International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2020): 7.803

# Categorize & Compare Cloud Automation & Devops Tools

**Pardeep Singh Virdi**

2nd Year, M. Tech, Computer Science, GITM Gurgaon, India

**Abstract:** ***Every enterprise either using or wants to migrate to cloudplatform and reason for this attraction is benefits provided – capital expense reduction, scalability, flexibility, quickly deploy new technologies i.e. Every cloud has a silver lining. cloud platform also has a positive impact and at the same time, some challenges are faced by the enterprises - Security and Privacy, Management of multi-cloud environment , Vendor lock-in, Interoperability i.e. Traditionally enterprise workloads were deployed and managed using manual process which was full of challenges that can put the enterprise at risk. Challenges: They are inefficient and often fraught with errors, expose security vulnerabilities, repetitive and manual steps delay the workload's availability. To mitigate such challenges, organizations need to apply cloud automation. Cloud automation tools enable enterprises to automate the manual efforts associated with managing cloud computing workloads – allowing you to manage traditional applications andinfrastructure in a more competitive, agile, and scalable way.***

**Keywords:** Cloud, Automation tools

## 1. Introduction

Cloud automation is process of making use of technology tools and resources to reduce manual efforts in provisioning and managing cloud computing tasks.

Efficient use of cloud resources and no security pitfalls and error-prone workflows which was in manual processes

DevOps Tools reduce the effort by bringing in a new flow across SDLC and focuses key aspects of your DevOps environment by automating the process chain using feature Build, Test, Deploy and Release

**Use Cases for Cloud Automation**

1) Infrastructure provisioning
2) Identity provisioning and management
3) Application deployment
4) Monitoring and remediation
5) Multi-cloud management
6) Data discovery and classification

## 2. Categorization Norms

We can categorize cloud automation tools on more than one parameter and some of the tools are not mutually exclusive so there are case when two tools are in different category in reference to one parameter and falls in same group for other parameter below are few categories on basis of different parameters

**Cloud platform specific vsplatform independent**

Platform Independent means it won't get effected by the System Software.It can be run on other clouds. Cloud Platform specific means it get effected by changes in cloud host

**Immutable vs mutable**

Immutable infrastructure is a process of replacing infrastructure components rather than modifying their configuration. This approach addresses common problems encountered when modifying configuration over time, namely configuration drift and Snowflake Servers. The immutable approach will, rather than modifying configuration, create a new instance with the new configuration, switch traffic to it, and then switch over. This makes deployments more predictable and recreatable, but isn't realistic for frequently reconfigured components like firewalls. Of the two products, Terraform is more compatible with this approach.

**Procedural vs Declarative**

A procedural approach to automation is similar in concept to what a recipe is for cooking. A procedural automation consists of a listing of each step required to achieve the goal of the automation. Procedural systems are completely general in application, capable of any automation task the underlying hardware is capable of.

Declarative automation is an approach where the end goal or attributes of the automation is described, and through a purpose built program it is interpreted in order to realize the goal. Declarative systems are not general purpose, and are applied to specific domains. Ultimately, **declarative** systems execute procedural code, and as such are at a higher level of automation.

**Agent vs Agentless**

An agent-based tool, requires you to deploy software on the system, which **performs** the task.

Agentless tools eliminate the problems inherent when deploying and managing software. Though they often don't provide enough information.

Apart from these we normally explain tools on the basis of their tasks

**Categorize explanation and comparison of tools**

**Automation and configuration management**

- **Terraform**: infrastructure provisioning;

- **Ansible**: management of the slave servers configuration;
- **Puppet**: management of the slave servers configuration;
- **Salt**: management of the slave servers configuration.

**1) Terraform**

Terraform is an infrastructure as code service/tool that allows user to build, change, and version infrastructure with safety and efficient manner .Suitable for Components such as compute instances, storage i.e s3, and networking, as well as components such as DNS entries, SaaS features, etc . Terraform is multiplatform service/tool and capable of handling existing service providers and custom in-house solutions.

**Features:**

a) **InfrastructureasCode**:
The infrastructure is configurable and reusable with use of high-level language in Terraform
It provides the facility to create a blue-print of infrastructure and can be versionedtoo.

b) **ExecutionPlans**:
- Terraform has three stages- Plan
- Plan- Creates an execution plan
- Apply- executes the actions proposed in plan stage
- Destroy- destroy remote object managed by config

c) **Resource Graph**
Parallel creation of independent resources using building graph. Parallelization help to build the infrastructure faster and efficiently.

d) **Change Automation**
For complex changes Terraform uses the execution plan and the resource graph. Thus, allowing minimum human errors and interaction

**2) Ansible**

Ansible is an open-source automation tool, or platform, used for IT tasks like configuration management, application deployment, intra service orchestration, and provisioning. Automation is crucial lately, with IT environments that are too complex and sometimes got to scale too quickly for system administrators and developers to stay up if they had to do everything manually. Automation simplifies complex tasks, not just making developers' jobs more manageable but allowing them to focus attention on other tasks that add value to a corporation. In other words, It reduce the effort as well as cost And Ansible, as is rapidly rising to the market of cloud automation tools.

