תכנות מונחה עצמים
תרגול 3

נערך ע"י: אוהד שירזי

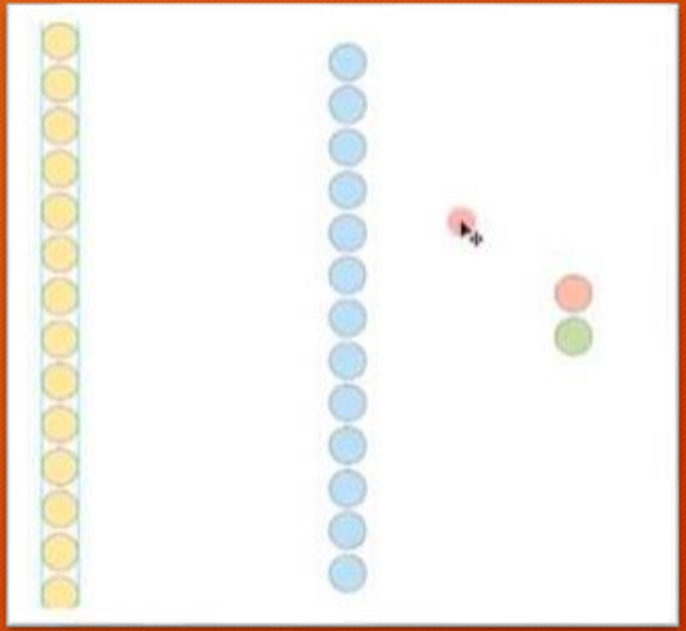
נושאים להיום:

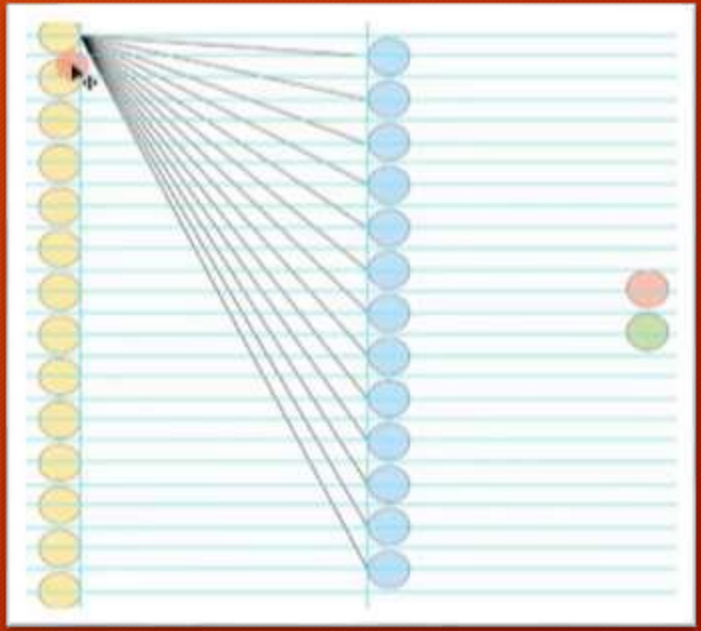
- דוגמאות שקשורות לתרגול האחרון
- Git
- Github
- Readme.md
- Python
- מעבר על המטלה

Theory

Git - Version Control System

- Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.



