Git

Part 3

By : Ali Moeinian

3 – کورس یودمی

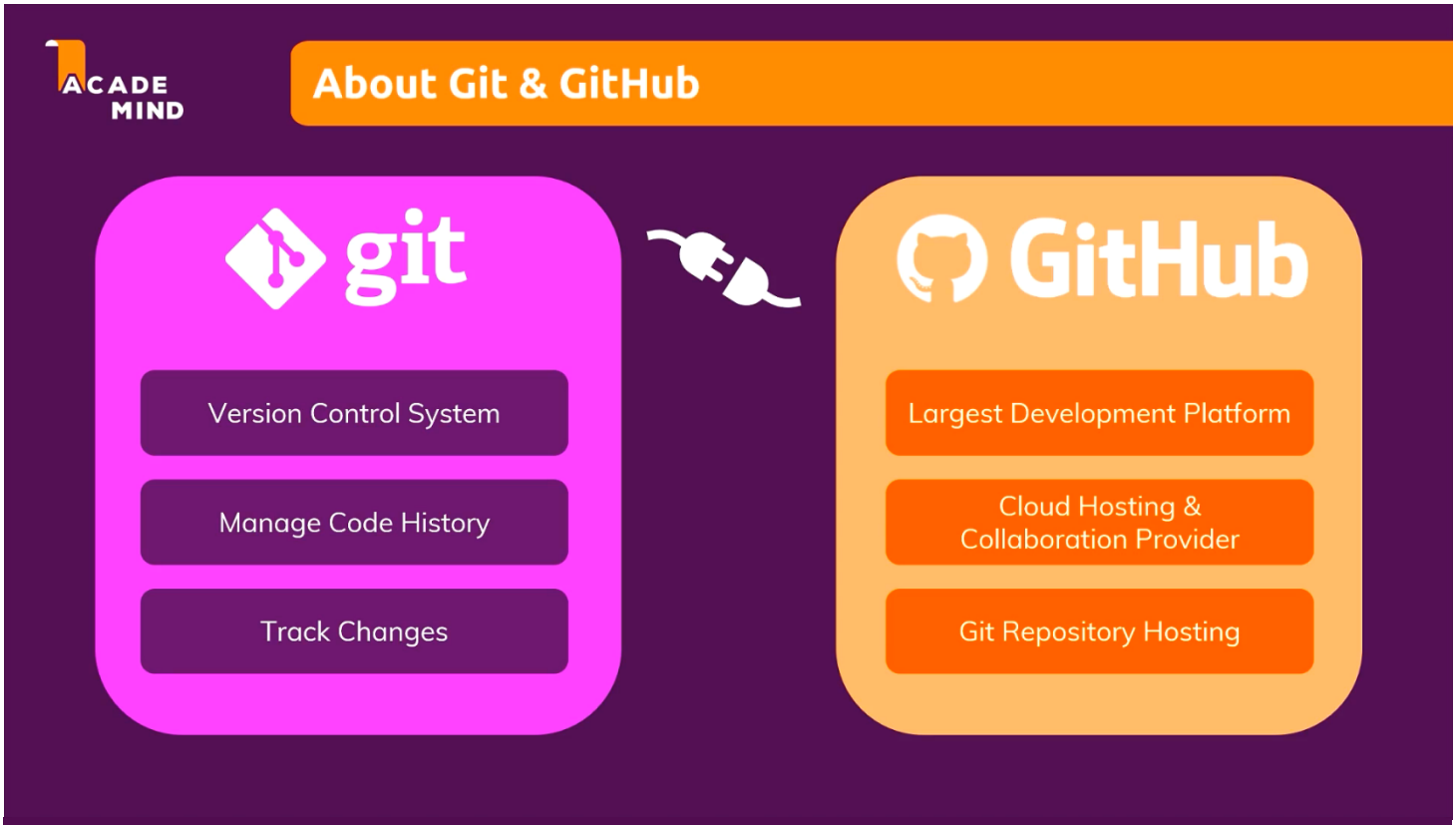
حالا دیگه وقت ایشونه 😎😎😎😎

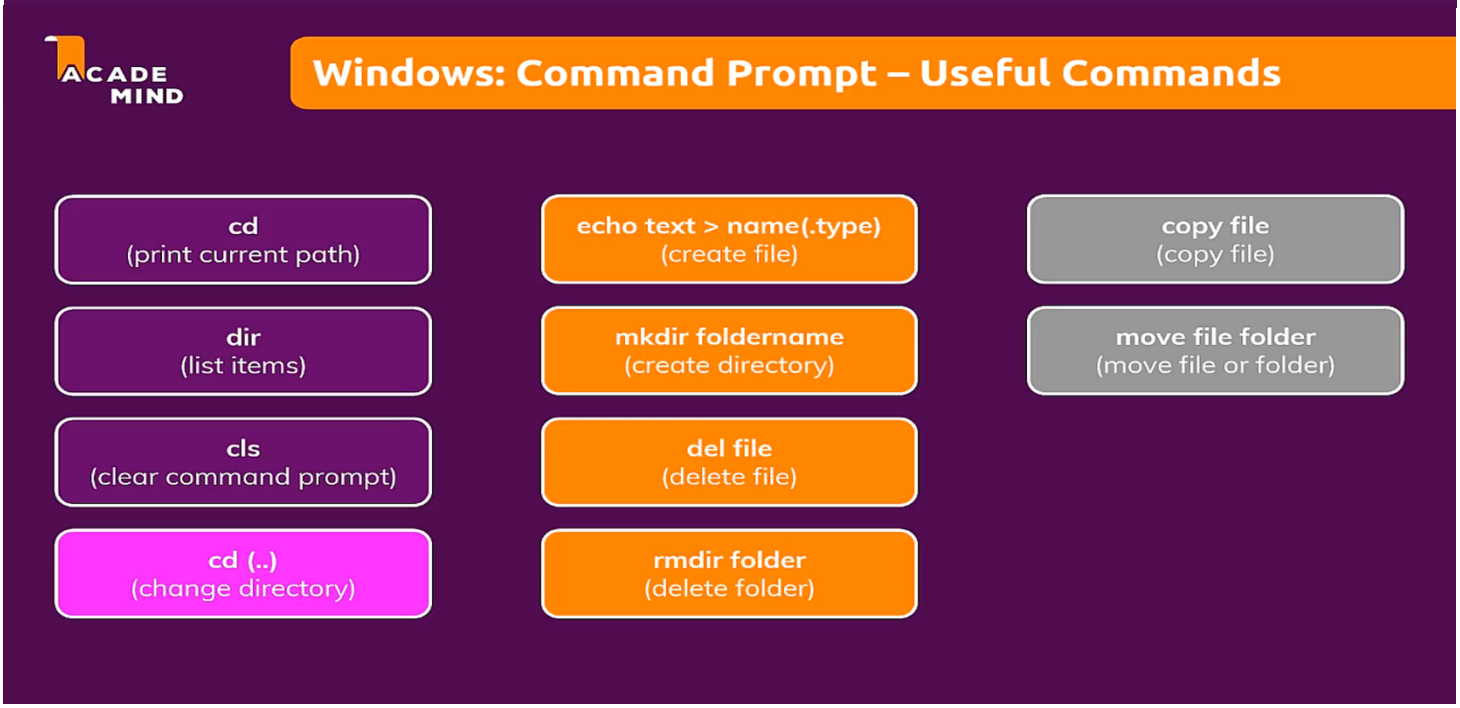
Maximilian schwarzmüller

بریم سراغ دوره ی ایشون که در سایت یودمی قرار داره و 120 قسمته که خیلی زیاده و قطعا خیلی هم طول میکشه ولی واسه ی اینکه کارم کامل باشه حتما باید ببینمش پس بزن بریم (از سایت دانلودلی دانلودش کردم 🙄😁 خدا ببخشه!)

