 **Isilon Git Workshop**

**Prerequisites**

**Setup Git**

Git is installed (<https://wiki.west.isilon.com/index.php/Git>)

Add SSH Key (<https://wiki.west.isilon.com/index.php/Git>)

**Workshop Setup**

Setup global user name and email

*git config –global user name “[your name]”*

*git config –global user email “[your email address]”*

Note: This is a per machine setting, not per repository

**Isilon Workflow (Prep)**

1. Navigate to git@github.west.isilon.com:IsiEngTraining/GitTraining.git
2. Fork a new personal repository from the team repository
   1. Click on the  icon in the upper right corner
   2. Chose your account as the destination of the fork
   3. Navigate to your account/OneFS repo
   4. In the lower right URL field select SSH as the type of clone URL
   5. Click on the  icon to copy the URl to the clipboard
3. Clone
   1. Open a command shell
   2. Navigate to the directory you typically store source

*git clone [paste SSH URL]*

**Lab Exercises**

**Blob Exercise**

1. From the directory you stash source code, create a new local repository

*git init NewRepo*

1. Create a blob object

*echo “Hello” | git hash-object –w --stdin*

*Note:*

* + *–w* Writes the object into the object database (.git/objects)

1. Verify that the object is in the object folder

*ls .git/objects/[first 2 digits from the hash]/[remaining hash digits]*

1. Verify that the object type is a blob

*git cat-file –t [first 8 digits of hash]*

1. Verify that the object value is “Hello”

*git cat-file –p [first 8 digits of hash]*

**Tree Exercise**

1. Add the blob to the index, adding file permissions and a file name

*git update-index --add --cacheinfo 100644 [blob hash] hello.txt*

1. Write the tree you just created to the object store

*git write-tree*

1. Verify the object type is a tree

*git cat-file –t [ short hash of tree object from step 2 ]*

1. Verify the object type, parent, author and committer

*git cat-file –p [short hash of tree object from step 2]*

1. Create a second blob

*Echo “World” | git hash-object –w --stdin*

1. Add the second blob to the index, adding file permissions and a file name

*git update-index --add --cacheinfo 100644 [full hash of blob from step 3] world.txt*

1. Write the tree you just created to the object store

*git write-tree*

1. Create a commit object referencing the first tree from step 2

*git commit-tree [short hash from step 6] –m “[commit message]”*

1. Create a commit object referencing the tree from step. This time add the commit object from step 7 as the parent commit

*git commit-tree [short hash of tree from step 6] –p [hash of commit object from step 7] –m “[commit message]”*

