KubeCon | CloudNativeCon

North America 2019

How Kubernetes components communicate securely in your cluster

Maya Kaczorowski, Google Cloud

Maya Kaczorowski

Product Manager,
Container Security

Google Cloud

@MayaKaczorowski

Agenda

How do Kubernetes components communicate securely in your cluster?

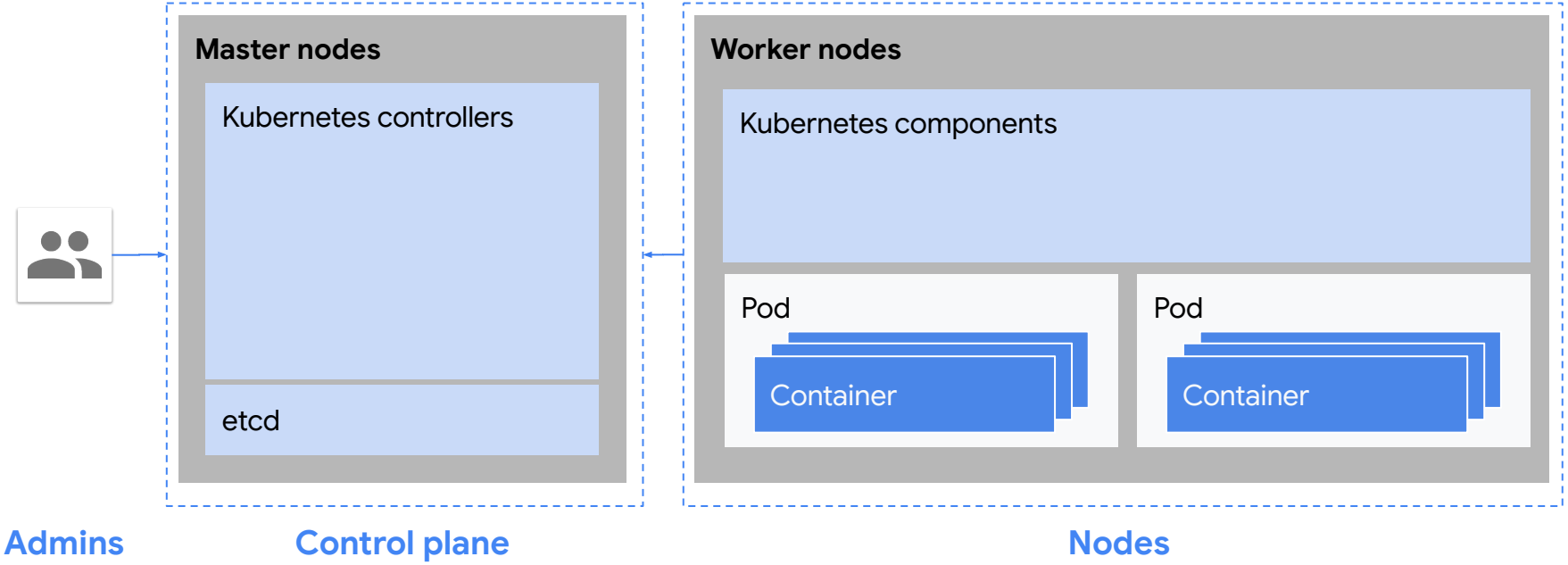
1. Components of Kubernetes
2. Communications security
3. Kubernetes' Certificate Authority (CA)
4. Protection of Kubernetes communications
5. Summary

