Amazon ECS

General ECS Concepts

Amazon Elastic Container Service (ECS) is a highly scalable, high performance container management service that supports Docker containers and allows you to easily run applications on a managed cluster of Amazon EC2 instances.

Amazon ECS eliminates the need for you to install, operate, and scale your own cluster management infrastructure.

Using API calls you can launch and stop container-enabled applications, query the complete state of clusters, and access many familiar features like security groups, Elastic Load Balancing, EBS volumes and IAM roles.

Amazon ECS can be used to schedule the placement of containers across clusters based on resource needs and availability requirements.

There is no additional charge for Amazon ECS. You pay for AWS resources (e.g. EC2 instances or EBS volumes) you create to store and run your application.

Possible to use Elastic Beanstalk to handle the provisioning of an Amazon ECS cluster, balancing load, auto-scaling, monitoring, and placing your containers across your cluster.

Alternatively use ECS directly for more fine-grained control for customer application architectures.

It is possible to associate a service on Amazon ECS to an Application Load Balancer (ALB) for the Elastic Load Balancing (ELB) service.

The ALB supports a target group that contains a set of instance ports. You can specify a dynamic port in the ECS task definition which gives the container an unused port when it is scheduled on the EC2 instance.

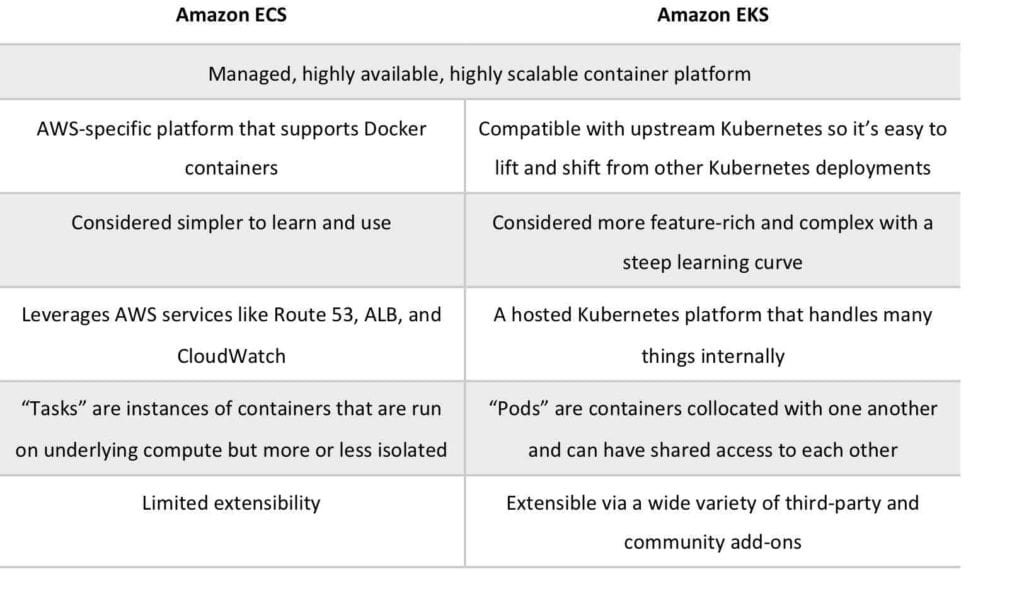
ECS provides Blox, a collection of open source projects for container management and orchestration. Blox makes it easy to consume events from Amazon ECS, store the cluster state locally and query the local data store through APIs.

You can use any AMI that meets the Amazon ECS AMI specification.

ECS vs EKS

Amazon also provide the Elastic Container Service for Kubernetes (Amazon EKS) which can be used to deploy, manage, and scale containerized applications using Kubernetes on AWS.

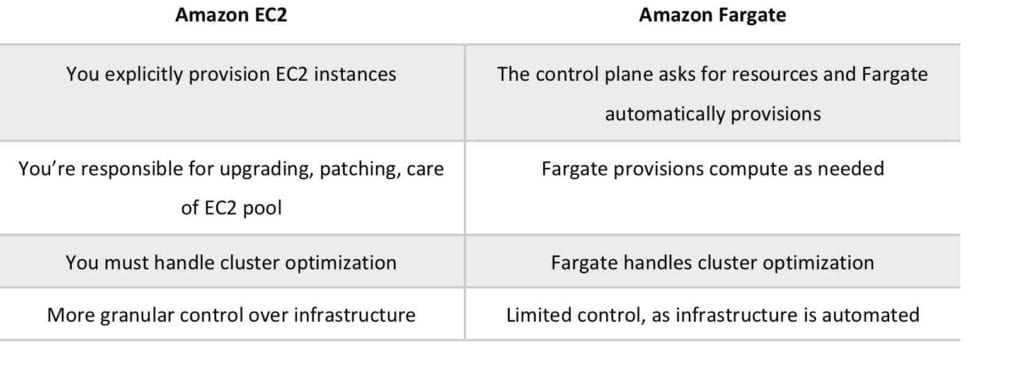
The table below describes some of the differences between these services to help you understand when you might choose one over the other:



Launch Types

An Amazon ECS launch type determines the type of infrastructure on which your tasks and services are hosted.

