DV1566- Project report

Group-49

Sai Nikhil Boyapati

Tabu Sravani Guduru

Problem Description:

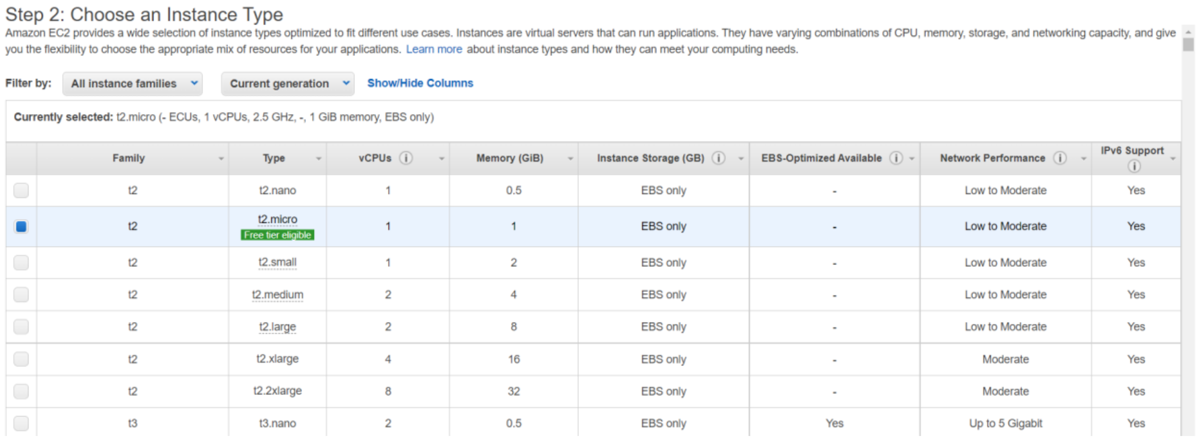
Scalability is one of the major concerns while deploying the application. The server should be automatically able to scale in and out depending upon the no of requests are made. If it could not scale in when there are less no of requests, then it leads to the wastage of resources, and money and if a server could not scale out, it results in disappointed clients and bad server errors.

Implementation Technique:

The solution is implemented on AWS cloud. The services that we will be using to test the solution are EC2 instances, auto scaling group, and Load Balancer. The application we chose to deploy on the EC2 instance is a simple Django application.

Procedure:

The initial step is to login to the AWS management console and try to find the ec2 instance. We have setup the OS type to ubuntu, and we have launched the VMs. The default VPC and subnets are selected.

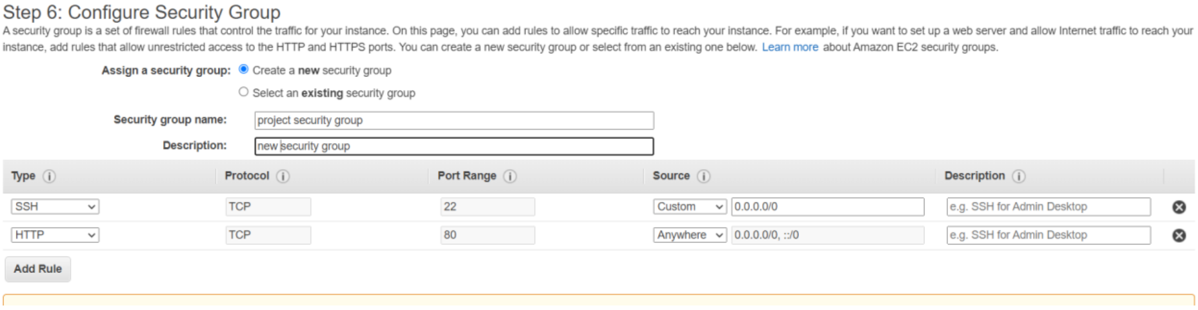


Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, -, 1 GiB memory, EBS only)

	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
□	t3	t3.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit	Yes



Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

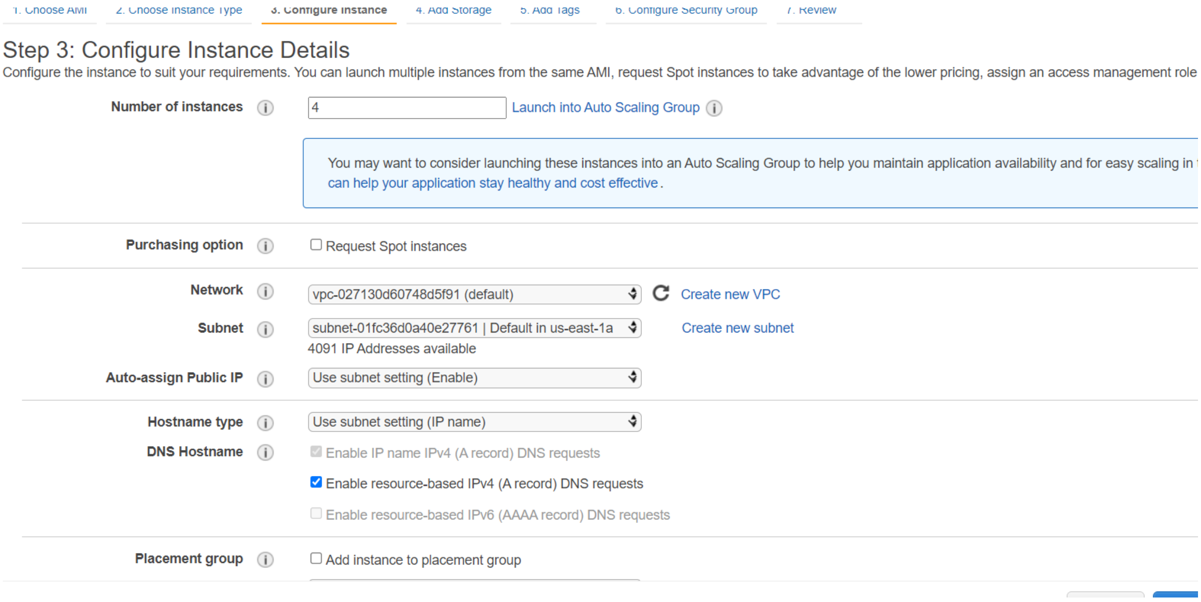
Assign a security group: ● Create a **new** security group
○ Select an **existing** security group

Security group name: project security group

Description: new security group

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop
HTTP	TCP	80	Anywhere 0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop

The security group is created according to our desired inbound rules.



Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role

Number of instances: 4 — Launch into Auto Scaling Group

You may want to consider launching these instances into an Auto Scaling Group to help you maintain application availability and for easy scaling in can help your application stay healthy and cost effective.

Purchasing option: □ Request Spot instances

Network: vpc-027130d60748d5f91 (default) — Create new VPC

Subnet: subnet-01fc36d0a40e27761 | Default in us-east-1a — Create new subnet
4091 IP Addresses available

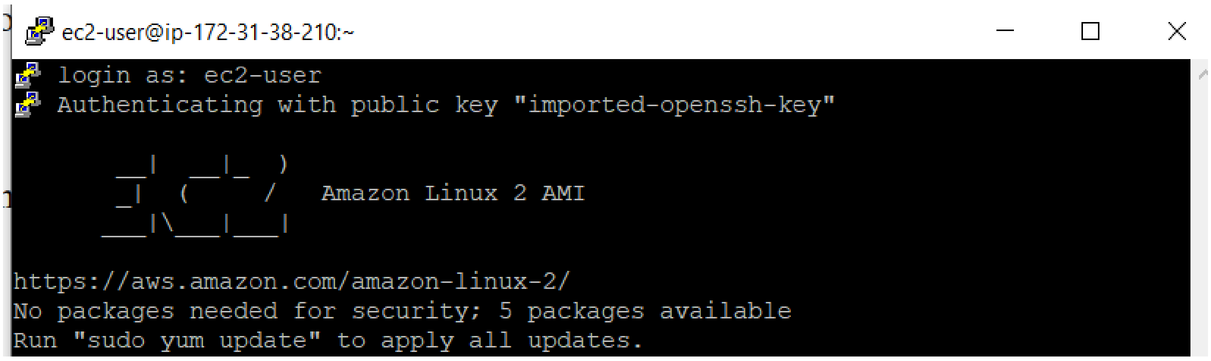
Auto-assign Public IP: Use subnet setting (Enable)

Hostname type: Use subnet setting (IP name)

DNS Hostname:
☑ Enable IP name IPv4 (A record) DNS requests
☑ Enable resource-based IPv4 (A record) DNS requests
□ Enable resource-based IPv6 (AAAA record) DNS requests

Placement group: □ Add instance to placement group

To enable the SSH connection, we have created the ssh key pair and it is added to the ~/.ssh folder and the permissions are set using chmod 600. The SSH connection is made to the linux.



ec2-user@ip-172-31-38-210:~

```
login as: ec2-user
Authenticating with public key "imported-openssh-key"

      __|  __|_  )
      _|  (     /   Amazon Linux 2 AMI
     ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
No packages needed for security; 5 packages available
Run "sudo yum update" to apply all updates.
```

The next step is to set up the environment inside the instance to deploy the Django application. For that, we need to install some packages, the step-to-step procedure is given below. The name of the application that we set up is simple-django-app.

The

- Installing the required python and django packages in the ec2 instance.

