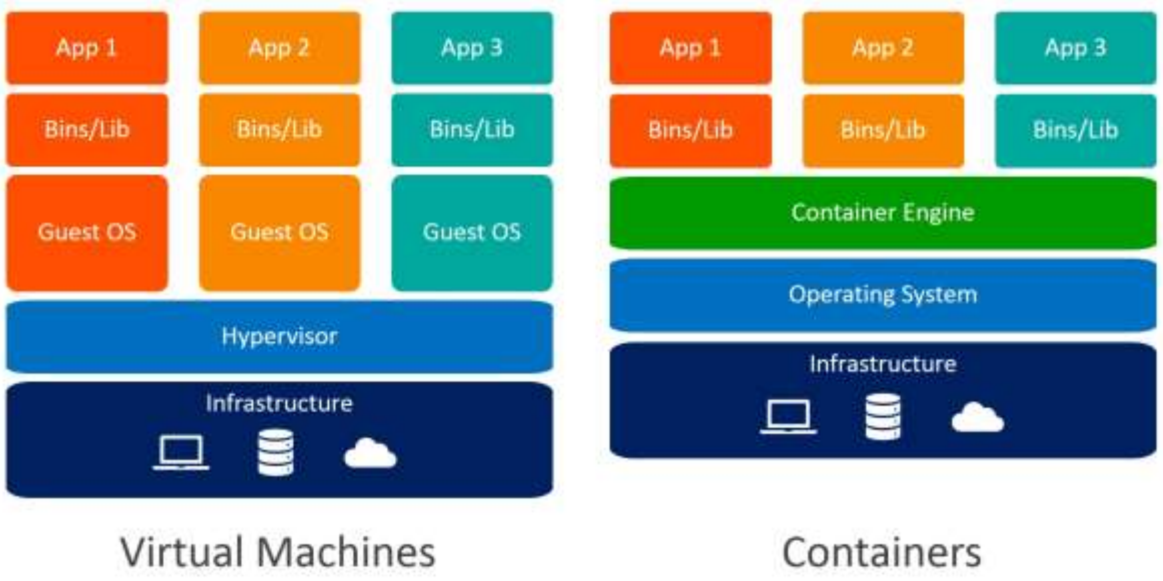
DOCKER AND KUBERNETES (K8)

INTRODUCTION

- ❑ Container is unit packing code with required libraries and runtime
- ❑ Docker is tool for creating and managing containers
- ❑ Always gives same result, when run anywhere

CONTAINERS VS VMS



- ❑ VM, though virtual but the spinning and using VM is not that lightweight
- ❑ Eats CPU and memory of underlying OS
- ❑ Containers are lightweight, low impact on underlying OS
- ❑ Requires minimum disk space
- ❑ Sharing, Re-building and distribution is easier

INSTALLATION AND TOOLS

- The encapsulation happens in something called as ***images***
- Storing, finding and sharing these images can be done by a service, called as ***Docker Hub*** *(A service provided by Docker)*
- Create an account on *https://login.docker.com/,* its free
- For building and containerizing applications, an opensource containerization technology is used called as ***Docker Engine***
- Installation steps

```
sudo apt-get update
sudo apt-get upgrade
Follow - https://docs.docker.com/engine/install/ubuntu/
sudo usermod -aG docker ${USER}
sudo groupadd docker
sudo gpasswd -a $USER docker
newgrp docker
restart system
sudo docker run hello-world (If this works fine, docker is installed successfully)
```

- ***Docker Engine*** consists of a server which runs docker (daemon), APIs to interact with docker daemon and a cli client which is ***docker***
- Docker Playground - https://labs.play-with-docker.com/

DEMO

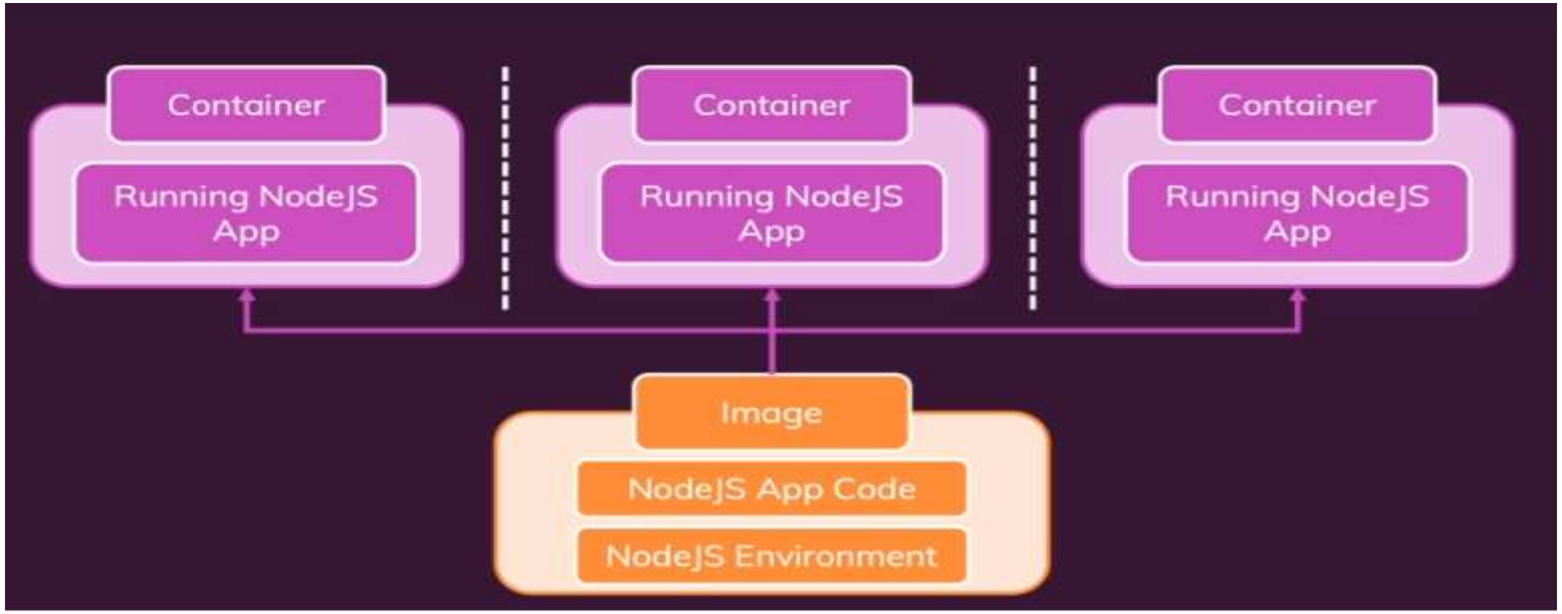
- ❑ First docker demo –

\home\chandra\docker-kubernetes-learnings\docker_demo

- ❑ There is no node or library installed on local machine
- ❑ Other developer can pull and run the image, will run exactly in same manner

