CLOUD SECURITY AUTOMATION

[1]Ankur Sharma, [2]Dr. Neetu Sharma
[1]M.Tech Scholar, Department of C.F.I.S, G.I.T.A.M, Kablana, Jhajjar, MDU, Rohtak, Haryana
[2]HOD, Department of C.F.I.S & C.S.E, G.I.T.A.M, Kablana, Jhajjar, MDU, Rohtak, Haryana

Abstract: Business & its computations are moving from datacenter to cloud at a very rapid rate and industries are choosing serverless computation platform where they just need to push already built code into a serverless service which is based on the cloud service provider. Due to the absence of the data link layer and its protocol in almost all public cloud, it is much difficult to implement inline legacy security solutions like firewall & IPS. Another challenge is to meet the scaling behavior of the modern application where traffic density increase and decrease instantaneously. These challenges demand several new research in security and automation domain for cloud Infrastructure and this can be determined by scalable security solution which can expand at the same scaling speed as that of application and hence security automation is required to overcome these challenges. The goal of this paper is to provide directions for security automation especially for the cloud environments.

***IndexTerms* – Cloud Security, Security Automation, Cloud Threats**

I. INTRODUCTION

The impact of security breaches can include remediation costs, reputation damage and massive negative impact on a company's customer base and hence reduced customer confidence. Organizations are transforming their business by moving on the cloud. As cloud provides several advantages like Fault resilient, measured service, Scalable and high performance, reliable, storage space, competitiveness, flexible, on-demand, and self-service. The modern application demands scalable resources these days to meet flash sale kind of traffic and hence companies require rapid expansion of network and infrastructure. One of the best examples is to expand and upgrade existing infrastructure up to the multiple times of its current capacity for an hour or few. This rapid expansion of large infra (servers, load-balancer, etc.) requires a number of human resources and always this is not possible to accomplish the requirement due to unavailability of ample time. This requirement can be easily accomplished by one technique known as security automation. In the enterprise of the present, when a large number of services is involved in the business of companies, there is a need for structure and consistency as well as a need for rapid deployments. Companies' services usually consist of several parts, two of them being software and network infrastructure, e.g., servers. Many companies tend to use cloud services either for themselves or to provide their services to another enterprise. In cloud environments, rapid deployments of Infrastructure is normal aspects as part of auto-scaling and hence the goal of this research arises to provide rapid deployment of security using automation.

II. THREAT TO CLOUD SECURITY

Over just four months in mid-2019, there were multiple high-profile breaches involving public cloud environments. Cloud Security is a shared model between customer and cloud service provider (like AWS, AZUR, and GOOGLE) as depicted in "Fig. 1". When a customer decides to use cloud services, locally stored data will be sent to a remote datacenter usually known as Availability Zone or AZ. This data in the remote data center can be accessed or managed with the help of services provided by the cloud service provider.

