1. Understand the networking configuration on the cluster nodes

Reference is here.

There are 4 distinct networking problems to solve:

- Highly-coupled container-to-container communications
- Pod-to-Pod communications
- Pod-to-Service communications
- External-to-internal communications

Explore Kubernetes Environment

- What is the port the kube-scheduler is listening on in the master node?
- Notice that ETCD is listening on two ports. Which of these have more client connections established?
- Inspect the kubelet service and identify the network plugin configured for Kubernetes.
- What is the path configured with all binaries of CNI supported plugins?
- What is the CNI plugin configured to be used on this kubernetes cluster?

show

```
netstat -nplt
netstat -anp | grep etcd
netstat -anp | grep etcd | grep 2379 | wc -l
netstat -anp | grep etcd | grep 2380 | wc -l
ps -aux | grep kubelet
systemctl status kubelet.service
ls /etc/cni/net.d/
```

Choosing different CNI networking solutions

- Deploy weave-net networking solution to the cluster
- Deploy calico networking solution to the cluster
- Deploy flannel networking solution to the cluster

show

- Deploy weave-net. Documentation is here.

  ```
  kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
  ```

- Deploy calico. Documentation is here.

  ```
  kubectl apply -f https://docs.projectcalico.org/v3.8/manifests/calico.yamlhttps://docs.projectcalico.org/v3.8/manifests/calico.yaml
  ```

- Deploy flannel. Documentation is here.

  ```
  kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.ymlhttps://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
  ```

2. Understand Pod networking concepts

show

3. Understand Service Networking

Service networking

- What network range are the nodes in the cluster part of? Run the command ip addr and look at the IP address assigned to the ens3 interfaces. Derive network range from that.
- What is the range of IP addresses configured for PODs on this cluster?
  - For weave-net, Check the weave pods logs using command kubectl logs weave -n kube-system and look for ipalloc-range
  - For calico, check the calico.yaml file and looking for CALICO_IPV4POOL_CIDR
- What is the IP Range configured for the services within the cluster?
  - Inspect the setting on kube-api server by running on command ps -aux | grep kube-api
- What type of proxy is the kube-proxy configured to use?
  - Check the logs of the kube-proxy pods. Command: kubectl logs kube-proxy-ft6n7 -n kube-system

show

4. Deploy and configure network load balancer

show

5. Know how to use Ingress rules

Note that the innotation in setting the Ingress Controller in very **IMPORTANT**: Reference is here. We can select between gce/nginx/traefik.

For instance,

```
metadata:
  name: foo
  annotations:
    kubernetes.io/ingress.class: "gce"
```

will target the GCE controller, forcing the nginx controller to ignore it, while an annotation like

```
metadata:
  name: foo
  annotations:
    kubernetes.io/ingress.class: "nginx"
```

will target the nginx controller, forcing the GCE controller to ignore it.

Traefik ingress controller

Installation guide is here.

- Create a ClusterRole 'traefik-ingress-controller' with two rules:
  - apiGroup: "", resources: services, endpoints, secrets, verbs: get, list, watch
  - apiGroup: extensions, resources: ingresses, verbs: get, list, watch
- Create a ClusterRoleBiding 'traefik-ingress-controller' that binds the ClusterRole 'traefik-ingress-controller' and the service account 'traefik-ingress-controller'.
- Deploy Traefik using the DaemonSet:
  - Download and save the ds file using this link.
  - Edit the file, delete securityContext lines and add 'hostNetwork: true' line that lines up with containers: line.
  - Create the ingress controller with kubectl create -f .
- Create a deployment name 'secondapp', with image 'nginx', then expose the new server as NodePort, port 80.
- Create an Ingress 'ingress-test', with the rules:
  - Host: www.secondapp.com
  - Backend: secondapp service, port 80
- Check with curl -H "Host: www.secondapp.com" http://20.0.0.11/
- Create a deployment name 'thirdpage', with image 'nginx', then expose the new server as NodePort, port 80.
- Execute the thirdpage pod to modify the title of the webpage to 'Third Page' locate at /usr/share/nginx/html/index.html
- Modify the ingress-test, adding second rules:
  - Host: www.thirdpage.com

- Check with curl -H "Host: www.thirdpage.com" http://20.0.0.11/
  - Backend: thirdpage service, port 80

show

- Create a ClusterRole 'traefik-ingress-controller'

```
kubectl create clusterrole traefik-ingress-controller --resource=ingresses --verb=get,list,watch \
          --dry-run -o yaml > traefik-cluster-role.yaml
vim traefik-cluster-role.yaml
kubectl create -f traefik-cluster-role.yaml

apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
    name: traefik-ingress-controller
rules:
- apiGroups:
    - extensions
    resources:
    - ingresses
    verbs:
    - get
    - list
    - watch
- apiGroups:
    - ""
    resources:
    - services
    - endpoints
    - secrets
    verbs:
    - get
    - list
    - watch
```

- Create a ClusterRoleBinding 'traefik-ingress-controller'

```
kubectl create clusterrolebinding traefik-ingress-controller --clusterrole=traefik-ingress-controller \
          --serviceaccount=kube-system:traefik-ingress-controller
```

