Sri Harsha Chennavajjala (SC9V9) | COMP-SCI 5555 | December 14, 2015

GIT Command-Line

Basics of git



# Introduction

Git is a widely used version control system for software development. It is a distributed revision control system with an emphasis on speed, data integrity, and support for distributed, non-linear workflows. As with most other distributed version control systems, and unlike most client–server systems, every Git working directory is a full-fledged repository with complete history and full version-tracking capabilities, independent of network access or a central server.

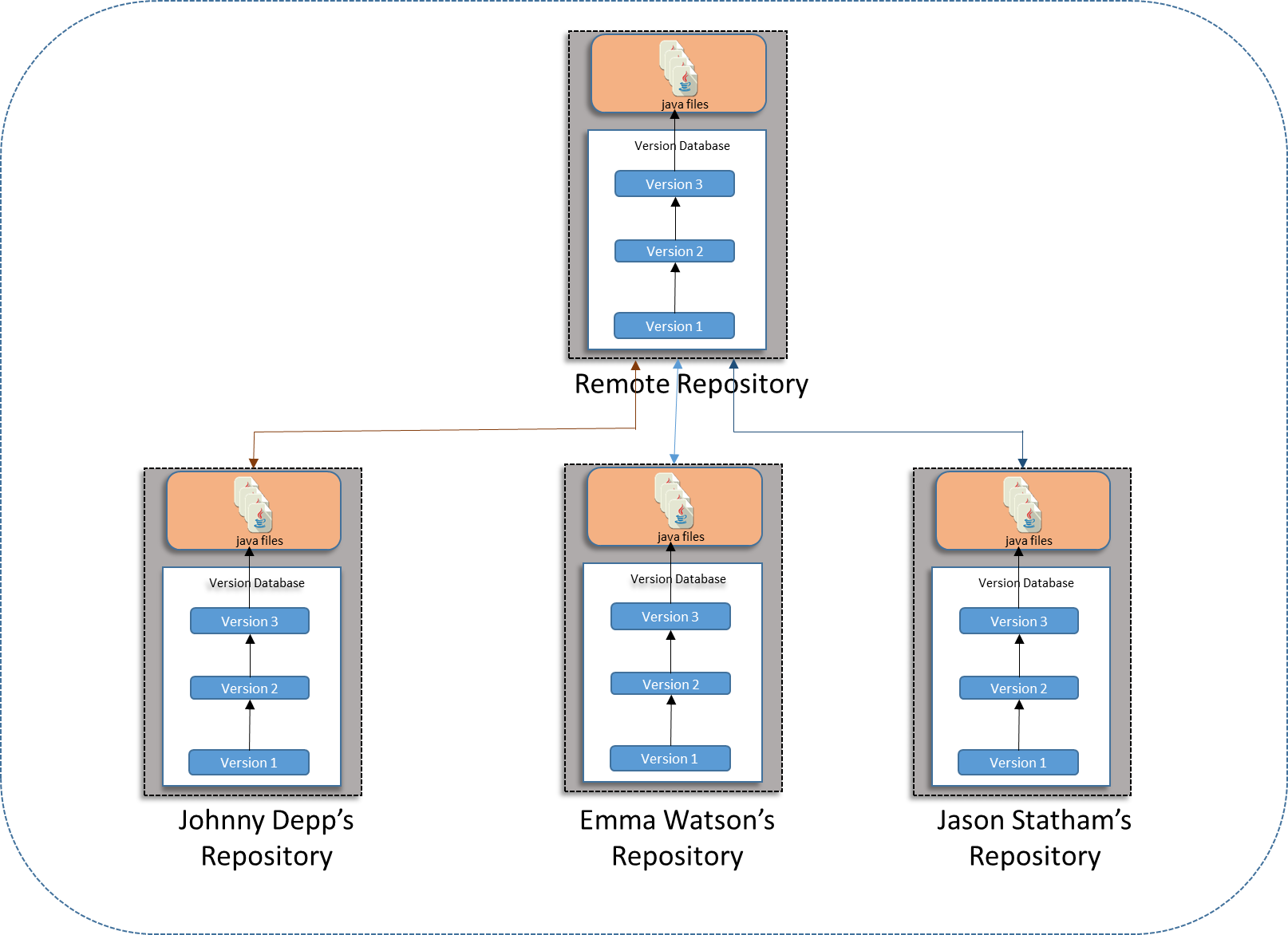
In this report, we will briefly go through the basic commands of Git shell.