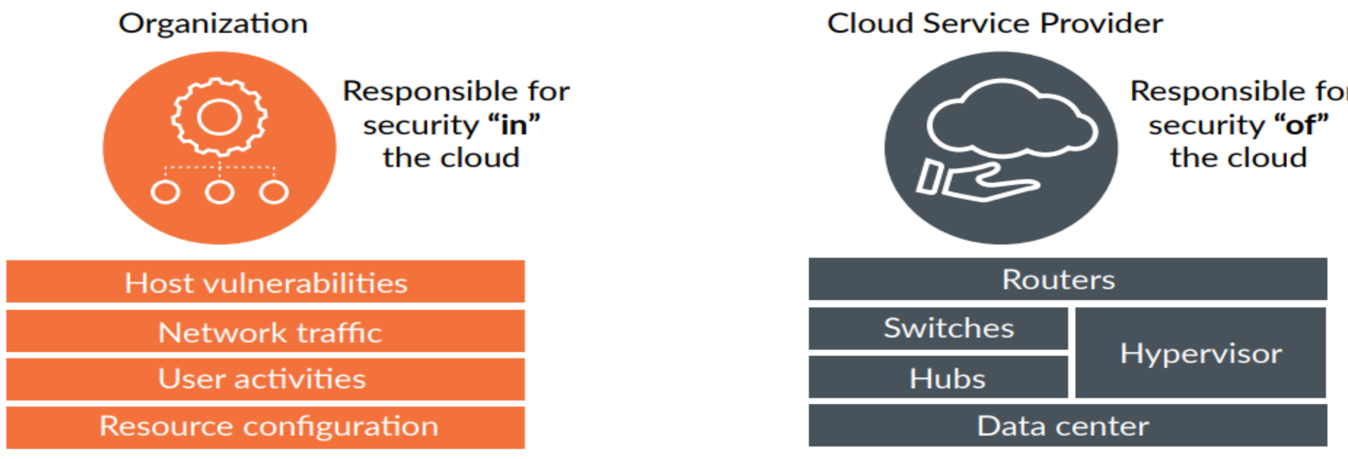

Figure 1 – Cloud Security Model

This makes clear that for a user to store or process a piece of data in the cloud, he/she needs to transmit the data to a remote server over a channel (internet or a kind of storage media). This data processing and storage need to be done with utmost care to avoid data breaches. Most high-profile data breaches in a cloud environment that took placed recently can be categorized as depicted in "Fig. 2"

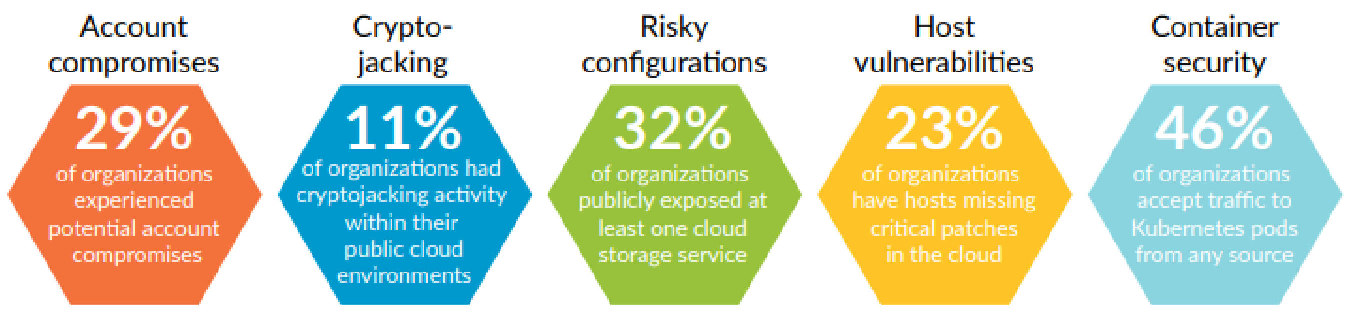

Figure. 2 – Recent Security Breaches in Cloud

If proper security measures are not implemented to the data transmitted and operated on the cloud, the data is at higher risk than when stored or operated in local repositories. A malicious user who wants to gain access to transmitted data in the cloud can do that by tapping into the connection between the user and remote location. He can also hack into a user's account and get access to sensitive information by creating another account in the same service provider with malicious intent. As shown in "Fig. 3" Since cloud computing provides different services to choose from for diversified group of users, the possibility of having data at risk when working in cloud computing systems is huge.