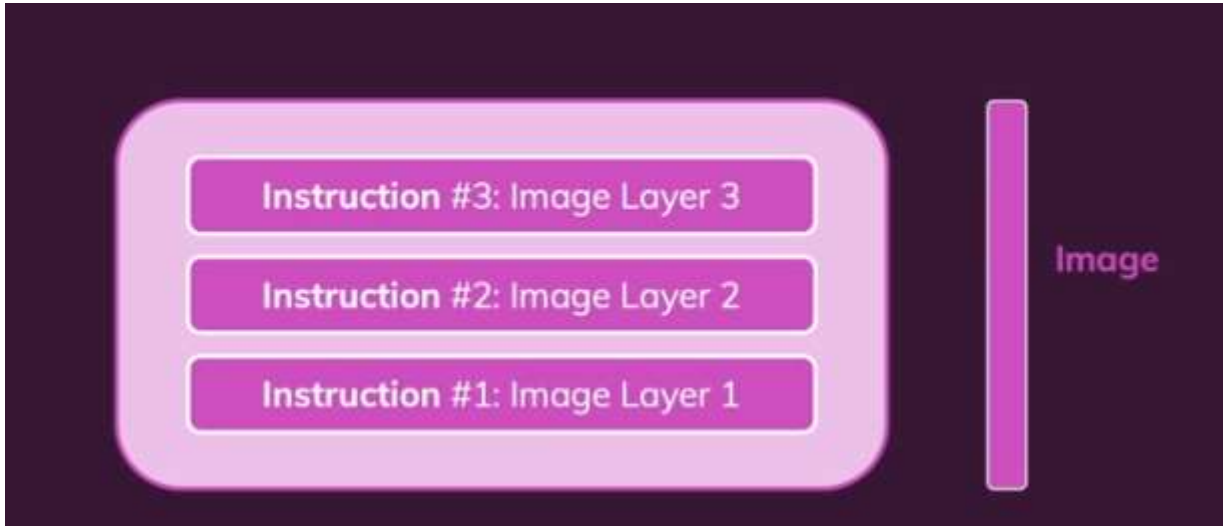
```
docker push cbagade/cl-first-prog:v1
```

IMAGES AND CONTAINERS



- ❑ Images are blueprint containing code, runtime, libraries
- ❑ Container is unit of software, created from image, and are running instances of image
- ❑ From 1 image, multiple containers can be created on multiple machines and multiple environments
- ❑ Lot of pre-built images are available like node-js , java, dotnet, ruby
- ❑ Try docker run -it node and then 1+1
- ❑ Try docker images, to see latest node js image

IMAGE LAYERS


Instruction #3: Image Layer 3
Instruction #2: Image Layer 2
Instruction #1: Image Layer 1
Image

- ❑ Every instruction is represented as layer on image
- ❑ After a first built, the instructions are cached
- ❑ When you changes something and rebuild, the instructions are evaluation against cached ones and only changed ones are built again, super-fast
- ❑ But when a layer is changed, all layers beyond that are re-built
- ❑ Say in previous application, app.js is changed to have new library, then npm install instruction and all onwards instructions are re-executed
- ❑ Layers are read-only

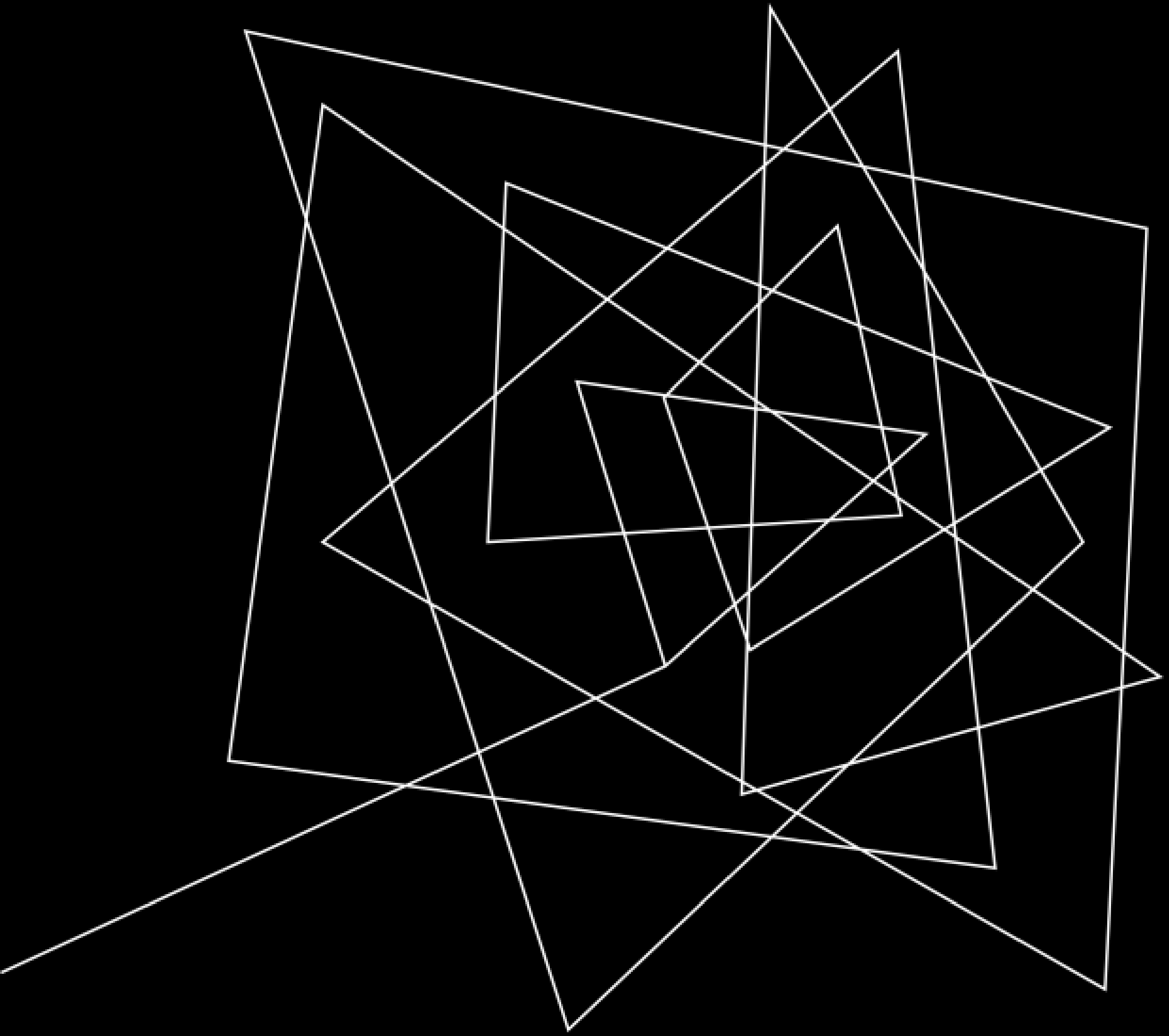
DEMO FROM PREVIOUS NODE LEARNINGS

❑ Docker demo –

\home\chandra\docker-kubernetes-learnings\docker_demo_2

❑ Delete image on local machine and execute demo

```
docker pull cbagade/cl-first-prog:v3
```



