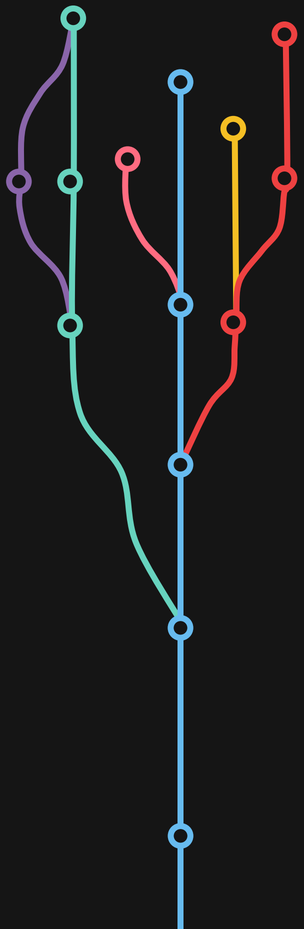
Gitting Started

An introduction to Git

Bailey Parker



bit.ly/GittingStarted

Many people can collaborate concurrently

Distributed Version Control System

Tracks the history of files in a project

Sketchy Alternatives

Keeping code in Dropbox, Google Drive, iCloud, etc.

Reply All email chains for distributing code

Making periodic copies of your code

Benefits of Git

Fine-grained history

Forgiving of mistakes

Multiple people can concurrently work on multiple features

Extensive infrastructure and tooling

Installing Git

Use the Ubuntu Subsystem

```
brew install git
```

```
apt install git
```

The Basics

Cloning a Repository

Download a copy

Directory for which git keeps history

```
git clone https://github.com/tensorflow/tensorflow
```

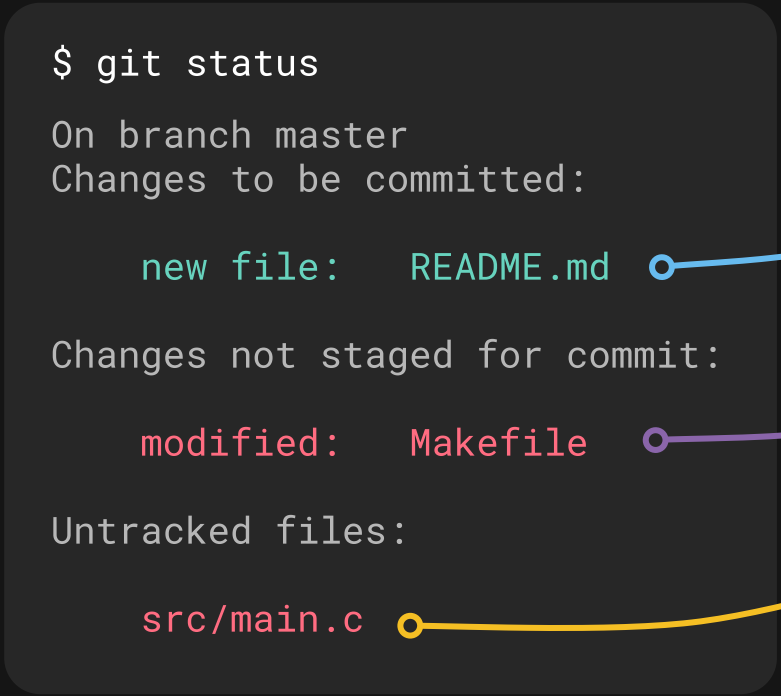
Setup

```
git config --global user.name "Your Name"

git config --global user.email "your@email.com"
```

Creating a Repository

```
git init my-new-repo
```

Checking Repository Status



```
$ git status

On branch master
Changes to be committed:

    new file:   README.md

Changes not staged for commit:

    modified:   Makefile

Untracked files:

    src/main.c
```

Stage changes to prepare to add them to the history

Unstaged changes won’t be added to the history

Untracked files aren’t yet being watched by git

Staging Changes

Prepare to add changes to the history

```
$ git add Makefile src/main.c

$ git status

On branch master
Changes to be committed:

    modified:   Makefile
    new file:   README.md
    new file:   src/main.c
```

Staging Changes

```
$ mkdir bin && touch bin/.gitkeep

$ git add bin/.gitkeep

$ git status

On branch master
Changes to be committed:

    modified:   bin/.gitkeep
    modified:   Makefile
    new file:   README.md
    new file:   src/main.c
```

Git only deals with files, so to add empty directories we must place an empty file inside them

Ignoring Files

```
$ git status

On branch master
Untracked files:

    __pycache__

$ echo __pycache__ >> .gitignore

$ git status

On branch master
nothing to commit
```

We almost never want to track executables, cache files, etc.

