Școala
informală
de IT

GIT and Version Control

Do, Undo, Collaborate

Agenda

- What is Git?
- Initializing a repo
- Adding
- Committing
- Branching
- Merging
- Tips and Tricks

What is Git?

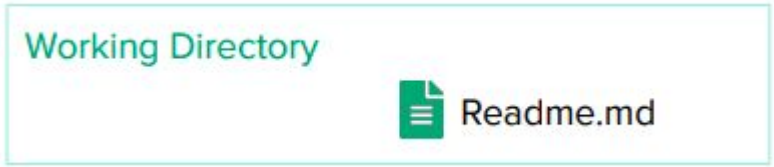
A version control tool (or system, hence the abbreviation VCS), for tracking changes in source code and enabling collaboration, coordination and integrity.

Works best for text files and worst for binary formats.

It’s an indispensable tool for working on the same codebase in a team.

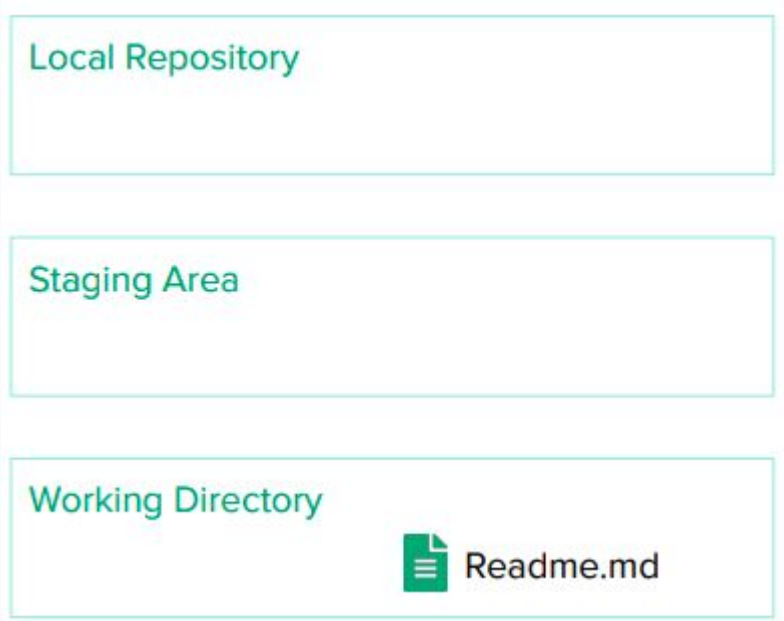
It’s a distributed version control tool.

Initializing a new repository



Initializing a new repository

```
$ git init

$ |
```



Glossary

Working directory

● The folder we have locally and in which we work and perform our changes

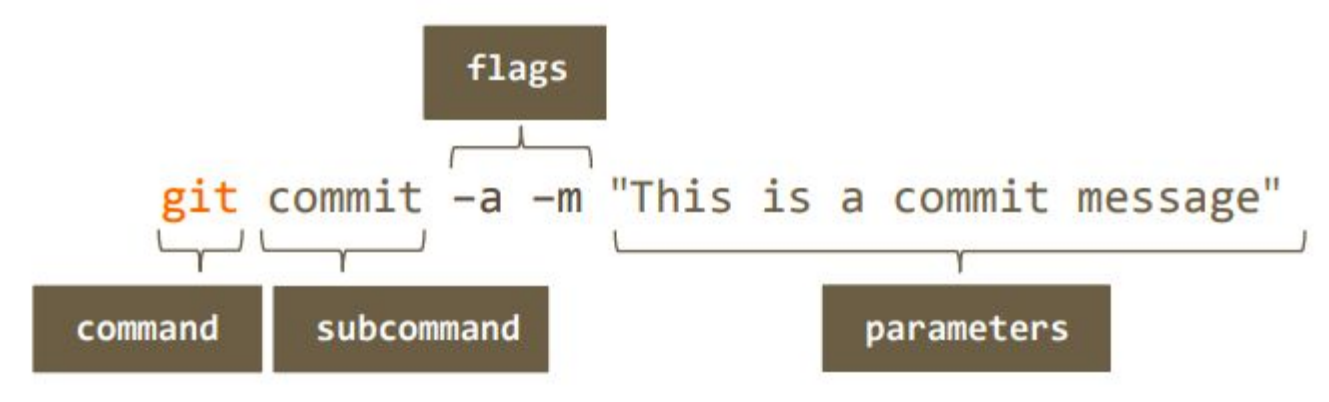
Staging area or Index

● A snapshot of the working directory at a certain point in time, typically holds all changes that are to be part of a new version

Repository

● A collection of all different versions (snapshots) of the working directory ever “recorded”, also called commits

Anatomy of a command



Anatomy of a command

- Flags can be one letter or full word, if the full word is used it’s delimited by a double dash

```
git push -u origin main
git push --set-upstream origin main
```

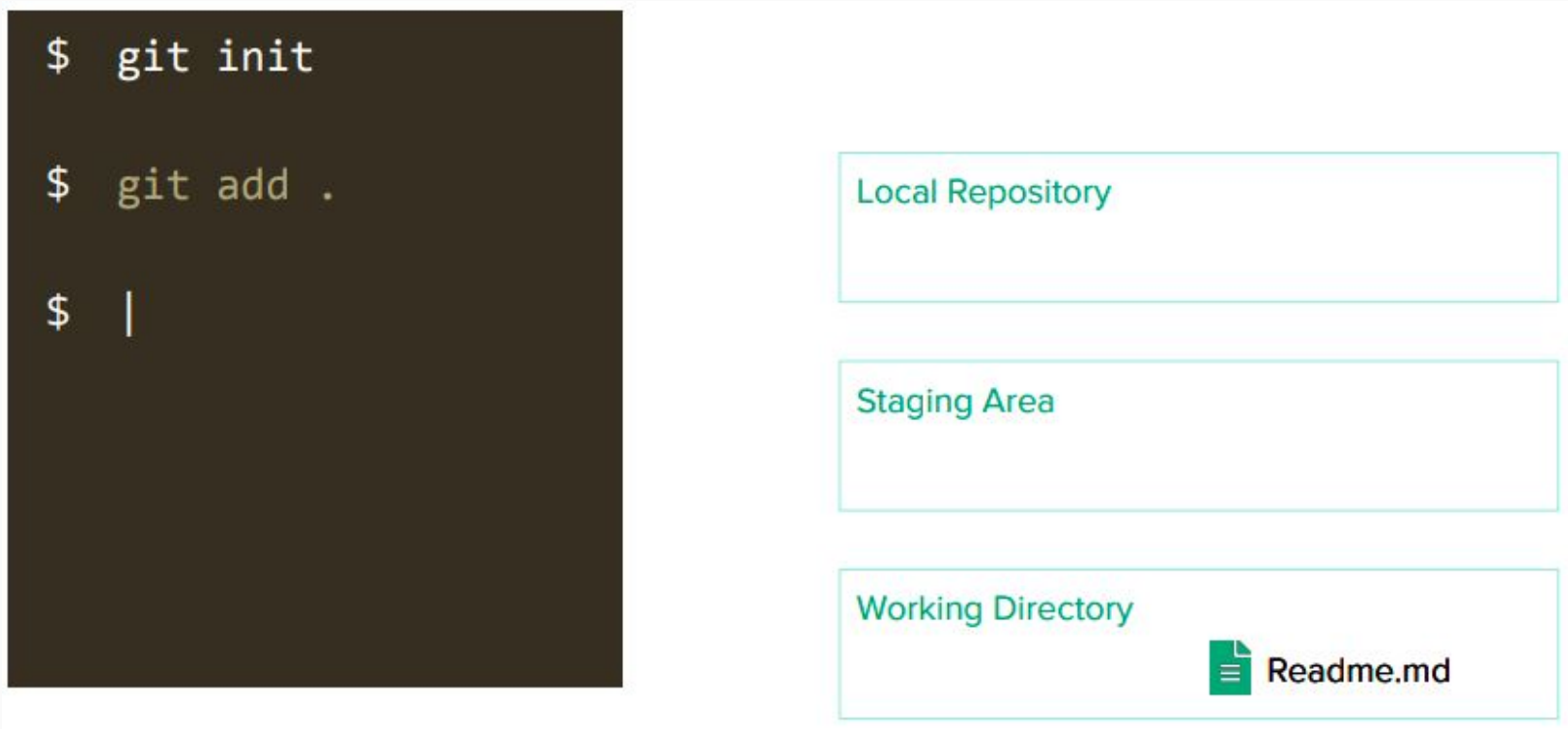
- You can easily get help on a command by using git help

```
git help commit
```

