API UPTIME TESTS

OVERVIEW

These tests exist to validate that an OpenStack API is available during an upgrade. Note that these tests complement the "during" tests which also run during an upgrade, but validate that services are available. The API uptime tests will only ensure that the API is available.

ARCHITECTURE

Many tools exist to drive traffic to an OpenStack API, including tempest and rally, however most require significant setup and configuration. For the POC for this test tool we wanted something that would be quick and easy to implement, so the decision was made to use the python clients to talk to the APIs directly.

The API uptime tests currently can talk to:

1. Nova
2. Cinder
3. Neutron
4. Swift

The tests run in their own processes (one per API) and can run either for a specified number of iterations, or until a sentinel file is found (this is known as daemon mode). Daemon mode makes it easy to start the tests and run them until someone/something decides that it's time to stop. Stopping the daemon is as easy as creating the sentinel file (normally via the "touch" command). Contents of the file are not important, only its existence. Note that you will want to be sure that the sentinel file doesn't exist before starting the tests in daemon mode.

FLOW

The flow for the tests is similar to that used by the "During" tests:

1. User executes "python call_test.py –d" from cmd line (daemon mode)
2. Credentials and system under test data read from tempest config or os.cnf
3. For each service to be tested:
    a. Start a process for the test.
    b. If running daemon check for the existence of the sentinel file. If it is found, then stop the test process.
    c. Write the test output

For each service to be tested a process is started (see 3a above). This process will:

1. Start a timer
2. Start a process for the API call (which will be made via the python client).
3. Report the tests results. The data format is:

```
{
serviceName: {
    "uptime_pct": uptime_pct,
    "total_requests": total,
    "successful_requests": success,
    "failed_requests": total - success,
    "start_time": start_time,
    "end_time": end_time
    }
}
```

COMMAND OPTIONS

The call_test command takes the following options:

Option	Description
-d	Run in daemon mode
-s <service>	Test the specified service(s)
-t <times>	Run for the specified number of iterations
-o <file>	Write to the specified file
-v	Produce verbose output (useful for debugging)

TODOS

Currently each instance of the test process gets ALL clients. It really should only get the client it needs based on the service to be tested.