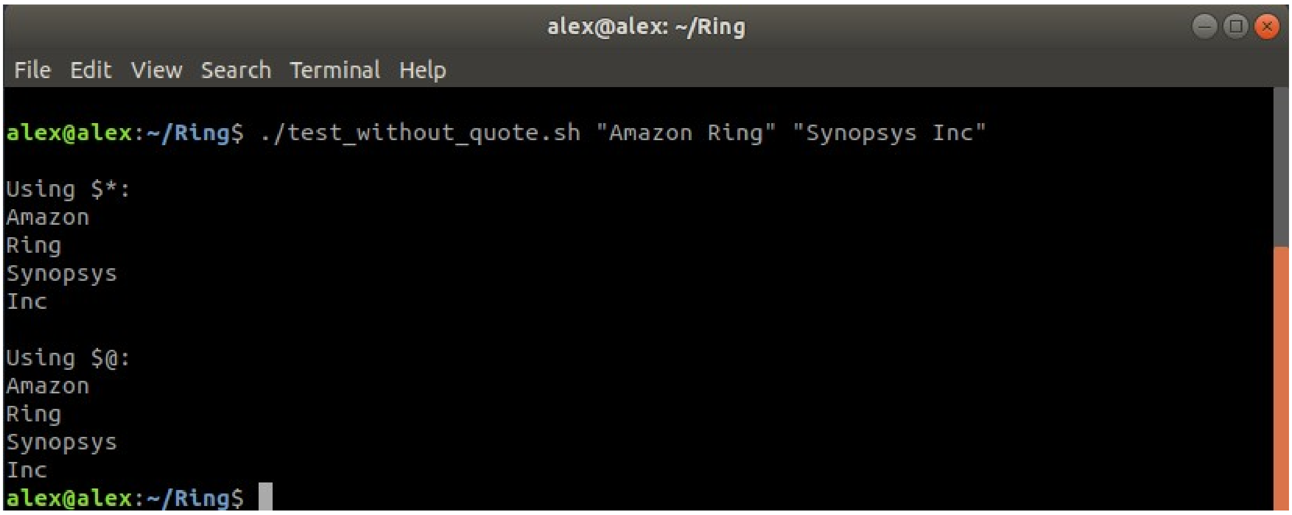
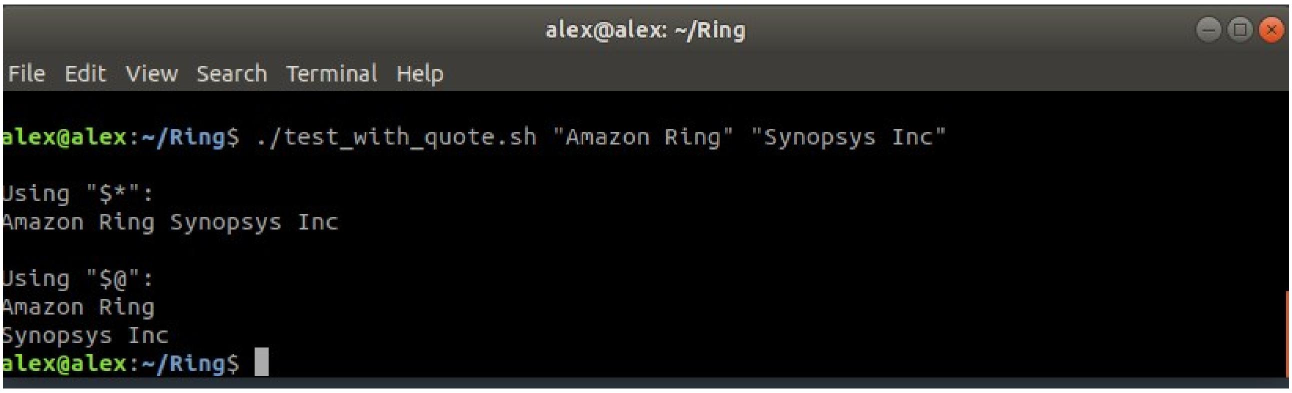
Question 1: What is the difference between $* and $@

