

Basic git concepts #3

Majid Babaei





#1: How can we skip the staging area?

#2: How can we remove a file from WD and SA?

#3: How can we ignore some files?

#4: How can we get a file status in git?

#5: How can we compare different versions?

#6: How can we view content of a commit?

#7: How can we restore a file using the git history?

#7: How can we restore a file using the git history?



We removed a file accidentally and committed the changes!

```
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git rm file3.txt
rm 'file3.txt'
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> ls
```

Directory: C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice

| Mode | | LastWriteTime | Length | Name |
|--------|--|--------------------|--------|------------|
| d---- | | 9/10/2023 12:01 AM | | build |
| d---- | | 9/10/2023 12:01 AM | | logs |
| -a---- | | 9/10/2023 12:03 AM | 28 | .gitignore |
| -a---- | | 9/9/2023 11:51 PM | 28 | file1.txt |
| -a---- | | 9/10/2023 1:27 PM | 99 | file2.txt |
| -a---- | | 9/10/2023 12:02 AM | 14 | logs.txt |

```
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    file3.txt
```

```
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git add .\file3.txt
fatal: pathspec '.\file3.txt' did not match any files
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git commit -m "file3 deleted!"
[master e9e96b0] file3 deleted!
1 file changed, 2 deletions(-)
delete mode 100644 file3.txt
```

Let's look at the history

We need to decide what version of the file3.txt we would like to restore!

If you don't remember what it contains,

```
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git log  
commit e9e96b0d60507d56dcbec3dde05781b338bdbc32 (HEAD -> master)  
Author: Majid Babaei <16mb64@gmail.com>  
Date: Sun Sep 10 13:51:29 2023 -0400
```

file3 deleted!

```
commit 50e403925544ba793011f393b825c6b71b31d6b3  
Author: Majid Babaei <16mb64@gmail.com>  
Date: Sun Sep 10 13:31:19 2023 -0400
```

file3.txt updated

```
commit 55d7fd07131bf06fa6a30ad19f638990f6a2b42b  
Author: Majid Babaei <16mb64@gmail.com>  
Date: Sun Sep 10 13:30:18 2023 -0400
```

file3.txt added

```
commit 58b7f73f8705a22c003c7cab44bed77f5056c18a  
Author: Majid Babaei <16mb64@gmail.com>  
Date: Sun Sep 10 13:28:13 2023 -0400
```

file2.txt converted to utf-8 format!

```
commit bbbde312a1444b4ed08dc8555e98fcd0050f13dd  
Author: Majid Babaei <16mb64@gmail.com>  
Date: Sun Sep 10 13:23:54 2023 -0400
```

new line added to file2.txt

```
commit 75a3dedd5277111bb29477d99bf27c998e119cf1  
Author: Majid Babaei <16mb64@gmail.com>  
Date: Sun Sep 10 13:22:28 2023 -0400
```

file2 updated!

Let's check the content of the file in some commits

```
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git show HEAD~1:file3.txt
file3
Let's add this line to file3.txt
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git show HEAD~2:file3.txt
file3
```

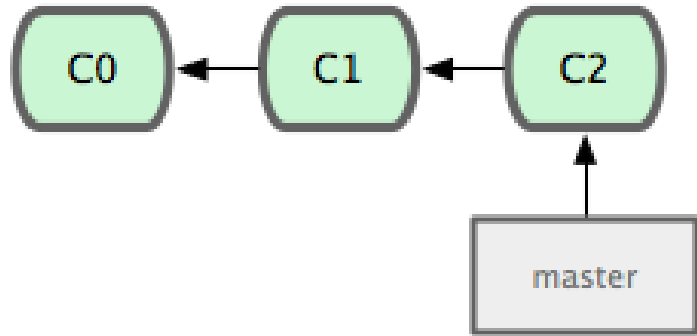
Okey! The version of the file in HEAD~2 looks good!

Let's define the source in the *git restore* command

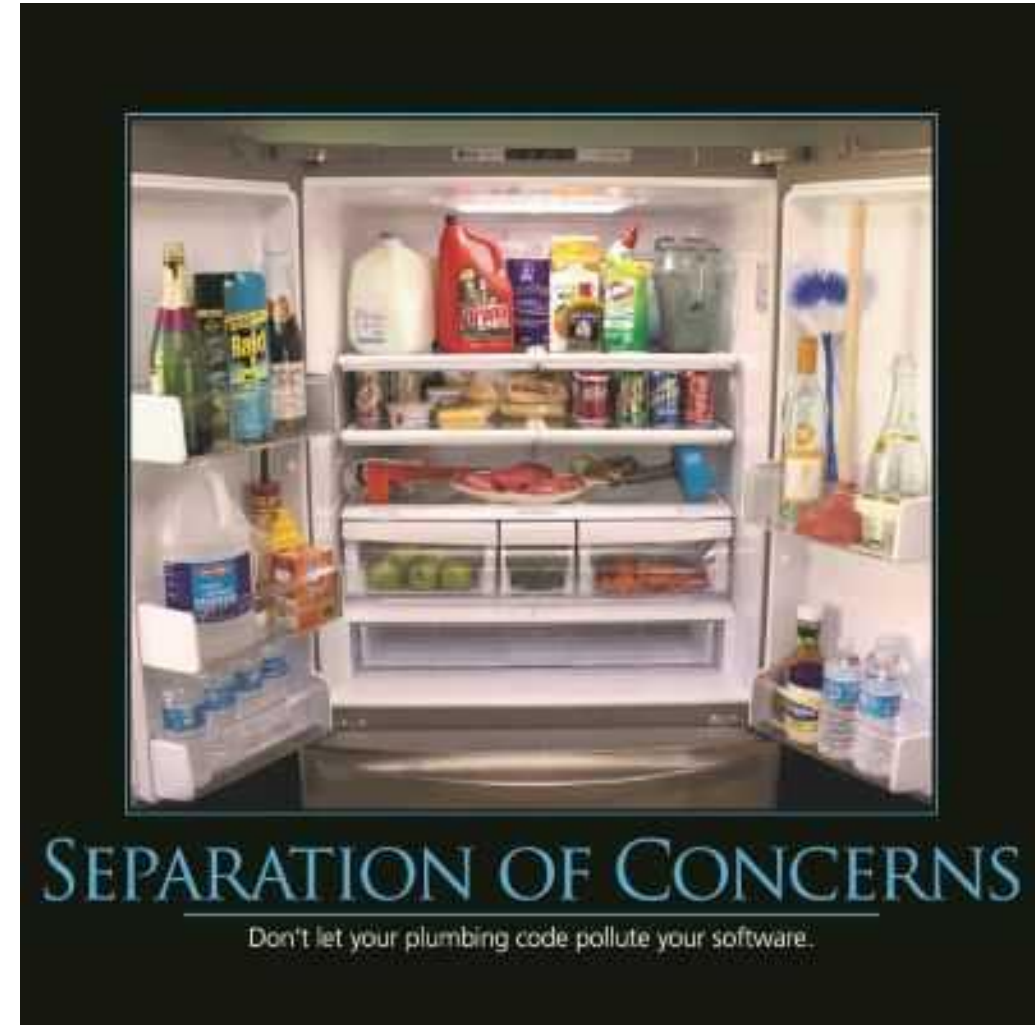
```
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git restore --source=HEAD~2 file3.txt
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    file3.txt

nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> cat .\file3.txt
file3
```


Single development line!



- What is the idea of having a single dev. Line?
- What are the Pros / Cons?
- The notion of *Separation of Concerns*
- Avoid co-locating different concerns within the design or code



DevOps



What is Branching?

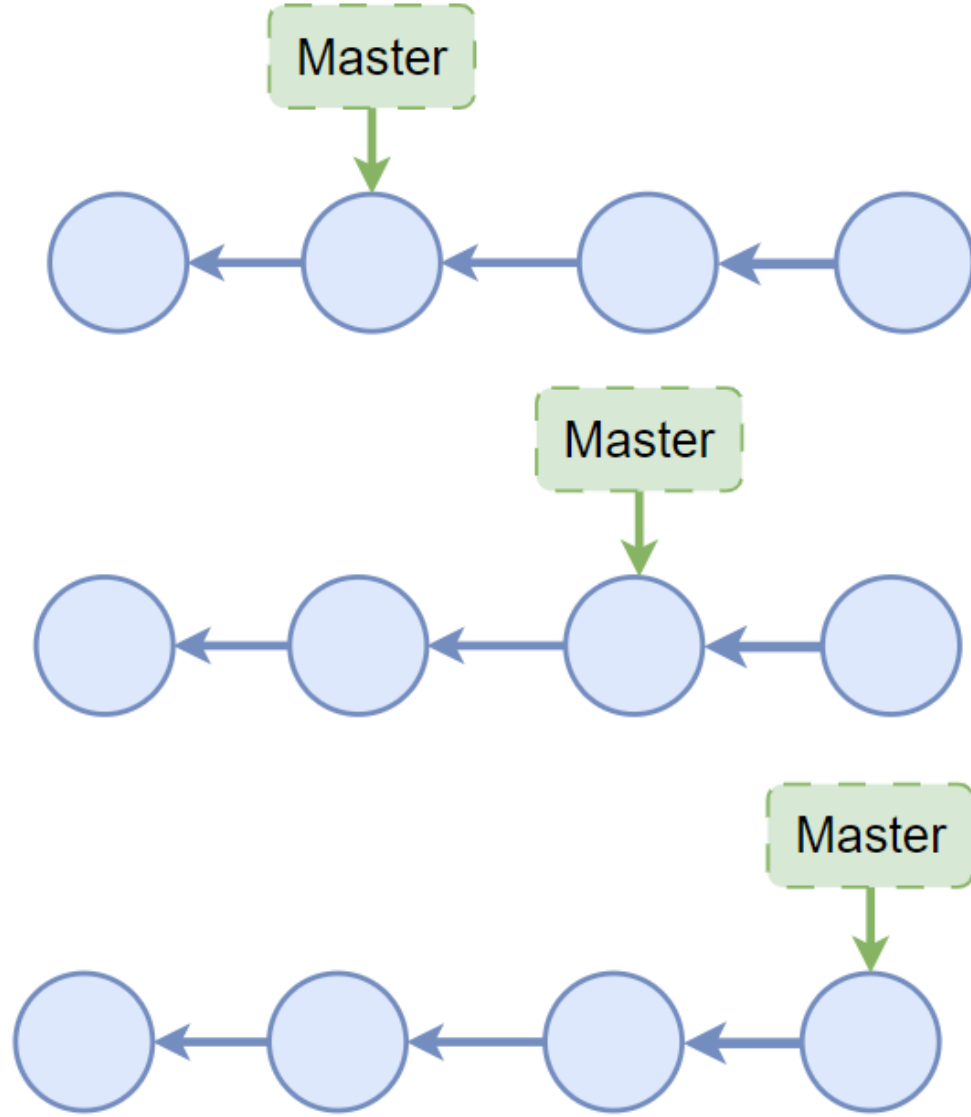
- What is branching?
 - Branching allows us to *diverge from the main line of work* and work on something else in isolation.
 - Conceptually, You can think of a branch like a *separate isolated workspace*



