# Git and GitHub in the Enterprise

The Container Store®

The Container Store®

+

The Container Store®

=

# Git Overview

# Why Git?

# Why Git?

- Distributed

# Why Git?

- Distributed
- Fast

# Why Git?

- Distributed
- Fast
- Small

# Why Git?

- Distributed
- Fast
- Small
- Cheap branching

# Why Git?

- Distributed
- Fast
- Small
- Cheap branching
- Flexible

# Why Git?

- Distributed
- Fast
- Small
- Cheap branching
- Flexible

In a nutshell

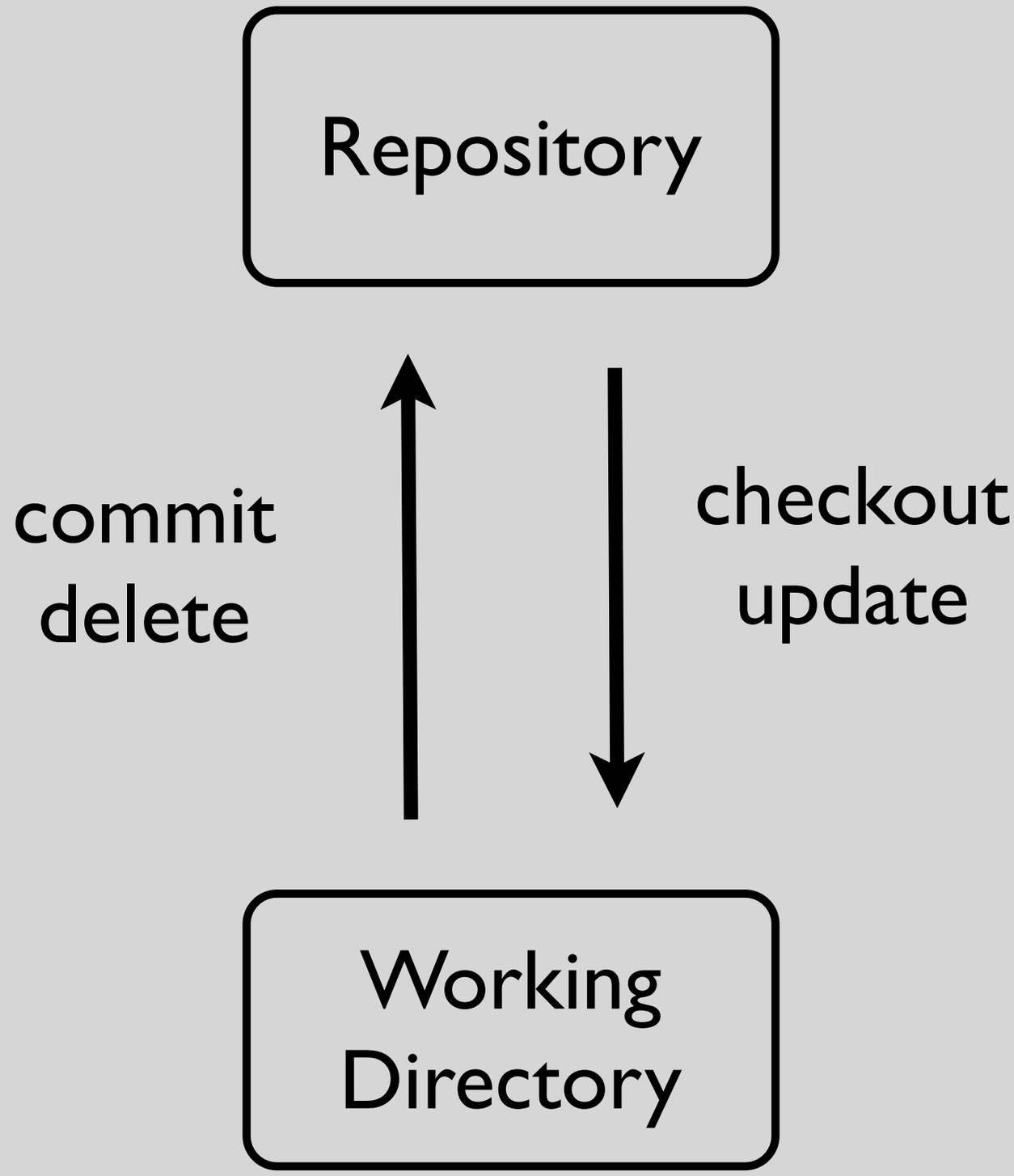
Git is better

Repository

Repository

checkout
update

Working
Directory

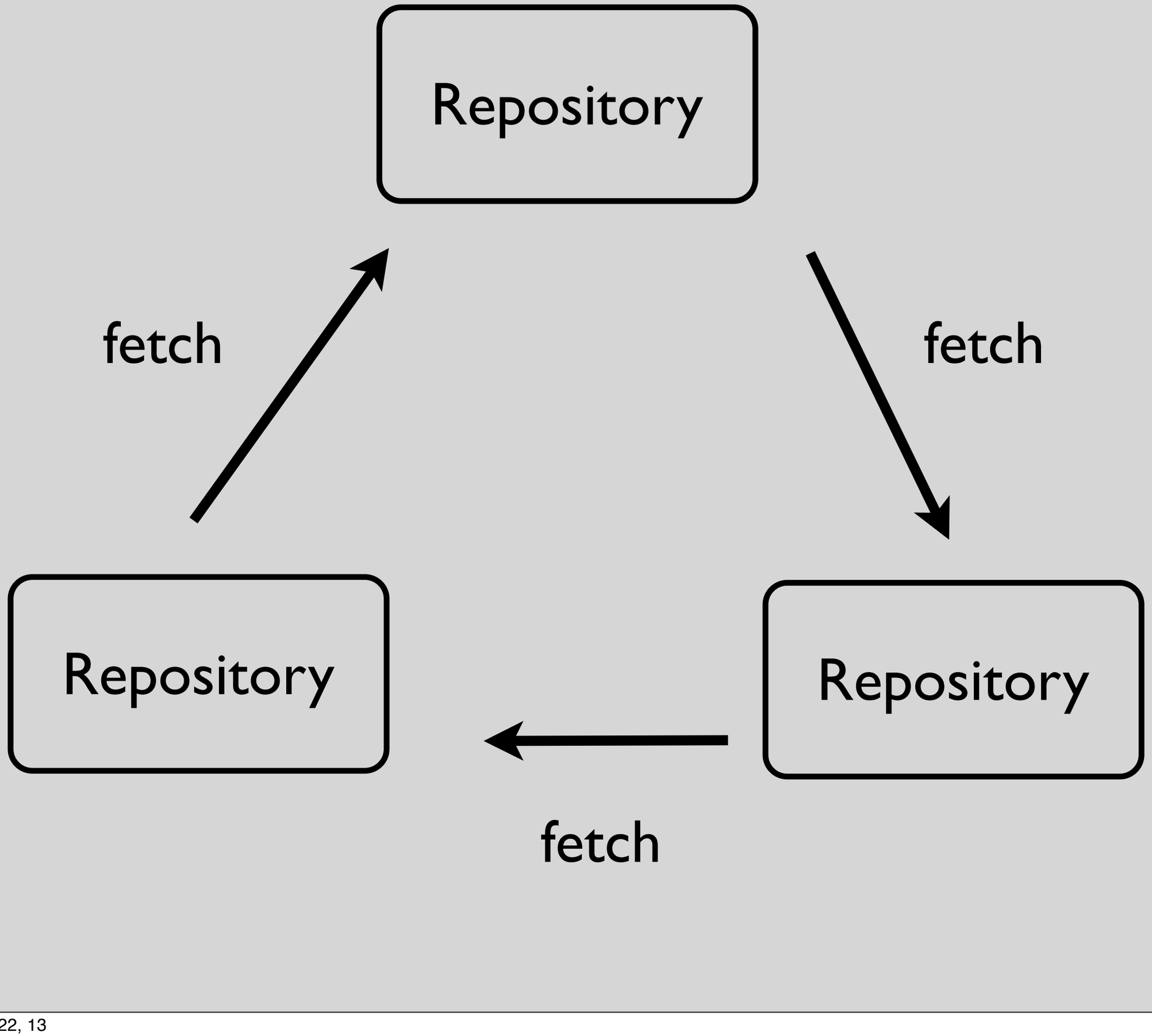

Repository
commit
delete
checkout
update
Working
Directory

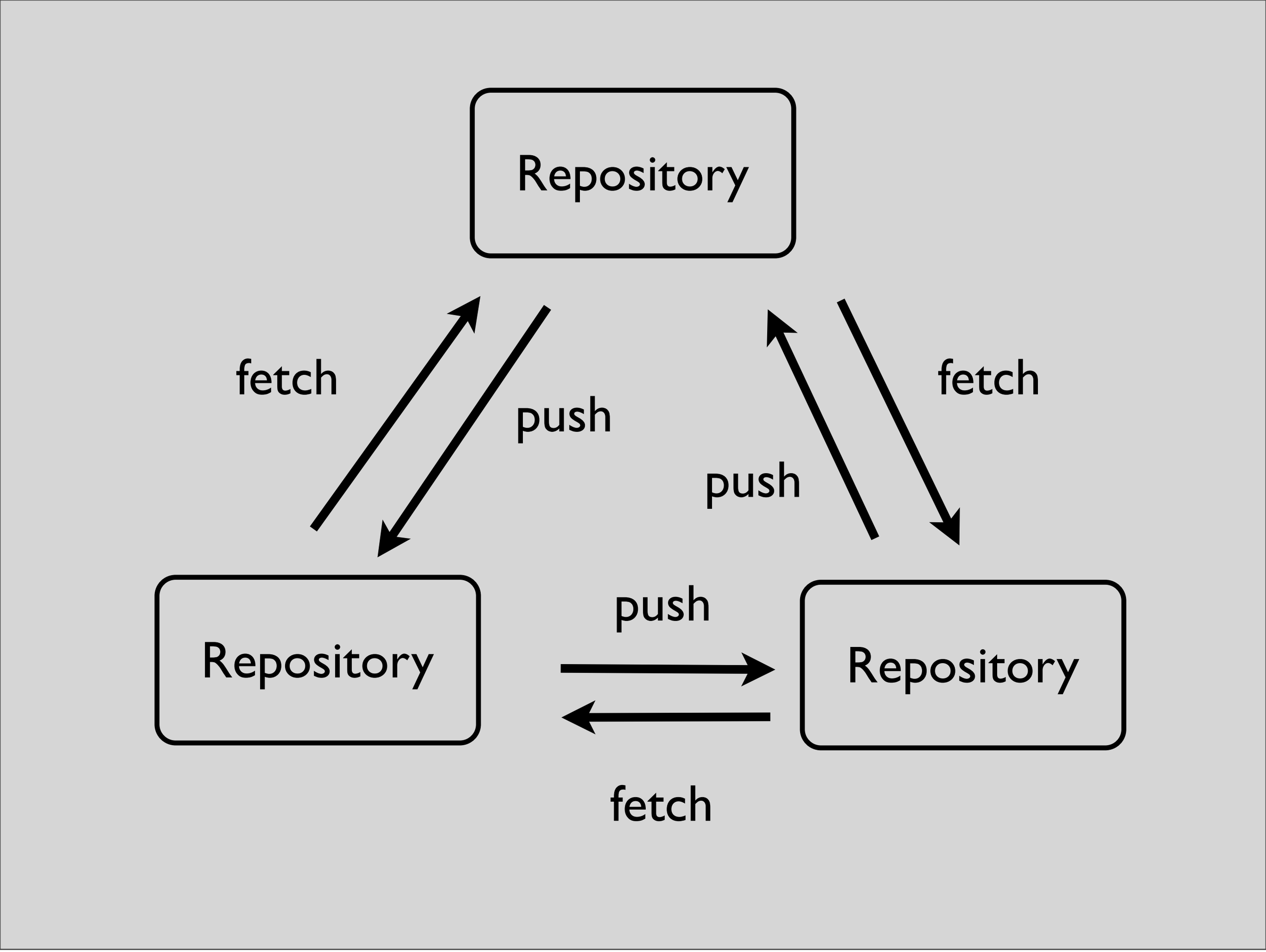
Repository

Repository

Repository

Repository


Repository
fetch
fetch
Repository
Repository
fetch


Repository
fetch
push
fetch
push
Repository
push
Repository
fetch

Repository

Repository

add
branch
commit
diff
log
merge

# Repository

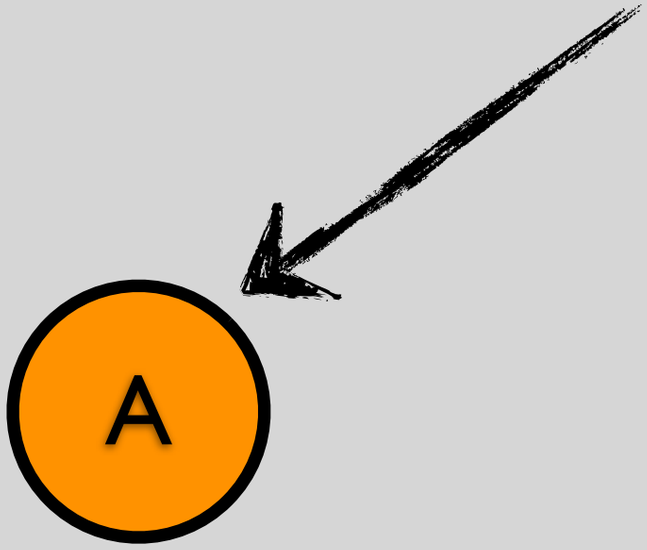
mv
rebase
reset
stash
status
tag

These are all
local commands

Think in terms of
commits, not files

A commit is a snapshot

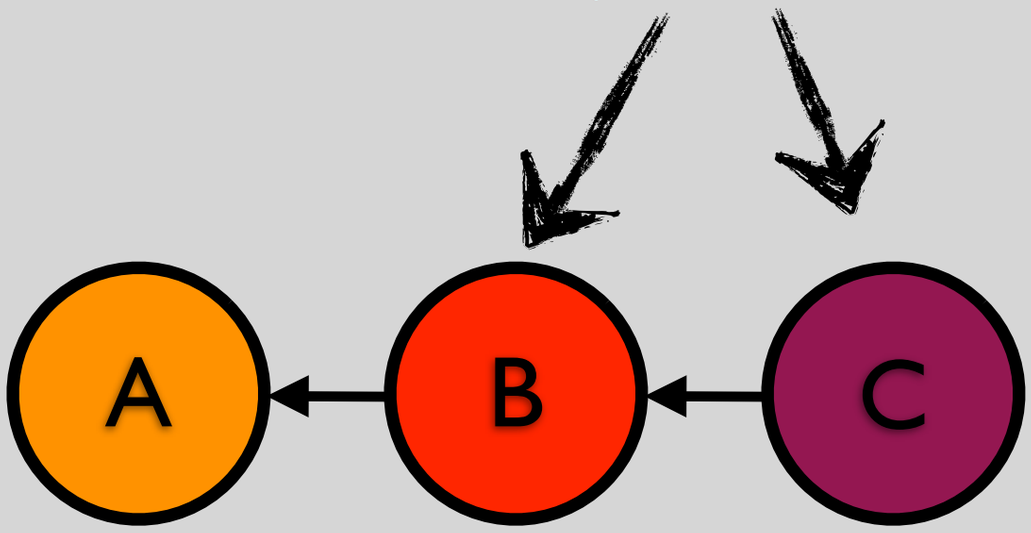
This is a snapshot of
the repository at the time
the commit was made




A

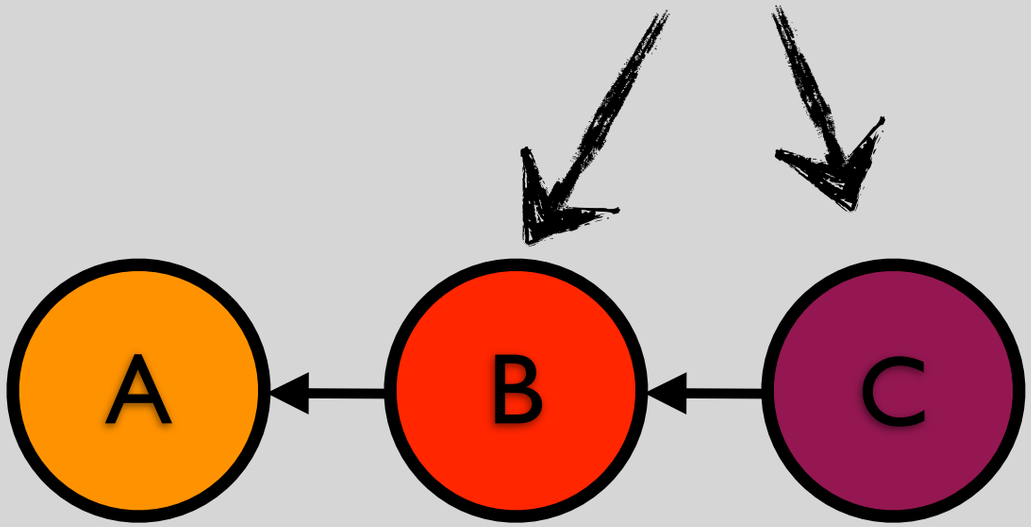
These are more commits.

Each one references the previous commit.



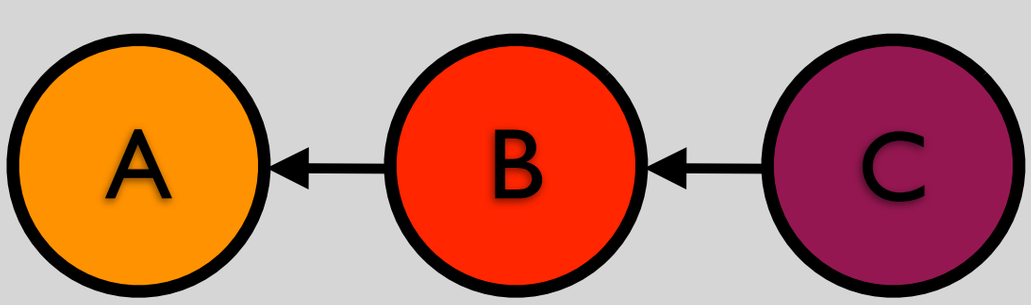
These are more commits.

Each one references the previous commit.