The package for python is installed using the sudo apt update and the necessary environment is also set up. The commands that we used are

Sudo apt-get update && sudo apt-get upgrade

The Django is installed using pip3 install Django. The github application is cloned using git clone command. The nginx and gunicorn is installed using the commands

‘sudo apt-get install -y nginx’ && ‘pip3 install gunicorn’

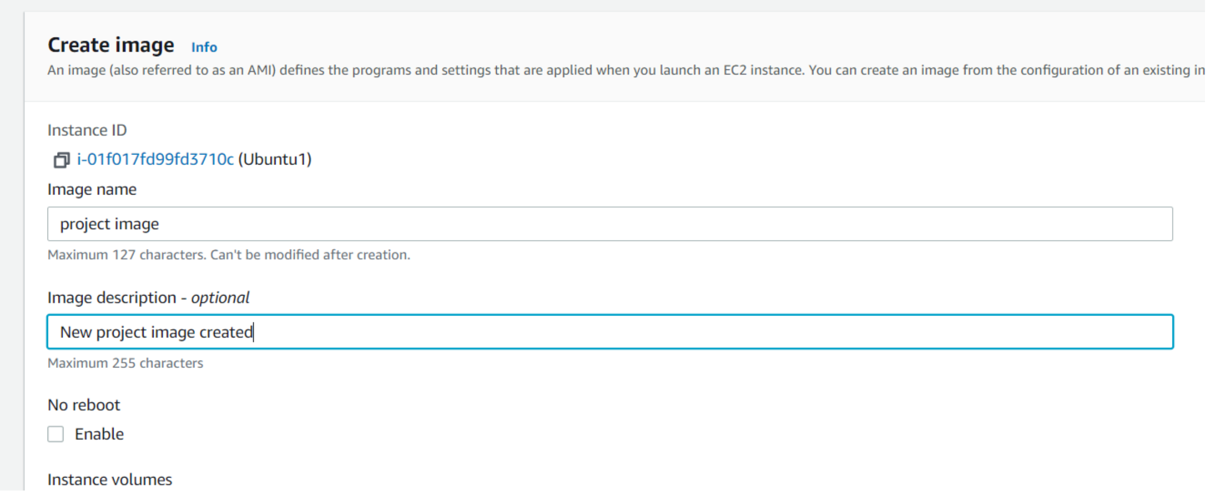
The supervisor is installed. The supervisor allows the users to monitor and control a number of processes on UNIX-like operating systems.

‘sudo apt-get install supervisor’

Under the /etc/supervisor/conf.d. We need to create a file gunicorn.conf. The necessary lines has to be added to have a successful deployment. Later, test the configuration and allow nginx to read and restart the service nginx. The website is deployed and access the website using the public IP address of the EC2 instance.

Set up the autoscaling group to test the scalability feature:

Before setting up the auto scaling group, the image for an instance needs to be created.



Create image Info

An image (also referred to as an AMI) defines the programs and settings that are applied when you launch an EC2 instance. You can create an image from the configuration of an existing in

Instance ID

i-01f017fd99fd3710c (Ubuntu1)

Image name

project image

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

Image description - *optional*

New project image created

Maximum 255 characters

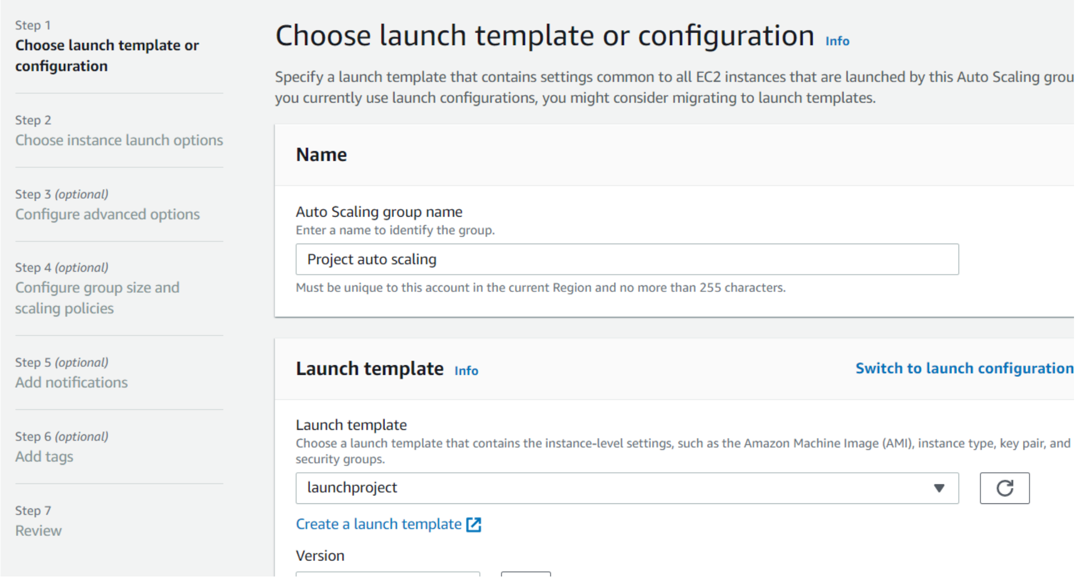
No reboot

□ Enable

Instance volumes

The load balancer and the target group is selected and the type of instance type and the desired number, the minimum and maximum capacity of the instances is selected. The keypair and the configuration is selected, and the process is shown below in the pictures.