# Environment set up

Let us consider we are working on the Tetris application and there are three contributors who accesses and updates the application code. Let us assume the users are Johnny Depp, Emma Watson and Jason Statham.

In this environment, I’ve used centralized workflow, i.e. all users will share same public repository.

The architecture is as shown below.



First, open Git shell for each user and set up the configuration parameters like username, email and so on…

**Local repositories for the users are:**

|  |  |
| --- | --- |
| Username | Repository Directory |
| Johnny Depp | /c/Users/meets/Assignment9/JohnnyDepp |
| Emma Watson | /c/Users/meets/Assignment9/EmmaWatson |
| Jason Statham | /c/Users/meets/Assignment9/JasonStatham |

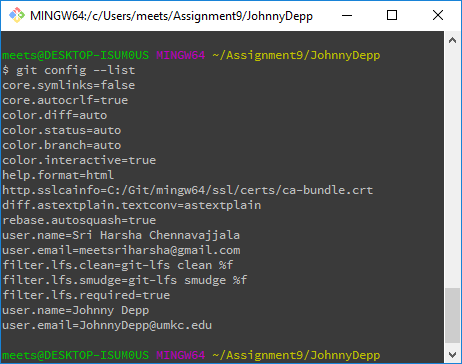
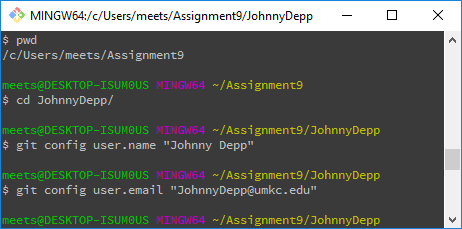
# User Configuration

**Commands for Johnny Depp:**

**git** config user.name "Johnny Depp"

**git** config user.email "JohnnyDepp@umkc.edu"

**git** config **–-**list

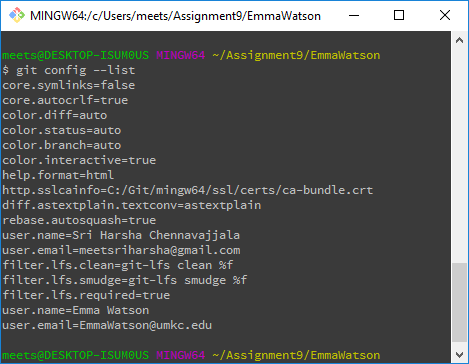
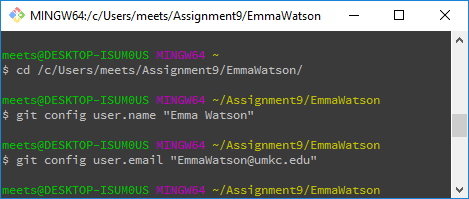


**Commands for Emma Watson:**

**git** config user.name "Emma Watson"

**git** config user.email "EmmaWatson@umkc.edu"

**git** config **--**list

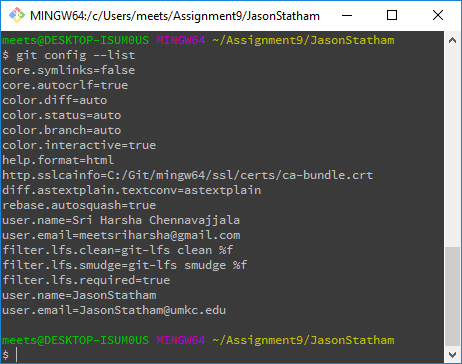
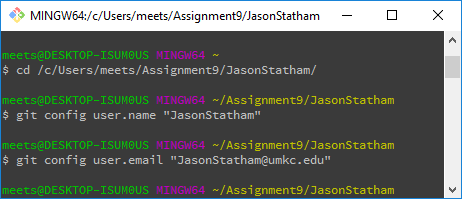


