2020 International Conference on Computational Performance Evaluation (ComPE),
North-Eastern Hill University, Shillong, Meghalaya, India. July 2–4, 2020

Infrastructure as a Code (IaC) to Software Defined Infrastructure using Azure Resource Manager (ARM)

Jagdish Chandra Patni
School of Computer Science, UPES,
Dehradun, India
patnijack@gmail.com

Souradeep Banerjee
School of Computer Science, UPES,
Dehradun, India
500052778@stu.upes.ac.in

Devyanshi Tiwari
School of Computer Science, UPES,
Dehradun, India
500052376@stu.upes.ac.in

***Abstract*—As the scale of cloud infrastructure is augmenting, so is the number of instances that needs to be provisioned. As a result, there is a requirement for automating the process of managing infrastructure instances to facilitate development, staging, and production environments. The purpose of this paper is to understand 'Infrastructure as a code' (IaC) in relation to Software Defined Infrastructure. Using Azure Resource Manager (ARM) Templates we will demonstrate the benefit and importance of IaC by deploying a web application on Microsoft Azure using only code. The use of ARM templates was found to increase the agility of the deployment and management process. We will also demonstrate IaC using Terraform and Pulumi. The paper also discusses the correlation of IaC with the concept of code reusability and repeatability.**

***Index Terms*—Infrastructure as a Code; Software Defined Infrastructure; Azure Resource Manager; Terraform, Pulumi**

I. Introduction

Over the years the IT infrastructure was provisioned manually which meant that people have to stack the servers physically; also, all the configuration process was done arduously [1]. This technique was expensive, extensive and posed a threat of lack of agility thus as the size of infrastructure grew there was a dire need for a new method to manage infrastructure. Infrastructure as Code (IaC) refers to the management of infrastructure components like networks, virtual machines, load balancer etc. in a model which uses automation rather than a manual process in a cloud-based environment. IaC is a critical DevOps technique and is used in concurrence with uninterrupted delivery. Software Defined Architecture (SDI) includes infrastructure control, management provisioning, configuration and other architectural operations not tied to any hardware. SDI allows the infrastructure to operate as self-aware, self-healing, self-scaling, self-optimizing IT environment to enable truly agile business processes [2]- [3]. Cloud gurus consider IaC as the primordial pillar to implement DevOps, which has led to major organizations like GitHub, Facebook, Google and Netflix to adopt IaC. Fig 1 shows the critical components of a DevOps pipeline.

IaC allows to spin up an entire infrastructure architecture by running a script thus all of the tasks can be done quickly and easily. Also, IaC standardizes the setup of infrastructure so there is a reduced possibility of any errors or deviations. Since code can be version controlled, IaC allows every change to server configurations to be documented, logged, and tracked. Infrastructure as Code allows companies to use Continuous Integration and Continuous Deployment capability while reducing the chances of human errors after the development period. Also because of the increased productivity, companies can save money on hiring costs. Best practices of IaC require managing infrastructure by source control, testing the infrastructure continuously, avoiding any written documentation and to increase collaboration with respect to configuration and provisioning.

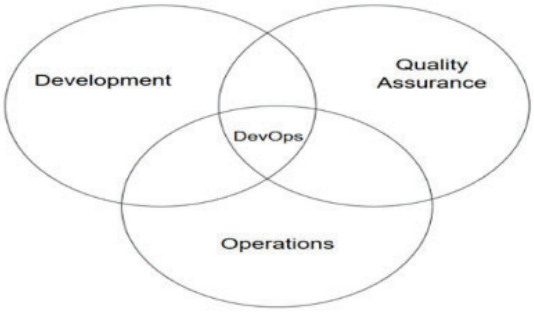

Figure 1: DevOPs Outline

This paper uses ARM templates, Pulumi and Terraform to explain IaC. Microsoft Azure is a cloud service being provided by Microsoft for developing, testing and managing applications and services through Microsoft data centers. Just like other cloud service providers Azure also follows a pay-as-you-go model that charges the user based on their usage. One of the key components in Microsoft Azure is the deployment of infrastructure using ARM templates or Azure Resource Manager Templates [4] [5].

ARM Template is a way of deploying infrastructure as a code to Azure. They define objects, their types, names and properties in a JSON file which can be understood

978-1-7281-6644-5/20/$31.00 © 2020 IEEE

by ARM API. It is important to note that the ARM API deploys resources to Azure, but doesn't deploy code onto those resources. For example, one can use ARM to deploy a virtual machine with SQL Server already installed but they can't use ARM to deploy a database onto it.

