

CMSC388T

Advanced Features of Git

Today's Lecture

1

Git Repositories

Clone, Mirror, Fork Repositories

2

Advance Git Branch Status

Other git commands useful when modifying code for repositories

3

README

What is the README and its Syntax



1. Git Repositories

Introduction to Mirroring, Cloning,
and Forking Repositories

Cloning Git Repositories

- Allows for a user to obtain a working copy of the repository
 - Cloning Repositories can only occur on a pre-existing repository
- Should only be used once
- Cloning also creates a remote connection titled “origin” that references the original repository

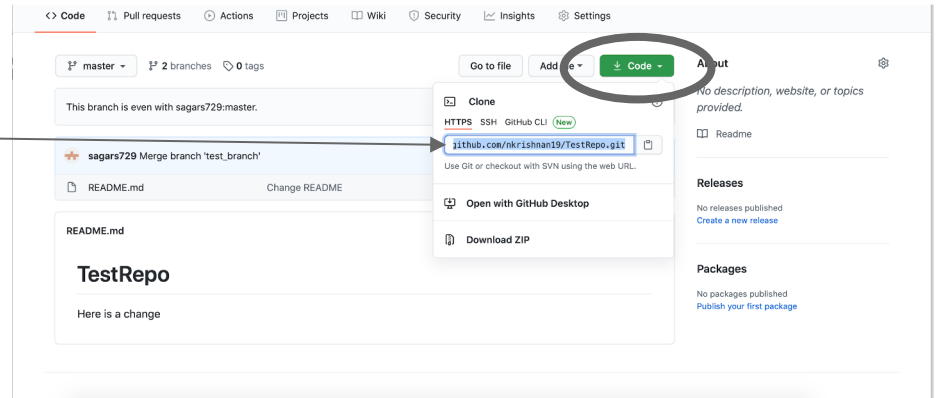
Git Clone

To Clone a Repository:

1. Create Folder/location on your computer where you would like to access the repository
2. Cd to the folder created
3. Open the repository on Github
4. Copy the URL from github website
5. Run the command line: **git clone <paste URL>** to clone the repository

Git Cloning

Retrieve the URL of the repository
Go to 'clone' and the drop down
arrow gives you the URL of the
repository to clone



Git Clone

The command line 'git clone <URL>' clones the repository into a directory locally

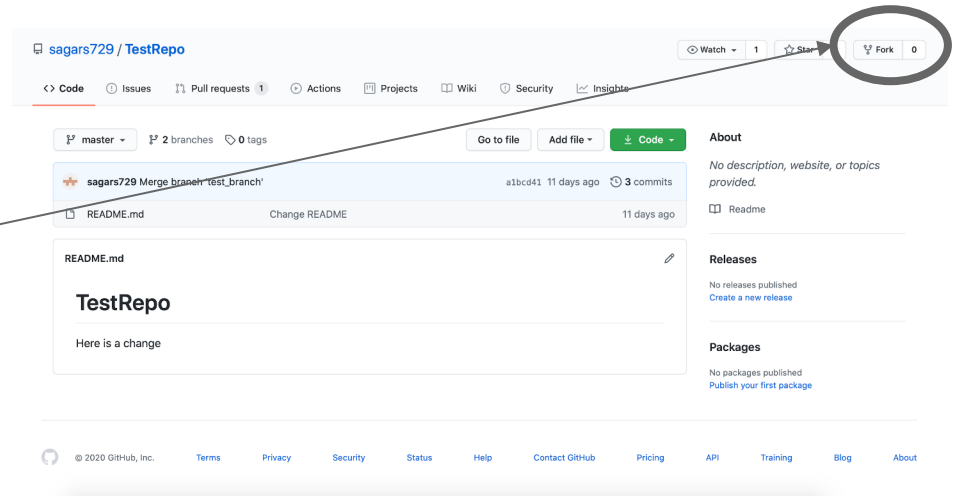
```
[ ~ ] git clone git@github.com:anwarmamat/cmsc330fall120.git
Cloning into 'cmsc330fall120'...
remote: Enumerating objects: 138, done.
remote: Counting objects: 100% (138/138), done.
remote: Compressing objects: 100% (110/110), done.
remote: Total 509 (delta 21), reused 127 (delta 13), pack-reused 371
Receiving objects: 100% (509/509), 679.07 KiB | 20.58 MiB/s, done.
Resolving deltas: 100% (110/110), done.
```