حتما باید بگم که فقط و فقط نکات مهم و جدید در این قسمت ذکر میشن و از نوشتن نکات تکراری پرهیز میکنم.

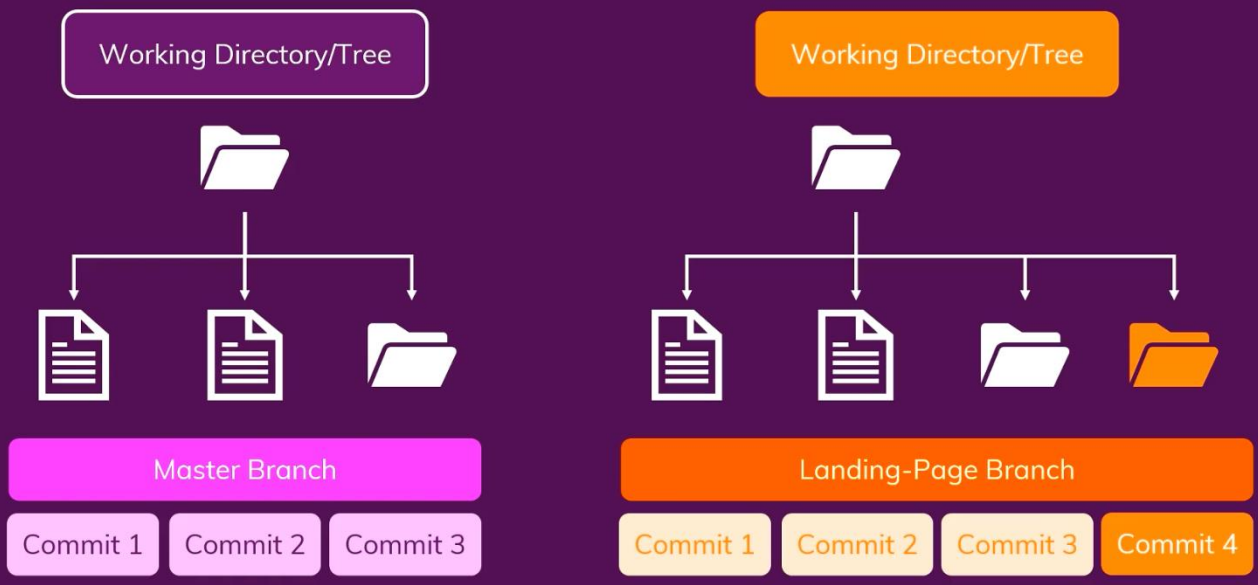
روند کلی به این شکل خواهد بود که من نکات جدید، اسلاید ها جمع بندی جالب و هر عکس جدیدی رو قرار میدم و در صورتی که نیاز به توضیح داشته باشه، توضیح مختصری در اون باره خواهم نوشت و از گفتن مطالب تکراری خودداری خواهم کرد.





