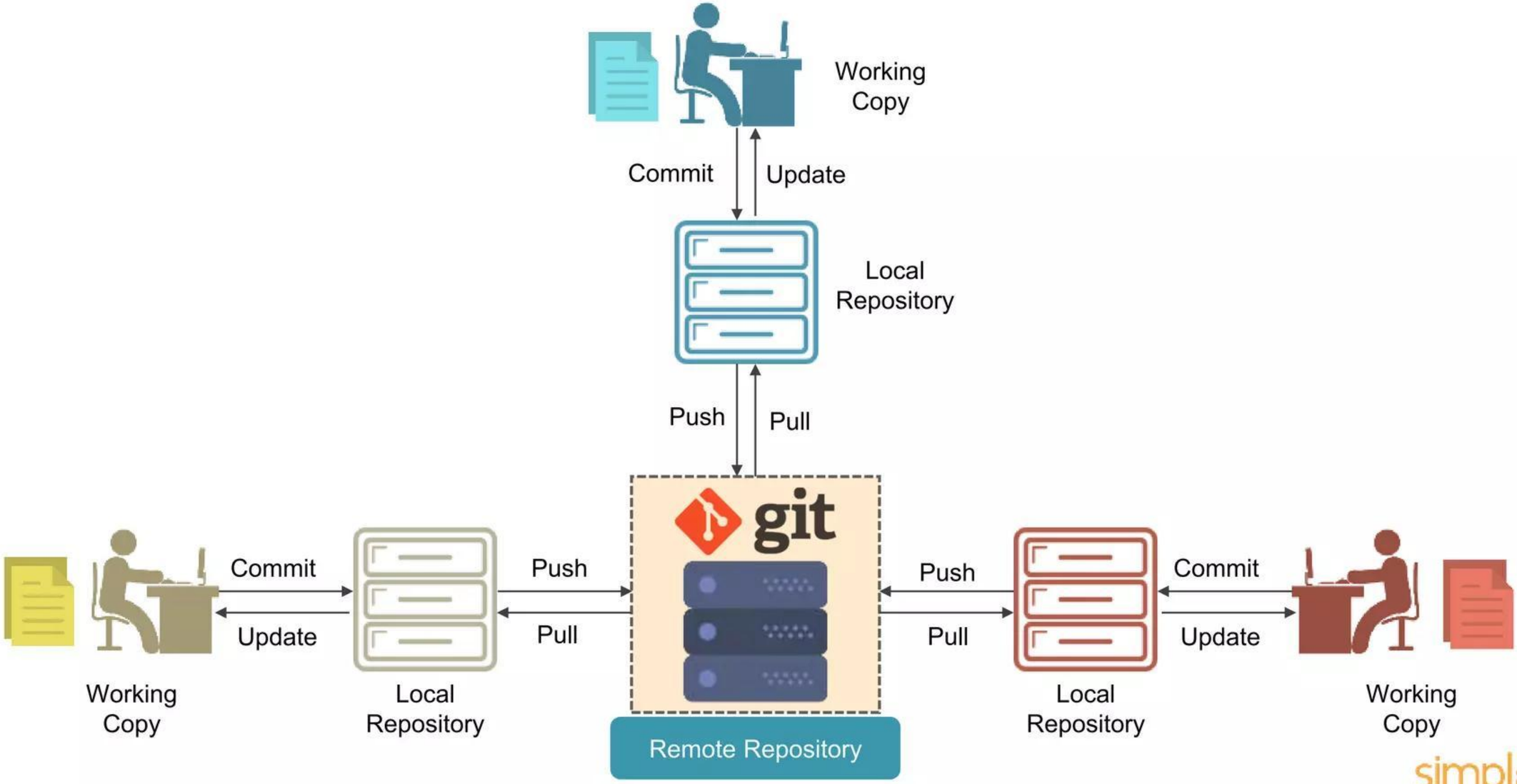
Distributed Version Control System



Distributed Version Control System



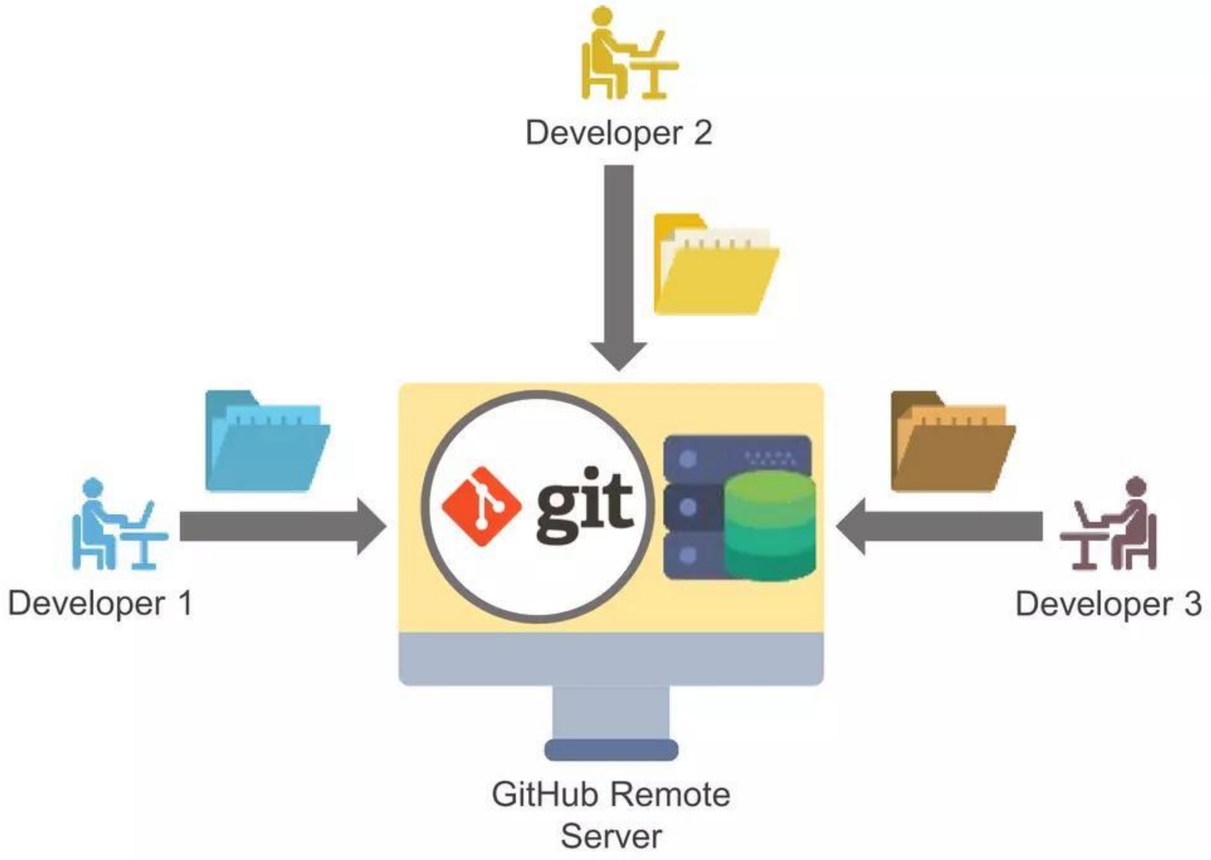
Distributed Version Control System



What is Git?

What is Git?

Git is a distributed version control tool used for source code management



- Git is used to track changes in the source code
- Allows multiple developers to work together
- Supports non-linear development because of thousands of parallel branches
- Has the ability to handle large projects efficiently

Git vs GitHub

Git vs GitHub

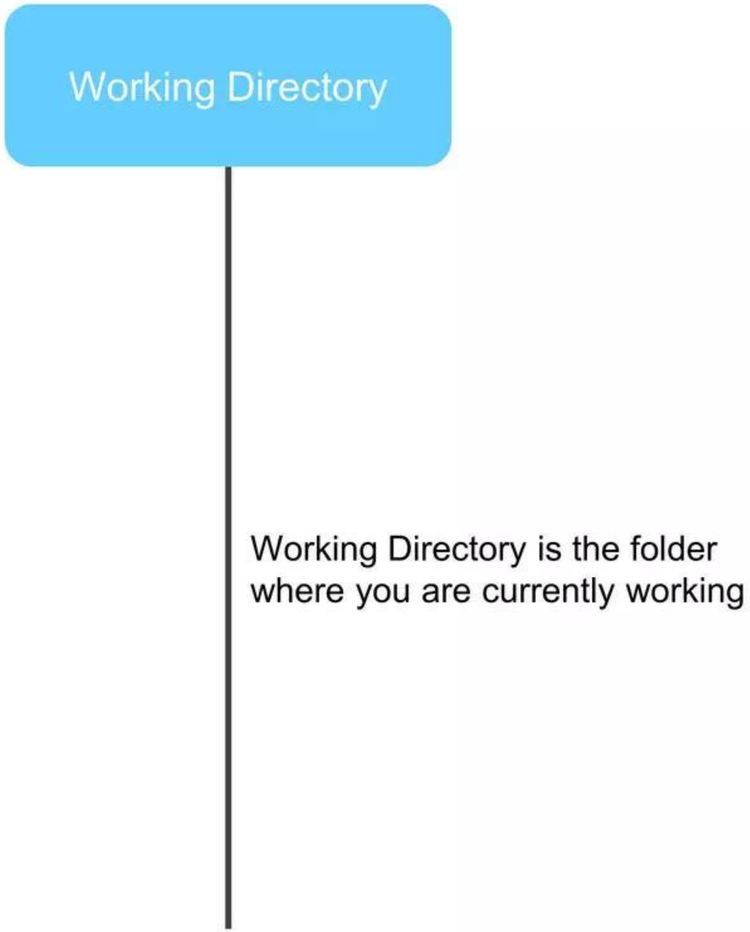
	git	GitHub	
1	Git is a software tool	GitHub is a service	1
2	It is installed on the local system	It is hosted on the web	2
3	It is used to manage different versions of the source code	It is used to have a copy of the local repository code	3
4	It provides a command line to interact with the files	It provides a graphical interface to store the files	4

