

# Project Report

by Group –18

Ananya Reddivari.

Harika Kosuru.

Monica Gattupalli.

Sri Sai Ganesh Satyadeva Naidu Totakura.

Syam Kumar Vemana.

## Problem statement and Problem Description

This project is to deploy and implement a scalable Django application (one-tier architecture), deploying it in Amazon Web Services (AWS) in Ubuntu free tier version, also implemented auto scaling for the application in case of increased load. To measure the load, average CPU utilization is used. This auto scaled instances satisfies the requirements like scalability with respect to computation and high availability of the computation.

## Design Description

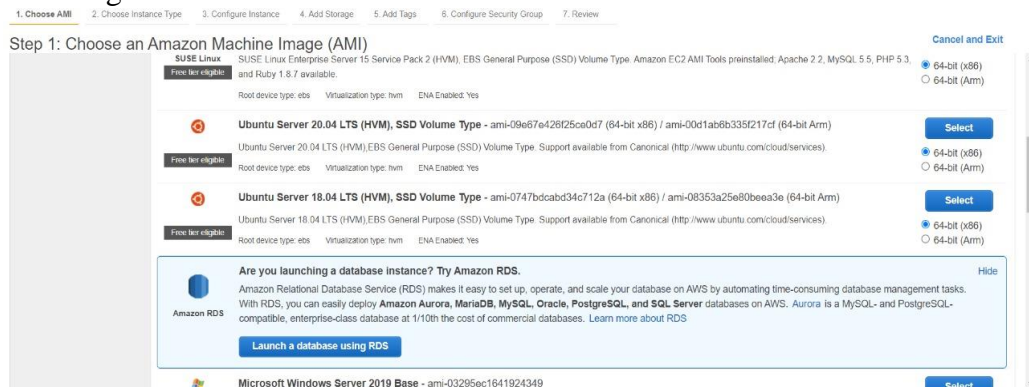
We have achieved the problem statement in 6 steps: -

1. Creation of **EC2** instance in AWS.
2. Deploying the scalable Django application into the created instance.
3. Creation of Amazon Machine Image (AMI) for the instance created in step 1.
4. Creation of launch template for the instance created in step 1.
5. Creation of target group for the instance created in step 1.
6. Configuring the auto scaling group for the instance created in step 1.

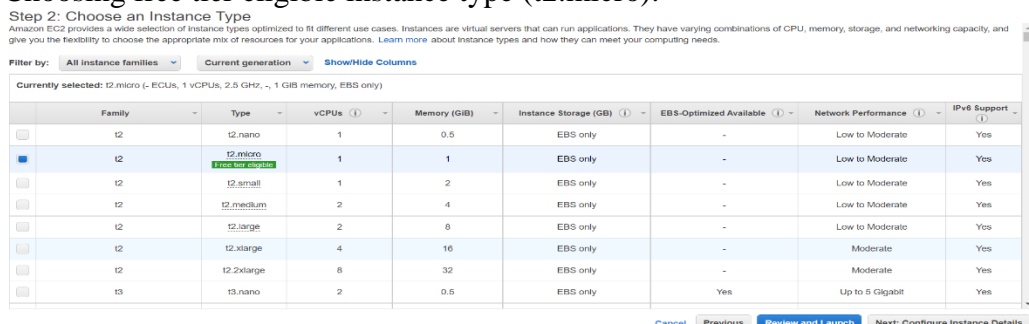
## Implementation

### 1. Creation of EC2 instance in AWS.

- Selecting the free tier version of Ubuntu server.



- Choosing free tier eligible instance type (t2.micro).



- Configuring the instance details

### 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 to the instance, and more.

Number of instances ① 1 Launch into Auto Scaling Group ①

Purchasing option ① ☐ Request Spot instances

Network ① vpc-49ea5e34 (default) Create new VPC

Subnet ① No preference (default subnet in any Availability Zone) Create new subnet

Auto-assign Public IP ① Enable

Placement group ① ☐ Add instance to placement group

Capacity Reservation ① Open

Domain join directory ① No directory Create new directory

IAM role ① None Create new IAM role

Shutdown behavior ① Stop

Stop - Hibernate behavior ① ☐ Enable hibernation as an additional stop behavior

Enable termination protection ① ☐ Protect against accidental termination

Monitoring ① ☐ Enable CloudWatch detailed monitoring

Cancel Previous Review and Launch Next: Add Storage

## ➤ Adding storage to instance

### Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Volume Type ①	Device ①	Snapshot ①	Size (GiB) ①	Volume Type ①	IOPS ①	Throughput (MB/s) ①	Delete on Termination ①	Encryption ①
Root	/dev/sda1	snap-0a52a8f51496c3782	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted

Add New Volume

## ➤ Adding tags to instance

### Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. A copy of a tag can be applied to volumes, instances or both. Tags will be applied to all instances and volumes. [Learn more](#) about tagging your Amazon EC2 resources.

