ONEPIECE OF K8S

前期准备

在开始之前，建议请先装好虚拟机，或者租一个Linux云服务器，本次课程重点任务是实践，一共给大家准备了18个实战的TASK。

DOCKER RECAP

首先我们来复习一下Docker，也让没有安装Docker的同学来安装一下环境。

1. 安装依赖：

```
sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg-agent \
    software-properties-common
```

2. 获取GPG KEY：

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-
```

3. 添加私有APT源：

```
sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable"
```

4. 安装Docker

```
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

5. 修改配置

```
vim /etc/docker/daemon.json
```

```
{
  "registry-mirrors": [
    "https://dockerhub.azk8s.cn",
    "https://reg-mirror.qiniu.com"
  ],
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "insecure-registries": ["0.0.0.0/0"],
  "storage-driver": "overlay2"
}
```

TASK 1

启动一个内有Web版2048游戏的容器：

```
docker run -d -P daocloud.io/daocloud/dao-2048
```

TASK 2

启动一个自己的镜像Registry服务：

```
docker service create --name registry --publish 5000:5000 registry:2
```

TASK 3

拉取一个Nginx镜像，并推送到自己的Registry：

```
docker pull nginx
docker tag nginx 127.0.0.1:5000/nginx
docker push 127.0.0.1:5000/nginx
```

TASK 4

这次我们来构建一个基于OpenAPI的Web APP：

1. 安装依赖：

```
pip3 install connexion flask_cors swagger-ui-bundle -i
https://mirrors.aliyun.com/pypi/simple/
```

2. 编写API配置：

```
openapi: "3.0.0"
```

```yaml
info:
  title: Hello Pet
  version: "1.0"
servers:
  - url: http://localhost:9090/v1.0

paths:
  /pets/{pet_id}:
    get:
      description: Returns pets based on ID
      summary: Find pets by ID
      operationId: app.get_pets_by_id
      responses:
        '200':
          description: pet response
          content:
            'application/json' :
              schema:
                type: object
                $ref: '#/components/schemas/Pet'
    parameters:
    - name: pet_id
      in: path
      description: ID of pet to use
      required: true
      schema:
        type: integer
components:
  schemas:
    Pet:
      type: object
      required:
      - petType
      properties:
        petType:
          type: string
      discriminator:
        propertyName: petType
        mapping:
          dog: Dog
          cat: Cat
          lizard: Lizard
    Cat:
      allOf:
      - $ref: '#/components/schemas/Pet'
      - type: object
```

3. 用Python写后端API:

```python
import connexion
from flask_cors import CORS
```

```
def get_pets_by_id(pet_id: int) -> list:
    return [{
                'name': 'lucky',
                'petType': 'dog',
                'bark': 'woof!'
            },
            {
                'name': 'kitty',
                'petType': 'cat',
                'meow': 'meow!'
            },
            {
                'name': 'jack',
                'petType': 'lizard',
                'loveRocks': True
            }][pet_id%3]


if __name__ == '__main__':
    app = connexion.FlaskApp(__name__, port=9090,
specification_dir='./')
    app.add_api('spec.yaml', arguments={'title': 'Simple Pet'})
    CORS(app.app)
    app.run()
```

4. 运行服务：

```
python app.py
```

TASK 5

基于上一个TASK，进一步构建Pet应用的镜像：

1. 写一个Dockerfile：

```
FROM python:alpine-3.8
copy ./ /opt/app
RUN pip3 install connexion swagger-ui-bundle flask_cors -i
https://mirrors.aliyun.com/pypi/simple/
ENV PYTHONPATH /opt/app
WORKDIR /opt/app
CMD ["python",  "app"]
```

2. 构建镜像：

```
docker build -t petapp -f ./Dockerfile .
```

3. 启动容器：

```
docker run petapp
```

TASK 6

继续，我们来发布Pet应用，来尝试把它发布到我们刚刚启动的私有Registry。

TASK 7

体验Potainer管理容器：

```
docker run  -d -p 9000:9000 \
--name portainer --restart=always \
-v /var/run/docker.sock:/var/run/docker.sock \
portainer/portainer
```

