

Module four

Working With Branches

1 - Moving, deleting and renaming files

2 - Switching branches without merging

3 - Merging

4 - Resolving conflicts

Create a new branch

```
15:18:56  alphabet_website  ↵\main  157ms
→ # Create a new branch and switch to it
15:19:46  alphabet_website  ↵\main
→ git branch test
15:20:04  alphabet_website  ↵\main  172ms
→ git checkout test
Switched to branch 'test'
15:20:17  alphabet_website  ↵\test  177ms
→ git branch
  main
* test
15:20:29  alphabet_website  ↵\test  233ms
→ # Let's take a look at the history of the test branch
15:21:27  alphabet_website  ↵\test
→ git log --oneline --decorate
ecd0ecb (HEAD -> test, origin/main, origin/HEAD, main) Revert "Unwanted changes"
76101f3 Unwanted changes
08f4586 (tag: v0.1) Added letter F
6a11142 Added letter E
b90e321 Added letter D
baeea75 Source code added
8fac59f Initial commit
```

Making changes moving all the stylesheets

```
15:30:43  alphabet_website ↴test ✘ 155ms
→ # Making changes moving all stylesheets of the website
15:31:51  alphabet_website ↴test
→ git mv style/*.css .
15:32:16  alphabet_website ↴test ✘ ~4 ✘ 138ms
→ tree

.
|-- README.md
|-- cities.css
|-- cities.html
|-- names.css
|-- names.html
|-- pilot.css
|-- pilots.html
|-- site.css
`-- style

1 directory, 8 files
```

Extract 'style' folder to fix all '*.html' files

```
<head>
  <title>Pilot's Alphabet</title>
  <meta charset="UTF-8">
  <link href="style/site.css" rel="stylesheet">
  <link href="style/pilot.css" rel="stylesheet">
</head>
```

```
<head>
  <title>Pilot's Alphabet</title>
  <meta charset="UTF-8">
  <link href="site.css" rel="stylesheet">
  <link href="pilot.css" rel="stylesheet">
</head>
```

Repeat this changes to cities.html and names.html

Adding changes to html files

```
06:34:32  alphabet_website ↴test ✘~4 | ✎~3 344ms
→ # Let's add the changes to the index to the next commit
06:35:28  alphabet_website ↴test ✘~4 | ✎~3
→ git add *.html
06:35:47  alphabet_website ↴test ✘~7 143ms
```

Removing style folder

```
06:38:53 alphabet_website ↵test ✓~7 145ms
→ # Remove the style folder
06:39:25 alphabet_website ↵test ✓~7
→ rmdir style
06:39:36 alphabet_website ↵test ✓~7 158ms
→ tree

.
|-- README.md
|-- cities.css
|-- cities.html
|-- names.css
|-- names.html
|-- pilot.css
|-- pilots.html
`-- site.css

0 directories, 8 files
```

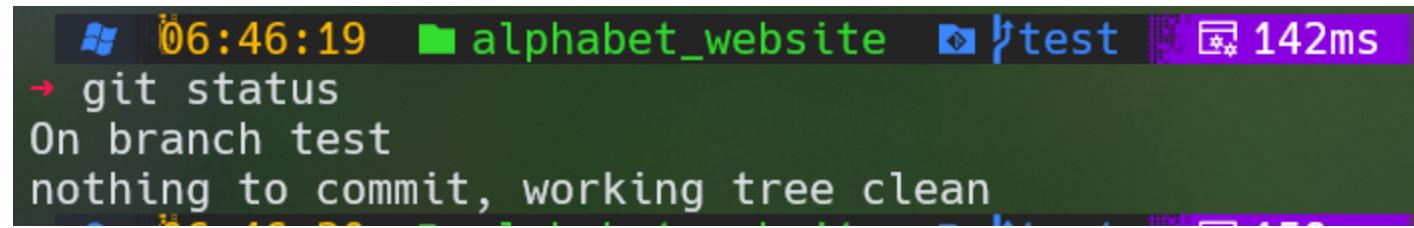
Checking status

```
06:41:21  alphabet_website  ↵test ✘ ~7  152ms
→ # Checking the status
06:41:45  alphabet_website  ↵test ✘ ~7
→ git status
On branch test
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    renamed:   style/cities.css -> cities.css
    modified:  cities.html
    renamed:   style/names.css -> names.css
    modified:  names.html
    renamed:   style/pilot.css -> pilot.css
    modified:  pilots.html
    renamed:   style/site.css -> site.css
```