- We have created a launch template named as launchproject and once creating is done and this template will be used in configuring the template which is used to creating auto scaling group.



Step 1
Choose launch template or configuration
Step 2
Choose instance launch options
Step 3 (optional)
Configure advanced options
Step 4 (optional)
Configure group size and scaling policies
Step 5 (optional)
Add notifications
Step 6 (optional)
Add tags
Step 7
Review

Choose launch template or configuration Info

Specify a launch template that contains settings common to all EC2 instances that are launched by this Auto Scaling grou you currently use launch configurations, you might consider migrating to launch templates.

Name

Auto Scaling group name

Enter a name to identify the group.

Project auto scaling

Must be unique to this account in the current Region and no more than 255 characters.

Launch template Info

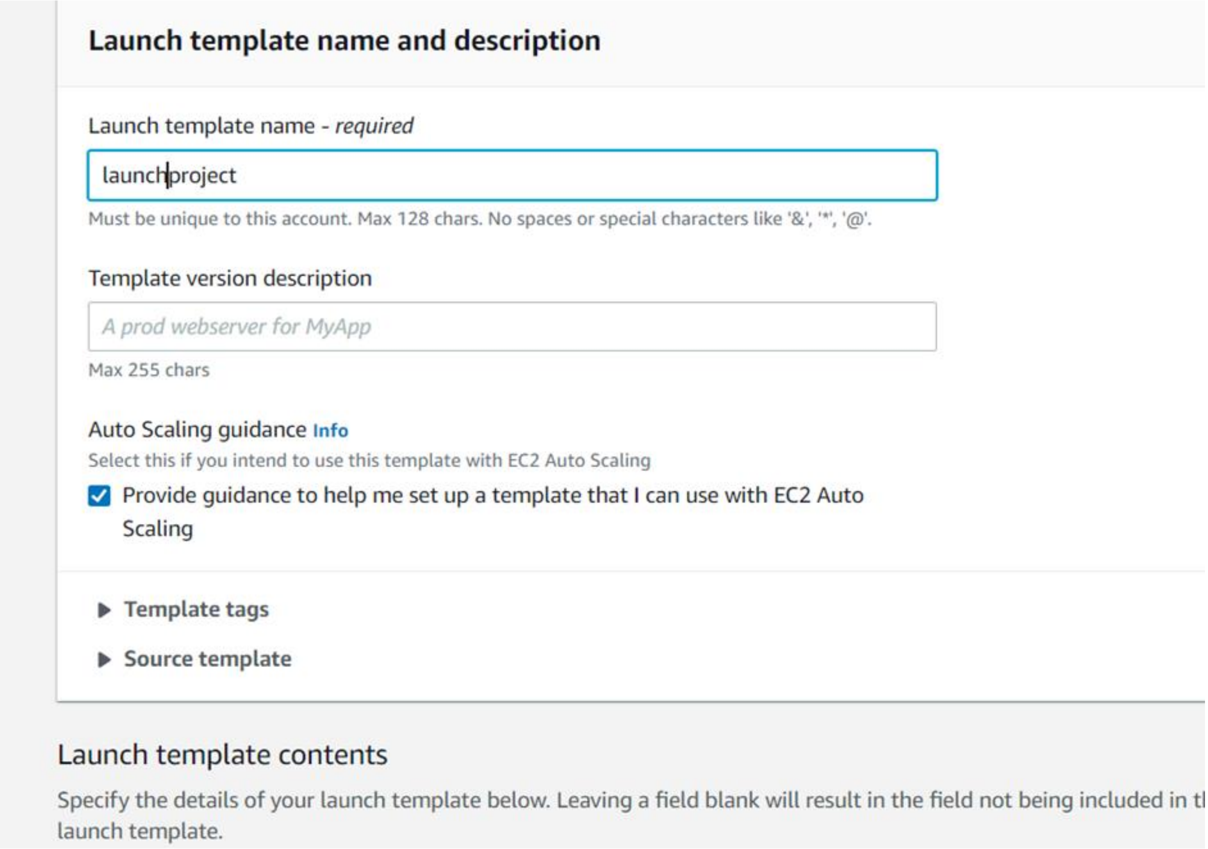
Launch template

Choose a launch template that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups.

launchproject ▼

Version

- In the figure below we can see that already created template is used while creating of auto scaling group process.



Launch template name and description

Launch template name - *required*

launchproject

Must be unique to this account. Max 128 chars. No spaces or special characters like '&', '*', '@'.

