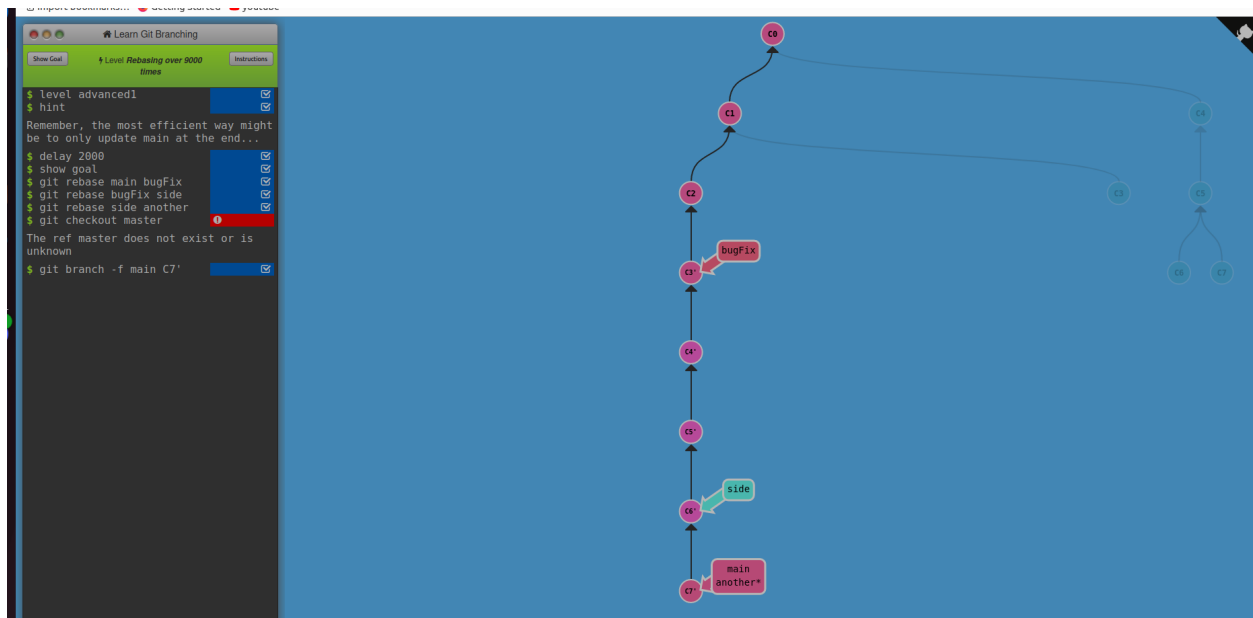
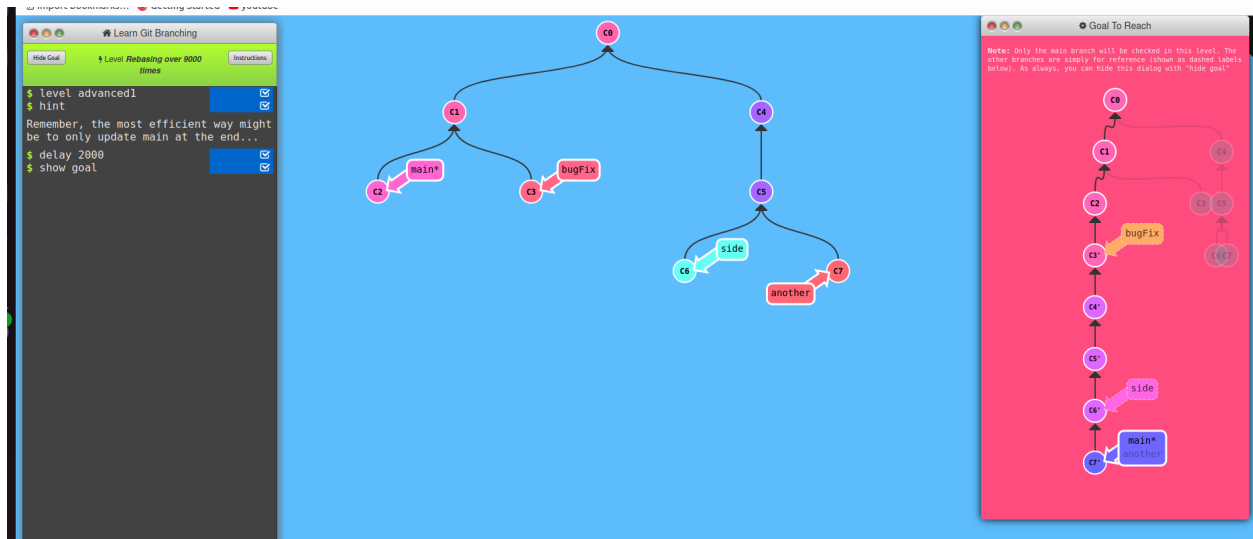


# Git:Level 5

In this task we have to rebase all the branch to main branch and achieve the following goal.



```
git rebase main bugFix
git rebase bugFix side
```

```
git rebase side another
git branch -f main C7' ot git rebase another master
```

