Intelligence Of Things®

GLOBAL EDGE

•ALTRAN GROUP

Docker

Docker Contents

- Use of Docker
- Difference between Docker and Virtual Machine
- Docker Architecture
- Creating the Docker Hub repository and push your docker image
- Use of Multi Stage build
- Network drivers
- Enable IPV6 support
- Container Networking
- Disadvantage of Container

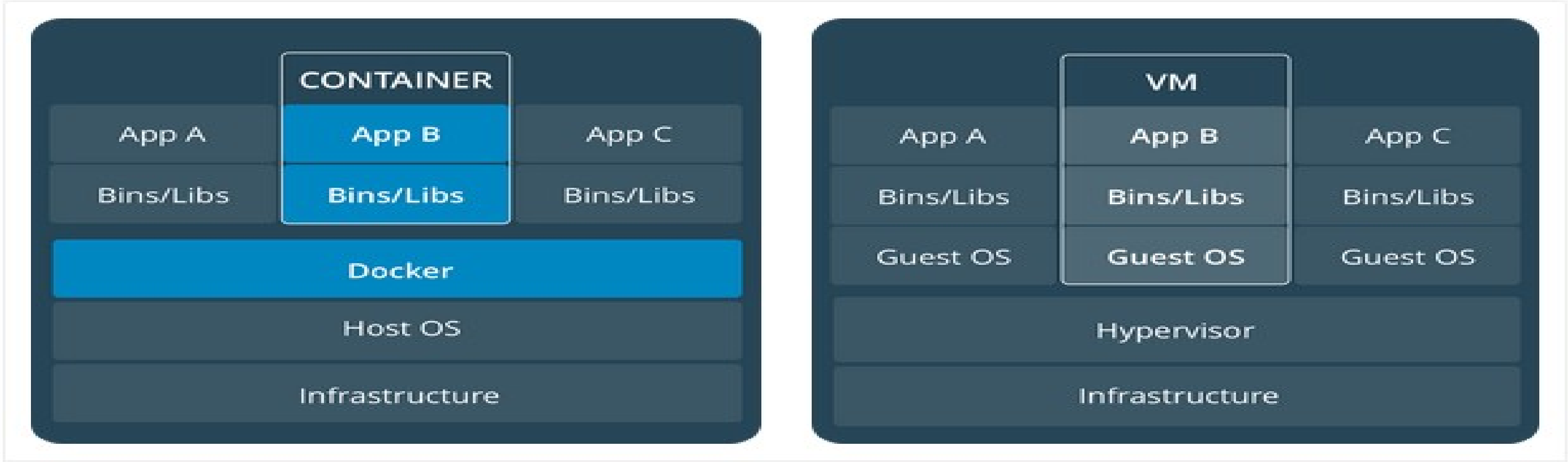
Use of Docker

- The Docker project’s main intent is to allow developers to create, deploy, and run applications easier through the use of containers.
- Docker provides the ability to package and run an application in a loosely isolated environment called a container.
- Containers, built from container images, make it possible to bundle an application up with all the required libraries, dependencies, and resources for easy deployment.
- Each instruction in a Dockerfile creates a layer in the image. When you change the Dockerfile and rebuild the image, only those layers which have changed are rebuilt. This is part of what makes images so lightweight, small, and fast, when compared to other virtualization technologies.

