Amazon EKS

你的 EKS 安全嗎？
由安全性角度切入
探討 Kubernetes on AWS

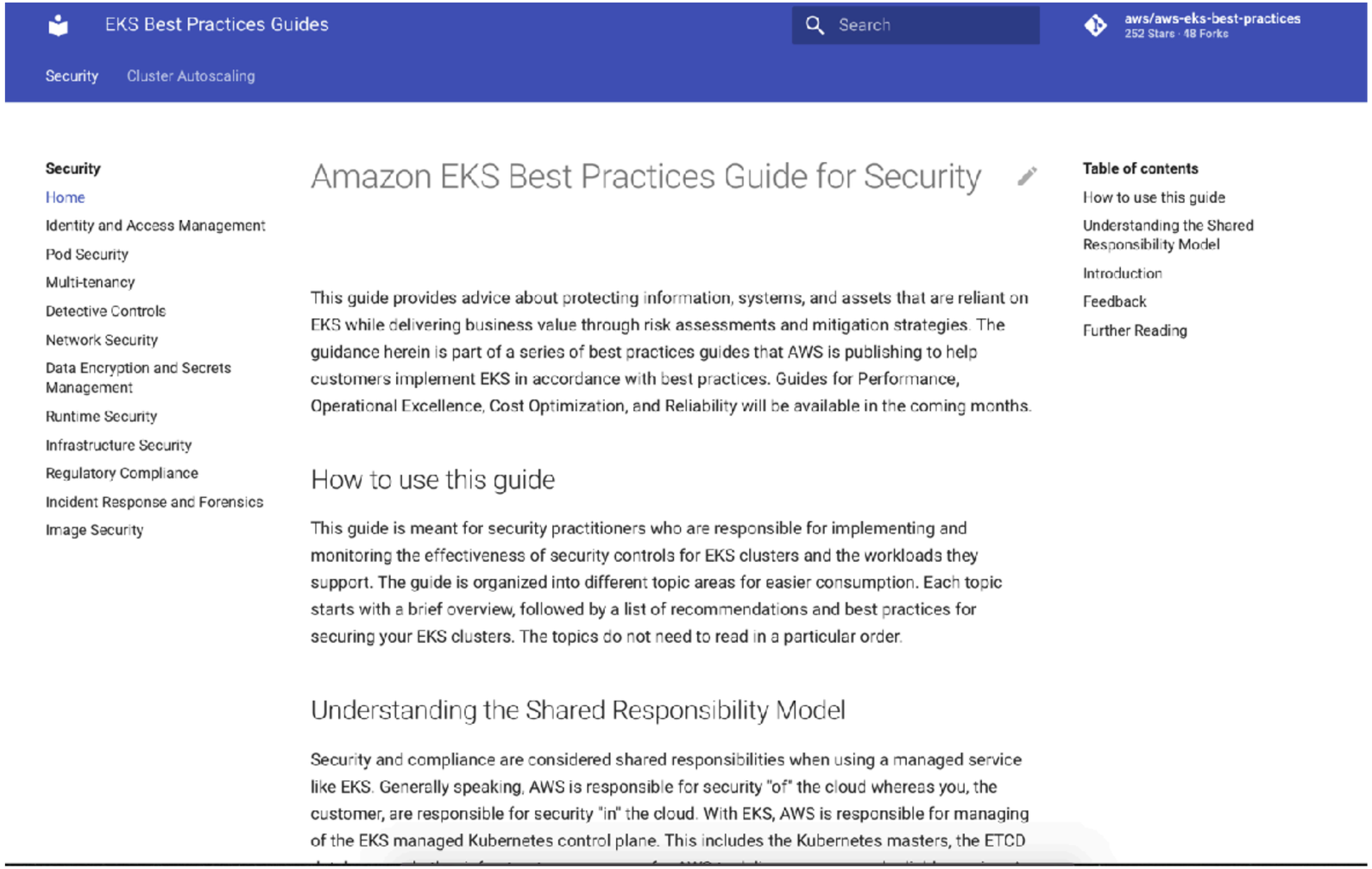
Is your EKS secured?

