Orchestration

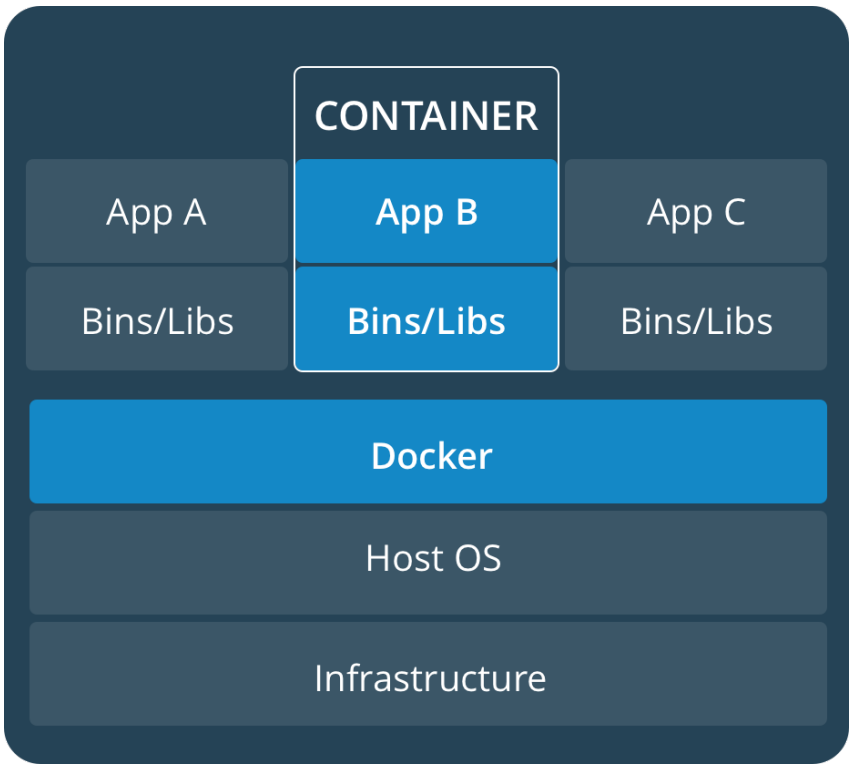
Tony Espinoza

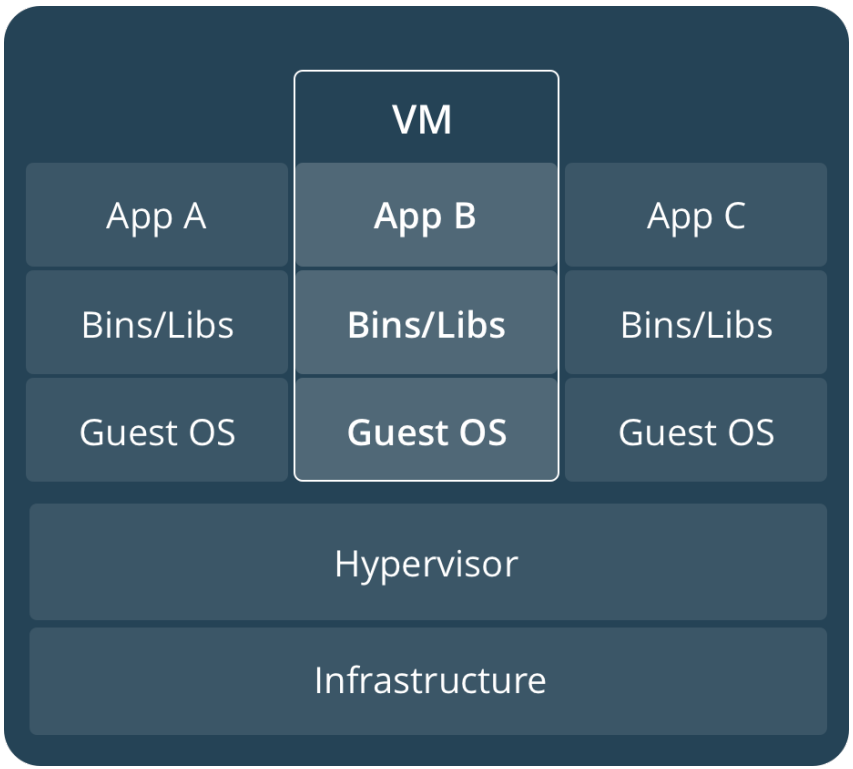
am.espinoza@utexas.edu

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

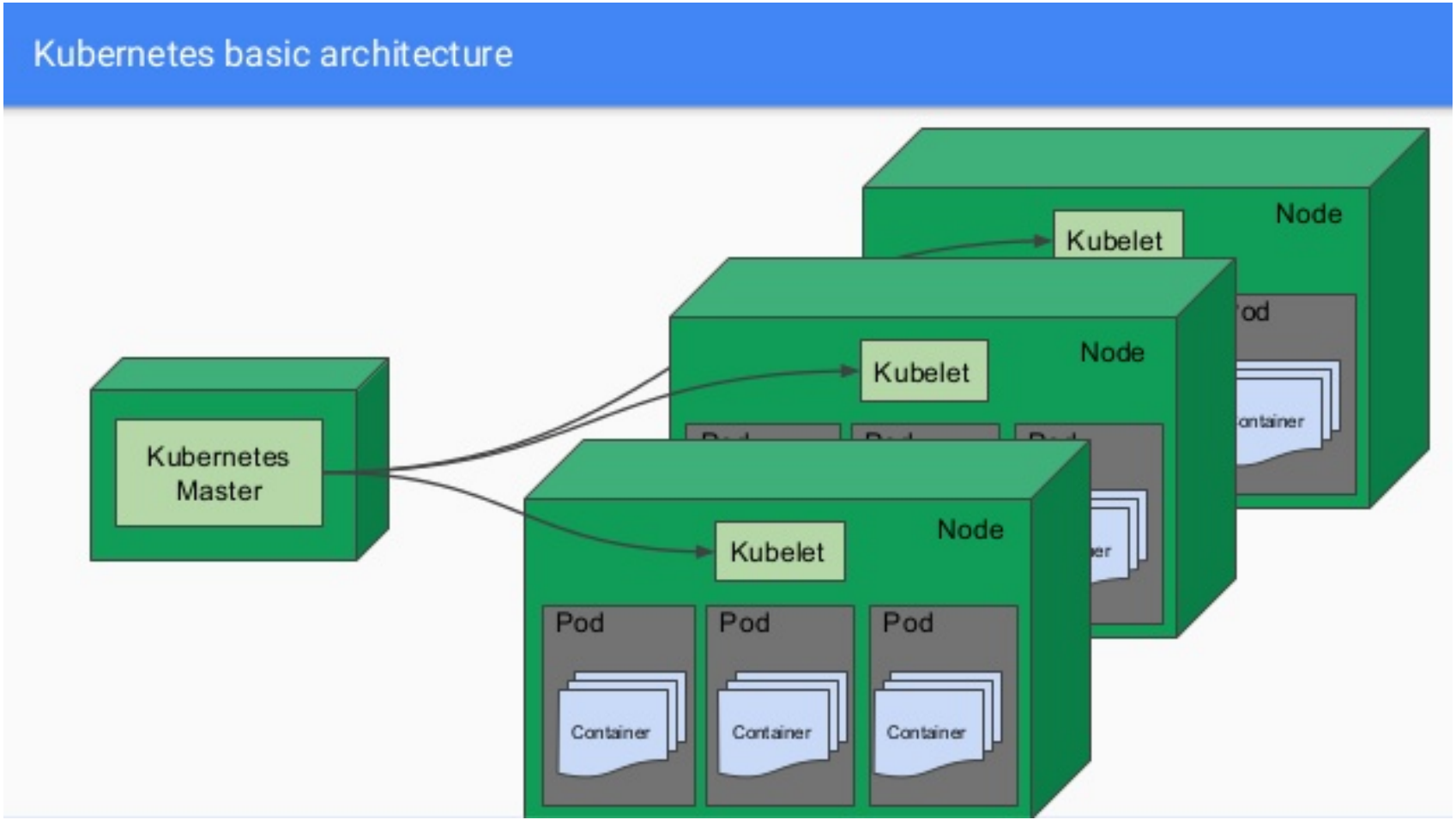
Kubernetes

- 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
Kubernetes Master
Kubelet
Node
Kubelet
Node
Kubelet
Node
Pod
Pod
Pod
Container
Container
Container

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.
- Every new instance is fresh.

Volumes

- Volumes are 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

- All pods and nodes are networked together.
- 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?

