TLS

Table of Contents

1 SYMMETRIC ENCRYPTION

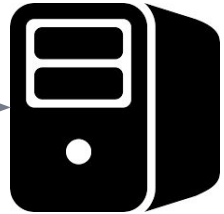
SYMMETRIC ENCRYPTION

User: James
Password: Pp123

http://clarus-commerce.com

SYMMETRIC ENCRYPTION

User: James
Password: Pp123



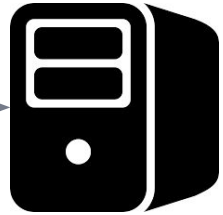
http://clarus-commerce.com

SYMMETRIC ENCRYPTION

User: James
Password: Pp123

User: James
Password: Pp123

User: James
Password: Pp123

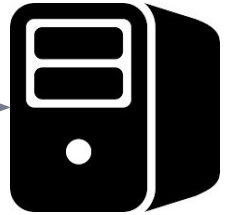


http://clarus-commerce.com

SYMMETRIC ENCRYPTION

User: James
Password: Pp123

DKFMGHLM98DSF89
3YH840FASLO142TU



http://clarus-commerce.com

SYMMETRIC ENCRYPTION

User: amFtZXMK
Password: UHAxMjMK

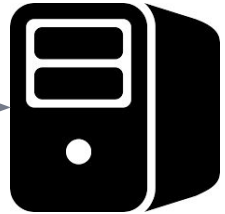
http://clarus-commerce.com

SYMMETRIC ENCRYPTION

User: amFtZXMK
Password: UHAxMjMK

User: amFtZXMK
Password: UHAxMjMK

User: amFtZXMK
Password: UHAxMjMK



http://clarus-commerce.com

User: amFtZXMK
Password: UHAxMjMK

User: amFtZXMK
Password: UHAxMjMK

http://clarus-commerce.com

SYMMETRIC ENCRYPTION

Symmetric encryption is a type of **encryption** where only one **key** (a secret **key**) is used to both **encrypt** and **decrypt** electronic information. The entities communicating via **symmetric encryption** must exchange the **key** so that it can be used in the decryption process.

2 ASYMMETRIC ENCRYPTION

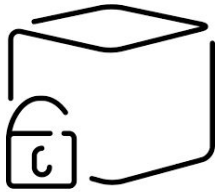
ASYMMETRIC ENCRYPTION

Asymmetric Encryption, also known as **Public-Key Cryptography**, is an example of encryption method. Unlike **symmetric encryption**, Asymmetric Encryption encrypts and decrypts the data using **two separate** yet mathematically connected **cryptographic keys.** These keys are known as a **Public Key** and a **Private Key**. Together, they're called a **Public and Private Key Pair**.

ASYMMETRIC ENCRYPTION

To understand well, we symbolize public key as a lock box.