Basic Commands

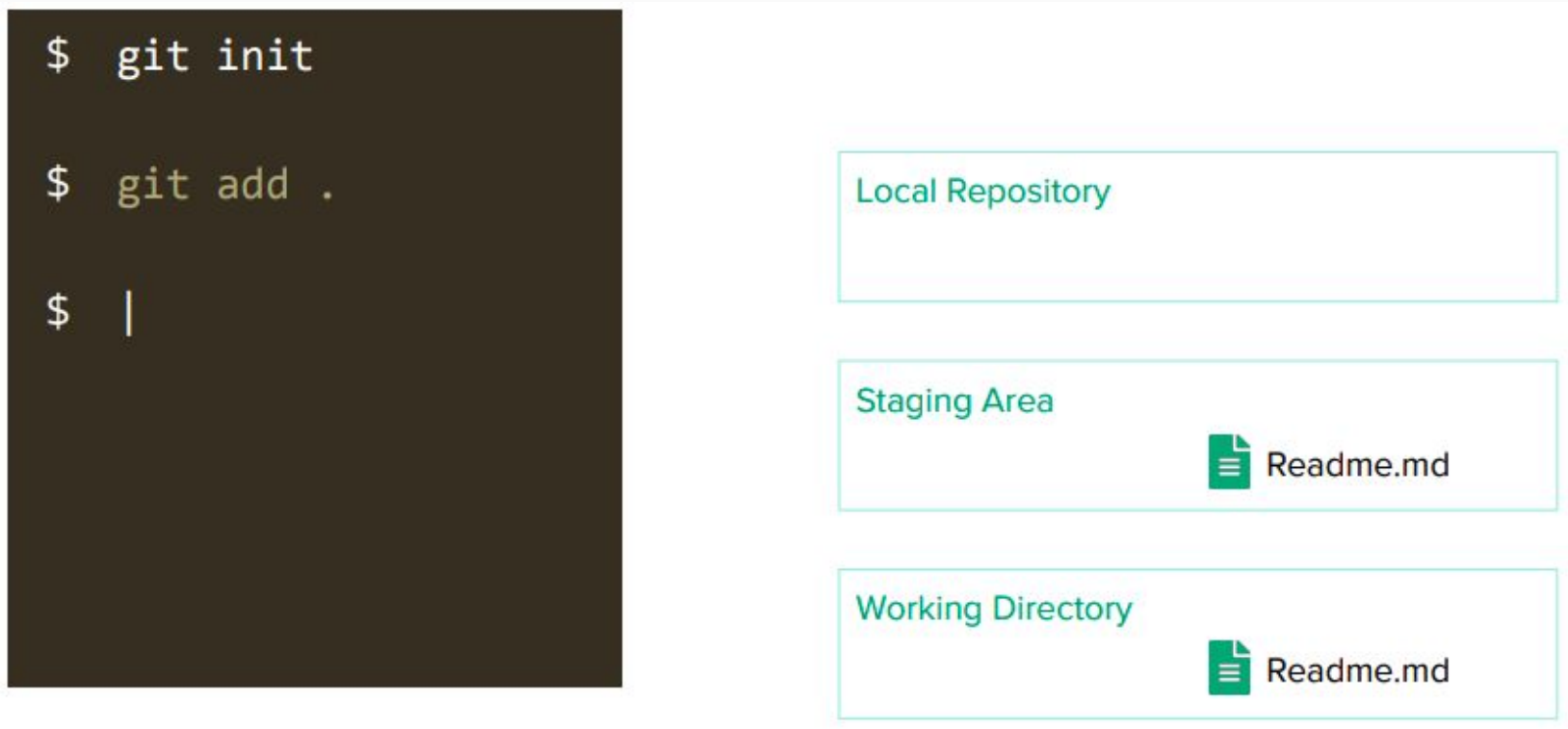
git add

```
$ git init

$ git add .

$ |
```



git add

```
$ git init

$ git add .

$ |
```



git add

Creates a snapshot of the current state of the working directory in the Staging Area.

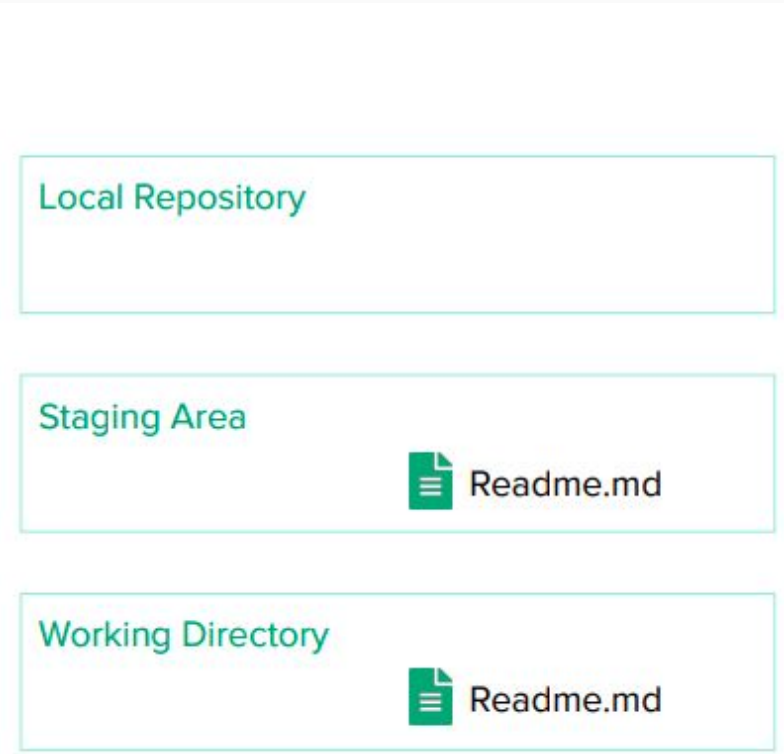
Essentially copies all changes in the working directory to the Staging Area so that they are prepared to be committed.

