项目2：追踪k8s中的用户操作的执行流程

1.搭建一个至少有两个节点的k8s集群

1.1 下载所需文件

下载并安装 VirtualBox

下载用于安装操作系统的 ISO 镜像文件

Ubuntu Server 22.04.5 LTS

VirtualBox中设置虚拟机

两个网卡

网卡1：用于内部通信，内部网络，k8s-net

网卡2：用于访问互联网，网络地址转换NAT，

1.2 启动虚拟机和引导安装程序

安装语言：English

网络配置

Adapter 1，enp0s3，静态 IP，内部网络

IP地址：192.168.56.10（control）；192.168.56.20（worker）

子网掩码：192.168.56.0/24

其他留空

用户和服务器信息配置

用户名：zhennan

密码：zhennan

服务器名称：k8s-control，k8s-worker

1.3 重新启动并首次登录

用户名：zhennan

密码：zhennan

1.4 在所有节点上执行通用设置

确保网络连通性：

```
# 在 k8s-control 上
ping 192.168.56.20
# 在 k8s-worker 上
ping 192.168.56.10
```

确认两台虚拟机都可以访问互联网（用于下载软件包）。

```
ping google.com
```

更新系统并安装必要的工具：

```
sudo apt update
sudo apt upgrade -y
sudo apt install -y apt-transport-https ca-certificates curl software-properties-
common
```

禁用 Swap：

```
sudo swapoff -a
# 为了永久禁用，编辑 /etc/fstab 文件，注释掉包含 swap 的那一行
sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab
```

配置内核参数和加载模块：

```
sudo swapoff -a
# 为了永久禁用
sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab
```

安装容器运行时 (Containerd)：

```
# 安装 containerd
sudo apt install -y containerd

# 配置 containerd，使其使用 systemd cgroup driver
sudo mkdir -p /etc/containerd
containerd config default | sudo tee /etc/containerd/config.toml
sudo sed -i 's/SystemdCgroup = false/SystemdCgroup = true/g'
/etc/containerd/config.toml

# 重启并启用 containerd 服务
sudo systemctl restart containerd
sudo systemctl enable containerd
```

安装 Kubernetes 组件 (kubeadm, kubelet, kubectl)：

```
# 添加 Kubernetes APT 仓库的 GPG 密钥
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --
dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
# 如果上面的v1.30的URL在2025年5月之后失效，可以尝试寻找最新的稳定版URL

# 添加 Kubernetes APT 仓库
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' | sudo tee
/etc/apt/sources.list.d/kubernetes.list
# 如果上面的v1.30的URL在2025年5月之后失效，可以尝试寻找最新的稳定版URL

# 更新包列表并安装
sudo apt update
sudo apt install -y kubelet kubeadm kubectl
```

```
# 锁定版本，防止自动更新造成不兼容
sudo apt-mark hold kubelet kubeadm kubectl
```

确保 kubelet 服务已启动并启用：

```
sudo systemctl enable --now kubelet
```

1.5 初始化控制平面节点 (仅在 k8s-control 上执行)

初始化集群：

```
zhennan@k8s-control:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --
apiserver-advertise-address=192.168.56.10
I0511 12:17:57.725417   22496 version.go:256] remote version is much newer:
v1.33.0; falling back to: stable-1.30
[init] Using Kubernetes version: v1.30.12
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your
internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config
images pull'
W0511 12:18:00.311642   22496 checks.go:844] detected that the sandbox image
"registry.k8s.io/pause:3.8" of the container runtime is inconsistent with that
used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.9" as the CRI
sandbox image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [k8s-control kubernetes
kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local]
and IPs [10.96.0.1 192.168.56.10]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [k8s-control localhost]
and IPs [192.168.56.10 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [k8s-control localhost]
and IPs [192.168.56.10 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "super-admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
```

```
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[kubelet-start] Writing kubelet environment file with flags to file
"/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Writing kubelet configuration to file
"/var/lib/kubelet/config.yaml"
[kubelet-start] Starting the kubelet
[wait-control-plane] Waiting for the kubelet to boot up the control plane as
static Pods from directory "/etc/kubernetes/manifests"
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz.
This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 503.786107ms
[api-check] Waiting for a healthy API server. This can take up to 4m0s
[api-check] The API server is healthy after 21.504720538s
[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in
the "kube-system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config" in namespace kube-system with the
configuration for the kubelets in the cluster
[upload-certs] Skipping phase. Please see --upload-certs
[mark-control-plane] Marking the node k8s-control as control-plane by adding the
labels: [node-role.kubernetes.io/control-plane node.kubernetes.io/exclude-from-
external-load-balancers]
[mark-control-plane] Marking the node k8s-control as control-plane by adding the
taints [node-role.kubernetes.io/control-plane:NoSchedule]
[bootstrap-token] Using token: 9jqcxp.e59mw01hb13v1bjl
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC
Roles
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to get
nodes
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to post
CSRs in order for nodes to get long term certificate credentials
[bootstrap-token] Configured RBAC rules to allow the csrapprover controller
automatically approve CSRs from a Node Bootstrap Token
[bootstrap-token] Configured RBAC rules to allow certificate rotation for all
node client certificates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public"
namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a
rotatable kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

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

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

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 192.168.56.10:6443 --token 9jqcxp.e59mw01hb13v1bjl \
        --discovery-token-ca-cert-hash
sha256:000b1058d7b2f4e05990cde4723abdb4ac16bd35de2e8d94e744ad71d116bc20
```

