Working with GIT repository

Git is a free & open source, distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Every Git clone is a full-fledged repository with complete history and full revision tracking capabilities, not dependent on network access or a central server. **Branching and merging are fast** and easy to do.

In order to work with GIT you first have to understand and get used to work with multiple branches.

For a better understanding of the branch meaning and philosophy, one can be inspired from here: http://www-cs-students.stanford.edu/~blynn/gitmagic/ch04.html

This document examples are based on SIPfoundry's repository on git-hub: http://github.com/SIPfoundry/sipxecs

Working with GIT repository in local mode

A developer should be aware of three things in order to locally work with GIT repositories:

1. Download source code from the repository
2. Make your changes
3. Generate patches

There are two repositories involved: the remote repository (Douglas's on git-hub called **origin** repository) and your copy, called **local** repository

Download source code from the repository

Sources have to be downloaded in read-only mode on a chosen path on local computer, so there is no danger to overwrite the original code:

```
git clone git://github.com/SIPfoundry/sipxecs.git
```

Example:

```
[user@localhost~]$ mkdir work
[user@localhost~]$ cd work
[user@localhost]$ git clone git://github.com/SIPfoundry/sipxecs.git
Initialized empty Git repository in /home/mirceac/work/sipxecs/.git/remote:
Counting objects: 127119, done.remote:
Compressing objects: 100% (29946/29946), done.remote:
Total 127119 (delta 84094), reused 126491 (delta 83609)Receiving objects: 100% (127119/127119), 751.56 MiB |
1.69 MiB/s, done.
Resolving deltas: 100% (84094/84094), done.Checking out files: 100% (11021/11021), done.
[user@localhost]$ ls
sipxecs
```

Now, verify to what branch you are positioned using the following command:

```
git status
```

Use the following command to list all visible branches

```
git branch
```

The origin repository has many branches and you have to relay your work on one of them: master Sometimes this branch is hidden and you have to make it visible

Use the following command to list all branches

```
git branch -a
```

How to make branch master visible and be positioned on it

```
git checkout -b master origin/master
```

Watch the following code-snipped to accomplish this

Example:

```
[user@localhostsipxecs]$ git branch
* master
[user@localhostsipxecs]$ git branch -a
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
  remotes/origin/release-4.2
[user@localhostsipxecs]$ git checkout -b master origin/master
Branch master set up to track remote branch master from origin.
Switched to a new branch 'master'
[user@localhostsipxecs]$ git status
# On branch master
nothing to commit (working directory clean)
[user@localhostsipxecs]$ git branch
  master
* master
[user@localhostsipxecs]$
```

Make your changes

Before starting working, you have to make sure that you are in sync with the latest source code. Assuming that you have to base your work on master branch follow the steps from below:

ATTENTION: do not make any change or commit on master branch. Always create a new branch for your changes

Run the commands from below:

```
git fetch origin //make sure that you are in sync with the remote repository (origin)
git checkout master //move to master branch, because we want to base our work on this
git rebase origin/master //rebase your master branch with the origin master branch just to keep in sync with
the latest source code
git checkout -b TEST //create an position to a new branch called TEST, the branch were your changes will be
performed
```

TIP: if there is a branch you don't like you can always delete it using:

```
git branch -D TEST
```

...after all your changes are done and you are prepared for the final step: generate patch, run the following commands

```
git status //to verify what files are new and what are changed
git add <file_path> //if there are newly created files on your branch
git commit -a //commit your work

use git commit -a --amend if you want that your changes to be on top on a previous commit - in this way all
will be included in a single patch
```

NOTE: for every commit will be generated one single patch (unless git commit -a --amend was used)

When git commit -a is used a vi editor is opened. Please the comments about the patch there and use :wq to actually write the changes (perform the commit)

git commt -a --amend will preserve previous commit comments and changes

```
TEST: Test Title

Description
# Please enter the commit message for your changes. Lines starting# with '#' will be ignored, and an empty
message aborts the commit.
# Committer: Mircea Carasel
# On branch TEST
# Changes to be committed:
#
```