LAB: EC2 ECS Cluster with Capacity Providers

You need:

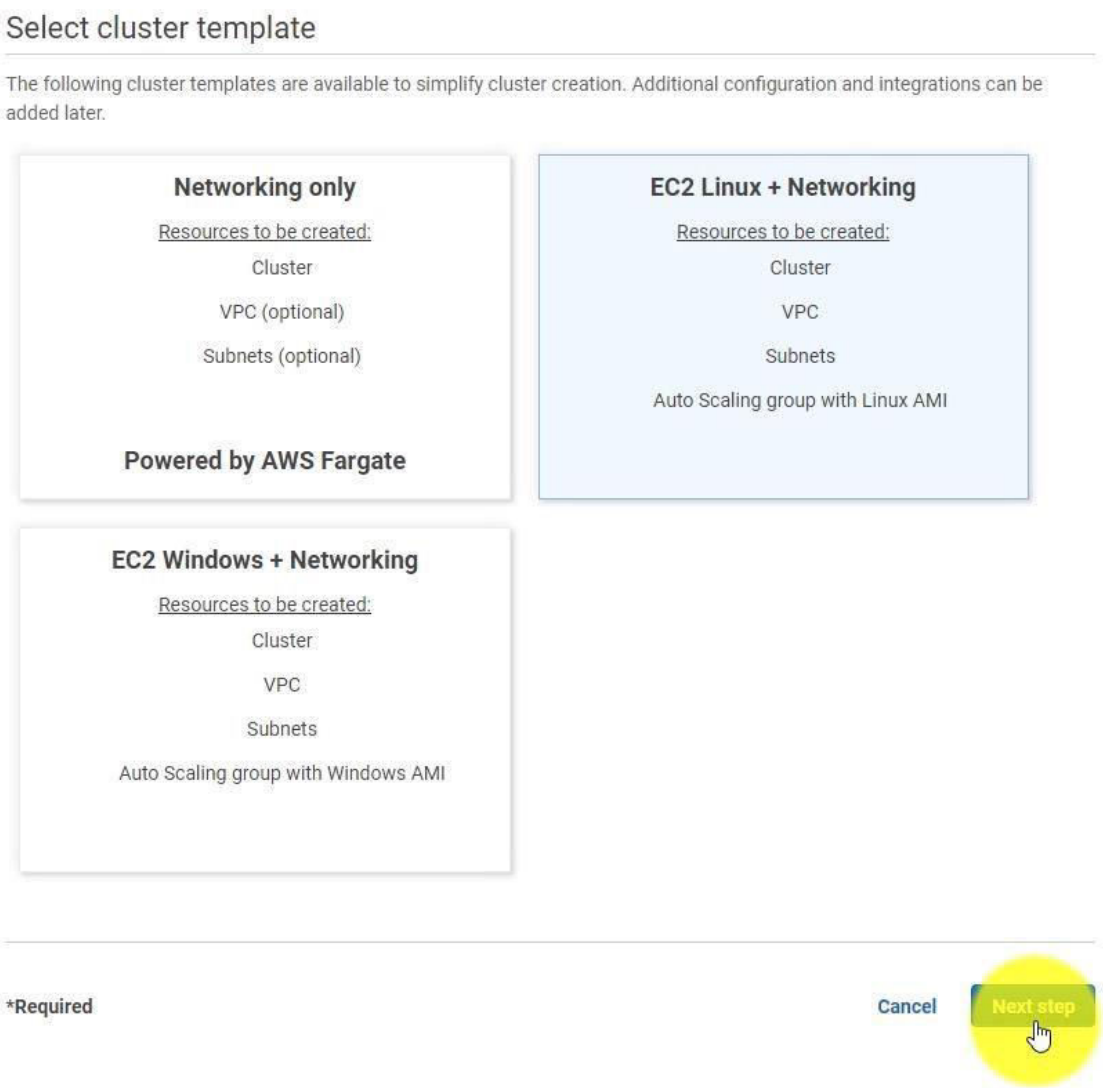
- An AWS Account

Duration of the Lab: 30 Minutes.

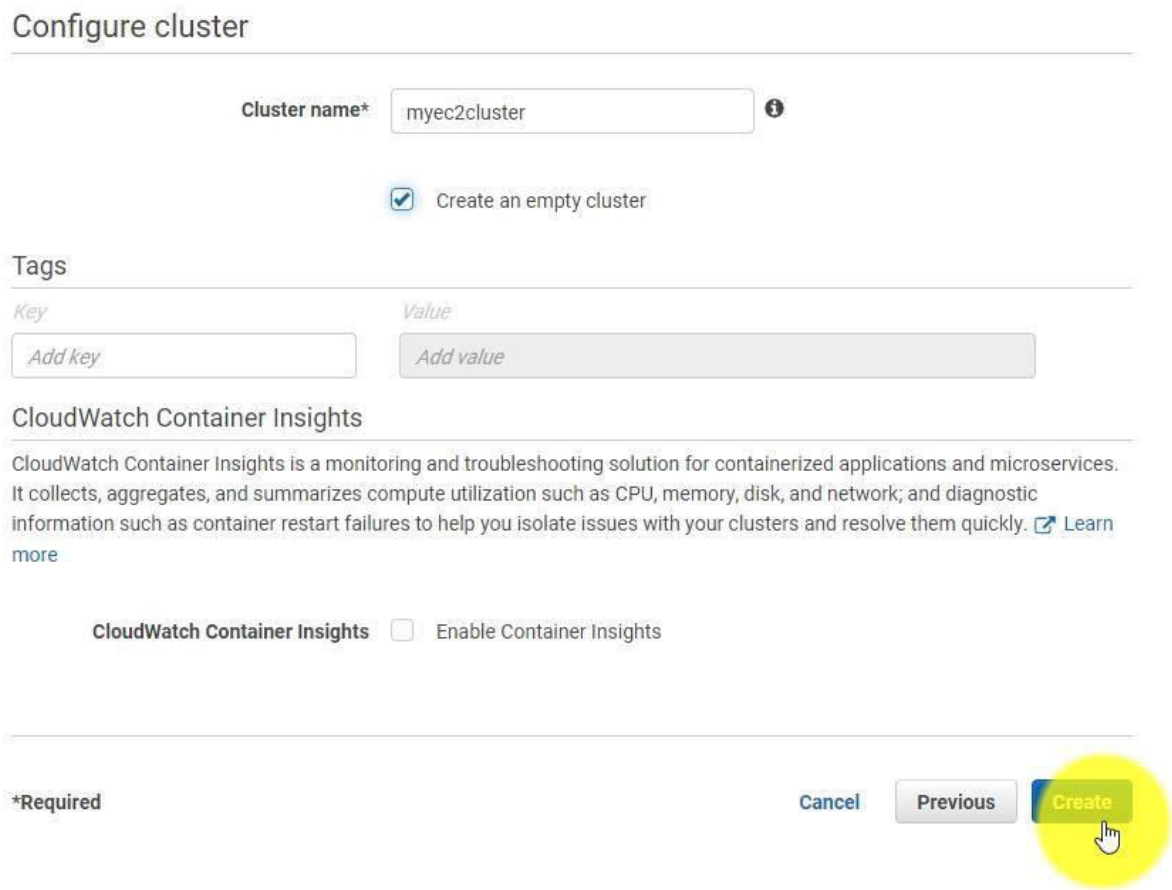
Difficulty: hard

Create an EC2 ECS Cluster

Create a new EC2 Launch Type Cluster:



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



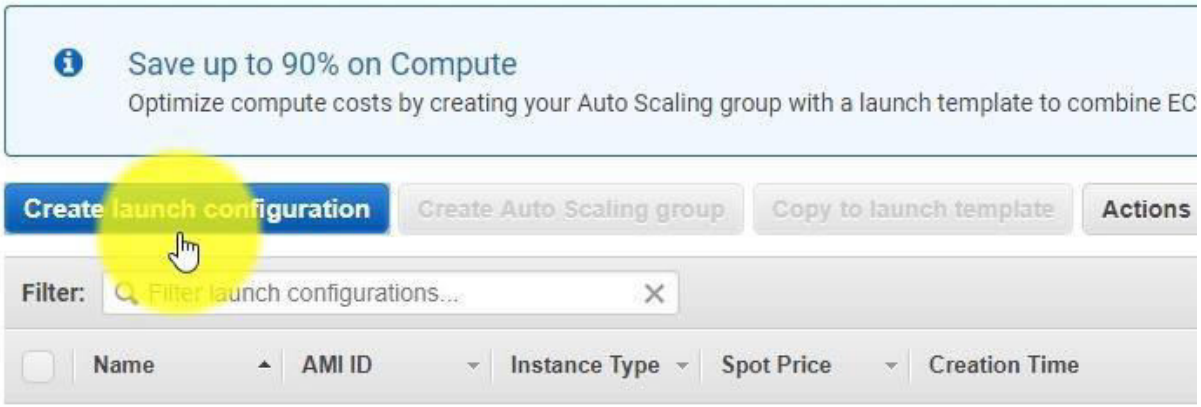
Create an Auto Scaling Group

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

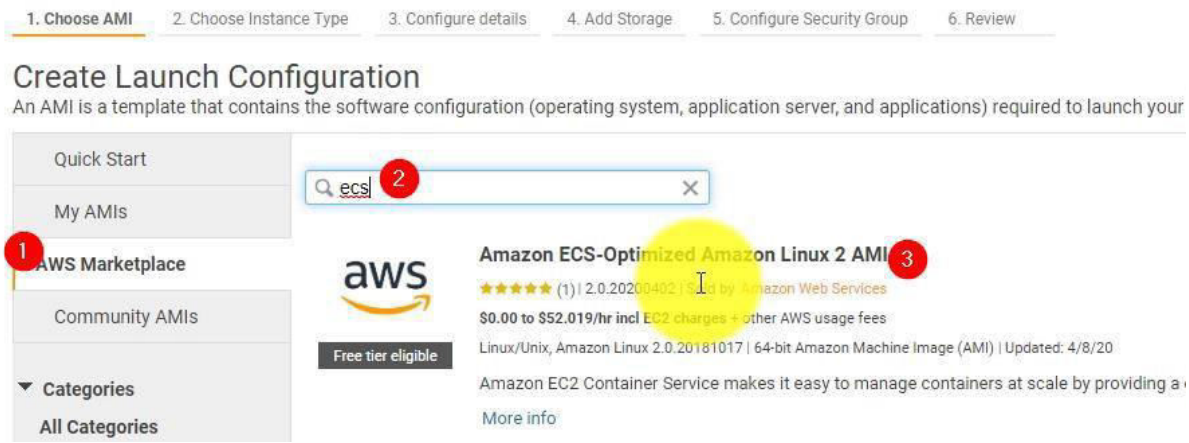
Create a new Launch Configuration

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

Create a new Launch Configuration:



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
```

