Introduction to Terraform

SUSHMITA SIVAPRASAD[1]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: sushsiva@umail.iu.edu*

project-000, April 12, 2017

This paper gives a brief introduction on Terraform and how infrastructure can be used as a code, which is the building block of Terraform. It is written in go scripting code and it is a server provisioning tool where we can specify what is the goal we require and Terraform creates steps of tasks on how to reach the goal. This paper provides information on what are the use cases of Terraform and how it differentiates itself from other tools. Resources for learning Terraform in much more detail has been provided as well.

Keywords: Terraform, IAC, Go code, Ansible

https://github.com/cloudmesh/sp17-i524/tree/master/paper2/S17-IR-2038/report.pdf

1. INTRODUCTION

Terraform is an open source tool created by HashiCorp. It is an infrastructure management tool written in Go programming language.It allows users to safely and predictably create , change and improve the production infrastructure and codifies the API's into a declarative configuration file that can be shared among other users as well[1]. The tool can even be treated as a code whereby we can edit, review and create versions at the same time. The tool is popular in allowing users to create a customized in-house solution. It allows users to build an executive plan which can be used for the purpose of creating applications and implementing the infrastructure[2].

2. INFRASTRUCTURE AS A CODE

Infrastructure as a code or IAC allows us to write and execute a code in order to define, deploy and update the infrastructure. There are 4 categories of the IAC tools,

2.1. Ad hoc scripts

Ad Hoc scripts allows to automate anything. We can use any scripting language and break down the tasks we were doing manually and execute the script on the server. Ad hoc script allows one to write the code in the required manner[3].

2.2. Configuration management tools

Some of the main configuration management tools are Chef, Ansible, SaltStack which are designed to install and manage software on any of the existing servers. These tools are designed to manage large number of remote servers. Ansible playbook can be used to configure multiple servers in a parallel mode[3]. A parameter called serial can be set in the playbook from which a rolling deployment can be done and the server will be updated batch wise. Tools such as Ansible are idempotent, that is they run correctly no matter how many times we run the code[3].

2.3. Server templating tools

Server templating tools are Docker, Packer and Vagrant. A server templating tool allows to create an image of a server that captures a snapshot image of the operating system, the software and the files in it. The image of the server can then be distributed across all of the servers using Ansible[3].

2.4. Server provisioning tools

Server provisioning tools includes Terraform, Cloudformation, Openstack Heat[3]. These tools allows to create a server by themselves. These tools can also be used to create databases, caches, load balancers, queues, firewall settings and other aspects from the infrastructure[3].

3. WORKING OF TERRAFORM

Infrastructure is the building block on which the application is created. Terraform provides us a declarative execution plan for building and running applications and infrastructure. All that we are required to do in a declarative state is we declare the required end of the state. The tool will take care of the steps required to reach the goal state. It allows the user to not be bothered about the commands to run or the settings required to change. The user has to declare the resources using a graph-based approach in order to model and apply the desired state[2]. The Go code allows Terraform to compile down into a single binary code for each of the supported operating systems. This binary is used to deploy infrastructure from our pc or a server and we would not require an additional infrastructure to make

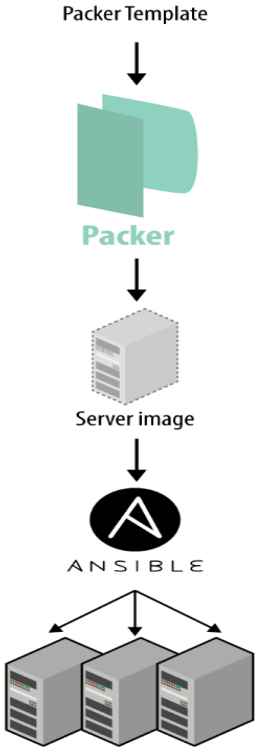

Fig. 1. Packer is used to create an image of a server which can be installed using Ansible across all other servers [3]

that happen. The binary makes API calls on behalf of us to one or more of the providers such as Azure, AWS, Google Cloud, etc. In this way terraform lets users to use the infrastructures provided by these providers as well as the authentication mechanism we are already using with these providers [3]. Terraform allows users to deploy interconnected resources across various cloud providers at the same time. It translates the contents of the configurations into API calls into the cloud providers as shown in figure 2.

4. USE CASES

4.1. Multi-Tier Application

It is useful for building multi-tier applicatin which consists of web, cache, application, middleware and tiers of database. Terraform helps to build N-tier applications. Each of the tier can be configured in Terraform and left independent and isolated which can be easily scaled and managed by the code[2].

