A story…
The Git Eureka Moment

by Dave Liddament (aged 38 and a few days)

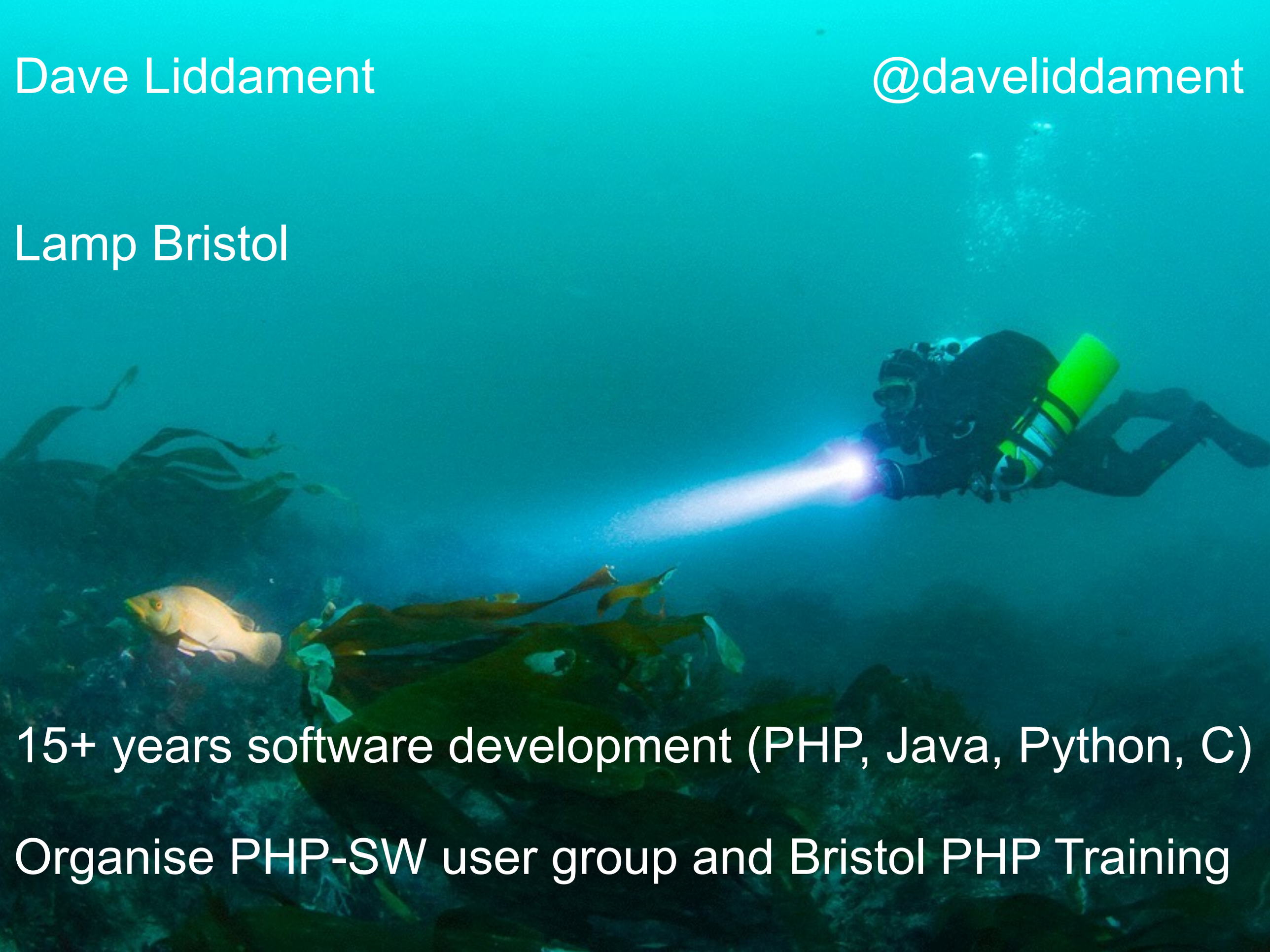
First a story…

… to be continued…

Dave Liddament

@daveliddament

Lamp Bristol

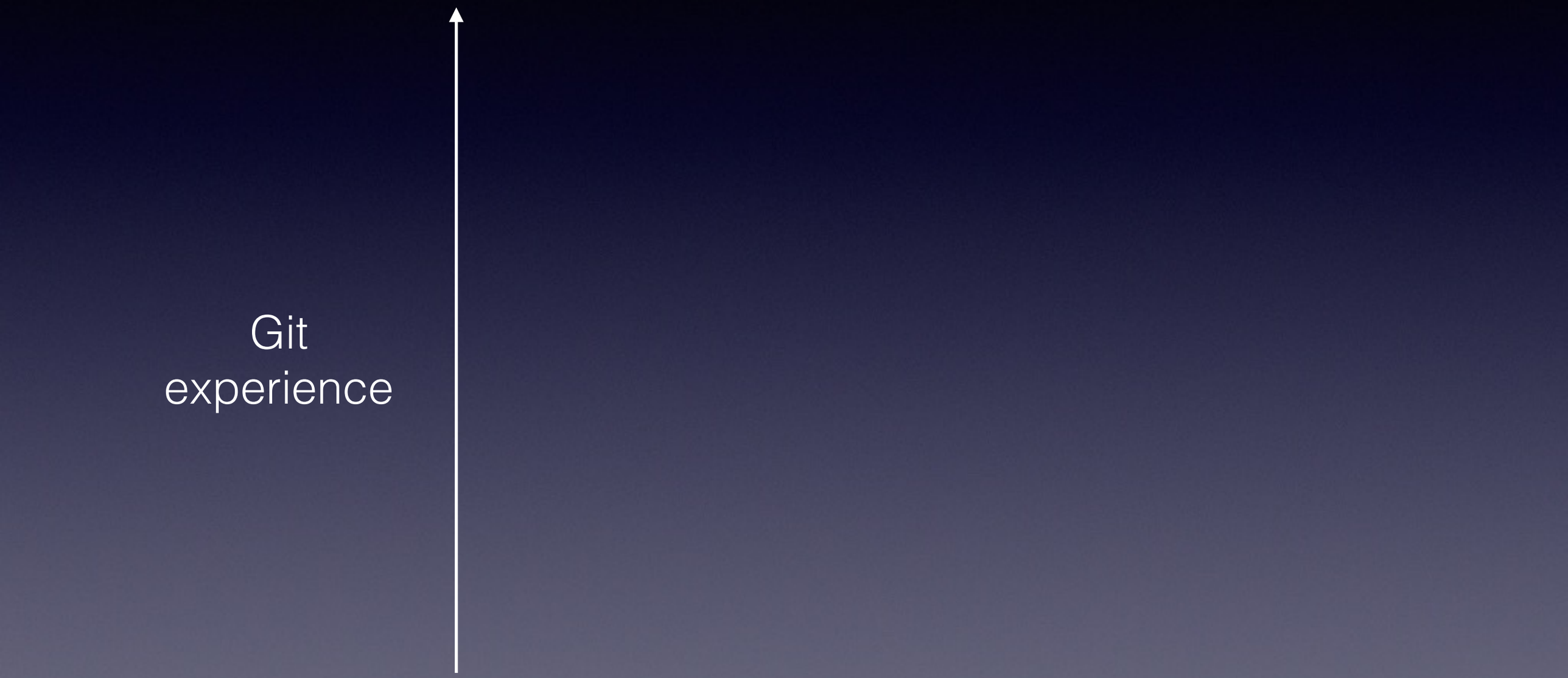


15+ years software development (PHP, Java, Python, C)

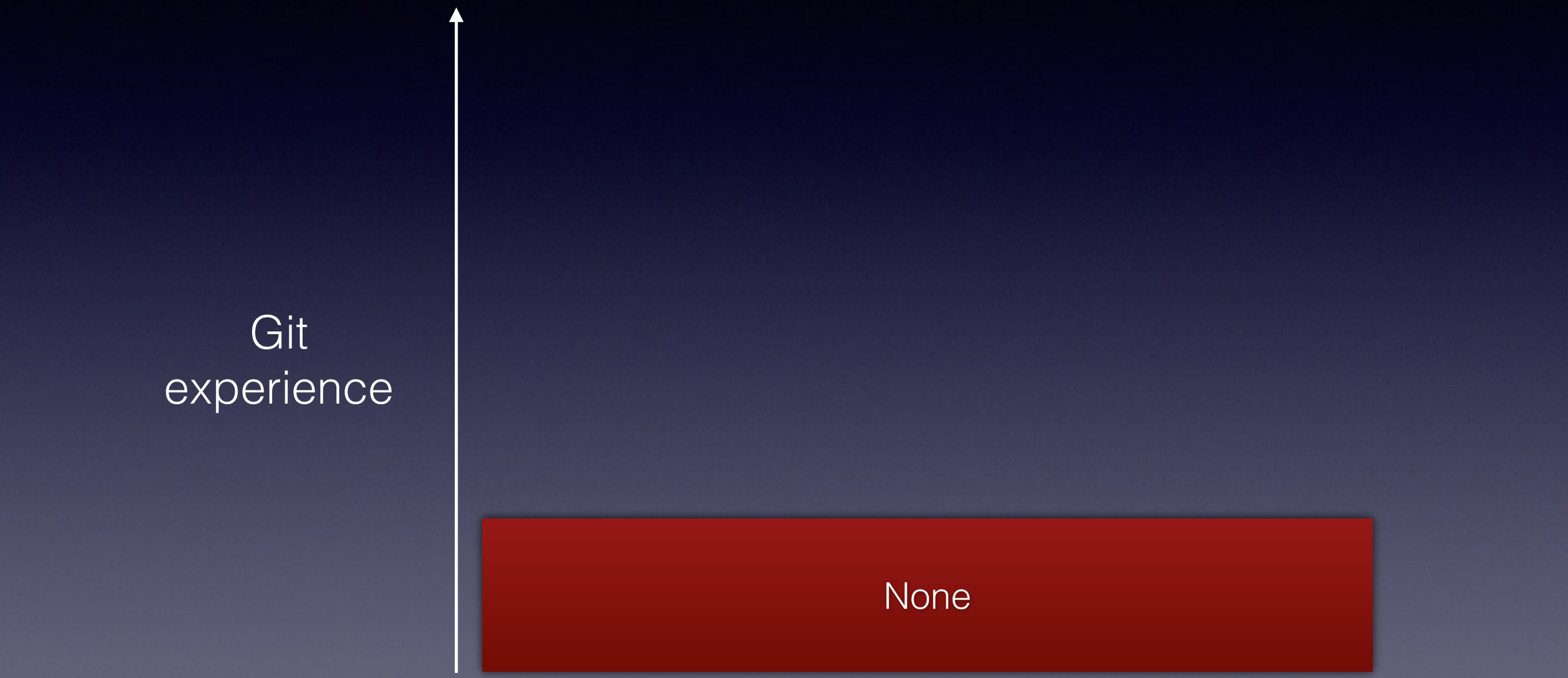
Organise PHP-SW user group and Bristol PHP Training

Poetic licence

Is this talk for you?



Is this talk for you?



Is this talk for you?

Git experience

None

Is this talk for you?

Git blackbelt

Git experience

None

Is this talk for you?

Git experience

Git blackbelt

None

Is this talk for you?

Git experience

Git blackbelt

Somewhere in-between

None

Git Commit

Creating a commit

Create a new file: days.txt

Creating a commit

```
git status

On branch master

Untracked files:
  (use "git add <file>..." to include in what will be committed)

  days.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Creating a commit

```
On branch master

Untracked files:
  (use "git add <file>..." to include in what will be committed)

  days.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Creating a commit

```
On branch master

Untracked files:
  (use "git add <file>..." to include in what will be committed)
```

days.txt

```
nothing added to commit but untracked files present (use "git add" to track)
```

Creating a commit

```
git add days.txt
```

Creating a commit

```
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

	new file:   days.txt
```

Creating a commit

```
On branch master
Changes to be committed:
   (use "git reset HEAD <file>..." to unstage)

   new file:   days.txt
```

Creating a commit

```
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   days.txt
```

Creating a commit

```
git commit
```

Creating a commit

```
[master adfabb7] ADD days
1 file changed, 7 insertions(+)
create mode 100644 days.txt
```

Creating a commit

Added a file containing the days of the week

Creating a commit

Untracked

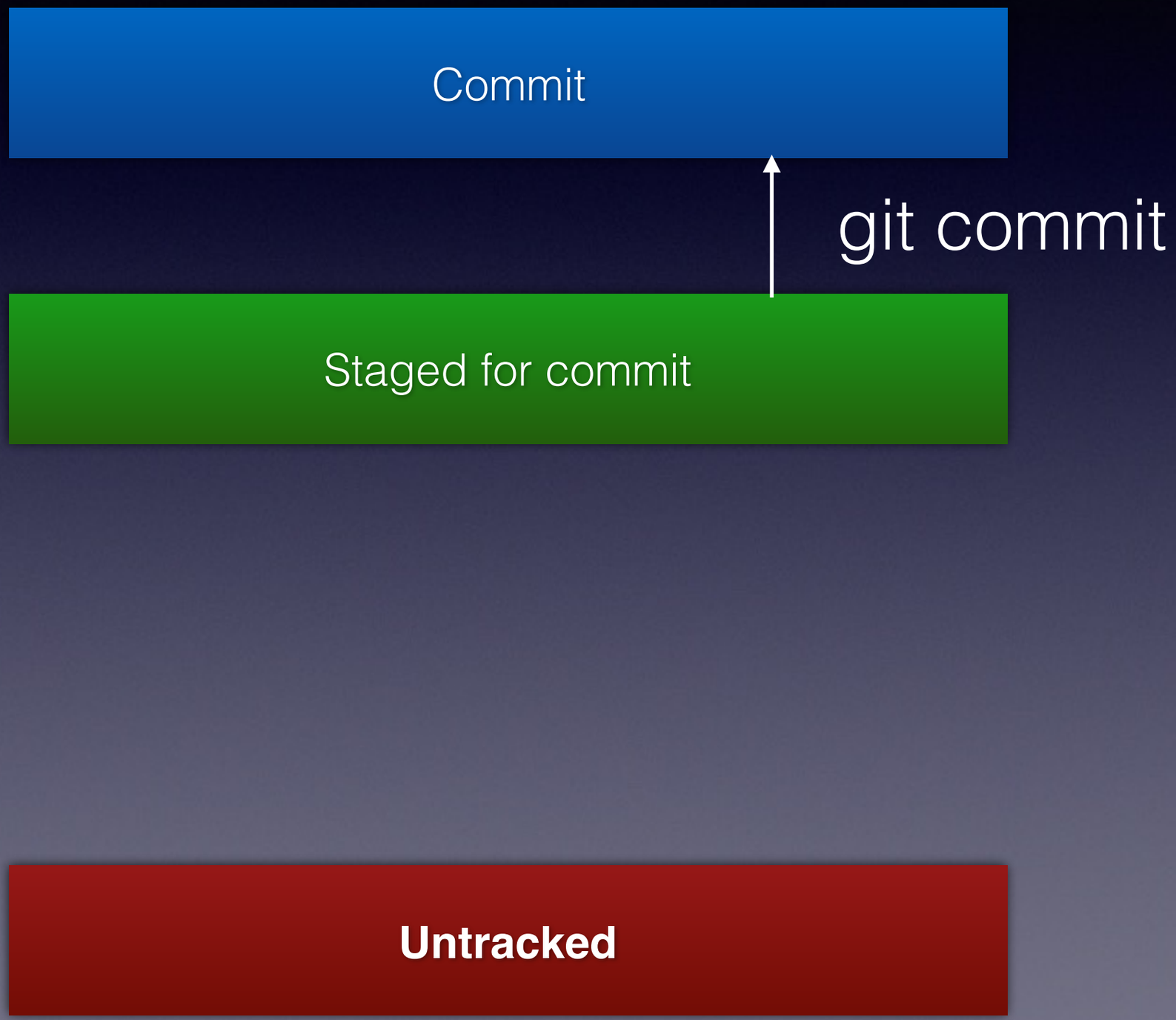
Creating a commit

Staged for commit

git add days.txt

Untracked

Creating a commit



Creating a commit (2)

Update an existing file: days.txt

Creating a commit (2)

```
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

  modified:   days.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

Creating a commit (2)

```
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

  modified:   days.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

Creating a commit (2)

```
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   days.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

Creating a commit (2)

```
git add days.txt
```

Creating a commit (2)

```
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

	modified:   days.txt
```

Creating a commit (2)

```
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   days.txt
```

Creating a commit (2)

```
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   days.txt
```

Creating a commit (2)

```
git commit
```

Creating a commit (2)

```
[master 6e88dad] FIX typo
1 file changed, 1 insertion(+), 1 deletion(-)
```

Creating a commit (2)

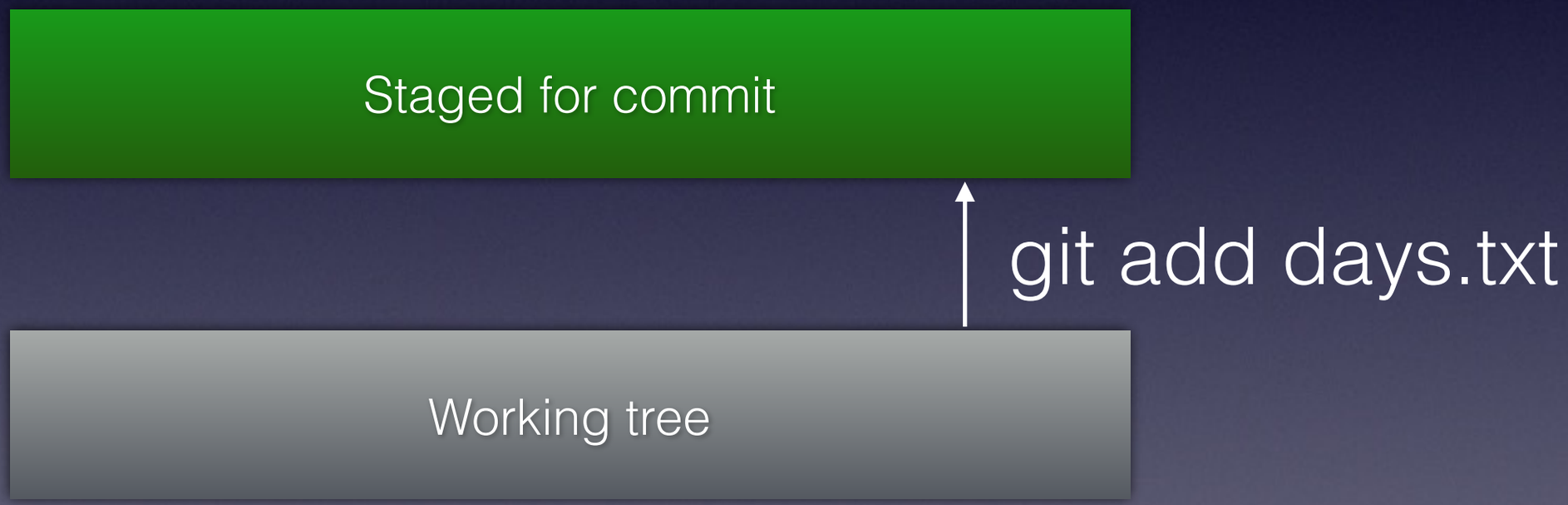
Added a file containing the days of the week

Fixed typo

Creating a commit (2)

Working tree

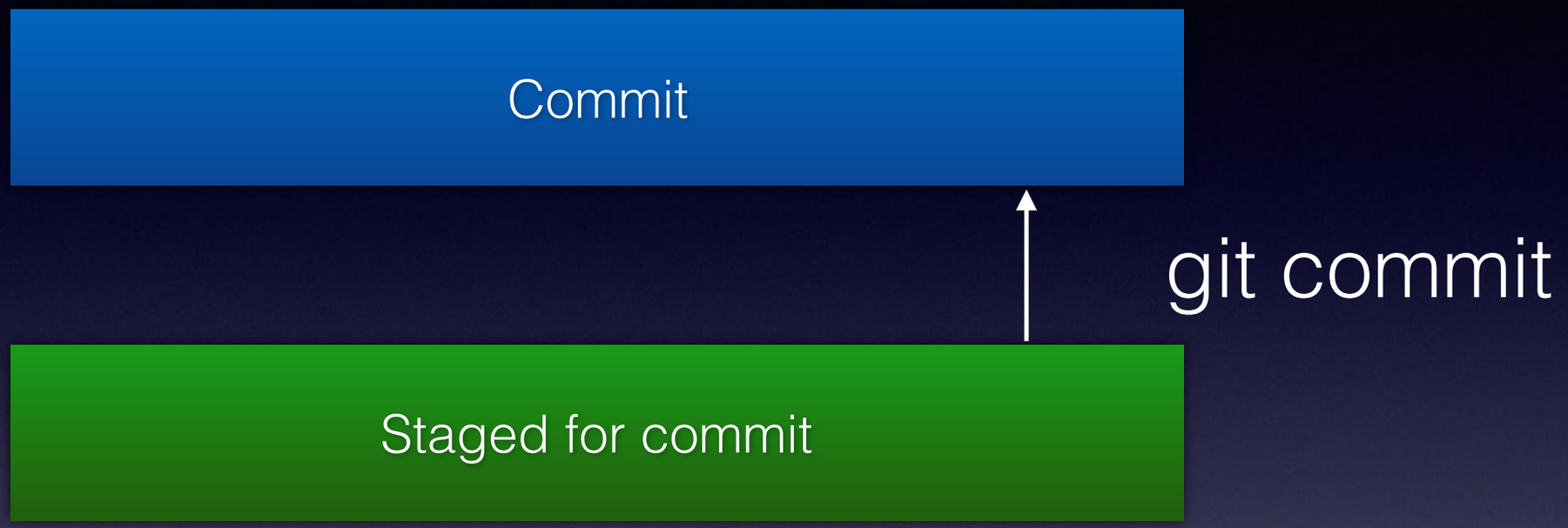
Creating a commit (2)



Creating a commit (2)

Staged for commit

Creating a commit (2)



Take away

Take away

```
git status

On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

  modified:   days.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

  modified:   months.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)

  seasons.txt
```

Take away

Commit

Staged for commit

Working tree

Untracked

Take away

Commit

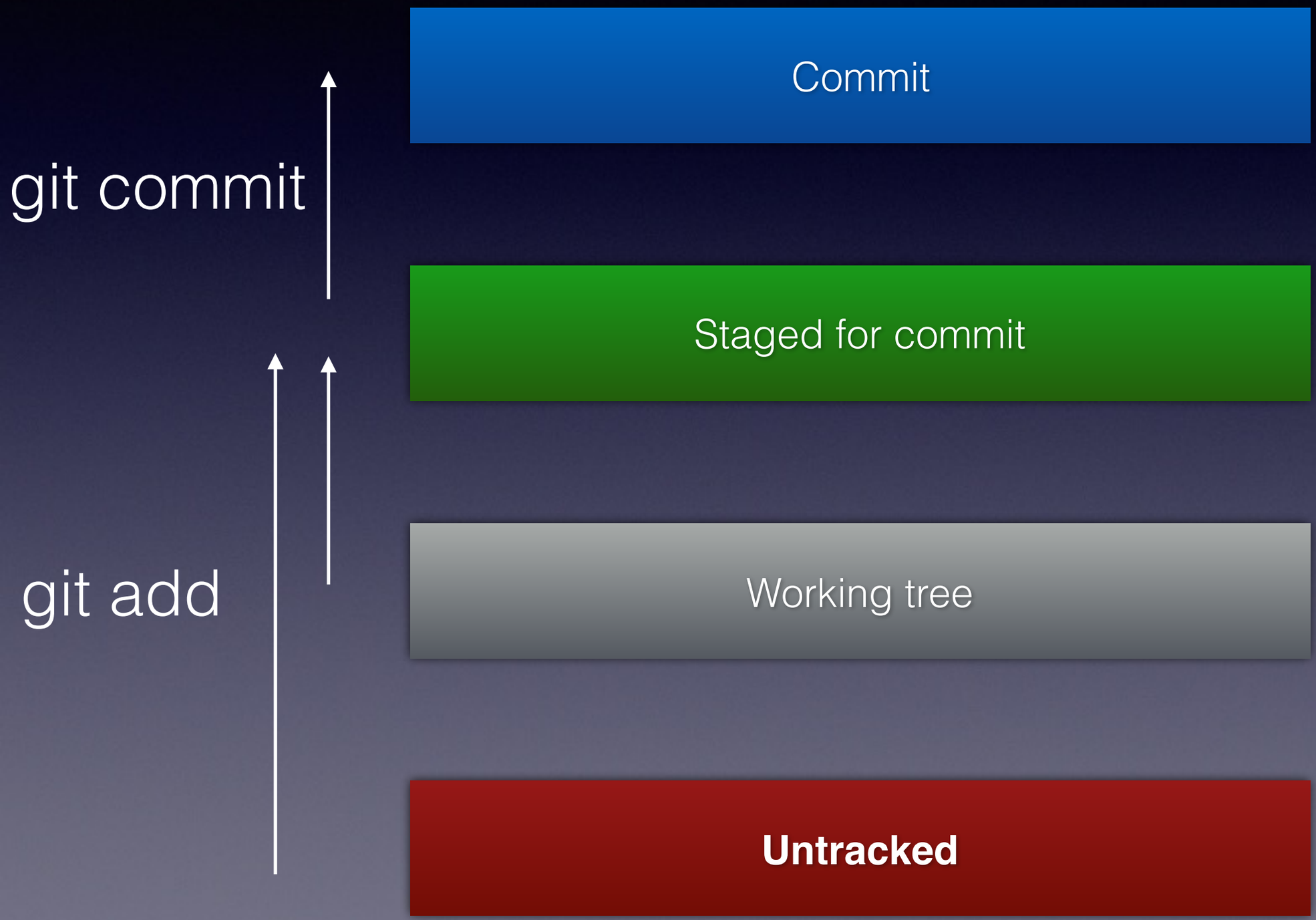
Staged for commit

git add

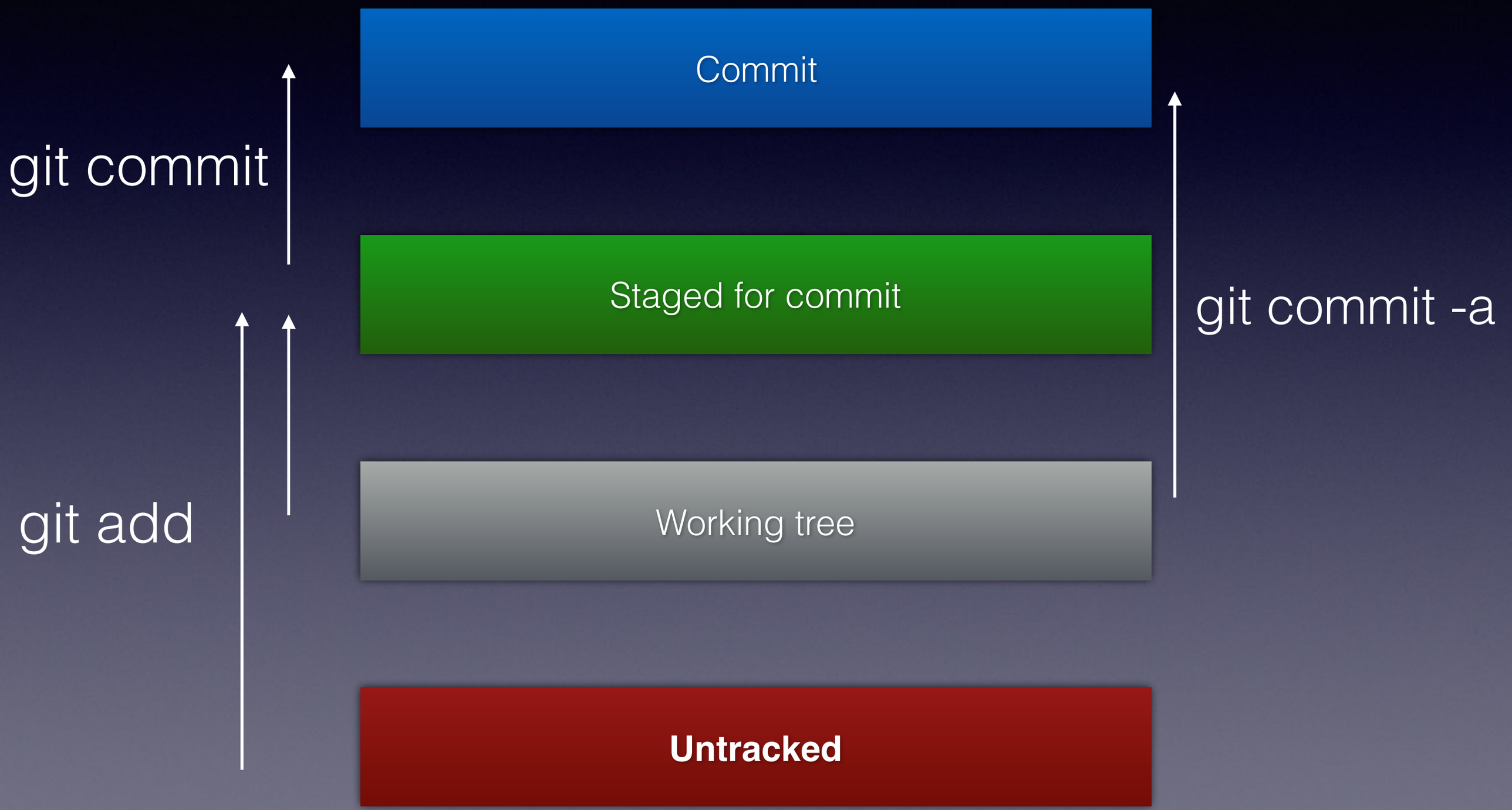
Working tree

Untracked

Take away


Commit
git commit
Staged for commit
Working tree
git add
Untracked

Take away


Commit
git commit
Staged for commit
git commit -a
git add
Working tree
Untracked

Viewing changes

Viewing changes

Staged for commit

git diff

Working tree

Viewing changes

git diff --cached

Staged for commit

Working tree

What is a commit?

Warning

This is a simplification

What is a commit?

What is a commit?

What is a commit?

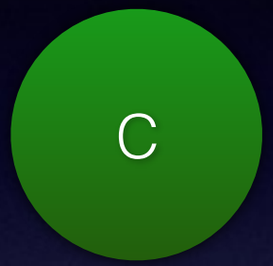
1. Metadata:

What is a commit?

1. Metadata:

```
Author: Dave Liddament <dave@lampbristol.com>
Date:   Mon Mar 13 16:02:32 2017 +0000

    FIX typo
```

What is a commit?



2. Patch:

What is a commit?

2. Patch:

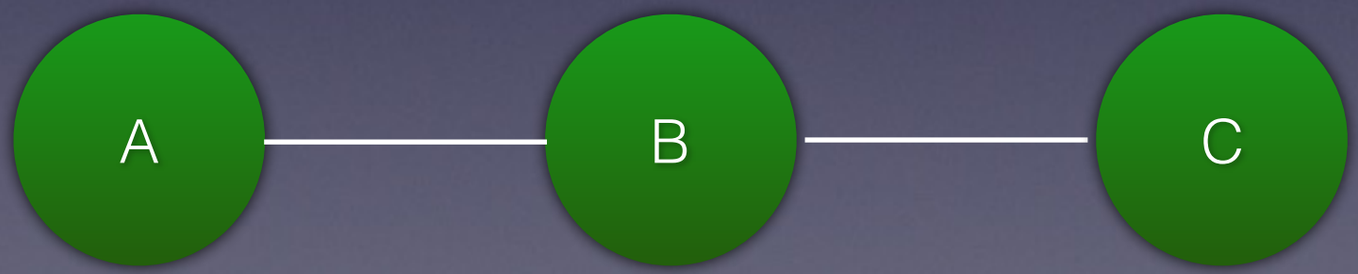
```
diff --git a/days.txt b/days.txt
index fe20ee3..1aee041 100644
--- a/days.txt
+++ b/days.txt
@@ -1,6 +1,6 @@
 Monday
 Tuesday
-Wendesday
+Wednesday
 Thursday
 Friday
 Saturday
```

What is a commit?

3. Parent commit(s):

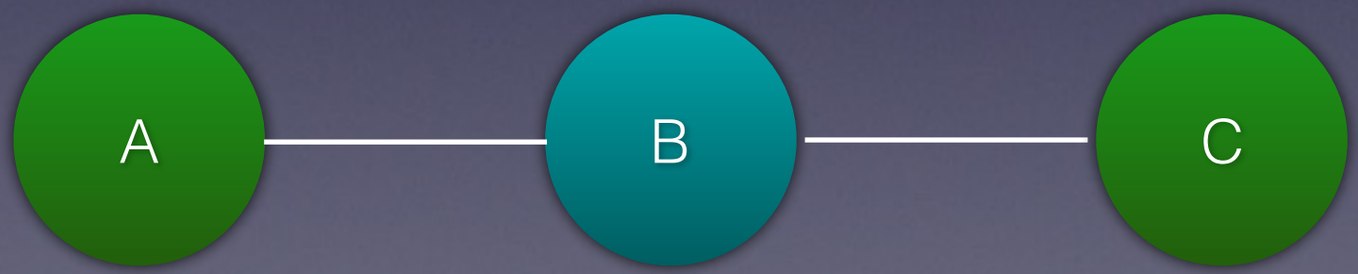
What is a commit?

3. Parent commit(s):



What is a commit?

3. Parent commit(s):

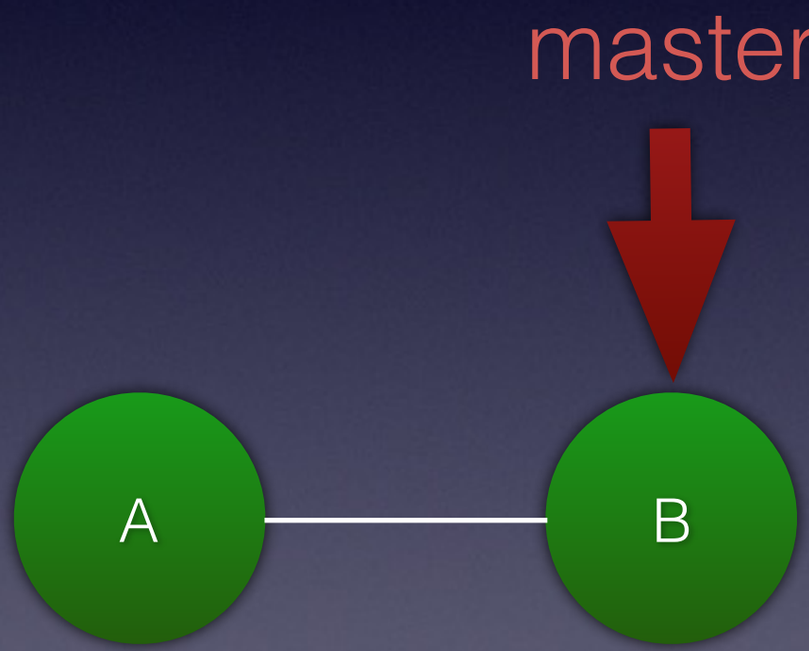


What is a commit?

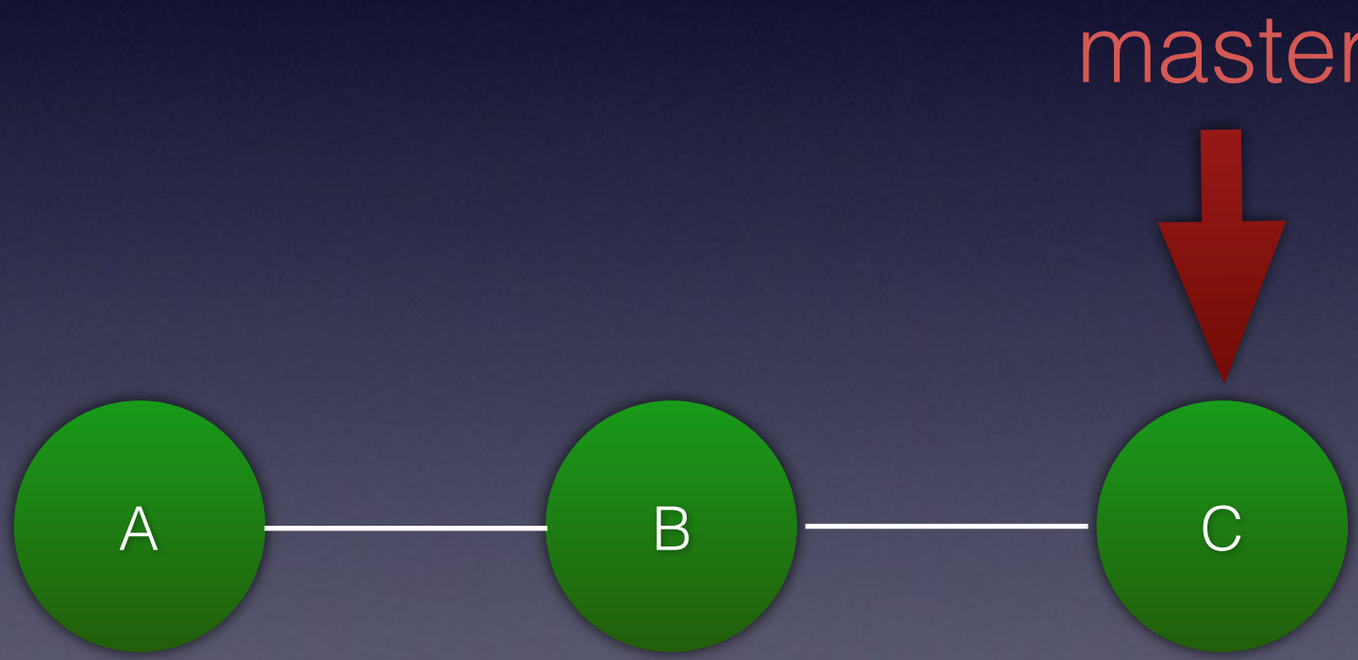
1. Metadata
2. Patch
3. Parent commits(s)

SHA: 6e88dad5d769b921d1a700bee8d57a7a82d67f29

What is a branch?



What is a branch?



What is a branch?

git log -n1

What is a branch?

git log -n1

```
commit 6e88dad5d769b921d1a700bee8d57a7a82d67f29
Author: Dave Liddament <dave@lampbristol.com>
Date:   Mon Nov 13 16:02:32 2017 +0000

    FIX typo
```

What is a branch?

git log -n1

```
commit 6e88dad5d769b921d1a700bee8d57a7a82d67f29
Author: Dave Liddament <dave@lampbristol.com>
Date:   Mon Nov 13 16:02:32 2017 +0000

    FIX typo
```

cat .git/refs/heads/master

What is a branch?

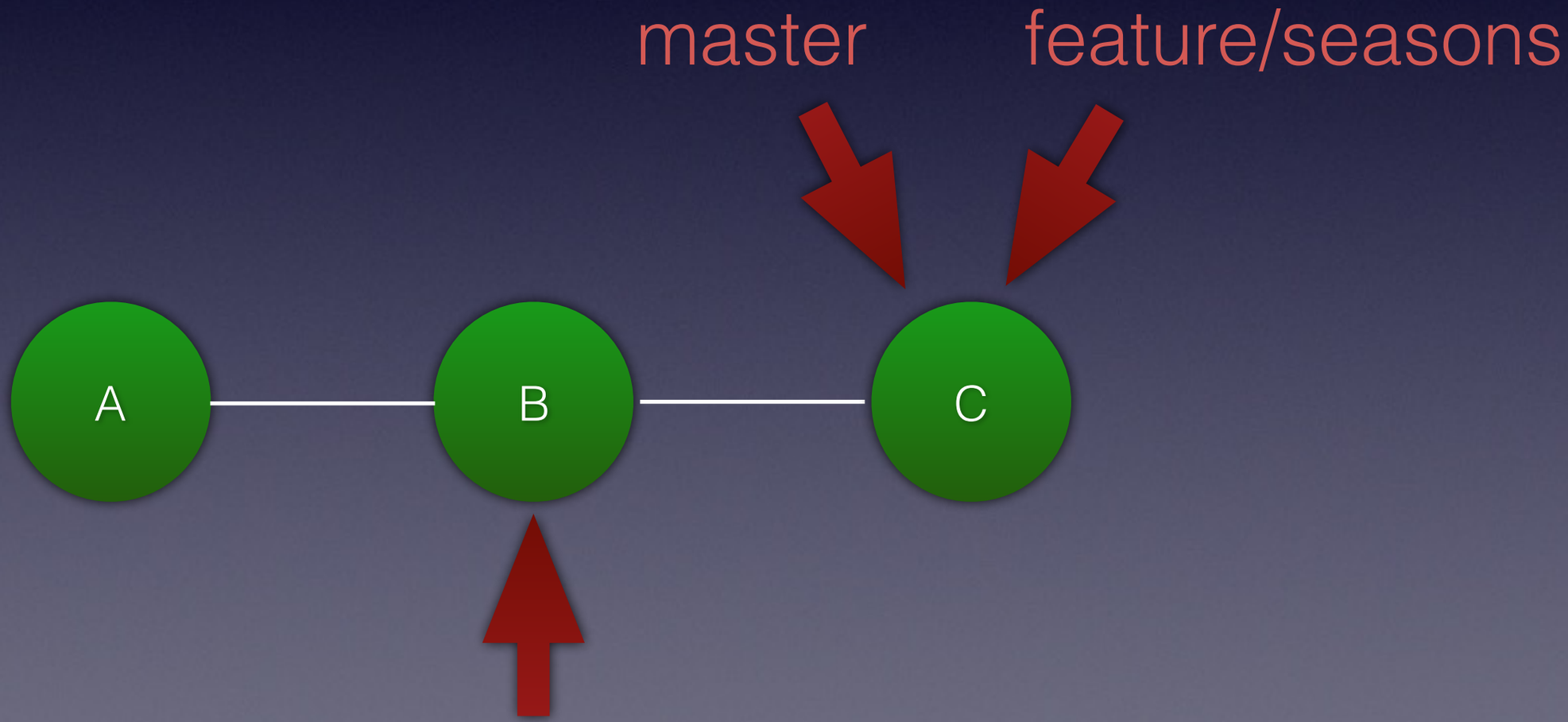
git log -n1

```
commit 6e88dad5d769b921d1a700bee8d57a7a82d67f29
Author: Dave Liddament <dave@lampbristol.com>
Date:   Mon Nov 13 16:02:32 2017 +0000

