



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.

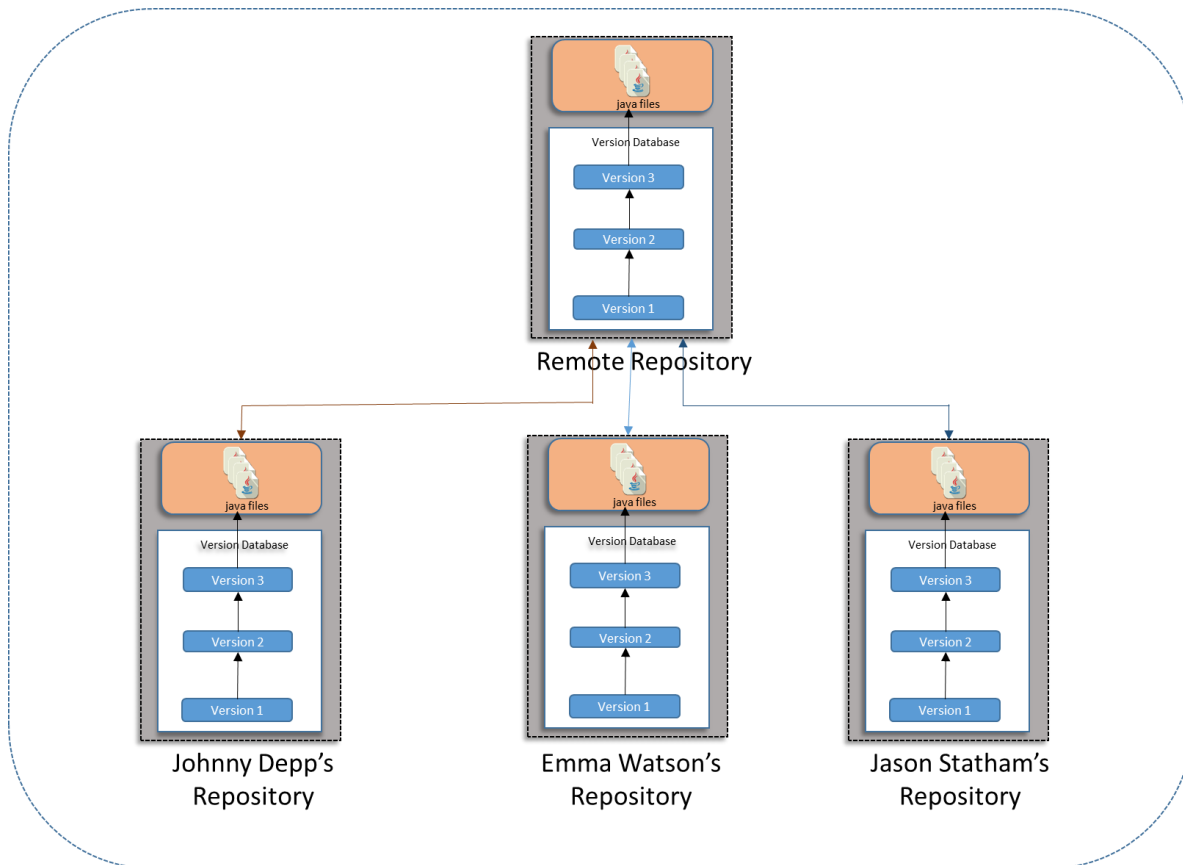
In this report, we will briefly go through the basic commands of Git shell.

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...

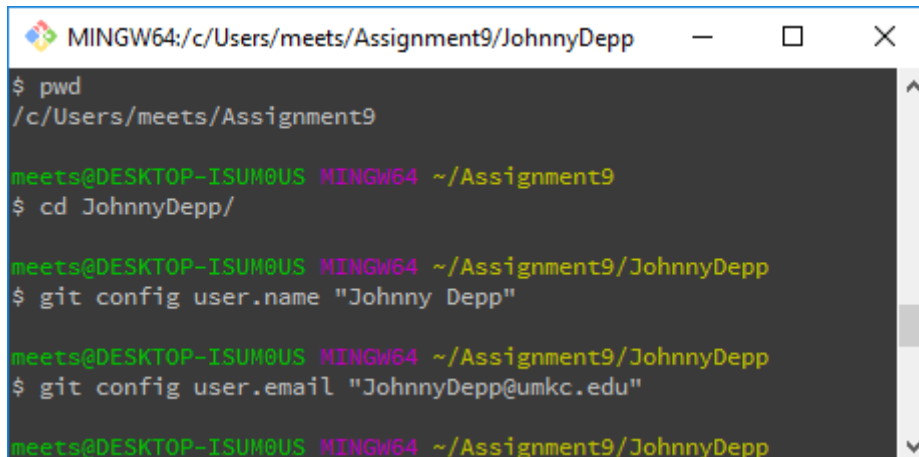
Local repositories for the users are:

