ABSTRACT

The purpose of this report is to describe and explain the development process of a cloud-based application that simulates a DDoS botnet and was used to conduct attacks on two websites hosted on Cloudflare and AWS.

The technologies used to build the application consisted of cloud computing and Docker containers. Considerable research was undertaken to understand how to implement these technologies and the necessary steps in building the simulator.

An iterative approach was applied while developing the simulator and two websites were built with the purpose of testing the system at the end of each stage. The development process started with the research of different DDoS tools and techniques that can be used to perform all types of DDoS attacks: Volumetric, Protocol-based, and Application Layer-based. A Docker image was then built for each one of the chosen tools. The image was added to AWS Elastic Container Registry and deployed in the cloud utilizing both launch types offered by AWS Elastic Container Service: EC2 launch and the serverless compute engine Fargate.

Final testing was only possible with a 50-container deployment due to budget limitations. The results of the attacks on the two websites were measured by analysing page load time and ICMP reply time. The testing outcome revealed the DDoS attacks had no effect on the functionalities of the two websites.

At the end of the report an analysis was conducted to find the techniques employed by the two cloud providers to mitigate the DDoS attacks that were carried out in this project.

TABLE OF FIGURES

TABLE OF CONTENTS

CHAPTER 1 - INTRODUCTION

A Distributed Denial-of-Service (DDoS) attack is a type of cyber-attack in which the perpetrator tries to disrupt the functionality of the victim system by taking advantage of a server misconfiguration, a vulnerability or by simply flooding it with huge amounts of traffic from multiple compromised computers.

DDoS attacks have increased exponentially over time and have become more sophisticated, some of them being associated with ransomware and even state-sponsored cyber-attacks. According to Yoachimik and Ganti (2022), in December 2021, one out of every three Cloudflare clients reported being targeted by a DDoS ransom attack or threatened by an attacker. Cloudflare is the biggest Content Delivery Network provider in the world with a 39.24% market share. (Kumar, 2022)

Numerous ways to protect systems against DDoS attacks exist, however there is no perfect solution to this problem. New DDoS attack toolkits built to evade defences are continually being developed by cyber-criminals so more research is needed to counteract these threats. From the literature review it has been found most DDoS experiments are performed using networking event simulators like Cisco Packet Tracer, Network Simulator Version 2 or in a lab simulation where one computer is attacked by other computers. While these simulations may have value, they are very simplistic and cannot be compared with real world scenarios where a huge number of other software agents exist and interact. To be one step ahead of attackers, researchers should conduct experiments on real world infrastructure.

This project was a hands-on experiment that took advantage of the latest cloud computing and virtualization solutions to build a realistic DDoS experimental platform. The costs related to the cloud computing resources used in this project have been obtained by opening a free-tier AWS account and receiving $300 in free credits from Amazon Web Services (AWS).

1.1 Project aim

The aim of the project is to determine if a DDoS experimental platform can be as realistic as a genuine DDoS attack from a botnet of computers. The proposed solution was to install DDoS attack tools in hundreds of Docker Containers and deploy these in a cloud environment.

1.2 Objectives

1.2.1 Main objectives

- To build a realistic DDoS simulator.
- To test and compare the DDoS defence capabilities of two cloud hosting providers: AWS and Cloudflare.

1.2.2 Specific objectives

1. To build two static websites using Cloudflare and AWS as hosting providers.
2. To research DDoS attack techniques and choose the suitable tools and scripts that can be used for testing the DDoS simulator.
3. To research Docker containers and build the tools/scripts into Docker images.
4. To research cloud computing and the services needed to build the simulator.
5. To add Docker images to AWS Elastic Container Registry.
6. To deploy the Docker containers in the cloud.
7. To test the two websites.
8. To analyse the results and the mitigation techniques used.

1.3 Methodology

The development methodology followed in this project was Agile-Scrum. Although this methodology is typically used by software development teams, its principles and lessons have been successfully applied in building the simulator. Using the Agile-Scrum framework the simulator was built in a series of iterations called sprints where change was expected, testing was done at the end of each stage and features were added as needed.

1.4 Report structure

Chapter 1 introduces the project. The topic of the project is presented along with project objectives, project aim and methodology.

Chapter 2 describes the building of two static websites hosted on Cloudflare and AWS that were used for testing purposes.

Chapter 3 presents the research and identification of the DDoS tools that were used to conduct the attacks.

Chapter 4 focuses on researching the concept of Docker containers and building the attack tools into Docker container images.

Chapter 5 details the cloud computing technologies required to build the DDoS simulator and deploy the Docker images in the cloud.

Chapter 6 describes the testing of the two websites. This was conducted by launching different types of DDoS attacks and analysing the web server response times.

Chapter 7 provides an analysis of DDoS mitigation techniques employed by Cloudflare and AWS to defend the websites from the attacks.

Chapter 8 presents the system evaluation, a personal reflection, and the overall conclusion of the experiment.

CHAPTER 2 – BUILDING TWO STATIC WEBSITES

Building an application such as a DDoS simulator involves continuous testing throughout the development of the project. For this purpose, before starting the work on the DDoS simulator, two websites have been built on two different hosting platforms: Cloudflare and GoDaddy.

2.1 Cloudflare hosted website

The first website was built using a free JAMstack platform called Cloudflare Pages. The prerequisites of building a website with Cloudflare Pages are a Cloudflare account, a GitHub account, and a domain name. The website building process involved a static website generator to create the website files, hosting the files in a GitHub repository, and then pulling and deploying the website with Cloudflare Pages. The static website generator used to build the website files is called Jekyll. This is a very popular framework written in Ruby that can be installed and used from the command line.

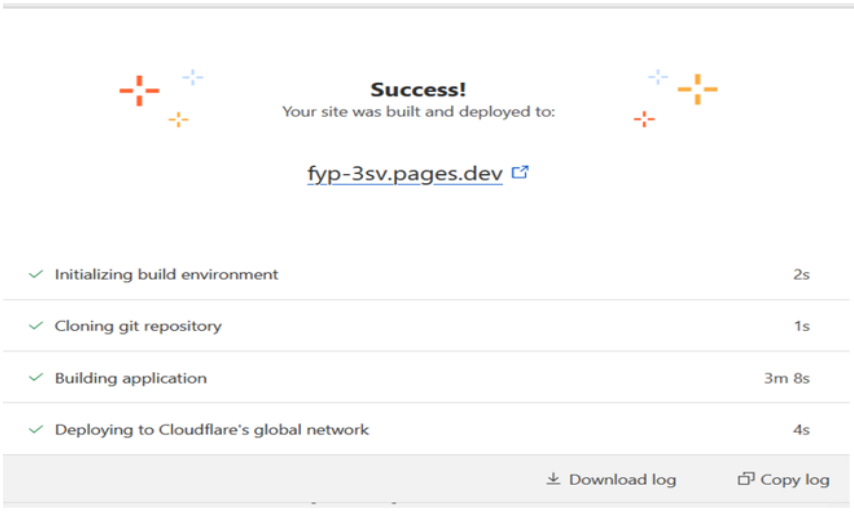
A GitHub Jekyll theme was forked into a new repository and the files were downloaded to the local machine. The file config.yml file was edited in order to customize the theme and the website was launched on localhost port 4000. (Figure 1)

Figure 19. Website launch on localhost port 4000

```
PS C:\Users\mihai\Desktop> cd .\WhatATheme\
PS C:\Users\mihai\Desktop\WhatATheme> bundle exec jekyll serve
Doing `require 'backports'` is deprecated and will not load any backport in the next major releas
e.
Require just the needed backports instead, or 'backports/latest'.
Configuration file: C:/Users/mihai/Desktop/WhatATheme/_config.yml
            Source: C:/Users/mihai/Desktop/WhatATheme
       Destination: C:/Users/mihai/Desktop/WhatATheme/_site
 Incremental build: disabled. Enable with --incremental
      Generating...
       Jekyll Feed: Generating feed for posts
                    done in 1.096 seconds.
 Auto-regeneration: enabled for 'C:/Users/mihai/Desktop/WhatATheme'
  JekyllAdmin mode: production
    Server address: http://127.0.0.1:4000
  Server running... press ctrl-c to stop.
```

The GitHub repository was then deployed to Cloudflare Pages and set up with the domain name “ddossimulation.com”.(Figure 2)

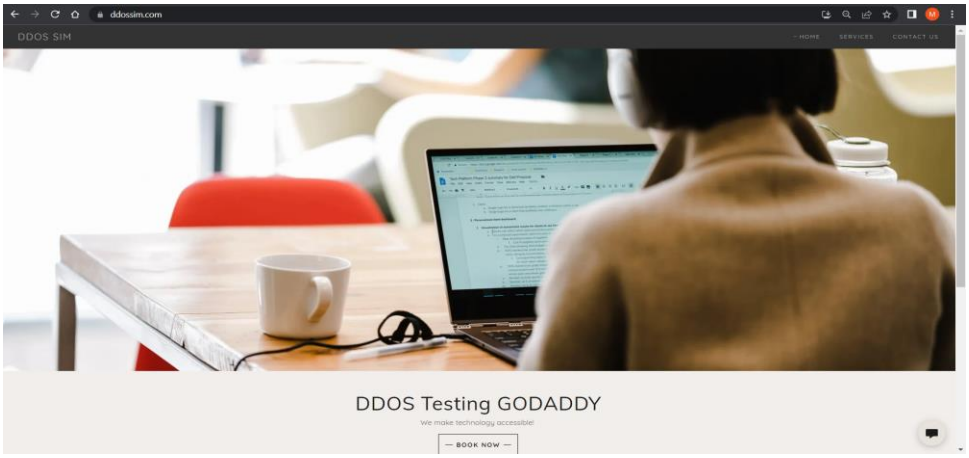
Figure 20. Website deployment



2.2 AWS hosted website

The second website was built with GoDaddy which use AWS cloud infrastructure to host their websites. (Lunden, 2022) This was a straightforward process due to the simple wizard configuration GoDaddy offer. The domain name chosen for this website was “ddossim.com”. (Figure 3)

Figure 21. GoDaddy website front page



CHAPTER 3 - DDOS TOOLS/SCRIPTS

3.1 Research

According to Chickowski(2022) there are multiple ways in which DDoS attacks can be performed, however all techniques tend to fall into 3 main categories:

3.1.1 Volumetric attacks

Servers and network interface cards have set bandwidth limitations. In this type of attack cyber criminals attempt to generate massive amounts of traffic to overwhelm the target, creating a bottleneck and thus making it impossible for the system to serve legitimate traffic. Examples of volumetric attacks include:

- Internet Control Message Protocol (ICMP) Flood is a type of attack in which the attacker attempts to bring down the target system by sending huge amounts of ICMP echo-requests. (Ping (ICMP) flood DDOS attack, 2021)
- User Datagram Protocol (UDP) Flood is launched by sending large amounts of UDP traffic, flooding random ports on the targeted server. (*UDP flood attack*, 2021)
- Domain Name System (DNS) amplification is a denial-of-service attack in which the attackers use spoofed IP addresses to send small requests such as “ANY” that require long replies from DNS resolvers. The spoofed attacker IP addresses point to the victim which receives large amounts of traffic from the DNS resolvers making their system inoperable. (DNS amplification attack, 2022)

3.1.2 Protocol-based attacks

Protocol-based attacks are designed to consume the processing capacity of network infrastructure by exploiting a weakness in Layer 3 or Layer 4 of the OSI model. Computers have a set number of TCP and UDP ports, if the attack keeps all ports busy, the server will be unable to accept new connections. Examples of Protocol-based attacks include:

- Transport Control Protocol (TCP) SYN Flood. TCP is a connection-oriented protocol. A three-way handshake must be done between the server and the client to establish a connection. The three-way handshake starts with the client sending a SYN request, receiving a SYN-ACK message back from the server, and in the final phase the client sends back an ACK message. In a TCP SYN flood attack the client sends numerous SYN requests to random ports on the server, it receives the SYN-ACK message, but it never sends the ACK message back, leaving the server hanging and keeping the connection open. This consumes the bandwidth and makes the server inoperable. (Jia et al., 2020)
- Ping of Death. This attack takes advantage of the ICMP protocol. The Internet Protocol states that the maximum size of a packet that can be transferred is 65,535 bytes. The attacker sends multiple packet fragments that, when reassembled by the target system, end up as oversized packets. This could lead to system overflow and crash the target system. (Ping of death DDoS attack, 2022)

3.1.3 Application Layer attacks

These are some of the more sophisticated DDoS attacks which exploit weaknesses in Layer 7 - the application layer by opening connections and initiating process and transaction requests that consume finite resources like disk space and available memory.

Examples of Application layer attacks:

- Slowloris is a type of attack in which the attacker takes advantage of poorly configured web servers. Slowloris sends multiple HTTP GET requests with an incomplete HTTP header to the web server making the server wait indefinitely for the complete message and thus depleting its resources. (Dhanapal and Nithyanandam, 2019)
- Cache-busting attack is an advanced type of HTTP attack that uses variations in the Uniform Resource Identifier (URL) to circumvent Content Delivery Network (CDN) providers. CDN providers serve cached pages of a website taking workload from the origin web server and making the website have faster response times. The attacker requests pages that are not cached so the CDN must contact the origin web server for every page request. A very large number of requests causes additional strain on the web server and can result in denial-of service. (Application layer attacks, 2022)

3.2 Choosing the attack tools

DDoS attack tools are software programs usually developed by security professionals to perform stress tests against their own networks however these tools can also be adapted by cyber-criminals to launch genuine attacks. Other tools like R.U.D.Y or Saphyra have been developed by hackers specifically for conducting DDoS attacks.

Two DDoS attack tools and one python script have been chosen for this project with the aim of experimenting all three types of attacks:

- Hping3 was used to conduct volumetric and protocol-based attacks. This tool is able to send TCP SYN flood, UDP flood, ICMP flood and RAW-IP packet attacks. It is a commonly used tool to test firewalls, network performance or determine the Maximum Transmission Unit on the network path between two IP hosts. (hping3 | Kali Linux Tools, 2022)
- R.U.D.Y. is an application layer, low-and-slow type of attack executed in two phases. In the first phase the tool crawls the application looking for a web form that allows users to enter data. In the second phase the tool creates a HTTP POST request in which the header informs the server that a long form is about to be submitted. The process is then dragged by submitting small pieces of content at random intervals and thus keeping the server connection open indefinitely. (R U Dead Yet? (R.U.D.Y.) attack, 2022)
- Saphyra is a python script developed by the hacker group Anonymous. This tool employs multiple attack vectors however the main one is sending a combination of over 3200 unique user agent request header strings and 300 referrer field strings to execute a cache-busting attack. (McMillen, 2022) User agent request header strings are sent when making a web request, they let the server identify the application, system, operating system vendor and version of the requesting user agent. The referrer field strings inform the server about the address of the last page the user was before making the request. (HTTP headers - HTTP | MDN, 2022) The combination of these two results in over a million unique user agent/referrer strings sent to the web server which allows the attacker to bypass the cache CDN engines and overload the web server.

CHAPTER 4 – BUILDING THE DDOS TOOLS/SCRIPTS INTO A DOCKER IMAGE

4.1 Research

Similar to virtual machines, containers are a technology utilized in creating virtualized computing environments. The primary difference between the two is containers are much lighter and thus much faster than virtual machines. While VM's are measured in gigabytes, containers are usually measured in megabytes. The reason containers are lighter than VM's is the different ways the two share the resources of the host system. While VM's use a software layer called Hypervisor to run their own operating system on top of the host operating system, containers take advantage of a Linux technology called namespaces that makes it possible to share the kernel of the host operating system with the host and other containers. A container does not need a whole operating system because it can use the host kernel. For example, Linux distributions like CentOS, Fedora or Arch running in container environments on a single Ubuntu host will all use the Ubuntu kernel. (Containers vs. Virtual Machines (VMs): What's the Difference?, 2022)

Docker uses a client-server architecture. The Docker client is the primary way users interact with Docker. This is done by using a set of Docker commands in a command line interface like Linux CLI or Windows PowerShell. The Docker daemon listens to API requests from Docker client and does all the background work of building and managing Docker objects like images, containers, or networks. (Docker overview, 2022)

The starting point with Docker containers is the Docker image. This image acts as a read-only template, a set of instructions used to build a container. An image can be built from another image with additional customization. The image needs to contain the application code and all the necessary dependencies to make the application run. Once created, the image can become one or more instances of a container. (Gillis, 2022)

4.2 The development process

4.2.1 Saphyra

The hands-on part of the project started with downloading and installing Docker engine on a local computer that runs Windows and on a virtual machine that runs Linux. The first image that was built was the python script Saphyra. The script was cloned from a GitHub repository to the local Linux machine, the script files consisted of two executable files and other dependencies.

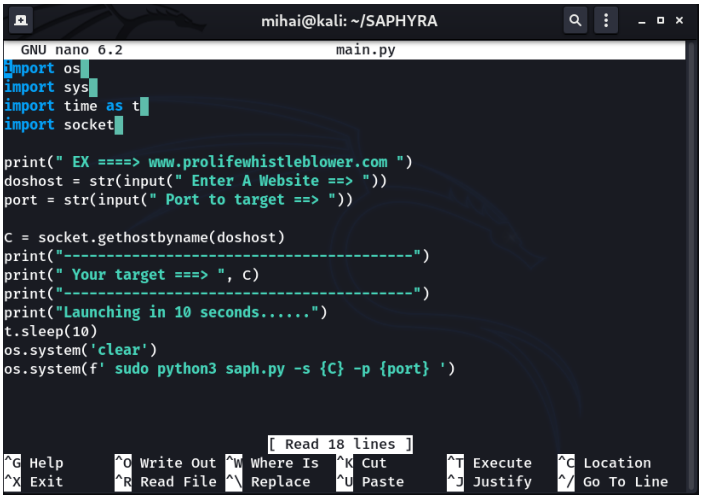
To build the image, a custom Docker file was created using the text editor Nano. The Docker file must be named "Dockerfile" without any extension so that the Docker engine understands this file is the one that contains the set of instructions to build the image. The contents of this file are split into three categories:

1. Specifying a base image
2. Installing additional programs
3. Setting up the command to run on container start up

The script was first tested in the Linux virtual machine before building it into a Docker image. The script was composed of two parts: "main.py" and "saph.py". Analysing the code in the two files, it was found "main.py" was used to get user input for the IP address and port of the target. Once the input was

taken from the user, the other script - “saph.py” was then executed with the command “sudo python3 saph.py -s {C} -p {port}”. (Figure 4)

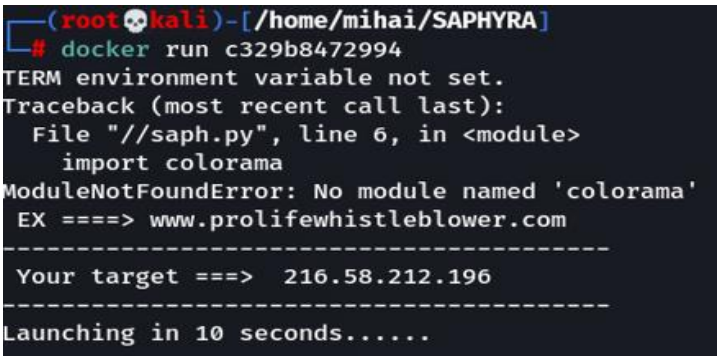
Figure 22. Saphyra "main.py" file



The problem encountered here was that executing the script in a container should be done automatically, without any user input. To solve this issue, “main.py” was not included in the Dockerfile and the command to execute “saph.py” was added to the Dockerfile as the main command for the container. (Figure 6)

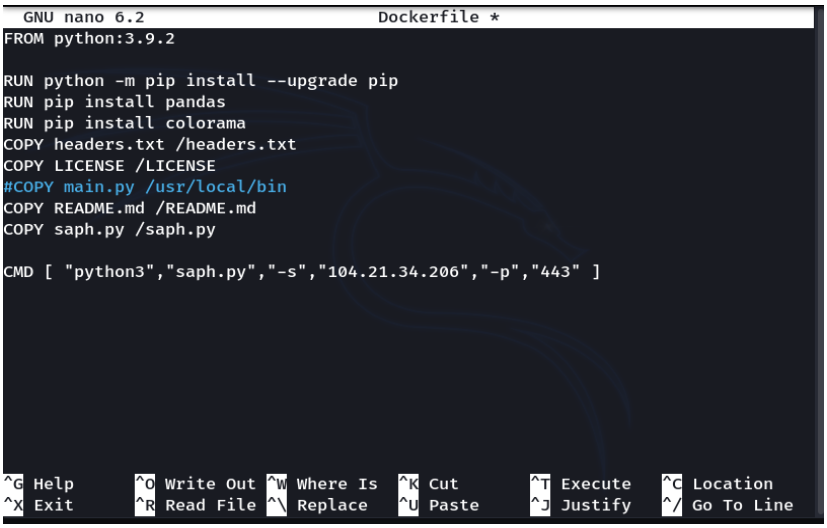
Next, Python 3.9.2 was added to Dockerfile as a base image but the error “No module named colorama” came up when running the container. (Figure 5)

Figure 23. “No module named colorama” error



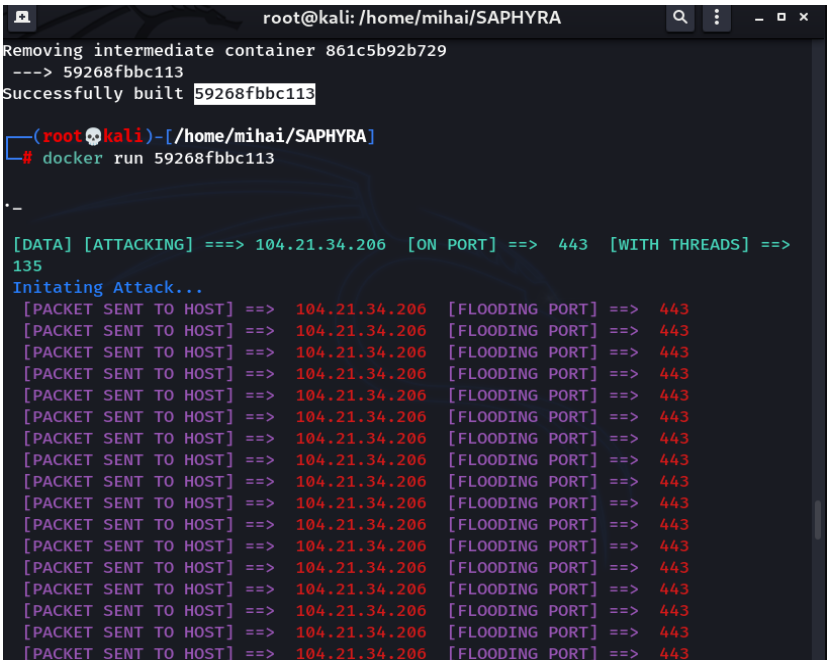
To fix this error, the module “colorama” together with the “pip” package management system have been added to the container image. ([Fixed] ModuleNotFoundError: No module named ‘colorama, 2022) (Figure 6)

Figure 24. Saphyra Dockerfile



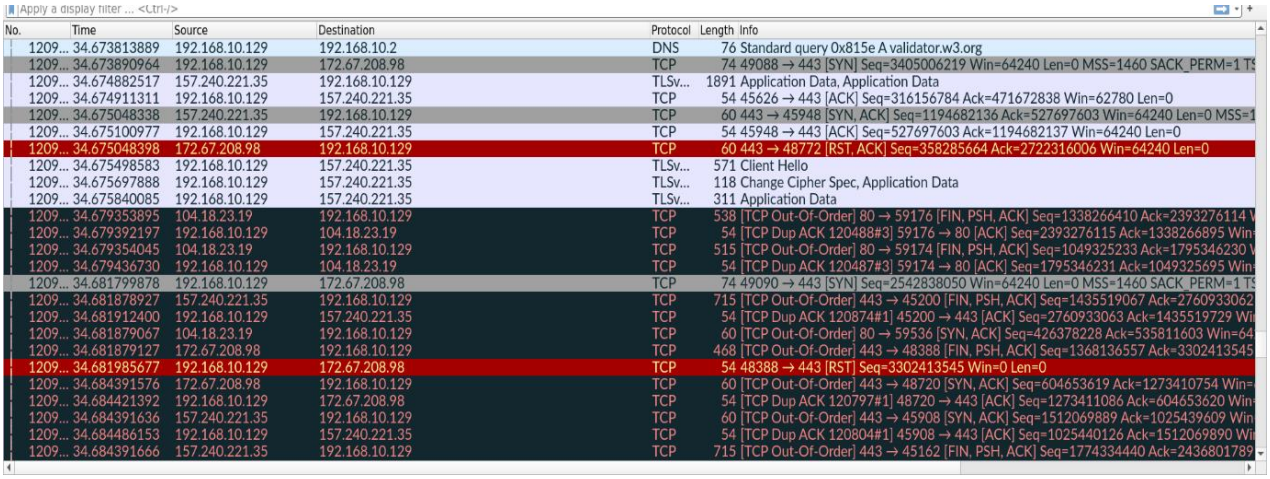
The container image was successfully built and set to attack “ddossimulation.com” port 443. (Figure7)

Figure 25. Saphyra attack from local Linux machine



Wireshark was then used to analyse the traffic between the local machine and the website. Saphyra is a low-and-slow attack tool but also employs other attack vectors. As it can be seen bellow, Saphyra opens multiple handshake connections using port 80 (HTTP) and 443 (HTTPS). The web server sends SYN ACK messages back but closes these connections as it does not receive an ACK message. (Figure 8)