**Commands for Jason Statham:**

**git** config user.name "JasonStatham"

**git** config user.email "JasonStatham@umkc.edu"

**git** config **--**list



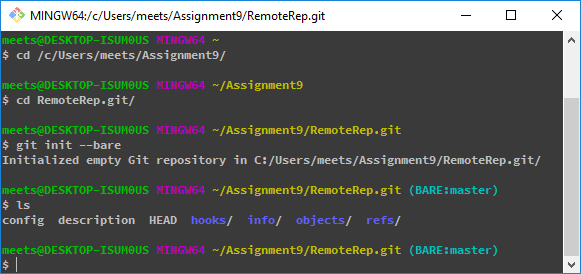
# Repository Creation

Now copy Tetris game files to local repositories of each user.

Next, create a bare repository “RemoteRep.git” in the directory “/c/Users/meets/Assignment9/”. All the users will share their changes through this repository.

**Command to initialize bare repository:**

**git** init **--**bare



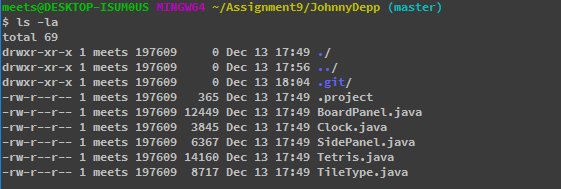
Now, go to each user’s directory and initialize the local repositories of each users using the below command.

**git** init

A Sample screenshot is as shown below.



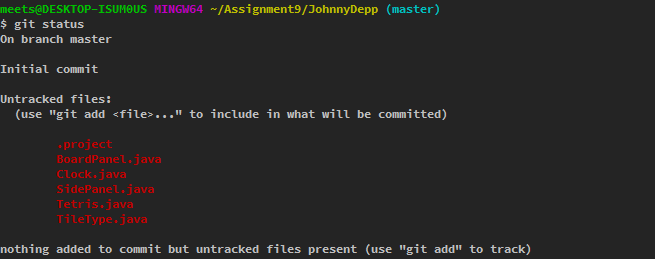
Each user’s directory hierarchy will be as shown below (e.g., Johnny Depp’s directory).



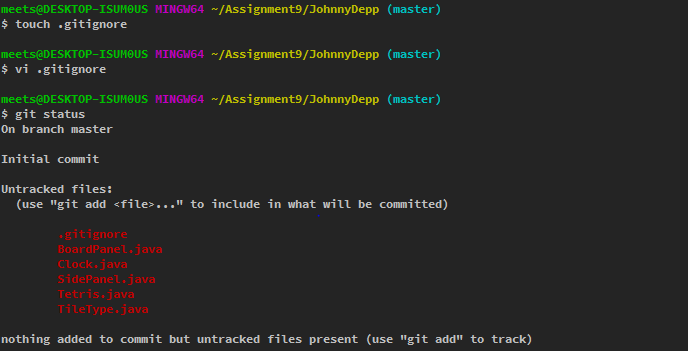
# Git Ignore

Here, I’ve added a new file “.project” which is a unique personal configuration file for each user. This file’s content is different for each user. So, we should not push this file to remote repository. To do so, we will create a new file “.gitignore”. This file contains list of all the files which should not be pushed to remote repository. Use “**git** status” command to get the current status of the repository.

**Before configuring git ignore:**



**After configuring git ignore:**



Each user has his own repository and a global public shared repository. Next, we need to track all the files in each user’s repository. To do this, perform the below steps in each user’s repository directory.

