**Basic git commands**

**Cache credential: git config --global credential.helper wincred // require git for Windows**

**Info: status ; branch ; log ; tag, remote -v ; branch -r**  
Ignore: create a file .gitignore**Stage/all: add {fname} ; add .  
Commit: commit ; commit –m “my comment”**UnWork: checkout -- {fname}  
UnStage: reset HEAD {fname} ; reset .  
**UnCommit to Stage: reset --soft HEAD^**UnCommit to Work: reset --hard HEAD^  
Stash: stash ; stash pop ; stash drop ; stash list  
**Branch Add: branch {bname}   
Branch Switch: checkout {bname/.}**

Branch & Switch: checkout –b {tbname} {fbname}  
**Branch Merge: merge {frombname}  
Branch Del: branch –d {bname}**

**Branch Del Remote: push origin –d {bname} // tag b4 delete!**Rename: mv {sfname} {dfname}  
**Del: rm {fname}**Diff Commit: diff HEAD ; diff HEAD^ //stage & unstage  
Skip Stage & Commit: commit -all -m "my comment"  
AppendCommit: commit --amend -m "my comment"  
**Clone: clone {url}**Add: remote add {rrepo} {url}  
**Push 1st time: push -u {rrepo} {lbname}** //-u aka --set-upstream

Push/Pull: push ; pull // i.e. fetch + merge {lbname}  
Fetch: fetch // into lrepo lbname  
Tag: tag -a vx.x.x.x -m "ver. comment" ; push --tags  
Color: config --global color.ui true  
Graph: log --oneline --graph  
Range: log --since=1.month.ago --until=1.day.ago  
VIM: ‘i’ ; [ESC] ; :x ; :q

## Git Setup

* Install “Git for Windows”, make sure to choose the Windows Explorer integration w/ Git Bash & Git Gui
* Run Git Bash
  + Generate SSH key for your machine if one not existed yet
  + Grab you new SSK key from Git Gui -> Help -> Show Key
  + Add the SSH key to GitHub
* Config your global name, email, username, API token for ALL of your local Git operations
  + E.g. git config --global user.name/email | github.user/token

## Visual Studio Git Source Provider

* NuGet “**Git Source Control Provider**”

## New Repo

* Created At GitHub
  + **Click “New Repository”**
  + **Copy the *remote repo url***
  + Shown a list of Git commands instructions
* At Local
  + Git Bash (The following git commands are copied from GitHub instructions)
    - **git init**
    - **git add**
    - **git commit -m** “First Comment”
    - **git remote add origin *remote repo url //create local alias ‘origin’ point to remote repo url***
    - **git push -u origin master // push to remote *repo* ‘origin’, *branch* ‘master’**

## Get Existing Repo from GitHub

* At GitHub:
  + **Copy down the git url**
  + grab the **upstream\_git\_url** if the current repo is created by fork
* Local
  + **git clone git\_url**
    - A default alias “**origin**” is **AUTO CREATED** for the git\_url
  + Optionally pull changes from upstream repo:
    - cd “into the above newly cloned directory“
    - Create local alias ‘upstream’ point to repo url: **git remote add upstream upstream\_git\_url**

## Git trees

## Branch / Fork

* If you own the repository, you “Branch” it
* If you DON’T OWN the repository, you “Fork” it.
  + e.g. xyzUsername/abcProjectName forked into vincenthome/abcProjectName

## Sync Remote Upstream & Forked Branches

* You ‘ve created your own fork, made some changes. Your fork **WILL NOT** magically update itself to the latest version of the original fork.
* Run the following commands
  + **git checkout master (if you are not currently at master)**
  + **git fetch upstream // get latest from remote upstream to local upstream**
  + **git merge upstream/master // merge upstream with forked local master branch**
  + **git push origin master**

## Fork & Submit Pull Request

* <http://blogs.msdn.com/b/cdndevs/archive/2016/01/06/submitting-your-first-pull-request.aspx>

## Track/Stage/Commit/Push/Pull

* Visual Studio
  + **Git pending changes window**
    - **Right-click**
      * **Stage File**
      * **Add comment**
      * **Commit**
    - **Switch Branch**
      * Switch Branch
      * Create Branch
    - **View History**
      * Visual History w/ Head, master, hotfix, origin/master
    - **Git Bash**
      * **git push -u origin master**
      * **git fetch origin** // w/o merging
      * **git pull** // w/ merging
* Git Bash
  + Track & Stage new file: git add <filename/directory> e.g. readme.txt, \*.cs, \*.aspx
  + Add tracked file to Stage: git add <filename/directory>
  + Commit file from stage to local repo: git commit –m “message comment”
    - ONLY the version of the file that is in Stage will be commited.
  + Repo w/o Upstream
    - Send commit to GitHub: git push -u origin master
    - Pull in all changes from repo w/o merge: git fetch orgin
  + Repo w/ Upstream
    - Send commit to GitHub: git push origin master
    - Pull in all changes from repo w/o merge: git fetch orgin
    - Pull in all changes from upstream repo w/o merge: git fetch upstream
  + Ignore Staging – i.e. no need to use git add
    - git commit -a –m “message comment”
  + p.s. git pull // will fetch and merge automatically
  + Branch
    - Create: git branch hotfix
    - Switch: Git checkout hotfix
    - Create & Switch: git checkout -b hotfix
    - Delete: git branch -d hotfix
  + Merge
    - git checkout master
    - git merge hotfix // i.e. merge hotfix branch into the current master branch

## Utils

* Ignore file: create a **.gitignore** file in the project root directory, and in sub directories if you need overrid
* Differences:
  + Not Staged: git diff
  + Staged: git diff --staged
* Remove
  + git rm filename // this will delete the file from directory & move to Stage
  + Must follow up by commit and push
* Move (Rename)
  + git mv README readme.txt
  + Must follow up by commit and push
* Changing your last commit
  + Fixing just the commit message, assuming Stage is empty
    - git commit –m “replacing message” --amend
  + Add additional staging stuff to last commit
    - git commit –m “message comment” --amend
* Unstage a file
  + git reset HEAD filename
* Revert the changes of a Non-Staged file - DANGER!!! You will lose your changes!!!
  + git checkout – filename
* Rename a remote alias
  + git remote rename orgin origin
* **Tag**
  + **git tag -a v1.0 -m “message comment”**
  + git push origin v1.0 // push the tag to remote repo
  + git tag // quick list of all tags
  + git show v1.0 // display the details of a single tag

## Info

* git <verb> --help
* git config --list
* git status // show file status – untracked, modified, committed
* git remote -v // list the remote alias and urls
* git log

## Visual Studio Team Service – hard to use

Setup Git

* [Install Git Credential Manager](https://www.visualstudio.com/docs/git/set-up-credential-managers) for Microsoft Account Login
* Now you can use Git Bash to clone etc…