

# LAB: EC2 ECS Cluster with Capacity Providers

## You need:

- An AWS Account

**Duration of the Lab:** 30 Minutes.

**Difficulty:** hard

## Create an EC2 ECS Cluster

Create a new EC2 Launch Type Cluster:

### Select cluster template

The following cluster templates are available to simplify cluster creation. Additional configuration and integrations can be added later.

**Networking only**

Resources to be created:

- Cluster
- VPC (optional)
- Subnets (optional)

**Powered by AWS Fargate**

**EC2 Linux + Networking**

Resources to be created:

- Cluster
- VPC
- Subnets
- Auto Scaling group with Linux AMI

**EC2 Windows + Networking**

Resources to be created:

- Cluster
- VPC
- Subnets
- Auto Scaling group with Windows AMI

\*Required

Cancel

Next step

Give the cluster a name, e.g. "myec2cluster", but this time choose an empty cluster:

### Configure cluster

**Cluster name\***  ⓘ

☒ Create an empty cluster

**Tags**

Key	Value
<input type="text" value="Add key"/>	<input type="text" value="Add value"/>

**CloudWatch Container Insights**

CloudWatch Container Insights is a monitoring and troubleshooting solution for containerized applications and microservices. It collects, aggregates, and summarizes compute utilization such as CPU, memory, disk, and network; and diagnostic information such as container restart failures to help you isolate issues with your clusters and resolve them quickly. [Learn more](#)

**CloudWatch Container Insights** ☐ Enable Container Insights

\*Required

[Cancel](#) [Previous](#) [Create](#)

## Create an Auto Scaling Group

Before we can create a capacity provider, we need an ASG.

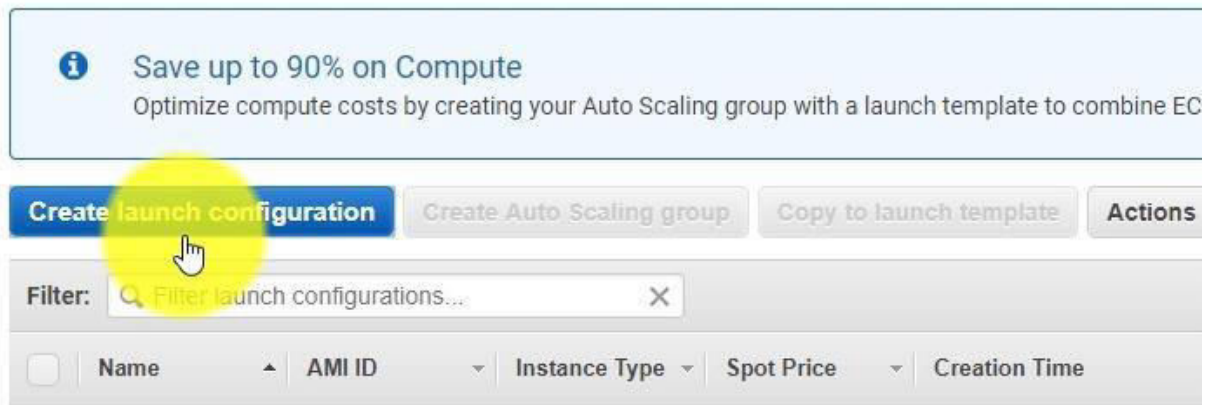
### Create a new Launch Configuration

Open the EC2 Dashboard and find Launch Configurations in the left panel:

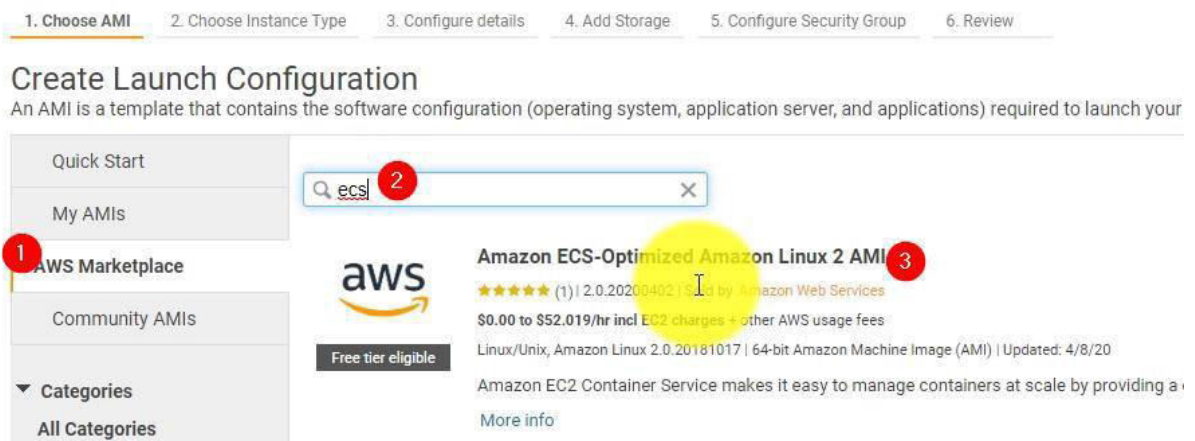


Create a new Launch Configuration:

## Complete AWS ECS DevOps Masterclass for Beginners



Find an Amazon ECS optimized AMI from the AWS Marketplace:



Select a t2.micro instance.

