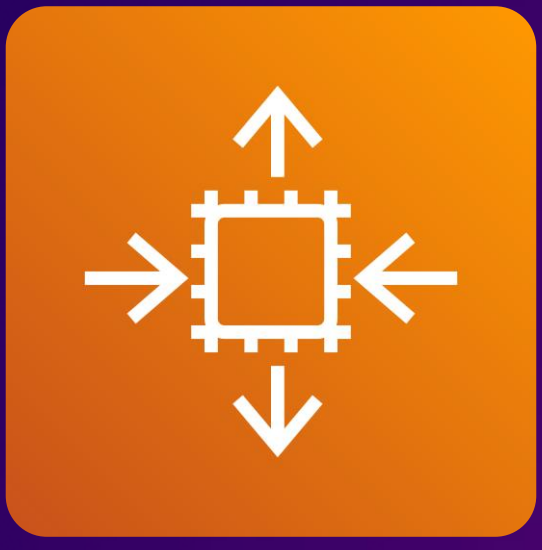
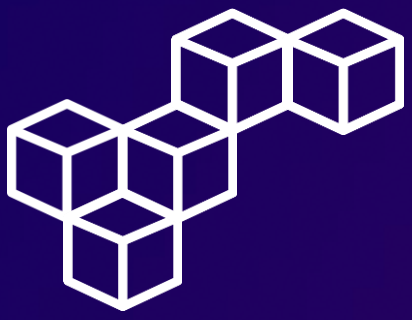
AWS re:Start LAB

Using Auto Scaling in AWS



WEEK 9

aws re/start



Overview

In this lab, you use the AWS Command Line Interface (AWS CLI) to create an Amazon Elastic Compute Cloud (EC2) instance to host a web server and create an Amazon Machine Image (AMI) from that instance. You then use that AMI as the basis for launching a system that scales automatically under a variable load by using Amazon EC2 Auto Scaling. You also create an Elastic Load Balancer to distribute the load across EC2 instances created in multiple Availability Zones by the auto scaling configuration.

Using Auto Scaling in AWS allows you to maintain application availability and optimize resource use by automatically adjusting the number of EC2 instances based on demand. With Auto Scaling, you can set up scaling policies and health checks to ensure that your application remains responsive and cost-effective. By leveraging these features, your system can dynamically respond to varying load conditions, minimizing manual intervention and enhancing overall performance.

Topics covered

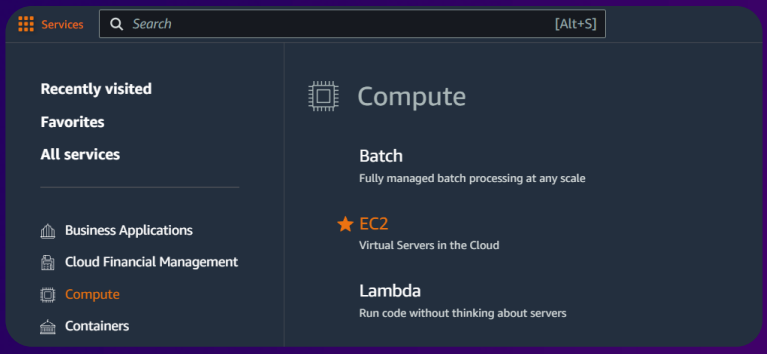
- Create an EC2 instance by using an AWS CLI command.
- Create a new AMI by using the AWS CLI.
- Create an Amazon EC2 launch template.
- Create an Amazon EC2 Auto Scaling launch configuration.
- Configure scaling policies and create an Auto Scaling group to scale in and scale out the number of servers based on a variable load.

Task 1

Creating a new AMI for Amazon EC2 Auto Scaling

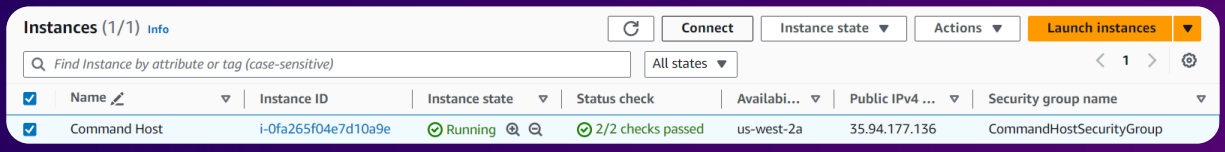
Step 1: Access the EC2 Management Console

Open the AWS Management Console, and select EC2.



