

# LAB: Create a CodePipeline

## You need:

- An AWS Account
- Code in the CodeCommit Repository
- An existing CodeBuild Project

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

**Difficulty:** medium

## Create a Cluster

First we need to have a cluster running where we can deploy new versions of our code to. Let's create a Fargate Cluster for this example:

The screenshot shows the AWS Management Console interface for creating a cluster. The top navigation bar includes the AWS logo, 'Services', 'Resource Groups', and a star icon. The main heading is 'Create Cluster'. On the left sidebar, 'Step 1: Select cluster template' is active, and 'Step 2: Configure cluster' is listed below it. The main content area is titled 'Select cluster template' and contains a sub-header: 'The following cluster templates are available to simplify cluster creation. Additional configuration and integrations can be added later.' There are three template cards: 1. 'Networking only' (highlighted with a blue border and 'Powered by AWS Fargate') with resources: Cluster, VPC (optional), and Subnets (optional). 2. 'EC2 Linux + Networking' with resources: Cluster, VPC, Subnets, and Auto Scaling group with Linux AMI. 3. 'EC2 Windows + Networking' with resources: Cluster, VPC, Subnets, and Auto Scaling group with Windows AMI. At the bottom right, there are 'Cancel' and 'Next step' buttons. A yellow circle highlights the 'Next step' button. A '\*Required' label is visible at the bottom left of the main content area.

Give the cluster a name:

## Complete AWS ECS DevOps Masterclass for Beginners

aws

Services

Resource Groups

### Create Cluster

Step 1: Select cluster template

Step 2: Configure cluster

#### Configure cluster

Cluster name\*

fargatecluster

#### Networking

Create a new VPC for your cluster to use. A VPC is an isolated portion of the AWS Cloud populated by AWS objects, such as Fargate tasks.

Create VPC

☐ Create a new VPC for this cluster

#### Tags

Key	Value
<div>Add key</div>	<div>Add value</div>

#### CloudWatch Container Insights

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

CloudWatch Container Insights

☐ Enable Container Insights

\*Required

Cancel

Previous

Create

## Create a new Task Definition

Next, we need a new TaskDefinition for our Service:

aws

Services

Resource Groups

### Create new Task Definition

Step 1: Select launch type compatibility

Step 2: Configure task and container definitions

#### Select launch type compatibility

Select which launch type you want your task definition to be compatible with based on where you want to launch your task.

**FARGATE**

Price based on task size

Requires network mode awsvpc

AWS-managed infrastructure, no Amazon EC2 instances to manage

**EC2**

Price based on resource usage

Multiple network modes available

Self-managed infrastructure using Amazon EC2 instances

\*Required

Cancel

Next step

## Complete AWS ECS DevOps Masterclass for Beginners


### 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\*

Requires Compatibilities\* FARGATE

Task Role    
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    
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.

### Task size

The task size allows you to specify a fixed size for your task. Task size is required for tasks using the Fargate launch type and is optional for the EC2 launch type. Container level memory settings are optional when task size is set. Task size is not supported for Windows containers.

Task memory (GB)   
The valid memory range for 0.25 vCPU is: 0.5GB - 2GB.

Task CPU (vCPU)   
The valid CPU for 0.5 GB memory is: 0.25 vCPU

Task memory maximum allocation for container memory reservation

Add the image from your Elastic Container Registry:

Task execution IAM role

This role is required by tasks to pull containers from Amazon ECR. This role must have the `ecsTaskExecutionRole` policy attached.

Task size

The task size allows you to specify a fixed size for the EC2 launch type. Container level memory settings are optional when task size is set. Task size is not supported for Windows containers.

Task memory

Task CPU (vCPU)

Task memory maximum allocation for container memory reservation

Task CPU maximum allocation for container memory reservation

Add container

Standard

Container name\*

Image\*

Private repository authentication\* ☐

Memory Limits (MiB)

[Add Hard 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 MiB as a starting point for web applications.

Port mappings

Container port	Protocol
<input type="text" value="80"/>	<input type="text" value="tcp"/>

[Add port mapping](#)

Then add the container and simply create the TaskDefinition.

