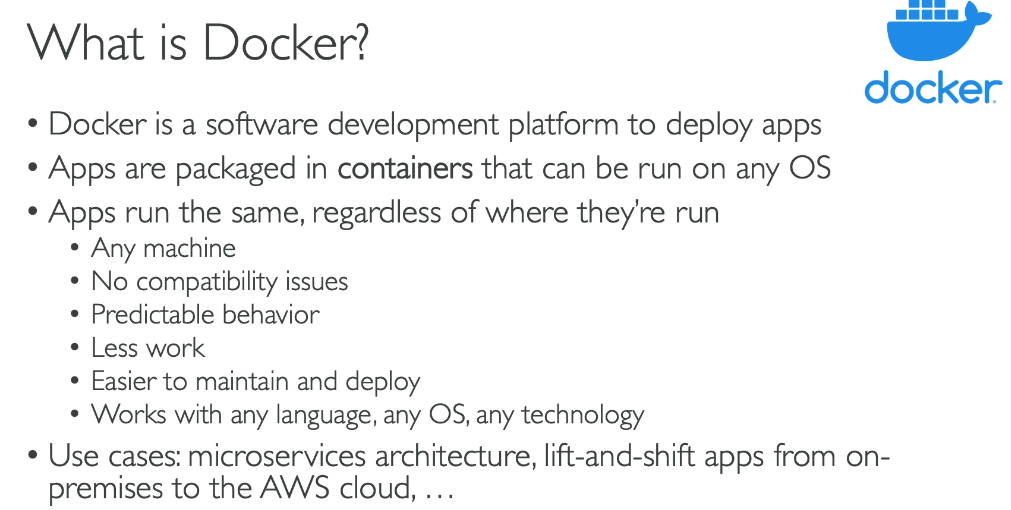
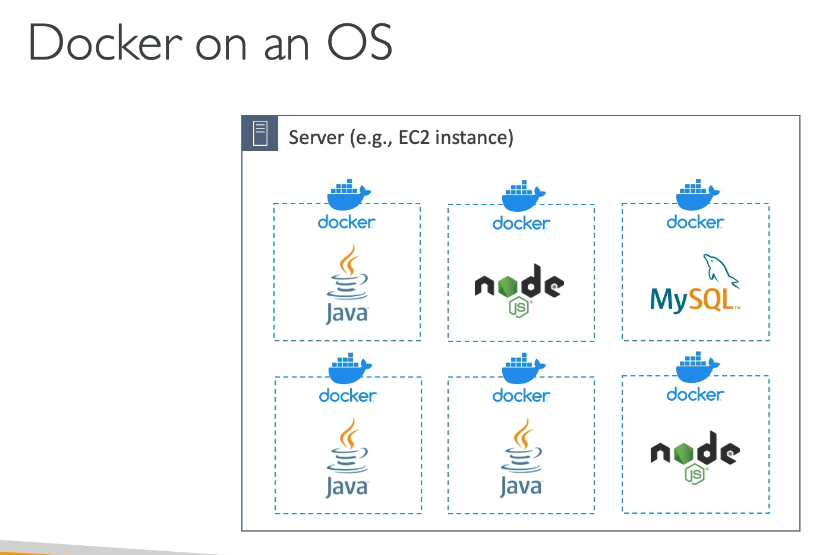
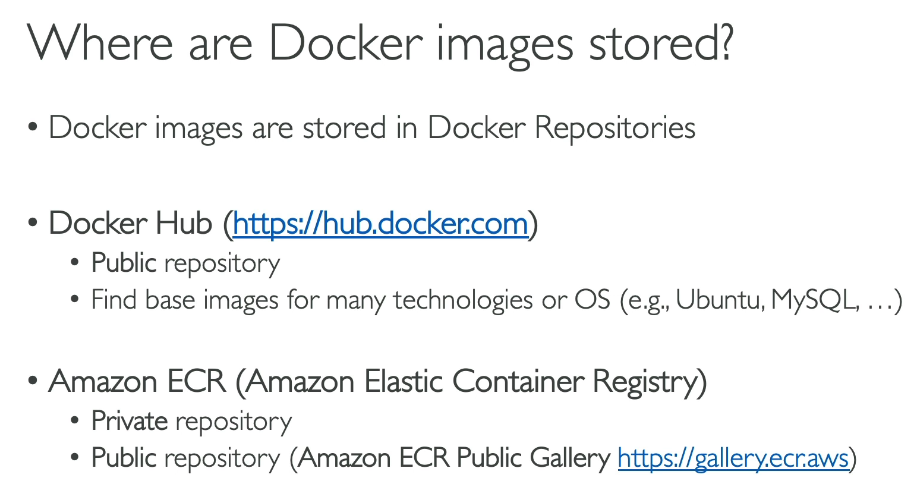
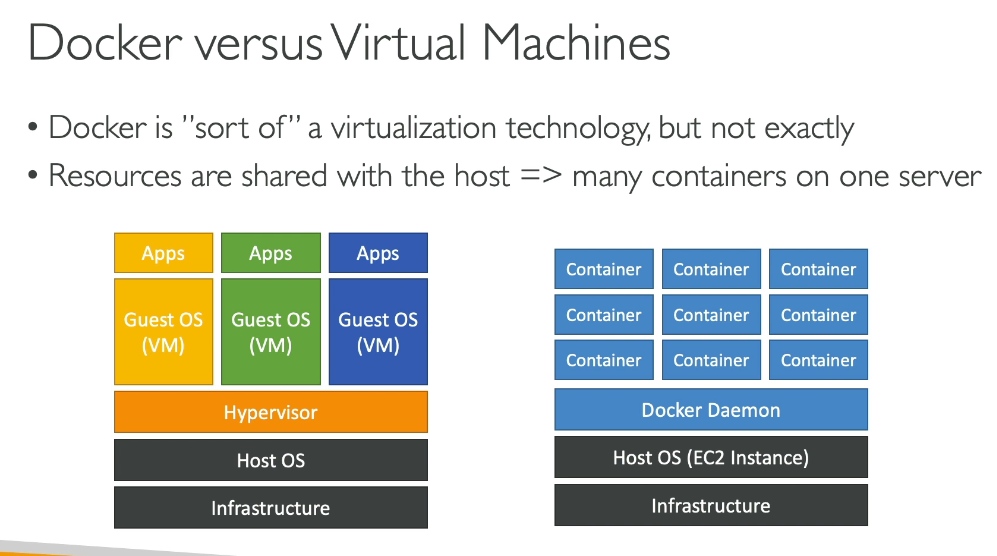
**ECS, EKS, ECR, APP RUNNER & FARGATE**

