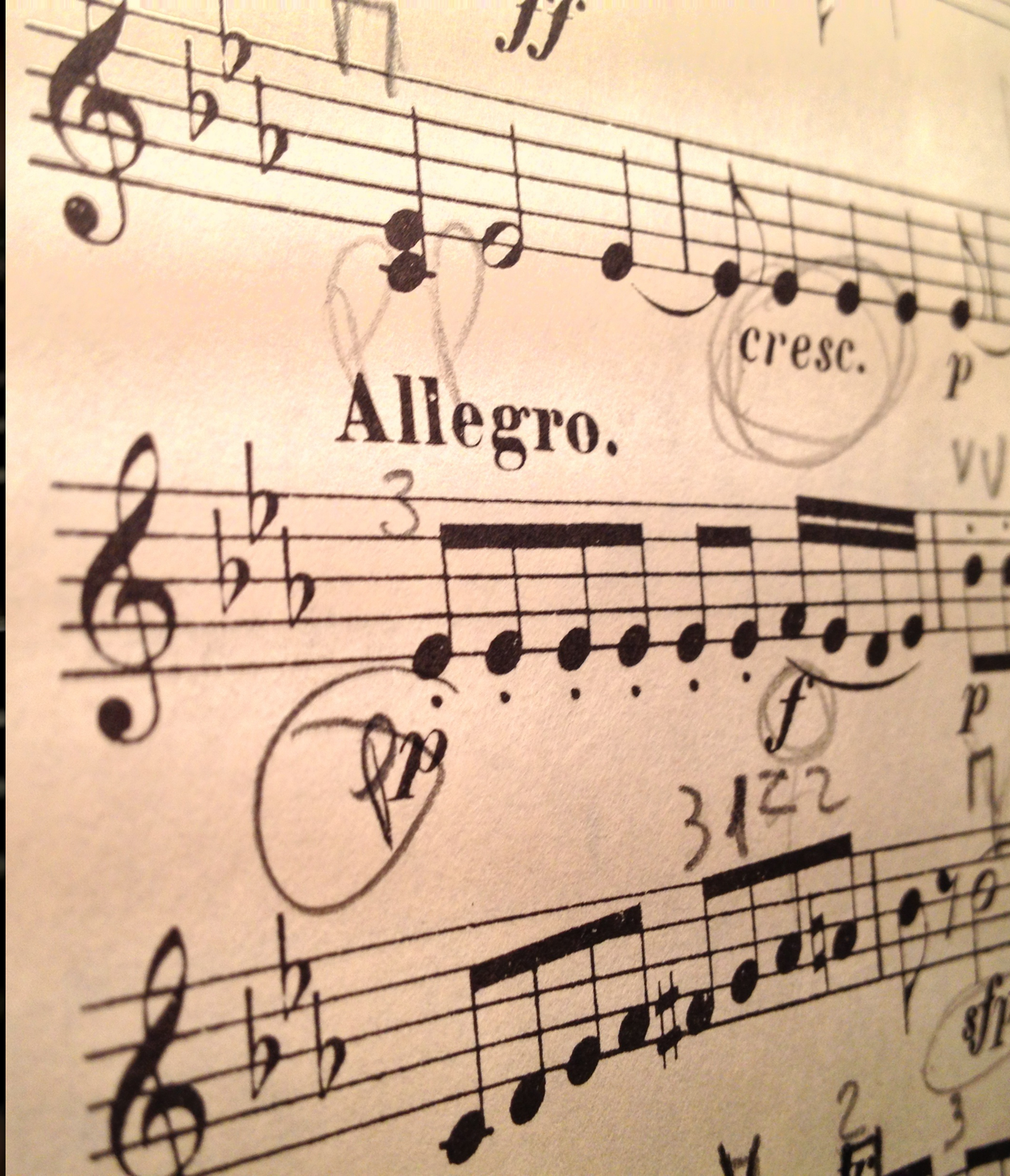
# How Effective Teams Use 


megakemp.com/git


megakemp.com/git

@ecampidoglio


ff
cresc.
p
Allegro.
3
pp
f
p
3122
sfz


HELP
ADVICE
SUPPORT
GUIDANCE


CVS


PERFORCE


Team Foundation
Server


SUBVERSION

## NAME

git - the stupid content tracker

## SYNOPSIS

```
git [--version] [--help] [-C <path>] [-c <name>=<value>]
    [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
    [-p|--paginate|--no-pager] [--no-replace-objects] [--bare]
    [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
    [--super-prefix=<path>]
    <command> [<args>]
```

## DESCRIPTION

Git is a fast, scalable, distributed revision control system with an unusually rich command set that provides both high-level operations and full access to internals.

See **gittutorial**(7) to get started, then see **giteveryday**(7) for a useful minimum set of commands. The **Git User's Manual**[1] has a more in-depth introduction.

After you mastered the basic concepts, you can come back to this page to learn what commands Git offers. You can learn more about individual Git commands with "git help command". **gitcli**(7) manual page gives you an overview of the command-line command syntax.

A formatted and hyperlinked copy of the latest Git documentation can be viewed at **https://git.github.io/htmldocs/git.html**.

:


git

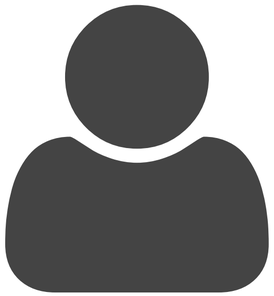

git


git
I'M WRITING THIS DOWN
IT'S GOOD STUFF

# What this talk is about

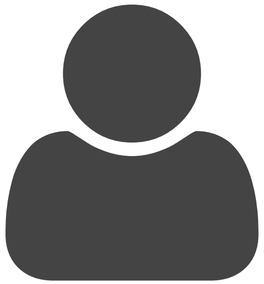
A collection of Git habits
to improve your development workflow
both individually and as a team

# Let’s Get Started



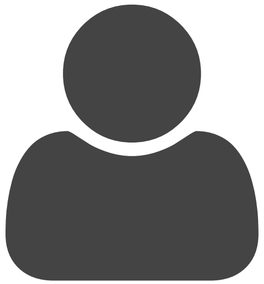
Personal Workflows

Team Workflows



1 2 3

1 2 3



1 2 3



1 2 3

# 1 Get Comfy

Configure your environment


# Git-aware Prompt

posh-git

bash-git-prompt

starship

# Pretty Diffs

```
core.pager "diff-so-fancy" | less -RFX
```

# ⟷ Whitespace Handling

```
core.whitespace tab-in-indent
```

```
git diff --check
```

# Better Conflict Handling

```
merge.conflictstyle diff3
```

# ② Craft Your History

Create ACID commits

# Autonomous

A commit should contain a

single logical change

independent from other commits

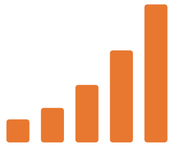
```
git add --patch
```

# Correct

A commit should not break
the state of the working tree

```
git stash save --keep-index
```

```
git rebase --exec
```



# Incremental

A commit should represent a

logical progression

compared to previous commits

```
git rebase --interactive
```

# Documented

A commit should clearly state

the purpose of the change

and the context in which it was made

# 3 Automate!

Let Git do the work for you

# Aliased Functions

```
git config alias.undo
```

# Reuse Recorded Resolutions

```
git config rerere.enabled true
```

# Client-side Hooks

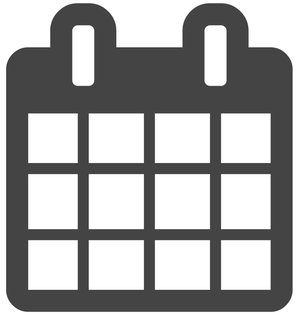
```
pre-commit # Validate the patch
```

```
commit-msg # Validate the message
```