ASYMMETRIC ENCRYPTION

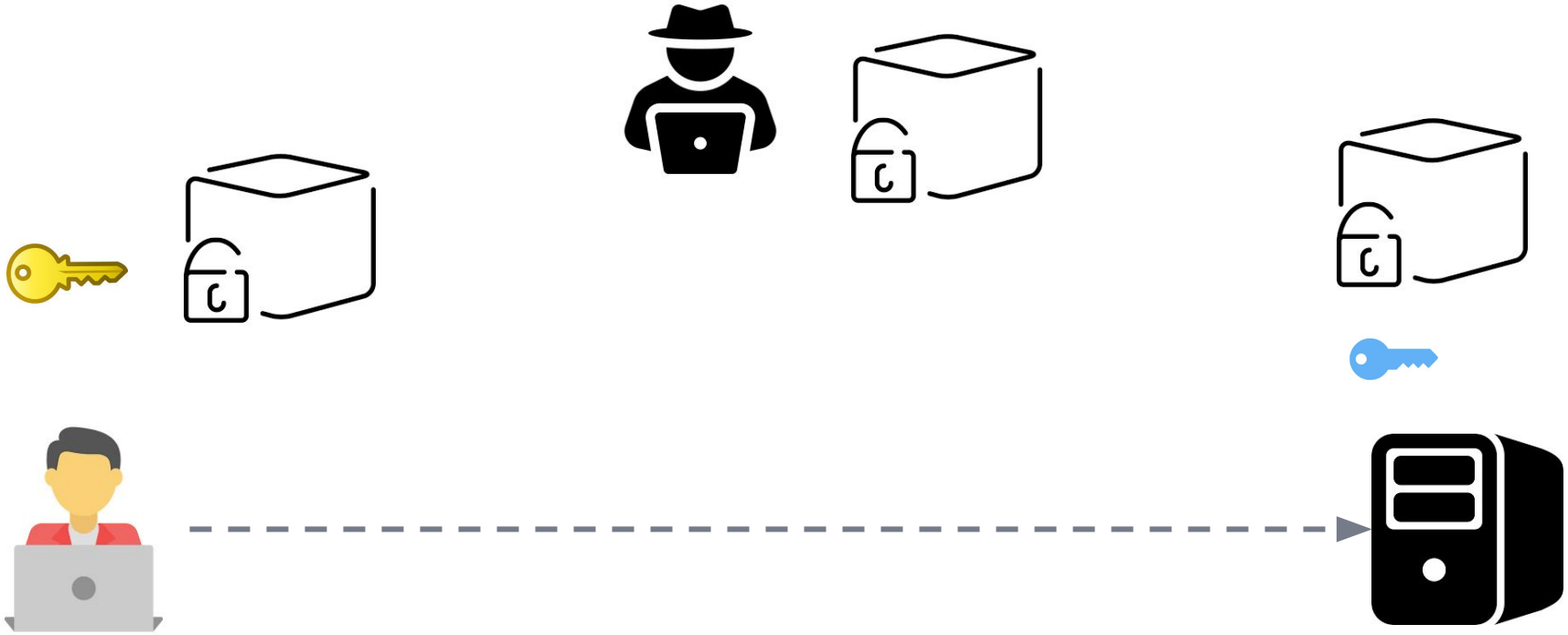


https://clarus-commerce.com

ASYMMETRIC ENCRYPTION

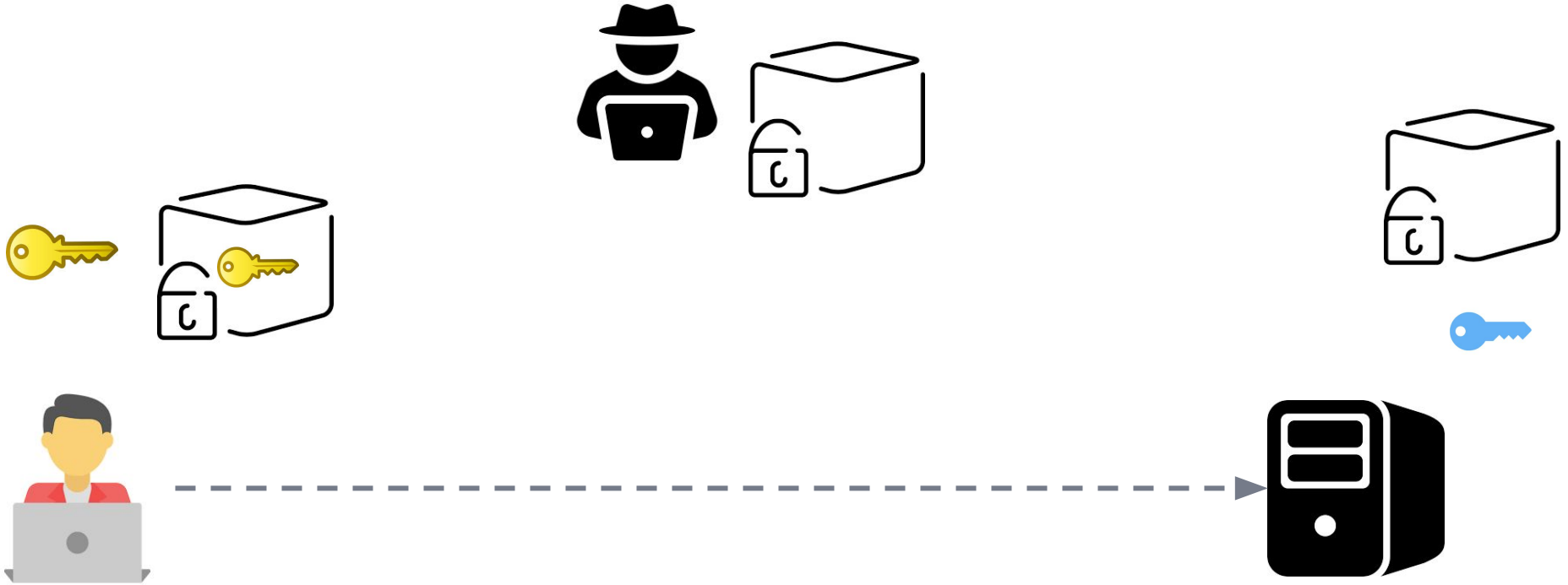
