

# Getting Started with Amazon EC2 Containers, Part 1

*AWS lets users create and manage containers through a graphical console -- perfect for those unused to working with containers or the Docker command-line environment.*

- By Brien Posey
- 07/17/2017

- Read Part 2 [here](#).

<https://awsinsider.net/articles/2017/07/17/amazon-ec2-containers-1.aspx>

Containers have become all the rage over the last few years.

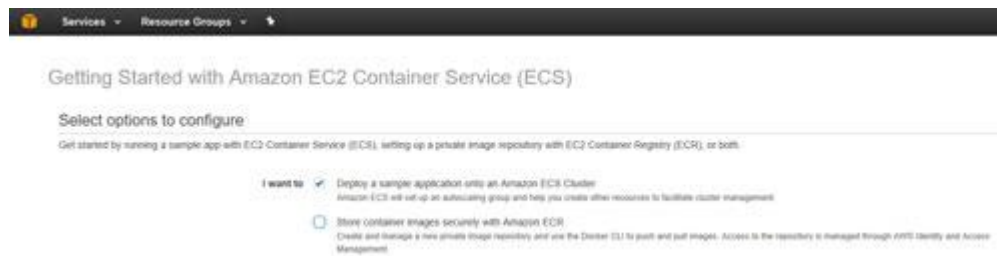
While you can certainly create an on-premises container infrastructure based on Linux or Windows, it is also possible to run containerized applications on Amazon Web Services' Elastic Compute Cloud (EC2).

In fact, EC2 is a great starting point for anyone who might have been curious about containers, but has thus far avoided them. The reason for this is that AWS allows containers to be created and managed through a graphical console, thereby eliminating the complexity of the Docker command-line environment.

In this article, I will show you how to deploy a simple containerized application on EC2 using the **EC2 Container Service**, or ECS.

Begin the process by logging in to the AWS console, and then choose EC2 Container Service from the list of available services (it's in the Compute section). At this point, you will be taken to the Amazon EC2 Container Service dashboard.

Rather than using the options presented at the bottom of the dashboard window, click the big blue Get Started button instead. You should now see a screen that asks you to select the option that you want to configure. Choose the option to deploy a sample application onto an Amazon ECS cluster, and clear the other checkbox, as shown in **Figure 1**.



[Click on image for larger view.] *Figure 1: Choose the*

*option to deploy a sample application onto an Amazon ECS cluster.*

Click Continue, and you will be taken to the Create a Task Definition screen. Under normal circumstances, you would be required to populate all of the fields that are displayed on the screen. In this case, however, AWS has pre-populated the fields in an effort to teach you how to create a containerized application, as shown in **Figure 2**.

Services → Resource Groups →

## Getting Started with Amazon EC2 Container Service (ECS)

**Step 1: Create a task definition**

Step 2: Configure service

Step 3: Configure cluster

Step 4: Review

### Create a task definition

An Amazon ECS task definition is a blueprint or recipe for containers. You can modify parameters in the task definition to suit your particular application (for example, to provide more CPU resources or change the port mappings). [Learn more](#)

**Task definition name\*** console-sample-app-static ⓘ

**Container name\*** simple-app ⓘ

**Image\*** httpd:2.4 ⓘ

Custom image format: {registry-url}/{namespace}/{image}:{tag}

**Memory Limits (MB)\*** Hard limit ▾ 300 ⓘ

[Add Soft limit](#)

Define hard and/or soft memory limits in MB for your container. Hard and soft limits correspond to the 'memory' and 'memoryReservation' parameters, respectively, in task definitions. ECS recommends 300-512 MB as a starting point for web applications.

**Port mappings** ⓘ

Host port	Container port	Protocol
0	80	tcp ▾

[Add port mapping](#)

[Advanced options](#)

[Click on image for larger view.] **Figure 2: AWS has**

***pre-populated the various fields.***

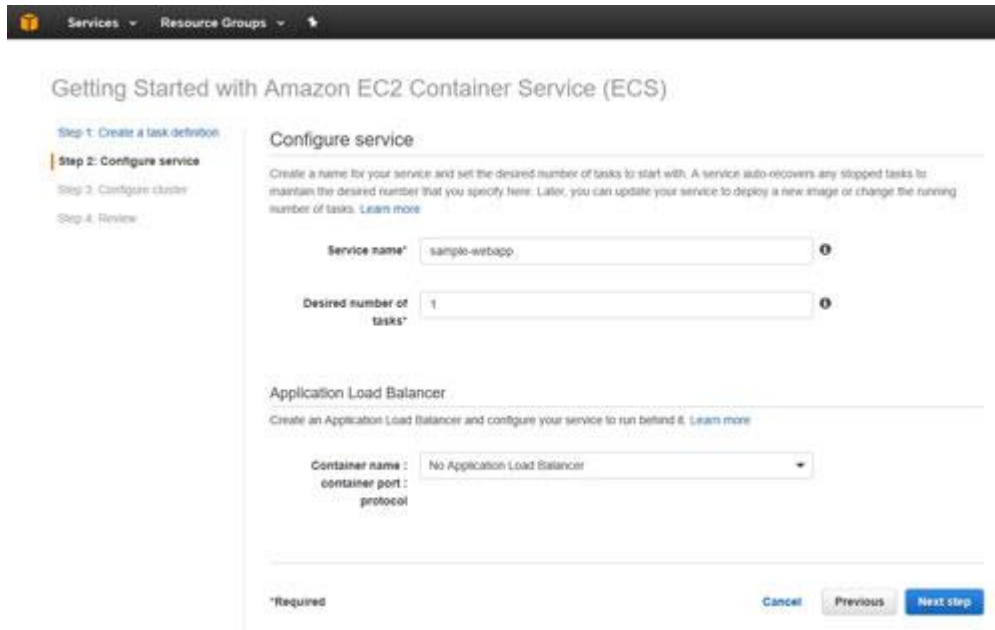
The first thing that you will have to enter is a task definition name. This is just a friendly name used to identify the task that you are creating. You will also have to enter a container name. As you can see in the figure above, the container name (simple-app) is pre-populated, as is the name of the container image (httpd:2.4).