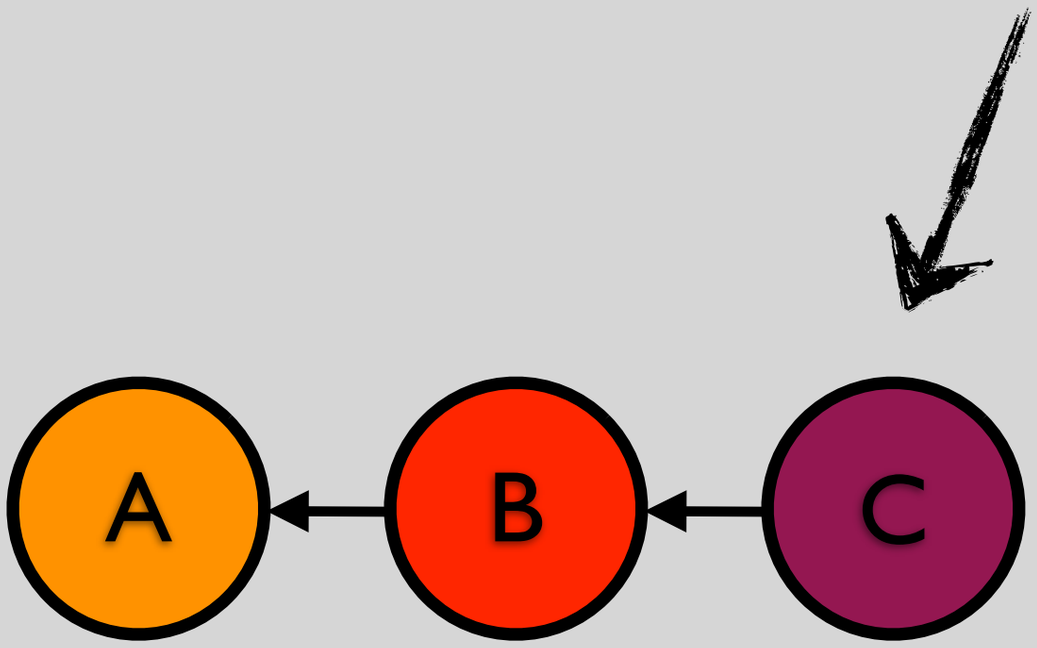



A commit contains:

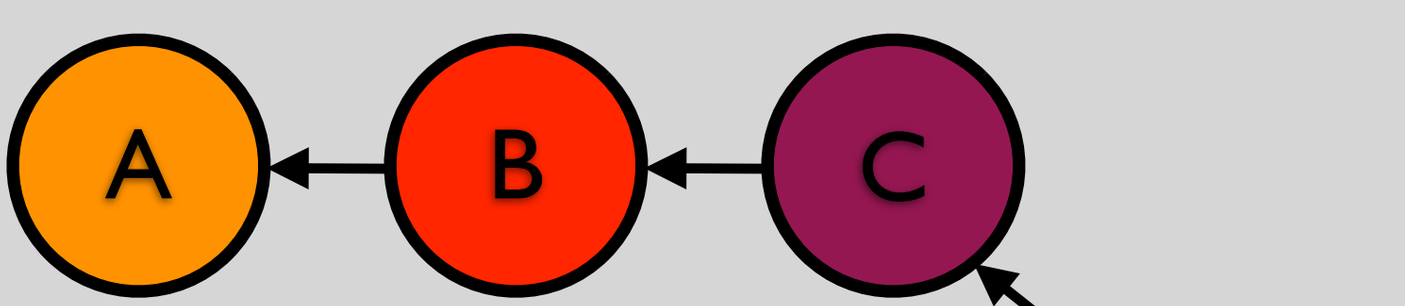
date/time
author
committer
comment


A
B
C

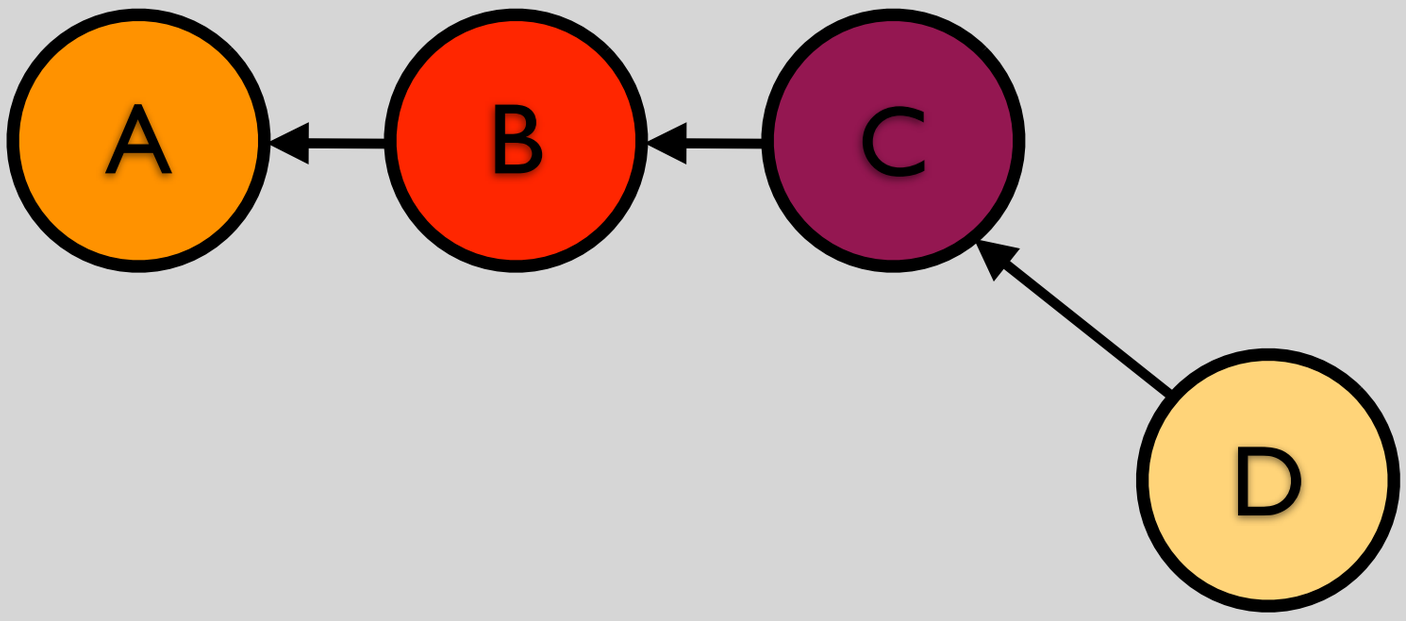
To create a branch, pick which commit to start it from.



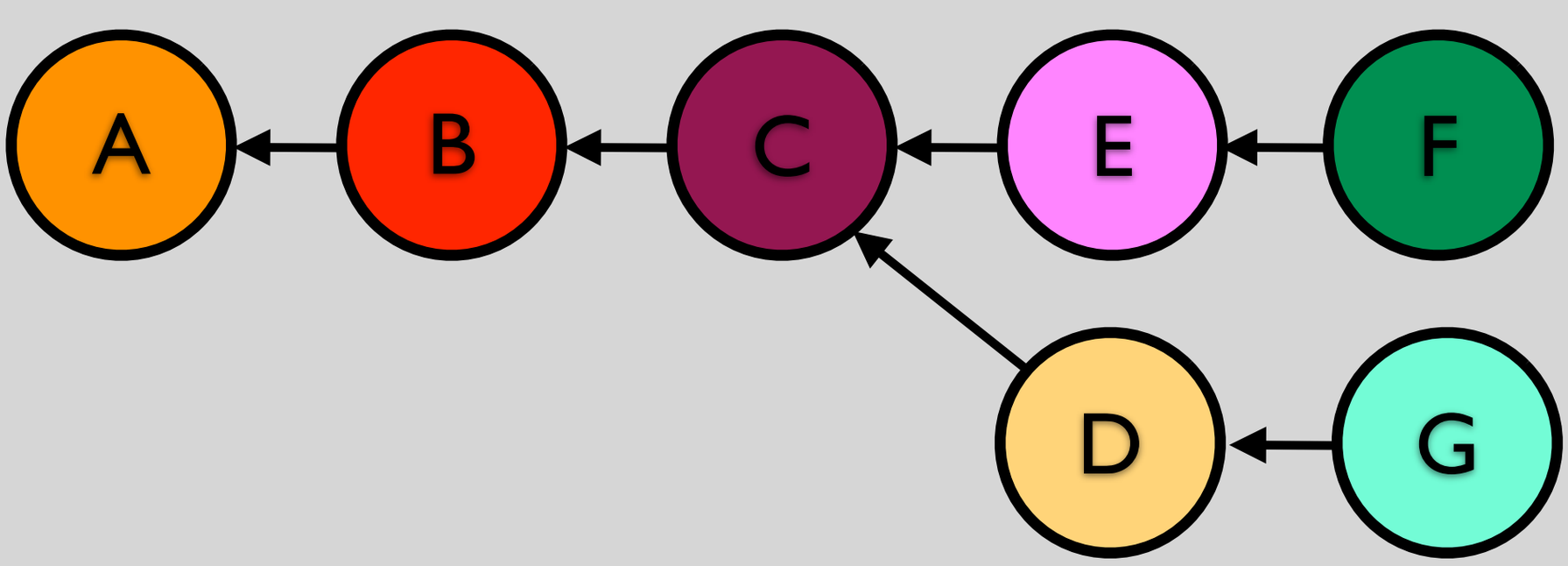
To create a branch, pick which commit to start it from.



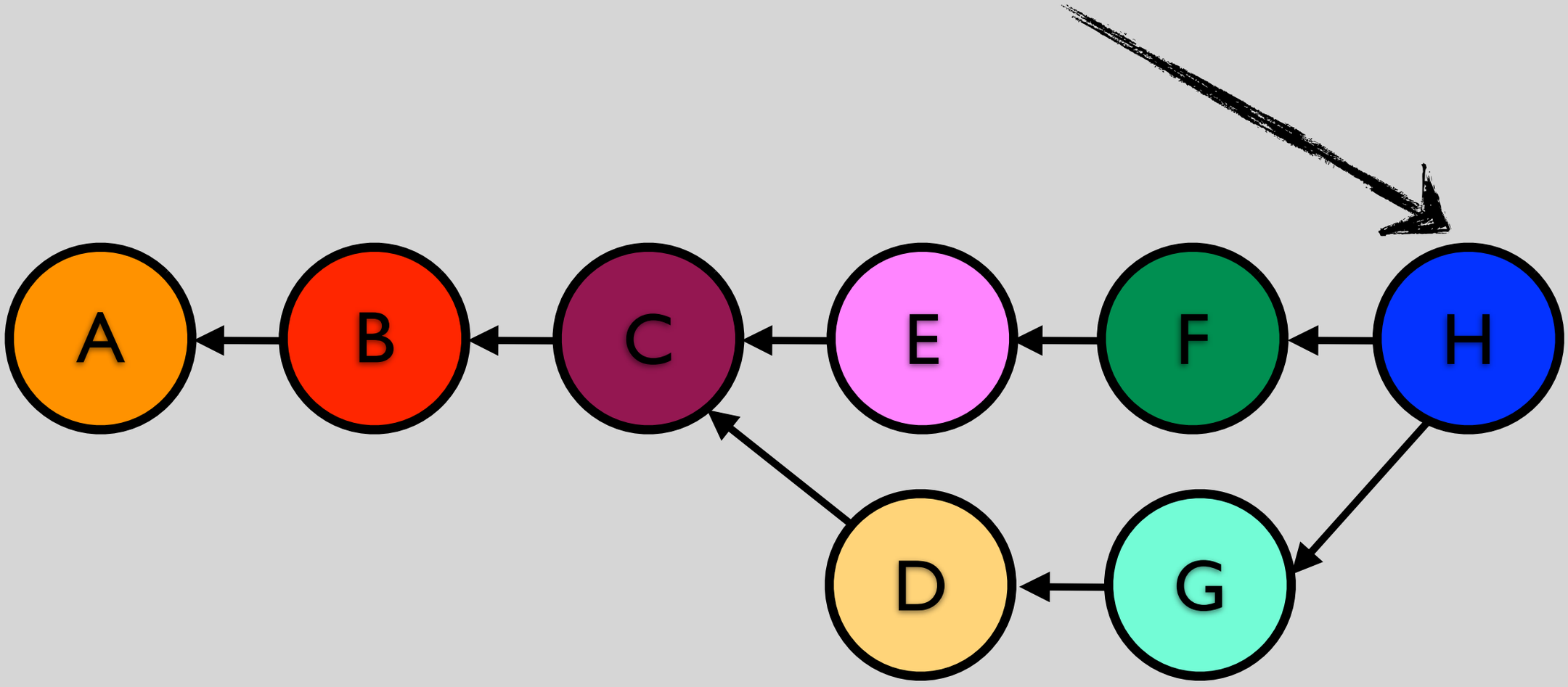

Next commit goes on new branch.


A
B
C
D

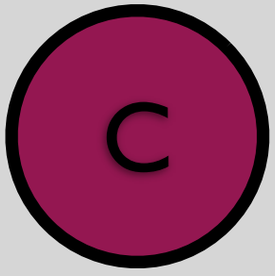
Once we have commits on both branches, we need to merge



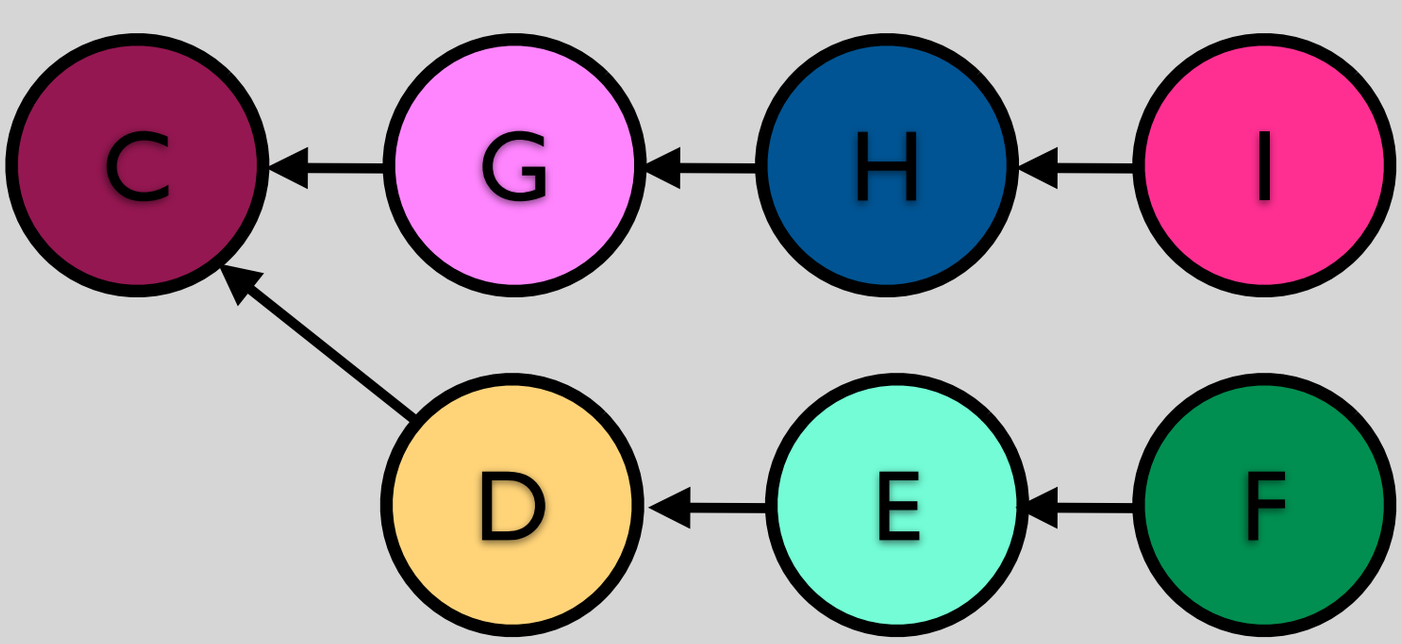
The merge commit points back to both of its parent commits



# What about long lived branches?

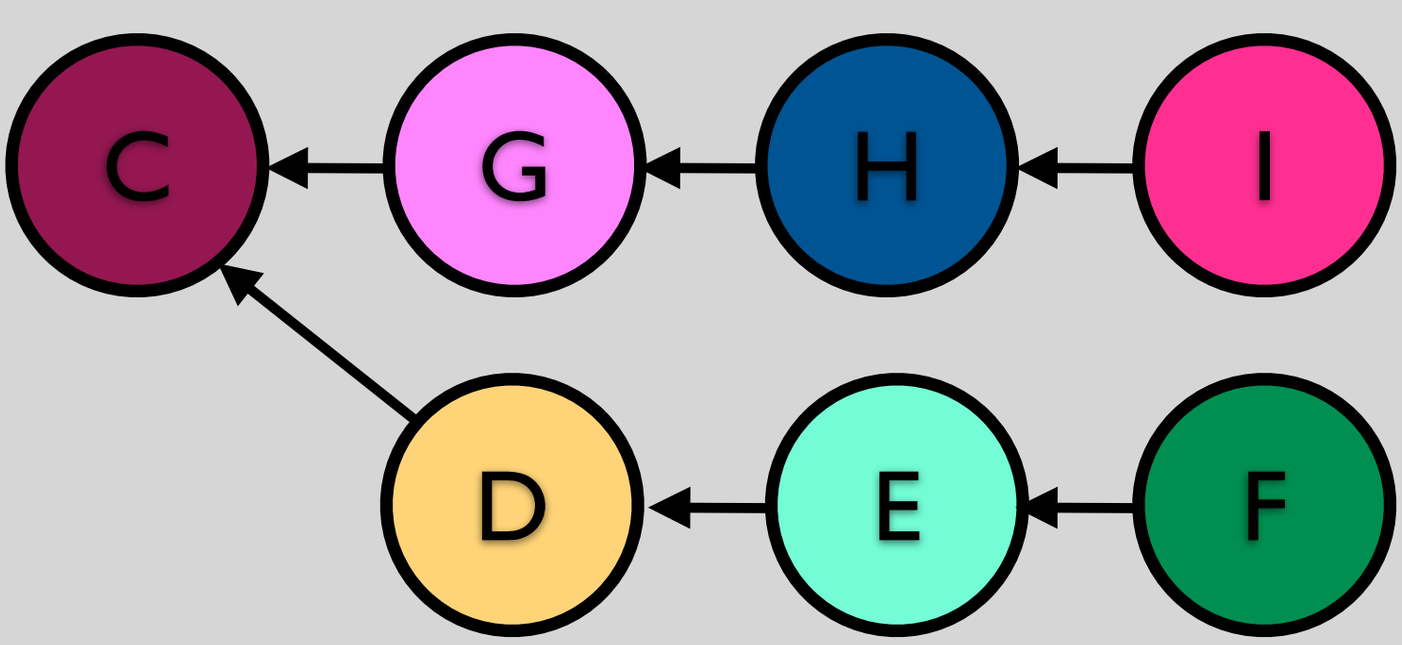


# What about Long Lived branches?

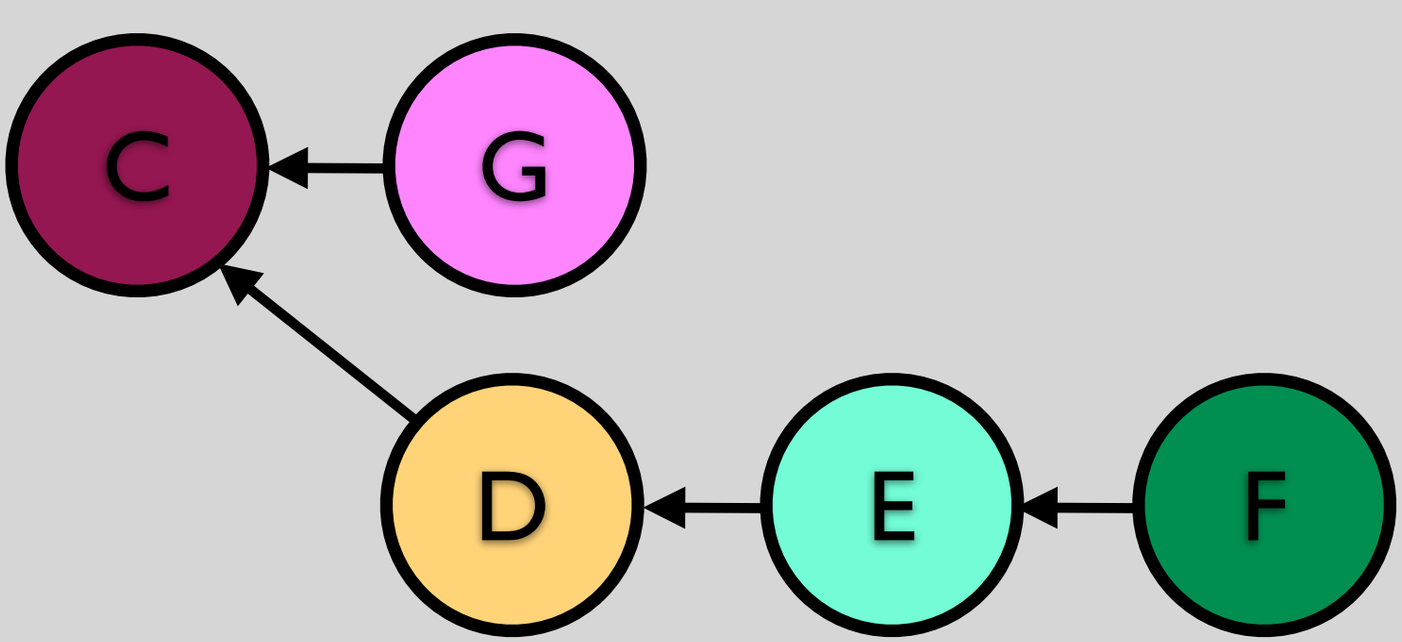


What about long lived branches?