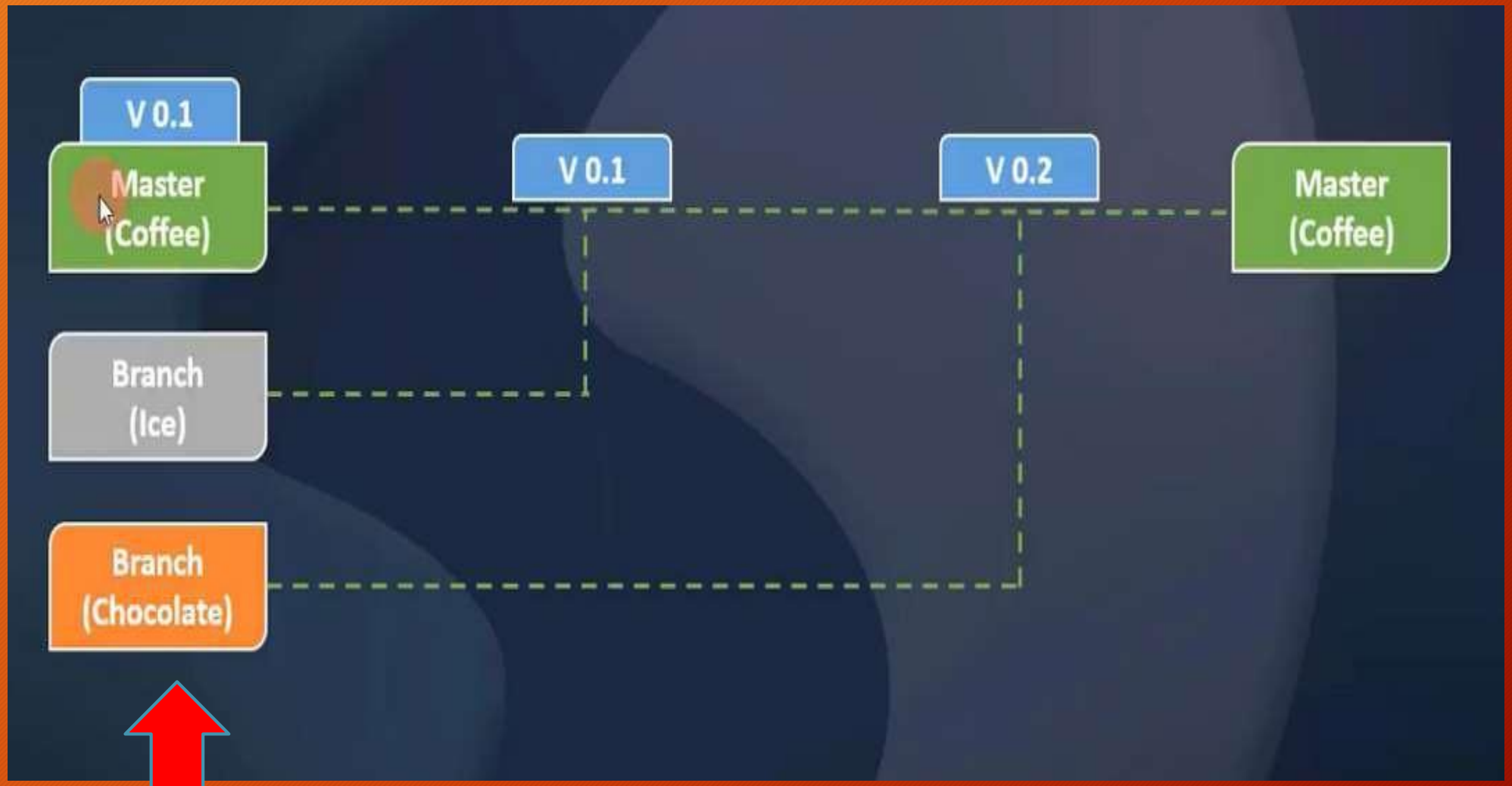


Branch

Master Branch

Branch 1

Branch


V 0.1
Master
(Coffee)
V 0.1
V 0.2
Master
(Coffee)
Branch
(Ice)
Branch
(Chocolate)

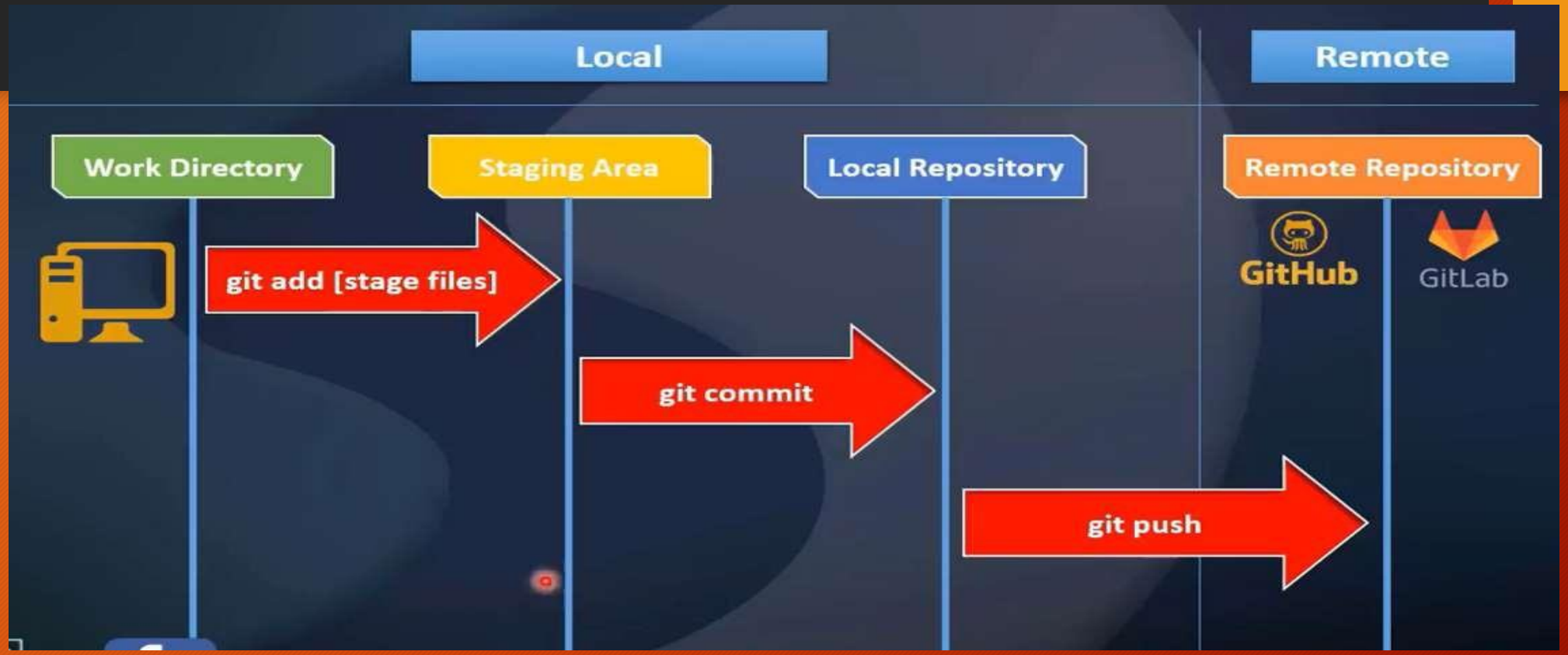
Branch


V 0.1
Master
(Coffee)
V 0.1
V 0.2
V 1.0
Chocolate
Cold Coffee
Branch
(Ice)
Branch
(Chocolate)

Repository


Local
Remote
Work Directory
Staging Area
Local Repository
Remote Repository
GitHub
GitLab

Repository


Local
Remote
Work Directory
Staging Area
Local Repository
Remote Repository
GitHub
GitLab
git add [stage files]
git commit
git push