Key (128 characters maximum)	Value (256 characters maximum)	Instances ①	Volumes ①	Network Interfaces ①
Name	project	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Add another tag (Up to 50 tags maximum)

## ➤ Successful creation of Ubuntu instance

New EC2 Experience Learn more

EC2 Dashboard

- Events
- Tags
- Limits
- Instances
  - Instances New
  - Instance Types
  - Launch Templates
  - Spot Requests
  - Savings Plans
  - Reserved Instances New
  - Dedicated Hosts
  - Scheduled Instances
  - Capacity Reservations
- Images
  - AMIs

Instances (1) Info

Filter Instances

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Pub
project	i-0a01ff425d8242d9b	Running	t2.micro	2/2 checks passed	No alarms	us-east-1c	ec2

Launch instance

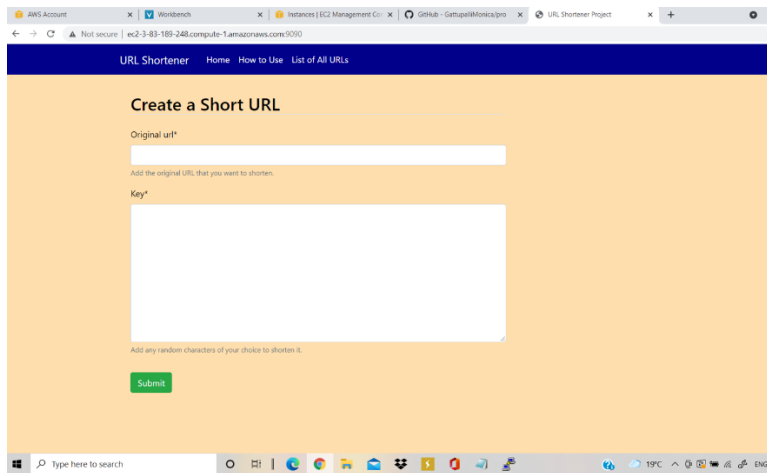
## 2. Deploying the scalable Django application into the created instance.

URL shortener using Django is selected as the scalable Django application for this project. The chosen application is used for customizing the URL's given by the user, and it simply replaces the URL with the key provided by the users. The List of all URLs which are customized is stored in the application. This URL shortener is cloned from our GitHub repository – <https://github.com/GattupalliMonica/pro>

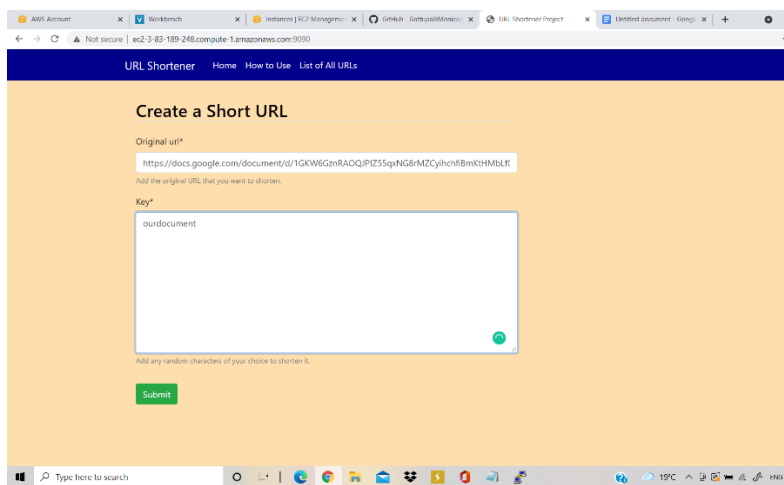
Commands for deploying the Django application into EC2 instance are as follows:

- **sudo apt update** – Updating the Ubuntu server to restore the new packages.
- **sudo apt install nginx** – To install nginx web server for faster performance.
- **cd /etc/nginx** – Changing the directory to nginx.
- **sudo systemctl start nginx** – Used to start the nginx web server.
- **sudo apt install python3-pip** – Used to install the python3 in the instance.
- **cd** – Changing the directory from nginx.
- **sudo apt install python3-virtualenv** – Installing a virtual environment with corresponding to python3.
- **ls** – Displays list of files in virtual environments.
- **mkdir project** – Creating a new directory with name “project”.
- **cd project** – Changing the directory to “project” directory.
- **git clone <https://github.com/GattupalliMonica/pro>** - Cloning the git repository to access the application files.
- **virtualenv venv** – Creating a virtual environment with name “venv”.
- **ls** – Displays list of files in virtual environments.
- **source venv/bin/activate** – Activating the virtual environment from the source.
- **pip install -r pro/requirements.txt** – Installing the required modules for the URL shortener application from “requirements.txt” file
- **pip install gunicorn** – Installing the gunicorn which is used to connect the Django application with nginx web server.
- **cd URL\_Shortener\_Project\_using\_Django** – Changing the directory to our main project directory.
- **gunicorn --bind 0.0.0.0:9090 URL\_Shortener\_Project\_using\_Django.wsgi**  
– Accessing the application from 9090 port number through “wsgi” interface.

