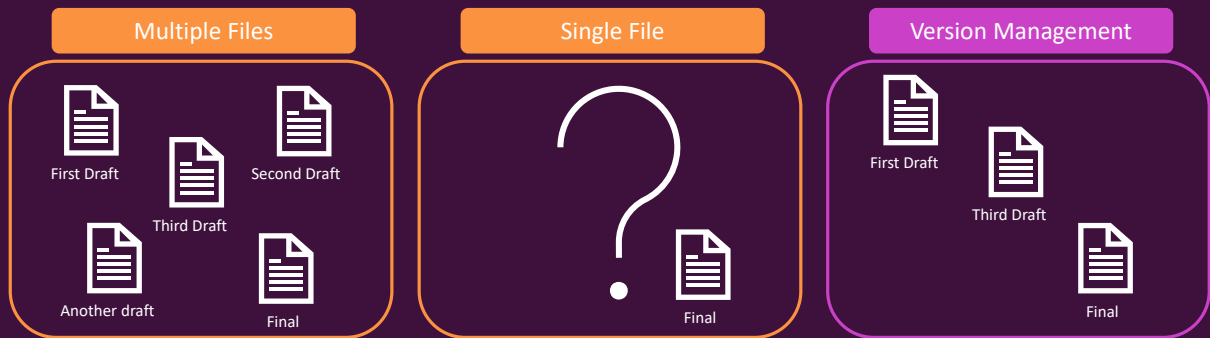


Getting Started

What & Why?

What are Git
& GitHub?

What is Version Management/Control?



Control & tracking
of code changes*
over time

About Git & GitHub



Version Control System

Manage Code History

Track Changes



Largest Development Platform

Cloud Hosting &
Collaboration Provider

Git Repository Hosting

Course Content

Windows
Command Prompt

Commits & Branches
The Basics

Local vs
Remote Repositories
The Basics

Mac Terminal
(Z shell)

Merging,
Rebasing & More
Deep Dive

Pull Requests,
Organizations & More
Deep Dive

Optional

Git

GitHub

Practice Project

How To Get The Most Out Of The Course



Watch the Videos
(choose your pace)



Code Along & Practice
(also without us telling you)



Debug Errors & Explore Solutions
(also check attachments)



Help Each Other & Learn Together
(Discord, Q&A Board)

Optional: Command Line Basics

Text Based Computer Interaction

Module Content

Text Based Computer Interaction – What & Why?

Mac User: Z-Shell Basics

Windows User: Command Prompt Basics

Text Based Computer Interaction – What & Why

Graphical User Interface (GUI)

Name	Date modified	Type
ESD	12/17/2018 1:10 PM	File folder
PerfLogs	5/13/2020 11:05 PM	File folder
Program Files	7/4/2020 10:08 AM	File folder
Program Files (x86)	9/2/2020 9:58 AM	File folder
ProgramData	4/3/2020 11:53 AM	File folder
Users	7/29/2019 3:24 PM	File folder
Windows	8/14/2020 6:04 PM	File folder

User Friendly

Easy to Explore

Command Prompt

```

Microsoft Windows [Version 10.0.18362.1016]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Manuel Lorenz>cd ..

C:\Users>cd ..

C:\>
  
```

Time Saving

More Possibilities

Start Servers

Download + Install Tools

Run Code

Execute Files

Working with Git

Mac Terminology



Text Input Environment ("Hardware")



Text Input Interface ("Software")

Bash

zsh (Z-Shell)



Windows Terminology



Command Prompt (cmd)

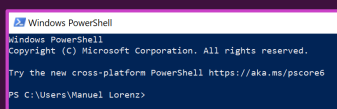
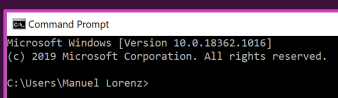
PowerShell

Git Bash

Initial Windows Shell

New Shell (Windows 7 Release)

Bash Emulation for Windows



Command Line Tools



Command Prompt
(cmd)

```

Command Prompt
Microsoft Windows [Version 10.0.18362.1016]
(c) 2019 Microsoft Corporation. All rights reserved.
C:\Users\Manuel Lorenz>
    
```



Terminal
(z-Shell)

```

LorenzM@Manuel1s-MBP ~ %
    
```

Mac: Terminal (z-Shell) – Useful Commands

pwd
(print working directory)

ls
(list items)

cd (..)
(change directory)

cd /
(root directory)

cd /Users
(users directory)

cd or cd ~
(home directory)

touch
(create/ "touch" file)

mkdir
(create directory)

rm
(delete file)

rmdir
(delete empty folder)

cp
(copy file/folder)

mv
(move file/folder)

Windows: Command Prompt – Useful Commands

cd
(print current path)

dir
(list items)

cls
(clear command prompt)

cd (..)
(change directory)

echo text > name(.type)
(create file)

mkdir foldername
(create directory)

del file
(delete file)

rmdir folder
(delete folder)

copy file
(copy file)

move file folder
(move file or folder)