Git add accepts a path as a parameter.

```
git add Readme.md
git add folder/subfolder/file.js
git add .
git add folder/subfolder
```

git commit

```
$ git init

$ git add

$ git commit -m "Abc"

$ |
```



git commit

```
$ git init

$ git add

$ git commit -m "Abc"

$ |
```

git commit

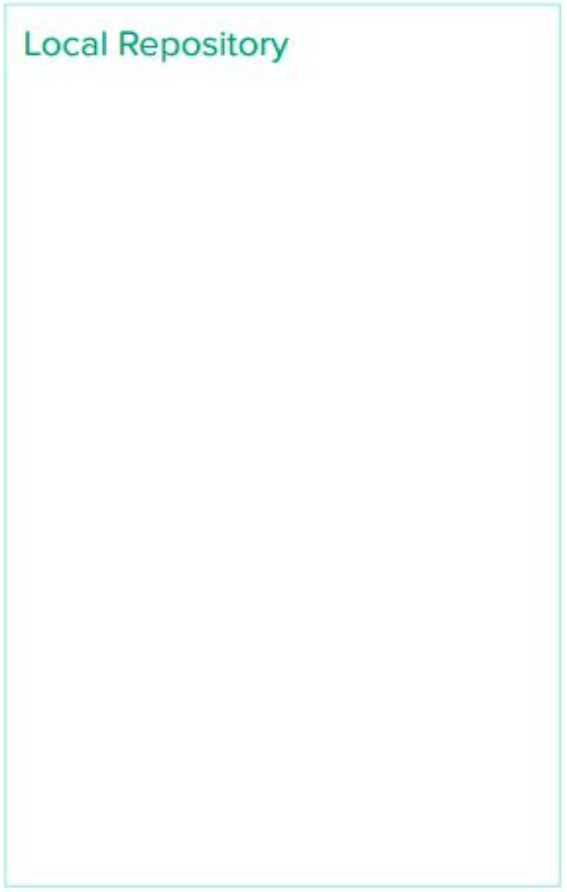
Creates a new version with all the changes from the staging area.

Commits contain whole files not only differences.

Commits have a SHA1 hash as their unique ID, it is calculated based on file contents, commit message and the parent of the commit.

```
git commit -m "Abc"
git commit
git commit -am "Abc"
```

git commit



git commit

```
$ git commit

$
```

git commit

```
$ git commit

$ git commit

$ |
```

git commit

```
$ git commit

$ git commit

$ |
```

Branching

Branches

Branches are just labels attached to certain commits.

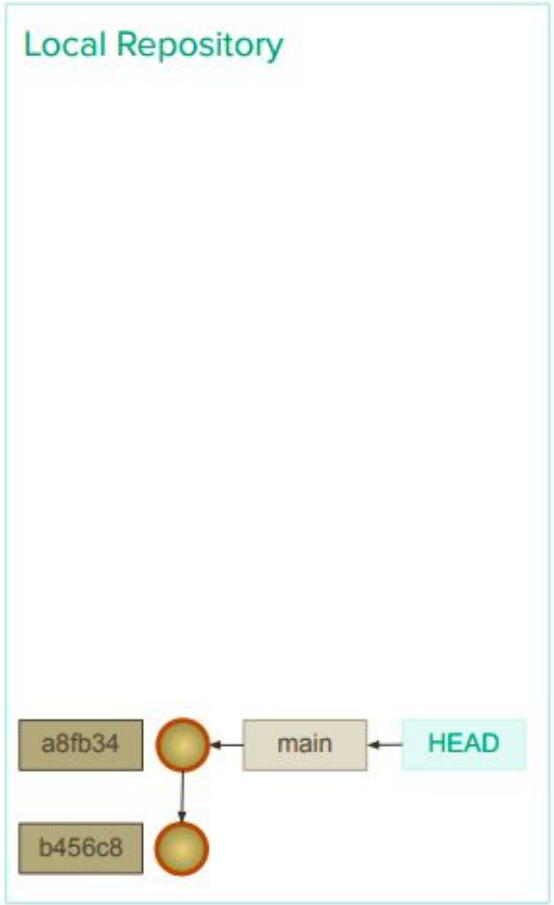
They are an easy, human-readable, way of accessing a commit.

Branches are used to separate work in progress until it is ready to be merged with the main codebase.

```
git branch  <branch name>
git switch -c <branch name>
```

