

Lab. ECS + deploy tools

- Introduction
- Creating the EC2 Container Repository
- Instructions
 - Summary
- Building the Docker Images
 - Instructions
 - Summary
- Creating the ECS Cluster
 - Introduction
 - Instructions
 - Summary
- Creating the Task Definitions
 - Introduction
 - Instructions
 - Summary
- Creating the Target Group and Load Balancer
 - Introduction
 - Instructions
 - Summary
- Creating the ECS Services
 - Introduction
 - Instructions
 - Summary
- Viewing Instances and Application Message
 - Introduction
 - Instructions
 - Summary
- Updating the Services and Application Deployments
 - Introduction
 - Instructions
 - Summary
- Deleting the ECS Laboratory Resources
 - Introduction
 - Instructions
 - Summary

Introduction

In this Lab Step you will use Amazon ECR to create your own fully-managed Docker container registry within Amazon ECS. ECR hosts your images in a highly available and scalable architecture, and features the same security features (secure transmission, encryption) as other Amazon services. The ECR also integrates with the Docker CLI and ECS to allow for easy and convenient push, pull, and deployment operations.

Creating the EC2 Container Repository

Instructions

1. Navigate to Services > Compute > EC2 Container Service, or search for *ECS* in the service search box:

The screenshot shows the AWS Services menu with the following items:

- Services ▾
- Resource Groups ▾
- History
- Console Home
- S3
- EC2
- IAM
- CloudFormation
- VPC

A red arrow points from the "EC2" item towards the "Compute" section on the right.

Find a service by name or feature (

- Compute
- EC2
- EC2 Container Service
- Lightsail
- Elastic Beanstalk
- Lambda
- Batch

2. Click the blue Get started button.

3. Uncheck the box next to Deploy a sample application onto an Amazon ECS Cluster. Leave the box next to Store container images securely with Amazon ECR checked. Click Continue:

Getting Started with Amazon EC2 Container Service (ECS)

Select options to configure

Get started by running a sample app with EC2 Container Service (ECS), setting up a private image repository with EC2 Container Registry (ECR), or both.

- I want to
- Deploy a sample application onto an Amazon ECS Cluster
Amazon ECS will set up an autoscaling group and help you create other resources to facilitate cluster management.
 - Store container images securely with Amazon ECR
Create and manage a new private image repository and use the Docker CLI to push and pull images. Access to the repository is managed through AWS Identity and Access Management.

[Cancel](#) [Continue](#)

4. In the Configure repository section, enter *ca-container-registry* as the Repository name.

5. Copy and paste the Repository URI (Uniform Resource Identifier). You will need this URI to reference your Docker images in a later Lab Step. Click Next step:

Get started with EC2 Container Registry

Step 1: Configure repository

Step 2: Build, tag, and push Docker image

Configure repository

This wizard will guide you through the steps of creating a repository in EC2 Container Registry. [Learn more](#)

Repository name*

Namespaces are optional, and they can be included in the repository name with a slash (for example, namespace/repo)

Repository URI

Permissions

As the owner, you have access to this repository by default. After completing this wizard, you can grant others permission to access this repository in the console.

*Required

[Cancel](#) [Next step](#)

6. Although not used in this Lab, Step 2: Build, tag, and push Docker image provides you with a convenient list of Docker commands configured to reference your new repository. You can reference these commands whenever you need by revisiting your repository and clicking the View Push Commands button. Click Done:

Build, tag, and push Docker image

Now that your repository exists, you can push a Docker image by following these steps:

 Successfully created repository

358259402422.dkr.ecr.us-west-2.amazonaws.com/ca-container-registry

To install the AWS CLI and Docker and for more information on the steps below, visit the ECR documentation page.

1) Retrieve the docker login command that you can use to authenticate your Docker client to your registry:

Note:

If you receive an "Unknown options: --no-include-email" error, install the latest version of the AWS CLI. [Learn more](#)

```
aws ecr get-login --no-include-email --region us-west-2
```

2) Run the docker login command that was returned in the previous step.

Note:

If you are using Windows PowerShell, run the following command instead.

```
Invoke-Expression -Command (aws ecr get-login --no-include-email --region us-west-2)
```

3) Build your Docker image using the following command. For information on building a Docker file from scratch see the instructions [here](#). You can skip this step if your image is already built:

```
docker build -t ca-container-registry .
```

4) After the build completes, tag your image so you can push the image to this repository:

```
docker tag ca-container-registry:latest 358259402422.dkr.ecr.us-west-2.amazonaws.com/ca-container-registry:latest
```

5) Run the following command to push this image to your newly created AWS repository:

```
docker push 358259402422.dkr.ecr.us-west-2.amazonaws.com/ca-container-registry:latest
```

*Required

[Done](#)

7. Click the Repositories link and view your new, currently-empty repository. This location is a convenient place to view your repository URI, push commands, and list of images whenever required:

Amazon ECS

Clusters

Task Definitions

Repositories

< All repositories : ca-container-registry

Repository ARN: arn:aws:ecr:us-west-2:358259402422:repository/ca-container-registry

Repository URI: 358259402422.dkr.ecr.us-west-2.amazonaws.com/ca-container-registry

View Push Commands

Images **Permissions**

Summary

In this Lab Step you created a repository for Docker images within AWS. This allows you to integrate your Docker images more closely with ECS and other Amazon services, such as CodeBuild.

Building the Docker Images

CodeBuild is a fully-managed build service. CodeBuild can compile code, run tests, and produce packages you can deploy on any compatible resource. Using CodeBuild in this Lab eliminates the need for you to create a build server or install any software on your local machine. In this Lab Step you will use CodeBuild to build the Docker images containing the two applications you need to complete the Lab.

Prepare files:

In your Lab folder `ecs_lab.zip` you have two more folders: `ecslabgreen` and `ecslabblue`. In each of them exists a file `server.js`. Replace `? in line app.use('/devops?/api', router);` with your number. Put all files in each directory in zip archives `ecslabgreen.zip` and `ecslabblue.zip`.

Instructions

1. Navigate to Services > Developer Tools > CodeBuild, or search for `CodeBuild` in the service search box:

Services

Resource Groups

History

Find a service by name or feature (for example, EC2, S3 or VM, storage).

EC2 Container Service

Compute

EC2

EC2 Container Service

Lightsail

Elastic Beanstalk

Lambda

Batch

Developer Tools

CodeStar

CodeCommit

CodeBuild

CodeDeploy

CodePipeline

X-Ray

2. Click Get started. If you arrive at the Build Projects page, click Create project:

Build projects

Review your build projects and start new builds.

Start build Create project Actions ▾

Project	Repository
---------	------------

i Start new build project

You don't currently have any build projects. [Create a new build project.](#)

3. In the Step 1: Configure project area specify the following details:

- Project name: *ecslab-blue-project*
- Source provider: Amazon S3
- Bucket: devops?
- S3 object key: *ecslabblue.zip*
- Environment image: Use an image managed by AWS CodeBuild
- Operating system: Ubuntu
- Runtime: Docker
- Version: aws/codebuild/docker:1.12.1

Note: The Docker version listed is the latest at lab creation. In the future, the version will update. Choose the latest version if the version listed is unavailable.

- Build specification: Use the `buildspec.yml` in the source code root directory

Configure your project

Specify settings for your build project.

Project name* i

Source: What to build

Source provider* ▼

Bucket* i

S3 object key* i

Environment: How to build

Environment image* Use an image managed by AWS CodeBuild
 Specify a Docker image

Operating system* ▼

Runtime* ▼

Version* ▼

Build specification Use the buildspec.yml in the source code root directory
 Insert build commands

The project source files and Amazon S3 bucket are created for you by the Cloud Academy Lab environment. The project uses Ubuntu Linux with Docker installed as the build environment to create the image. The build spec is a collection of build commands and settings that provide instructions on how to build the Docker image and where to push it once complete. For the purposes of differentiating the images and the applications they contain in this Lab, the build spec also tags the images with distinct names. CodeBuild will tag your blue application image with *testblue* and your green application image with *testgreen* due to instructions in the source files.

4. Scroll down the page to continue entering the final project details. Click Continue:

- Artifacts type: No artifacts
- Service role: Choose an existing service role from your account
- Role name: CodeBuildServiceRole (Remove the check from Allow AWS CodeBuild to modify this service role...)

