OpenStack Post Deployment Config, Logs and Metrics Management System

Mahesh D N
Department of Computer Science and Engineering
Reva Institute of Technology and Management,
Bengaluru, India
Email: maheshdn18@gmail.com

Pavithra P
Department of Computer Science and Engineering
Reva Institute of Technology and Management,
Bengaluru, India

Abstract **-The OpenStack cloud platform has more number of configuration variables. Hence there is a need for an efficient management system to manage the configuration variables of the OpenStack cloud platform post deployment. It is very necessary to identify the differences in the configuration variables across multi datacenter environment. To keep track values of configuration variables from time to time in the cloud environment. Further to modify the configuration variables selectively in the cloud environment. The management system can also be extended to manage the system information such as memory, CPU info etc. The OpenStack cloud services generate huge amount of logs. The logs and performance metrics have to be selectively collected for analyzing the failures and performance issues. To achieve this I propose a light weight post deployment configuration, logs and metrics management system for OpenStack based cloud platforms. The system will have agents to collect the configuration variables, logs and performance metrics and stores on a centralized common node. The user can query the collected information for analyzing configuration differences, root cause for failures and performance issues.**

I. INTRODUCTION

OpenStack is an IaaS (Infrastructure as a Service) solution provider [5]. It is an open source cloud platform that offers a bunch of interrelated services. Each OpenStack service exposes APIs for interacting with other services. Based on the requirements the some or all services can be installed. Ceilometer provides metering service, which collects all the required metrics for billing. Ceilometer is also used for statistical purpose, benchmark and scalability evaluation. OpenStack collects a lot of data which goes unused apart from getting used for its core functionalities. Most of the cloud deployments span across multiple datacenters. So there is a need for having a centralized monitoring system for configurations across the system. Some of the tools available for deployment and management of the OpenStack cloud are like puppet, chef, Ansible etc. But there is a need for managing the system post deployment. When the size of the cloud is big and complex, it will be very difficult to manage and diagnose [1] the cloud resources. It is very necessary for the users to quickly identify the root cause for the failures.

In order to effectively analyze the root cause for the problem it is necessary to collect all the operational logs from the cloud environment. The production environment generates huge amount of logs every day. So the task of looking for root cause from the huge logs is very inefficient. It is more time consuming, require more skilled human resources.

Hence the proposed post deployment management system for OpenStack cloud can efficiently analyze the differences between the multiple datacenters and across components. Also keep track of configuration changes from time to time. Further to modify the configuration variables selectively in the cloud environment. The management system can also be extended to manage the system information such as memory, CPU info etc. And logs and performance metrics have to be selectively collected for analyzing the failures and performance issues. The proposed light weight post deployment configuration, logs and metrics management system for OpenStack based cloud platforms. The system will have agents to collect the configuration variables, logs and performance metrics and stores on a centralized common node. The user can query the collected information for analyzing configuration differences, root cause for failures and performance issues.

II. MOTIVATION

As the world is moving towards cloud services, it is very important to manage the production environment to provide best services. When the size of the cloud is big and complex, it will be very difficult to manage and diagnose the cloud resources. Most of the cloud deployments span across multiple datacenters. It is very necessary for the users to quickly identify the root cause for the failures. The production cloud environment generates huge amount of logs on every day. Hence a specific centralized tool which is being proposed will be much helpful to maintain the OpenStack cloud deployments post deployment.

III. LITERATURE REVIEW

Review of the literature has been done with following sources:

- The Internet
- Publications and articles

Cloud Computing is a rapidly growing technology in the market currently. So there is a need for quick automated deployment and scalable infrastructure to serve the purpose. The base of the cloud computing technology is Infrastructure as a Service (IaaS). The providers have to deploy both physical resources as well as virtual resources while building cloud infrastructure with lot of difficulties. Hence the cloud infrastructure management has to be automated so that the resources provisioned quickly and released with minimal effort from cloud service providers. Infrastructure scalability is another big task for the cloud infrastructure providers. The cloud deployment models can be broadly classified into four types: 1) public; 2) private; 3) hybrid; 4) community [2]. OpenStack is a widely used distributed IaaS cloud platform [3]. It is best known for its speedy release cycle and reliability [4].

