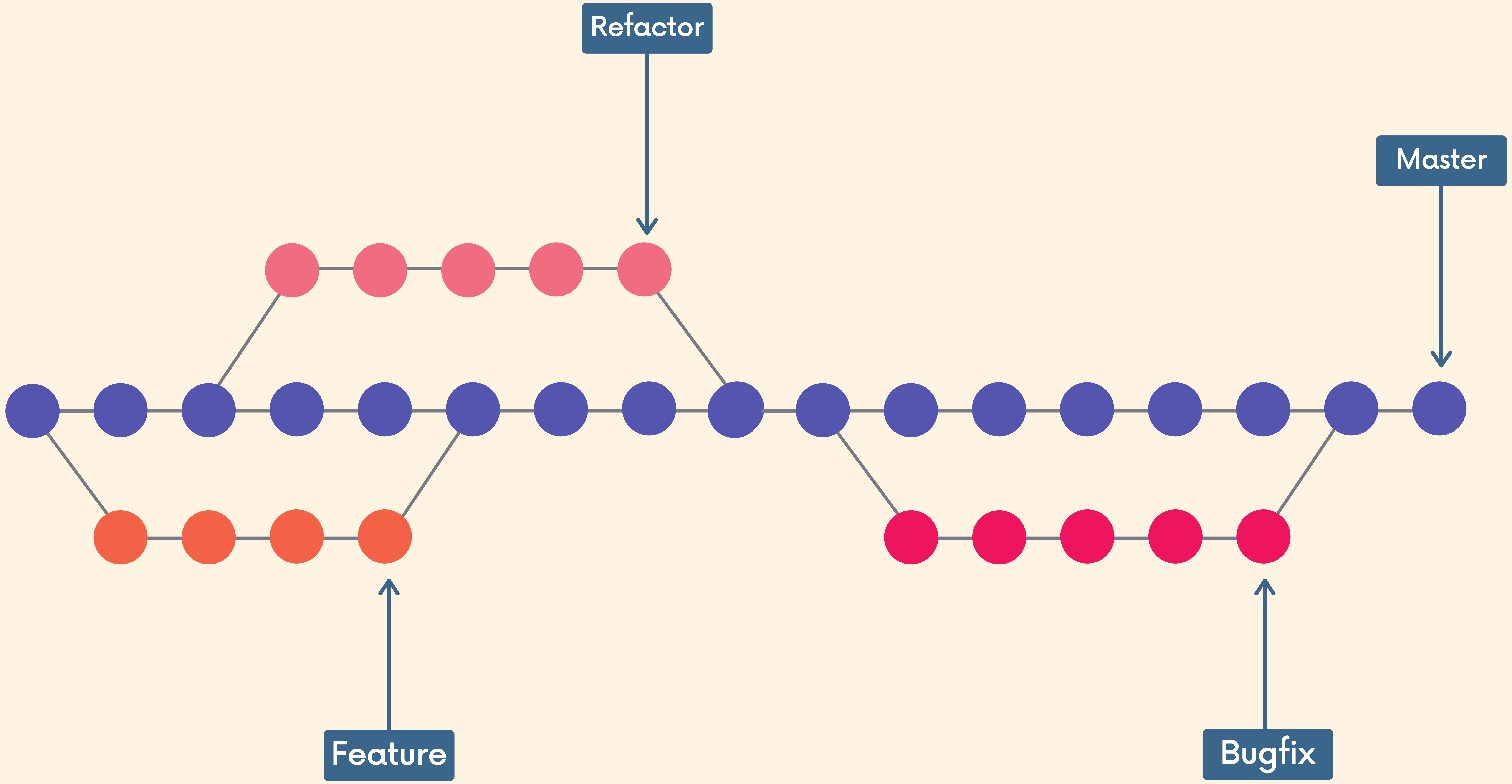
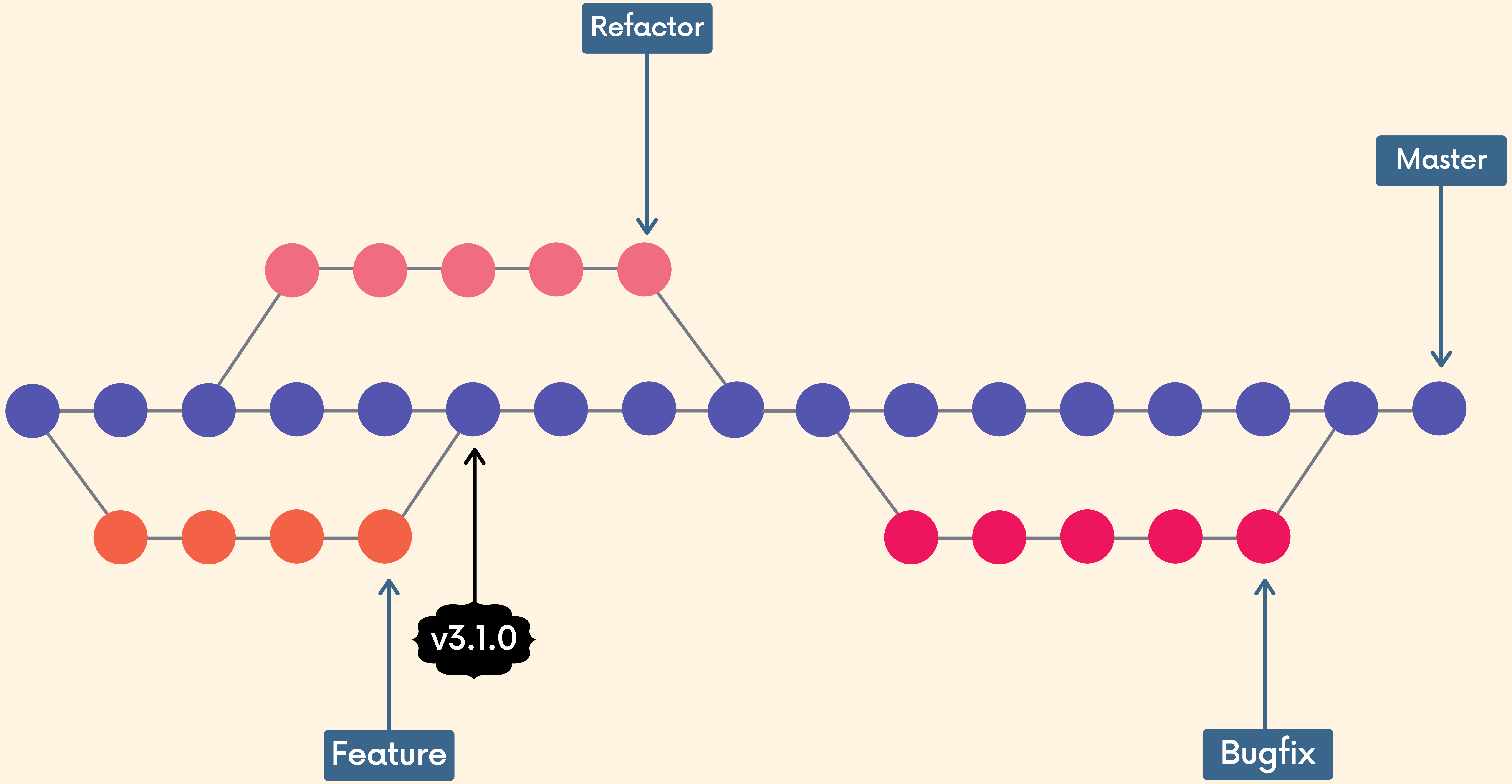
Git Tags