GitHub provides a repo of common .gitignore files

Ignoring Files Globally

```
$ git config --global core.excludesfile ~/.gitignore_global
$ vim ~/.gitignore_global
```

Committing Changes

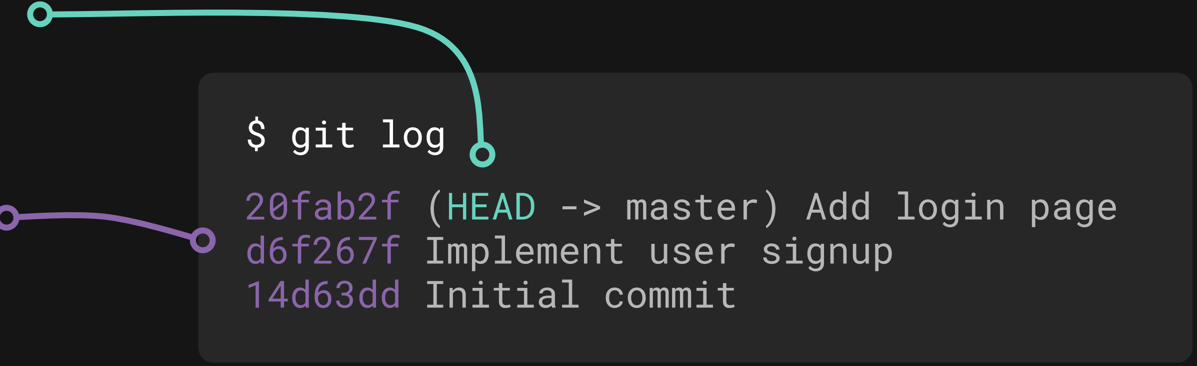
```
$ git status

On branch master
Changes to be committed:

    modified:   bin/.gitkeep
    modified:   Makefile
    new file:   README.md
    new file:   src/main.c

$ git commit -m "Initial commit"

$ git status

On branch master
nothing to commit
```

Add staged changes to the history

Keep commits small and focused on a single change

Commit messages are short, imperative summaries of what you changed

Viewing the History


HEAD is the current point in time
SHAs uniquely identify a commit

```
$ git log

20fab2f (HEAD -> master) Add login page
d6f267f Implement user signup
14d63dd Initial commit
```

Better log

```
$ git config --global alias.lg "log --graph
--pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr)
%C(bold blue)<%an>%Creset' --abbrev-commit --date=relative"

$ git lg
```

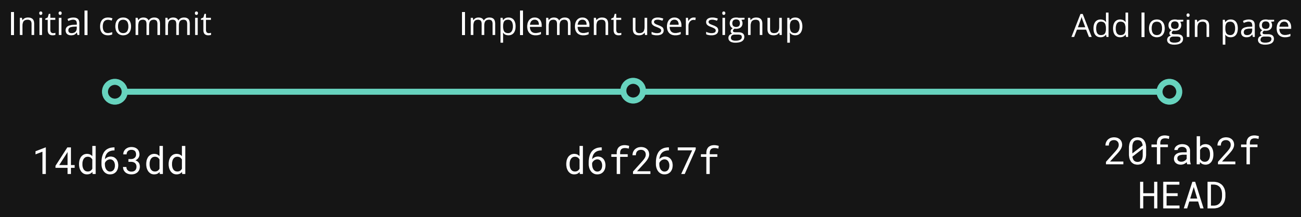
Diffing Commits

Viewing the changes between commits

```
$ git diff 20fab2f d6f267f
$ git diff HEAD d6f267f
$ git diff HEAD HEAD^
```

Use SHAs or names to refer to commits

The caret refers to the previous commit



Remotes

Other places where you keep your history

Setting up an SSH Key

Allows you to avoid entering your password

```
$ ssh-keygen -t rsa -b 4096 -C "your@email.com"

$ cat ~/.ssh/id_rsa.pub
```

Add this on GitHub’s website

Adding a Remote

A convenient nickname for this remote

```
$ git remote add origin git@github.com:your-username/your-project.git
```

Get this from GitHub's website after creating the repo

Pushing to a Remote

Uploading the history to a remote

```
$ git push origin master
```

The remote to which to push

More File Operations

```
$ git rm src/old_code.py
$ git rm --cached src/old_code.py
$ git mv src/tree.py src/dag.py
```

Untracks the file in git, but does not delete it from your working directory

Undoing Mistakes

Unstages any staged changes

```
$ git reset HEAD bloom_filter.py
$ git checkout test_bloom_filter.py
```

Restores file to its state in last commit (undoes any changes in the working directory)

Forgetting to Stage Files

```
$ git add forgotten_file.cpp
$ git commit --amend
```

Appends staged changes to the last commit

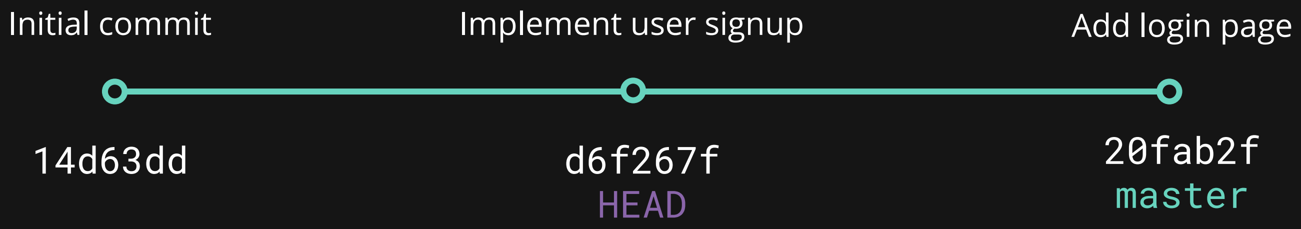
Git More Advanced

Time Traveling

```
$ git checkout d6f267f
$ git status
HEAD detached at d6f267f
$ git checkout master
```