Branching in Git vs Branching in subversion

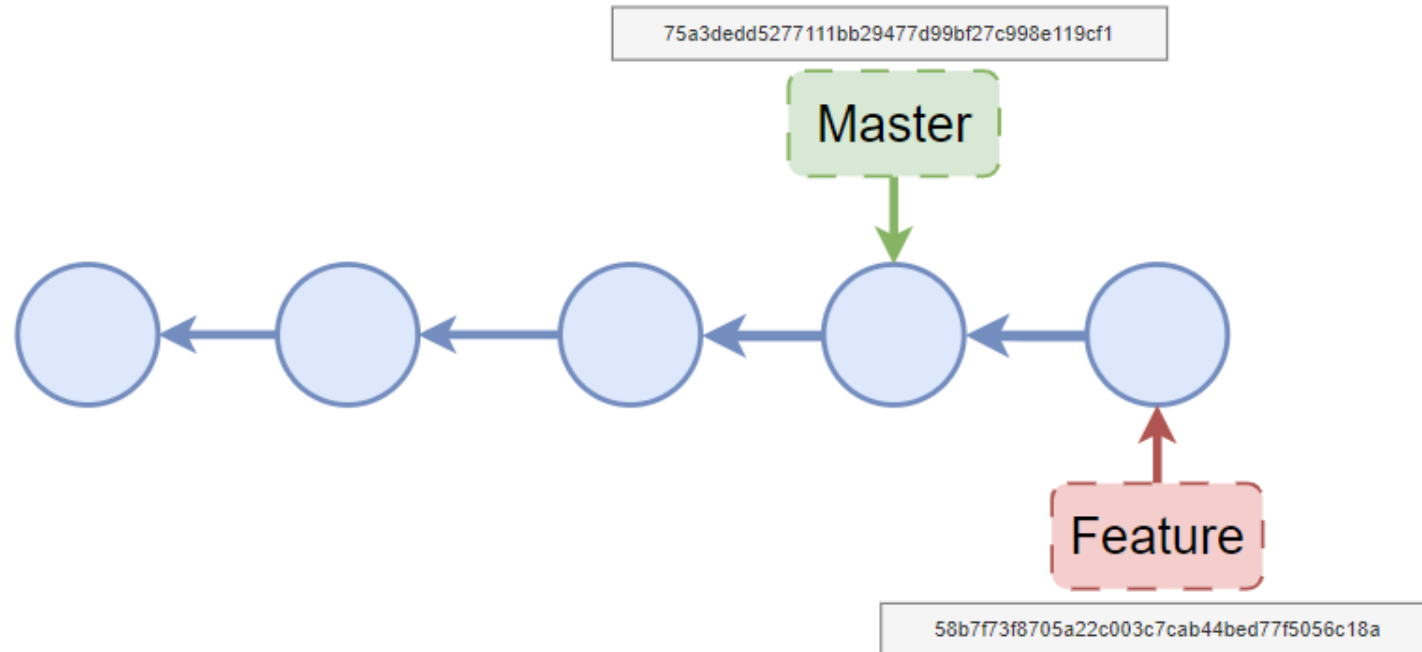
- Branching in subversion is performed by *copying all the files* in another place in the disk.
 - It consumes a lot of space in the disk
 - It is slow
- Branching in git is *just a pointer to a commit!*
 - The master branch is just a pointer to the last commit to the main line of work
 - As we make new commit git moves the pointer automatically
 - It is *fast* and efficient

Git stores compact version of actual files!



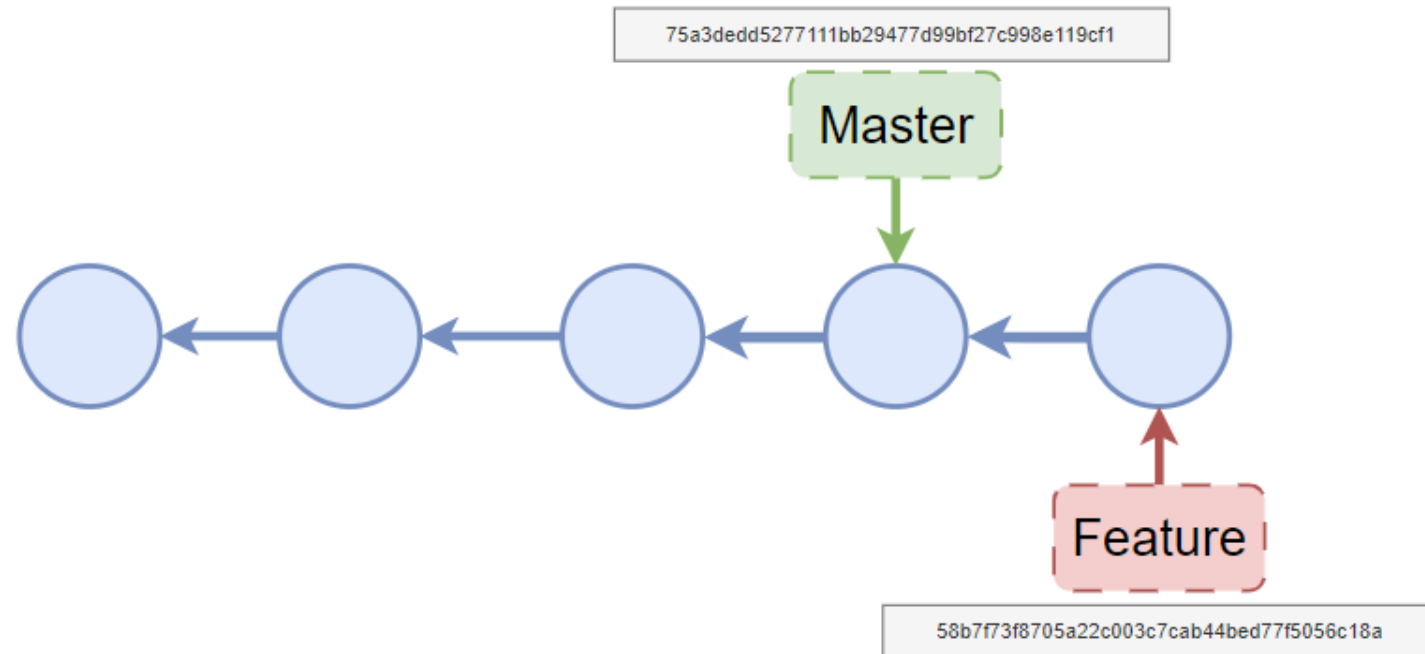
Create a new branch

- When we create a new branch git *creates a new pointer that can move around*
- A branch is just a file that contains 40 byte commit id
- Gits know the latest code in each branch