ACADEMIND

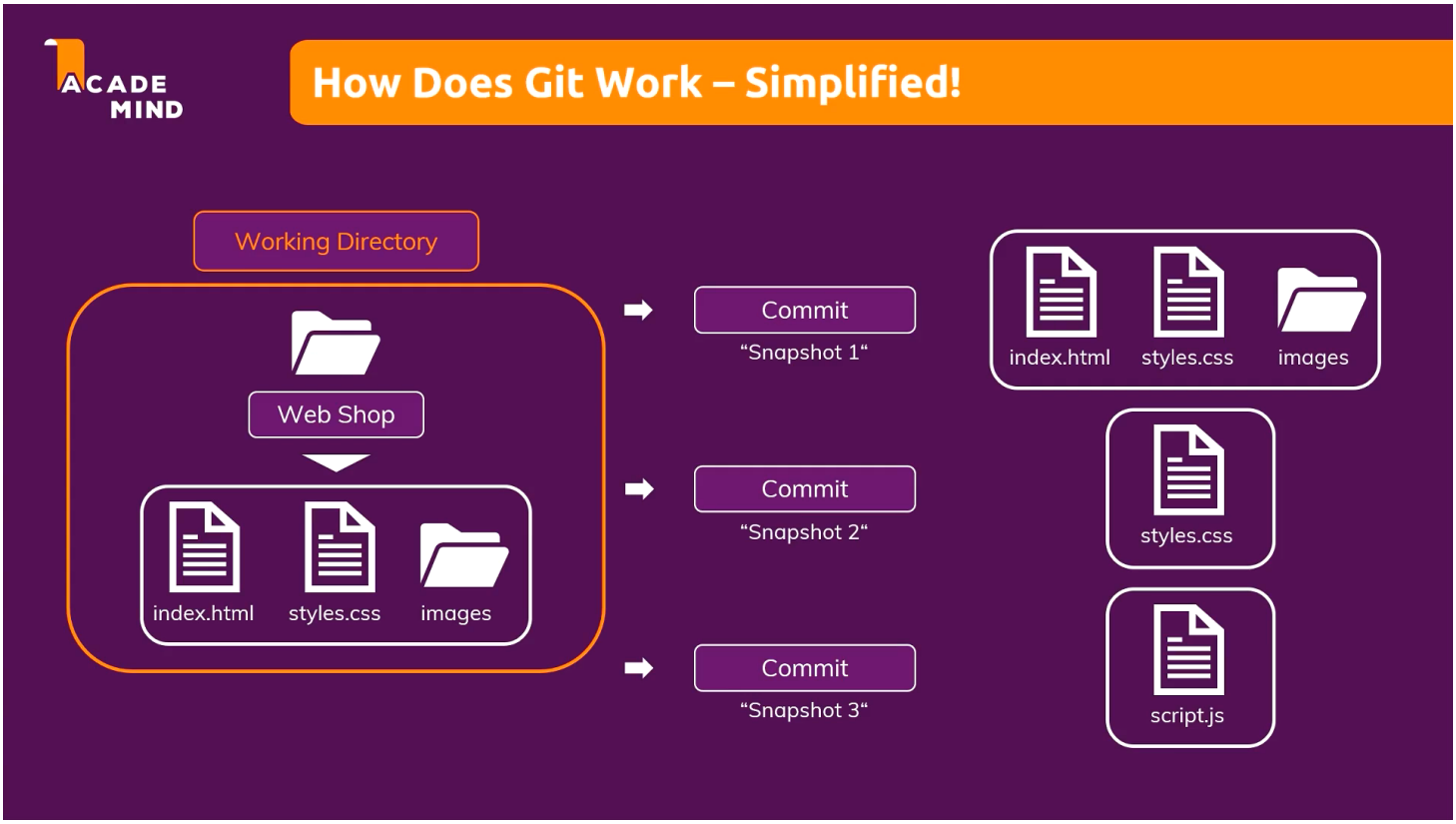
Branches & Commits



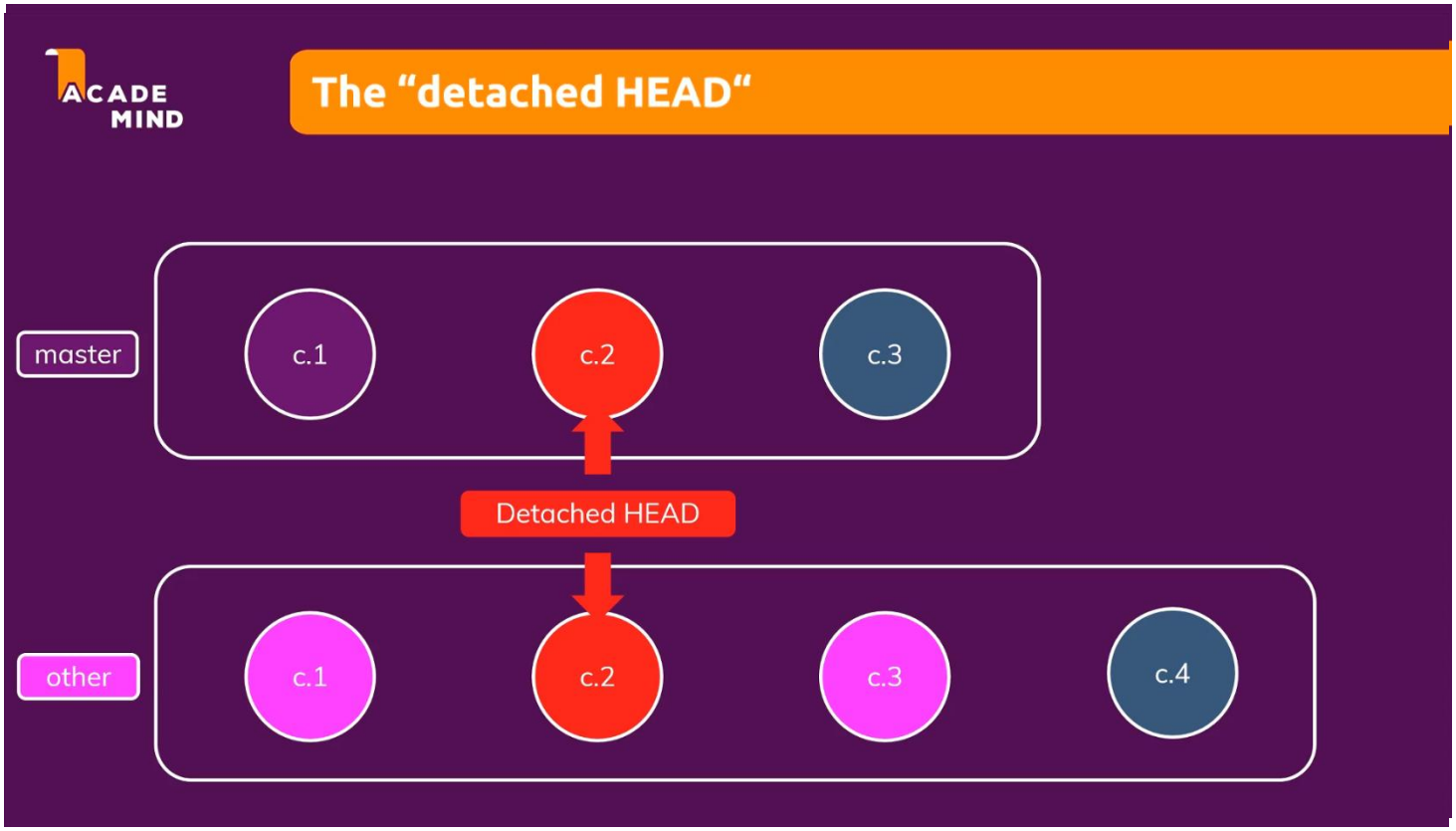
Play ACADEMIND

Mac: Terminal (z-Shell) – Useful Commands

pwd (print working directory)	**cd /Users** (users directory)	**r** (delete file)
ls (list items)	**cd or cd ~** (home directory)	**rmdir** (delete empty folder)
cd (..) (change directory)	**touch** (create/ "touch" file)	**cp** (copy file/folder)
cd / (root directory)	**mkdir** (create directory)	**mv** (move file/folder)



نکته ی جالبی که بیان شد، این بود که در vs code در قسمت terminal دقیقا میتونید تمام دستور های مربوط به گیت رو استفاده کنید.



یکی از دستوراتی که به نظرم جدید و جالب بود :

```
git switch third-branch
```

با این دستور به راحتی میتونید به برنچ های برنامتون تغییر مسیر بدید که فکر کنم قبلا بیان نکرده بودم.

یک نکته ی جدید اینکه اگه به طور دستی یک فایلی رو از دایرکتوری مد نظرتون حذف کردید، حتما باید از دستور git rm هم استفاده کنید تا فایل حذف شده دیگر تحت گیت نباشه و بعد هم حذف کردن این فایل رو کامیت کنید و در نهایت هم push و تماااام 😊

ACADEMIND

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!)