    FIX typo
```

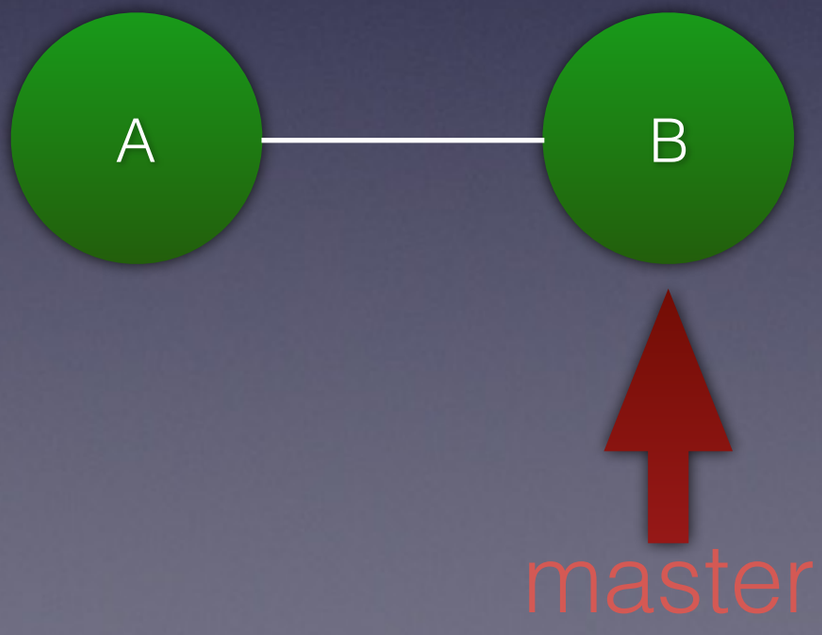
cat .git/refs/heads/master

6e88dad5d769b921d1a700bee8d57a7a82d67f29

What is a branch?

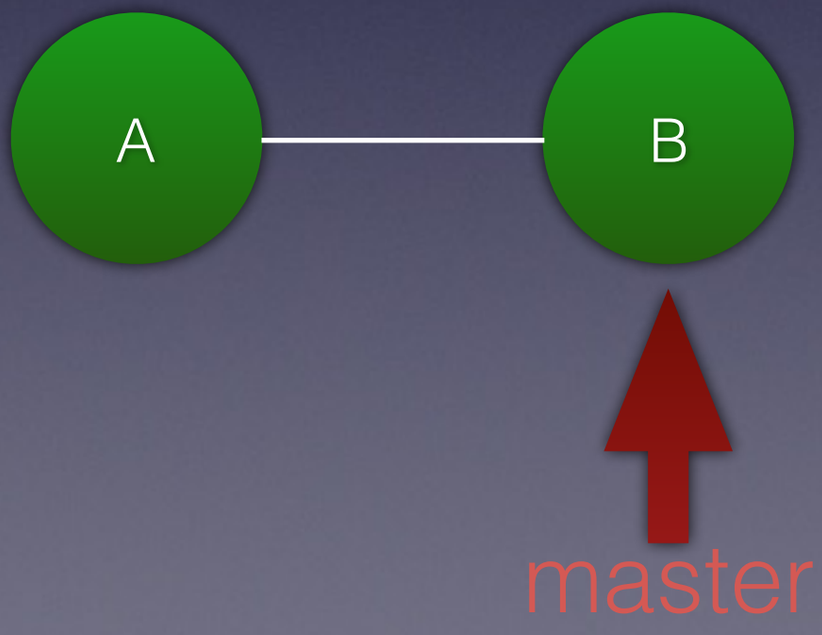


Branching



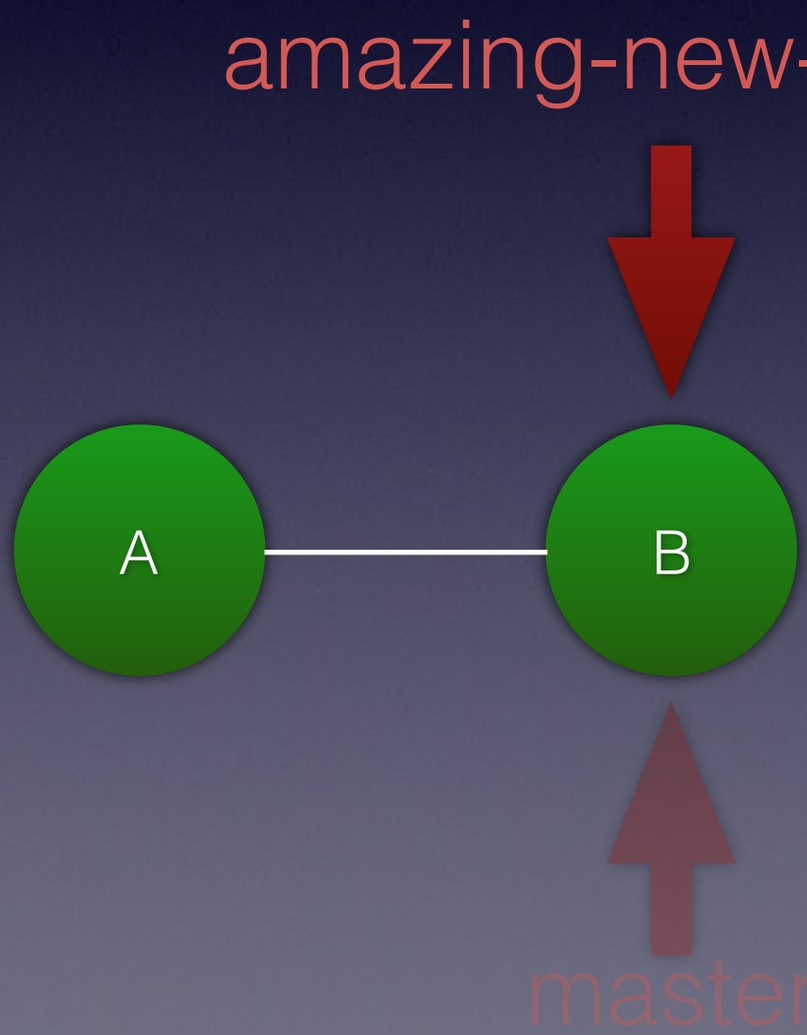
Branching

git checkout -b amazing-new-feature master

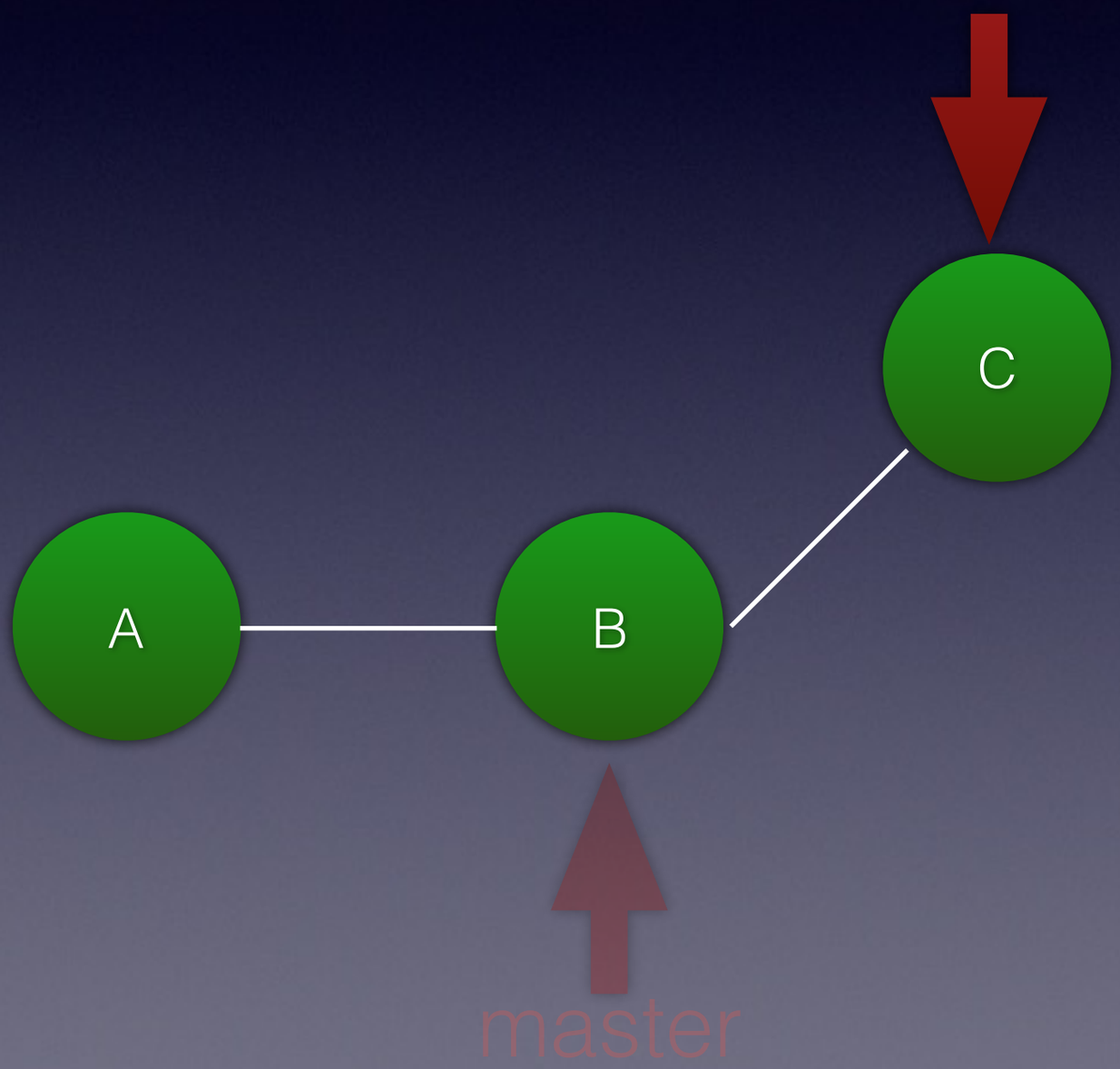


Branching

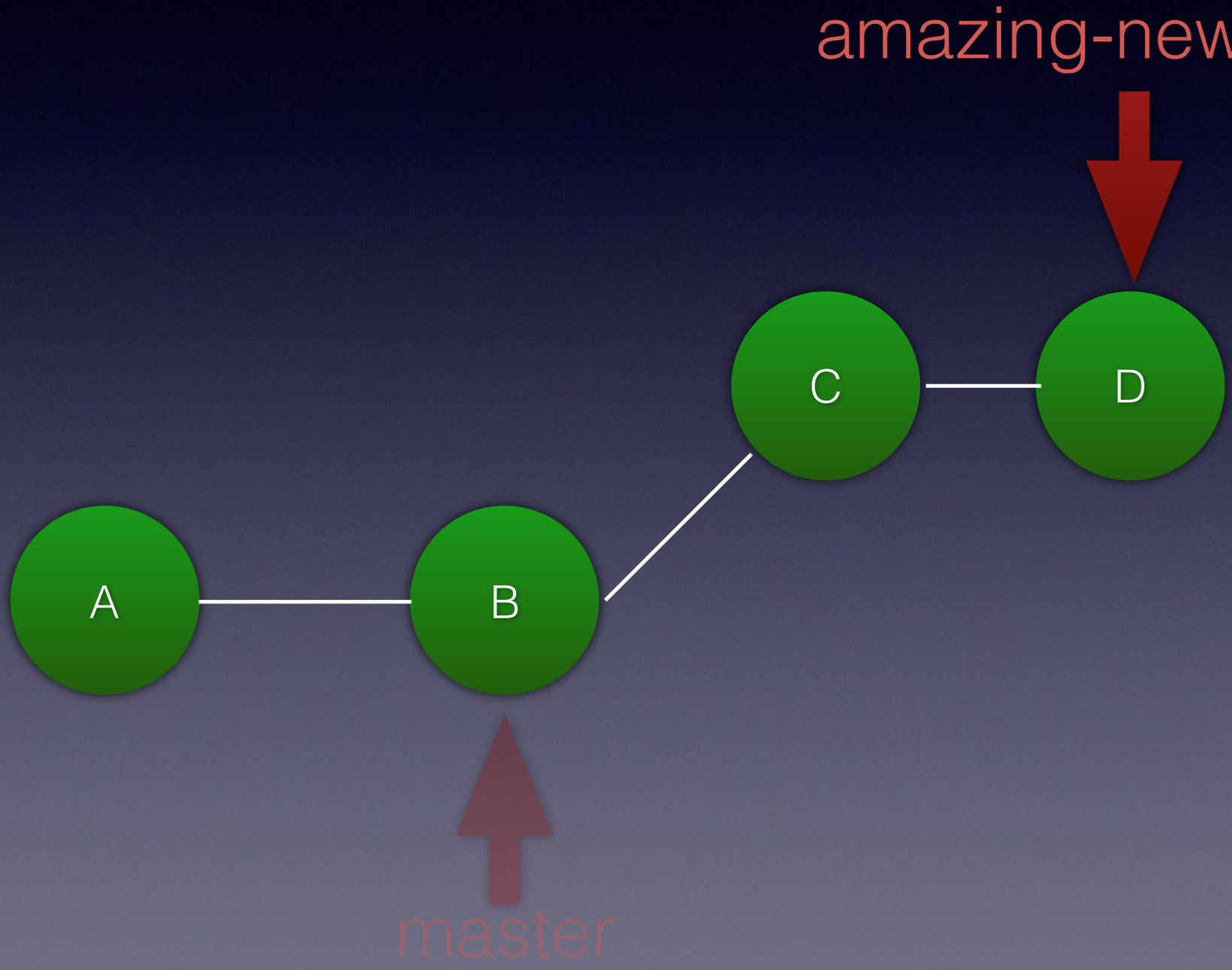
git checkout -b amazing-new-feature master



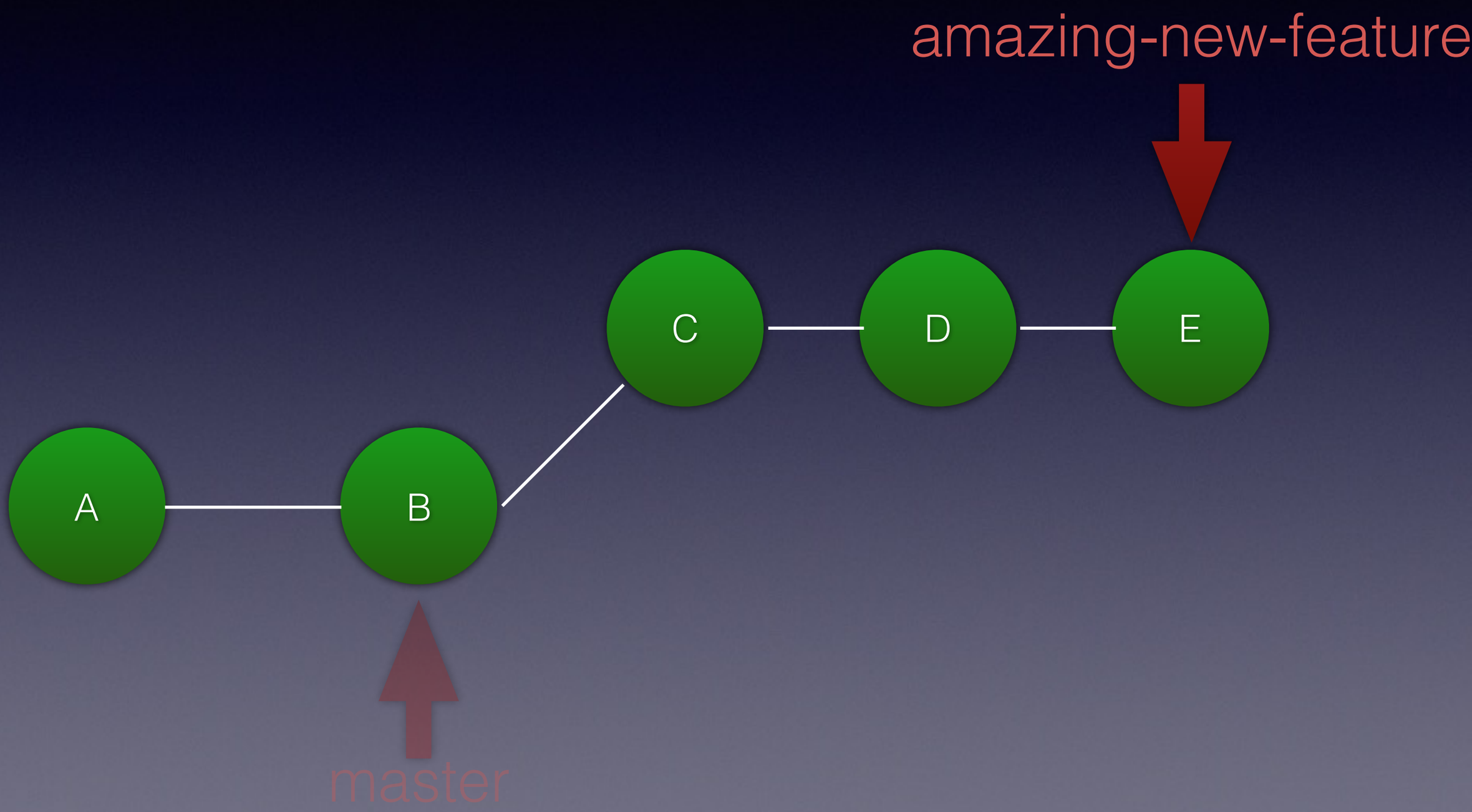
Branching



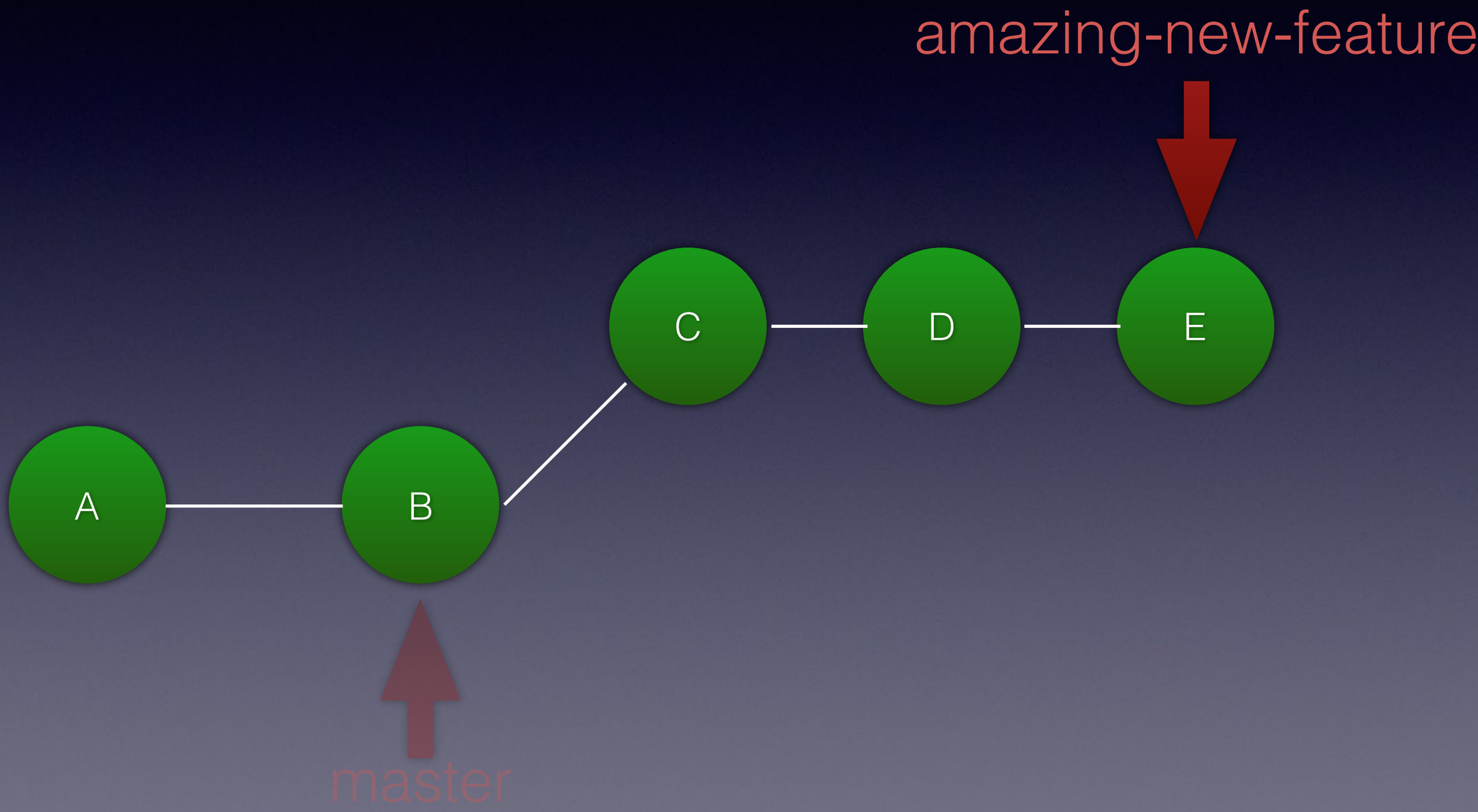
Branching



Branching



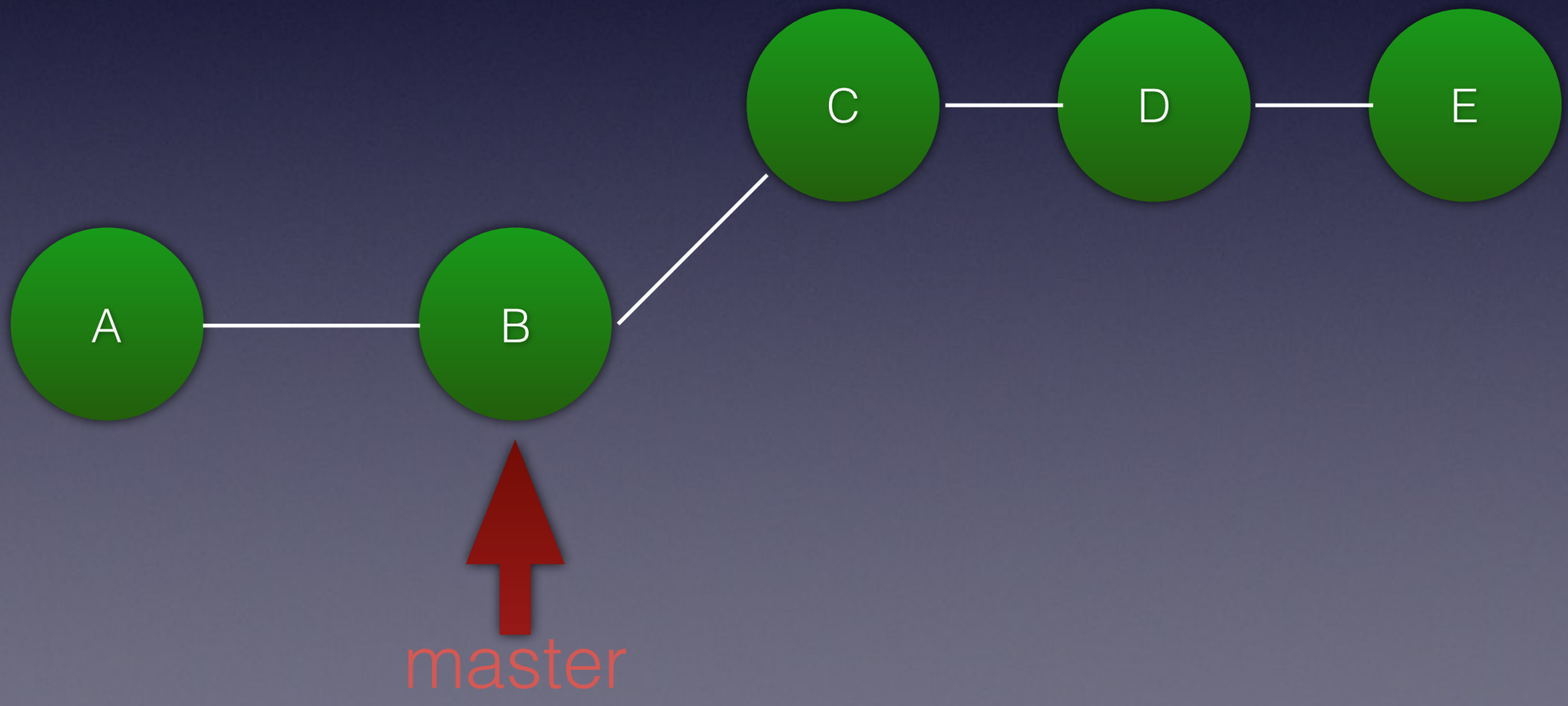
Merging



Merging

git checkout master

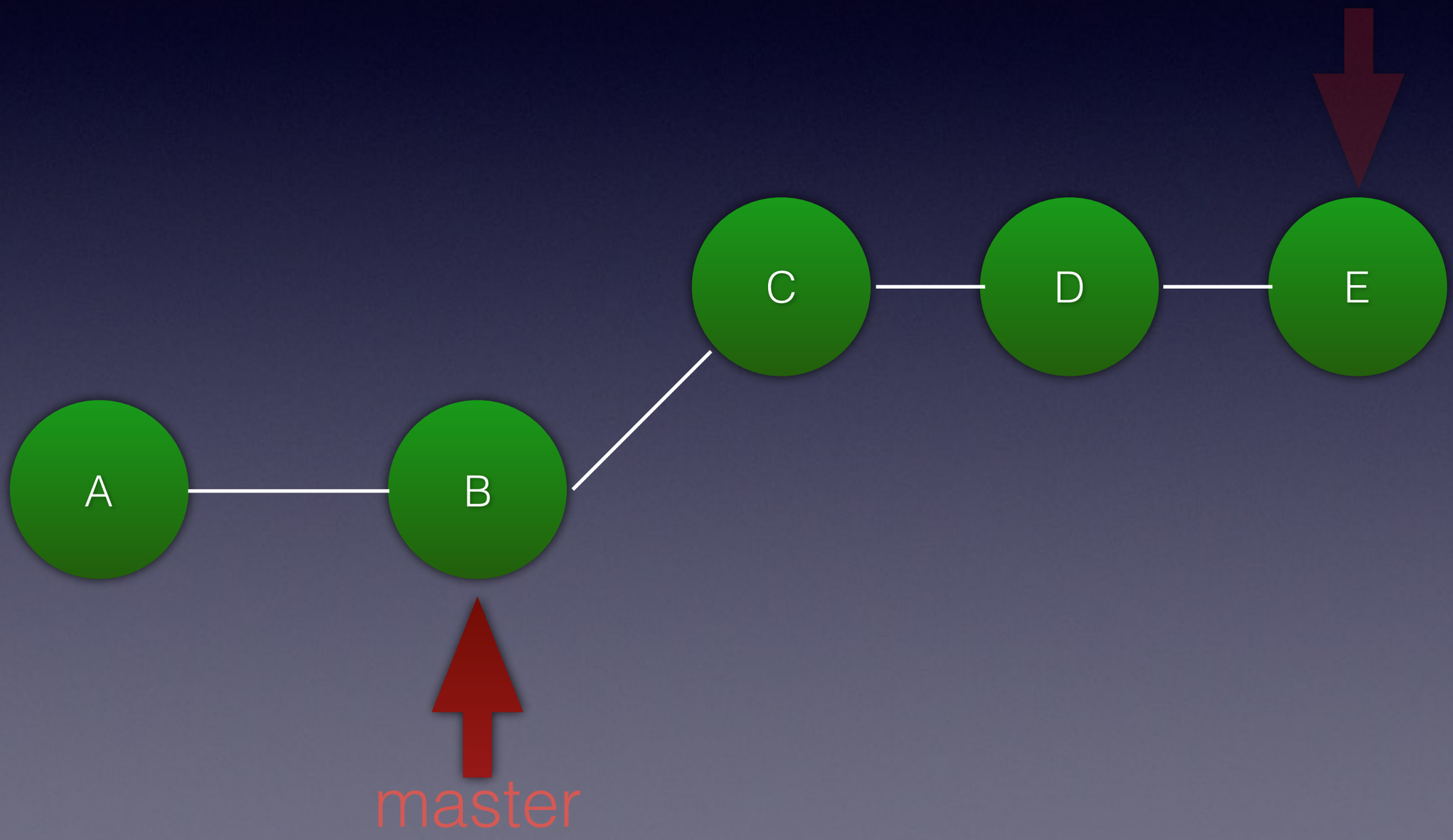
amazing-new-feature



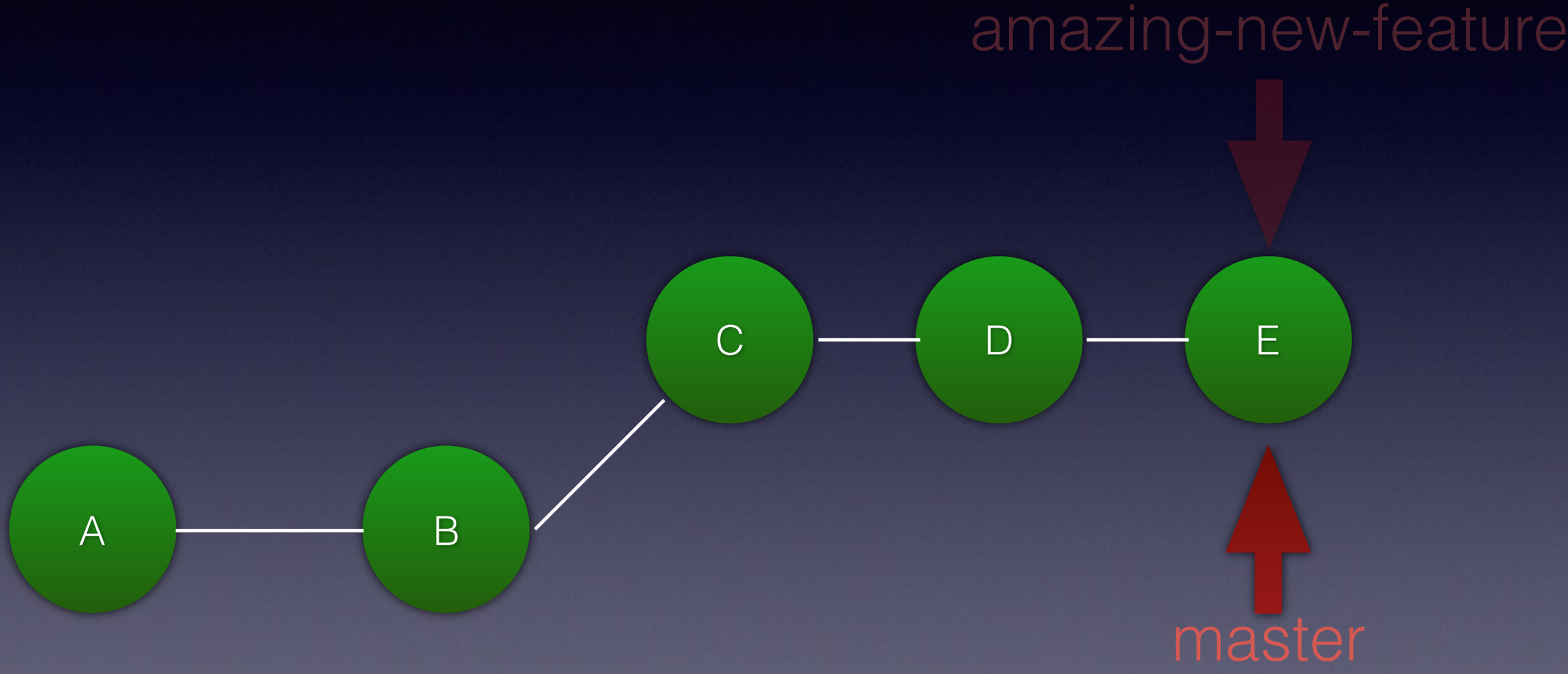
Merging

git merge amazing-new-feature

amazing-new-feature

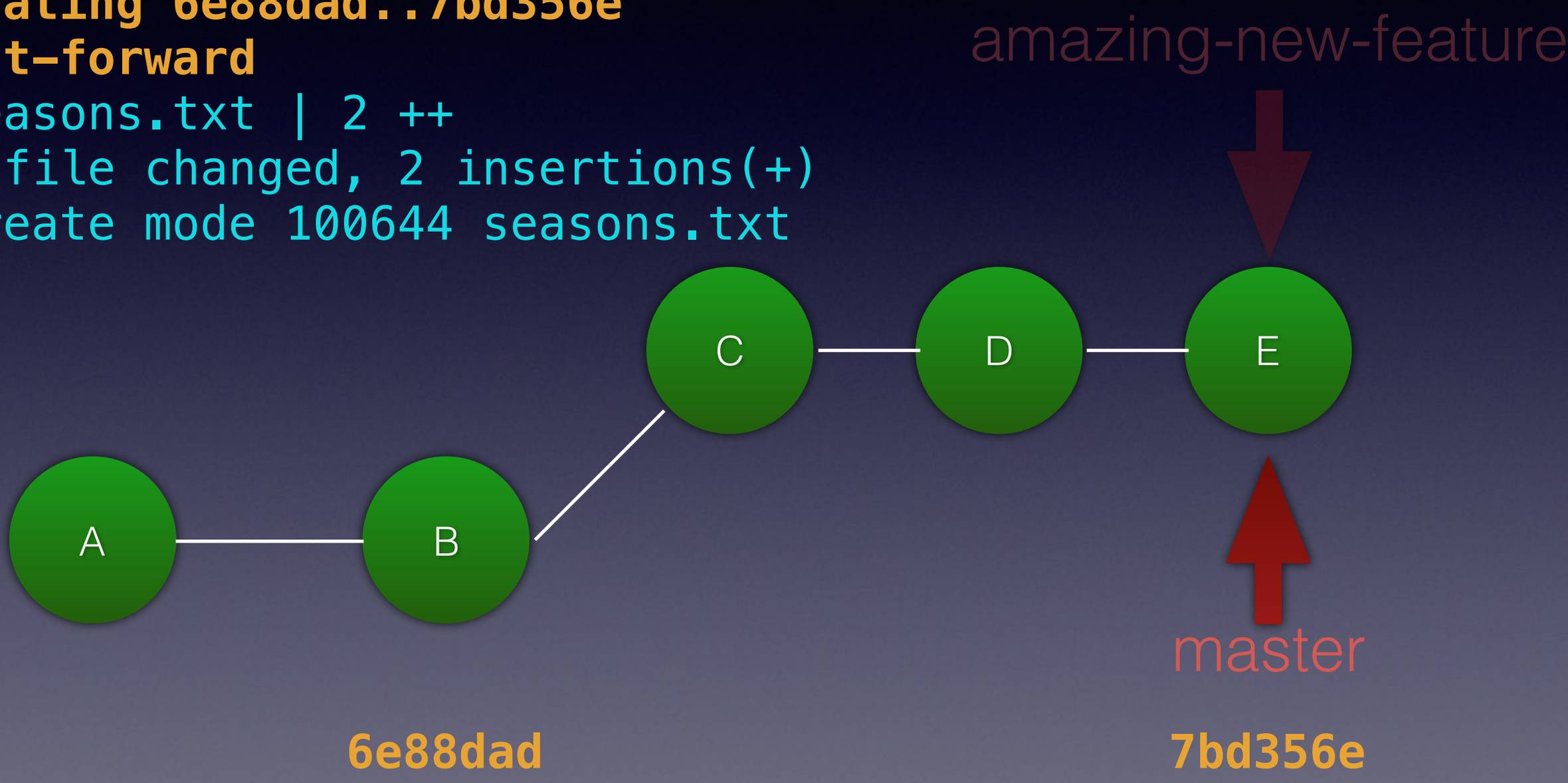


Merging

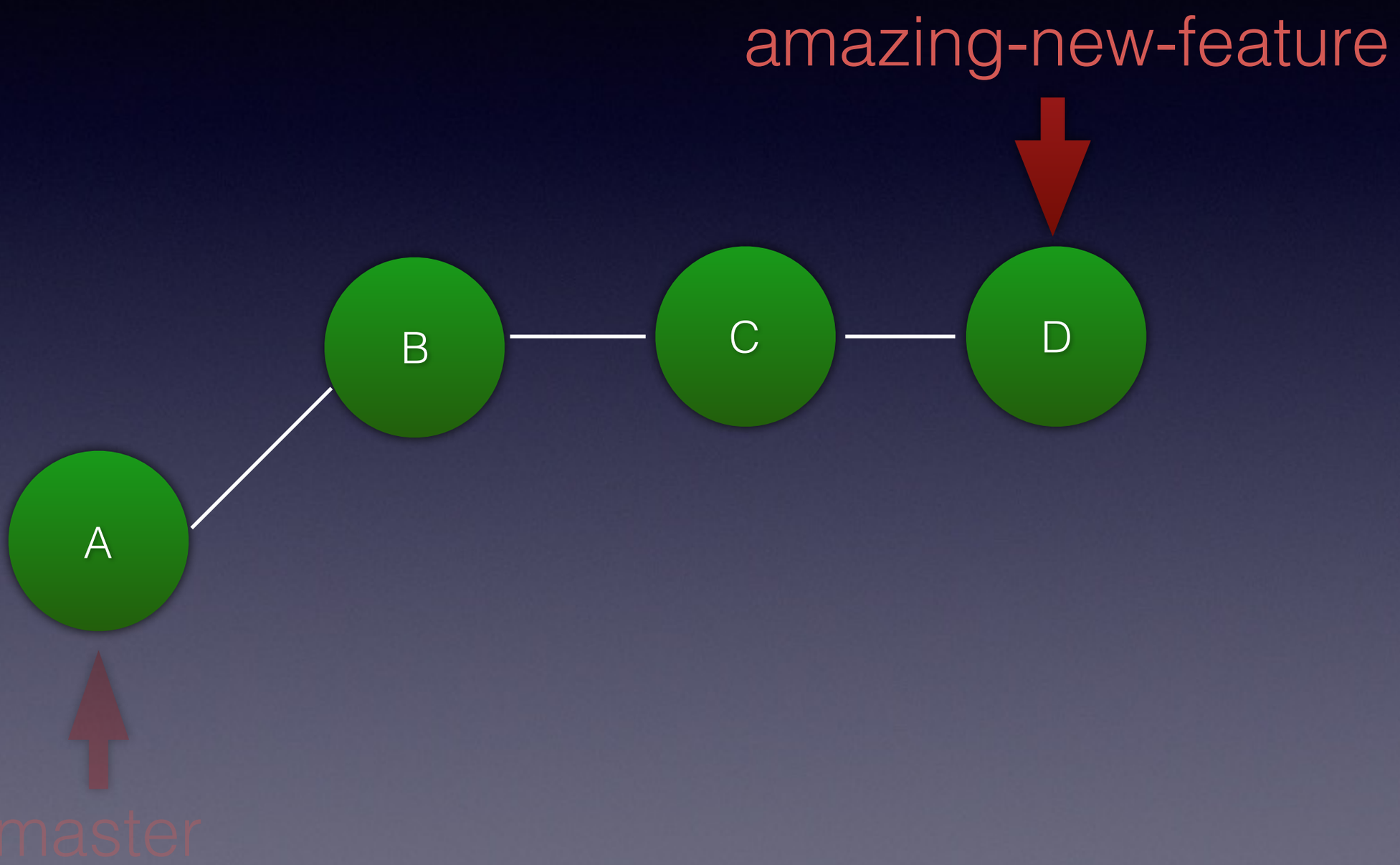


Merging

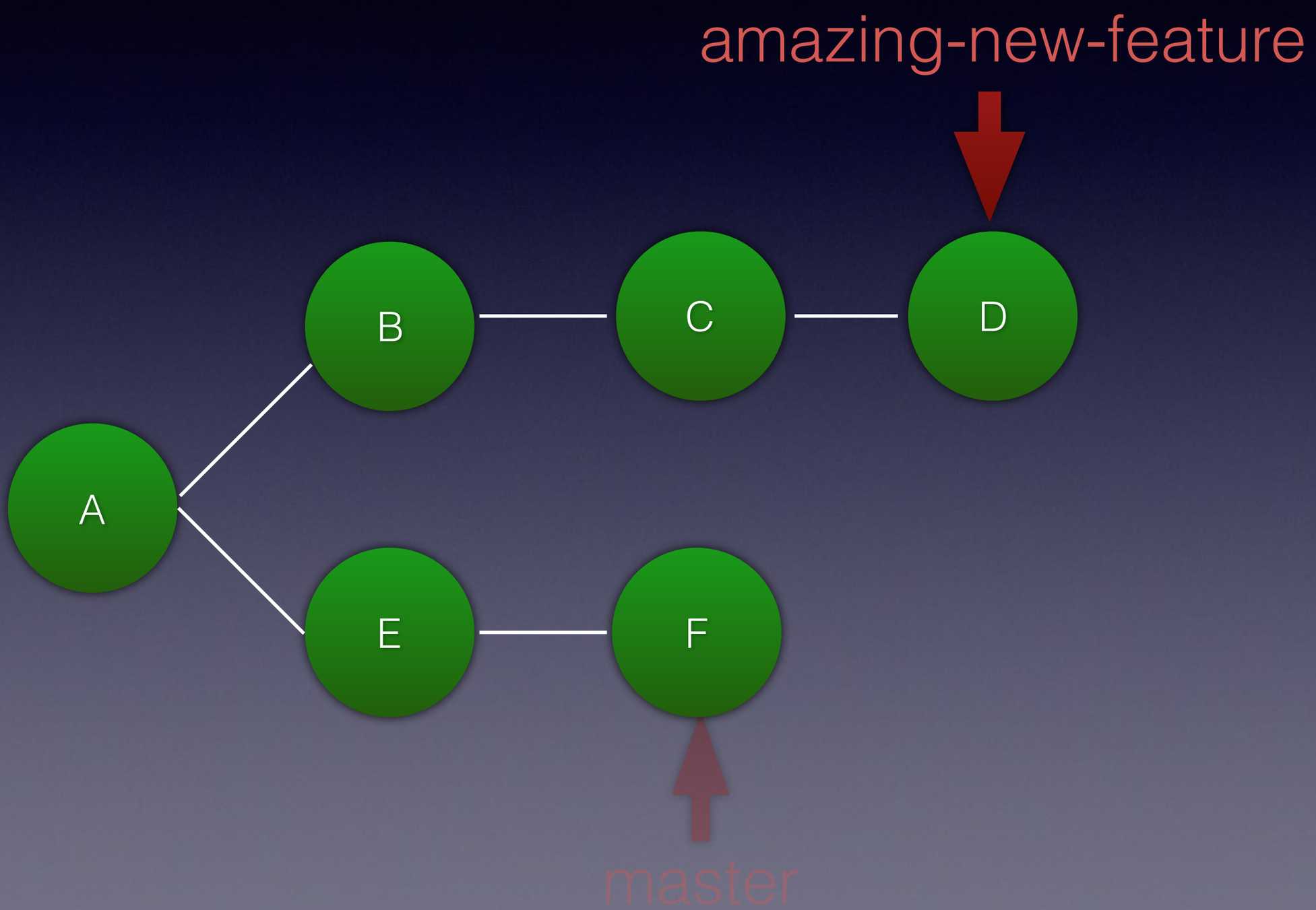
```
Updating 6e88dad..7bd356e
Fast-forward
 seasons.txt | 2 ++
 1 file changed, 2 insertions(+)
 create mode 100644 seasons.txt
```



Merging (2)



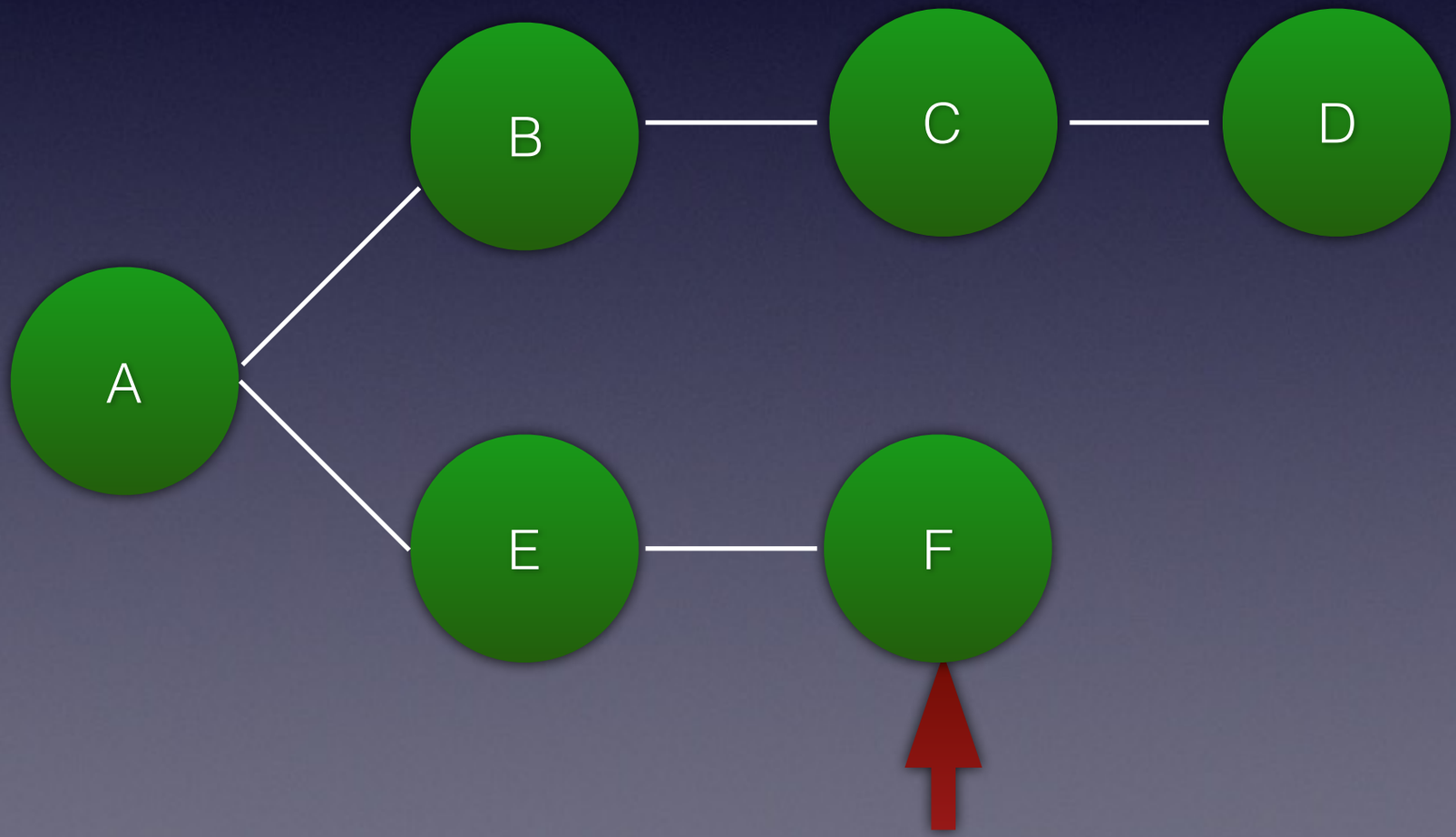
Merging (2)



Merging (2)

amazing-new-feature

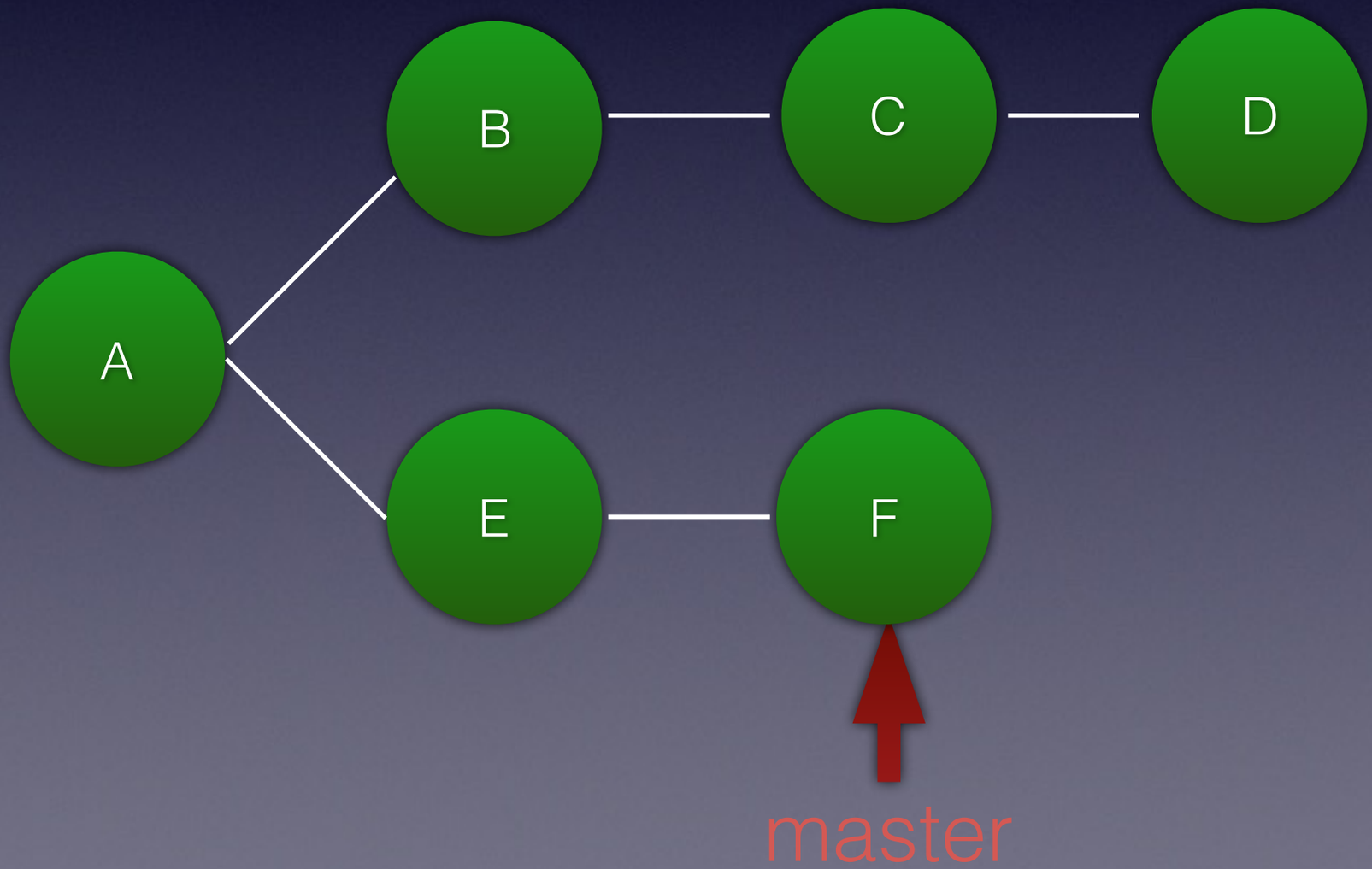
git checkout master



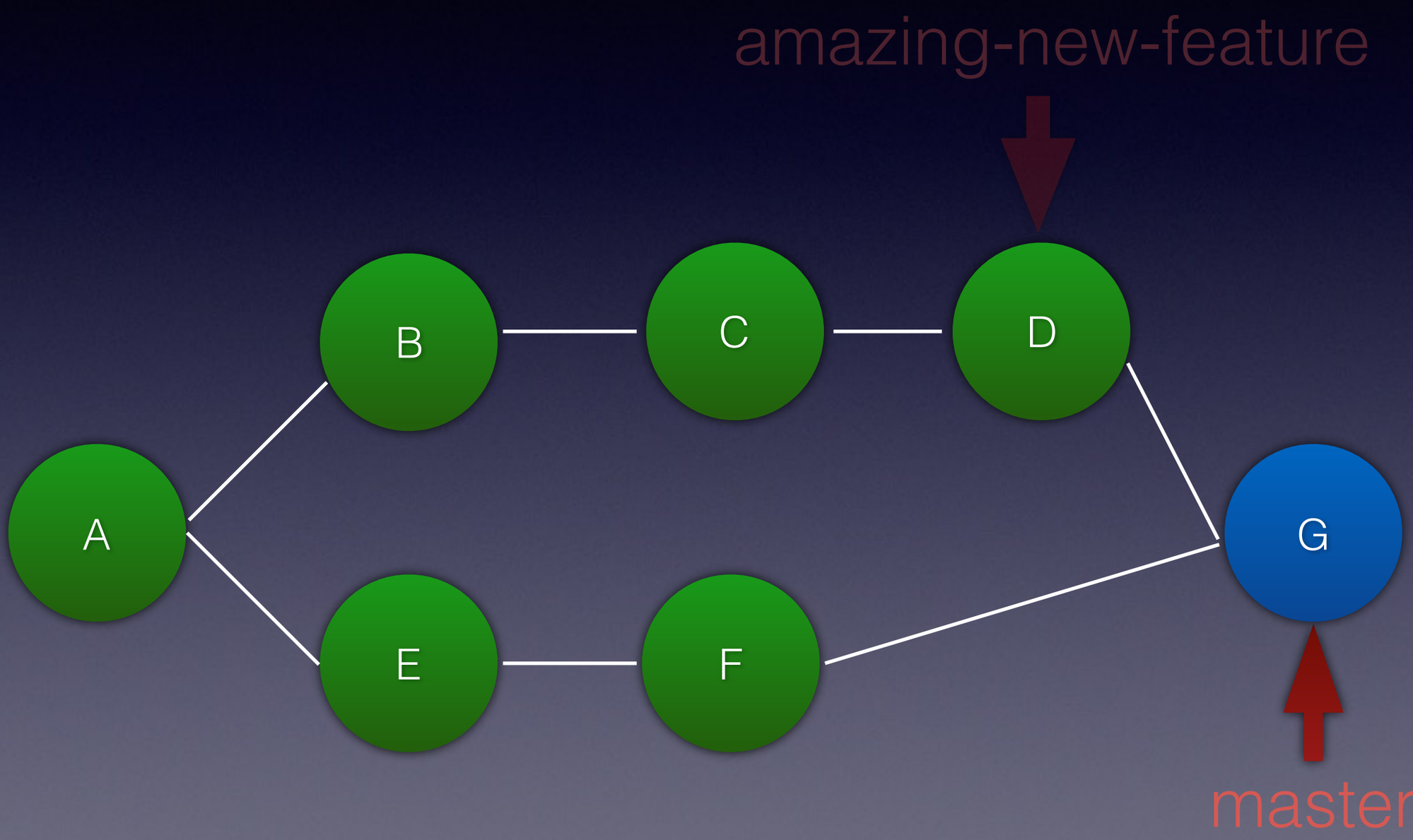
Merging (2)

amazing-new-feature

git merge amazing-new-feature



Merging (2)

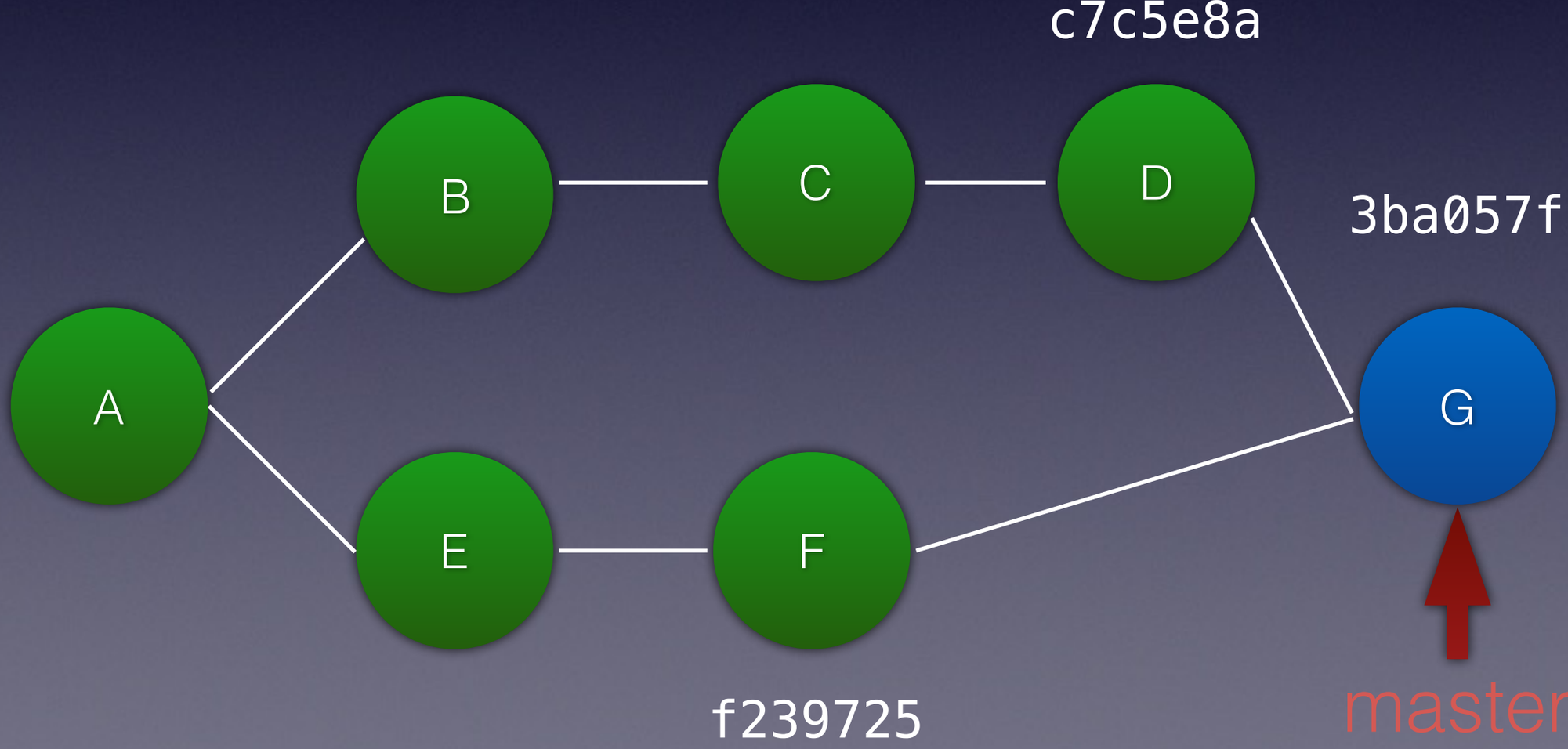


What is a commit?

1. Metadata
2. Patch
3. Parent commits(s)

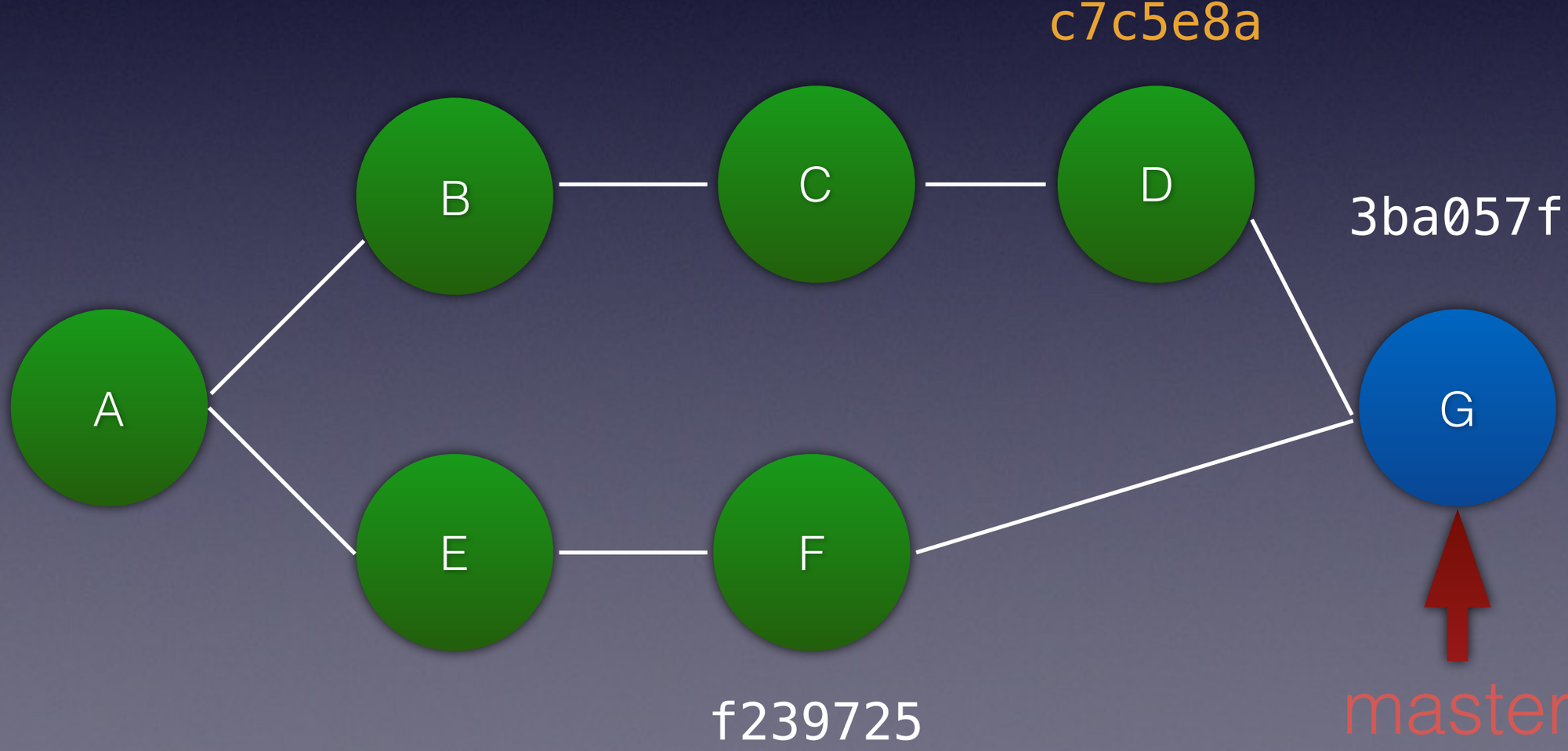
Merging (2)

