Åbo Akademi University

Cloud Computing

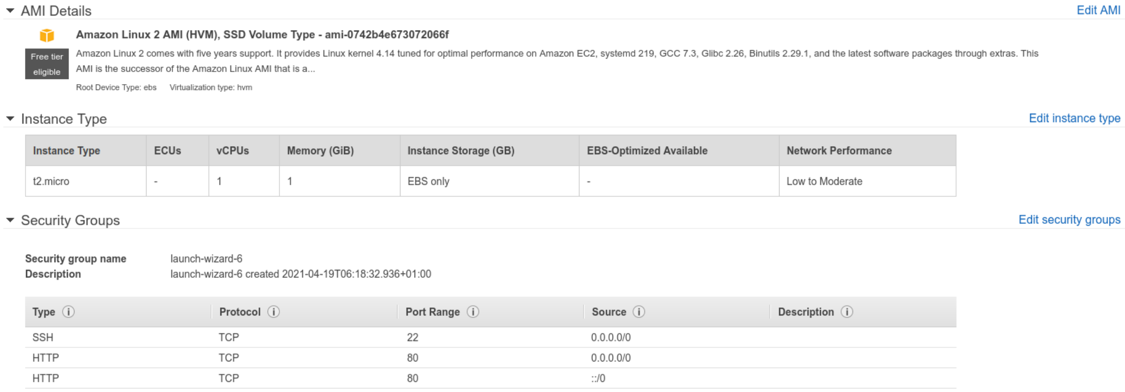
Assignment 3

Filipe Felício (2004623)

April 14, 2021

Loadbalanced service

1) Create a new instance (for example a t2.micro) using the Amazon Linux AMI



Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0742b4e673072066f

Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is a...

Root Device Type: ebs Virtualization type: hvm

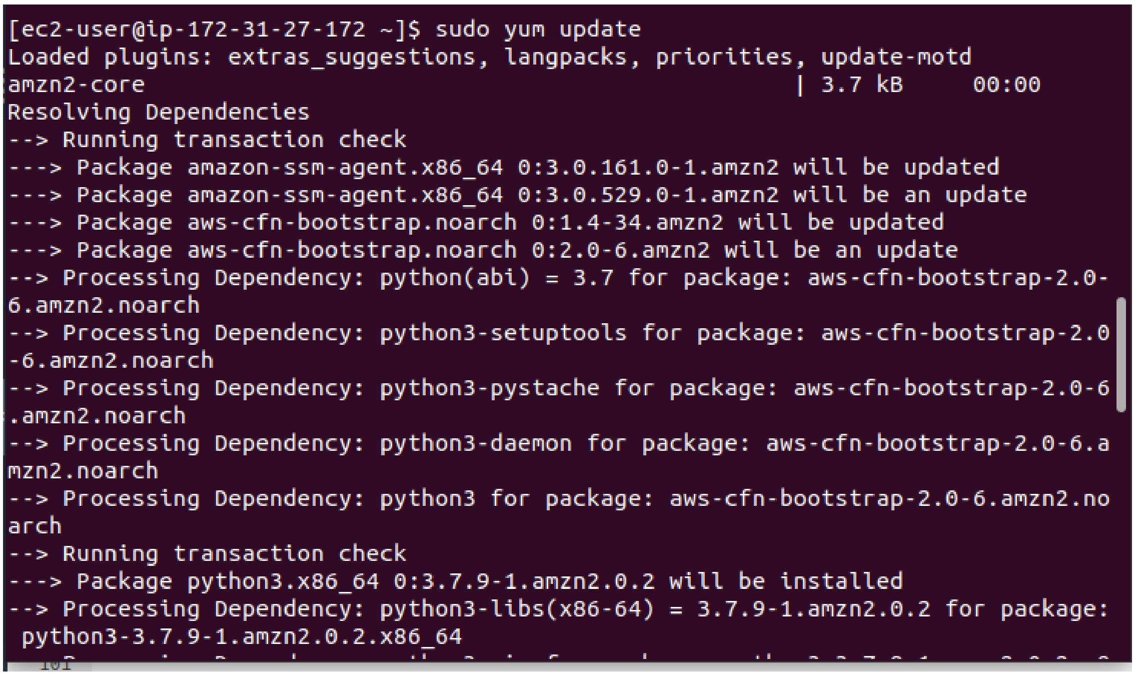
Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	-	1	1	EBS only	-	Low to Moderate

Security group name launch-wizard-6
Description launch-wizard-6 created 2021-04-19T06:18:32.936+01:00

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	0.0.0.0/0	
HTTP	TCP	80	0.0.0.0/0	
HTTP	TCP	80	::/0	