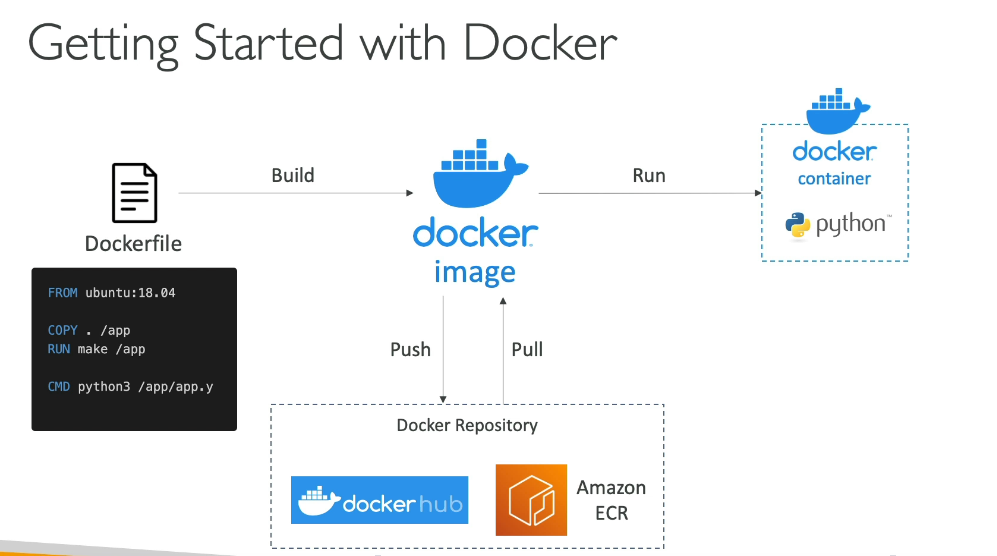
**Docker**:





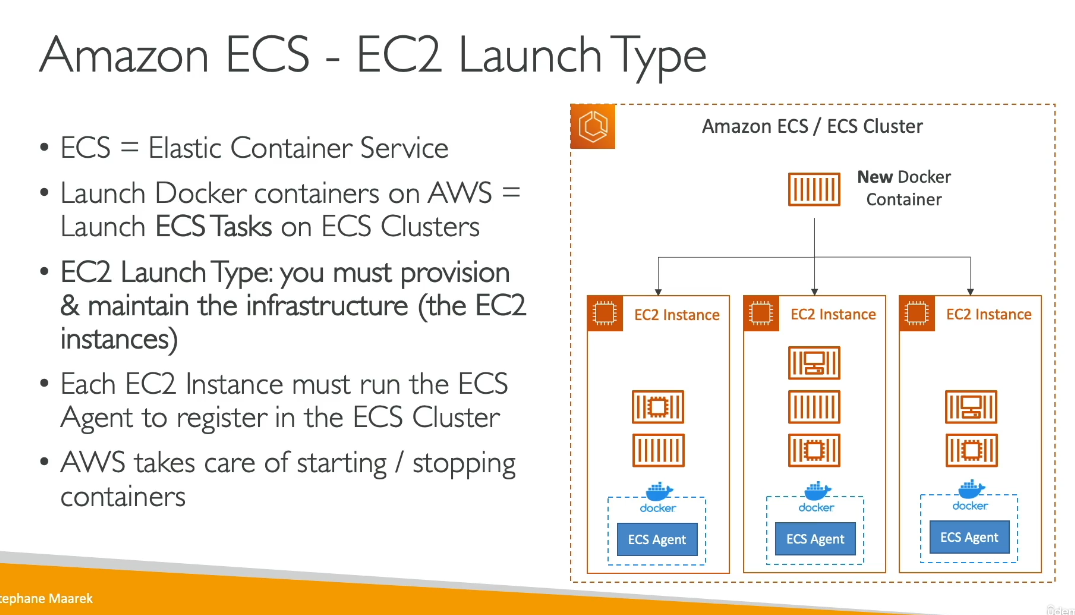




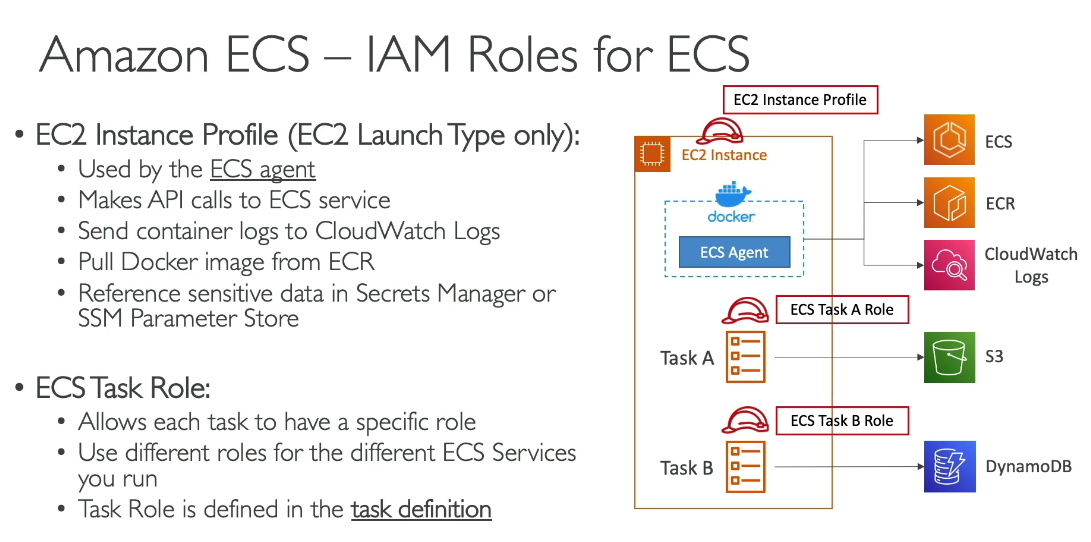


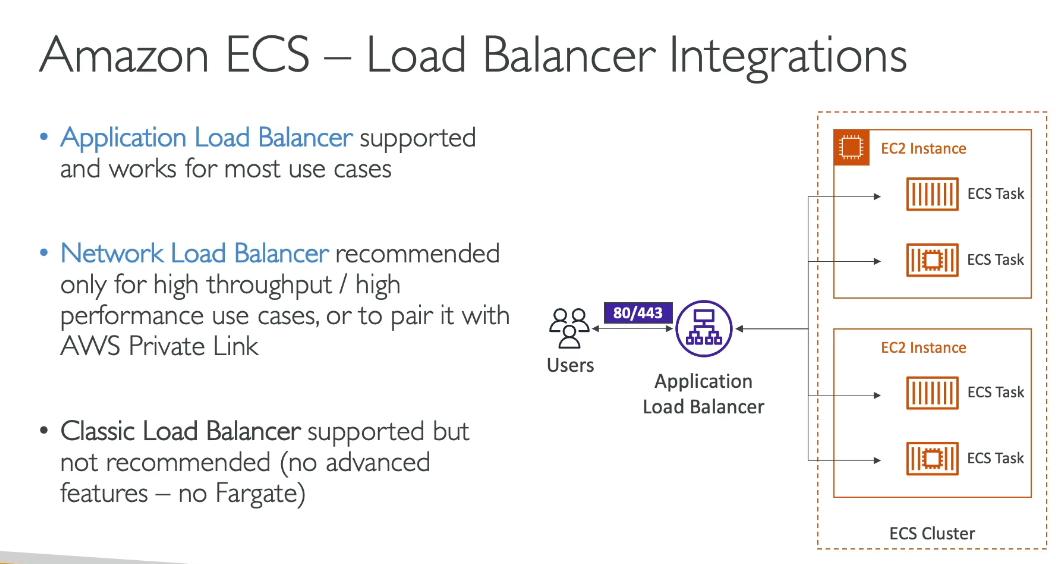


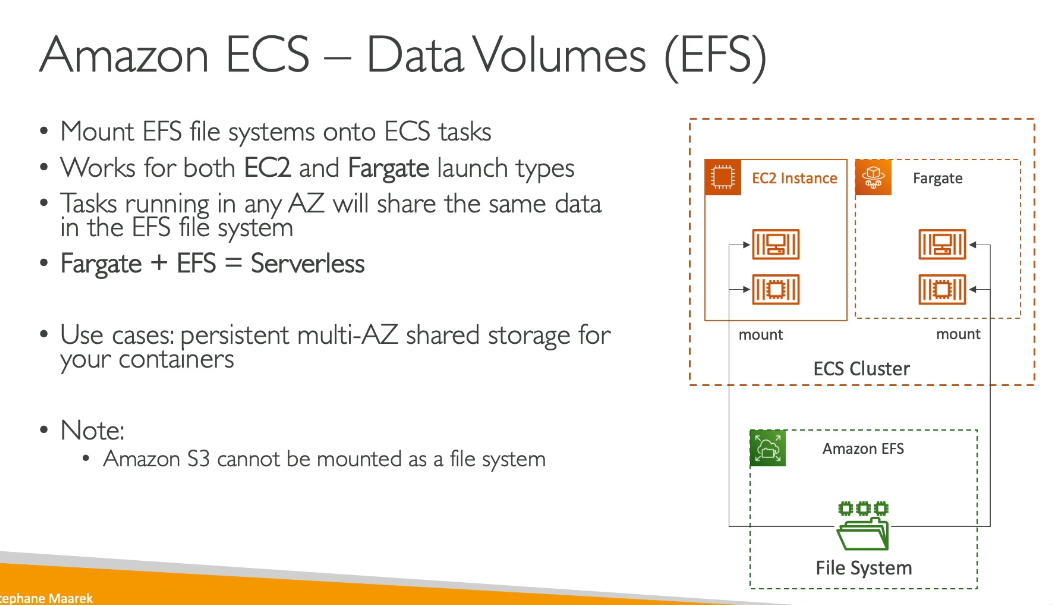
**ECS**









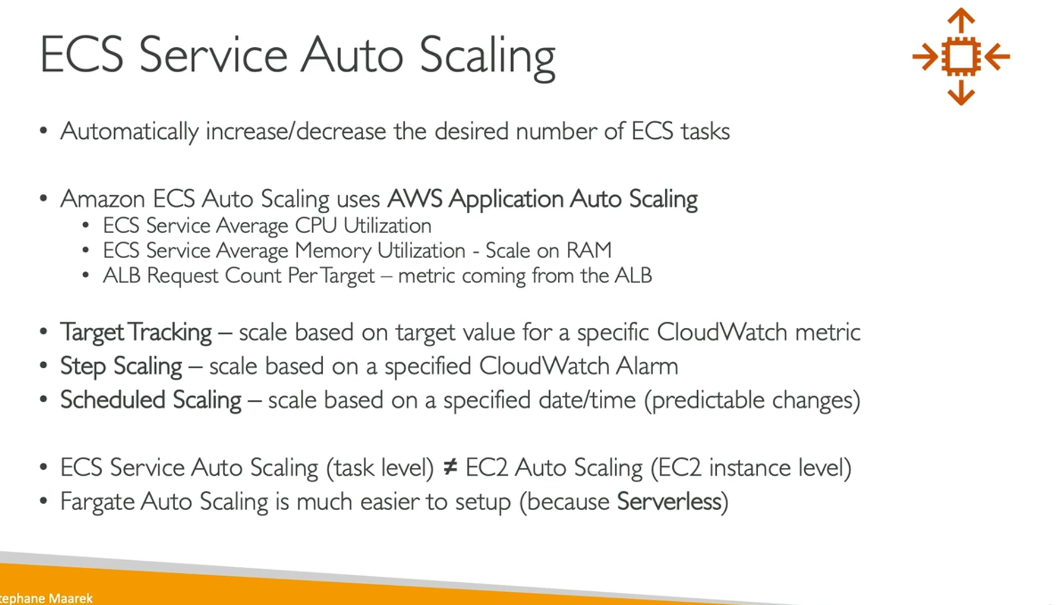


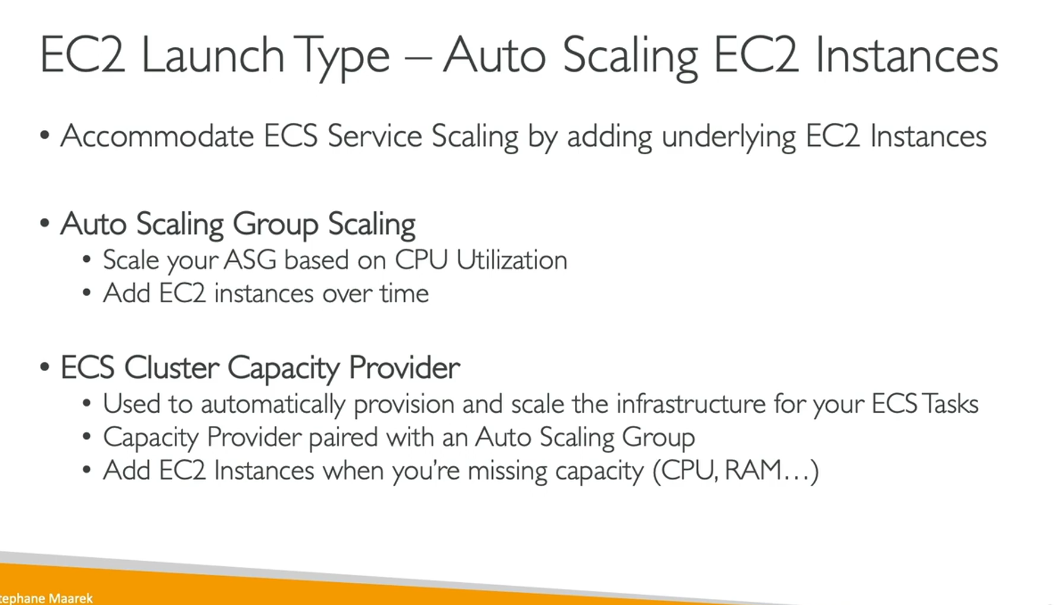
**Cluster** A logical grouping of tasks or services that provides infrastructure capacity for your containerized applications.

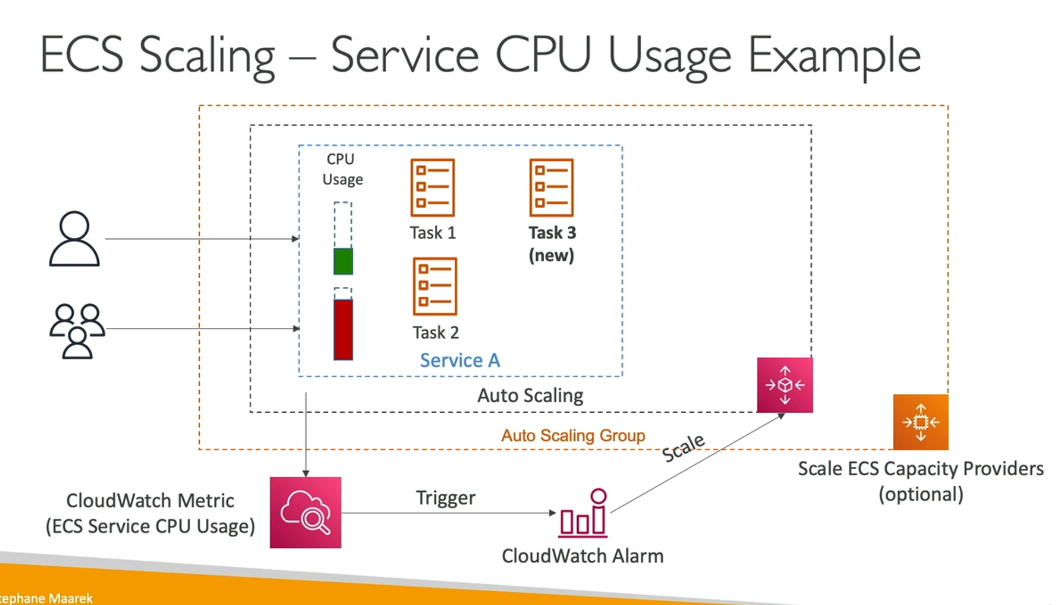
