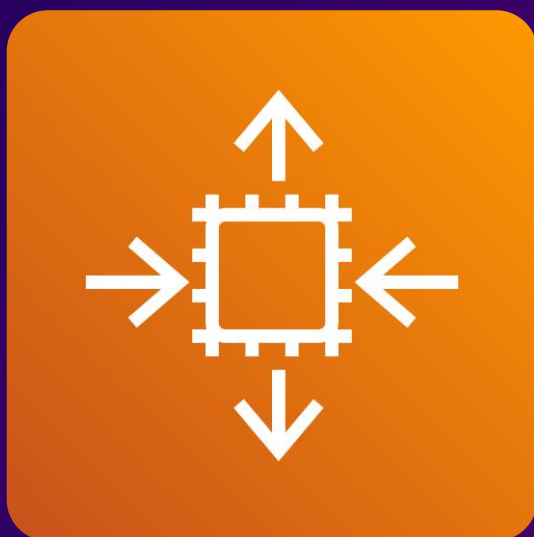




AWS
re:Start
LAB

Using Auto Scaling in AWS



WEEK 9





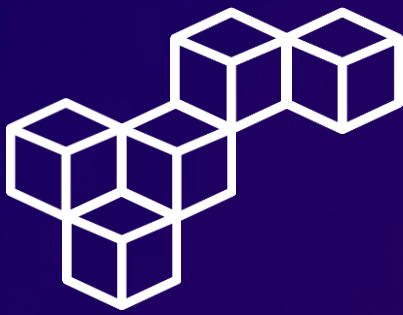
Overview

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.

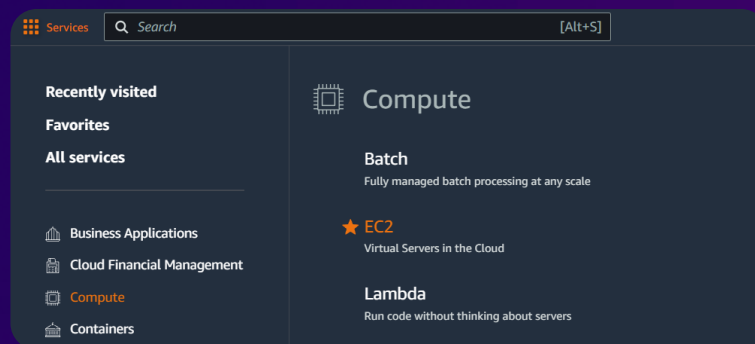


Task 1

Creating a new AMI for Amazon EC2 Auto Scaling

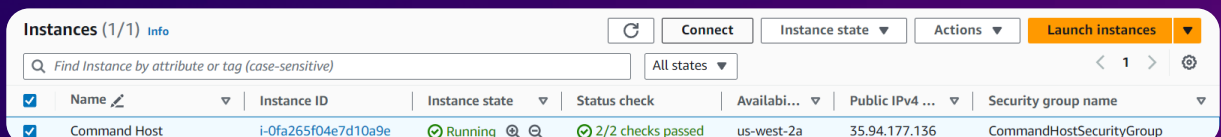
Step 1: Access the EC2 Management Console

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.





Task 1

Creating a new AMI for Amazon EC2 Auto Scaling

Step 3: Confirm Region

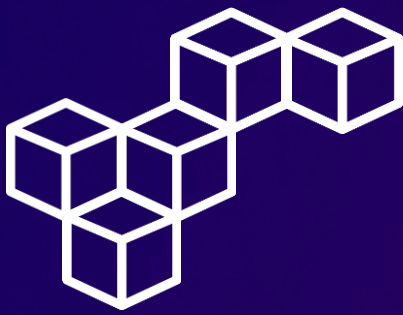
To confirm that the Region in which the **Command Host** instance is running is the same as the lab (the us-west-2 Region), run the following command.

```
[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 ~]$
```



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-0ced0e6c71bc8e17 \
> --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

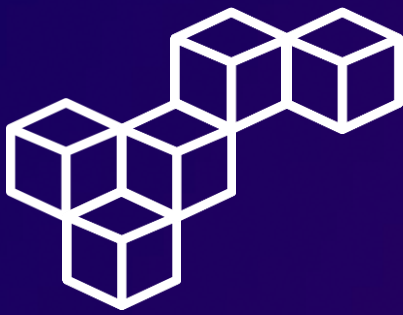
Review the web server installation using the instance public DNS name. The web server appears to be running.