## Task 2

### Specifying Parents

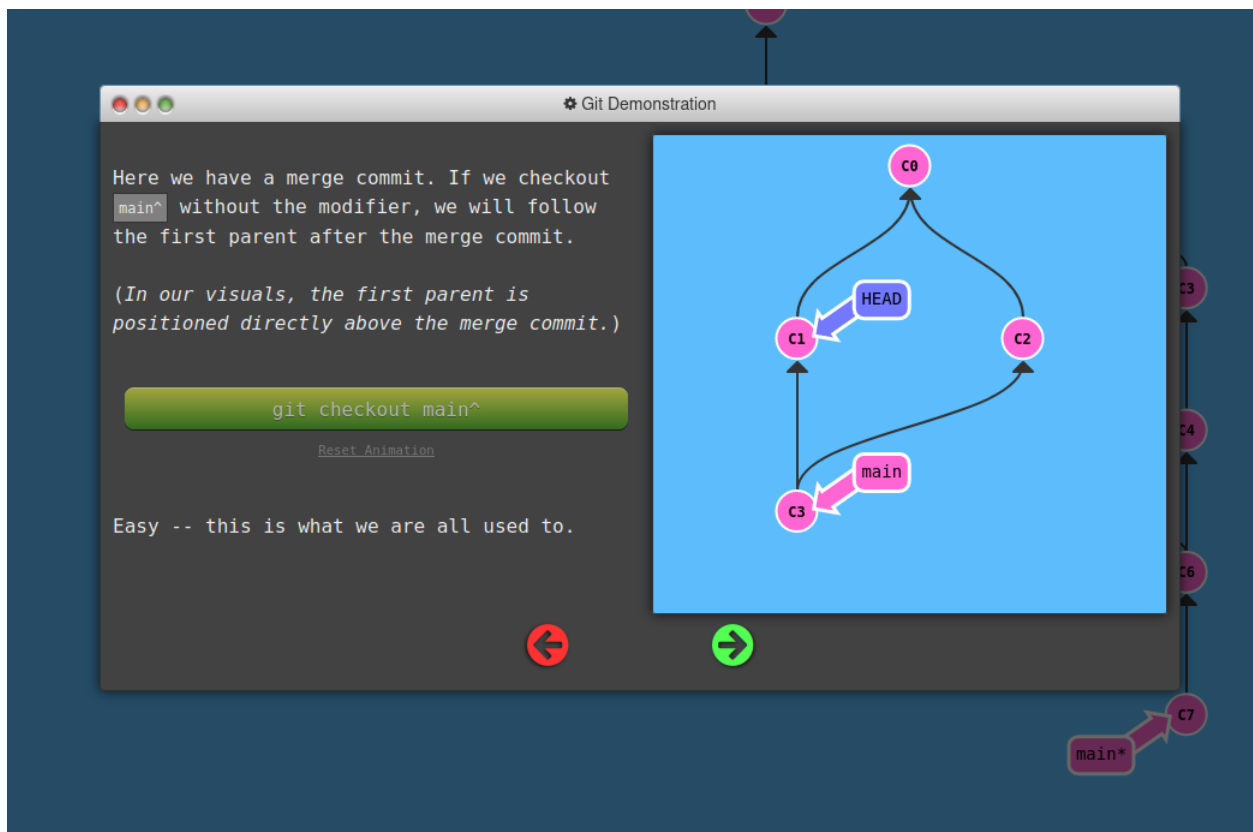
Like the `-` modifier, the `^` modifier also accepts an optional number after it.

Rather than specifying the number of generations to go back (what `-` takes), the modifier on `^`

specifies which parent reference to follow from a merge commit.

Remember that merge commits have multiple parents, so the path to choose is ambiguous.

Git will normally follow the "first" parent upwards from a merge commit, but specifying a number with `^` changes this default behavior.



Now let's try specifying the second parent instead...

```
git checkout main^2
```

[Reset Animation](#)

See? We followed the other parent upwards.

The diagram illustrates a Git commit history. At the top is commit C0, which is a merge of C1 and C2. C1 and C2 both point to C0. A blue arrow labeled 'HEAD' points to C2. A pink arrow labeled 'main' points to C3, which is a child of C1. To the right, a vertical chain of commits C3, C4, C6, and C7 is shown, with a pink arrow labeled 'main\*' pointing to C7. Below the diagram are two navigation arrows: a red one pointing left and a green one pointing right.

Even crazier, these modifiers can be checked out together! Check this out:

```
git checkout HEAD~^2-2
```

[Reset Animation](#)

The same movement as before, but all in one command.

The diagram illustrates a Git commit history. At the top is commit C0, which is a merge of C1 and C2. C1 and C2 both point to C0. A blue arrow labeled 'HEAD' points to C3, which is a child of C2. A pink arrow labeled 'main' points to C7, which is a child of C6. To the right, a vertical chain of commits C3, C4, C5, C6, and C7 is shown, with a pink arrow labeled 'main\*' pointing to C7. Below the diagram are two navigation arrows: a red one pointing left and a green one pointing right.

