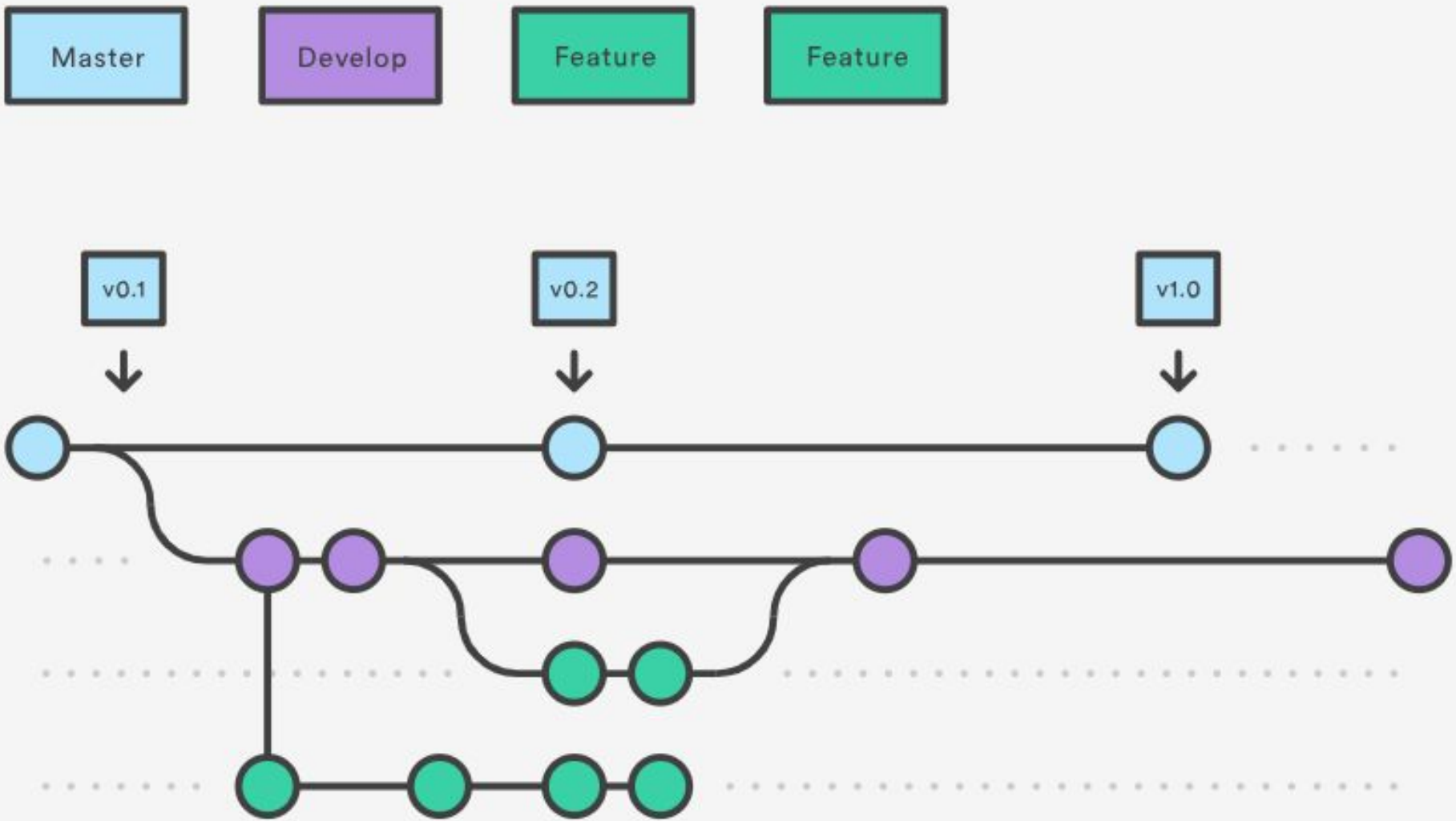
Git merges

Prakhar Tripathi


Master
Develop
Feature
Feature
v0.1
v0.2
v1.0

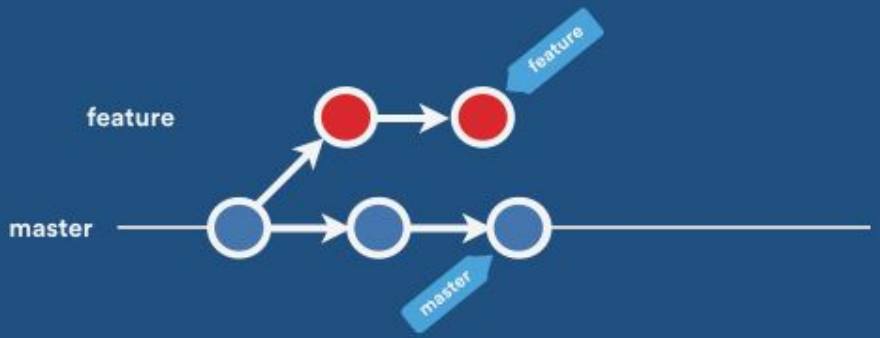
3 Common Strategies

1. Explicit, or non fast-forward merge (**--no-ff**)
2. Implicit - via rebase or fast-forward merge (**--ff-only**)