Forking

- Creates a Personal Development Environment
- Copy of the original repository on the same server, but with **private access** so that only developer that 'forked' the repository can push changes
 - Developer can then clone the newly forked repository locally for a personal environment
- When ready to publish a change, a developer needs to:
 - Push changes to personal repository
 - Create a **pull request** of the new file to the main repository

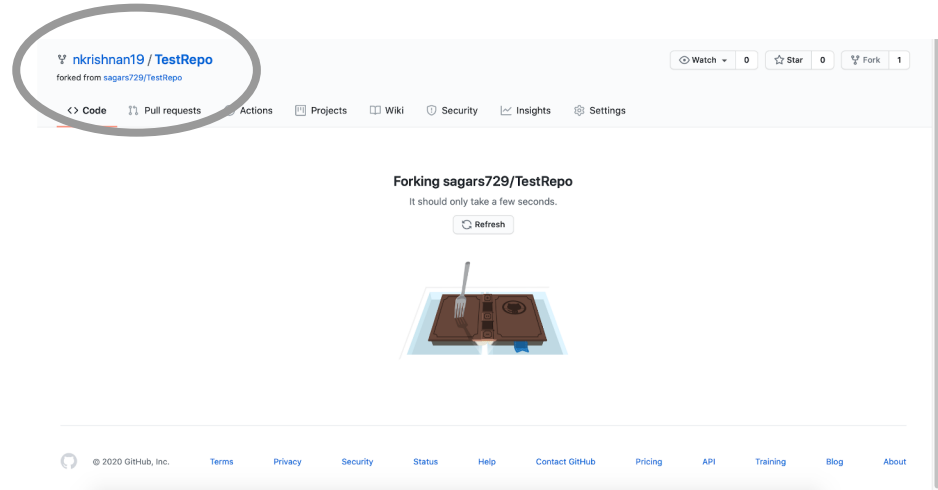
Forking

On your repository, the forking option is in the top right.



Forking

After clicking 'fork' github automatically forks the repository for you



Forking

A forked version of the original 'TestRepo' has been created

The screenshot shows a GitHub repository page for 'TestRepo' forked by 'nkrishnan19'. The repository is forked from 'sagars729/TestRepo'. The page displays the repository name, navigation tabs (Code, Pull requests, Actions, Projects, Wiki, Security, Insights, Settings), and repository details. A commit by 'sagars729' is highlighted, showing a merge of branch 'test_branch' with commit hash 'a1bcd41' and 3 commits. The README file is visible, containing the text 'TestRepo' and 'Here is a change'. The right sidebar shows sections for 'About', 'Releases', and 'Packages', all indicating no content has been published yet.

Forking

In order to access the forked repository locally, you have to clone the forked repository

```
[ ~ ] git clone https://github.com/nkrishnan19/TestRepo.git
Cloning into 'TestRepo'...
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 16 (delta 2), reused 12 (delta 1), pack-reused 0
Receiving objects: 100% (16/16), done.
Resolving deltas: 100% (2/2), done.
```

Mirroring

- Allows to duplicate a repository without forking it - private repo
- Requires updates to be fetched from main repository
 - Configure Remote Fetch and a origin push
- command line to mirror: `git clone --mirror`
- command line to update: `git remote update`

Mirroring

First, clone bare the repository that you want to mirror

```
$ git clone --bare https://github.com/exampleuser/old-repository.git
```