MANAGING DATA

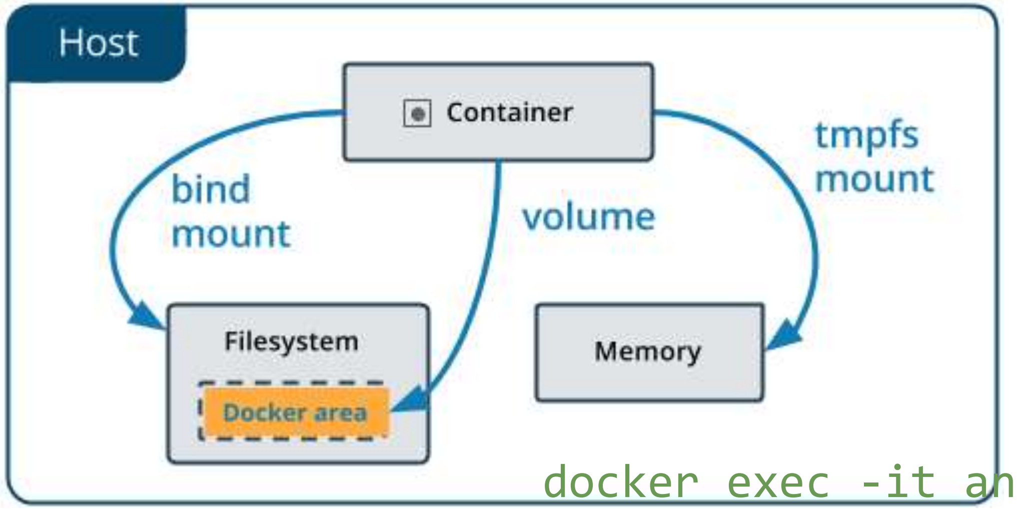
DATA INSIDE CONTAINER

- ❑ Demo for changing data inside container

\home\chandra\docker-kubernetes-learnings\managing_data\data_till_container

- ❑ The changes made in above demo, do not persist after container is recreated
- ❑ The changes are happening inside container and not on host

VOLUMES



```
docker exec -it anonymous-vol /bin/bash
```

- ❑ Volumes are some folders on your host which are mounted on containers
- ❑ Helps to persist data
- ❑ Volumes are like connection between folders, outside and inside of container
- ❑ Volumes are not affected by container shutdown

ANONYMOUS VOLUMES

- The volume will be a directory on local host, mounted on container
- The changes made to the files inside mounted directory, will be reflected on host
- The anonymous volume can be mounted inside Dockerfile as

```
VOLUME [ "/app/userchanges" ]
```

- The mounted volume on container, will be available on host but the directory on host is controlled by docker system
- This volume has no name and controlled by docker system, hence called as anonymous volume

\home\chandra\docker-kubernetes-learnings\managing_data\anonymous_volume

- The volume is gone with container

NAMED VOLUMES

- ❑ The volume will be a directory on local host, mounted on container
- ❑ The changes made to the files inside mounted directory, will be reflected on host
- ❑ The named volumes can be mounted while running docker run command
- ❑ The mounted volume on container, will be available on host but like anonymous volumes the directory on host is controlled by docker system

\home\chandra\docker-kubernetes-learnings\managing_data\named_volume

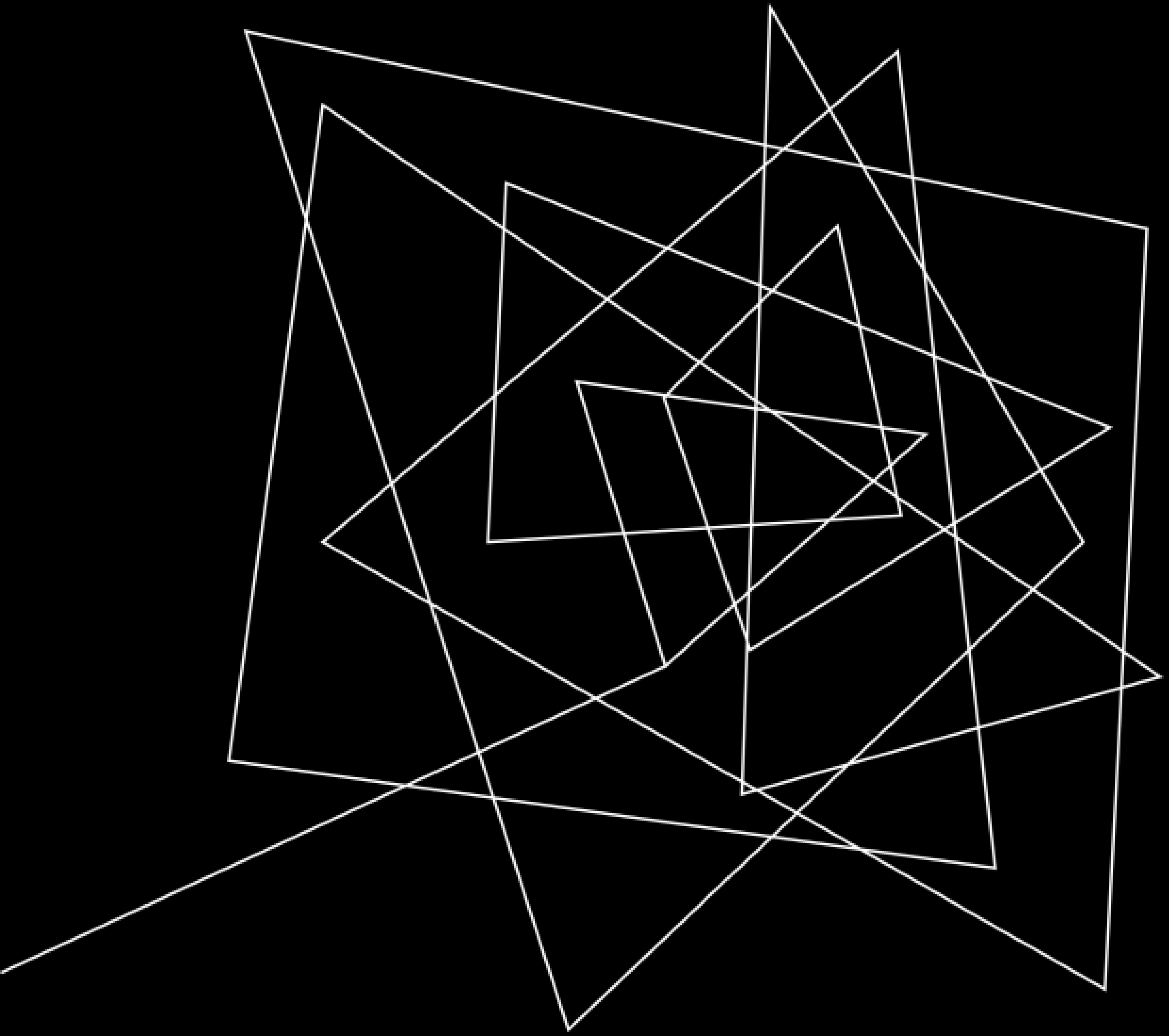
- ❑ But this volume survives container restart

BIND VOLUMES

- ❑ User specify a directly on host to be mounted on container
- ❑ The changes made to the files inside mounted directory, will be reflected on host
- ❑ Not to be used on production, only for local testing

\home\chandra\docker-kubernetes-learnings\managing_data\bind_volume

- ❑ This volume survives container restart
- ❑ Use case can be putting entire code to a bind vol so that code changes can be made inside container and tested
- ❑ Only use for dev and not production