Figure 26. Wireshark caption of the Saphyra attack

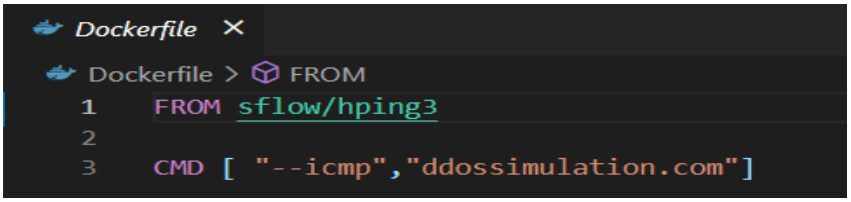


No.	Time	Source	Destination	Protocol	Length	Info
1209...	34.673813889	192.168.10.129	192.168.10.2	DNS	76	Standard query 0x815e A validator.w3.org
1209...	34.673890964	192.168.10.129	172.67.208.98	TCP	74	49088 → 443 [SYN] Seq=3405006219 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TS
1209...	34.674882517	157.240.221.35	192.168.10.129	TLSv...	1891	Application Data, Application Data
1209...	34.674911311	192.168.10.129	157.240.221.35	TCP	54	45626 → 443 [ACK] Seq=316156784 Ack=471672838 Win=62780 Len=0
1209...	34.675048338	157.240.221.35	192.168.10.129	TCP	60	443 → 45948 [SYN, ACK] Seq=1194682136 Ack=527697603 Win=64240 Len=0 MSS=1
1209...	34.675100977	192.168.10.129	157.240.221.35	TCP	54	45948 → 443 [ACK] Seq=527697603 Ack=1194682137 Win=64240 Len=0
1209...	34.675048398	172.67.208.98	192.168.10.129	TCP	60	443 → 48772 [RST, ACK] Seq=358285664 Ack=2722316006 Win=64240 Len=0
1209...	34.675498583	192.168.10.129	157.240.221.35	TLSv...	571	Client Hello
1209...	34.675697888	192.168.10.129	157.240.221.35	TLSv...	118	Change Cipher Spec, Application Data
1209...	34.675840085	192.168.10.129	157.240.221.35	TLSv...	311	Application Data
1209...	34.679353895	104.18.23.19	192.168.10.129	TCP	538	[TCP Out-Of-Order] 80 → 59176 [FIN, PSH, ACK] Seq=1338266410 Ack=2393276114
1209...	34.679392197	192.168.10.129	104.18.23.19	TCP	54	[TCP Dup ACK 120488#3] 59176 → 80 [ACK] Seq=2393276115 Ack=1338266895 Win
1209...	34.679354045	104.18.23.19	192.168.10.129	TCP	515	[TCP Out-Of-Order] 80 → 59174 [FIN, PSH, ACK] Seq=1049325233 Ack=1795346230
1209...	34.679436730	192.168.10.129	104.18.23.19	TCP	54	[TCP Dup ACK 120487#3] 59174 → 80 [ACK] Seq=1795346231 Ack=1049325695 Win
1209...	34.681799878	192.168.10.129	172.67.208.98	TCP	74	49090 → 443 [SYN] Seq=2542838050 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TS
1209...	34.681878927	157.240.221.35	192.168.10.129	TCP	715	[TCP Out-Of-Order] 443 → 45200 [FIN, PSH, ACK] Seq=1435519067 Ack=2760933062
1209...	34.681912400	192.168.10.129	157.240.221.35	TCP	54	[TCP Dup ACK 120874#1] 45200 → 443 [ACK] Seq=2760933063 Ack=1435519729 Wi
1209...	34.681879067	104.18.23.19	192.168.10.129	TCP	60	[TCP Out-Of-Order] 80 → 59536 [SYN, ACK] Seq=426378228 Ack=535811603 Win=64
1209...	34.681879127	172.67.208.98	192.168.10.129	TCP	468	[TCP Out-Of-Order] 443 → 48388 [FIN, PSH, ACK] Seq=1368136557 Ack=3302413545
1209...	34.681985677	192.168.10.129	172.67.208.98	TCP	54	48388 → 443 [RST] Seq=3302413545 Win=0 Len=0
1209...	34.684391576	172.67.208.98	192.168.10.129	TCP	60	[TCP Out-Of-Order] 443 → 48720 [SYN, ACK] Seq=604653619 Ack=1273410754 Win=
1209...	34.684421392	192.168.10.129	172.67.208.98	TCP	54	[TCP Dup ACK 120797#1] 48720 → 443 [ACK] Seq=1273411086 Ack=604653620 Win
1209...	34.684391636	157.240.221.35	192.168.10.129	TCP	60	[TCP Out-Of-Order] 443 → 45908 [SYN, ACK] Seq=1512069889 Ack=1025439609 Win
1209...	34.684486153	192.168.10.129	157.240.221.35	TCP	54	[TCP Dup ACK 120804#1] 45908 → 443 [ACK] Seq=1025440126 Ack=1512069890 Wi
1209...	34.684391666	157.240.221.35	192.168.10.129	TCP	715	[TCP Out-Of-Order] 443 → 45162 [FIN, PSH, ACK] Seq=1774334440 Ack=2436801789

4.2.2 Hping3

For the second tool, Hping3, a different approach was taken to build a container image. Unlike Saphyra, Hping3 is a commonly used tool to test systems against DoS attacks, and it comes preinstalled in Kali Linux. Container images for commonly used programs can usually be found in Docker Hub. Like GitHub, Docker Hub is a cloud-based repository where users can download or publish container images. A Hping3 image was pulled from Docker Hub and the container was configured to run an ICMP flood attack against the website “ddossimulation.com”. (Figures 9 & 10) The Hping3 container image was later modified to execute other types of attacks like TCP and UDP flood against the two websites.

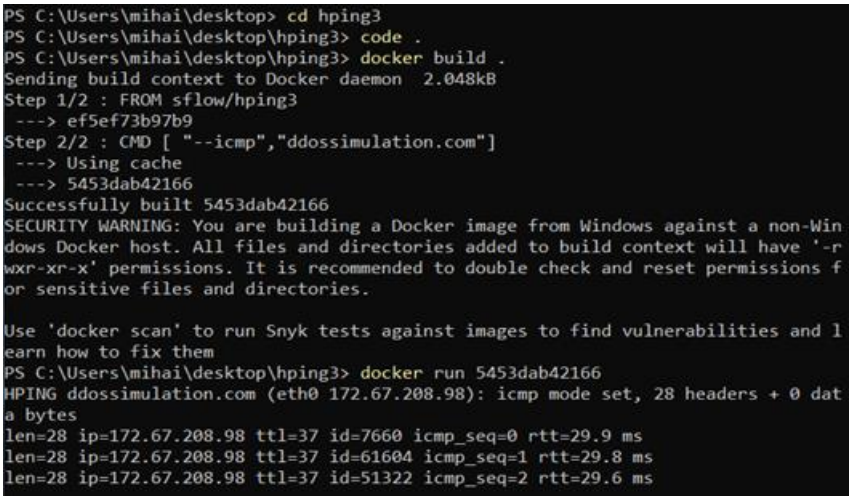
Figure 27. Hping3 ICMP attack Dockerfile



Dockerfile

```
Dockerfile > FROM
1   FROM sflow/hping3
2
3   CMD [ "--icmp","ddossimulation.com"]
```

Figure 28. Hping3 ICMP attack configuration

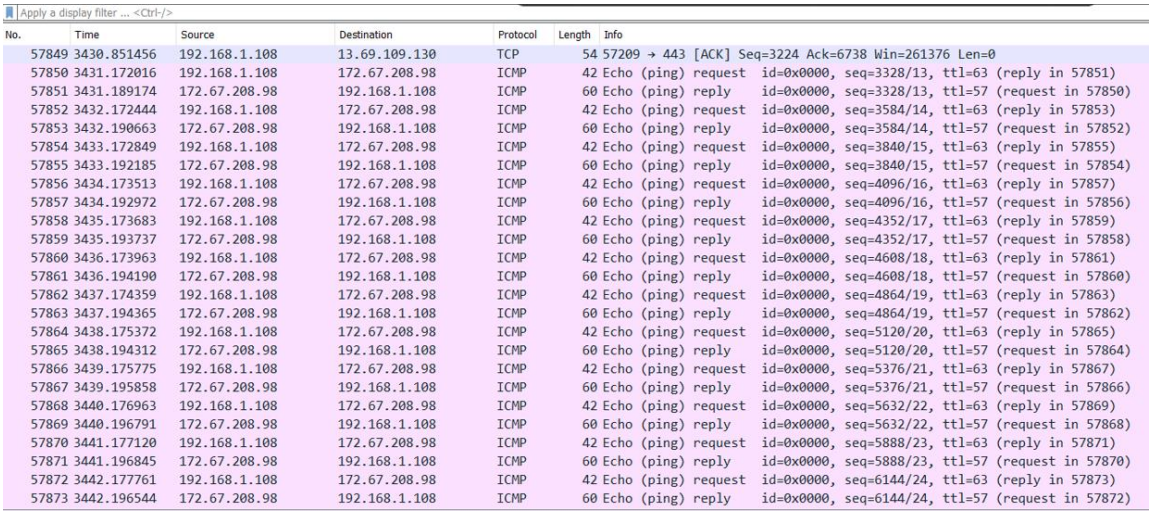


```
PS C:\Users\mihai\desktop> cd hping3
PS C:\Users\mihai\desktop\hping3> code .
PS C:\Users\mihai\desktop\hping3> docker build .
Sending build context to Docker daemon  2.048kB
Step 1/2 : FROM sflow/hping3
 ---> ef5ef73b97b9
Step 2/2 : CMD [ "--icmp","ddossimulation.com"]
 ---> Using cache
 ---> 5453dab42166
Successfully built 5453dab42166
SECURITY WARNING: You are building a Docker image from Windows against a non-Win
dows Docker host. All files and directories added to build context will have '-r
wxr-xr-x' permissions. It is recommended to double check and reset permissions f
or sensitive files and directories.

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and l
earn how to fix them
PS C:\Users\mihai\desktop\hping3> docker run 5453dab42166
HPING ddossimulation.com (eth0 172.67.208.98): icmp mode set, 28 headers + 0 dat
a bytes
len=28 ip=172.67.208.98 ttl=37 id=7660 icmp_seq=0 rtt=29.9 ms
len=28 ip=172.67.208.98 ttl=37 id=61604 icmp_seq=1 rtt=29.8 ms
len=28 ip=172.67.208.98 ttl=37 id=51322 icmp_seq=2 rtt=29.6 ms
```

Analysing the local traffic with Wireshark shows Hping3 flooding the web server with ICMP requests. (Figure 11)

Figure 29. Wireshark caption of the ICMP attack



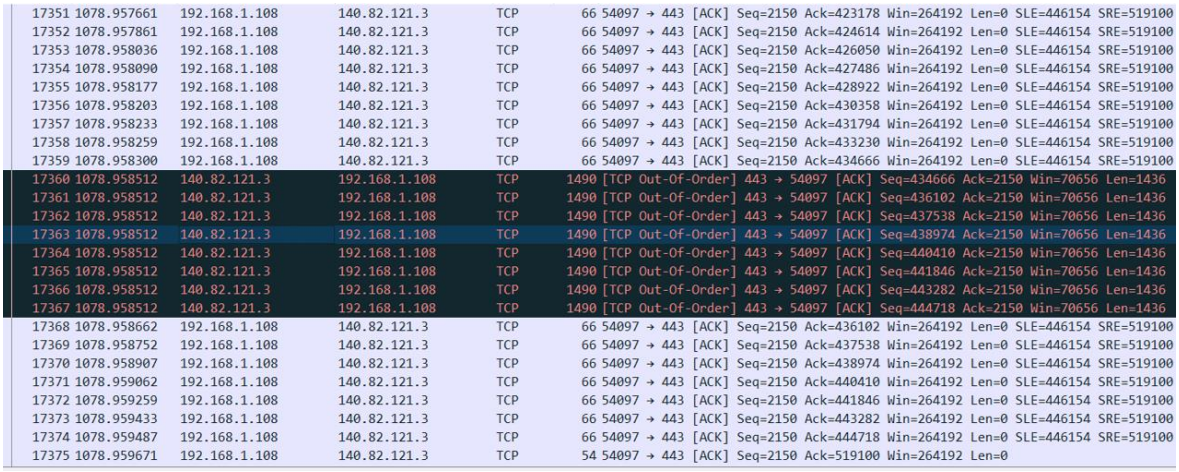
Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
57849	3430.851456	192.168.1.108	13.69.109.130	TCP	54	57209 → 443 [ACK] Seq=3224 Ack=6738 Win=261376 Len=0
57850	3431.172016	192.168.1.108	172.67.208.98	ICMP	42	Echo (ping) request id=0x0000, seq=3328/13, ttl=63 (reply in 57851)
57851	3431.189174	172.67.208.98	192.168.1.108	ICMP	60	Echo (ping) reply id=0x0000, seq=3328/13, ttl=57 (request in 57850)
57852	3432.172444	192.168.1.108	172.67.208.98	ICMP	42	Echo (ping) request id=0x0000, seq=3584/14, ttl=63 (reply in 57853)
57853	3432.190663	172.67.208.98	192.168.1.108	ICMP	60	Echo (ping) reply id=0x0000, seq=3584/14, ttl=57 (request in 57852)
57854	3433.172849	192.168.1.108	172.67.208.98	ICMP	42	Echo (ping) request id=0x0000, seq=3840/15, ttl=63 (reply in 57855)
57855	3433.192185	172.67.208.98	192.168.1.108	ICMP	60	Echo (ping) reply id=0x0000, seq=3840/15, ttl=57 (request in 57854)
57856	3434.173513	192.168.1.108	172.67.208.98	ICMP	42	Echo (ping) request id=0x0000, seq=4096/16, ttl=63 (reply in 57857)
57857	3434.192972	172.67.208.98	192.168.1.108	ICMP	60	Echo (ping) reply id=0x0000, seq=4096/16, ttl=57 (request in 57856)
57858	3435.173683	192.168.1.108	172.67.208.98	ICMP	42	Echo (ping) request id=0x0000, seq=4352/17, ttl=63 (reply in 57859)
57859	3435.193737	172.67.208.98	192.168.1.108	ICMP	60	Echo (ping) reply id=0x0000, seq=4352/17, ttl=57 (request in 57858)
57860	3436.173963	192.168.1.108	172.67.208.98	ICMP	42	Echo (ping) request id=0x0000, seq=4608/18, ttl=63 (reply in 57861)
57861	3436.194190	172.67.208.98	192.168.1.108	ICMP	60	Echo (ping) reply id=0x0000, seq=4608/18, ttl=57 (request in 57860)
57862	3437.174359	192.168.1.108	172.67.208.98	ICMP	42	Echo (ping) request id=0x0000, seq=4864/19, ttl=63 (reply in 57863)
57863	3437.194365	172.67.208.98	192.168.1.108	ICMP	60	Echo (ping) reply id=0x0000, seq=4864/19, ttl=57 (request in 57862)
57864	3438.175372	192.168.1.108	172.67.208.98	ICMP	42	Echo (ping) request id=0x0000, seq=5120/20, ttl=63 (reply in 57865)
57865	3438.194312	172.67.208.98	192.168.1.108	ICMP	60	Echo (ping) reply id=0x0000, seq=5120/20, ttl=57 (request in 57864)
57866	3439.175775	192.168.1.108	172.67.208.98	ICMP	42	Echo (ping) request id=0x0000, seq=5376/21, ttl=63 (reply in 57867)
57867	3439.195858	172.67.208.98	192.168.1.108	ICMP	60	Echo (ping) reply id=0x0000, seq=5376/21, ttl=57 (request in 57866)
57868	3440.176963	192.168.1.108	172.67.208.98	ICMP	42	Echo (ping) request id=0x0000, seq=5632/22, ttl=63 (reply in 57869)
57869	3440.196791	172.67.208.98	192.168.1.108	ICMP	60	Echo (ping) reply id=0x0000, seq=5632/22, ttl=57 (request in 57868)
57870	3441.177120	192.168.1.108	172.67.208.98	ICMP	42	Echo (ping) request id=0x0000, seq=5888/23, ttl=63 (reply in 57871)
57871	3441.196845	172.67.208.98	192.168.1.108	ICMP	60	Echo (ping) reply id=0x0000, seq=5888/23, ttl=57 (request in 57870)
57872	3442.177761	192.168.1.108	172.67.208.98	ICMP	42	Echo (ping) request id=0x0000, seq=6144/24, ttl=63 (reply in 57873)
57873	3442.196544	172.67.208.98	192.168.1.108	ICMP	60	Echo (ping) reply id=0x0000, seq=6144/24, ttl=57 (request in 57872)

4.2.3 R.U.D.Y.

The container image for the third tool, R.U.D.Y., was created similarly to Saphyra. A RUDY python script was cloned from GitHub and added to the image directory. The problem encountered here was that the script needed the module “socks” to run and the python image from Docker Hub did not have it. To solve this problem the module “socks” was downloaded and added to the Docker file as a dependency. In the Wireshark capture it can be seen how RUDY opens multiple TCP connections trying to keep them open for a long time, but these are closed by the web server. (Figure 12)

Figure 30. Wireshark caption of the R.U.D.Y. attack



17351	1078.957661	192.168.1.108	140.82.121.3	TCP	66	54097 → 443 [ACK] Seq=2150 Ack=423178 Win=264192 Len=0 SLE=446154 SRE=519100
17352	1078.957861	192.168.1.108	140.82.121.3	TCP	66	54097 → 443 [ACK] Seq=2150 Ack=424614 Win=264192 Len=0 SLE=446154 SRE=519100
17353	1078.958036	192.168.1.108	140.82.121.3	TCP	66	54097 → 443 [ACK] Seq=2150 Ack=426050 Win=264192 Len=0 SLE=446154 SRE=519100
17354	1078.958090	192.168.1.108	140.82.121.3	TCP	66	54097 → 443 [ACK] Seq=2150 Ack=427486 Win=264192 Len=0 SLE=446154 SRE=519100
17355	1078.958177	192.168.1.108	140.82.121.3	TCP	66	54097 → 443 [ACK] Seq=2150 Ack=428922 Win=264192 Len=0 SLE=446154 SRE=519100
17356	1078.958203	192.168.1.108	140.82.121.3	TCP	66	54097 → 443 [ACK] Seq=2150 Ack=430358 Win=264192 Len=0 SLE=446154 SRE=519100
17357	1078.958233	192.168.1.108	140.82.121.3	TCP	66	54097 → 443 [ACK] Seq=2150 Ack=431794 Win=264192 Len=0 SLE=446154 SRE=519100
17358	1078.958259	192.168.1.108	140.82.121.3	TCP	66	54097 → 443 [ACK] Seq=2150 Ack=433230 Win=264192 Len=0 SLE=446154 SRE=519100
17359	1078.958300	192.168.1.108	140.82.121.3	TCP	66	54097 → 443 [ACK] Seq=2150 Ack=434666 Win=264192 Len=0 SLE=446154 SRE=519100
17360	1078.958512	140.82.121.3	192.168.1.108	TCP	1490	[TCP Out-Of-Order] 443 → 54097 [ACK] Seq=434666 Ack=2150 Win=70656 Len=1436
17361	1078.958512	140.82.121.3	192.168.1.108	TCP	1490	[TCP Out-Of-Order] 443 → 54097 [ACK] Seq=436102 Ack=2150 Win=70656 Len=1436
17362	1078.958512	140.82.121.3	192.168.1.108	TCP	1490	[TCP Out-Of-Order] 443 → 54097 [ACK] Seq=437538 Ack=2150 Win=70656 Len=1436
17363	1078.958512	140.82.121.3	192.168.1.108	TCP	1490	[TCP Out-Of-Order] 443 → 54097 [ACK] Seq=438974 Ack=2150 Win=70656 Len=1436
17364	1078.958512	140.82.121.3	192.168.1.108	TCP	1490	[TCP Out-Of-Order] 443 → 54097 [ACK] Seq=440410 Ack=2150 Win=70656 Len=1436
17365	1078.958512	140.82.121.3	192.168.1.108	TCP	1490	[TCP Out-Of-Order] 443 → 54097 [ACK] Seq=441846 Ack=2150 Win=70656 Len=1436
17366	1078.958512	140.82.121.3	192.168.1.108	TCP	1490	[TCP Out-Of-Order] 443 → 54097 [ACK] Seq=443282 Ack=2150 Win=70656 Len=1436
17367	1078.958512	140.82.121.3	192.168.1.108	TCP	1490	[TCP Out-Of-Order] 443 → 54097 [ACK] Seq=444718 Ack=2150 Win=70656 Len=1436
17368	1078.958662	192.168.1.108	140.82.121.3	TCP	66	54097 → 443 [ACK] Seq=2150 Ack=436102 Win=264192 Len=0 SLE=446154 SRE=519100
17369	1078.958752	192.168.1.108	140.82.121.3	TCP	66	54097 → 443 [ACK] Seq=2150 Ack=437538 Win=264192 Len=0 SLE=446154 SRE=519100
17370	1078.958907	192.168.1.108	140.82.121.3	TCP	66	54097 → 443 [ACK] Seq=2150 Ack=438974 Win=264192 Len=0 SLE=446154 SRE=519100
17371	1078.959062	192.168.1.108	140.82.121.3	TCP	66	54097 → 443 [ACK] Seq=2150 Ack=440410 Win=264192 Len=0 SLE=446154 SRE=519100
17372	1078.959259	192.168.1.108	140.82.121.3	TCP	66	54097 → 443 [ACK] Seq=2150 Ack=441846 Win=264192 Len=0 SLE=446154 SRE=519100
17373	1078.959433	192.168.1.108	140.82.121.3	TCP	66	54097 → 443 [ACK] Seq=2150 Ack=443282 Win=264192 Len=0 SLE=446154 SRE=519100
17374	1078.959487	192.168.1.108	140.82.121.3	TCP	66	54097 → 443 [ACK] Seq=2150 Ack=444718 Win=264192 Len=0 SLE=446154 SRE=519100
17375	1078.959671	192.168.1.108	140.82.121.3	TCP	54	54097 → 443 [ACK] Seq=2150 Ack=519100 Win=264192 Len=0

CHAPTER 5 – CLOUD DEPLOYMENT

5.1 Research

Cloud computing is the delivery of computing services and resources over the Internet to offer faster innovation, flexible resources, and economies of scale. Some of the services cloud computing delivers include servers, databases, storage, networking, software, or analytics. (What Is Cloud Computing? A Beginner's Guide | Microsoft Azure, 2022)

Cloud computing offers many benefits compared to on-premises IT infrastructure; these are some of the reasons organizations are shifting to cloud computing:

- Lower costs. The capital expenses of buying hardware and software, costs like electricity, security and IT experts that maintain on-site data centers are eliminated by cloud computing and typically replaced with pay-as-you-go pricing.
- Improved performance. Computing services data centers run on the latest and most efficient hardware which offer reduced network latency for applications.
- Enhanced Security. Cloud providers offer a broad set of policies and technologies that can strengthen the security of an organization overall.
- Agility at a global scale. Organizations can deploy applications in minutes by provisioning vast amounts of computing resources anywhere in the world.
- Reliability. Cloud computing makes data backup and disaster recovery easier by saving data in multiple redundant locations.

Depending on the needs of an organization, cloud computing can be of three types:

- Public cloud. This type of cloud is owned and operated by a third-party provider like Amazon Web Services (AWS), Microsoft Azure or Google Cloud Platform (GCP). Resources like servers and storage are delivered over the Internet and can be accessed and managed using a web browser or a Command Line Interface.
- Private cloud. This refers to the use of cloud computing resources on-premises in a private network. Organizations can have their own data centers, or they can pay a cloud provider to host their private cloud.
- Hybrid cloud. Hybrid cloud is a combination of Private and public clouds, linked by networking technologies that allow applications and services to be shared between them. A hybrid cloud gives more flexibility and helps optimize the existing infrastructure and security.

All cloud services tend to fall into 4 different categories sometimes called the cloud computing stack because they build on top of one another:

1 IaaS (infrastructure-as-a-service) is the most basic type of cloud service. Servers, storage, or networks can be rented on a pay–as–you-go basis. The Elastic Compute Cloud (EC2) instances used in this project fall in the IaaS category.

2 PaaS(platform-as-a-service) refers to services that supply on-demand environments for developing and deploying software applications without worrying about the underlying infrastructure. One example of a PaaS service is the AWS Elastic Container Service (EC2 launch type) that was used in this project.

3 Serverless computing is an execution model in which the cloud provider handles all the setup, capacity planning and server management so the user will only need to focus on app functionality. Serverless architectures are highly scalable, only using resources when an event or function is triggered. AWS Fargate is a serverless compute engine that was configured to launch DDoS attacks in this project.

4 SaaS(software-as-a-service) is a delivery model in which software is fully hosted and managed by the cloud provider and typically offered to users on a subscription model. Some examples of SaaS include mail services Gmail, Customer Relationship Management (CRM) applications like SalesForce or document management software like DocuWare. (What Is Cloud Computing? A Beginner's Guide | Microsoft Azure, 2022)

5.2 Adding the Docker image to AWS Elastic Container Registry (ECR)

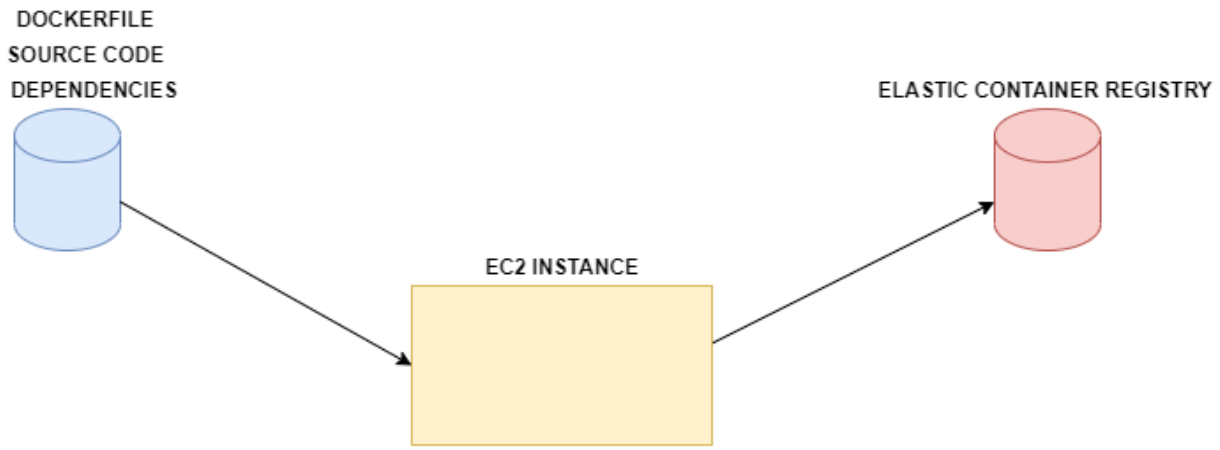
5.2.1 Research

AWS ECR is a fully managed container repository from where applications can be quickly deployed into a production environment. AWS provides a command line interface, the AWS CLI, to push or pull container images to or from an AWS region. ECR can be used wherever a container is running and is supported by other AWS services like Elastic Container Services (ECS), Fargate or Elastic Beanstalk.

AWS ECR is highly available, very secure and completely managed by AWS so the user will not need to install any extra software. (Carty, 2022)

As Figure 13 shows, the process of adding a container image to AWS ECR involves creating an EC2 instance, transferring the code and all necessary dependencies from the local computer or from other sources to the EC2 instance, and pushing the container image to ECR.

Figure 31. Process of adding a container image to AWS ECR



Amazon Elastic Compute Cloud (EC2) instances are virtual machines used to run applications in the AWS infrastructure. AWS provides different types of configurations of memory, CPU, storage, and networking to suit various workload requirements.