WHOAMI

David Hsu 徐維澤 @davidh83110

4+ years DevOps
AWS Certified Solutions Architect Associate

Source



https://aws.github.io/aws-eks-best-practices

01 IAM 身份驗證與授權

IAM Management

02 Pod 容器安全

Pod Security

03 多租戶隔離

Multi-Tenancy Management

04 網路與運行期間

Network and Runtime

05 節點

Infrastructure (Host) Management

06 監控與稽核

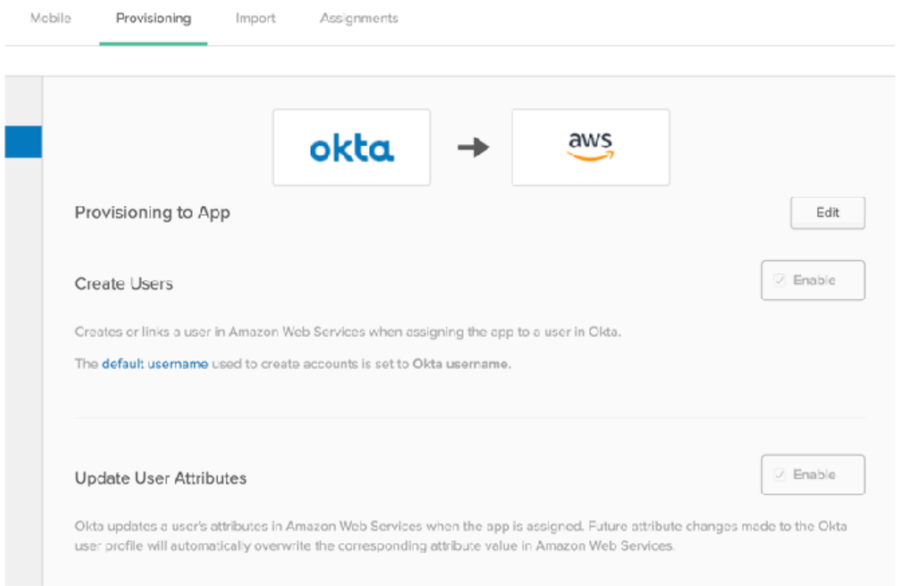
Monitoring and Auditing

When You Are Provisioning a Cluster

- 設定 Private Endpoint EKS Cluster
  - 會需要 VPN / Bastion Instance
  - Public Endpoint 必須使用白名單限制

- AWS-Auth 使用 IAM Role 取代 IAM User
  - DevOpsRole / BackendRole / QARole
  - 集中在 IAM 做權限中心化管理

- OAuth 結合 OIDC 去存取 EKS Cluster
  - *Okta*



IAM 身份驗證與授權

IAM Management

kubectl get pod → →

→ → Result

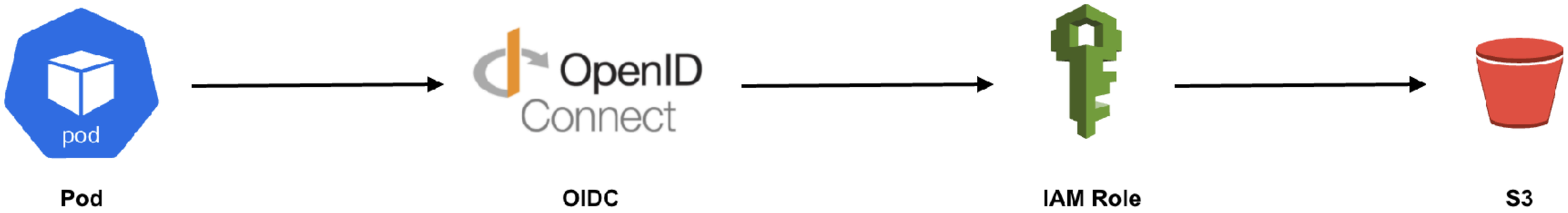
```
$ aws eks get-token --cluster <cluster_name>

{
  "kind": "ExecCredential",
  "apiVersion": "client.authentication.k8s.io/v1alpha1",
  "spec": {},
  "status": {
    "expirationTimestamp": "2020-02-19T16:08:27Z",
    "token": "k8s-aws-v1.aHR0cHM6Ly9zdHMuYW1hem9uYXdzLmNvbS8_QWN0aW9uPUdldENhbGxlck
  }
}
```

Token 每 15 分鐘過期

Service Account

- 避免使用 Service Account Token
  - Long-Lived 的 Token 風險較高
- 每個 Application 都要有專用的 Service Account
- 避免 ["*"] 在 RoleBinding 裡面