Practical

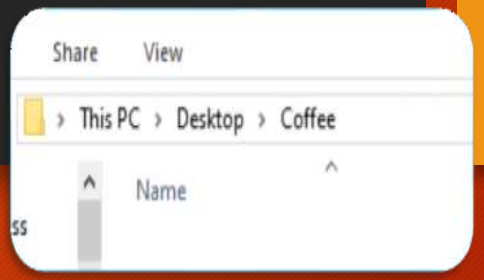
Let’s talk דוגרי

נפתח רפוסיטורי חדש בגיטהאב שלנו עליו נרצה לעבוד
נפתח את תיקיית הפרויקט עליו אנחנו עובדים ושם נפתח את הטרמינל ועם פקודה ראשונה נתחיל לעבוד
לאחר מכן נוסיף עם פקודה שניה את הקבצים שאנחנו רוצים לגיט הלוקאלי שלנו
בעזרת פקודה שלישית נוסיף גרסא חדשה לפרויקט זה עם הודעה מסוימת
נקשר את הגיט שלנו לגיטהאב בעזרת פקודה רביעית
נפתח בראנצ' נוסף עם פקודה חמישית
ופקודה שישית תדחוף את הקבצים לגיטאהב

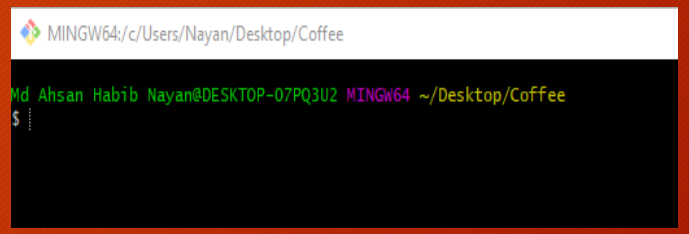
1. Git init
2. Git add first.java second.py
3. Git commit –m “my msg”
4. Git remote add “remote_name” “url_to_repo”
5. Git branc –M “b_name”
6. Git push –u “remote_name” “b_name”

Initialization

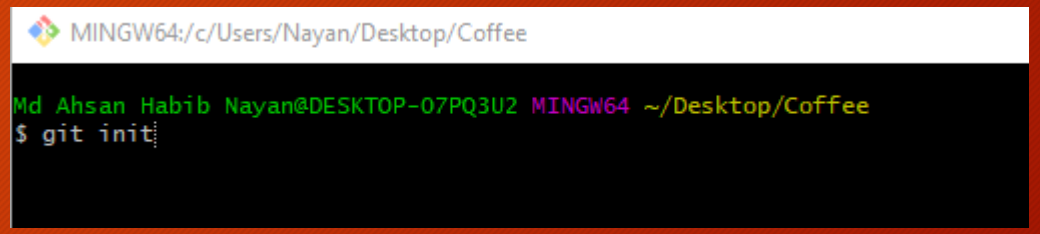
- Create a folder in any directory. For example, I created a folder named “Coffee” in my desktop.



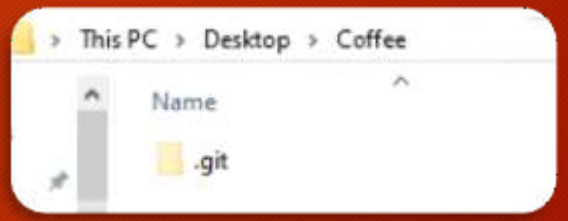
- Then open Git Bash here.



- Then type “git init” for initialization.



- Now you can see “.git” folder in you “Coffee” directory.



Configuration

- To set username globally:

  git config --global user.name “xxxx”

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee (master)
$ git config --global user.name "nayan"
```

- To set email globally:

  git config --global user.email “xxxx@gmail.com”

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee (master)
$ git config --global user.email "habib@cse.green.edu.bd"
```

- To set username and email locally:

  git config user.name “xxxx”

  git config user.email “xxxx@gmail.com”

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee (master)
$ git config --list
```

```
user.email=habib@cse.green.edu.bd
user.name=nayan
```

- Now type “git config --list” to show your configuration.

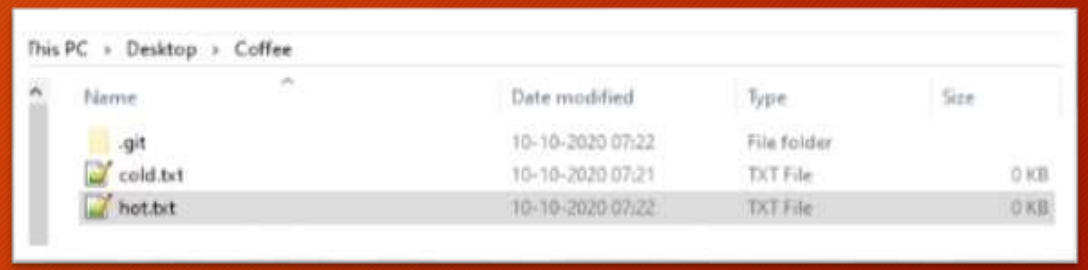
Commands

- To clear your screen just type:

  clear

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW
$ clear
```

- Let's create two files in "Coffee" dir.
  - cold.txt
  - hot.txt



- Now type: git status

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        cold.txt
        hot.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Commands

- Let’s use add command:

  git add cold.txt

```
Md Ahsan Habib Nayan@DESKTOP-07
$ git add cold.txt
```

- Now type: git status

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   cold.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        hot.txt
```

COMMANDS

- Add all files using single command: git add –all or

  git add .

```
Md Ahsan Habib Nayan@DESKTO
$ git add .
```

```
Md Ahsan Habib Nayan@DESKTOP-
$ git add --all
```

- Now type:

git status

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   cold.txt
        new file:   hot.txt
```

Commit

Commit

- Staging area to Local repo:

  git commit -m “Added two files”

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW
$ git commit -m "Added two files"
```

- Now type:

  git status

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64
$ git status
On branch master
nothing to commit, working tree clean
```

```
Md Ahsan Habib Nayan@DESKTOP
$ touch chocolate.txt
```

- Let’s commit again:
  - First create a file “chocolate.txt” in “Coffee”.
  - Now add this file.
  - Then commit.

```
Md Ahsan Habib Na
$ git add .
```