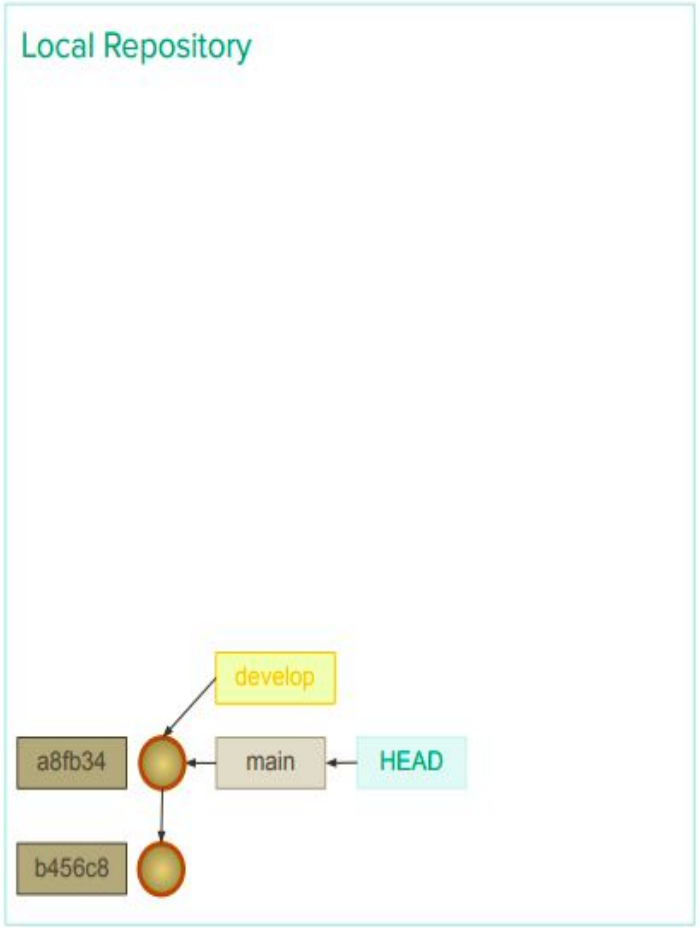
Branching

```
$
```



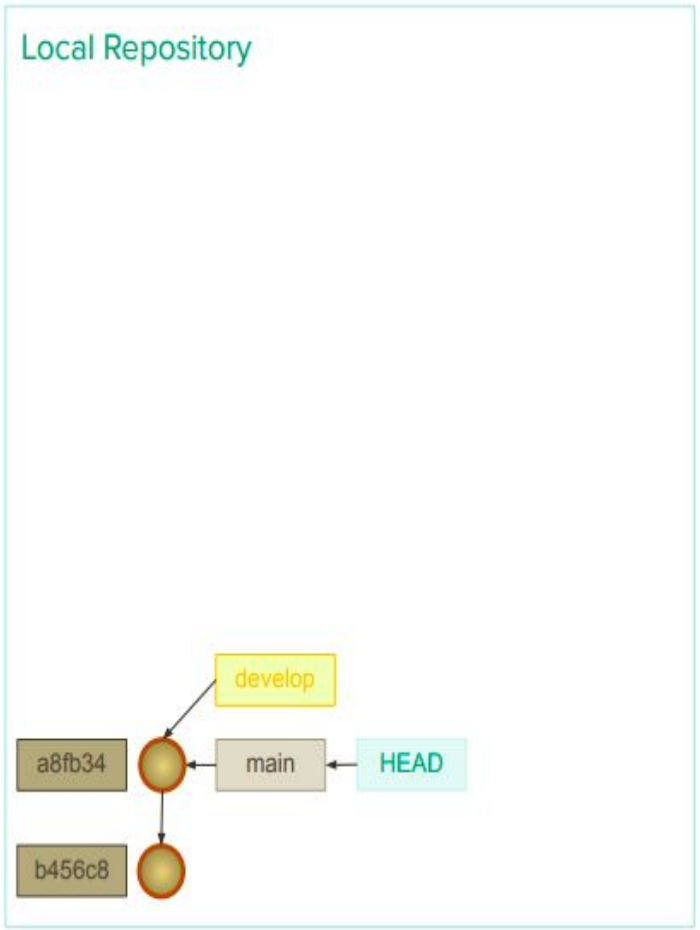
Branching

```
$ git branch develop

$ |
```



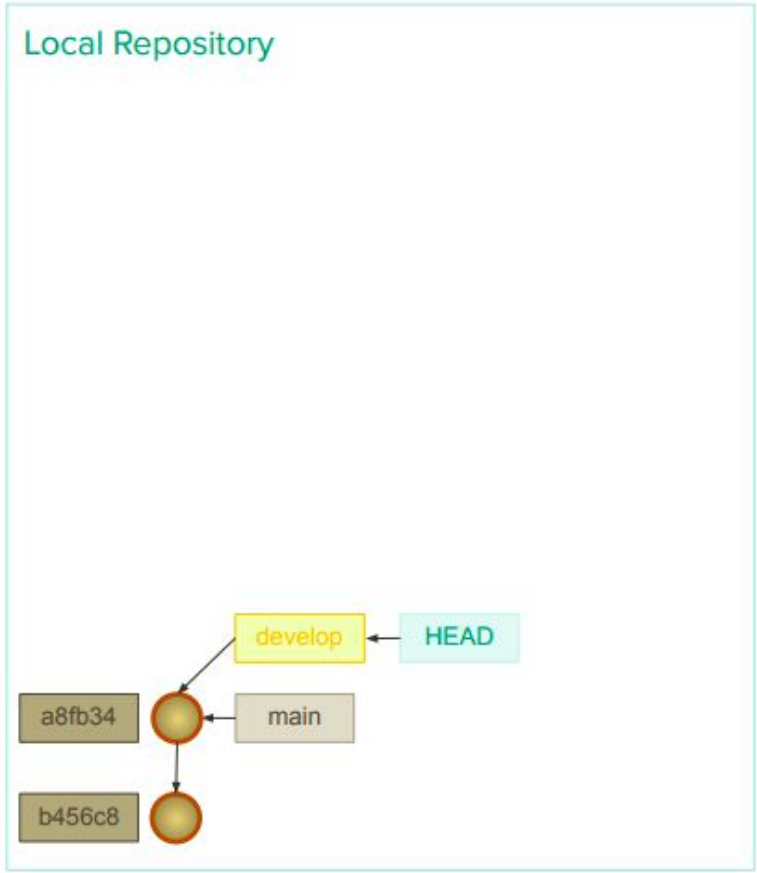
Branching

```
$ git branch develop

$ git switch develop

$ |
```



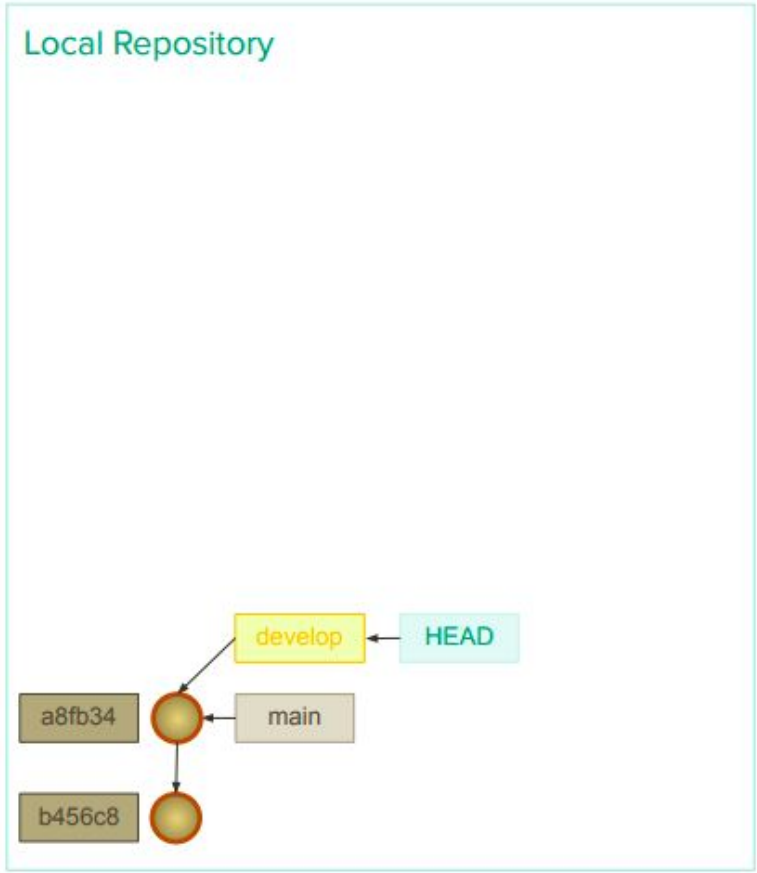
Branching

```
$ git branch develop

$ git switch develop

$ |
```



