1 git --version => For check the install version on local system

2 mkdir folderName => for create new directory or folder

3 cd folderName/ => for move inside/others folder

4 touch fileName.java secondFileName.xyz => for create file

5 git init => initialize current folder as local git

6 ls => get list of /folderfile into the current folder

7 ls -la => get list of folder/file into current folder with details

8 ls .git => for get all hidden file/folder which created by git

9 whoami => to get current login user details

10 echo $HOME => to get full path/url

11 ls -ls /root =>

12 ls -la /root

13 git config --global user.name nameOfUsers => for get the global user or author name

14 git config --global user.email [globaluser@gmail.com](mailto:globaluser@gmail.com) => for set global user/author gmailid

15 cat /root/.gitconfig

16 ls -la /root

17 cat /root/.gitconfig

18 git status => for get the status of change file

19 git add deposit.java => for add file

20

21 git rm --cached deposit.java => for delete file

22

23 git add . => for add all file which are created in current folder

24

25 git commit -m "adding deposit file " deposit.java => Commit the deposite.java into staging/local are

26 git log => display the list of commit log

27

28 git commit -m "adding withdraw file " => Commit all change inside the current folder

29

30 git log --oneline => for display list of log into singleline

31 touch internet.java => create file

32

33 cat .git/config => get all login users

34 git config --local user.name localuser => for set local user/author Name

35 cat .git/config

36 git config --local user.email [localuser@gmail.com](mailto:localuser@gmail.com) => For set local user/aithor emailid

37 cat .git/config => to display the login user(local & global)

41 git commit -m "internet.java file is added"

44 history => for get all previous history command list

51 git reset HEAD~1 => for reset last one commit

52 git log --oneline => for display onlineline log

53 git status => for check the status of current/local git repo

54 git add . => All new file

55 git commit -m "tmp.java is added" => commit all tracked file into local repo

56 git log --oneline => display the log into oneline

57 git reset --soft HEAD~1 => for rset/remove last 1 in soft mode

58 git status

59 git commit -m "temp.java"

60 git log --oneline

61 git reset --hard HEAD~1 => for rset/remove last 1 in hard mode

64 git commit --amend –m "New Message"=> For change/update the commit message

65 git --help => For get help list

66 git commit --help => Get help

67 git commit --amend –m "new message" => for update the

69 cat .git/config => For get all users details

70 git remote add origin https://github.com/onlineTrainingguy/BankingApp.git

71 cat .git/config

72 git branch

73 git branch -M main

74 git branch

75 git push -u origin main

76 ls

77 git fetch

78 git merge origin/main

79 ls

80 git pull origin main

81 ls

82 cd ..

83 mkdir notes

84 cd notes/

85 git clone https://github.com/onlineTrainingguy/DevOpsNotes.git

86 ls

87 cd DevOpsNotes/

88 ls

89 cd ..

90 rm -r notes/

91 ls

92 cd BankingApp/

93 ls

94 git tag release-1

95 ls

96 touch release-2.java release-2.1.java => for create multiple file

97 git add .

98 git commit -m "Release2"

99 ls

100 git tag release-2

101 touch release-3.java release-3.1.java

102 git add .

103 git commit -m "release3"

104 git tag release-3

105 git tag

106 git checkout release-1

107 ls

108 git checkout release-2

109 ls

110 git checkout release-3

112 git push --tags => Tags are basically used with release or any others case for final commit and know about separate details

113 history => to display the list/history of all previous command

114 git push: For push the commited code into live/remote server

114 git push --set-upstream origin master => The current branch master has no upstream branch.

To push the current branch and set the remote as upstream, use

123 cd BankingApp/

124 ls

125 git tags

126 git tag

127 git tag -d release-2

128 git tag -d release-3

129 git checkout main

130 ls

131 git log --oneline

132 git reset --hard HEAD~2

133 git log --oneline

134 git push origin main

135 git branch

136 git branch dev1

137 git branch

138 git checkout dev1

139 git branch

140 ls

141 git log --oneline

142 git checkout -

143 git branch -D dev1

144 git branch

145 git branch dev1

146 git checkout dev1

147 git checkout -

148 git branch

149 git branch dev2

150 git branch

151 git push origin dev1

152 git branch

153 git push origin dev2

154 git checkout dev1

155 git pull origin dev1

156 git branch

157 ls

158 git checkout main

159 git merge dev1

160 ls

161 git push origin main

162 git checkout dev2

163 git pull origin dev2

164 ls

165 git checkout -

166 git merge dev2

167 ls

168 git log --oneline

169 git push origin main

170 git checkout dev1

171 git merge main

172 ls

173 git checkout dev2

174 git branch

175 git merge main

176 ls

177 git checkout main

178 git push origin dev1

179 git push origin dev2

180 git branch

181 git checkout dev1

182 ls

183 echo "This file is edited by dev1" > deposit.java

184 git add .

185 git commit -m "dev1 modified deposit file"

186 cat deposit.java

187 git checkout dev2

188 cat deposit.java

189 echo "This file is edited by dev2" >deposit.java

190 git add .

191 git commit -m "dev2 modified deposit file"

192 git checkout main

193 git merge dev1

194 cat deposit.java

195 git merge dev2

196 git mergetool --tool=vimdiff

197 cat deposit.java

198 ls

199 cat deposit.java.orig

200 rm deposit.java.orig

201 git status

202 git commit -m "merge confilict is resolved"

203 cat deposit.java

204 git checkout dev1

205 git merge main

206 cat deposit.java

207 git checkout dev2

208 git merge main

209 cat deposit.java

210 git branch

211 git checkout main

212 git --help

213 git log --oneline

214 git diff 68a170a 4371dac

215 cd ..

