Cloud Formation Mini network

Atlantic Technological University

Computing in DevOps

Mara Lima

2022

Table of Contents

[Introduction 3](#_Toc117356447)

[Method 3](#_Toc117356448)

[Results and Testing 3](#_Toc117356449)

[Conclusion 4](#_Toc117356450)

[References & Bibliography 6](#_Toc117356451)

[Appendices 7](#_Toc117356452)

## **Introduction**

This Lab introduces and implements a mini network using Cloud Formation infrastructure as a code tool from AWS, which discusses how to implement, how to deploy, how it works, and how it helps to easily manage resources.

**Aims/Objectives**

* Create a repository using GIT and commit all code.
* Create a Cloud Formation mini network template using the designer in AWS.
* Deploy the stack and demonstrate the changes that were done to the infrastructure code.
* Describe how Infrastructure as Code helps improve scripted pipeline on the creation and deployment of resources.

## **Method**

* GIT Hub was used to creating a new Infrastructure as Code repository, see Appendix, Figure 1.
* A cloud formation template was created using the Designer feature in AWS, see
* All resources were linked and referenced to their specific components.
* Comments were added to the infrastructure code to easily identify the resources used.
* The IDE Visual Studio Code was used to update the yaml files.
* The code changes were then pushed to the remote repository.
* A new stack ML-MiniNetowrk was initiated in AWS.
* All resources were deployed and properly start-up and have a current state of running.

## **Results and Testing**

* A mini network template was created using the designer in AWS, the template was updated using Visual Studio Code.
* All code changes were pushed to the remote repository in GIT Hub.
* A stack was created, and the mini network was deployed without errors.
* Research was done into how infrastructure as code helps on the implementation and deployment of resources enhancing the efficiency and delivery of resources.

## 

## **Conclusion**

The infrastructure as code tool used in this lab was Cloud Formation from AWS, Cloud Formation is an automated platform that uses templates or yaml files to setup, configure and deploy resources in AWS, speeding up the deployment as multiple resources and services can be initiated at the same time using the same template, enabling a consistent and repeatable deployment procedure that also allows for easily updates and/or changes.

As has been demonstrated in this lab, a mini network was created through Cloud Formation, a template was created using the designer feature, the designer allows you to easily select and drop the resources required into the template. The first resource added was a Virtual Private Network, VPC, along with four subnets, two Publics and two Privates, one EC2 instance was created inside of each subnet followed by their respective security groups. An internet gateway was then added to allow the communication between the instances, a public and private route tables were added, and routes were then created to direct the network traffic, a NAT Gateway followed by an Elastic IP was also dropped into the template.

Although I found it easy to drag and drop the resources and the designer normally associates the resources within the VPC, I had a few issues where the association was being duplicated, for example the network interfaces association were being duplicated on the instances and I had to manually deleted each one of them, another association that was done manually was the internet gateway as it was dropped outside of the VPC.

After creating all the resources, attaching each one to their respective connections and dependencies I saved the template and I switched to an IDE, Visual Studio Code, to easily update the file but also to push the infrastructure code to the GIT Hub repository. The use of an IDE helps on coding best practices as there are a few extensions, such as Cloud Formation Linter, that can be added that suggest improvements.

The security groups were updated with inbound and outbound rules to control the traffic between each instance, the JumpBox is the instance attached to public subnet and its security group allows the connection to the AppServer instance, also on a public subnet. The AppServer security group have a rule that allows connection to the WebApp that exist on a private subnet, and from the WebApp a connection to the Database server its allowed, I found it somewhat difficult to create the right rules and deeper research is needed to fully understand networking, although the reachability within the instances was successfully created.

A NAT Gateway was created, and an Elastic IP address was linked to it to allow the instances in the private subnets to be seen, the WebApp and Database server

Taking everything into account, the use of

Although I do believe

In the final analysis,

## 

## **References & Bibliography**

What is Cloud Formation? (2022) Available at:

<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html>

(Accessed: 02 November 2022)

How does AWS CloudFormation work? (2022) Available at:

<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-whatis-howdoesitwork.html>

(Accessed: 02 November 2022)

How Amazon VPC works (2022) Available at:

<https://docs.aws.amazon.com/vpc/latest/userguide/how-it-works.html#vpc-private-ipv4-addresses>

(Accessed: 04 November 2022)

Control traffic to resources using security groups (2022) Available at:

<https://docs.aws.amazon.com/vpc/latest/userguide/VPC_SecurityGroups.html#VPCSecurityGroups>

(Accessed: 08 November 2022)

## **Appendices**

[Figure 1 - Git Bash Initiation 8](#_Toc117098084)

Figure 1 - Git Bash Initiation