- Deploy Traefik using the DaemonSet:

```
wget https://raw.githubusercontent.com/containous/traefik/v1.7/examples/k8s/traefik-ds.yamlhttps://raw.githubusercontent.com/containous/traefik/v1.7/examples/k8s/traefik-ds.yaml -O traefik-ds.yaml
vim traefik-ds.yaml
kubectl create -f traefik-ds.yaml
```

- Create deployment secondapp

```
kubectl create deployment secondapp --image=nginx
kubectl expose deployment secondapp --type=NodePort --port=80
```

- Create an Ingress 'ingress-test': reference is here.

```
vim ingress-test.yaml

apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
    name: ingress-test
spec:
    rules:
    - host: www.secondapp.comwww.secondapp.com
      http:
        paths:
        - backend:
            serviceName: secondapp
            servicePort: 80

kubectl create -f ingress-test.yaml
curl -H "Host: www.secondapp.com"www.secondapp.com" http://20.0.0.11http://20.0.0.11
```

- Create deployment thirdpage, edit the deployment to add the custom dnspolicy.

```
kubectl create deployment thirdpage --image=nginx
kubectl expose deployment thirdpage --type=NodePort --port=80
kubectl edit deployement thirdpage

      spec:
        containers:
        - image: nginx
          imagePullPolicy: Always
          name: nginx
        dnsConfig:
          nameservers:
          - 8.8.8.8
        dnsPolicy: None
```

- Execute the thirdpage pod to modify nginx webpage title.

```
kubectl exec thirdpage-658458994f-77mhs -it -- /bin/bash
apt-get update
apt-get install vim -y
vim /usr/share/nginx/html/index.html
```

- Modify the ingress-test, adding second rules:

```
kubectl edit ingress ingress-test

spec:
    rules:
    - host: www.secondapp.comwww.secondapp.com
      http:
        paths:
        - backend:
            serviceName: secondapp
            servicePort: 80
    - host: www.thirdpage.comwww.thirdpage.com
      http:
        paths:
        - backend:
            serviceName: thirdpage
            servicePort: 80
```

- Check with curl

```
curl -H "Host: www.thirdpage.com"www.thirdpage.com" http://20.0.0.11http://20.0.0.11
curl -H "Host: www.secondapp.com"www.secondapp.com" http://20.0.0.11http://20.0.0.11
```

Nginx ingress controller

- Create 2 new namespaces 'app-space', 'critical-space'
- Create 4 deployments:
    - default-backend (image: kodekloud/ecommerce:404),
    - webapp-food (image: kodekloud/ecommerce:food),
    - webapp-video (image: kodekloud/ecommerce:video),
    - webapp-wear (image: kodekloud/ecommerce:apparels).
- Expose 4 deployments of type ClusterIP:
    - default-http-backend: port 80
    - food-service: port 8080
    - video-service: port 8080
    - wear-service: port 8080
- Create ingress name 'ingress-wear-watch' in 'app-space', to have the service at different paths: /wear, /stream, /eat.
- Create deployment: webapp-pay (image: kodekloud/ecommerce:pay) in 'critical-space', expose as 'pay-service' at port 8282.
- Create ingress name 'ingress-pay' in 'critical-space', to have the service at path: /pay.

show

- Create two namespaces:

```
kubectl create ns app-space
kubectl create ns critical-space
```

- Create 4 deployments

```
kubectl create deployment default-backend --image=kodekloud/ecommerce:404 --namespace=app-space
kubectl create deployment webapp-food --image=kodekloud/ecommerce:food --namespace=app-space
kubectl create deployment webapp-video --image=kodekloud/ecommerce:video --namespace=app-space
kubectl create deployment webapp-wear --image=kodekloud/ecommerce:apparels --namespace=app-space
```

- Expose 4 deployments:

```
kubectl -n app-space expose deployment default-backend --type=ClusterIP --port=80 --name=default-http-backend
kubectl -n app-space expose deployment webapp-food --type=ClusterIP --port=8080 --name=food-service
kubectl -n app-space expose deployment webapp-video --type=ClusterIP --port=8080 --name=video-service
kubectl -n app-space expose deployment webapp-wear --type=ClusterIP --port=8080 --name=wear-service
```

- Create ingress name 'ingress-wear-watch': **ANNOTATIONS is very IMPORTANT**. Reference is here.

```
vim ingress-wear-watch.yaml

apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
    name: ingress-wear-watch
    namespace: app-space
    annotations:
      kubernetes.io/ingress.class: traefik
      traefik.frontend.rule.type: PathPrefixStrip
spec:
    rules:
    - http:
        paths:
        - path: /wear
          backend:
            serviceName: wear-service
            servicePort: 8080
        - path: /stream
          backend:
            serviceName: video-service
            servicePort: 8080
        - path: /eat
          backend:
            serviceName: food-service
            servicePort: 8080

kubectl create -f ingress-wear-watch.yaml
```

6. Know how to configure and use the cluster DNS

Explore CoreDns

show

7. Understand CNI

show