Git Training with Tinkertoys

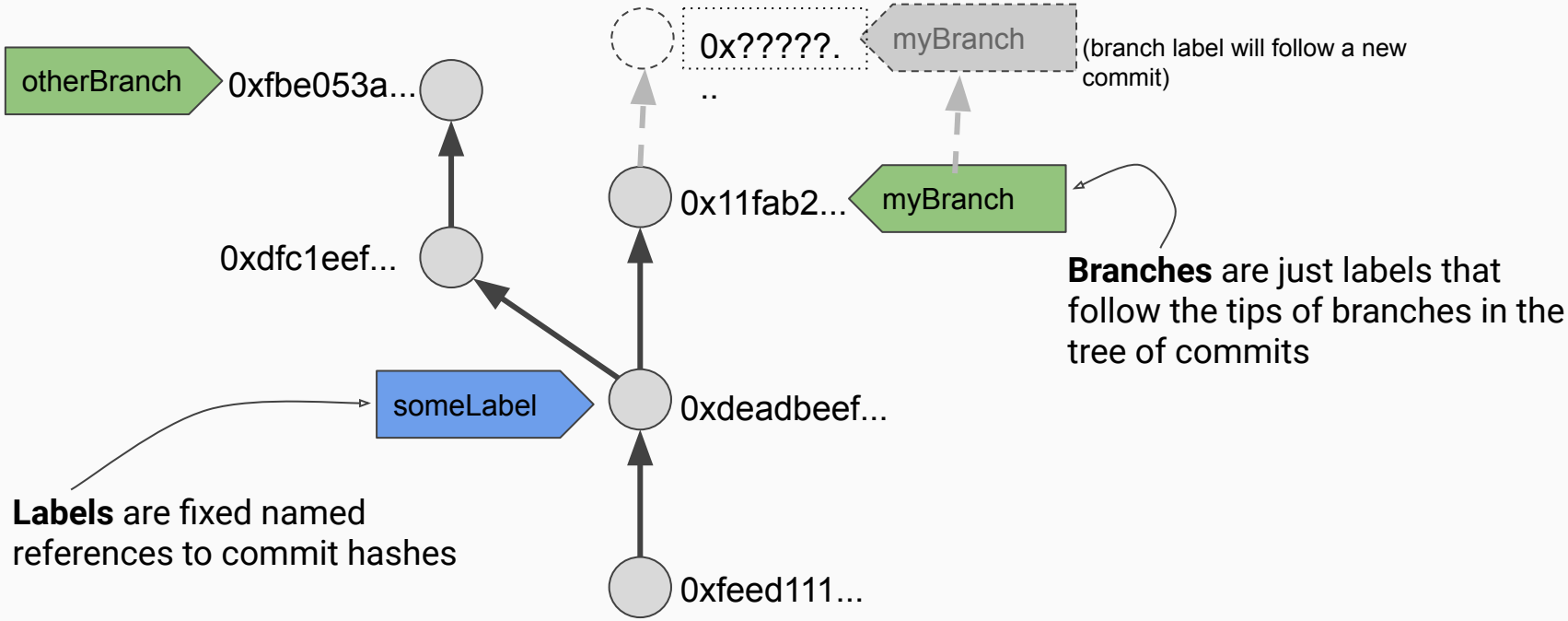
http://go/java+espresso-training/git

Version Management: Git (and TinkerToys)

