Lesson 0 - Environment Setup

March 28, 2020

Lesson 0: Environment Setup

You will need the following:

- Linux (Ubuntu 18.04)
- Python
- Virtualenv
- Code Editor
- Git

- 1. Linux (Ubuntu 18.04)

– 1.1 Setting up your Ubuntu 18.04 environment

— Option A: Ubuntu 18.04 on your local machine If you're already using Ubuntu 18.04 on your local machine, you can use that.

— Option B: Ubuntu 18.04 on AWS Cloud9 If you don't have Ubuntu 18.04 on your local machine, you can use AWS Cloud 9 https://aws.amazon.com/cloud9/

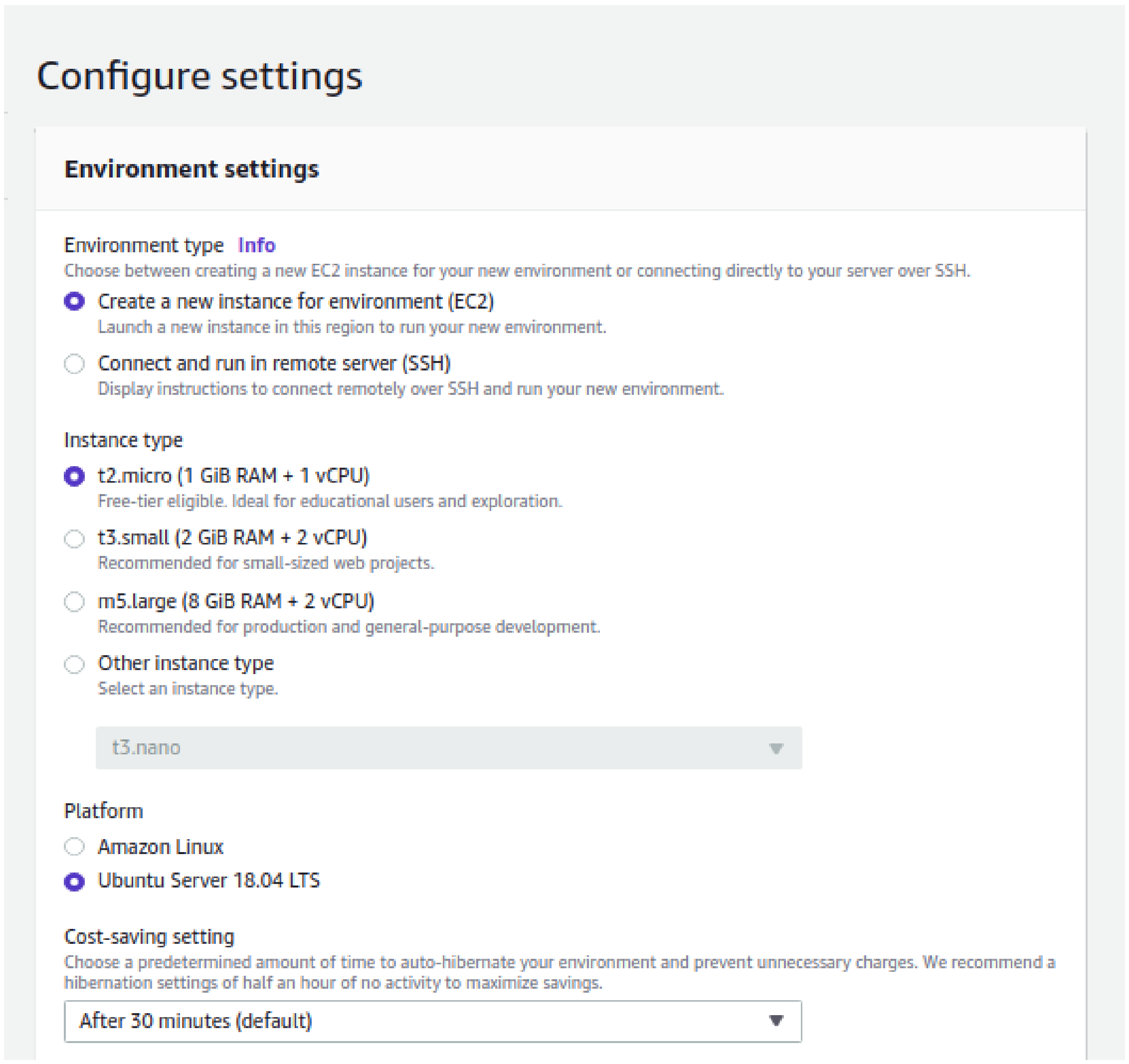
AWS Cloud9 is a cloud-based IDE and an easy way for you to try working on Linux (Ubuntu) environment.

If you don't have an AWS account yet, you can create a free-tier account here https://aws.amazon.com/free/

Create an AWS Cloud9 development environment.