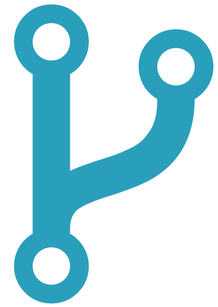
# 1 Branching

Be aware of the different

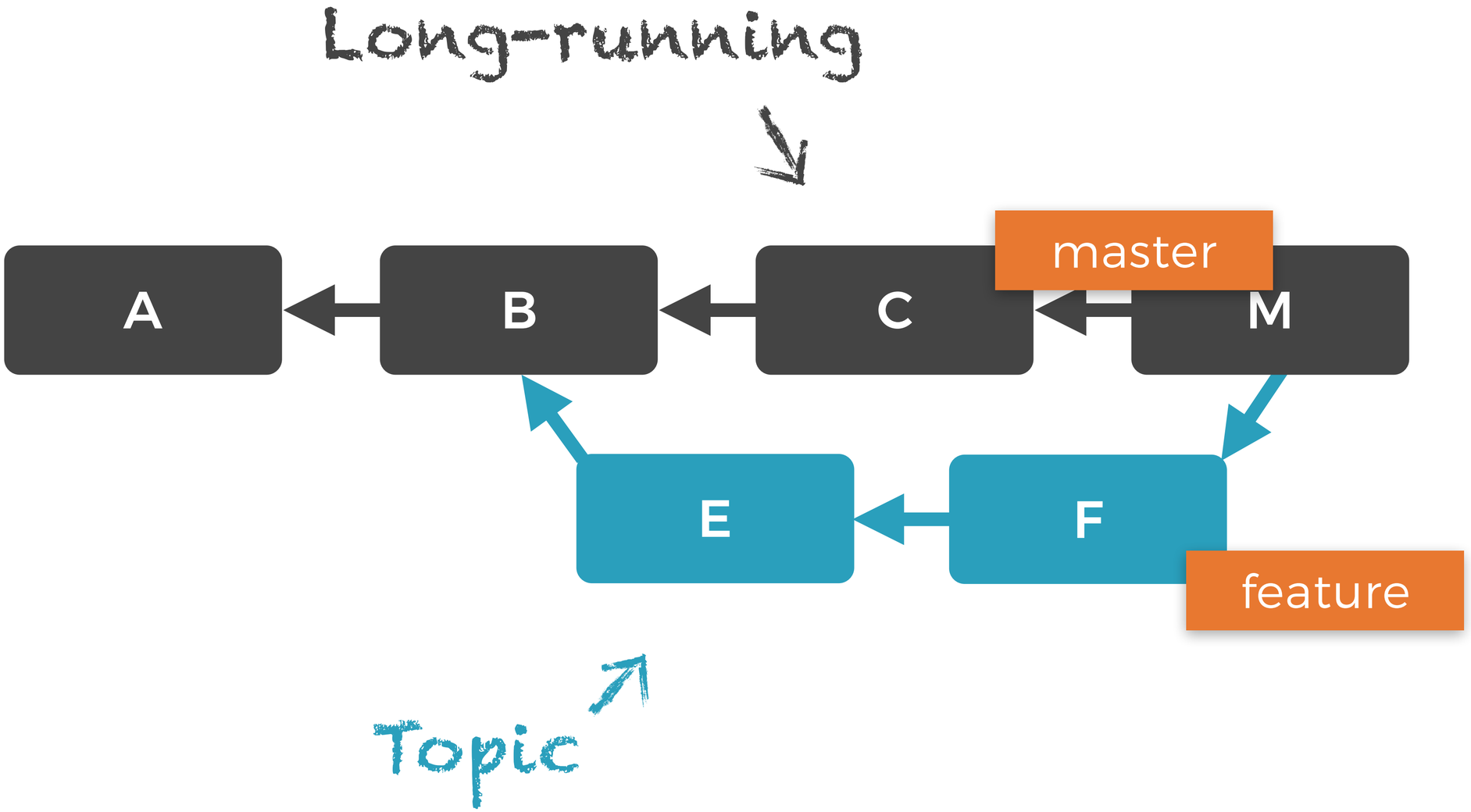
kinds of branches

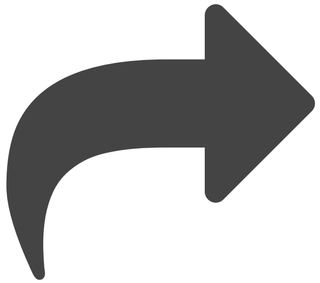


Long-running



Topic

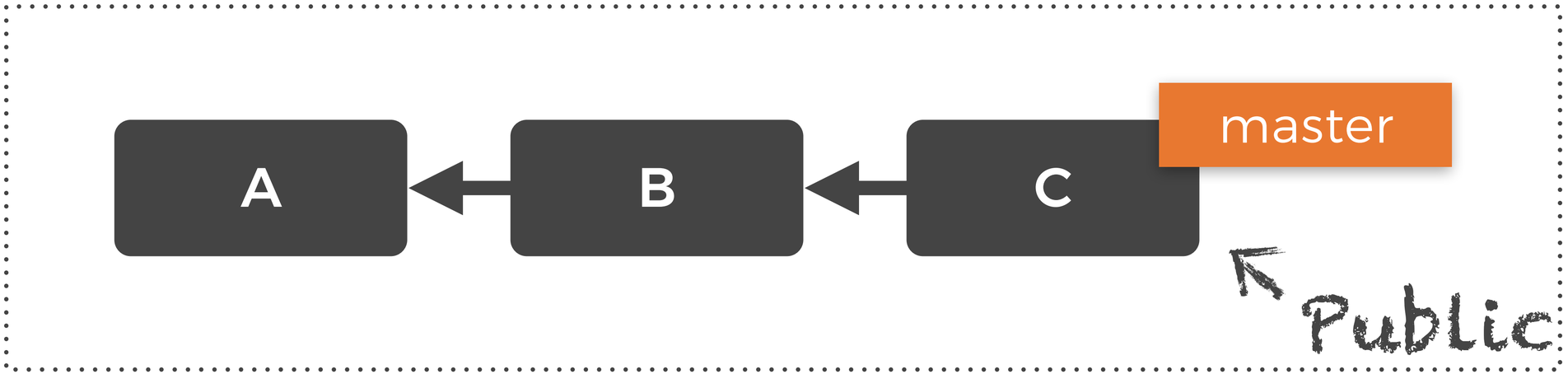

Long-running
master
A
B
C
M
E
F
feature
Topic



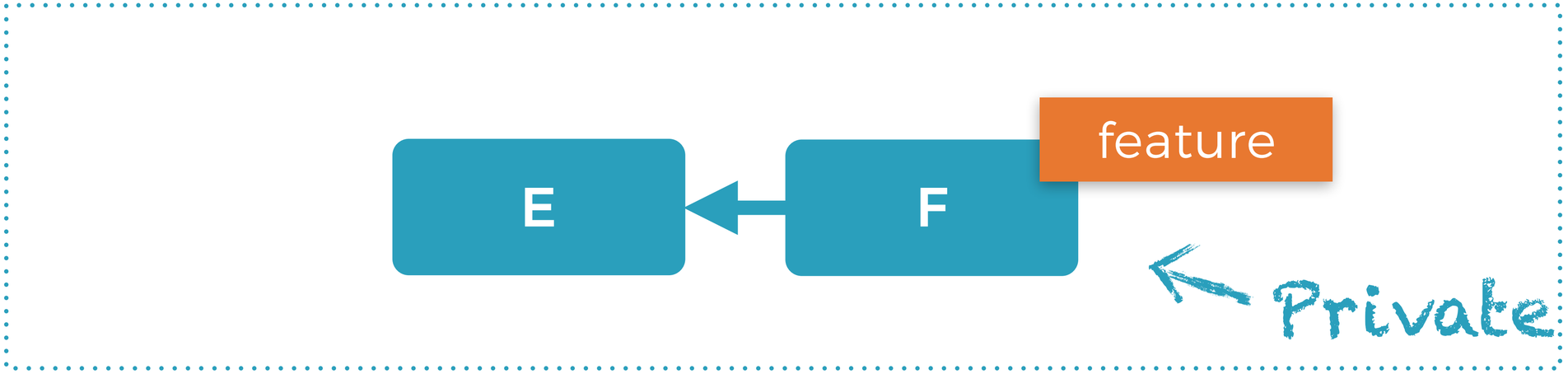
Public

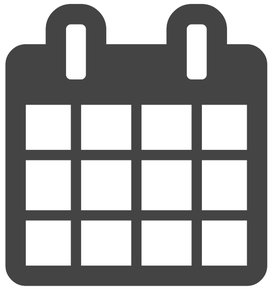
Private