Version Management with Git

The Basics

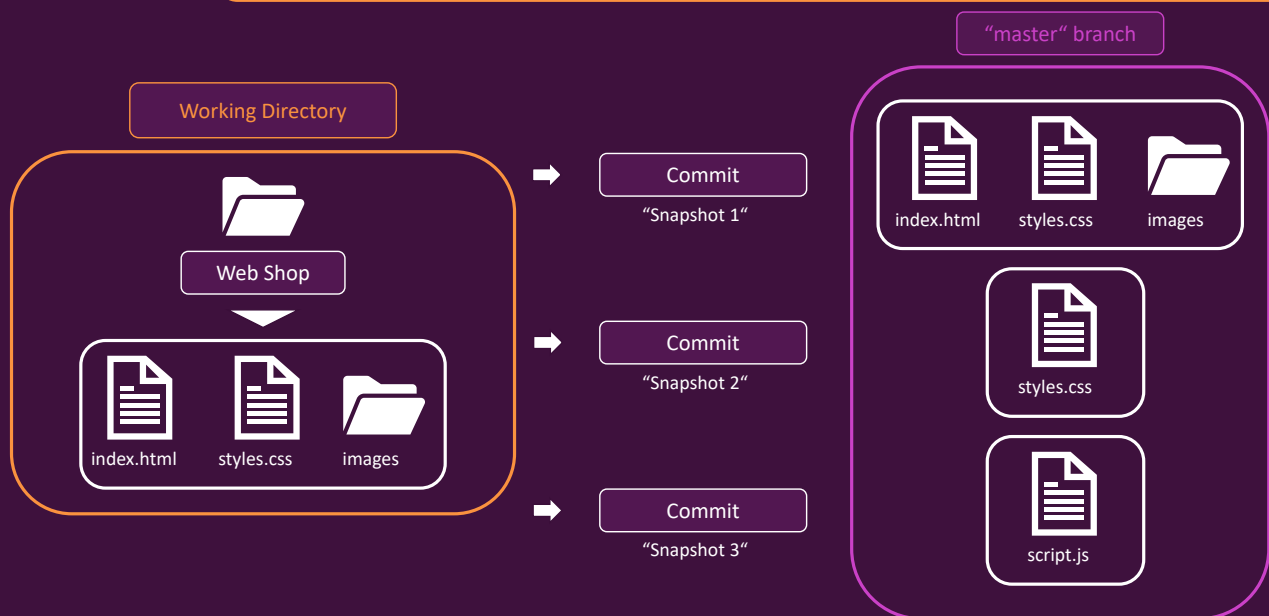
Module Content

Theory – How Git Works

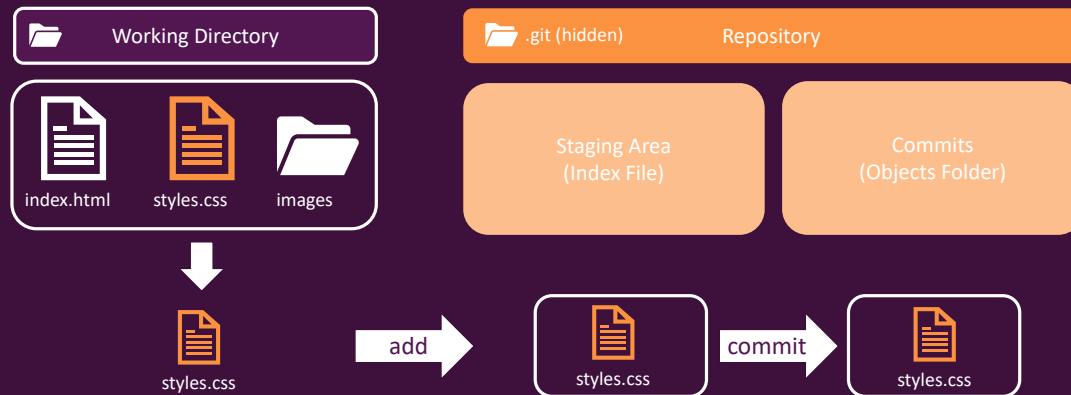
Installation & Development Environment

Repositories, Branches & Commits

How Does Git Work – Simplified!

