kubernetes

What is kubernetes?

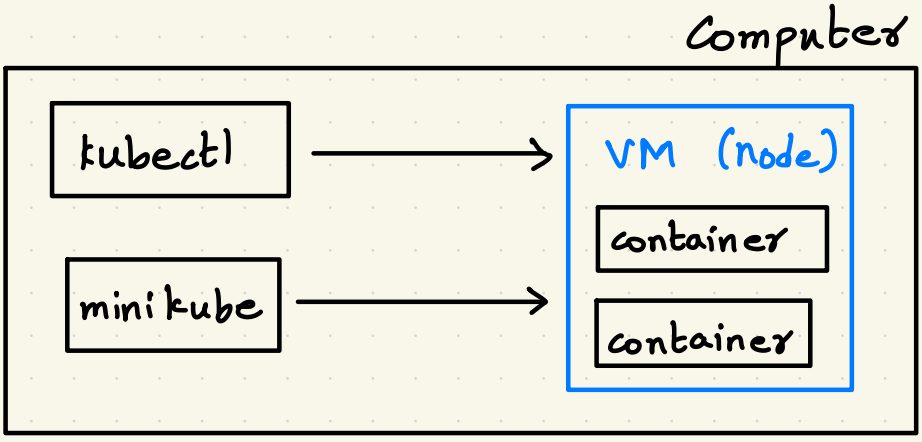
→ System for running many different containers over multiple different machines

Why use kubernetes?

→ When you need to run many different containers with different images.

To run kubernetes,

→ Need Minikube



kubectl → Useful for managing containers in the node.

minikube → Useful for managing the Virtual Machine itself

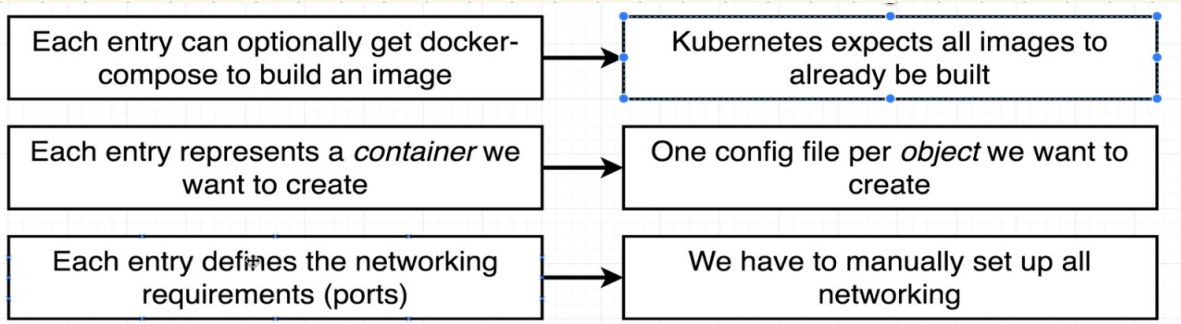
* → Local Only *

> minikube status

→ Check the status of minikube

> kubectl cluster-info

Docker Compose Vs Kubernetes



Get a simple container running on our local Kubernetes Cluster running

1. Make sure our image is hosted on docker hub
2. Make one config file to create the container
3. Make one config file to set up networking

Step 1: Host your docker file on docker Hub

Step 2: Create a config file to create the container

Note: k8s , abbreviation for kubernetes
→ 8 represents 8 letters

→ client-pod.yaml for writing the configurations

Step 3: Create a config file to setup networking

→ client-node-port.yaml

Step 4: Feed .yaml files to kubectl

client-pod.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: client-pod
  labels:
    component: web
spec:
  containers:
    - name: client
      image: stephengrider/multi-client
      ports:
        - containerPort: 3000
```

client-node-port.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: client-node-port
spec:
  type: NodePort
  ports:
    - port: 3050
      targetPort: 3000
      nodePort: 31515
  selector:
    component: web
```

config files

↓ Used to create 'objects'

- statefulSet
- ReplicaController
- Pod
- Service

Few examples of objects

→ Objects serve different purposes,

→ Running a container

→ Monitoring a container

→ Setting up networking, etc..

apiVersion

→ Provides access to predefined objects

→ Different versions contain different objects.