Be sure to choose the following configurations:

- Environment type: EC2
- Instance type: t2.micro
- Platform: Ubuntu Server 18.04


Configure settings

Environment settings

Environment type Info

Choose between creating a new EC2 instance for your new environment or connecting directly to your server over SSH.

[✓ Create a new instance for environment (EC2)]
Launch a new instance in this region to run your new environment.

Connect and run in remote server (SSH)
Display instructions to connect remotely over SSH and run your new environment.

Instance type

[✓ t2.micro (1 GiB RAM + 1 vCPU)]
Free-tier eligible. Ideal for educational users and exploration.

t3.small (2 GiB RAM + 2 vCPU)
Recommended for small-sized web projects.

m5.large (8 GiB RAM + 2 vCPU)
Recommended for production and general-purpose development.

Other instance type
Select an instance type.

t3.nano

Platform

Amazon Linux

[✓ Ubuntu Server 18.04 LTS]

Cost-saving setting

Choose a predetermined amount of time to auto-hibernate your environment and prevent unnecessary charges. We recommend a hibernation settings of half an hour of no activity to maximize savings.

After 30 minutes (default)

– 1.2 Linux Command Line

We highly recommend being familiar with **Linux Command Line**. You may refer to this tutorial https://tutorial.djangogirls.org/en/intro_to_command_line

2. Python

On Ubuntu Linux, it's very likely that you already have Python 3 installed out of the box.

To check if you already have Python, open a console and type the following command:

(command-line)

```
$ python3 --version
Python 3.6.8
```

3. Virtualenv

Virtualenvs allow you to create sandboxed Python environments.

Python 3 already ships with virtualenv under the module name of `venv`.

As you start working on multiple projects, your projects will have different dependencies (different package versions) and would need different virtual environment for each project.

`Virtualenv` helps you create an isolated environment specific to the project you're working on.

- 3.1 Creating a Virtualenv

First, we need to create a directory to store our virtualenvs called `venvs` (command-line)

```
ubuntu:~/environment $ mkdir venvs
```

To create a virtualenv, run the following command:

```
$ python3 -m venv [PATH FOR VIRTUALENV]
```

(command-line)

```
ubuntu:~/environment $ python3 -m venv venvs/pyscripting
```

- 3.2 Activating a Virtualenv

To activate a virtualenv run the following command

```
$ source [PATH FOR VIRTUALENV]/bin/activte
```

(command-line)

```
ubuntu:~/environment $ source venvs/pyscripting/bin/activate
(pyscripting) ubuntu:~/environment $
```

Notice that the prompt changed to indicate what virtualenv is currently active.

With the virtualenv activated, the python and pip binaries point to the local Python 3 variations, so we don't need to append the `3` or `3.x` to our python commands

(command-line)

```
(pyscripting) ubuntu:~/environment $ which python
/home/ubuntu/environment/venvs/pyscripting/bin/python
(pyscripting) ubuntu:~/environment $ python --version
Python 3.6.9
(pyscripting) ubuntu:~/environment $ pip list
Package       Version
------------- -------
pip           9.0.1
pkg-resources 0.0.0
setuptools    39.0.1
(pyscripting) ubuntu:~/environment $
```

- 3.3 Deactivating a Virtualenv

To deactivate the virtualenv, just run the `deactivate` command:

(command-line)

```
(pyscripting) ubuntu:~/environment $ deactivate
ubuntu:~/environment $
```

4. Code Editor (Optional)

If you're going to try coding on your personal machine, we recommend installing a code editor - Sublime Text 3.

Sublime Text is a very popular editor with a free evaluation period and it's available for all operating systems.

You can download it here: https://www.sublimetext.com/3

5. Git

- 5.1 Installing Git

For Linux users, if you are using Debian/Ubuntu, you can install Git using the following commands:

(command-line)

```
$ sudo apt update
$ sudo apt install git
```

- 5.1 Checking if you have Git

To check if you have git on your computer, go to your terminal/cmd and type: `git`

You should get something similar to this:

```
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone      Clone a repository into a new directory
   init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add        Add file contents to the index
   mv         Move or rename a file, a directory, or a symlink
   reset      Reset current HEAD to the specified state
   rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect     Use binary search to find the commit that introduced a bug
   grep       Print lines matching a pattern
   log        Show commit logs
   show       Show various types of objects
   status     Show the working tree status

grow, mark and tweak your common history
   branch     List, create, or delete branches
   checkout   Switch branches or restore working tree files
   commit     Record changes to the repository
   diff       Show changes between commits, commit and working tree, etc
   merge      Join two or more development histories together
   rebase     Reapply commits on top of another base tip
   tag        Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch      Download objects and refs from another repository
   pull       Fetch from and integrate with another repository or a local branch
   push       Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
```