simplilearn

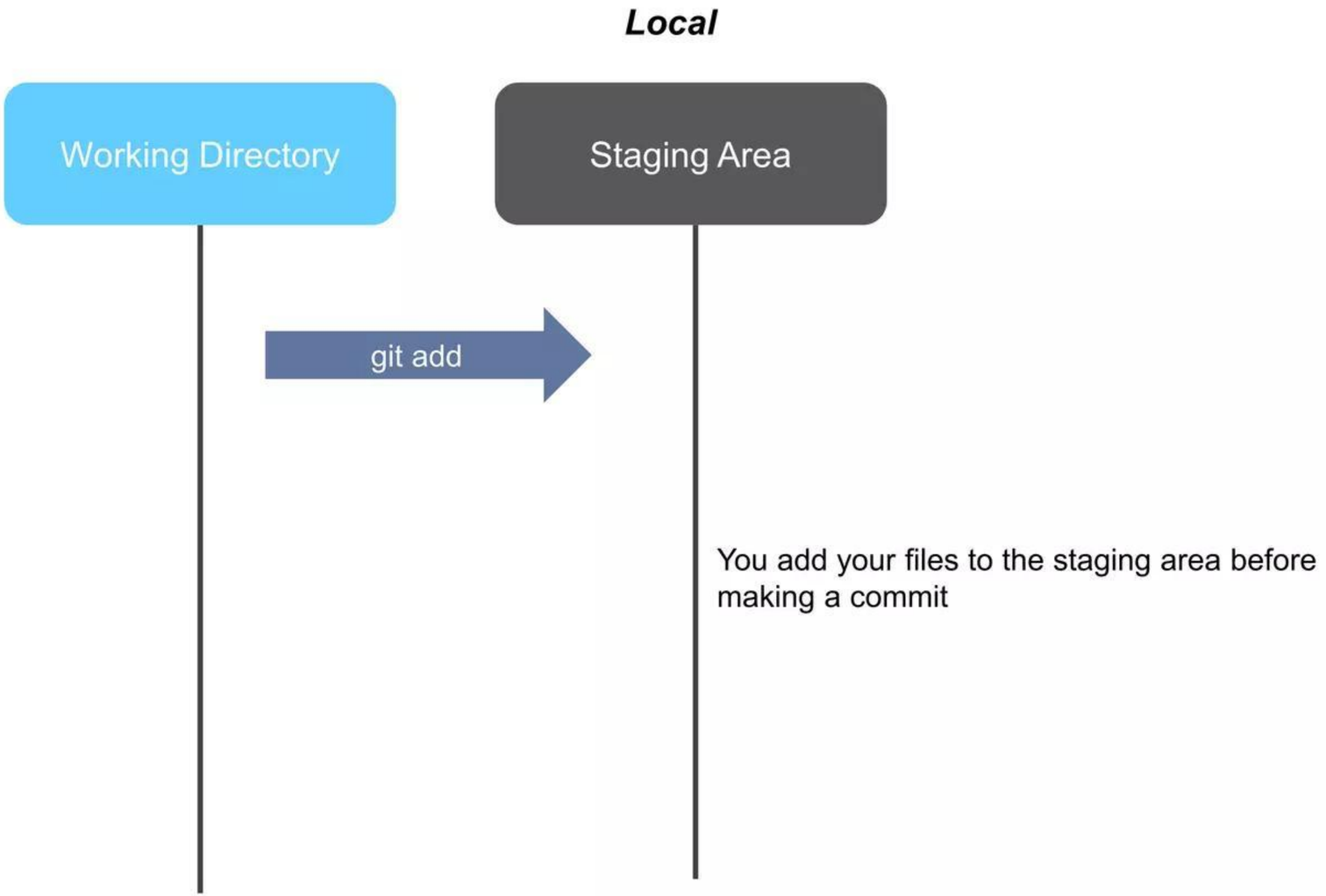
Git Architecture

Git Architecture

Local

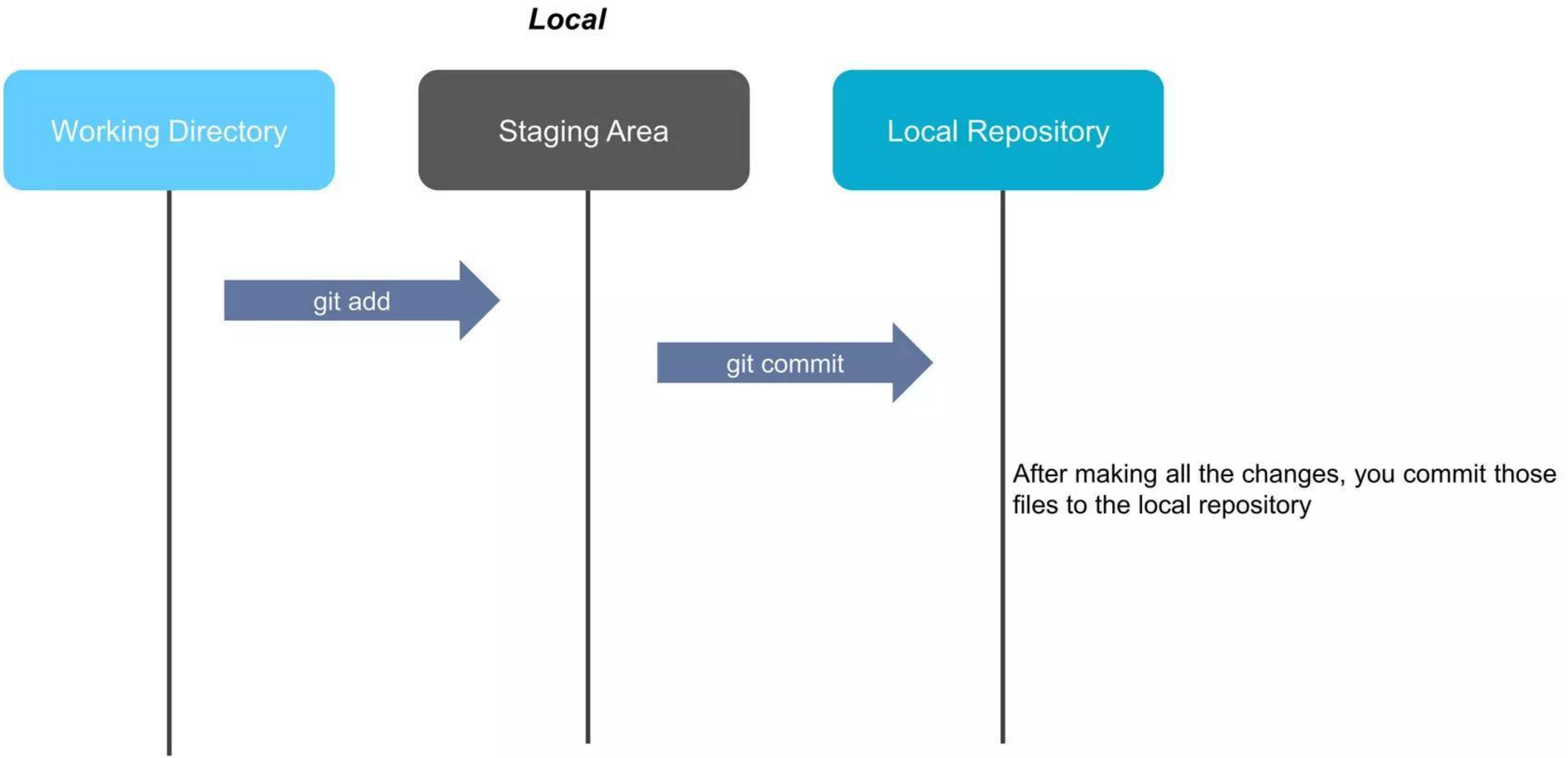


Git Architecture

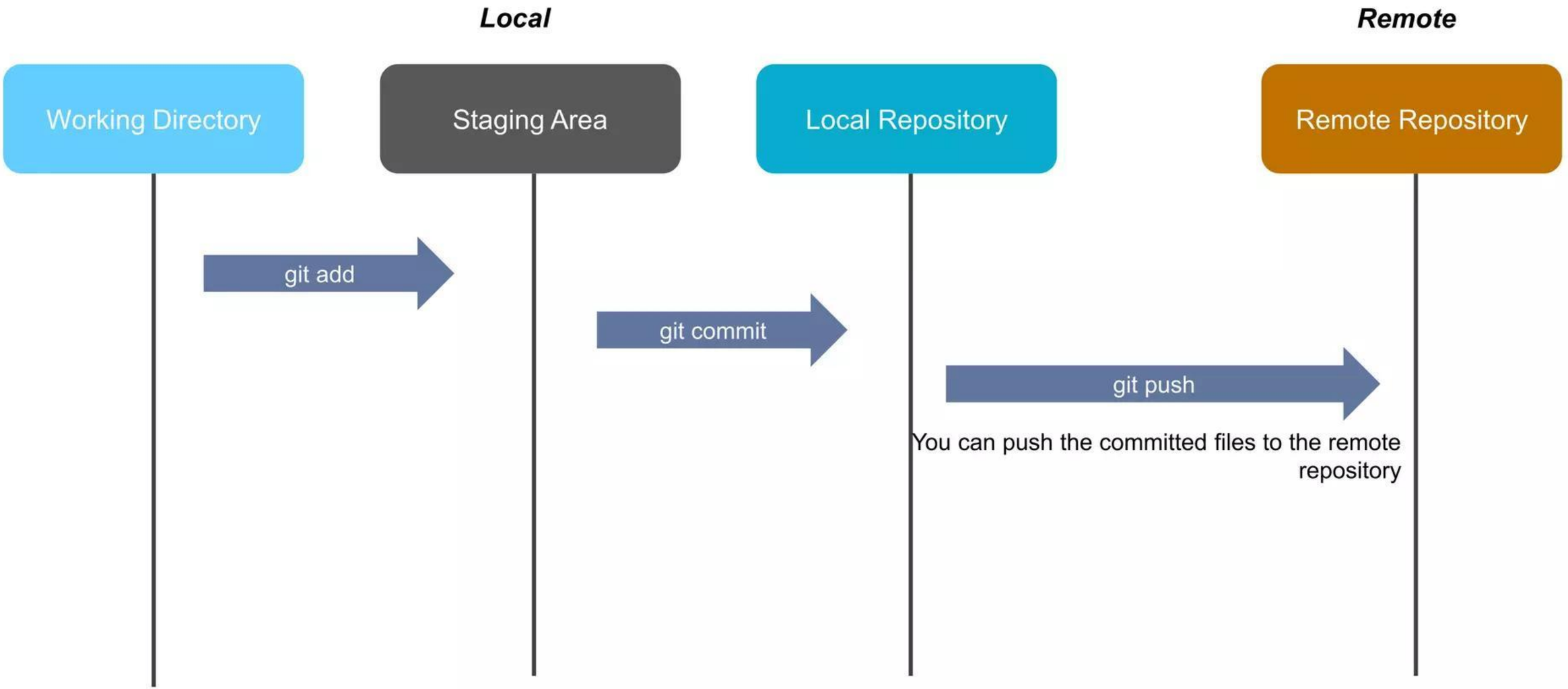


You add your files to the staging area before making a commit

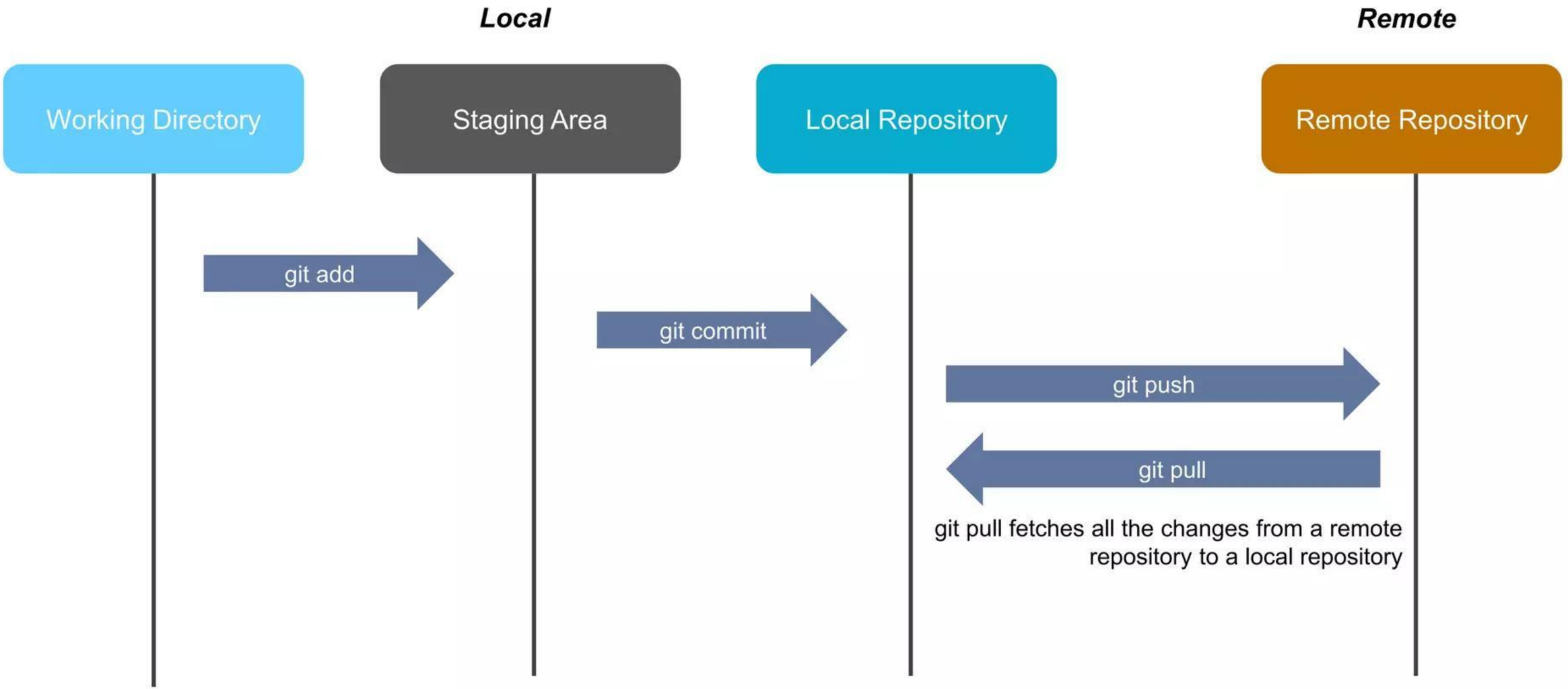
Git Architecture



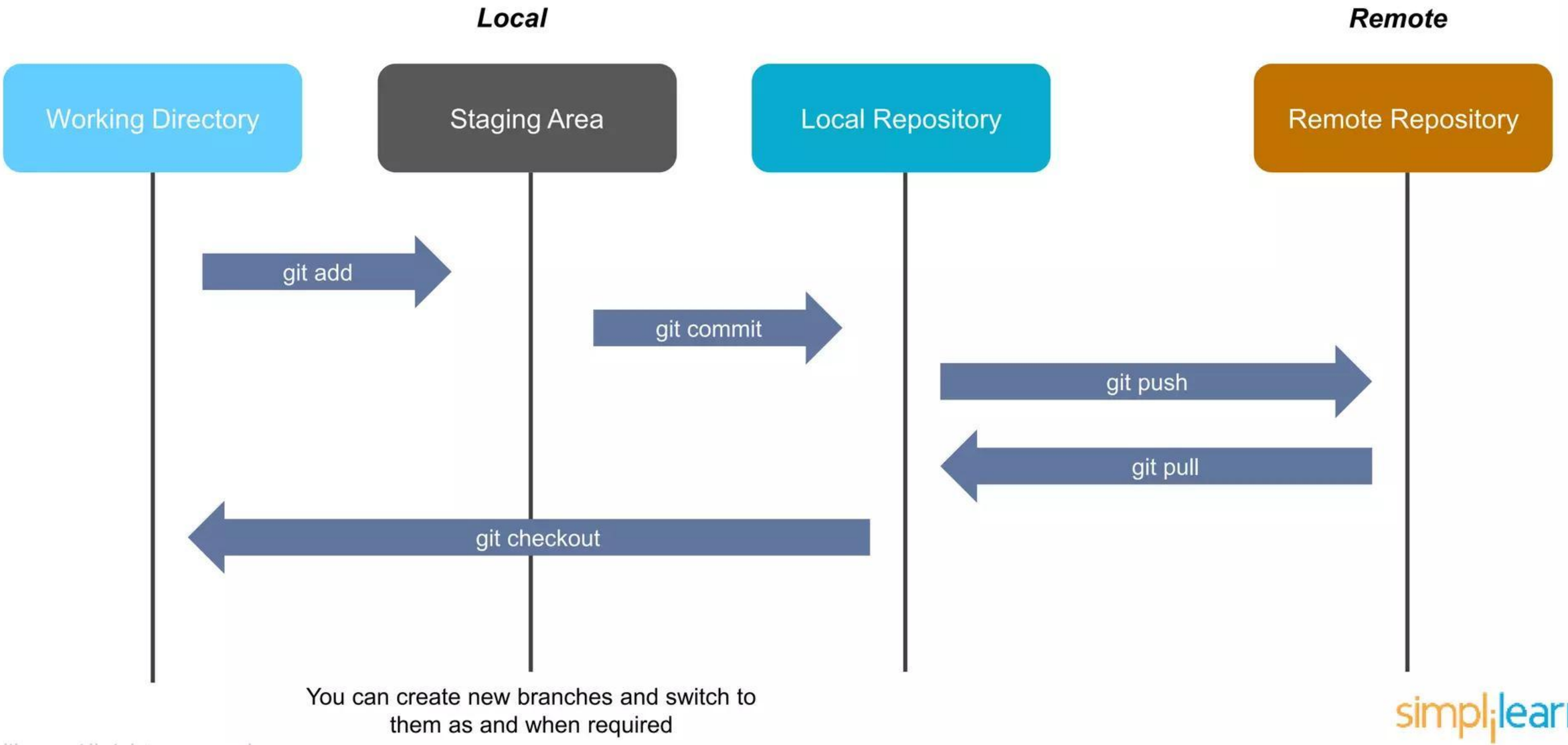
Git Architecture



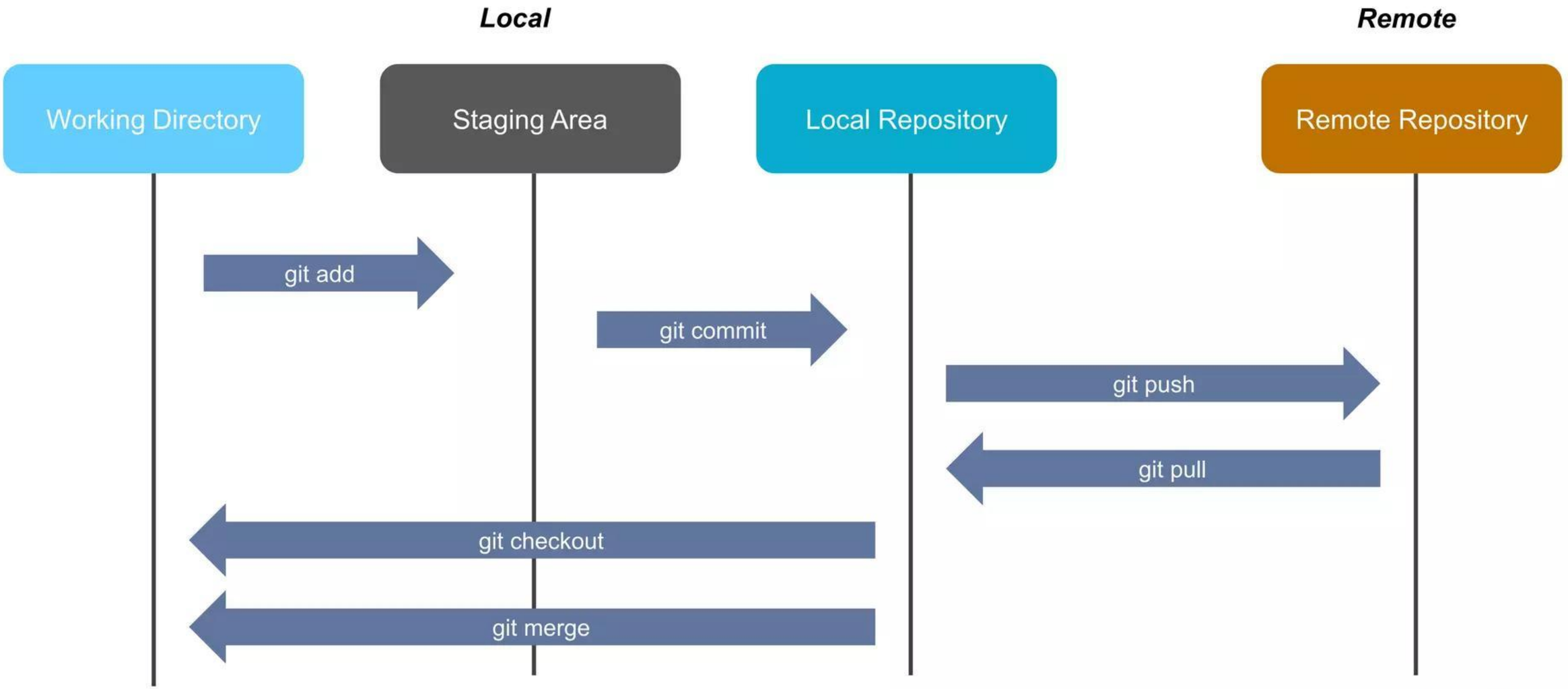
Git Architecture



Git Architecture



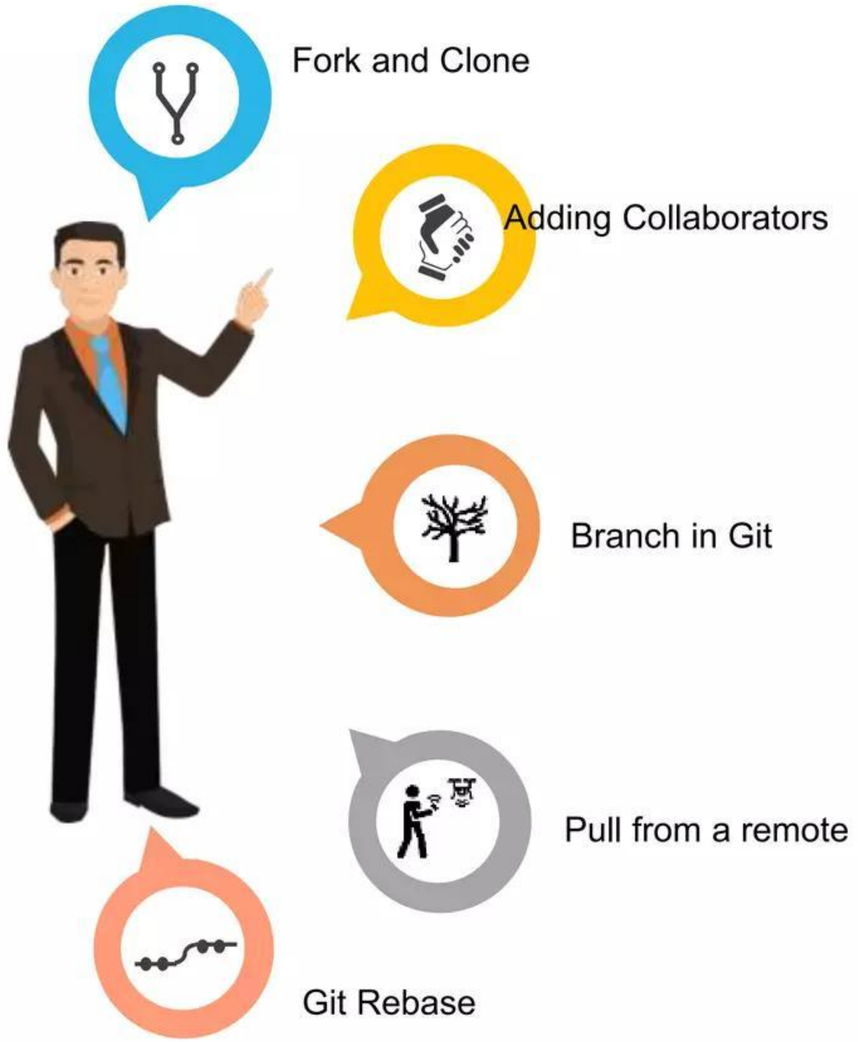
Git Architecture



After you are done with the changes, you can merge the new branches to the master branch

