Final Project Report

Group-5

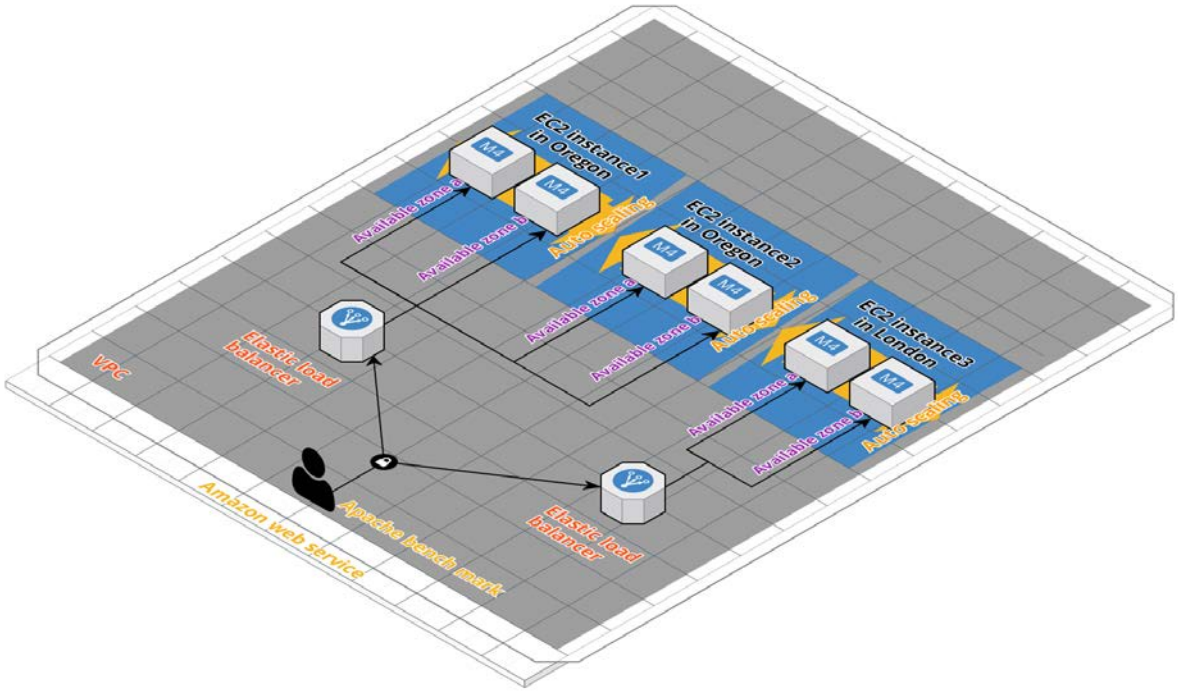
Name	E-mail	PNumber
Sri Harsha Arakatavemula	sraa16@student.bth.se	9501036991
Suman Barua	subr@student.bth.se	8210133370
Narasimha Krishna Nannuri	nana18@student.bth.se	9208309535
Bing Li	bili18@student.bth.se	8911151242
Guojun Lai	gula18@student.bth.se	9706102572

1.Architecture:

The target of this project is design and implement a high available and scalable virtual architecture. The implementation is done in AWS IaaS service and a client/server application is deployed on AWS virtual machine to build a highly scalable application.

A client/server application is developed in PHP, the functionality of the application is to display a page on the browser with some data showing which server is running. To satisfy the inter and intra data-centers load balancing, application is deployed on the VM's created. In total there are three VM's, two of them are launched on the same region Oregon in multiple availability zones and one VM on region London. Later, auto-scaling groups are created in multiple availability zones and these auto-scaling groups are connected to load balancer target groups which will be created next on two regions. Two load-balancer's are launched in either regions, one in Oregon and other in London. These target groups are connected to the load-balancer in their specific region. Cloudwatch is selected to monitor the incoming http requests from a stress tool – Apache Bench. According to the incoming http traffic the load-balancer diverts the requests to different VM's running in round-robin process. A threshold is maintained while creating the load-balancer, i.e. minimum number of EC2 instances is two, maximum number of EC2 instances is six, and auto-scaling threshold is 1000 requests. If there is large number of incoming traffic, auto-scaling is done which in turn spins new instances (VM's), this is called as auto-scaling-in. In case of less incoming traffic, newly created instances are terminated and maintains the initial number of instances as mentioned, this is called as auto-scaling-out. All the auto-scaling information is monitored by cloudwatch and can be shown in the form of graphs.

The following figure explains about the architecture of this scalable elastic application.



The terms like auto-scaling, load-balancing, Apache bench, target groups e.c.t will be explained in the following:

Load-balancer:

An Elastic Load Balancing which automatically distributes the incoming application traffic across multiple EC2 instances created in multiple availability zones. And Elastic Load Balancing offers three types of load balancers that all feature the high availability, automatic scaling, and robust security necessary to make the applications fault tolerant. In our implementation, we use Application Load Balancer, which is best suited for load balancing of HTTP and HTTPS traffic. When operating at the individual request, Application Load Balancer routes traffic to targets within Amazon Virtual Private Cloud (Amazon VPC) based on the content of request.

Auto-scaling:

Auto-scaling keeps the track of all the instances or applications running and easy to scale instances in and out according to the stress or incoming traffic on the specific instance.

A minimum and maximum number of instances to be run are selected during auto-scaling configuration. Auto-scaling group is created for each VM launched. According to the mentioned minimum and maximum number of EC2 instances, the auto-scaling id performed. If the traffic of more then, auto-scaling in is performed, if the traffic or stress is low then auto-

scaling out is performed. The number of instances doesn't exceed the mentioned number of instances in the launch configuration. For each EC2 instance, auto-scaling group is created to configure the security group.

Target groups:

Each target group is created to direct the requests to different targets registered. When each listener rule created, target group and the conditions must be specified. The traffic is forwarded to the corresponding target group when the rule is met. This target group is routed to the auto-scaling group.

