GIT Use Cases

//1. first commit:
1. go to project directory
2. git init
Initialized empty Git repository in
/Users/naveenautomationlabs/Documents/workspace/March2021EvePOMSeries/.git/
3. one .git (hidden dir) will be created
4. git remote add origin <repo url>
5. git status
6. add .gitignore file
7. git add .
8. git commit -m "reason"
9. git push origin master
10. go to the repo url and check the code at remote side

//2. new team member:
1. get the repo url
2. create a directory in your local : C/D drive
3. clone the repo in your local: git clone <repo url>
4. import this project into Eclipse/IntelliJ
5. start looking into the project: run/debug it/check the code
6. add/update/delete some files
7. git add <files>
8. git commit -m "reason"
9. git push origin master
10. go to the repo url and check the code at remote side

//3. Local branching Process:

1. git branch -- point to master only
2. create/cut the brach from master: git branch cart
3. switch to the cart branch: git checkout cart
4. git branch : it should point to cart
5. start working on cart branch (make sure in eclipse cart is reflected)
6. add/update/delete the files/code in your working copy (eclipse)
7. git status
8. git add <files>
9. git commit -m <reason>
10. git push origin cart
11. changes should be reflected at remote side : a new cart brnach should be created at remote

//4. PR (Pull Request + Merge):

12. Raise a PR to the reviewers with summary
13. Rev will check the code and put the comments and PR is not approved
14. Req will read those comments and will update the code in WC (eclipse)
15. Req will add--> commit --> push (git push origin cart)
16. PR is updated with latest car page changes
17. Rev will again check the latest changes as per the given review comments
18. This time PR is approved
19. Req/Rev will merge the PR to master (Cart --> Master)
20. Check the master branch (remote) : make sure cart changes are refelcted
21. git checkout master (point to master branch)
22. Req has to take the latest pull : git pull origin master
23. In local, Master is updated with latest Pull (cartpage)

//5. A new assignment process (with exisitng team members):

1. team member has to take the latest pull from master (whenever there are chnages in master - remote): git master---> git pull origin master
2. cut the branch: git branch <name>
3. follow #3 and #4 processes

//6. Merge Conflict:

1. Naveen is making some changes in local --> demopage.java--> demo() -- master branch
2. Shailesh also making some changes in local --> demopage.java--> demo() -- master branch
3. Shailesh will merge the code to master after PR
4. Naveen will try to take the latest pull:git pull origin master
5. PULL will be aborted
6. Naveen has to move the code to stash: git add <file> : add to stage and then stash

git stash

7. then take the latest pull: git pull origin master
8. Remote changes will be reflect in Naveen's local WC
9. Naveen has to take the stash code back to WC: git stash pop
10. Merge conflict will happen

```
<<<<<<<<upstream


    remote code
    =======
    >>>>>>>>>downstream stash
    local code
```

11. communicate and resolve it, accept the respective changes (local/remote)
12. Naveen has to push the code to master (if Naveen's local changes are accepted)
13. Shailesh has to take the latest pull

//4. Reset:

1. do some wrong changes at remote (wrong push)
2. git log --oneline --> it will give you the commit history
3. copy the (N-1) commit hash code (ID)
4. and use reset: git reset --hard 66744b1 (ID)
5. finally do the force push to the remote side

git push -f origin master

6. that wrong file should be deleted from remote side(reverted back to the previous commit)

//.gitignore file:

```
allure-results/
screenshots/
screenshot/
test-output/
build/


##############################
## Java
##############################
.mtj.tmp/
*.class
*.jar
*.war
*.ear
*.nar
hs_err_pid*
activityLog.log
##############################
```

```
## Maven
###############################
target/
pom.xml.tag
pom.xml.releaseBackup
pom.xml.versionsBackup
pom.xml.next
pom.xml.bak
release.properties
dependency-reduced-pom.xml
buildNumber.properties
.mvn/timing.properties
.mvn/wrapper/maven-wrapper.jar
```

```
###############################
## IntelliJ
###############################
out/
.idea/
.idea_modules/
*.iml
*.ipr
*.iws
###############################
```

```
## Eclipse
##############################
.settings/
bin/
tmp/
.metadata
.classpath
.project
*.tmp
*.bak
*.swp
*~.nib
local.properties
.loadpath
.factorypath


## OS X
##############################
.DS_Store
```