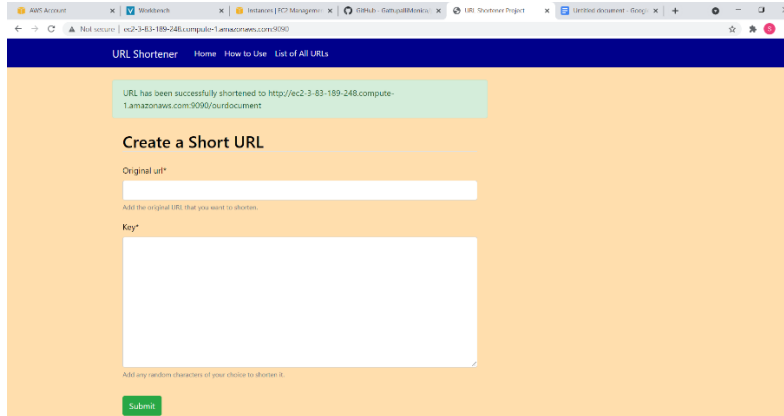
Accessing the URL shortener application using Django from the public DNS of the Ubuntu instance.



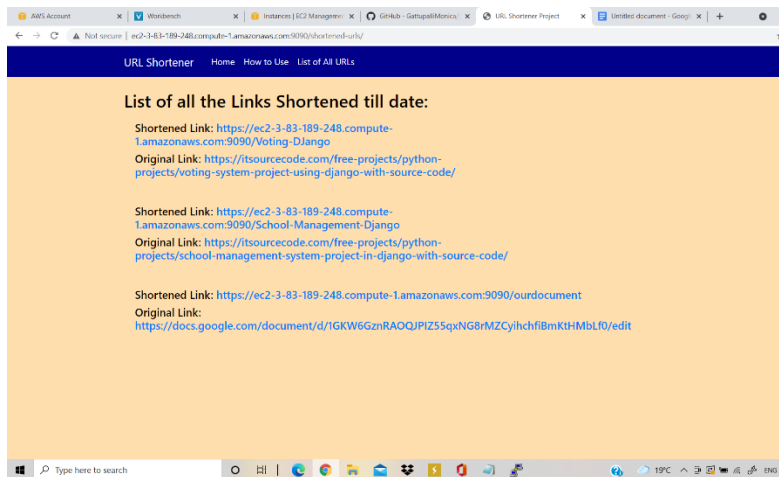
A random URL and key is entered by the user to check the application.



The entered URL is customised by the key after clicking submit button.



