**Container Services**

Containers provide a standard way to package your application's code, configurations, and dependencies into a single object. Containers share an operating system installed on the server and run as resource-isolated processes, ensuring quick, reliable, and consistent deployments, regardless of environment.

When you want to run lots of Docker containers, orchestration tools like Docker Swarm, Kubernetes, and Amazon ECS make it possible to start, stop, and monitor thousands (or millions) of containers.

Docker Swarm is container orchestration software made by Docker that you run and manage yourself. Kubernetes is a popular open source, community maintained container orchestration software that you run and manage yourself. Amazon **EKS** makes it easier to run Kubernetes on AWS by managing the Kubernetes control plane for your containers. Amazon **ECS** is a fully managed AWS service that makes it easy to run containers on AWS with deep integrations to AWS services such as VPC, load balancing, service discovery, and IAM roles.

Amazon **ECR** is integrated with both Amazon ECS and Amazon EKS, allowing you to easily store, run, and manage container images for applications running on ECS or EKS. All you need to do is specify the Amazon ECR repository in your task or pod definition and ECS or EKS will retrieve the appropriate images for your applications.

**Elastic Container Registry** Description

Amazon **Elastic Container Registry** (Amazon **ECR**) is an AWS managed container image registry service that is secure, scalable, and reliable. Amazon ECR supports private container image repositories with resource-based permissions using AWS IAM. This is so that specified users or Amazon EC2 instances can access your container repositories and images. You can use your preferred CLI to push, pull, and manage Docker images, Open Container Initiative (OCI) images, and OCI compatible artifacts.

**Amazon Elastic Container Registry Public** is a managed AWS container image registry service that is secure, scalable, and reliable. Amazon ECR supports public image repositories with resource-based permissions using AWS IAM so that specific users can access your public repositories to push images. Developers can use their preferred CLI to push and manage Docker images, Open Container Initiative (OCI) images, and OCI compatible artifacts. Your images are publicly available to pull, either anonymously or using an Amazon ECR Public authentication token.

## Service Overview

Amazon ECR contains the following components:

* An **Amazon ECR registry** is provided to each AWS account; you can create image repositories in your registry and store images in them. For more information, see [Amazon ECR private registries](https://docs.aws.amazon.com/AmazonECR/latest/userguide/Registries.html).
* Your client must authenticate to Amazon ECR registries as an AWS user with **Authorization token** before it can push and pull images. For more information, see [Private registry authentication](https://docs.aws.amazon.com/AmazonECR/latest/userguide/registry_auth.html).
* An **Amazon ECR image repository** contains your Docker images, Open Container Initiative (OCI) images, and OCI compatible artifacts. For more information, see [Amazon ECR private repositories](https://docs.aws.amazon.com/AmazonECR/latest/userguide/Repositories.html).
* You can control access to your repositories and the images within them with **repository policies**. For more information, see [Repository policies](https://docs.aws.amazon.com/AmazonECR/latest/userguide/repository-policies.html).
* You can push and pull container **images** to your repositories. You can use these images locally on your development system, or you can use them in Amazon ECS task definitions and Amazon EKS pod specifications. For more information, see [Using Amazon ECR images with Amazon ECS](https://docs.aws.amazon.com/AmazonECR/latest/userguide/ECR_on_ECS.html) and [Using Amazon ECR Images with Amazon EKS](https://docs.aws.amazon.com/AmazonECR/latest/userguide/ECR_on_EKS.html).

Amazon ECR Public contains the following components:

* The **Amazon ECR Public Gallery** is the public portal that lists all public repositories hosted on Amazon ECR Public. Visit the Amazon ECR Public Gallery at [https://gallery.ecr.aws](https://gallery.ecr.aws/" \t "_blank). For more information, see [Using the Amazon ECR Public Gallery](https://docs.aws.amazon.com/AmazonECR/latest/public/public-gallery.html).
* A **public registry** is provided to each AWS account; you can create public image repositories in your public registry and store images in them. For more information, see [Amazon ECR public registries](https://docs.aws.amazon.com/AmazonECR/latest/public/public-registries.html).
* Your client must authenticate to a public registry as an AWS user before it can push images to a public repository. For image pulls, Amazon ECR Public accepts both anonymous pulls and pulls using an **authentication token**. For more information, see [Registry authentication](https://docs.aws.amazon.com/AmazonECR/latest/public/public-registries.html" \l "public-registry-auth).
* Repository
* An Amazon ECR image repository contains your Docker images, Open Container Initiative (OCI) images, and OCI compatible artifacts. For more information, see [Amazon ECR public repositories](https://docs.aws.amazon.com/AmazonECR/latest/public/public-repositories.html).
* You can control access to your repositories and the images within them with **repository policies**. For more information, see [Public repository policies](https://docs.aws.amazon.com/AmazonECR/latest/public/public-repository-policies.html).
* You can push and pull container **images** to your repositories. You can use these images locally on your development system, or you can use them in Amazon ECS task definitions and Amazon EKS pod specifications.