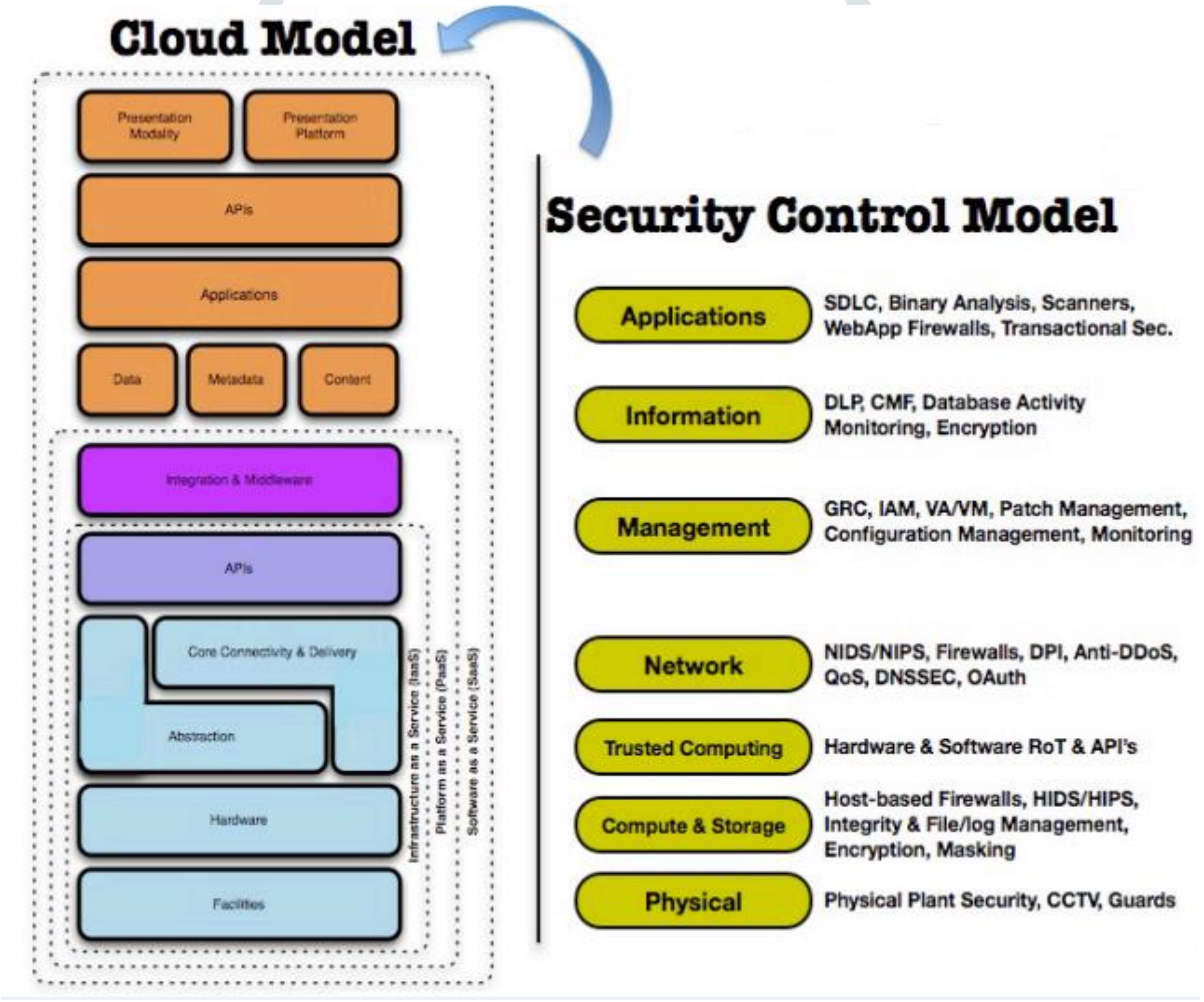
Cloud Model
Presentation Modality
Presentation Platform
APIs
Applications
Data
Metadata
Content
Integration & Middleware
APIs
Core Connectivity & Delivery
Abstraction
Hardware
Facilities
Infrastructure as a Service (IaaS)
Platform as a Service (PaaS)
Software as a Service (SaaS)
Security Control Model
Applications
SDLC, Binary Analysis, Scanners, WebApp Firewalls, Transactional Sec.
Information
DLP, CMF, Database Activity Monitoring, Encryption
Management
GRC, IAM, VA/VM, Patch Management, Configuration Management, Monitoring
Network
NIDS/NIPS, Firewalls, DPI, Anti-DDoS, QoS, DNSSEC, OAuth
Trusted Computing
Hardware & Software RoT & API's
Compute & Storage
Host-based Firewalls, HIDS/HIPS, Integrity & File/log Management, Encryption, Masking
Physical
Physical Plant Security, CCTV, Guards

Figure 3 – Mapping of Cloud Model with Security Control Model

From the above discussion in "Fig.3", it is summarized that security challenges with cloud adoption and cloud interoperability need to be addressed first, before implementing cloud computing in organizations. A non-exhaustive search on cloud computing challenges also reveals that most of the organizations consider security as an important challenge that needs to be addressed. Security of cloud workloads is customer responsibility. In software as a service and platform as a service models, the responsibility of security is of the provider. Customer doesn't have any concern of security of its data. However on the other side, in Infrastructure as a service model, security is a shared responsibility of customer and provider. The provider needs to take care of underline infra and OS.