Google Cloud

Components of Kubernetes

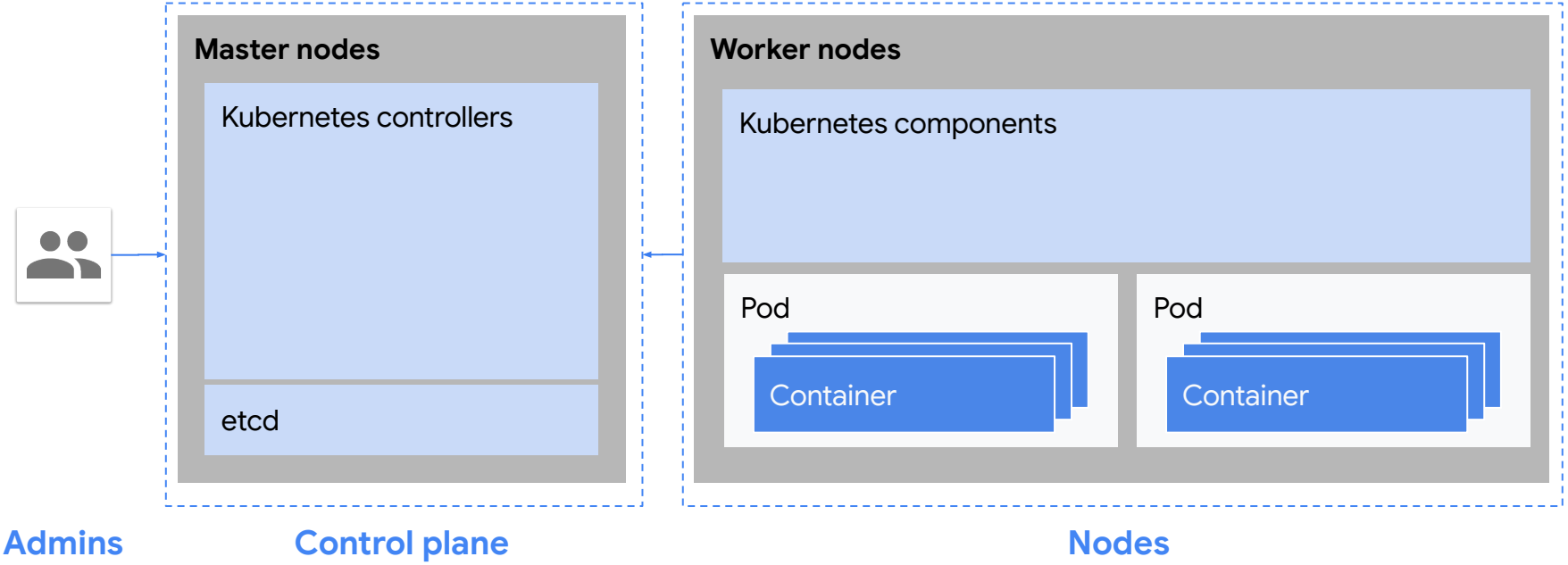
Google Cloud

Your cluster - managed by Kubernetes components



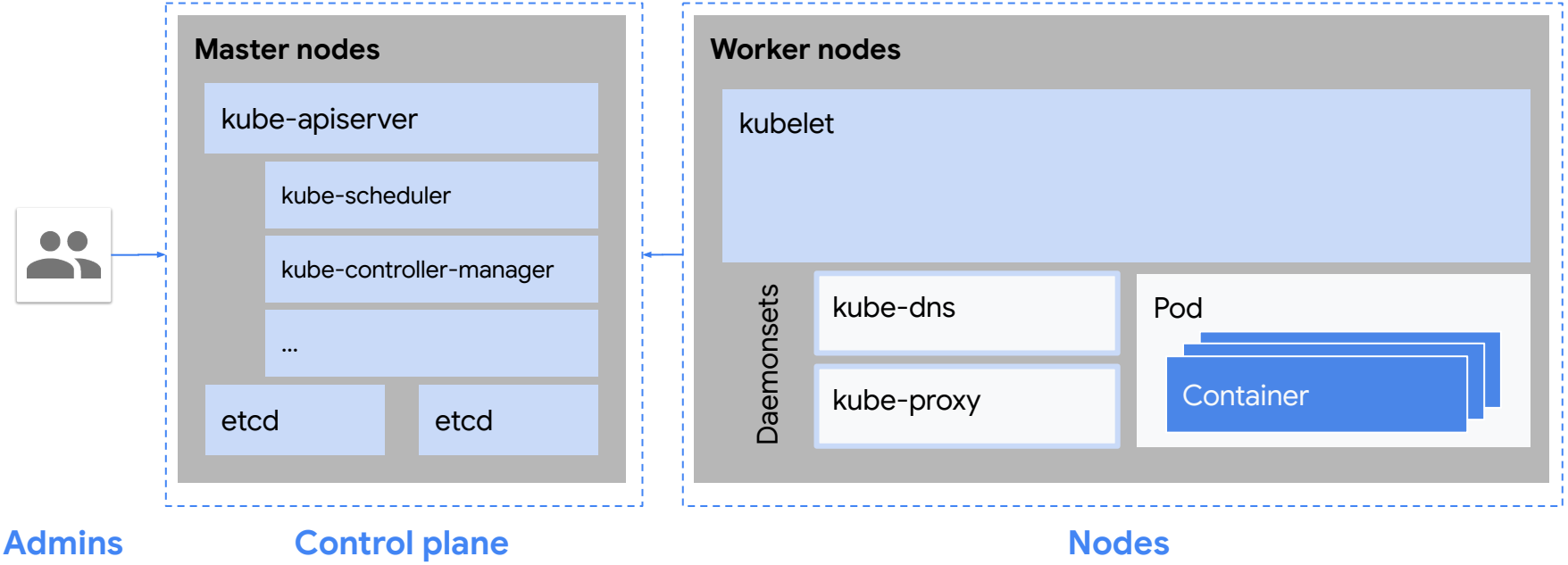
Google Cloud

Your cluster - managed by Kubernetes components



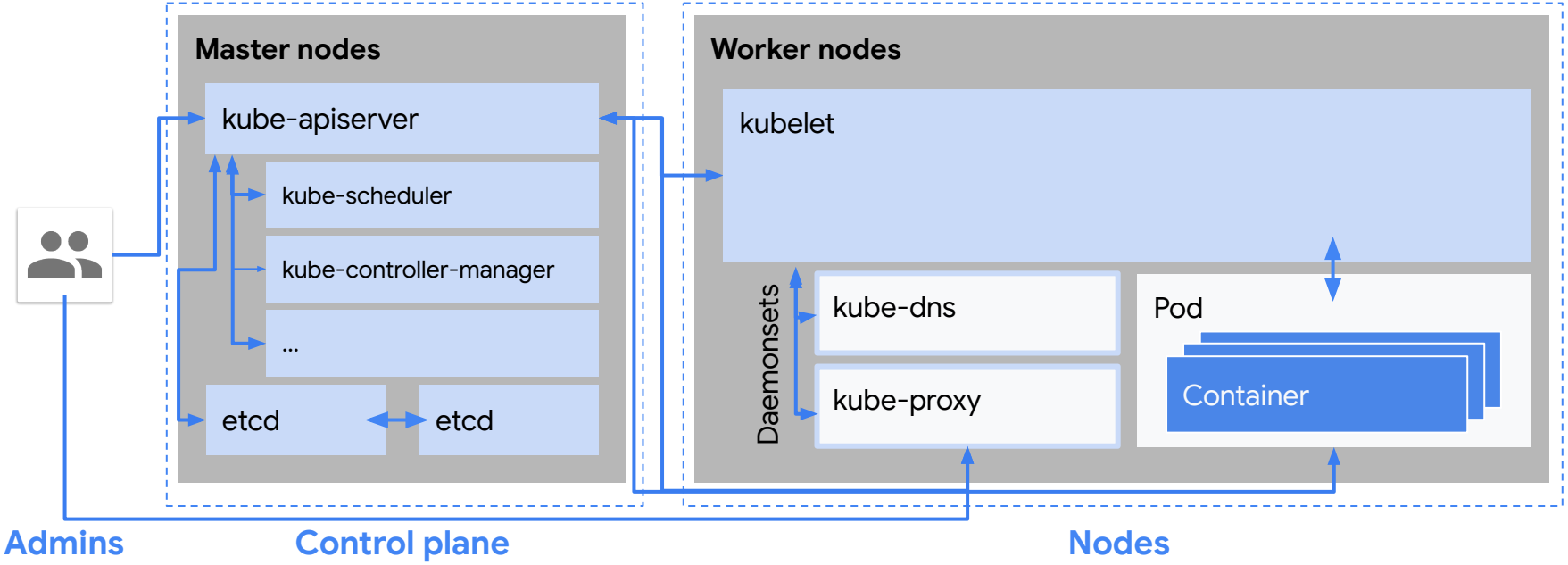
Google Cloud

Your cluster - lots of Kubernetes components!



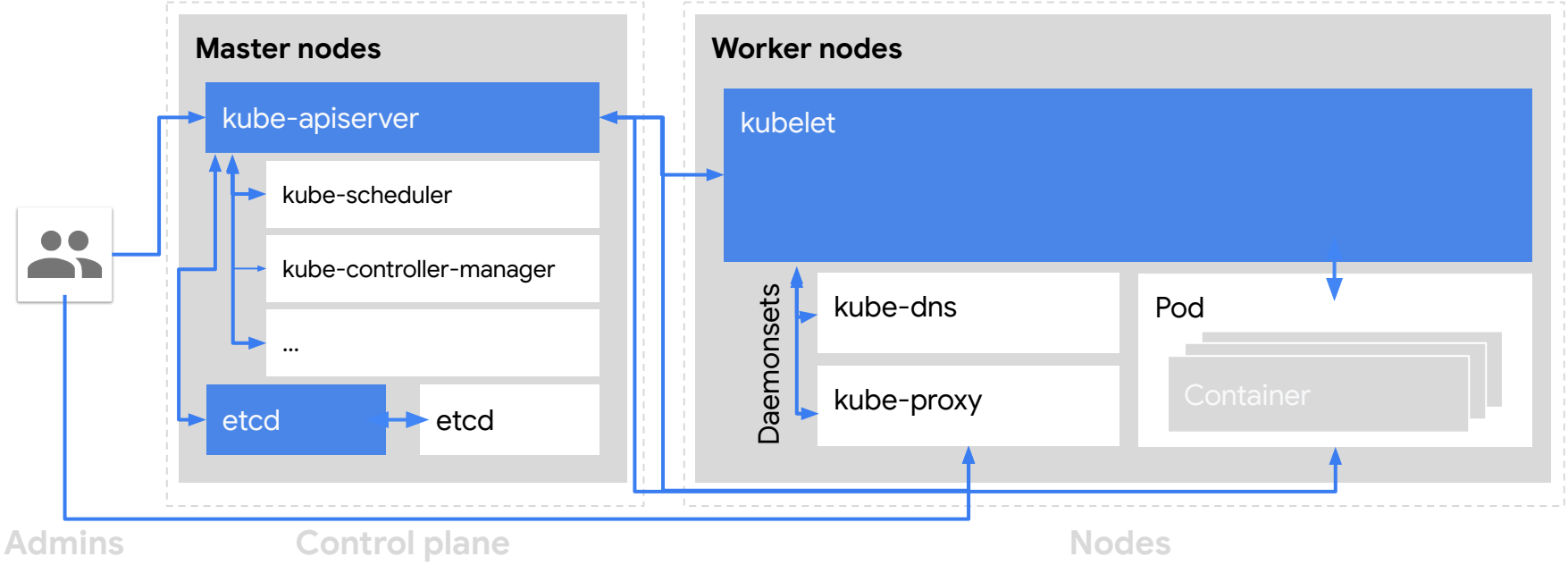
Google Cloud

Your cluster - communications between components



Google Cloud

Your cluster - communications between components



Google Cloud

Communications security

Authentication

Integrity

Encryption

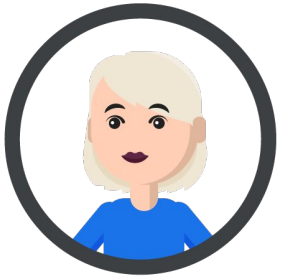
Different trust is required for different parties

Client -> Server

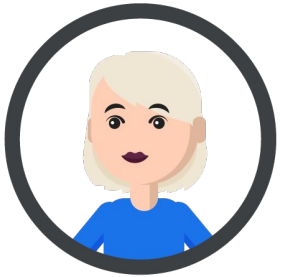
Alice
Client
i.e. Requestor

Bob
Server
i.e. Receiver

Peer



Carol
Client + Server



Carlos
Client + Server

Transport Layer Security (TLS)