Apache bench:

Apache Bench is a load testing and benchmarking tool for a Hypertext Transfer Protocol (HTTP) web server. With this tool, we can quickly know how many requests per second the web server is capable of serving. In our task, we use it to generate the load to the load balancer.

2.Implementation

1. Intra datacenters

The implementation of the infrastructure and deployment of the application in the same region will be explained below.

Step 1:

Two EC2 instance are launched in Oregon region at different availability zones with ubuntu18.04 as platform. EC2 instance console management page displays the instance running.

vm1	i-0109320325d15c201	t2.micro	us-west-2a	running	2/2 checks ...	None
vm2	i-0a1ba63928df42164	t2.micro	us-west-2b	running	2/2 checks ...	None

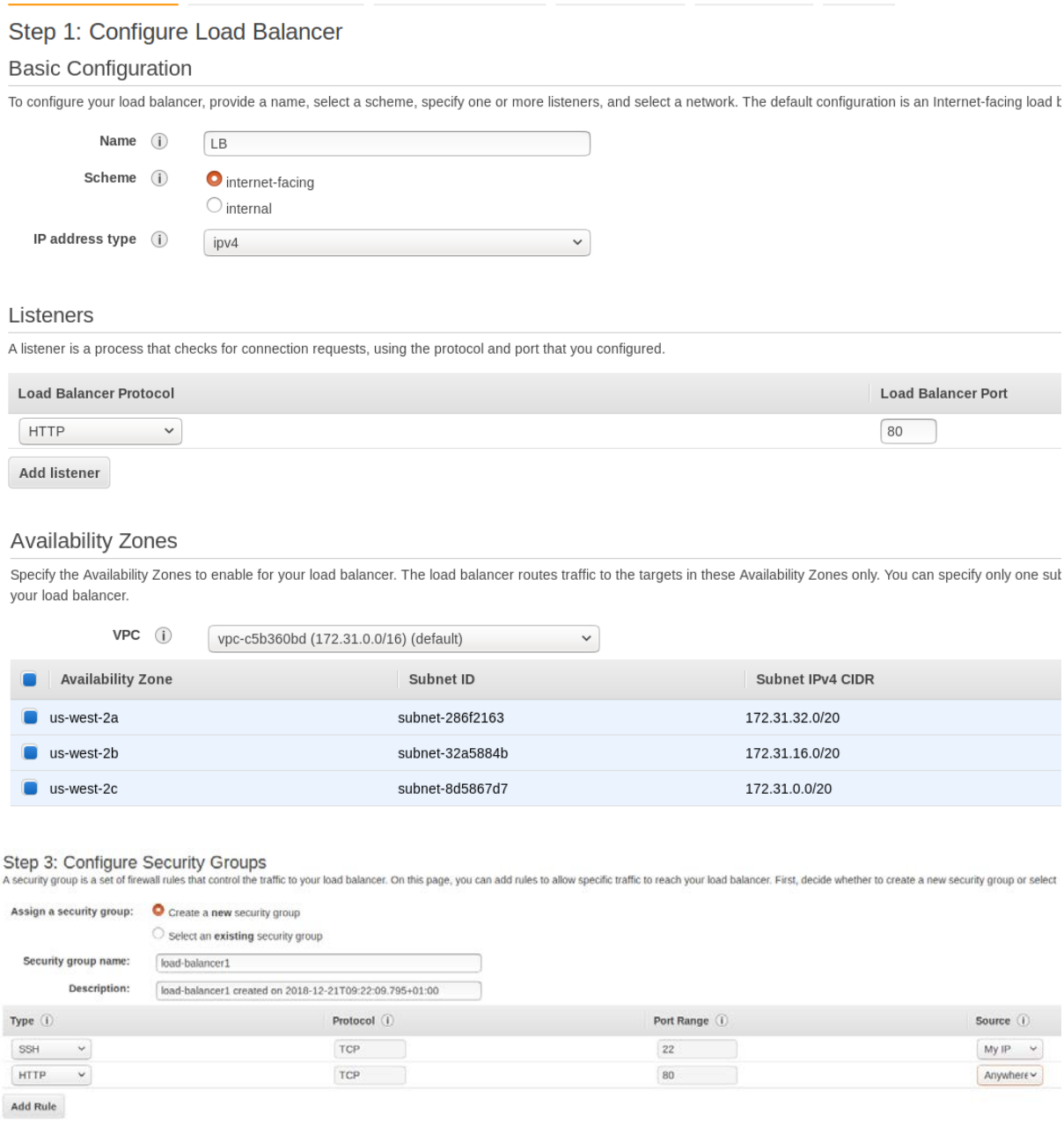
Step 2:

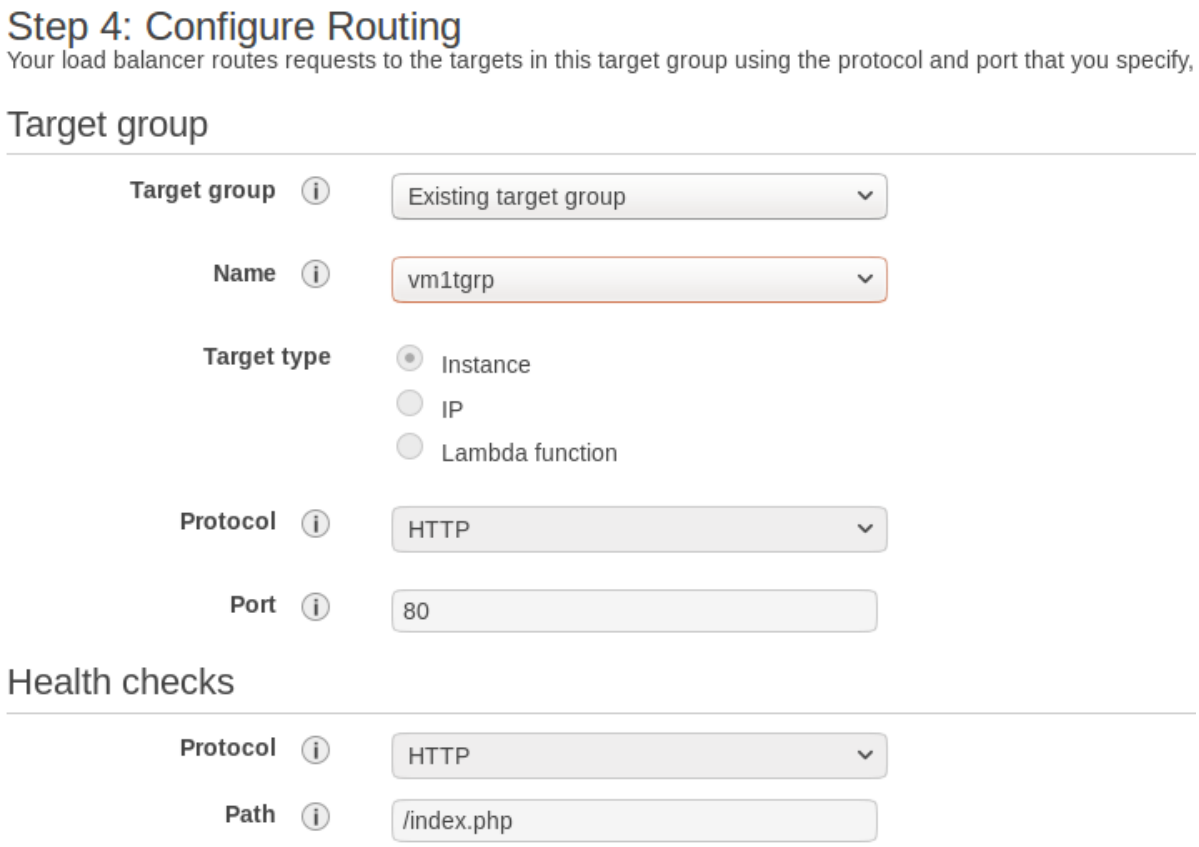
For the running EC2 instances an image (ami – amazon machine image) is created. This image is the exact copy of the original EC2 instance running.

vm1ami	ami-06ec44788fa601ba3	451968294493/v...	451968294493	Private	available
vm2ami	ami-0ae60dc6315d132ab	451968294493/v...	451968294493	Private	available

Step 3:

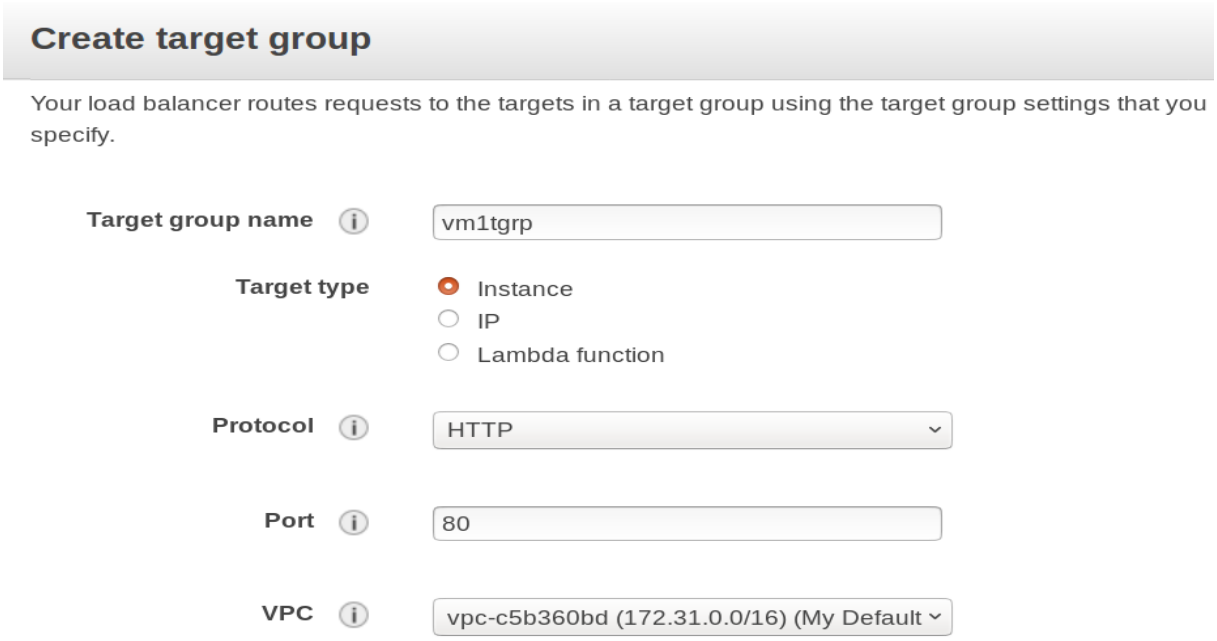
A single Load balancer is launched for the instances running.





Step 4:

A target group is created for each EC2 instance running.



Health check settings

Protocol: HTTP

Path: /index.php

Advanced health check settings

Port: traffic port / override

Healthy threshold: 5

Unhealthy threshold: 2

Timeout: 5 seconds

Create target group | Actions

Filter by tags and attributes or search by keyword

Name	Port	Protocol	Target type	Load Balanc	VPC ID
vm1tgrp	80	HTTP	instance		vpc-c5b360bd
vm2tgrp	80	HTTP	instance		vpc-c5b360bd

Step 5:

Launch configuration is made for both instances running to configure the auto-scaling. Once the launch configuration is made, it cannot be changes later.

Create Launch Configuration

Review the details of your launch configuration. You can go back to edit the details of each section before you finish.