## Shared Repo

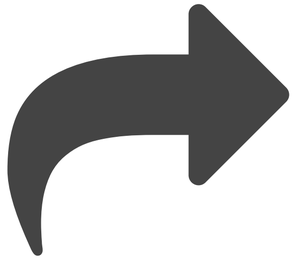



## Local Repo

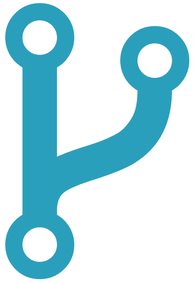




Long-running

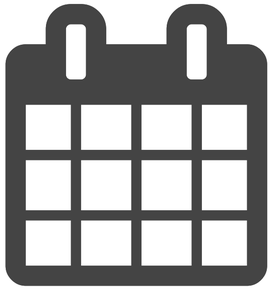


Public

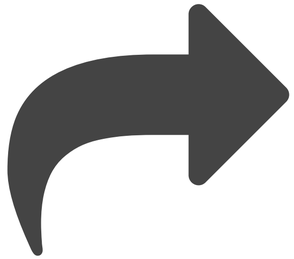


Topic

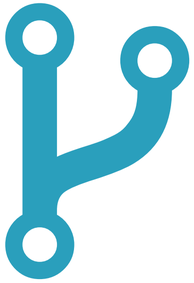
Private



Long-running

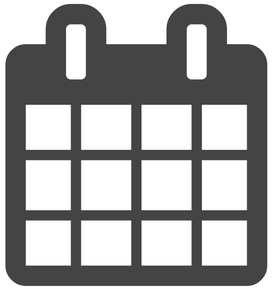


Public

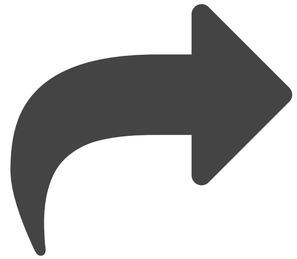


Topic

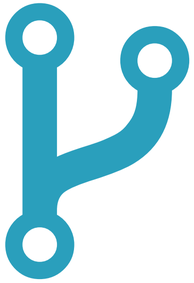
Private