Committing the changes

```
06:43:13 alphabet_website ↵test ✘~7 100ms
→ # Commit the changes
06:43:33 alphabet_website ↵test ✘~7
→ git commit -m "CSS files renamed"
[test 82a50eb] CSS files renamed
 7 files changed, 6 insertions(+), 6 deletions(-)
 rename style/cities.css => cities.css (100%)
 rename style/names.css => names.css (100%)
 rename style/pilot.css => pilot.css (100%)
 rename style/site.css => site.css (100%)
06:44:21 alphabet_website ↵test 254ms
→ git log --oneline --decorate
82a50eb (HEAD -> test) CSS files renamed
ecd0ecb (origin/main, origin/HEAD, main) Revert "Unwanted changes"
76101f3 Unwanted changes
08f4586 (tag: v0.1) Added letter F
6a11142 Added letter E
b90e321 Added letter D
baeea75 Source code added
8fac59f Initial commit
```

Status up to now



A screenshot of a Windows command prompt window. The title bar shows the path 'alphabet_website' and the branch 'test'. The system tray indicates the date as '06/19/2024' and the time as '142ms'. The command 'git status' is run, showing the output:

```
06:46:19 alphabet_website test 142ms
→ git status
On branch test
nothing to commit, working tree clean
```

Switching branches without merging

```
07:41:55 alphabet_website test 168ms
→ # Let's see the extent of the changes made to the test branch
07:42:44 alphabet_website test
→ tree

.
|-- README.md
|-- cities.css
|-- cities.html
|-- names.css
|-- names.html
|-- pilot.css
|-- pilots.html
`-- site.css

0 directories, 8 files
07:42:52 alphabet_website test 140ms
→ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

Alphabet_website

Nome	Status
📁 .git	✓
📁 style	✓
🌐 cities.html	✓
🌐 names.html	✓
🌐 pilots.html	✓
📄 README.md	✓

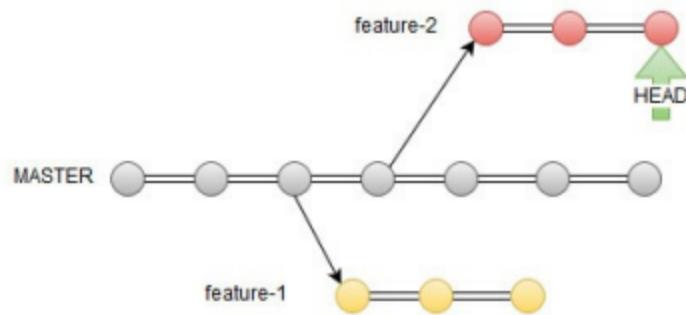
How is that possible?

Every time you switch branches Git will *fetch from the repository database* the contents of the working directory from the last commit on that branch.

When you run the *git checkout <branch>* command the HEAD pointer moves to the last commit of the branch you are checking out, and the content of the working directory reflects what is pointed to by HEAD.

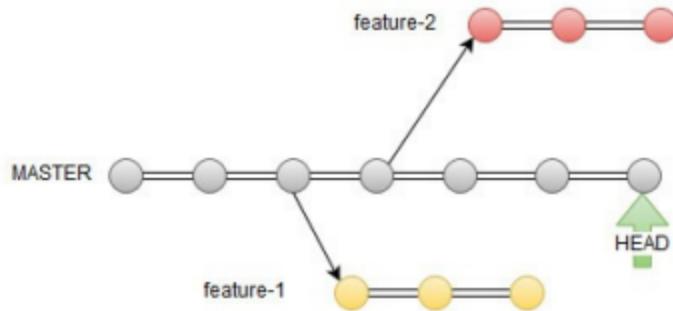
1- Graphic illustration of the switch process

The picture the branch feature-2 is the current branch



2- Graphic illustration of the switch process

The picture is showing when you command `'git checkout main'` the HEAD pointer moves to the last commit of the main branch and the contents of the working directory.



You can always verify which branch is current with either `git branch` or `git log`.

