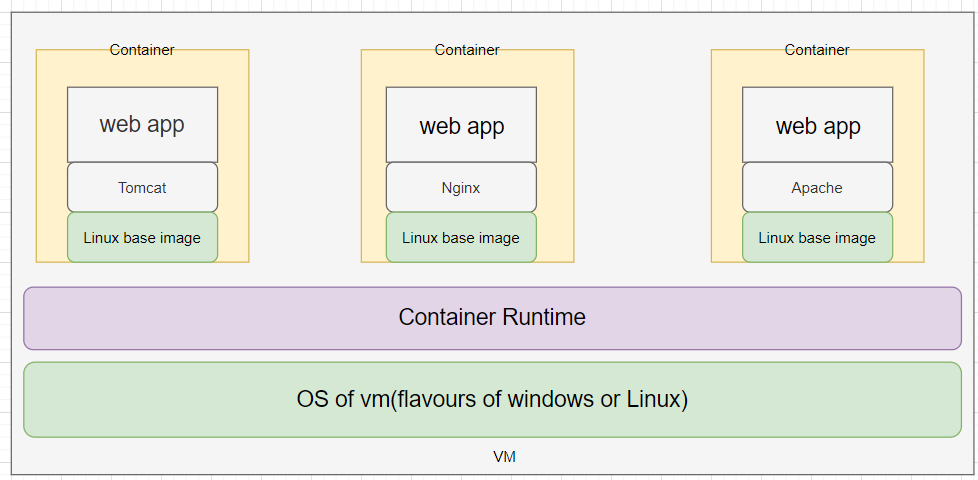
1. **DOCKER**
2. Introduction:

Traditionally in order to deploy any application on the cloud vms were used. But the drawback was that it was not possible to deploy multiple applications with different dependencies on the same vm. And another major problem was that applications were not working the same in all environments. It was also difficult to maintain configurations in case multiple micro services were used. In order to overcome all these problems and pitfalls containers were introduced. Docker is a company that has implemented the concept containers.

1. Containers:

Application and all its dependencies put together are known as a container. Multiple containers can be mad to run on a single VM as long as they belong to the same kernel either Windows/Linux. Containers can be built as a windows container or a Linux container. And applications will behave the exact same way in all environments. It is also very easy to maintain micro services along with their configurations.



1. Container Image:

A container image is a lightweight, standalone, executable piece of software that includes everything needed to run an application (code, runtime, system tools, system libraries and settings).

A container image becomes a container instance at runtime.

Docker images can be created using docker file. A docker file is used to specify the contents of a container and their configuration in other words docker file is a set of instructions to build the docker image. These images can be stored in repositories like Dockerhub, Azure Container Registry or Git hub and pulled whenever needed.

1. Task 1:

Dockerize a static html page and deploy the image.

In order to dockerize an application a docker file has to be written

Dockerfile:

FROM nginx:alpine // Specify the base image of the container

COPY . /usr/share/nginx/html //Copy code and files to the container

EXPOSE 80 // Container listens to the specified port at run time

Building and running the docker file:

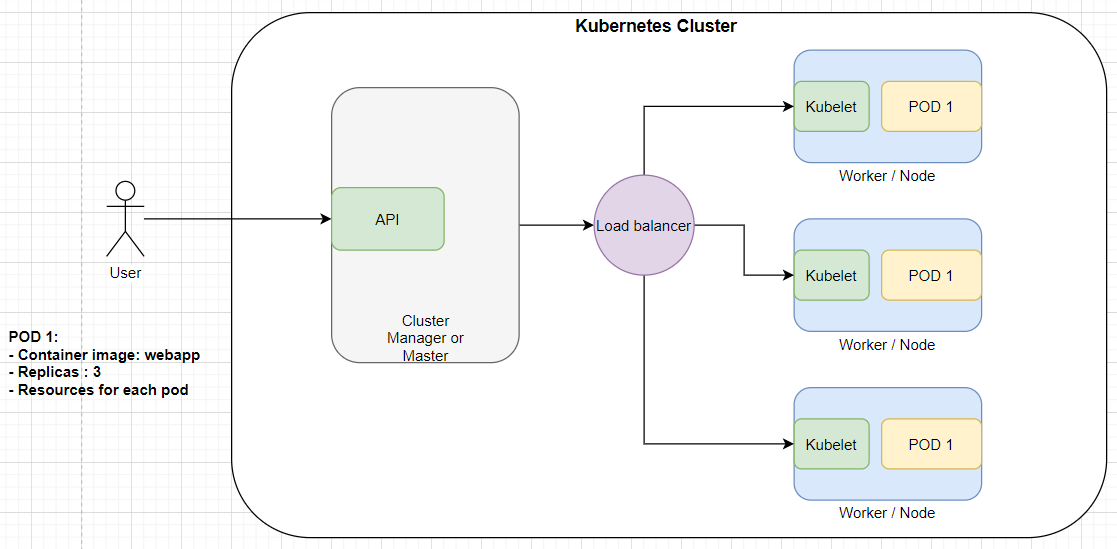
* docker build img\_name . // builds docker image form docker file
* docker images // list all images
* docker run –d -- name any\_name –p 80:80 image\_name // run an image
* docker ps // lists all running containers