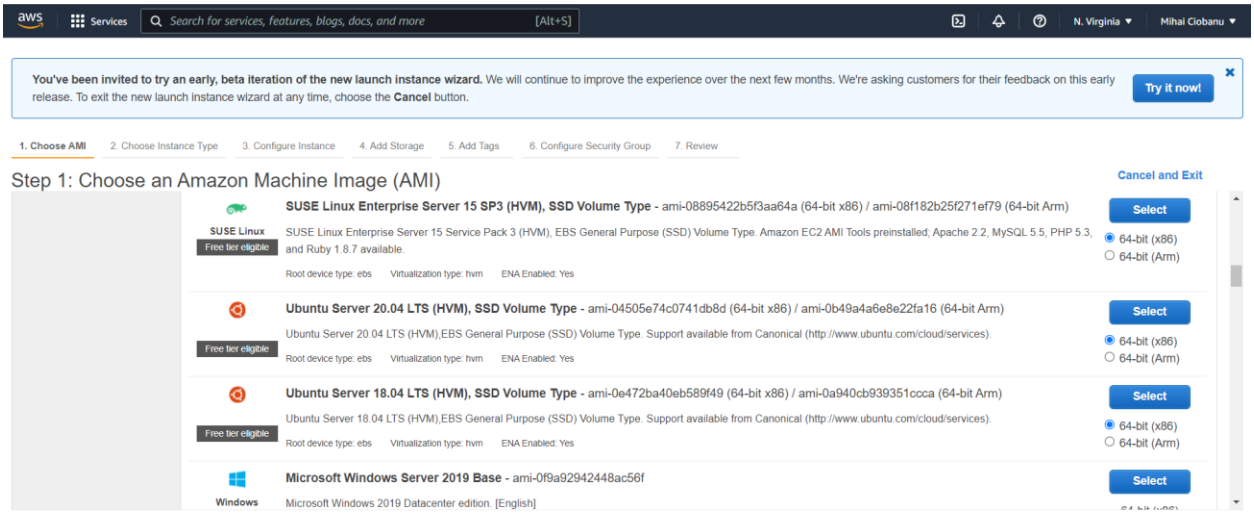
Users can build EC2 instances using preconfigured images that AWS provides, they can create their own images, or they can purchase images from AWS Marketplace. These preconfigured images are called Amazon Machine Images (AMI) and they include an operating system that can be different Linux distributions or Windows and other software.

AMI's are split into different categories based on the target application profiles. These include General Purpose, Compute Optimized, Memory Optimized, Graphics optimized and Micro. (Wigmore, 2022)

5.2.2 The development process

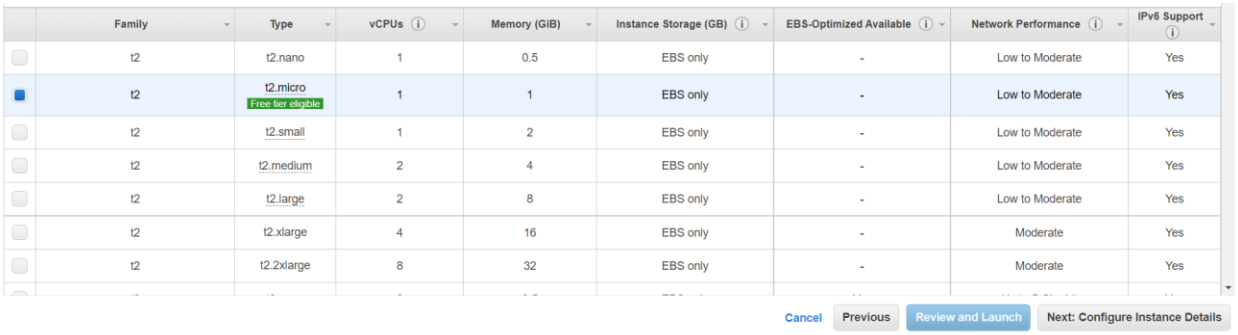
The Amazon Machine Image selected for the EC2 instance was the Ubuntu 20.04 as it already has python installed and is a versatile operating system. (Figure 14)

Figure 32. Amazon Machine Images



Amazon provides a variety of instance types with different CPU, memory, and networking configurations. The type of instance selected for this experiment was "t2 micro" which has 1 virtual CPU and 1 GB memory and is free to use in a free-tier account. (Figure 15)

Figure 33. EC2 instance types



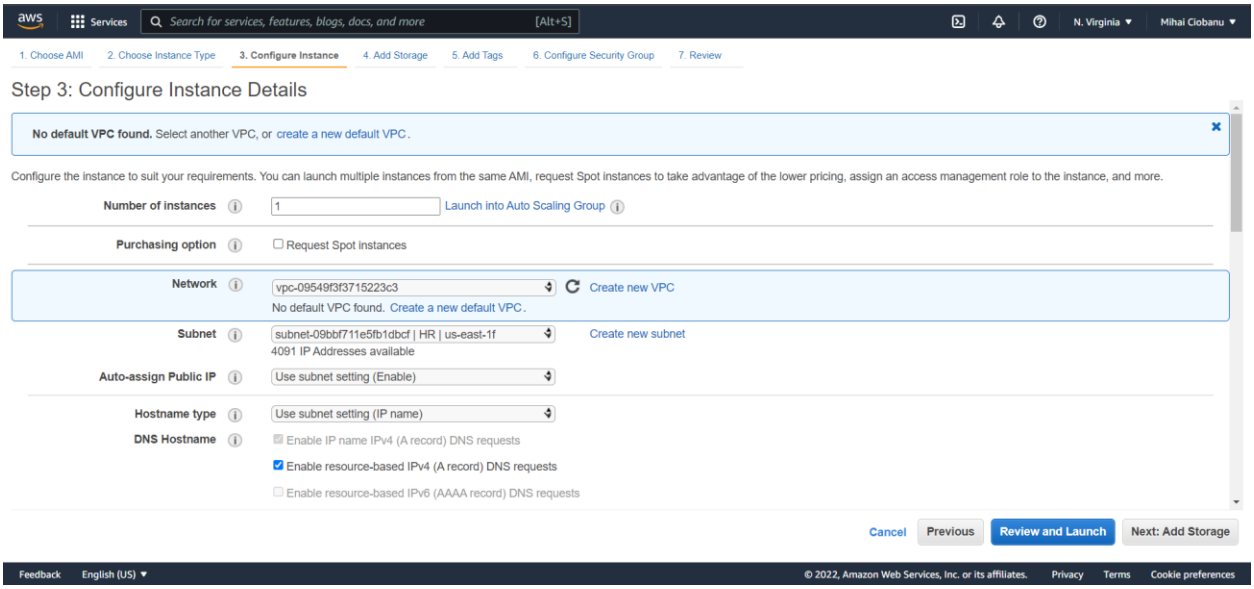
	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
☐	t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
☑	t2	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
☐	t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes
☐	t2	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
☐	t2	t2.large	2	8	EBS only	-	Low to Moderate	Yes
☐	t2	t2.xlarge	4	16	EBS only	-	Moderate	Yes
☐	t2	t2.2xlarge	8	32	EBS only	-	Moderate	Yes

Step 3 required the configuration of the networking part so the EC2 instance was added to a Virtual Private Cloud (VPC) and a subnet.

A VPC is similar to a traditional network that connects the user to the cloud and enables the launch of resources. The VPC is a virtual network dedicated to a user's account and it is logically isolated from other virtual networks in the AWS cloud. The VPC can be associated with an IP address range and subnets can be added. The VPC can be secured by adding it to a security group and creating Network Access Control lists (ACL). (How Amazon VPC works, 2022)

Only one EC2 instance was required so the default VPC and default subnet were selected. (Figure 16)

Figure 34. EC2 networking configuration



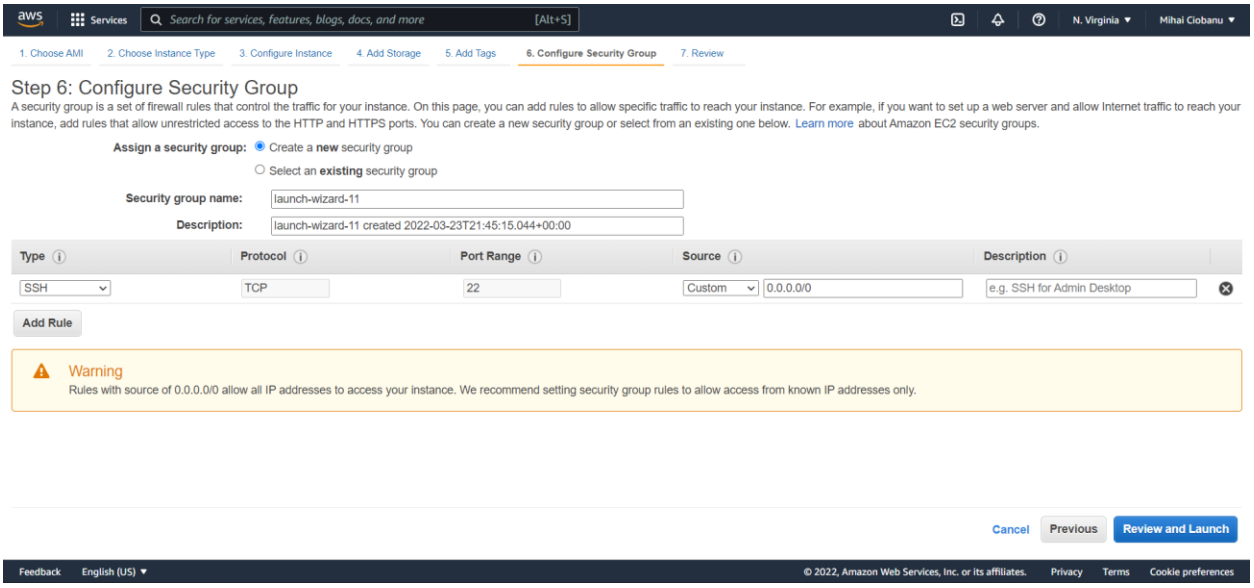
In the next step 20GiB of SSD storage were added to the instance Hard Drive. Step 6 allows the user to add tags however this was not necessary as only one instance was launched. Finally, in step 7 a security group has been assigned to the EC2 instance.

A security group acts like a virtual firewall, controlling the incoming and outgoing traffic. When first creating a security group, it has no inbound rules, all traffic is denied. To permit inbound traffic,

allow rules can be created. These rules must include the protocol, port range and source or destination. (Control traffic to resources using security groups, 2022)

For this EC2 instance no inbound traffic was allowed except for SSH which was needed to connect from the local computer. (Figure 17)

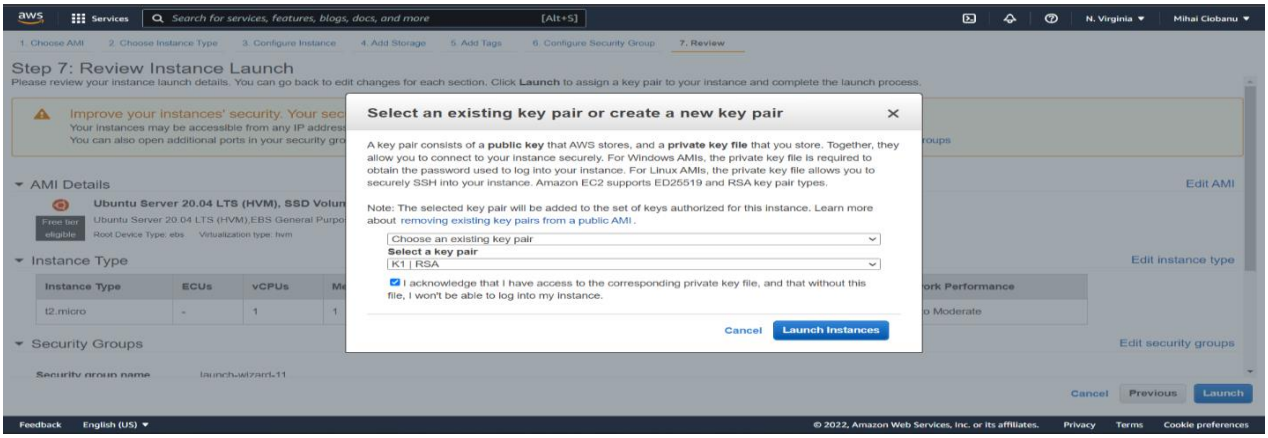
Figure 17. EC2 Security Group configuration



The configuration of a key pair was required before launching the instance. The key pair consists of a public key and a private key. The public key is stored on the EC2 instance, and the private key is downloaded and kept by the user. This key can be used to SSH to the EC2 instance and needs to be kept in a very secure place as anyone who has this key can SSH to the instance.

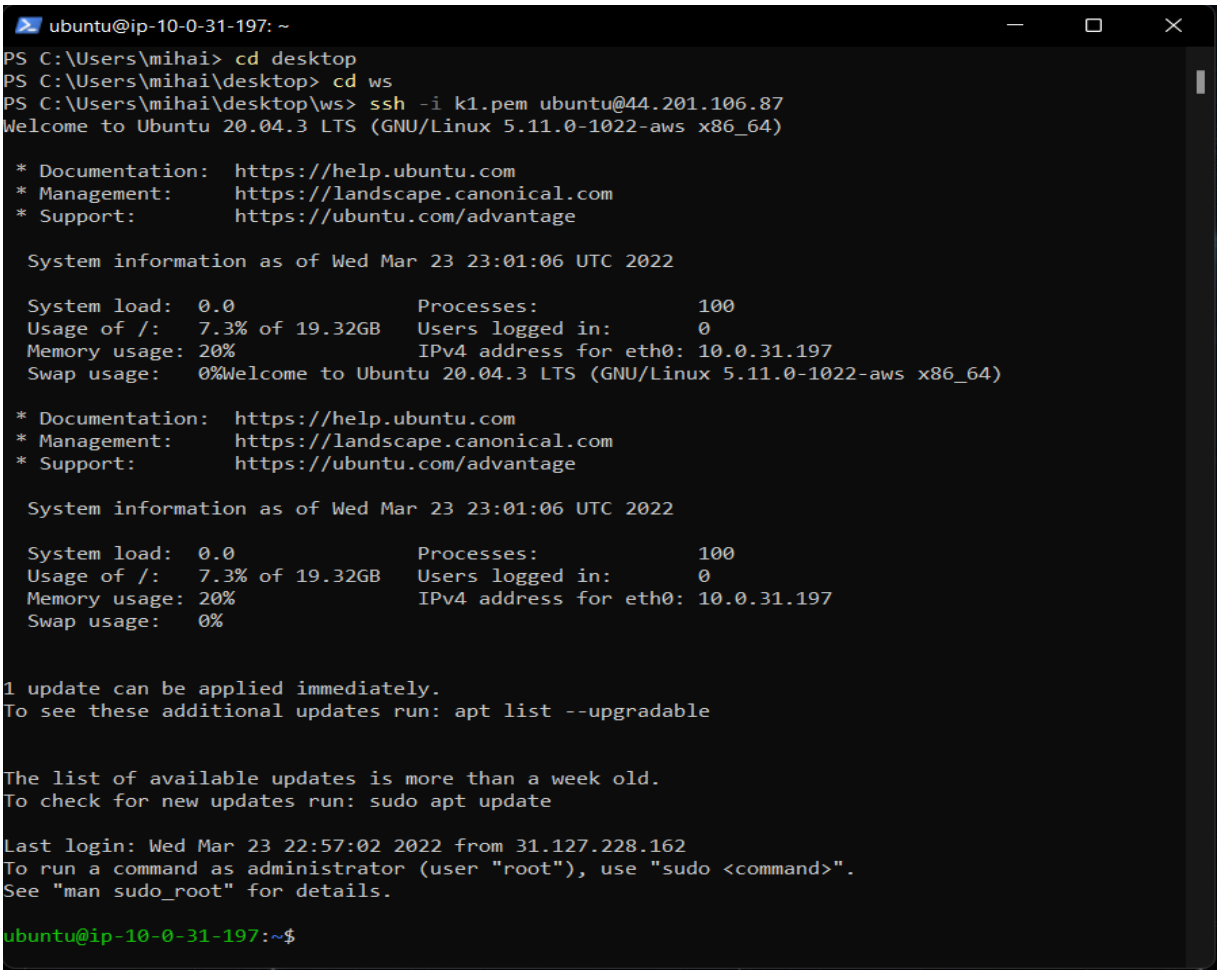
The keys Amazon EC2 uses are ED25519 or 2048-bit SSH-2 RSA keys. (Figure 18) (Amazon EC2 key pairs and Linux instances, 2022)

Figure 18. EC2 key pair association



The instance was successfully launched, and the key pair was used to initiate a SSH connection from a local Windows PowerShell. (Figure 19)

Figure 19. SSH connection to the EC2 instance



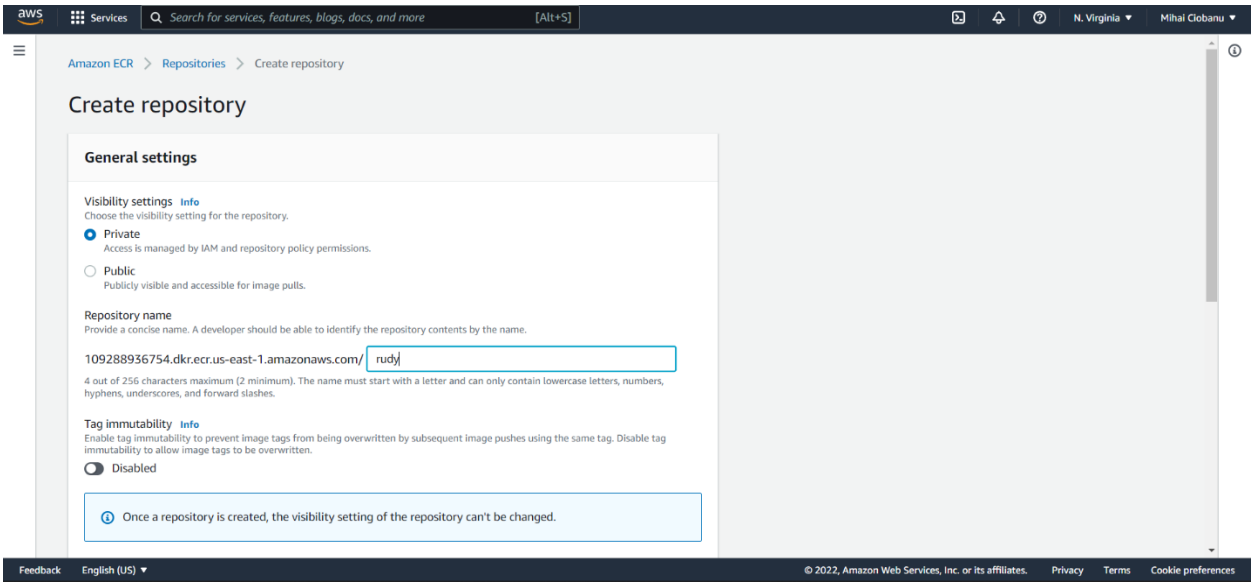
After launching the instance, Docker engine was installed, the files of the three DDoS tools were transferred to the instance and then built into container images, the same as it was done on the local Linux machine in Chapter 4. The Docker images were now ready to be pushed to AWS ECR.

AWS ECR works in the following way:

- The code is packaged in the form of a container
- The container image is compressed and encrypted at rest and then managed throughout its lifecycle
- The image is transferred to other services over HTTPS by checking their security credentials

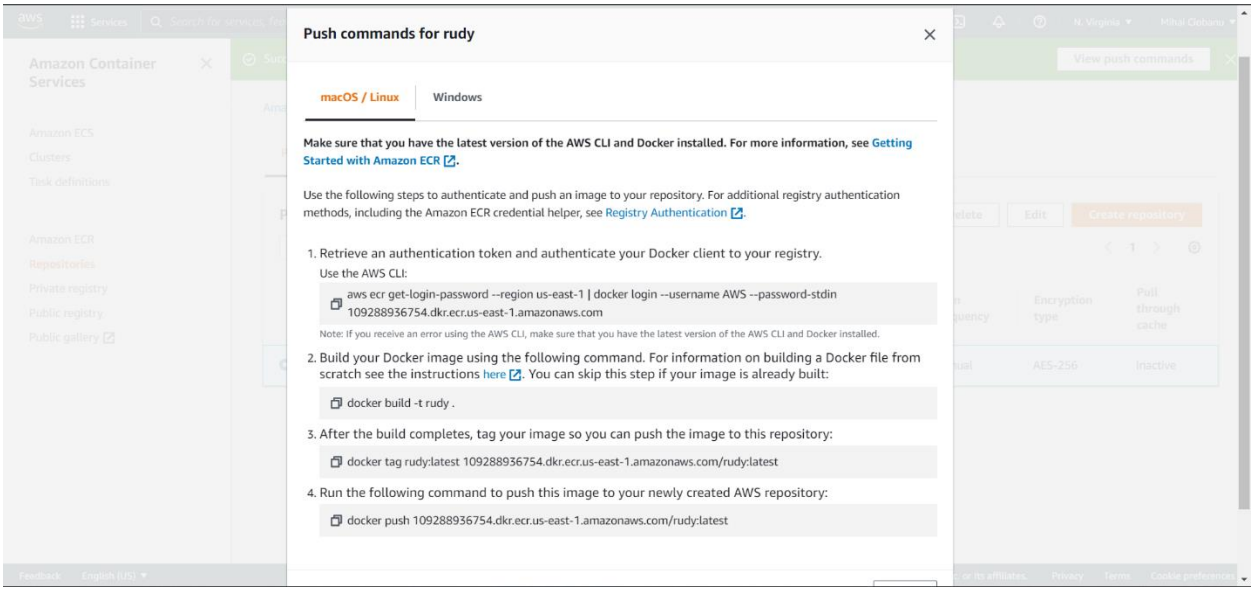
The first repository that was created was the DDoS tool RUDY. (Figure 20)

Figure 20. R.U.D.Y repository configuration



To push the image into ECR, a set of custom commands found in the section “View push commands” need to be executed from the EC2 command line. (Figure 21)

Figure 21. Push commands for R.U.D.Y. image



The first push command requires the EC2 instance to authenticate to ECR. This command returned an error at first because the AWS CLI was not installed. (Figure 22)

The AWS CLI is a tool developed by AWS that enables Linux users to manage AWS cloud services from a command line interface. AWS CLI is one of several methods users can manage AWS services, other methods include the AWS Management Console and the AWS application programming interfaces. AWS also offers tools for Windows users who script with Windows PowerShell.

Figure 22. "Command 'aws' not found" error

```
ubuntu@ip-10-0-31-197:~/RUDY$ aws ecr get-login-password --region us-east-1 | docker login --user
name AWS --password-stdin 109288936754.dkr.ecr.us-east-1.amazonaws.com

Command 'aws' not found, but can be installed with:

sudo apt install awscli

Error: Cannot perform an interactive login from a non TTY device
ubuntu@ip-10-0-31-197:~/RUDY$
```

The AWS CLI was installed, and the push command was executed again, however this prompted another error stating the EC2 instance did not have valid credentials. (Figure 23)

Figure 2335. "Unable to locate credentials" error

```
ubuntu@ip-10-0-31-197:~/RUDY$ aws ecr get-login-password --region us-east-1 | docker login --user
name AWS --password-stdin 109288936754.dkr.ecr.us-east-1.amazonaws.com
Unable to locate credentials. You can configure credentials by running "aws configure".
Error: Cannot perform an interactive login from a non TTY device
ubuntu@ip-10-0-31-197:~/RUDY$
```

Troubleshooting the issue, it was found that to enable an EC2 instance to use another AWS service this instance needs to have security access specific for this service. To achieve this AWS recommends defining an Identity and Access Management (IAM) role that has access to that service and associating the EC2 instance with the IAM role.

An IAM role is an identity management tool with permission policies that determine which AWS resources a user can and cannot use. A role is intended to be assumable by anyone who needs it and does not have standard credentials such as access keys or password. A role can provide temporary security credentials for users or AWS services. (IAM roles, 2022)

The IAM role “AmazonEC2ContainerRegistryFullAccess” was assigned to the EC2 instance which resulted in a successful login. (Figure 24)

Figure 24. ECR successful login

```
ubuntu@ip-10-0-31-197:~/RUDY$ aws ecr get-login-password --region us-east-1 | docker login --user
name AWS --password-stdin 109288936754.dkr.ecr.us-east-1.amazonaws.com
Unable to locate credentials. You can configure credentials by running "aws configure".
Error: Cannot perform an interactive login from a non TTY device
ubuntu@ip-10-0-31-197:~/RUDY$ aws ecr get-login-password --region us-east-1 | docker login --user
name AWS --password-stdin 109288936754.dkr.ecr.us-east-1.amazonaws.com
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/dock
er.sock: Post http://%2Fvar%2Frun%2Fdocker.sock/v1.24/auth: dial unix /var/run/docker.sock: conne
ct: permission denied
ubuntu@ip-10-0-31-197:~/RUDY$ sudo su
root@ip-10-0-31-197:/home/ubuntu/RUDY# aws ecr get-login-password --region us-east-1 | docker log
in --username AWS --password-stdin 109288936754.dkr.ecr.us-east-1.amazonaws.com
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@ip-10-0-31-197:/home/ubuntu/RUDY#
```

Next, the second command builds the docker image. The image is then tagged and pushed to Elastic Container Registry. (Figure 25)

Figure 25. Docker image pushed to ECR

```
root@ip-10-0-31-197:/home/ubuntu/RUDY# docker tag rudy:latest 109288936754.dkr.ecr.us-east-1.amaz
onaws.com/rudy:latest
root@ip-10-0-31-197:/home/ubuntu/RUDY# docker push 109288936754.dkr.ecr.us-east-1.amazonaws.com/r
udy:latest
The push refers to repository [109288936754.dkr.ecr.us-east-1.amazonaws.com/rudy]
a02f1d2f4963: Pushed
0d53c252f535: Pushed
18b36ab1593d: Pushed
```

The same steps have been taken to push the other two DDoS tool container images to AWS ECR.

5.3 Amazon Elastic Container Service (ECS)

5.3.1 Research

AWS ECS is the service offered by Amazon to manage and run containers in its cloud infrastructure. ECS allows users to create logical groupings of servers called clusters. The applications are deployed by pulling the necessary container images from the Elastic Container Registry or Docker Hub. There are two launch types that can be used to deploy applications: EC2 launch type and the serverless compute engine Fargate.

In EC2 launch type the user explicitly provisions and manages the EC2 instances that run the containers. The user needs to take into consideration the computing power needed, the number of instances and the cluster optimization which may include scaling and load balancing. This type offers a more granular control over the infrastructure but also more operational overhead. (Carty, 2022)

Fargate is a serverless compute engine that can be used to launch applications without worrying about the underlying infrastructure. Fargate automatically provisions the required resources, manages the computing capacity, and handles cluster optimization. These are some obvious benefits however there is a more limited control compared to EC2 launch type. (AWS Fargate–Amazon Web Services, 2022)

To gain a better understanding of the benefits the two launch types offer, and which one is more suitable for the DDoS simulator, both launch types have been deployed and then compared running the container image R.U.D.Y.

