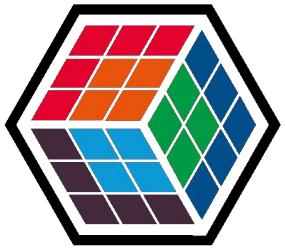
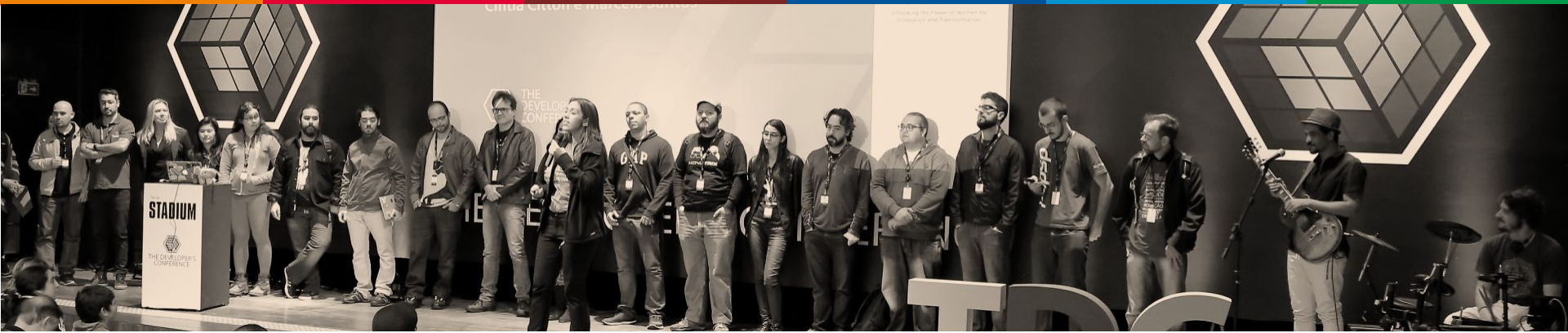


THE DEVELOPER'S CONFERENCE

# Trilha – Containers

## Kubernetes é seguro por default ou à prova de má configuração?

João Freire




## João Freire

Entusiasta em segurança da informação e cloud computing, apaixonado pelo universo Open Source.

System Engineer na Mandic Cloud Solutions

CKA - CKAD - AWS SSA

THE DEVELOPER'S CONFERENCE

THE DEVELOPER'S CONFERENCE

# machine:~# ./intro.sh

Quando pensamos em serviços em container rodando em um cluster Kubernetes, uma das primeiras coisas que nos vem à mente são suas vantagens e é quase impossível não pensar no isolamento dos recursos e granularidade de acesso providos por ele.

THE DEVELOPER'S CONFERENCE

# machine:~#./overview.sh

Atualmente muitas das plataformas de cloud oferecem um serviço de Kubernetes, sendo ele gerenciado ou não.

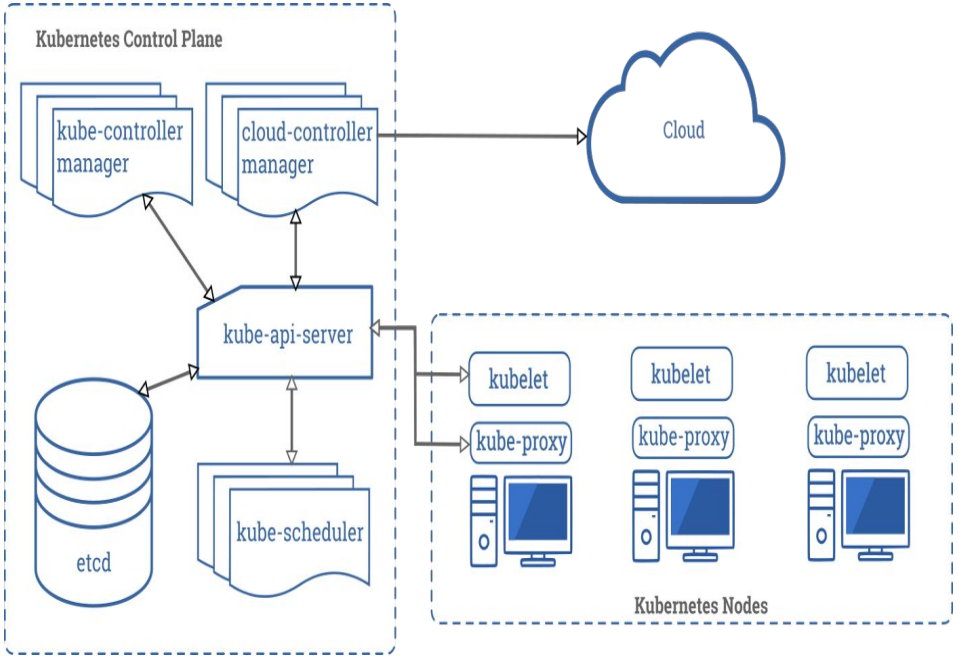
Isso tornou muito fácil a adesão na plataforma para alguns, entretanto existem algumas preocupações e ações que dependem de você para manter o ambiente seguro.

