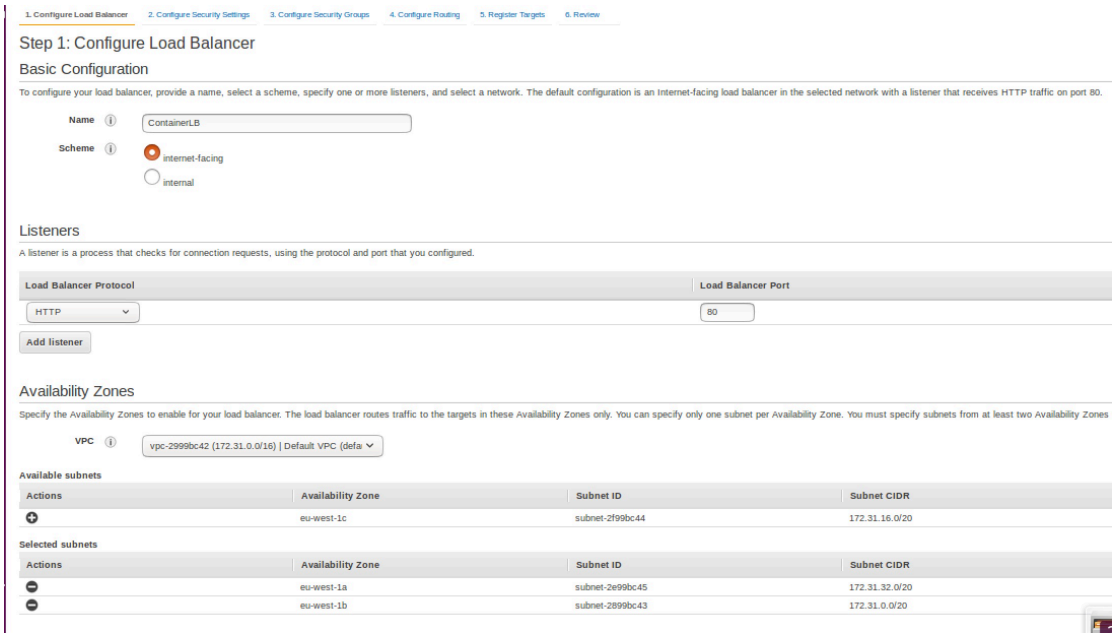
Scaling AWS EC2 Containers

This exercise assumes you have completed the previous lab exercises in which you dockerised an app and used the AWS Container service to pull the app from Dockerhub and ran the app in a container. In this exercise you will trigger the creation of multiple container instances based on a Cloudwatch alarm.

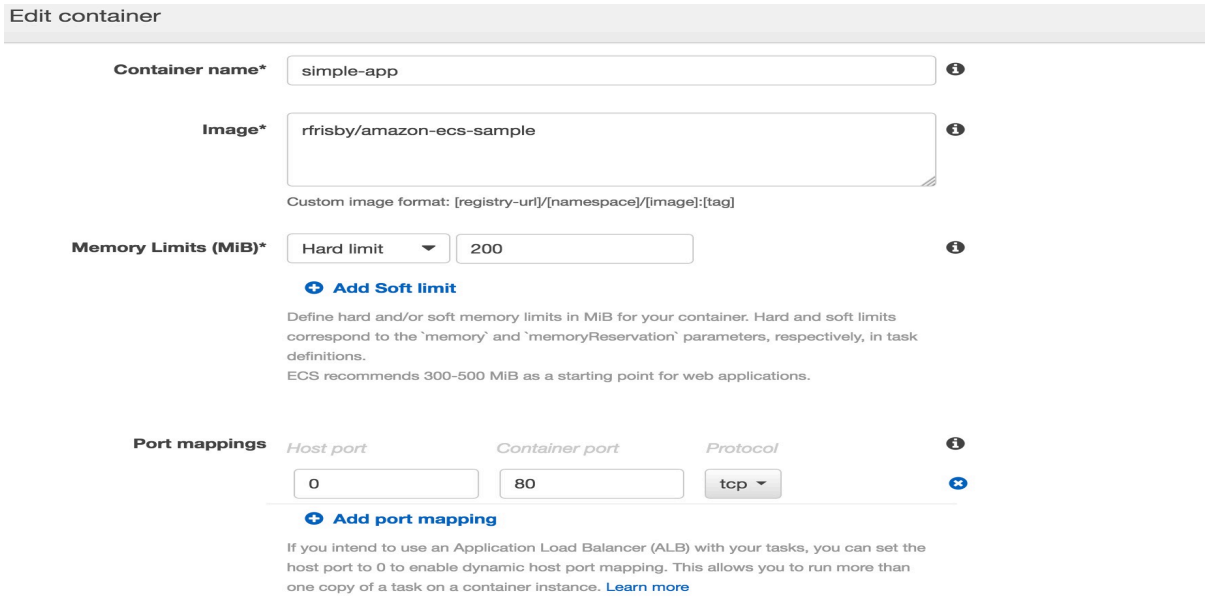
You will need to create an Elastic Load Balancer from the EC2 service page. Choose the Application Load Balancer and configure it