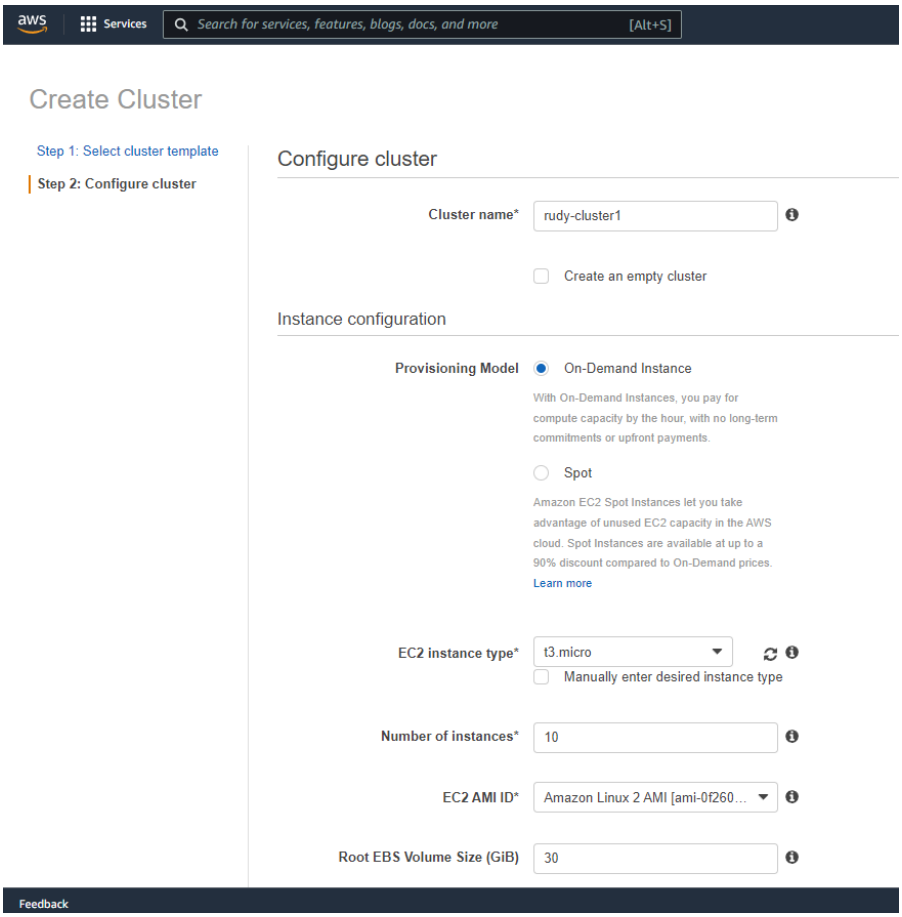
5.3.2 The development process

5.3.2.1 EC2 launch type

EC2 launch type configuration starts by creating a cluster. The cluster settings include:

- EC2 instance type (RUDY was tested on a local computer and it was found it only uses small CPU and RAM resources therefore the free “t2.micro” instance type was enough to handle the workload)
- Number of instances (10 is maximum allowed for a cluster) (Figure 26)

Figure 26. EC2 cluster basic configuration

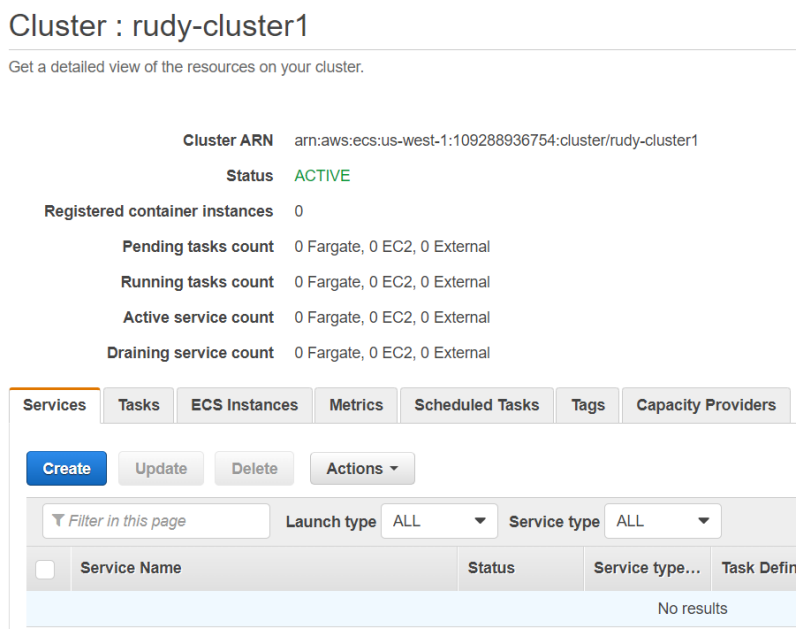


In the networking part the cluster was assigned to a VPC and a subnet and a new security group that permits inbound and outbound HTTP and HTTPS traffic was created.

The cluster dashboard consists of several tabs used to manage the applications running on the cluster:

- "Services" is used to scale the number of tasks
- "Tasks" are basically the containers running the application, as defined in Task Definition
- "ECS instances" are the EC2 instances registered in the cluster
- "Metrics" shows the CPU and memory utilisation
- "Scheduled tasks"
- "Tags"
- "Capacity providers" are an extension to auto-scaling groups used for preferential auto-scaling. (Figure 27)

Figure 27. EC2 cluster dashboard

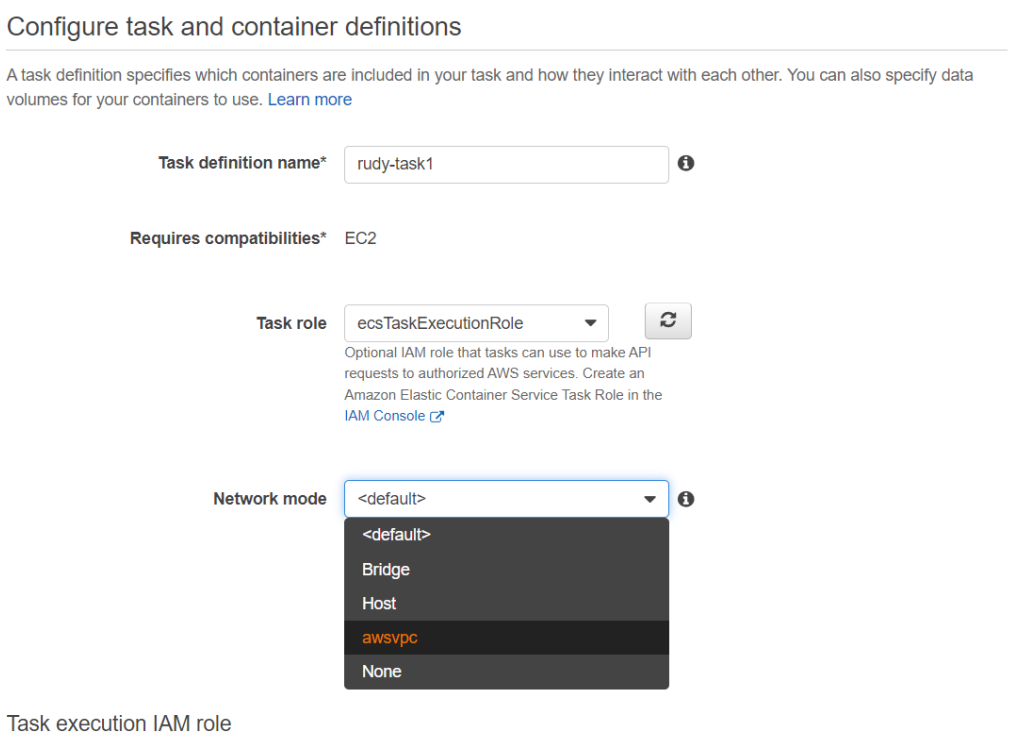


A task definition must be created to launch an application in the cluster. The task definition is the blueprint that describes one or more containers that form the application and defines the parameters the containers will use. (Task definitions, 2022)

The task definition settings include:

- Compatibility-EC2 or Fargate
- Task role – the task role needed by the application to access other AWS resources. The automatically created Task Execution Role was selected.
- Network Mode – There are four types: Bridge, awsvpc, Host or None.
  In “Bridge Mode” the container port is mapped to the host port explicitly. If more than one container is running the same program on the same host only one container will be able to run. For this reason, this was not a suitable option.
  Similar to bridge mode, in “Host mode” only one container is mentioned, and the corresponding host port is automatically mapped to the container port.
  “None mode” means the container will not have any outside connectivity.
  “Awsvpc” uses Dynamic Port Mapping which maps the same container port with different host ports. Multiple containers with the same image can be used on the same host which makes it the most suitable choice for the DDoS simulator. (Figure 28) (Dey, 2022)

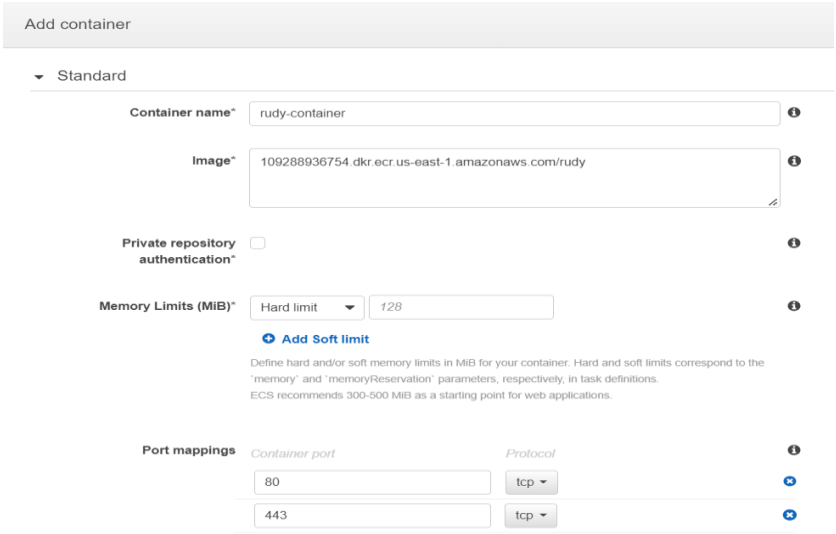
Figure 28. Task definition configuration



Next the automatically created task execution role was selected and the task size was added: 512 MB RAM and 1 virtual CPU.

The container image was added by copying and pasting the image Uniform Resource Identifier from ECR and the port mappings 80-HTTP and 443-HTTPS were selected which means the containers used these two ports to map the traffic between container and the EC2 instance. (Figure 30)

Figure 29. Container added to task definition



Two services each running 10 tasks have been launched from the cluster dashboard. Figure 30 shows the 20 containers successfully running.

Figure 30. EC2 cluster containers

Filter in this page | Launch type: ALL

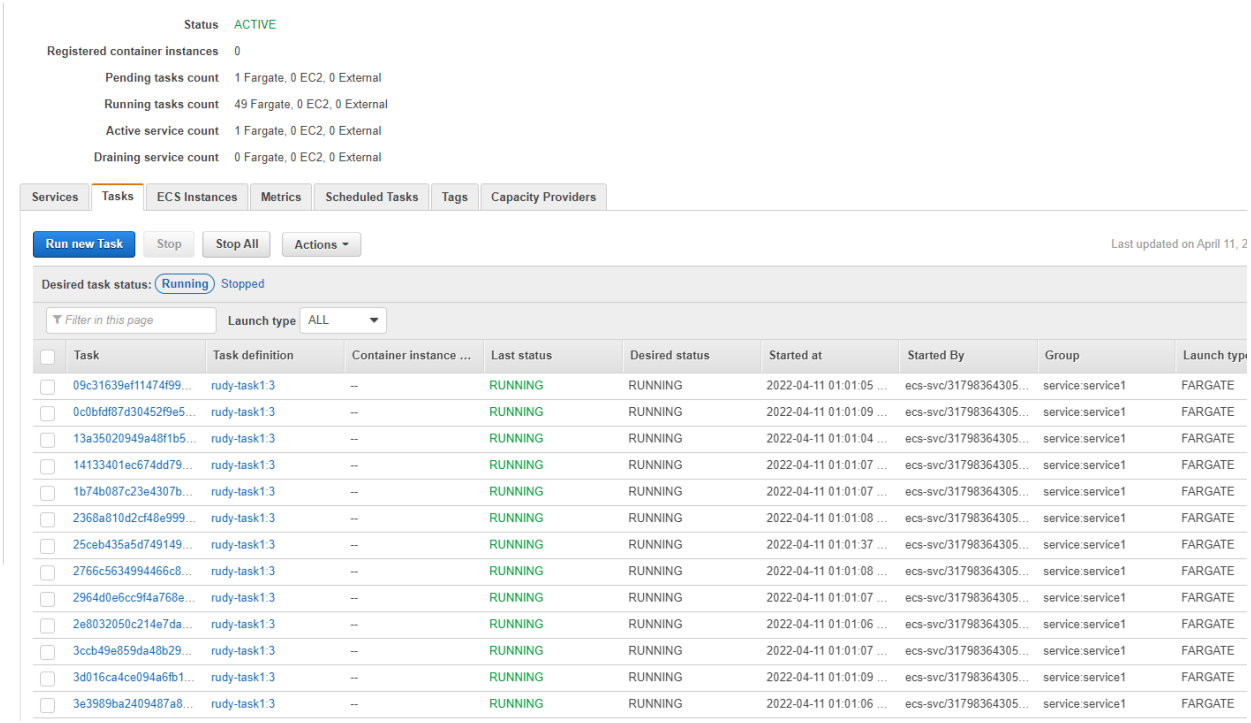
Task	Task definition …	Container insta…	Last status	Desired status …	Started at	Started By
0a48b2b296a04…	rudy-task1:3	--	RUNNING	RUNNING	2022-03-25 20:0…	ecs-svc/231654…
0eea1b6d45944…	rudy-task1:3	--	RUNNING	RUNNING	2022-03-25 20:0…	ecs-svc/231654…
16927092d98a4…	rudy-task1:3	--	RUNNING	RUNNING	2022-03-25 20:0…	ecs-svc/231654…
3aa4742852f04…	rudy-task1:3	--	RUNNING	RUNNING	2022-03-25 20:0…	ecs-svc/231654…
51abb4f80cd14…	rudy-task1:3	--	RUNNING	RUNNING	2022-03-25 20:0…	ecs-svc/231654…
5b9d9e8a75b64…	rudy-task1:3	--	RUNNING	RUNNING	2022-03-25 20:0…	ecs-svc/231654…
71fb2c5f519b42…	rudy-task1:3	--	RUNNING	RUNNING	2022-03-25 20:0…	ecs-svc/231654…
767cd79a77b24…	rudy-task1:3	--	RUNNING	RUNNING	2022-03-25 20:0…	ecs-svc/231654…
76997d11f32e4c…	rudy-task1:3	--	RUNNING	RUNNING	2022-03-25 20:0…	ecs-svc/231654…
77f0f902ea7c44…	rudy-task1:3	--	RUNNING	RUNNING	2022-03-25 20:0…	ecs-svc/231654…
87cd2c1382524…	rudy-task1:3	--	RUNNING	RUNNING	2022-03-25 20:0…	ecs-svc/231654…
88bffae745554b…	rudy-task1:3	--	RUNNING	RUNNING	2022-03-25 20:0…	ecs-svc/231654…
ad856abc51134…	rudy-task1:3	--	RUNNING	RUNNING	2022-03-25 20:0…	ecs-svc/231654…
b3a2cedaba084…	rudy-task1:3	--	RUNNING	RUNNING	2022-03-25 20:0…	ecs-svc/231654…
cfce0d62894441…	rudy-task1:3	--	RUNNING	RUNNING	2022-03-25 20:0…	ecs-svc/231654…
d0bc740afb4c46…	rudy-task1:3	--	RUNNING	RUNNING	2022-03-25 20:0…	ecs-svc/231654…
d502479c1f724…	rudy-task1:3	--	RUNNING	RUNNING	2022-03-25 20:0…	ecs-svc/231654…
dd0d6801c57f4…	rudy-task1:3	--	RUNNING	RUNNING	2022-03-25 20:0…	ecs-svc/231654…
e2d48188f0054…	rudy-task1:3	--	RUNNING	RUNNING	2022-03-25 20:0…	ecs-svc/231654…
f8ccb24a3fde4a…	rudy-task1:3	--	RUNNING	RUNNING	2022-03-25 20:0…	ecs-svc/231654…

5.3.2.2 Fargate launch type

Because it is a serverless platform, configuring Fargate was a less difficult process compared with the EC2 launch type. Another advantage is that the services are not limited at running only 10 tasks, up to 300 containers can be run from only one service and Fargate will scale automatically. This leads to much less overhead compared with EC2 launch type.

As Figure 31 shows, a number of 50 containers running the R.U.D.Y. tool have been successfully deployed within a single Fargate service.

Figure 361. Fargate cluster containers



Status	ACTIVE
Registered container instances	0
Pending tasks count	1 Fargate, 0 EC2, 0 External
Running tasks count	49 Fargate, 0 EC2, 0 External
Active service count	1 Fargate, 0 EC2, 0 External
Draining service count	0 Fargate, 0 EC2, 0 External

Services | Tasks | ECS Instances | Metrics | Scheduled Tasks | Tags | Capacity Providers

Run new Task | Stop | Stop All | Actions | Last updated on April 11, 2

Desired task status: Running Stopped

Filter in this page | Launch type ALL

Task	Task definition	Container instance ...	Last status	Desired status	Started at	Started By	Group	Launch type
09c31639ef11474f99...	rudy-task1:3	--	RUNNING	RUNNING	2022-04-11 01:01:05 ...	ecs-svc/31798364305...	service:service1	FARGATE
0c0bfdf87d30452f9e5...	rudy-task1:3	--	RUNNING	RUNNING	2022-04-11 01:01:09 ...	ecs-svc/31798364305...	service:service1	FARGATE
13a35020949a48f1b5...	rudy-task1:3	--	RUNNING	RUNNING	2022-04-11 01:01:04 ...	ecs-svc/31798364305...	service:service1	FARGATE
14133401ec674dd79...	rudy-task1:3	--	RUNNING	RUNNING	2022-04-11 01:01:07 ...	ecs-svc/31798364305...	service:service1	FARGATE
1b74b087c23e4307b...	rudy-task1:3	--	RUNNING	RUNNING	2022-04-11 01:01:07 ...	ecs-svc/31798364305...	service:service1	FARGATE
2368a810d2cf48e999...	rudy-task1:3	--	RUNNING	RUNNING	2022-04-11 01:01:08 ...	ecs-svc/31798364305...	service:service1	FARGATE
25ceb435a5d749149...	rudy-task1:3	--	RUNNING	RUNNING	2022-04-11 01:01:37 ...	ecs-svc/31798364305...	service:service1	FARGATE
2766c5634994466c8...	rudy-task1:3	--	RUNNING	RUNNING	2022-04-11 01:01:08 ...	ecs-svc/31798364305...	service:service1	FARGATE
2964d0e6cc9f4a768e...	rudy-task1:3	--	RUNNING	RUNNING	2022-04-11 01:01:07 ...	ecs-svc/31798364305...	service:service1	FARGATE
2e8032050c214e7da...	rudy-task1:3	--	RUNNING	RUNNING	2022-04-11 01:01:06 ...	ecs-svc/31798364305...	service:service1	FARGATE
3ccb49e859da48b29...	rudy-task1:3	--	RUNNING	RUNNING	2022-04-11 01:01:07 ...	ecs-svc/31798364305...	service:service1	FARGATE
3d016ca4ce094a6fb1...	rudy-task1:3	--	RUNNING	RUNNING	2022-04-11 01:01:09 ...	ecs-svc/31798364305...	service:service1	FARGATE
3e3989ba2409487a8...	rudy-task1:3	--	RUNNING	RUNNING	2022-04-11 01:01:06 ...	ecs-svc/31798364305...	service:service1	FARGATE

CHAPTER 6 – TESTING THE DDOS SIMULATOR

Different types of DDoS attacks were launched on the two websites to test the efficiency of the simulator. To ensure the legality of the attacks, the DDoS testing terms and conditions of the two hosting providers, Cloudflare and GoDaddy, have been reviewed, and testing has been approved by both providers (Appendices 14 and 15). A permission has also been obtained from AWS to use their cloud resources to build the simulator and launch the DDoS attacks. (Appendix 13)

Applying the Agile-Scrum methodology, tests were carried out at every stage while implementing the DDoS simulator. Testing was done incremental, and the development of the next stage would not begin until all prior stage tests had passed. This type of testing presents several benefits like reduced time, increased flexibility and highly adaptable to changes.

AWS offers a service called Cloud Watch to monitor and analyse data from applications. This service collects operational data in the form of logs, metrics and events offering a unified view of all the resources, applications and services running in AWS. (Amazon CloudWatch - Application and Infrastructure Monitoring, 2022) The container and task logs collected by Cloud Watch during testing can be viewed in Appendices 1-12.

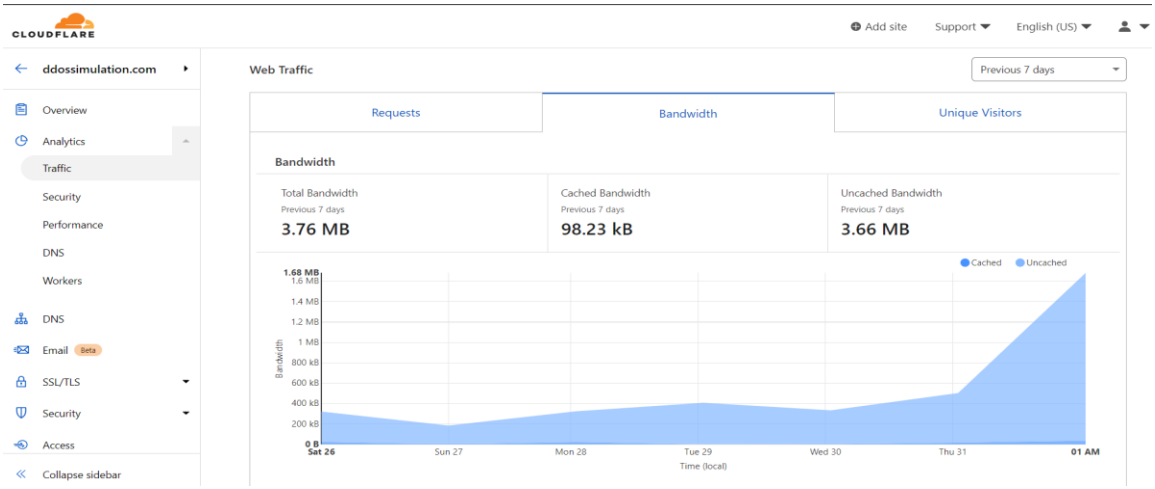
Two testing plans have been developed to test the two websites against all three types of DDoS attacks: Protocol-based, Application Layer-based, and Volumetric. The duration of each test was set to 10 minutes and the number of containers deployed in the DDoS simulator cluster was 50 for the first test plan and 500 for the second test plan. Unfortunately, AWS only allow free-tier customers to deploy a maximum of 50 containers per account. A request for a limit increase was submitted to AWS however by the time this was approved the $300 free credits offered by AWS when opening a free-tier account had already been spent by testing each DDoS tool in 50 container clusters plus the testing done during the development process. The web server response was analysed by checking the page loading time and ping request reply time. As seen in Table 1, the impact of the DDoS simulator on the web servers was non-existent.

Table 1. Web server response times

Tool/Script	No. of containers	Duration	Ping delay(milliseconds)				Loading time(seconds)			
			Cloudflare	GoDaddy	No simulation		Cloudflare	GoDaddy	No simulation	
					CF	GD			CF	GD
Hping3-ICMP	50	10 minutes	17	19	17	19	0	0	0	0
Hping3-TCP SYN	50	10 minutes	17	19	17	19	0	0	0	0
Hping3-UDP Flood	50	10 minutes	17	19	17	19	0	0	0	0
RUDY-low and slow	50	10 minutes	17	19	17	19	0	0	0	0
Saphyra-HTTP flood	50	10 minutes	17	19	17	19	0	0	0	0

To further test the efficiency of the DDOS simulator, the traffic received by the Cloudflare website was examined in the analytics page found in the Cloudflare account. As it can be seen in the Figure 32, there was a small spike in bandwidth usage on 31.03.2022, the day the attacks were launched, however this was very limited, only 1.2MB more than normal usage.

Figure 372. Cloudflare website traffic analysis



CHAPTER 7 – ANALYSIS OF DDOS MITIGATION TECHNIQUES EMPLOYED BY CLOUDFLARE AND AWS

7.1 Cloudflare DDoS defence techniques

The servers and data centres of Cloudflare span globally to form a network of 121 Terabytes per second making it 15 times greater than the largest DDoS attack ever recorded. (Cloudflare on IBM Cloud, 2022)

Cloudflare is a Content Delivery Network provider that acts as a reverse proxy to protect their customers' online activities. A reverse proxy is a server that sits in front of web servers and forwards client requests to those web servers. The difference between a reverse proxy and a normal proxy is that the normal proxy sits in front of the client computers and forwards client requests to a web server or other services on the Internet while the reverse proxy sits in front of a web server and forwards the requests to the web server.

The reverse proxy can offer DDOS protection in many ways:

- Load balancing. Anycast is a networking addressing and routing method in which the traffic is evenly distributed between a pool of web servers to improve the performance of websites or other computing resources and prevent any single server being overloaded. This networking method takes advantage of Cloudflare's massive resources and allows their clients to be resilient in the face of high traffic volumes such as DDoS attacks by scaling and absorbing huge amounts of traffic. (What is Anycast? | How does Anycast work?, 2022)
- Hiding the IP address. A website never reveals the IP address of the origin server with a reverse proxy in place. This makes it much harder for an attacker to leverage a DDoS attack as he/she will only be able to target the Cloudflare network which has tighter security and more resources to mitigate a DDOS attack.
- Caching content. The reverse proxy also caches content which results in faster performance and less demand for the web servers. (What is a reverse proxy? | Proxy servers explained, 2022)
- Web Application Firewall. This is a tool that can be used to protect against Layer 7 DDoS attacks by filtering HTTP requests based on a series of managed rules. These rules can match known attack tools, suspicious patterns, protocol violations, requests causing large amounts of origin errors, excessive traffic, and other attack vectors. To quickly respond to an attack, custom rules can be implemented to the firewall. (Understanding Cloudflare DDoS protection, 2022)
- Machine Learning. The Web Application Firewall can sometimes miss attacks due to pattern variations in managed rules. To protect against such attacks, Cloudflare uses machine learning with the purpose of identifying anomalies and variations of managed rules before they are exploited or identified by human researchers. Once an anomaly has been detected, the user will be informed, and the Web Application Firewall can be updated with a new filter. (Molteni, 2022)

7.2 AWS DDoS defence techniques

The standard DDoS protection service offered by AWS is called AWS Shield Standard. This service provides always-on detection and defends against most common, frequently occurring DDoS attacks. To receive a more comprehensive DDoS protection, AWS customers can use AWS Shield Standard with other services like AWS CloudFront-a Content Delivery Network with over 310 globally

dispersed Points of Presence and Amazon Route 53-a highly available and scalable cloud Domain Name System service. (AWS Shield - Amazon Web Services (AWS), 2022)

Similar to Cloudflare, AWS employs DDoS defence techniques like Web application Firewall, machine learning, load balancing and caching.

