

Whenever we create a project on GitLab, we should create a fork.

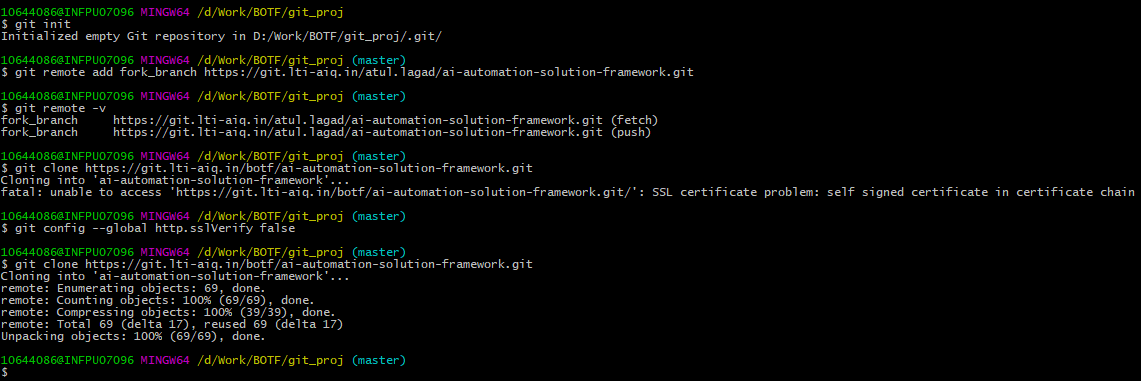
**GitLab**

Operations

* We can create it on web browser.
  + Open a project
  + Click on “Fork”
  + Select your id

First step is to clone

* Open Git Bash
  + $ git init
  + $ git remote add fork <https://git.lti-aiq.in/atul.lagad/ai-automation-solution-framework.git>
  + $ git remote –v
  + $ git config --global http.sslVerify false
  + $ git clone <https://git.lti-aiq.in/botf/ai-automation-solution-framework.git>



How to push the code

* $ git status

(You will get to see all the changes made in your directory)

To add specific files,

* $ git add <filename1> <filename2>

To remove files from git add

* $ git reset <filename1> <filename2>
* $ git status

(to check if we are pushing the right files)

To commit

* $ git commit -m'workspace APIs, models'

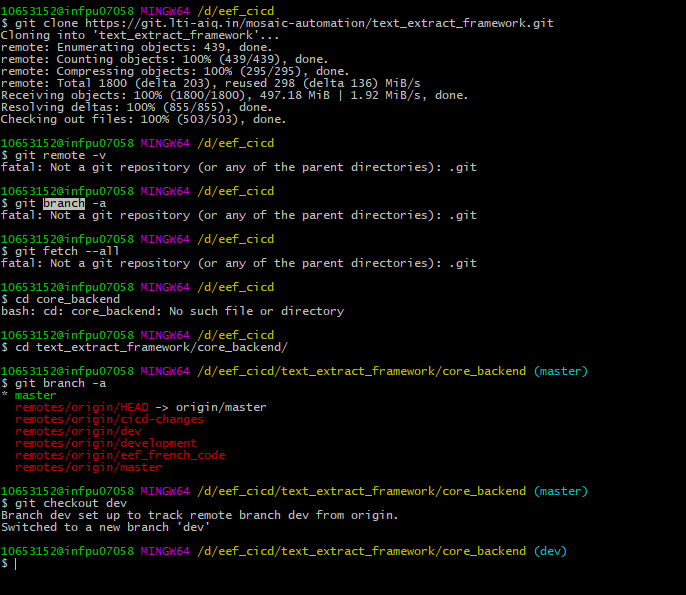
Check if fork is created against your name

* $ git remote –v

Push the code to the fork

* $ git push fork master

To take pull of a specific branch:



git config --global http.proxy http://10639518:JanU+00402019@proxycluster.puneodc.lntinfotech.com:8002

U+0040 (for @)

For pulling code:

->git remote -v

->git fetch upstream

->git reset --hard (if we don't want our changes to get added)

->git git upstream/master

->git push origin master

For pushing code:

->git diff(optional)

->git status

============================

to unset proxy

git config --global --unset http.proxy

to create branch

$ git checkout -b "Ai\_tutor\_updated\_code"

$ git status

$ git add core\_backend/manager/ai\_tutor.py

$ git commit -m "Ai tutor changed code with swagger"

$ git push origin Ai\_tutor\_updated\_code

to create branch

git branch -b qc\_model\_branch

to rename branch

git branch -m qc\_model\_branch

switch to new branch

$ git checkout -b "qcmodel"

switch to existing branch

$ git checkout "qcmodel"

->git add .