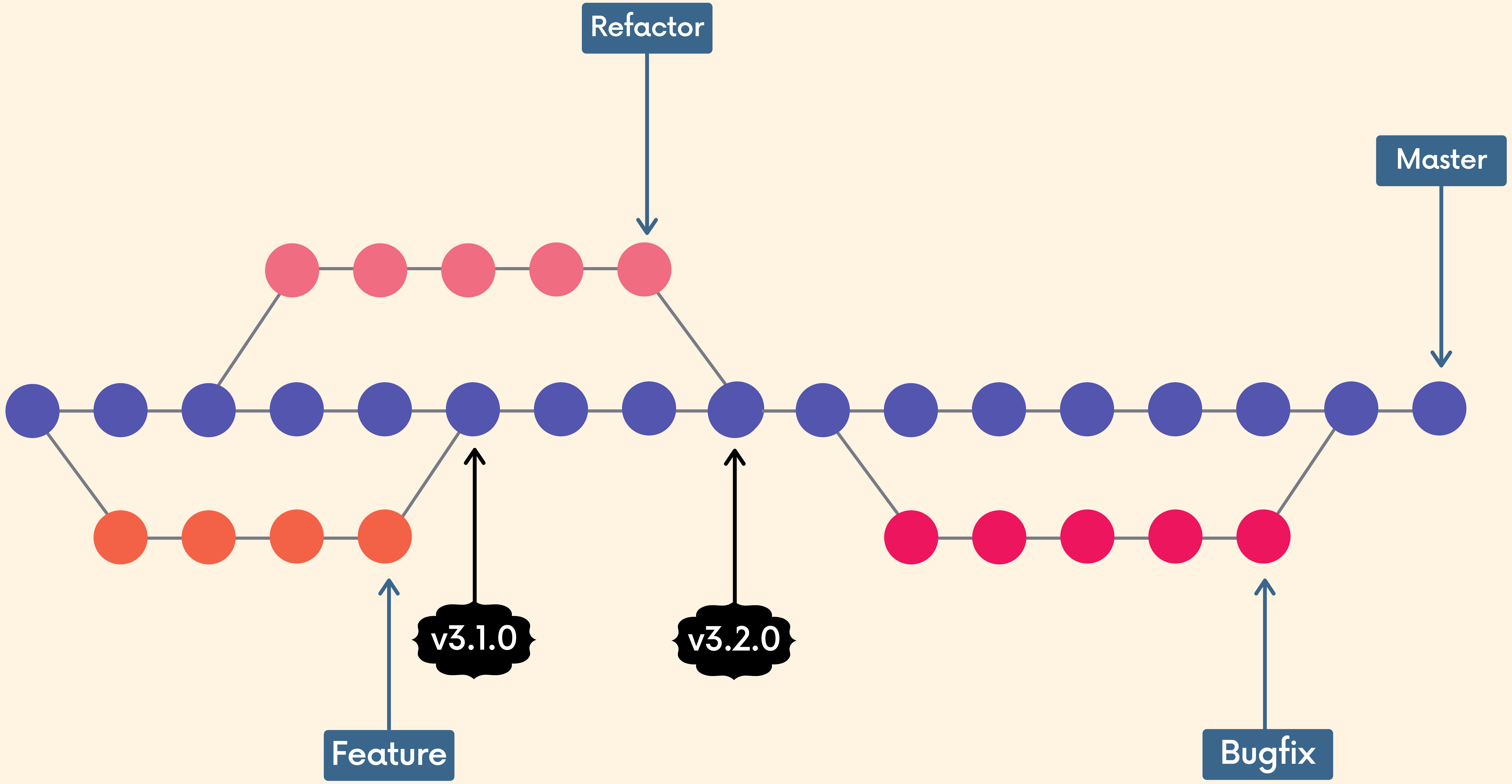
Git Tags

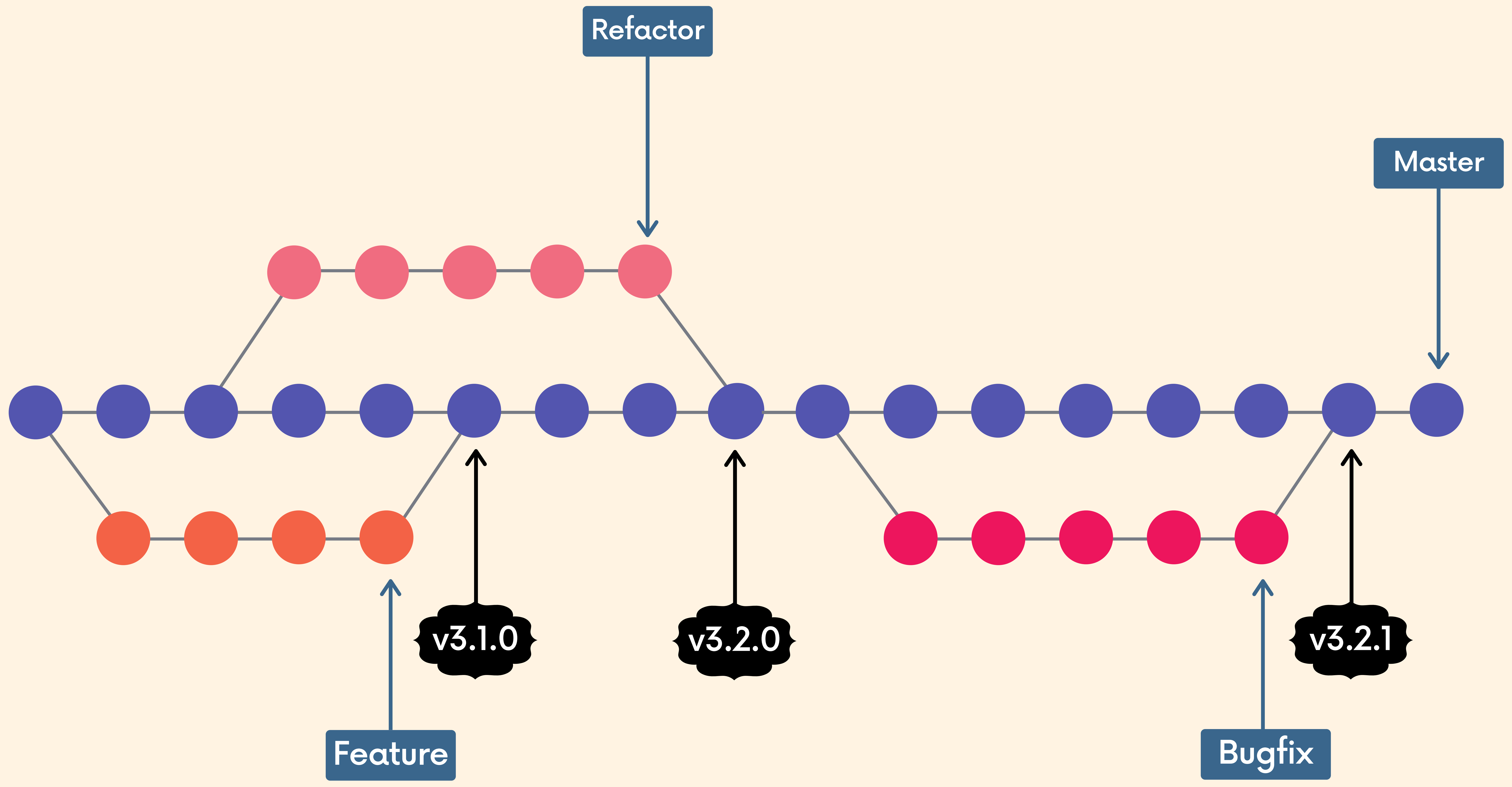
Tags are pointers that refer to particular points in Git history. We can mark a particular moment in time with a tag. Tags are most often used to mark version releases in projects (v4.1.0, v4.1.1, etc.)