7.3 Defence techniques against the tools/scripts used in the project.

7.3.1 Hping3 ICMP attack

Cloudflare mitigates this type of attack by standing between the attacker and the server and placing all the traffic on their networks. A network administrator can mitigate this type of attack by simply disabling the ICMP functionality of the targeted router, however this is not the best option as legitimate ICMP traffic will also be disabled. (How is a Ping flood attack mitigated?, 2022)

7.3.2 Hping3 TCP SYN Flood

One response to TCP SYN Flood attacks is to increase the maximum number of possible connections the operating system will allow. Depending on the size of the attack, the system performance might be negatively impacted but that may still be better than the server crashing.

Another technique is to overwrite the oldest half-open connections once the backlog has been filled. This allows legitimate connections to be established, however it can fail when the attack volume is increased, or the backlog size is too small to be practical.

The creation of SYN cookies is another strategy. The web server responds to each connection with a SYN-ACK, creates a cookie and then drops the connection leaving the port open. If the request is legitimate and the server receives an ACK message back, it will reconstruct the request from the cookie and rebuild the connection.

Cloudflare mitigates TCP SYN Flood attacks by handling the handshake process in the cloud, and only sending the connection to the targeted server when the handshake is complete. (SYN flood attack, 2022)

7.3.3 UDP Flood

To mitigate this attack, Cloudflare drops any UDP traffic not related to DNS at the network edge, and leverages Anycast to load balance the other traffic across the network. (UDP flood attack, 2022)

7.3.4 R.U.D.Y.

Low-and-slow attacks are harder to detect than volumetric attacks. One measure that can protect against such attacks is setting stricter timeout intervals on a web server. The downside of this measure is denying service to legitimate users with slow connections. A reverse proxy like Cloudflare will filter low-and-slow connections before reaching the server without disconnecting legitimate users. (R U Dead Yet? (R.U.D.Y.) attack, 2022)

7.3.5 Saphyra

Layer 7 attacks are very difficult to differentiate from legitimate traffic because they use standard HTTP requests. One way to protect the server against DDoS bots is to send a JavaScript

computational challenge like a Captcha to all users. DDoS bots typically are not able to solve the Captcha therefore the connection will be closed.

To defend against cache-busting attacks, Cloudflare analyses the response from the origin web server and, if the request was invalid, it will not cache and will drop other requests with the same parameters. However, this technique will not work if the website is configured to be flexible about what types of paths it can handle and treats a path that does not exist the same as a valid path. (Understanding Our Cache and the Web Cache Deception Attack, 2022)

Other defence techniques include Web Application Firewall and machine learning based on IP reputational databases and sets of manged rules. The scale of cloud platforms like Cloudflare or AWS gives them the advantage of analysing traffic from millions of users being able to efficiently update their firewalls and block malicious traffic.

CHAPTER 8 – EVALUATION AND CONCLUSION

8.1 System evaluation

Although it was finished on time, the development of the project took longer than expected. A lot of research was needed before starting the hands-on part as concepts like Docker containers and cloud computing were relatively new and only basic cloud computing was studied in University. A proper understanding of DDoS techniques used by the attackers was also needed to build a realistic simulator.

The hands-on part was the most challenging, there was a lot of troubleshooting, many unexpected errors due to the number of services, applications and programs used to build the system.

The attack tools were built into Docker images and successfully deployed in the AWS cloud using both EC2 launch type and the Fargate service. Through this experience it was found Fargate provides a much smoother deployment with advantages like auto-scaling and load-balancing. The container activity logs showed the attack tools were running correctly in each container making them resemble a DDoS botnet.

The project aim was to run the attack tools in 500 containers and to make a comparison of the DDoS defence capabilities of two websites hosted on Cloudflare and AWS. This was not possible due to budget limitations, only 50 containers were used without any effect on the websites. However, through research it was found even with 500 containers the effect on the web servers would have been very limited if not zero because they are protected by cloud reverse proxies that have the ability to absorb huge amounts of traffic. The simulator would probably be useful in testing web servers that are not protected by any cloud platforms and there are still many hosting providers like this on the Internet.

8.2 Personal reflection

This was a more challenging project than I had anticipated however it was also very rewarding. The two Udemy courses on Docker containers and cloud computing were very helpful but other websites like Stack Overflow or YouTube were more useful in troubleshooting the errors during the development stage.

The most difficult part was the cloud deployment, specifically the EC2 launch type because many other services like security groups or IAM roles had to be taken into consideration and all the settings had to be done manually.

The research at the beginning of the project and particularly the hands-on experience helped me developed a better understanding of cloud computing and Docker container concepts and improve the networking and cyber security skills I had acquired during my studies at University.

The specific learning outcomes include the ability to set up and work with Docker containers, deploying applications in the cloud and a deeper understanding on the networking principles behind different types of DDOS attacks.

My personal aim for this project was to use this experience to develop my skills so that I will be able to develop other freelance cloud and container projects that will help me obtain my AWS Solutions Architect Certification and land a job in the field. During the development of the project, I have obtained my AWS Cloud Practitioner Certification which is an entry-level cloud certification and made progress that had not been possible without this hands-on experience.

8.3 Conclusion

The two main objectives of this project were to build a cloud-based DDoS simulator and to test the DDoS defences of two cloud hosting providers. A cloud-based DDoS simulator built with Docker containers has been developed however, testing the two websites was only partially achieved due to budget limitations. Through research it was found the simulator would have had little to no effect even with hundreds of attacking containers because the cloud platforms have great DDoS defences in place. Even so, the simulator could prove useful in testing other smaller website hosting providers that can still be found on the Internet today.

The research and development of the system have been a very good learning process and new, important skills have been picked up along the way.

The result of this project can be considered a success because it is a working framework that can be used by researchers to conduct real-world DDoS experiments at a large scale.

CHAPTER 9 - REFERENCES

Amazon CloudWatch - Application and Infrastructure Monitoring, 2022 [online] Available at: <https://aws.amazon.com/cloudwatch/> [Accessed 25 March 2022].

Amazon EC2 key pairs and Linux instances, 2022 [online] Available at: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html> [Accessed 23 March 2022].

Application layer attacks, 2022 [online] Available at: <https://docs.aws.amazon.com/whitepapers/latest/aws-best-practices-ddos-resiliency/application-layer-attacks.html> [Accessed 21 April 2022].

AWS Fargate–Amazon Web Services, 2022 [online] Available at: <https://aws.amazon.com/fargate/> [Accessed 24 March 2022].

AWS Shield - Amazon Web Services (AWS), 2022 [online] Available at: <https://aws.amazon.com/shield/?whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc> [Accessed 1 April 2022].

Carty, D., 2022. *What is Amazon Elastic Container Registry (Amazon ECR)? Definition from SearchITOperations*. [online] SearchITOperations. Available at: <https://www.techtarget.com/searchitoperations/definition/Amazon-EC2-Container-Registry> [Accessed 23 March 2022].

Carty, D., 2022. *What is Amazon Elastic Container Service? Definition from WhatIs.com*. [online] SearchAWS. Available at: <https://www.techtarget.com/searchaws/definition/Amazon-EC2-Container-Service> [Accessed 24 March 2022].

Chickowski, E., 2022. *Types of DDoS attacks explained*. [online] Available at: <https://cybersecurity.att.com/blogs/security-essentials/types-of-ddos-attacks-explained> [Accessed 20 March 2022].

Cloudflare on IBM Cloud, 2022 [online] Available at: <https://www.ibm.com/cloud/cloudflare> [Accessed 1 April 2022].

Containers vs. Virtual Machines (VMs): What's the Difference?, 2022 [online] Available at: <https://www.ibm.com/cloud/blog/containers-vs-vms> [Accessed 21 April 2022].

Control traffic to resources using security groups, 2022 [online] Available at: <https://docs.aws.amazon.com/vpc/latest/userguide/VPC_SecurityGroups.html> [Accessed 23 March 2022].

Dey, S., 2022. *Demystifying AWS ECS Task Definition - Part 1*. [online] Linkedin.com. Available at: <https://www.linkedin.com/pulse/demystifying-aws-ecs-task-definition-part-1-soumyadeep-dey/> [Accessed 25 March 2022].

Dhanapal, A. and Nithyanandam, P., 2019. *The Slow HTTP DDOS Attacks: Detection, Mitigation and Prevention in the Cloud Environment*. [online] Available at: <https://www.researchgate.net/publication/337760460_The_Slow_HTTP_DDOS_Attacks_Detection_Mitigation_and_Prevention_in_the_Cloud_Environment> [Accessed 21 March 2022].

DNS amplification attack, 2022 [online] Available at: <https://www.cloudflare.com/en-gb/learning/ddos/dns-amplification-ddos-attack/> [Accessed 20 March 2022].

Docker overview, 2022 [online] Available at: <https://docs.docker.com/get-started/overview/> [Accessed 10 April 2022].

Fixed ModuleNotFoundError: No module named 'colorama, 2022 [online] Available at: <https://blog.finxter.com/fixed-modulenotfounderror-no-module-named-colorama/> [Accessed 1 April 2022].

Gillis, A., 2022. *What is a Docker Image? Introduction and use cases*. [online] SearchITOperations. Available at: <https://www.techtarget.com/searchitoperations/definition/Docker-image> [Accessed 21 March 2022].

How Amazon VPC works, 2022 [online] Available at: <https://docs.aws.amazon.com/vpc/latest/userguide/how-it-works.html> [Accessed 23 March 2022].

How is a Ping flood attack mitigated?, 2022 [online] Available at: <https://www.cloudflare.com/en-gb/learning/ddos/ping-icmp-flood-ddos-attack/#:~:text=The%20DDoS%20form%20of%20a,IP%20address%20as%20a%20response.> [Accessed 1 April 2022].

H*ping3 | Kali Linux Tools*, 2022 [online] Available at: <https://www.kali.org/tools/hping3/> [Accessed 20 March 2022].

HTTP headers - HTTP | MDN, 2022. [online] Available at: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers> [Accessed 21 April 2022].

IAM roles, 2022 [online] Available at: <https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html> [Accessed 24 March 2022].

Jia, Y., Zhong, F., Alrawais, A., Gong, B. and Cheng, X., 2020. *FlowGuard: An Intelligent Edge Defense Mechanism Against IoT DDoS Attacks*. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/document/9090824> [Accessed 21 March 2022].

Kumar, R., 2022. *Cloudflare Market Share 2022 [Statistics & Report]*. [online] WPOven Blog. Available at: <https://www.wpoven.com/blog/cloudflare-market-share/#:~:text=Cloudflare%20market%20share%20is%20about,Fiverr> [Accessed 18 April 2022].

Lunden, I., 2022. *TechCrunch is part of the Yahoo family of brands*. [online] Techcrunch.com. Available at: <https://techcrunch.com/2018/03/28/godaddy-to-move-most-of-its-infrastructure-to-aws-not-including-domain-management-for-its-75m-domains/> [Accessed 31 March 2022].

McMillen, D., 2022. *Dissecting a Hacktivist's DDoS Tool: Saphyra Revealed*. [online] Available at: <https://www.cloudflare.com/learning/ddos/ddos-attack-tools/r-u-dead-yet-rudy/> [Accessed 20 March 2022].

Molteni, D., 2022. *Improving the WAF with Machine Learning*. [online] Available at: <https://blog.cloudflare.com/waf-ml/> [Accessed 1 April 2022].

Ping (ICMP) flood DDoS attack, 2022 [online] Available at: <https://www.cloudflare.com/en-gb/learning/ddos/ping-icmp-flood-ddos-attack/> [Accessed 21 April 2022].

Ping of death DDoS attack, 2022[online] Available at: <https://www.cloudflare.com/en-gb/learning/ddos/ping-of-death-ddos-attack/> [Accessed 21 April 2022].

R U Dead Yet? (R.U.D.Y.) attack, 2022 [online] Available at: <https://www.cloudflare.com/learning/ddos/ddos-attack-tools/r-u-dead-yet-rudy/> [Accessed 20 March 2022].

SYN flood attack, 2022 [online] Available at: <https://www.cloudflare.com/en-gb/learning/ddos/syn-flood-ddos-attack/> [Accessed 2 April 2022].

Task definitions, 2022 [online] Available at: <https://ecsworkshop.com/introduction/ecs_basics/task_definition/#:~:text=The%20task%20definition%20is%20a,various%20parameters%20for%20your%20application.> [Accessed 24 March 2022].

UDP flood attack, 2022 [online] Available at: <https://www.cloudflare.com/en-gb/learning/ddos/udp-flood-ddos-attack/> [Accessed 2 April 2022].

Understanding Cloudflare DDoS protection, 2022 [online] Available at: <https://support.cloudflare.com/hc/en-us/articles/200172676-Understanding-Cloudflare-DDoS-protection> [Accessed 1 April 2022].

Understanding Our Cache and the Web Cache Deception Attack, 2022 [online] Available at: <https://blog.cloudflare.com/understanding-our-cache-and-the-web-cache-deception-attack/> [Accessed 21 April 2022].

What is a reverse proxy? | Proxy servers explained, 2022 [online] Available at: <https://www.cloudflare.com/en-gb/learning/cdn/glossary/reverse-proxy/#:~:text=A%20reverse%20proxy%20is%20a,security%2C%20performance%2C%20and%20reliability.> [Accessed 1 April 2022].

What is Anycast? | How does Anycast work?, 2022 [online] Available at: <https://www.cloudflare.com/learning/cdn/glossary/anycast-network/> [Accessed 1 April 2022].

What Is Cloud Computing? A Beginner's Guide | Microsoft Azure, 2022 [online] Available at: <https://azure.microsoft.com/en-gb/overview/what-is-cloud-computing/#benefits> [Accessed 10 April 2022].

Wigmore, I., 2022. *What is an Amazon EC2 instance?*. [online] SearchAWS. Available at: <https://www.techtarget.com/searchaws/definition/Amazon-EC2-instances> [Accessed 23 March 2022].

Yoachimik, O. and Ganti, V., 2022. *DDoS Attack Trends for Q4 2021*. [online] Available at: <https://blog.cloudflare.com/ddos-attack-trends-for-2021-q4/> [Accessed 21 April 2022].

CHAPTER 10 - BIBLIOGRAPHY

Deploy a Container Web Application with Amazon ECS | Introduction, 2022 [online] Available at: <https://aws.amazon.com/getting-started/guides/deploy-webapp-ecs/> [Accessed 13 April 2022].

What is Cloud Computing, 2022 [online] Available at: <https://aws.amazon.com/what-is-cloud-computing/> [Accessed 13 April 2022].

Docker Container – Aqua, 2022 [online] Available at: <https://www.aquasec.com/cloud-native-academy/docker-container/> [Accessed 13 April 2022].

Behal, S. and Kumar, K., 2019. [online] Ijns.jalaxy.com.tw. Available at: <http://ijns.jalaxy.com.tw/contents/ijns-v19-n3/ijns-2017-v19-n3-p383-393.pdf> [Accessed 13 April 2022].

Simulation of Internet DDoS Attacks and Defense, 2022. [online] Available at: < http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.101.7003&rep=rep1&type=pdf [Accessed 13 April 2022].

Davis, N., 2022. *AWS Certified Solutions Architect Associate SAA-C02 [2022]*. [online] Available at: <https://www.udemy.com/course/aws-certified-solutions-architect-associate-hands-on/> [Accessed 13 April 2022].

Still using Alpine as your docker's Python development base image? In fact, Ubuntu is better - Develop Paper, 2022 [online] Available at: <https://developpaper.com/still-using-alpine-as-your-dockers-python-development-base-image-in-fact-ubuntu-is-better/> [Accessed 13 April 2022].

Install Docker Engine on Ubuntu, 2022 [online] Available at: <https://docs.docker.com/engine/install/ubuntu/#install-using-the-convenience-script> [Accessed 13 April 2022].

Grider, S., 2022. *Docker and Kubernetes: The Complete Guide*. [online] Available at: <https://www.udemy.com/course/docker-and-kubernetes-the-complete-guide/> [Accessed 13 April 2022].

Jekyll • Simple, blog-aware, static sites. *Command Line Usage*, 2022[online] Available at: <https://jekyllrb.com/docs/usage/> [Accessed 13 April 2022].

Kumar, V. and Kumar, K., 2016. *Classification of DDoS attack tools and its handling techniques and strategy at application layer*. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/abstract/document/7749002> [Accessed 13 April 2022].

Load Balancing without Load Balancers. 2013 [online] Available at: <https://blog.cloudflare.com/cloudflares-architecture-eliminating-single-p/> [Accessed 13 April 2022].

Luan, S., 2022. [online] Cs.unm.edu. Available at: <https://www.cs.unm.edu/~compmed/workshop2011/talks/17a.pdf> [Accessed 13 April 2022].

Nagpal, B., Sharma, P., Chauhan, N. and Panesar, A., 2022. *DDoS tools: Classification, analysis and comparison*. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/abstract/document/7100270> [Accessed 13 April 2022].

Poston, H., 2022. *Understanding DoS attacks and the best free DoS attacking tools [updated in 2020] - Infosec Resources*. [online] Infosec Resources. Available at: <https://resources.infosecinstitute.com/topic/dos-attacks-free-dos-attacking-tools/> [Accessed 13 April 2022].

Simic, S., 2022. *Docker Image VS Container: What is the difference?*. [online] Knowledge Base by phoenixNAP. Available at: <https://phoenixnap.com/kb/docker-image-vs-container> [Accessed 13 April 2022].

Simulating test DDoS attacks. 2022 [online] Available at: <https://developers.cloudflare.com/ddos-protection/reference/simulate-ddos-attack/> [Accessed 13 April 2022].

SUSE Communities. *How to Build and Run Your Own Container Images | SUSE Communities*, 2022 [online] Available at: <https://www.suse.com/c/rancher_blog/how-to-build-and-run-your-own-container-images/> [Accessed 13 April 2022].

AWS ECS Tutorial | Amazon Elastic Container Service | AWS ECS Tutorial For Beginners | Simplilearn, 2022. [online] Available at: <https://www.youtube.com/watch?v=46mFdtpy3NQ> [Accessed 13 April 2022].

How to setup Docker Registry In Amazon ECR | Create Docker Image and Push to Amazon ECR | ECR Docker, 2022. [online] Available at: <https://www.youtube.com/watch?v=D8ym8RP1yvo> [Accessed 13 April 2022].

Creating your first Dockerfile, Image and Container, 2022. [online] Available at: <https://www.youtube.com/watch?v=hnxI-K10auY&t=50s> [Accessed 13 April 2022].

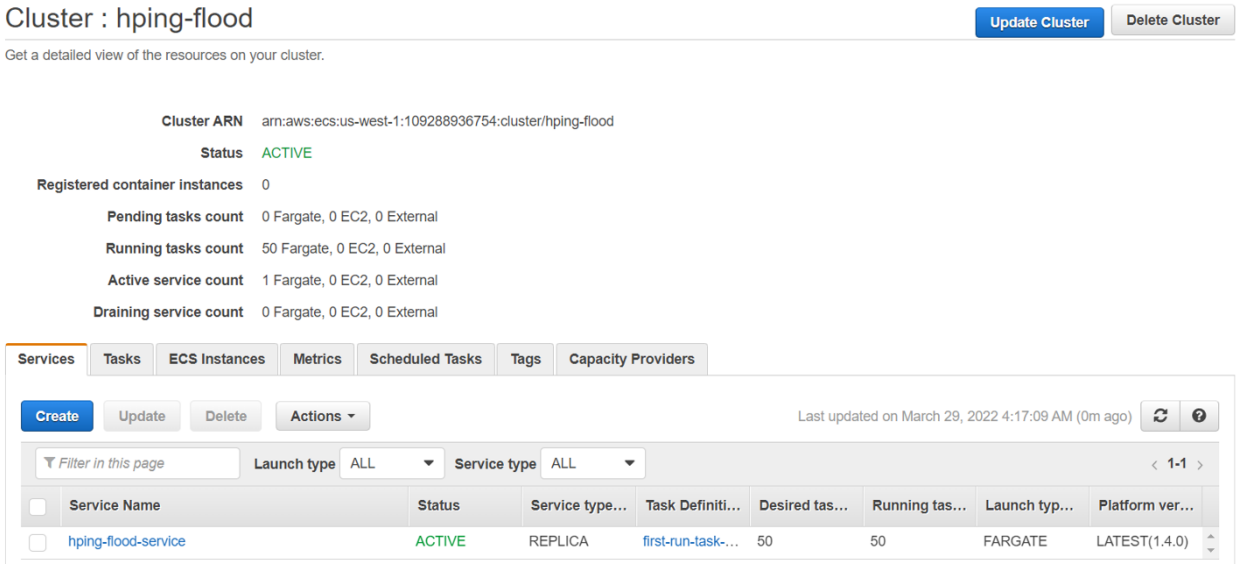
[AWS 24] Running containers in ECS using Fargate, 2022. [online] Available at: <https://www.youtube.com/watch?v=NPiMar8OTjY> [Accessed 13 April 2022].

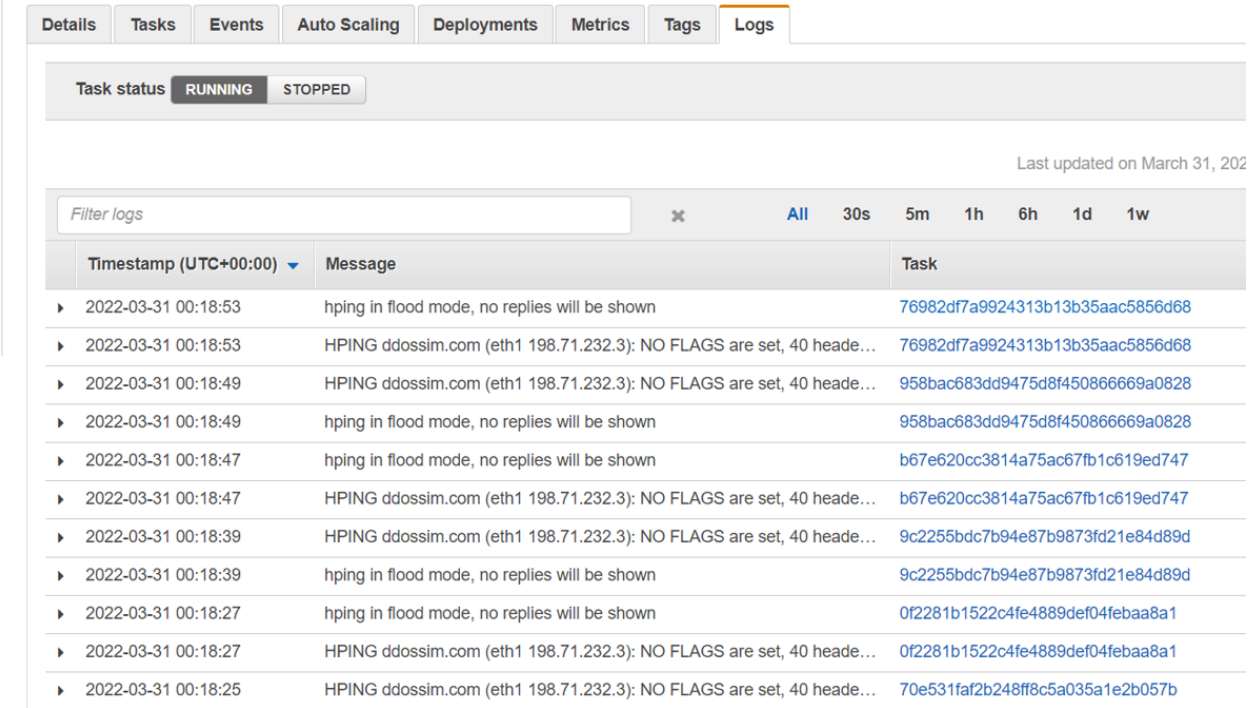
I bought a DDoS attack on the Dark Web (don't do this), 2022. [online] Available at: <https://www.youtube.com/watch?v=eZYtnzODpW4> [Accessed 13 April 2022].

Deep dive on Amazon Elastic Container Service (Amazon ECS), 2022. [online] Available at: <https://www.youtube.com/watch?v=qbEPae8YNbs> [Accessed 13 April 2022].

