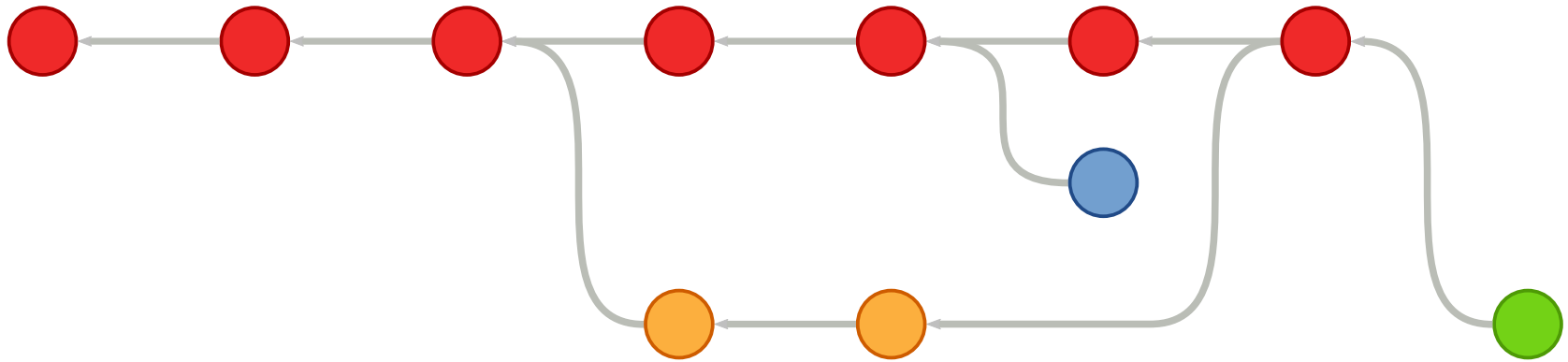
git

What happens at the repo-level



git

What happens at the repo-level

Pointers

Floating or not

You are where your HEAD's at

From one commit to another

Merging

With and without fast-forward

Rebasing

Meet your ancestors and rewrite history

git

What happens at the repo-level

Pointers

Floating or not

You are where your HEAD's at

From one commit to another

Merging

With and without fast-forward

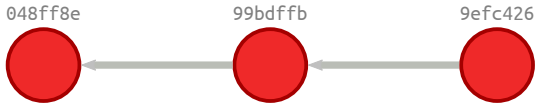
Rebasing

Meet your ancestors and rewrite history

A git repository is constituted of successions of commits

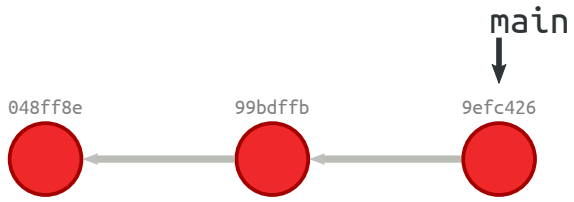
A git repository is constituted of successions of commits

which have each an individual SHA/checksum



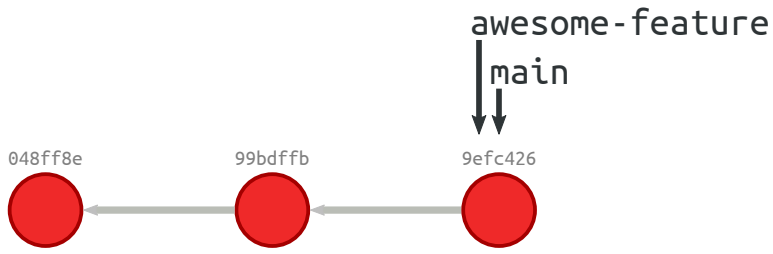
A branch is a pointer to a given commit

One can think of a “pointer” as an alias for the commit SHA



A branch is a pointer to a given commit

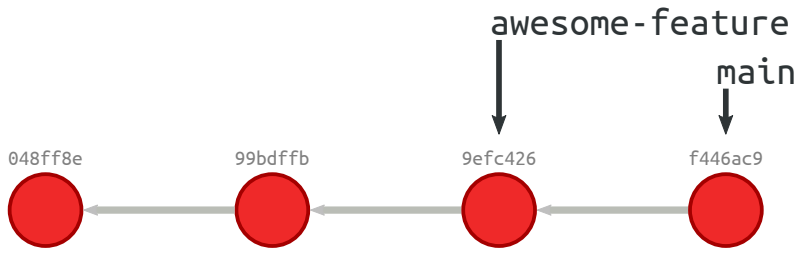
Creating a branch simply means creating another pointer




```
(main) $ git branch awesome-feature
```