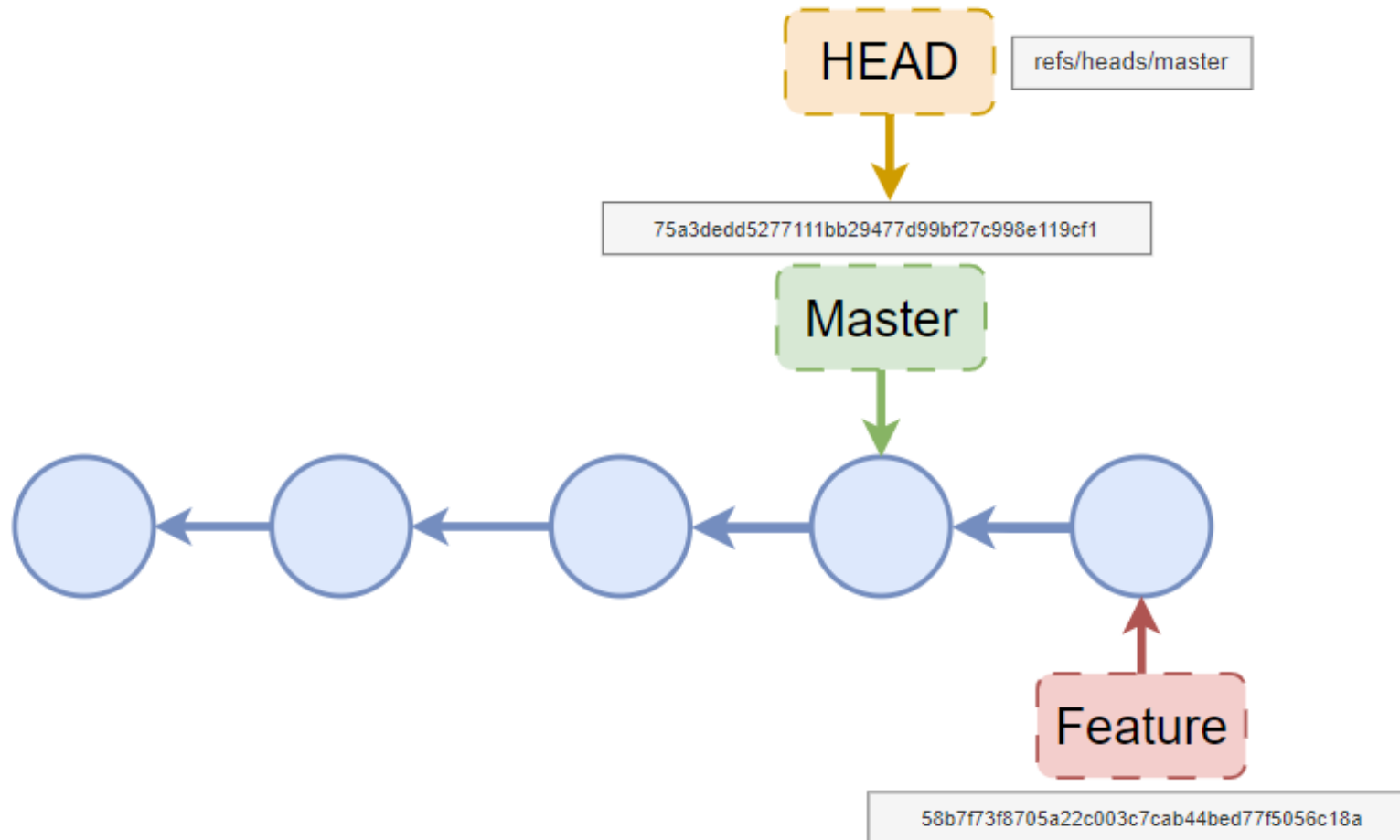
Switch to a branch

- When we switch to the *Master* branch git takes the snapshot from the commit that *Master* branch points to and reset the working directory to that commit
- How does git know which branch we are working on?



HEAD

- Another pointer comes into play!
- Using a special pointer called *HEAD* (another file that keeps the name of the branch)