IAM Role for Service Account (IRSA)



IAM Role for Service Account (IRSA)

- 不使用 Instance Profile 去管理，使用 IRSA
  - 安全控制精細度提升到 Pod 等級
- OIDC Token 會透過 kubelet 自動替換
  - 經過 24 HR
- Application SDK 版本過舊會有不支援 IRSA 的問題

Pod 容器安全

Pod Security

Image Security

- 避免使用 root 運行 Container
  - 提權攻擊
  - Dockerfile 裡使用 non-root 的 user
  - PodSpec 裡的 "spec.securityContext" 也可以指定運行用戶 (runAsUser/runAsGroup)
- 製作自己的 Based Image
- 移除用不到的套件
  - 精簡 Image Size
- 不要放 Secret 在裡面
  - Hashicorp Vault
  - AWS Secrets Manager
  - AWS Parameter Store
- 掃描 Docker Image

Pod Security

- Never Docker in Docker
  - 會沒有辦法透過 Node 來管理第二層的容器
- 限制掛載 Host 的目錄 (hostPath)
  - 不易管理且有風險
  - PSP 設定 Read-Only

```
allowedHostPaths:
- pathPrefix: "/foo"
    readOnly: true # only allow read-only mounts
```

Pod Security Policy (PSP)

- PSP 是 Kubernetes 的 API 對象，用來控制 Pod 權限
  - EKS 預設是 eks.privileged 也就是有特權的
  - Privileged Pod 擁有 Node 上很大權限
  - 不取得不需要的權限

```
Name:  eks.privileged

Settings:
  Allow Privileged:                       true
  Allow Privilege Escalation:             0xc0004ce5f8
  Default Add Capabilities:               <none>
  Required Drop Capabilities:             <none>
  Allowed Capabilities:                   *
  Allowed Volume Types:                   *
  Allow Host Network:                     true
  Allow Host Ports:                       0-65535
  Allow Host PID:                         true
  Allow Host IPC:                         true
  Read Only Root Filesystem:              false
  SELinux Context Strategy: RunAsAny
    User:                                 <none>
    Role:                                 <none>
    Type:                                 <none>
    Level:                                <none>
  Run As User Strategy: RunAsAny
    Ranges:                               <none>
  FSGroup Strategy: RunAsAny
    Ranges:                               <none>
  Supplemental Groups Strategy: RunAsAny
    Ranges:                               <none>
```

Quality of Service (QoS)

request 和 *limit* 到底該不該設定？兩個都要設定還是擇一就好？

- 妥善的設定 request 和 limit
  - 避免 Single-Point Overload 演變成 DDos
  - 節點資源有效利用
  - 避免叢集裡不同服務相互影響
  - 在 Namespace 可以透過 Resource Quota / Limit Range 去強制
  - 不同重要性的服務給予不同等級的設定

Name	Priority	Condition	Result
Guaranteed	highest	limit = request ≠ 0 只設 limit, 那 request 會自動等於 limit	除非超過 memory limit 不然不會被殺
Burstable	medium	limit ≠ request ≠ 0	超過 request memory 後如果 Node 資源不夠就會被殺
Best Effort	lowest	limit & request = null	資源不夠最早被殺

多租戶隔離

Multi-Tenancy Management

Multi-Tenancy

What is tenant(租戶)？ 多租戶？ 單租戶？

Photo credit: https://www.maxpixel.net/

Soft Multi-Tenancy

- 邏輯層的隔離
  - Role-Based Access Control (RBAC)
  - Quotas
  - Limit Ranges
  - Different Namespaces

但是 Pod 還是仍然共享 Hosts !!!