K8S RECAP

接下来我们来复习一下Kubernetes，首先是几个基本的组件：

- etcd：etcd 用于 Kubernetes 的后端存储。所有集群数据都存储在此处，始终为您的 Kubernetes 集群的 etcd 数据提供备份计划。
- API Server：kube-apiserver对外暴露了Kubernetes API。它是的 Kubernetes 前端控制层。它被设计为水平扩展，即通过部署更多实例来缩放。
- controller manager：kube-controller-manager运行控制器，它们是处理集群中常规任务的后台线程。逻辑上，每个控制器是一个单独的进程，但为了降低复杂性，它们都被编译成独立的可执行文件，并在单个进程中运行。
- scheduler：kube-scheduler监视没有分配节点的新创建的 Pod，选择一个节点供他们运行。

TASK 8

我们现在来用kubeadm 安装 k8s。

准备工作

- 三台节点 (systemd 管理的)
- 比如debian, ubuntu, fedora, centos等
- 可以是云端节点或者虚拟机
- 分配好ip
- 配置好ssh 允许 root 登录
- 开启ipforward
- 关闭swap分区

硬件要求

- master节点 内存2核3G(最小2G)
- node节点 内存2核2G

配置 /etc/hosts

```
172.19.0.21 debian-21
172.19.0.22 debian-22
172.19.0.23 debian-23
```

安装/开启 ssh

```
apt install openssh-server
vim /etc/ssh/sshd_config
```

```
PermitRootLogin yes
```

开启 ipv4 的 forward机制

```
vim /etc/sysctl.conf
```

```
net.ipv4.ip_forward=1
```

```
sysctl --system
```

关闭swap

注释掉swap分区：

```
vim /etc/fstab
```

```
swapoff -a
```

安装docker运行时（略）

具体请看之前Docker的部分。

安装 MASTER NODE/ CONTROL PLANE

使用kubeadm安装 control plane：

```
apt-get update && apt-get install -y apt-transport-https
curl https://mirrors.aliyun.com/kubernetes/apt/doc/apt-key.gpg | apt-
key add -

cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
deb https://mirrors.aliyun.com/kubernetes/apt/ kubernetes-xenial main
EOF
apt-get update
apt-get install -y kubelet kubeadm kubectl
```

使用如下脚本:

```
images=(
    kube-apiserver:v1.17.0
    kube-controller-manager:v1.17.0
    kube-scheduler:v1.17.0
    kube-proxy:v1.17.0
    pause:3.1
    etcd:3.4.3-0
    coredns:1.6.5
)

for imageName in ${images[@]} ; do
    docker pull registry.cn-
hangzhou.aliyuncs.com/google_containers/$imageName
    docker tag registry.cn-
hangzhou.aliyuncs.com/google_containers/$imageName
k8s.gcr.io/$imageName
    docker rmi registry.cn-
hangzhou.aliyuncs.com/google_containers/$imageName
done
```

```
kubeadm init --apiserver-advertise-address 172.19.0.31 --pod-network-
cidr 10.30.0.0/16 --service-cidr 10.31.0.0/16
```

这个时候,似乎还是比较顺畅的,部分输出如下:

```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular
user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed
at:
```

```
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following
on each as root:

kubeadm join 172.19.0.31:6443 --token ay1y19.hyemvesfsl66wiic \
    --discovery-token-ca-cert-hash
sha256:b98a644c2f163ec19996599856b2d9b537ec78a0e61d920251239ec3131e5430
```

master节点运行：

```
  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

node节点运行(在node节点搭建完成后进行)：

```
kubeadm join 172.19.0.31:6443 --token ay1y19.hyemvesfsl66wiic \
    --discovery-token-ca-cert-hash
sha256:b98a644c2f163ec19996599856b2d9b537ec78a0e61d920251239ec3131e5430
```