The following sections will describe the list of deployment and management tools available for deploying and managing OpenStack clouds.

A. Salt Stack

It is an open source python based Configuration management system [6]. It supports “Infrastructure as Code” for the deployment and management of cloud. It is a new approach for deploying and managing cloud infrastructure dynamically. It is easy to plug with the cloud platform as it is written in python. It maintains large number of remote nodes in a defined state. The best feature of salt is parallelism in the executing of commands remotely using AES algorithm. The architecture of Salt Stack is as shown in Fig. 1:

The Centralized Salt Stack Master: It issues commands to the Slat masters for managing authentication and communication with salt minion.

The Salt Stack Master: It simple delegates commands from centralized slat master to Minions to achieve parallelism.

The Slat Stack Minion: It executes the commands that are received from the salt master and returns back the results to the salt master. It has no default deployment method for OpenStack, but community is growing to provide a promising model for OpenStack deployment. Puppet is written in RUBY. It used DSL for manifest files and ERB for writing templates.

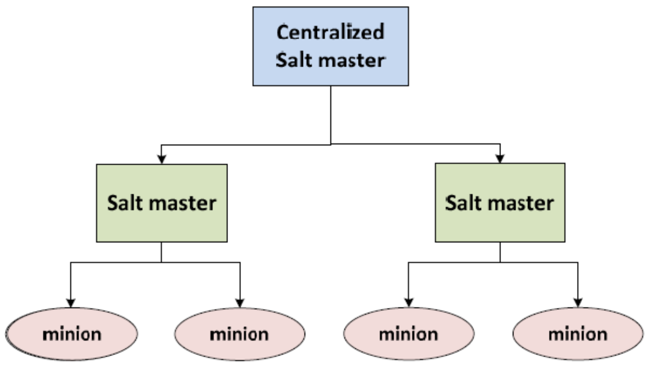

Fig. 1. Salt Architecture

B. Puppet

This provides client-server architecture for cloud deployment and management [7]. The client will periodically poll the server for the desired state and returns the status to the server. Puppet has a lifecycle that works in highly distributed manner for deploying and managing the cloud efficiently. It is easy to upgrade and manage the cloud using puppet. The architecture of puppet is as shown in Fig. 2.

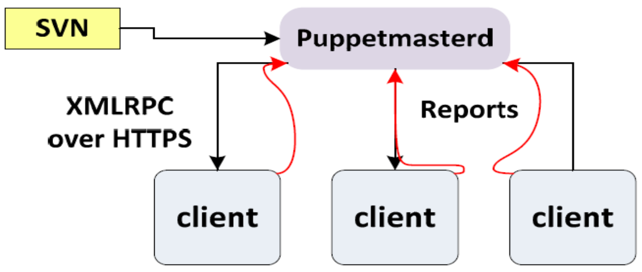

Fig. 2. Puppet Architecture

The Puppet Master: It will send commands to the clients (puppet agents) and receives back the status from clients.

The Puppet Agent: It executes the commands as directed by the puppet master and returns the results back to the puppet master.

C. Chef

Chef is an automated configuration management system. It supports “Infrastructure as Code” for describing and deploying the cloud infrastructure. It brings the resources (servers and services) into life. Since the infrastructure is managed by code, it can be easily automated and tested with efficiency. It is widely used for cloud infrastructure deployment on a bare metal. It uses RUBY for defining the environment. The chef environment is organized into cookbooks and recipes for deployment and management [8].