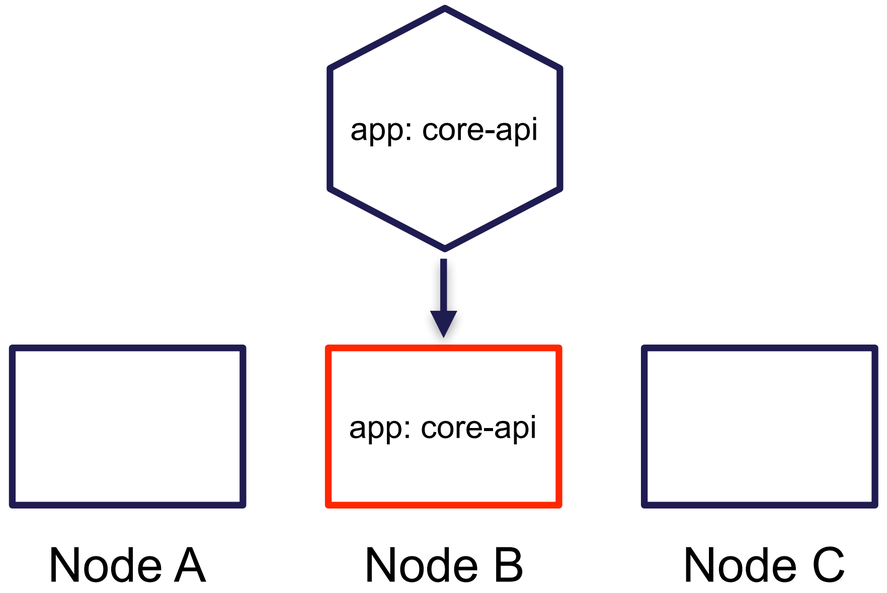
pod pod pod

Secret / Volume / ConfigMap

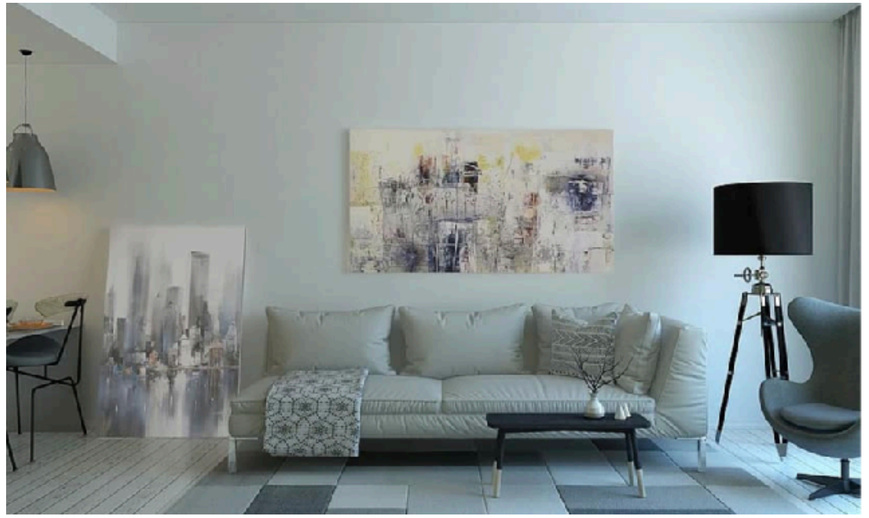
Consider to Use More Powerful Method

- Sole Tenant Nodes 唯一租戶節點
  - Node Selector
  - Anti-Affinity Rules
  - Taints / Tolerations

這些方法都可以讓我們限制 Pod 只調度到指定的節點



COVID-19
DO NOT VISIT
HOME UNDER QUARANTINE
From
To
Name
Address
No. of Persons
If this person is found outside the home within the period mention above, please contact
COVID 19 Distt West Help line
01125100093
01125100094
01125100096
01125100097
O/O DISTRICT MAGISTRATE, WEST
GOVT. OF NCT DELHI
3, SHIVAJI PLACE, RAJA GARDEN, NEW DELHI - 110027



Hard Multi-Tenancy

Hard Multi-Tenancy

- 資源利用率會變差
- 花費更高
- 人力管理成本也更高

Multi-Tenancy Management

DEV Environment

- 切分不同 Namespace 給不同 Team 做測試部署
- 便於管理，彼此也比較不受影響

Production Environment

- 為不同的 Service 以及 Application 切分不同的 Namespace
- 切分不同用途的 Nodes
- 要求隔離性的服務部署到不同 Cluster
- 設定好 PSP / Resource Quotas

