**What is Amazon Elastic Container Service?**

Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast, container management service that makes it easy to run, stop, and manage Docker containers on a cluster. You can host your cluster on a serverless infrastructure that is managed by Amazon ECS by launching your services or tasks using the Fargate launch type. For more control you can host your tasks on a cluster of Amazon Elastic Compute Cloud (Amazon EC2) instances that you manage by using the EC2 launch type. For more information about launch types, see [Amazon ECS Launch Types](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/launch_types.html).

Amazon ECS lets you launch and stop container-based applications with simple API calls, allows you to get the state of your cluster from a centralized service, and gives you access to many familiar Amazon EC2 features.

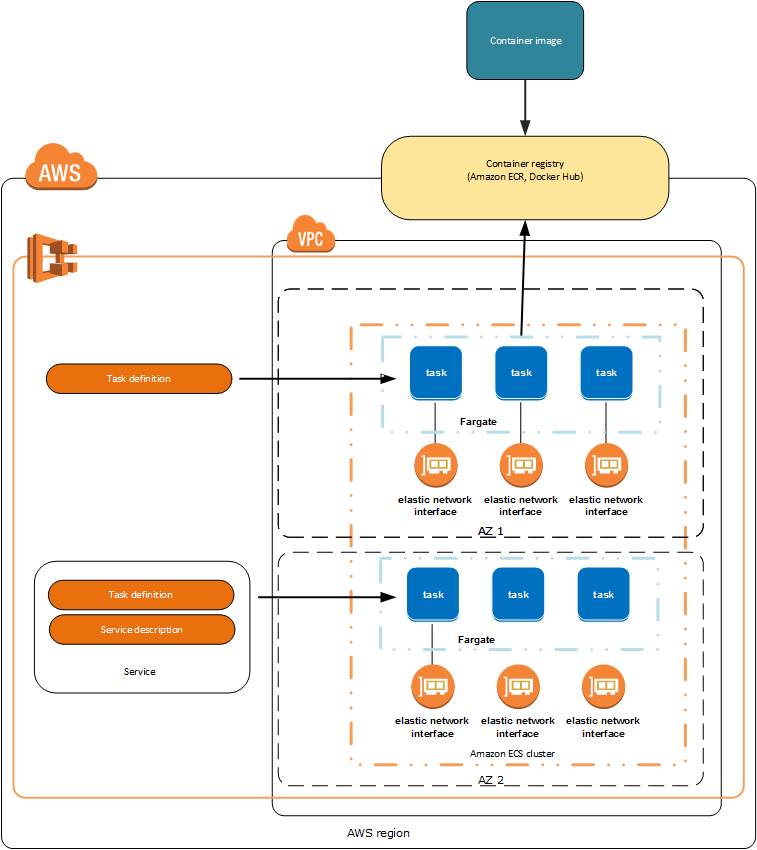
You can use Amazon ECS to schedule the placement of containers across your cluster based on your resource needs, isolation policies, and availability requirements. Amazon ECS eliminates the need for you to operate your own cluster management and configuration management systems or worry about scaling your management infrastructure.

Amazon ECS can be used to create a consistent deployment and build experience, manage, and scale batch and Extract-Transform-Load (ETL) workloads, and build sophisticated application architectures on a microservices model. For more information about Amazon ECS use cases and scenarios, see [Container Use Cases](https://aws.amazon.com/containers/use-cases/).

**Features of Amazon ECS**

Amazon ECS is a regional service that simplifies running application containers in a highly available manner across multiple Availability Zones within a region. You can create Amazon ECS clusters within a new or existing VPC. After a cluster is up and running, you can define task definitions and services that specify which Docker container images to run across your clusters. Container images are stored in and pulled from container registries, which may exist within or outside of your AWS infrastructure.

The following diagram shows the architecture of an Amazon ECS environment using the Fargate launch type:



The following sections dive into these individual elements of the Amazon ECS architecture in more detail.

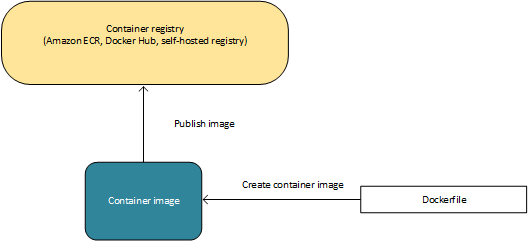
**Containers and Images**

To deploy applications on Amazon ECS, your application components must be architected to run in *containers*. A Docker container is a standardized unit of software development, containing everything that your software application needs to run: code, runtime, system tools, system libraries, etc. Containers are created from a read-only template called an *image*.