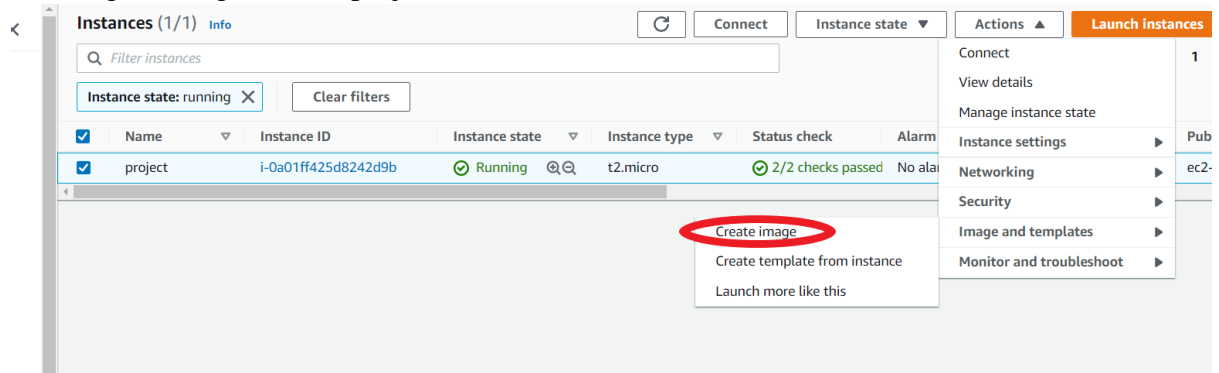
## List of all customised URLs



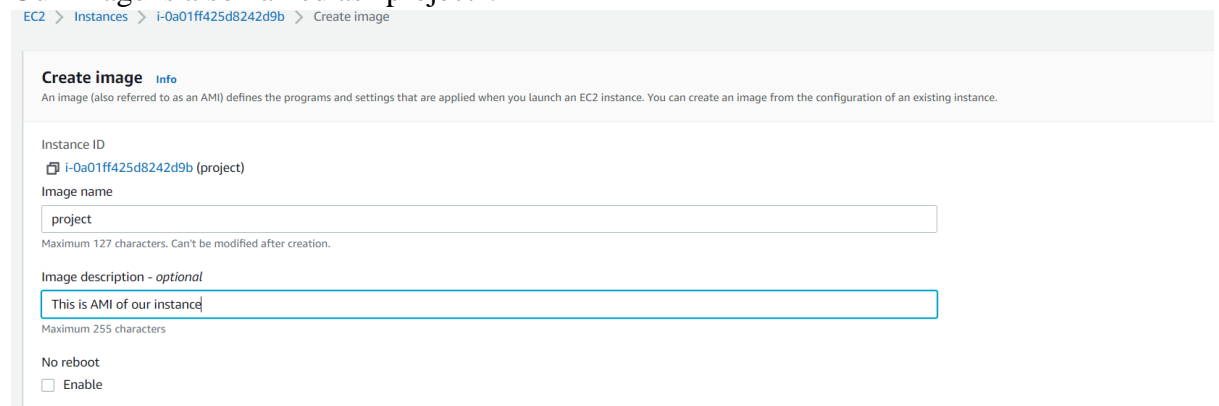
### 3. Creation of Amazon Machine Image (AMI)

AMI provides required information to launch an instance and used to deploy multiple instances with same configuration.

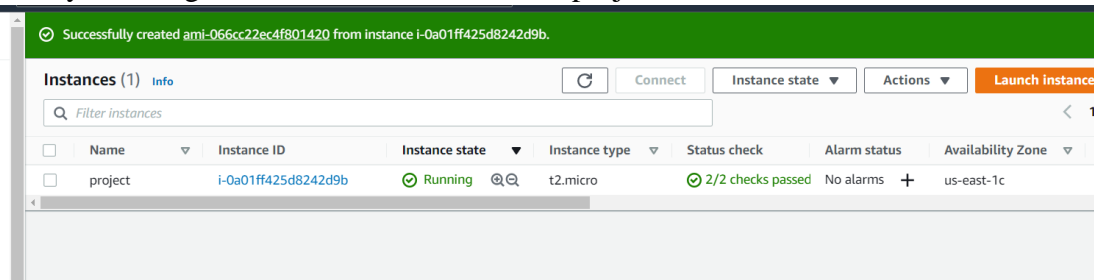
Creating an image to our “project” instance.



Our image is also named as “project”.



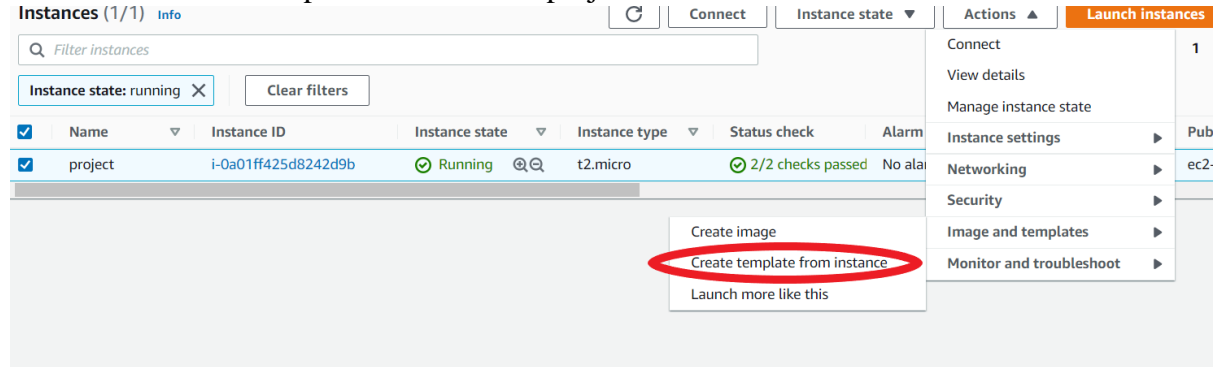
Finally, an image is created for the instance “project”



#### 4. Creation of launch template

Launch template specifies the instance configuration details and enables us to use the auto scaling groups.

Creation of launch template for instance “project”.



Configuring the launch template with template name and the template version description.

EC2 > Launch templates > Create launch template

### Create launch template

Creating a launch template allows you to create a saved instance configuration that can be reused, shared and launched at a later time. Templates can have multiple versions.

#### Launch template name and description

Launch template name - *required*

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

Template version description

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

Successful creation of launch template.

Launch templates (1/3) Info

Filter by tags or properties or search by keyword

Launch template ID	Launch template name	Default version	Latest version
lt-02799681afcb85094	projecttemplate	1	1

projecttemplate (lt-02799681afcb85094)

Launch template details

## 5. Creation of target group.

Target group is used to control the network traffic under the specified protocols and ports to specified instance. This ensures the “high availability of computation” by avoiding the network traffic of the specified instance. This target group is attached to the auto scaling group to manage the increase and decrease of instances due to load variations.

Creation of target group by selecting the instances as the basic configuration and the name of the target group is “projecttargetgroup”.

EC2 > Target groups > Create target group

Step 1  
Specify group details

Step 2  
Register targets

### Specify group details

Your load balancer routes requests to the targets in a target group and performs health checks on the targets.