網路與運行期間

Network and Runtime

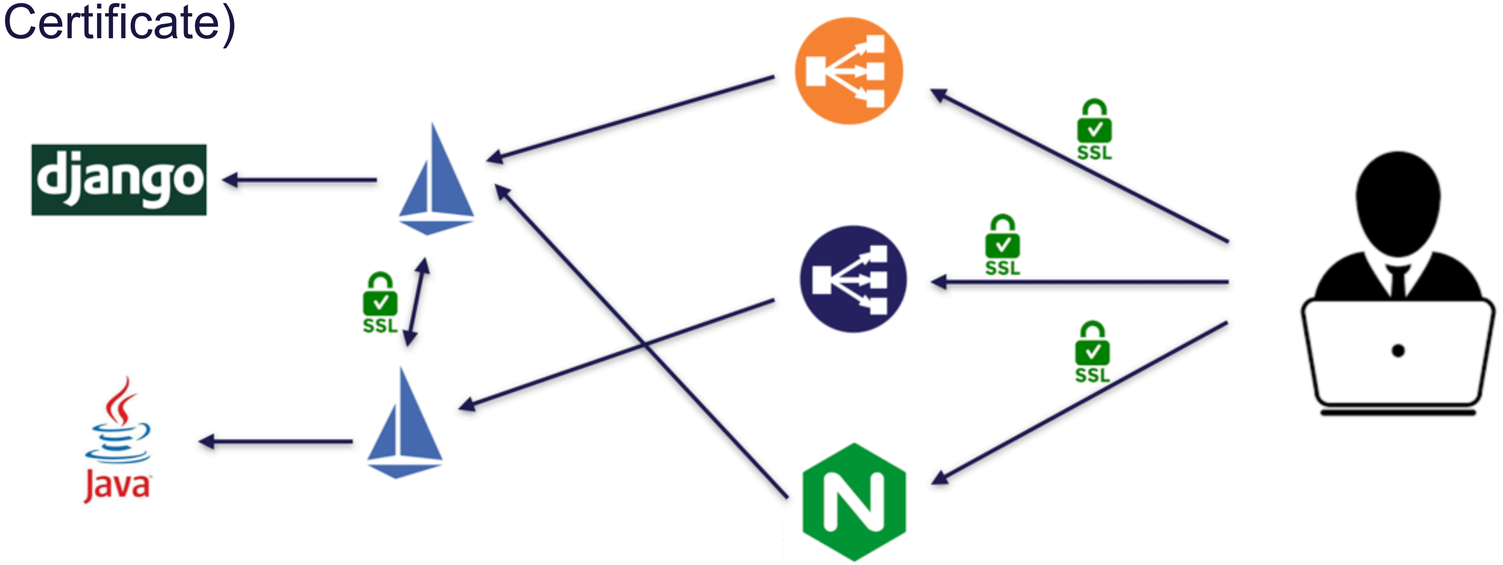
EKS Encryption in Transit

- 防禦 Man-In-The-Middle Attack (MITM) 中間人攻擊

- 負責對內 Service Mesh 也可以協助做到傳輸加密
  - AWS AppMesh (SSL via ACM Private Certificate)
  - Linkerd (SSL)
  - Istio (SSL)

- 負責對外使用 Ingress Controller
  - ALB Ingress Controller
  - Nginx Ingress Controller & NLB
  - 都要加上 SSL 憑證

