ShifuDemo-柳加兆

1、安装docker并配置docker源

```Bash
yum install -y docker-ce
yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
```

2、开启docker并设置开机自启

```Bash
systemctl enable docker
systemctl start docker
```

```
[root@localhost ~]# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

3、安装Shifu

```
[root@localhost ~]# curl -sfL https://raw.githubusercontent.com/Edgenesis/shifu/mai
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
 83  700M   83  585M    0     0  5615k      0  0:02:07  0:01:46  0:00:21  931k
```

Bash

```
curl -sfL https://raw.githubusercontent.com/Edgenesis/shifu/main/test/scripts/shifu-demo-install.sh | sudo sh -
```

安装过程中出现错误，原因是内核不支持cgroup

```
Loaded image: edgehub/shifu-controller:v0.38.0
Deleting cluster "kind" ...
Creating cluster "kind" ...
 ✓ Ensuring node image (kindest/node:v1.27.3)
 ✗ Preparing nodes
Deleted nodes: ["kind-control-plane"]
ERROR: failed to create cluster: command "docker run --name kind-control-plane --hostname kind-control-plane --label io.x-k8s.kind.role=control-plane --privileged --security-opt seccomp=unconfined --security-opt apparmor=unconfined --tmpfs /tmp --tmpfs /run --volume /var --volume /lib/modules:/lib/modules:ro -e KIND_EXPERIMENTAL_CONTAINERD_SNAPSHOTTER --detach --tty --label io.x-k8s.kind.cluster=kind --net kind --restart=on-failure:1 --init=false --cgroupns=private --volume /dev/mapper:/dev/mapper --publish=127.0.0.1:37632:6443/TCP -e KUBECONFIG=/etc/kubernetes/admin.conf kindest/node:v1.27.3" failed with error: exit status 125
Command Output: WARNING: Your kernel does not support cgroup namespaces.  Cgroup namespace setting discarded.
40512565a626529a83da031d3fec8a4215092384da8195f61221e6058a17a7bc
docker: Error response from daemon: failed to create task for container: failed to create shim task: OCI runtime create failed: runc create failed: cgroup namespaces aren't enabled in the kernel: unknown.
```

解决方案：升级内核

Bash

```
#1 更新系统，确保所有安装的包都是最新的
yum update

#2安装 elrepo 仓库，该仓库提供了最新的稳定内核
rpm --import https://www.elrepo.org/RPM-GPG-KEY-elrepo.org
yum install -y https://www.elrepo.org/elrepo-release-7.el7.elrepo.noarch.rpm

#3 安装新的内核
yum --enablerepo=elrepo-kernel install kernel-ml -y

#4 更新GRUB引导菜单
grub2-mkconfig -o /boot/grub2/grub.cfg

#5 修改默认引导顺序，使新内核成为默认引导
grub2-set-default 0

#6 重新启动系统，确认新内核成功安装并生效
reboot
```

最终安装成功

```
deployment.apps/agv created
edgedevice.shifu.edgenesis.io/edgedevice-agv created
service/agv created
configmap/agv-configmap-0.0.1 created
deployment.apps/deviceshifu-agv-deployment created
service/deviceshifu-agv created
Finished setting up Demo !
```

```
[root@localhost ~]# kubectl get pods -A
NAMESPACE           NAME                                        READY   STATUS    RESTARTS        AGE
devices             agv-84bd788d74-5qmpx                        1/1     Running   0               4m6s
deviceshifu         deviceshifu-agv-deployment-5b49c6dbf5-wdmcr  1/1     Running   0               4m4s
kube-system         coredns-5d78c9869d-6b8xj                    1/1     Running   0               4m36s
kube-system         coredns-5d78c9869d-lsqm5                    1/1     Running   0               4m36s
kube-system         etcd-kind-control-plane                     1/1     Running   0               4m50s
kube-system         kindnet-mbg47                               1/1     Running   0               4m36s
kube-system         kube-apiserver-kind-control-plane           1/1     Running   0               4m50s
kube-system         kube-controller-manager-kind-control-plane  1/1     Running   0               4m50s
kube-system         kube-proxy-kqsrt                            1/1     Running   0               4m36s
kube-system         kube-scheduler-kind-control-plane           1/1     Running   0               4m52s
local-path-storage  local-path-provisioner-6bc4bddd6b-lvdjp     1/1     Running   0               4m36s
shifu-crd-system    shifu-crd-controller-manager-67c4cb4655-5wk56  2/2  Running   1 (2m55s ago)   4m7s
```

4、运行一个酶标仪的数字孪生

准备工作：

Bash

