Orchestration

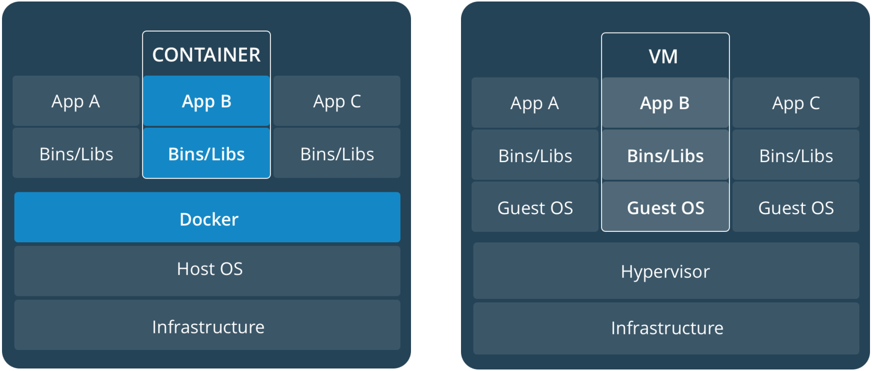
Tony Espinoza

am.espinoza@utexas.edu

Docker overview

- Docker is not a virtual machine
- Docker is a containerization system.
    - Runs on your OS natively

Docker VS Virtual Machine[1]



[1] https://docs.docker.com

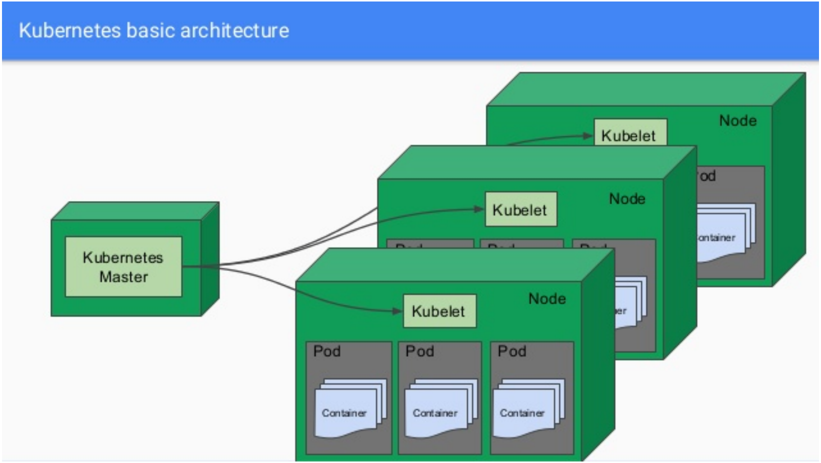
Kubernetes

- Orchestration software
  - Deployment
  - Management
  - Scaling

Terminology

- Pod
  - One or more containers on a machine.
  - Smallest deployable unit.
- Node
  - Is the worker machine.
  - Nodes run pods.
  - Kubelet runs in a node to monitor pods.
- Master
  - Coordinates all activity in your cluster.
  - Communicates with kubelet.
- yaml
  - Configuration file
  - Yet Another Markup Language

Layout



Kubernetes basic architecture

Deployment

- Kubernetes is software that aids in the deployment of containers (we'll use docker).
- Can specify how to deploy in detail.
    - How many instances.
    - What services.
    - Layout.
    - Resources.
    - Exposed ports.
    - All with a yaml.

yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webserver
  labels:
    app: apache
spec:
  replicas: 3 #how many webservers to deploy
  selector:
    matchLabels:
      app: apache
  template:
    metadata:
      labels:
        app: apache
    spec:
      containers:
      - name: php-apache
        image: localhost:32000/website:k8s
        imagePullPolicy: Always
        ports:
        - containerPort: 80
```

Management

Kubernetes master node:

- Manages networking between nodes.
- Communication between nodes.
- In event of a crashed pod:
    - Kubernetes will start a new instance.
    - Pods are monitored by kubelets
    - Kubelets: service monitor for a Node.

Kubelets

- Keep track of pods in the node.
- Communicate with the master node.
- Helps the master node to keep the cluster a reflection of the yaml file.

Scaling

- Kubernetes can be scaled to work across systems.
- Load balancing
  - Balance access across containers (duplicate).
  - Spin up new machines under heavy load.

Storage

- Like docker, Kubernetes does not have persistent storage.
    - You must set up storage separately.
- Ever new instance is fresh.

Volumes

- Is the way you create persistent storage.
- In the container section of the yaml file specify mount point.

Volumes

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pd
spec:
  containers:
  - image: k8s.gcr.io/test-webserver
    name: test-container
    volumeMounts:
    - mountPath: /test-pd #inside the container
      name: test-volume
  volumes:
  - name: test-volume
    hostPath:
      # directory location on host
      path: /data #on the host machine
      # this field is optional
      type: Directory
```

