Deploying Pods

Life of a pod

- Pending : in progress
- Running
- Succeeded : successfully exited
- Failed
- Unknown

Resource Configs

Each entity created with kubernetes is a resource including pod, service, deployments, replication controller etc. Resources can be defined as YAML or JSON. Here is the syntax to create a YAML specification.

AKMS => Resource Configs Specs

```
apiVersion: v1
kind:
metadata:
spec:
```

Spec Schema: https://kubernetes.io/docs/user-guide/pods/multi-container/

To list supported version of apis

```
kubectl api-versions
```

Writing Pod Spec

Lets now create the Pod config by adding the kind and specs to schme given in the file vote-pod.yaml as follows.

Filename: k8s-code/pods/vote-pod.yaml

```
apiVersion:
kind: Pod
metadata:
spec:
```

Lets edit this and add the pod specs

Filename: k8s-code/pods/vote-pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: vote
  labels:
    app: python
    role: vote
    version: v1
spec:
  containers:
    - name: app
      image: schoolofdevops/vote:v1
```

Use this link to refer to pod spec

Launching and operating a Pod

To launch a monitoring screen to see whats being launched, use the following command in a new terminal window where kubectl is configured.

```
watch -n 1  kubectl get pods,deploy,rs,svc
```

kubectl Syntax:

```
kubectl
```

```
kubectl apply --help
```

```
kubectl apply -f FILE
```

To Launch pod using configs above,

```
kubectl apply -f vote-pod.yaml
```

To view pods

```
kubectl get pods
```

```
kubectl get po -o wide
```

```
kubectl get pods vote
```

To get detailed info

```
kubectl describe pods vote
```

[Output:]

```
Name:           vote
Namespace:      default
Node:           kube-3/192.168.0.80
Start Time:     Tue, 07 Feb 2017 16:16:40 +0000
```

```
Labels:         app=voting
Status:         Running
IP:             10.40.0.2
Controllers:    <none>
Containers:
  vote:
    Container ID:       docker://48304b35b9457d627b341e424228a725d05c2ed97cc9970bbff32a1b365d9a5d
    Image:              schoolofdevops/vote:latest
    Image ID:           docker-pullable://schoolofdevops/vote@sha256:3d89bfc1993d4630a58b831a6d44ef73d2be76a7862153e02e7a7c0cf2936731
    Port:               80/TCP
    State:              Running
      Started:          Tue, 07 Feb 2017 16:16:52 +0000
    Ready:              True
    Restart Count:      0
    Volume Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-2n6j1 (ro)
    Environment Variables:      <none>
Conditions:
  Type          Status
  Initialized   True
  Ready         True
  PodScheduled  True
Volumes:
  default-token-2n6j1:
    Type:       Secret (a volume populated by a Secret)
    SecretName: default-token-2n6j1
QoS Class:      BestEffort
Tolerations:    <none>
Events:
  FirstSeen     LastSeen        Count   From                    SubObjectPath           Type            Reason          Message
  ---------     --------        -----   ----                    -------------           --------        ------          -------
  21s           21s             1       {default-scheduler }                            Normal          Scheduled       Successfully
assigned vote to kube-3
  20s           20s             1       {kubelet kube-3}        spec.containers{vote}   Normal          Pulling         pulling image
"schoolofdevops/vote:latest"
  10s           10s             1       {kubelet kube-3}        spec.containers{vote}   Normal          Pulled          Successfully
pulled image "schoolofdevops/vote:latest"
  9s            9s              1       {kubelet kube-3}        spec.containers{vote}   Normal          Created         Created container
with docker id 48304b35b945; Security:[seccomp=unconfined]
  9s            9s              1       {kubelet kube-3}        spec.containers{vote}   Normal          Started         Started container
with docker id 48304b35b945
```

Commands to operate the pod

```
kubectl logs vote
kubectl exec -it vote  sh
```

Inside the container in a pod

```
ifconfig
```

```
cat /etc/issue
```

```
hostname
```

```
cat /proc/cpuinfo
```

```
ps aux
```

Attach a Volume to the Pod

Lets create a pod for database and attach a volume to it. To achieve this we will need to

- create a **volumes** definition
- attach volume to container using **VolumeMounts** property

Local host volumes are of two types:
* emptyDir
* hostPath

We will pick hostPath. Refer to this doc to read more about hostPath.

File: db-pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: db
  labels:
    app: postgres
    role: database
    tier: back
spec:
  containers:
    - name: db
      image: postgres:9.4
      ports:
        - containerPort: 5432
      volumeMounts:
      - name: db-data
        mountPath: /var/lib/postgresql/data
  volumes:
  - name: db-data
    hostPath:
      path: /var/lib/pgdata
      type: DirectoryOrCreate
```

To create this pod,

```
kubectl apply -f db-pod.yaml
kubectl describe pod db
```

```
kubectl get events
```

Exercise : Examine **/var/lib/pgdata** on the systems to check if the directory is been created and if the data is present.

Creating Multi Container Pods

file: multi_container_pod.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: web
  labels:
    tier: front
    app: nginx
    role: ui
spec:
  containers:
    - name: nginx
      image: nginx:stable-alpine
      ports:
        - containerPort: 80
          protocol: TCP
      volumeMounts:
        - name: data
          mountPath: /var/www/html-sample-app

    - name: sync
      image: schoolofdevops/sync:v2
      volumeMounts:
        - name: data
          mountPath: /var/www/app

  volumes:
    - name: data
      emptyDir: {}
```

To create this pod

```
kubectl apply -f multi_container_pod.yml
```

Check Status

```
root@kube-01:~# kubectl get pods
NAME      READY     STATUS              RESTARTS   AGE
nginx     0/2       ContainerCreating   0          7s
vote      1/1       Running             0          3m
```

Checking logs, logging in

```
kubectl logs  web  -c sync
```

```
kubectl logs  web  -c nginx
```

```
kubectl exec -it web  sh  -c nginx
```

```
kubectl exec -it web  sh  -c sync
```

Observe whats common and whats isolated in two containers running inside the same pod using the following commands,

shared

```
hostname
```

```
ifconfig
```

isolated

```
cat /etc/issue
```

```
ps aux
```

```
df -h
```

Exercise

Create a pod definition for redis and deploy.

Reading List

- PodSpec: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.9/#pod-v1-core
- Managing Volumes with Kubernetes: https://kubernetes.io/docs/concepts/storage/volumes/
- Node Selectors, Affinity: https://kubernetes.io/docs/concepts/configuration/assign-pod-node/