*Version 2019-Spring-1.0, Revised 13 February 2019*

Activity 4:

Undoing Your Changes With Git

Sometimes you make changes to your files that you don’t want to keep. How you undo them when working with Git varies by what state your file is in.

## Content Learning Objectives

*After completing this activity, students should be able to:*

* Undo changes to files in the working state.
* Undo changes to files in the staged state.
* Undo changes to files in the committed state.

## Process Skill Goals

*During the activity, students should make progress toward:*

* Carefully reading Git command messages for understanding. (Information Processing)

## Team Roles

*Decide what role each of you will play for today. Choose a role that you have not played before, or recently. The goal should be to have all team members rotate through the roles on a regular basis to become comfortable with all the roles. If you have only three people, one should have two roles. If you have five people, two may share the same role. Record role assignments here.*

|  |  |
| --- | --- |
| Manager |  |
| Presenter |  |
| Recorder |  |
| Reflector |  |

# Model 1: Undoing a change you made to a file.

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10** | $ **ls**  A.java    $ **git log**  commit db967767f9ab675dba21e08c5d20627b34cbe133 (HEAD -> master, origin/master, origin/HEAD)  Author: Karl R. Wurst <karl@w-sts.com>  Date: Fri Jan 18 14:25:44 2019 -0500  Add Feature X.  $ **git status**  On branch master  Nothing to commit, working tree clean  $ **cat A.java**  This is change #1.  $ **edit A.java**    $ **git status**  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: A.java  no changes added to commit (use "git add" and/or "git commit -a")  $ **cat A.java**  This is modification #1.  This is modification #2.  $ **git checkout -- A.java**  $ **git status**  On branch master  Nothing to commit, working tree clean  $ **cat A.java**  This is change #1. |

## Questions (10 min)

1. For command #1 through command #4, which of the three states (Working, Staging, Committed) is A.java in?
2. What are the contents of A.java before command #5?
3. For command 6 through command #7, which of the three states is A.java in?
4. What are the contents of A.java between command #5 and command #8?
5. What was the effect of the editing in command #5?
6. After listing the contents of A.java in command #7, you decide that you didn’t want to make those changes. What is the effect ot command #8? Be sure your explanation refers to at least one of the three states.
7. If you couldn’t remember command #8, how could you figure out what command to give (without resorting to checking the Internet)?
8. Describe a situation where you would find command #8 useful - and probably not possible to accomplish in any other way?

# Model 2: Remove changes from a commit to be made.

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11**  **12** | $ **git log**  commit 63b66cf74632ab3f2ebabb333e71d13938d7f2d4  Author: Karl R. Wurst <karl@w-sts.com>  Date: Fri Jan 18 16:02:12 2019 -0500  Add Feature Y  commit db967767f9ab675dba21e08c5d20627b34cbe133 (HEAD -> master, origin/master, origin/HEAD)  Author: Karl R. Wurst <karl@w-sts.com>  Date: Fri Jan 18 14:25:44 2019 -0500  Add Feature X.  **$ git status**  On branch master  Nothing to commit, working tree clean  $ **cat A.java**  This is change #1  $ **edit A.java**  $ **cat A.java**  This is modification #1.  This is modification #2.  $ **edit B.java**    $ **git status**  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: A.java  modified: B.java  no changes added to commit (use "git add" and/or "git commit -a")  $ **git add .**  $ **git status**  git status  On branch master  Changes to be committed:  (use "git reset HEAD <file>..." to unstage)  modified: A.java  modified: B.java  $ **git reset HEAD A.java**  Unstaged changes after reset:  M A.java  $ **git status**  On branch master  Changes to be committed:  (use "git reset HEAD <file>..." to unstage)  modified: B.java  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: A.java  $ **cat A.java**  This is change #1 |

## Questions (12 min)

1. What is the effect of command #8?
2. After issuing command #9, you decide that you do not want to commit your changes to A.java. What is the effect of command #10? Be sure your explanation refers to at least two of the three states.
3. What does HEAD refer to in command #10? How did you figure that out?
4. Explain why command #8 is a dangerous command. Or, at least, why you should always follow command #8 with command #9.
5. If you couldn’t remember command #10, how could you figure out what command to give (without resorting to checking the Internet)?
6. Describe a situation where you would find command #10 useful?

# Model 3: Removing changes that have already been committed.

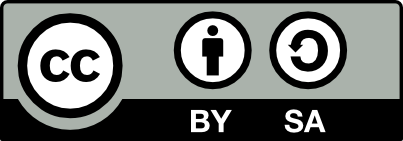
|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11**  **12** | $ **ls**  A.java    $ **git log**  commit db967767f9ab675dba21e08c5d20627b34cbe133 (HEAD -> master, origin/master, origin/HEAD)  Author: Karl R. Wurst <karl@w-sts.com>  Date: Fri Jan 18 14:25:44 2019 -0500  Add Feature X.  $ **git status**  On branch master  Nothing to commit, working tree clean  $ **cat A.java**  This is change #1.  $ **edit A.java**  $ **cat A.java**  This is modification #1.  This is modification #2.  $ **git add A.java**  $ **git commit -m”Add Feature Y.”**  [master (root-commit) 63b66cf] Add Feature Y.  1 files changed, 2 insertions(+)  create mode 100644 A.java  $ **git log**  commit 63b66cf74632ab3f2ebabb333e71d13938d7f2d4 (HEAD -> master, origin/master, origin/HEAD)  Author: Karl R. Wurst <karl@w-sts.com>  Date: Fri Jan 18 16:02:12 2019 -0500  Add Feature Y.  commit db967767f9ab675dba21e08c5d20627b34cbe133  Author: Karl R. Wurst <karl@w-sts.com>  Date: Fri Jan 18 14:25:44 2019 -0500  Add Feature X.  $ **git revert HEAD**  [master 685f4ab] Revert "First commit"  1 files changed, 2 deletions(-)  delete mode 100644 A.java  $ **git log**  commit 685f4abfccda64d68145bae6a6123bf1e8a88021 (HEAD -> master)  Author: Karl R. Wurst <karl@w-sts.com>  Date: Fri Jan 18 16:19:36 2019 -0500  Revert "Add Feature Y."  This reverts commit 63b66cf74632ab3f2ebabb333e71d13938d7f2d4.  commit 63b66cf74632ab3f2ebabb333e71d13938d7f2d4  Author: Karl R. Wurst <karl@w-sts.com>  Date: Fri Jan 18 16:02:12 2019 -0500  Add Feature Y.  commit db967767f9ab675dba21e08c5d20627b34cbe133  Author: Karl R. Wurst <karl@w-sts.com>  Date: Fri Jan 18 14:25:44 2019 -0500  Add Feature X.  $ **cat A.java**  This is change #1. |

## Questions (15 min)

1. What are the contents of A.java before command #5?
2. What are the contents of A.java after command #5?
3. Explain what happens in command #7 through command #9.

After issuing command #9, you decide that you should not have committed your changes to A.java.

1. What are the contents of A.java after command #10?
2. What is the effect of command #10? Be sure your explanation refers to at least one of the three states, and to the log.
3. What is the default commit message from command #10?
4. How does **revert** relate to **HEAD** in command #10? How did you figure that out?
5. Are the changes you made to A.java in command #5 really undone? Explain your answer. Why do you think git works this way?
6. Describe a situation where you would find command #10 useful?

Copyright © 2019 Karl R. Wurst. This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.