```
15:17:43  alphabet_website  ↵main  193ms
→ # Verifying which branch is current
15:18:07  alphabet_website  ↵main
→ git branch
* main
  test
15:18:14  alphabet_website  ↵main  218ms
→ # Using git log to verify
15:18:40  alphabet_website  ↵main
→ git log --oneline --decorate
ecd0ecb (HEAD -> main, origin/main, origin/HEAD) Revert "Unwanted changes"
76101f3 Unwanted changes
08f4586 (tag: v0.1) Added letter F
6a11142 Added letter E
b90e321 Added letter D
baeea75 Source code added
8fac59f Initial commit
```

Special note:

In case you have **uncommitted changes** when you switch to another branch, Git will try to merge those changes on to the target branch. If the changes are incompatible, then Git will not allow the switch. You can force the switch using the `-f` option e.g. `git checkout -f main` if you don't care about loosing the uncommitted changes. It is good practice to **switch branches only when the working directory is clean**.

Merging

In a typical Git workflow developers normally implement a feature request on a separate branch. This way they can freely experiment with changes without affecting the main code base. The changes are thoroughly tested in the feature branch. Only after successful testing the changes are merged on to the main development branch.

Implementing new features requests

First one is to add the letters **G,H,I** to the `alphabet_website`

Second one is to add the letters **J,K,L** to the `alphabet_website`

Creating new branches

```
 15:34:43 alphabet_website ↵main 157ms
→ # Checking if the main branch is clean
 15:35:34 alphabet_website ↵main
→ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
 15:35:44 alphabet_website ↵main 163ms
→ # Create a branch called g-h-i
 15:36:28 alphabet_website ↵main
→ git branch g-h-i
 15:36:44 alphabet_website ↵main 158ms
→ # Create a branch called j-k-l
 15:37:06 alphabet_website ↵main
→ git branch j-k-l
 15:37:22 alphabet_website ↵main 172ms
→ git branch
g-h-i
j-k-l
* main
test
```

Let's add the letters G-H-I to the branch g-h-i

To add the letters G-H-I first switch to the g-h-i branch

```
15:44:18  alphabet_website ↗main 144ms
→ # Switch to the branch g-h-i
15:44:54  alphabet_website ↗main
→ git checkout g-h-i
Switched to branch 'g-h-i'
```

Adding letters G,H,I to the alphabet_website

Add the letter G to pilots.html, cities.html and names.html (e.g. "Golf", "Greenville" and "Gloria"). Browse the website to test the changes, stage and commit with the comment "Added letter G".

Repeat the same process to add the letter H (e.g. "Hotel", "Houston" and "Henry") then stage and commit.

Repeat the same process to add the letter I (e.g. "India", "Illinois" and "Isabel") then stage and commit.

Adding letters G-H-I

```
15:55:10  alphabet_website  ↵g-h-i ✎~3  172ms
→ # Let's add and commit the letters added to g-h-i branch
15:56:12  alphabet_website  ↵g-h-i ✎~3
→ git commit -am "Add letter G"
[g-h-i 23508e0] Add letter G
 3 files changed, 4 insertions(+), 1 deletion(-)
15:56:43  alphabet_website  ↵g-h-i  174ms
→ git commit -am "Add letter H"
[g-h-i e73f3d9] Add letter H
 3 files changed, 3 insertions(+)
16:02:08  alphabet_website  ↵g-h-i  175ms
→ git commit -am "Add letter I"
[g-h-i 3615464] Add letter I
 3 files changed, 3 insertions(+)
```

g-h-i branch history look like:

```
16:12:05 alphabet_website g-h-i 143ms
→ git log --oneline
3615464 (HEAD -> g-h-i) Add letter I
e73f3d9 Add letter H
23508e0 Add letter G
ecd0ecb (origin/main, origin/HEAD, main, j-k-l) Revert "Unwanted changes"
76101f3 Unwanted changes
08f4586 (tag: v0.1) Added letter F
6a11142 Added letter E
b90e321 Added letter D
baeea75 Source code added
8fac59f Initial commit
16:12:17 alphabet_website g-h-i 191ms
→ git status
On branch g-h-i
nothing to commit, working tree clean
```

Note that the working directory is clean.

We are ready to merge the changes onto the main branch.

Switch back to main and check the history

