|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Cues**NoteGem Horizontal Line | **Notes**  NoteGem Horizontal Line | |  | * **ls:** list files in current directory * **cd:** go to directory, quotes for spaces * **ls-la:** display current directory and hidden files * **mkdir \_\_\_:** create new directory named \_\_\_\_      * **Output:**   + **cat *<file>*:** output file contents   + **less <*file>*:** output file contents with pagination etc     - last line will contain either file name or colon     - **space:** one page forward     - **b:** one page backward     - **q:** quits less   + **head *<file>*:** output first 10 lines of file   + **cmd > *<file>*:** direct cmd output into file   + **cmd1 | cmd2:**  append cmd1 output to cmd2 output   + **clear:** clear command line window * **Files:**   + **rm *<file>*:** delete file   + **rm -r directory**: delete directory   + **rm -f *<file>***: force delete file (-r to force delete directory)   + **mv fileold filenew:** rename file   + **cp file directory:**  copy file to directory/overwrite   + **cp -r directory1 directory2:** copy directory/overwrite   + **touch file:** update file access & mod time & create file if not exist * **Permissions**   + **chmod 755 *<file>:*** change file permissions to 755   + **chmod -r 600 directory:** change directory permissions to 600   + **chown user:group *<file>*:** change file ownership to user/group   + **find directory -name *"<file>":*** find **"*<file>*"** files in directory     - \* = wild card search     - -i = case sensitive search     - ; = combines commands     - && = runs only if previous successful   + **grep "text" *<file>*:** outputs all text occurrences in file   + **grep -rl "text" directory:** search for files containing "text" in directory      * **Network**   + **ping host: ping host and display status**   + **whois domain: output domain whois info**   + **curl -o url/to/file: download file via https/ftp**   + **ssh username@host: ssl connect to host**   + **scp *<file>* user@host: /path/remote: copy file to remote host**      * **Processes**   + **ps ax: output currently running processes**   + **top: display live info about currently running processes**   + **kill pid: quit process with ID \_\_\_**      * **Help**   + **man command**   + **command -help** * **File Permissions**   + **# | # | #**   + **owner | group | everyoneelse**   + **add together for each digit:**     - **4 = access/read**     - **2 = modify/write**     - **1 = execute** * **ctrl + A: beginning of line** * **ctrl + E: end of line** * **ctrl + K: deletes all characters after** * **ctrl + U: deletes all characters before** * **tab: autocompletes when unambiguous** * **arrow key up/down: cycle through previous commands**   + **hit tab two times to view possible matches when ambiguous** * **~: home folder /users/username** * **whoami: your username** * **git clone ssh://user@domain/repo.git: clones a repo** * **git init: create new repo** * **Local Changes**   + **git status: changed files in working directory**   + **git diff: changes to tracked files**   + **git add .: add all current changes to next commit**   + **git add -p file: add file's changes to next commit**   + **git commit -a: commit all local changes in tracked files**   + **git commit: commit all previously staged commits**   + **git commit --amend: change last commit**   + **git commit -m "message": commit all current changes with an accompanying message** * **Commit History**    + **git log: show all commits starting with latest**   + **git log -p file: show file's changes over time**   + **git blame file: who changed what and when in file** * **Branches & Tags**   + **git branch -av: list all existing branches**   + **git checkout branch: switch head branch**   + **git branch newbranch: create new branch with current head**   + **git checkout -track remote/branch: create new branch with remote head**   + **git branch -d branch: delete local branch**   + **git tag tagname: mark current commit with tag** * **Update & Publish**   + **git remote -v: list currently configured remotes**   + **git remote show remote: show info about remote**   + **git remote add shortname URL: add new remote repo**   + **git fetch remote: download changes, no integration**   + **git pull remote branch: download changes, yes integration**   + **git push remote branch: publish local changes on remote**   + **git branch -dr remote/branch: delete remote branch**   + **git push tags: publish tags** * **Merge & Rebase**   + **git merge branch: merge branch into current head**   + **git rebase branch: rebase current head onto branch**   + **git rebase -abort: abort rebase**   + **git rebase -continue: continue rebase after resolving conflicts**   + **git mergetool: solve conflicts**   + **git add resolvedfile: manually resolve conflict and mark as resolved**   + **git rm resolvedfile: manually resolve conflict and mark as resolved**      * **Undo**   + **git reset -hard HEAD: discard local changes in working directory**   + **git checkout HEAD file: discard file's local specific changes**   + **git revert commit: revert a commit (produce new commit w/ contrary changes)**   + **git reset -hard commit: reset head to previous commit and discard all changes since**   + **git reset commit: preserve all changes in unstaged commits??**   **git reset -keep commit: preserve uncommitted local changes???** | | What are best practices of Git?                                            Why are branches used in Git?                      What's the default branch?  For what reasons do you want separate branches?    What's tagging for?    How do view one line commit?                  What are a repository's trees?              What is the git flow process? | * **commit: wrapper for related changes** * **fix two bugs = two separate commits** * **small commits make it easier to understand changes and roll back if wrong** * **commit often** * **only commit code when complete**   + **do not commit half-done work**   + **split into logical chunks**   + **don't commit just to commit**   + **if you need a clean working copy or to commit unfinished work, consider using stash** * **test code before committing, no side effects** * **version control is not backup** * **write good commit messages (50 characters)**   + **include blank line at bottom of files to separate from body**   + **motivation for the change**   + **how is it different**   + **present tense** * **use branches**   + **helps avoid mixing up different lines of development**   + **use for new features, bug fixes, ideas** * **agree on workflow**   + **select workflow: long-running branches, topic branches, merge or rebase, git-flow**   + **choice factors: project, overall development, personal preferences, deployment workflows** * **\*\*\*INSERT PICTURE OF GIT WORKFLOW** * **branch is used to develop features isolated from each other** * master is default branch * separate branches = separate features * different people = separate branches * tagging = good for software releases   + git tag name 1st10commitID * one line per commit = git config format.pretty oneline * colorful git output = git config color.ui true * use interactive adding = git add -j * can host websites free on github * set username for every repo on computer:   + git config --global user.name username   + git config --global user.email email * repository has three trees   + working directory holds actual files   + index or staging area   + head points to last commit you made * **can store any files in repository** * **Git Process:**    + **fork repo**   + **clone repo (git clone url)**   + **add branch**   + **add collaborator**   + **make your changes**   + **add one file or all changes with git add . to stage changes**   + **git commit -m "message" to commit changes with a message**   + **git push to publish local commits**   + **check status with git status** | |
| |  | | --- | | **Summary** | |