K8s安全专家 CKS认证

宽哥带你学

ke.qq.com/course/2738602

K8s 技术交流群

和宽哥一对一交流

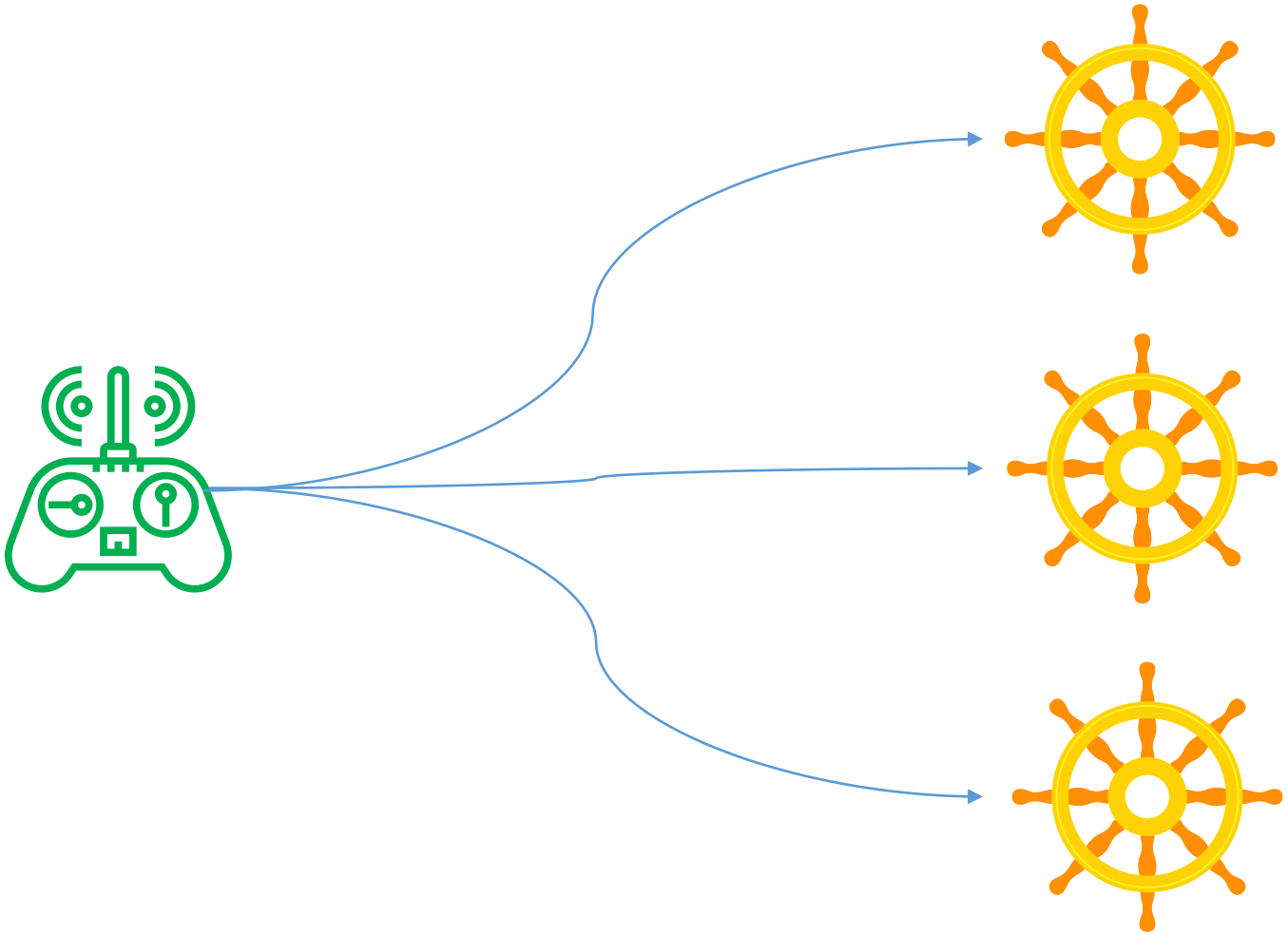
K8s核心资源扫盲

Namespace：命令空间，相当于把一个集群虚拟成多个集群，部分资源具有隔离性
Pod：K8s最小单元，可以简单理解为一个、多个或一组容器的集合。
Deployment：无状态应用控制器
StatefulSet：有状态应用控制器
DaemonSet：守护进程控制器
Service：逻辑上的一组Pod，用于集群服务通信
Ingress：集群服务入口，通过域名发布服务
ConfigMap：K8s配置管理
Secret：K8s加密数据管理

