Introducing Containers and Container Services

CONTAINERS VS VMS · DOCKER IMAGES · MICROSERVICES · ECS · EKS · ECR · FARGATE · ELASTIC BEANSTALK STUDY & REVIEW

01 MUST KNOW

if you remember five things, remember these

01 **A container is a standardized unit of software:** it packages everything an application needs to run — **application code, a runtime engine, system tools, system libraries, and settings** — into one lightweight, standalone unit. Containers provide **OS virtualization**, running the app in resource-isolated processes, and a single host can run many containers that all **share the host OS kernel**.

02 **Containers are not lightweight VMs:** deployment evolved bare-metal → VMs → containers. A **VM** runs a full guest OS on a hypervisor (heavy, redundant, slow to boot); a **container** shares the host's kernel, so it is lightweight, starts and stops faster, and runs *identically* across environments from development to production.

03 **Dockerfile → image → container:** a **Dockerfile** is a plain-text file whose instructions each add a **read-only layer** to build an immutable **image**; a **container** is a runnable instance of an image with a thin **read/write layer**. A **registry** stores images. Editing the Dockerfile rebuilds only the changed layers — and a container is a compute unit, *not* a place to store data.

04 **AWS gives you three container services: Amazon ECS** (fully managed orchestration for Docker), **Amazon EKS** (managed open-source Kubernetes), and **Amazon ECR** (fully managed image registry). Orchestration handles **scheduling, placement, and service integration** so you can run thousands of containers across a cluster.

05 **Fargate vs. EC2, and Beanstalk on top:** ECS and EKS run tasks on two launch types — **AWS Fargate** (near-serverless; AWS manages the infrastructure) or **Amazon EC2** (you manage the cluster for more control). **Elastic Beanstalk** is the quickest on-ramp: hand it a `Dockerrun.aws.json` and code, and it stands up a multicontainer Docker environment using ECS and AWS CloudFormation for you.

02 KEY TERMS

TERM	DEFINITION
Container	A standardized, lightweight unit of software that packages an application with everything it needs to run and shares the host OS kernel
Abstraction	Hiding the working details of a subsystem to separate concerns — the idea (borrowed from shipping) that increases agility, interoperability, and platform independence
Docker	A lightweight container virtualization platform with tooling to create, store, manage, and run containers, and to integrate with build, test, and deployment pipelines
Dockerfile	A plain-text file of instructions used to build a container image; each instruction creates a read-only layer
Container image	A read-only, immutable, layered template used to create writable containers; portable to any environment that supports Docker
Container (instance)	A runnable instance of an image, given a thin read/write layer when instantiated; a discrete compute unit, not storage
Container registry	A repository of public or private images that you can pull and base other images on (for example, Amazon ECR)
Microservices	An architecture of small, independent components that each perform a single function and communicate through lightweight API operations
Container orchestration	Automated scheduling and placement of containers across infrastructure, plus integration with networking, storage, security, monitoring, and logging
Amazon ECS	Amazon Elastic Container Service — a fully managed container orchestration service for Docker containers

TERM	DEFINITION
Amazon ECR	Amazon Elastic Container Registry — a fully managed Docker image registry integrated with ECS and the Docker CLI
Amazon EKS	Amazon Elastic Kubernetes Service — a managed service that runs open-source Kubernetes on AWS across multiple Availability Zones

03 VIRTUAL MACHINES VS. CONTAINERS

same goal, very different weight

VIRTUAL MACHINE

virtualizes the hardware

- Hypervisor runs a **full guest OS** for each VM
- Strong isolation; better utilization than bare-metal
- Heavy — redundant OSes mean more patching and disk
- Boots slower; larger footprint

CONTAINER

virtualizes the OS

- Shares the **host OS kernel**; isolated processes
- Lightweight, efficient, fast to start and stop
- Share or isolate libraries per application
- Highly portable — runs identically dev → production