NETWORKING

CONTAINER COMMUNICATION

- ❑ OOB container can make request to outside world

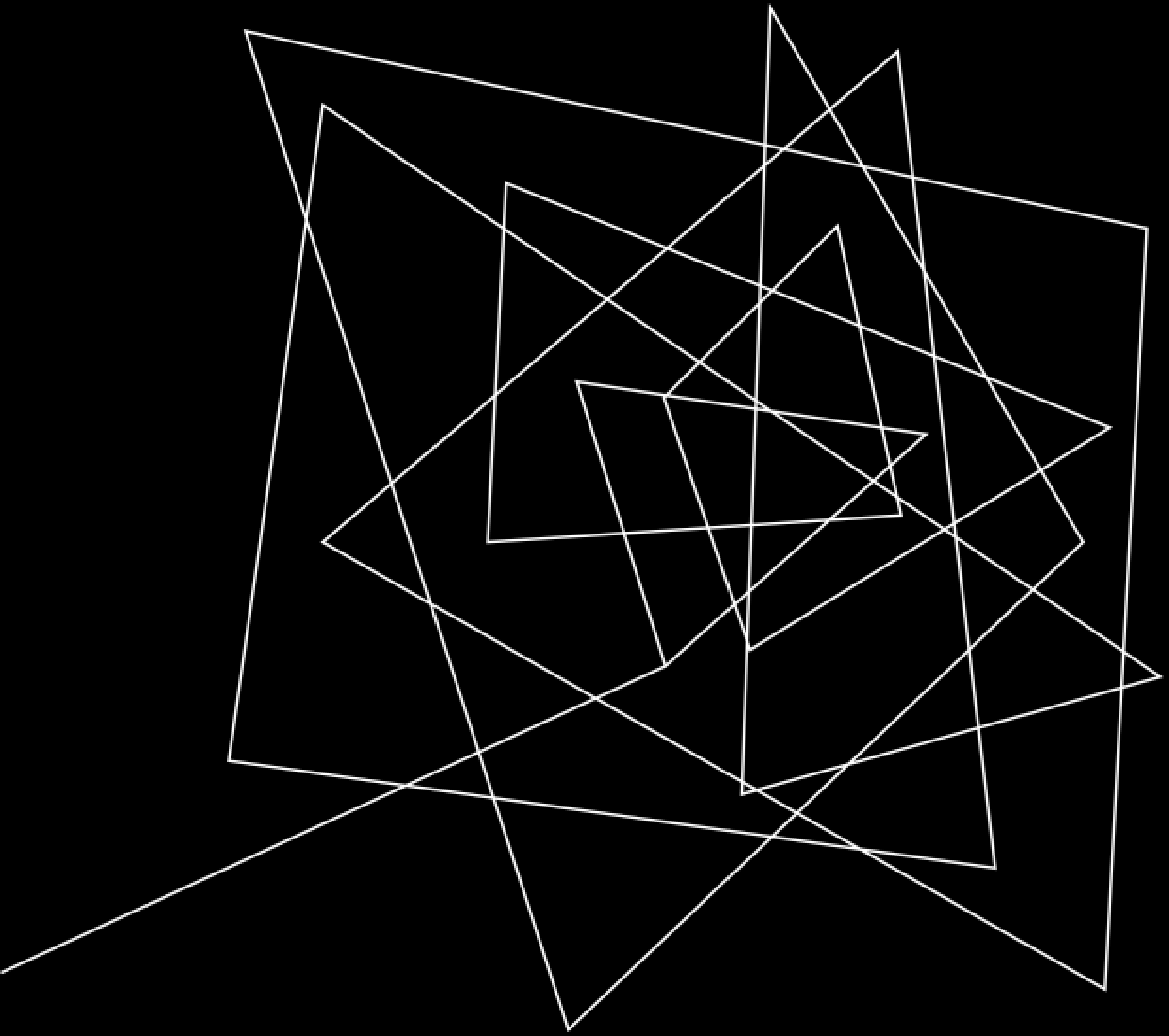
\home\chandra\docker-kubernetes-learnings\networking\www

- ❑ Cross container communication
- ❑ Demo shows communication with mongo container, this demo will have ip of mongo container hardcoded in app.js

\home\chandra\docker-kubernetes-learnings\networking\cross_container\ip_hardcoding

- ❑ If containers are created on same network, then these can communicate with names

\home\chandra\docker-kubernetes-learnings\networking\cross_container\container_name



DOCKER COMPOSE

DOCKER COMPOSE

- ❑ Till now, we created all pods, volume, networks with command line and we also deleted those with command line
- ❑ This is cumbersome when there are lot many containers and artefacts
- ❑ ***docker-compose*** is tool, which makes managing multi-container setup easier
- ❑ ***docker-compose*** helps to replace ;*docker build and docker run’* commands with one configuration file
- ❑ ***docker-compose*** does not replace *Dockerfile, Images or Containers*
- ❑ *docker-compose* file is composed with yaml syntax

\home\chandra\docker-kubernetes-learnings\dockercompose\mongocompose

\home\chandra\docker-kubernetes-learnings\dockercompose\app_by_compose

- ❑ By default, everything inside docker-compose file will be created in one network

MISCELLANEOUS

MISC

Execute command in running container

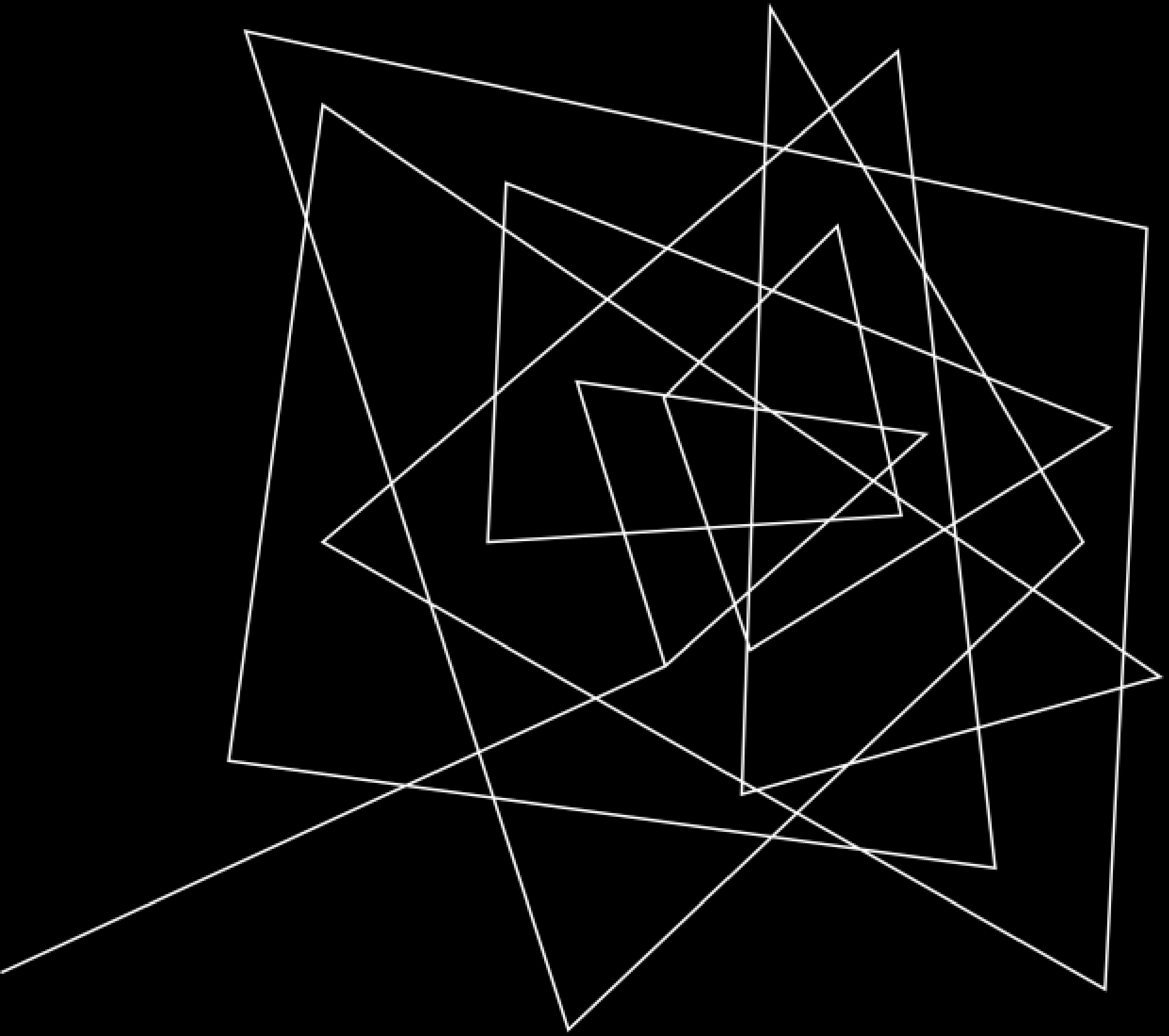
- □ ***docker exec*** executes command in a running container
- □ Helpful for debugging