**git** add -A

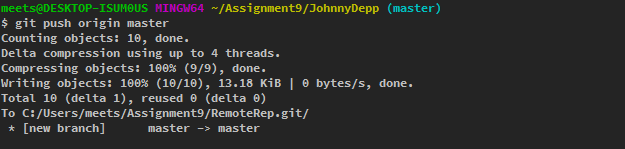
**git** commit **-**m "Initial import for Johnny Depp"

**git** status

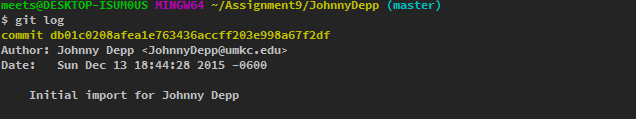
**git** remote add origin **/**c**/**Users**/**meets**/**Assignment9**/**RemoteRep.git**/**

Johnny Depp pushed his local repository copy to remote repository using the below command.

**git** push origin master



We can see the log using “git log” command.



# Pushing changes to Remote Repository

Now, Emma Watson want to edit the “startGame” method of “Tetris.java” file and also she wants to push his changes to the remote repository. She can edit and save the file in any text editor. Here I consider vi editor.

**vi** Tetris.java

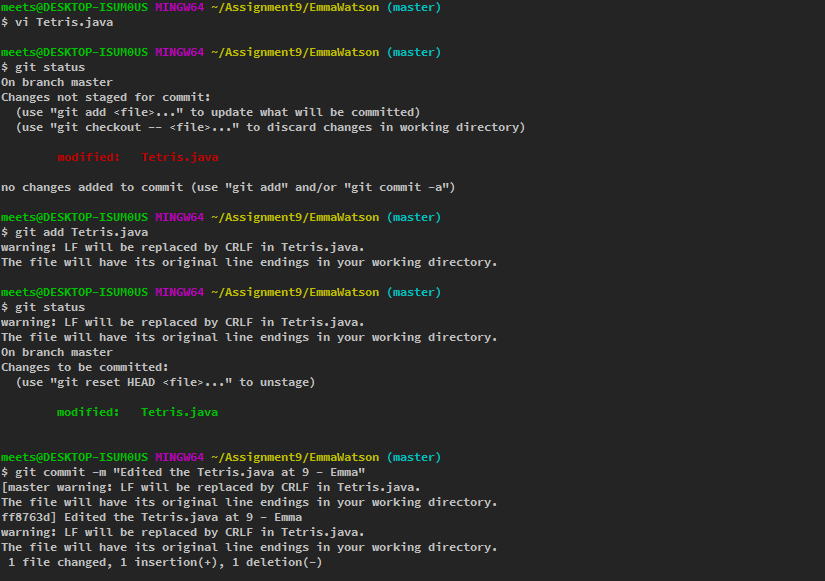
After making the changes, she has to perform the below steps.

1. Add the modified files to staging

git add Tetris.java

1. Commit the changes to local repository

git commit -m "Edited the Tetris.java at 9 - Emma"

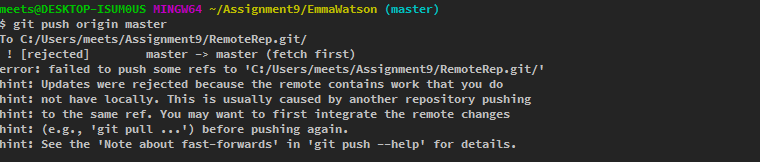


At this point, Emma Watson’s changes are successfully added to her local repository. Next, she has to push her changes to remote repository so that other users can pull her changes. Command is

**git** push origin master

# Merge Conflict Resolution

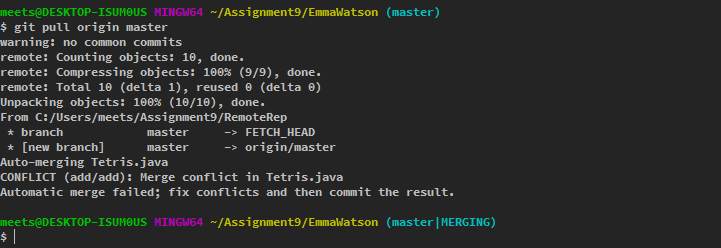
If Emma Watson doesn’t have a latest copy of the remote repository, then she will get the below error message saying “You may want to first integrate the remote changes”. In this case, Emma Watson has to first pull the remote repository to local before she can successfully push her changes.