ACADEMIND

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

ACADEMIND

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	`git checkout (--) .` 2.23 `git restore filename or .`	Revert changes in tracked files
	`git clean -df`	Delete untracked files

Staged Changes	`git reset filename &`
	`git checkout -- filename`
	2.23 `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

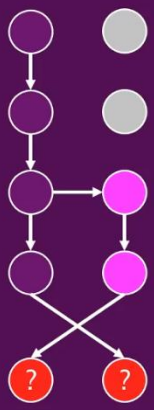
Branches	`git branch -D branchname`	Delete branch

* Already part of previous commit

ACADEMIND

Combining Master & Feature Branches

Master = Main project branch



Feature = Separate branch “based“ on master branch

Goal:
Combining master & feature branch

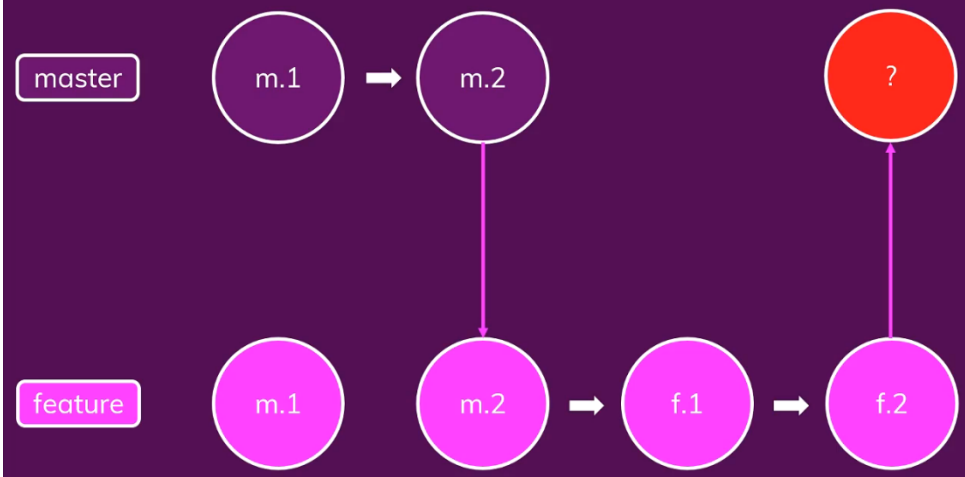
ACADEMIND

Merge Types

- Fast-Forward
- Non Fast-Forward
  - Recursive
  - Octopus
  - Ours
  - Subtree

ACADEMIND

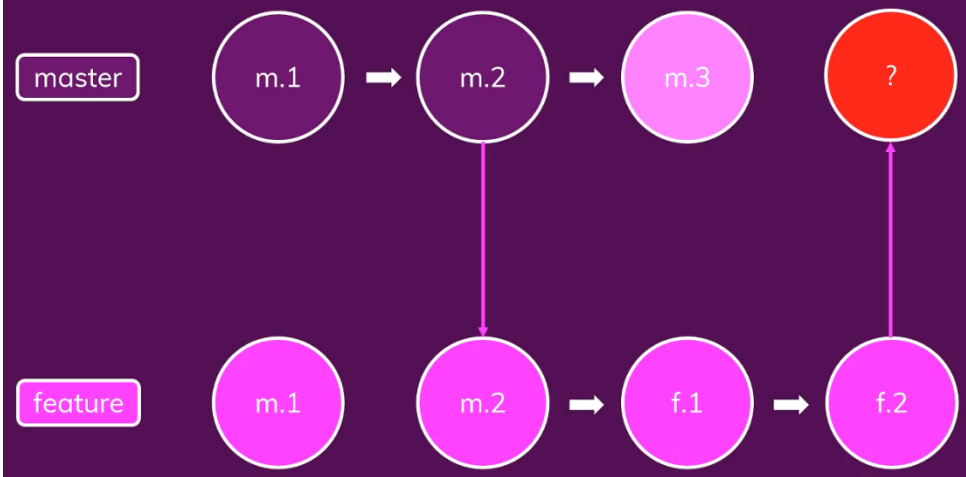
Master & Feature – Merge ("fast-forward")



No additional commit in master (after feature branch was created)

Merge moves HEAD forward but does not create new commit!