docker exec -it <container_id> cmd (docker exec -it <container_id> ls docker exec -it <container_id> pwd docker exec -it <container_id> /bin/bash)

CMD and ENTRYPOINT

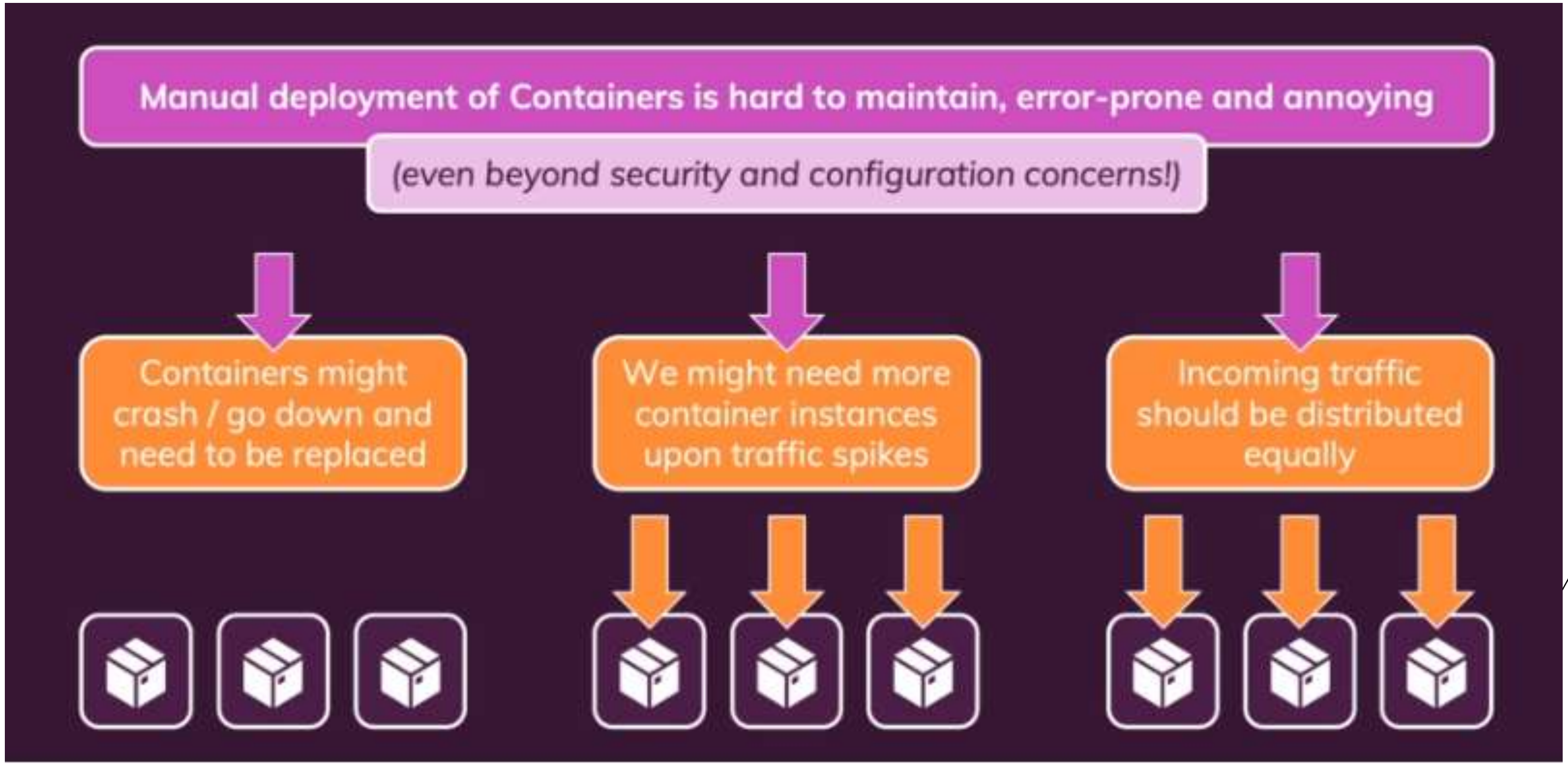
\home\chandra\docker-kubernetes-learnings\misc

- □ Prefer ENTRYPOINT
- □ Try examples with ENTRYPOINT



KUBERNETES

MANUAL DEPLOYMENT



Manual deployment of Containers is hard to maintain, error-prone and annoying

(even beyond security and configuration concerns!)

Containers might crash / go down and need to be replaced

We might need more container instances upon traffic spikes

Incoming traffic should be distributed equally

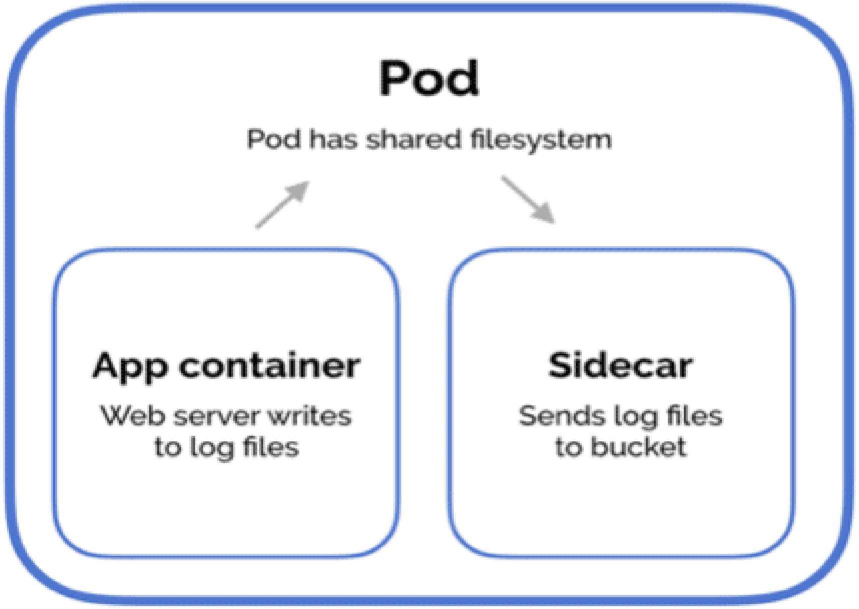
❑ A scenario for manual deployment of containers and managing those