After entering the directory for the original repository, run the command `git push --mirror` and change the name of the repository to indicate the new one which will mirror the old one

```
$ cd old-repository.git  
$ git push --mirror https://github.com/exampleuser/new-repository.git
```

Submodules

- Project within a Project
- Feature: want to have 2 separate projects but be able to pull from one another and access one another
- Common example: creating libraries to use in projects
- Command line: **git submodule add <url of project>**

Submodules

Create New repository to add to a current repository

```
⌋ $ git submodule add https://github.com/nkrishnan19/Sub-Module-Example.git
Cloning into '/Users/nandhinikrishnan/git-test/TestRepo/Sub-Module-Example'...
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.
⌋ $
```


Submodules

Notice when you run `git status` 2 new files were added. These established the path for the submodule

```
⌋$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   .gitmodules
    new file:   Sub-Module-Example
```

Submodules

Notice the 'create mode 1XXXX' this implies that you are committing as a directory and not a file

```
--
[$ git commit -m "added a sub module"
[master 2c7d2cd] added a sub module
Committer:
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

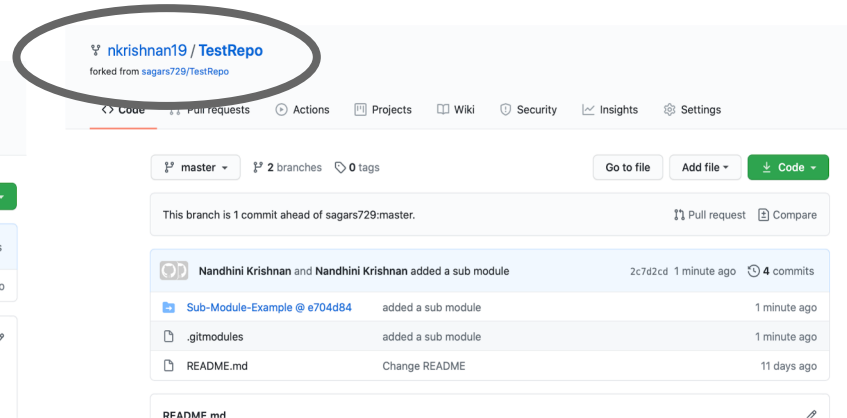
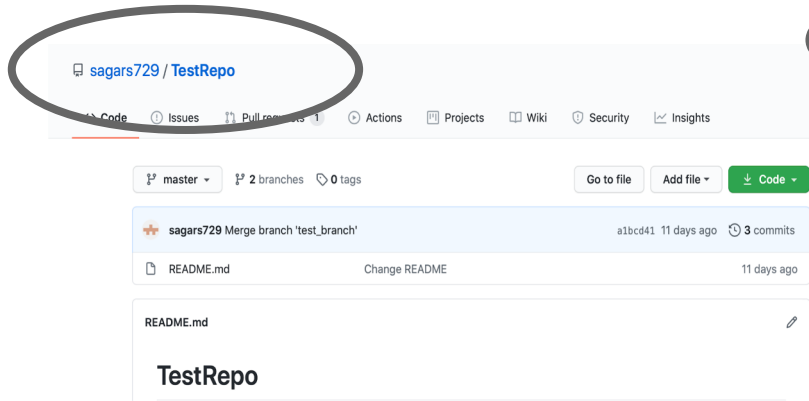
2 files changed, 4 insertions(+)
create mode 100644 .gitmodules
create mode 160000 Sub-Module-Example
$ █
```

Submodule

The screenshot shows a GitHub repository page for 'nkrishnan19 / TestRepo'. The repository is forked from 'sagars729/TestRepo'. The main branch is 'master', which is 2 branches ahead and 0 tags behind. The commit history shows a recent commit by 'Nandhini Krishnan and Nandhini Krishnan' titled 'added a sub module' with commit hash '2c7d2cd' and '4 commits' 1 minute ago. Below this, another commit by 'Nandhini Krishnan' is shown with the title 'added a sub module' and '1 minute ago'. The commit message for this commit is 'Sub-Module-Example @ e704d84'. An arrow points from the text on the left to this commit. Below the commit history, the 'README.md' file is visible, containing the text 'TestRepo' and 'Here is a change'.