Git Concepts

Git Concepts

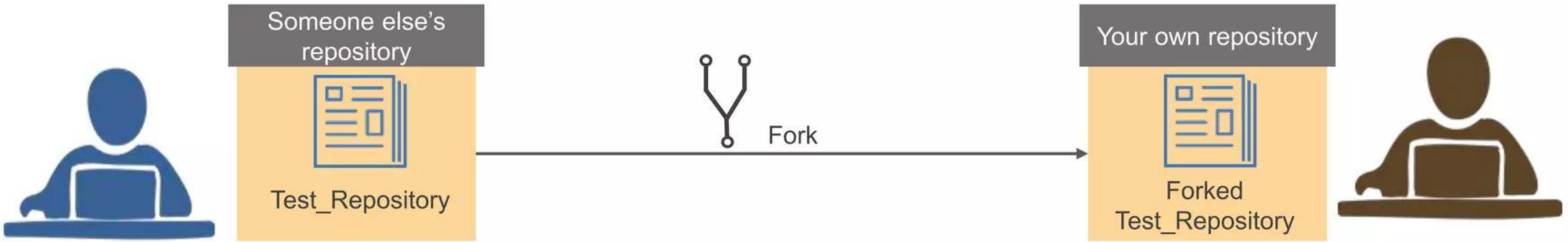


Fork and Clone

Fork and Clone

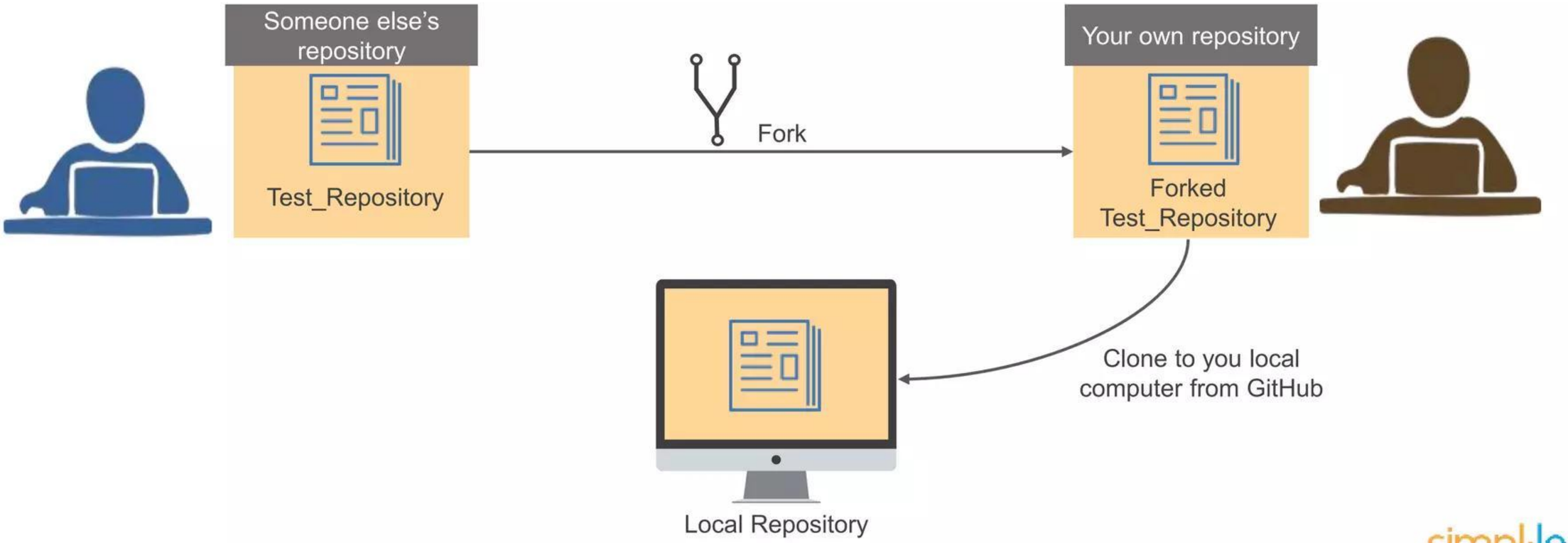
Fork and Clone

Git allows you to fork an open source repository. When you fork a repository, you create a copy of it on your GitHub account.



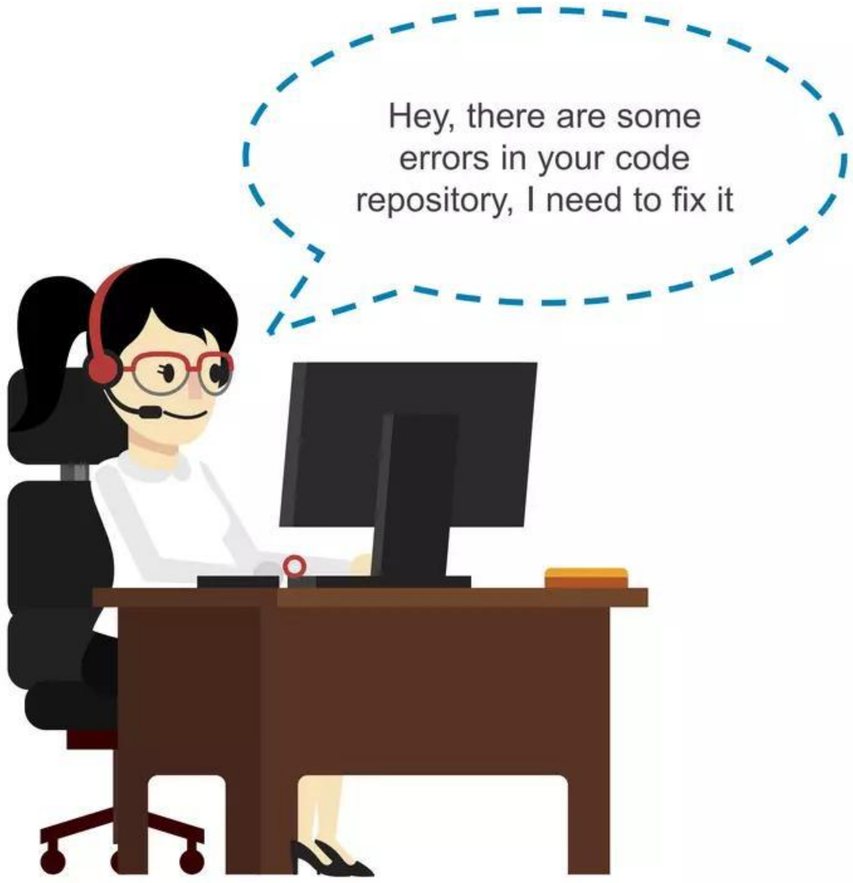
Fork and Clone

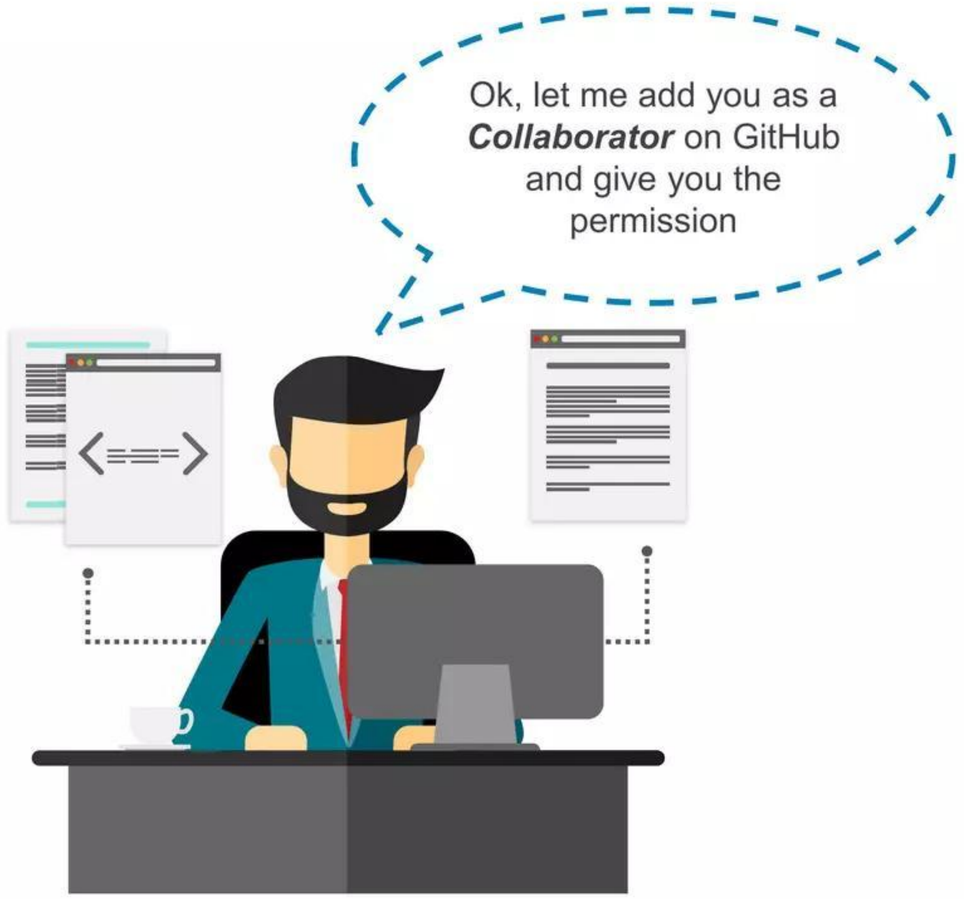
After you fork a repository, you can clone it and have a copy of it on your local system.



Collaborators

Collaborators

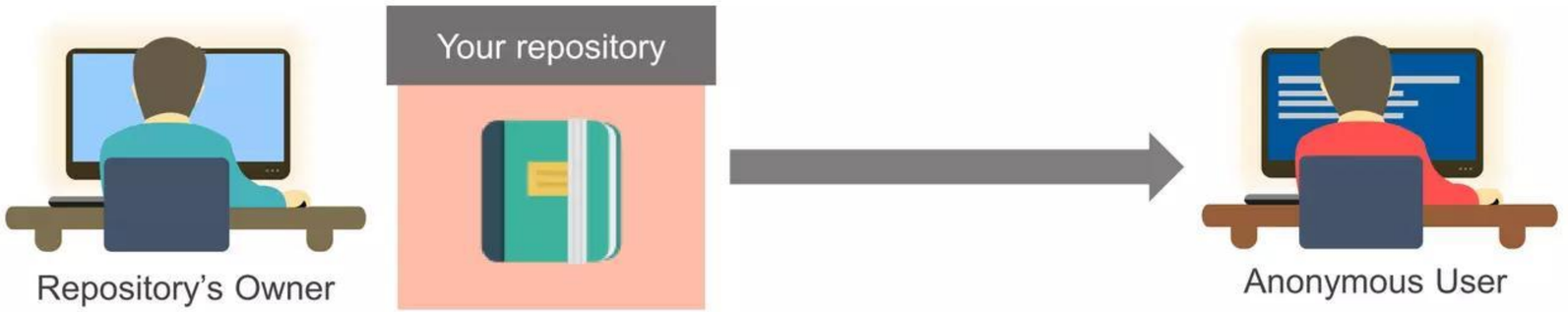




Collaborators

GitHub allows you to work with users from all over the world at any given time

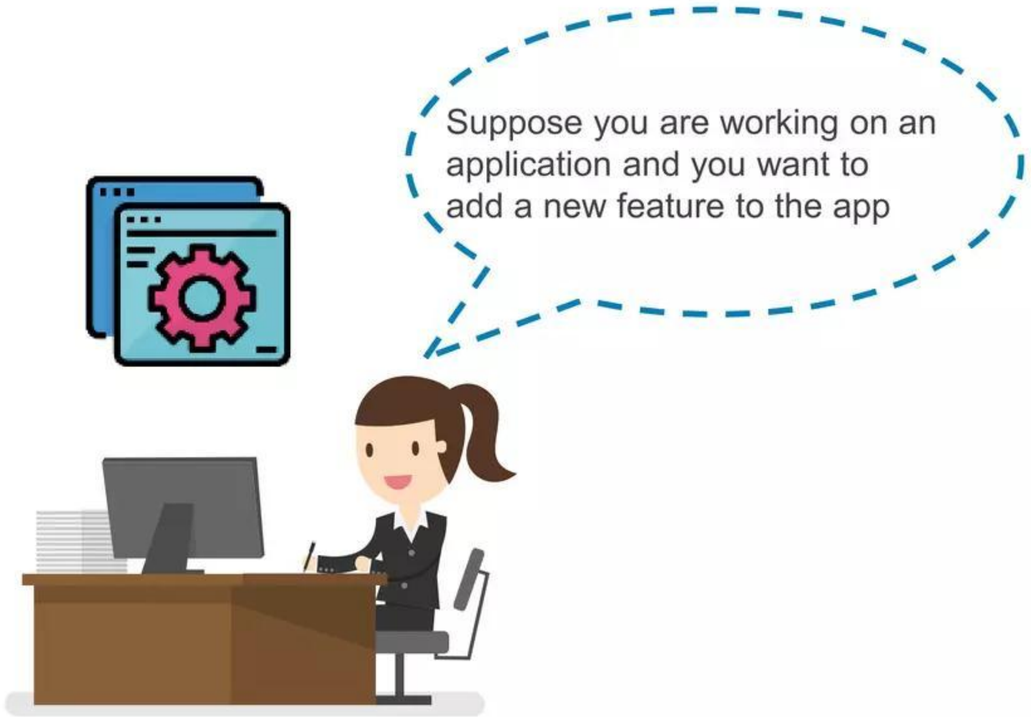
Collaborators are GitHub users who are given permission to edit a repository owned by someone else



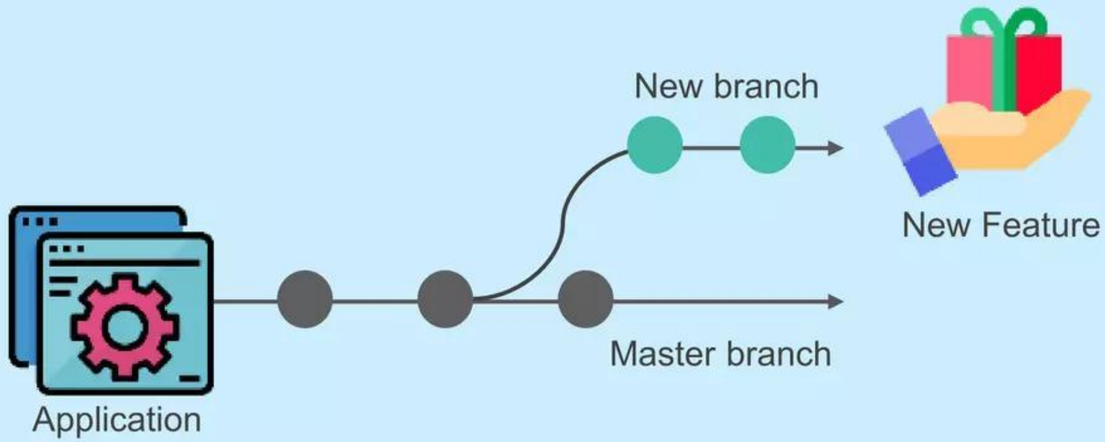
- ❑ Access all the files
- ❑ View the files
- ❑ Make changes to the files
- ❑ Commit the files

Branch in Git

Branch in Git



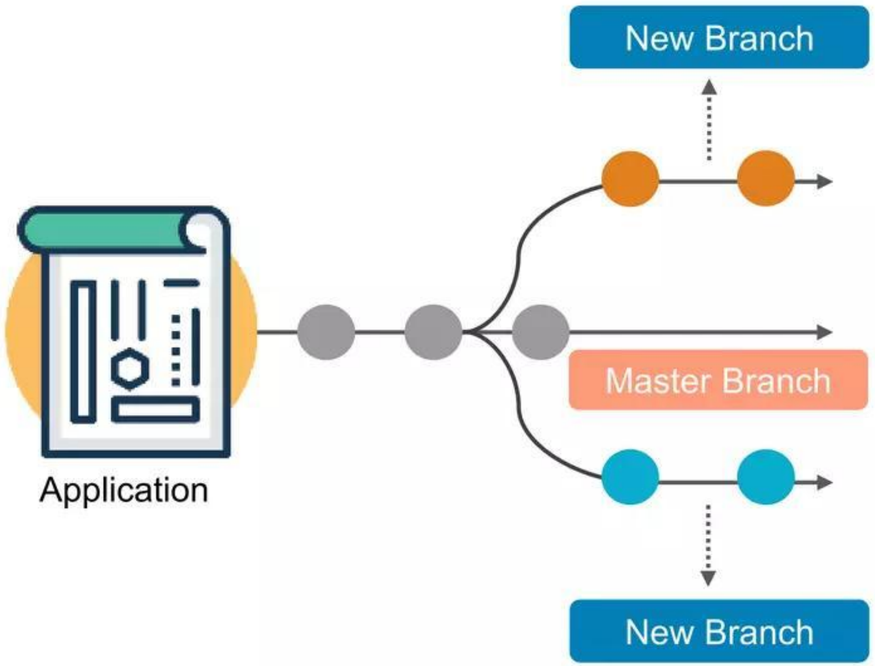
You can create a new branch and build the new feature on that branch



- By default, you always work on the master branch
- The circles on the branch represent various commits made on the branch

Branch in Git

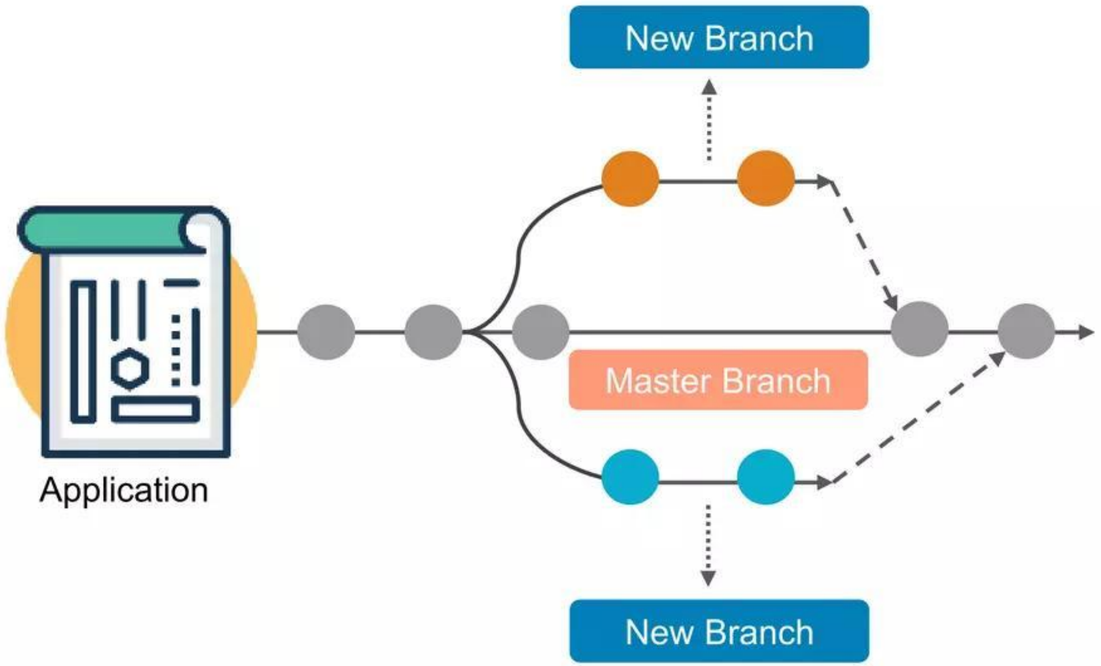
Branch in Git is used to keep your changes until they are ready



