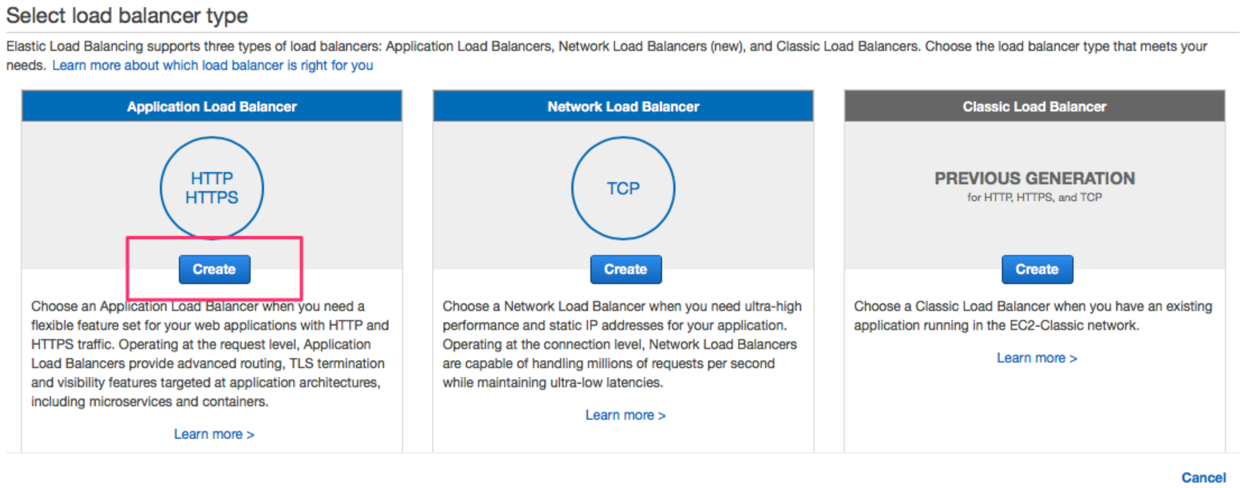
AWS Container Immersion Day: Lab 2

Lab 2 will build on Lab 1.

9. Creating the ALB

Now that we've pushed our images, we need an Application Load Balancer ALB to route traffic to our endpoints. An ALB lets you direct traffic between different endpoints and in this lab, we'll use two separate endpoints: `/web` and `/api`.

To create the ALB, navigate to the EC2 Console, and select **Load Balancers** from the left-hand menu. Choose **Create Load Balancer**. Create an Application Load Balancer:



Select load balancer type

Elastic Load Balancing supports three types of load balancers: Application Load Balancers, Network Load Balancers (new), and Classic Load Balancers. Choose the load balancer type that meets your needs. Learn more about which load balancer is right for you

Application Load Balancer

HTTP HTTPS

Create

Choose an Application Load Balancer when you need a flexible feature set for your web applications with HTTP and HTTPS traffic. Operating at the request level, Application Load Balancers provide advanced routing, TLS termination and visibility features targeted at application architectures, including microservices and containers.

Learn more >

Network Load Balancer

TCP

Create

Choose a Network Load Balancer when you need ultra-high performance and static IP addresses for your application. Operating at the connection level, Network Load Balancers are capable of handling millions of requests per second while maintaining ultra-low latencies.

Learn more >

Classic Load Balancer

PREVIOUS GENERATION for HTTP, HTTPS, and TCP

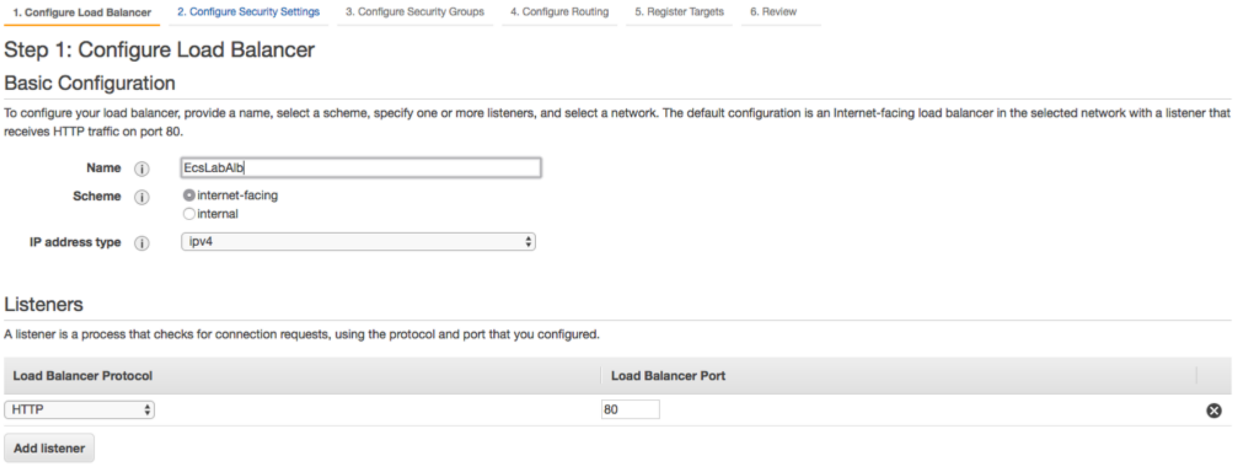
Create

Choose a Classic Load Balancer when you have an existing application running in the EC2-Classic network.

Learn more >

Cancel

Name your ALB **EcsLabAlb** and add an HTTP listener on port 80:



1. Configure Load Balancer
2. Configure Security Settings
3. Configure Security Groups
4. Configure Routing
5. Register Targets
6. Review

Step 1: Configure Load Balancer

Basic Configuration

To configure your load balancer, provide a name, select a scheme, specify one or more listeners, and select a network. The default configuration is an Internet-facing load balancer in the selected network with a listener that receives HTTP traffic on port 80.

Name: EcsLabAlb

Scheme: ● internet-facing ○ internal

IP address type: ipv4

Listeners

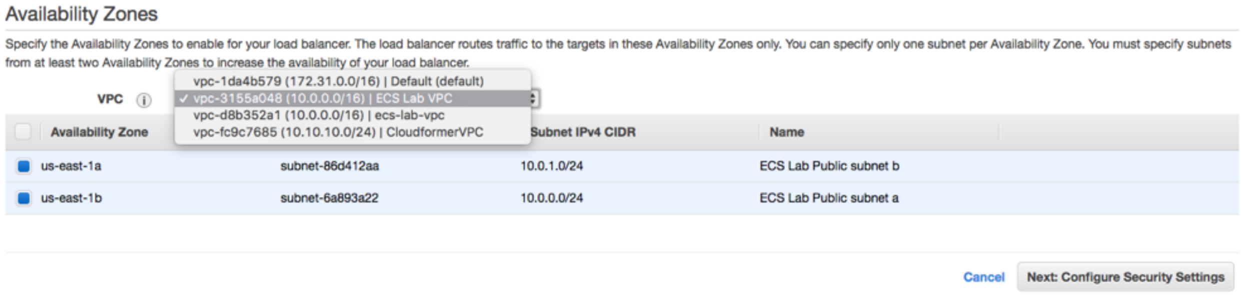
A listener is a process that checks for connection requests, using the protocol and port that you configured.

Load Balancer Protocol	Load Balancer Port
HTTP	80

Add listener

Note: in a production environment, you should also have a secure listener on port 443. This will require an SSL certificate, which can be obtained from AWS Certificate Manager, or from your registrar/CA. For the purposes of this lab, we will only create the insecure HTTP listener. DO NOT RUN THIS IN PRODUCTION.

Next, select your VPC and we need at least two subnets for high availability. Make sure to choose the VPC that was used in Lab 1.



Availability Zones

Specify the Availability Zones to enable for your load balancer. The load balancer routes traffic to the targets in these Availability Zones only. You can specify only one subnet per Availability Zone. You must specify subnets from at least two Availability Zones to increase the availability of your load balancer.