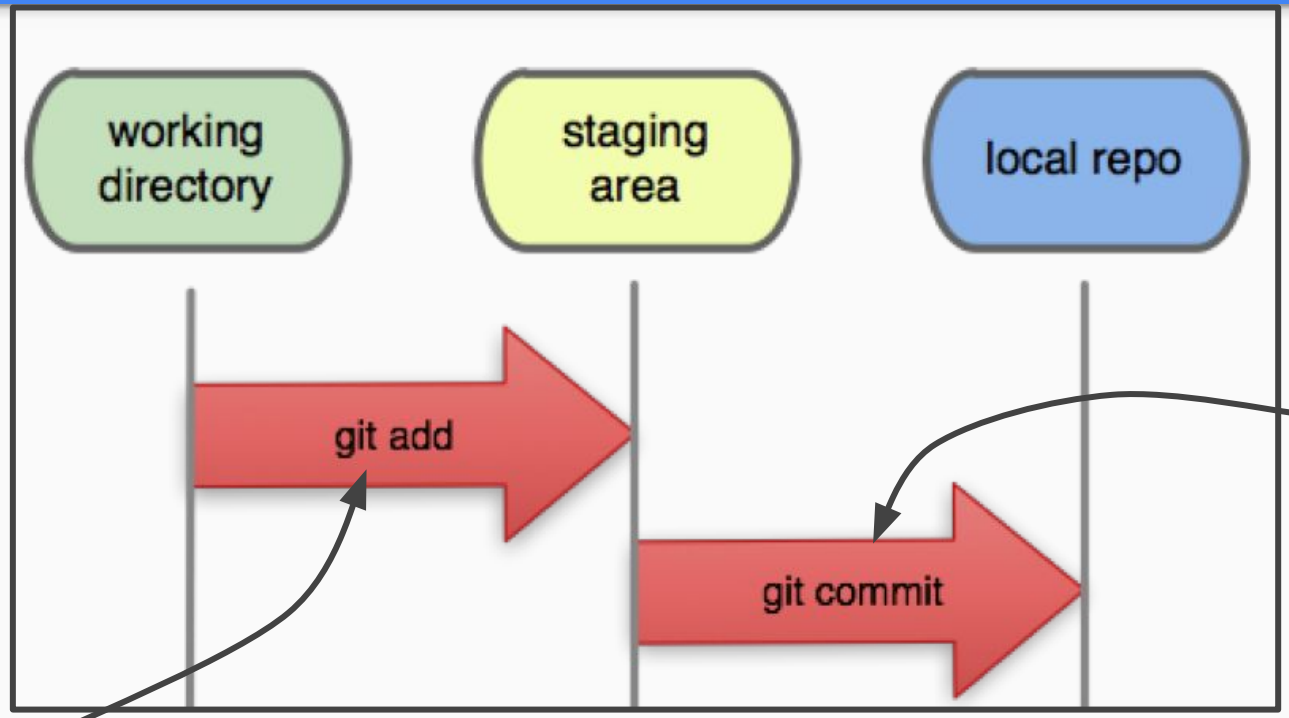
- Created by Linus Torvalds in 2005 to support Linux kernel development
- Developed because the existing distributed version control systems *sucked* in terms of performance, transparency and/or usability (I was there using: SVN, RCS, CVS, etc.. ; version control before Git kinda did suck).
- Git appears daunting and cryptic, but how it works is actually very simple (if you can understand TinkerToys, you can understand the innards of Git)
- I recommend watching the 2013 Linux Conf presentation: “Git For Ages 4 And Up”(https://www.youtube.com/watch?v=1ffBJ4sVUb4).

The Mechanics of Git

- Git stores **commits** indexed by *universally* unique* SHA1 **hash** values
- It tracks the parent-child relationship between **commits**
- Git *never* deletes a **commit** and you can always check-out a **commit** using its **hash**



The States of Git (local repository)



Commit message is added, the hash is generated and the commit is added to the repository tree

The '`git add`' transition groups changes to files in the working directory into **staged** "chunks"

Git Exercise / Tinker Toy Demo

We'll work through a series of **git** operations on a very simple git repository while I model our progress using ***Tinker Toys***.

Keep one terminal open and switch between the text editor and the command line. (otherwise git will confuse the editor)

Let's begin....