There are two launch types and the table below describes some of the differences between the two launch types:



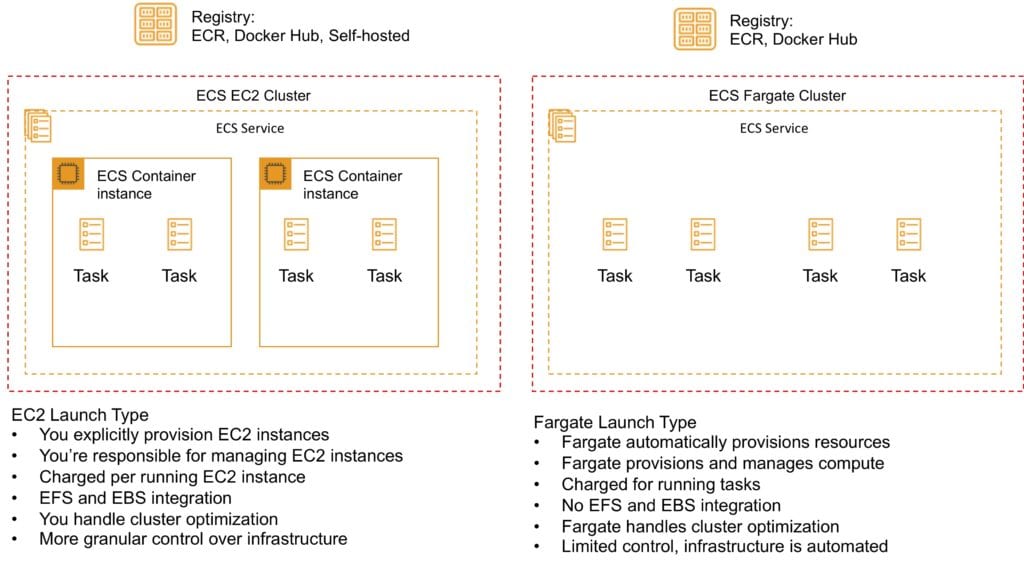
Fargate Launch Type

* The Fargate launch type allows you to run your containerized applications without the need to provision and manage the backend infrastructure. Just register your task definition and Fargate launches the container for you.
* Fargate Launch Type is a serverless infrastructure managed by AWS.
* Fargate only supports container images hosted on Elastic Container Registry (ECR) or Docker Hub.

EC2 Launch Type

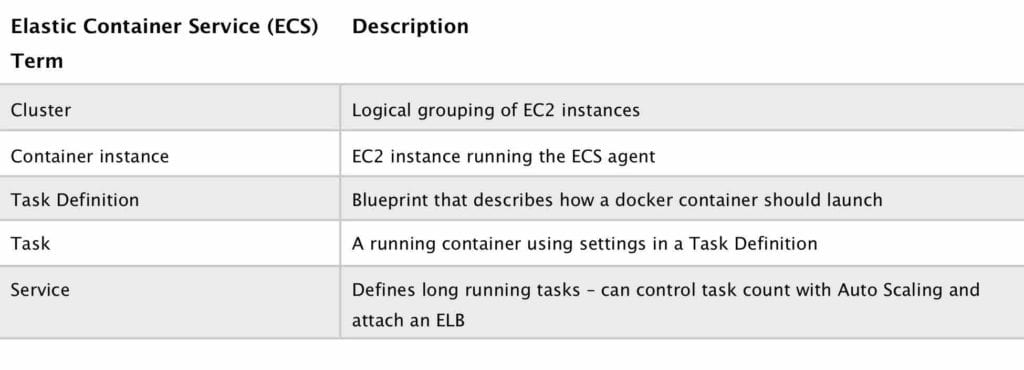
* The EC2 launch type allows you to run your containerized applications on a cluster of Amazon EC2 instances that you manage.
* Private repositories are only supported by the EC2 Launch Type.

The following diagram shows the two launch types and summaries some key differences:



ECS Terms

The following table provides an overview of some of the terminology used with Amazon ECS:



Images

Containers are created from a read-only template called an image which has the instructions for creating a Docker container.

Images are built from a Dockerfile.

Only Docker containers are currently supported.

An image contains the instructions for creating a Docker container.

Images are stored in a registry such as DockerHub or AWS Elastic Container Registry (ECR).

ECR is a managed AWS Docker registry service that is secure, scalable and reliable.

ECR supports private Docker repositories with resource-based permissions using AWS IAM in order to access repositories and images.

Developers can use the Docker CLI to push, pull and manage images.

Tasks

A task definition is required to run Docker containers in Amazon ECS.

A task definition is a text file in JSON format that describes one or more containers, up to a maximum of 10.

Task definitions use Docker images to launch containers.

You specify the number of tasks to run (i.e. the number of containers).

