Git continued

Computer Science Practice and Experience: Development Basics
CS1XC3

Professor: Kevin Browne
E-mail: brownek@mcmaster.ca

This is a continuation of the examples from the previous lecture…

Visualizing branches

- We can visualize branches using git log
- **git log --all --graph**
  - Produces a graph like representation of the different branches in the commit log
- Let's try using this command...

```
[brownek@pascal codenew]$ git log --all --graph
* commit c881fd071c66c2e21f138c9ca06d29ea070129e9 (HEAD -> master)
| Author: brownek <brownek@pascal.cas.mcmaster.ca>
| Date:   Thu Mar 25 08:58:00 2021 -0400
|
|     modified another.txt on master branch
|
| * commit d4fccbf75d4f0c9725263389a16da38d001867d4 (dev)
|/  Author: brownek <brownek@pascal.cas.mcmaster.ca>
|   Date:   Thu Mar 25 08:53:45 2021 -0400
|
|       Modified file.txt on the dev branch
|
* commit a689dc297b8fa56179c86512624cdafae30465b5
| Author: brownek <brownek@pascal.cas.mcmaster.ca>
| Date:   Thu Mar 25 08:52:56 2021 -0400
|
|     Added another.txt to the repository
|
* commit 806f144a1fb19c246b18c669ee574c0348de6c6b
  Author: brownek <brownek@pascal.cas.mcmaster.ca>
  Date:   Thu Mar 25 08:51:58 2021 -0400

      Adding file.txt to the repository
```

Merging changes

- What if we want to merge changes from the dev branch into the master branch
- While on the master branch we can use the command: **git merge dev**
- Changes from the dev branch will be merged into the master branch
- Another useful command: **git branch**
  - Will show all branches, highlights current branch with *

Merging the dev branch…

```
[brownek@pascal codenew]$ git branch
  dev
* master
[brownek@pascal codenew]$ git merge dev
Merge made by the 'recursive' strategy.
 file.txt | 1 +
 1 file changed, 1 insertion(+)
[brownek@pascal codenew]$ git branch
  dev
* master
```

Merging branches

- When branches are merged a commit message is required
  - By default git will generally launch **vi** to create commit messages
- Remember when using vi you can escape "writing mode" by hitting escape
  - And then :wq will allow you to save and quit

Now changes from both branches are present…

```
[brownek@pascal codenew]$ cat file.txt
Added some content to file.txt.
[brownek@pascal codenew]$ cat another.txt
Making modifications to another.txt.
```

Merging the branches creates a new commit…

```
[brownek@pascal codenew]$ git log --all
commit c6fb6d1f3cac519cd2d7faf222de470a32c6903b (HEAD -> master)
Merge: c881fd0 d4fccbf
Author: brownek <brownek@pascal.cas.mcmaster.ca>
Date:   Thu Mar 25 09:05:06 2021 -0400

    Merge branch 'dev'

commit c881fd071c66c2e21f138c9ca06d29ea070129e9
Author: brownek <brownek@pascal.cas.mcmaster.ca>
Date:   Thu Mar 25 08:58:00 2021 -0400

    modified another.txt on master branch

commit d4fccbf75d4f0c9725263389a16da38d001867d4 (dev)
Author: brownek <brownek@pascal.cas.mcmaster.ca>
Date:   Thu Mar 25 08:53:45 2021 -0400

    Modified file.txt on the dev branch
```

Graph after merging

```
*   commit 033302d2c30550815c2505611461d228e0a4f343 (HEAD -> master)
|\  Merge: b572b42 027e2d9
| | Author: brownek <brownek@pascal.cas.mcmaster.ca>
| | Date:   Fri Mar 26 09:01:06 2021 -0400
| |
| |     Merge branch 'dev'
| |
| * commit 027e2d961523c206b7eed22962b0556285a0f99f (dev)
| | Author: brownek <brownek@pascal.cas.mcmaster.ca>
| | Date:   Fri Mar 26 08:59:29 2021 -0400
| |
| |     Modified file.txt on the dev branch
| |
* | commit b572b4231a62b90304ea0e8a85760a8029241dca
|/  Author: brownek <brownek@pascal.cas.mcmaster.ca>
|   Date:   Fri Mar 26 09:00:23 2021 -0400
|
|       Modified another.txt on the master branch
|
* commit ec9535a6039a55e0ae6f3e7ea491cd842faac37a
```

How did the merge work?

- In this case, a different file was modified in each branch, and so a merge was trivial... new versions of the files from the old branch can overwrite the old versions from master

- What if the ***same*** files are modified?
  - git will attempt to merge the file content
  - This may or may not "work"... if some lines were added to different sections of each file it may be OK
  - If the same lines were modified or lines in the same place of the file are modified we may have a **conflict**
  - A conflict is when git cannot figure out how to merge files together
  - Conflicts are up to the developer to figure out!

Note: we can still use the dev branch after a merge…

```
[brownek@pascal dev2]$ git switch dev
Switched to branch 'dev'
[brownek@pascal dev2]$ touch file2.txt
[brownek@pascal dev2]$ git add .
[brownek@pascal dev2]$ git status
On branch dev
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   file2.txt

[brownek@pascal dev2]$ git commit -m "Adding file2.txt to dev branch"
[dev fcc81bd] Adding file2.txt to dev branch
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file2.txt
```

After running git log --all --graph

```
* commit fcc81bd81e9839185bbbd6b6f4f881e35f36bcdb (HEAD -> dev)
| Author: brownek <brownek@pascal.cas.mcmaster.ca>
| Date:   Fri Mar 26 09:07:09 2021 -0400
|
|     Adding file2.txt to dev branch
|
| *   commit 033302d2c30550815c2505611461d228e0a4f343 (master)
| |\  Merge: b572b42 027e2d9
| |/  Author: brownek <brownek@pascal.cas.mcmaster.ca>
|/|   Date:   Fri Mar 26 09:01:06 2021 -0400
| |
| |       Merge branch 'dev'
| |
* | commit 027e2d961523c206b7eed22962b0556285a0f99f
| | Author: brownek <brownek@pascal.cas.mcmaster.ca>
| | Date:   Fri Mar 26 08:59:29 2021 -0400
| |
| |     Modified file.txt on the dev branch
| |
| * commit b572b4231a62b90304ea0e8a85760a8029241dca
|/  Author: brownek <brownek@pascal.cas.mcmaster.ca>
```