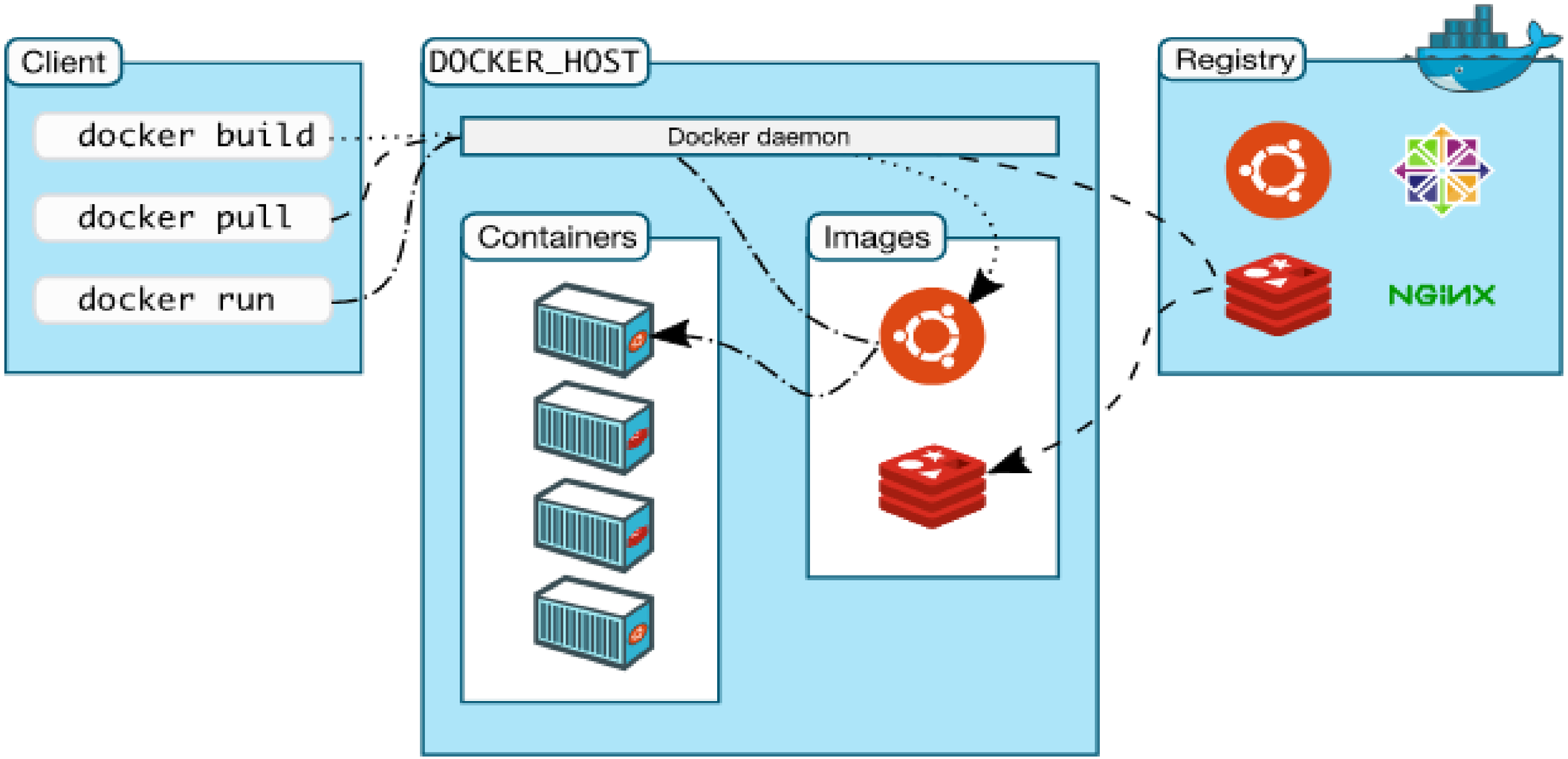
Containers and virtual machines

A container runs natively on Linux and shares the kernel of the host machine with other containers. It runs a discrete process, taking no more memory than any other executable, making it lightweight.

By contrast, a virtual machine (VM) runs a full-blown “guest” operating system with virtual access to host resources through a hypervisor. In general, VMs incur a lot of overhead beyond what is being consumed by your application logic.



