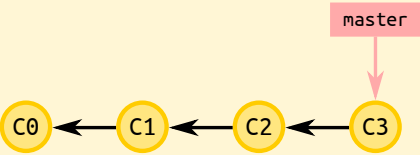


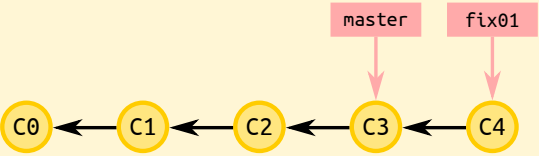
Starting scenario



```
graph RL; C0((C0)) --> C1((C1)); C1 --> C2((C2)); C2 --> C3((C3)); master[master] --> C3;
```

- We have a repository with several commits C0..C3
- Each black arrow points to the ancestor of a commit
- The pink arrow shows the pointer of the HEAD of the master branch

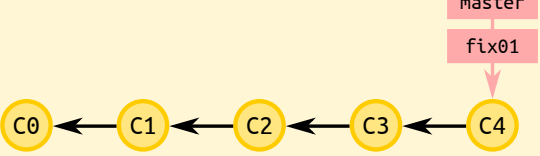
Scenario #1



```
graph RL; C0((C0)) --> C1((C1)); C1 --> C2((C2)); C2 --> C3((C3)); C3 --> C4((C4)); master[master] --> C3; fix01[fix01] --> C3;
```

- Starting from the previous scenario, we create a new branch "fix01"
`$ git checkout -b fix01`
- We make our changes and then commit creating C4
`$ git commit -m "Made some important changes"`
- The master branch goes on pointing to C3

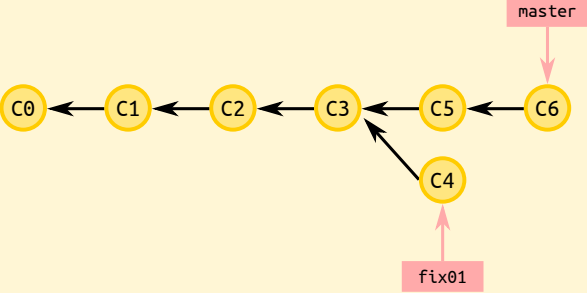
Merge for scenario #1



```
graph RL; C0((C0)) --> C1((C1)); C1 --> C2((C2)); C2 --> C3((C3)); C3 --> C4((C4)); master[master] --> C4; fix01[fix01] --> C4;
```

- In this very simple case, we can move to master branch:
`$ git checkout master`
- and type the following command to merge:
`$ git merge fix01`
At the end of this step, the master branch will point to the C4 commit
- If we want, we can remove the fix01 branch, since we no longer need it, with:
`$ git branch -d fix01`

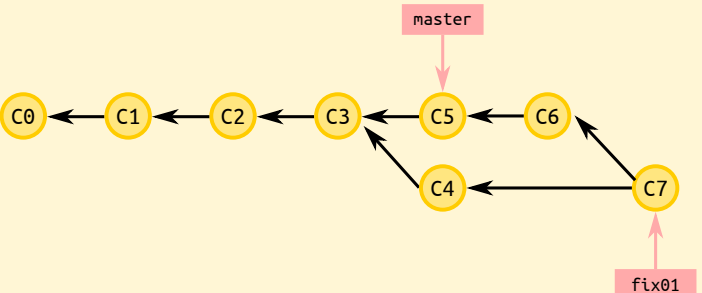
Scenario #2



```
graph RL; C0((C0)) --> C1((C1)); C1 --> C2((C2)); C2 --> C3((C3)); C3 --> C5((C5)); C5 --> C6((C6)); C3 --> C4((C4)); master[master] --> C6; fix01[fix01] --> C4;
```

- Starting from the previous scenario, we create a new branch "fix01"
`$ git checkout -b fix01`
- We make our changes and then commit creating C4
`$ git commit -m "Made some important changes"`
- On the master branch we make other changes that we commit as C5
- The branches have diverged! We now have three options...