We can switch to master and merge the change…

```
[brownek@pascal dev2]$ git switch master
Switched to branch 'master'
[brownek@pascal dev2]$ git merge dev
Merge made by the 'recursive' strategy.
 file2.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file2.txt
```

git log --all --graph visualization

```
*   commit bc43cafaf4298350db1d835937b598e48f8a69cd (HEAD -> master)
|\  Merge: 033302d fcc81bd
| | Author: brownek <brownek@pascal.cas.mcmaster.ca>
| | Date:   Fri Mar 26 09:09:47 2021 -0400
| |
| |     Merge branch 'dev'
| |
| * commit fcc81bd81e9839185bbbd6b6f4f881e35f36bcdb (dev)
| | Author: brownek <brownek@pascal.cas.mcmaster.ca>
| | Date:   Fri Mar 26 09:07:09 2021 -0400
| |
| |     Adding file2.txt to dev branch
| |
* | commit 033302d2c30550815c2505611461d228e0a4f343
|\| Merge: b572b42 027e2d9
| | Author: brownek <brownek@pascal.cas.mcmaster.ca>
| | Date:   Fri Mar 26 09:01:06 2021 -0400
| |
| |     Merge branch 'dev'
| |
| * commit 027e2d961523c206b7eed22962b0556285a0f99f
```

What if we modify the contents of the same file on each branch?

Let's edit file.txt, first on the dev branch...

```
Modifying file.txt on the dev branch.
Added a line to the bottom of file.txt on the dev branch.
~
~
~
~
~
```

Commit the modified file.txt on the dev branch…

```
[brownek@pascal dev2]$ git switch dev
Switched to branch 'dev'
[brownek@pascal dev2]$ vi file.txt
[brownek@pascal dev2]$ cat file.txt
Modifying file.txt on the dev branch.
Added a line to the bottom of file.txt on the dev branch.
[brownek@pascal dev2]$ git add .
[brownek@pascal dev2]$ git commit -m "Added line to bottom of file.txt on dev
branch"
[dev b4d9c58] Added line to bottom of file.txt on dev branch
 1 file changed, 1 insertion(+)
```

Next let's modify file.txt on the master branch...

```
Adding a line to the top of file.txt on the master branch.
Modifying file.txt on the dev branch.
~
~
~
~
~
~
```

Switch to the master branch, modify file.txt and commit it...

```
[brownek@pascal dev2]$ git switch master
Switched to branch 'master'
[brownek@pascal dev2]$ vi file.txt
[brownek@pascal dev2]$ cat file.txt
Adding a line to the top of file.txt on the master branch.
Modifying file.txt on the dev branch.
[brownek@pascal dev2]$ git add .
[brownek@pascal dev2]$ git commit -m "Added line to the top of file.txt on the
 master branch"
[master 1e48e4f] Added line to the top of file.txt on the master branch
 1 file changed, 1 insertion(+)
```

What will happen now after we merge?

The file is different on both branches.

And the merge… just works!

```
[brownek@pascal dev2]$ git merge dev
Auto-merging file.txt
Merge made by the 'recursive' strategy.
 file.txt | 1 +
 1 file changed, 1 insertion(+)
[brownek@pascal dev2]$ cat file.txt
Adding a line to the top of file.txt on the master branch.
Modifying file.txt on the dev branch.
Added a line to the bottom of file.txt on the dev branch.
```

The file just contains both modifications now…

git log --all --graph visualization

```
*   commit 4b6a1d05a24ca7dc2f2064e2c97cda0ce497c263 (HEAD -> master)
|\  Merge: 1e48e4f b4d9c58
| | Author: brownek <brownek@pascal.cas.mcmaster.ca>
| | Date:   Fri Mar 26 09:24:13 2021 -0400
| |
| |     Merge branch 'dev'
| |
| * commit b4d9c587088d9eec95cb62787261b09aa73bb981 (dev)
| | Author: brownek <brownek@pascal.cas.mcmaster.ca>
| | Date:   Fri Mar 26 09:16:18 2021 -0400
| |
| |     Added line to bottom of file.txt on dev branch
| |
* | commit 1e48e4ffa1ab5a7388dfd80cb8b113bb33200d1b
| | Author: brownek <brownek@pascal.cas.mcmaster.ca>
| | Date:   Fri Mar 26 09:20:29 2021 -0400
| |
| |     Added line to the top of file.txt on the master branch
| |
* | commit bc43cafaf4298350db1d835937b598e48f8a69cd
|\| Merge: 033302d fcc81bd
```

Merging

- The reason it "just works" even though we've modified the same file is that git's algorithm can figure out what was changed
  - It's a bit like diff!
- In this case, we have non-overlapping changes
  - A line was changed before a line of text
  - A line was changed after a line of text
- What is happening on the **dev** branch after a merge?

If we switch back to the dev branch and look at file.txt...

```
[brownek@pascal dev2]$ git switch dev
Switched to branch 'dev'
[brownek@pascal dev2]$ cat file.txt
Modifying file.txt on the dev branch.
Added a line to the bottom of file.txt on the dev branch.
```

It's missing a line, but how? Didn't we merge the branches?!