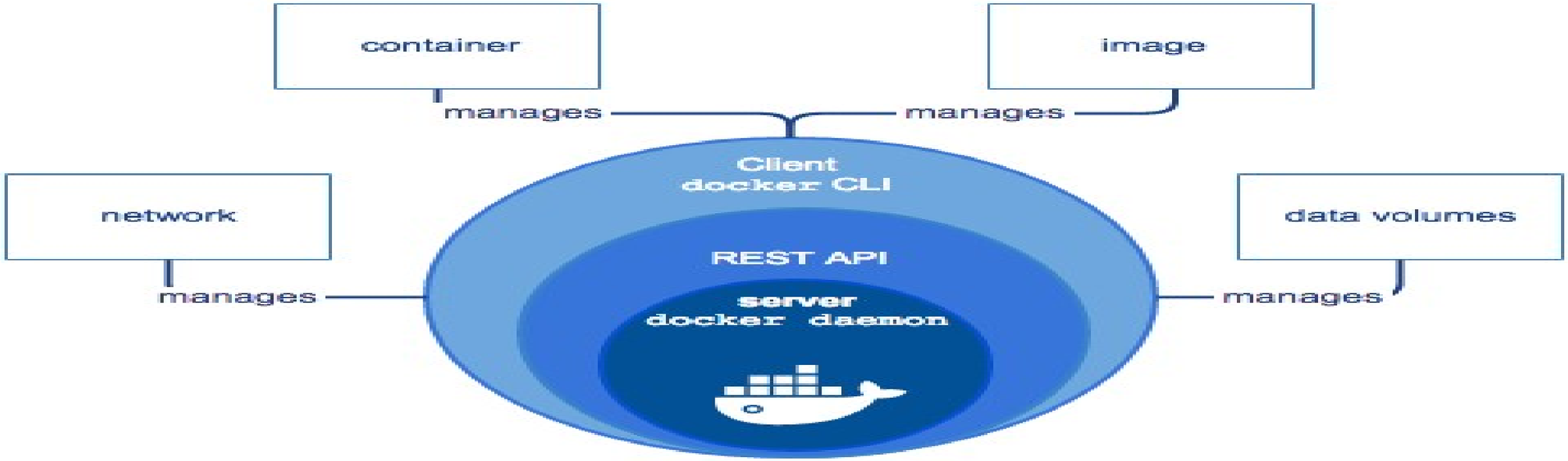
Docker architecture



Cntd . . .

The CLI uses the Docker REST API to control or interact with the Docker daemon through scripting or direct CLI commands. Many other Docker applications use the underlying API and CLI.

The daemon creates and manages Docker objects, such as images, containers, networks, and volumes.



Cntd . . .

- The Docker daemon

  The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. A daemon can also communicate with other daemons to manage Docker services.

- The Docker client

  The Docker client (docker) is the primary way that many Docker users interact with Docker. When you use commands such as docker run, the client sends these commands to dockerd, which carries them out. The docker command uses the Docker API. The Docker client can communicate with more than one daemon.

- Docker registries

  A Docker registry stores Docker images. Docker Hub is a public registry that anyone can use, and Docker is configured to look for images on Docker Hub by default. You can even run your own private registry. If you use Docker Datacenter (DDC), it includes Docker Trusted Registry (DTR).

Creating a Docker Hub repository and push your image Docker

- Visit the Docker Hub sign up page, https://hub.docker.com/signup.
- Fill out the form and submit to create your Docker ID.
- Verify your email address to complete the registration process.
- Click on the Docker icon in your toolbar or system tray, and click Sign in / Create Docker ID.
- Fill in your new Docker ID and password. After you have successfully authenticated, your Docker ID appears in the Docker Desktop menu in place of the ‘Sign in’ option you just used.

Use multi-stage builds

With multi-stage builds, you use multiple FROM statements in your Dockerfile. Each FROM instruction can use a different base, and each of them begins a new stage of the build. You can selectively copy artifacts from one stage to another, leaving behind everything you don’t want in the final image. To show how this works, let’s adapt the Dockerfile from the previous section to use multi-stage builds.

```
FROM golang:1.7.3
WORKDIR /go/src/github.com/alexellis/href-counter/

FROM alpine:latest
RUN apk --no-cache add ca-certificates
WORKDIR /root/
```

Network drivers

- User-defined **bridge** networks are best when you need multiple containers to communicate on the same Docker host.
- Overlay networks are best when you need containers running on different Docker hosts to communicate, or when multiple applications work together using swarm services.
- Macvlan networks are best when you are migrating from a VM setup or need your containers to look like physical hosts on your network, each with a unique MAC address.
- Third-party network plugins allow you to integrate Docker with specialized network stacks.