- Without double quote, there are no difference between $* and $@. They both take the list of arguments as a list
- Execute **test_without_quote.sh** script, we will see all arguments [Amazon, Ring, Synopsys, Inc] are printed line by line



```
alex@alex:~/Ring$ ./test_without_quote.sh "Amazon Ring" "Synopsys Inc"

Using $*:
Amazon
Ring
Synopsys
Inc

Using $@:
Amazon
Ring
Synopsys
Inc
alex@alex:~/Ring$
```

- But there will be difference between “$*” and “$@”. “$*” treat arguments as a single string, while “$@” treat arguments as an array. Execute **test_with_quote.sh** script, we will observe the difference
- With “$*”, all arguments are printed in a single line
- With “$@”, arguments are separated by “”, there for “Amazon Ring” and “Synopsys Inc” are printed in two lines



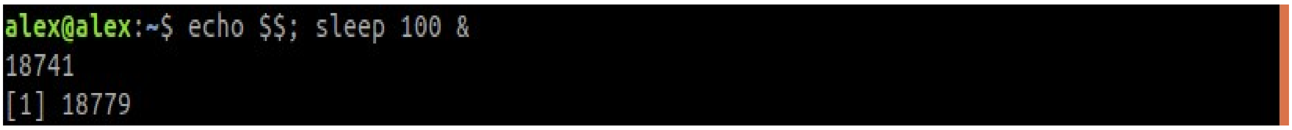
```
alex@alex:~/Ring$ ./test_with_quote.sh "Amazon Ring" "Synopsys Inc"

Using "$*":
Amazon Ring Synopsys Inc

Using "$@":
Amazon Ring
Synopsys Inc
alex@alex:~/Ring$
```

Question 2: What is the difference between $$ and $!

- Call **CMD** is the current executed command
- **$$** give the process id of **CMD**
- **$!** give the process id of the last background process , which is child process of **CMD**

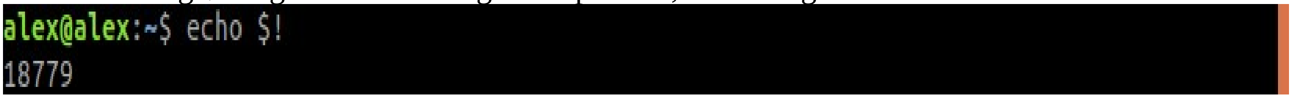
Example 1: **CMD** = echo $$; sleep 100 &
Basically to print the current process id and sleep for 100 seconds

Current process id printed as 18741. Also **[1] 18779** is printed as : There are one, also the last background process with id 18779.



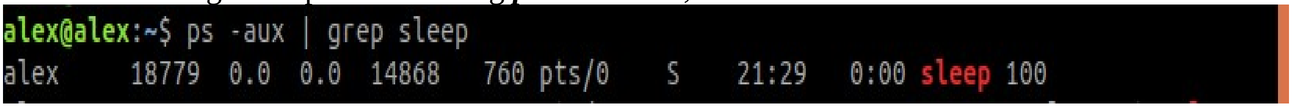
```
alex@alex:~$ echo $$; sleep 100 &
18741
[1] 18779
```

Now run using $! to get the last background process, it would give **18779**



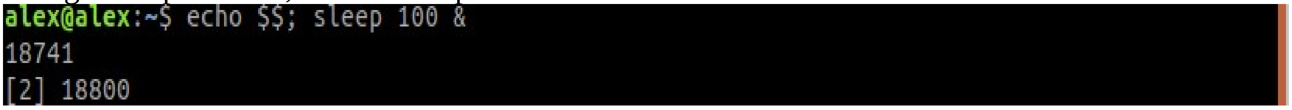
```
alex@alex:~$ echo $!
18779
```

Search the background processes using ***ps*** command , **18779** will be found



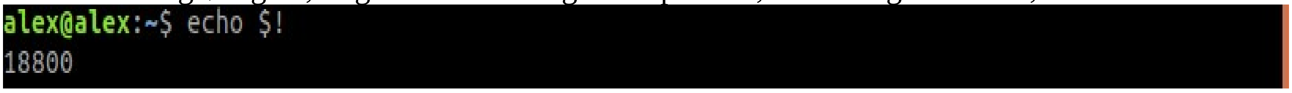
```
alex@alex:~$ ps -aux | grep sleep
alex     18779  0.0  0.0  14868   760 pts/0    S    21:29   0:00 sleep 100
```

