# My Git experience with Sutter Health

My git was initially configured with [capitalch@gmail.com](mailto:capitalch@gmail.com). Then I tried to clone sutterRDD enterprise account through [sagarwal@netwoven.com](mailto:sagarwal@netwoven.com). It did not work even if I uninstalled and reinstalled git. Then I checked credential manager from control panel. I removed old credentials. Then I again tried for git clone rddurl and it worked.

# Branch in git

$ git checkout -b [name\_of\_your\_new\_branch]

$ git push origin [name\_of\_your\_new\_branch]

# Cloning from single branch

git clone -b mybranch --single-branch git://sub.domain.com/repo.git

## Another effective way of cloning from a branch

Clone the repository

git clone <repository\_url>

List all branches

git branch -a

Checkout the branch that you want

git checkout <name\_of\_branch>

# Deleting a repository

Append settings to the url of your repository as …/ecomm/settings

You will get a delete area at the bottom.

**Quick bits**

git init

git add README.md

git add -A

git commit -m "first commit"

git remote add origin https://github.com/capitalch/HomeDelivery.git

git push -u origin master

# Start

Created a new account from [capitalch@gmail.com:s...3](mailto:capitalch@gmail.com:s...3)

Create a new repository. This is master branch. You can directly create files / upload in master branch.

Create a branch. Branches are created to incorporate changes without disturbing the master. Branch automatically has everything of master. Make some changes in branch. Then you do pull request in branch then do merge request in branch. At this point the changes of your branch are merged in master.

# Working with git

Use git bash. It is wonderful.

Install git in your system. At command prompt if you type git it will respond. Create a folder say project1 and go in that folder through command prompt. Type:

Git init

Will create a .git hidden folder. In Project1you can add many files. The .git folder is git compressed database. Don’t add files to .git folder. Add files to its parent folder.

Git status will show which files are added / left from the git repository.

Git add \*.\* will add all files to staging.

Git commit –m my message will commit the staging to repository.

git clone <repository URL> will pull in a complete copy of the remote repository to your local system

git branch mybranch will create a branch.

When you create git repository there is only one branch that is master. You do git branch myBranch to create a new branch. This new branch will have everything from master. To switchover between branches you do git checkout myBranch.

## Following a course at <https://www.git-tower.com/learn/git/ebook/en/command-line/basics/what-is-version-control>

To add a remote url:

git remote add origin <https://username@stash/scm/PROJECT/repo.git>

To change the remote url

git remote set-url origin https://github.com/username/repo

Git is version control system. When you make any changes in any of the files and commit them a snapshot is created. You can compare each snapshot. Each snapshot can be called as separate version. So you can track every version.

You can install git from installer. Select git bash from program menu and you will presented linux like command window. It is recommended to first go by git bash and then gui.

Configure Git

$ git config --global user.name "John Doe"

$ git config --global user.email "john@doe.org"

$ git config --global color.ui auto

## Repository and commit

It is git database. You should not touch it. This is .git hidden folder. You can create your own repository or you can clone a remote repository. A commit is wrapper of changes. Author of commit must specify what changes are made in a short ‘commit message’. Every commit is a set of changes, it is a snapshot of project but stored in much efficient manner. Git status gives all the changes since last commit.

You can make many changes in several files but don’t wat to yet include them in git. So you add only intended changes to ‘staging’ area. After adding you need to commit with short description. Then a version is created. The log command shows all the commits made over period of time. Exchange of changes by many people is performed through remote repository.

There are two types of repositories:

Local repository is .git which resides in root folder of your project.

Remote Repository is on internet or network. This has only .git folder and is used to exchange code.

## Working copy

The project root folder is your working copy or working directory. There is only one working copy. you can ask git to replace working copy with any version.