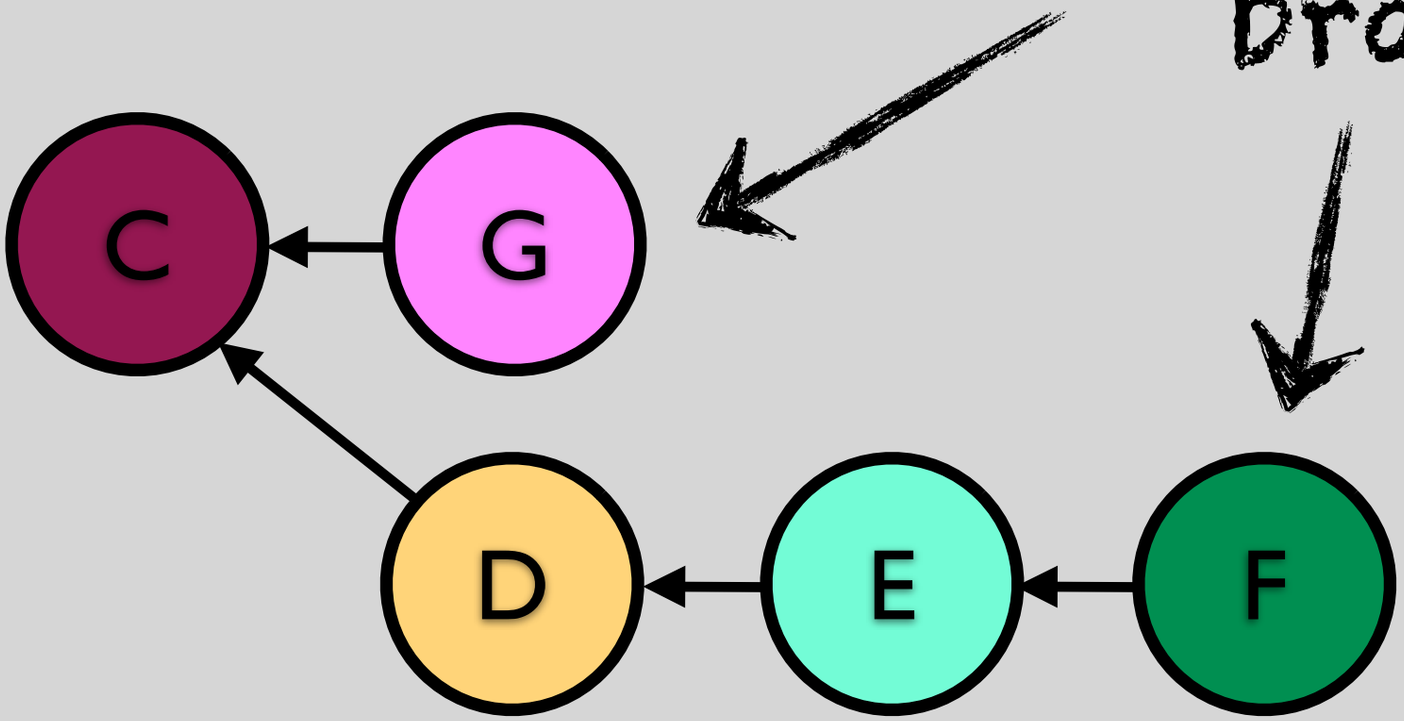
What if you could rewrite history?

