Containers and Containerization

Tessema Mengistu (Ph.D.)

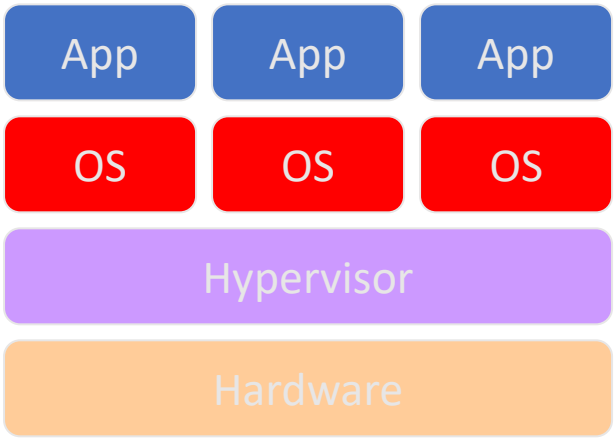
mengistu@vt.edu

Outline

- What are Containers?
- What is Containerization?
- Docker and Kubernetes

What are Containers?

- Virtual Machines (VMs)
  - Use hypervisor to virtualize the physical hardware
  - Contains independent operating systems
  - Heavy weight

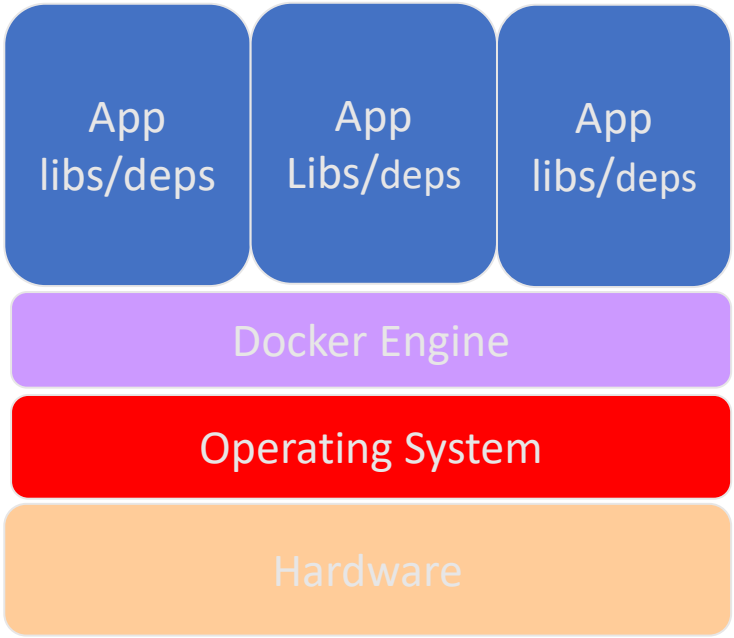


- Containers are light weight alternatives to VMs

What are Containers?

- Containers
  - Are based on operating-system-level virtualization
    - Rather than hardware virtualization
  - Contains only the application and its libraries and dependencies.
    - Small, fast, and portable
  - Multiple containers running on the same machine share the OS kernel
    - Have a smaller memory footprint
    - A shorter start-up time than VMs
  - Example: Docker , Mesos Containerizer, LXC Linux Containers, OpenVZ, etc.

What are Containers?



What are Containers?

- Advantages:
  - Lightweight
    - Startup speed,
  - Platform independent - Portable
    - Applications are decoupled from the infrastructure
  - Improves resource utilization
  - Supports modern software architecture
    - Microservices, serverless

Containers vs. Virtual Machines

Feature	Virtual Machine	Container
Purpose	Efficiently use increasing physical hardware capacity and processing power	package and run applications in a predictable and repeatable way across multiple environments
Virtualization	Virtualizes physical machine	Virtualizes OS
Size	Much larger (think in terms of GB).	Lighter weight (think in terms of MB).
Isolation	Provides complete isolation from the host operating system and other VMs.	Typically provides lightweight isolation from the host and other containers, but doesn't provide as strong a security boundary as a VM.
Operating system	Runs a complete operating system including the kernel, thus requiring more system resources (CPU, memory, and storage).	Runs the user mode portion of an operating system, and can be tailored to contain just the needed services for your app, using fewer system resources.

https://dl.acm.org/doi/pdf/10.1145/3365199, 2020

What is Containerization?