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

部署 Pod 网络插件：

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

几分钟后检查节点状态：

```
zhennan@k8s-control:~$ kubectl get nodes
NAME          STATUS    ROLES           AGE     VERSION
k8s-control   Ready     control-plane   3m58s   v1.30.12
```

1.6 将工作节点加入集群 (仅在 k8s-worker 上执行)

```
kubeadm join 192.168.56.10:6443 --token 9jqcxp.e59mw01hb13v1bjl \
        --discovery-token-ca-cert-hash
sha256:000b1058d7b2f4e05990cde4723abdb4ac16bd35de2e8d94e744ad71d116bc20
```

1.7 验证集群状态 (仅在 k8s-control 上执行)

几分钟后检查节点状态：

```
zhennan@k8s-control:~$ kubectl get nodes
NAME          STATUS    ROLES           AGE     VERSION
k8s-control   Ready     control-plane   8m43s   v1.30.12
k8s-worker    Ready     <none>          4m10s   v1.30.12
```

2.追踪跨节点执行命令的执行过程

2.1 准备工作

检查网络连接：

```
curl www.google.com
```

验证集群状态：

```
zhennan@k8s-control:~$ kubectl get nodes
NAME          STATUS   ROLES           AGE   VERSION
k8s-control   Ready    control-plane   90m   v1.30.12
k8s-worker    Ready    <none>          89m   v1.30.12
```

启动终端会话:

- **终端 C1 (k8s-control)**: `kube-apiserver`
- **终端 C2 (k8s-control)**: `kube-scheduler`
- **终端 C3 (k8s-control)**: `kubectl` 命令
- **终端 W1 (k8s-worker)**: `kubelet`
- **终端 W2 (k8s-worker)**: `containerd`
- **终端 C4 (k8s-control)**: 监控 Pod 状态

2.2 启动 `strace`

在 k8s-control - 终端 C1 (跟踪 kube-apiserver):

```
sudo strace -p $(pgrep -o kube-apiserver) -o /tmp/trace_apiserver_$(date +%s).txt
-f -tt -T -s 4096 -v -y
```

在 k8s-control - 终端 C2 (跟踪 kube-scheduler):

```
sudo strace -p $(pgrep -o kube-scheduler) -o /tmp/trace_scheduler_$(date +%s).txt
-f -tt -T -s 4096 -v -y
```

在 k8s-worker - 终端 W1 (跟踪 kubelet):

```
sudo strace -p $(pgrep -o kubelet) -o /tmp/trace_kubelet_$(date +%s).txt -f -tt -
T -s 4096 -v -y
```

在 k8s-worker - 终端 W2 (跟踪 containerd):

```
sudo strace -p $(pgrep -o containerd) -o /tmp/trace_containerd_$(date +%s).txt -f
-tt -T -s 4096 -v -y
```

2.3 启动 Pod 状态监控

在 k8s-control - 终端 C4:

```
kubectl get pods -w
```

2.4 执行 `kubectl` 命令创建 Pod

在 k8s-control - 终端 C3:

```
sudo strace -o /tmp/trace_kubectl_$(date +%s).txt -f -tt -T -s 4096 -v -y kubectl
run mynginx-trace-test --image=nginx --kubeconfig=/home/zhennan/.kube/config
```

2.5 等待 Pod 创建完成

观察终端 C4，等待 `mynginx-trace-test` Pod 的状态变为 `Running`

```
zhennan@k8s-control:~$ kubectl get pods -w
NAME                 READY   STATUS    RESTARTS   AGE
mynginx-trace-test   0/1     Pending   0          0s
mynginx-trace-test   0/1     Pending   0          0s
mynginx-trace-test   0/1     ContainerCreating   0          0s
mynginx-trace-test   1/1     Running             0          7s
```