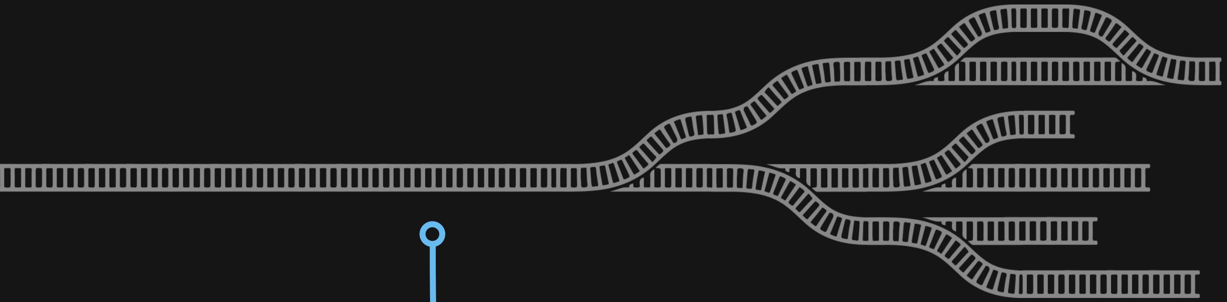
View the repository's state at a certain commit

Get back to the latest commit



Branches

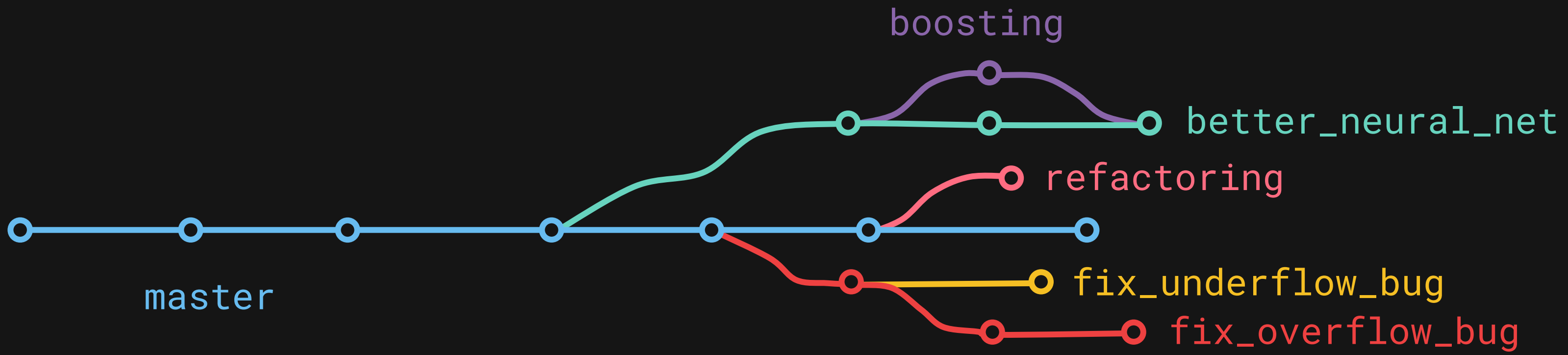
Lines of commits that occur sequentially in the history