Artifacts: Where to put the artifacts from this build project

Artifacts type* No artifacts  

Service role

Specify a service role that enables AWS CodeBuild to call dependent AWS services on your behalf. [Learn more.](#)

- Create a service role in your account
 Choose an existing service role from your account

Role name*

CodeBuildServiceRole 

Allow AWS CodeBuild to modify this service role so it can be used with this build project

*Required

[Cancel](#)

[Continue](#)

5. Review the project parameters in Step 2: Review. Click Save and Build.

6. Expand the student account information in the upper-right corner of the screen and record the Account number:



student @ 3582-5940-2422 ^

Oregon ^

Support ^

IAM User:

student



Account:

3582-5940-2422

[My Account](#)

[My Organization](#)

[My Billing Dashboard](#)

[My Security Credentials](#)

[Switch Role](#)

[Sign Out](#)

7. Expand the Environment variables area in the Start new build section. In the Name field enter *AWS_ACCOUNT_ID*. In the Value field enter the 12-digit student account ID. Remove the hyphens as shown in the following image. Also you will need *AWS_REGION* set to your region.

Click Start build:

Start new build

Choose the build project you want to use. Optionally, you can build a specific version of the source code, and you can override any of the build project's settings for this build only.

Project name*

Source provider Amazon S3

Repository arn:aws:s3:::cloudacademylab-caecslab-9bfvbfvcvgq8l/ecslabblue.zip

Source version ⓘ

▶ Show advanced options

▼ Environment variables

Add environment variables (custom file paths, AWS resource IDs) that you want AWS CodeBuild to use.

Name	Value
AWS_ACCOUNT_ID	358259402422

+ Add row

*Required Cancel Start build

The sample application uses the AWS account variable to locate the Docker image repository you created earlier. Once CodeBuild finishes building the Docker image, it executes a Docker push command that stores the image in the repository.

8. Observe the Phase Details section as each step of the build process proceeds and completes. Wait approximately one minute until the Status field shows Succeeded. CodeBuild is building a Docker container image containing one of the custom applications (blue) made for this Lab:

ecslab-blue-project:e5c6ee2a-b25f-4aca-89d1-1d1030be6d59 Succeeded

Review your build details as it progresses.

Build

Build ARN	arn:aws:codebuild:us-west-2:358259402422:build/ecslab-blue-project:e5c6ee2a-b25f-4aca-89d1-1d1030be6d59
Build project	ecslab-blue-project
Source provider	Amazon S3
Repository	arn:aws:s3:::cloudacademylab-caecslab-9bfvbfqvqg8l/ecslabblue.zip
Start time	1 minute ago
End time	1 second ago
Status	Succeeded
Initiator	student

► Build details

► Environment variables

Phase details

Name	Status	Duration	Completed
► SUBMITTED	Succeeded		1 minute ago
► PROVISIONING	Succeeded	21 secs	48 seconds ago
► DOWNLOAD_SOURCE	Succeeded	4 secs	44 seconds ago
► INSTALL	Succeeded		44 seconds ago
► PRE_BUILD	Succeeded	4 secs	39 seconds ago
► BUILD	Succeeded	18 secs	21 seconds ago
► POST_BUILD	Succeeded	9 secs	12 seconds ago
► UPLOAD_ARTIFACTS	Succeeded		12 seconds ago
► FINALIZING	Succeeded	5 secs	6 seconds ago

9. Scroll down to the Build logs section to view information about the build process. This screen only shows the last 20 lines of the build log. The lines in the following image show part of the POST_BUILD stage where the layers of the image are pushed to the repository before finalizing the build:

Build logs

Showing the last 20 lines of build log below. [View entire log](#)

```
5bef08742407: Preparing
3e89bd0d5acf: Waiting
f7e883283ebc: Waiting
0a19bde117a5: Waiting
5bef08742407: Waiting
2531a8d82cf7: Pushed
e64991fc8bf5: Pushed
4d9359363687: Pushed
8253865cc65c: Pushed
26c56125406d: Pushed
3e89bd0d5acf: Pushed
f7e883283ebc: Pushed
5bef08742407: Pushed
0a19bde117a5: Pushed
testblue: digest: sha256:bfebadb6e7ebaeea21d2859d2c1ee8bae0b31db35d2dfa789d673a8aa0039fc1 size: 2200

[Container] 2017/07/21 20:17:17 Phase complete: POST_BUILD Success: true
[Container] 2017/07/21 20:17:17 Phase context status code: Message:
[Container] 2017/07/21 20:17:17 Preparing to copy artifacts
[Container] 2017/07/21 20:17:17 No artifact files specified
```

10. Click [View entire log](#) to go to CloudWatch logs and view detailed information about every step in the build process. Examining and retaining these logs can be useful for troubleshooting purposes:

Time (UTC +00:00)	Message
2017-07-21	
▶ 20:17:09	Step 10 : CMD npm start
▶ 20:17:09	---> Running in bb4ff8fbf96d
▶ 20:17:09	---> 9e5ff243296d
▶ 20:17:09	Removing intermediate container bb4ff8fbf96d
▶ 20:17:09	Successfully built 9e5ff243296d
▶ 20:17:09	
▼ 20:17:09	[Container] 2017/07/21 20:17:08 Running command docker tag ca-container-registry:testblue \$AWS_ACCOUNT_ID.dkr.ecr.\$AWS_REGION.amazonaws.com/ca-container-registry:testblue
[Container]	2017/07/21 20:17:08 Running command docker tag ca-container-registry:testblue \$AWS_ACCOUNT_ID.dkr.ecr.\$AWS_REGION.amazonaws.com/ca-container-registry:testblue
▶ 20:17:09	
▼ 20:17:09	[Container] 2017/07/21 20:17:08 Phase complete: BUILD Success: true
[Container]	2017/07/21 20:17:08 Phase complete: BUILD Success: true
▼ 20:17:09	[Container] 2017/07/21 20:17:08 Phase context status code: Message:
[Container]	2017/07/21 20:17:08 Phase context status code: Message:
▼ 20:17:09	[Container] 2017/07/21 20:17:08 Entering phase POST_BUILD
[Container]	2017/07/21 20:17:08 Entering phase POST_BUILD
▼ 20:17:09	[Container] 2017/07/21 20:17:08 Running command echo Build completed on `date`
[Container]	2017/07/21 20:17:08 Running command echo Build completed on `date`
▶ 20:17:09	Build completed on Fri Jul 21 20:17:08 UTC 2017
▶ 20:17:09	
▶ 20:17:09	[Container] 2017/07/21 20:17:08 Running command echo Pushing the Docker image...
▶ 20:17:09	Pushing the Docker image...
▶ 20:17:09	
▼ 20:17:09	[Container] 2017/07/21 20:17:08 Running command docker push \$AWS_ACCOUNT_ID.dkr.ecr.\$AWS_REGION.amazonaws.com/ca-container-registry:testblue
[Container]	2017/07/21 20:17:08 Running command docker push \$AWS_ACCOUNT_ID.dkr.ecr.\$AWS_REGION.amazonaws.com/ca-container-registry:testblue

11. Return to CodeBuild > Build projects and click Create project:

Build projects

Review your build projects and start new builds.

Start build Create project Actions

	Project	Repository
●	ecslab-blue-project	arn:aws:s3:::cloudacademylab-caecslab-9bfvbfqvqg8l/ecslabblue.zip

< Viewing 1 item

For convenience's sake and for the purposes of this lab demonstration, you will create both the blue and green application container images at the same time. In a real environment your releases may be weeks or months apart, or several times a day, depending on your development and deployment approach.

12. The steps and details for creating the green project are almost identical to those for the blue project. Enter the following details in the Configuration section:

- Project name: *ecslab-green-project*
- Source provider: Amazon S3
- Bucket: cloudacademylab-caecslab-XXXXXXXXXX (The final values are randomized to prevent name conflicts in Amazon S3.)
- S3 object key: *ecslabgreen.zip*
- Environment image: Use an image managed by AWS CodeBuild
- Operating system: Ubuntu
- Runtime: Docker
- Version: aws/codebuild/docker:1.12.1
- Privileged: Yes

Note: The Docker version listed is the latest at lab creation. In the future, the version will update. Choose the latest version if the version listed is unavailable.

- Build specification: Use the *buildspec.yml* in the source code root directory
- Artifacts type: No artifacts
- Service role: Choose an existing service role from your account
- Role name: CodeBuildServiceRole (Remove the check from Allow AWS CodeBuild to modify this service role...)