Figure 1: Creating a new instance using the Amazon Linux AMI

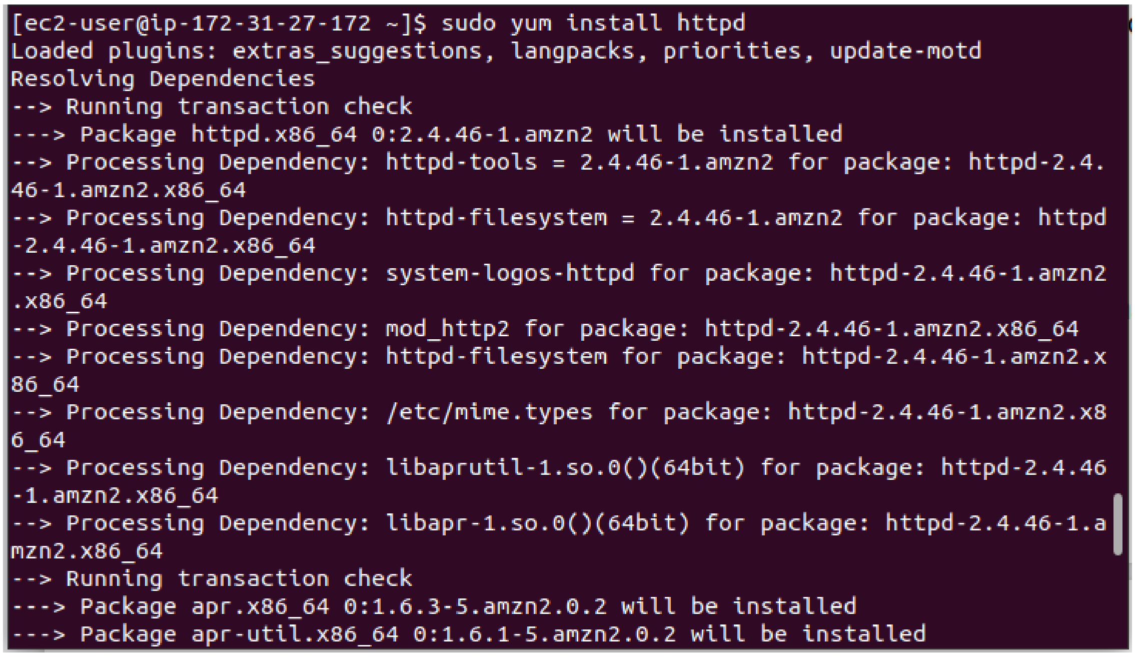
2) Update the packages of the Linux distribution



```
[ec2-user@ip-172-31-27-172 ~]$ sudo yum update
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                               | 3.7 kB     00:00
Resolving Dependencies
--> Running transaction check
---> Package amazon-ssm-agent.x86_64 0:3.0.161.0-1.amzn2 will be updated
---> Package amazon-ssm-agent.x86_64 0:3.0.529.0-1.amzn2 will be an update
---> Package aws-cfn-bootstrap.noarch 0:1.4-34.amzn2 will be updated
---> Package aws-cfn-bootstrap.noarch 0:2.0-6.amzn2 will be an update
--> Processing Dependency: python(abi) = 3.7 for package: aws-cfn-bootstrap-2.0-
6.amzn2.noarch
--> Processing Dependency: python3-setuptools for package: aws-cfn-bootstrap-2.0
-6.amzn2.noarch
--> Processing Dependency: python3-pystache for package: aws-cfn-bootstrap-2.0-6
.amzn2.noarch
--> Processing Dependency: python3-daemon for package: aws-cfn-bootstrap-2.0-6.a
mzn2.noarch
--> Processing Dependency: python3 for package: aws-cfn-bootstrap-2.0-6.amzn2.no
arch
--> Running transaction check
---> Package python3.x86_64 0:3.7.9-1.amzn2.0.2 will be installed
--> Processing Dependency: python3-libs(x86-64) = 3.7.9-1.amzn2.0.2 for package:
 python3-3.7.9-1.amzn2.0.2.x86_64
```

