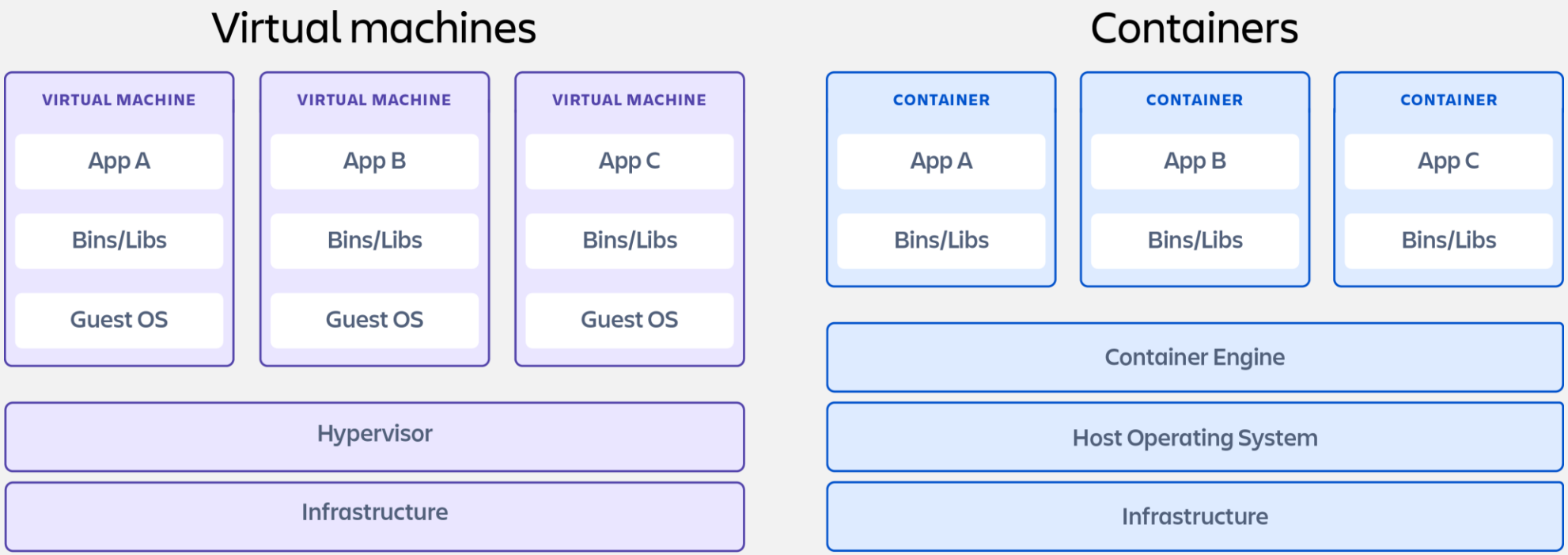
# INTRODUCTION TO DOCKER & KUBERNETES

CS5224 Tutorial

# INTRODUCTION TO CONTAINERS




https://www.atlassian.com/microservices/cloud-computing/containers-vs-vms

# WHAT IS DOCKER?

- Docker is a container provider for developing, shipping, and running applications.
- It allows for automatic container creation from your code (Dockerfile).
- Key features include Docker Engine, Docker Hub, and Docker Compose.

# WHAT IS KUBERNETES

- a.k.a K8s
- Kubernetes is an open-source system for container orchestration.
- Automates deployment, scaling, and operation of multiple application containers within a cluster.
- Handles workloads with automatic binpacking, self-healing, and scaling.

# DOCKER

# INSTALLING DOCKER

- Use Docker Desktop for Windows and macOS; Docker engine for Linux.
- Follow official installation guides for troubleshooting.
  - https://docs.docker.com/desktop/setup/install/windows-install/
  - https://docs.docker.com/desktop/setup/install/mac-install/
  - https://docs.docker.com/engine/install/