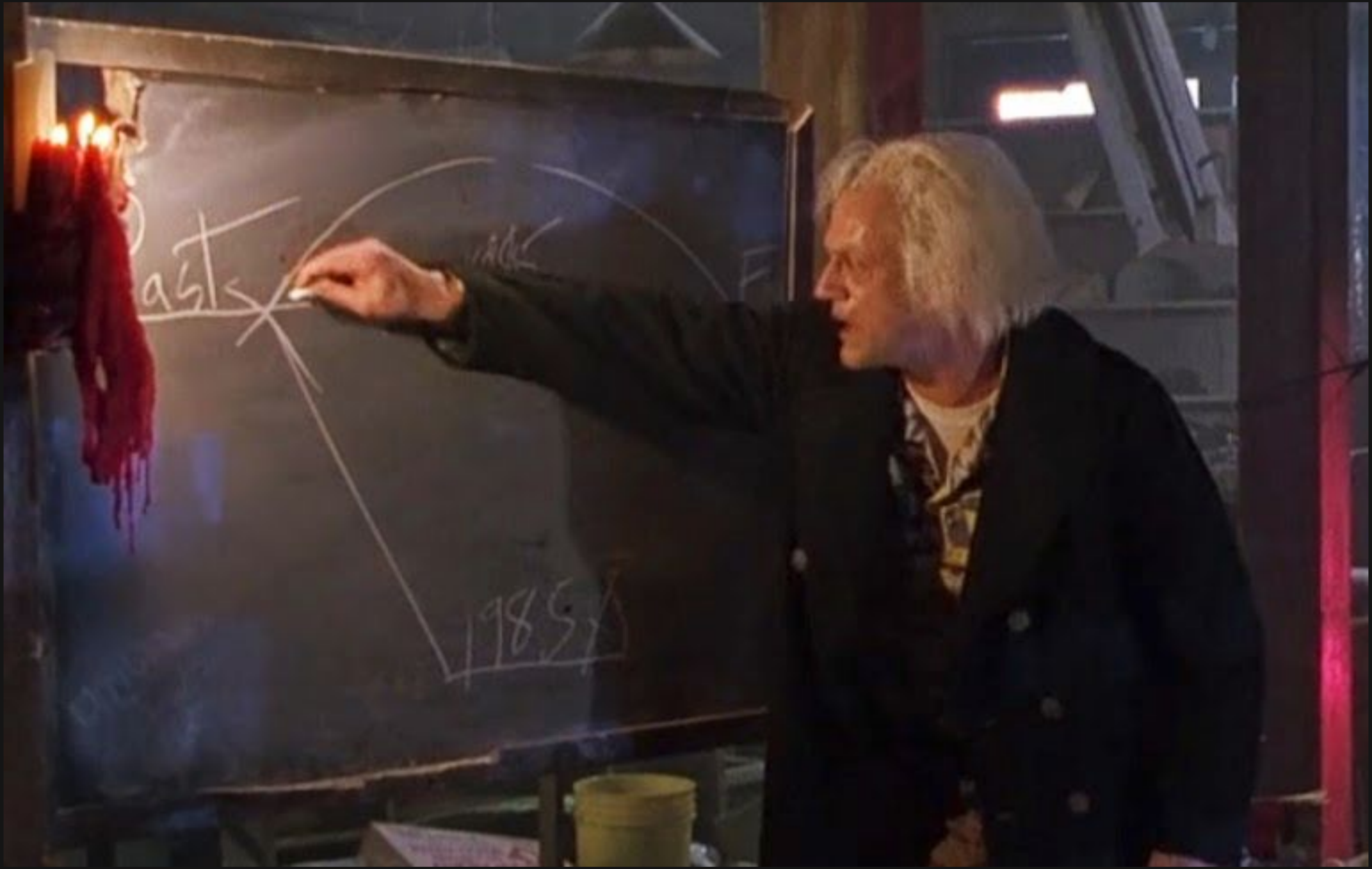


The main branch is called master

Branches

Lines of commits that occur sequentially in the history



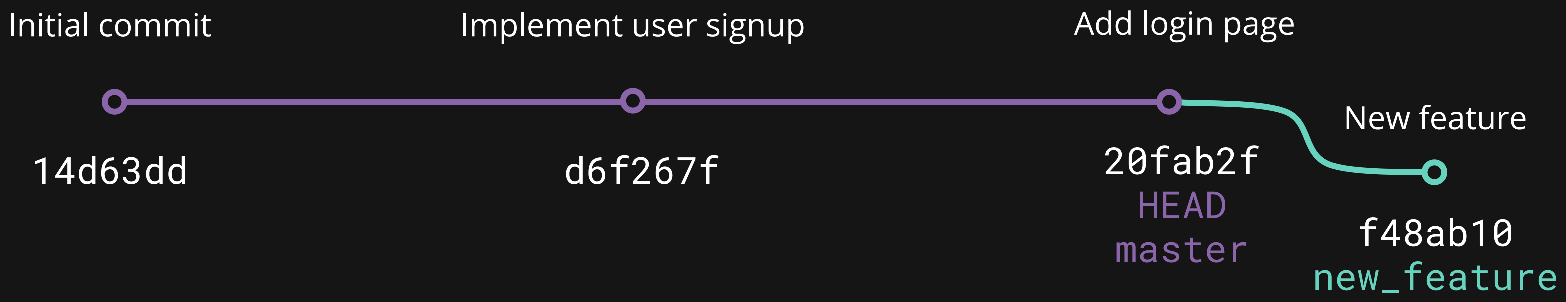

Pasts
1885

Creating Branches

```
$ git checkout -b new_features
Switched to a new branch 'new_feature'
$ git commit -m "New feature"
$ git checkout master
```