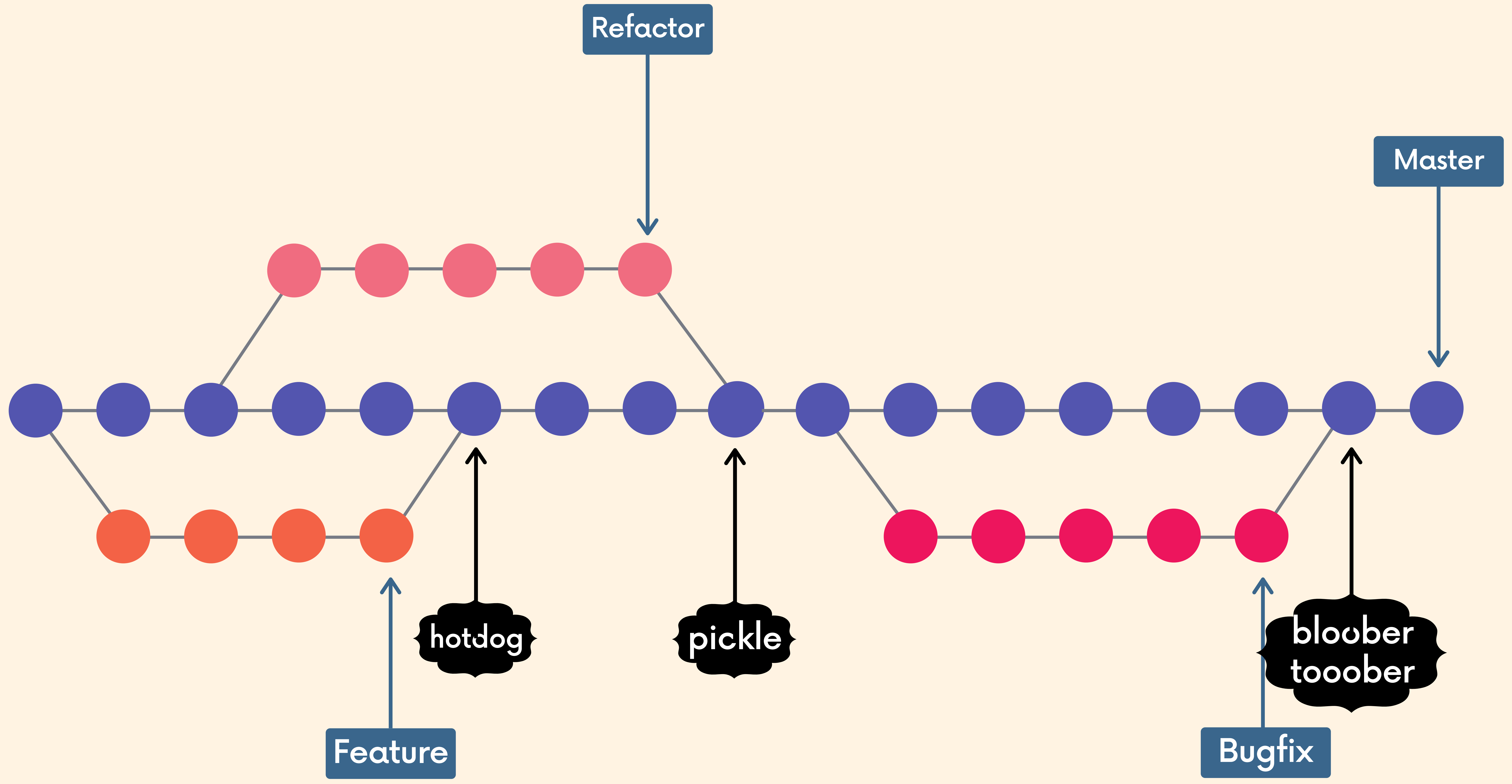
Think of tags as branch references that do NOT CHANGE. Once a tag is created, it always refers to the same commit. It's just a label for a commit.