13. One important difference for your second CodeBuild project is the ability to enter advanced settings in the initial project creation screen. Expand the Show advanced settings section. Add the *AWS_ACCOUNT_ID* in the Environment variables section. Remember to delete the hyphens from the AWS account number. Click Continue:

▼ Hide advanced settings

Timeout hours minutes i

Encryption key i

arn:aws:kms:<region-ID>:<account-ID>/key/<key-ID>

Privileged Enable this flag if you want to build Docker images or want your builds to get elevated privileges.

Compute type 3 GB memory, 2 vCPU
 7 GB memory, 4 vCPU
 15 GB memory, 8 vCPU

Environment variables Add environment variables (custom file paths, AWS resource IDs) that you want AWS CodeBuild to use.

Name	Value
AWS_ACCOUNT_ID	358259402422

14. Review the project details in the Review section and click Save and Build.

15. Click Start build.

16. Wait approximately one minute for the Status field in the Build section to show Succeeded. Examine the build log and details, if you wish. All of the build process information is very similar to the blue project, but the new green container has different unique identifiers.

17. Return to CodeBuild > Build projects and view your two projects. You can initiate a new build and create new containers at any time by selecting a project and clicking Start build. This is not necessary for this Lab, but a useful tool in case you upload a new version of the application to one of your source locations:

Build projects ?

Review your build projects and start new builds.

Actions	
Start build	Create project
	Actions ▾
	↻

Project		Repository
<input checked="" type="radio"/>	ecslab-green-project	arn:aws:s3:::cloudacademylab-caecslab-48p2fknvydz7/ecslabgreen.zip
<input type="radio"/>	ecslab-blue-project	arn:aws:s3:::cloudacademylab-caecslab-48p2fknvydz7/ecslabblue.zip

So where do your container images go? You are building Docker container images, but where can you view the output of your build operations?

18. Return to Services > EC2 Container Service and click Repositories. Click on ca-container-registry to view your Docker container images:

Amazon ECS

Clusters

Task Definitions

Repositories

< All repositories : ca-container-registry

Repository ARN arn:aws:ecr:us-west-2:358259402422:repository/ca-container-registry

Repository URI 358259402422.dkr.ecr.us-west-2.amazonaws.com/ca-container-registry

[View Push Commands](#)

Images [Permissions](#)

Amazon ECR limits the number of images to 1,000 per repository. [Request a limit increase.](#)

Image sizes may appear compressed. [Learn more](#)

[Delete](#) Last updated on July 21, 2017 1:54:33 PM (1m ago) [refresh](#)

Filter in this page	Tag Status:	All	1-2	Page size	100
<input type="checkbox"/> Image tags	Digest	Size (MIB)	Pushed at		
<input type="checkbox"/> testgreen	view all sha256:22d2b2cbcb431cac79be121f97f1820af...	21.01	2017-07-21 13:33:27 -0700		
<input type="checkbox"/> testblue	view all sha256:bfebadb6e7ebaeea21d2859d2c1ee8ba...	21.01	2017-07-21 13:17:17 -0700		

The application buildspec.yml files tag the container images with testgreen or testblue, depending on the application they contain.

Summary

In this Lab Step you used CodeBuild to create two Docker container images with two different applications. You modified some variables and examined the build phases and logs. You also viewed your Docker images and details in your ECR repository.

Creating the ECS Cluster

Introduction

An ECS cluster is a logical grouping of container instances where you can define task definitions to execute tasks. A task is the instantiation of a task definition on a container instance within an ECS cluster. You can customize several features of a cluster such as EC2 instance type, storage allocation, and networking specifics. In this Lab Step you will create an ECS cluster using a single t2.micro EC2 instance and some pre-configured networking resources.

Instructions

1. Click on Clusters and click Create Cluster:

The screenshot shows the AWS CloudWatch Metrics interface. At the top, there's a navigation bar with icons for Services, Resource Groups, and a star icon. Below the navigation bar, the left sidebar lists 'Amazon ECS' services: 'Clusters' (which is selected and highlighted in orange), 'Task Definitions', and 'Repositories'. The main content area has a title 'Clusters' and a description: 'An Amazon ECS cluster is a regional grouping of one or more container instances running on Amazon EC2 instance type.' It includes a blue 'Create Cluster' button and a 'View' dropdown menu with 'list' and 'card' options. There are two large, empty rectangular boxes representing where cluster details would be listed.

2. In the Create Cluster section enter the following details. Leave other settings as the defaults. Click Create:

- Cluster name: *ecslab-cluster*
- EC2 instance type: t2.micro

Warning! The lab environment is restricted to only allow the t2.micro instance type. You must select a t2.micro or the cluster will not launch.

- VPC: Choose the VPC with a name ending in clouddacademylab. The Lab environment creates this VPC specifically for this Lab.
- Subnets: Select all available subnets.
- Security group: Select the Security Group (SG) whose name contains clouddacademylab-UsingECSLabSecurity...
- Container instance IAM role: ecsInstanceRole

Create Cluster

When you run tasks using Amazon ECS, you place them on a cluster, which is a logical grouping of EC2 instances. This wizard will guide you through the process to [create](#) container instances that your tasks can be placed on, the security group for your container instances to use, and the IAM role to associate with your container instance.

Cluster name* 

Create an empty cluster

Instance configuration

Provisioning Model On-Demand Instance
With On-Demand Instances, you pay for compute capacity by the hour, with no long-term commitments or upfront payments.

Spot
Amazon EC2 Spot Instances allow you to bid on spare Amazon EC2 computing capacity for up to 90% off the On-Demand price. [Learn more](#)

EC2 instance type* 

Number of instances* 

EC2 Ami Id* amzn-ami-2017.03.d-amazon-ecs-optimized [ami-57d9cd2e] 

EBS storage (GiB)* 

Key pair   
You will not be able to SSH into your EC2 instances without a key pair. You can create a new key pair in the [EC2 console](#).

Networking

Configure the VPC for your container instances to use. A VPC is an isolated portion of the AWS cloud populated by AWS objects, such as Amazon EC2 instances. You can choose an existing VPC, or create a new one with this wizard.

VPC   
Check the structure for [vpc-9f0fd9f9](#) in the Amazon EC2 console.

Subnets   
   
 

Security group   
Rules for [sg-bc2e9cc6](#) in the EC2 Console.

Container instance IAM role

The Amazon ECS container agent makes calls to the Amazon ECS API actions on your behalf, so container instances that run the agent require the `ecsInstanceRole` IAM policy and role for the service to know that the agent belongs to you. If you do not have the `ecsInstanceRole` already, we can create one for you.

Container instance IAM role 

*Required  

3. Observe the cluster creation process on the Launch status screen. Wait until all three steps complete and click View Cluster:

Launch status

Your container instances are launching, and it may take a few minutes until they are in the running state and ready to access. Usage hours on your new container instances will begin once they are in the running state.

Back

View Cluster

ECS status - 3 of 3 complete **ecslab-cluster**

✓ ECS cluster

ECS Cluster **ecslab-cluster** successfully created

✓ ECS Instance IAM Policy

IAM Policy for the role **cloudacademy-lab-InstanceProfileForECs-UI80I86309NB** successfully attached

✓ CloudFormation Stack

CloudFormation stack **EC2ContainerService-ecslab-cluster** and its resources successfully created

Cluster Resources

Instance type	t2.micro
Desired number of instances	1
Key pair	
ECS AMI ID	ami-57d9cd2e
VPC	vpc-9f0fd9f9
Subnets	subnet-b47393d2, subnet-9fd9e3d6
VPC Availability Zones	us-west-2b, us-west-2a, us-west-2c
Security group	sg-bc2e9cc6
Launch configuration	EC2ContainerService-ecslab-cluster-EcsInstanceLc-13VUJ6S1USLHT
Auto Scaling group	EC2ContainerService-ecslab-cluster-EcsInstanceAsg-109J399S7R3P5

The ECS cluster creation process is a fairly complex operation that utilizes the following Amazon services:

- Elastic Compute Cloud (EC2) - The cluster is powered by EC2 instances and networking features.
- Identity and Access Management - The EC2 instances require roles with defined policies to securely interact with other services.
- CloudFormation - CloudFormation is the mechanism that deploys and manages your cluster as configured.
- Auto Scaling - Though not used in this Lab, it is important to note you can configure Auto Scaling rules and conditions for your cluster.