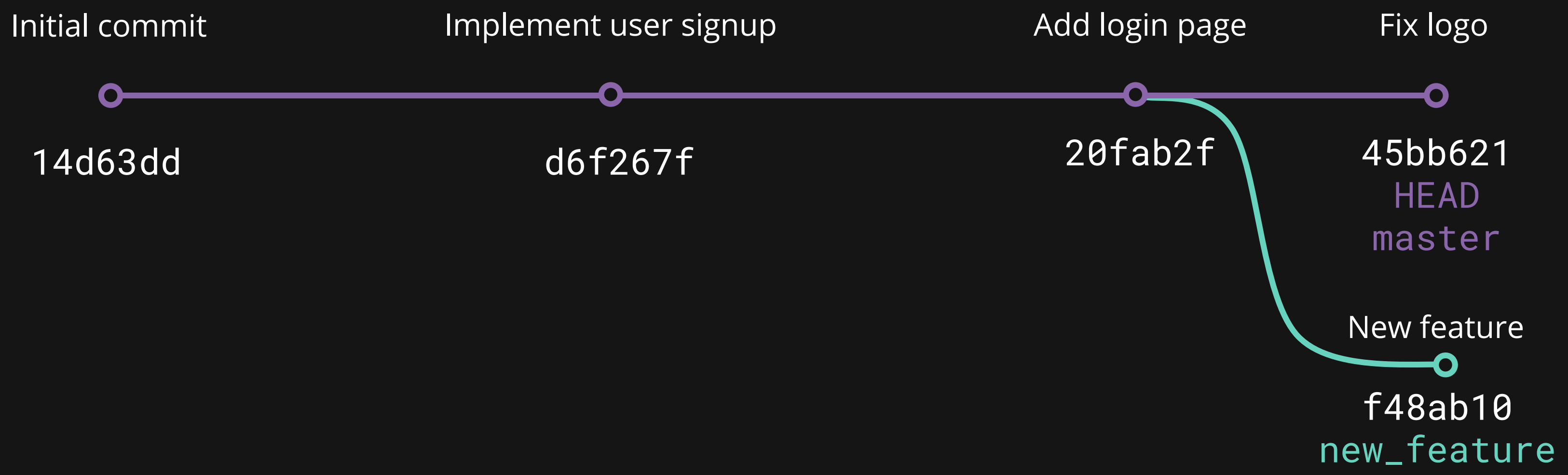
Create a new branch at HEAD

Return back to master



Diverging Branches

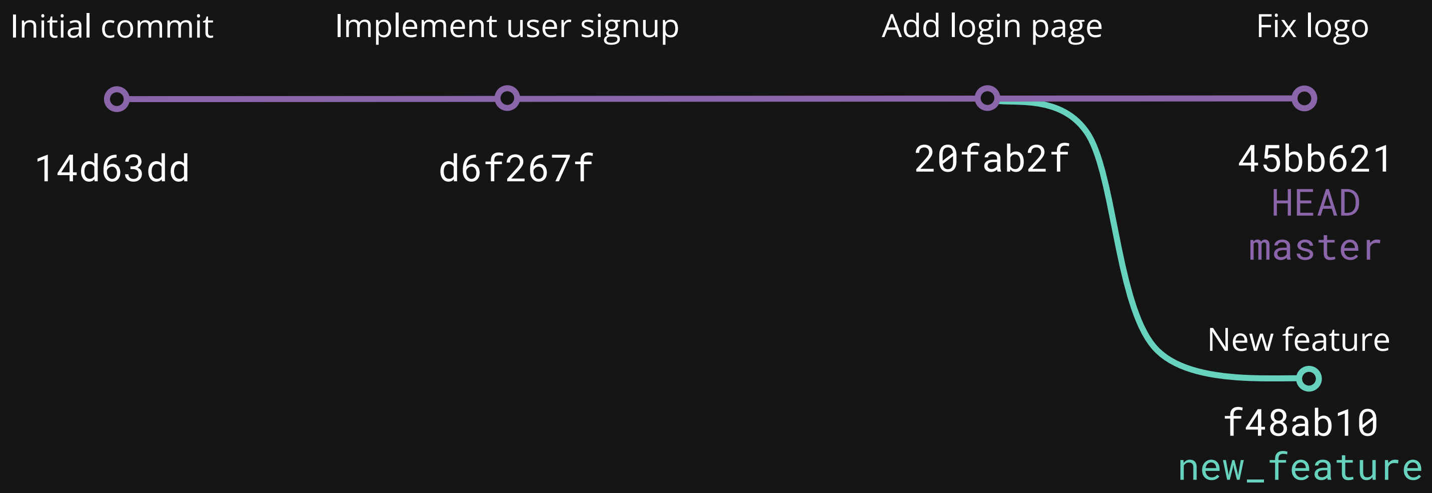
```
# ...
$ git commit -m "Fix logo"
```