AMI Details

vm1ami - ami-06ec44788fa601ba3

Root device type: ebs Virtualization Type: hvm

Instance Type

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

Launch configuration details

Name	vm1as
Purchasing option	On demand
EBS Optimized	No
Monitoring	Yes
IAM role	None
Tenancy	Shared tenancy (multi-tenant hardware)
Kernel ID	Use default
RAM Disk ID	Use default
User data	
IP Address Type	Only assign a public IP address to instances launched in the default VPC and subnet. (default)

Storage

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput	Delete on Termination	Encrypted
Root	/dev/sda1	snap-067bb0a25341d	8	gp2	N/A	N/A	Yes	No

Security Groups

Security group name AutoScaling-Security-Group-3
Description AutoScaling-Security-Group-3 (2018-12-21 09:28:15.019+01:00)

Type	Protocol	Port Range	Source
SSH	TCP	22	80.78.218.18/32
HTTP	TCP	80	0.0.0.0/0

Step 6:

Auto-scaling groups are created for each launch configuration of specific instance.

In this configuration, we set that the number of minimum instances is 2, the maximum is 6, which means there can be a minimum of 2 instances running and a maximum of 6 instances depending upon the load generated.

Create Auto Scaling Group

Please review your Auto Scaling group details. You can go back to edit changes for each section. Click **Create Auto Scaling group** to complete the creation of an Auto Scaling group.

Auto Scaling Group Details

Group name	vm2asgrp
Group size	2
Minimum Group Size	2
Maximum Group Size	6
Subnet(s)	subnet-32a5884b,subnet-286f2163
Load Balancers	
Target Groups	vm2tgrp
Health Check Type	EC2
Health Check Grace Period	100
Detailed Monitoring	Yes
Instance Protection	None
Service-Linked Role	AWSServiceRoleForAutoScaling

Scaling Policies

Scale Group Size	Maintain metric type Application Load Balancer Request Count Per Target at target value 1000, with 30 seconds for instances to warm up.

Now, in the EC2 instance console management page totally six instances are running. Two of them are Original Ec2 instance launched and other four are EC2 instance created by auto-scaling groups. Copy the DNS of load balancer and run it on the browser. The application is running with the server number from which it is running. Try reloading the page, now different server number will be displayed, meaning running from different a server in different availability zone.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Statu
vm1	i-0109320325d15c201	t2.micro	us-west-2a	running	2/2 checks ...	None
	i-01e14054088f6b730	t2.micro	us-west-2a	running	2/2 checks ...	None
in2	i-06742dddaa973846d	t2.micro	us-west-2a	stopped		None
vm2	i-0a1ba63928df42164	t2.micro	us-west-2b	running	2/2 checks ...	None
	i-0b14396d02a2c96b4	t2.micro	us-west-2b	terminated		None
	i-0bf2b7875c43b3c74	t2.micro	us-west-2b	running	2/2 checks ...	None
	i-0c057c3235f5b486f	t2.micro	us-west-2a	terminated		None
	i-0c9454bb12cd32409	t2.micro	us-west-2b	running	2/2 checks ...	None
	i-0e732eaf554747ee5	t2.micro	us-west-2a	running	2/2 checks ...	None

Step 7:

A stress tool – Apache bench is installed on VM1 and VM2. A stress is created from VM2 in the region Oregon with apache bench running to VM1 which is in the same region like VM2.

```
ubuntu@ip-172-31-26-120:~$ sudo ab -n 80000 -c 60 http://loadb-973245738.us-west
-2.elb.amazonaws.com/index.php
```

Step 8:

When the high stress automatically generated, new instances are created called as scale in and can be seen running in EC2 instance console.

vm1	i-0109320325d15c201	t2.micro	us-west-2a	running	2/2 checks ...	None
vm2asg1	i-01746ff11f08059c0	t2.micro	us-west-2a	running	2/2 checks ...	None
	i-01e14054088f6b730	t2.micro	us-west-2a	terminated		None
vm2asg2	i-025646282530d1a41	t2.micro	us-west-2b	running	2/2 checks ...	None
vm1asg4	i-04785ee1040ad6d08	t2.micro	us-west-2a	running	2/2 checks ...	None
vm1asg3	i-05703d88010e99742	t2.micro	us-west-2b	running	2/2 checks ...	None
vm1asg1	i-066bfa061c14f5ddb	t2.micro	us-west-2b	running	2/2 checks ...	None
in2	i-06742dddaa973846d	t2.micro	us-west-2a	stopped		None
vm1asg5	i-06d847ab4ce731720	t2.micro	us-west-2c	running	2/2 checks ...	None
vm2	i-0a1ba63928df42164	t2.micro	us-west-2b	running	2/2 checks ...	None
vm1asg2	i-0bcc90899fff1b793	t2.micro	us-west-2c	running	2/2 checks ...	None
	i-0bf2b7875c43b3c74	t2.micro	us-west-2b	terminated		None
	i-0c9454bb12cd32409	t2.micro	us-west-2b	terminated		None
	i-0e732eaf554747ee5	t2.micro	us-west-2a	terminated		None
vm1asg6	i-0f0e1b8d6ab2dcff8	t2.micro	us-west-2a	running	2/2 checks ...	None

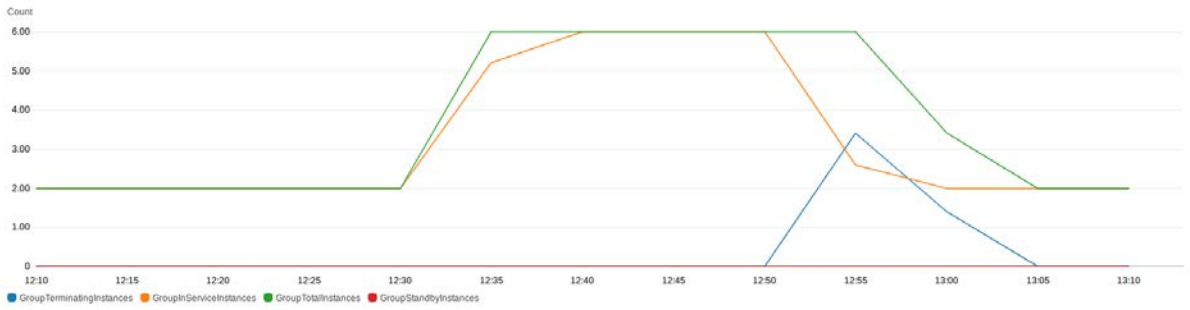
When the stress is decreased automatically, newly generated instances are terminated, which can be called scale out.