Summary

In this Lab Step you created an ECS cluster. Currently, the cluster does nothing, but it is a vital logical grouping of resources for the tasks and services you will soon create. This step serves as a bridge to connect the services involved with the source code compilation and those responsible for the final application deployment. In the next Lab Step you will create task definitions for your cluster.

Creating the Task Definitions

Introduction

Task definitions are required to run Docker containers in ECS. Task definitions tell the services which Docker images to use for the container instances, what kind of resources to allocate, network specifics, and other details. In this Lab Step you will create two task definitions, one for your blue application and one for your green application. This is a somewhat simplified demonstration. In reality, you can define multiple containers and data volumes in a task definition, use variables, modify source files, or a number of other approaches to maintaining an application.

Instructions

1. Select Task Definitions. Click Create new Task Definition:

The screenshot shows the Amazon ECS console interface. On the left, there is a navigation sidebar with the following options: 'Amazon ECS' (selected), 'Clusters' (disabled), 'Task Definitions' (selected and highlighted in orange), and 'Repositories'. The main content area is titled 'Task Definitions' and contains the following elements:

- A sub-instruction: 'Task definitions specify the container information for your application, such as how many CPU and memory resources to allocate, and which Docker image to use.'
- Three buttons at the top right: 'Create new Task Definition' (blue), 'Create new revision' (gray), and 'Actions ▾'.
- A status indicator: 'Status: ACTIVE INACTIVE'.
- A search/filter bar with the placeholder 'Filter in this page'.
- A table header row with columns: 'Task Definition' (with a checkbox icon) and 'Actions' (with a dropdown arrow).
- An empty table body below the header.

2. Enter a Task Definition Name of `ecslab-blue-taskdef`. Leave the other settings as default. Click Add container:

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 u

Task Definition Name*	ecslab-blue-taskdef	
Task Role	Select a role...	
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	

Constraint

Constraints allow you to filter the instances used for your placement strategies using built-in or custom attributes. The scheduler first filters the instances that match th

Type	Expression
+ Add constraint	

Container Definitions

[Add container](#)

3. Enter the following information in the Standard section of the Add container page. Leave other settings as default.

- Container name: *ecslab-blue-container*
- Image: <YOURREPOSITORYURL>:*testblue*, for example - 339748811106.dkr.ecr.us-west-2.amazonaws.com/ca-container-registry:
testblue
- Memory Limits (MB): Hard limit *128*
- Port mappings: Host port = *0*, Container port = *8081*

Scroll down and click Add:

Add container

▼ Standard

Container name* i

Image* i

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

Memory Limits (MB)* Hard limit i

+ 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 i

0	8081	tcp	x
---	------	-----	---

+ Add port mapping

▼ Advanced container configuration

The Host port 0, Container port 8081 tells the EC2 instance hosting the containers to look for traffic on any port an application is using. Port 8081 is the traffic port exposed by the Docker container.

4. When you return to the Create a Task Definition page, note that you now have an entry in the Container Definitions section. Click Create:

Container Definitions				
Add container ?				
Container Name	Image	Hard/Soft memory limits (MB)	Essential	x
ecslab-blue-container	358259402422.dkr.ecr.us-west-2.amazonaws.co...	128/--	true	x
Volumes				
Name	Source Path	No results		
+ Add volume				
Configure via JSON				

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

Container definitions are a set of parameters passed to the Docker daemon on a container instance. In this case you have specified the name, image, and memory limit parameters.

5. Verify you see the Created Task Definition successfully message and can now view details about the `ecslab-blue-taskdef` task definition:

The screenshot shows the AWS Task Definitions console. At the top, a green success message box says "Created Task Definition successfully". Below it, the navigation path is "Task Definitions > `ecslab-blue-taskdef` > 1". The main title is "Task Definition: `ecslab-blue-taskdef:1`". A sub-header says "View detailed information for your task definition. To modify the task definition, you need to create a new revision and then make the revision active." Below this, there are two buttons: "Create new revision" (blue) and "Actions" (dropdown). Under "Builder", the "JSON" tab is selected. The task definition details are as follows:

Task Definition Name	<code>ecslab-blue-taskdef</code>
Task Role	<code>None</code> 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	<code>Bridge</code>

6. Return to Task Definitions and click Create new Task Definition again. As with the Docker container images, you must create another Task Definition for your green application. Enter a Task Definition Name of `ecslab-green-taskdef`. Leave the other settings as default. Click Add container.

7. Enter the following information in the Standard section of the Add container page. Leave other settings as default:

- Container name: `ecslab-green-container`
- Image: <YOURREPOSITORYURL>:`testgreen`, for example - `339748811106.dkr.ecr.us-west-2.amazonaws.com/ca-container-registry:testgreen`
- Memory Limits (MB): Hard limit `128`
- Port mappings: Host port = `0`, Container port = `8081`

Click Add.

8. Verify you see your `ecslab-green-container` in the Container Definitions section. Click Create.

9. Verify you see the Created Task Definition successfully message and can now view details about the `ecslab-green-taskdef` task definition.

Summary

In this Lab Step you created two task definitions for your applications. You specified information about the Docker container images, resources, and networking details to use.

Creating the Target Group and Load Balancer

Introduction

Using multiple containers is a good practice to avoid a single point of failure and maintain availability. However, making the front-end access point of an application reliant on fixed references to dynamic and scalable cloud resources is not the best idea. In this Lab Step you will create a target group and load balancer to consolidate access to your resources in one place with one Domain Name Server (DNS) entry.

Instructions

1. Navigate to Services > Compute > EC2, or search for *EC2* in the service search box:

The screenshot shows the AWS Services navigation bar with 'Services' selected. Below it, a dropdown menu for 'Compute' is open, showing options like EC2, EC2 Container Service, Lightsail, Elastic Beanstalk, Lambda, and Batch. A red arrow points from the 'Compute' heading towards the 'EC2' option. To the left of the dropdown, there's a sidebar with links to History, EC2 Container Service, CodeBuild, CloudWatch, Console Home, S3, and EC2.

2. Select LOAD BALANCING > Target Groups from the EC2 Dashboard column.

3. Click Create target group. A target group is a group of resources, such as container or EC2 instances, registered and grouped together for reference by an Application Load Balancer (ALB).

4. Enter the following information in the Create target group window. (Change ? to your number) Leave other settings as default. Click Create:

- Target group name: *devops?-targetgroup*
- VPC: Select the VPC with cloudacademylab in the name.
- Path: */devops?-api/*

Create target group

X

Your load balancer routes requests to the targets in a target group using the protocol and port that you specify, and performs health checks on the targets using the health check settings that you specify.

Target group name (i)

Protocol (i)

Port (i)

VPC (i)

Health check settings

Protocol (i)

Path (i)

► Advanced health check settings

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

The /api/ path reference is not an essential part of creating every load balancer. It is a specific requirement for your Lab example application.

5. Verify you receive a Successfully created target group message and click Close.

6. Select LOAD BALANCING > Load Balancers from the EC2 Dashboard column.

7. Click Create Load Balancer.

8. Leave Application Load Balancer selected and click Continue. An ALB is a type of load balancer that functions at the application layer, the seventh layer of the Open Systems Interconnection (OSI) model. It is similar to a classic Elastic Load Balancer (ELB) in that it distributes traffic among targets using a routing algorithm. However, ALBs have the ability to examine content and route traffic accordingly. The essential ALB feature for this Lab is support for containerized applications. ALBs have the ability to select an unused port when scheduling a task and registering a task with a target group using this port.

9. On the Step 1: Configure Load Balancer page, enter the following information. Leave the other settings as default. Click Next: Configure Security Settings:

- Name: **ecslab-alb**
- VPC: Select the VPC with **cloudacademylab** in the name. Select all available Availability Zones.

Step 1: Configure Load Balancer

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	<input type="text" value="ecslab-alb"/>
Scheme	<input checked="" type="radio"/> internet-facing <input type="radio"/> internal
IP address type	<input type="text" value="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	

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-9f0fd9f9 (10.10.10.0/24) cloudacademylab		
<input checked="" type="checkbox"/> Availability Zone	Subnet ID	Subnet IPv4 CIDR	Name
<input checked="" type="checkbox"/> us-west-2a	subnet-b47393d2	10.10.10.0/25	
<input checked="" type="checkbox"/> us-west-2b	subnet-9fd9e3d6	10.10.10.128/25	

[Cancel](#) [Next: Configure Security Settings](#)

10. Ignore the warning about a secure listener. It is not important for this lab exercise. Click Next: Configure Security Groups.

