
# CMPSC 104 – Document Engineering

Prof. Hang Zhao


ALLEGHENY COLLEGE

# Git Adding New Files

Copy the codes and save the file as index.html to our new folder

```
<!DOCTYPE html>
<html>
<head>
    <title>Document</title>
</head>
<body>
  <h1>Hello World</h1>
  <p> This is the first file in my new Git Repo. </p>
</body>
</html>
```

Command:

- ls (Linux)
- dir (Windows)

# Git Adding New Files

- Command: *git status*

```
git status
On branch master
No commits yet
Untracked files:   (use "git add ..." to include in what will be committed)     index.html
nothing added to commit but untracked files present (use "git add" to track)
```

# Git Status

Command: *git status*

```
git status
On branch master
No commits yet
Untracked files:   (use "git add ..." to include in what will be committed)     index.html
nothing added to commit but untracked files present (use "git add" to track)
```

Now Git is **aware** of the file, but has not **added** it to our repository!

Files in your Git repository folder can be in one of 2 states:

- Tracked - files that Git knows about and are added to the repository
- Untracked - files that are in your working directory, but not added to the repository

# Git Staging Environment

When you first add files to an empty repository, they are all untracked. To get Git to track them, you need to stage them, or add them to the staging environment.
**Staged** files are files that are ready to be **committed** to the repository you are working on

```
git add index.html
```

```
git status
On branch master
No commits yet
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   index.html
```

# Git Add More than One File

Now add all files in the current directory to the Staging Environment:

```
git add –all, git add –A or git add .
```

```
git status
On branch master
No commits yet
Changes to be committed:
 (use "git rm --cached <file>..." to unstage)
        new file:   index.html
```

# Git Commit

We are ready move from **stage** to **commit** for our repo.

When we commit, we should **always** include a **message**.

```
git commit -m "message"
```

```
git commit -m "First Commit!"
[master (root-commit) e53662d] First commit!
 2 files changed, 17 insertions(+)
         create mode 100644 index.html
```

# Git Commit without Stage

```
<!DOCTYPE html>
<html>
<head>
    <title>Document</title>
</head>
<body>
  <h1>Hello World</h1>
  <p> This is the first file in my new Git Repo. </p>
  <p>A new line in our file!</p>
</body>
</html>
```

The -a option will automatically stage every changed, already tracked file.

```
git commit -a -m "Updated index.html with a new line"
```

```
[master 4a31f44] Updated index.html with a new line
 1 file changed, 2 insertions(+)
```

# Git Commit Log

To view the history of commits for a repository, you can use the log command:

```
git log
commit 4a31f4459c70ce1bdc3a876b33976624d054d2ce (HEAD -> master)
Author: hangzhaogogogo <hzhao@allegheny.edu>
Date:   Sun Feb 11 13:28:37 2024 -0500

    Updated index.html with a new line

commit e53662d2c53bc2b5ef4266bfbd91e217aa61a84d
Author: hangzhaogogogo <hzhao@allegheny.edu>
Date:   Sun Feb 11 13:21:20 2024 -0500

    First commit!
```

# Git Help

If you are having trouble remembering commands or options for commands:

1. `git commit -help`
2. `git help –all or git help –a`

# Git Branch

## Workflow without Git:

1. Make copies of all the relevant files to avoid impacting the live version
2. Start working with the design and find that code depend on code in other files, that also need to be changed!
3. Make copies of the dependant files as well. Making sure that every file dependency references the correct file name
4. EMERGENCY! There is an unrelated error somewhere else in the project that needs to be fixed ASAP!
5. Save all your files, making a note of the names of the copies you were working on
6. Work on the unrelated error and update the code to fix it
7. Go back to the design, and finish the work there
8. Copy the code or rename the files, so the updated design is on the live version
9. (2 weeks later, you realize that the unrelated error was not fixed in the new design version because you copied the files before the fix)

## With Git:

1. With a new branch called new-design, edit the code directly without impacting the main branch
2. EMERGENCY! There is an unrelated error somewhere else in the project that needs to be fixed ASAP!
3. Create a new branch from the main project called small-error-fix
4. Fix the unrelated error and merge the small-error-fix branch with the main branch
5. You go back to the new-design branch, and finish the work there
6. Merge the new-design branch with main (getting alerted to the small error fix that you were missing)

# New Git Branch

- create a new branch: `git branch hello-world-images`
- Let's confirm that we have created a new branch:

```
git branch
hello-world-images
* master
```

the * beside master specifies that we are currently on that branch.