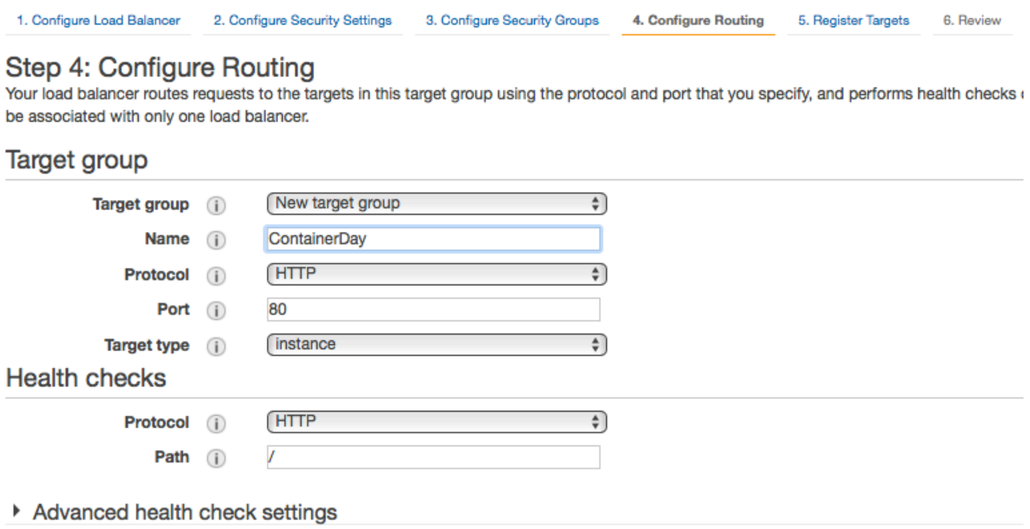
VPC
- vpc-1da4b579 (172.31.0.0/16) | Default (default)
- ✓ vpc-3155a048 (10.0.0.0/16) | ECS Lab VPC
- vpc-d8b352a1 (10.0.0.0/16) | ecs-lab-vpc
- vpc-fc9c7685 (10.10.10.0/24) | CloudformerVPC

Availability Zone		Subnet IPv4 CIDR	Name
■ us-east-1a	subnet-86d412aa	10.0.1.0/24	ECS Lab Public subnet b
■ us-east-1b	subnet-6a893a22	10.0.0.0/24	ECS Lab Public subnet a

Click **Next**, and create a new security group (`sgecslabloadbalancer`) with the following rule:

```
Ports          Protocol          Source
 80              tcp           0.0.0.0/0
```

Continue to the next step: **Configure Routing**. For this initial setup, we're just adding a dummy health check on `/`. We'll add specific health checks for our service endpoints when we register them with the ALB.


1. Configure Load Balancer
2. Configure Security Settings
3. Configure Security Groups
4. Configure Routing
5. Register Targets
6. Review

Step 4: Configure Routing

Your load balancer routes requests to the targets in this target group using the protocol and port that you specify, and performs health checks be associated with only one load balancer.

Target group

Target group	New target group
Name	ContainerDay
Protocol	HTTP
Port	80
Target type	instance

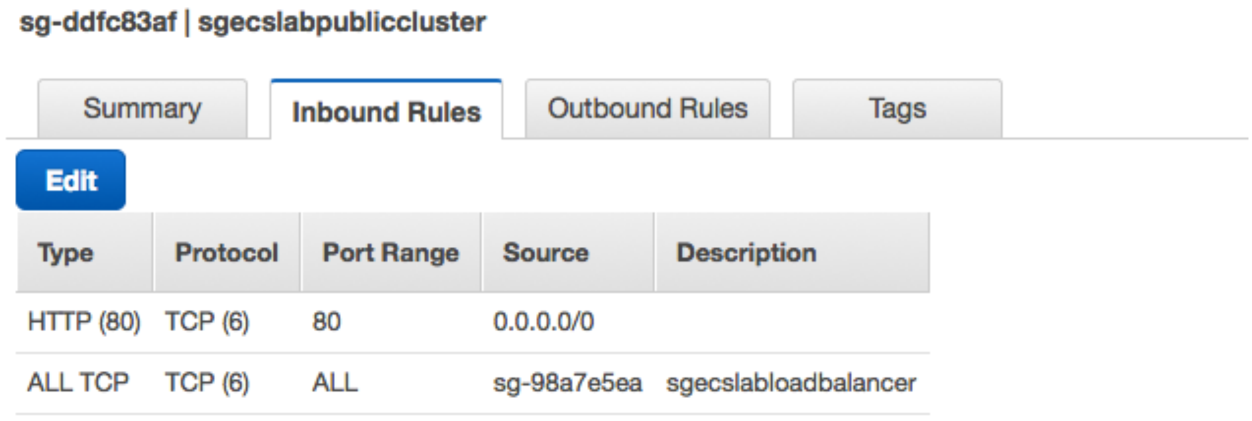
Health checks

Protocol	HTTP
Path	/

▸ Advanced health check settings

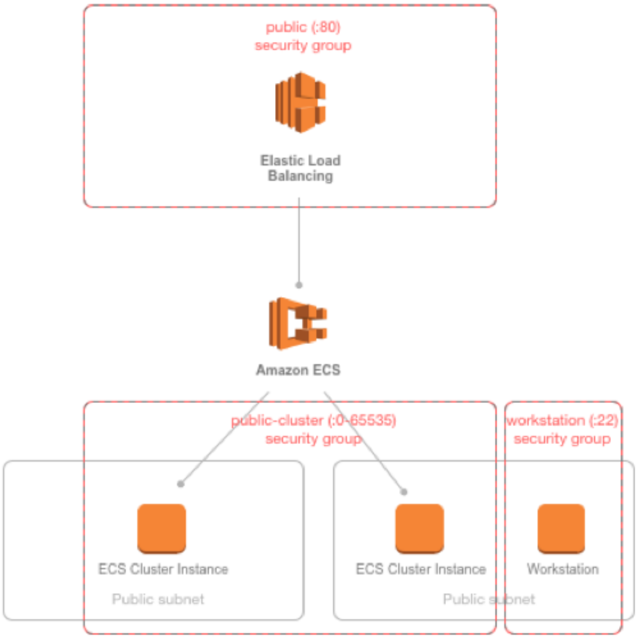
Click through the "Next:Register targets" step, and continue to the **Review** step. If your values look correct, click **Create**.

Note: If you created your own security group for the ECS Cluster (`sgecslabpubliccluster`), and only added a rule for port 80, you'll need to add one more. Edit your security group and add a rule to allow your ALB security group (`sgecslabloadbalancer`) to access the port range for ECS (0-65535) for port mapping. This rule references itself and you will see the security group appears when you start typing “sg-” in the Source textbox for the All TCP rule.


sg-ddfc83af | sgecslabpubliccluster

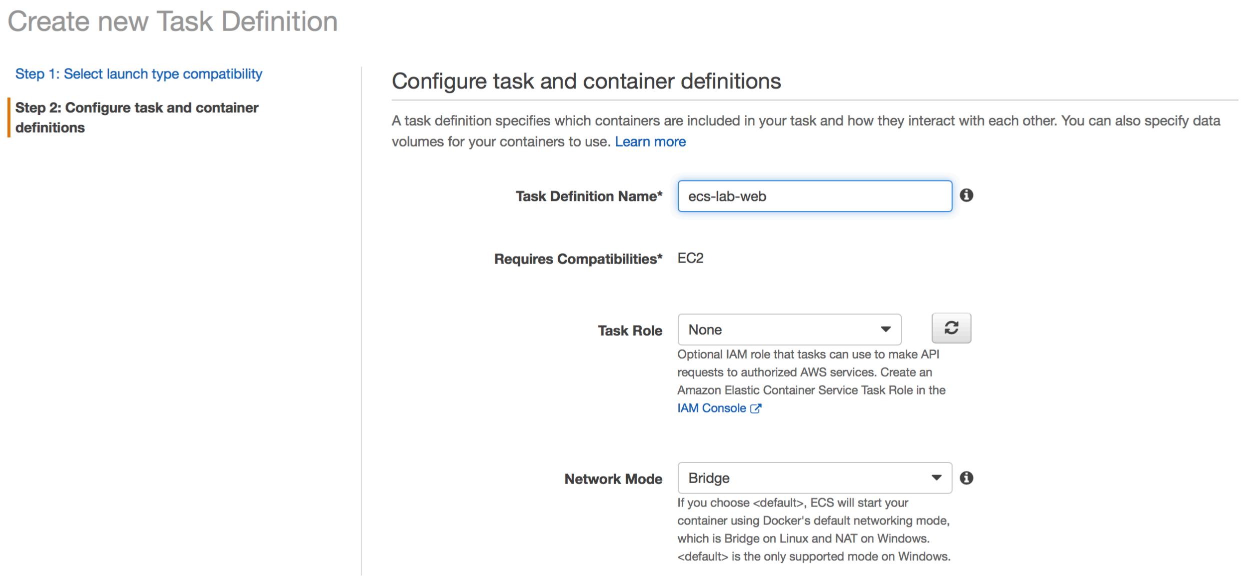
Type	Protocol	Port Range	Source	Description
HTTP (80)	TCP (6)	80	0.0.0.0/0	
ALL TCP	TCP (6)	ALL	sg-98a7e5ea	sgecslabloadbalancer

We now have the following security group setup:



10. Creating the Task Definitions

We need to create a service in ECS but before that can be done, the container needs be a part of a Task Definition. Task Definitions define things like environment variables, the container image you wish to use, and the resources you want to allocate to the service (port, memory, CPU). To create a Task Definition, choose Task Definitions from the ECS console menu. Then, choose **Create a Task Definition**. For launch type compatibility, select EC2, **Next Step**.