2.6 停止所有 `strace` 进程

按 Ctrl+C 停止所有 `strace` 进程

2.7 收集和分析跟踪文件

`k8s-control`：

- `trace_apiserver_<timestamp>.txt`
- `trace_scheduler_<timestamp>.txt`
- `trace_kubectl_<timestamp>.txt`

`k8s-worker`：

- `trace_kubelet_<timestamp>.txt`
- `trace_containerd_<timestamp>.txt`

分析

```
┌─ 控制节点 ─────────────────────────────────────────────┐
│kubectl            kube-apiserver          kube-scheduler│
│  execve           (HTTPS 6443)            (watch/bind)  │
│     │ socket+connect ─────▶ │    TLS read/write │        │
└─────────────────────────────────────────────────────────┘

               ▲                           │
               │                           ▼
┌─ 工作节点 ─────────────────────────────────────────────┐
│kubelet  (watch) → unix write → containerd → runc → nginx│
└─────────────────────────────────────────────────────────┘
```

遇到的问题和解决方法

问题1：虚拟机无法正常启动

VirtualBox 虚拟机无法正常启动，报错信息如下：

```
Error relaunching VirtualBox VM process: 5
Command line:'C:\ProgramFiles\Oracle\VirtualBox\VirtualBoxVM.exe 60eaff78-4bdd-
042d-2e72-669728efd737-suplib-3rdchild --comment k8s-control --startvm 44536571-
f95a-404e-b235-d3a698526007 --no-startvm-errormsgbox "--sup-hardening-
log=D:\VirtualBoxVMs\k8s-control\Logs\VBoxHardening.log"' (rc=-104)
Please try reinstalling VirtualBox.
where: supR3HardenedWinReSpawn what:5 VERR_INVALID_NAME(-104) - Invalid
(malformed) file/path name.
```

解决方法:

1. Completely uninstall any VirtualBox currently installed
2. Restart the computer
3. Install the latest version of VirtualBox
4. After install completes do not restart the computer

参考资料:

https://forums.virtualbox.org/viewtopic.php?f=6&t=82277

https://stackoverflow.com/questions/39850418/oracle-vm-error-at-startup

问题2：虚拟机启动文件系统错误

关键报错信息:

```
/dev/mapper/ubuntu--vg-ubuntu--lv: UNEXPECTED INCONSISTENCY; RUN fsck MANUALLY.
fsck exited with status code 4
Failure: File system check of the root filesystem failed
(initramfs)
```

解决方法:

```
sudo fsck -y /dev/mapper/ubuntu--vg-ubuntu--lv
```

问题3：互联网连接问题

问题描述:

中国大陆地区无法正常访问 Nginx 服务，无法拉取镜像

解决方法:

使用 v2ray 代理，在自己电脑上配置 v2ray 代理，启用Tun模式，路由规则设置为全局，然后虚拟机通过 v2ray 代理访问互联网

验证:

```
curl www.google.com
```

问题4：Kubernetes 集群重启

问题描述：

worker节点无法正常运行，需要重启集群

解决方法：

在工作节点 (k8s-worker) 上执行：

```
sudo systemctl stop kubelet
sudo kubeadm reset -f

sudo rm -rf /etc/cni/net.d

# 刷新所有规则，删除所有非默认链，并重置计数器
sudo iptables -F
sudo iptables -X
sudo iptables -Z
sudo iptables -t nat -F
sudo iptables -t nat -X
sudo iptables -t nat -Z
sudo iptables -t mangle -F
sudo iptables -t mangle -X
sudo iptables -t mangle -Z
sudo iptables -t raw -F
sudo iptables -t raw -X
sudo iptables -t raw -Z
sudo iptables -P INPUT ACCEPT
sudo iptables -P FORWARD ACCEPT
sudo iptables -P OUTPUT ACCEPT
sudo ipvsadm --clear

sudo rm -rf /var/lib/kubelet/*

sudo systemctl restart containerd
```

在控制平面节点 (k8s-control) 上执行：

```
sudo systemctl stop kubelet
sudo kubeadm reset -f

sudo rm -rf /etc/cni/net.d

sudo rm -rf $HOME/.kube

sudo iptables -F
sudo iptables -X
sudo iptables -Z
sudo iptables -t nat -F
sudo iptables -t nat -X
sudo iptables -t nat -Z
sudo iptables -t mangle -F
sudo iptables -t mangle -X
sudo iptables -t mangle -Z
sudo iptables -t raw -F
sudo iptables -t raw -X
sudo iptables -t raw -Z
sudo iptables -P INPUT ACCEPT
sudo iptables -P FORWARD ACCEPT
sudo iptables -P OUTPUT ACCEPT
sudo ipvsadm --clear

sudo rm -rf /var/lib/etcd
sudo rm -rf /var/lib/kubelet/*

sudo systemctl restart containerd
```