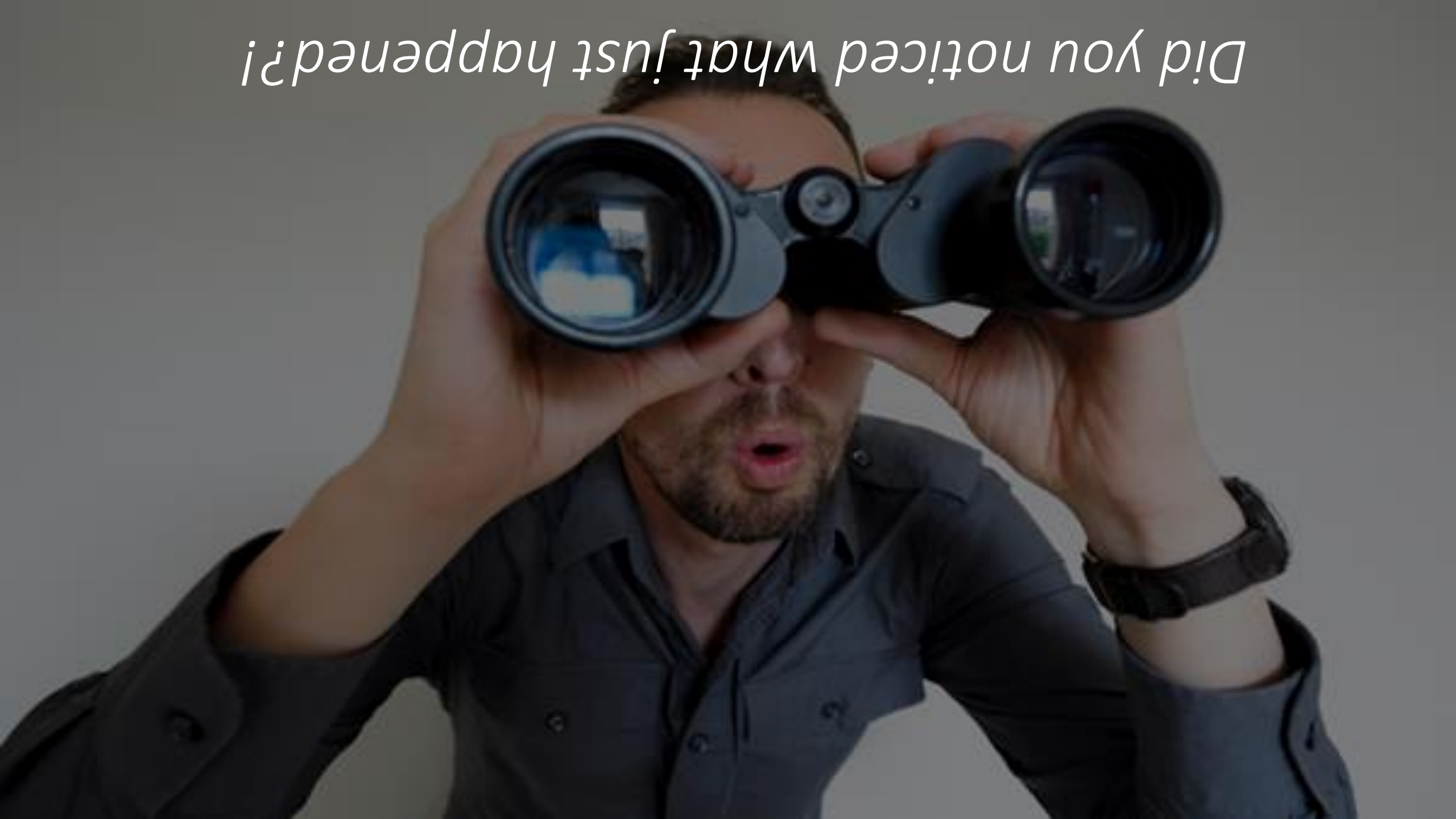


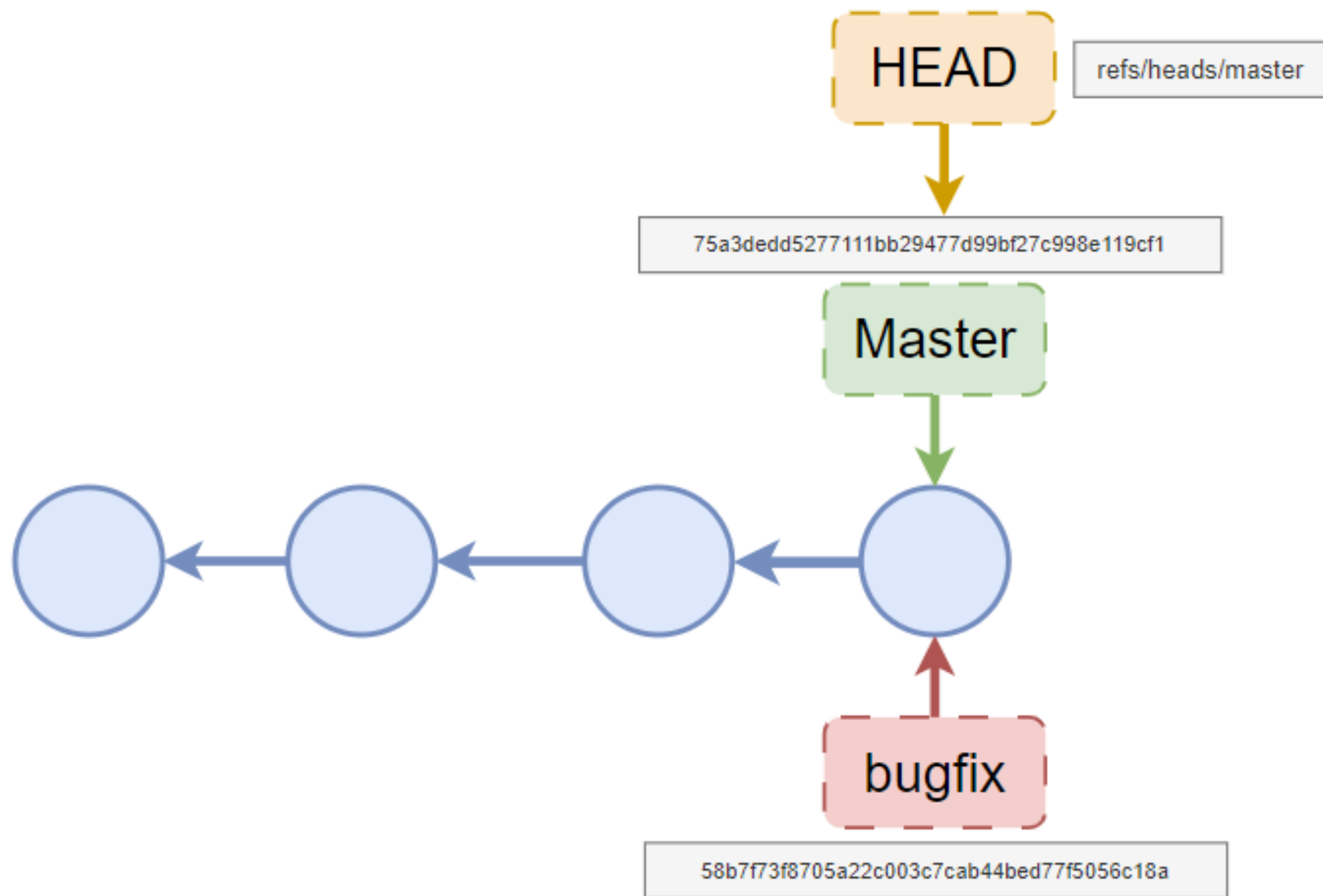
Create, show and switch to a branch

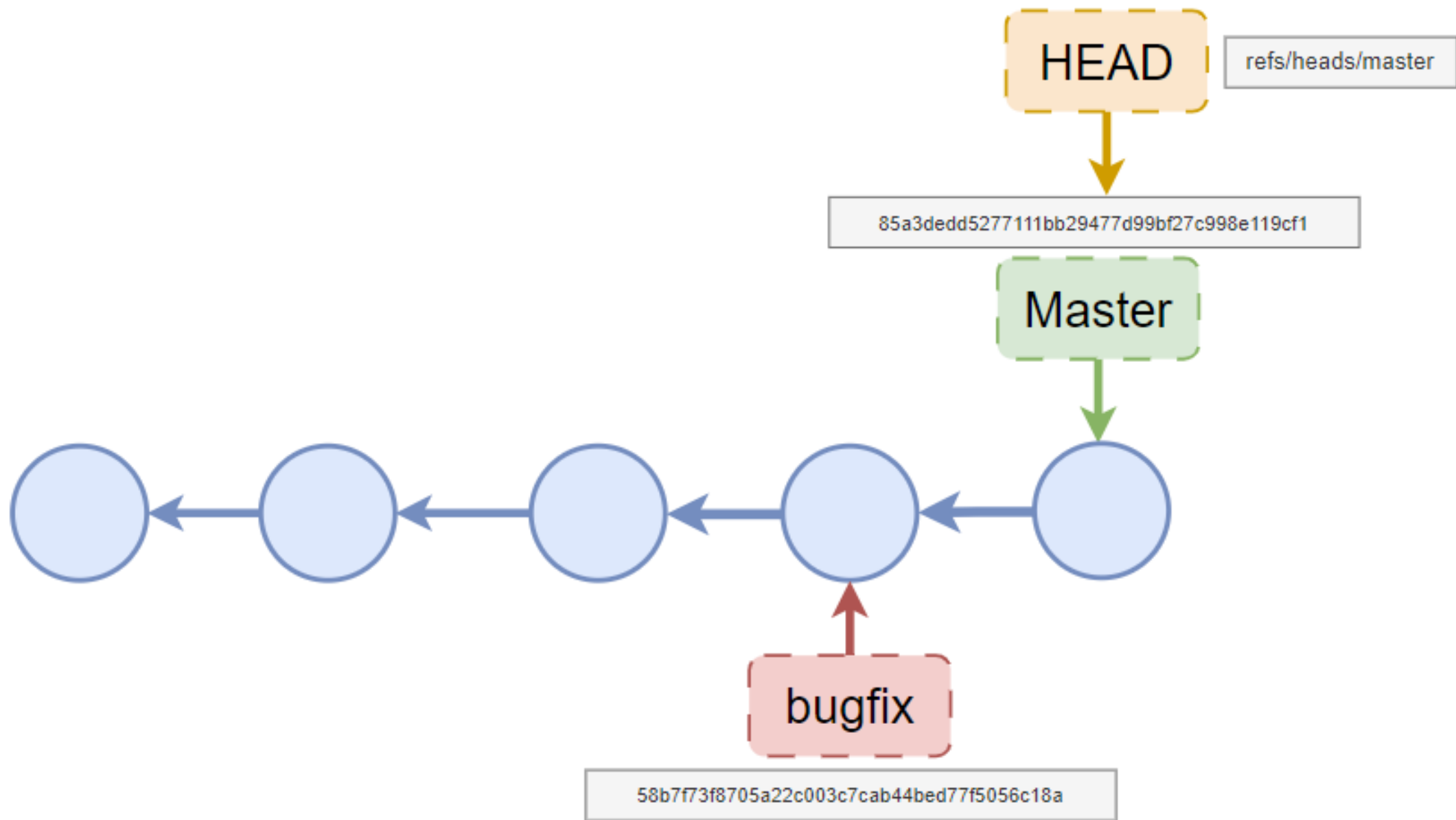
```
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git branch bugfix
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git branch
bugfix
* master
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    file3.txt

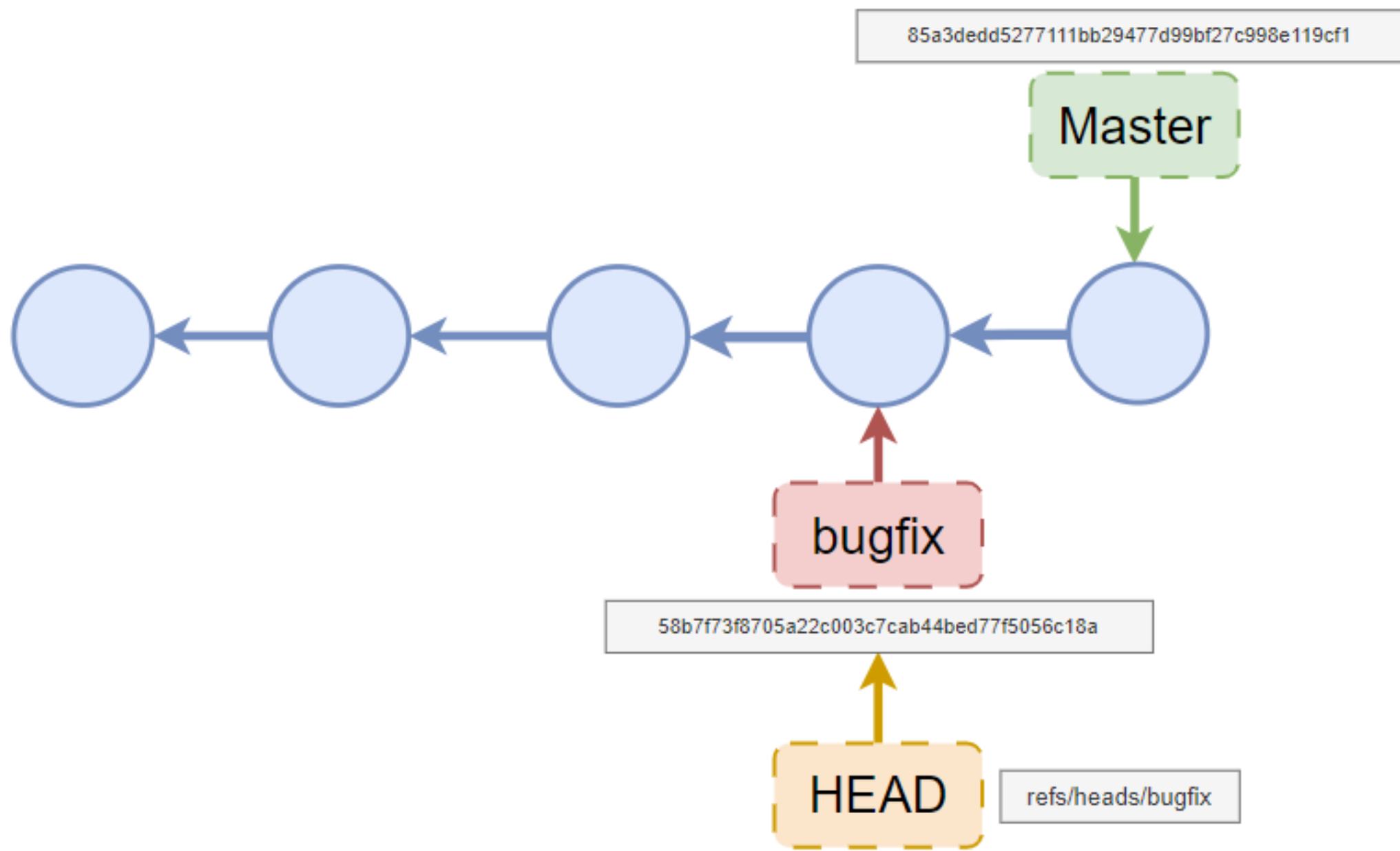
nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git add .\file3.txt
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git commit -m "file3 added!"
[master d4b1d25] file3 added!
1 file changed, 1 insertion(+)
create mode 100644 file3.txt
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git switch bugfix
Switched to branch 'bugfix'
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git branch -m bugfix bugfix/signup-form
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git branch
* bugfix/signup-form
master
```

Did you notice what just happened?!









