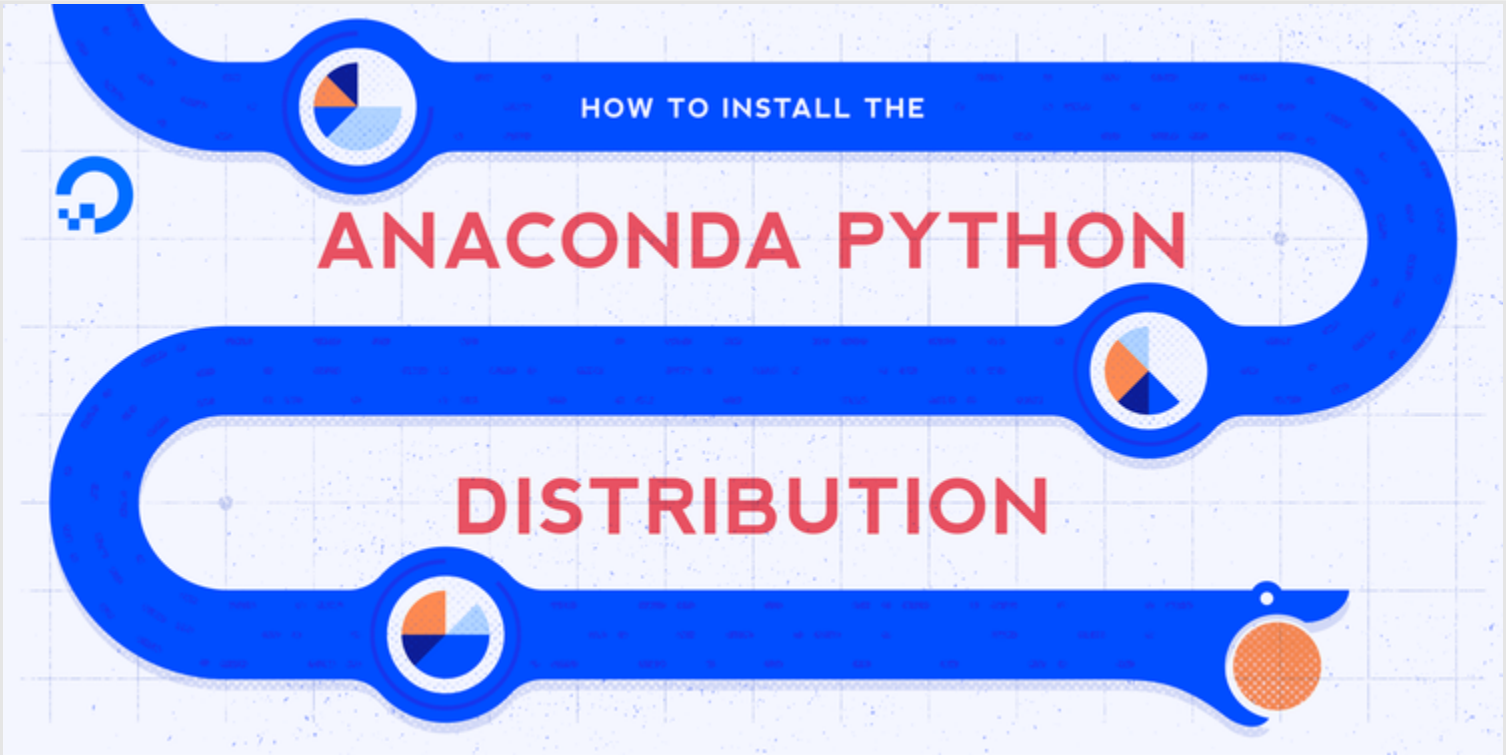

How To Install the Anaconda Python Distribution on Ubuntu 16.04

Updated December 27, 2017 610.3k DEVELOPMENT PYTHON DATA ANALYSIS UBUNTU 16.04

By: Lisa Tagliaferri

Not using **Ubuntu 16.04**? Choose a different version:

Introduction

Anaconda is an open-source package manager, environment manager, and distribution of the Python and R programming languages. It is commonly used for large-scale data processing, scientific computing, and predictive analytics, serving data scientists, developers, business analysts, and those working in DevOps.

Anaconda offers a collection of over 720 open-source packages, and is available in both free and paid versions. The Anaconda distribution ships with the `conda` command-line utility. You can learn more about Anaconda and `conda` by reading the Anaconda Documentation pages.