THE DEVELOPER'S CONFERENCE

# machine:~#./architecture_overview.sh

Para contextualizar, vamos ver um pouco da arquitetura da plataforma:

- Nodes
- Namespaces
- Services
- Deployment
- Pods

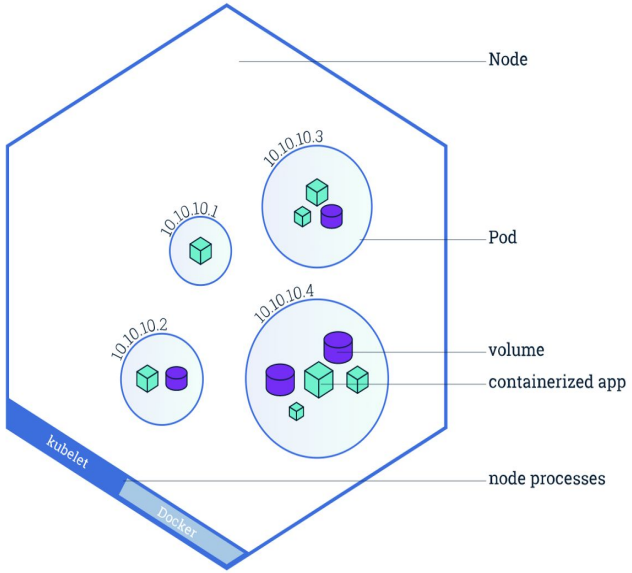


THE DEVELOPER'S CONFERENCE

# machine:~# kubectl get node

- Um node pode ser uma máquina virtual ou física, dependendo do cluster, com suas responsabilidades master ou worker. O nó master é responsável pelo control plane.

- Cada node contém os serviços necessários para executar pods e é gerenciado pelos masters. Os serviços em um nó incluem o container-runtime, kubelet e kube-proxy.



THE DEVELOPER'S CONFERENCE

# \Kube-apiserver

REST

API server é um front end para os control plane do Kubernetes

- Todos os clientes e componentes se comunicam através do API Server.
- Componente core do control Plane.
- A API Kubernetes permite consultar e manipular os objetos do Kubernetes.

THE DEVELOPER'S CONFERENCE

# \etcd

- Armazenamento de valor-chave consistente e com alta disponibilidade para persistir os estados do cluster.

- Armazena objetos e informações de configuração.

THE DEVELOPER'S CONFERENCE

# \kube-controller-manager

Serve como o daemon principal que gerencia todos os loops de controle.

- Node controller
- Replication controller
- Endpoints controller
- Service Account & Token controllers

THE DEVELOPER'S CONFERENCE

# \kube-scheduler

Componente do control plane do Kubernetes responsável por atribuir um pod recém criado a um node.