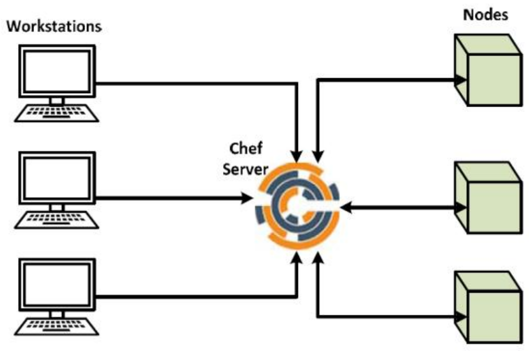

Fig. 3. Chef mini architecture

The chef core is written in Erlang and it is mainly designed for scalability to thousands of physical and virtual servers. It uses Infrastructure as a Code with integrated version control for configuration management.

The cookbook contains the main configuration information for the desired state of the node. The configuration information is defined with attributes, recipes, templates, metadata, resources and libraries. The architectural components of chef are [9]:

The Chef Server: This is the master node where all the configuration data is stored. It is associated with version control system. It directs the chef nodes to execute the chef cookbooks to configure the cloud infrastructure.

The Chef Workstation: All the chef cookbooks and recipes are developed on the chef workstation.

The Chef Nodes: The chef nodes execute the cookbooks and recipes as directed by the chef server to deploy and configure the cloud infrastructure.

D. Ansible

Ansible is simple and powerful open source tool for automation platform. It helps in task automation, configuration management and application management. In short it is an IT Orchestration system. Unlike puppet and chef it will not use agents on remote nodes. In a large organization it is very challenging to manage cloud resources. When it comes to cloud platform such as OpenStack, It has to manage individual services as well as manage relationship between the various cloud services [10].

Ansible is not just a deployment tool. Once the cloud is deployed, various OpenStack modules can be used to manage the cloud operations. Ansible can also be used for PaaS and SaaS deployments on top of IaaS. That is Ansible is used to provision application and services on top of cloud infrastructure [11]. It serves all cloud operators, users and developers.

E. DevStack

DevStack is an open source tool which is a series of extensible scripts for deploying OpenStack environment quickly. It will install latest version by default or a specific version as mentioned from the git repository. It is used to deploy environments for developmental and testing purposes. DevStack is not a production deployment tool [12].

F. TripleO

TripleO (OpenStack on OpenStack) is a tool that provides facilities such as deploying, Operating, Upgrading and managing OpenStack services using its own cloud services [13]. It has two layers of deployment:

The "Under-Cloud": This is an OpenStack deployment on the bare metal servers that spawns VMs for over-cloud deployment. It will manage the VMs where the over-cloud services are deployed.

The "Over-Cloud": This is the provider cloud deployment that is managed by the under-cloud. The over-cloud configuration is defined in the OpenStack Heat template and is orchestrated by the Heat engine.

This is a dedicated OpenStack deployment project that is integrated with OpenStack. It is targeted for large scale production deployments. It supports continuous integration and continuous deployment. It has itself proved as a long-term project. Many tech giants such as Red Hat, HP, IBM and other are using contributing towards the development of TripleO and using for the deployment of cloud infrastructure.

G. LOGAN

In a large and complex compute environment it is very challenging for managing cloud infrastructure. Debugging a failure is one of the challenges in large compute environment [14], [1]. Debugging a problem in cloud environment and quickly find the root cause without compromising on the quality of service is very challenging. Further in a multi datacenter deployments and distributed datacenters it is much more difficult for identifying a root cause of the problems. When a failure occurs, it may be in any of the cloud components. Hence in order to effectively analyze the root cause for the problem it is necessary to collect all the operational logs from the cloud environment. The production environment generates huge amount of logs every day. So the task of looking for root cause from the huge logs is very inefficient. It is more time consuming, require more skilled human resources.

LOGAN (LOG ANalytics) helps to quickly identify the root cause of the problems. LOGAN used an analytical algorithm to keep track of the previous occurrences of problems and try to match the new problems to find the root cause quickly.

H. ELK Stack

ELK is a stack of three components Elastic Search, Logstash and Kibana. It is a widely used log monitoring system. It is used to gather large amounts of logs (audit and event-logs) from the computers. The process involves collection of logs, Storing and Indexing on a centralized system, analyzing and reporting. The ELK stack has three main components:

Logstash: This is a collector agent that is deployed on the systems from where logs have to be collected. The Logstash forwards agents reside on the remote nodes and forward the logs to the centralized Logstash server where the logs are filtered and refined before feeding it to elastic search engine.