- 盡量避免在 Pod 裡面做加解密的動作
  - 開發上會有不方便
  - 消耗額外的 CPU 資源去做 Hankshack



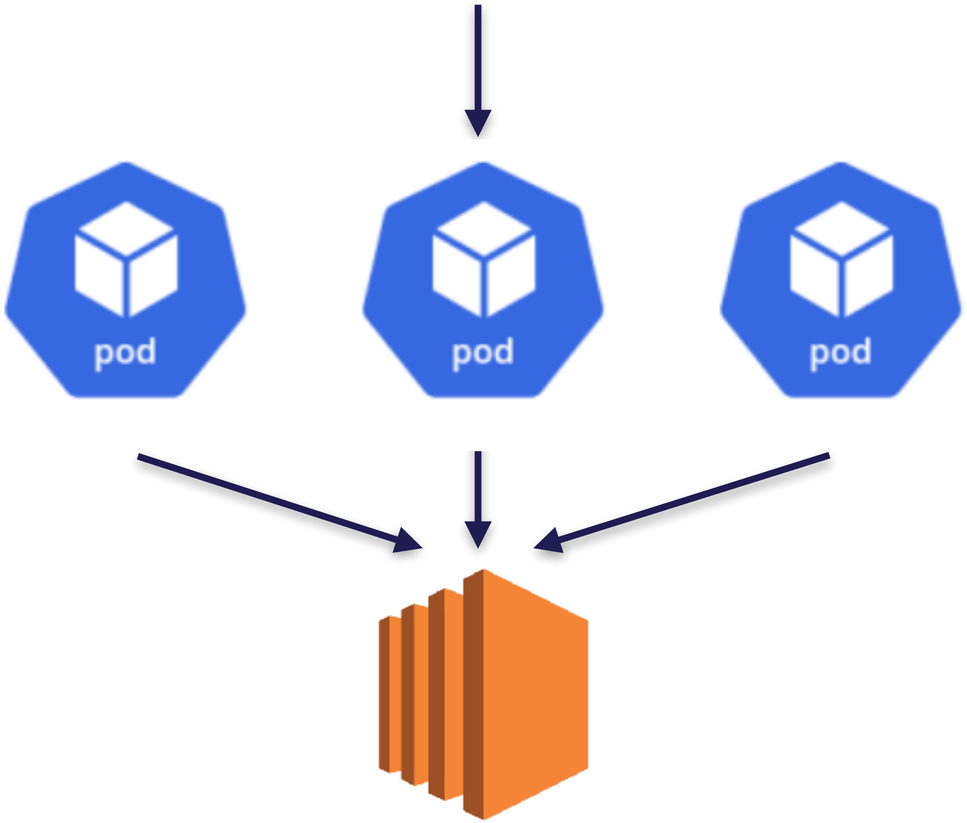
Network Security

Pods 預設是接收來自任何地方的流量

- Namespace 建立 Kubernetes Network Policy 去限制
- Calico Global Network Policy
- Pod 層級的流量控管是很重要的，將安全控制精細度提升到最細

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny
  namespace: default
spec:
  podSelector: {}
  policyTypes:
  - Ingress
  - Egress
```