Images are typically built from a Dockerfile, a plain text file that specifies all of the components that are included in the container. These images are then stored in a *registry* from which they can be downloaded and run on your cluster. For more information about container technology, see [Docker Basics for Amazon ECS](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/docker-basics.html).

**Note**

The Fargate launch type only supports using container images hosted in Amazon ECR or publicly on Docker Hub. Private repositories are currently only supported using the EC2 launch type.



**Task Definitions**

To prepare your application to run on Amazon ECS, you create a *task definition*. The task definition is a text file, in JSON format, that describes one or more containers, up to a maximum of ten, that form your application. It can be thought of as a blueprint for your application. Task definitions specify various parameters for your application. Examples of task definition parameters are which containers to use, which launch type to use, which ports should be opened for your application, and what data volumes should be used with the containers in the task. The specific parameters available for the task definition depend on which launch type you are using. For more information about creating task definitions, see [Amazon ECS Task Definitions](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/task_definitions.html).

The following is an example of a simple task definition containing a single container that runs an NGINX web server using the Fargate launch type. For a more extended example demonstrating the use of multiple containers in a task definition, see [Example Task Definitions](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/example_task_definitions.html).

{

"family": "webserver",

"containerDefinitions": [

{

"name": "web",

"image": "nginx",

"memory": "100",

"cpu": "99"

},

],

"requiresCompatibilities": [

"FARGATE"

],

"networkMode": "awsvpc",

"memory": "512",

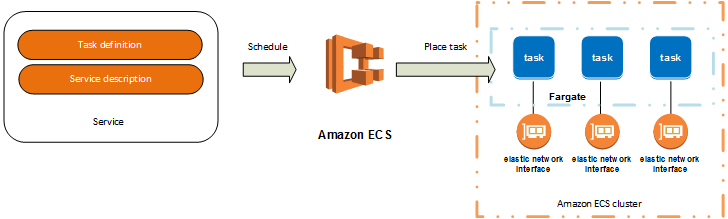
"cpu": "256",

}

**Tasks and Scheduling**

A *task* is the instantiation of a task definition within a cluster. After you have created a task definition for your application within Amazon ECS, you can specify the number of tasks that will run on your cluster.

The Amazon ECS task scheduler is responsible for placing tasks within your cluster. There are several different scheduling options available. For example, you can define a *service* that runs and maintains a specified number of tasks simultaneously. For more information about the different scheduling options available, see [Scheduling Amazon ECS Tasks](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/scheduling_tasks.html).



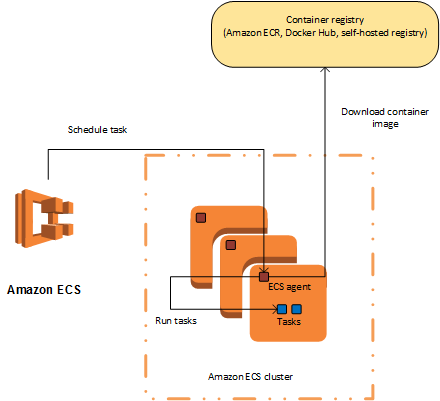
**Clusters**

When you run tasks using Amazon ECS, you place them on a *cluster*, which is a logical grouping of resources. If you use the Fargate launch type with tasks within your cluster, Amazon ECS manages your cluster resources. If you use the EC2 launch type, then your clusters will be a group of container instances you manage. Amazon ECS downloads your container images from a registry that you specify, and runs those images within your cluster.

For more information about creating clusters, see [Amazon ECS Clusters](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/ECS_clusters.html). If you are using the EC2 launch type, you can read about creating container instances at [Amazon ECS Container Instances](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/ECS_instances.html).

**Container Agent**

The *container agent* runs on each infrastructure resource within an Amazon ECS cluster. It sends information about the resource's current running tasks and resource utilization to Amazon ECS, and starts and stops tasks whenever it receives a request from Amazon ECS. For more information, see [Amazon ECS Container Agent](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/ECS_agent.html).