## Run the Service

Next we need to run the service, otherwise we don't have anything to update through our CodePipeline:

Open your Cluster and run a new service:

The screenshot shows the AWS Management Console interface for the 'fargatecluster'. The left sidebar contains navigation links for Amazon ECS (Clusters, Task Definitions, Account Settings), Amazon EKS (Clusters), Amazon ECR (Repositories), AWS Marketplace (Discover software, Subscriptions), and a search bar. The main content area displays the cluster details for 'fargatecluster', including its ARN, status (ACTIVE), and counts for registered container instances, pending tasks, running tasks, active services, and draining services. Below this, there are tabs for Services, Tasks, ECS Instances, Metrics, Scheduled Tasks, Tags, and Capacity. The 'Services' tab is selected, and the 'Create' button is highlighted with a yellow circle and a hand cursor. The 'Create' button is located in the top left of the Services section, next to 'Update', 'Delete', and 'Actions' buttons. Below these buttons is a search bar labeled 'Filter in this page' and two dropdown menus for 'Launch type' (set to ALL) and 'Service type' (set to ALL). A table with columns 'Service Name' and 'Status' is visible at the bottom of the Services section.

## Complete AWS ECS DevOps Masterclass for Beginners

### 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  
myphpcontainer ▼

Revision  
1 (latest) ▼

**Platform version** LATEST ⓘ

**Cluster** fargatecluster ⓘ

**Service name** phpservice ⓘ

**Service type\*** REPLICA ⓘ

**Number of tasks** 1| ⓘ

**Minimum healthy percent** 100 ⓘ

**Maximum percent** 200 ⓘ

## Complete AWS ECS DevOps Masterclass for Beginners

### Configure network

#### VPC and security groups

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

**Cluster VPC\*** vpc-6570b40f (172.31.0.0/16) ⓘ

**Subnets\*** ⓘ

- subnet-cfd47ba5  
(172.31.16.0/20) - eu-central-1a  
assign ipv6 on creation: Disabled
- subnet-bc21c8f0  
(172.31.0.0/20) - eu-central-1c  
assign ipv6 on creation: Disabled
- subnet-d5f6eca8  
(172.31.32.0/20) - eu-central-1b  
assign ipv6 on creation: Disabled

**Security groups\*** phpser-3639 **Edit** ⓘ

**Auto-assign public IP** ENABLED ⓘ

Don't select any load balancer and also remove the service discovery. In the next step simply Do not adjust the service's desired count and create the service.

Clusters > fargatecluster > Service: phpservice

**Service : phpservice** Update Delete

Cluster: fargatecluster  
Status: ACTIVE  
Task definition: myphpcontainer:1  
Service type: REPLICAS  
Launch type: FARGATE  
Platform version: LATEST(1.3.0)  
Service role: AWSServiceRoleForECS

Desired count: 1  
Pending count: 0  
Running count: 0

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

Last updated on April 9, 2020 1:25:48 PM (0m ago) ⓘ

Task status: Running Stopped

Filter in this page

Task	Task Definition	Last status	Desired status	Group	Launch type	Platform version
a998368e-9799-40eb-857a-cf8b...	myphpcontainer:1	PROVISIONING	RUNNING	service:phpservice	FARGATE	1.3.0

Open the Endpoint in your Browser:

## Complete AWS ECS DevOps Masterclass for Beginners

The screenshot shows the AWS Management Console interface. On the left, the navigation menu includes Amazon ECS, Clusters, Task Definitions, Account Settings, Amazon EKS, Clusters, Amazon ECR, Repositories, AWS Marketplace, Discover software, and Subscriptions. The main content area displays the details for a specific Task: **Task : a998368e-9799-40eb-857a-cf8beda25028**. The task is associated with the **fargatecluster** and is in a **RUNNING** state. The **Launch type** is **FARGATE**, and the **Platform version** is **1.3.0**. The **Task definition** is **myphpcontainer:1**, and the **Group** is **service:phpservice**. The **Task role** is **None**. The **Last status** is **RUNNING**, and the **Desired status** is **RUNNING**. The **Created at** timestamp is **2020-04-09 13:25:48 +0200**, and the **Started at** timestamp is **2020-04-09 13:26:34 +0200**. Below the task details, the **Network** section shows the **Network mode** as **awsvpc**, the **ENI Id** as **eni-08edbe2c6e584ef15**, the **Subnet Id** as **subnet-d5f6eca8**, the **Private IP** as **172.31.35.86**, and the **Public IP** as **3.127.65.62**. A red arrow points from the **Public IP** to the browser address bar in the screenshot below.