系统安装

- ✓ VMWare 14
- ✓ Ubuntu 18.04
- ✓ K8s Master一台 2C4G+
- ✓ K8s Node两台+ 2C4G+

考试环境准备



控制端

集群

安装K8s

首先切换到root用户，然后打开sshd的配置文件，修改允许Root用户登录：

```
# vim /etc/ssh/sshd_config
PermitRootLogin yes # 改为yes
```

重启sshd即可：

```
# systemctl daemon-reload
# systemctl restart sshd
```

安装Kubernetes：https://www.sealyun.com/instructions

```
# wget -c https://sealyun.oss-cn-beijing.aliyuncs.com/latest/sealos && \
    chmod +x sealos && mv sealos /usr/bin
# wget -c https://sealyun.oss-cn-beijing.aliyuncs.com/05a3db657821277f5f3b92d834bbaf98-v1.22.0/kube1.22.0.tar.gz
# sealos init --passwd '13343323' \
    --master 192.168.1.6  \
    --node 192.168.1.7 \
    --pkg-url /root/kube1.22.0.tar.gz \
    --version v1.22.0
```

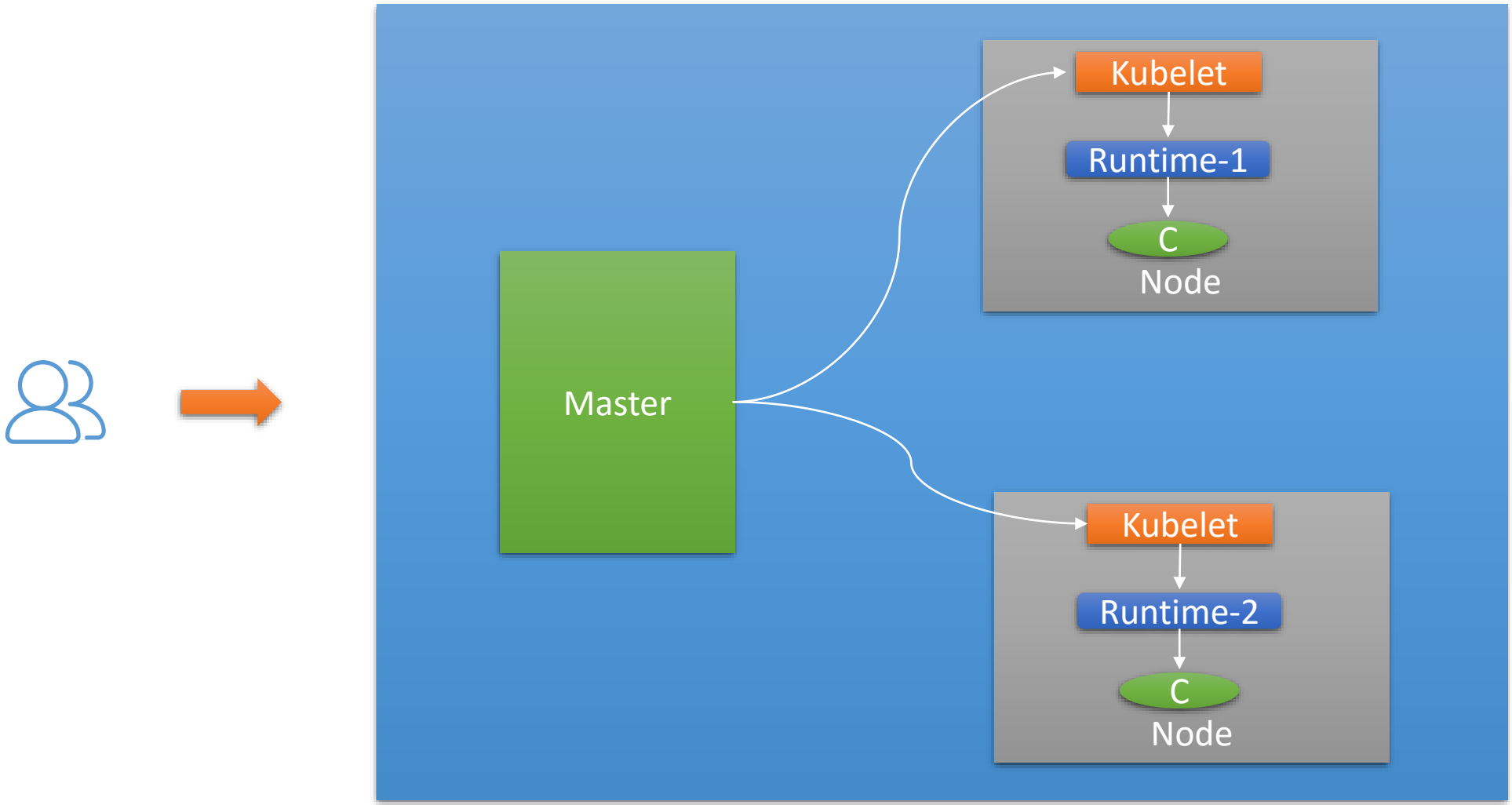
考题分析

RuntimeClass

RuntimeClass：是一个让Pod选择容器运行时的一个特性，用于多运行时环境。

2014年6月	Kubernetes 1.3	Kubernetes 1.5
Kubernetes开源，Docker是Kubernetes唯一的运行时	RKT合入Kubernetes，成为第二个运行时	Kubernetes推出CRI，符合该标准就能成为运行时，比如Kata、gVisor

RuntimeClass工作流程



Kube-Bench

Kube-bench：基于Go语言开发，可以帮助研究人员和使用人员对部署的Kubernetes集群进行安全检测。可以基于CIS Kubernetes Benchmark准则检查出当前集群存在的隐患，是一款针对Kubernetes的安全检测工具。

[INFO] 1 Master Node Security Configuration
[INFO] 1.1 Master Node Configuration Files
[PASS] 1.1.1 Ensure…
…
[WARN] 1.1.9 Ensure …
[FAIL] 1.2.16 Ensure that the admission control plugin PodSecurityPolicy is set (Automated)
== Remediations master ==
1.1.9 Run the below command (based on the file location on your system) on the master node.