```
 17:50:09  alphabet_website ↵g-h-i 151ms
→ # First switch back to main and check the history
 17:52:44  alphabet_website ↵g-h-i
→ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
 17:52:57  alphabet_website ↵main 148ms
→ git log --oneline
ecd0ecb (HEAD -> main, origin/main, origin/HEAD, j-k-l) Revert "Unwanted changes"
76101f3 Unwanted changes
08f4586 (tag: v0.1) Added letter F
6a11142 Added letter E
b90e321 Added letter D
baeea75 Source code added
8fac59f Initial commit
```

Let's browse the site

Now browse the Phonetic Website and you will find that it is still on the letter F. This will change once we have merged the changes from the g-h-i branch. The git merge command lets you integrate separate timelines of development into a single branch. To incorporate the feature you have implemented in the g-h-i branch into master we just need to run the following command:

`gti merge g-h-i`

Merging

```
18:03:01 alphabet_website ↵ main 213ms
→ # Let's merge g-h-i
18:04:23 alphabet_website ↵ main
→ git merge g-h-i
Updating ecd0ecb..3615464
Fast-forward
  cities.html | 3 +++
  names.html | 3 +++
  pilots.html | 5 +++.-
  3 files changed, 10 insertions(+), 1 deletion(-)
18:04:38 alphabet_website ↵ main 207ms
→ git log --oneline
3615464 (HEAD -> main, g-h-i) Add letter I
e73f3d9 Add letter H
23508e0 Add letter G
ecd0ecb (origin/main, origin/HEAD, j-k-l) Revert "Unwanted changes"
76101f3 Unwanted changes
08f4586 (tag: v0.1) Added letter F
6a11142 Added letter E
b90e321 Added letter D
baeea75 Source code added
8fac59f Initial commit
```

Resolving conflicts

Git does a good job of merging changes in different parts of the same file automatically.

Sometimes a conflict may arise during the merging operation. That happens when the changes you made collide with the existing code in the branch you are merging into.

Adding letters J,K,L to the website

Now we're going to implement the second feature request adding letters J,K,L to the website.

Using the branch j-k-l already created.

```
07:25:23  alphabet_website ↵\main 153ms
→ # Let's move to the j-k-l branch
07:25:50  alphabet_website ↵\main
→ git checkout j-k-l
Switched to branch 'j-k-l'
07:26:03  alphabet_website ↵\j-k-l 175ms
→ # Look the history of the branch
07:26:34  alphabet_website ↵\j-k-l
→ git log --oneline
ecd0ecb (HEAD -> j-k-l, origin/main, origin/HEAD) Revert "Unwanted changes"
76101f3 Unwanted changes
08f4586 (tag: v0.1) Added letter F
6a11142 Added letter E
b90e321 Added letter D
baeea75 Source code added
8fac59f Initial commit
```

Exercise

Add the letter J to `pilots.html`, `cities.html` and `names.html` ("Juliet", "Jackson" and "James") **immediately after** the letter F.

There is a gap caused by the missing G, H and I letters. Don't worry. We will sort out when merging the changes on to the main branch.

Letter J added to the website

<!-- ALPHABET START -->	19	<!-- ALPHABET START -->	19	<!-- ALPHABET START -->
Alpha	20	Atlanta	20	Andrew
Bravo	21	Boston	21	Brigitte
Charlie	22	Chicago	22	Charles
Delta	23	Detroit	23	David
Echo	24	Eldorado	24	Eva
Foxtrot	25	Fillmore	25	Fred
Juliet	26	Jackson	26	James
<!-- ALPHABET END -->	27	<!-- ALPHABET END -->	27	<!-- ALPHABET END -->

Stage and commit the letters J,K,L

