Using Git with Repl.it: A Short Guide

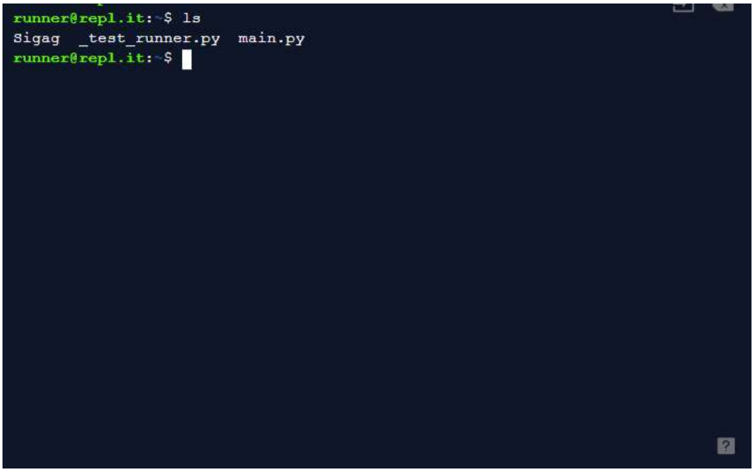
I stumbled upon this post, which described a method to access Git commands from within your repl. Using a Version Control System (VCS) like Git is incredibly useful, and even more so when augmented with GitHub.

In the post, the accepted answer recommended using the `os` Python module and accessing system commands from there.

```
import os
os.system('git clone https://github.com/EanKeen/Sigag')
os.chdir('./Sigag')
os.system('git status')
```

I created a little repl that demonstrates this. Make sure you delete the `Sigag` directory before starting the program (although it's not a strict requirement). After the clone has finished, I'm able to leave the repl, reopen the repl, and have the Git repository still there.

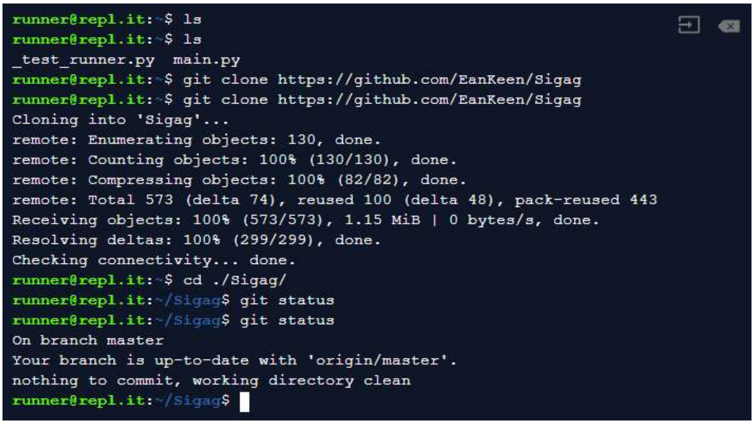
However, there is a much easier way to use Git commands. In most repls, you're able to enter the shell. Press `F1`, and type `shell`. Note that with some keyboards, you may need to press `Fn + F1`. (You can also press Ctrl+Shift+p - thanks @ArchieMaclean!)



```
runner@repl.it:~$ ls
Sigag  _test_runner.py  main.py
runner@repl.it:~$
```

Now, you can just clone it the usual way.

```
git clone https://github.com/EanKeen/Sigag
cd ./Sigag
git status
```



```
runner@repl.it:~$ ls
runner@repl.it:~$ ls
_test_runner.py  main.py
runner@repl.it:~$ git clone https://github.com/EanKeen/Sigag
runner@repl.it:~$ git clone https://github.com/EanKeen/Sigag
Cloning into 'Sigag'...
remote: Enumerating objects: 130, done.
remote: Counting objects: 100% (130/130), done.
remote: Compressing objects: 100% (82/82), done.
remote: Total 573 (delta 74), reused 100 (delta 48), pack-reused 443
Receiving objects: 100% (573/573), 1.15 MiB | 0 bytes/s, done.
Resolving deltas: 100% (299/299), done.
Checking connectivity... done.
runner@repl.it:~$ cd ./Sigag/
runner@repl.it:~/Sigag$ git status
runner@repl.it:~/Sigag$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
runner@repl.it:~/Sigag$
```

Once the clone has finished, you should see the `Sigag` directory in your file tree!

However, the output of `ls` and your file tree may be different sometimes. For example, I would type `ls` into the shell, and it will show `Sigag` as a directory. However, my file tree would only show `main.py`. To fix this, simply refresh the page.

It may seem a bit convoluted getting this to work, but easier methods of using git will be introduced at a later date, according to the post below. The screenshot below was taken on the publish date of this guide.