https://clarus-commerce.com

ASYMMETRIC ENCRYPTION



https://clarus-commerce.com

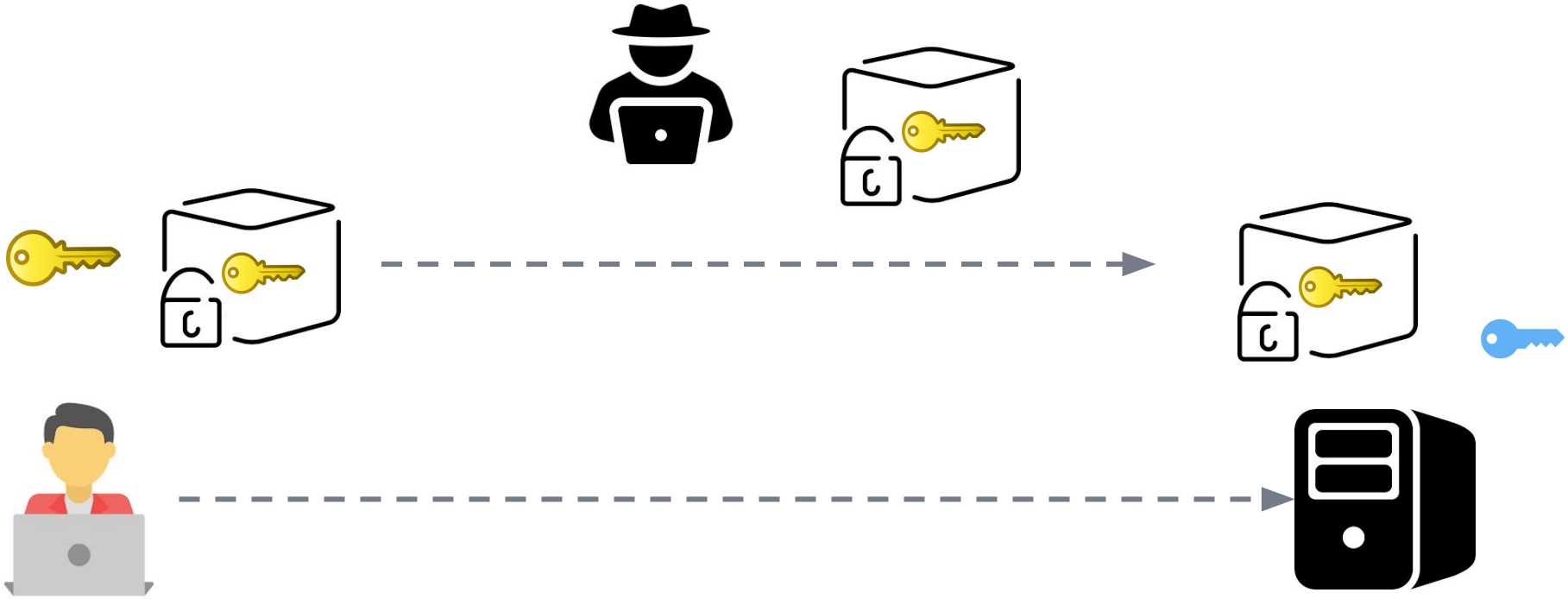