Refactor
Master
Feature
Bugfix


Refactor
Master
v3.1.0
Feature
Bugfix


Refactor
Master
v3.1.0
v3.2.0
Feature
Bugfix


Refactor
Master
v3.1.0
v3.2.0
v3.2.1
Feature
Bugfix

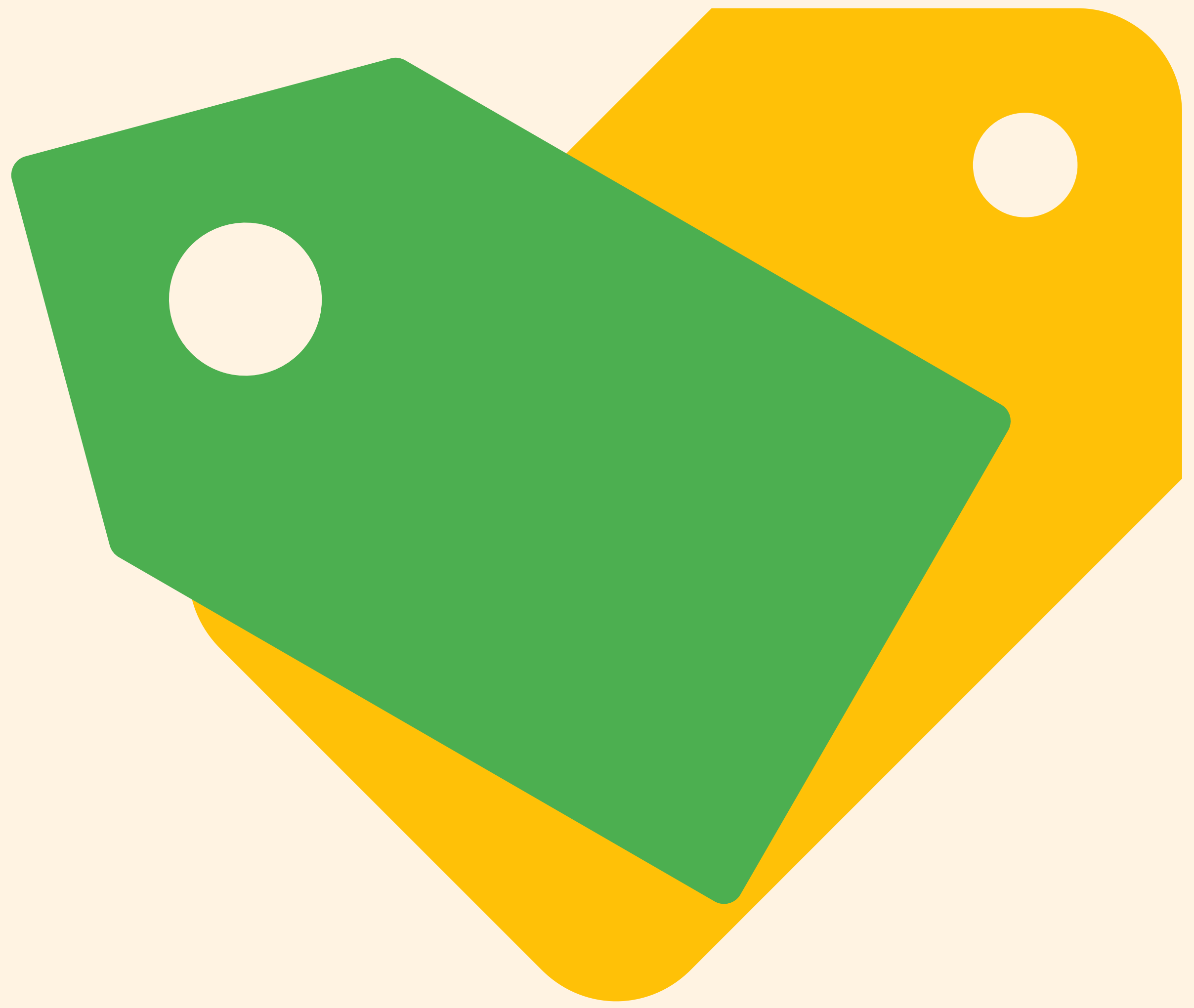

Refactor
Master
hotdog
pickle
bloober tooober
Feature
Bugfix

The Two Types

There are two types of Git tags we can use: lightweight and annotated tags

lightweight tags are...lightweight. They are just a name/label that points to a particular commit.

annotated tags store extra meta data including the author's name and email, the date, and a tagging message (like a commit message)



Semantic Versioning

The semantic versioning spec outlines a standardized versioning system for software releases. It provides a consistent way for developers to give meaning to their software releases (how big of a change is this release??)

Versions consist of three numbers separated by periods.

2.4.1

2.4.1

MAJOR RELEASE

MINOR RELEASE

PATCH RELEASE

Initial Release

Typically, the first release is 1.0.0

1.0.0

Patch Release

Patch releases normally do not contain new features or significant changes. They typically signify bug fixes and other changes that do not impact how the code is used

1.0.1

Minor Release

Minor releases signify that new features or functionality have been added, but the project is still backwards compatible. No breaking changes. The new functionality is optional and should not force users to rewrite their own code.

1.1.0

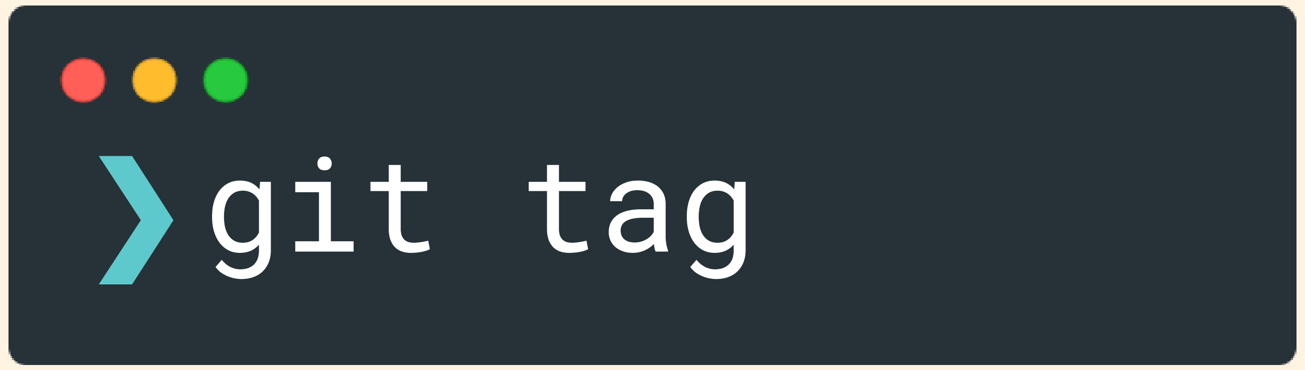
Major Release

Major releases signify significant changes that is no longer backwards compatible. Features may be removed or changed substantially.

2.0.0

Viewing Tags

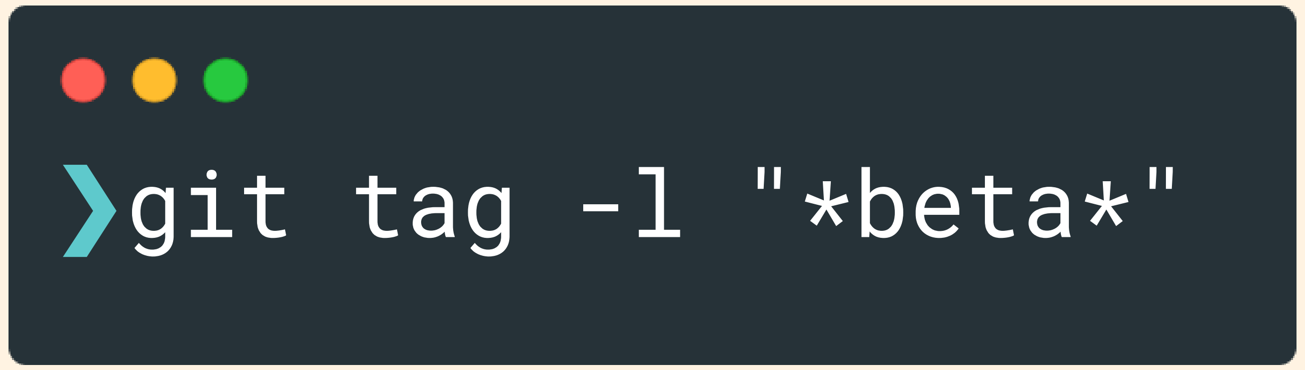
git tag will print a list of all the tags in the current repository.



```
❯git tag
```