216 mkdir mr

217 cd mr

218 git init

219 touch m1 m2 m3

220 git add .

221 git commit -m "m1" m1

222 git commit -m "m2" m2

223 git commit -m "m3" m3

224 git log --oneline

225 git branch feature

226 git checkout feature

227 ls

228 git log --oneline

229 touch f1

230 git add .

231 git commit -m "f1"

232 git log --oneline

233 git checkout -

234 touch m4

235 git add .

236 git commit -m "m4"

237 git checkout -

238 git merge master

239 ls

240 git log --oneline

241 cd ..

242 mkdir rm

243 cd rm

244 touch m1 m2 m3

245 git add .

246 git init

247 git add .

248 git commit -m "m1" m1

249 git commit -m "m2" m2

250 git commit -m "m3" m3

251 git branch feature

252 git checkout feature

253 ls

254 touch f1

255 git ad .

256 git add .

257 git commit -m "f1"

258 git log --oneline

259 git checkout -

260 touch m4

261 git add .

262 git commit -m "m4"

263 git checkout -

264 ls

265 git log --oneline

266 git reabse master

267 git rebase master

268 ls

269 git log --oneline

270 history

GIT MY Work

1. Create repository on git with name : IntellipaatStudyMaterial and URL: <https://github.com/abhishekraics001/IntellipaatStudyMaterial.git>
2. Create one local folder with same name (IntellipaatStudyMaterial)
3. Initalize IntellipaatStudyMaterial folder in local system as git repogitory by typing **git init**
4. Create & Add some file into this folder (for creating file through command you can use **touch xyz.java**)
5. Check the file status inside local folder(whatever we have done the chnages & which file, added or chnages deleted, modify by typing **git status**

Add all newaly created file which are inside untracked area to staging area in local repository by typing **git add xyz.java** or **git add .**

1. Commit all change (which all are inside staging area) by typing to save it into local repogitory **git commit –m”your commit message”**
2. Note:
   1. Make sure to add authors details
      1. Add Author Name: **git config --global user.name “Abhishek Rai”**
      2. Add Author Email ID **git config --global user.email** [**abhishekraics001@gmail.com**](mailto:abhishekraics001@gmail.com)
   2. if you have already respogitory on git with old codebase then you 1st need to get clone of your repository at before the 1st step

**git clone https://github.com/abhishekraics001/IntellipaatStudyMaterial.git**

* 1. If you created newaly repo on git then you 1st need to run below command

**git remote add origin** [**https://github.com/abhishekraics001/IntellipaatStudyMaterial.git**](https://github.com/abhishekraics001/IntellipaatStudyMaterial.git)

* 1. First time it’s ask pass to push details on remote server
     1. Go github account details Profile => Setting=> Developer Setting => Parsonal Access Tocken => Generate new Tocken <https://github.com/settings/tokens?type=beta>
     2. Copy the tocken and put it as password

1. Push all local/staging area file/chnages into remote repository by typing **git push --set-upstrem origin master**
2. Get the update code from remote repositiry by typing **git pull**

10.Git the comit log by typing **git log or git log –oneline**

**11.** create new local branch in same resposity branch by typing **git branch DevTesting**

**12.** Get all branch list inside local or remote by typing **git branch**

**13.** How to check active/current branch**: It’s simply shoing \* with green color**

**14.** Swich from one brach to another branch by typing git branch **git switch DevTesting** or **git checkout branchName**

**15.** Add/Do The chnages, Commit and push your branch on remote

**Git add .**

**Git commit –m”** **Add New DevTesting Branch”**

**Git push –u origin DevTesting**

**17.** Get Checkout from remote respository **git checkout master**

**18**. Merge your DevTesting Branch into/to Master Branch by typing

**git merge DevTesting**

**git push**

**19.** If you want to change the message of last commit

**git commit --amend -m"Changing commit message: Add chnages for RestHead"**

**20.** Reset Your last or any **git reset HEAD~1**

**21.** Clear/clean the local **caches git clean –n**

**22. In case of Conflict fllow bellow step to get it resolved:**

**How conflict happen:** Supposes you have one respository and now inside **master** branch & there is one file like **ABC.text**. After that you just create two more branches from master **Dev1** and **Dev2. 1st** you checkout **Dev1** branch and do some chnages inside the **Abc.text** file & Add + Commit it inside the same(Dev1) branch and after that you checkout **Dev2** branch alos do some chnages insdie **Abc.text** file & Add + Commit it inside the same(Dev2) branch. After that we checkout the **master** branch and marge Dev1 branch with master(will done successfulyy) but when we try to mager Dev2 branch with master then conflict will accoure

touch A.text

git a .

git add .

git commit -m"Adding A.text file into master branch"

git branch Dev1

git branch Dev2

git branch

git checkout Dev1

nano A.text

git add .

git commit -m"Modifinig A.text file into Dev1 Branch"

git checkout Dev2

nano A.text

git add .

git commit -m"ModifingA.text file into Branch Dev2"

checkout master

git checkout master

git merge Dev1

git merge Dev2

git mergetool --tool=vimdiff

git commit -m"Resolving conflick and masger into Master"

**How to resolved**

* **git mergetool --tool=vimdiff** => Editor will open then => For view the conflict file content
* Review the connect whare conflick is happed and edit with atual onc and type below command multiple time(4/5 time)
* **Esc** => press Escape keyword button
* **:wq** => for save & quit
* Then type the commit command to done the changes **git commit –m”Conflick resoluton done”**