11. Choose Select an existing security group and select the SG with cloudacademylab in the name. Click Next: Configure Routing:

Step 3: Configure Security Groups

A security group is a set of firewall rules that control the traffic to your load balancer. On this page, you can add rules to allow specific traffic to reach your load balancer. First, decide whether to create a new security group or select an existing one.

Assign a security group: Create a new security group
 Select an existing security group

Security Group ID	Name	Description	Actions
<input checked="" type="checkbox"/> sg-bc2e9cc6	cloudacademylab-UsingECSLabSecurityGroup-1LE8UU4Z0RPBP	Using ECS Lab	Copy to new
<input type="checkbox"/> sg-032e9c79	default	default VPC security group	Copy to new

[Cancel](#) [Previous](#) [Next: Configure Routing](#)

12. Enter the following information in the Target group and Health checks sections. Leave other settings as default. Click Next: Register Targets:

- Target group: Existing target group
- Name: ecslab-targetgroup
- Path: /api/

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 on the targets using these health check settings. Note that each target group can be associated with only one load balancer.

Target group

Target group	<input type="button" value="Existing target group"/>
Name	<input type="text" value="ecslab-targetgroup"/>
Protocol	<input type="button" value="HTTP"/>
Port	<input type="text" value="80"/>

Health checks

Protocol	<input type="button" value="HTTP"/>
Path	<input type="text" value="/api/"/>

Advanced health check settings

Port	<input checked="" type="radio"/> traffic port <input type="radio"/> override
Healthy threshold	<input type="text" value="5"/>
Unhealthy threshold	<input type="text" value="2"/>
Timeout	<input type="text" value="5"/> seconds
Interval	<input type="text" value="30"/> seconds
Success codes	<input type="text" value="200"/>

[Cancel](#) [Previous](#) [Next: Register Targets](#)

This Lab shows the Advanced health check settings for informational purposes. You can specify a port for routing purposes and control the health check thresholds. The thresholds determine how quickly a target registers as healthy or unhealthy.

13. There are no targets available in Step 5: Register Targets. Click Next: Review. As your ECS services launch tasks and container instances, they will dynamically register with the target group and load balancer.

14. Review the ALB information in Step 6: Review and click Create.

15. Verify you receive a Successfully created load balancer message. Click Close.

Summary

In this Lab Step you created and configured a target group and ALB. You learned about ALBs and how their support for containerized applications makes them essential for providing a single point of access for dynamic resources.

Creating the ECS Services

Introduction

An ECS service is a mechanism that allows ECS to run and maintain a specified number of instances of a task definition. If any tasks or container instances should fail or stop, the ECS service scheduler launches another instance to replace it. This is similar to Auto Scaling in that it maintains a desired number of instances, but it does not scale instances up or down based on CloudWatch alarms or other Auto Scaling mechanisms. Services behind a load balancer provide a relatively seamless way to maintain a certain amount of resources while keeping a single application reference point. In this Lab Step you will create two services, one for your blue application and one for the green application.

Instructions

1. Return to Services > EC2 Container Service > Clusters and click on the ecslab-cluster:

The screenshot shows the 'Clusters' section of the Amazon ECS console. On the left, there's a navigation sidebar with 'Amazon ECS' at the top, followed by 'Clusters' (which is highlighted in orange), 'Task Definitions', and 'Repositories'. The main content area has a title 'Clusters' with a subtitle explaining what an ECS cluster is. Below that is a 'Create Cluster' button. The main view shows a single cluster card for 'ecslab-cluster'. The card displays metrics: 0 Services, 0 Running tasks, 0 Pending tasks, 0.00% CPUUtilization, 0.00% MemoryUtilization, and 1 Container instances. There are 'View' buttons for 'list' and 'card', and a 'view all' link. Navigation arrows are at the bottom.

2. Click the Create button on the Services tab:

The screenshot shows the 'Cluster : ecslab-cluster' page. At the top, it says 'Status ACTIVE' and has a 'Delete Cluster' button. Below that is a summary of cluster resources: Registered container instances (1), Pending tasks count (0), and Running tasks count (0). A navigation bar at the top of the main content area includes tabs for 'Services', 'Tasks', 'ECS Instances', 'Metrics', and 'Scheduled Tasks'. Under the 'Services' tab, there are buttons for 'Create', 'Update', and 'Delete'. A 'Filter in this page' input field is also present. The main table has columns for Service Name, Status, Task Definition, Desired tasks, and Running tasks. A message 'No results' is shown below the table. At the top right of the main content area, there's a note 'Last updated on July 21, 2017 2:16:21 PM (0m ago)' and some navigation icons.

3. Enter the following information on the Create Service page. Leave the other settings as default:

- Task Definition: `ecslab-blue-taskdef:1`
- Service name: `ecslab-blue-service`
- Number of tasks: `2`

Create 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.

Task Definition	ecslab-blue-taskdef:1	
Cluster	ecslab-cluster	
Service name	ecslab-blue-service	
Number of tasks	2	
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)

Optional configurations

Elastic load balancing

Connect an Elastic Load Balancing load balancer to distribute traffic to each task in your service from a single DNS endpoint.

[Configure ELB](#)

Service Auto scaling

Automatically adjust your service's desired count up and down in response to CloudWatch alarms.

[Configure Service Auto Scaling](#)

The Number of tasks determines the number of container instances the service launches and maintains. In this case you specify two tasks to launch two container instances. Even though for the sake of simplicity and to minimize costs this Lab uses a single EC2 host instance, you can alleviate the risk of having a single point of failure in the number of container instances you launch. The Minimum healthy percent and Maximum percent parameters help determine deployment strategies. The maximum parameter enables you to define the deployment batch size. With the value of 200%, the scheduler may start two new tasks before stopping the two older tasks. This can help you maintain consistent active capacity, but you may want to reduce the value to control costs if your environment is large and expensive to run. The minimum healthy parameter controls the lower limit of your service's tasks and is a way to maintain minimum capacity.

4. Click Configure ELB in the Optional configurations section. Choose the following options in the Elastic Load Balancing (optional) section. Click Add to ELB:

- ELB type: Application Load Balancer
- Select IAM role for service: ecsServiceRole
- ELB Name: ecslab-alb
- Container to load balance: ecslab-blue-container:0:8081

Elastic Load Balancing (optional)

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](#). When you are finished configuring your load balancer, choose **Save** to continue.

ELB type: <input type="radio"/> None Your service will not use a load balancer.	<input checked="" type="radio"/> 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.	<input type="radio"/> Classic Load Balancer Requires static host port mappings (only one task allowed per container instance); rule-based routing and paths are not supported.
---	---	---

Select IAM role for service (i)

ELB Name (i)

Container to load balance

Select a Container (i) Add to ELB

Back Save

5. In the Container to load balance section, select `ecslab-targetgroup` from the Target group name drop-down list. This automatically populates the other settings with the target group information you entered in an earlier Lab Step. Click Save:

Container to load balance

ecslab-blue-container : 8081 [Remove](#) X

Listener port Enter a listener port (i)

Listener protocol (i)

Target group name (i)

Target group protocol (i)

Path pattern (i)

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 (i)

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

6. Click Configure Service Auto Scaling. Though not used in this Lab, it is useful to examine the Auto Scaling options and understand how you can use Auto Scaling to maintain a target service level in your cluster. You can specify a minimum, maximum, and desired number of tasks:

Service Auto Scaling (optional)

Automatically adjust your service's desired count up and down within a specified range in response to CloudWatch alarms. You can modify your Service Auto Scaling configuration at any time to meet the needs of your application.

- Service Auto Scaling**
- Do not adjust the service's desired count
 - Configure Service Auto Scaling to adjust your service's desired count

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 You are giving permission to EC2 Container Service to create and use ecsAutoscaleRole. ⓘ

You can also set task scaling policies based on CloudWatch alarms. This allows your cluster to respond to increases or decreases in activity or traffic.

Automatic task scaling policies

Scale out (increase desired count) Delete X

Policy name

- Execute policy when**
- Create new Alarm
 - Use an existing Alarm

This wizard uses ECS metrics for new alarms. To scale your service with other metrics, create your alarms in the [CloudWatch console](#), and then refresh the alarm list here.