The new files pushed will be in the submodule folder of the repository

Submodules & Forking



Notice that pushing changes to the forked repository does not update the original repository

Cloning the submodules

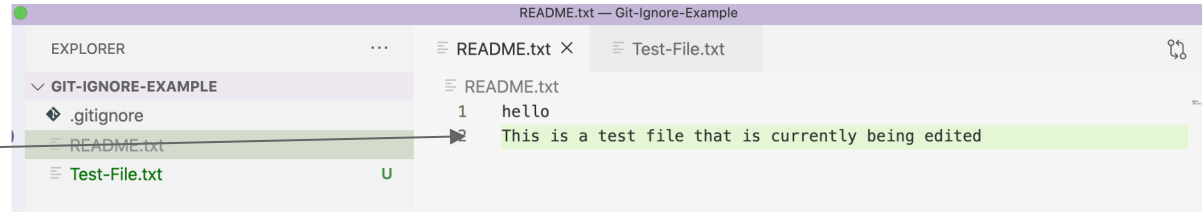
- `git clone --recurse-submodules -j8 'reponame'`

.gitignore

- Specifies untracked files to purposely ignore
- Set of patterns specified for Git to decide where or not to ignore a path
- Patterns read from a .gitignore file where the higher level files are overridden by those in lower level files
- The .gitignore file must be committed and updated with patterns as needed as changes are made
- Does not affect files that are currently tracked
- Common example: .gitignore files have patterns for files that generate project build

.gitignore & git status

2 files were modified
'README.txt' and
'Test-File.txt'



Because the
README.txt file is
specified in the git
ignore, though it
was modified it will
not show in the
cmd git status

```
[ $ git status  
On branch main  
Your branch is up to date with 'origin/main'.  
  
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
   Test-File.txt  
  
nothing added to commit but untracked files present (use "git add" to track)  
$ █ ]
```

Creating Repository with .gitignore

There is an option when creating a new repository to add a .gitignore component



A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * / **Repository name ***

Great repository names are short and memorable. Need inspiration? [How about bug-free-dollop?](#)

Description (optional)

Public
Anyone on the internet can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

Add a README file
This is where you can write a long description for your project. [Learn more.](#)

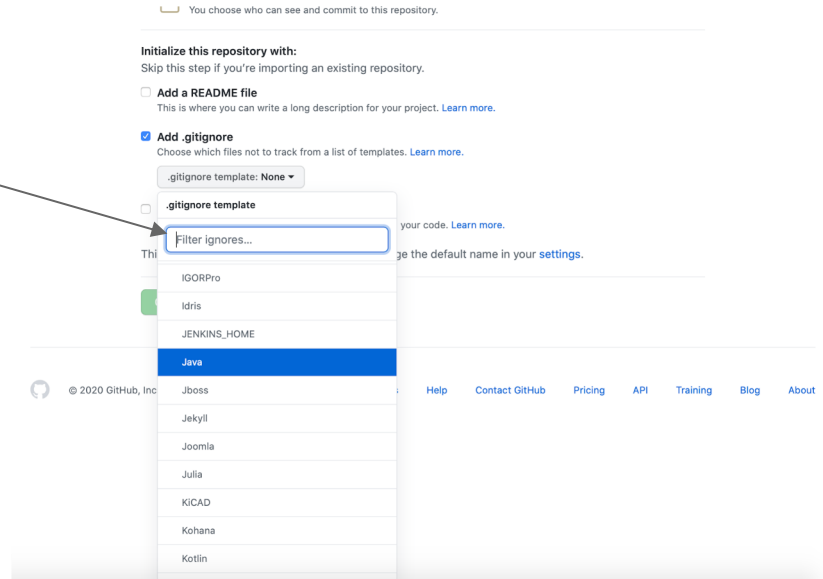
Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

