



Containers on AWS



At the core of the lesson

You will learn how to do the following:

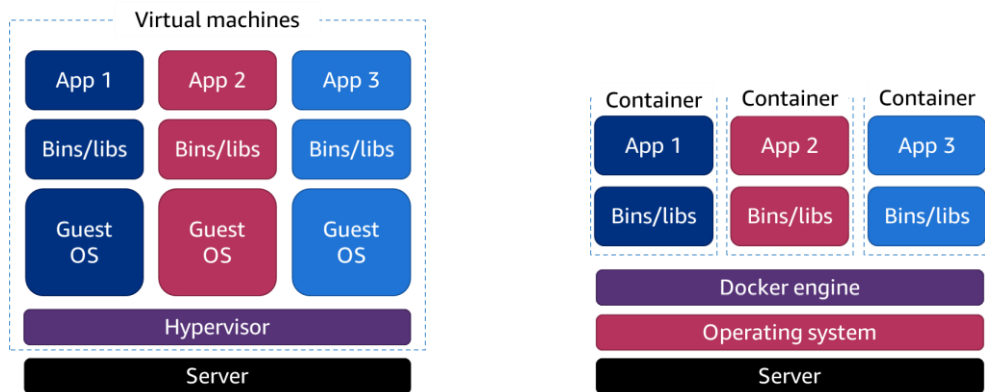
- Explain the purpose of containers and their benefits.
- Define Docker, its benefits, and its components.
- Identify container-related services on Amazon Web Services (AWS).



Containers

Problem

How can you simplify portability of applications between environments?



Prior to the introduction of containers, developers and administrators were often faced with a challenging web of complex compatibility restrictions in which an application or workload was built specifically for its predetermined environment. If this workload needed to be migrated—for example, from bare metal to a virtual machine (VM), from a VM to the cloud, or between service providers—migrating the workload generally meant rebuilding the application or workload entirely to help ensure compatibility in the new environment.

What is a container?



Containers

A container is an application and its dependencies, which can be run in resource-isolated processes.



The capabilities of containers can intuitively sound similar to a VM; however, the differences are in the details.

VMs are isolated but do not share the same operating system (OS) and bins/libraries.

Containers are isolated but share an OS and, where appropriate, bins/libraries.

The main difference is the lack of a hypervisor requirement. Containers can run on any Linux system with appropriate kernel-feature support and the Docker daemon present. This ability makes containers portable. Your laptop, your VM, your Amazon Elastic Compute Cloud (Amazon EC2) instance, and your bare metal server are all potential hosts. The lack of a hypervisor requirement also results in almost no noticeable performance overhead. The processes are communicating directly to the kernel and are largely unaware of their container silo. Most containers boot in only a couple of seconds.

Benefits of containers

- Environmental consistency
- Process isolation
- Operational efficiency
- Developer productivity
- Version control



Container 1



Container 2



Container 3



Containers deliver environmental consistency because the application's code, configurations, and dependencies are packaged into a single object. Containers serve as a building block that can be deployed on any compute resource regardless of software, OS, or hardware configurations.

Containers also provide process isolation:

- They have no shared dependencies or incompatibilities because each container is isolated from the other.
- Whatever you package as a container locally will deploy and run the same way whether in testing or production.
- Process isolation provides operational efficiency.

With containers, you gain operational efficiency because you can run multiple applications on the same instance:

- You can specify the exact amount of memory, disk space, and CPU that a container will use on an instance.
- Containers boot quickly, and because of the reduced footprint, you can create and end applications or tasks that are encapsulated in a container. By using these features, you can scale applications up and down rapidly.

Containers increase developer productivity by removing cross-service dependencies and conflicts:

- Each application component can be broken into different containers that run a different microservice.
- Containers are isolated from each other, so you do not have to worry about synchronizing libraries or dependencies for each service.
- Developers can independently upgrade each service because the libraries have no conflicts.

With containers, you can track versions of your application code and their dependencies. Docker container images have a manifest file (Dockerfile) that gives you the ability to do the following:

- Maintain and track versions of a container.
- Inspect differences between versions.
- Roll back to previous versions.



Docker

What is Docker?



- Docker is an application platform used to create, manage, and run containers.
- With Docker, developers and engineers can build, test, deploy, and run containers.



©2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

8

Docker is a software platform that packages software (such as applications) into containers. With Docker, developers and engineers can build, test, and deploy applications. The containers hold everything that the software needs to run. Docker is installed on each server that will host containers, and it provides commands that you can use to build, start, or stop containers.

Docker containers include only the things an application needs to run so that applications can be moved between environments.

