Git workflow

In jclouds we try hard to keep a clean commit history and try to avoid having merge commits. To achieve this, we require all pull requests to be rebased to the latest version of master, and to have a clean commit list (or ideally just one single squashed commit) without merge commits.

The following Git workflow is just a proposal so users can follow it when creating pull requests, to make sure we will be able to merge it without issues.

Prepare your fork

The first thing to do is to make sure you have the jclouds Git repository configured as a remote. In this case we will add it as a remote called "jclouds":

```
cd jclouds
git remote add jclouds https://git-wip-us.apache.org/repos/asf/jclouds.git
```

Create the feature branch

When beginning working on the feature, take a branch from the latest master version:

```
git checkout master
git fetch jclouds
git reset --hard jclouds/master
git checkout -b feature-branch
```

As we want a clean master and we should never use directly use the master branch (it should be only used to sync the fork with the upstream repo), so we can just reset hard it instead of rebasing.

Develop the feature and address the review comments

Develop the feature normally, commit the changes (you can create N commits if needed), and open the pull request.

Once the pull request is open and has been reviewed, address any review comments and add the changes **in new commits**. This will make it easier for the reviewer to focus only in the last changes without having to go again through the entire pull request. Once the PR is ready to be merged, you'll be asked to squash the commits.

Preparing the pull request to be merged

First squash the commits, to have a cleaner rebase (it is easier to rebase having one single commit than having N commits):

```
git rebase -i HEAD~N     # Where N is the number of commits in the pull request
```

Now that there is only one commit, update the branch to the latest version of master:

```
git fetch jclouds
git rebase jclouds/master
```

At this point the local branch with one single commit on top of the latest master and it can be pushed to your fork:

```
git push <your-fork's-remote> feature-branch -f   # You might need to push with --force because of the rebase
to the latest master
```

Now the pull request will be clean with one single commit and up to date with the latest master. Ready to be merged!

A note about keeping the pull request up to date

The key point is to *avoid* updating your feature branch with master until the PR is ready to be merged (or until you're asked to rebase). This way you'll avoid accidental merge commits and fixing conflicts when you shouldn't be dealing with that.
If for you want to keep your branch up to date with the latest master changes anyway, always rebase it; don't git pull or introduce merge commits, as it will complicate the rebasing process and will make it difficult to merge the pull request in a clean way