4.2. Disposable Environments

In order to test a new application before it has been sent for production , a staging or QA Environment is required in order to avoid complexity and confusion. For such a situation terraform can be very handy, where the production environment can be codified and shared along with staging[2]. It can even create new environment for it to test on and which can be later disposed of , keeping only what is required. Terraform makes it easy to maintain paralle environements and easily dispose them of [2].

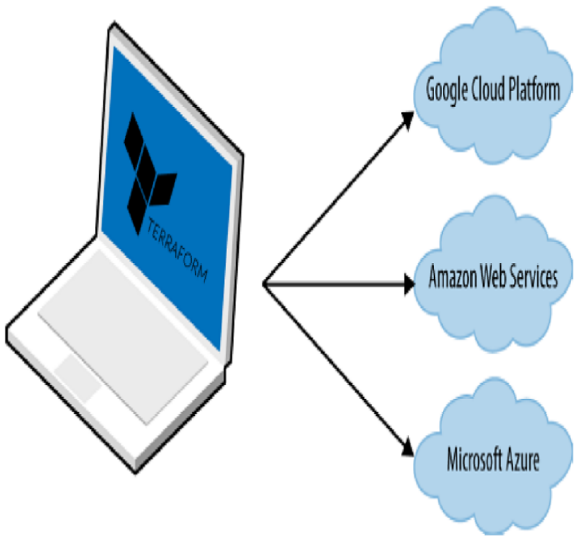

Fig. 2. Figure depicting terraform transforming the configurations to API calls into the cloud providers [2]

4.3. Multi-Cloud Deployment

In order to increase the fault tolerance, infrastructure is spread across multiple clouds to continue it's progress irrespective of any faults. The current multi-cloud deployement consists of cloud specific tool for infrastructure management. Terraform allows a single configuration to be used across multiple providers and also handle any cross-cloud dependencies[1].

5. ADVANTAGES OF USING TERRAFORM

Terraform allows the existing tools to focus on its strength such as bootstrapping and initializing resources. It makes the infrastructure deployment easy by focusing on a higher level of abstraction of the datacenter and also allowing the same codification for the tools. It is cloud agnostic and allows multiple providers and services to be combined [4]. It separates the planning and execution phase. It generates an action plan when we provide a goal state, which keeps getting updated when we add or remove new resources. The terraform graph feature allows the user to visualize the plan in order for them to know exactly the effects of the changes before they get implemented [4].

6. EDUCATIONAL MATERIAL

Hashicorp has provided a documentation for describing Terraform, it's download and installation procedure. There has been 2 books published by authors Yevgeniy Brikman[3] and James Turnbull[2] which gives a very detailed insight about the working and usage of Terraform.

7. CONCLUSION

Terraform simplifies the process of creating applications and implementing the infrastructure by just specifying the required goal. It uses Go code which allows Terraform to compile down into a single binary code for each of the supported operating systems. We can create N- tier applications with ease by just

providing the resources and the end state of the required application. It allows multiple user to work on a cloud-agnostic platform thus being a very versatile tool.

8. ACKNOWLEDGEMENT

A very special thanks to Professor Gregor von Laszewski and the teaching assistants Miao Zhang and Dimitar Nikolov for all the support and guidance in getting this paper done and resolving all the technical issues faced. The paper is written during the spring 2017 semester course I524: Big Data and Open Source Software Projects at Indiana University Bloomington.

9. AUTHOR BIOGRAPHY

Sushmita Sivaprasad is a graduate student in Data Science at Indiana University under the department of Informatics and Computing. She had completed her bachelors in Electronics and Communication from SRM University , India and her master's in International Business from Hult International Business School, UAE.

REFERENCES

[1] "Introduction to terraform," Web Page, Oct. 2014, accessed: 2017-03-25. [Online]. Available: https://www.terraform.io/intro/index.html

[2] J. Turnbull, *The Terraform Book*, 110th ed. Turnbull Press, Nov. 2016, accessed:2017-03-25.

[3] Yevgeniy Brikman, *Terraform: Up and Running:Writing Infrastructure as Code*, 1st ed. O'reilly media, Mar. 2017, accessed : 2017-2-24. [Online]. Available: http://shop.oreilly.com/product/0636920061939.do

[4] "Terraform vs. other software," Web Page, Oct. 2014, accessed: 2017-03-27. [Online]. Available: https://www.terraform.io/intro/vs/cloudformation.html