04 AWS CONTAINER SERVICES AT A GLANCE

what each one is for

SERVICE	CATEGORY	WHAT IT DOES
Amazon ECS	Orchestration	Fully managed orchestration for Docker; schedules placement across a managed cluster; integrates with IAM, CloudWatch, and Route 53
Amazon EKS	Orchestration	Managed open-source **Kubernetes** across multiple Availability Zones; conformant, compatible, and secure by default
Amazon ECR	Registry	Fully managed Docker image registry; store, manage, and deploy images; HTTPS transfer, encrypted at rest, IAM-controlled
AWS Fargate	Launch type	Near-serverless compute for ECS/EKS tasks — AWS manages the infrastructure, placement, CPU, and memory

05 ECS LAUNCH TYPES: FARGATE OR EC2

who manages the infrastructure

CONSIDERATION	AWS FARGATE	AMAZON EC2
Who manages infra	AWS — near-serverless; no cluster to run	You create and manage the EC2 cluster and placement
Best for	Wide demand swings; low-overhead or small test environments; scheduled batch jobs	Predictable resource needs; price-optimized large workloads; excess EC2 capacity
Control	Focus on the tasks and application architecture	Granular control for security/compliance and reserved-instance savings

06 IMAGE LIFECYCLE: BUILD → PUSH → RUN

`01` Author a **Dockerfile** (each instruction = a read-only layer) → `02` `docker build` → an immutable **image** → `03` `docker tag` for your registry → `04` `docker push` to **Amazon ECR** → `05` Orchestrator (**ECS**/**EKS**) **pulls** the image → `06` `docker run` → a running **container**

07 EXAM TRAPS

where the knowledge check gets you

- × “A container is just a lightweight VM.” — No. A VM runs a **full guest OS** on a hypervisor; a container **shares the host OS kernel** and isolates only processes. That is why containers start faster and use far less space.
- × “Store the application's data inside the container.” — The container's read/write layer is **ephemeral** and not for long-term storage. A container is a discrete **compute** unit — persist data in a **volume**.
- × “Editing one line of a Dockerfile rebuilds the entire image.” — Only the **changed layers** (and those above them) rebuild; unchanged read-only layers are reused. That is what keeps images small and fast.
- × “Container images are editable in place.” — Images are **read-only and immutable**; the writable layer belongs to the running **container**, not the image.
- × “Amazon ECR runs my containers.” — ECR is a **registry** that stores images. **Amazon ECS** and **Amazon EKS** are the orchestration services that actually run them.
- × “Amazon EKS is a proprietary AWS orchestrator.” — EKS runs **open-source Kubernetes** (certified conformant and compatible). **Amazon ECS** is the AWS-native orchestrator.
- × “With AWS Fargate you still manage the EC2 cluster.” — Fargate is **near-serverless**: AWS manages the infrastructure. Choose the **EC2 launch type** when you need control over the instances.
- × “Elastic Beanstalk means configuring Amazon ECS yourself.” — Beanstalk uses **ECS** (and **CloudFormation**) under the hood; you provide a `Dockerrun.aws.json` and code, and it creates the cluster, task definition, and tasks for you.

08 SELF-QUIZ

answers are upside-down below

Q1 What does a container share with its host that a virtual machine does not?

Q2 Name the components a container packages so an application runs the same anywhere.

Q3 What plain-text file builds a container image, and what does each of its instructions create?

Q4 An image is a read-only template — what is the running thing created from it called?

Q5 Where are container images stored so that ECS or other hosts can pull them?

Q6 Which AWS service is a fully managed container orchestration service, and what two launch types can run its tasks?

Q7 Which AWS service runs open-source Kubernetes on AWS?

Q8 Which ECS launch type gives a near-serverless experience in which AWS manages the infrastructure?

Q9 Which Elastic Beanstalk deployment policy routes a portion of traffic to a full set of new instances to support canary testing?

Q10 A team new to AWS and containers wants to get started quickly to experiment with a web app on containers — which AWS solution fits best?

ANSWER KEY (rotate the page)

Q1 the host operating system's kernel **Q2** application code, a runtime engine, system tools, system libraries, and settings **Q3** a Dockerfile; each instruction creates a read-only layer **Q4** a container (a runnable instance of the image, with a thin read/write layer) **Q5** in a container registry, such as Amazon ECR **Q6** Amazon ECS; the AWS Fargate and Amazon EC2 launch types **Q7** Amazon EKS **Q8** AWS Fargate **Q9** traffic splitting **Q10** use Elastic Beanstalk to launch a multicontainer Docker environment