```
cd shifudemos

sudo kubectl run --image=nginx:1.21 nginx
sudo kubectl get pods -A | grep nginx
```

```
[root@localhost shifudemos]# kubectl run --image=nginx:1.21 nginx
pod/nginx created
[root@localhost shifudemos]# kubectl get pods -A | grep nginx
default             nginx                                       1/1     Running   0               28s
```

启动酶标仪的数字孪生

Bash

```
kubectl apply -f run_dir/shifu/demo_device/edgedevice-plate-reader
```

```
[root@localhost shifudemos]# kubectl apply -f run_dir/shifu/demo_device/edgedevice-plate-reader
configmap/plate-reader-configmap-0.0.1 created
deployment.apps/deviceshifu-plate-reader-deployment created
service/deviceshifu-plate-reader created
deployment.apps/plate-reader created
edgedevice.shifu.edgenesis.io/edgedevice-plate-reader created
service/plate-reader created
```

```bash
kubectl get pods -A | grep plate
```

```
[root@localhost shifudemos]# kubectl get pods -A | grep plate
devices             plate-reader-6bff7df894-n77kn                      1/1     Running   0             36s
deviceshifu         deviceshifu-plate-reader-deployment-7bfc8bd7f5-qzfm6   1/1     Running   0             36s
```

5、与数字孪生交互

进入nginx，然后通过 http://deviceshifu-plate-reader.deviceshifu.svc.cluster.local 进行交互，得到酶标仪的测量结果

```bash
sudo kubectl exec -it nginx -- bash

curl "deviceshifu-plate-reader.deviceshifu.svc.cluster.local/get_measurement"
```

```
[root@localhost shifudemos]# kubectl exec -it nginx -- bash
root@nginx:/# curl "deviceshifu-plate-reader.deviceshifu.svc.cluster.local/get_measurement"
0.88 0.16 1.15 2.85 2.32 2.42 0.51 1.36 2.98 1.17 0.14 2.68
2.41 1.58 2.62 2.55 0.67 2.62 0.62 1.56 2.12 0.26 0.94 0.74
0.05 1.24 1.50 2.37 0.59 1.89 1.02 0.07 1.35 0.65 1.14 1.49
1.83 1.30 1.36 0.42 1.84 1.26 0.05 0.94 0.89 0.69 2.08 1.84
2.27 1.20 0.25 1.14 2.57 0.24 0.92 0.02 0.98 1.48 0.88 2.51
0.36 0.05 2.67 1.75 1.66 0.14 1.07 1.62 1.44 1.84 1.63 0.65
1.73 0.88 1.58 0.88 1.59 0.46 2.66 0.48 1.61 2.34 1.55 1.67
0.67 2.81 1.97 1.86 2.23 2.72 2.58 2.72 1.19 1.76 1.88 1.32
```

6、编写GoLang程序

Go

```go
package main

import (
    "fmt"
    "io/ioutil"
    "net/http"
    "strings"
    "time"
)

const URL = "deviceshifu-plate-reader.deviceshifu.svc.cluster.local/get_measurement"
const TimeInterval = 10

func main() {
    for {
        _, err := getData()
        if err != nil {
            fmt.Printf("Error getting measurement: %v\n", err)
            continue
        }

        time.Sleep(time.Second * time.Duration(TimeInterval))
    }
}

func getData() ([]float64, error) {
    resp, err := http.Get(URL)
    if err != nil {
        return nil, err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return nil, err
    }

