Note: LiCO should be deployed before running the steps of this document. In this document, unless stated, all the operations are executed in the microk8s host.

The used yaml file are under https://hpc.lenovo.com/lico/downloads/6.0/examples/microk8s/.

1. Rename microk8s.kubectl to kubectl for convenient usage.

[root@chaofeng-k8s ddd]# snap alias microk8s.kubectl kubectl

2. Add hostname into /etc/hosts.

[root@chaofeng-k8s ddd]# cat /etc/hosts

127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4

::1 localhost localhost.localdomain localhost6 localhost6.localdomain6

10.241.34.15 chaofeng-k8s

3. Enable microk8s add-ons

[root@chaofeng-k8s ddd]# microk8s enable dns dashboard metrics-server rbac

4. Install ingress controller, (microk8s has ingress add-on, but it has bug, so we should not enable the mircork8s ingress, we need install it)

[root@chaofeng-k8s ddd]# kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/nginx-0.30.0/deploy/static/mandatory.yaml

[root@chaofeng-k8s ddd]# kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/nginx-0.30.0/deploy/static/provider/baremetal/service-nodeport.yaml

5. Enable clusterrole system:anonymous

[root@chaofeng-k8s ddd]# kubectl apply -f system-anonymous.yaml

6. Create LiCO cluster role

[root@chaofeng-k8s ddd]# kubectl apply -f clusterrole.yaml

7. Configure prometheus for gpu monitoring

[root@chaofeng-k8s ddd]# kubectl label node `hostname` hardware-type=NVIDIAGPU

[root@chaofeng-k8s ddd]# kubectl create namespace monitoring

#In the file Prometheus-deployment.yaml, change the <gpu node address> to the ip address of #`hostname`

prometheus-deployment.yaml | pod-gpu-metrics-exporter-daemonset.yaml

```
apiVersion: v1
data:
  prometheus.yml: |-
    global:
      scrape_interval:     15s
      evaluation_interval: 15s
    scrape_configs:
    - job_name: pod-gpu
      metrics_path: /gpu/metrics
      scheme: http
      static_configs:
      - targets:
        - <gpu node address>:9400
kind: ConfigMap
metadata:
  name: prometheus-config
```

[root@chaofeng-k8s ddd]# kubectl apply -f prometheus-deployment.yaml

[root@chaofeng-k8s ddd]# kubectl apply -f pod-gpu-metrics-exporter-daemonset.yaml

8. Disable streaming connection timeout, by default the value is 4 hours.

Add *--streaming-connection-idle-timeout=0* to file */var/snap/microk8s/current/args/kubelet,* then restart the kubelet service.

[root@chaofeng-k8s ddd]# systemctl restart snap.microk8s.daemon-kubelet.service

9. Setup docker registry

you can use microk8s build-in docker registry, https://microk8s.io/docs/registry-built-in

[root@chaofeng-k8s ddd]# microk8s enable registry

or you can also use existing private docker registry, you should add docker registry to docker daemon and microk8s. https://microk8s.io/docs/registry-private

[root@chaofeng-k8s ddd]# cat /etc/docker/daemon.json

{

"insecure-registries": ["10.240.208.138:5000","10.240.212.106"]

}

[root@chaofeng-k8s ddd]# service docker restart

[root@chaofeng-k8s args]# vi /var/snap/microk8s/current/args/containerd-template.toml

[plugins.cri.registry]

[plugins.cri.registry.mirrors]

[plugins.cri.registry.mirrors."docker.io"]

endpoint = ["https://registry-1.docker.io"]

[plugins.cri.registry.mirrors."localhost:32000"]

endpoint = ["http://localhost:32000"]

[plugins.cri.registry.mirrors."10.240.212.106"]

endpoint = ["http://10.240.212.106"]

[root@chaofeng-k8s ddd]# microk8s stop

[root@chaofeng-k8s ddd]# microk8s start

10. Build docker image and push them to docker registry

Follow LiCO installation guide to build build-in images.