Give the Launch Configuration a name and select the ecsInstanceRole which was hopefully already created earlier by the ecs cluster lab. Then also add a user-data to register the EC2 instances in the right cluster:

```
#!/bin/bash
echo ECS_CLUSTER=myec2cluster >> /etc/ecs/ecs.config
```

## Complete AWS ECS DevOps Masterclass for Beginners

1. Choose AMI 2. Choose Instance Type 3. Configure details 4. Add Storage 5. Configure Security Group 6. Review

### Create Launch Configuration

**Name** ⓘ

**Purchasing option** ⓘ ☐ Request Spot Instances

**IAM role** ⓘ

**Monitoring** ⓘ ☐ Enable CloudWatch detailed monitoring  
[Learn more](#)

▼ **Advanced Details**

**Kernel ID** ⓘ

**RAM Disk ID** ⓘ

**User data** ⓘ ☒ As text ☐ As file ☐ Input is already base64 encoded

```
#!/bin/bash
echo ECS_CLUSTER=myec2cluster >> /etc/ecs/ecs.config
```

**IP Address Type** ⓘ ☒ Only assign a public IP address to instances launched in the default VPC and subnet. (default)  
☐ Assign a public IP address to every instance.  
☐ Do not assign a public IP address to any instances.  
Note: this option only affects instances launched into an Amazon VPC

Skip to review and Create launch configuration, you can have, but you don't need to have SSH access for the remaining lab:

1. Choose AMI 2. Choose Instance Type 3. Configure details 4. Add Storage 5. Configure Security Group 6. Review

### Create Launch Configuration

Amazon Linux AMI 2.0.20200402 x86\_64 ECS HVM GP2  
Root device type: ebs Virtualization Type: hvm

▼ **Instance Type** [Edit 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** [Edit details](#)

**Name** lc-ecs-ami  
**Purchasing option** On demand  
**EBS Optimized** No  
**Monitoring** No  
**IAM role** ecsInstanceRole  
**Tenancy** Shared tenancy (multi-tenant hardware)  
**Kernel ID** Use default  
**RAM Disk ID** Use default  
**User data** iY6YmUL2Jhc7gK2VhnbY8FQ1NFQ6vXU18PUJ1wHrVfM=NuSDN0ZZxgFjAgL2V0ryRy3MAZWNuLmhvbmZpZw==  
**IP Address Type** Only assign a public IP address to instances launched in the default VPC and subnet. (default)

► **Storage** [Edit storage](#)

▼ **Security Groups** [Edit security groups](#)

**Security group name** Amazon ECS-Optimized Amazon Linux 2 AMI-2.0.20200402-AutoGenByAWSMP  
**Description** This security group was generated by AWS Marketplace and is based on recommended settings for Amazon ECS-Optimized Amazon Linux 2 AMI version 2.0.20200402 provided by Amazon Web Services

Type ⓘ	Protocol ⓘ	Port Range ⓘ	Source ⓘ
SSH	TCP	22	0.0.0.0/0

[Cancel](#) [Previous](#) [Create launch configuration](#)

Then directly proceed to create an Auto Scaling group:

Launch configuration creation status

✓ **Successfully created launch configuration: lc-ecs-ami**  
[View creation log](#)

▼ **View**  
[View your launch configurations](#)  
[View your Auto Scaling groups](#)

► Here are some helpful resources to get you started

[Create an Auto Scaling group using this launch configuration](#) [Close](#)

## Create an Auto Scaling Group

Create the ASG:

## Complete AWS ECS DevOps Masterclass for Beginners

1. Give it a name
2. Start with 0 Instances
3. Select all available Subnets from your default VPC
4. Enable Instance Protection

1. Configure Auto Scaling group details 2. Configure scaling policies 3. Configure Notifications 4. Configure Tags 5. Review

### Create Auto Scaling Group

Group name  1

Launch Configuration

Group size  2

Network  3

Subnet  3

Each instance in this Auto Scaling group will be assigned a public IP address. [\(i\)](#)

▼ Advanced Details

Load Balancing ☐ Receive traffic from one or more load balancers. [Learn about Elastic Load Balancing](#)

Health Check Grace Period  seconds

Monitoring ☐ Amazon EC2 Detailed Monitoring metrics, which are provided at 1 minute frequency, are not enabled for the launch configuration lc-ecs-ami. Instances launched from it will use Basic Monitoring metrics, provided at 5 minute frequency.

Learn more

Instance Protection ☐ Protect From Scale In 4

Service-Linked Role

Then proceed until the ASG is created.

Filter:

Name	Launch Configuration	Instances	Desired	Min	Max	Availability Zones	Default Cooldown	Health Check Grace
asg-cluster	lc-ecs-ami	0	0	0	0	eu-central-1a, eu-central-1c, ...	300	300

Auto Scaling Group: asg-cluster

Launch Configuration

Desired Capacity

Min

Max

Availability Zone(s)

Subnet(s)

Classic Load Balancers

Target Groups

Health Check Type

Health Check Grace Period

Instance Protection

Now edit the ASG to set the maximum capacity to a higher number than 0:

**Edit details - asg-cluster**

**Launch Instances Using** *i*

- ☐ Launch Template
- ☒ Launch Configuration

**Launch Configuration** *i* lc-ecs-ami

**Desired Capacity** *i* 0

**Min** *i* 0

**Max** *i* 10

**Availability Zone(s)** *i* eu-central-1a x eu-central-1b x eu-central-1c x

**Subnet(s)** *i*

- subnet-cfd47ba5(172.31.16.0/20) | Default in eu-central-1a x
- subnet-bc21c8f0(172.31.0.0/20) | Default in eu-central-1c x
- subnet-d5f6eca8(172.31.32.0/20) | Default in eu-central-1b x

Now we need to attach a new Capacity Provider to the Auto Scaling Group.

