Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 05, May-2020

Review on Cloud Automation Tools

Varsha C L
4th year UG Student
Computer Science dept.
RV College of Engineering

Dr. Ashok Kumar A R
Associate Professor
Computer Science dept.
RV College of Engineering

***Abstract*— Traditionally enterprise workloads were deployed and managed using manual process. It was time consuming because of the repetitive tasks performed in terms of scaling, configuring and provisioning of the clusters, virtual machines. The cloud automation tools help us to speed the process. There are multiple cloud automation tools, whilst there is no single tool suitable for every situation. This paper summarizes the benefits offered by various cloud automation tools.**

Keywords—Cloud, Automation tools, Virtual Machine

I. INTRODUCTION

Cloud automation is a software solution which enables the developers and the IT team to install, configure and manage the cloud computing services. Thus, allows businesses to choose the right amount of resources required for cloud computing [1]. Providing services on demand is the main aim of cloud computing. But in reality, someone has to create them, keep monitoring them continuously and delete them when they are no longer needed. This can require a huge manual effort.

Cloud automation largely revolves around the Infrastructure as a code. Cloud automation processes and tools use the resource pools from the cloud to create common configuration items such as Virtual machines, virtual private networks and containers. Instances can be created and deployed using these configuration items [2]. For example, a specific number of containers can be created using a cloud automation template which can be used for a microservices application. Also used for connecting a storage and a database, virtual network configuration and creating load balancers [3].

Apart from the deployment cloud automation can be used for workload management and monitor the performance of application and workload.

II. CLOUD AUTOMATION TOOLS

A. AWS CloudFormation

The Amazon Web Services CloudFormation tools provide administrators and developers with a simple method to build set of related resources, supply and upgrade them in organized and predictable manner. CloudFormation offers sample templates, or we can build our templates to represent AWS tools, relevant dependencies our device. Upon implementation of the AWS tools, we can change and upgrade them in a managed and consistent manner, effectively adding version control for AWS infrastructure [4].

***Features*:**

1. Authoring with familiar programming:
   The cloud development kit of the AWS enables user to define applications using familiar programming languages like Typescript, Python, Java and .NET. Additionally, it enables us to provide our infrastructure using AWS CloudFormation directly from our IDE
2. Authoring with JSON/YAML:
   Using AWS CloudFormation, an entire network can be modelled in text. YAML or JSON file are used to define resources required for configuring or building AWS .
3. Safety Controls:
   The provisioning and updating an AWS infrastructure is automated by AWS CloudFormation in a secured manner. Rollback Triggers can be used to specify the CloudWatch We can use Rollback Triggers to specify the CloudWatch alarm monitored by the CloudFormation and used for the introduction of slack and replace process. If any of the alarms are breached, the whole stack operation is preceded back to deployed state by CloudFormation[5].
4. Dependency Management:
   During stack management behavior AWS CloudFormation automatically handles dependencies between our resources. We don't have to worry about specifying in which order the resource is generated, modified, or deleted. The appropriate actions to be performed for stack actions are determined by the CloudFormation for each resource [5].
5. Managing of Cross-region Cross-Account:
   AWS StackSets allows us to provide collection of tools provided by AWS with a single CloudFormation template across multiple accounts and regions. StackSets guarantees that multiple accounts and regions stacks are automatically and safely supplied, changed or removed [5].