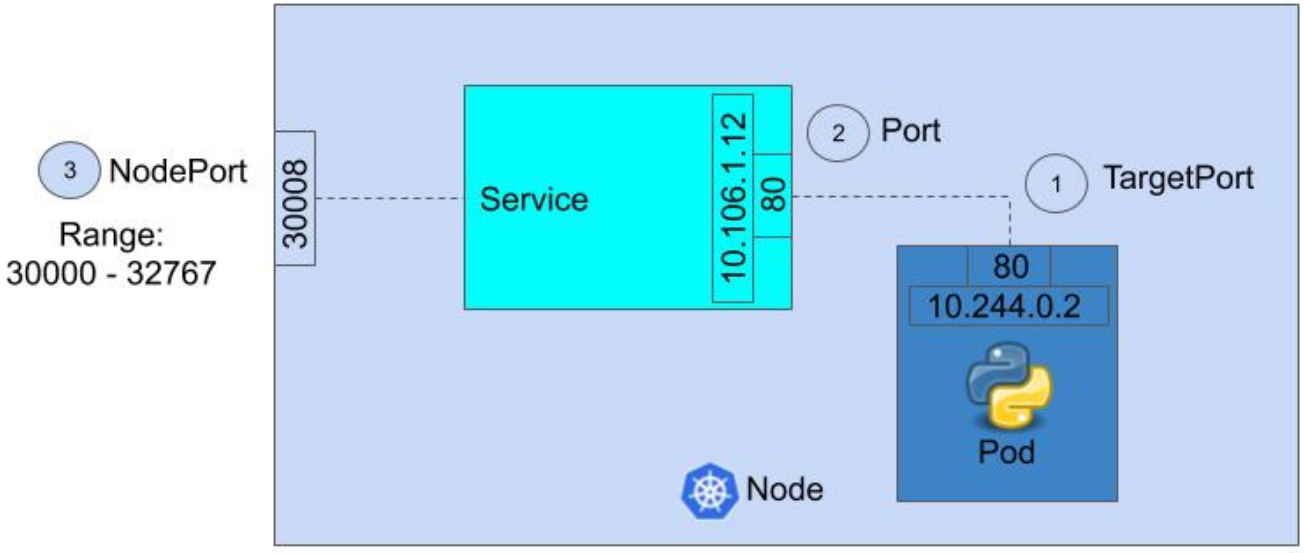
Networking

- Pods have 3 service types
  - Node port
    - Exposes the application on a port across each of your nodes.
  - Load balance
    - Does load balancing.
  - ClusterIP
    - Virtual IP inside the cluster to enable communication between services.

Networking

- A NodePort service is associated with 3 ports:
  - Node port
  - Target port
  - Port
- All ports are from the perspective of the service.