## Create a Capacity Provider

Go back to your ECS Cluster and open the Capacity Provider Tab:

Clusters > myec2cluster

Cluster: **myec2cluster**

Get a detailed view of the resources on your cluster.

Cluster ARN: arn:aws:ecs:eu-central-1:161952721022:cluster/myec2cluster

Status: **ACTIVE**

Registered container instances: 0

Pending tasks count: 0 Fargate, 0 EC2

Running tasks count: 0 Fargate, 0 EC2

Active service count: 0 Fargate, 0 EC2

Draining service count: 0 Fargate, 0 EC2

Services | Tasks | ECS Instances | Metrics | Scheduled Tasks | Tags | **Capacity Providers**

**Create** Deactivate

Filter in this page

	Capacity Provider Nam...	Type	ASG	Managed Scaling	Managed Instance Pro...	Current Size
No results						

Give it a name, select the ASG you created in the previous step. Leave Managed Scaling to be enabled, set the target capacity to 100% and leave Managed termination protection also enabled:

## Complete AWS ECS DevOps Masterclass for Beginners

 Capacity provider cp-myec2cluster was successfully created.  
[View in cluster](#)

### Create Capacity Provider

Cluster name

myec2cluster

Capacity provider name\*

cp-myec2cluster

The name of the capacity provider. Up to 255 characters are allowed, including letters (upper and lowercase), numbers, underscores, and hyphens. The name cannot be prefixed with "aws", "ecs", or "fargate".

Auto Scaling group

asg-cluster

☐ Manually enter desired Auto Scaling group ARN

You must create an Auto Scaling group before creating a capacity provider. Use the [Amazon EC2 console](#) to create an Auto Scaling group.

Managed scaling

☒ Enabled

☐ Disabled

Target capacity %

100

Managed termination protection

☒ Enabled

☐ Disabled

\*Required

Cancel

Create

## Attach the Capacity Provider to the Cluster

Now you need to update the Cluster and attach the Capacity Provider. Hit "Update Cluster"

Clusters > myec2cluster

Cluster : myec2cluster

Update Cluster

Delete Cluster

Get a detailed view of the resources on your cluster.

Cluster ARN

arn:aws:ecs:eu-central-1:161952721022:cluster/myec2cluster

Status

ACTIVE

Registered container instances

0

Pending tasks count

0 Fargate, 0 EC2

Running tasks count

0 Fargate, 0 EC2

Active service count

0 Fargate, 0 EC2

Draining service count

0 Fargate, 0 EC2

Services

Tasks

ECS Instances

Metrics

Scheduled Tasks

Tags

Capacity Providers

Create

Deactivate

Last updated on April 8, 2020 1:04:59 PM (0m ago)

Filter in this page

<input type="checkbox"/>	Capacity Provider Name...	Type	ASG	Managed Scaling	Managed Instance Pro...	Current Size	Desired Size	Min Size	Max Size
<input type="checkbox"/>	cp-myec2cluster	ASGProvider	asg-cluster	Yes	Yes	0	0	0	10


Add the Capacity Provider we just created:

Cluster myec2cluster

Default capacity provider strategy

Provider 1

cp-myec2cluster



+ Add provider

Save it and go back to the Cluster Dashboard to start a new service.

## Start a Service

In this step we are starting a service to see how the Capacity Provider scales out and in.

Create a new Service:

## Complete AWS ECS DevOps Masterclass for Beginners

Select again the simple apache server task definition (1). Give it a service name (2). Launch 3 tasks initially (3). Leave the rest on default, then hit “Next step”.

### Configure service

A service lets you specify how many copies of your task definition to run and maintain in a cluster. You can optionally use an Elastic Load Balancing load balancer to distribute incoming traffic to containers in your service. Amazon ECS maintains that number of tasks and coordinates task scheduling with the load balancer. You can also optionally use Service Auto Scaling to adjust the number of tasks in your service.