- Moving us **from** the current branch, **to** the one specified at the end of the command:

```
git checkout hello-world-images
Switched to branch 'hello-world-images'
```

Now we have moved our current workspace from the master branch, to the new branch

# New Git Branch

Now, let's add an image (img_hello_world.jpg) to the working folder and a line of code in the index.html file

```
<!DOCTYPE html>
<html>
<head>
        <title>Hello World!</title>
</head>
<body>
        <h1>Hello world!</h1>
        <div><img src="img_hello_world.jpg" style="width: 450px; height: auto;"></div>
        <p>This is the first file in my new Git Repo.</p>
        <p>A new line in our file!</p>
</body>
</html>
```

# New Git Branch

Check the status of the current branch:

```
git status
On branch hello-world-images
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:   index.html

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

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

# New Git Branch

So we need to add both files to the Staging Environment for this branch: `git add .`

Check the status of the branch: `git status`

```
On branch hello-world-images
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   img_hello_world.jpg
        modified:   index.html
```

Now, let's commit them to the branch: `git commit -m "Added image to Hello World"`

```
[hello-world-images e01302b] Added image to Hello World
 2 files changed, 1 insertions(+)
 create mode 100644 img_hello_world.jpg
```

# Switching Between Branches

let's list the files in the current directory: `ls or dir`

```
02/11/2024  05:39 PM            41,598 img_hello_world.jpg
02/11/2024  05:38 PM               313 index.html
```

Now, let's see what happens when we change branch to master:

```
git checkout master
Switched to branch 'master'
```

List the files in the current directory again:

```
02/11/2024  06:18 PM               215 index.html
```

# Emergency Branch

```
git checkout -b emergency-fix
Switched to a new branch 'emergency-fix'
```

**Note:** Using the -b option on checkout will create a new branch, and move to it, if it does not exist

# Emergency Branch

```
git checkout -b emergency-fix
```

Now we have created a new branch from master, and changed to it. We can safely fix the error without disturbing the other branches.

```
<!DOCTYPE html>
<html>
<head>
        <title>Hello World!</title>
</head>
<body>
        <h1>Hello world!</h1>
        <p>This is the first file in my new Git Repo.</p>
        <p>This line is here to show how merging works.</p>
</body>
</html>
```

# Emergency Branch

After fixing the error: `git status`

```
On branch emergency-fix
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:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
git add index.html
git commit -m "updated index.html with emergency fix"
[emergency-fix 9e679d2] updated index.html with emergency fix
 1 file changed, 1 insertion(+), 1 deletion(-)
```

# Emergency Branch Merge

**Merge the master and emergency-fix branches.**

First, we need to change to the master branch:

```
git checkout master
Switched to branch 'master'
```

Merge the current branch (master) with emergency-fix:

```
git merge emergency-fix
Updating 9d6066e..8d5f163
Fast-forward
 index.html | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

As master and emergency-fix are essentially the same now, we can delete emergency-fix, as it is no longer needed:

```
git branch -d emergency-fix
Deleted branch emergency-fix (was
9e679d2).
```

# Merge Conflict

Add another image file (img_hello_world_2.jpg) and change index.html

```
git checkout hello-world-images
Switched to branch 'hello-world-images'
```

```
<!DOCTYPE html>
<html>
<head>
        <title>Hello World!</title>
</head>
<body>
        <h1>Hello world!</h1>
        <div><img src="img_hello_world.jpg" style="width: 450px; height: auto;"></div>
        <p>This is the first file in my new Git Repo.</p>
        <p>A new line in our file!</p>
        <div><img src="img_hello_world_2.jpg" style="width: 450px; height: auto;"></div>
</body>
</html>
```

# Merge Conflict

```
stage and commit for this branch: git add .
                                  git commit -m "added new image“
                                  [hello-world-images 34aee12] Added new image
                                   2 files changed, 1 insertion(+)
                                   create mode 100644 img_hello_world_2.jpg

git checkout master
git merge hello-world-images
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```

# Merge Conflict

```
git status
On branch master
You have unmerged paths.
(fix conflicts and run "git commit")
(use "git merge --abort" to abort the merge)

Changes to be committed:
new file: img_hello_world.jpg
new file: img_hello_world_2.jpg

Unmerged paths:
(use "git add ..." to mark resolution)
both modified: index.html
```

# Merge Conflict

**Now we can stage index.html and check the status:**

```
git add index.html
git status
On branch master
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:
        new file:   img_hello_git.jpg
        new file:   img_hello_world.jpg
        modified:   index.html
```

**The conflict has been fixed, and we can use commit to conclude the merge:**

```
git commit -m "merged with hello-world after fixing conflicts“
```

**And delete the hello-world-images branch:**

```
git branch -d hello-world-images
Deleted branch hello-world-images (was 1f1584e).
```