For example,chmod 644 <path/to/cni/files>
== Summary master ==
45 checks PASS
…
== Summary total ==
45 checks PASS
10 checks FAIL
10 checks WARN
0 checks INFO

Pod Security Policy

Pod 安全策略（Pod Security Policy）是集群级别的资源，它能够控制 Pod 在安全相关的各个行为，比如不能用特权模式运行。 PodSecurityPolicy 对象定义了一组 Pod 运行时必须遵循的条件及相关字段的默认值，只有 Pod 满足这些条件才能创建。
注意：PSP在1.21废弃，1.25版本删除

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: restricted
spec:
  privileged: false # 禁止特权容器的Pod
  allowPrivilegeEscalation: false # 禁止容器进程提升权限
  requiredDropCapabilities: # 禁止使用的Linux能力
    - ALL
  volumes: # 允许Pod使用的volume类型
    - 'configMap'
  hostNetwork: false # 禁止使用宿主机网络
  hostIPC: false # 禁止Pod共享宿主机IPC命名空间
  hostPID: false  # 禁止共享宿主机进程空间
  runAsUser:  # 设置运行容器的用户ID
    rule: 'MustRunAsNonRoot'
  seLinux: # 设置Linux参数
     rule: 'RunAsAny'
  fsGroup: # 设置访问volume的Group ID
    rule: 'MustRunAs'
  readOnlyRootFilesystem: false # 根文件系统是否是只读属性
```

审计日志

Kubernetes审计功能可以记录每个用户和使用Kubernetes API的应用以及控制平面自身的活动。
比如：

- 发生了什么？
- 什么时候发生的？
- 谁触发的？
- 对哪些对象操作的？
- 从哪里触发的？
- 从哪里观察到的？
- 后续处理行为是什么？

审计日志根据不同的策略将每个请求的不同阶段的操作日志记录到文件或其他后端（通过webhook）。

请求阶段如下：

- RequestReceived：审计处理器接收到请求后，并且在分配其他处理之前的事件，一般不记录该阶段日志；
- ResponseStarted：开始响应Header信息，但未响应消息体之前的事件，一般用于长时间运行的请求才会有这个阶段，比如watch；
- ResponseComplete：事件响应完毕，并且没有更多数据需要传输的时候；
- Panic：访问出现Panic错误时记录。

日志记录等级：

- None：不记录日志；
- Metadata：记录请求的元数据（请求的用户、时间戳、资源、verb等），但不包括Request和RequestResponse；
- Request：记录包括Metadata和请求体，但不包含RequestResponse；
- RequestResponse：包含Metadata、Request、响应的消息体。

审计日志配置示例


