IRACST – International Journal of Computer Networks and Wireless Communications (IJCNWC), ISSN: 2250-3501
Vol.5, No.2, April 2015

Automation of Network Functionality using Openstack Tempest Framework

Abhishek.G.M, Second year M.Tech (CN&E)
Department of Information Science & Engineering
The National Institute of Engineering
Mysuru, INDIA

Dr. K. Raghuveer, Professor & Head
Department of Information Science & Engineering
The National Institute of Engineering
Mysuru, INDIA

***Abstract-* OpenStack is a global collaboration of developers and cloud computing technologists producing the open standard cloud operating system for both public and private clouds Cloud computing provides users with access to a shared collection of computing resources: networks for transfer, servers for storage, and applications or services for completing tasks.**

The operator's want to verify their cloud works well easily in a short time in order to avoid any regression and to compare the stability of different software version. And operator's want to verify during/after setup, after adding compute/ controller nodes, minor software updates, minor bug fixes and periodically.

In order to verify we use Tempest a set of integration tests to be run against a live OpenStack cluster. Verifying multiple components is one of the greatest concerns from developer's and operator's viewpoint. We need to have Tempest code for integrated projects to ensure their validity. Currently there are limited Test cases to validate the network scenarios. Validation is very important to check network operation. Adding more test cases in network will validate the network.

***Keywords* - Openstack, Tempest**

I. INTRODUCTION

Cloud computing provides users with access to a shared collection of computing resources: networks for transfer, servers for storage, and applications or services for completing tasks.

The compelling features of a cloud are:

On-demand self-service: Users can automatically provision needed computing capabilities, such as server time and network storage, without requiring human interaction with each service provider.

Network access: Any computing capabilities are available over the network. Many different devices are allowed access through standardized mechanisms.

Resource pooling: Multiple users can access clouds that serve other consumers according to demand.
Elasticity: Provisioning is rapid and scales out or is based on need.
Metered or measured service: Cloud systems can optimize and control resource use at the level that is appropriate for the service. Services include storage, processing, bandwidth, and active user accounts.
Cloud computing offers different service models depending on the capabilities a consumer may require.
SaaS: Software-as-a-Service. Provides the consumer the ability to use the software in a cloud environment, such as web-based email for example.
PaaS: Platform-as-a-Service. Provides the consumer the ability to deploy applications through a Programming language or tools supported by the cloud platform provider. An example of Platform-as-a-service is an Eclipse/Java programming platform provided with no downloads required.
IaaS: Infrastructure-as-a-Service. Provides infrastructure such as computer instances, network connections, and storage so that people can run any software or operating system.

II. OPENSTACK ARCHITECTURE

OpenStack has a modular architecture with various code names for its components

Compute (Nova):
OpenStack Compute (Nova) is a cloud computing fabric controller, which is the main part of an IaaS system. It is designed to manage and automate pools of computer resources and can work with widely available virtualization technologies, as well as bare metal and high-performance computing (HPC) configurations.

Object Storage (Swift):

OpenStack Object Storage (Swift) is a scalable redundant storage system. Objects and files are written to multiple disk drives spread throughout servers in the data center, with the OpenStack software responsible for ensuring data replication and integrity across the cluster.

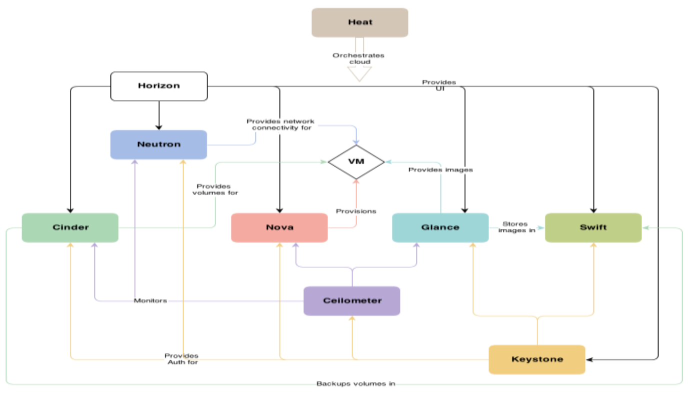

Figure. 1 Conceptual architecture

.

Block Storage (Cinder)

OpenStack Block Storage (Cinder) provides persistent block-level storage devices for use with OpenStack compute instances. The block storage system manages the creation, attaching and detaching of the block devices to servers. Block storage volumes are fully integrated into OpenStack Compute and the Dashboard allowing for cloud users to manage their own storage needs.