Now run **CMD** again, it will print the current process id as well as **[2] 18800** as: There are two background processes, and the last process ‘s id is **18800**



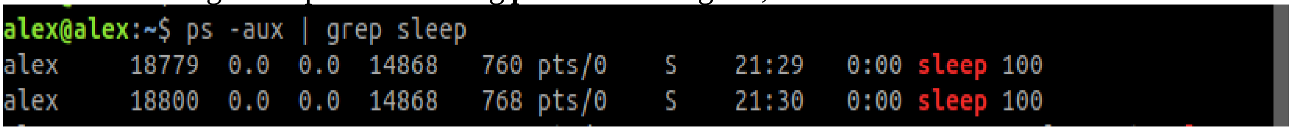
```
alex@alex:~$ echo $$; sleep 100 &
18741
[2] 18800
```

Now run using $! again, to get the last background process, it would give **18800**, instead of **18779**



```
alex@alex:~$ echo $!
18800
```

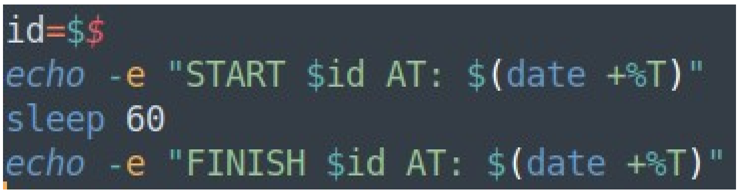
Search the background processes using ***ps*** command again , both **18779** and **18800** will be found



```
alex@alex:~$ ps -aux | grep sleep
alex     18779  0.0  0.0  14868   760 pts/0    S    21:29   0:00 sleep 100
alex     18800  0.0  0.0  14868   768 pts/0    S    21:30   0:00 sleep 100
```

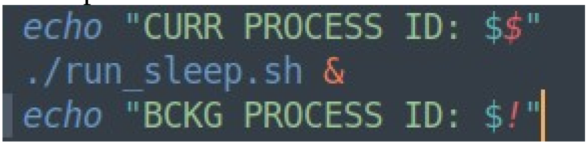
Example 2:

- **run_sleep.sh**: print current process ID and sleep for 60 seconds



```
id=$$
echo -e "START $id AT: $(date +%T)"
sleep 60
echo -e "FINISH $id AT: $(date +%T)"
```

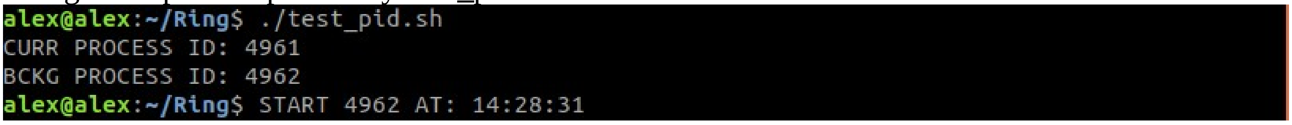
- **test_pid.sh**: Print current process DI, call run_sleep.sh as background, and print the lastback ground process



```
echo "CURR PROCESS ID: $$"
./run_sleep.sh &
echo "BCKG PROCESS ID: $!"
```

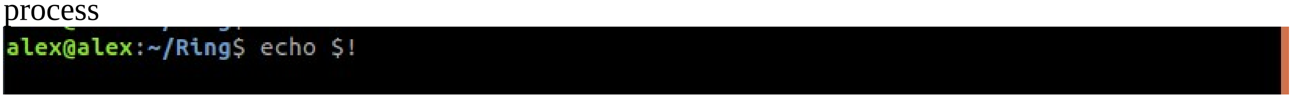
Execute **test_pid.sh**, we will see current process ID is 4961, and last background process ID is **4962**, which is ID of the process executing **run_sleep.sh.**

run_sleep.sh in turn of executing will print its current process ID **4962**, matching with last background process printed by test_pid.sh