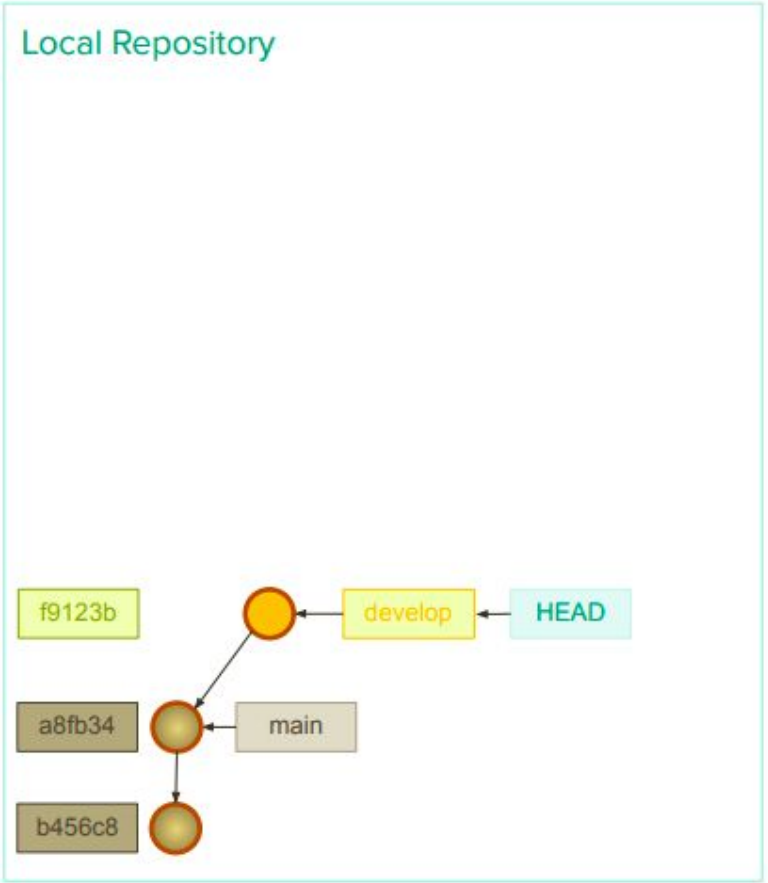
Branching

```
$ git branch develop

$ git switch develop

$ git commit

$ |
```



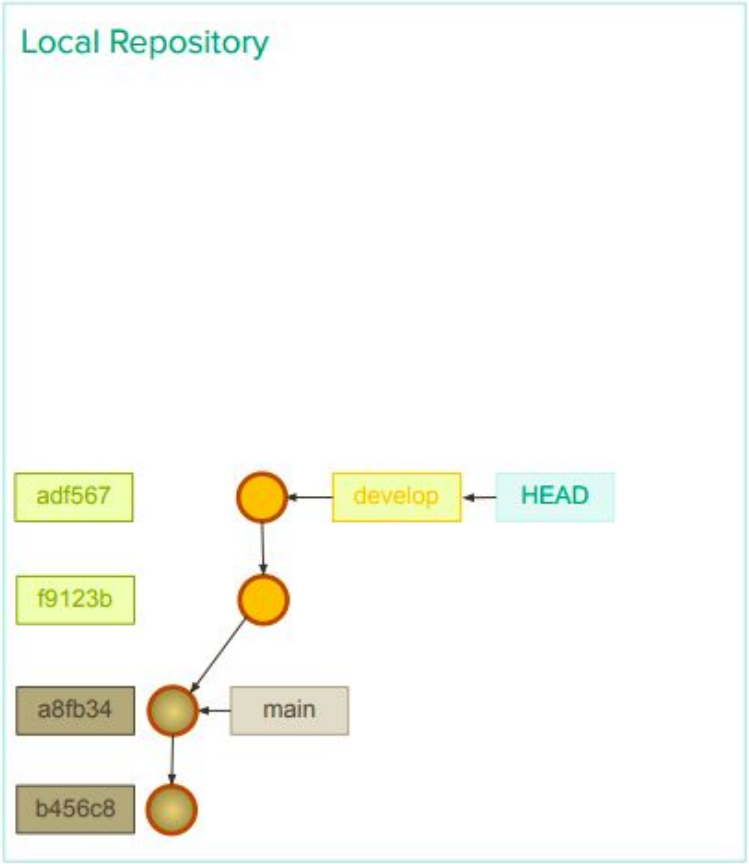
Branching

```
$ git branch develop

$ git switch develop

$ git commit

$ |
```



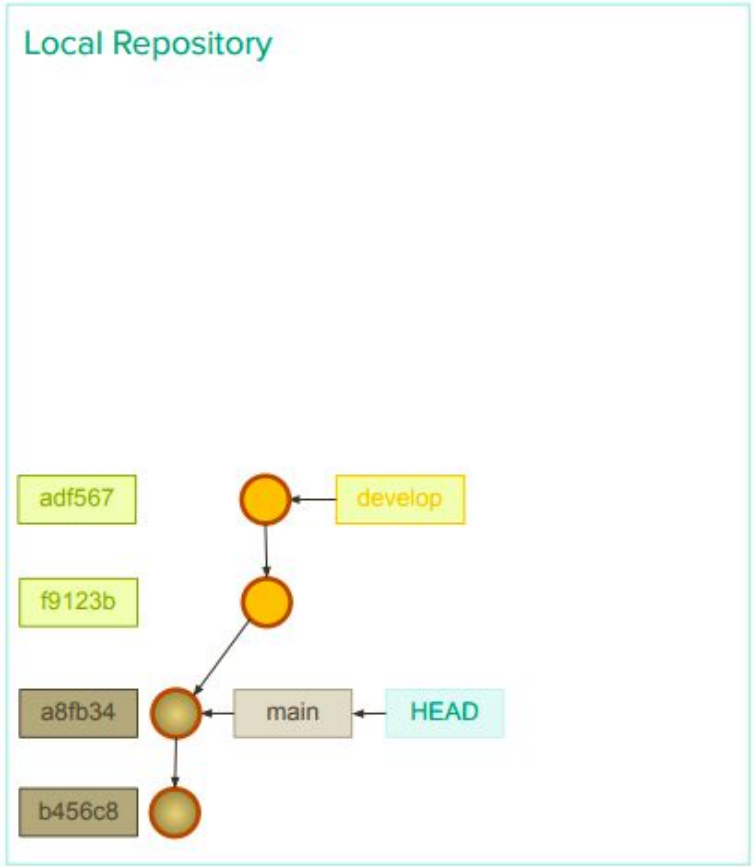
Branching

```
$ git branch develop

$ git switch develop

$ git commit

$ git commit

$ |
```