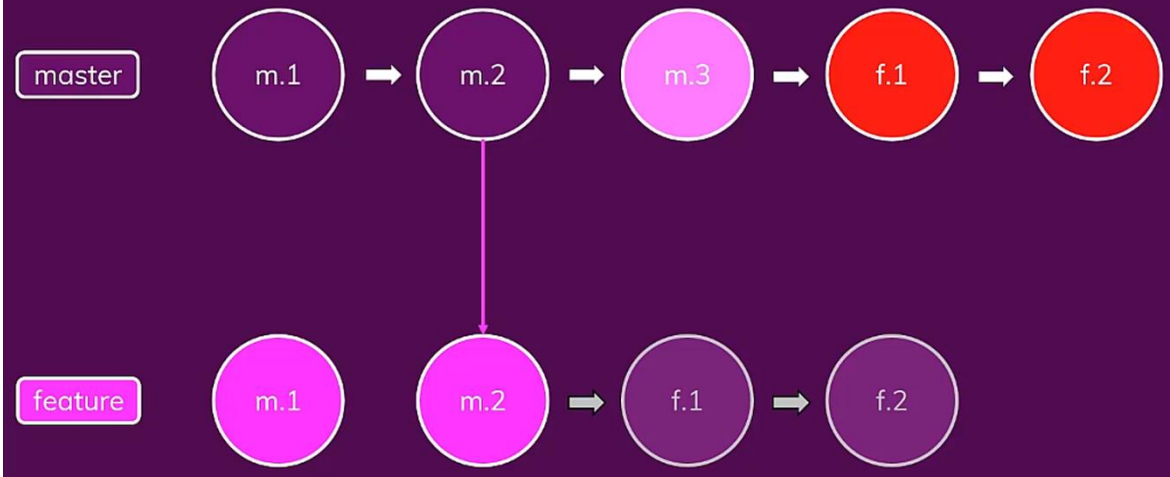
Master & Feature – Merge ("recursive")



Additional commits in both master & feature branch after feature branch was created

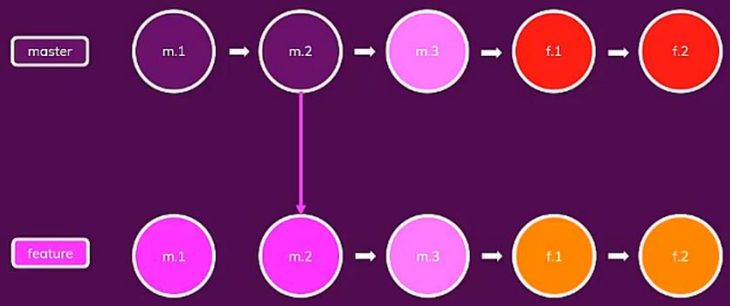
Additional (merge) commit is created in master branch

Master & Feature – Rebase



ACADEMIND

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!

ACADEMIND

Rebase – When to Apply?

New commits in master branch while working in feature branch

Feature relies on additional commits in master branch

Feature is finished – Implementation into master **without merge commit**

Rebase master into feature branch

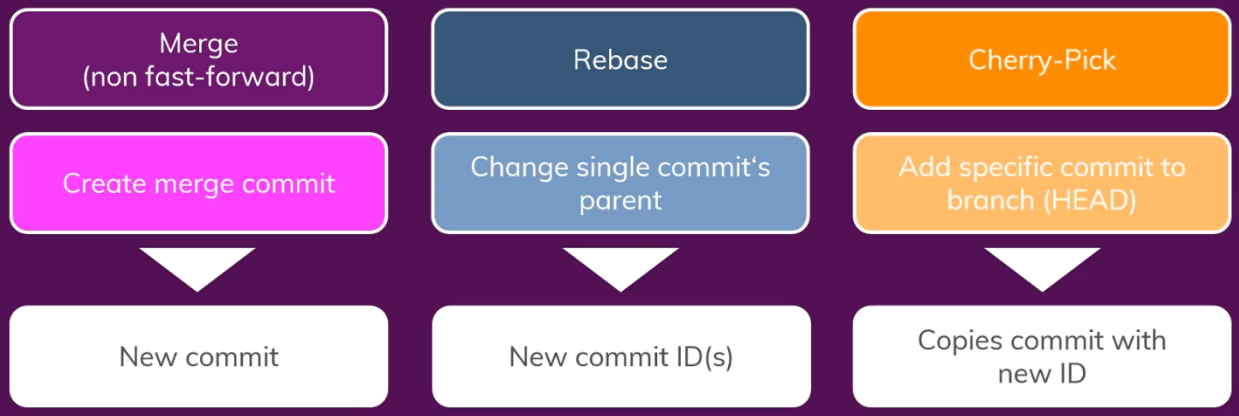
Rebase master into feature + (fast-forward) merge feature into master

!

Remember: Rebasing re-writes code history!

