</talentlabs>

CHAPTER 1

Git & GitHub

</talentlabs>

AGENDA

- Version Control System
- Git & GitHub
- Creating Git Repository
- Pushing Changes to GitHub
- Working with Git Branches

Version Control System (VCS)

How do you track your document changes?

Proposal_V1.docx

Proposal_V2.docx

Proposal_Final.docx

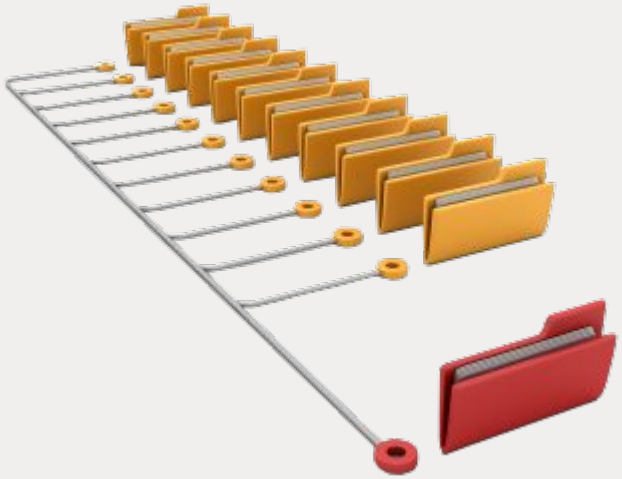
Proposal_Final_Final .docx

Proposal_Final_
Final_Final.docx

</talentlabs>

Challenges in tracking code files

01



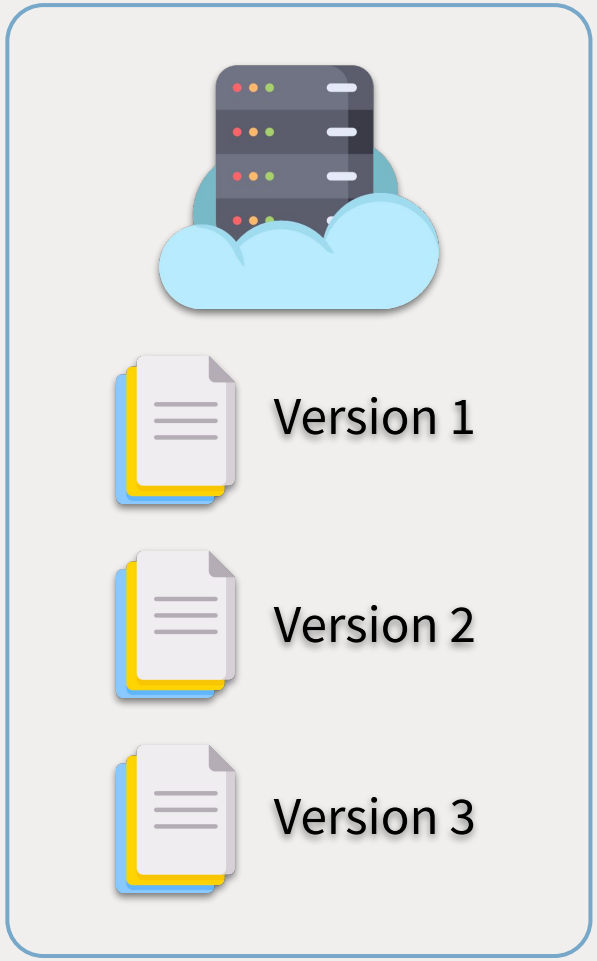
Many files
Many changes

02

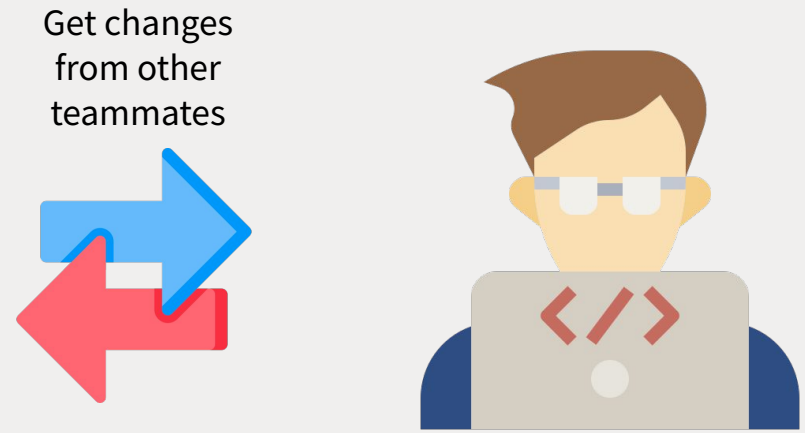
Many people are making changes

</talentlabs>

Version Control System



Version Control System

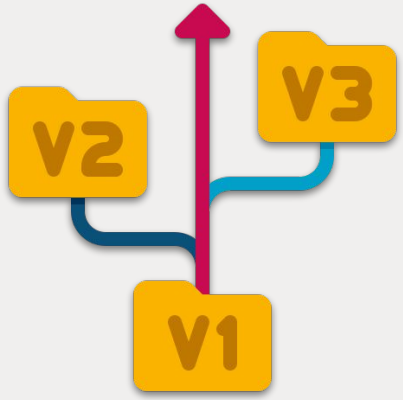


Software Engineer

</talentlabs>

Git & GitHub

What is Version Control System (VCS)?



Keep all the history of changes from everyone

Recover an old version if needed

Help to find answers for changes: What, who, when, why

What is Git?

- invented by Linus Torvalds in 2005
- the most commonly used VCS
- open sourced
- totally free

Who is providing Git Service?

- Git is the **Protocol**, but we need Git Servers so we can use it

- Most companies/developers would use one of the Git providers below:

GitHub Bitbucket GitLab

GitHub

History

- Founded in 2008
- Acquired by Microsoft in 2018
- The biggest code repository in the world

