讲师：李振良（阿良）

今天课题：《监控、审计和运行时安全》

学院官网：www.aliangedu.cn

阿良个人微信

DevOps技术栈公众号

第六章 监控、审计和运行时安全

- 分析容器系统调用：Sysdig
- 监控容器运行时：Falco
- Kubernetes 审计日志

分析容器系统调用：Sysdig

Sysdig：一个非常强大的系统监控、分析和故障排查工具。

汇聚 strace+tcpdump+htop+iftop+lsof 工具功能于一身！

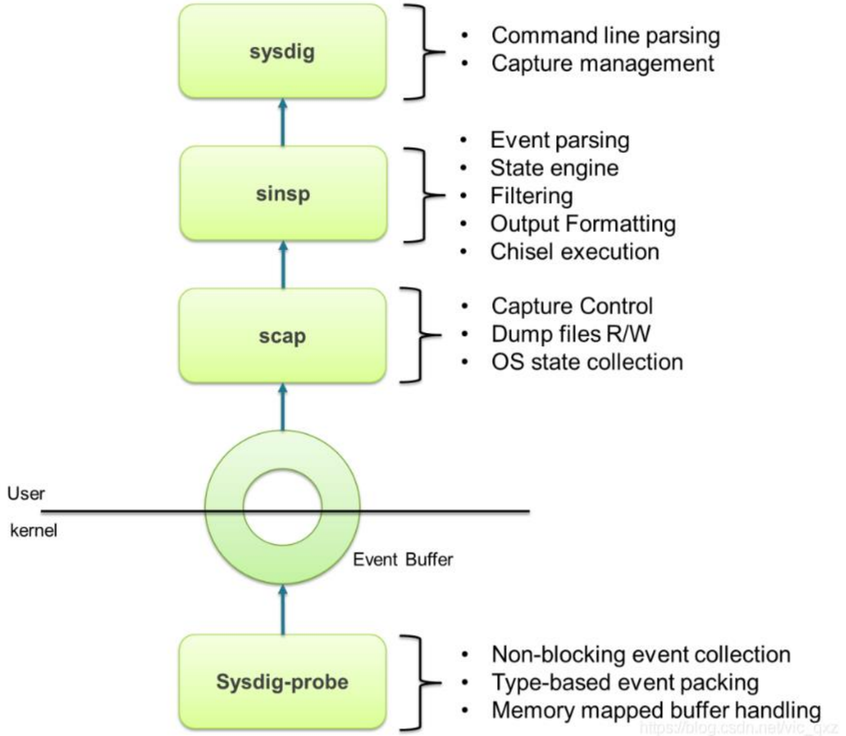
sysdig 除了能获取系统资源利用率、进程、网络连接、系统调用等信息，
还具备了很强的分析能力，例如：

- 按照CPU使用率对进程排序
- 按照数据包对进程排序
- 打开最多的文件描述符进程
- 查看进程打开了哪些文件
- 查看进程的HTTP请求报文
- 查看机器上容器列表及资源使用情况

项目地址：https://github.com/draios/sysdig
文档：https://github.com/draios/sysdig/wiki

分析容器系统调用：Sysdig

sysdig 通过在内核的驱动模块注册系统调用的 hook，这样当有系统调用发生和完成的时候，它会把系统调用信息拷贝到特定的 buffer，然后用户态组件对数据信息处理（解压、解析、过滤等），并最终通过 sysdig 命令行和用户进行交互。



分析容器系统调用：Sysdig

安装sysdig：

```
rpm --import https://s3.amazonaws.com/download.draios.com/DRAIOS-GPG-KEY.public
curl -s -o /etc/yum.repos.d/draios.repo https://s3.amazonaws.com/download.draios.com/stable/rpm/draios.repo
yum install epel-release -y
yum install sysdig -y
/usr/bin/sysdig-probe-loader # 加载驱动模块
```

分析容器系统调用：Sysdig

sysdig常用参数：

- -l, --list：列出可用于过滤和输出的字段
- -M <num_seconds> ： 多少秒后停止收集
- -p <output_format>, --print=<output_format> ： 指定打印事件时使用的格式
  - 使用-pc或-pcontainer 容器友好的格式
  - 使用-pk或-pkubernetes k8s友好的格式
- -c <chiselname> <chiselargs>：指定内置工具，可直接完成具体的数据聚合、分析工作
- -w <filename>：保存到文件中
- -r <filename>：从文件中读取