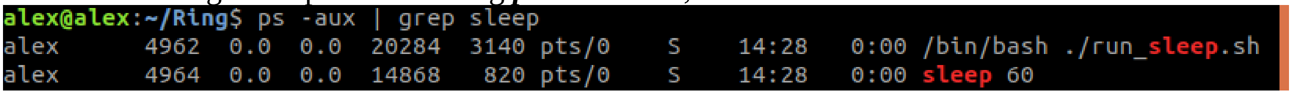
```
alex@alex:~/Ring$ ./test_pid.sh
CURR PROCESS ID: 4961
BCKG PROCESS ID: 4962
alex@alex:~/Ring$ START 4962 AT: 14:28:31
```

Note that if we try execute: echo $! , it will print nothing, not like in Example1. This because **run_sleep.sh** is child process of **test_pid.sh**. The current terminal process has no child background process



```
alex@alex:~/Ring$ echo $!
```

Search the background processes using ***ps*** command , **4962** will be found



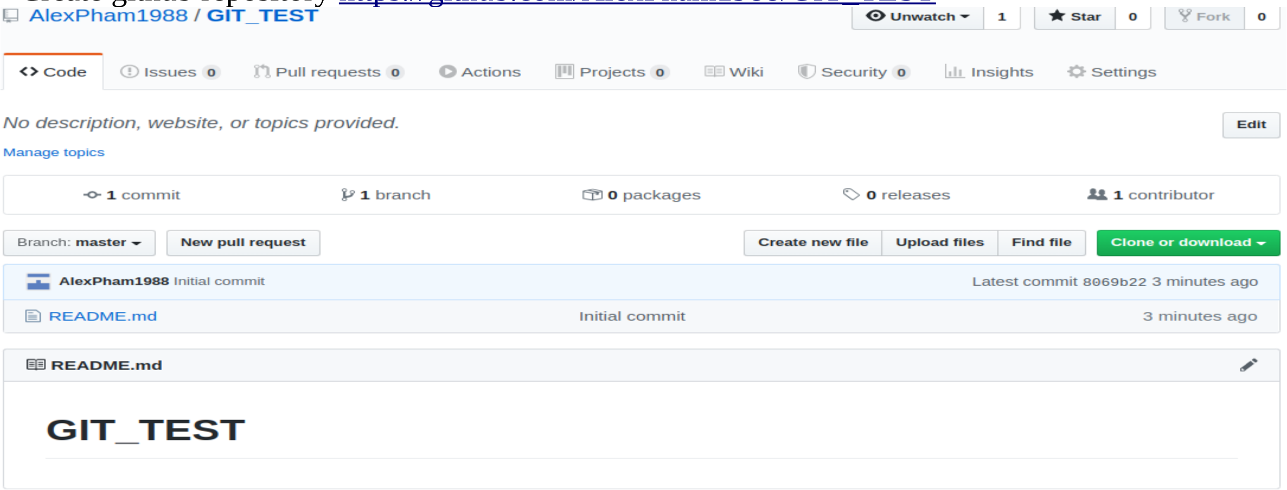
```
alex@alex:~/Ring$ ps -aux | grep sleep
alex      4962  0.0  0.0  20284  3140 pts/0    S    14:28   0:00 /bin/bash ./run_sleep.sh
alex      4964  0.0  0.0  14868   820 pts/0    S    14:28   0:00 sleep 60
```

Question 3: What is the difference between *git pull* and *git merge*

git fetch : Download contents from remote repository
git merge : Update contents to local repository
git pull is combination of **git fetch** and **git merge**. It would download contents from remote repository and update to local repository

We can verify those command via this example:

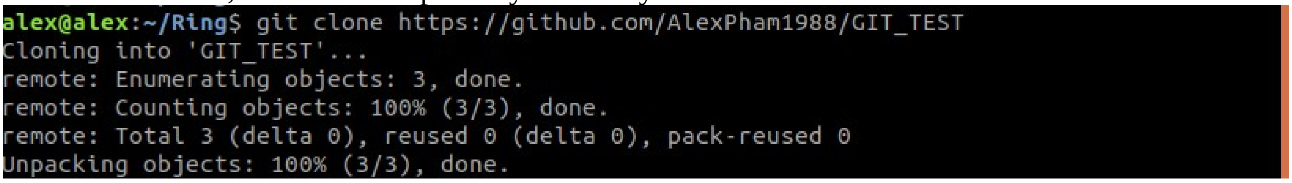
- Create github repository https://github.com/AlexPham1988/GIT_TEST



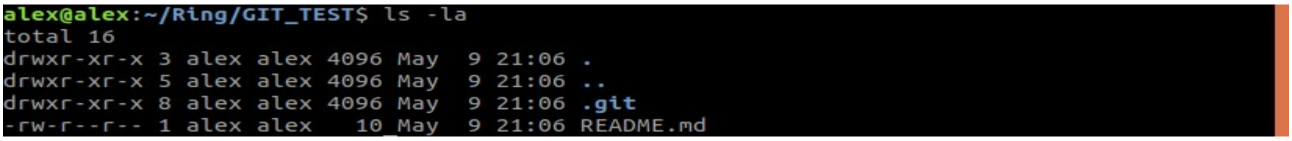
AlexPham1988 Initial commit — Latest commit 8069b22 3 minutes ago

README.md — Initial commit — 3 minutes ago

- Clone it to local, we will see repository with only README.ms

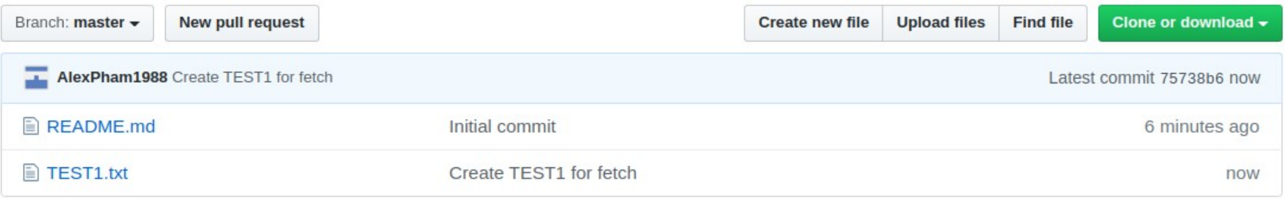


```
alex@alex:~/Ring$ git clone https://github.com/AlexPham1988/GIT_TEST
Cloning into 'GIT_TEST'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
```



```
alex@alex:~/Ring/GIT_TEST$ ls -la
total 16
drwxr-xr-x 3 alex alex 4096 May  9 21:06 .
drwxr-xr-x 5 alex alex 4096 May  9 21:06 ..
drwxr-xr-x 8 alex alex 4096 May  9 21:06 .git
-rw-r--r-- 1 alex alex   10 May  9 21:06 README.md
```

