A logo of a cheese head

Description automatically generated

|  |
| --- |
|  |
| Cheesehead hosting  Test report |
| |  |  |  | | --- | --- | --- | | Author: Group 1 | Date: 10/01/2024 | Version: 1.0 | |

Table of Contents

[Introduction 2](#_Toc156567562)

[Website functionality 3](#_Toc156567563)

[Visiting the website over DNS 3](#_Toc156567564)

[Registering a new user 4](#_Toc156567565)

[Logging in 5](#_Toc156567566)

[Deploy website environment. 6](#_Toc156567567)

[Ticket portal 7](#_Toc156567568)

[Terraform 10](#_Toc156567569)

[VPC overview Private 10](#_Toc156567570)

[VPC overview Public 10](#_Toc156567571)

[Check if the subnets are created. 11](#_Toc156567572)

[Check if the routing table is created. 11](#_Toc156567573)

[Check if the security groups are created. 11](#_Toc156567574)

[Check if the ECS cluster is created. 12](#_Toc156567575)

[Check if the EC2 instances are created. 12](#_Toc156567576)

[Check if the EC2 instances are registered in the cluster. 12](#_Toc156567577)

[Check if the EC2 template is created. 13](#_Toc156567578)

[Check if the Autoscaler is created. 13](#_Toc156567579)

[Check if the loadbalancer is created. 13](#_Toc156567580)

[Visit the website is accessible from the DNS name. 14](#_Toc156567581)

[Connecting to sFTP container. 15](#_Toc156567582)

[Stress testing 16](#_Toc156567583)

[Testing load balancer & auto scaling. 16](#_Toc156567584)

[Monitoring 18](#_Toc156567585)

[Security Measures 19](#_Toc156567586)

[Conclusion 21](#_Toc156567587)

# Introduction

In the test we performed we login to the website with an account that was created and specify the details of the infrastructure we want to create. After the deploy button is clicked the API was triggered and the scripts triggered. This test was successful, all the scripts did their job, and we will show the results in this document.

# Website functionality

## Visiting the website over DNS

To evaluate if the website was accessible over DNS I browsed to: <https://www.cheeseheadhosting.nl/>

A screenshot of a computer

Description automatically generated

Figure 1

In figure 1 you can see that the website is accessible and responsive. The content is properly loaded, meaning that the server hosting and DNS record is working correctly. https://www.cheeseheadhosting.nl/ is functioning correctly.

A screenshot of a computer

Description automatically generatedIn figure 1 you could see that the website is accessible over HTTPS meaning that the certificate that is attached to the website is also working correctly. The certificate itself can be seen in figure 2.

Figure 2

## Registering a new user

To test if the creation of user accounts goes correctly, we visited <https://www.cheeseheadhosting.nl/register> and created an account using finalDemo as username and [finalDemo@hotmail.com](mailto:finalDemo@hotmail.com) as email address. The Password must contain at least one uppercase letter, and one number and one special character. As can be seen in figure 3.

A screenshot of a form

Description automatically generated

Figure 3

In figure 4 you can see the accounts table in our database. This shows that the account has succesfully been created and registered in the database. As a future of security we also applied password salting and hashing.

A computer screen shot of a computer

Description automatically generated

Figure 4

## Logging in

To test the login functionality we went to the login page of the website and used the credentials from the account we have made in the previous step phase, registering an account. This can be seen in figure 5.

A screenshot of a computer

Description automatically generated

Figure 5

In figure 6 you can see that after using the correct credentials the user is send the home page of the customer portal.

A screenshot of a computer

Description automatically generated

Figure 6

## Deploy website environment.

In figure 7 you can see the deployment page. We have created a sample setup for the infrastructure to be deployment. We filled in the required sections and click on deploy.

A screenshot of a computer

Description automatically generated

Figure 7

In figure 8 you can see the email that we have received mentioning that the enviornment has been succesfully deployed and thus that the website functionality is able to kick off the creation of an enviornment.