Template version description

A prod webserver for MyApp

Max 255 chars

Auto Scaling guidance Info

Select this if you intend to use this template with EC2 Auto Scaling

☑ Provide guidance to help me set up a template that I can use with EC2 Auto Scaling

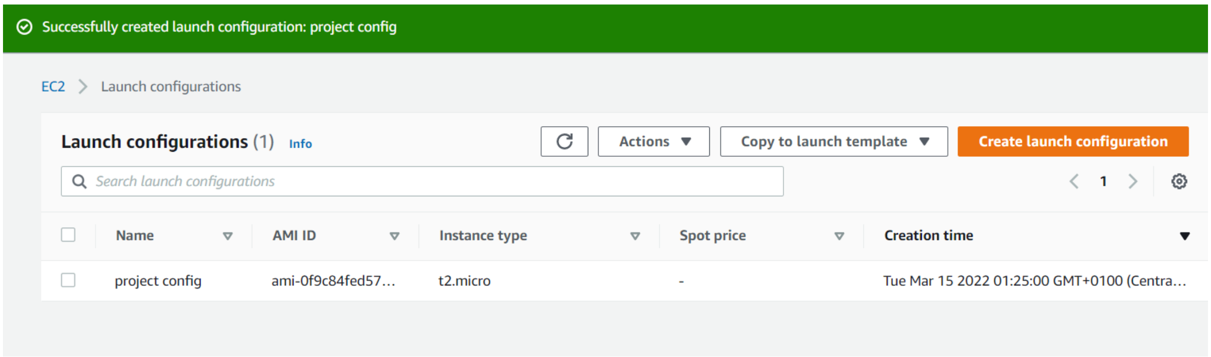
▶ Template tags

▶ Source template

Launch template contents

Specify the details of your launch template below. Leaving a field blank will result in the field not being included in tl launch template.

- In the launching process we have chosen instance type of t2.micro because it is free tier.



Successfully created launch configuration: project config

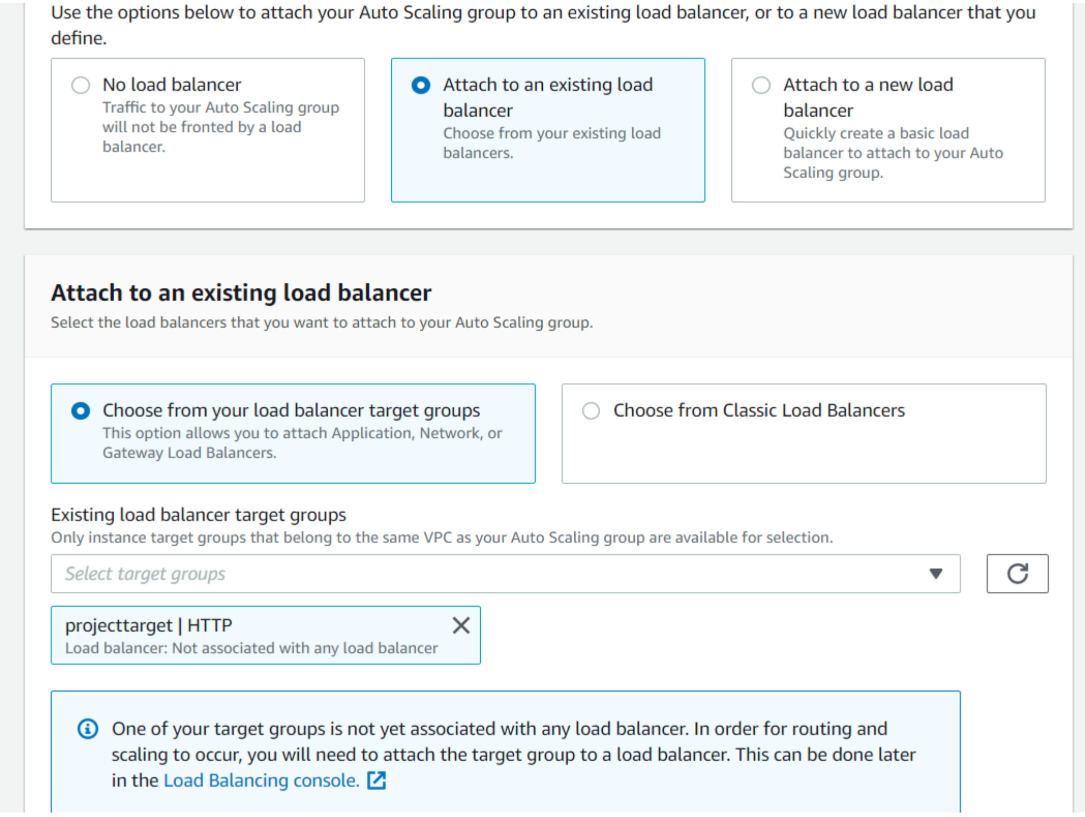
EC2 > Launch configurations

Launch configurations (1) Info

Actions ▼
Copy to launch template ▼
Create launch configuration
Search launch configurations
1

□	Name	AMI ID	Instance type	Spot price	Creation time
□	project config	ami-0f9c84fed57…	t2.micro	-	Tue Mar 15 2022 01:25:00 GMT+0100 (Centra…

- In the next step we have chosen option existing load balancer which we already created target group which is named as projecttarget in the figure below.



Use the options below to attach your Auto Scaling group to an existing load balancer, or to a new load balancer that you define.

○ No load balancer
Traffic to your Auto Scaling group will not be fronted by a load balancer.