Create new Task Definition

Step 1: Select launch type compatibility

Step 2: Configure task and container definitions

Configure task and container definitions

A task definition specifies which containers are included in your task and how they interact with each other. You can also specify data volumes for your containers to use. Learn more

Task Definition Name* ecs-lab-web

Requires Compatibilities* EC2

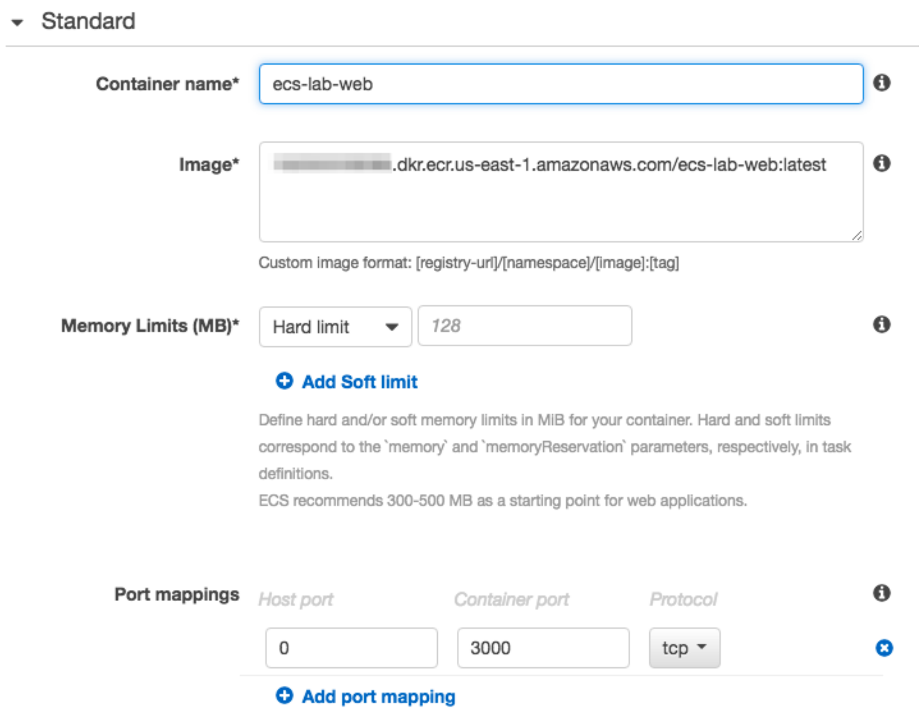
Task Role None

Optional IAM role that tasks can use to make API requests to authorized AWS services. Create an Amazon Elastic Container Service Task Role in the IAM Console

Network Mode Bridge

If you choose <default>, ECS will start your container using Docker's default networking mode, which is Bridge on Linux and NAT on Windows. <default> is the only supported mode on Windows.

Scroll down and leave the default values for the “Task execution IAM role” and “Task size” sections. Click on the **Add Container** button. Use *ecs-lab-web* for **Container name**. In the **Image** textbox, paste the Image URI that you used to push the web image to ECR from the previous lab. You can also find the web URI in the ECR web repo (look for the value for **Repository URI**). For **Memory Limit**, use a value of 128.



▾ Standard

Container name*: ecs-lab-web

Image*: [redacted].dkr.ecr.us-east-1.amazonaws.com/ecs-lab-web:latest

Custom image format: [registry-url]/[namespace]/[image]:[tag]

Memory Limits (MB)*: Hard limit ▾ 128

Add Soft limit

Define hard and/or soft memory limits in MiB for your container. Hard and soft limits correspond to the `memory` and `memoryReservation` parameters, respectively, in task definitions.
ECS recommends 300-500 MB as a starting point for web applications.

Port mappings

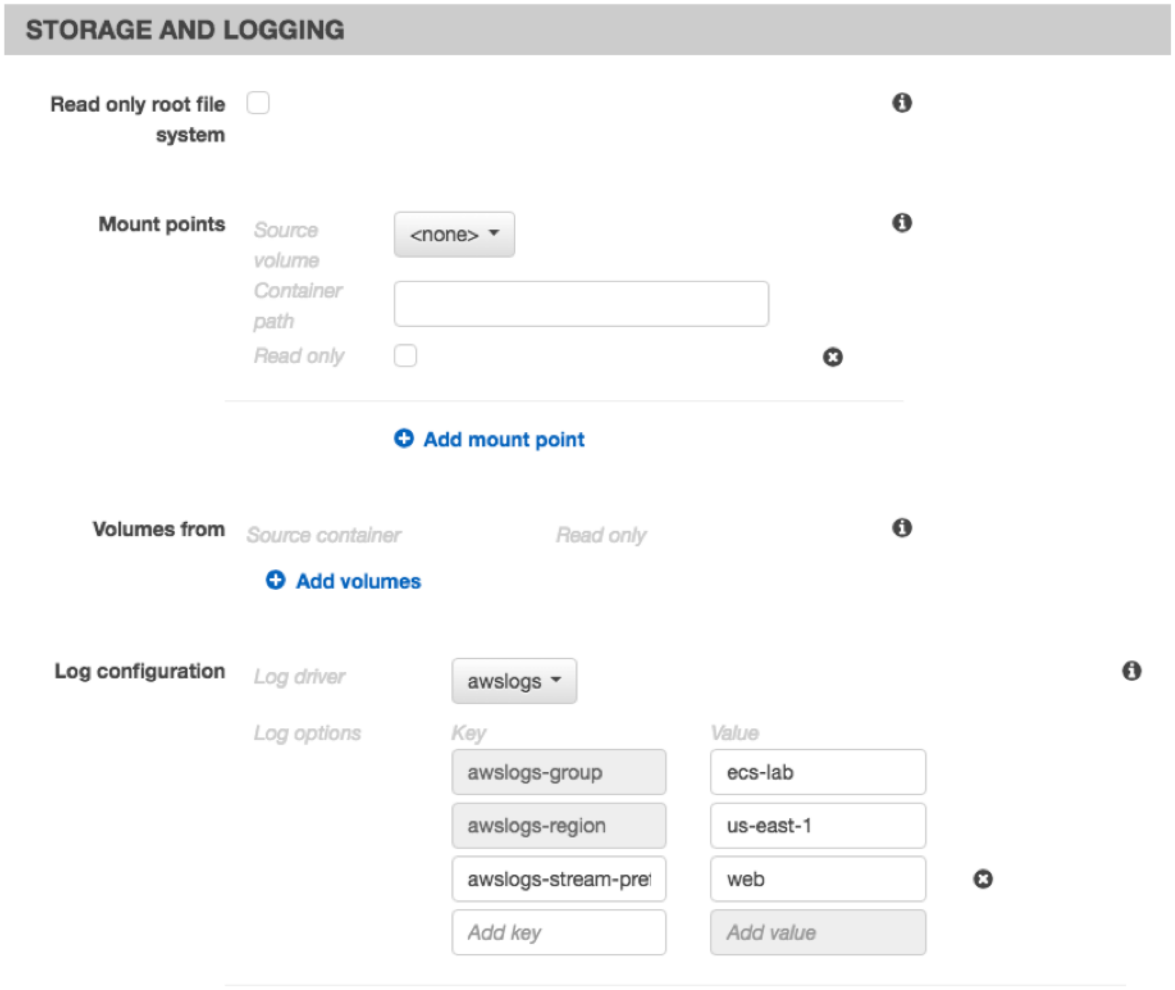
Host port	Container port	Protocol
0	3000	tcp ▾

Add port mapping

A few things to note here:

- We've specified a specific container image, including the `:latest` tag. Although it's not important for this lab, in a production environment where you were creating Task Definitions programmatically from a CI/CD pipeline, Task Definitions could include a specific SHA hash, or a more accurate tag.
- Under **Port Mappings**, we've specified a **Container Port** (3000), but left **Host Port** as 0. This is required to facilitate dynamic port allocation. This means that we don't need to map the Container Port to a specific Host Port in our Container Definition; instead, we can let the ALB allocate a port during task placement. To learn more about port allocation, check out the ECS documentation here.