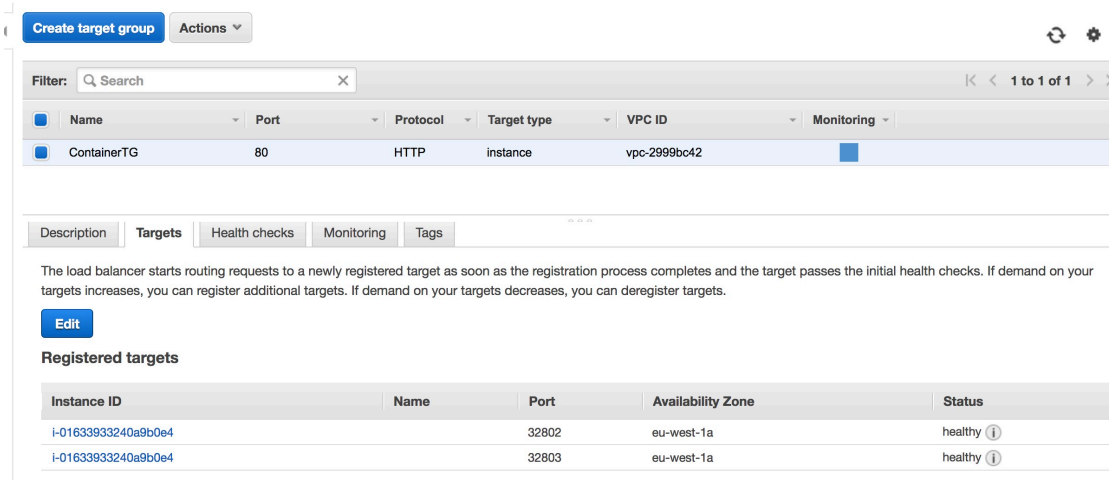
You will need to configure a new Target group (e.g. ContainerTG) and register your Cluster Instance with this group. If you are using the sample PHP application you can change the Health check Path to /index.php.

Once your Application Load Balancer has been created you will need to create a Service on your ECS Cluster and configure Load Balancing and Auto Scaling

Note to overcome the issue of running multiple containers on the same instance you will need to create a task definition that specifies Host Port 0 and container port 80. This will dynamically assign any available port when it runs the docker container and register this with the Target Group.



You should see the that the containers have been registered as targets on the dynamically generated ports.