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/C
$ git commit -m "Added chocolate.txt"
[master ef752e6] Added chocolate.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 chocolate.txt
```

COMMIT

- To show our commit:

  git log

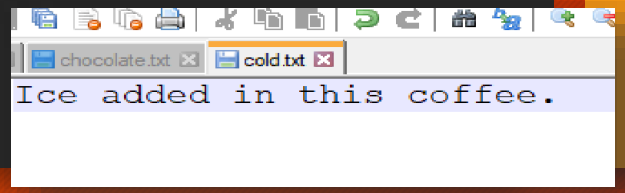
```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee (master)
$ git log
commit ef752e694363edf2bd55a1abe4250703fe68e23a (HEAD -> master)
Author: nayan <habib@cse.green.edu.bd>
Date:   Sat Oct 10 08:41:54 2020 +0600

    Added chocolate.txt

commit c0dbc9d758fc91af1a76418606f1bc16952e8ba6
Author: nayan <habib@cse.green.edu.bd>
Date:   Sat Oct 10 08:31:52 2020 +0600

    Added two files
```

- Another way [Simple]: git log --oneline

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Cof
$ git log --oneline
ef752e6 (HEAD -> master) Added chocolate.txt
c0dbc9d Added two files
```

COMMIT

- Let’s write something in cold.txt file.



- Now type:

  git status

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   cold.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

- Let’s commit again:
  - Add the changed file [cold.txt].
  - Then commit.
  - Now use git log command to show commit

```
Md Ahsan Habib Na
$ git add .
```

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/De
$ git commit -m "ice added in the coffee"
[master 2c433cb] ice added in the coffee
 1 file changed, 1 insertion(+)
```

Checkout

Checkout

- Let's we need to go back our previous version of cold.t
- Can we do that?
- Yes, It's simple. Just use checkout command.

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desk
$ git log --oneline
2c433cb (HEAD -> master) ice added in the coffee
ef752e6 Added chocolate.txt
c0dbc9d Added two files
```

- To do this.
  - First log the commits.
  - Then use checkout using commit id.
  - You can see that you cold.txt file got empty again.
  - Note: we are not in master branch now.

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2
$ git checkout ef752e6
Note: switching to 'ef752e6'.
```

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee ((ef752e6...))
```

CHECKOUT

- Let’s use git log again. You can see there are only two commits exist.

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MI
$ git log --oneline
ef752e6 (HEAD) Added chocolate.txt
c0dbc9d Added two files
```

- Can you switch one of the commit-ids ?

- Have a try

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MING
$ git checkout master
```

- Let’s back to our master branch.

Diff

Diff

- Let’s compare two files. We have to use git diff command.

- To do this.
  - First add some text in chocolate.txt.
  - You can use git status to show if chocolate.txt changes or not.
  - Then use git diff comman.
  - You can see the changes.

  - Note: git diff always works between latest changes.

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   chocolate.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Des
$ git diff
diff --git a/chocolate.txt b/chocolate.txt
index e69de29..494e135 100644
--- a/chocolate.txt
+++ b/chocolate.txt
@@ -0,0 +1 @@
+some chocolate added here.
\ No newline at end of file
```

DIFF

- To show changes at a particular commit you can use git show with commit-id.

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee (master)
$ git log --oneline
2c433cb (HEAD -> master) ice added in the coffee
ef752e6 Added chocolate.txt
c0dbc9d Added two files

Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee (master)
$ git show 2c433cb
commit 2c433cb01cb5a31673fa668702a92d27c573ea21 (HEAD -> master)
Author: nayan <habib@cse.green.edu.bd>
Date:   Sat Oct 10 08:52:57 2020 +0600

    ice added in the coffee

diff --git a/cold.txt b/cold.txt
index e69de29..7450139 100644
--- a/cold.txt
+++ b/cold.txt
@@ -0,0 +1 @@
+Ice added in this coffee.
\ No newline at end of file
```

DIFF

- We haven't add the updated chocolate.txt file yet. Just add this file. Then commit it.
- Now we again added some text in chocolate.txt file. Then add and commit it.
- Using git log you can see like this.

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffe
$ git log --oneline
ef8fd3a (HEAD -> master) dark chocolate added
c0126f2 some chocolate added
2c433cb ice added in the coffee
ef752e6 Added chocolate.txt
c0dbc9d Added two files
```

- Now we want to show differences between two chocolate.txt files [previous and current].
- To do use use git diff with two commit-ids.

  git diff id1 id2

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/De
$ git diff c0126f2 ef8fd3a
diff --git a/chocolate.txt b/chocolate.txt
index 494e135..76a8f6b 100644
--- a/chocolate.txt
+++ b/chocolate.txt
@@ -1 +1,3 @@
-some chocolate added here.
\ No newline at end of file
+some chocolate added here.
+
+dark chocolate added here.
\ No newline at end of file
```

DIFF

- What if we want to see the differences in the staged section ie. Afer adding and before commit.
- Use: git diff --staged
- To do this:
  - Update chocolate.txt again. Use git diff to show changes.
  - Now use git add command. If you want to show changes now and use git diff command it shows nothing.
  - To show changes use:

    Git diff --staged

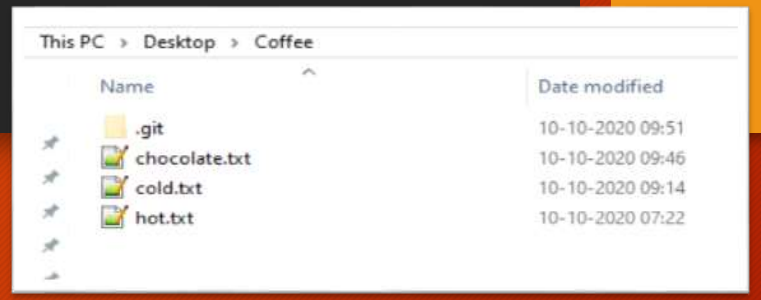
```
Md Ahsan Habib Nayan@DESKTOP-07PQ
$ git add .

Md Ahsan Habib Nayan@DESKTOP-07PQ
$ git diff
```

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/C
$ git diff --staged
diff --git a/chocolate.txt b/chocolate.txt
index 76a8f6b..d196d2b 100644
--- a/chocolate.txt
+++ b/chocolate.txt
@@ -1,3 +1,5 @@
 some chocolate added here.