A screenshot of a computer

Description automatically generated

Figure 8

## Ticket portal

The ticket portal is indicated with a “?” symbol on the bottom right of the screen. This is where customers can explain their issue and if there

A yellow rectangular object with white text

Description automatically generated

Figure 9

This figure shows the first greeting of the ticket portal where customers can either visit a service desk or get direct help from customer support by filling out a request.

A screenshot of a contact form

Description automatically generated

Figure 10

This figure shows the form the customers can fill out to request for help.

A screenshot of a computer

Description automatically generated

Figure 11

Once submitted, they are notified that they will be contacted about the issue soon.

A screenshot of a computer

Description automatically generated

Figure 12

This figure shows our notification system once someone fills out the ticket form to receive assistance.

# Terraform

Terraform successfully created the components specified in its script as will be shown below using screenshots of the final product in the AWS console. Keep in mind that “finalDemo” is the name provided as the project name on the website, this was stored in a variable and used for name convention.

## VPC overview Private

A screenshot of a computer

Description automatically generated

Figure 13, Resource map

In the figure above we can see the networking in the private customer VPC.

## VPC overview Public

A screenshot of a chat

Description automatically generated

Figure 14, Resource map

In the figure above we can see the networking in the public customer VPC.

## Check if the subnets are created.

A screenshot of a computer

Description automatically generated

Figure 15, Subnet list

In the figure above we see the 6 subnets that are created for the client network.

## Check if the routing table is created.

A screenshot of a computer

Description automatically generated

Figure 16, Route tables

In this figure the client route tables are dispalyed.

## Check if the security groups are created.

A screen shot of a computer

Description automatically generated

Figure 17, Security groups

The figure above show the client security groups.

## Check if the ECS cluster is created.

A screenshot of a computer

Description automatically generated

Figure 18, ECS cluster

In the image above the cluster is created.

## Check if the EC2 instances are created.

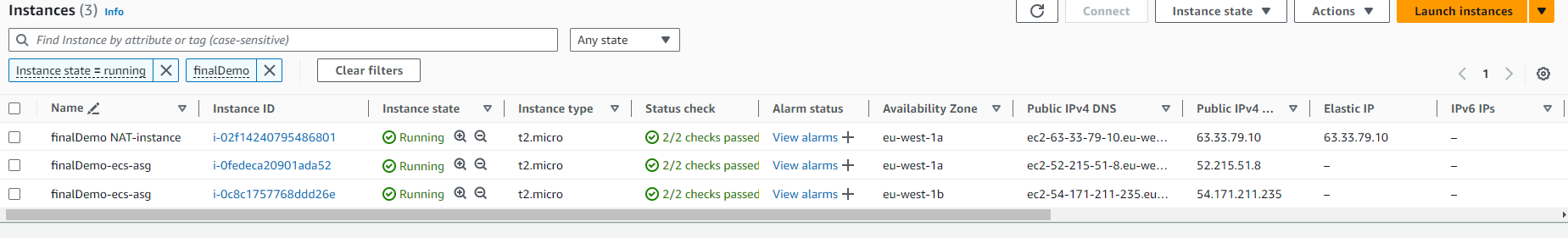


Figure 19, EC2 instances

Figure of the created EC2 instances that are created.

## Check if the EC2 instances are registered in the cluster.

A screenshot of a computer

Description automatically generated

Figure 20, Container instance

One of the EC2 instances is registered to the ECS-cluster to run all the containers on.

## Check if the EC2 template is created.

A screenshot of a computer

Description automatically generated

Figure 21, Launch template

To spin up a EC2 instance using terraform we need to utilize a launch template that defines all the properties for the instance. This is displayed in the image above.

## Check if the Autoscaler is created.

A screenshot of a computer

Description automatically generated

Figure 22, Autoscaling group

In the figure above we can see that the autoscaling group is created.