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

Create Launch Configuration

Name: lc-ecs-ami

Purchasing option: Request Spot Instances

IAM role: ecsInstanceRole

Monitoring: Enable CloudWatch detailed monitoring
Learn more

Advanced Details

Kernel ID: Use default

RAM Disk ID: Use default

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	IyEvYmluL2JhczgKZWNoby8FQ1NfQ0xVU1RFUj1teWVjMmNsdXN0ZXIgPj4gL2V0Yy9lY3MvZWNzLmNvbmZpZw==
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 grou:

Launch configuration creation status

Successfully created launch configuration: lc-ecs-ami
View creation log

View
View your launch configurations
View your Auto Scaling groups

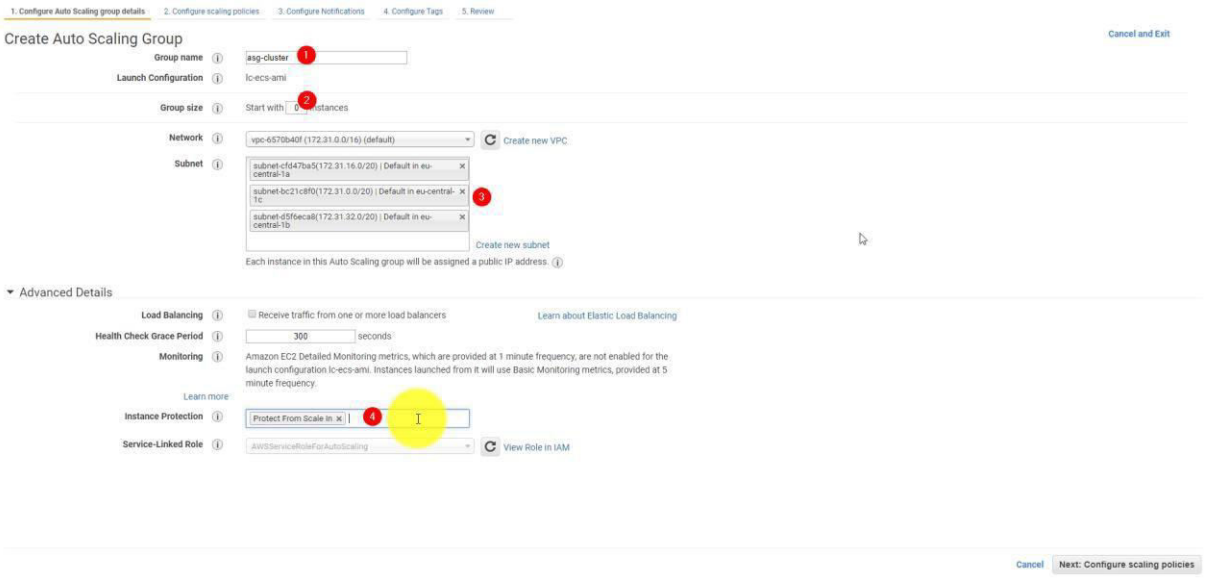
Here are some helpful resources to get you started

Create an Auto Scaling group using this launch configuration | Close

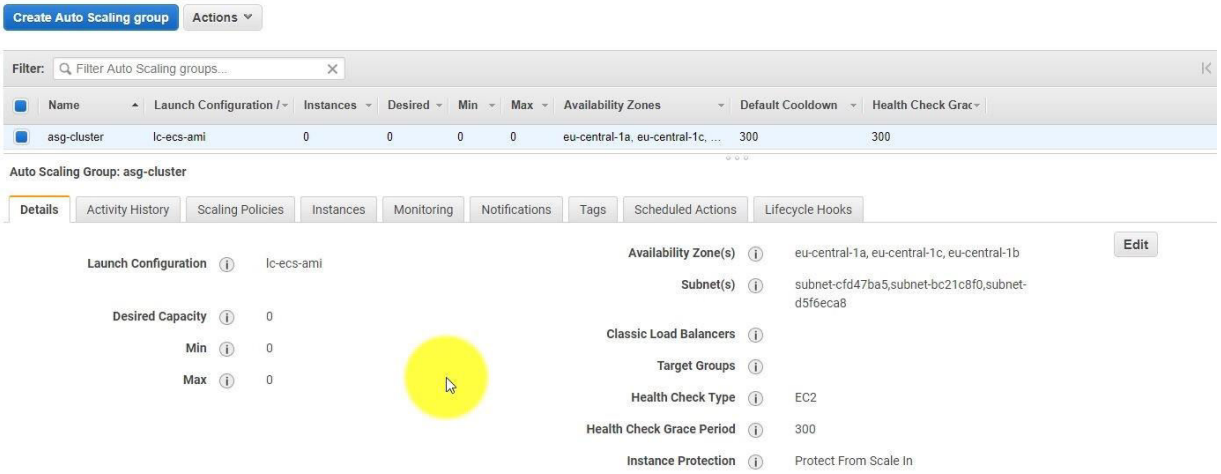
Create an Auto Scaling Group

Create the ASG:

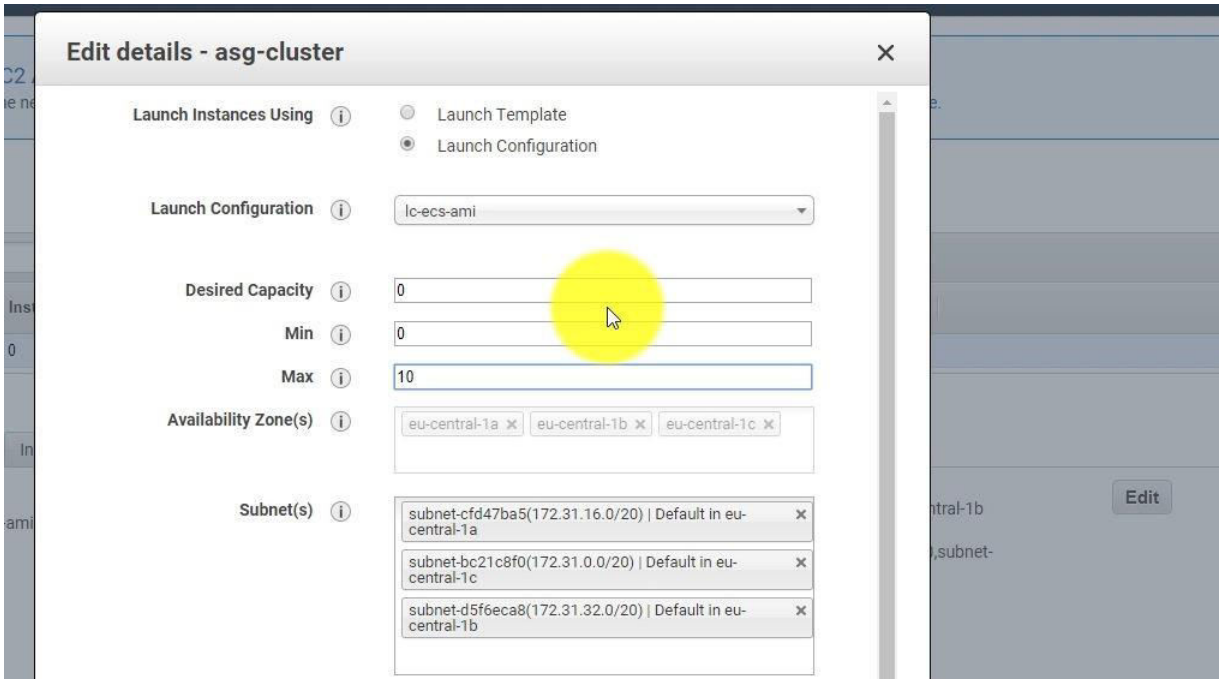
1. Give it a name
2. Start with 0 Instances
3. Select all available Subnets from your default VPC
4. Enable Instance Protection



Then proceed until the ASG is created.



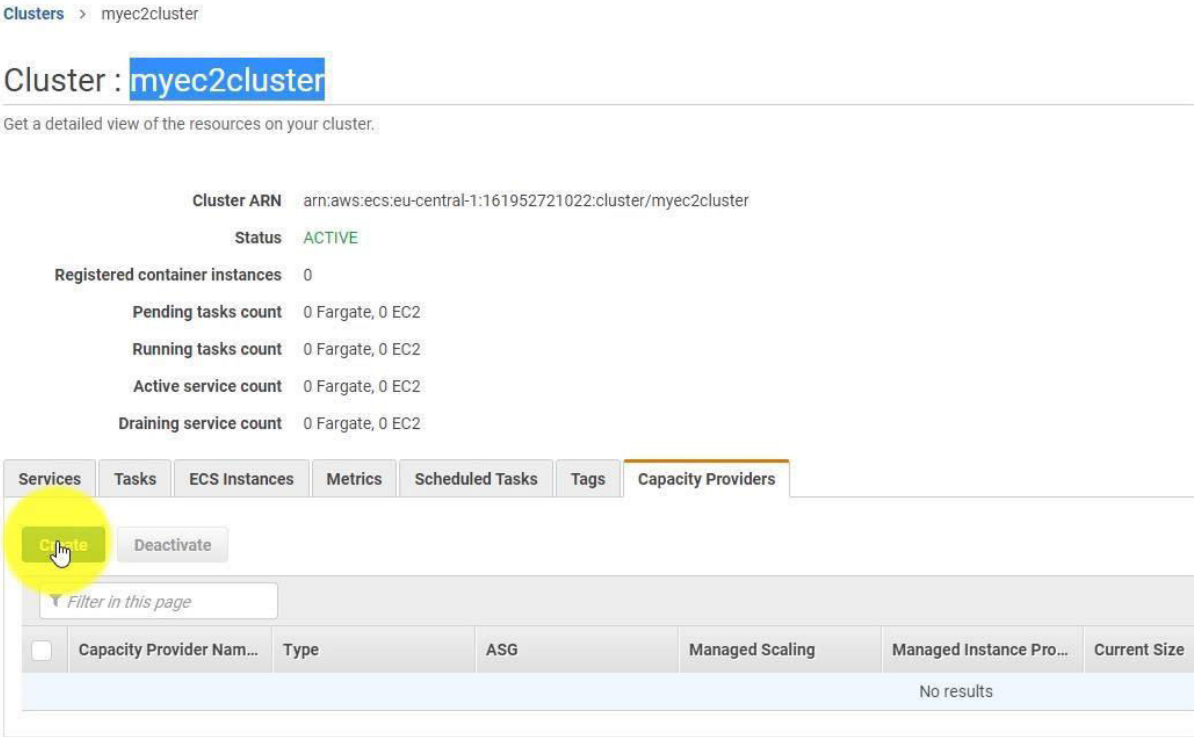
Now edit the ASG to set the maximum capacity to a higher number than 0:



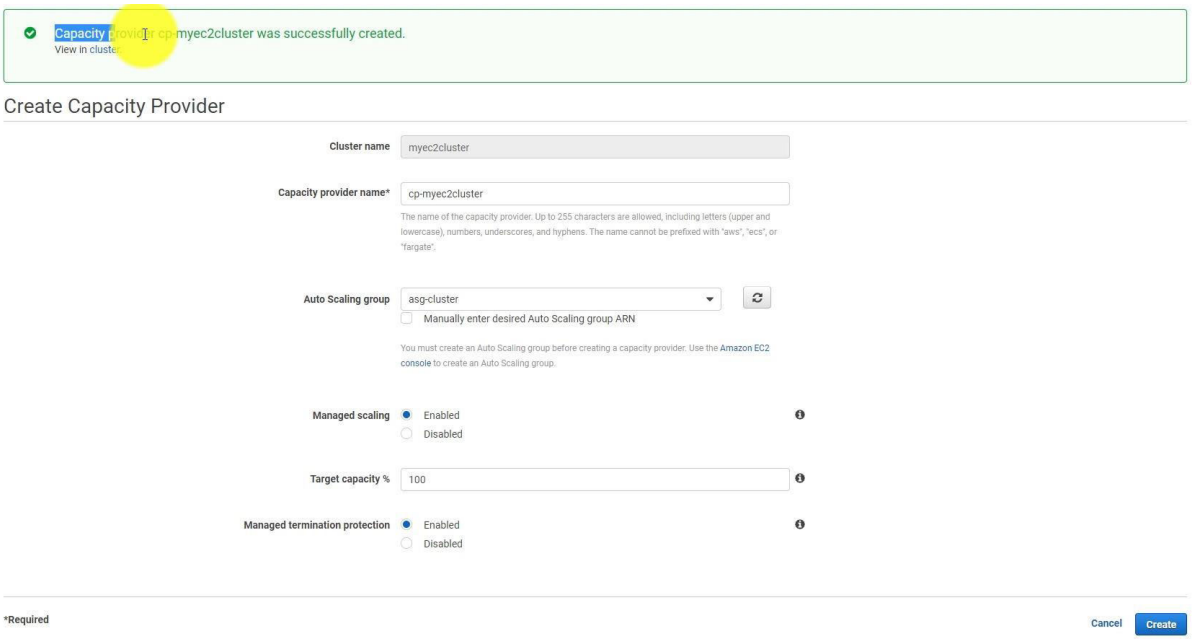
Now we need to attach a new Capacity Provider to the Auto Scaling Group.