Faster way of creating and switching to a branch

To create a branch and switch to it we have two options

1. create and then switch

git branch bugfix

git switch bugfix

2. create and switch at the same time

git switch -C bugfix

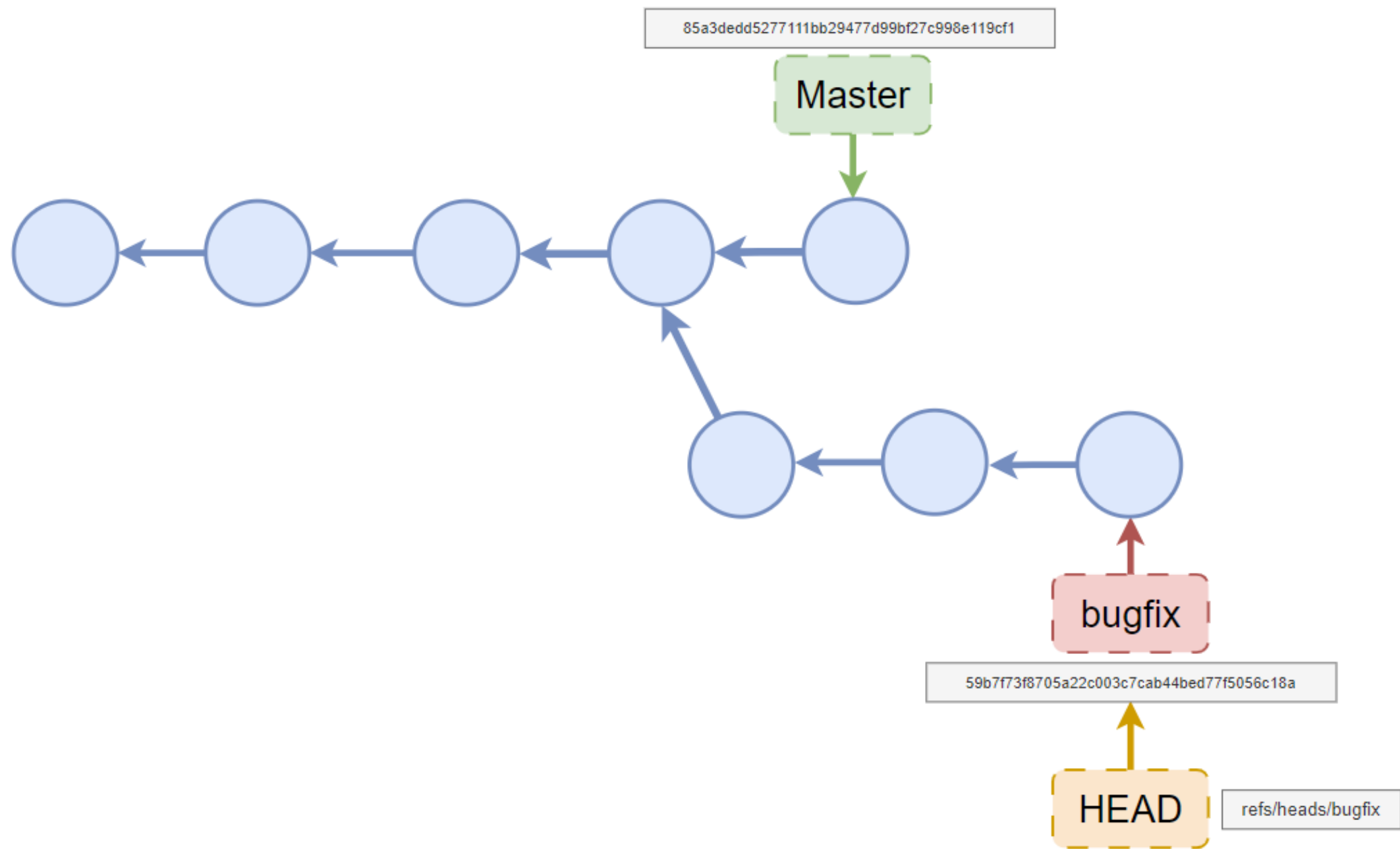
Let's create some commits in this branch and then check the history!

git log with two branches

```
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git log --oneline --all
8c30a4d (bugfix/signup-form) file5.txt added
c90ab6c file4.txt updated
c516911 file4.txt added
d4b1d25 (HEAD -> master) file3 added!
8cd307e gitignore file added!
e9e96b0 file3 deleted!
50e4039 file3.txt updated
55d7fd0 file3.txt added
58b7f73 file2.txt converted to utf-8 format!
bbbde31 new line added to file2.txt
75a3ded file2 updated!
```

As you can see the master branch is tree steps behind the bugfix/signup-form branch!

What will we see if we just switch to the master branch and check the history?



Comparing branches

- As we commit to our branches, we need to know how they are diverging from *Master*
- *Why is it important to check the divergence of branches from Master branch?*

You should check the last commit in the master branch and make sure it has not progressed while we have been working on our branch!

The changes in the new commit might have conflicts with your changes

Comparing branches – *shallow comparison*

- All the commits that are in *bugfix/signup-form* branch that **are not** in *Master* branch
- You can use *--oneline* option to make the result more concise!

```
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git log master..bugfix/signup-form
commit 8c30a4dc1ef103dd535748a7d393ede5bd1ee627 (bugfix/signup-form)
Author: Majid Babaei <16mb64@gmail.com>
Date:   Sun Sep 10 14:55:31 2023 -0400
```

file5.txt added

```
commit c90ab6c7f357746d3d0e2de7984652577dc8ec81
Author: Majid Babaei <16mb64@gmail.com>
Date:   Sun Sep 10 14:54:51 2023 -0400
```

file4.txt updated

```
commit c516911a9edf5adba40f7d0dc101098455cbeef0
Author: Majid Babaei <16mb64@gmail.com>
Date:   Sun Sep 10 14:54:11 2023 -0400
```

file4.txt added

```
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git log master..bugfix/signup-form --oneline
8c30a4d (bugfix/signup-form) file5.txt added
c90ab6c file4.txt updated
c516911 file4.txt added
```

Comparing branches – *in-depth comparison*

- To get the diff between the last commits of two branches

```
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git diff master..bugfix/signup-form
diff --git a/file3.txt b/file3.txt
deleted file mode 100644
index 7c8ac2f..0000000
--- a/file3.txt
+++ /dev/null
@@ -1,0 @@
-file3
diff --git a/file4.txt b/file4.txt
new file mode 100644
index 0000000..e3ab3c6
--- /dev/null
+++ b/file4.txt
@@ -0,0 +1,2 @@
+file4
+this is file4 that is used to record some important info
diff --git a/file5.txt b/file5.txt
new file mode 100644
index 0000000..d92b9c3
--- /dev/null
+++ b/file5.txt
@@ -0,0 +1 @@
+file5
\ No newline at end of file
```

Comparing branches – *in-depth comparison*

- If we only want to see the file that are going to be affected, we can just use:
 - *--name-only*
 - *--name-status*

```
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git diff master..bugfix/signup-form --name-only
file3.txt
file4.txt
file5.txt
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git diff master..bugfix/signup-form --name-status
D      file3.txt
A      file4.txt
A      file5.txt
```

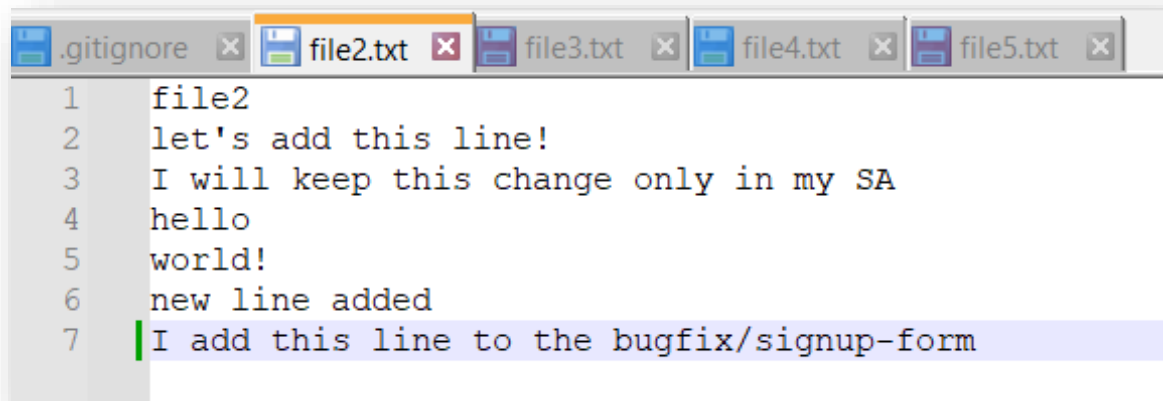
File2.txt is in both branch!

```
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git ls-tree master
100644 blob 8049d96786391bdf6fcf6a3023a334c88bd2f329      .gitignore
100644 blob b4bda5702fe47b9c04bbe940679e678d9e033df7      file2.txt
100644 blob 7c8ac2f8d82a1eb5f6aaece6629ff11015f91eb4      file3.txt
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git ls-tree bugfix/signup-form
100644 blob 8049d96786391bdf6fcf6a3023a334c88bd2f329      .gitignore
100644 blob b4bda5702fe47b9c04bbe940679e678d9e033df7      file2.txt
100644 blob e3ab3c6231e098a9ad06c900febc62d25f2a570e      file4.txt
100644 blob d92b9c31802fd4d80600b004f3b34d95f5d9e080      file5.txt
```

Let's back to the Master branch and make some changes!