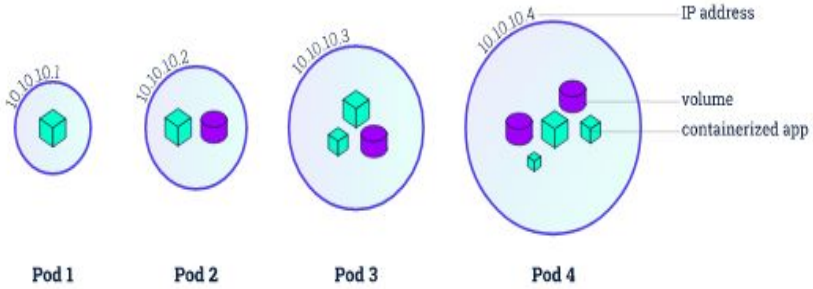
Fatores levados em consideração:

- affinity e anti-affinity
- recursos definidos
- tolerances e taints

THE DEVELOPER'S CONFERENCE

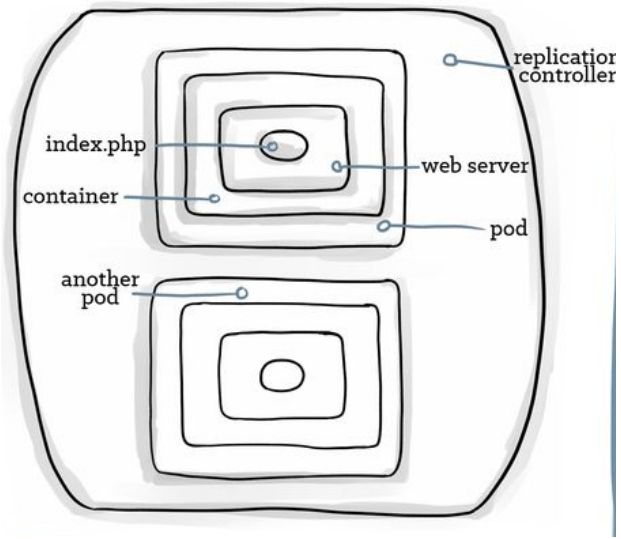
# machine:~# kubectl get pods

- Pod, na hierarquia do Kubernetes, é a menor entidade.
- "Dentro" de um pod é onde temos os containers.
- Um pod pode possuir um ou mais containers.
- Os containers dentro de um pod dividem a mesma camada de rede.
- Efêmeros



# machine:~# kubectl get deployment

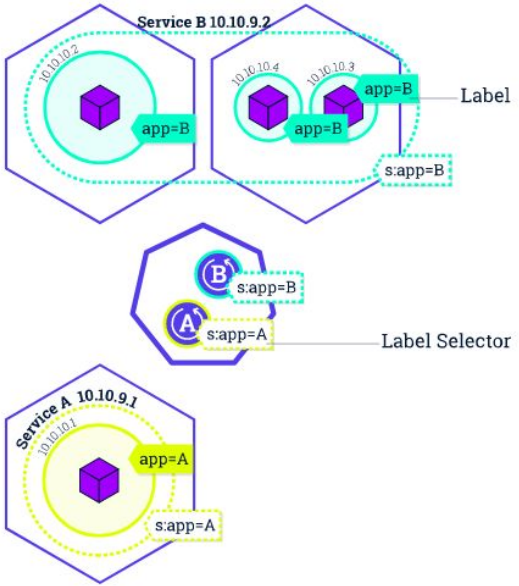
- Prove o controle das configurações dos pods.
- "Alta hierarquia"
- Possibilita o scale up, scale down
- Rolling deploys

THE DEVELOPER'S CONFERENCE

# machine:~# kubectl get services

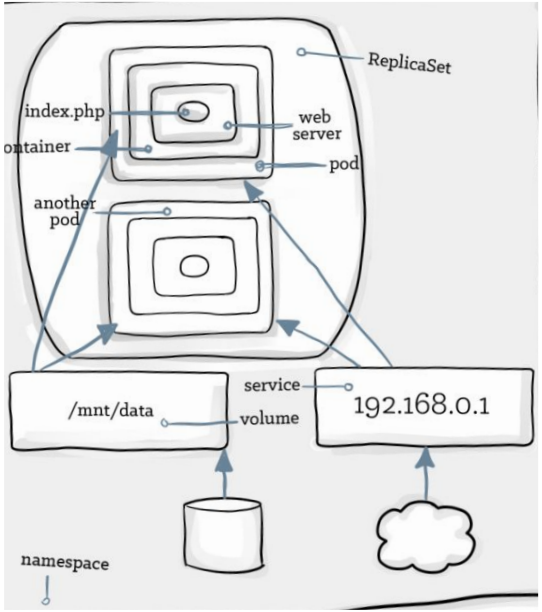
- Fornece IP e um único nome DNS
- Balanceador de carga
- Não efêmeros
- Utiliza  label selector
- Discovery