❑ Impossible

KUBERNETES (K8)

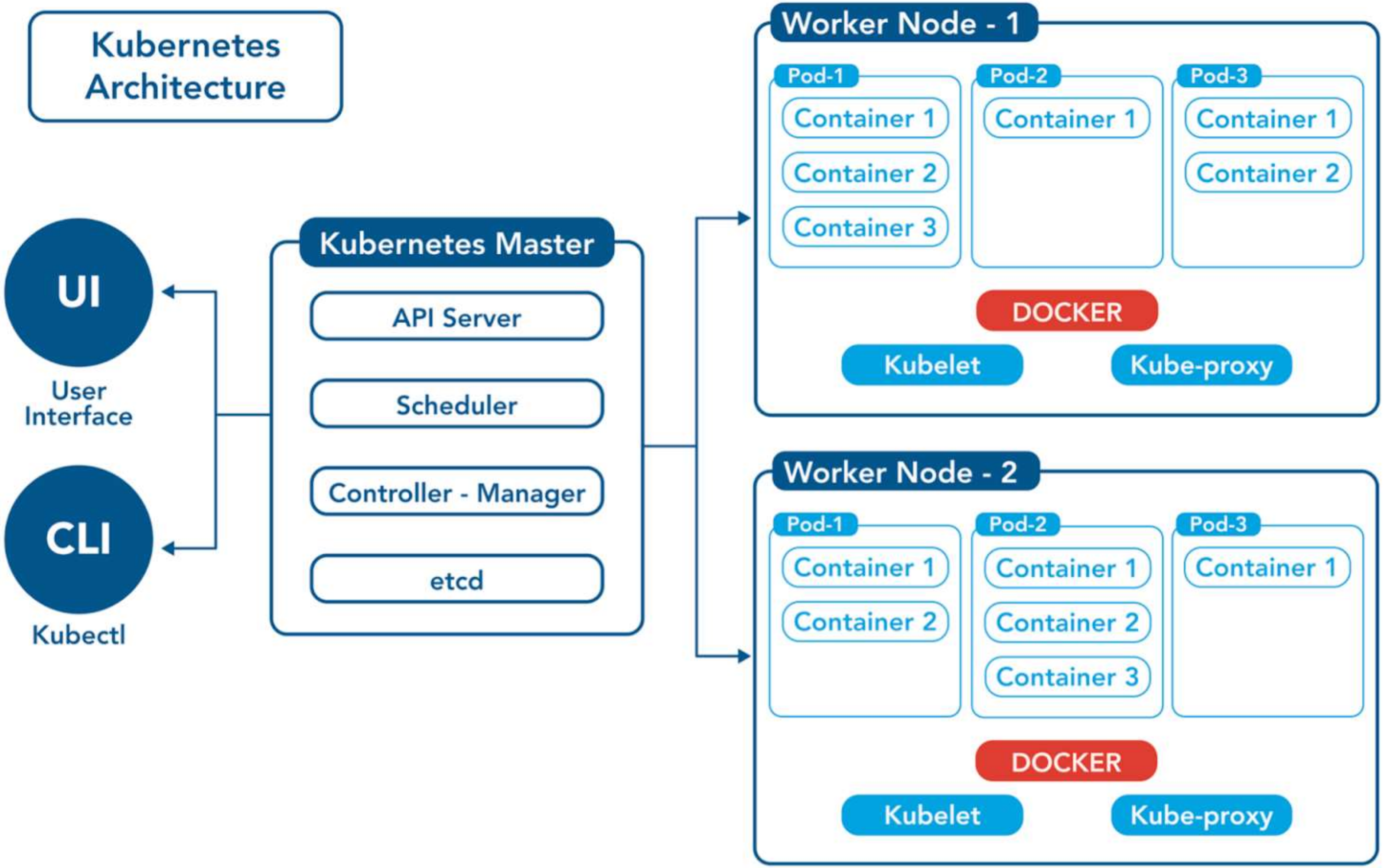
- ❑ Kubernetes is container orchestration tool, which facilitates large scale deployment of containers
- ❑ Opensource and independent of any cloud provider
- ❑ It automates deployment, scaling and management of containerized applications
- ❑ It is NOT a cloud service provider, or a service provided by cloud service provider
- ❑ Its is NOT restricted to any cloud
- ❑ It is NOT replacement for docker but works with docker to deploy and manage containers at large scale
- ❑ It's a free service, but cloud providers also provide managed k8 services which are paid like Elastic Kubernetes Service (EKS) by AWS and Azure Kubernetes Service (AKS) by Azure

CONTAINERS AND PODS


Pod
Pod has shared filesystem
App container
Web server writes to log files
Sidecar
Sends log files to bucket

- ❑ Containers consist of application and all the libraries and dependencies required to run the application, packaged as single software unit
- ❑ Pod is group of one or more containers, with shared storage and network resources
- ❑ Pod is smallest deployable unit of computing, that is created and managed in k8
- ❑ If pod is holding multiple containers, then they can communicate with each other by localhost
- ❑ Pod are ephemeral in nature, meaning stateless, K8 will stop, start replace them when needed
- ❑ IP changes every time, when a pad is restarted

K8 ARCHITECTURE



INSTALLATION

- ❑ For dev , setup can be done with minikube, KinD (Kubernetes in Docker)
- ❑ Playground - https://killercoda.com/playgrounds/scenario/kubernetes
- ❑ Cloud providers provides managed k8 services like EKS, AKS

KinD installation on Ubuntu (WSL)

```
sudo curl -L "https://kind.sigs.k8s.io/dl/v0.18.0/kind-$(uname)-amd64" -o /usr/local/bin/kind

sudo chmod +x /usr/local/bin/kind

kind get clusters
```

kubectl installation on Ubuntu (WSL)

```
curl -k -LO https://storage.googleapis.com/kubernetes-release/release/v1.27.1/bin/linux/amd64/kubectl

sudo chmod +x ./kubectl

sudo mv ./kubectl /usr/local/bin/kubectl

kubectl version --short
```

INSTALLATION

Create kind cluster (WSL ubuntu)

cd \home\chandra\docker-kubernetes-learnings\kubernetes_demos

kind create cluster --config=cluster-config.yml --name=chandra-learnings

If there are issues like, kubelet not healthy then follow -

On ubuntu - echo -e "[boot]\nsystemd=true" | sudo tee /etc/wsl.conf

On powershell (admin) -> wsl -shutdown

wsl

exit from powershell and log onto ubuntu - and create cluster again

Restart machine will shutdown kind cluster

docker ps -a

restrat container of kind image

LEARNING K8

Nodes – Display master and worker nodes

```
kubectl get nodes
```

Namespace – Mechanism to isolate groups of resources within cluster

```
kubectl get namespace

kubectl config view --minify | grep namespace

kubectl create namespace kube-learnings

kubectl config set-context --current --namespace=kube-learnings
```

❑ Let’s build a docker image first (refer Dockerfile)

```
\home\chandra\docker-kubernetes-learnings\kubernetes_demo\confirm_setup
```

Deployment – Deployment tells kubernetes how to create and modify instances of pods that hold contenarized application

```
kubectl create deployment kube-setup --image=cbagade/kube-setup:v1

kubectl get deployment

kubectl get pods

Kubectl describe ….. <deployment> and <pod>
```

LEARNING K8

Dashboard– UI View for Kubernetes artifacts

- ❑ The Pod IP is reachable inside cluster only. To make it reachable outside cluster, Service need to be used.