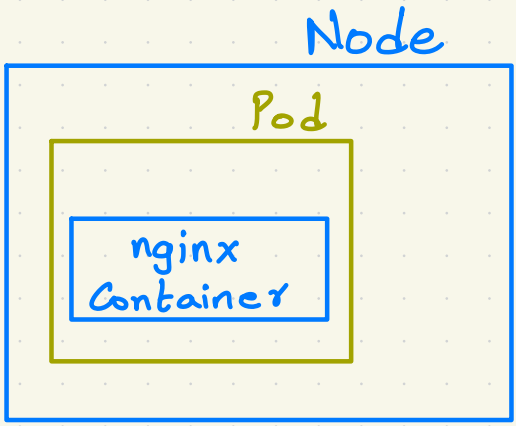
kind

→ Type of object you want to make

Ex: Pod, used to run a container.

Service, used to setup networking.

Pod



>minikube start

→ Creates a virtual machine (node)

→ Node is useful for creating objects

→ Always declare containers inside Pod.

→ Pod contains 1 (or) more containers.

→ Containers inside Pod are tightly coupled.

Service

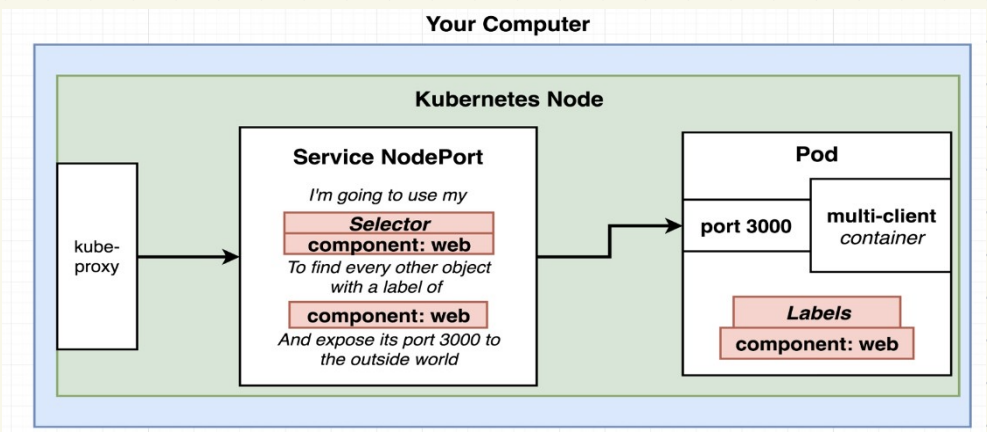
→ Sets up networking in kubernetes.

→ Four sub types

1) Cluster IP
2) NodePort
3) Load Balancer
4) Ingress

Node Port

→ Exposes a container to outside world.



Cluster IP

→ Expose a set of pods to other objects in the cluster.

client-pod.yaml

```
:
labels:
  component: web
:
```

client-node-port.yaml

```
:
selector:
  component: web
:
```

→ labels are used to a label for objects

→ Selector uses the label to assign properties to that object.

Feed a Config file to kubectl

> kubectl apply -f <filename>

- kubectl — CTL
- -f — we want to specify a file
- <filename> — config file path.

Printing the status of Pods

> kubectl get pods

- kubectl — CTL
- get — we want to retrieve
- pods — Specify the object that we want to get