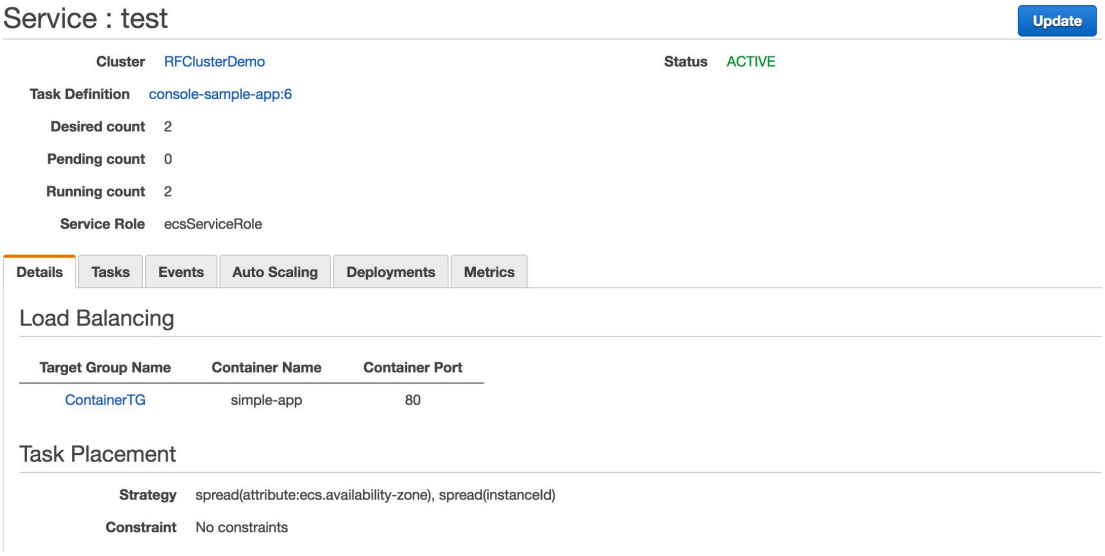


You can check the same on your server using the ‘docker ps’ command. Two different containers are running on two random ports by the same docker image that you mentioned in your task definition.

```
[ec2-user@ip-172-31-44-147 ~]$ docker ps
CONTAINER ID        IMAGE                            COMMAND                  CREATED             STATUS              PORTS                     NAMES
539d0982a23c        busybox                          "sh -c '/bin/sh -c..."   About an hour ago   Up About an hour                              ecs-console-sample-app-6-busybox-9acb93e38bf7
0dc1df01
63dfa6104c55        busybox                          "sh -c '/bin/sh -c..."   About an hour ago   Up About an hour                              ecs-console-sample-app-6-busybox-98a482f0c1df
b4bd5000
6f939827cf10        rfrisby/amazon-ecs-sample        "/usr/sbin/apache2..."   About an hour ago   Up About an hour    0.0.0.0:32803->80/tcp     ecs-console-sample-app-6-simple-app-e2eeba92a
ce5dda98d01
0f195313ef24        rfrisby/amazon-ecs-sample        "/usr/sbin/apache2..."   About an hour ago   Up About an hour    0.0.0.0:32802->80/tcp     ecs-console-sample-app-6-simple-app-b6b4faa7d
cd685a75e00
584b9d82e1cb        amazon/amazon-ecs-agent:latest   "/agent"                 4 hours ago         Up 4 hours                                    ecs-agent
[ec2-user@ip-172-31-44-147 ~]$
```

So If you are running multiple containers of a single service, you don’t need multiple servers for them. ALB allows to maximize the usage of servers and offers you a high-performance load balancing option. It gives you the flexibility of running multiple containers of a service on a single server by using the random available port.

Create a new service from the task and in ‘Configure ELB’ section, select application load balancer in ELB type section, select your ALB & target group and create service. If your ECS cluster has single ECS instance and tasks count is two, it will start two new containers in your instance on two different dynamic ports.



Service Auto Scaling instructions taken from -
http://docs.aws.amazon.com/AmazonECS/latest/developerguide/service_autoscaling_tutorial.html

When configuring the Auto Scaling Service:

For Minimum number of tasks, enter 2 for the lower limit of the number of tasks for Service Auto Scaling to use. Your service's desired count will not be automatically adjusted below this amount.

For Desired number of tasks, this field is pre-populated with the value you entered earlier. This value must be between the minimum and maximum number of tasks specified on this page. Leave this value at 2.

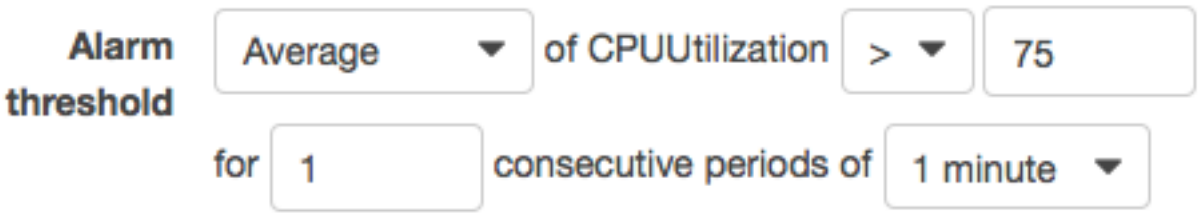
For Maximum number of tasks, enter 3 for the upper limit of the number of tasks for Service Auto Scaling to use. Your service's desired count will not be automatically adjusted above this amount.

For IAM role for Service Auto Scaling, choose an IAM role to authorize the Application Auto Scaling service to adjust your service's desired count on your behalf. If you have not previously created such a role, choose Create new role and the role is created for you. For future reference, the role that is created for you is called `ecsAutoscaleRole`

To configure scaling policies for your service

These steps will help you create scaling policies and CloudWatch alarms that can be used to trigger scaling activities for your service. You can create a scale out alarm to increase the desired count of your service, and a scale in alarm to decrease the desired count of your service.