CHAPTER 11 - APPENDICES

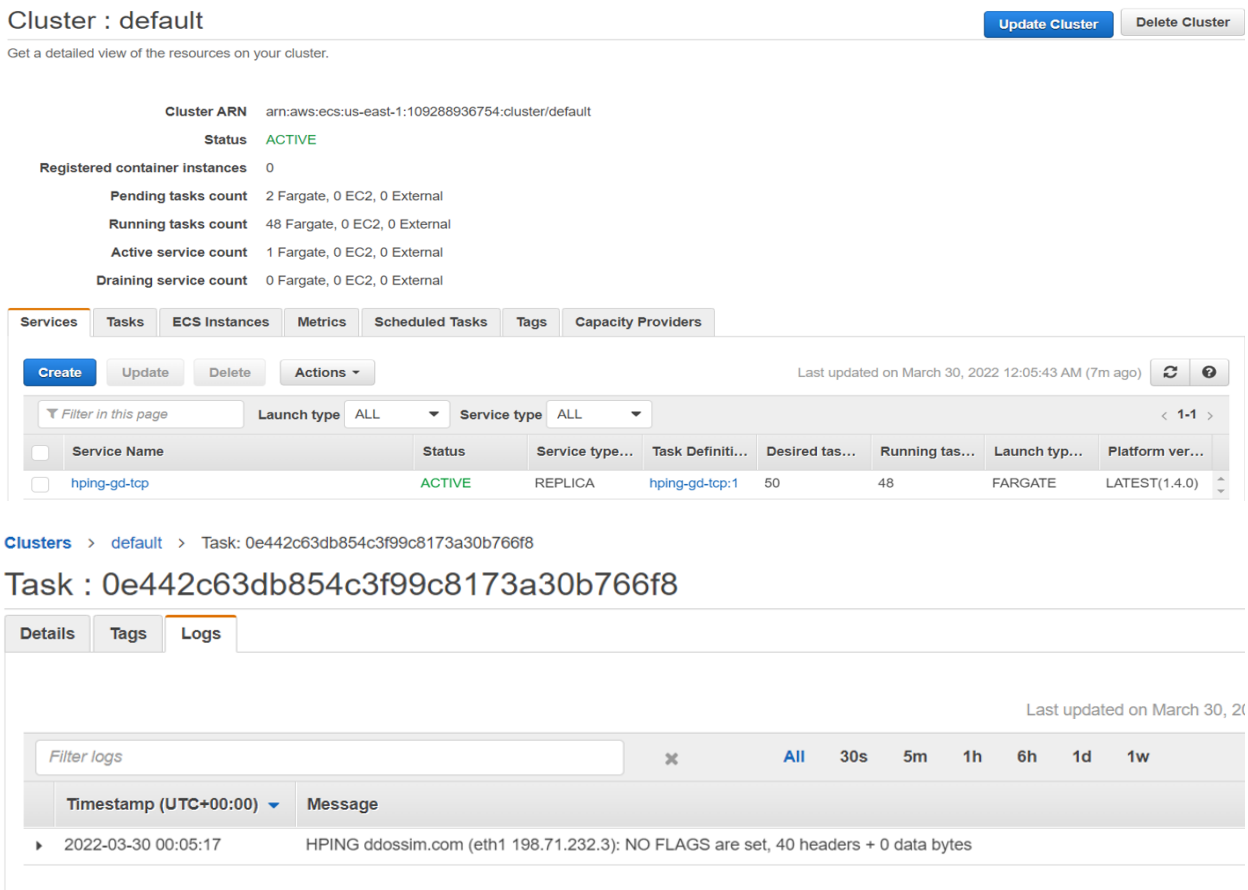
Appendix 1 - Hping3 Flood attack on ddossim.com-GoDaddy hosting





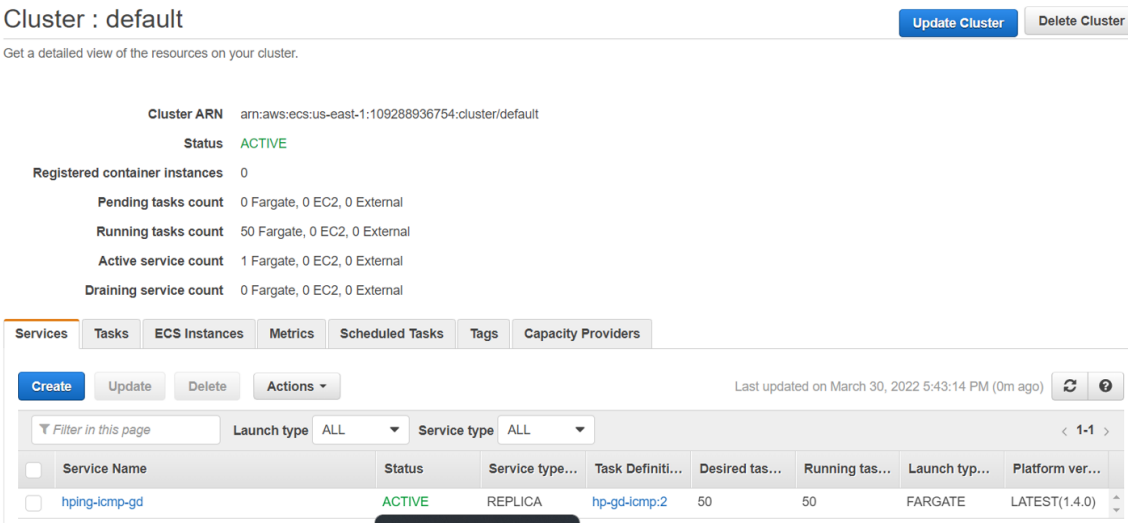
```
┌──(mihai㉿kali)-[~]
└─$ ping ddossim.com
PING ddossim.com (160.153.136.3) 56(84) bytes of data.
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=1 ttl=128 time=16.4 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=2 ttl=128 time=24.4 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=3 ttl=128 time=24.6 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=4 ttl=128 time=24.1 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=5 ttl=128 time=24.5 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=6 ttl=128 time=18.6 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=7 ttl=128 time=24.4 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=8 ttl=128 time=24.5 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=9 ttl=128 time=71.0 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=10 ttl=128 time=25.0 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=11 ttl=128 time=24.5 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=12 ttl=128 time=24.9 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=13 ttl=128 time=24.1 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=14 ttl=128 time=25.1 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=15 ttl=128 time=25.1 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=16 ttl=128 time=22.8 ms
^C
```

Appendix 2 - Hping3 TCP attack on ddossim.com-GoDaddy hosting



```
┌──(mihai㉿kali)-[~]
└─$ ping ddossim.com
PING ddossim.com (160.153.136.3) 56(84) bytes of data.
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=1 ttl=128 time=99.5 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=2 ttl=128 time=25.8 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=3 ttl=128 time=25.8 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=4 ttl=128 time=25.1 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=5 ttl=128 time=24.6 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=6 ttl=128 time=17.4 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=7 ttl=128 time=26.2 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=8 ttl=128 time=25.0 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=9 ttl=128 time=25.7 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=10 ttl=128 time=25.6 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=11 ttl=128 time=20.3 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=12 ttl=128 time=25.6 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=13 ttl=128 time=24.6 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=14 ttl=128 time=21.5 ms
c^C64 bytes from 160.153.136.3: icmp_seq=15 ttl=128 time=26.6 ms
```

Appendix 3 - Hping3 ICMP attack on ddossim.com-GoDaddy hosting



Clusters > default > Task: 104da8dbc766404cb9dbe03e595fe961

Task : 104da8dbc766404cb9dbe03e595fe961

Details | Tags | Logs

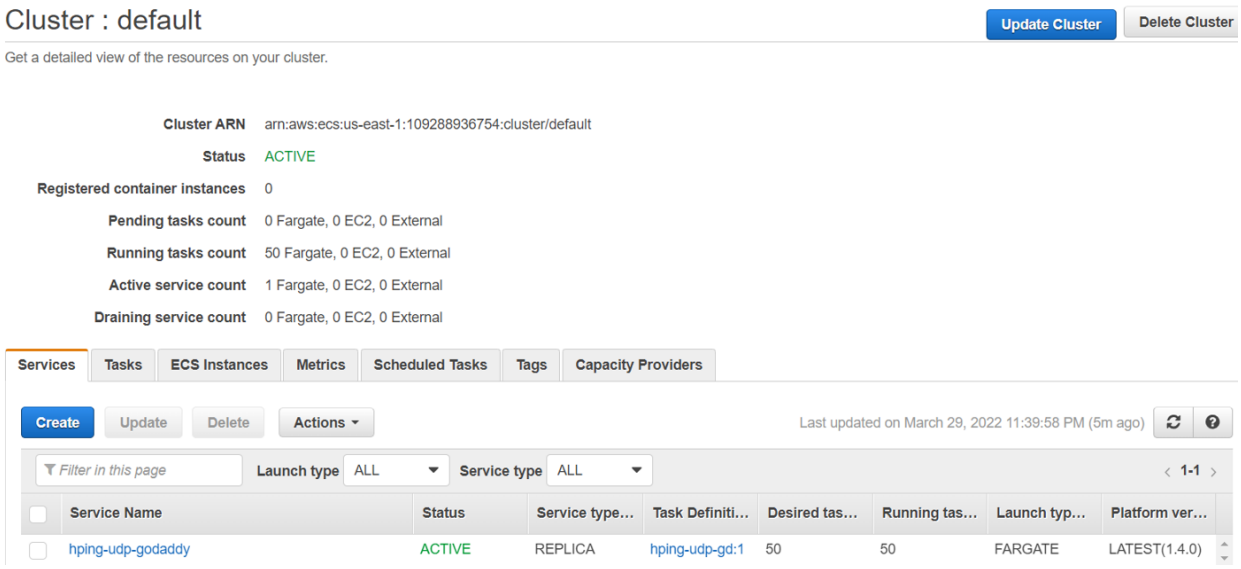
Last updated on

Filter logs | All | 30s | 5m | 1h | 6h | 1d | 1w

Timestamp (UTC+00:00)	Message
2022-03-30 17:42:14	len=46 ip=198.71.232.3 ttl=47 id=64969 icmp_seq=6 rtt=7.4 ms
2022-03-30 17:42:14	len=46 ip=198.71.232.3 ttl=47 id=19146 icmp_seq=7 rtt=7.3 ms
2022-03-30 17:42:14	len=46 ip=198.71.232.3 ttl=47 id=41418 icmp_seq=8 rtt=7.2 ms
2022-03-30 17:42:14	len=46 ip=198.71.232.3 ttl=47 id=61130 icmp_seq=9 rtt=7.1 ms
2022-03-30 17:42:14	len=46 ip=198.71.232.3 ttl=47 id=15307 icmp_seq=10 rtt=7.1 ms
2022-03-30 17:42:14	len=46 ip=198.71.232.3 ttl=47 id=36811 icmp_seq=11 rtt=7.0 ms
2022-03-30 17:42:14	len=46 ip=198.71.232.3 ttl=47 id=61131 icmp_seq=12 rtt=2.9 ms
2022-03-30 17:42:14	len=46 ip=198.71.232.3 ttl=47 id=16076 icmp_seq=13 rtt=2.8 ms
2022-03-30 17:42:14	len=46 ip=198.71.232.3 ttl=47 id=37836 icmp_seq=14 rtt=2.7 ms
2022-03-30 17:42:14	len=46 ip=198.71.232.3 ttl=47 id=60364 icmp_seq=15 rtt=2.6 ms
2022-03-30 17:41:57	HPING ddossim.com (eth1 198.71.232.3): icmp mode set, 28 headers + 0 data bytes

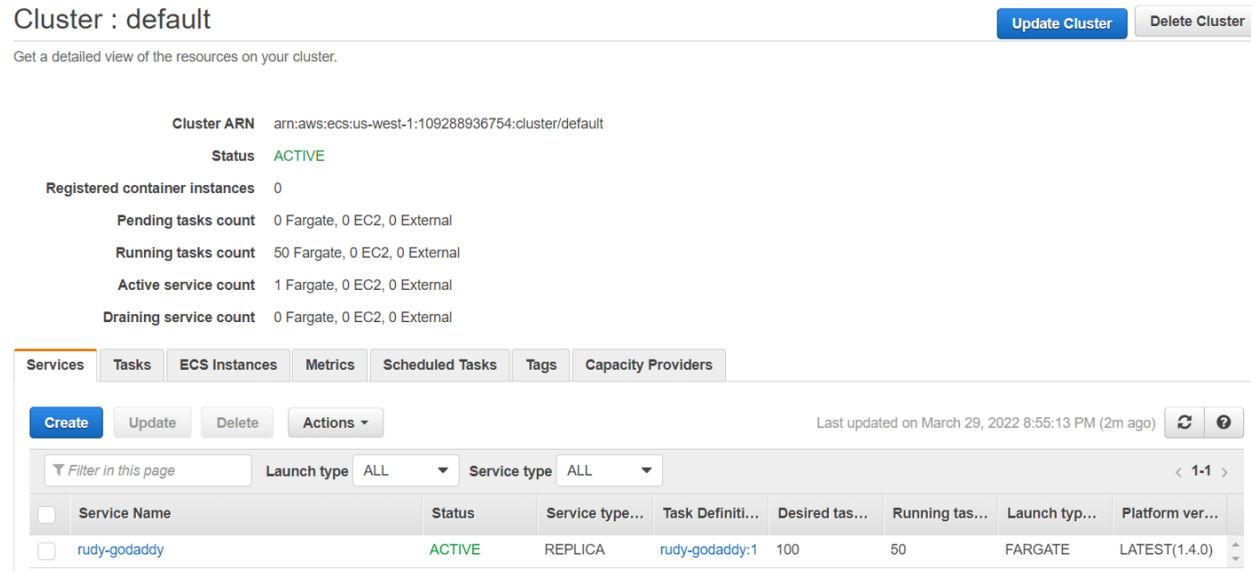
```
┌──(mihai㉿kali)-[~]
└─$ ping ddossim.com
PING ddossim.com (160.153.136.3) 56(84) bytes of data.
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=1 ttl=128 time=14.7 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=2 ttl=128 time=17.0 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=3 ttl=128 time=18.4 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=4 ttl=128 time=16.5 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=5 ttl=128 time=14.7 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=6 ttl=128 time=17.3 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=7 ttl=128 time=17.1 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=8 ttl=128 time=16.1 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=9 ttl=128 time=15.8 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=10 ttl=128 time=15.6 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=11 ttl=128 time=16.7 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=12 ttl=128 time=16.8 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=13 ttl=128 time=16.8 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=14 ttl=128 time=17.1 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=15 ttl=128 time=15.6 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=16 ttl=128 time=17.0 ms
^C
```

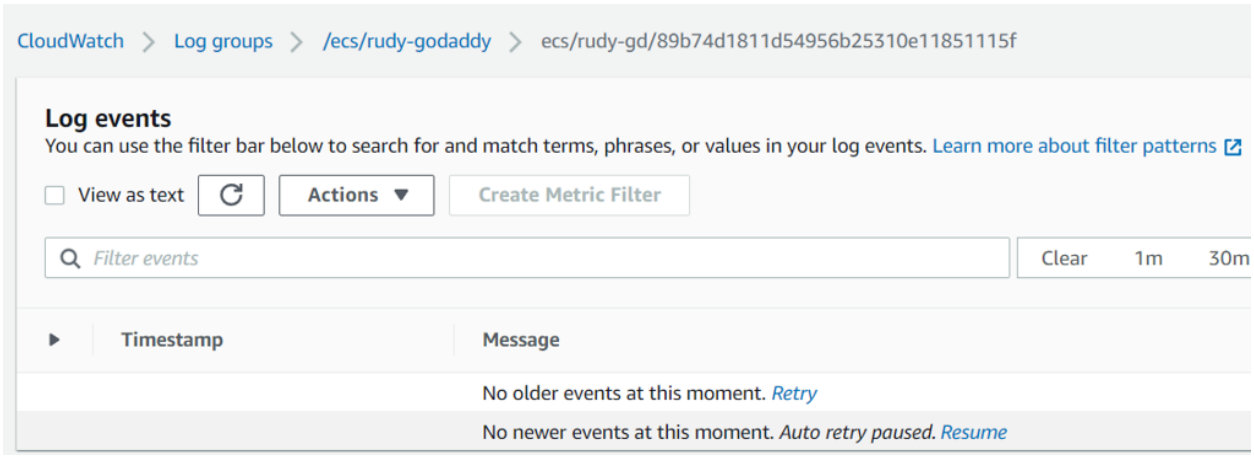
Appendix 4 - Hping3 UDP attack on ddossim.com-GoDaddy hosting




```
┌──(mihai㉿kali)-[~]
└─$ ping ddossim.com
PING ddossim.com (160.153.136.3) 56(84) bytes of data.
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=1 ttl=128 time=16.5 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=2 ttl=128 time=22.4 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=3 ttl=128 time=24.5 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=4 ttl=128 time=24.4 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=5 ttl=128 time=22.4 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=6 ttl=128 time=16.1 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=7 ttl=128 time=23.3 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=8 ttl=128 time=23.8 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=9 ttl=128 time=171 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=10 ttl=128 time=24.6 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=11 ttl=128 time=24.4 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=12 ttl=128 time=25.0 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=13 ttl=128 time=24.4 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=14 ttl=128 time=22.4 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=15 ttl=128 time=24.7 ms
^C
```

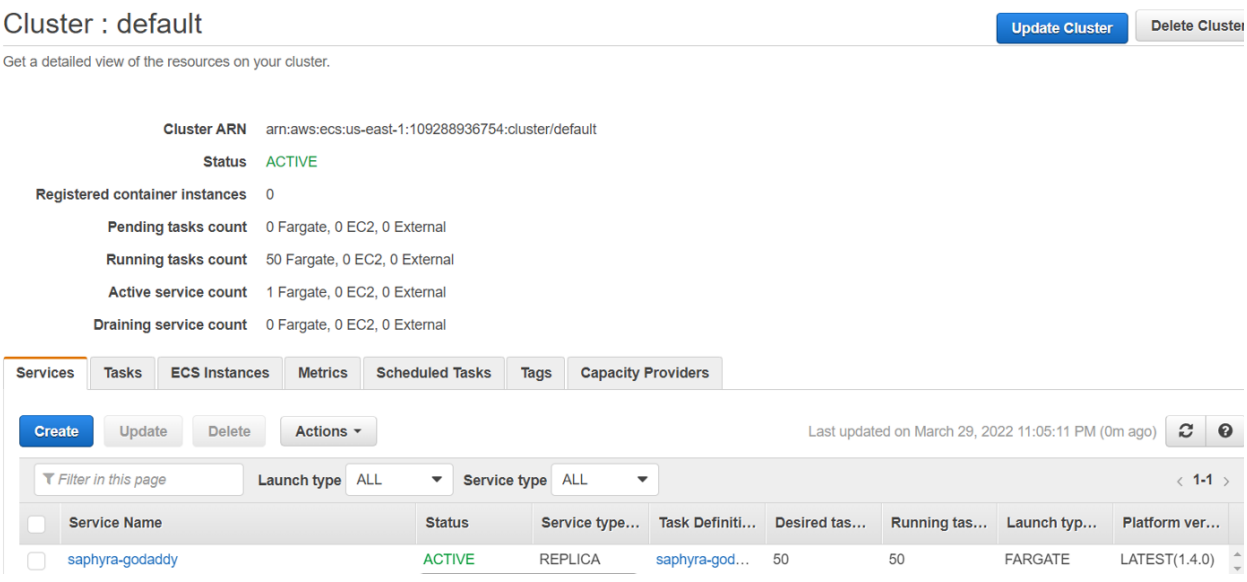
Appendix 5 - R.U.D.Y. attack on ddossim.com-GoDaddy hosting

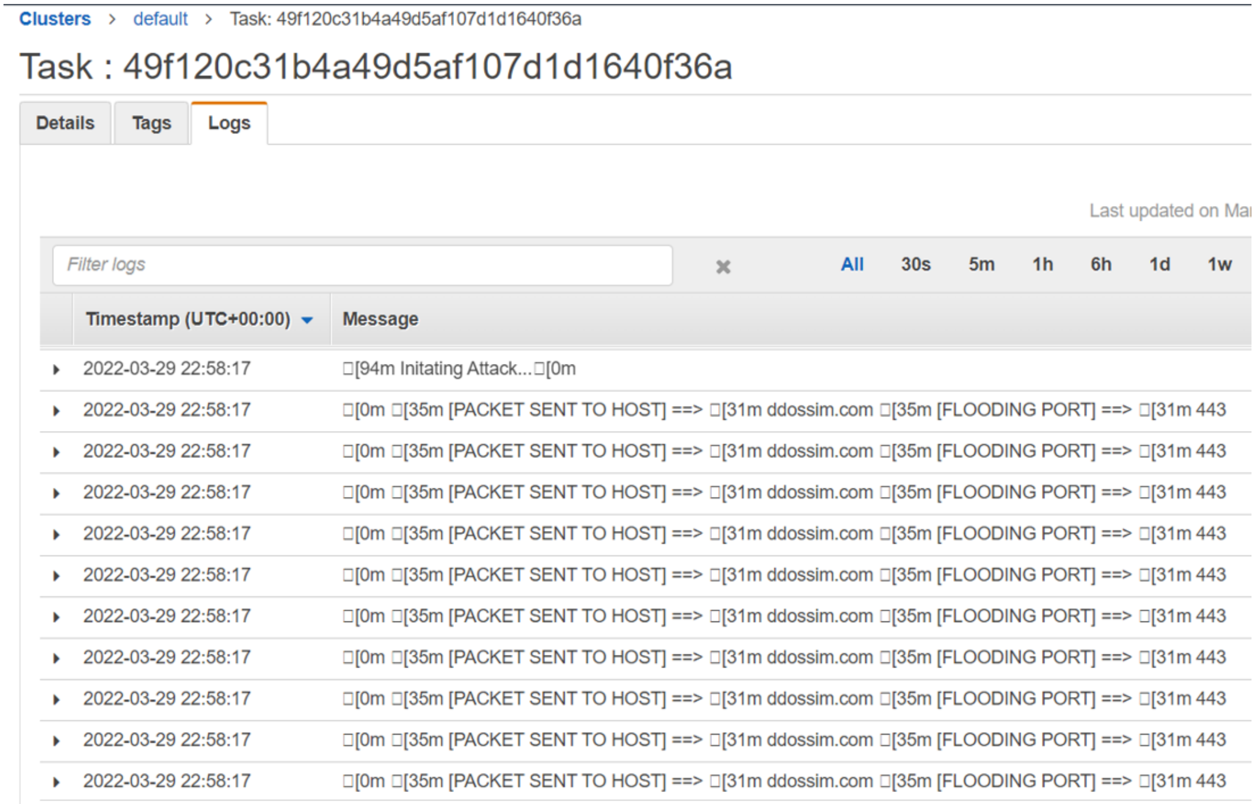




```
root@ip-172-31-1-7:/home/ubuntu/rudy/RUDY# ping ddossim.com
PING ddossim.com (72.167.191.69) 56(84) bytes of data.
64 bytes from ip-72-167-191-69.ip.secureserver.net (72.167.191.69): icmp_seq=1 ttl=230 time=18.5 ms
64 bytes from ip-72-167-191-69.ip.secureserver.net (72.167.191.69): icmp_seq=2 ttl=230 time=18.5 ms
64 bytes from ip-72-167-191-69.ip.secureserver.net (72.167.191.69): icmp_seq=3 ttl=230 time=18.5 ms
64 bytes from ip-72-167-191-69.ip.secureserver.net (72.167.191.69): icmp_seq=4 ttl=230 time=18.5 ms
64 bytes from ip-72-167-191-69.ip.secureserver.net (72.167.191.69): icmp_seq=5 ttl=230 time=18.5 ms
64 bytes from ip-72-167-191-69.ip.secureserver.net (72.167.191.69): icmp_seq=6 ttl=230 time=18.4 ms
64 bytes from ip-72-167-191-69.ip.secureserver.net (72.167.191.69): icmp_seq=7 ttl=230 time=18.6 ms
64 bytes from ip-72-167-191-69.ip.secureserver.net (72.167.191.69): icmp_seq=8 ttl=230 time=18.5 ms
64 bytes from ip-72-167-191-69.ip.secureserver.net (72.167.191.69): icmp_seq=9 ttl=230 time=18.4 ms
64 bytes from ip-72-167-191-69.ip.secureserver.net (72.167.191.69): icmp_seq=10 ttl=230 time=18.5 ms
64 bytes from ip-72-167-191-69.ip.secureserver.net (72.167.191.69): icmp_seq=11 ttl=230 time=18.4 ms
64 bytes from ip-72-167-191-69.ip.secureserver.net (72.167.191.69): icmp_seq=12 ttl=230 time=18.7 ms
64 bytes from ip-72-167-191-69.ip.secureserver.net (72.167.191.69): icmp_seq=13 ttl=230 time=21.0 ms
64 bytes from ip-72-167-191-69.ip.secureserver.net (72.167.191.69): icmp_seq=14 ttl=230 time=18.4 ms
64 bytes from ip-72-167-191-69.ip.secureserver.net (72.167.191.69): icmp_seq=15 ttl=230 time=18.4 ms
64 bytes from ip-72-167-191-69.ip.secureserver.net (72.167.191.69): icmp_seq=16 ttl=230 time=18.4 ms
64 bytes from ip-72-167-191-69.ip.secureserver.net (72.167.191.69): icmp_seq=17 ttl=230 time=18.4 ms
64 bytes from ip-72-167-191-69.ip.secureserver.net (72.167.191.69): icmp_seq=18 ttl=230 time=18.4 ms
64 bytes from ip-72-167-191-69.ip.secureserver.net (72.167.191.69): icmp_seq=19 ttl=230 time=18.5 ms
64 bytes from ip-72-167-191-69.ip.secureserver.net (72.167.191.69): icmp_seq=20 ttl=230 time=18.5 ms
64 bytes from ip-72-167-191-69.ip.secureserver.net (72.167.191.69): icmp_seq=21 ttl=230 time=18.5 ms
64 bytes from ip-72-167-191-69.ip.secureserver.net (72.167.191.69): icmp_seq=22 ttl=230 time=18.5 ms
64 bytes from ip-72-167-191-69.ip.secureserver.net (72.167.191.69): icmp_seq=23 ttl=230 time=18.4 ms
64 bytes from ip-72-167-191-69.ip.secureserver.net (72.167.191.69): icmp_seq=24 ttl=230 time=18.4 ms
^C
```