- On github create a file named TEST1.txt

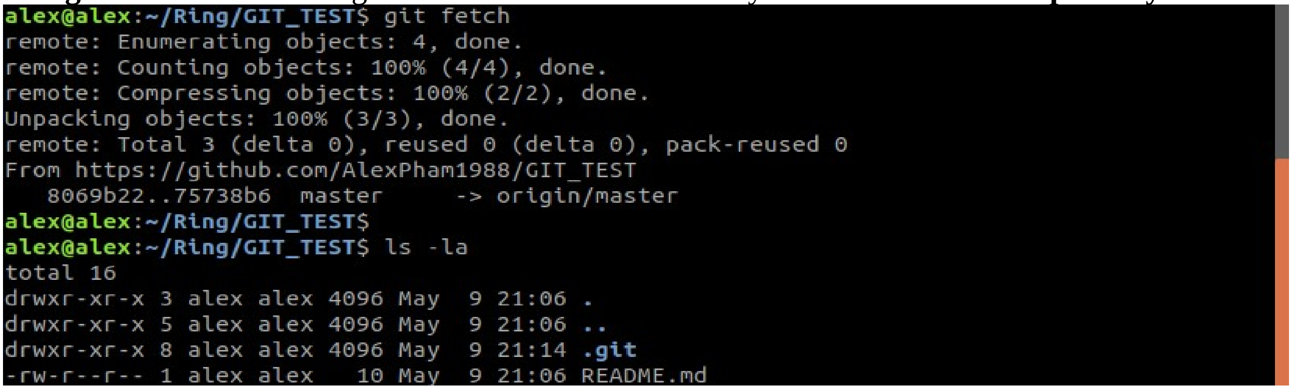


AlexPham1988 Create TEST1 for fetch — Latest commit 75738b6 now

README.md — Initial commit — 6 minutes ago

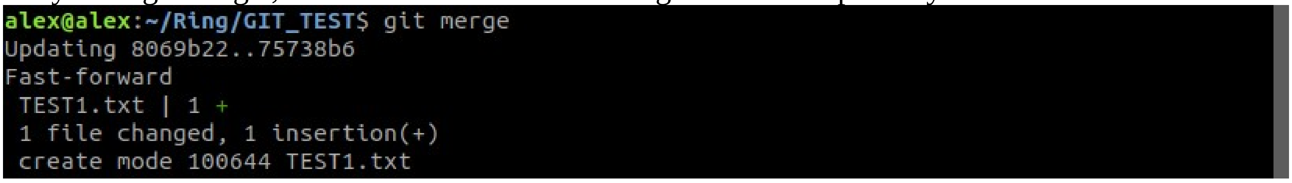
TEST1.txt — Create TEST1 for fetch — now

Run **git fetch** command to get TEST1.txt. However it not yet available in local **repository**



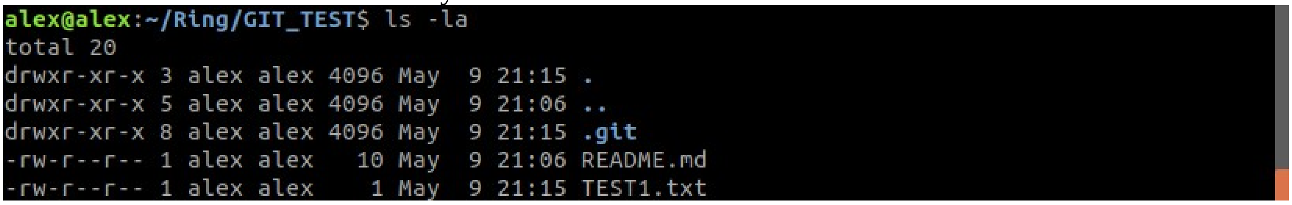
```
alex@alex:~/Ring/GIT_TEST$ git fetch
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
Unpacking objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
From https://github.com/AlexPham1988/GIT_TEST
   8069b22..75738b6  master     -> origin/master
alex@alex:~/Ring/GIT_TEST$
alex@alex:~/Ring/GIT_TEST$ ls -la
total 16
drwxr-xr-x 3 alex alex 4096 May  9 21:06 .
drwxr-xr-x 5 alex alex 4096 May  9 21:06 ..
drwxr-xr-x 8 alex alex 4096 May  9 21:14 .git
-rw-r--r-- 1 alex alex   10 May  9 21:06 README.md
```