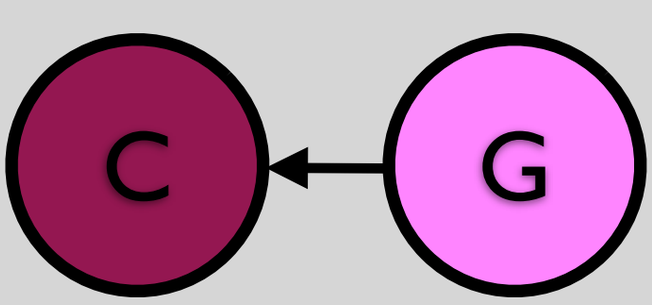


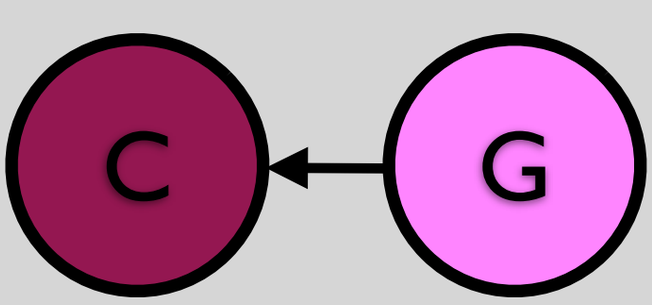
C

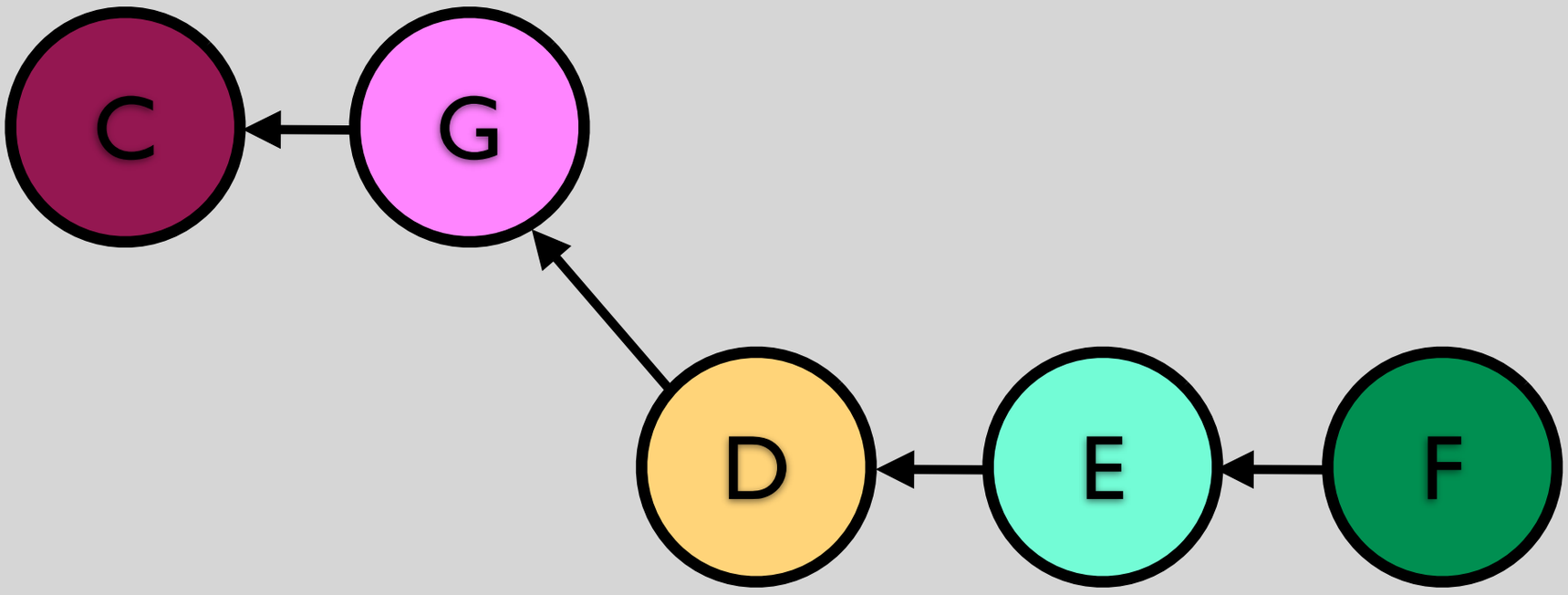

C
G
D
E
F

We now need to merge these two branches

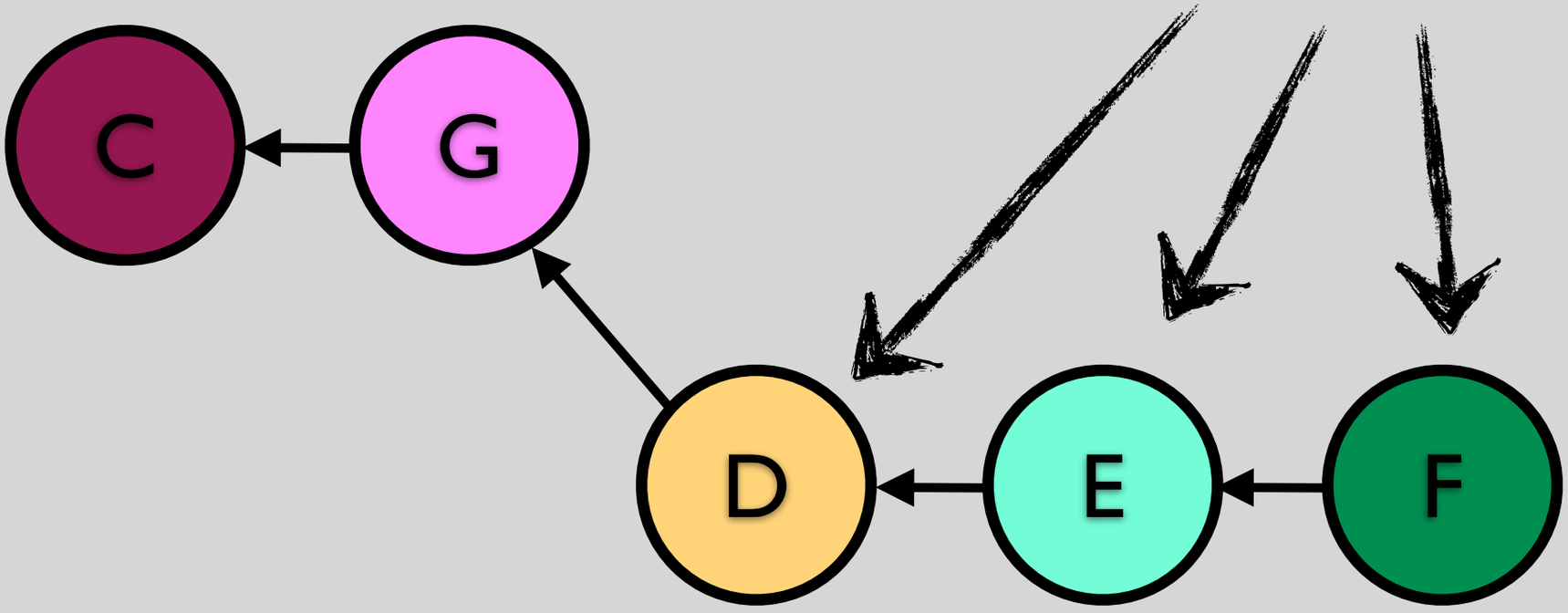


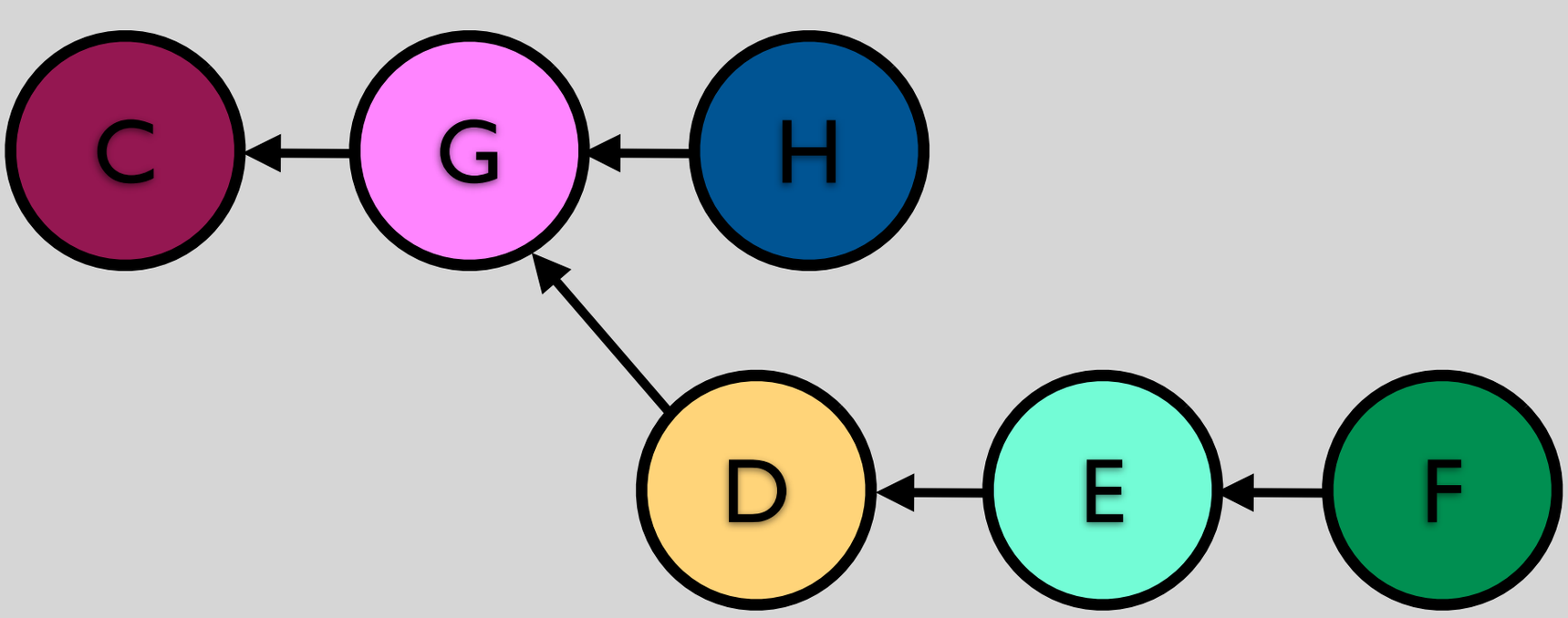

C
G

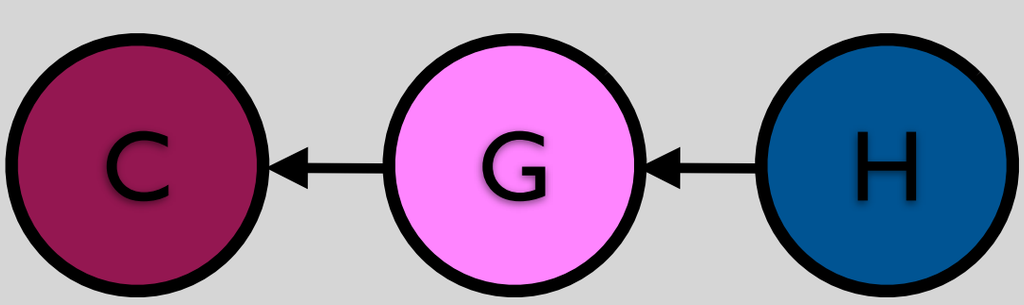

C
G

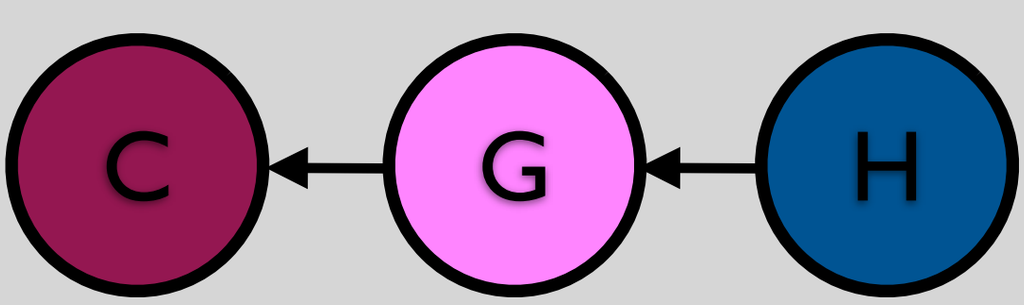

C
G
D
E
F

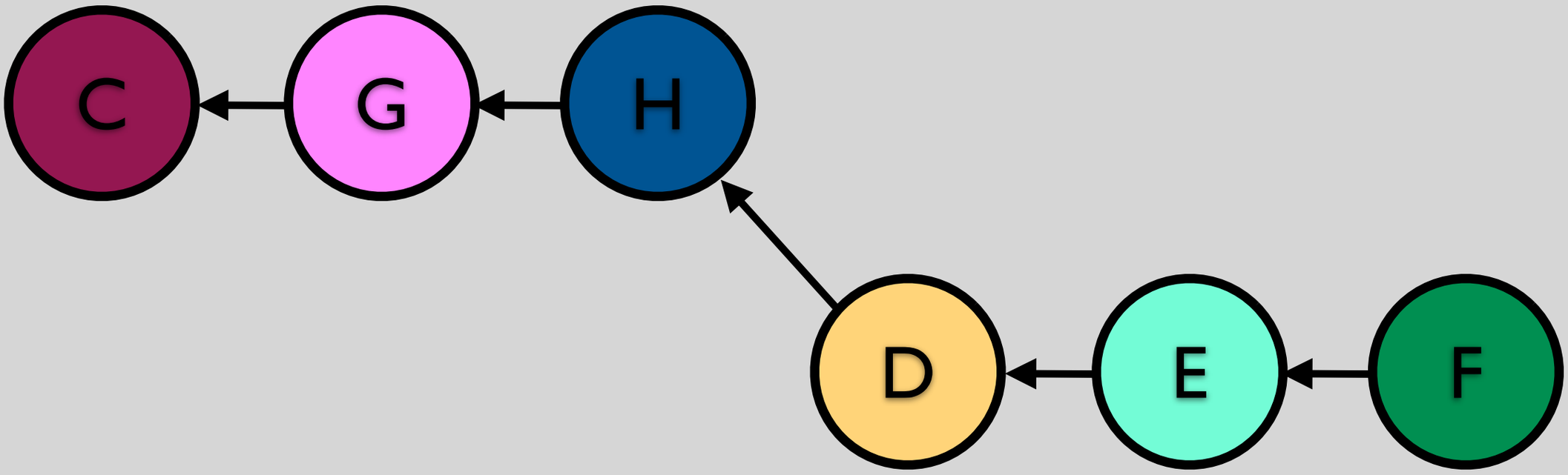
Same changes,
but new commits



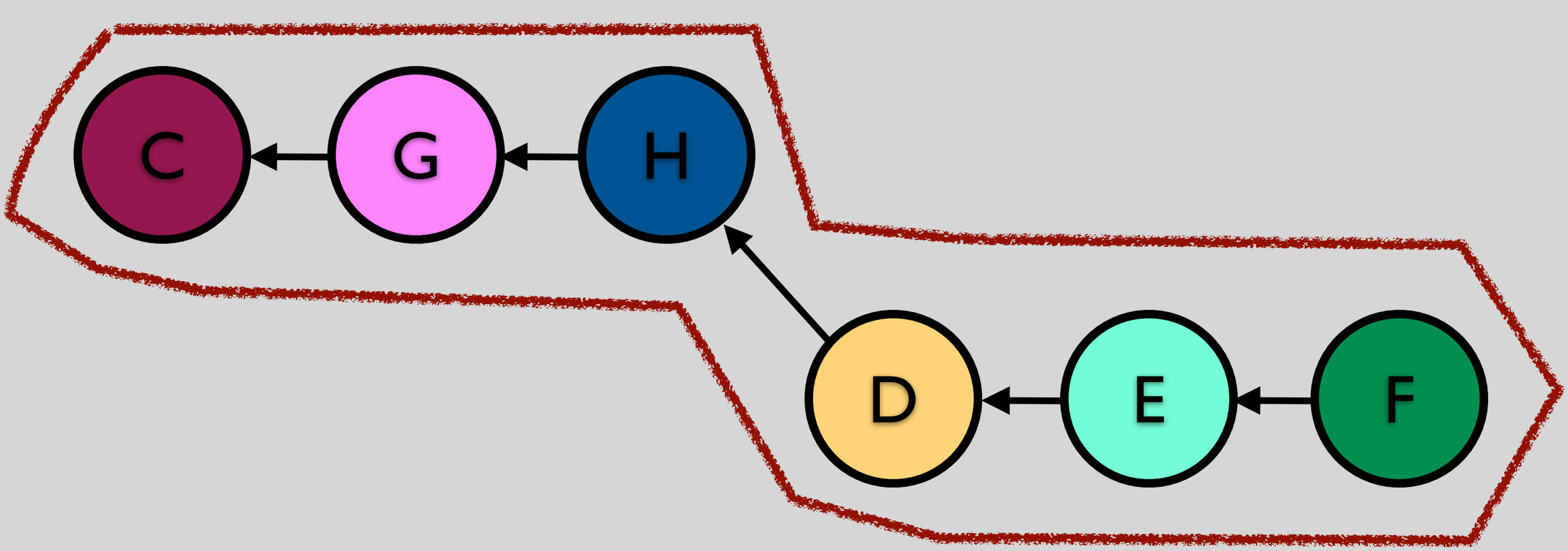

C
G
H
D
E
F


C
G
H


C
G
H


C
G
H
D
E
F

Effectively there
is no branch

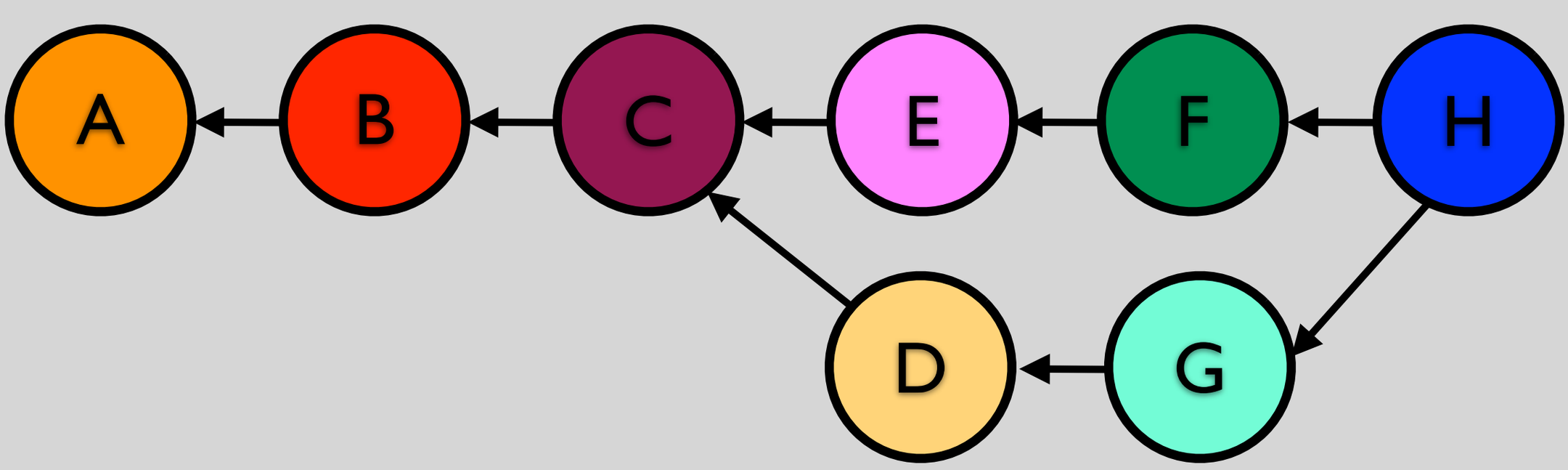


# NO MORE

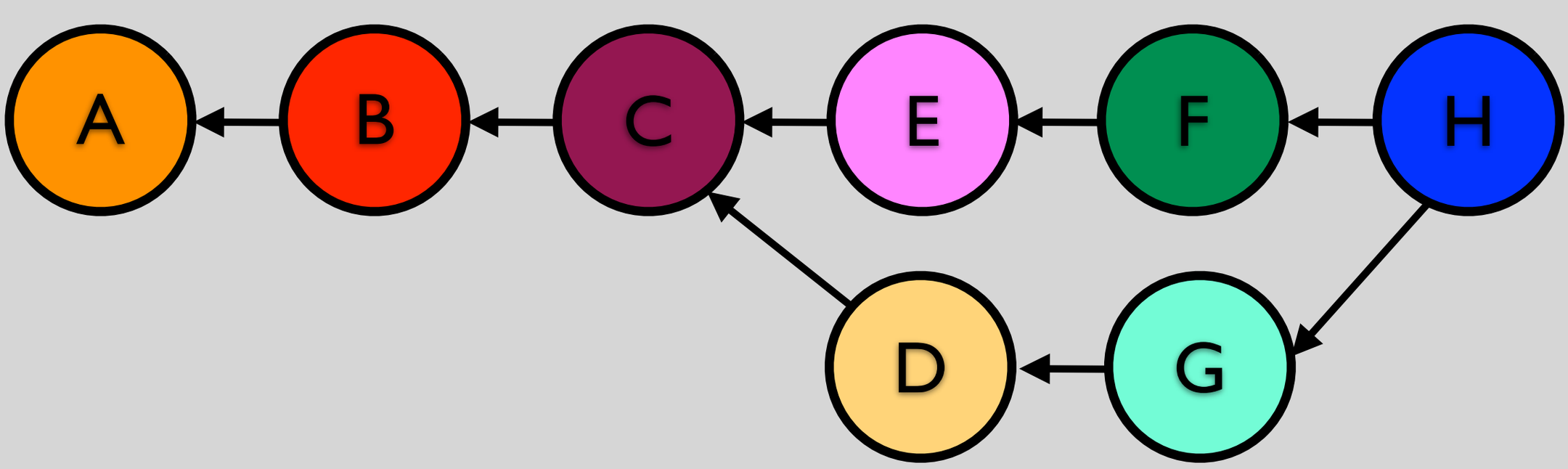
# MERGE HELL

This is a key data structure in Git:

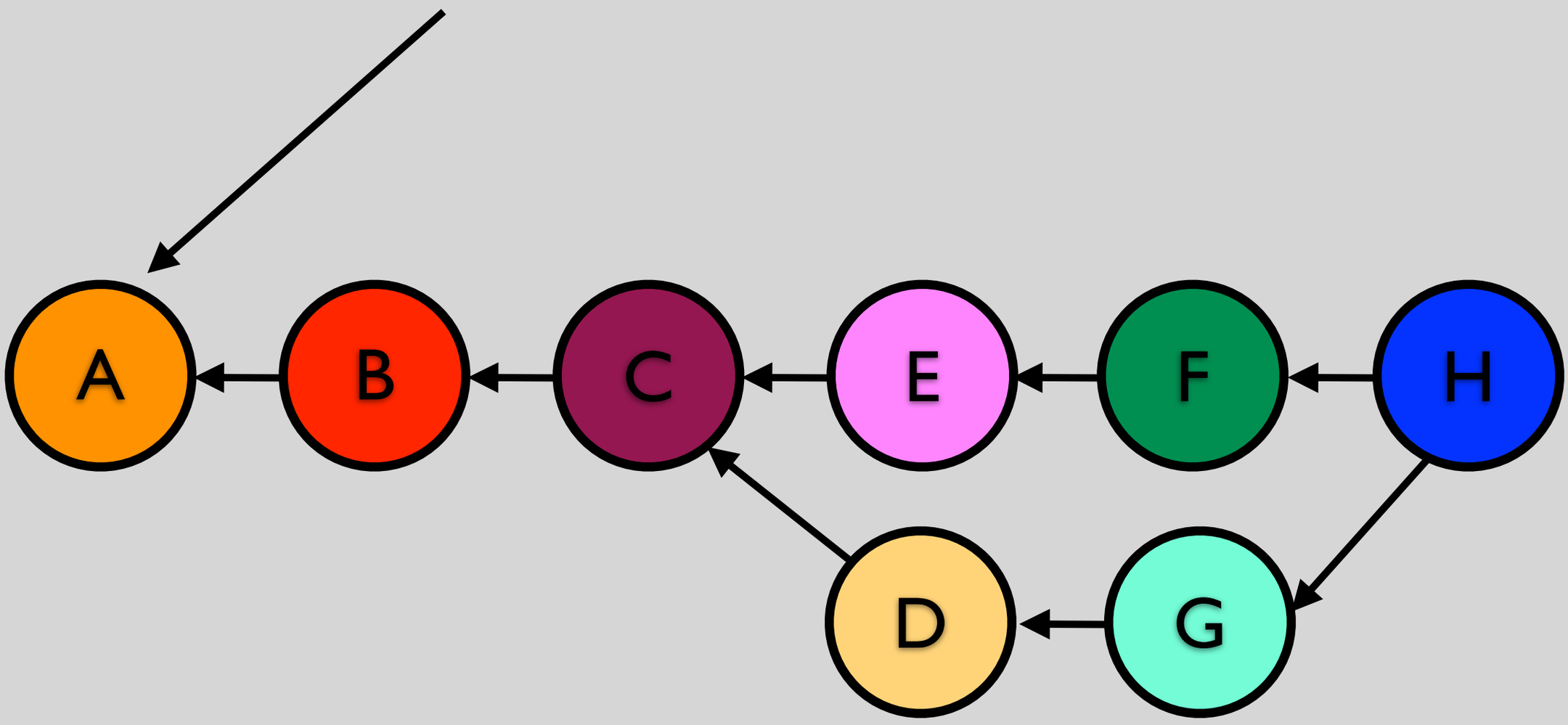
Directed Acyclic Graph,
or DAG



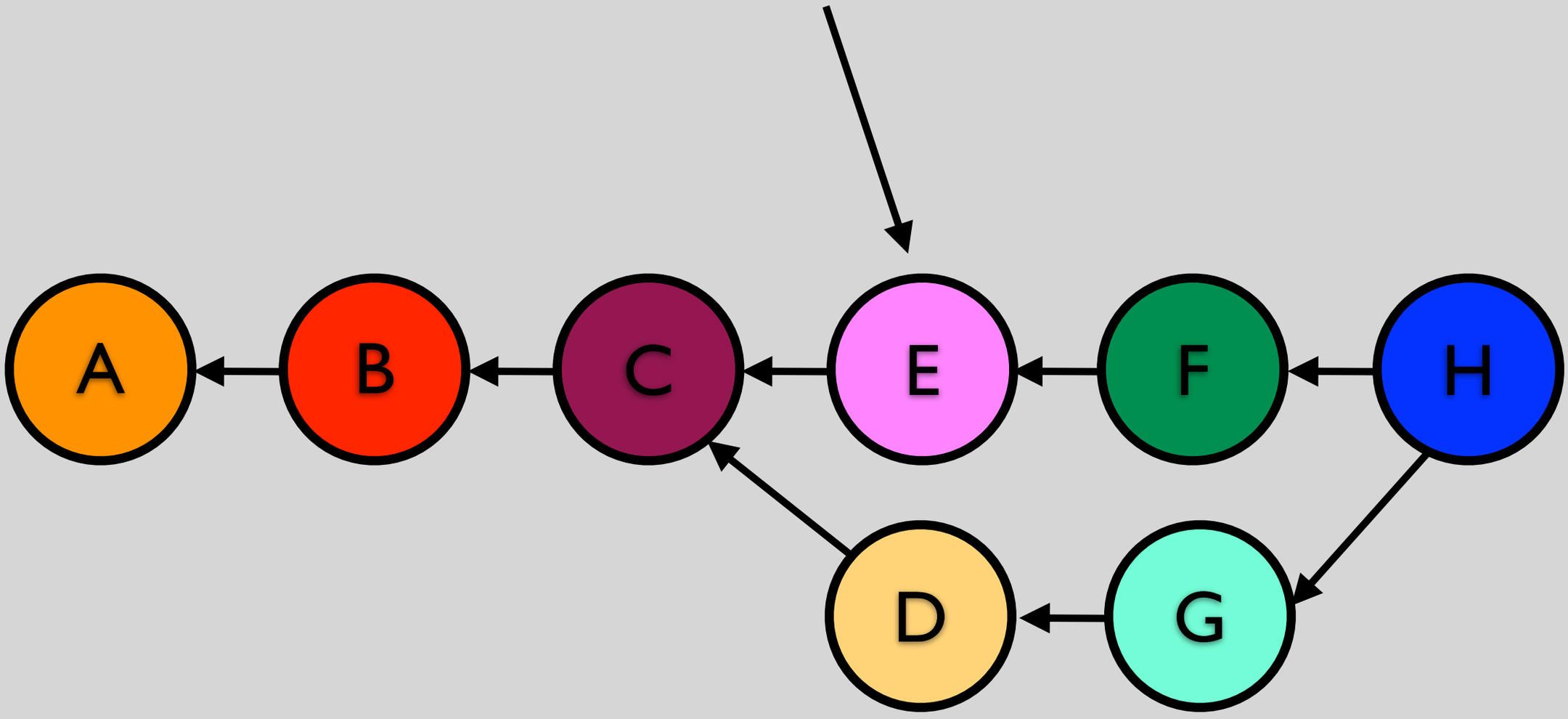
Each commit is addressable via a SHA-1 hash



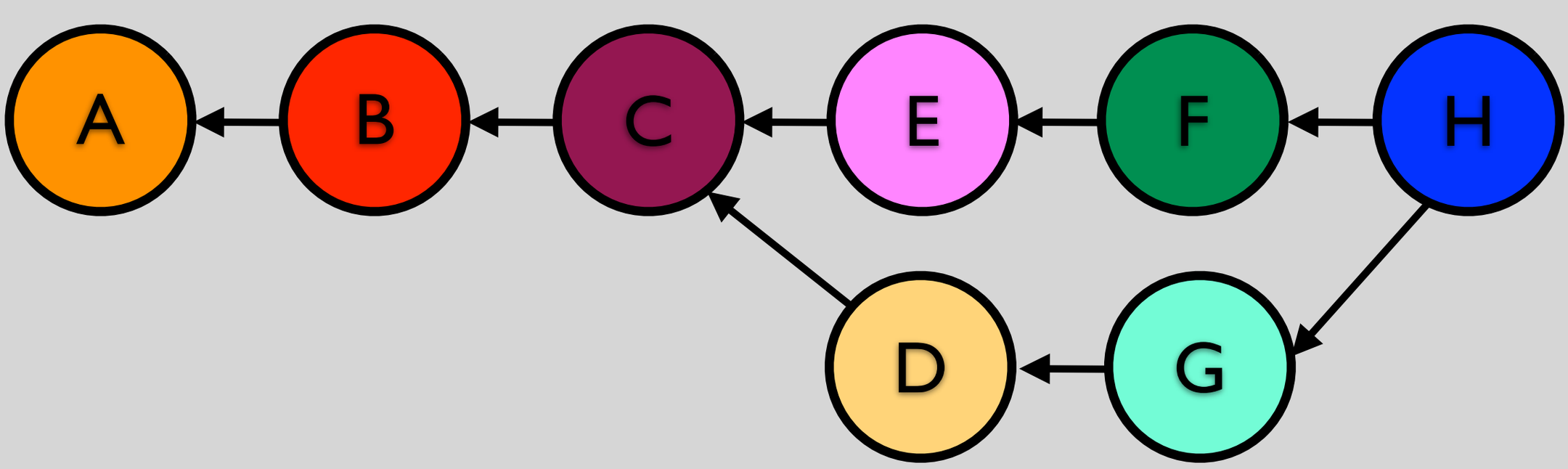
04c232610c981bd6a1cfe222f967adde036034b6



9b37ee782b63acf30fdc6c4f9c43d6657817f104

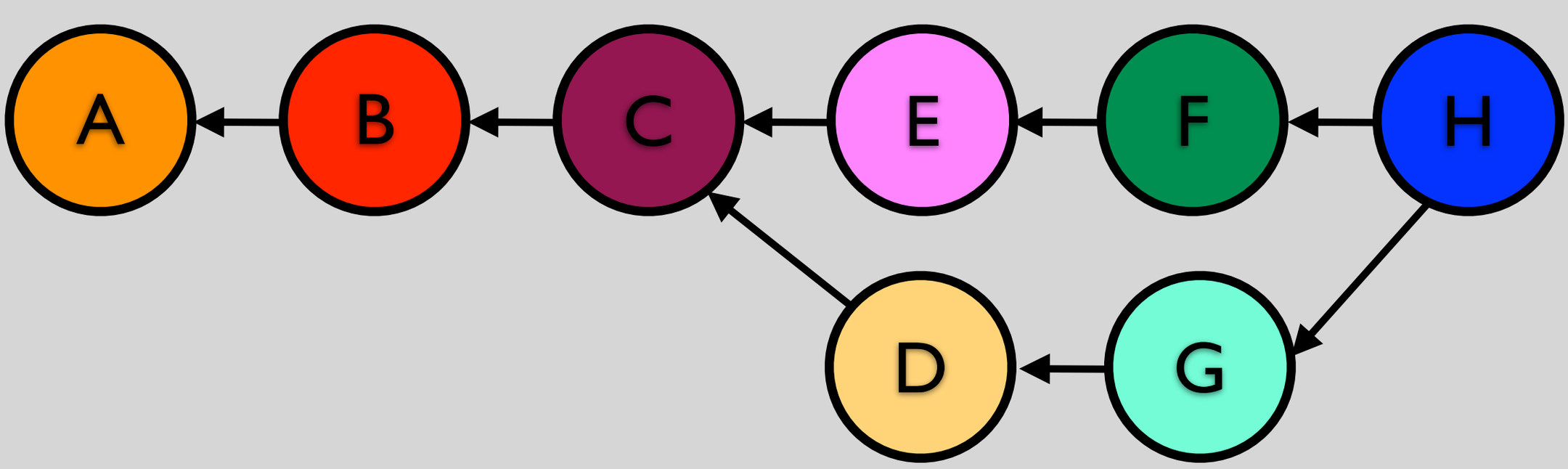


Using this, you can do some interesting things:



Using this, you can do some interesting things:

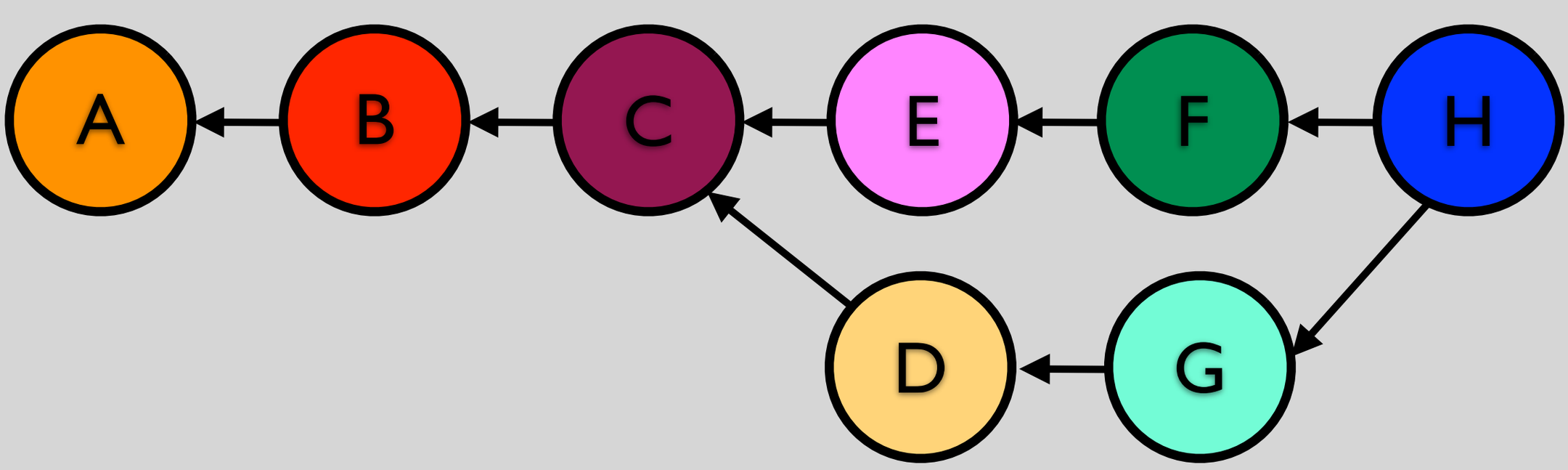
diff



Using this, you can do some interesting things:

diff

branch

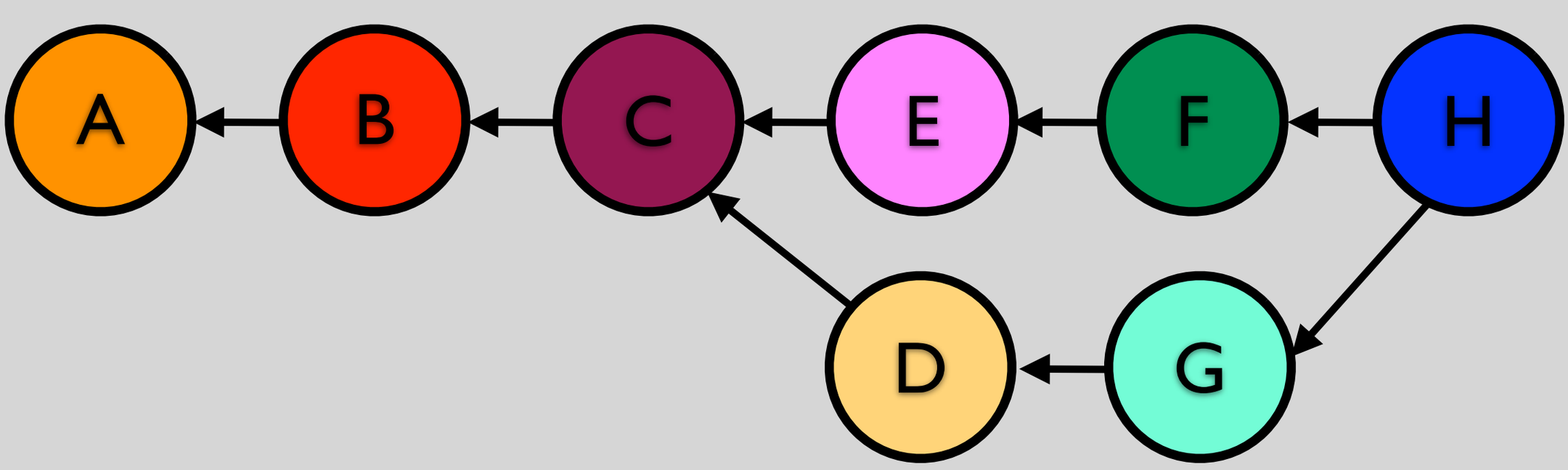


Using this, you can do some interesting things:

diff

tag

branch



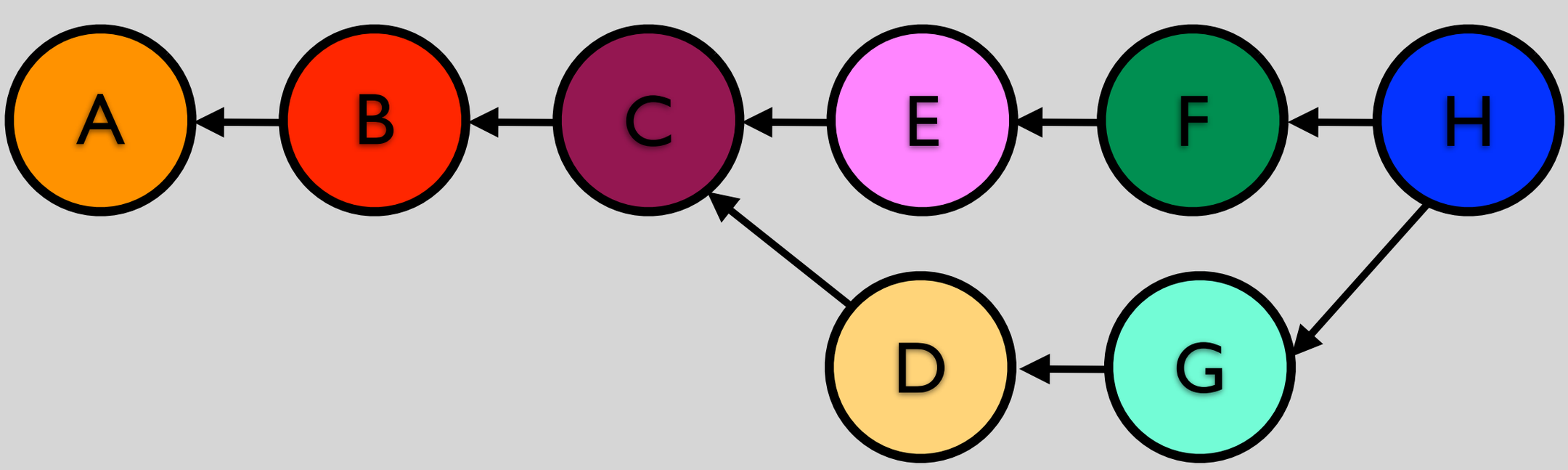
Using this, you can do some interesting things:

diff

tag

branch

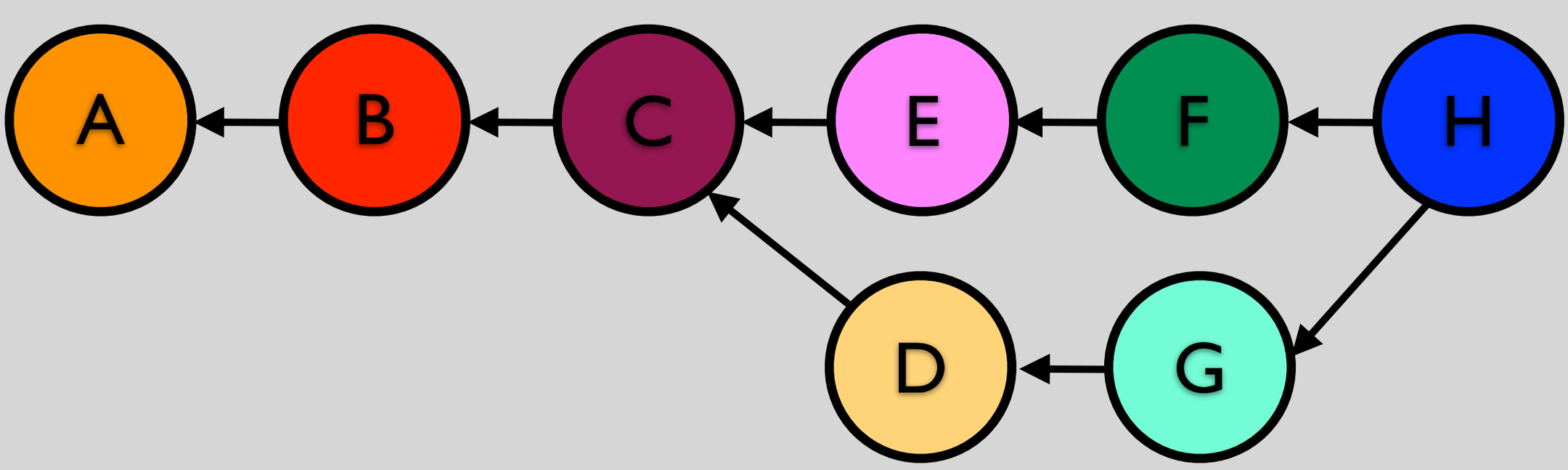
revert



Using this, you can do some interesting things:

diff tag checkout

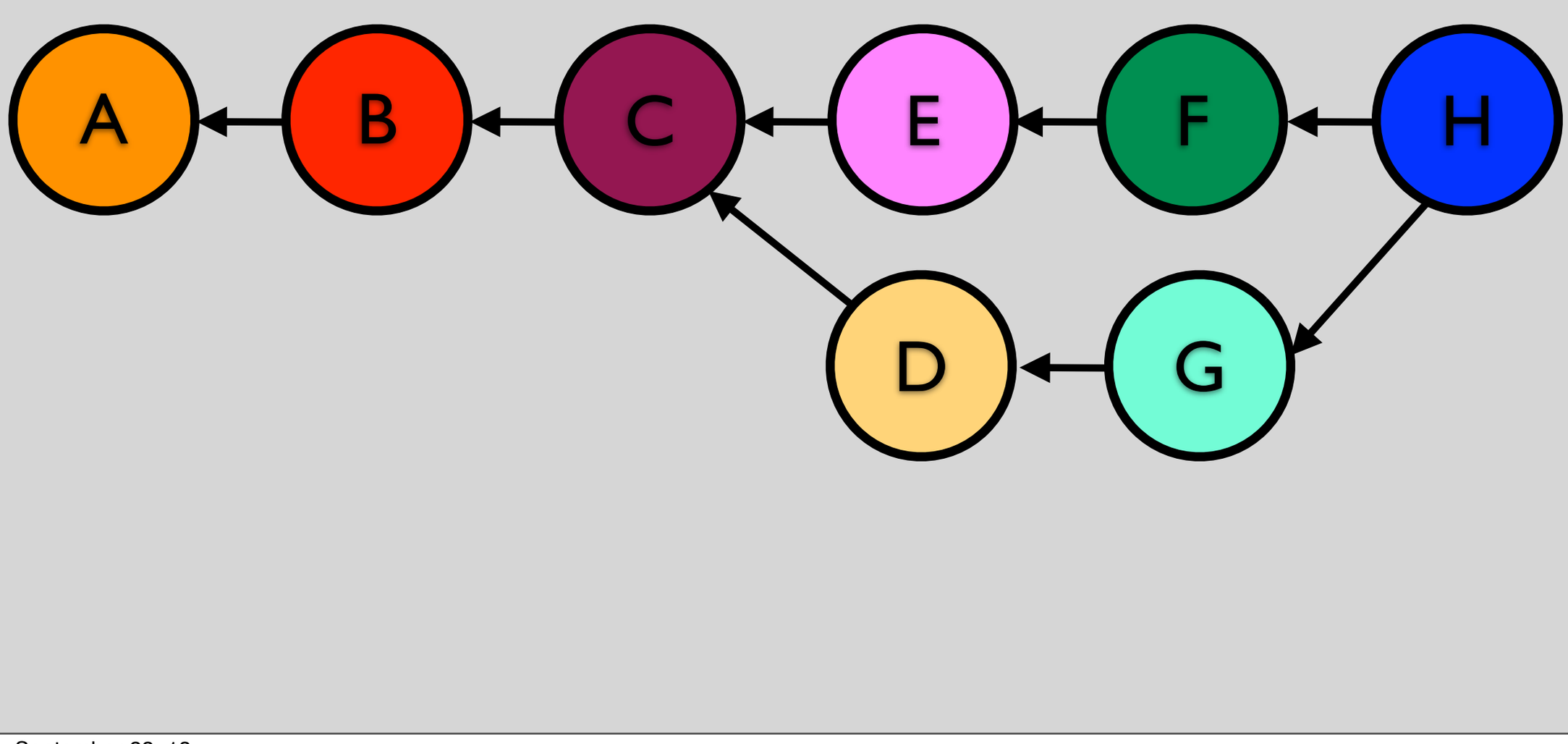
branch revert

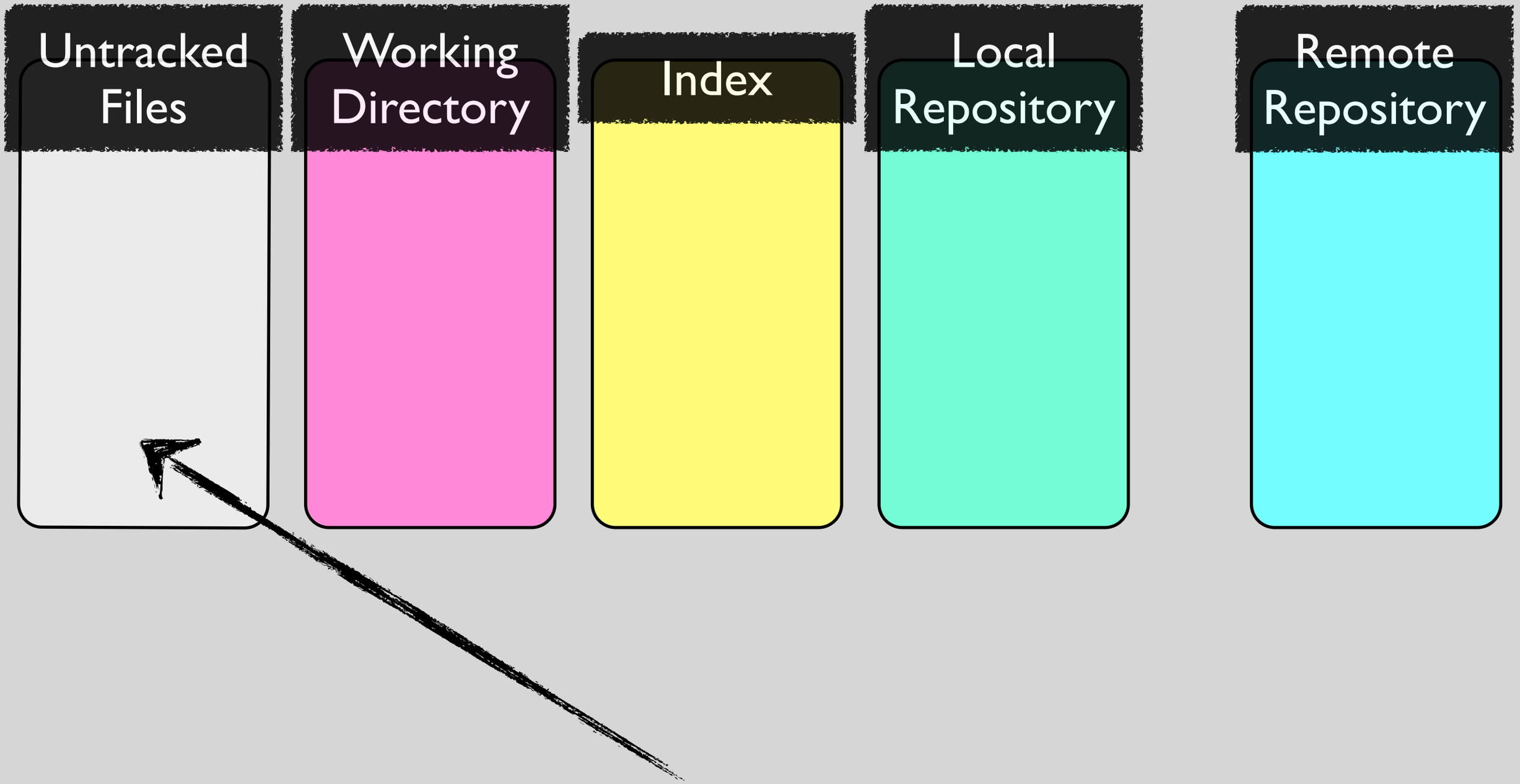


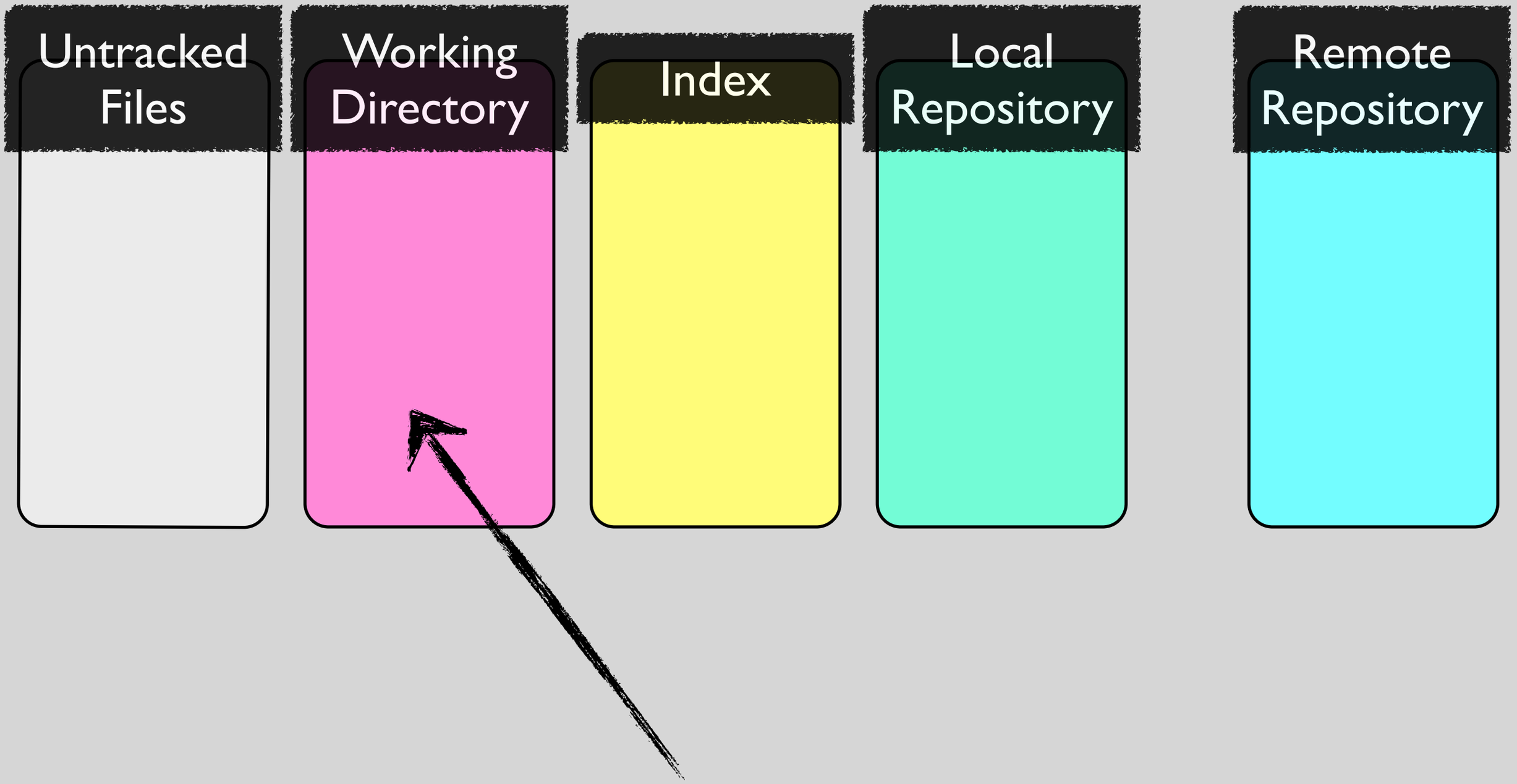
Using this, you can do some interesting things:

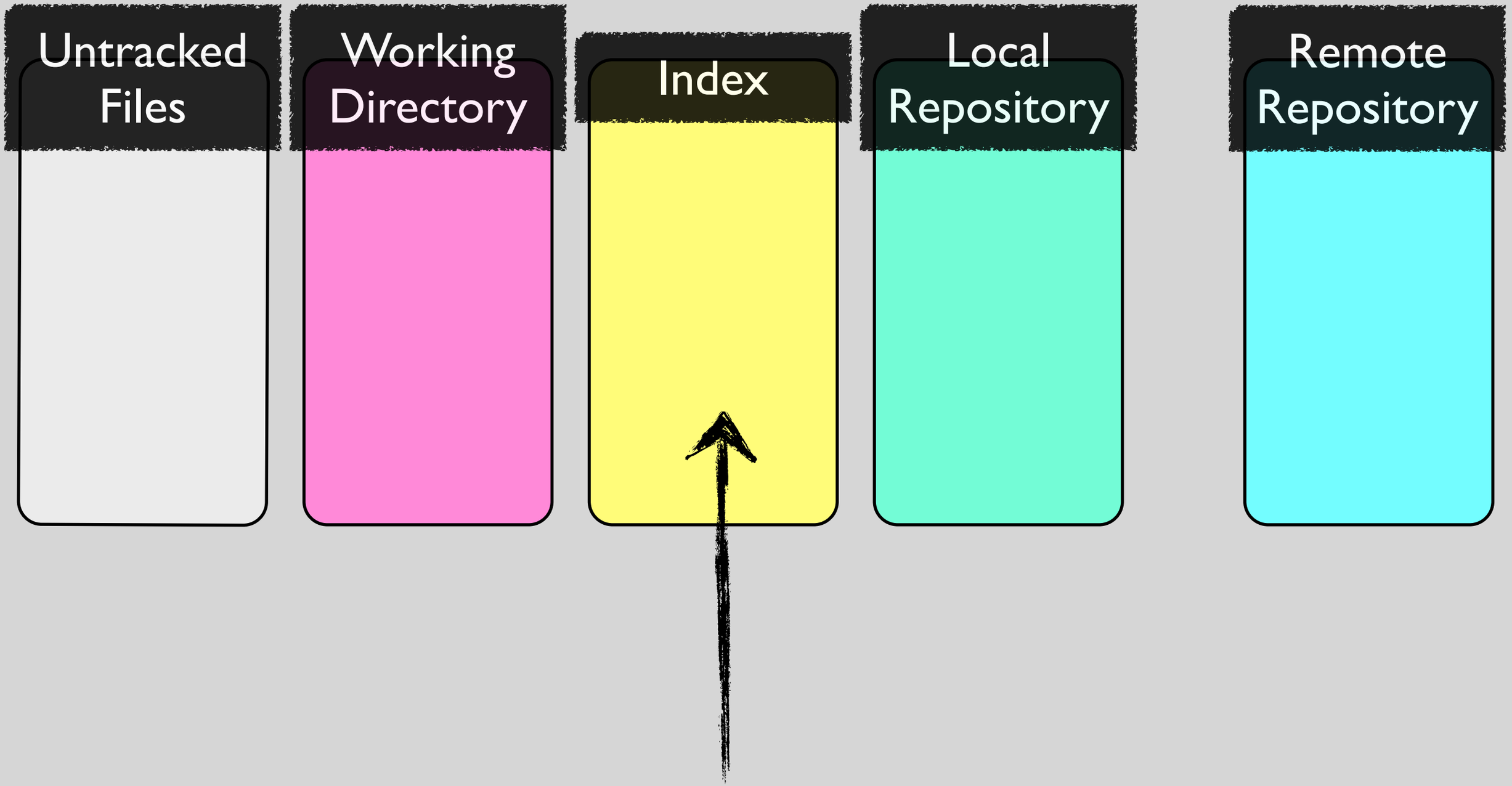
diff tag checkout

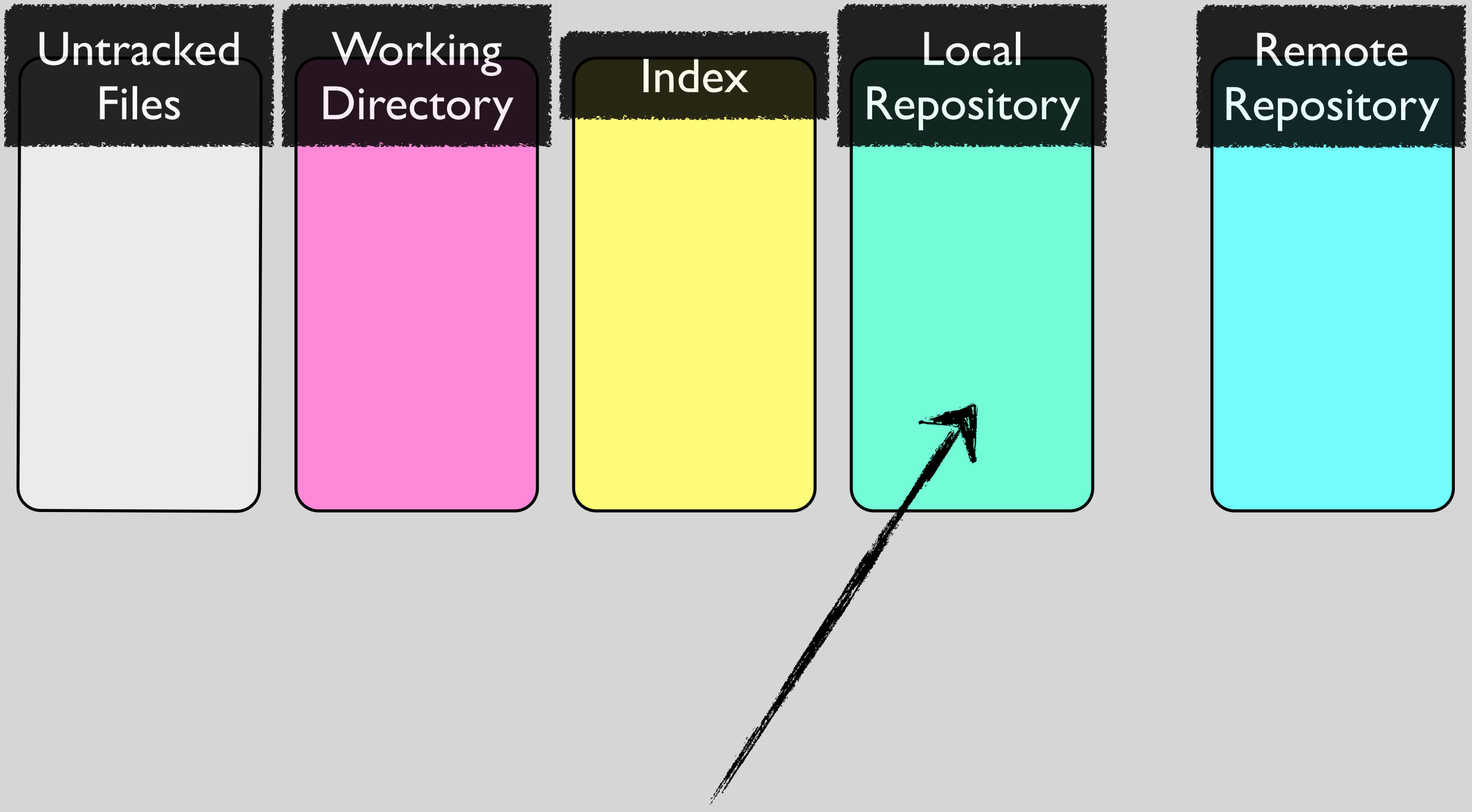
branch revert cherry-pick

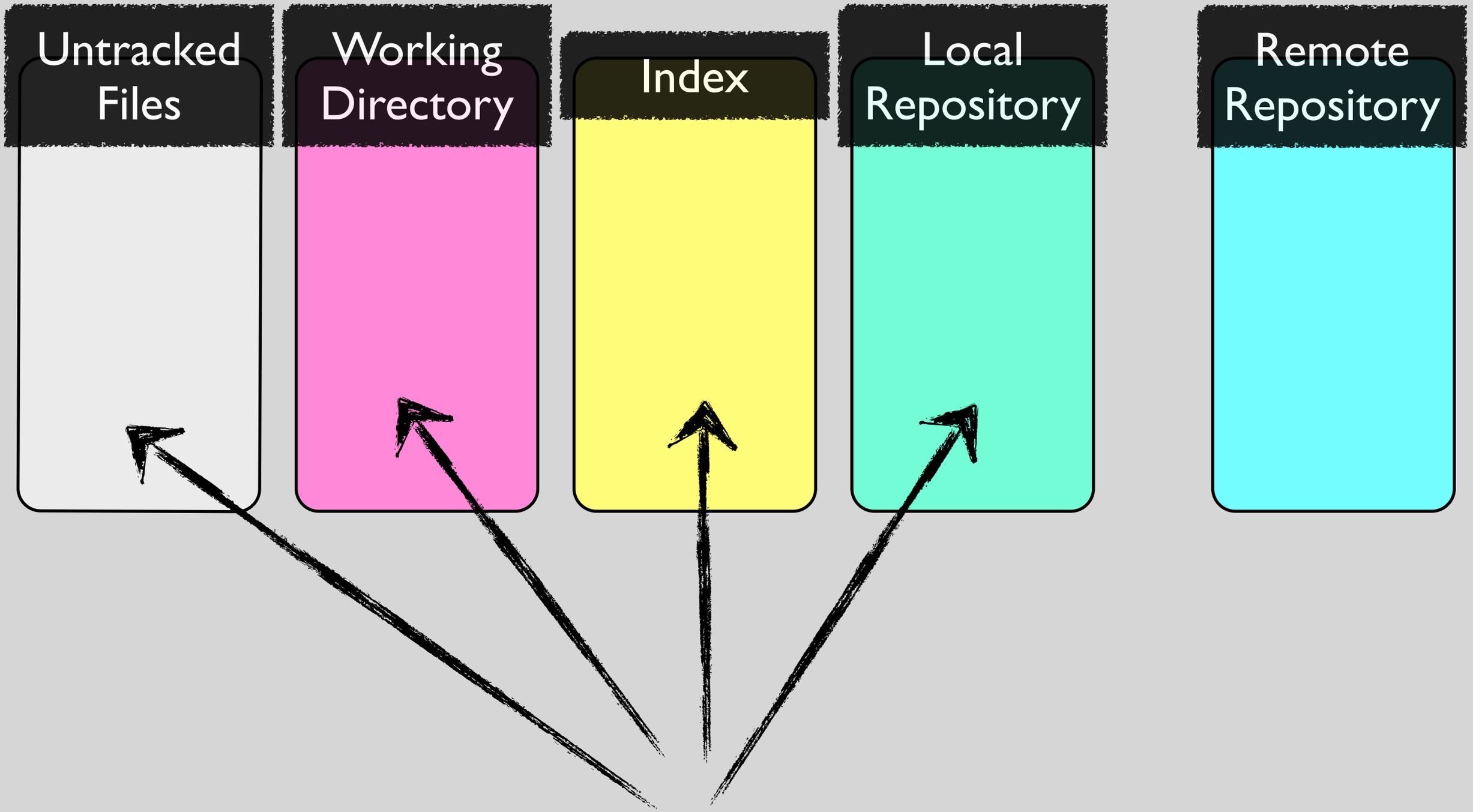


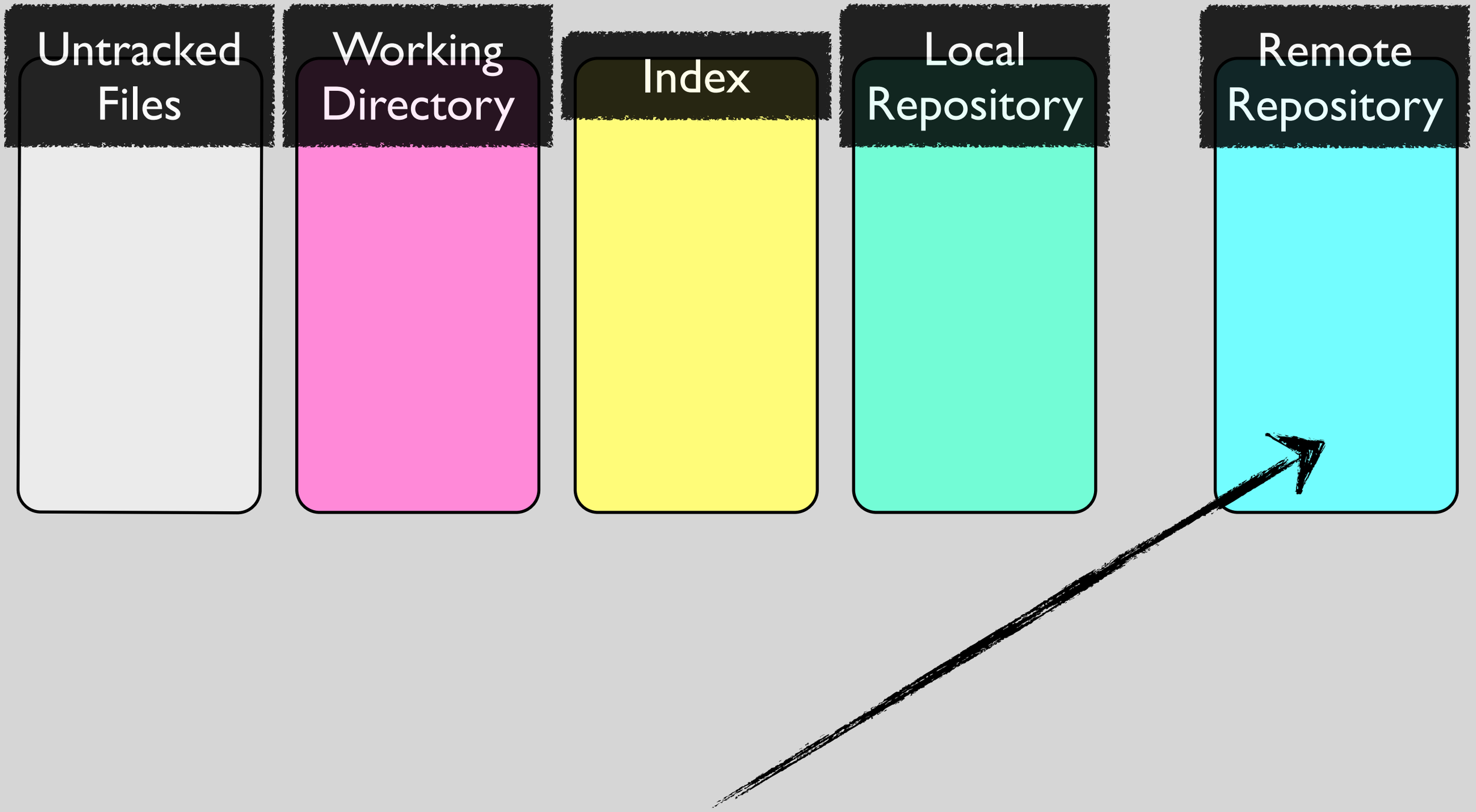

Untracked Files
Working Directory
Index
Local Repository
Remote Repository
Files that Git does not "know" about


Untracked Files
Working Directory
Index
Local Repository
Remote Repository
Files that Git is tracking


Untracked Files
Working Directory
Index
Local Repository
Remote Repository
Change that are staged to be committed


Untracked Files
Working Directory
Index
Local Repository
Remote Repository
Your local Git repository.


Untracked Files
Working Directory
Index
Local Repository
Remote Repository
These are all local.


Untracked Files
Working Directory
Index
Local Repository
Remote Repository
Remote Git repository.