**Capacity provider strategy** Cluster default strategy ⓘ

**Task Definition** Family  
simple-apache-server 1 ⓘ Enter a value

Revision  
2 (latest) ⓘ

**Cluster** myec2cluster ⓘ

**Service name** apacheservice 2 ⓘ

**Service type\*** ☒ REPLICAS ☐ DAEMONS ⓘ

**Number of tasks** 3 ⓘ

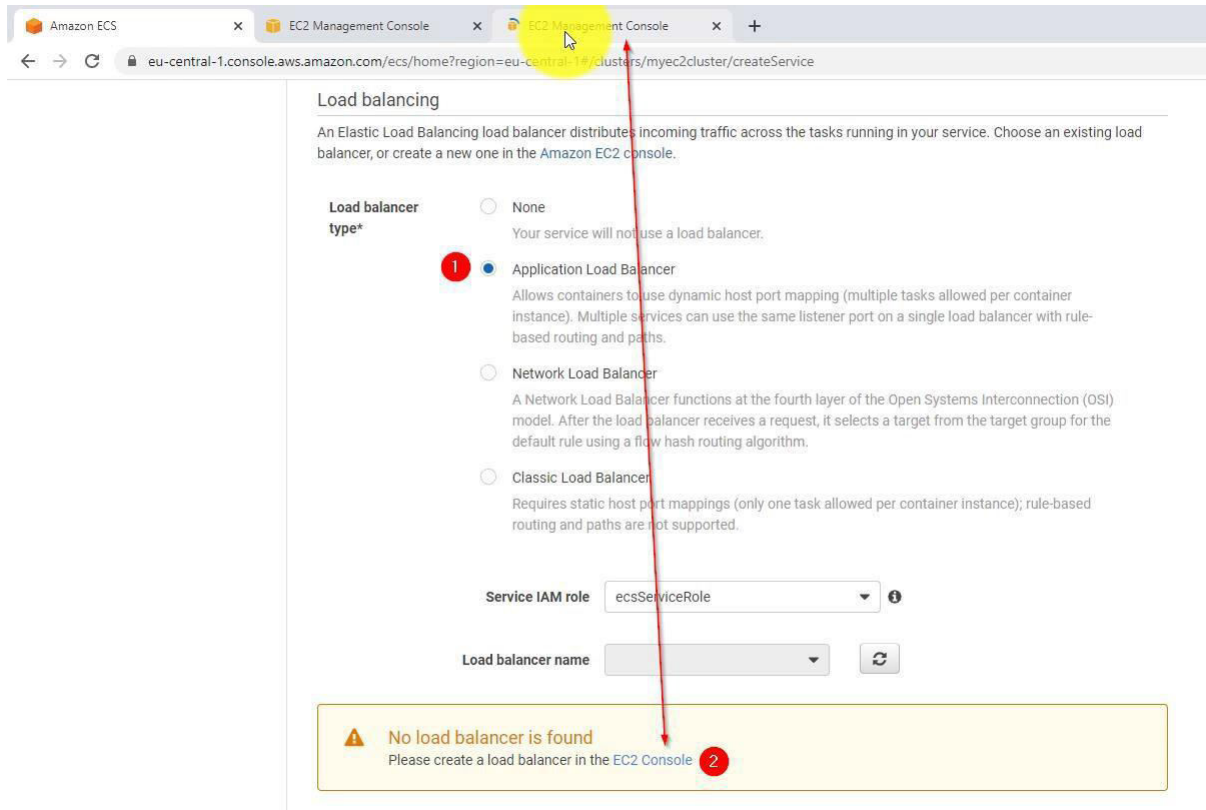
**Minimum healthy percent** 100 ⓘ

**Maximum percent** 200 ⓘ

On the next page add an application load balancer. Again, go to the EC2 Dashboard in a new tab to create an application load balancer first:



## Complete AWS ECS DevOps Masterclass for Beginners



Give the load balancer a name, and place it in all 3 AZ. Create a new target group type instance. Review -> create.

Then go back to your ECS Tab and reload the ALB Drop Down:

## Complete AWS ECS DevOps Masterclass for Beginners

Service IAM role  ⓘ

Load balancer name  ⓘ

Container to load balance

**apachecontainer : 80** [Remove](#) ⓘ

Production listener port\*  ⓘ

Production listener protocol\*  ⓘ

Target group name  ⓘ

Target group protocol

Target type  ⓘ

Path pattern

Path pattern: The first path pattern for a listener is the default path (/), which accepts all traffic that does not match another rule. You can later add additional patterns and priority values to this listener for other services.

Health check path  ⓘ

Additional health check options can be configured in the ELB console after you create your service.

Remove the Service Discovery for this example and hit “Next Step”.

On the next page add a Service Auto Scaling based on the CPU usage:

Minimum Tasks: 3, Desired: 3, Maximum: 9

Add a CPU Target Tracking with 50%:

## Complete AWS ECS DevOps Masterclass for Beginners

Minimum number of tasks  ⓘ

Automatic task scaling policies you set cannot reduce the number of tasks below this number.

Desired number of tasks  ⓘ

Maximum number of tasks  ⓘ

Automatic task scaling policies you set cannot increase the number of tasks above this number.

IAM role for Service Auto Scaling  ⓘ

### Automatic task scaling policies

Scaling policy type ☒ Target tracking ⓘ  
☐ Step scaling

Policy name\*  ⓘ

ECS service metric\*  ⓘ

Target value\*  ⓘ

Scale-out cooldown period  seconds between scaling actions ⓘ

Scale-in cooldown period  seconds between scaling actions ⓘ

Disable scale-in ☐ ⓘ

\*Required

Cancel

Previous

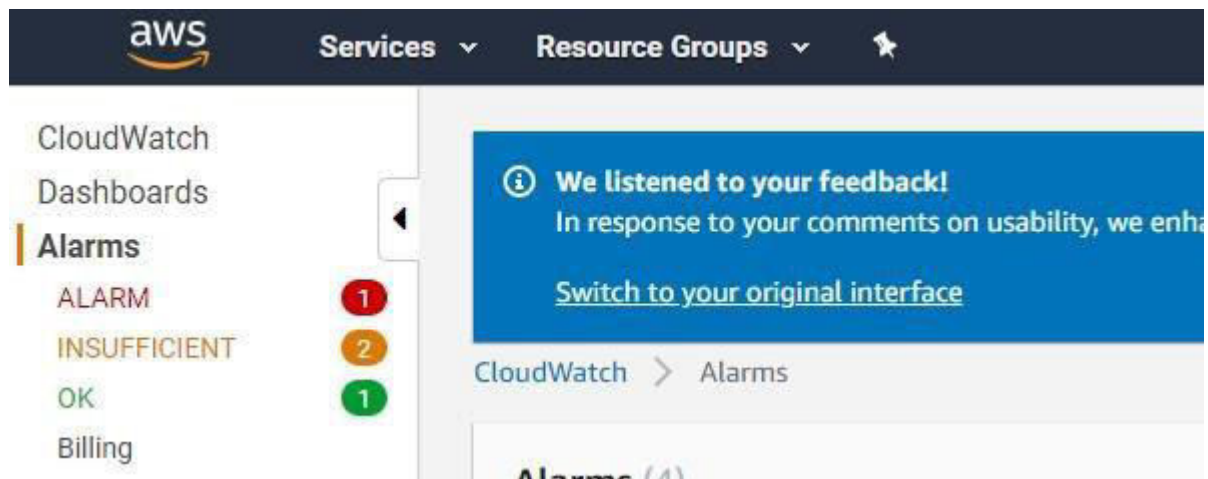
Next step

Then create the service.