Elastic Search: This is database engine that stores all the log messages and indexes for optimal query and reporting.

Kibana: Kibana is visualizing interface where the logs statistics can be viewed and queried. Kibana also provides to filter the logs and debug the issues efficiently.

Hence efficiently analyzing the logs will helps in proper decision making and good services [15]. Most of the cloud platforms use ELK for monitoring logs.

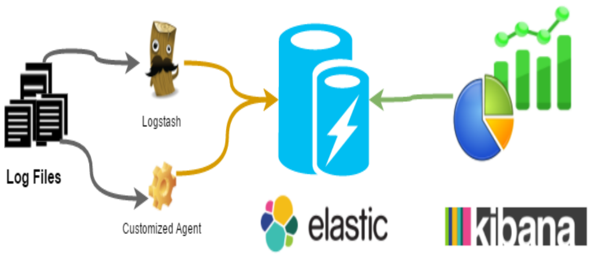

Fig 4: ELK Stack

I. Ceilometer Metrics

Ceilometer is OpenStack open monitoring source. It is main project for billing in OpenStack which is current de facto standard. Analysis of monitoring structure in cloud will gives good idea of monitoring Network virtualization. Ceilometer provides metering service, which collects all the required metrics for billing. Ceilometer is also used for statistical purpose, benchmark and scalability evaluation [16], [17], [18]. As a result, initial and last motives are billing. At first time, basic monitoring parameters (ex. instance, CPU, disk, ram and image upload event…) are monitored. Also alarm and event in the way of triggering mechanism are introduced. And further monitoring of network parameter in virtual machine or bare metal machine is added.

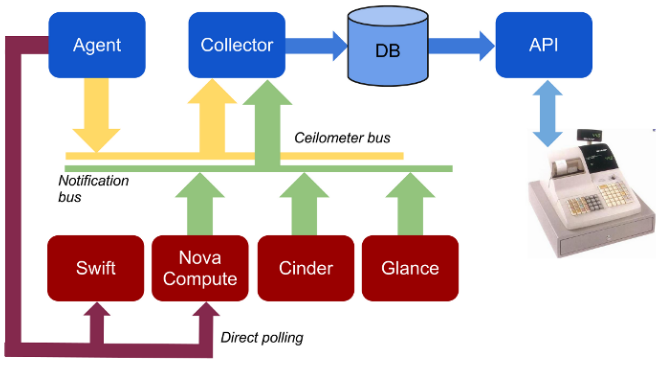

Fig 5: Ceilometer Architecture

IV. PROPOSED SYSTEM

The project aims at providing a user friendly tool to manage the configurations, logs and metrics of post deployment of OpenStack clouds. The OpenStack cloud platform has huge number of configuration variables. Hence there is a need for an efficient management system to manage the configurations of the OpenStack cloud. The management system should also be extended to manage the system information such as memory, CPU info etc.

A. Architecture of proposed system

The configuration management system shall be deployed on one of the common node which has access to all the data centers. The configuration management system shall be configured to read configuration information from the OpenStack deployments at regular intervals.

The proposed system aims at:

- Gather OpenStack configurations from multiple data centers and store in a common centralized node.
- Track the changes in a data center configuration from time to time.
- Check the configuration differences across different data centers.
- Check the configuration differences across multiple instances of a particular component type within the same data center.
- Generate comparison data to simplify correlation of changes in configurations with performance variations and other faults.
- Push configuration changes to OpenStack nodes (future potential work item).
- Gather the required logs to analyze the failures.
- Also gather physical host and system information such as Basic OS Info, Network

Info, Storage Info, Processor Info, Memory Info etc.

- Check the differences in the physical host info and other system information across different machines.

