讲师：李振良（阿良）

今天课题：《K8s集群部署与安全配置》

学院官网：www.aliangedu.cn

阿良个人微信

DevOps技术栈公众号

第一章 K8s集群部署与安全配置

- K8s安全运维概述
- 部署一套完整的K8s高可用集群
- CIS 安全基准介绍与K8s安全基准工具kube-bench
- Ingress 配置证书
- 网络策略控制集群内部网络通信

K8s安全运维概述

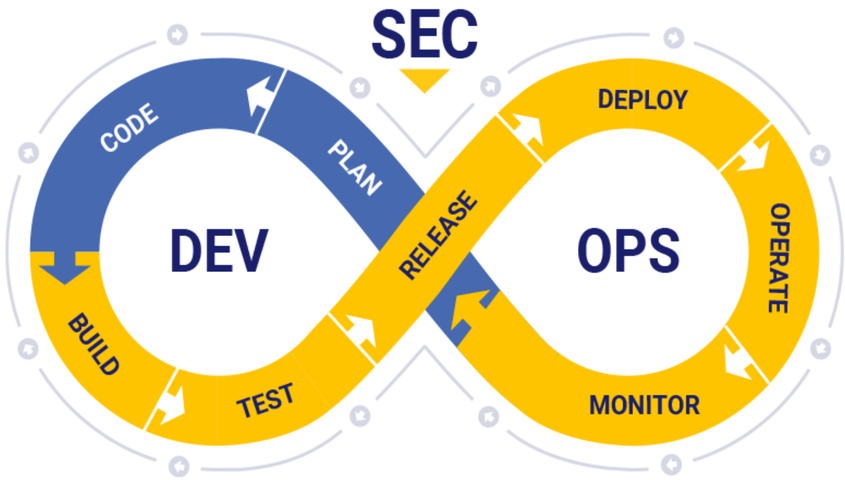
- 安全运维的重要性
- SecDevOps
- K8s提供的安全机制
- K8s安全运维实践思路

安全运维的重要性

- 万物互联，安全为基，企业的网络安全不可小视
- 服务器被黑事件频发
- 公司重要数据资产在运维手中

SecDevOps

SecDevOps与DevOps相似，是一种哲学，鼓励运维人员、开发人员、测试人员和安全人员进行更高水水平协作，将信息安全被放在事前考虑，将安全性注入自动化流程中，以确保整个产品周期内的信息安全。