- The diagram shows there are 2 new branches
- You can develop the features you want separately

Branch in Git

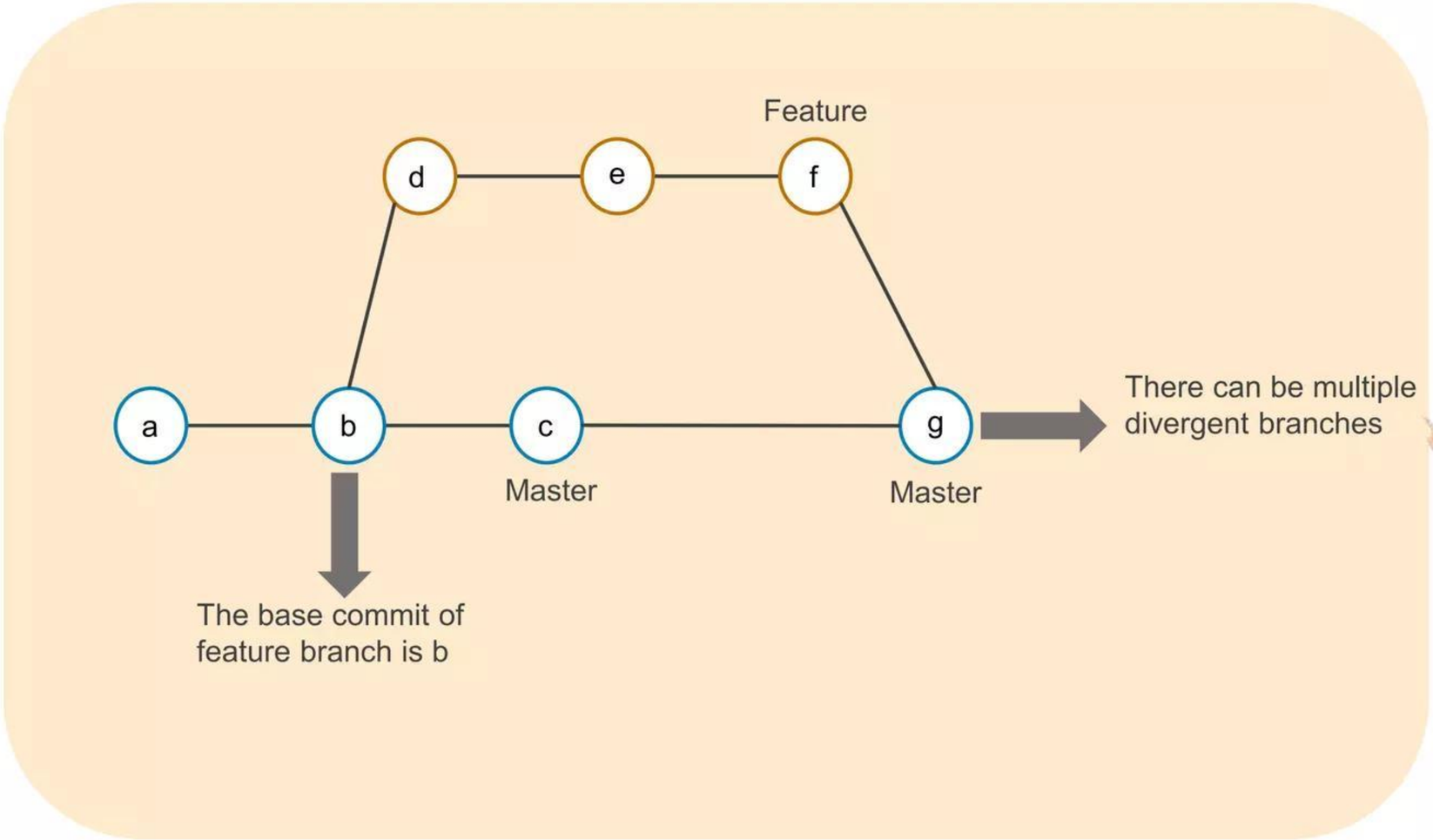
Branch in Git is used to keep your changes until they are ready



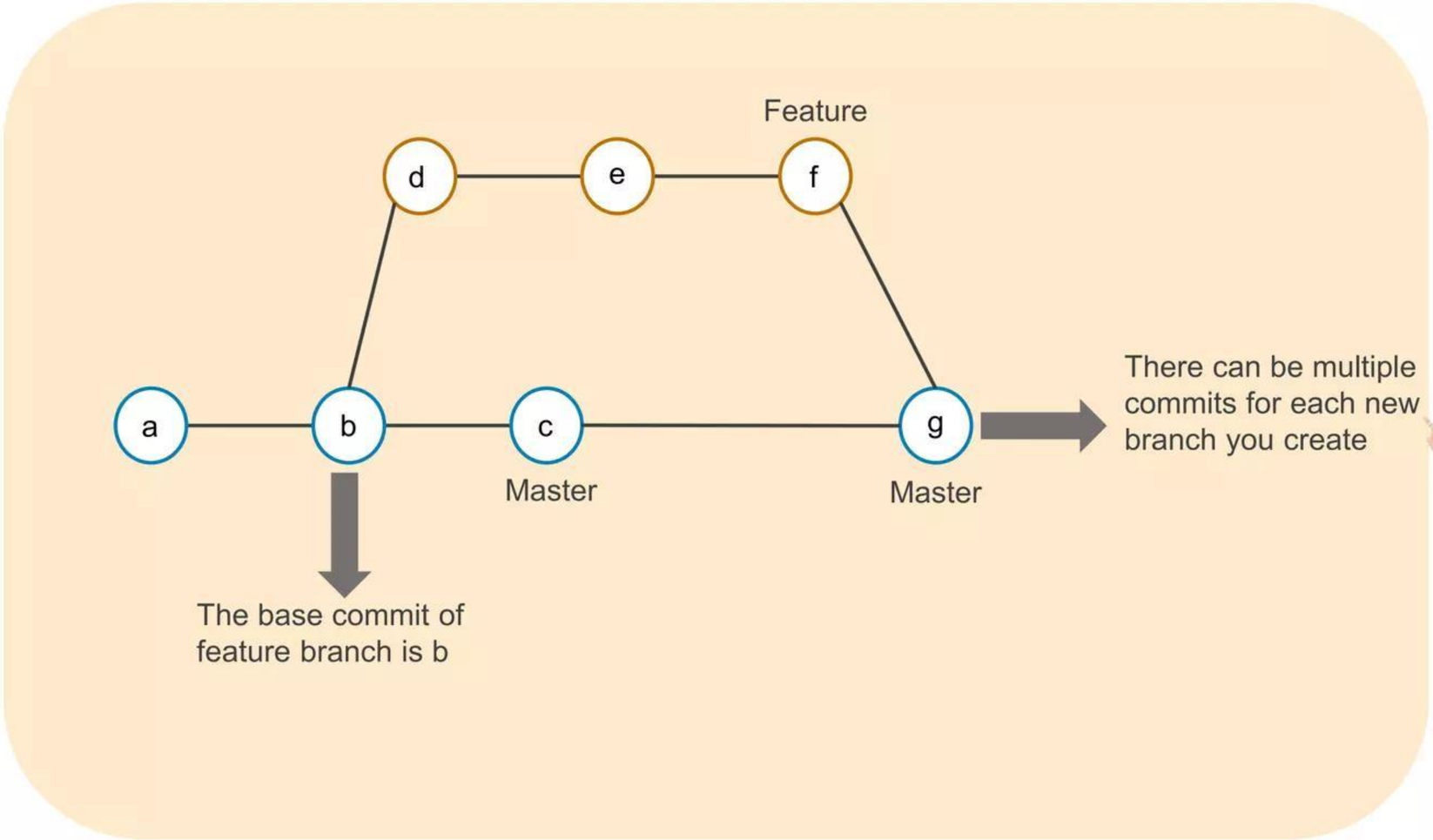
- The diagram shows there are 2 new branches
- You can develop the features you want separately
- After you develop them completely, you can merge the newly created branches to the master branch