Networking

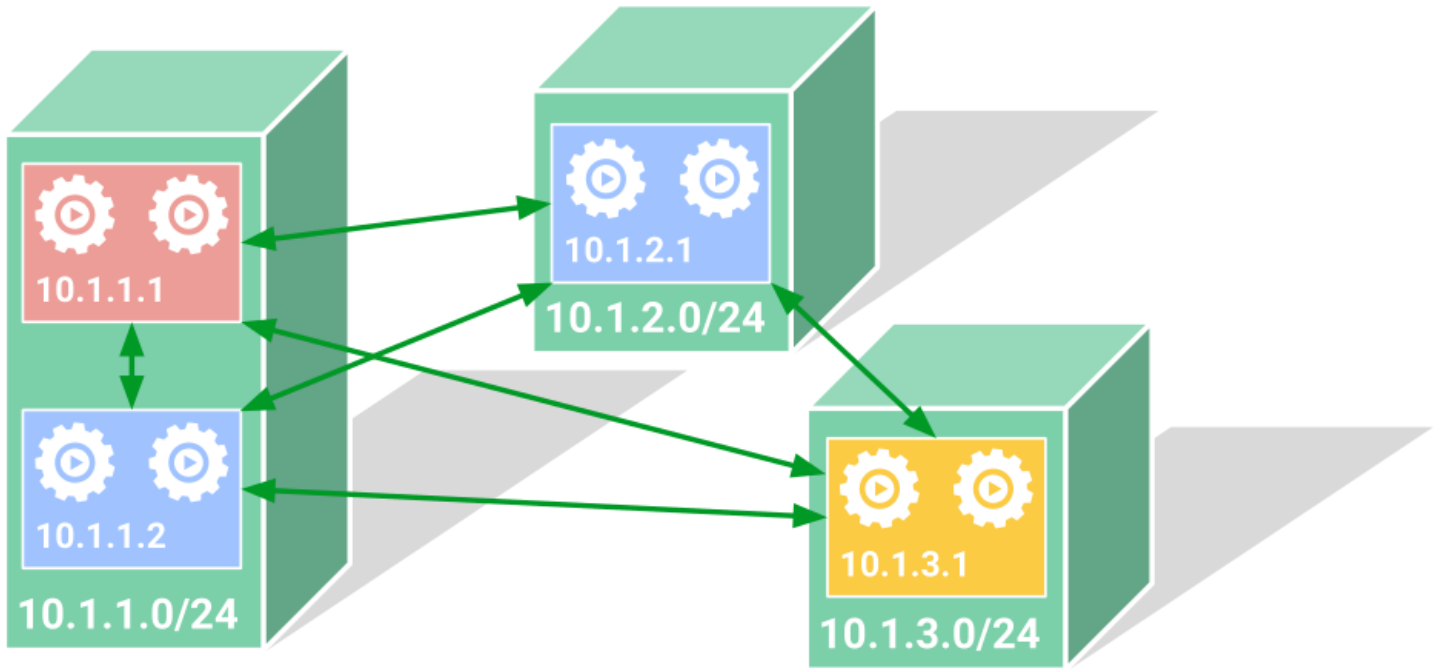

3
NodePort
Range:
30000 - 32767
30008
Service
10.106.1.12
80
2
Port
1
TargetPort
80
10.244.0.2
Pod
Node

NodePort

- If the type is NodePort you may not see all thee port types defined.
- Target port is assumed to be the same as port if not provided.
- If you don't provide a target port a free one is assigned.

Networking

Kubernetes networking



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.

Security of containers.

- Containers are still vulnerable.
  - https://cve.mitre.org/index.html

Security of containers and k8s.

- Container scanners.
  - clair
  - anchore
- Configuration checkers.
  - Docker bench security.
    - The Docker Bench for Security is a script that checks for dozens of common best-practices around deploying Docker containers in production

Monitoring.

- Prometheus.
- microk8s metrics-server.
- Sonobuoy.
- ...

Secrets

- Allow OAuth with secrets.
- Can combine with RBAC and give a user a token allowing them access to only what the need.
- Don't have to give out username and password credentials, can give a token instead.
- RFC 5849.
- Demo.

Monitoring

Today we will cover:

- System monitoring
  - Sysdig
  - Osquery
  - Prometheus
- Container verification
  - Anchore

Monitoring

- You will use OS concepts we reviewed.
  - `/proc`.
  - PID.
  - Namespaces.
  - System calls.

Container verification

Anatomy of a docker file

- All docker files start with:
  - `FROM <docker file name>`
- Docker file can use docker files that use other docker files.

Nginx

If I wanted to use nginx as a base:

Nginx

If I wanted to use nginx as a base:

```
FROM nginx
COPY nginx.conf /etc/nginx/nginx.conf
RUN apk add vim
ADD /src_host_folder /dst_container_folder
....
```

Nginx dockerfile

```
FROM alpine:3.10

LABEL maintainer="NGINX Docker Maintainers <docker-maint@nginx.com>"

ENV NGINX_VERSION 1.17.4
ENV NJS_VERSION   0.3.5
ENV PKG_RELEASE   1A
...
```

Nesting dolls

- We can make a container based on Nginx
- Nginx is based on alpine
- We have 2 levels of indirection of docker containers to check.
  - Each can import packages it need.

More on secrets

- Can make secret environment variables.
- Can make secret values to mount.

Why secrets

- Don't want to put sensitive information into the image when we can put it into the configuration file for the pod.
- Removes the availability of sensitive information.

Create secret

- `kubectl create secret generic secret_name --from-literal=username=devuser --from-literal=password='S!B\*d$zDsb'`
- Made two secrets
  - username
    - devuser
  - password
    - S!B*d$zDsb
- Secret name is secret_name
- From literal allows us to use plain text rather than base 64 encoded stings.

Using secrets

```
apiVersion: v1
kind: Pod
metadata:
  name: secret-env-pod
spec:
  containers:
  - name: mycontainer
    image: sql_db
    env:
      - name: SECRET_USERNAME #Environment variable name
        valueFrom:
          secretKeyRef:
            name: secret_name #name of secret created
            key: username # the key we are using to access devuser
      - name: SECRET_PASSWORD #Environment variable name
        valueFrom:
          secretKeyRef:
            name: secret_name #name of secret created
            key: password # the key we are using to access S!B\*d$zDsb'
  restartPolicy: Never
```

When to use?

- TLS keys
- SQL keys
- SSH keys (to clone a private git repo)
- Anytime you don't want to hard code secrets into an image.

Other creation methods.

- There are a few other ways to create secrets
  - yaml file.
  - manually with base64 encoding.
    - antiquated.
- Can also mount the secrets to a volume.

Falco demo

Click this git link