Alarm name

ECS service metric

Alarm threshold Average of CPUUtilization \geq 80
for 1 consecutive periods of 5 minu...

Save

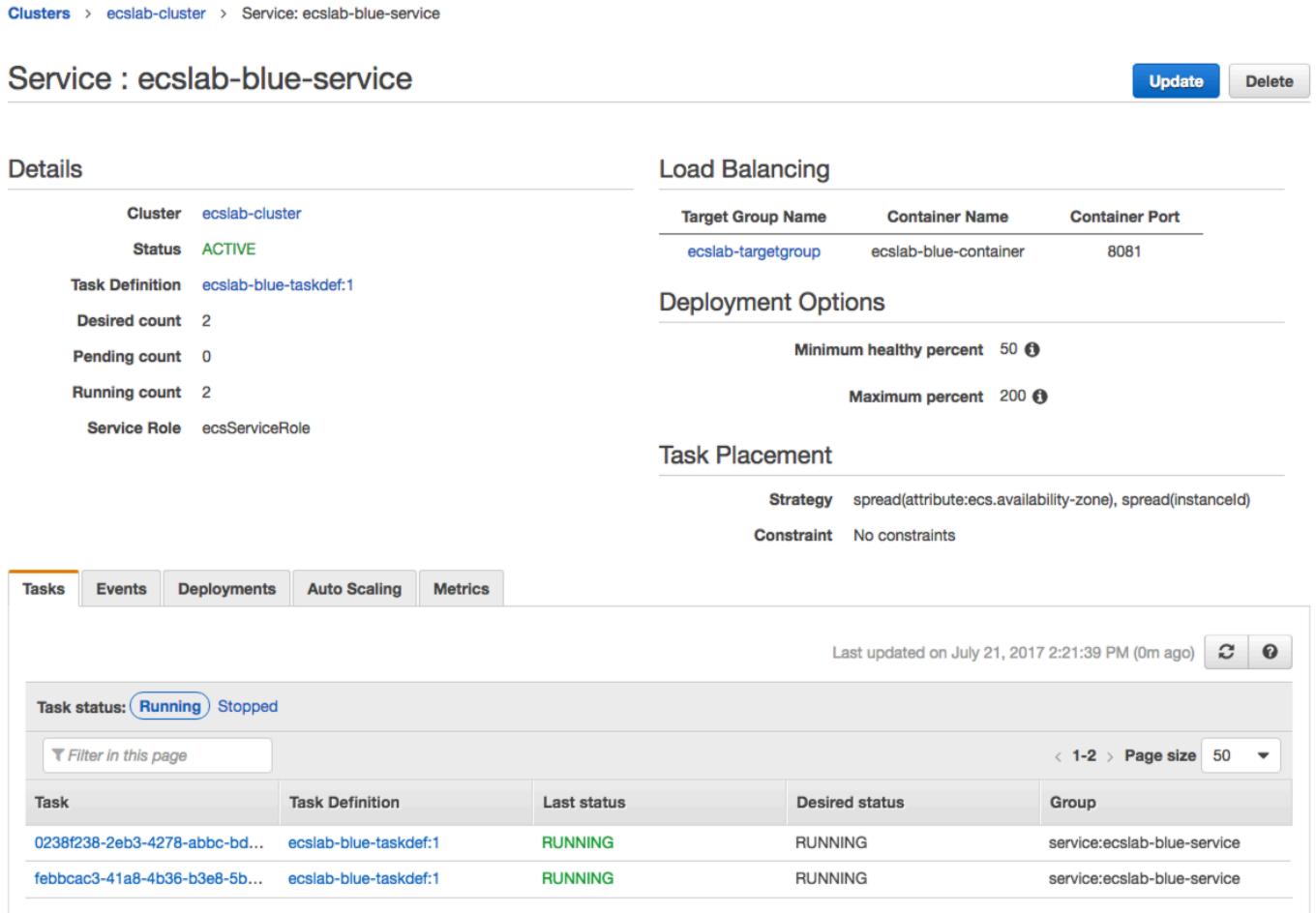
Scaling action Please select an alarm

Scale in (decrease desired count) Delete X

7. After exploring the scaling options, click Cancel to return to the Create Service page. Click Create Service.

8. After the Create Load Balancer and Create Service tasks complete, click View Service.

9. The `ecslab-blue-service` starts two tasks and begins the process of deploying them to the ECS cluster. Within approximately two minutes the service Status will change to ACTIVE and the Tasks tab will show two running tasks, as shown in the following image:



Clusters > `ecslab-cluster` > Service: `ecslab-blue-service`

Service : `ecslab-blue-service`

Details

Cluster	<code>ecslab-cluster</code>
Status	ACTIVE
Task Definition	<code>ecslab-blue-taskdef:1</code>
Desired count	2
Pending count	0
Running count	2
Service Role	<code>ecsServiceRole</code>

Load Balancing

Target Group Name	Container Name	Container Port
<code>ecslab-targetgroup</code>	<code>ecslab-blue-container</code>	8081

Deployment Options

Minimum healthy percent	50
Maximum percent	200

Task Placement

Strategy	spread(attribute:ecs.availability-zone), spread(instanceld)
Constraint	No constraints

Tasks Events Deployments Auto Scaling Metrics

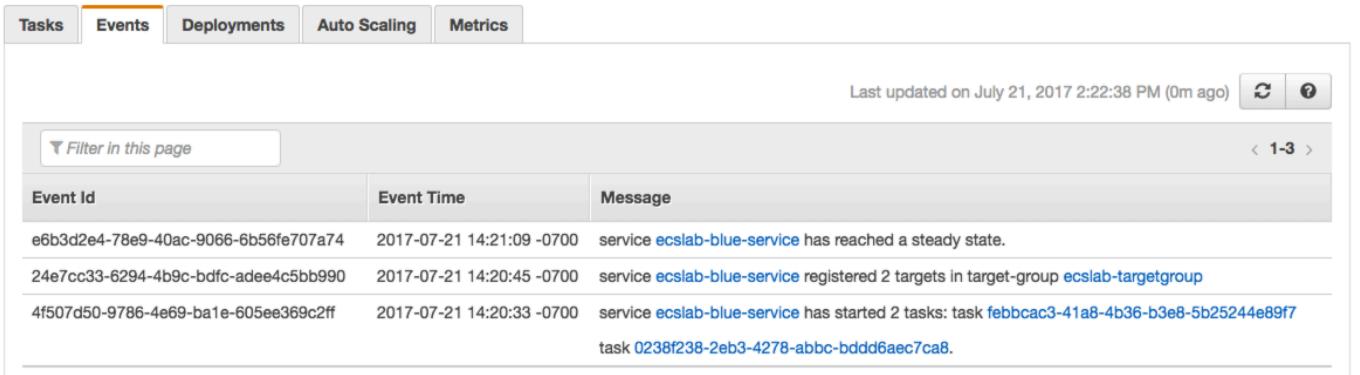
Last updated on July 21, 2017 2:21:39 PM (0m ago)

Task status: Running Stopped

Filter in this page < 1-2 > Page size 50

Task	Task Definition	Last status	Desired status	Group
<code>0238f238-2eb3-4278-abbc-bd...</code>	<code>ecslab-blue-taskdef:1</code>	RUNNING	RUNNING	service: <code>ecslab-blue-service</code>
<code>febbcac3-41a8-4b36-b3e8-5b...</code>	<code>ecslab-blue-taskdef:1</code>	RUNNING	RUNNING	service: <code>ecslab-blue-service</code>

10. Click the Events tab to view the service events:



Tasks Events Deployments Auto Scaling Metrics

Last updated on July 21, 2017 2:22:38 PM (0m ago)

Filter in this page < 1-3 >

Event Id	Event Time	Message
<code>e6b3d2e4-78e9-40ac-9066-6b56fe707a74</code>	2017-07-21 14:21:09 -0700	service <code>ecslab-blue-service</code> has reached a steady state.
<code>24e7cc33-6294-4b9c-bdfc-addee4c5bb990</code>	2017-07-21 14:20:45 -0700	service <code>ecslab-blue-service</code> registered 2 targets in target-group <code>ecslab-targetgroup</code>
<code>4f507d50-9786-4e69-ba1e-605ee369c2ff</code>	2017-07-21 14:20:33 -0700	service <code>ecslab-blue-service</code> has started 2 tasks: task <code>febbcac3-41a8-4b36-b3e8-5b25244e89f7</code> task <code>0238f238-2eb3-4278-abbc-bddd6aec7ca8</code> .