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

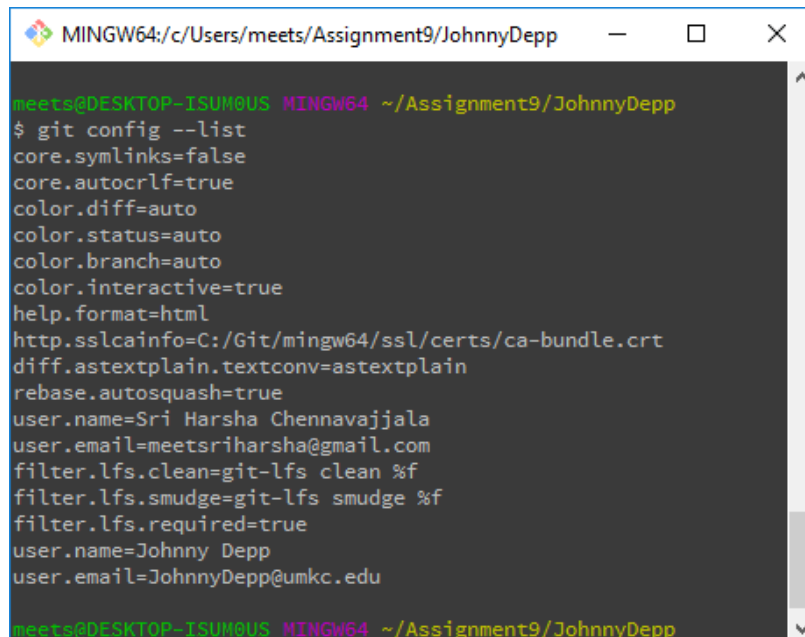
User Configuration

Commands for Johnny Depp:

```
git config user.name "Johnny Depp"  
git config user.email "JohnnyDepp@umkc.edu"  
git config --list
```



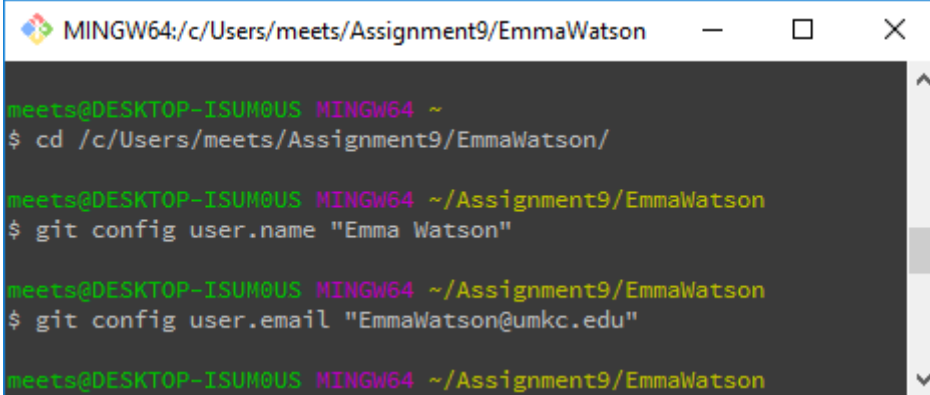
```
MINGW64:/c/Users/meets/Assignment9/JohnnyDepp  
$ pwd  
/c/Users/meets/Assignment9  
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9  
$ cd JohnnyDepp/  
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JohnnyDepp  
$ git config user.name "Johnny Depp"  
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JohnnyDepp  
$ git config user.email "JohnnyDepp@umkc.edu"  
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JohnnyDepp
```