B. Kubernetes

Kubernetes is a containerized software running and managing through a community of machines at its core stage. It has been built with methods that allow predictability, scalability and high availability for the life of containerized applications and services. You will decide how your apps operate and how they will communicate with other applications or the outside world as a Kubernetes user. You can update or uninstall your services, update gracefully, and transfer traffic between various versions of your apps to check functionality or rollback problems. Kubernetes provides primitive interfaces and platform composable that enable high levels of flexibility, power, and confidence to define and manage your application.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 05, May-2020

Architecture:

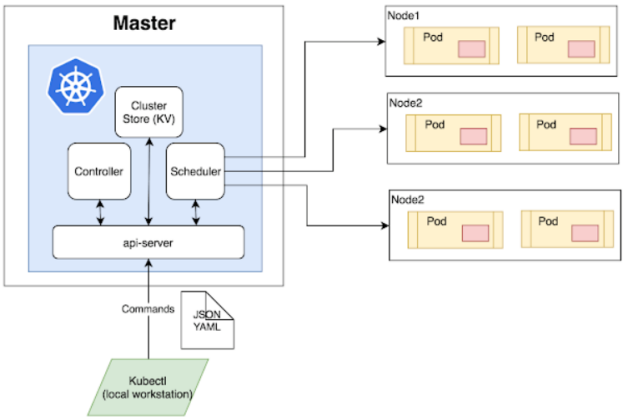

Fig.1 Kubernetes Architecture

It is important to know how Kubernetes can have these functionalities, how it is structured and organized at a high level. Kubernetes can be regarded as a layer-built network, with the complexity at the lower levels being resumed by any higher layer [6].

1. Master Node:

The master node is the first and most critical part for cluster management of Kubernetes. An administrative gateway for activities of all kinds. In the cluster, fault tolerance can be monitored by multiple master nodes. There are many components to the master node such as API Server, Scheduler, Controller Manager and ETCD.

API Server: Acts as entry point for all the commands (REST) used in cluster.

2. Scheduler:

The programmer running node functions. Stores information about the usage of resources by each node. The company shall divide the workload.

It also lets you monitor how cluster nodes use the working load. It helps you to position the workload on the available resources and to accept the workload.

3. Master/Slave Node:

These are the worker nodes that play an important role in providing the necessary services for networking and collaboration between containers, allowing you to allocate resources to scheduled containers. worker nodes are also important.

Kubelet: gets an API server Pod setup and ensures the containers listed are up and running.

Docker Container: These Containers run on worker's nodes that operates on the *Kube-proxy* (the cube-proxy helps in balancing the load and network proxy for the output of a single working node) pods that are configured.

Kubernetes essentially binds multiple individual virtual and/or physical machines in cluster that is linked to each server through a common network. This cluster is the physical framework for configuring all the modules, functions and workloads of Kubernetes**.**

In the Kubernetes ecosystem, the machines in each cluster have a particular function. The master machine acts as a machine (or as a small community in highly accessible installations). API is provided to users and customers, safety checks on the other servers, how best to disperse and delegate work ("scheduling") and orchestrate coordination with other components, as the brain for the cluster. The server serves as a portal and brain. The master server serves as the main point of contact with the cluster and mainly supports the centralized logic provided by Kubernetes [7].

Certain computers in the cluster are known as nodes: servers with local and external resources for workload acceptability and operation. The software and services are made to run in containers to help with isolation, flexibility and management so that node is fitted with the runtime of its container (e.g. Docker).The containers are either created or destroyed based on the instructions received by the node from the master. Network rules are changed according to traffic routes and transit

C. Puppet

An open sourced configuration management tool, used for private , public and hybrid clouds. It provides its own configuration language Puppet DSL (Domain -specific language). The system configurations and infrastructure as code are defined using a DSL. Puppet enterprise orchestrates the task-based multi-device management and command execution. It provides the GUI console to classify and manage all the deployed cloud machines [8].