● Attach to an existing load balancer
Choose from your existing load balancers.

○ Attach to a new load balancer
Quickly create a basic load balancer to attach to your Auto Scaling group.

Attach to an existing load balancer

Select the load balancers that you want to attach to your Auto Scaling group.

● Choose from your load balancer target groups
This option allows you to attach Application, Network, or Gateway Load Balancers.

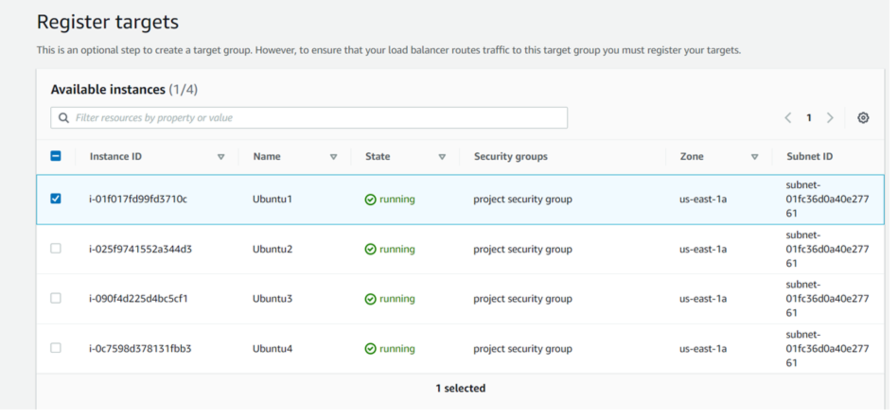
○ Choose from Classic Load Balancers

Existing load balancer target groups
Only instance target groups that belong to the same VPC as your Auto Scaling group are available for selection.

projecttarget | HTTP
Load balancer: Not associated with any load balancer

One of your target groups is not yet associated with any load balancer. In order for routing and scaling to occur, you will need to attach the target group to a load balancer. This can be done later in the Load Balancing console.

- After that we have chosen the instance which we want to register the target from four available instances.



Register targets

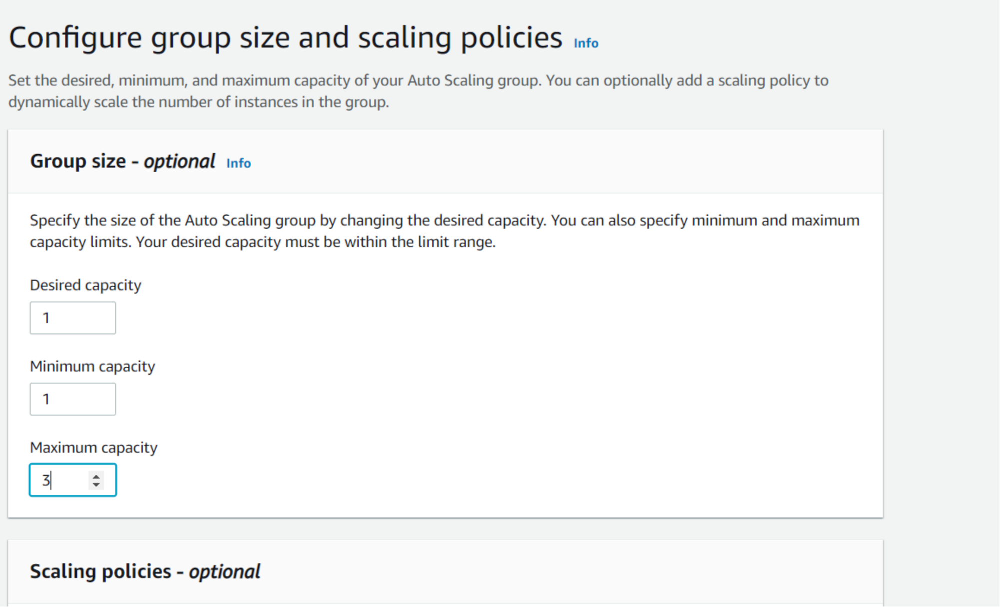
This is an optional step to create a target group. However, to ensure that your load balancer routes traffic to this target group you must register your targets.

Available instances (1/4)

	Instance ID	Name	State	Security groups	Zone	Subnet ID
☑	i-01f017fd99fd3710c	Ubuntu1	running	project security group	us-east-1a	subnet-01fc36d0a40e27761
□	i-025f9741552a344d3	Ubuntu2	running	project security group	us-east-1a	subnet-01fc36d0a40e27761
□	i-090f4d225d4bc5cf1	Ubuntu3	running	project security group	us-east-1a	subnet-01fc36d0a40e27761
□	i-0c7598d378131fbb3	Ubuntu4	running	project security group	us-east-1a	subnet-01fc36d0a40e27761

1 selected

- In the last step we need capacities of instances for the application. In this project we have one instance for desired and minimum capacity and given three for maximum capacity as shown in figure below.