Create a Capacity Provider

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

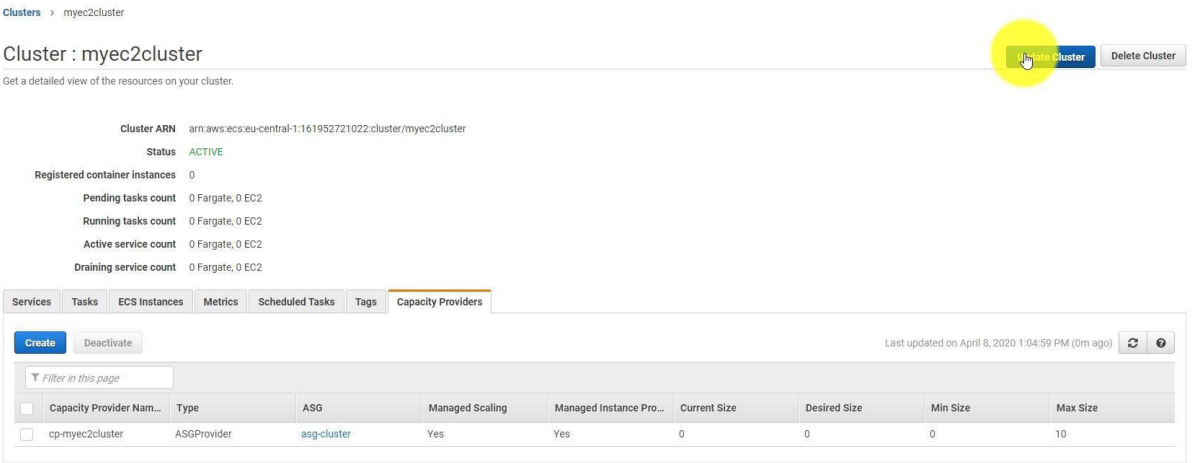


Give it a name, select the ASG you created in the previous step. Leave Managed Scaling to be enabled, set the target capacity to 100% and leave Managed termination protection also enabled:



Attach the Capacity Provider to the Cluster

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



Add the Capacity Provider we just created:

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:

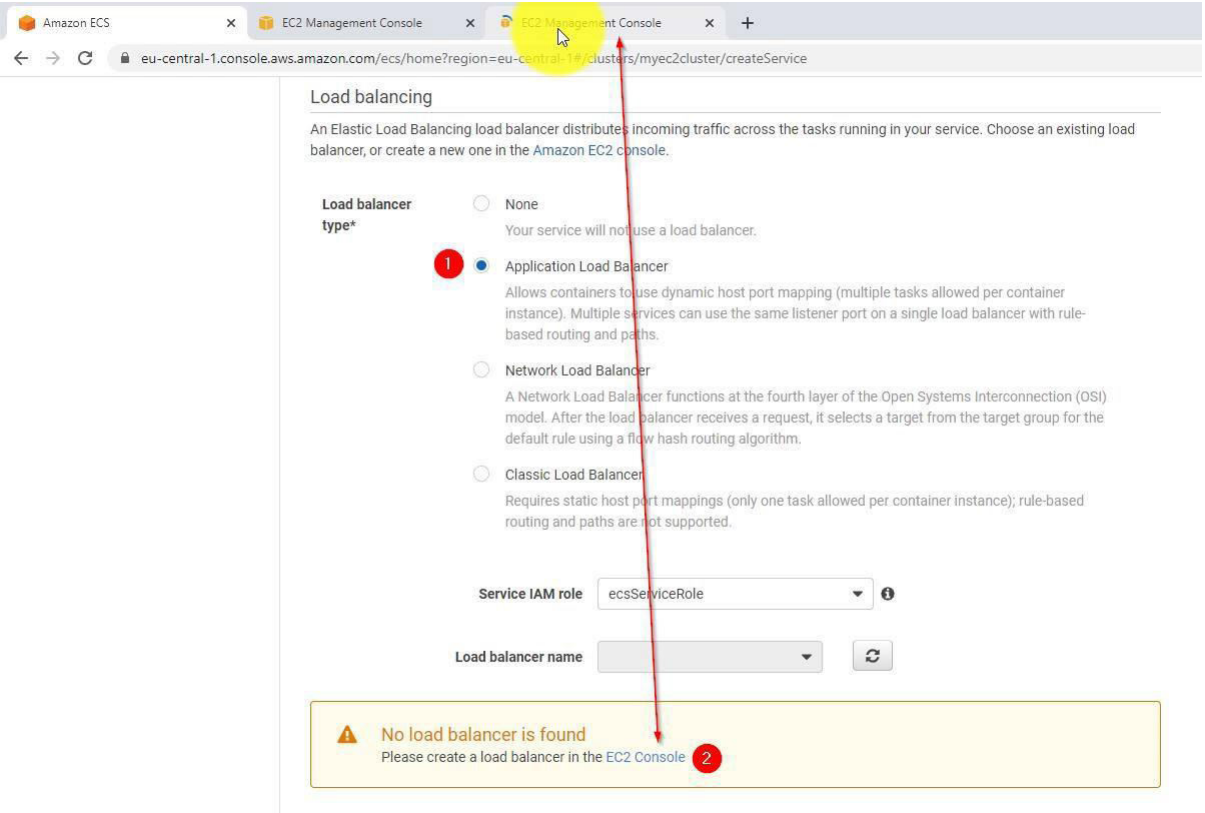
Select again the simple apache server task definition (1). Give it a service name (2). Launch 3 tasks initially (3). Leave the rest on default, then hit “Next step”.

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.