Through 2020, 95 percent of cloud security failures will be customer fault as per the latest Gartner report. Configuration of the cloud platform is completely managed by the customer and DevOps user are not so much worry about security. Devops engineer wants to make his application up and running on a rapid path and ignoring security controls while implementation and during software development.

III. HOW SECURITY AUTOMATION WORKS

In the past, a number of IT professionals was not as large as it is today. Over the years the complexity of systems and IT grew. Professions as a system administrator, System engineer as well as more specialized jobs as Security specialist, threat analyst or malware researcher. With this trend the need for automated processes became apparent. System administrators as a part of operations were responsible for keeping systems up and running. “Fig. 4” shows how Terraform code can be used to build security infrastructure to any of cloud provider.

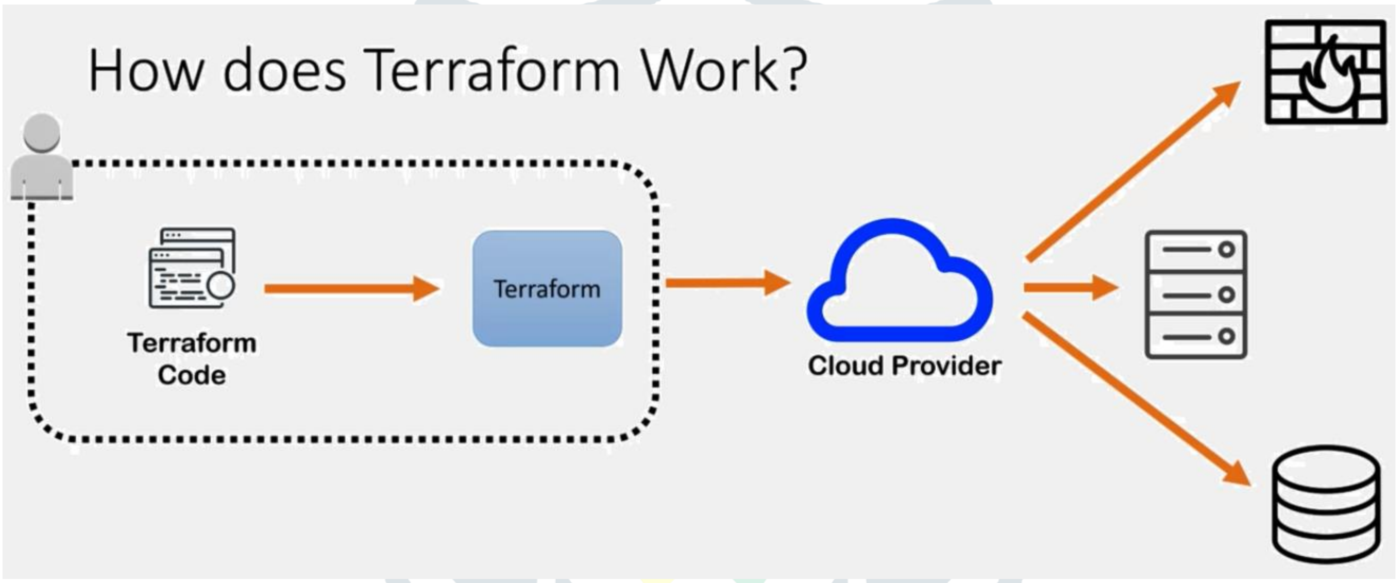

Figure 4 – Deployment of Security Infrastructure using Terraform

DevOps was in a way a result of this continual fast expansion of IT but isn't as new as it might seem at first glance. The term DevOps is split-able into two separate words "Development" and "Operations". This particular methodology is a convergence of several principles, as the Lean movement, Agile principles, Theory of constraints, change management, quality assurance (QA), product management and others, two of the most notable being Lean and Agile. Agile arose around the end of the second millennium and is in a way successor to the waterfall model, with its incremental releases, small changes, and highly self-motivated teams. There is an emphasis on delivering software in a short time scale, coping with requirements changes, customer satisfaction, and continuous quality delivery.