Step 2: Connect to the Command Host instance

Navigate to the **Instances** section, select the **Command Host** instance, and connect to the instance using EC2 Instance Connect.



aws re/start

Task 1

Creating a new AMI for Amazon EC2 Auto Scaling

Step 3: Confirm Region

To confirm that the Region in which the **Command Host** instance is running is the same as the lab (the us-west-2 Region), run the following command.

```
[ec2-user@ip-10-0-1-120 ~]$ curl http://169.254.169.254/latest/dynamic/instance-identity/document | grep region
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   475  100   475    0     0   204k      0 --:--:-- --:--:-- --:--:--  231k
  "region" : "us-west-2",
[ec2-user@ip-10-0-1-120 ~]$
```

Step 4: Set AWS CLI credentials

To update the AWS CLI software with the correct credentials, run the aws configure command, and enter the following information.

```
[ec2-user@ip-10-0-1-120 ~]$ aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]: us-west-2
Default output format [None]: json
[ec2-user@ip-10-0-1-120 ~]$
```

aws re/start

Task 1

Creating a new AMI for Amazon EC2 Auto Scaling

Step 5: Review User Data

Review the UserData.txt script. This script performs a number of initialization tasks, including updating all installed software on the box and installing a small PHP web application that you can use to simulate a high CPU load on the instance.

```
[ec2-user@ip-10-0-1-120 ~]$ more UserData.txt
#!/bin/bash
yum update -y --security
amazon-linux-extras install epel -y
yum -y install httpd php stress
systemctl enable httpd.service
systemctl start httpd
cd /var/www/html
wget http://aws-tc-largeobjects.s3.amazonaws.com/CUR-TF-100-TULABS-1/10-lab-autoscaling-linux/s3/ec2-stress.zip
unzip ec2-stress.zip

echo 'UserData has been successfully executed. ' >> /home/ec2-user/result
find -wholename /root/.*history -wholename /home/*/.*history -exec rm -f {} \;
find / -name 'authorized_keys' -exec rm -f {} \;
rm -rf /var/lib/cloud/data/scripts/*
[ec2-user@ip-10-0-1-120 ~]$
```

Step 6: Create a new EC2 instance

Run the following aws ec2 run-instances command to create a new instance that hosts a web server. Make note of the InstanceId for the newly created **Web Server** instance.

```
[ec2-user@ip-10-0-1-120 ~]$ aws ec2 run-instances \
> --key-name vockey \
> --instance-type t3.micro \
> --image-id ami-060aed23281407591 \
> --user-data file:///home/ec2-user/UserData.txt \
> --security-group-ids sg-0cede0e6c71bc8e17 \
> --subnet-id subnet-07ec7c7fc0e48d61e \
> --associate-public-ip-address \
> --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=WebServer}]' \
> --output text \
> --query 'Instances[*].InstanceId'
i-074caa990b3488cc8
[ec2-user@ip-10-0-1-120 ~]$
```

Task 1

Creating a new AMI for Amazon EC2 Auto Scaling

Step 7: Monitor instance status

Use the aws ec2 wait instance-running command to monitor the **Web Server** instance status. This command just waits until the instance is running. Wait for the command to return to a prompt before proceeding.

```
[ec2-user@ip-10-0-1-120 ~]$ aws ec2 wait instance-running --instance-ids i-074caa990b3488cc8
[ec2-user@ip-10-0-1-120 ~]$
```

Step 8: Obtain the public DNS name

To obtain the instance public DNS name, run the following aws ec2 describe-instances command.

```
[ec2-user@ip-10-0-1-120 ~]$ aws ec2 describe-instances --instance-id i-074caa990b3488cc8 \
> --query 'Reservations[0].Instances[0].NetworkInterfaces[0].Association.PublicDnsName'
"ec2-35-91-137-65.us-west-2.compute.amazonaws.com"
[ec2-user@ip-10-0-1-120 ~]$
```

Task 1

Creating a new AMI for Amazon EC2 Auto Scaling

Step 9: Test the Web Server

Review the web server installation using the instance public DNS name. The web server appears to be running.