This screen also contains options for configuring the container's memory usage and port mappings. The sample application is set to use 300MB of memory and TCP port 80.

Click Next, and you will be taken to the Configure Service screen, which you can see in **Figure 3**. This screen asks you to provide a service name and the number of tasks that you want to use. Think of a task as

a copy of the containerized application. Thus, two tasks would correspond to two instances of the application. The service's job is to make sure that the correct number of tasks are running.

Incidentally, this screen also gives you the option of load balancing the application, but load balancing isn't necessary in this case because we are just deploying a sample Web app in an effort to get a feel for how containers work.

The screenshot shows the 'Configure service' step in the Amazon ECS console. On the left, a sidebar lists four steps: 'Step 1: Create a task definition', 'Step 2: Configure service' (which is highlighted with a yellow bar), 'Step 3: Configure cluster', and 'Step 4: Review'. The main area is titled 'Configure service' and contains a sub-header 'Configure service' with a description: 'Create a name for your service and set the desired number of tasks to start with. A service auto-recovers any stopped tasks to maintain the desired number that you specify here. Later, you can update your service to deploy a new stage or change the running number of tasks. Learn more'. Below this, there are two input fields: 'Service name' with the value 'sample-webapp' and 'Desired number of tasks' with the value '1'. Both fields have an information icon to their right. Further down, there is a section for 'Application Load Balancer' with a description: 'Create an Application Load Balancer and configure your service to run behind it. Learn more'. Below this, there is a dropdown menu for 'Container name' with the value 'No Application Load Balancer'. Below the dropdown, there are labels for 'container port' and 'protocol'. At the bottom of the form, there is a '\*Required' label and three buttons: 'Cancel', 'Previous', and 'Next step'.

[Click on image for larger view.] **Figure 3: Enter a**

***service name and specify the number of tasks that you want to link to the service.***

Click Next Step, and you will be prompted to set up a cluster on which to run your container. A cluster is really nothing more than a collection of EC2 instances that work together to run the containers that you are creating. As you can see in **Figure 4**, you will need to provide a name for the cluster that you are creating, select an instance type and specify the number of instances that you want to use.

The interesting thing about the demo configuration is that this particular cluster will be based around a single EC2 instance. As such, the argument could be made that this isn't a "true cluster." It's more like running a container on a standalone PC.

## Getting Started with Amazon EC2 Container Service (ECS)

Step 1: Create a task definition  
Step 2: Configure service  
**Step 3: Configure cluster**  
Step 4: Review

### Configure cluster

Your Amazon ECS tasks run on container instances (Amazon EC2 instances that are running the ECS container agent). Configure the instance type, instance quantity, and other details of the container instances to launch into your cluster.

Cluster name\*  ⓘ

EC2 instance type\*  ⓘ

Number of instances\*  ⓘ

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 [↗](#).

### Security group

By default, your instances are accessible from any IP address. We recommend that you update the below security group ingress rule to allow access from known IP addresses only. ECS automatically opens up port 80 to facilitate access to the application or service you're running.

Allowed ingress source(s)\*  ⓘ

### 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 You are giving permission to EC2 Container Service to create and use `ecsInstanceRole`. ⓘ

[Click on image for larger view.] **Figure 4: This is the**

***screen that you will use to configure the container cluster.***

You will also need to provide a key pair if you want to be able to SSH into the EC2 instance. Use the drop-down list to select a previously existing key pair. As an alternative, you can go to the EC2 console and create a new key pair.

Another thing that you will have to do before continuing is to specify an allowed ingress source. By default, the container instance will be accessible from any IP address, but AWS recommends changing the rule to allow access only from authorized IP addresses.

Now, click the Review and Launch button, and you will see a container summary. If everything looks good, click on the Launch Instance & Run Service button.

As you can see, AWS makes it really easy to create a sample container. In [Part 2 here](#), I will show you what the container looks like once it is up and running, and how to manage the container.

### **About the Author**

*Brien Posey is a seven time Microsoft MVP with over two decades of IT experience. As a freelance writer, Posey has written many thousands of articles and written or contributed to several dozen books on a wide variety of IT topics. Prior to going freelance, Posey was a CIO for a national chain of hospitals and healthcare facilities. He has also served as a network administrator for some of the country's largest insurance companies and for the Department of Defense at Fort Knox. When He isn't busy writing, Brien Posey enjoys exotic travel, scuba diving, and racing his Cigarette boat. You can visit his personal Web site at: [www.brienposey.com](http://www.brienposey.com).*