1. Verify object type, parent, author, committer of second tree commit

*git cat-file -p [short hash of commit object from step 8]*

1. Create a reference for the master branch to the commit from step 8

*echo [short hash of commit object from step 8]” > .git/refs/heads/master*

1. Run and think about the results

*git status*

1. Create the files that the blobs describe

*echo “Hello” > hello.txt*

*echo “World” > world.txt*

**Tag Exercise**

1. Create a tag

*git tag -a -m “Annotate tag example” tag-1 master*

1. Show tag reference

*git show-ref tag-1*

1. Show branch reference

*git show-ref master*

1. Verify object type, tag reference, tagger

*git cat-file -p [first 8 digits of tag-1 hash]*

**Files Exercise**

1. Create 5 files with 2 files in subdirectory
2. Stage a file

*git add [file name]*

1. Commit the file

*git commit [file name] -m “[commit message]”*

1. Check the file status

*git status*

1. Stage a second file

*git add [file name]*

1. Edit the newly staged file
2. Stage the file

*git add [file name]*

1. Commit the file

*git commit [file name] -m “[commit message]”*

1. Stage a third file

*git add [file name]*

1. Move/Rename a file

*git mv [source file] [sub directory/destination file]*

1. Stage the moved/renamed file

*git add [moved/renamed file]*

1. Check the file status

*git status*

1. Commit the file

*git commit [file name] -m “[commit message]”*

**Branches**

1. Create a new branch

*git branch[new branch name]*

1. Checkout the branch

*git checkout [branch name]*

1. Edit file
2. Stage file

*git add [filename]*

1. Commit the file

*git commit [file name]*

1. Checkout branch you wish to merge TO

*git checkout [destination branch]*

1. Merge the branch to the master branch

*git merge [target branch name]*

Note: At this point you should have 2 branches- **Master** and your **newly created branch**. This exercise is a local branch merge so the branch name being merged to should be **Master**.

1. Back out of this repository

*cd ..*

1. Run branch conflict script

*./git-create-conflict.sh*

1. Change directory into ***i-merge-conflict*** repository
2. Merge the update branch to master

*git checkout master*

*git merge update*

1. Merge fix branch

*git merge fix*

1. Resolve merge conflict

Manually edit, add, commit

1. Merge topic branch

*git merge topic*

1. Resolve merge conflict

Manually edit, add, commit

1. Look at merge history with log command

*git log –-oneline --graph*

1. Verify the changes to the file

*cat FileA*

1. Back out of repository and delete

*rm –rf i-merge-conflict*)

**Remote Work**

1. Re-use repository we’ve been working in for previous exercises (NewRepo)
2. Create a branch with your desktop id (i.e. mprice)

*git branch [desktop]*

*git checkout [desktop]*

1. Set up remote repository

*Git remote add training git@github.west.isilon.com:IsiEngTraining/GitTraining.git*

1. Fetch

*git fetch training*

1. Merge remote master branch into [desktop id] branch

*git merge training master*

1. Push into github

*git push –set-upstream training [desktop id]*

1. Pull Request

Go into github and create a pull request for your branch into master

1. Instructor will merge one pull request, then show others in conflict, unable to merge

**Manipulating History**

1. Run git-prep-interactive-rebase.sh
2. Rebase interactive BR\_TWO to combine commits into one commit.

*cd i-rebase*

*git log --oneline*

*git checkout BR\_TWO*

*git log –-oneline*

*git rebase –i [short hash of last commit common to two log messages]*

*edit to manipulate commits*

1. Change the commit message for the commit you just created

*git log --oneline*

*git rebase foo*

use ‘r’ to change commit message in editor

1. Rebase BR\_ONE against master

*git checkout BR\_ONE*

*git rebase master*

1. Merge BR\_ONE into master

*git checkout master*

*git merge BR\_ONE*

1. Rebase BR\_TWO against master

*git checkout BR\_TWO*

*git rebase master*

Resolve conflict

*git rebase --continue*

1. merge BR\_TWO into master

*git checkout master*

*git merge BR\_TWO*

**Isilon Workflow**

1. Navigate to git@github.west.isilon.com:IsiEngTraining/GitTraining.git
2. Fork a new personal repository from the team repository
   1. Click on the  icon in the upper right corner
   2. Chose your account as the destination of the fork
   3. Navigate to your account/OneFS repo
   4. Select branch “training”
      1. Create a branch in your account with your username
3. In the lower right URL field select SSH as the type of clone URL
4. Click on the  icon to copy the URl to the clipboard
5. Cloning
   1. Open a command shell
   2. Navigate to the directory you want the repository to be cloned to
   3. Type:

*git clone [paste SSH URL]*

[git@github.west.isilon.com:[your-git-username]/[team-branch].git]

Wait (OneFS can take time to clone)

1. Updates

Edit a .c file, add, commit, rebase against “training\_next” and push the results to github (*origin branch [desktop id]*)

1. Review Board

From command prompt, use rbt to post to Review Board

1. Pull Requests
   1. Create a pull request from your new branch back to “training” branch
   2. Instructor will approve and merge pull request
2. Team Leads extra work
   1. Reviewing pull requests for quality
   2. Approving pull requests