Step 10: Create a Custom AMI

To create a new AMI based on the **Web Server** instance, run the following aws ec2 create-image command.

```
[ec2-user@ip-10-0-1-120 ~]$ aws ec2 create-image --name WebServerAMI --instance-id i-074caa990b3488cc8
{
    "ImageId": "ami-0d686c1781e0eeceb"
}
[ec2-user@ip-10-0-1-120 ~]$
```

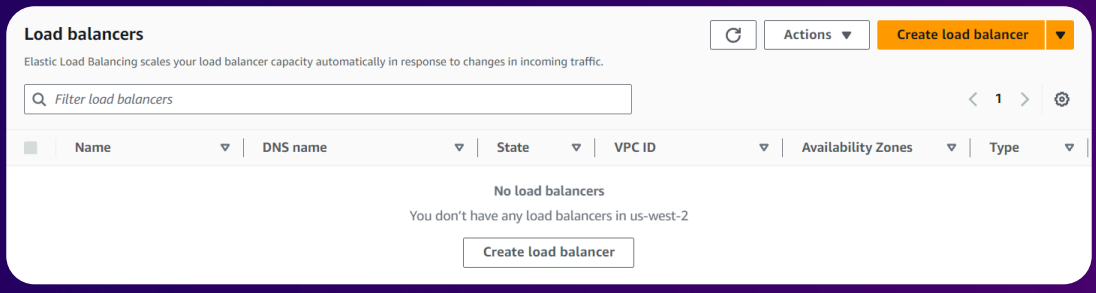
aws re/start

Task 2

Creating an auto scaling environment

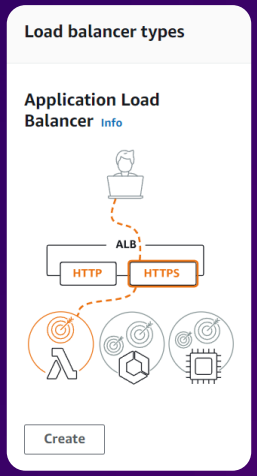
Step 1: Create load balancer

Navigate to the **Load balancers** section, and select Create load balancer.



Step 2: Create Application Load Balancer

In the **Load balancer types** section, select **Application Load Balancer**, and choose Create.



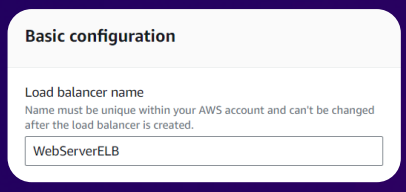
aws re/start

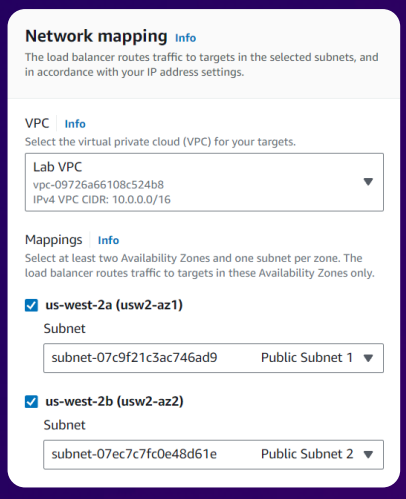
Task 2

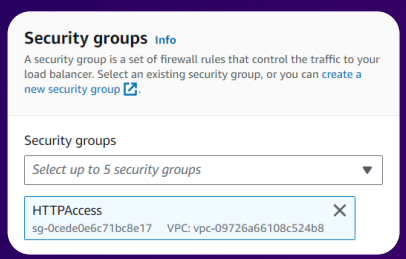
Creating an auto scaling environment

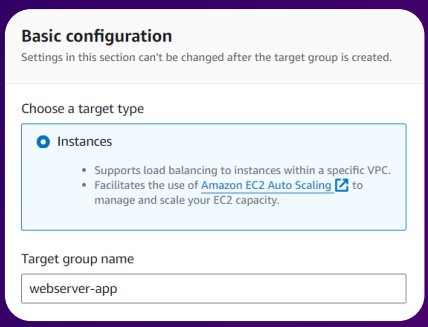
Step 3: Set up the Application Load Balancer