```
kubectl apply -f flannel.yaml
```

flannel.yaml：

```
---
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: psp.flannel.unprivileged
  annotations:
    seccomp.security.alpha.kubernetes.io/allowedProfileNames:
docker/default
    seccomp.security.alpha.kubernetes.io/defaultProfileName:
docker/default
    apparmor.security.beta.kubernetes.io/allowedProfileNames:
runtime/default
    apparmor.security.beta.kubernetes.io/defaultProfileName:
runtime/default
spec:
  privileged: false
  volumes:
    - configMap
    - secret
    - emptyDir
    - hostPath
  allowedHostPaths:
    - pathPrefix: "/etc/cni/net.d"
    - pathPrefix: "/etc/kube-flannel"
```

```
    - pathPrefix: "/run/flannel"
  readOnlyRootFilesystem: false
  # Users and groups
  runAsUser:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  fsGroup:
    rule: RunAsAny
  # Privilege Escalation
  allowPrivilegeEscalation: false
  defaultAllowPrivilegeEscalation: false
  # Capabilities
  allowedCapabilities: ['NET_ADMIN']
  defaultAddCapabilities: []
  requiredDropCapabilities: []
  # Host namespaces
  hostPID: false
  hostIPC: false
  hostNetwork: true
  hostPorts:
  - min: 0
    max: 65535
  # SELinux
  seLinux:
    # SELinux is unused in CaaSP
    rule: 'RunAsAny'
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: flannel
rules:
  - apiGroups: ['extensions']
    resources: ['podsecuritypolicies']
    verbs: ['use']
    resourceNames: ['psp.flannel.unprivileged']
  - apiGroups:
      - ""
    resources:
      - pods
    verbs:
      - get
  - apiGroups:
      - ""
    resources:
      - nodes
    verbs:
      - list
      - watch
  - apiGroups:
      - ""
```

```
    resources:
      - nodes/status
    verbs:
      - patch
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: flannel
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: flannel
subjects:
- kind: ServiceAccount
  name: flannel
  namespace: kube-system
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: flannel
  namespace: kube-system
---
kind: ConfigMap
apiVersion: v1
metadata:
  name: kube-flannel-cfg
  namespace: kube-system
  labels:
    tier: node
    app: flannel
data:
  cni-conf.json: |
    {
      "name": "cbr0",
      "cniVersion": "0.3.1",
      "plugins": [
        {
          "type": "flannel",
          "delegate": {
            "hairpinMode": true,
            "isDefaultGateway": true
          }
        },
        {
          "type": "portmap",
          "capabilities": {
            "portMappings": true
          }
        }
      ]
```

```
    }
  net-conf.json: |
    {
      "Network": "10.30.0.0/16",
      "Backend": {
        "Type": "vxlan"
      }
    }
---
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: kube-flannel-ds-amd64
  namespace: kube-system
  labels:
    tier: node
    app: flannel
spec:
  selector:
    matchLabels:
      app: flannel
  template:
    metadata:
      labels:
        tier: node
        app: flannel
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                  - key: beta.kubernetes.io/os
                    operator: In
                    values:
                      - linux
                  - key: beta.kubernetes.io/arch
                    operator: In
                    values:
                      - amd64
      hostNetwork: true
      tolerations:
      - operator: Exists
        effect: NoSchedule
      serviceAccountName: flannel
      initContainers:
      - name: install-cni
        image: quay.io/coreos/flannel:v0.11.0-amd64
        command:
        - cp
        args:
        - -f
```