A branch is a *floating* pointer to a given commit

They update when a new commit is added to the branch




```
(main) $ git commit
```

git

What happens at the repo-level

Pointers
: Floating or not

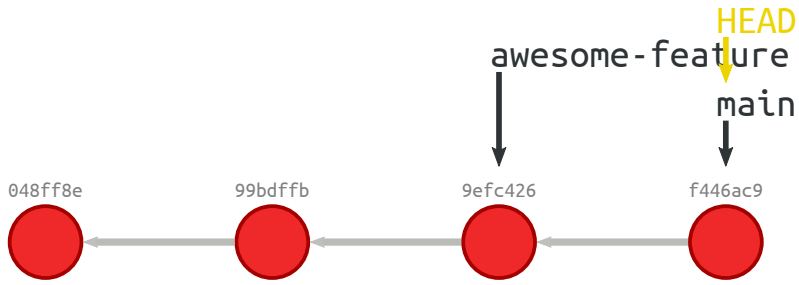
You are where your HEAD's at
: From one commit to another

Merging
: With and without fast-forward

Rebasing
: Meet your ancestors and rewrite history

HEAD is a pointer to a pointer

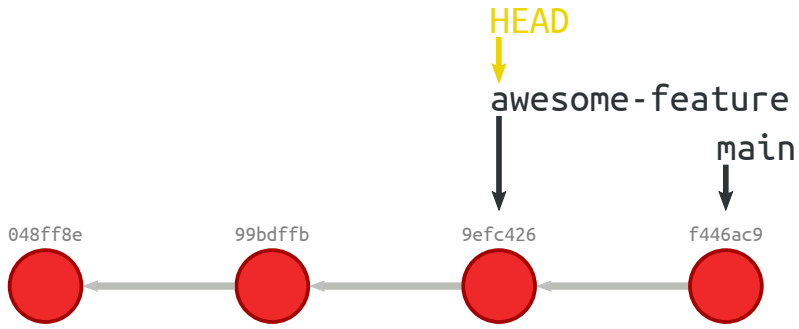
It indicates which commit is the base of your working directory




```
(main) $ git commit
```

HEAD is a pointer to a pointer

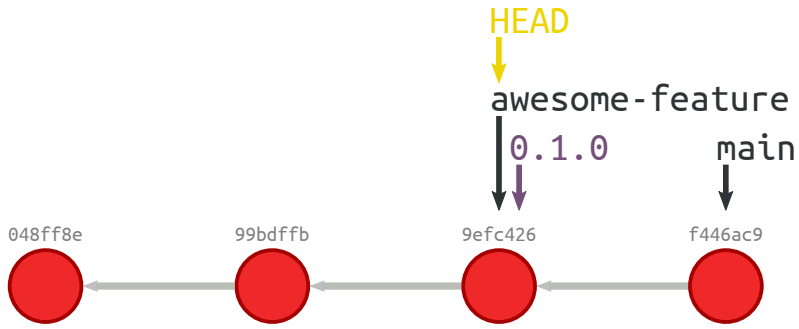
It indicates which commit is the base of your working directory




```
(main) $ git checkout awesome-feature
(awesome-feature) $
```