ACADEMIND

Merge vs Rebase vs Cherry-Pick



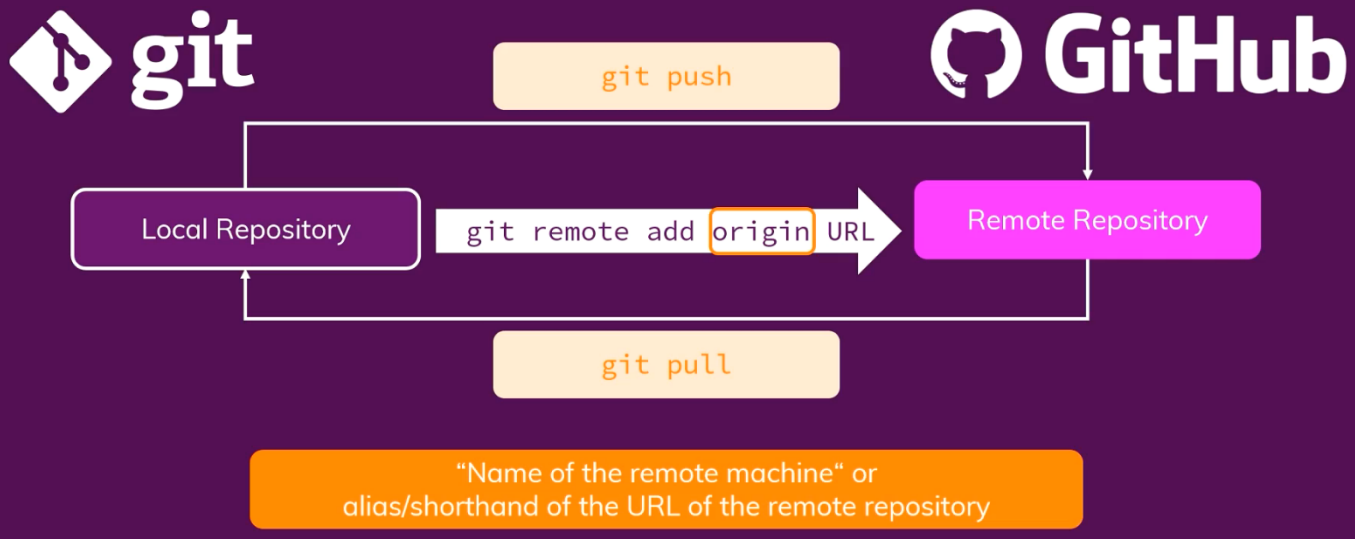
ACADEMIND

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

ACADEMIND

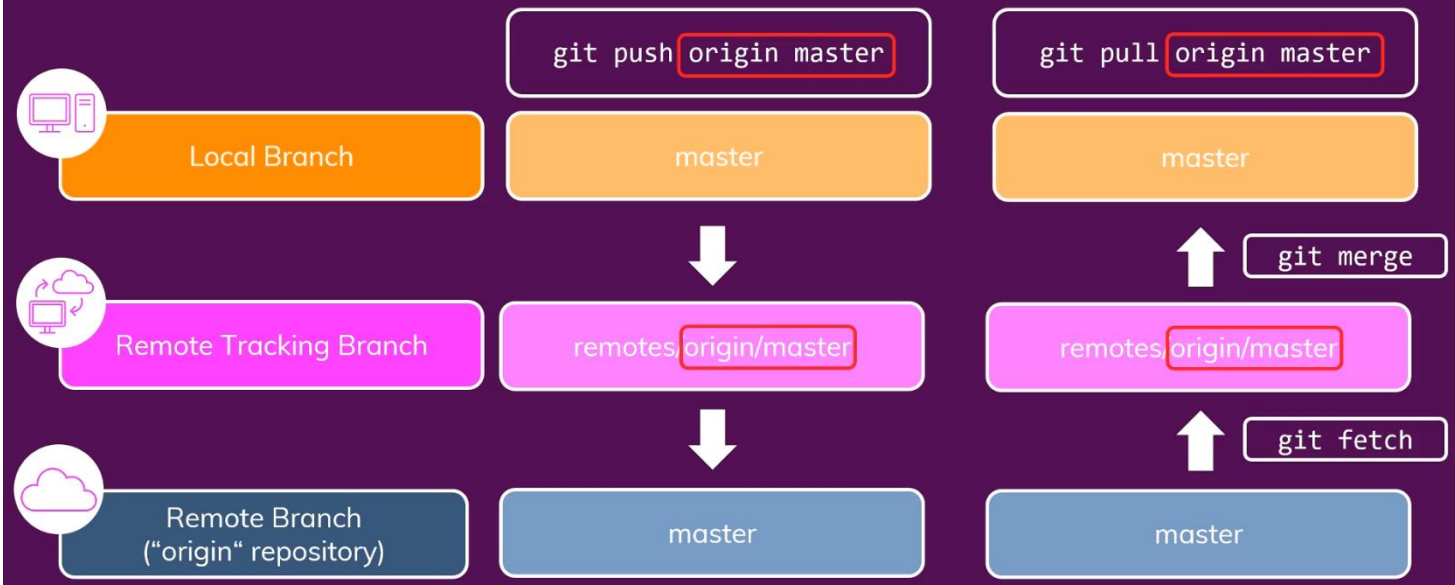
Connecting Git & Github – Local to Empty Remote Repo



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

ACADEMIND

More Branches?