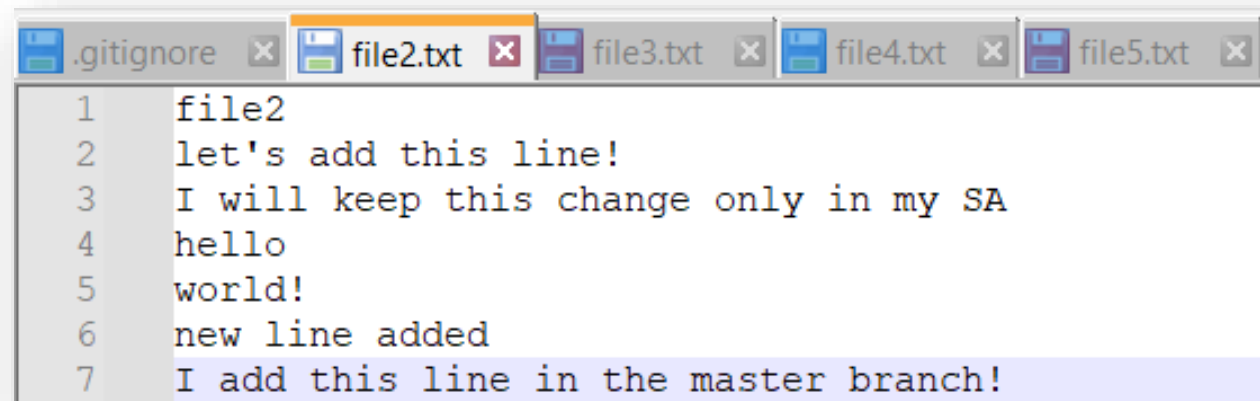
```
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git add .\file2.txt
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git commit -m "new line added to file2.txt"
[master 5286388] new line added to file2.txt
1 file changed, 1 insertion(+)

PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git checkout bugfix/signup-form
Switched to branch 'bugfix/signup-form'
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git add .\file2.txt
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git commit -m "file2.txt updated!"
[bugfix/signup-form 922514d] file2.txt updated!
1 file changed, 1 insertion(+)
```



The screenshot shows a code editor with a tab bar at the top containing .gitignore, file2.txt (selected), file3.txt, file4.txt, and file5.txt. The file2.txt tab is active, displaying the following content:

```
1 file2
2 let's add this line!
3 I will keep this change only in my SA
4 hello
5 world!
6 new line added
7 I add this line to the bugfix/signup-form
```



The screenshot shows a code editor with a tab bar at the top containing .gitignore, file2.txt (selected), file3.txt, file4.txt, and file5.txt. The file2.txt tab is active, displaying the following content:

```
1 file2
2 let's add this line!
3 I will keep this change only in my SA
4 hello
5 world!
6 new line added
7 I add this line in the master branch!
```

```
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git diff master..bugfix/signup-form file2.txt
diff --git a/file2.txt b/file2.txt
index e73fa76..271253f 100644
--- a/file2.txt
+++ b/file2.txt
@@ -4,4 +4,4 @@ I will keep this change only in my SA
 hello
 world!
 new line added
-I add this line in the master branch!
+I add this line to the bugfix/signup-form
\ No newline at end of file
```

```
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git diff master~1..bugfix/signup-form file2.txt
diff --git a/file2.txt b/file2.txt
index b4bda57..271253f 100644
--- a/file2.txt
+++ b/file2.txt
@@ -4,3 +4,4 @@ I will keep this change only in my SA
 hello
 world!
 new line added
+I add this line to the bugfix/signup-form
\ No newline at end of file
```

```
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git diff master~1..bugfix/signup-form~1 file2.txt
```

Stashing

- When we switch branches, git reset our working directory to the snapshot stored in the last commit of the target branch.
- If you have local changes in the working directory that we haven't committed yet, they could get lost!
- In these situations, *git doesn't allow us to switch branches!*
- We should store the local changes somewhere in git repository, but it is not going to be a part of the history.

Stashing

- By default, running `git stash` will stash:
 - changes that have been added to your index (staged changes)
 - changes made to files that are currently tracked by Git (unstaged changes)
- But it will not stash:
 - new files in your working copy that have not yet been staged
 - files that have been ignored

```
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> echo "only for the testing purpose!" >> file5.txt
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git stash push -m "this is a draf version!"
Saved working directory and index state On bugfix/signup-form: this is a draf version!
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git stash list
stash@{0}: On bugfix/signup-form: this is a draf version!
```

Stashing

- By default, `git stash pop` will re-apply the most recently created stash: `stash@{0}`
- You can choose which stash to re-apply by passing its identifier as the last argument.

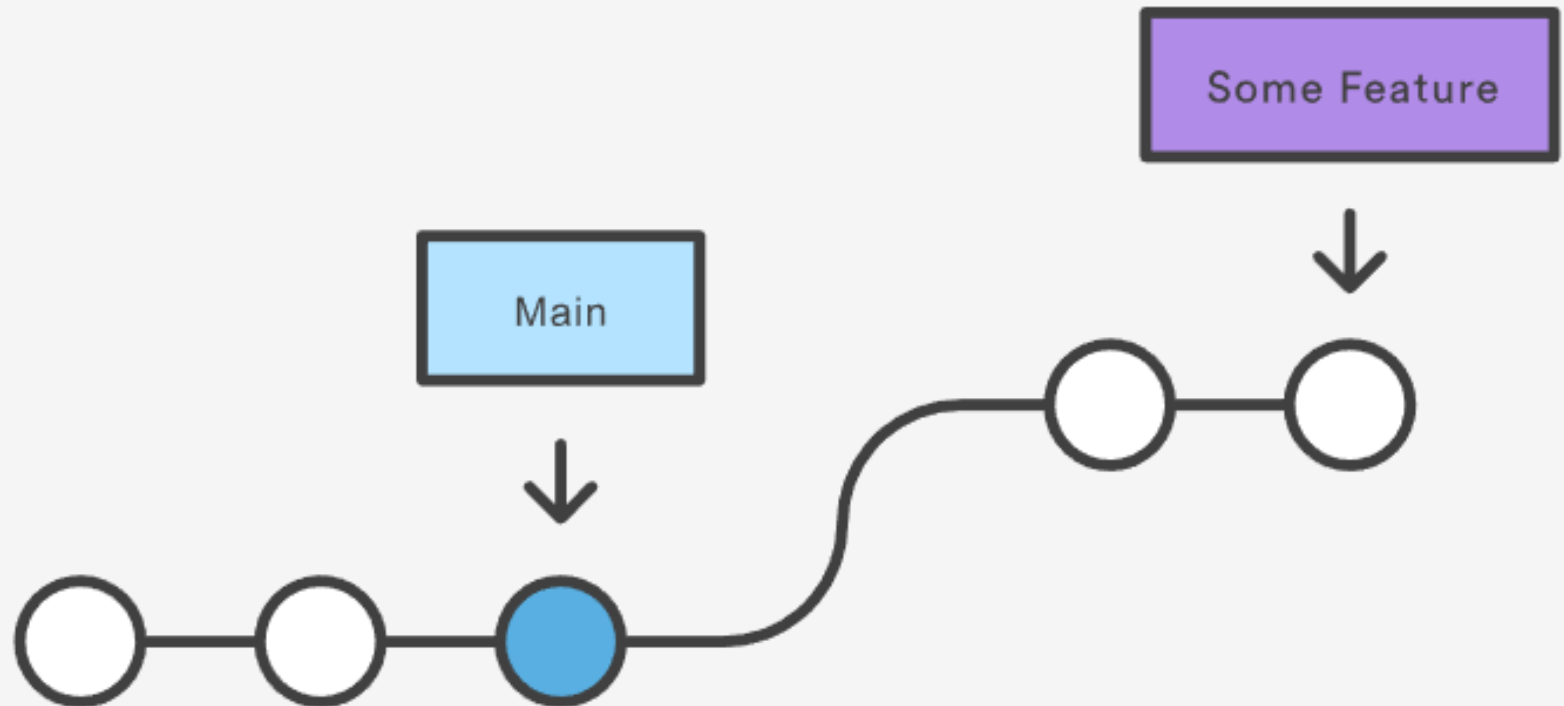
```
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git stash pop 'stash@{0}'
On branch bugfix/signup-form
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file5.txt

no changes added to commit (use "git add" and/or "git commit -a")
Dropped stash@{0} (3afc3bd266ca8c2e4e61e06292ae824cd6eab2d1)
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice>
```