Figure 2: Updating the packages of the Linux distribution

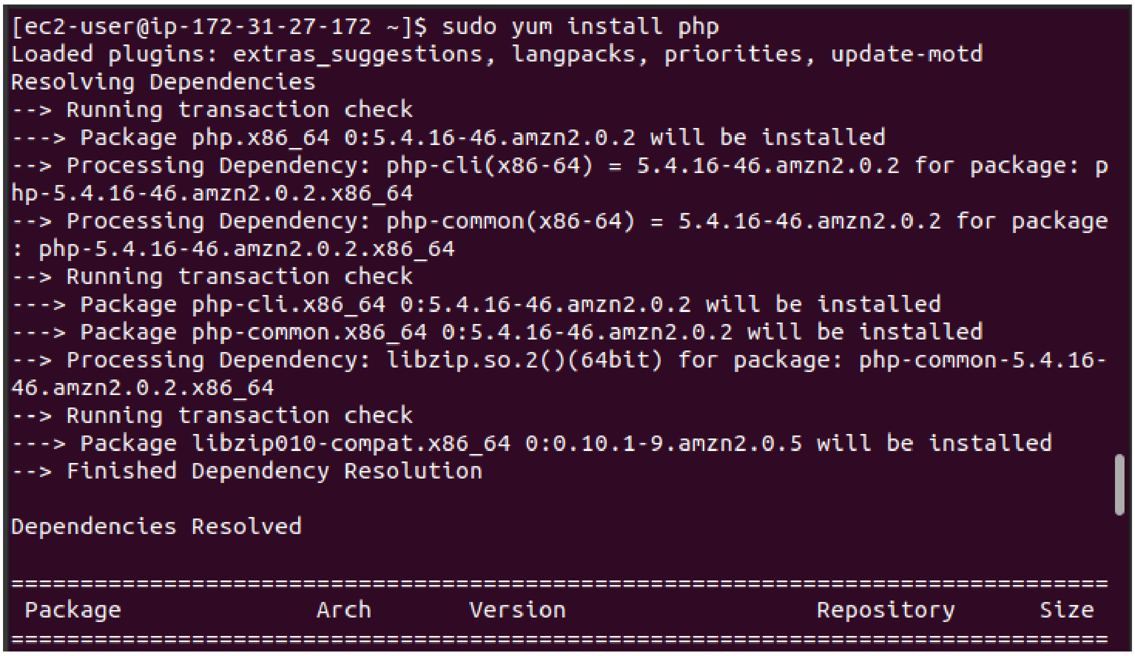
3) Install apache server on my VM



```
[ec2-user@ip-172-31-27-172 ~]$ sudo yum install httpd
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Resolving Dependencies
--> Running transaction check
---> Package httpd.x86_64 0:2.4.46-1.amzn2 will be installed
--> Processing Dependency: httpd-tools = 2.4.46-1.amzn2 for package: httpd-2.4.
46-1.amzn2.x86_64
--> Processing Dependency: httpd-filesystem = 2.4.46-1.amzn2 for package: httpd
-2.4.46-1.amzn2.x86_64
--> Processing Dependency: system-logos-httpd for package: httpd-2.4.46-1.amzn2
.x86_64
--> Processing Dependency: mod_http2 for package: httpd-2.4.46-1.amzn2.x86_64
--> Processing Dependency: httpd-filesystem for package: httpd-2.4.46-1.amzn2.x
86_64
--> Processing Dependency: /etc/mime.types for package: httpd-2.4.46-1.amzn2.x8
6_64
--> Processing Dependency: libaprutil-1.so.0()(64bit) for package: httpd-2.4.46
-1.amzn2.x86_64
--> Processing Dependency: libapr-1.so.0()(64bit) for package: httpd-2.4.46-1.a
mzn2.x86_64
--> Running transaction check
---> Package apr.x86_64 0:1.6.3-5.amzn2.0.2 will be installed
---> Package apr-util.x86_64 0:1.6.1-5.amzn2.0.2 will be installed
```

Figure 3: Installing apache server on my VM

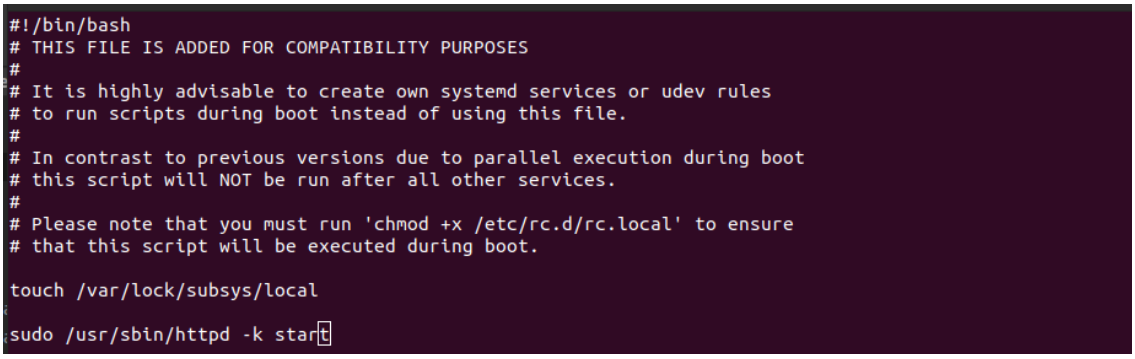
4) Install PHP



```
[ec2-user@ip-172-31-27-172 ~]$ sudo yum install php
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Resolving Dependencies
--> Running transaction check
---> Package php.x86_64 0:5.4.16-46.amzn2.0.2 will be installed
--> Processing Dependency: php-cli(x86-64) = 5.4.16-46.amzn2.0.2 for package: p
hp-5.4.16-46.amzn2.0.2.x86_64
--> Processing Dependency: php-common(x86-64) = 5.4.16-46.amzn2.0.2 for package
: php-5.4.16-46.amzn2.0.2.x86_64
--> Running transaction check
---> Package php-cli.x86_64 0:5.4.16-46.amzn2.0.2 will be installed
---> Package php-common.x86_64 0:5.4.16-46.amzn2.0.2 will be installed
--> Processing Dependency: libzip.so.2()(64bit) for package: php-common-5.4.16-
46.amzn2.0.2.x86_64
--> Running transaction check
---> Package libzip010-compat.x86_64 0:0.10.1-9.amzn2.0.5 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package                    Arch        Version                  Repository      Size
================================================================================
```