```
apiVersion: audit.k8s.io/v1 # This is required.
kind: Policy
omitStages: # 忽略某个阶段
  - "RequestReceived" # 不记录RequestReceived阶段的事件
rules: # 审计规则配置
  - level: RequestResponse # 记录RequestResponse级别的日志
    resources: # 记录哪些资源
    - group: ""
      resources: ["pods"]
  - level: Metadata 记录Metadata级别的日志
    resources:
    - group: ""
      resources: ["pods/log", "pods/status"]
  - level: None # 不记录日志
    resources:
    - group: ""
      resources: ["configmaps"]
      resourceNames: ["controller-leader"] # 不记录名字为controller-leader的ConfigMap的事件
  - level: None
    users: ["system:kube-proxy"]
    verbs: ["watch"]
    resources:
    - group: "" # core API group
      resources: ["endpoints", "services"]
  - level: None
    userGroups: ["system:authenticated"]
    nonResourceURLs:
    - "/api*" # Wildcard matching.
    - "/version"
  - level: Request
    resources:
    - group: "" # core API group
      resources: ["configmaps"]
    namespaces: ["kube-system"]
- level: Request
    resources:
    - group: "" # core API group
    - group: "extensions" # Version of group should NOT be included
  - level: Metadata
    omitStages:
      - "RequestReceived"
```

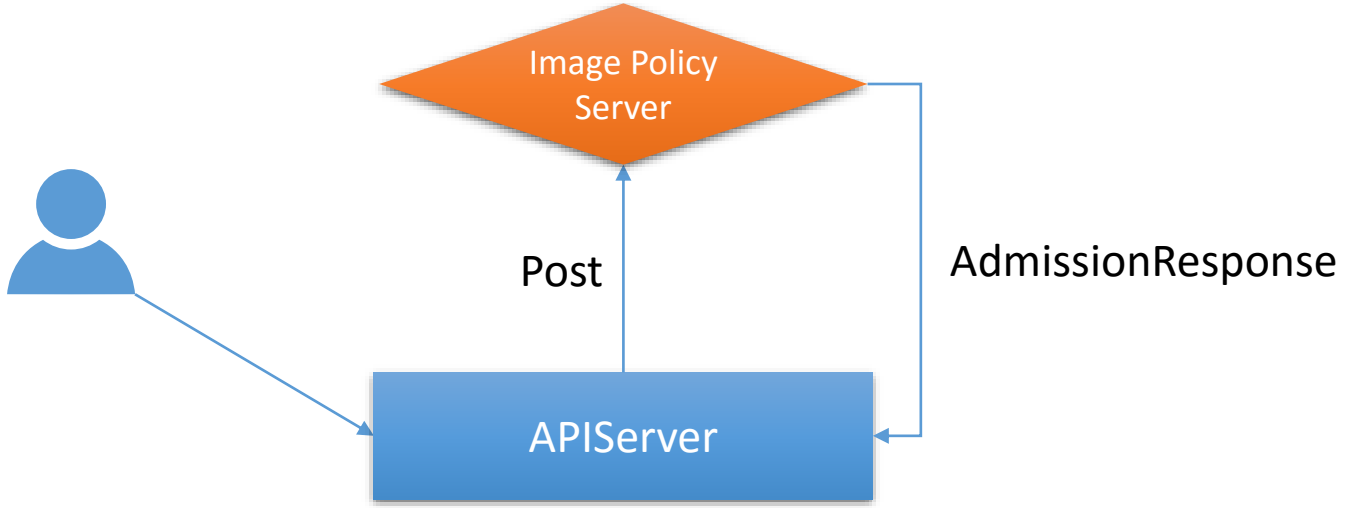
开启审计日志

 # vim /etc/kubernetes/manifests/kube-apiserver.yaml

```
- --audit-policy-file=/etc/kubernetes/logpolicy/sample-policy.yaml  # 审计策略文件
- --audit-log-path=/var/log/kubernetes/kubernetes-logs.txt    # 审计日志文件
- --audit-log-maxage=30    # 日志文件最大保留天数
- --audit-log-maxbackup=10  # 审计日志文件最大保留数据
- --audit-log-maxsize=100 # 审计日志最大文件大小,MB
```

ImagePolicyWebhook

ImagePolicyWebhook：评估镜像的准入控制器，通过该Webhook可以决定镜像是否合规，不合规会拒绝创建Pod。规则配置通过--admission-control-config-file参数指定。



AppArmor

AppArmor是一个Linux内核安全模块，可以通过一个安全配置文件限制程序的功能。比如可以指定某个程序可以读、写或运行某些文件，也可以限制能够打开一些网络端口等。

```
#include <tunables/global> # 导入变量

profile nginx-profile-1 flags=(attach_disconnected) { #设定配置文件的名称解析为相对路径
  #include <abstractions/base>

  file, # 文件类型的限制

  # Deny all file writes.
  deny /** rwx,
}
```

Thanks.

宽哥带你学