vm1asg4	i-04785ee1040ad6d08	t2.micro	us-west-2a	terminated
vm1asg3	i-05703d88010e99742	t2.micro	us-west-2b	terminated
vm1asg1	i-066bfa061c14f5ddb	t2.micro	us-west-2b	terminated
vm1asg2	i-0bcc90899fff1b793	t2.micro	us-west-2c	terminated

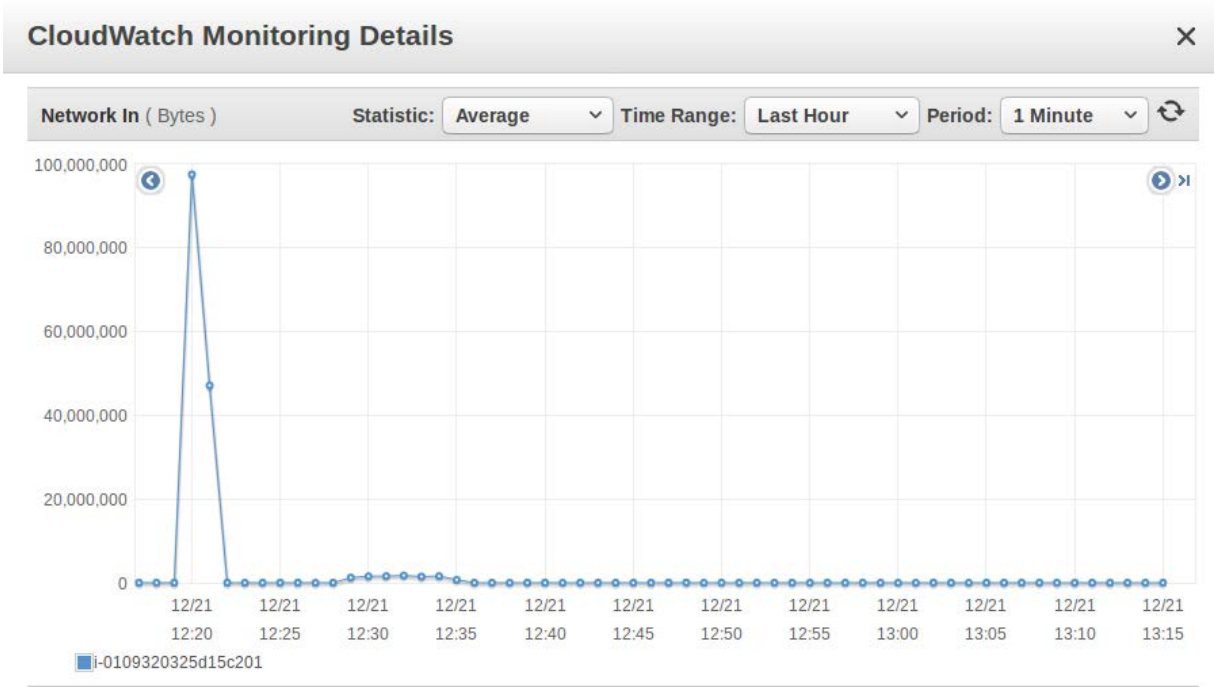
Step 9:

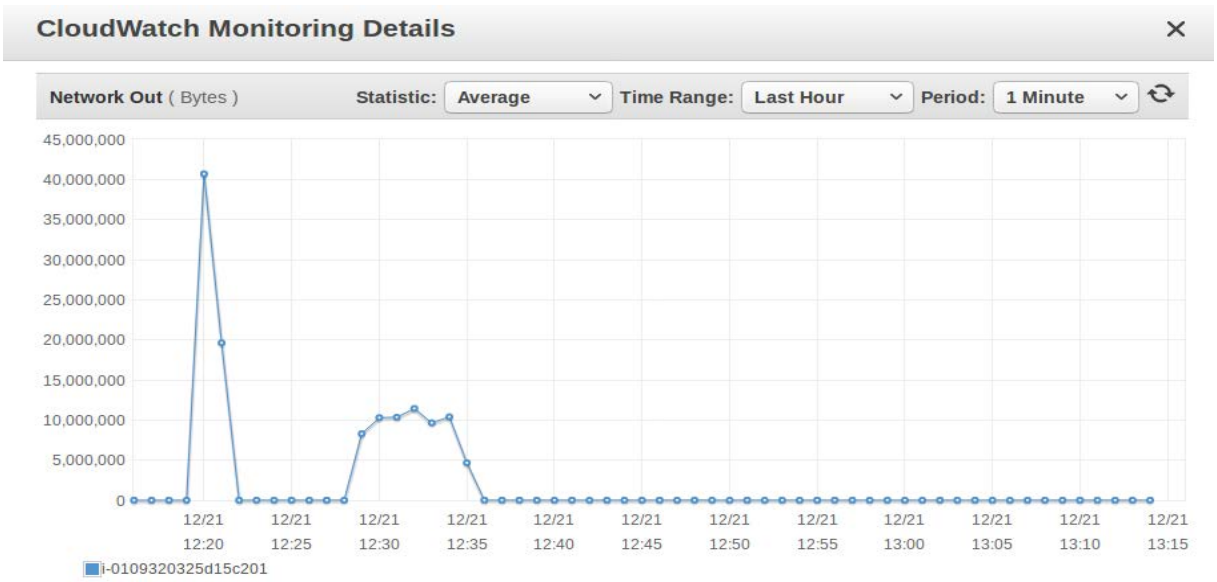
Cloudwatch monitors the traffic and the instances spinning which can be seen in the form of graphs.