Once you've specified your Port Mappings, scroll down and add a log driver. There are a few options here, but for this lab, choose **awslogs**:



STORAGE AND LOGGING

Read only root file system □

Mount points

Source volume: <none>

Container path:

Read only □

Add mount point

Volumes from: Source container, Read only

Add volumes

Log configuration

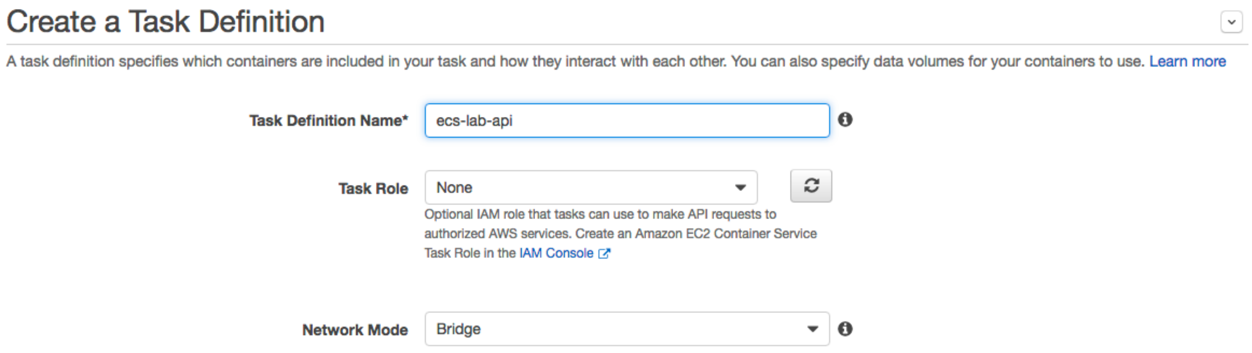
Log driver: awslogs

Log options

Key	Value
awslogs-group	ecs-lab
awslogs-region	us-east-1
awslogs-stream-pre	web
Add key	Add value

For this web container, make sure the **awslogs-stream-prefix** is **web**. Once you've added your log driver, save the Container Definition by clicking **Add**, and click on Create to complete the Task Definition.

Repeat the Task Definition creation process with the API container, taking care to use the api container image registry, and the correct port (8000) for the **Container Port** option. For the log driver, make sure the **awslogs-stream-prefix** is **api**.



Create a Task Definition

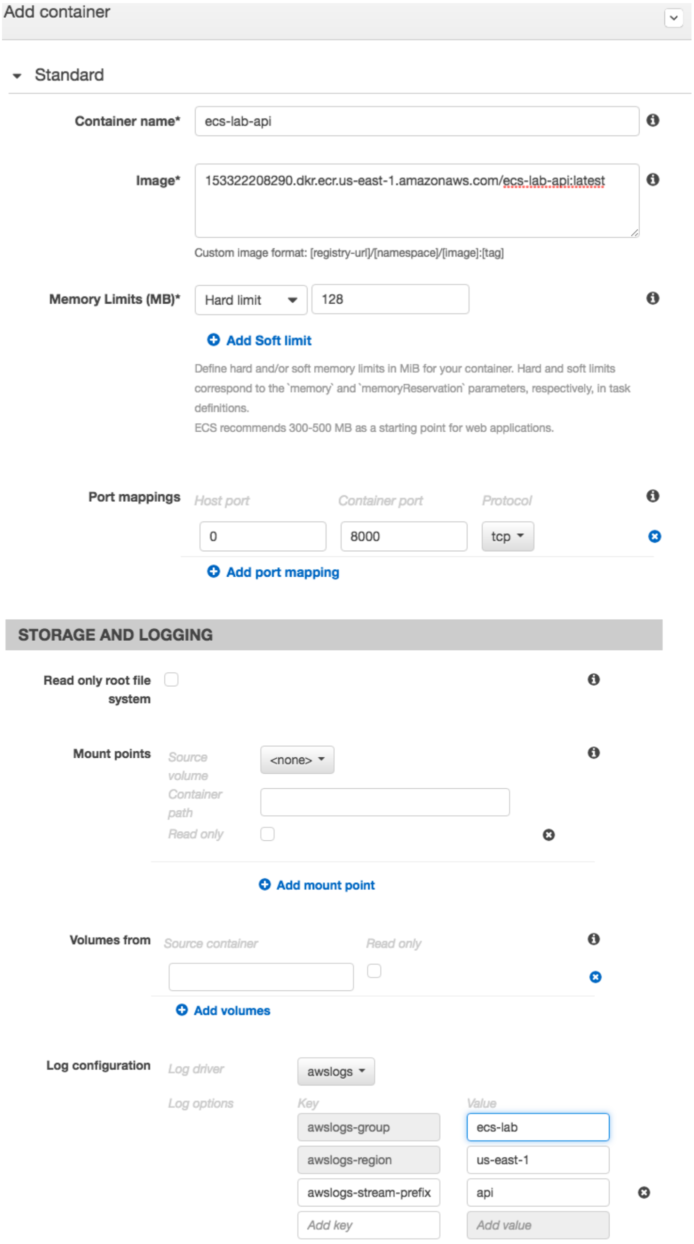
A task definition specifies which containers are included in your task and how they interact with each other. You can also specify data volumes for your containers to use. Learn more

Task Definition Name* ecs-lab-api

Task Role None

Optional IAM role that tasks can use to make API requests to authorized AWS services. Create an Amazon EC2 Container Service Task Role in the IAM Console

Network Mode Bridge



Add container

Standard

Container name* ecs-lab-api

Image* 153322208290.dkr.ecr.us-east-1.amazonaws.com/ecs-lab-api:latest

Custom image format: [registry-url]/[namespace]/[image]:[tag]

Memory Limits (MB)* Hard limit 128

Add Soft limit

Define hard and/or soft memory limits in MiB for your container. Hard and soft limits correspond to the `memory` and `memoryReservation` parameters, respectively, in task definitions.
ECS recommends 300-500 MB as a starting point for web applications.

Port mappings

Host port	Container port	Protocol
0	8000	tcp

Add port mapping

STORAGE AND LOGGING

Read only root file system □

Mount points

Source volume <none>

Container path

Read only □

Add mount point

Volumes from

Source container

Read only □

Add volumes

Log configuration

Log driver awslogs

Log options

Key	Value
awslogs-group	ecs-lab
awslogs-region	us-east-1
awslogs-stream-prefix	api
Add key	Add value

Don’t forget to click on the **Create** button to complete the Task Definition.

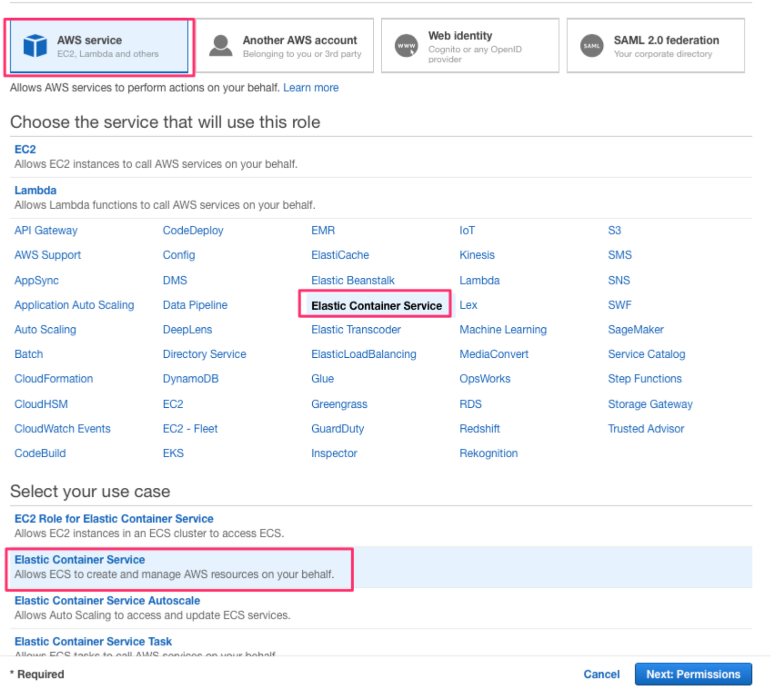
Next, create the log group by navigating to the CloudWatch **> Logs > Actions > Create Log Group**

Field	Value
Log Group Name	`ecs-lab`

11. Creating the Services

Next, we're going to create the service based on our Task Definition. A service is a group of tasks (which are containers). You can define how many tasks you want to run simultaneously, specify load balancing, auto scaling and configure many other options.

First, we need to create a IAM role for this Service. Navigate to IAM > Roles > Create new role:



AWS service
EC2, Lambda and others

Another AWS account
Belonging to you or 3rd party

Web identity
Cognito or any OpenID provider

SAML 2.0 federation
Your corporate directory

Allows AWS services to perform actions on your behalf. Learn more

Choose the service that will use this role

EC2
Allows EC2 instances to call AWS services on your behalf.

Lambda
Allows Lambda functions to call AWS services on your behalf.