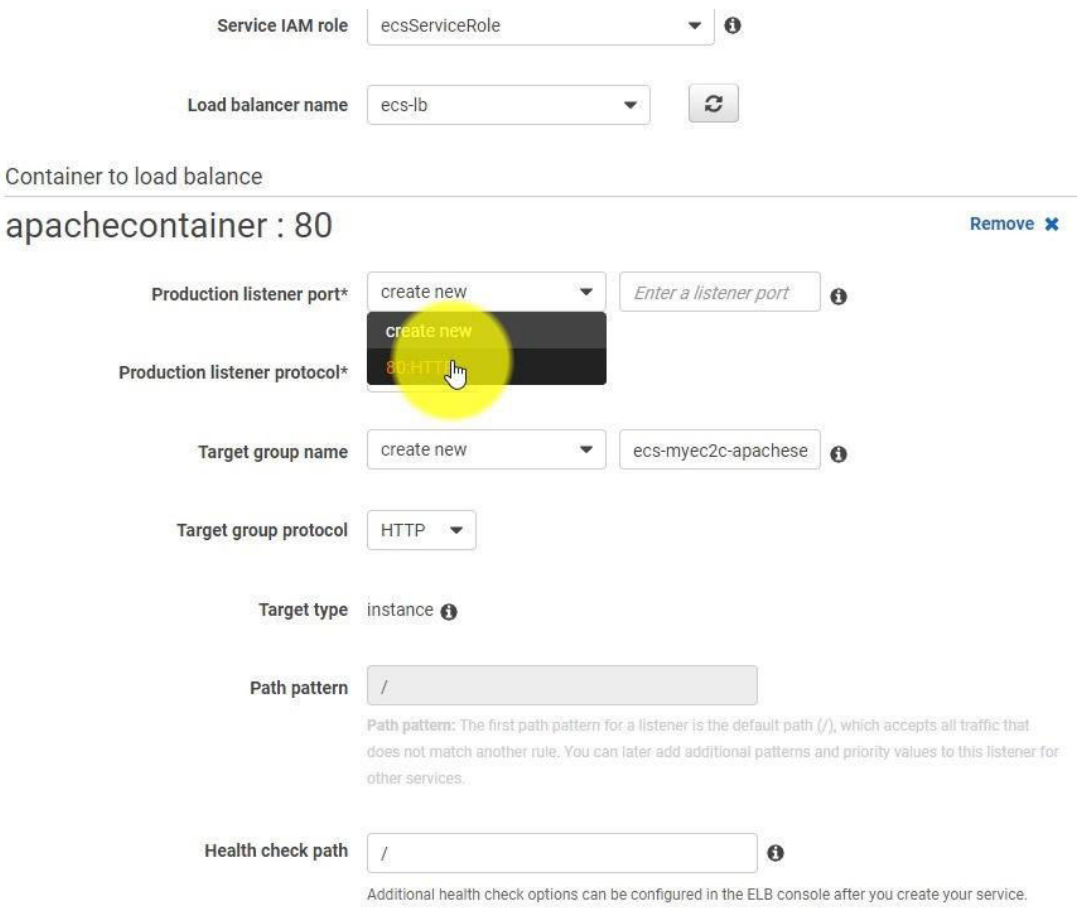
Capacity provider strategy	Cluster default strategy
Task Definition	Family: simple-apache-server (1) — Enter a value Revision: 2 (latest)
Cluster	myec2cluster
Service name	apacheservice (2)
Service type*	REPLICA / DAEMON
Number of tasks	(3)
Minimum healthy percent	100
Maximum percent	200

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:



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:



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%:

Minimum number of tasks: 3

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

Desired number of tasks: 3

Maximum number of tasks: 9

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

IAM role for Service Auto Scaling: ecsAutoscaleRole

Automatic task scaling policies

Scaling policy type: Target tracking / Step scaling

Policy name*: cpuloadscaling