```yaml
apiVersion: crd.projectcalico.org/v1
kind: GlobalNetworkPolicy
metadata:
  name: default-deny
spec:
  selector: all()
  types:
  - Ingress
  - Egress
```



Secret / Volume / ConfigMap

Security Group

Amazon EKS now supports assigning EC2 security groups to Kubernetes pods

Posted On: Sep 9, 2020

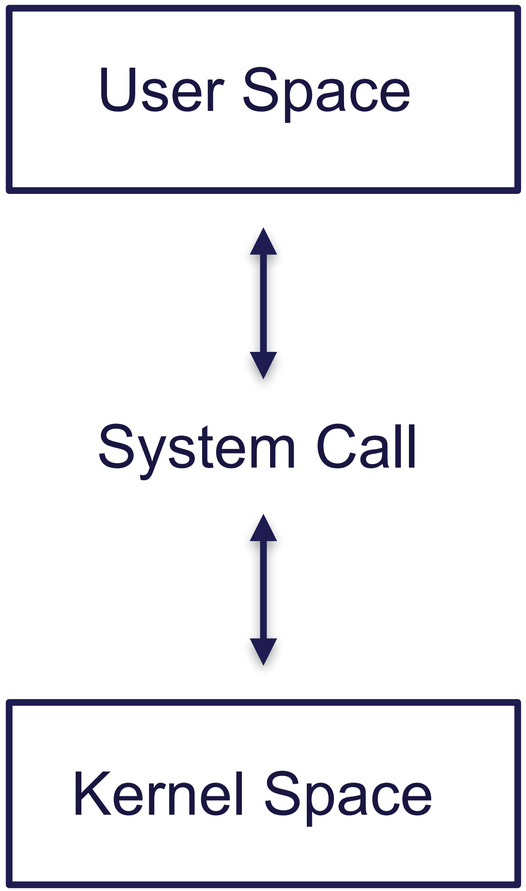
Amazon Elastic Kubernetes Service (EKS) customers can now leverage EC2 security groups to secure applications with varying network security requirements on shared cluster compute resources.

Previously, all pods on a node shared the same security groups. While IAM roles for service accounts solves the pod level security challenge at the authentication layer, many organization's compliance requirements also mandate network segmentation as an additional defense in depth step. Kubernetes network policies provide an option for controlling network traffic within the cluster, but do not support controlling access to AWS resources outside the cluster.

Now, network security rules that span pod to pod and pod to external AWS service traffic can be defined in a single place with EC2 security groups, and applied to individual pods and applications with Kubernetes native APIs. This makes it easy to achieve network security compliance in clusters that are shared across multiple teams and applications.

Support for assigning security groups to pods is available for most AWS Nitro based instances launched with new EKS clusters running Kubernetes version 1.17. Support for existing clusters will be rolled out over the coming weeks. To get started, visit the Amazon EKS documentation.

System Call



- Process 控制 (kill)
- 檔案控制 (read / write / open)
- Device 控制
- 系統資料存取
- 通訊管理

Runtime Security

- 攻擊方式
  - Pod 裡面做 System Call 攻擊，進而打進 Kernel

- 防禦方式
  - System Call 的範圍需要限制
  - 監控 System Call 使用狀況

- Docker 上也有預設的 Seccomp Profile 可以去做限制
- kubelet 上可以使用--seccomp-profile-root 去啟用

節點安全

Infrastructure Security

Host Security

- 定期升級 Node
  - 機器內升級或換新的機器