Enable IPv6 support

- Before you can use IPv6 in Docker containers or swarm services, you need to enable IPv6 support in the Docker daemon. Afterward, you can choose to use either IPv4 or IPv6 (or both) with any container, service, or network.

  Edit the daemon.json file in the pathe /etc/docker/daemon.json

  ```
  {
    "ipv6": true
  }
  ```

  systemctl reload docker

Container networking

- From the container’s point of view, it has a network interface with an IP address, a gateway, a routing table, DNS services, and other networking details (assuming the container is not using the none network driver).

- Published ports

  when you create a container, it does not publish any of its ports to the outside world. To make a port available to services outside of Docker, or to Docker containers which are not connected to the container’s network, use the --publish or -p flag. This creates a firewall rule which maps a container port to a port on the Docker host.

  If you to connect the containers to the different networks use the command

  - Docker network connect -p 8080:80/udp

Problems with scaling up the Containers

- Containers could not communicate with each other
- Containers had to be deployed appropriately
- Containers had to be managed carefully
- Auto scaling was not possible
- Distributing traffic was still challenging

A Container Management Tool

- Kubernetes is an open-source Container Management tool which automates container deployment,container (de)scaling & container load balancing
- Kubernetes was written on Golang, it has a huge community because it was first developed by Google & later donated to CNCF
- Can group ‘n’ no of containers into one logical unit for managing & deploying them easily.

Features Of Kubernetes

- Automatic Binpacking

  Places the Automatic Container based on the requirement and resources.

- Service Discovery & Load Balancing

  If we use the Kerbernetes there is no need to worry about the Networking and Communication because Kubernetes will automatically assign Container there own ip address and single dns name for a set of Containers which are performing a logical aperation and also there will be Load balancing across them

- Storage Orchestration

  With Kerbernetes we can automatically mount our storage system we can choice that can be local or public cloud providers or even a network storage system such as nfs.

Cntd . . .

- Self Healing

  Whenever Kubernetes realizes that one of your containers failed it will restart that container on its own I will create another container in the crashed one. In case node itself failed in this situation what kubernetes will do what ever containers are running on this failed node will be restarted in other node.

- Batch Execution

  Along with your service kubernetes will manage your Batch and CI work loads which will helpful in Devops role.

- Secret & Configuration Management

  We can deploy and update your secret & application Configuration without rebuilding the entire image and without having to expose your secret in stack configuration or any or any thing

Cntd . . .