3. Squash on merge

Explicit merges (aka non fast-forward)

What is a merge?

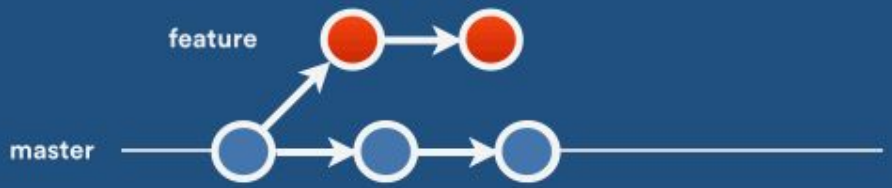
A process that unifies the work done in two branches



Implicit merges

What is a rebase?

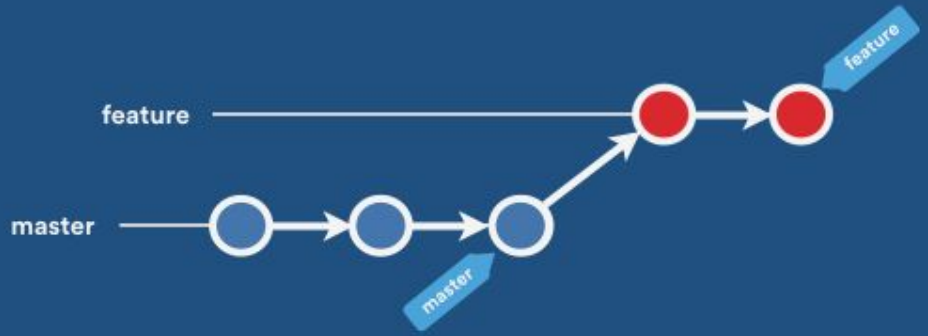
It's a way to replay commits,
one by one, on top of a branch