```
        - /etc/kube-flannel/cni-conf.json
        - /etc/cni/net.d/10-flannel.conflist
        volumeMounts:
        - name: cni
          mountPath: /etc/cni/net.d
        - name: flannel-cfg
          mountPath: /etc/kube-flannel/
      containers:
      - name: kube-flannel
        image: quay.io/coreos/flannel:v0.11.0-amd64
        command:
        - /opt/bin/flanneld
        args:
        - --ip-masq
        - --kube-subnet-mgr
        resources:
          requests:
            cpu: "100m"
            memory: "50Mi"
          limits:
            cpu: "100m"
            memory: "50Mi"
        securityContext:
          privileged: false
          capabilities:
            add: ["NET_ADMIN"]
        env:
        - name: POD_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
        - name: POD_NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
        volumeMounts:
        - name: run
          mountPath: /run/flannel
        - name: flannel-cfg
          mountPath: /etc/kube-flannel/
      volumes:
        - name: run
          hostPath:
            path: /run/flannel
        - name: cni
          hostPath:
            path: /etc/cni/net.d
        - name: flannel-cfg
          configMap:
            name: kube-flannel-cfg
---
apiVersion: apps/v1
```

```
kind: DaemonSet
metadata:
  name: kube-flannel-ds-arm64
  namespace: kube-system
  labels:
    tier: node
    app: flannel
spec:
  selector:
    matchLabels:
      app: flannel
  template:
    metadata:
      labels:
        tier: node
        app: flannel
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                  - key: beta.kubernetes.io/os
                    operator: In
                    values:
                      - linux
                  - key: beta.kubernetes.io/arch
                    operator: In
                    values:
                      - arm64
      hostNetwork: true
      tolerations:
      - operator: Exists
        effect: NoSchedule
      serviceAccountName: flannel
      initContainers:
      - name: install-cni
        image: quay.io/coreos/flannel:v0.11.0-arm64
        command:
        - cp
        args:
        - -f
        - /etc/kube-flannel/cni-conf.json
        - /etc/cni/net.d/10-flannel.conflist
        volumeMounts:
        - name: cni
          mountPath: /etc/cni/net.d
        - name: flannel-cfg
          mountPath: /etc/kube-flannel/
      containers:
      - name: kube-flannel
        image: quay.io/coreos/flannel:v0.11.0-arm64
```

```
        command:
        - /opt/bin/flanneld
        args:
        - --ip-masq
        - --kube-subnet-mgr
        resources:
          requests:
            cpu: "100m"
            memory: "50Mi"
          limits:
            cpu: "100m"
            memory: "50Mi"
        securityContext:
          privileged: false
          capabilities:
             add: ["NET_ADMIN"]
        env:
        - name: POD_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
        - name: POD_NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
        volumeMounts:
        - name: run
          mountPath: /run/flannel
        - name: flannel-cfg
          mountPath: /etc/kube-flannel/
      volumes:
        - name: run
          hostPath:
            path: /run/flannel
        - name: cni
          hostPath:
            path: /etc/cni/net.d
        - name: flannel-cfg
          configMap:
            name: kube-flannel-cfg
---
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: kube-flannel-ds-arm
  namespace: kube-system
  labels:
    tier: node
    app: flannel
spec:
  selector:
    matchLabels:
```

```
      app: flannel
  template:
    metadata:
      labels:
        tier: node
        app: flannel
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                  - key: beta.kubernetes.io/os
                    operator: In
                    values:
                      - linux
                  - key: beta.kubernetes.io/arch
                    operator: In
                    values:
                      - arm
      hostNetwork: true
      tolerations:
      - operator: Exists
        effect: NoSchedule
      serviceAccountName: flannel
      initContainers:
      - name: install-cni
        image: quay.io/coreos/flannel:v0.11.0-arm
        command:
        - cp
        args:
        - -f
        - /etc/kube-flannel/cni-conf.json
        - /etc/cni/net.d/10-flannel.conflist
        volumeMounts:
        - name: cni
          mountPath: /etc/cni/net.d
        - name: flannel-cfg
          mountPath: /etc/kube-flannel/
      containers:
      - name: kube-flannel
        image: quay.io/coreos/flannel:v0.11.0-arm
        command:
        - /opt/bin/flanneld
        args:
        - --ip-masq
        - --kube-subnet-mgr
        resources:
          requests:
            cpu: "100m"
            memory: "50Mi"
          limits:
```