**Task Definition** A blueprint that describes how to run your containers, specifying container images, CPU/memory requirements, port mappings, environment variables, and data volumes.

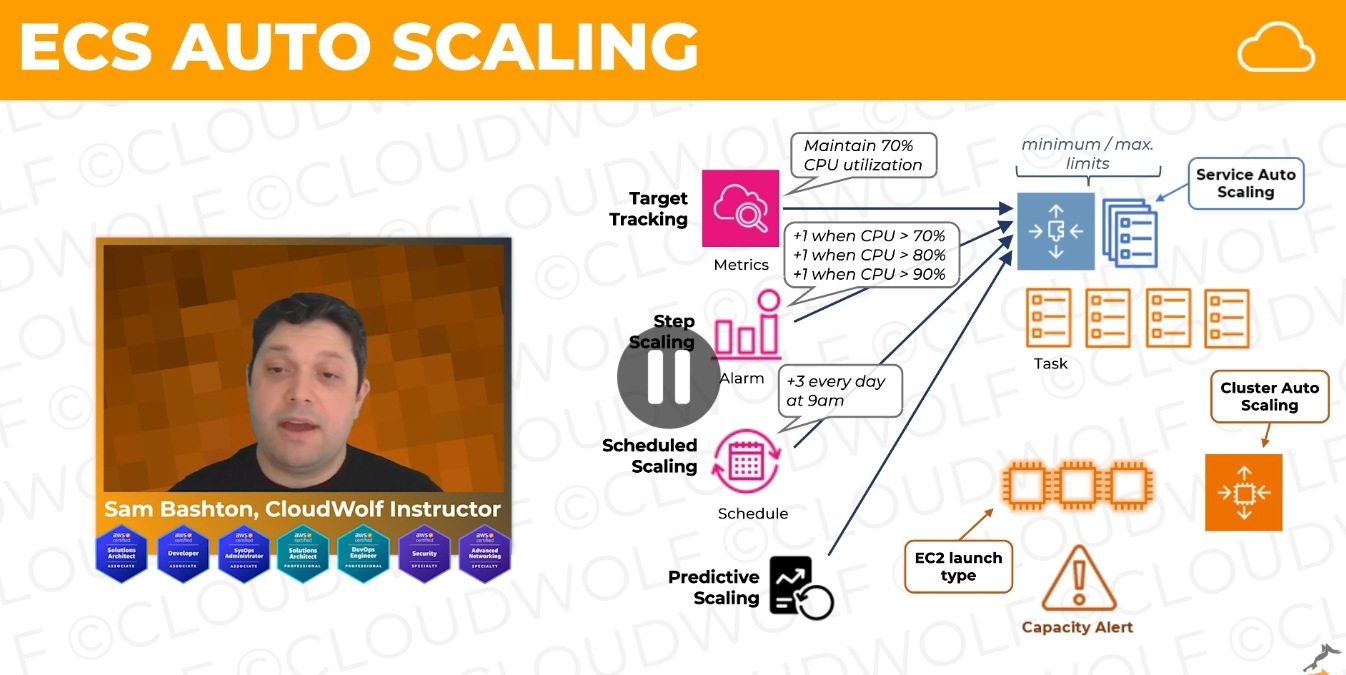
**Task** The instantiation of a task definition within a cluster - a running instance of your containers based on the task definition.