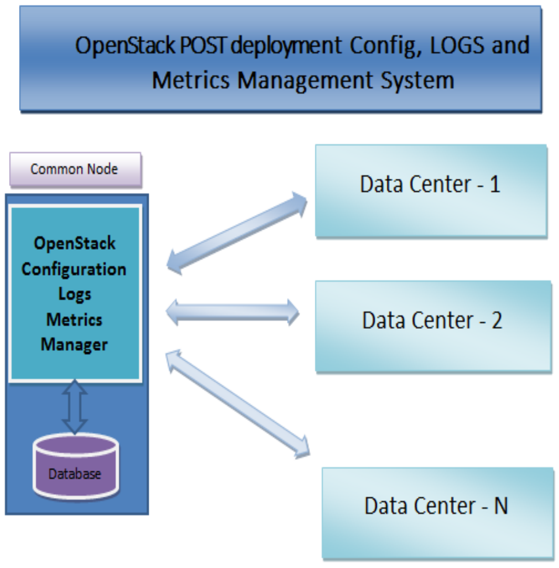
OpenStack POST deployment Config, LOGS and Metrics Management System
Common Node
OpenStack Configuration Logs Metrics Manager
Database
Data Center - 1
Data Center - 2
Data Center - N

Fig.6: The Block diagram of Proposed System

B. *Components of proposed system*

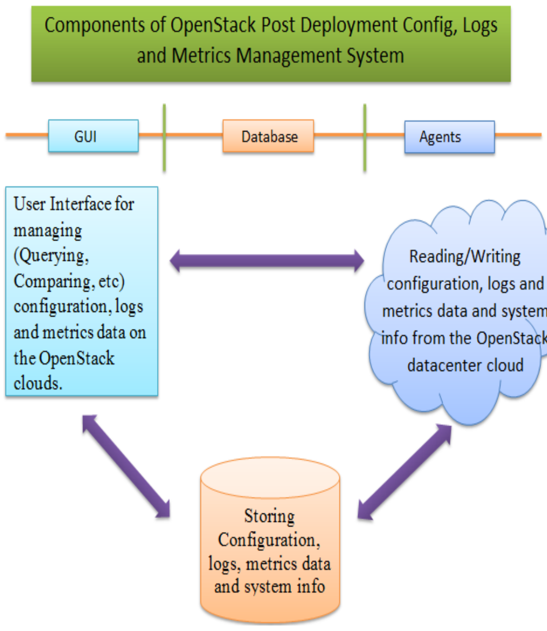
Components of OpenStack Post Deployment Config, Logs and Metrics Management System
GUI
Database
Agents
User Interface for managing (Querying, Comparing, etc) configuration, logs and metrics data on the OpenStack clouds.
Reading/Writing configuration, logs and metrics data and system info from the OpenStack datacenter cloud
Storing Configuration, logs, metrics data and system info

Fig 7: Components of Proposed System

The tool will have 3 major components/parts as follows

1. GUI Component
2. Database
3. Collector Agents

The GUI serves following needs:

- Allow user to fire Config, logs and metrics operation.
- Allow user to query data base to view configuration, logs and metric details.
- Allow user to generate comparison results across and within the data centers.
- Allow user to view reports.

The Database serves following needs:

- The database will be used to store and query the configuration, logs and metrics by the user.

The Collector Agents serves following needs:

- Reads the Config, logs and metrics information's from the OpenStack data center nodes.
- The tool will takes an input configuration file which gives the details of the data center nodes and authentication information.

C. *Software and Hardware Requirements*

Software

- Python 2.7 and above
- Python gtk (pygtk) GUI SDK
- Database (Mongo DB)
- PyMongo Client to interact with MongoDB
- Python xlwt for writing report to spreadsheet
- Python Glade UI designer
- OpenStack DevStack Deployment

Hardware

- Should work on both 32 bit and 64bit machines.
- Memory and storage requirements depend on the number of data centers and number of machines/VMs each data center hosts. Estimated figures to be updated.
- Test setup can be deployed on a basic laptop with minimal specs using DevStack deployment.

V. CONCLUSIONS AND FUTURE WORK

In the complex and distributed environment it is very difficult to know the configuration values. And very difficult to find the route cause in case of failures due to complexity and huge amount of logs produced. Hence a specific centralized tool which is being proposed will be much helpful to maintain the OpenStack cloud deployments post deployment.

