GIT Workflow for COTS

Last updated on 3/1/2021 by Sungwook Kwon

Version: 1.4

# Installation

This is the workflow for COTS team to maintain the version of FileNet/Kofax Projects using GIT.

## Configuration



### Local Repository

This is the repository that each user has in their local workstation. Once a user is participated in a project, the user will clone a whole project from the remote, COTS. Users will maintain their own version locally until the change is completely tested and ready for production. It is suggested to use an umbrella folder, “git” for all projects that are maintained by git under the folder.

### user (Remote – OneDrive - State of Ohio)

This is an optional repository that each user can have to make a backup of their local repository. Through this repository, users store any changes that they are working so that they will not be tied to a single workstation. If a change is small and it can be made quickly, users could use origin repository directly.

### origin (Remote – https://odjfs.visualstudio.com)

“origin” is the main repository for COTS team that is residing on https://odjfs.visualstudio.com. This is the repository that team members copy projects from and submit the changes that have been made, and completely tested. The source code and artifacts in this repository should be error-free and ready for deployment to production environment.

### Deployment Manager (Local - Optional)

Deployment Manager takes care of the deployment of all the changes that have been pushed to COTS repository. This repository should be on deployment server so that the change can easily be deployed to the designated environment. The manager pulling the most updated version from COTS repository and synchronize the changes with Dimension and perform the deployment.

### Folder Structure

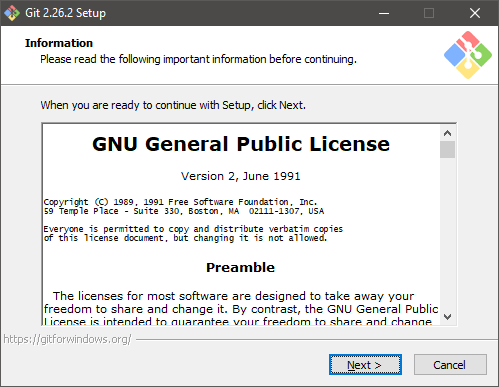
Each project will have its own root folder under GIT folder and the structure should be remained the same consistently with other project. New folders can be added as needed.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **COTS GIT Folder Structure**   |  |  |  |  | | --- | --- | --- | --- | | Git |  | | | | Proj-A | .git | Local Git Repository | | Documents | Folder for documents. | | Folders | Folders can be added as needed. | | Files | .gitignore  README.md | |

## GUI Client - Installation

The current version as of 5/6/2020 is 2.26.2. Get the code from the following site:  
<http://git-scm.com/downloads>

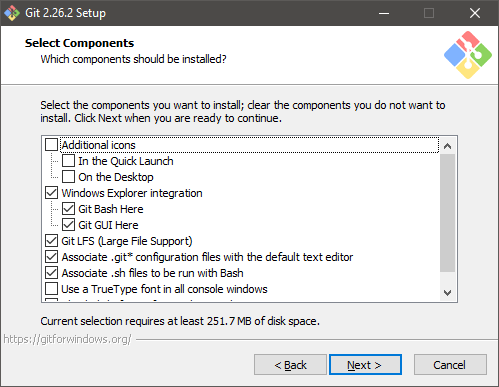
Download 61-bit Git for Windows Setup and follow the step for installation. Run the installer as Administrator.



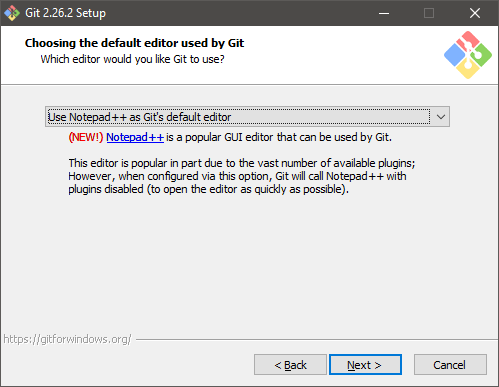
Accept the default installation location:

