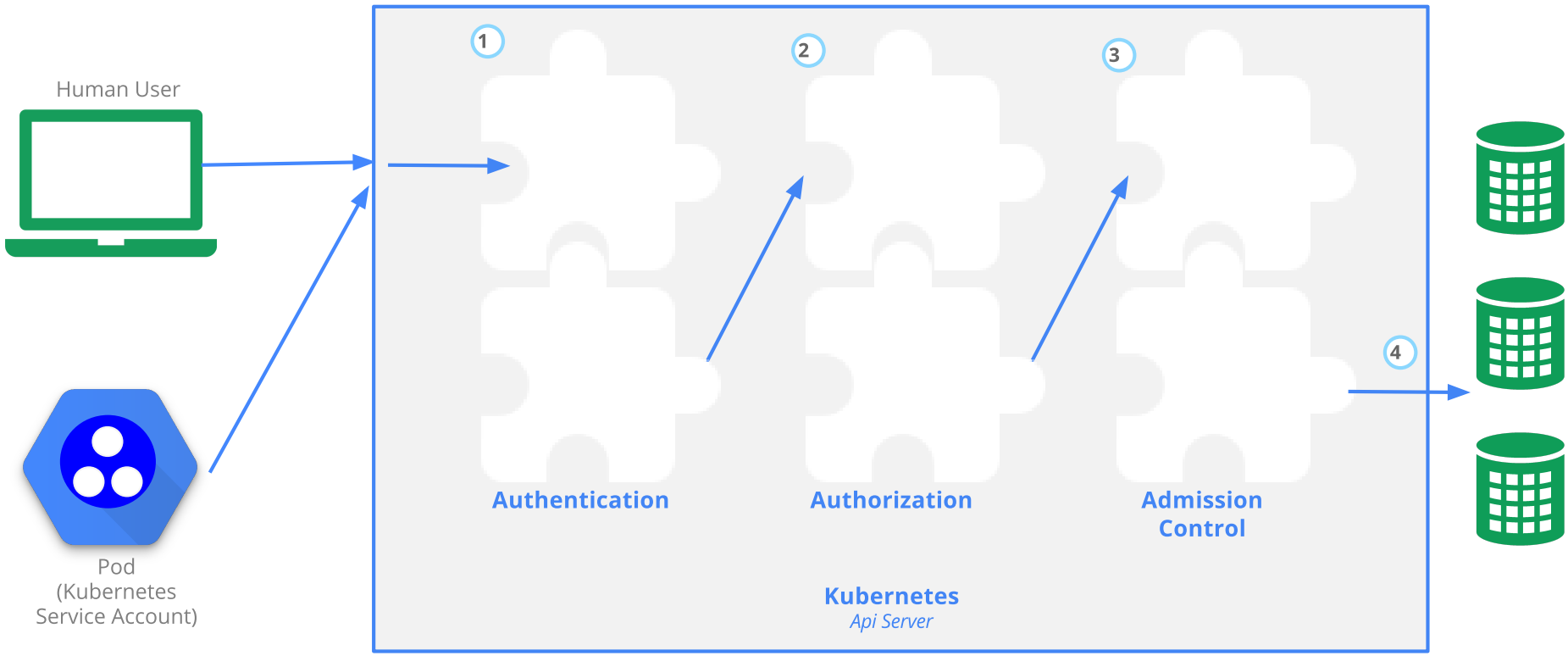
Kubernetes Authentication: Client Certificate

Dec 18, 2019



1

For access control, Kubernetes steps the procedures above for each API operation: authentication (who can access), authorization (what can be accessed), and admisssion control. This post is about Kubernetes **authentication**.

All API accesses are handled by Kubernetes api server. All accesses have to be authenticated by the API server for Kubernetes operations. Kubernetes API server serve on 2 ports: one for testing, and the other for all other cases. By default, these ports are:

- `https://<ip>:6443` : use TLS (and certificate), `<ip>` is the first non-localhost network interface, request are handled by authentication and authorization modules

The HTTP request moves to the authentication step when users access to the API server through the port 6443 and establishes a TLS connection.