This graph shows changes of the instances of the auto-scaling group vm1asgrp from metrics GroupTeminatingInstances, GroupServiceInstance, and GroupTotalInstance.



These two graphs shows the traffic on the vm1.





2. Inter datacenters

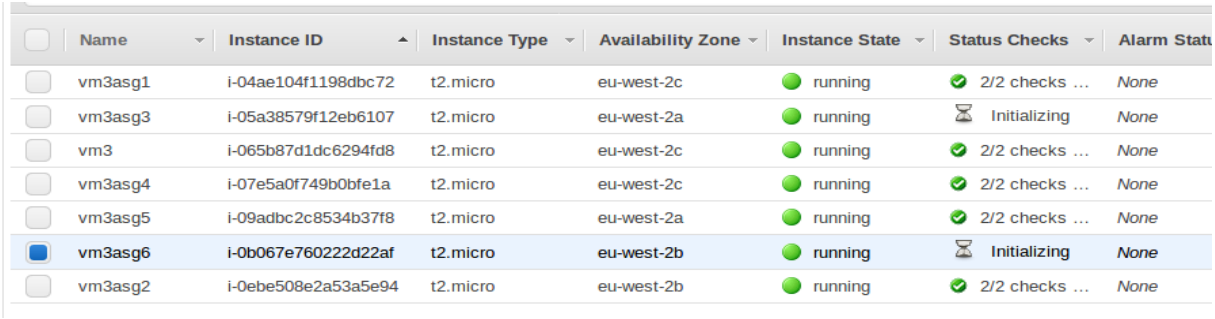
The implementation of the infrastructure and deployment of the application in another region London will be explained below.

Now, region London is selected. The same process from step 1 to step 6 above is repeated with only one VM running, and we call this virtual machine VM3. Thus, we create the auto-scaling elastic load-balancing application on VM3 in Amazon Web Services.

Then we use Apache bench to generate the workload from VM2 in the region Oregon to VM3 in the region London.

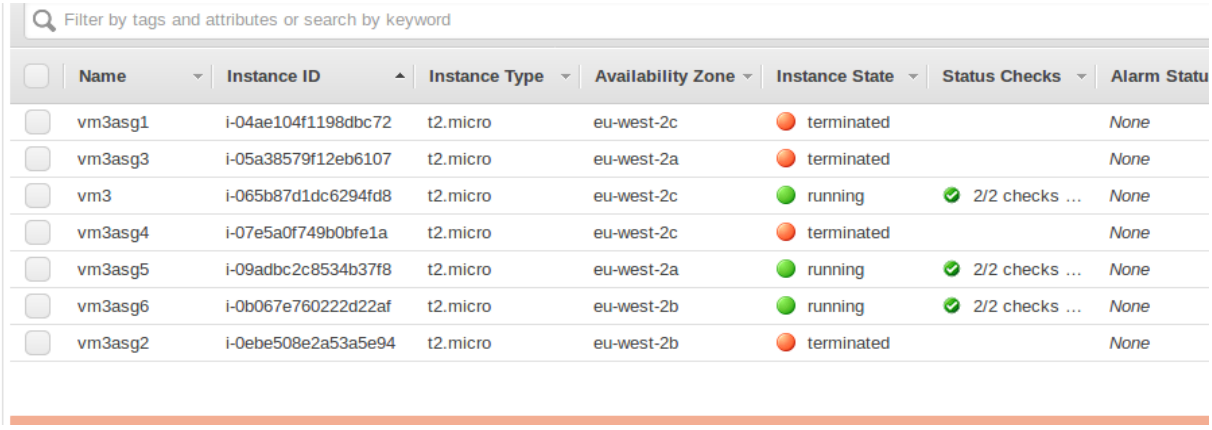
```
ubuntu@ip-172-31-26-120:~$ sudo ab -n 50000 -c 50 http://LB-954460869.eu-west-2.
elb.amazonaws.com/index.php
```

A high stress automatically generated, since there are load-balancer and auto scale, new instances are created and can be seen running in EC2 instance console.



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Statu
vm3asg1	i-04ae104f1198dbc72	t2.micro	eu-west-2c	running	2/2 checks ...	None
vm3asg3	i-05a38579f12eb6107	t2.micro	eu-west-2a	running	Initializing	None
vm3	i-065b87d1dc6294fd8	t2.micro	eu-west-2c	running	2/2 checks ...	None
vm3asg4	i-07e5a0f749b0bfe1a	t2.micro	eu-west-2c	running	2/2 checks ...	None
vm3asg5	i-09adbc2c8534b37f8	t2.micro	eu-west-2a	running	2/2 checks ...	None
vm3asg6	i-0b067e760222d22af	t2.micro	eu-west-2b	running	Initializing	None
vm3asg2	i-0ebe508e2a53a5e94	t2.micro	eu-west-2b	running	2/2 checks ...	None

After the stress is decreased automatically, newly generated instances are terminated,

