unstage ⟶ git rm --cached 1.txt
⟶ git span
⟶ git span 1.txt ⟵ redeout

Git is a version control sym for tracking changes
in computer files & co-ordinating work on those
files amoung multiple people. Used for src-code
mgmt in s/w dvv. and to track changes in any
set of files

    ↳ creat Repository (init ↳ Synching
    ↳ make change) [Track status]   Repositorries
    2) Parallel Development

> mkdir devops   ·> git status
> cd devops   1-txl } red
> gedit 1-txt   2- txt } untracked
> gedit 2-txt   > git add.   ↳stage files
              space ⟶ all flex to track
> ls
  1-txt 2-txt   > git status
             newfile : 1-txt
> git init   new file : 2-txt
  initialized empty Git repository /devops/.git/

// commit to repository

> git commit -m 'first comit adding 1-txt and 2-txt'

> git status
   Nothing to commit.

> gedit 3-txt _____

> git status
                    3-txt ← red
   (nothing add to commit but untracked files present).

> nano .gedit 2-txt ⟶ make changes

> git status.
         modified : 2-txt
         untracked 3-txt

// Sync repository   git add origin.

/devops > git remote add origin. "____https:g .git"

   > git push origin master

username ?
password ?
   done.

≥ Refresh githup.

> git status

> git add 2-txt

> git status. modified : 2-txt ✓green

                        3-txt ←—red

└git hub
   ↓
new reposity
└repo name
   (publu
   (create)

3.txt ←─ed

> git commit —m "modified 2.txt"
> git push origin master ────────────→ to production.
> git add.   2r. txt with
                                                 commit
> git commit —m " adding 3.txt "   modify
> git push origin master
usernam :
password :   . 3.txt added
                                               to github.

/devops > git status
[git clone to download remote reposivy to
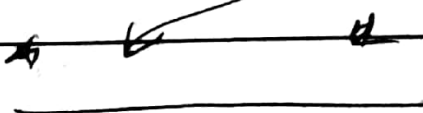   local system.

                                             go to github
/devops > cd   (Clone order)
> ~~cd test~~   clone
> mkdir test
> cd test
> git clone
~/test> ls

devops-course .

```
                    cd devopscourse
tes/devopscourse:   gedit 4.txt
          > git add .
          > git commit -m "adding clone file "
          > git push origin master.

          > cd.
          > mkdir hello
          > cd hello
hello > git init
          > gedit helloworld.txt
          > git add .
          > git commit -m "adding file "
          > git push origin master

          > cd
          > cd test
/test > git pull origin master
          nor git reps)

          > cd
          > ls
       devops .
/devops > git pull origin master
          1 file change, 1 inversion
```

```
> ls.
  1.txt    2.txt    3.tx    4.txt

/devops  > cd
         > ls

devops     hello   test
         > cd test.

   test > ls.
       devops -course.

       > cd devops-course

   test              folder devops.

>ls
1  2  3  4              >gedit 5.txt

>git pull origin master  >git add.

>ls                      >git commt -m
·1  2  3  4  5-            " test "

                         >git push origin
                          master

                         > ls

                         1 2 3 4 5

                         github

                         (5).
```

gitpull
[used to pull latest
changes from
repository.

git clone :- only
work inside
an initialized
git repository]

> all ready
working on
clone repository
I want to pull
latest changes.
that other might
have pushed to
remote repository)

Parallel Development

Use branches to handle workspace of multiple developers

git branch < name-of-new-branch >

to dele   git branch -D < branch . name >

to switch to new branch   git checkout < branch-name >

> cd devops

/devops > git branch feature 1

> git branch.

feature 1

* master

> git branch -D feature 1

> git branch feature 1

> ls

1   2   3   4 5

> git checkout feature 1

switched to branch feature 1

> ls

1   2   3   4   5

> gedit 6. txt.

> git add .

> git commit -m " added to feature

branch"

> ls

(1   2   3   4   5   6)   6 in feature
                                        branch
> git branch

* feature1

master

> git checkout master

switch to mas

> ls   → only 5

(1   2   3   4   5)

> git status

on branch master

> git checkout feature1

> git status

on branch feature 1

> git push origin feature 1   branch master
                                          ↓
> git log   branch feature
                                        1
                                   → 6.txt.

[log for every
   commit in
   repository]

git stash

∴ save work without committing code.

(Helpful when wer want to switch branches but do not want to save work to git repository

[To stash your staged files w/o committing type "git stash"

stash untracked files git stash -u once back and want to retreive working type git stash pop.

> git branch
    feature 1
  * master.

> git checkout feature 1
> gedit 7.txt   (stash file check)
> ls   → not commited   staged
1 - 6, 7.txt   ↓ means
> git checkout master   commit
> ls
1, 2. 6, (7.txt) → not tracked and not part of any branches

it is saved

> git checkout feature1
> git add .   [Now staged but not
> ls   commited]

|---.   7 txt

> git checkout master
> ls

≠ |. - - - (7.txt) can still see 7 as its supposed
                                        to be in feature1
> git checkout feature1   only
> gedit 5.txt   (change connd
> git add.   [stage]
> git checkout master
      M   5.txt
      A   7.txt

> ls

|---. .5, 7.

> git checkout feature1
   M. 5.txt
   A   7.txt
> git stash
     (saved working
        directory
> git checkout master
   7.txt not visible anymore.

```
> cat 5.txt
test file
```

get
back
to work
in
featur

```
> git checkout feature 1
> ls
1 - - - . 6.
> cat 5.txt
test file
> git stash pop        [restore to
> ls                    working stash
1 - - 6 . 7.txl
> cat 5.txt
test file 2
> git status
new file 7.txt
modified : 5.txt
> git add .
> git commit -m "committed
stashed changes"
> clear.
```

```
> ls
1 - - - - 7.txt
> git status
```