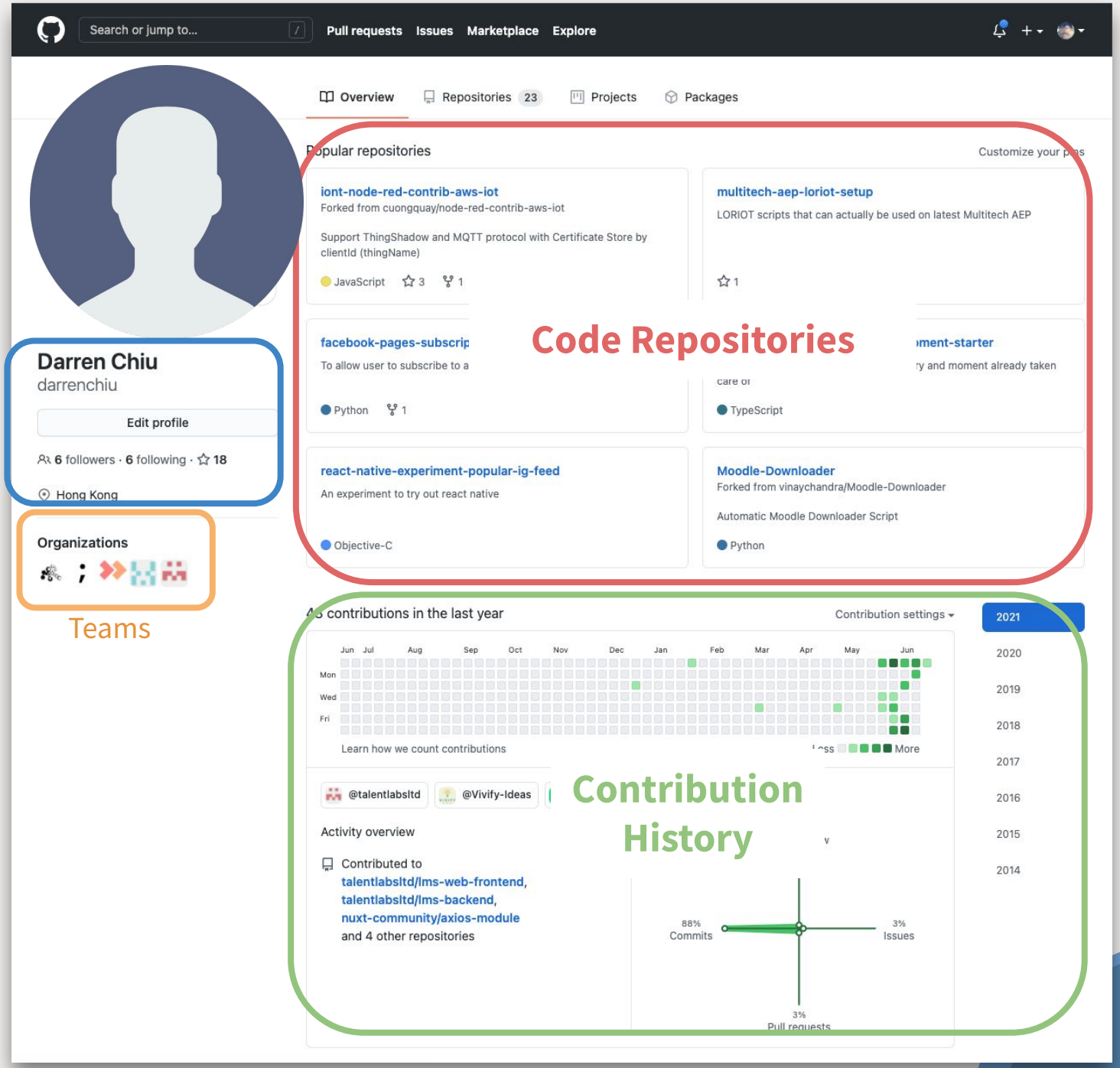
Offerings

- Free and paid version
- Paid version: advanced team collaboration

Alternative Use

- Showcasing coding skills and experiences
- Personal code storage

A Sample GitHub Profile



</talentlabs>

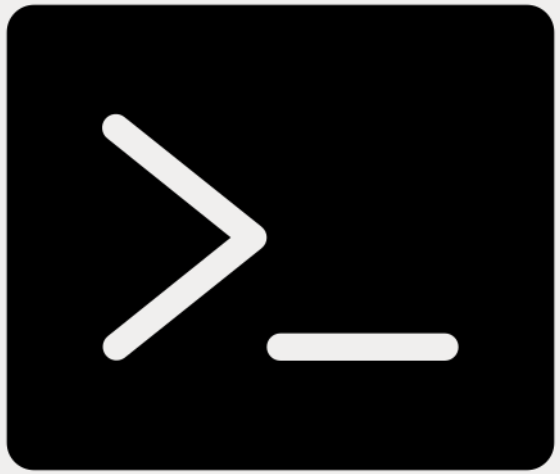
Creating GitHub Repository

Git Repository

Repository

- A git project that **contains all of the files** and **folders** associated with a project, along with each file's **revision history**
- Basically it is just a **project folder**, just it is version controlled (storing all the histories of the changes)

Prior Setup 1:
Command Line Knowledge Required



	Command	Meaning
01	cd	