- Horizontal Scaling

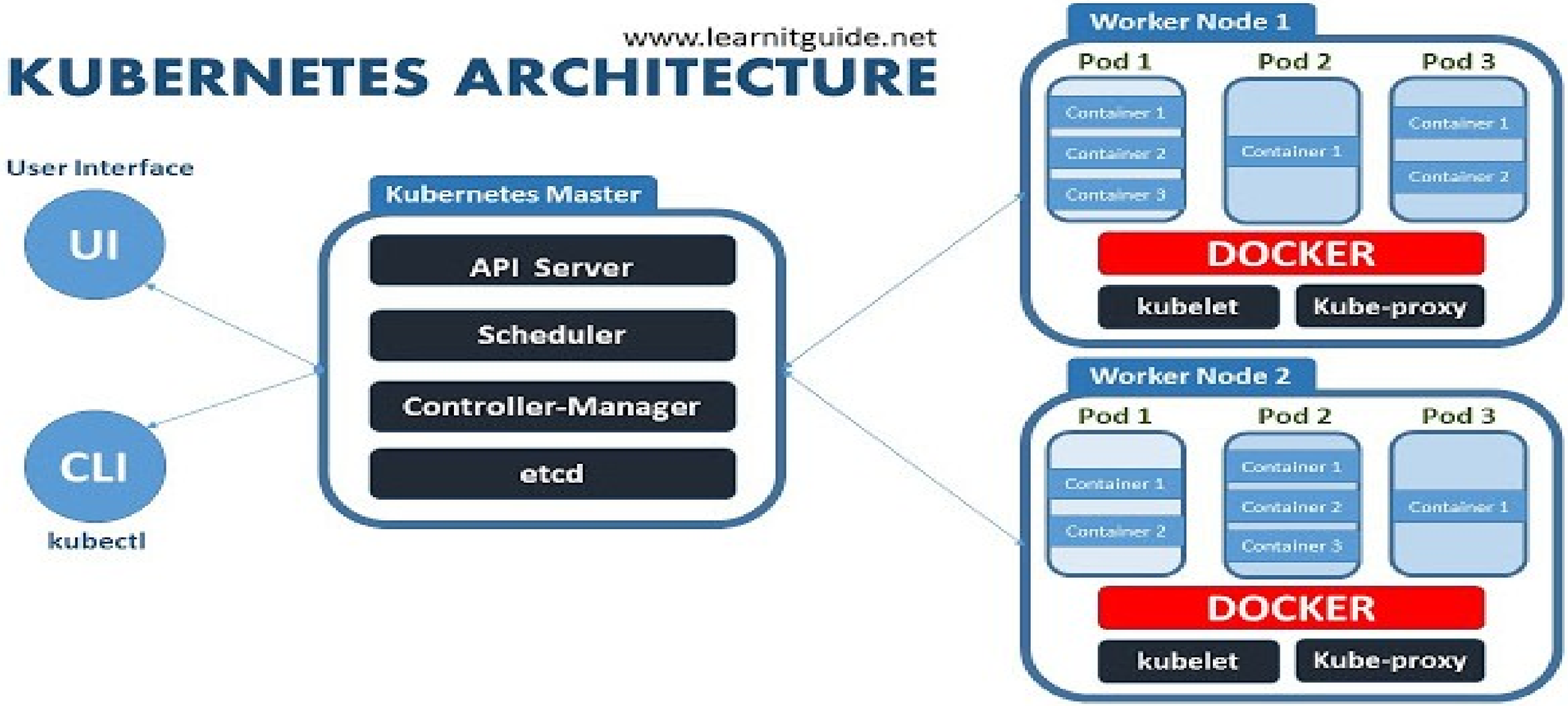
  We can scaling your applications up and down easily with the command that can be run on the cli or can be done easiy with GUI or automatic scaling is possible.

- Automatic Rollbacks & Rollouts

  When ever there is a update which you want to release kubernets progressively rollsout this changes and updates to the applications or its configurations.

-

Kubernetes Architecture



Cntd . . .

- The Kubernetes Master consists of Scheduler , API Server & Controller Manager
- API Server

  API Server is the gate keeper for the entire cluster if we want to create ,update ,delete or even display any kubernetes object it has goes through the API Server
- Scheduler

  It is responsible for physically scheduling pods across the nodes inside the cluster
- Controller Manager

  Controller is responsible for overall health of entire cluster it ensures nodes are up and running and current no of pods are running as mentioned in the configuration file
-

Cntd . . .

- Etcd

  etcd is going to be kubernetes central data base to store entire kubernetes configuration include objets it is creating ,secrets and more .It is the key value database any component of kubernetes enquiring in etcd to understand the state of the cluster in real time.

- Node Components
- In Kubernetes worker node we have four more important components. They are
- Kubelet
- Kube-Proxy
- Pod
- Containers

Cntd . . .

- Kubelet

  It is the primary node agent that runs on each worker node inside the cluster it looks at pods speaks that was submitted to the API Server on the kubernetes master and ensures that the container are discribing in the pods spec are running and healthy incase kubelet notices any failures with the pods running on this worker node then it tries to restart the pod on the same node

- Kube-Proxy

  kube-proxy is a creatical element inside the cluster it is responsible for manintaing the entire network configuration it assentially maintains the distributed network across all the node,all the pods and across all the containers it also exposes the service outside world .

- Pod

  Pod is basically schduling unit in kubernetes.So each Pod consists of one more containers in most cases there will one container.so the primary advantage of pod is

Cntd . . .