THE DEVELOPER'S CONFERENCE

# machine:~# kubectl get namespaces

Uma abstração usada pelo Kubernetes para oferecer suporte a vários clusters virtuais no mesmo cluster físico



THE DEVELOPER'S CONFERENCE

# $ k get serviceaccount,roles

**RBAC(Role-Based Access Control)**:

Permite configurar um conjuntos de permissões específicas e detalhadas que definem como um determinado usuário ou grupo de usuários podem interagir com qualquer objeto do Kubernetes ou em um namespace em particular.

**ServiceAccount**:

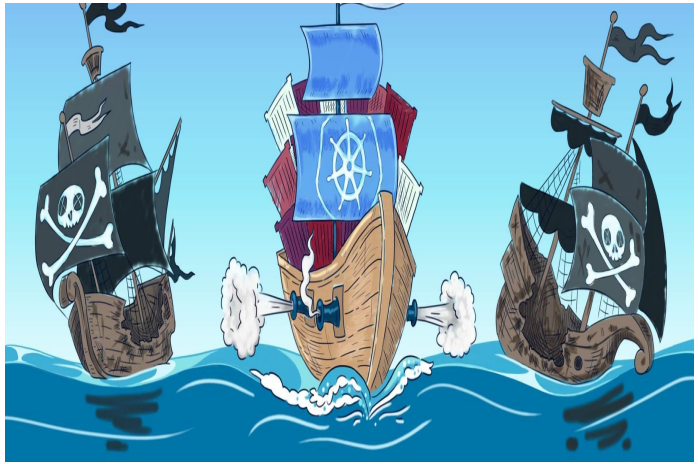
Fornece uma identidade para processos executados pod.

THE DEVELOPER'S CONFERENCE

# machine:~#./init_attack.sh

Comprometer uma aplicação exposta para internet é um dos principais vetores de ataque para um cluster Kubernetes.

Existem outros pontos como, recursos sensíveis do Kubernetes expostos para internet sem nenhum tipo de autenticação, além dos CVE'S publicados como por exemplo o recente CVE-2020-8559



THE DEVELOPER'S CONFERENCE

# machine:~#./pod_committed.sh

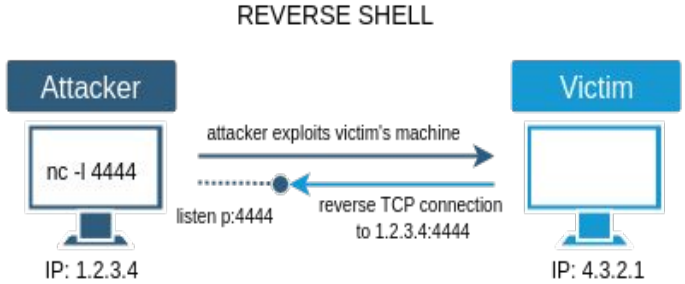
Comprometer um pod pode ter sérias consequências como:

- Possibilitar a interação com outros pods.
- Interagir com com o API server.
- Obter as secrets e configs maps utilizados no pod além da service account.
- Enumerar o dns interno do cluster para descobrir outros serviços.
- Possibilidade de interagir com o ETCD.
- Escalar privilégios do container para o S.O.

