GIT AND GITHUB (LEC)

Presented by
Christian Nicole Delfin

OVERVIEW FOR FINALS

1. Git and Github
2. Git Bash / Visual Studio Code
3. Extension of Github account
4. Benefits of having a GitHub Pro Account
5. GitHub Foundation Certification
6. MongoDB Certification - PHP Developer (optional)
7. CI/CD Pipeline

WHAT IS GIT?

Distributer version control system (VCS) that helps their codebases, collaborate with others and manage multiple version of a project

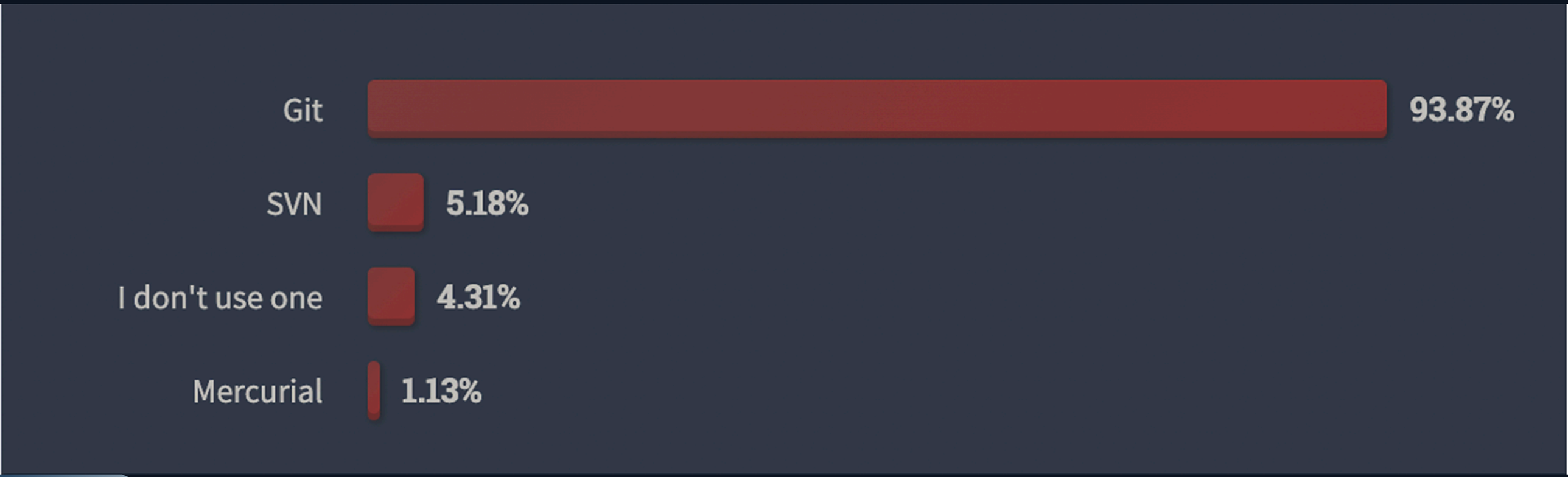
Git is decentralized, which means you don’t need a central server. instead every developer has a full copy of the repo on their machine.

KEY FEATURES OF GIT

- Distributed system
- Version Tracking
- Collaboration
- Branching
- Merging
- Remote Repositories
- Extensive tooling
- Staging Area
- Speed
- Open-Sournce & Free

GIT'S POPULARITY

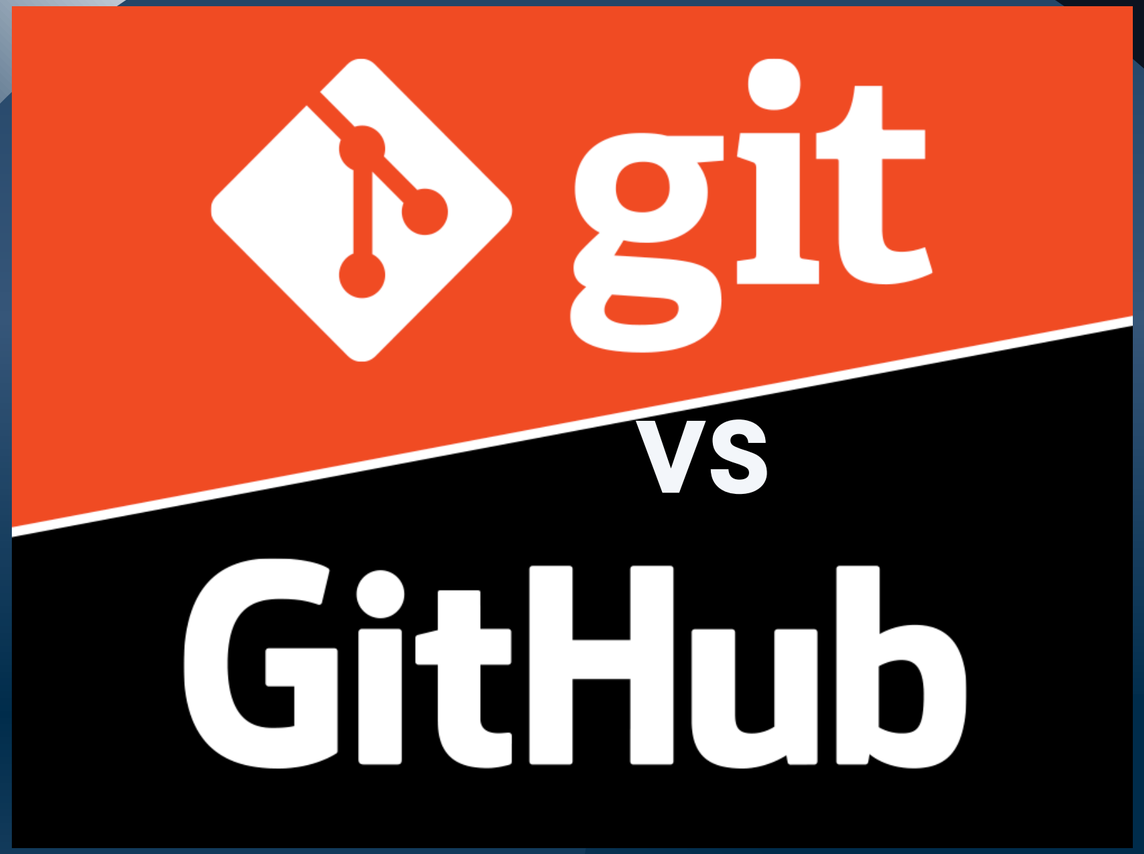
The most commonly used version control technology is Git. Git is a version control system that enables developers to track software projects and work on them together. (Stack overflow developer survey as of 2022)