分析容器系统调用：Sysdig

执行sysdig命令，实时输出大量系统调用。

示例：59509 23:59:19.023099531 0 kubelet (1738) < epoll_ctl

格式：%evt.num %evt.outputtime %evt.cpu %proc.name (%thread.tid) %evt.dir %evt.type %evt.info

- evt.num： 递增的事件号
- evt.time： 事件发生的时间
- evt.cpu： 事件被捕获时所在的 CPU，也就是系统调用是在哪个 CPU 执行的
- proc.name： 生成事件的进程名字
- thread.tid： 线程的 id，如果是单线程的程序，这也是进程的 pid
- evt.dir： 事件的方向（direction），> 代表进入事件，< 代表退出事件
- evt.type： 事件的名称，比如 open、stat等，一般是系统调用
- evt.args： 事件的参数。如果是系统调用，这些对应着系统调用的参数

自定义格式输出：sysdig -p "user:%user.name time:%evt.time proc_name:%proc.name"

分析容器系统调用：Sysdig

sysdig过滤：

- fd：根据文件描述符过滤，比如 fd 标号（fd.num）、fd 名字（fd.name）
- process：根据进程信息过滤，比如进程 id（proc.id）、进程名（proc.name）
- evt：根据事件信息过滤，比如事件编号、事件名
- user：根据用户信息过滤，比如用户 id、用户名、用户 home 目录
- syslog：根据系统日志过滤，比如日志的严重程度、日志的内容
- container：根据容器信息过滤，比如容器ID、容器名称、容器镜像

查看完整过滤器列表：sysdig -l

示例：

1、查看一个进程的系统调用

sysdig proc.name=kubelet

2、查看建立TCP连接的事件

sysdig evt.type=accept

3、查看/etc目录下打开的文件描述符

sysdig fd.name contains /etc

4、查看容器的系统调用

sysdig -M 10 container.name=web

注：还支持运算操作符，=、!=、>=、>、<、<=、contains、in 、exists、and、or、not

分析容器系统调用：Sysdig

Chisels：实用的工具箱，一组预定义的功能集合，用来分析特定的场景。

sysdig –cl 列出所有Chisels，以下是一些常用的：

- topprocs_cpu：输出按照 CPU 使用率排序的进程列表，例如sysdig -c
- topprocs_net：输出进程使用网络TOP
- topprocs_file：进程读写磁盘文件TOP
- topfiles_bytes：读写磁盘文件TOP
- netstat：列出网络的连接情况

分析容器系统调用：Sysdig

网络	# 查看使用网络的进程TOP sysdig -c topprocs_net # 查看建立连接的端口 sysdig -c fdcount_by fd.sport "evt.type=accept" -M 10 # 查看建立连接的端口 sysdig -c fdbytes_by fd.sport # 查看建立连接的IP sysdig -c fdcount_by fd.cip "evt.type=accept" -M 10 # 查看建立连接的IP sysdig -c fdbytes_by fd.cip
硬盘	# 查看进程磁盘I/O读写 sysdig -c topprocs_file # 查看进程打开的文件描述符数量 sysdig -c fdcount_by proc.name "fd.type=file" -M 10 # 查看读写磁盘文件 sysdig -c topfiles_bytes sysdig -c topfiles_bytes proc.name=etcd # 查看/tmp目录读写磁盘活动文件 sysdig -c fdbytes_by fd.filename "fd.directory=/tmp/"
CPU	# 查看CPU使用率TOP sysdig -c topprocs_cpu # 查看容器CPU使用率TOP sysdig -pc -c topprocs_cpu container.name=web sysdig -pc -c topprocs_cpu container.id=web
容器	# 查看机器上容器列表及资源使用情况 csysdig –vcontainers # 查看容器资源使用TOP sysdig -c topcontainers_cpu/topcontainers_net/topcontainers_file