Figure 4: Installing PHP

5) Edit the file /etc/rc.local

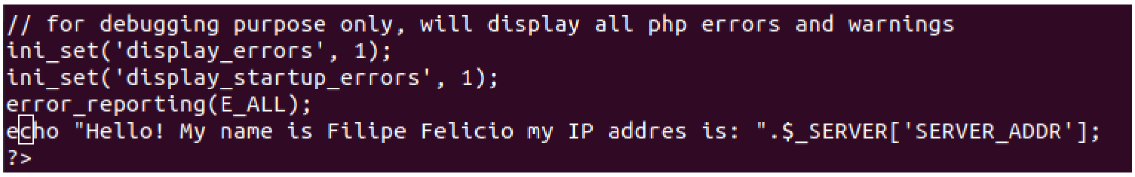


```
#!/bin/bash
# THIS FILE IS ADDED FOR COMPATIBILITY PURPOSES
#
# It is highly advisable to create own systemd services or udev rules
# to run scripts during boot instead of using this file.
#
# In contrast to previous versions due to parallel execution during boot
# this script will NOT be run after all other services.
#
# Please note that you must run 'chmod +x /etc/rc.d/rc.local' to ensure
# that this script will be executed during boot.

touch /var/lock/subsys/local

sudo /usr/sbin/httpd -k start
```

Figure 5: Editing the file /etc/rc.local

After I run this command on the terminal:
$ chmod +x /etc/rc.d/rc.local

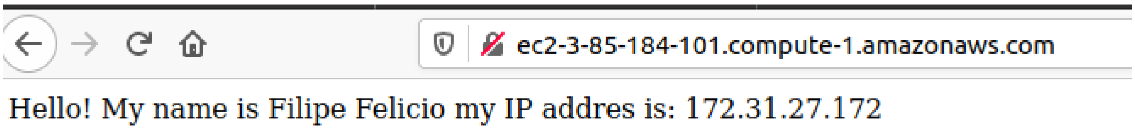
6) Create a file index.php in /var/www/html



```
// for debugging purpose only, will display all php errors and warnings
ini_set('display_errors', 1);
ini_set('display_startup_errors', 1);
error_reporting(E_ALL);
echo "Hello! My name is Filipe Felicio my IP addres is: ".$_SERVER['SERVER_ADDR'];
?>
```