WHAT IS GITHUB?

Web-based platform used for version control and collaboration. It hosts repositories and provides a interface to manage your code and do many other things.

Github offers collaboration features like bug tracking, feature requests, task management and wikis.

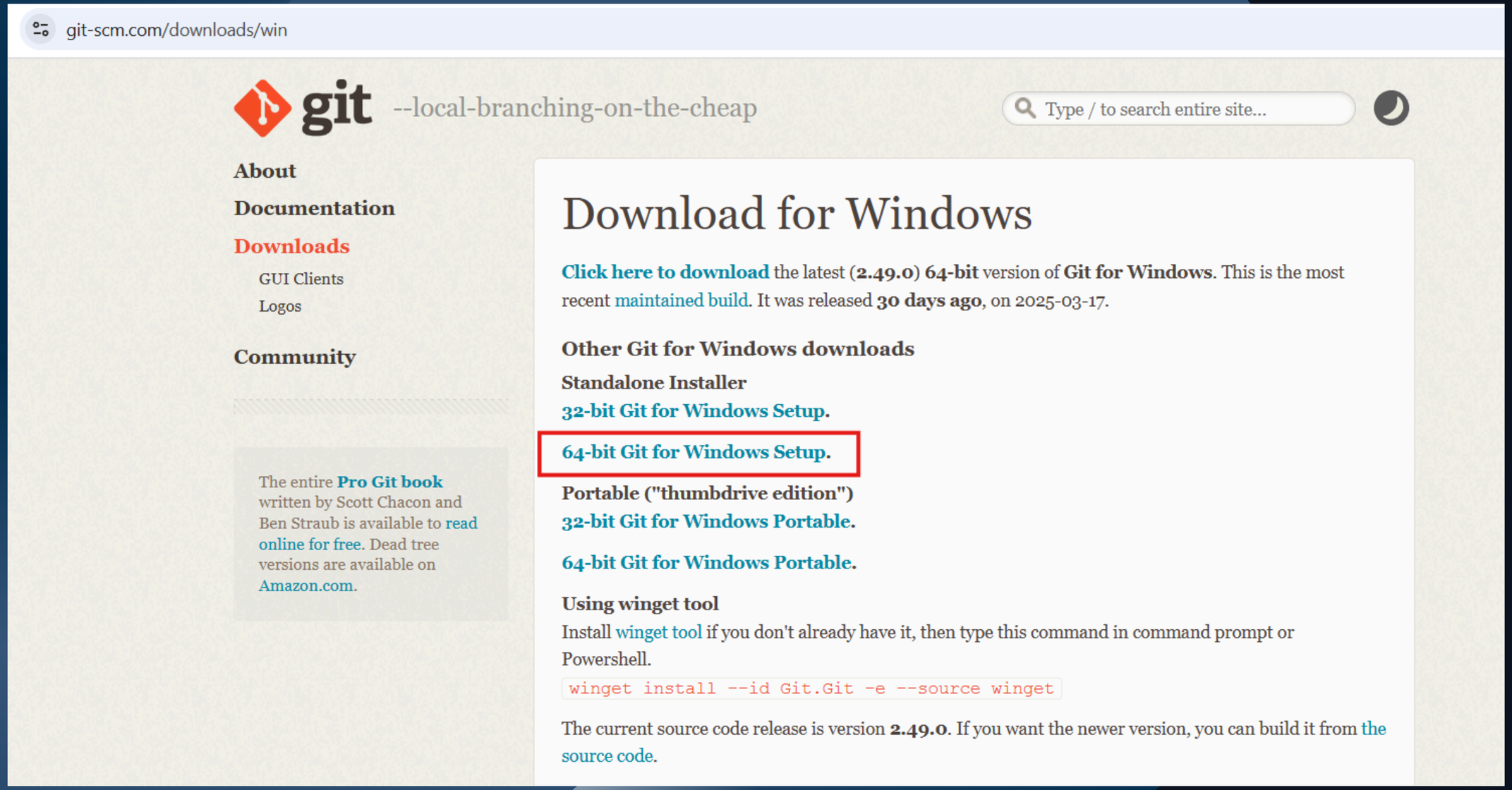


Git is the VCS (version control system) and Github hosts repositories. there are many other platform that can host git repos, such as gitlab & bitbucket.