- Containerization
  - The process of packaging an application with its relevant environment variables, configuration files, libraries and software dependencies
  - Results in a container image that can then be run on a container platform

Docker and Kubernetes

- Docker
  - An open platform for developing, shipping, and running applications
  - Uses a client-server architecture
  - Salient Components
    - An **image** is a blueprint of an application
    - A **container** consists of one or more images and runs the actual application
    - The **daemon** is the process that runs under the operating system to which clients talk to
    - A **client** is a command line tool used by a user to interact with the daemon
    - A **hub** is a registry of Docker images, a directory of all available Docker images

Docker and Kubernetes

- Images
  - Immutable (read-only) and very portable
  - Composed of layers
  - Built from Dockerfile
    - Plain text file that specifies all of the components that are included in the container
  - Can be created from scratch or based on an existing image

Docker and Kubernetes

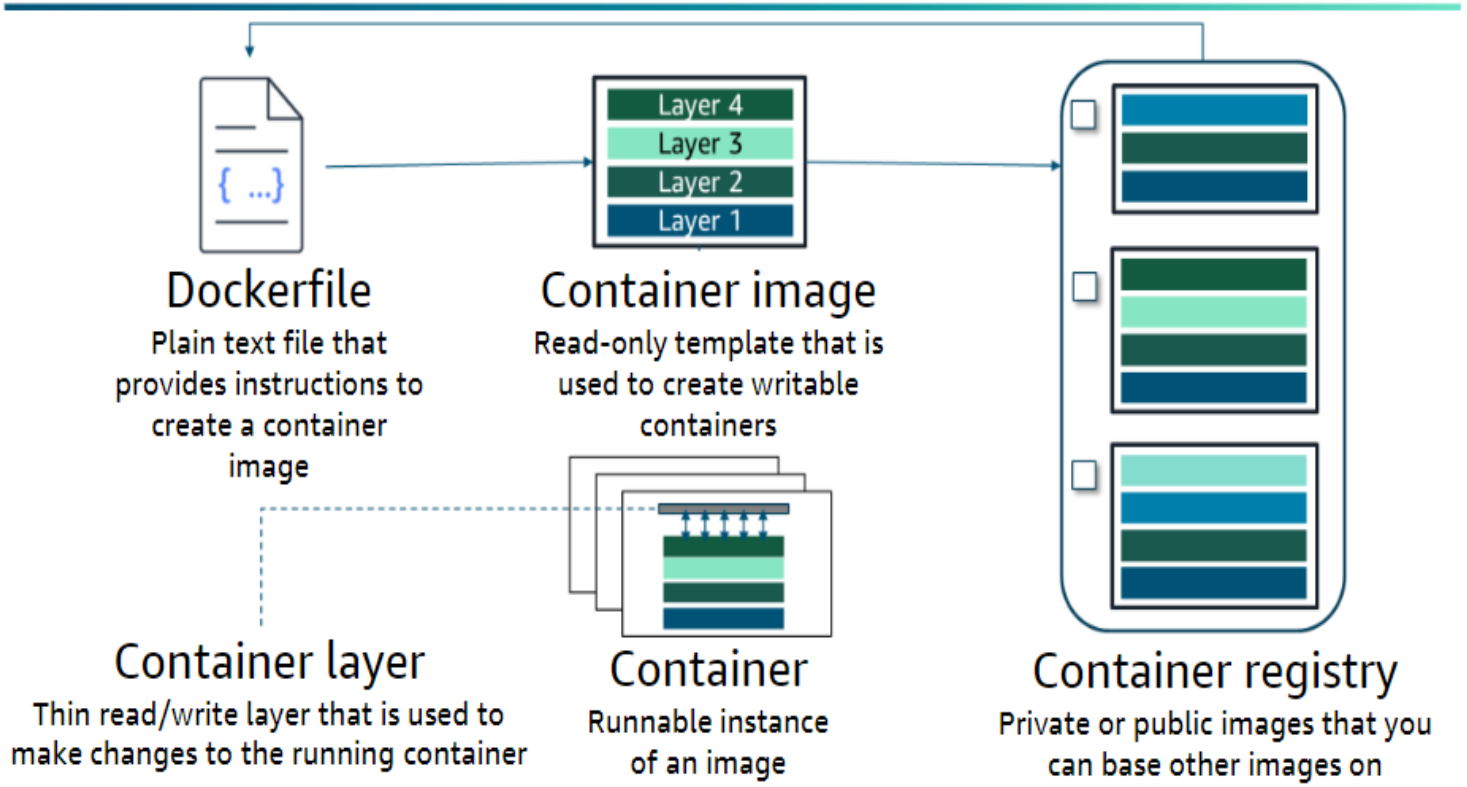
- Dockerfile example:

```
# Start with CentOS 7 image
FROM centos:7

# Update the OS and install Apache
RUN yum -y update && yum -y install httpd

# Expose port 80–the port that the web server "listens to"
EXPOSE Port 80

# Copy shell script and give it run permissions
ADD run-httpd.sh /run-httpd.sh
RUN chmod -v +x /run-httpd.sh

# Run shell script
CMD ["/run-httpd.sh"]
```