# COMMON DOCKER COMMANDS

- docker pull [imageName]: Pull an image from a registry
- docker run [imageName]
  - docker run --memory="256m" nginx
  - docker run --cpus=".5" nginx
- docker run -d [imageName]: Run image in detached mode
- docker start [containerName]: Start stopped containers
- docker ps: List running containers
- docker ps -a: List running & stopped containers
- docker stop [containerName]
- docker kill [containerName]
- docker image inspect [imageName]: Get image info
- container: name of the running image
- docker rm [containerName] : Removes **stopped** container
  - docker rm $(docker ps -a -q): Removes **all** stopped containers
- docker images: List images

https://docs.docker.com/get-started/docker_cheatsheet.pdf

# BUILDING A DOCKER IMAGE

- Use a Dockerfile (.yaml) to define the build steps.
- Example Dockerfile:
  - FROM python:3.8-slim
  - COPY . /app
  - RUN pip install -r /app/requirements.txt
  - CMD ["python", "/app/app.py"]
- Run **docker build -t myapp:latest .** to create the image.
- Run **docker run myapp** to run the container

# KUBERNETES

# SETTING UP KUBERNETES

- Install Minikube or Kind or enable Kubernetes in Docker Desktop for local Kubernetes cluster.
- Use **kubectl** command-line tool to interact with clusters.
- Verify setup with **kubectl cluster-info**.

```
[jeremy@Mac ~ % kubectl cluster-info                                    ]
Kubernetes control plane is running at https://127.0.0.1:6443
CoreDNS is running at https://127.0.0.1:6443/api/v1/namespaces/kube-system/servi
ces/kube-dns:dns/proxy
```

# KUBERNETES CONCEPTS

- Pod: The smallest deployable unit; one or more containers.
- Node: A worker machine in Kubernetes, can be a VM or physical machine.
- Cluster: A set of nodes that run containerized applications.
- Deployment: Maintains desired state of Pods.
- Service: Abstraction to expose applications.
- Namespaces: Multiple virtual clusters on the same physical cluster.

# DEPLOYING TO KUBERNETES

- Write definition YAML file
- Use **kubectl apply –f <definition_file.yaml>** to deploy
- Verify with **kubectl get deployments/pods/services/…**
- Detail info with **kubectl describe deployments <name>**

```
jeremy@Mac ~ % kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
express-app-85c58657f9-8wv2p      1/1     Running   0          38m
express-app-85c58657f9-j6jf4      1/1     Running   0          38m
```

Pod definition

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
    type: front-end
spec:
  containers:
  - name: nginx-container
    image: nginx
```

```
Node:             docker-desktop/192.168.65.3
Start Time:       Wed, 10 Sep 2025 10:59:36 +0800
Labels:           app=express-app
                  pod-template-hash=85c58657f9
Annotations:      <none>
Status:           Running
IP:               10.1.0.106
IPs:
  IP:           10.1.0.106
Controlled By:  ReplicaSet/express-app-85c58657f9
Containers:
  express-app:
    Container ID:   docker://f59e4244ff359fdb9863ef804c2c659a74ad37bc3c9782262b59f26de426aecc
    Image:          itsjeremyhsieh0107/express:v1
    Image ID:       docker-pullable://itsjeremyhsieh0107/express@sha256:10d5bb97d34b82849122f92700683f239a77257528b7
b7d2a22e388a4f385ebe
    Port:           3000/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Wed, 10 Sep 2025 10:59:36 +0800
    Ready:          True
    Restart Count:  0
    Limits:
      cpu:     500m
      memory:  128Mi
    Requests:
      cpu:      250m
      memory:   64Mi
    Liveness:   http-get http://:3000/ delay=30s timeout=1s period=10s #success=1 #failure=3
    Readiness:  http-get http://:3000/ delay=5s timeout=1s period=5s #success=1 #failure=3
    Environment:
      PORT:  3000
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-b7sm8 (ro)
```

# DEMO

# DEMO

1. Dockerize a simple Node.js web application
2. Push the image to Docker Hub
3. Pull the image from Docker Hub to local
4. Deploy the container of the image to a K8s cluster

# PREREQUISITES

1. An account on Docker Hub
2. Docker desktop installed with K8s enabled

# SUMMARY

- Docker's role is to containerize applications
- Kubernetes's role is to orchestrate containers
- Further learning:
  - Docker YouTube Crash Course
  - Kubernetes YouTube Crash Course
  - Docker/Kubernetes documentations

# THANK YOU ☺