II. Related Work

In the book "Infrastructure as Code" [6] Kief Morris explains that IaC makes use of consistent and repetitive approach for provisioning of hardware and services such as computing and storage. There are many advantages of Infrastructure as Code. As discussed by Matej Artač et al. [7] Infrastructure as Code is the key practice to speed up DevOps based lifecycles. However, the authors did not describe how various software can be applied to implement IaC. n their white paper, [8] HashiCorp explains that Terraform automates through an infrastructure as code approach to provisioning cloud infrastructure and services. It allows users to specify their required topology of infrastructure by using version control. Terraform offers a powerful alternative to that traditional models by utilizing infrastructure as code approach for provisioning and management of the white paper explains how an organization can make use Terraform to ensure multi-cloud compliance. Like Terraform one very powerful IaC tool that has evolved is Pulumi. In [9] the authors explain that the rapidity with which the cloud is evolving demands a different approach to cloud development and Pulumi as a tool for IaC can be very beneficial as it provides a multi cloud and multi technology tool that deals with real languages and provides new workflows to deliver IaC.

III. Implementation

In order to implement Infrastructure as Code we will be following a particular software development life cycle model in the whole process. In our case it is going to be the agile model since we can iterate over the needs of the customer according the model designed and them enhance the future functionalities.

As part of this research we are going to implement the 3-tier architecture or the WEB-API-DATABASE Architecture as shown is fig. 2 using different techniques and deploy the infrastructure needed using code. At the base of the model lies the database tier where data gets stored in a database server and various CRUD operations are being performed here. Above the database tier lies the middle tier which contains the application logic and does all the communication between the user and the database. Lastly on the top lies the web tier which is the web client that is used to interact with the user. User puts their request through the web tier, which gets processed by the middle tier and accordingly operations are being performed on the data base tier.

The Web Application that we are going to develop is supposed to be based on Java. We are going to use servlets and jsp pages to develop the system. We are going to have a single java file which will act as the controller and do all the transport of data. This controller will act as the middleware

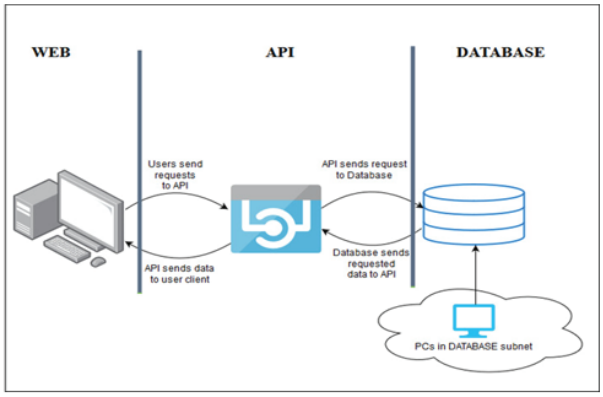

Figure 2: Web-API-Database Architecture

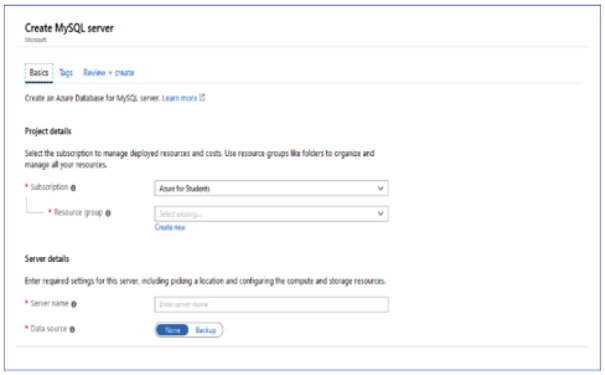

Figure 3: Azure database for MySQL servers

for communication between the user and the database. The database we will be using will be a relational one and it will be running on a MySQL server. We will be using the JDBC driver in-order to connect the web application to the MySQL server database so as to deploy the system offline.

After we have implemented the system on-premises, we are going to outsource it to Microsoft Azure. We are going to create an Azure application Service plan as shown in fig. 3, in Azure using the ARM templates which will deploy this web app using the proper resources.

After this we go to Azure database for MySQL servers (fig. 3). We are going to create a table which we store the username, email, password, last login date and time of the user, etc. The username and email will be the primary keys of the database and will be used to uniquely identify each user. This database is to be deployed on Azure using ARM template.