API Gateway
AWS Support
AppSync
Application Auto Scaling
Auto Scaling
Batch
CloudFormation
CloudHSM
CloudWatch Events
CodeBuild

CodeDeploy
Config
DMS
Data Pipeline
DeepLens
Directory Service
DynamoDB
EC2
EC2 - Fleet
EKS

EMR
ElastiCache
Elastic Beanstalk
Elastic Container Service
Elastic Transcoder
ElasticLoadBalancing
Glue
Greengrass
GuardDuty
Inspector

IoT
Kinesis
Lambda
Lex
Machine Learning
MediaConvert
OpsWorks
RDS
Redshift
Rekognition

S3
SMS
SNS
SWF
SageMaker
Service Catalog
Step Functions
Storage Gateway
Trusted Advisor

Select your use case

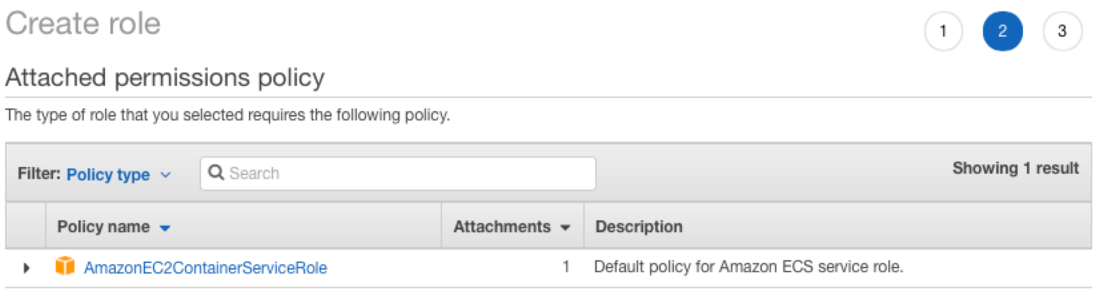
EC2 Role for Elastic Container Service
Allows EC2 instances in an ECS cluster to access ECS.

Elastic Container Service
Allows ECS to create and manage AWS resources on your behalf.

Elastic Container Service Autoscale
Allows Auto Scaling to access and update ECS services.

Elastic Container Service Task
Allows ECS tasks to call AWS services on your behalf.

Click Next: Permissions

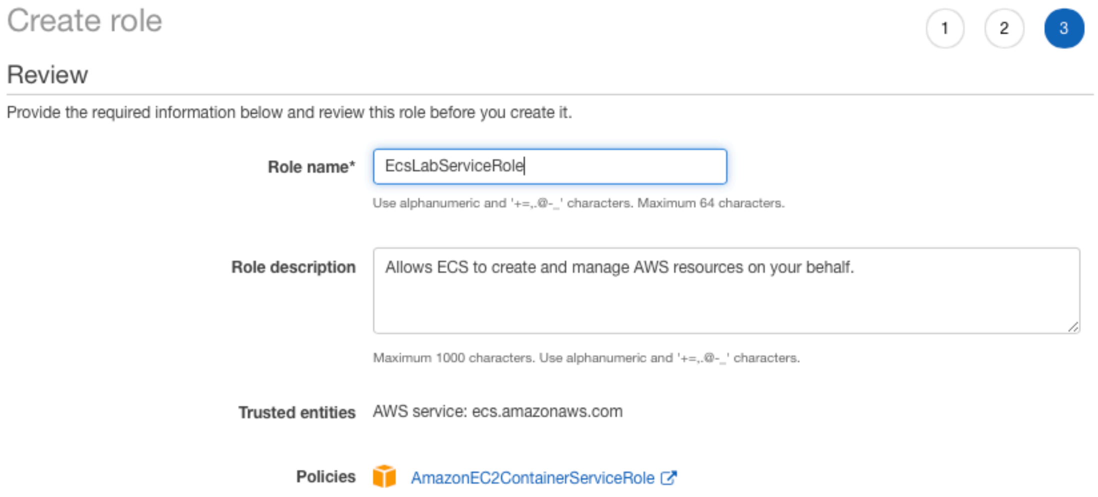


Attached permissions policy

The type of role that you selected requires the following policy.

Policy name	Attachments	Description
AmazonEC2ContainerServiceRole	1	Default policy for Amazon ECS service role.

Click **Next:Review**. In the Review page, use **EcsLabServiceRole** for the role name and click the **Create Role** button.



Review

Provide the required information below and review this role before you create it.

Role name* EcsLabServiceRole

Use alphanumeric and '+=,.@-_' characters. Maximum 64 characters.

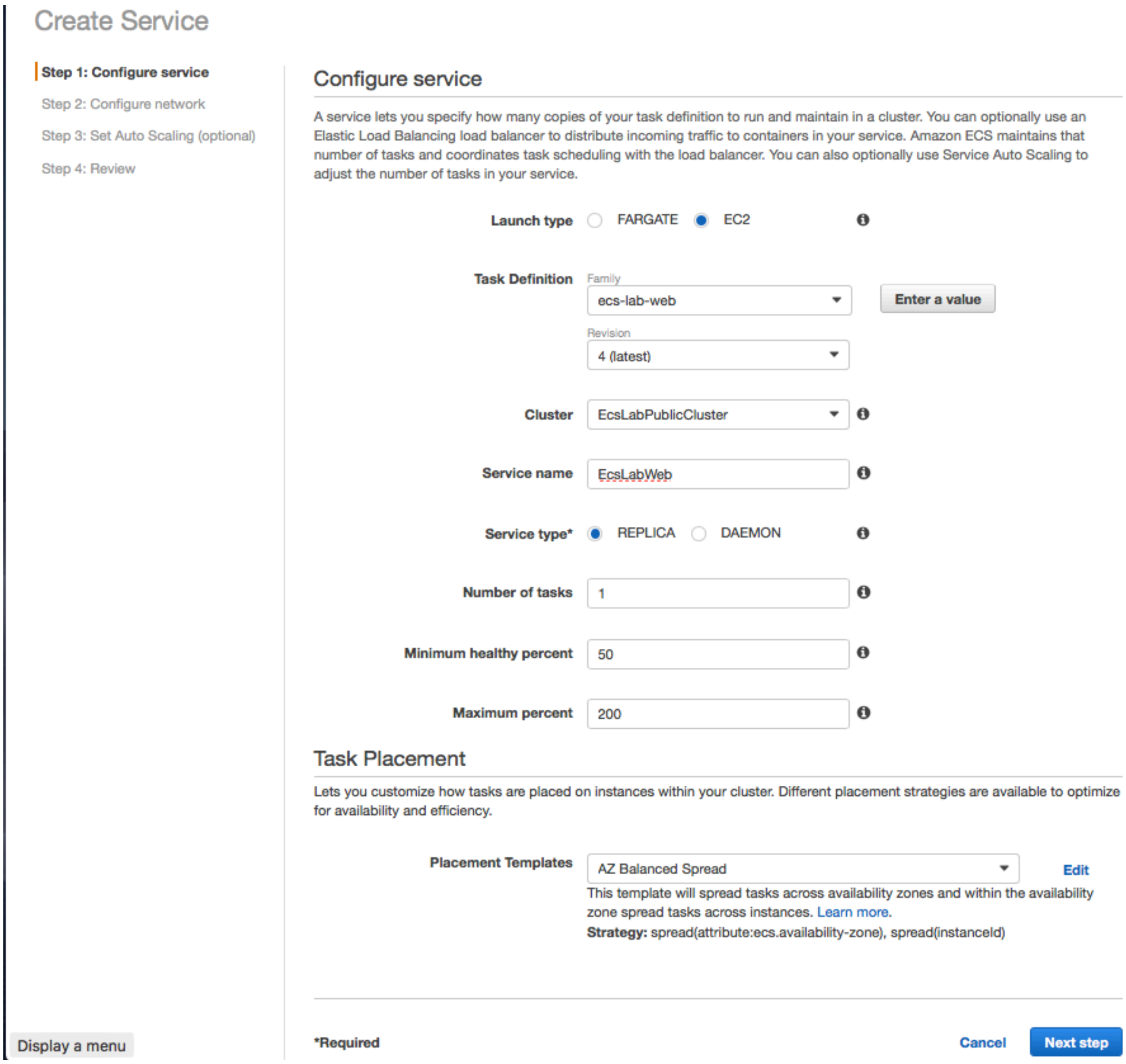
Role description Allows ECS to create and manage AWS resources on your behalf.

Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.

Trusted entities AWS service: ecs.amazonaws.com

Policies AmazonEC2ContainerServiceRole

Navigate back to the ECS console, and choose the cluster that you created. This should be named **EcsLabPublicCluster**. From the cluster detail page, choose **Services** > **Create**. Make sure the launch type is EC2 (not Fargate) and configure the service as follows:



Create Service

- Step 1: Configure service
- Step 2: Configure network
- Step 3: Set Auto Scaling (optional)
- Step 4: Review

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.

Launch type ○ FARGATE ● EC2

Task Definition Family: ecs-lab-web

Enter a value

Revision: 4 (latest)

Cluster: EcsLabPublicCluster

Service name: EcsLabWeb

Service type* ● REPLICA ○ DAEMON

Number of tasks: 1

Minimum healthy percent: 50

Maximum percent: 200

Task Placement

Lets you customize how tasks are placed on instances within your cluster. Different placement strategies are available to optimize for availability and efficiency.

Placement Templates: AZ Balanced Spread

This template will spread tasks across availability zones and within the availability zone spread tasks across instances. Learn more.