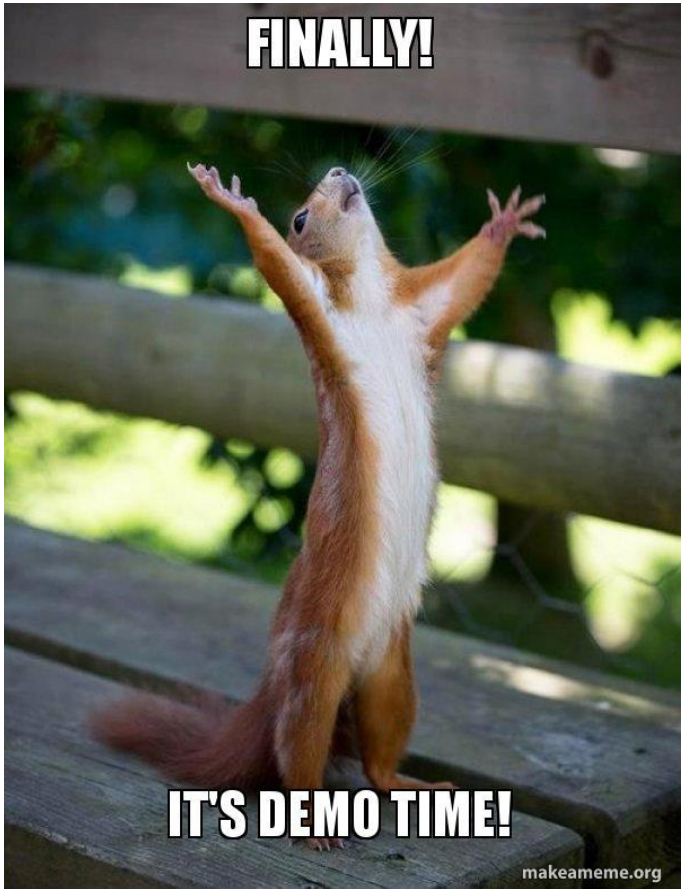
THE DEVELOPER'S CONFERENCE

# machine:~#./init_attack.sh

Analisaremos o seguinte cenário: possuímos uma aplicação web vulnerável na qual conseguimos executar comandos arbitrários, conseguindo fazer uma shell reversa.




Não conhecemos nada dentro do ambiente apenas conseguimos o acesso




THE DEVELOPER'S CONFERENCE

THE DEVELOPER'S CONFERENCE

# machine:~# kubectl get answers

THE DEVELOPER'S CONFERENCE

# machine:~# kubectl describe answers

## Sim !

O Kubernetes foi projetado pensando em vários pontos de segurança, mas isso não significa que ao lançar sua aplicação você está seguro.

THE DEVELOPER'S CONFERENCE

# machine:~# hardening_to_apps.sh

- Sempre usar limits e requests nos containers.
- Separar recursos por namespace.
- Remover as permissões da service account default.
- Não usar contêineres como root.
- Faça suas imagens.
- Tentar utilizar PodSecurityPolicy.
- SElinux e AppArmor
- Evitar os privileged containers.

```
---
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "128Mi"
        cpu: "500m"
```

THE DEVELOPER'S CONFERENCE

# machine:~# kubectl get polices

```yaml
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: access-myapp
spec:
  podSelector:
    matchLabels:
      app: myapp
  ingress:
  - from:
    - podSelector:
        matchLabels:
          access: "true"
```

```yaml
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny
  namespace: default
spec:
  podSelector: {}
  egress:
  - to:
    - podSelector:
        matchLabels:
          k8s-app: kube-dns
  - ports:
    - protocol: UDP
      port: 53
  policyTypes:
    - Ingress
    - Egress
```

THE DEVELOPER'S CONFERENCE

# machine:~#./hardening_to_nodes.sh

- Sempre utilizar RBAC e aproveitar de sua granularidade.
- Não utilize a insecure-port
- API Server
  - --authorization-mode=Node,RBAC
  - --anonymous-auth=false
- Controle o acesso  kubelet.
  - --anonymous-auth=false
  - --authorization-mode=Webhook
- Utilizar autenticação no ETCD.

THE DEVELOPER'S CONFERENCE

# machine:~#./tools.sh

- Kube-Hunter
- Kube-bench
- Trivy
- Starboard
- Kube-scan

THE DEVELOPER'S CONFERENCE

# Dúvidas ?

THE DEVELOPER'S CONFERENCE

# Obrigado !


## João Freire

/in/joaopaulocunhafreire

@p0ssuidao

/P0ssuidao


Lab + explicação: