Tests

Actual tests are written in Python methods that accept optional fixtures. These fixtures come with interesting attributes to help with remote assertions.

As described in Conventions, tests need to go into `tests/functional/tests/`. These are collected and *mapped* to a distinct node type, or *mapped* to run on all nodes.

Simple Python asserts are used (these tests do not need to follow the Python `unittest.TestCase` base class) that make it easier to reason about failures and errors.

The test run is handled by `py.test` along with testinfra for handling remote execution.

Test Files

Test Fixtures

Test fixtures are a powerful feature of `py.test` and most tests depend on this for making assertions about remote nodes. To request them in a test method, all that is needed is to require it as an argument.

Fixtures are detected by name, so as long as the argument being used has the same name, the fixture will be passed in (see pytest fixtures for more in-depth examples). The code that follows shows a test method that will use the `node` fixture that contains useful information about a node in a ceph cluster:

```
def test_ceph_conf(self, node):
    assert node['conf_path'] == "/etc/ceph/ceph.conf"
```

The test is naive (the configuration path might not exist remotely) but explains how simple it is to “request” a fixture.

For remote execution, we can rely further on other fixtures (tests can have as many fixtures as needed) like `File`:

```
def test_ceph_config_has_inital_members_line(self, node, File):
    assert File(node["conf_path"]).contains("^mon initial members = .*$")
```

node fixture

The `node` fixture contains a few useful pieces of information about the node where the test is being executed, this is captured once, before tests run:

- `address`: The IP for the `eth1` interface
- `subnet`: The subnet that `address` belongs to
- `vars`: all the ansible vars set for the current run
- `osd_ids`: a list of all the OSD IDs
- `num_mons`: the total number of monitors for the current environment
- `num_devices`: the number of devices for the current node
- `num_osd_hosts`: the total number of OSD hosts
- `total_osds`: total number of OSDs on the current node
- `cluster_name`: the name of the Ceph cluster (which defaults to ‘ceph’)
- `conf_path`: since the cluster name can change the file path for the Ceph configuration, this gets sets according to the cluster name.
- `cluster_address`: the address used for cluster communication. All environments are set up with 2 interfaces, 1 being used exclusively for the cluster
- `docker`: A boolean that identifies a Ceph docker cluster

- `osds`: A list of OSD IDs, unless it is a docker cluster, where it gets the name of the devices (e.g. `sda1`)

Other Fixtures

There are a lot of other fixtures provided by testinfra as well as `py.test`. The full list of `testinfra` fixtures are available in testinfra_fixtures

`py.test` builtin fixtures can be listed with `pytest -q --fixtures` and they are described in pytest builtin fixtures