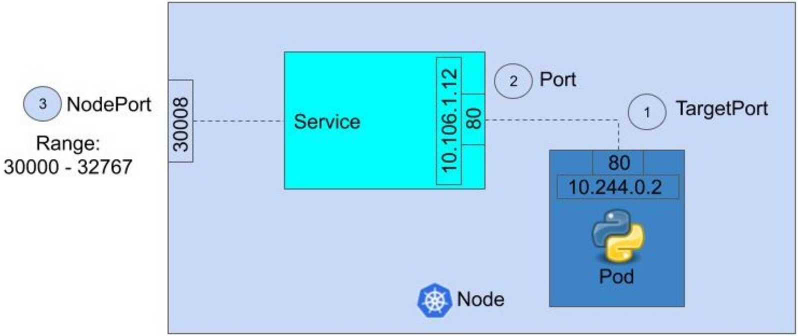
Volumes

- Can be shared across pods.
- Can set capacity.
- Other specifications (access modes R,W . . .)

Networking

- Pods have 3 types
    - Load balance
        - Does load balancing.
    - Node port
        - Exposes the application on a port across each of your nodes
    - Cluster

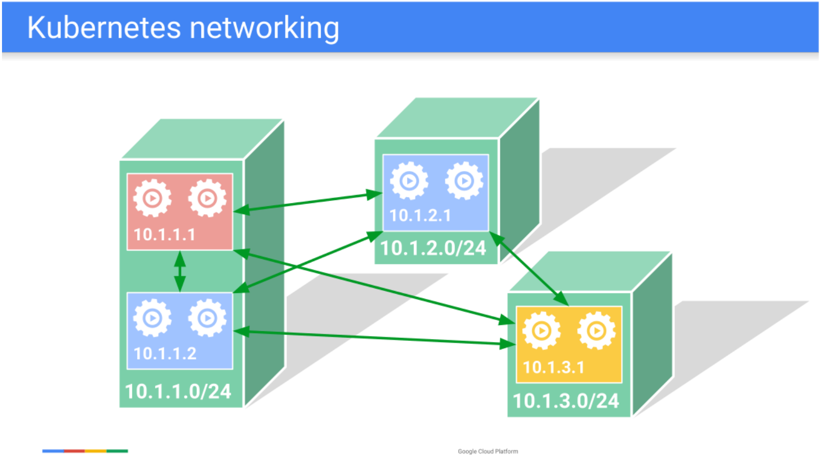
Networking


3
NodePort
Range:
30000 - 32767
30008
Service
10.106.1.12
80
2
Port
1
TargetPort
80
10.244.0.2
Pod
Node

Networking

- Every pod has its own unique IP
- Containers in a pod share namespaces
  - Does this mean that they have the same view of the network?

Networking



Kubernetes networking

Google Cloud Platform

Role based access control

- There are users, and service accounts.
    - RBAC allows us to limit what resources are available and what they can do to those resources.
- Normal users assumed to be managed by outside independent service.
- Service account, managed by Kubernetes.

RBAC

- Verb: get, list, create, delete. . .
- Resources: pod, volume, secret, service, endpoint. . .

RBAC

Two types of roles.

- Namespace
  - Can do RBAC limiting namespace
- Cluster
  - Can do RBAC limiting clusters
- RBAC Kubernetes manual

RBAC

Implement RBAC in two steps

1. Create a role with a list of rules.
2. Bind the created role to a user or service account.