```
git log

commit 3ba057fab8af25a9345a63e63690d8219cfe4b46
Merge: c7c5e8a f239725
Author: Dave Liddament <dave@lampbristol.com>
…
```



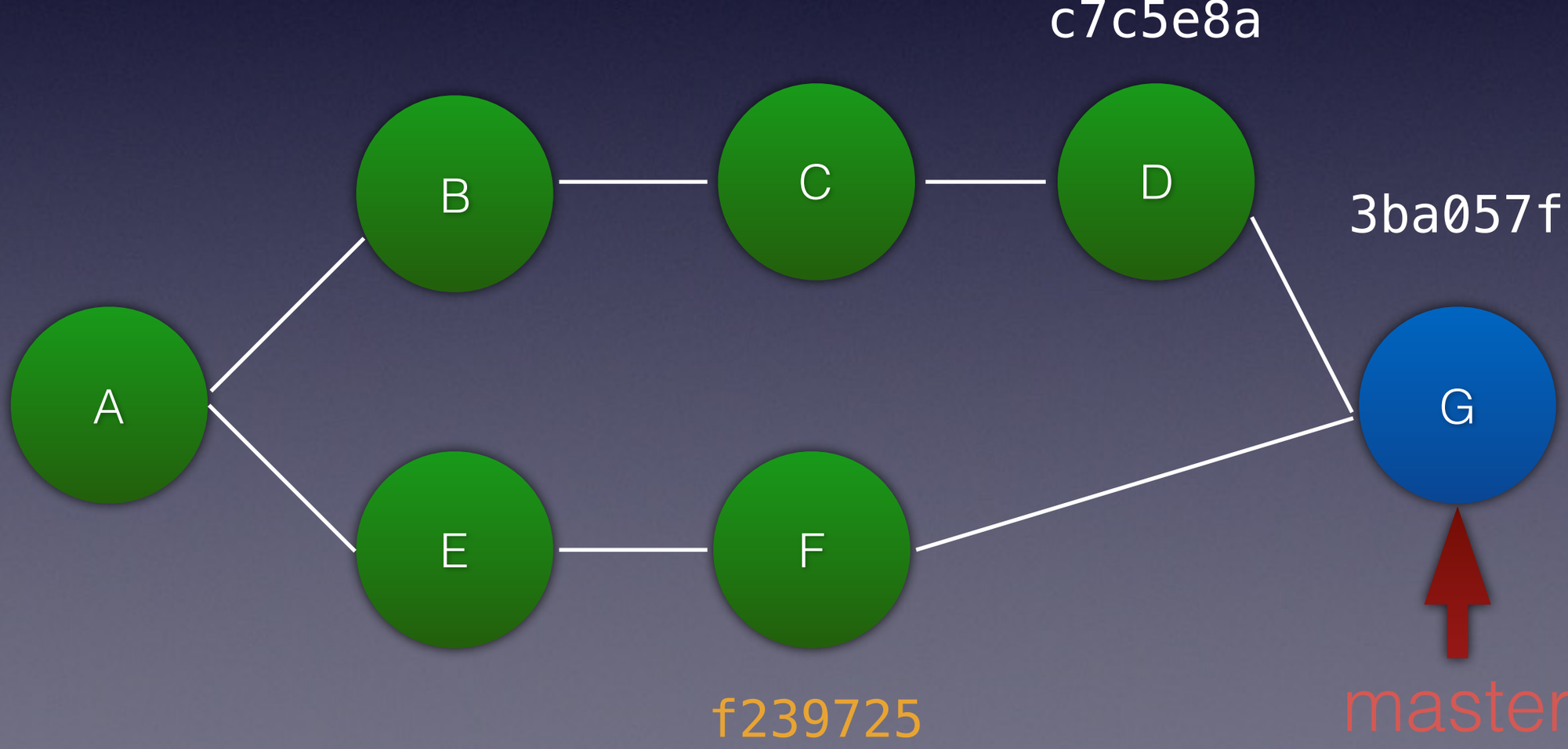
Merging (2)

```
git log

commit 3ba057fab8af25a9345a63e63690d8219cfe4b46
Merge: c7c5e8a f239725
Author: Dave Liddament <dave@lampbristol.com>
…
```



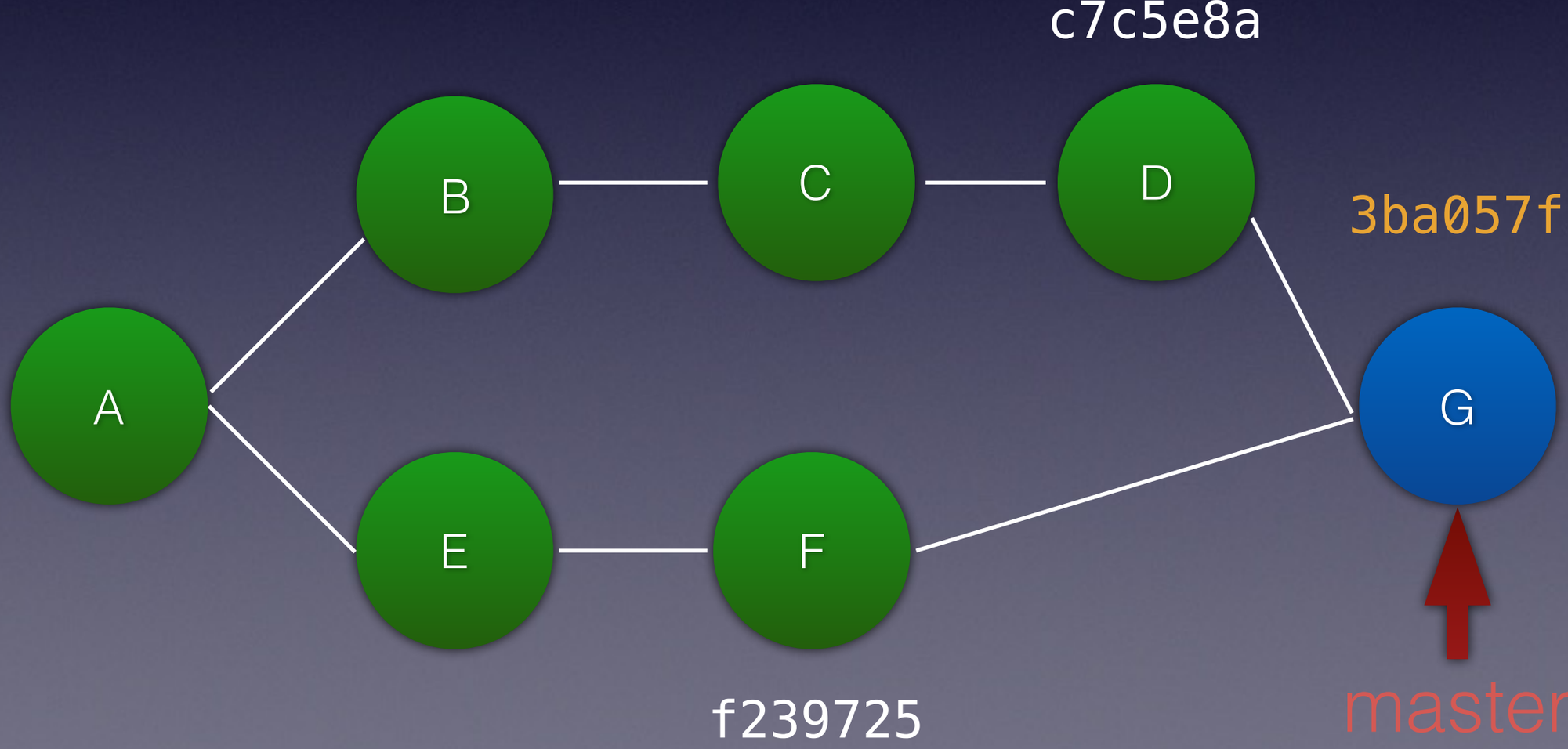
Merging (2)

```
git log

commit 3ba057fab8af25a9345a63e63690d8219cfe4b46
Merge: c7c5e8a f239725
Author: Dave Liddament <dave@lampbristol.com>
…
```



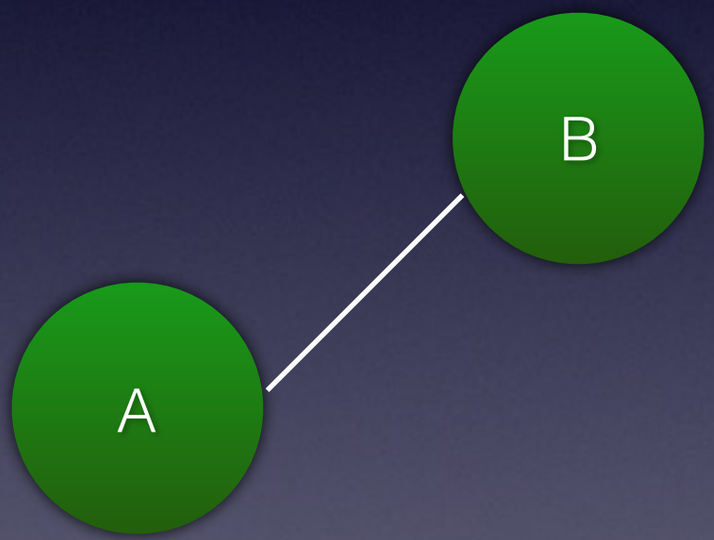
Merging (2)

```
git log

commit 3ba057fab8af25a9345a63e63690d8219cfe4b46
Merge: c7c5e8a f239725
Author: Dave Liddament <dave@lampbristol.com>
…
```



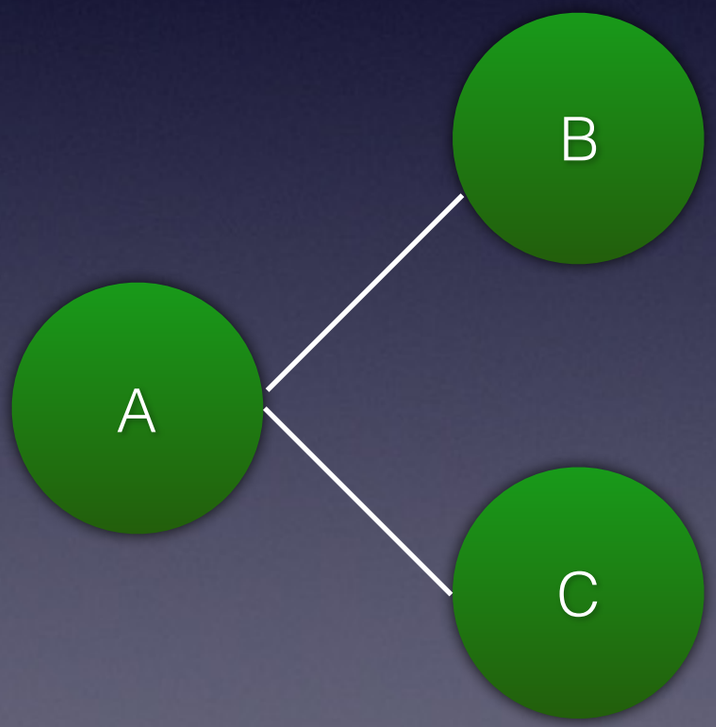
Merge conflicts

```
days.txt
 Monday
-tuseday
+Tuseday
 Wednesday
```



Merge conflicts

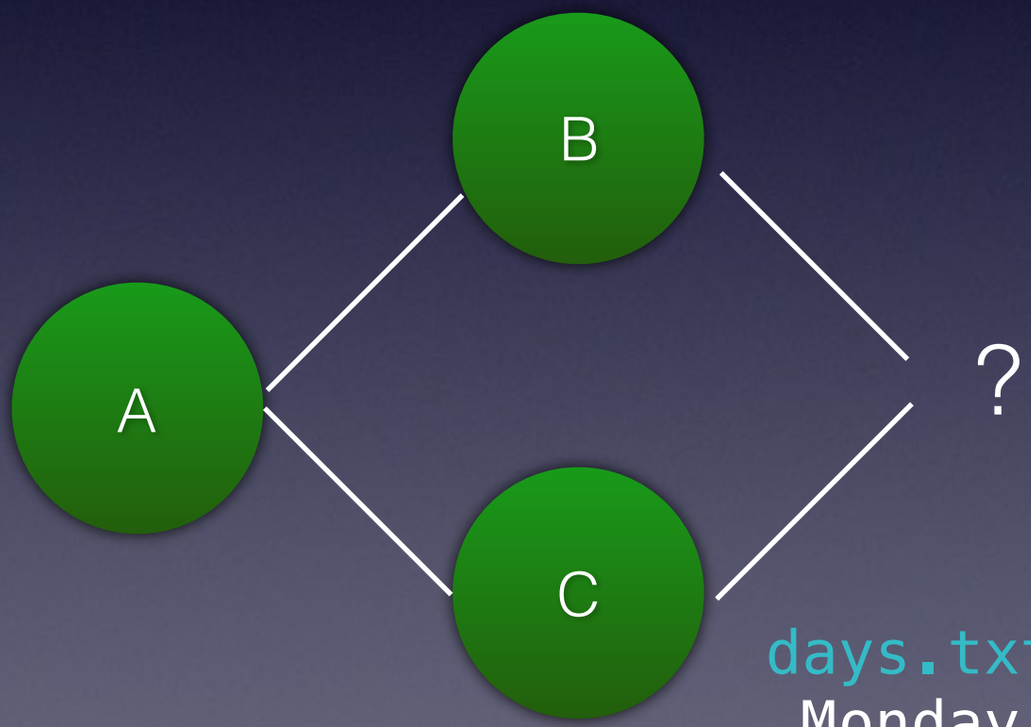
```
days.txt
 Monday
-tuseday
+Tuseday
 Wednesday
```




```
days.txt
 Monday
-tuesday
+tuesday
 Wednesday
```

Merge conflicts

```
days.txt
 Monday
-tuseday
+Tuseday
 Wednesday
```




```
days.txt
 Monday
-tuesday
+tuesday
 Wednesday
```

Merge conflicts

```
Auto-merging days
CONFLICT (content): Merge conflict in days.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Merge conflicts

```
cat days.txt
```

```
Monday
<<<<<<< HEAD
Tuseday
=======
tuesday
>>>>>>> fix1
Wednesday
```

Merge conflicts

```
cat days.txt
```

```
Monday
Tuesday
Wednesday
```

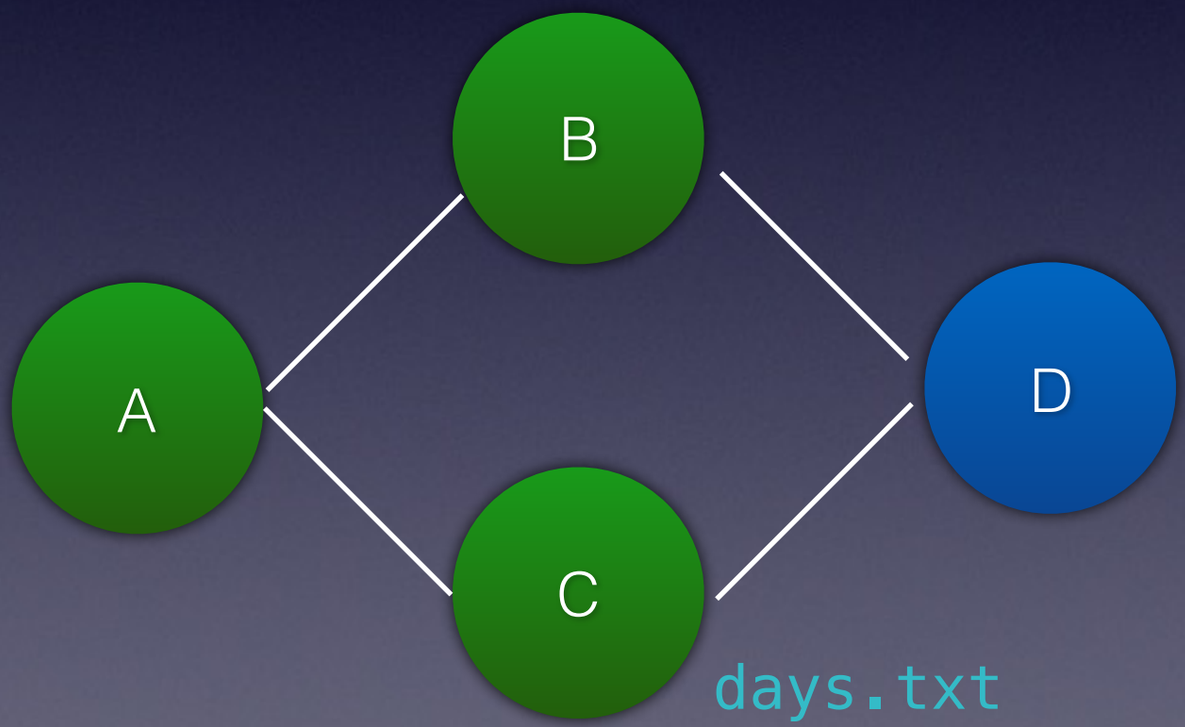
Merge conflicts

```
git add days.txt
```

```
git commit
```

Merge conflicts

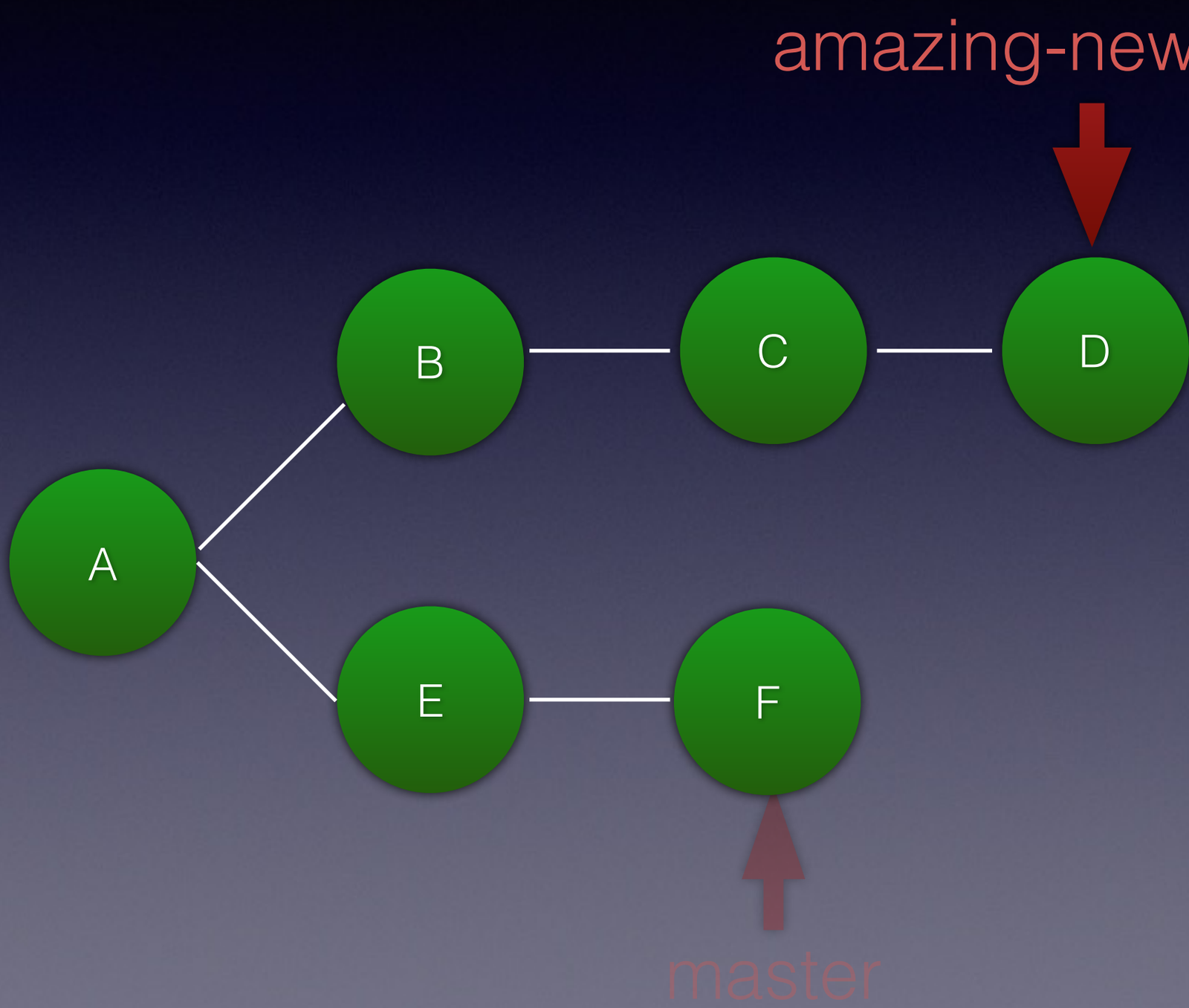
```
days.txt
 Monday
-tuseday
+Tuseday
 Wednesday
```



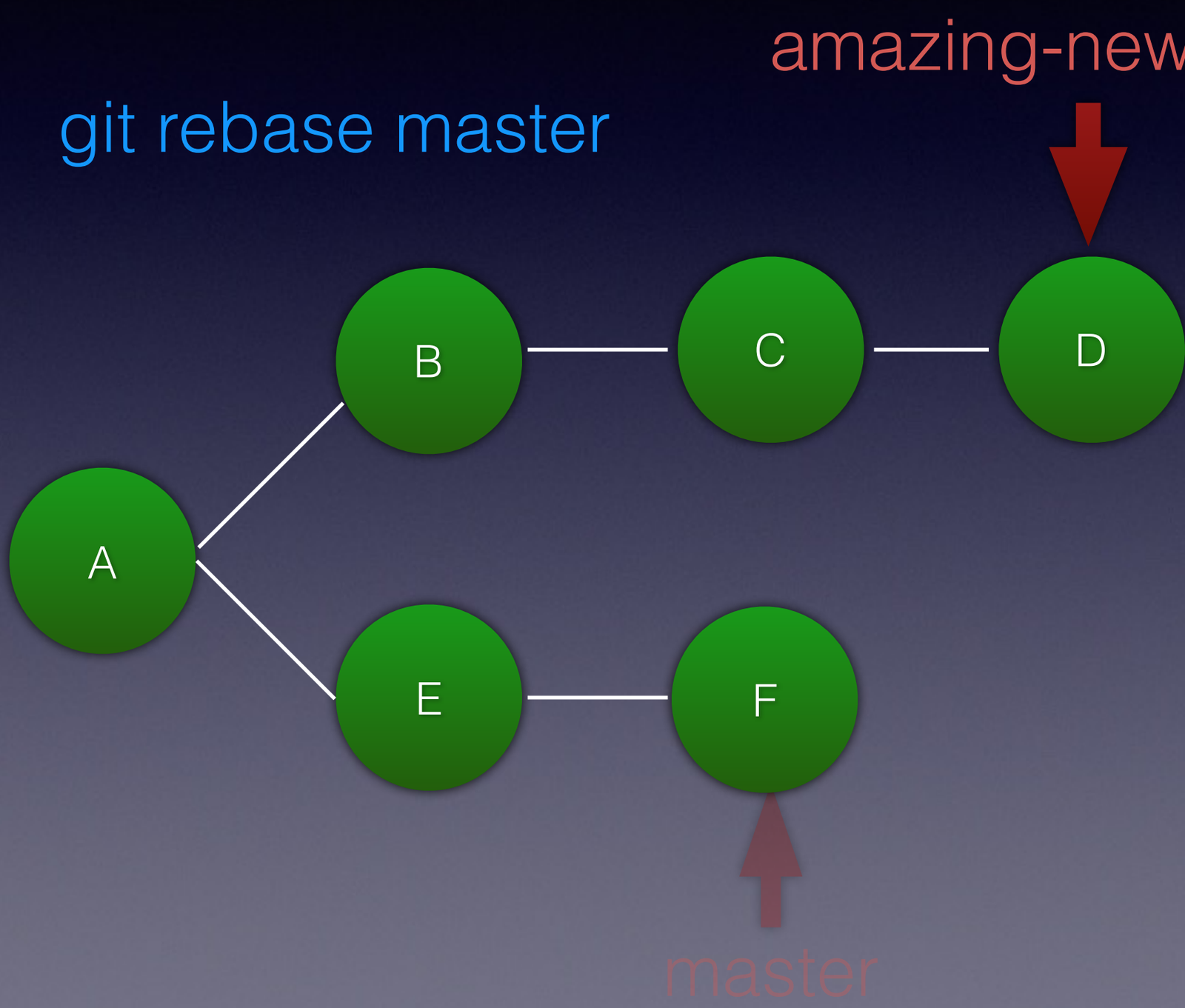

```
days.txt
 Monday
 Tuesday
 Wednesday
```

```
days.txt
 Monday
-tuseday
+tuesday
 Wednesday
```

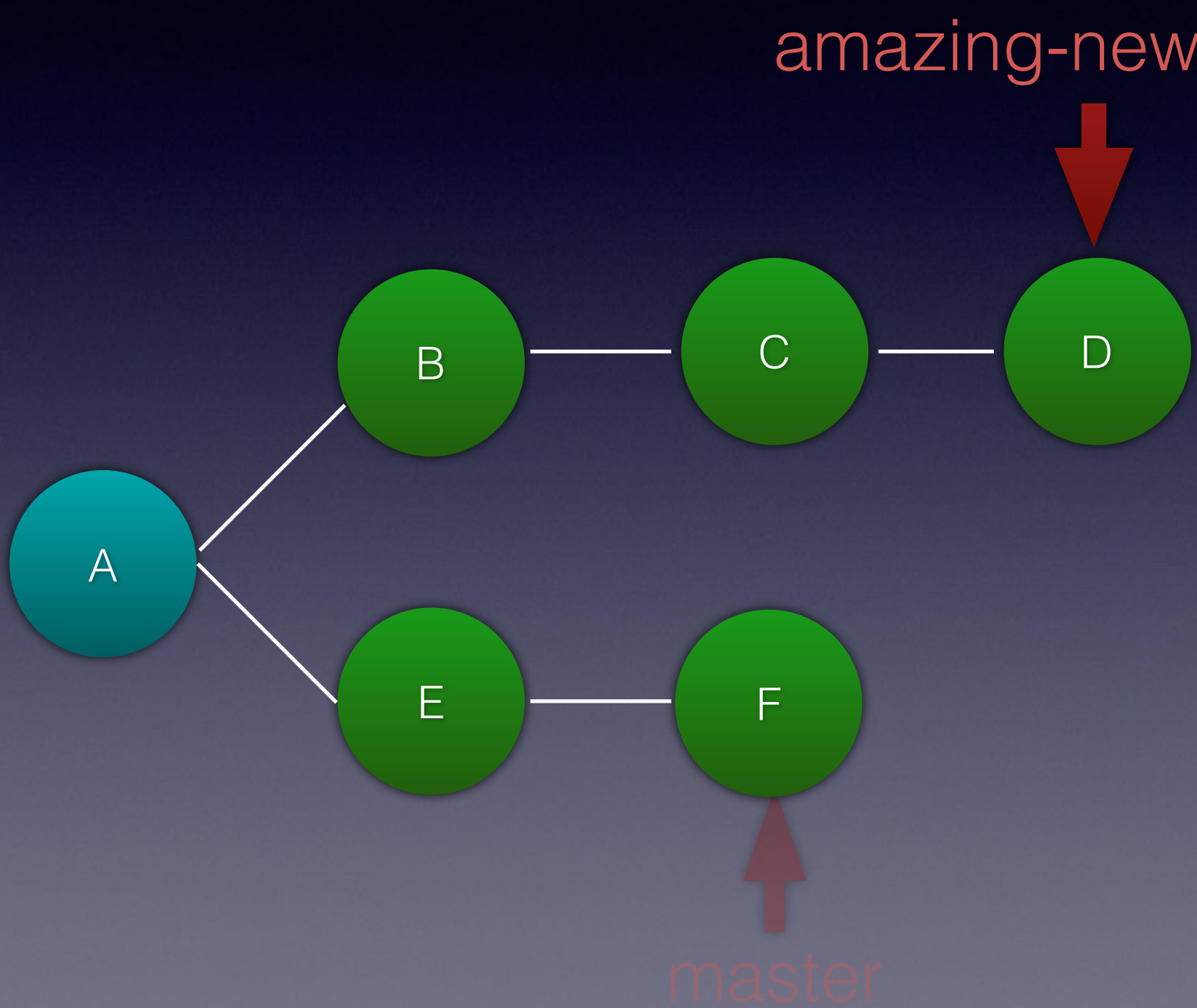
Rebase



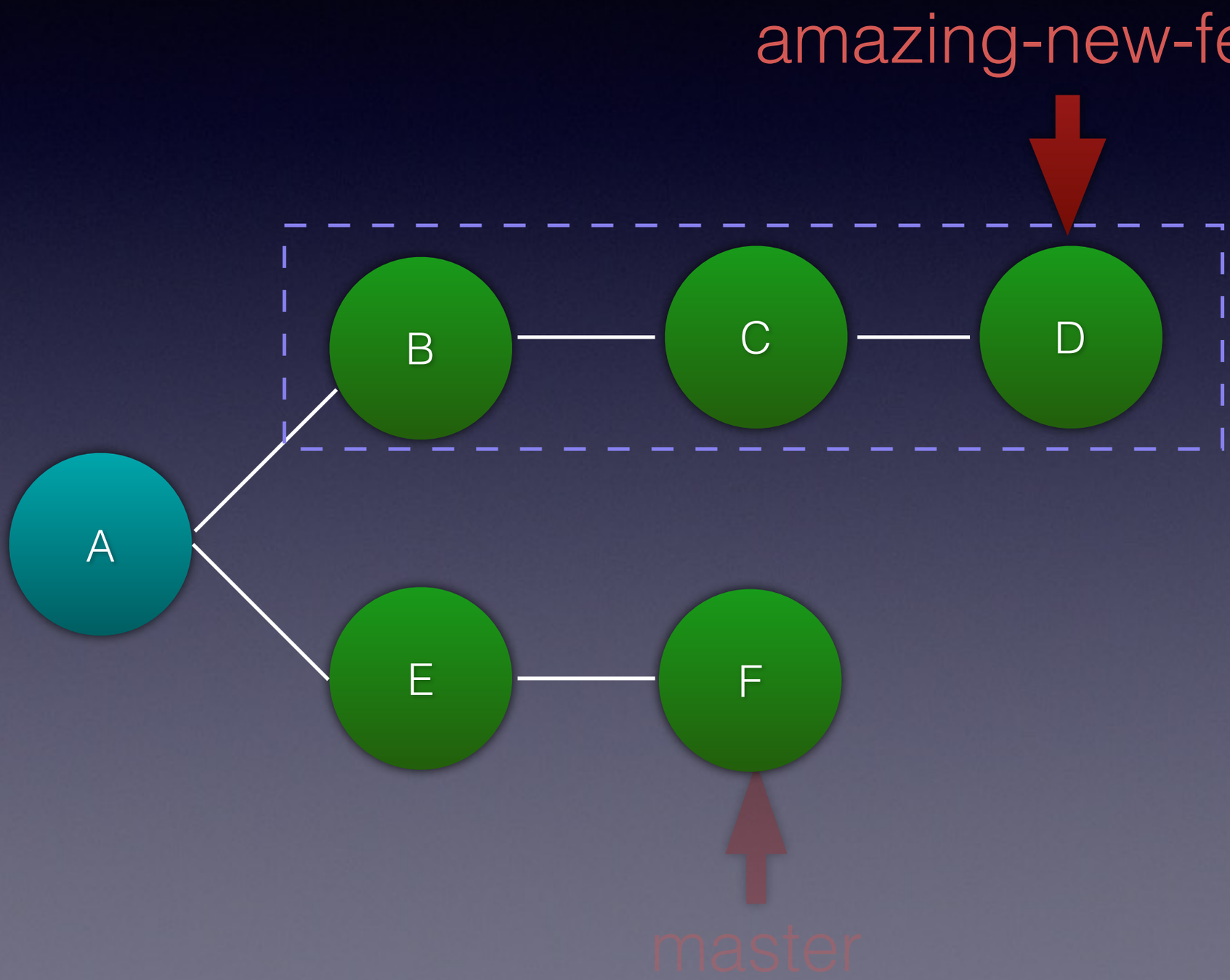
Rebase



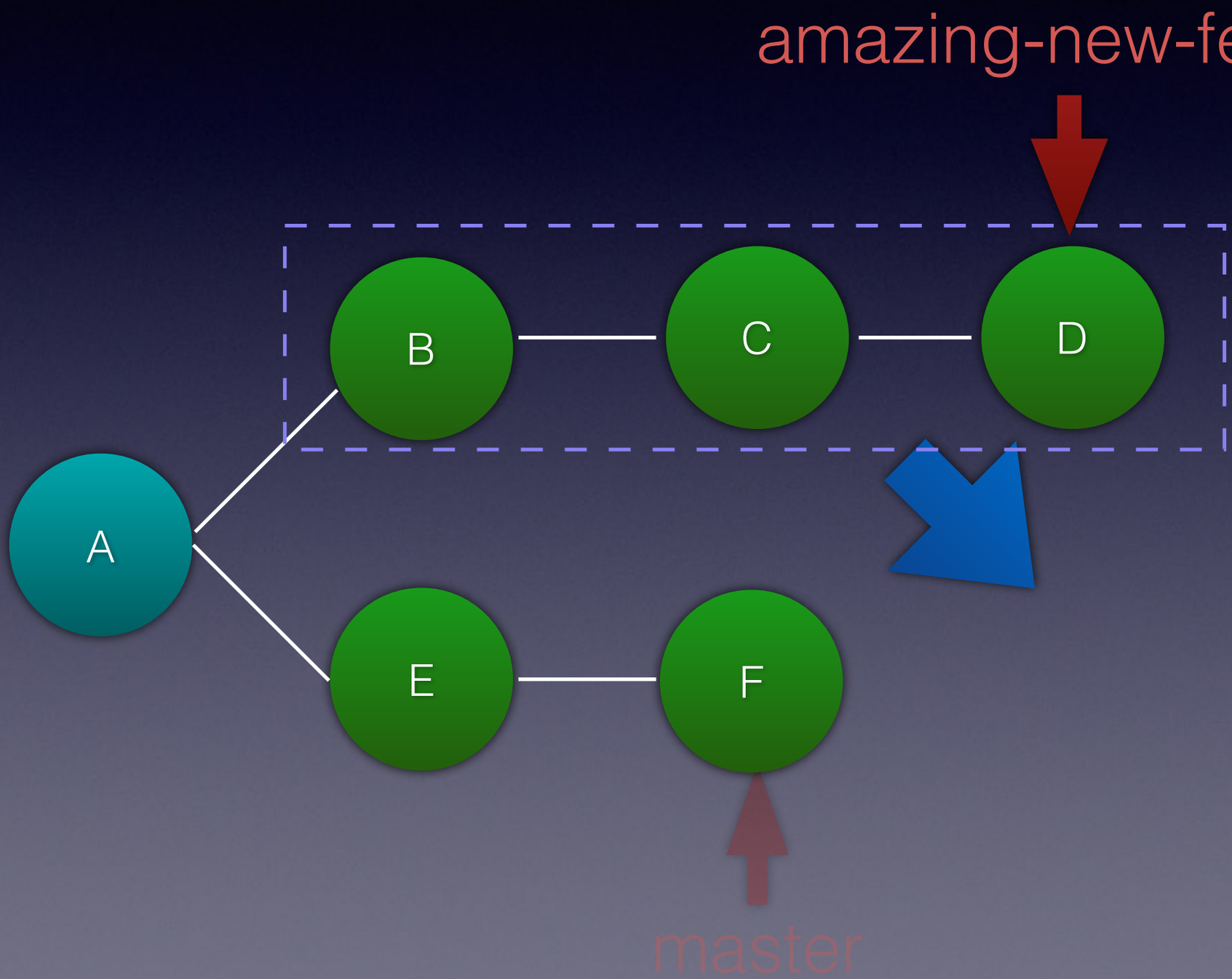
Rebase



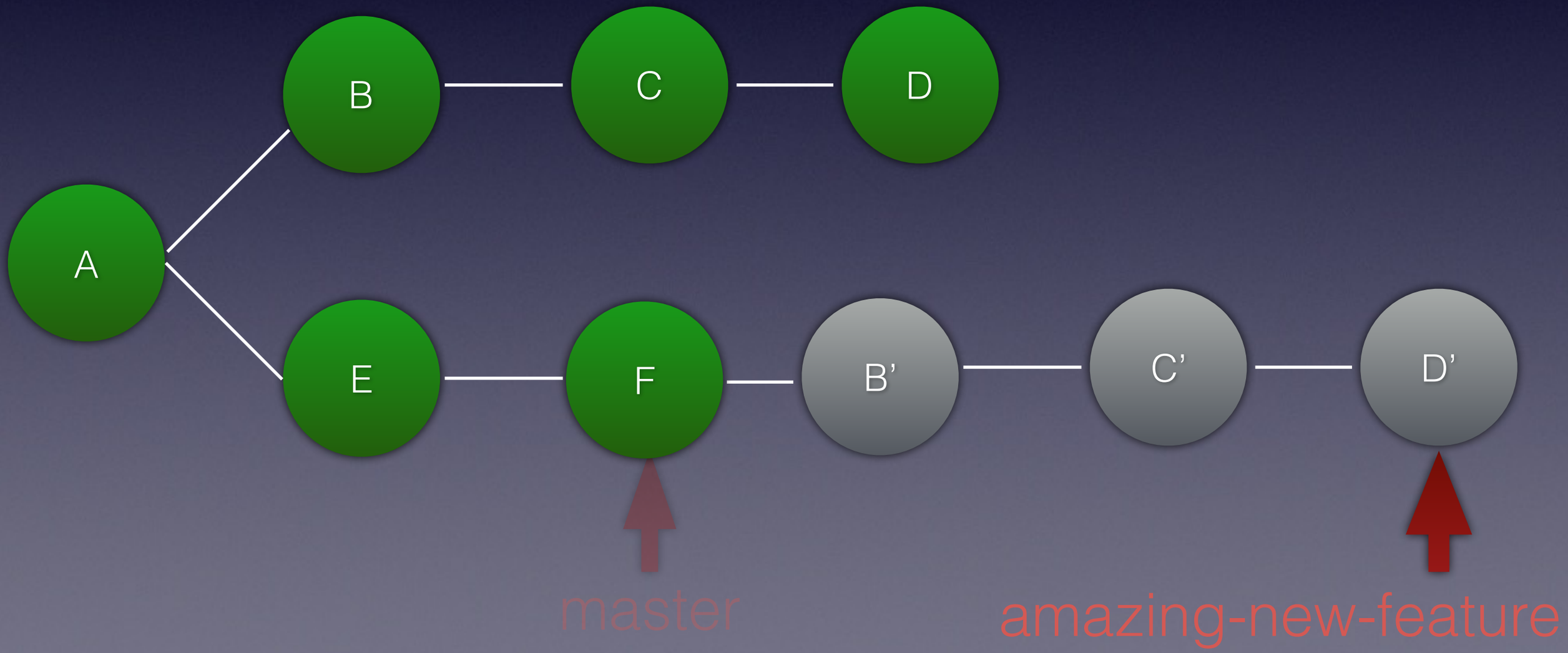
Rebase



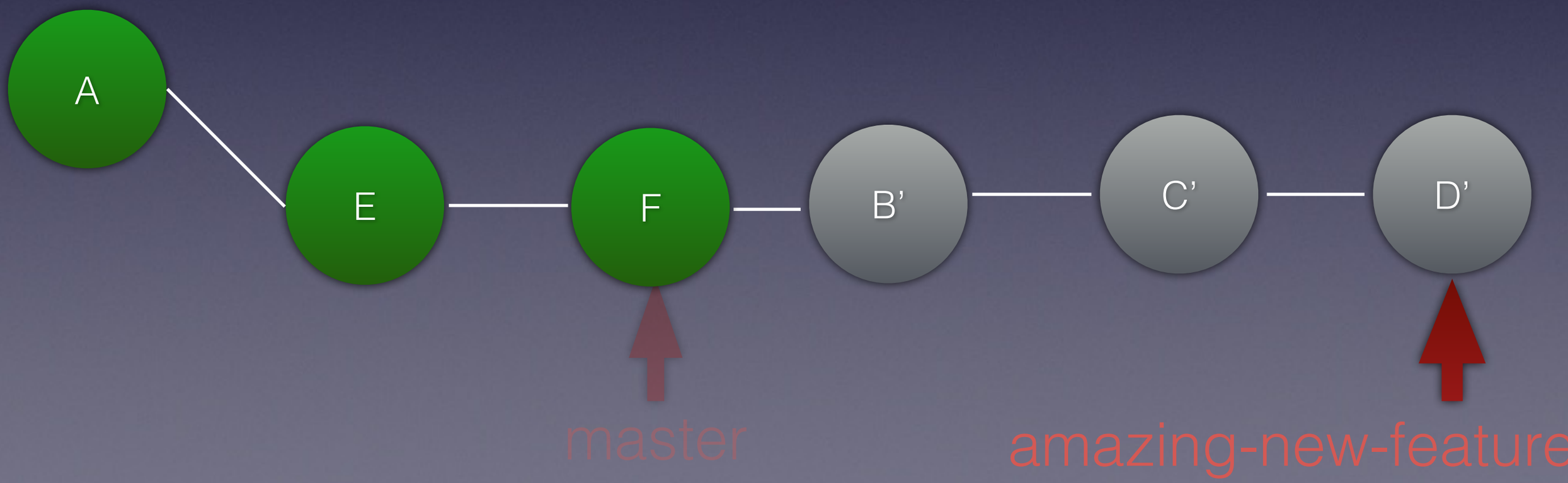
Rebase



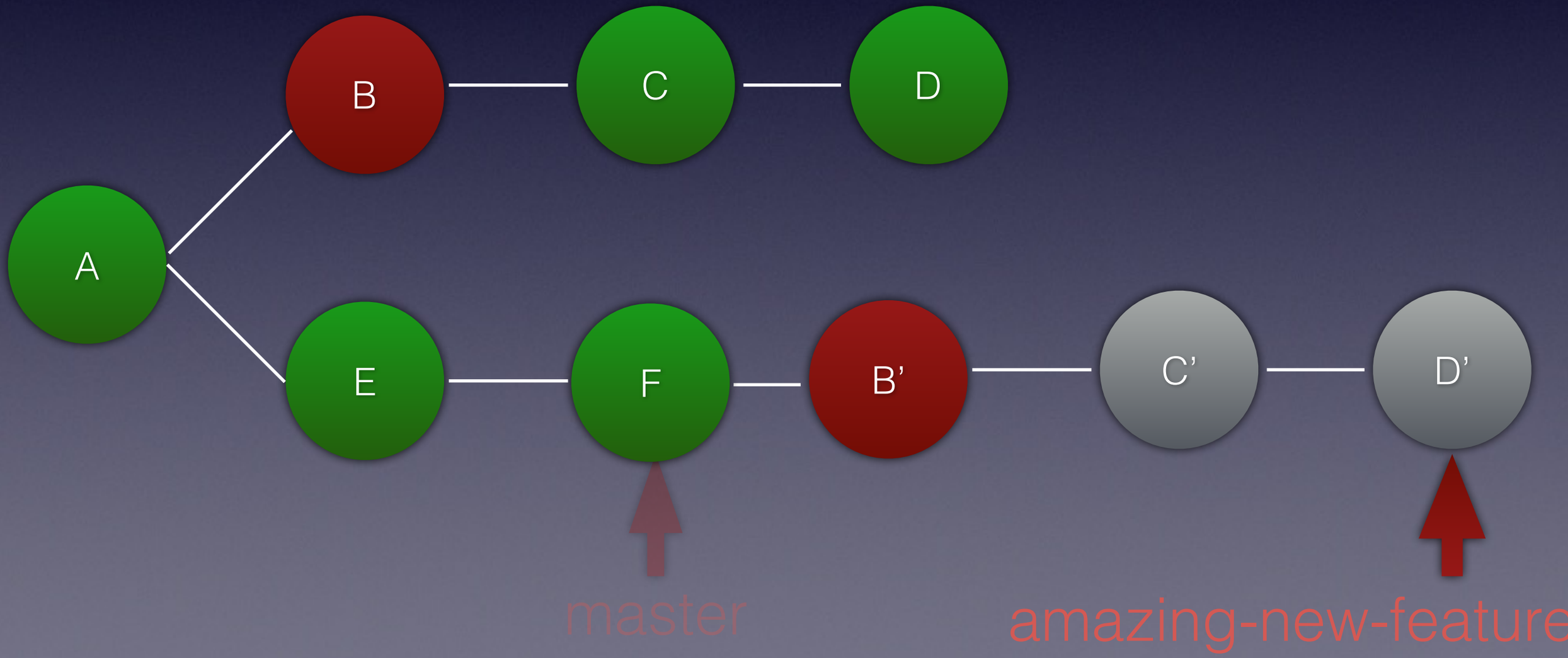
Rebase



Rebase



Rebase



What is a commit?

1. Metadata
2. Patch
3. Parent commits(s)

What is a commit?

1. Metadata
2. Patch
3. Parent commits(s)

B parent is A
B’ parent is F

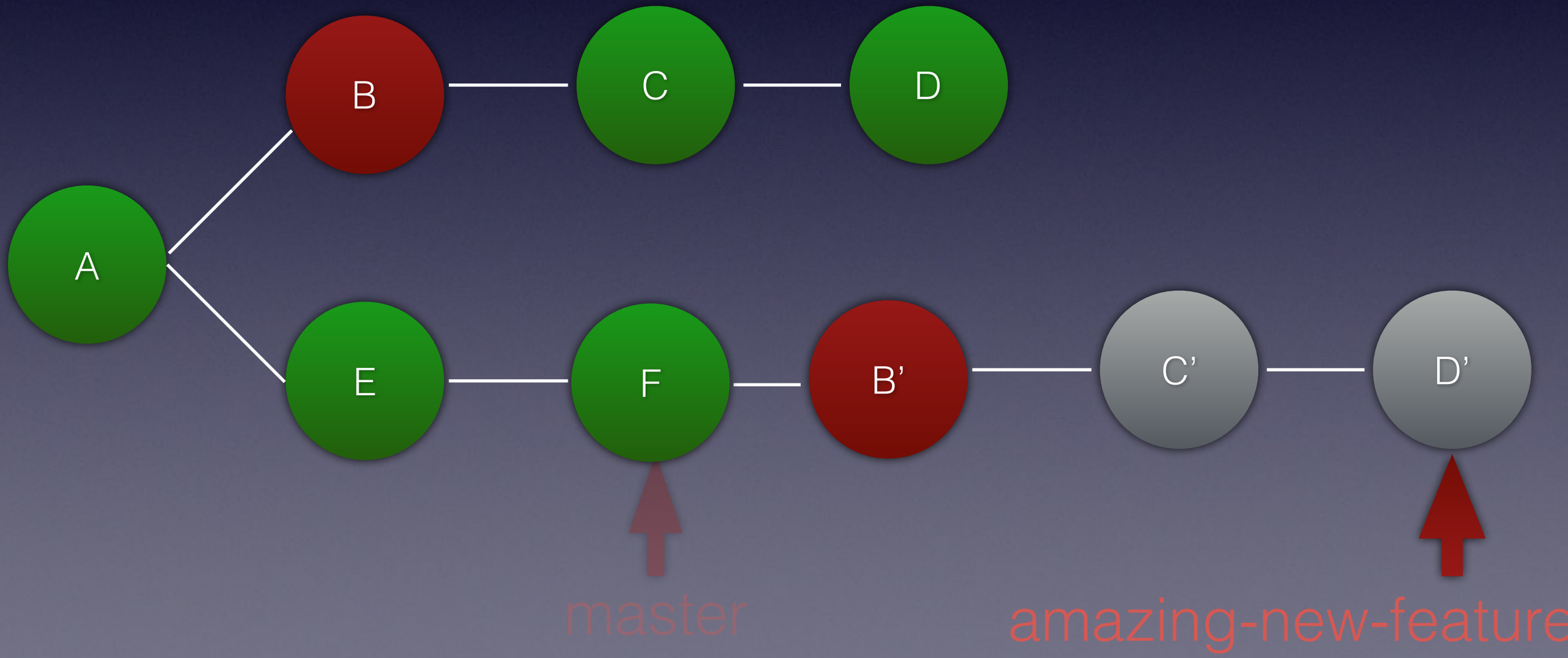
What is a commit?