**Basic configuration**

Choose a target type

- ☒ **Instances**
  - Supports load balancing to instances within a specific VPC.
- ☐ **IP addresses**
  - Supports load balancing to VPC and on-premises resources.
  - Facilitates routing to multiple IP addresses and network interfaces on the same instance.
  - Offers flexibility with microservice based architectures, simplifying inter-application communication.
- ☐ **Lambda function**
  - Facilitates routing to a single Lambda function.
  - Accessible to Application Load Balancers only.

Target group name

projecttargetgroup

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.



## Registering the target group

EC2 > Target groups > Create target group

Step 1  
Specify group details

Step 2  
Register targets

### Register targets

Available instances (1/1)

Filter resources by property or value

<input checked="" type="checkbox"/>	Instance ID	Name	State	Security groups	Zone	Subnet ID
<input checked="" type="checkbox"/>	i-0a01ff425d8242d9b	project	running	launch-wizard-8	us-east-1c	subnet-6fb92109

1 selected

Ports for the selected instances  
Ports for routing traffic to the selected instances (separate multiple ports with commas):

80

Include as pending below

1 selection is now pending below. Include more or register targets when ready.

Include as pending below

1 selection is now pending below. Include more or register targets when ready.

### Targets (1)

Remove all pending

All Filter resources by property or value

Remove	Health status	Instance ID	Name	Port	State	Security groups	Zone	Subnet ID
X	Pending	i-0a01ff425d8242d9b	project	80	running	launch-wizard-8	us-east-1c	subnet-6fb92109

1 pending

Cancel Previous Create target group

## Successful creation of target group under HTTP protocol with port 80.

Successfully created target group: projecttargetgroup

EC2 > Target groups

### Target groups (1) info

Search or filter target groups

	Name	ARN	Port	Protocol	Target type	Load balancer
<input type="checkbox"/>	projecttargetgroup	arn:aws:elasticloadbalancin...	80	HTTP	Instance	-

## 6. Configuring the auto scaling group.

Auto scaling is a technique for dynamically adding or removing the computational resources under the measure of average CPU utilisation or network in-bytes and out-bytes. In this project average CPU utilisation is taken as a scale to measure the load on the instance and the auto scaling is done.

## Choosing the launch template for creating the auto scaling group.

[EC2](#) > [Auto Scaling groups](#) > Create Auto Scaling group

Step 1  
**Choose launch template or configuration**

Step 2  
Configure settings

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 group. If you currently use launch configurations, you might consider migrating to launch templates.

**Name**

**Auto Scaling group name**  
Enter a name to identify the group.  
  
Must be unique to this account in the current Region and no more than 255 characters.

**Launch template** [Info](#) [Switch to launch configuration](#)

**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.

[Create a launch template](#)

**Version**

[Create a launch template version](#)

## Adhere to launch template is used as configuration setting for the auto scaling group.

[EC2](#) > [Auto Scaling groups](#) > Create Auto Scaling group

Step 1  
[Choose launch template or configuration](#)

**Step 2  
Configure settings**

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

### Configure settings [Info](#)

Configure the settings below. Depending on whether you chose a launch template, these settings may include options to help you make optimal use of EC2 resources.

**Instance purchase options** [Info](#)

Use the launch template to create a uniform configuration among all of the instances in the group. Or define options to accommodate a wide variety of requirements, such as launching Spot and On-Demand Instances.

☒ **Adhere to launch template**  
The launch template determines the purchase option (On-Demand or Spot) and instance type.

☐ **Combine purchase options and instance types**  
Specify how much On-Demand and Spot capacity to launch and multiple instance types (optional). This choice is most helpful for optimizing the scale and cost for a fleet of instances.

**Network** [Info](#)

For most applications, you can use multiple Availability Zones and let EC2 Auto Scaling balance your instances across the zones. The default VPC and default subnets are suitable for getting started quickly.

**VPC**  
  
172.31.0.0/16 Default

Attached to an existing load balancer by selecting the target groups of the instance “project”.

Step 2  
Configure settings

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

Load balancing - optional [Info](#)

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.

Select target groups

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

Configuring the group size and scaling policies like desired capacity, maximum capacity, and minimum capacity.

EC2 > Auto Scaling groups > Create Auto Scaling group

Step 1  
Choose launch template or configuration

Step 2  
Configure settings

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

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

2

Scaling policies - optional

Choose whether to use a scaling policy to dynamically resize your Auto Scaling group to meet changes in demand. [Info](#)

Successful auto scaling group with scaling policies.