INSTALLING GIT

https://git-scm.com/downloads/win

INSTALLING GIT



INSTALLING GIT

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\xtian> git
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--no-lazy-fetch]
           [--no-optional-locks] [--no-advice] [--bare] [--git-dir=<path>]
           [--work-tree=<path>] [--namespace=<name>] [--config-env=<name>=<envvar>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone      Clone a repository into a new directory
   init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add        Add file contents to the index
   mv         Move or rename a file, a directory, or a symlink
   restore    Restore working tree files
   rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect     Use binary search to find the commit that introduced a bug
   diff       Show changes between commits, commit and working tree, etc
   grep       Print lines matching a pattern
   log        Show commit logs
```

TERMINAL VS GUI

TERMINAL VS GUI

You can either use Git by typing commands in the terminal or you can use a graphical user interface (GUI) such as Sourcetree, GitKraken or Visual studio code .

configuring git

The first thing you should do when you install Git is to set your user name and email address. This is important because every Git commit uses this information, and it’s immutably baked into the commits you start creating.

CONFIGURING GIT

```
xtiannicole215@Christian MINGW64 ~
$ git config --global user.name "Christian Nicole Delfin"

xtiannicole215@Christian MINGW64 ~
$ git config --global user.email "christiannicole.delfin@cvsu.edu.ph"

xtiannicole215@Christian MINGW64 ~
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=schannel
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.name=Christian Nicole Delfin
user.email=christiannicole.delfin@cvsu.edu.ph