**How to Get Started with Amazon ECS**

If you are using Amazon ECS for the first time, the AWS Management Console for Amazon ECS provides a first-run wizard that steps you through defining a task definition for a web server, configuring a service, and launching your first Fargate cluster. The first-run wizard is highly recommended for users who have no prior experience with Amazon ECS. For more information, see the [Getting Started with Amazon ECS using Fargate](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/ECS_GetStarted.html)tutorial.

Alternatively, you can install the AWS Command Line Interface (AWS CLI) to use Amazon ECS. For more information, see [Setting Up with Amazon ECS](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/get-set-up-for-amazon-ecs.html).

**Related Services**

Amazon ECS can be used along with the following AWS services:

**AWS Identity and Access Management**

IAM is a web service that helps you securely control access to AWS resources for your users. Use IAM to control who can use your AWS resources (authentication) and what resources they can use in which ways (authorization). In Amazon ECS, IAM can be used to control access at the container instance level using IAM roles, and at the task level using IAM task roles. For more information, see [Amazon ECS IAM Policies, Roles, and Permissions](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/IAM_policies.html).

**Amazon EC2 Auto Scaling**

Auto Scaling is a web service that enables you to automatically scale out or in your tasks based on user-defined policies, health status checks, and schedules. You can use Auto Scaling with a Fargate task within a service to scale in response to a number of metrics or with a EC2 task to scale the container instances within your cluster. For more information, see [Tutorial: Scaling Container Instances with CloudWatch Alarms](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/cloudwatch_alarm_autoscaling.html).

**Elastic Load Balancing**

Elastic Load Balancing automatically distributes incoming application traffic across multiple EC2 instances in the cloud. It enables you to achieve greater levels of fault tolerance in your applications, seamlessly providing the required amount of load balancing capacity needed to distribute application traffic. You can use Elastic Load Balancing to create an endpoint that balances traffic across services in a cluster. For more information, see [Service Load Balancing](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/service-load-balancing.html).

**Amazon Elastic Container Registry**

Amazon ECR is a managed AWS Docker registry service that is secure, scalable, and reliable. Amazon ECR supports private Docker repositories with resource-based permissions using IAM so that specific users or EC2 instances can access repositories and images. Developers can use the Docker CLI to push, pull, and manage images. For more information, see the [Amazon Elastic Container Registry User Guide](http://docs.aws.amazon.com/AmazonECR/latest/userguide/).

**AWS CloudFormation**

AWS CloudFormation gives developers and systems administrators an easy way to create and manage a collection of related AWS resources, provisioning and updating them in an orderly and predictable fashion. You can define clusters, task definitions, and services as entities in an AWS CloudFormation script. For more information, see [AWS CloudFormation Template Reference](http://docs.aws.amazon.com/AWSCloudFormation/latest/APIReference/).

**Accessing Amazon ECS**

You can work with Amazon ECS in the following ways:

**AWS Management Console**

The console is a browser-based interface to manage Amazon ECS resources. For a tutorial that guides you through the console, see [Getting Started with Amazon ECS using Fargate](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/ECS_GetStarted.html).

**AWS command line tools**

You can use the AWS command line tools to issue commands at your system's command line to perform Amazon ECS and AWS tasks; this can be faster and more convenient than using the console. The command line tools are also useful for building scripts that perform AWS tasks.

AWS provides two sets of command line tools: the [AWS Command Line Interface](http://docs.aws.amazon.com/cli/latest/userguide/) (AWS CLI) and the [AWS Tools for Windows PowerShell](http://docs.aws.amazon.com/powershell/latest/userguide/). For more information, see the [AWS Command Line Interface User Guide](http://docs.aws.amazon.com/cli/latest/userguide/) and the [AWS Tools for Windows PowerShell User Guide](http://docs.aws.amazon.com/powershell/latest/userguide/).

**Amazon ECS CLI**

In addition to using the AWS CLI to access Amazon ECS resources, you can use the Amazon ECS CLI, which provides high-level commands to simplify creating, updating, and monitoring clusters and tasks from a local development environment using Docker Compose. For more information, see [Using the Amazon ECS Command Line Interface](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/ECS_CLI.html).

**AWS SDKs**

We also provide SDKs that enable you to access Amazon ECS from a variety of programming languages. The SDKs automatically take care of tasks such as:

* Cryptographically signing your service requests
* Retrying requests
* Handling error responses

For more information about available SDKs, see [Tools for Amazon Web Services](https://aws.amazon.com/tools/).