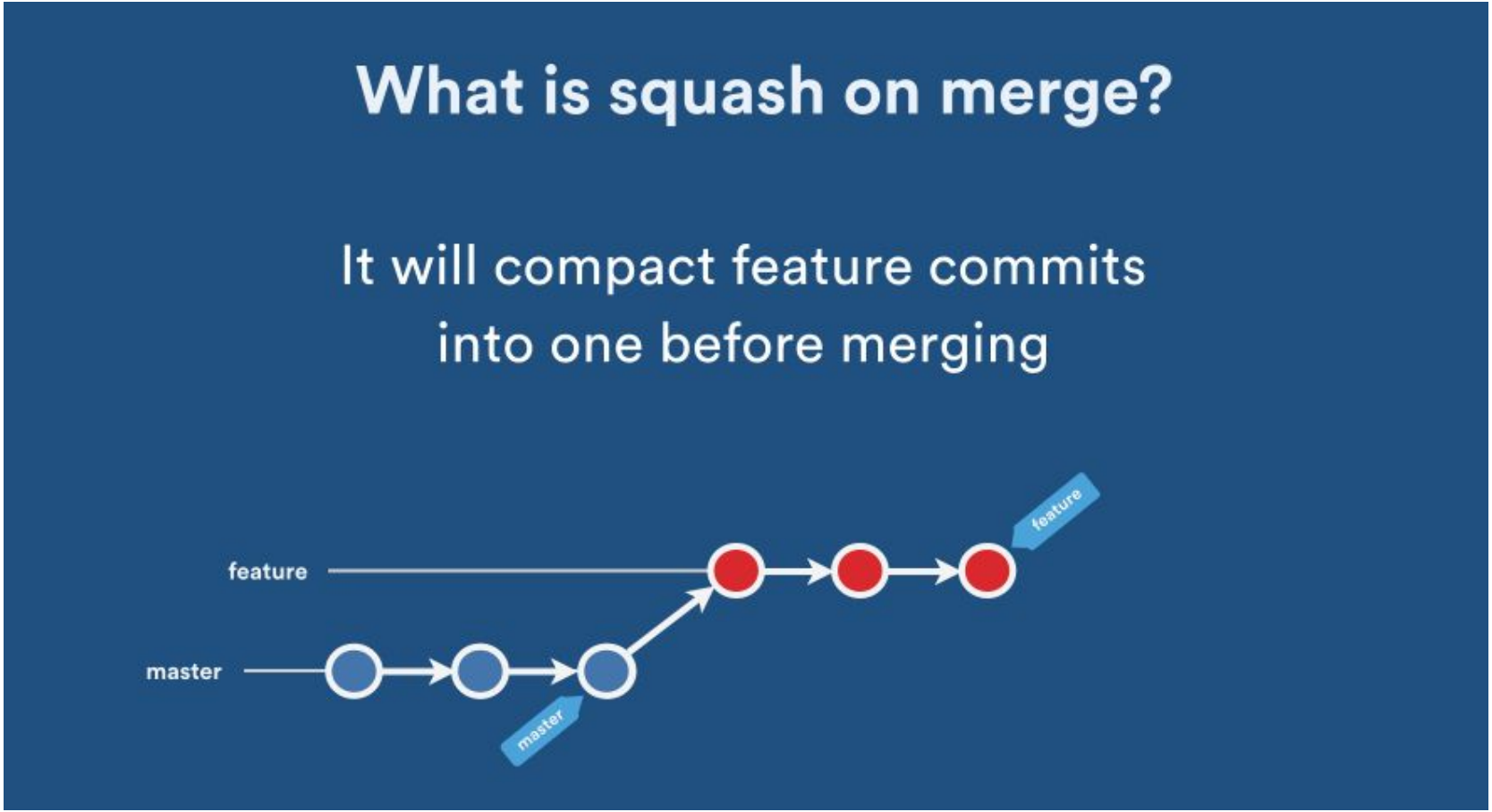
Implicit merges

What is a fast-forward merge?

It will just shift the
master HEAD

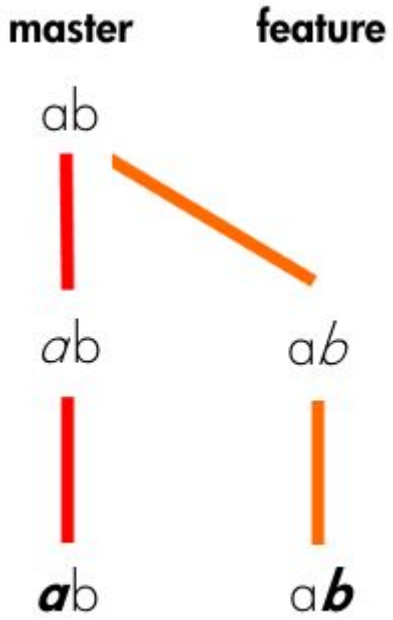


Squash on merge (generally without explicit)



