Cloud Formation Mini network

Atlantic Technological University

Computing in DevOps

Mara Lima

2022

Table of Contents

[Introduction 3](#_Toc119261989)

[Method 3](#_Toc119261990)

[Results and Testing 3](#_Toc119261991)

[Conclusion 4](#_Toc119261992)

[References & Bibliography 6](#_Toc119261993)

[Appendices 7](#_Toc119261994)

## **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 create a new Infrastructure as a Code repository, see Appendix, Figure 1.
* A cloud formation template was created using the Designer feature in AWS.
* All resources were linked and referenced to their specific components, see Appendix, Figure 2.
* 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, see Appendix, Figure 3.
* A new stack ML-MiniNetowrk was initiated in AWS, see Appendix, Figure 4.
* All resources were deployed and properly start-up and have a current state of running, see Appendix, Figure 5.
* A ping request was successfully made to test the connection to the JumpBox instance, see Appendix, Figure 6.
* VPC Reachability Analyzer was used to test connection within the VPC, see Appendix, Figure 7 to Figure 12.

## **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.
* The research was done into how infrastructure as code helps in 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 easy 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 communication between the instances, 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 delete 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 and attaching each one to their respective connections and dependencies I saved the template and switched to an IDE, Visual Studio Code, to easily update the file and to push the infrastructure code to the GIT Hub repository. The use of an IDE helps with coding best practices as there are a few extensions, such as Cloud Formation Linter, that can be added that suggest code 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 a 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 exists on a private subnet, and from the WebApp a connection to the Database server is allowed, I found it somewhat difficult to create the right rules and deeper research is needed to fully understand networking, although the VPC reachability within the instances was successfully created and tested with the VPC Reachability tool provided by AWS.

To enhance the performance and high availability, each subnet was deployed in a different availability zone forming an isolated domain that runs in complete autonomy, as a consequence of that, the template can be conveniently updated to deploy a duplicated identical set of resources that will allow the instances to continue operating in case of failure with minimal disruptions and minimal downtime window to the application and users.

A unique set of parameters for the Key Name, Instance type and SSH location that allows connection to the JumpBox were added to the YAML template along with an AMI mapping based on the region, an output to export the VPC details was also added, and the use of these features give flexibility to the templates helping achieve code reusability, doing so if an update is needed to the deployed stack the parameters and mapping can without too much effort be customized to accomplish the desired values.

Taking everything into account, I found the AWS CloudFormation platform challenging and complex to understand and implement, therefore it takes some time and attempts to nicely fit every resource together, a better networking understanding is also needed beforehand the whole implementation.

Although I do believe, the use of AWS Cloud Formation infrastructure as a code tool helps automate the resource provision by defining its configuration only once through a template that can be easily updated when needed, the stack configuration can be tracked and monitored in a way that helps find the changes made to the deployed resources over time, in addition allowing to scale additional resources rapidly, streamlining and enhancing the efficiency of the deployment at the same time improving the use of scripted pipelines in a DevOps environment.

