Advance Cheat Sheet for Git	
Author	**Haradhan Pal**
YouTube	**https://www.youtube.com/c/HaradhanAutomationLibrary?sub_confirmation=1**
Version Control Systems (VCS)	
Version control systems are a category of software tools which helps in recording changes made to files by keeping a track of modifications done in the code.	
Examples of Version Control Systems (VCS)	
Helix Core	ClearCase
Git	Mercurial
SVN	TFS
Most popular Git hosting Repositories	
GitHub	Beanstalk
GitLab	FogBugz
BitBucket	Buddy
What is Git?	
GIT is a distributed version control system and source code management (SCM) system with an emphasis to handle small and large projects with speed and efficiency.	
Components of Git	
GitBash	An application for the Windows environment. It is used as Git command line for windows. Git Bash provides an emulation layer for a Git command-line experience. Git package installer contains Bash, bash utilities, and Git on a Windows operating system. Bash is a standard default shell on Linux and mac OS. A shell is a terminal application which is used to create an interface with an operating system through commands. By default, Git Windows package contains the Git Bash tool.
Git GUI	It is a powerful alternative to Git BASH. It offers a graphical version of the Git command line function, as well as comprehensive visual diff tools. User can access it by simply right click on a folder or location in windows explorer. Also, user can access it through the command line by typing command: git gui
Gitk	It is a graphical history viewer tool. It's a robust GUI shell over git log and git grep. This tool is used to find something that happened in the past or visualize user project's history. Gitk can invoke from the command-line. Just change directory into a Git repository, and type: gitk
Basic Git Operations	
Initialize	Pull
Add	Push
Commit	Merge
Important GIT Commands	
git config command is used initially to configure the user.name and user.email. This specifies what email id and username will be used from a local repository. This command sets the author's name and email address respectively to be used with user commits.	
Set User name	git config --global user.name “username”
Set User email address	git config --global user.email “useremailaddress”
Check the setting	git config -list
To create an Alias to Command	git config --global alias.lo "log --oneline"
To Remove an Alias	git config --global --unset alias.lo
To Remove username	git config --global --unset user.name
git init initializes a brand-new Git repository and begins tracking an existing directory. It adds a hidden subfolder within the existing directory that houses the internal data structure required for version control.	
Create a local repository	git init <Repository Name>
Make a local copy of the server repository	git clone <remote Url>
Create new files (say file1.txt. File2.txt and file3.txt) in directory	
Add a single file to staging (Index) area	git add Filename
Add multiple files of a repo to staging (Index) area	git add* git add file1.txt file2.txt file3.txt
After staging files, user can commit them into Git	
Commands to commit changes	git commit -m “commit message"

	git commit -a
	git commit -a -m "commit message"
Status command displays a list of files that user have changed and those user still need to add or commit.	
Display the state of the working directory and the staging area	git status
Command used to view expanded details on	git show <commit id>
Git objects such as blobs, trees, tags, and	git show <options> <object>
git log command is used to find specific commits in user project history- by author, date, content or history	
To show the Git Commits	git log
To show Recent 5 Commits	git log -5
To display the each commit in one line	git log --oneline
To display all commit from certain date	git log --since=2022-01-15
To display all commit till certain date	git log --until=2023-02-18
To display all commit for any user	git log --author="user_name"
To display the modification on each line of a file	git blame <file name>
To display files that have been modified	git log –stat
To display the modified files with location	git log -p
Git fetch command use to downloads branches and tags from one or more repositories	
Fetch the remote repository	git fetch< repository Url>
Fetch a specific branch	git fetch <branch URL><branch name>
Fetch all the branches simultaneously	git fetch –all
Synchronize the local repository	git fetch origin
Working with Git Branch	
Create any branch	git branch <branch name>
To display lists of branch	git branch --list
To delete any branch	git branch -d<branch name>
To delete any remote branch	git push origin -delete <branch name>
To rename any branch	git branch -m <old branch name><new branch name>
To display local and remote branch lists	git branch -a
To display lists of remote branch	git remote show origin
	git branch -r
To checkout to a branch	git checkout <branch_name
To create branch while checkout	git checkout -b <branch_name>
Checkout to a Remote branch	git checkout <remotebranch>
To see difference between 2 branches	git diff <branch1>..<branch2 >
To merge two branches	git merge <branch name>
Merge the specified commit to currently active branch	git merge <commit>
To lists the branches that have been merged into the current branch	git branch -merged
To lists the branches that have not been merged	git branch -no-merged
git push command used to transfer the commits from user local repository to a remote server	
Push data to remote server	git push origin master
Force push data	git push <remote><branch> -f
Delete a remote branch by push command	git push origin -delete edited
Push new branch and its data to remote	git push <github_repository_path> <branch_name>
repository	git push <github_repository_path> <branch_name>
Stashing takes the Temporary stored state of user working directory	
To store the data into stash	git stash save "<message>"
To display the stash list	git stash list
To copy the data into branches	git stash apply <stash#>
To move the data into branches	git stash pop <stash#>
To delete any particular stash	git stash drop <stash#>
To delete the entire stash list	git stash clear

Track changes of Commit	
Track the changes that have not been staged	git diff
To look at the changes to a particular file	git diff <file>
Track the changes that have staged but not committed	git diff --staged
Track the changes after committing a file	git diff HEAD
Track the changes between two commits	git diff <commit1-sha> <commit2-sha>
Git rebase	
Apply a sequence of commits from distinct branches into a final commit	git rebase <branch name>
Continue the rebasing process	git rebase –continue
Abort the rebasing process	git rebase --skip
Git remote	
Add a remote for the repository	git remote add <short name><remote URL>
Change remote	git remote set-url <remote name><newURL>
Check the configuration of the remote server	git remote -v
Fetch the data from remote server	git fetch <Remote>
Show additional information about a particular remote	git remote show <remote>
Rename remote server	git remote rename <old name><new name>
Remove a remote connection from the repository	git remote rm <destination>
Misllaneous important GIT Commands	
git pull origin master	git pull origin master
Pull a file from particular remote branch	git pull origin <branch_name>
To skip from merge conflict	git merge --abort
To remove the file from the work area/staging area	git rm
Remove files from the Git But keep the files in user local repository	git rm --cached
To get back a commit to staging area	git reset --soft <previous_commit id>
To get back a file from staging area to working area	git reset head <file_name>
To get back a commit to work area	git reset --mixed <previous commit id>
Undo the changes	git revert
Revert a particular commit	git revert <commit-ish>
To fix a broken commit	git commit --amend -m "This is user new Git Message"
Author	**Haradhan Pal**
YouTube	**https://www.youtube.com/c/HaradhanAutomationLibrary?sub_confirmation=1**