C:\Program Files\Git

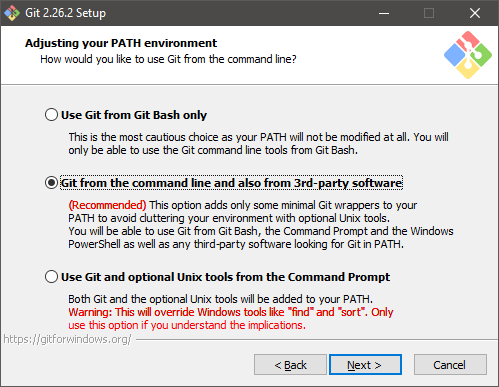
Select options. You can accept defaults for most cases.



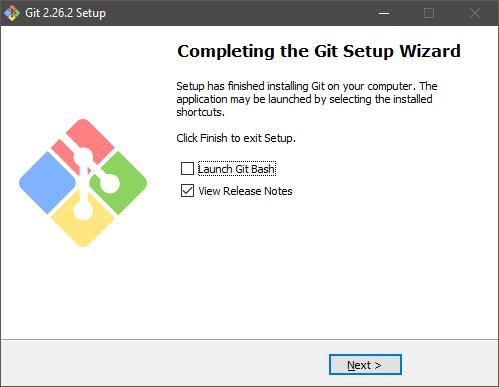
Change the default editor to your preferred editor.



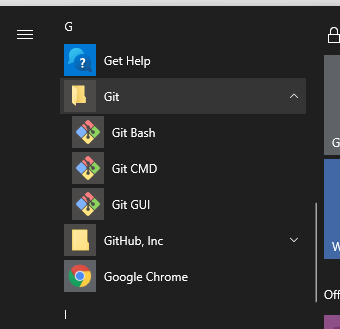
Accept recommendation for PATH.



Accept default selection for the rest of options.



Make sure new entries for GIT have been created in start menu.



## GIT Documentation – Pro Git

<https://git-scm.com/book/en/v2>

# User’s Guide

## Initial Set Up

This is one-time task that needs to be done before you start using GIT.

### Setup Identity

|  |  |
| --- | --- |
| The username and email address will be recorded with every commits that you make. | |
|  | Open Git CMD from start menu.  $ git config --global user.name "Sungwook Kwon"  $ git config --global user.email [sungwook.kwon@jfs.ohio.gov](mailto:sungwook.kwon@jfs.ohio.gov)  $ git config --global --list  user.name=SungWook Kwon  user.email=sungwook.kwon@jfs.ohio.gov  … |
| Ref. | <https://git-scm.com/docs/git-config> |

## Set up remote repository

### Clone remote repository

|  |  |
| --- | --- |
| Both local and remote don’t exist yet. Create a remote repository first then clone it to the local. | |
|  | Login Visual Studio repository with your state ID.  <http://odjfs.visualstudio.com>  All FileNet projects are found under “FileNet” project. Select “FileNet”.    Select the repository that needs to be copied (cloned) to the local copy (clone) of it.    To clone the repository, you need a url for the repository. Click “Clone” button to see the url and copy it.  Go to GIT Bash, then clone the repository with copied URL.  KWONS@JFSLT427170 MINGW64 ~  $ cd c:/git  KWONS@JFSLT427170 MINGW64 /c/git  KWONS@JFSLT427170 MINGW64 /c/git  $ git clone https://odjfs.visualstudio.com/Filenet/\_git/COTS-TEST  Cloning into 'COTS-TEST'...  warning: You appear to have cloned an empty repository.  KWONS@JFSLT427170 MINGW64 /c/git  $ cd cots-test  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (master)  $ git status  On branch master  No commits yet  nothing to commit (create/copy files and use "git add" to track)  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (master)  $ git remote -v  origin https://odjfs.visualstudio.com/Filenet/\_git/COTS-TEST (fetch)  origin https://odjfs.visualstudio.com/Filenet/\_git/COTS-TEST (push)  Verify local repository and link between local and remote repositories. Also, if there are any in the remote repository they must have been copied to the local folder. |
| Ref. | <https://git-scm.com/docs/git-clone> |