project auto scaling, 1 Scaling policy created successfully

EC2 > Auto Scaling groups

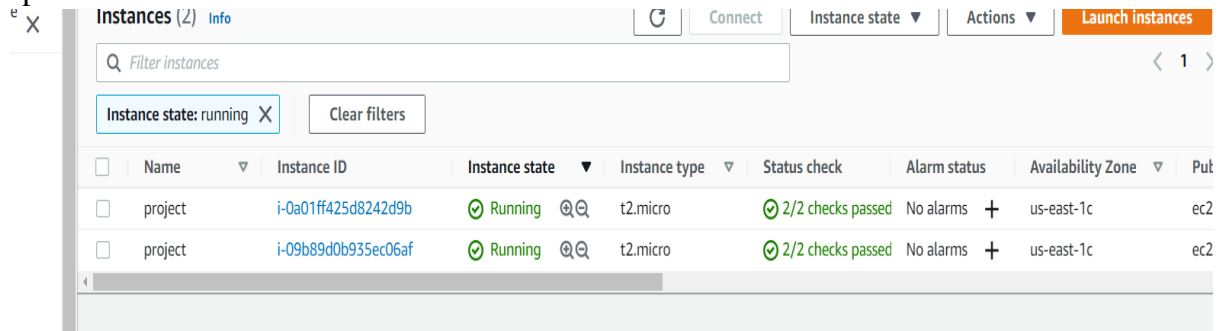
Auto Scaling groups (1)

Search your Auto Scaling groups

< 1 >

<input type="checkbox"/>	Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Availability zones
<input type="checkbox"/>	project auto scaling	projecttemplate   Version Default	0	Updating capacity	1	1	2	us-east-1a

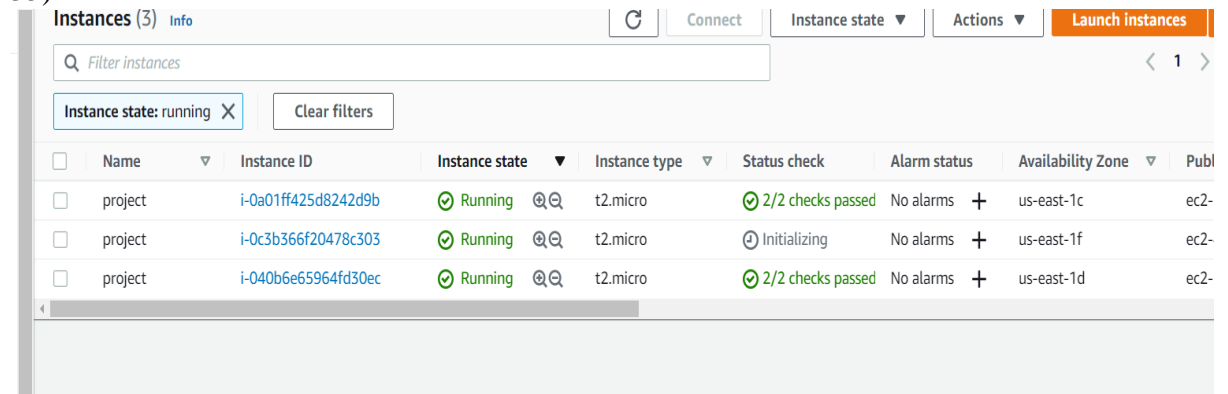
Now auto scaling group is attached to the instance “project”, as we have given desired capacity as 1 after attaching scaling group an instance with similar configuration is spawn.



	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Publ
<input type="checkbox"/>	project	i-0a01ff425d8242d9b	Running	t2.micro	2/2 checks passed	No alarms	us-east-1c	ec2
<input type="checkbox"/>	project	i-09b89d0b935ec06af	Running	t2.micro	2/2 checks passed	No alarms	us-east-1c	ec2

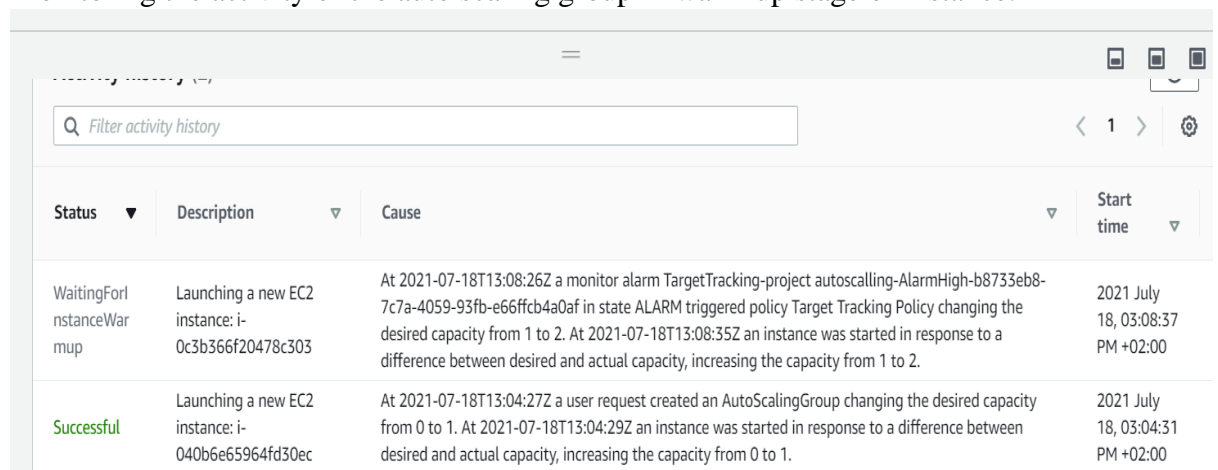
For our Django application multiple requests are taken from n number of users, this increases load on CPU. Another instance is created when the load threshold value is greater than 35.

As the maximum capacity of the auto scaling group is 2, another resource is added to the instance project by satisfying the auto scaling policies (average CPU utilization > 35)



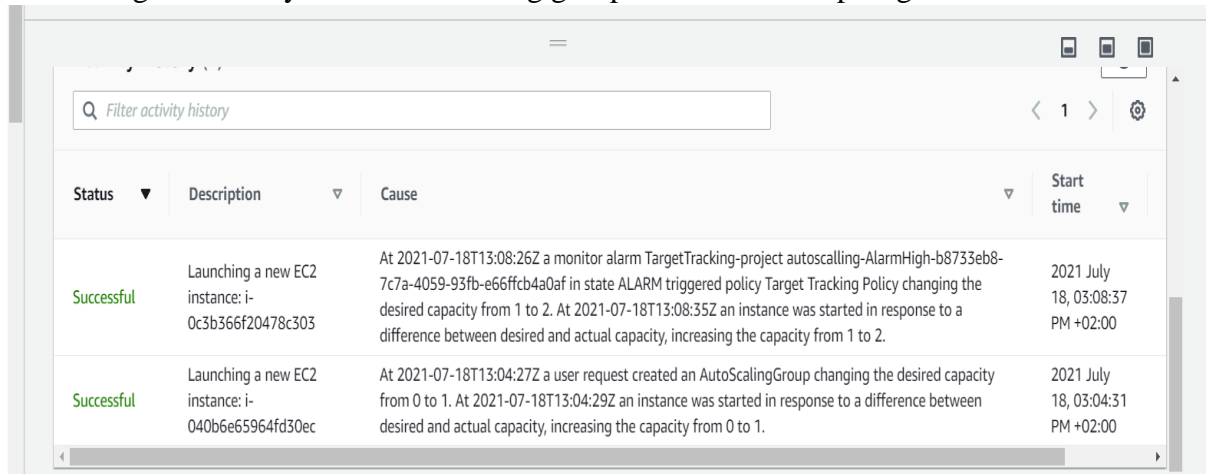
	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Publ
<input type="checkbox"/>	project	i-0a01ff425d8242d9b	Running	t2.micro	2/2 checks passed	No alarms	us-east-1c	ec2-
<input type="checkbox"/>	project	i-0c3b366f20478c303	Running	t2.micro	Initializing	No alarms	us-east-1f	ec2-
<input type="checkbox"/>	project	i-040b6e65964fd30ec	Running	t2.micro	2/2 checks passed	No alarms	us-east-1d	ec2-

Monitoring the activity of the auto scaling group in warm up stage of instance.



Status	Description	Cause	Start time
WaitingForInstanceWarmup	Launching a new EC2 instance: i-0c3b366f20478c303	At 2021-07-18T13:08:26Z a monitor alarm TargetTracking-project autoscaling-AlarmHigh-b8733eb8-7c7a-4059-93fb-e66ffcb4a0af in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 1 to 2. At 2021-07-18T13:08:35Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 2.	2021 July 18, 03:08:37 PM +02:00
Successful	Launching a new EC2 instance: i-040b6e65964fd30ec	At 2021-07-18T13:04:27Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 1. At 2021-07-18T13:04:29Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 1.	2021 July 18, 03:04:31 PM +02:00