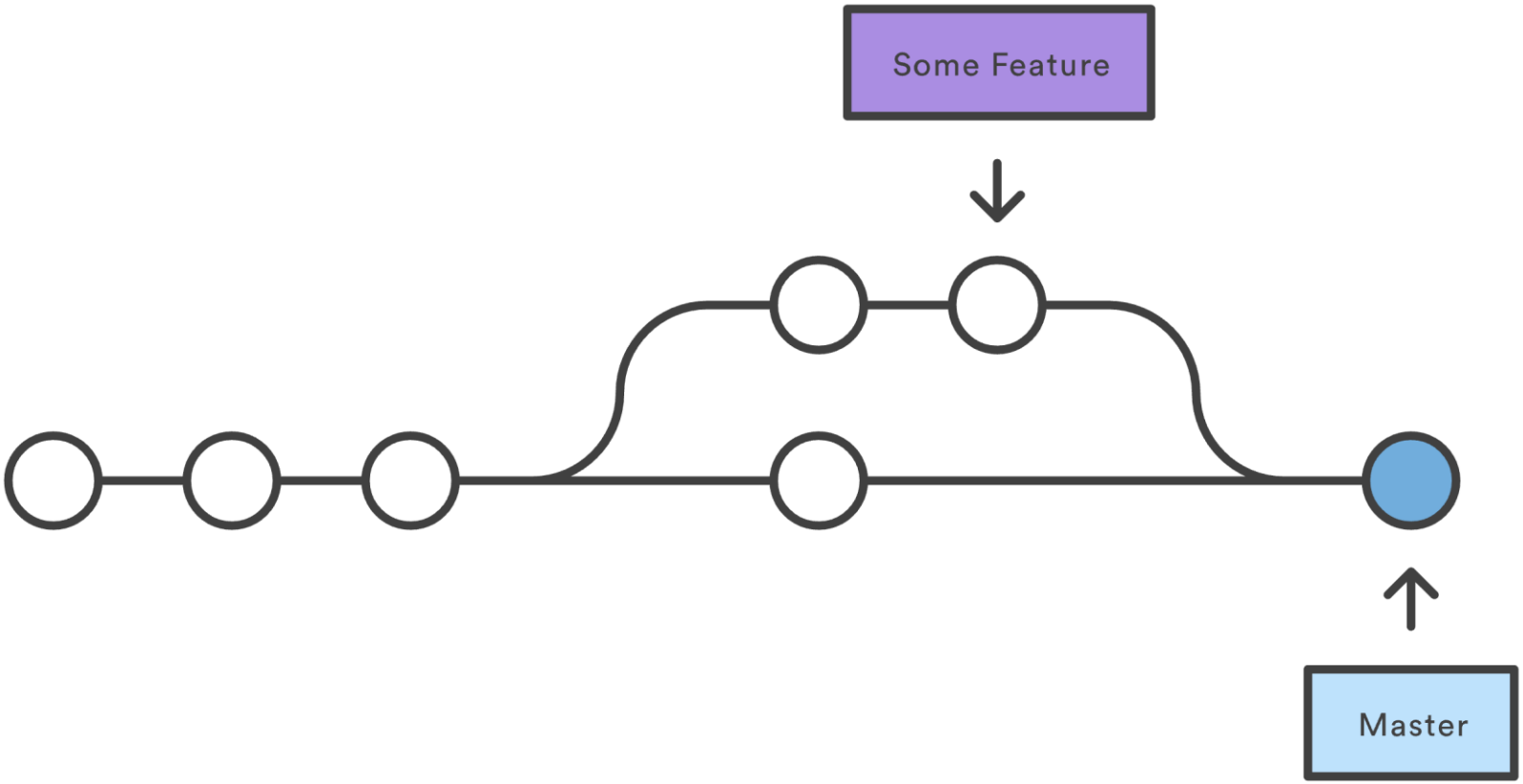
Merging

- We merge the changes from a branch (like dev) into another branch (like master)
- The branches themselves don't "merge" so much as we merge changes from a branch into another
- Let's visualize this to see what's going on...

Notice how the head of the dev branch is still the last commit on the dev branch?

```
*   commit 4b6a1d05a24ca7dc2f2064e2c97cda0ce497c263 (master)
|\  Merge: 1e48e4f b4d9c58
| | Author: brownek <brownek@pascal.cas.mcmaster.ca>
| | Date:   Fri Mar 26 09:24:13 2021 -0400
| |
| |     Merge branch 'dev'
| |
| * commit b4d9c587088d9eec95cb62787261b09aa73bb981 (HEAD -> dev)
| | Author: brownek <brownek@pascal.cas.mcmaster.ca>
| | Date:   Fri Mar 26 09:16:18 2021 -0400
| |
| |     Added line to bottom of file.txt on dev branch
| |
* | commit 1e48e4ffa1ab5a7388dfd80cb8b113bb33200d1b
| | Author: brownek <brownek@pascal.cas.mcmaster.ca>
| | Date:   Fri Mar 26 09:20:29 2021 -0400
| |
| |     Added line to the top of file.txt on the master branch
| |
```

After a merge, our branches look like this...



Source: https://www.atlassian.com/git/tutorials/using-branches/git-merge

Branches

- Branches are effectively pointers to commits
- And after a merge, our dev branch pointer is still pointing to the previous commit on this branch
- If we were to make changes on this branch, they wouldn't reflect the differences on the master branch

How do we handle this situation?

- While it's not ***universally*** practiced, it's actually very common practice to just delete a branch after a feature has been created
  - i.e. branches are created to make changes
  - Once the change is made, the branch is merged to master
  - After which the branch is not used again, instead it is deleted
- We could also merge the master branch to the dev branch
- My advice? Use new branches for new features!

Merging the master branch to the dev branch…

```
[brownek@pascal dev2]$ git branch
* dev
  master
[brownek@pascal dev2]$ git merge master -m "Merging master to dev branch"
Updating b4d9c58..4b6a1d0
Fast-forward (no commit created; -m option ignored)
 another.txt | 1 +
 file.txt    | 1 +
 2 files changed, 2 insertions(+)
[brownek@pascal dev2]$ cat file.txt
Adding a line to the top of file.txt on the master branch.
Modifying file.txt on the dev branch.
Added a line to the bottom of file.txt on the dev branch.
```

Our dev branch was also missing the change to another.txt from before…

Now both branches refer to the same commit…

```
*   commit 4b6a1d05a24ca7dc2f2064e2c97cda0ce497c263 (HEAD -> dev, master)
|\  Merge: 1e48e4f b4d9c58
| | Author: brownek <brownek@pascal.cas.mcmaster.ca>
| | Date:   Fri Mar 26 09:24:13 2021 -0400
| |
| |     Merge branch 'dev'
| |
| * commit b4d9c587088d9eec95cb62787261b09aa73bb981
| | Author: brownek <brownek@pascal.cas.mcmaster.ca>
| | Date:   Fri Mar 26 09:16:18 2021 -0400
| |
| |     Added line to bottom of file.txt on dev branch
| |
* | commit 1e48e4ffa1ab5a7388dfd80cb8b113bb33200d1b
| | Author: brownek <brownek@pascal.cas.mcmaster.ca>
| | Date:   Fri Mar 26 09:20:29 2021 -0400
| |
| |     Added line to the top of file.txt on the master branch
| |
```

We can also delete a branch entirely with **git branch –d branch_name**

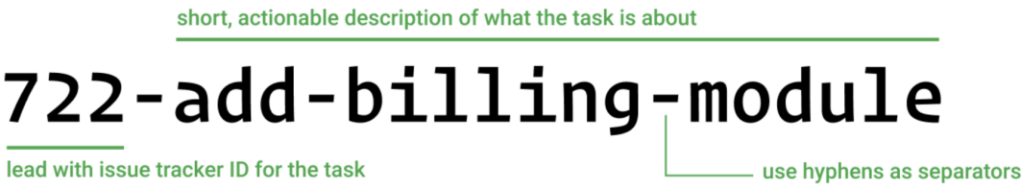
```
[brownek@pascal dev2]$ git branch
  dev
* master
[brownek@pascal dev2]$ git branch -d dev
Deleted branch dev (was 4b6a1d0).
```