Git Tinker Toy Demo: Creating the Repository

1. Create new file: **`animals.txt`** and add a single line consisting of the word: **`Lions`**
2. Save the file and on the command-line type: **`git init`**

This should a message like: `Initialized empty Git repository in /Users/.../.git`

Git has created a hidden .git directory called `.git` which contains the files needed to maintain the repository

You can inspect the contents using: **`ls -al`**

Git Tinker Toy Demo:
Changes to the Working Directory

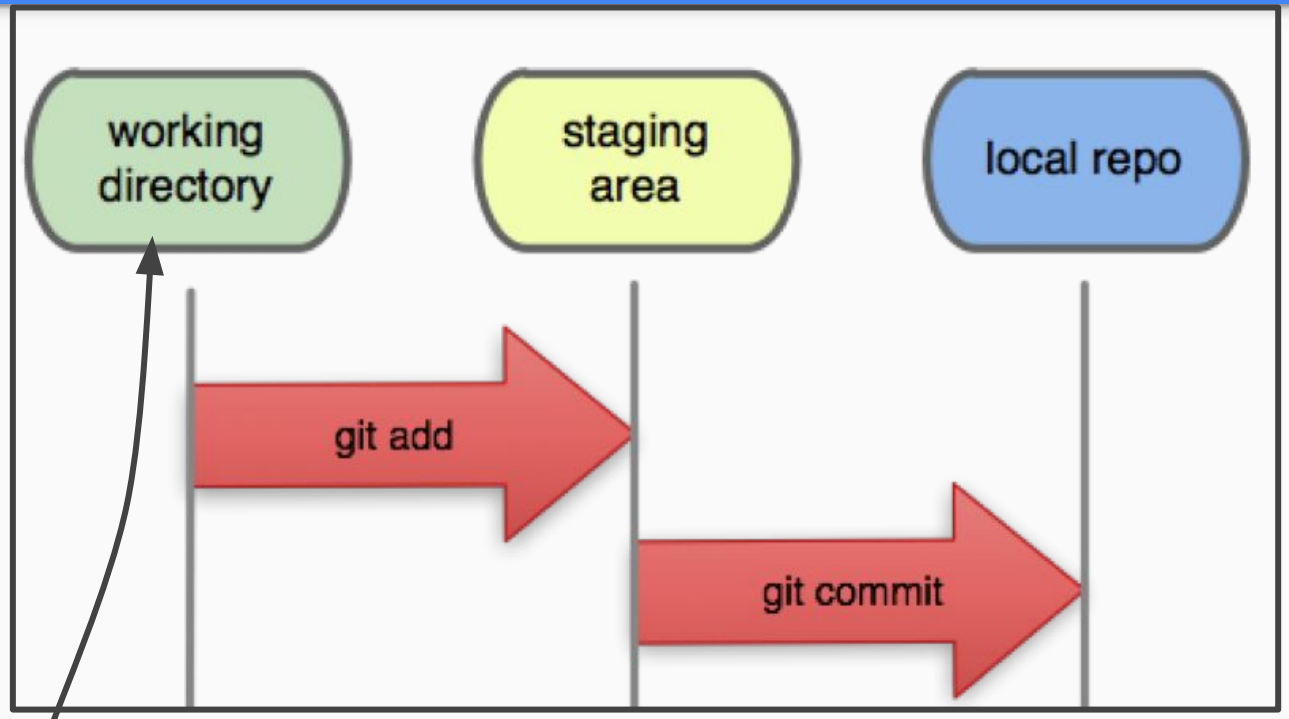
1. enter: **`git status`**

```
On branch master
No commits yet
Untracked files:
  (use "git add <file>..." to include in what will be
committed)

    animals.txt

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

`Animals.txt` has been modified, but the changes are not yet staged



`animals.txt` changes are here and ***untracked*** until we run **`"git add"`**