Viewing Tags

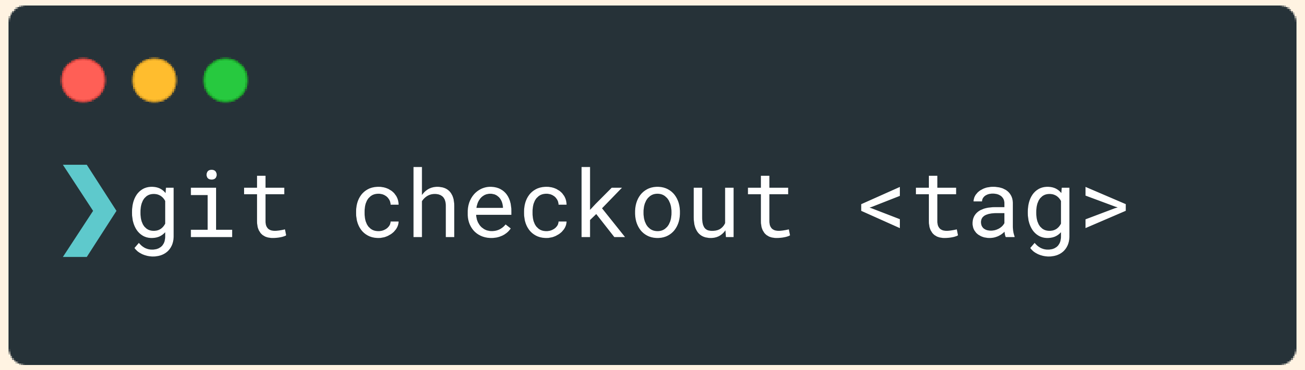
We can search for tags that match a particular pattern by using **git tag -l** and then passing in a wildcard pattern. For example, **git tag -l "*beta*"** will print a list of tags that include "beta" in their name.



```
❯git tag -l "*beta*"
```

Checking Out Tags

To view the state of a repo at a particular tag, we can use **git checkout <tag>**. This puts us in detached HEAD!



