# Development Guide

This is a guide for developers who would like to contribute to this project.

If you're interested in contributing to mycli, thank you. We'd love your help!

You'll always get credit for your work.

## GitHub Workflow

1. [Fork the repository](https://github.com/dbcli/mycli) on GitHub.

2. Clone your fork locally:

```bash

$ git clone <url-for-your-fork>

```

3. Add the official repository (`upstream`) as a remote repository:

```bash

$ git remote add upstream git@github.com:dbcli/mycli.git

```

4. Set up a [virtual environment](http://docs.python-guide.org/en/latest/dev/virtualenvs)

for development:

```bash

$ cd mycli

$ pip install virtualenv

$ virtualenv mycli\_dev

```

We've just created a virtual environment that we'll use to install all the dependencies

and tools we need to work on mycli. Whenever you want to work on mycli, you

need to activate the virtual environment:

```bash

$ source mycli\_dev/bin/activate

```

When you're done working, you can deactivate the virtual environment:

```bash

$ deactivate

```

5. Install the dependencies and development tools:

```bash

$ pip install -r requirements-dev.txt

$ pip install --editable .

```

6. Create a branch for your bugfix or feature based off the `master` branch:

```bash

$ git checkout -b <name-of-bugfix-or-feature> master

```

7. While you work on your bugfix or feature, be sure to pull the latest changes from `upstream`. This ensures that your local codebase is up-to-date:

```bash

$ git pull upstream master

```

8. When your work is ready for the mycli team to review it, push your branch to your fork:

```bash

$ git push origin <name-of-bugfix-or-feature>

```

9. [Create a pull request](https://help.github.com/articles/creating-a-pull-request-from-a-fork/)

on GitHub.

## Running the Tests

While you work on mycli, it's important to run the tests to make sure your code

hasn't broken any existing functionality. To run the tests, just type in:

```bash

$ ./setup.py test

```

Mycli supports Python 2.7 and 3.4+. You can test against multiple versions of

Python by running tox:

```bash

$ tox

```

### Test Database Credentials

The tests require a database connection to work. You can tell the tests which

credentials to use by setting the applicable environment variables:

```bash

$ export PYTEST\_HOST=localhost

$ export PYTEST\_USER=user

$ export PYTEST\_PASSWORD=myclirocks

$ export PYTEST\_PORT=3306

$ export PYTEST\_CHARSET=utf8

```

The default values are `localhost`, `root`, no password, `3306`, and `utf8`.

You only need to set the values that differ from the defaults.

### CLI Tests

Some CLI tests expect the program `ex` to be a symbolic link to `vim`.

In some systems (e.g. Arch Linux) `ex` is a symbolic link to `vi`, which will

change the output and therefore make some tests fail.

You can check this by running:

```bash

$ readlink -f $(which ex)

```

## Coding Style

Mycli requires code submissions to adhere to

[PEP 8](https://www.python.org/dev/peps/pep-0008/).

It's easy to check the style of your code, just run:

```bash

$ ./setup.py lint

```

If you see any PEP 8 style issues, you can automatically fix them by running:

```bash

$ ./setup.py lint --fix

```

Be sure to commit and push any PEP 8 fixes.