Kubernetes authentication strategies [2][3]

Kubernetes provides the following modules for authentication.

- client certificates (default)
- bearer tokens (authentication proxy)
- HTTP basic auth

Client certificate

By default, Kubernetes set by `kubeadm` uses X509 based client certificate for authentication.

Official documentation[4] says:

> To enable X509 client certificate authentication to the kubelet's HTTPS endpoint:
> - start the kubelet with the –client-ca-file flag, providing a CA bundle to verify client certificates with
> - start the apiserver with –kubelet-client-certificate and –kubelet-client-key flags
> - see the apiserver authentication documentation for more details

Let's see how thiese configurations are set by default.

`kubeadm` initialize kubelet as a systemd service:

```
Environment="KUBELET_KUBECONFIG_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/et
Environment="KUBELET_CONFIG_ARGS=--config=/var/lib/kubelet/config.yaml"
## This is a file that "kubeadm init" and "kubeadm join" generates at runtime, populating the KUBELET_KUBEADM_ARGS
EnvironmentFile=-/var/lib/kubelet/kubeadm-flags.env
## This is a file that the user can use for overrides of the kubelet args as a last resort. Preferably, the user sh
## the .NodeRegistration.KubeletExtraArgs object in the configuration files instead. KUBELET_EXTRA_ARGS should be s
EnvironmentFile=-/etc/default/kubelet
ExecStart=
ExecStart=/usr/bin/kubelet $KUBELET_KUBECONFIG_ARGS $KUBELET_CONFIG_ARGS $KUBELET_KUBEADM_ARGS $KUBELET_EXTRA_ARGS
```

that uses `/etc/kubernetes/kubelet.confg` as a value of `--kubeconfig` flag, which contains:

```
authentication:
  anonymous:
    enabled: false
  x509:
    clientCAFile: /etc/kubernetes/pki/ca.crt
```

The client certificate authority (CA) file is stored in `/etc/kubernetes/pki` , the default path of certificates.

Kubernetes api-server runs on kubernetes master node as a static pod. Inspecting it, we know `--kubelet-client-certificate` and `--kubelet-client-key` flags are set as well.

```
$ kubectl describe pods kube-apiserver-kube-test --namespace=kube-system
Name:                 kube-apiserver-kube-test
Namespace:            kube-system
Priority:             2000000000
Priority Class Name:  system-cluster-critical
```

```
  Containers:
    kube-apiserver:
      Container ID:  cri-o://4537833ae99fca1fcf26f4ec3b9bcb6da99ef2b2e7da88d9674881c3c25e2f9a
      Image:         k8s.gcr.io/kube-apiserver:v1.16.4
      Image ID:      k8s.gcr.io/kube-apiserver@sha256:b24373236fff6dcc0e154433b43d53a9b2388cdf39f05fbc46ac73082c9b05f
      Port:          <none>
      Host Port:     <none>
      Command:
        kube-apiserver
        ...
        --kubelet-client-certificate=/etc/kubernetes/pki/apiserver-kubelet-client.crt
        --kubelet-client-key=/etc/kubernetes/pki/apiserver-kubelet-client.key
        ...
```

kubectl access

When we use `kubectl`, everything works fine. This does not mean `kubectl` is special, nor bypasses authentication module. With `KUBECONFIG` environment variable, `kubectl` automatically loads a configuration file with certificate information before accessing the api server. With higher level of verbose, you can see this flow.

```
$ echo $KUBECONFIG
/etc/kubernetes/admin.conf
$ kubectl --v=7 get pods
I1218 16:27:24.481836   11192 loader.go:375] Config loaded from file:  /etc/kubernetes/admin.conf
I1218 16:27:24.485689   11192 round_trippers.go:420] GET https://ip:6443/api/v1/namespaces/default/pods?limit=500
I1218 16:27:24.485700   11192 round_trippers.go:427] Request Headers:
I1218 16:27:24.485704   11192 round_trippers.go:431]     User-Agent: kubectl/v1.16.3 (linux/amd64) kubernetes/b3cbb
I1218 16:27:24.485708   11192 round_trippers.go:431]     Accept: application/json;as=Table;v=v1beta1;g=meta.k8s.io,
```