-dark chocolate added here.
\ No newline at end of file
+dark chocolate added here.
+
+just chocolate added.
\ No newline at end of file
```

Remove

Remove file

- How can we delete a file?
- If you delete from “Coffee” directory then it is removed from current stage [not all commits].



- What if we want to delete permanently ie. delete a file from all commits ?
- We have to use: git rm file_name
- It deletes the file from all commits.

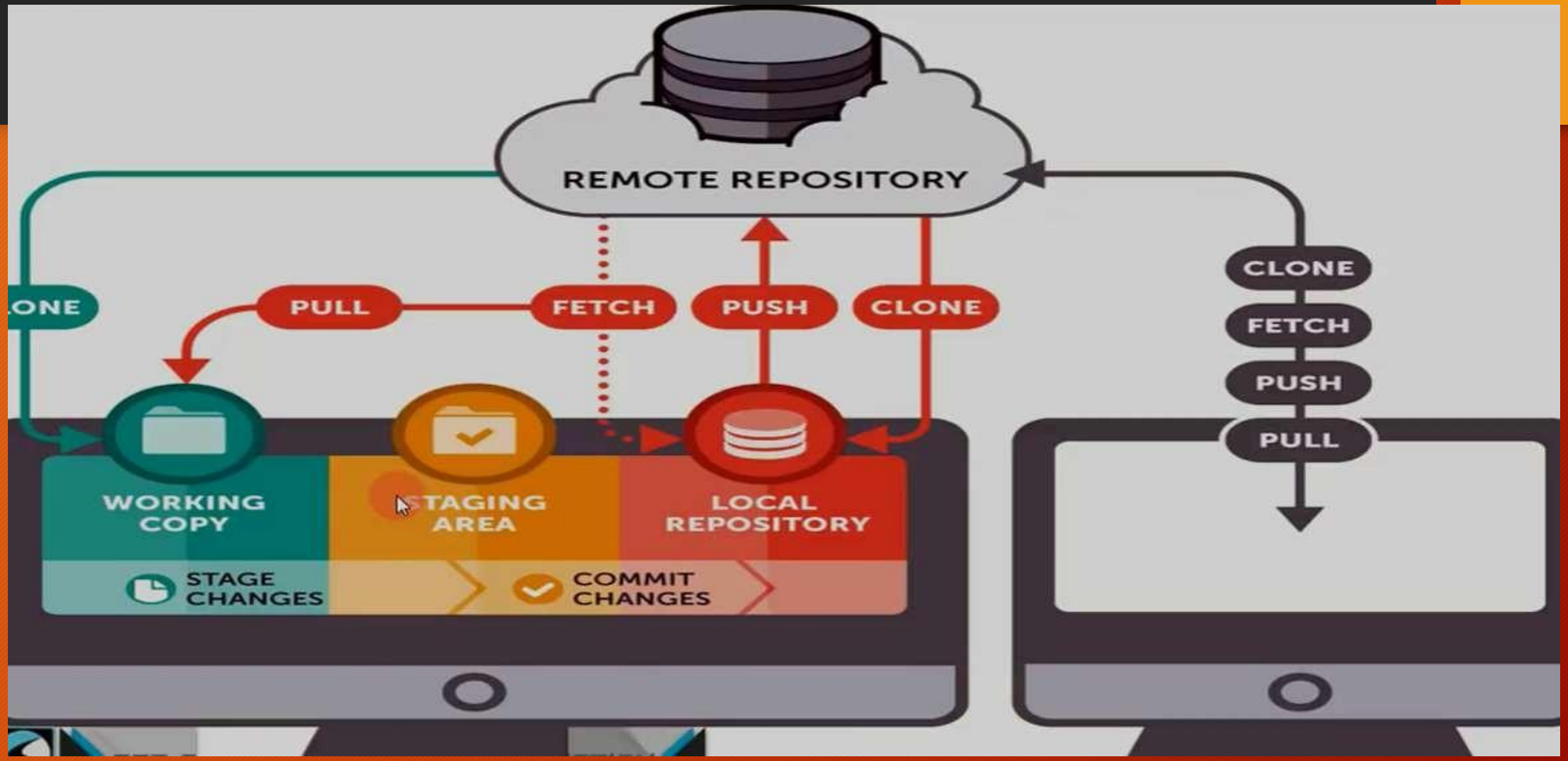
```
Md Ahsan Habib Nayan@DESKTOP-
$ git rm hot.txt
rm 'hot.txt'
```

- If we run git status then we can see that the hot.txt file is in staged section. If we want to delete from staged section then we have to use: git reset HEAD hot.txt. Now again run git status command to see the chages.