02	mv	
03	cp	

</talentlabs>

Prior Setup 2: Installing Git

Mac Users: run "git --version" in command line and follow the instructions

Windows Users:

1. Download the latest version of Git from https://git-scm.com/download/win (Standalone Installer, 64-bit Git for Windows Setup)
2. Open up the installer and just keep pressing next.

Prior Setup 3:
Generate Personal Access Token

1. Setup a GitHub account at https://github.com
2. Following the step-by-step guide at
(link also included in the lab assignment manual):

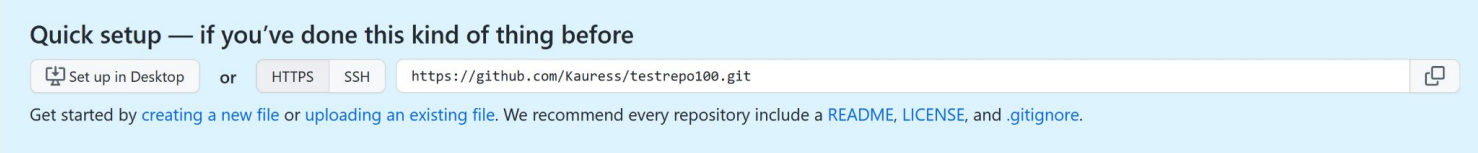
https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token

</talentlabs>

How to create a Git Repository?