1. Metadata
2. Patch
3. Parent commits(s)

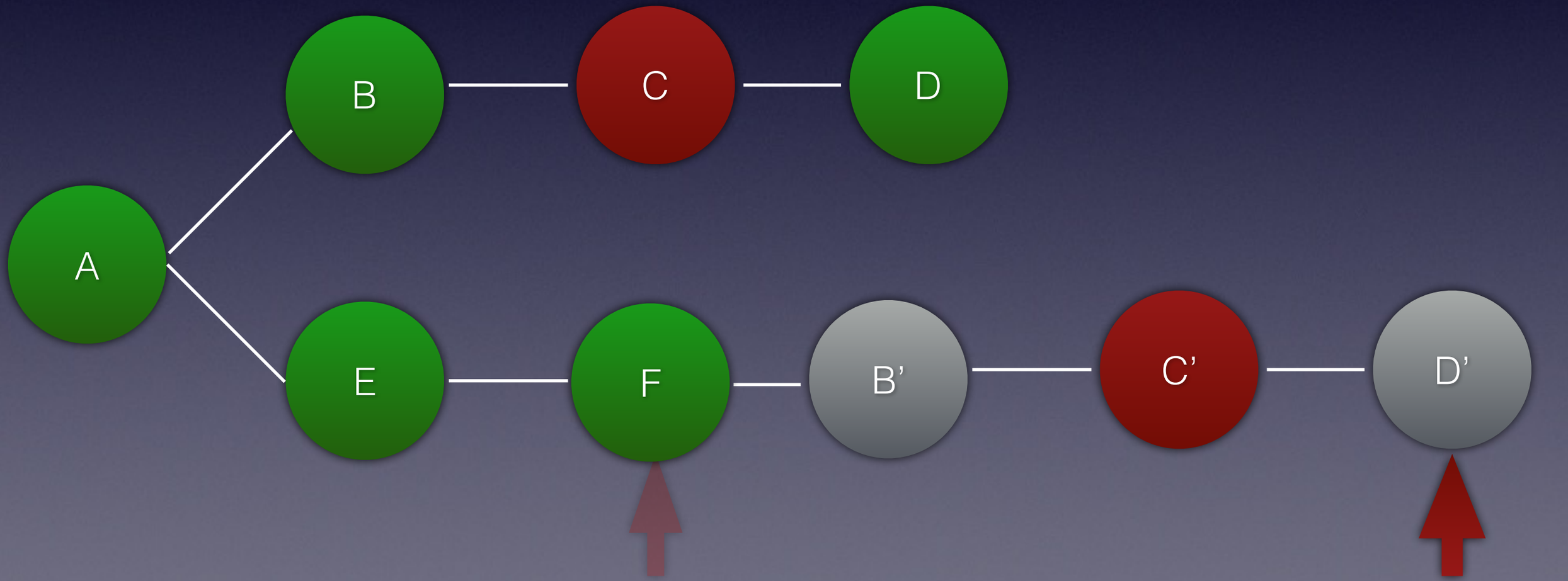
B parent is A
B’ parent is F

B’ has different hash to B

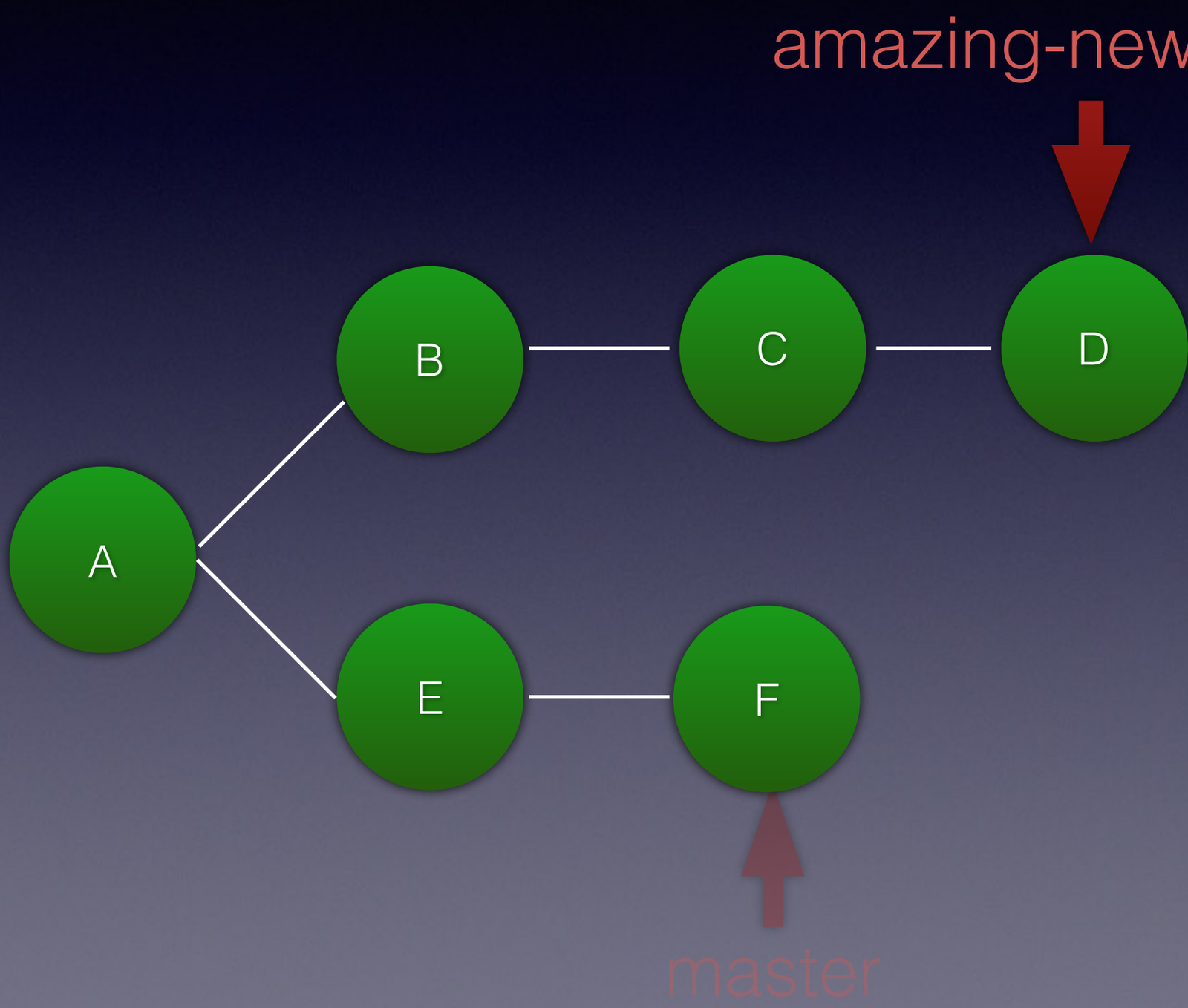
Rebase



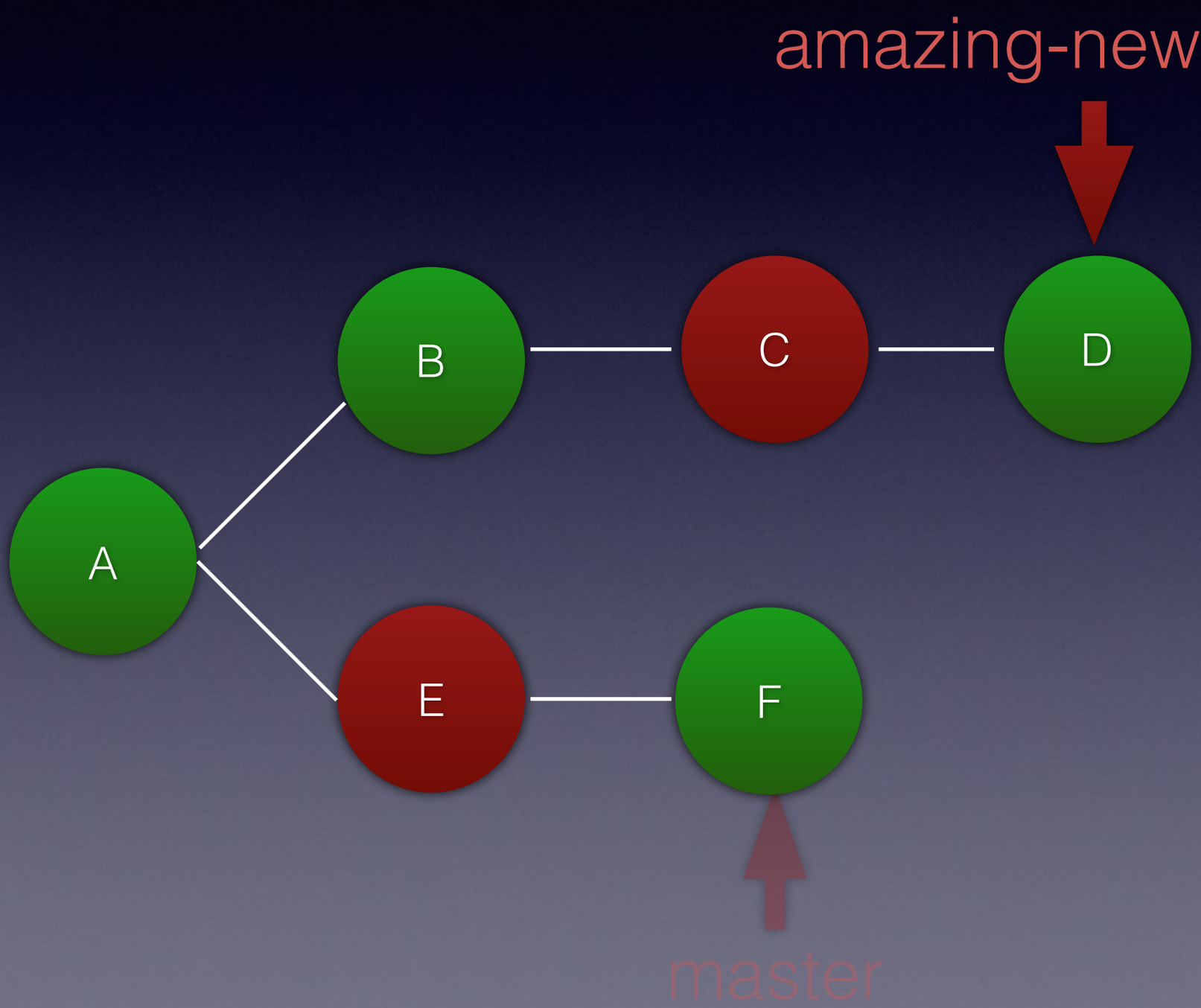
Rebase



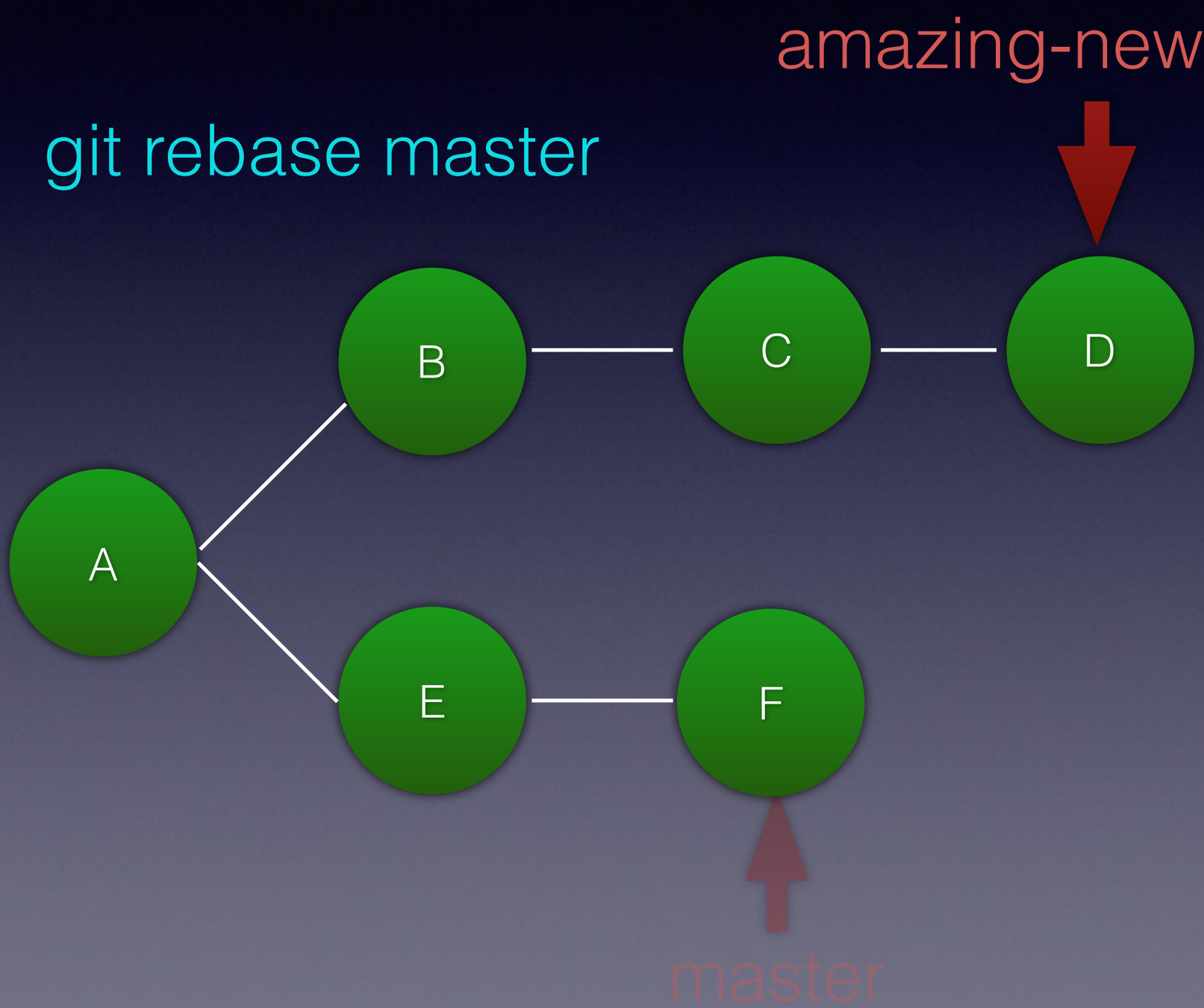
Conflicts with rebase



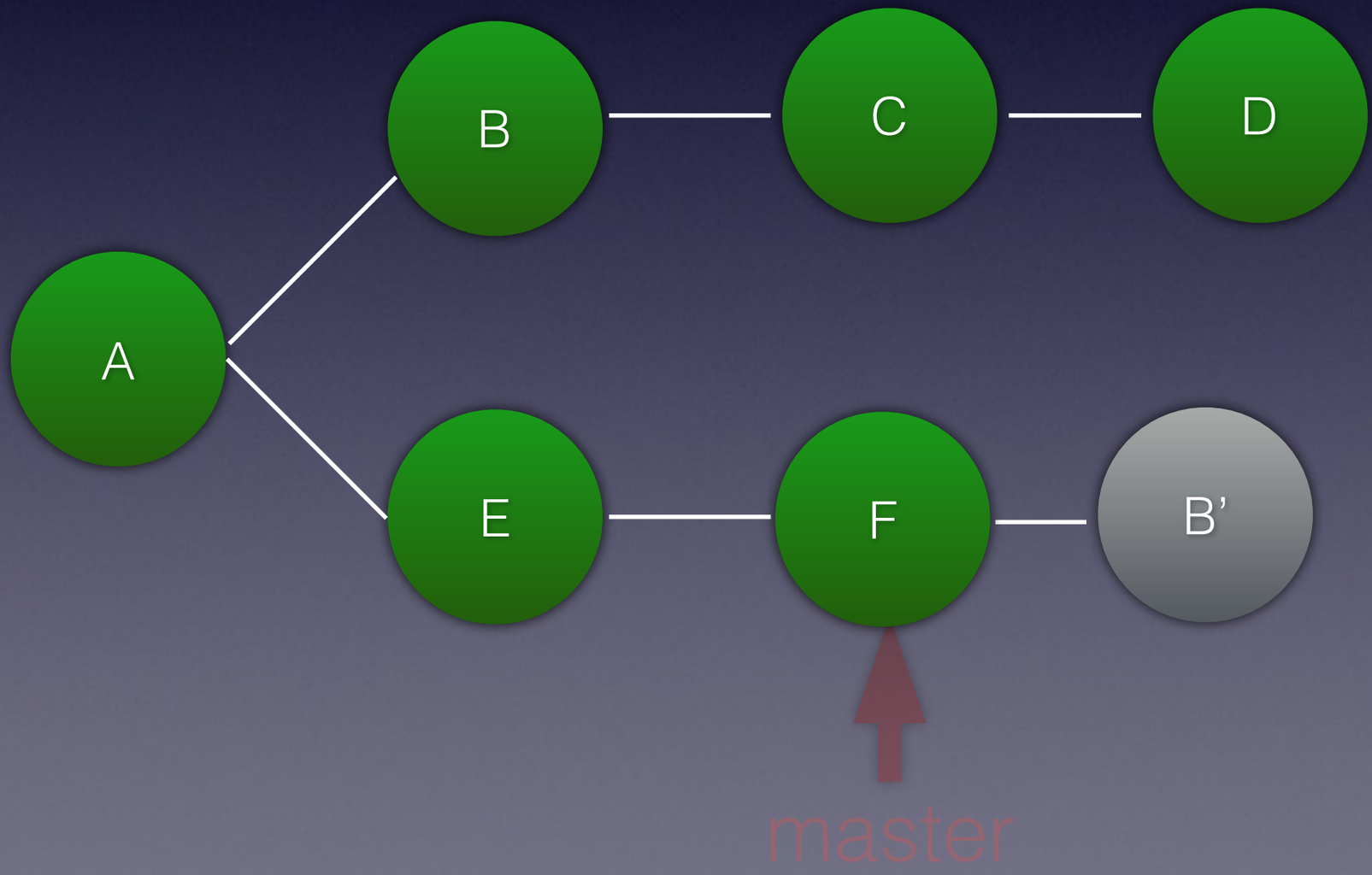
Conflicts with rebase



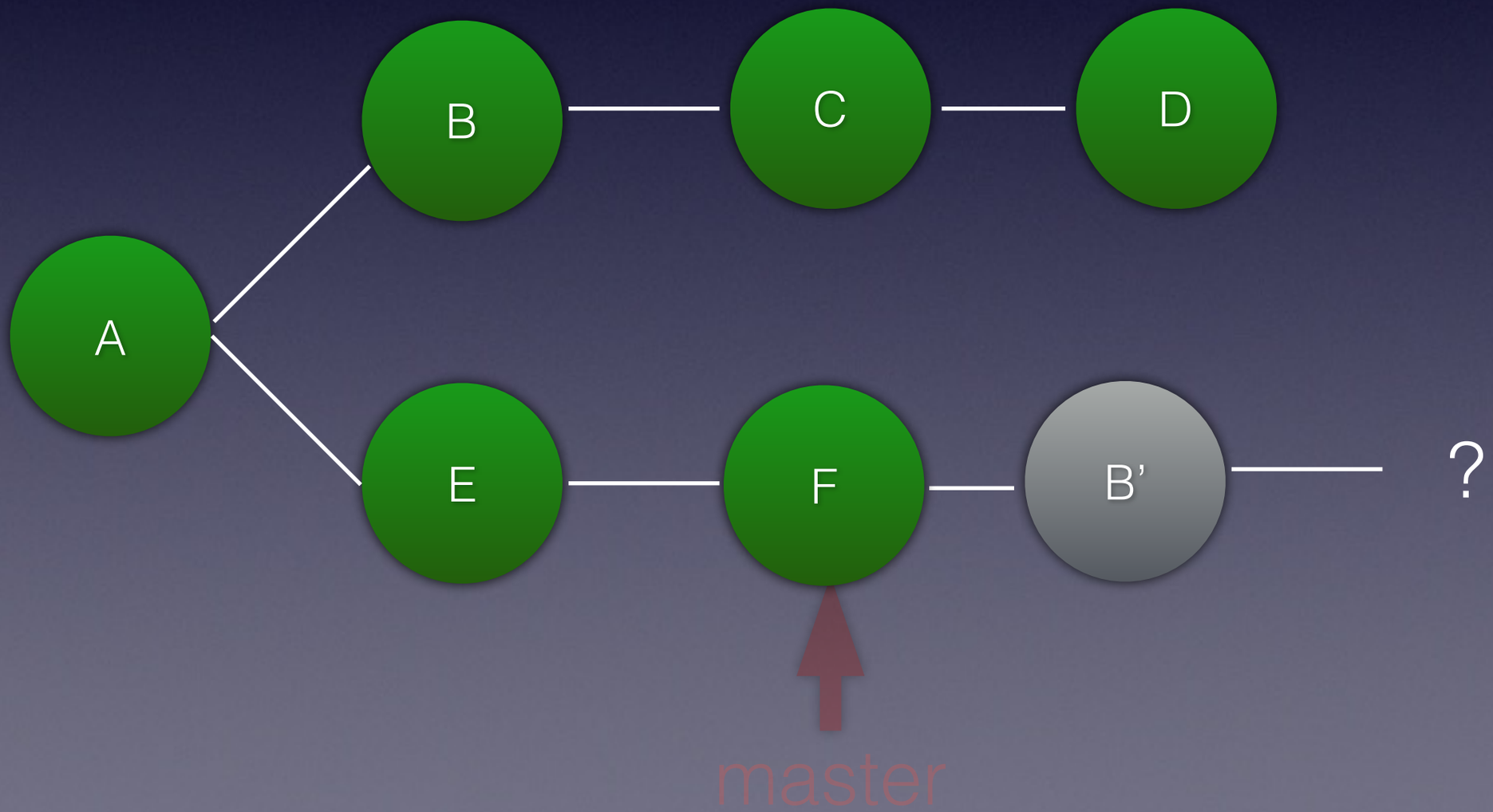
Conflicts with rebase



Conflicts with rebase



Conflicts with rebase



Conflicts with rebase

```
git rebase master

First, rewinding head to replay your work on top of it...
Applying: FIX 2
Using index info to reconstruct a base tree...
M days.txt
Falling back to patching base and 3-way merge...
Auto-merging days.txt
CONFLICT (content): Merge conflict in days.txt
error: Failed to merge in the changes.
Patch failed at 0001 FIX 2
The copy of the patch that failed is found in: .git/rebase-apply/patch

When you have resolved this problem, run "git rebase --continue".
If you prefer to skip this patch, run "git rebase --skip" instead.
To check out the original branch and stop rebasing, run "git rebase --abort".
```

Conflicts with rebase

```
git rebase master

First, rewinding head to replay your work on top of it...
Applying: FIX 2
Using index info to reconstruct a base tree...
M days.txt
Falling back to patching base and 3-way merge...
Auto-merging days.txt
CONFLICT (content): Merge conflict in days.txt
error: Failed to merge in the changes.
Patch failed at 0001 FIX 2
The copy of the patch that failed is found in: .git/rebase-apply/patch

When you have resolved this problem, run "git rebase --continue".
If you prefer to skip this patch, run "git rebase --skip" instead.
To check out the original branch and stop rebasing, run "git rebase --abort".
```

Conflicts with rebase

```
git rebase master

First, rewinding head to replay your work on top of it...
Applying: FIX 2
Using index info to reconstruct a base tree...
M days.txt
Falling back to patching base and 3-way merge...
Auto-merging days.txt
CONFLICT (content): Merge conflict in days.txt
error: Failed to merge in the changes.
Patch failed at 0001 FIX 2
The copy of the patch that failed is found in: .git/rebase-apply/patch

When you have resolved this problem, run "git rebase --continue".
If you prefer to skip this patch, run "git rebase --skip" instead.
To check out the original branch and stop rebasing, run "git rebase --abort".
```

Rebase conflicts

```
cat days.txt
```

```
Monday
<<<<<<< HEAD
Tuseday
=======
tuesday
>>>>>>> fix1
Wednesday
```

Rebase conflicts

```
cat days.txt
```

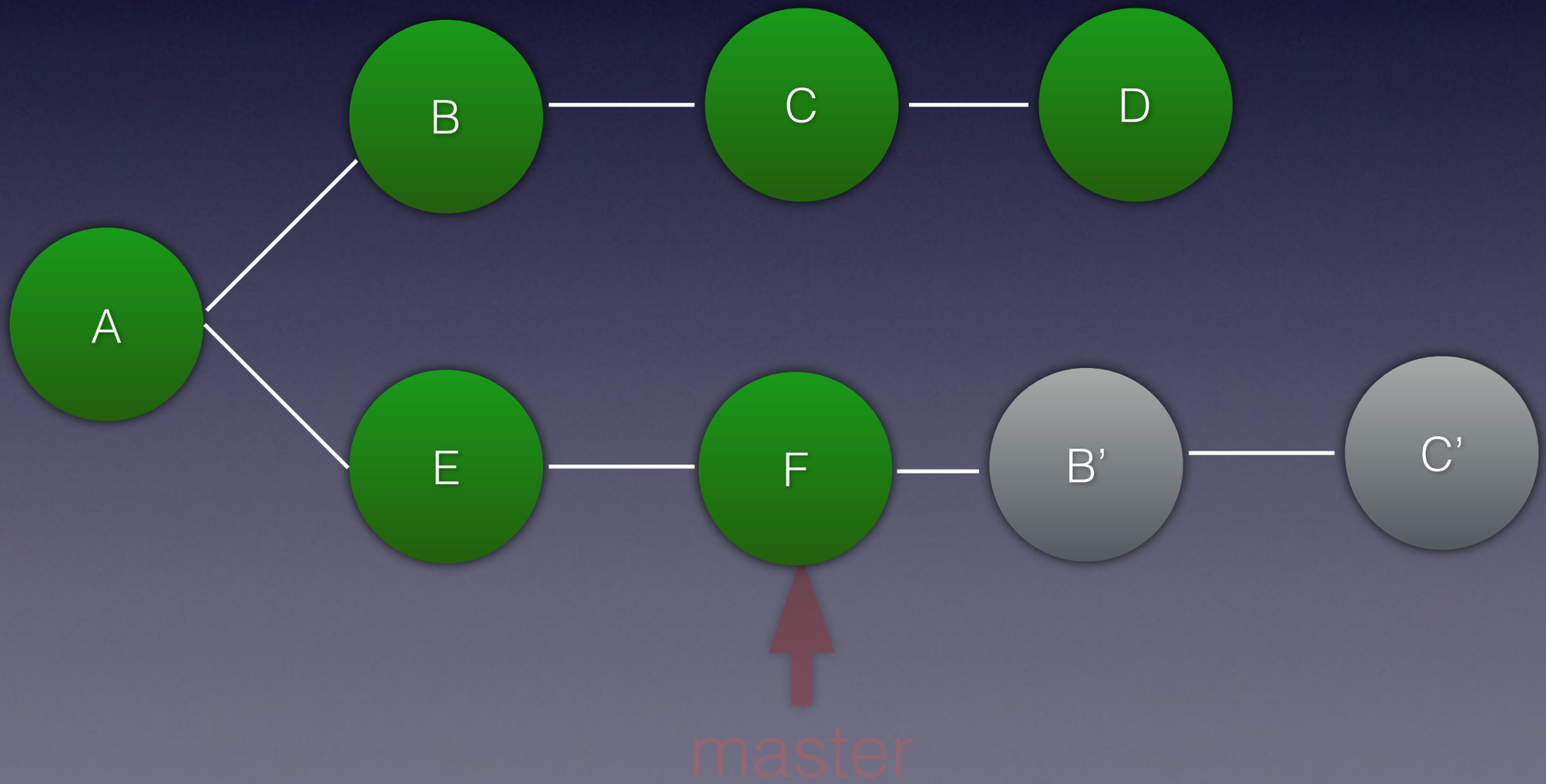
```
Monday
Tuesday
Wednesday
```

Rebase conflicts

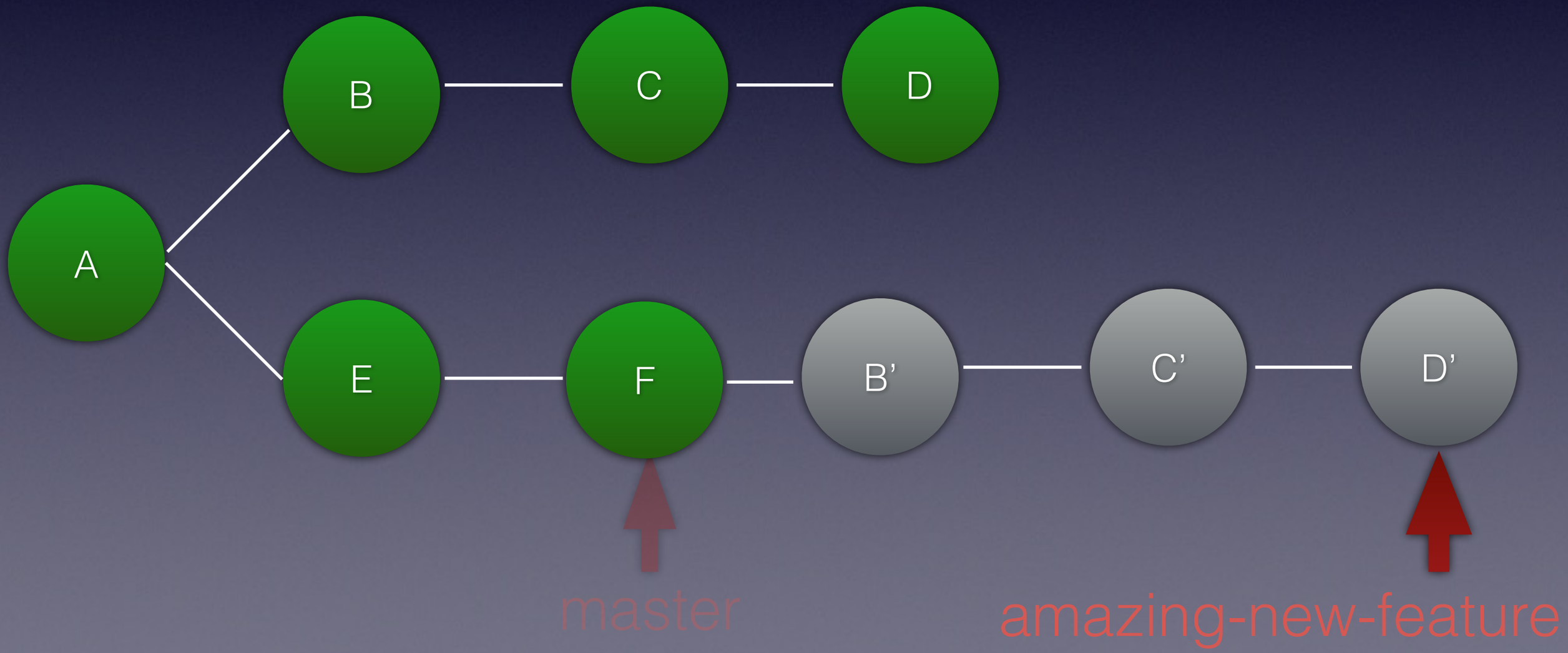
```
git add days.txt
```

```
git rebase --continue
```