Git Demonstration

The `^` and `~` modifiers can make moving around a commit tree very powerful:

```
git checkout HEAD~; git checkout HEAD^2;
git checkout HEAD~2
```

[Reset Animation](#)

Lightning fast!

Learn Git Branching

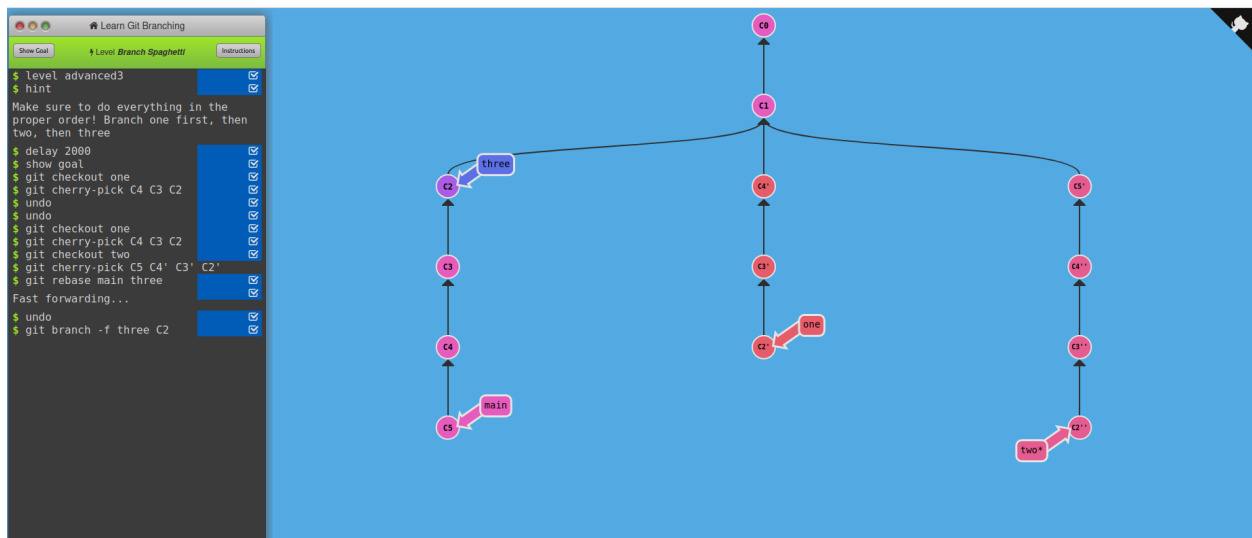
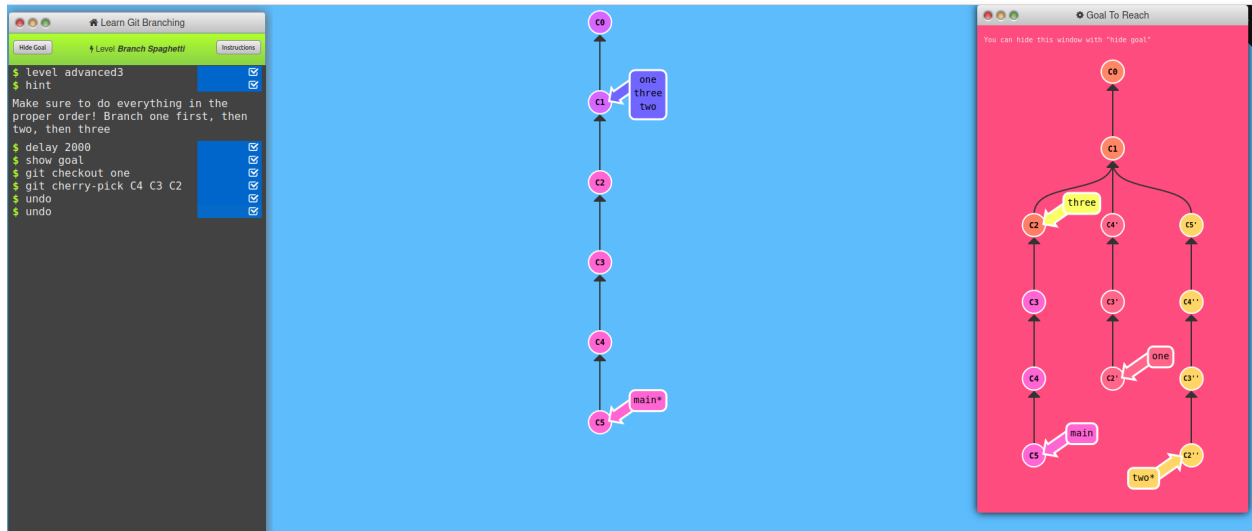
Show Goal **Level Multiple parents** Instructions

```
$ level advanced2
$ hint
Use 'git branch bugWork' with a target
commit to create the missing
reference.
$ delay 2000
$ show goal
$ git branch -f bugWork C2
```

```
git branch -f bugWork C2
```

## Task 3

In this task we have to rebase the branches one two and three and achieve the following goal.



```
git checkout one
git cherry-pick C4 C3 C2
git checkout two
git cherry-pick C5 C4' C3' C2'
git branch -f three C2
```