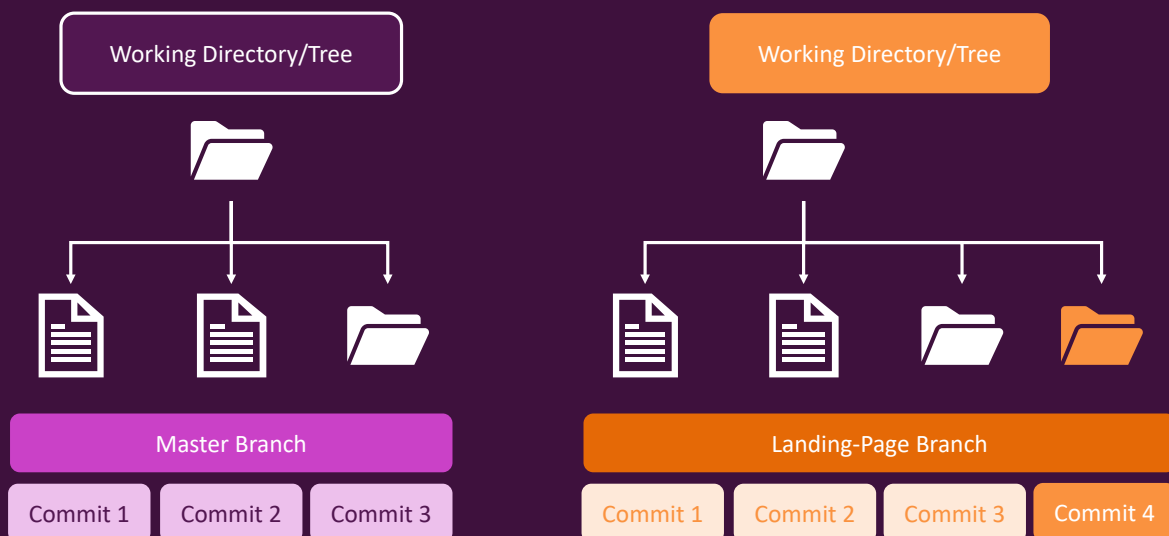


Git under the Hood

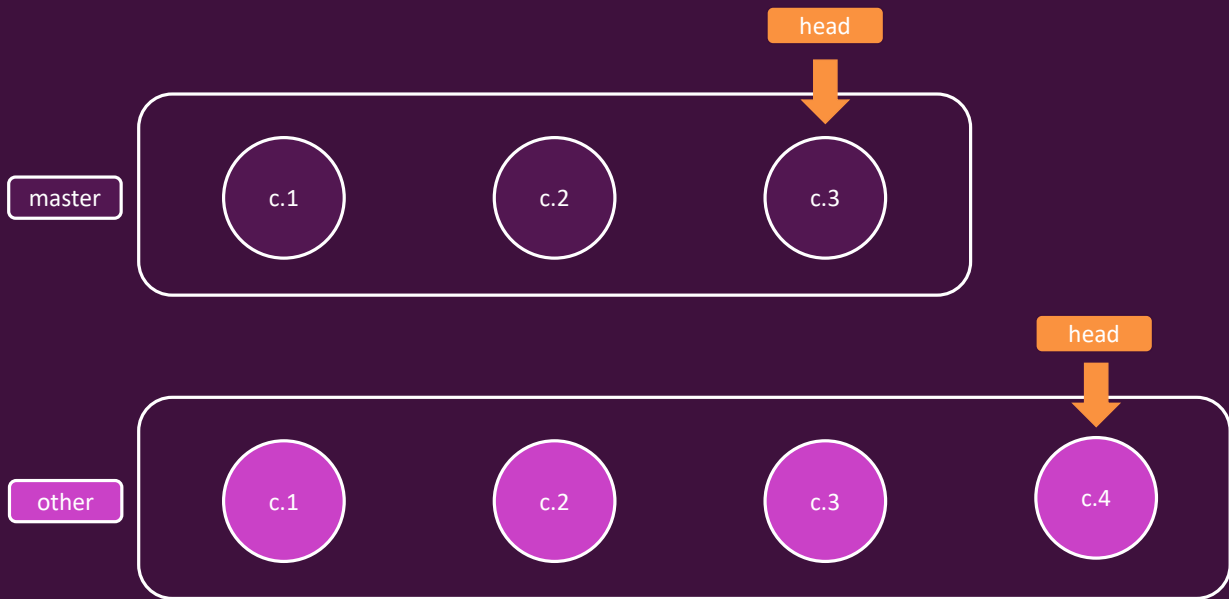


Git = Tracking changes - NOT storing files again and again!

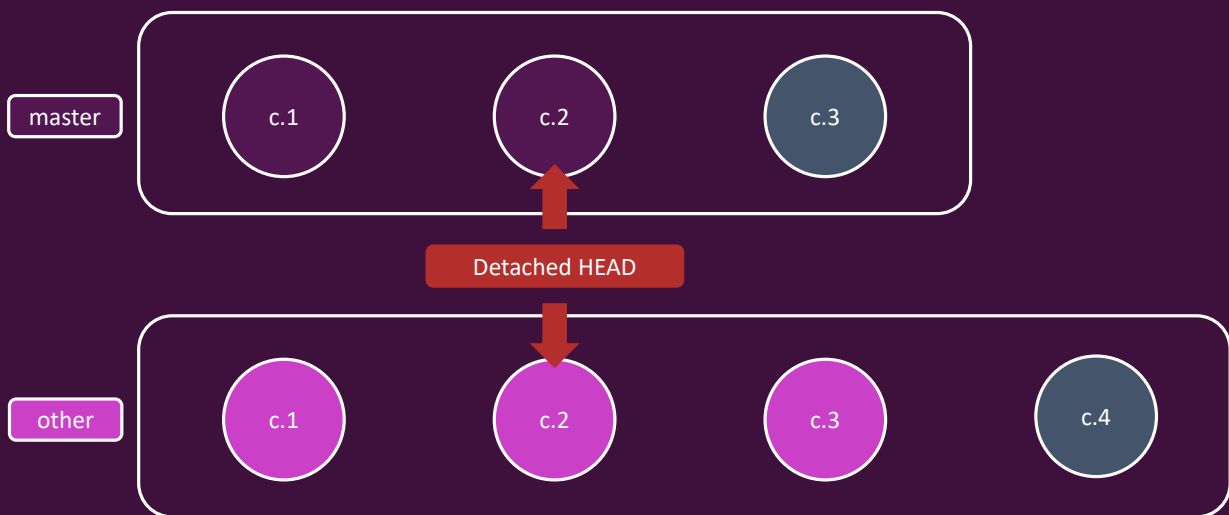
Branches & Commits



What is the "HEAD"?