Long-running



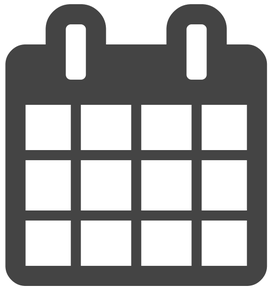
Public



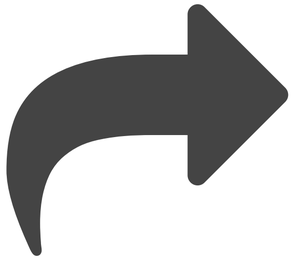
Topic

Private

## Be Conservative

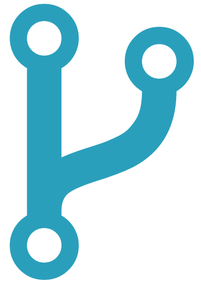


Long-running



Public

## Be Liberal

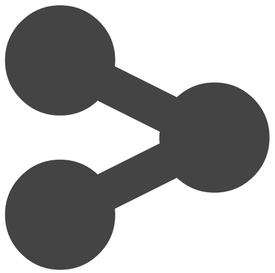


Topic

Private

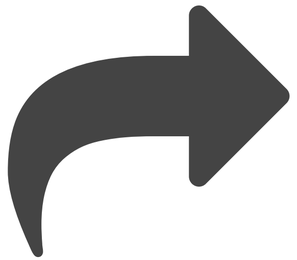
# 2 Merging

Let the branches decide
your merge strategy

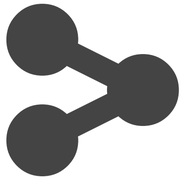