Architecture:

Puppet is based on master-slave architecture. The client and server are interconnected by the secure socket layer. The puppet architecture has following components [9]:

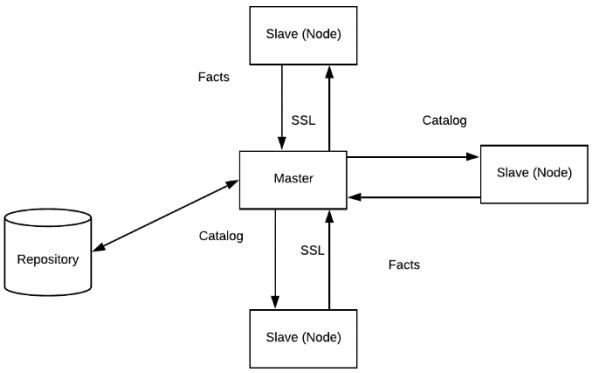

Fig. 2: Simple Puppet architecture

1. Puppet Master:

Puppet master is a Linux based system which handles all the configuration related process in the form of puppet codes. The SSL certificates are checked and marked by the master.

2. Puppet Slave:

Puppet Slave are working systems used by the client. Puppet master maintains and manages the slave. The Puppet agent daemon service runs inside the slave.

3. Repository:

Repository stores the node and server related configuration. Puppet keeps the official bundle archives of operating systems. Puppet collections help in gathering the majority of software required for the utilitarian Puppet deployment.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 05, May-2020

4. *Catalog:*
The compiled format of configuration and manifest files written in Puppet are called as catalog. It defines the state and dependency data for all the assets that ought to be overseen by hub in a specific request.

5. *Facts:*
The Facts are the key-value pair that contain the information about the node and master machine. Facts are used for determining the state of any slave as facts represent the client states such as operating systems, IP, network interface.

Features

1. *Idempotency:*
Puppet supports idempotency, thus same set of configurations can be run multiple times on the same machine[10]. Puppet basically checks for the current status of the target machine and makes changes only if there is change in configuration
2. Cross-platform:
Puppet helps configuring a the system. Implementation details is not taken into consideration as it is handled with the help of Resource Abstraction Layer.

D. Terraform

A tool created by HashiCorp, helps in provisioning the infrastructure as code. HashiCorp configuration Language is used to provision a datacenter infrastructure. Used a tool for versioning, building, and changing infrastructure efficiently. Many existing service providers are managed by Terraform with the help of custom in-house solutions [11]

Features:

1. Infrastructure as Code:
The infrastructure can be described using high-level which is configurable and reusable
It allows us to create a blue-print of infrastructure which can be versioned too.
2. Execution Plans:
Terraform provides as step named as planning step which allows user to know the complete details of execution when the apply is called.
3. Resource Graph:
The resources which can created or modified independently are parallelized in Terraform by building graphs. This help the terraform to build the infrastructure faster and efficiently.
4. Change Automation:
Terraform uses the execution plan and the resource graph to make a complex change. Thus, allowing a complex modification with minimum human errors and interaction.

How Terraform Works

Terraform architecture which is plugin-based , allows the developers to extend Terraform by either writing new plugins or modify the existing one. Terraform has two main parts: Core and Plugins [12].

Terraform Core:

GO programming based command line tool. It helps in creating infrastructure as code, resource state management, resource graph construction, execution of plan and communication over RPC with plugins.

Terraform Plugins:

Invoked by the Terraform core over Remote Procedure call as binaries which are executable. Terraform plugins are used for authentication, defining resources and make API calls required for libraries initialization.

E. SaltStack

A cloud automation tool that uses the Infrastructure as code for configuration and deployment automation. It is open-source and python-based software used for remote execution, configuration management and cloud control. Salt supports multiple cloud providers such as Azure, AWS, OpenStack, IBM Cloud and VMware. Salt Stack provisions server and infrastructure with help of central repository.

Architecture:

SaltStack has highly modular design configured to work with multiple servers ranging from network system in local to data centers deployment. It has a simple client-server model with multiple daemons working in co-ordination. Salt architecture composes of following components:

1. SaltMaster:
A Master daemon that sends various commands and configuration to slave daemons.
2. SaltMinions:
A slave daemon that receives commands and configuration from the master daemon.
3. Execution:
Monitoring in real time using the adhoc and module commands executed against the slave daemons.
4. Formulas:
These are the states used for various tasks such as starting a service, monitoring permissions and installing packages.
5. Grains:
System used for detecting various information and storing in RAM.
6. SaltCloud:
Cloud hosts are monitored by the salt cloud
7. SaltSSH:
Used to SSH on systems and execute various commands.
8. Runners: Applications on the master end used using the run command of salt.