Git Tinker Toy Demo: Staging Changes

1. **`git add animals.txt`**
2. **`git status`**

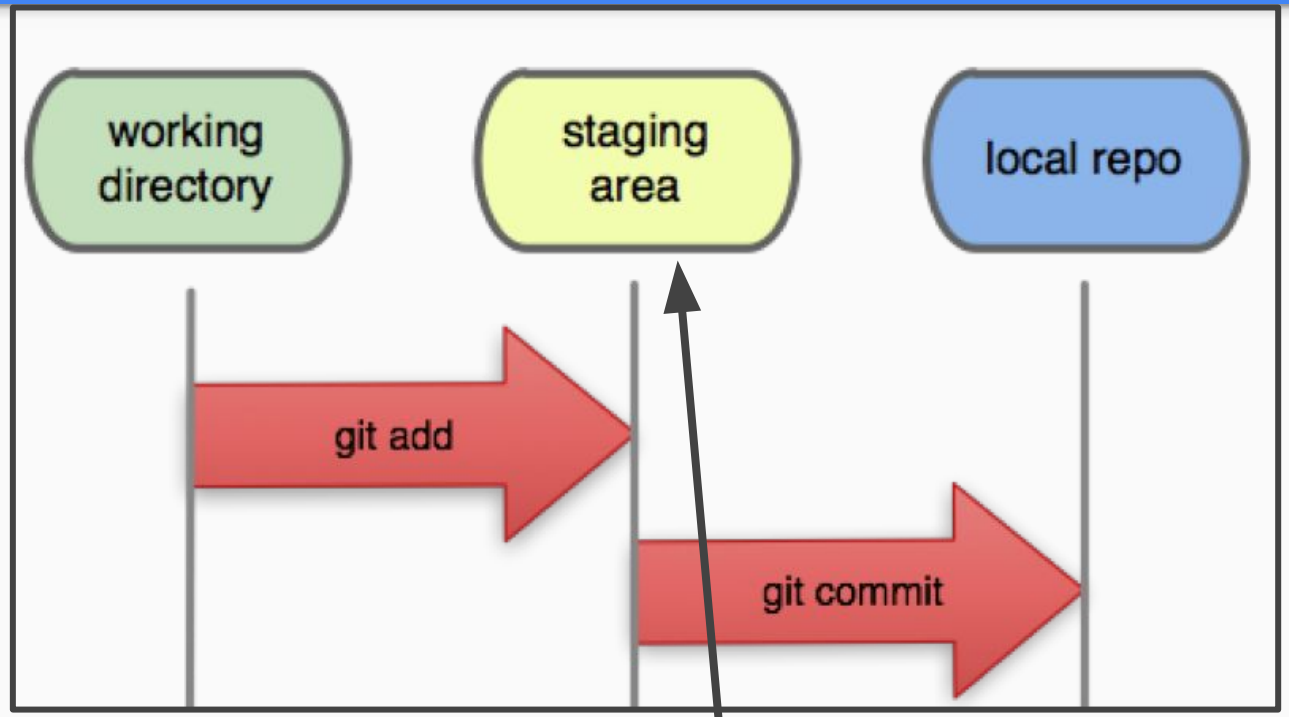
```
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   animals.txt
```

`Animals.txt`, the changes are staged



`animals.txt` changes are ***staged***, but not ***committed***

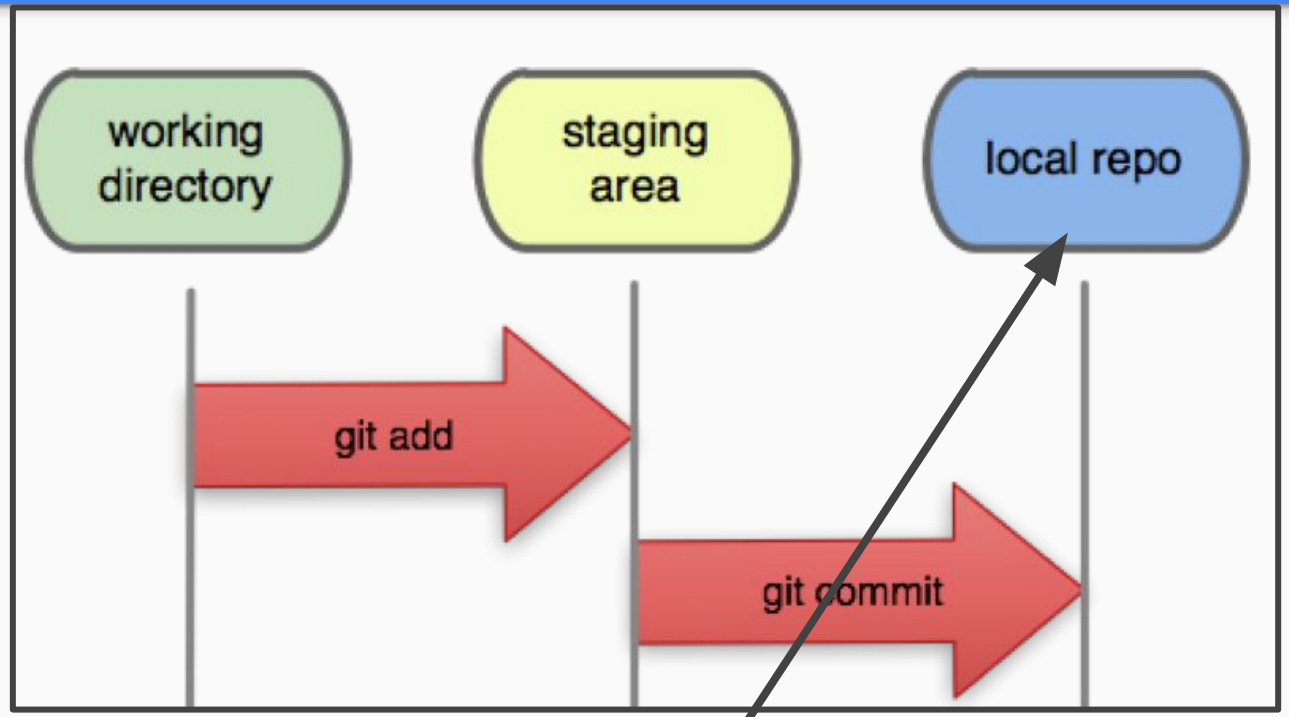
Git Tinker Toy Demo: Committing

1. **`git commit -m "Lions only"`**