xtiannicole215@Christian MINGW64 ~
$
```

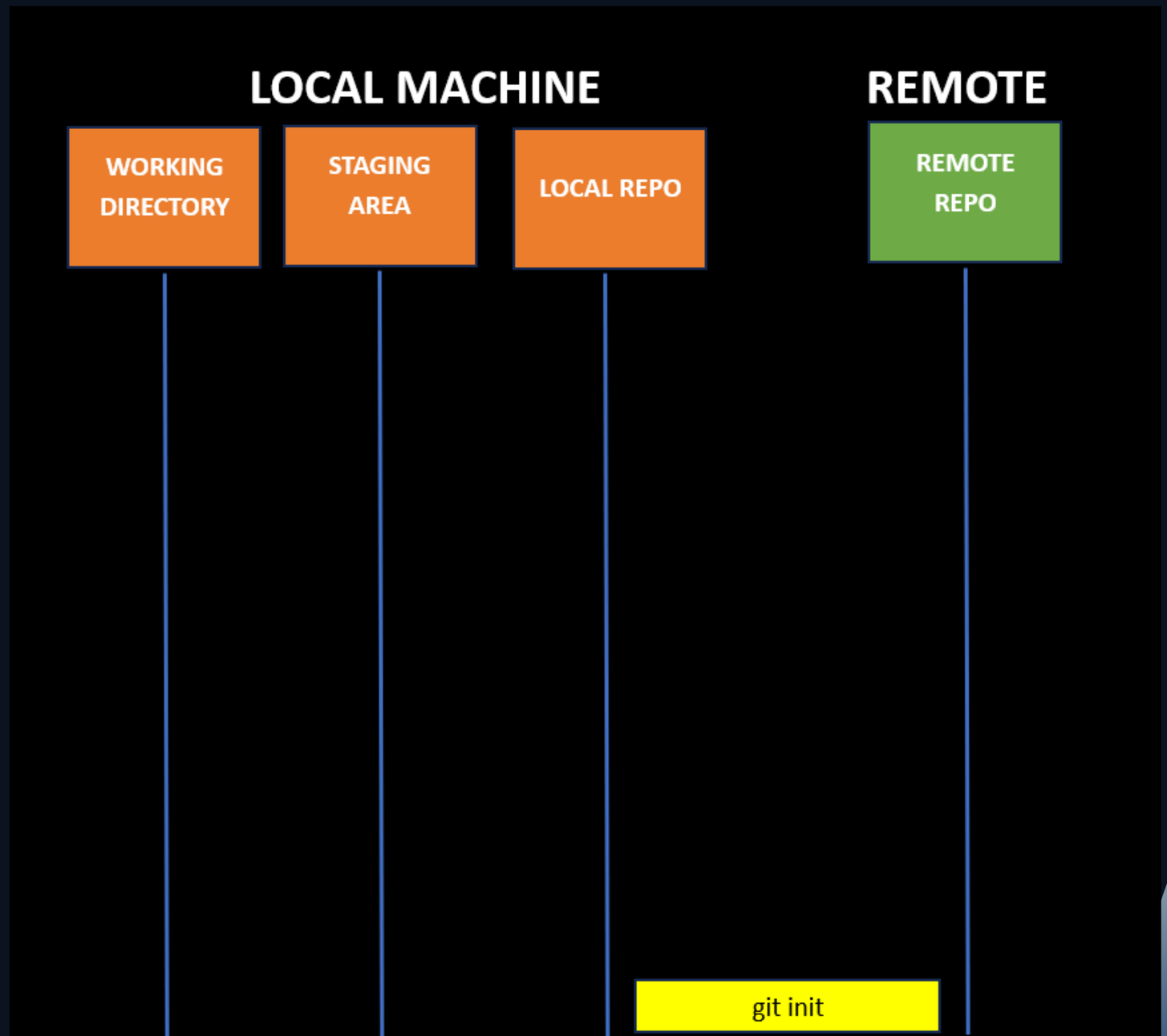
GIT WORKFLOW

Given Git's focus on flexibility, there is no standardized process on how to interact with Git. When working with a team on a Git-managed project, it’s important to make sure the team is all in agreement on how the flow of changes will be applied. To ensure the team is on the same page, an agreed-upon Git workflow should be developed or selected. There are several publicized Git workflows that may be a good fit for your team. Here, we will discuss some of these Git workflow options.

GIT WORKFLOW

git init - initialize a new repository in your project folder (working directory)

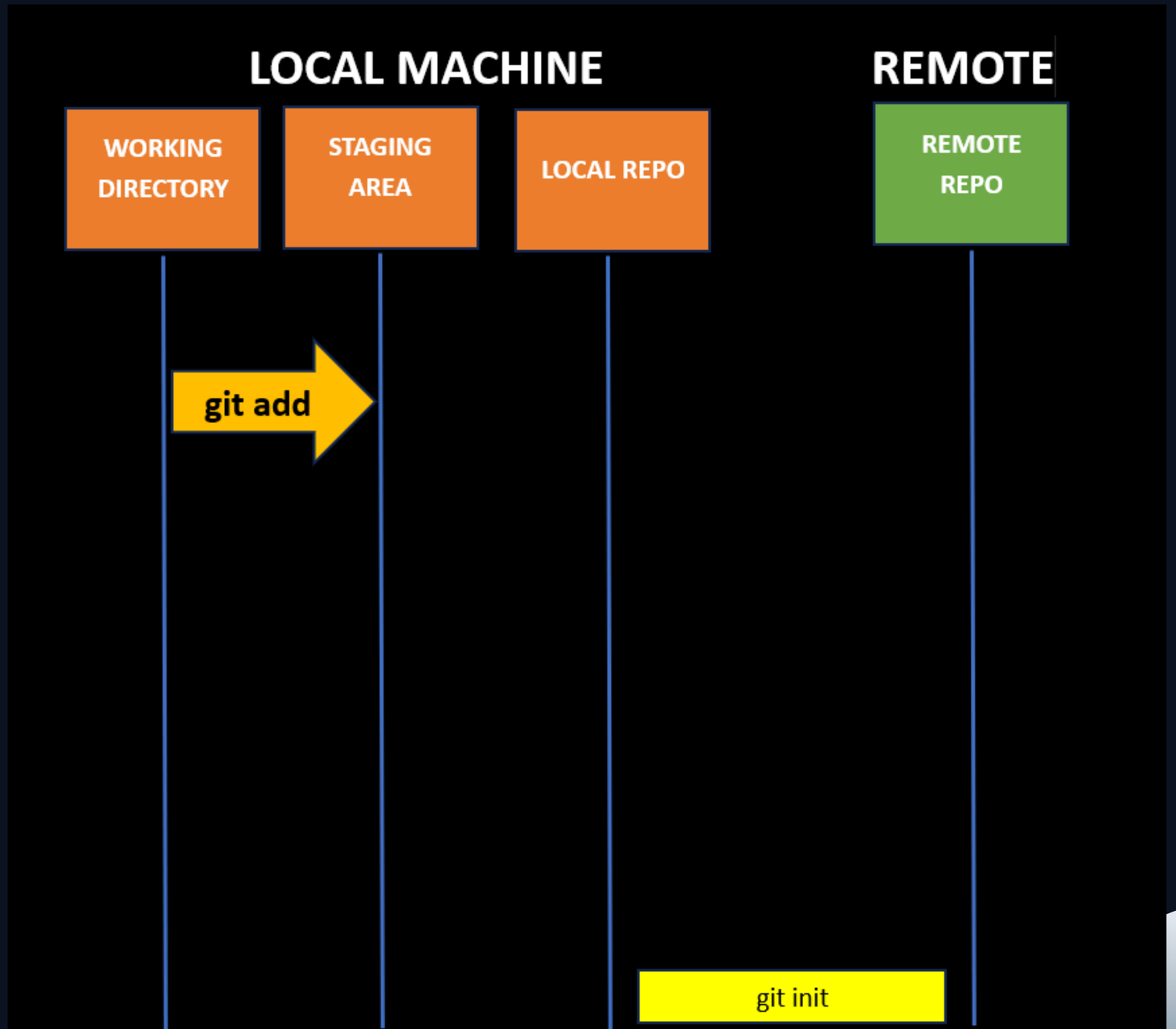
This creates a new hidden folder called *".git"*



GIT WORKFLOW

git add - adds your file in staging area.

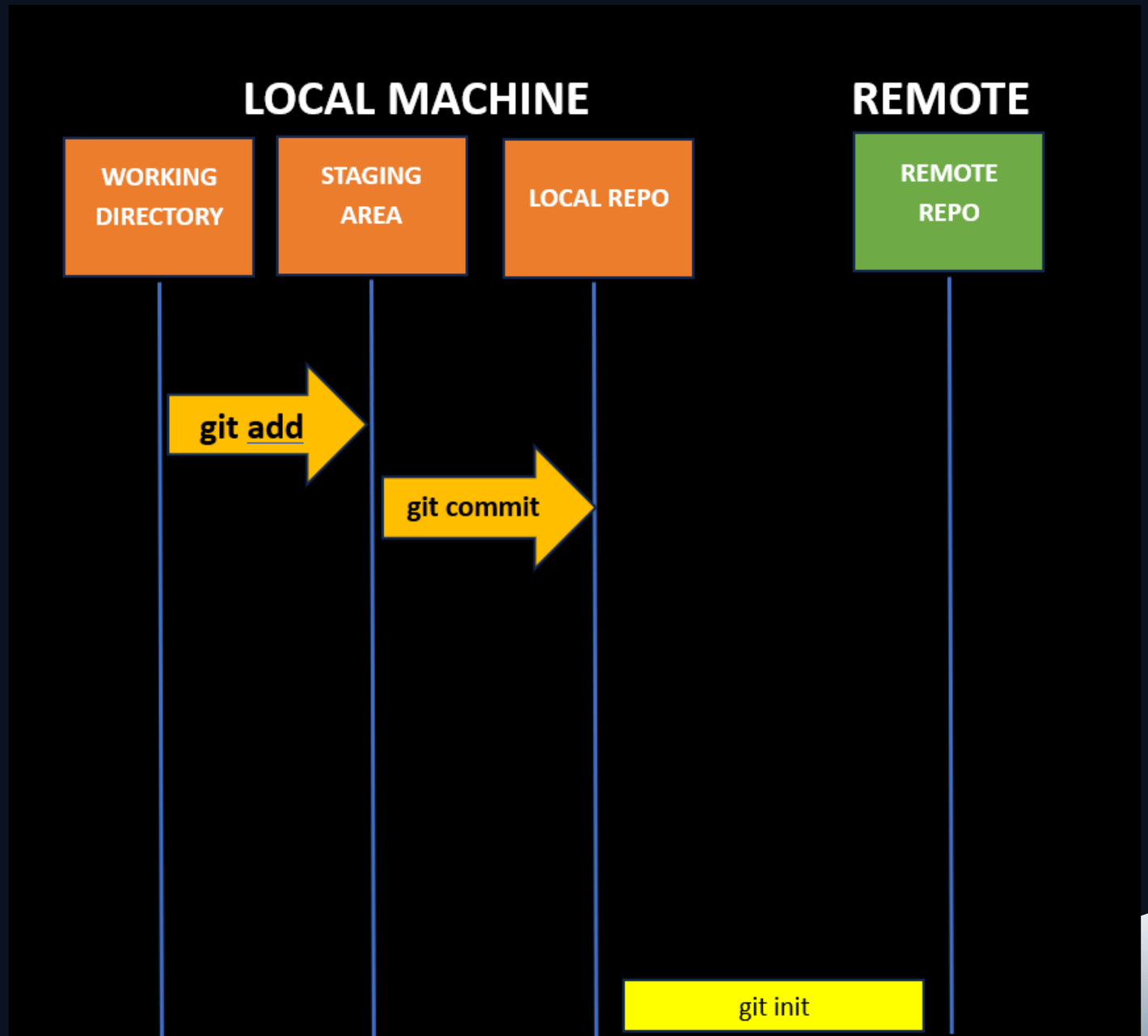
You specify the files or use a period “.” for all files.



GIT WORKFLOW

git commit - commits your file to your local repository on your machine.

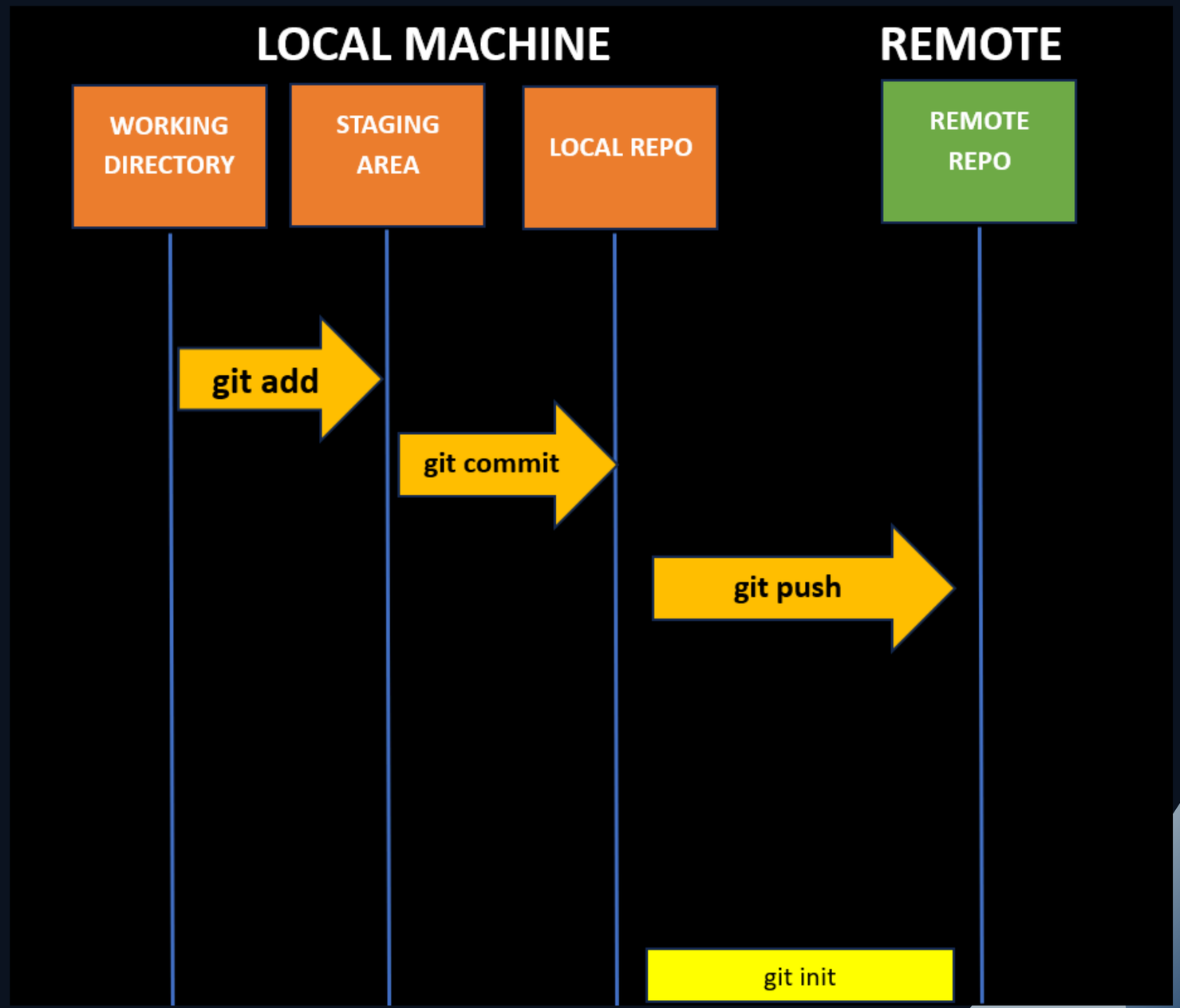