ACADEMIND

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`

ACADEMIND

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

ACADEMIND

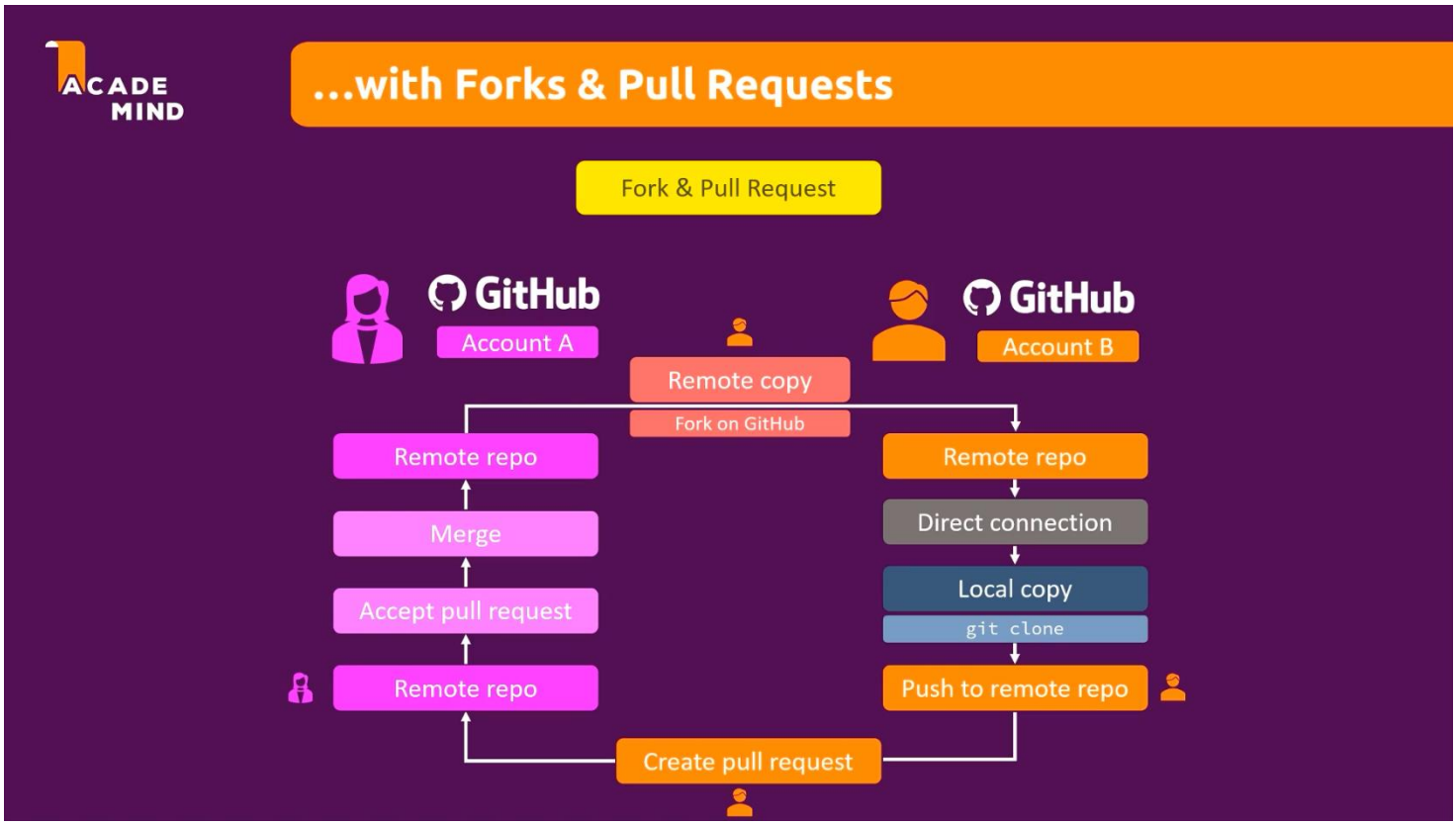
GitHub – Summary

	git	GitHub	
Repository	Local	Remote	`git remote add origin URL`
Branches	Local-Tracking	Remote	`git branch --track branchname origin/branchname`
	Remote-Tracking		`git pull/push origin branch`

ACADEMIND

GitHub Security & Access

git		GitHub	
Local Git User	Personal Access Token	GitHub Account	
	GitHub account access via Git	Personal user account	Part of organization
		Owner access	Collaborator access



✅ این کورس بیش از 100 قسمت بود.

✅ من فقط اسلاید های مهم و خلاصه ی مطالب رو براتون آوردم.

✅در این کورس سناریو های مختلفی مورد بررسی قرار گرفت و بر عکس قسمت اول این سری PDF ها، تمام دستورات گیت در vs code زده شد و اصلا اشاره ای به git bash نشد.

✅در قسمت های اخر هم یک پروژه در گیت هاب قرار داده شد و برخی سناریو های مختلف هم مورد بررسی قرار گرفت.

❌در کل وقتی به طور جدی پروژه هاتون رو تحت گیت بردید و با چالش های جدیدش روبرو شدید، به مشکل خوردید و شروع به جستجو کردید، مطالب خیلی بیشتری رو میتونید یاد بگیرید و هدف از این 3 فایل، صرفا دوره و آموزش ابتدایی گیت بود.

فراموش نکنید که بهترین منبع گیت کتاب git pro خواهد بود که در همین ریپوزیتوری میتونید دانلود کنید.

پایان