We can deploy multiple dependent container together.So it acts as a wrapper around this containers

- Containers

  It provides the run time environment for your applications.So you run containerised applications inside this containers.

The steps to install the Kubernetes

- Update the repository

  Sudo apt-get update

- Turn off the Swap Space and SELinux on all nodes

  setenforce 0

  swapoff -a

  nano *etc*/fstab

- Update the Hostname

  nano *etc*/hostname

- Note the IP address

  ifconfig

- Update the hosts file

  nano etc/hosts

Cntd . . .

- Set a static IP address

  nano etc/network/interfaces

- Add the below at the end of the file

  auto <interface_name>

  iface <interface_name> inet static

  address <IP_Address_of_VM>

- Install the OpenSSH Server

  sudo apt-get install openssh-server

- Install Docker

  sudo su

  apt-get update

  aptg-get install -y docker.io

Cntd . . .

- Run the following commands before Installing the kubernetes environment

  apt-get update && apt-get install -y apt – transport – https curl

  curl -s https://packages.cloud.google.com/apt/doc/apt - key.gpg | apt-key add -

  cat <EOF>/etc/apt/sources.list.d/kubernetes.list

  deb https://apt.kubernetes.io/ kubernetes-xenial main

  EOF

  apt-get update

- Install kubeadm,kubelet & kunectl

  apt-get install -y kubelet kubeadm kubectl

- Update the kubernetes configuration

  nano *etc*/systemd/system/kubelet.service.d/10-kubeadm.conf

Cntd . . .

- Add the below after the last line

  Enironment=”<group.driver-systemd/<group.driver-cgroupfs”

Configuring Kubernetes by Kubeadm

- Create Vms which are part of K8s cluster (master & worker nodes)
- Disable SELinux and SWAP on all nodes
- Install kubeadm,kubelet ,kubectl and Docker on all nodes

  start and enable docker and kubelet on all nodes
- Initialize the master node
- Configure Pod network
- Join worker nodes to the cluster

Kubeadm

- Kubeadm init //on master

  kubeadm init [flags]

- Kubeadm join   //on worker

  kubeadm join –token [] --discovery-token-ca-cert-hash []

- Kubeadm token

  kubeadm token [create|delete|list|generate] [flags]

- Kubeadm version

  kubeadm version [flags]

- Kubeadm upgrade

  kubeadm upgrade plan [version] [flags]

Cntd . . .

- Disabel SWAP on all nodes

  swapoff -a

- Disable SELinux on all nodes

  setenforce 0

  sed -i ‘s/enforcing/disabled/g’ *etc*/selinux/config

  grep disabled *etc*/selinux/config | grep -v ‘#’

- Reboot all nodes
- Install Docker

  yun update -y

  yum install -y docker

Cntd . . .

- Start and enable docker

  systemctl start docker

  systemctl enable docker

  systemctl status docker

- Add kubernetes Repo:

  cat <<EOF > *etc/yum.repos.d/kubernetes.repo*

- Install kubelet,kubeadm,kubectl and start kubelet

  yum install -y kubeadm kubelet kubectl –disableexcludes=kubernetes

  systemctl enable kubelet && systemctl start kubelet

Cntd . . .

- This is used to fixing the issues

  cat <<EOF > *etc*/sysctl.d/k8s.conf

  net.bridge.bridge-nf-call-ip6tables = 1

  net.bridge.bridge-nf-call-iptables = 1

  EOF

  sysctl --system

- Only on **master** node

  kubeadm init –pod-network-cidr=10.240.0.0/16

  kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/v0.9.1/Documentation/kube-flannel.yml

Cntd . . .

- Kubectl get pods --all-namespaces
- Only on **worker** nodes

  kubeadm join –token

Thank You!

GLOBAL EDGE
Intelligence Of Things®
•ALTRAN GROUP

India: Bengaluru, Hyderabad | US: California

| www.globaledgesoft.com

FAIRNESS • LEARNING • RESPONSIBILITY • INNOVATION • RESPECT