其他常用命令：

sysdig -c netstat

sysdig -c ps

sysdig -c lsof

监控容器运行时：Falco

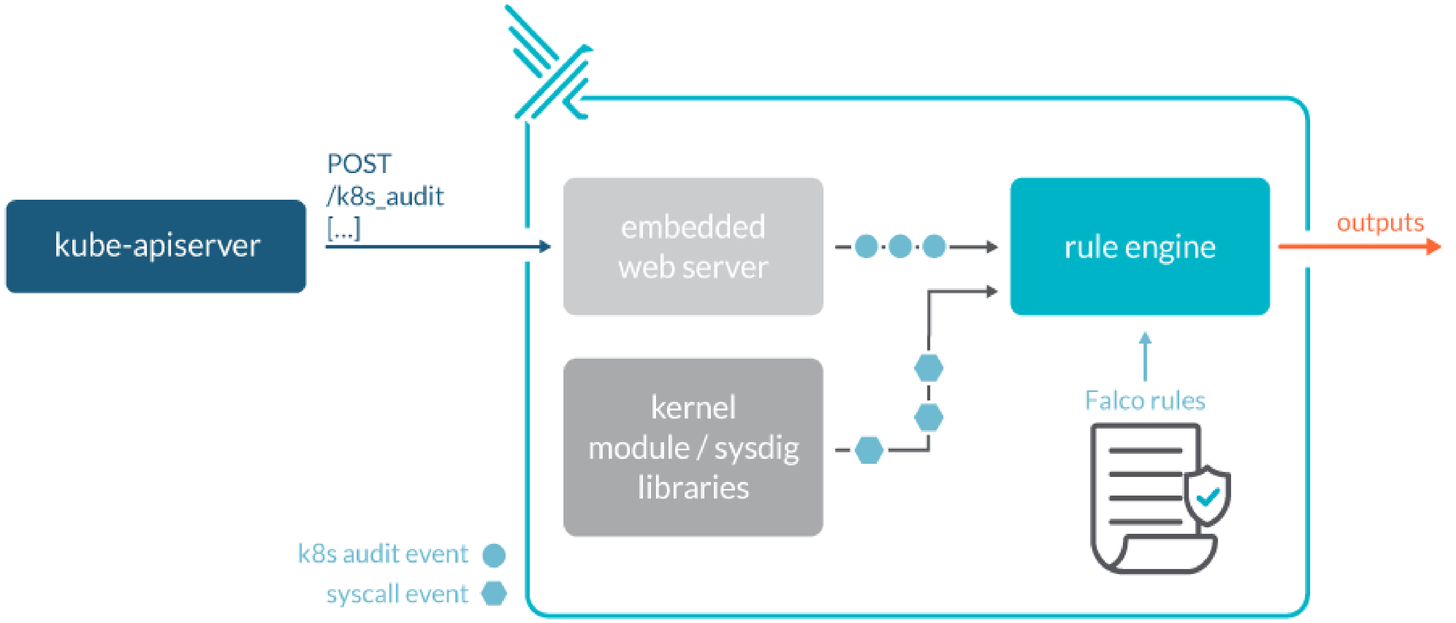
Falco 是一个 Linux 安全工具，它使用系统调用来保护和监控系统。
Falco最初是由Sysdig开发的，后来加入CNCF孵化器，成为首个加入CNCF的运行时安全项目。

Falco提供了一组默认规则，可以监控内核态的异常行为，例如：

- 对于系统目录/etc, /usr/bin, /usr/sbin的读写行为
- 文件所有权、访问权限的变更
- 从容器打开shell会话
- 容器生成新进程
- 特权容器启动

项目地址：https://github.com/falcosecurity/falco

监控容器运行时：Falco



Falco 架构

监控容器运行时：Falco

安装falco：

```
rpm --import https://falco.org/repo/falcosecurity-3672BA8F.asc
curl -s -o /etc/yum.repos.d/falcosecurity.repo https://falco.org/repo/falcosecurity-rpm.repo
yum install epel-release -y
yum update
yum install falco -y
systemctl start falco
systemctl enable falco
```

falco配置文件目录：/etc/falco

- falco.yaml falco配置与输出告警通知方式
- falco_rules.yaml 规则文件，默认已经定义很多威胁场景
- falco_rules.local.yaml 自定义扩展规则文件
- k8s_audit_rules.yaml K8s审计日志规则

安装文档：https://falco.org/zh/docs/installation

监控容器运行时：Falco

告警规则示例（falco_rules.local.yaml）：

```
 - rule: The program "sudo" is run in a container
   desc: An event will trigger every time you run sudo in a container
   condition: evt.type = execve and evt.dir=< and container.id != host and proc.name = sudo
   output: "Sudo run in container (user=%user.name %container.info parent=%proc.pname
cmdline=%proc.cmdline)"
   priority: ERROR
   tags: [users, container]
```

参数说明：

- rule：规则名称，唯一
- desc：规则的描述
- condition： 条件表达式
- output：符合条件事件的输出格式
- priority：告警的优先级
- tags：本条规则的 tags 分类