Appendix 6 - Saphyra attack on ddossim.com-GoDaddy hosting

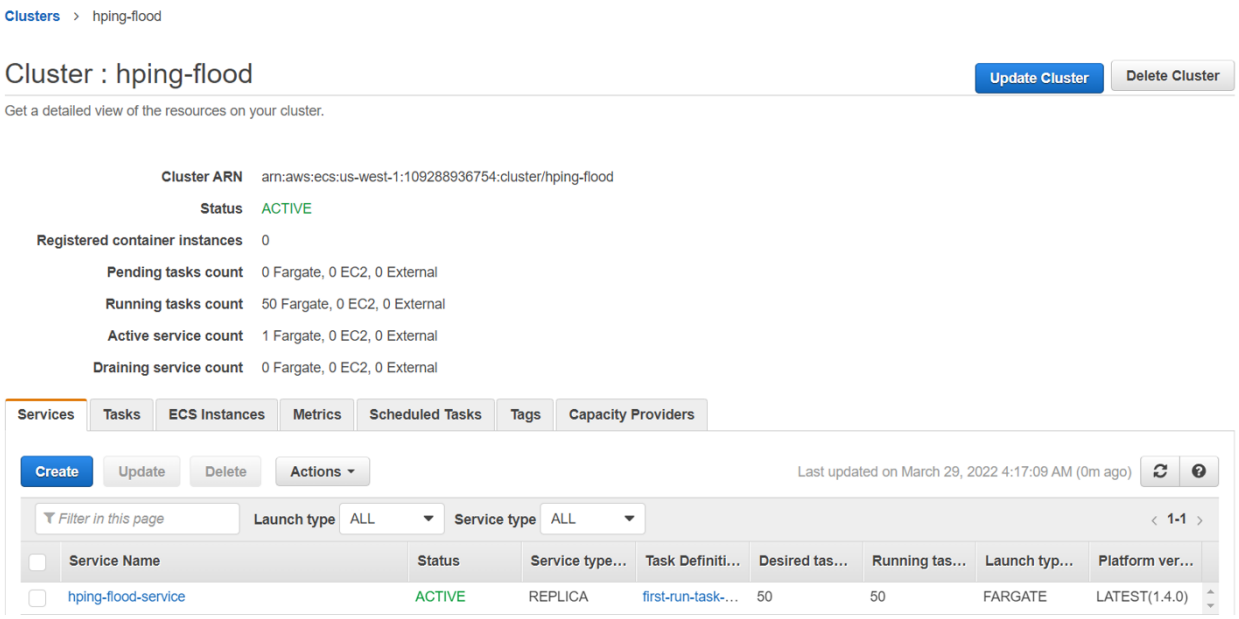


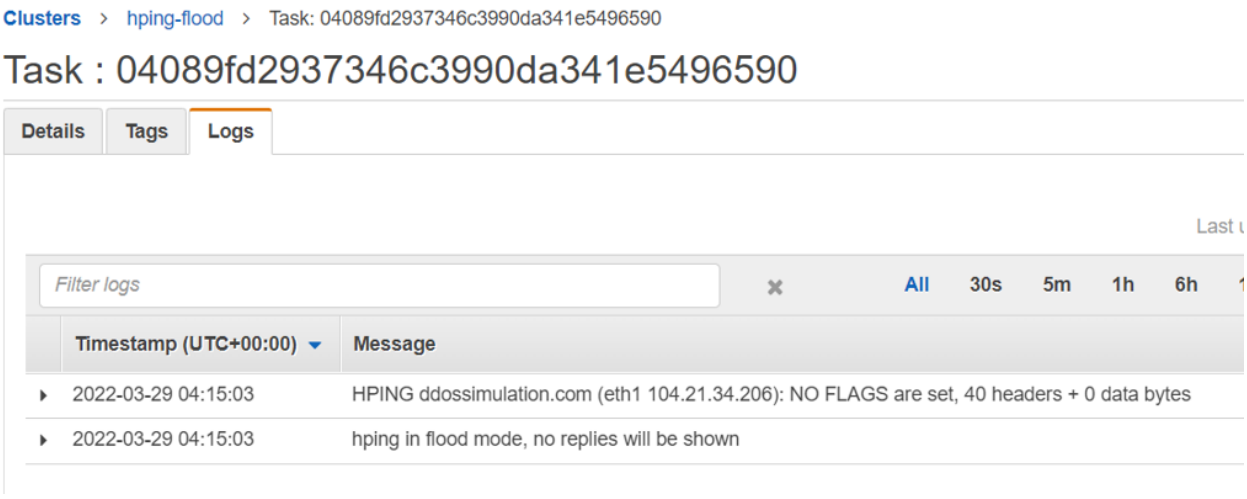


Timestamp (UTC+00:00)	Message
2022-03-29 22:58:17	□[94m Initating Attack...□[0m
2022-03-29 22:58:17	□[0m □[35m [PACKET SENT TO HOST] ==> □[31m ddossim.com □[35m [FLOODING PORT] ==> □[31m 443
2022-03-29 22:58:17	□[0m □[35m [PACKET SENT TO HOST] ==> □[31m ddossim.com □[35m [FLOODING PORT] ==> □[31m 443
2022-03-29 22:58:17	□[0m □[35m [PACKET SENT TO HOST] ==> □[31m ddossim.com □[35m [FLOODING PORT] ==> □[31m 443
2022-03-29 22:58:17	□[0m □[35m [PACKET SENT TO HOST] ==> □[31m ddossim.com □[35m [FLOODING PORT] ==> □[31m 443
2022-03-29 22:58:17	□[0m □[35m [PACKET SENT TO HOST] ==> □[31m ddossim.com □[35m [FLOODING PORT] ==> □[31m 443
2022-03-29 22:58:17	□[0m □[35m [PACKET SENT TO HOST] ==> □[31m ddossim.com □[35m [FLOODING PORT] ==> □[31m 443
2022-03-29 22:58:17	□[0m □[35m [PACKET SENT TO HOST] ==> □[31m ddossim.com □[35m [FLOODING PORT] ==> □[31m 443
2022-03-29 22:58:17	□[0m □[35m [PACKET SENT TO HOST] ==> □[31m ddossim.com □[35m [FLOODING PORT] ==> □[31m 443
2022-03-29 22:58:17	□[0m □[35m [PACKET SENT TO HOST] ==> □[31m ddossim.com □[35m [FLOODING PORT] ==> □[31m 443
2022-03-29 22:58:17	□[0m □[35m [PACKET SENT TO HOST] ==> □[31m ddossim.com □[35m [FLOODING PORT] ==> □[31m 443

```
┌──(mihai㉿kali)-[~]
└─$ ping ddossim.com
PING ddossim.com (160.153.136.3) 56(84) bytes of data.
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=1 ttl=128 time=23.8 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=2 ttl=128 time=25.9 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=3 ttl=128 time=25.1 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=4 ttl=128 time=24.6 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=5 ttl=128 time=20.3 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=6 ttl=128 time=22.5 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=7 ttl=128 time=25.6 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=8 ttl=128 time=23.3 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=9 ttl=128 time=23.0 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=10 ttl=128 time=22.9 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=11 ttl=128 time=138 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=12 ttl=128 time=23.2 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=13 ttl=128 time=24.9 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=14 ttl=128 time=22.1 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=15 ttl=128 time=22.1 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=16 ttl=128 time=23.3 ms
64 bytes from ip-160-153-136-3.ip.secureserver.net (160.153.136.3): icmp_seq=17 ttl=128 time=25.1 ms
^C
```

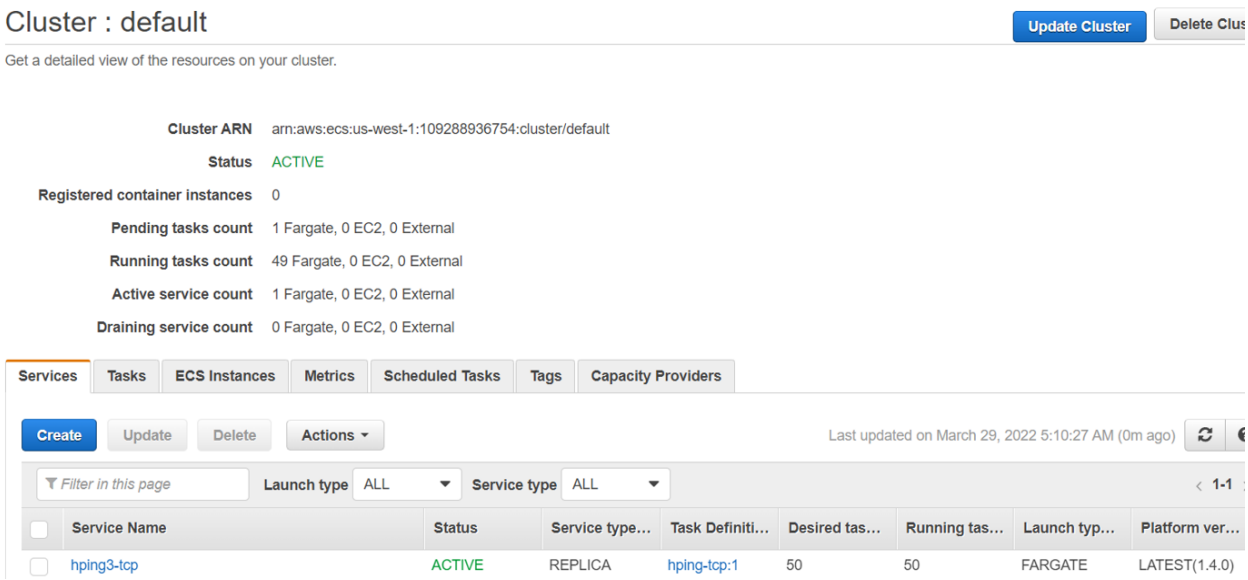
Appendix 7 - Hping3 Flood attack on ddossimulation.com-Cloudflare hosting

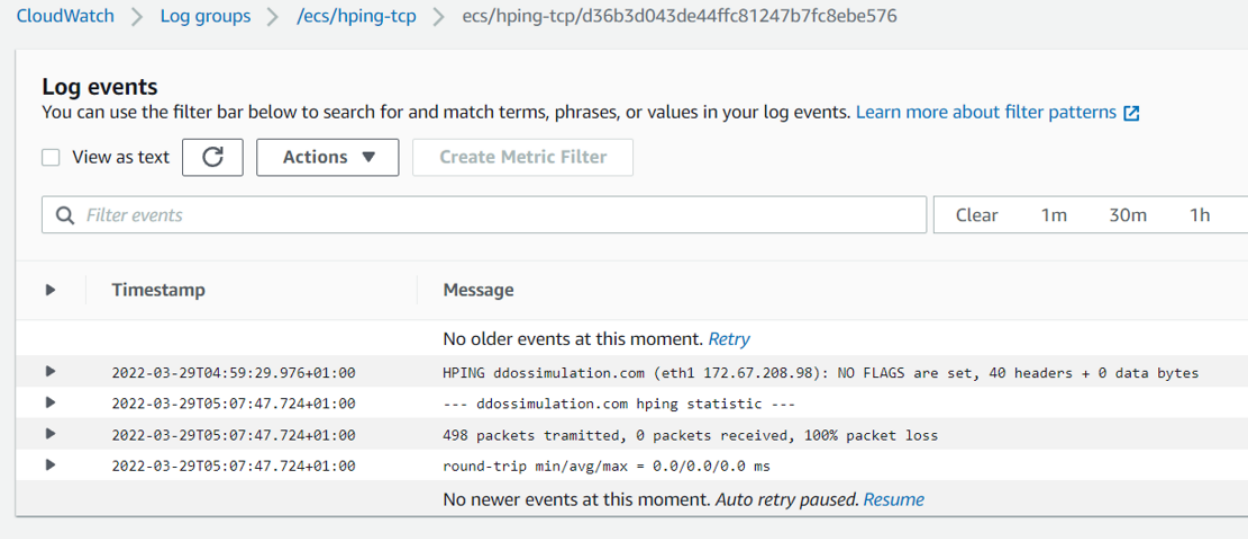




```
┌──(mihai㉿kali)-[~]
└─$ ping ddossimulation.com                                                    1 ✕
PING ddossimulation.com (172.67.208.98) 56(84) bytes of data.
64 bytes from 172.67.208.98 (172.67.208.98): icmp_seq=1 ttl=128 time=10.0 ms
64 bytes from 172.67.208.98 (172.67.208.98): icmp_seq=2 ttl=128 time=11.1 ms
64 bytes from 172.67.208.98 (172.67.208.98): icmp_seq=3 ttl=128 time=11.1 ms
64 bytes from 172.67.208.98 (172.67.208.98): icmp_seq=4 ttl=128 time=11.3 ms
64 bytes from 172.67.208.98 (172.67.208.98): icmp_seq=5 ttl=128 time=11.6 ms
64 bytes from 172.67.208.98 (172.67.208.98): icmp_seq=6 ttl=128 time=11.7 ms
64 bytes from 172.67.208.98 (172.67.208.98): icmp_seq=7 ttl=128 time=14.1 ms
64 bytes from 172.67.208.98 (172.67.208.98): icmp_seq=8 ttl=128 time=11.7 ms
64 bytes from 172.67.208.98 (172.67.208.98): icmp_seq=9 ttl=128 time=11.4 ms
64 bytes from 172.67.208.98 (172.67.208.98): icmp_seq=10 ttl=128 time=11.3 ms
64 bytes from 172.67.208.98 (172.67.208.98): icmp_seq=11 ttl=128 time=11.2 ms
64 bytes from 172.67.208.98 (172.67.208.98): icmp_seq=12 ttl=128 time=11.7 ms
64 bytes from 172.67.208.98 (172.67.208.98): icmp_seq=13 ttl=128 time=11.5 ms
64 bytes from 172.67.208.98 (172.67.208.98): icmp_seq=14 ttl=128 time=11.5 ms
64 bytes from 172.67.208.98 (172.67.208.98): icmp_seq=15 ttl=128 time=11.1 ms
64 bytes from 172.67.208.98 (172.67.208.98): icmp_seq=16 ttl=128 time=12.3 ms
64 bytes from 172.67.208.98 (172.67.208.98): icmp_seq=17 ttl=128 time=12.4 ms
64 bytes from 172.67.208.98 (172.67.208.98): icmp_seq=18 ttl=128 time=11.1 ms
64 bytes from 172.67.208.98 (172.67.208.98): icmp_seq=19 ttl=128 time=11.6 ms
64 bytes from 172.67.208.98 (172.67.208.98): icmp_seq=20 ttl=128 time=11.0 ms
64 bytes from 172.67.208.98 (172.67.208.98): icmp_seq=21 ttl=128 time=13.6 ms
64 bytes from 172.67.208.98 (172.67.208.98): icmp_seq=22 ttl=128 time=11.7 ms
64 bytes from 172.67.208.98 (172.67.208.98): icmp_seq=23 ttl=128 time=12.2 ms
64 bytes from 172.67.208.98 (172.67.208.98): icmp_seq=24 ttl=128 time=11.0 ms
64 bytes from 172.67.208.98 (172.67.208.98): icmp_seq=25 ttl=128 time=11.2 ms
```

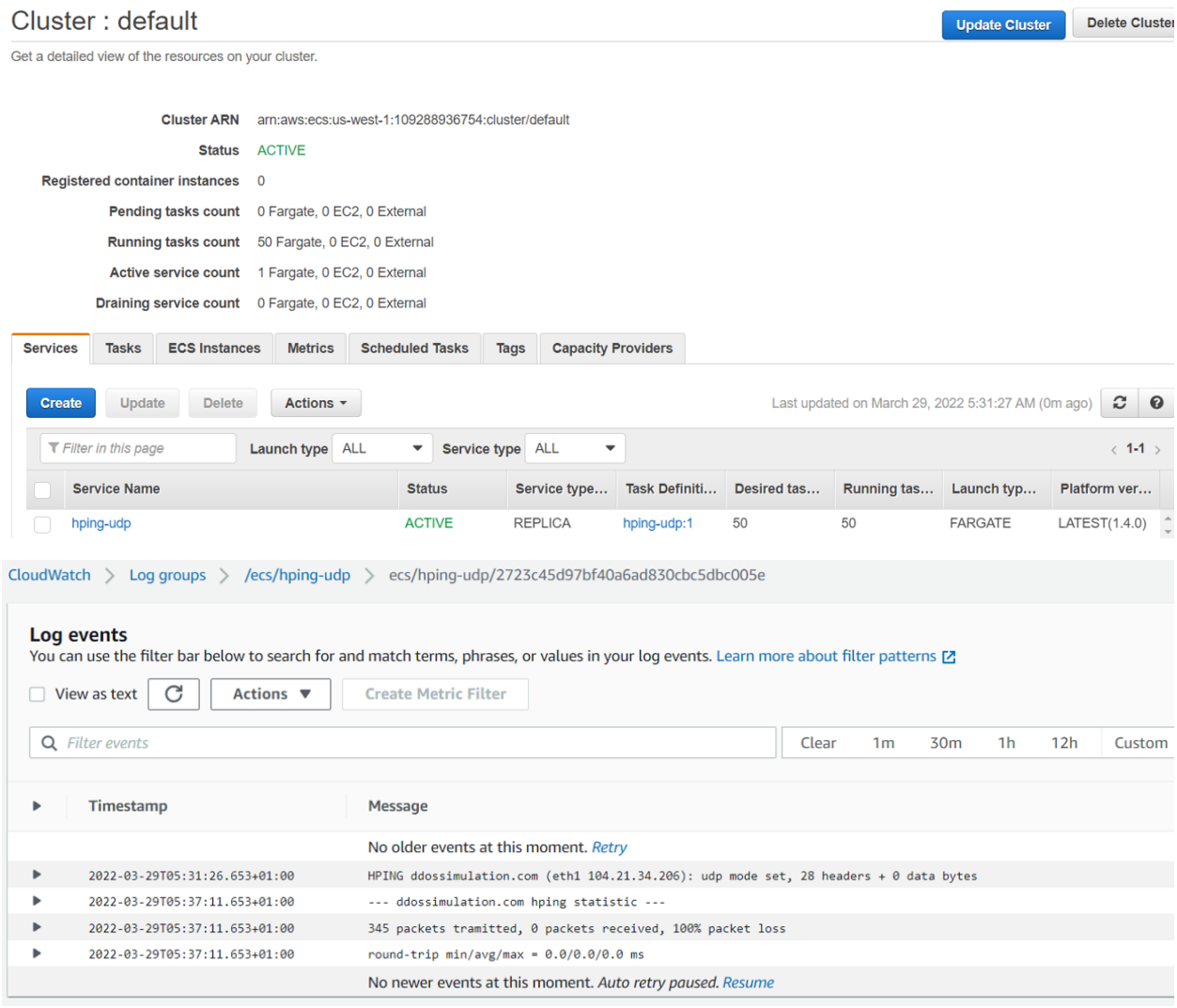
Appendix 8 - Hping3 TCP attack on ddossimulation.com-Cloudflare hosting





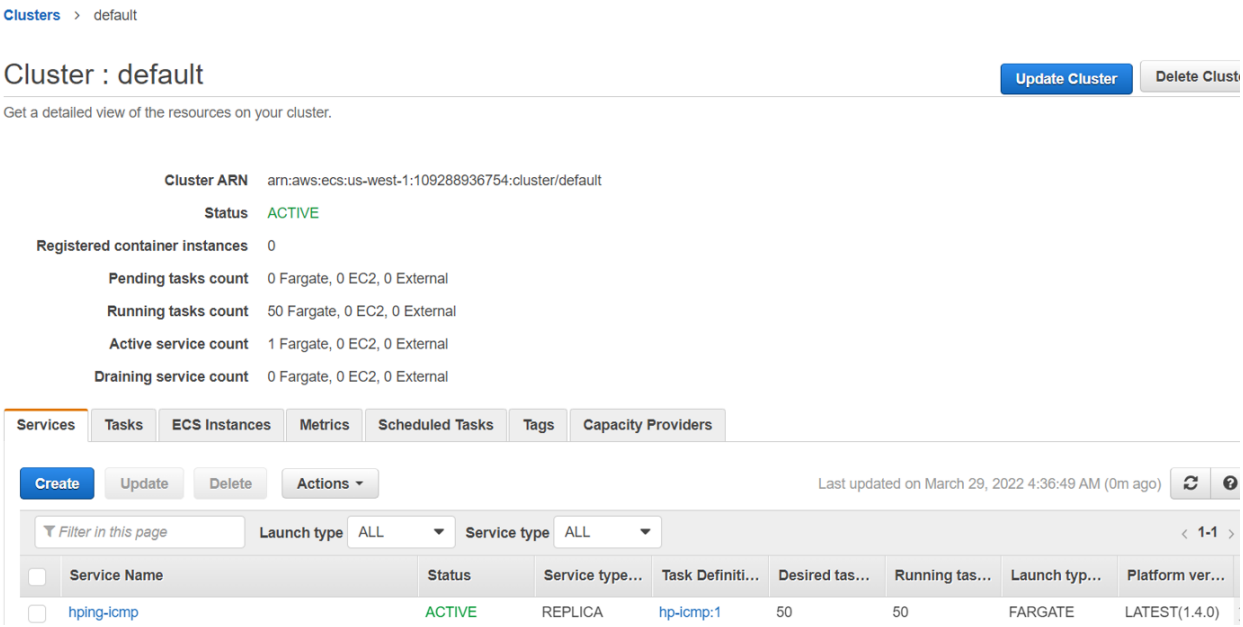

```
┌──(mihai㉿kali)-[~]
└─$ ping ddossimulation.com
PING ddossimulation.com (104.21.34.206) 56(84) bytes of data.
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=1 ttl=128 time=99.7 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=2 ttl=128 time=20.2 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=3 ttl=128 time=21.5 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=4 ttl=128 time=17.6 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=5 ttl=128 time=20.8 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=6 ttl=128 time=17.2 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=7 ttl=128 time=18.2 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=8 ttl=128 time=21.2 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=9 ttl=128 time=176 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=10 ttl=128 time=17.9 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=11 ttl=128 time=19.9 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=12 ttl=128 time=17.9 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=13 ttl=128 time=20.5 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=14 ttl=128 time=20.9 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=15 ttl=128 time=19.5 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=16 ttl=128 time=19.4 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=17 ttl=128 time=20.5 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=18 ttl=128 time=20.2 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=19 ttl=128 time=11.9 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=20 ttl=128 time=12.3 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=21 ttl=128 time=16.7 ms
```

Appendix 9 - Hping3 UDP attack on ddossimulation.com-Cloudflare hosting



```
┌──(mihai㉿kali)-[~]
└─$ ping ddossimulation.com
PING ddossimulation.com (104.21.34.206) 56(84) bytes of data.
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=1 ttl=128 time=88.9 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=2 ttl=128 time=13.4 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=3 ttl=128 time=19.1 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=4 ttl=128 time=19.9 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=5 ttl=128 time=76.0 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=6 ttl=128 time=19.9 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=7 ttl=128 time=17.6 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=8 ttl=128 time=17.9 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=9 ttl=128 time=19.9 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=10 ttl=128 time=20.3 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=11 ttl=128 time=17.5 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=12 ttl=128 time=17.5 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=13 ttl=128 time=147 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=14 ttl=128 time=20.4 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=15 ttl=128 time=18.0 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=16 ttl=128 time=15.1 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=17 ttl=128 time=20.6 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=18 ttl=128 time=18.3 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=19 ttl=128 time=73.0 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=20 ttl=128 time=21.0 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=21 ttl=128 time=18.0 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=22 ttl=128 time=20.0 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=23 ttl=128 time=20.7 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=24 ttl=128 time=20.2 ms
```

Appendix 10 - Hping3 ICMP attack on ddossimulation.com-Cloudflare hosting



CloudWatch > Log groups > /ecs/hp-icmp > ecs/hping-icmp/e00b44ea7c1c49bda0a758069581ffbd

Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

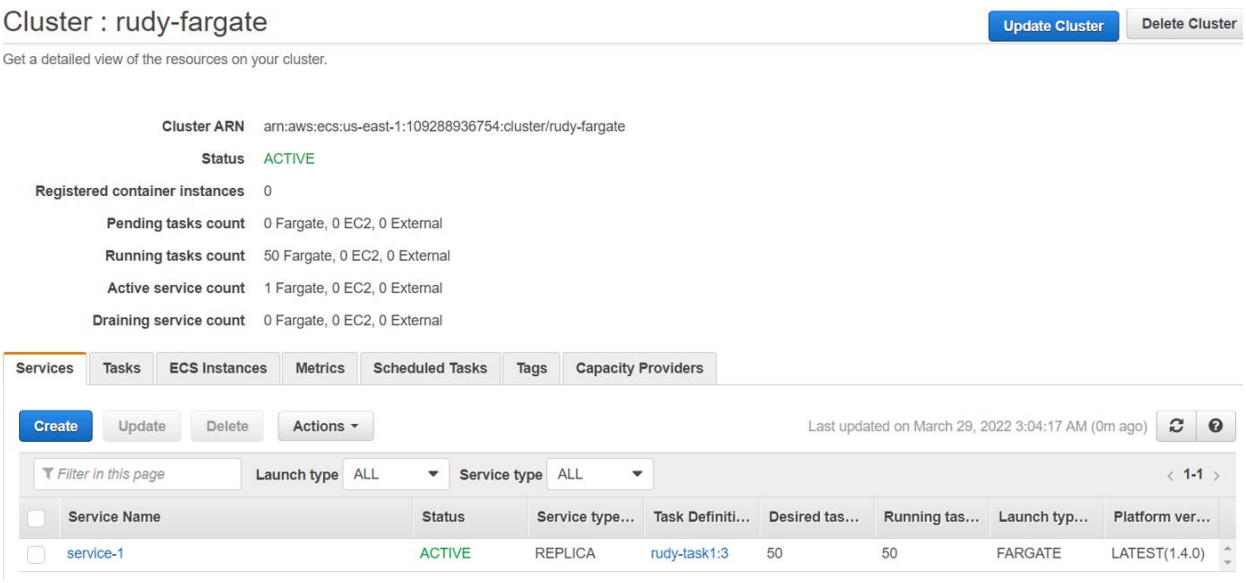
View as text | Actions | Create Metric Filter