ASYMMETRIC ENCRYPTION



https://clarus-commerce.com

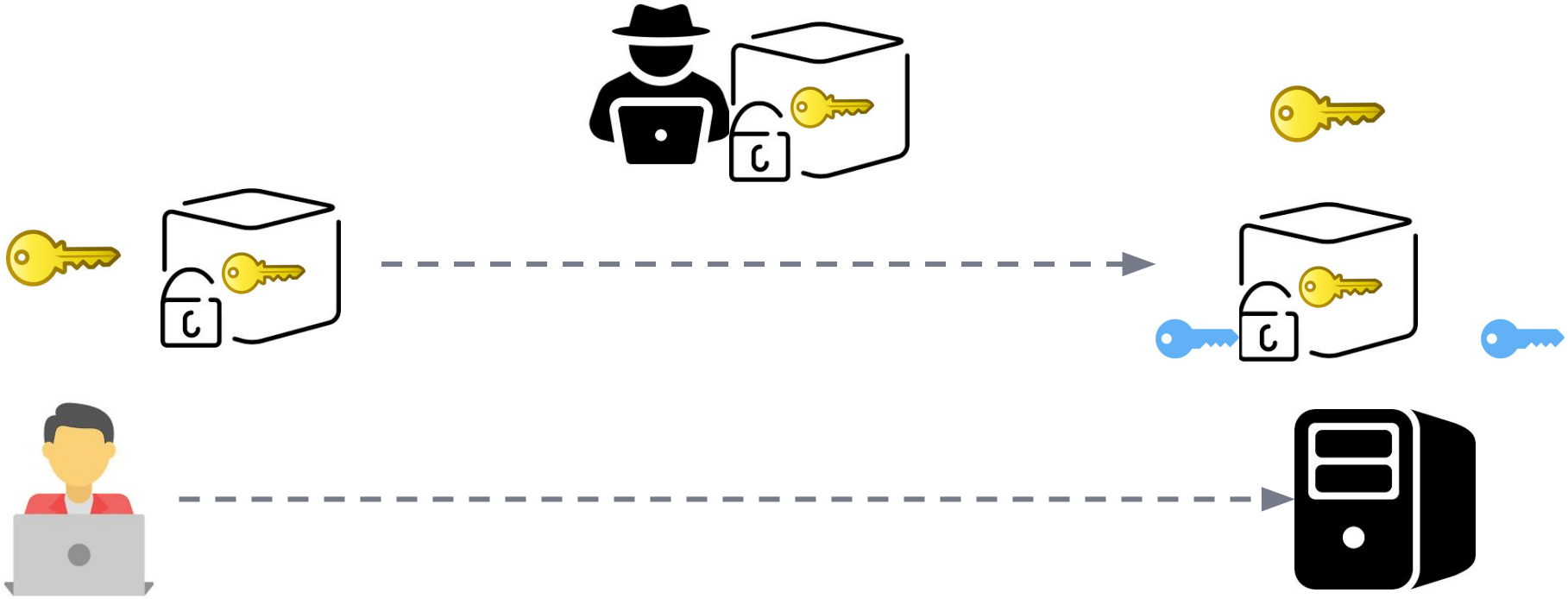
We make our private key secure with server public key