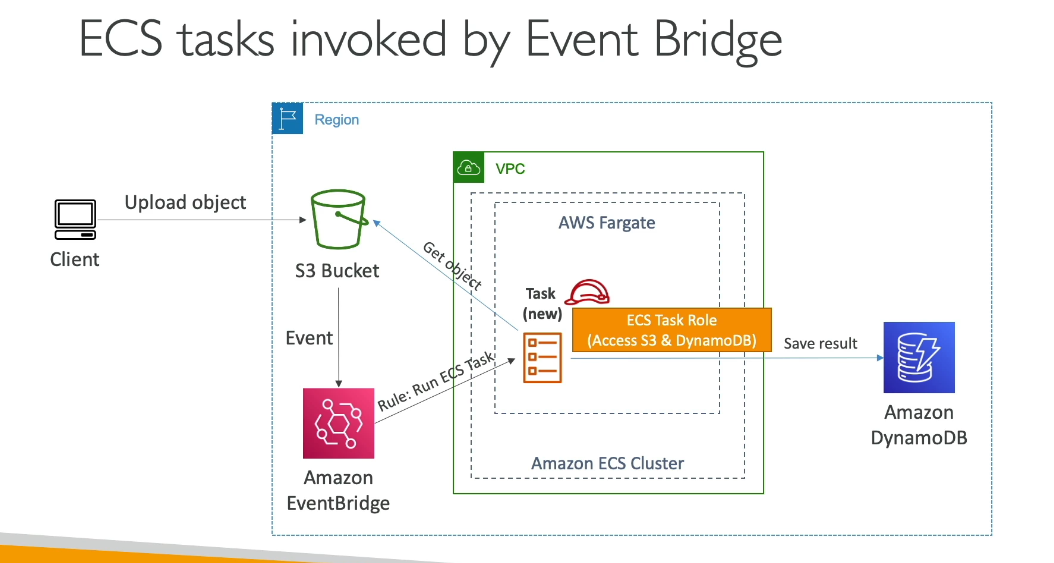
**Service** A resource that runs and maintains a specified number of tasks si multaneously in a cluster, automatically replacing failed tasks to maintain the desired count.