Another core principle of DevOps lays in lean manufacturing. Lead time is of great importance when practicing lean, as it is considered to be an indicator of the quality and satisfaction of the customer. The short delivery date is to be achieved mostly by working in small batches. Holding onto those principles should result in value creation, thus customers' satisfaction. Lean tries to remove everything which is of no value to the customer, simplifying procedures as much as possible.

The continuous delivery movement, which also participates greatly in DevOps and Agile respectively, is responsible for frequent and quick delivery of software and process automation in this aspect, which is what both of them have in common. DevOps itself has a broader scope, as it is more of a cultural change, bringing together different teams like QA, operations, development, and others. It is apparent that DevOps is not simple in any way, it is a complex software engineering culture and practice.

IV. SELECTION OF SECURITY AUTOMATION TOOLS

The selection process of a configuration management tool to be used in the real-world enterprise is not trivial, as many characteristics have to be taken into account. There are a lot of comparisons available on the internet, but data validity is low.

Usually, reviewers only compare the essential general features, which is insufficient, without any real-world data or hands-on experience. Selection of right tools is critical as the initial implementation of DevOps, or individually, IaC is time-consuming and often costly. The company has to hire a DevOps professional or has to dedicate enough human resources to do the job. The realization, that company has chosen the wrong tool in the middle of the implementation process, might result in management dissatisfaction, lost time, delayed projects. "Fig. 5" shows how Terraform can be used with Ansible for security and infrastructure rapid deployment respectively. Hence in this research, I have evaluated multiple tools for security automation and found Terraform is the best one of them.

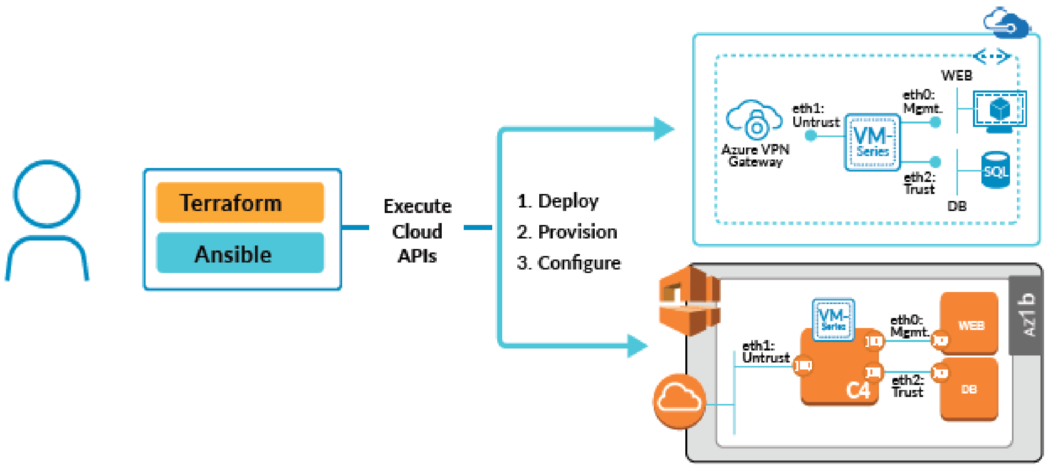

Figure 5 – Terraform Code Deployment in AWS and Azure Cloud

4.1 Terraform

Terraform is an automation tool for building, modifying, and versioning infrastructure safely, easily and efficiently. Using Terraform we can manage existing and almost all cloud service providers as well as custom in-house solutions. Terraform has configuration files which describe the components needed to run a single application or multi-tier application or your entire datacenter. There are two major states in Terraform execution, first one is the desired state and another is the current state, Terraform generates an execution plan describing what action it will take to reach the desired state, and then executes the plan to build the demanded infrastructure. As the configuration changes, during Terraform plan execution, Terraform is able to determine what changed and create incremental execution plans which can be applied using command Terraform apply. Terraform can manage all level of infrastructure which includes low-level components such as security group, network access control list, IAM, WAF, CloudWatch, CloudTrail, Guard Duty, AWS Shield, and networking, as well as high-level components such as DNS entries, SaaS features, etc.