The screenshot shows a web browser window with the address bar displaying **3.127.65.62**. The browser tabs include **CodeBuild - AWS Developer Tool** and **Amazon ECS**. The address bar shows **Not secure | 3.127.65.62**. Below the address bar, a warning message is displayed: **Warning: mysqli::\_\_construct(): (HY000/2002): No such file or directory in /var/www/html/index.php on line 7**. The message is followed by **Connection unsuccessful No such file or directory**. A yellow circle highlights the warning message.

## Create a CodePipeline

Now open the Developer Tools and the CodePipeline. Create your First CodePipeline:

The screenshot shows the AWS CodePipeline console. The left sidebar contains the **Developer Tools** menu with options for **Source** (CodeCommit), **Build** (CodeBuild), **Deploy** (CodeDeploy), **Pipeline** (CodePipeline), **Getting started**, **Pipelines**, and **Settings**. The main content area displays the **AWS CodePipeline** overview, which states: **visualize and automate the different stages of your software release process**. Below this, a description reads: **AWS CodePipeline is a continuous integration and continuous delivery service for fast and reliable application and infrastructure updates. CodePipeline builds, tests, and deploys your code every time there is a code change, based on the release process models you define.** A **Create pipeline** button is visible. On the right, a **Pricing (US)** section shows **Each active pipeline\*\*** at **\$1/month\***. A note states: **\*All pipelines are free for the first 30 days.** A **Learn more** link is provided. A yellow circle highlights the **How it works** section.

## Complete AWS ECS DevOps Masterclass for Beginners

Give the pipeline a name, like “ecrtoecsexample” and leave the rest of the values at their default value:

Step 4  
Add source stage

Step 3  
Add build stage

Step 4  
Add deploy stage

Step 5  
Review

**Pipeline name**  
Enter the pipeline name. You cannot edit the pipeline name after it is created.  
ecrtoecsexample  
No more than 100 characters

**Service role**

☒ New service role  
Create a service role in your account

☐ Existing service role  
Choose an existing service role from your account

**Role name**  
AWSCodePipelineServiceRole-eu-central-1-ecrtoecsexample  
Type your service role name

☒ Allow AWS CodePipeline to create a service role so it can be used with this new pipeline

**▼ Advanced settings**

**Artifact store**

☐ Default location  
Use the default artifact store (Amazon S3 codepipeline-eu-central-1-87126036835) designated in the same region and account as your pipeline

☒ Custom location  
Choose an existing S3 location from your account in the same region and account as your pipeline

**Bucket**  
codepipeline-eu-central-1-87126036835

**Encryption key**

☒ Default AWS Managed Key  
Use the AWS managed customer master key for CodePipeline in your account to encrypt the data in the artifact store.

☐ Customer Managed Key  
To encrypt the data in the artifact store under an AWS KMS customer managed key, specify the key ID, key ARN, or alias ARN.

Cancel Next

As a source select codeCommit:



## Complete AWS ECS DevOps Masterclass for Beginners

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1  
Choose pipeline settings

Step 2  
**Add source stage**

Step 3  
Add build stage

Step 4  
Add deploy stage

Step 5  
Review

### Add source stage

#### Source

**Source provider**  
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.

AWS CodeCommit

**Repository name**  
Choose a repository that you have already created where you have pushed your source code.

myAwesomeProject

**Branch name**  
Choose a branch of the repository

master

**Change detection options**  
Choose a detection mode to automatically start your pipeline when a change occurs in the source code.

☒ **Amazon CloudWatch Events (recommended)**  
Use Amazon CloudWatch Events to automatically start my pipeline when a change occurs

☐ **AWS CodePipeline**  
Use AWS CodePipeline to check periodically for changes

Cancel Previous **Next**

Select the codeBuild project from the previous lab:

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1  
Choose pipeline settings

Step 2  
Add source stage

Step 3  
**Add build stage**

Step 4  
Add deploy stage

Step 5  
Review

### Add build stage

#### Build - optional

**Build provider**  
This is the tool of your build project. Provide build artifact details like operating system, build spec file, and output file names.