1. **Login** on GitHub and **create** a **new** repository.
   Give your repository a **name**. You do not need to add a README file
   ⚠️ Do not check-mark: *choose a license*, or *add .gitignore*.

2. **Copy the url** of the repository:
   https://github.com/username/yourRepositoryName.git



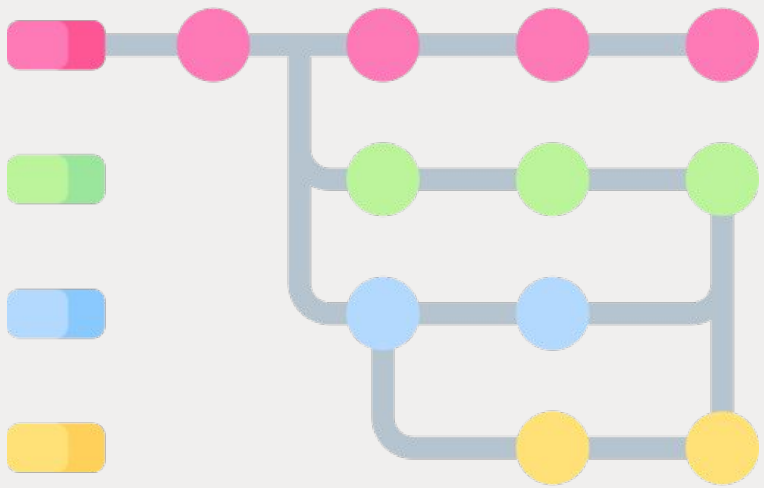
3. **Link the Repo** on GitHub to your local project folder
   a. *Navigate to your folder from your terminal (by using cd command)*
   b. *git init*
   c. *git remote add origin url (url should be replaced with the url from step 2)*

</talentlabs>

Pushing Changes to GitHub

Git Branch

- Branching lets you have **different versions of a repository** at one time
- A Git branch represents an **independent line** of development
- You can work on the branch so you won't affect the main one



Master/Main —— Default Main branch

</talentlabs>

Pushing Changes to GitHub

01

git add

Adds file changes in your working directory to the staging area

(pending for commit)

02

git commit

Takes the ***changes*** in the staging area and ***"version"***

(the version is still sitting on your computer only)

