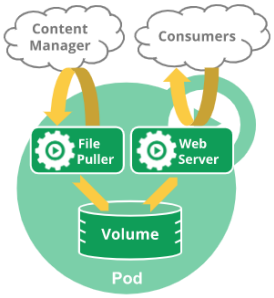
20Oct2022
Day**12**
Kubernetes Network
Kubernetes Service

Pods



Pods are the smallest deployable units of computing that you can create and manage in Kubernetes.
Pods natively provide two kinds of shared resources for their constituent containers: **networking** and **storage**.

after create K8s cluster, by default one **network plug-in** be available and has been installed, name is CNI
when create POD, ip will set on POD not container, not deployment, not ReplicaSet,…

what is Container Network Interface-CNI

https://github.com/containernetworking/cni
CNI-Container Network Interface, a **CNCF** project, consists of a specification and libraries for writing plugins to configure network interfaces in Linux containers, along with a number of supported plugins.

```
# ip a s
4: cni0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    inet 10.85.0.1/16 brd 10.85.255.255 scope global cni0
# cat /etc/cni/net.d/100-crio-bridge.conf
```

CNCF-CNI advantage
-default network plugin
CNCF-CNI disadvantage
-application accessible from hosted nide, not from other nodes in cluster

```
# kubectl run pod1 --image nginx -o yaml --dry-run=client >pod1.yaml
# vim pod1.yaml
# cat pod1.yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: pod1
  name: pod1
spec:
  containers:
  - image: nginx
    name: nginxcnt
# kubectl apply -f pod1.yaml
# kubectl get pods -o wide
pod1   1/1    Running   0        16m   10.85.0.2   node2.example.com
root@master:~# curl 10.85.0.2
curl: (7) Failed to connect to 10.85.0.2 port 80: No route to host
root@master:~# ssh root@node2.example.com
root@node2:~# curl 10.85.0.2
<h1>Welcome to nginx!</h1>
root@node2:~# logout
root@master:~#
```

Installing Addons

https://kubernetes.io/docs/concepts/cluster-administration/addons/#networking-and-network-policy
https://github.com/containernetworking/cni

Calico

https://github.com/projectcalico/calico
https://projectcalico.docs.tigera.io/about/about-calico
its opensource CNI plugin for Kubernetes.
Calico is a widely adopted, battle-tested open-source networking and network security solution for Kubernetes, virtual machines, and bare-metal workloads.
Calico provides two major services for Cloud Native applications:
1-Network connectivity between workloads.
2-Network security policy enforcement between workloads.

Implement Calico on K8s Cluster

https://docs.projectcalico.org/manifests/calico.yaml

```
# wget https://docs.projectcalico.org/manifests/calico.yaml
# ls
calico.yaml
# kubectl apply -f calico.yaml
# watch kubectl get pods -A                    -A ->whole namespace
# watch kubectl get pods –namespace kube-system
# kubectl get pods -o wide
# kubectl delete pod pod1 --force --grace-period=0
# kubectl create -f pod1.yaml
# kubectl get pods -o wide
pod1   1/1    Running   0        7m45s   172.16.221.1   node2.example.com
```

NOW, from every node in cluster container is accessible

```
root@node3:~# curl 172.16.221.1
<h1>Welcome to nginx!</h1>
```

issue will happen when create deployment
kubectl create deployment dpl1 --image redis --replicas=3 -o yaml --dry-run=client >dpl1.yaml
kubectl create -f dpl1.yaml
kubectl get deployments.apps -o wide
kubectl get pods -o wide
```
dpl1-7b5786d846-6cqr6  1/1   Running  0      24s  172.16.221.2     node2.example.com
dpl1-7b5786d846-bztd8  1/1   Running  0      24s  172.16.11.65     node1.example.com
dpl1-7b5786d846-dxsp8  1/1   Running  0      24s  172.16.178.193   node3.example.com
```

Points
-after create deployment, it depends on pods inside it, ip will assign to pod
-next issue is when a pod inside deployment will delete again new one will generate immediately but with new ip from CNI. IP, its temporary.
above issues will solve by SERVICE in K8s cluster

Service

https://kubernetes.io/docs/concepts/services-networking/service/
An abstract way to expose an application running on a set of Pods as a network service.
kubectl create service --help
Create a service using a specified subcommand.
Available Commands:

clusterip	Create a ClusterIP service	->exposes the service just inside the cluster
externalname	Create an ExternalName service	
loadbalancer	Create a LoadBalancer service	
nodeport	Create a NodePort service	->application accessible from outside through static, permanent ip

-create NGINX deployment first with 3replicas
kubectl get deployment
kubectl get pods
kubectl expose deployment dpl1 --name dpl1int --port 80 --protocol TCP --type ClusterIP -o yaml --dry-run=client >dpl1int.yaml
cat dpl1int.yaml
```
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels:
    app: dpl1
  name: dpl1int
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: dpl1
  type: ClusterIP
status:
  loadBalancer: {}
```
kubectl expose deployment dpl1 --name dpl1int --port 80 --protocol TCP --type ClusterIP
kubectl get svc
```
dpl1int     ClusterIP   10.110.147.73   <none>      80/TCP    32s
```
kubectl describe svc dpl1int
kubectl expose deployment dpl1 --name dpl1ext --port 80 --protocol TCP --type NodePort -o yaml --dry-run=client >dpl1ext.yaml
cat dpl1ext.yaml
```
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels:
    app: dpl1
  name: dpl1ext
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: dpl1
  type: NodePort
status:
  loadBalancer: {}
```
kubectl apply -f dpl1ext.yaml
kubectl get svc
```
dpl1ext     NodePort    10.97.239.101   <none>      80:30991/TCP   4s
dpl1int     ClusterIP   10.110.147.73   <none>      80/TCP         6m8s
```
NOTE: NodePort type its between **30000** to **32767**
Verify
open web browser and type
http://192.168.0.240:30991/
Welcome to nginx!