The "detached HEAD"



"HEAD" vs "detached HEAD"

HEAD

Indirectly points to commit

`git checkout branchname`

Points to branch which points to commit

Detached HEAD

Directly points to commit

`git checkout commitid`

Points to commit with specified ID

Commit is **not related** to specific branch

Deleting Data

Working Directory Files
(already part of previous commit)

Unstaged Changes

Staged Changes

Latest Commit(s)

Branches

Basic Commands Summary: General

```
git --version
```

Check installed Git version

```
git init
```

Create empty Git repository

```
git status
```

Check working directory &
staging area status

```
git log
```

Display all commits of current
branch

```
git ls-files
```

List tracked files

Basic Commands Summary: Commit Creation & Access

```
git add filename  
git add .
```

Add single file or all WD files to
staging area

```
git commit -m  
"message"
```

Create new commit

```
git checkout commitid
```

Checkout commit (detached
head!)

Basic Commands Summary: Branch Creation & Access

Git 2.23+

`git branch branchname`

`git switch branchname`

Create new branch

`git checkout
branchname`

Go to branch

`git checkout -b
branchname`

`git switch -c
branchname`

Create and access new branch

`git merge otherbranch`

Bring other branch's changes to
current branch

Basic Commands Summary: Deleting Data

Delete/Undo

WD File*

`git rm filename`
`git add filename`

Run command after file was
deleted from working directory

Unstaged
Changes

2.23

`git checkout (--)` .
`git restore filename` or .

Revert changes in tracked files

`git clean -df`

Delete untracked files

Staged
Changes

2.23

`git reset filename &`
`git checkout -- filename`
`git restore --staged filename` or .

Remove file(s) from
staging area

Latest
Commit(s)