->git status

->git commit -m "Message"

->git push origin master

git rebase -i

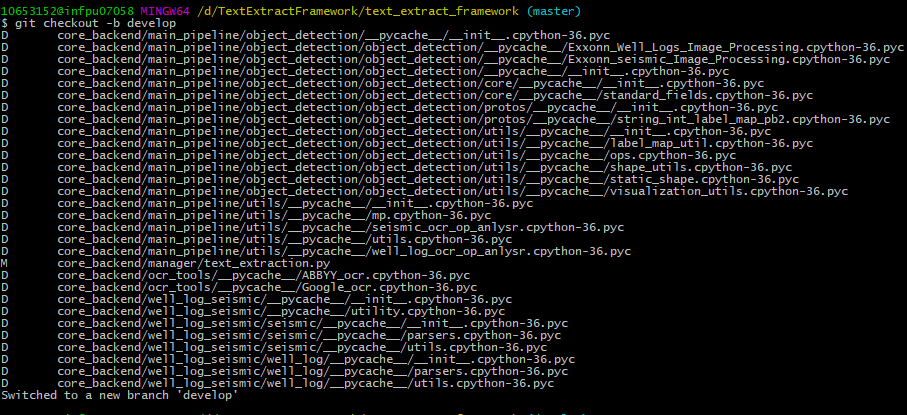
============================

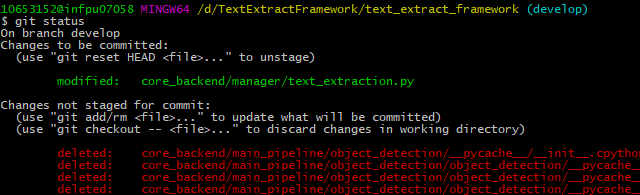
git fetch upstream

git rebase -i upstream/master

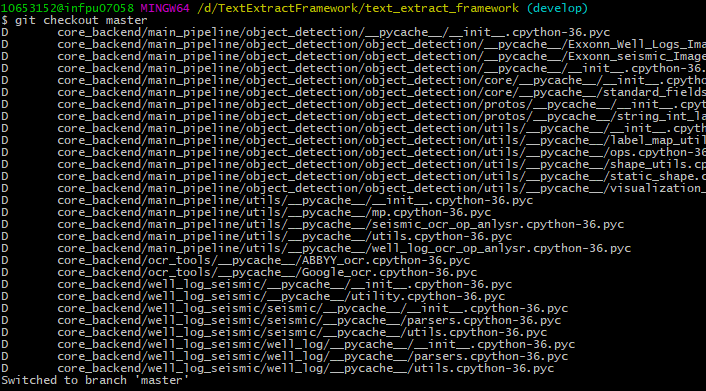
git rebase abort

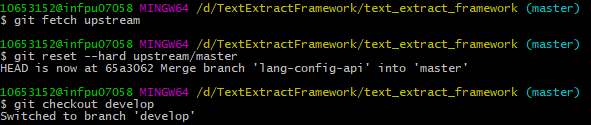
1. Created a new branch named “develop” in my fork

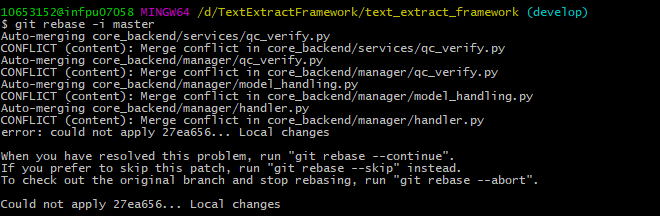




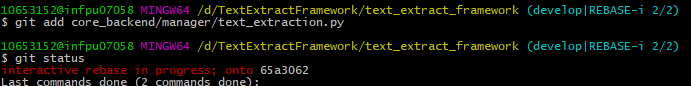
1. Git add
2. Git commit
3. Git checkout master