Alice
Client

TCP
Establish connection

SYN

ACK

SYNACK

TLS
Authenticate and establish encryption

ClientHello

ClientKeyExchange
ChangeCipherSpec
Finished

ServerHello
Certificate
ServerHello Done

ChangeCipherSpec
Finished

Request/response
Transmit data

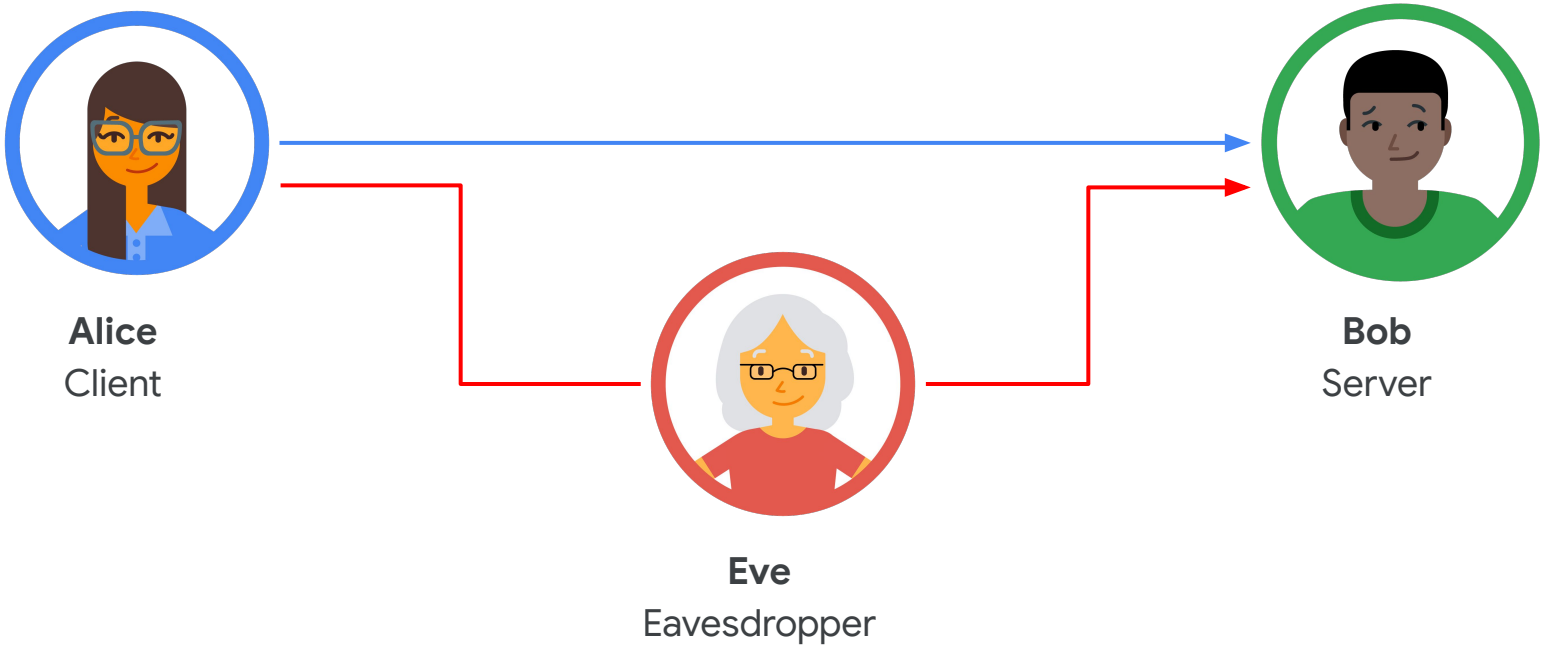
Request

Response

Bob
Server

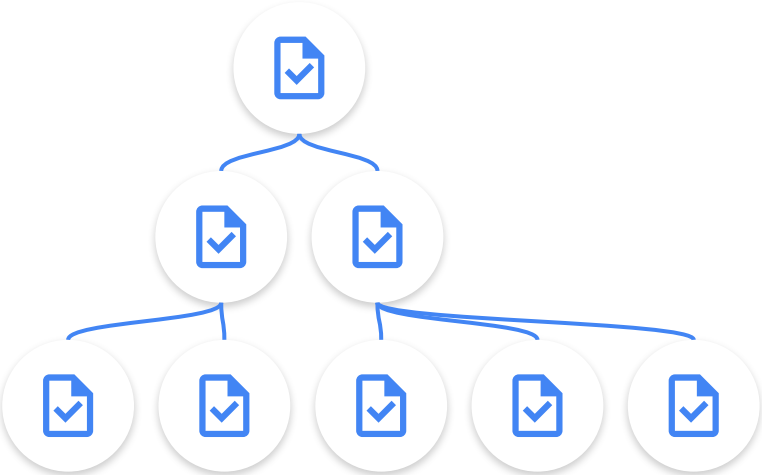
Google Cloud

Person-in-the-middle (MitM) attack



Google Cloud

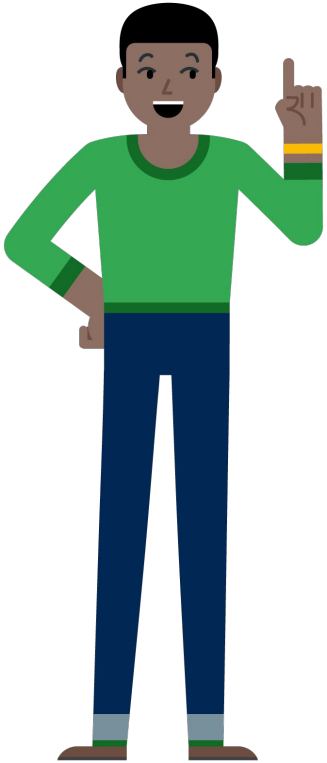
What's a Certificate Authority?
It's a trusted third party