### Create a new repository for existing project

|  |  |
| --- | --- |
| Local repository already exists and remote repository needs to be created and linked to the local repository. | |
|  | 1. Create a remote repository with the same name as local repository (e.g. NewCETool) 2. Change directory to the local repository.   C:\Git\COTSWorkspace>cd newce\*  C:\Git\COTSWorkspace\NewCETool>git hist  \* 0d5e712 (HEAD -> master) adds SLF4J and Log4j for logging  \* cf1f7ed initial commit  C:\Git\COTSWorkspace\NewCETool>git remote -v  origin   1. Add the remote repository   C:\Git\COTSWorkspace\NewCETool>git remote add origin https://odjfs.visualstudio.com/Filenet/\_git/NewCETool  C:\Git\COTSWorkspace\NewCETool>git remote -v  origin https://odjfs.visualstudio.com/Filenet/\_git/NewCETool (fetch)  origin https://odjfs.visualstudio.com/Filenet/\_git/NewCETool (push)   1. Push the commits to the remote repository   C:\Git\COTSWorkspace\NewCETool>git push origin --all  Enumerating objects: 38, done.  Counting objects: 100% (38/38), done.  Delta compression using up to 4 threads  Compressing objects: 100% (23/23), done.  Writing objects: 100% (38/38), 5.42 KiB | 179.00 KiB/s, done.  Total 38 (delta 5), reused 0 (delta 0), pack-reused 0  remote: Analyzing objects... (38/38) (9 ms)  remote: Storing packfile... done (90 ms)  remote: Storing index... done (52 ms)  To https://odjfs.visualstudio.com/Filenet/\_git/NewCETool  \* [new branch] master -> master  C:\Git\COTSWorkspace\NewCETool>git hist  \* 0d5e712 (HEAD -> master, origin/master) adds SLF4J and Log4j for logging  \* cf1f7ed initial commit  C:\Git\COTSWorkspace\NewCETool> |

### Exclude files - .gitignore

|  |  |
| --- | --- |
| .gitignore file is a plain text file where each line contains a pattern for files/directories to ignore. The following items are found useful in FileNet projects. This file is located under each project folder. | |
| Steps | 1. create .gitignore file using any text editor  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (master)  $ npp .gitignore   |  | | --- | | # General Executable  \*.tmp  \*.temp  \*.exe  \*.log  # Git  \*.orig  # FileNet Deployment Manager  FDM/Temp/\*  FDM/Samples/\*  FDM/Environments/\*\*/Assets/\*  # Microsoft Office – All temporary files  ~$\* |   2. Make sure the file is detected by Git and add it to index KWONS@JFSLT427170 MINGW64 /c/git/cots-test (master)  $ git status  On branch master  No commits yet  Untracked files:  (use "git add <file>..." to include in what will be committed)  .gitignore  nothing added to commit but untracked files present (use "git add" to track)  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (master)  $ git add .  warning: CRLF will be replaced by LF in .gitignore.  The file will have its original line endings in your working directory  3. Verify the file is indexed and will be included in the next commit.  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (master)  $ git status  On branch master  No commits yet  Changes to be committed:  (use "git rm --cached <file>..." to unstage)  new file: .gitignore  4. Commit index including the new .gitignore file to local repository.  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (master)  $ git commit -m "initial commit"  [master (root-commit) febe914] initial commit  1 file changed, 17 insertions(+)  create mode 100644 .gitignore  5. Push the commit to remote repository, origin.  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (master)  $ git push origin master  Enumerating objects: 3, done.  Counting objects: 100% (3/3), done.  Delta compression using up to 4 threads  Compressing objects: 100% (2/2), done.  Writing objects: 100% (3/3), 374 bytes | 374.00 KiB/s, done.  Total 3 (delta 0), reused 0 (delta 0), pack-reused 0  remote: Analyzing objects... (3/3) (6 ms)  remote: Storing packfile... done (190 ms)  remote: Storing index... done (81 ms)  To https://odjfs.visualstudio.com/Filenet/\_git/COTS-TEST  \* [new branch] master -> master  6. branches are up to date with remote and local working tree is clean.  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (master)  $ git status  On branch master  Your branch is up to date with 'origin/master'.  nothing to commit, working tree clean  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (master)  $ git hist  \* febe914 (HEAD -> master, origin/master) initial commit  Now a new .gitignore file has been added to both local and remote repository. |
| Ref. | <https://git-scm.com/docs/gitignore> |