## Check if the loadbalancer is created.

A screenshot of a computer

Description automatically generated

Figure 23, Target group

In the figure above we can see the target group for the loadbalancer is created.

A screenshot of a computer

Description automatically generated

Figure 24, Loadbalancer

The loadbalancer itself is also created as can be seen in the figure above.

## Visit the website is accessible from the DNS name.

A screenshot of a computer

Description automatically generated

Figure 25, Hosted zone subdomain

In the figure above we can see the created subdomain for the client.

A screenshot of a computer

Description automatically generated

Figure 26, Client sample website

In the figure above we can see what happens when we visit the subdomain. It displays a sample website for the client.

## Connecting to sFTP container.

A screenshot of a computer

Description automatically generated

Figure 27, FTP credentials

In the figure above we see an attempt to connect to the FTP container.

A black text on a white background

Description automatically generated

Figure 28, Failed connection

Unfortunately, we see in the figure above that the FTP connection is not successful.

This means that the Terraform script is able to create the infrastructure successfully unfortunately one component of our network doesn’t work, the sFTP container.

# Stress testing

## Testing load balancer & auto scaling.

To test the load balancer and autoscaling functionality of our environment we have used Locust to start swarming our website. In figure 29 you can see the setup we have used. We kept things small to prevent bigger costs.

A screenshot of a computer

Description automatically generated

Figure 29

In figure 30 you can see that we swarmed the website for a while with clear indicators going over the 50% mark. We initially had it set to 70% but it would not scale up very often because it was harder to create the necessary load.

A screen shot of a graph

Description automatically generated

Figure 30

In figure 31 you can see that after going over the threshold a few times it is able to scale up.

A screenshot of a computer

Description automatically generated

Figure 31

In figure 32 you can see that after we have ended our Locust script the auto scaler also scaled down accordingly.

A screenshot of a computer

Description automatically generated

Figure 32

# Monitoring

When setting the CloudTrail log group for CloudWatch, it can be seen that its data can be successfully used by CloudWatch and implemented into our Dashboard.

A close up of a screen

Description automatically generated

Figure 33

This figure shows the CloudTrail log group being used on the CloudWatch dashboard to show recent activitiues such as EC2 instance shutoffs restarts, etc.

A screenshot of a computer

Description automatically generated

Figure 34

This figure shows the metrics for the CloudWatch dashboard working with no issues.

# Security Measures

DDoS Detection

The lambda function to run the Athena query is successful and so are the notifications. This was tested by simply lowering the alarm threshold and then running the function since we could not test it any other way.

A black screen with many colorful text

Description automatically generated

Figure 35

This figure shows the lambda function being successfully run with no issues. No alarm was sent because there was no attack.

A screenshot of a computer

Description automatically generated

Figure 36

This figure shows the same query being run in athena to show that there were no results. The query checks for packets received over 150000 from 443, 80 and UDP protocols (from UDP flooding possibilties) and there were no results.

A screenshot of a computer

Description automatically generated

Figure 37

This figure shows where the athena queries are stored in S3.

A screenshot of a computer error

Description automatically generated

Figure 38

Once I ran the same thing with a lower threshold, I received an alarm successfully.

# Conclusion

This test was close to 95% successful. The website worked along with its login and register features. The automation scripts worked, Ansible triggered the Terraform script to create the infrastructure and it caught a standard error. Terraform in turn also worked as you may have seen in the chapter above. It created all the specified resources. The Lambda scripts worked, and a confirmation email was sent when the infrastructure was created.

This is all good, but there were a few things that did not work unfortunately. For starters, the FTP is not accessible. The container itself is healthy and running but is not accessible, probably because of a port issue. The other part that is not working is also a container, this is the MySQL container. This one does not start up at all, probably because there is an issue within the Terraform we couldn’t figure out.

These are the results of this test report, as you can see a lot of it works except for a few minor functionalities but this is something that could be worked on in future sprints.