ECS service metric*: ECSServiceAverageCPUUtilization

Target value*: 50

Scale-out cooldown period: 300 seconds between scaling actions

Scale-in cooldown period: 300 seconds between scaling actions

Disable scale-in

*Required

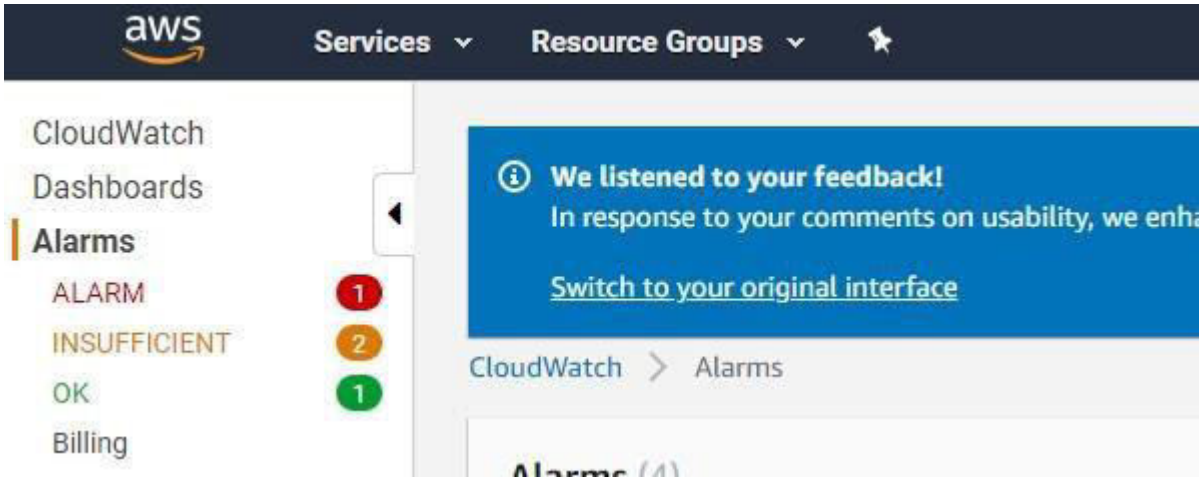
Cancel | Previous | Next step

Then create the service.

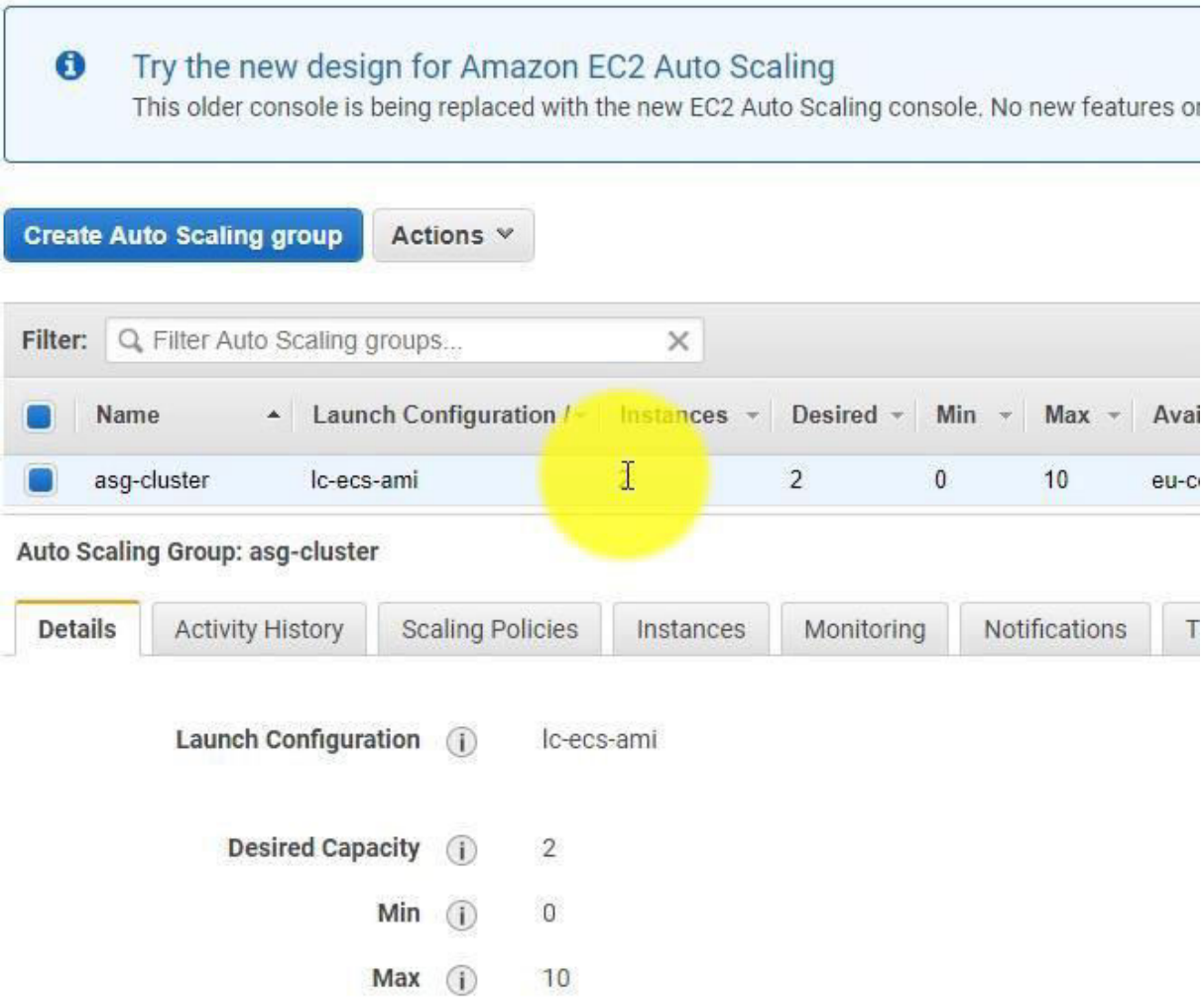
Watch the Scaling

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

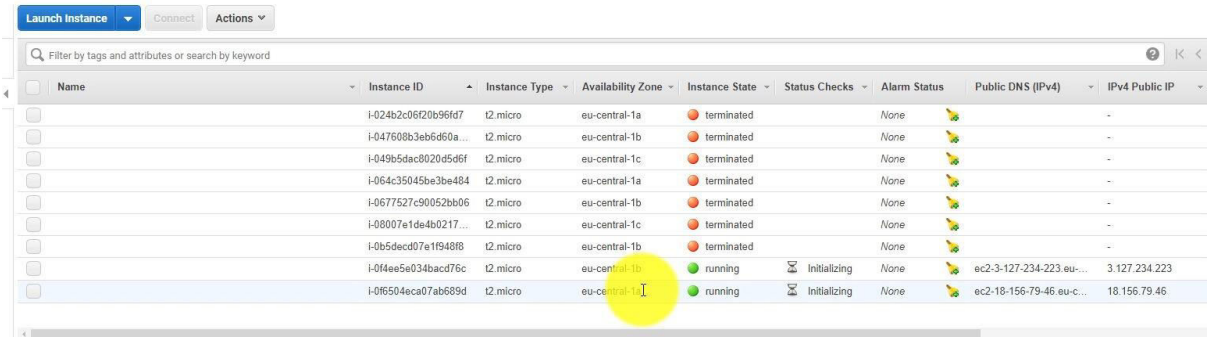
Open the CloudWatch Dashboard and have a look on the Alarms on the left side:



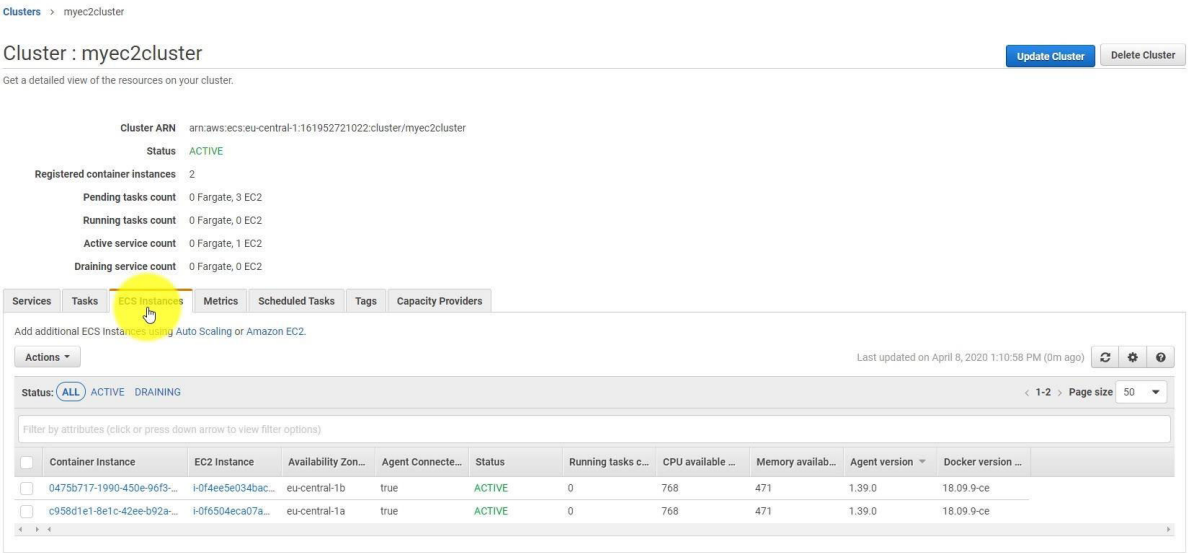
Have also a look at the ASG in the EC2 Dashboard. It will try to modify the desired capacity so that EC2 Instances are started by the ASG:



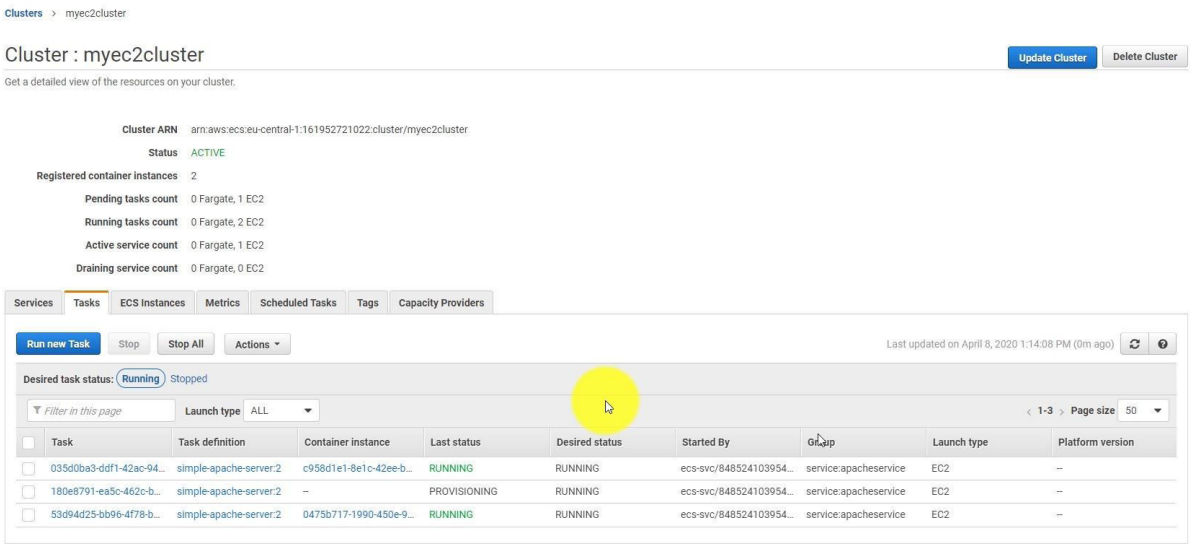
Have a look at the EC2 Dashboard, it automatically started the EC2 Instances through the ASG:



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:

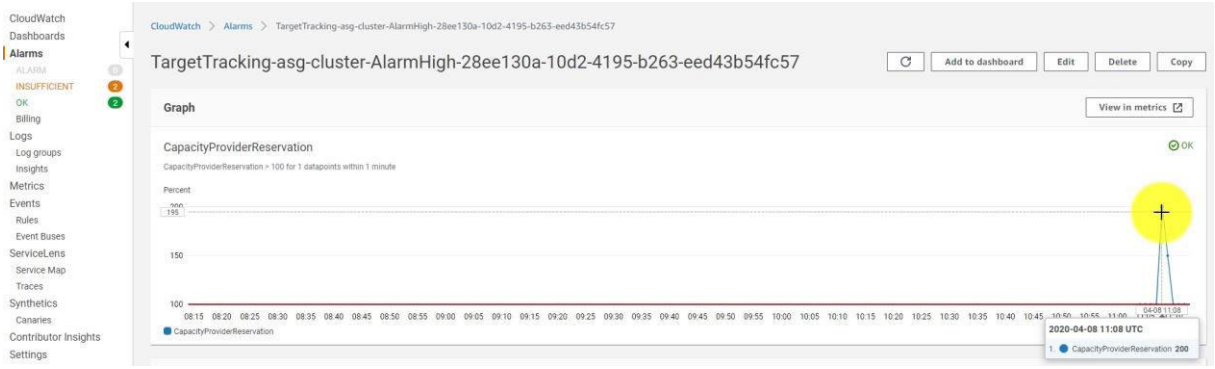


If you check the service, then it should have started the Containers successfully:



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

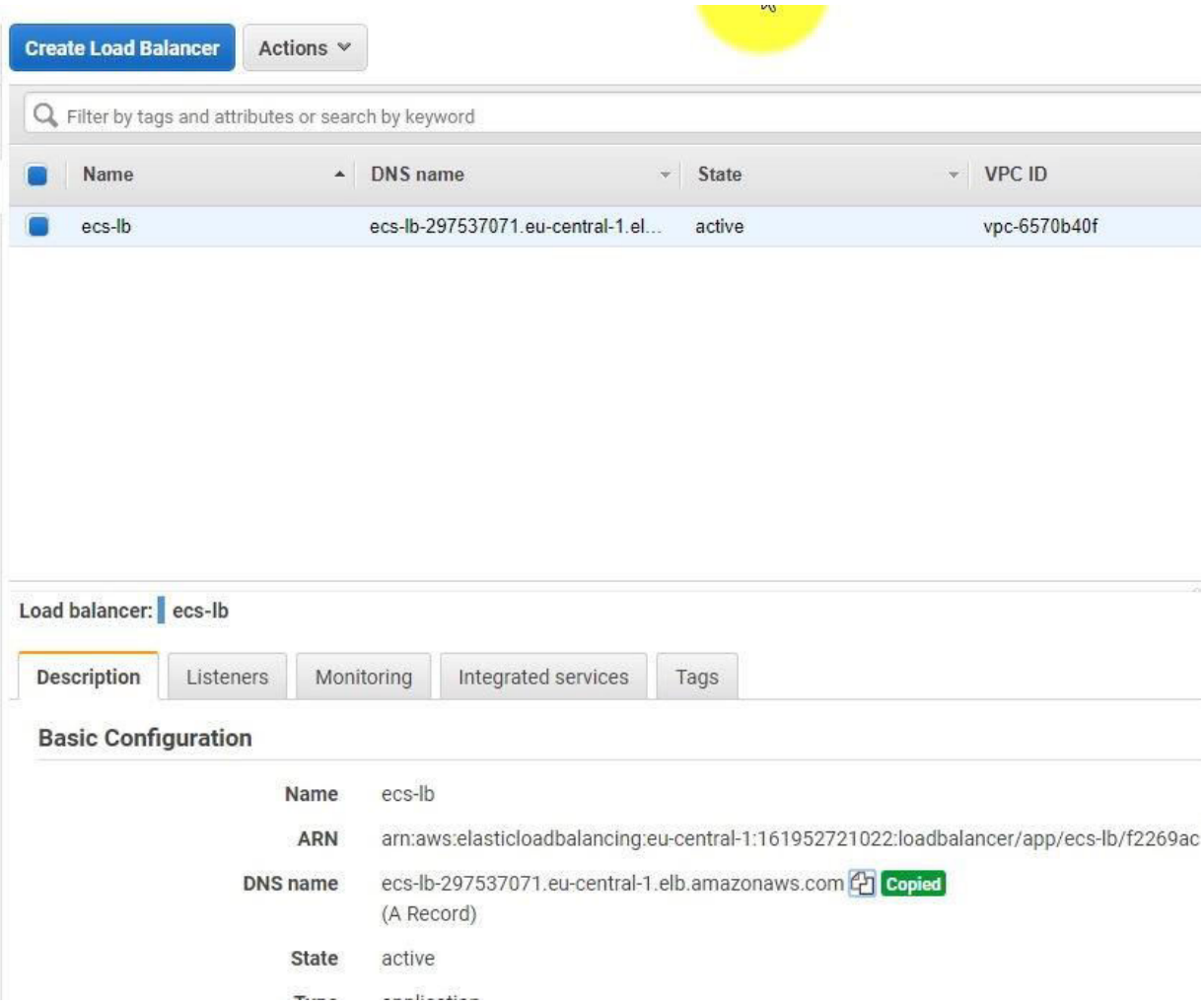
Check again the CloudWatch Alarms – they should be OK now. And if you drill down into the "AlarmHigh" you should see a spike where it went up to 200, then the ASG started the EC2 Instances and the Undercapacity was successfully mitigated by starting more compute resources, bringing the target capacity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

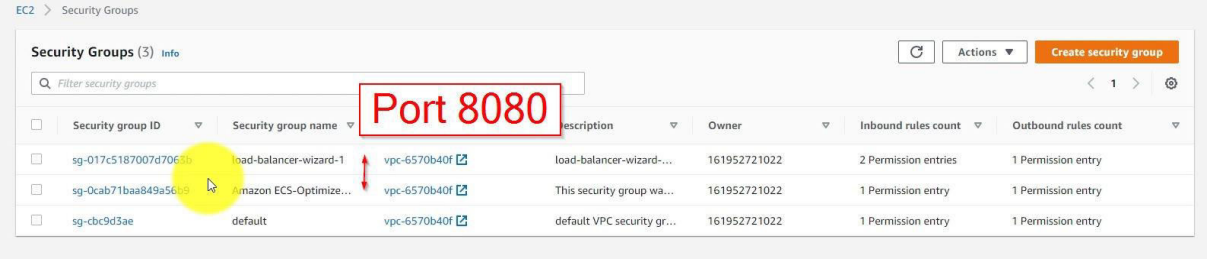
Open the EC2 Dashboard -> Load Balancers and copy the DNS of your ALB:

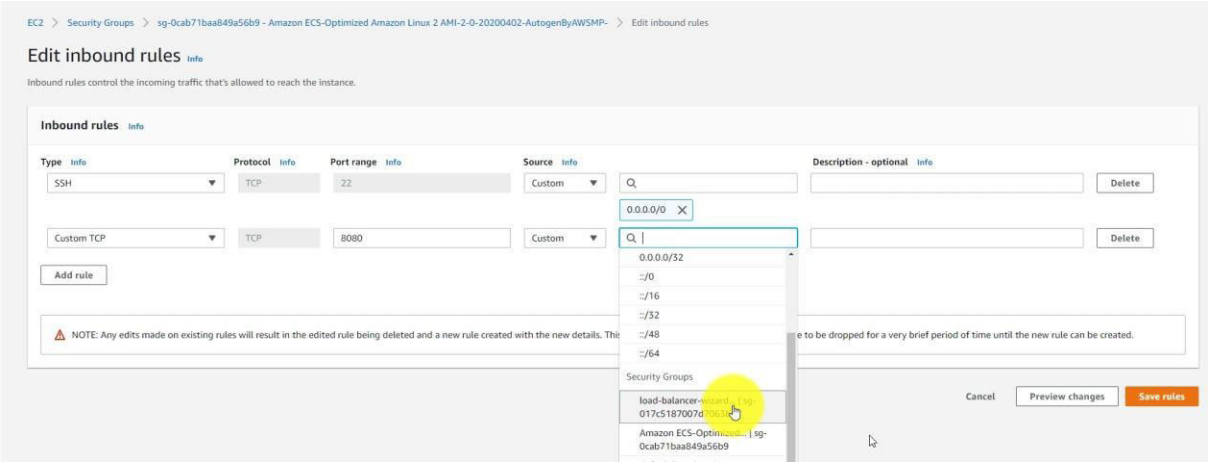


Then open a new tab and try to open this.