11. Configure LiCO

In LiCO host(VM), change the configure file kube_server.csv under /etc/lico. then run command: *lico sync_kube_server* to enable the configure changes in LiCO host.

```
[root@lico lico]# vi kube_server.csv
"# This file is used to define basic information about a Kubernetes cluster."
"# It is recommended that you edit this file by using Excel or other table editing software."
"# Notes:"
"# Lines beginning with the hash sign (#) are comment lines. Delete them if you do not need any comments."
"# The following is an example. Replace the parameter values with actual data."
"# Columns:"
"# name - Name of a Kubernetes cluster, which is unique and used to identify this cluster in the LiCO system."
"# display_name - Name of a Kubernetes cluster displayed in the LiCO system. A meaningful display name is recom
mended."
"# kube_cluster_addr - API server address of a Kubernetes cluster."
"# ingress_ctrl_addr - Ingress-controller service address of a Kubernetes cluster."
"# gpu_resource_name - GPU resource name of a Kubernetes cluster, which can be left blank. If it is left blank,
 the default value nvidia.com/gpu is used."
name,display_name,kube_cluster_addr,ingress_ctrl_addr,gpu_resource_name,prometheus_server,metrics_server
mykube,My k8s,https://10.240.208.138:6443,http://10.240.208.136:33453,nvidia.com/gpu,http://10.240.208.138:9090
,https://10.240.208.140:43731
microkube,mirco k8s,https://10.241.34.15:16443,http://10.241.34.15:31398,nvidia.com/gpu,http://10.241.34.15:325
45,https://10.241.34.15:30082
```

The below is how to get the information needed by kube_server.csv.

kube_cluster_addr:

```
[root@chaofeng-k8s args]# cat /var/snap/microk8s/current/args/kube-apiserver
--cert-dir=${SNAP_DATA}/certs
--service-cluster-ip-range=10.152.183.0/24
--authorization-mode=RBAC,Node
--basic-auth-file=${SNAP_DATA}/credentials/basic_auth.csv
--service-account-key-file=${SNAP_DATA}/certs/serviceaccount.key
--client-ca-file=${SNAP_DATA}/certs/ca.crt
--tls-cert-file=${SNAP_DATA}/certs/server.crt
--tls-private-key-file=${SNAP_DATA}/certs/server.key
--kubelet-client-certificate=${SNAP_DATA}/certs/server.crt
--kubelet-client-key=${SNAP_DATA}/certs/server.key
--secure-port=16443
--token-auth-file=${SNAP_DATA}/credentials/known_tokens.csv
--token-auth-file=${SNAP_DATA}/credentials/known_tokens.csv
--etcd-servers='https://127.0.0.1:12379'
--etcd-cafile=${SNAP_DATA}/certs/ca.crt
--etcd-certfile=${SNAP_DATA}/certs/server.crt
--etcd-keyfile=${SNAP_DATA}/certs/server.key
--insecure-port=0

# Enable the aggregation layer
--requestheader-client-ca-file=${SNAP_DATA}/certs/front-proxy-ca.crt
--requestheader-allowed-names=front-proxy-client
--requestheader-extra-headers-prefix=X-Remote-Extra-
--requestheader-group-headers=X-Remote-Group
--requestheader-username-headers=X-Remote-User
--proxy-client-cert-file=${SNAP_DATA}/certs/front-proxy-client.crt
--proxy-client-key-file=${SNAP_DATA}/certs/front-proxy-client.key
#~Enable the aggregation layer
[root@chaofeng-k8s args]#
```

Ingress ctl:

```
[root@chaofeng-k8s args]# kubectl get svc -n ingress-nginx
NAME            TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)                      AGE
ingress-nginx   NodePort   10.152.183.87   <none>        80:31398/TCP,443:30155/TCP   3h5m
[root@chaofeng-k8s args]#
```

Prometheus server:

```
[root@c1 test]# kubectl get service -n monitoring
NAME         TYPE       CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
prometheus   NodePort   10.152.183.184   <none>        9090:32543/TCP   7h53m
```

Metric server:

By default, the service is not exposed as hostport, so you need use the command *kubectl edit svc metrics-server -n kube-system* to export the service as hostport

```
[root@chaofeng-k8s args]# kubectl get svc -n kube-system
NAME                        TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                  AGE
dashboard-metrics-scraper   ClusterIP   10.152.183.193   <none>        8000/TCP                 5h51m
heapster                    ClusterIP   10.152.183.242   <none>        80/TCP                   5h51m
kube-dns                    ClusterIP   10.152.183.10    <none>        53/UDP,53/TCP,9153/TCP   5h52m
kubelet                     ClusterIP   None             <none>        10250/TCP                5h4m
kubernetes-dashboard        NodePort    10.152.183.135   <none>        443:31281/TCP            5h51m
metrics-server              NodePort    10.152.183.58    <none>        443:30082/TCP            4h4m
monitoring-grafana          ClusterIP   10.152.183.252   <none>        80/TCP                   5h51m
monitoring-influxdb         ClusterIP   10.152.183.237   <none>        8083/TCP,8086/TCP        5h51m
```

12. Create namespace, pv, pvc and roles for user1, see the namespace.yaml.

user1 will use /opt/user1 directory as share folder, and create a PV user1-pv01 pointing to /opt/user1. You can change the path and change the storage sizing from 20Gi to what you need in the namespace.yaml

lico.host.ini | clusterrole.yaml | namespace.yaml

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: user1-namespace

---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: user1-pv01
  labels:
    pv: user1-pv01
spec:
  capacity:
    storage: 20Gi
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  hostPath:
    path: /opt/user1
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: user1-pvc01
  namespace: user1-namespace
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: ""
  resources:
    requests:
      storage: 20Gi
  selector:
    matchLabels:
      pv: user1-pv01
---
```

[root@chaofeng-k8s ddd]# mkdir /opt/user1

[root@chaofeng-k8s ddd]# kubectl apply -f namespace.yaml

13. LiCO administrator creates a user user1 through LiCO GUI, then login LiCO using user1, when user1 login LiCO, it needs inputing namespace, pvc, token.

The namespace is user1-namespace, the pvc is user1-pvc01, you can use the following command to get the token of user1.

```
[root@chaofeng-k8s args]# kubectl describe secret -n user1-namespace
Name:         default-token-f59ct
Namespace:    user1-namespace
Labels:       <none>
Annotations:  kubernetes.io/service-account.name: default
              kubernetes.io/service-account.uid: 232e1dfb-3b5b-49ff-9e78-0bbd9d93aed1

Type:  kubernetes.io/service-account-token

Data
====
ca.crt:     1103 bytes
namespace:  15 bytes
token:      eyJhbGciOiJSUzI1NiIsImtpZCI6InNOVDVCTjhNUWZVZTc2alNOZnh6OVZEZGZiby1RdlRTaXNzTjNoaEl5TDQifQ.eyJpc3Mi
OiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJ1c2VyMS1uYW1lc3B
hY2UiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlY3JldC5uYW1lIjoiZGVmYXVsdC10b2tlbi1mNTljdCIsImt1YmVybmV0ZXMuaW
8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50Lm5hbWUiOiJkZWZhdWx0Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2a
WNlLWFjY291bnQudWlkIjoiMjMyZTFkZmItM2I1Yi00OWZmLTllNzgtMGJiZDlkOTNhZWQxIiwic3ViIjoic3lzdGVtOnNlcnZpY2VhY2NvdW50
OnVzZXIxLW5hbWVzcGFjZTpkZWZhdWx0In0.zO7fHbFPMMBmxslwWoRGNX9u2FZcE0twZR28IC-jPDGzu53lQeaOPDCUVTgSZLhdWStrp8DAt8b
1aontK15nBKEiiI7tZtqfHS0JxaOroFwQqR4IMDrKUee0qdHm8zzx1hlfJbvnPuW4YbvQ3fnhjON-ogEb4vHh50ANUUgDSj3pT0Q1Pj-Ne5PosD
n4SlzBlFiHwwZL9w4rai8M4nCZCW1KO9iLK9_R3sAjbrlGr0ggUSMeRPCSZRt7k_9OBZesXeSW5FgTKPn-TMlx7ngDWlfNfCx-wbe16vLYM0Qmf
FTtJYFQpbjZZoaD6CbqcM-cIymT67SeEqGm8VKPcolEkQ
```

14. If you want to use kubernetes dashboard

By default, the service is not exposed as hostport, you need using command *kubectl edit svc kubernetes-dashboard -n kube-system* to export the service as hostport.

The below is about how to get exposed port of kubernetes dashboard.

