PKI certificates and requirements

Kubernetes requires PKI certificates for authentication over TLS. If you install Kubernetes with kubeadm, the certificates that your cluster requires are automatically generated. You can also generate your own certificates -- for example, to keep your private keys more secure by not storing them on the API server. This page explains the certificates that your cluster requires.

How certificates are used by your cluster

Kubernetes requires PKI for the following operations:

- Client certificates for the kubelet to authenticate to the API server
- Server certificate for the API server endpoint
- Client certificates for administrators of the cluster to authenticate to the API server
- Client certificates for the API server to talk to the kubelets
- Client certificate for the API server to talk to etcd
- Client certificate/kubeconfig for the controller manager to talk to the API server
- Client certificate/kubeconfig for the scheduler to talk to the API server.
- Client and server certificates for the front-proxy

> **Note:** `front-proxy` certificates are required only if you run kube-proxy to support an extension API server.

etcd also implements mutual TLS to authenticate clients and peers.

Where certificates are stored

If you install Kubernetes with kubeadm, certificates are stored in `/etc/kubernetes/pki`. All paths in this documentation are relative to that directory.

Configure certificates manually

If you don't want kubeadm to generate the required certificates, you can create them in either of the following ways.

Single root CA

You can create a single root CA, controlled by an administrator. This root CA can then create multiple intermediate CAs, and delegate all further creation to Kubernetes itself.

Required CAs:

path	Default CN	description
ca.crt,key	kubernetes-ca	Kubernetes general CA
etcd/ca.crt,key	etcd-ca	For all etcd-related functions
front-proxy-ca.crt,key	kubernetes-front-proxy-ca	For the front-end proxy

On top of the above CAs, it is also necessary to get a public/private key pair for service account management, `sa.key` and `sa.pub`.

All certificates

If you don't wish to copy the CA private keys to your cluster, you can generate all certificates yourself.

Required certificates:

Default CN	Parent CA	O (in Subject)	kind	hosts (SAN)
kube-etcd	etcd-ca		server, client	`localhost`, `127.0.0.1`
kube-etcd-peer	etcd-ca		server, client	`<hostname>`, `<Host_IP>`, `localhost`, `127.0.0.1`
kube-etcd-healthcheck-client	etcd-ca		client	
kube-apiserver-etcd-client	etcd-ca	system:masters	client	
kube-apiserver	kubernetes-ca		server	`<hostname>`, `<Host_IP>`, `<advertise_IP>`, [1]
kube-apiserver-kubelet-client	kubernetes-ca	system:masters	client	
front-proxy-client	kubernetes-front-proxy-ca		client	

[1]: any other IP or DNS name you contact your cluster on (as used by kubeadm the load balancer stable IP and/or DNS name, `kubernetes`, `kubernetes.default`, `kubernetes.default.svc`, `kubernetes.default.svc.cluster`, `kubernetes.default.svc.cluster.local`)

where `kind` maps to one or more of the x509 key usage types:

kind	Key usage
server	digital signature, key encipherment, server auth
client	digital signature, key encipherment, client auth

> **Note:** Hosts/SAN listed above are the recommended ones for getting a working cluster; if required by a specific setup, it is possible to add additional SANs on all the server certificates.

Note:
For kubeadm users only:

- The scenario where you are copying to your cluster CA certificates without private keys is referred as external CA in the kubeadm documentation.
- If you are comparing the above list with a kubeadm generated PKI, please be aware that `kube-etcd`, `kube-etcd-peer` and `kube-etcd-healthcheck-client` certificates are not generated in case of external etcd.

Certificate paths

Certificates should be placed in a recommended path (as used by kubeadm). Paths should be specified using the given argument regardless of location.

Default CN	recommended key path	recommended cert path	command	key argument	cert argument
etcd-ca	etcd/ca.key	etcd/ca.crt	kube-apiserver		--etcd-cafile
kube-apiserver-etcd-client	apiserver-etcd-client.key	apiserver-etcd-client.crt	kube-apiserver	--etcd-keyfile	--etcd-certfile
kubernetes-ca	ca.key	ca.crt	kube-apiserver		--client-ca-file
kubernetes-ca	ca.key	ca.crt	kube-controller-manager	--cluster-signing-key-file	--client-ca-file, --root-ca-file, --cluster-signing-cert-file
kube-apiserver	apiserver.key	apiserver.crt	kube-apiserver	--tls-private-key-file	--tls-cert-file
kube-apiserver-kubelet-client	apiserver-kubelet-client.key	apiserver-kubelet-client.crt	kube-apiserver	--kubelet-client-key	--kubelet-client-certificate
front-proxy-ca	front-proxy-ca.key	front-proxy-ca.crt	kube-apiserver		--requestheader-client-ca-file
front-proxy-ca	front-proxy-ca.key	front-proxy-ca.crt	kube-controller-manager		--requestheader-client-ca-file
front-proxy-client	front-proxy-client.key	front-proxy-client.crt	kube-apiserver	--proxy-client-key-file	--proxy-client-cert-file

Default CN	recommended key path	recommended cert path	command	key argument	cert argument
etcd-ca	etcd/ca.key	etcd/ca.crt	etcd		--trusted-ca-file, --peer-trusted-ca-file
kube-etcd	etcd/server.key	etcd/server.crt	etcd	--key-file	--cert-file
kube-etcd-peer	etcd/peer.key	etcd/peer.crt	etcd	--peer-key-file	--peer-cert-file
etcd-ca		etcd/ca.crt	etcdctl		--cacert
kube-etcd-healthcheck-client	etcd/healthcheck-client.key	etcd/healthcheck-client.crt	etcdctl	--key	--cert

Same considerations apply for the service account key pair:

private key path	public key path	command	argument
sa.key		kube-controller-manager	--service-account-private-key-file
	sa.pub	kube-apiserver	--service-account-key-file

Configure certificates for user accounts

You must manually configure these administrator account and service accounts:

filename	credential name	Default CN	O (in Subject)
admin.conf	default-admin	kubernetes-admin	system:masters
kubelet.conf	default-auth	system:node: `<nodeName>` (see note)	system:nodes
controller-manager.conf	default-controller-manager	system:kube-controller-manager	
scheduler.conf	default-scheduler	system:kube-scheduler	

Note: The value of `<nodeName>` for `kubelet.conf` **must** match precisely the value of the node name provided by the kubelet as it registers with the apiserver. For further details, read the Node Authorization.

1. For each config, generate an x509 cert/key pair with the given CN and O.

2. Run `kubectl` as follows for each config:

```
KUBECONFIG=<filename> kubectl config set-cluster default-cluster --server=https://<h
KUBECONFIG=<filename> kubectl config set-credentials <credential-name> --client-key
KUBECONFIG=<filename> kubectl config set-context default-system --cluster default-cl
KUBECONFIG=<filename> kubectl config use-context default-system
```

These files are used as follows:

filename	command	comment
admin.conf	kubectl	Configures administrator user for the cluster
kubelet.conf	kubelet	One required for each node in the cluster.
controller-manager.conf	kube-controller-manager	Must be added to manifest in `manifests/kube-controller-manager.yaml`
scheduler.conf	kube-scheduler	Must be added to manifest in `manifests/kube-scheduler.yaml`

Feedback

Was this page helpful?

Yes No

Last modified March 05, 2021 at 10:44 PM PST : Replace redirect links of kubeadm (e7a25a823)