Conflicts with rebase

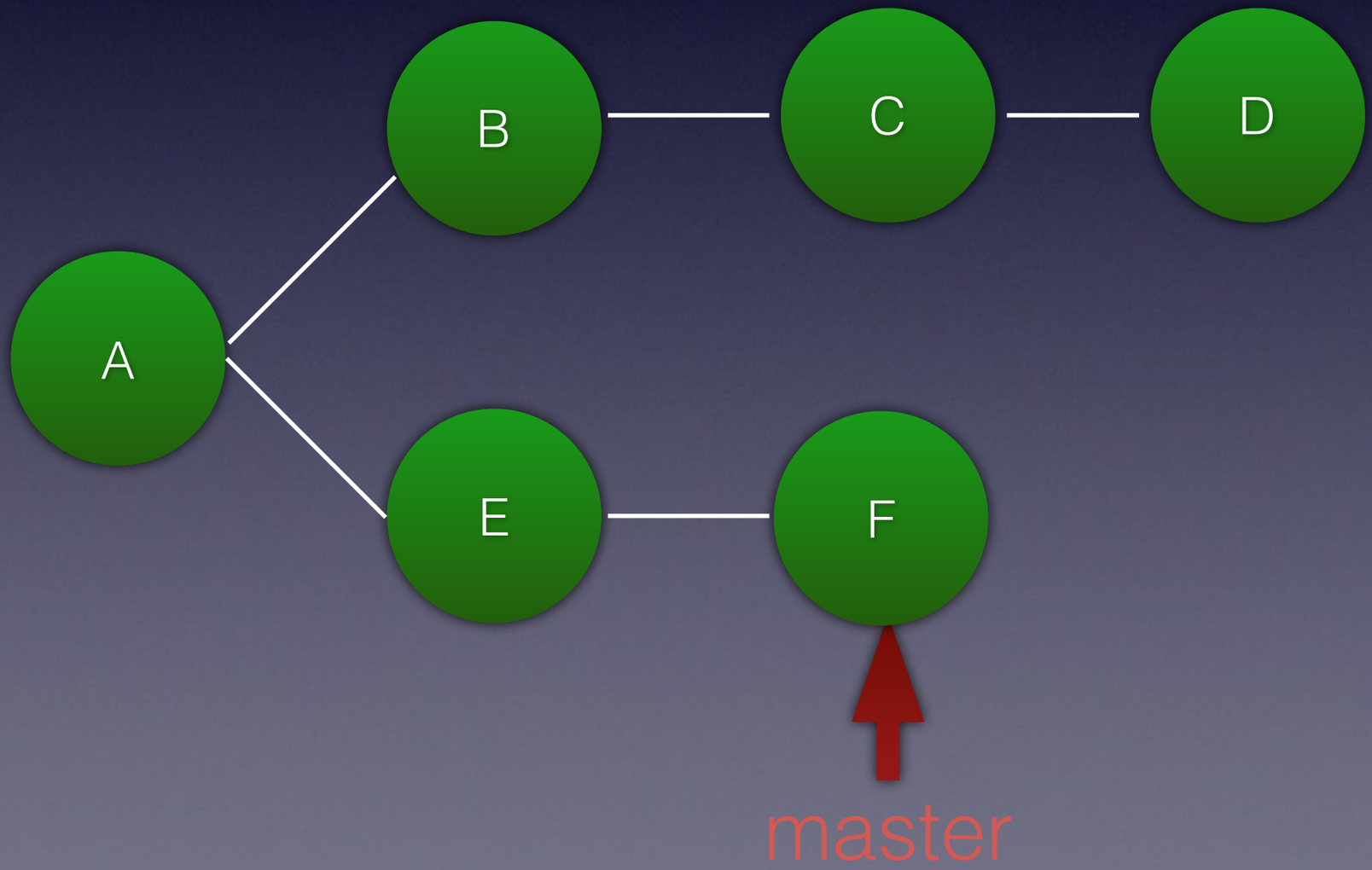


Conflicts with rebase

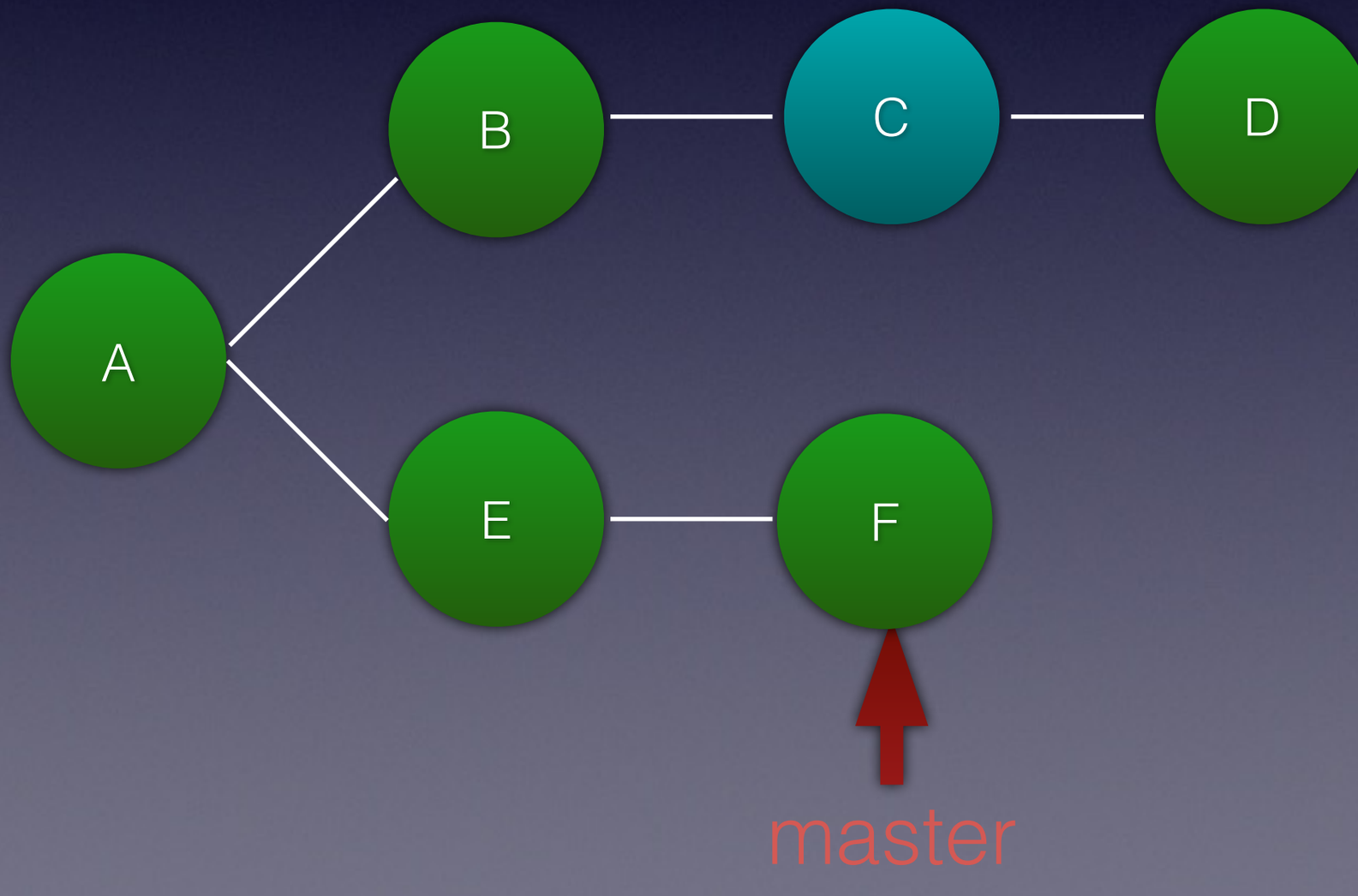


Conflicts

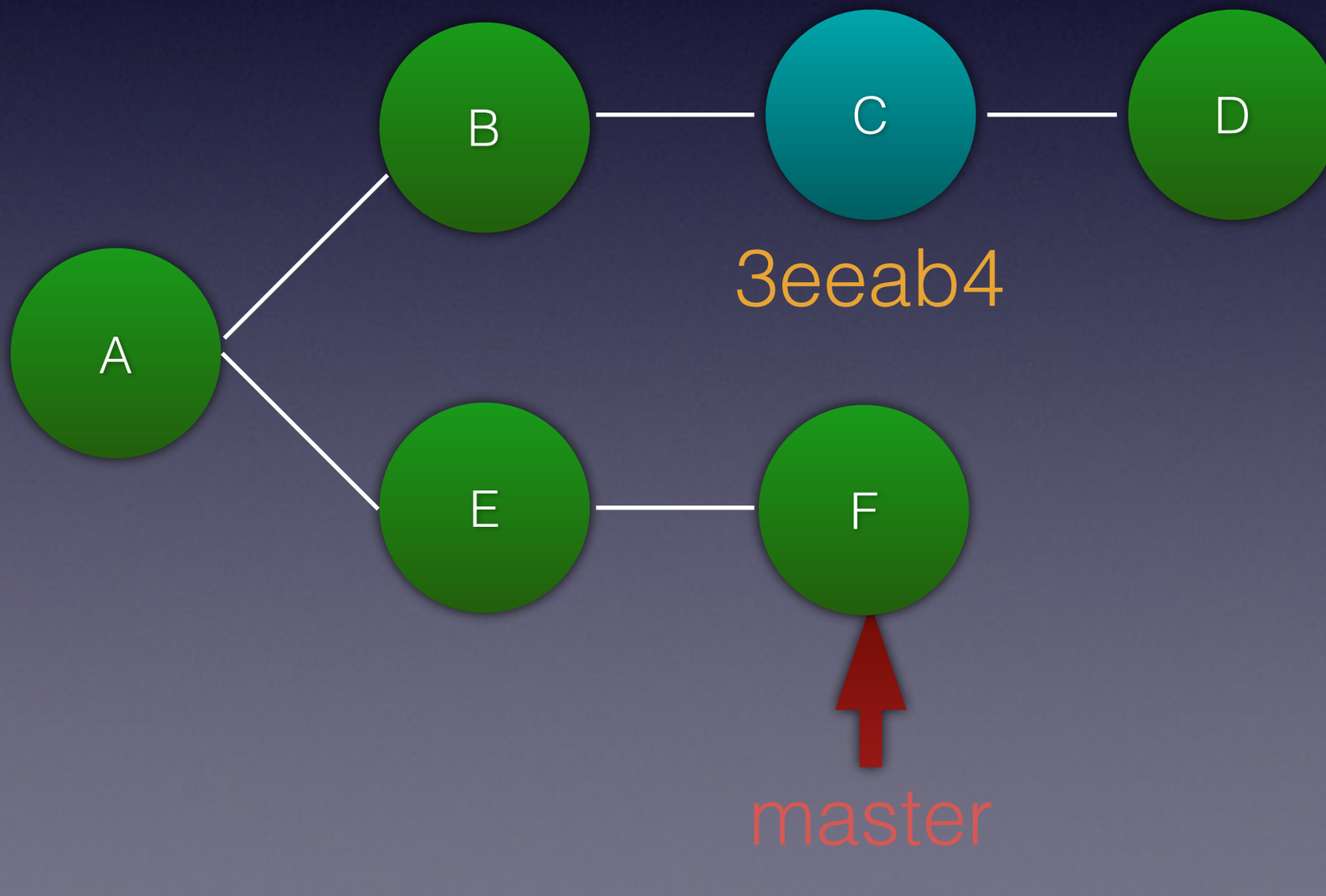
Cherry pick



Cherry pick

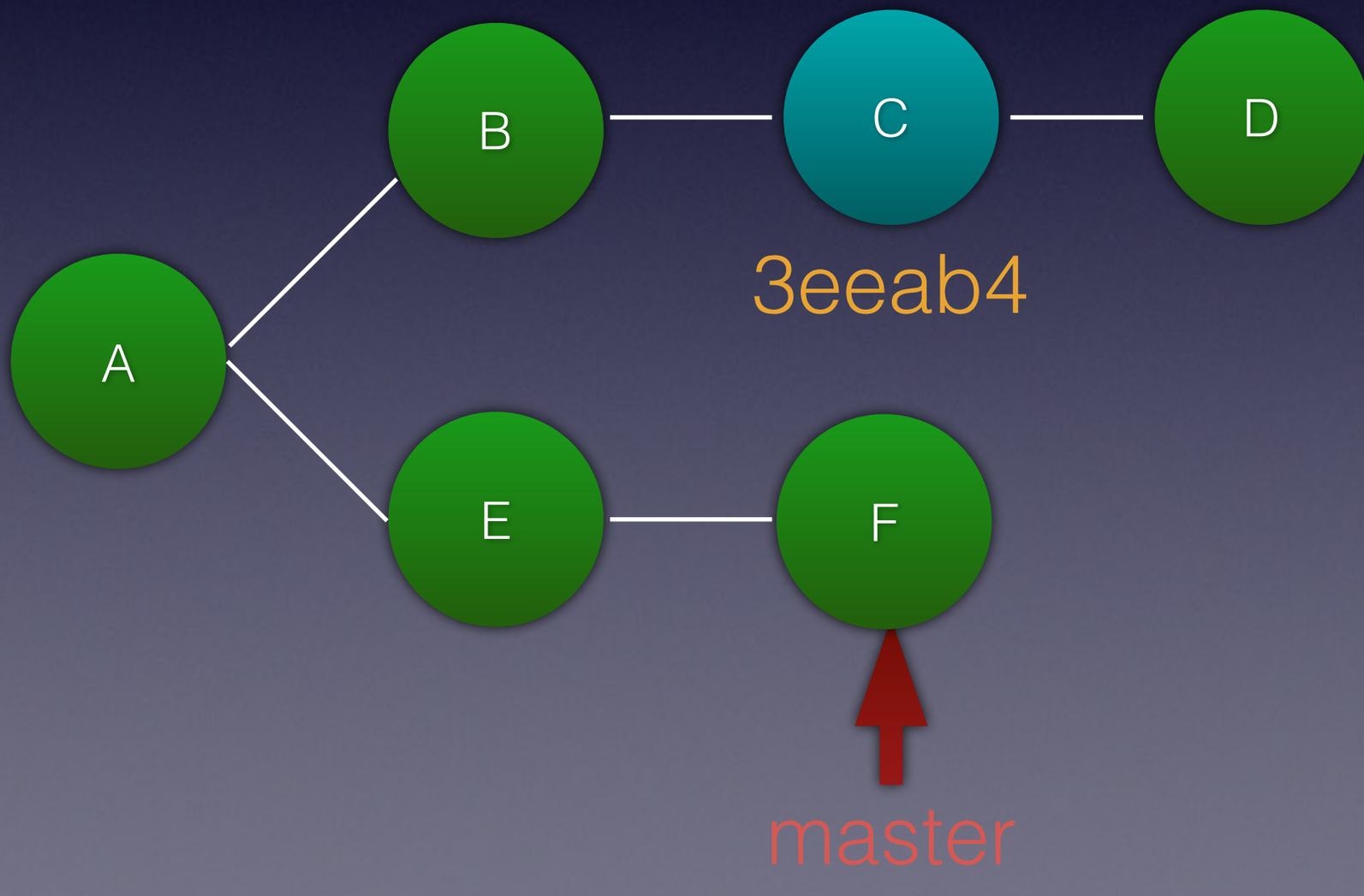


Cherry pick

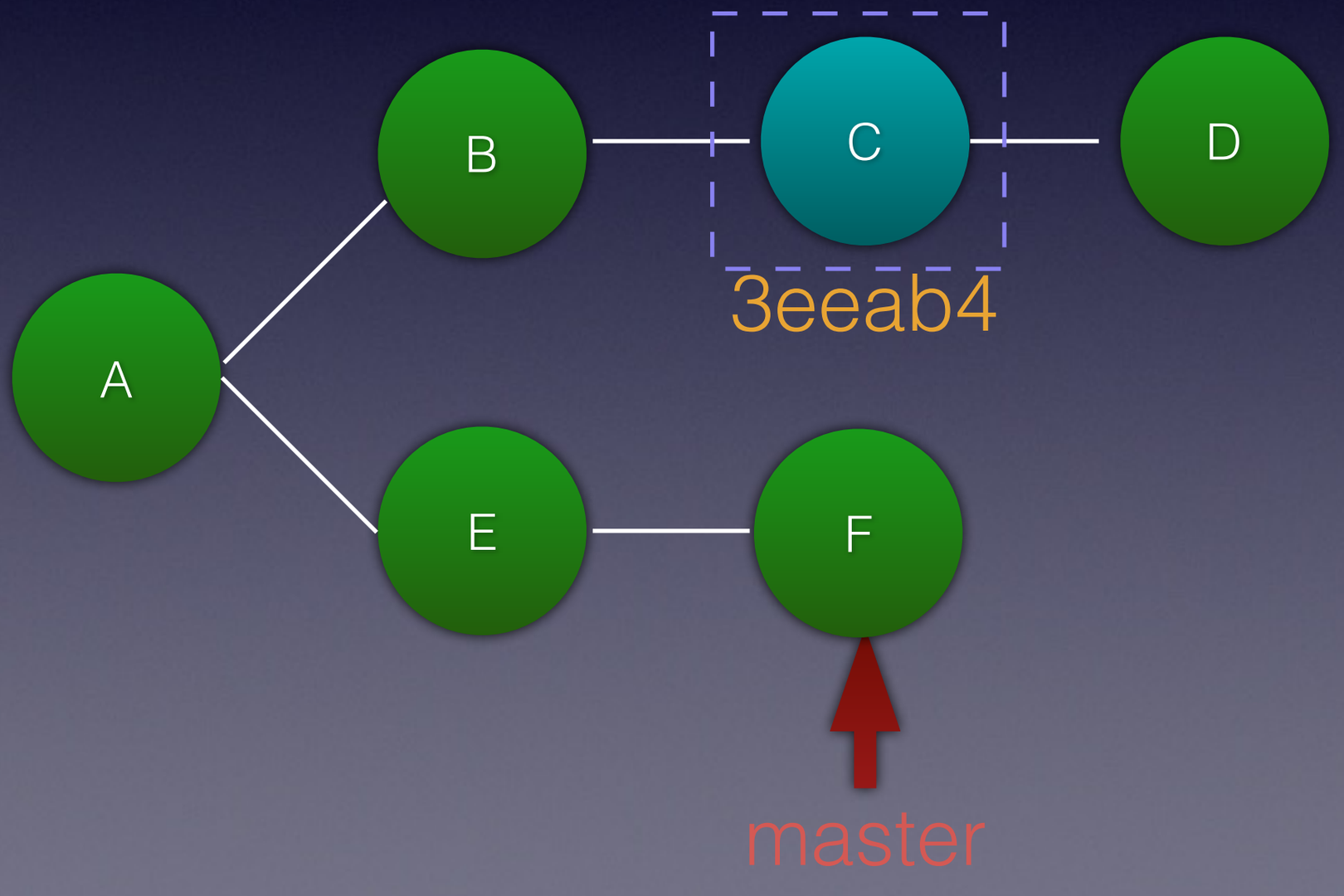


Cherry pick

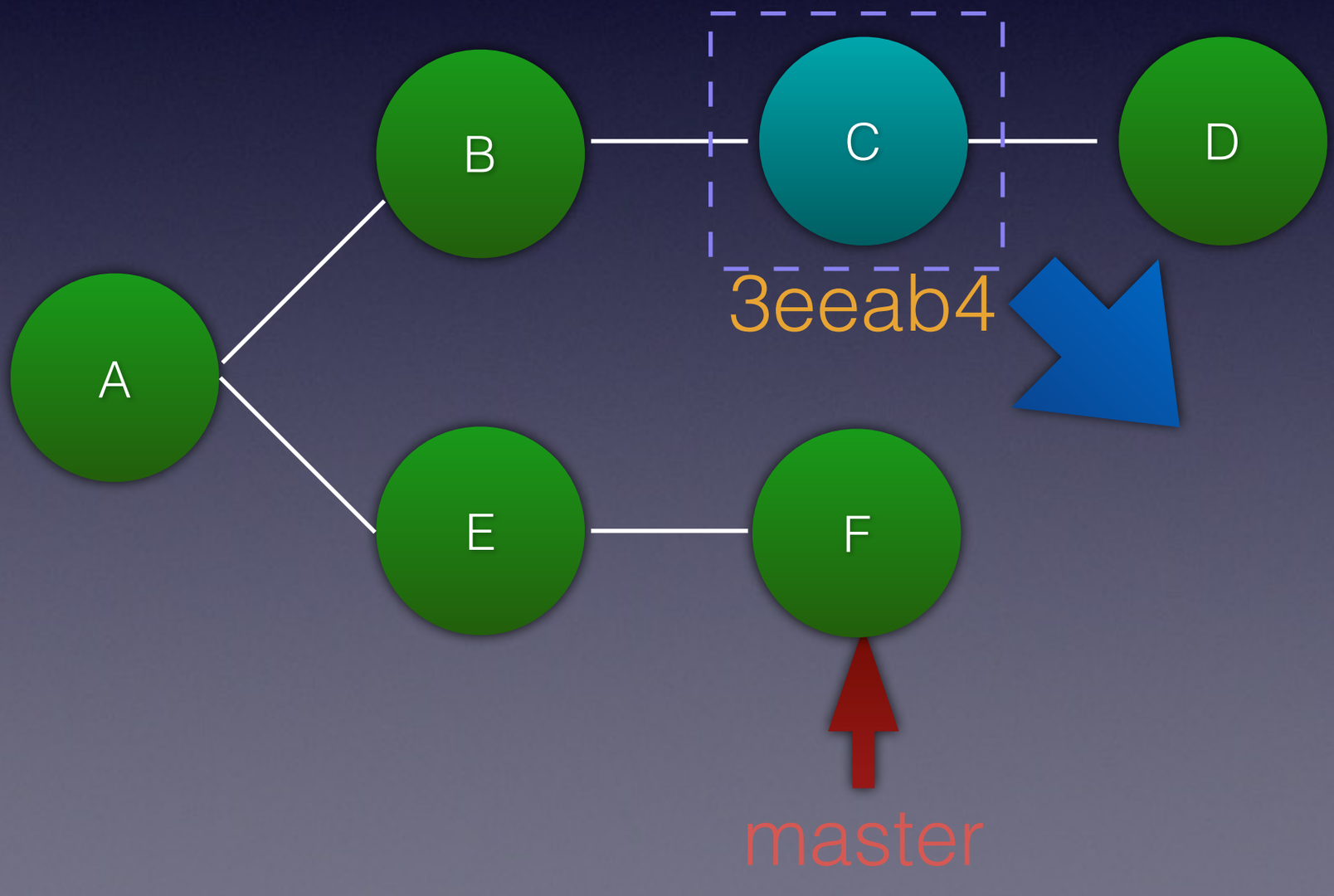
git cherry-pick 3eeab4



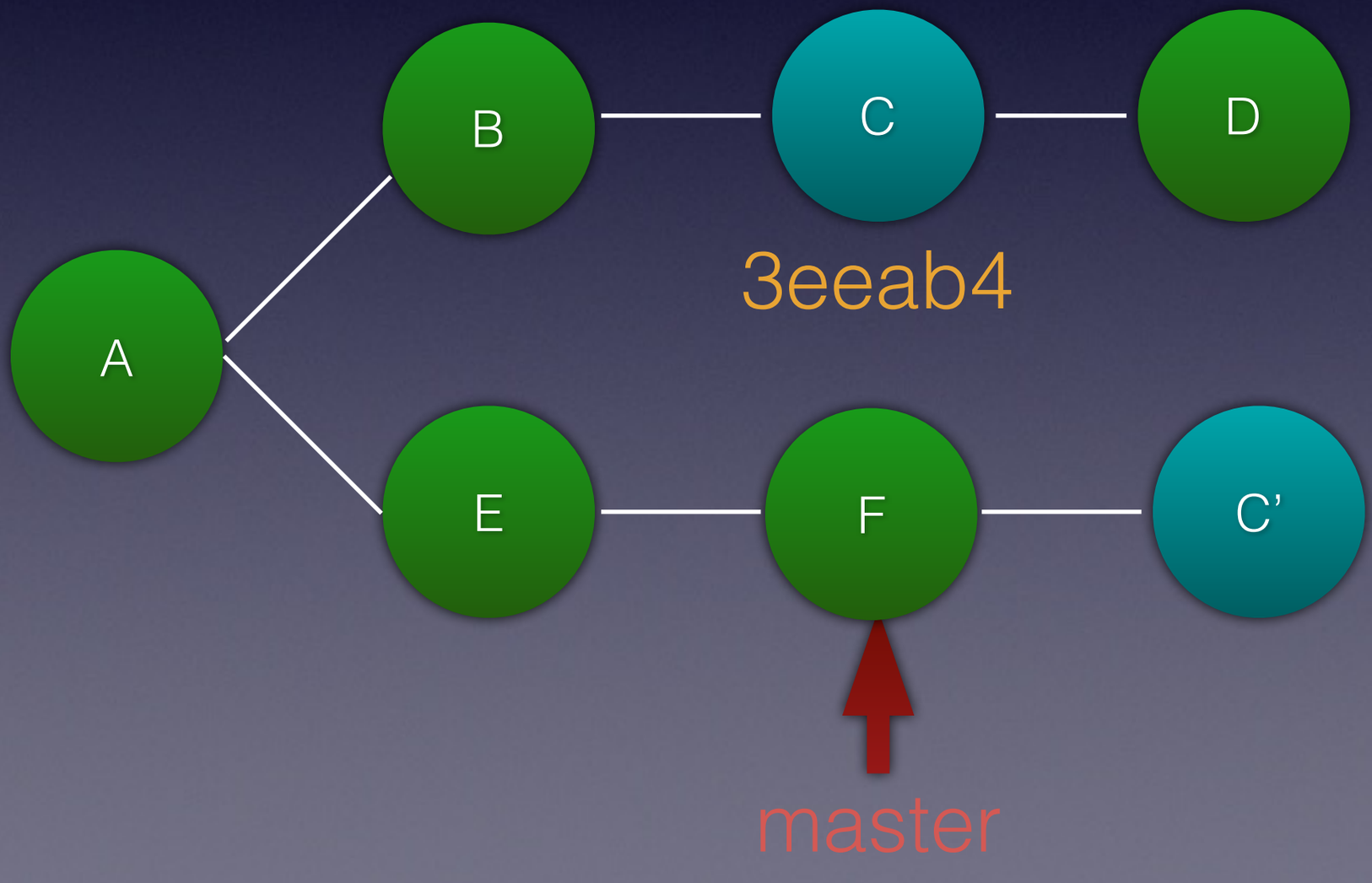
Cherry pick



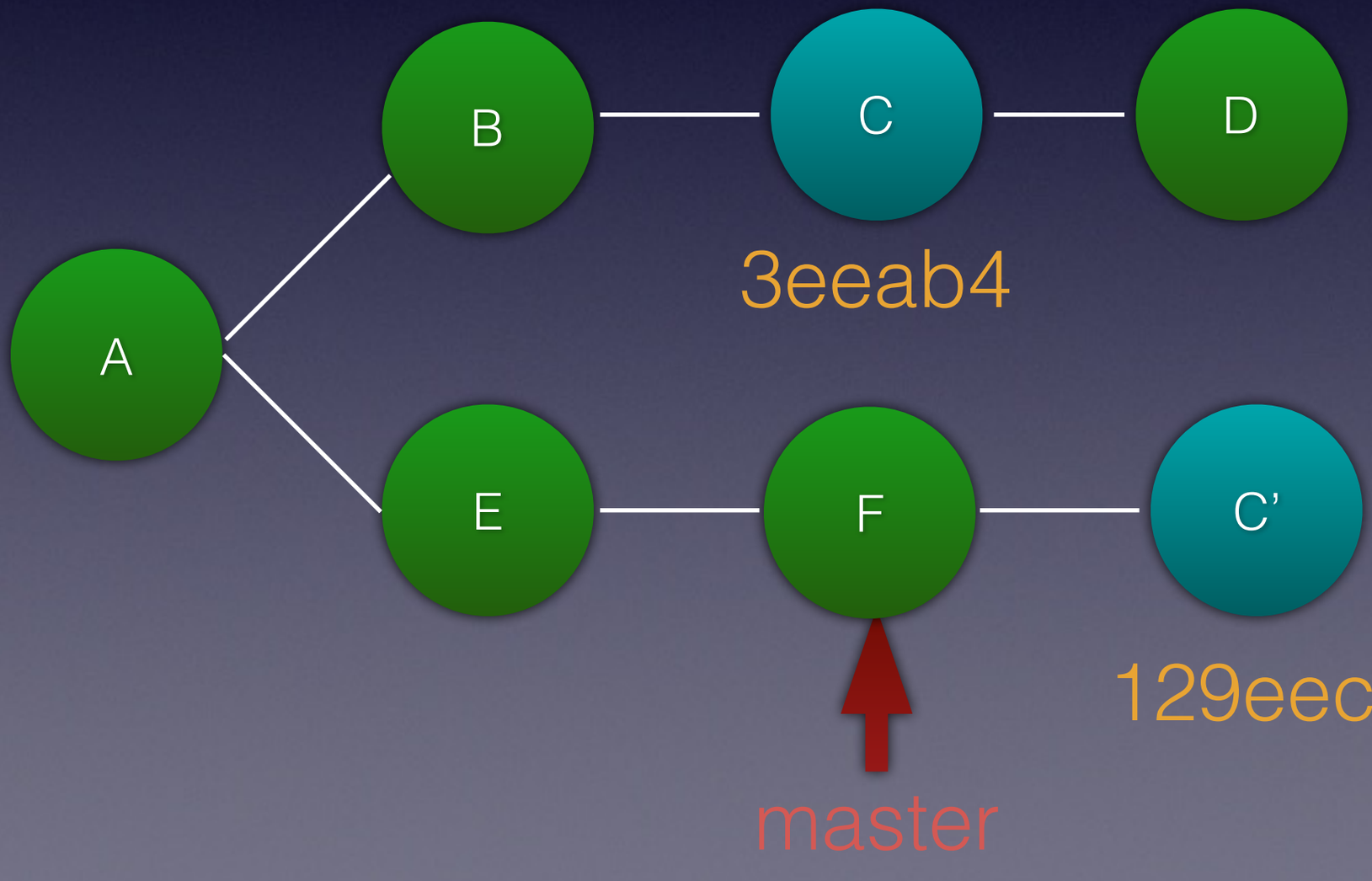
Cherry pick



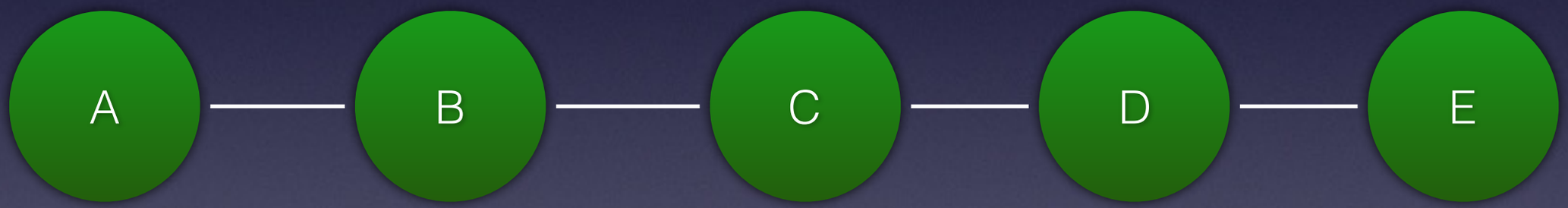
Cherry pick



Cherry pick

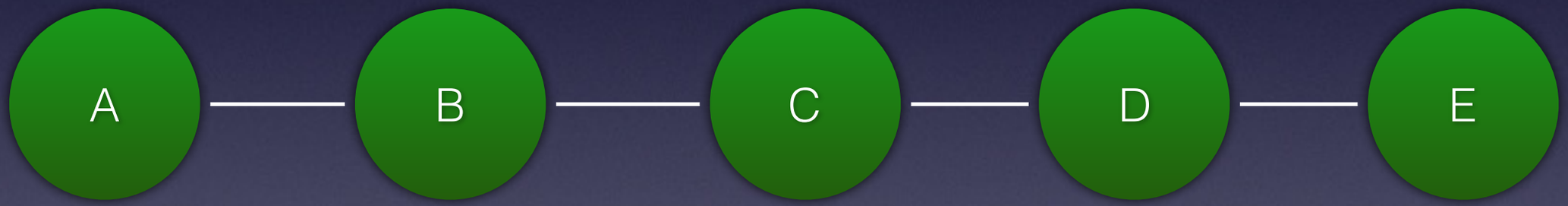


Rewriting history



Rewriting history

git rebase -i 7f66b2b



7f66b2b

Rewriting history

```
pick 6b9dde1 Message for commit B
pick f1c4ebc Message for commit C
pick f10a354 Message for commit D
pick 6f46cd9 Message for commit E

# Rebase 7f66b2b..6f46cd9 onto 7f66b2b (4 command(s))
#
# Commands:
# p, pick = use commit
# r, reword = use commit, but edit the commit message
# e, edit = use commit, but stop for amending
# s, squash = use commit, but meld into previous commit
# f, fixup = like "squash", but discard this commit's log message
# x, exec = run command (the rest of the line) using shell
# d, drop = remove commit
#
# These lines can be re-ordered; they are executed from top to bottom.
```

Rewriting history

```
pick 6b9dde1 Message for commit B
pick f1c4ebc Message for commit C
pick f10a354 Message for commit D
pick 6f46cd9 Message for commit E

# Rebase 7f66b2b..6f46cd9 onto 7f66b2b (4 command(s))
#
# Commands:
# p, pick = use commit
# r, reword = use commit, but edit the commit message
# e, edit = use commit, but stop for amending
# s, squash = use commit, but meld into previous commit
# f, fixup = like "squash", but discard this commit's log message
# x, exec = run command (the rest of the line) using shell
# d, drop = remove commit
#
# These lines can be re-ordered; they are executed from top to bottom.
```

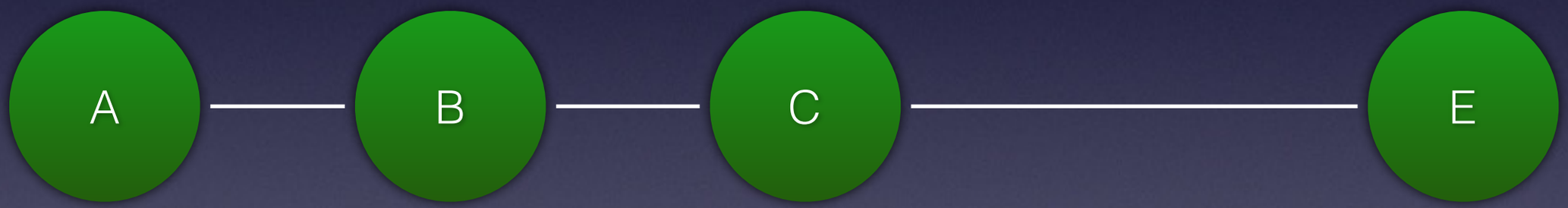
Rewriting history

```
pick 6b9dde1 Message for commit B
pick f1c4ebc Message for commit C
pick f10a354 Message for commit D
pick 6f46cd9 Message for commit E

# Rebase 7f66b2b..6f46cd9 onto 7f66b2b (4 command(s))
#
# Commands:
# p, pick = use commit
# r, reword = use commit, but edit the commit message
# e, edit = use commit, but stop for amending
# s, squash = use commit, but meld into previous commit
# f, fixup = like "squash", but discard this commit's log message
# x, exec = run command (the rest of the line) using shell
# d, drop = remove commit
#
# These lines can be re-ordered; they are executed from top to bottom.
```

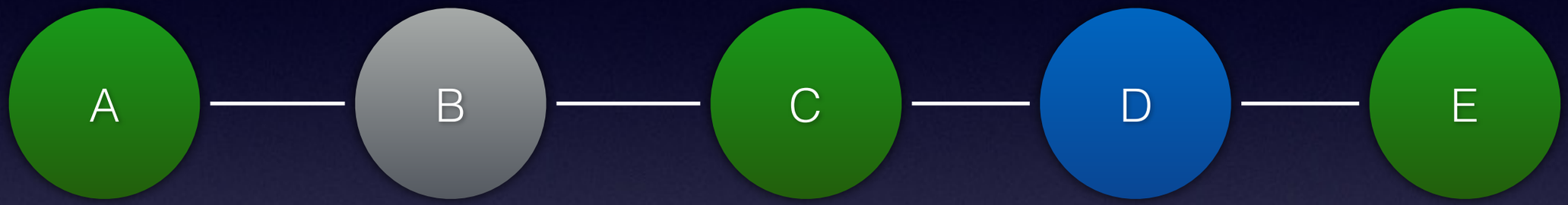
Rewriting history

```
pick 6b9dde1 Message for commit B
pick f1c4ebc Message for commit C
drop f10a354 Message for commit D
pick 6f46cd9 Message for commit E
```

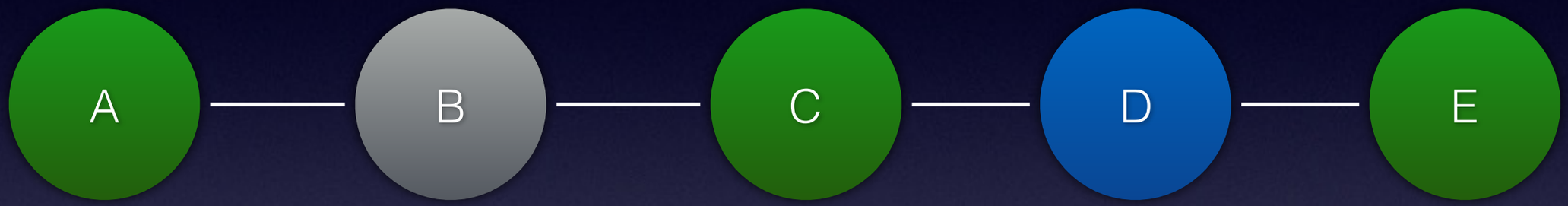
Rewriting history

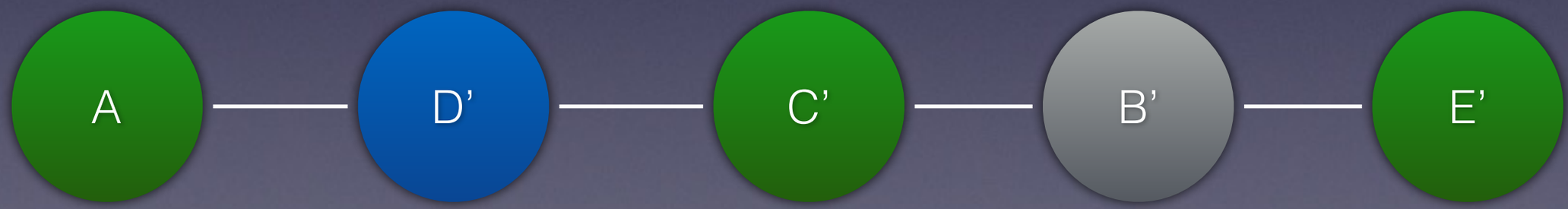


Rewriting history



Rewriting history





Rewriting history

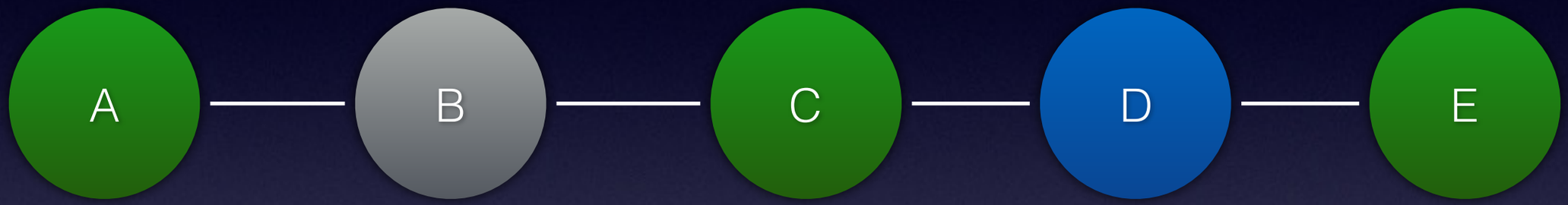
```
pick 6b9dde1 Message for commit B
pick f1c4ebc Message for commit C
pick f10a354 Message for commit D
pick 6f46cd9 Message for commit E
```

Rewriting history

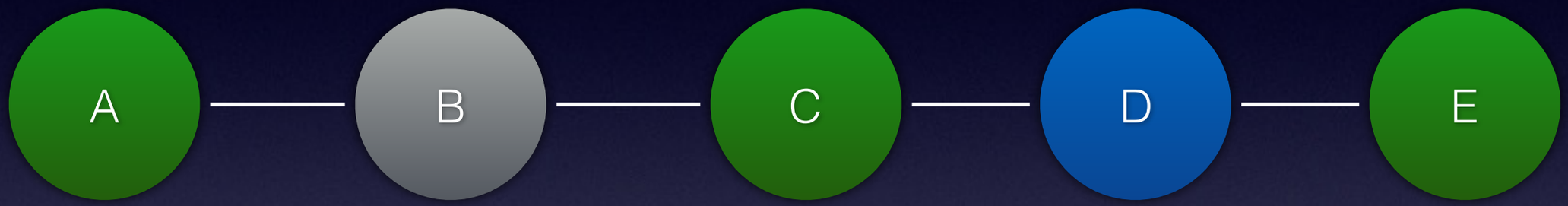
```
pick 6b9dde1 Message for commit B
pick f1c4ebc Message for commit C
pick f10a354 Message for commit D
pick 6f46cd9 Message for commit E
```

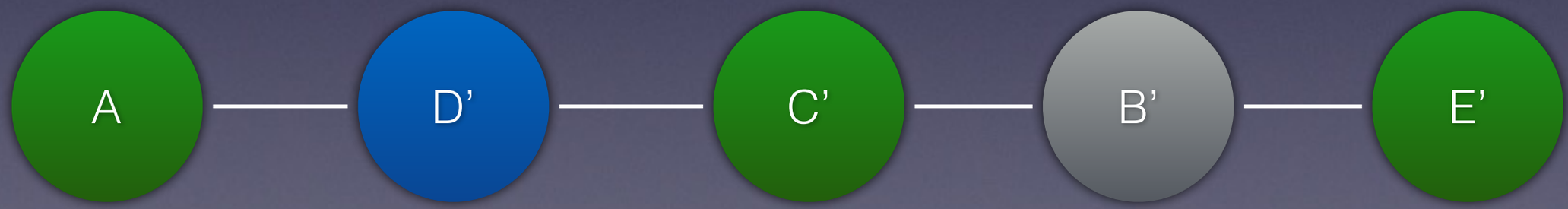
```
pick f10a354 Message for commit D
pick f1c4ebc Message for commit C
pick 6b9dde1 Message for commit B
pick 6f46cd9 Message for commit E
```

Rewriting history

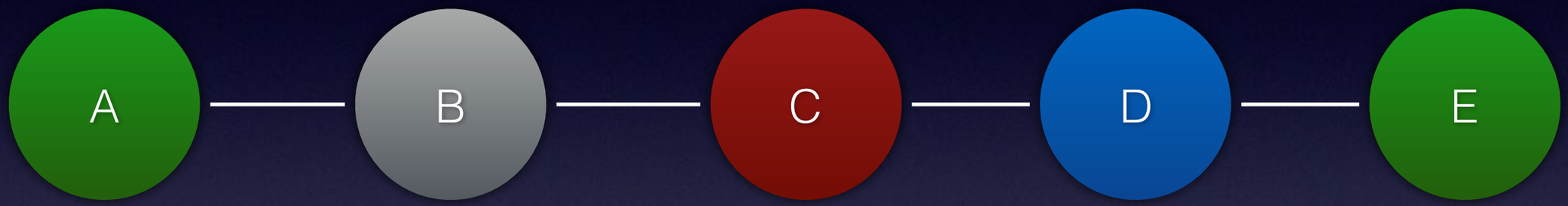


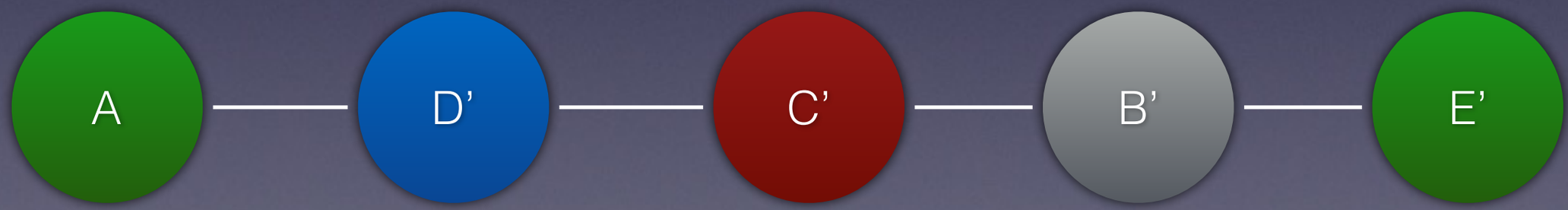
Rewriting history



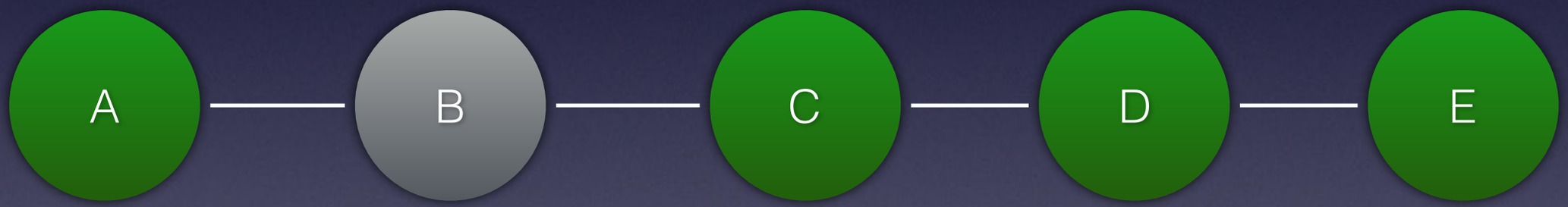


Rewriting history





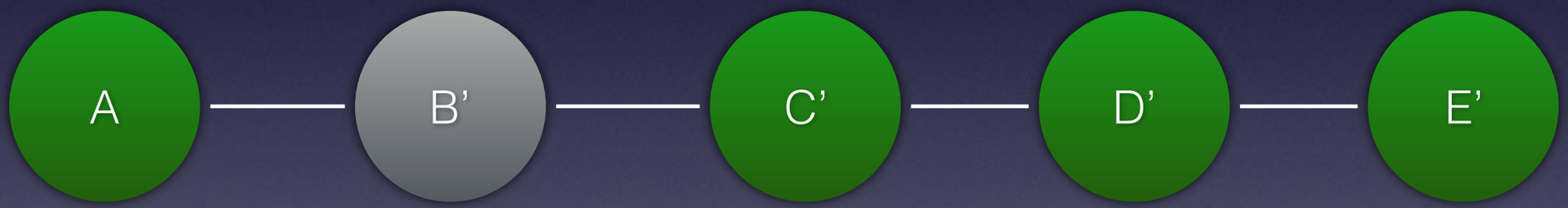
Rewriting history



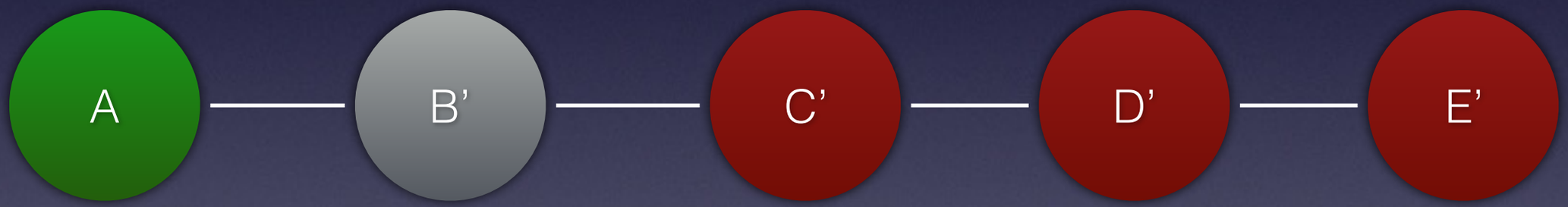
Rewriting history

```
edit 6b9dde1 Message for commit B
pick f1c4ebc Message for commit C
pick f10a354 Message for commit D
pick 6f46cd9 Message for commit E
```

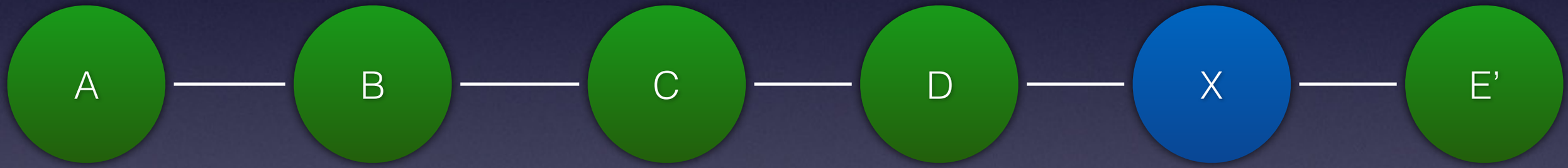
Rewriting history



Rewriting history



Rewriting history



Rewriting history

This is all very well,

but…

Why this?

Commit

Staged for commit

Working tree

Untracked

Why rewrite history?

Make small commits that are easier to review.
Commit history can tell a story.

Single commit

FEATURE: Implement new pricing code and refactor old code to use this.

Many commits

COMPOSER: Add joe-blogs/pricing package
ADD: New domain model for price
ADD: Business logic for simple pricing
ADD: Ability to use a gift voucher
DEPRECATE: Legacy price calculation ad
REFACTOR: Pricing endpoint to use new pricing logic
REFACTOR: Batch job to use new pricing logic
REMOVE: Deprecated classes

Commits should be small

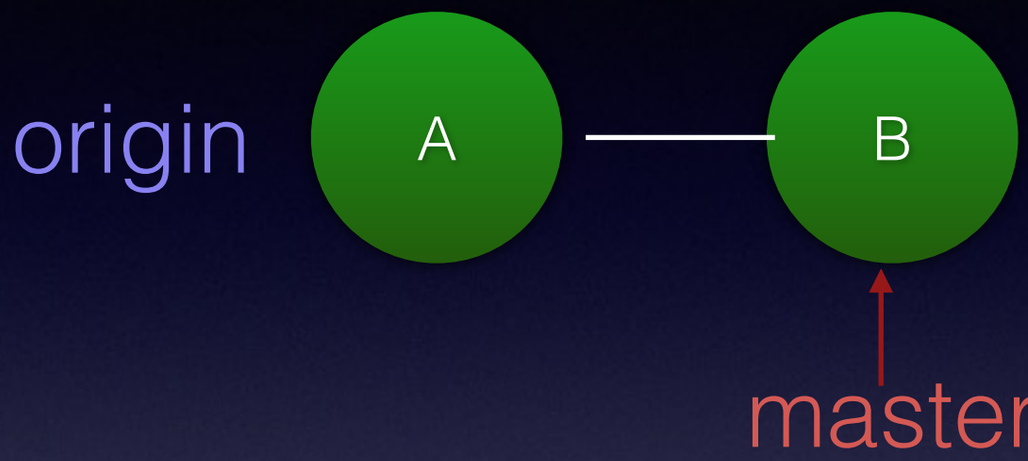
10 lines of code = 10 issues

500 lines of code = “all fine”