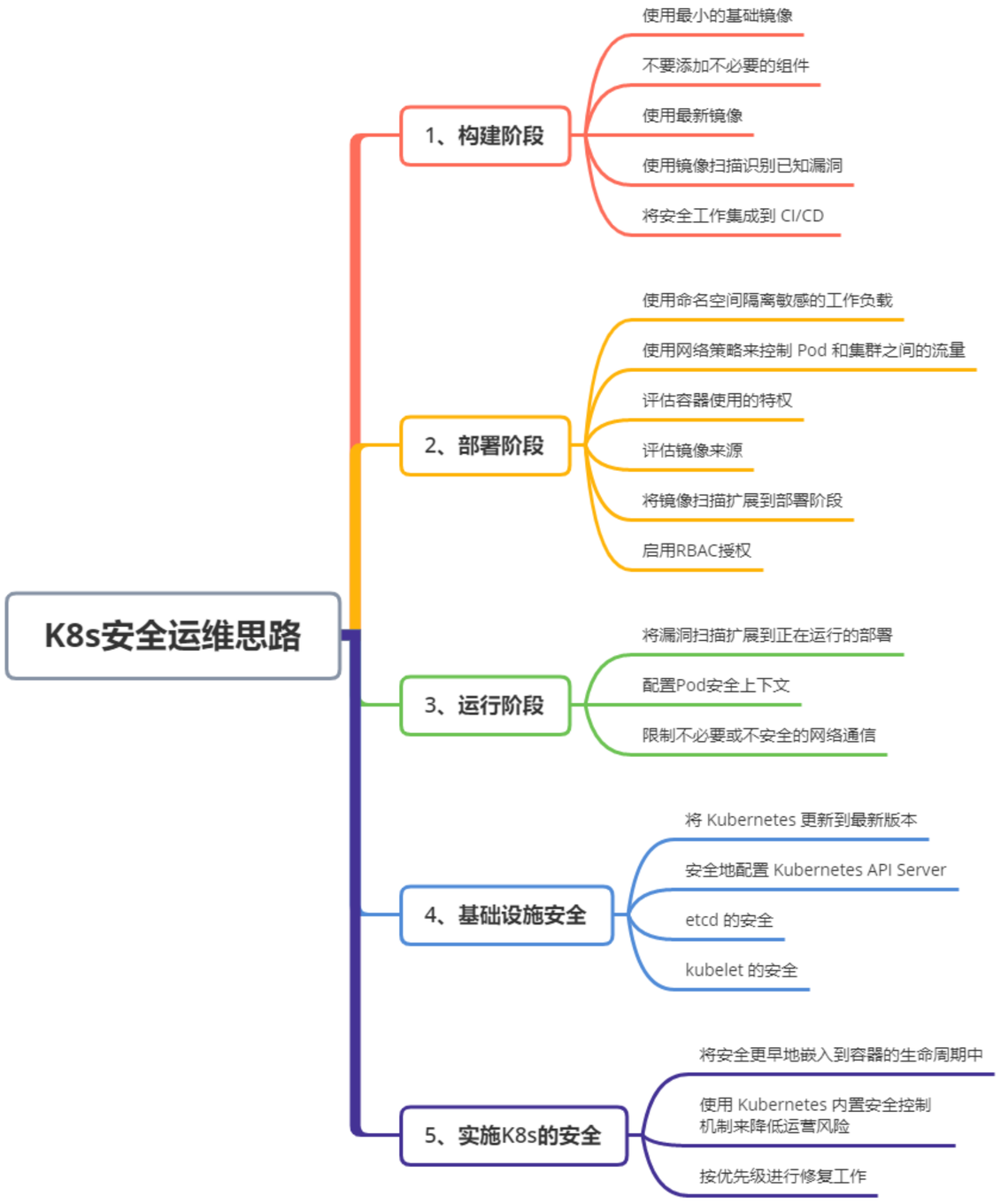


K8s提供的安全机制

为了保证集群以及容器应用的安全，Kubernetes 提供了多种安全机制，限制容器的行为，减少容器和集群的攻击面，保证整个系统的安全性。

- **集群安全：**TLS证书认证、RBAC
- **Security Context：**限制容器的行为，例如只读文件系统、特权、运行用户等
- **Pod Security Policy：**集群级的 Pod 安全策略，自动为集群内的 Pod 配置安全策略
- **Sysctls：**允许容器设置内核参数
- **AppArmor：**限制容器中应用对资源的访问权限
- **Network Policies：**控制集群中网络通信
- **Seccomp：**限制容器内进程的系统调用

K8s安全运维实践思路



部署一套完整的K8s高可用集群

CIS 安全基准

- CIS安全基准介绍
- K8s安全基准工具 kube-bench

CIS安全基准

互联网安全中心（CIS，Center for Internet Security），是一个非盈利组织，致力为互联网提供免费的安全防御解决方案。

官网：https://www.cisecurity.org/
Kubernetes CIS基准：https://www.cisecurity.org/benchmark/kubernetes/

CIS基准测试工具

下载pdf后，根据里面的基准来检查K8s集群配置，但内容量太大，一般会采用相关工具来完成这项工作。

Kube-bench是容器安全厂商Aquq推出的工具，以CIS K8s基准作为基础，来检查K8s是否安全部署。
主要查找不安全的配置参数、敏感的文件权限、不安全的帐户或公开端口等等。

项目地址：https://github.com/aquasecurity/kube-bench

CIS基准测试工具：kube-beach部署

1、下载二进制包

https://github.com/aquasecurity/kube-bench/releases

2、解压使用

```
tar zxvf kube-bench_0.6.3_linux_amd64.tar.gz
mkdir /etc/kube-bench   # 创建默认配置文件路径
mv cfg /etc/kube-bench/cfg
```

CIS基准测试工具：kube-beach使用

使用kube-bench run进行测试，该指令有以下常用参数：

常用参数：

- -s, --targets 指定要基础测试的目标，这个目标需要匹配cfg/<version>中的文件名称，已有目标：master, controlplane, node, etcd, policies
- --version：指定k8s版本，如果未指定会自动检测
- --benchmark：手动指定CIS基准版本，不能与--version一起使用

Source	Kubernetes Benchmark	kube-bench config	Kubernetes versions
CIS	1.5.1	cis-1.5	1.15-
CIS	1.6.0	cis-1.6	1.16-
CIS	GKE 1.0.0	gke-1.0	GKE
CIS	EKS 1.0.0	eks-1.0	EKS
CIS	ACK 1.0.0	ack-1.0	ACK
RHEL	RedHat OpenShift hardening guide	rh-0.7	OCP 3.10-3.11
CIS	OCP4 1.1.0	rh-1.0	OCP 4.1-