# Git Commands

```
add          fetch      rebase
blame        init       remote
branch       log        reset
checkout     merge      rm
clone        mv         stash
commit       pull       status
diff         push       tag
```

# Initializing Projects

```
add          fetch        rebase
blame        init         remote
branch       log          reset
checkout     merge        rm
clone        mv           stash
commit       pull         status
diff         push         tag
```

# Managing Local Commits

```
add
blame
branch
checkout
clone
commit
diff
fetch
init
log
merge
mv
pull
push
rebase
remote
reset
rm
stash
status
tag
```

# Branching and Merging

```
add       fetch     rebase
blame     init      remote
branch    log       reset
checkout  merge     rm
clone     mv        stash
commit    pull      status
diff      push      tag
```

# Sharing and Updating

```
add         fetch     rebase
blame       init      remote
branch      log       reset
checkout    merge     rm
clone       mv        stash
commit      pull      status
diff        push      tag
```

SALT-N-PEPA

SALT-N-PEPA

THIS SIDE

# PUSH IT

(U.S. REMIX)

THAT SIDE

# I AM DOWN

•

# TRAMP

Hyped-up by

HURB

The Supa Def Dope Produsa

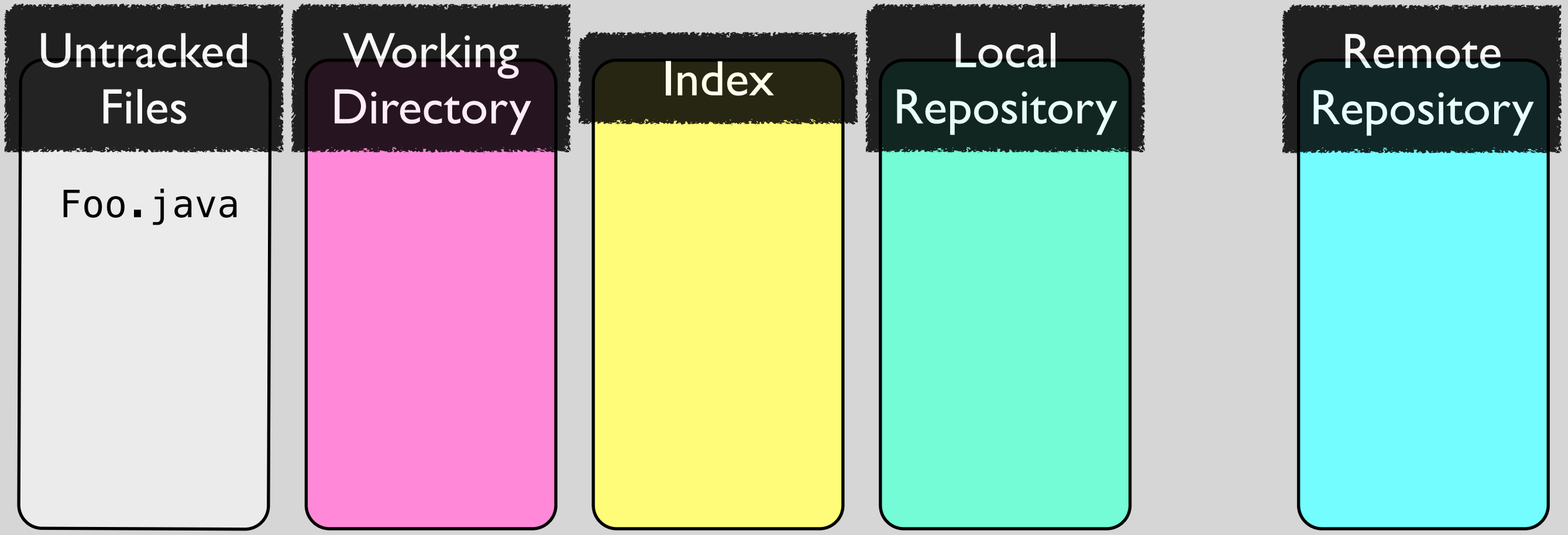
Untracked Files

Working Directory

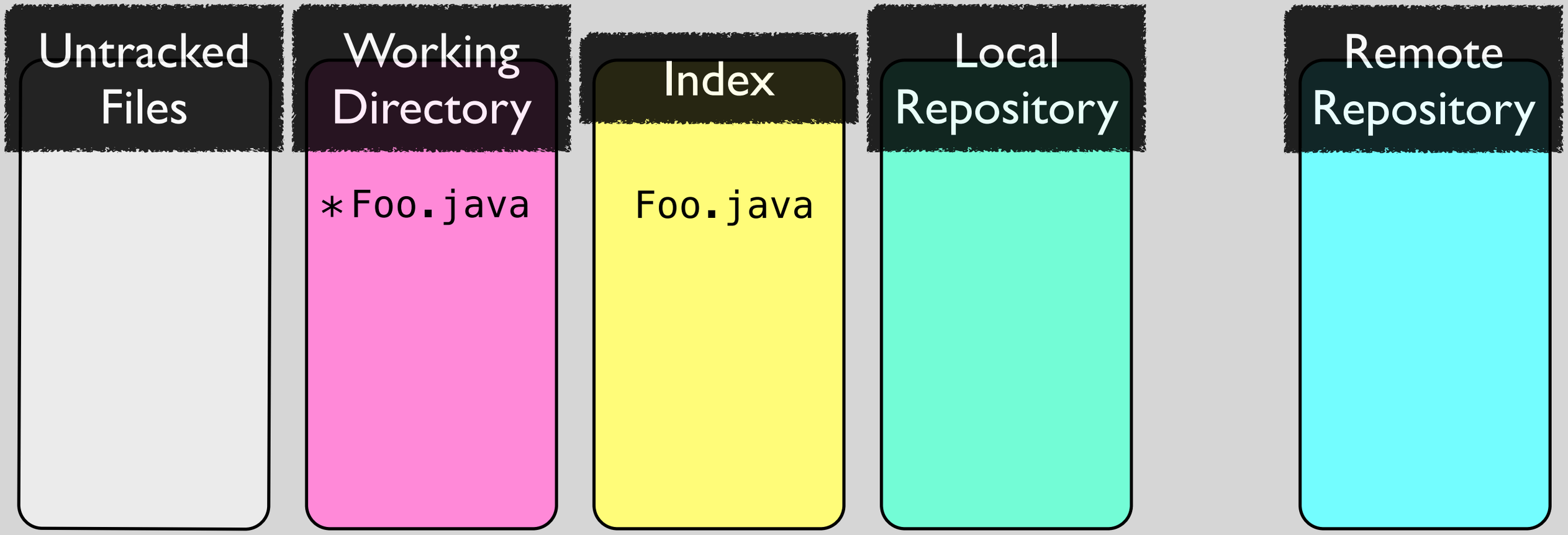
Index

Local Repository

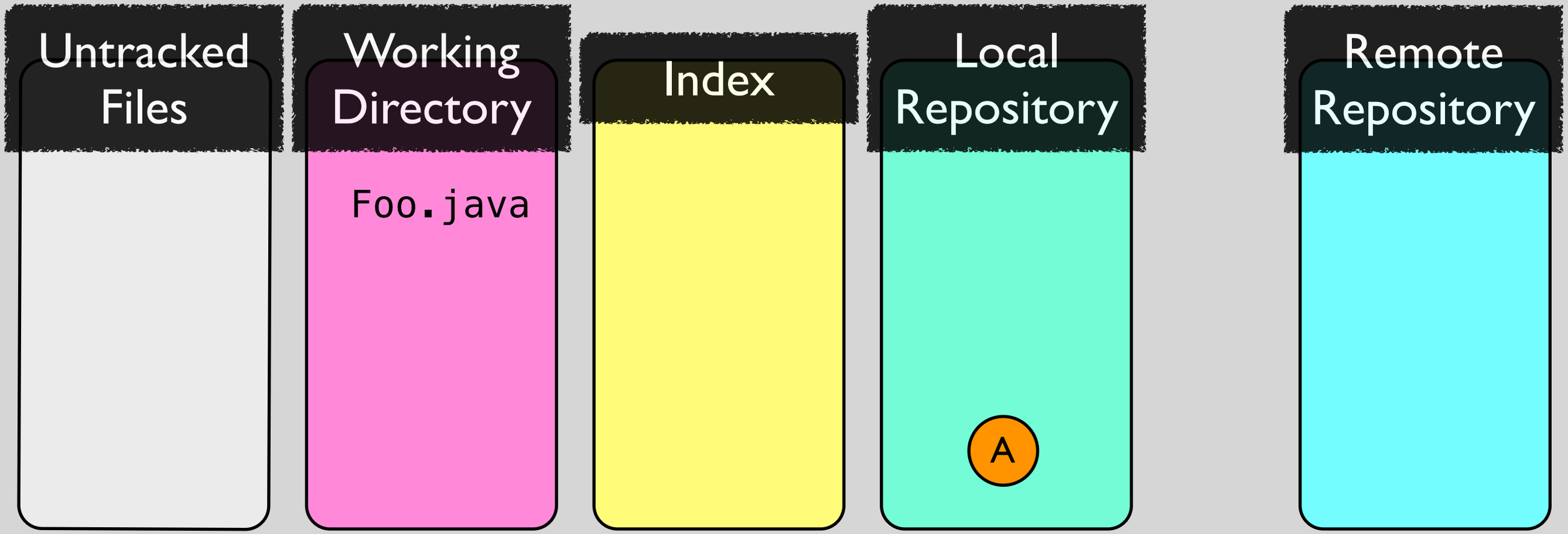
Remote Repository


Untracked Files
Foo.java
Working Directory
Index
Local Repository
Remote Repository

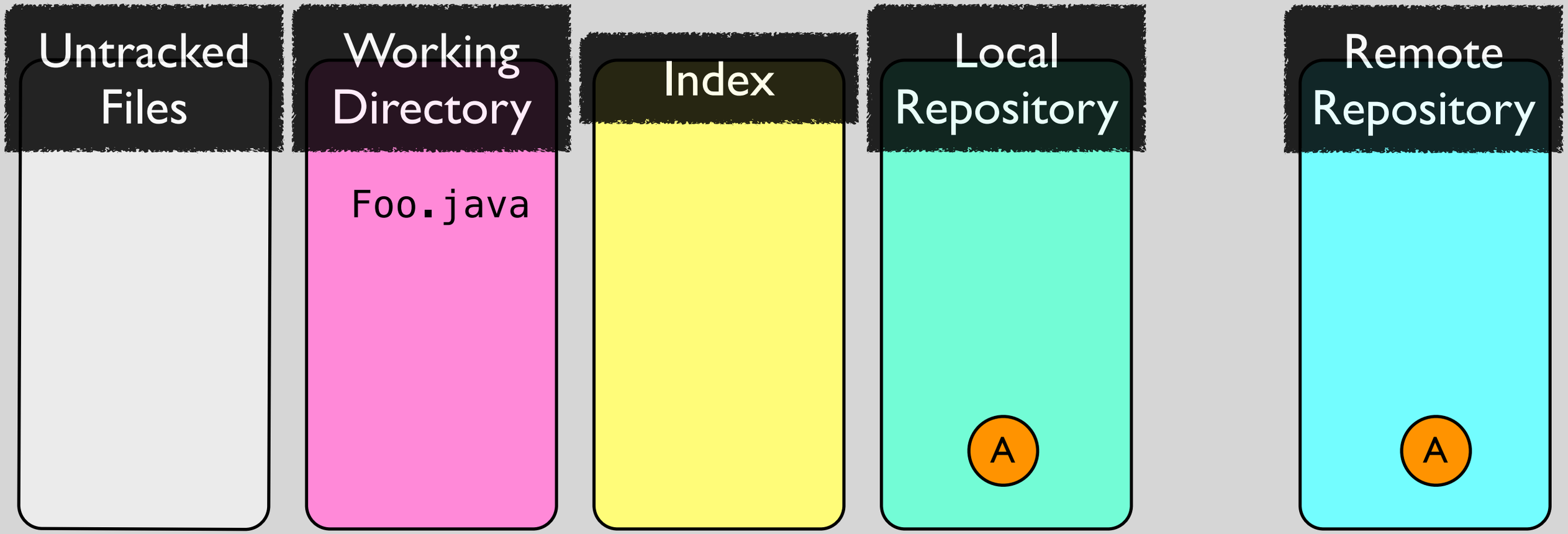
```
vi Foo.java
```



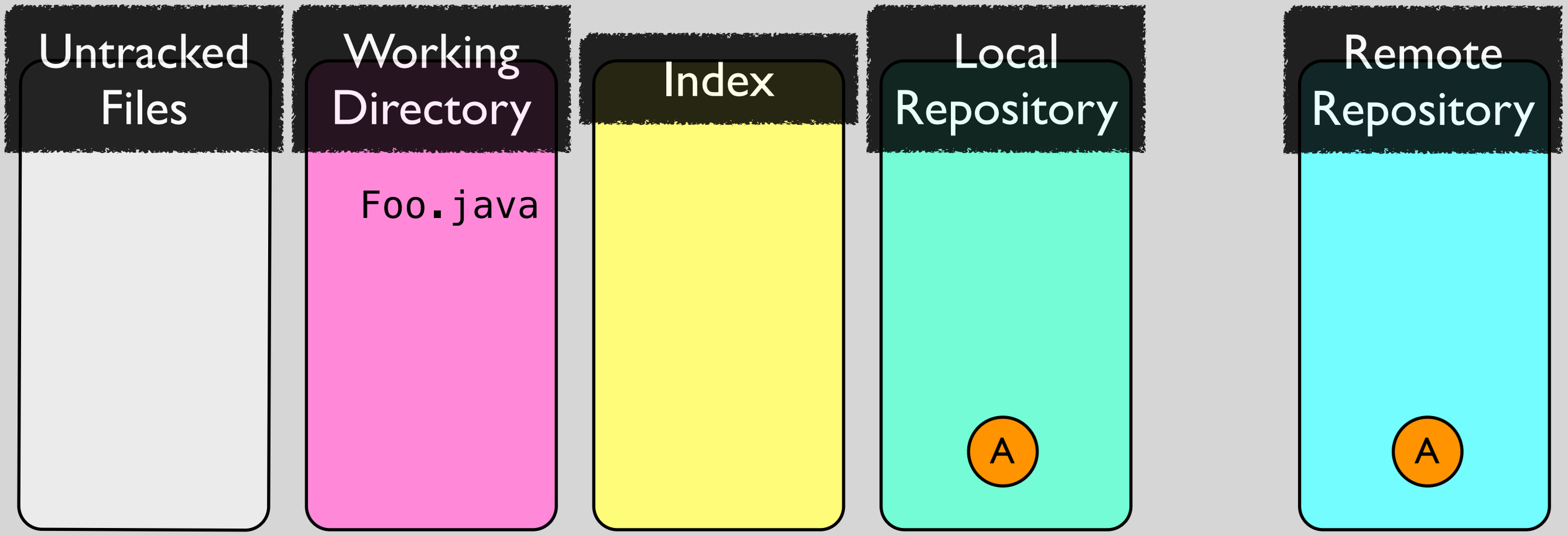

```
vi Foo.java
git add Foo.java
```

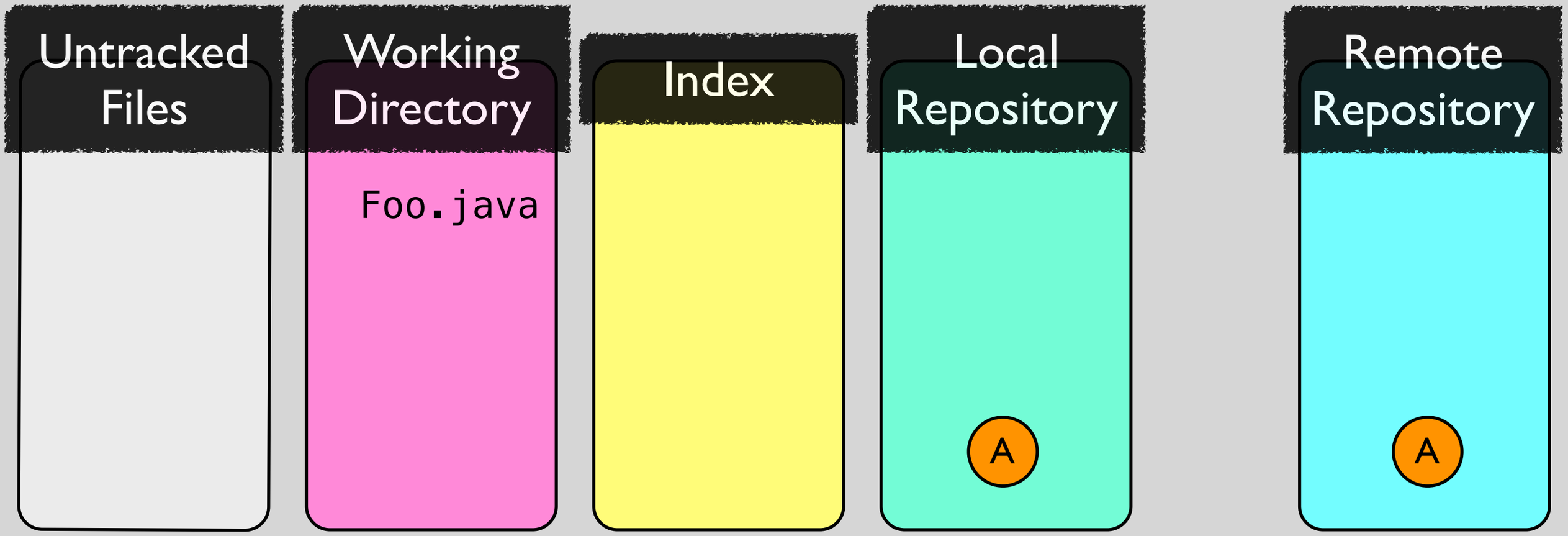



```
vi Foo.java
git add Foo.java
git commit –m “Added Foo.java”
```

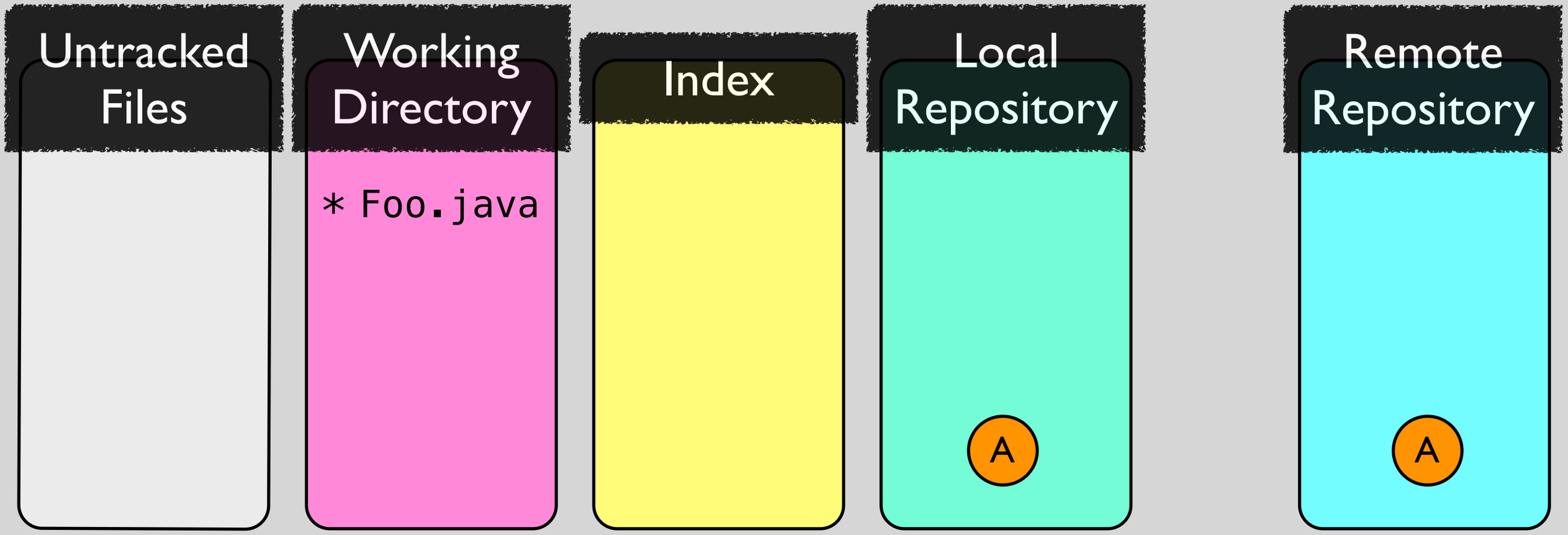



```
vi Foo.java
git add Foo.java
git commit –m “Added Foo.java”
git push origin master
```

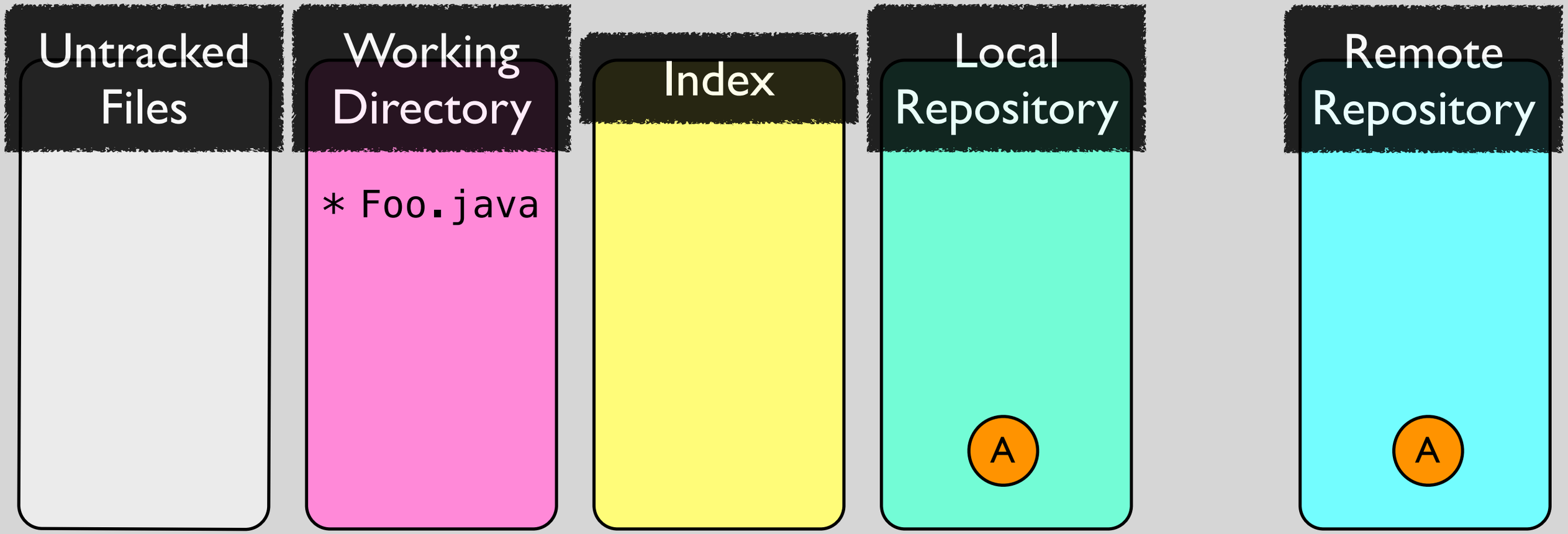

Untracked Files
Working Directory
Foo.java
Index
Local Repository
A
Remote Repository
A


