Resource R3 Installing and configuring `git`

This resource will help you install and configure `git` on your machine. It assumes you are using Ubuntu or some similar OS. It also takes you through setting up your own `git` repository for your ROS packages!

Resource R3.1 Installing `git`

Open a Terminal window. Update `apt`itude.

```
sudo apt update
```

Install `git`.

```
sudo apt install git
```

Check that it is correctly installed.

```
git --version
```

```
git version 2.26.0
```

The specific version of `git` isn't important.

Resource R3.2 Configuring `git`

Set your name and email.

```
git config --global user.name "Your Name"
git config --global user.email "youremail@yourdomain.com"
```

These are stored in `~/.gitconfig`. Change them there, as desired.

Resource R3.3 Setting up GitHub

GitHub (`github.com`) is a place to remotely host a `git` repository. A remote host such as this is important for backup, sharing, and collaboration with `git`. Create a GitHub account here:

`github.com/join`.

To avoid having to re-enter your GitHub username and password frequently, consider setting up an SSH key from the following article:

`git.io/Jeo3f`.

Resource R3.4 Create your own package repo

You'll want a repository for your ROS repositories. Create a new repository in GitHub:

`git.io/JeDDp`.

Consider using the following options.

- ☑ Initialize this repository with a README[1]
- Select[2] Add .gitignore: ROS

Now you can `clone` this repo by navigating to it in the web interface and copying the URL provided by the green Clone or download button.[3]

Open a Terminal window and `cd` to a ROS workspace's `src` directory. Clone your remote repo with the following.

```
git clone <copied repo URL>
```

If this is successful, you should now have local copy of your repository in the current directory.

Now set up your package repository with `catkin_create_pkg`, as we learned in Lecture 06.02.3.1. Once your package is created, stage your changes for commit. First, see which files have changed.

```
git status
```

Stage all changes for commit.

```
git add -A # careful with this
```

Commit changes.

```
git commit -m 'created a package'
```

Now we can push these changes up to the remote repo.

```
git push
```

[1] It is a good idea to have a README in every `git` repository. GitHub makes this easy and even renders it in a nice format on the website.

[2] The `.gitignore` file includes a list of extensions for `git` to ignore in your repository. It is convenient that there is a default one for ROS.

[3] If you have set up SSH, use the SSH URL. Otherwise, use the HTTPS URL.

If it fails, it will probably suggest you set `remote` as `upstream` with the `set-upstream` option. If so, just copy/paste the suggestion and try it.

You now have a `git` repository for your ROS packages!

Resource R3.5 Forking the book code repository

Go to this book's GitHub code repository:

`github.com/ricopicone/robotics-book-code`.

On the upper-right, click the Fork button. This will give you a copy of the repository in *your* GitHub account. Now you can `clone` this fork by navigating to it in the web interface and copying the URL provided by the green Clone or download button.[4]

Open a Terminal window and `cd` to a ROS workspace's `src` directory. Clone your remote repo with the following.

```
git clone <copied repo URL>
```

Follow the directions in the README to use `catkin_make` to make the repo packages available to ROS.

Resource R3.6 Updating from the original repo

If you'd like to bring updates to the book's GitHub repo into your fork, first add it as an upstream.

```
git remote add upstream \
  https://github.com/ricopicone/robotics-book-code.git # or ssh
```

Fetch remote branches.

```
git fetch upstream
```

Make sure you're working on your `master` branch.

```
git checkout master
```

Now merge your and my `master` branches.

```
git rebase upstream/master
```

[4] If you have set up SSH, use the SSH URL. Otherwise, use the HTTPS URL.