然后重新按照1.5节步骤重新初始化 Kubernetes 集群

问题5：Kubernetes 集群网络问题

问题描述：

无法正常创建Pod，检查发现 Flannel Pods 状态异常：kube-flannel-ds-sgk4s 和 kube-flannel-ds-tjm6m 这两个 Pod 都处于 CrashLoopBackOff 状态。这意味着它们正在反复启动失败。Flannel 是负责集群内部 Pod 之间网络通信的组件。如果 Flannel 无法正常工作，Pod 将无法获取 IP 地址，也无法相互通信，这将导致集群大部分功能瘫痪。

```
zhennan@k8s-control:~$ kubectl get pods --all-namespaces
NAMESPACE      NAME                                  READY   STATUS
RESTARTS      AGE
default        nginx-test-basic                      0/1     ContainerCreating
 0             26m
kube-flannel   kube-flannel-ds-sgk4s                 0/1     CrashLoopBackOff
8 (81s ago)   17m
kube-flannel   kube-flannel-ds-tjm6m                 0/1     CrashLoopBackOff
8 (88s ago)   17m
kube-system    coredns-55cb58b774-jdj9j              0/1     ContainerCreating
 0             52m
kube-system    coredns-55cb58b774-qv2q4              0/1     ContainerCreating
 0             52m
kube-system    etcd-k8s-control                      1/1     Running
 3             52m
kube-system    kube-apiserver-k8s-control            1/1     Running
 0             52m
kube-system    kube-controller-manager-k8s-control   1/1     Running
 0             52m
kube-system    kube-proxy-bxl69                      1/1     Running
 0             52m
kube-system    kube-proxy-h7fz9                      1/1     Running
 0             51m
kube-system    kube-scheduler-k8s-control            1/1     Running
 0             52m
```

问题分析：

查看 Flannel Pod 的日志：

```
kubectl logs kube-flannel-ds-sgk4s -n kube-flannel
kubectl logs kube-flannel-ds-tjm6m -n kube-flannel
```

输出信息：

```
zhennan@k8s-control:~$ kubectl logs kube-flannel-ds-sgk4s -n kube-flannel
kubectl logs kube-flannel-ds-tjm6m -n kube-flannel
Defaulted container "kube-flannel" out of: kube-flannel, install-cni-plugin
(init), install-cni (init)
Error from server (NotFound): the server could not find the requested resource (
pods/log kube-flannel-ds-sgk4s)
Defaulted container "kube-flannel" out of: kube-flannel, install-cni-plugin
(init), install-cni (init)
I0521 06:36:57.949621       1 main.go:211] CLI flags config:
{etcdEndpoints:http://127.0.0.1:4001,http://127.0.0.1:2379
etcdPrefix:/coreos.com/network etcdKeyfile: etcdCertfile: etcdCAFile:
etcdUsername: etcdPassword: version:false kubeSubnetMgr:true kubeApiUrl:
kubeAnnotationPrefix:flannel.alpha.coreos.com kubeConfigFile: iface:[]
ifaceRegex:[] ipMasq:true ifaceCanReach: subnetFile:/run/flannel/subnet.env
publicIP: publicIPv6: subnetLeaseRenewMargin:60 healthzIP:0.0.0.0 healthzPort:0
iptablesResyncSeconds:5 iptablesForwardRules:true netConfPath:/etc/kube-
flannel/net-conf.json setNodeNetworkUnavailable:true}
W0521 06:36:57.951975       1 client_config.go:618] Neither --kubeconfig nor --
master was specified.  Using the inClusterConfig.  This might not work.
I0521 06:36:57.972982       1 kube.go:139] Waiting 10m0s for node controller to
sync
I0521 06:36:57.973356       1 kube.go:469] Starting kube subnet manager
I0521 06:36:58.973827       1 kube.go:146] Node controller sync successful
I0521 06:36:58.973941       1 main.go:231] Created subnet manager: Kubernetes
Subnet Manager - k8s-control
I0521 06:36:58.973948       1 main.go:234] Installing signal handlers
I0521 06:36:58.975269       1 main.go:468] Found network config - Backend type:
vxlan
E0521 06:36:58.975367       1 main.go:268] Failed to check br_netfilter: stat
/proc/sys/net/bridge/bridge-nf-call-iptables: no such file or directory
```

问题核心：br_netfilter 模块缺失或配置不当

解决方法：

在所有 Kubernetes 节点上

1. 加载 br_netfilter 模块

```
sudo modprobe br_netfilter
```

2. 设置内核参数

创建一个配置文件 /etc/sysctl.d/k8s.conf，添加以下内容：

```
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
```

3. 应用配置

```
sudo sysctl --system
```