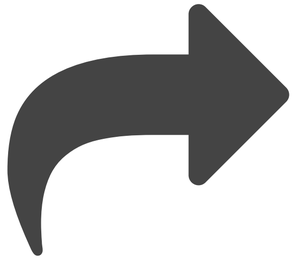
True Merge

Rebase

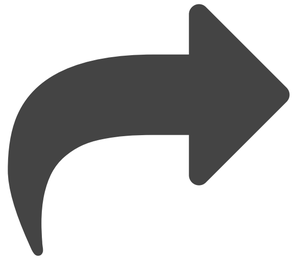


Public



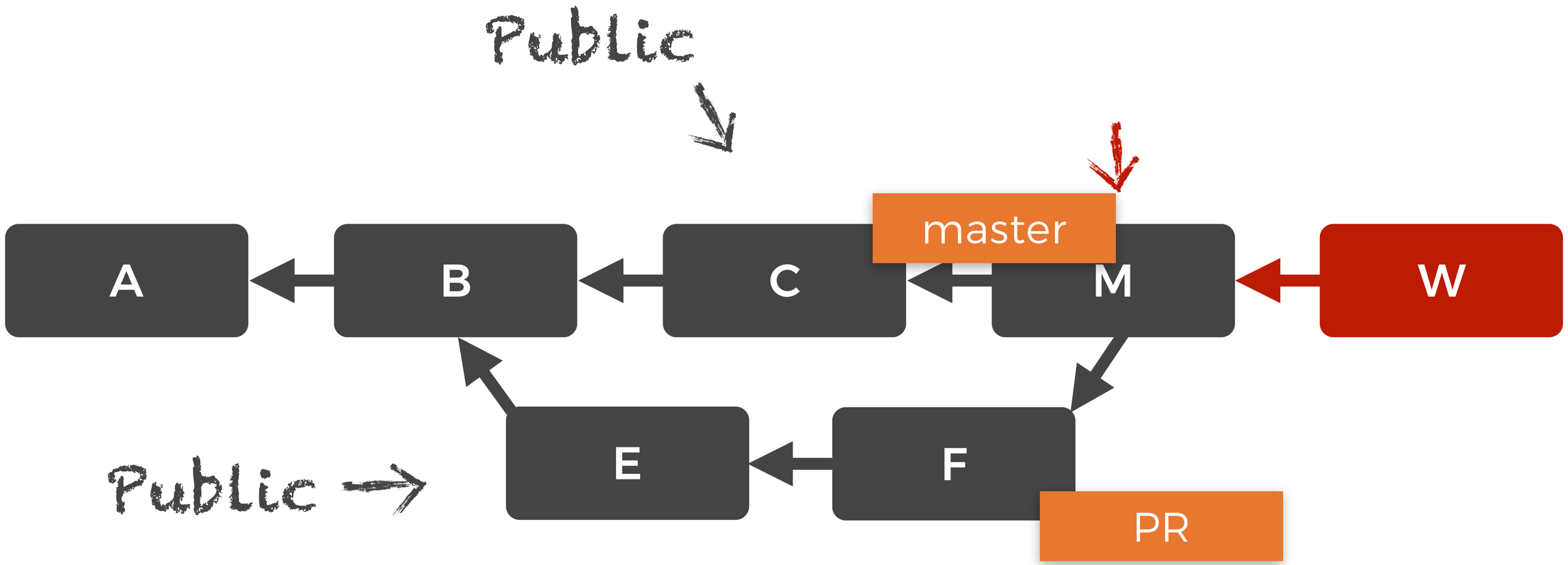


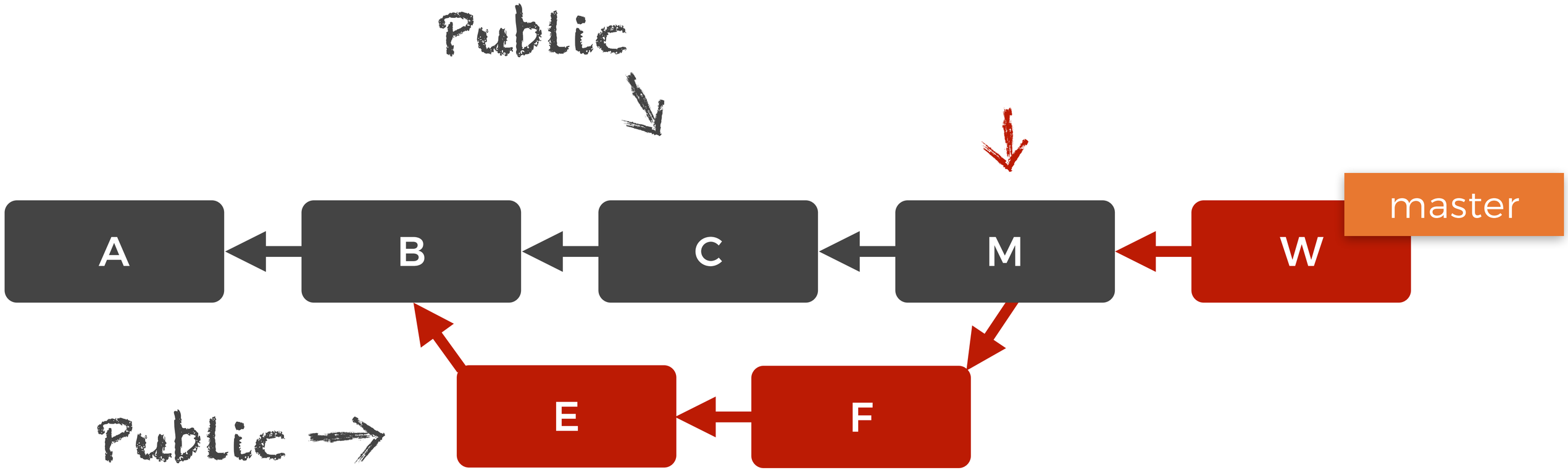
Public

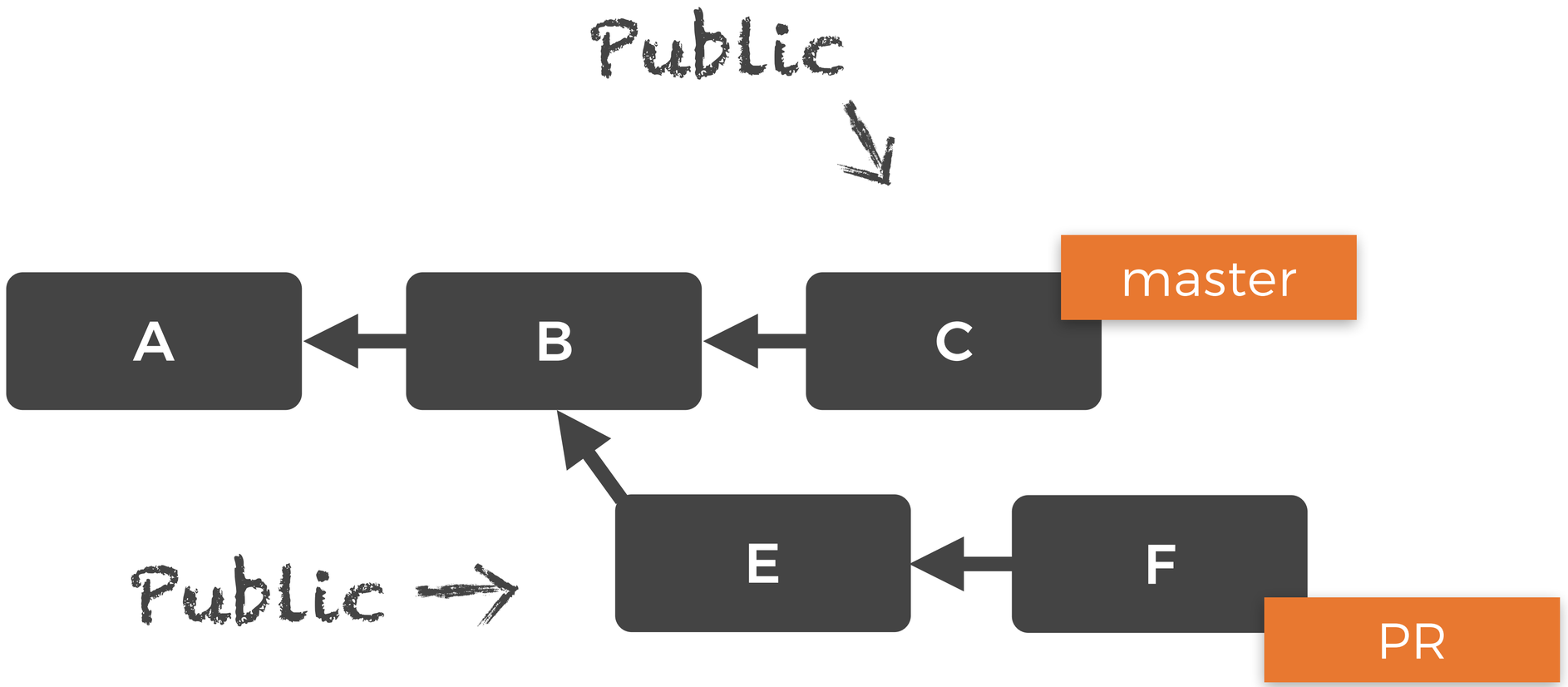


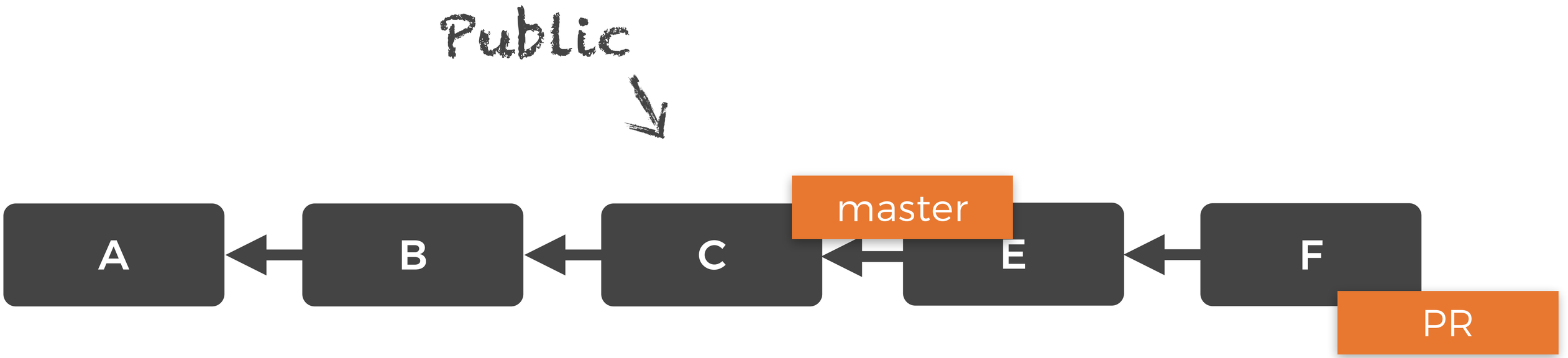
Public

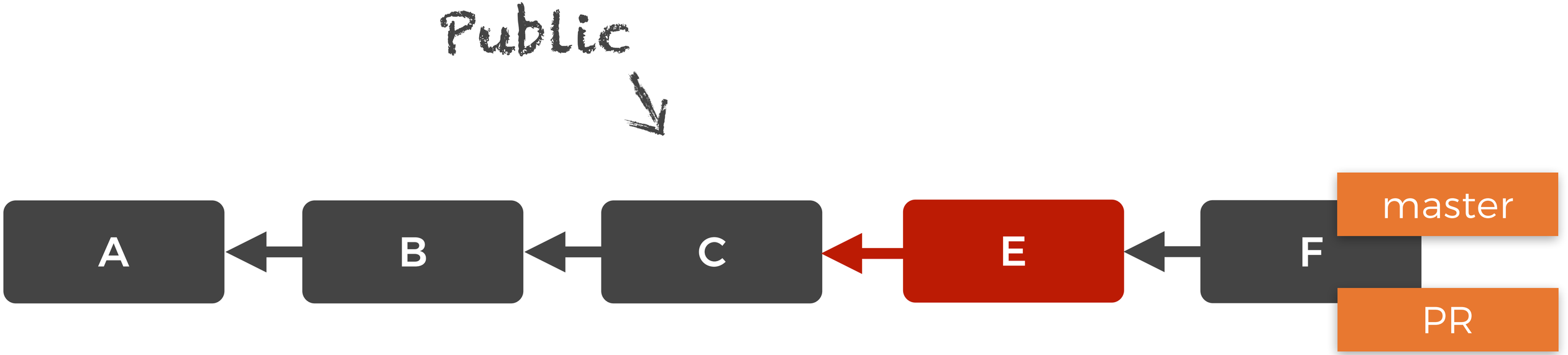
Private

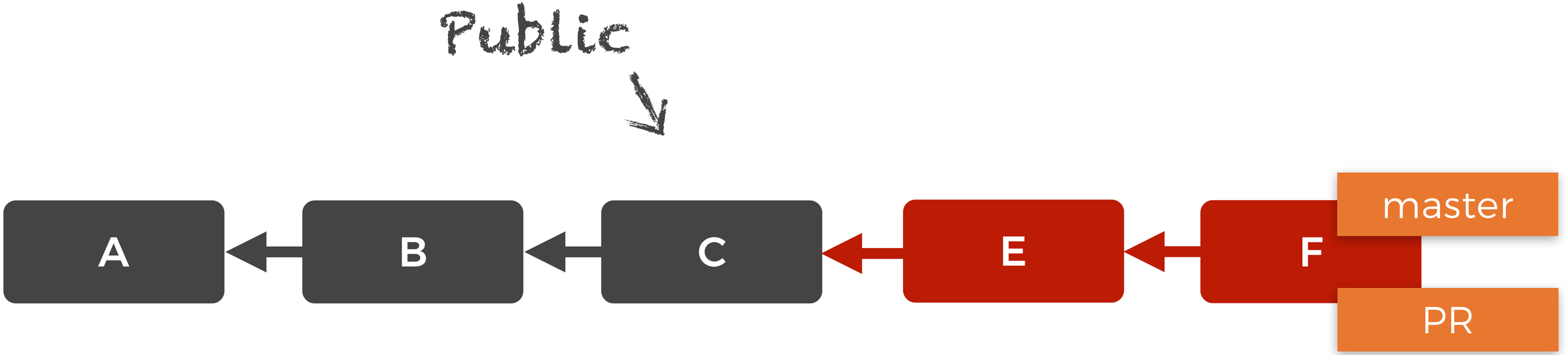

Public
master
A
B
C
M
W
E
F
PR
Public