Note that our commit log still reflects the previous commit structure, what we've ***really*** deleted is the branch pointer

```
*   commit 4b6a1d05a24ca7dc2f2064e2c97cda0ce497c263 (HEAD -> master)
|\  Merge: 1e48e4f b4d9c58
| | Author: brownek <brownek@pascal.cas.mcmaster.ca>
| | Date:   Fri Mar 26 09:24:13 2021 -0400
| |
| |     Merge branch 'dev'
| |
| * commit b4d9c587088d9eec95cb62787261b09aa73bb981
| | Author: brownek <brownek@pascal.cas.mcmaster.ca>
| | Date:   Fri Mar 26 09:16:18 2021 -0400
| |
| |     Added line to bottom of file.txt on dev branch
| |
* | commit 1e48e4ffa1ab5a7388dfd80cb8b113bb33200d1b
| | Author: brownek <brownek@pascal.cas.mcmaster.ca>
| | Date:   Fri Mar 26 09:20:29 2021 -0400
| |
| |     Added line to the top of file.txt on the master branch
| |
* | commit bc43cafaf4298350db1d835937b598e48f8a69cd
```

Many projects have a **naming convention** for branches (e.g. for each feature)

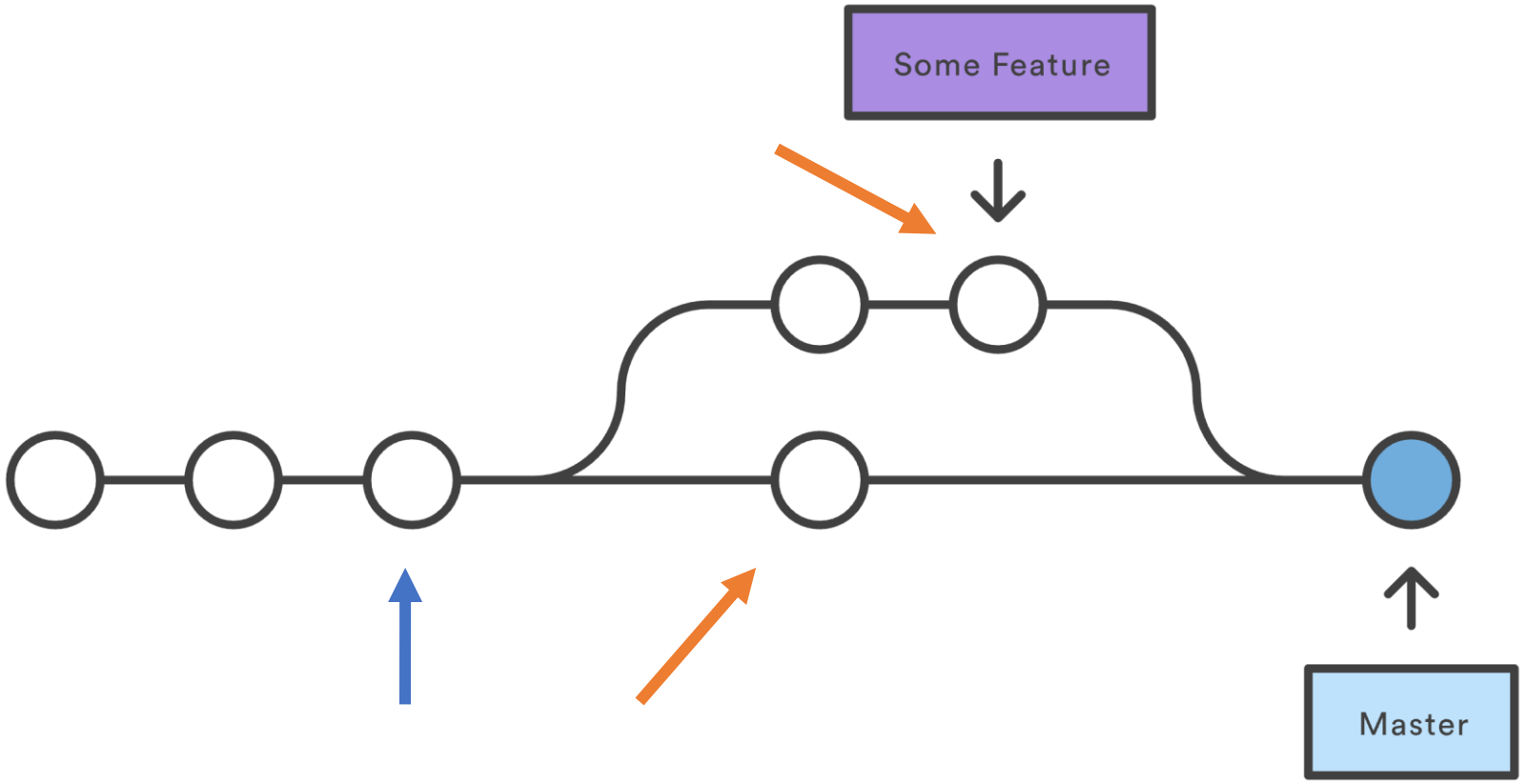
We use git branches at DeepSource to organize ongoing work to ensure that software delivery stays effective. If you use git today, there are high chances that you're either using the famed git-flow or the more recent GitHub flow. Both these workflows depend extensively on using branches effectively — and naming a new branch is something many developers struggle with.



A consistent branch naming convention is part of code review best practices, and can make life much more easier for anyone who's collaborating and reviewing your code, in addition to using static analysis tools.

