Adding multiple remotes

When you do `git init`, you initialize a local Git repository. In general, the purpose is to synchronize this repo with a remote Git repo. To be able to synchronize code with a remote repo, you need to specify where the remote repo exists.

The first step is to add remote repos to your project.

```
# Syntax to add a git remote
git remote add REMOTE-ID REMOTE-URL
```

By convention, the original / primary remote repo is called `origin`. Here's a real example:

```
# Add remote 1: GitHub.
git remote add origin git@github.com:jigarius/toggl2redmine.git
# Add remote 2: BitBucket.
git remote add upstream git@bitbucket.org:jigarius/toggl2redmine.git
```

In the above example, we add the remote repository of a project called Toggl 2 Redmine found on GitHub. Use the above command to add one or more remote Git repos – make sure that each repo has its unique ID, i.e. `origin`, `upstream` in the above example.

Configure primary remote

Though you can add multiple remotes, usually, each branch of your project can be configured to track a single remote branch. You can setup a branch to track a remote branch as follows:

```
# Change local branch.
git checkout BRANCH
# Configure local branch to track a remote branch.
git branch -u origin/BRANCH
```

Here, `BRANCH` is the name of the remote branch, which is usually the same as your local branch.

Change remote URL

If you want to change the URL associated to a remote that you've already added, you can do it with the following command:

```
# The syntax is: git remote set-url REMOTE-ID REMOTE-URL
git remote set-url upstream git@foobar.com:jigarius/toggl2redmine.git
```

List all remotes

To see a list of all remotes, simply use the following command:

```
$git remote -v
origin      git@github.com:jigarius/toggl2redmine.git (fetch)
origin      git@github.com:jigarius/toggl2redmine.git (push)
upstream    git@bitbucket.org:jigarius/toggl2redmine.git (fetch)
upstream    git@bitbucket.org:jigarius/toggl2redmine.git (push)
```

Remove a remote

If you've added a remote which you no longer require, you can remove it as follows:

```
# The syntax is: git remote remove REMOTE-ID
git remote remove upstream
```

Push to multiple remotes

Now that you have a primary remote repo and other remotes as well, it's time to configure the push. The objective is to push to multiple Git remotes with a single `git push` command.

To do this, choose a remote ID which will refer to all the remotes. I usually call it `all`, but there are developers who prefer `origin`. The idea is to add all the remote repo URLs as "push URLs" to this remote. Here's what you do:

```
# Create a new remote called "all" with the URL of the primary repo.
git remote add all git@github.com:jigarius/toggl2redmine.git
# Re-register the remote as a push URL.
git remote set-url --add --push all
git@github.com:jigarius/toggl2redmine.git
# Add a push URL to a remote. This means that "git push" will also push
```

```
to this git URL.
git remote set-url --add --push all
git@bitbucket.org:jigarius/toggl2redmine.git
```

If you don't want to create an extra remote named `all`, you can skip the first command and use the remote `origin` instead of `all` in the subsequent command(s).

Now, you can push to all remote repositories with a single command!

```
# Replace BRANCH with the name of the branch you want to push.
git push all BRANCH
```

Pull from multiple remotes

It is not possible to `git pull` from multiple repos. However, you can `git fetch` from multiple repos with the following command:

```
git fetch --all
```

This will *fetch* information from all remote repos. You can switch to the latest version of a branch on a particular remote with the command:

```
# Checkout the branch you want to work with.
git checkout BRANCH
# Reset the branch to match the state as on a specific remote.
git reset --hard REMOTE-ID/BRANCH
```

Conclusion

It is easy to synchronize code between multiple git repositories, especially, pushing to multiple remotes. This is helpful when you're maintaining mirrors / copies of the same repository. All you need to do is set up multiple push URLs on a remote and then perform `git push` to that remote as you usually do.