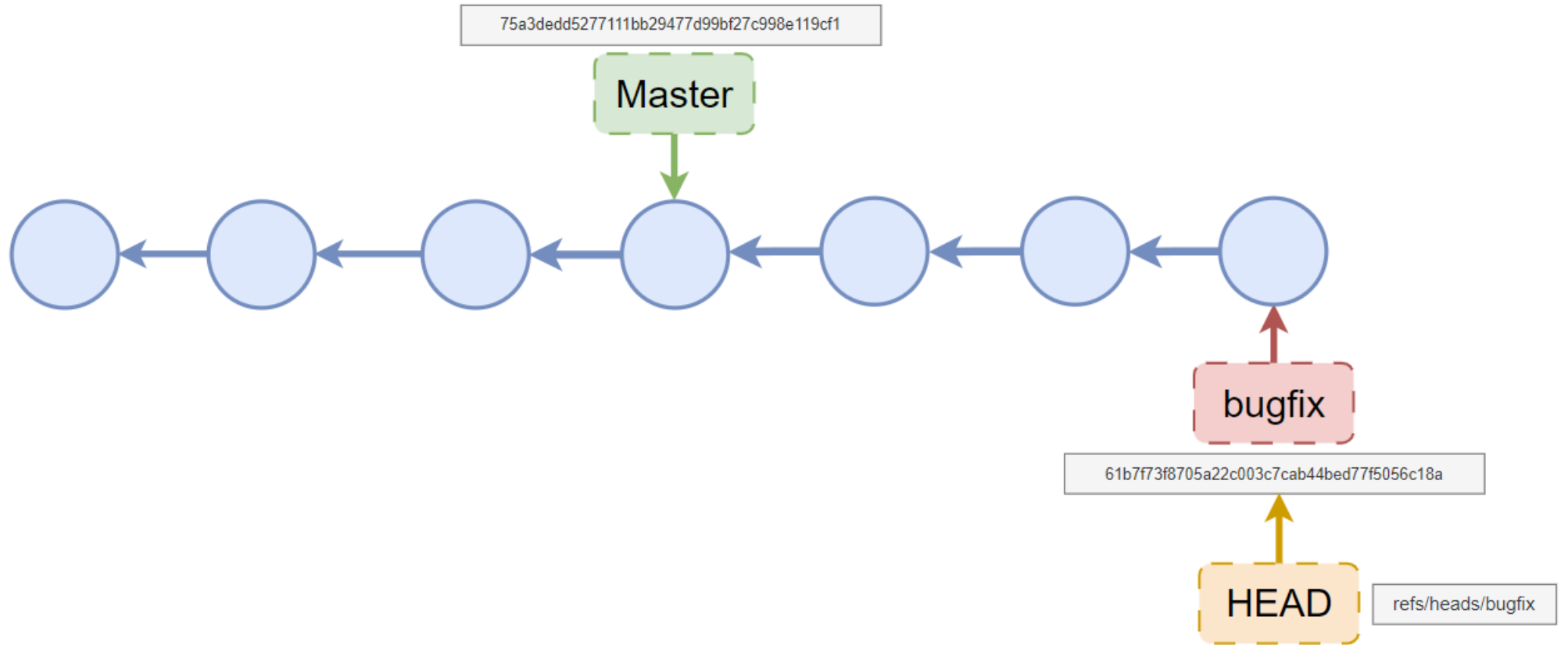
- If you decide you no longer need a particular stash, you can delete it with:
 - *git stash drop*

Merging

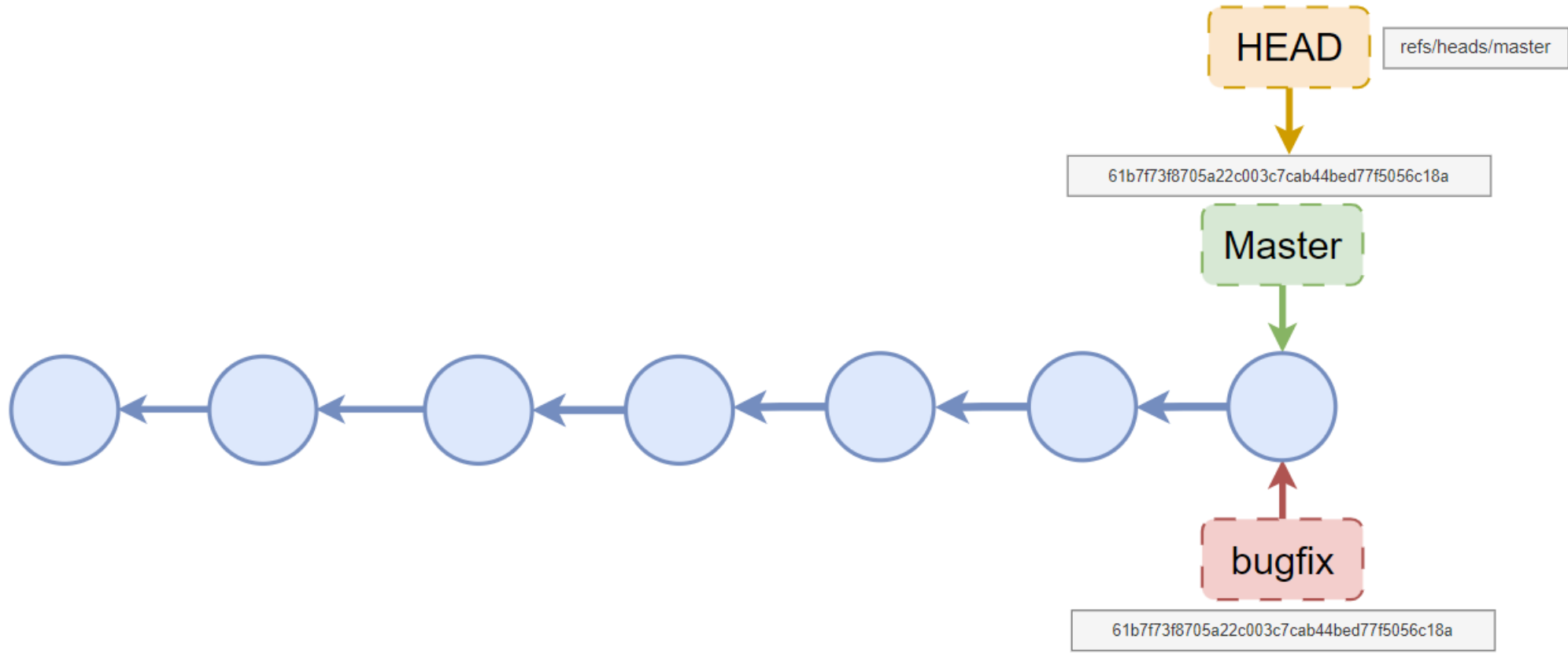
- It is all about bringing changes from one branch to another branch
- In git we have two types of merges:
 - Fast-Forward
 - 3-way



Fast-Forward Merger



Fast-Forward Merger

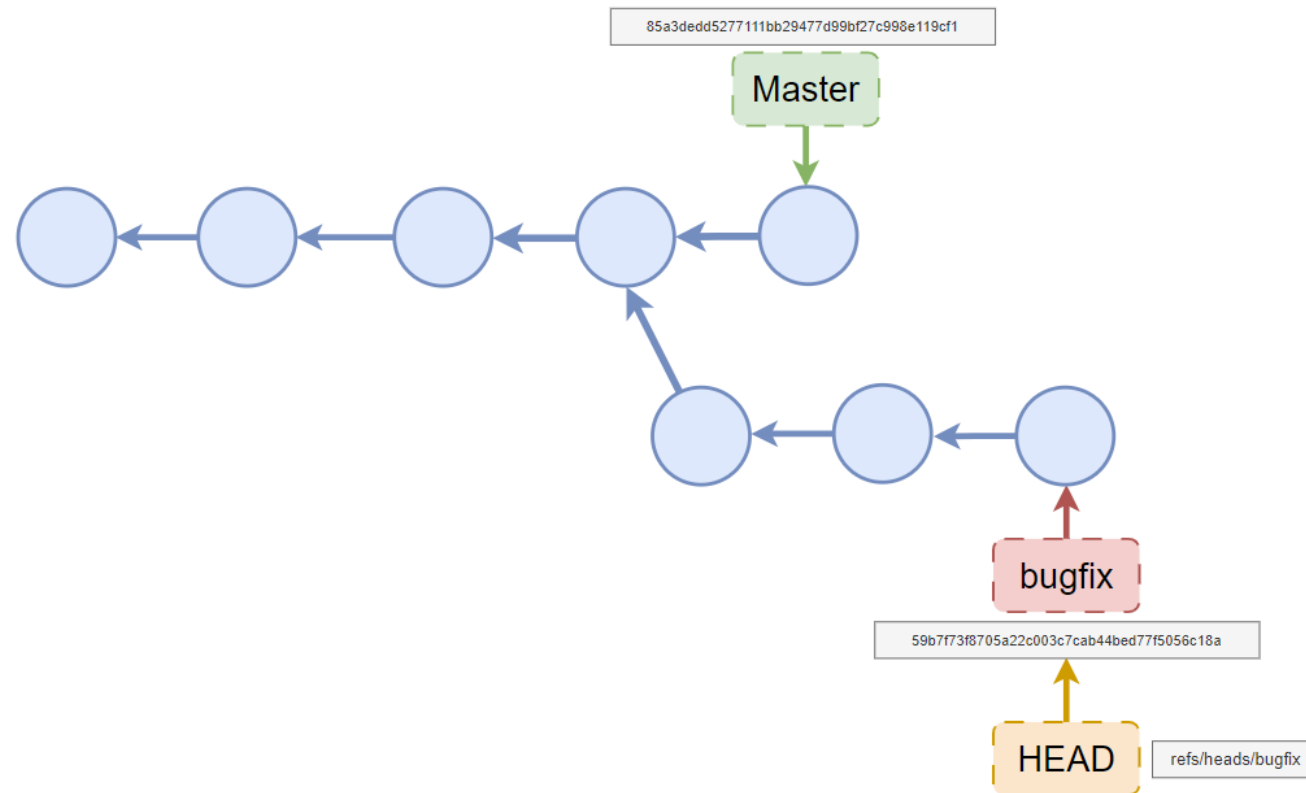


Fast-Forward Merger

- At the beginning they are pointing to the same commit
- Then we switch to the *bugfix* branch and make some commit
- Now we want to bring changes back to master
- Because these branches have not diverged
- There is direct (*linear path*) from master to bugfix branch
- All git needs to do is to *bring the pointer of master branch forward!*

Fast-Forward is possible only when ...