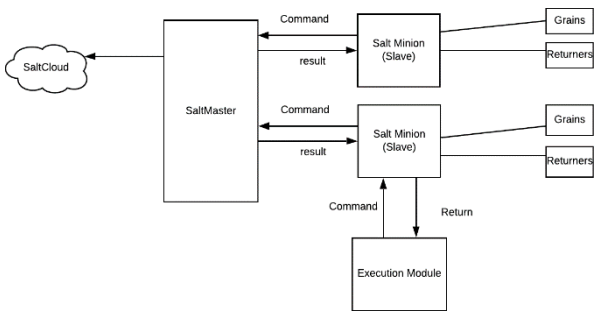

Fig 3: Simple architecture of SaltStack.

Features:

1. Scalable and fault tolerant:
Salt stack has high fault tolerance, it can connect to multiple masters at once and used YAML to configure for

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 05, May-2020

all the masters at once. Salt master can handle around ten thousand minions.

2. Python API:

Salt is a python based and provides a modular, extensible programming interface to configure and monitor applications.

3. Authentication:

Salt provides and secure SSH key pairs used for authentication.

4. Execution model:

Salt provides tool which can run commands in remote systems parallelly.

III. CONCLUSION

This paper discussion the features, architecture for each of the cloud automation tool, also defines the importance of same. The future work involves the creation of infrastructure in each of the tool discussed and deploy them. Also analyse the performance, reliability and scalability of the deployed system.

IV. REFERENCE

[1] Cloud automation [Online]: https://searchcloudcomputing.techtarget.com/definition/cloud-automation
[2] R. Zhang, Y. Shang and S. Zhang, "An Automatic Deployment Mechanism on Cloud Computing Platform," 2016 IEEE 6th International Conference on Cloud Computing Technology and Science, Singapore, 2016, pp. 511-518.
[3] S. Callanan, D. O'Shea and E. O'Regan, "Automated Environment Migration to the Cloud," 2016 27th Irish Signals and Systems Conference (ISSC), Londonderry, 2016, pp. 1-6.
[4] Amazon Web services white paper on Infrastructure as code:https://d0.awsstatic.com/whitepapers/DevOps/infrastructure-as-code.pdf
[5] CloudFormation details [Online]: https://www.aws.amazon.com/cloudformation/.
[6] L. A. Vayghan, M. A. Saied, M. Toeroe, and F. Khendek, "Deploying Microservice Based Applications with Kubernetes: Experiments and Lessons Learned," in 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), 2018, pp. 970–973.
[7] "Documentation on Kubernetes, https://kubernetes.io/docs/home/.
[8] J. Hintsch, C. Görling and K. Turowski, "Modularization of Software as a Service Products: A Case Study of the Configuration Management Tool Puppet," 2015 International Conference on Enterprise Systems (ES), Basel, 2015, pp. 184-191.
[9] Puppet architecture documentation https://puppet.com/docs/puppet/latest/architecture.html
[10] Puppet features documentation: https://puppet.com/docs/pe/2018.1/pe_new_features.html
[11] Terraform https://www.terraform.io/intro/index.html
[12] Terraform Architecture [Online]: https://www.terraform.io/docs/enterprise/before-installing/reference-architecture/index.html
[13] SaltStack [Online]: https://www.saltstack.com/
[14] SaltStack architecture: https://www.saltstack.com/blog/salt-air-25-overview-salt-architecture/
[15] J. O. Benson, J. J. Prevost and P. Rad, "Survey of automated software deployment for computational and engineering research," 2016 Annual IEEE Systems Conference (SysCon), Orlando, FL, 2016, pp. 1-6.