```
            cpu: "100m"
            memory: "50Mi"
        securityContext:
          privileged: false
          capabilities:
             add: ["NET_ADMIN"]
        env:
        - name: POD_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
        - name: POD_NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
        volumeMounts:
        - name: run
          mountPath: /run/flannel
        - name: flannel-cfg
          mountPath: /etc/kube-flannel/
      volumes:
        - name: run
          hostPath:
            path: /run/flannel
        - name: cni
          hostPath:
            path: /etc/cni/net.d
        - name: flannel-cfg
          configMap:
            name: kube-flannel-cfg
---
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: kube-flannel-ds-ppc64le
  namespace: kube-system
  labels:
    tier: node
    app: flannel
spec:
  selector:
    matchLabels:
      app: flannel
  template:
    metadata:
      labels:
        tier: node
        app: flannel
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
```

```
      nodeSelectorTerms:
        - matchExpressions:
            - key: beta.kubernetes.io/os
              operator: In
              values:
                - linux
            - key: beta.kubernetes.io/arch
              operator: In
              values:
                - ppc64le
hostNetwork: true
tolerations:
- operator: Exists
  effect: NoSchedule
serviceAccountName: flannel
initContainers:
- name: install-cni
  image: quay.io/coreos/flannel:v0.11.0-ppc64le
  command:
  - cp
  args:
  - -f
  - /etc/kube-flannel/cni-conf.json
  - /etc/cni/net.d/10-flannel.conflist
  volumeMounts:
  - name: cni
    mountPath: /etc/cni/net.d
  - name: flannel-cfg
    mountPath: /etc/kube-flannel/
containers:
- name: kube-flannel
  image: quay.io/coreos/flannel:v0.11.0-ppc64le
  command:
  - /opt/bin/flanneld
  args:
  - --ip-masq
  - --kube-subnet-mgr
  resources:
    requests:
      cpu: "100m"
      memory: "50Mi"
    limits:
      cpu: "100m"
      memory: "50Mi"
  securityContext:
    privileged: false
    capabilities:
       add: ["NET_ADMIN"]
  env:
  - name: POD_NAME
    valueFrom:
      fieldRef:
```

```
              fieldPath: metadata.name
        - name: POD_NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
        volumeMounts:
        - name: run
          mountPath: /run/flannel
        - name: flannel-cfg
          mountPath: /etc/kube-flannel/
      volumes:
        - name: run
          hostPath:
            path: /run/flannel
        - name: cni
          hostPath:
            path: /etc/cni/net.d
        - name: flannel-cfg
          configMap:
            name: kube-flannel-cfg
---
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: kube-flannel-ds-s390x
  namespace: kube-system
  labels:
    tier: node
    app: flannel
spec:
  selector:
    matchLabels:
      app: flannel
  template:
    metadata:
      labels:
        tier: node
        app: flannel
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                  - key: beta.kubernetes.io/os
                    operator: In
                    values:
                      - linux
                  - key: beta.kubernetes.io/arch
                    operator: In
                    values:
                      - s390x
```

```
      hostNetwork: true
      tolerations:
      - operator: Exists
        effect: NoSchedule
      serviceAccountName: flannel
      initContainers:
      - name: install-cni
        image: quay.io/coreos/flannel:v0.11.0-s390x
        command:
        - cp
        args:
        - -f
        - /etc/kube-flannel/cni-conf.json
        - /etc/cni/net.d/10-flannel.conflist
        volumeMounts:
        - name: cni
          mountPath: /etc/cni/net.d
        - name: flannel-cfg
          mountPath: /etc/kube-flannel/
      containers:
      - name: kube-flannel
        image: quay.io/coreos/flannel:v0.11.0-s390x
        command:
        - /opt/bin/flanneld
        args:
        - --ip-masq
        - --kube-subnet-mgr
        resources:
          requests:
            cpu: "100m"
            memory: "50Mi"
          limits:
            cpu: "100m"
            memory: "50Mi"
        securityContext:
          privileged: false
          capabilities:
             add: ["NET_ADMIN"]
        env:
        - name: POD_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
        - name: POD_NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
        volumeMounts:
        - name: run
          mountPath: /run/flannel
        - name: flannel-cfg
          mountPath: /etc/kube-flannel/
```