Service – Mechanism to expose network application, running as one or more pods in cluster

Scaling – Can scale Pods ups and down to desire number

Update Deployment – to new image

Rollback Deployment – to previous image

Delete Service – to previous image

Delete Deployment – to previous image

Follow Dockerfile at \home\chandra\docker-kubernetes-learnings\kubernetes_demo\confirm_setup

SERVICE

❑ Service is used to expose the pod, outside the cluster

❑ 3 types of services –

1. ClusterIP – Reachable from within cluster
2. NodePort - A NodePort service builds on top of the ClusterIP service, exposing it to a port accessible from outside the cluster. Basically, it exposes a port onto node.
3. LoadBalancer - A LoadBalancer service is based on the NodePort service, and adds the ability to configure external load balancers in public and private clouds

Follow Dockerfile at \home\chandra\docker-kubernetes-learnings\kubernetes_demo\confirm_setup

DECLARATIVE APPROACH

- ❑ Imperative approach got lot of commands
- ❑ Same problem and docker or docker-compose
- ❑ Tough to write commands for larger deployments
- ❑ Kubernetes allow us to create resources inside yaml file having configuration objects
- ❑ These files will have objects, which k8 can understand

\home\chandra\docker-kubernetes-learnings\kubernetes_demo\declarative_approach\pod.yaml

\home\chandra\docker-kubernetes-learnings\kubernetes_demo\declarative_approach\deployment.yaml

\home\chandra\docker-kubernetes-learnings\kubernetes_demo\declarative_approach\service.yaml

- ❑ Mostly app specific resources would be incorporated in single yaml file

\home\chandra\docker-kubernetes-learnings\kubernetes_demo\declarative_approach\app_deployment.yaml

LIVENESS PROBE

- ❑ Deployment will try to bring container from error state to running state, when pod goes in error state
- ❑ The liveness probe will go one step further and explicitly makes request to a configured path to check if pod is in healthy state, if not will restart
- ❑ This is useful in deadlock situation or a bug in system to keep application running

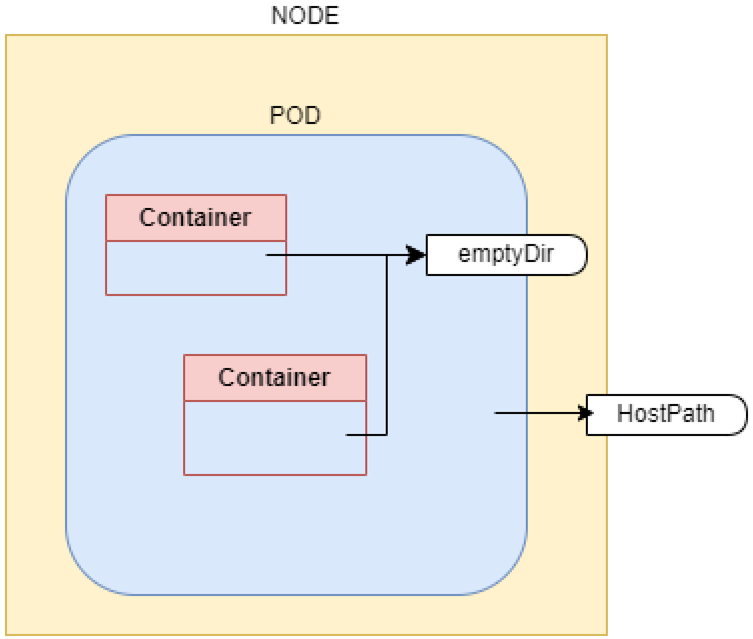
\home\chandra\docker-kubernetes-learnings\kubernetes_demo\declarative_approach\app_deployment_liveliness.yaml

KUBERNETES VOLUMES

Changes, without Volume

- ❑ This demo make temp changes to a file, but all are localized inside container.
- ❑ Can’t survive container or pod restart

\home\chandra\docker-kubernetes-learnings\kubernetes_demo\k8_with_volumes\without_volume



KUBERNETES VOLUMES

emptyDir

- ❑ Creates an empty directory on pod when pod starts and keep alive till pod is alive
- ❑ Kind of temporary folder created, when pod starts
- ❑ As name suggest it is initially empty
- ❑ Can survive container restart, but can’t survive pod restart

\home\chandra\docker-kubernetes-learnings\kubernetes_demo\k8_with_volumes\empty_dir

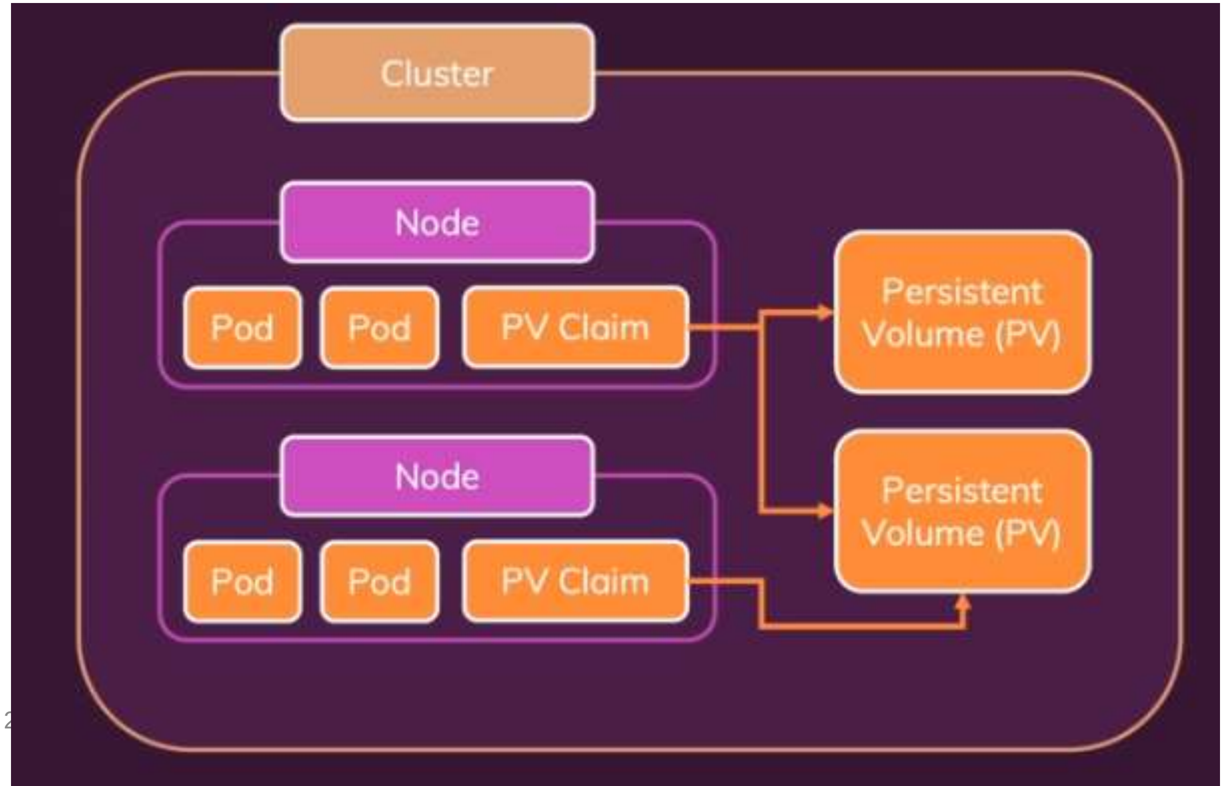
hostPath

- ❑ Mounts a file/directory from node’s filesystem into the pod
- ❑ Can survive container restart and pod
- ❑ Won’t work on multi-node cluster