This tutorial will guide you through installing the Python 3 version of Anaconda on an Ubuntu 16.04 server.

Prerequisites

Before you begin with this guide, you should have a non-root user with sudo privileges set up on your server. You can learn how to do this by completing our Ubuntu 16.04 initial server setup guide.

Installing Anaconda

The best way to install Anaconda is to download the latest Anaconda installer bash script, verify it, and then run it.

Find the latest version of Anaconda for Python 3 at the Anaconda Downloads page. At the time of writing, the latest version is 5.0.1, but you should use a later stable version if it is available.

Next, change to the `/tmp` directory on your server. This is a good directory to download ephemeral items, like the Anaconda bash script, which we won't need after running it.

```
$ cd /tmp
```

Use `curl` to download the link that you copied from the Anaconda website:

```
$ curl -O https://repo.continuum.io/archive/Anaconda3-5.0.1-Linux-x86_64.sh
```

We can now verify the data integrity of the installer with cryptographic hash verification through the SHA-256 checksum. We'll use the `sha256sum` command along with the filename of the script:

```
$ sha256sum Anaconda3-5.0.1-Linux-x86_64.sh
```

You'll receive output that looks similar to this:

```
Output
55e4db1919f49c92d5abbf27a4be5986ae157f074bf9f8238963cd4582a4068a  Anaconda3-5.0.1-Linux-x86_64.sh
```

You should check the output against the hashes available at the Anaconda with Python 3 on 64-bit Linux page for your appropriate Anaconda version. As long as your output matches the hash displayed in the `sha2561` row then you're good to go.

Now we can run the script:

```
$ bash Anaconda3-5.0.1-Linux-x86_64.sh
```

You'll receive the following output:

```
Output

Welcome to Anaconda3 5.0.1 (by Continuum Analytics, Inc.)

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
```

Press `ENTER` to continue and then press `ENTER` to read through the license. Once you're done reading the license, you'll be prompted to approve the license terms:

```
Output
Do you approve the license terms? [yes|no]
```

As long as you agree, type `yes`.

At this point, you'll be prompted to choose the location of the installation. You can press `ENTER` to accept the default location, or specify a different location to modify it.

```
Output
Anaconda3 will now be installed into this location:
/home/sammy/anaconda3

  - Press ENTER to confirm the location
  - Press CTRL-C to abort the installation
  - Or specify a different location below

[/home/sammy/anaconda3] >>>
```

The installation process will continue, it may take some time.

Once it's complete you'll receive the following output:

```
Output
...
installation finished.
Do you wish the installer to prepend the Anaconda3 install location
to PATH in your /home/sammy/.bashrc ? [yes|no]
[no] >>>
```

Type `yes` so that you can use the `conda` command. You'll next see the following output:

```
Output
Prepending PATH=/home/sammy/anaconda3/bin to PATH in /home/sammy/.bashrc
A backup will be made to: /home/sammy/.bashrc-anaconda3.bak
...
```

In order to activate the installation, you should source the `~/.bashrc` file:

```
$ source ~/.bashrc
```

Once you have done that, you can verify your install by making use of the `conda` command, for example with `list`:

```
$ conda list
```

You'll receive output of all the packages you have available through the Anaconda installation:

```
Output
# packages in environment at /home/sammy/anaconda3:
#
_ipyw_jlab_nb_ext_conf    0.1.0            py36he11e457_0
alabaster                 0.7.10           py36h306e16b_0
anaconda                  5.0.1            py36hd30a520_1
...
```

Now that Anaconda is installed, we can go on to setting up Anaconda environments.

Setting Up Anaconda Environments

Anaconda virtual environments allow you to keep projects organized by Python versions and packages needed. For each Anaconda environment you set up, you can specify which version of Python to use and can keep all of your related programming files together within that directory.

First, we can check to see which versions of Python are available for us to use:

```
$ conda search "^python$"
```

You'll receive output with the different versions of Python that you can target, including both Python 3 and Python 2 versions. Since we are using the Anaconda with Python 3 in this tutorial, you will have access only to the Python 3 versions of packages.

Let's create an environment using the most recent version of Python 3. We can achieve this by assigning version 3 to the `python` argument. We'll call the environment my_env, but you'll likely want to use a more

descriptive name for your environment especially if you are using environments to access more than one version of Python.

```
$ conda create --name my_env python=3
```

We'll receive output with information about what is downloaded and which packages will be installed, and then be prompted to proceed with `y` or `n`. As long as you agree, type `y`.

The `conda` utility will now fetch the packages for the environment and let you know when it's complete.

