Setting up git to use a client-server workflow

Setting up a new local repository

First, start using git in your local directory (skip this if you already have a repository):

```bash
cd root_of_project
git init
git add .
git commit -a -m 'initial commit'
```

Setting up a new repository on the server

Then set up an empty repository on your server (skip this if you are given a fresh server repository):

```bash
ssh user@server
mkdir project.git
cd project.git
git --bare init
```

The `--bare` tells it to make the server just a repository with no working directory tree.

Linking the client to the server

Then back on the client side, link the two.

If your repository was already linked to a server, first sever that link:

```bash
git remote rm origin
```

If not, skip that step. In either case, link the client to the new server:

```bash
git remote add origin ssh://user@server/~/project.git
```

Finally, push your local repository to the server. The first time you do this, use the command:

```bash
git push -u origin master
```

Using the server repository

Now you are set up. From now on, use “```git push```” to push (committed) changes to the server, and “```git pull```” to merge server changes into your local copy. To start another client, go to where you want it and use:

```bash
git clone ssh://user@server/~/project.git [local directory name]
```

If you don’t specify `[local directory name]`, it will copy the name from the server.

From then on I mainly use, “```git status```”, “```git add```”, “```git commit -a```”, “```git push```” and “```git pull```”. Every few months, you can throw in a “```git repack && git gc```” for good measure.