Merge for scenario #2 -- Option 1

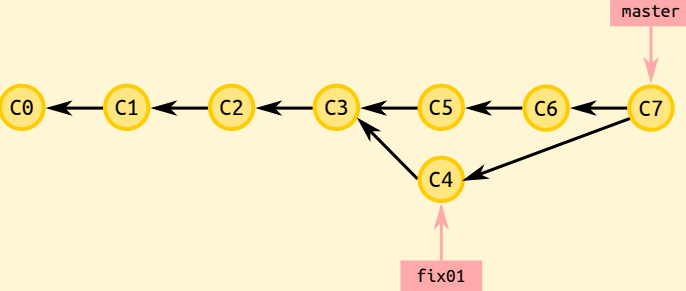


```
graph RL; C0((C0)) --> C1((C1)); C1 --> C2((C2)); C2 --> C3((C3)); C3 --> C5((C5)); C5 --> C6((C6)); C3 --> C4((C4)); C5 --> C7((C7)); master[master] --> C6; fix01[fix01] --> C7;
```

Choose this if you need to have changes from "master" branch into "fix01" before going on with your work:

- Switch to "fix01" branch:
`$ git checkout fix01`
- Merge master changes:
`$ git merge master`

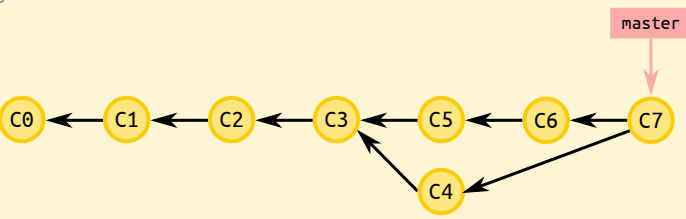
Merge for scenario #2 -- Option 3



```
graph RL; C0((C0)) --> C1((C1)); C1 --> C2((C2)); C2 --> C3((C3)); C3 --> C5((C5)); C5 --> C6((C6)); C3 --> C4((C4)); C6 --> C7((C7)); master[master] --> C7;
```

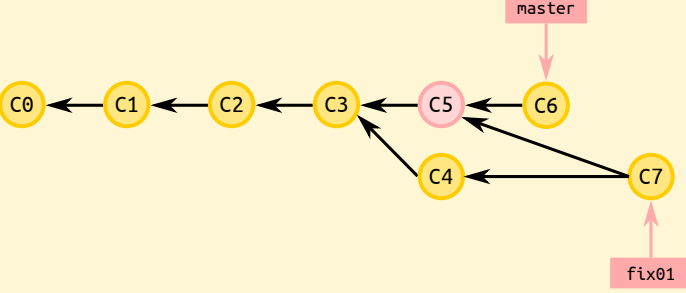
Choose this if you need to have completed your work on "fix01" and ready to put changes into "master" branch:

- Move to the "master" branch:
`$ git checkout master`
- Merge the "fix01" branch"
`$ git merge fix01`
- Then, it is possible to remove "fix01" branch:
`$ git branch -d fix01`



```
graph RL; C0((C0)) --> C1((C1)); C1 --> C2((C2)); C2 --> C3((C3)); C3 --> C5((C5)); C5 --> C6((C6)); C3 --> C4((C4)); C6 --> C7((C7)); master[master] --> C7;
```

Merge for scenario #2 -- Option 3 (Cherry picking)



```
graph RL; C0((C0)) --> C1((C1)); C1 --> C2((C2)); C2 --> C3((C3)); C3 --> C5((C5)); C5 --> C6((C6)); C3 --> C4((C4)); C5 --> C7((C7)); master[master] --> C6; fix01[fix01] --> C7;
```

Choose this if you need to take only the changes of one specific commit. In this case, suppose you want to merge commit C5 into the branch "fix01"

- Move to the "fix01" branch:
`$ git checkout fix01`
- Merge the commit C5 (suppose it has id 2bfbb02)
`$ git cherry-pick 2bfbb02`

Notes

- To visualise the graph structure of your repository, simply type:
`$ git log --oneline --graph`
- As an alternative, you can use a git UI (for example gitk). If you need to run gitk on a repository hosted on Zeus, you can mount the filesystem using sshfs, then exploit the instance of gitk that you installed on your os.