Some of the parameters you can specify in a task definition include:

* Which Docker images to use with the containers in your task.
* How much CPU and memory to use with each container.
* Whether containers are linked together in a task.
* The Docker networking mode to use for the containers in your task.
* What (if any) ports from the container are mapped to the host container instances.
* Whether the task should continue if the container finished or fails.
* The commands the container should run when it is started.
* Environment variables that should be passed to the container when it starts.
* Data volumes that should be used with the containers in the task.
* IAM role the task should use for permissions.

You can use Amazon ECS Run task to run one or more tasks once.

Clusters

ECS Clusters are a logical grouping of container instances the you can place tasks on.

A default cluster is created but you can then create multiple clusters to separate resources.

ECS allows the definition of a specified number (desired count) of tasks to run in the cluster.

Clusters can contain tasks using the Fargate and EC2 launch type.

For clusters with the EC2 launch type clusters can contain different container instance types.

Each container instance may only be part of one cluster at a time.

“Services” provide auto-scaling functions for ECS.

Clusters are region specific.

You can create IAM policies for your clusters to allow or restrict users’ access to specific clusters.

Service Scheduler

You can schedule ECS using Service Scheduler and Custom Scheduler.

Ensures that the specified number of tasks are constantly running and reschedules tasks when a task fails.

Can ensure tasks are registered against an ELB.

Custom Scheduler

You can create your own schedulers to meet business needs.

Leverage third party schedulers such as Blox.

The Amazon ECS schedulers leverage the same cluster state information provided by the Amazon ECS API to make appropriate placement decisions.

ECS Container Agent

The ECS container agent allows container instances to connect to the cluster.

The container agent runs on each infrastructure resource on an ECS cluster.

The ECS container agent is included in the Amazon ECS optimized AMI and can also be installed on any EC2 instance that supports the ECS specification (only supported on EC2 instances).

Linux and Windows based.

For non-AWS Linux instances to be used on AWS you must manually install the ECS container agent.

Auto Scaling

Service Auto Scaling

Amazon ECS service can optionally be configured to use Service Auto Scaling to adjust the desired task count up or down automatically.

Service Auto Scaling leverages the Application Auto Scaling service to provide this functionality.

Amazon ECS Service Auto Scaling supports the following types of scaling policies:

* [**Target Tracking Scaling Policies**](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/service-autoscaling-targettracking.html)—Increase or decrease the number of tasks that your service runs based on a target value for a specific CloudWatch metric. This is similar to the way that your thermostat maintains the temperature of your home. You select temperature and the thermostat does the rest.
* [**Step Scaling Policies**](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/service-autoscaling-stepscaling.html)—Increase or decrease the number of tasks that your service runs in response to CloudWatch alarms. Step scaling is based on a set of scaling adjustments, known as step adjustments, which vary based on the size of the alarm breach.

Cluster Auto Scaling

This is a new feature released in December 2019. It is unlikely that this will appear on the SAA-C01 exam but could appear on the SAA-C02 exam.

Uses a new ECS resource type called a Capacity Provider.

A Capacity Provider can be associated with an EC2 Auto Scaling Group (ASG).

When you associate an ECS Capacity Provider with an ASG and add the Capacity Provider to an ECS cluster, the cluster can now scale your ASG automatically by using two new features of ECS:

1. **Managed scaling**, with an automatically-created scaling policy on your ASG, and a new scaling metric (Capacity Provider Reservation) that the scaling policy uses; and
2. **Managed instance termination protection**, which enables container-aware termination of instances in the ASG when scale-in happens.

Prefer to learn by doing? Watch the AWS Hands-On Labs video tutorial below to learn how to create an Amazon ECS cluster and a task running WordPress. We’ll show you how to do this using a combination of the AWS ECS CLI and the console.

Security/SLA

EC2 instances use an IAM role to access ECS.

IAM can be used to control access at the container level using IAM roles.

The container agent makes calls to the ECS API on your behalf through the applied IAM roles and policies.

You need to apply IAM roles to container instances before they are launched (EC2 launch type).

AWS recommend limiting the permissions that are assigned to the container instance’s IAM roles.

Assign extra permissions to tasks through separate IAM roles (IAM Roles for Tasks).

ECS tasks use an IAM role to access services and resources.

Security groups attach at the instance or container level.

You have root level access to the operating system of the EC2 instances.

The Compute SLA guarantees a Monthly Uptime Percentage of at least 99.99% for Amazon ECS.

Limits

Soft limits (default):

* Clusters per region = 1000.
* Instances per cluster = 1000.
* Services per cluster = 500.

Hard limits:

* One load balancer per service.
* 1000 tasks per service (the “desired” count).
* Max 10 containers per task definition.
* Max 10 tasks per instance (host).

Pricing

EC2 Launch Type:

* No additional charge – you pay for the EC2 resources you launch including instances, EBS volumes and load balancers

Fargate:

* You pay for the vCPU and memory allocated to the containers you run