### Alias - hist

|  |  |
| --- | --- |
|  | |
|  | KWONS@JFSLT427170 MINGW64 /c/git/cots-test (master)  $ git config --global alias.hist "log --all --graph --decorate --oneline"  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (master)  $ git hist  \* b8193a4 (HEAD -> master, origin/master) adds README.MD file, RFC-002  \* febe914 initial commit  $ git log --since="yesterday" –oneline  $ git log --since="1/1/2020" --oneline |
| Ref. | <https://git-scm.com/book/en/v2/Git-Basics-Git-Aliases> |

## Basic Operation

### Happy Path

|  |  |
| --- | --- |
| Users add or update files in the working directory. Users will select all or some of the files to add them to a commit. A commit is a group of files to be changed to the local repository. Each commit has an ID. Users can make multiple commits to local before they push all the local commit to remote repository. When the change passes testing, free of errors, and ready for production, the commit will be pushed to the remote repository, “origin”. | |
| Steps | 1. Launch “Git Bash” from start. 2. Go to the working directory. (c:/git/COTS-TEST).   KWONS@JFSLT427170 MINGW64 ~  $ cd c:/git/cots-test   1. Check pending changes and history (log)   KWONS@JFSLT427170 MINGW64 /c/git/cots-test (master)  $ git status  On branch master  Your branch is up to date with 'origin/master'.  nothing to commit, working tree clean   1. Create a new branch (e.g. “RFC-002”) for your change and switch to the branch.   KWONS@JFSLT427170 MINGW64 /c/git/cots-test (master)  $ git checkout -b "RFC-002"  Switched to a new branch 'RFC-002'  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (RFC-002)  $ git hist  \* febe914 (HEAD -> RFC-002, origin/master, master) initial commit   1. Make change by adding a new file to the working tree.   KWONS@JFSLT427170 MINGW64 /c/git/cots-test (RFC-002)  $ npp README.MD  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (RFC-002)  $ cat readme.md  #TOPIC  1. Checkout  2. Index  3. Commit  4. Push  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (RFC-002)  $ ls -al  total 26  drwxr-xr-x 1 KWONS 1049089 0 Feb 26 13:50 ./  drwxr-xr-x 1 KWONS 1049089 0 Feb 26 12:17 ../  drwxr-xr-x 1 KWONS 1049089 0 Feb 26 13:52 .git/  -rw-r--r-- 1 KWONS 1049089 211 Feb 26 12:24 .gitignore  -rw-r--r-- 1 KWONS 1049089 51 Feb 26 13:50 README.MD   1. Index the file for a commit.   KWONS@JFSLT427170 MINGW64 /c/git/cots-test (RFC-002)  $ git add README.MD  warning: CRLF will be replaced by LF in README.MD.  The file will have its original line endings in your working directory  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (RFC-002)  $ git status  On branch RFC-002  Changes to be committed:  (use "git restore --staged <file>..." to unstage)  new file: README.MD   1. Commit with a commit message.   KWONS@JFSLT427170 MINGW64 /c/git/cots-test (RFC-002)  $ git commit -m "adds README.MD file, RFC-002"  [RFC-002 b8193a4] adds README.MD file, RFC-002  1 file changed, 5 insertions(+)  create mode 100644 README.MD  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (RFC-002)  $ git status  On branch RFC-002  nothing to commit, working tree clean  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (RFC-002)  $ git hist  \* b8193a4 (HEAD -> RFC-002) adds README.MD file, RFC-002  \* febe914 (origin/master, master) initial commit  Now RGC-002 branch has its own commit.   1. Apply the change to “master” branch. Check out master branch first.   KWONS@JFSLT427170 MINGW64 /c/git/cots-test (RFC-002)  $ git checkout master  Switched to branch 'master'  Your branch is up to date with 'origin/master'.  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (master)  $ git hist  \* b8193a4 (RFC-002) adds README.MD file, RFC-002  \* febe914 (HEAD -> master, origin/master) initial commit   1. Identify the commit to merge, b8193a4 is the commit id to be merged.   KWONS@JFSLT427170 MINGW64 /c/git/cots-test (master)  $ git merge b8193a4  Updating febe914..b8193a4  Fast-forward  README.MD | 5 +++++  1 file changed, 5 insertions(+)  create mode 100644 README.MD  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (master)  $ git hist  \* b8193a4 (HEAD -> master, RFC-002) adds README.MD file, RFC-002  \* febe914 (origin/master) initial commit   1. All the changes made to RFC-002 branch have been merged to master branch. RFC-002 branch is no longer needed. Delete RFC-002.   KWONS@JFSLT427170 MINGW64 /c/git/cots-test (master)  $ git branch -d RFC-002  Deleted branch RFC-002 (was b8193a4).  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (master)  $ git hist  \* b8193a4 (HEAD -> master) adds README.MD file, RFC-002  \* febe914 (origin/master) initial commit   1. Finally, the remote repository needs to be updated. Push the master branch to remote   KWONS@JFSLT427170 MINGW64 /c/git/cots-test (master)  $ git push origin master  Enumerating objects: 4, done.  Counting objects: 100% (4/4), done.  Delta compression using up to 4 threads  Compressing objects: 100% (2/2), done.  Writing objects: 100% (3/3), 351 bytes | 175.00 KiB/s, done.  Total 3 (delta 0), reused 0 (delta 0), pack-reused 0  remote: Analyzing objects... (3/3) (20 ms)  remote: Storing packfile... done (134 ms)  remote: Storing index... done (79 ms)  To https://odjfs.visualstudio.com/Filenet/\_git/COTS-TEST  febe914..b8193a4 master -> master  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (master)  $ git hist  \* b8193a4 (HEAD -> master, origin/master) adds README.MD file, RFC-002  \* febe914 initial commit  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (master)  $ git status  On branch master  Your branch is up to date with 'origin/master'.  nothing to commit, working tree clean |
| cmd | status, log (hist), add, branch, checkout, commit, push |