```
[master (root-commit) 9a9811f] Lions only
 1 file changed, 1 insertion(+)
 create mode 100644 animals.txt
```

We now have a local repository with one branch(***master***) and one commit with the log message "**Lions only**" and a hash that begins with: **`9a9811f`**

`Animals.txt` after commit



`animals.txt` changes are now committed to the repository

Git Tinker Toy Demo: Examining the Repository

1. enter: **`git branch`**

```
* master
```

2. **`git log`**

```
commit 9a9811f838b42d92ff01f9668b8aa07ccc46d84a (HEAD -> master)
Author: Bjorn Chambless <bchambless@ebay.com>
Date:   Thu Feb 15 09:44:34 2018 -0800

    Lions only
```

Git Tinker Toy Demo: More Changes

1. enter: **`git status`**

```
On branch master
nothing to commit, working tree clean
```

2. Edit `animals.txt` and add another line: **`tigers`**
3. Save the file and enter: **`git status`**

You should see that `animals.txt` has more pre-staging changes

Git Tinker Toy Demo:
More Changes continued...

1. `git add .`

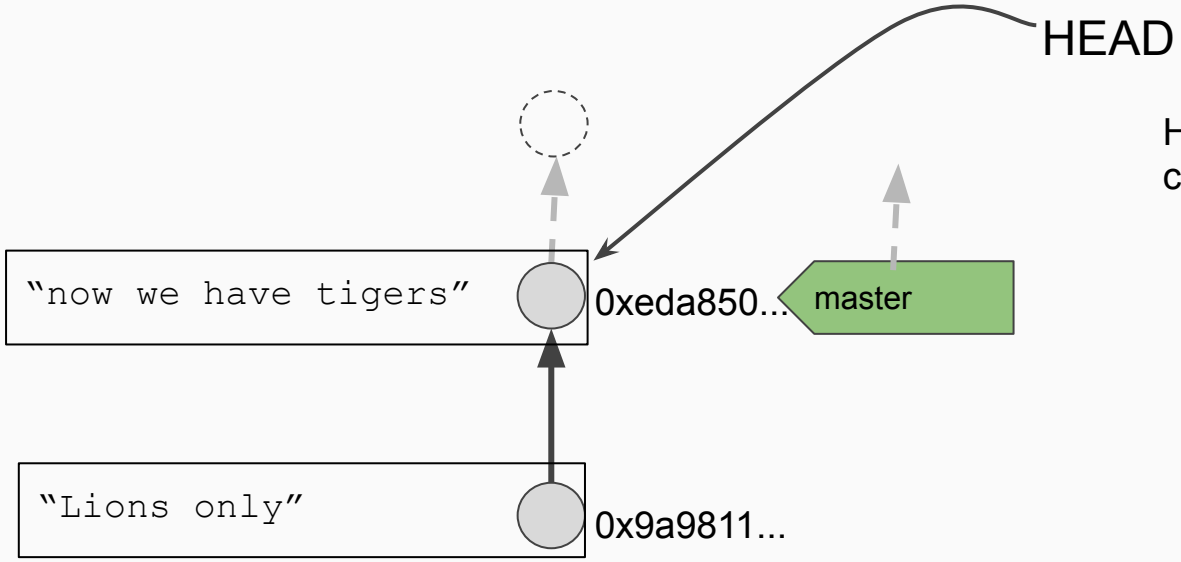
This adds all modified files in the current directory and all subdirectories to the staging area.

2. `git status`

You should see that `animals.txt` has more changes staged for commit

3. `git commit -m “now we have tigers”`
4. `git log --graph`