The proposed system helps in efficiently manage the OpenStack environment post deployment. The tool manages multi datacenter configuration, logs and metrics on a centralized common node. It allows user to query for a particular configuration variable. Allow comparing configuration across datacenters and within the datacenter from time to time. Further logs and metrics help in analyzing the failures and performance issues.

The proposed system is very specific to OpenStack, hence same can be generalised to monitor other cloud platforms. Security for the data collected from cloud deployments has to be implemented.

REFERENCES

[1] J. Neville, K. Nagaraj and C. Killian. “The Structured comparative analysis of systems logs to diagnose performance problems”. In the 9th USENIX NSDI Conference 2012 (Networked Systems Design and Implementation).

[2] "Cloud Computing: The Concept, Impacts and the Role of Government Policy", Link: http://dx.doi.org/10.1787/5jxzf4lcc7f5-en, [Internet Source], OECD Digital Economy Papers, No. 240, OECD Publishing, 2014, [Last Accessed: April 2017]

[3] "Home » OpenStack Cloud Computing Platform Software", 2017. [Internet Source]. Link: http://www.openstack.org/ [Last Accessed: April 2017]

[4] "What is OpenStack?" 2017. [Internet Source]. Link: https://opensource.com/resources/what-is-openstack [Last Accessed: April 2017]

[5] "Openstack IaaS Platform", 2017. [Internet Source]. Link: https://wiki.openstack.org [Last Accessed: April 2017]

[6] "SaltStack Documentation", 2017. [Online]. Link: https://docs.saltstack.com/en/latest/ [Last Accessed: April 2017]

[7] "How it works", Puppet, 2017. [Internet Source]. Link: https://puppet.com/product/how-puppet-works [Last Accessed: April 2017]

[8] "Site Map — Chef Docs", 2017. [Internet Source]. Link: https://docs.chef.io/ [Last Accessed: April 2017]

[9] Andrei Bogdan Rus, Eduard Luchian, Virgil Dobrota, Iustin-Alexandru Ivanciu, Cosmin Filip “Automation of the Infrastructure and Services for an OpenStack Deployment Using Chef Tool”, Networking in Education and Research 2016 15th RoEduNet Conference.

[10] Nishant Kumar Singh, Himanshu Nagdev and Sanjeev Thakury Himanshu Chaurasiyaz, “Automated Provisioning of Application in IAAS Cloud using Ansible Configuration Management”, 2015 1st International Conference NGCT-2015 (Next Generation Computing Technologies)

[11] "Ansible Configuration Management", 2017. [Internet Source]. Link: https://www.ansible.com/configuration-management [Last Accessed: April 2017]

[12] "DevStack Deployment", 2017. [Internet Source]. Link: https://docs.openstack.org/developer/devstack/ [Last Accessed: April 2017]

[13] "TripleO Deployment and Management", 2017. [Internet Source]. Link: https://wiki.openstack.org/wiki/TripleO [Last Accessed: April 2017]

[14] Tao Lin, Byung Chul, Yang Chao, Zhu Yaoping Ruan, Tak Shu, IBM Research Center “LOGAN: Problem Diagnosis in the Cloud Using Log-based Reference Models”, 2016 IEEE Conference.

[15] "ELK usage document", 2017. [Internet Source]. Link: https://www.elastic.co/guide/index.html [Last Accessed: April 2017]

[16] Baek Dongmyoung, Lee Bumchul “Analysis of Telemetering Service in OpenStack”, 2015 ICTC Conference (Information and Communication Technology Convergence)

[17] Monica O, M. Abhishek Sharma; Joshi, Openstack Ceilometer Data Analytics & Predictions, 2016 IEEE CCEM Conference (Cloud Computing in Emerging Markets)

[18] "Ceilometer Architecture", 2017. [Internet Source]. Link: http://docs.openstack.org/developer/ceilometer/architecture.html [Last Accessed: April 2017].