Networking (Neutron):

OpenStack Networking (Neutron, formerly Quantum) is a system for managing networks and IP addresses.

Dashboard (Horizon):

OpenStack Dashboard (Horizon) provides administrators and users a graphical interface to access, provision, and automate cloud-based resources

Identity Service (Keystone):
Provides an authentication and authorization service for other Open-Stack services. Provides a catalog of endpoints for all OpenStack services.

Image Service (Glance):

OpenStack Image Service (Glance) provides discovery, registration, and delivery services for disk and server images. Stored images can be used as a template. It can also be used to store and catalog an unlimited number of backups.

Telemetry (Ceilometer):

OpenStack Telemetry Service (Ceilometer) provides a Single Point of Contact for billing systems, providing all the counters they need to establish customer billing, across all current and future OpenStack components.

Orchestration (Heat):

Heat is a service to orchestrate multiple composite cloud applications using templates.

Database (Trove):

Trove is a database-as-a-service provisioning relational and non-relational database engines.

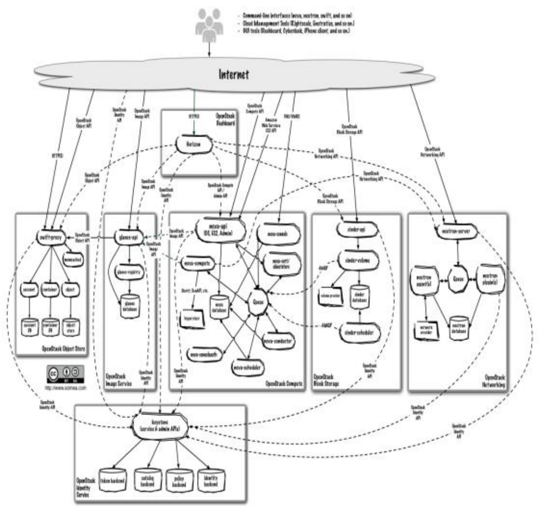

Figure. 2 Logical Architecture

End users can interact through a common web interface (Horizon) or directly to each service through their
API
• All services authenticate through a common source (facilitated through keystone)
• Individual services interact with each other through their public APIs (except where privileged administrator commands are necessary).

III. RELATED WORK

Before tempest, every project has its own unit tests and some of the projects has functional tests will bring up entire stack and will test against that but there is nothing really out there to test the whole picture.

IV. PROPOSED WORK

Tempest: The openstack Integration Test suite

Tempest is a set of integration tests to be run against a live OpenStack cluster. Tempest has batteries of tests for OpenStack API validation, Scenarios, and other specific tests useful in validating an OpenStack deployment.

Openstack is complicated that is different components and API's talking to each other asynchronously. we need a solution to test all together. That's the reason why tempest came in to picture.

Tempest interacts with openstack REST APIs and is used in gating test for every single commits for a openstack tempest

Tempest will run 100 - 1000's of times a day to verify that openstack patches work correctly and don't brake expected behavior.

Writing test cases will clear many bugs.

Growing pains:

- Huge continued increase in the number of projects and tests.
- The common code to handle all APIs become more complex.
- Managing individual tests become very difficult.
- Total run time continues to grow.
- Tempest configuration becomes more involved to handle all the services and configurations.

Purpose and use cases of Tempest:
For Operators:
1. Checking their cloud works correctly.
2. To check/avoid regression while software upgrade.
3. To compare the stability of different software versions.

For Developers:
1. New code can be tested to check the expected behavior.
2. To verify whether new code introduce any regression.
3. On each new patch, Tempest runs in Zuul to make sure it qualify the expected quality/stability.

V. DESIGN PRINCIPLES:

- Should be able to run against any openstack cloud from devstack to a public cloud
- Everything explicit – tempest should know entire configuration upfront by doing this we can prevent things disappearing.
- Only use public interfaces (REST APIs)
- Should be self-cleaning - whatever the resources are created during testing are to be tear down
- Self-testing - to verify tempest itself is working properly.

Types of tests:

1) API tests
2) Scenario tests

API tests:

- Directly test the openstack REST APIs.
- Uses a unified tempest REST client.
- Fall in to two categories positive and negative.
- Negative tests are starting to be automatically generated from api schemas.