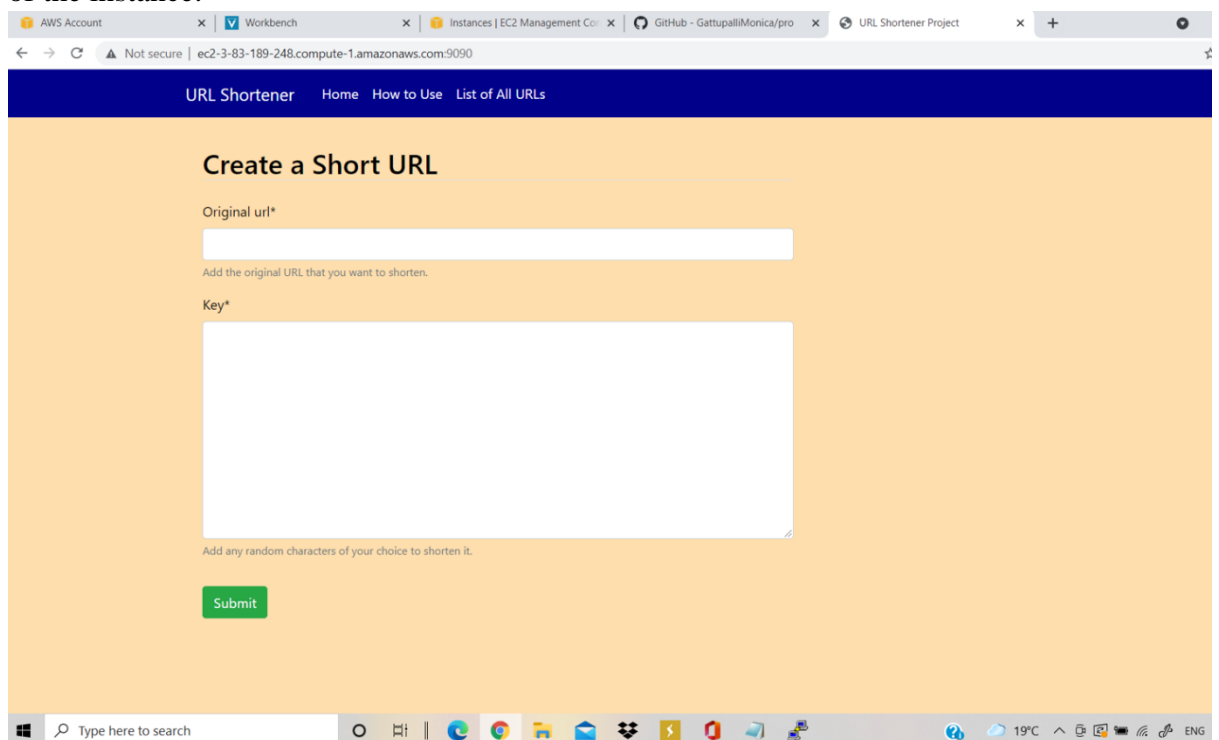
Monitoring the activity of the auto scaling group after the warmup stage of instance.



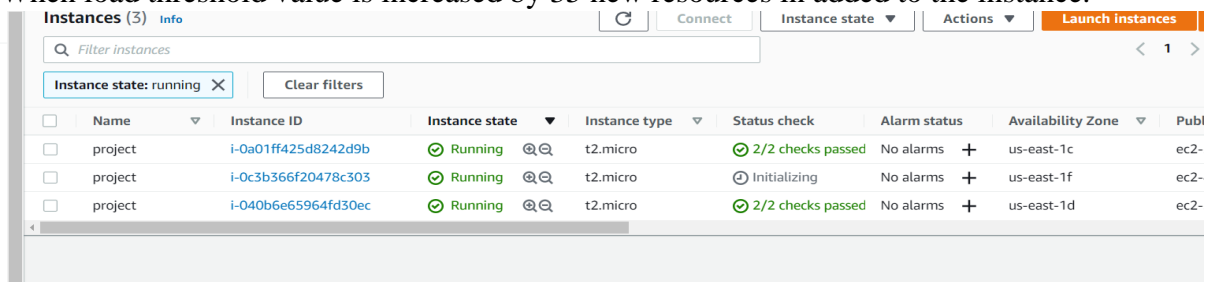
Status	Description	Cause	Start time
Successful	Launching a new EC2 instance: i-0c3b366f20478c303	At 2021-07-18T13:08:26Z a monitor alarm TargetTracking-project autoscalling-AlarmHigh-b8733eb8-7c7a-4059-93fb-e66ffcb4a0af in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 1 to 2. At 2021-07-18T13:08:35Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 2.	2021 July 18, 03:08:37 PM +02:00
Successful	Launching a new EC2 instance: i-040b6e65964fd30ec	At 2021-07-18T13:04:27Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 1. At 2021-07-18T13:04:29Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 1.	2021 July 18, 03:04:31 PM +02:00

## Validation

- ❖ Deploying the URL shortener project into the instance, executed it using public DNS of the instance.



- ❖ When load threshold value is increased by 35 new resources in added to the instance.



	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Publ
<input type="checkbox"/>	project	i-0a01ff425d8242d9b	Running	t2.micro	2/2 checks passed	No alarms	us-east-1c	ec2-
<input type="checkbox"/>	project	i-0c3b366f20478c303	Running	t2.micro	Initializing	No alarms	us-east-1f	ec2-
<input type="checkbox"/>	project	i-040b6e65964fd30ec	Running	t2.micro	2/2 checks passed	No alarms	us-east-1d	ec2-

- ❖ Monitoring the CPU utilisation through the cloud watch within the warmup period of 5mins.



## Results

### ➤ Scalability with respect to computation

It is a property of a system to handle the amount of work by adding or removing resource to the system.

In this project scalability with respect to computation is achieved through auto scaling where a resource is added to the instance when threshold value is greater than 35.

### ➤ High availability of computation

High availability means access to the data, services, and tools whenever the user requires them.

In this project high availability of computation is achieved with two different features such as EC2 instance and the target group. Where the EC2 instance holds the Django application and make it available to the users. The target group avoids the network traffic, increases the ease of access to application and the users.

### ➤ One tier architecture

All the functions and the elements of the application like storage, database and server should be accessed from a single system.

In this project we achieved one tier architecture by cloning the application from Git repository.

- Thus, Django application selected for this project satisfies scalability with respect to computation, high availability of computation and one tier architecture.