Promise Installation and Configuration Guide

Release brahmaputra.1.0 (9921dd3)

OPNFV

February 25, 2016

CONTENTS

1 Promise Feature Configuration Overview **1**
1.1 Promise installation 1
1.2 Testing 1

CHAPTER

ONE

PROMISE FEATURE CONFIGURATION OVERVIEW

1.1 Promise installation

Install nodejs, npm and promise

```
curl -sL https://deb.nodesource.com/setup_4.x | sudo -E bash -
sudo apt-get install -y nodejs
sudo npm -g install npm@latest
git clone https://github.com/opnfv/promise.git
cd promise
npm install
```

Please note that the last command 'npm install' will install all needed dependencies for promise (including yangforge and mocha)

1.2 Testing

Please perform the following preparation steps:

1. Set OpenStack environment parameters properly (e.g. source openrc admin demo in DevStack)
2. Create OpenStack tenant (e.g. promise) and tenant user (e.g. promiser)
3. Create a flavor in Nova with 1 vCPU and 512 MB RAM
4. Create a private network, subnet and router in Neutron
5. Create an image in Glance

Once done, the promise test script can be invoked as follows (as a single line command):

```
NODE_ENV=mytest \
OS_TENANT_NAME=promise \
OS_USERNAME=promiser \
OS_PASSWORD=<user password from Step 2> \
OS_TEST_FLAVOR=<flavor ID from Step 3> \
OS_TEST_NETWORK=<network ID from Step 4> \
OS_TEST_IMAGE=<image ID from Step 5> \
npm run -s test -- --reporter json > promise-results.json
```

The results of the tests will be stored in the promise-results.json file.

The results can also be seen in the console ("npm run -s test")

All 33 tests passing?! Congratulations, promise has been successfully installed and configured.

```
  allocation using reservation for immediate use
    create-reservation
      ✓ should create reservation record (no start/end) without error (130ms)
      ✓ should update promise.reservations with a new entry
      ✓ should contain a new ResourceReservation record in the store
    create-instance
      ✓ should create a new server in target provider (with reservation) without error (1189ms)
      ✓ should contain a new ResourceAllocation record in the store
      ✓ should be referenced in the reservation record
      ✓ should have high priority state
  reservation for future use
    create-reservation
      ✓ should create reservation record (for future) without error (276ms)
      ✓ should update promise.reservations with a new entry (81ms)
      ✓ should contain a new ResourceReservation record in the store
    query-reservation
      ✓ should contain newly created future reservation (129ms)
    update-reservation
      ✓ should modify existing reservation without error (244ms)
    cancel-reservation
      ✓ should modify existing reservation without error (104ms)
      ✓ should no longer contain record of the deleted reservation
  capacity planning
    decrease-capacity
      ✓ should decrease available capacity from a provider in the future (101ms)
    increase-capacity
      ✓ should increase available capacity from a provider in the future (104ms)
    query-capacity
      ✓ should report available collections and utilizations (201ms)
  reservation with conflict
    create-reservation
      ✓ should fail to create immediate reservation record with proper error (233ms)
      ✓ should fail to create future reservation record with proper error (122ms)
  cleanup test allocations
    destroy-instance
      ✓ should successfully destroy all allocations (1462ms)


33 passing (7s)
```