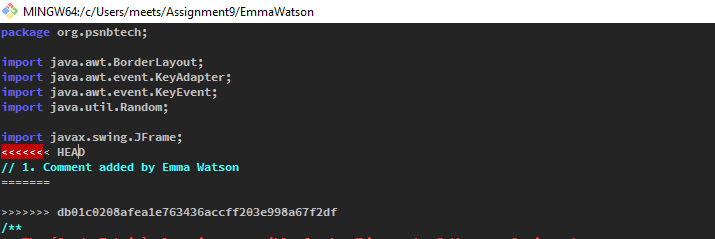
**Command to pull the latest copy of remote repository:**

**git** pull origin master

In this scenario, as Emma Watson has a different local copy, she will get conflicts during the pull process. This is because, Emma changed the “Tetris.java” file. So, git will throw a merge error asking the user to manually merge the file changes.



Now, Emma has to open the “Tetris.java” file and has to merge the changes. Git highlights the conflicted lines as shown below.



After modified the file, Emma has to stage the file and has to commit the changes as shown below.

**Commands:**

1. Merge the file by opening it in vi editor

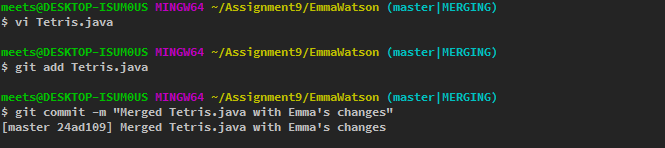
**vi** Tetris.java

1. After chages, add Tetris.java to staging area

**git** add Tetris.java

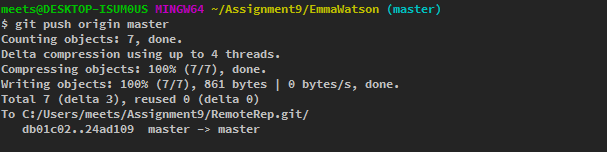
1. Commit the changes to local repository

**git** commit **-**m "Merged Tetris.java with Emma's changes"

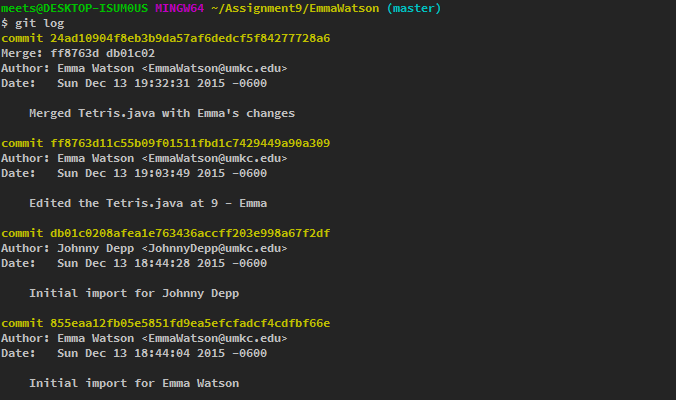


Now, Emma can successfully push her changes to remote repository

**git** push origin master



Now we can check the log using “git log”

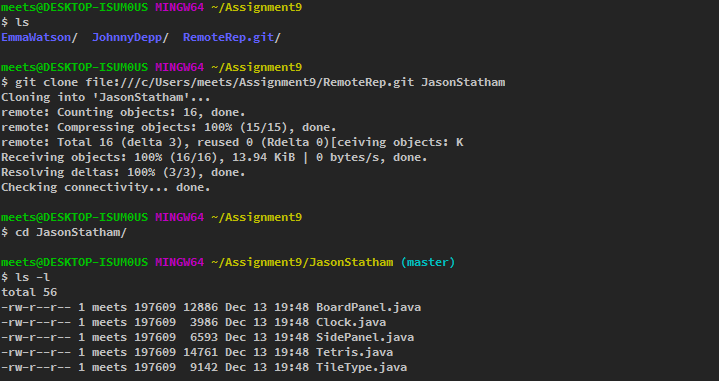


# Cloning a Repository

Now, suppose Jason Statham doesn’t have any files in his repository. So, he has to clone the remote repository to local repository using the below command.