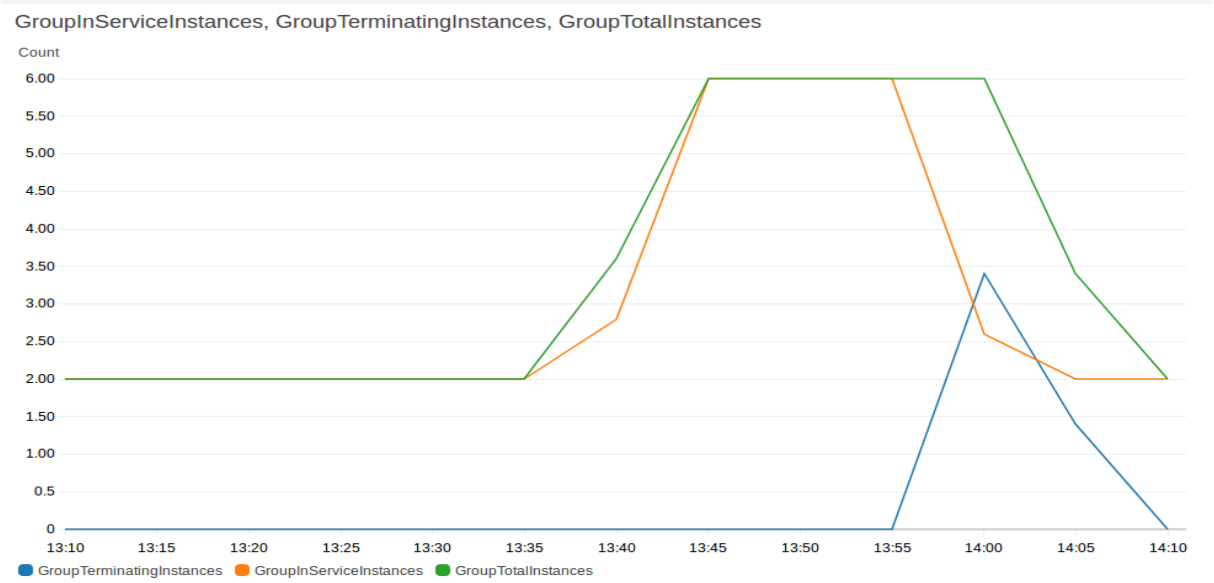


Filter by tags and attributes or search by keyword

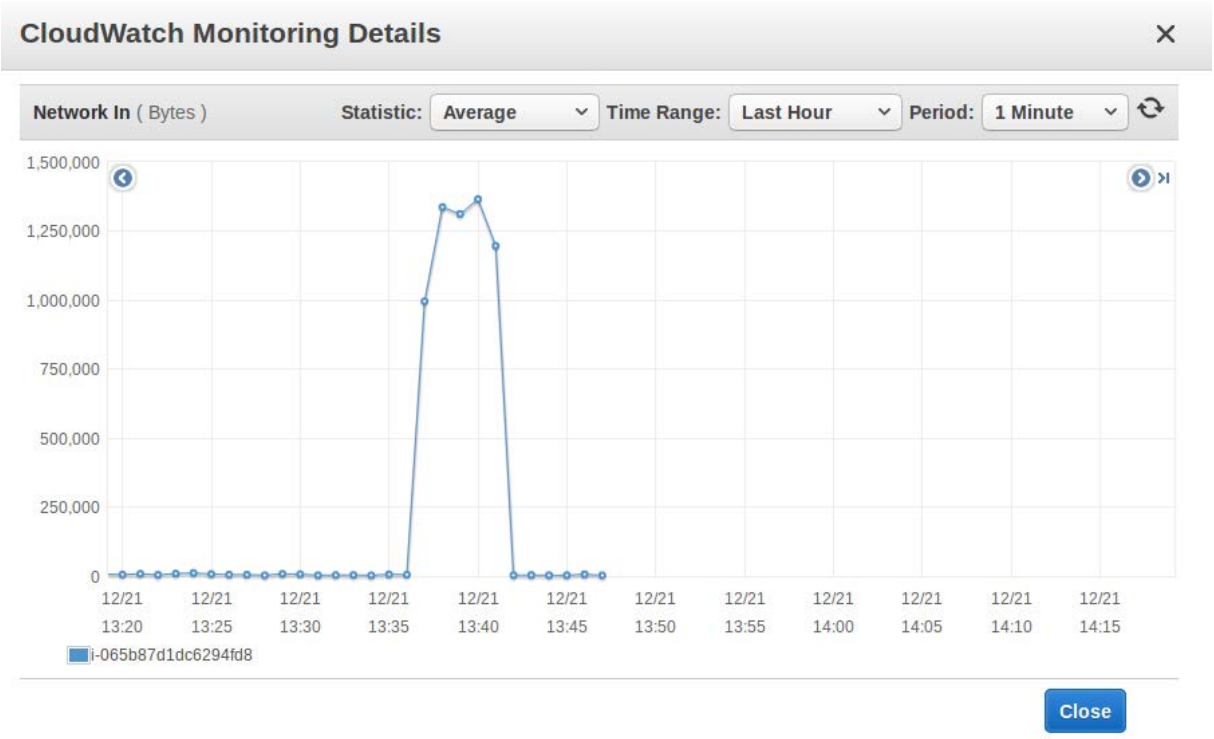
Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Statu
vm3asg1	i-04ae104f1198dbc72	t2.micro	eu-west-2c	terminated		None
vm3asg3	i-05a38579f12eb6107	t2.micro	eu-west-2a	terminated		None
vm3	i-065b87d1dc6294fd8	t2.micro	eu-west-2c	running	2/2 checks ...	None
vm3asg4	i-07e5a0f749b0bfe1a	t2.micro	eu-west-2c	terminated		None
vm3asg5	i-09adbc2c8534b37f8	t2.micro	eu-west-2a	running	2/2 checks ...	None
vm3asg6	i-0b067e760222d22af	t2.micro	eu-west-2b	running	2/2 checks ...	None
vm3asg2	i-0ebe508e2a53a5e94	t2.micro	eu-west-2b	terminated		None