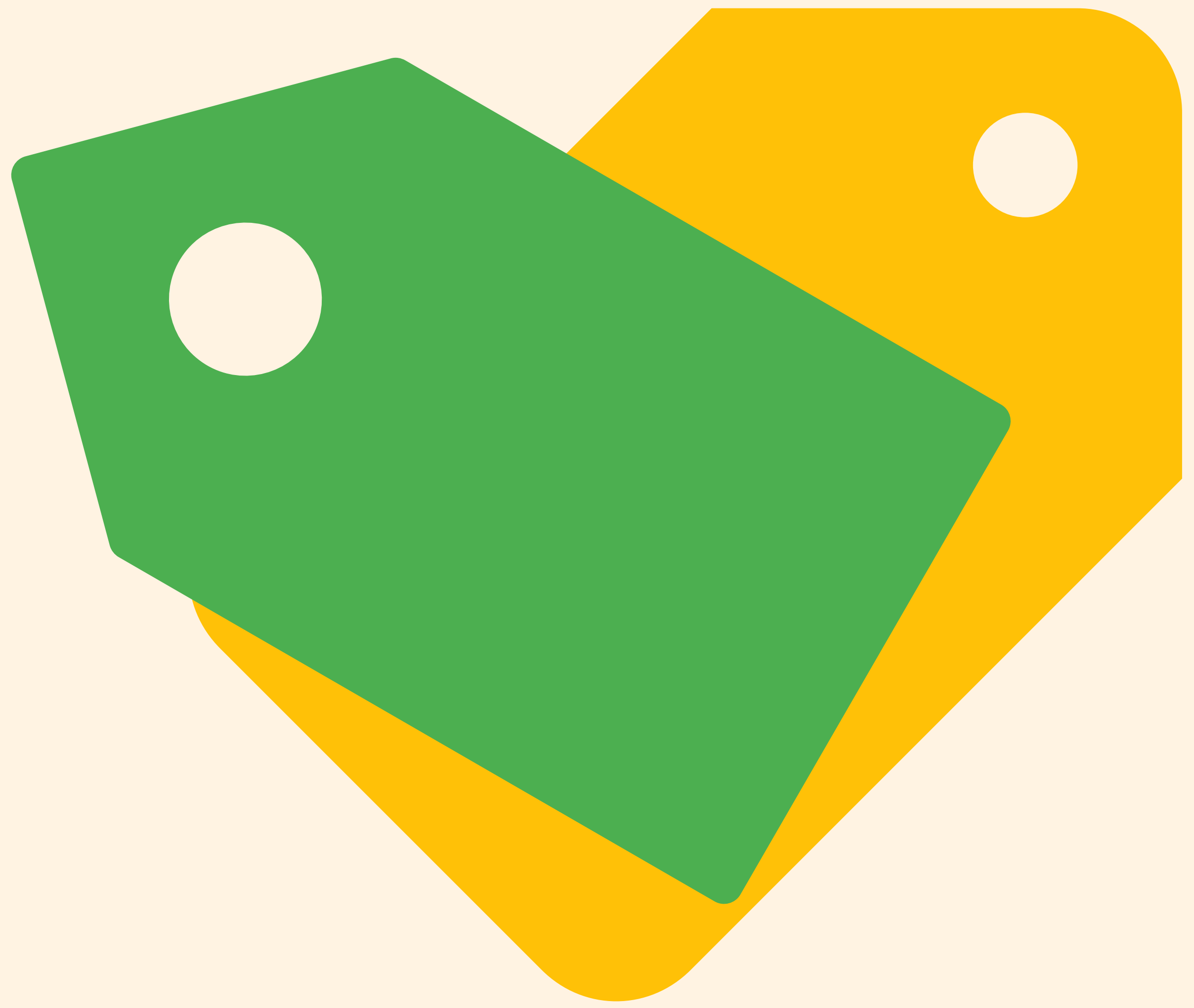
```
❯git checkout <tag>
```

The Two Types

There are two types of Git tags we can use: lightweight and annotated tags

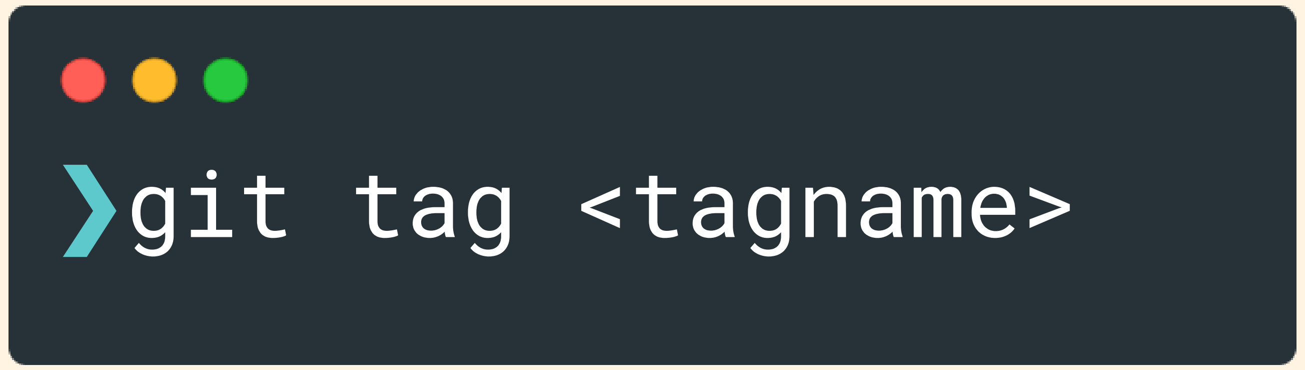
lightweight tags are...lightweight. They are just a name/label that points to a particular commit.

annotated tags store extra meta data including the author's name and email, the date, and a tagging message (like a commit message)



Creating Lightweight Tags

To create a lightweight tag, use **git tag <tagname>**