Figure 6: Creating a file index.php in /var/www/html

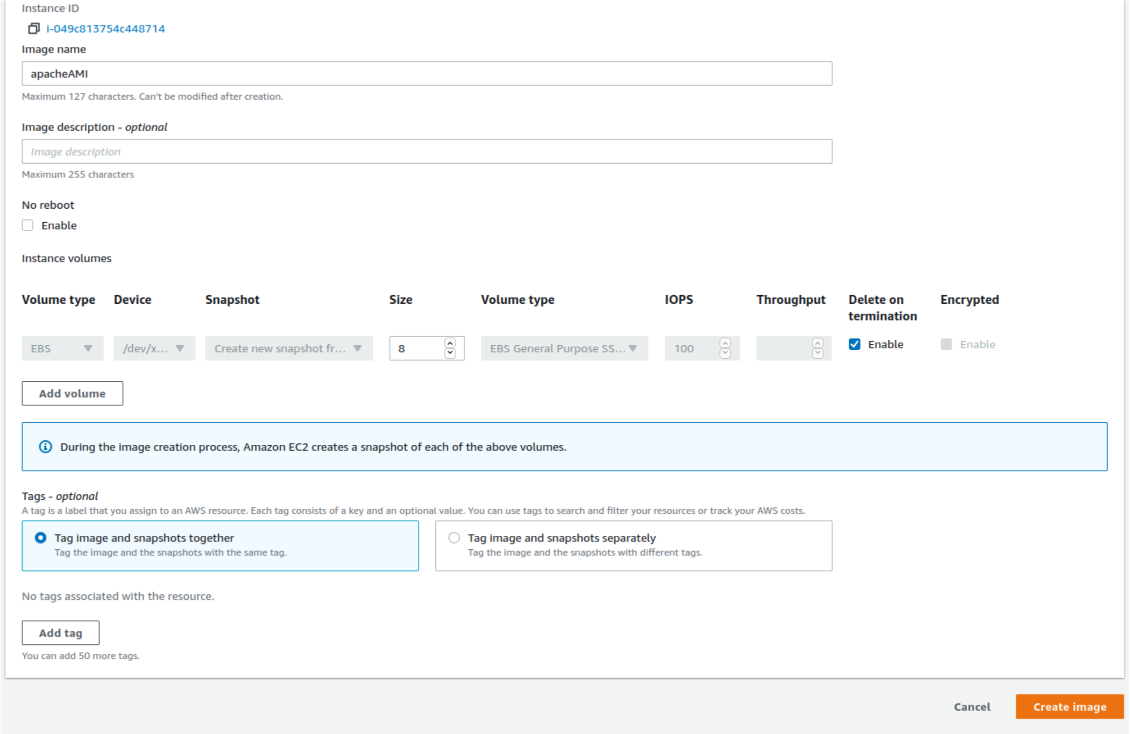
7) Start Apache (httpd) and test it can serve your index.php file



```
Hello! My name is Filipe Felicio my IP addres is: 172.31.27.172
```

Figure 7: Starting Apache (httpd) and test it can serve your index.php file

8) Create and save an AMI out of my instance running Apache



Instance ID

i-049c813754c448714

Image name

apacheAMI

Maximum 127 characters. Can't be modified after creation.

Image description - *optional*

Image description

Maximum 255 characters

No reboot

□ Enable

Instance volumes

Volume type	Device	Snapshot	Size	Volume type	IOPS	Throughput	Delete on termination	Encrypted
EBS	/dev/x...	Create new snapshot fr...	8	EBS General Purpose SS...	100		☑ Enable	□ Enable

During the image creation process, Amazon EC2 creates a snapshot of each of the above volumes.

Tags - *optional*

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

● Tag image and snapshots together
Tag the image and the snapshots with the same tag.

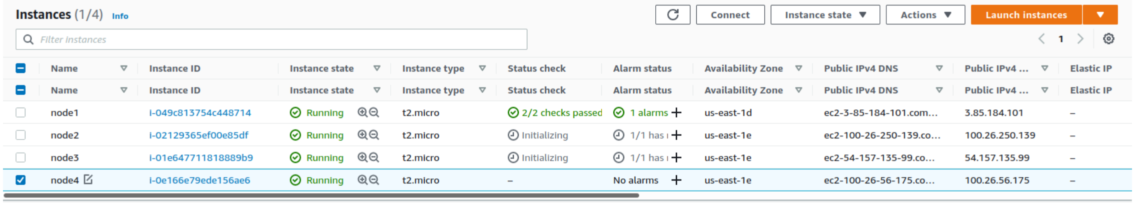
○ Tag image and snapshots separately
Tag the image and the snapshots with different tags.

No tags associated with the resource.

You can add 50 more tags.