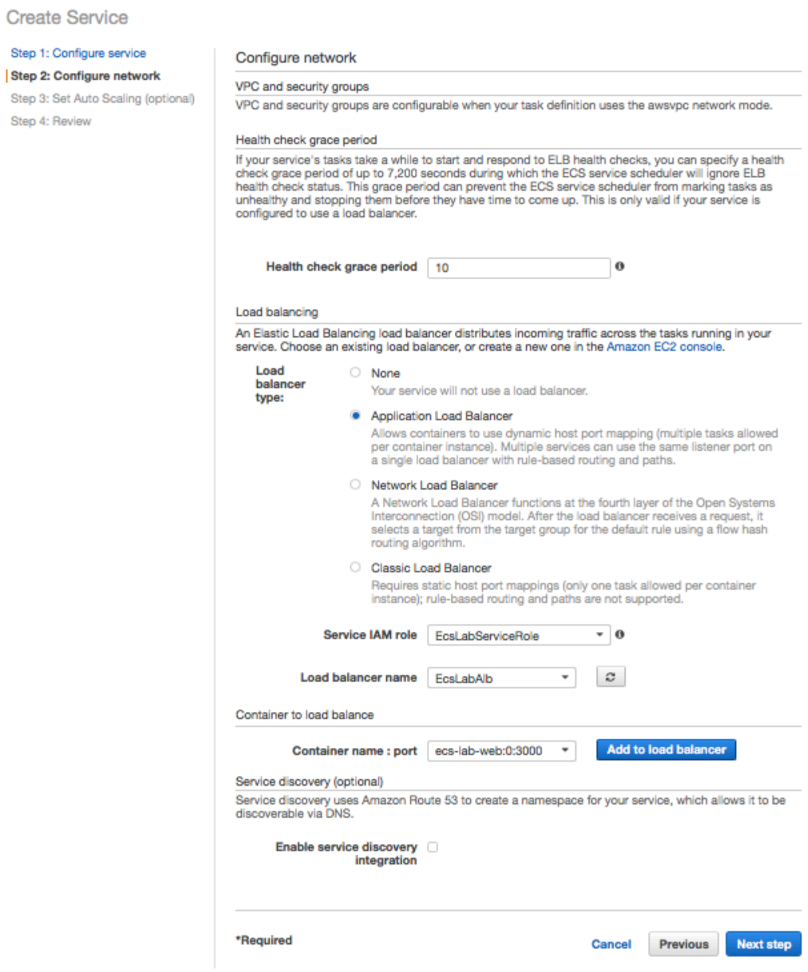
Strategy: spread(attribute:ecs.availability-zone), spread(instanceId)

*Required

Choose the web Task Definition you created in the previous section. For the purposes of this lab, we'll only start one copy of each task. In a production environment, you will always want more than one copy of each task running for reliability and availability.

You can keep the default **AZ Balanced Spread** for the Task Placement Policy. To learn more about the different Task Placement Policies, see the documentation, or this blog post. Click **Next step** to configure load balancing.

Choose Application Load Balancer and configure as follows:



Create Service

Step 1: Configure service
Step 2: Configure network
Step 3: Set Auto Scaling (optional)
Step 4: Review

Configure network

VPC and security groups

VPC and security groups are configurable when your task definition uses the awsvpc network mode.

Health check grace period

If your service's tasks take a while to start and respond to ELB health checks, you can specify a health check grace period of up to 7,200 seconds during which the ECS service scheduler will ignore ELB health check status. This grace period can prevent the ECS service scheduler from marking tasks as unhealthy and stopping them before they have time to come up. This is only valid if your service is configured to use a load balancer.

Health check grace period 10

Load balancing

An Elastic Load Balancing load balancer distributes incoming traffic across the tasks running in your service. Choose an existing load balancer, or create a new one in the Amazon EC2 console.

Load balancer type:

○ None
Your service will not use a load balancer.

● Application Load Balancer
Allows containers to use dynamic host port mapping (multiple tasks allowed per container instance). Multiple services can use the same listener port on a single load balancer with rule-based routing and paths.

○ Network Load Balancer
A Network Load Balancer functions at the fourth layer of the Open Systems Interconnection (OSI) model. After the load balancer receives a request, it selects a target from the target group for the default rule using a flow hash routing algorithm.

○ Classic Load Balancer
Requires static host port mappings (only one task allowed per container instance); rule-based routing and paths are not supported.

Service IAM role EcsLabServiceRole

Load balancer name EcsLabAlb

Container to load balance

Container name : port ecs-lab-web:0:3000

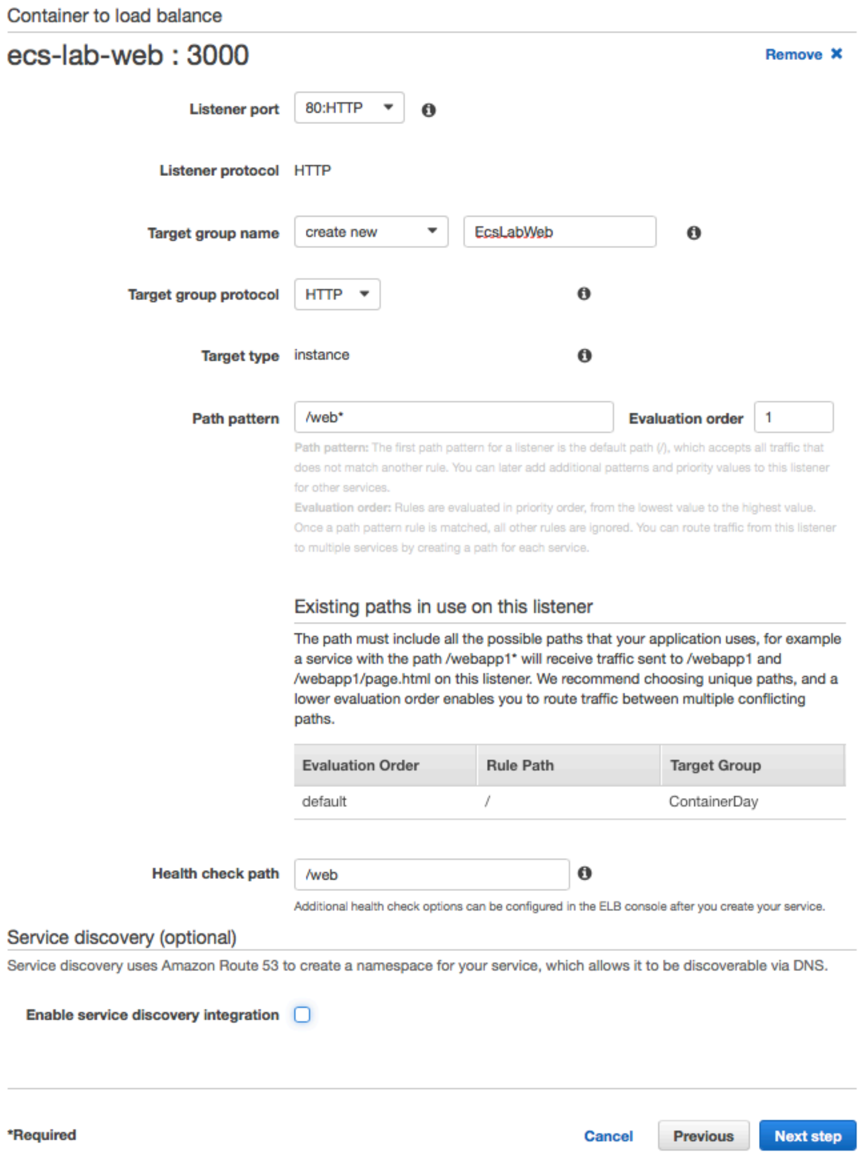
Service discovery (optional)

Service discovery uses Amazon Route 53 to create a namespace for your service, which allows it to be discoverable via DNS.

Enable service discovery integration □

*Required

Select the web container, choose **Add to load balancer** and configure load balancing.



Container to load balance

ecs-lab-web : 3000

Listener port: 80:HTTP

Listener protocol: HTTP

Target group name: create new | EcsLabWeb

Target group protocol: HTTP

Target type: instance

Path pattern: /web*

Evaluation order: 1

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.

Evaluation order: Rules are evaluated in priority order, from the lowest value to the highest value. Once a path pattern rule is matched, all other rules are ignored. You can route traffic from this listener to multiple services by creating a path for each service.

Existing paths in use on this listener

The path must include all the possible paths that your application uses, for example a service with the path /webapp1* will receive traffic sent to /webapp1 and /webapp1/page.html on this listener. We recommend choosing unique paths, and a lower evaluation order enables you to route traffic between multiple conflicting paths.

Evaluation Order	Rule Path	Target Group
default	/	ContainerDay

Health check path: /web

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

Service discovery (optional)

Service discovery uses Amazon Route 53 to create a namespace for your service, which allows it to be discoverable via DNS.