### Fetch

|  |  |
| --- | --- |
|  | |
|  | Before making any changes to the local repository, it is strongly recommended to download objects and refs from origin repository to your local repository using fetch command.  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (master)  $ git fetch origin master  From https://odjfs.visualstudio.com/Filenet/\_git/COTS-TEST  \* branch master -> FETCH\_HEAD  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (master)  $ git hist  \* b8193a4 (HEAD -> master, origin/master) adds README.MD file, RFC-002  In this case, there has been no changes made to remote repository since the user fetched the changes last time, therefore the user is good to go and make changes to the local repository.  While the user is making changes to a file, it is possible and allowed for other users to update the same file from their own local repository. In that case, user will need to resolve the conflict before the change can be committed and pushed back to the remote repository. Resolving conflict will be discussed in detail later. At this time, it would be much easier to avoid changing the same file simultaneously. |
| cmd | fetch |
| Ref | <https://git-scm.com/docs/git-fetch> |

### Add/Update

|  |  |
| --- | --- |
| Adding/Updating files to the local GIT repository to track the changes is two-step process: staging and commit.  All changes in the local working tree and its subdirectories are tracked by GIT, but they will not automatically be included to repository unless they are explicitly staged to index. Once the files with changes are staged to the index, they will be included in every commit from that time on. | |
| Steps | 1. The following example shows two changes; one is made to an existing file (README.MD) and the other is made to a new file(NEWFILE.TXT).   KWONS@JFSLT427170 MINGW64 /c/git/cots-test (master)  $ git status  On branch master  Your branch is up to date with 'origin/master'.  Changes not staged for commit:  (use "git add <file>..." to update what will be committed)  (use "git restore <file>..." to discard changes in working directory)  modified: README.MD  Untracked files:  (use "git add <file>..." to include in what will be committed)  NEWFILE.TXT  no changes added to commit (use "git add" and/or "git commit -a") |
|  | <https://git-scm.com/docs/git-add> |