监控容器运行时：Falco

威胁场景测试：

1、监控系统二进制文件目录读写（默认规则）

2、监控根目录或者/root目录写入文件（默认规则）

3、监控运行交互式Shell的容器（默认规则）

4、监控容器创建的不可信任进程（自定义规则）

验证：tail -f /var/log/messages（告警通知默认输出到标准输出和系统日志）

监控容器运行时：Falco

监控容器创建的不可信任进程规则，在falco_rules.local.yaml文件添加：

```
- rule: Unauthorized process on nginx containers
  condition: spawned_process and container and container.image startswith nginx and not proc.name in (nginx)
  desc: test
  output: "Unauthorized process on nginx containers (user=%user.name container_name=%container.name
container_id=%container.id image=%container.image.repository shell=%proc.name parent=%proc.pname
cmdline=%proc.cmdline terminal=%proc.tty)"
  priority: WARNING
```

condition表达式解读：

- spawned_process 运行新进程
- container 容器
- container.image startswith nginx 以nginx开头的容器镜像
- not proc.name in (nginx) 不属于nginx的进程名称（允许进程名称列表）

重启falco应用新配置文件：

systemctl restart falco

监控容器运行时：Falco

Falco支持五种输出告警通知的方式：

- 输出到标准输出（默认启用）
- 输出到文件
- 输出到Syslog（默认启用）
- 输出到HTTP服务
- 输出到其他程序（命令行管道方式）

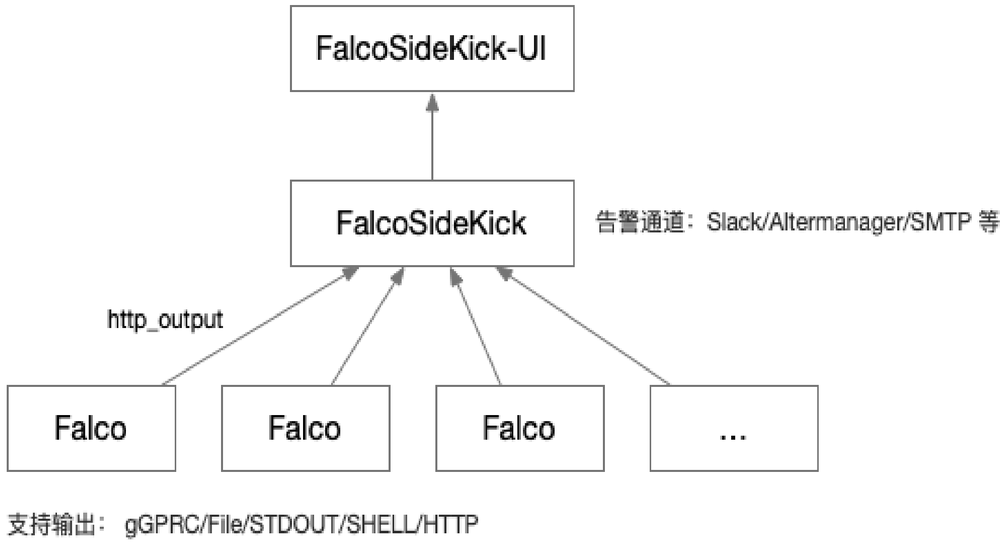
告警配置文件：/etc/falco/falco.yaml

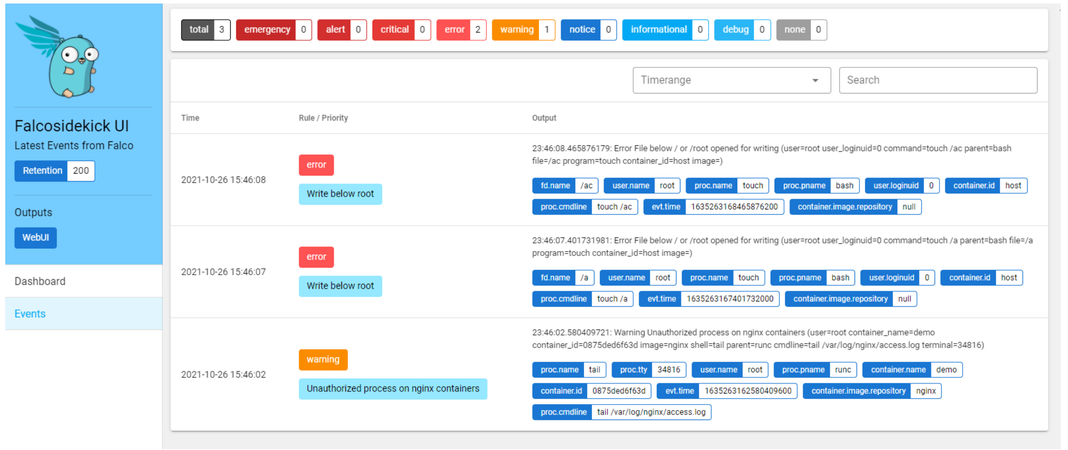
例如输出到指定文件：

```
file_output:
  enabled: true
  keep_alive: false
  filename: /var/log/falco_events.log
```