- 汰換流程
  - 先新增 AutoScaling Group / NodeGroup 並調度 Pod 到新的節點
  - 確保舊結點上沒有運行的 Pod (Draining)
  - Terminate 舊的節點

- 使用 Fargate 作為節點
  - 每次 Launch Pod 都會使用新的機器
  - 不支援 DaemonSet
  - 不支援具有特殊權限的 Pod
  - 不支援 Security Group for Pod
  - 可以適當搭配 EC2 節點做混合使用

Host Security

- 最小化 Node 權限
  - Session-Manager 取代 SSH 方便稽核以及控管
  - Production 環境的節點，不應該是進去 Debug 的地方
  - 可以做 Pod 的登入警報

- Node 要在 Private Subnet 內
  - 減少在 Internet 上的暴露
  - 非不得已要在 Public Subnets 的話，透過 Security Group 去控管

- 使用對容器量身打造的 OS 也可以提升安全性
  - RancherOS
  - Bottlerocket 🚀

監控與稽核

Monitoring and Auditing

Monitoring & Detective

- 開啟 Audit Log 並做異常警報
  - "authorization.k8s.io/decision" 指出 Request 是否已授權
  - "authorization.k8s.io/reason" 指出授權原因
  - EKS API Call 監控並發送警報
  - Cloudwatch Alarm 偵測 401 / 403 Counts 並發送警報
  - 透過 CloudwatchLogs Insight 顯示在 Dashboard 上做日常稽核

- 監控 CloudTrial Logs
  - 可以監控 IRSA 的使用情況
  - 是否有 Pod 異常送出非必要請求

Monitoring & Detective

- Audit Log 會成長的又快又多
  - 多儲存就會變貴
  - 過濾及篩選上也會更麻煩

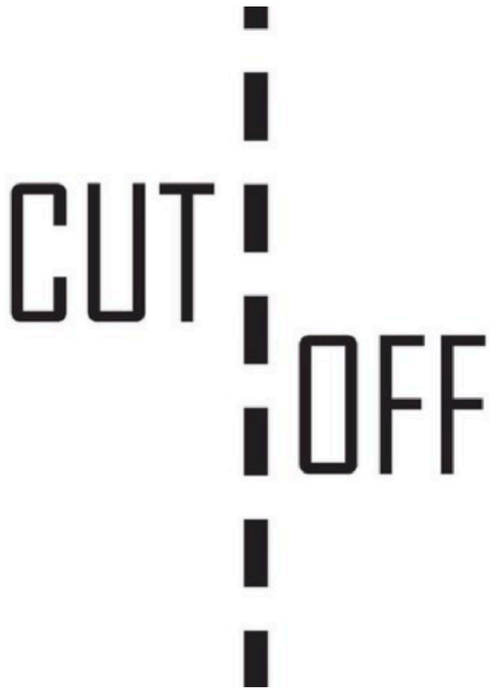
- 透過工具去分析 Audit Log
  - Falco (Runtime Security Tool)

- 建立 Log Lifecycle

Incident Response

如果已經發現 Pod 被污染 / 入侵了？

- 立即切斷與其他資源的所有接觸
  - 停止調度 Pod 到節點
  - Network Policy 阻斷該 Pod Inbound / Outbound 流量
  - Security Group 阻斷該 Host Inbound / Outbound 流量
  - 替換 Application 使用的 Credentials



Incident Response

攻擊者會試圖抹去攻擊的痕跡

- 開啟 Termination Protection 避免機器直接被終止
- Dump Memory / Netstat 去找出漏洞
- 查找 Log 去找出漏洞

總結

Conclusion

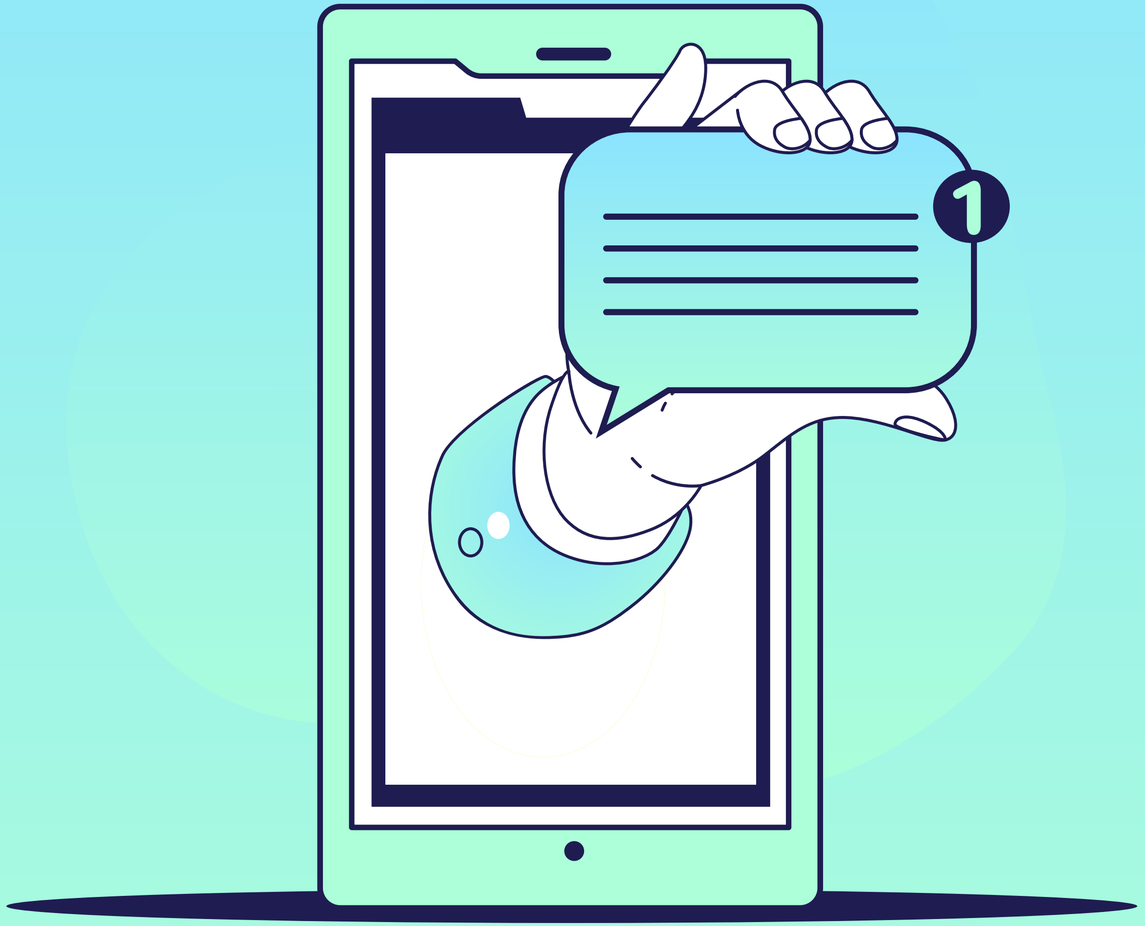
Conclusion

- 權限最小化，精細化，盡量減少被攻擊的可能性，沒有 100% 的防禦
- 安全性與方便性是雙面刃
- 防禦最重要的依舊是 "人"
  - DevOps 的電腦權限都是最大的
    - AWS Organizations 的方式 Switch Role
    - *saml2aws* 可以透過 SSO 產生臨時的 Token，電腦不留存 Credentials
    - 離開座位電腦要上鎖
    - 定期更換各項密碼
    - 工作用的電腦盡量不用瀏覽奇怪的網站
  - DevOps 面向的通常是 Developers
    - 定期舉辦教育訓練加強資安意識
    - 導入工具協助管理密碼 *(e.g. LastPass)*

THANKS!

Do you have any questions?

davidh83110@gmail.com
https://blog.davidh83110.com



CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, infographics & images by **Freepik** and illustrations by Stories