Scenario Tests:

- Through path test of functionality – functional validation of the openstack deployment i.e there will be integration between the projects.
- Scenario tests are test cases across multiple components, it can be used for system testing for OpenStack cloud.
- By using the scenario tests, it is possible to reduce the evaluation cost of your cloud environment in comparison with making a test suite from scratch.

*Features of tempest***:**

- Tempest is capable of running its tests in parallel to decrease the run time and to improve the openstack quality.
- Tenant isolation - Each test class creates its own user and tenant in keystone before it runs any tests and this context will be used for all of the testing within that class that helps in execution of tests in parallel at the class level.
- Tempest configuration—Intent of config file is to tell tempest what can be run i.e what is actually running on the openstack deployments and what should be run is specified in test runner.

VI. PROJECT FLOW

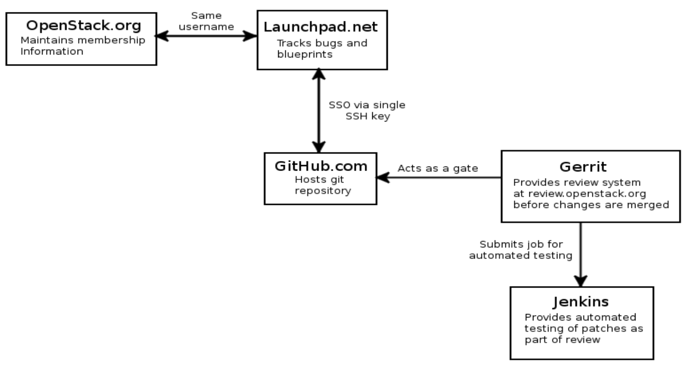
OpenStack.org
Maintains membership information
Same username
Launchpad.net
Tracks bugs and blueprints
SSO via single SSH key
GitHub.com
Hosts git repository
Acts as a gate
Gerrit
Provides review system at review.openstack.org before changes are merged
Submits job for automated testing
Jenkins
Provides automated testing of patches as part of review

Figure 3. Workflow

Devstack is a documented shell script to build complete OpenStack development environments and is written in bash which will pull code from the upstream OpenStack git repositories and deploy it on host or virtual machine and will run on Fedora or Ubuntu.
Launchpad is a web application and website that allows users to develop and maintain software, particularly open-source software.

GitHub is a web-based Git repository hosting service, which offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features.
Gerrit is a web based code review system, facilitating online code reviews for projects using the Git version control system.
Jenkins is a Continuous Integration system that runs tests and automates some parts of project operations. It is controlled for the most part by *Zuul* which determines what jobs are run when.
Setting up git,:

- Install git
- Configure git

Setting up git-review:

- Install Python's setuptools
- Install pip
- Install git-review
- Add Gerrit

Setting up and configuring ssh:

- Generate a new key
- Add key in review.openstack.org

Setting up the tempest:

- Create a local directory
- Clone the tempest repository
- Write test by using python language in a new branch.
- Committing changes

VII. CONCLUSION

By tempest we can verify the cloud that works well easily in a short time. And to check stability of different software version to avoid regression.

ACKNOWLEDGEMENT

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. We would like to extend our sincere thanks to all of them.

REFERENCES

[1] https://www.openstack.org/
[2] http://docs.openstack.org/
[3] http://developer.openstack.org/api-ref.html
[4] http://docs.openstack.org/api/quick-start/content/
[5] http://docs.openstack.org/user-guide/content/
[6] http://docs.openstack.org/user-guide-admin/content/
[7] http://docs.openstack.org/cli-reference/content/
[8] http://developer.openstack.org/
[9] https://wiki.openstack.org/wiki/Documentation/HowTo
[10] https://wiki.openstack.org/wiki/Gerrit_Workflow
[11] http://docs.openstack.org/developer/tempest/
[12] http://en.wikipedia.org/wiki/OpenStack

AUTHORS PROFILES

Received a B.E degree in computer science and engineering from Kuvempu University, karnataka, India in 2012. Currently pursuing M.tech in computer network and engineering from National Institute of Engineering, mysuru, karnataka India.

Has been at the National Institute of Engineering since 1984, as a B.E student (1984-88) Department of Computer Science & Engineering, National Merit Scholarship Holder, Government of India. Completed M.E from Devi Ahilya VishwaVidyalaya, formerly known as University of Indore (1991-93) and Ph.D from Visvesvaraya Technological University (2003-2007). Currently working as Professor and Head, Department of Information Science & Engineering.