Root cert: self-signed

Intermediate cert

Leaf cert



Google Cloud

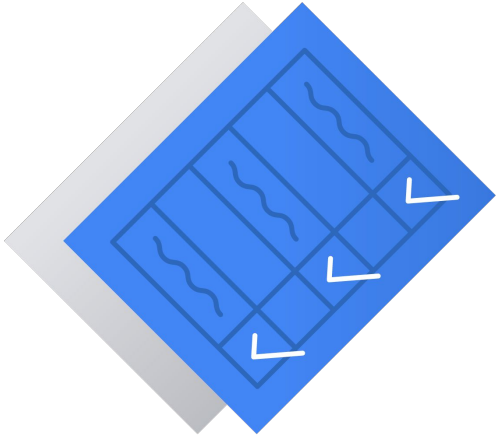
Private keys

- You generate it
- Stays private
- Can be used to sign things to attest identity

Certificates

- You or a CA signs it
- Is public and easily discoverable
- Can be used to verify identity

Kubernetes' Certificate Authority (CA)



Google Cloud

“The various Kubernetes components have a TON of different places where you can put in a certificate/certificate authority. When we were setting up a cluster **I felt like there were like 10 billion different command line arguments for certificates and keys and certificate authorities and I didn’t understand how they all fit together.**”

- Julia Evans
https://jvns.ca/blog/2017/08/05/how-kubernetes-certificates-work/

Everything needs a certificate!

Use kubeadm or generate and import the certificates yourself

PKI certificates and requirements

kubernetes.io/docs/setup/best-practices/certific...