03

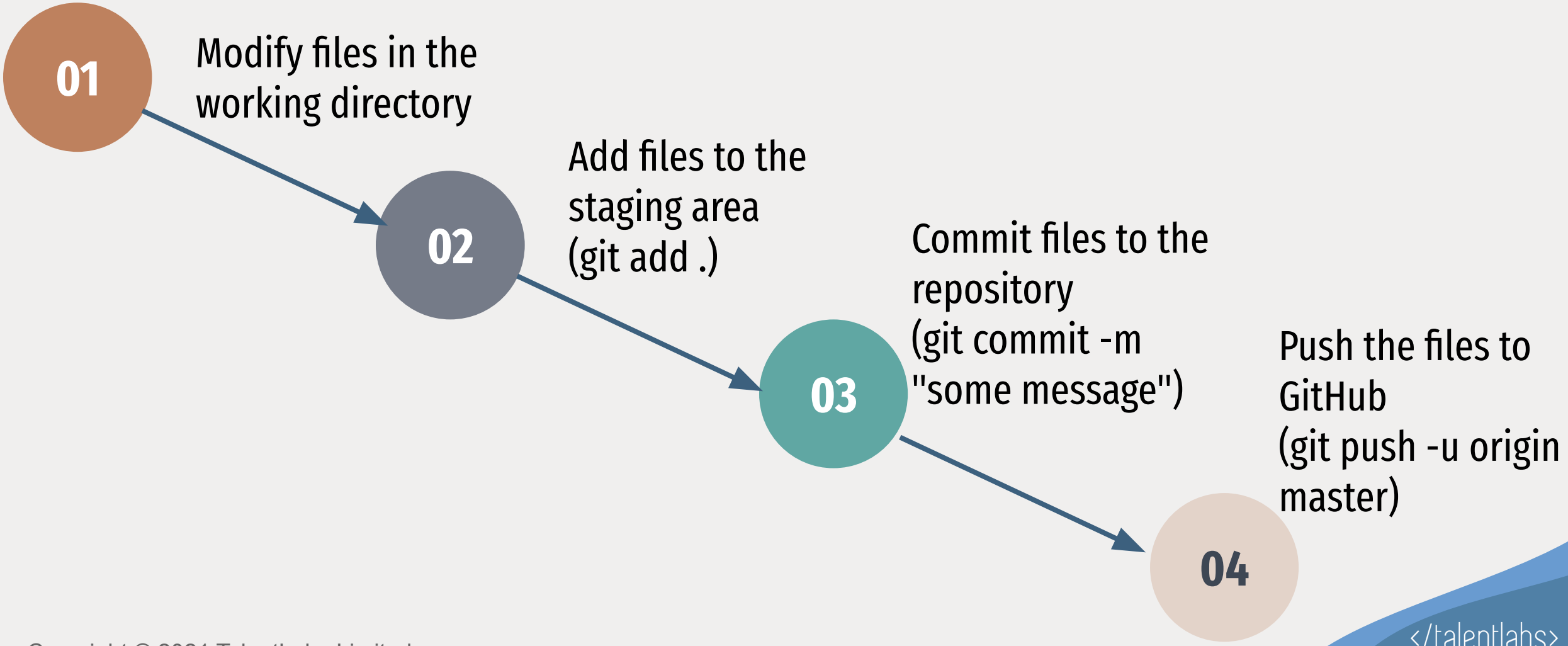
git push

Pushes all versions to the remote repository

(push the versions to GitHub)