4.2 Key Feature of Terraform

Infrastructure as Code: Infrastructure can be described using a high-level configuration syntax. This allows a blueprint of your complete datacenter to be versioned and treated as you would any other code. Additionally, infrastructure can be shared and re-used.

Execution Plans: After building code in Terraform we can proceed for "planning" step where it generates an execution plan. The execution plan displays what Terraform will do when you call apply this code. These planning steps avoid doing any big mistake when Terraform manipulates infrastructure rapidly.

Resource Graph: Terraform creates a graph of all your resources, and parallelizes the creation and modification of any non-dependent resources. By using resource graph, Terraform able to build infrastructure efficiently as possible.

Change Automation: Complex change sets can be applied to your infrastructure with the least human interaction.

4.3 Why Terraform for Cloud Security Automation

Terraform provides a flexible generalization of resources and providers. This model allows for representing everything from physical hardware, virtual machines, security groups, network ACL, containers, notification service, email and DNS providers. Because of this flexibility and easy to use, Terraform can be used to automate security features in the cloud. In the cloud, we receive logs from a various component like VPC flow logs and CloudTrail logs, CloudWatch logs and AWS Guard Duty logs and Shield logs. Appropriate enabling all these logging manually requires a lot of human involvement and time consuming, but using Terraform we can enable all security features in a single deployment and consequently can receive an alert in case of credentials compromisation, risky configuration, crypto-jacking, etc.

4.3.1 Multi-Tier Application

A very common architecture is N-tier. The most common 2-tier architecture consists of web servers that use a database tier. One more tier may get added for API servers, caching servers, routing meshes, etc. This pattern allows the tiers can be scaled independently and provide a separation of concerns.

Terraform is a modern and proficient tool for building and managing these infrastructures. Each tier can be described as a collection of resources, and the dependencies between each tier are handled automatically; Terraform will ensure the database tier is available before the web servers are started and that the load balancers are aware of the web nodes. Each tier can then be scaled easily using Terraform by modifying a single count configuration value. Hence using Terraform the creation and provisioning of a resource or infrastructure is codified and automated, it automatically delivers elastically scaling with load becomes trivial.

4.3.2 Self-Service Clusters

In a big organization, it becomes very confusing for a centralized operations team to manage a huge and developing infrastructure. On the other hand, it becomes more interesting to make "self-serve" infrastructure and hence allow different product teams to manage their own infrastructure and resource using Terraform provided by the central operations team.
Using Terraform, the knowledge of how to build and scale a service can be codified in a configuration. Terraform standard template which consists of standard security policies can be shared within an organization and hence allowing customer teams to use the configuration as a template and use Terraform as a tool to manage their services.

4.4 Securing Two-Tier Application with Terraform

Given are the basic component of Terraform which can be used while writing automation script for N-tier of architecture.

Providers: A Terraform plugin that understands how to work with a specific service provider. Terraform has multiple providers. We have to specify the provider details for which we want to launch infrastructure. While specifying the provider, we also have to add the tokens which will be used for authentication. For example, AWS access key and secret key in case of AWS provider and Digital Ocean token in case of Digital Ocean.

Resource: Resource is the reference to the individual services which the provider has to offer. For example, resource aws_intance, resource aws_alb, resource iam_user, and resource digitalocean_droplet.

State File: Terraform stores the state of Infrastructure that is being created from the TF files. This state allows Terraform to map real-world resource to your existing configuration.

Attributes: Terraform has the capability to output the property of a resource in the form of an attribute. An attribute not only is used for the user but it can also act as input to the other resource being created via Terraform.

Interpolation: If something is unknown that can be valued using an Interpolation process. It is directly associated and works with the help of attributes. It is directly associated and works with the help of attributes.