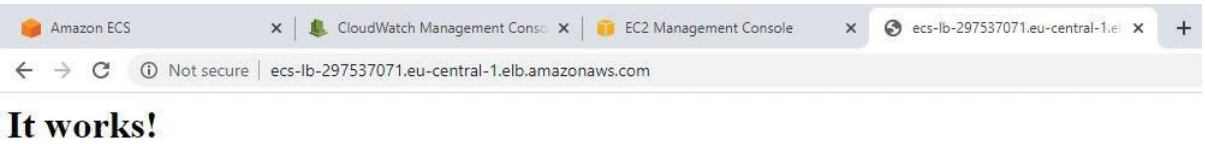
It times out, why? … security groups!

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:



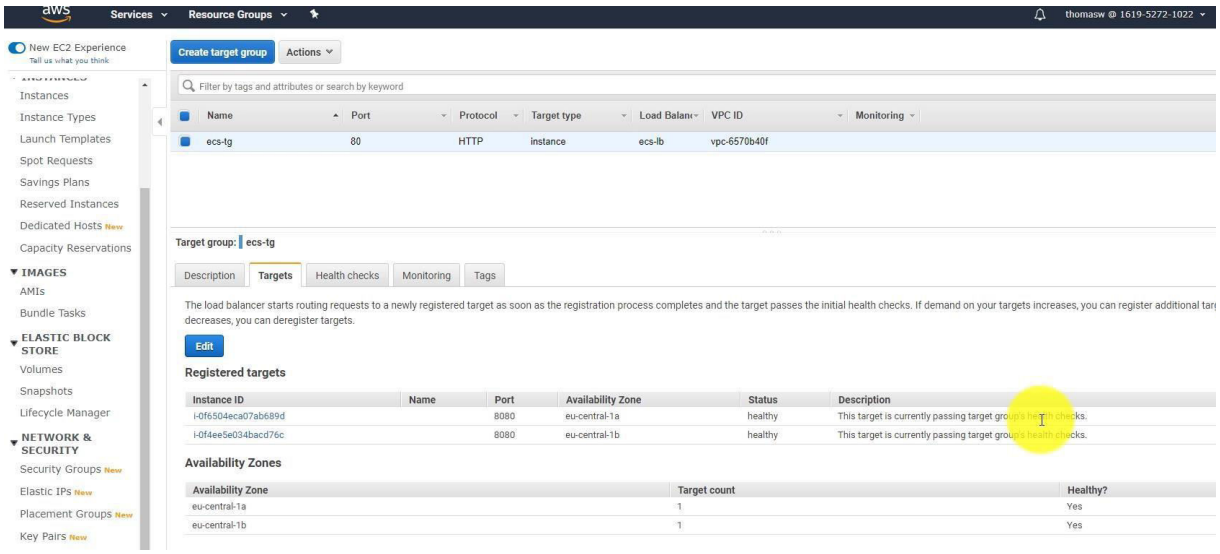


Try again to reload the browser tab, and it should work:

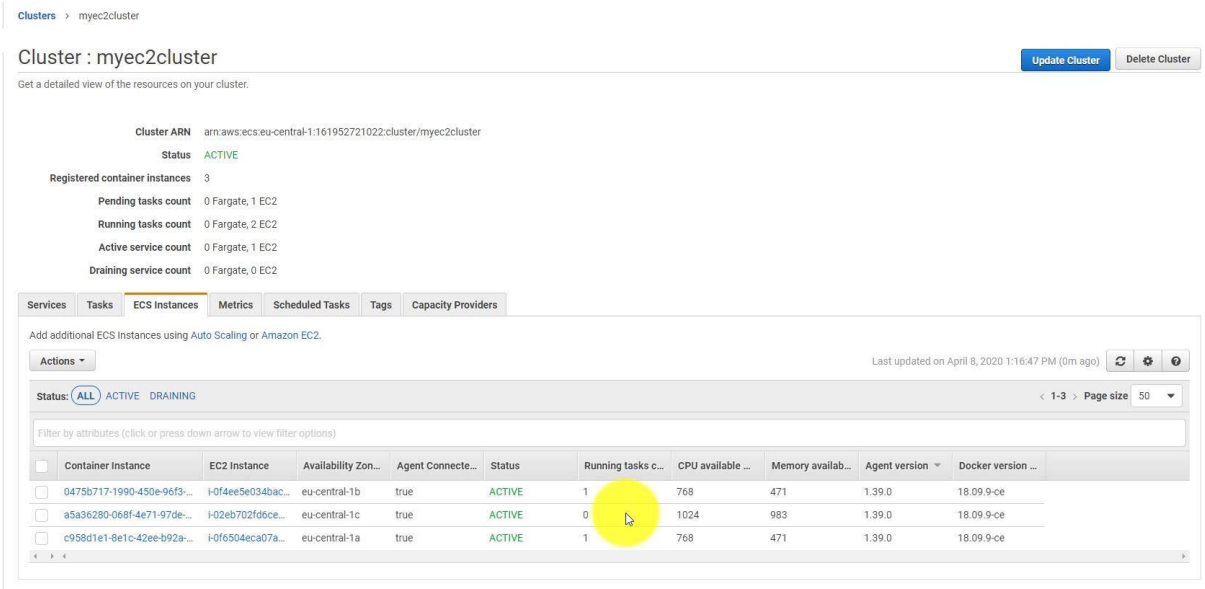


It works!

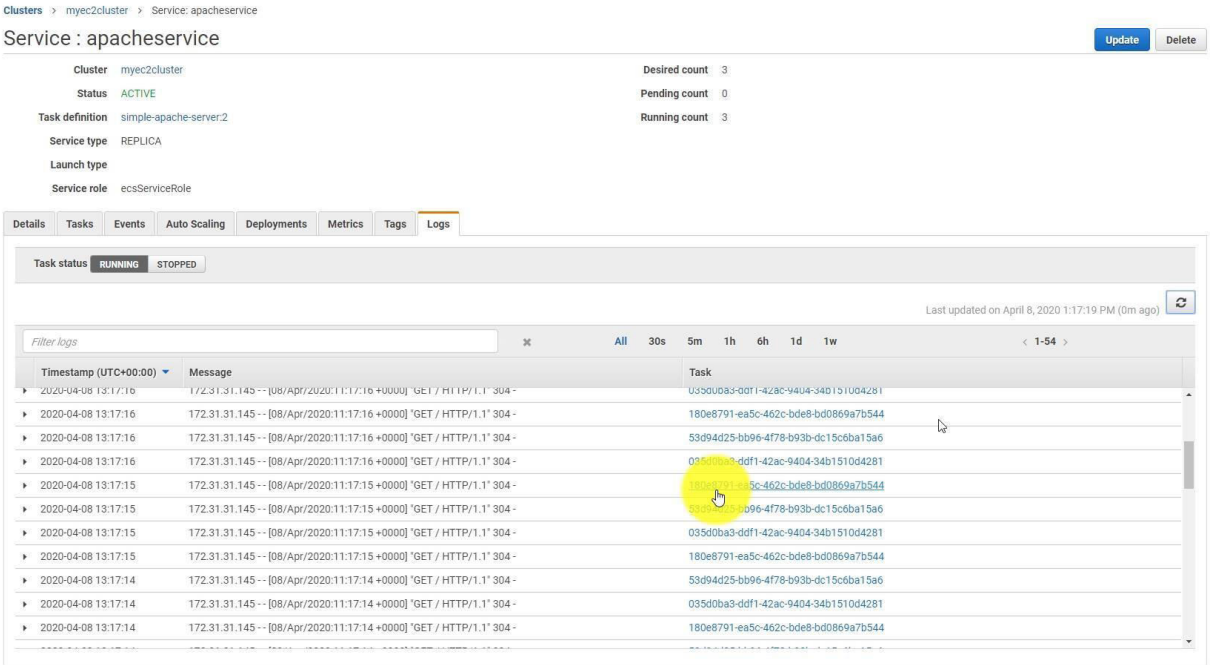
Also check out the Target Group of the Load Balancer and make sure the targets become healthy (might take a few minutes after updating the security groups):



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.



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:



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