`git reset HEAD~1`
`git reset --soft HEAD~1`
`git reset --hard HEAD~1`

Undo latest (~1) commit

Branches

`git branch -D branchname`

Delete branch

* Already part of previous commit

Diving Deeper Into Git

Beyond The Basics

Module Content

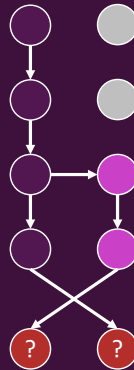
Diving Deeper into Commits

Managing & Combining Different Branches

Resolving Conflicts

Combining Master & Feature Branches

Master = Main project branch



Feature = Separate branch "based" on master branch

Goal:
Combining master & feature branch

Merge Types

Fast-Forward

Non Fast-Forward

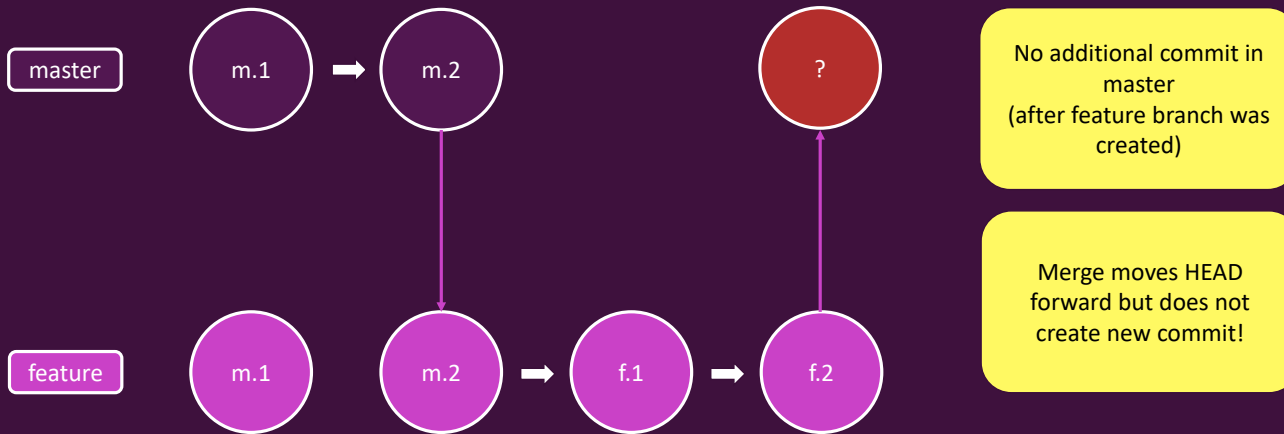
Recursive

Octopus

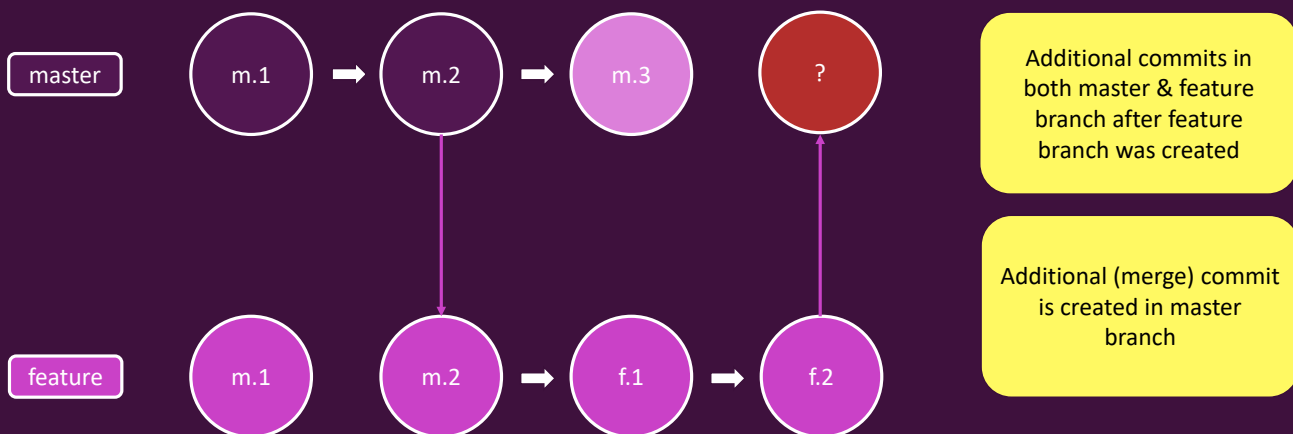
Ours

Subtree

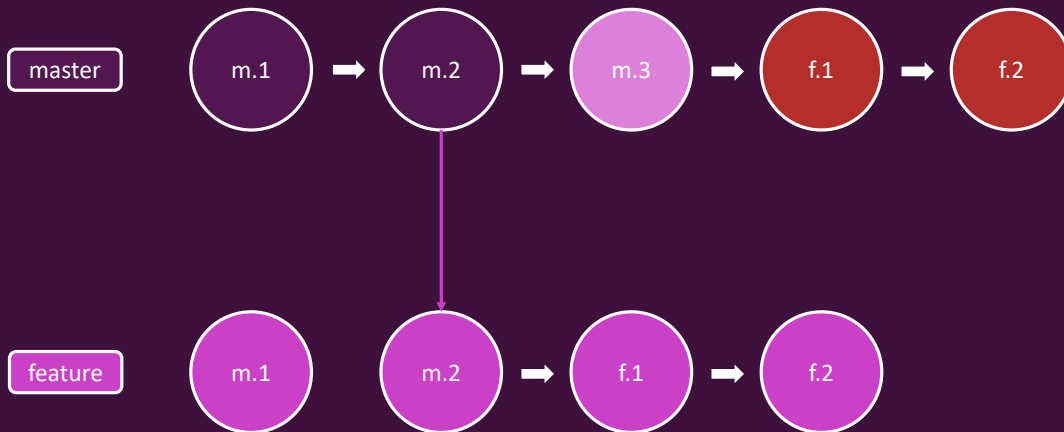
Master & Feature – Merge (“fast-forward”)



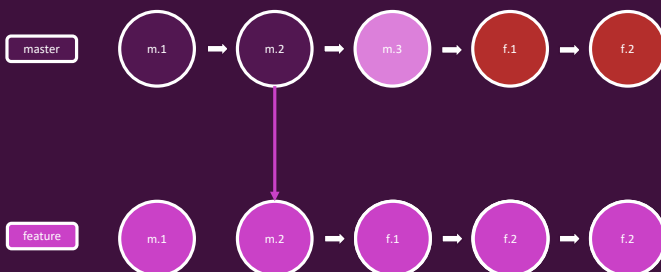
Master & Feature – Merge (“recursive”)



Master & Feature – Rebase



Rebase – What happened?