You can activate your new environment by typing the following:

```
$ source activate my_env
```

With your environment activated, your command prompt prefix will change:

```
(my_env) sammy@ubuntu:~$
```

Within the environment, you can verify that you're using the version of Python that you had intended to use:

```
(my_env) sammy@ubuntu:~$  python --version
```

```
Output
Python 3.6.0 :: Continuum Analytics, Inc.
```

When you're ready to deactivate your Anaconda environment, you can do so by typing:

```
(my_env) sammy@ubuntu:~$ source deactivate
```

Note that you can replace the word `source` with `.` to achieve the same results.

To target a more specific version of Python, you can pass a specific version to the `python` argument, like `3.5`, for example:

```
$ conda create -n my_env35 python=3.5
```

You can update your version of Python along the same branch (as in updating Python 3.5.1 to Python 3.5.2) within a respective environment with the following command:

```
(my_env35) sammy@ubuntu:~$ conda update python
```

If you would like to target a more specific version of Python, you can pass that to the `python` argument, as in `python=3.3.2`.

You can inspect all of the environments you have set up with this command:

```
$ conda info --envs
```

```
Output
# conda environments:
#
my_env                   /home/sammy/anaconda3/envs/my_env
my_env35                 /home/sammy/anaconda3/envs/my_env35
root                  *  /home/sammy/anaconda3
```

The asterisk indicates the current active environment.

Each environment you create with `conda create` will come with several default packages:

- `openssl`
- `pip`
- `python`
- `readline`
- `setuptools`
- `sqlite`
- `tk`
- `wheel`
- `xz`
- `zlib`

You can add additional packages, such as `numpy` for example, with the following command:

```
$ conda install --name my_env35 numpy
```

If you know you would like a `numpy` environment upon creation, you can target it in your `conda create` command:

```
$ conda create --name my_env python=3 numpy
```

If you are no longer working on a specific project and have no further need for the associated environment, you can remove it. To do so, type the following:

```
$ conda remove --name my_env35 --all
```

Now, when you type the `conda info --envs` command, the environment that you removed will no longer be listed.

Updating Anaconda

You should regularly ensure that Anaconda is up-to-date so that you are working with all the latest package releases.

To do this, you should first update the `conda` utility:

```
$ conda update conda
```

When prompted to do so, type `y` to proceed with the update.

Once the update of `conda` is complete, you can update the Anaconda distribution:

```
$ conda update anaconda
```

Again when prompted to do so, type `y` to proceed.

This will ensure that you are using the latest releases of `conda` and Anaconda.

Uninstalling Anaconda

If you are no longer using Anaconda and find that you need to uninstall it, you should start with the `anaconda-clean` module which will remove configuration files for when you uninstall Anaconda.

```
$ conda install anaconda-clean
```

Type `y` when prompted to do so.

Once it is installed, you can run the following command. You will be prompted to answer `y` before deleting each one. If you would prefer not to be prompted, add `--yes` to the end of your command:

```
anaconda-clean
```

This will also create a backup folder called `.anaconda_backup` in your home directory:

```
Output
Backup directory: /home/sammy/.anaconda_backup/2017-01-25T191831
```

You can now remove your entire Anaconda directory by entering the following command:

```
$ rm -rf ~/anaconda3
```

Finally, you can remove the PATH line from your `.bashrc` file that Anaconda added. To do so, first open nano:

```
$ nano ~/.bashrc
```

Then scroll down to the end of the file (if this is a recent install) or type `CTRL + W` to search for Anaconda. Delete or comment out the following lines:

/home/sammy/.bashrc

```
# added by Anaconda3 4.2.0 installer
export PATH="/home/sammy/anaconda3/bin:$PATH"
```

When you're done editing the file, type `CTRL + X` to exit and `y` to save changes.

Anaconda is now removed from your server.

Conclusion

This tutorial walked you through the installation of Anaconda, working with the `conda` command-line utility, setting up environments, updating Anaconda, and deleting Anaconda if you no longer need it.

You can use Anaconda to help you manage workloads for data science, scientific computing, analytics, and large-scale data processing.

By: Lisa Tagliaferri

Related Tutorials

How To Ensure Code Quality with SonarQube on Ubuntu 18.04

How to Manually Set Up a Prisma Server on Ubuntu 18.04

How To Display Data from the DigitalOcean API with React

How To Set Up Jupyter Notebook with Python 3 on Ubuntu 18.04

How to Install Node.js and Create a Local Development Environment on macOS

16 Comments

Leave a comment...