ASYMMETRIC ENCRYPTION



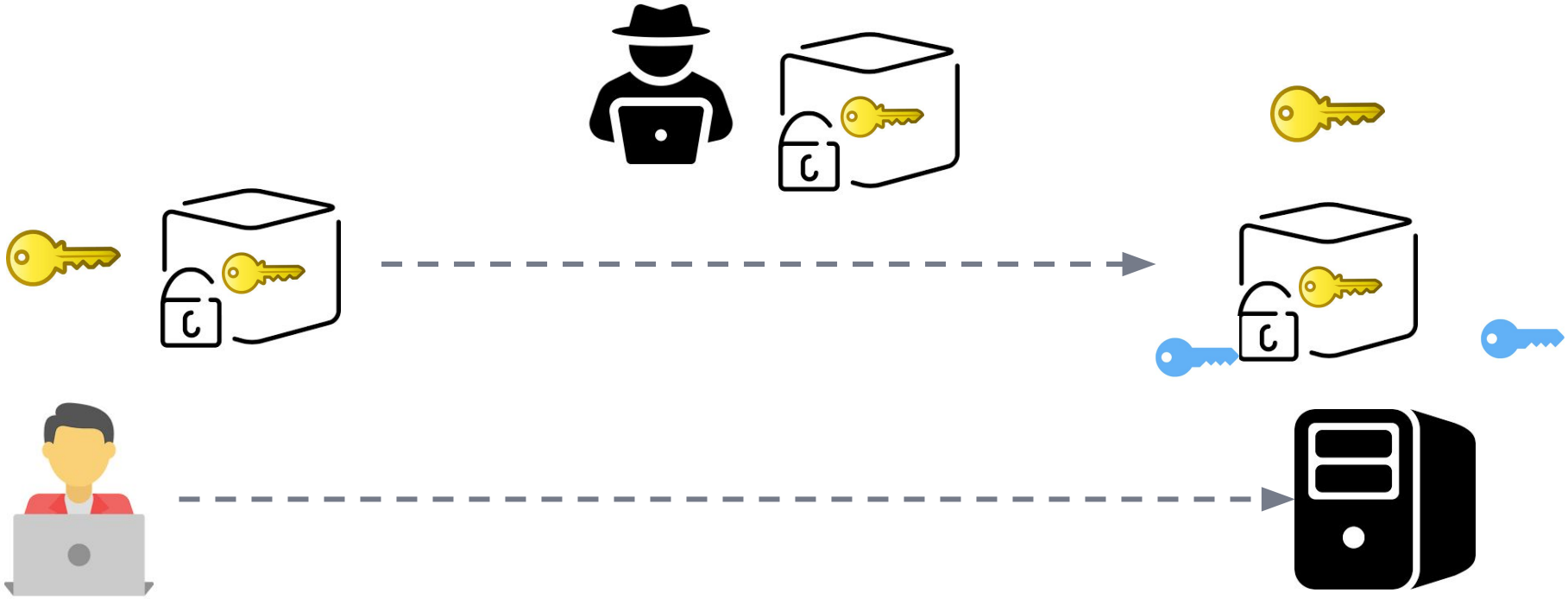
https://clarus-commerce.com

ASYMMETRIC ENCRYPTION



https://clarus-commerce.com

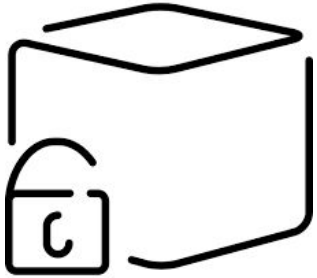
ASYMMETRIC ENCRYPTION



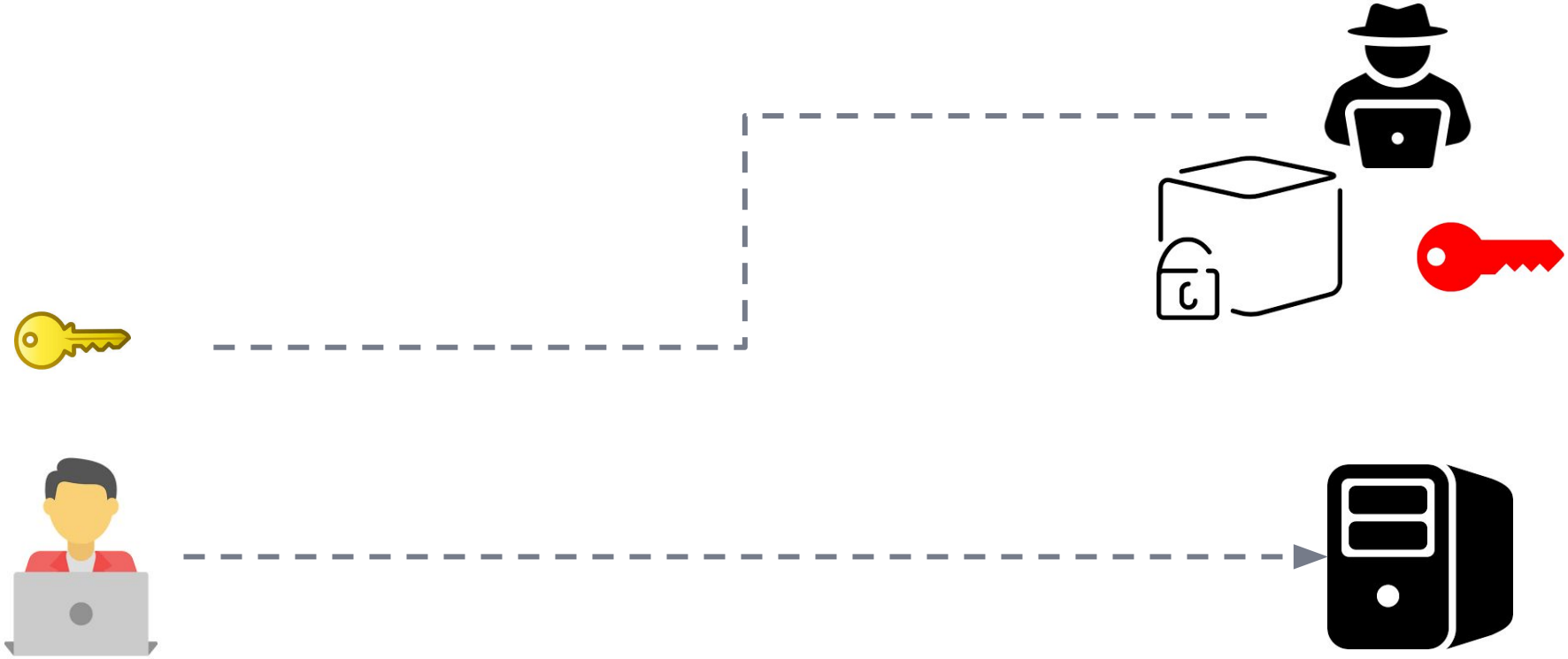
https://clarus-commerce.com

Server decrypt our private key with server private key. Hacker doesn't have private key of server. So the hacker couldn't get our private key.