Current repository state (`git log --graph`)



HEAD is the commit that is currently checked-out

Git Tinker Toy Demo: Checking-Out Commits

1. Enter: **`cat animal.txt`** to verify what the current state is

Using the hash-value (the first few characters, in my case this is **`9a9811f838b4`**) of the first commit

2. **`git checkout 9a9811f838b4`**

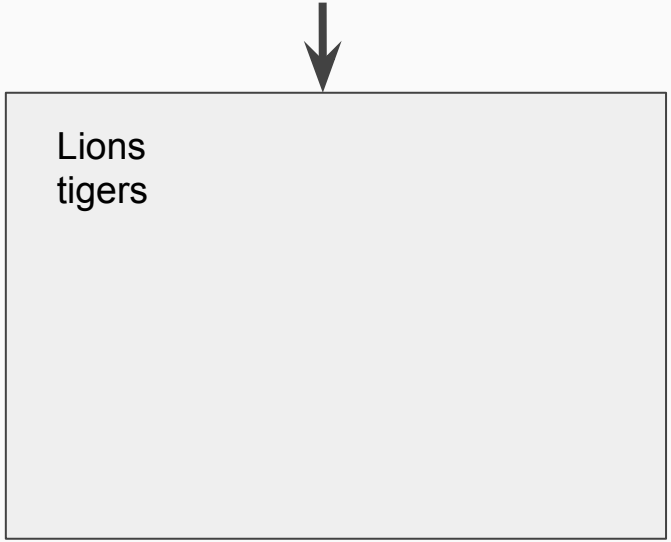
```
You are in 'detached HEAD' state…..
If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:
  git checkout -b <new-branch-name>
HEAD is now at 9a9811f... Lions only
```

`git checkout` switches *HEAD* to a different commit

`git checkout eda850f6`

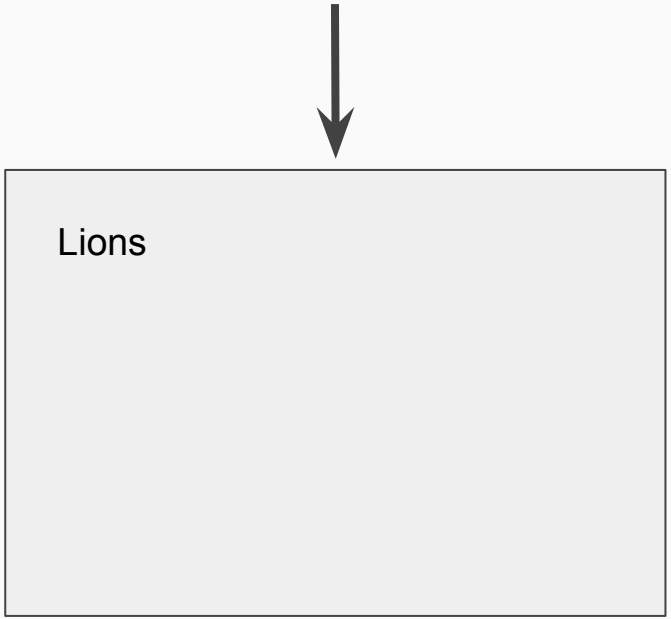
or

`git checkout master`



master (`eda850f676c`)

`git checkout 9a9811f`



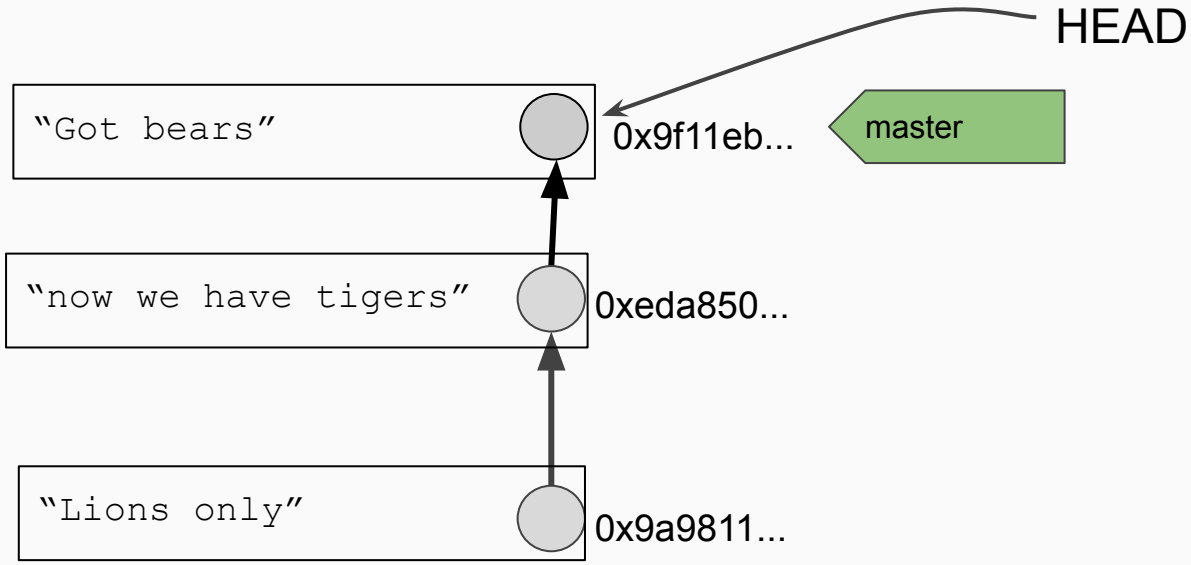
9a9811f838b42

Switch back and forth and confirm you've done so with: `cat animals.txt`

Git Tinker Toy Demo: Another commit to *master*

1. Switch `HEAD` back to the tip of `master` then...
2. Add a third line: `Bears`, stage the file, then commit the change (with an appropriate message) and display a graph of the repository

Current repository state (`git log --graph`)



Git Tinker Toy Demo: Creating a new branch

1. Now checkout the commit where you added “tigers” to animals.txt