Only after git merge , the new contents will be merged to local repository



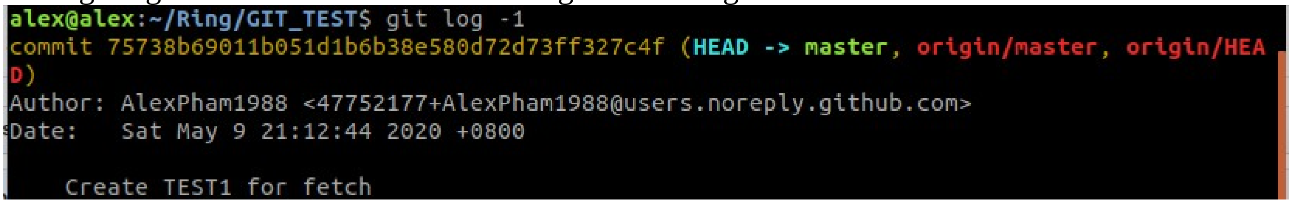
```
alex@alex:~/Ring/GIT_TEST$ git merge
Updating 8069b22..75738b6
Fast-forward
 TEST1.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 TEST1.txt
```

Now we can see TEST1.txt already available



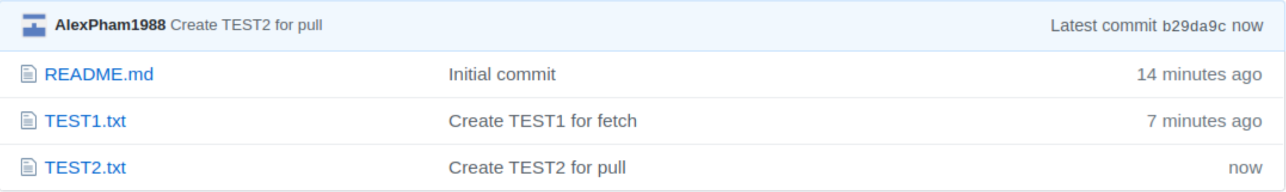
```
alex@alex:~/Ring/GIT_TEST$ ls -la
total 20
drwxr-xr-x 3 alex alex 4096 May  9 21:15 .
drwxr-xr-x 5 alex alex 4096 May  9 21:06 ..
drwxr-xr-x 8 alex alex 4096 May  9 21:15 .git
-rw-r--r-- 1 alex alex   10 May  9 21:06 README.md
-rw-r--r-- 1 alex alex    1 May  9 21:15 TEST1.txt
```

View git log we can observe the latest change on creating TEST1.txt



```
alex@alex:~/Ring/GIT_TEST$ git log -1
commit 75738b69011b051d1b6b38e580d72d73ff327c4f (HEAD -> master, origin/master, origin/HEA
D)
Author: AlexPham1988 <47752177+AlexPham1988@users.noreply.github.com>
Date:   Sat May 9 21:12:44 2020 +0800

    Create TEST1 for fetch
```

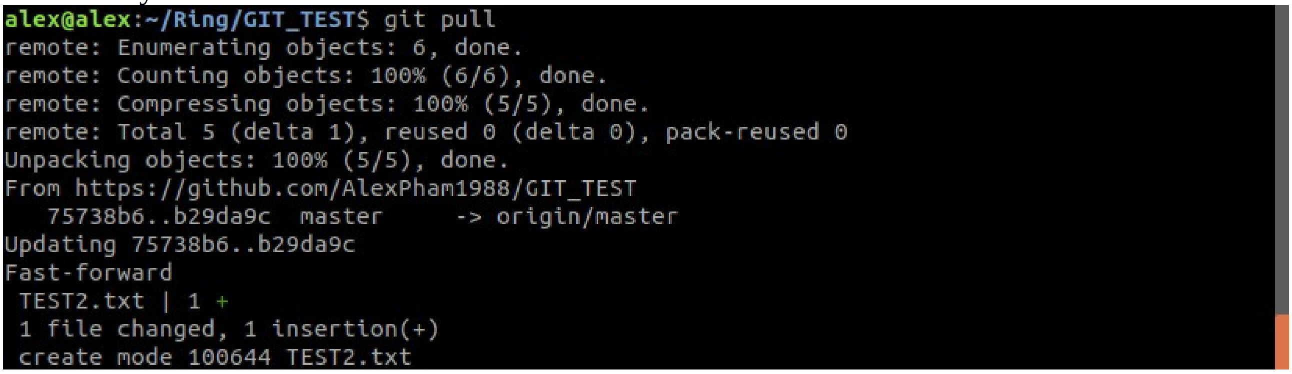
Now create another file called TEST2.txt