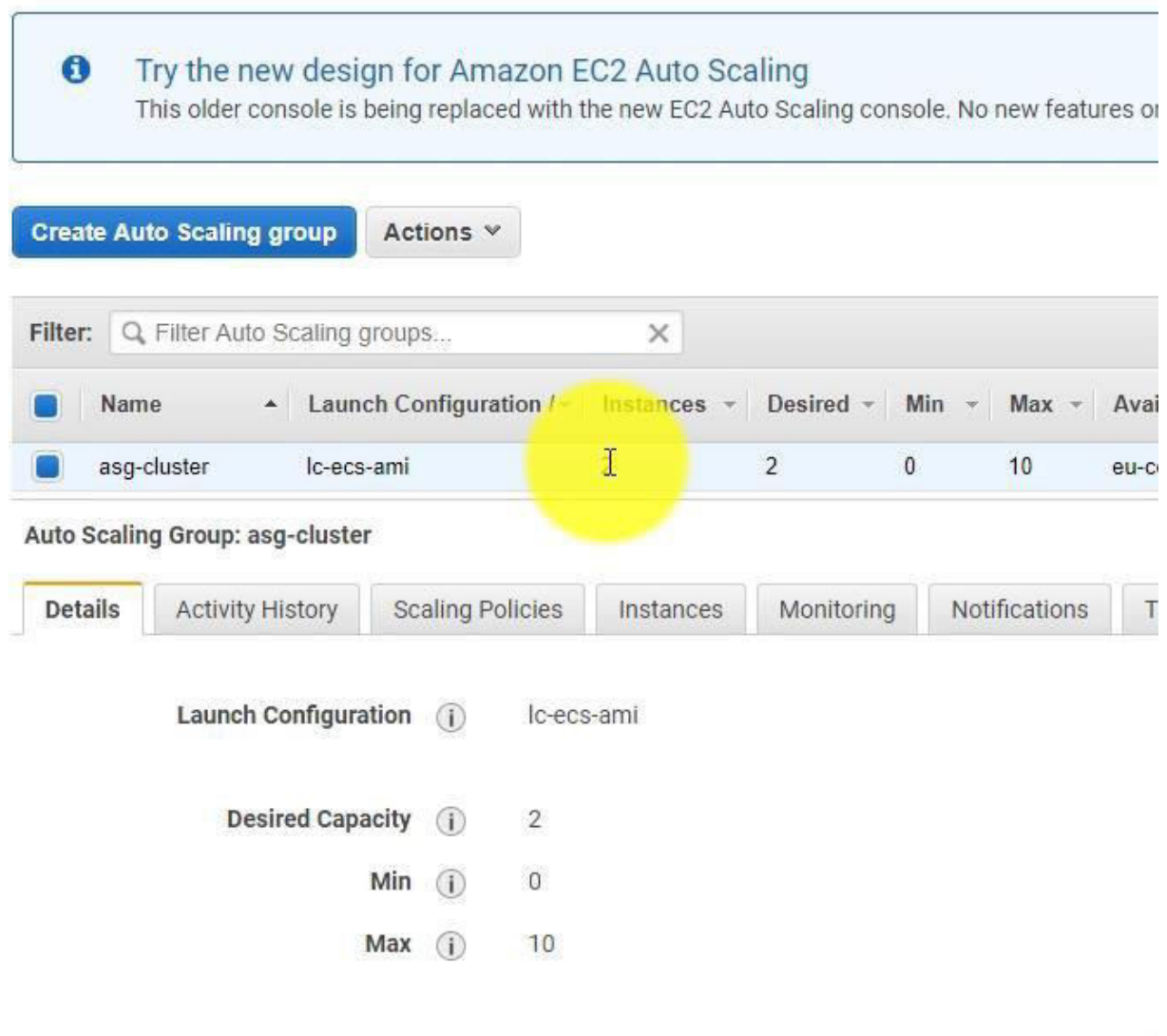
### Watch the Scaling

The Capacity Provider needs a few minutes so the target capacity is tracked correctly. You can follow this also in the CloudWatch Alarms.

Open the CloudWatch Dashboard and have a look on the Alarms on the left side:



Have also a look at the ASG in the EC2 Dashboard. It will try to modify the desired capacity so that EC2 Instances are started by the ASG:



Have a look at the EC2 Dashboard, it automatically started the EC2 Instances through the ASG:

## Complete AWS ECS DevOps Masterclass for Beginners

Launch Instance									
Connect Actions									
Filter by tags and attributes or search by keyword									
Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	
	i-024b2c06f20b96fd7	t2.micro	eu-central-1a	terminated		None			
	i-047608b3eb6d60a...	t2.micro	eu-central-1b	terminated		None			
	i-049b5dac8020d5d6f	t2.micro	eu-central-1c	terminated		None			
	i-064c35045b3be484	t2.micro	eu-central-1a	terminated		None			
	i-0677527c90052bb06	t2.micro	eu-central-1b	terminated		None			
	i-08007e1de4b0217...	t2.micro	eu-central-1c	terminated		None			
	i-0b5dec07e1f948f8	t2.micro	eu-central-1b	terminated		None			
	i-04ee5e034bacd76c	t2.micro	eu-central-1b	running	Initializing	None	ec2-3-127-234-223.eu-...	3.127.234.223	
	i-06504eca07ab689d	t2.micro	eu-central-1a	running	Initializing	None	ec2-10-156-79-46.eu-c...	10.156.79.46	

If you go back to the ECS Dashboard, to your cluster, and choose the ECS Instances, you will see the two EC2 Instances should have registered as Compute Resources for the Cluster:

Clusters > myec2cluster

Cluster : myec2cluster

Get a detailed view of the resources on your cluster.