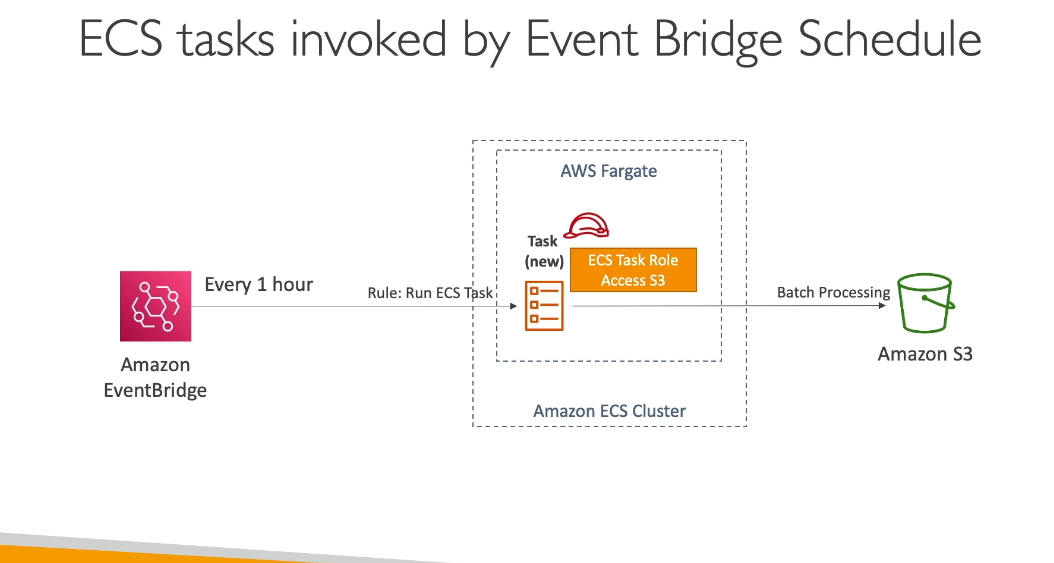


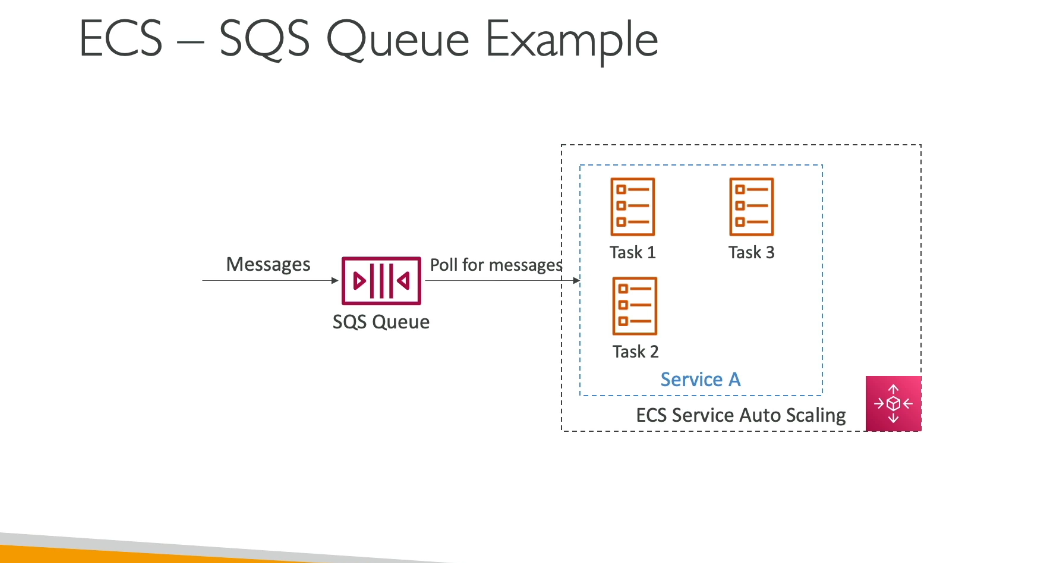


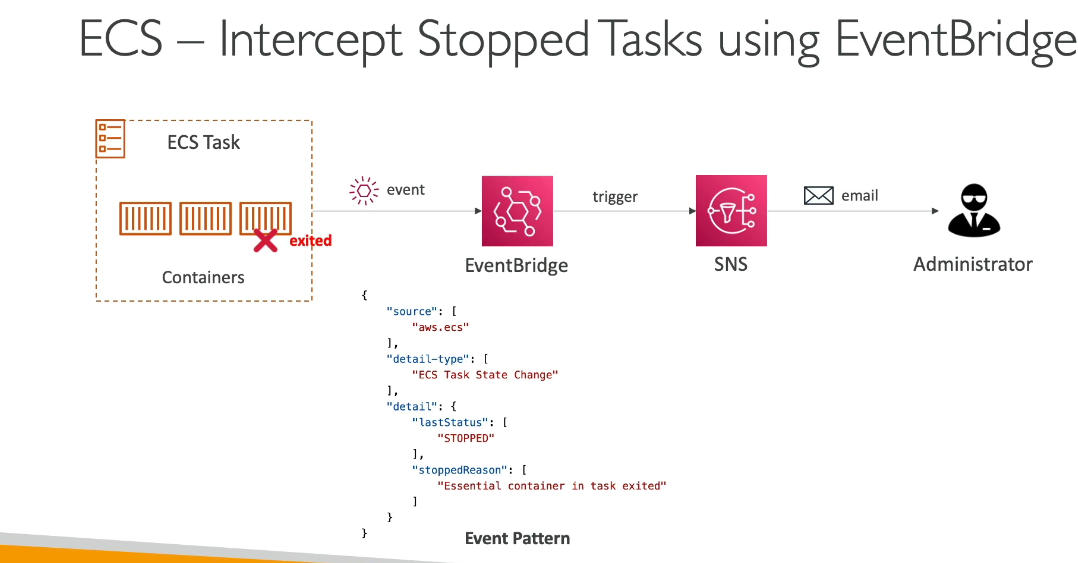




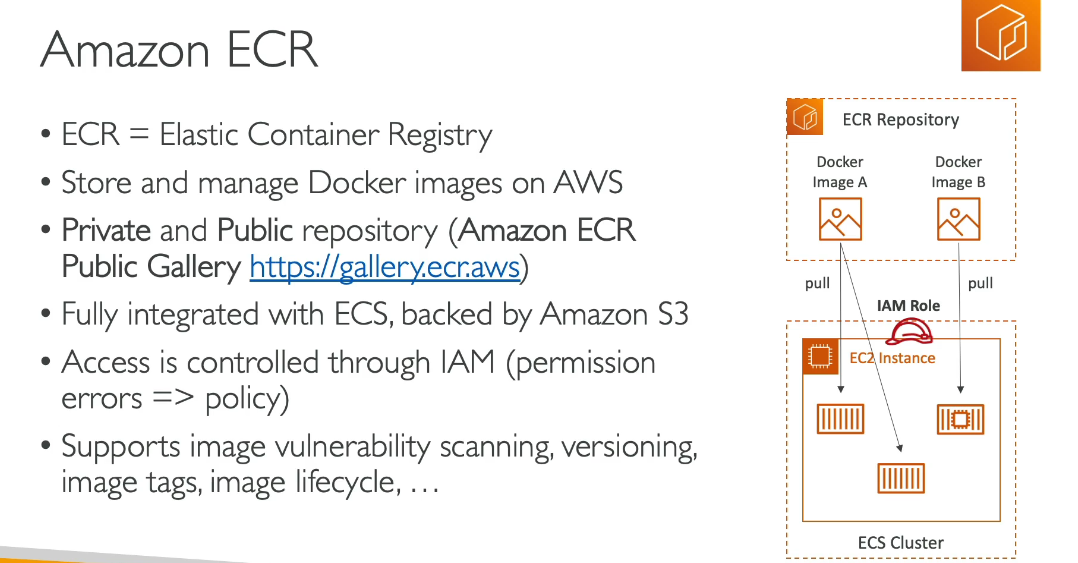




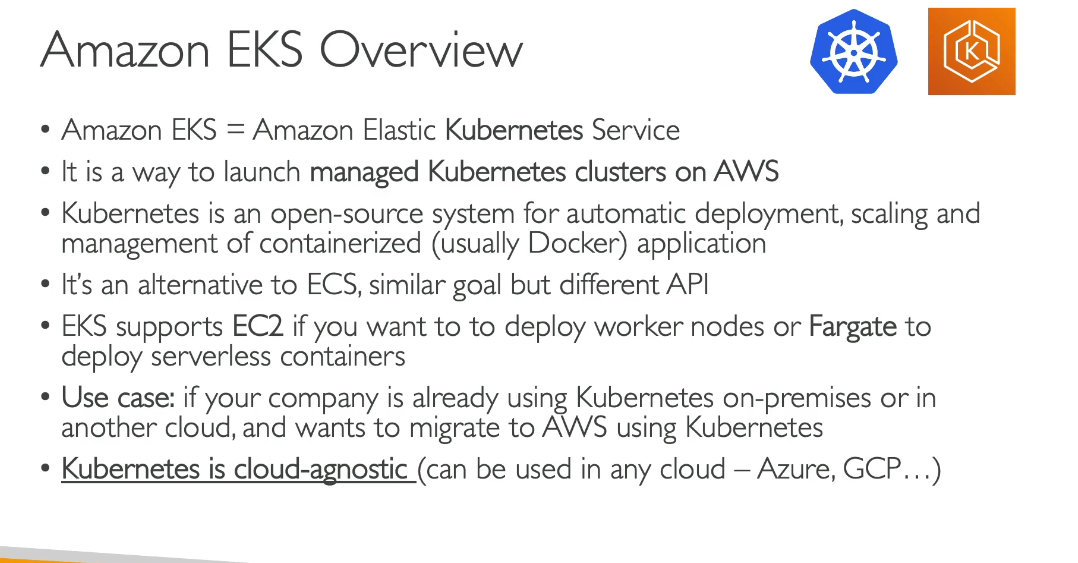