AlexPham1988 Create TEST2 for pull — Latest commit b29da9c now

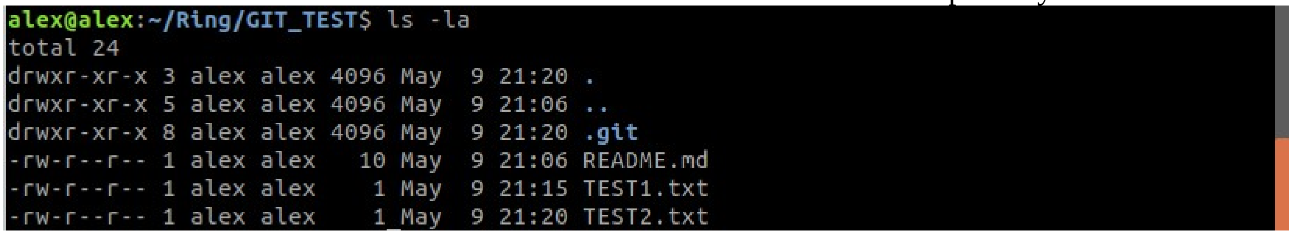
README.md	Initial commit	14 minutes ago
TEST1.txt	Create TEST1 for fetch	7 minutes ago
TEST2.txt	Create TEST2 for pull	now

Run **git pull** command to get TEST2.txt. This time TEST2.txt will be synced to local repository without any additional action.



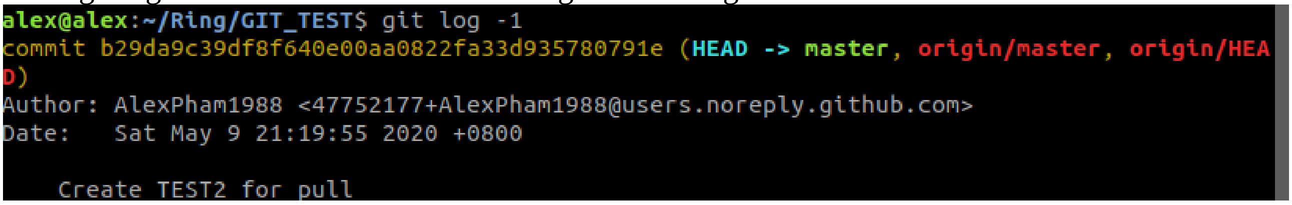
```
alex@alex:~/Ring/GIT_TEST$ git pull
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (5/5), done.
From https://github.com/AlexPham1988/GIT_TEST
   75738b6..b29da9c  master     -> origin/master
Updating 75738b6..b29da9c
Fast-forward
 TEST2.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 TEST2.txt
```

git pull and **git merge** command will be aborted if there are any conflicts. In our case there are no conflict therefore we can observed TEST2.txt has available in our local repository



```
alex@alex:~/Ring/GIT_TEST$ ls -la
total 24
drwxr-xr-x 3 alex alex 4096 May  9 21:20 .
drwxr-xr-x 5 alex alex 4096 May  9 21:06 ..
drwxr-xr-x 8 alex alex 4096 May  9 21:20 .git
-rw-r--r-- 1 alex alex   10 May  9 21:06 README.md
-rw-r--r-- 1 alex alex    1 May  9 21:15 TEST1.txt
-rw-r--r-- 1 alex alex    1 May  9 21:20 TEST2.txt
```

View git log we can observe the latest change on creating TEST2.txt



```
alex@alex:~/Ring/GIT_TEST$ git log -1
commit b29da9c39df8f640e00aa0822fa33d935780791e (HEAD -> master, origin/master, origin/HEA
D)
Author: AlexPham1988 <47752177+AlexPham1988@users.noreply.github.com>
Date:   Sat May 9 21:19:55 2020 +0800

    Create TEST2 for pull
```