

Purpose of document: guide on lectures with commands used and description

Command	Explanation	Remarks / Common arguments	Example
git clone	Clone a remote repository given the URL	99% of the time you will pass the repository URL as the only argument to git clone.	git clone https://github.com/Pierian-Data/Git-and-GitHub-Zero-to-Hero.git

Command	Explanation	Remarks / Common arguments	Example
git add	Stage files / add files to the index for subsequent committing	<p><i>git add -A</i> : Stage all files</p> <p><i>git add -u</i> : Stage all tracked files (i.e files which have been added before after they were altered)</p> <p><i>git add /path/to/file</i> : Stage the file identified by its path</p>	git add ReadMe.md
git status	Lists all added, changed and newly created files.	Typically no arguments necessary. You can use <i>git status -s</i> to get a shorter version	git status
git reset	Undo changes / unstage files / go back to commit	<p><i>git reset</i> without any arguments unstages all added files but preserves all changes.</p> <p><i>git reset --hard</i> unstages all added files and deletes all changes you made since the last commit. CAUTION!</p> <p><i>git reset /path/to/file</i> to unstage a single file</p> <p><i>git reset --hard commitId</i> to jump back to the commit with <i>commitId</i></p>	git reset git reset test.txt git reset --hard 5b331f3
git restore	Unstage specific files / undo specific changes	<p><i>git restore --staged</i> to unstage added files</p> <p><i>git restore /path/to/file</i> to undo changes since last commit. Only possible when unstaged.</p> <p>Difference between reset and restore</p>	git restore test.txt git restore --staged test.txt

git log	Show commit history	<p>Pass no arguments to get the full log of the corresponding branch or use filtering arguments such as <code>--after</code>, <code>--author</code> or <code>-n</code></p> <p>More information on formatting and filtering can be found here</p>	<pre>git log --after="2022-1-1" git log --after="yesterday" git log -n 10 git log --author="Jose"</pre>
git diff	Visualize changes	<p><i>git diff</i> to list all changes since the last commit (unstaged files)</p> <p><i>git diff --cached</i> for staged files</p> <p><i>git diff /path/to/file</i> for a single file</p> <p><i>git diff commitID1 commitID2</i> to compare between commits</p> <p><i>git diff branch1 branch2</i> to compare between branches</p>	<pre>git diff --cached test.txt git diff 4598 3g62 test.txt git diff main development</pre>
git commit	Commit changes after staging them	<p>Typically you only use:</p> <p><i>git commit -m "Message"</i></p> <p>An empty message aborts the commit command.</p> <p>If <code>-m</code> is not passed, git opens a text editor to write the message</p> <p><i>git commit --amend -m "Changed message"</i> to change the commit message of the previous commit</p>	<pre>git commit -m "Updated ReadMe.md" git commit --amend -m "your new message"</pre>

git push	Push new commits to the remote repository	<p><i>git push</i> to push to the branch you are currently on and the remote repository defined in .git/config</p> <p><i>git push repository</i> to push to a different remote repository</p> <p><i>git push repository sourceBranch</i> to push the desired branch (i.e not one you are currently on)</p> <p><i>git push repository sourceBranch:targetBranch</i> to push to the targetBranch from the sourceBranch</p> <p><i>git push --force</i> to force push your current commit ignoring potential conflicts</p>	<pre>git push git push origin git push origin main git push origin main:test</pre>
--------------------------	---	--	--

Command	Explanation	Remarks / Common arguments	Example
git branch	"List, create, or delete branches"	<p><i>git branch</i> to list all branches</p> <p><i>git branch name</i> to create a new branch called <i>name</i></p> <p><i>git branch -delete name</i> to delete the branch called <i>name</i></p>	<p>git branch</p> <p>git branch development</p> <p>git branch -delete development</p>
git switch	Switch to another branch	<p><i>git switch name</i> to switch to the branch name</p> <p><i>git switch -c name</i> to create the branch name if it does not exist and switch to it</p> <p><i>git switch -d commitId</i> to switch to a previous commit</p> <p><i>git switch -m name</i> merges the changes of the current branch into name and switches to <i>name</i></p>	<p>git switch development</p> <p>git switch -c development2</p> <p>git switch -d h98uab</p> <p>git switch -m main</p>
git checkout	"Switch branches or restore working tree files"	<p><i>git checkout name</i> to switch to the branch name</p> <p><i>git checkout -b name</i> to create the branch name if it does not exist and switch to it</p> <p><i>git checkout commitId</i> to switch to a previous commit</p> <p><i>git checkout -m name</i> merges the changes</p>	<p>git checkout development</p> <p>git checkout -b development2</p> <p>git checkout h98uab</p> <p>git checkout -m main</p> <p>git checkout test.txt</p>