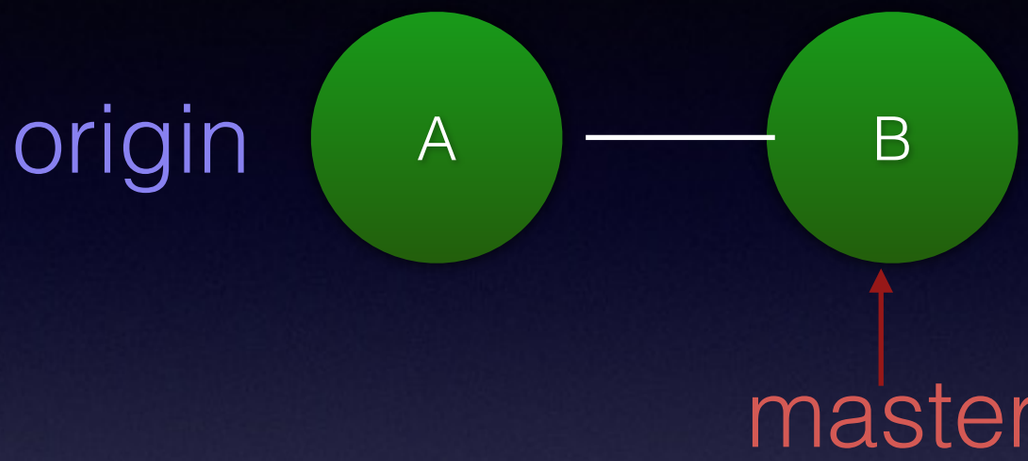
Be careful rewriting history

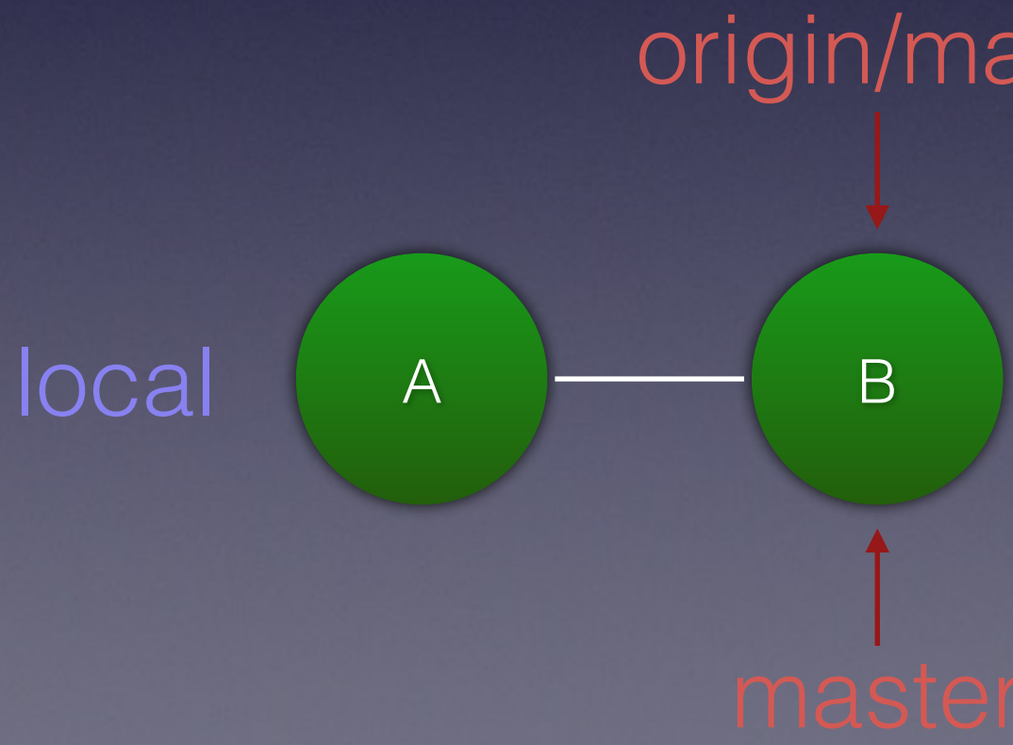
Remote repositories

Remote repositories



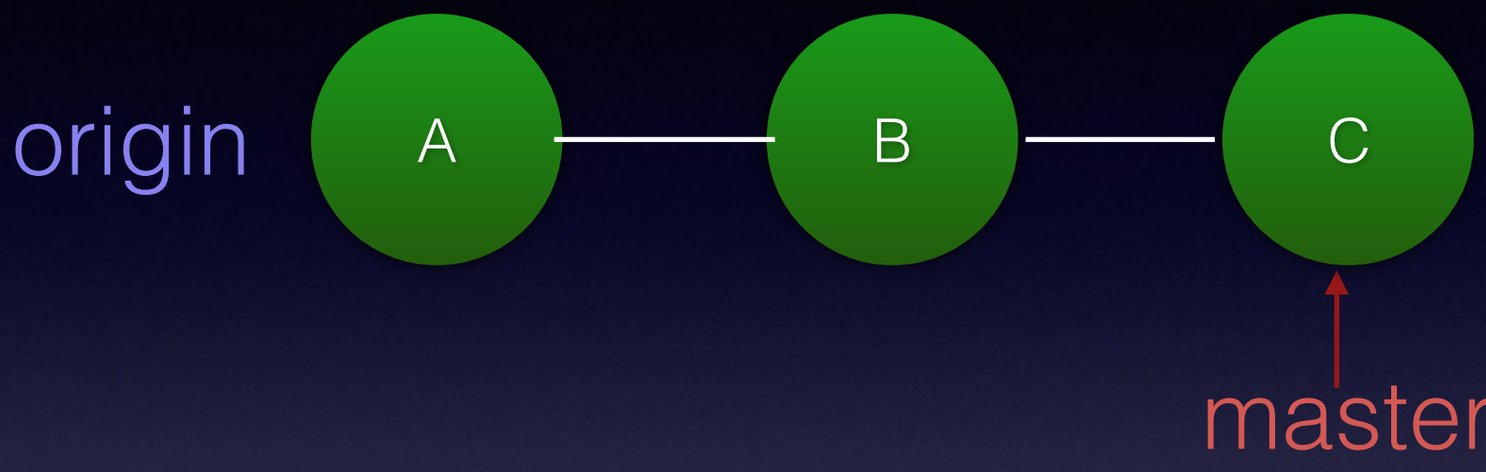
Remote repositories

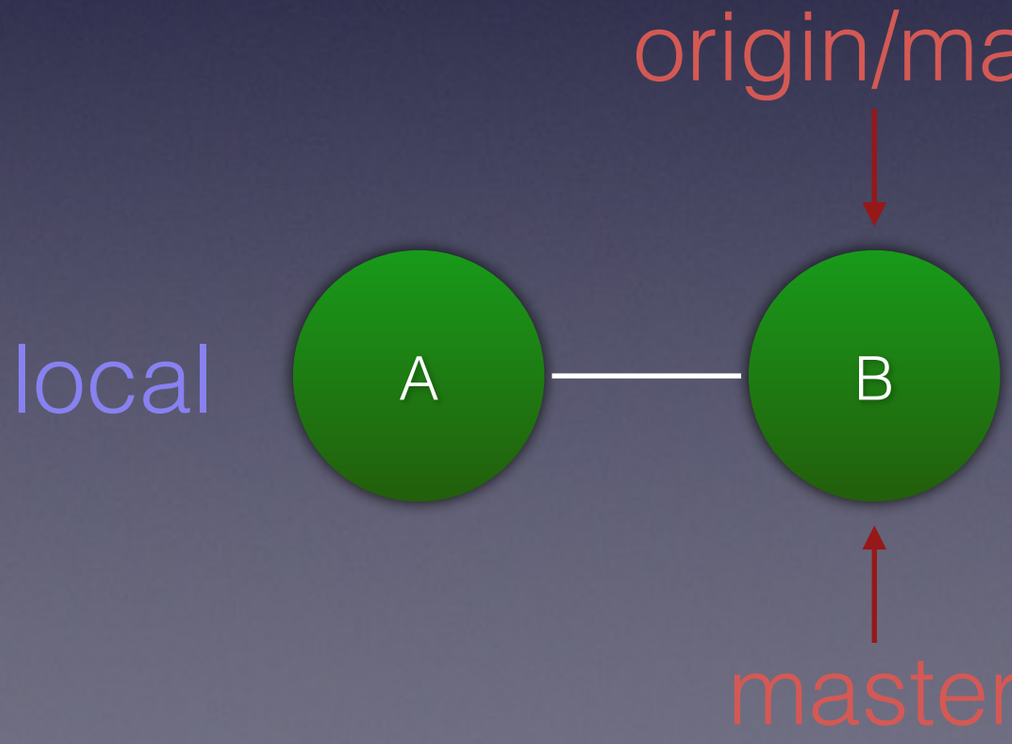




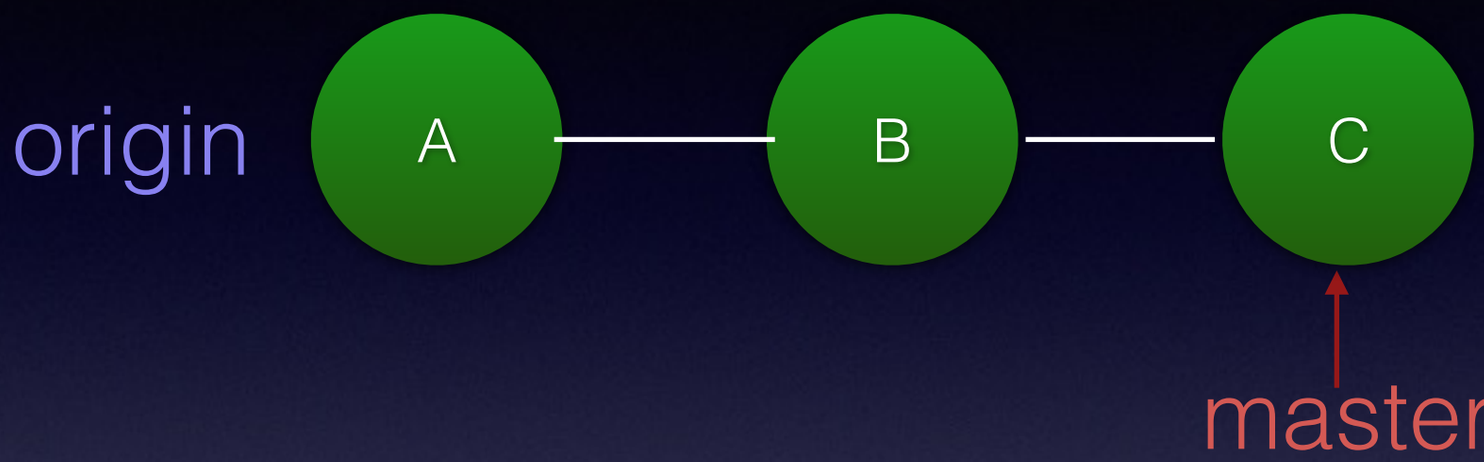
Updating local repository

Remote repositories

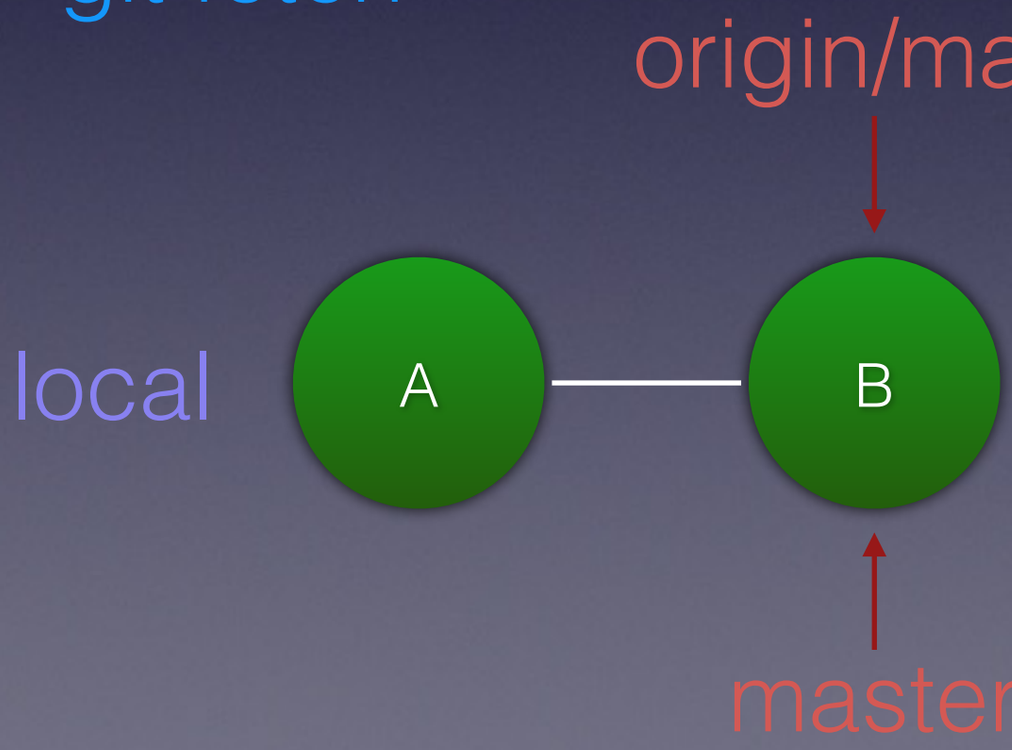




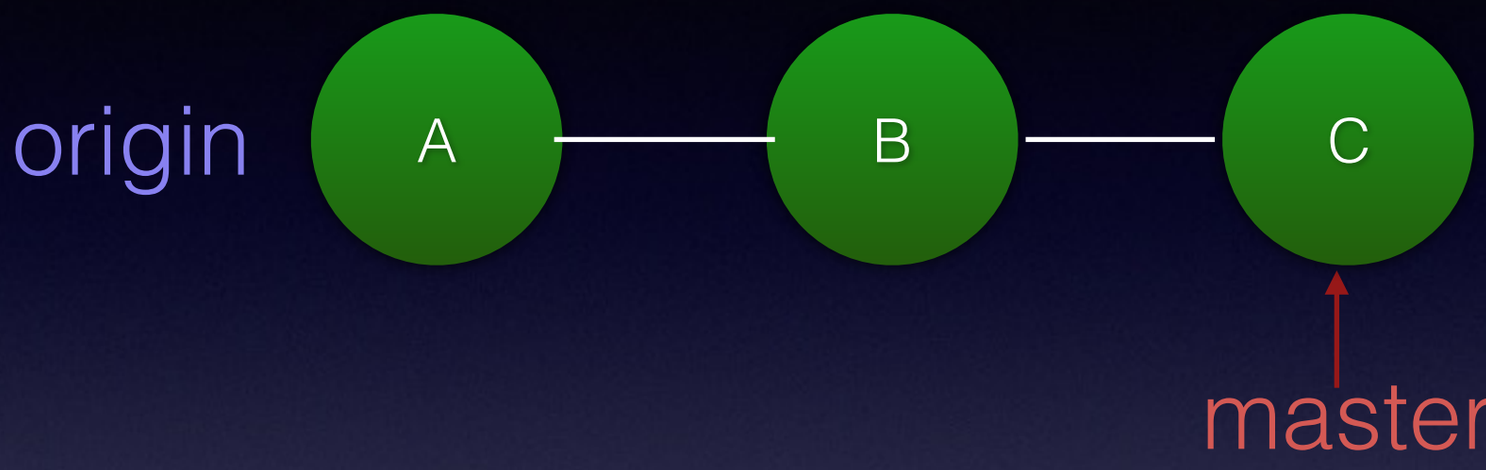
Remote repositories

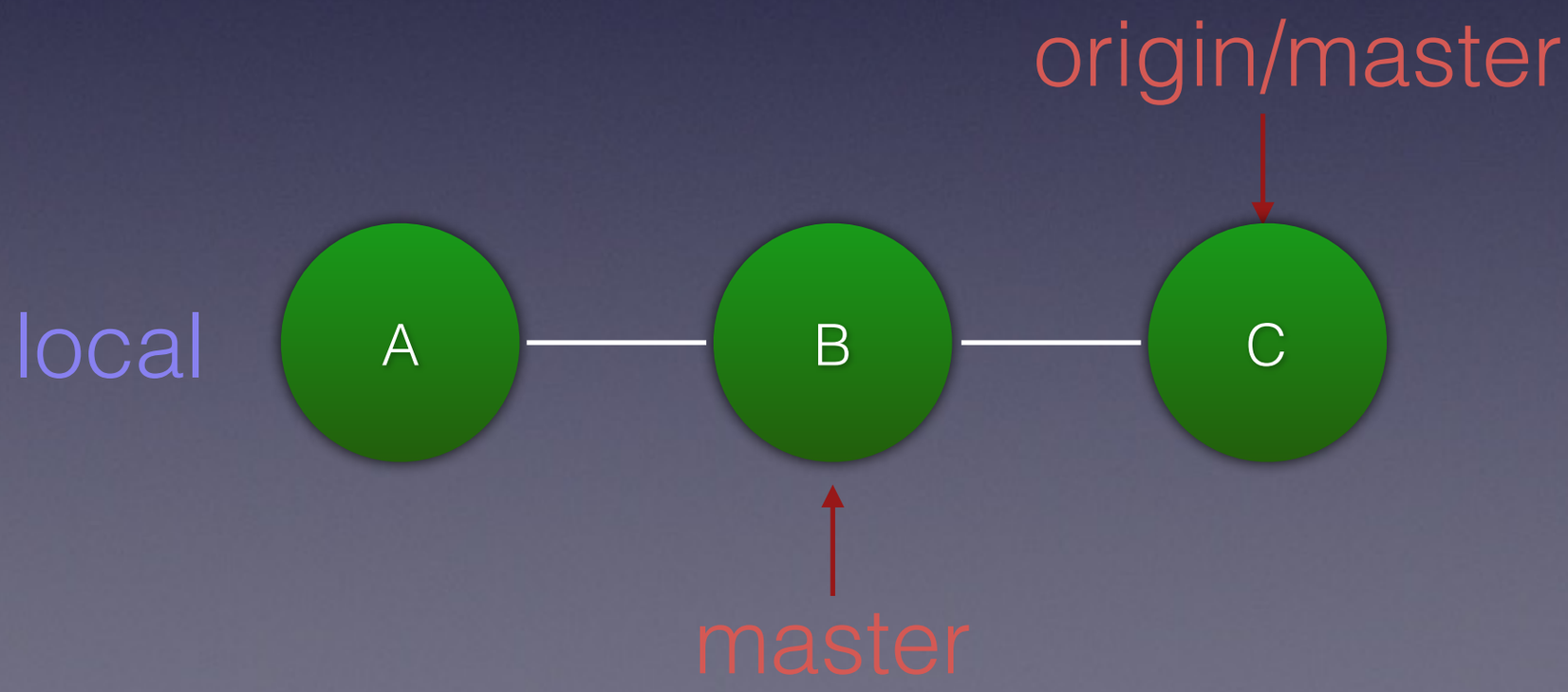


git fetch

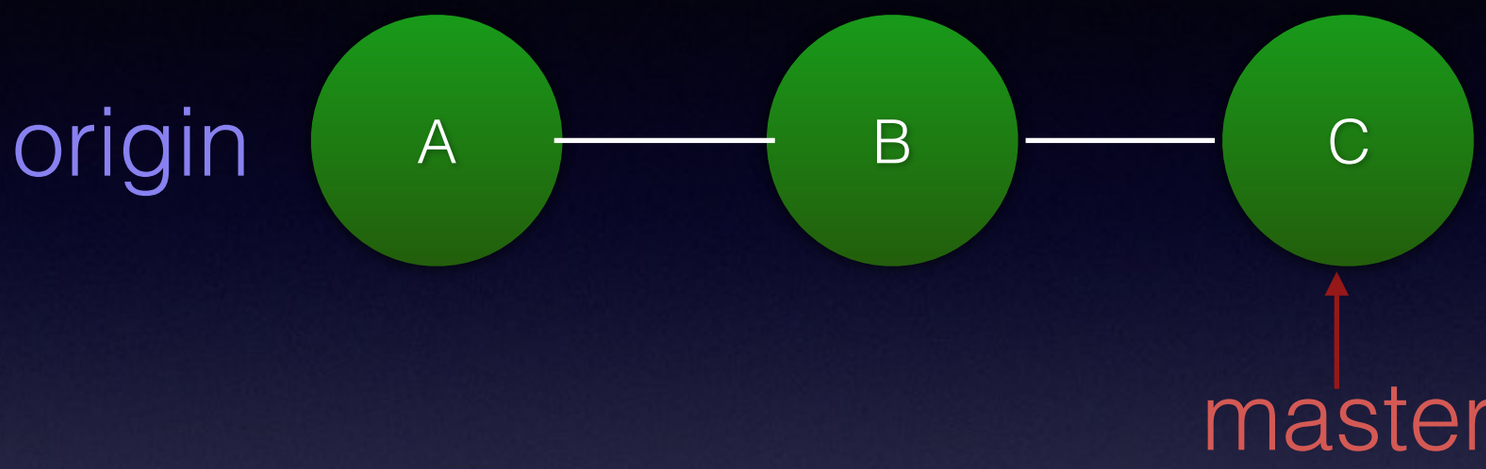


Remote repositories

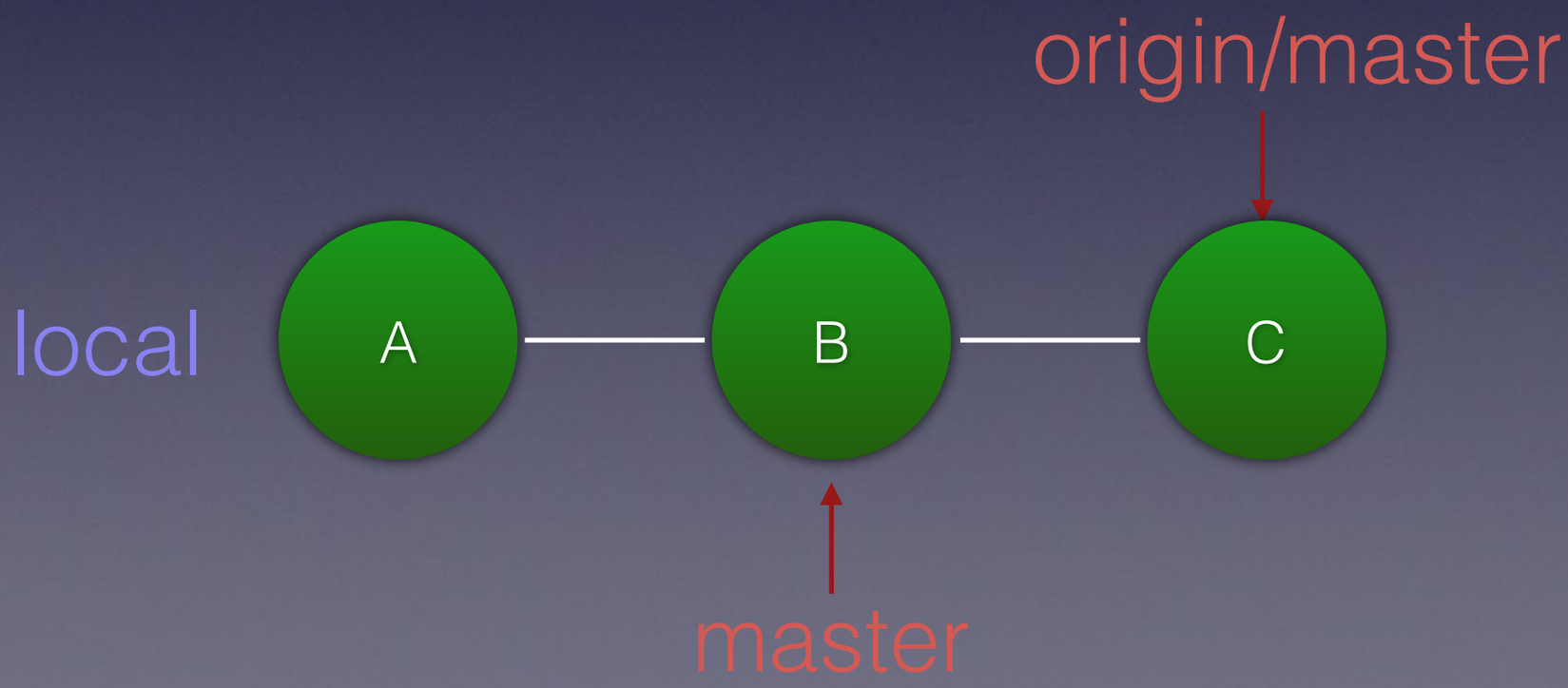




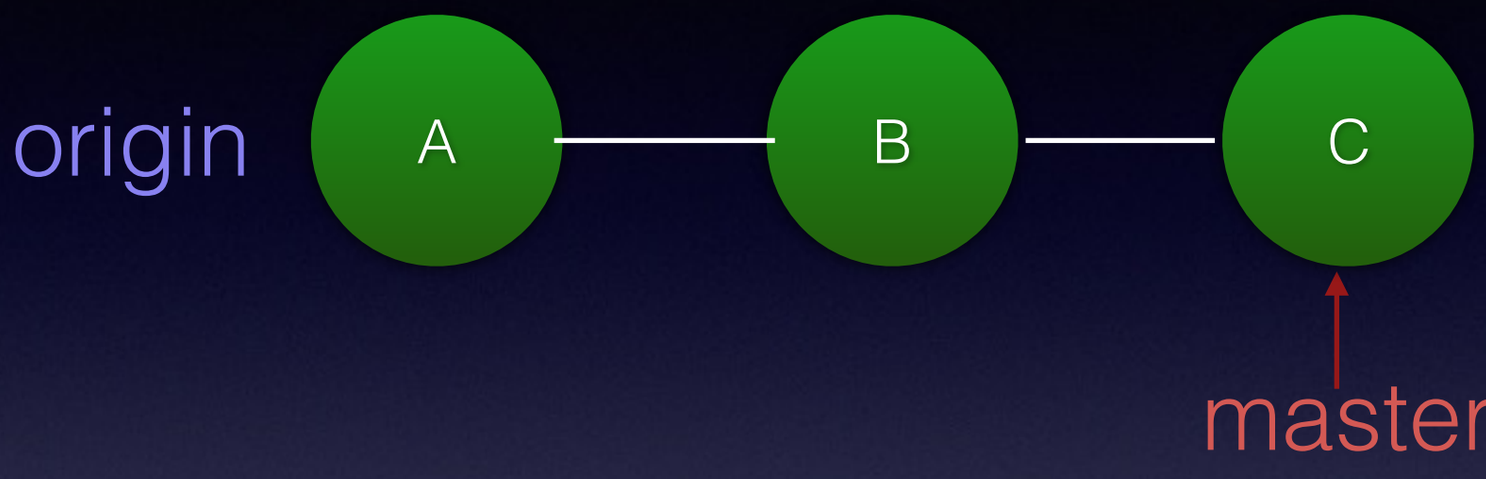
Remote repositories

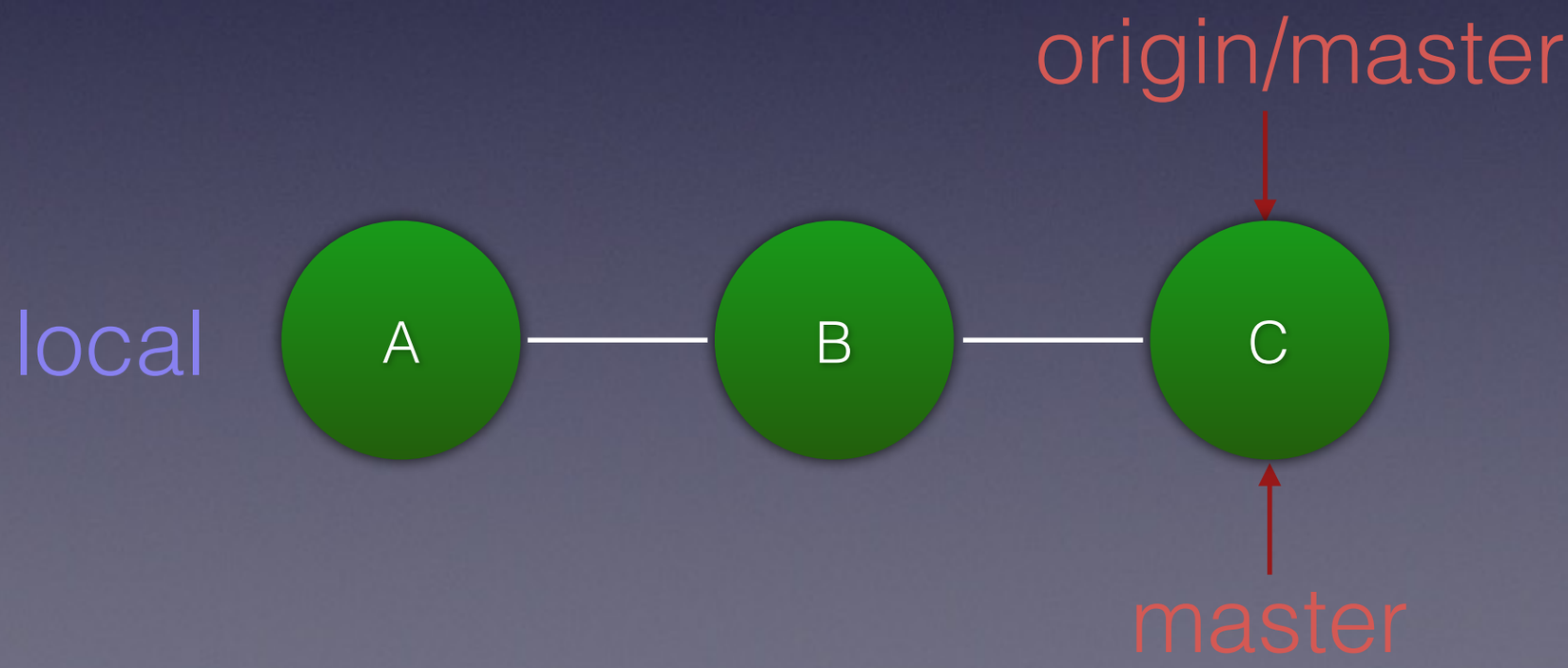


git merge origin/master

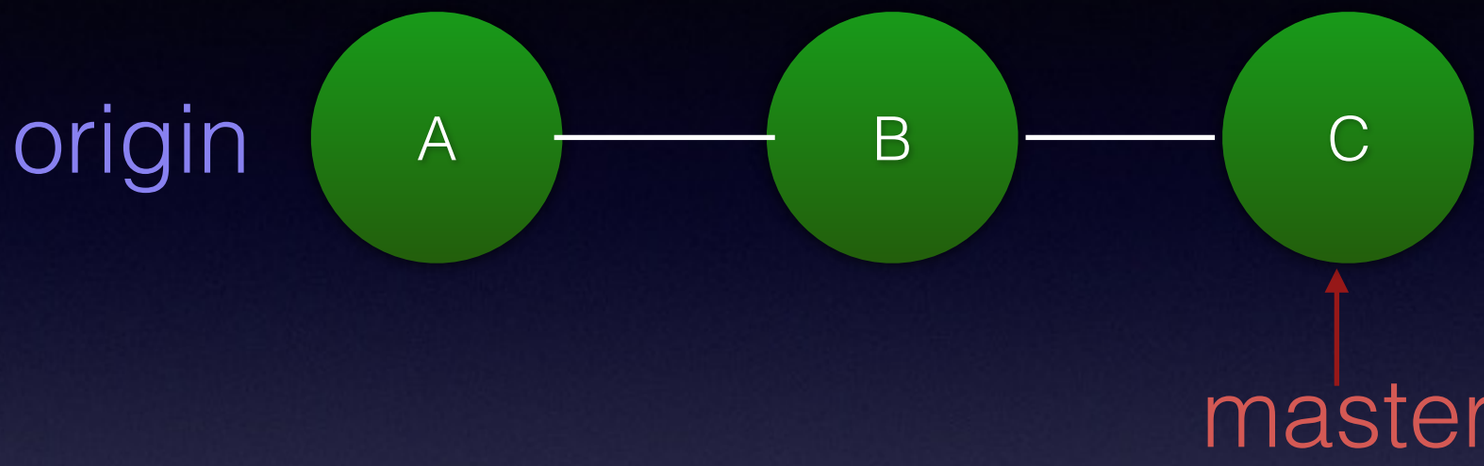


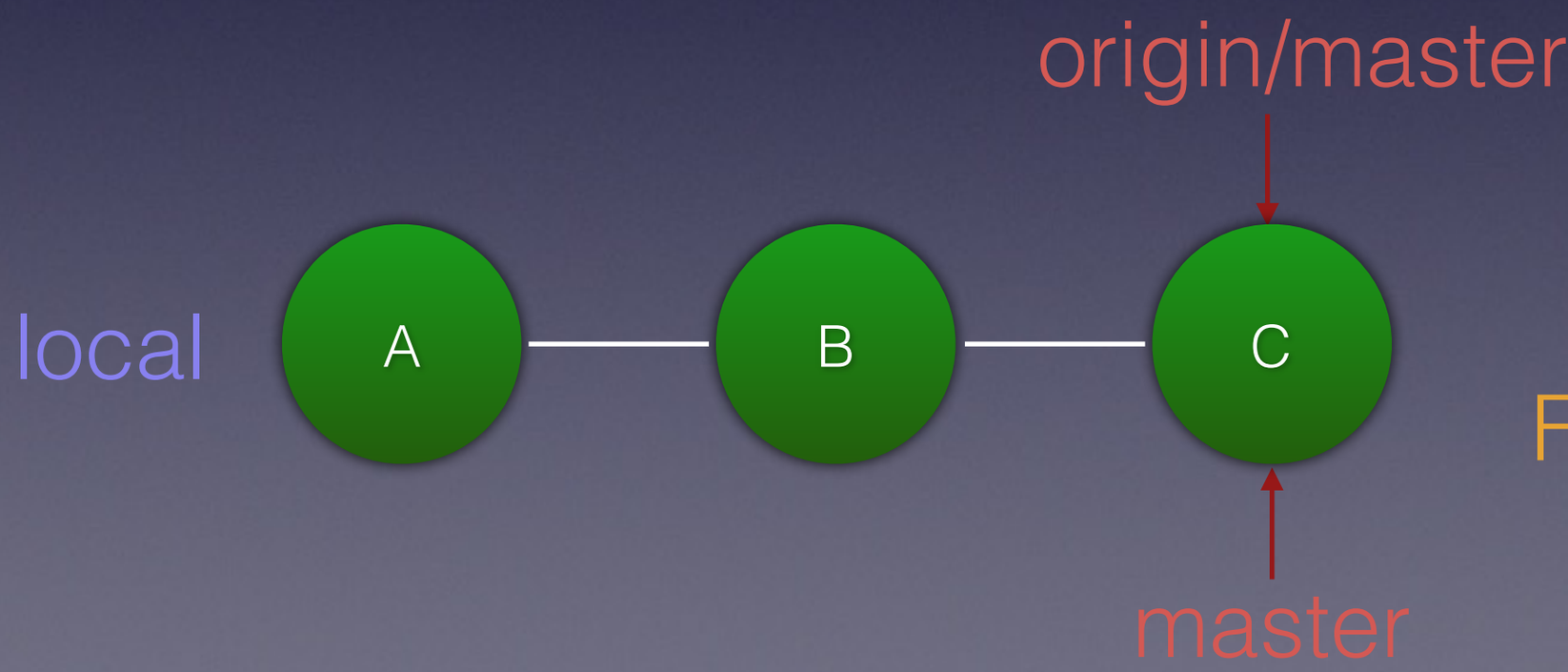
Remote repositories





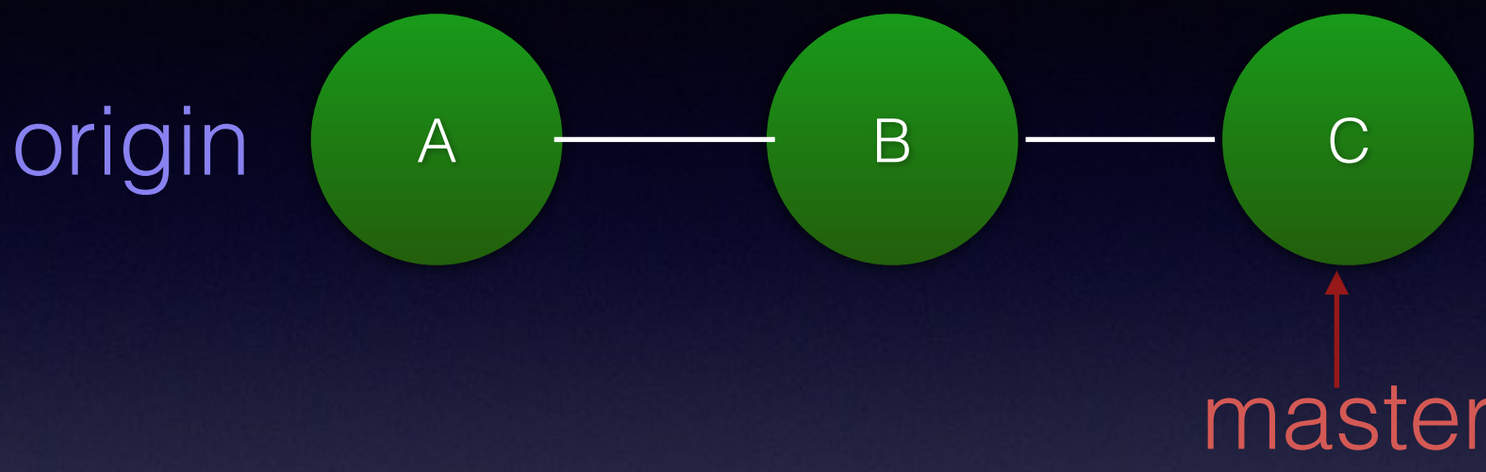
Remote repositories

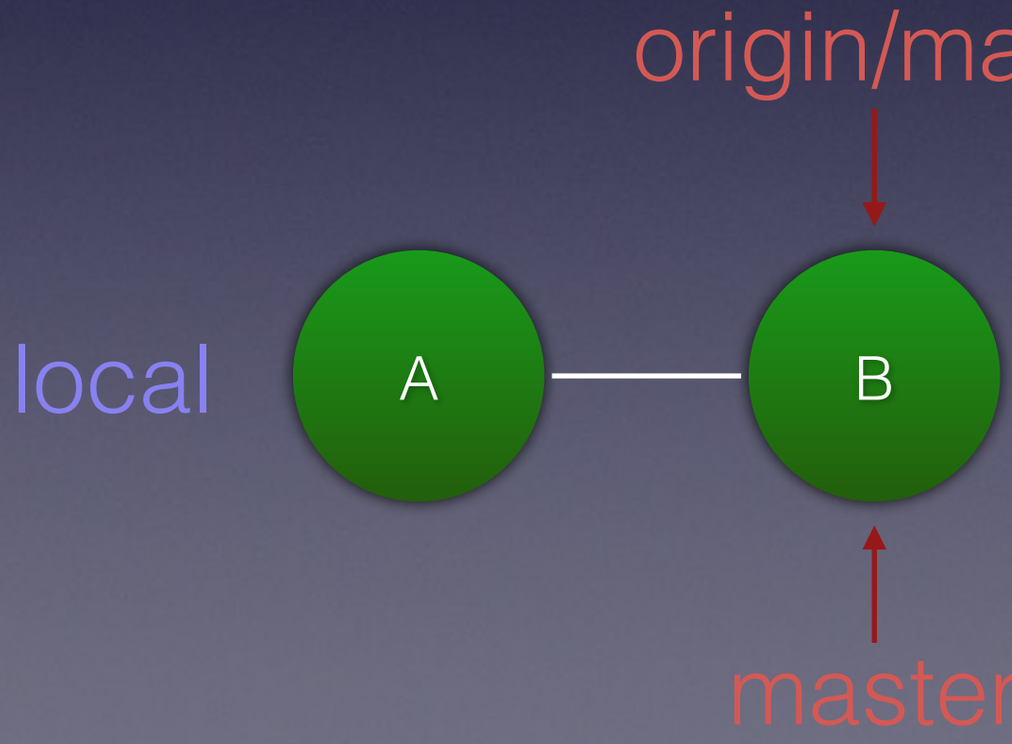




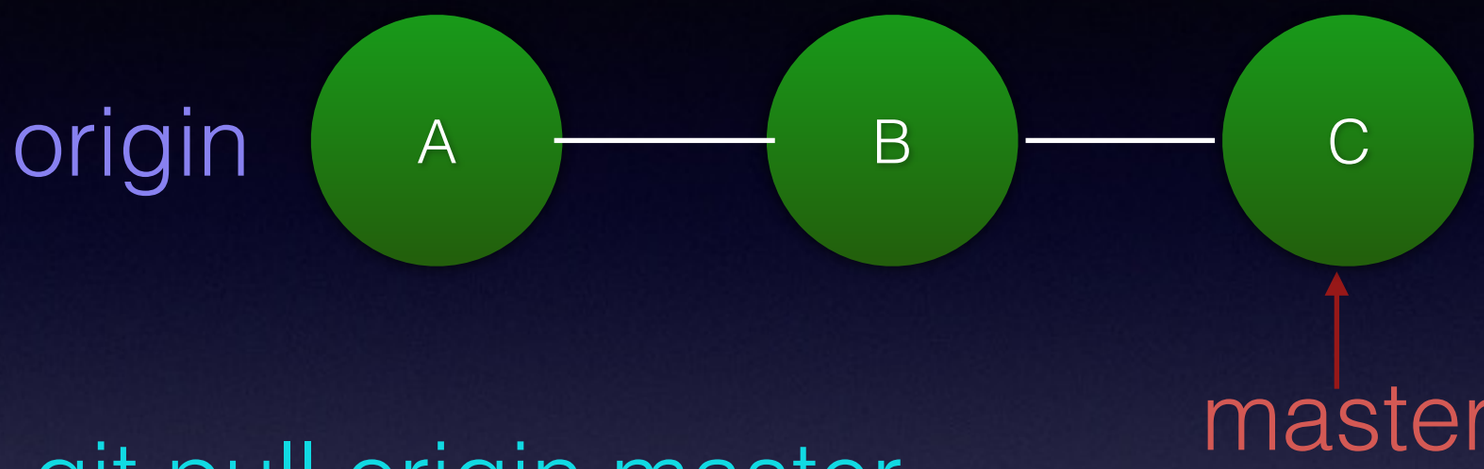
Fast forward merge

Remote repositories

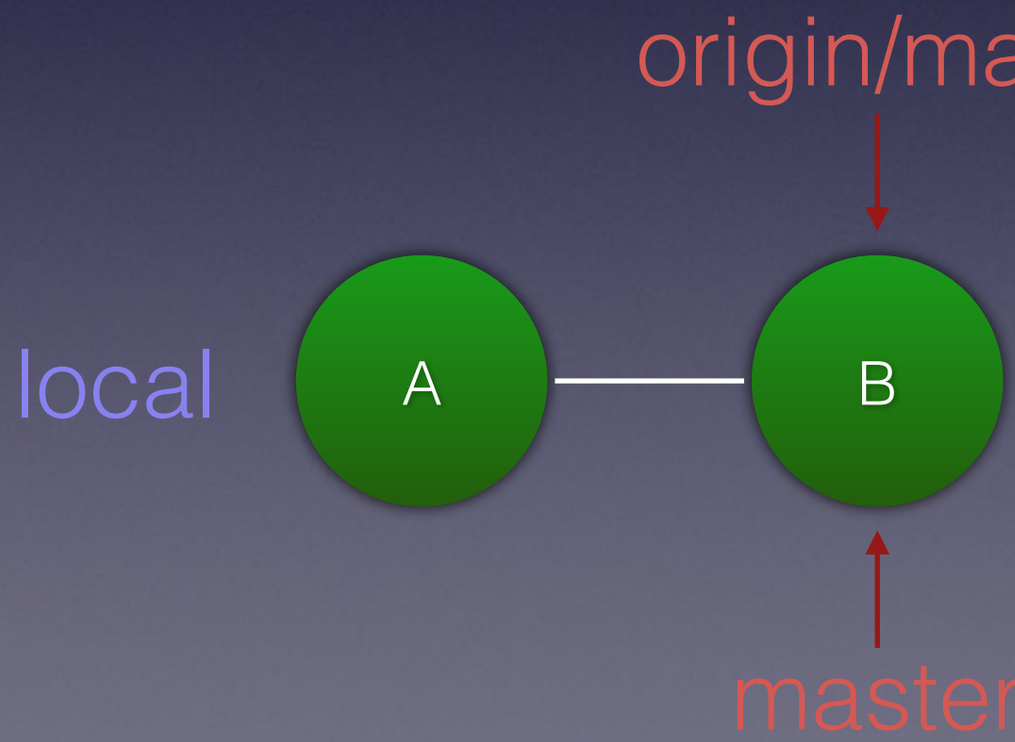




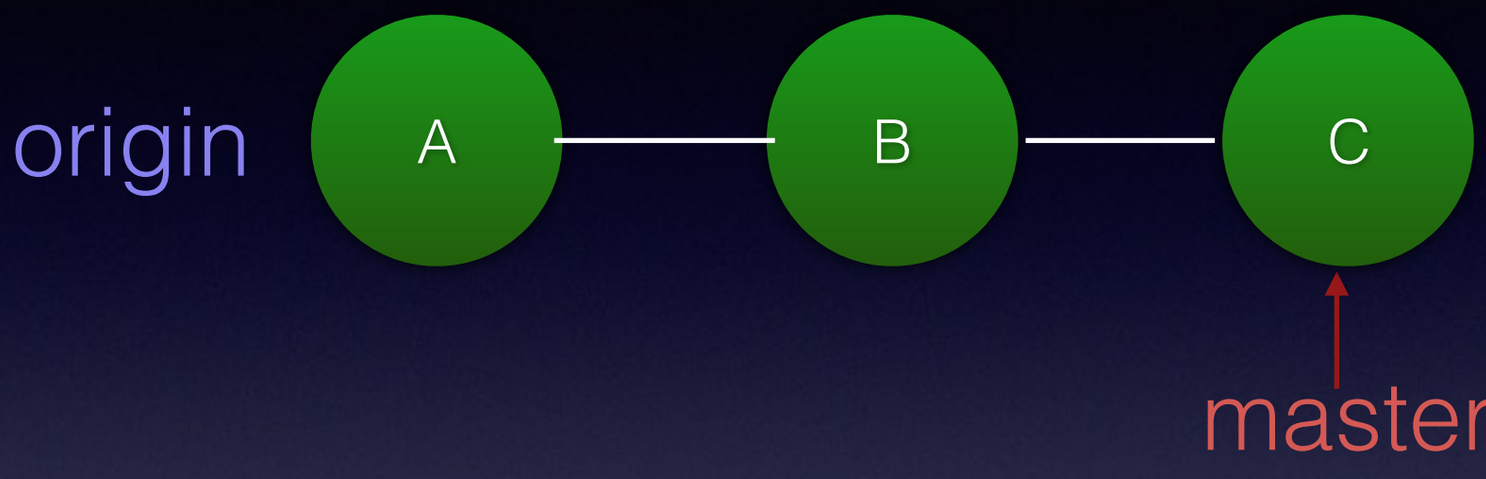
Remote repositories

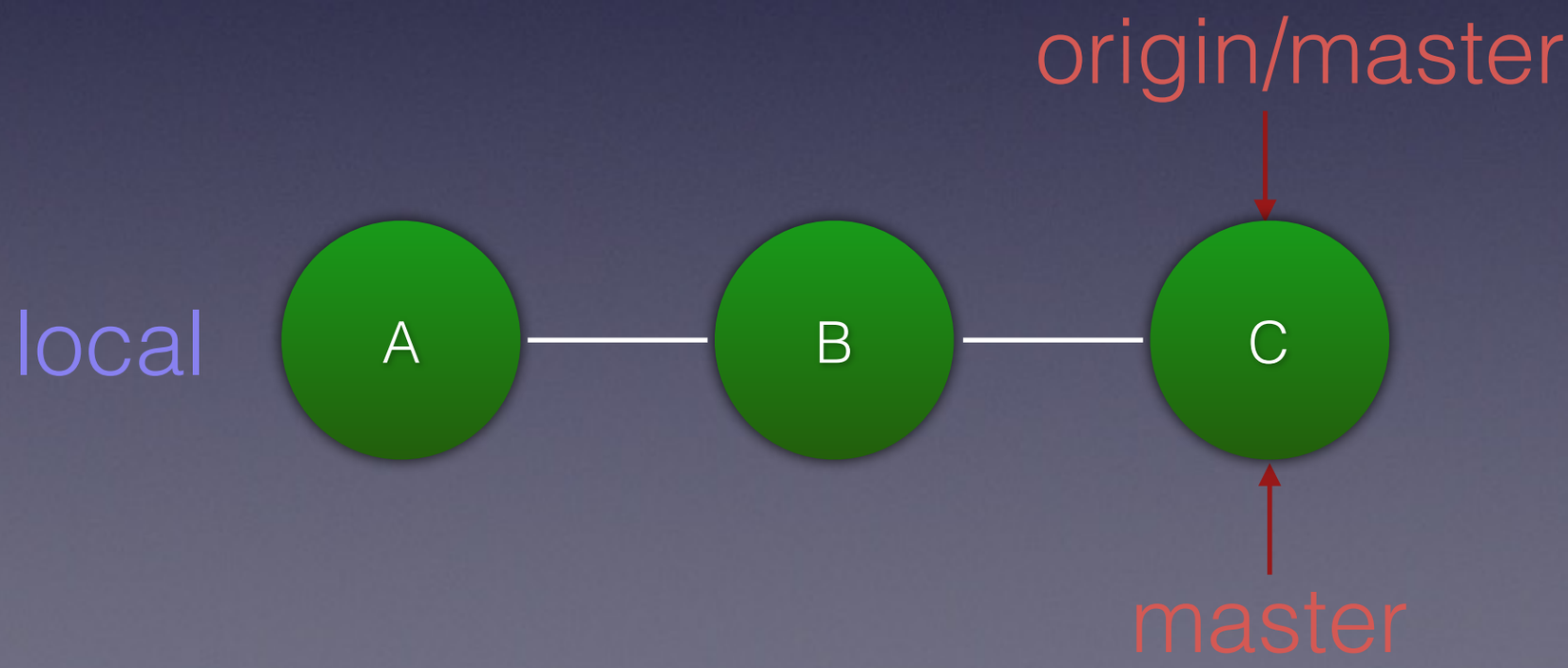


git pull origin master

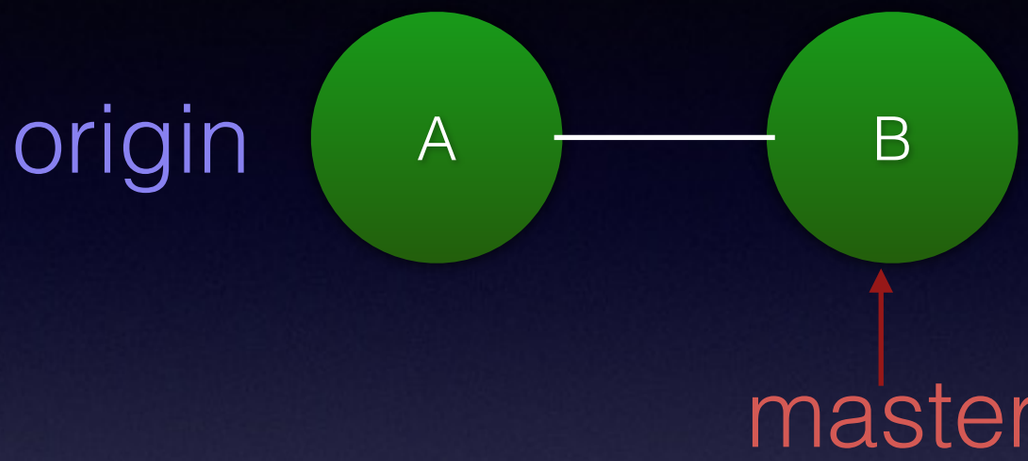


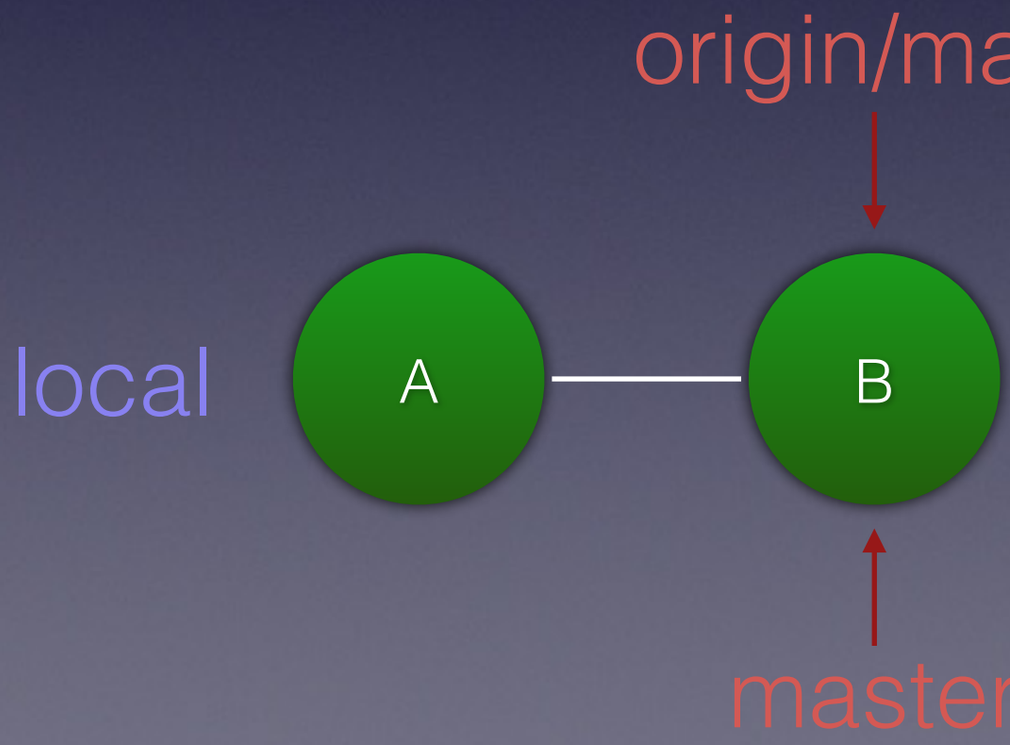
Remote repositories



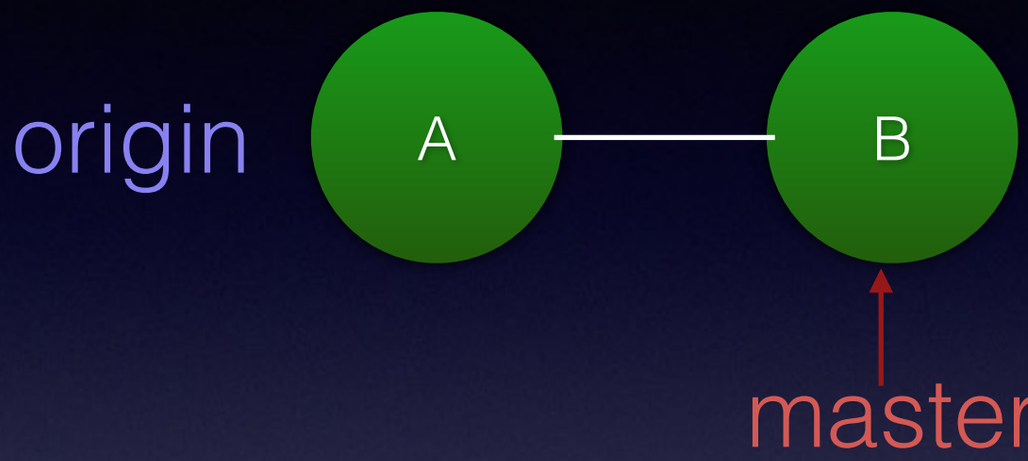


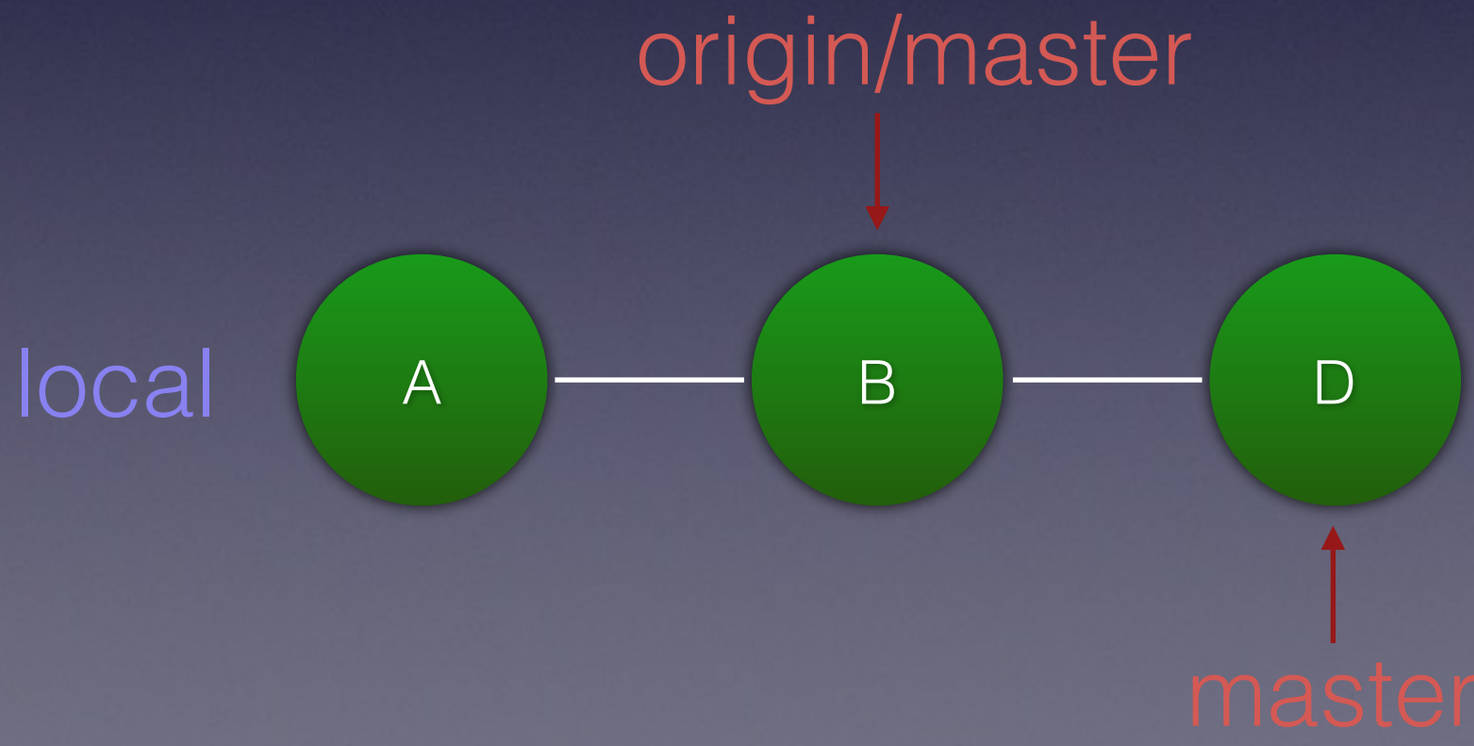
Remote repositories



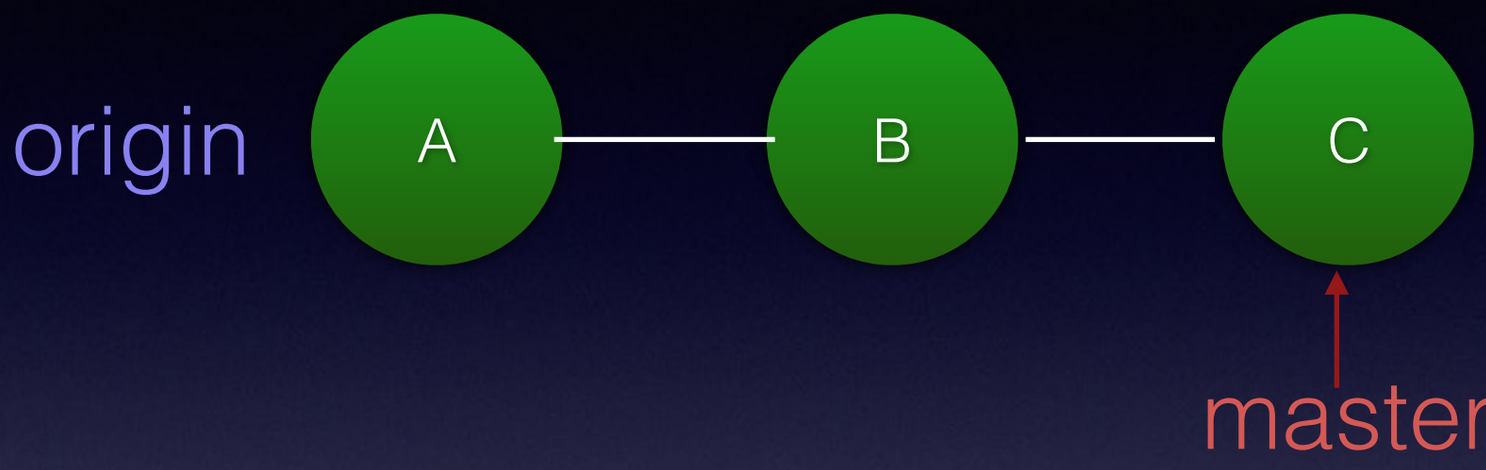


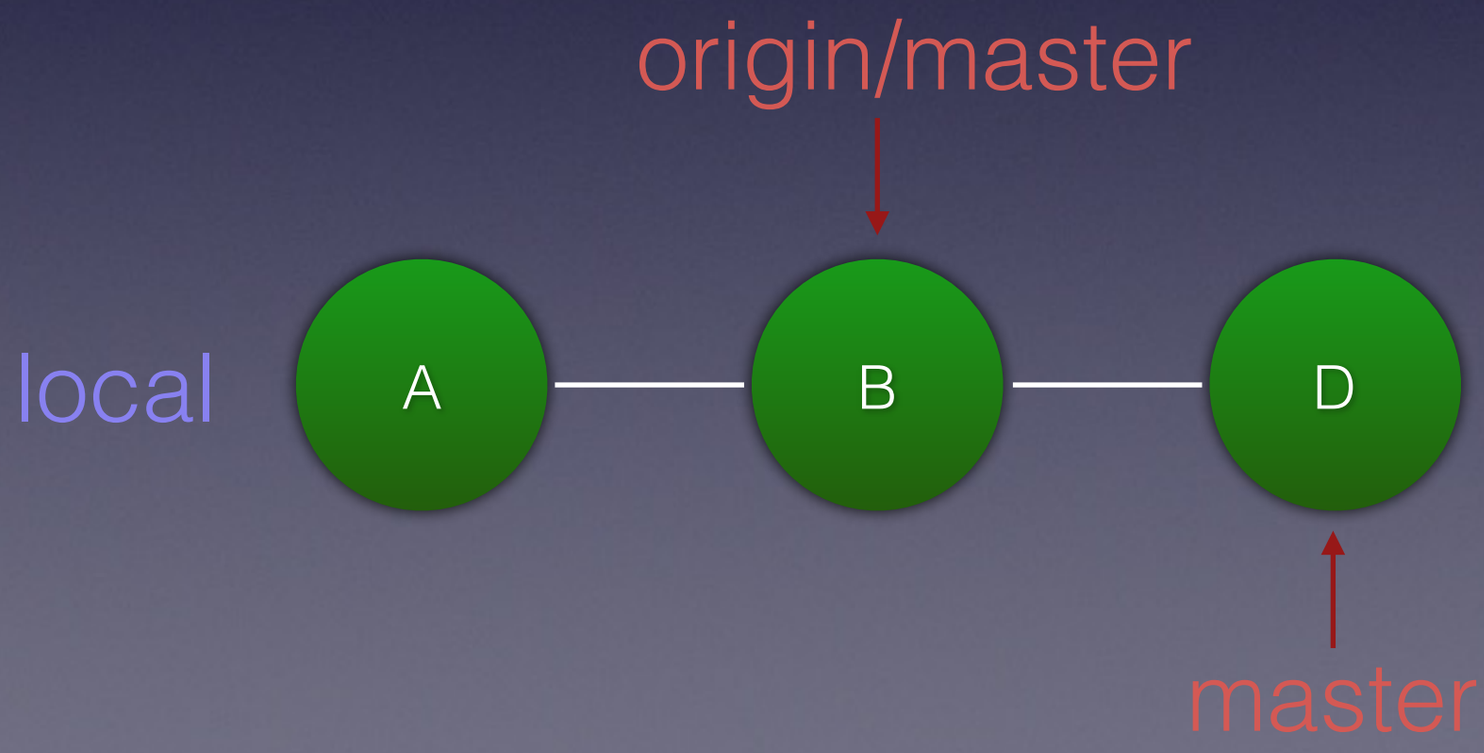
Remote repositories



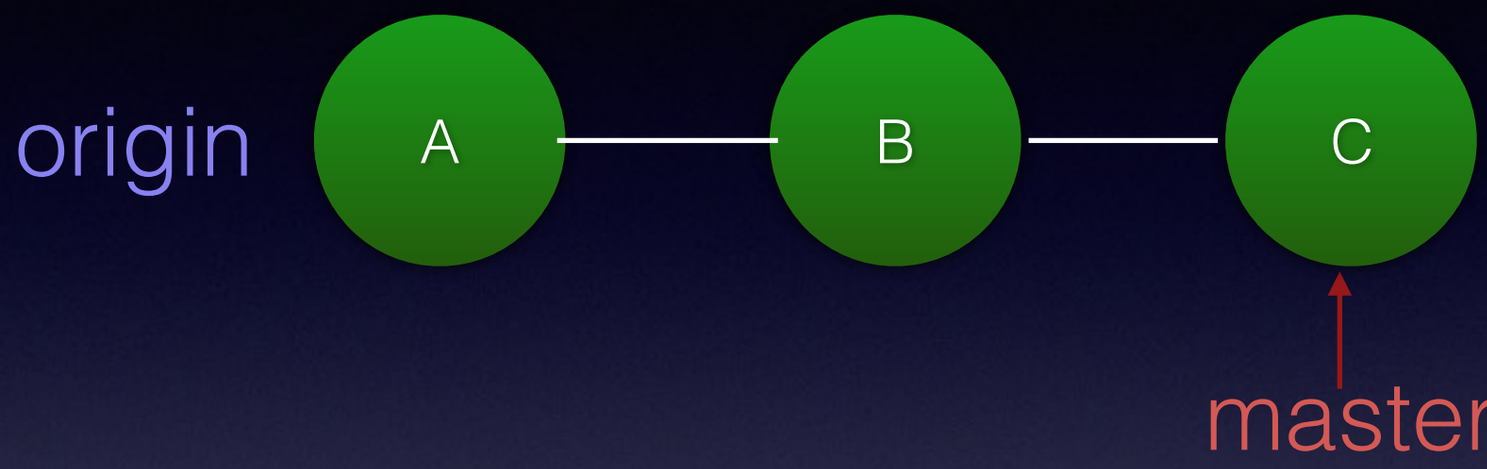


Remote repositories

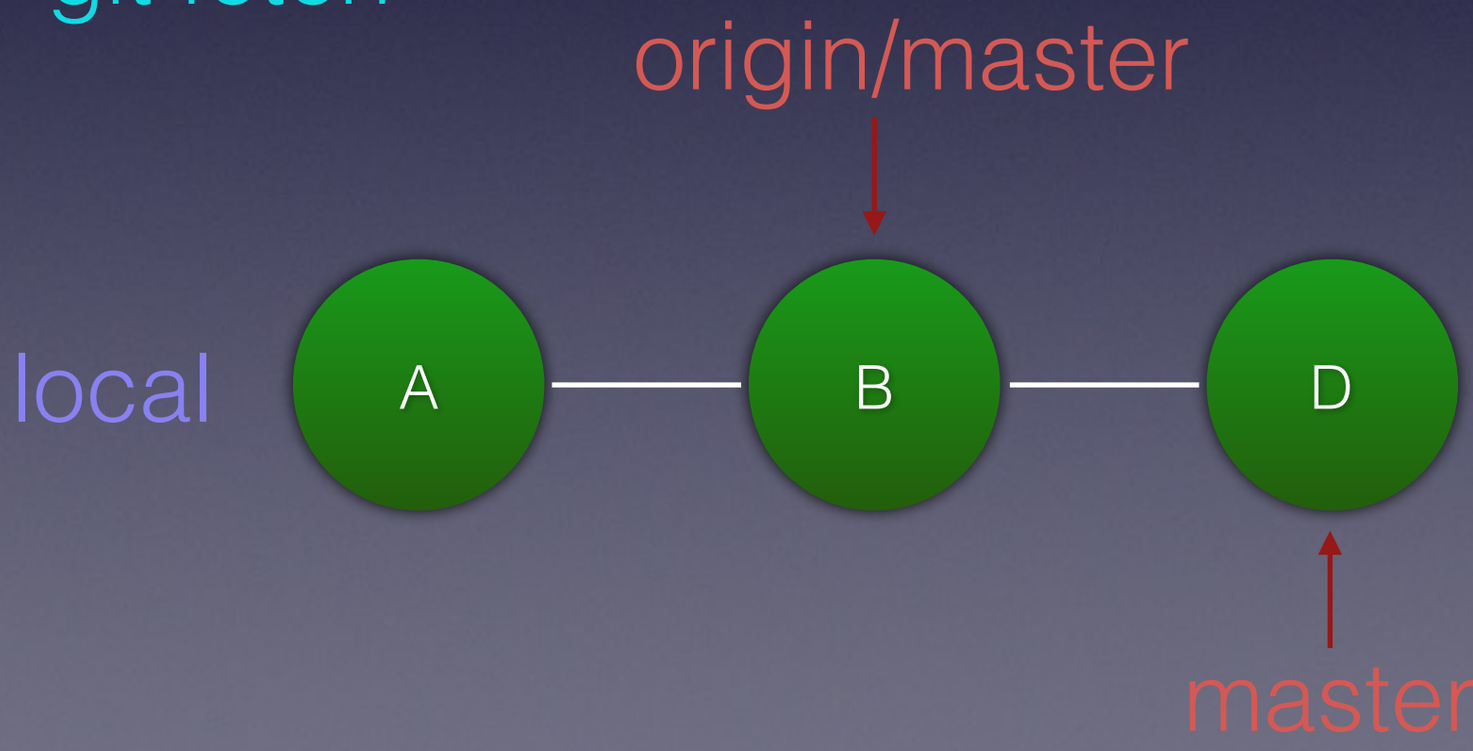




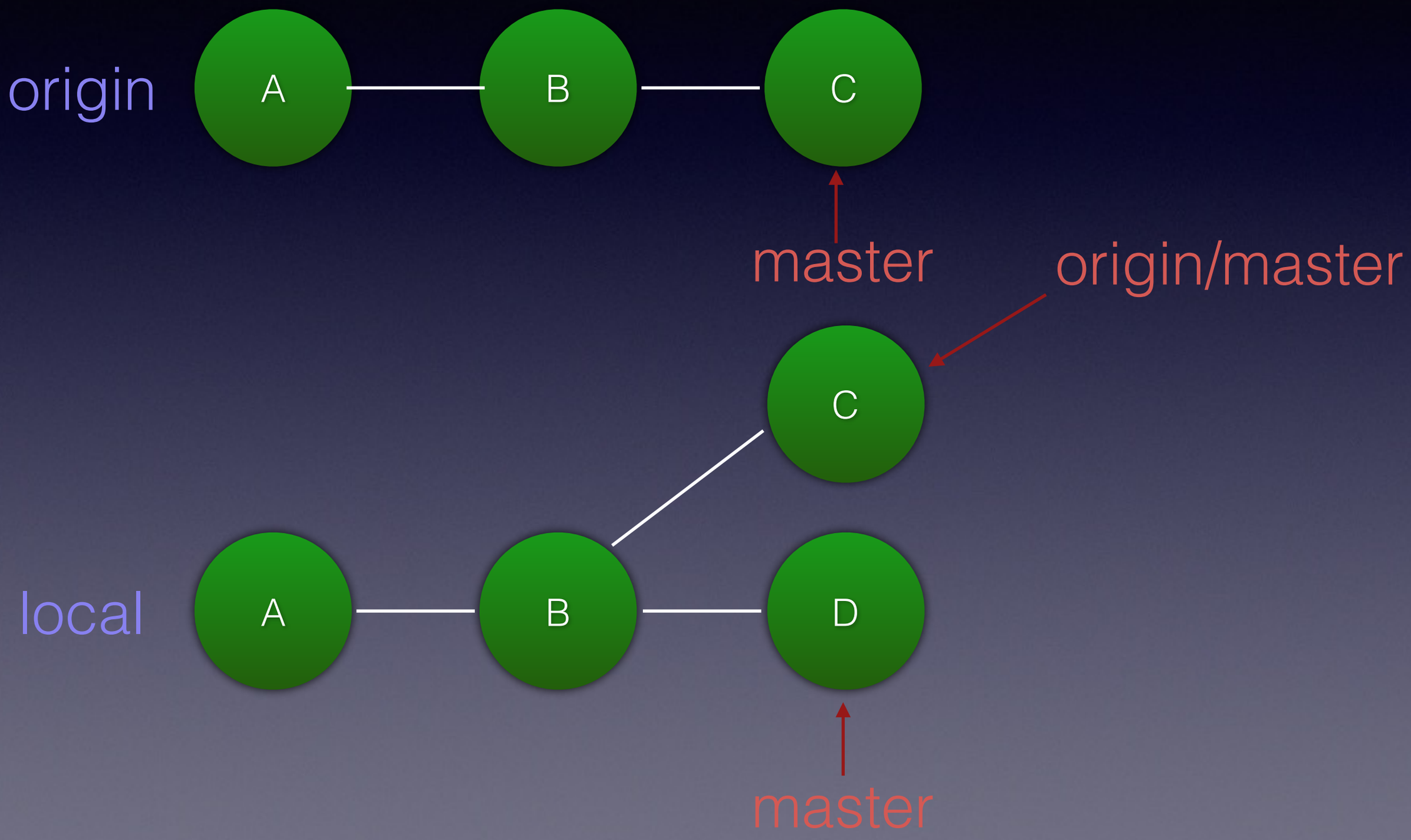
Remote repositories



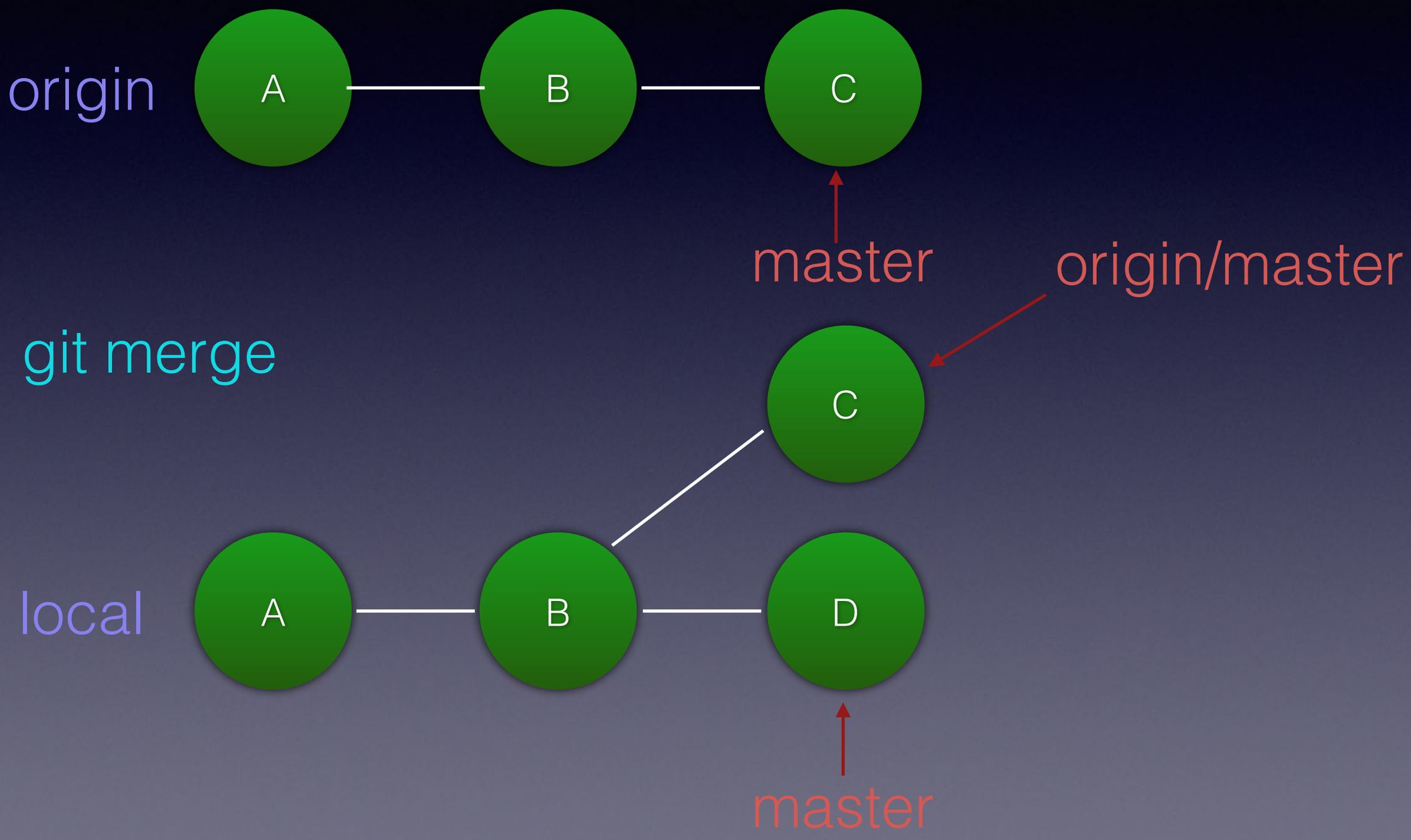
git fetch



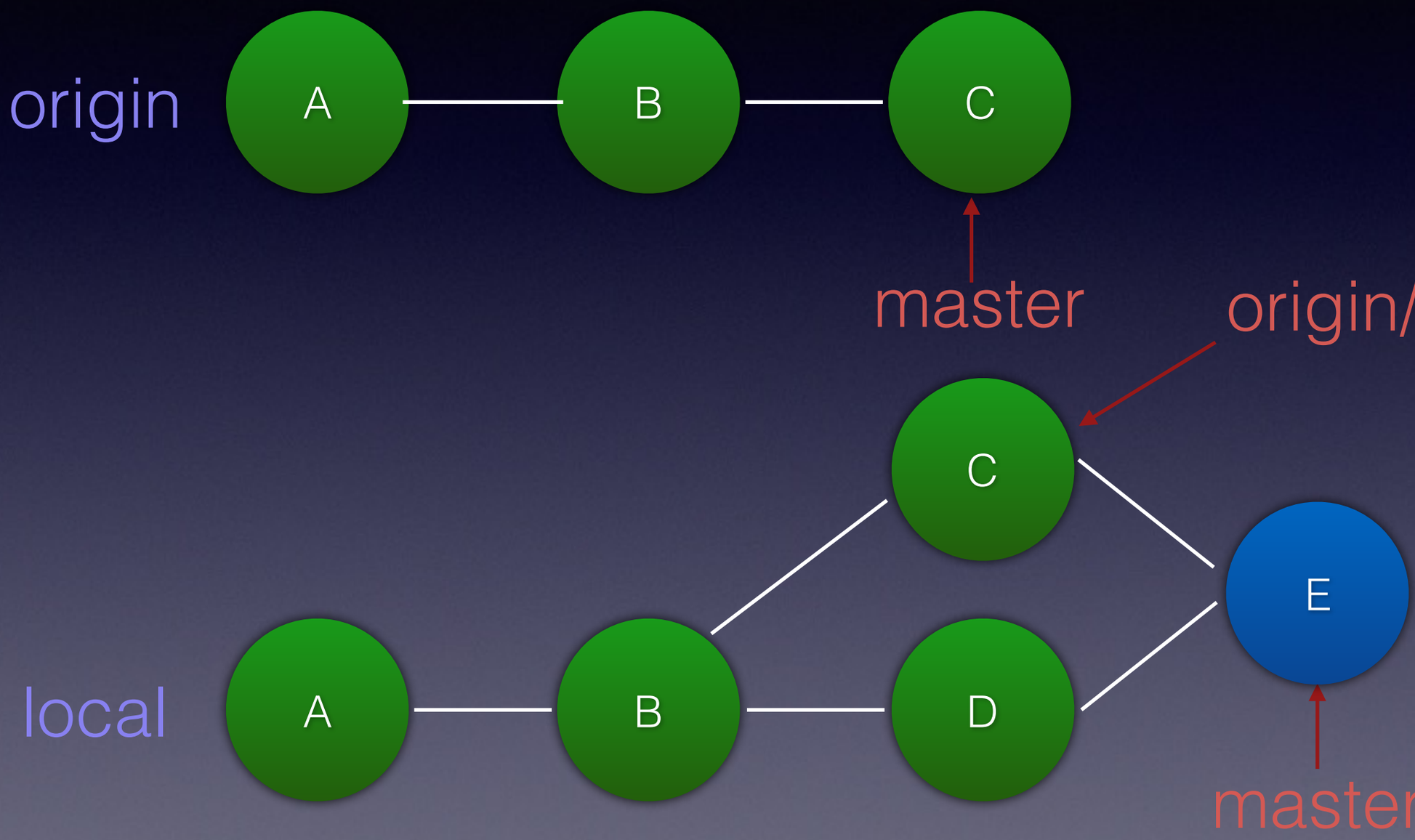
Remote repositories



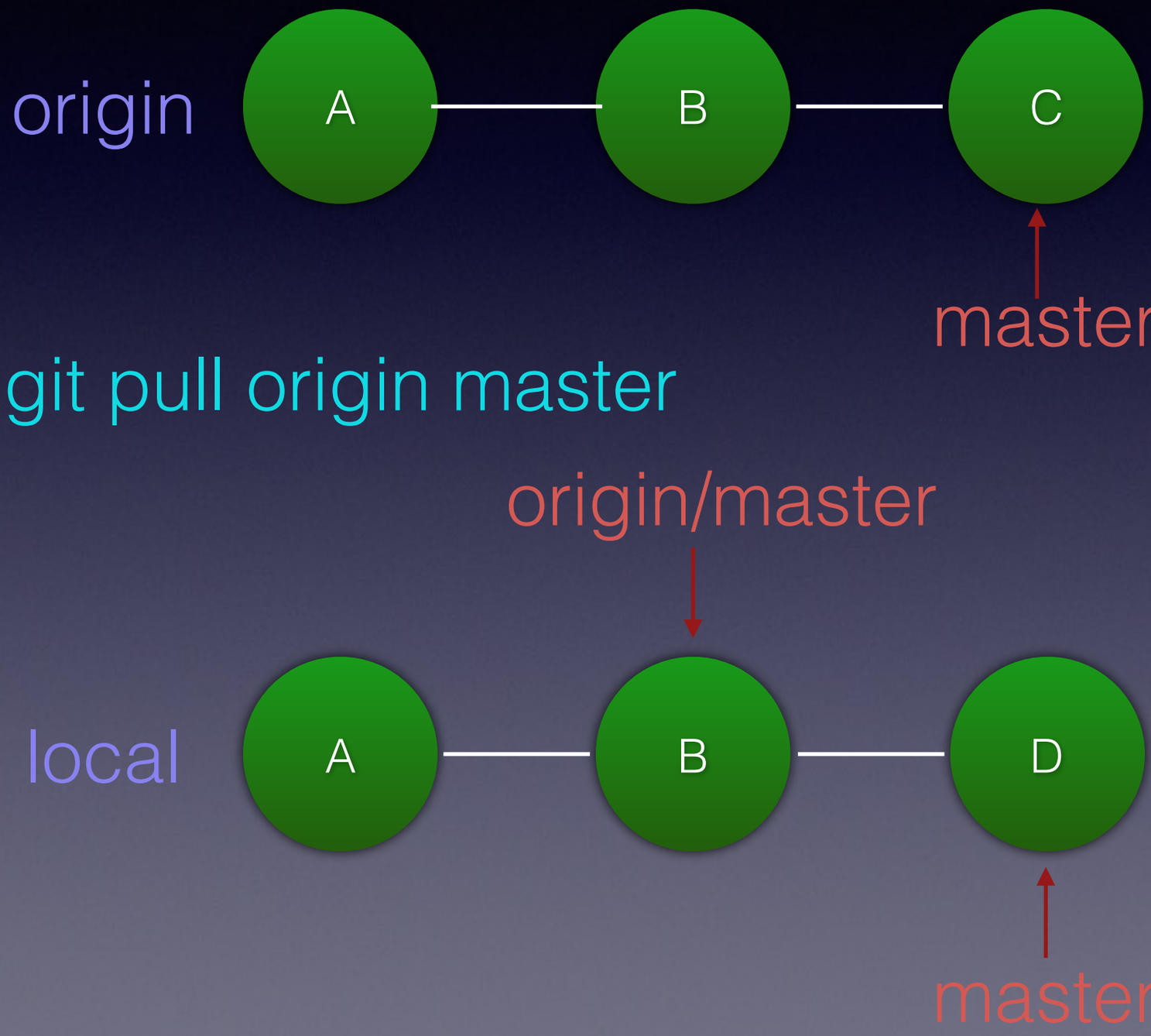
Remote repositories



Remote repositories

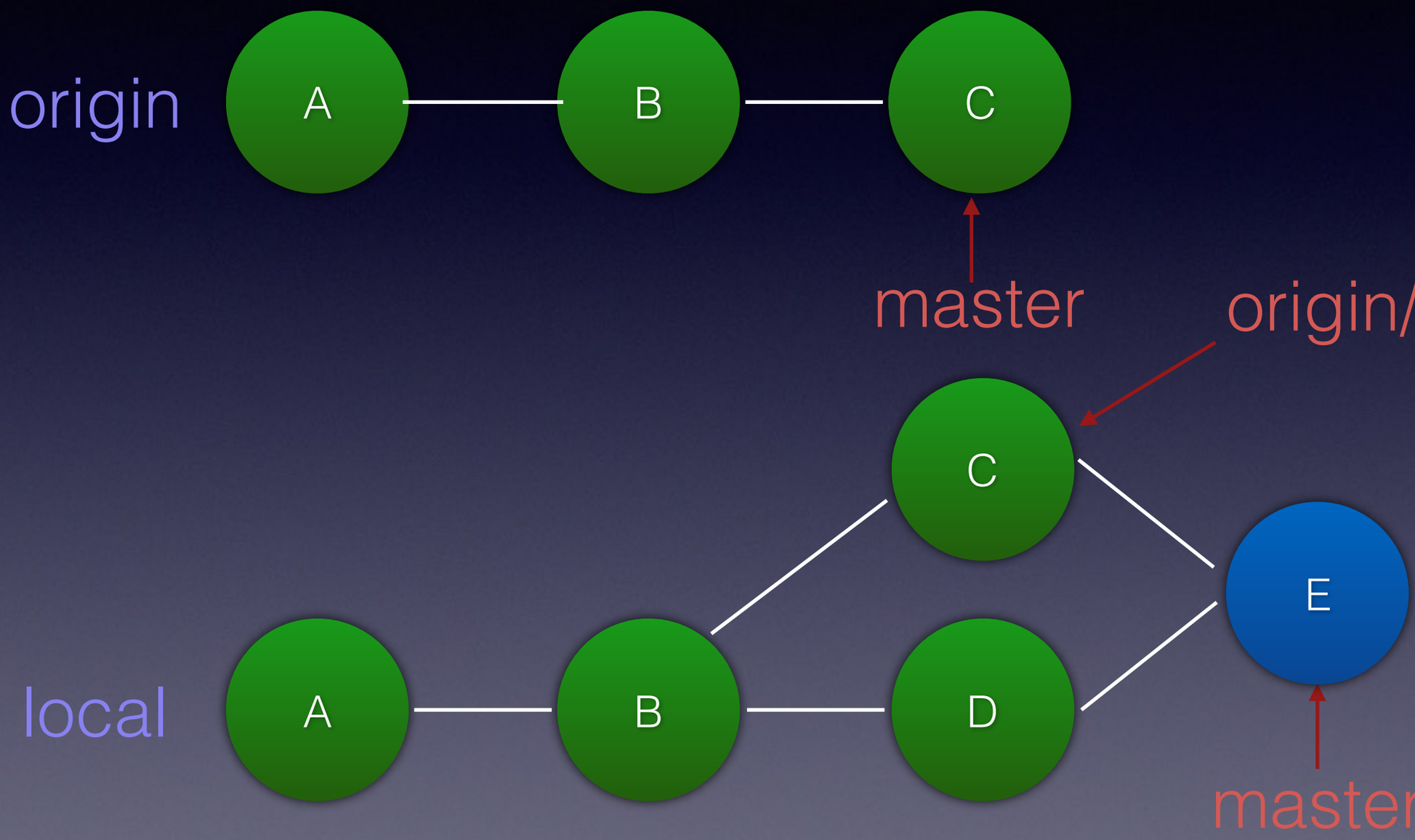


Remote repositories

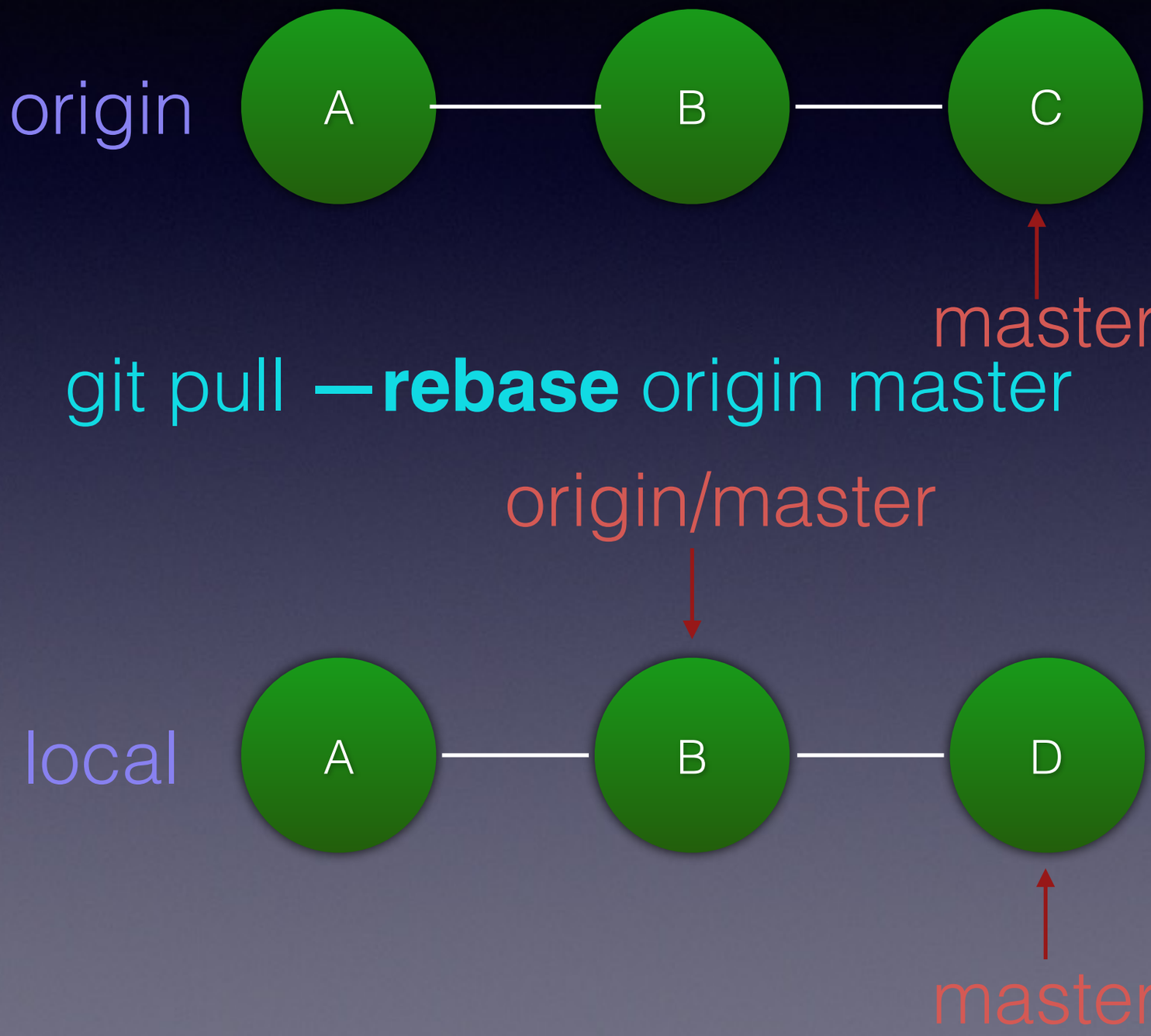


git pull origin master

Remote repositories

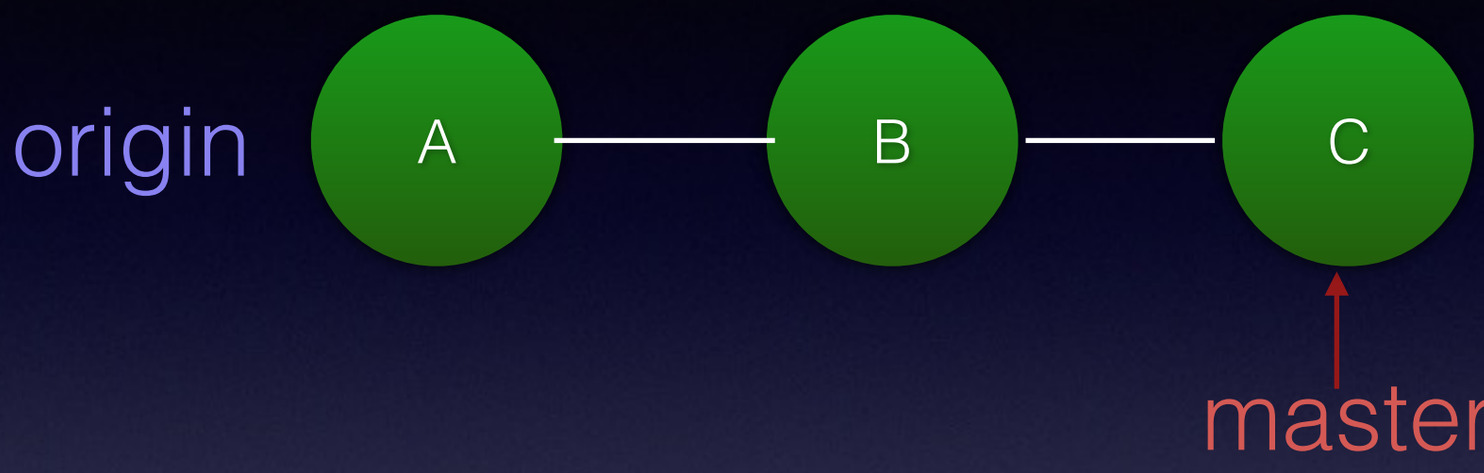


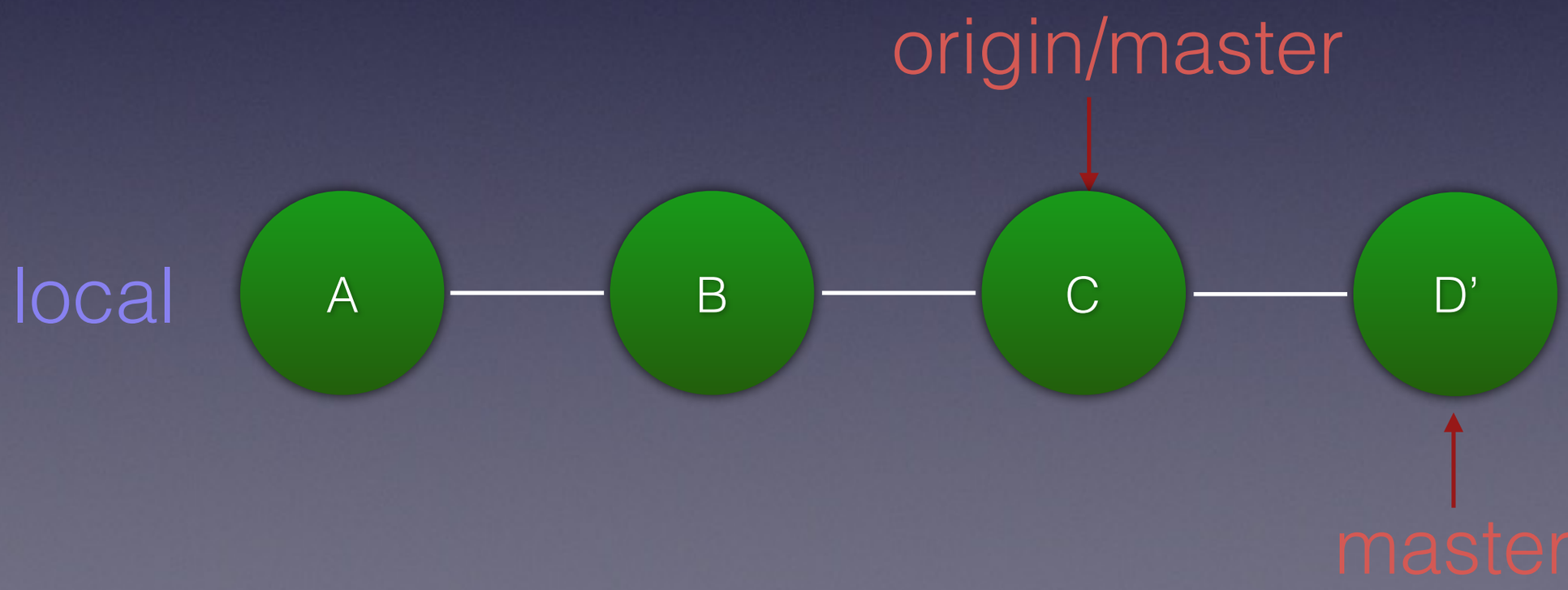
Remote repositories



git pull **—rebase** origin master