AWS CodeBuild

**Region**

Europe (Frankfurt)

**Project name**  
Choose a build project that you have already created in the AWS CodeBuild console. Or create a build project in the AWS CodeBuild console and then return to this task.

buildMyAwesomeProject or Create project

**Environment variables - optional**  
Choose the key, value, and type for your CodeBuild environment variables. In the value field, you can reference variables generated by CodePipeline. [Learn more](#)

Add environment variable

Cancel Previous Skip build stage **Next**

Select Amazon ECS as Deploy provider and then the service we just created:

## Complete AWS ECS DevOps Masterclass for Beginners

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1  
Choose pipeline settings

Step 2  
Add source stage

Step 3  
Add build stage

Step 4  
**Add deploy stage**

Step 5  
Review

### Add deploy stage

**Deploy - optional**

**Deploy provider**  
Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.

Amazon ECS

**Region**

Europe (Frankfurt)

**Cluster name**  
Choose a cluster that you have already created in the Amazon ECS console. Or create a cluster in the Amazon ECS console and then return to this task.

Q fargatecluster X

**Service name**  
Choose a service that you have already created in the Amazon ECS console for your cluster. Or create a new service in the Amazon ECS console and then return to this task.

Q phpservice X

**Image definitions file - optional**  
Enter the JSON file that describes your service's container name and the image and tag.

MyFilename.json

**Deployment timeout - optional**  
Enter the timeout in minutes for the deployment action.

Cancel Previous Skip deploy stage Next

Review everything and create the pipeline:

**Source action provider**

Source action provider  
AWS CodeCommit  
RepositoryName  
myAwesomeProject  
BranchName  
master  
PollForSourceChanges  
false

Step 3: Add build stage

**Build action provider**

Build action provider  
AWS CodeBuild  
ProjectName  
buildMyAwesomeProject

Step 4: Add deploy stage

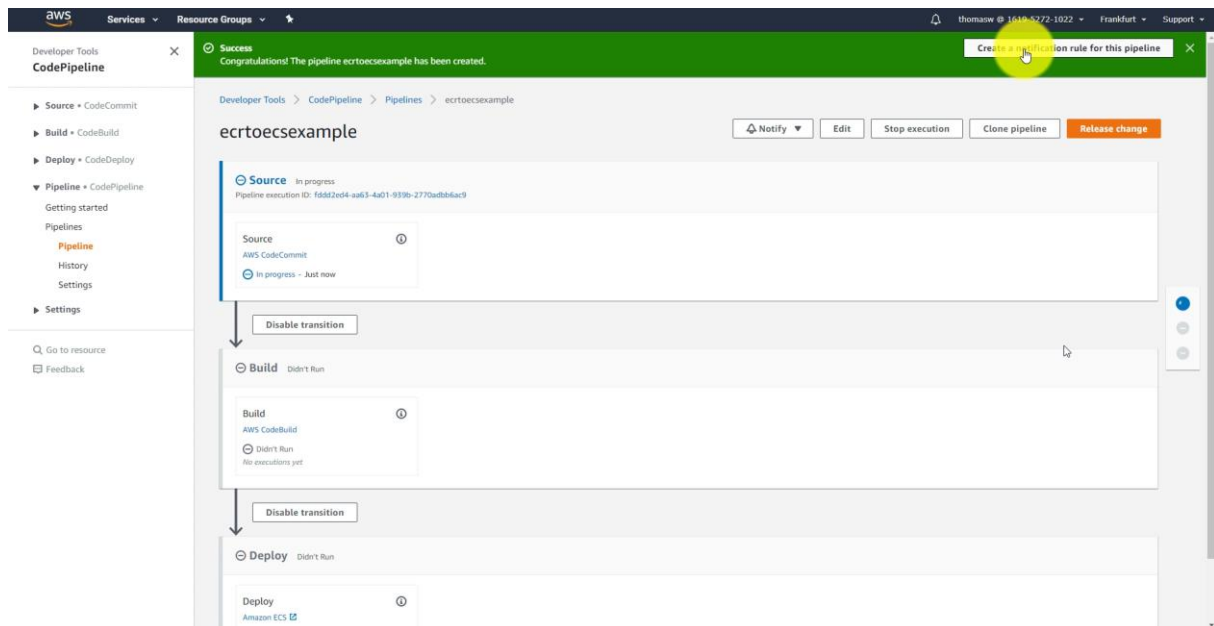
**Deploy action provider**

Deploy action provider  
AWS ECS  
ClusterName  
fargatecluster  
ServiceName  
phpservice