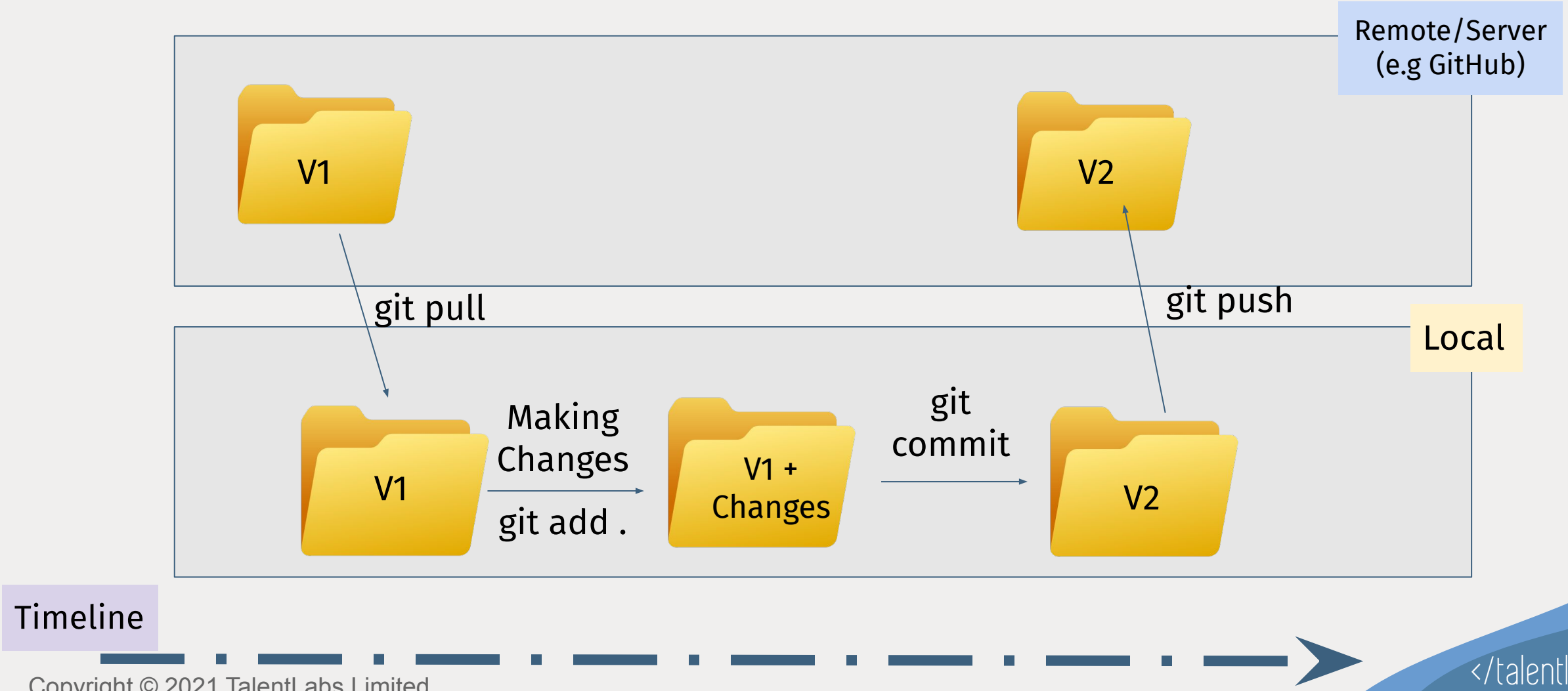
</talentlabs>

Step-by-step Workflow



</talentlabs>

Git Process Summary



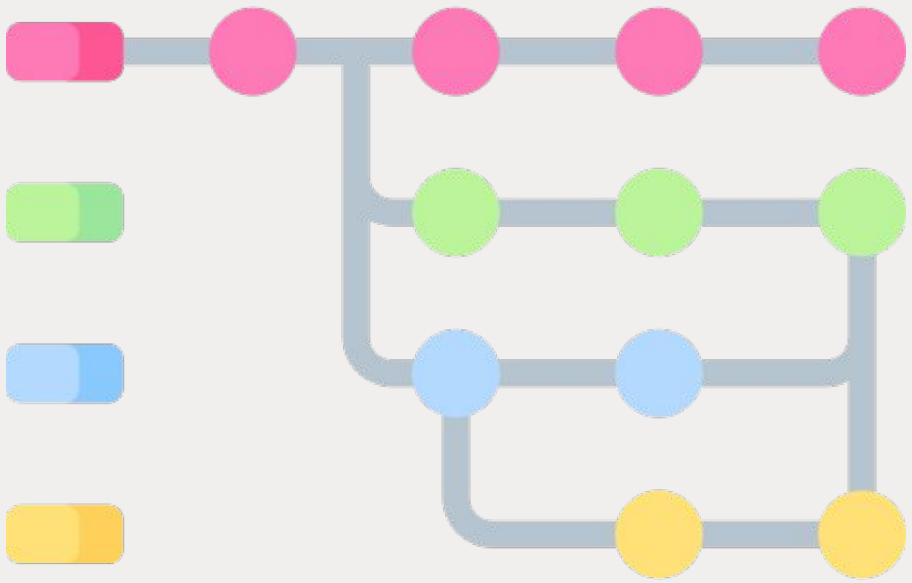
</talentlabs>

Working with Branches

Why we want to use branches?

