1. config git
2. make sure Origin point to own folk repo, and clone from folk repo
3. add remote upstream repo
4. pull from the upstream develop branch
5. checkout a feature branch (if not create yet, create a feature branch)
6. push code to origin repo stream's feature branch

**Git/GitHub**

configure your git - in command line:

|  |
| --- |
| git config --global [user.name](http://user.name/) "FirstName LastName (yourStaffId)"  git config --global user.email "yourHSBCemailSame[AsInGithub@email.com](mailto:AsInGithub@email.com)"  git config --global credential.helper wincred  git config --global http.sslVerify false |

Make sure you are a member of GlobalStandardsCDD [group](https://alm-github.systems.uk.hsbc/orgs/GlobalStandardsCDD/people)

Fork cdd-ui-webapp [repository](https://alm-github.systems.uk.hsbc/GlobalStandardsCDD/cdd-ui-webapp)

clone your fork:

|  |
| --- |
| git clone https://[alm-github.systems.uk](http://alm-github.systems.uk/).hsbc/{yourGithubIdHere}/cdd-ui-webapp |

add [upstream remote](https://help.github.com/articles/configuring-a-remote-for-a-fork/) to your repo:

|  |
| --- |
| cd to the repo  git remote add upstream https://[alm-github.systems.uk](http://alm-github.systems.uk/).hsbc/GlobalStandardsCDD/cdd-ui-webapp.git  git checkout develop to make sure you're on the develop branch  git pull upstream develop to get the latest code on develop from GlobalStandardsCDD/cdd-ui-webapp  pull from upstream at least once a day, better to have 3-5 times (more often the better) |

Built the project

|  |
| --- |
| mvn -U clean install |

Run the project

|  |
| --- |
| mvn jetty:run -Dstub=true -Denv=sit2  if want to skip download dependancy:  mvn jetty:run -Dstub=true -Denv=familiarisation -Dmdep.skip=true |

go to

http:localhost:8080/cdd/home.html

Writing Codes

**Contributing**

get the latest code from upstream develop to your develop fork

git checkout develop && git pull upstream develop

create a new branch for your feature (with rtc number and feature name)

git checkout -b 34587\_superAwesomeFeature

push your branch to your fork

git push -u origin 34587\_superAwesomeFeature

Make sure your work is in line with commit checklist (see below)

When you're ready to have your feature merged to GlobalStandardsCDD/cdd-ui-webapp

go to [PRs tab](https://alm-github.systems.uk.hsbc/GlobalStandardsCDD/cdd-ui-webapp/pulls)

click New pull request

click compare across forks

point to a branch in GlobalStandardsCDD/cdd-ui-webapp that you want to merge your code into (like develop) and your branch (34587\_superAwesomeFeature)

add a description to your PR and a screenshot pinpointing any visual changes

If your PR is not ready to be merged, and you want to get early feedback from other devs, push your changes to branch on your fork, open a PR and add [DO NOT MERGE] in front of PR description: [DO NOT MERGE] #34587/34199 @alberto: Super awesome feature

 Two+ people working on same feature

add other dev's remote

git remote add myBestFriend https://alm-github.systems.uk.hsbc/{devStaffId}/cdd-ui-webapp.git

fetch (update) your info about their repository: \*git fetch myBestFriend

checkout their branch you want to work on

git checkout --track myBestFriend/letsCollaborate

set upstream to your origin

git branch --set-upstream letsCollaborate origin/letsCollaborate

commit and push your changes to your remote (origin)

one of you opens a PR when you're finished

original branch owner can pull your changes by adding your remote and pulling the branch:

git pull thisGuyPushedSomething letsCollaborate

Useful Git commands

**0. Config and check config**

|  |
| --- |
| //this is Important for EOL issue  git config --global core.autocrlf false  git config --global core.safecrlf warn    //add alias to git  git config --global [alias.co](http://alias.co/) checkout  git config --global [alias.ci](http://alias.ci/) commit  git config --global [alias.st](http://alias.st/) status  git config --global [alias.br](http://alias.br/) branch  git config --global alias.hist "log --pretty=format:'%h %ad | %s%d [%an]' --graph --date=short"  git config --global alias.type "cat-file -t"  git config --global alias.dump "cat-file -p"    //check current config and manually edit them in VI  git config --global --edit |

**1. Check remotes, and make sure Origin point to a folk remote**

|  |
| --- |
| git remote -v |

**2. To add remote to the master repo**

|  |
| --- |
| git remote add upstream https://[alm-github.systems.uk](http://alm-github.systems.uk/).hsbc/GlobalStandardsCDD/cdd-ui-webapp.git  //to remove existing remote  git remote remove origin |

**3. Pull from the remote master repo**

|  |
| --- |
| //at least once pull from upstream a day, better to have 3-5 times (more often the better)  git pull upstream develop |

**4. Sync to remote Origin Repo**

|  |
| --- |
| git push origin develop |

**5. Create a new Feature Branch**

|  |
| --- |
| git checkout -b 533501\_CancelReviewReasons  //check local branches  git branch  //check online branches  git branch -r    //Then you can make changes |

**6. Check changes on your local repo**

|  |
| --- |
| git status |

**7. Remove modification**

|  |
| --- |
| git checkout -- xxxxxx |

**8. Add, commit and push files to remote repo**

|  |
| --- |
| git add xxxxxx  git commit  git push origin 533501\_CancelReviewReasons |

**Troubleshooting:**

**1. remove untracked files (cannot undo)**

|  |
| --- |
| git clean -dn  git clean -df |

**2.  undo a committed snapshot by creating a new commit**

|  |
| --- |
| git revert <commit> |

**3.  undo a committed snapshot by removing the subsequent commits**

|  |
| --- |
| git reset HEAD hello.html    git reset <file>  //Remove the specified file from the staging area, but leave the working directory unchanged.  //This unstages a file without overwriting any changes.    git reset  //Reset the staging area to match the most recent commit, but leave the working directory unchanged.  //This  unstages all files without overwriting any changes, giving you the  opportunity to re-build the staged snapshot from scratch.    git reset --hard  //Reset the staging area and the working directory to match the most recent commit.  //In addition to unstaging changes, the --hard flag tells Git to overwrite all changes in the working directory, too.  //Put  another way: this obliterates all uncommitted changes, so make sure you  really want to throw away your local developments before using it.    git reset <commit>  //Move  the current branch tip backward to <commit>, reset the staging  area to match, but leave the working directory alone.  //All changes made since <commit> will reside in the working directory,  //which lets you re-commit the project history using cleaner, more atomic snapshots.    git reset --hard <commit>  //Move the current branch tip backward to <commit> and reset both the staging area and the working directory to match.  //This obliterates not only the uncommitted changes, but all commits after <commit>, as well. |

**4. go back to the previous versions**

|  |
| --- |
| 1) get harsh (first 7 chars):  git hist    git checkout <harsh> |

**5. remove a commit**

please see：<https://githowto.com/undoing_committed_changes>

**6. remove a commit from a branch**

[please see：https://githowto.com/removing\_commits\_from\_a\_branch](https://githowto.com/removing_commits_from_a_branch)

**7. change a commit**

|  |
| --- |
| git add hello.html  git commit --amend -m "Add an author/email comment" |

**8. move files**

|  |
| --- |
| mkdir lib  git mv hello.html lib  git status |

**9. Maven clean install and start Jetty**

|  |
| --- |
| mvn clean install -U  mvn jetty:run -Dstub=false -Denv=familiarisation -Dmdep.skip=true |

**10. fetch and merge**

git fetch origin

git fetch upstream

git commit -m "noStory @anthony: merging develop