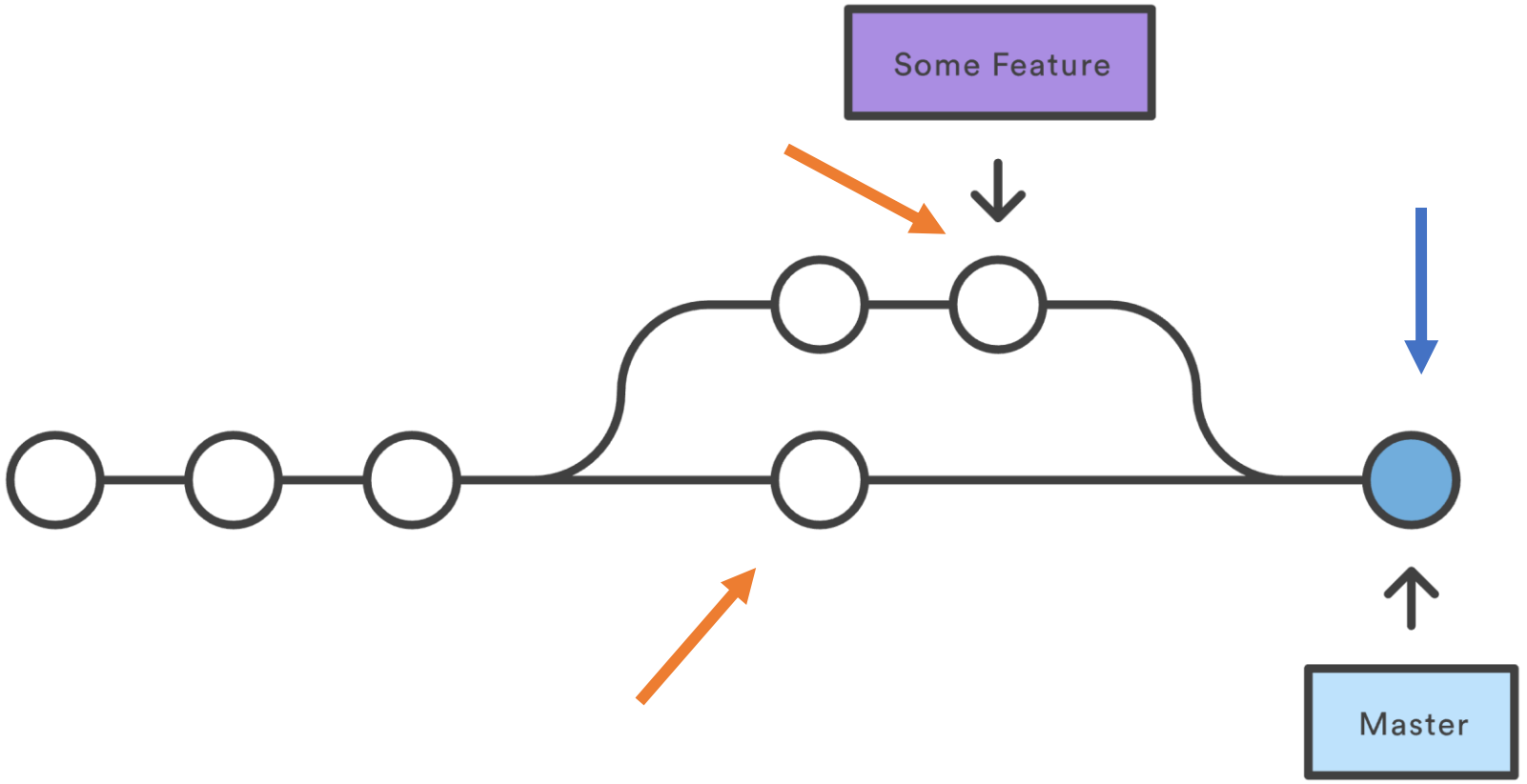
Source: https://deepsource.io/blog/git-branch-naming-conventions/

During the last merge, since the last common ancestor commit of the two branches, changes had been made on **both** branches...



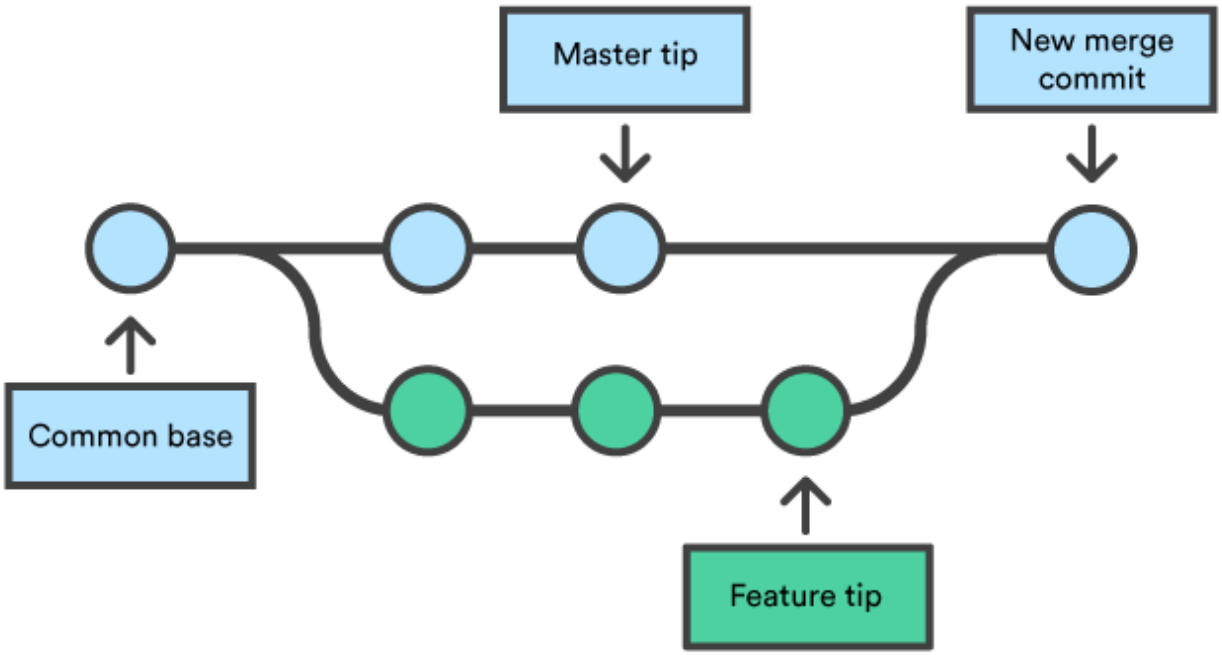
Source: https://www.atlassian.com/git/tutorials/using-branches/git-merge

When a merge took place, the changes from the dev branch *really did* have to be integrated with the changes on the master branch



Source: https://www.atlassian.com/git/tutorials/using-branches/git-merge

This type of merge is called a **3-way merge** because the heads (tips) of the two branches and the common ancestor are all used to produce the merge commit



Source: https://www.atlassian.com/git/tutorials/using-branches/git-merge

What if only the "feature" branch had changes, but not the master?