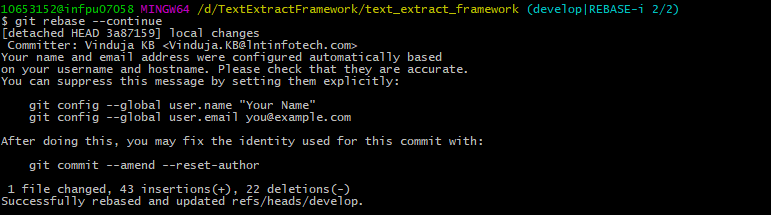


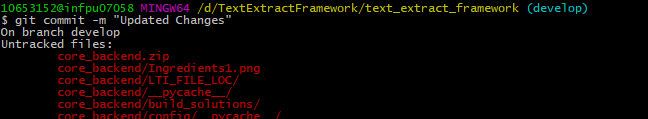


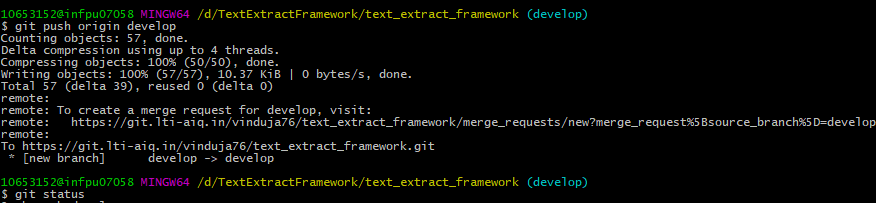


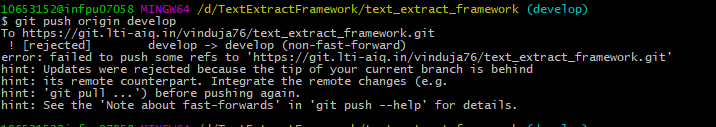
Resolve conflicts if any











git config --global http.proxy <http://10653152:OreoU+0040123@proxycluster.puneodc.lntinfotech.com:8002>

**from sticky::::::::::::::::**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*GIT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

git status(develop branch)

git add(develop branch)

git commit -m "" (develop branch)

--------------------------------------

git checkout master

git fetch upstream

git

--------------------------------

git checkout develop

git rebase -i master

-----------Resolve conflicts if any

git rebase --continue

git commit -m ""

git push origin develop

==============================

==============================

revert git add. ---------> git reset

===========================

Squashing commits:::::

git

git rebase -i master

edit the file by insert (pick a

squash b)

save and quit

===========================

Oreo@1223

For pulling code:

->git remote -v

->git fetch upstream

->git reset --hard (if we don't want our changes to get added)

->git git upstream/master

->git push origin master

For pushing code:

->git diff(optional)

->git status

============================

to unset proxy

git config --global --unset http.proxy

to create branch

$ git checkout -b "Ai\_tutor\_updated\_code"

$ git status

$ git add core\_backend/manager/ai\_tutor.py

$ git commit -m "Ai tutor changed code with swagger"

$ git push origin Ai\_tutor\_updated\_code

to create branch

git branch -b qc\_model\_branch

to rename branch

git branch -m qc\_model\_branch

switch to new branch

$ git checkout -b "qcmodel"

switch to existing branch

$ git checkout "qcmodel"

->git add .

->git status

->git commit -m "Message"

->git push origin master

git rebase -i

============================

git fetch upstream

git rebase -i upstream/master

git rebase abort

GIT COMMANDS KATAKODA:

git log --pretty=format:"%h %an %ar - %s"

While git log tells you the commit author and message, to view the changes made in the commit you need to use the the command git show

Like with other commands, by default it will show the changes in the HEAD commit. Use git show <commit-hash> to view older changes.

Use the command git log --grep="#1234" to find all the commits containing #1234