Docker is best used as a solution when you want to do the following:

- Standardize environments.
- Reduce conflicts between language stacks and versions.
- Use containers as a service.
- Run microservices by using standardized code deployments.
- Require portability for data processing.

Benefits of Docker

- Microservices architecture
- Stateless
- Portable
- Single, immutable artifact
- Reliable deployments



The many benefits that containers provide fuel the rapid growth of Linux containers, and Docker specifically. These benefits are apparent across an organization, from developers and operations to quality assurance. The primary benefits of Docker are speed, consistency, density, and flexibility.

- **Microservices architecture:** Public documentation on Docker recommends that you run one service per container. For example, if your application has five different processes, you should run five different containers. Single-process containers provide the advantage of more granular updates. If you want to update only your web server, there is no need to shut down the database process. This practice leads to an efficient architecture for building microservices-based applications.
- **Stateless:** Dockers are considered to be stateless. They consist of read-only layers. This means that after the container image has been created, it does not change.
- **Portable:** With Docker, your application is independent from the configurations of low-level resources, such as networking, storage, and OS details. This feature provides portability. For example, if your application runs in a Docker container, it will run anywhere.
- **Single, immutable artifact:** Docker also assists with packaging your applications and dependencies in a single, immutable artifact.
- **Reliable deployments:** Oftentimes, developers can get code running on their own machines, but it doesn't run properly when deployed to the server. This issue occurs because the server environment could be different, requiring the developer to spend an enormous amount of time debugging the difference and fixing the issue. Developers can set up a local development environment that matches a production server. When a developer finishes writing and testing code, they can wrap it in a container and publish it directly to the cloud, and it will instantly work because the environment is the same.

Components of Docker



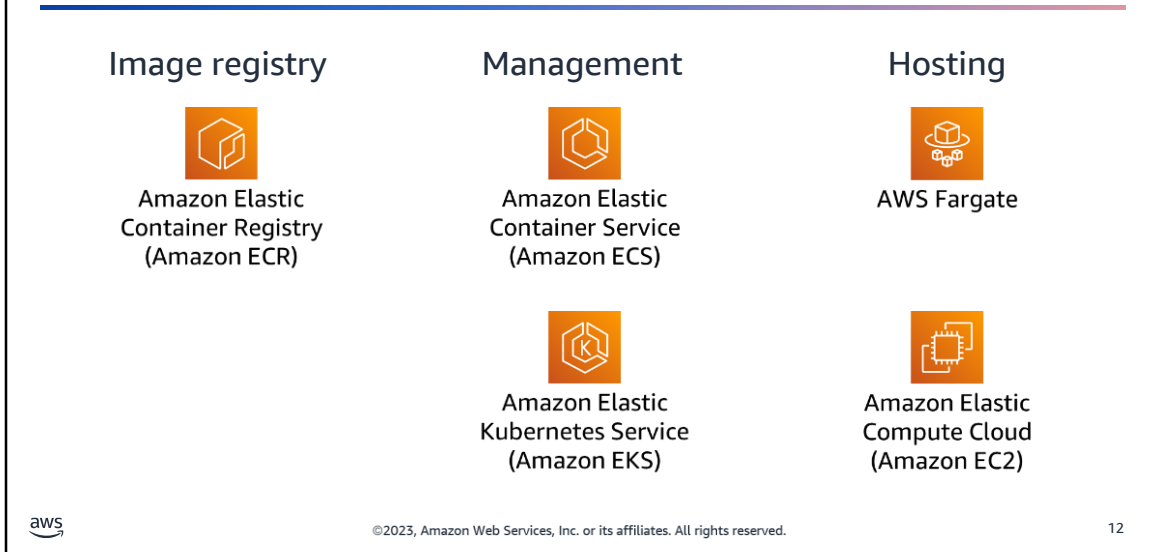
The following are key components of Docker:

- Docker file: The blueprint used to build a Docker image
- Image: A file for Docker containers at runtime
- Registry: The version control system for Docker images (a centralized image repository)
- Container: A runnable instance of an image
- Host: A machine that runs, or hosts, containers



AWS container services

Container services on AWS



The current AWS container services landscape covers a broad set of products.

AWS provides a registry service, Amazon Elastic Container Registry (Amazon ECR), where you can store your container images.

Management is the deployment, scheduling, and scaling of containerized applications. AWS services are Amazon Elastic Container Service (Amazon ECS) and Amazon Elastic Kubernetes Service (Amazon EKS). Amazon ECS provisions new application container instances and compute resources. Use Amazon EKS to deploy, manage, and scale containerized applications using Kubernetes on AWS.