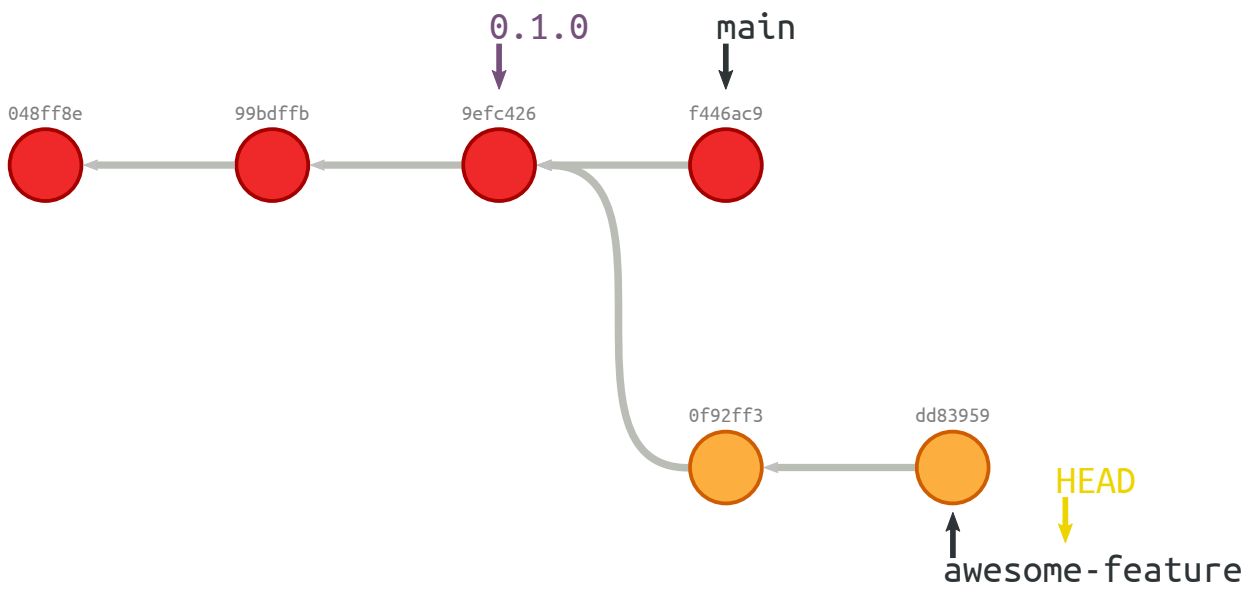
Tags are *non-floating* pointers

Well, you could but you oughtn't




```
(awesome-feature) $ git tag 0.1.0
```

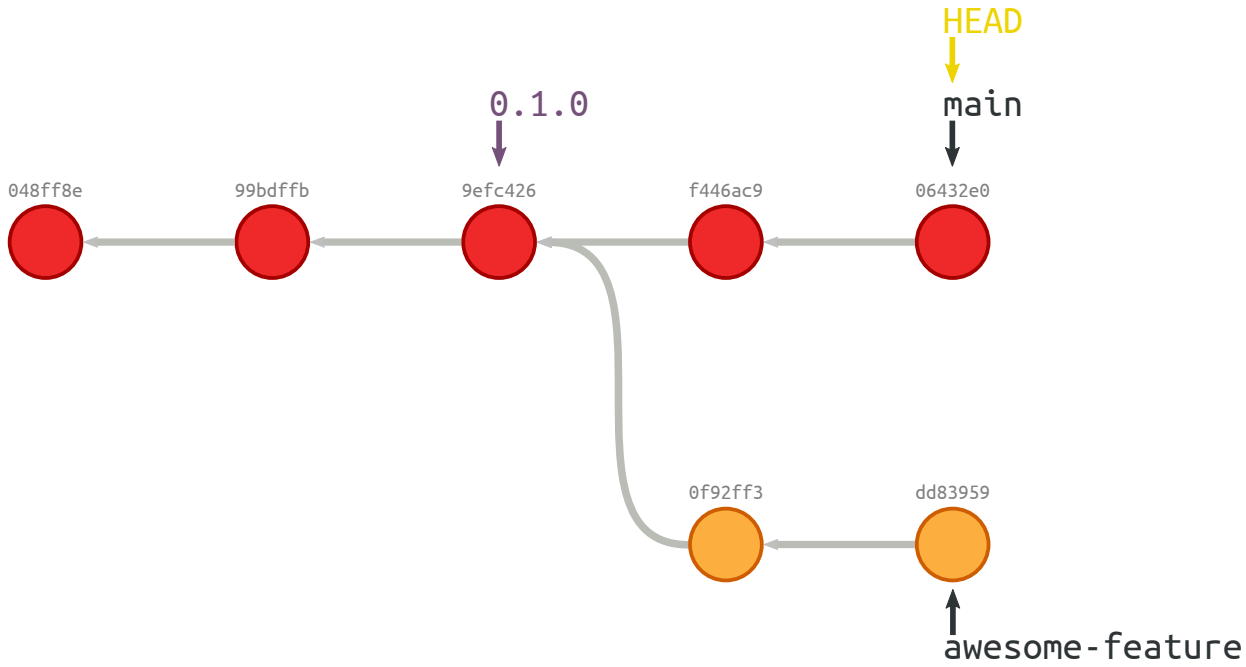
Tags are *non-floating* pointers




```
(awesome-feature) $ git commit
(awesome-feature) $ git commit
```

