GIT Basic Usage guide

[ Objective 2](#_Toc29845341)

[ WebEx Link to training session 2](#_Toc29845342)

[ What is GIT 2](#_Toc29845343)

[ Git clients & Tools 2](#_Toc29845344)

[ Configuring GIT Clients 2](#_Toc29845345)

[ GitHub URL 2](#_Toc29845346)

[ GIT commit/push procedure and standards 2](#_Toc29845347)

[ Clone repository 3](#_Toc29845348)

[ Checkout remote branch 3](#_Toc29845349)

[ Pull latest code from server for the current branch 4](#_Toc29845350)

[ Commit & Push Code 4](#_Toc29845351)

[ See differences in file 5](#_Toc29845352)

[ Using Stash 5](#_Toc29845353)

[ Resolving Merge conflicts 5](#_Toc29845354)

[ Down merging a branch 5](#_Toc29845355)

[ Pull requests 5](#_Toc29845356)

[ Further Reading & References 6](#_Toc29845357)

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Author | Action - Comment |
| 13 Jan 2020 | 1.0.0 | Milan | V1 |
|  |  |  |  |
|  |  |  |  |

# Objective

To get started with GIT, Best Practices, Basic Git commands and references

# WebEx Link to training session

Webex link: <https://ibm.webex.com/recordingservice/sites/ibm/recording/playback/461d3b1de1c24116bcce37005cded86c>

Code: BvaZWn66

# What is GIT

Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows.

# Git clients & Tools

To work with GIT we need a git client, following are some useful tools

* CLI GIT : [git-scm.com](https://git-scm.com/)
* UI GIT Client
  + [desktop.github.com](https://desktop.github.com/)
  + [sourcetreeapp.com](https://www.sourcetreeapp.com/)
* Plugins
  + Sublime Git gutter <https://github.com/jisaacks/GitGutter>

The above tools are just for reference and may not be approved, make sure to install only approved tools

# Configuring GIT Clients

Refer:

* <https://w3-connections.ibm.com/wikis/home?lang=en-us#!/wiki/W871140b04194_4d36_a993_cf7efe1bf660/page/GitHub%20Enterprise%20&%20SourceTree>
* <https://w3.ibm.com/services/lighthouse/documents/136797>
* <https://github.ibm.com/settings/tokens>

# GitHub URL

* <https://github.ibm.com/DAL-WH-Services>

# GIT commit/push procedure and standards

For latest Refer Confluence : <https://vbc-confluence.watson-health.ibm.com/pages/viewpage.action?pageId=135674226>

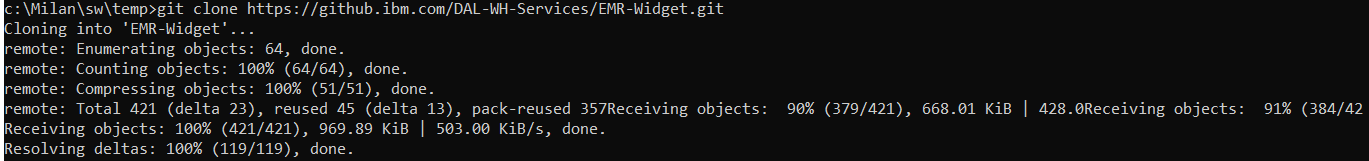
1. Team Decide a  common master branch
2. Every team member to create their own branch for everyday work (consider adding JIRA number in that branch name)
3. Every code commits SHOULD have a comment and associated JIRA number in comment
4. Make sure you push you code daily in your own branch (even if not functional or partial code).
5. Once your code reasonably ready is not breaking others stuff,  then create a pull request to the  master branch which Team decides is common master (in step 1)
6. Ask others peer developers to review your pull request
7. Team members to add review comments in Github itself
8. Developer Fix any review comments to his code and acknowledges in Githbub review comment
9. Only Team leads need to merge individuals PR to common master branch once review comments are resolved.
10. Delete the old branch if it wont be use any more