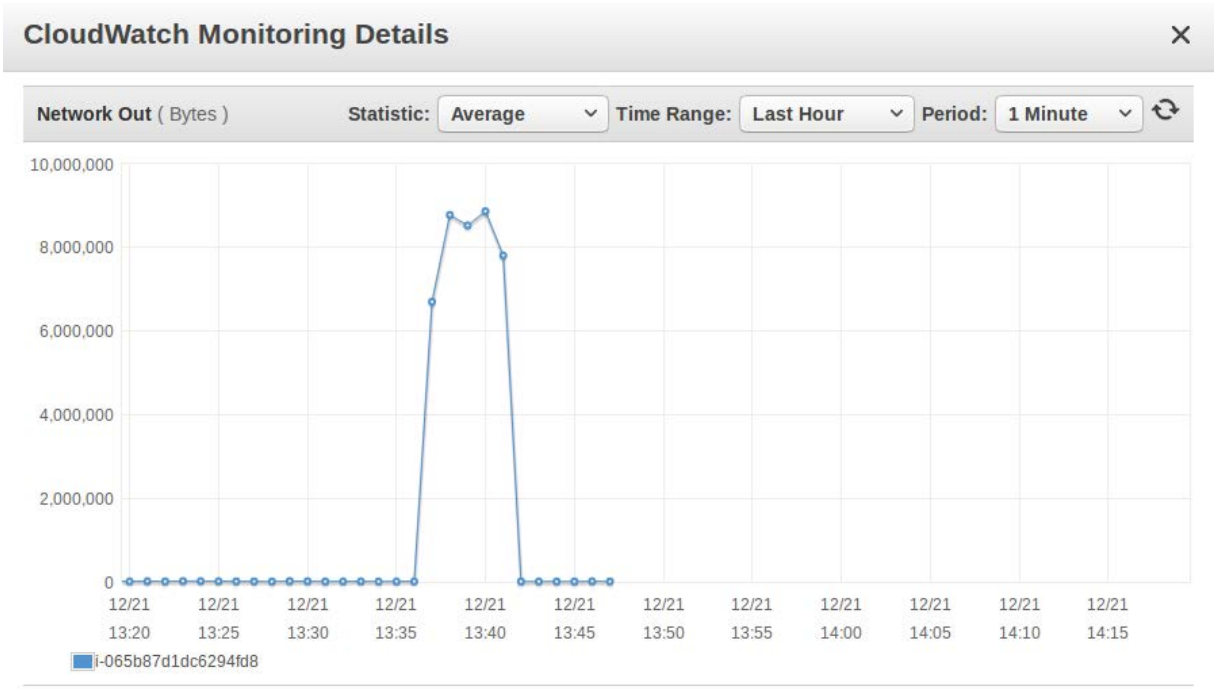
Cloudwatch monitors the traffic and the instances spinning which can be seen in the form of graphs.

This graph shows changes of the instances of the auto-scaling group vm3asgrp from metrics GroupTeminatingInstances, GroupServiceInstance, and GroupTotalInstance.



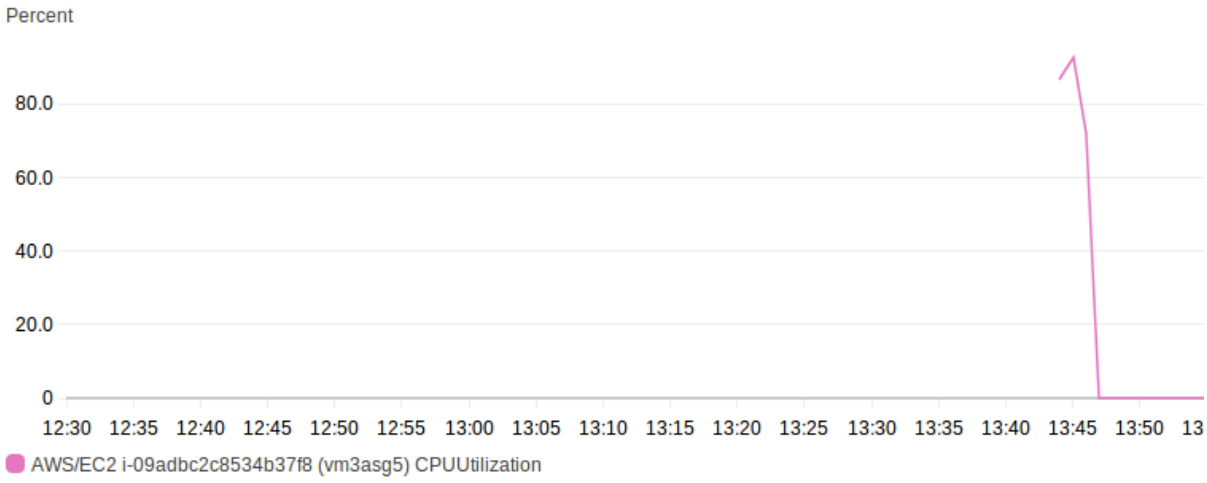
These two graphs shows the traffic on the vm3.





3. The application is elastic

Graphs below indicate that the application is elastic. After a high workload is generated to VM3, a new instance of the application is created, which is named vm3asg5. From its CPUUtilization and NetworkIn metric graphs, we know that it could manage the increased load immediately.



4. Self-healing

This infrastructure supports self-healing. Because after terminating one instance, it generates a new instance immediately.

		i-0f02b670534b876fe	t2.micro	us-west-2b	pending	Initializing	None	ec2-5
☐	vm2asg5	i-038e6605d48af94e7	t2.micro	us-west-2c	terminated			None
☐	in2	i-06742dddaa973846d	t2.micro	us-west-2a	stopped			None
☐	vm2asg6	i-06a6f6966efdc0d50	t2.micro	us-west-2a	running	2/2 checks ...		None
☐	vm1asg5	i-06d847ab4ce731720	t2.micro	us-west-2c	running	2/2 checks ...		None
☐	vm2	i-0a1ba63928df42164	t2.micro	us-west-2b	running	2/2 checks ...		None
☐		i-0f02b670534b876fe	t2.micro	us-west-2b	running	Initializing		None
☐	vm1asg6	i-0f0e1b8d6ab2dcff8	t2.micro	us-west-2a	running	2/2 checks ...		None

Consequently, when accidents or failures happening, it can restart automatically.