Cluster ARN: arn:aws:ecs:eu-central-1:161952721022:cluster/myec2cluster

Status: ACTIVE

Registered container instances: 2

Pending tasks count: 0 Fargate, 3 EC2

Running tasks count: 0 Fargate, 0 EC2

Active service count: 0 Fargate, 1 EC2

Draining service count: 0 Fargate, 0 EC2

Services Tasks **ECS Instances** Metrics Scheduled Tasks Tags Capacity Providers

Add additional ECS Instances using Auto Scaling or Amazon EC2.

Actions

Last updated on April 8, 2020 1:10:58 PM (0m ago)

Status: ALL ACTIVE DRAINING

Filter by attributes (click or press down arrow to view filter options)

Container Instance	EC2 Instance	Availability Zone	Agent Connect...	Status	Running tasks c...	CPU available ...	Memory availab...	Agent version	Docker version ...
<input type="checkbox"/> 0475b717-1990-450e-96f3...	i-0f4ee5e034bac...	eu-central-1b	true	ACTIVE	0	768	471	1.39.0	18.09.9-ce
<input type="checkbox"/> c958d1e1-8e1c-42ee-b92a...	i-0f6504eca07a...	eu-central-1a	true	ACTIVE	0	768	471	1.39.0	18.09.9-ce

If you check the service, then it should have started the Containers successfully:

Clusters > myec2cluster

Cluster : myec2cluster

Get a detailed view of the resources on your cluster.

Cluster ARN: arn:aws:ecs:eu-central-1:161952721022:cluster/myec2cluster

Status: ACTIVE

Registered container instances: 2

Pending tasks count: 0 Fargate, 1 EC2

Running tasks count: 0 Fargate, 2 EC2

Active service count: 0 Fargate, 1 EC2

Draining service count: 0 Fargate, 0 EC2

Services Tasks **ECS Instances** Metrics Scheduled Tasks Tags Capacity Providers

Run new Task Stop Stop All Actions

Last updated on April 8, 2020 1:14:08 PM (0m ago)

Desired task status: Running Stopped

Filter in this page Launch type: ALL

1-3 Page size: 50

Task	Task definition	Container instance	Last status	Desired status	Started By	Group	Launch type	Platform version
<input type="checkbox"/> 035d0ba3-ddf1-42ac-94...	simple-apache-server:2	c958d1e1-8e1c-42ee-b...	RUNNING	RUNNING	ecs-svc/848524103954...	service:apacheservice	EC2	--
<input type="checkbox"/> 180e8791-ea5c-462c-b...	simple-apache-server:2	--	PROVISIONING	RUNNING	ecs-svc/848524103954...	service:apacheservice	EC2	--
<input type="checkbox"/> 53d94d25-bb9e-4f78-b...	simple-apache-server:2	0475b717-1990-450e-9...	RUNNING	RUNNING	ecs-svc/848524103954...	service:apacheservice	EC2	--

Hint: If it does not start the containers successfully, have a look at the stopped containers failed message.

## Complete AWS ECS DevOps Masterclass for Beginners

Check again the CloudWatch Alarms – they should be OK now. And if you drill down into the “AlarmHigh” you should see a spike where it went up to 200, then the ASG started the EC2 Instances and the Undercapacity was successfully mitigated by starting more compute resources, bringing the target capacity back to 100%:



## Open the Website from Apache

Now, let's see the configuration for the Load Balancer. We can try and open up the DNS of the Load Balancer and see if we can reach out apache containers:

Open the EC2 Dashboard -> Load Balancers and copy the DNS of your ALB:

The screenshot shows the AWS Management Console for the Elastic Load Balancing (ALB) service. The top bar has a 'Create Load Balancer' button and an 'Actions' dropdown. Below the bar is a search bar labeled 'Filter by tags and attributes or search by keyword'. A table lists the load balancers:

Name	DNS name	State	VPC ID
ecs-lb	ecs-lb-297537071.eu-central-1.elb...	active	vpc-6570b40f

Below the table, the 'Load balancer: ecs-lb' section is expanded. It shows tabs for 'Description', 'Listeners', 'Monitoring', 'Integrated services', and 'Tags'. The 'Description' tab is selected, showing the 'Basic Configuration' section:

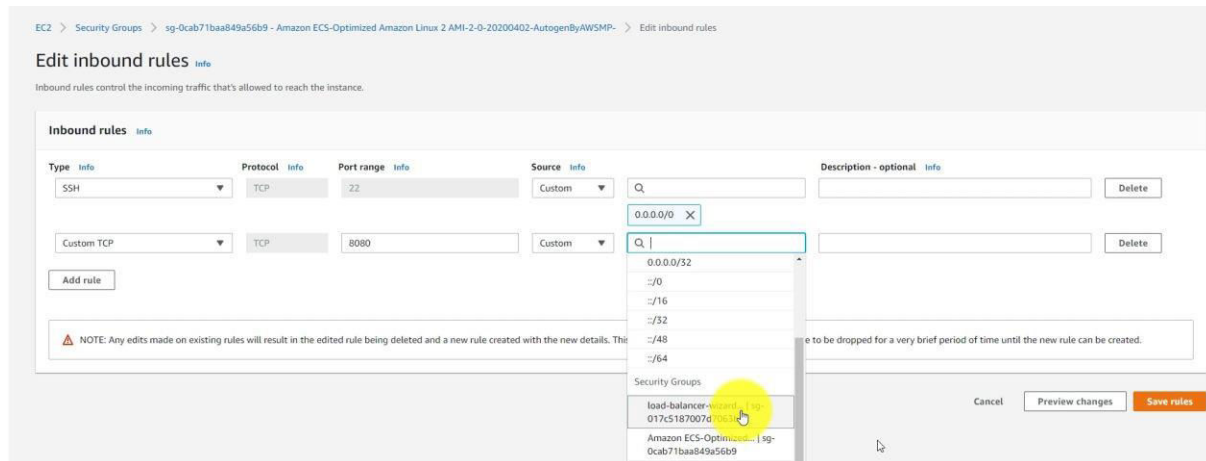
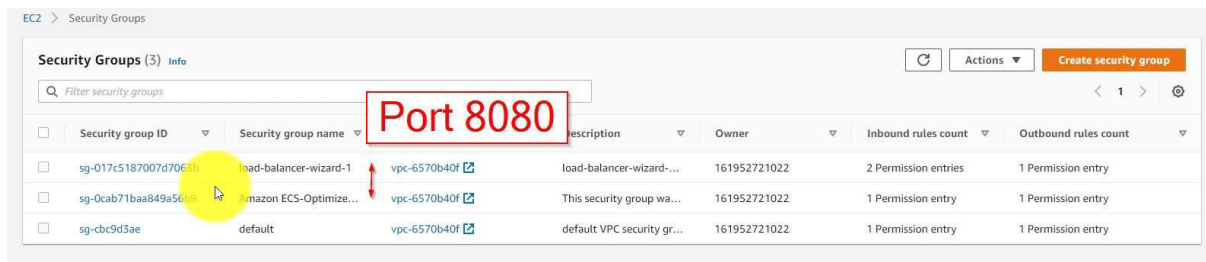
Property	Value
Name	ecs-lb
ARN	arn:aws:elasticloadbalancing:eu-central-1:161952721022:loadbalancer/app/ecs-lb/f2269ac
DNS name	ecs-lb-297537071.eu-central-1.elb.amazonaws.com (A Record)
State	active

Then open a new tab and try to open this.

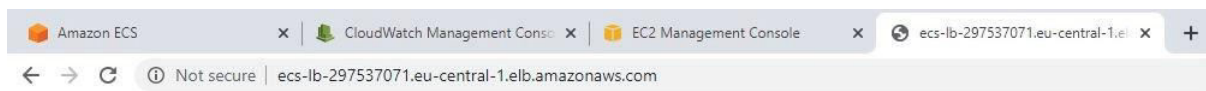
It times out, why? ... security groups!

## Complete AWS ECS DevOps Masterclass for Beginners

We have to add an inbound rule to the Security group for our EC2 Instances, so that the Application Load Balancer is allowed to access port 8080 on the containers:

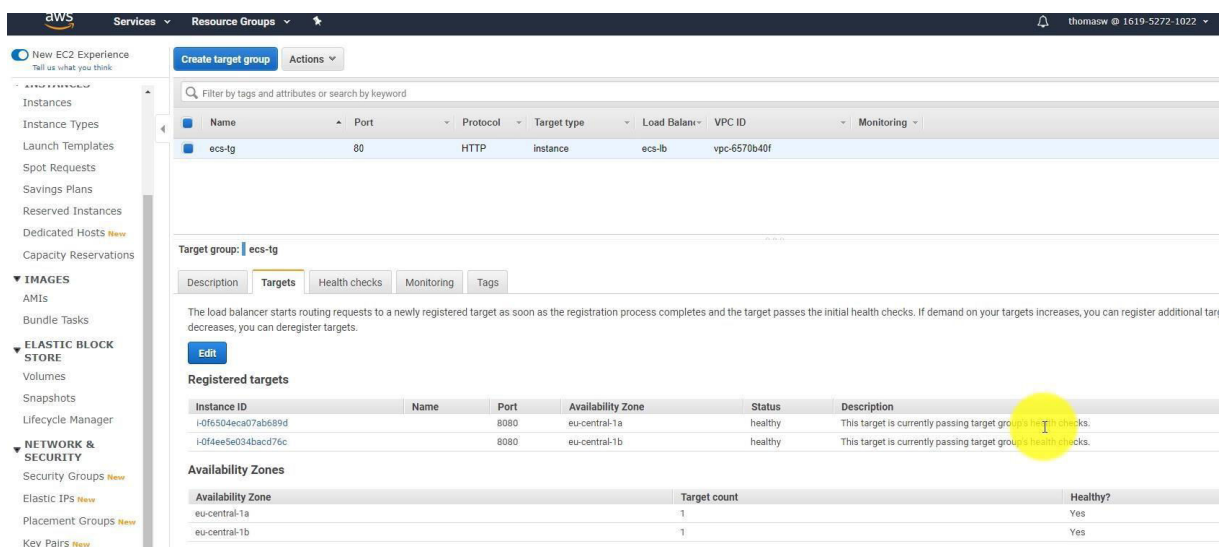


Try again to reload the browser tab, and it should work:



**It works!**

Also check out the Target Group of the Load Balancer and make sure the targets become healthy (might take a few minutes after updating the security groups):



It takes a while until the third instance is really running. The capacity provider is slowly ramping up the capacity until the desired capacity is met.