## Use cases / Considerations

**Amazon ECR** provides the following features:

* **Lifecycle policies** help with managing the lifecycle of the images in your repositories. You define rules that result in the cleaning up of unused images. You can test rules before applying them to your repository. For more information, see [Lifecycle policies](https://docs.aws.amazon.com/AmazonECR/latest/userguide/LifecyclePolicies.html).
* Image scanning helps in identifying software vulnerabilities in your container images. Each repository can be configured to scan on push. This ensures that each new image pushed to the repository is scanned. You can then retrieve the results of the image scan. For more information, see [Image scanning](https://docs.aws.amazon.com/AmazonECR/latest/userguide/image-scanning.html).
* Cross-Region and cross-account replication makes it easier for you to have your images where you need them. This is configured as a registry setting and is on a per-Region basis. For more information, see [Private registry settings](https://docs.aws.amazon.com/AmazonECR/latest/userguide/registry-settings.html).

**Amazon ECR** is often used as source point for [ECS](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/ecr-repositories.html) and [EKS](https://docs.aws.amazon.com/AmazonECR/latest/userguide/ECR_on_EKS.html) workflows.

It can perfectly suit as a backup solution for your on-premise docker registry.

## Governance

**Cross region replication** aimed to meet backup and disaster recovery requirements for your application. For more information, see [Private image replication.](https://docs.aws.amazon.com/AmazonECR/latest/userguide/replication.html)

For available **metrics** follow [Amazon ECR usage metrics](https://docs.aws.amazon.com/AmazonECR/latest/userguide/monitoring-usage.html) page.

## Cautions

Relevant quotas for **ECR** could be found on the [appropriate Amazon ECR service quotas page](https://docs.aws.amazon.com/AmazonECR/latest/userguide/service-quotas.html).  
**ECR Public** has separate [Amazon ECR Public service quotas page.](https://docs.aws.amazon.com/AmazonECR/latest/public/public-service-quotas.html)

## Pricing considerations

[*Amazon Elastic Container Registry pricing*](https://aws.amazon.com/ecr/pricing/) *(It has section regarding Public repositories as well)*

## More details

[*Containers Roadmap*](https://github.com/aws/containers-roadmap) [*Amazon Elastic Container Registry Public: A New Public Container Registry*](https://aws.amazon.com/blogs/aws/amazon-ecr-public-a-new-public-container-registry/)[*AWS on Air 2020: AWS What’s Next ft. Amazon Elastic Container Registry Public*](https://youtu.be/JMO50AoHHvY)

[*AWS re:Invent 2016: Amazon ECR Deep Dive on Image Optimization (CON401)*](https://youtu.be/Gk_7TR1QAss)

[*Managing Container Images with Amazon ECR - AWS Online Tech Talks*](https://youtu.be/JAlD2fNOPB4)

**Elastic Container Service** Description

## Service Overview

Amazon **Elastic Container Service** (Amazon **ECS**) is a highly scalable, fast container management service that makes it easy to run, stop, and manage containers on a cluster. Your containers are defined in a task definition that you use to run individual tasks or tasks within a service. In this context, a service is a configuration that enables you to run and maintain a specified number of tasks simultaneously in a cluster. You can run your tasks and services on a serverless infrastructure that is managed by AWS Fargate. Alternatively, for more control over your infrastructure, you can run your tasks and services on a cluster of Amazon EC2 instances that you manage.

## Use cases / Considerations

Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast container management service that makes it easy to run, stop, and manage containers on a cluster. Your containers are defined in a task definition that you use to run individual tasks or tasks within a service. In this context, a service is a configuration that enables you to run and maintain a specified number of tasks simultaneously in a cluster. You can run your tasks and services on a serverless infrastructure that is managed by AWS Fargate. Alternatively, for more control over your infrastructure, you can run your tasks and services on a cluster of Amazon EC2 instances that you manage.

Amazon ECS enables you to launch and stop your container-based applications by using simple API calls. You can also retrieve the state of your cluster from a centralized service and have access to many familiar Amazon EC2 features.