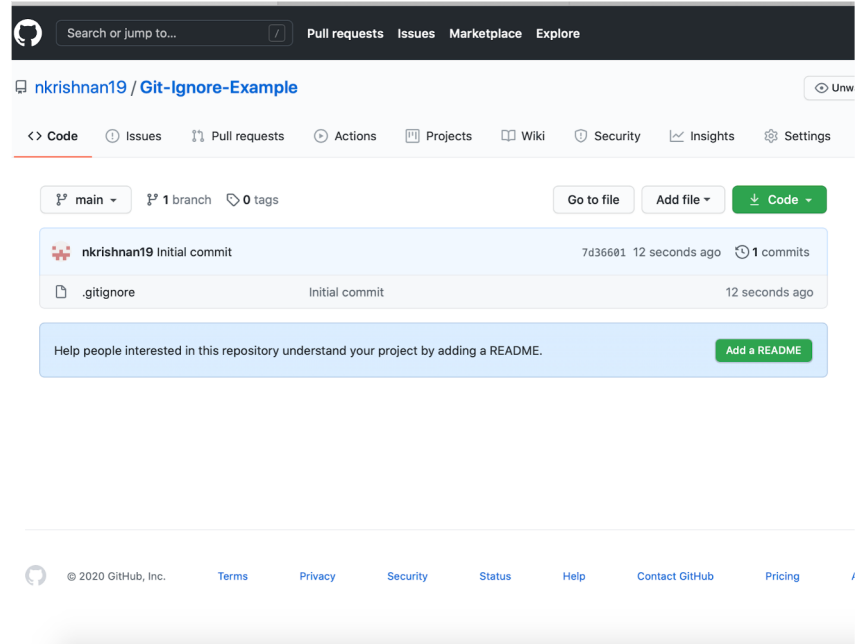
.gitignore

You can specify the template for the .gitignore component



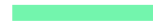
.gitignore

When creating a repository with a .gitignore component, it defaults to include a .ignore file instead of a readme file.





Advance Git Commands



Other useful git commands

Git Stash & Git Pop

- Git stash - allows you to temporarily store changes to your working tree so that it matches the HEAD commit
- Git stash list - shows you all the changes that are being 'stashed away'
- Git pop - restores the stashed modification to the working tree

```
[$ git add -A
[$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
   new file:   Dog.jpeg

$
```

File added to branch

```
[$ git stash
Saved working directory and index state WIP on master: a1bcd41 Merge branch 'tes
t_branch'
[$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
$
```

Stash remove file from commit

git fetch

- Downloads contents from remote repository to local repository
- Difference between git fetch & git pull:
 - git fetch: downloads from remote repository but does not update local repository working state
 - git pull: downloads contents from remote repository and merges

```
git fetch <remote>
```

Fetches all branches from remote repo

```
git fetch <remote> <branch>
```

Fetches a specific branch from remote repo

git log

Allows you to view your commit history

```
$ git log
commit ca82a6dff817ec66f44342007202690a93763949
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Mon Mar 17 21:52:11 2008 -0700
```

Change version number

```
commit 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Sat Mar 15 16:40:33 2008 -0700
```

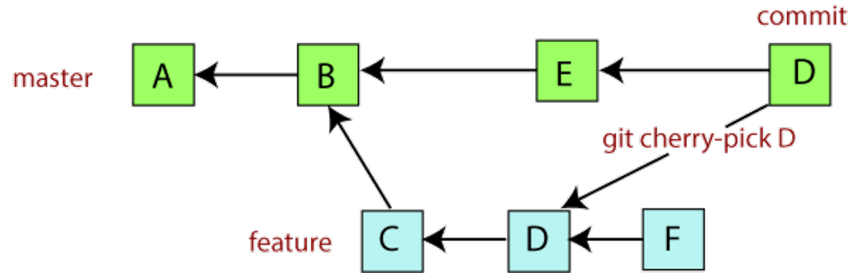
Remove unnecessary test

```
commit a11bef06a3f659402fe7563abf99ad00de2209e6
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Sat Mar 15 10:31:28 2008 -0700
```