```
[root@chaofeng-k8s ~]# kubectl get svc -n kube-system
NAME                        TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                  AGE
dashboard-metrics-scraper   ClusterIP   10.152.183.193   <none>        8000/TCP                 2d1h
heapster                    ClusterIP   10.152.183.242   <none>        80/TCP                   2d1h
kube-dns                    ClusterIP   10.152.183.10    <none>        53/UDP,53/TCP,9153/TCP   2d1h
kubelet                     ClusterIP   None             <none>        10250/TCP                2d
kubernetes-dashboard        NodePort    10.152.183.135   <none>        443:31281/TCP            2d1h
metrics-server              NodePort    10.152.183.58    <none>        443:30082/TCP            47h
monitoring-grafana          ClusterIP   10.152.183.252   <none>        80/TCP                   2d1h
monitoring-influxdb         ClusterIP   10.152.183.237   <none>        8083/TCP,8086/TCP        2d1h
```

Then you can login *https://<hostip>:31281* with firefox browser.

The below is how to get the token for kubernetes dashboard login.

```
[root@chaofeng-k8s ~]# kubectl describe secret kubernetes-dashboard-token-545hd -n kube-system
Name:         kubernetes-dashboard-token-545hd
Namespace:    kube-system
Labels:       <none>
Annotations:  kubernetes.io/service-account.name: kubernetes-dashboard
              kubernetes.io/service-account.uid: d31026dd-8ce4-4026-96b7-684c60e7bc38

Type:  kubernetes.io/service-account-token

Data
====
token:      eyJhbGciOiJSUzI1NiIsImtpZCI6InNOVDVCTjhNUWZVZTc2a1NOZnh6OVZEZGZiby1RdlRTaXNzTjNoaEl5TDQifQ.eyJpc3MiO
iJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJlLXN5c3RlbSIsI
mt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJrdWJlcm5ldGVzLWRhc2hib2FyZC10b2tlbi01NDVoZCIsImt1YmVyb
mV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50Lm5hbWUiOiJrdWJlcm5ldGVzLWRhc2hib2FyZCIsImt1YmVybmV0ZXMuaW8vc
2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50LnVpZCI6ImQzMTAyNmRkLThjZTQtNDAyNi05NmI3LTY4NGM2MGU3YmMzOCIsInN1YiI6InN5c
3RlbTpzZXJ2aWNlYWNjb3VudDprdWJlLXN5c3RlbTprdWJlcm5ldGVzLWRhc2hib2FyZCJ9.M-zHTpaqfE02DcEln4-hDO57LynJUOaITfPNlTfk
3rdSg_5SPjxr1hsIVyWXVr2kB4KhCJxNbeFom18-WGWxcna8wGdZ6MuXChF6PMadTFlOmi6KJBqxoLxmA6VX1MakzSJIEf69XPUhxTe369x7JT9R
p7UXayPMEwWkCS5nsEVHZrDBjxgFy0Kn9-IBAeXXskPUYJTsDDkd_ak9D0UUUa88cihTXM20Gc4PW1Usi4virgBbfsKHbY4U63PMdk7kZtGs7_I-
Kl8hBFKdT-AZbS29yvi3n5gLb-thNm4toNAsNf_hcdIcBw_W5q_4xduOpWBYnHnKw57QeFhx8EIBOg
ca.crt:     1103 bytes
namespace:  11 bytes
```

After login, you will find you have no right to access many resources, then use the below command to change the role so that you can manage all namespaces throughthe dashboard.

[root@chaofeng-k8s ddd]# kubectl create clusterrolebinding kubernetes-dashboard-1 -n kube-system --clusterrole=cluster-admin --serviceaccount=kube-system:kubernetes-dashboard