Git Merge

Git Merge

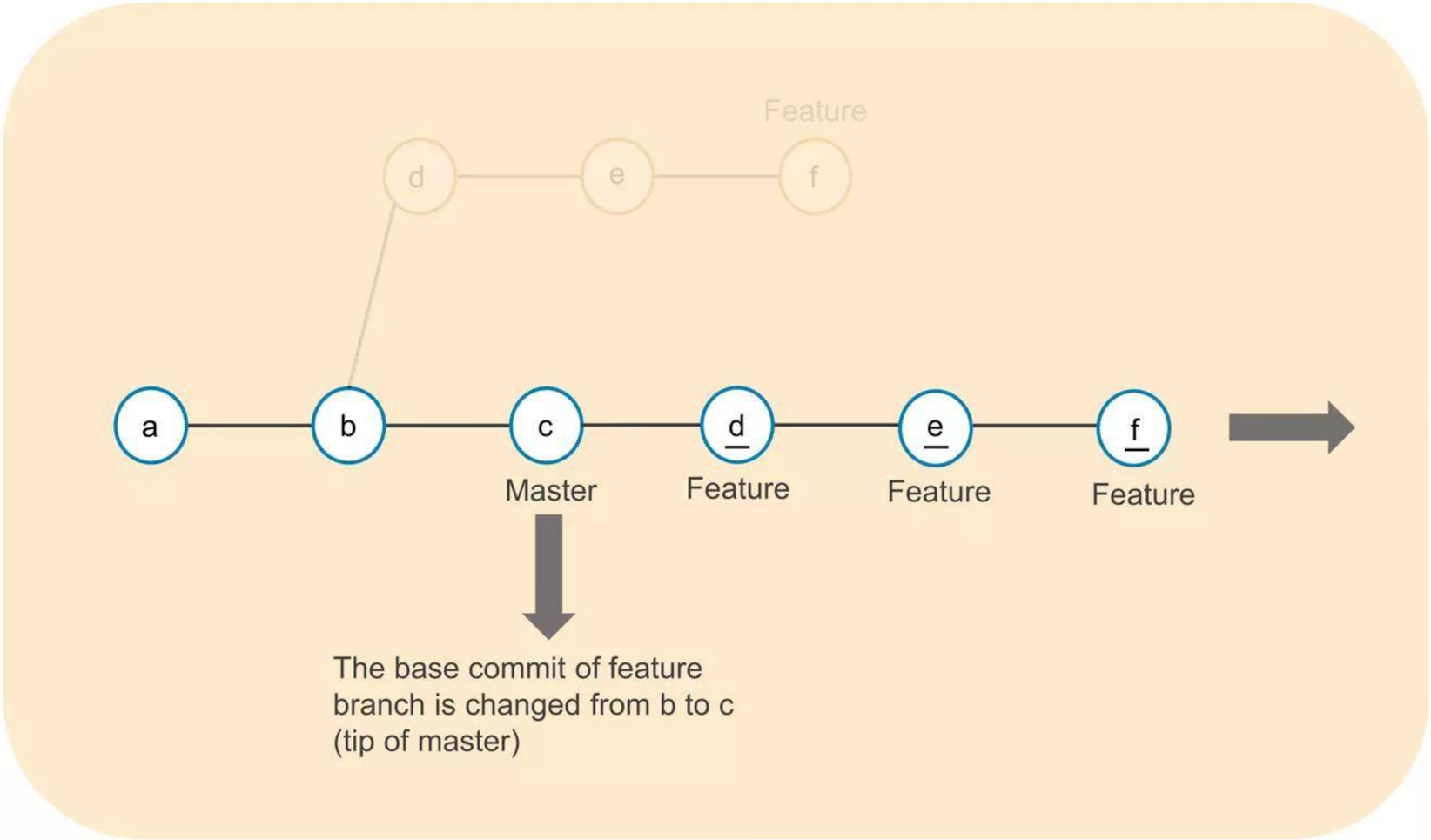


Git Merge

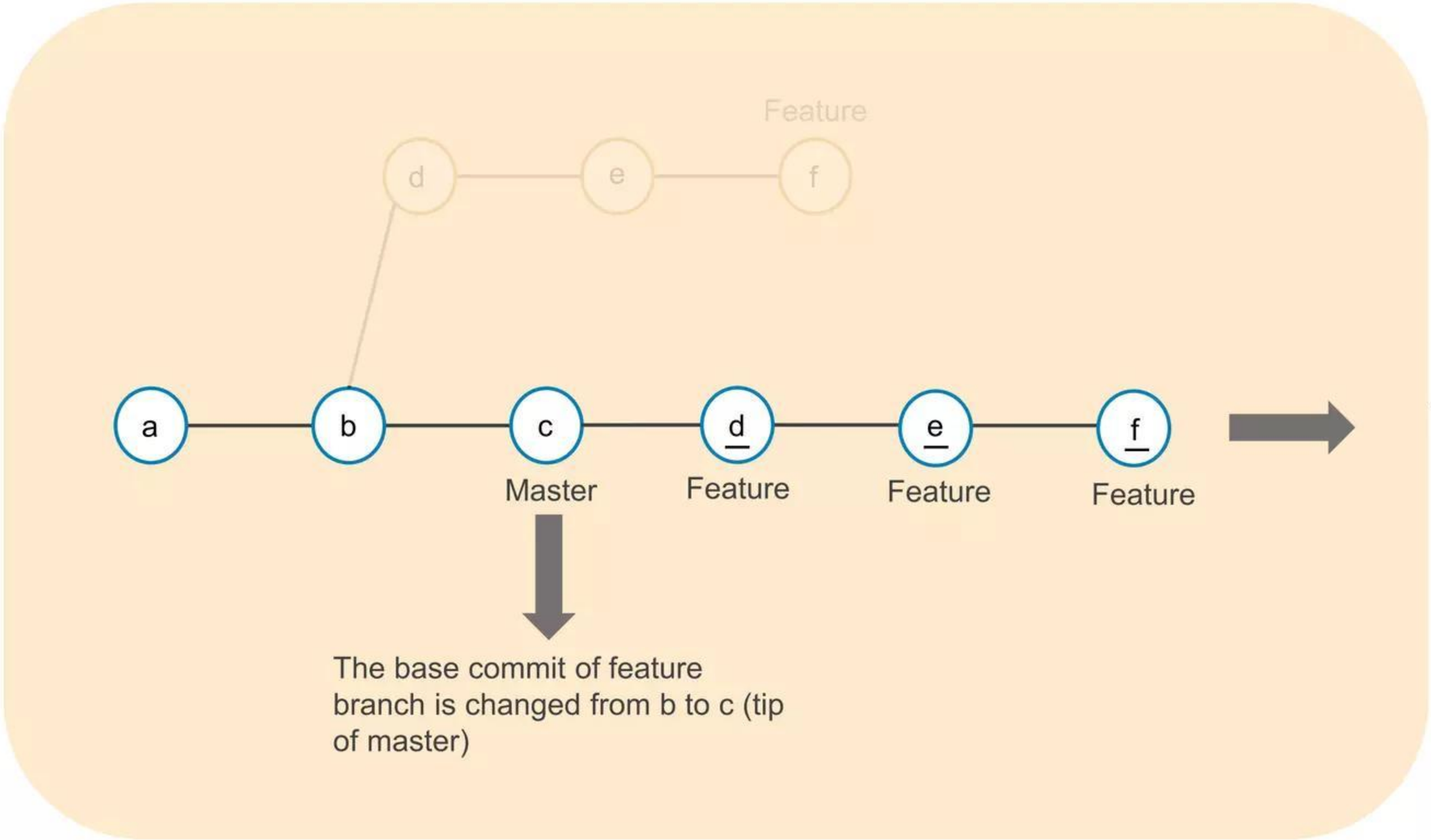


Git Rebase

Git Rebase

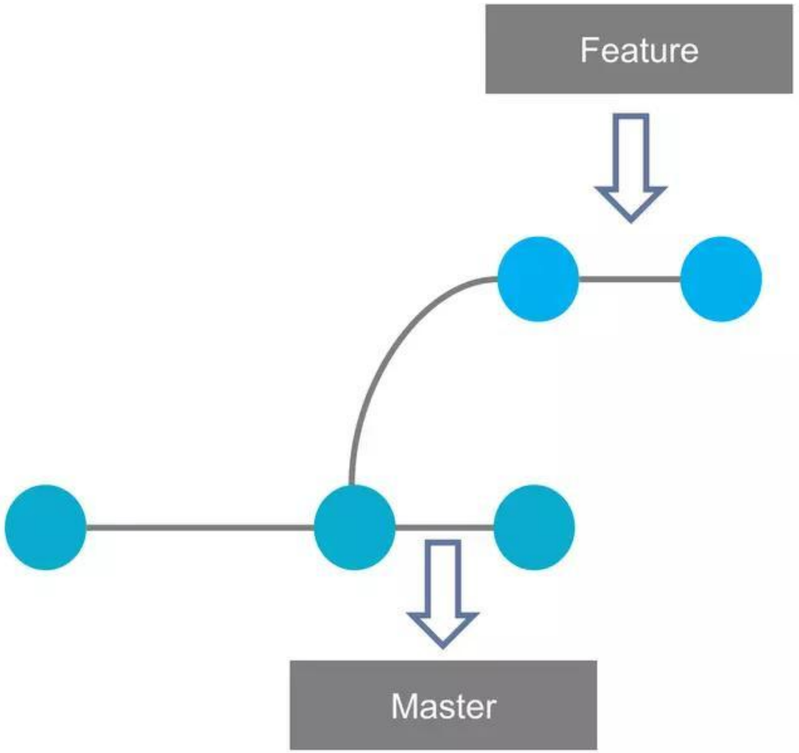


Git Rebase



Git Rebase

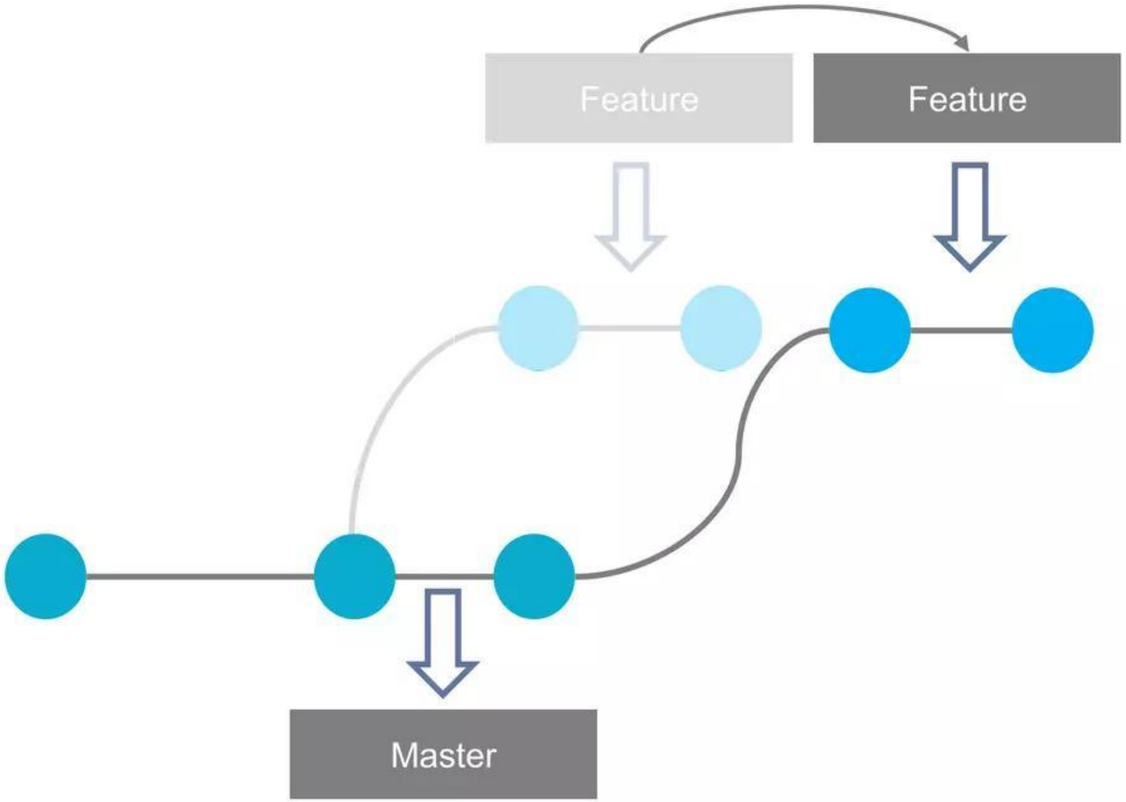
Git Rebase is the process of combining a sequence of commits to a new base commit



- The primary reason for rebasing is to maintain a linear project history

Git Rebase

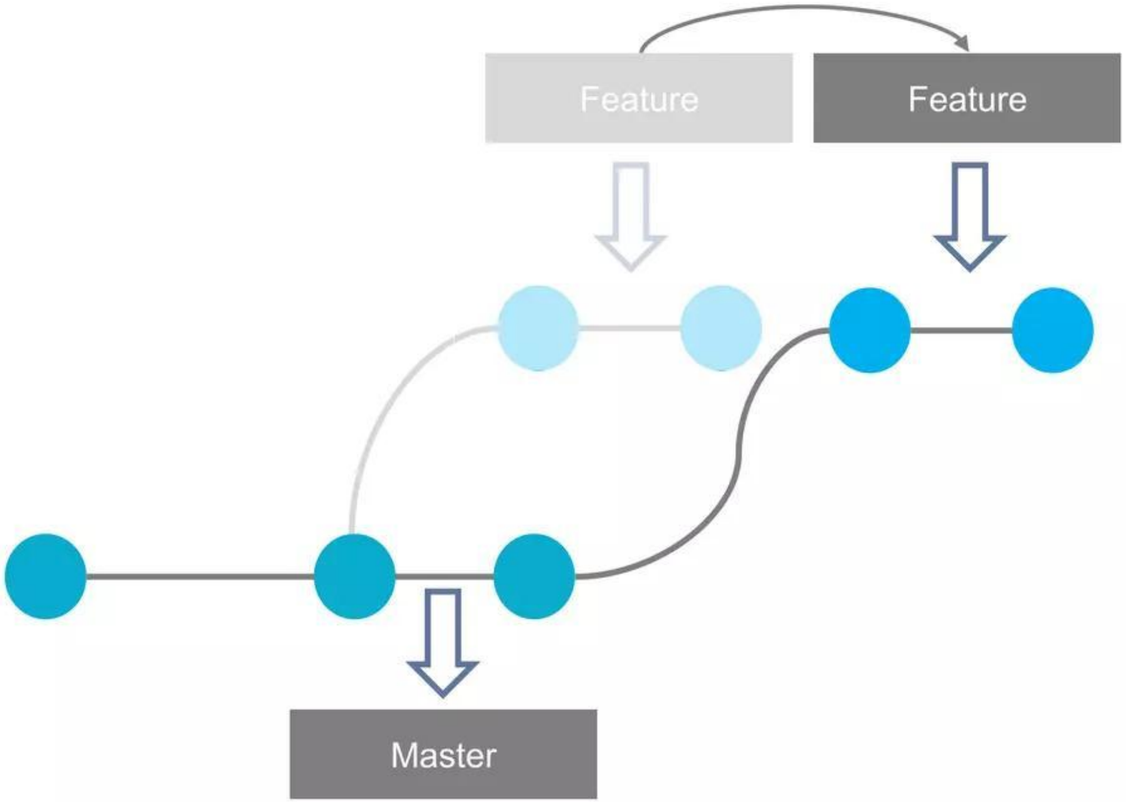
Git Rebase is the process of combining a sequence of commits to a new base commit



- The primary reason for rebasing is to maintain a linear project history
- When you rebase, you "**unplug**" a branch and "**replug**" it on the tip of another branch (usually master)

Git Rebase

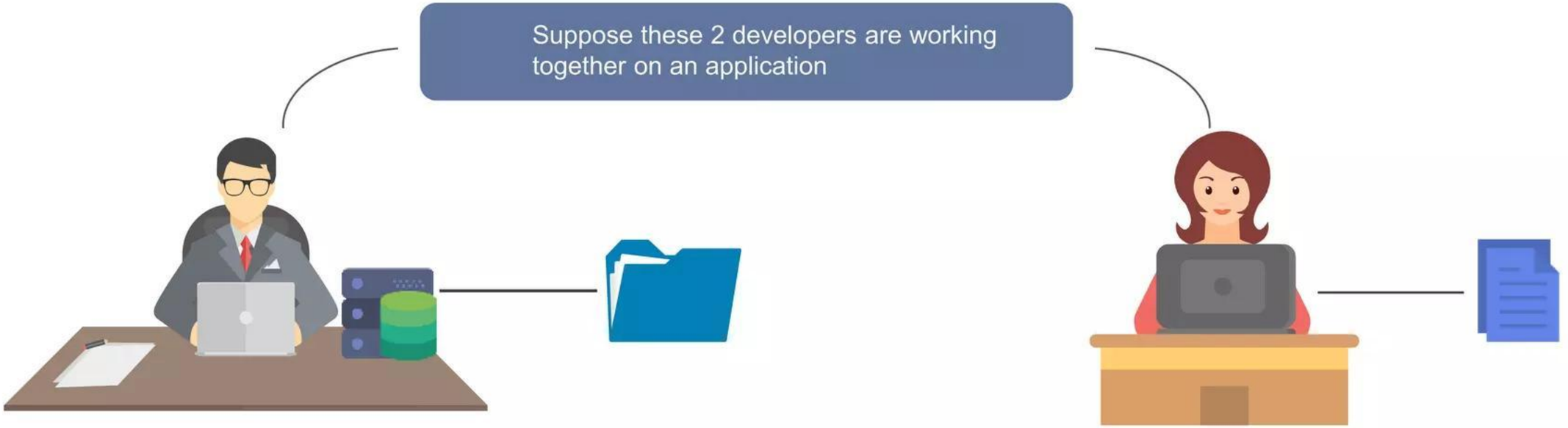
Git Rebase is the process of combining a sequence of commits to a new base commit



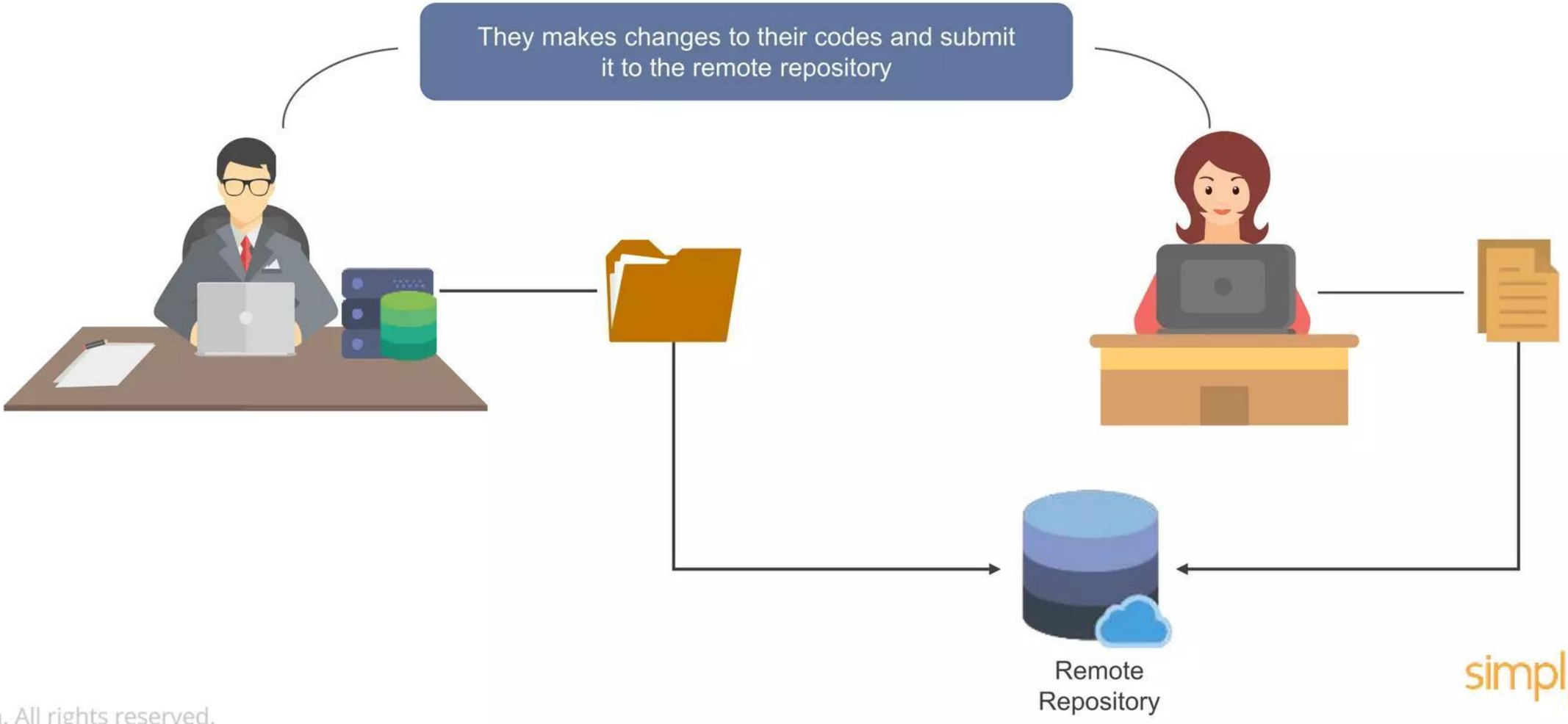
- The primary reason for rebasing is to maintain a linear project history
- When you rebase, you "**unplug**" a branch and "**replug**" it on the tip of another branch (usually master)
- The goal of rebasing is to take all the commits from a feature branch and put it on the master branch