git switch is the new *git checkout*

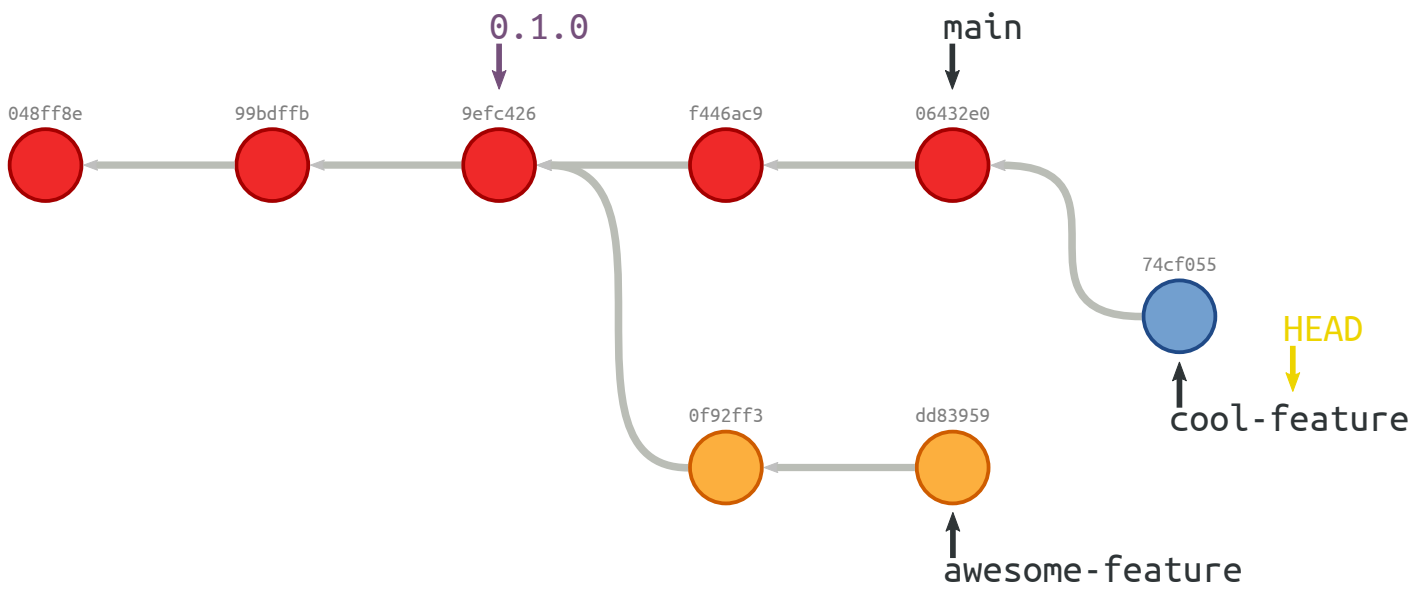
(when dealing with branches)




```
(awesome-feature) $ git switch main
(main) $ git commit
```

git checkout -b creates a branch and switches to it

all in one command




```
(main) $ git checkout -b cool-feature
(cool-feature) $ git commit
```