> kubectl get services

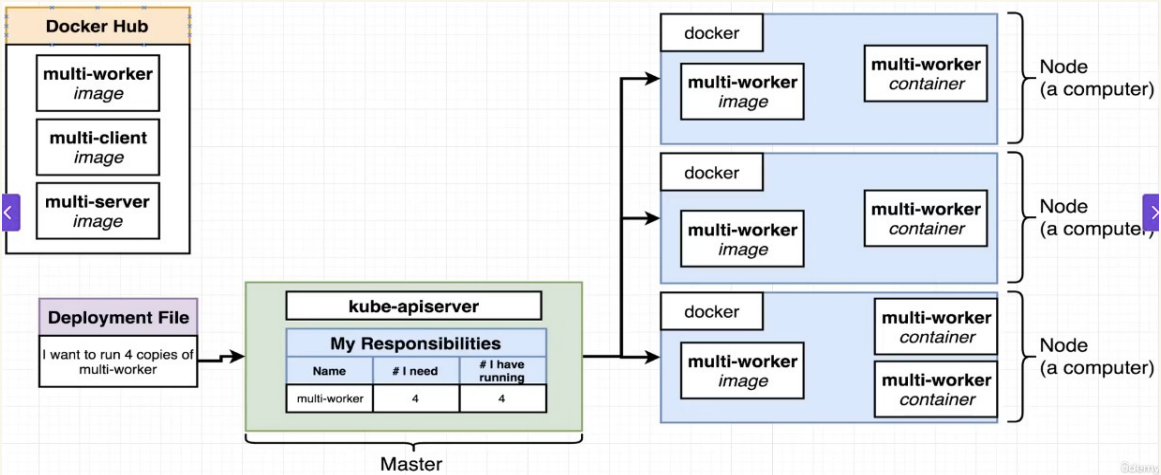
Note: localhost:31515 won't work

→ Everything is running in Virtual Machine

→ get ip address of VM.

> minikube ip

How kubernetes work



Important takeaways

→ kubernetes is a system to deploy containerized apps

→ Nodes are individual machines (or VM's) that run containers

→ Masters are machines (or VM's) with a set of programs to manage nodes

→ kubernetes didn't build images, it gets them from Docker Hub.

→ kubernetes decides where to run each container, each node can run a dissimilar set of containers.

→ To deploy something, we update the desired state of the master with a config file.

→ Master works constantly to meet the desired state.

Imperative Deployments

'Do exactly these steps to arrive at this container setup'

Declarative Deployment

'Our container setup should look like this, make it happen.

To get the detailed info about an object

> kubectl describe <object type> <object name>

kubectl — ctl

describe — want to get the detailed info

<object type> — Object type

<object name> — object name. (optional)

Deployment Object

→ Maintains a set of identical pods, ensuring that they have correct config and the right number of exists.

→ Alternative to pods.

Pods	Deployment
Runs a single set of containers	Runs a set of identical pods (one or more)
Good for one-off dev purposes	Monitors the state of each pod, updating as necessary
Rarely used directly in production	Good for dev
	Good for productino

lets create a client-deployment.yaml

client-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: client-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      component: web
  template:
    metadata:
      labels:
        component: web
    spec:
      containers:
        - name: client
          image: stephengrider/multi-client
          ports:
            - containerPort: 3000
```

Remove an existing Pod

```
> kubectl delete -f <config file>
```

Get IP address of pods

```
> kubectl get pods -o wide
```

→ If there is a newer version of image, Deployment can't detect the change in version after deployment of pods.

Sol 1: Manually delete pods, and let Deployment create new pods with latest version of image.

Sol 2: Add version number to image, whenever there is a change, update config file and apply to ctl.

Sol 3: Use Imperative Command to update the image version.

step 1: Tag the docker image

step 2: Run the command

> kubectl set image <object type> / <object name> <Container_name> = <new image to use>

Configure the VM to use your Docker server

> eval $(minikube docker-env)

→ Only configures current terminal

Volume

Vs

Persistent Volume

Vs

Persistent Volume Claims

(Availability of different volumes)

Access Modes

1) ReadWrite Once

2) Read OnlyMany

3) Read write Many.

Environment Variables

```
spec:
  containers:
    - name: worker
      image: ...
      env:
        - name: REDIS_HOST
          value: redis-cluster-ip-service
        - name: REDIS_PORT
          value: '6379'
        - name: PGUSER
          value: postgres
        - name: PGHOST
          value: postgres-cluster-
                 ip-service
        - name: PGPORT
          value: '5432'
```

Secrets

→ Object to store passwords

→ Object similar to Pods, services, etc

> kubectl create secret generic (Type of secret)

< secret name > --from-literal key=value

> kubectl get secrets

```
- name: PGPASSWORD
  valueFrom:
    secretKeyRef:
      name: pgpassword
      key : PGPASSWORD
```

Load Balancer (works with single Deployment)

Vs

Ingress (works with multiple Deployment)