1. For Policy name, enter `ScaleOutPolicy`
2. For Execute policy when, choose Create new alarm.
    a. For Alarm name, enter `sample-webapp-cpu-gt-75`.
    b. For ECS service metric, choose CPUUtilization.
    c. For Alarm threshold, enter the following information to match the image below. This causes the CloudWatch alarm to trigger when the service's CPU utilization is greater than 75% for one minute.



    d. Choose Save to save your alarm.
3. For Scaling action, enter the following information to match the image below. This causes your service's desired count to increase by 1 task when the alarm is triggered.

4. For Cooldown period, enter 60 for the number of seconds between scaling actions and choose Save to save your `ScaleOutPolicy`.
5. For your scaling down enter Policy name, enter `ScaleInPolicy`
6. For Execute policy when, choose Create new alarm.
    a. For Alarm name, enter `sample-webapp-cpu-lt-25`.
    b. For ECS service metric, choose CPUUtilization.
    c. For Alarm threshold, enter the following information to match the image below. This causes the CloudWatch alarm to trigger when the service's CPU utilization is less than 25% for one minute.

    d. Choose Save to save your alarm.
7. For Scaling action, enter the following information to match the image below. This causes your service's desired count to decrease by 1 task when the alarm is triggered.

8. For Cooldown period, enter 60 for the number of seconds between scaling actions and choose Save to save your `ScaleOutPolicy`.

Trigger a Scaling Activity

After your service is configured with Service Auto Scaling, you can trigger a scaling activity by pushing your service's CPU utilization into the `ALARM` state. Because this example is a web application that is running behind a load balancer, you can send thousands of HTTP requests to your service (using the ApacheBench utility) to spike the service CPU utilization above our threshold amount. This spike should trigger the alarm, which in turn triggers a scaling activity to add one task to your service.

After the ApacheBench utility finishes the requests, the service CPU utilization should drop below your 25% threshold, triggering a scale in activity that returns the service's desired count to 1

To trigger a scaling activity for your service

1. From your service's main view page in the console, choose the load balancer name to view its details in the Amazon EC2 console. You need the load balancer's DNS name, which should look something like this: `EC2Contai-EcsElast-SMAKV74U23PH-96652279.us-east-1.elb.amazonaws.com`.
2. Use the ApacheBench (**ab**) utility to make thousands of HTTP requests to your load balancer in a short period of time. To install ab on Ubuntu :

   ```
   $ sudo apt-get install apache2-utils
   ```

   Run the following command, substituting your load balancer's DNS name.

   ```
   $ ab -n 100000 -c 1000 http://EC2Contai-EcsElast-SMAKV74U23PH-96652279.us-east-1.elb.amazonaws.com/
   ```

3. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
4. Choose Alarms in the left navigation pane.
5. Wait for your **ab** HTTP requests to trigger the scale out alarm in the CloudWatch console. You should see your Amazon ECS service scale out and add 1 task to your service's desired count.
6. Shortly after your **ab** HTTP requests complete (between 1 and 2 minutes), your scale in alarm should trigger and the scale in policy reduces your service's desired count back to 1.

NOTE : You should look at the Service Events which will tell you what is happening. For example below the Cloudwatch alaram was triggered requesting ECS to laucn another task on the Cluster however the t2.micro instance did not have enough memory available to allow the container/task to run.

Event Id	Event Time	Message
e2df845d-2c52-4773-b083-9b2bca21a465	2017-11-12 15:59:44 +0000	service test was unable to place a task because no container instance met all of its requirements. The closest matching container-instance 85a9ee1f-c8f9-4c0a-9969-b9021ed3aa9b has insufficient memory available. For more information, see the Troubleshooting section.
2f9d7c68-cd49-49fd-8c41-6827f9a224be	2017-11-12 15:59:08 +0000	Message: Successfully set desired count to 3. Waiting for change to be fulfilled by ecs. Cause: monitor alarm scaleupcontainer in state ALARM triggered policy ScaleUpTestContainer
c64bb534-c0cb-4c86-901c-e964cccadf3d	2017-11-12 15:18:14	service test has reached a steady state.

Cleaning Up

When you have completed this exercise, you may choose to keep your cluster, Auto Scaling group, load balancer, and EC2 instances. However, if you are not actively using these resources, you should consider cleaning them up so that your account does not incur unnecessary charges.