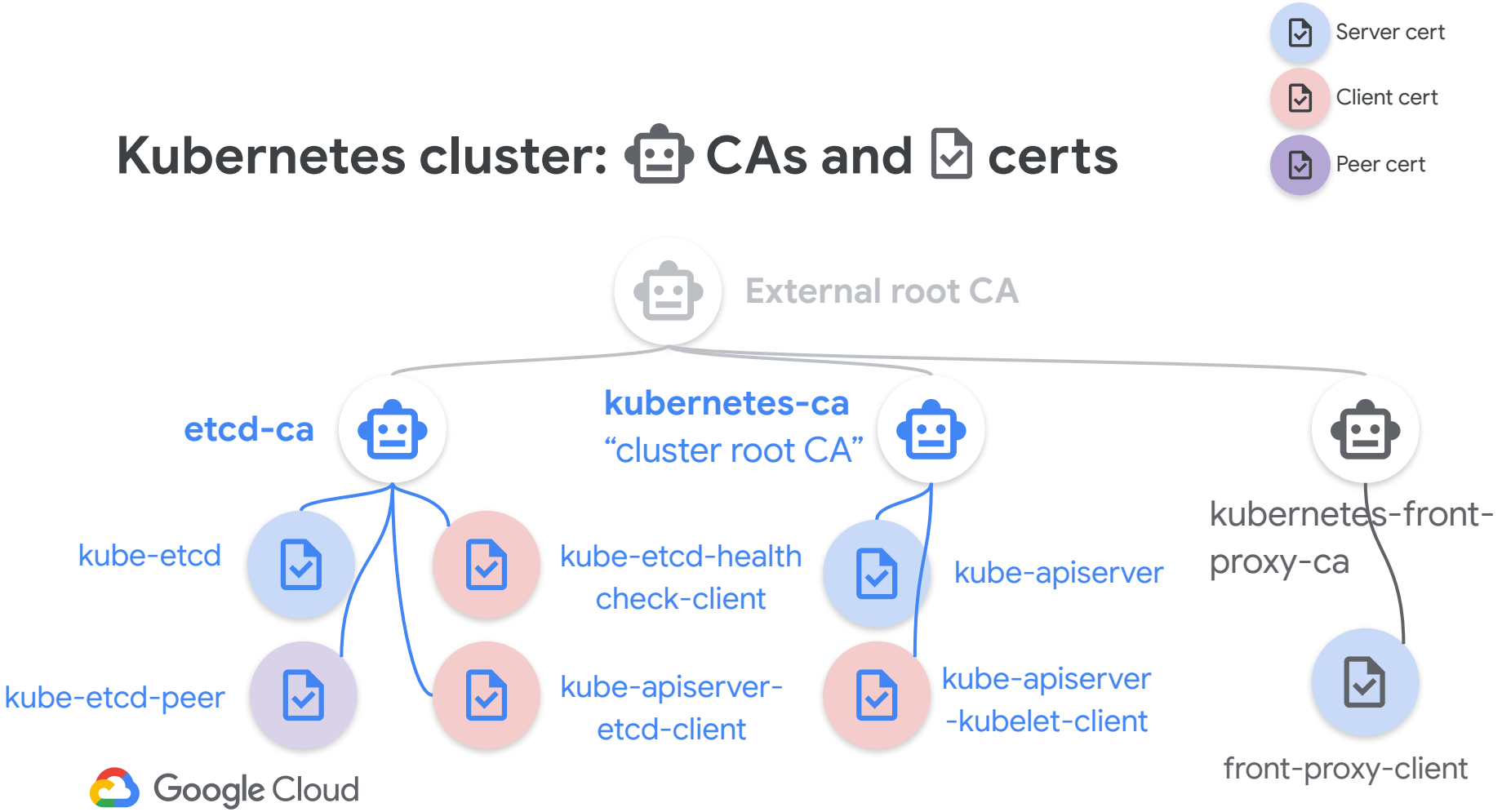
Certificate paths

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

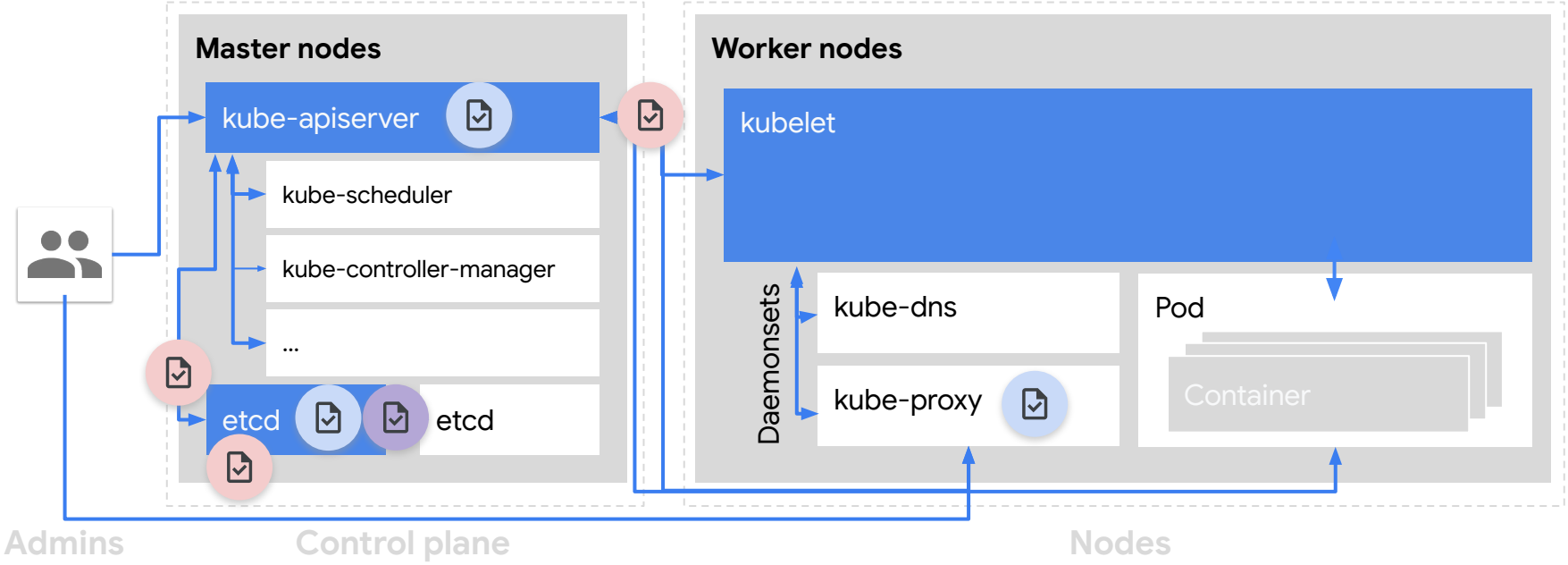
Default CN	recommended key path	recommended cert path	command	key argument	cert argument
etcd-ca	etcd/ca.key	etcd/ca.crt	kube-apiserver		–etcd-cafile
etcd-client	apiserver-etcd-client.key	apiserver-etcd-client.crt	kube-apiserver	–etcd-keyfile	–etcd-certfile
kubernetes-ca	ca.key	ca.crt	kube-apiserver		–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
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
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

Google Cloud

Kubernetes cluster: CAs and certs



Certs in your cluster



Google Cloud

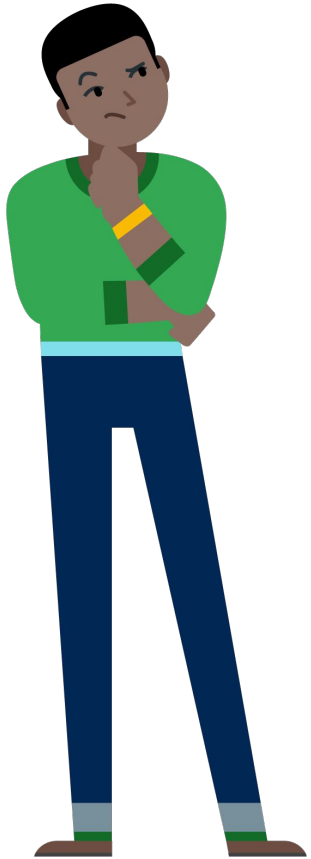
How a component gets a cert

certificates.k8s.io

Activate: `--cluster-signing-cert-file`
`--cluster-signing-key-file`

Certificate Signing Request (Beta in Kubernetes)

- Create request
- Send request to apiserver
- Approve request
- Download and use cert



Google Cloud

Kubelet certificate renewal and rotation

Default: Kubelet certs issued with 1 year expiration

To check certificate: (Stable in Kubernetes 1.15)

```
kubeadm alpha certs check-expiration
```

To renew certificate: (Stable in Kubernetes 1.15, default in 1.17 for node)

