Cheatography

Git Cheat Sheet

by KYS1477 via cheatography.com/186487/cs/39697/

Principles

Create a git repository for every new project
Create a new branch for every new feature
Branch early, and branch often
SSH keys are how we securely communicate btw our computer and GitLab

Typical Workflow

git clone <repo>	clone repo located at <repo> onto local machine, clone with SSH
cd my_project	after git clone, go to the directory, before using all the git commands
git checkout -b my_branch origin/remote_branch1	create and checkout my_branch, that is tracking remote_branch1
git add <file_name>	
git commit -m "message"	commit the change to my_branch
git pull	update local my_branch with remote commits and update all remote tracking branches
git merge origin/remote_branch2	merge remote_branch2 with the branch that you are currently on (my_branch)
git push origin	push the branch that you are on (my_branch) to origin (remote repo)
(create a merge request on GitLab UI)	rmb to change target branch

Typical Workflow 2 - PyCharm

git pull	before pushing your changes, sync with the remote and make sure your local copy of the repository is up to date to avoid conflicts
git push	push changes from the current branch
define remote and select target branch	click on Define remote link (appears when there is no remotes in the repository), click on the branch name

https://www.jetbrains.com/help/pycharm/commit-and-push-changes.html#force-push

Pushing and merging code change

git add <file_name> / git add --all	add the file to the staging area
git commit -m "short_msg"	takes a permanent snapshot of the current state of your repository that is associated with a unique identifier
git push origin <branch_name>	pushes a local branch(es) to a remote repository (origin - the conventional shorthand name of the url for the remote repository)
git checkout main	checkout the default branch of your repo
git merge <branch_name>	merge your branch into the default branch
git push	push the changes

https://docs.gitlab.com/ee/tutorials/make_first_git_commit/

Cheatography

Git Cheat Sheet

by KYS1477 via cheatography.com/186487/cs/39697/

Check

git status	list which files are staged, unstaged, and untracked
git branch	list all of the branches in your repo -r to list the remote branches -a to see all branches
git diff	show unstaged changes between your index and working directory
git log	list the version history for the current branch
git ls-files	check which files are in your staged area
git branch -vv	check tracking branches

Useful Commands

git init	creates a new git repository, use this command while inside the project folder, this will create a .git folder
git branch <branch_name>	create a branch
git checkout <branch_name>	change to the branch
git branch -u origin/remote_branch	-u = --set-upstream set the tracking branch to be remote_branch
git fetch	pulls in all the commits from your remote but doesn't make any changes to your local files (will overwrite your current files)
git branch -m new_branch_name	rename a branch
git branch --delete branch_name	delete a branch

Useful Commands (cont)

git stash apply stash@{n}	restore a git stash, run git stash list to see the list

More info on setting upstream https://devconnected.com/how-to-set-upstream-branch-on-git/

Origin is the name which git gives to the remote repo that you cloned from

Making Changes

git reset --hard	to discard all local changes (new files created in the local Git workspace that have never been added to the index will remain in the project folder after the hard reset)
git reset <file_name>	undo git add - remove staged version of the file
git reset HEAD~1	undo the prev commit
git revert HEAD	undo the prev commit (for remote branch)
git commit --amend -m "new message"	edit commit msg

To be classified

git rebase <branch_name>	copy our work from the current branch we are on to branch_name
git checkout <branch_name>^	move up one commit of branch _name
git checkout HEAD^ (~4)	move upwards in a commit tree
git branch -f <branch_name> <commit_hash>	reassign a branch to a commit
git stash	takes your uncommitted changes (both staged and unstaged), saves them away for later use