```
Md Ahsan Habib Nayan@DESKTOP-07PQ
$ git reset HEAD hot.txt
Unstaged changes after reset:
D       hot.txt
```

Let’s Store/Push the files into Remote Repository!!

Full System Pictorial View



Push

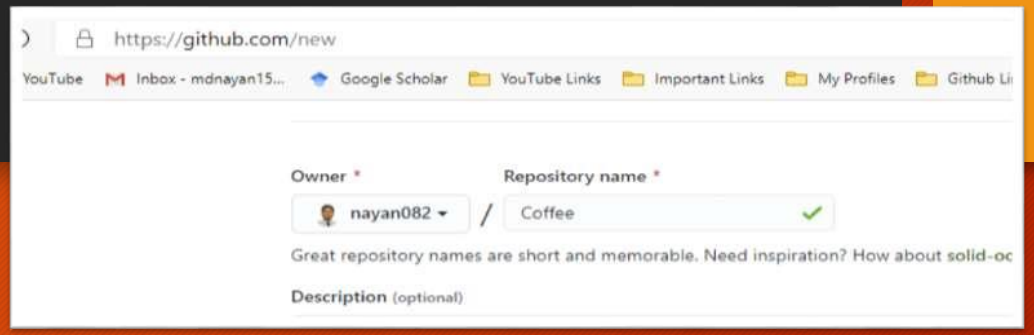
- How can we push the files into Remote Repo?

- To do this:
  - We need to have a github account.
  - Now let's create a Repository.
  - Then follow some steps.
    - Copy the remote repo link and run in the Bash Terminal.
    - Use push command.
    - Now refresh your repo.

- Let's see graphically.

PUSH

- First, create a repository.



- Then copy the link (remote repo) and run.

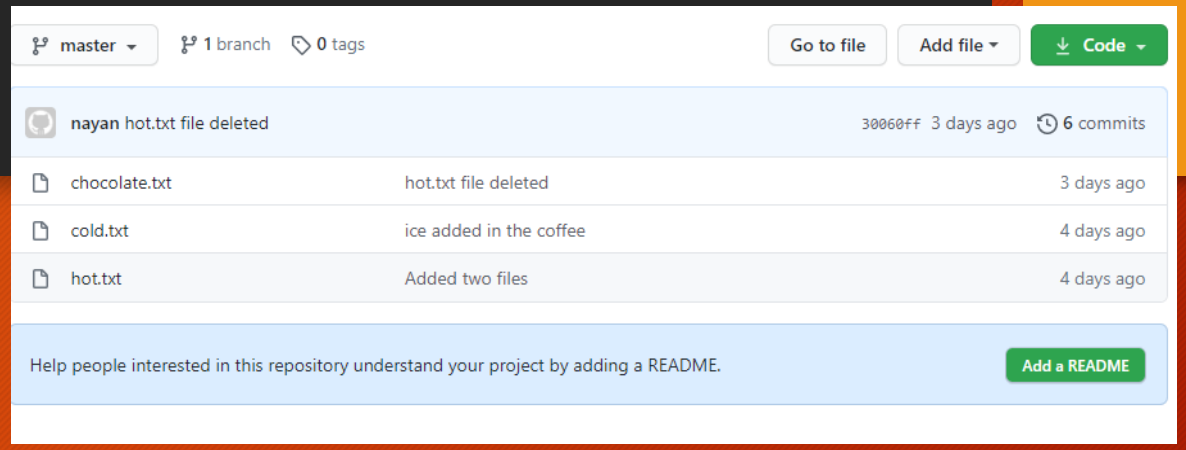
```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee (master)
$ git remote add origin https://github.com/nayan082/Coffee.git
```

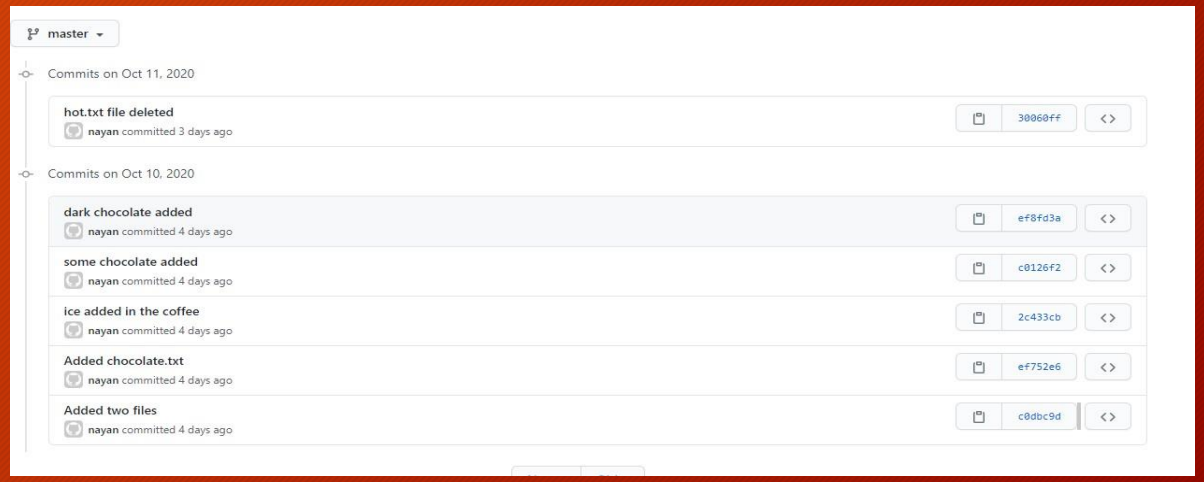
- Now run push command. git push origin master or git push -u origin master

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee (master)
$ git push origin master
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 4 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (17/17), 1.54 KiB | 32.00 KiB/s, done.
Total 17 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/nayan082/Coffee.git
 * [new branch]      master -> master
```