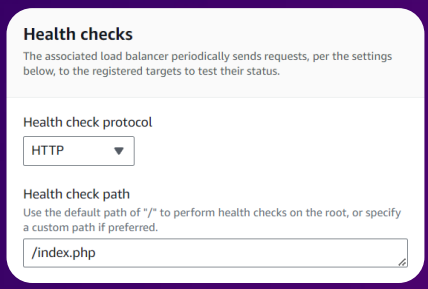
Configure the Load Balancer using the following parameters.

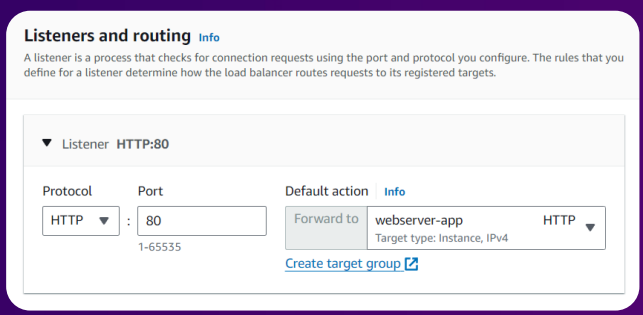












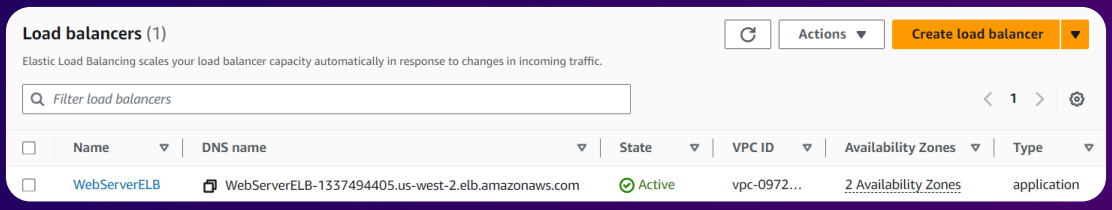
aws re/start

Task 2

Creating an auto scaling environment

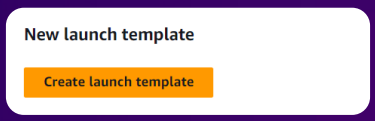
Step 4: Review load balancer creation

Navigate to the **Load balancers** section, and review the newly created load balancer **WebServerELB**, and copy the DNS name.



Step 5: Create launch template

Navigate to the **Launch Templates** section, and select Create launch template.