		<p>of the current branch into name and switches to <i>name</i></p> <p><i>git checkout /path/to/file</i> to undo changes since the last commit (i.e git restore)</p> <p>Difference switch and checkout</p>	
git merge	Merge / join two branches	<p><i>Typically done on github, but for the sake of completion:</i></p> <p><i>git merge branch1 branch2</i> merges <i>branch1</i> and <i>branch2</i> on the current branch (i.e a new commit is created)</p> <p><i>git merge branch</i> to merge <i>branch</i> into the current branch</p> <p><i>git merge -s strategy branch</i> to define the merging strategy</p>	<pre>git merge devel git merge devel1 devel2 git merge -s ours devel</pre>
git tag	"Create, list, delete or verify a tag"	<p>Can also be done on github</p> <p><i>git tag</i> to list all tags</p> <p><i>git tag tagname</i> to create a tag called <i>tagname</i></p> <p><i>git tag tagname -a</i> to add an annotated tag with the name <i>tagname</i> (will open editor)</p> <p><i>git tag -delete tagname</i> to delete the tag <i>tagname</i></p>	<pre>git tag v1.0 git tag v1.0 -a git tag -delete v1.0</pre>

git fetch	Fetch changes from the remote repository (does not update head)	<p><i>git fetch</i> to get the new commits from the branch you are currently on and the remote repository defined in .git/config</p> <p><i>git fetch repository</i> to update from a different remote repository</p> <p><i>git fetch repository sourceBranch</i> to get the desired branch (i.e not one you are currently on)</p> <p><i>git fetch repository sourceBranch:targetBranch</i> to get the sourceBranch into the targetBranch</p>	<pre>git fetch git fetch origin git fetch origin main git fetch origin main:test</pre>
git pull	Update local version with remote version. git fetch + git merge	<p><i>git pull</i> to pull from the branch you are currently on and the remote repository defined in .git/config</p> <p><i>git pull repository</i> to pull from a different remote repository</p> <p><i>git pull repository sourceBranch</i> to pull the desired branch (i.e not one you are currently on)</p> <p><i>git pull repository sourceBranch:targetBranch</i> to pull the sourceBranch into the targetBranch</p>	<pre>git pull git pull origin git pull origin main git pull origin main:test</pre>

Command	Explanation	Remarks / Common arguments	Example
---------	-------------	----------------------------	---------

git rebase	Rewrite commit history	<p><i>git rebase -i HEAD~n</i> to rebase the last <i>n</i> commits in the interactive mode</p> <p><i>git rebase main</i> to rebase <i>main</i> on the current branch</p> <p><i>git rebase --onto newbase oldbase</i> for more advanced rebasing with specific branches</p> <p>More in depth guide here, here Caution: Do not rebase after pushing!</p>	git rebase -i HEAD~5
git revert	Revert existing commits and create new commit with these changes	<p><i>git revert commitId</i> creates a new commit containing the state of <i>commitId</i>. The editor will be opened to enter the commit message</p> <p>Difference revert reset</p>	git revert 2fc0df

Command	Explanation	Remarks / Common arguments	Example
---------	-------------	----------------------------	---------

Day 4			
git stash	Stash changes for later use	<p><i>git stash</i> to add a new stash entry with the current modifications and reset your state to the current HEAD</p> <p><i>git stash list</i> to get all stash entries</p> <p><i>git stash show</i> to visualize the changes (diff)</p> <p><i>git stash pop</i> to pop the first element of <i>git stash list</i></p> <p><i>git stash pop stash@{i}</i> to get the <i>ith</i> element of the stack</p> <p><i>git stash apply</i> works similar to <i>git stash pop</i> but does not remove the stash from the list</p> <p>More examples</p>	<p>git stash git stash list git stash show git stash pop git stash pop stash@{2}</p>
git clean	Delete all files not tracked by git	<p><i>git clean</i> to recursively remove all files not tracked</p> <p><i>git clean -n</i> to list the files which would be deleted</p> <p><i>git clean -x</i> to also delete files ignored via <i>.gitignore</i></p> <p><i>git clean -X</i> to only delete ignored files</p>	<p>git clean git clean -n git clean -x</p>