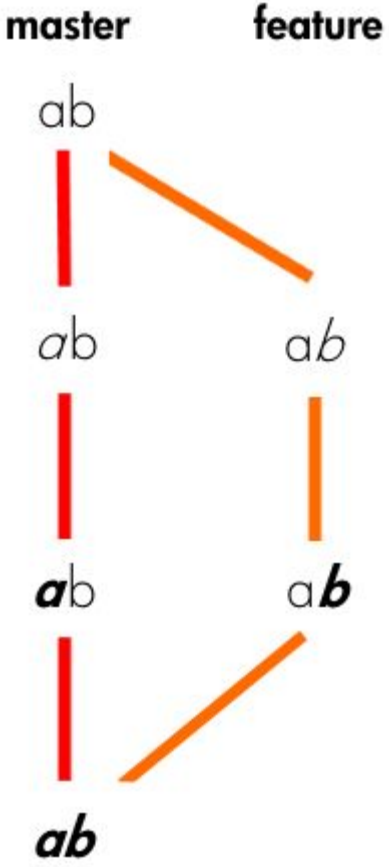
Example

ab



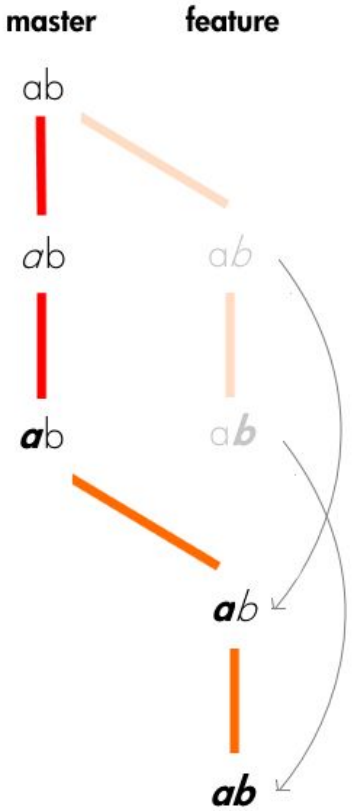
Master	Feature
Add the letters 'ab'	Add the letters 'ab'
Italicize the 'a'	Italicize the 'b'
Bold the 'a'	Bold the 'b'

Merge commit



Master	Feature
Add the letters 'ab'	Add the letters 'ab'
Italicize the 'a'	Italicize the 'b'
Bold the 'a'	Bold the 'b'
Italicize and bold the 'b'	

Rebase and merge



Golden rule of rebasing

Don’t rebase a branch unless you are the only one who uses it.