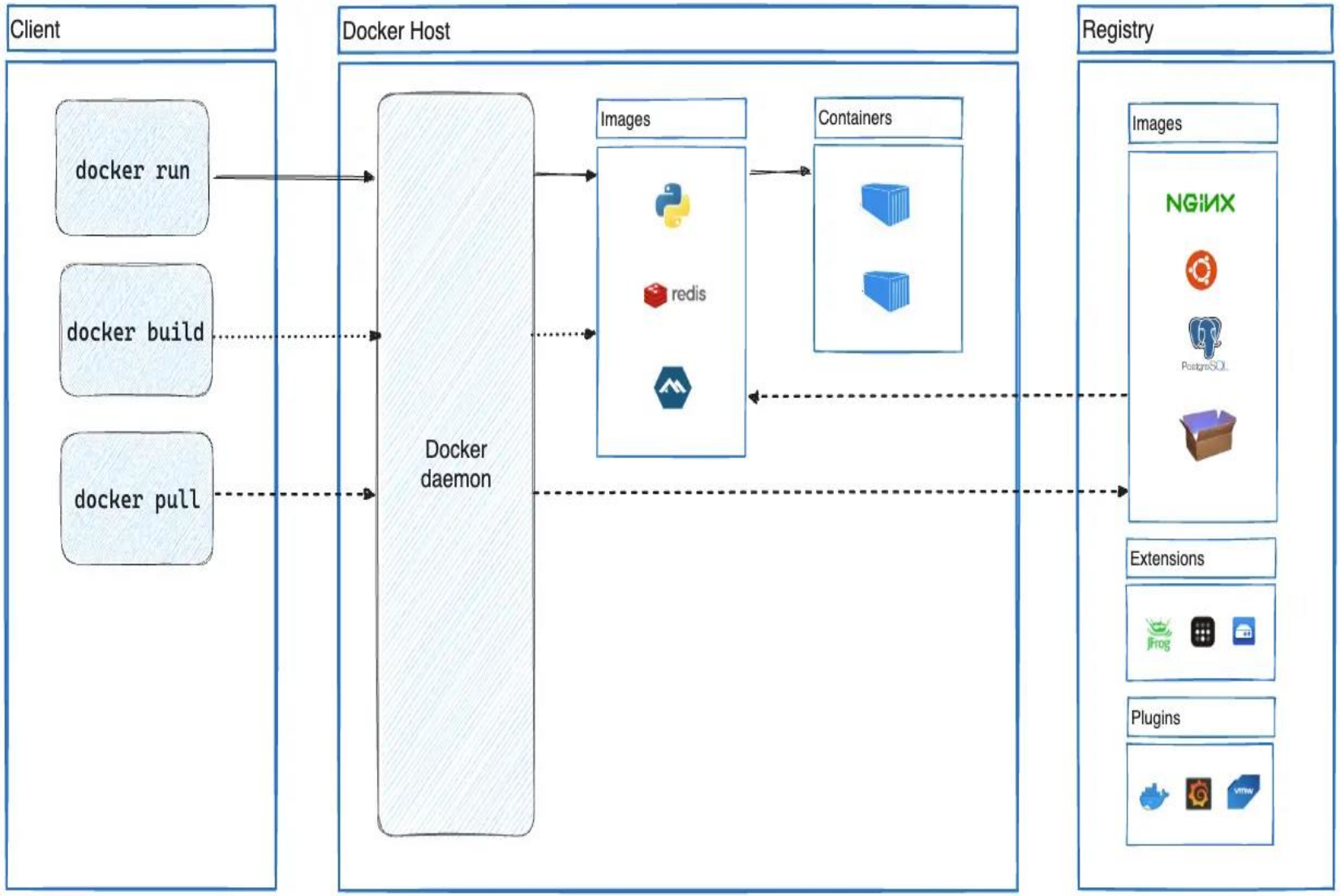
Docker and Kubernetes

- Container
  - Runnable instance of an image
  - Has a thin read/write layer on top of the image when instantiated

Docker and Kubernetes



Docker and Kubernetes



Docker and Kubernetes

Docker CLI commands

Command	Description
`docker build`	Build an image from a Dockerfile.
`docker images`	List images on the Docker host.
`docker run`	Launch a container from an image.
`docker ps`	List the running containers.
`docker stop`	Stop a running container.
`docker start`	Start a container.
`docker push`	Push the image to a registry.
`docker tag`	Tag an image.

Command	Description
`docker logs`	View container log output.
`docker port`	List container port mappings.
`docker inspect`	Inspect container information.
`docker exec`	Run a command in a container.
`docker rm`	Remove one or more containers.
`docker rmi`	Remove one or more images from the host.
`docker update`	Dynamically update the container configuration.
`docker commit`	Create a new image from a container's changes.

Docker and Kubernetes

- Managing thousands of containers is challenging
- Container Orchestration
  - A way of managing large volumes of containers throughout their lifecycle:
    - Provisioning
    - Redundancy
    - Health monitoring
    - Resource allocation
    - Scaling and load balancing
    - Moving between physical hosts
  - Example
    - Apache Mesos, Docker Swarm, **Kubernetes**

Docker and Kubernetes

- Kubernetes
  - An **opensource** software that manages containerized applications in a **clustered** environment
  - Developed by google using Go language
  - The system is lightweight, modular, portable and extensible
  - Provides:
    - Service discovery and load balancing
    - Storage orchestration
    - Automated rollouts and rollbacks
    - Automatic bin packing
    - Self-healing
    - Horizontal scaling
    - Secrete and configuration management
    - Etc.

