A CLOUD GURU

Lesson Description - Installing Third-Party Packages Using 'pip'

We installed `pip3.6` when we built Python 3, and now we're ready to start working with Third-Party code.

Python Documentation For This Video

- `pip`
- `boto3`

Viewing Installed Packages

We can check out your installed packages using the `list` subcommand:

```
$ pip3.6 list
DEPRECATION: The default format will switch to columns in the
future. You can use --format=(legacy|columns) (or define a
format=(legacy|columns) in your pip.conf under the [list]
section) to disable this warning.
pip (9.0.1)
setuptools (28.8.0)
```

You may have gotten a deprecation warning. To fix that, let's create a `$HOME/.config/pip/pip.conf` file:

```
$ mkdir -p ~/.config/pip
$ vim ~/.config/pip/pip.conf
```

~/.config/pip/pip.conf

```
[list]
format=columns
```

Now if we use `list` we'll get a slightly different result:

```
$ pip3.6 list
Package    Version
---------- -------
pip        9.0.1
setuptools 28.8.0
```

Installing New Packages

Later in this course, we'll be using the `boto3` package to interact with AWS S3. Let's use that as an example package to install using the `install` subcommand:

```
$ pip3.6 install boto3
...
PermissionError: [Errno 13] Permission denied: '/usr/local/lib/
python3.6/site-packages/jmespath'
```

Since we installed Python 3.6 into `/usr/local`, it's meant to be usable by all users, but we can only add or remove packages if we're `root` (or via `sudo`).

```
$ sudo pip3.6 install boto3
```

Managing Required Packages with `requirements.txt`

If we have a project that relies on `boto3`, we probably want to keep track of that dependency somewhere, and pip can facilitate this through a "requirements file" traditionally called `requirements.txt`. If we've already installed everything manually, then we can dump the current dependency state using the `freeze` subcommand that pip provides.

```
$ pip3.6 freeze
boto3==1.5.22
botocore==1.8.36
docutils==0.14
jmespath==0.9.3
python-dateutil==2.6.1
s3transfer==0.1.12
six==1.11.0
$ pip3.6 freeze > requirements.txt
```

Now we can use this file to tell pip what to install (or uninstall) using the `-r` flag to either command. Let's uninstall these packages from the global site-packages:

```
$ sudo pip3.6 uninstall -y -r requirements.txt
```

Installing Packages Local To Our User

We need to use `sudo` to install packages globally, but sometimes we only want to install a package for ourselves, and we can do that by using the `--user` flag to the `install` command. Let's reinstall `boto3` so that it's local to our user by using our `requirements.txt` file:

```
$ pip3.6 install --user -r requirements.txt
$ pip3.6 list --user
$ pip3.6 uninstall boto3
```