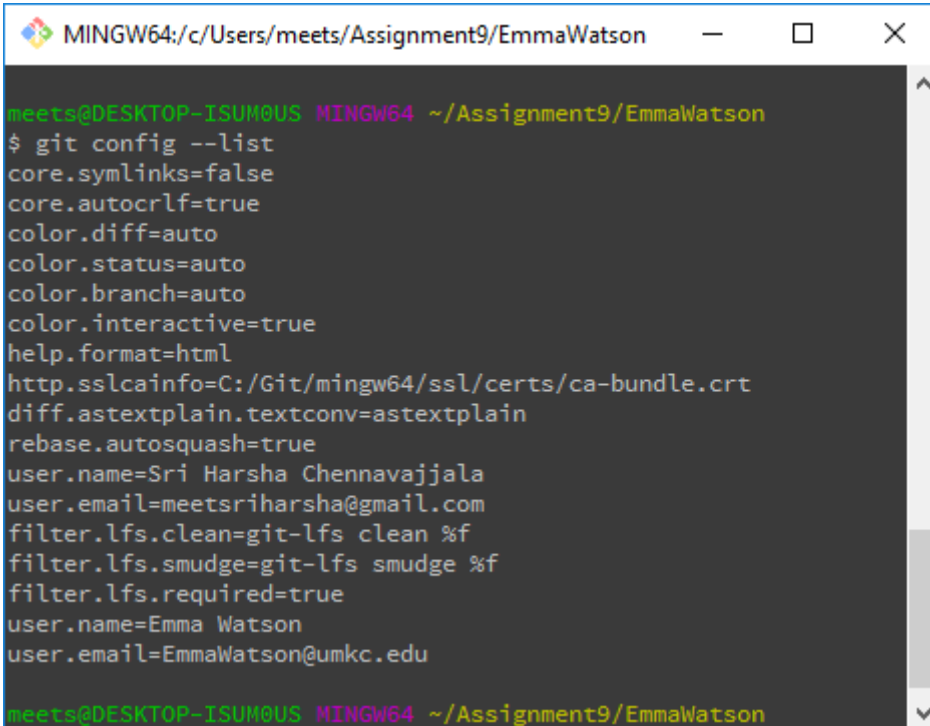
```
MINGW64:/c/Users/meets/Assignment9/JohnnyDepp  
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JohnnyDepp  
$ git config --list  
core.symlinks=false  
core.autocrlf=true  
color.diff=auto  
color.status=auto  
color.branch=auto  
color.interactive=true  
help.format=html  
http.sslcainfo=C:/Git/mingw64/ssl/certs/ca-bundle.crt  
diff.astextplain.textconv=astextplain  
rebase.autosquash=true  
user.name=Sri Harsha Chennavajjala  
user.email=meetsriharsha@gmail.com  
filter.lfs.clean=git-lfs clean %f  
filter.lfs.smudge=git-lfs smudge %f  
filter.lfs.required=true  
user.name=Johnny Depp  
user.email=JohnnyDepp@umkc.edu  
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JohnnyDepp
```

Commands for Emma Watson:

```
git config user.name "Emma Watson"  
git config user.email "EmmaWatson@umkc.edu"  
git config --list
```

A terminal window titled "MINGW64:/c/Users/meets/Assignment9/EmmaWatson" with standard window controls. The terminal shows the following commands and output:

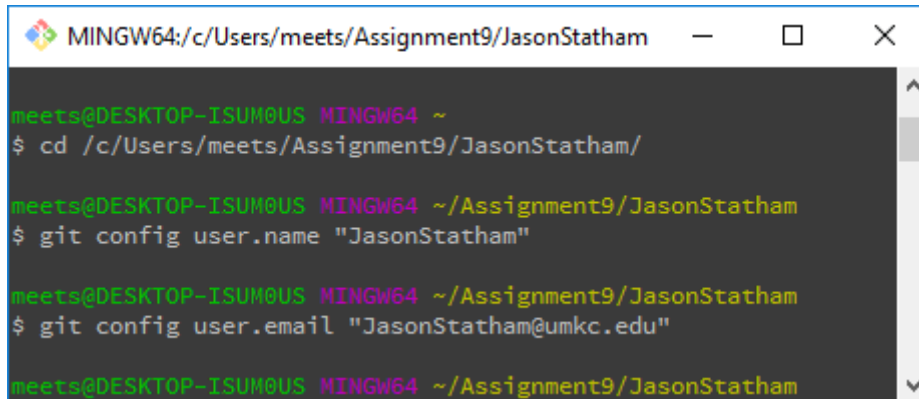
```
meets@DESKTOP-ISUM0US MINGW64 ~  
$ cd /c/Users/meets/Assignment9/EmmaWatson/  
  
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/EmmaWatson  
$ git config user.name "Emma Watson"  
  
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/EmmaWatson  
$ git config user.email "EmmaWatson@umkc.edu"  
  
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/EmmaWatson
```

