International Journal of Research in Engineering and Science (IJRES)
ISSN (Online): 2320-9364, ISSN (Print): 2320-9356
www.ijres.org Volume 12 Issue 8 ǁ August 2024 ǁ PP. 09-14

Streamlining Cloud Deployments: The Power of Terraform and Google Deployment Manager in Error Reduction

Rajendraprasad Chittimalla
MS in Information System Security
Software Engineer – Team Lead

Abstract: *Cloud deployment serves as a solution for organizations willing to scale the project with their current resources. The organizational data and brand value remain at risk while enjoying cloud services and therefore it should be less exposed to errors. The manual deployment of cloud operations has more potential for error occurrence. The tools that are less prone to errors are provided as a solution which majorly includes Terraform and Google Cloud Deployment Manager. The key features of these two tools help reduce errors as well as provide further advantages*
Keywords: *cloud deployment, terraform, error reduction, google deployment manager, terraform, Infrastructure as Code*

Date of Submission: 24-07-2024 Date of acceptance: 07-08-2024

I. Introduction

Cloud Deployment saves resources for organizations and provides more scalability in their business by providing modern applications, storage space, and access to the servers. More organizations in modern times are choosing cloud services due to efficient results, reliability, and strong infrastructure provided by these remote servers [1]. However, there are a few difficulties in their configuration along the way. The manual configurations of cloud deployment can expose organizations to business risks. There are high chances of error occurrence when adopting the manual implementation of cloud services [2].

Considering these issues, organizations nowadays rather prefer to use effective deployment tools like Terraform and Google Deployment Managers. These tools are designed as a solution for industry to reduce deployment errors and automate the process to reduce organizational resources [3]. The overall operations of cloud deployment become effective with the use of these two tools. Figure 1 shows the overall interaction methods with the cloud.

This research writing first explores the problems or complications that lie with the use of manual cloud deployments. It then gets extended to providing Terraform and Google Deployment Manager as automated tools that work as a solution to reducing errors in implementation. These tools are explained in detail with their key features because of which they are considered to be effective in the industry. After that the benefits of these automation tools are discussed along with the future potential they have. Furthermore, the research impact helps to understand the importance of this writing and its impact factor on the industry.

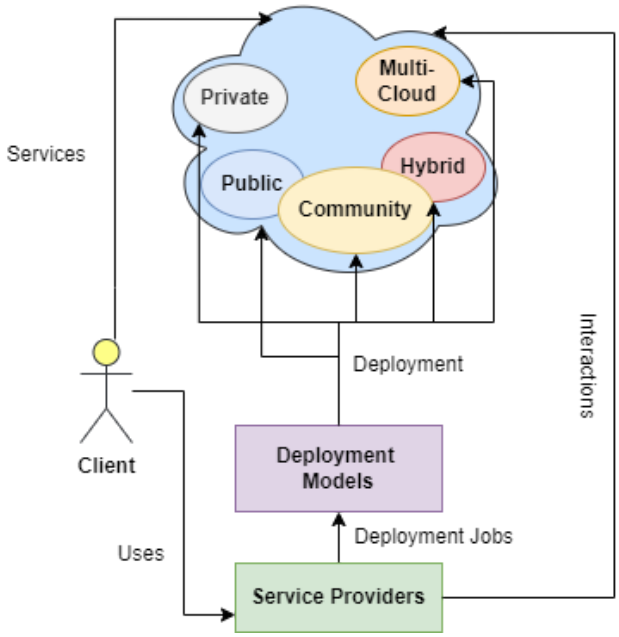

Figure 1: *Overview of Cloud Deployment*

II. Literature Review

Cloud Computing has been a center of attention for the past decade considering the advantages it provides to organizations and individuals. It gives remote access to avail the storage and services from remote servers. With the required of just internet connectivity, one can avail the remote access in an efficient manner reducing the extra cost. However, cloud computing is associated with different threats and risks to organizations. These need to be addressed appropriately for proper cloud deployment for a better experience [4].

The field of cloud computing and deployment of storage and services on remote servers is accompanied by potential problems like repeated error occurrence. Errors are one of the biggest challenges for industry and methods are being introduced to tackle it. Tools like Terraform and Google Cloud Deployment Manager seem to have the potential to reduce the error risks in cloud deployment [5]. The efficiency of the system however varies with the effective use of these tools. The appropriate job needs to be assigned to a suitable tool to get the desired results.

III. Problem Statement

There are inadequate resources available in organizations to carry on the manual tasks in the management of complicated cloud environments. The bigger problem is that manual deployments are highly prone to error and can lead to many vulnerabilities. The use of cloud deployment has increased over the decade and there arises a need for a reliable and efficient deployment process. Organizations are now interested in tools that are efficient enough to exponentially reduce errors compared to manual work, provide them secure environment, and save them from the wastage of valuable resources

IV. Complications in Manual Deployments

Cloud Deployment with manual implementation methods is highly exposed to errors because of the complex and large cloud environments [6]. The following issues may arise due to restricted options of manual deployment:

- Misconfiguration
- Unpredictable Results
- Security Problems
- Cost Problems
- Loss of Human Resources

Therefore, automated tools are required to provide automated systems, effectively proceed the tasks, save efforts in repeated tasks, and faster deployments.