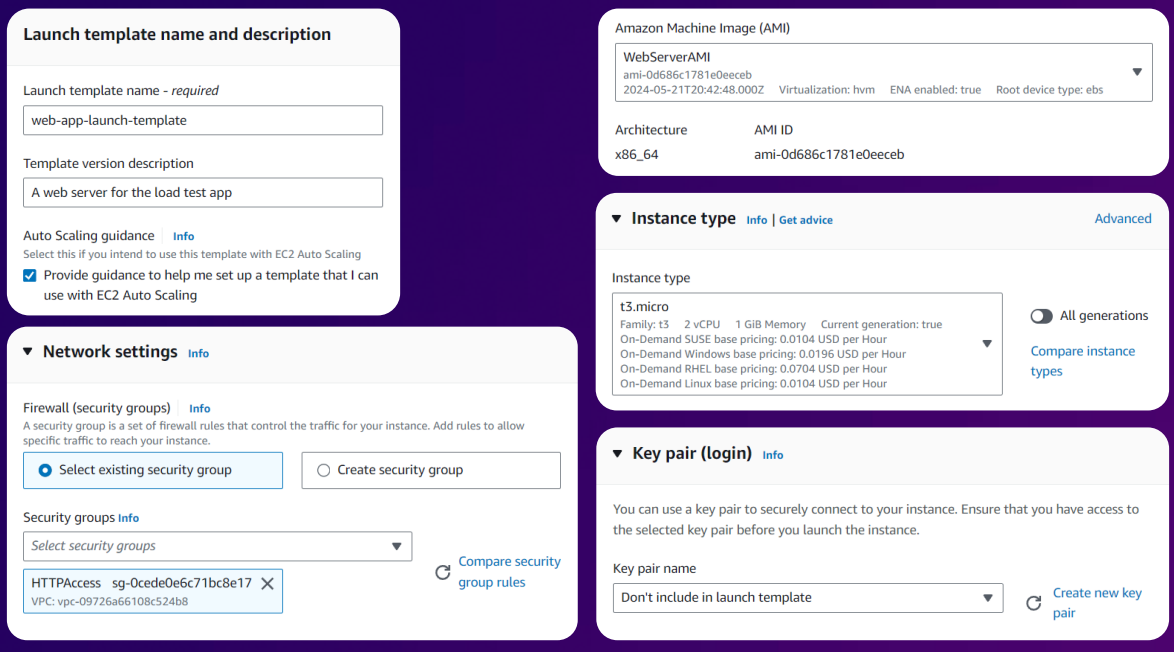
aws re/start

Task 2

Creating an auto scaling environment

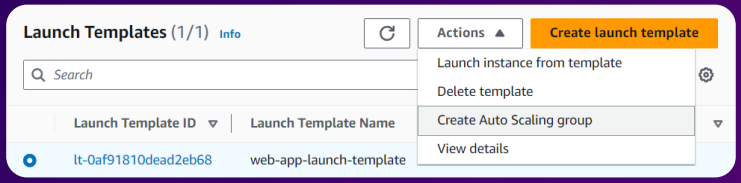
Step 6: Set up the Launch Template

Configure the launch template using the following parameters.



Step 7: Create Auto Scaling group

In the **Launch Templates** section, choose the **web-app-launch-template**, and select Create Auto Scaling group.