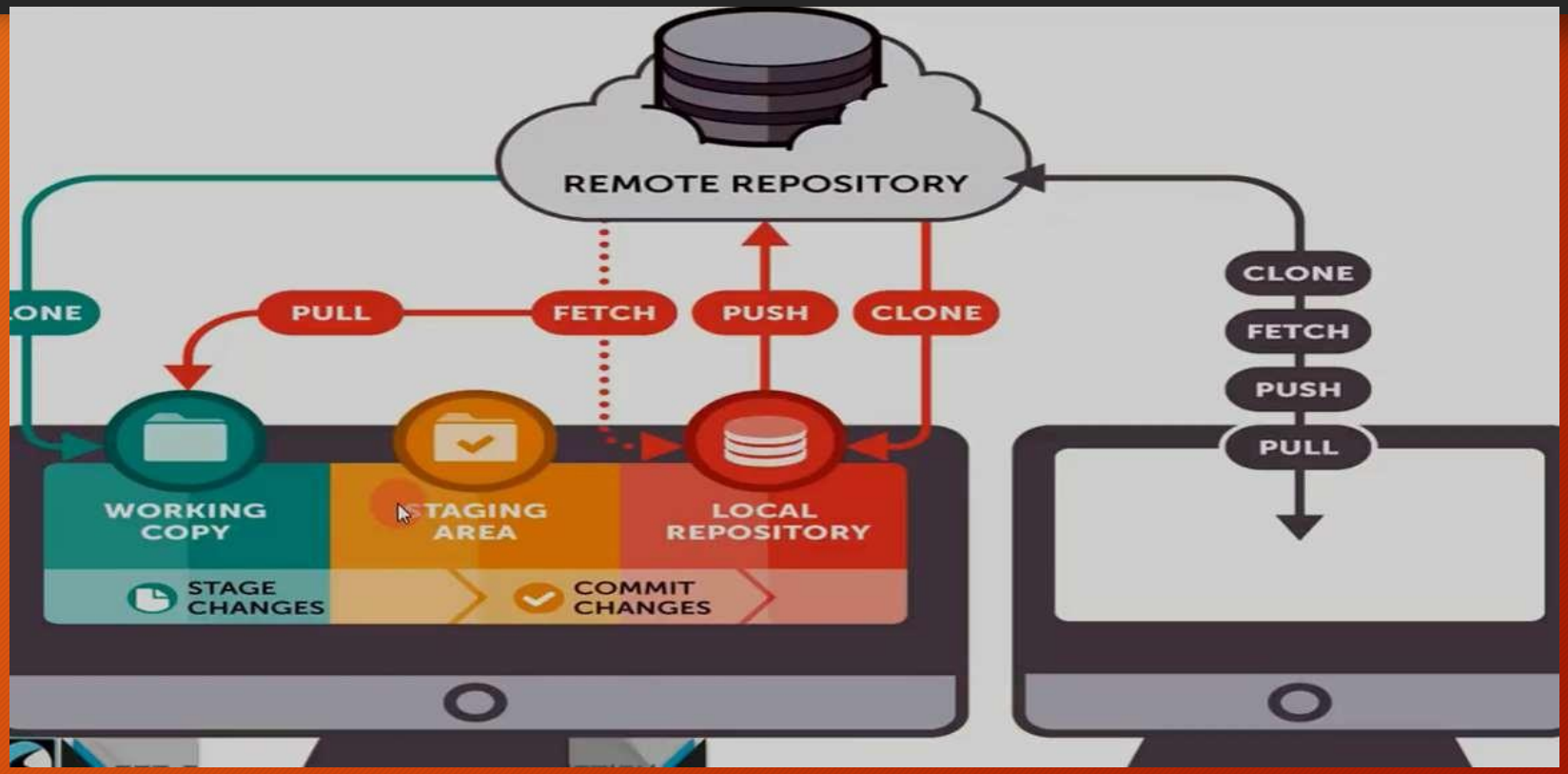
Repository

- We can see here all commits we have done.
- And we can go back any stage of those commits.



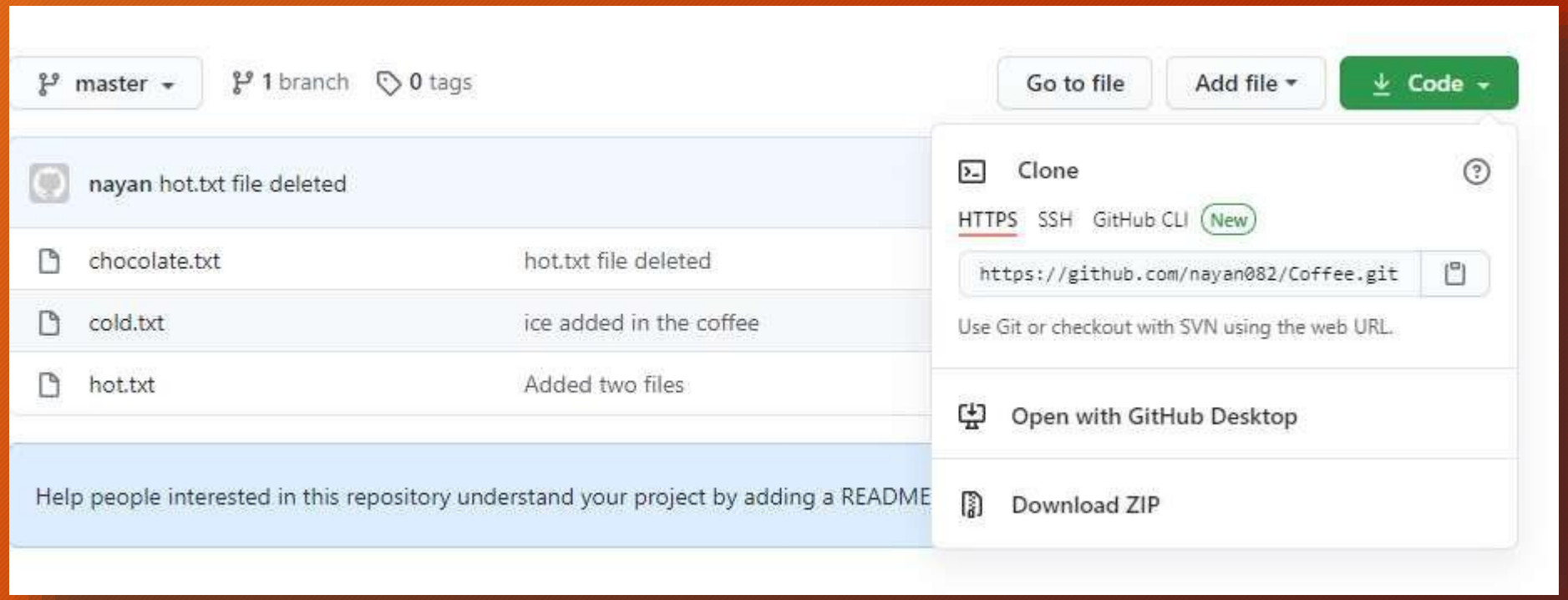


Download/Clone from remote repository



Clone

- How can we download/clone the projects into Remote Repo?
- To do this:
  - We can download as a zip or
  - We can clone the repo.