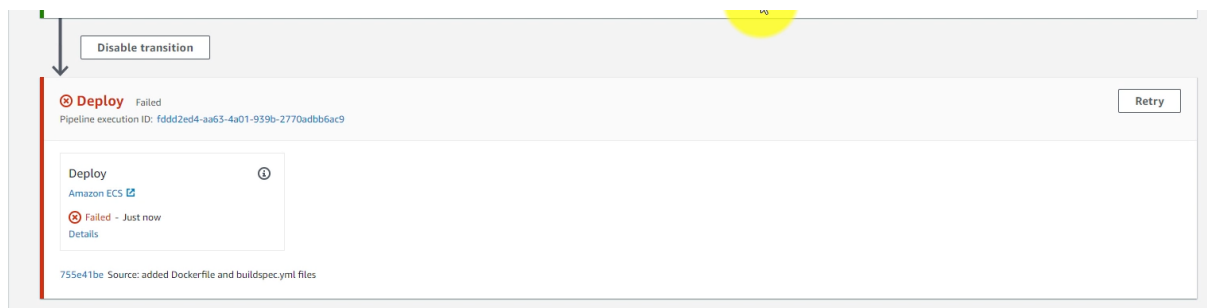
CancelPrevious>Create pipeline

You will see that CodePipeline immediately starts the pipeline:

## Complete AWS ECS DevOps Masterclass for Beginners



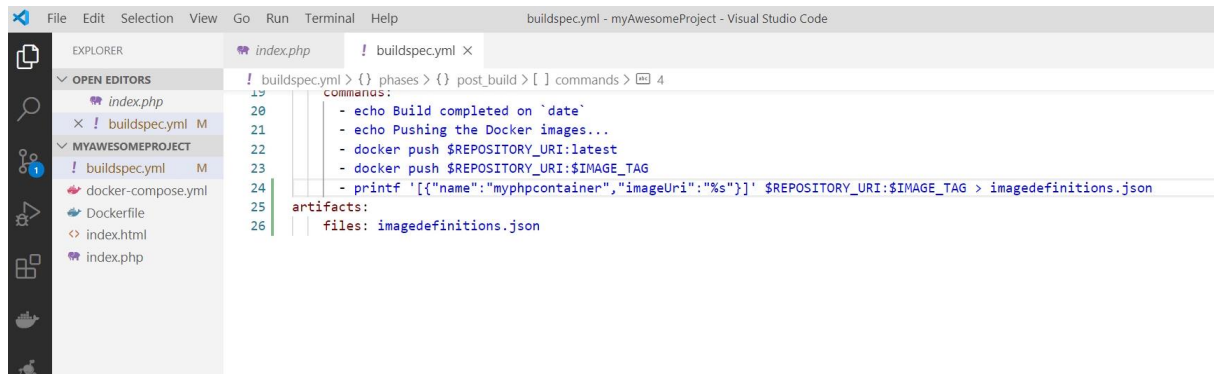
After a while it fails to deploy, because it's missing the deployment information:



From the blog entry <https://docs.aws.amazon.com/AmazonECS/latest/developerguide/ecs-cd-pipeline.html> we can derive the information we need. In our buildspec.yml file we need to add an output artifact with the necessary information for ecs to deploy the right image version:

```
post_build:
  commands:
    - echo Build completed on `date`
    - echo Pushing the Docker images...
    - docker push $REPOSITORY_URI:latest
    - docker push $REPOSITORY_URI:$IMAGE_TAG
    - echo Writing image definitions file...
    - printf '["name":"myphpcontainer", imageUri:"%s"]]'
      $REPOSITORY_URI:$IMAGE_TAG > imagedefinitions.json
  artifacts:
    files: imagedefinitions.json
```