监控容器运行时：Falco

Falco告警集中化展示：





- FalcoSideKick：一个集中收集并指定输出，支持大量方式输出，例如Influxdb、Elasticsearch等

项目地址 https://github.com/falcosecurity/falcosidekick

- FalcoSideKick-UI：告警通知集中图形展示系统

项目地址 https://github.com/falcosecurity/falcosidekick-ui

监控容器运行时：Falco

部署Falco UI：

```
docker run -d \
-p 2801:2801 \
--name falcosidekick \
-e WEBUI_URL=http://192.168.31.71:2802 \
falcosecurity/falcosidekick

docker run -d \
-p 2802:2802 \
--name falcosidekick-ui \
falcosecurity/falcosidekick-ui
```

修改falco配置文件指定http方式输出：

```
json_output: true
json_include_output_property: true
http_output:
  enabled: true
  url: "http://192.168.31.71:2801/"
```

UI访问地址：http://192.168.31.71:2802/ui/

Kubernetes 审计日志

在Kubernetes集群中，API Server的审计日志记录了哪些用户、哪些服务请求操作集群资源，并且可以编写不同规则，控制忽略、存储的操作日志。
审计日志采用JSON格式输出，每条日志都包含丰富的元数据，例如请求的URL、HTTP方法、客户端来源等，你可以使用监控服务来分析API流量，以检测趋势或可能存在的安全隐患。

这些可能服务会访问API Server：

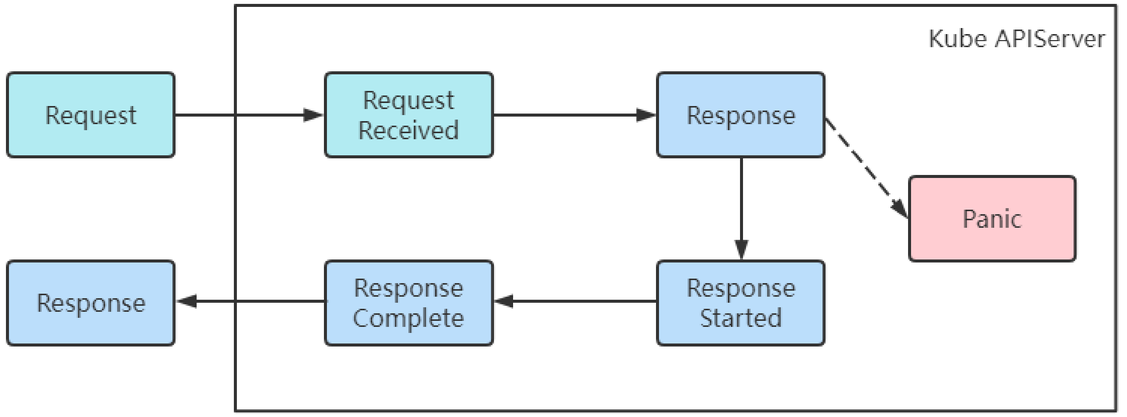
- 管理节点（controller-manager、scheduler）
- 工作节点（kubelet、kube-proxy）
- 集群服务（CoreDNS、Calico、HPA等）
- kubectl、API、Dashboard

Kubernetes 审计日志

事件和阶段：

当客户端向 API Server发出请求时，该请求将经历一个或多个阶段：

阶段	说明
RequestReceived	审核处理程序已收到请求
ResponseStarted	已发送响应标头，但尚未发送响应正文
ResponseComplete	响应正文已完成，不再发送任何字节
Panic	内部服务器出错，请求未完成