A terminal window titled "MINGW64:/c/Users/meets/Assignment9/EmmaWatson" with standard window controls. The terminal shows the output of the command 'git config --list':

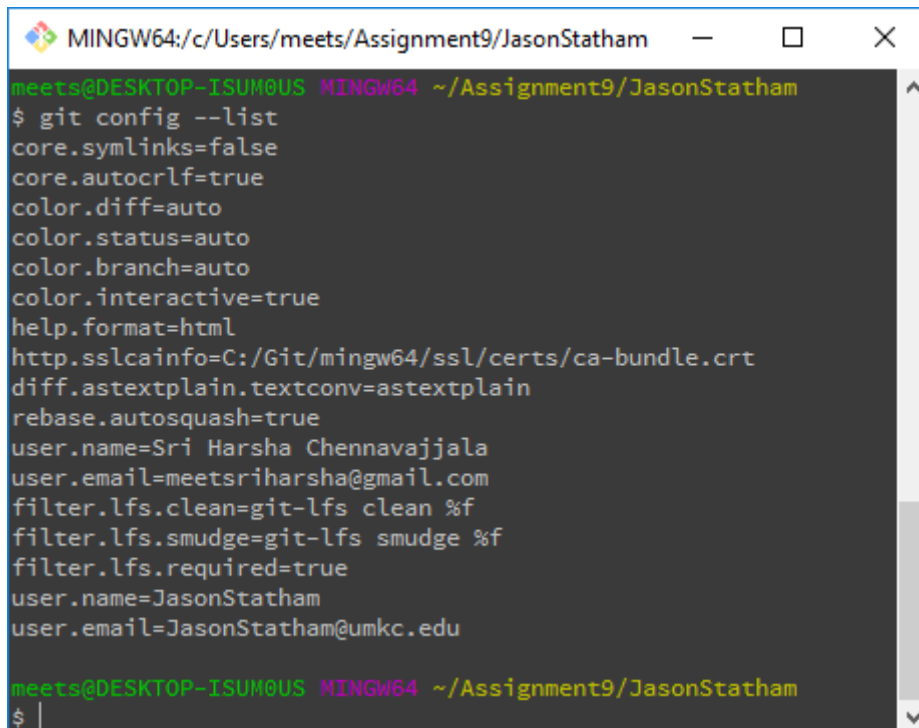
```
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/EmmaWatson  
$ git config --list  
core.symlinks=false  
core.autocrlf=true  
color.diff=auto  
color.status=auto  
color.branch=auto  
color.interactive=true  
help.format=html  
http.sslcainfo=C:/Git/mingw64/ssl/certs/ca-bundle.crt  
diff.astextplain.textconv=astextplain  
rebase.autosquash=true  
user.name=Sri Harsha Chennavajjala  
user.email=meetsriharsha@gmail.com  
filter.lfs.clean=git-lfs clean %f  
filter.lfs.smudge=git-lfs smudge %f  
filter.lfs.required=true  
user.name=Emma Watson  
user.email=EmmaWatson@umkc.edu  
  
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/EmmaWatson
```

Commands for Jason Statham:

```
git config user.name "JasonStatham"  
git config user.email "JasonStatham@umkc.edu"  
git config --list
```

A terminal window titled 'MINGW64:/c/Users/meets/Assignment9/JasonStatham'. The prompt is 'meets@DESKTOP-ISUM0US MINGW64 ~'. The user enters 'cd /c/Users/meets/Assignment9/JasonStatham/' and the prompt changes to 'meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JasonStatham'. Then, the user enters 'git config user.name "JasonStatham"' and 'git config user.email "JasonStatham@umkc.edu"'.

```
MINGW64:/c/Users/meets/Assignment9/JasonStatham  
meets@DESKTOP-ISUM0US MINGW64 ~  
$ cd /c/Users/meets/Assignment9/JasonStatham/  
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JasonStatham  
$ git config user.name "JasonStatham"  
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JasonStatham  
$ git config user.email "JasonStatham@umkc.edu"  
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JasonStatham
```

A terminal window titled 'MINGW64:/c/Users/meets/Assignment9/JasonStatham'. The prompt is 'meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JasonStatham'. The user enters 'git config --list', which outputs a list of git configuration settings. The prompt then changes to 'meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JasonStatham' and the user enters '\$ |'.

```
MINGW64:/c/Users/meets/Assignment9/JasonStatham  
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JasonStatham  
$ git config --list  
core.symlinks=false  
core.autocrlf=true  
color.diff=auto  
color.status=auto  
color.branch=auto  
color.interactive=true  
help.format=html  
http.sslcainfo=C:/Git/mingw64/ssl/certs/ca-bundle.crt  
diff.astextplain.textconv=astextplain  
rebase.autosquash=true  
user.name=Sri Harsha Chennavajjala  
user.email=meetsriharsha@gmail.com  
filter.lfs.clean=git-lfs clean %f  
filter.lfs.smudge=git-lfs smudge %f  
filter.lfs.required=true  
user.name=JasonStatham  
user.email=JasonStatham@umkc.edu  
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JasonStatham  
$ |
```

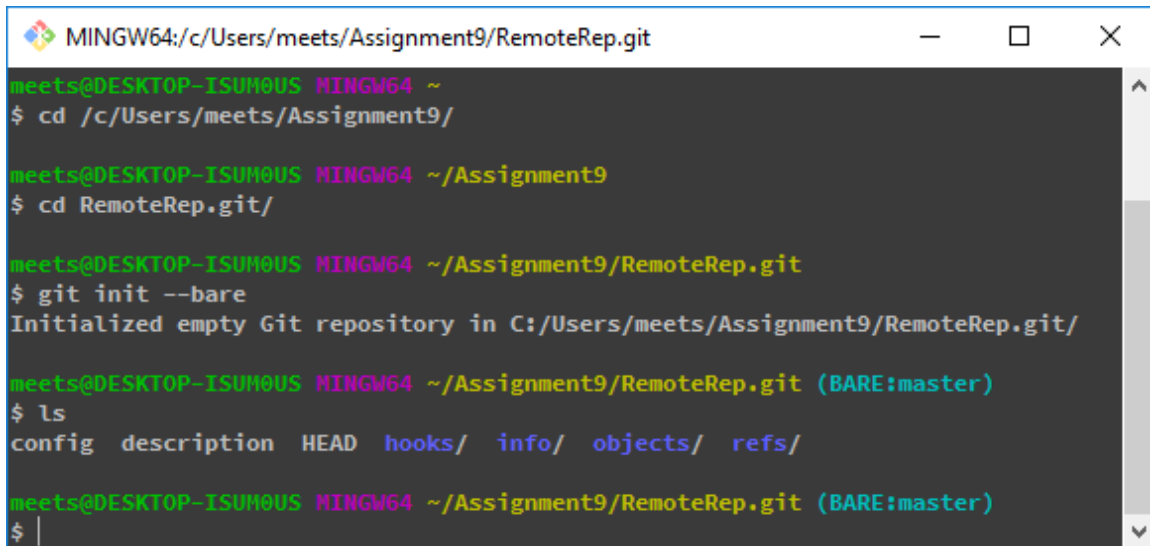
Repository Creation

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
```



```
MINGW64:/c/Users/meets/Assignment9/RemoteRep.git
meets@DESKTOP-ISUM0US MINGW64 ~
$ cd /c/Users/meets/Assignment9/

meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9
$ cd RemoteRep.git/

meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/RemoteRep.git
$ git init --bare
Initialized empty Git repository in C:/Users/meets/Assignment9/RemoteRep.git/

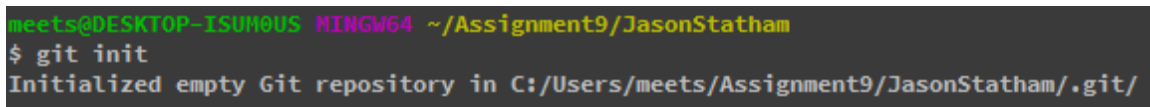
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/RemoteRep.git (BARE:master)
$ ls
config  description  HEAD  hooks/  info/  objects/  refs/

meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/RemoteRep.git (BARE:master)
$ |
```

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

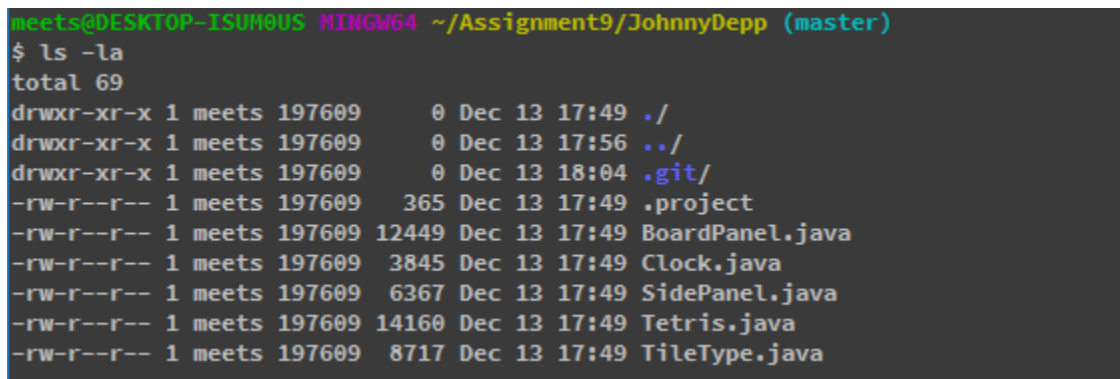
```
git init
```

A Sample screenshot is as shown below.



```
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JasonStatham
$ git init
Initialized empty Git repository in C:/Users/meets/Assignment9/JasonStatham/.git/
```

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



```
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JohnnyDepp (master)
$ ls -la
total 69
drwxr-xr-x 1 meets 197609  0 Dec 13 17:49 ./
drwxr-xr-x 1 meets 197609  0 Dec 13 17:56 ../
drwxr-xr-x 1 meets 197609  0 Dec 13 18:04 .git/
-rw-r--r-- 1 meets 197609  365 Dec 13 17:49 .project
-rw-r--r-- 1 meets 197609 12449 Dec 13 17:49 BoardPanel.java
-rw-r--r-- 1 meets 197609  3845 Dec 13 17:49 Clock.java
-rw-r--r-- 1 meets 197609  6367 Dec 13 17:49 SidePanel.java
-rw-r--r-- 1 meets 197609 14160 Dec 13 17:49 Tetris.java
-rw-r--r-- 1 meets 197609  8717 Dec 13 17:49 TileType.java
```

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:

```
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JohnnyDepp (master)
$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .project
        BoardPanel.java
        Clock.java
        SidePanel.java
        Tetris.java
        TileType.java

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

After configuring git ignore:

```
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JohnnyDepp (master)
$ touch .gitignore

meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JohnnyDepp (master)
$ vi .gitignore

meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JohnnyDepp (master)
$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gitignore
        BoardPanel.java
        Clock.java
        SidePanel.java
        Tetris.java
        TileType.java

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

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
```

```
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JohnnyDepp (master)
$ git push origin master
Counting objects: 10, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (9/9), done.
Writing objects: 100% (10/10), 13.18 KiB | 0 bytes/s, done.
Total 10 (delta 1), reused 0 (delta 0)
To C:/Users/meets/Assignment9/RemoteRep.git/
 * [new branch]      master -> master
```

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

```
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JohnnyDepp (master)
$ git log
commit db01c0208afea1e763436acff203e998a67f2df
Author: Johnny Depp <JohnnyDepp@umkc.edu>
Date:   Sun Dec 13 18:44:28 2015 -0600

    Initial import for Johnny Depp
```

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`
2. Commit the changes to local repository
`git commit -m "Edited the Tetris.java at 9 - Emma"`

```
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/EmmaWatson (master)
$ vi Tetris.java

meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/EmmaWatson (master)
$ 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:   Tetris.java

no changes added to commit (use "git add" and/or "git commit -a")

meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/EmmaWatson (master)
$ git add Tetris.java
warning: LF will be replaced by CRLF in Tetris.java.
The file will have its original line endings in your working directory.

meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/EmmaWatson (master)
$ git status
warning: LF will be replaced by CRLF in Tetris.java.
The file will have its original line endings in your working directory.
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    committed:   Tetris.java

meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/EmmaWatson (master)
$ git commit -m "Edited the Tetris.java at 9 - Emma"
[master warning: LF will be replaced by CRLF in Tetris.java.
The file will have its original line endings in your working directory.
ff8763d] Edited the Tetris.java at 9 - Emma
warning: LF will be replaced by CRLF in Tetris.java.
The file will have its original line endings in your working directory.
1 file changed, 1 insertion(+), 1 deletion(-)
```


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.

```
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/EmmaWatson (master)
$ git push origin master
To C:/Users/meets/Assignment9/RemoteRep.git/
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'C:/Users/meets/Assignment9/RemoteRep.git/'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

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.

```
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/EmmaWatson (master)
$ git pull origin master
warning: no common commits
remote: Counting objects: 10, done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 10 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (10/10), done.
From C:/Users/meets/Assignment9/RemoteRep
 * branch        master      -> FETCH_HEAD
 * [new branch]   master      -> origin/master
Auto-merging Tetris.java
CONFLICT (add/add): Merge conflict in Tetris.java
Automatic merge failed; fix conflicts and then commit the result.

meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/EmmaWatson (master|MERGING)
$ |
```

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

```
MINGW64:/c/Users/meets/Assignment9/EmmaWatson
package org.psnbtech;

import java.awt.BorderLayout;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.util.Random;

import javax.swing.JFrame;
<<<<<< HEAD
// 1. Comment added by Emma Watson
=====

>>>>>> db01c0208afea1e763436accff203e998a67f2df
/**
```

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`
2. After changes, add Tetris.java to staging area
`git add Tetris.java`
3. Commit the changes to local repository
`git commit -m "Merged Tetris.java with Emma's changes"`

```
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/EmmaWatson (master|MERGING)
$ vi Tetris.java

meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/EmmaWatson (master|MERGING)
$ git add Tetris.java

meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/EmmaWatson (master|MERGING)
$ git commit -m "Merged Tetris.java with Emma's changes"
[master 24ad109] Merged Tetris.java with Emma's changes
```

Now, Emma can successfully push her changes to remote repository

`git push origin master`

```
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/EmmaWatson (master)
$ git push origin master
Counting objects: 7, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 861 bytes | 0 bytes/s, done.
Total 7 (delta 3), reused 0 (delta 0)
To C:/Users/meets/Assignment9/RemoteRep.git/
db01c02..24ad109 master -> master
```

Now we can check the log using “`git log`”

```
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/EmmaWatson (master)
$ git log
commit 24ad10904f8eb3b9da57af6dedcf5f84277728a6
Merge: ff8763d db01c02
Author: Emma Watson <EmmaWatson@umkc.edu>
Date: Sun Dec 13 19:32:31 2015 -0600

    Merged Tetris.java with Emma's changes

commit ff8763d11c55b09f01511fbd1c7429449a90a309
Author: Emma Watson <EmmaWatson@umkc.edu>
Date: Sun Dec 13 19:03:49 2015 -0600

    Edited the Tetris.java at 9 - Emma

commit db01c0208afea1e763436acff203e998a67f2df
Author: Johnny Depp <JohnnyDepp@umkc.edu>
Date: Sun Dec 13 18:44:28 2015 -0600

    Initial import for Johnny Depp

commit 855eaa12fb05e5851fd9ea5efcfadcf4cdfbf66e
Author: Emma Watson <EmmaWatson@umkc.edu>
Date: Sun Dec 13 18:44:04 2015 -0600

    Initial import for Emma Watson
```

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:

```
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9
$ ls
EmmaWatson/ JohnnyDepp/ RemoteRep.git/

meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9
$ git clone file:///c:/Users/meets/Assignment9/RemoteRep.git JasonStatham
Cloning into 'JasonStatham'...
remote: Counting objects: 16, done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 16 (delta 3), reused 0 (delta 0)[ceiving objects: K
Receiving objects: 100% (16/16), 13.94 KiB | 0 bytes/s, done.
Resolving deltas: 100% (3/3), done.
Checking connectivity... done.

meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9
$ cd JasonStatham/

meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JasonStatham (master)
$ ls -l
total 56
-rw-r--r-- 1 meets 197609 12886 Dec 13 19:48 BoardPanel.java
-rw-r--r-- 1 meets 197609 3986 Dec 13 19:48 Clock.java
-rw-r--r-- 1 meets 197609 6593 Dec 13 19:48 SidePanel.java
-rw-r--r-- 1 meets 197609 14761 Dec 13 19:48 Tetris.java
-rw-r--r-- 1 meets 197609 9142 Dec 13 19:48 TileType.java
```

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
```

```
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JasonStatham (master)
$ git checkout -b dev_js
Switched to a new branch 'dev_js'

meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JasonStatham (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"
```

```
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JasonStatham (dev_js)
$ vi Clock.java

meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JasonStatham (dev_js)
$ git status
On branch dev_js
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:   Clock.java

no changes added to commit (use "git add" and/or "git commit -a")

meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JasonStatham (dev_js)
$ git add Clock.java

meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JasonStatham (dev_js)
$ git status
On branch dev_js
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   Clock.java

meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JasonStatham (dev_js)
$ git commit -m "Changed from private to public in Clock.java by Jason Statham"
[dev_js 1fa5d0b] Changed from private to public in Clock.java by Jason Statham
1 file changed, 2 insertions(+), 1 deletion(-)
```

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

```
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JasonStatham (dev_js)
$ git push origin dev_js
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 420 bytes | 0 bytes/s, done.
Total 3 (delta 2), reused 0 (delta 0)
To file:///c:/Users/meets/Assignment9/RemoteRep.git
 * [new branch]      dev_js -> 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

```
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JasonStatham (master)
$ git merge dev_js
Updating 24ad109..1fa5d0b
Fast-forward
 Clock.java | 3 ++-
1 file changed, 2 insertions(+), 1 deletion(-)
```

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
```

```
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JohnnyDepp (master)
$ git fetch origin
remote: Counting objects: 10, done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 10 (delta 5), reused 0 (delta 0)
Unpacking objects: 100% (10/10), done.
From C:/Users/meets/Assignment9/RemoteRep
* [new branch]      dev_js      -> origin/dev_js
db01c02..24ad109  master      -> origin/master
```

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

```
git branch -avv
```

```
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JohnnyDepp (master)
$ git branch -avv
* master          db01c02 Initial import for Johnny Depp
remotes/origin/dev_js 1fa5d0b Changed from private to public in Clock.java by Jason Statham
remotes/origin/master 24ad109 Merged Tetris.java with Emma's changes
```

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

3. Create a local branch "dev_jd" pointing to "origin/dev_js".

```
git checkout -b dev_jd origin/dev_js
```

```
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JohnnyDepp (master)
$ git checkout -b dev_jd origin/dev_js
Branch dev_jd set up to track remote branch dev_js from origin.
Switched to a new branch 'dev_jd'

meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JohnnyDepp (dev_jd)
```

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

```
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JohnnyDepp (dev_jd)
$ git status
On branch dev_jd
Your branch is up-to-date with 'origin/dev_js'.
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:   TileType.java

no changes added to commit (use "git add" and/or "git commit -a")
```

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.

```
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JohnnyDepp (dev_jd)
$ git status
warning: LF will be replaced by CRLF in TileType.java.
The file will have its original line endings in your working directory.
On branch dev_jd
Your branch is up-to-date with 'origin/dev_js'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   TileType.java
```

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`

```
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JohnnyDepp (dev_jd)
$ git reset TileType.java
warning: LF will be replaced by CRLF in TileType.java.
The file will have its original line endings in your working directory.
Unstaged changes after reset:
M       TileType.java

meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JohnnyDepp (dev_jd)
$ git status
On branch dev_jd
Your branch is up-to-date with 'origin/dev_js'.
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:   TileType.java

no changes added to commit (use "git add" and/or "git commit -a")
```

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`

```
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JohnnyDepp (master)
$ git branch -d dev_jd
warning: deleting branch 'dev_jd' that has been merged to
        'refs/remotes/origin/dev_js', but not yet merged to HEAD.
Deleted branch dev_jd (was 1fa5d0b).
```

- On remote repository:
`git push origin --delete dev_js`

```
meets@DESKTOP-ISUM0US MINGW64 ~/Assignment9/JohnnyDepp (master)
$ git push origin --delete dev_js
To C:/Users/meets/Assignment9/RemoteRep.git/
- [deleted]          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.