git

What happens at the repo-level

Pointers

Floating or not

You are where your HEAD's at

From one commit to another

Merging

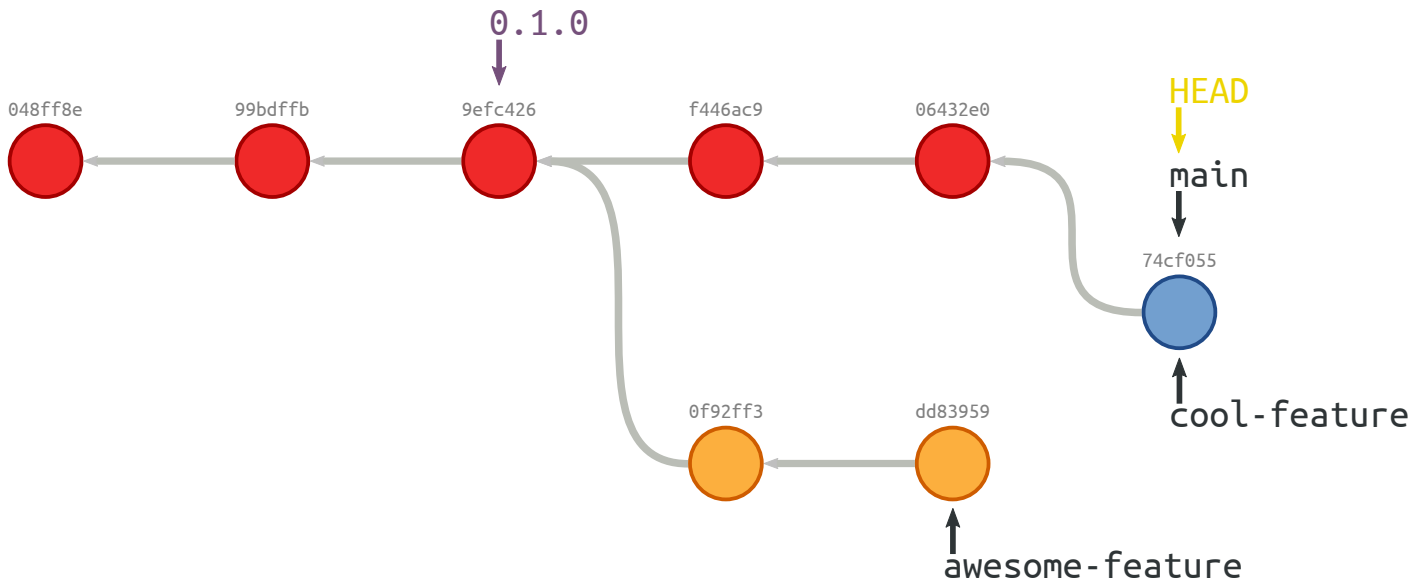
With and without fast-forward

Rebasing

Meet your ancestors and rewrite history

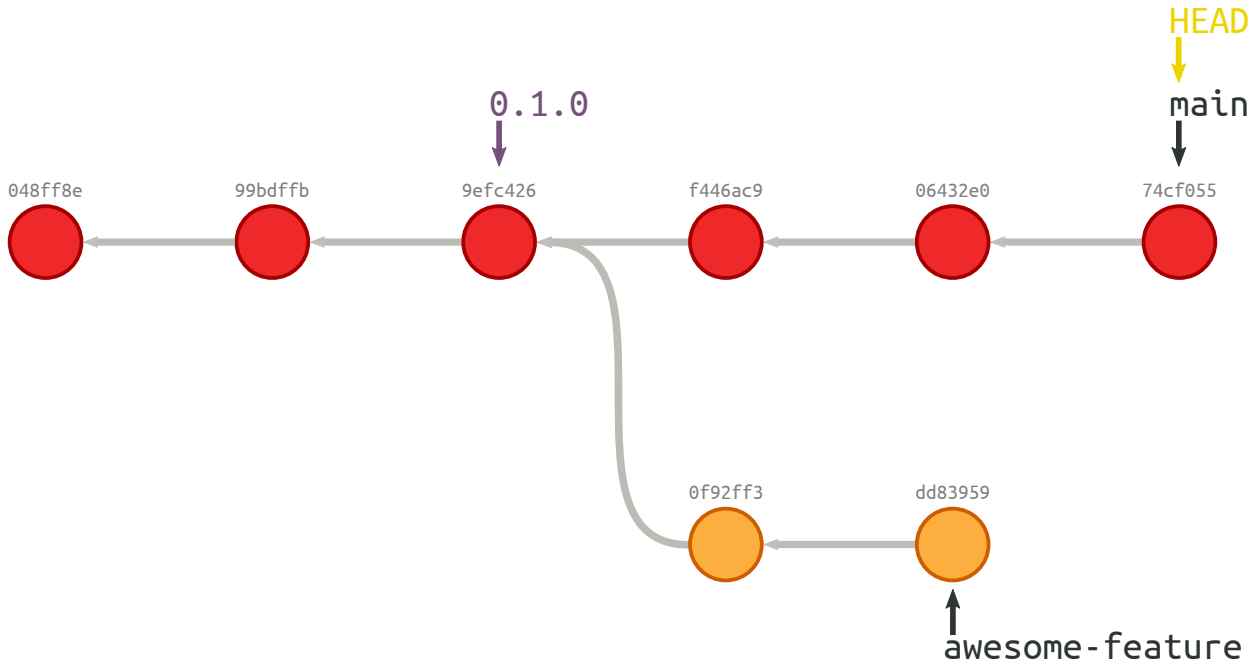
Merging means combining two snapshots

… which is simple when one can *--fast-forward*




```
(cool-feature) $ git checkout main
(main) $ git merge cool-feature
```