Note how the service registered two targets with the application load balancer target group. Earlier you created a load balancer but did not register any target instances. Instead, the service instantiates container instances based on the number of tasks you specify. You added the load balancer and target group to the service, so any new container instances the service creates automatically register as a target in the target group.

11. Return to Clusters > *ecslab-cluster*. Click the Create button on the Services tab. You are going to create the green service now, but you will not give it any tasks at this moment.

12. Enter the following information on the Create Service page. Leave the other settings as default.

- Task Definition: *ecslab-green-taskdef:1*
- Service name: *ecslab-green-service*
- Number of tasks: *0*

Note: You can create services with zero tasks. They will not launch any container instances upon creation. This allows you to prepare for future operations before your deployment is fully ready.

13. Click Configure ELB in the Optional configurations section. Choose the following options in the Elastic Load Balancing (optional) section. Click Add to ELB:

- ELB type: Application Load Balancer
- Select IAM role for service: *ecsServiceRole*
- ELB Name: *ecslab-alb*
- Container to load balance: *ecslab-green-container:0:8081*

14. In the Container to load balance section, select *ecslab-targetgroup* from the Target group name drop-down list. Click Save.

15. Click Create Service. After the Create Load Balancer and Create Service tasks complete, click View Service.

Summary

In this Lab Step you created two services for your blue and green applications. You learned how these services control the desired capacity and how you can configure them to scale, if necessary. You also learned more about how they interact with the target group and load balancer you created earlier. You started two tasks for the blue application upon service creation. These tasks launched and registered two container instances.

Viewing Instances and Application Message

Introduction

Now that you have put all of the pieces in place to build and store Docker images, dynamically register container instances, and maintain a target capacity, it is time to learn more about the interdependent service actions and see some results. In this Lab Step you will view the resources launched by ECS and access the blue application launched when you created the blue service.

Instructions

1. Navigate to Services > Compute > EC2:

The screenshot shows the AWS Services menu with the 'Compute' section expanded. A red arrow points from the 'Compute' heading to the 'EC2' link. Other services listed under Compute include Lightsail, Elastic Beanstalk, Lambda, and Batch.

Services ▾ Resource Groups ▾

History

Find a service by name or feat

- EC2 Container Service
- CodeBuild
- CloudWatch
- Console Home
- S3
- EC2

Compute

- EC2
- EC2 Container Service
- Lightsail
- Elastic Beanstalk
- Lambda
- Batch

2. Select INSTANCES > Instances from the EC2 Dashboard column. The ECS blue service automatically launched an EC2 instance to host the ECS container instances running your tasks:

The screenshot shows the AWS EC2 Instances dashboard. The left sidebar is collapsed. The main area displays a table of instances. One instance is listed: 'ECS Instance - EC2ContainerService-ecslab-cluster' with Instance ID i-0f8f42c360f32a63c, Instance Type t2.micro, Availability Zone us-west-2b, Status running, and 2/2 checks passed.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks
ECS Instance - EC2ContainerService-ecslab-cluster	i-0f8f42c360f32a63c	t2.micro	us-west-2b	running	2/2 checks ...

3. Select LOAD BALANCING > Target Groups from the EC2 Dashboard column.

4. Click the Targets tab and view the Registered targets section. Notice that you have two healthy targets despite having only one running EC2 instance. ECS deployed two container instances on the EC2 host instance. Each container instance dynamically registered with the target group on an assigned port in the ephemeral port range:

Target group: **eclslab-targetgroup**

Description Targets Health checks Monitoring Tags

The load balancer starts routing requests to a newly registered target as soon as the registration process completes and the target passes the initial health checks. If demand on your targets increases, you can register additional targets. If demand on your targets decreases, you can deregister targets.

Edit

Registered targets

Instance ID	Name	Port	Availability Zone	Status
i-0f8f42c360f32a63c	ECS Instance - EC2ContainerService-eclslab-cluster	32769	us-west-2b	healthy ⓘ
i-0f8f42c360f32a63c	ECS Instance - EC2ContainerService-eclslab-cluster	32768	us-west-2b	healthy ⓘ

Availability Zones

Availability Zone	Target count	Healthy?
us-west-2b	2	Yes

5. Select LOAD BALANCING > Load Balancers from the EC2 Dashboard column.

6. Copy the DNS name from the Description tab and paste it into a browser window or tab. Append `/api/` to the DNS value to form a Uniform Resource Locator (URL) that resembles this example:

<http://eclslab-alb-793477761.us-west-2.elb.amazonaws.com/api/>

Load balancer: **eclslab-alb**

Description Listeners Monitoring Tags

Basic Configuration

Name:	eclslab-alb ⓘ	Creation time:	July 21, 2017 at 2:15:31 PM UTC-7
ARN:	arn:aws:elasticloadbalancing:us-west-2:358259402422:loadbalancer/app/eclslab-alb/69747778856f43d6	Hosted zone:	Z1H1FL5HABSF5
DNS name:	eclslab-alb-793477761.us-west-2.elb.amazonaws.com (A Record)	State:	active
Scheme:	internet-facing	VPC:	vpc-9fd9f9
Type:	application	IP address type:	ipv4
Availability Zones:	subnet-9fd9e3d6 - us-west-2b, subnet-b47393d2 - us-west-2a	AWS WAF Web ACL:	
Edit availability zones			

Security

Security groups:	sg-bc2e9cc6 , cloudacademylab-UsingECSLabSecurityGroup-1LE8UU4Z0RPBP • Using ECS Lab
Edit security groups	

The load balancer DNS name provides a single point of access for your users/application for all instances behind the load balancer, even if those instances fail or are replaced.

7. Your browser will display the following message, though the format or appearance may vary slightly depending on the browser you use. This is a simple JavaScript Object Notation (JSON) message delivered by the application running on the container instances:

```
▼ {  
  "message": "Hello - I'm BLUE"  
}
```

Summary

In this Lab Step you looked at the resources created by your ECS services and tasks. You also looked at the end result of your application, a JSON message on a website.

Updating the Services and Application Deployments

Introduction

Now that you know how the services interact and how to launch tasks into your ECS cluster, it is time to switch from one application to the next. A blue-green deployment is a technique that uses two identical production environments to reduce downtime. There are different cutover strategies, but generally only one environment should be serving production traffic. In this Lab Step you will launch both applications with all of the container instances you need, including a brief overlap period, to demonstrate the load balancer's round-robin distribution and container resource efficiency.

Instructions

1. Return to Services > EC2 Container Service > Clusters and click on the `ecslab-cluster`.

2. Select the `ecslab-green-service` on the Services tab and click Update:

The screenshot shows the AWS EC2 Container Service Cluster details page. The top navigation bar has tabs for Services, Tasks, ECS Instances, Metrics, and Scheduled Tasks. The Services tab is highlighted with an orange border. Below the tabs are buttons for Create, Update, and Delete. A filter input field contains the placeholder 'Filter in this page'. A status message indicates '1 selected'. The main table lists services with columns for Service Name, Status, and Task Definition. Two services are listed: 'ecslab-blue-service' and 'ecslab-green-service'. The 'ecslab-green-service' row has a checked checkbox in the first column. The 'Status' column shows 'ACTIVE' for both services. The 'Task Definition' column shows 'ecslab-blue-taskdef:1' for the blue service and 'ecslab-green-taskdef:1' for the green service.

	Service Name	Status	Task Definition
<input type="checkbox"/>	ecslab-blue-service	ACTIVE	ecslab-blue-taskdef:1
<input checked="" type="checkbox"/>	ecslab-green-service	ACTIVE	ecslab-green-taskdef:1

3. Change the Number of tasks value from *0* to *2*. Click Update Service.

4. Wait for the Service updated successfully message and click View Service.

5. Wait approximately two minutes for the new green tasks to launch. You can get service information on the service Events tab or wait for the Running count to reach 2.