\home\chandra\docker-kubernetes-learnings\kubernetes_demo\k8_with_volumes\host_path

PERSISTENT VOLUMES (PV AND PVC)

- Pod and Node independent volumes are needed
- Persistent volumes are kubernetes entities like pod, deployment
- There are independent of pods and nodes
- PV don’t store data on nodes
- When pods need PVs, they need to raise a claim, called as Persistent Volume Claims (PVCs)
- PVCs are then bound to PVs, making PVs available to pod



PERSISTENT VOLUMES (PV AND PVC)

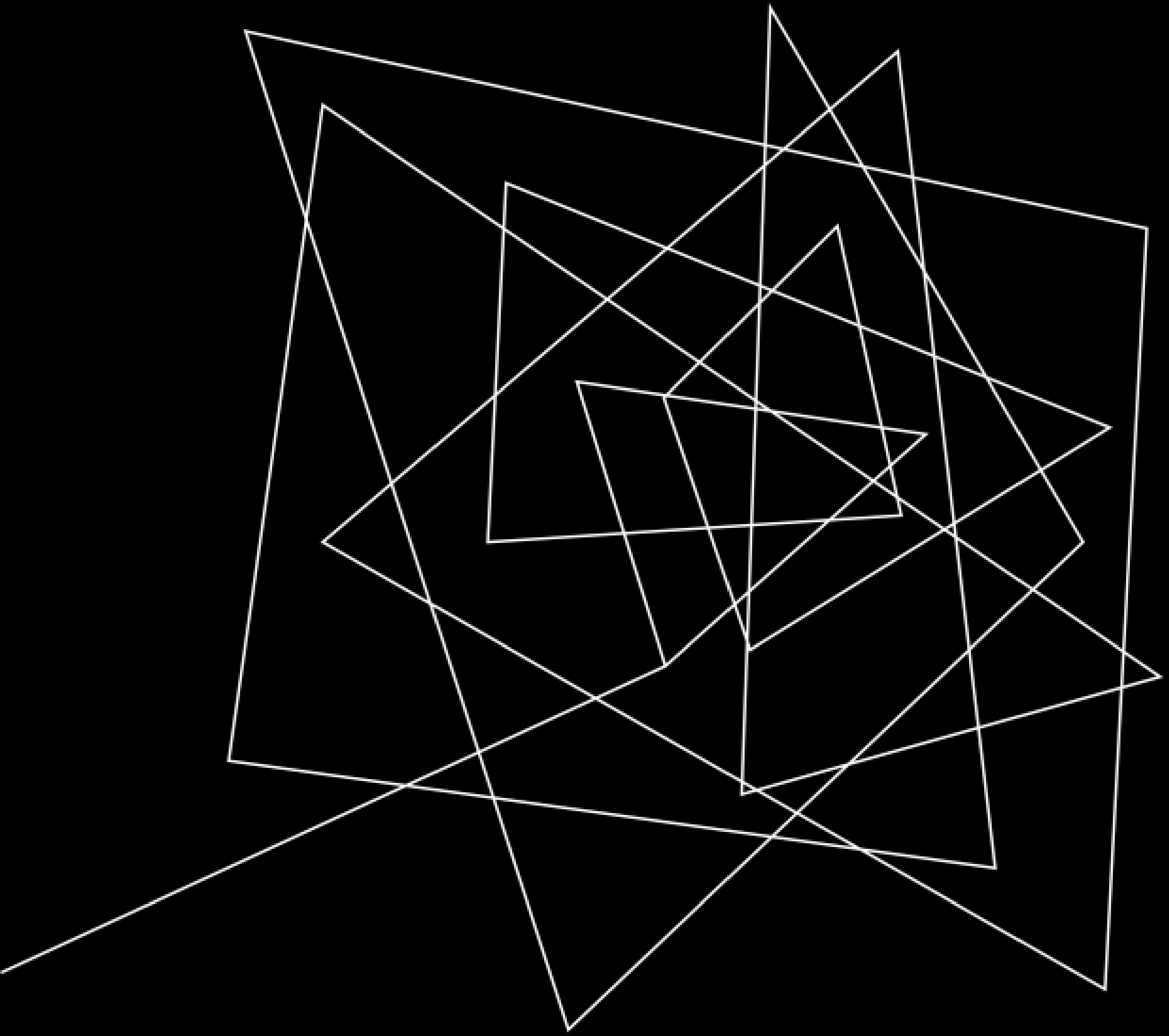
- ❑ PV is abstract and actual storage can come from anywhere like cloud providers (aws, azure, google etc.,) or nfs
- ❑ To bind PV, there should be a storage class and appropriate driver
- ❑ kind provides a standard storage class to create volume of type hostPath
- ❑ Go through Kubernetes Persistent Volume documentation to learn more about storage classes, drivers and various types of storages

\home\chandra\docker-kubernetes-learnings\kubernetes_demo\k8_with_volumes\persistent_volume

ENVIRONMENTAL VARIABLES

- ConfigMap is a kubernetes object, used to store non-confidential information
- Decouple environment specific information from container images, so that application is easily portable across environment
- Imagine database host value as localhost in your machine and something else on development or production environment, can be incorporated in config map

\home\chandra\docker-kubernetes-learnings\kubernetes_demo\k8_with_volumes\persistent_volume



NETWORKING

POD INTERNAL COMMUNICATION

- Containers within pod can communicate to each other by ‘localhost’
- Decouple environment specific information from container images, so that application is easily portable across environment
- Imagine database host value as localhost in your machine and something else on development or production environment, can be incorporated in config map

\home\chandra\docker-kubernetes-learnings\kubernetes_demo\networking\pod_internal

POD 2 POD COMMUNICATION

❑ Pod to Pod communication can be established either with auto generated service name or DNS

\home\chandra\docker-kubernetes-learnings\kubernetes_demo\networking\pod_2_pod

SECRETS

- ❑ Secrets can be defined as Kubernetes objects used to store sensitive data such as username and passwords with encryption
- ❑ Can help to externalize authentication information of external application which pod may need to access

\home\chandra\docker-kubernetes-learnings\kubernetes_demo\k8_secret

- ❑ Following example demonstrate pod-2-pod communication, with mongo db container in place.

\home\chandra\docker-kubernetes-learnings\kubernetes_demo\networking\pod_2_pod_with_mongo

FURTHER LEARNING

Kubernetes Objects like StatefulSet, Jobs , Cron Jobs, ReplicationSet, ReplicationController

Try to deploy mongo as StatefulSet

Try to read about Cloud provider kubernetes services (EKS, AKS)

Try to read about Openshift

SUMMARY

We learnt basic features of Kubernetes.

But there is lot to kubernetes, which users can learn. New additions keep happening to this orchestration tool and user should keep himself updated.

Openshift is a tool by RedHat and make operating Kubernetes easier. User can learn about Openshift.