```
      volumes:
        - name: run
          hostPath:
            path: /run/flannel
        - name: cni
          hostPath:
            path: /etc/cni/net.d
        - name: flannel-cfg
          configMap:
            name: kube-flannel-cfg
```

flannel 国内镜像：

```
docker pull quay-mirror.qiniu.com/coreos/flannel:v0.11.0
docker tag quay-mirror.qiniu.com/coreos/flannel:v0.11.0
quay.io/coreos/flannel:v0.11.0-amd64
```

部署Node节点

部分内容和master节点雷同，比如运行时安装与配置，kubelet安装。

1. 拉取国内镜像。
2. 部署kubelet， 配置join加入集群。

TASK 9

部署petapp到k8s集群。

（扩展：挂载在nginx后，scale到4个pod，创建NodePortService）

kubectl运行的三种方式

1. Generators (Run, Expose)

```
kubectl run  --generator=run-pod/v1  nginx --image=nginx --image-pull-
policy=IfNotPresent
```

2. 用解释的方法（Create）

```
kubectl create deployment --image=nginx  nginx
```

3. 用声明的方法（Apply）

```
# deployment.yaml

apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: nginx-deployment
```

```
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2 # tells deployment to run 2 pods matching the template
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 80
```

```
kubectl apply -f https://k8s.io/examples/application/deployment.yaml
```

K8S：存储部分

TASK 10

emptyDir

当 Pod 指定到某个节点上时，首先创建的是一个 `emptyDir` 卷，并且只要 Pod 在该节点上运行，卷就一直存在。 就像它的名称表示的那样，卷最初是空的。 尽管 Pod 中的容器挂载 `emptyDir` 卷的路径可能相同也可能不同，但是这些容器都可以读写 `emptyDir` 卷中相同的文件。 当 Pod 因为某些原因被从节点上删除时， `emptyDir` 卷中的数据也会永久删除。

挂载一个emptyDir的pod

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pd
spec:
  containers:
  - image: k8s.gcr.io/test-webserver
    name: test-container
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
  volumes:
  - name: cache-volume
    emptyDir: {}
```

TASK 11

搭建NFS（Network File System）

service端的安装

安装包：

```
sudo apt-get install nfs-kernel-server nfs-common
```

创建共享目录：

```
sudo mkdir -p /var/nfsshare/xxx
sudo chmod -R 777 /var/nfsshare/xxx
```

在 /etc/exports 加入：

```
/var/nfsshare/xxx    xxx.xxx.xxx.*
(rw,sync,no_root_squash,no_all_squash)
```

或者类似：

```
/var/nfsshare/xxx    *(rw,sync,no_root_squash,no_all_squash)
```

启动服务：

```
/etc/init.d/nfs-kernel-server restart
```

TASK 12

pvc与pv的绑定

hostpath 挂载：

```
kind: PersistentVolume
metadata:
  name: data
  namespace: data
spec:
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: /home/data
---
```

```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: data
  namespace: data
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
  volumeName: "data"
```

使用nfs卷进行挂载：

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: data
  namespace: data
spec:
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteMany
  nfs:
    # FIXME: use the right IP
    server: 172.19.0.11
    path: "/var/nfsshare/data"
  persistentVolumeReclaimPolicy: Recycle
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: data
  namespace: data
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 5Gi
  volumeName: "data"
```

K8S：网络部分

TASK 13

nginx ingress controller的安装

配置nginx ingress controller

我们这里使用独立的ingress controller，可以使用 nginx-ingress。nginx ingress controller可以使用多种方式部署（比如pod或daemonset），也可以使用helm完成部署。

这里给出daemonset直接配置的部署方式：

