5113

Use same names

ConsistentNaming

<https://www.codeproject.com/Articles/838097/CRUD-Operations-Using-the-Generic-Repository-Pat>

Ref: <http://tfs2017.hosts.application.enet:8080/tfs/Edu_Collection/ITSP/_git/dev-training?_a=preview&path=%2Fdocs%2Fbranching-strategy%2Fteam-branching-strategy.md&version=GBmaster>

**New Local Branch**:

**Team Explorer** => Branches => **Right-click** the parent branch (usually **master**) to base your changes => choose New Local Branch From => Type user Name (**nk2774)/branch name** (feature or a bug fix name) => **Checkout branch ticked** => click **Create Branch**

Commit (commit change is for local check in)

Visual Studio automatically performs a checkout to the newly created branch.

Naming conventions to be used:

•users/**nk2774**/description

•users/**nk2774**/workitem

•fixes/workitem

•releases/version

**Push** (changes go to the server/publish)

Sync => Fetch (get changes from server/remote branch)

**remote branch**: The remote branch is automatically created when you push it to the remote server. So when you feel ready for it, you can just do:

git push <remote-name> <branch-name>

Pull is (fetch/get latest & merger to the local) : > git pull origin master

**Pull = Fetch + Merge**

If there is code conflict than pull fails and we need to **Fetch first and before merge compare**

After Fetch (for merging) go to Branches => merge => origin/master (from dropdown) => into current branch => Merge

**Pull Request**: Merging from a branch to the master is called Pull request (actually pushing to the master).

Mark auto complete by this changes will be **squashed** into master (single comment history). Branch will be **deleted**.

To Delete:

Team Explorer -> **Branches** -> select development Branch -> right click -> Delete -> select development\_ Branch under remotes/origin -> Delete Branch From Remote.

VS -> Team Explorer -> **Branches** -> double click master branch -> Merge -> select your BranchName from Branch menu => Merge

Click home => Pull request => Create New => master < branch1 => comments => Create pull request

Confirm the request: go to browser => Pull request tab => select the pull request => ask for review => confirm the request

For local master: In VS Team Explorer => Branches => master => Sync => Pull

Reset master: in case if we commit some thing by mistake against local master just and we don’t want to keep it do a rest:

Git checkout master

git reset head~1 --hard

**Cherry Pick:** Cherry picking in git means to choose a commit from one branch and apply it onto another.

This is in contrast with other ways such as merge and rebase which normally apply many commits onto another branch.

In case of conflicts

Got path till directory

* Open .git
* Git branch –h
* Git status
* Git branch –m users/Naveed/NewBranch1

Git Ignore file is a list of files like bin, debug folder, result of builds, or nugget packages etc. which we don’t want to commit.

Get latest: Pull = Fetch + Merge. In case conflict fetch and then merge from origin master to the local branch.

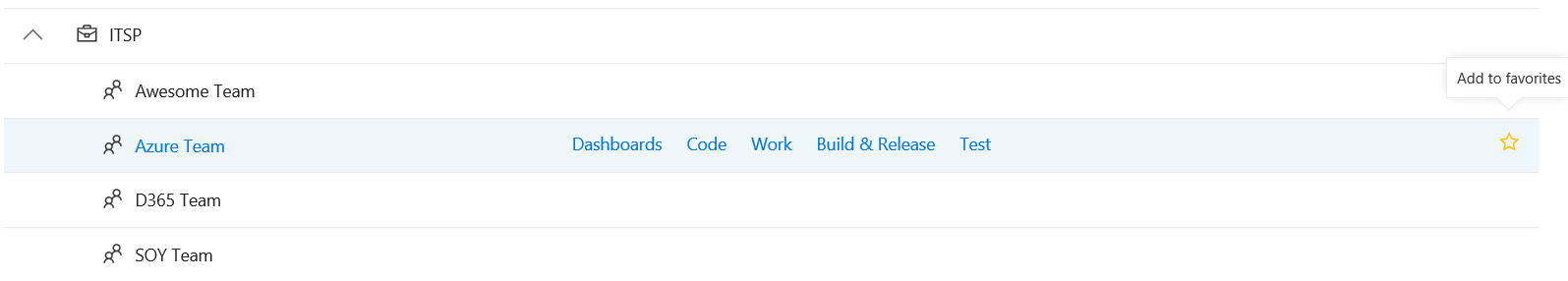
Git is a distributed version control system because every user has a local copy of the code along with history/commits (local full working copy).

**Pull request**:

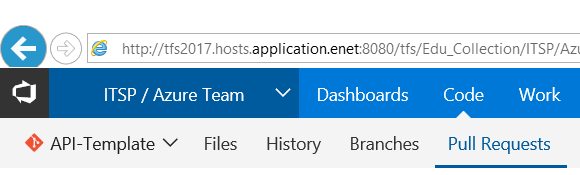
When you create a pull request the team that is the current context is added as a reviewer.  You can see the current team context in the top left corner of the web page.  If your context is ITSP, use the drop down to select **ITSP/Azure Team** or browse.

cid:image002.png@01D42329.D849BF10

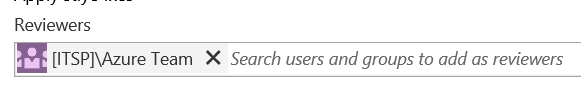
From the browse screen select the expand icon to the left of the project name and choose the Azure Team, you can also add it to your favourites so it appears on your TFS Collection home page.