Variable: We can have a central source from which we can import the values from**.**

Provisioners: provisioners is code or programs that are used to bootstrap upon creation. Provisioners normally is used to executing scripts on a remote or local machine as part of infrastructure creation or destruction.

4.4.1 Deployment of Web Server and its Security

Here I am explaining the two-tier architecture configuration code. The first layer is a web layer and the second layer is the application layer. In this section, we are using AWS as a provider. Below given code will deploy one web server and its private IP get attached with public IP also known as static elastic IP and for security purpose, we are using AWS security group which allows web traffic from anyone (i.e., 0.0.0.0/0) and SSH access for server management only from one public IP address (1.1.1.1). The access key and secret key are used to authentic against AWS account. There is no need to specify account details as it can be identified by given access key and secret key. There is need to specify instance type which determines its computation capacity and also it is mandatory to define AMI ID which specifies which kind of OS with this server boot, it may Firewall, VPN concentrator, Linux server or Window server.

```
provider "aws" {
  region = "us-west-1"
  access_key = "YOUR-ACCESS-KEY"
  secret_key = "YOUR-SECRET-KEY"
}
resource "aws_instance" "myweb" {
   ami = "ami-bf5540df"
   instance_type = "t2.micro"
   security_groups = ["${aws_security_group.mysg.name}"]
   tags {
      Name = "web-server"
  }
}
resource "aws_security_group" "mysg" {
  name = "web-server-sg"
  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["1.1.1.1/32"]
  }
resource "aws_eip" "myeip" {
  instance = "${aws_instance.myweb.id}"
}
```

4.4.2 Deployment of Application Server and its Security

Below built code will deploy one application server and its private IP get attached with public IP also known as static or elastic IP and for security purpose we are using AWS security group which allows HTTP & https traffic only from the web server's security group and SSH access for server management only from the client's public IP address (1.1.1.1).

```
              resource "aws_instance" "myapp" {
   ami = "ami-bf5540df"
   instance_type = "t2.micro"
   security_groups = ["${aws_security_group.myappsg.name}"]
   tags {
      Name = "app-server"
  }
}
resource "aws_security_group" "myappsg" {
  name = "app-server-sg"
  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["${aws_eip.myeip.public_ip}/32"]
  }
  ingress {
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["${aws_eip.myeip.public_ip}/32"]
  }
  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["1.1.1.1/32"]
  }
 }
resource "aws_eip" "myappeip" {
  instance = "${aws_instance.myapp.id}"
}
```

V. RESULTS OF EVALUATION

I have implemented this research concept for one of the famous cloud service provider which is AWS (Amazon Web Service). However, this tool is open source and can be used to automate security for almost all cloud providers. After getting motivated from real-world problem, I have found that these days in an organization, DevOps engineers themselves are taking care of public cloud security and organization don't have dedicated resources to manage cloud security and there is no doubt that they are expert in rapid infrastructure deployment but they are not much proficient in best cloud security implementation and this is because of their non-security background.

After automating the logs from each service using terraform during part of rapid infrastructure deployment, I was able to receive alert for a various malicious activity like login via root account, login from blacklisted IP address, intrusion activity, allowing SSH service for all hosts, any major change in Infra including many suspicious activities.

VI. ACKNOWLEDGMENT

This research conducted to meet the current challenges that are being faced to improve the security in cloud providers. We have used Terraform for security automation which is an open source tool. We have implemented this research concept in AWS cloud and evaluate its output and it was very positive and problem-solving. I would like to thank Dr. Neetu Sharma who supported me during this journey.

REFERENCES

[1] Prashant Priyam. 2018. Cloud Security Automation: Get to Grips with Automating Your Cloud Security on AWS and OpenStack.
[2] Raghuram Yeluri and Enrique Castro-Leon. 2014. Building the Infrastructure for Cloud Security: A Solutions View (Expert's Voice in Internet Security)
[3] Brian T. O'Hara and Ben Malisow. CCSP (ISC)2 Certified Cloud Security Professional Official Study Guide
[4] Yevgeniy Brikman. 2017. Terraform: Up and Running- Writing Infrastructure as Code