V. Terraform

The tool is provided by HashiCorp as an Infrastructure as Code (IaC) to automate infrastructure tasks in cloud deployment. Terraform supports a wide range of cloud providers, including Google Cloud Platform.
The following key steps are followed when deploying infrastructure with this tool:

1. **Scope**: Infrastructure is identified first.
2. **Author**: Configurations are created next.
3. **Initialize**: Installation of required plugins comes after that.
4. **Plan**: The staging can be done by first reviewing the changes applied by Terraform.
5. **Apply**: Final changes are deployed.

5.1. Key Features

This tool is equipped with the following key features [7],

Declarative Configuration

The infrastructure as code is defined using HashiCorp Configuration Language (HCL). The required end-state is declared for the infrastructure. Therefore, it doesn't require a step-wise procedure to perform the desired tasks.

Definite Language

The Human-Readable Configuration Language is used in this tool to define the technical infrastructure. It helps the users to adopt the desired state without the actual steps to get the state of the infrastructure.

Modules

The less redundant infrastructure is provided by this tool as there are reusable components in it and all come with effective configuration. The reduced redundancy of different chunks helps mitigate errors in it. The shareable components are available for common configurations.

State Management

It tracks resource state and makes incremental changes. The log files are maintained as a record of states and capture all the current states of the infrastructure. The consistency is ensured with the use of this effective trail of log which is stored against useful information like timestamp.

5.2. Technical Details

1. Creating Terraform Configurations

Configuration Files: Define infrastructure in .tf files using HCL.

Example Configuration:

hcl

```
provider "google" {
  credentials = file("<YOUR-CREDENTIALS-FILE>")
  project     = "<YOUR-PROJECT-ID>"
  region      = "us-central1"
}

resource "google_compute_instance" "default" {
  name         = "my-instance"
  machine_type = "n1-standard-1"
  zone         = "us-central1-a"

  boot_disk {
    initialize_params {
      image = "debian-cloud/debian-9-stretch-v20210721"
    }
  }

  network_interface {
    network = "default"
  }
}
```

2. Initializing Terraform

Initialize Terraform: Run terraform init to download provider plugins and initialize the working directory.

sh

```
terraform init
```

3. Applying Configurations

Plan: Review changes before applying.

sh

terraform plan
Apply: Create or update resources based on the configuration.
sh
terraform apply
4. Managing State
State File: Terraform keeps track of the state of your infrastructure in a state file. This file can be stored locally or remotely (e.g., in Google Cloud Storage for collaboration).
5. Destroying Resources
Remove Resources: Delete all resources defined in your configuration.
sh
terraform destroy
6. Viewing State and Logs
State File: View the state file for current infrastructure status.
Logs: Check Terraform logs for detailed operation results.

1. Google Deployment Manager

The Google Cloud Deployment Manager (GCDM) is another tool under Infrastructure as Code (IaC). It allows you to create, configure, and manage Google Cloud resources using YAML or JSON configuration files.

1.1. Key Features

The following key features are provided in it that help reduce the errors in the deployment process [8].

Resource Management

The management of the lifecycle of GCP resources such as Compute Engine Instances and Cloud Storage Buckets with versioning and updates. So there remains minimal unused resources and the workload is deployed all over.

Templating

It uses Jinja2 templates to dynamically generate configurations based on parameters.

GCP Services Integration

The management of resources without extra effort has become possible with the integration of GCP services. The cloud storage and other resources can be managed without causing errors due to an enriched Google Cloud Platform that comes with GCDM.

Declarative Configurations

The required state of infrastructure is provided in this Deployment Manager with the appropriate use of Python with its enriched features. It lets you define infrastructure in configuration files (YAML or JSON) and allows you to deploy it as a single unit. Strong configuration methods are employed to reduce the risks of exposing the system to attacks or internal errors.

1.2. Technical Details:

1. Creating Deployment Configurations

- **Configuration Files:** Write configuration files in YAML or JSON to specify resources.
- **Templates:** Optionally, use Jinja2 or Python templates for dynamic content.