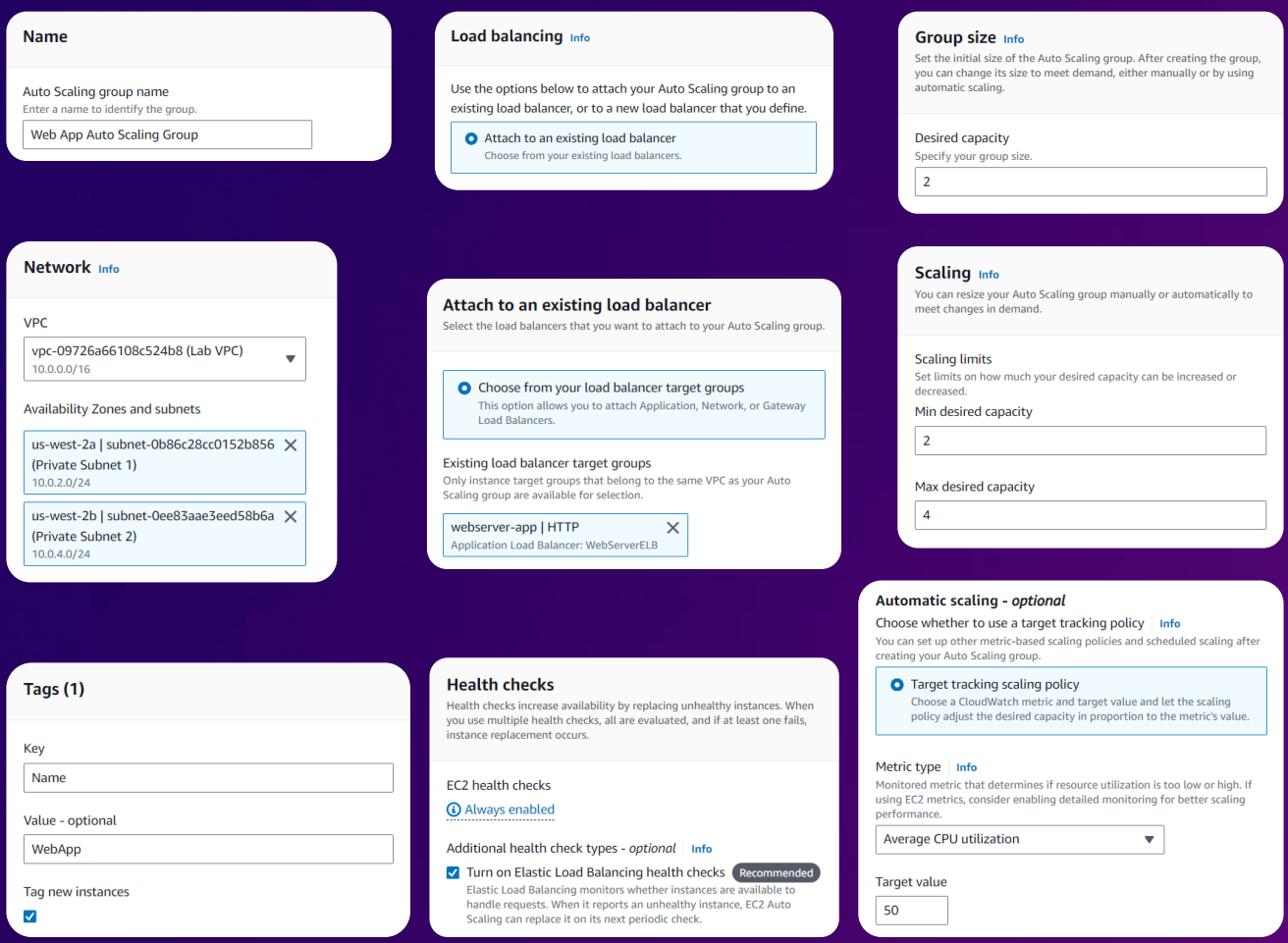
aws re/start

Task 2

Creating an auto scaling environment

Step 8: Set up the Auto Scaling Group

Configure the Auto Scaling Group using the following parameters.



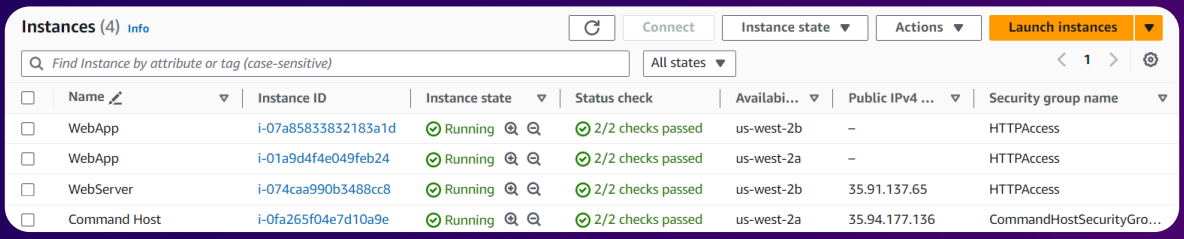
aws re/start

Task 3

Verifying the auto scaling configuration

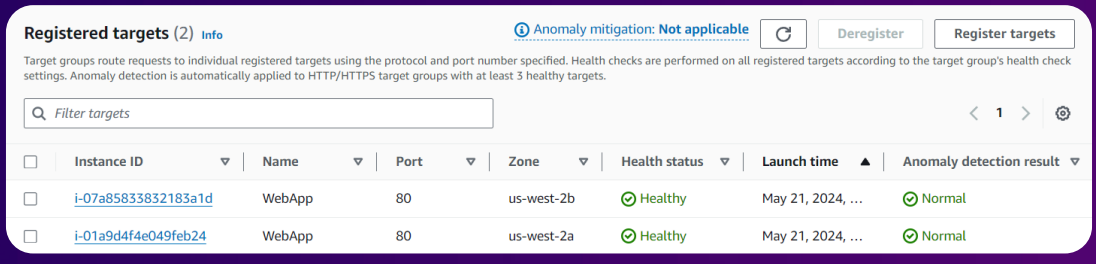
Step 1: Review Auto Scaling

Navigate to the **Instances** section, two new instances named **WebApp** have been created as part of your Auto Scaling group.



Step 2: Review Health status

Navigate to the **Target Groups** section, and select the **webserver-app** target group. In the **Registered targets** section, review the Healthy **Health status** of the two instances.