```
07:52:19  alphabet_website ↵j-k-l ✎~3 170ms
→ # Let's add and commit letter "J"
07:52:48  alphabet_website ↵j-k-l ✎~3
→ git commit -am "Added letter J"
[j-k-l d42aba3] Added letter J
 3 files changed, 3 insertions(+)
07:53:20  alphabet_website ↵j-k-l 191ms
→ # Let's add and commit letter "K"
07:56:32  alphabet_website ↵j-k-l ✎~3
→ git commit -am "Added letter K"
[j-k-l 3be6dfa] Added letter K
 3 files changed, 3 insertions(+)
07:56:57  alphabet_website ↵j-k-l 239ms
→ Le
07:59:35  alphabet_website ↵j-k-l ✎~3
→ # Let's add and commit letter "L"
07:59:56  alphabet_website ↵j-k-l ✎~3
→ git commit -am "Added letter L"
[j-k-l f611d41] Added letter L
 3 files changed, 3 insertions(+)
```

The history of the branch j-k-l

```
08:06:13 alphabet_website ↵j-k-l 205ms
→ # The history of the branch j-k-l
08:06:56 alphabet_website ↵j-k-l
→ git log --oneline
f611d41 (HEAD -> j-k-l) Added letter L
3be6dfa Added letter K
d42aba3 Added letter J
ecd0ecb (origin/main, origin/HEAD) Revert "Unwanted changes"
76101f3 Unwanted changes
08f4586 (tag: v0.1) Added letter F
6a11142 Added letter E
b90e321 Added letter D
baeea75 Source code added
8fac59f Initial commit
08:07:06 alphabet_website ↵j-k-l 233ms
→ git status
On branch j-k-l
nothing to commit, working tree clean
```

Preparing to merge

We are ready now to merge the changes on to the master branch. First switch back to master and check the history:

```
08:16:42 alphabet_website ↵j-k-l 133ms
→ # Let's back to the main branch
08:17:06 alphabet_website ↵j-k-l
→ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 3 commits.
  (use "git push" to publish your local commits)
08:17:16 alphabet_website ↵main 187ms
→ # Let's check the history
08:17:44 alphabet_website ↵main
→
08:17:44 alphabet_website ↵main
→ git log --oneline
3615464 (HEAD -> main, g-h-i) Add letter I
e73f3d9 Add letter H
23508e0 Add letter G
ecd0ecb (origin/main, origin/HEAD) Revert "Unwanted changes"
76101f3 Unwanted changes
08f4586 (tag: v0.1) Added letter F
6a11142 Added letter E
b90e321 Added letter D
baeea75 Source code added
8fac59f Initial commit
```

Browsing the website we can find that it is still on letter "I".



Pilot's Alphabet

- Alpha
- Bravo
- Charlie
- Delta
- Echo
- Foxtrot
- Golf
- Hotel
- India

Merge the last feature request

Let's merge the feature we implemented on the j-k-l branch and see what happens:

```
08:31:17  alphabet_website  ↵\main  112ms
→ # Let's merge the last feature request
08:31:32  alphabet_website  ↵\main
→ git merge j-k-l
Auto-merging cities.html
CONFLICT (content): Merge conflict in cities.html
Auto-merging names.html
CONFLICT (content): Merge conflict in names.html
Auto-merging pilots.html
CONFLICT (content): Merge conflict in pilots.html
Automatic merge failed; fix conflicts and then commit the result.
08:31:47  alphabet_website  ↵\j-k-l into \main x3 | ✎ x3  179ms
```

Understanding the conflict

The git merge output is flagging conflicts on the three HTML files. That was to be expected as we made changes in the same lines of code on both branches.

Running git status will tell you that the files have been modified:

```
08:48:59  alphabet_website ↵ j-k-l into ↵ main x3 | ↵ x3 155ms
→ git status
On branch main
Your branch is ahead of 'origin/main' by 3 commits.
  (use "git push" to publish your local commits)

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

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified: cities.html
    both modified: names.html
    both modified: pilots.html

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

What is the way to resolve the conflicts

In case of conflicts during a merge, Git cleverly highlights the conflicts in the source code in each file keeping both changes (the one in the feature branch and the one in the target branch) leaving it to the developer to **resolve the conflicts manually**.

Fixing the three 'html' files

Open the files in your favourit text editor. The **conflicting changes will be clearly marked** by Git.