Initial commit

Cherry Picking

- git-cherry-pick: allows you to “cherry pick” which commits should be applied
 - Decide which commits from a branch should be applied to another



Git Cherry Pick Demo

The Demo will show cherry picking by editing the README.md

```
[ $ git branch
    feature/change2
* master
[ $ git checkout -b feature/change
Switched to a new branch 'feature/change'
[ $ ls
  README.md
[ $ vim README.md
```


Git Cherry Pick Demo

Edit the ReadMe on
Master Branch

```
# TestRepo
Hello World - This is a README
MASTER BRANCH - Change 1
MASTER BRANCH - Change 2
```

Add & Commit
Changes to Master

```
modified: README.md

no changes added to commit (use "git add" and/or "git commit -a")
$ git add -A
$ git commit -m "master change 1"
[master e40ef8d] master change 1
1 file changed, 1 insertion(+)
$ vim README.md
$ git status
On branch master
Your branch is ahead of 'origin/master' by 8 commits.
(use "git push" to publish your local commits)

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:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
$ git add -A
$ git commit -m "master change 2"
```

Git Cherry Pick Demo

Create a Feature Branch and Edit
README.md

```
# TestRepo
Hello World - This is a README
MASTER BRANCH - Change 1
MASTER BRANCH - Change 2
FEATURE BRANCH - Change 3
```

Create README.md, add
and commit changes to
README.md

```
[$ git status
On branch feature/cherry
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:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
[$ git add -A
[$ git commit -m "feature change 3"
```


Git Cherry Pick Demo

Make more changes to the
Master Branch



```
# TestRepo
Hello World – This is a README
MASTER BRANCH – Change 1
MASTER BRANCH – Change 2
MASTER BRANCH – Change 4
```

add and commit changes



```
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 9 commits.
(use "git push" to publish your local commits)
$ vim README.md
$ git status
On branch master
Your branch is ahead of 'origin/master' by 9 commits.
(use "git push" to publish your local commits)

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:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
$ git add -A
$ git commit -m "master change 4"
[master e37a060] master change 4
 1 file changed, 1 insertion(+)
$
```

Git Cherry Pick Demo

Let's add 1 more change to
the Feature Branch



```
# TestRepo  
Hello World - This is a README  
MASTER BRANCH - Change 1  
MASTER BRANCH - Change 2  
FEATURE BRANCH - Change 3  
FEATURE Branch - Change 5
```

Git Cherry Pick Demo

git log on the feature branch shows the 2 commits we made to it... let's use the second to last commit as the one we want to cherry-pick

We need the commit address

```
feature/branch02
* feature/cherry
master
$ git log
commit 1cc224c1ec4bbdfaa705c1ab2cb6ceb51879c9a1 (HEAD -> feature/cherry)
Author:
Date:   Tue Dec 29 22:03:03 2020 -0500

    feature change 5

commit 3132daab78fb72ad33caf2d8ef32ff1b0b84cb07
Author:
Date:   Tue Dec 29 21:58:09 2020 -0500

    feature change 3
```

please note that using the last commit (though the demo uses it) is equivalent to a dangerous and typically cherry-pick is used when you want to use a middle commit.

Git Cherry Pick Demo

run the command line
git cherry-pick <commit-
address> to pick that
commit from the
feature branch

```
merge conflict  
$ git cherry-pick 3b32daab78fb72ad33caf2d8ef32ff1b0b84cb07  
Auto-merging README.md  
CONFLICT (content): Merge conflict in README.md  
error: could not apply 3b32daa... feature change 3  
hint: after resolving the conflicts, mark the corrected paths  
hint: with 'git add <paths>' or 'git rm <paths>'  
hint: and commit the result with 'git commit'  
$
```

Git Cherry Pick Demo

calling `git cherry-pick <commit-address>` again on the master branch will take that commit from the feature branch and put it in the master branch.