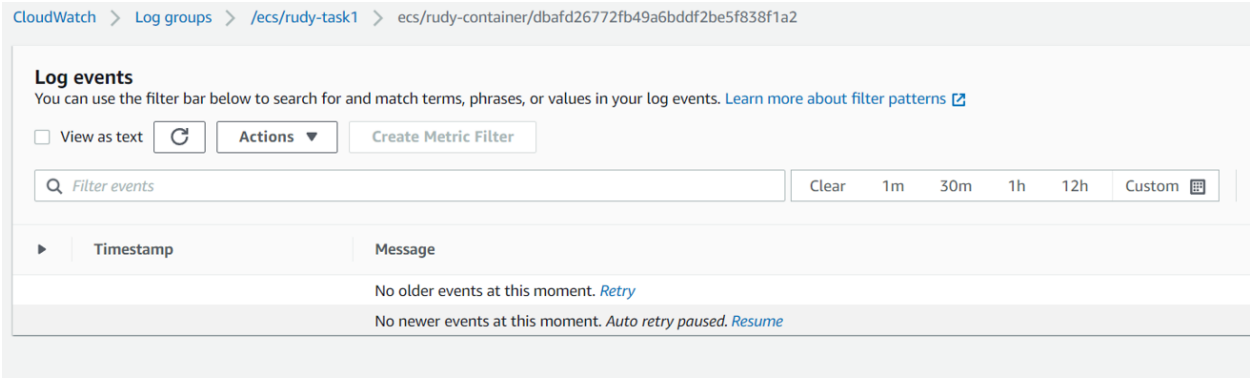
Filter events | Clear | 1m | 30m | 1h

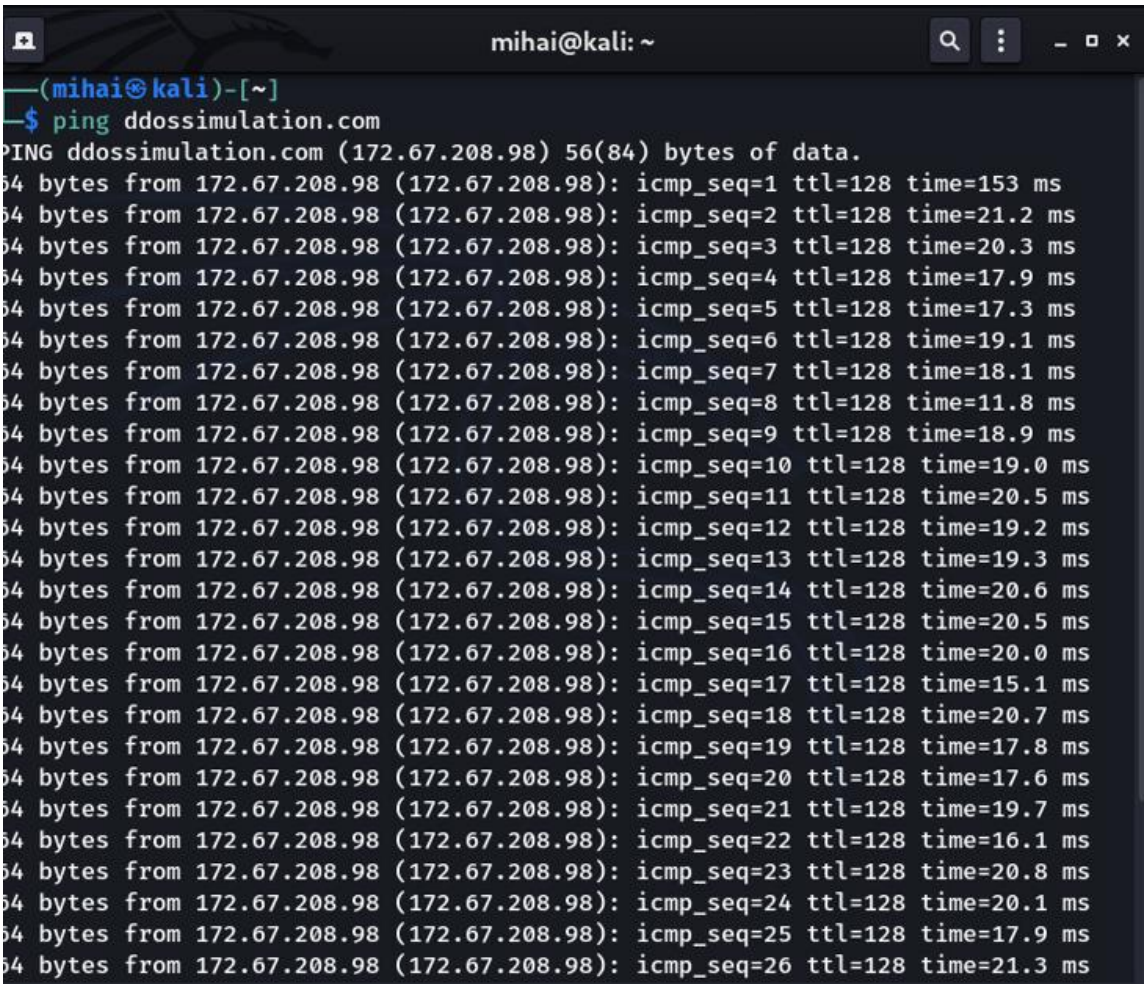
Timestamp	Message
	No older events at this moment. *Retry*
2022-03-29T04:34:25.579+01:00	HPING ddossimulation.com (eth1 172.67.208.98): icmp mode set, 28 headers + 0 data bytes
2022-03-29T04:34:41.595+01:00	len=46 ip=172.67.208.98 ttl=46 id=18731 icmp_seq=0 rtt=15.8 ms
2022-03-29T04:34:41.595+01:00	len=46 ip=172.67.208.98 ttl=46 id=14089 icmp_seq=1 rtt=15.8 ms
2022-03-29T04:34:41.595+01:00	len=46 ip=172.67.208.98 ttl=46 id=44657 icmp_seq=2 rtt=15.7 ms
2022-03-29T04:34:41.595+01:00	len=46 ip=172.67.208.98 ttl=46 id=52204 icmp_seq=3 rtt=15.7 ms
2022-03-29T04:34:41.595+01:00	len=46 ip=172.67.208.98 ttl=46 id=30055 icmp_seq=4 rtt=11.6 ms
2022-03-29T04:34:41.595+01:00	len=46 ip=172.67.208.98 ttl=46 id=31282 icmp_seq=5 rtt=11.6 ms
2022-03-29T04:34:41.595+01:00	len=46 ip=172.67.208.98 ttl=46 id=57347 icmp_seq=6 rtt=11.5 ms
2022-03-29T04:34:41.595+01:00	len=46 ip=172.67.208.98 ttl=46 id=26344 icmp_seq=7 rtt=11.5 ms
2022-03-29T04:34:41.595+01:00	len=46 ip=172.67.208.98 ttl=46 id=34240 icmp_seq=8 rtt=15.5 ms
2022-03-29T04:34:41.595+01:00	len=46 ip=172.67.208.98 ttl=46 id=37153 icmp_seq=9 rtt=15.4 ms
2022-03-29T04:34:41.595+01:00	len=46 ip=172.67.208.98 ttl=46 id=60930 icmp_seq=10 rtt=15.4 ms
2022-03-29T04:34:41.595+01:00	len=46 ip=172.67.208.98 ttl=46 id=17624 icmp_seq=11 rtt=11.3 ms

```
┌──(mihai㉿kali)-[~]
└─$ ping ddossimulation.com
PING ddossimulation.com (104.21.34.206) 56(84) bytes of data.
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=1 ttl=128 time=42.2 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=2 ttl=128 time=16.3 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=3 ttl=128 time=20.5 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=4 ttl=128 time=17.3 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=5 ttl=128 time=17.6 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=6 ttl=128 time=20.5 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=7 ttl=128 time=69.9 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=8 ttl=128 time=20.4 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=9 ttl=128 time=15.2 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=10 ttl=128 time=20.0 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=11 ttl=128 time=20.9 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=12 ttl=128 time=19.8 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=13 ttl=128 time=20.2 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=14 ttl=128 time=19.8 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=15 ttl=128 time=20.6 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=16 ttl=128 time=19.8 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=17 ttl=128 time=20.6 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=18 ttl=128 time=12.5 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=19 ttl=128 time=18.3 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=20 ttl=128 time=21.1 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=21 ttl=128 time=19.6 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=22 ttl=128 time=17.5 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=23 ttl=128 time=19.6 ms
^C
```

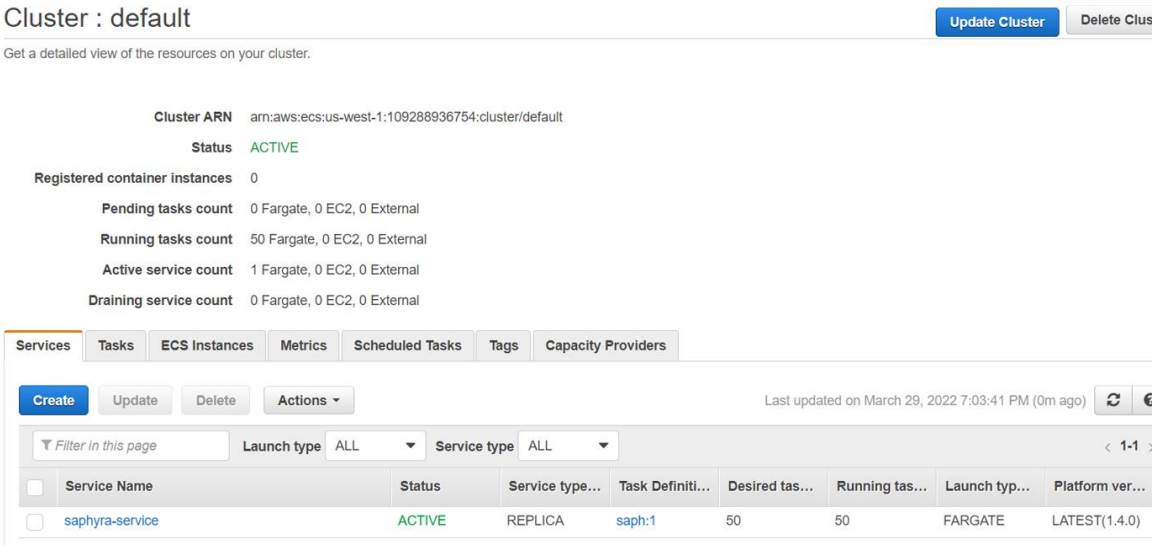
Appendix 11 - R.U.D.Y attack on ddossimulation.com-Cloudflare hosting

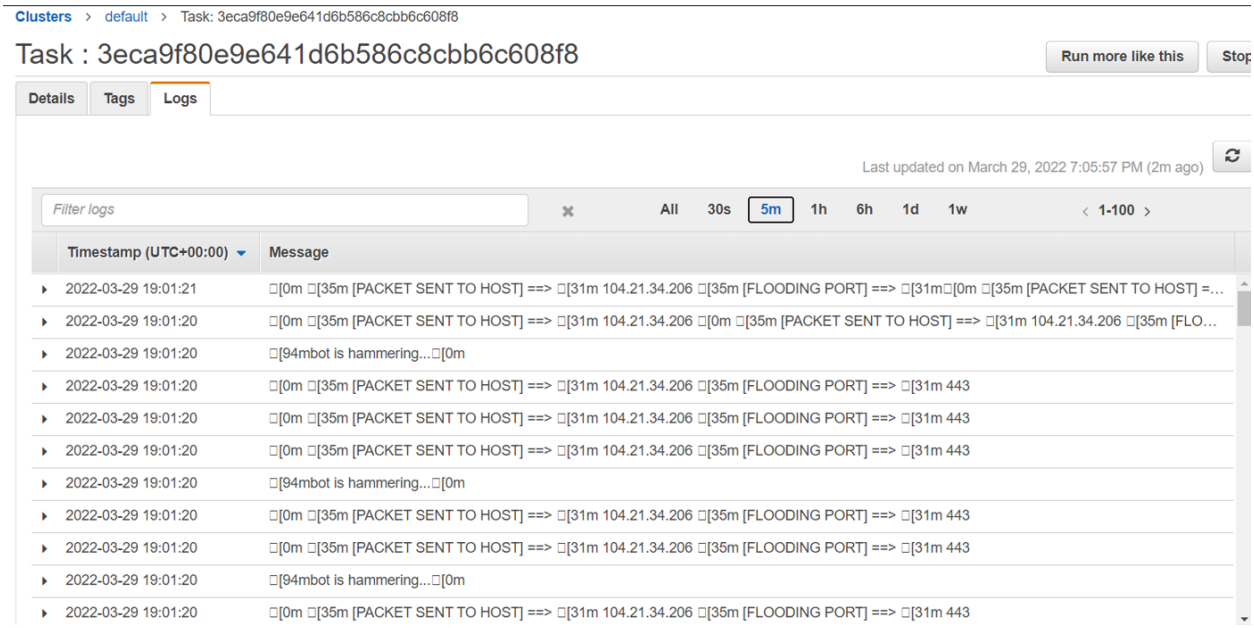


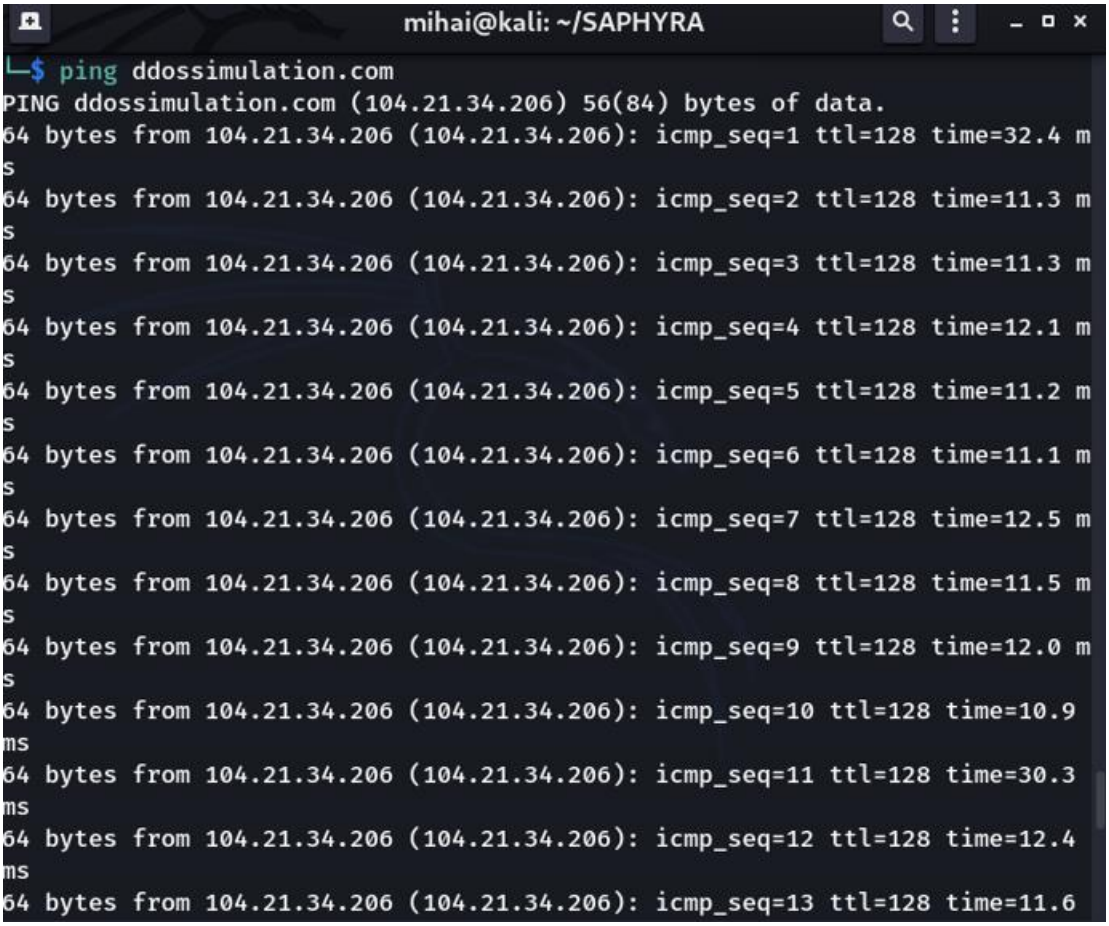




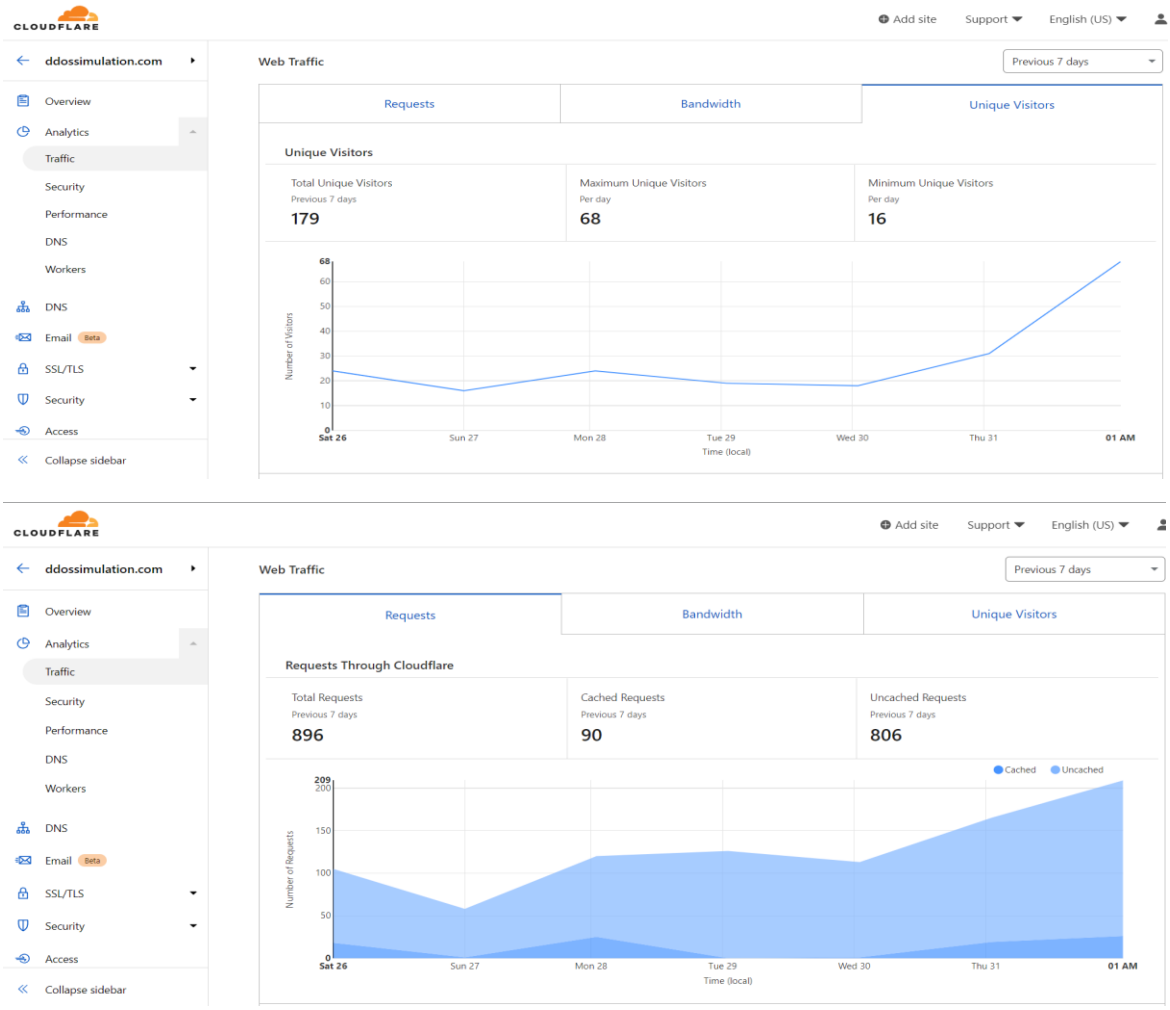
Appendix 12 - Saphyra attack on ddossimulation.com-Cloudflare hosting



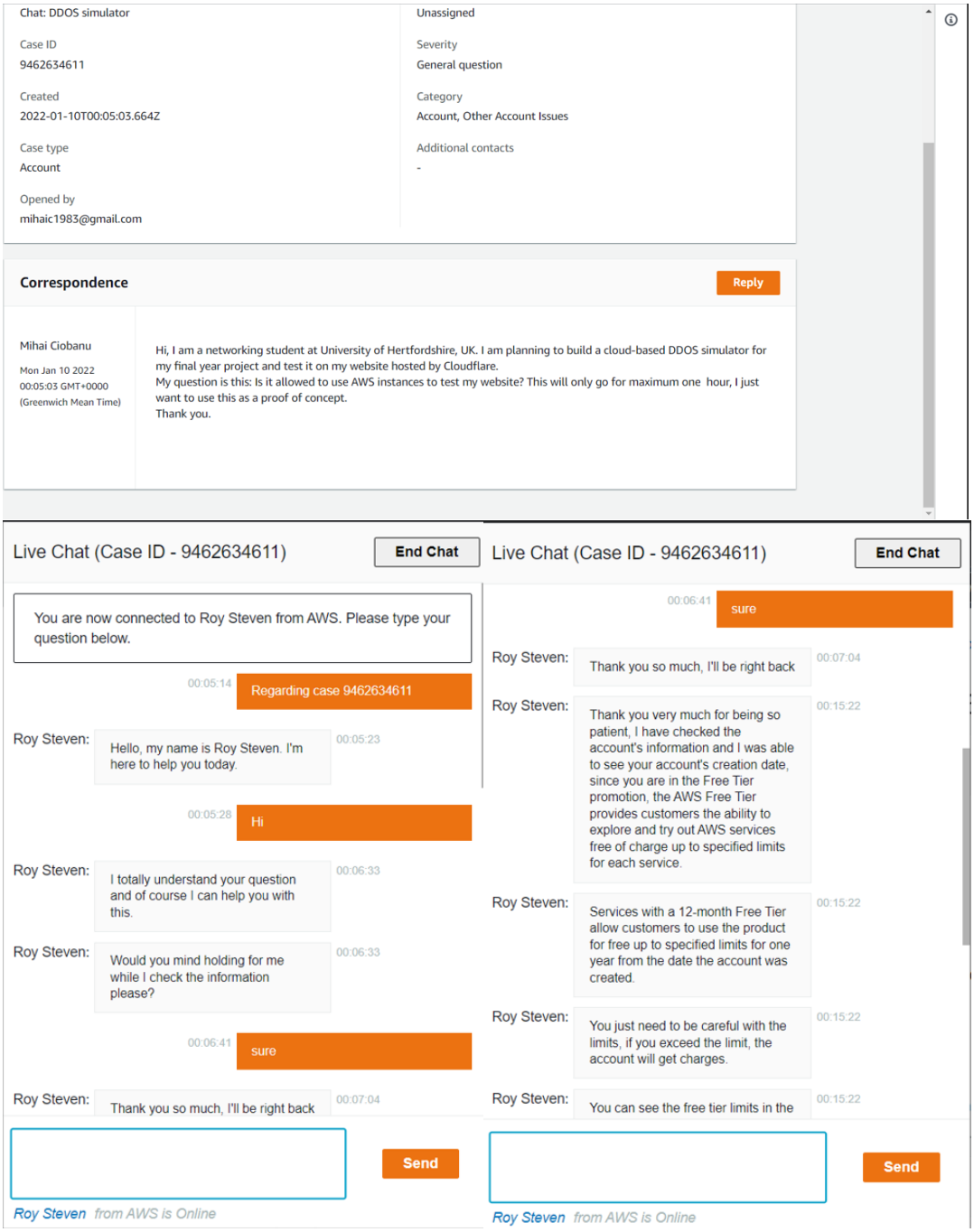

Clusters > default > Task: 3eca9f80e9e641d6b586c8cbb6c608f8
Task : 3eca9f80e9e641d6b586c8cbb6c608f8
Run more like this
Stop
Details
Tags
Logs
Last updated on March 29, 2022 7:05:57 PM (2m ago)
Filter logs
All
30s
5m
1h
6h
1d
1w
1-100
Timestamp (UTC+00:00)
Message
2022-03-29 19:01:21 □[0m □[35m [PACKET SENT TO HOST] ==> □[31m 104.21.34.206 □[35m [FLOODING PORT] ==> □[31m□[0m □[35m [PACKET SENT TO HOST] =...
2022-03-29 19:01:20 □[0m □[35m [PACKET SENT TO HOST] ==> □[31m 104.21.34.206 □[0m □[35m [PACKET SENT TO HOST] ==> □[31m 104.21.34.206 □[35m [FLO...
2022-03-29 19:01:20 □[94mbot is hammering...□[0m
2022-03-29 19:01:20 □[0m □[35m [PACKET SENT TO HOST] ==> □[31m 104.21.34.206 □[35m [FLOODING PORT] ==> □[31m 443
2022-03-29 19:01:20 □[0m □[35m [PACKET SENT TO HOST] ==> □[31m 104.21.34.206 □[35m [FLOODING PORT] ==> □[31m 443
2022-03-29 19:01:20 □[0m □[35m [PACKET SENT TO HOST] ==> □[31m 104.21.34.206 □[35m [FLOODING PORT] ==> □[31m 443
2022-03-29 19:01:20 □[94mbot is hammering...□[0m
2022-03-29 19:01:20 □[0m □[35m [PACKET SENT TO HOST] ==> □[31m 104.21.34.206 □[35m [FLOODING PORT] ==> □[31m 443
2022-03-29 19:01:20 □[0m □[35m [PACKET SENT TO HOST] ==> □[31m 104.21.34.206 □[35m [FLOODING PORT] ==> □[31m 443
2022-03-29 19:01:20 □[94mbot is hammering...□[0m
2022-03-29 19:01:20 □[0m □[35m [PACKET SENT TO HOST] ==> □[31m 104.21.34.206 □[35m [FLOODING PORT] ==> □[31m 443


mihai@kali: ~/SAPHYRA
└─$ ping ddossimulation.com
PING ddossimulation.com (104.21.34.206) 56(84) bytes of data.
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=1 ttl=128 time=32.4 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=2 ttl=128 time=11.3 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=3 ttl=128 time=11.3 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=4 ttl=128 time=12.1 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=5 ttl=128 time=11.2 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=6 ttl=128 time=11.1 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=7 ttl=128 time=12.5 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=8 ttl=128 time=11.5 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=9 ttl=128 time=12.0 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=10 ttl=128 time=10.9 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=11 ttl=128 time=30.3 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=12 ttl=128 time=12.4 ms
64 bytes from 104.21.34.206 (104.21.34.206): icmp_seq=13 ttl=128 time=11.6

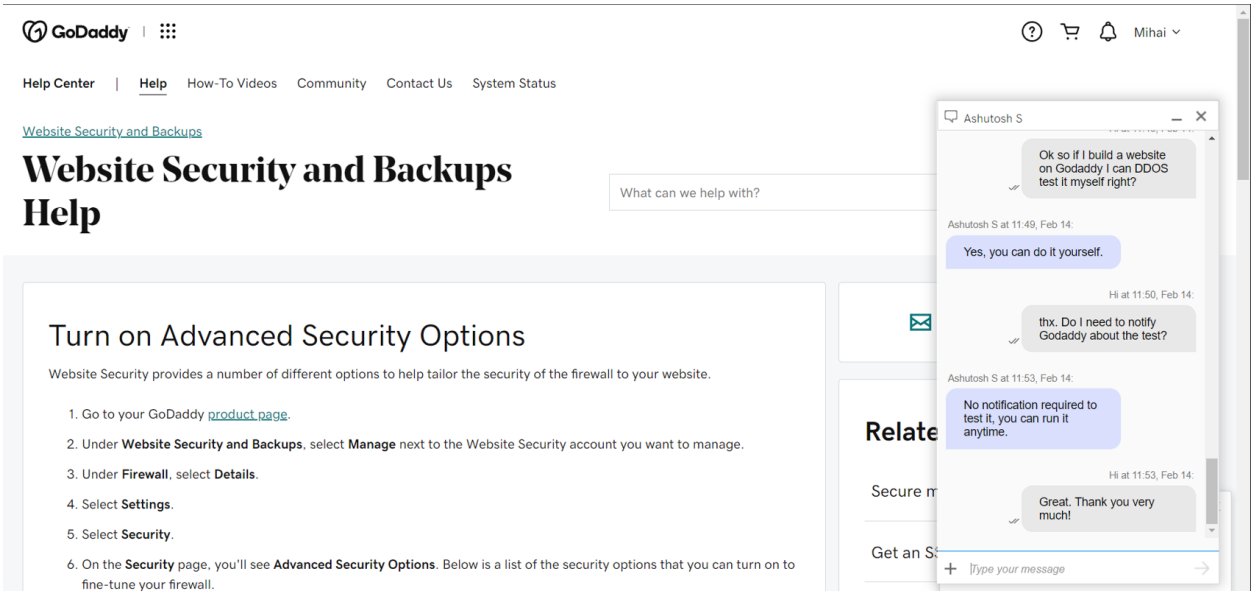
Appendix 14 – Cloudflare website traffic analytics



Appendix 13 – AWS resource usage approval



Appendix 14 - GoDaddy testing approval



Appendix 15 - Cloudflare approval was not necessary, their terms state that only opening a support ticket is required before conducting DDoS testing.