You can commit as often as you would like.



GIT WORKFLOW

git push- push to your remote repository.

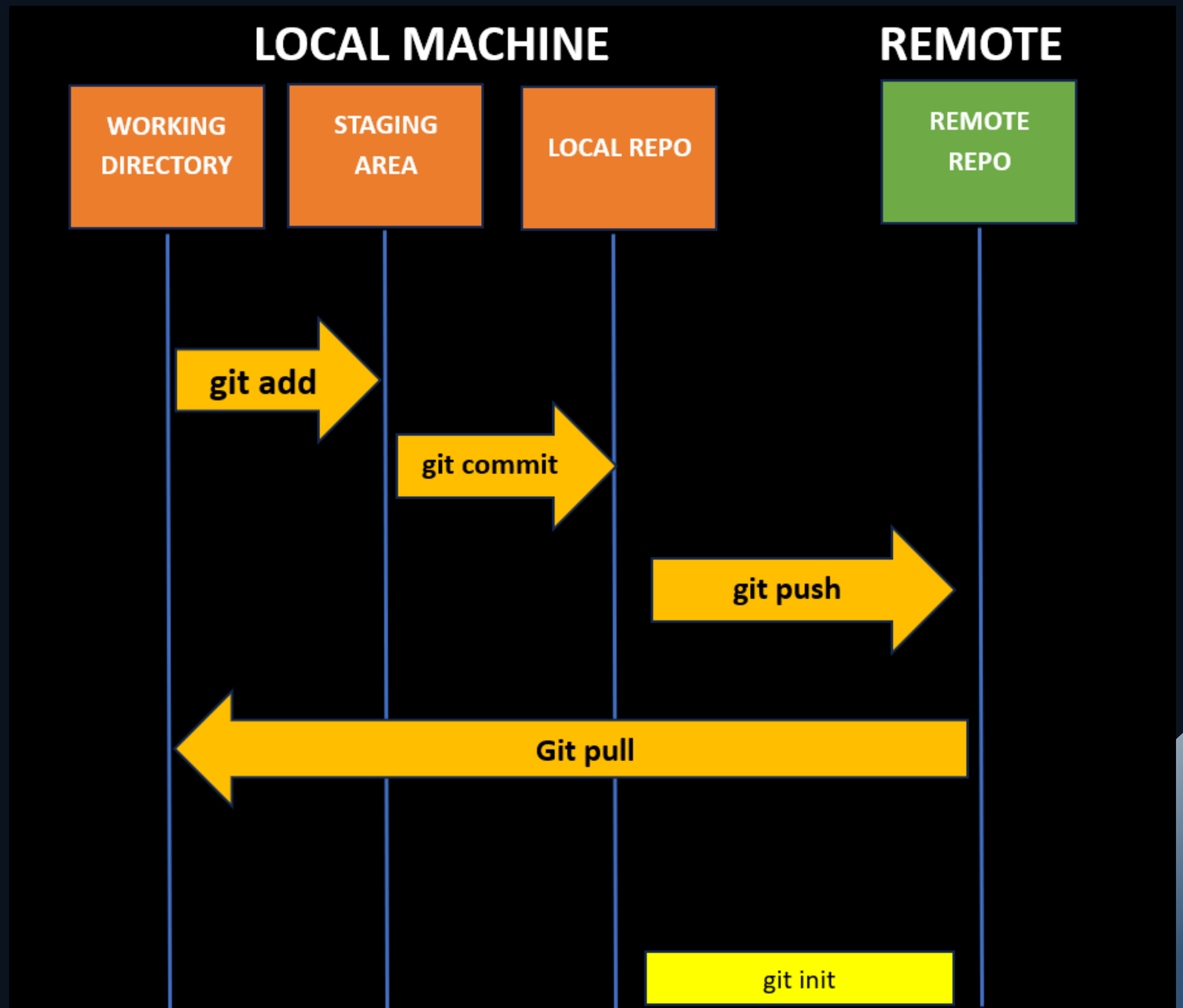
This could be Github, Gitlab or Bitbucket.



GIT WORKFLOW

git pull- pull changes from the remote repo to your local machine.

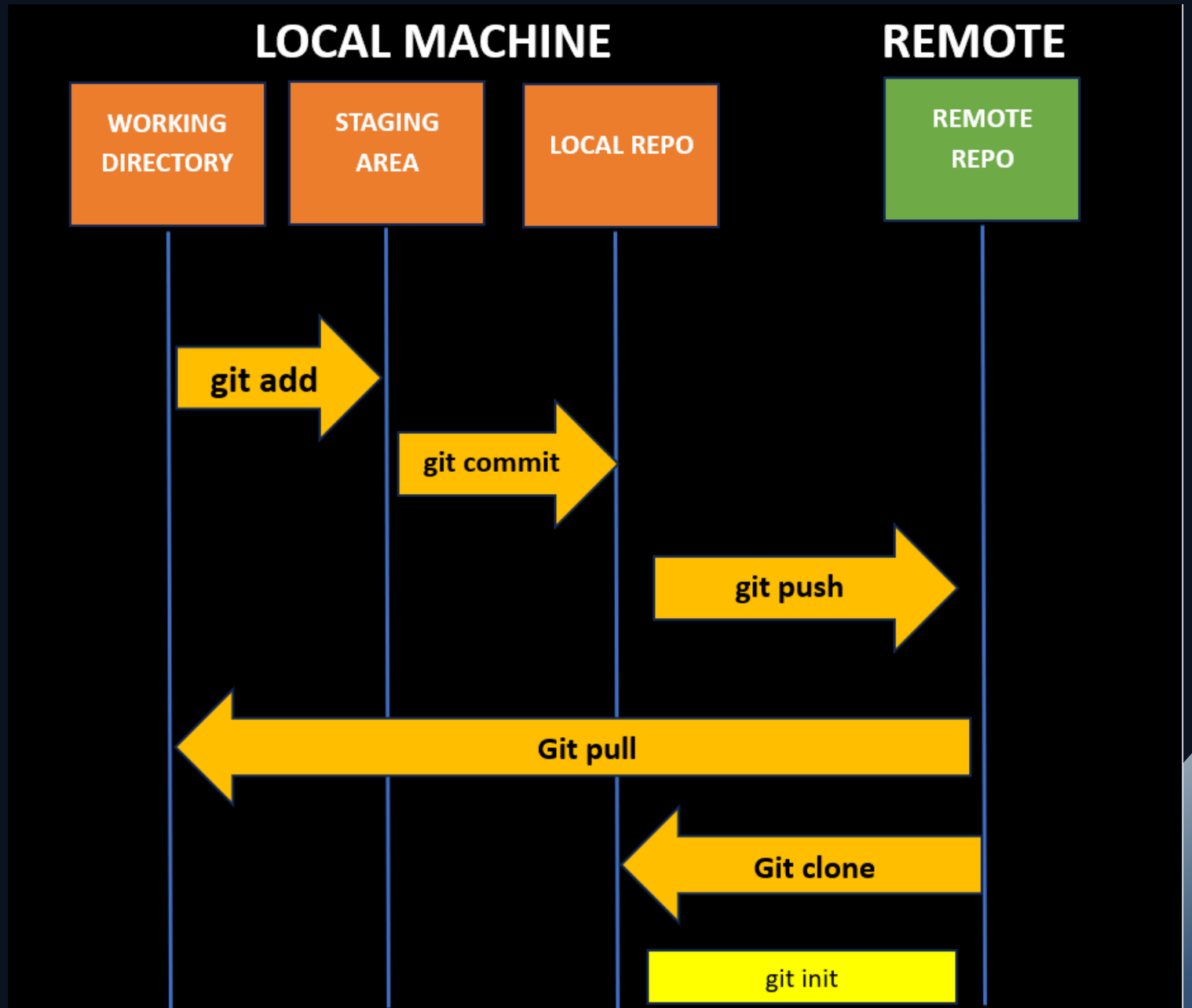
Other user may make changes. You pull their changes to your machine.



GIT WORKFLOW

git clone- clone an entire remote repository to your local machine.

You would do this when you want to start working on an existing project.



THE WORKFLOW

Commit

Once you've saved your files, you need to commit them - this means the changes you have made to files in your repo will be saved as a version of the repo, and your changes are now ready to go up on GitHub (the online copy of the repository).

Pull

Now, before you send your changes to Github, you need to pull, i.e. make sure you are completely up to date with the latest version of the online version of the files - other people could have been working on them even if you haven't. You should always pull before you start editing and before you push.