```
git checkout [hashForTigers commit]
```

2. cat animals.txt
3. git checkout -b withZebras
4. git log --graph --all
5. git branch

The `withZebras` branch (`git log --graph --all`)

```
LM-PDX-11005160:tst bchambless$ git log --graph --all
* commit 9f11eb981e1c50face1de761609bc74a29d39a7b (master)
| Author: Bjorn Chambless <bchambless@ebay.com>
| Date:   Fri Feb 16 14:17:06 2018 -0800
|
|     got dem Bears
|
* commit eda850f676c3fd4dec37e05780a9af2508148512 (HEAD -> withZebras)
| Author: Bjorn Chambless <bchambless@ebay.com>
| Date:   Thu Feb 15 12:34:01 2018 -0800
|
|     now we have tigers
|
* commit 9a9811f838b42d92ff01f9668b8aa07ccc46d84a
  Author: Bjorn Chambless <bchambless@ebay.com>
  Date:   Thu Feb 15 09:44:34 2018 -0800

      Lions only
LM-PDX-11005160:tst bchambless$ git branch
  master
* withZebras
```

Git Tinker Toy Demo: Commits to a new branch

1. Now (while still on the `withZebras` branch), change `tigers` to `zebras` and commit the change(with an appropriate message) and display a graph of the repository

`git log --graph --all` (after committing zebra change)

```
* commit d091cab8262a6e5e38bfb576ccf9cb8ba014903c (HEAD -> withZebras)
| Author: Bjorn Chambless <bchambless@ebay.com>
| Date:   Tue Feb 20 09:47:57 2018 -0800
|
|     I see zebras
|
| * commit 9f11eb981e1c50face1de761609bc74a29d39a7b (master)
|/  Author: Bjorn Chambless <bchambless@ebay.com>
|   Date:   Fri Feb 16 14:17:06 2018 -0800
|
|       got dem Bears
|
* commit eda850f676c3fd4dec37e05780a9af2508148512
| Author: Bjorn Chambless <bchambless@ebay.com>
| Date:   Thu Feb 15 12:34:01 2018 -0800
|
|     now we have tigers
|
* commit 9a9811f838b42d92ff01f9668b8aa07ccc46d84a
  Author: Bjorn Chambless <bchambless@ebay.com>
  Date:   Thu Feb 15 09:44:34 2018 -0800

      Lions only
```

Git Tinker Toy Demo: Uh oh

1. Now type: `git branch -D withZebras`, then examine the repository tree

OH NOES, WE'VE LOST OUR IMPORTANT ZEBRA WORK!?!

Git Tinker Toy Demo: Coping with loss

We really haven't lost anything. Branches are just labels and commits are not deleted.
All we need is a hash....

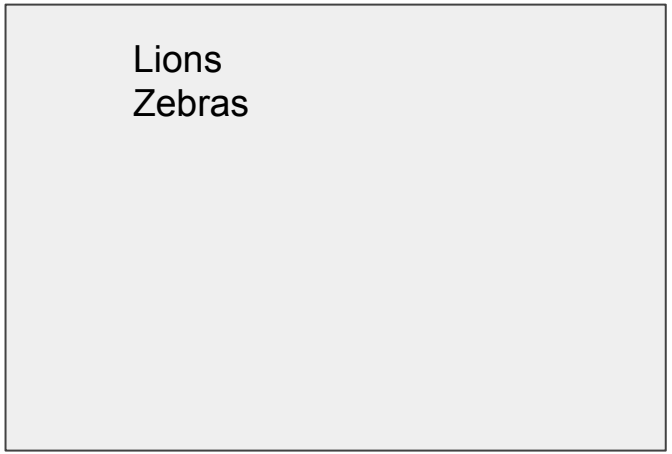
1. Scroll up in the terminal window and find the hash for the tip of the zebra branch, then: `git checkout [hash]`
2. Restore the branch name: `git checkout -b withZebras`
3. Take a look at the repository with: `git log --all --graph`

Git Tinker Toy Demo: Merging

Merging is not very interesting unless there is some conflict, and thankfully we have a conflict between our branches:

Master

withZebras



Git Tinker Toy Demo: Merge Conflicts

1. Switch to master then: `git merge withZebras`
2. `cat animals.txt`

```
Lions
<<<<< HEAD
tigers
Bears
=====
Zebras
>>>>>> withZebras
```

Edit the file so it has all the animals except “Bears” the stage and commit the file and examine the repository

Git States (with a remote repository)