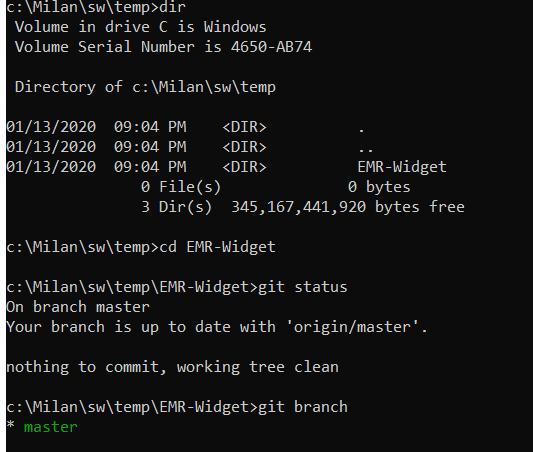
# Clone repository

Use the below command to clone the repository in the folder you want

* git clone <GIT URL>

|  |
| --- |
| **Example** :  git clone https://github.ibm.com/DAL-WH-Services/EMR-Widget.git |



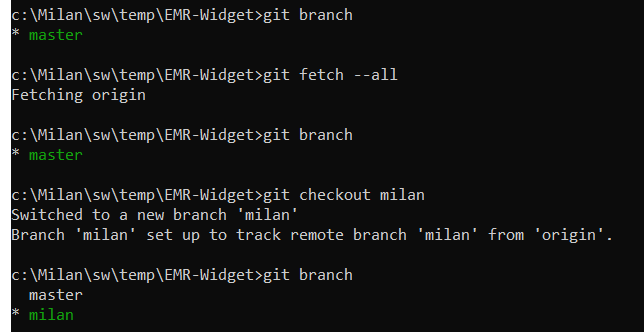


# Checkout remote branch

Use the below command to clone the repository in the folder you want

* git fetch --all
* **git checkout <remoteBranchName>**

|  |
| --- |
| **Example** :  git fetch –all  git checkout milan |



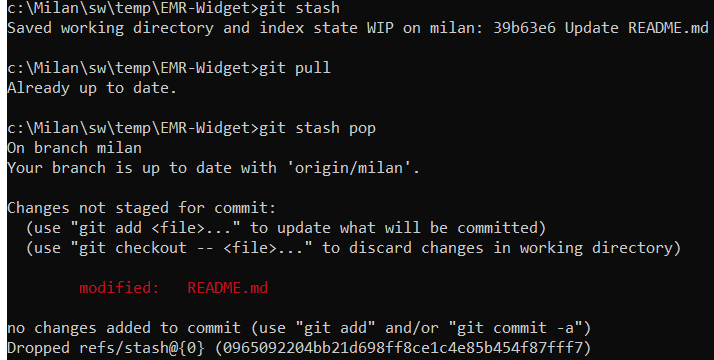
# Pull latest code from server for the current branch

Make sure you are on the branch you want to get latest code (use git branch & git checkout)

Then, use below commands

* git stash (will stash any working code to make room for new code)
* **git pull** (will pull latest code from server)
* git stash pop (will unstash/restore any working code)