Push

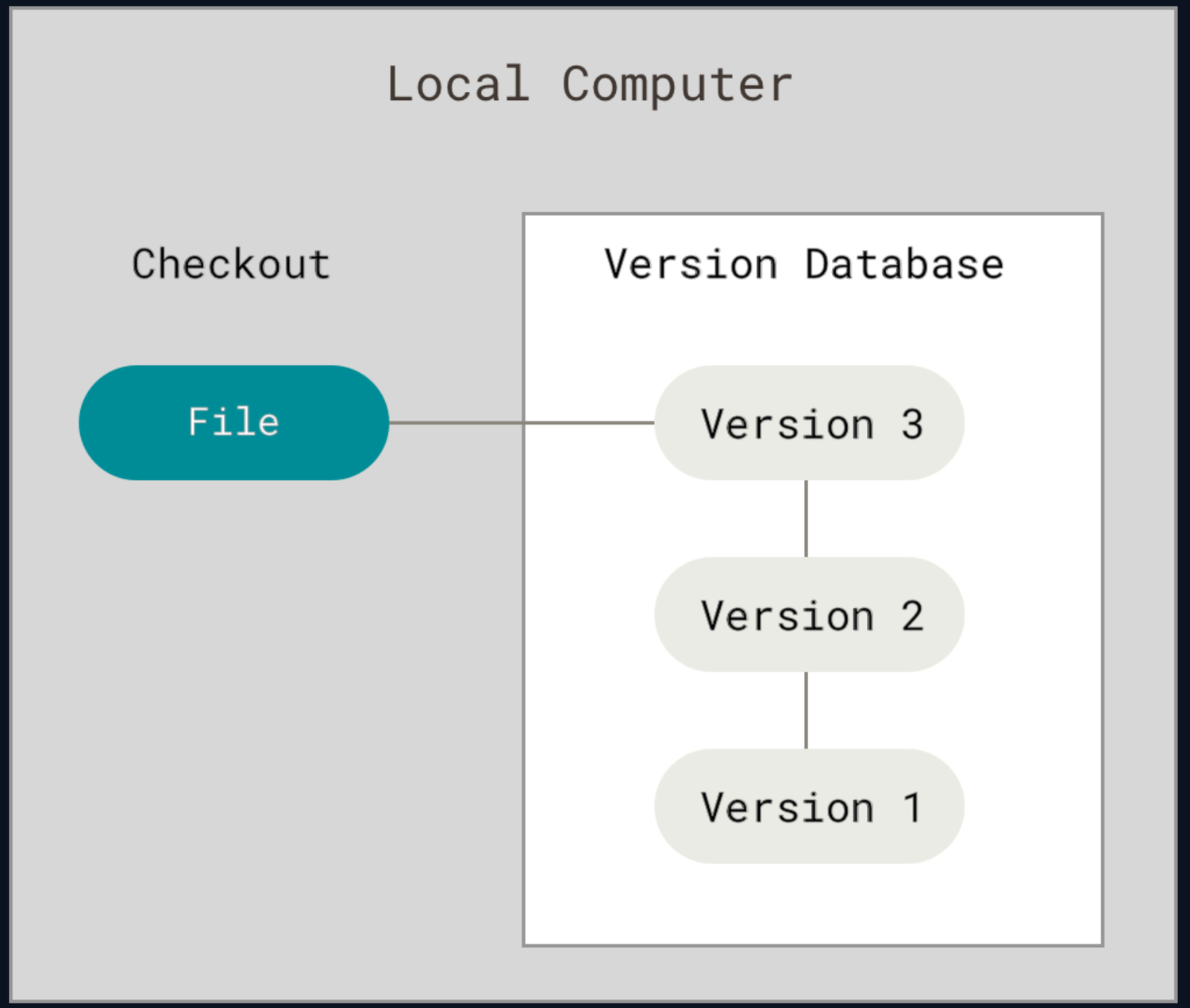
Once you are up to date, you can push your changes - at this point in time your local copy and the online copy of the files will be the same.

Each file on GitHub has a history, so instead of having many files, you can have only on
exploring its history, you can see what it looked at different points in time.

VERSION CONTROL

VERSION CONTROL

Version control allows you to keep track of your work and helps you to easily explore the changes you have made, be it data, coding scripts, notes, etc. You are probably already doing some type of version control, if you save multiple files, such as Dissertation_script_25thFeb.R, Dissertation_script_26thFeb.R, etc. This approach will leave you with tens or hundreds of similar files, making it rather cumbersome to directly compare different versions, and is not easy to share among collaborators. With version control software such as Git, version control is much smoother and easier to implement.


Local Computer
Checkout
File
Version Database
Version 3
Version 2