Log In to Comment

BabulMiah *February 14, 2017*

4 great article, you are a brilliant.

lamek *February 14, 2017*

0 Collateral learning: I'd never verified a download with a sha256sum before. There must be ways that people automatically compare the two strings?

---

ltagliaferri MOD *February 14, 2017*

0 There's a semi-automatic method detailed in the Ubuntu documentation.

---

jamessmith79 *December 7, 2017*

1 My route is like so:

1. copy the hash from the site
2. `echo "HASH GOES HERE" > hashcheck.txt`
3. `sha256sum Anaconda3-5.0.1-Linux-x86_64.sh | awk '{print $1;}' >> hashcheck.txt`
4. [optional] `less hashcheck.txt`
5. `cat hashcheck.txt | uniq | wc -l`

Comments:
1 - pretty obvious.
2 - this creates `hashcheck.txt` with the hash as the only line of content.
3 - this runs the checksum, but then pipes (passes) that result to the awk command, which here takes everything up to the first space (in this case, the hash resulting from the checksum), and then appends that result to the `hashcheck.txt` file.
4 - [optional] this just displays the contents of the file so you can give it the eye test.
5 - if you don't trust your eyes with those long hash strings, even when mashed together in the file, run this command. this passes the contents of the file to check uniqueness, by line. The output is thus: 1 == they match, 2 == they do not match, and you should run away. :)

Happy conda-ing!

---

ltagliaferri MOD *December 7, 2017*

1 Thanks for sharing your solution!

Steven18508 *June 8, 2017*

2 Very Good !

---

sasebot *June 21, 2017*

3 I loved your article :) !

secodneamil960 *October 20, 2017*

1 Thank u ... It help me much :D

---

mimounidali *December 17, 2017*

1 thanks

---

pblayo *December 26, 2017*

0 Don't the `curl` command line should use a redirect toward a file?

```
curl -O https://repo.continuum.io/archive/Anaconda3-4.2.0-Linux-x86_64.sh
```

Right now this command's output is the console. I did something like:

```
curl -O https://repo.continuum.io/archive/Anaconda3-4.2.0-Linux-x86_64.sh > Anaconda3-4.2.0-
```

Or maybe before a wget was used instead of curl? (like in https://github.com/menikhilpandey/DO-One-Click-Anaconda-Install/blob/master/commands.txt#L4)

---

ltagliaferri MOD *December 27, 2017*

1 The installation step of the tutorial uses best practices to install Anaconda, which is via the Anaconda installer bash script. We first verify it by checking the SHA-256 checksum and then run the script with the `bash` command.

The `curl -O` command that we use gets the installer and stores it in a local file so that we can verify the data integrity of the installer with cryptographic hash verification prior to running the script.

Thank you for the comment, it has prompted me to fully review the tutorial and I am going to update it for the new Anaconda version.

---

zaidafzal *March 7, 2018*

0 Hi @pblayo. curl by default prints file content to terminal, which is annoying. That output should be redirected to a file. Like

curl -0 https://repo.continuum.io/archive/Anaconda3-5.1.0-Linux-x86_64.sh > Anaconda3-5.1.0-Linux-x86_64.sh

Note -O is alphabet not number zero. There is no mistake in article. IDK why it looks like zero to me. That is confusing.

wavapexudo *March 23, 2018*

0 if you perform the step ::conda create --name my_env python=3
and type python and also check the version, but when you import the conda installed packages you cant . for me:: conda create -n "name of environment" worked in ubuntu 16.04

---

vinodkinoni *June 22, 2018*

0 great article
thanks u for this
but my anaconda installation got error following message
bin/pathon: can not execute binary file : exec format error plz help me
it my 4th time installation

---

elliotandersoninit5 *July 2, 2018*

0 Actually, I have a doubt:
Recently I've followed community tutorial:https://www.digitalocean.com/community/tutorials/how-to-install-python-3-and-set-up-a-programming-environment-on-an-ubuntu-16-04-server
Now if I'm installing conda will it conflict with my previous packages?

---

ltagliaferri MOD *July 2, 2018*

0 Following either of these setups will provide you with a separate programming environment, so they should not conflict. That is, if you install matplotlib into one but not the other, you will only be able to use it in the environment you installed it into.

You may find that one particular package manager better suits your needs. The general Python 3 setup that you mention in your comment is good for general use cases, while the Anaconda setup in this tutorial is best suited to data analysis and machine learning workflows.