A. Create, push and view references of a commit

- ✓ In your projectb working tree, create a file named fileA.txt containing the simple string "feature 1" (without the quotes).

```
repos$ cd project project
$ echo "feature 1" > fileA.txt
```

- ✓ Add fileA.txt to the staging area. (As a reminder, you can do this with the git add command.)

- ✓ Commit this file to the local repository. Specify a commit message of "add feature 1". (As a reminder, you use the git commit command with the -m option.)

- ✓ Push the commit to the remote repository.

```
projectb$ git push origin master
```

- ✓ View the commit on Bitbucket.

- ✓ At the command line, use git log --oneline to view your local repository's commit history. Notice that the SHA-1 values of the commit objects have been shortened. Also notice that there are labels/references associated with the commit(s). Labels starting with "origin" are related to the remote repository. We will discuss those more later.

Question 1 What local branch are you on? (answers at end of file)
Question 2 What does HEAD -> master mean?

- ✓ Execute the git show command, adding an argument of the latest commit. This will show the contents of the latest commit object, as well as some information on what has changed (we will discuss "diff" later). All of these commands should produce the same result:

```
$ git show (first four characters of SHA-1)
$ git show (complete SHA-1)
$ git show HEAD
$ git show master
```

- ✓ (If you have multiple commits) Execute git show HEAD~ to view the details of the parent of the latest commit. Execute git show HEAD^ to see the same details. As we learned in the video, ~ and ^ coincidently show the same results here, but in general they refer to different things. For example, ~2 and ^2 have different meanings. ~2, ~~ or ^^ refer to the grandparent. ^2 refers to the second parent in a merge commit.

Congratulations, you have created, pushed and viewed the references of a commit.

B. Tag the commit

- ✓ Execute git tag to view the tags in your local repository. There should be no tags.
- ✓ Let's say that you want to "permanently" attach a version label to the commit that implements feature 1. Use the git tag command for this. Create an annotated tag by specifying the -a option. Use the -m option to specify a tag message of "includes feature 1". Since the most recent commit implements feature 1, you do not need to specify a commit, because the command defaults to HEAD. Name the tag "v0.1".

```
$ git tag -a -m "includes feature 1" v0.1
```

- ✓ Execute git tag again to view the tags. You should see your "v0.1" tag.
- ✓ Execute git show v0.1 . You should see your tag information, as well as information on its associated commit object.
- ✓ Push the tag to the remote repository.

```
$ git push origin v0.1
```

- ✓ View the commit on Bitbucket. Notice that it has the v0.1 tag associated with it.
- ✓ (If you have multiple commits) Tag the parent of the current commit with a "temp" tag. You can use the HEAD~ reference to refer to this commit.

```
$ git tag -a -m "just a temp tag" temp HEAD~
```

- ✓ Delete this tag using the -d option.

```
$ git tag -d temp
```

- ✓ You will not use the projectb repository in future labs. You can delete the remote and local repositories. Delete the local repository by deleting the projectb directory. Delete the remote repository in Bitbucket by selecting Settings , Delete Repository and Delete .

Congratulations, you have created a tag and pushed it to the remote repository.

Question 1 What local branch are you on? master

Question 2 What does HEAD -> master mean? HEAD is a reference to the current commit. The content of that commit is in your working tree. master is the name of the branch, and this commit is the tip of the branch, so this is the master branch label. The arrow represents that the HEAD reference points to the master branch label, which points to the SHA-1 of the commit.