Ansible Features
a) Configuration Management
b) Application Deployment
c) Orchestration
d) Security and Compliance
e) Cloud Provisioning

**3) Puppet**

Puppet is a infrastructure provisioning tool that helps you manage and automate the configuration of servers. When you use Puppet, you define the specified state of the systems in your infrastructure that you simply want to manage. You are doing this by writing infrastructure code in Puppet's Domain-Specific Language (DSL) — Puppet Code — which you'll use with a good array of devices and operating systems. Puppet code is declarative, which suggests that you simply describe the specified state of your systems, not the steps needed to urge there. Puppet then automates the method of getting these systems into that state and keeping them there. Puppet does this through Puppet primary server and a Puppet agent. The Puppet primary server is that the server that stores the code that defines your required state. The Puppet agent translates your code into commands then executes it on the systems you specify, in what's called a Puppet run.

***Features of Puppet Tool***

Below are the most important features of Puppet.

**Idempotency**

Puppet supports Idempotency which makes it unique. almost like Chef, in Puppet, one can safely run an equivalent set of configuration multiple times on an equivalent machine. during this flow, Puppet checks for the present status of the target machine and can only make changes when there's any specific change within the configuration. Idempotency helps in managing any particular machine throughout its lifecycle ranging from the creation of machine, configurational changes within the machine, till the end-of-life. Puppet Idempotency feature is extremely helpful keep the machine updated for years instead of rebuilding an equivalent machine multiple times, when there's any configurational change.

**Cross-platform**

In Puppet, Puppet resources used by Resource Abstraction Layer (RAL), one can target the specified configuration of system without worrying about the implementation details and how the configuration command will work ion target system, which are defined in the configuration files.

**Comparison table of Ansible, Puppet and Terraform**

| *Point of Difference* | *Ansible* | *Puppet* | *Terraform* |
|---|---|---|---|
| Management and Scheduling | instantaneous deployments are possible.<br><br>Scheduling:<br>Ansible Tower, the enterprise version, has the capabilities while the free version | push and pulls configuration, which is written in Puppet's language.<br><br>Scheduling: Puppet's default settings allow it to check all nodes for desired state | In Terraform, resource schedulers work similarly for all providers. |
| Ease of Setup and Use | simpler to install and use. Master without agents, running on the client machines. Agentless. Ansible uses YAML syntax in Python language, | model-driven, client-server or agent-master model. meant for system administrators. Installation times ten to thirty minutes approx | Terraform is also simpler to understand from setup as well as usage point of view. Allow Proxy server use. |
| Availability | secondary node exist in case an active | one or more masters | NA in Terraform's case |

| | | | |
|---|---|---|---|
| | node falls. | | |
| Scalability | Scalability is easier to achieve | Scalability is less easy to achieve | Scalability is comparatively easily achieved |
| **Modules** | Ansible Galaxy is repo or library for Ansible.<br><br>No separate sorting capabilities. Needed manual intervention. | Puppet Forge : puppet repo or Library . 6000 modules. Users have capable of mark puppet modules as approved or supported by Puppet which saves time | modules allow users to abstract away any reusable parts. Which can be configured once and can be used everywhere. It thus enables users to group resources, as well as defining argument |
| **GUI** | Introduced as a command-line tool. the enterprise version has UI, but not mature | Puppet's GUI is far betterthan of Ansible, Can perform many complex tasks. | Only third party GUIs |
| **Support** | Enterprise version has two levels of professional support. AnsibleFest is a big gathering of users and contributors, held annually. Puppet has big community | Dedicated support portal, and knowledge base. Standard and Premium support available. The Puppet community produced "state of DevOps" report annually. | a web portal is provided , having direct access to HashiCorp's support channel. |

## 3. Monitoring and Alerting

**Prometheus**
**ELK**

### A. Prometheus

Prometheus built at SoundCloud is an open-source systems monitoring and alerting. Since its inception in 2012, many companies and organizations are using Prometheus, and the project is in active development and community. In 2016 Cloud Native Computing Foundation include Prometheus as the second hosted project, after Kubernetes.