6. Return to Services > Compute > EC2 > LOAD BALANCING > Target Groups and verify you have four healthy targets representing the four running container instances:

The screenshot shows the AWS Lambda console with the target group 'ecslab-targetgroup' selected. The 'Targets' tab is active, displaying four healthy targets (ECS instances) with ports 32771, 32770, 32769, and 32768, all in the 'us-west-2a' availability zone. The 'Registered targets' table lists these four instances. Below it, the 'Availability Zones' section shows one availability zone ('us-west-2a') with a target count of 4 and a healthy status.

Instance ID	Name	Port	Availability Zone	Status
i-0722705e5301ec1f3	ECS Instance - EC2ContainerService-ecslab-cluster	32771	us-west-2a	healthy ⓘ
i-0722705e5301ec1f3	ECS Instance - EC2ContainerService-ecslab-cluster	32770	us-west-2a	healthy ⓘ
i-0722705e5301ec1f3	ECS Instance - EC2ContainerService-ecslab-cluster	32769	us-west-2a	healthy ⓘ
i-0722705e5301ec1f3	ECS Instance - EC2ContainerService-ecslab-cluster	32768	us-west-2a	healthy ⓘ

Availability Zone	Target count	Healthy?
us-west-2a	4	Yes

The reduced resource requirements for a Docker container allow you to run multiple container instances on one EC2 instance. This works well for applications that require little resources, such as this simple message application.

7. Refresh your browser tab with the "Hello - I'm BLUE" message. It may take more than one refresh, but you will see the message alternate between "Hello - I'm BLUE" and "Hello - I'm GREEN" as you refresh the page and cycle through the container instances. Right now you have both applications running in their own pair of container instances.

8. Return to Services > EC2 Container Service > Clusters and click on the ecslab-cluster.

9. Select the ecslab-blue-service on the Services tab and click Update.

10. Change the Number of tasks value from 2 to 0. Click Update Service.

11. Wait for the Service updated successfully message and click View Service.

12. Return to Services > Compute > EC2 > LOAD BALANCING > Target Groups. You can watch two of the container instance targets draining as ECS stops the tasks:

Target group: **ecslab-targetgroup**

Description Targets Health checks Monitoring Tags

The load balancer starts routing requests to a newly registered target as soon as the registration process completes and the target passes the initial health checks. If demand on your targets increases, you can register additional targets. If demand on your targets decreases, you can deregister targets.

Edit

Registered targets

Instance ID	Name	Port	Availability Zone	Status
i-0f8f42c360f32a63c	ECS Instance - EC2ContainerService-ecslab-cluster	32769	us-west-2b	draining ⓘ
i-0f8f42c360f32a63c	ECS Instance - EC2ContainerService-ecslab-cluster	32768	us-west-2b	draining ⓘ
i-0f8f42c360f32a63c	ECS Instance - EC2ContainerService-ecslab-cluster	32771	us-west-2b	healthy ⓘ
i-0f8f42c360f32a63c	ECS Instance - EC2ContainerService-ecslab-cluster	32770	us-west-2b	healthy ⓘ

Availability Zones

Availability Zone	Target count	Healthy?
us-west-2b	4	Yes

13. Refresh the browser until the only message visible is "Hello - I'm GREEN". You have successfully transitioned from one application to another with no downtime or launching any new EC2 instances.

Note: There are different approaches to transitioning between one application or version to another. In this Lab Step you bring up both applications and leave them running simultaneously and externally accessible for a short time. Another approach might be to schedule a maintenance window and set the number of running tasks for the blue application to 0. Wait for the blue targets to successfully drain. Then bring up the green application if you need to avoid overlap. Total downtime remains only minutes.

Summary

In this Lab Step you launched the green application alongside the blue application in the ECS cluster. You watched as the round-robin load balancer distribution served messages from both applications running on one EC2 host instance. You then set the number of tasks for the blue application to 0 and watched as ECS drained the tasks and stopped serving the application's message.

Deleting the ECS Laboratory Resources

Introduction

One of the great features of ECS is it helps make your application robust and available. If you manually terminate the EC2 instance running your ECS tasks, ECS starts another instance and relaunches the tasks. If you try to delete a service with running tasks, ECS will not allow it. ECS does an admirable job safeguarding running services and tasks inside a cluster. To stop an application you must either bring services down one at a time or delete the entire cluster.

Instructions

1. Return to Services > EC2 Container Service > Clusters and click on the **ecslab-cluster**.
2. Click the Delete Cluster button in the upper-right.
3. Note the warning about deleting a CloudFormation stack. CloudFormation powers much of the ECS functionality and allows you to delete all of the resources in a cluster in an orderly fashion. Click **Delete**:

Delete Cluster

X

Deleting the cluster also deletes the CloudFormation stack [EC2ContainerService-ecslab-cluster](#).

Are you sure you want to delete the cluster **ecslab-cluster** and all the ECS resources within it?

[Cancel](#)

[Delete](#)

4. Wait for the Delete Cluster processes to complete. After approximately one minute you will see the Deleted cluster **ecslab-cluster** successfully message.

5. Click Repositories in the Amazon ECS column. Select the ca-container-registry repository and click Delete repository. Click Delete at the Delete repository warning indicating your images will also be deleted.

6. Return to Services > Compute > EC2 > LOAD BALANCING > Load Balancers from the EC2 Dashboard column. Select **ecslab-alb** and select Actions > Delete:

Create Load Balancer

Actions ▾

Filter: Search

Name	State
ecslab-alb	active

Delete

Edit health check

Edit subnets

Edit IP address type

Edit instances

Edit listeners

Edit security groups

Load balancer: **ecslab-alb**

Description Listeners Monitoring Tags

The screenshot shows the AWS Elastic Load Balancing (ELB) console. At the top, there's a blue header bar with the text 'Create Load Balancer'. Below it is a search bar labeled 'Search' and a 'Actions' dropdown menu. The main area displays a table with one row, 'ecslab-alb', which is active. A context menu is open over this row, listing options like 'Delete', 'Edit health check', etc. Below the table, the text 'Load balancer: **ecslab-alb**' is displayed, followed by tabs for 'Description', 'Listeners', 'Monitoring', and 'Tags'. The 'Description' tab is currently selected.

Basic Configuration

Name: `ecslab-alb`

ARN: `arn:aws:elasticloadbalancing:us-west-2:358259402422:loadbalancer/app/ecslab-alb/69747778856f43d6`

DNS name: `ecslab-alb-793477761.us-west-2.elb.amazonaws.com`
(A Record)

7. Click Yes, Delete at the Delete Load Balancer message.

8. Select LOAD BALANCING > Target Groups from the EC2 Dashboard column. Select `ecslab-targetgroup` and select Actions > Delete:

The screenshot shows the AWS CloudFormation console interface. At the top, there's a navigation bar with 'Create target group' and 'Actions' dropdown. The 'Actions' dropdown is open, showing options: 'Edit target group', 'Register and deregister targets', 'Edit attributes', and 'Delete'. Below this, there's a table with columns: 'Name' (with a checkbox), 'Protocol', 'VPC ID', and 'Monitoring'. A row is selected with the name 'ecslab-targetgroup', port 80, protocol 'HTTP', VPC ID 'vpc-9f0fd9f9', and monitoring status 'Enabled'. Below the table, it says 'Target group: ecslab-targetgroup'. Underneath, there are tabs: 'Description', 'Targets' (which is selected and highlighted in orange), 'Health checks', 'Monitoring', and 'Tags'. A note below the tabs states: 'The load balancer starts routing requests to a newly registered target as soon as the registration process completes and the target passes the initial health checks. If demand on your targets increases, you can register additional targets. If demand on your targets decreases, you can deregister targets.' An 'Edit' button is visible. The 'Registered targets' section has a table header with columns: 'Instance ID', 'Name', 'Port', 'Availability Zone', and 'Status'. A message below the header says 'There are no targets registered to this target group'. The 'Availability Zones' section has a table header with columns: 'Availability Zone', 'Target count', and 'Healthy?'. A message below the header says 'There are no targets registered to this target group'.

9. Click Yes at the Delete target group message.

Summary

In this Lab Step you deleted the key components of your ECS application including the cluster, image repository, target group, and load balancer. Other, ancillary resources remain such as the S3 source files, VPC, and CodeBuild projects. The Cloud Academy lab environment will delete these resources for you. In a real environment you may want to keep your source files and network infrastructure in case you want to update or redeploy an application later. These deletion steps stop the most expensive items from incurring additional charges. For many resource types, Amazon will charge a full hour if the resource is used during even one minute of that hour, so it is a good idea to delete resources when you are done using them.