Hosting is where the containers run. You can currently run your containers on Amazon ECS using the Amazon EC2 launch type (where you get to manage the underlying instances on which your containers run), or you can choose to run your containers in a serverless manner with the AWS Fargate launch type.

Amazon ECR



Amazon ECR

A fully managed Docker container registry that developers can use to store, manage, and deploy Docker container images




©2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

13

Amazon ECR is a fully managed Docker container registry. With Amazon ECR, developers can store, manage, and deploy Docker container images. Amazon ECR is integrated with Amazon ECS so that you can store, run, and manage container images for applications that run on Amazon ECS. Specify the Amazon ECR repository in your task definition, and Amazon ECS will retrieve the appropriate images for your applications.


- Amazon ECR supports Docker Registry HTTP API Version 2, which gives you the ability to interact with Amazon ECR by using Docker CLI commands or your preferred Docker tools. In this way, you can maintain your existing development workflow. You can access Amazon ECR from any Docker environment whether its in the cloud, on premises, or on your local machine.
- Amazon ECR stores your container images in Amazon Simple Storage Service (Amazon S3) so it benefits from the high availability and durability of Amazon S3.
- You can define and organize repositories in your Amazon ECR registry by using namespaces. As a result, you can organize your repositories based on your team's existing workflows. You can also set the API actions that another user can perform on your repository. In this way, you can share your repositories with different users and AWS accounts.
- Amazon ECR uses AWS Identity and Access Management (IAM) for access control. With IAM, you can control who and what (for example, EC2 instances) can access your container images through defined policies.
- You can transfer your container images to and from Amazon ECS via HTTPS. Your images are also automatically encrypted at rest by using Amazon S3 server-side encryption.
- Amazon ECR also supports third-party integrations.

Amazon ECS



Amazon ECS

A highly scalable, high-performance container management service that supports Docker containers



©2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

14

Amazon ECS will provision new application container instances and compute resources based on scaling policies and Amazon ECS configurations. Infrastructure resources, such as load balancers, will need to be created outside of Amazon ECS.

With Amazon ECS, you can run applications on a managed cluster of EC2 instances. It provides flexible scheduling. Amazon ECS uses a built-in scheduler or it uses a third-party scheduler, such as Apache Mesos. You can also perform task, service, or daemon scheduling.

The Amazon ECS APIs make it straightforward to integrate third-party solutions—such as schedulers—or support your software delivery process.

Amazon ECS launches your containers in your own virtual private cloud (VPC) so that you can use your VPC security groups and network access control lists (network ACLs).


- No compute resources are shared with other customers.
- You can assign granular access permissions for each of your containers. You can use IAM to restrict access to each service and to configure which resources a container can access.
- These configurable isolation features of Amazon ECS are designed to help you build secure and reliable applications.

Amazon ECS is built on technology that was developed from many years of experience running highly scalable services. You can use Amazon ECS to launch tens (or tens of thousands) of Docker containers in seconds with no additional complexity. Amazon ECS tasks are defined through a declarative JSON template that is called a task definition. With this template, you can specify one or more containers that are required for your task, including the following:


- Docker repository and image
- Memory and CPU requirements
- Shared data volumes
- Relationships between containers

You can launch as many tasks as you want from a single task definition, which you register with the service.

Amazon EKS



A managed service that you can use to run Kubernetes on AWS without needing to install and operate your own Kubernetes clusters



©2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

15

Kubernetes is an open-source system for automating the deployment, scaling, and management of containerized applications. To use Kubernetes, you must have a cluster to run your applications.

With Amazon EKS, AWS manages upgrades and high availability services for you. Amazon EKS runs three Kubernetes managers across three Availability Zones to provide high availability. Amazon EKS automatically detects and replaces unhealthy managers and provides automated version upgrades and patching for the managers. Amazon EKS is also integrated with many AWS services and features to provide scalability and security for your applications, including the following:

- Elastic Load Balancing (ELB) for load distribution
- IAM for authentication
- Amazon Virtual Private Cloud (Amazon VPC) for isolation
- AWS PrivateLink for private network access
- AWS CloudTrail for logging


Amazon EKS runs the latest version of the open-source Kubernetes software, so you can use all of the existing plugins and tooling from the Kubernetes community. Applications running on Amazon EKS are fully compatible with applications running on any standard Kubernetes environment, whether they are running in on-premises data centers or public clouds.

You can use Amazon EKS to install, manage, and update common operational software for your cluster directly through the Amazon EKS console, AWS Command Line Interface (AWS CLI), and API. AWS validates all operational software, which you can deploy and update during cluster setup or at any time.