“m.3” becomes new base commit for commits created in feature branch

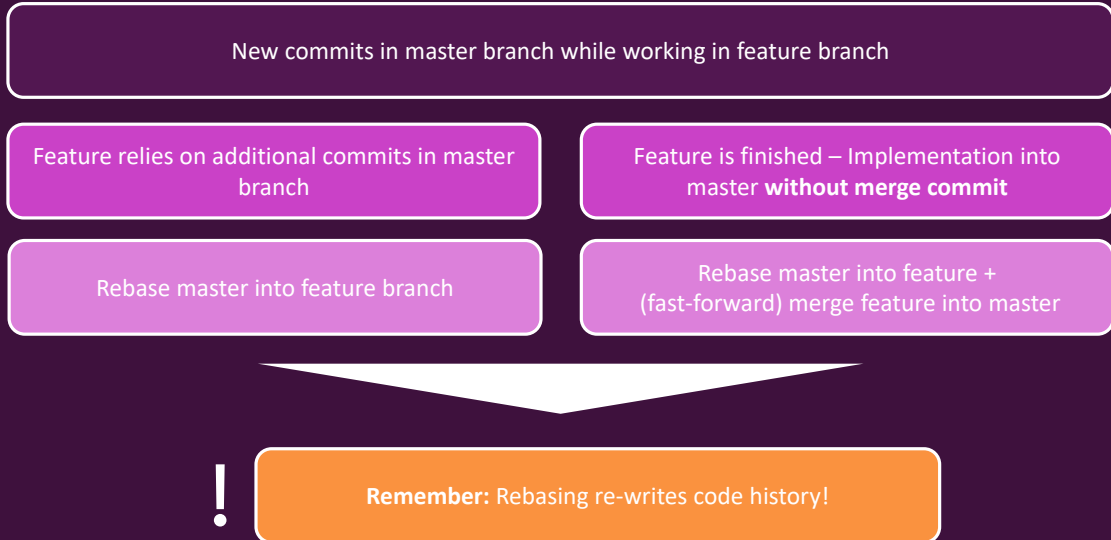
rebase master to feature branch

merge rebased feature into master

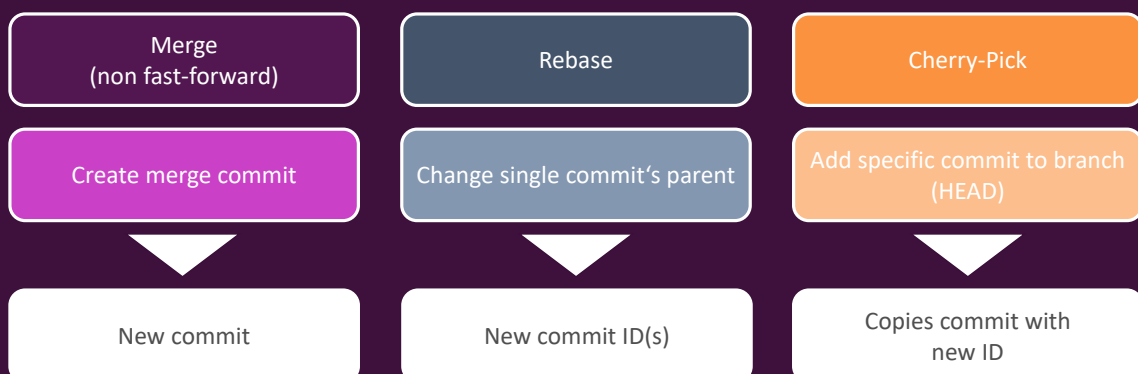
“rebase” does **not** move commits, it creates new commits

Never rebase commits outside your repository!

Rebase – When to Apply?



Merge vs Rebase vs Cherry-Pick



Deep Dive Summary

git stash

Temporary storage for unstaged and uncommitted changes

git reflog

A log of all project changes made including deleted commits

git merge

Combining commits from different branches by creating a new merge commit (recursive) or by moving the HEAD (fast-forward)

git rebase

Change the base (i.e. the parent commit) of commits in another branch

git cherry-pick

Copy commit including the changes made only in this commit as HEAD to other branch