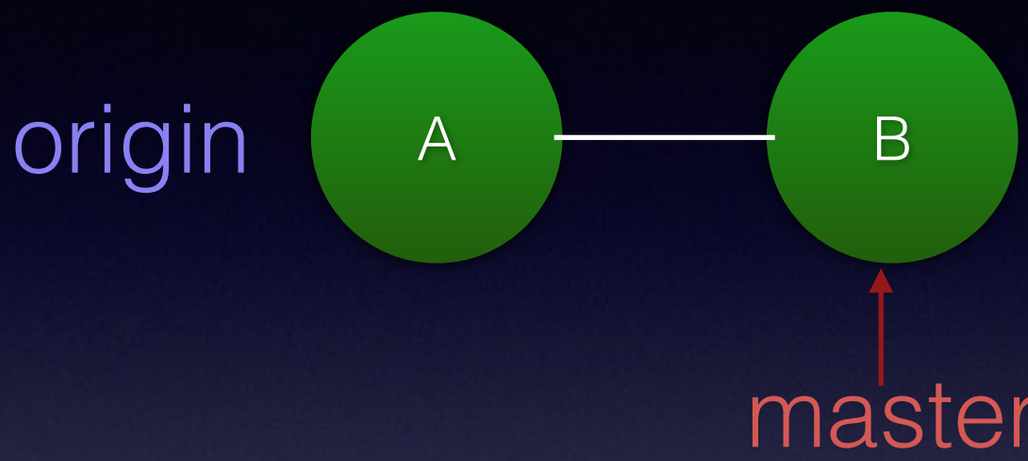
Remote repositories

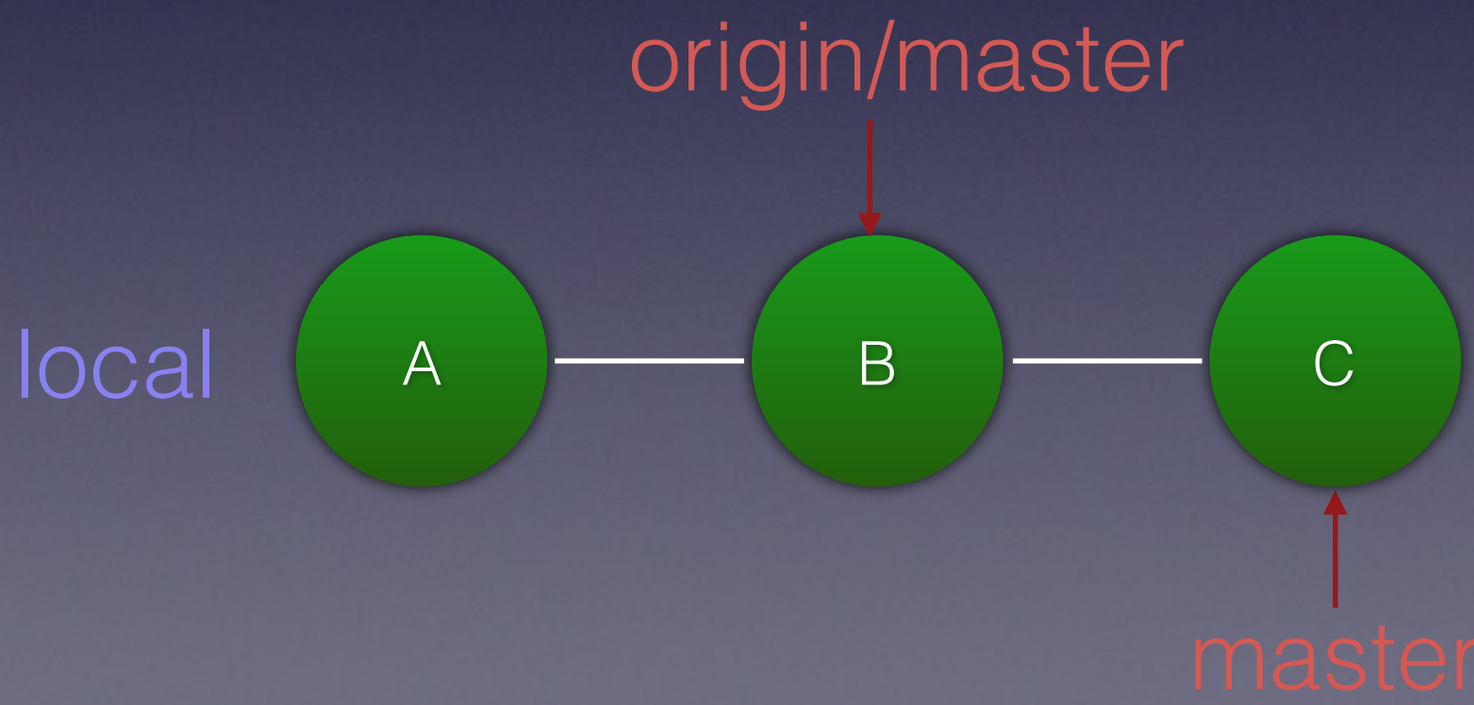




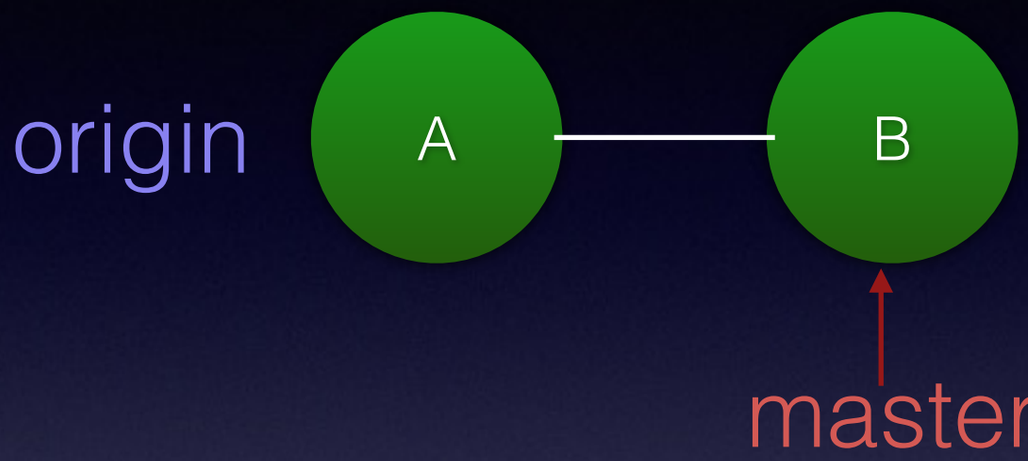
Updating remote repository

Remote repositories

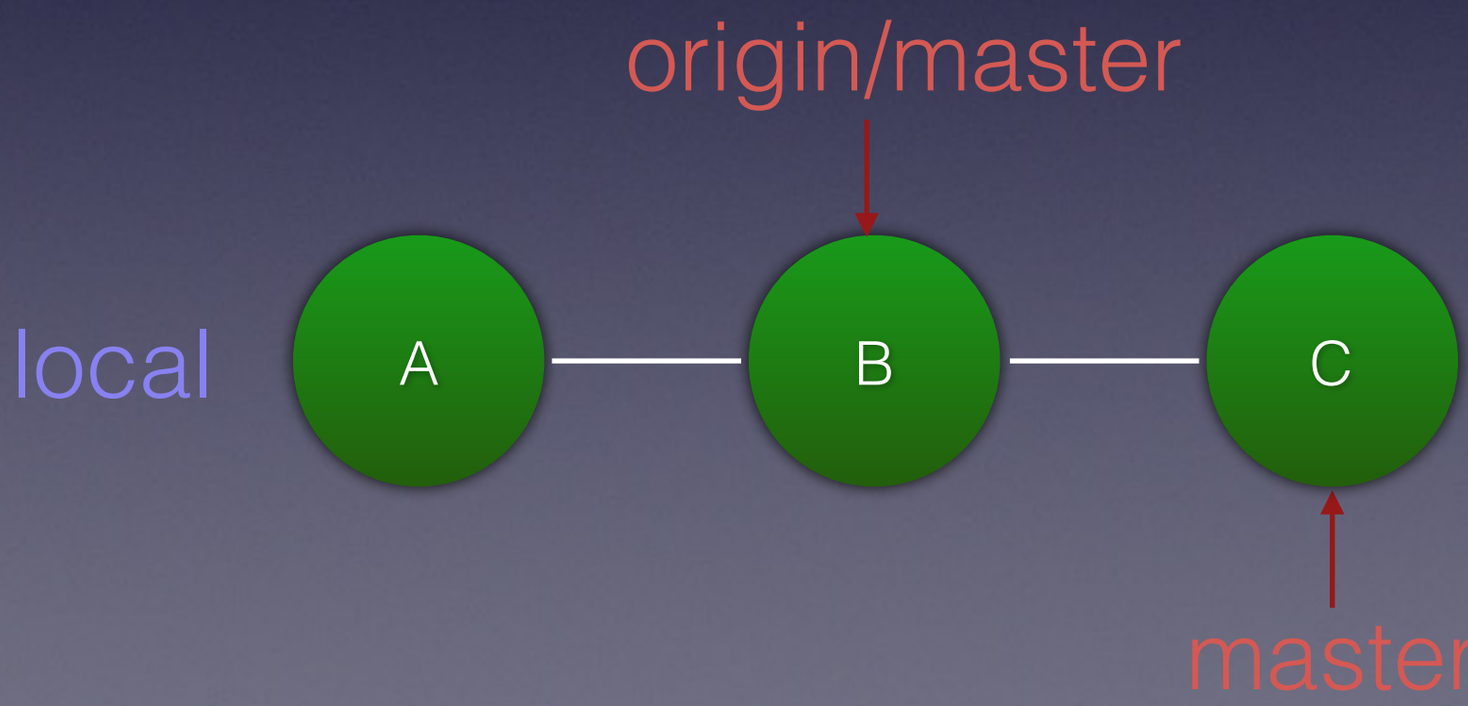




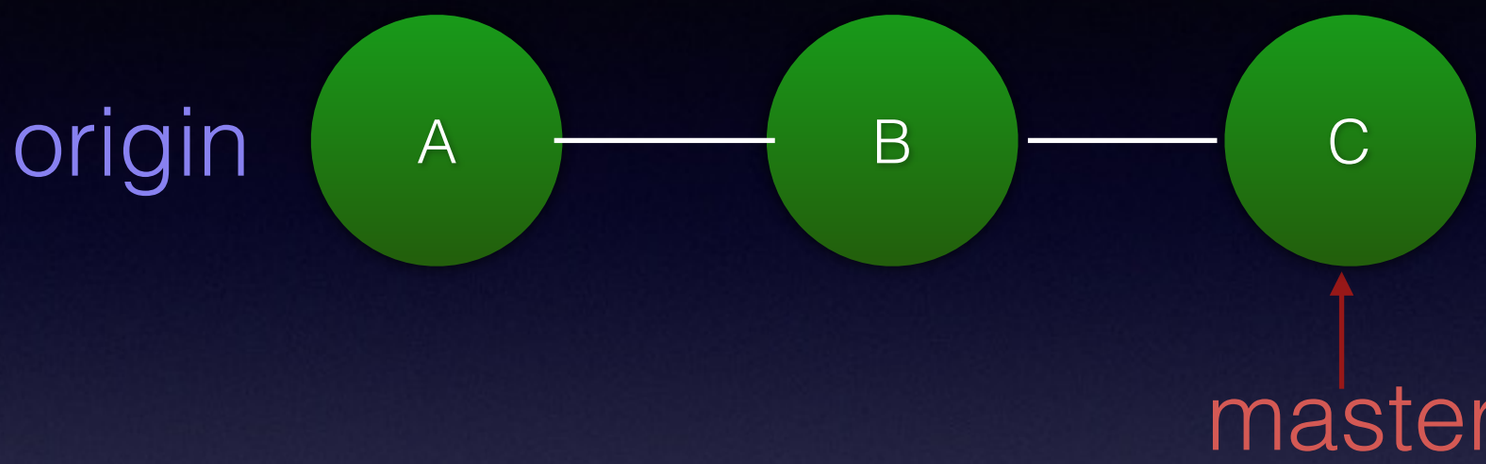
Remote repositories

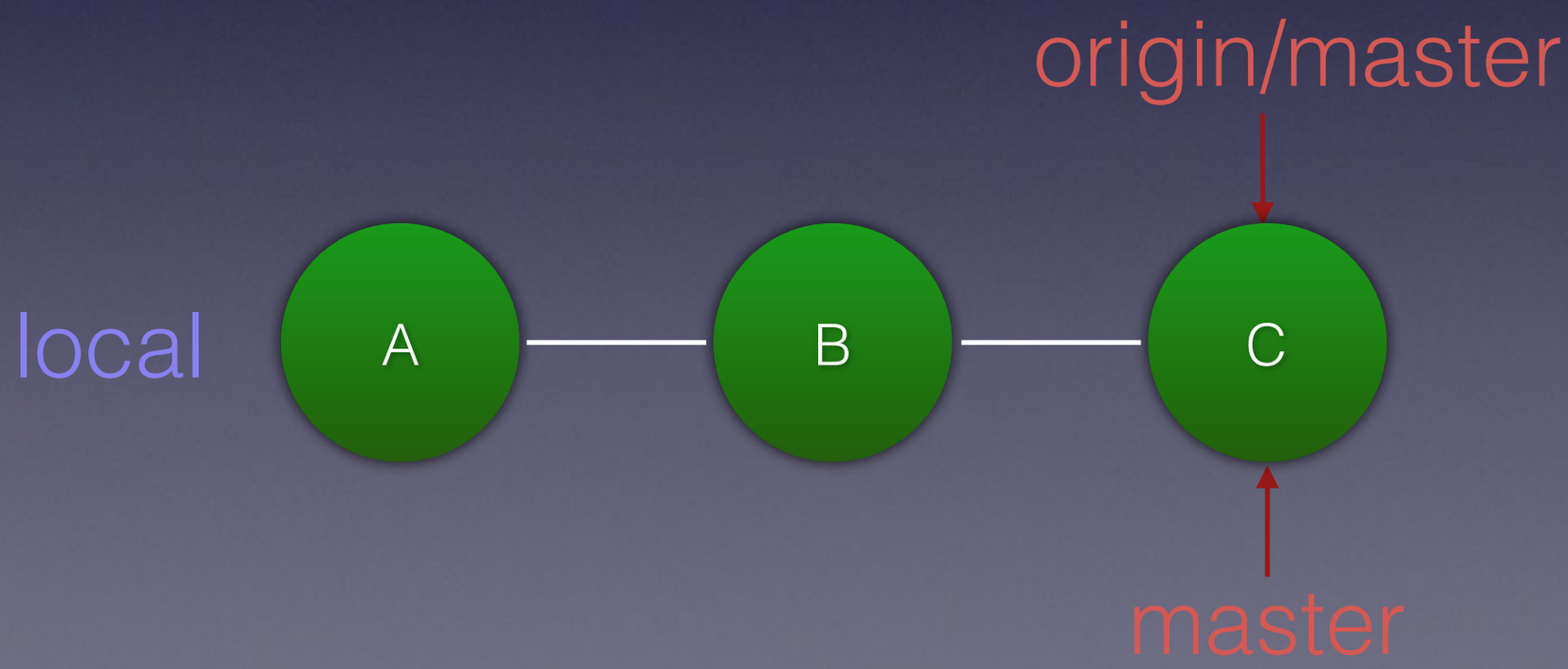


git push origin master

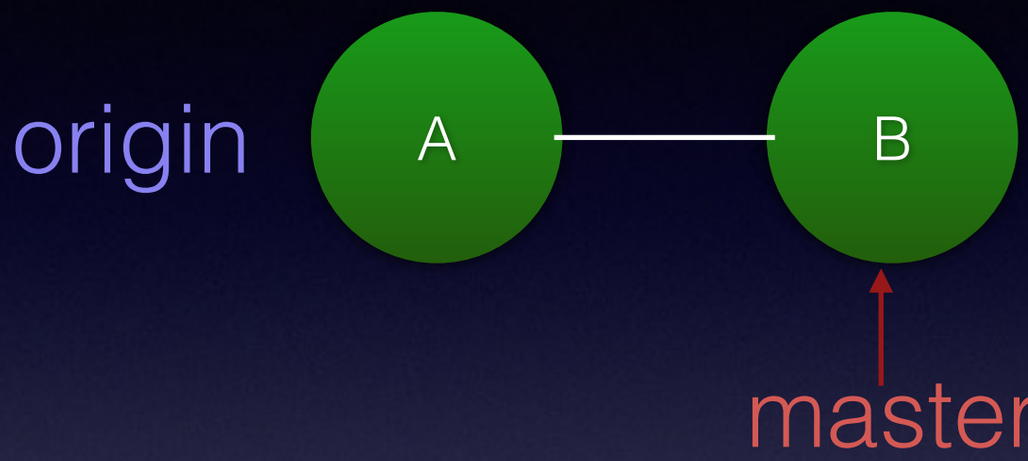


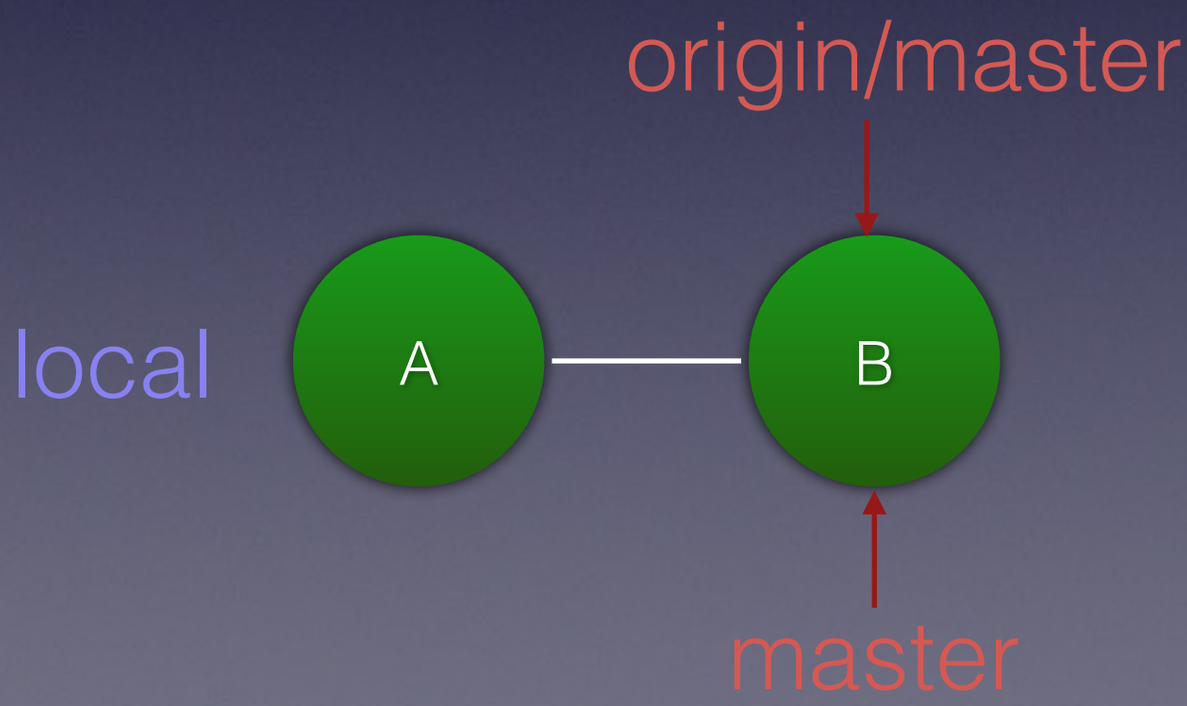
Remote repositories



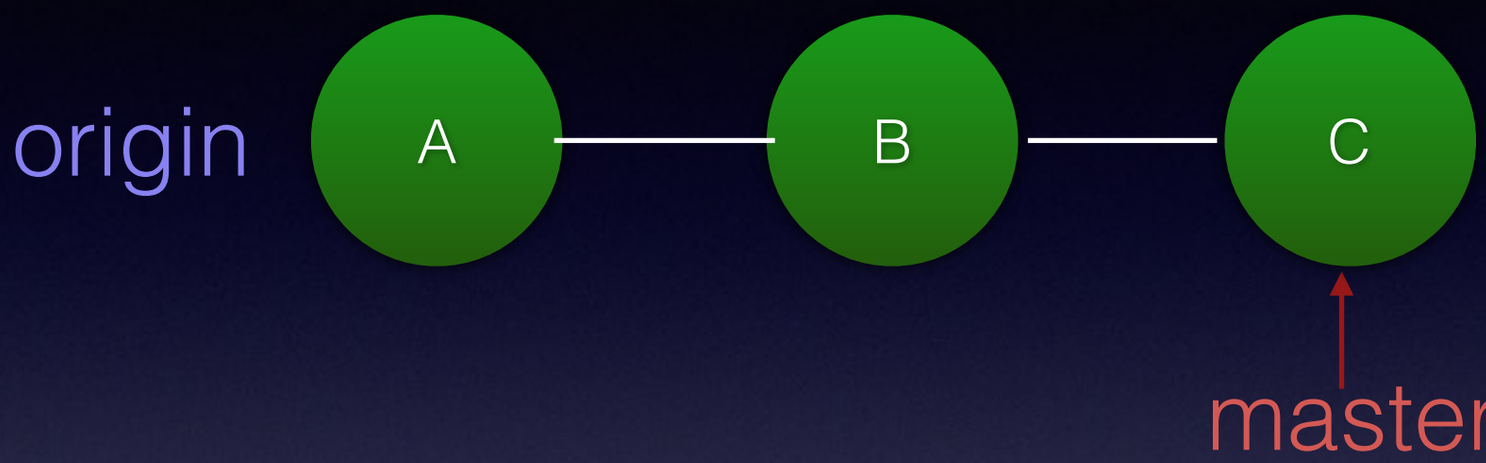


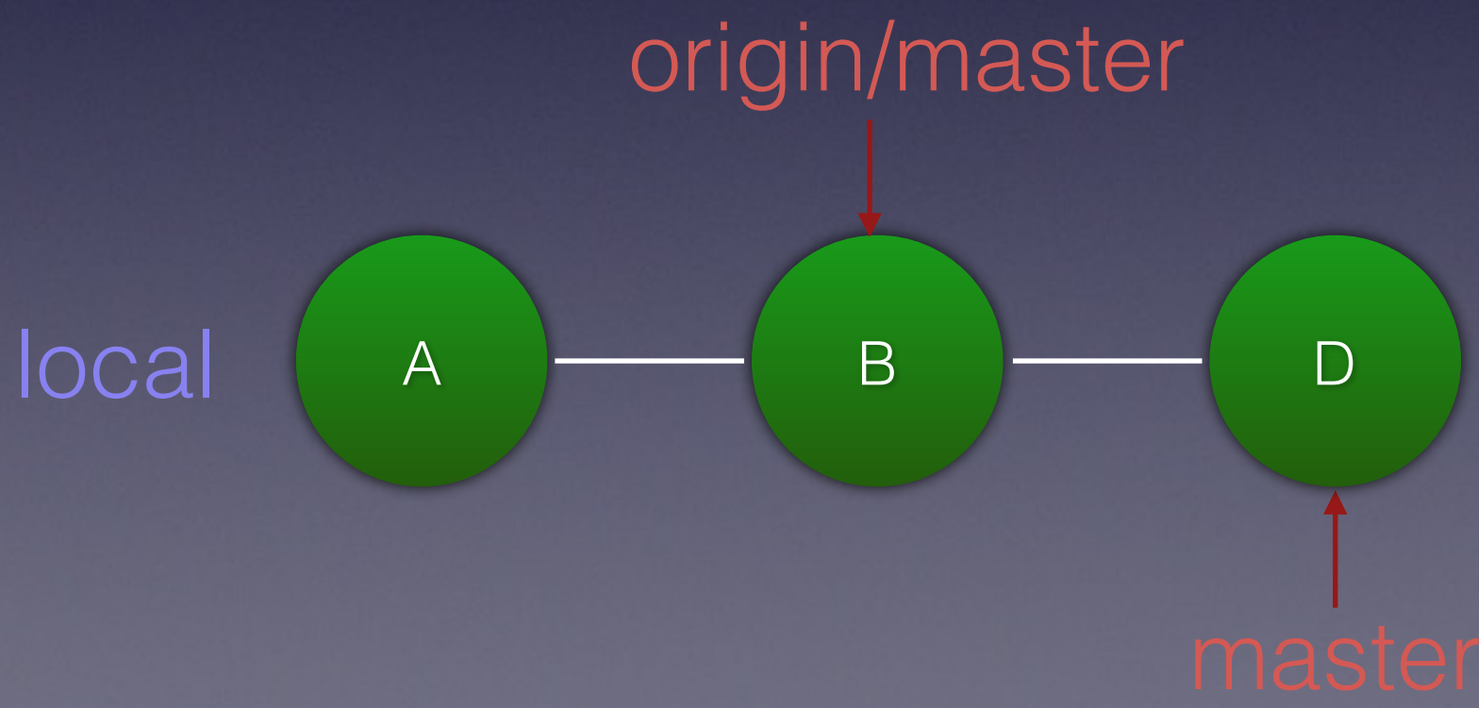
Remote repositories



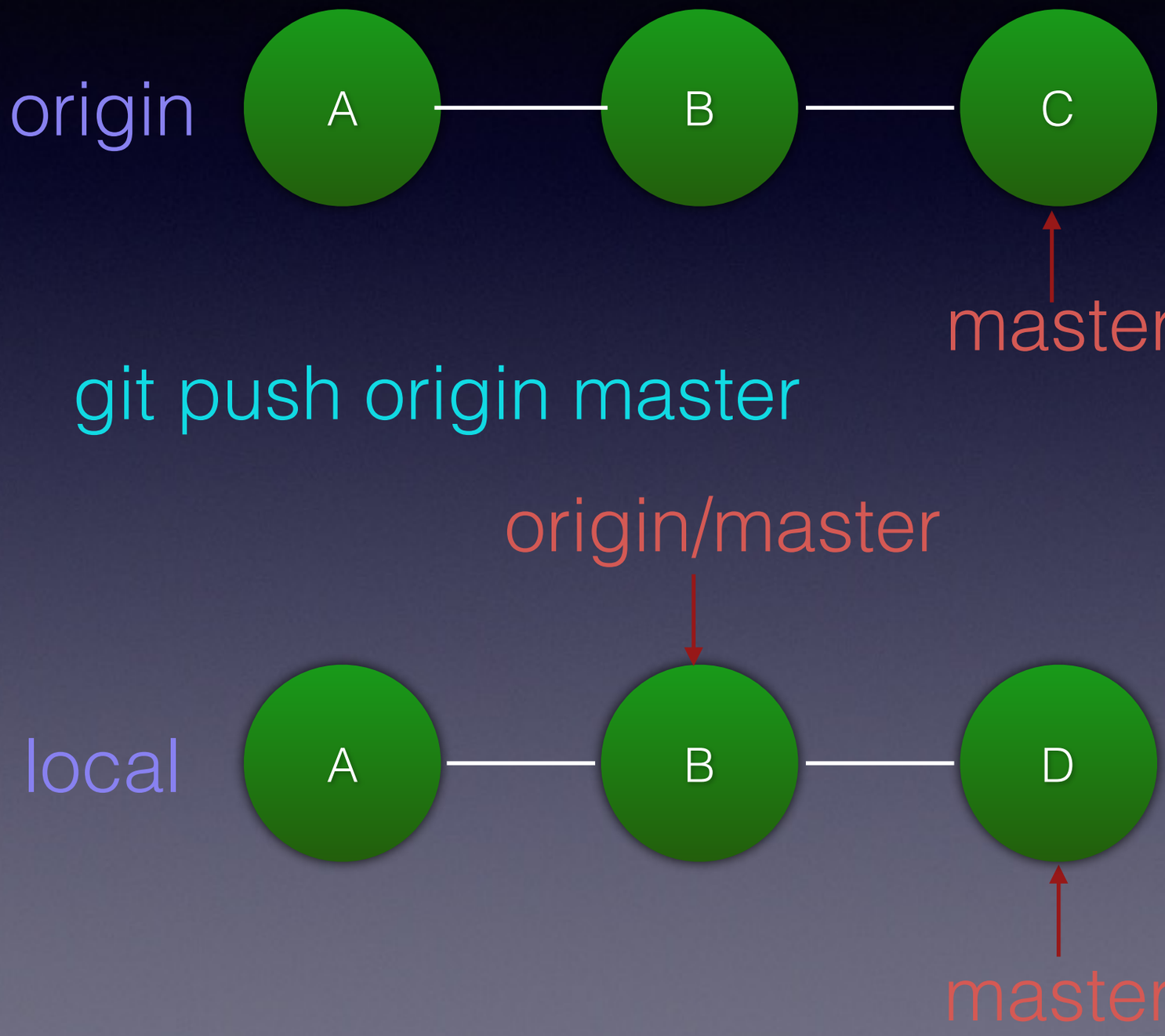


Remote repositories





Remote repositories



git push origin master

Remote repositories

```
git push origin master

To git@bitbucket.org:demo/demo.git
 ! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to 'git@bitbucket.org:demo/demo.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

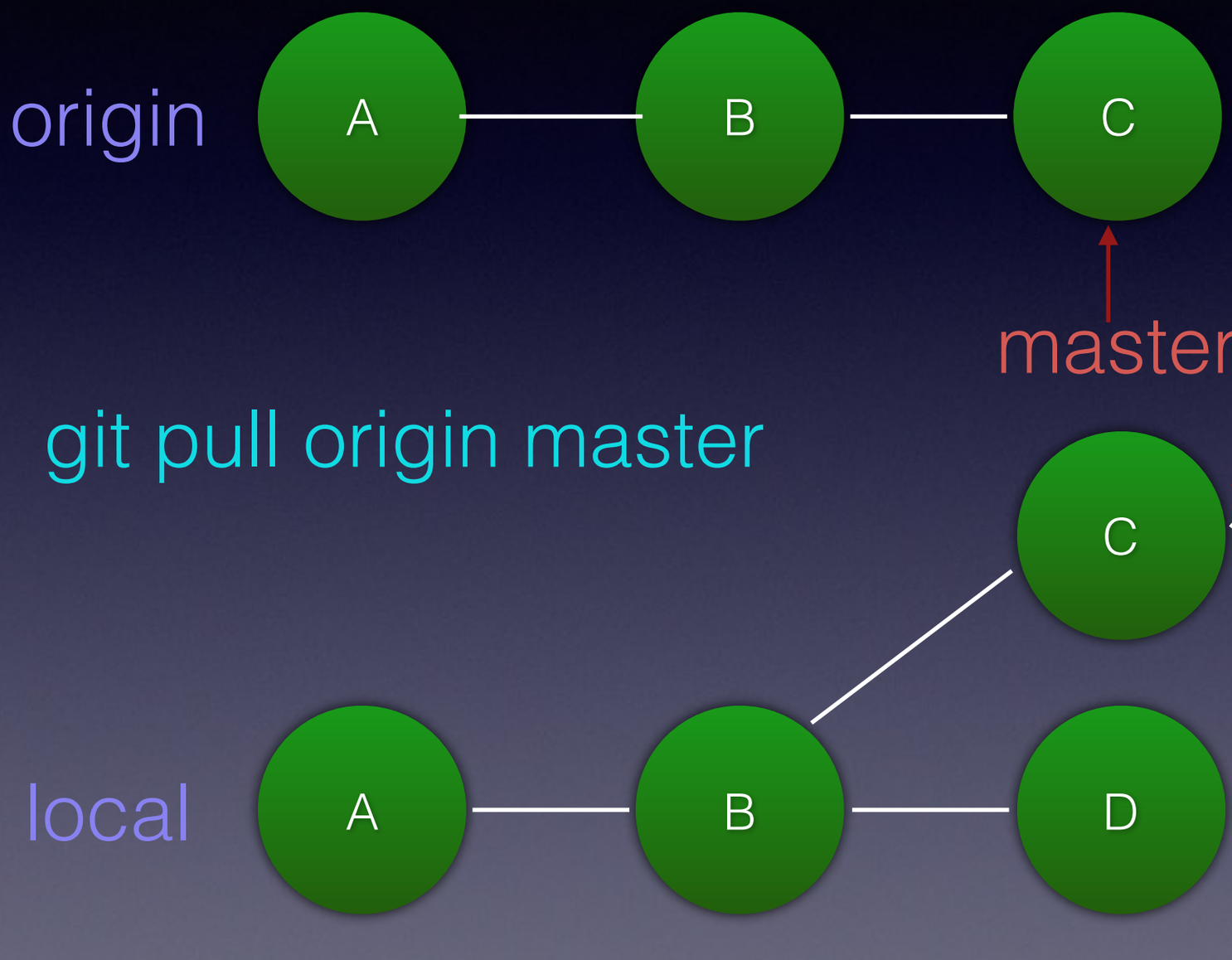
Remote repositories

```
git push origin master

To git@bitbucket.org:demo/demo.git
 ! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to 'git@bitbucket.org:demo/demo.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

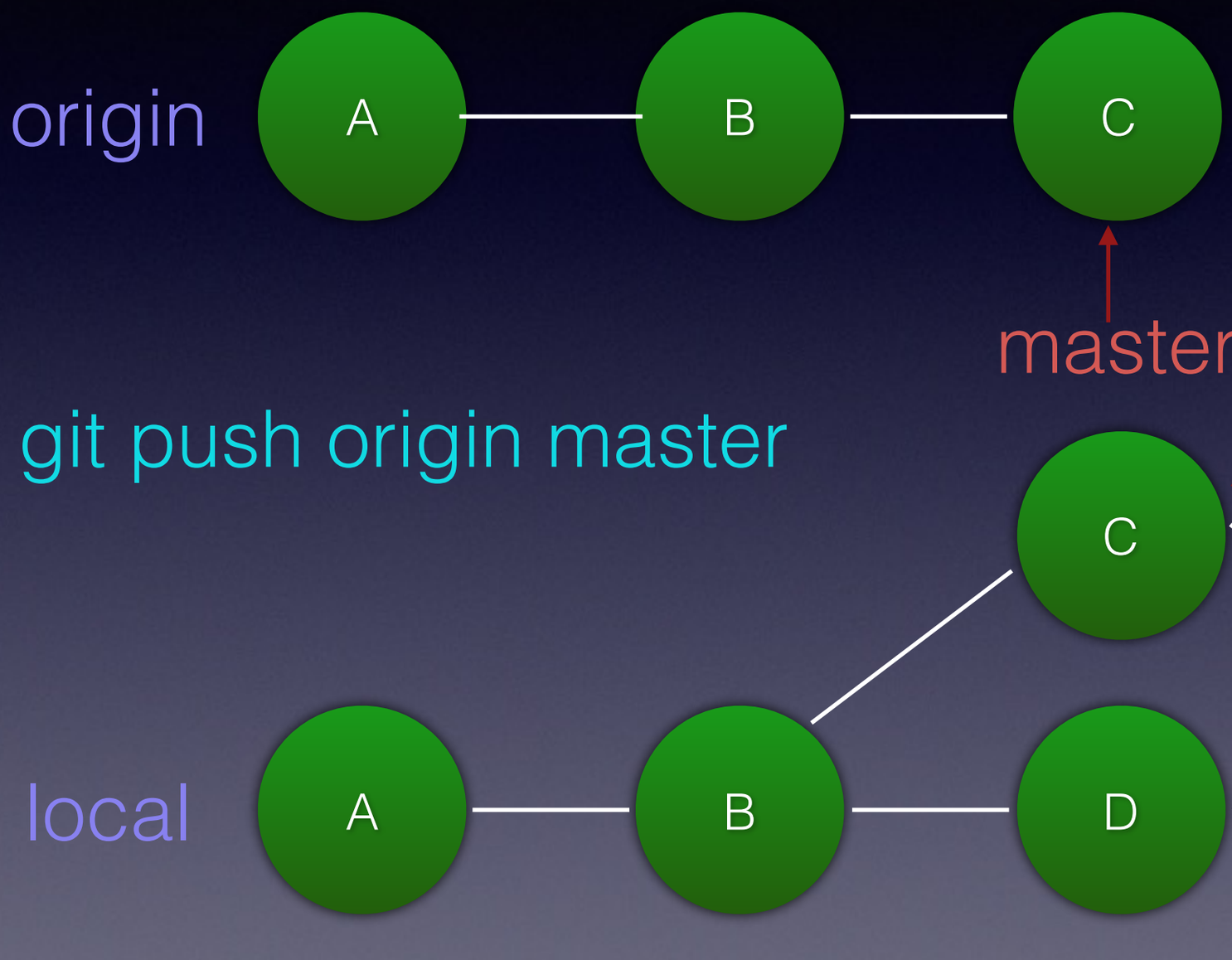
Remote repositories

```
git push origin master

To git@bitbucket.org:demo/demo.git
 ! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to 'git@bitbucket.org:demo/demo.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

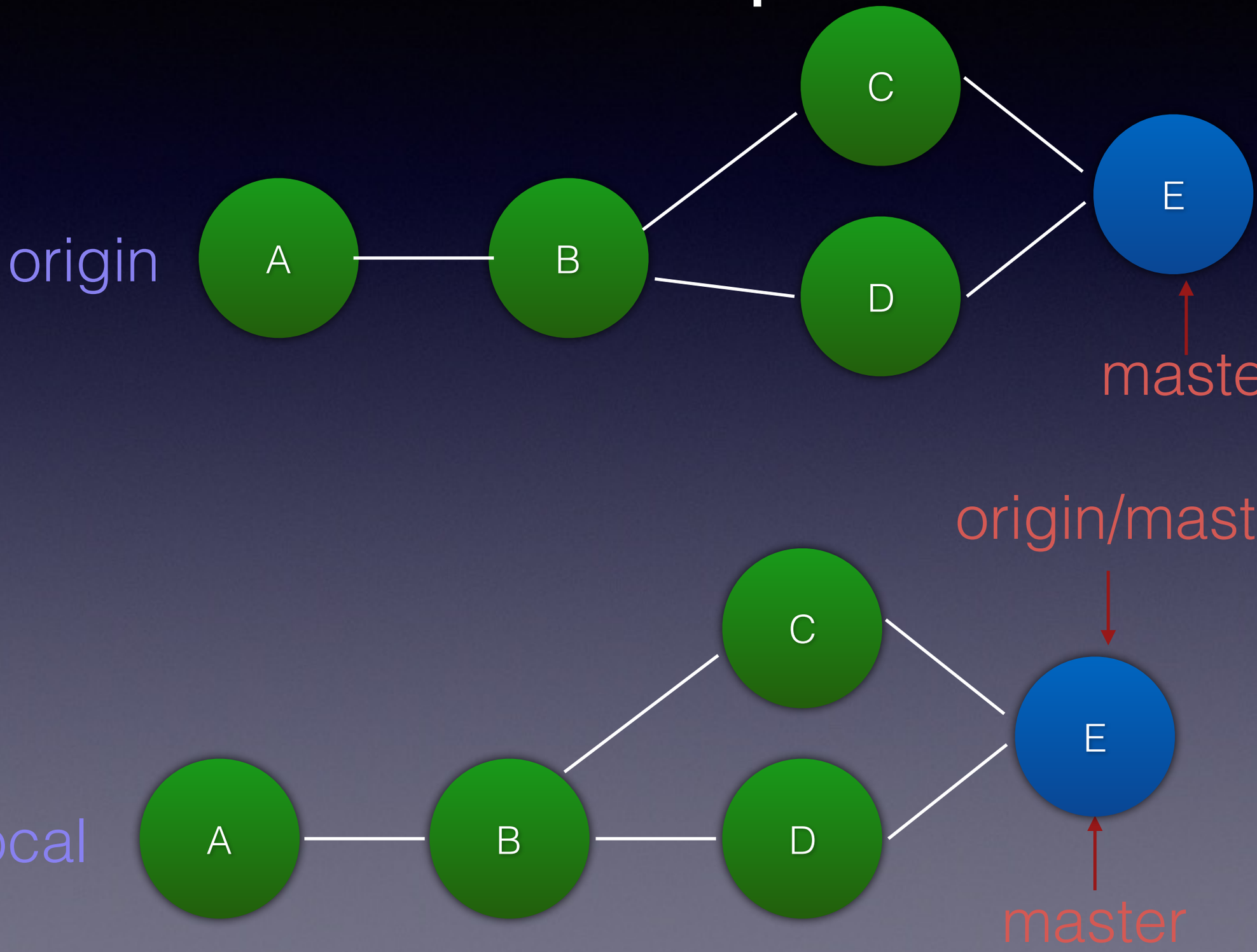
Remote repositories



Remote repositories

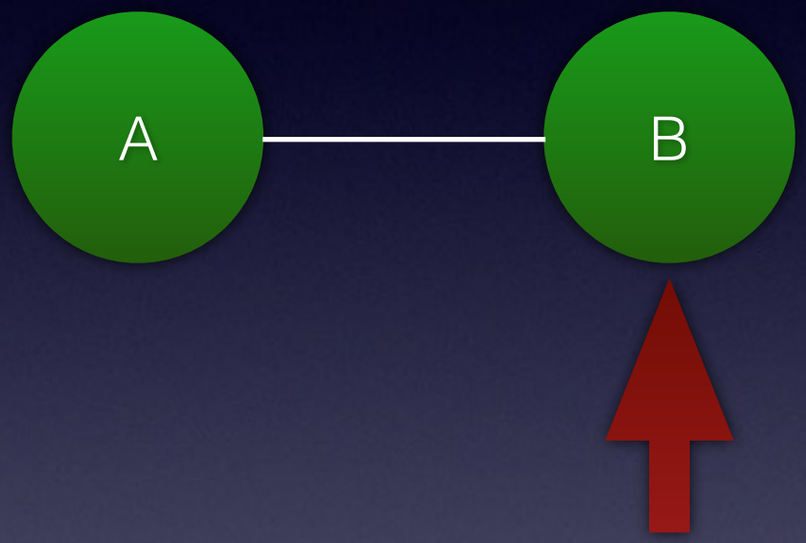


Remote repositories



Detached heads

What is a branch?



cat .git/refs/heads/master

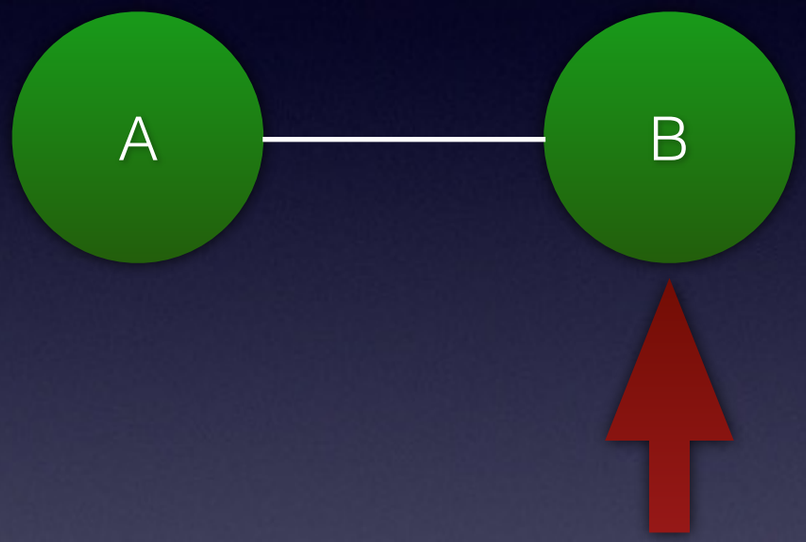
6e88dad5d769b921d1a700bee8d57a7a82d67f29

What is the current branch?

cat .git/HEAD

```
ref: refs/heads/master
```

What is a branch?



cat .git/**refs/heads/master**

```
6e88dad5d769b921d1a700bee8d57a7a82d67f29
```

What is the current branch?

git checkout fix1

cat .git/HEAD

```
ref: refs/heads/fix1
```

git checkout

git checkout <branch name>

git checkout <hash>

Detached head

git checkout f9f4b3d

```
Note: checking out 'f9f4b3d'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again.
Example:

  git checkout -b <new-branch-name>