Configure group size and scaling policies Info

Set the desired, minimum, and maximum capacity of your Auto Scaling group. You can optionally add a scaling policy to dynamically scale the number of instances in the group.

Group size - *optional* Info

Specify the size of the Auto Scaling group by changing the desired capacity. You can also specify minimum and maximum capacity limits. Your desired capacity must be within the limit range.

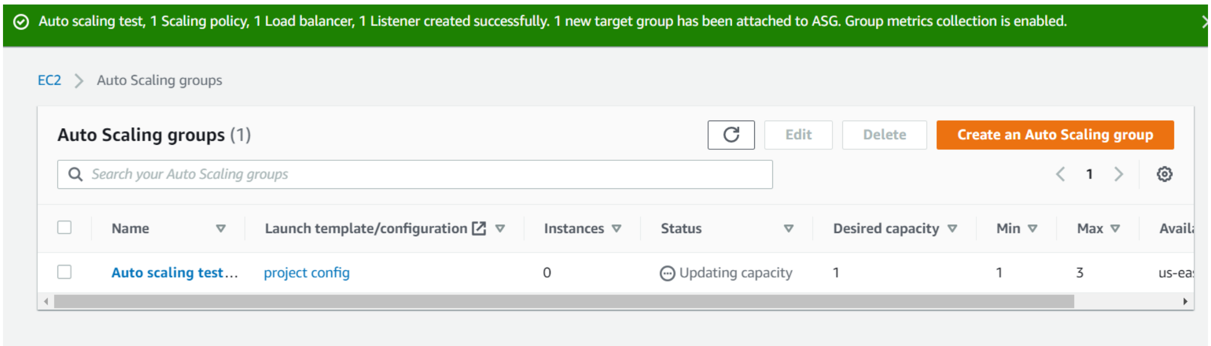
Desired capacity: 1

Minimum capacity: 1

Maximum capacity: 3

Scaling policies - *optional*

- The last step is to create the scaling the group. The following figure shows we have successfully created auto scaling group. In the scaling policy we have given CPU utilization upto 60%.



Auto scaling test, 1 Scaling policy, 1 Load balancer, 1 Listener created successfully. 1 new target group has been attached to ASG. Group metrics collection is enabled.

EC2 > Auto Scaling groups

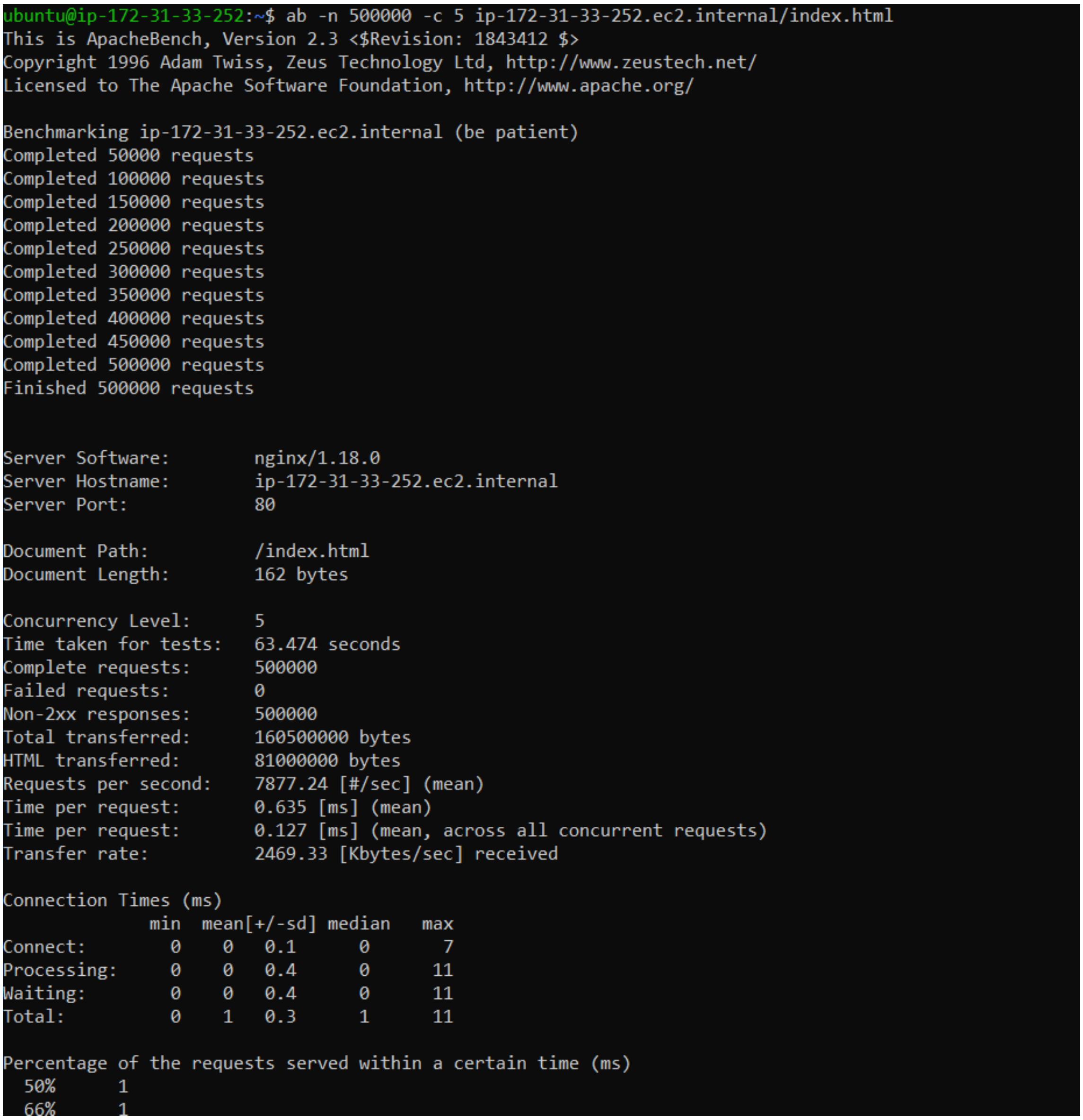
Auto Scaling groups (1)