**git** clone file:///c/Users/meets/Assignment9/RemoteRep.git JasonStatham

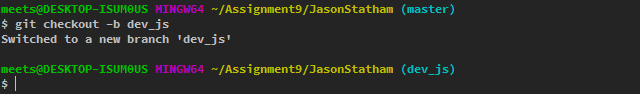
Local repository JasonStatham will be automatically created and all the files from remote repository will be copied to local repository as shown below:



# Branching

Now Jason Statham wants to make some changes to “Clock.java” file in a separate branch so that the master branch will remain intact. So he creates a new branch for development purpose with naming convention <env>\_<user>. For e.g. dev\_js.

**git** checkout -b dev\_js



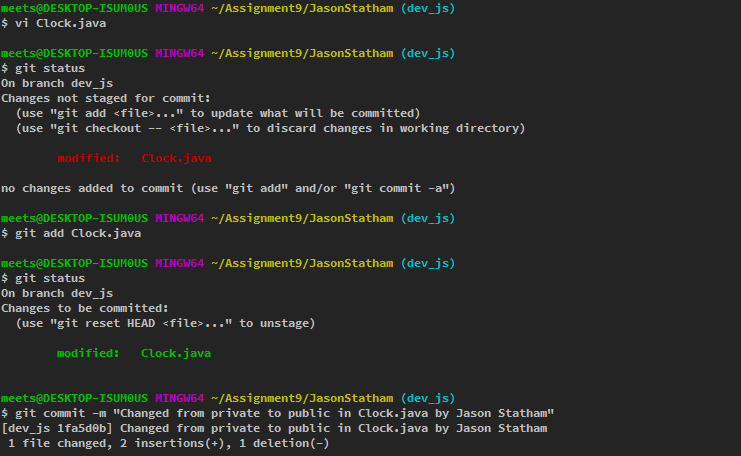
Jason Statham has to repeat the same steps as Emma did for “Tetris.java” file. (Edit, Stage and Commit).

**Commands:**

**vi** Clock.java

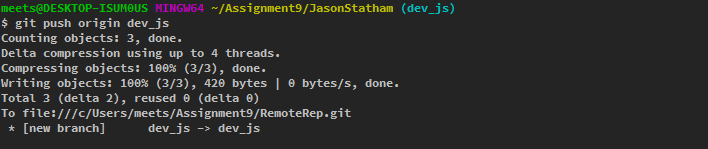
**git** add Clock.java

**git** commit **-**m "Changed from private to public in Clock.java by Jason Statham"



In real life project management scenario, master branch of repository will work as “Production copy” and all other branches will represent different environment setup copies (e.g. DEV, ACC, TST). In this scenario, Jason Statham has made changes to his DEV branch (dev\_js). So, he will push his changes to his own branch on the remote repository (for e.g. origin/dev\_js).

**git** push origin dev\_js

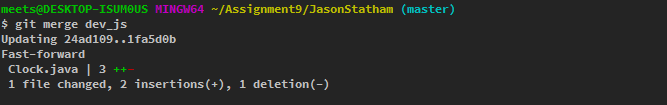


Note: The remote branch will be automatically created.

Jason can change to master branch using “**git** checkout master” command.

If Jason is satisfied with the new changes in dev\_js branch, he can merge these changes with master branch using the below command.

**git** merge dev\_js

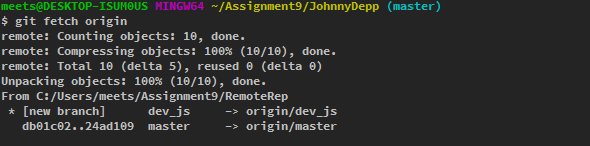


# Fetch a Repository

Now, Johnny Depp wants to work on Jason Statham’s branch “dev\_js”. So, first, he has to get the latest remote repository structure to his local repository. He needs to perform the below steps.