Figure 8: Creating and saving an AMI out of my instance running Apache

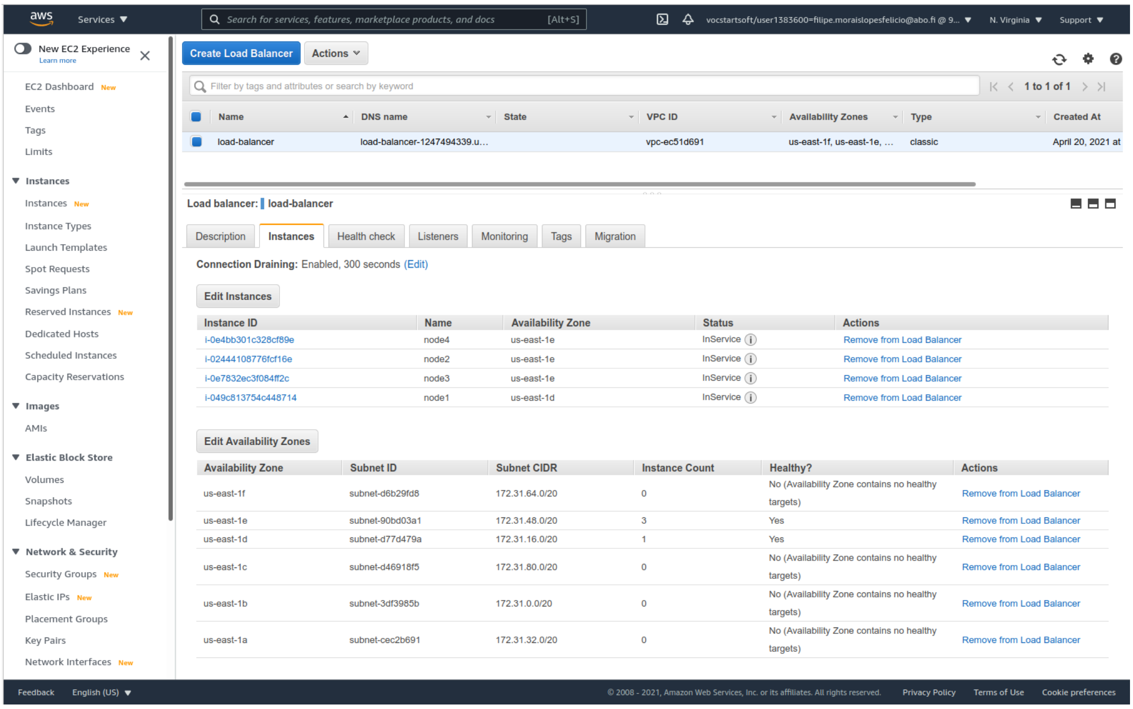
9) When you AMI is available create few (3-4) new instances instantiated with my newly created AMI



	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
□	node1	i-049c813754c448714	Running	t2.micro	2/2 checks passed	1 alarms	us-east-1d	ec2-3-85-184-101.com...	3.85.184.101	–
□	node2	i-02129365ef00e85df	Running	t2.micro	Initializing	1/1 has i	us-east-1e	ec2-100-26-250-139.co...	100.26.250.139	–
□	node3	i-01e647711818889b9	Running	t2.micro	Initializing	1/1 has i	us-east-1e	ec2-54-157-135-99.co...	54.157.135.99	–
☑	node4	i-0e166e79ede156ae6	Running	t2.micro	–	No alarms	us-east-1e	ec2-100-26-56-175.co...	100.26.56.175	–

Figure 9: Creating a few (3-4) new instances instantiated with my newly created AMI

10) Create a load balancer via the AWS console



Name	DNS name	State	VPC ID	Availability Zones	Type	Created At
load-balancer	load-balancer-1247494339.u...		vpc-ec51d691	us-east-1f, us-east-1e, ...	classic	April 20, 2021 at

Load balancer: load-balancer

Connection Draining: Enabled, 300 seconds (Edit)

Instance ID	Name	Availability Zone	Status	Actions
i-0e4bb301c328cf89e	node4	us-east-1e	InService	Remove from Load Balancer
i-02444108776fcf16e	node2	us-east-1e	InService	Remove from Load Balancer
i-0e7832ec3f084ff2c	node3	us-east-1e	InService	Remove from Load Balancer
i-049c813754c448714	node1	us-east-1d	InService	Remove from Load Balancer

Availability Zone	Subnet ID	Subnet CIDR	Instance Count	Healthy?	Actions
us-east-1f	subnet-d6b29fd8	172.31.64.0/20	0	No (Availability Zone contains no healthy targets)	Remove from Load Balancer
us-east-1e	subnet-90bd03a1	172.31.48.0/20	3	Yes	Remove from Load Balancer
us-east-1d	subnet-d77d479a	172.31.16.0/20	1	Yes	Remove from Load Balancer
us-east-1c	subnet-d46918f5	172.31.80.0/20	0	No (Availability Zone contains no healthy targets)	Remove from Load Balancer
us-east-1b	subnet-3df3985b	172.31.0.0/20	0	No (Availability Zone contains no healthy targets)	Remove from Load Balancer
us-east-1a	subnet-cec2b691	172.31.32.0/20	0	No (Availability Zone contains no healthy targets)	Remove from Load Balancer