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 ~]$ █
```

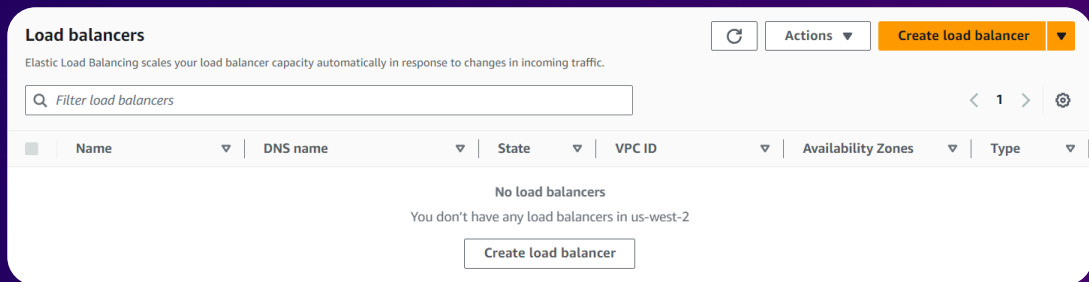



Task 2

Creating an auto scaling environment

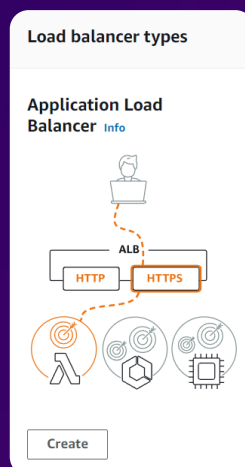
Step 1: Create load balancer

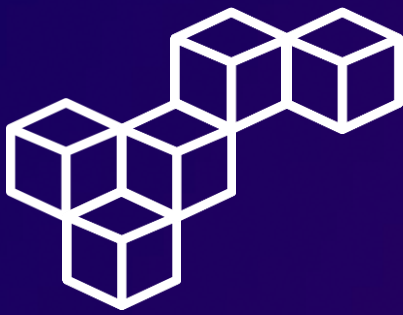
Navigate to the **Load balancers** section, and select [Create load balancer](#).



Step 2: Create Application Load Balancer

In the **Load balancer types** section, select **Application Load Balancer**, and choose [Create](#).





Task 2

Creating an auto scaling environment

Step 3: Set up the Application Load Balancer

Configure the Load Balancer using the following parameters.

Basic configuration

Load balancer name
Name must be unique within your AWS account and can't be changed after the load balancer is created.

WebServerELB

Network mapping [Info](#)

The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

VPC [Info](#)
Select the virtual private cloud (VPC) for your targets.

Lab VPC
vpc-09726a66108c524b8
IPv4 VPC CIDR: 10.0.0.0/16

Mappings [Info](#)
Select at least two Availability Zones and one subnet per zone. The load balancer routes traffic to targets in these Availability Zones only.

☒ us-west-2a (usw2-az1)
Subnet
subnet-07c9f21c3ac746ad9 Public Subnet 1

☒ us-west-2b (usw2-az2)
Subnet
subnet-07ec7c7fc0e48d61e Public Subnet 2

Security groups [Info](#)

A security group is a set of firewall rules that control the traffic to your load balancer. Select an existing security group, or you can [create a new security group](#).

Security groups
Select up to 5 security groups

HTTPAccess
sg-0ced0e6c71bc8e17 VPC: vpc-09726a66108c524b8

Basic configuration

Settings in this section can't be changed after the target group is created.

Choose a target type

☒ Instances

- Supports load balancing to instances within a specific VPC.
- Facilitates the use of [Amazon EC2 Auto Scaling](#) to manage and scale your EC2 capacity.

Target group name
webserver-app

Health checks

The associated load balancer periodically sends requests, per the settings below, to the registered targets to test their status.

Health check protocol
HTTP

Health check path
Use the default path of "/" to perform health checks on the root, or specify a custom path if preferred.
/index.php

Listeners and routing [Info](#)

A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

▼ Listener HTTP:80

Protocol	Port	Default action
HTTP	: 80 1-65535	Forward to webserver-app Target type: Instance, IPv4

[Create target group](#)



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.

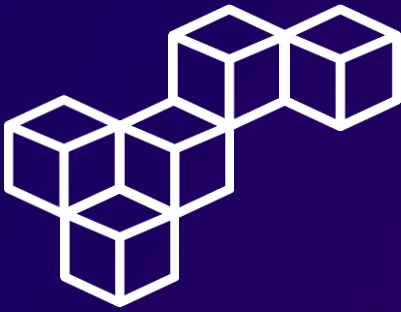
Load balancers (1)							
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.							
<input type="text" value="Filter load balancers"/>							
< 1 > ⚙							
<input type="checkbox"/>	Name	DNS name	State	VPC ID	Availability Zones	Type	
<input type="checkbox"/>	WebServerELB	WebServerELB-1337494405.us-west-2.elb.amazonaws.com	Active	vpc-0972...	2 Availability Zones	application	

Step 5: Create launch template

Navigate to the **Launch Templates** section, and select [Create launch template](#).

New launch template

Create launch template



Task 2

Creating an auto scaling environment

Step 6: Set up the Launch Template