## Ignore files

A list of files to ignore is kept in a simple file called .gitignore in root folder of project. It is recommended to populate this file at beginning before first commit. You should manually create the file .gitignore in root folder and populate it with ignore rules:

path/filename.ext

filename.ext: ignore all files with this name in project

\*.ext, path/\* (to ignore all files in a folder)

## Working

Create a root folder. Provide some files. Do git init. Create .gitignore and then do:

$ git add -A $ git

commit -m "Initial commit"

Clone from a remote directory provided by author

$ cd your/development/folder/ $ git clone https://github.com/gittower/git-crash-course.git

This will download a test project in your working directory. Now you can make any changes in any file, add them and commit them.

There is only one of two status of a file a) tracked, which can be committed. B) untracked, which are new files or in .gitignored.

Any changes you make must be ‘staged’ or to be put in staging area. Only files in staging can be committed. Git status command provides a) Changes to be committed, i.e files in staging b) Changes not staged for commit, i.e newly created files c) Untracked files, i.e in .gitignore

I did:

Git add <files separated by space>, to add to staging

Git reset <files separated by space>, to remove from staging

Git rm <file>, to remove file from working directory

Git add –A, adds all unstaged to staging

If you give command git commit, without –m switch, git will open a configured editor for you.

Git log shows all commits with messages you gave with –m switch. Git log –p shows detailed log.

## My Workshop

* I created a github at <https://github.com/capitalch/ecomm>
* Git clone <https://github.com/capitalch/ecomm>, creates local copy
* Added a few folders containing files in ecomm folder, which is newly created.
* Git add –A, git commit –m “my test commit 1”
* Git push

This updates the github site create by me.

## Branching

This is working with multiple context. With each context of the code you can create a branch. There will be a working branch. Each changes which you apply will be made to the working branch and other branches will be left untouched. Using branch you can work on different things in parallel. By default ‘master’ branch is automatically created by git.

Git branch ecomm-contact

Git checkout ecomm-contact

Above commands will create a new branch ecomm-contact and make that branch as working branch.

Git branch will list all branches along with indication of current branch. Git branch –v will give more details on all branches. Current branch is also called HEAD branch.

## Stash

Saving changes temporarily. Stash is multiple clipboards. It takes all changes from your working copy and save it in clipboard. You are left with a clean working copy. Later on you can copy back from clipboard to your working copy and start at place where you had left. There can be multiple clipboards and none of them are limited to one particular branch. You can copy back to any branch which is HEAD.

Suppose you are working in master. You have made some changes in files and performed git add -A. Now many files are there to be committed. At this point you have to work upon a new branch with a new context. You cannot commit in between. You need to start a new branch with last commit. So you do git stash. This will move all uncommitted changes to stash and change the master state to last commit. Now you can create a branch and do your work. In master branch you can get back the changes by git stash pop. This will clear up the last stash and apply to your HEAD branch. Git stash list will list all stashes.

Git checkout branchname is used to switch over to another branch. At any point of time only one branch is HEAD.

Very important thing to note is: Your working directory files are maintained as per the committed files of HEAD branch. Say for example if branch1 has a committed file myfile1 and master does not have that file then when you switch over to master then the file myfile1 is not available in working folder.

All the above talk is applicable with the committed files. Files which are not committed are shown in all branches. It is highly recommended to commit before switching over or before creating a branch.

## Merge

Git merge branchname. This will merge the HEAD branch with branchname.

## Branching Strategy

There must be one long running branch which is master. Other branches should merge in master. Master branch should also be production.

## Remote repository

This does not have a working copy, just has a .git repository. Remote repository should not be changed directly. Local copy should be modified instead.



Two ways to proceed with remote rep. 1) Create a local repository and from it create a remote rep. 2) Create a remote rep and git clone url, to create a local copy.

If you clone remote rep its remembered as origin in local rep. Otherwise if local rep already exists you need to make it remember the remote rep as follows:

$ git remote add crash-course-remote https://github.

com/gittower/git-crash-course-remote.git

git remote –v will list out.

You can add many remote rep to your local rep. Above command will only put a connection but will not fetch any data. To fetch data: git fetch crash-course-remote

Fetch will not integrate remote data to working copy or branches. It will only show the data.

## Git Guides

**Simple guide**

<http://rogerdudler.github.io/git-guide/>

**Professional ebook**

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