Doing this caused some conflicts we need to resolve

```
Otherwise, please use 'git cherry-pick --skip'
[$ git checkout master
Switched to branch 'master'
warning: cancelling a cherry picking in progress
Your branch is ahead of 'origin/master' by 10 commits.
  (use "git push" to publish your local commits)
[$ git cherry-pick 3b32daab78fb72ad33caf2d8ef32ff1b0b84cb07
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
error: could not apply 3b32daa... feature change 3
hint: after resolving the conflicts, mark the corrected paths
hint: with 'git add <paths>' or 'git rm <paths>'
hint: and commit the result with 'git commit'
$
```

Git Cherry Pick Demo

```
# TestRepo
Hello World - This is a README
MASTER BRANCH - Change 1
MASTER BRANCH - Change 2
<<<<<< HEAD
MASTER BRANCH - Change 4
=====
FEATURE BRANCH - Change 3
>>>>>> 3b32daa... feature change 3
```

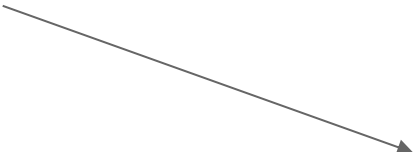
modify the
README.md to
resolve conflict



```
# TestRepo
Hello World - This is a README
MASTER BRANCH - Change 1
MASTER BRANCH - Change 2
MASTER BRANCH - Change 4
FEATURE BRANCH - Change 3
```


Git Cherry Pick Demo

The cherry-pick is seen as a change you have to add and commit.



```
On branch master
Your branch is ahead of 'origin/master' by 10 commits.
  (use "git push" to publish your local commits)

You are currently cherry-picking commit 3b32daa.
  (fix conflicts and run "git cherry-pick --continue")
  (use "git cherry-pick --skip" to skip this patch)
  (use "git cherry-pick --abort" to cancel the cherry-pick operation)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
   both modified:   README.md

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

git diff

- Displays the differences between two files
- Often used with git log and git status to analyze the state of the local repository

git diff

Here git diff shows the difference between 2 branches making changes to the README.md the red indicates what feature/change does not have that feature/change2 has and the green shows what feature/change does have and feature/change2 does not

```
[$ git diff feature/change feature/change2
diff --git a/README.md b/README.md
index 59b3943..7a33b9e 100644
--- a/README.md
+++ b/README.md
@@ -1,8 +1,5 @@
 # TestRepo
-Hello World - This is a README
-Lets Add some new Changes - Change 1
-Goodbye - Change 2
-I love Git! - Change 3
-Lets make another commit so we can really
-understand how cherry picking works! - change 7
-
+change1
+change2
+change3
+Here is a change
$
```

git rm

- Remove files or a collection of files
- Not a permanent change - updates working directory and staging directory
- Undo a git rm: git reset HEAD

```
HIMANSHU@HIMANSHU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git rm --cached newfile1.txt
rm 'newfile1.txt'

HIMANSHU@HIMANSHU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git status
on branch master
changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    deleted:   newfile1.txt

untracked files:
  (use "git add <file>..." to include in what will be committed)
    newfile1.txt
```

Clicker Quiz

Which is the main purpose of forking a repository ?

- A. Allowing for team members to access and collaborate on a project
- B. Creates Personal Development Environment
- C. Creates a local repository to push changes to the remote one
- D. Creates a project within the repository

Clicker Quiz

Which is the main purpose of forking a repository ?

- A. Allowing for team members to access and collaborate on a project
- B. Creates Personal Development Environment**
- C. Creates a local repository to push changes to the remote one
- D. Creates a project within the repository