### Rename/Move

|  |  |
| --- | --- |
|  | |
|  | KWONS@JFSLT427170 MINGW64 ~/projects/starter-web/level1 (master)  $ ls -al  total 8  drwxr-xr-x 1 KWONS 1049089 0 May 14 10:51 ./  drwxr-xr-x 1 KWONS 1049089 0 May 14 09:48 ../  -rw-r--r-- 1 KWONS 1049089 739 May 14 10:51 level1.txt  drwxr-xr-x 1 KWONS 1049089 0 May 14 10:52 level2/  KWONS@JFSLT427170 MINGW64 ~/projects/starter-web/level1 (master)  $ git mv level1.txt level1\_mod.txt  KWONS@JFSLT427170 MINGW64 ~/projects/starter-web/level1 (master)  $ git status  On branch master  Your branch is ahead of 'origin/master' by 6 commits.  (use "git push" to publish your local commits)  Changes to be committed:  (use "git restore --staged <file>..." to unstage)  renamed: level1.txt -> level1\_mod.txt  KWONS@JFSLT427170 MINGW64 ~/projects/starter-web/level1 (master) |
| Ref. | <https://git-scm.com/docs/git-mv> |

### init

|  |  |
| --- | --- |
|  | |
|  | Move to the root folder of Git projects  $ cd c:\git  KWONS@JFS398323 /c/git  $ mkdir OFC-ICPCDMS  KWONS@JFS398323 /c/git  $ cd OFC-ICPCDMS  KWONS@JFS398323 /c/git/OFC-ICPCDMS  $ git init  Initialized empty Git repository in c:/git/OFC-ICPCDMS/.git/  KWONS@JFS398323 /c/git/OFC-ICPCDMS (master)  $ |
| Ref. | [Git - git-init Documentation (git-scm.com)](https://git-scm.com/docs/git-init) |

### init -bare

|  |  |
| --- | --- |
|  | |
|  | (remote repository can be different that P: drive)  $ cd p:\git  KWONS@JFS398323 ~/git  $ mkdir OFC-ICPCDMS  KWONS@JFS398323 ~/git  $ cd OFC-ICPCDMS  KWONS@JFS398323 ~/git/OFC-ICPCDMS  $ git init --bare  Initialized empty Git repository in p:/git/OFC-ICPCDMS/  KWONS@JFS398323 ~/git/OFC-ICPCDMS (BARE:master)  $ |

### Remote

|  |  |
| --- | --- |
|  | |
|  | KWONS@JFS398323 /c/git/OFC-ICPCDMS (master)  $ git remote add KWONS file://p:/git/OFC-ICPCDMS  KWONS@JFS398323 /c/git/OFC-ICPCDMS (master)  $ git remote add COTS file://ms-kofax-pa10/git/OFC-ICPCDMS  KWONS@JFS398323 /c/git/OFC-ICPCDMS (master)  $ git remote -v  COTS file://ms-kofax-pa10/git/OFC-ICPCDMS (fetch)  COTS file://ms-kofax-pa10/git/OFC-ICPCDMS (push)  KWONS file://p:/git/OFC-ICPCDMS (fetch)  KWONS file://p:/git/OFC-ICPCDMS (push) |

### Branch

|  |  |
| --- | --- |
|  | |
|  | KWONS@JFSLT427170 MINGW64 /c/git/cots-test (master)  $ git hist  \* b8193a4 (HEAD -> master, origin/master) adds README.MD file, RFC-002  \* febe914 initial commit  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (master)  $ git checkout -b "NEW-BRANCH"  Switched to a new branch 'NEW-BRANCH'  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (NEW-BRANCH)  $ git branch -a  \* NEW-BRANCH  master  remotes/origin/master  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (NEW-BRANCH)  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (NEW-BRANCH)  $ git hist  \* b8193a4 (HEAD -> NEW-BRANCH, origin/master, master) adds README.MD file, RFC-002  \* febe914 initial commit  KWONS@JFSLT427170 MINGW64 /c/git/cots-test (NEW-BRANCH) |