1. **KUBERNETES**
2. Introduction:

Kubernetes is a system for running and coordinating containerized application across a cluster of machines. K8S can be used to manage the lifecycle of containerized applications, scalability and also ensures high availability. We can define how our application should interact with other applications and with the outside world. We can scale applications up or down, perform rolling updates and switch traffic between different versions of the application to test features.

1. Kubernets architecture:

Kubernetes puts together individual physical or virtual machines into a cluster using a shared network to communicate with each other. All kubernets components and workloads are configured in the cluster.



There are two major components of a kubernetes cluster, the master and the nodes. The **master** is like the brain of the k8s cluster it exposes API for the clients, checking the health of the nodes, splitting up assigned tasks and managing communication between components. It takes care of handling deployments and managing worker nodes. Azure provides cluster manager for free of cost.

**Nodes** are machines in which the work loads are run i.e. nodes are vms in which actual deployment happens. Applications are run as containers, so each node must have a container runtime. Kublet is a service that is running in the nodes that is responsible for taking up tasks from the master and creates or destroys containers accordingly. K8s is responsible for maintaining desired state configuration.

To deploy an application YAML files are used. These files contain information like what to create and how it should be managed in other words this file is the desired state configuration. The master node takes up this plan and decides how to run it in the existing infrastructure.

1. Master Server Components:
2. etcd:

etcd is used to store configuration data that can be accessed by all nodes of the cluster. It helps in maintaining cluster state. It helps services to configure according to up-to-date information.

1. Kube-apiserver:

It helps to assign workloads to nodes analyzing the current infrastructure and places work on acceptable node.

1. Cloud-controller-manager:

K8s can be deployed in many different environments and can interact with various infrastructure providers to manage state of resources. Cloud controller manager acts as a bridge that allows K8s to interact with various providers with different features and APIs, this allows K8s to update its state information according to information gathered from the cloud provider.

1. Node Server Components:
2. Container Runtime:

Container Runtime is a mandatory component in every node that is made available when docker is installed. It helps managing the containers.

1. Kubelet:

Kubelet is a service that runs in all nodes that acts as a contact point for the master. Kubelet receives commands from the master and begins the work, it also interacts with etcd to maintain states. Kubelet process takes responsibility for maintaining state of work in the node. It controls the container runtime to launch and destroy containers as needed.

1. Kube-proxy:

Kube-proxy navigates requests to the correct containers in the node it can perform a basic level of load balancing. It makes sure that services are available to other components.

1. Kubernetes Objects and Workloads:

Kubernetes uses additional layers of abstraction over containers to provide features like scaling, life cycle management.

1. Pods:

A pod is the most basic unit of kubernetes. A pod is generally one or more tightly coupled containers that are to be deployed in the same node. Pods are fundamental units of deployment. Container instances are not deployed directly, when pods are deployed container instances are a part of that pod.

1. Replication Controller and Replication Sets:

These are used to horizontally scale applications. Replication defines a pod template and parameters to deploy identical pods. It can be used to either increase or decrease the number of running nodes. Controller is responsible for maintaining the specified number of pods or instances running. If one instance fails controller automatically start new pod to meet the desired state configuration.

1. Services:

A service groups logical collections of pods to represent them as a single entity. This allows us to deploy a service that can monitor all containers of a particular type. IP address remains the same regardless of changes pods it route to. In order to give access to a pod to other applications or the end user a service has to be configured. the **Load Balancer** service type creates an external load balancer to route to the service using a cloud provider’s Kubernetes load balancer integration. The cloud controller manager will create the appropriate resource and configure it using the internal service service addresses.

1. Terraform:

Terraform is a platform that helps maintain and deploy infrastructure as code. Terraform is an open source tool and can be used to provision resources in almost all cloud platform. Spinning up a kubernetes cluster is more practical when done using terraform because the code can be reused and if a similar setup is required it can be done in no time. Using Terraform plan we can know what are the changes that is going to be made even before actually making the deployment. Resources are deployed based on resource dependencies.