By default, Git will create the tag referring to the commit that HEAD is referencing.

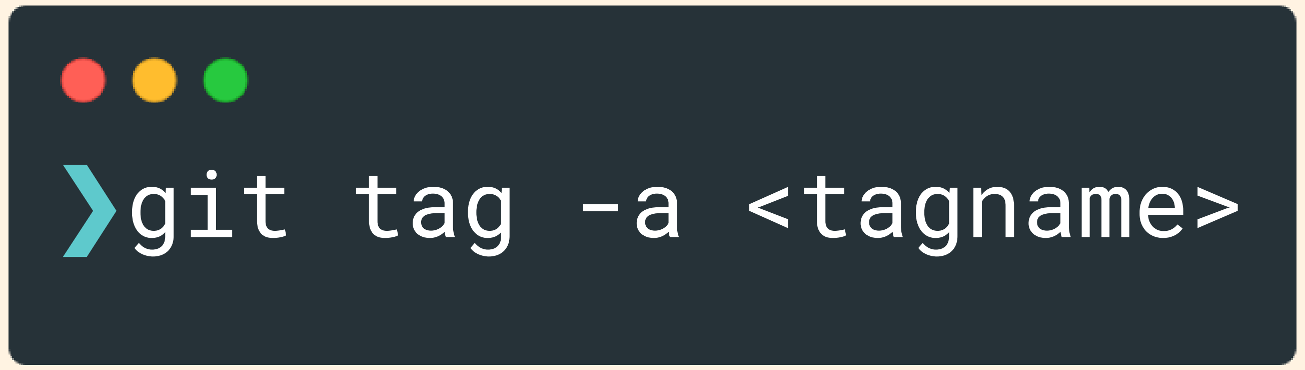


```
❯git tag <tagname>
```

Annotated Tags

Use **git tag -a** to create a new annotated tag. Git will then open your default text editor and prompt you for additional information.

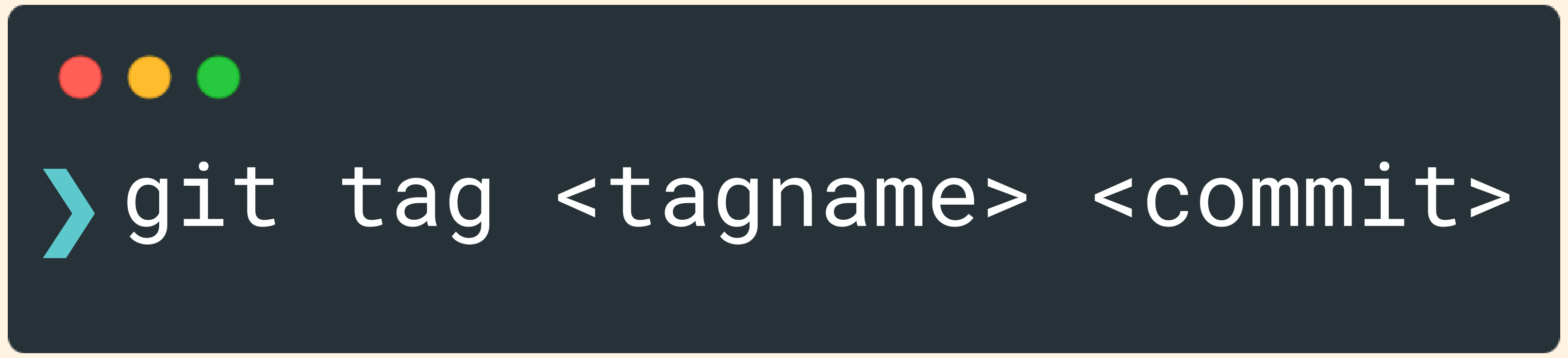
Similar to git commit, we can also use the **-m** option to pass a message directly and forgo the opening of the text editor



```
❯git tag -a <tagname>
```