Operating Kubernetes for production applications presents a number of challenges:


- You must manage the scaling and availability of your Kubernetes control plane by ensuring that you have chosen appropriate instance types, run them across multiple Availability Zones, monitored their health, and replaced unhealthy nodes.
- You need to patch and upgrade your control plane and nodes to ensure that you run the latest version of Kubernetes. This patching and upgrading require expertise and a lot of manual work.

AWS Fargate



AWS Fargate

A compute engine for Amazon ECS that you can use to run containers without needing to manage servers or clusters

 ©2023, Amazon Web Services, Inc. or its affiliates. All rights reserved. 16

AWS Fargate is a serverless compute engine for containers.

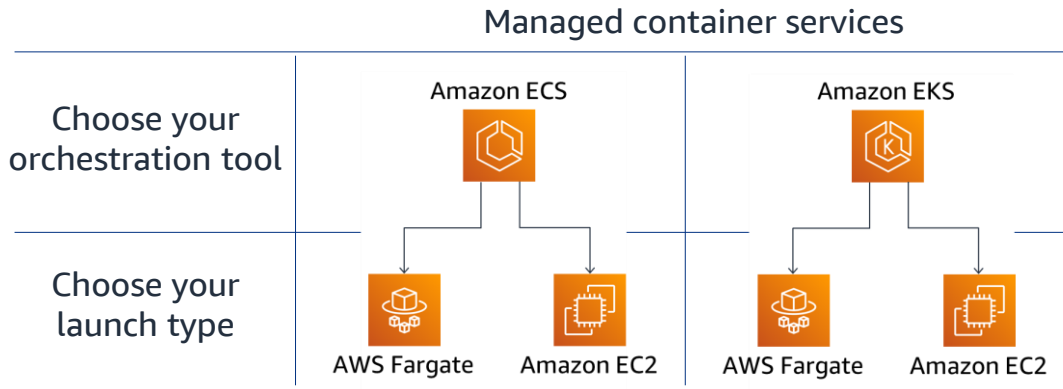
Fargate makes running containers the focus for developers. Whether you chose to use Amazon ECS or run your own orchestration software, such as Kubernetes on Amazon EC2, you previously had to worry about provisioning and scaling a cluster (or servers) to meet your application's needs. You also had to manage the patching and software updates for the servers.

You no longer have to provision, configure, and scale clusters of virtual machines to run containers. AWS Fargate is a compute engine for Amazon ECS and Amazon EKS that gives you the ability to run containers without having to manage servers or clusters.

With Fargate, you gain the following benefits:

- You no longer have to provision and scale server clusters, or patch and update each server.
- Fargate supports both Amazon ECS and Amazon EKS, so you can choose the orchestration solution that you prefer to manage and run your containers. You define your application as you do today for Amazon ECS or Kubernetes and the resources that your application needs, and Fargate seamlessly handles the underlying infrastructure needs.
- Pay for only the resources that you use. You pay for those resources on a per-second usage basis.
- With Amazon EKS, you can forward container logs from pods running on Fargate to AWS services for log storage and analytics, including Amazon CloudWatch, Amazon OpenSearch Service (successor to Amazon Elasticsearch Service), Amazon Kinesis Data Firehose, and Amazon Kinesis Data Streams.
- With Fargate, you can focus on designing and running the application—not the infrastructure.

Deploying to AWS



©2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

17

Deploying your managed container solutions on AWS involves selecting an orchestration tool and a launch type.

Amazon ECS is a fully managed container orchestration service that provides the most secure, reliable, and scalable way to run containerized applications.

Amazon EKS is a fully managed Kubernetes service that provides the most secure, reliable, and scalable way to run containerized applications using Kubernetes.

Checkpoint questions

1. What is the difference between a VM and a container?
2. What is Docker?
3. What are two orchestration tools used to manage container services?



The answers to the questions are as follows:

1. What is the difference between a VM and a container?

A VM virtualizes hardware, and a container is a virtualized OS. Containers are smaller and do not contain an entire OS. Instead, containers share a virtualized OS, and they run as resource-isolated processes, which helps ensure quick, reliable, and consistent deployments.

2. What is Docker?

Docker is a software platform that packages software (such as applications) into units that are called containers.

3. What are two orchestration tools used to manage container services?

Amazon ECS and Amazon EKS

Key ideas



- Containers provide a common interface for migrating applications between environments.
- Docker is a software platform that packages software (such as applications) into containers.
- Container services on AWS include Amazon ECR, Amazon ECS, Amazon EKS, and AWS Fargate.



Thank you

Corrections, feedback, or other questions?
Contact us at <https://support.aws.amazon.com/#/contacts/aws-training>.
All trademarks are the property of their owners.

©2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

20