Understanding GitHub

Leaving the Local Repository

Module Content

What is GitHub &
How Git & GitHub are Connected

Remote Branches, Remote Tracking Branches & Local Tracking
Branches

Understanding Upstreams & Git Clone

About Git & GitHub



Version Control System

Manage Code History

Track Changes

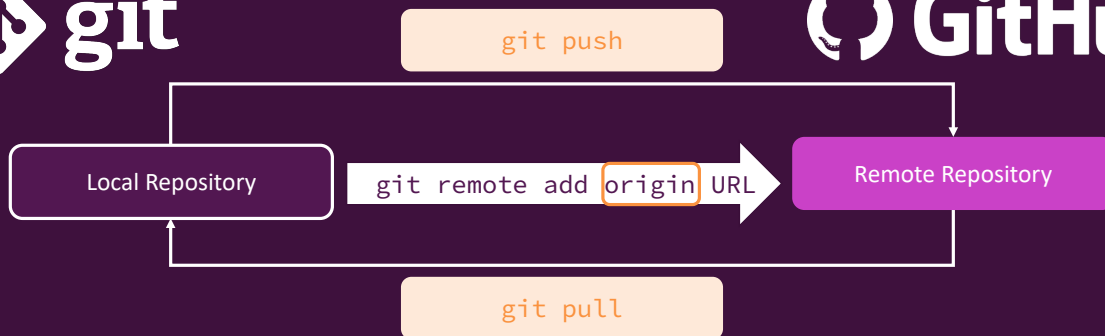


Largest Development Platform

Cloud Hosting &
Collaboration Provider

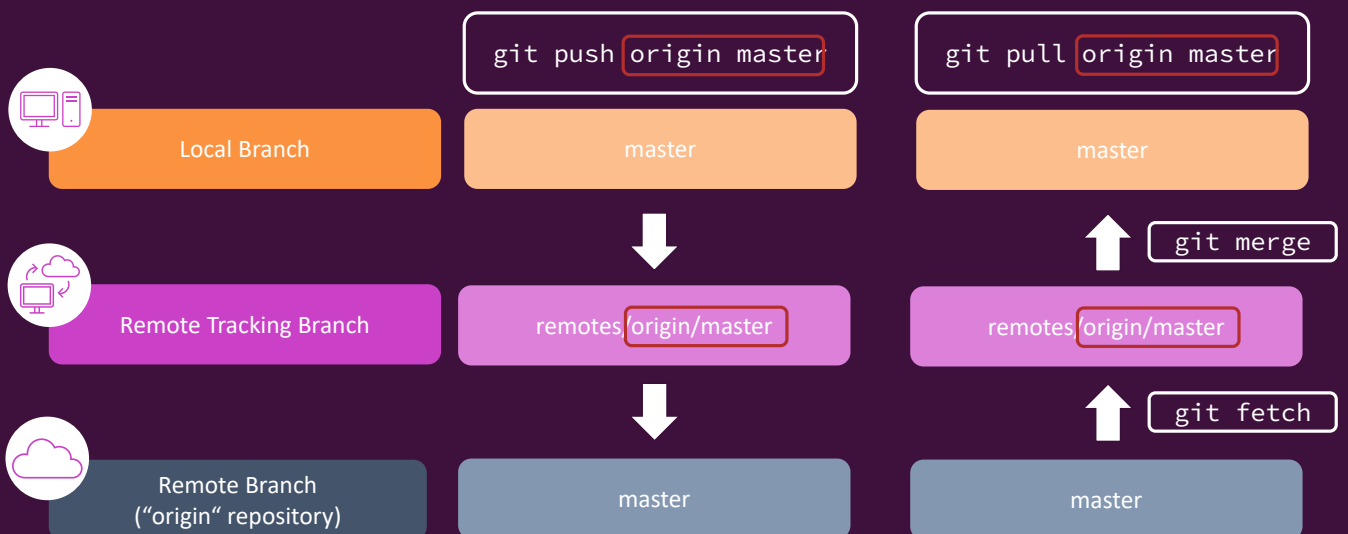
Git Repository Hosting

Connecting Git & Github – Local to Empty Remote Repo



"Name of the remote machine" or
alias/shorthand of the URL of the remote repository

More Branches?



Remote repository's name ("origin") and branch name must always be added

Branch Types - Overview



Local Branch

Branch on your machine only (as seen in the course)



Remote Branch

Branch in remote location (e.g. in GitHub)



Tracking Branch

Remote Tracking Branch

Local copy of remote branch (not to be edited)

`git fetch`

Local Tracking Branch

Local reference to remote tracking branch (to be edited)