Tagging Previous Commits

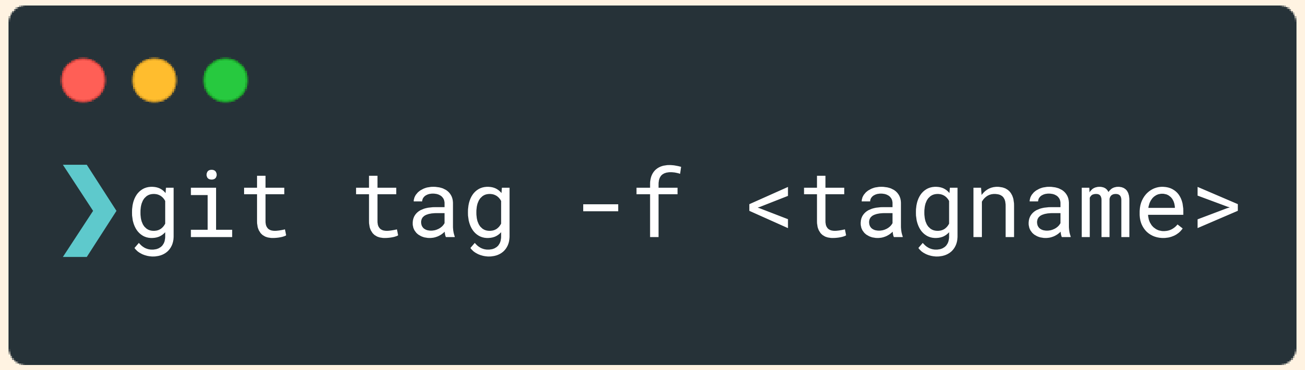
So far we've seen how to tag the commit that HEAD references. We can also tag an older commit by providing the commit hash: **git tag -a <tagname> <commit-hash>**



```
❯ git tag <tagname> <commit>
```

Forcing Tags

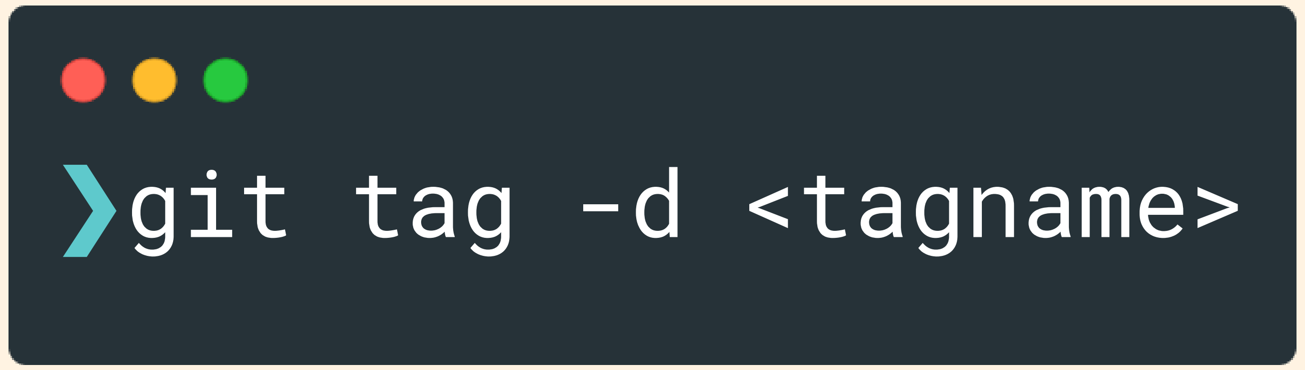
Git will yell at us if we try to reuse a tag that is already referring to a commit. If we use the -f option, we can FORCE our tag through.



```
❯git tag -f <tagname>
```

Deleting Tags

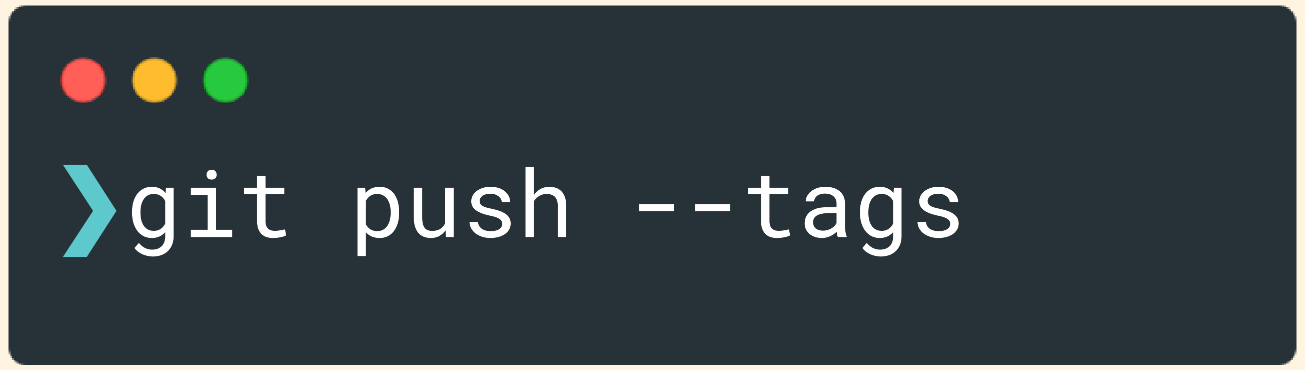
To delete a tag, use **git tag -d <tagname>**



```
❯git tag -d <tagname>
```

Pushing Tags

By default, the **git push** command doesn't transfer tags to remote servers. If you have a lot of tags that you want to push up at once, you can use the --tags option to the **git push** command. This will transfer all of your tags to the remote server that are not already there.



```
❯git push --tags
```