Pull from Remote

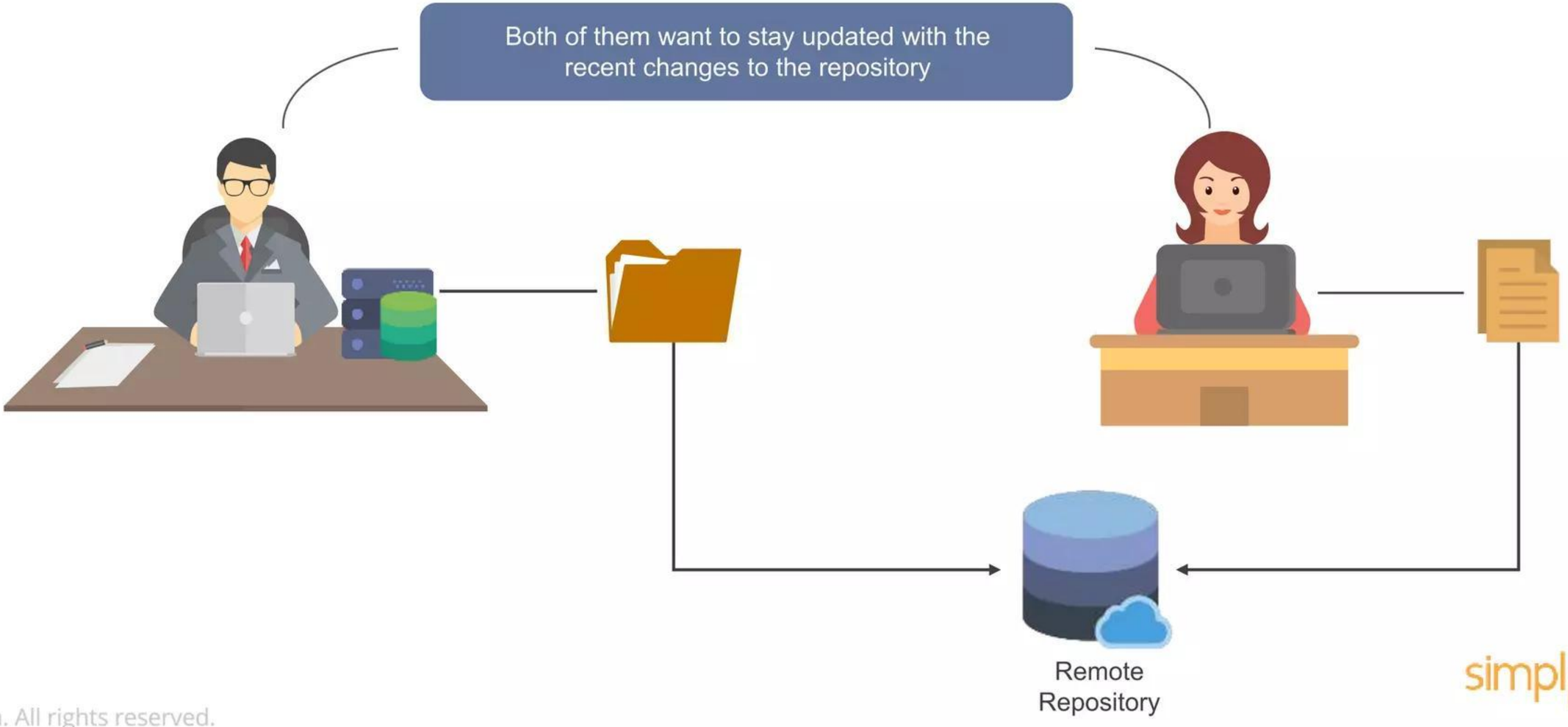
Pull from a Remote



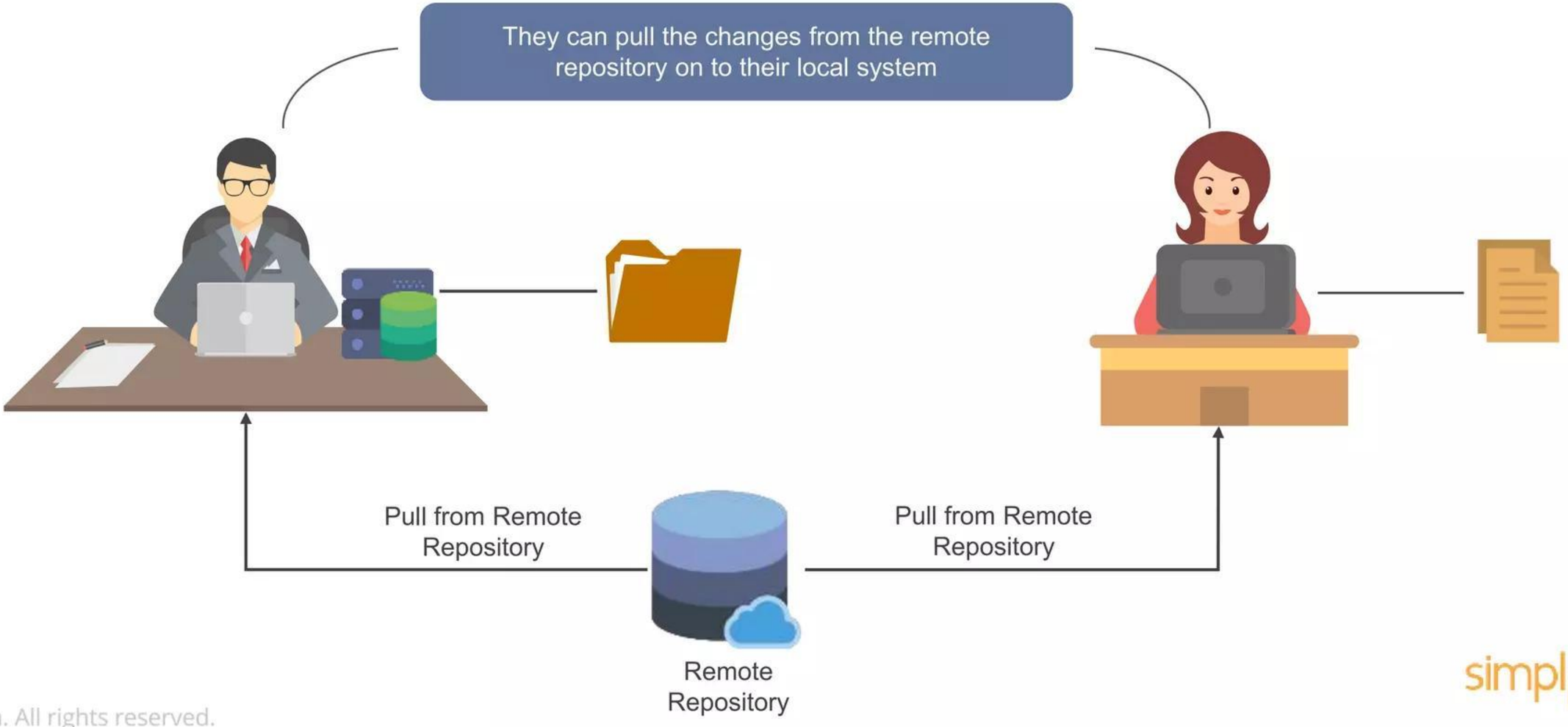
Pull from a Remote



Pull from a Remote

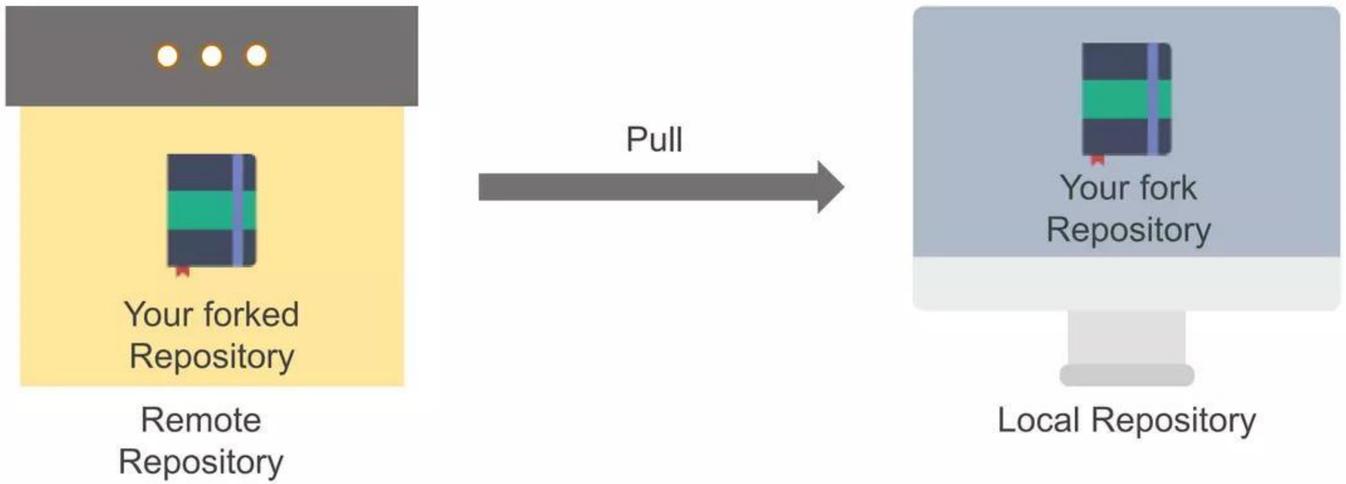


Pull from a Remote



Pull from a Remote

You can pull in any changes that have been made from your forked remote repository to the local repository



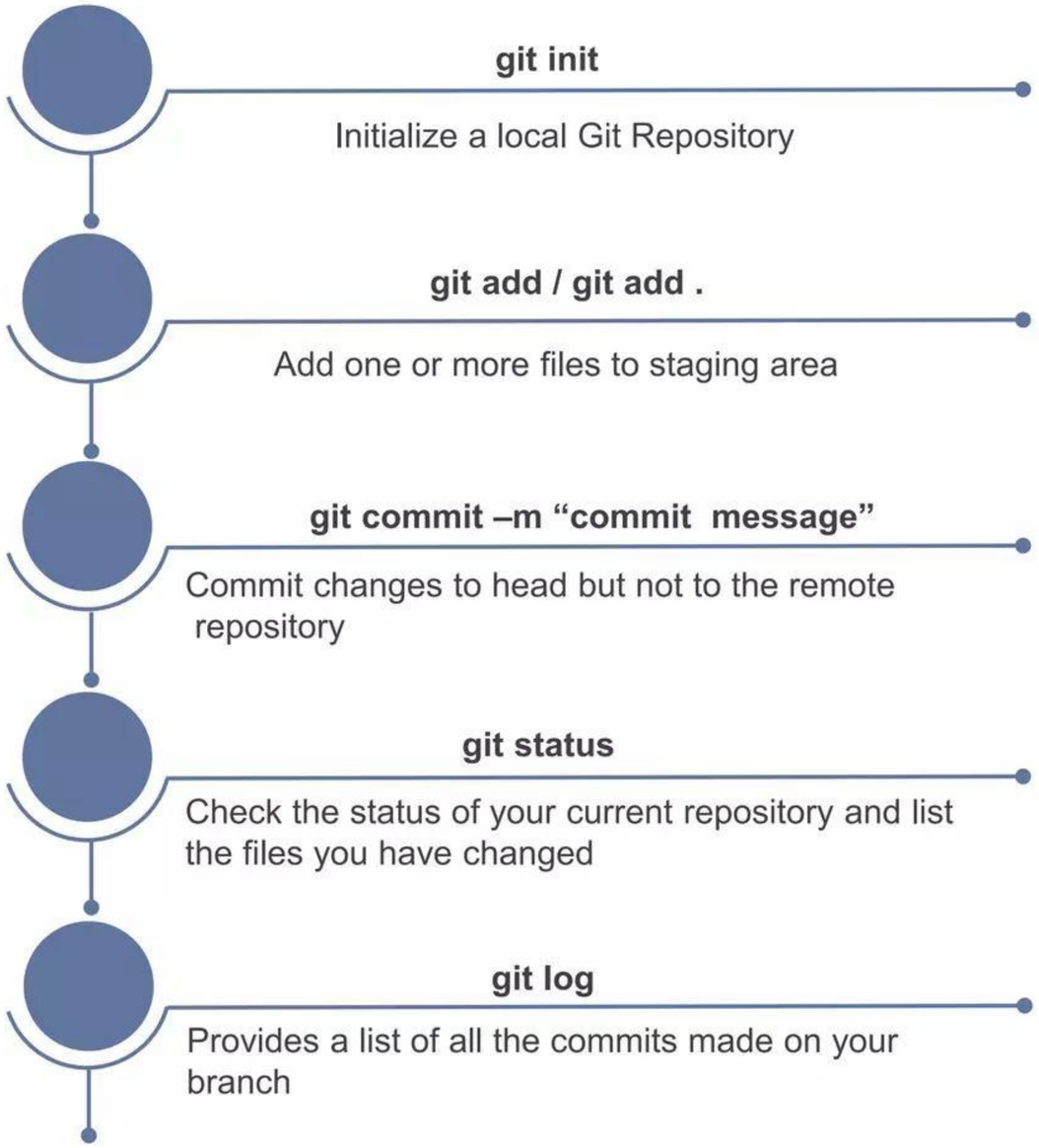
- Use the following command to check if there has been any change

$ git pull <RemoteName> <BranchName>

- If there is no change, it will notify “Already up-to-date”. If there is an change, it will merge those changes to your local repository

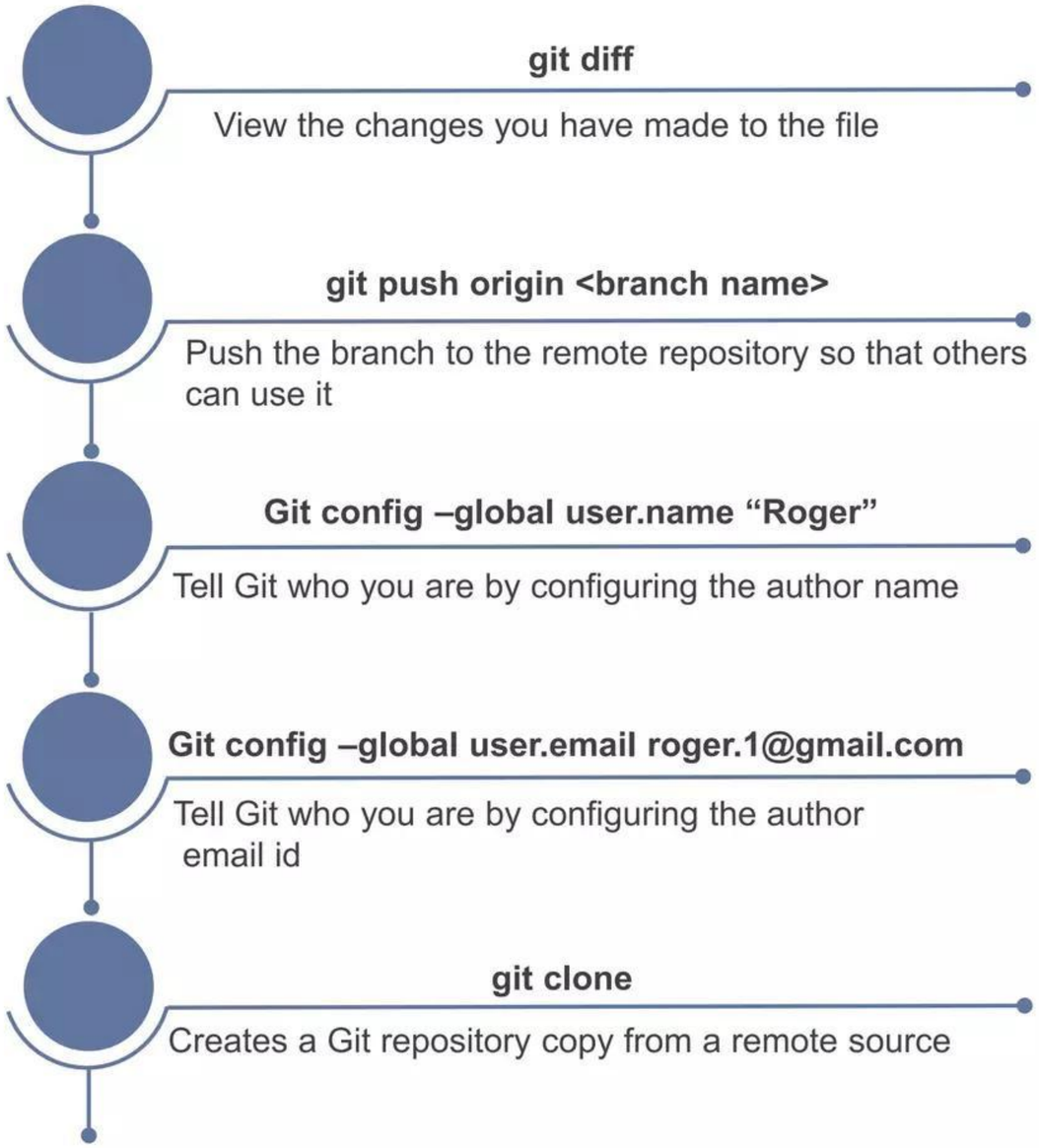
Commands in Git

Popular Git Commands



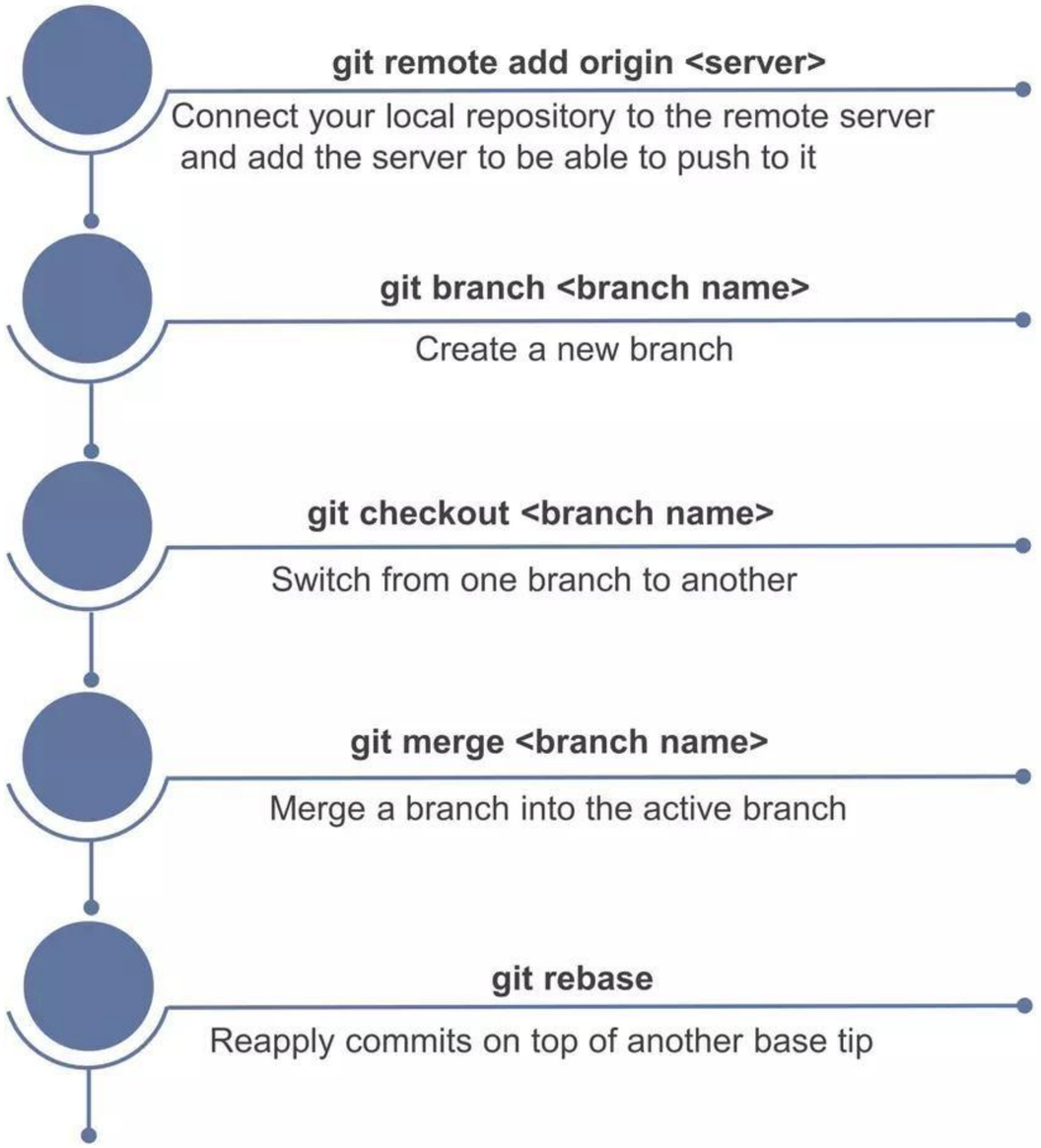
simplilearn

Popular Git Commands



simplilearn

Popular Git Commands



simplilearn

Demo on Git

Demo on Git

1. Create a Repository

Create a "***hello-world***" folder

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~
$ mkdir hello-world
```

Move to hello-world folder

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~
$ cd hello-world
```

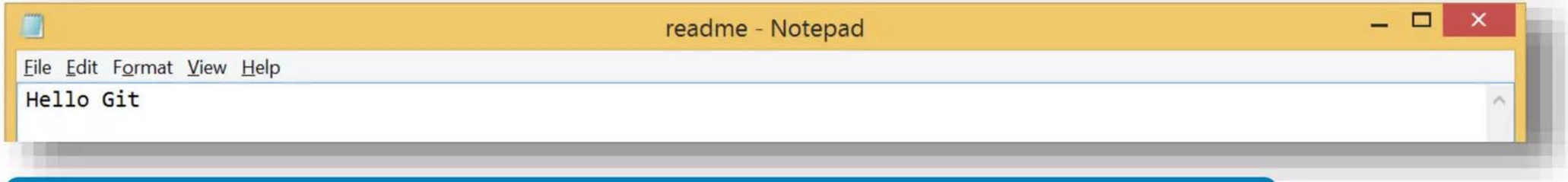
Create a new Git instance for a project

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/hello-world
$ git init
Initialized empty Git repository in C:/Users/Simplilearn/hello-world/.git/
```

Demo on Git

1. Create a Repository

Create a file called ***readme.txt*** in the ***hello-world*** folder



Check the status of the repository to find out if there have been changes