Let's make a new branch and see what happens...

```
[brownek@pascal dev2]$ git branch
* master
[brownek@pascal dev2]$ git branch new_feature
[brownek@pascal dev2]$ git branch
* master
  new_feature
[brownek@pascal dev2]$ git switch new_feature
Switched to branch 'new_feature'
```

We'll modify file.txt again...

```
Adding another line to the file on our new_feature branch.

Adding a line to the top of file.txt on the master branch.
Modifying file.txt on the dev branch.
Added a line to the bottom of file.txt on the dev branch.
~
~
~
~
~
~
~
```

And we'll commit our change on the new_feature branch

```
[brownek@pascal dev2]$ vi file.txt
[brownek@pascal dev2]$ cat file.txt
Adding another line to the file on our new_feature branch.

Adding a line to the top of file.txt on the master branch.
Modifying file.txt on the dev branch.
Added a line to the bottom of file.txt on the dev branch.
[brownek@pascal dev2]$ git add .
[brownek@pascal dev2]$ git commit -m "Modified file.txt on the new_fe
ature branch"
[new_feature 28b01f2] Modified file.txt on the new_feature branch
 1 file changed, 2 insertions(+)
```

And here's what our commit structure looks like now...

```
* commit 28b01f2311f0991489c83e47d5d1806260e2ebba (HEAD -> new_feature)
| Author: brownek <brownek@pascal.cas.mcmaster.ca>
| Date:   Fri Mar 26 10:40:04 2021 -0400
|
|     Modified file.txt on the new_feature branch
|
*   commit 4b6a1d05a24ca7dc2f2064e2c97cda0ce497c263 (master)
|\  Merge: 1e48e4f b4d9c58
| | Author: brownek <brownek@pascal.cas.mcmaster.ca>
| | Date:   Fri Mar 26 09:24:13 2021 -0400
| |
| |     Merge branch 'dev'
| |
| * commit b4d9c587088d9eec95cb62787261b09aa73bb981
| | Author: brownek <brownek@pascal.cas.mcmaster.ca>
| | Date:   Fri Mar 26 09:16:18 2021 -0400
| |
| |     Added line to bottom of file.txt on dev branch
```

How would a merge work?

- Conducting a "merge" of new_feature to master would be ***trivial*** at this point

- The new_feature branch is just a modification of the previous master branch

- In other words, git doesn't really have to do a proper "merge" as in a 3-way merge
  - It can simply move the master branch pointer to the head of the new_feature branch!

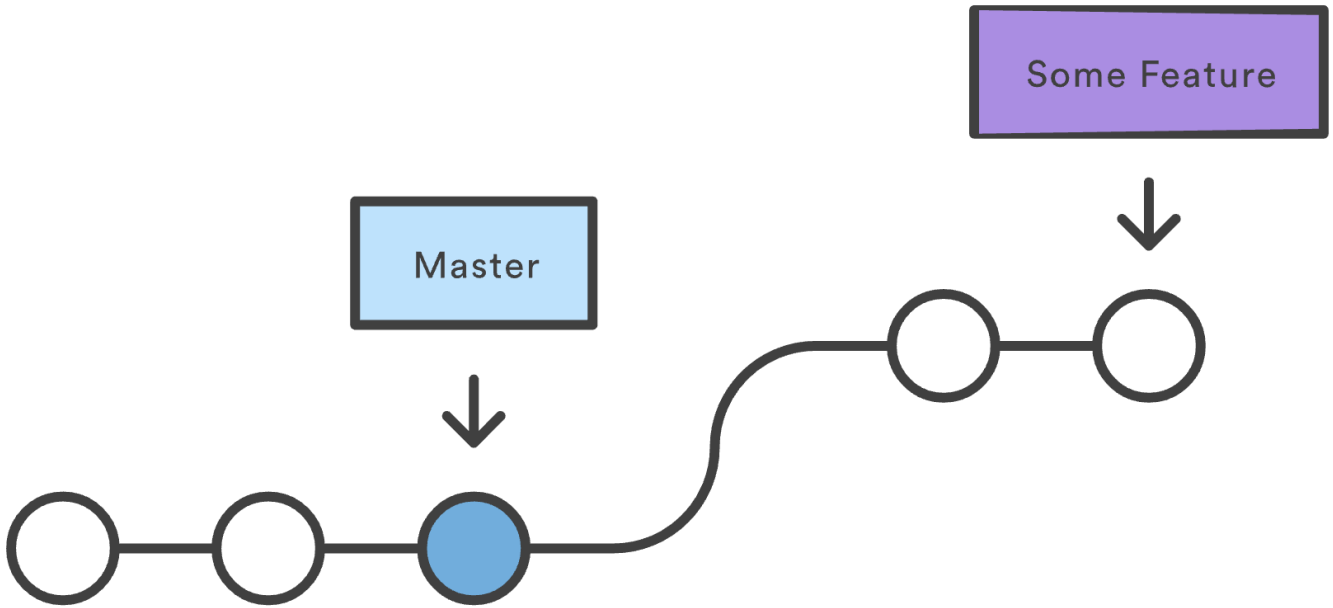
Merging new_feature to master

```
[brownek@pascal dev2]$ git switch master
Switched to branch 'master'
[brownek@pascal dev2]$ git merge new_feature
Updating 4b6a1d0..28b01f2
Fast-forward
 file.txt | 2 ++
 1 file changed, 2 insertions(+)
```

All git does is set the master branch pointer to point to the head of new_feature

```
* commit 28b01f2311f0991489c83e47d5d1806260e2ebba (HEAD -> master, new_feature)
| Author: brownek <brownek@pascal.cas.mcmaster.ca>
| Date:   Fri Mar 26 10:40:04 2021 -0400
|
|     Modified file.txt on the new_feature branch
|
*   commit 4b6a1d05a24ca7dc2f2064e2c97cda0ce497c263
|\  Merge: 1e48e4f b4d9c58
| | Author: brownek <brownek@pascal.cas.mcmaster.ca>
| | Date:   Fri Mar 26 09:24:13 2021 -0400
| |
| |     Merge branch 'dev'
| |
| * commit b4d9c587088d9eec95cb62787261b09aa73bb981
| | Author: brownek <brownek@pascal.cas.mcmaster.ca>
| | Date:   Fri Mar 26 09:16:18 2021 -0400
| |
| |     Added line to bottom of file.txt on dev branch
```