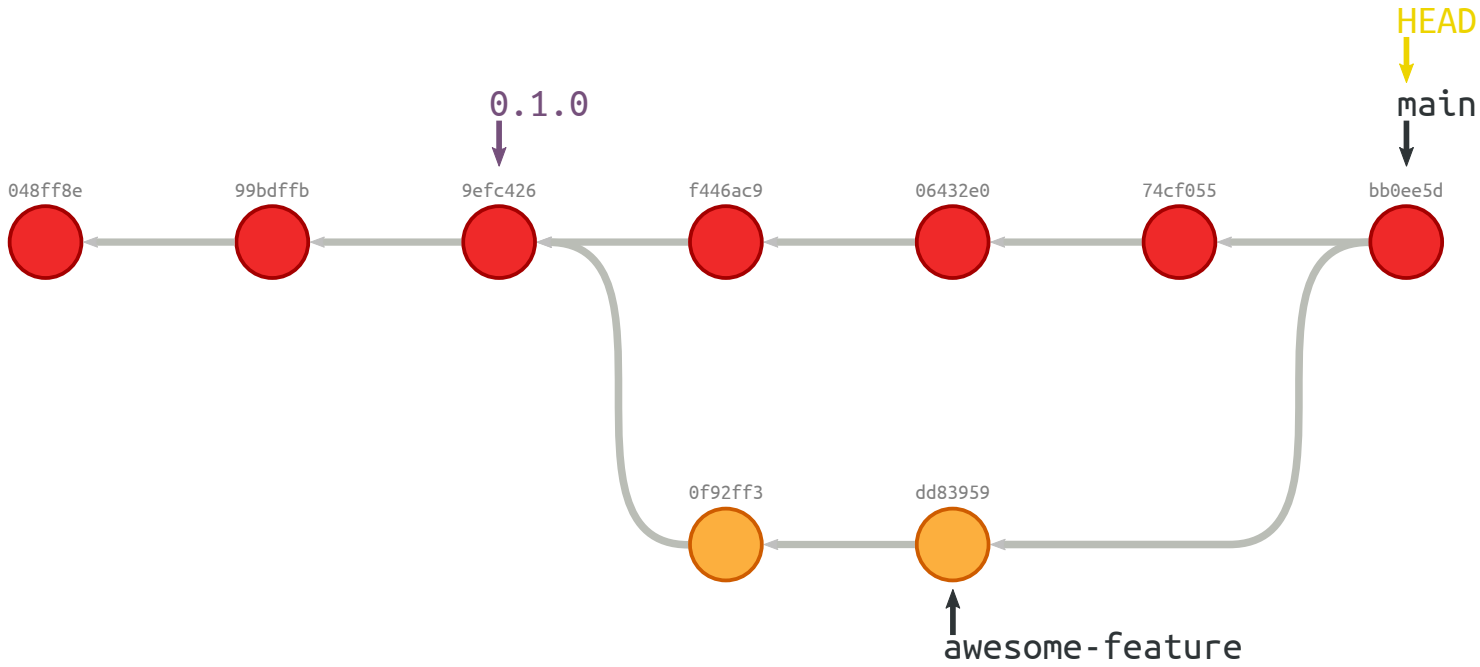
Colour and position don't matter to git




```
(main) $ git branch --delete cool-feature
```

A merge commit is created when fast-forward is not possible

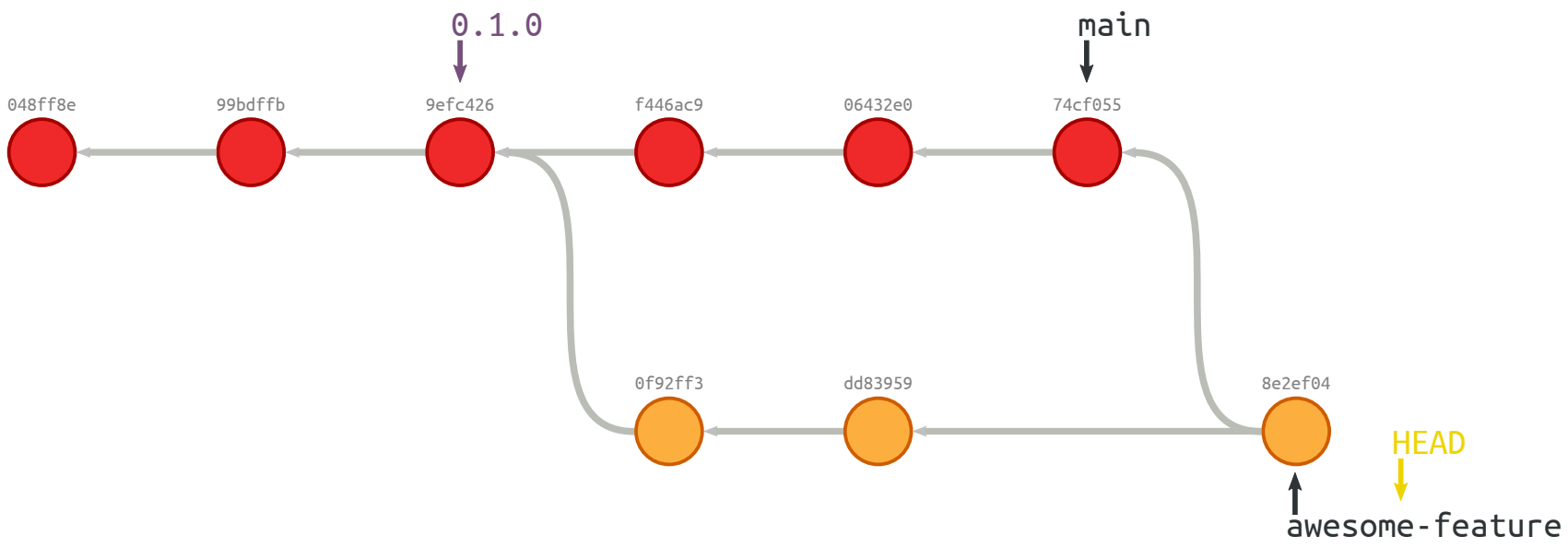
The *--no-ff* flag would force creating a merge commit; *--ff-only* wouldn't allow this merge




```
(main) $ git merge awesome-feature
```