`git push`

`git pull`

Local & Remote Tracking Branches



Remote Branch

`git remote`

Show remote servers



`git fetch`



Remote Tracking Branch

Local cache of remote branch's content

`git branch -a`

List all branches

`git branch -r`

Show remote tracking branches

`git remote show origin`

Show detailed configuration



`git merge`

`git push`



Local Tracking Branch

Remote repository's name ("origin") and branch name can be omitted

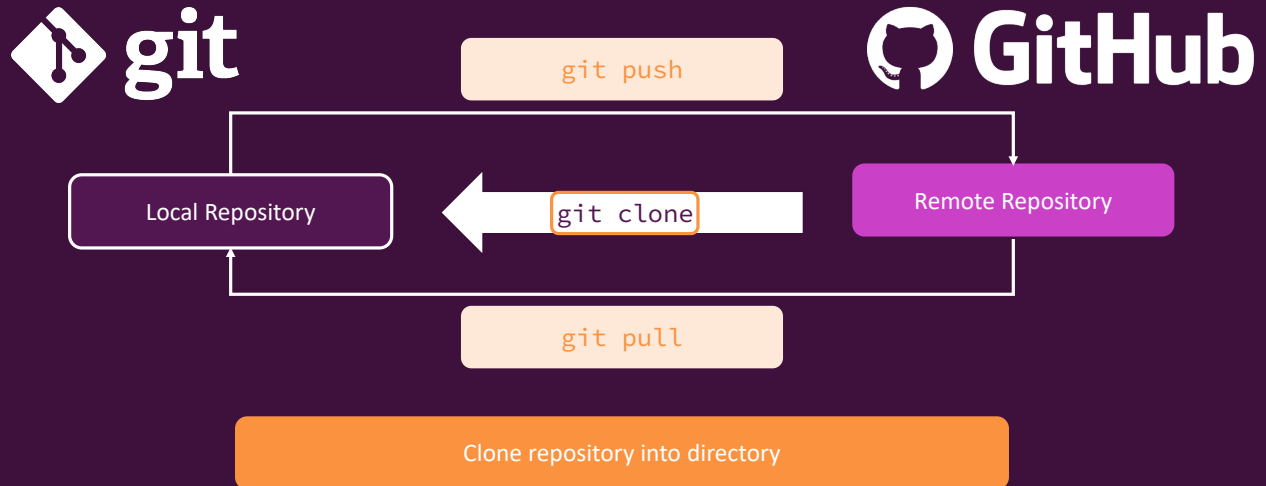
`git branch -vv`

List local tracking branches and their remotes

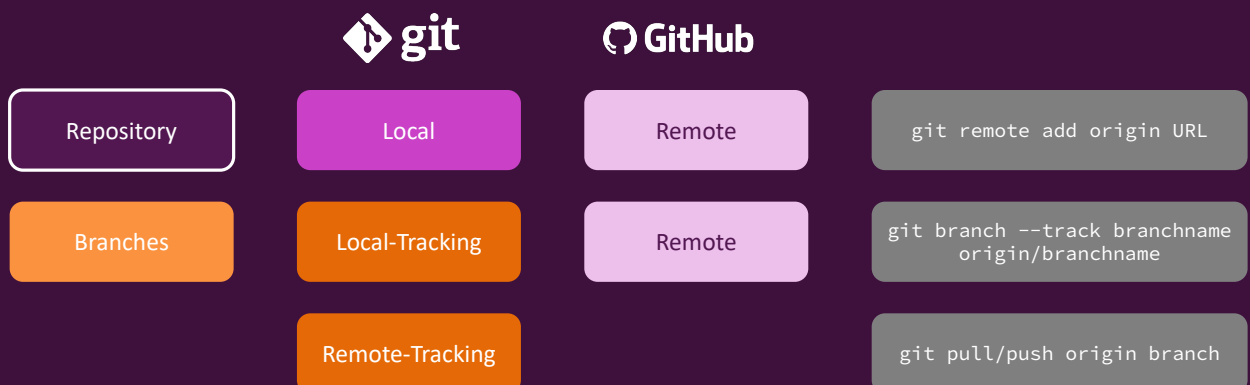
`git branch --track
branchname origin/branchname`

Create local tracking branch

Connecting Git & Github – Remote to Empty Local Repo



GitHub – Summary



GitHub – Deep Dive

Collaboration & Contribution

Module Content

Understanding GitHub Accounts
& Repository Types

Collaborating in GitHub
& Contributing to Open Source Projects

Creating your GitHub Portfolio Page
& More Features to Explore

Why We Use GitHub



Cloud Storage



Portfolio Page



Collaboration



Contribution

Understanding Account Types

Personal User Account

Organizational Account

Enterprise Account

Every GitHub user's
user account

Shared account for groups of
people to collaborate

Central management of multiple
GitHub accounts

Unlimited public & private
repositories

Base features as for
personal account

GitHub Enterprise Cloud & GitHub
Enterprise Server

Unlimited collaborators

Advanced features with
GitHub Team/Enterprise

Enterprise level only

Base features included in "Free" pricing plan

"Enterprise" pricing only

GitHub Security & Access



Local Git User

Personal Access Token

GitHub Account

GitHub account
access via Git

Personal user account

Part of organization

Owner
access

Collaborator
access

Member role access

Repository 1

Repository 1

Repository 2

Repository 2

...

...

Comparing Clone...

Clone



GitHub

Account A



GitHub

Account B

Remote repo

Direct connection

Local copy

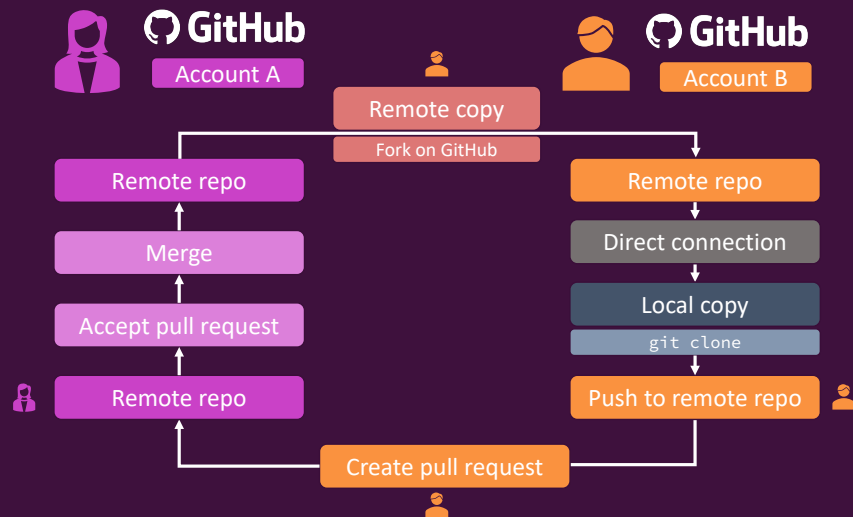
git clone

Push to remote repo



...with Forks & Pull Requests

Fork & Pull Request



Module Summary

GitHub

Account Types

Repository Types

Security

Collaboration

Collaborators

Organizations

Teams

Contribution & Project Management

Forks & Pull Requests

Issues

Projects

Applying Our Knowledge: Food Order Project

A Real-World Example

Module Content

Creating & Using A Local Git Repository

Managing Code On Github

Collaboration: Merging & Pull Requests

Congratulations!

You Finished this Course!

Congratulations!



CONGRATULATIONS!

Working Directory

Stash

Tags

Staging Area

Reflog

Origin & Remote

Repository

Merge

Clone

Commits & Branches

Rebase

Tracking & Upstreams