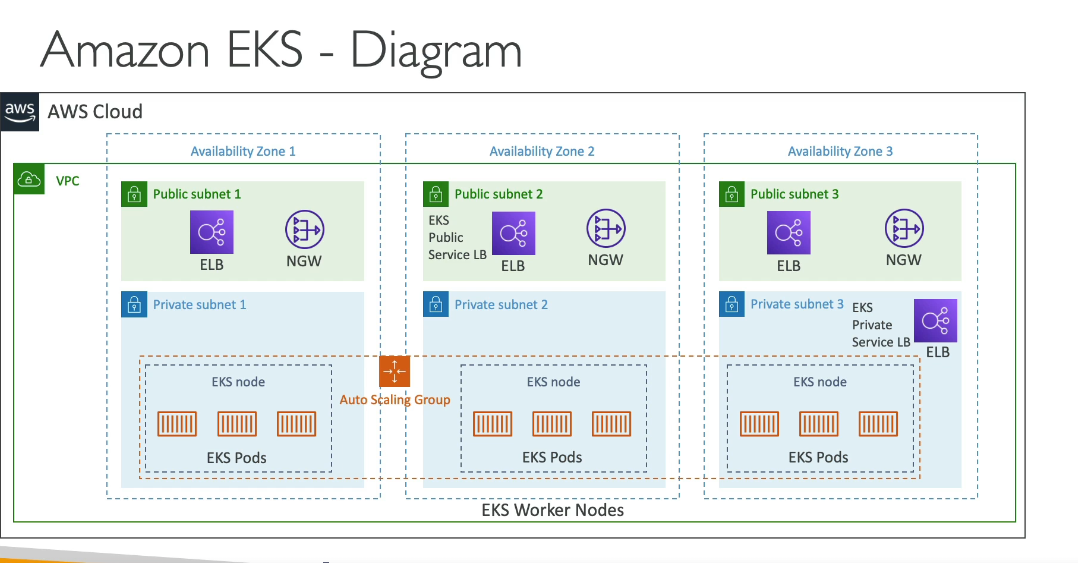


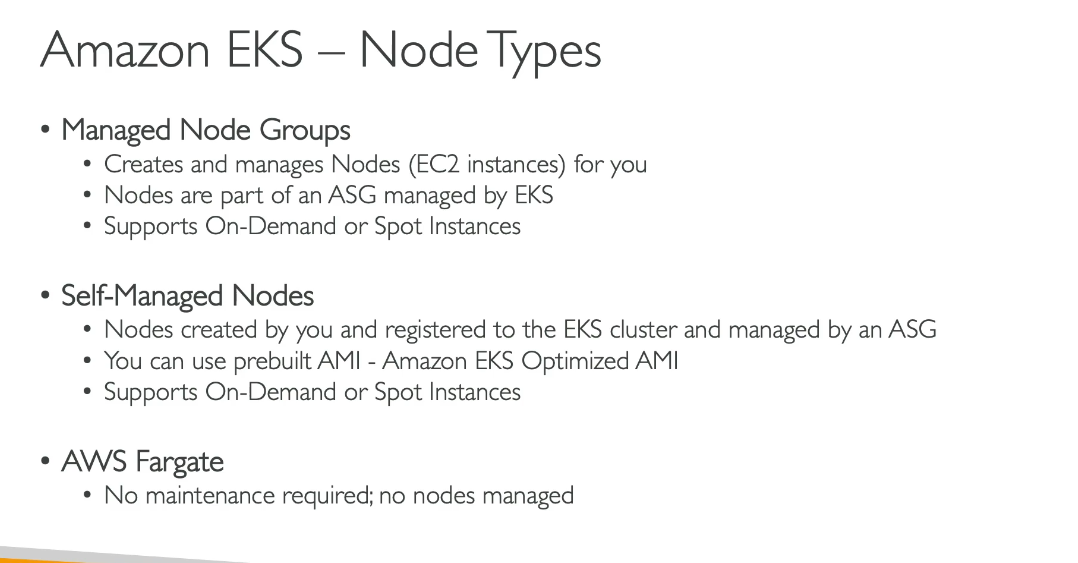
ECR:

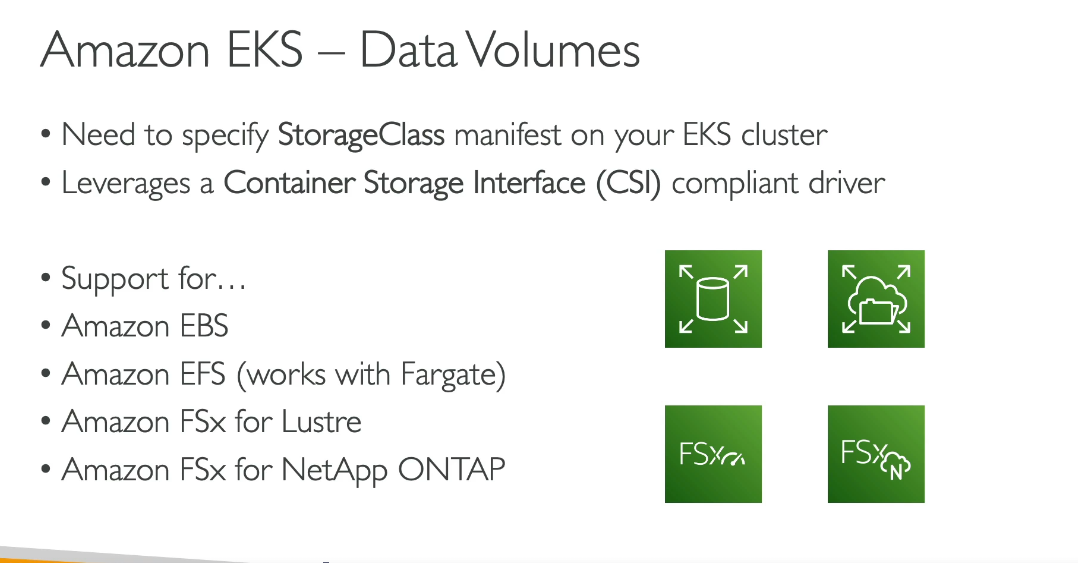


EKS:









CSI = **Container Storage Interface**.