Example YAML Configuration:

```
yaml
resources:
- name: my-instance
  type: compute.v1.instance
  properties:
    zone: us-central1-a
    machineType: zones/us-central1-                a/machineTypes/n1-standard-1
    disks:
    - deviceName: my-instance
      type: PERSISTENT
      boot: true
      autoDelete: true
      initializeParams:
        sourceImage: projects/debian-cloud/global/images/debian-9-stretch-v20210721
    networkInterfaces:
    - network: global/networks/default
```

2. Deploying Resources

Create Deployment: Use the Google Cloud Console, gcloud command-line tool, or REST API to create a deployment.
Using gcloud CLI:

Sh
gcloud deployment-manager deployments create my-deployment --config config.yaml

3. Manage Deployments

Update: Modify your configuration files and update the deployment.
sh
gcloud deployment-manager deployments update my-deployment --config updated-config.yaml
Delete: Remove the deployment and associated resources.
sh
gcloud deployment-manager deployments delete my-deployment

4. Viewing Deployment Status

Google Cloud Console: Navigate to "Deployment Manager" to view status and manage deployments.
CLI: Use gcloud deployment-manager deployments describe my-deployment to get detailed information.

2. Benefits of Cloud Deployment Tools

Both the Terraform and Google Deployment Manager tools are beneficial in cloud deployments to reduce potential errors. These tools open the door to multiple possibilities. Here is how,

2.1. Consistent Deployment

Both of these tools tend to provide consistent and repeated deployment. Thus, the problem of unpredictable elements will be eradicated unlike manual deployment

2.2. Cost Reduction

The optimization increases with the use of these tools rather than just wasting human efforts. The configuration errors are reduced with the use of these tools and therefore staff doesn't have to work on these.

2.3. Efficient Results

The results of automation with the help of these tools are more efficient and are less likely to be impacted by human errors. This also helps in achieving the results fast by using fewer resources for a short time.

2.4. Compliance Control

The audit trailing can be done due to the use of version controls and therefore security issues are less likely to occur.

2.5. Disaster Recovery

The Terraform and Google Deployment Manager are more effective in recovery plans as humans are not trained enough to handle emergencies. Thus rapid recovery of the infrastructure is possible while facing failure.

2.6. Easy Cloud Migrations

The migrations are also possible with the use of these tools and organizations can automate the migration process. This results in consistent and error-free deployments while transferring to the new cloud environment.

Research Impact

The findings of the presented work provide the impact of cloud deployment tools especially Terraform and Google Deployment Manager to not only understand their importance but convince the organizations to replace the manual deployment procedures with these highly competitive tools. It provides insights into efficient operational methods, secure implementations, reduced cost, innovative methods, and appropriate skill development in understanding these tools.

Future Developments

Further improvements are predicted to be inculcated in the latest versions of tools like Terraform and Google Deployment Manager for a more enhanced experience. There are high chances of the availability of advanced security methods, easy interfaces, real-time collaboration through virtual experiences, and improved support with AI-based Chatbots. This industry is currently aiming to provide a better user experience and therefore there will be more enhanced automation which will come with useful integration mechanisms.

VI. Conclusion

To sum up, cloud deployments provide different advantages to organizations and startups but also accompany a few problems. There is a very high chance of errors when the manual deployment method is adopted due to the complexity of this process. The sensitive data and tasks are also at risk which most organizations can't afford which is why automated cloud deployment tools are required.

Tools like Terraform and Google Cloud Deployment Manager help to eradicate the occurrence of repeated errors and provide a secure way to enjoy cloud services. These tools are equipped with the required key features to not only reduce errors but also save resources for organizations. The research document therefore explores the key features that are provided by these tools along with the benefits they collectively present if utilized appropriately.

References

[1]. A. Sunyaev, "Cloud Computing," in Internet Computing, Springer, 13 Feb 2020, pp. 195-236.
[2]. M. M. A. N. M. Sadeeq, S. R. M. Zeebaree, D. M. Ahmed, A. S. Sami and R. R. Zebari, "IoT and Cloud Computing Issues, Challenges and Opportunities: A Review," Qubahan Academic Journal, vol. 1, no. 2, pp. 1-7, 15 Mar 2015.
[3]. M. Valkeinen, "Cloud Infrastructure Tools For Cloud Applications : Infrastructure management of multiple cloud platforms," Tampere University, 2022.
[4]. H. Tabrizchi and M. K. Rafsanjani, "A survey on security challenges in cloud computing: issues, threats, and solutions," The Journal of Super Computing, vol. 76, pp. 9493-9532, 28 Feb 2020.
[5]. L. Nguyen, "Comparing Terraform and Cloud Deployment Manager," Medium, 17 Jan 2022.
[6]. A. Tchernykh, U. Schwiegelsohn, E.-g. Talbi and M. Babenko, "Towards understanding uncertainty in cloud computing with risks of confidentiality, integrity, and availability," Journal of Computer Science, vol. 36, Sep 2019.
[7]. B. Campbell, "Terraform In-Depth," in The Definitive Guide to AWS Infrastructure Automation, Berkeley, CA, Apress, 2020, p. 123–203.
[8]. W. Bessovi, "Hands on GCP Deployment Manager," LINKBYNET, Medium, 5 Mar 2021.