Clone

- To clone the repository copy the project link and run the command:

  git clone repo_link

- The folder named with the repository name.
- If you want to assign a different name use:

  git clone repo_link new_name

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee (master)
$ git clone https://github.com/nayan082/Coffee.git
Cloning into 'Coffee'...
remote: Enumerating objects: 17, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 17 (delta 2), reused 17 (delta 2), pack-reused 0
Unpacking objects: 100% (17/17), 1.52 KiB | 0 bytes/s, done.
```

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee (master)
$ git clone https://github.com/nayan082/Coffee.git new
```

Fetch

Fetch

- First, let's change cold.txt file (add a line into cold.txt) and make a commit (from remote repo).
- To fetch the repository we use:

  git fetch

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee (master)
$ git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 729 bytes | 5.00 KiB/s, done.
From https://github.com/nayan082/Coffee
   30060ff..90101d7  master     -> origin/master
```

Fetch

- Now run git status to see the commit.
- Now let's check the cold.txt file. Can you see any update ?
- No. Our machine only know that something changes. And they suggest us to use git pull command.!!
- Now use: git pull. Now check the cold.txt file.

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee (master)
$ git status
On branch master
Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)
```

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee (master)
$ git pull
Updating 30060ff..90101d7
Fast-forward
 cold.txt | 5 ++++-
 1 file changed, 4 insertions(+), 1 deletion(-)
```

Branch

Branch

- To create a branch we use: git branch branch_name
- To check how many branch we have: git branch
- * Sign represent the current branch.

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee (master)
$ git branch new_branch

Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee (master)
$ git branch
* master
  new_branch
```

BRANCH

- To switch a branch we use:

  git checkout branch_name

- We can create and switch at a time to the branch using:

  git checkout -b branch_name

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee (master)
$ git checkout new_branch
Switched to branch 'new_branch'
D       hot.txt

Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee (new_branch)
$ git checkout -b new1_branch
Switched to a new branch 'new1_branch'

Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee (new1_branch)
$
```

BRANCH

- Now we are in “new1_branch”. Anything changes in the Coffee repository, the changes will be happen in the new1_branch not affects other branches (master, new_branch).
- Let’s create a file in the current branch (new1_branch) and use git add and commit command.
- If you switch to master branch, you can see the changes in the Coffee repo.

CONFLICT

Conflicts generally arise when two people have changed the same lines in a file, or if one developer deleted a file while another developer was modifying it. In these cases, Git cannot automatically determine what is correct. Conflicts only affect the developer conducting the merge, the rest of the team is unaware of the conflict. Git will mark the file as being conflicted and halt the merging process. It is then the developers' responsibility to resolve the conflict.

Merge

Merge

- If you want to merge/add all the changes in the master branch. You can run:

  git merge new1_branch

- By running this you can see the changes happen to the master branch.

Delete a Branch

- If you want to delete a branch. You can run: git branch -d branch_name

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee (new1_branch)
$ git branch -d new_branch
Deleted branch new_branch (was 90101d7).
```

Git Ignore

Git Ignore

- If you want some files do not need to be commit or add we can make a list of them into “.gitignore” file.
- Let’s first create a file which should be ignored (ex: no_need.txt).
- Now we need to create “.gitignore” file and write no_need.txt inside it. Now run git status, you can see that the no_need.txt file not be tracked.

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee (new1_branch)
$ touch no_need.txt
```

```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee (new1_branch)
$ touch .gitignore
```

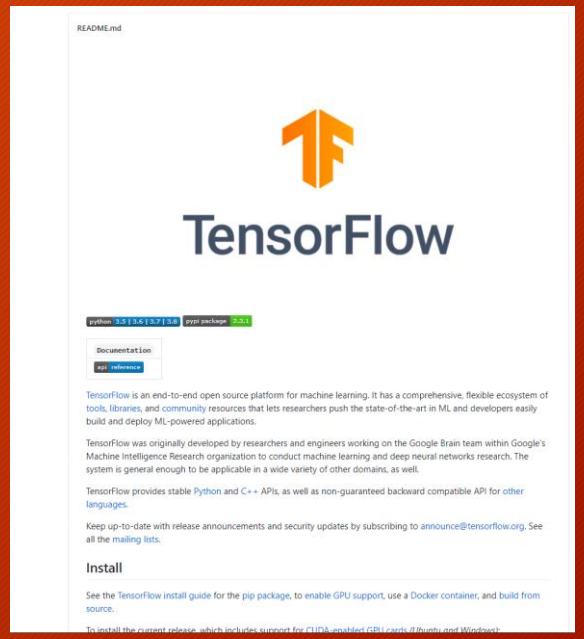
```
Md Ahsan Habib Nayan@DESKTOP-07PQ3U2 MINGW64 ~/Desktop/Coffee (new1_branch)
$ git status
On branch new1_branch
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
```

ReadMe.md

Markdown

Markdown is a way to style text on the web. You control the display of the document; formatting words as bold or italic, adding images, and creating lists are just a few of the things we can do with Markdown. Mostly, Markdown is just regular text with a few non-alphabetic characters thrown in, like # or *.

https://www.markdownguide.org/basic-syntax/



Markdown

https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet

https://www.markdownguide.org/cheat-sheet/

https://pdf2md.morethan.io/

Issues

Issues are a great way to keep track of tasks, enhancements, and bugs for your projects. They're kind of like email—except they can be shared and discussed with the rest of your team. Most software projects have a bug tracker of some kind. GitHub's tracker is called Issues, and has its own section in every repository.