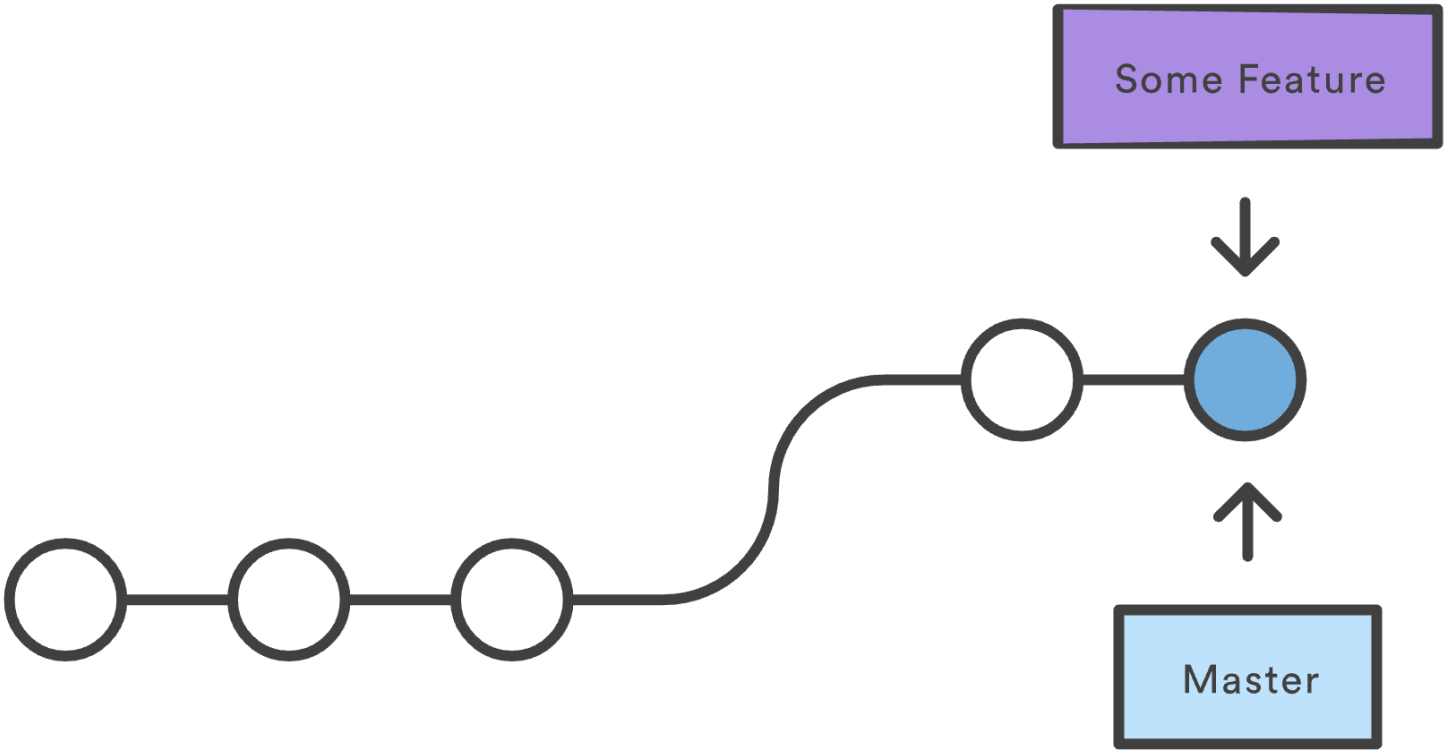
Before merging…



Source: https://www.atlassian.com/git/tutorials/using-branches/git-merge

...and after merging...

This type of merge is called a **fast-forward** merge!



Source: https://www.atlassian.com/git/tutorials/using-branches/git-merge

Conflicts

- The last time we modified two files on two different branches and conducted a merge it worked fine
  - Because we modified non-overlapping areas of the file
- What if we had modified overlapping areas of the file?
  - We might get a **conflict** when we conduct the merge
  - Git will signal a conflict has occurred and ask us to resolve the conflict
  - The file will be modified in a way that highlights the two differences between the versions, and we'll need to fix the issue and commit again

Let's create another branch another_feature...

```
[brownek@pascal dev2]$ git branch another_feature
[brownek@pascal dev2]$ git switch another_feature
Switched to branch 'another_feature'
[brownek@pascal dev2]$ vi file.txt
```

And we'll modify the 4th and same line of file.txt...

```
Adding another line to the file on our new_feature branch.

Adding a line to the top of file.txt on the master branch.
MODIFY THE SAME LINE DIFFERENTLY Modifying file.txt on the dev branch.
Added a line to the bottom of file.txt on the dev branch.
~
~
~
~
~
```

And then we'll commit the change...

```
[brownek@pascal dev2]$ vi file.txt
[brownek@pascal dev2]$ git add .
[brownek@pascal dev2]$ git commit -m "Made a modification to file.txt on another_
feature branch"
[another_feature a940e70] Made a modification to file.txt on another_feature bran
ch
 1 file changed, 1 insertion(+), 1 deletion(-)
```

And then we'll modify the same line in the same file differently on the master branch...

```
Adding another line to the file on our new_feature branch.

Adding a line to the top of file.txt on the master branch.
Modifying file.txt on the dev branch. MODIFY THE LINE DIFFERENTLY ON MASTER.
Added a line to the bottom of file.txt on the dev branch.
~
~
```

And we'll commit the change on the master branch…

```
[brownek@pascal dev2]$ git switch master
Switched to branch 'master'
[brownek@pascal dev2]$ vi file.txt
[brownek@pascal dev2]$ git add .
[brownek@pascal dev2]$ git commit -m "Modified file.txt on the
 master branch too"
[master 55162dc] Modified file.txt on the master branch too
 1 file changed, 1 insertion(+), 1 deletion(-)
```

And we're back to a situation of requiring a 3-way merge to occur

```
* commit 55162dc3fa202ef581bcd7f5c26af821dd10c086 (HEAD -> master)
| Author: brownek <brownek@pascal.cas.mcmaster.ca>
| Date:   Fri Mar 26 10:55:41 2021 -0400
|
|     Modified file.txt on the master branch too
|
| * commit a940e70e05203d718c72ad96886d206eff792d75 (another_feature)
|/  Author: brownek <brownek@pascal.cas.mcmaster.ca>
|   Date:   Fri Mar 26 10:53:42 2021 -0400
|
|       Made a modification to file.txt on another_feature branch
|
* commit 28b01f2311f0991489c83e47d5d1806260e2ebba (new_feature)
| Author: brownek <brownek@pascal.cas.mcmaster.ca>
| Date:   Fri Mar 26 10:40:04 2021 -0400
|
|     Modified file.txt on the new_feature branch
|
```

Now when we attempt a merge, we get a **conflict**…

```
[brownek@pascal dev2]$ git merge another_feature
Auto-merging file.txt
CONFLICT (content): Merge conflict in file.txt
Automatic merge failed; fix conflicts and then commit the result.
[brownek@pascal dev2]$
```

We need to open file.txt, fix the conflict, and then try to commit again!