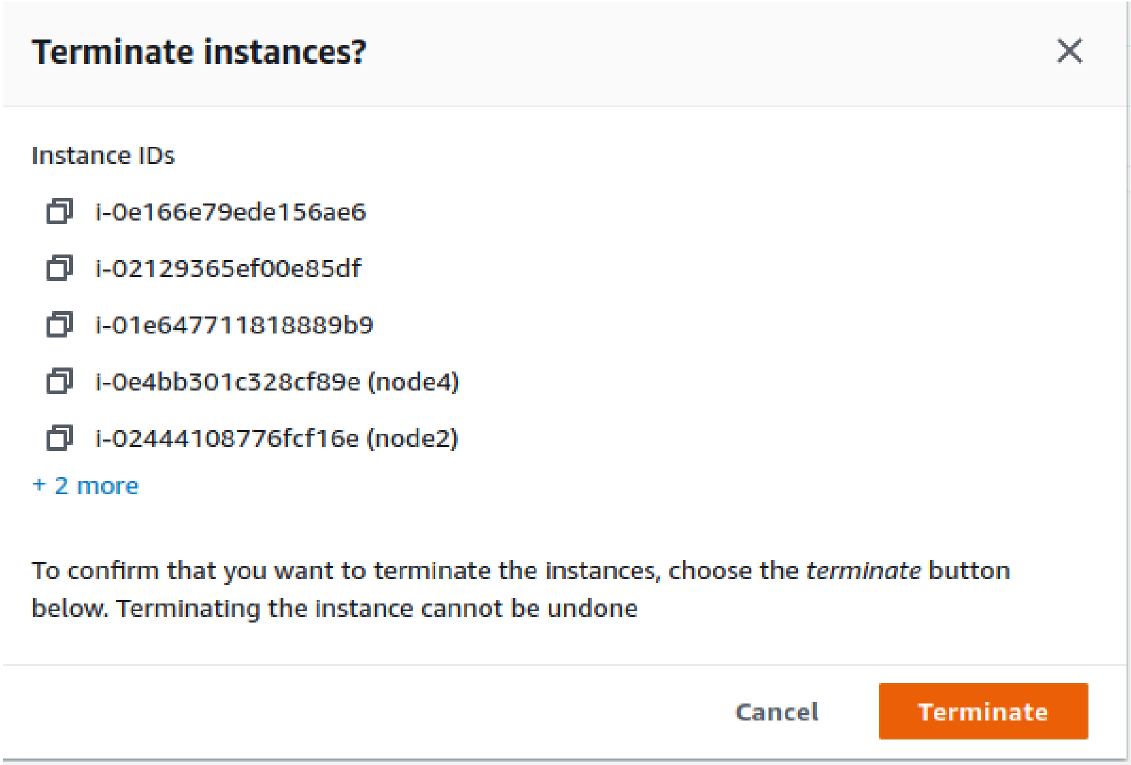
Figure 10: Creating a load balancer via the AWS console

11) Test my load balancer by pointing a browser on your local machine to the URL of the load balancer.

Link to the video:

https://www.youtube.com/watch?v=2JfsSueeFBs

12) Terminate all VMs, Terminate the load balancer and Deregister my AMI



Terminate instances?

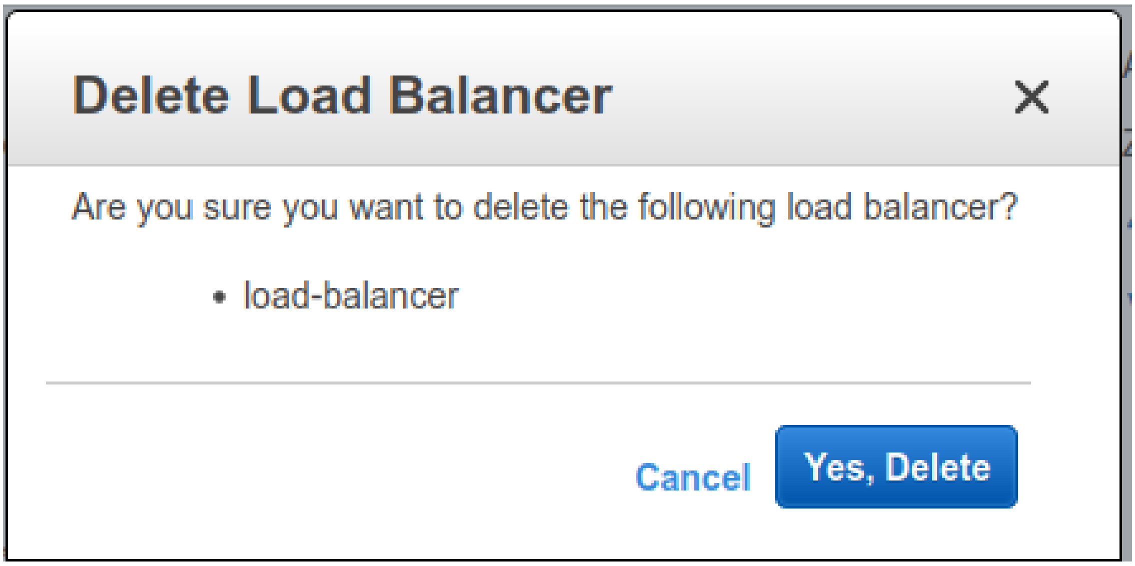
Instance IDs

- i-0e166e79ede156ae6
- i-02129365ef00e85df
- i-01e647711818889b9
- i-0e4bb301c328cf89e (node4)
- i-02444108776fcf16e (node2)