In this node `kubectl` uses `/etc/kubernetes/admin.conf` as its credentials, which contains:

```
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRU...
    server: https://ip:6443
  name: kubernetes
...
users:
- name: kubernetes-admin
  user:
    client-certificate-data: LS0tLS1CRU...
    client-key-data: LS0tLS1CRU...
```

`certificate-authority-data` is a base64-encoded string of `/etc/kubernetes/ca.crt` [5]. `client-certificate-data` and `client-key-data` are base64-encoded kubernetes-admin certificate and key, respectively. This admin certificate is automatically created and managed by kubeadm.

```
$ kubeadm alpha certs check-expiration
CERTIFICATE                EXPIRES                  RESIDUAL TIME   EXTERNALLY MANAGED
admin.conf                 Dec 17, 2020 07:20 UTC   364d            no
apiserver                  Dec 17, 2020 07:20 UTC   364d            no
apiserver-etcd-client      Dec 17, 2020 07:20 UTC   364d            no
apiserver-kubelet-client   Dec 17, 2020 07:20 UTC   364d            no
controller-manager.conf    Dec 17, 2020 07:20 UTC   364d            no
etcd-healthcheck-client    Dec 17, 2020 07:20 UTC   364d            no
etcd-peer                  Dec 17, 2020 07:20 UTC   364d            no
```

`kubeadm alpha certs` command shows the client certificates in the `/etc/kubernetes/pki` [6] and the client certificate embedded in `KUBECONFIG` files (admin.conf, controller-manager.conf, and scheduler.conf).

For more details, refer to [4], [7], and [8].

1. Controlling access: https://kubernetes.io/docs/reference/access-authn-authz/controlling-access/ ↩
2. Authentication strategies: https://kubernetes.io/docs/reference/access-authn-authz/authentication/#authentication-strategies ↩
3. 쿠버네티스 #16: 보안 계정 인증과 권한 인가 https://bcho.tistory.com/1272 ↩
4. Understanding Kubernetes Authentication and Authorization http://cloudgeekz.com/1045/kubernetes-authentication-and-authorization.html ↩
5. Access Kubernetes API with Client Ceritifcate. https://codefarm.me/2019/02/01/access-kubernetes-api-with-client-certificates/ ↩
6. Certificate Management with kubeadm. https://kubernetes.io/docs/tasks/administer-cluster/kubeadm/kubeadm-certs/ ↩
7. Authentication and Authorization in Kubernetes https://www.sovsystems.com/blog/authentication-and-authorization-in-kubernetes ↩
8. 쿠버네티스 인증 https://arisu1000.tistory.com/27847 ↩

#kubernetes #study

Previous
Generate a Self-signed Certificate

Next
Cmake for Custom Library Build System in Go

KVM Internal: How a VM is Created?

4 years ago • 2 comments

KVM Internal: How a VM is Created? Contents References KVM is an ...

Introduction to VFIO

4 years ago • 1 comment

Implementation is in /linux/drviers/vfio/pci . Main driver code is ...

Go Modules: an Alternative to ...

a year ago • 1 comment

This post introduces Go modules, introduced in Go version 1.11. Go Modules? ...

Programm Kubernete

2 years ago • 9 c

Programming CRDs In [prev briefly introdu

Cmake for Custom Library Build System in Go
Implementing Kubernetes C++ Client Library using Go Client Library
Installing Kubernetes and cri-o in Debian
Interactions between cri-o and conmon in Kubernetes
Kubernetes