The three architecture is a basic model for developing a web database application and communication is done using this 3-level application logic. The database tier is where data gets stored in a database server and is known as fully secure. Various CRUD operations are being performed here. Next is the API layer inside the API subnet which provides a communication between the web server and the data base server. Finally, there is a web server which accepts all the internet requests and send requests to the API server. The basic system design is being visualized in fig. 4. In this ARM

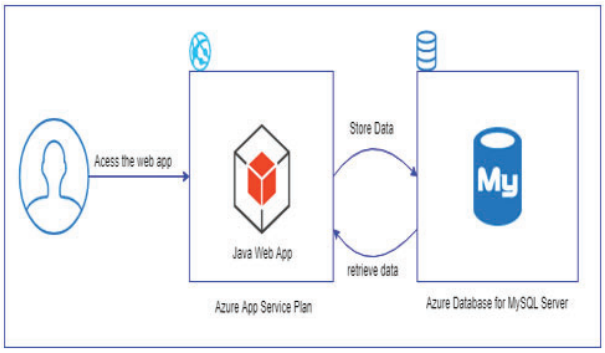

Figure 4: Basic Design of the System

Template two virtual networks are created with three subnets each and having three network security groups associated with the subnets. Also Peering is enabled between the two virtual networks. In case of Disaster Recovery, we are modifying the database NSG so that the database subnet of virtual network 1 can send its replica to database subnet of virtual network 2 through a specific port.

IV. Proposed Technique

The three-tier web architecture can be implemented using three techniques, each of which uses code to deploy the infrastructure needed and thus implement Infrastructure as Code.

A. Using Azure Resource Manager Templates

ARM templates are being used in order to deploy infrastructure with the help of code. ARM templates are being designed in JSON language which can be understood by the ARM api. Prior to the release of ARM, the Azure Service Management api used to handle all kinds of deployments which were then termed as classic. This was being replaced by the ARM api.

The main benefits of ARM api is that we can deploy several units of infrastructures as a single unit. In ARM template user defines the resource and type of resource to deploy and the ARM api either creates a new object or modifies the existing one to get the required deployment.

1) Modes of Deployment:

a) Incremental: This mode is used to add resources to an existing resource group to which the template is being deployed. Benefit is that we don't lose any resources pre-existing the resource group but on the other hand we need to manually remove the resources that we don't need.
b) Complete: This mode deletes any object i.e. resource that is not mentioned in the template from the resource group.

2) Execution of ARM templates: REST API plays an important role in the execution of ARM templates. There is this one set of REST API called "Resource Management" where we send an ARM template. The REST API does the following:

a) a) Parse the JSON file.
b) Filling the parameters that are being passed to the JSON template file.
c) Execution of the functions inside the ARM template.
d) Calling the REST API of the concerned resource that needs to be created.

B. Using Terraform

Terraform is a tool, developed by HashiCorp, that is being used to deploy infrastructure safely and easily via code. It makes the use of configuration files in order to run applications on premises. It makes use of the concept of 'desired state'. It compares the existing state with the desired state and then does the required deployments.

It mainly does the implementation of what is known as Infrastructure as a Code using high level configuration syntax. It introduces the idea of re-usability and repeatability.

1) Execution of Plan: Terraform uses the concept of "Plan" where it shows us the execution plan of the current deployment before being it is actually deployed. This helps us to notice if the deployment is actually according to our needs before terraform starts to manipulate the infrastructure.

2) Configuration: Before using terraform, we need to configure the same with a service principal, A service principal is a type of security principal that represents an application. Service principal provides an identity for our application, allowing the application to do the necessary deployments.

C. Using Pulumi

Pulumi is an open source platform for building and deploying infrastructure in different languages such as JavaScript, Python, Go or typescript and on multiple cloud platforms like Microsoft Azure, Amazon Web Services, Google Cloud Platform and Kubernetes.

D. Benefits

Benefits of using these languages gives us the advantage of:

1) Familiarity: We do not need to learn any kind of separate YAML templating language or any other Domain-Specific Language (DSL) to write code in Pulumi.

2) Abstraction: Pulumi provides a kind of abstraction layer, like we can create files for deploying different kinds of resources and then use them to create something big. We can hide the implementation details of the smaller resources.

3) Sharing and Reuse: We can use existing packages in the community to do the deployments we require or share our code with others so that they can use the same. This enables code re-usability and repeatability.

4) Full Control: We can apply the full control of the programming language in Pulumi. Loops, conditions can be used in Pulumi to deploy infrastructure.

V. Results

The Web Application that we have implemented in this paper has a landing page that provides the user with Register and Login Options.

If the User clicks on the register button the user gets the form to enter in the details about the users along with the photo of the student.