**Prometheus features:**

- A multi-dimensional data model
- PromQL, a flexible query language
- Distributed storage have no reliance; autonomous single server nodes
- A pull model over HTTP for time series collecttion
- Intermediary gateway to support pushing time series
- Service discovery or static configuration for target discover
- Graphing and dashboarding support mode

### B. ELK

The ELK Stack is a combination of three open-source products- Elasticsearch, Logstash, and Kibana. They all developed, managed, and maintained by the Organization Elastic.
**E** represents for Elastic Search: used for storing logs
**L** represents for LogStash: used for both shipping as well as processing and storing logs
**K** represents for Kibana: is a visualization tool (a web interface) which is hosted via Nginx or Apache

**Role in centralise logging:**
ELK is useful in attempting to identify issue with servers or applications. It enablesuser to search all application logs in a single place. It supports to find issues that occur in multiple servers by connecting their logs in particular time frame.

**Comparison table of Prometheus and Elastic search**

| Name | Elasticsearch | Prometheus |
|---|---|---|
| Primary db model | Search Engine | Time series Database Management System |
| Secondary db model | Document store Spatial DBMS | |
| Cloud-based only | No | No |
| Implementation language | Java | Go |
| Data scheme | Schema-free | yes |
| Partitioning methods | Sharding | Sharding |
| Concurrency | yes | yes |

**Secrets management**

- **Vault:** Vault allows the static and dynamic organization of secrets.
- **Secrets:** Kubernetes' secret management service.

### C. Vault

Vault is a tool for securely accessing secrets. A secret is anything that you simply want to tightly control access to , like API keys, passwords, or certificates. Vault provides a unified interface to any secret, while providing tight access control and recording an in depth audit log.

**Vault key features:**

- **Secure Secret Storage**: Arbitrary key/value secrets stored in Vault. Vault encrypts sensitive info and then after stored them to persistent storage, so in case of unwanted access of secret hacker will not be able to use it. Vault can write to a number of locations like Consul, disk and more.
- **Dynamic Secrets**: Vault can generate secrets on-demand for few systems, such as AWS or SQL databases. , when an application requires to access an S3 bucket it asks Vault for credentials, and Vault will generate an AWS keypair with valid permissions on demand. when an application ask vault to get access of resources like S3 ,After dynamic secrets creation, Vault also will automatically revoke them after the lease is up.
- **Data Encryption**: encryption and decryption happened without storing data. This allows security teams to define encryption arguments and developers to store encrypted data in a fields such as SQL without designing their own encryption methods

- **Leasing and Renewal**: Vault Secret have a lease associated . When lease ended, Vault will automatically revoke that secret. Clients can renew leases via built-in renew APIs.
- **Revocation**: for secret revocation Vault has built-in support. Vault can revoke not only a tree of secrets, for example all secrets read by a particular user, or all secrets of exact type.

**Secrets**

Kubernetes Secrets allow you to store and manage sensitive information, like passwords, OAuth tokens, and ssh keys. Storing confidential information in a Secret is safer and more flexible than putting it verbatim in a very Pod definition or in a very container image.

A Secret is K8s object that contains sensitive data like password, a token, or a key. Such critical information might otherwise be describe in a Pod specification or in an image which is not a secure way. Secrets can be created either by user or system.

Using Secrets gives you more flexibility in a Pod Life cycle definition and control over how sensitive data is used. It provided security against the risk of exposing the data to unauthorised users.

1) Secrets are name spaced objects.
2) Secrets used by container either by mounting data volume or as environment varables
3) Secret data stored in tmpfs in nodes hosting containers
4) API server stores secrets as plain text in etcd
5) A per-secret size limit of 1MB

**Comparison Vault and Secret**

| | K8s Secret | Vault with K8s | Vault with K8s auth method |
|---|---|---|---|
| Secret zeroprovisioning requirement to our APP/CLUSTER for bootstraptrust? | Yes, DB encryption key and cert required | K8s vault controller need to be authenticated with Vault | Auth need to be setup between K8s and vault |
| Is it cloud provider agnostic? | Yes, but limited to app running on k8s | Yes, but Relies on Hashicorp vault | Relies on Hashicorp vault |
| Amount of effort to integrate into app | Litttle | Moderate | Little |
| Recommended scenario | Good for containers orchestrated By k8s | Good for secrets required to be shared across platform | Best for secret required to shared across platform |

## 4. Conclusion

This paper discussion the features, comparison for each of the cloud automation tool in same category, also defines the importance of same. The future work involves the cost estimation and security challenges of infrastructure in each of the tool discussed. Also analyse the performance, reliability and scalability of the deployed system.

## References

[1] Vault: https://www.vaultproject.io/docs/what-is-vault
[2] Terraform : https://www.terraform.io/intro/index.html
[3] Review on Cloud Automation Tools International Journal of EngineeringResearch & Technology (IJERT) ISSN: 2278-0181 Vol. 9 Issue 05, May-2020
[4] Prometheus https://prometheus.io/docs/introduction/overview/