```
SSPL-LP-DNS-YTO+Simplilearn@SSPL-LP-DNS-YTO1 MINGW64 ~/hello-world (master)
$ git status
On branch master

No commits yet

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

        readme.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Demo on Git

2. Create a new file, add something to that file and commit those changes to Git

Add the file you just created to the files you would like to commit to change

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/hello-world (master)
$ git add readme.txt
```

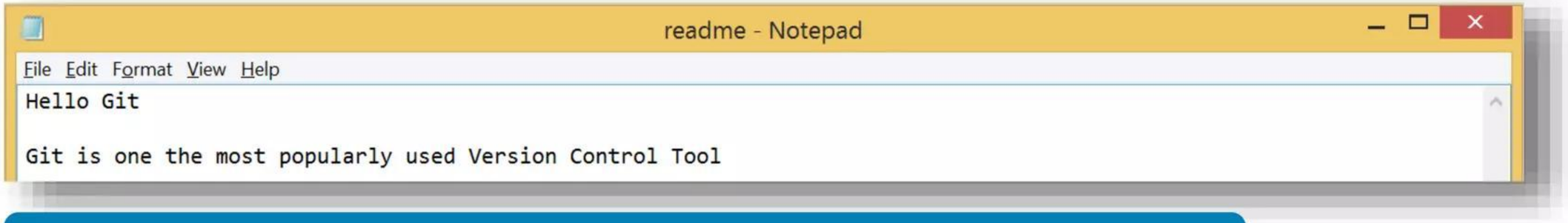
Commit those changes to the repository's history

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/hello-world (master)
$ git commit -m "first commit"
[master (root-commit) 363f726] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 readme.txt
```

Demo on Git

2. Create a new file, add something to that file and commit those changes to Git

Add another line to readme.txt and save



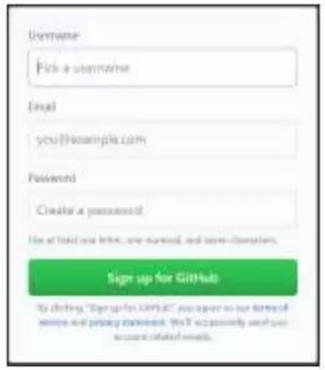
View the difference between the file now and how it was at your last commit

```
SSPL-LP-DNS-YTO+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/hello-world (master)
$ git diff
diff --git a/readme.txt b/readme.txt
index e51ca0d..5b9be25 100644
--- a/readme.txt
+++ b/readme.txt
@@ -1 +1,6 @@
-Hello Git
\ No newline at end of file
+Hello Git
+
+Git is one the most popularly used Version Control Tool
```

Demo on Git

3. Create a GitHub account, add username to your Git configuration

Sign up to GitHub account by visiting github.com



Add your GitHub username to your Git Configuration

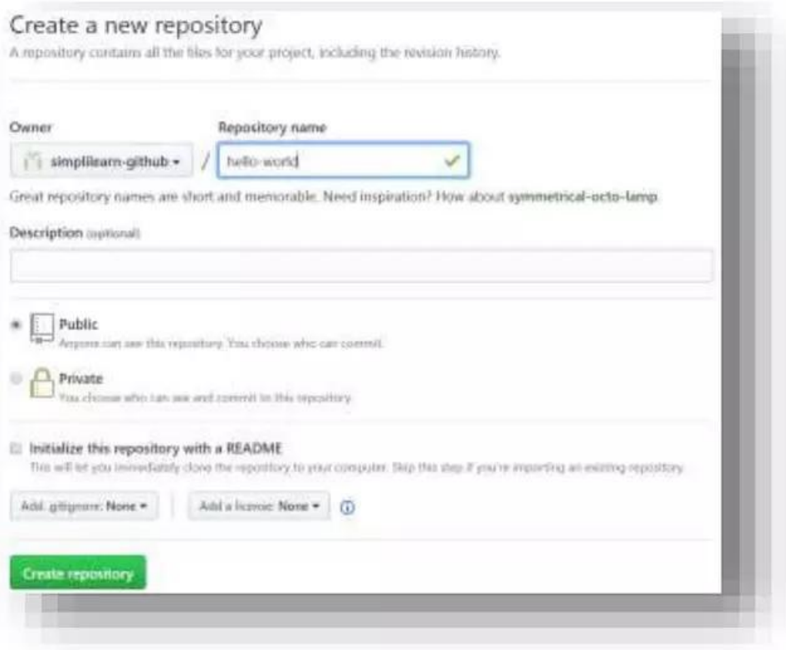
```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/hello-world (master)
$ git config --global user.name simplilearn-github
```

Demo on Git

4. Connect your local and remote repositories and push changes

Create a Remote repository

Github.com → Log in → Click the + sign in the top right



Demo on Git

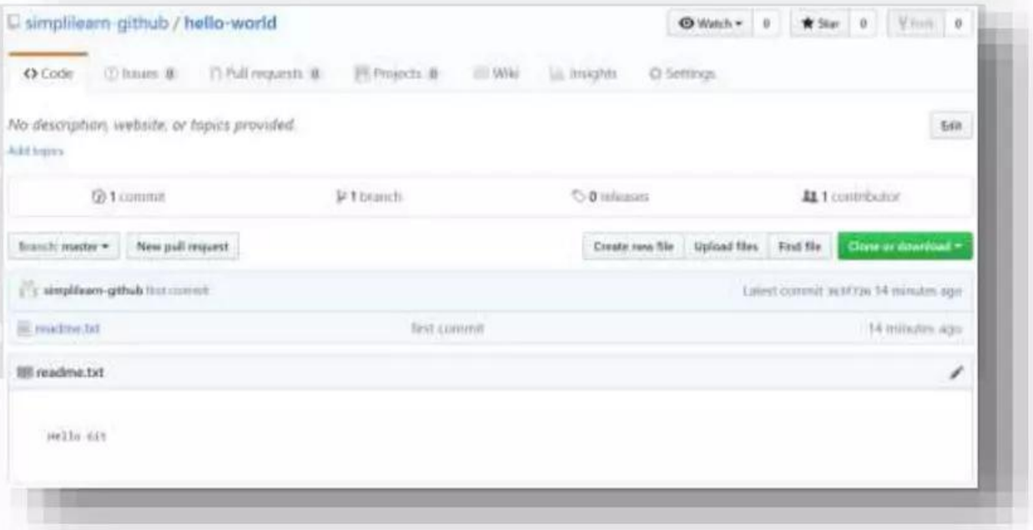
4. Connect your local and remote repositories and push changes

Connect your local to your Remote

```
SSPL-LP-DNS-YTO+Simplilearn@SSPL-LP-DNS-YTO1 MINGW64 ~/hello-world (master)
$ git remote add origin https://github.com/simplilearn-github/hello-world.git
```

Push your file to remote

```
$ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), do
Writing objects: 100% (3/3), 225
Total 3 (delta 0), reused 0 (del
To https://github.com/simplilear
 * [new branch]      master -> m
```

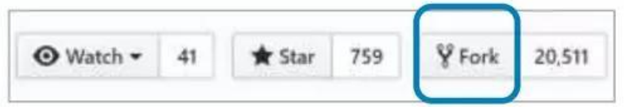


Demo on Git

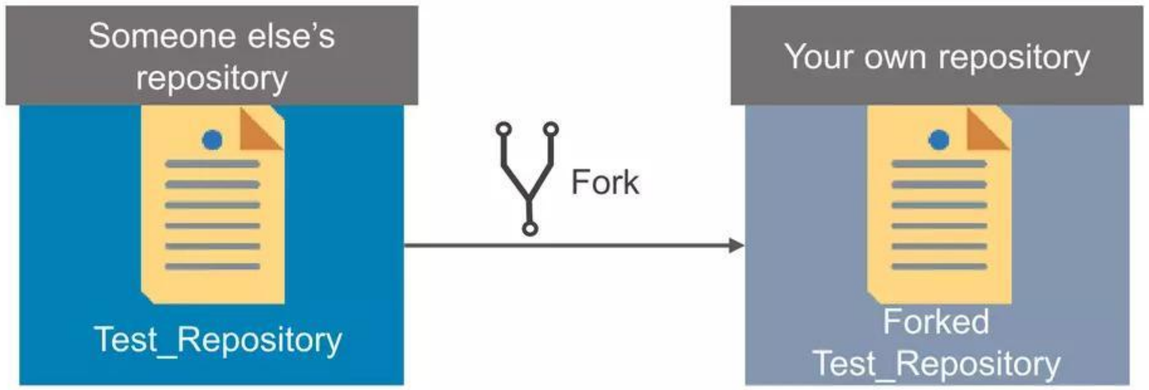
5. Fork a project from github.com and clone it locally

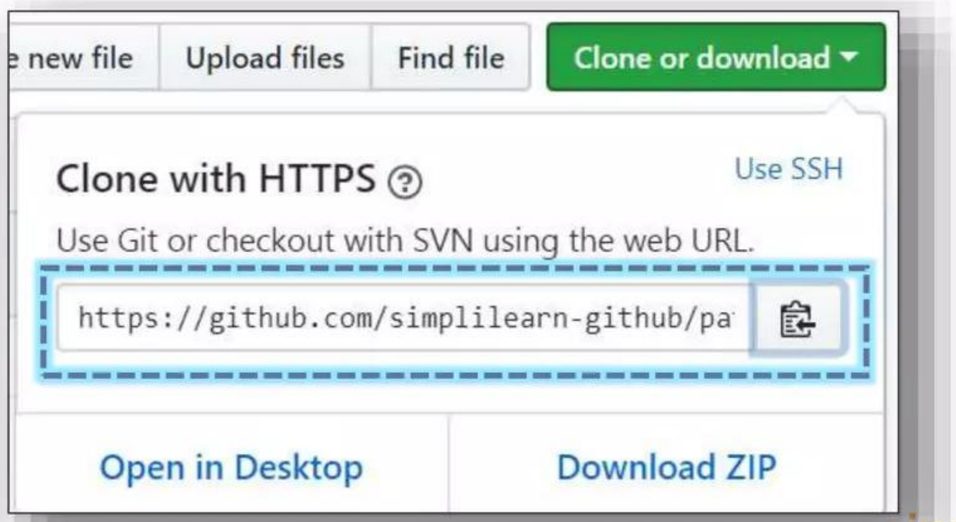
Fork Patchwork Repository: We'll be using github.com/jlord/patchwork

Github.com/jlord/patchwork → Click the fork button at the top right →



Once the fork is complete, you will get a copy on your account. Copy your fork's HTTP URL on the right sidebar





Demo on Git

5. Fork a project from github.com and clone it locally

Change the directory

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/hello-world (master)
$ cd ..
```

Clone the repository onto your computer

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~
$ git clone https://github.com/simplilearn-github/patchwork.git
Cloning into 'patchwork'...
remote: Counting objects: 303549, done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 303549 (delta 12), reused 26 (delta 11), pack-reused 303518
Receiving objects: 100% (303549/303549), 114.75 MiB | 627.00 KiB/s, done.
Resolving deltas: 100% (139049/139049), done.
Checking out files: 100% (1285/1285), done.
```

Demo on Git

5. Fork a project from github.com and clone it locally

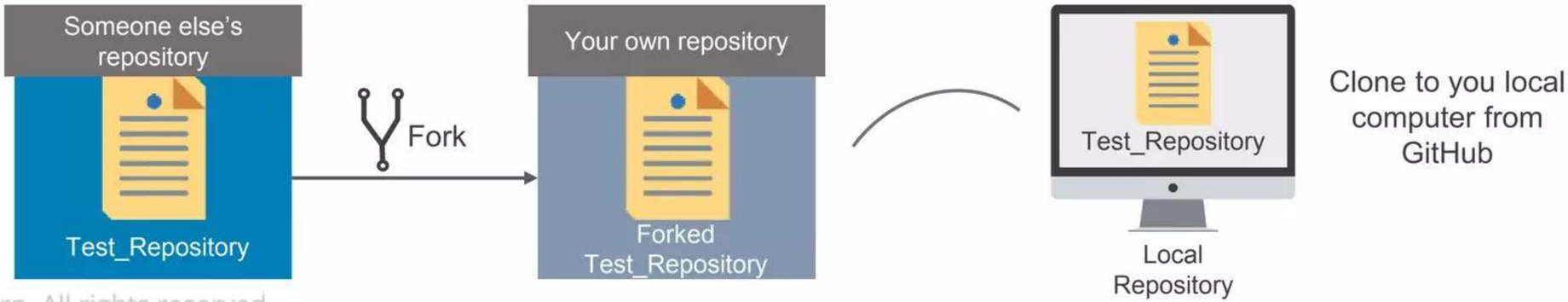
Go into the folder for the fork it created

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~
$ cd patchwork
```

Now, you have got a copy of the repository on your local computer which is automatically connected to the remote repository

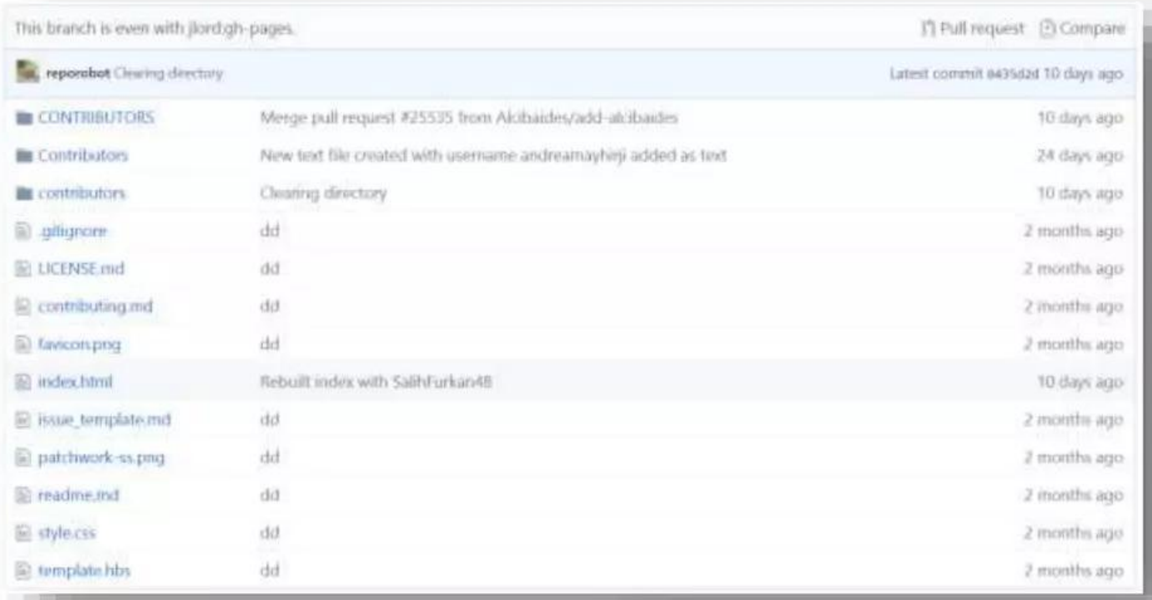
If the original repository you forked from has some changes, you'd want to pull those changes as well. So, you need to add another remote connection to the original repository with its URL

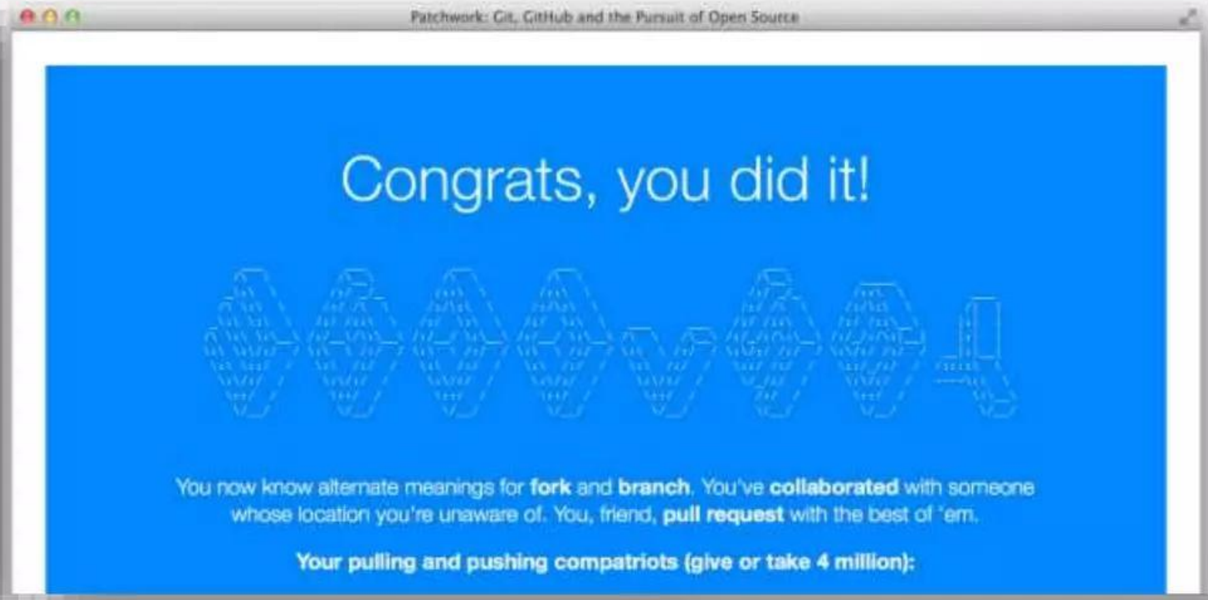
```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/patchwork (gh-pages)
$ git remote add upstream https://github.com/jlord/patchwork.git
```

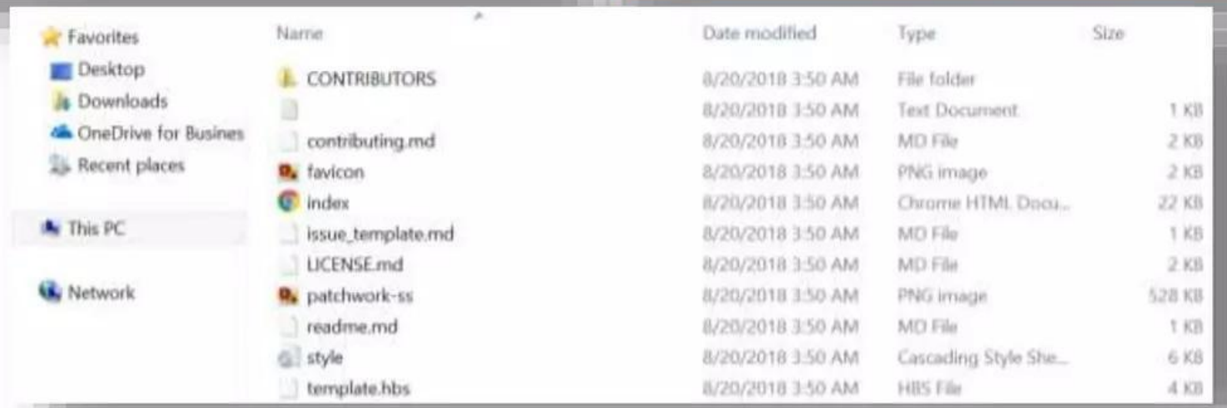


Demo on Git

5. Fork a project from github.com and clone it locally







simplilearn

Demo on Git

6. Create a new branch on your fork for your contribution

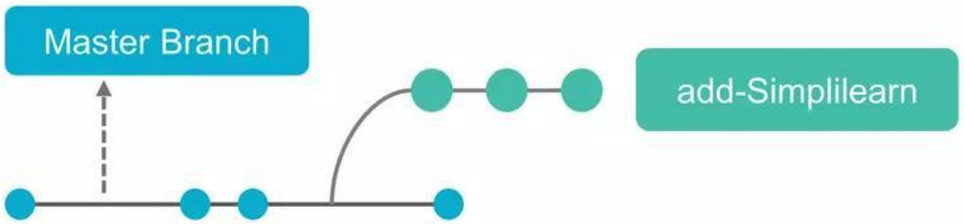
Create a new branch

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/patchwork (gh-pages)
$ git status
On branch gh-pages
Your branch is up to date with 'origin/gh-pages'.

nothing to commit, working tree clean
```