Docker and Kubernetes

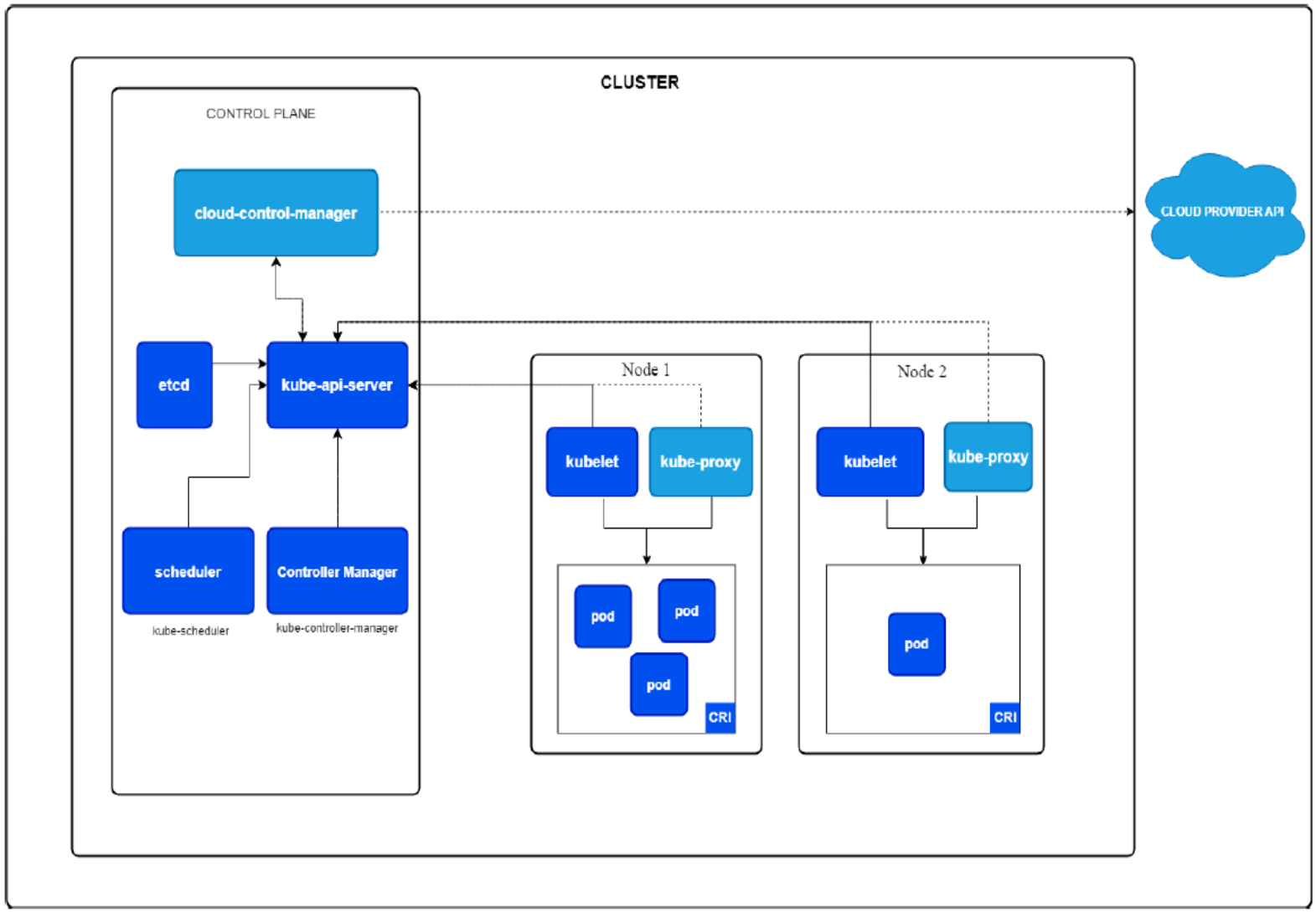
- Architecture
  - Cluster consists of a set of nodes
    - Master node
      - The master node coordinates the cluster
    - Worker nodes - Nodes
      - The worker nodes provide a source of resources for the cluster
      - Kubernetes runs workloads by placing containers into Pods to run on Nodes

Docker and Kubernetes

- Master server/node has
  - Etcd
    - A lightweight, distributed key-value store to share configuration data with the cluster nodes
  - kube-apiserver
    - A HTTP/JSON API that exposes Kubernetes API
  - The Kubernetes scheduler
    - Tracks resources available and those allocated to the workloads on each host
  - Controller Manager
    - Cloud-control-manager
    - Kube-control-manager
      - Node controller
      - Job controller

Docker and Kubernetes

- Worker nodes:
  - Use a docker service to run encapsulated application containers
  - Host the **Pods** that are the components of the application workload
  - Components:
    - A **kubelet** - the primary "node agent“ that makes sure the containers are running
      - Uses PodSpec – JSON/YAML
    - A **kube-proxy** – maintains network rules that allow pods to communicate
    - **Container runtime** - responsible for managing the execution and lifecycle of containers within the Kubernetes environment



Source: https://kubernetes.io/

Containerization in the Cloud

- Containers as a Service (CaaS)
  - Google Container Engine (GKE)
    - Based on Kubernetes- an open-source cluster manager
  - Amazon
    - ECS – Elastic Container Service
  - Microsoft
    - Azur Container Service
    - The Azure Resources Manager API supports multiple orchestrations including Docker, and Apache Mesos

Containerization in the Cloud

- Amazon Elastic Container Service (ECS)
  - A container **orchestration service** that supports Docker containers
  - A task definition
    - JSON format
  - ECS offers two launch types
    - Fargate launch
      - Amazon managed Cluster
    - EC2 launch
      - User managed Cluster

