PLUGINS

`ceph-volume` started initially to provide support for using `lvm` as the underlying system for an OSD. It is included as part of the tool but it is treated like a plugin.

This modularity, allows for other device or device-like technologies to be able to consume and re-use the utilities and workflows provided.

ADDING PLUGINS

As a Python tool, plugins `setuptools` entry points. For a new plugin to be available, it should have an entry similar to this in its `setup.py` file:

```
setup(
    ...
    entry_points = dict(
        ceph_volume_handlers = [
            'my_command = my_package.my_module:MyClass',
        ],
    ),
```

The `MyClass` should be a class that accepts `sys.argv` as its argument, `ceph-volume` will pass that in at instantiation and call them `main` method.

This is how a plugin for ZFS could look like for example:

```
class ZFS(object):

    help_menu = 'Deploy OSDs with ZFS'
    _help = """
Use ZFS as the underlying technology for OSDs

--verbose   Increase the verbosity level
    """

    def __init__(self, argv):
        self.argv = argv

    def main(self):
        parser = argparse.ArgumentParser()
        args = parser.parse_args(self.argv)
        ...
```

And its entry point (via `setuptools`) in `setup.py` would looke like:

```
entry_points = {
    'ceph_volume_handlers': [
        'zfs = ceph_volume_zfs.zfs:ZFS',
    ],
},
```

After installation, the `zfs` subcommand would be listed and could be used as:

```
ceph-volume zfs
```