Version 1

BENEFITS OF VERSION CONTROL

Each file on GitHub has a history, making it easy to explore the changes that occurred to it at different time points. You can review other people's code, add comments to certain lines or the overall document, and suggest changes. For collaborative projects, GitHub allows you to assign tasks to different users, making it clear who is responsible for which part of the analysis. You can also ask certain users to review your code. For personal projects, version control allows you to keep track of your work and easily navigate among the many versions of the files you create, whilst also maintaining an online backup.

WHAT IS A REPOSITORY?

Repository is a folder containing code where any changes to the code are tracked by git. (To create a repository, we create a new folder on our computer, and then run git init)

GIT COMMANDS (LAB)

Presented by
Christian Nicole Delfin

CONFIGURE NAME & EMAIL FOR COMMITS

```
git config --global user.name "Your Name"
git config --global user.email "email@example.com"
```

CREATING COMMITS

IN GIT, VERSION = COMMIT

VERSION HISTORY = COMMIT HISTORY

git init	Git will start tracking all changes in the current folder
git status	Show all changes since the previous commit
git add <file>	Pick individual file
git add folder/	Pick all files inside a folder (and subfolders)
git add .	Pick all files (in folder command line is running in)
git commit -m "message"	Creates a commit with a message attached
git commit -m "message" - -amend one	Update previous commit instead of creating new

CREATING COMMITS

IN GIT, VERSION = COMMIT VERSION HISTORY = COMMIT HISTORY

git log	View the commit history
git log - -all	Show all commits (not just current branch)
git log - -all - -graph	Show branching visually in the command line

STAGING

IN GIT, VERSION = COMMIT VERSION HISTORY = COMMIT HISTORY

git add .	working > staging
git commit -m "message"	staging > commit history
git reset <file/folder>	
git reset file	
git reset folder/	
git reset .	staging > working
git checkout - - file git checkout -- folder/	working > remove the changes
git checkout - -	
git checkout - - .	

THANK YOU