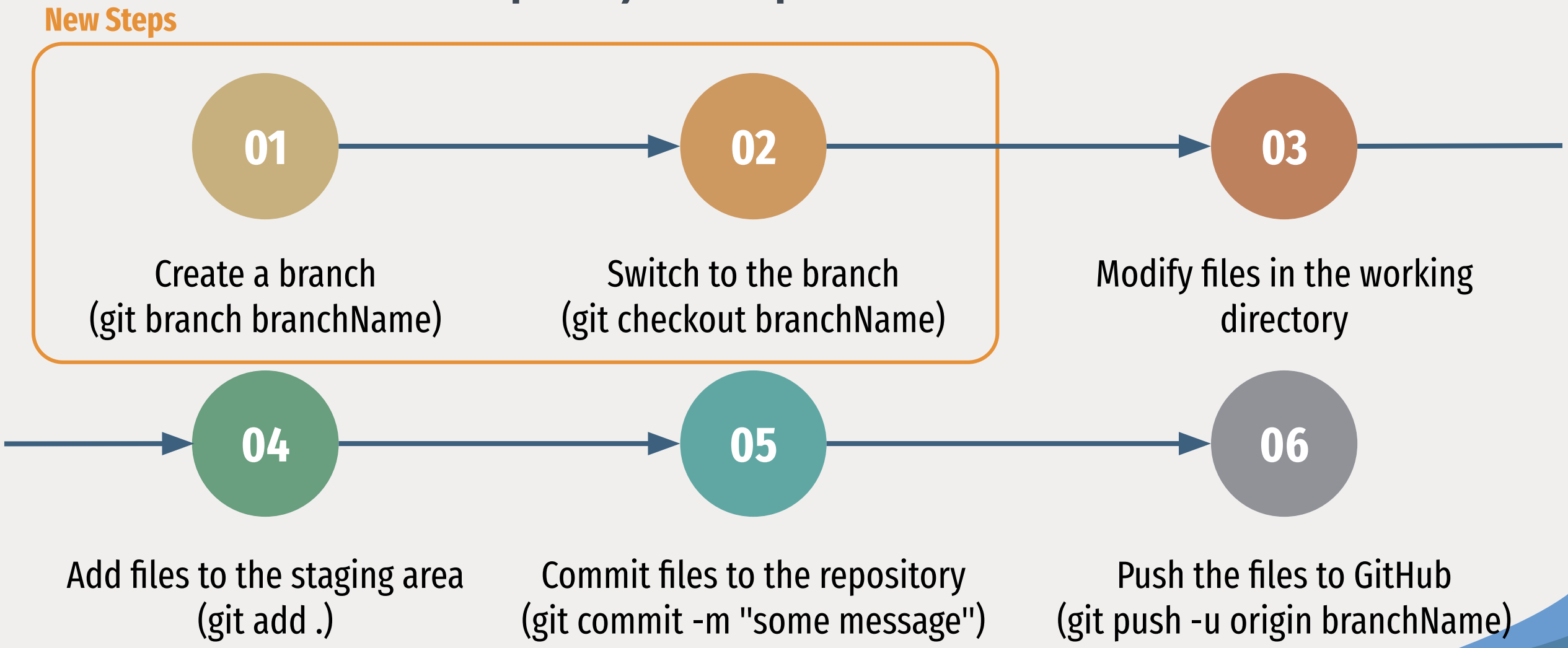
01 When we are making changes, we don't want to impact other teammates on main/master branch.

02 Easily discard your changes if your changes doesn't work. Just delete the branch and forget about it.



</talentlabs>

Step-by-step Workflow



</talentlabs>

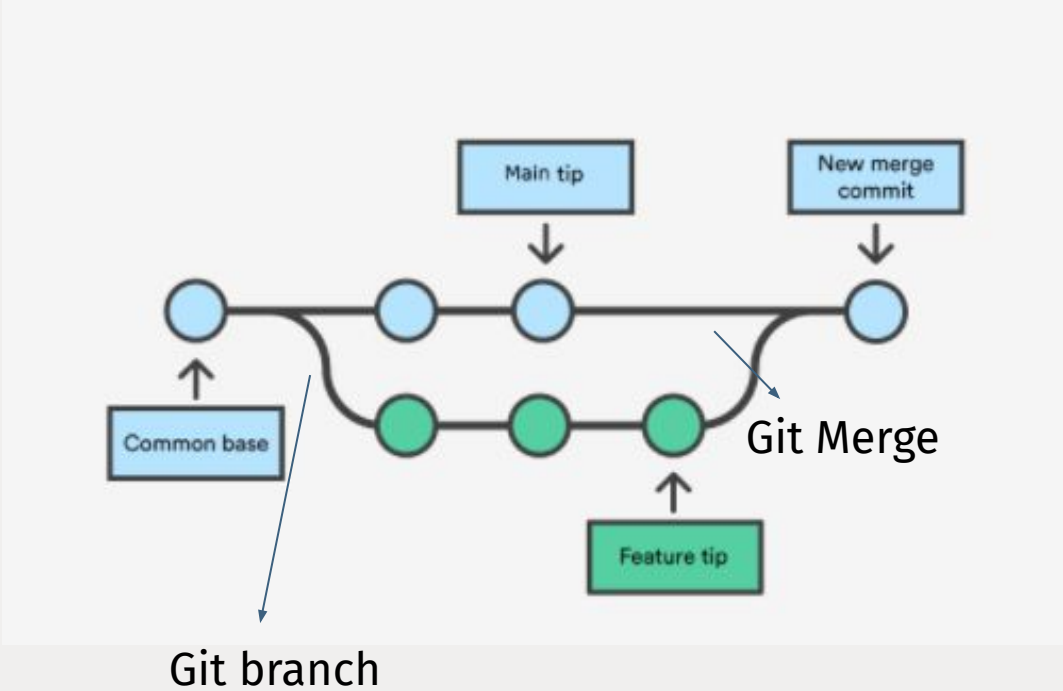
What’s next after completing your changes in branches?