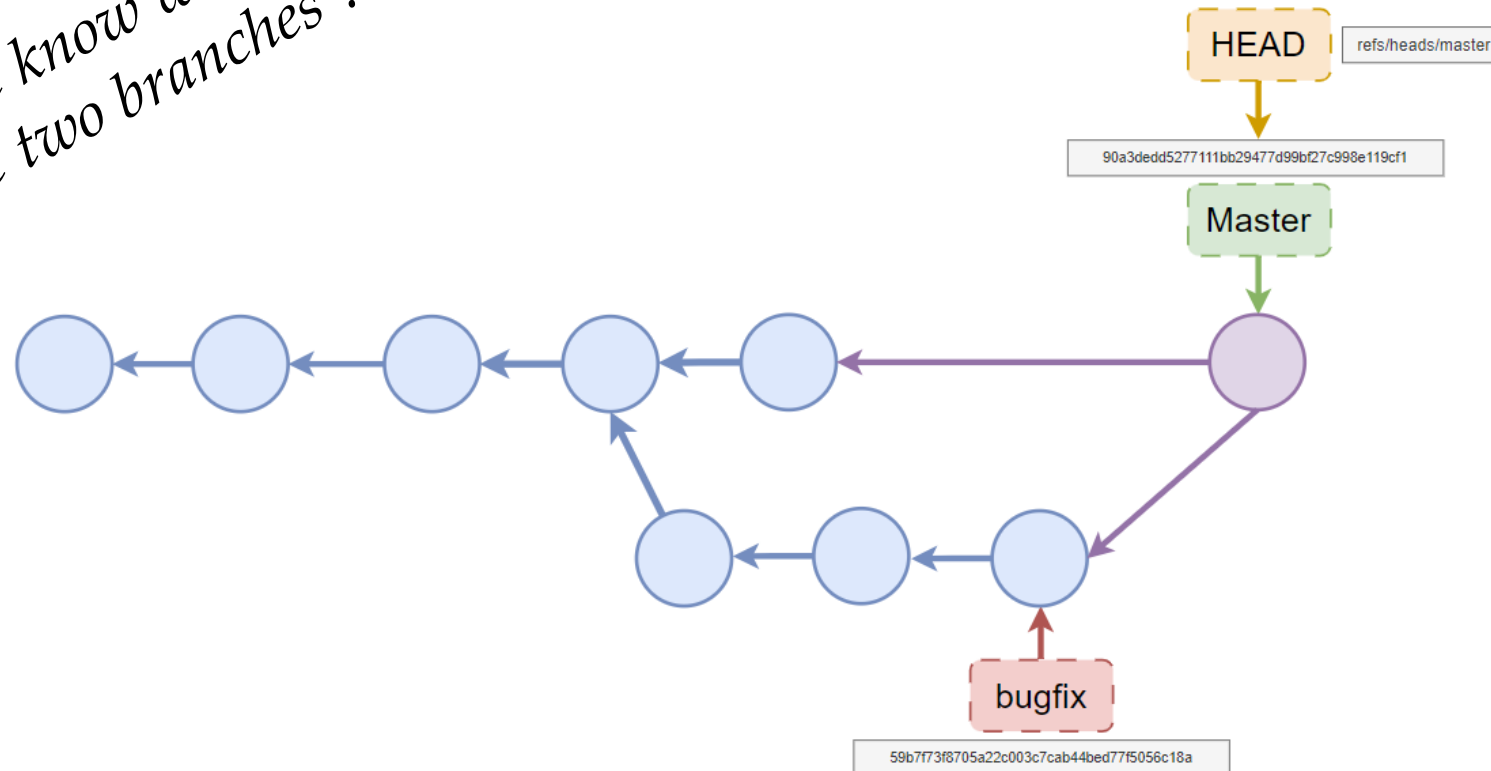
- There is a linear path!
- If you added additional commits to the master branch before merging branches, fast-forward merge is not possible



3-way Merger

- In this case git creates a new commit based on:
 - The *tips of the two branches* and the *common ancestor* of the branches

How does the git know what are the tips of these two branches?



Example

- First, we let's look at the history
- To get the better view of the branches, you can use `--graph`

```
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git log --oneline --all --graph
* 95b6a42 (HEAD -> bugfix/signup-form) the content of the file5.txt finilized!
* 922514d file2.txt updated!
* 8c30a4d file5.txt added
* c90ab6c file4.txt updated
* c516911 file4.txt added
| * 5286388 (master) new line added to file2.txt
| * d4b1d25 file3 added!
|/
* 8cd307e gitignore file added!
* e9e96b0 file3 deleted!
* 50e4039 file3.txt updated
* 55d7fd0 file3.txt added
* 58b7f73 file2.txt converted to utf-8 format!
* bbbde31 new line added to file2.txt
* 75a3ded file2 updated!
```

Master has diverged from bugfix branch!

Preparing to merge

Before performing a merge there are a couple of preparation steps to take to ensure the merge goes smoothly.

1. Confirm the receiving branch

Execute git status to ensure that HEAD is pointing to the correct merge-receiving branch. If needed, execute git checkout to switch to the receiving branch

2. Fetch latest remote commits

Make sure the receiving branch and the merging branch are up-to-date with the latest remote changes. Execute git fetch to pull the latest remote commits. Once the fetch is completed ensure the main branch has the latest updates by executing git pull.

3. Merging

Once the previously discussed "preparing to merge" steps have been taken a merge can be initiated by executing git merge <target_branch> where <target_branch> is the name of the branch that will be merged into the receiving branch

```

PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git status
On branch bugfix/signup-form
nothing to commit, working tree clean
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git switch master
Switched to branch 'master'
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git fetch
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git pull
There is no tracking information for the current branch.
Please specify which branch you want to merge with.
See git-pull(1) for details.

```

```
git pull <remote> <branch>
```

If you wish to set tracking information for this branch you can do so with:

```
git branch --set-upstream-to=<remote>/<branch> master
```

```
PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> ls
```

Directory: C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice

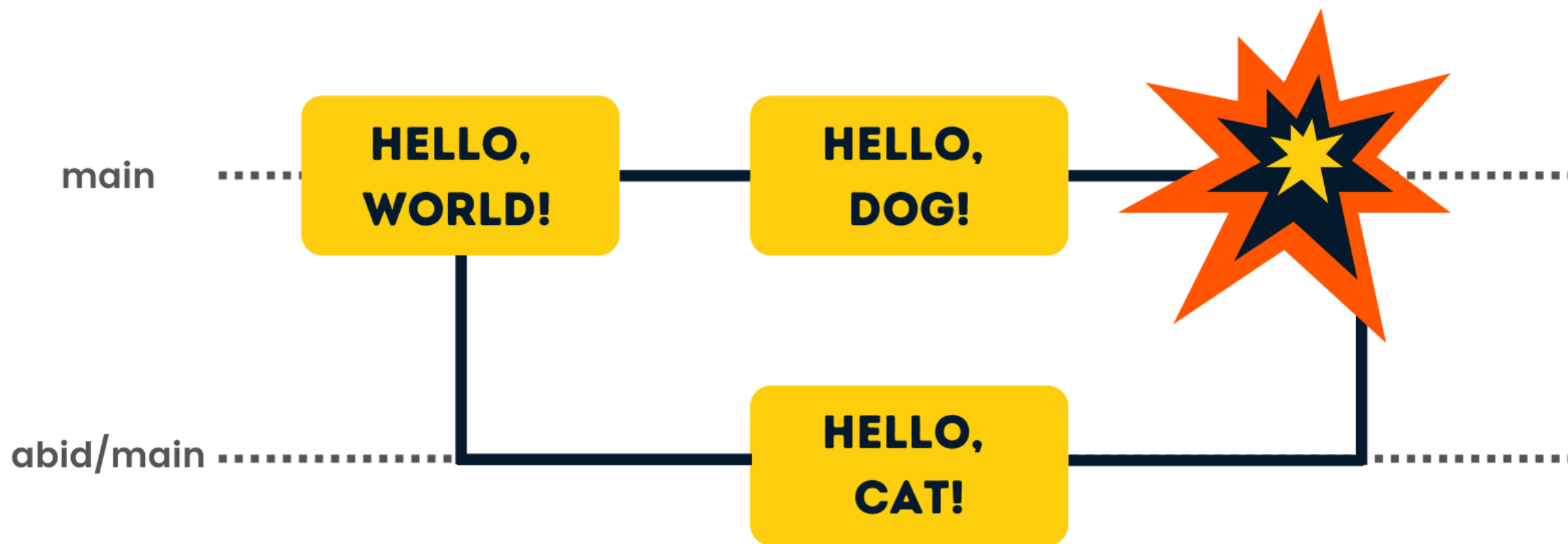
| Mode | LastWriteTime | | Length | Name |
|---------|---------------|----------|--------|------------|
| ---- | ----- | | ----- | ---- |
| d----- | 9/10/2023 | 12:01 AM | | build |
| d----- | 9/10/2023 | 12:01 AM | | logs |
| -a----- | 9/10/2023 | 12:03 AM | 28 | .gitignore |
| -a----- | 9/9/2023 | 11:51 PM | 28 | file1.txt |
| -a----- | 9/13/2023 | 1:54 PM | 138 | file2.txt |
| -a----- | 9/13/2023 | 1:54 PM | 7 | file3.txt |
| -a----- | 9/10/2023 | 12:02 AM | 14 | logs.txt |



```

PS C:\Users\drbab\OneDrive\Desktop\tmp\workspace\git-practice> git merge bugfix/signup-form
Auto-merging file2.txt
CONFLICT (content): Merge conflict in file2.txt
Automatic merge failed; fix conflicts and then commit the result.

```



Any Questions?