Untracked Files
Working Directory
Foo.java
Index
Local Repository
A
Remote Repository
A

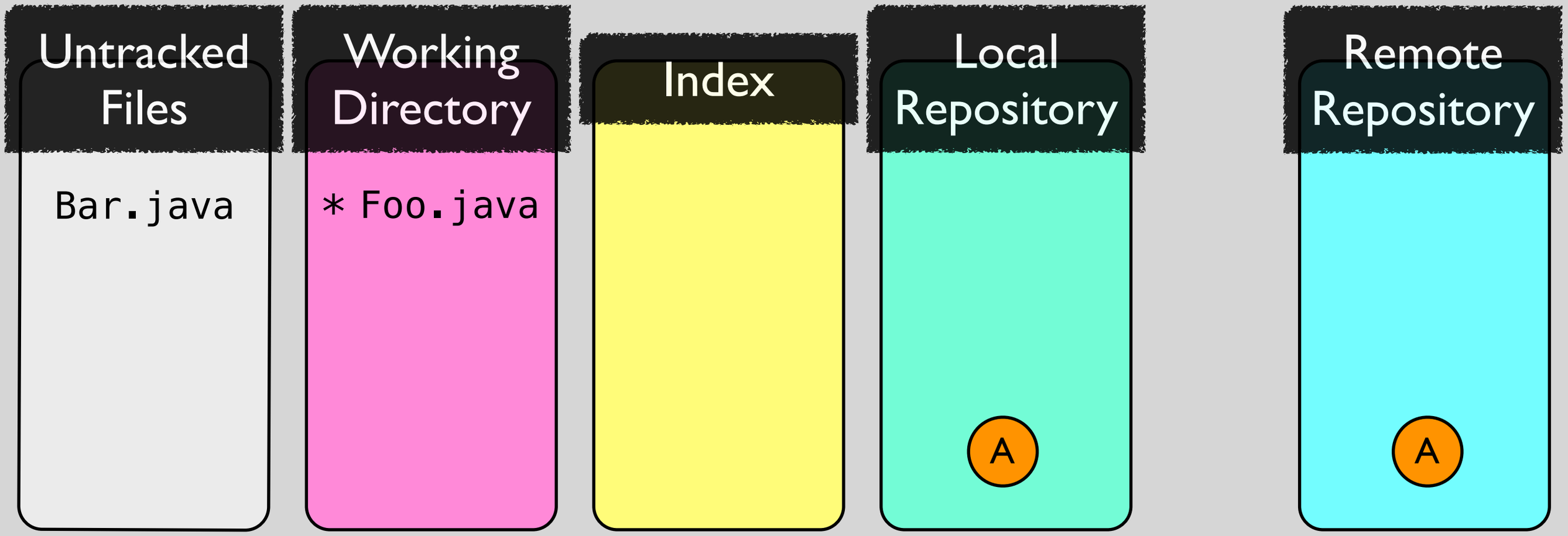
```
vi Foo.java
```


Untracked Files
Working Directory
* Foo.java
Index
Local Repository
A
Remote Repository
A

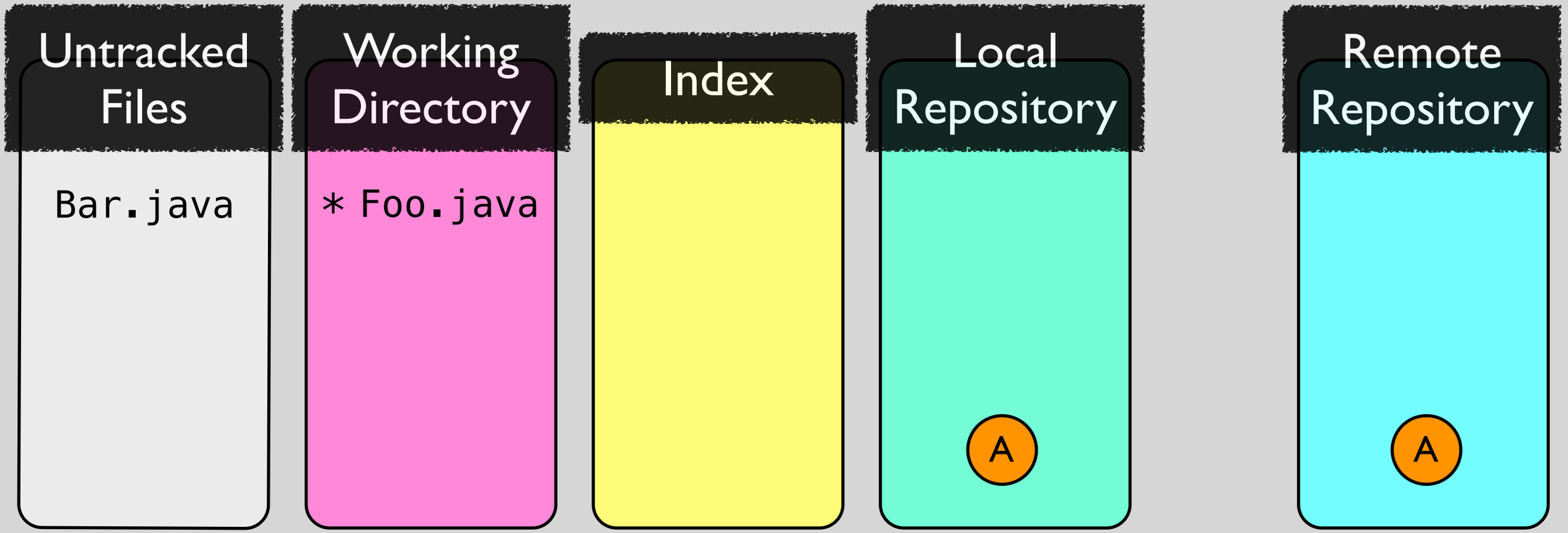
```
vi Foo.java
```



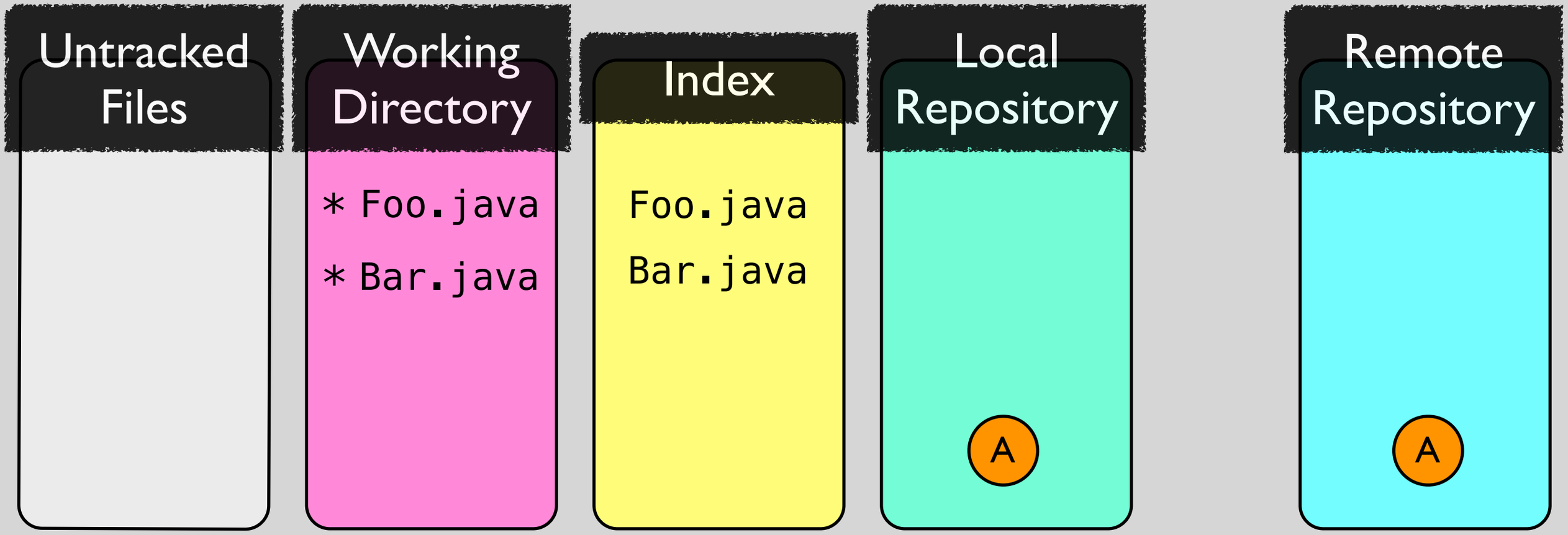

```
vi Foo.java
vi Bar.java
```



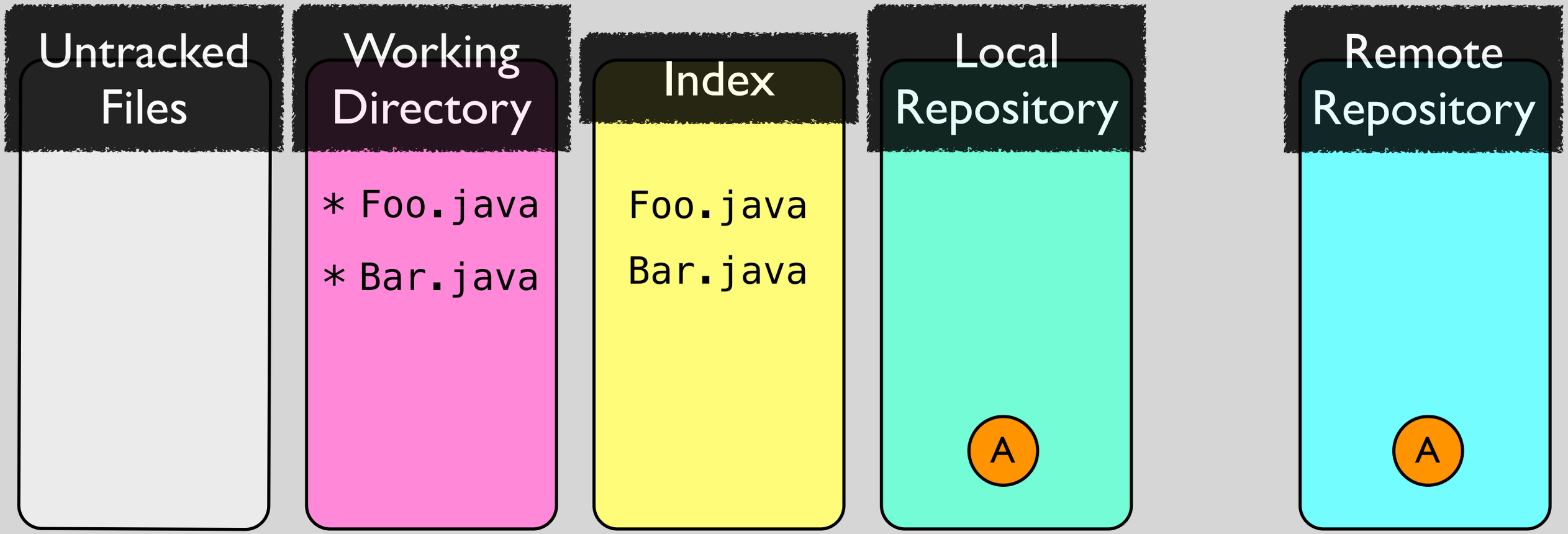

```
vi Foo.java
vi Bar.java
```



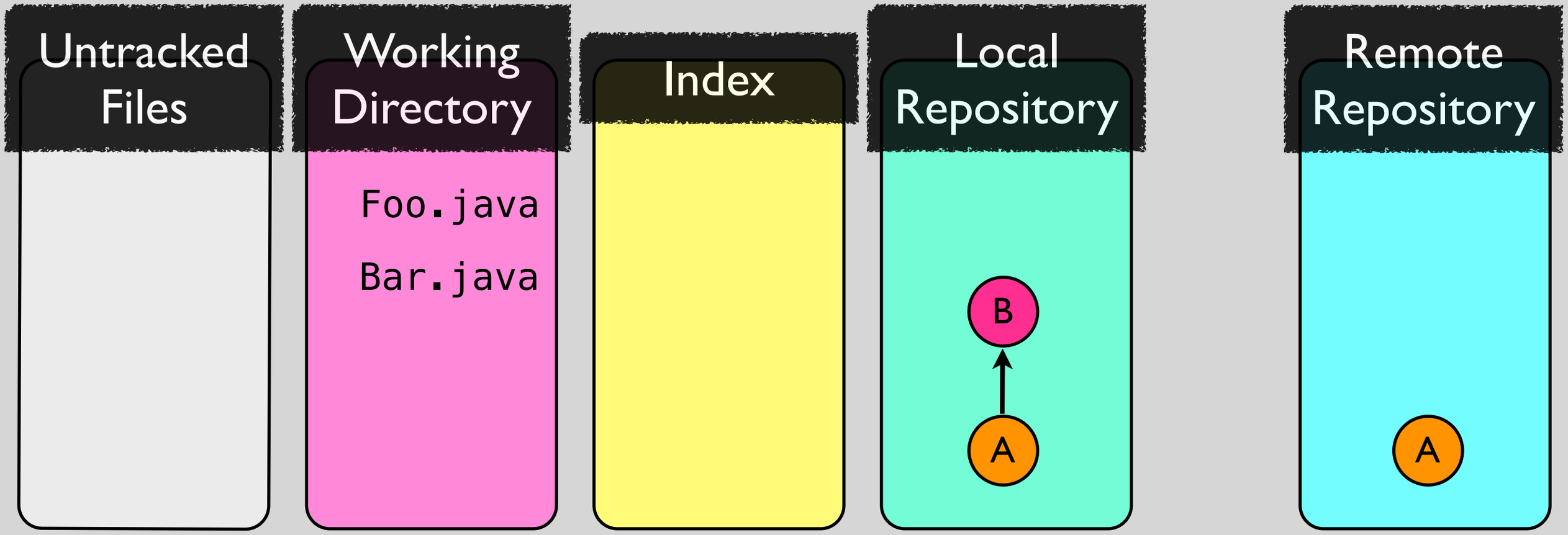

```
vi Foo.java
vi Bar.java
git add .
```



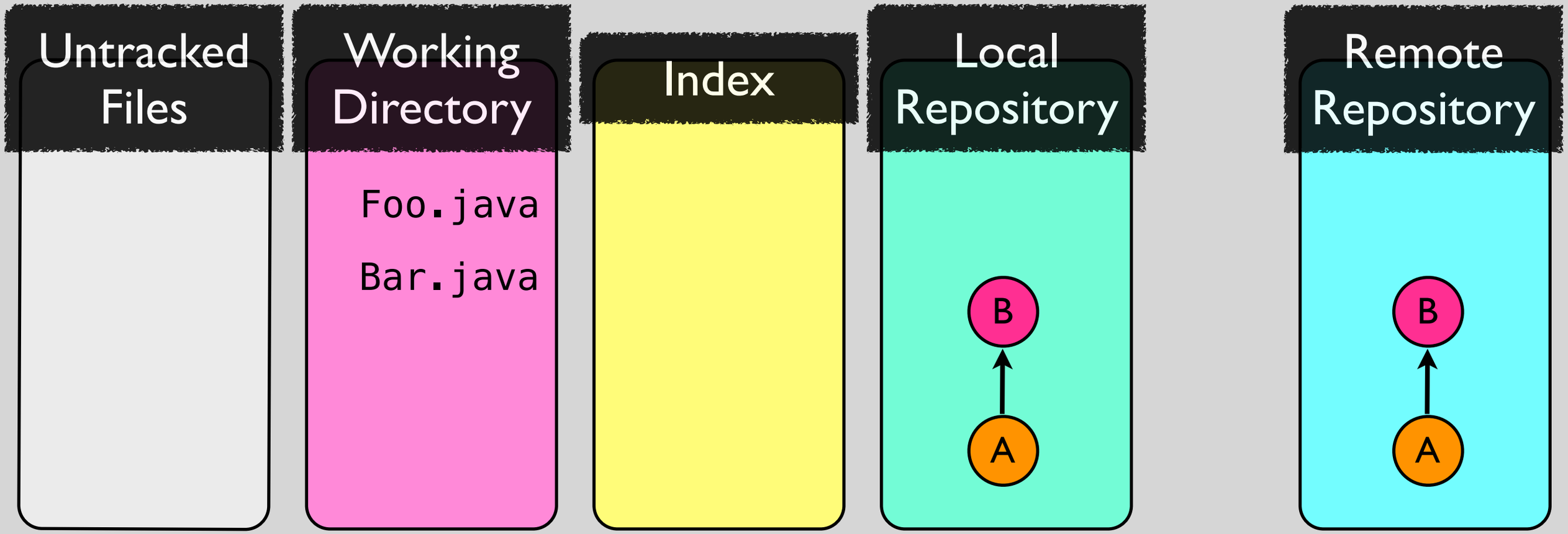

```
vi Foo.java
vi Bar.java
git add .
```




```
vi Foo.java
vi Bar.java
git add .
git commit –m “More changes”
```




```
vi Foo.java
vi Bar.java
git add .
git commit –m “More changes”
```




```
vi Foo.java
vi Bar.java
git add .
git commit –m “More changes”
git push origin master
```

# Live Examples

To the command line...

# GitHub Overview

# Why GitHub?

# Why GitHub?

- In “the cloud”

# Why GitHub?

- In “the cloud”
- Inexpensive

# Why GitHub?

- In "the cloud"
- Inexpensive
- Secure

# Why GitHub?

- In "the cloud"
- Inexpensive
- Secure
- Feature rich

# Why GitHub?

- In "the cloud"
- Inexpensive
- Secure
- Feature rich
- Stable

# Live Examples

## To GitHub...

# Git and GitHub Adoption

# Where to start

# Where to start

- Git advocates

# Where to start

- Git advocates
- Proof of concept

# Where to start

- Git advocates
- Proof of concept
- Incremental adoption

# Where to start

- Git advocates
- Proof of concept
- Incremental adoption
- Training

# Where to start

- Git advocates
- Proof of concept
- Incremental adoption
- Training
- Don’t change workflow immediately

# Resources

- http://sixrevisions.com/resources/git-tutorials-beginners/
- http://www.atlassian.com/git/resources
- McCullough and Berglund - Mastering Git (O'Reilly video series)

# Thank you!

ryan.breidenbach@gmail.com

http://twitter.com/twoqubed

http://github.com/twoqubed