Rebase to the Rescue

```
$ git checkout new_feature
$ git rebase master
```

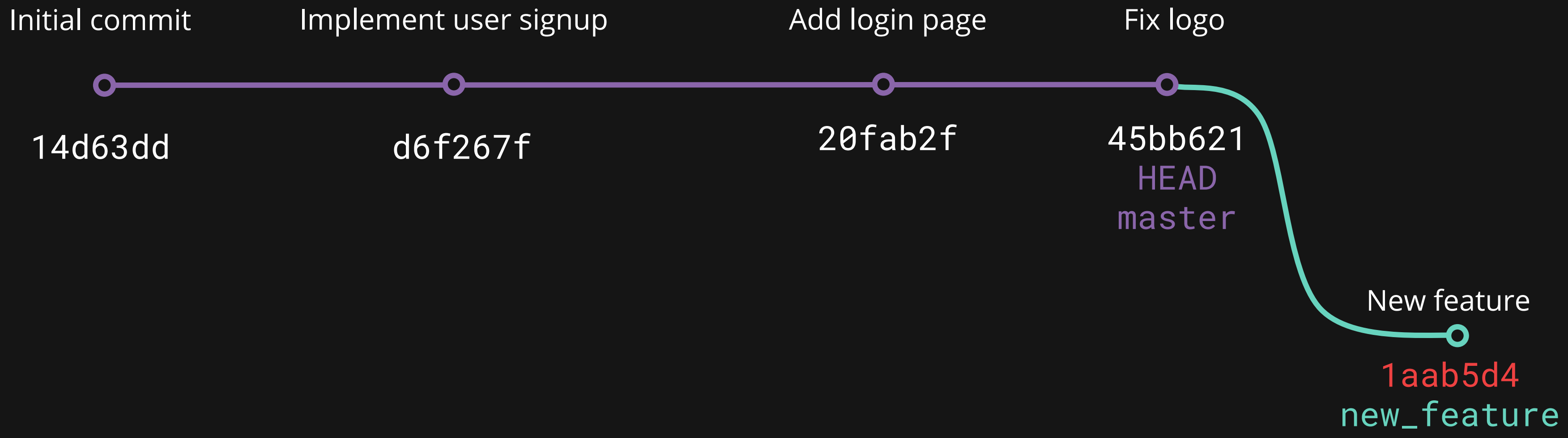
Rebase rewrites history by inserting all commits the current branch doesn't have that the target branch does



Rebase to the Rescue

```
$ git checkout new_feature
$ git rebase master
```

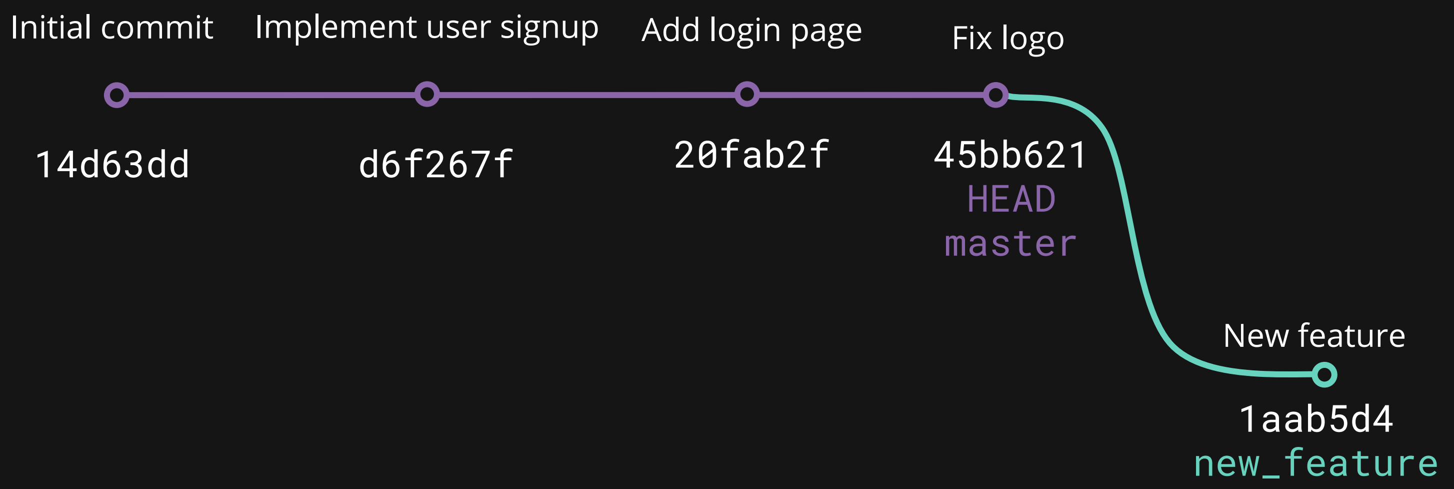
Rebase rewrites history by inserting all commits the current branch doesn't have that the target branch does



Merging back into master

```
$ git checkout master
$ git merge new_feature
```

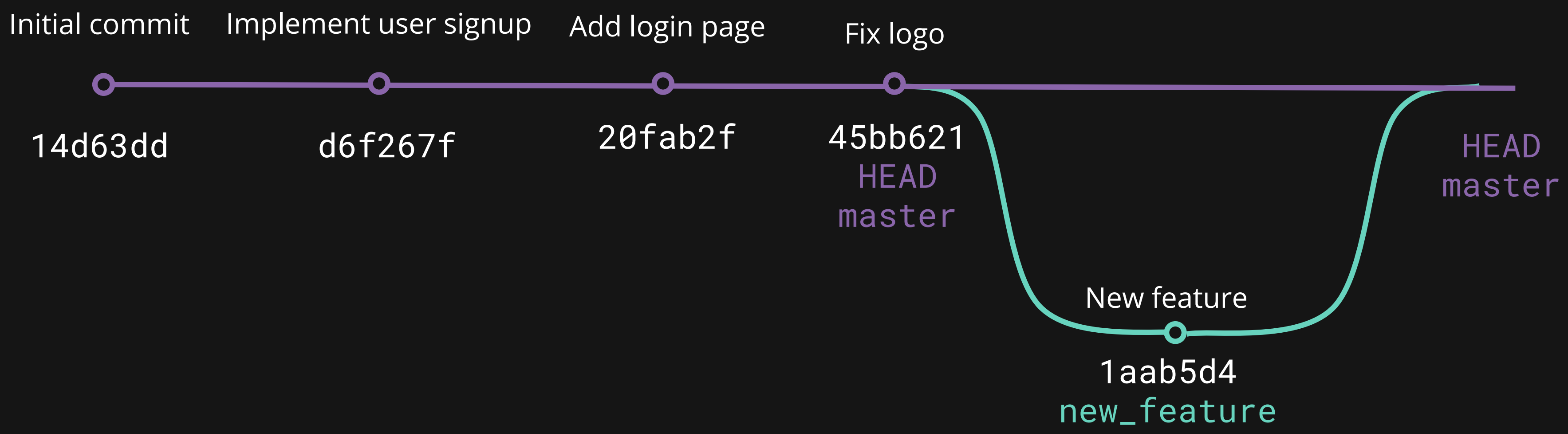
Merge will append all commits the current branch doesn't have that the target branch does