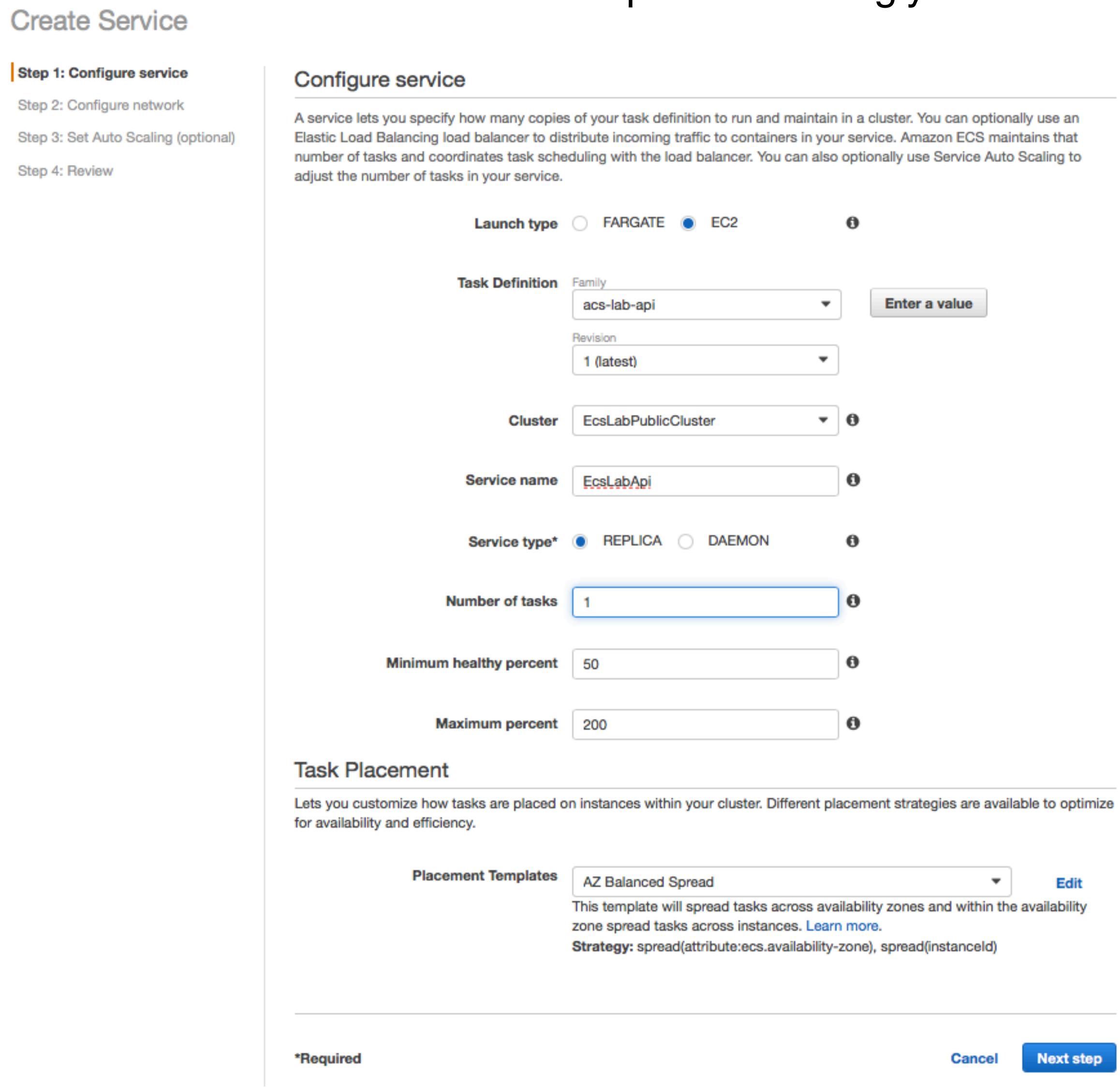
Enable service discovery integration ☐

*Required

When we created our ALB, we only added a listener for HTTP:80. Select this from the dropdown as the value for **Listener**. For **Target Group Name**, enter a value that will make sense to you later, like **ecs-lab-web**. For **Path Pattern**, the value should be `/web*`. This is the route that we specified in our Python application.

If the values look correct, click **Next Step**, click through the optional Auto Scaling page click **Create Service.**

Repeat this process for the `api` microservice and task definition. Don't forget to adjust the target group name, path pattern, evaluation order and health check path accordingly.



Create Service

- Step 1: Configure service
- Step 2: Configure network
- Step 3: Set Auto Scaling (optional)
- Step 4: Review

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.

Launch type ○ FARGATE ● EC2

Task Definition Family: acs-lab-api — Enter a value

Revision: 1 (latest)

Cluster: EcsLabPublicCluster

Service name: EcsLabApi

Service type* ● REPLICA ○ DAEMON

Number of tasks: 1

Minimum healthy percent: 50

Maximum percent: 200

Task Placement

Lets you customize how tasks are placed on instances within your cluster. Different placement strategies are available to optimize for availability and efficiency.

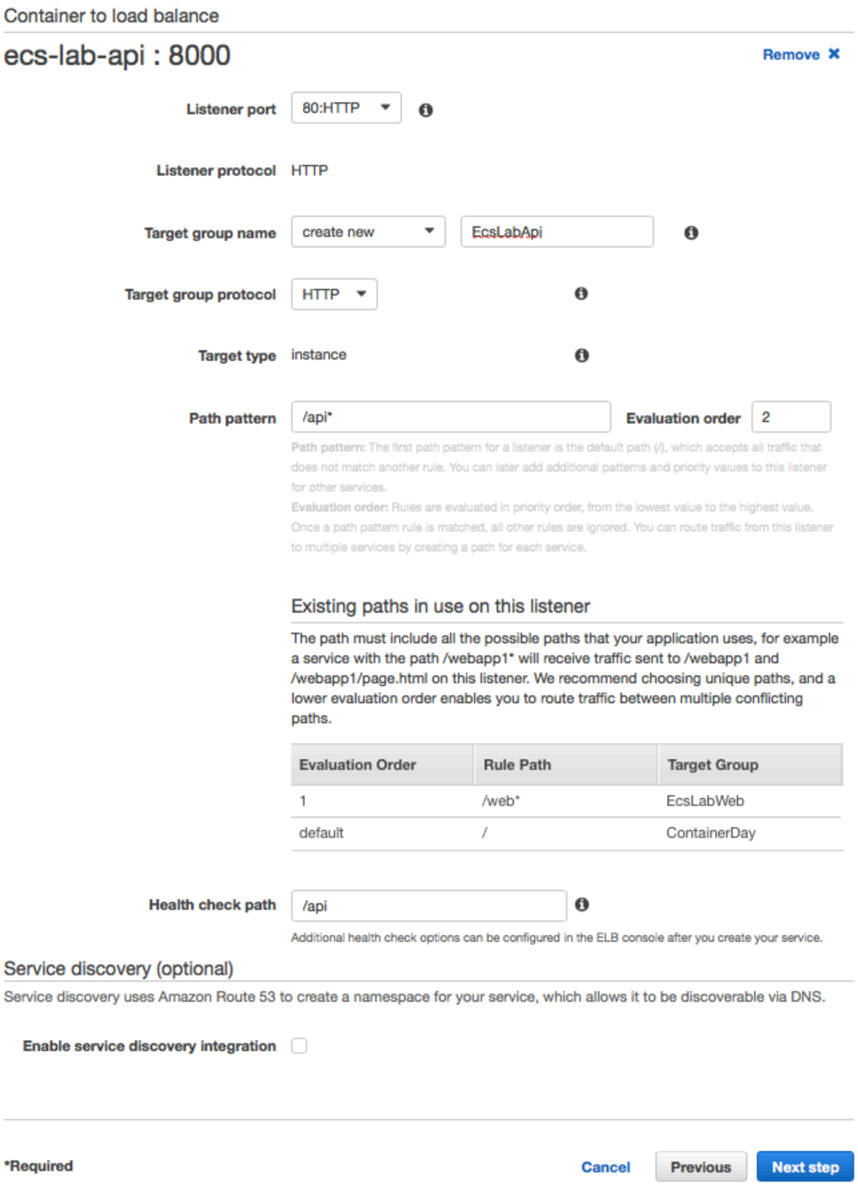
Placement Templates: AZ Balanced Spread — Edit

This template will spread tasks across availability zones and within the availability zone spread tasks across instances. Learn more.

Strategy: spread(attribute:ecs.availability-zone), spread(instanceId)

*Required

Cancel | Next step



Container to load balance

ecs-lab-api : 8000

Listener port: 80:HTTP

Listener protocol: HTTP

Target group name: create new | EcsLabApi

Target group protocol: HTTP

Target type: instance

Path pattern: /api*

Evaluation order: 2

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.

Evaluation order: Rules are evaluated in priority order, from the lowest value to the highest value. Once a path pattern rule is matched, all other rules are ignored. You can route traffic from this listener to multiple services by creating a path for each service.

Existing paths in use on this listener

The path must include all the possible paths that your application uses, for example a service with the path /webapp1* will receive traffic sent to /webapp1 and /webapp1/page.html on this listener. We recommend choosing unique paths, and a lower evaluation order enables you to route traffic between multiple conflicting paths.