## **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)

Mara Lima – Git Hub repository

<https://github.com/L00177676/ml_mini_network>

## **Appendices**

Figure 1 - GitHub repository 8

Figure 2 - CloudFormation Designer 8

Figure 3 - Git commits 9

Figure 4 - Stack 10

Figure 5 - Resources 11

Figure 6 - JumpBox Ping 12

Figure 7 - VPC Analyzer JumpBox to AppServer 12

Figure 8 - JumpBox to AppServer detailed path 13

Figure 9 - AppServer to WebServer 14

Figure 10 - AppServer to WebServer detailed path 14

Figure 11 - WebApp to Database 15

Figure 12 - WebApp to Database detailed path 16

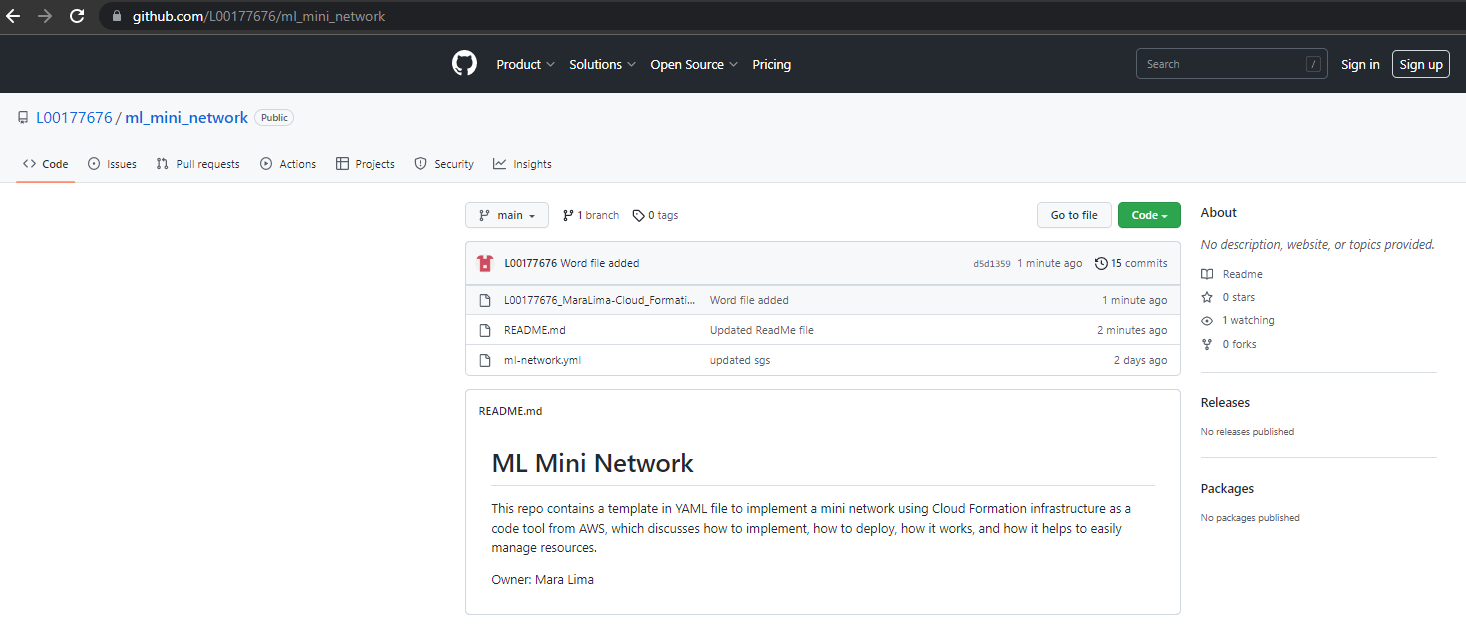


Figure 1 - GitHub repository

A screenshot of a computer

Description automatically generated with low confidence

Figure 2 - CloudFormation Designer

Graphical user interface, text, application

Description automatically generated

Figure 3 - Git commits

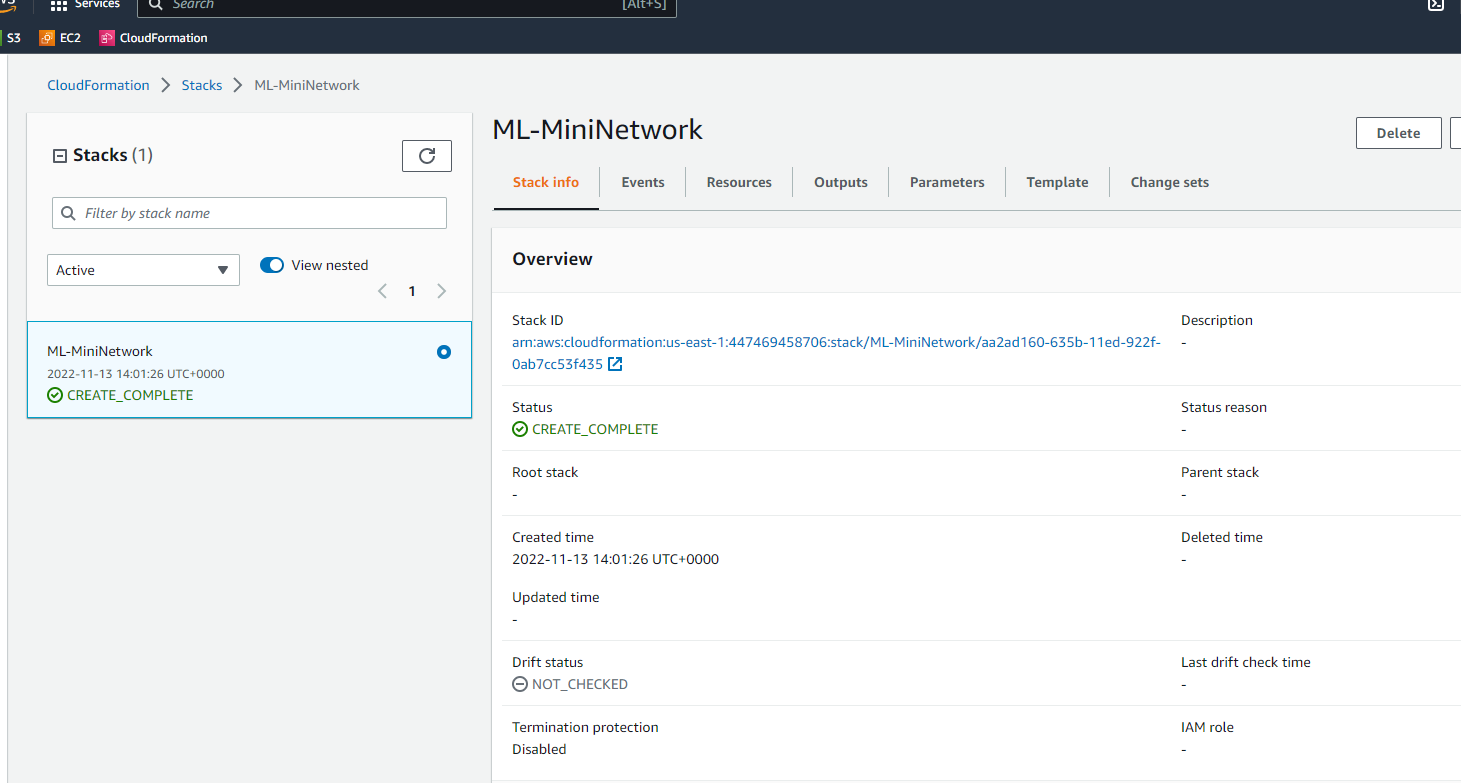


Figure 4 - Stack

Graphical user interface, application

Description automatically generated

Figure 5 - Resources

Graphical user interface

Description automatically generated

Figure 6 - JumpBox Ping

Figure 7 - VPC Analyzer JumpBox to AppServer

Graphical user interface, text, application, email

Description automatically generated

Table

Description automatically generated with low confidence

Figure 8 - JumpBox to AppServer detailed path

Graphical user interface, text, application, email

Description automatically generated

Figure 9 - AppServer to WebServer

Graphical user interface, text, application

Description automatically generated

Figure 10 - AppServer to WebServer detailed path

Graphical user interface, application

Description automatically generated

Figure 11 - WebApp to Database

Text

Description automatically generated with low confidence

Figure 12 - WebApp to Database detailed path