When we open file.txt we will the conflict highlighted with <<<< content1 ======= content2 >>>>>

```
Adding another line to the file on our new_feature branch.

Adding a line to the top of file.txt on the master branch.
<<<<<<< HEAD
Modifying file.txt on the dev branch. MODIFY THE LINE DIFFERENTLY ON MASTER.
=======
MODIFY THE SAME LINE DIFFERENTLY Modifying file.txt on the dev branch.
>>>>>>> another_feature
Added a line to the bottom of file.txt on the dev branch.
~
~
~
~
~
```

There is no magic formula for fixes, you need to use your brain as a programmer to figure out what should be done. :-)

To fix the issue we remove the conflict <,>,= characters and provide what we believe to be the correct resolution in terms of the text itself.

```
Adding another line to the file on our new_feature branch.

Adding a line to the top of file.txt on the master branch.
MODIFY THE SAME LINE DIFFERENTLY Modifying file.txt on the dev branch.  MODIFY TH
E LINE DIFFERENTLY ON MASTER.
Added a line to the bottom of file.txt on the dev branch.
~
~
~
~
```

We can then add and commit the file again…

```
[brownek@pascal dev2]$ git add file.txt
[brownek@pascal dev2]$ git commit -m "fixed conflict in file.txt"
[master 8d6ab03] fixed conflict in file.txt
```

And we can see that our merge is now successful!

```
*   commit 8d6ab03fc9a3d49f8e71ad081ad927126b6e0f15 (HEAD -> master)
|\  Merge: 55162dc a940e70
| | Author: brownek <brownek@pascal.cas.mcmaster.ca>
| | Date:   Fri Mar 26 11:02:47 2021 -0400
| |
| |     fixed conflict in file.txt
| |
| * commit a940e70e05203d718c72ad96886d206eff792d75 (another_feature)
| | Author: brownek <brownek@pascal.cas.mcmaster.ca>
| | Date:   Fri Mar 26 10:53:42 2021 -0400
| |
| |     Made a modification to file.txt on another_feature branch
| |
* | commit 55162dc3fa202ef581bcd7f5c26af821dd10c086
|/  Author: brownek <brownek@pascal.cas.mcmaster.ca>
|   Date:   Fri Mar 26 10:55:41 2021 -0400
|
|       Modified file.txt on the master branch too
|
```

Conflicts

- Fixing conflicts is not fun, it's probably one of the least enjoyable activities as a programmer
  - You're not creating something new so much as fixing something that's broken
  - And it may involve figuring out someone else's new code!
- Teams develop workflows that will minimize conflicts
  - e.g. working on different areas of code in parallel
- But they are inevitable, and you'll need to learn how to resolve them

Viewing old versions of files

- What if we want to view an old commit in the repository?

- Perhaps we want to see how something was once implemented, or we want to roll back to a previous version

- We can use **git checkout ID**
  - Where ID is the 40-bit ID

Let's make more changes to file.txt... first adding one line, and then doing a commit...

```
I've made yet another change to file.txt.

Adding another line to the file on our new_feature branch.

Adding a line to the top of file.txt on the master branch.
MODIFY THE SAME LINE DIFFERENTLY Modifying file.txt on the dev branch.  MODIFY TH
E LINE DIFFERENTLY ON MASTER.
Added a line to the bottom of file.txt on the dev branch.
~
~
~
```

And then another line... and another commit...

```
One more change to file.txt.

I've made yet another change to file.txt.

Adding another line to the file on our new_feature branch.

Adding a line to the top of file.txt on the master branch.
MODIFY THE SAME LINE DIFFERENTLY Modifying file.txt on the dev branch.  MODIFY TH
E LINE DIFFERENTLY ON MASTER.
Added a line to the bottom of file.txt on the dev branch.
~
~
~
```

So on the command-line we've done the following adds and commits...

```
[brownek@pascal dev2]$ vi file.txt
[brownek@pascal dev2]$ git add .
[brownek@pascal dev2]$ git commit -m "Yet another change made to file.txt"
[master cb5f6bb] Yet another change made to file.txt
 1 file changed, 2 insertions(+)
[brownek@pascal dev2]$ vi file.txt
[brownek@pascal dev2]$ git add .
[brownek@pascal dev2]$ git commit -m "Made one more change to file.txt"
[master 973bf7a] Made one more change to file.txt
 1 file changed, 2 insertions(+)
```

And our commit structure looks like this… what if we want to go back to the cb5f6b**** commit?

```
* commit 973bf7ae4ff0f420ddd555639815799c8e2b2af9 (HEAD -> master)
| Author: brownek <brownek@pascal.cas.mcmaster.ca>
| Date:   Fri Mar 26 11:07:36 2021 -0400
|
|     Made one more change to file.txt
|
* commit cb5f6bb33c839dfbb43c4942d89b16e9d37c56c5
| Author: brownek <brownek@pascal.cas.mcmaster.ca>
| Date:   Fri Mar 26 11:07:00 2021 -0400
|
|     Yet another change made to file.txt
|
*   commit 8d6ab03fc9a3d49f8e71ad081ad927126b6e0f15
|\  Merge: 55162dc a940e70
| | Author: brownek <brownek@pascal.cas.mcmaster.ca>
| | Date:   Fri Mar 26 11:02:47 2021 -0400
| |
| |     fixed conflict in file.txt
```

We can run git checkout with this commit ID…

```
[brownek@pascal dev2]$ git checkout cb5f6bb33c839dfbb43c4942d89b16e9d37c56c5
Note: switching to 'cb5f6bb33c839dfbb43c4942d89b16e9d37c56c5'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at cb5f6bb Yet another change made to file.txt
```

And now we've rolled back to the previous version of file.txt!

```
[brownek@pascal dev2]$ cat file.txt
I've made yet another change to file.txt.

Adding another line to the file on our new_feature branch.

Adding a line to the top of file.txt on the master branch.
MODIFY THE SAME LINE DIFFERENTLY Modifying file.txt on the dev branch.  MODIFY TH
E LINE DIFFERENTLY ON MASTER.
Added a line to the bottom of file.txt on the dev branch.
```

And we have that the "head" pointer is pointing to the commit before master...

```
* commit 973bf7ae4ff0f420ddd555639815799c8e2b2af9 (master)
| Author: brownek <brownek@pascal.cas.mcmaster.ca>
| Date:   Fri Mar 26 11:07:36 2021 -0400
|
|     Made one more change to file.txt
|
* commit cb5f6bb33c839dfbb43c4942d89b16e9d37c56c5 (HEAD)
| Author: brownek <brownek@pascal.cas.mcmaster.ca>
| Date:   Fri Mar 26 11:07:00 2021 -0400
|
|     Yet another change made to file.txt
|
```

We can "look around" this code if we like, if we wish to make commits, we can create a branch and do commits to retain changes somewhere...

```
[brownek@pascal dev2]$ git checkout cb5f6bb33c839dfbb43c4942d89b16e9d37c56c5
Note: switching to 'cb5f6bb33c839dfbb43c4942d89b16e9d37c56c5'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at cb5f6bb Yet another change made to file.txt
```