Kube-bench与k8s版本支持

CIS基准测试工具：kube-beach使用

例如：检查master组件安全配置

```
kube-bench run --targets=master
```

执行后会逐个检查安全配置并输出修复方案及汇总信息输出：

```
== Summary master ==
44 checks PASS
10 checks FAIL
11 checks WARN
0 checks INFO

== Summary total ==
44 checks PASS
10 checks FAIL
11 checks WARN
0 checks INFO
```

[PASS]：测试通过

[FAIL]：测试未通过，重点关注，在测试结果会给出修复建议

[WARN]：警告，可做了解

[INFO]：信息

CIS基准测试工具：kube-beach使用

测试项目配置文件：/etc/kube-bench/cfg/cis-1.6/

```
- id: 1.2.21
  text: "Ensure that the --profiling argument is set to false (Automated)"
  audit: "/bin/ps -ef | grep $apiserverbin | grep -v grep"
  tests:
    test_items:
      - flag: "--profiling"
        compare:
          op: eq
          value: false
  remediation: |
    Edit the API server pod specification file $apiserverconf
    on the master node and set the below parameter.
    --profiling=false
  scored: true
```

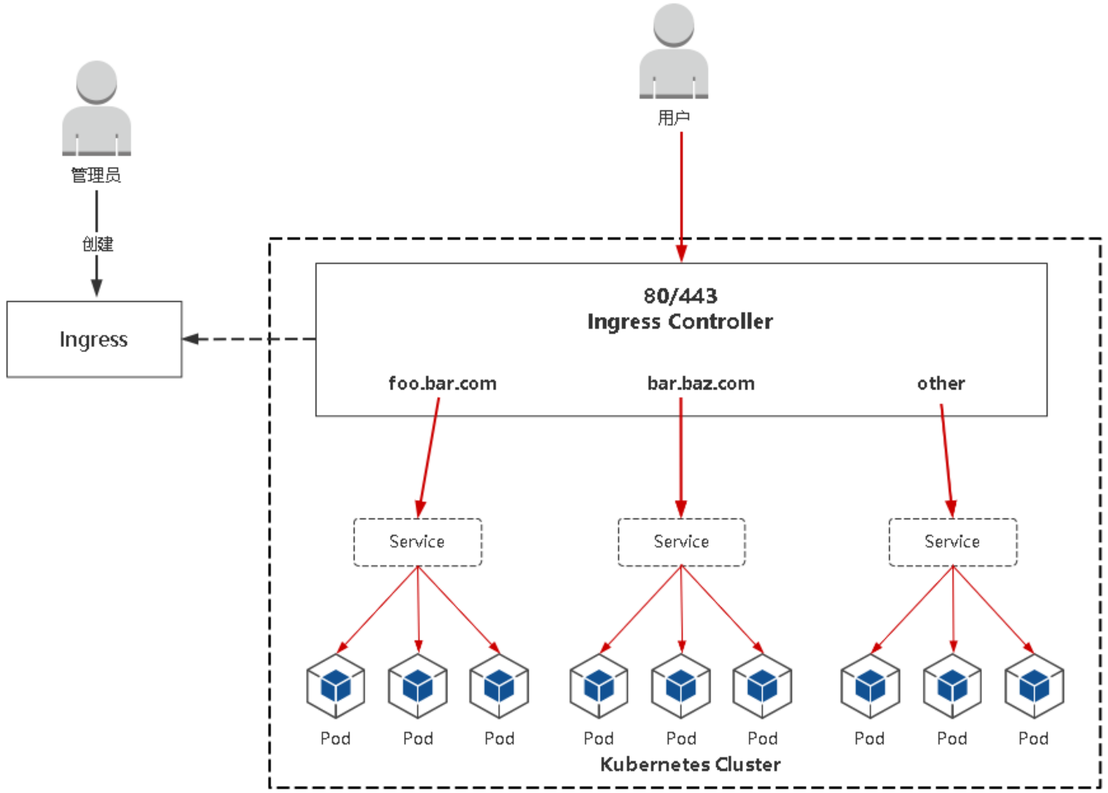
- id：编号
- text：提示的文本
- audit：
- tests：测试项目
- remediation：修复方案
- scored：如果为true，kube-bench无法正常测试，则会生成FAIL，如果为false，无法正常测试，则会生成WARN。
- type：如果为manual则会生成WARN，如果为skip，则会生成INFO