Merge it Back to master/main!

1. **Open a pull request:** Share the changes with your team members in the branch so they can see your work

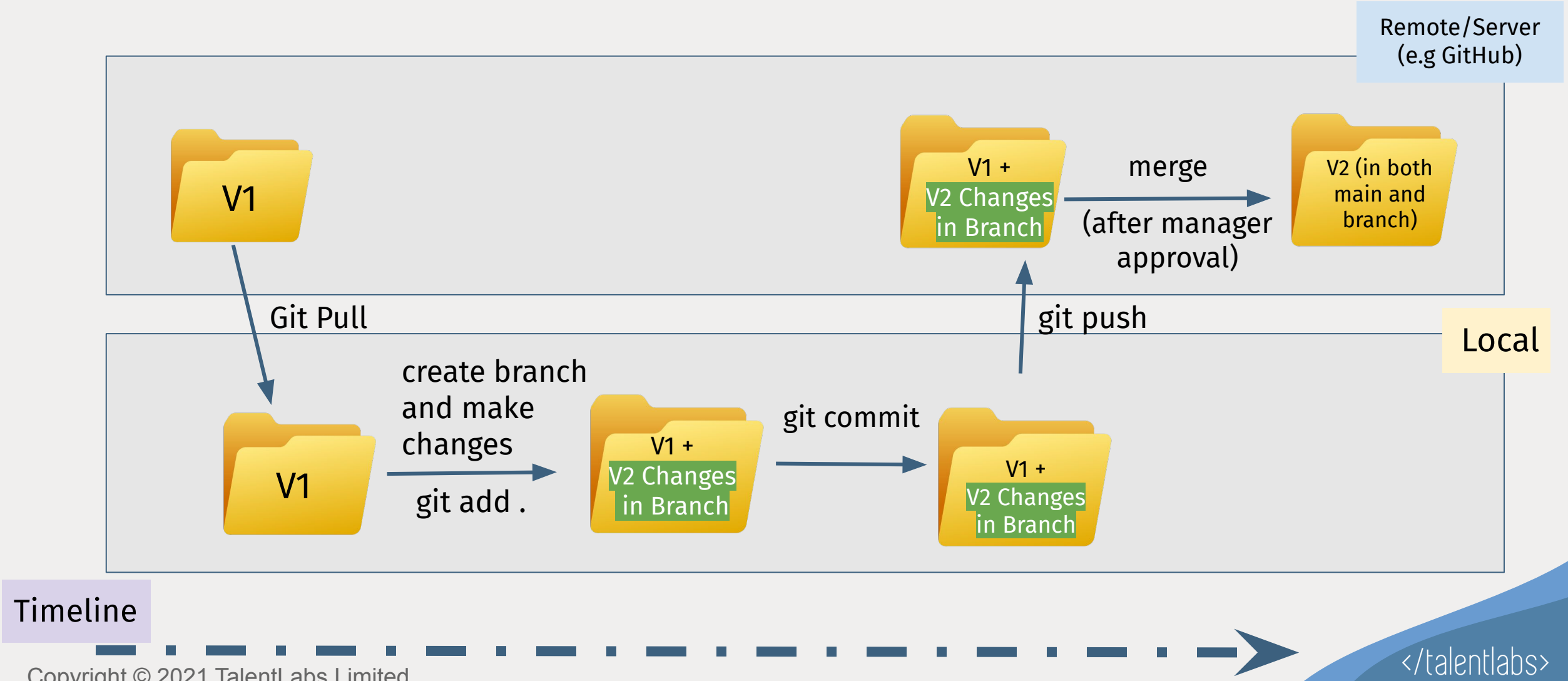
2. **Discuss and review code:** project team or manager to review your code and give comments.

3. **Merge:** If everything is ok, then you can click the merge button on GitHub, to merge the changes from branch to main.



</talentlabs>

Git Process Summary (Team Version)



</talentlabs>