You can schedule the placement of your containers across your cluster based on your resource needs, isolation policies, and availability requirements. With Amazon ECS, you don't have to operate your own cluster management and configuration management systems or worry about scaling your management infrastructure.

* [*Amazon ECS: Core Concepts*](https://youtu.be/eq4wL2MiNqo)
* [*Writing Task Definitions for Amazon ECS*](https://youtu.be/o_qSS4S1g34)
* [*Task Placement with Amazon ECS*](https://youtu.be/8XwNPX4AV2M)
* [*Amazon ECS: Load Balancing for Containers*](https://youtu.be/hu7SyJHWJZ0)
* [*Amazon ECS: Autoscaling for Containers*](https://youtu.be/YEvU6uIckDc)

[*Common use cases*](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/common_use_cases.html)

## Governance

[*Available CloudWatch metrics*](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/cloudwatch-metrics.html)

## Cautions

[*Relevant quotas*](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/service-quotas.html)

## Pricing considerations

Pricing for EC2 and Fargate launch model is available on [this page](https://aws.amazon.com/ecs/pricing/)

## More details

[*Containers Roadmap*](https://github.com/aws/containers-roadmap)

[*AWS re:Invent 2020: Getting up and running with Amazon ECS*](https://youtu.be/9u_HKS_Lv6o)

[*AWS re:Invent 2019: [NEW LAUNCH!] Enabling application-first thinking with Amazon ECS (CON325-R1)*](https://youtu.be/v9xuKAdShFw)[*AWS re:Invent 2019: [REPEAT 1] AWS Fargate under the hood (CON423-R1)*](https://youtu.be/Hr-zOaBGyEA)

[*Deep Dive into AWS Fargate*](https://youtu.be/IEvLkwdFgnU)

[*Deep Dive on Amazon Amazon Elastic Container Service (Amazon ECS)*](https://youtu.be/qbEPae8YNbs)

**Amazon Elastic Kubernetes Service (AWS EKS)** Description

## Service Overview

[Amazon Elastic Kubernetes Service (Amazon EKS)](https://docs.aws.amazon.com/eks/latest/userguide/what-is-eks.html) is a managed service that you can use to run Kubernetes on AWS without needing to install, operate, and maintain your own Kubernetes control plane or nodes. Kubernetes is an open-source system for automating the deployment, scaling, and management of containerized applications.

Note: if you are not familiar with Kubernetes(k8s), check out [k8s description](https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/) and [brief overview of components](https://kubernetes.io/docs/concepts/overview/components/).

Your Amazon EKS cluster can schedule pods on any combination of [Self-managed nodes](https://docs.aws.amazon.com/eks/latest/userguide/worker.html), Amazon EKS [Managed node groups](https://docs.aws.amazon.com/eks/latest/userguide/managed-node-groups.html), and [AWS Fargate](https://docs.aws.amazon.com/eks/latest/userguide/fargate.html).

EKS cluster [uses VPC](https://docs.aws.amazon.com/eks/latest/userguide/eks-networking.html) to setup networking in cluster.

## Use cases / Considerations

If your project uses micro-service architecture and relies on AWS, EKS is a good choice for such cases. There is also an option for on-prem clusters with [EKS Anywhere](https://aws.amazon.com/eks/eks-anywhere/).

Several other tools for creating k8s clusters exist like Kops, Kubespray, Kubeadm, however, EKS offers a fully managed experience without the need to worry about setting up everything from scratch.

## Governance

Amazon EKS regularly backs up all managed clusters, and mechanisms exist to recover clusters if necessary. Due to a big amount of k8s resources, I recommend to monitor [control planes](https://docs.aws.amazon.com/eks/latest/userguide/control-plane-logs.html). Here is [a bit more info](https://docs.aws.amazon.com/eks/latest/userguide/logging-monitoring.html) you might want to know.

## Cautions

*By default, EKS cluster is public, which means that anyone can connect to it. Be sure to* [*setup secure access to it*](https://docs.aws.amazon.com/eks/latest/userguide/cluster-endpoint.html)

## Pricing considerations

With AWS EKS, you have to pay for:

1. EC2/Fargate instances deployed
2. Volumes attached to them
3. EKS cluster itself

More info can be found on [AWS EKS pricing page](https://aws.amazon.com/eks/pricing/)

## More details

[*https://www.youtube.com/watch?v=7vxDWDD2YnM*](https://www.youtube.com/watch?v=7vxDWDD2YnM)