## Complete AWS ECS DevOps Masterclass for Beginners

Clusters > myec2cluster

### Cluster : myec2cluster

Get a detailed view of the resources on your cluster.

Cluster ARN: arn:aws:ecs:eu-central-1:161952721022:cluster/myec2cluster

Status: **ACTIVE**

Registered container instances: 3

Pending tasks count: 0 Fargate, 1 EC2

Running tasks count: 0 Fargate, 2 EC2

Active service count: 0 Fargate, 1 EC2

Draining service count: 0 Fargate, 0 EC2

Services | **Tasks** | ECS Instances | Metrics | Scheduled Tasks | Tags | Capacity Providers

Add additional ECS Instances using Auto Scaling or Amazon EC2.

Actions

Status: **ALL** ACTIVE DRAINING

Filter by attributes (click or press down arrow to view filter options)

Container instance	EC2 Instance	Availability Zone	Agent Connected	Status	Running tasks count	CPU available	Memory available	Agent version	Docker version
0475b717-1990-450e-96f3-...	i-0f4ee5e034bac...	eu-central-1b	true	ACTIVE	1	768	471	1.39.0	18.09.9-ce
a5a36280-068f-4e71-97de-...	i-02eb702fd6ce...	eu-central-1c	true	ACTIVE	0	1024	983	1.39.0	18.09.9-ce
c958d1e1-8e1c-42ee-b92a-...	i-0f6504eca07a...	eu-central-1a	true	ACTIVE	1	768	471	1.39.0	18.09.9-ce

As soon as everything is running, make one last check at the logs if everything looks ok. Reload the apache website via the load balancer a few times and make sure the load is spread evenly across the instances:

Clusters > myec2cluster > Service: apacheservice

### Service : apacheservice

Cluster: myec2cluster

Status: **ACTIVE**

Task definition: simple-apache-server:2

Service type: REPLICAS

Launch type: Launch type

Service role: ecsServiceRole

Desired count: 3

Pending count: 0

Running count: 3

Details | Tasks | Events | Auto Scaling | Deployments | Metrics | Tags | **Logs**

Task status: **RUNNING** STOPPED

Filter logs

Last updated on April 8, 2020 1:17:19 PM (0m ago)

Timestamp (UTC+00:00)	Message	Task
2020-04-08 13:17:16	172.31.31.145 -- [08/Apr/2020:11:17:16 +0000] "GET / HTTP/1.1" 304 -	U3500b3-d0f1-42ac-9404-34b1510d4281
2020-04-08 13:17:16	172.31.31.145 -- [08/Apr/2020:11:17:16 +0000] "GET / HTTP/1.1" 304 -	180e8791-ea5c-462c-bde8-bd0869a7b544
2020-04-08 13:17:16	172.31.31.145 -- [08/Apr/2020:11:17:16 +0000] "GET / HTTP/1.1" 304 -	53d94d25-bb96-4f78-b93b-dc15c6ba15a6
2020-04-08 13:17:16	172.31.31.145 -- [08/Apr/2020:11:17:16 +0000] "GET / HTTP/1.1" 304 -	035d0ba3-ddf1-42ac-9404-34b1510d4281
2020-04-08 13:17:15	172.31.31.145 -- [08/Apr/2020:11:17:15 +0000] "GET / HTTP/1.1" 304 -	180e8791-ea5c-462c-bde8-bd0869a7b544
2020-04-08 13:17:15	172.31.31.145 -- [08/Apr/2020:11:17:15 +0000] "GET / HTTP/1.1" 304 -	53d94d25-bb96-4f78-b93b-dc15c6ba15a6
2020-04-08 13:17:15	172.31.31.145 -- [08/Apr/2020:11:17:15 +0000] "GET / HTTP/1.1" 304 -	035d0ba3-ddf1-42ac-9404-34b1510d4281
2020-04-08 13:17:15	172.31.31.145 -- [08/Apr/2020:11:17:15 +0000] "GET / HTTP/1.1" 304 -	180e8791-ea5c-462c-bde8-bd0869a7b544
2020-04-08 13:17:14	172.31.31.145 -- [08/Apr/2020:11:17:14 +0000] "GET / HTTP/1.1" 304 -	53d94d25-bb96-4f78-b93b-dc15c6ba15a6
2020-04-08 13:17:14	172.31.31.145 -- [08/Apr/2020:11:17:14 +0000] "GET / HTTP/1.1" 304 -	035d0ba3-ddf1-42ac-9404-34b1510d4281
2020-04-08 13:17:14	172.31.31.145 -- [08/Apr/2020:11:17:14 +0000] "GET / HTTP/1.1" 304 -	180e8791-ea5c-462c-bde8-bd0869a7b544

## Clean Up

To tear down everything:

1. Delete the Service
  - a. Potentially directly remove the tasks
2. The Capacity Provider *should* take care of scaling in, but it will take very long, so better directly delete the capacity provider
  - a. Update the Cluster -> Remove Capacity Provider
  - b. Deactivate the Capacity Provider
3. Open the Auto Scaling Group
  - a. Delete the ASG



- b. Delete the Launch Configuration
- 4. Load Balancer:
  - a. Delete the Application Load Balancer
  - b. Delete the Target Groups
- 5. Terminate any remaining EC2 Instances from the Lab manually
- 6. Remove the Security Groups
- 7. Also delete the cluster

---

*Lab End*

---