**Multiple remote repositories steps**

This are the following steps that explain the multiple remote configuration.

Before all the code lets explain the endpoints and definitions:

* *oss\_repository*: the local folder where the code and .git configuration are.
* *bitbucket.org:onlyangelred/oss\_repository* : the remote repository for the main code (internal and private)
* *github.com/onlyangel/oss\_repository* : public repository
* local repository: is the repository located in a local machine

**Set up**

First we need to create a folder for the project code. There all the development code is in. And also all the information from git resides. And enter the folder.

mkdir oss\_repository

cd oss\_repository

Let’s create an empty repository in the local repository.

git init

Now let’s add a fist file as content and add it to the local repository.

echo "# oss\_repository" >> README.md

git add .

git commit -m "Fist commit"

At this point the local repository have only one commit. But there is no presence in any remote service. To do that let’s name the remote repositories in the local repository.

* **origin**: *bitbucket.org:onlyangelred/oss\_repository*
* **github**: *github.com/onlyangel/oss\_repository*

And then add them to the local repository as follow:

git remote add origin git@bitbucket.org:onlyangelred/oss\_repository.git

git remote add github git@github.com:onlyangel/oss\_repository.git

**Note**: as you can see the repository url's are in the shape of ssh authentication for security reasons. Git have the ability of share the public key of a ssh local key with the remote services. Making an automatic identification/authentication of the user.

In this moment we have 2 *empty remotes* and a local repository with just one commit in its *master* (only) branch.

**Upload to one repository**

For example lets add some content to the local repository:

echo "Second commit" >> second.md

echo "second commit 3th file" >> 3th.md

git add .

git commit -m "second commit"

and then push it (the *master* branch) to the *origin* (bitbucket) repository.

git push origin master

**Synchronization of repositories**

Let’s imagine that another user (with commit permission) has added other commits to the origin remote repository and we don't have those commits in our local repository, to synchronize both remotes we need to "copy" the content of one into another.

The correct way to implement this "copy" procedure is by implementing a series of steps that orchestrated the copy of the via a local repository.

As an example we are going to pull the data from the origin to the github remote repository with the following steps:

1. Sync the origin remote repository with the local repository:

git fetch origin

This will fetch all the remote branches and tags to our local index

1. Once the local repository has all the data from the remote origin we need to send it to the github remote. With all the tags

git push --all github

git push --tags github

There are a few new ways to do this with the new version of Git but it has to be tested yet. With this approach I recommend to have a specific local folder for this local repository. And avoid to make testing or development in the code that holds the working directory inside this local repository, this to avoid the corruption of the states of the remote repositories.

**External contributors**

If an external developer wants to contribute to our public facing repository they have to do it via pull requests.

The steps are the following: 1. Fist the external user have to fork our public repository @ github named: github.com/*onlyangel*/oss\_repository 2. In its own fork of the repository (ej. github.com/*externaldeveloper*/oss\_repository) 3. Then the developer creates an new branch from the branch he want to modify. 4. In it new branch he makes all the progress and changes. 5. Once he finishes the changes he starts a Pull Request to our public repository. 6. For this in github.com he selects it branch (where he does all the work) and then selects the button that says "New Pull Request"

This are all the steps for an external contributor for each change or feature that they want to build.

After that, the manager need to approve the pull request. For this exists an interface that let the administrator review the requests, where he can see the comments on commits, identify the changes in files, identify if the pull request break something in the current head of the project. And even list all the contributors that helped to make that Request. With all this information the manager can clone the branch that is about to be merged to make some testing in it before it gets to the master branch in the public repository.

**Security**

The security is in charge of the git underline infrastructure and its support for ssh connections between the client and the servers. In this case we will encourage to the users to use this methodology to authenticate with the servers, because it has more advantages than the only usage of user and passwords.