Merging back into master

```
$ git checkout master
$ git merge new_feature
```

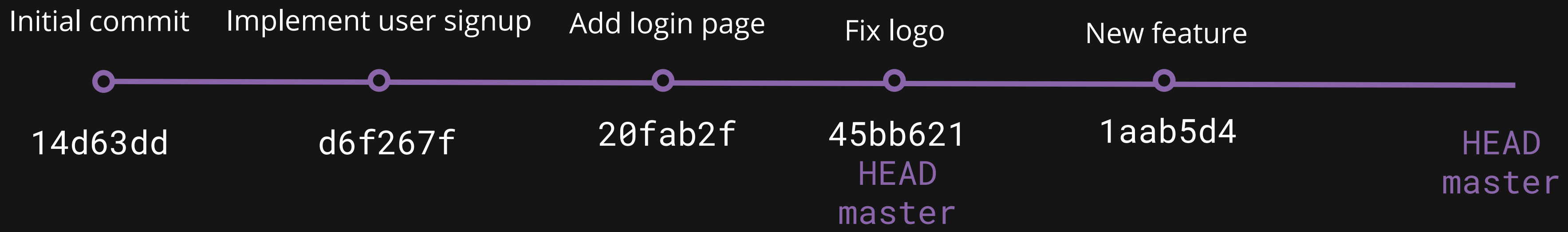
Merge will append all commits the current branch doesn't have that the target branch does



Merging back into master

```
$ git checkout master
$ git merge new_feature
```

Merge will append all commits the current branch doesn't have that the target branch does



Pulling from a remote

```
$ git checkout master
$ git pull --rebase origin master
```

Pull will prepend all new commits from the remote branch to the current branch

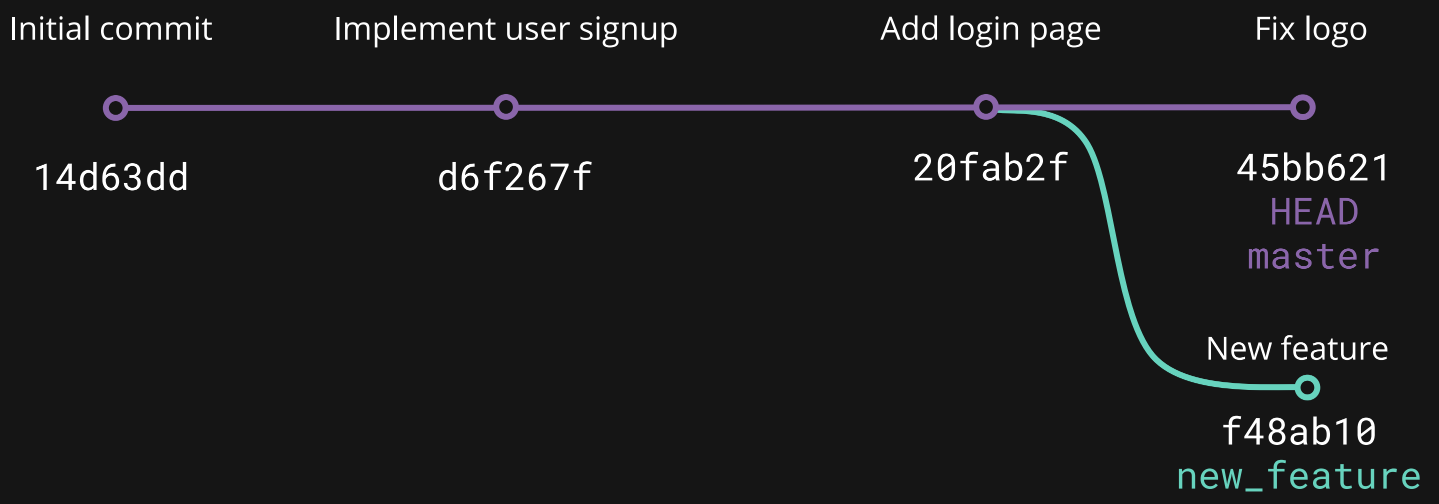
Undoing Mistakes

Hard reset will erase the last commit and all its changes from the current branch