Configure the launch template using the following parameters.

Launch template name and description

Launch template name - *required*

Template version description

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

▼ Network settings [Info](#)

Firewall (security groups) [Info](#)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

☒ Select existing security group ☐ Create security group

Security groups [Info](#)

HTTPAccess sg-0cede0e6c71bc8e17 ✕

VPC: vpc-09726a66108c524b8

[Compare security group rules](#)

Amazon Machine Image (AMI)

WebServerAMI

ami-0d686c1781e0eeceb

2024-05-21T20:42:48.000Z Virtualization: hvm ENA enabled: true Root device type: ebs

Architecture AMI ID

x86_64 ami-0d686c1781e0eeceb

▼ Instance type [Info](#) [Get advice](#) [Advanced](#)

Instance type

t3.micro

Family: t3 2 vCPU 1 GiB Memory Current generation: true

On-Demand SUSE base pricing: 0.0104 USD per Hour

On-Demand Windows base pricing: 0.0196 USD per Hour

On-Demand RHEL base pricing: 0.0704 USD per Hour

On-Demand Linux base pricing: 0.0104 USD per Hour

☐ All generations [Compare instance types](#)

▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name

[Create new key pair](#)

Step 7: Create Auto Scaling group

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

Launch Templates (1/1) [Info](#)

Launch Template ID	Launch Template Name
lt-0af91810dead2eb68	web-app-launch-template

Actions ▲

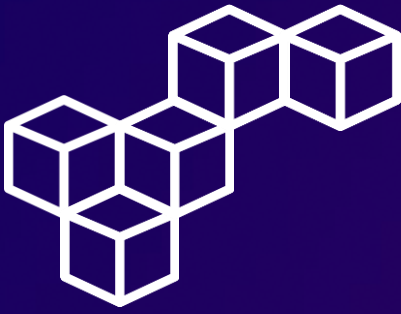
Create launch template

Launch instance from template

Delete template

Create Auto Scaling group

View details



Task 2

Creating an auto scaling environment

Step 8: Set up the Auto Scaling Group

Configure the Auto Scaling Group using the following parameters.

Name

Auto Scaling group name

Enter a name to identify the group.

Web App Auto Scaling Group

Load balancing [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.

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

Group size [Info](#)

Set the initial size of the Auto Scaling group. After creating the group, you can change its size to meet demand, either manually or by using automatic scaling.

Desired capacity

Specify your group size.

2

Network [Info](#)

VPC

vpc-09726a66108c524b8 (Lab VPC)
10.0.0.0/16

Availability Zones and subnets

us-west-2a | subnet-0b86c28cc0152b856 X
(Private Subnet 1)
10.0.2.0/24

us-west-2b | subnet-0ee83aae3eed58b6a X
(Private Subnet 2)
10.0.4.0/24

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.

Existing load balancer target groups

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

webserver-app | HTTP X
Application Load Balancer: WebServerELB

Scaling [Info](#)

You can resize your Auto Scaling group manually or automatically to meet changes in demand.

Scaling limits

Set limits on how much your desired capacity can be increased or decreased.

Min desired capacity

2

Max desired capacity

4

Tags (1)

Key

Name

Value - optional

WebApp

Tag new instances

☒

Health checks

Health checks increase availability by replacing unhealthy instances. When you use multiple health checks, all are evaluated, and if at least one fails, instance replacement occurs.

EC2 health checks

☒ Always enabled

Additional health check types - optional [Info](#)