It’s a standard API that lets Kubernetes (including EKS) talk to storage systems **without relying on in-tree drivers**. In plain English: it’s the middleman that keeps Kubernetes from being tied to a specific storage vendor.

Why it exists / what problem it solves:

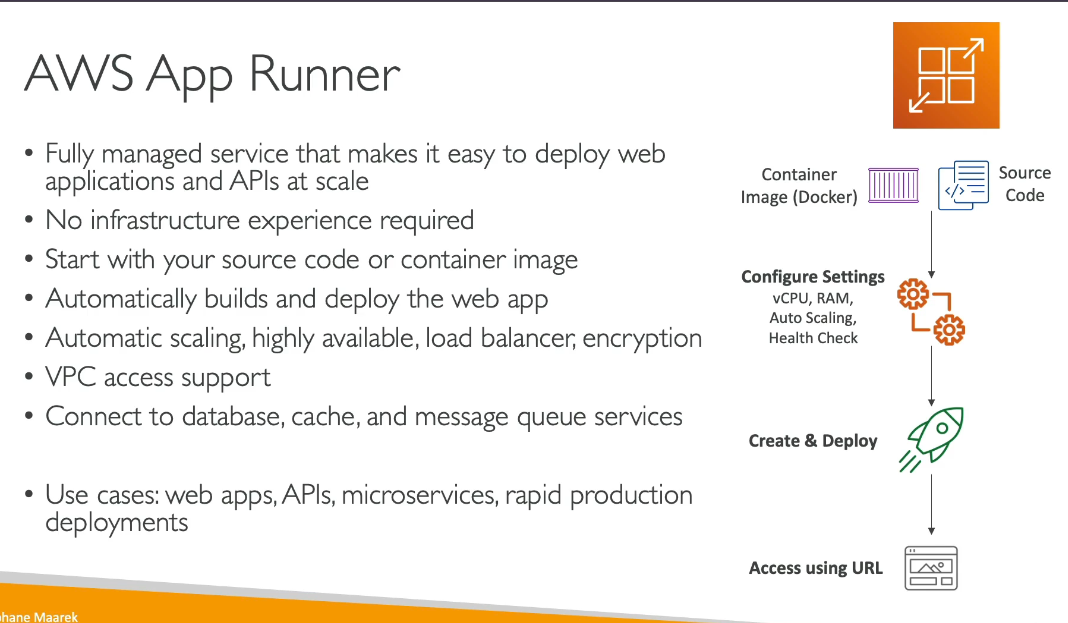
* Kubernetes used to ship built-in storage drivers → upgrade K8s = break storage, vendor lock-in, slow.
* CSI moved storage drivers **out** of Kubernetes core → vendors ship their own drivers, no waiting for K8s release cycles.
* You can plug in EBS, EFS, FSx, whatever, and Kubernetes doesn’t give a damn as long as there’s a CSI driver.

What you should actually remember:

* CSI driver = the plugin you install so K8s can provision/attach/mount storage.
* StorageClass uses that driver to create volumes on demand.
* Your Pods reference PVCs → CSI handles the rest.

If you think CSI is just some buzzword: it’s not. It’s the foundation of modern storage on Kubernetes.

**APP RUNNER:**



**Difference between AWS Fargatte and APP Runner**

AWS Fargate is a serverless compute engine for containers that works with Amazon ECS and Amazon EKS. It removes the need to manage underlying infrastructure while giving you control over container orchestration, networking, and scaling configurations.

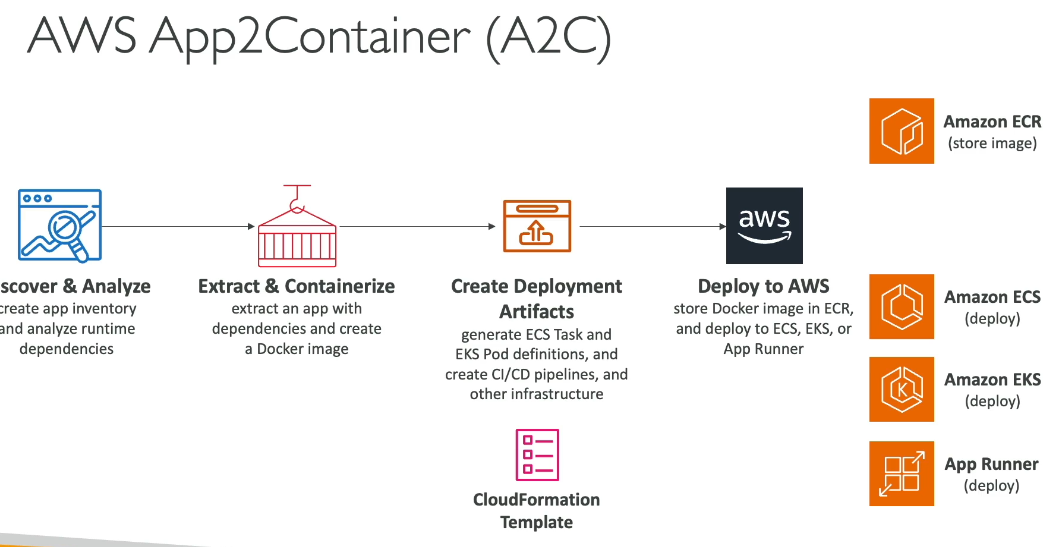
AWS App Runner is a fully managed service specifically designed for deploying web applications and APIs quickly. It handles the entire application lifecycle from source code or container image to production.

**Key Differences:**

* Complexity: Fargate requires you to configure task definitions, services, networking, and load balancers. App Runner simplifies this with automatic configuration.
* Use Case: Fargate is ideal for complex containerized workloads requiring custom networking and orchestration. App Runner is best for straightforward web applications and APIs.
* Control: Fargate offers more granular control over container orchestration, networking, and scaling. App Runner abstracts these details for simplicity.
* Integration: Fargate integrates with ECS/EKS ecosystems. App Runner provides direct deployment from source code repositories (GitHub, Bitbucket) or container registries. In Simple Terms: Choose Fargate when you need flexibility and control over your container infrastructure. Choose App Runner when you want the fastest path to deploy web applications without infrastructure management.

**APP2CONTAINER:**





Quiz:

