Git Cheat-Sheet

Install Git

- **$ sudo apt install git** # Debian & Ubuntu

Local repository commands

- **$ git help <command> or $ git <command> –help** # Get a manual page regarding a certain git command
- **$ git init** # Initialize local Git repository
- **$ git add <files>** # Add files to index (staging area to be ready for commit)
- **$ git status** # Check status of the working tree of branches (which files are staging and which are already on stage)
- **$ git commit** # Commit changes in Index (from staging area to local repository)
- **$ git config** # Setting config values to git system (done after intalling git and could be modified later on)
  - **$ git config –list** # List all configuration values
  - **$ git config --global user.name 'Aleksandr Gaisov'** # Configure repository author name
  - **$ git config --global user.email 'aleksgaisov@net-c.com'** # Configure repository author email

Remote repository commands

- **$ git fetch** # Downloads commits, files and refs from a remore repository to local
  - **$ git fetch <name>** # Fetch all of the branches from the repository
  - **$ git fetch <name> <branch>** # Fetch only a specified branch
- **$ git remote** # Lets you create, view and delete connections to other repositories
  - **$ git remote -v** # View the remore connections to other repositories
  - **$ git remote add <name> <url>** # Create a new connection to a remote repository
  - **$ git remote rm <name>** # Remove the connection to the remote repository

  - **$ git remote rename <old-name> <new-name>** # Rename a remote connection
- **$ git push** # Upload content from Local to Remote Repository
  - **$ git push <name> <branch>** # Push the specified branch of local repository to remote (created the named branch at destination)
  - **$ git push <name> –all** # Push all local branches to the specified remore repository
- **$ git pull** # Fetch and download content from a remote repository and immediately update the local repository
  - *The **git pull** command first runs **git fetch**, which downloads content from the specified repository. Then **git merge** is executed to merge the remote content into a new local commit.*
  - **$ git pull <name>** # Fetch the specified remote copy of the current branch and immediately merge it into the local copy
- **$ git clone** # Clone the Remote Repository into Local Directory
  - **$ git clone <url> <local path>**

- *Execute **git pull origin master** to pull the latest project changes from remote repository (if multiple people are working with that repository).*
- *Then run **git push origin master** to push your changer, as well as the updated project from pull command, to the remote repository.*

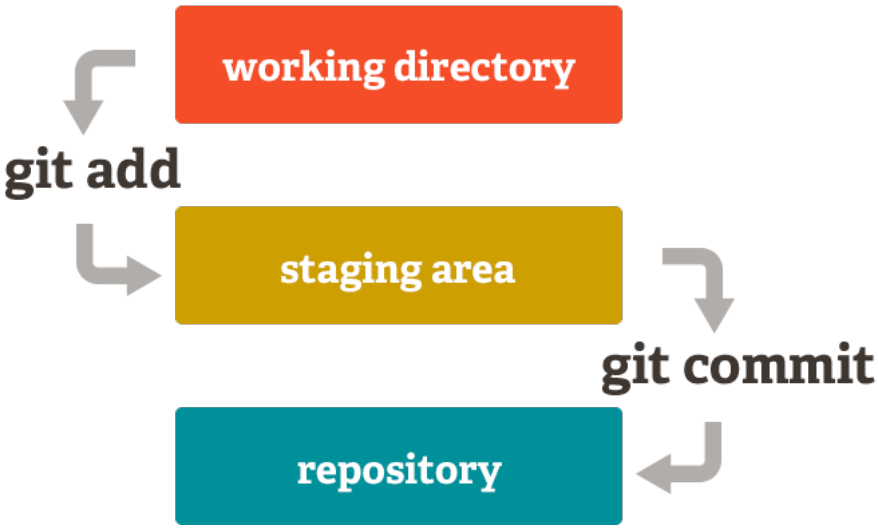

Figure 1: Three states of Git

Examples for working with local repository

mkdir myProject # Create local directory

touch app.py module.py # Create project files in the local directory

echo “# my new app” >> README.md # Create a readme file

git init # Initialize a local repository in the current directory

git add app.py # Add ‘app.py’ to Index

git status # Check the status of the working tree of local repository

git rm --cached app.py # Remove ‘app.py’ from Index (or git reset app.py)

git add . # Add all files in local repository to Index (or git add -A)

echo “some change” >> module.py # Make a change to added file

git diff # See the recent changes within the working directory

git status # Check if we need to add something to Index (‘module.py’ was changed)

git add module.py # Add the updated version of a file to Index

git commit -m ‘Initial Commit’ # Commit all files from Index to local repository with a appropriate comment

git log # See commits logs

touch .gitignore # Create a file to list project files that are to ignored during the commit command (‘.gitignore’ will also be ignored)

(It is considered a good practice to always include the ‘.gitignore’ to the commit, for other people who would want to work on your repository)

touch log.txt # Create a file that we do not want to commit

echo “log.txt” >> .gitignore # Now ‘log.txt’ will be ignored during the commit command execution (you can also add whole directories ‘/dirName’)

git branch mybranch # Create a new branch (you are in master branch by default)

*(To push the new branch to remote repository use **git push -u origin mybranch**)*

git branch # List all existing brances for current local repository

*(Use **git branch -a** to also see all currently connected remote branches)*

git checkout mybranch # Checkout from ‘master’ and login to ‘mybranch’ branch

touch module2.py && git add . && git commit -m ‘New commit’ # Create a new file, add all files from master to Index and commit it to ‘mybranch’ branch

git checkout master # Checkout from ‘mybranch’ and login back to ‘master’

*(Also run **git pull origin master** if you need the get the latest changes from a remote repository)*

git merge mybranch # Merge the commit from ‘mybranch’ with current branch ‘master’

git branch –merged # See all merged branches (check if the merge process went succesfuly)

*(Run **git push origin master** if necessary)*

git branch -d mybranch # Delete no loger needed branch

*(To also delete associated branch from repository use **git push origin –delete mybranch**)*

Examples for working with remote repository

Example 1

(continue from local repository)

git remote add origin <https://github.com/…> # Add a remote repository with a link on GitHub

git remote -v # View all remote repositories (origin by default)

*(Use **git pull origin master --allow-unrelated-histories** if your remote repository already contains some files)*

git push -u origin master # Push current local repository ‘myProject’ to the remote repository ‘myProject/origin’ on master branch

*(**-u** makes the current local branch (here ‘master’) to be later associated with the same-named brach on remore repository (remote ‘master’)).*

login & *password*

touch module3.py

git add .

git commit -m ‘module3 added’

git push

git clone <https://github.com/...> # Clone the existing remote repository to the local directory

Example 2

(just a workflow reference; we start on a master branch)

git branch newbranch

git checkout newbranch *(Making some changes to the project files...)*

git status

git add -A

git commit -m “Some changes”

git push -u origin newbranch *(Assuming that all the test went well and we are ready to merge...)*

git checkout master

git pull origin master

git merge newbranch

git push origin master *(Deleting unnecessary branch...)*

git -d newbranch

git push origin –delete newbranch