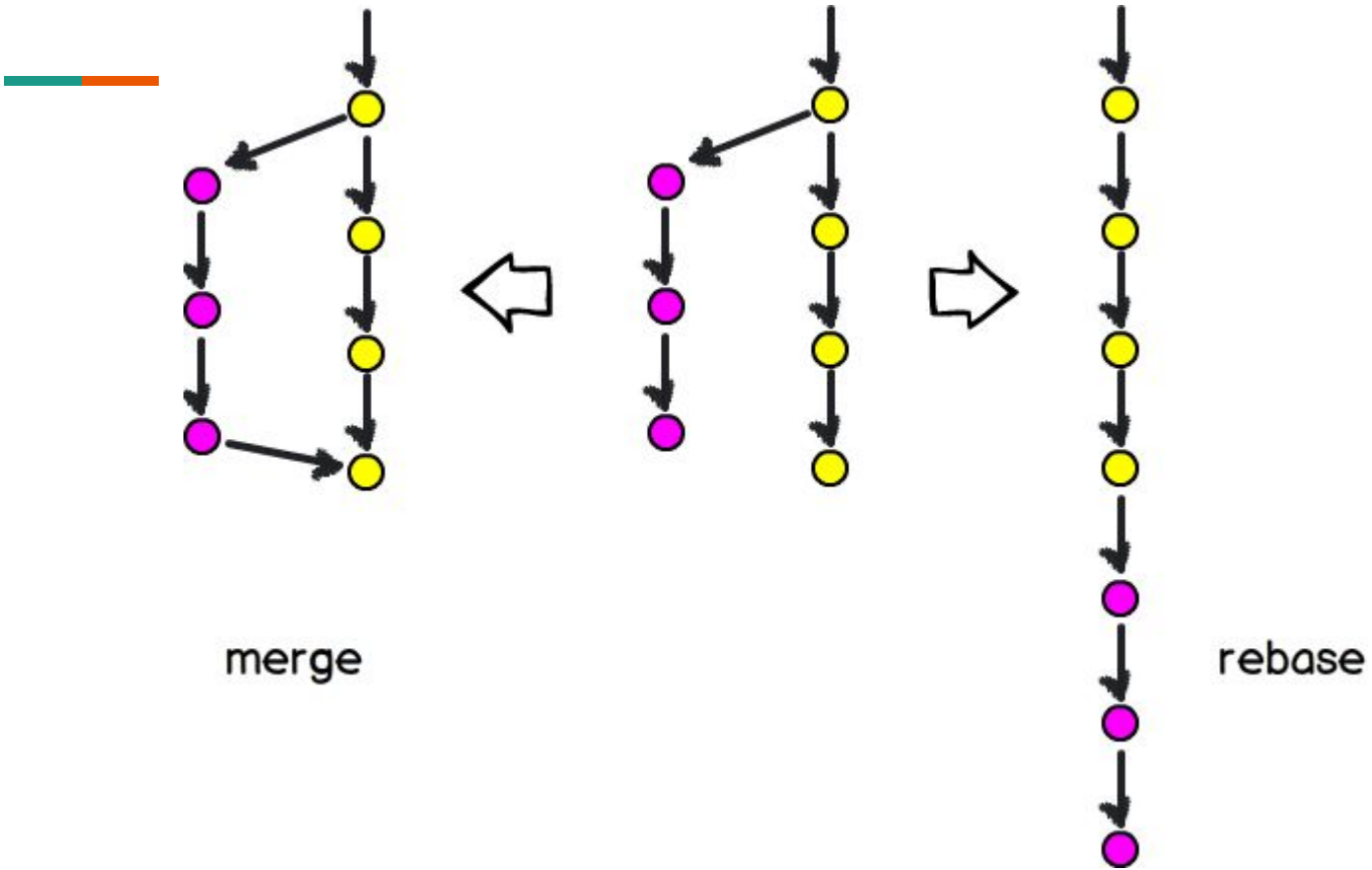
Master	Feature	Mark's copy of Feature
Add the letters 'ab'	Add the letters 'ab'	Add the letters 'ab'
Italicize the 'a'	**Italicize the 'a'**	Italicize the 'b'
Bold the 'a'	**Bold the 'a'**	Add the letter c
	Italicize the 'b'	
	Bold the 'b'	

How do you decide what to use?

- Git merge is the safest option
- **Do you value more a clean, linear history? Or the traceability of your branches?**

- Rebase - Linear, cleaner history

- Merge - Better traceability


merge
rebase

Summary

1. Use merge in cases where you want a set of commits to be clearly grouped together in history
2. Use rebase when you want to keep a linear commit history
3. DON’T use rebase on a public/shared branch

Bonus 1

Git Releases

Bonus 2

// start git bisect

git bisect start

// give git a commit where there is not a bug

git bisect good a09c728

// give git a commit where there is a bug

git bisect bad b6a0692

Thank you