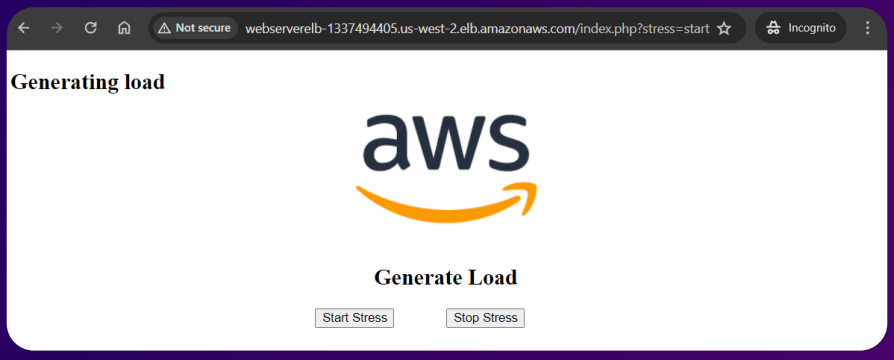
aws re/start

Task 4

Testing auto scaling configuration

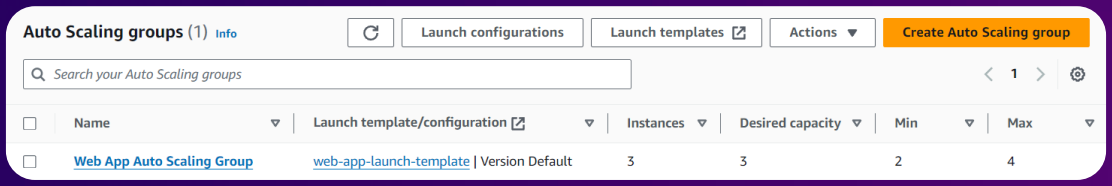
Step 1: Start Stress

Access web page using the load balancer DNS name, and choose **Start Stress**. This step calls the application **stress** in the background, which causes the CPU utilization on the instance that serviced this request to spike to 100 percent.



Step 2: Review Auto Scaling Groups

Navigate to the **Auto Scaling Groups** section, and select the Web App Auto Scaling Group.



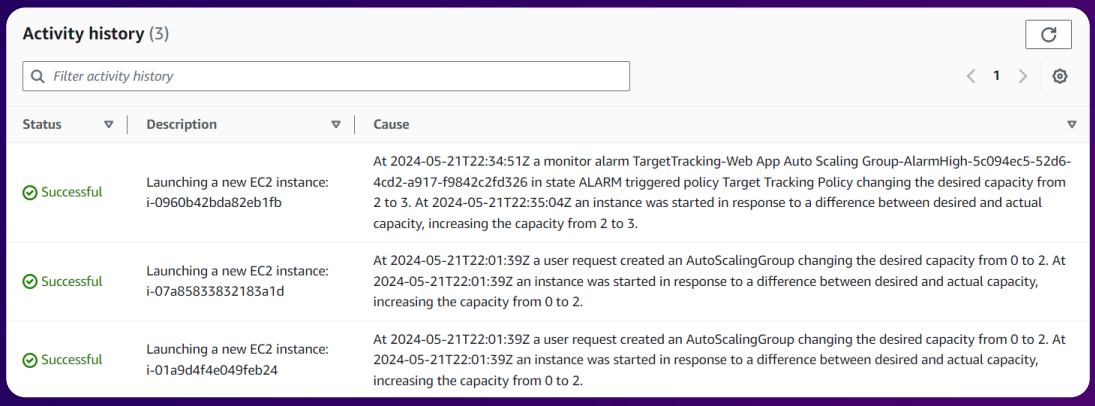
aws re/start

Task 4

Testing auto scaling configuration

Step 3: Review Activity history

Choose the **Activity** tab, and review the **Activity history** section. The Auto Scaling group added a new instance. This occurs because Amazon CloudWatch detected that the average CPU utilization of your Auto Scaling group exceeded 50 percent, and your scale-up policy has been invoked in response.