Merging from ***main*** can be used to “update” a branch

but it is painful




```
(main) $ git switch awesome-feature
(awesome-feature) $ git merge main
```

git

What happens at the repo-level

Pointers

Floating or not

You are where your HEAD's at

From one commit to another

Merging

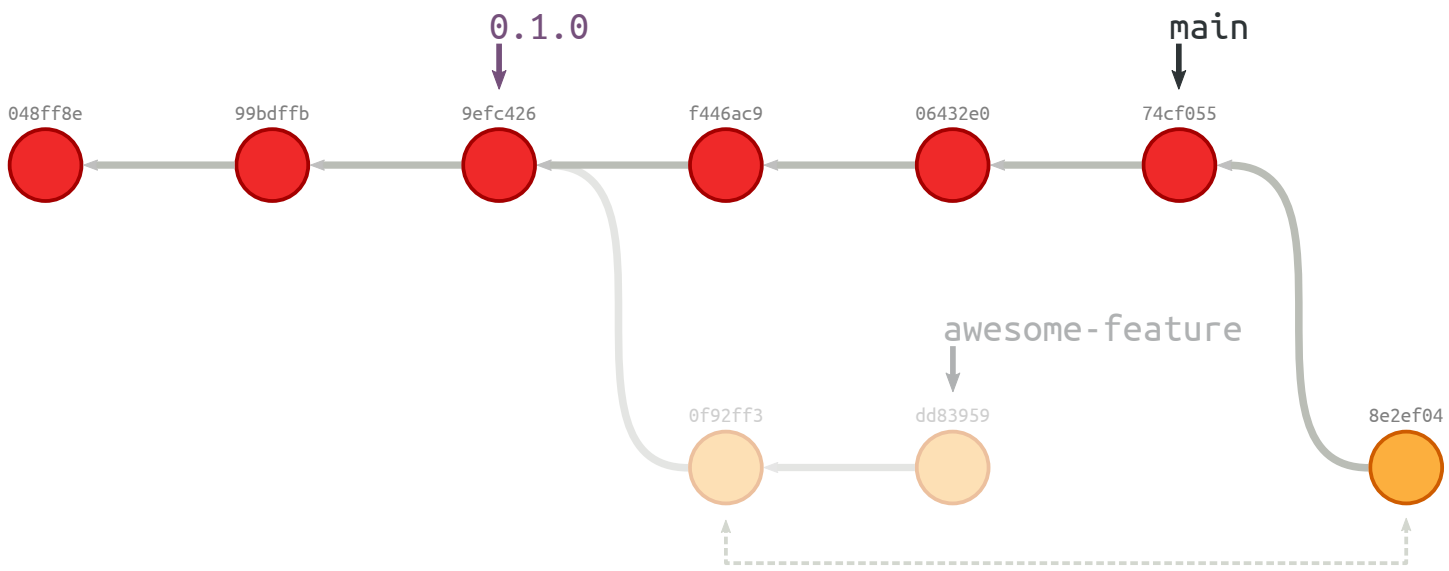
With and without fast-forward

Rebasing

Meet your ancestors and make history

Rebasing is updating the parent of the first non-common ancestor

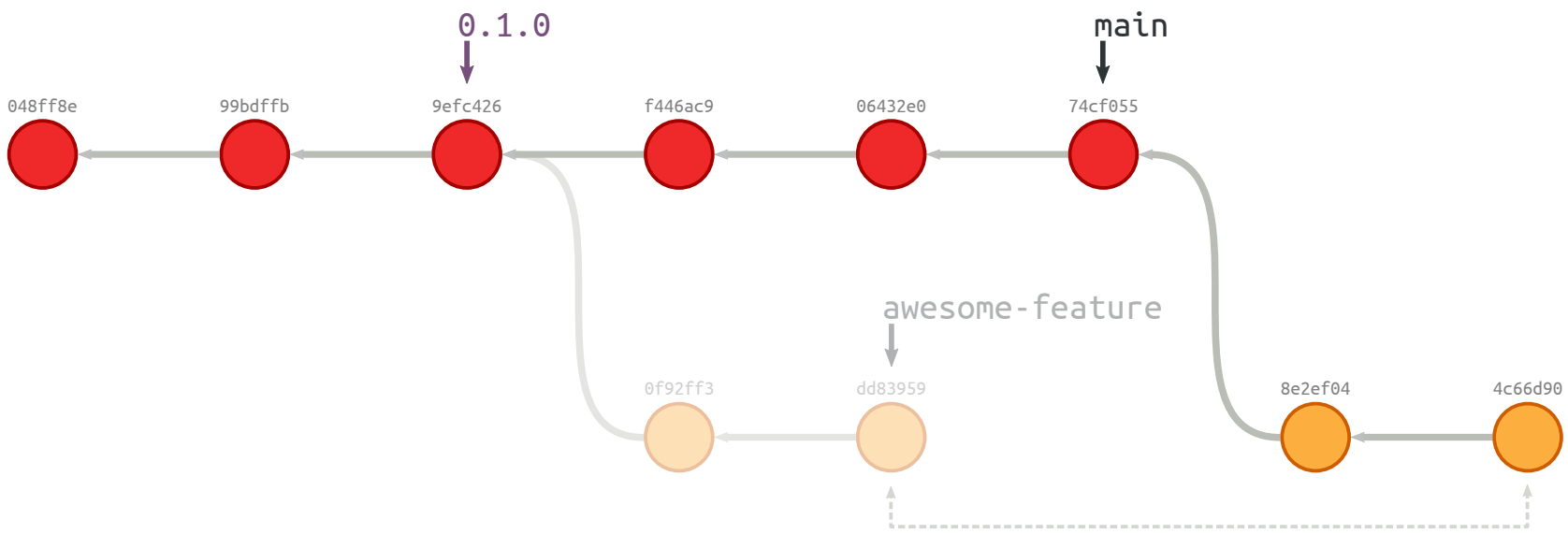
Each commit of the branch is updated iteratively




```
(awesome-feature) $ git rebase main
```

Rebasing is updating the parent of the first non-common ancestor

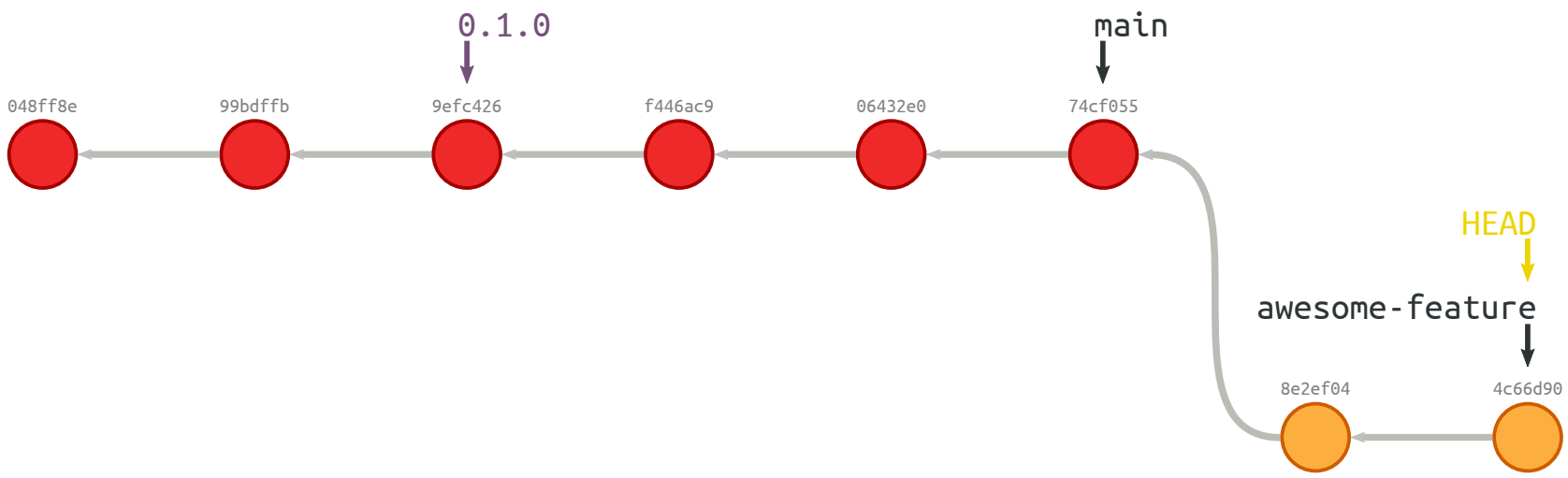
Each commit of the branch is updated iteratively




```
(awesome-feature) $ git rebase main
```

Rebasing is updating the parent of the first non-common ancestor

Each commit of the branch is updated iteratively




```
(awesome-feature) $ git rebase main
```