Automatic `kubeadm upgrade apply --certificate-renewal=true`

Manual `kubeadm alpha certs renew (--use-api)`

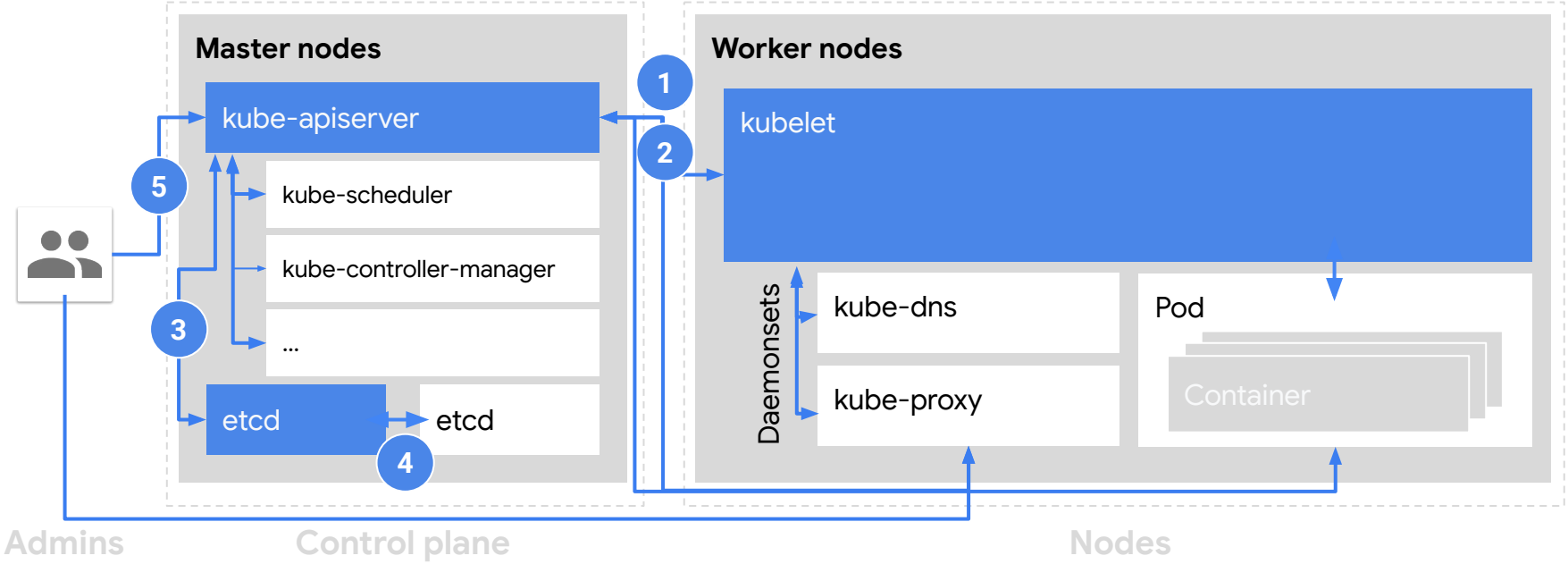
To set certificate rotation: (Beta in Kubernetes 1.8)

kubelet	`--rotate-certificates`
kube-controller-manager	`--experimental-cluster-signing-duration`

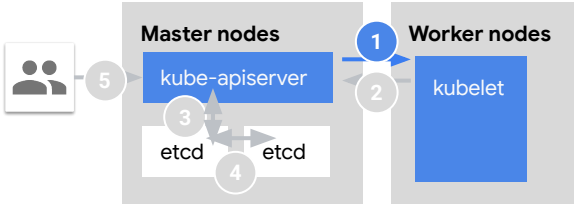
Protection of Kubernetes communications

Google Cloud

Communications in your cluster



Google Cloud



1 From the API server to the kubelet

Default Options

From the API server to the kubelet: Unauthenticated TLS

- `--kubelet-certificate-authority` to specify CA to verify kubelet's server certificate
- SSH tunnel (deprecated) - and still unauthenticated

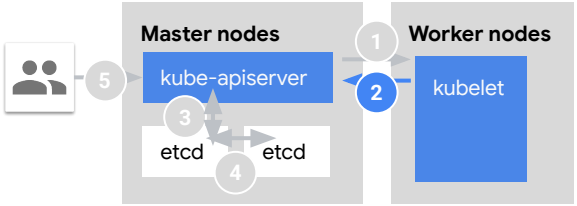
Authentication
Integrity
Encryption

From the API server to node, pod or service: Plain HTTP

- Shouldn't happen
- Specify HTTPs endpoint

Authentication
Integrity
Encryption

Google Cloud



2 From the kubelet to the API server

Default

Authentication

Integrity

Encryption

From the kubelet to the API server: Mutual TLS, if using Node Authorizer

- Requests over TLS
- apiserver listens on HTTPs port 443
- Node Authorizer authentication mode - kubelets use a credential in the system:nodes group

From the pod to the API server: Server-only authentication TLS, and client authenticates with bearer token

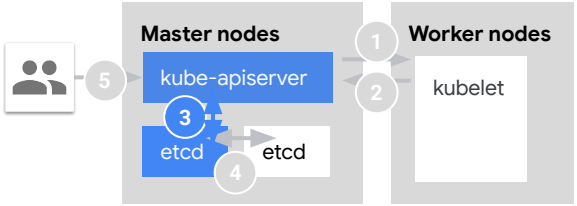
Authentication

Integrity

Encryption

Google Cloud

3 Between the API server and etcd



From the API server to etcd: Local host HTTP port 80

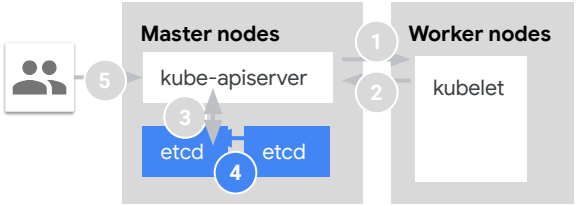
- mTLS with `--etcd-certfile` and `--etcd-keyfile`