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/patchwork (gh-pages)
$ git branch add-Simplilearn
```

Name of the branch is:
add-Simplilearn



Demo on Git

6. Create a new branch on your fork for your contribution

Checkout your branch and go onto a new branch

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/patchwork (gh-pages)
$ git checkout add-Simplilearn
Switched to branch 'add-Simplilearn'
```

Push your update to your fork on GitHub

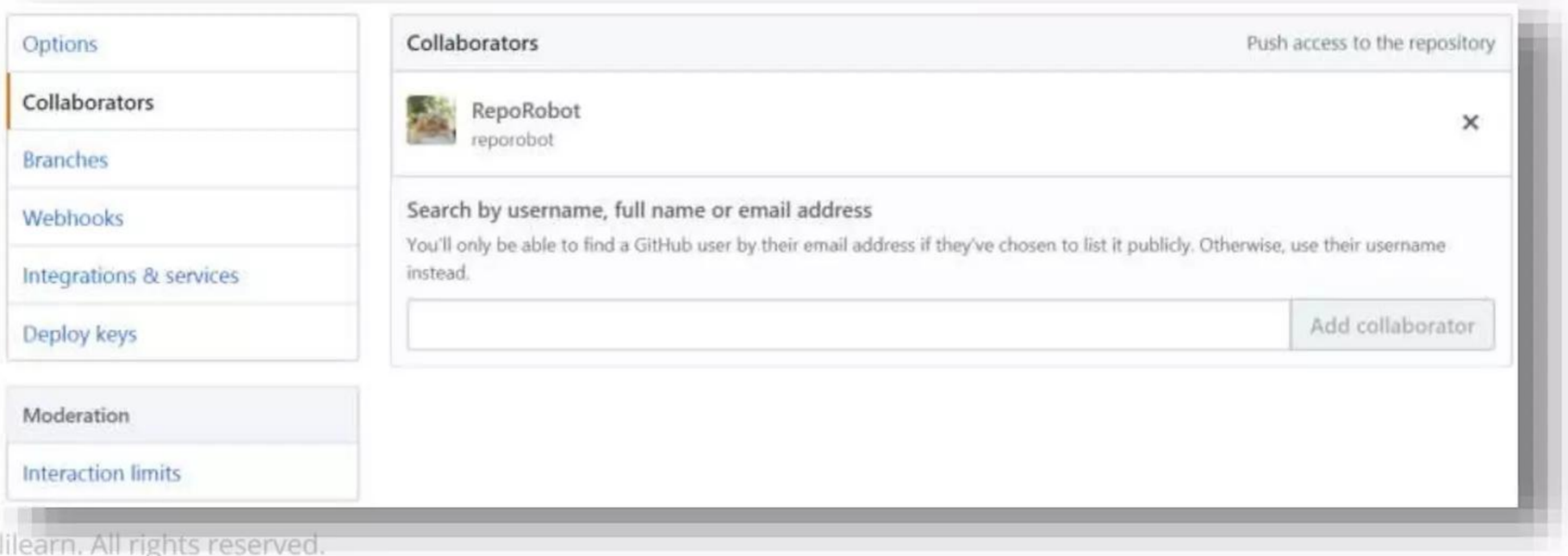
```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/patchwork (add-Simplilearn)
$ git push origin add-Simplilearn
Total 0 (delta 0), reused 0 (delta 0)
To https://github.com/simplilearn-github/patchwork.git
 * [new branch]            add-Simplilearn -> add-Simplilearn
```

Demo on Git

7. Add a collaborator to your project

Add “reporobot” as a collaborator to your forked Patchwork repository’s page

Visit the repository’s GitHub page → Click settings icon on the right side menu → Select the collaborators tab → Type the username and click add



Demo on Git

8. Keep your file up to date by pulling in changes from collaborators

Check if ***Reporobot*** has made any changes to your branch

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/patchwork (add-Simplilearn)
$ git pull origin add-Simplilearn
From https://github.com/simplilearn-github/patchwork
 * branch                  add-Simplilearn -> FETCH_HEAD
Already up to date.
```

It will give a message "***Already up to date***", if nothing has changed. If there are changes, it will merge those changes into your local version.

Demo on Git

9. Create a new branch, make changes and merge it to the master branch

Create a new local repository called test

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~
$ mkdir test
```

Move to the test folder

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~
$ cd test
```

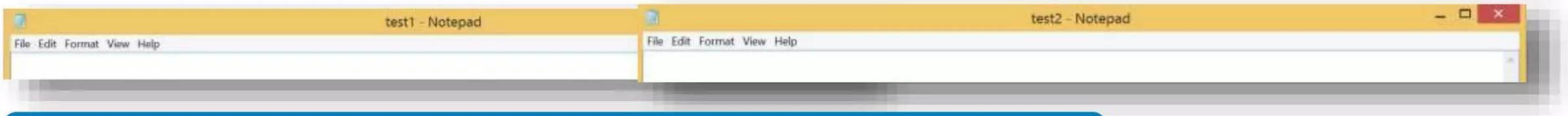
Create a new git instance

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test
$ git init
Initialized empty Git repository in C:/Users/Simplilearn/test/.git/
```

Demo on Git

9. Create a new branch, make changes and merge it to the master branch

Create 2 text files “**test1**” and “**test2**” in the test repository



Add the two files to master branch and make a commit

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test (master)
$ git add .
```

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test (master)
$ git commit -m "files committed"
[master (root-commit) d4fbb17] files committed
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test1.txt
 create mode 100644 test2.txt
```

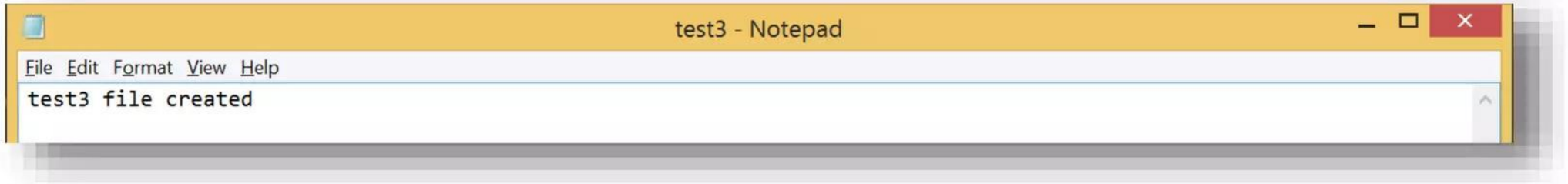
Demo on Git

9. Create a new branch, make changes and merge it to the master branch

Create a new branch "**test_branch**"

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test (master)
$ git branch test_branch
```

Create a new text file "**test3**"



Demo on Git

9. Create a new branch, make changes and merge it to the master branch

Add test3 file to the new branch

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test (master)
$ git add test3.txt
```

Move from the master branch to test_branch

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test (master)
$ git checkout test_branch
Switched to branch 'test_branch'
A       test3.txt
```

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test (master)
$ ls
test1.txt  test2.txt
```

Demo on Git

9. Create a new branch, make changes and merge it to the master branch

Merge test_branch file to master branch

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test (master)
$ git merge test_branch
Updating d4fbb17..4233046
Fast-forward
 test3.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 test3.txt
```

The master branch has all the files now

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test (master)
$ ls
test1.txt  test2.txt  test3.txt
```

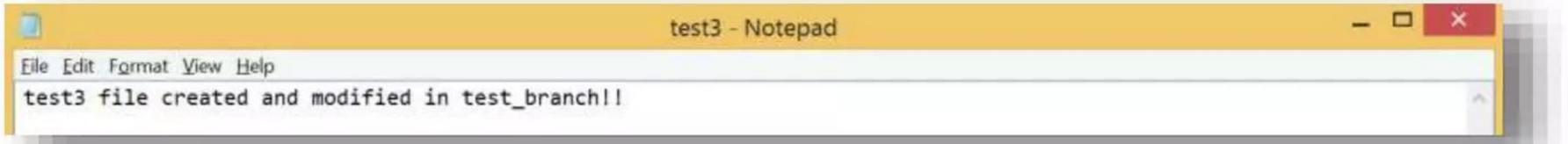
Demo on Git

9. Create a new branch, make changes and merge it to the master branch

Move from the master branch to test_branch

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test (master)
$ git checkout test_branch
Switched to branch 'test_branch'
```

Modify test3 file and commit the changes




```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test (test_branch)
$ git commit -a -m "test3 file modified"
[test_branch e2e2854] test3 file modified
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Demo on Git

9. Create a new branch, make changes and merge it to the master branch

Check the file in test_branch

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test (test_branch)
$ cat test3.txt
test3 file created and modified in test_branch!!
```

Switch to the master branch

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test (test_branch)
$ git checkout master
Switched to branch 'master'
```

Check the test3 file in master branch

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test (master)
$ cat test3.txt
test3 file created
```

The file has not been modified in the master branch

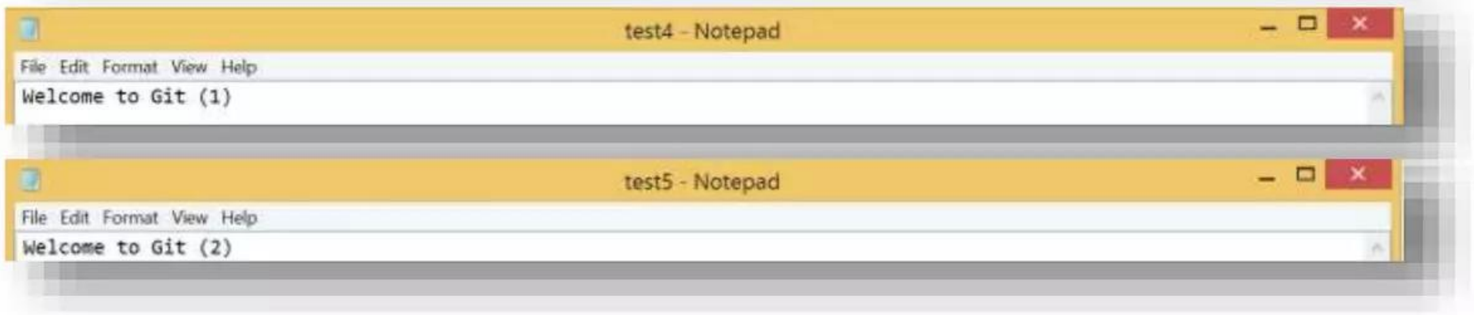
Demo on Git

10. Rebase the newly created files on to master branch

Switch to test_branch

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test (master)
$ git checkout test_branch
Switched to branch 'test_branch'
```

Create 2 new text files **test4** and **test5**



Demo on Git

10. Rebase the newly created files on to master branch

Add the files to test branch

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test (test_branch)
$ git add -A
```

Commit the files for rebasing

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test (test_branch)
$ git commit -a -m "adding for rebasing"
[test_branch 7d7c2cb] adding for rebasing
 2 files changed, 2 insertions(+)
 create mode 100644 test4.txt
 create mode 100644 test5.txt
```

Demo on Git

10. Rebase the newly created files on to master branch

List the files in test_branch

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test (test_branch)
$ ls
test1.txt  test2.txt  test3.txt  test4.txt  test5.txt
```

Switch to the master branch

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test (test_branch)
$ git checkout master
Switched to branch 'master'
```

Demo on Git

10. Rebase the newly created files on to master branch

List the files in the master branch

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test (master)
$ ls
test1.txt  test2.txt  test3.txt
```

Switch to test_branch

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test (master)
$ git checkout test_branch
Switched to branch 'test_branch'
```

Demo on Git

10. Rebase the newly created files on to master branch

Rebase master branch

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test (test_branch)
$ git rebase master
Current branch test_branch is up to date.
```

Switch to master branch

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test (test_branch)
$ git checkout master
Switched to branch 'master'
```

Demo on Git

10. Rebase the newly created files on to master branch

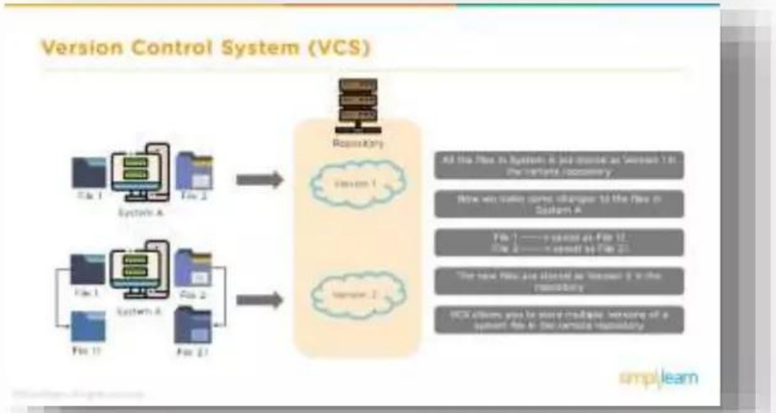
Rebase test_branch

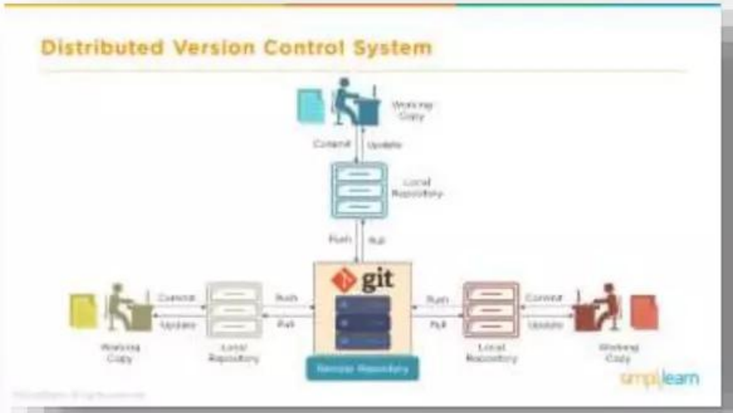
```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test (master)
$ git rebase test_branch
First, rewinding head to replay your work on top of it...
Fast-forwarded master to test_branch.
```

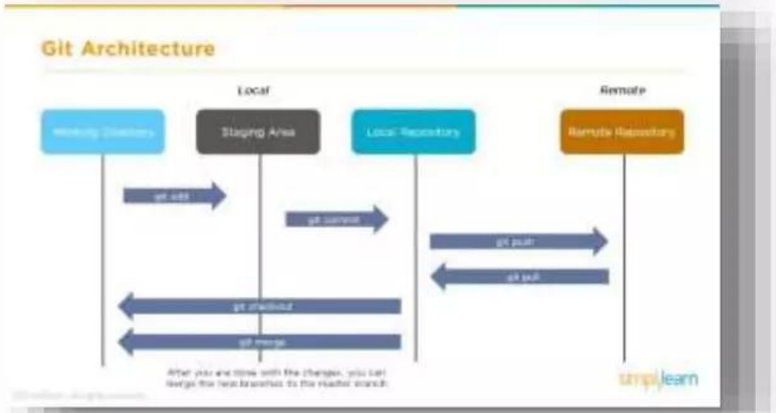
List the files in master branch

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test (master)
$ ls
test1.txt  test2.txt  test3.txt  test4.txt  test5.txt
```

Key Takeaways







simplilearn


THANK YOU
For more information, visit
www.simplilearn.com
simplilearn