□	Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Avail
□	Auto scaling test…	project config	0	Updating capacity	1	1	3	us-ea

Testing through performing stress test:

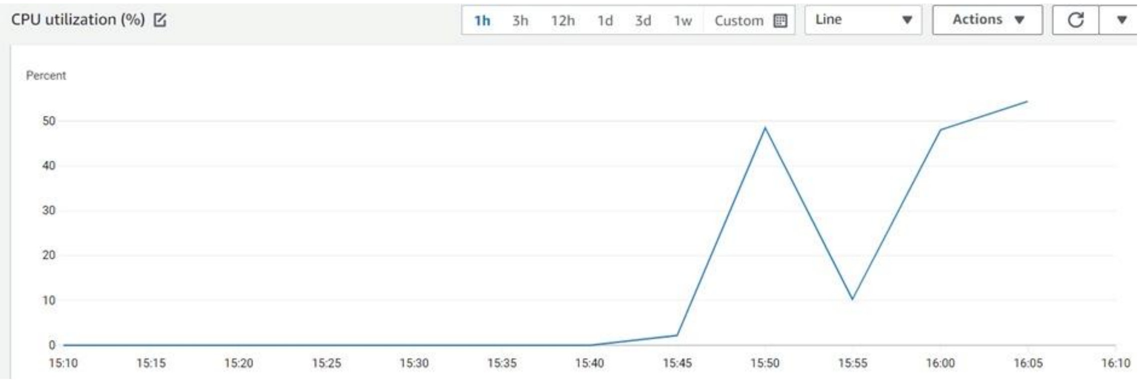
After successfully, setting up the auto scaling group, stress test is performed to check if the auto scaling group is working or not. We have same type of test which we have used in lab2. To perform the stress test, the command that is given in the ubuntu is

ab -n 50000 -c ip-172-31-33-252.ec2.internal/index.html



```
ubuntu@ip-172-31-33-252:~$ ab -n 500000 -c 5 ip-172-31-33-252.ec2.internal/index.html
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking ip-172-31-33-252.ec2.internal (be patient)
Completed 50000 requests
Completed 100000 requests
Completed 150000 requests
Completed 200000 requests
Completed 250000 requests
Completed 300000 requests
Completed 350000 requests
Completed 400000 requests
Completed 450000 requests
Completed 500000 requests
Finished 500000 requests


Server Software:        nginx/1.18.0
Server Hostname:        ip-172-31-33-252.ec2.internal
Server Port:            80

Document Path:          /index.html
Document Length:        162 bytes

Concurrency Level:      5
Time taken for tests:   63.474 seconds
Complete requests:      500000
Failed requests:        0
Non-2xx responses:      500000
Total transferred:      160500000 bytes
HTML transferred:       81000000 bytes
Requests per second:    7877.24 [#/sec] (mean)
Time per request:       0.635 [ms] (mean)
Time per request:       0.127 [ms] (mean, across all concurrent requests)
Transfer rate:          2469.33 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       7
Processing:     0    0   0.4      0      11
Waiting:        0    0   0.4      0      11
Total:          0    1   0.3      1      11

Percentage of the requests served within a certain time (ms)
  50%      1
  66%      1
```

After, the performance of the stress test, the CPU utilization has been raised a lot. It is shown in the picture given below. Increase in stress test resulted in increase in CPU utilization. Due to the auto scaling group, the instance has been created according to the autoscaling settings.



Conclusion:

The main aim of the problem description to test the scalability feature is verified. To perform this test, we have chosen a simple Django application. The environment is set up in the AWS cloud to deploy the application and to test the feature. We have incorporated the load balancer and the auto scaling group. After setting up the environment, we have performed stress test to check the CPU utilization, and after increase in the CPU utilization, the instance has been created. We have concluded that it is very feasible to set up the scalability of the instances through the environment provided by the AWS cloud provider.