Now when creating a pull request the reviewer added is [ITSP]\Azure Team.  We have a reviewer set using the branch policy, so you can delete the default reviewer and let the branch policy manage it.



Default reviewer based on the current context



The team context is also used when looking at the backlogs (Work).

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* rebase \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**rebase**: the main uses of rebase are to squash commits into a single one (so you don't have 12 commits that are saying things like "minor change"), and you pick all the commits you want to use. Since you're new to git- you probably want to pick all of them- but if you ever want to know how to pick only certain commits- this is how! Read more about git rebase here:

<https://robots.thoughtbot.com/git-interactive-rebase-squash-amend-rewriting-history>

caller

**To Do**:

Checkout master and pull

=> git checkout master

=> git pull

Switch back to work branch

=> git checkout {users/username/description}

Tip if you donot want to type the branch name

=> git checkout @{-1}

Rebase from master

* git rebase master
* git rebase -–continue

Resolve conflicts (close your solution in VS and use only Team Explorer)

Push your local branch (force-with-lease)

* git push --force-with-lease

**Tips**:

If you get lost at any stage during rebase just abort/undo

* git rebase – abort

to see the log

* git log –online

if log is very long and you want to get out of it press q

To **squash all** the commits into single one (instead of resolving many conflicts during rebase do rest to a point e.g. 629fedd and after that rebase). (we need to do a push after git rest 629fedd –soft)

* git rest 629fedd –soft
* git push

After resolving the conflict manually and updating the index with the desired resolution, you can continue the rebasing process with

git rebase -–continue

**--continue** is Restart the rebasing process after having resolved a merge conflict.

you can undo the git rebase with

git rebase --abort

<https://git-scm.com/docs/git-rebase>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* end rebase \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Git/NPM Behind a corporate Proxy Server**:

git config --global http.proxy <http://NK2774:password@proxy.dmz.ige:8080>

git config --global https.proxy <https://NK2774:password@proxy.dmz.ige:8080>

check setting: git config --global -l

Git Dir=> 'C:\Program Files\Git\' 'C:\Program Files (x86)\Git\'

$ npm config set proxy http://proxy-server-url:<port>

$ npm config set https-proxy [http://proxy-server-url:<port](http://proxy-server-url:%3cport)>

**With user/password**:

npm config set proxy http://NK2774:password@proxy.dmz.ige:8080

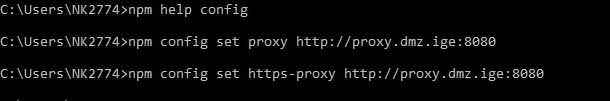
npm config set https-proxy [http://NK2774: password@proxy.dmz.ige:8080](http://NK2774:%20password@proxy.dmz.ige:8080)

npm config set strict-ssl=false

npm config set proxy http://<username>:<password>@<proxy-server-url>:<port>

npm config set https-proxy http://<username>:<password>@<proxy-server-url>:<port>

npm config set proxy "http://domain\username:password@servername:port/"

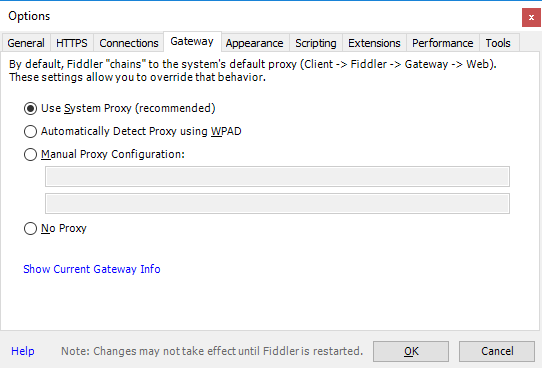


**Git**: git config --global http.proxy <http://mydomain\\myusername:mypassword@myproxyserver:8080>

git config --global http.proxy <http://proxy.mycompany:80>

To find proxy use fiddler and type: http://wpad/**wpad.dat** in any browser to open wpad.dat file.

Click on current Gateway info



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*----------++++++++++

**Accidentally committing to local master**: If you want to prevent yourself from accidentally committing to your local master add a pre-commit hook to the “{Path to local repo}\.git\hooks” folder.

• Go to your repository.

Go to folder .git\hooks (note .git\hooks is hidden by default)

• Create file .git/hooks/**pre-commit** with following content, make sure the file has no file extension:

#!/bin/sh

branch="$(git rev-parse --abbrev-ref HEAD)"

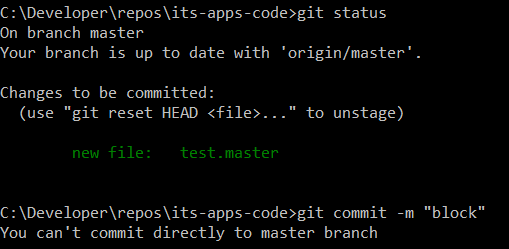
if [ "$branch" = "master" ]; then

echo "You can't commit directly to master branch"

exit 1

fi

If done correctly you will get a message like the one here: do some change => stage change => git status



**git hooks**: Git hooks are scripts that are run when certain actions occur, such as pre-commit (before commit), commit-msg (commit message), etc.

**How to ignore certain files in git?**

Open file .gitignore and type/past /HelloFile.txt

If you want to exclude all .txt files from git, add the following line to .gitignore: \*.txt