ApiServer处理请求流程图

Kubernetes 审计日志

Kubernetes审核策略文件包含一系列规则，描述了记录日志的级别，
采集哪些日志，不采集哪些日志。
规则级别如下表所示：

级别	说明
None	不为事件创建日志条目
Metadata	创建日志条目。包括元数据，但不包括请求正文或响应正文
Request	创建日志条目。包括元数据和请求正文，但不包括响应正文
RequestResponse	创建日志条目。包括元数据、请求正文和响应正文

```yaml
apiVersion: audit.k8s.io/v1
kind: Policy
rules:
  # RequestResponse级别记录pod操作
  - level: RequestResponse
    resources:
    - group: ""
      resources: ["pods"]
  # 不记录日志
  - level: None
    # system:kube-proxy用户请求不记录
    users: ["system:kube-proxy"]
    # watch请求不记录
    verbs: ["watch"]
    # endpoints、services资源不记录
    resources:
    - group: "" # core API group
      resources: ["endpoints", "services"]
```

示例

参考资料：https://kubernetes.io/zh/docs/tasks/debug-application-cluster/audit/

Kubernetes 审计日志

日志格式示例：

审计日志属性

用户信息

客户端IP

用户标识
对象

响应状态

时间
注释


```
{
  "kind": "Event",
  "apiVersion": "audit.k8s.io/v1",
  "level": "Metadata",
  "auditID": "4becec1a-0d3d-4304-904d-70c695c3d938",
  "stage": "ResponseComplete",
  "requestURI": "/apis/apps/v1/namespaces/default/deployments?fieldManager=kubectl-create",
  "verb": "create",
  "user": {
    "username": "kubernetes-admin",
    "groups": [
      "system:masters",
      "system:authenticated"
    ]
  },
  "sourceIPs": [
    "192.168.31.71"
  ],
  "userAgent": "kubectl/v1.21.0 (linux/amd64) kubernetes/cb303e6",
  "objectRef": {
    "resource": "deployments",
    "namespace": "default",
    "name": "web-demo",
    "apiGroup": "apps",
    "apiVersion": "v1"
  },
  "responseStatus": {
    "metadata": {},
    "code": 201
  },
  "requestReceivedTimestamp": "2021-11-08T16:16:53.551850Z",
  "stageTimestamp": "2021-11-08T16:16:53.563907Z",
  "annotations": {
    "authorization.k8s.io/decision": "allow",
    "authorization.k8s.io/reason": ""
  }
}
```

Kubernetes 审计日志

审计日志支持写入本地文件和Webhook（发送到外部HTTP API）两种方式。

启用审计日志功能：

```
vi /etc/kubernetes/manifests/kube-apiserver.yaml
...
    - --audit-policy-file=/etc/kubernetes/audit/audit-policy.yaml
    - --audit-log-path=/var/log/k8s_audit.log
    - --audit-log-maxage=30
    - --audit-log-maxbackup=10
    - --audit-log-maxsize=100
...
 volumeMounts:
    ...
 - mountPath: /etc/kubernetes/audit/audit-policy.yaml
      name: audit
    - mountPath: /var/log/k8s_audit.log
      name: audit-log
  volumes:
  - name: audit
    hostPath:
      path: /etc/kubernetes/audit/audit-policy.yaml
      type: File
  - name: audit-log
    hostPath:
      path: /var/log/k8s_audit.log
      type: FileOrCreate
```

audit-policy-file	审计日志策略文件
audit-log-path	审计日志输出文件
audit-log-maxage	审计日志保留的最大天数
audit-log-maxbackup	审计日志最大分片存储多少个日志文件
audit-log-maxsize	单个审计日志最大大小，单位MB

注：需要使用hostpath数据卷将宿主机策略文件和日志文件挂载到容器中

Kubernetes 审计日志

示例：只记录指定资源操作日志

```
apiVersion: audit.k8s.io/v1
kind: Policy
# 忽略步骤，不为RequestReceived阶段生成审计日志
omitStages:
  - "RequestReceived"
rules:
  # 不记录日志
  - level: None
    users:
      - system:apiserver
      - system:kube-controller-manager
      - system:kube-scheduler
      - system:kube-proxy
      - kubelet
  # 针对资源记录日志
  - level: Metadata
    resources:
    - group: ""
      resources: ["pods"]
  #   - group: "apps"
  #     resources: ["deployments"]
  # 其他资源不记录日志
  - level: None
```

Kubernetes 审计日志

收集审计日志方案：

- 审计日志文件+filebeat
- 审计webhook+logstash
- 审计webhook+falco

谢谢

阿良个人微信

DevOps技术栈公众号

阿良教育：www.aliangedu.cn