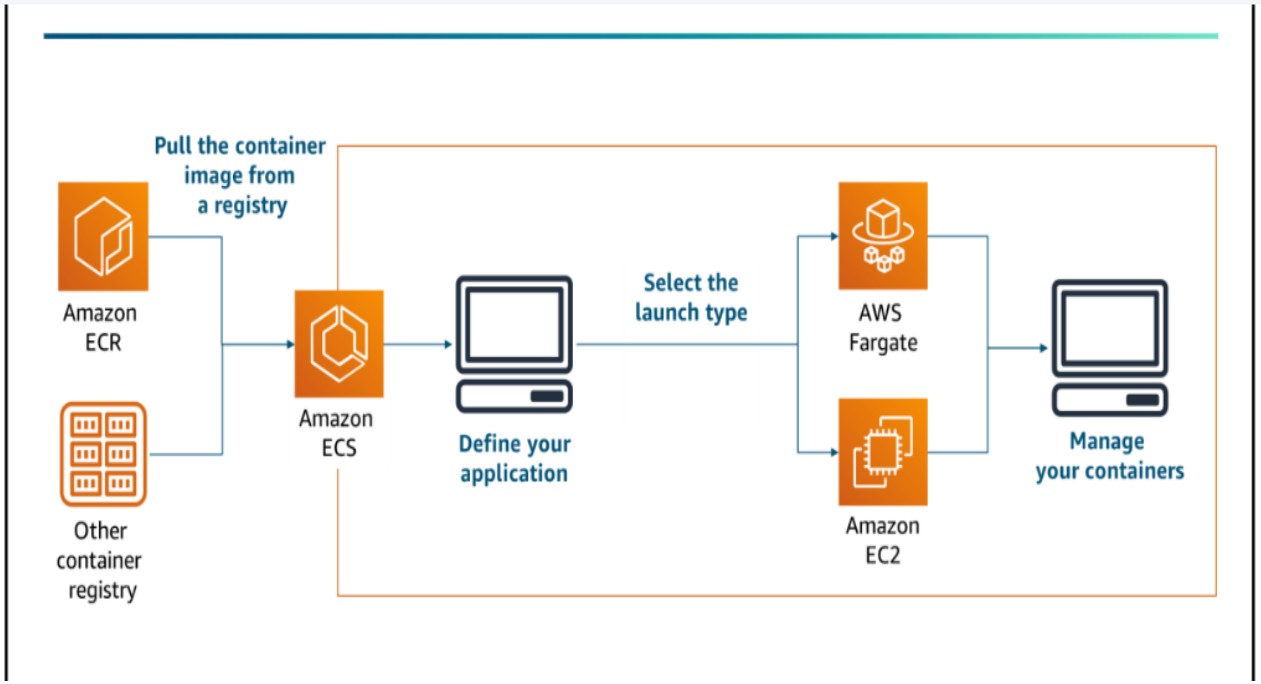
Containerization in the Cloud

- Amazon Elastic Container Registry
  - Docker container registry that makes it easy for developers to store, manage, and deploy Docker container images
  - Integrated with ECS

Containerization in the Cloud

```
# Create a repository called hello-world
> aws ecr create-repository \
    --repository-name hello-world \
    --region us-east-1

# Build and tag an image
> docker build -t hello-world .
> docker tag hello-world:latest aws_account_id.dkr.ecr.us-east-1.amazonaws.com/hello-
world:latest

# Authenticate Docker to your Amazon ECR registry
# You can skip the `docker login` step if you have amazon-ecr-credential-helper set up
> aws ecr get-login-password --region region | docker login --username AWS --password-stdin
aws_account_id.dkr.ecr.region.amazonaws.com

# Push an image to your repository
> docker push aws_account_id.dkr.ecr.us-east-1.amazonaws.com/hello-world:latest
```

Containerization in the Cloud



Containerization in the Cloud

- Amazon Kubernetes Service (EKS)
  - Runs Kubernetes management infrastructure cross multiple availability zones (AZs)
    - Fault tolerance

References

- *Containers 101: Linux containers and Docker explained*
- https://www.ibm.com/topics/containers
- https://docs.docker.com/get-started/overview/
- https://kubernetes.io/docs/home/