### push

|  |  |
| --- | --- |
|  | |
|  | $ git log  commit d0fb2450d7961d295fdd158d1550ad8b9cbacf5b  Author: Sungwook Kwon <swooki@gmail.com>  Date: Wed Apr 30 11:40:36 2014 -0400  Initial Commit for ICPCDMS  KWONS@JFS398323 /c/git/OFC-ICPCDMS (master)  $ git push KWONS master  Counting objects: 6, done.  Delta compression using up to 4 threads.  Compressing objects: 100% (5/5), done.  Writing objects: 100% (6/6), 284.86 KiB | 0 bytes/s, done.  Total 6 (delta 0), reused 0 (delta 0)  To file://p:/git/OFC-ICPCDMS  \* [new branch] master -> master  KWONS@JFS398323 /c/git/OFC-ICPCDMS (master) |

### Commit to the new branch

|  |  |
| --- | --- |
| Make a change to one of the file and commit the change to the branch. | |
|  | KWONS@JFS398323 /c/git/OFC-ICPCDMS (ACL)  $ git status  On branch ACL  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: DOCS/OFC ICPCDMS\_ACL.xlsx  no changes added to commit (use "git add" and/or "git commit -a")  KWONS@JFS398323 /c/git/OFC-ICPCDMS (ACL)  $ git add \*  KWONS@JFS398323 /c/git/OFC-ICPCDMS (ACL)  $ git status  On branch ACL  Changes to be committed:  (use "git reset HEAD <file>..." to unstage)  modified: DOCS/OFC ICPCDMS\_ACL.xlsx  KWONS@JFS398323 /c/git/OFC-ICPCDMS (ACL)  $ git log  commit d0fb2450d7961d295fdd158d1550ad8b9cbacf5b  Author: Sungwook Kwon <swooki@gmail.com>  Date: Wed Apr 30 11:40:36 2014 -0400  Initial Commit for ICPCDMS  KWONS@JFS398323 /c/git/OFC-ICPCDMS (ACL)  $ git commit -m "ACL file has been cleaned"  [ACL a56fbde] ACL file has been cleaned  1 file changed, 0 insertions(+), 0 deletions(-)  KWONS@JFS398323 /c/git/OFC-ICPCDMS (ACL)  $ git log  commit a56fbde4aad276f8d5c2289060b5bcd9344e0e5e  Author: Sungwook Kwon <swooki@gmail.com>  Date: Wed Apr 30 13:37:16 2014 -0400  ACL file has been cleaned  commit d0fb2450d7961d295fdd158d1550ad8b9cbacf5b  Author: Sungwook Kwon <swooki@gmail.com>  Date: Wed Apr 30 11:40:36 2014 -0400  Initial Commit for ICPCDMS  KWONS@JFS398323 /c/git/OFC-ICPCDMS (ACL)  $ |