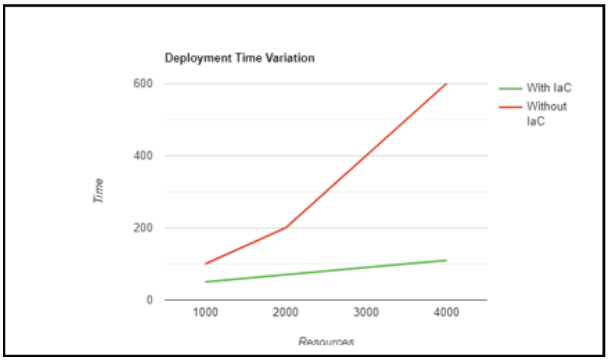

Figure 5: Graphical Representation for Speed Variation

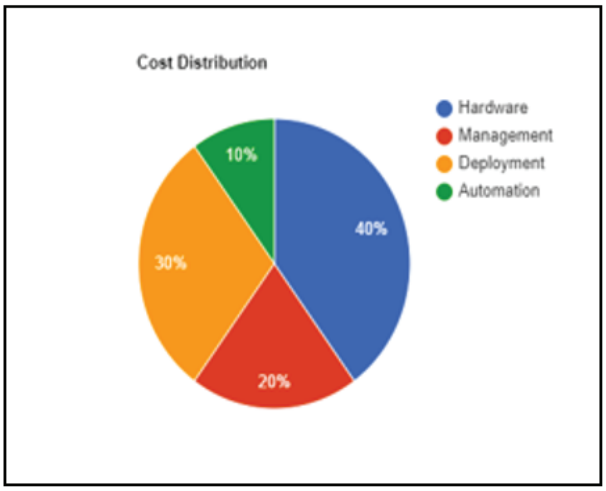

Figure 6: Cost Distribution

If the User clicks on the login page, he gets the option to login with registered email id and password as shown in Using Infrastructure as Code we are able to overcome the different drawbacks associated with manual approach to infrastructure deployment and management. Adopting Infrastructure as Code as a solution gives us the following benefits:

A. Speed

Infrastructure as Code helps us to spin up complex infrastructure in minutes using a simple script whereas manual intervention would have taken days to achieve the same. Fig.5 shows us the variation in speed of deployment of resources with time.

B. Cost

Infrastructure as Code helps to dramatically reduce costs for resource deployment since we no longer need to spend much money on hardware, people to operate it and many other attributes. The only thing we need to focus on is automation. Fig. 6 give us an overview of cost management for deploying an on-premise solution.

VI. Conclusion

Hence using Infrastructure as a code (Iac) for the management of our cloud infrastructure, provides us with the benefits of minimal user effort required to deploy and manage our system. Following advantages have been observed:

1. Using ARM templates, we can not only deploy but also monitor our Azure resources.
2. It provides us with code reusability as the ARM templates can be executed. Same template can be used to deploy multiple times for a variety of organization's infrastructure requirements.
3. We can define dependencies to ensure that correct order was used to deploy our templates.
4. If very large number of resources need to be deployed on our Azure platform, instead of manually configuring and deploying each of these resources which would require a lot of manual effort as well would be a very time-consuming process we can use an ARM template. Hence by using an ARM template we can provides a single JSON file which automates the entire process and creates the required number of resources with required configurations with just a click of a button.

[**?**] [**?**].

References

[1] A. Rahman, R. Mahdavi-Hezaveh, and L. Williams, "A systematic mapping study of infrastructure as code research," *Information and Software Technology*, vol. 108, pp. 65–77, 2019.

[2] A. Sharma, A. R. Ahmad, D. Singh, and J. C. Patni, "Cloudbox—a virtual machine manager for kvm based virtual machines," in *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*. IEEE, 2016, pp. 588–594.

[3] A. Gupta, H. Bhadauria, A. Singh, and J. C. Patni, "A theoretical comparison of job scheduling algorithms in cloud computing environment," in *2015 1st International Conference on Next Generation Computing Technologies (NGCT)*. IEEE, 2015, pp. 16–20.

[4] J. C. Patni, "Hierarchical load balancing model by optimal resource utilization," *International Journal of Business Analytics (IJBAN)*, vol. 6, no. 3, pp. 29–42, 2019.

[5] J. C. Patni and M. S. Aswal, "Distributed approach of load balancing in dynamic grid computing environment," *International Journal of Communication Networks and Distributed Systems*, vol. 19, no. 1, pp. 1–18, 2017.

[6] K. Morris, *Infrastructure as code: managing servers in the cloud.* " O'Reilly Media, Inc.", 2016.

[7] M. Artac, T. Borovssak, E. Di Nitto, M. Guerriero, and D. A. Tamburri, "Devops: introducing infrastructure-as-code," in *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. IEEE, 2017, pp. 497–498.

[8] "Unlocking the cloud operating model," https://www.hashicorp.com/cloud-operating-model, (Accessed on 08/13/2020).

[9] A. M. P. P. J. C. Sharma, Abhishek; Kumar, "Datacenter virtualization with optimization and customization-an efficient way to administer and monitor vsphere and hyper-v," *International Journal of Business Analytics (IJBAN)*, vol. 9, no. 44, pp. 1–6, 2016.