Ingress配置证书

- Ingress是什么
- HTTPS重要性
- 将一个项目对外暴露HTTPS访问

Ingress是什么

- **Ingress**：K8s中的一个抽象资源，给管理员提供一个暴露应用的入口定义方法
- **Ingress Controller**：根据Ingress生成具体的路由规则，并对Pod负载均衡器



HTTPS重要性

HTTPS是安全的HTTP，HTTP 协议中的内容都是明文传输，HTTPS 的目的是将这些内容加密，确保信息传输安全。最后一个字母 S 指的是 SSL/TLS 协议，它位于 HTTP 协议与 TCP/IP 协议中间。

HTTPS优势：

1、加密隐私数据：防止您访客的隐私信息(账号、地址、手机号等)被劫持或窃取。

2、安全身份认证：验证网站的真实性，防止钓鱼网站。

3、防止网页篡改：防止数据在传输过程中被篡改，保护用户体验。

4、地址栏安全锁：地址栏头部的“锁”型图标，提高用户信任度。

5、提高SEO排名：提高搜索排名顺序，为企业带来更多访问量。

将一个项目对外暴露HTTPS访问

配置HTTPS步骤：

1、准备域名证书文件（来自：openssl/cfssl工具自签或者权威机构颁发）

2、将证书文件保存到Secret

kubectl create secret tls web-aliangedu-cn --

cert=web.aliangedu.cn.pem --key=web.aliangedu.cn-key.pem

3、Ingress规则配置tls

4、kubectl get ingress

5、测试，本地电脑绑定hosts记录对应ingress里面配置的域名，IP是

Ingress Concontroller Pod节点IP

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: aliangedu-https
spec:
  tls:
  - hosts:
      - web.aliangedu.cn
    secretName: web-aliangedu-cn
  rules:
  - host: web.aliangedu.cn
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: web
            port:
              number: 80
```

网络访问控制

- 网络策略应用场景
- 网络策略概述
- 网络访问控制5个案例

网络访问控制：应用场景

默认情况下，Kubernetes 集群网络没任何网络限制，Pod 可以与任何其他 Pod 通信，在某些场景下就需要进行网络控制，减少网络攻击面，提高安全性，这就会用到网络策略。

网络策略（Network Policy）：是一个K8s资源，用于限制Pod出入流量，提供Pod级别和Namespace级别网络访问控制。

网络策略的应用场景：

- 应用程序间的访问控制，例如项目A不能访问项目B的Pod
- 开发环境命名空间不能访问测试环境命名空间Pod
- 当Pod暴露到外部时，需要做Pod白名单
- 多租户网络环境隔离

网络访问控制：网络策略概述

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - ipBlock:
        cidr: 172.17.0.0/16
        except:
        - 172.17.1.0/24
    - namespaceSelector:
        matchLabels:
          project: myproject
    - podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: TCP
      port: 6379
  egress:
  - to:
    - ipBlock:
        cidr: 10.0.0.0/24
    ports:
    - protocol: TCP
      port: 5978
```