+ 2 more

To confirm that you want to terminate the instances, choose the *terminate* button below. Terminating the instance cannot be undone

Figure 11: Terminating all VMs

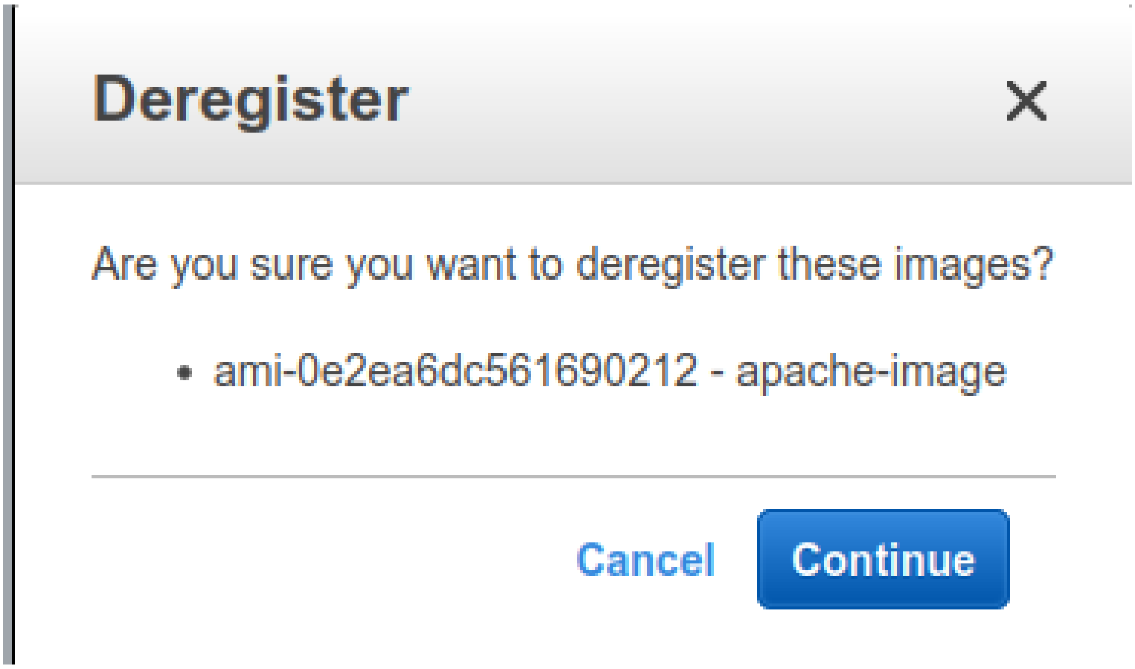


Delete Load Balancer

Are you sure you want to delete the following load balancer?

- load-balancer

Figure 12: Terminating the load balancer



Deregister

Are you sure you want to deregister these images?

- ami-0e2ea6dc561690212 - apache-image

Figure 13: Deregistering my AMI

Report

1) Why is it recommended to reboot the instance before creating the image?

Amazon EC2 powers down the instance before creating the AMI to ensure that everything on the instance is stopped and in a consistent state during the creation process. Without rebooting AWS can't guarantee the file system integrity of the created image.

2) Why does it make sense to assign the new instances in different availability zones?

If we distribute our instances across multiple Availability Zones and one instance fails, we can design our application so that an instance in another Availability Zone can handle requests. This is like an emergency load balancer without using an actual load balance. In general, AWS Availability Zones give us the flexibility to launch production apps and resources that are highly available, resilient/fault-tolerant, and scalable as compared to using a single data center.

3) Why in this context (serving a basic php file) we do not need a more complex Application Load Balancer?

In this context we do not need a more complex Application Load Balancer because of the simplicity of the requests.

4) What is going on? What happen if you terminate only one of the instances running apache?

We have a load balance managing the requests and distributing across all instances, increasing the capacity of leading with multiple requests simultaneously. If we only terminate one of the instances running apache, the Load Balancer will distribute the jobs that would be given to that instance to the others and the application would continue to run, with a lesser performance than with the instance.

Conclusion

In this assignment, like in previous I learned something completely new. It has been with great joy that I feel with each assignment that my understanding of Cloud Computing and AWS as been increasing. With this in particular, makes total sense if I want to scale the server capabilities when deploying a service in a VM using a load Balancer to distribute the work load to VM instances launched from the same AMI. This will not only increase the reliability but also the fault-tolerance because when one instance fails the others can continue running instead of the all service be unavailable. I can see myself using this skills in my professional life in a very close future. For that I am particular happy/grateful that I pick this course.