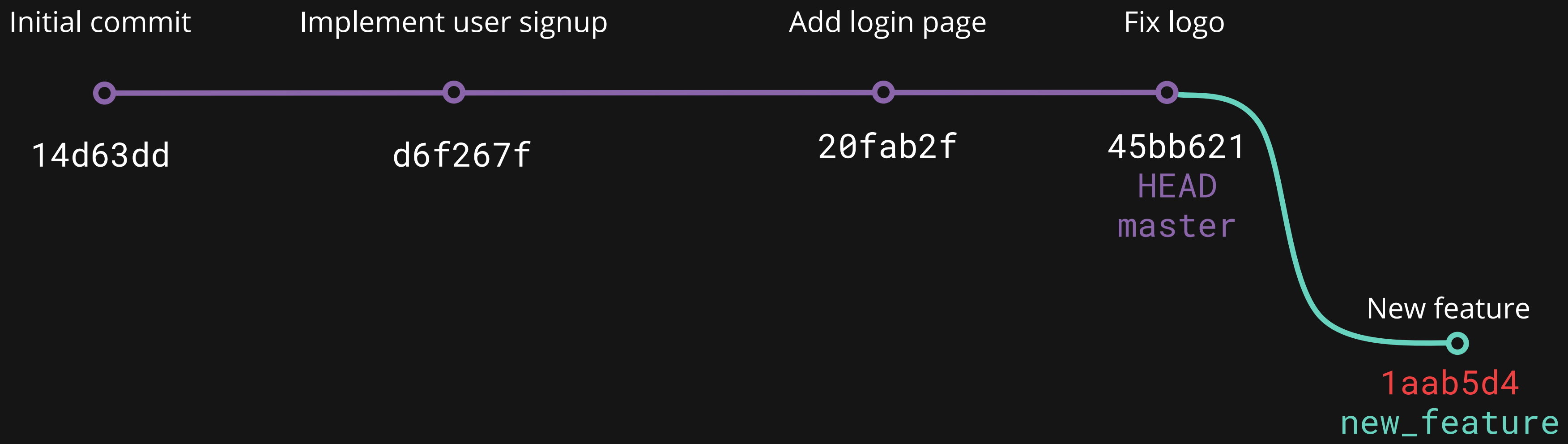
```
$ git reset --hard HEAD^

$ git reset --soft HEAD^
```

Soft reset will remove the last commit from the current branch, but keep all changes staged

Merge Conflicts



Merge Conflicts

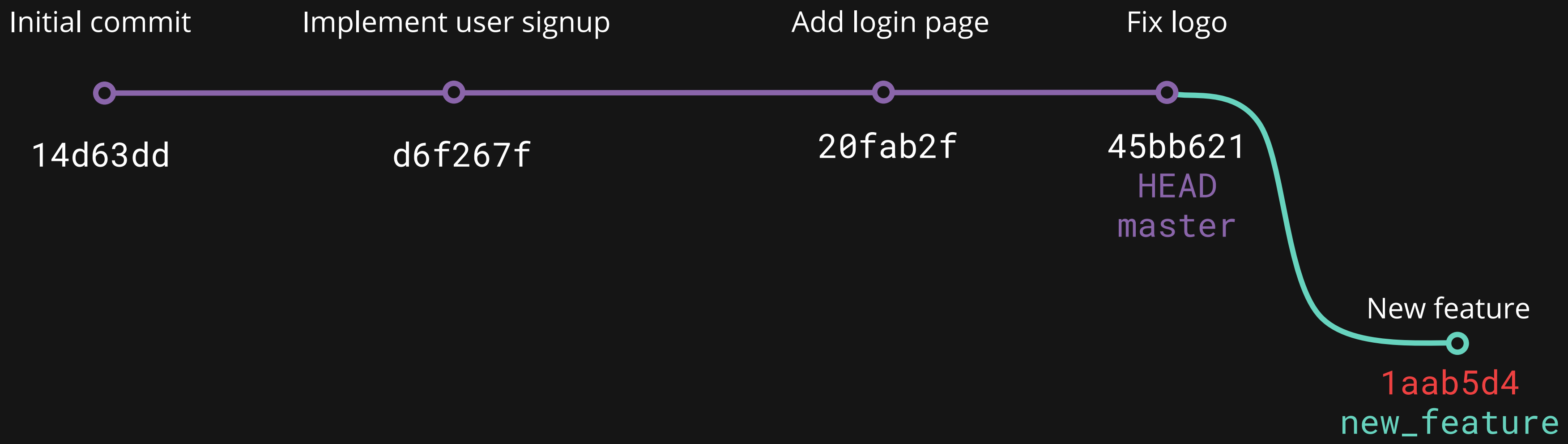


What if both branches have changes to nearby lines of code?

Rebase early, rebase often

Rebase first, then merge (fast-forward)

Merge Conflicts



Pull Request Workflow

1. Create issue (optional)
2. Fork and make a new branch
3. Commit changes to this branch
4. Push this branch
5. Submit a pull request on Github/Gitlab/Bitbucket

Lightning Round

gh-pages
Tags
Add partial
Stash
Blame
Revert
Bisect
Cherry pick