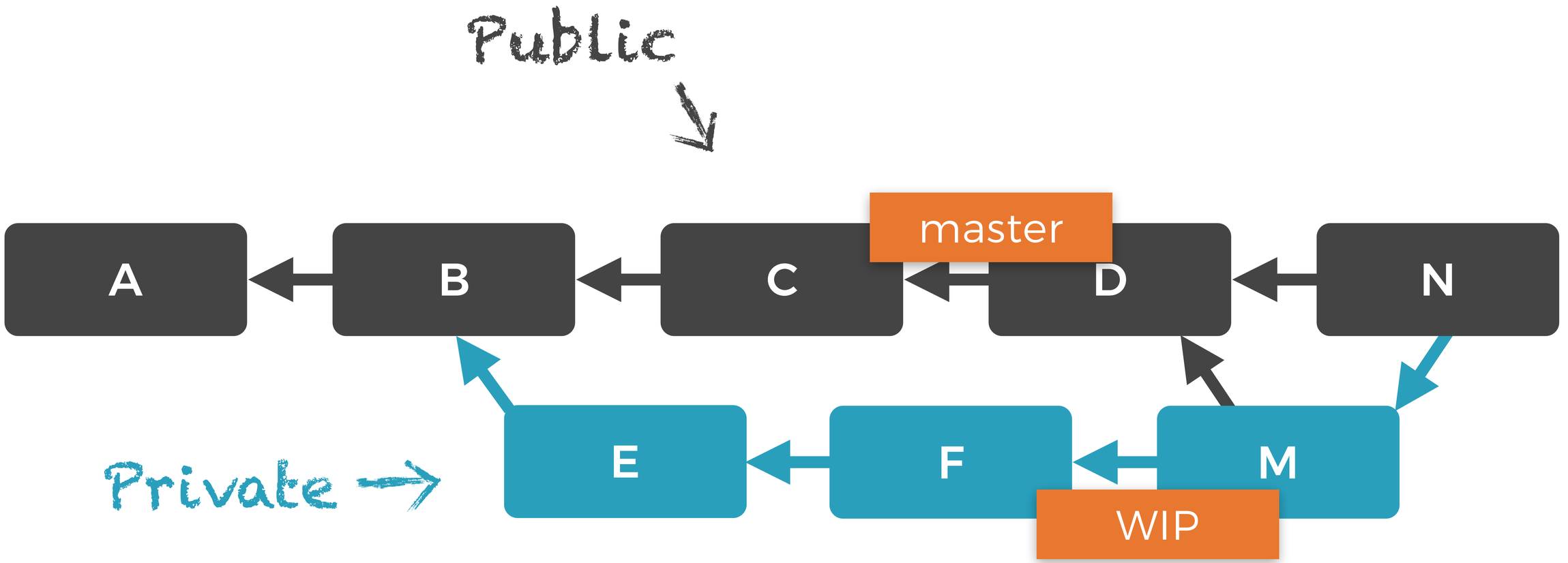

Public
A
B
C
M
W
master
Public
E
F


Public
A
B
C
master
Public
E
F
PR

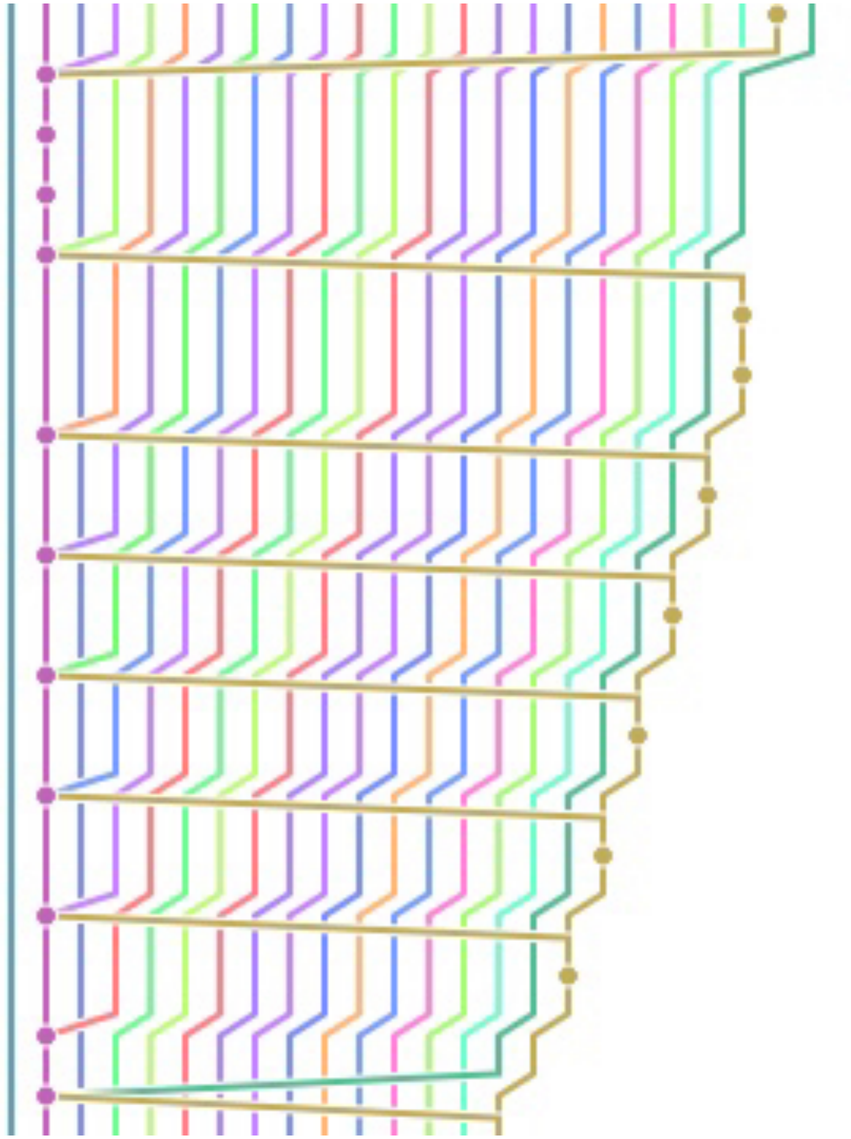

Public
master
A
B
C
E
F
PR


Public
A
B
C
E
F
master
PR

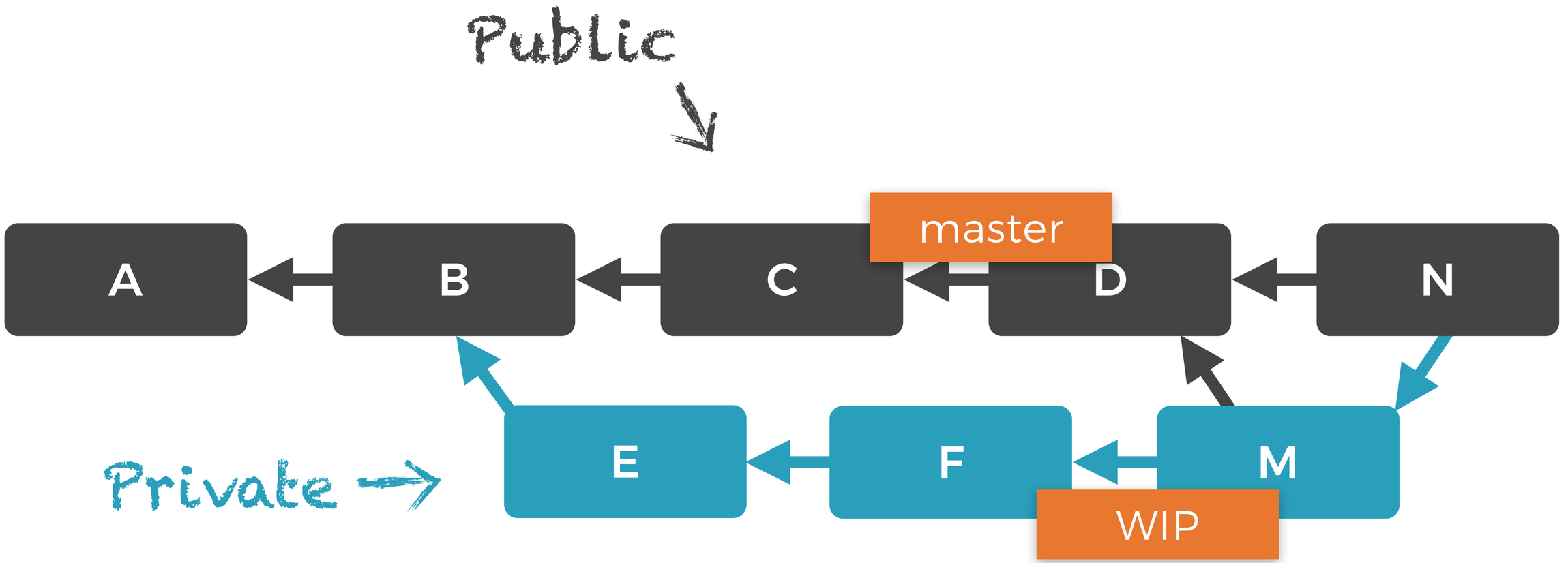

Public
A
B
C
E
F
master
PR

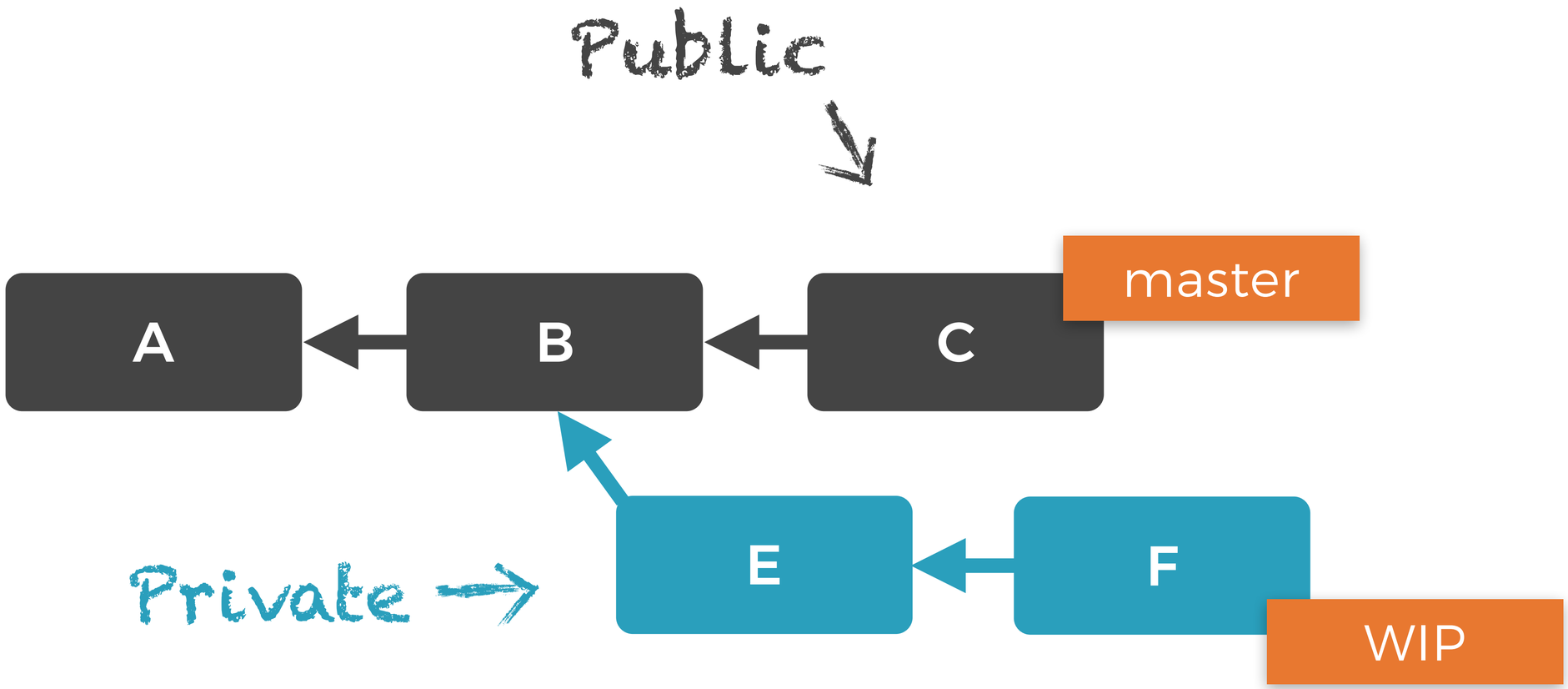

Public
master
A
B
C
D
N
Private
E
F
M
WIP

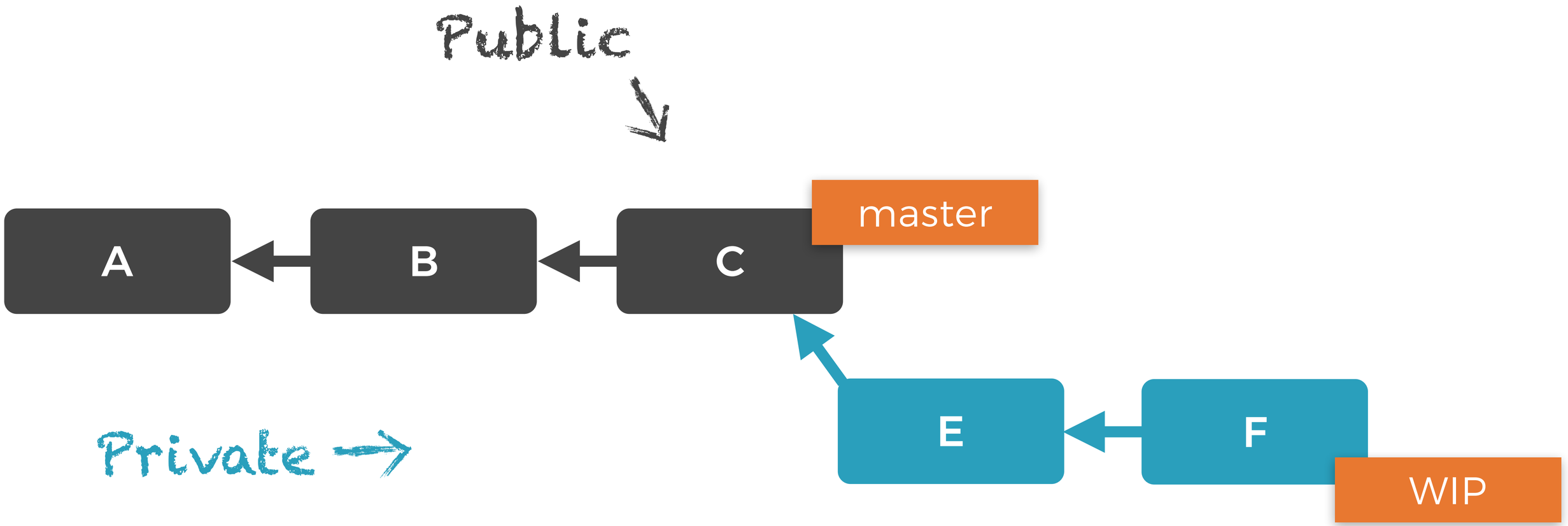
Guitar Hero History →

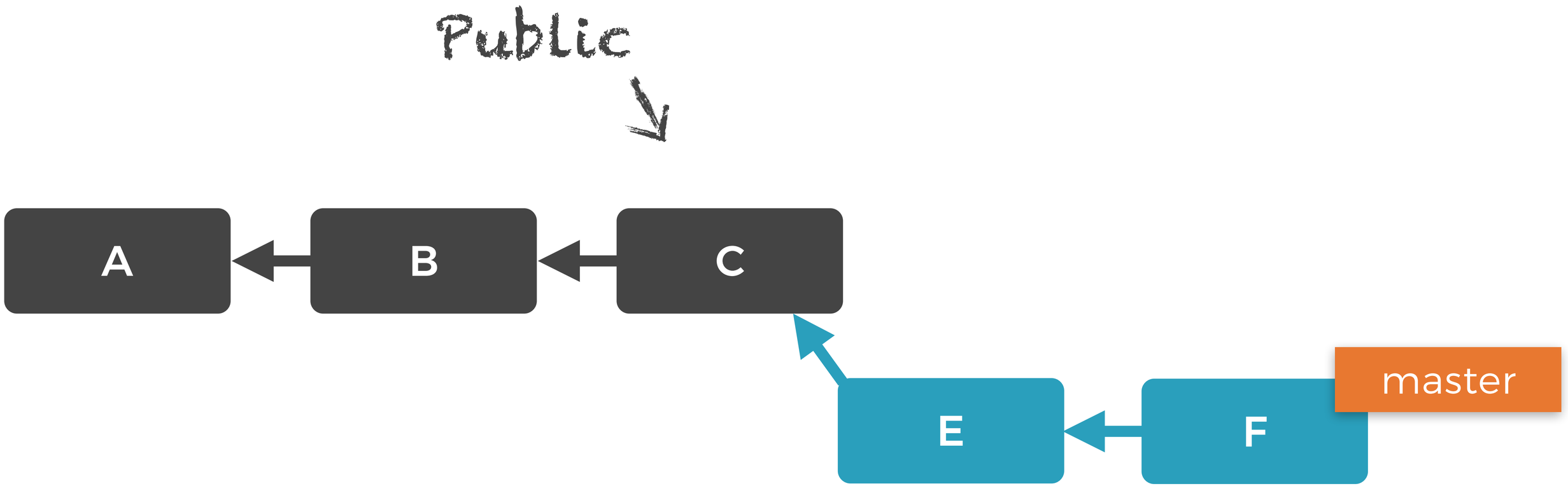


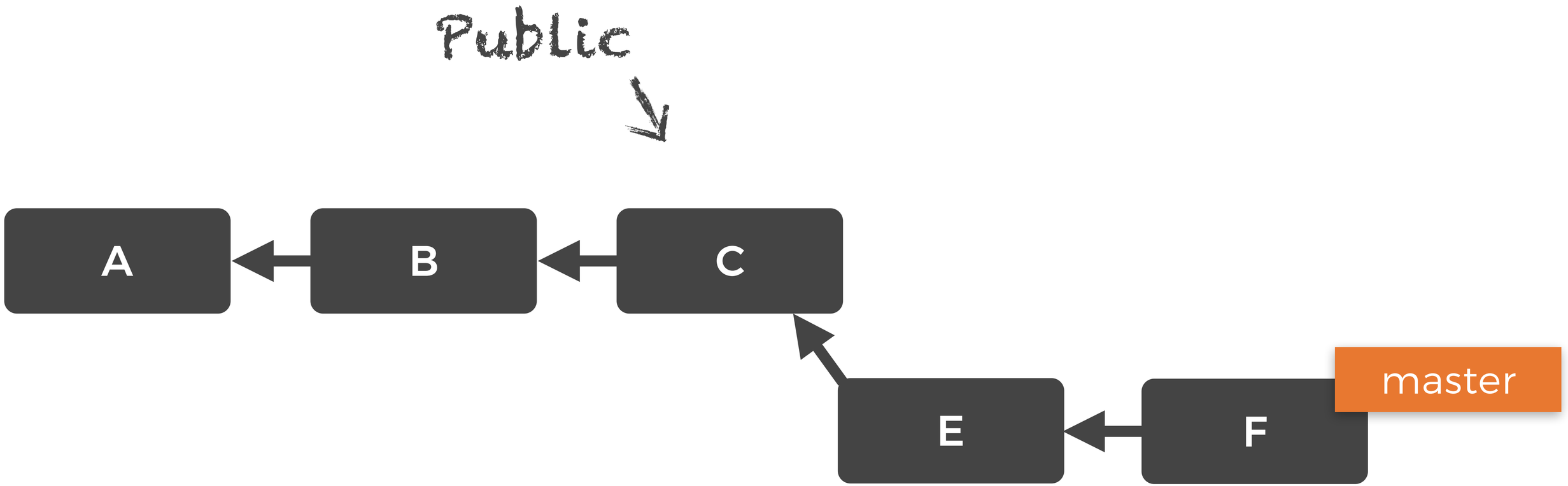

Git Hero
7138
038
ROCK


Public
master
A
B
C
D
N
Private
E
F
M
WIP


Public
A
B
C
master
Private
E
F
WIP


Public
A
B
C
master
Private
E
F
WIP


Public
A
B
C
E
F
master

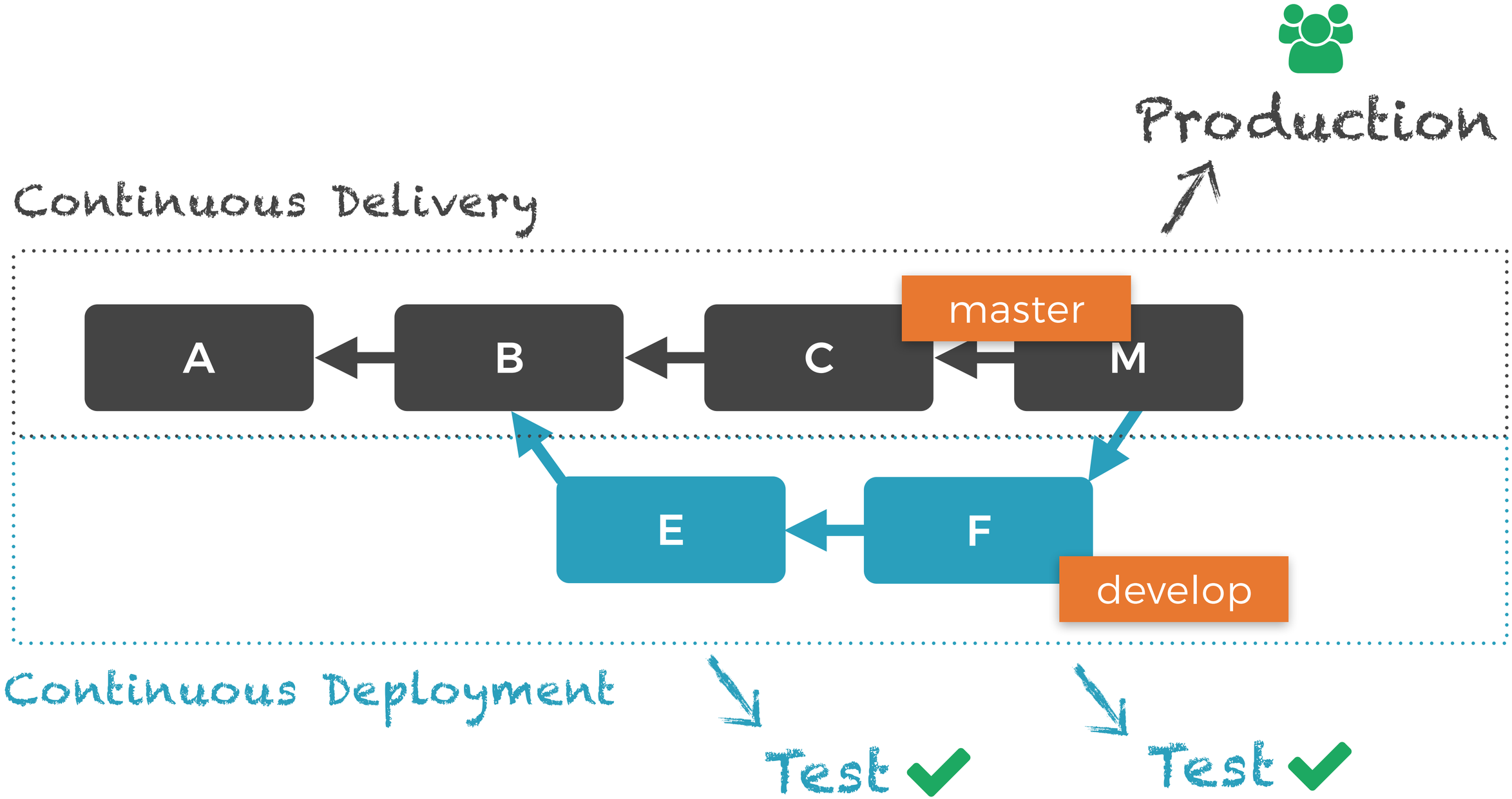

Public
A
B
C
E
F
master

# 3 Releasing

Map your branches to
your release environments

# 3 Releasing

Know where your branches are going


Production
Continuous Delivery
master
A
B
C
M
E
F
develop
Continuous Deployment
Test
Test

# 3 Releasing

Let your branches determine
your versioning scheme

# Let’s Wrap Up

1. Get Comfy
2. Craft History
3. Automate!

1. Branching
2. Merging
3. Releasing

# Pluralsight | Advanced Git Tips and Tricks

bit.ly/git-tips-tricks

@ecampidoglio

# Thank you.

 @ecampidoglio

 megakemp.com/git

 bit.ly/git-tips-tricks