☒ Turn on Elastic Load Balancing health checks **Recommended**

Elastic Load Balancing monitors whether instances are available to handle requests. When it reports an unhealthy instance, EC2 Auto Scaling can replace it on its next periodic check.

Automatic scaling - optional

Choose whether to use a target tracking policy [Info](#)

You can set up other metric-based scaling policies and scheduled scaling after creating your Auto Scaling group.

☒ Target tracking scaling policy
Choose a CloudWatch metric and target value and let the scaling policy adjust the desired capacity in proportion to the metric's value.

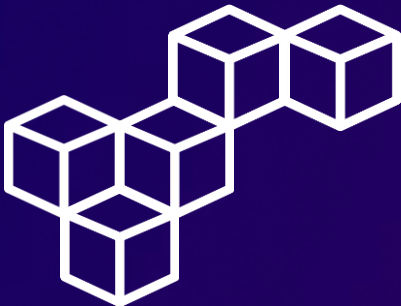
Metric type [Info](#)

Monitored metric that determines if resource utilization is too low or high. If using EC2 metrics, consider enabling detailed monitoring for better scaling performance.

Average CPU utilization

Target value

50



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.

Instances (4) Info							
<input type="text" value="Find Instance by attribute or tag (case-sensitive)"/>				All states ▾		< 1 > ⚙	
<input type="checkbox"/>	Name ↗	Instance ID	Instance state ▾	Status check	Availabi... ▾	Public IPv4 ... ▾	Security group name ▾
<input type="checkbox"/>	WebApp	i-07a85833832183a1d	Running 🔍 🔍	2/2 checks passed	us-west-2b	–	HTTPAccess
<input type="checkbox"/>	WebApp	i-01a9d4f4e049feb24	Running 🔍 🔍	2/2 checks passed	us-west-2a	–	HTTPAccess
<input type="checkbox"/>	WebServer	i-074caa990b3488cc8	Running 🔍 🔍	2/2 checks passed	us-west-2b	35.91.137.65	HTTPAccess
<input type="checkbox"/>	Command Host	i-0fa265f04e7d10a9e	Running 🔍 🔍	2/2 checks passed	us-west-2a	35.94.177.136	CommandHostSecurityGro...

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.

Registered targets (2) Info								
Anomaly mitigation: Not applicable 🔍 Deregister Register targets								
Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.								
<input type="text" value="Filter targets"/>								
< 1 > ⚙								
<input type="checkbox"/>	Instance ID	Name	Port	Zone	Health status	Launch time	Anomaly detection result ▾	
<input type="checkbox"/>	i-07a85833832183a1d	WebApp	80	us-west-2b	Healthy 🔍	May 21, 2024, ...	Normal 🔍	
<input type="checkbox"/>	i-01a9d4f4e049feb24	WebApp	80	us-west-2a	Healthy 🔍	May 21, 2024, ...	Normal 🔍	



Task 4

Testing auto scaling configuration

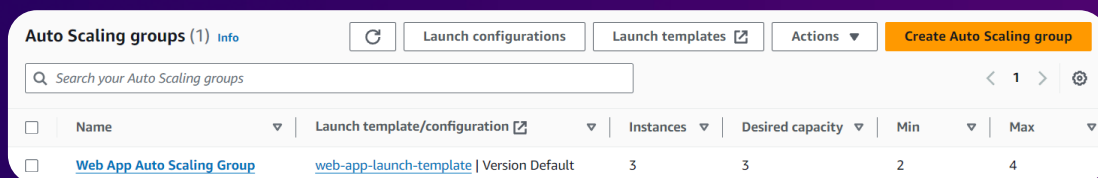
Step 1: Start Stress

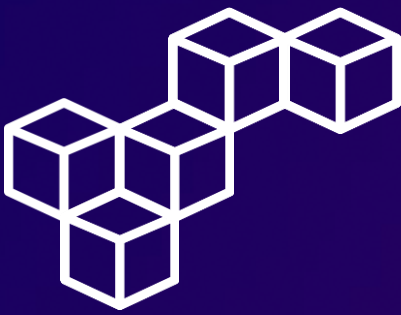
Access web page using the load balancer DNS name, and choose **Start Stress**. This step calls the application **stress** in the background, which causes the CPU utilization on the instance that serviced this request to spike to 100 percent.



Step 2: Review Auto Scaling Groups

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





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)			
<input type="text" value="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)

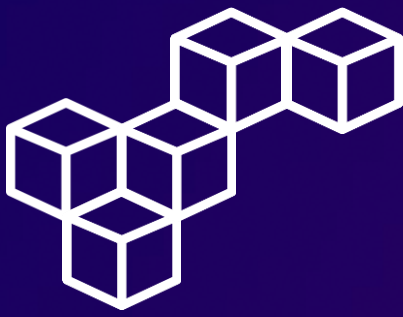
All states

<

1

>

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



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.



Cristhian Becerra



[cristhian-becerra-espinoza](#)



+51 951 634 354



cristhianbecerra99@gmail.com



Lima, Peru