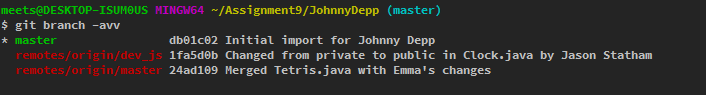
1. Fetch the latest repository information.

**git** fetch origin



1. Check whether the branch information is updated or not.

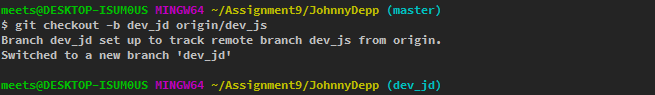
**git** branch **-**avv



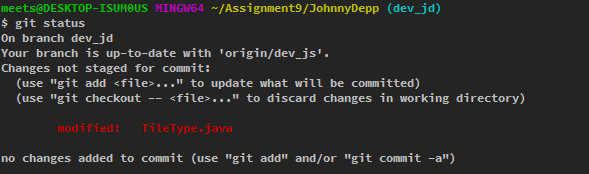
Note: Asterisk in the above figure denotes the current branch.

1. Create a local branch “dev\_jd” pointing to “origin/dev\_js”.

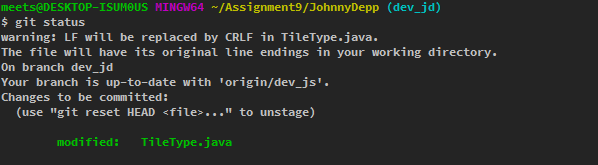
**git** checkout -b dev\_jd origin**/**dev\_js



Now, Johnny Depp modified the TileType.java file and checked the status of branch using “**git** status”.

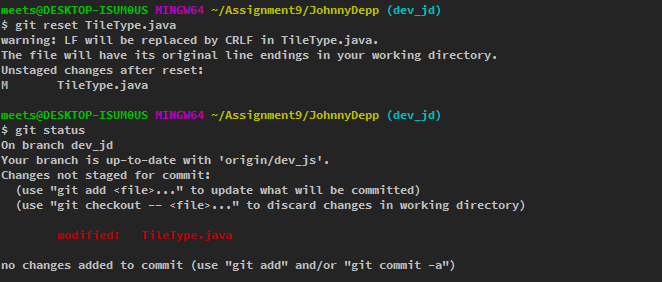


As Johnny did not add the modified “TileType.java” file to staging area, it is in “not staged area” (red color). Now, Johnny decided to add this file to staging area. He can do so using the command “**git** add TileType.java”. Now the file will be moved to staged area as shown below.



Later, Johnny changed his mind and decided to remove “TileType.java” from staging area. He can do so, by using the below command:

**git** reset TileType.java

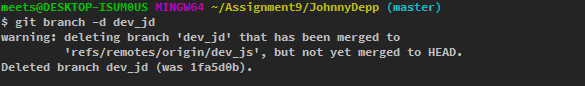


# Deleting a Branch

Once a user is finished with his work in the branches, he may want to delete the branch. In this case, Johnny Depp can delete the branch “dev\_jd” using:

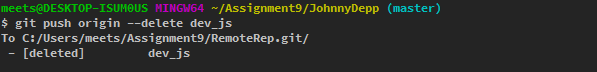
* On local repository:

**git** branch -d dev\_jd



* On remote repository:

**git** push origin **--**delete dev\_js



# Tracking a Branch

If Emma wants to track remote branch dev\_js, she can do so using the below command.

**git** fetch origin

**git** branch **--**track dev\_ew origin**/**dev\_js

First command make sure that the local repository is up to date with remote repository. Second command creates a branch “dev\_ew” which tracks remote branch “dev\_js”.

Git tracks the remote branches by default. So we need not to mention the “--track” option.