HEAD is now at f9f4b3d... initial check in
```

Detached head

git checkout f9f4b3d

```
Note: checking out 'f9f4b3d'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again.
Example:

  git checkout -b <new-branch-name>

HEAD is now at f9f4b3d... initial check in
```

Detached head

git checkout f9f4b3d

```
Note: checking out 'f9f4b3d'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again.
Example:

  git checkout -b <new-branch-name>

HEAD is now at f9f4b3d... initial check in
```

Detached head

git status

```
HEAD detached at f9f4b3d
nothing to commit, working directory clean
```

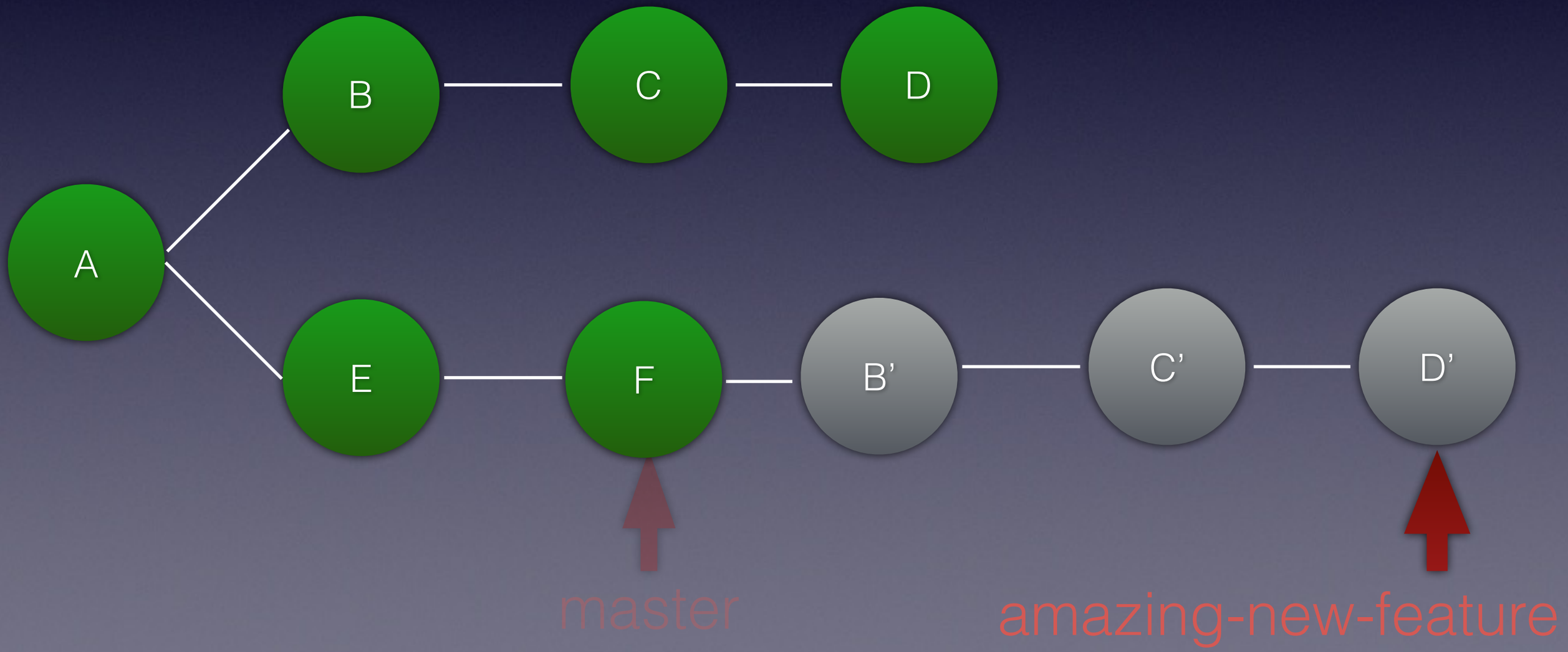
Detached head

cat .git/HEAD

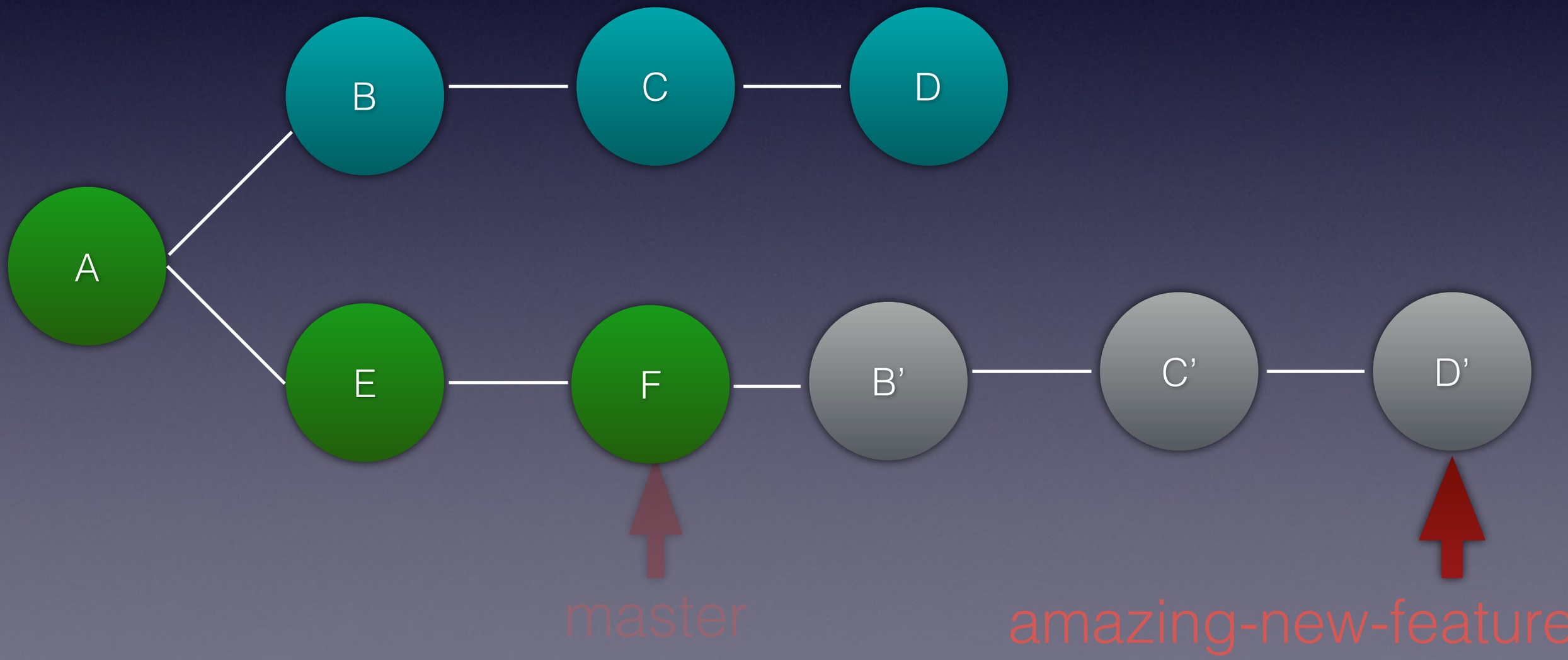
```
f9f4b3d7f21ca6e6aab3c1d9cfd8d0bf05bc78a7
```

What happens to old commits?

Old commits

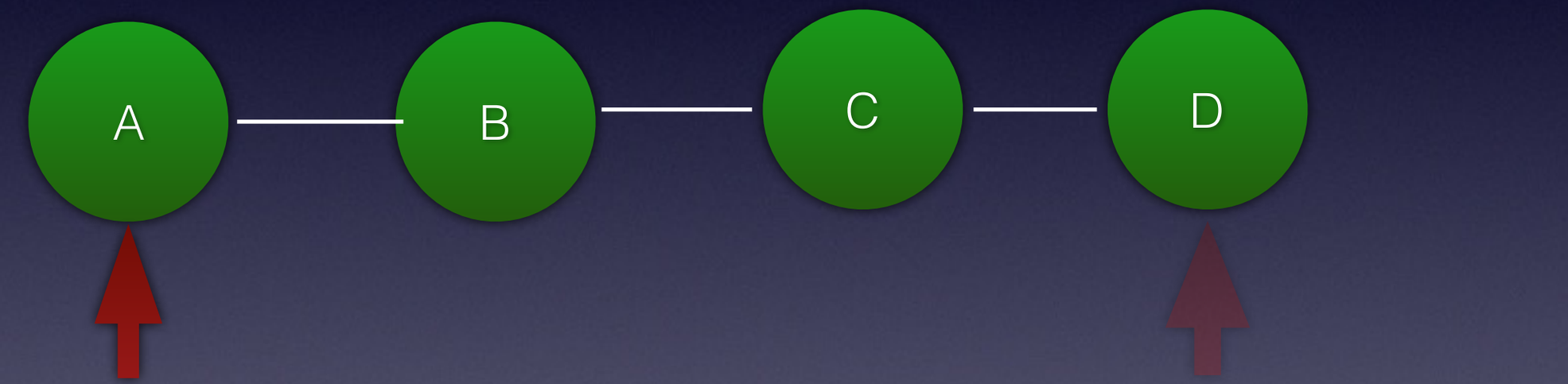


Old commits

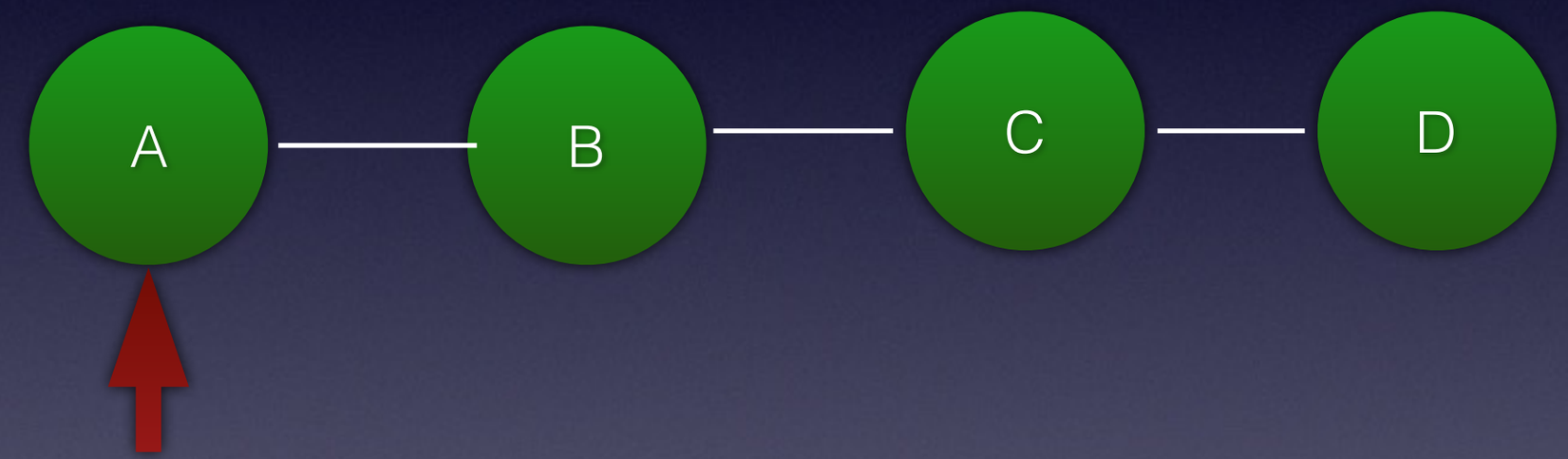


… story continues…

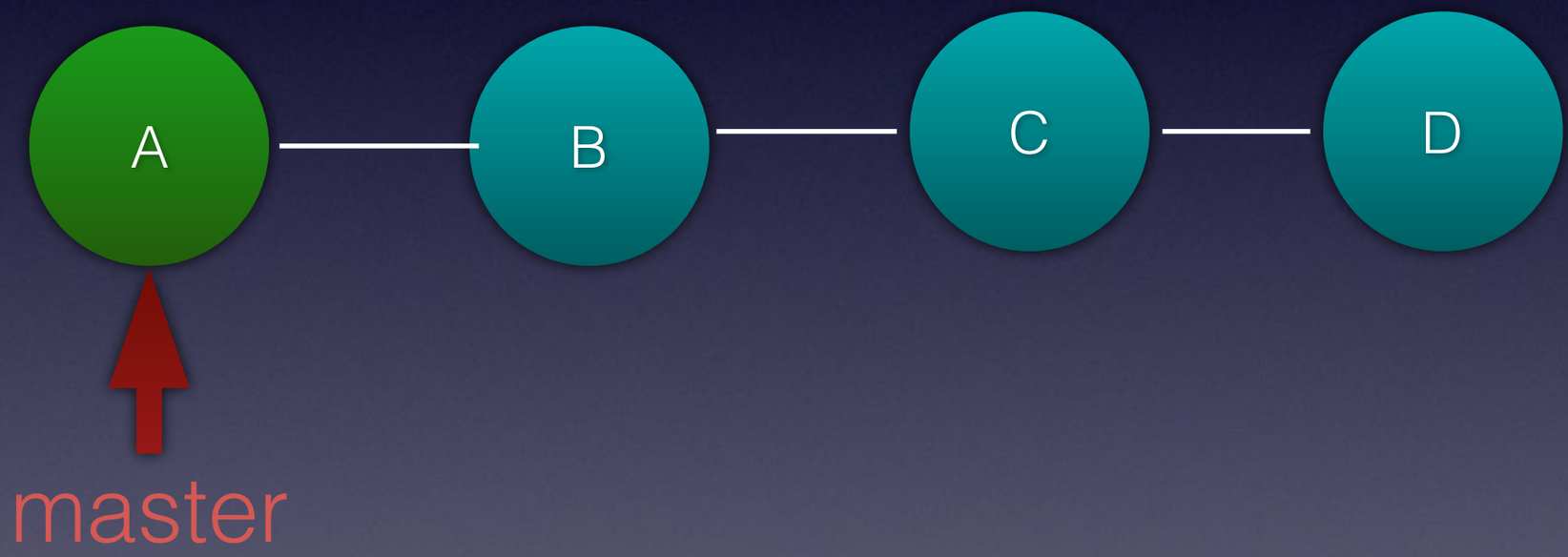
What happens if I delete a branch?



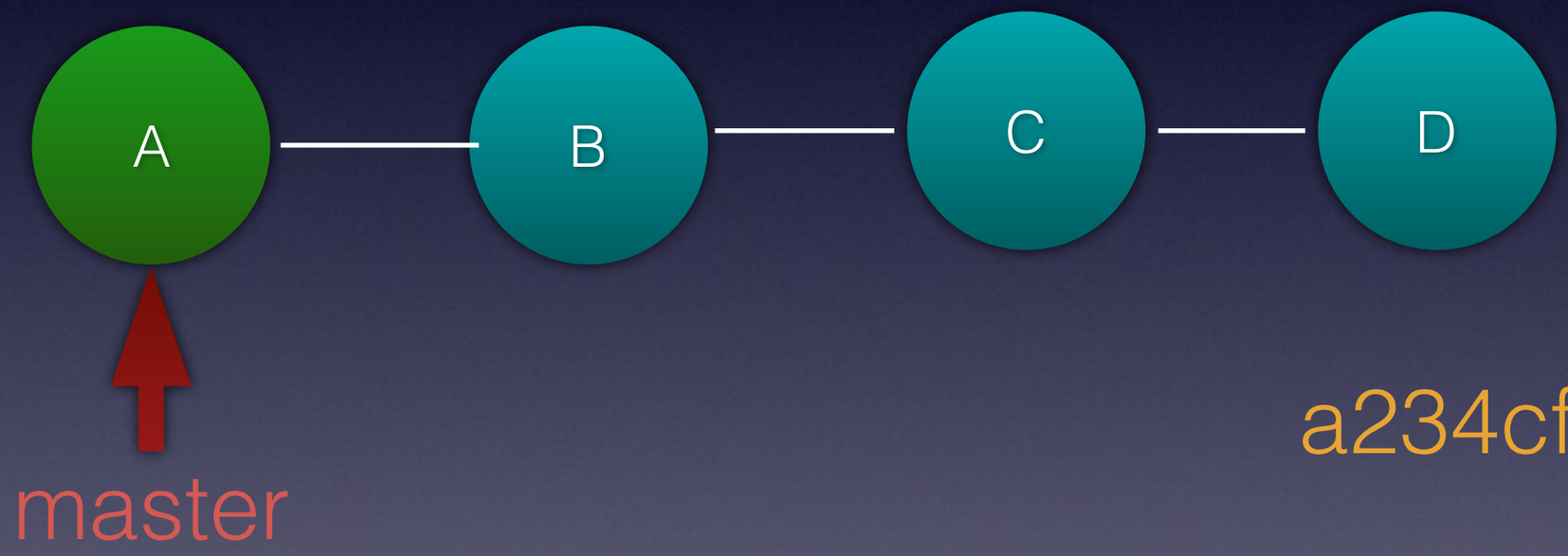
What happens if I delete a branch?



What happens if I delete a branch?

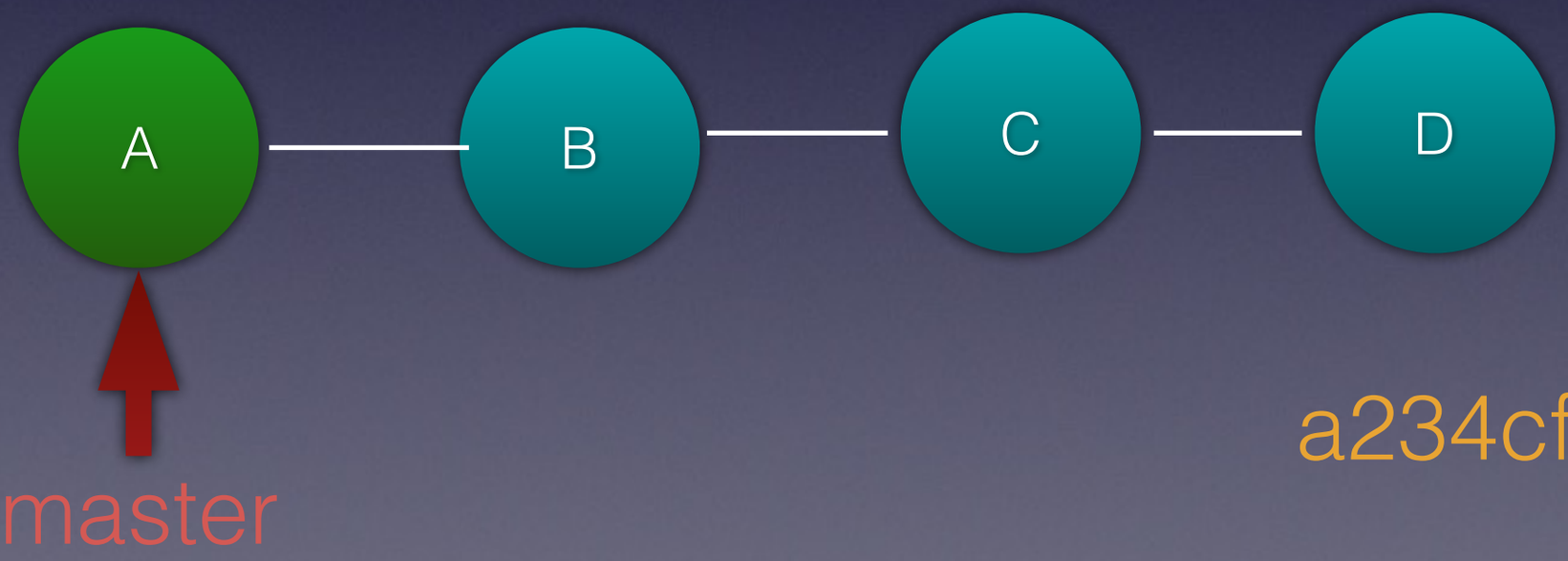


What happens if I delete a branch?

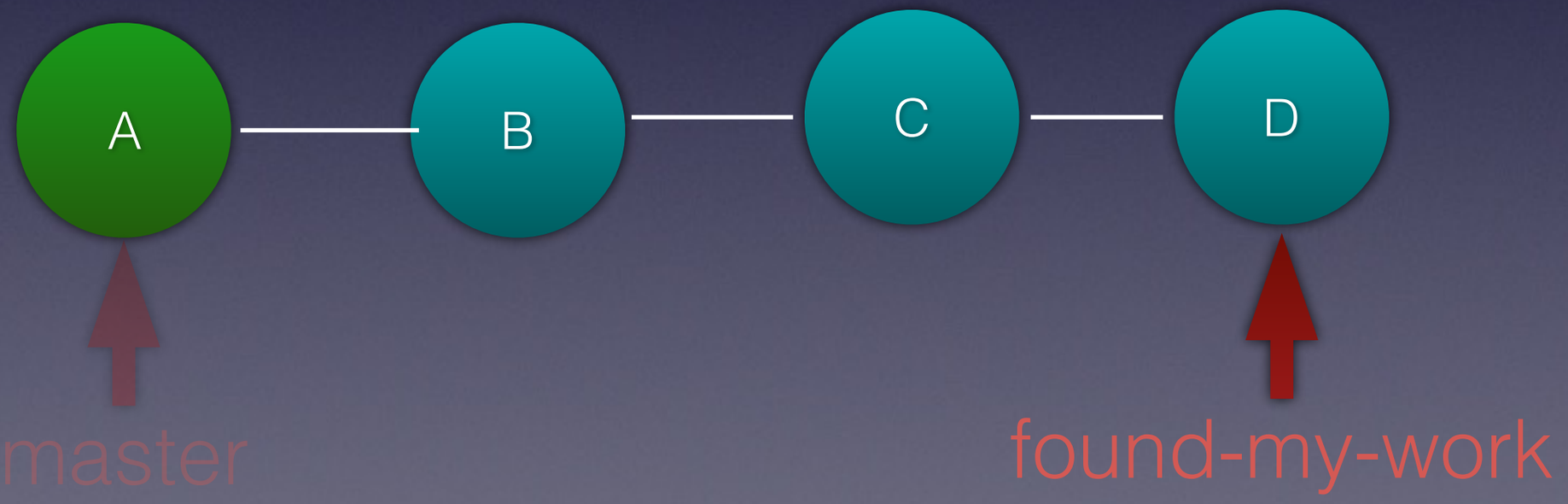


I can get it back...

git checkout -b found-my-work a234cf



I can get it back...



But I don’t know the hash of the deleted branch….

… to be continued…

Reflog

Reflog

```
git reflog

f9f4b3d HEAD@{0}: checkout: moving from fix1 to f9f4b3d
f239725 HEAD@{1}: checkout: moving from master to fix1
f9f4b3d HEAD@{2}: checkout: moving from fix2 to master
1c12f98 HEAD@{3}: rebase -i (finish): returning to refs/heads/fix2
1c12f98 HEAD@{4}: rebase -i (pick): ADD sun
a40867d HEAD@{5}: rebase -i (pick): ADD sat
3cd7f7e HEAD@{6}: commit: ADD boo
f1c4ebc HEAD@{7}: cherry-pick: fast-forward
…
f9f4b3d HEAD@{30}: commit (initial): initial check in
```

Reflog found the branch

Summary

Commit

Staged for commit

Working tree

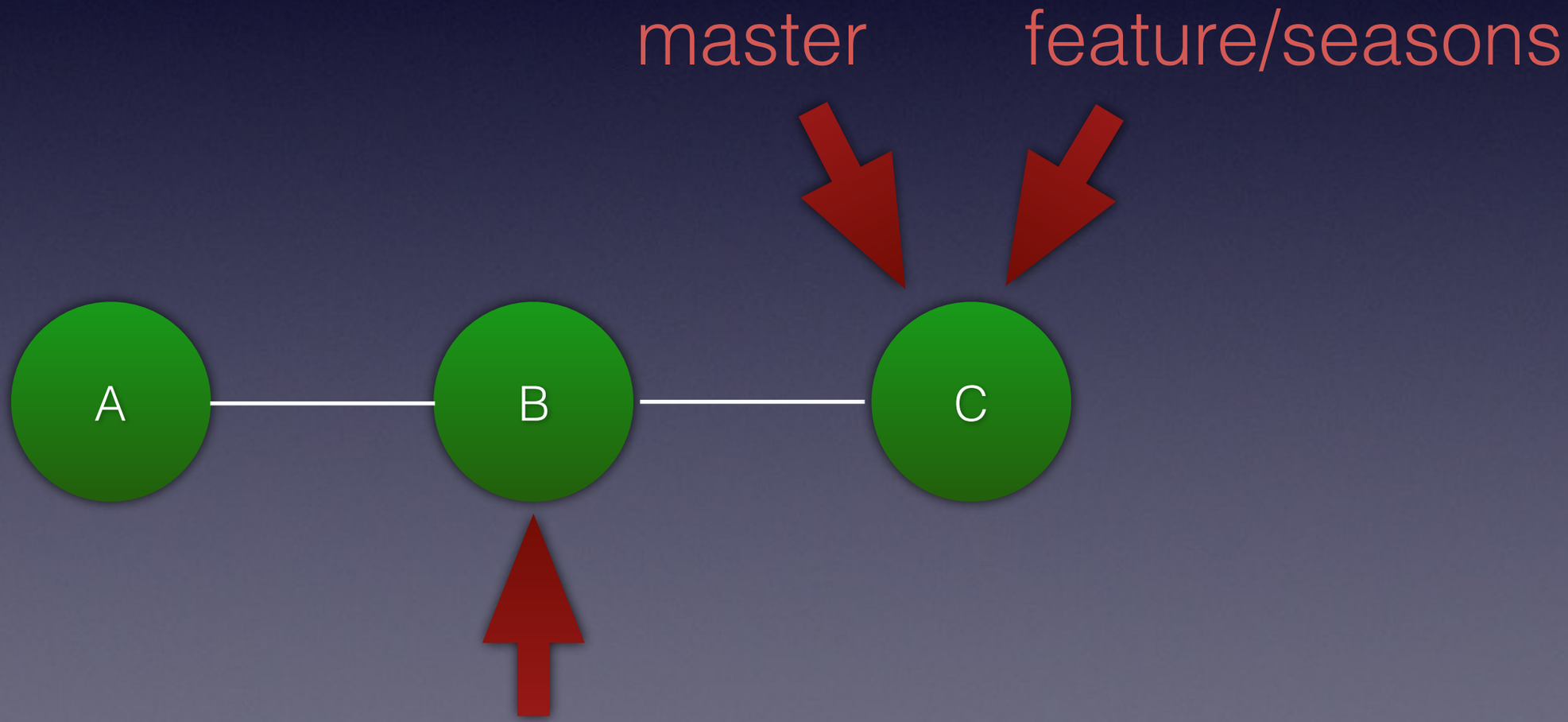
Untracked

What is a commit?

1. Metadata
2. Patch
3. Parent commits(s)

SHA: 6e88dad5d769b921d1a700bee8d57a7a82d67f29

What is a branch?



Actions

- Merge
- Rebase
- Cherry pick

How did the story end?

Questions