    valuesStr := strings.Fields(string(body))
    fmt.Println(valuesStr)
    return nil, nil
}
```

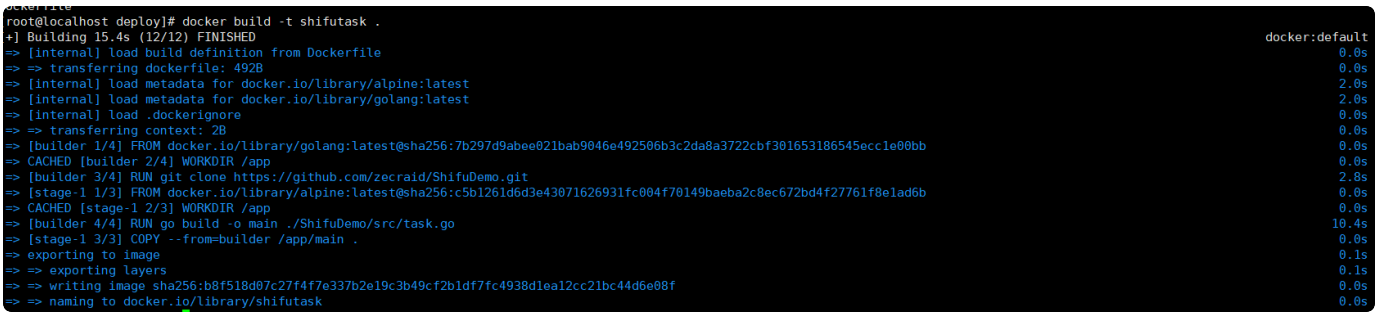
7、Go应用容器化

编写Dockerfile

```Dockerfile
# 阶段1：拉取代码并构建应用
FROM golang:latest AS builder

ENV GOPROXY https://goproxy.cn,direct

WORKDIR /app

# 拉取代码
RUN git clone https://github.com/zecraid/ShifuDemo.git

# 构建应用
RUN go build -o main ./ShifuDemo/src/task.go

# 启动应用
CMD ["./main"]
```

构建Docker镜像

```Dockerfile
docker build -t shifutest .
```



运行docker镜像

```Bash
docker run -itd --name shifutest shifutest
```

登录docker并提交代码

```bash
docker login
docker push shifutest
```

```
[root@localhost deploy]# docker push zecraid/shifutask:latest
The push refers to repository [docker.io/zecraid/shifutask]
62caa8f95abd: Pushed
ff0ebce690f0: Pushed
0abc3411d91f: Pushed
5f70bf18a086: Layer already exists
0017c0f23ae9: Layer already exists
e8dc3405fae7: Layer already exists
f7fd8efb7500: Layer already exists
909275a3eaaa: Layer already exists
f3f47b3309ca: Layer already exists
1a5fc1184c48: Layer already exists
latest: digest: sha256:2c0cdb82c8edf5eeba1c782f75b8ae3c7710aa9c304c7b2a1f1419d783f424e7 size: 2416
```

8、创建部署

编写部署yaml文件

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: aatest-shifu
spec:
  replicas: 1
  selector:
    matchLabels:
      app: aatest-shifu
  template:
    metadata:
      labels:
        app: aatest-shifu
    spec:
      containers:
        - name: shifutask
          image: zecraid/shifutask:latest
```

部署

```bash
kubectl apply -f deployment.yaml
```

部署完成后使用kubectl打印结果

```
[root@localhost deploy]# kubectl get pods
NAME                            READY   STATUS             RESTARTS        AGE
aatest-shifu-b5d65b4b5-p56ts    1/1     Running            0               25m
nginx                           1/1     Running            1 (3h10m ago)   10h
shifu-test-7dcfcf567d-tlhvd     0/1     ImagePullBackOff   0               3h15m
shifutask-6c45fd7954-vnx66      0/1     ImagePullBackOff   0               3h25m
test-shifu-5cfc7dd648-n6v7f     0/1     ImagePullBackOff   0               3h
test-shifu-75f77664b5-wr8vh     0/1     ImagePullBackOff   0               3h24m
[root@localhost deploy]# kubectl logs aatest-shifu-b5d65b4b5-p56ts
Test Shifu
Test Shifu
Test Shifu
Test Shifu
```

9、备注

整个大概的过程思路我走了一遍，但是后面不知道为啥，镜像中git拉取不到新的代码下来，就导致部署在k8s中的代码一直是之前的测试代码（也就是打印`Test Shifu`），GoLang程序是根据第五步的打印结果跑了一遍的模拟程序，未被测试。