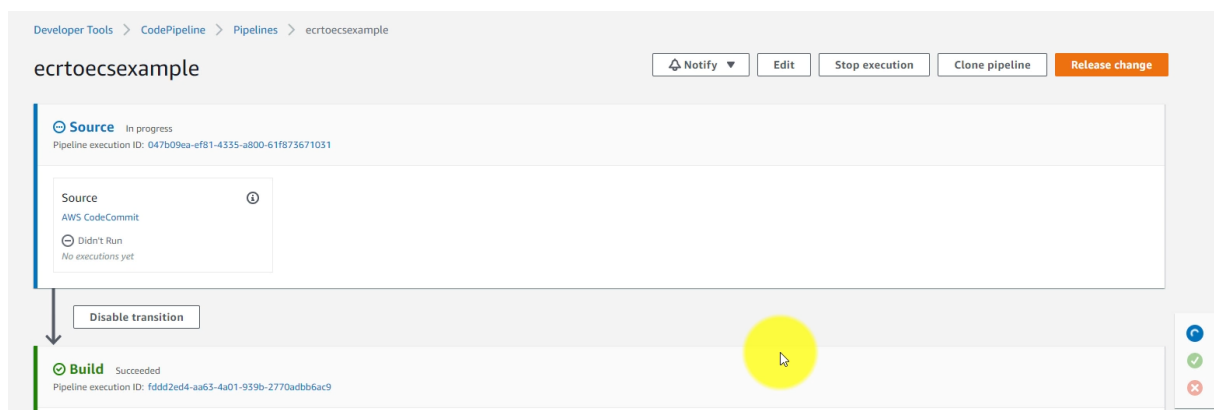
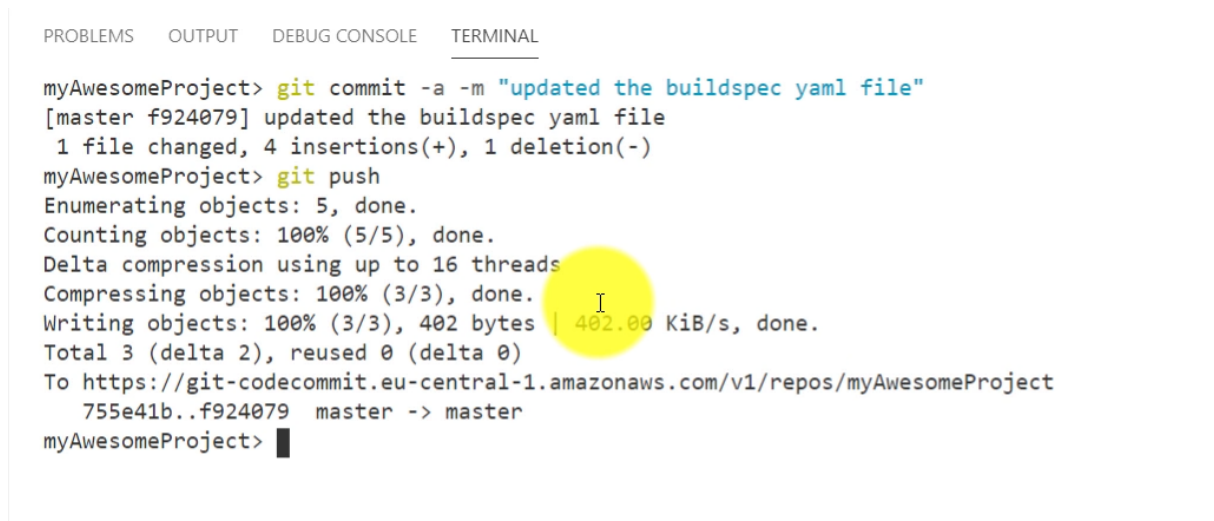
## Complete AWS ECS DevOps Masterclass for Beginners



Then commit and push the new buildspec.yml file:

```
git add . && git commit -a -m "updated buildspec.yml"
git push origin master
```

This will push the new information to the repository and at the same time start a new iteration of the codepipeline:



Wait until the build-phase is successfully completed. This might take a few minutes. Then head over to the cluster and observe how a new container get deployed into your service:

## Complete AWS ECS DevOps Masterclass for Beginners

Clusters > fargatecluster

### Cluster : fargatecluster

Update Cluster Delete Cluster

Get a detailed view of the resources on your cluster.

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

Status: **ACTIVE**

Registered container instances: 0

Pending tasks count: 1 Fargate, 0 EC2

Running tasks count: 1 Fargate, 0 EC2

Active service count: 1 Fargate, 0 EC2

Draining service count: 0 Fargate, 0 EC2

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

Run new Task Stop Stop All Actions

Desired task status: **Running** Stopped

Filter in this page Launch type: ALL

Task	Task definition	Container instance	Last status	Desired status	Started By	Group	Launch type	Platform version
<input type="checkbox"/> a998368e-9799-40eb-8...	myphpcontainer	-	<b>RUNNING</b>	RUNNING	ecs-svc/756612196971...	service:phpservice	FARGATE	1.3.0
<input type="checkbox"/> f977438c-544e-45a3-8...	myphpcontainer	-	PROVISIONING	RUNNING	ecs-svc/020110725120...	service:phpservice	FARGATE	1.3.0

The new container will run under a new IP address. That means, to open it, you have to copy the new IP and open it in a new tab, because we didn't forward anything to a load balancer or anything alike.

In the next few labs we will see how we can get database access.

---

*Lab End*

---