|  |
| --- |
| **Example** :  git stash  git pull  git stash pop |



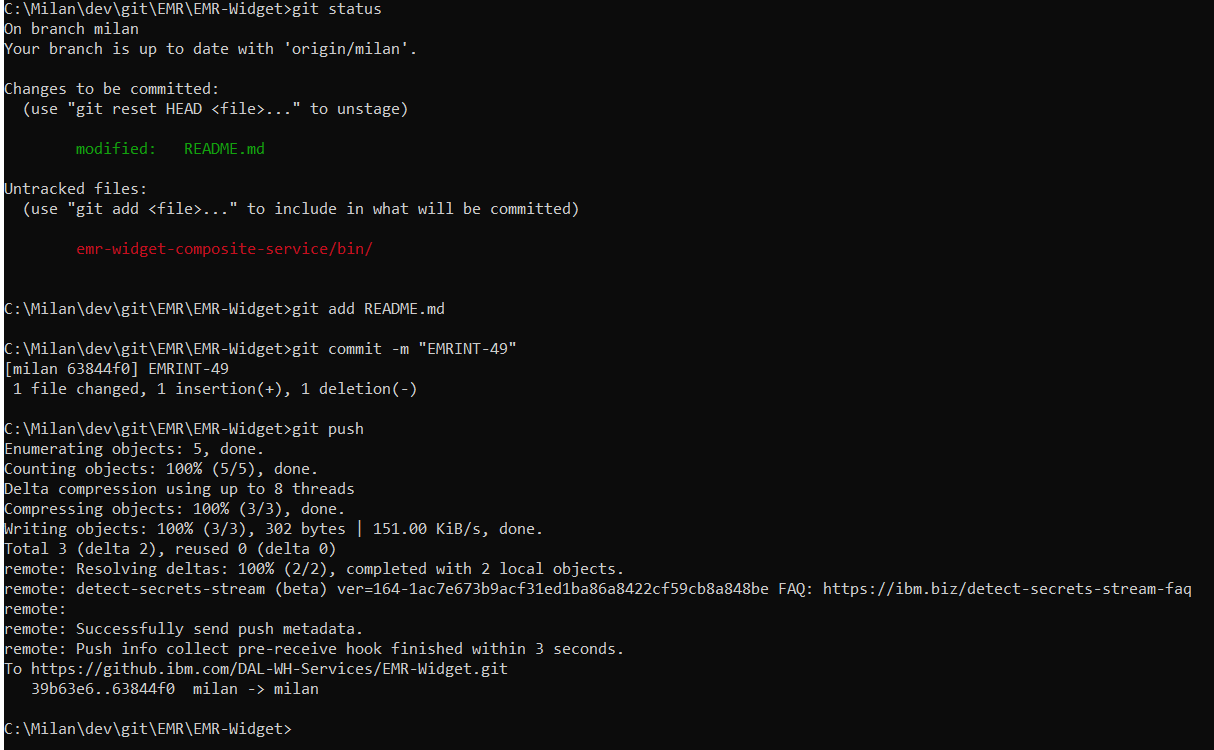
# Commit & Push Code

Make sure to pull latest code from server before you push your change else it will give error

Then, use below commands

* git status (display which files/folders are updated and needs to be staged)
* git add <file/folder> (stages file names which you want to commit & push)
* git commit -m "<JIRA\_NUMBER>: some comment" (commits the code still in your local, use push to send to server)
* git push (pushes all committed code to server)
* git status (to review the final status)

|  |
| --- |
| **Example** :  git status  git add README.md  git commit -m "EMRINT-49"  git push  git status |

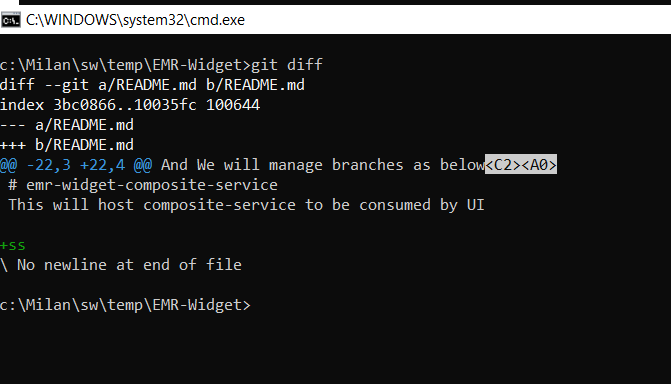


# See differences in file

See changes made in a file / all files

* git diff (to see difference of all files)
* git diff <fileName> (to see difference of specific file)

|  |
| --- |
| **Example** :  git diff README.md |



# Using Stash

You can use git stash to stash the current working code and restore when required

**Basic usage**

* git stash (will stash any working code to make room for new code)
* git stash pop (will unstash/restore any working code)

**Advance usage**

* <https://git-scm.com/docs/git-stash>

# Resolving Merge conflicts

TBD..

# Down merging a branch

TBD..

# Pull requests

TBD..

# Further Reading & References

* Branching Strategy :
* Cheat Sheets
  + https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet
  + https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf
* Git Docs : <https://git-scm.com/docs>
* <https://www.atlassian.com/git/tutorials/syncing>
* Branching strategy : <https://vbc-confluence.watson-health.ibm.com/pages/viewpage.action?pageId=135674226>
* Webex link using git: <https://ibm.webex.com/recordingservice/sites/ibm/recording/playback/461d3b1de1c24116bcce37005cded86c>

Code: BvaZWn66