From etcd to the API server: HTTPs port 443

Default	Options	
		Authentication
		Integrity
		Encryption
		Authentication
		Integrity
		Encryption

Google Cloud

4 Between two instances of etcd

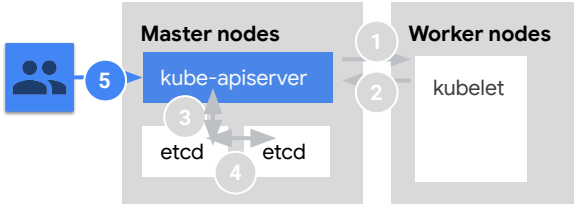


From an instance of etcd to a peer instance of etcd: Mutual TLS

Default

- Authentication
- Integrity
- Encryption

5 From the admin to the API server



From the admin to the API server:

Depends on your authentication method(s)!

- OAuth tokens
- x509 client certificates
- static passwords
- Authenticating proxy
- … don't forget about anonymous auth!

Options

- Authentication
- Integrity
- Encryption

Other connections

From a node to another node: Depends on your infrastructure

From a pod to another pod: Neither authenticated nor encrypted

- Restrict traffic with Network Policy
- Encrypt traffic with a service mesh like Istio

Summary

Authentication
Integrity
Encryption

Communications in your cluster - summary

	From	To	Default	What you should do	Options
1	API server	kubelet		`--kubelet-certificate-authority`	
	API server	nodes, pods, services		Specify HTTPs endpoint	
2	kubelet	API server		Use Node Authorizer	
	nodes, pods, services	API server			
3	API server	etcd	Local connection	`--etcd-certfile`, `--etcd-keyfile`	
	etcd	API server			
4	etcd	etcd			
5	Admin	API server		Don't allow anonymous authentication	

Google Cloud

Your options on GKE

Default

- Cluster root CA, etcd CA set up for you
- Cluster root CA certs have an expiration of 5 years
- API server to kubelet traffic is authentication in GKE v1.13+
- Certificate Signing Request API uses the cluster root CA, with automated approval of CSRs

Learn more: https://cloud.google.com/kubernetes-engine/docs/concepts/cluster-trust

Options

- Credential rotation
  https://cloud.google.com/kubernetes-engine/docs/how-to/credential-rotation
- Shielded GKE nodes: CSR bound to machine identity, protected by vTPM element
  https://cloud.google.com/kubernetes-engine/docs/how-to/shielded-gke-nodes
- Disable basic authentication and client certificates unless needed (disabled in v1.12+)

Kubernetes CIS v1.5.0 benchmark - cluster trust

1.2 API server

- 1.2.6 `--kubelet-certificate-authority`
- 1.2.29 --etcd-certfile and --etcd-keyfile
- 1.2.30 --tls-cert-file and --tls-private-key-file

1.3 Controller manager

- 1.3.6 RotateKubeletServerCertificate

2 etcd

- Basically this whole section

3.1 Authentication and authorization

- 3.1.1. Client certificate authentication should not be used for users

4 Kubelet

- 4.2.10 --tls-cert-file and --tls-private-key-file
- 4.2.11 --rotate-certificates
- 4.2.12 RotateKubeletServerCertificate

Google Cloud

Best practices

- Set up your cluster's CAs using kubeadm, if not using a managed service
- Rotate your certs!
- For specific paths in your cluster,
  - Specify a kubelet CA to authenticate the API server to the kubelet
  - Use Node Authorizer for kubelet to API server authentication

On GKE:

- Use Shielded GKE Nodes
- Perform a credential rotation for your cluster root CA

Google Cloud

Learn more

For Kubernetes certificates:

- https://kubernetes.io/docs/setup/best-practices/certificates/
- https://jvns.ca/blog/2017/08/05/how-kubernetes-certificates-work/
- https://github.com/kelseyhightower/kubernetes-the-hard-way/blob/master/docs/04-certificate-authority.md

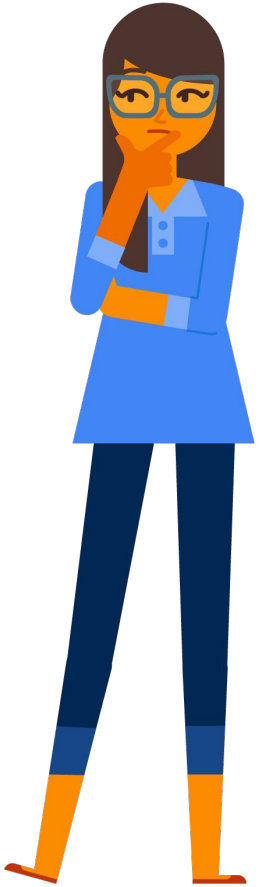
For GKE cluster trust:

- https://cloud.google.com/kubernetes-engine/docs/concepts/cluster-trust

For container security on GKE:

- cloud.google.com/containers/security
- g.co/gke/security
- g.co/gke/hardening

Q&A



Google Cloud


KubeCon
CloudNativeCon
North America 2019