### Delete a remote branch

|  |  |
| --- | --- |
|  | |
|  | $ git push COTS --delete <branch name> |

### Tagging

|  |  |
| --- | --- |
|  | |
|  | **Simple Tagging Example / Lightweight Tags**  Simple example with *lightweight* tags and some basic tag commands.  **Command Listing**  pwd  cd projects/starter-web/  git status  git log --oneline --decorate --graph –all (hist)  git tag myTag  git log --oneline --decorate --graph --all  git tag --list  git show myTag  git tag --list  git tag --delete myTag  git tag --list  git log --oneline --decorate --graph --all  **Annotated Tags (“-a”)**  Annotated tags are tags with more information, like comments, associated with them.  **Command Listing**  pwd  git status  clear  git tag -a v-1.0  git tag --list  git log --oneline --decorate --graph --all  git show v-1.0  **Comparing Tags**  Tags mark important milestones in a project, so they are great way to compare what has happened between those milestones.  **Command Listing**  git tag -a v-1.1  mate simple.html  git commit -am "Updating for tag 1.1"  git commit --amend  git tag v-1.2 -m "Release 1.2"  git tag --list  git log --oneline --decorate --graph --all  git diff v-1.0 v-1.2  git difftool v-1.0 v-1.2  **Tagging a Specific Comment**  If you forget to tag a commit in the past, you can simply pass in the commit id while creating the tag.  **Command Listing**  pwd  git status  git log --oneline --decorate --graph --all  git tag -a v-0.9-beta 96ef75b  git log --oneline --decorate --graph --all  git tag -a v-0.8-alpha ab0d621  git log --oneline --decorate --graph --all  **Updating a Tag**  Sometimes mistakes happen and you'll need to update a tag.  **Command Listing**  pwd  git status  git log --oneline --decorate --graph --all  git tag -a v-0.8-alpha -f bd35d46  git log --oneline --decorate --graph --all  **Remote Tagging (GitHub)**  Working with remote tags on GitHub.  **Command Listing**  pwd  git status  git tag --list  git log --oneline --decorate --graph --all  git push origin v-0.9-beta  git push origin v-1.1  clear  git push origin master --tags  git push origin :v-0.8-alpha (remove tag from remote repository-github) |

### Reset and Reflog

|  |  |
| --- | --- |
|  | |
|  | $git reset HEAD^1  $ git reflog  1e68130 (HEAD -> master, tag: v1.0.2, origin/master, origin/HEAD) HEAD@{0}: pull origin master: Fast-forward  9cffe78 (tag: v1.0.1) HEAD@{1}: reset: moving to HEAD^1  1e68130 (HEAD -> master, tag: v1.0.2, origin/master, origin/HEAD) HEAD@{2}: commit (amend): Release 1.0.2  c1d7ca2 HEAD@{3}: commit: v1.0.1  9cffe78 (tag: v1.0.1) HEAD@{4}: commit: updates index.html  ecb1e10 (tag: v1.0.0) HEAD@{5}: merge new-changes: Fast-forward  1f282bd HEAD@{6}: checkout: moving from new-changes to master  ecb1e10 (tag: v1.0.0) HEAD@{7}: commit: commit into a new branch  1f282bd HEAD@{8}: reset: moving to HEAD  1f282bd HEAD@{9}: reset: moving to HEAD  1f282bd HEAD@{10}: reset: moving to HEAD  Use reset to a commit in reflog to undo what you have performed to GIT |

### How to resolve a binary file conflict with Git

|  |  |
| --- | --- |
|  | |
|  | Auto-merging somefile.dll  CONFLICT (content): Merge conflict in somefile.dll  Automatic merge failed; fix conflicts and then commit the result.  In this scenario, somefile.dll is a binary file that has been modified in both the current branch, and the branch you are attempting to merge in to the current branch. Since the file cannot be textually merged, you need to make a decision: do you keep the version of the file in your current branch, or the version in the other branch.  ## Resolve using mine  The file in your working copy is still the copy from your current branch – in other words, it was not modified by the merge attempt. To resolve the conflict and keep this file:  git add somefile.dll  git commit –m “My commit message for the merge”  ## Resolve using theirs  If you prefer to resolve the conflict using their copy, you need to get the version of the file from the branch you were trying to merge in:  git checkout otherbranch somefile.dll  Now that you have the correct version of the file in your working copy, you can mark it as resolved (by adding it), and commit:  git add somefile.dll  git commit –m “My commit message for the merge”  Note that in place of otherbranch, you can use any name (treeish) that refers to a branch: a local branch name (otherbranch), a remote branch name (origin/master), a specific commit SHA (980e3cc), etc. For example, if you were merging in from your remote when you received the conflict, and you wanted to resolve using the remote version, you would retrieve that copy of the file using:  git checkout origin/master somefile.dll |