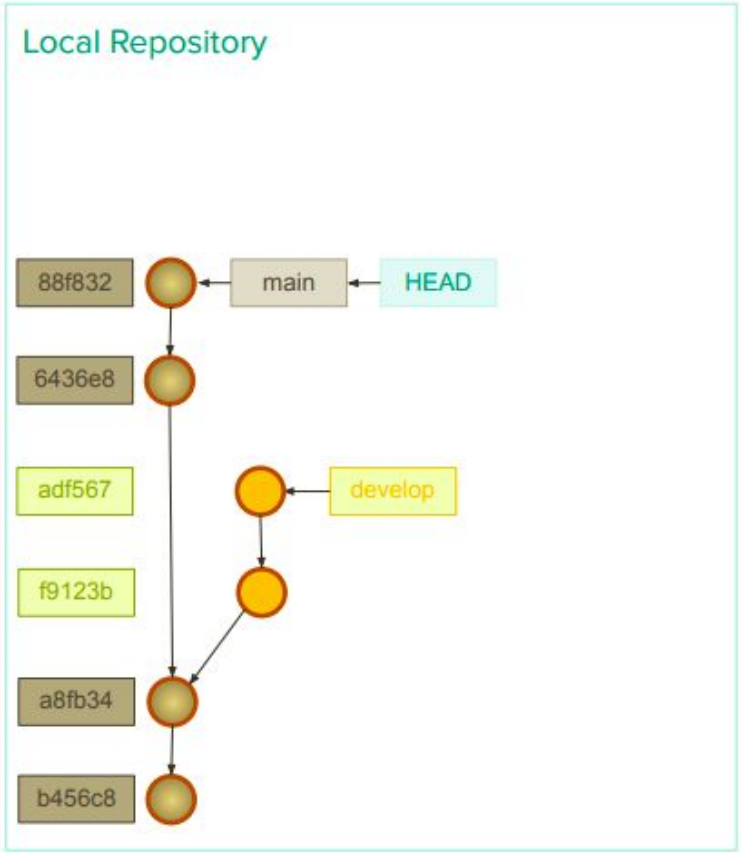
Branching

```
$ git switch develop

$ git commit

$ git commit

$ git switch main

$ |
```



Branching

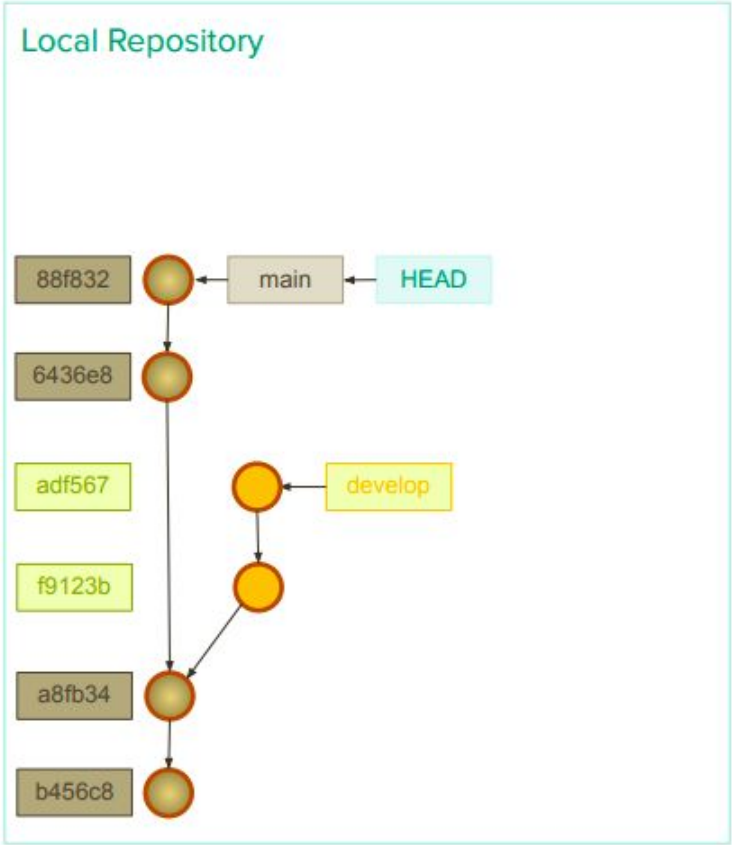
```
$ git commit

$ git switch main

$ git commit

$ git commit

$ |
```



Merging Changes

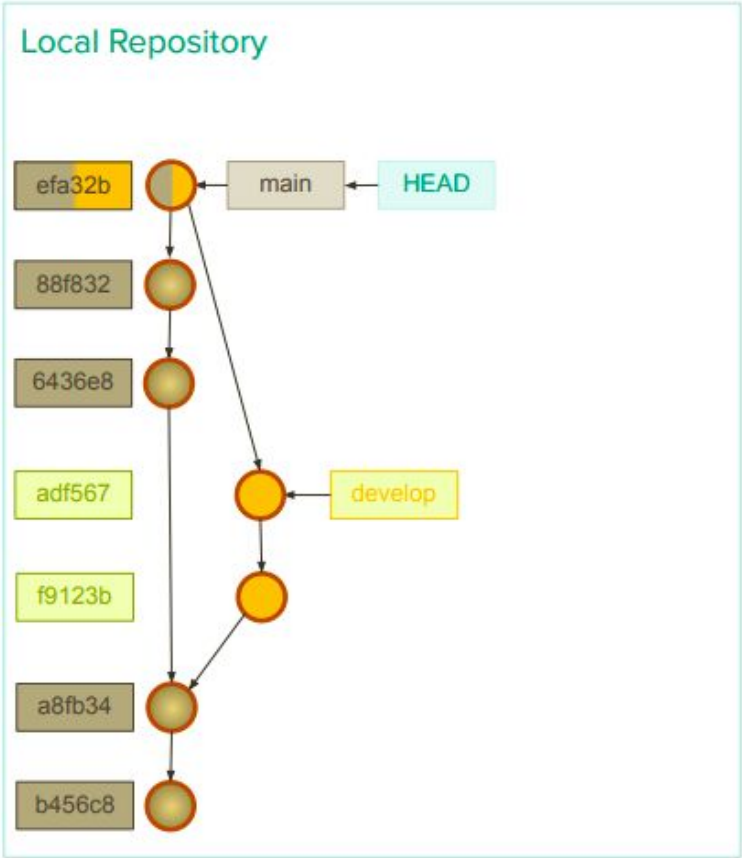
Merging

```
$ git merge develop

$ |
```



Merging

```
$ git merge develop

$ |
```

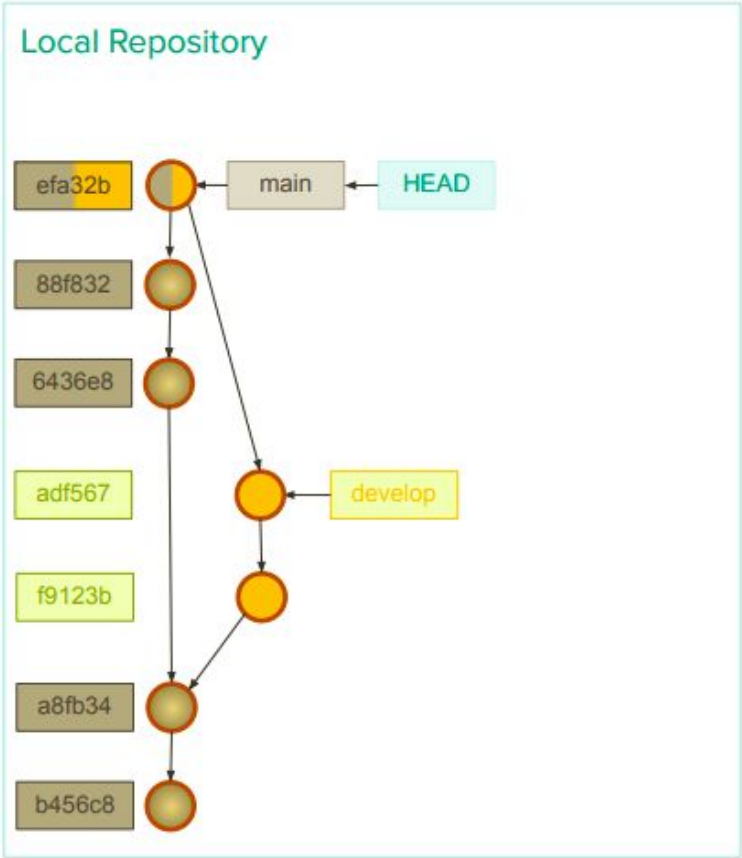


Merging

- In the case where the branches have diverged a **merge commit** is created
- A merge commit is a commit with two parent commits and incorporates changes from both parents
- Merging can cause conflicts which have to be solved only once for the merge to finish

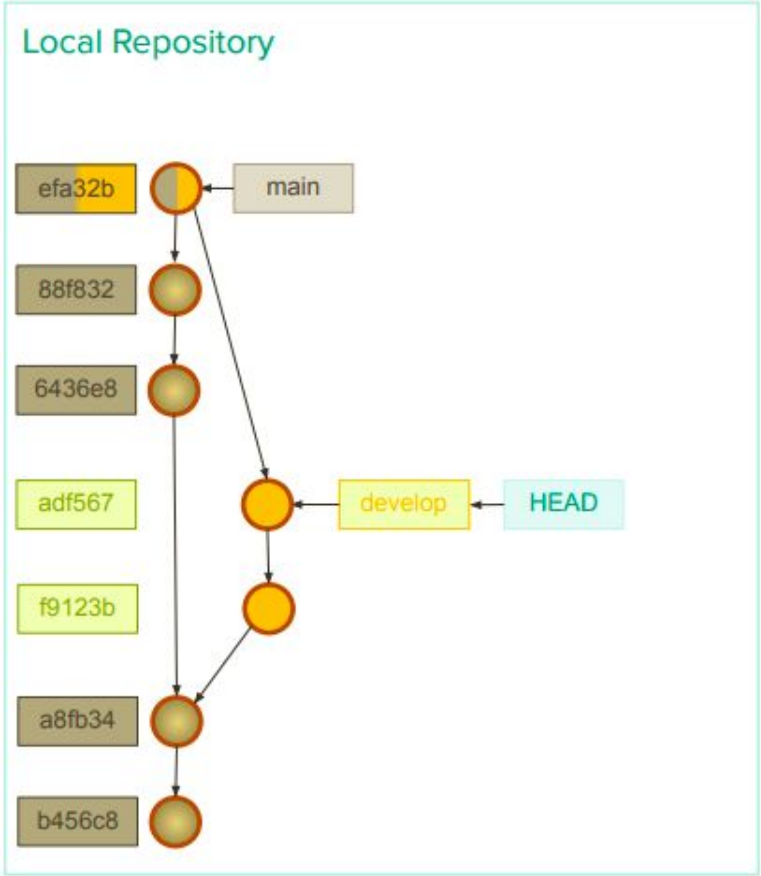
Merging

```
$ git merge develop

$ |
```



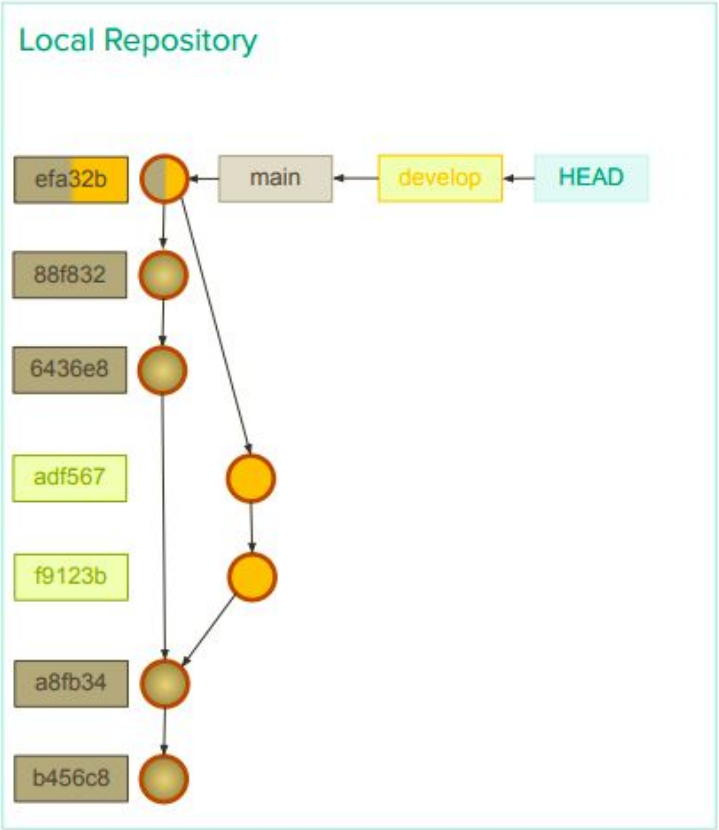
Merging

```
$ git merge develop

$ git switch develop

$ |
```



Merging

```
$ git merge develop

$ git switch develop

$ git merge main

$ |
```



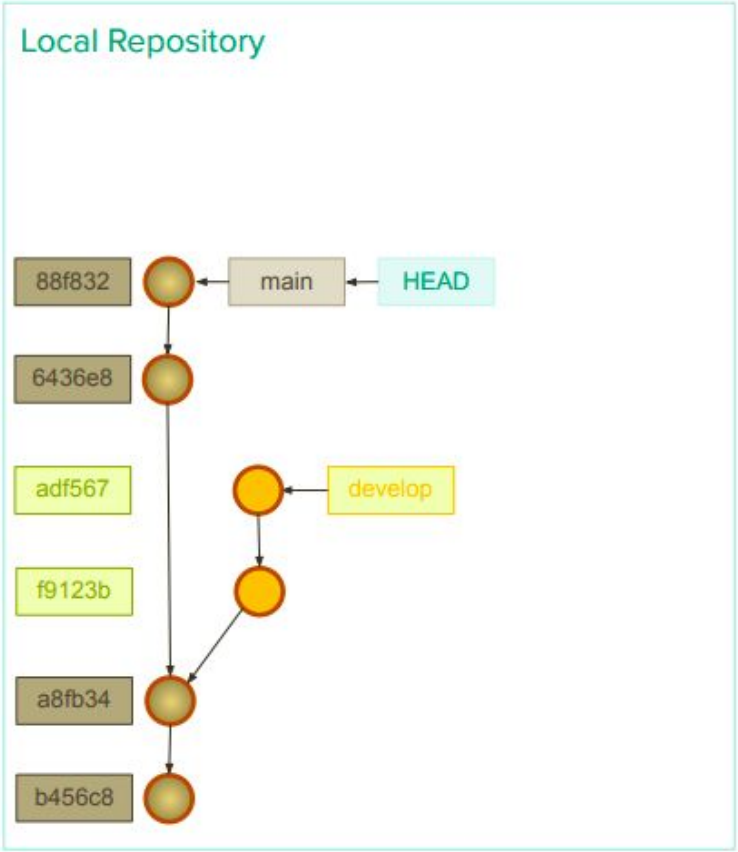
The Fast Forward

- In the case where we **CAN** connect to the latest commit on the other branch by going straight back along the line of commits a **Fast Forward** is possible
- A fast forward just moves the current branch to the tip of the other branch, no commits are created
- A fast forward cannot end in a conflict by definition
- You can force the creation of a merge commit if you want to even if fast forwarding is possible (git merge develop --no-ff)
- You can constrain merges to only happen if fast forwarding is possible (--ff-only)

Undo changes in a commit

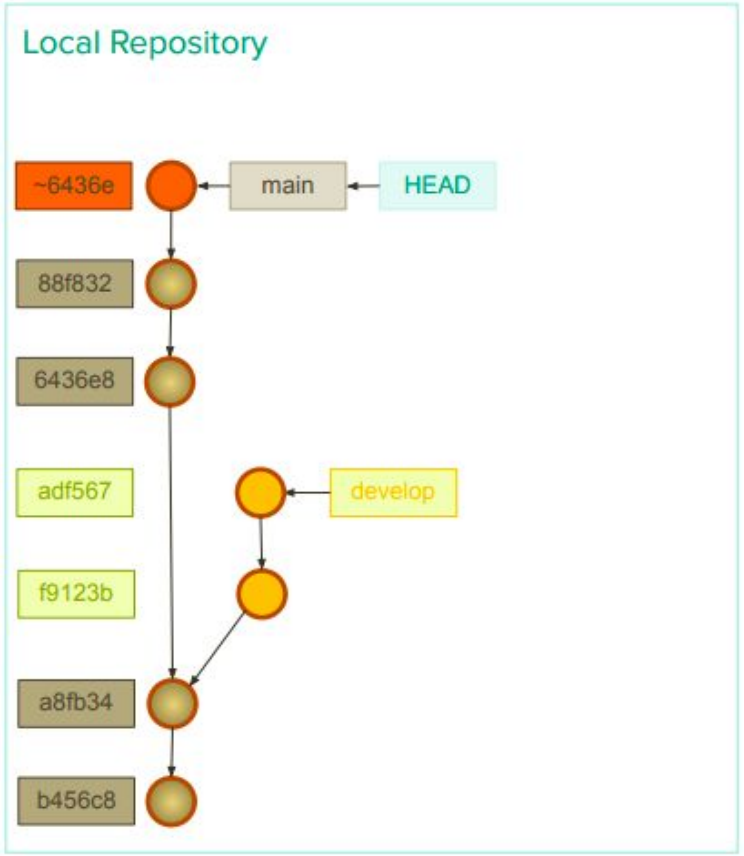
Undoing a commit

```
$ git revert 6436e8

$
```



Undoing a commit

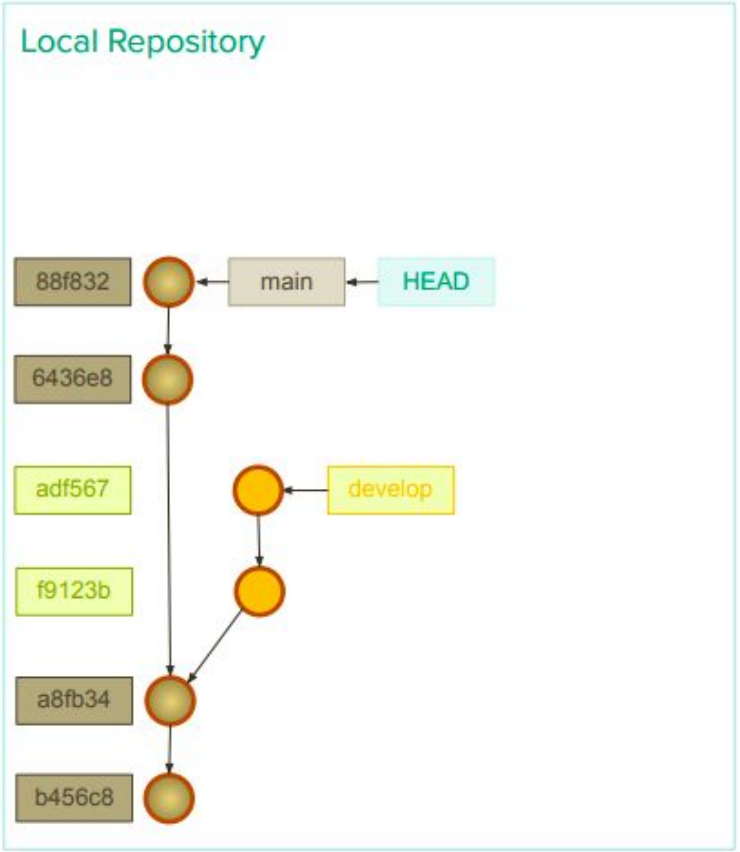
```
$ git revert 6436e8

$
```



Update commit message

Update message

```
$ git commit --amend
  -m "New Message"

$
```



See a graph of commits

git log

```
git log --graph --oneline --decorate --all
```

```
git config --global alias.l "log --graph --oneline --decorate --all "
```

```
git l
```