```
kubectl apply -f common/ns-and-sa.yaml
kubectl apply -f common/default-server-secret.yaml
kubectl apply -f common/nginx-config.yaml
kubectl apply -f rbac/rbac.yaml
kubectl apply -f daemon-set/nginx-ingress.yaml
```

一些自定义的配置，需要修改common/nginx-config.yaml，比如：

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: nginx-config
  namespace: nginx-ingress
data:
  proxy-connect-timeout: "10s"
  proxy-read-timeout: "10s"
  client-max-body-size: "100m"
```

TASK 14

根据域名的不同将流量导向指定的app

apple-app

```
kind: Pod
apiVersion: v1
metadata:
  name: apple-app
  labels:
    app: apple
spec:
  containers:
    - name: apple-app
      image: hashicorp/http-echo
      args:
        - "-text=apple"
---
```

```
kind: Service
apiVersion: v1
metadata:
  name: apple-service
spec:
  selector:
    app: apple
  ports:
    - port: 5678 # Default port for image
```

banana-app

```
kind: Pod
apiVersion: v1
metadata:
  name: banana-app
  labels:
    app: banana
spec:
  containers:
    - name: banana-app
      image: hashicorp/http-echo
      args:
        - "-text=banana"
---
kind: Service
apiVersion: v1
metadata:
  name: banana-service
spec:
  selector:
    app: banana
  ports:
    - port: 5678 # Default port for image
kubectl apply -f apple.yaml
kubectl apply -f banana.yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: example-ingress
  annotations:
    ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
        - path: /apple
          backend:
            serviceName: apple-service
            servicePort: 5678
        - path: /banana
```

```
          backend:
            serviceName: banana-service
            servicePort: 5678
```

```
kubectl apply -f apple.yaml
kubectl apply -f banana.yaml
```

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: example-ingress
  annotations:
    ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
        - path: /apple
          backend:
            serviceName: apple-service
            servicePort: 5678
        - path: /banana
          backend:
            serviceName: banana-service
            servicePort: 5678
```

```
kubectl apply -f ingress.yaml
```

K8S：监控部分

TASK 15

metric server的安装

1. 手动安装metrics-server：

```
git clone https://github.com/kubernetes-incubator/metrics-server.git
cd metrics-server/
kubectl create -f deploy/1.8+/
```

2. 拉取国内镜像

```
images=(
    metrics-server-amd64:v0.3.6
)

for imageName in ${images[@]} ; do
    docker pull registry.cn-
hangzhou.aliyuncs.com/google_containers/$imageName
    docker tag registry.cn-
hangzhou.aliyuncs.com/google_containers/$imageName
gcr.io/google_containers/$imageName
    docker rmi registry.cn-
hangzhou.aliyuncs.com/google_containers/$imageName
done
```

deploy前要修改参数， 找到

```
      - name: metrics-server
        image: k8s.gcr.io/metrics-server-amd64:v0.3.6
        args:
          - --cert-dir=/tmp
          - --secure-port=4443
```

注意添加args

```
--kubelet-insecure-tls=true
--kubelet-preferred-address-types=InternalIP,Hostname,ExternalIP
```

TASK 16

kube dashboard的安装

```
kubectl apply -f
https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0-
beta6/aio/deploy/recommended.yaml
```

TASK 17

使用kubectl top以及kube dashboard观察k8s平台的指标

创建访问用的USER

admin-user-sa.yaml

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kubernetes-dashboard
```

admin-user-crb.yaml

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kubernetes-dashboard
root@debian-11:~/app/dashboard# cat admin-user-crb.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-user
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin-user
  namespace: kubernetes-dashboard
```

获取secret token

```
kubectl -n kubernetes-dashboard describe secret `kubectl -n kubernetes-
dashboard get secret | grep admin-user | awk '{print $1}'`
```

进阶实战

TASK 18

创建Node节点，动手配置TLS加入kube集群。