podSelector：目标Pod，根据标签选择。

policyTypes：策略类型，指定策略用于入站、出站流量。

Ingress：from是可以访问的白名单，可以来自于IP段、命名空间、Pod标签等，ports是可以访问的端口。

Egress：这个Pod组可以访问外部的IP段和端口。

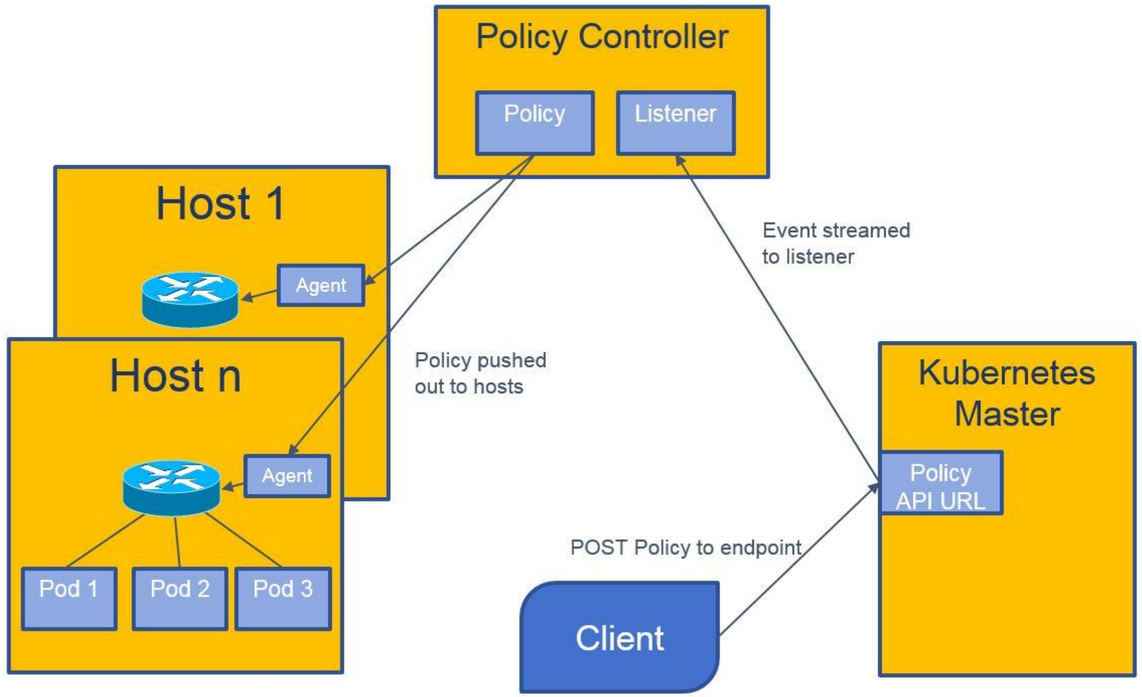
网络访问控制：网络策略概述

网络策略工作流程：

1、创建Network Policy资源

2、Policy Controller监控网络策略，同步并通知节点上程序

3、节点上DaemonSet运行的程序从etcd中获取Policy，调用本地Iptables创建防火墙规则



Network Policy工作流程

网络访问控制案例

案例1：拒绝命名空间下所有Pod出入站流量

案例2：拒绝其他命名空间Pod访问

案例3：允许其他命名空间Pod访问指定应用

案例4：同一个命名空间下应用之间限制访问

案例5：只允许指定命名空间中的应用访问

案例1：拒绝命名空间下所有Pod入、出站流量

需求：拒绝test命名空间下所有Pod入、出站流量

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: deny-all
  namespace: test
spec:
  podSelector: {}  # 匹配本命名空间所有pod
  policyTypes:
  - Ingress
  - Egress
  # ingress和egress没有指定规则，则不允许任何流量进出pod
```

案例2：拒绝其他命名空间Pod访问

需求：test命名空间下所有pod可以互相访问，也可以访问其他命名空间Pod，但其他命名空间不能访问test命名空间Pod。

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: deny-all-namespaces
  namespace: test
spec:
  podSelector: {}
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector: {}   #  匹配本命名空间所有pod
```

案例3：允许其他命名空间Pod访问指定应用

需求：允许其他命名空间访问test命名空间指定Pod

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-all-namespaces
  namespace: test
spec:
  podSelector:
    matchLabels:
      app: web
  policyTypes:
  - Ingress
  ingress:
  - from:
    - namespaceSelector: {} # 匹配所有命名空间的pod
```

案例4：同一个命名空间下应用之间限制访问

需求：将test命名空间中标签为run=web的pod隔离，

只允许标签为run=client1的pod访问80端口

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: app-to-app
  namespace: test
spec:
  podSelector:
    matchLabels:
      run: web
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          run: client1
    ports:
    - protocol: TCP
      port: 80
```

案例5：只允许指定命名空间中的应用访问

需求：限制dev命名空间标签为env=dev的pod，只允许prod命名空间中的pod访问和其他所有命名空间app=client1标签pod访问

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: dev-web
  namespace: dev
spec:
  podSelector:
    matchLabels:
      env: dev
  policyTypes:
  - Ingress
  ingress:
  # 满足允许prod命名空间中的pod访问
  - from:
    - namespaceSelector:
        matchLabels:
          env: prod
  # 允许pod标签为app=client1的pod访问，所有命名空间
  - from:
    - namespaceSelector: {}
      podSelector:
        matchLabels:
          app: client1
```

课后作业

1、网络策略

需求1：在test命名空间创建一个名为deny-all的网络策略，拒绝本命名空间所有Pod的Ingress和Egress流量

需求2：限制dev命名空间标签为env=dev的pod，只允许prod命名空间中的pod访问和其他所有命名空间app=client1标签pod访问

2、使用kube-bench工具检查集群组件配置文件存在的问题与修复，并重启对应组件确保新配置生效。

修复：1.2.21 Ensure that the --profiling argument is set to false (Automated)

谢谢

阿良个人微信

DevOps技术栈公众号

阿良教育：www.aliangedu.cn