Evaluation Order	Rule Path	Target Group
1	/web*	EcsLabWeb
default	/	ContainerDay

Health check path: /api

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

Service discovery (optional)

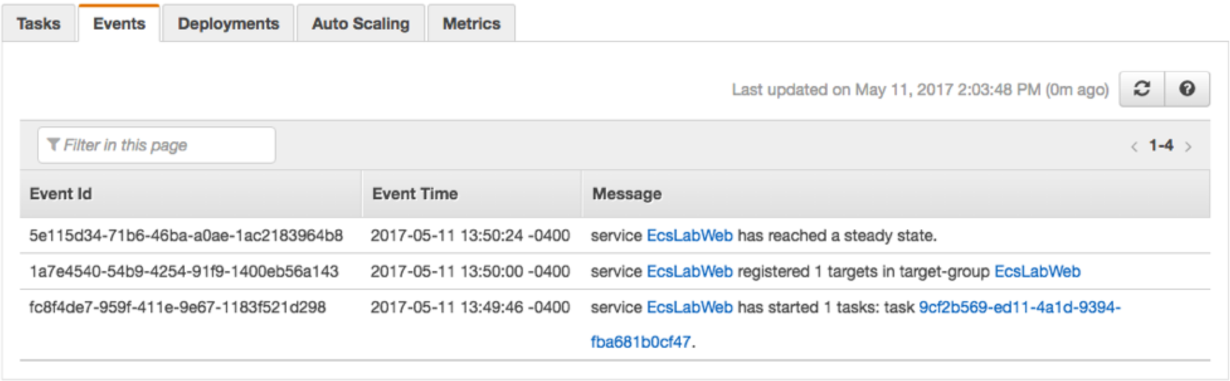
Service discovery uses Amazon Route 53 to create a namespace for your service, which allows it to be discoverable via DNS.

Enable service discovery integration □

*Required

12. Testing our service deployments from the console and the ALB

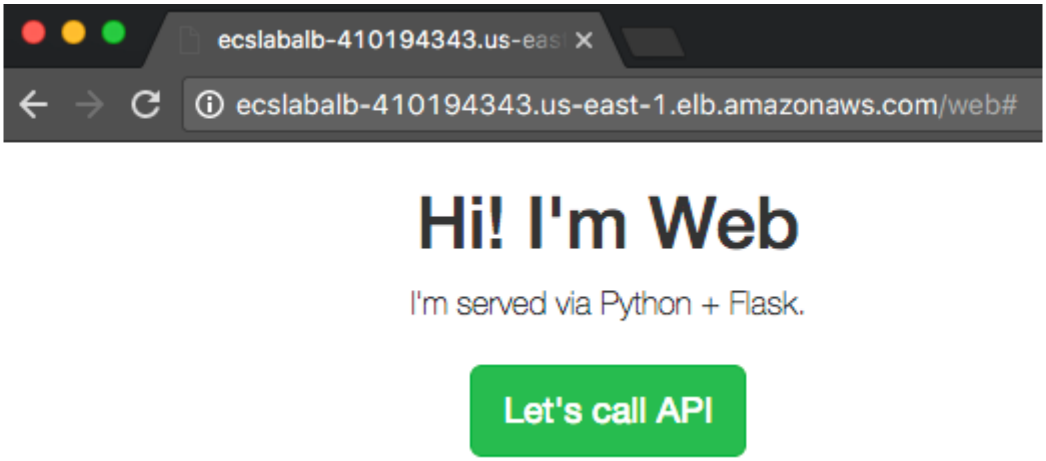
You can see service level events from the ECS console. This includes deployment events. You can test that both of your services are deployed and registered properly with the ALB by looking at the service's **Events** tab:



Last updated on May 11, 2017 2:03:48 PM (0m ago)

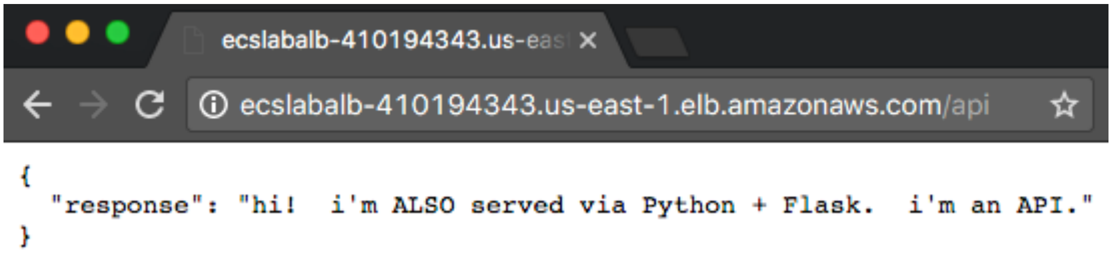
Event Id	Event Time	Message
5e115d34-71b6-46ba-a0ae-1ac2183964b8	2017-05-11 13:50:24 -0400	service EcsLabWeb has reached a steady state.
1a7e4540-54b9-4254-91f9-1400eb56a143	2017-05-11 13:50:00 -0400	service EcsLabWeb registered 1 targets in target-group EcsLabWeb
fc8f4de7-959f-411e-9e67-1183f521d298	2017-05-11 13:49:46 -0400	service EcsLabWeb has started 1 tasks: task 9cf2b569-ed11-4a1d-9394-fba681b0cf47.

We can also test from the ALB itself. To find the DNS A record for your ALB, navigate to the EC2 Console > **Load Balancers** > **Select your Load Balancer**. Under **Description**, you can find details about your ALB, including a section for **DNS Name**. You can enter this value in your browser, and append the endpoint of your service, to see your ALB and ECS Cluster in action:


ecslabalb-410194343.us-eas
ecslabalb-410194343.us-east-1.elb.amazonaws.com/web#
Let's call API

Hi! I'm Web

I'm served via Python + Flask.

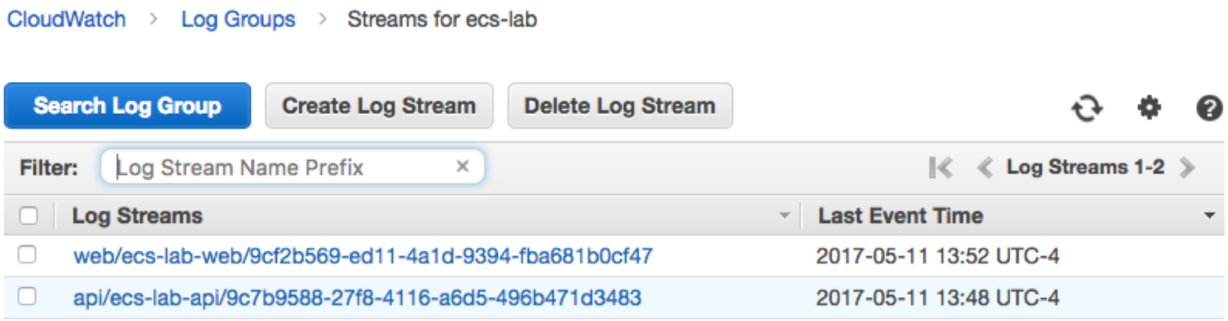

ecslabalb-410194343.us-eas
ecslabalb-410194343.us-east-1.elb.amazonaws.com/api

```
{
  "response": "hi!  i'm ALSO served via Python + Flask.  i'm an API."
}
```

The ALB routes traffic appropriately based on the paths we specified when we registered the containers: `/web*` requests go to our web service, and `/api*` requests go to our API service.

13. More in-depth logging with CloudWatch

When we created our Container Definitions, we also added the awslogs driver, which sends logs to CloudWatch. You can see more details logs for your services by going to the CloudWatch console, and selecting first our log group `ecs-lab` and then choosing an individual stream:



☐	Log Streams	Last Event Time
☐	web/ecs-lab-web/9cf2b569-ed11-4a1d-9394-fba681b0cf47	2017-05-11 13:52 UTC-4
☐	api/ecs-lab-api/9c7b9588-27f8-4116-a6d5-496b471d3483	2017-05-11 13:48 UTC-4

That's a wrap!

Congratulations! You've deployed an ECS Cluster with two working endpoints.

Clean up

Don't forget to do the following, after you're finished with the lab:

- Delete the `ecs-lab` stack
- Go to **CloudWatch Console > Logs** and delete Log Group `ecs-lab`
- Go to **ECS Console > Repositories** and delete the cluster, deregister the 2 task definitions, delete the 2 created repositories
- Go to the **EC2 Console**, terminate the `ecs-lab-workstation` EC2 Instance, the Application Load Balancer and the 3 Target Groups
- Go to **IAM console** and delete the 2 roles `EcslabInstanceRole` and `EcsWorkstationRole`

Find the above a little boring?

Here are some ideas to make it more interesting:

- The development team refactored our api and now it requires a host with GPU. Deploy the api containers to EC2 P2 GPU instances by defining a Task Placement Constraint.