Let's fix `pilots.html` first. Open it in your favourite text editor. The conflicting changes will be clearly marked by Git as shown in the next slide.

```
<!-- ALPHABET START -->
<li>Alpha</li>
<li>Bravo</li>
<li>charlie</li>
<li>Delta</li>
<li>Echo</li>
<li>Foxtrot</li>
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<< HEAD (Current Change)
<li>Golf</li>
<li>Hotel</li>
<li>India</li>
<!-- ALPHABET END -->
=====
<li>Juliet</li>
<li>Kilo</li>
<li>Lima</li>
<!-- ALPHABET END -->
>>>>> j-k-l (Incoming Change)
```

Fixing (pilots.html)

The section of code between <<<<<< HEAD and ===== shows the code in the current branch. The section of code between ===== and >>>>>> j-k-l shows the conflicting changes merged from the j-k-l branch. Resolving the conflict in this case is quite easy. As we want to keep both changes we just need to delete the Git markings and retain the sequence of words in alphabetical order.

After fixing

```
<!-- ALPHABET START -->
<li>Alpha</li>
<li>Bravo</li>
<li>Charlie</li>
<li>Delta</li>
<li>Echo</li>
<li>Foxtrot</li>
<li>Golf</li>
<li>Hotel</li>
<li>India</li>
<li>Juliet</li>
<li>Kilo</li>
<li>Lima</li>
<!-- ALPHABET END -->
```

Repeat the process for cities.html and names.html.

Pilots site after fixing

Pilots	Cities	Names
--------	--------	-------

Pilot's Alphabet

- Alpha
- Bravo
- Charlie
- Delta
- Echo
- Foxtrot
- Golf
- Hotel
- India
- Juliet
- Kilo
- Lima

Fixing the others files

Delete also the Git markings in cities.html and names.html to resolve the conflicts and save. Browse the website to test. When all looks good, add the files to the index and commit as usual.

```
09:41:00 alphabet_website ↵ j-k-l into main x3 | x3 155ms
→ # Adding files to index and commit
09:41:59 alphabet_website ↵ j-k-l into main x3 | x3
→ git add *.html
09:42:16 alphabet_website ↵ j-k-l into main ~3 166ms
→ git commit -m "Resolved j-k-l merge conflict"
[main 28c8498] Resolved j-k-l merge conflict
09:42:46 alphabet_website ↵ main 174ms
→ git log --oneline --decorate
28c8498 (HEAD -> main) Resolved j-k-l merge conflict
f611d41 (j-k-l) Added letter L
3be6dfa Added letter K
d42aba3 Added letter J
3615464 (g-h-i) Add letter I
e73f3d9 Add letter H
23508e0 Add letter G
ecd0ecb (origin/main, origin/HEAD) Revert "Unwanted changes"
76101f3 Unwanted changes
08f4586 (tag: v0.1) Added letter F
6a11142 Added letter E
b90e321 Added letter D
baeea75 Source code added
8fac59f Initial commit
```

Final step

All the commits from the j-k-l branch have been integrated. The last commit is the one you have just executed following the merge conflict resolution. The `git merge` command is designed to preserve history whenever possible.

Tagging version

To conclude let's tag this version as v0.2 and update the remote repository on GitHub.

```
09:50:03  alphabet_website  ↵main  140ms
→ # Let's tag this version
09:50:30  alphabet_website  ↵main
→ git tag v0.2
09:50:49  alphabet_website  ↵main  111ms
→ git push origin main
Enter passphrase for key 'C:/users/clldu/onedrive/documentos/ssh_keys/.ssh/id_ed25519':
Enumerating objects: 39, done.
Counting objects: 100% (39/39), done.
Delta compression using up to 16 threads
Compressing objects: 100% (35/35), done.
Writing objects: 100% (35/35), 3.02 KiB | 1.51 MiB/s, done.
Total 35 (delta 22), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (22/22), completed with 3 local objects.
To github.com:linduarte/alphabet_website
  ecd0ecb..28c8498  main -> main
09:51:14  alphabet_website  ↵main  12.304s
→ git status
On branch main
Your branch is up to date with 'origin/main'
```