Activity history (3)

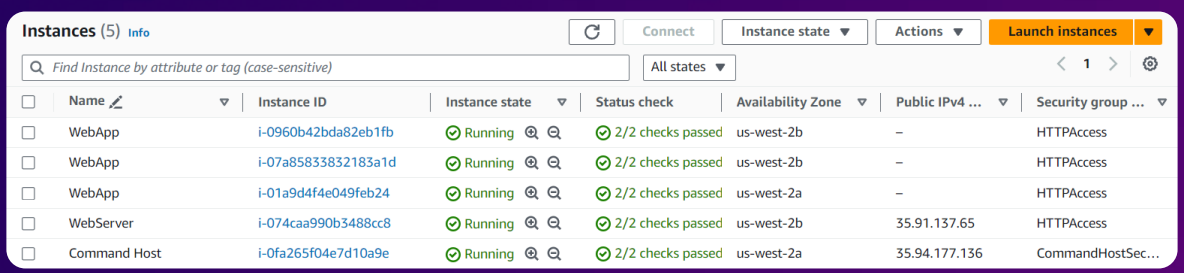
Filter activity history

1

Status	Description	Cause
Successful	Launching a new EC2 instance: i-0960b42bda82eb1fb	At 2024-05-21T22:34:51Z a monitor alarm TargetTracking-Web App Auto Scaling Group-AlarmHigh-5c094ec5-52d6-4cd2-a917-f9842c2fd326 in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 2 to 3. At 2024-05-21T22:35:04Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 2 to 3.
Successful	Launching a new EC2 instance: i-07a85833832183a1d	At 2024-05-21T22:01:39Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 2. At 2024-05-21T22:01:39Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 2.
Successful	Launching a new EC2 instance: i-01a9d4f4e049feb24	At 2024-05-21T22:01:39Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 2. At 2024-05-21T22:01:39Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 2.

Step 4: Review new instances

Review the new launched instance on the EC2 Dashboard.



Instances (5) Info

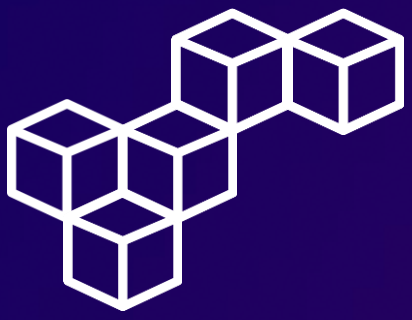
Connect | Instance state | Actions | Launch instances

Find Instance by attribute or tag (case-sensitive) | All states

1

Name	Instance ID	Instance state	Status check	Availability Zone	Public IPv4 ...	Security group ...
WebApp	i-0960b42bda82eb1fb	Running	2/2 checks passed	us-west-2b	–	HTTPAccess
WebApp	i-07a85833832183a1d	Running	2/2 checks passed	us-west-2b	–	HTTPAccess
WebApp	i-01a9d4f4e049feb24	Running	2/2 checks passed	us-west-2a	–	HTTPAccess
WebServer	i-074caa990b3488cc8	Running	2/2 checks passed	us-west-2b	35.91.137.65	HTTPAccess
Command Host	i-0fa265f04e7d10a9e	Running	2/2 checks passed	us-west-2a	35.94.177.136	CommandHostSec...

aws re/start



Conclusions

Launching EC2 instances
Launching EC2 instances via AWS CLI provides efficient, scriptable, and repeatable deployment processes.

Creating Custom AMIs
Creating Custom AMIs from the AWS CLI allows for consistent and pre-configured instance setups tailored to specific needs.

Application Load Balancers
Application Load Balancers offer advanced traffic distribution and routing features, enhancing application availability and performance.

Launch templates
Launch templates streamline instance configuration and deployment, ensuring consistency and reducing manual setup effort.

Auto Scaling Groups
Auto Scaling Groups automatically adjust the number of EC2 instances based on demand, optimizing resource utilization and maintaining performance.

Stress tests
Stress tests evaluate system performance under high load, identifying bottlenecks and ensuring reliability and scalability.

aws re/start

Cristhian Becerra

cristhian-becerra-espinoza

+51 951 634 354

cristhianbecerra99@gmail.com

Lima, Peru