ASYMMETRIC ENCRYPTION

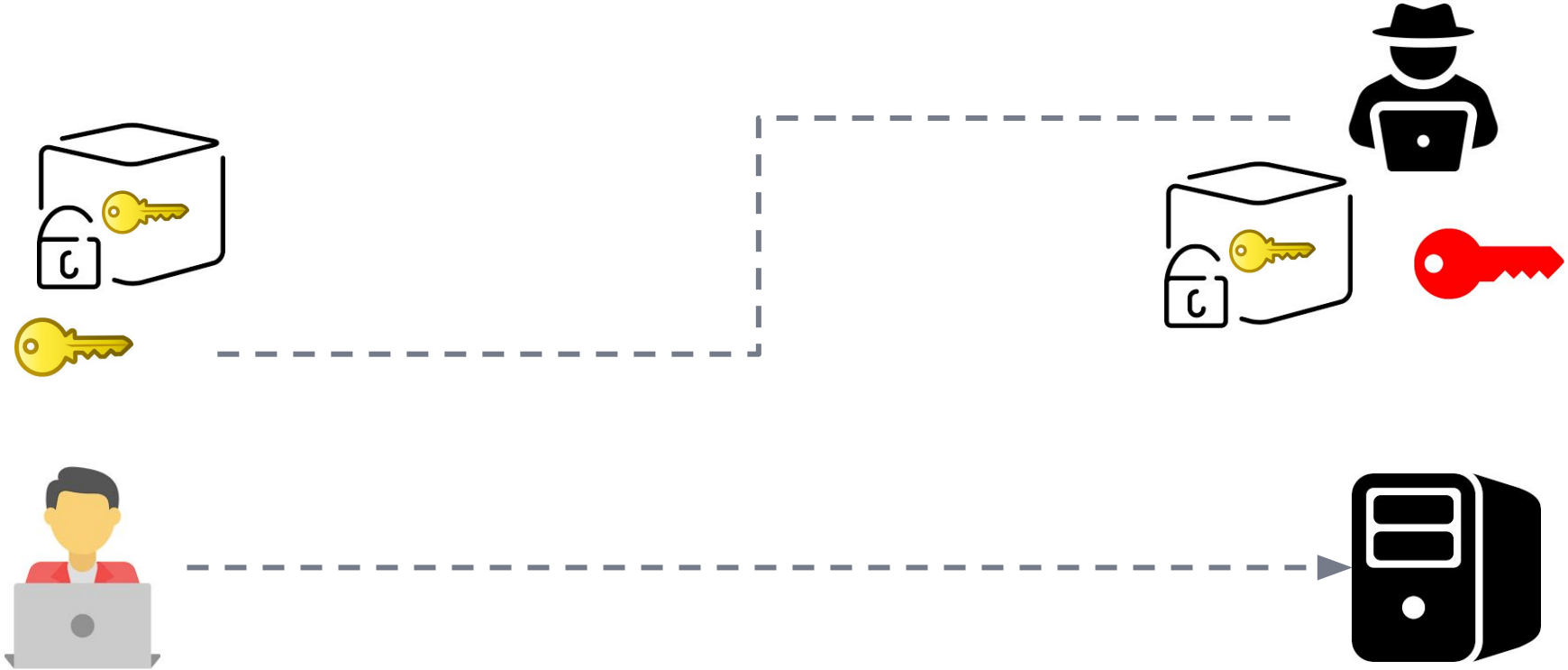


ASYMMETRIC ENCRYPTION



Hacker can allure and trick us. Hacker can get our private key.

ASYMMETRIC ENCRYPTION



Hacker can allure and trick us. Hacker can get our private key.

3

TLS/SSL CERTIFICATE

TLS/SSL CERTIFICATE

Transport Layer Security is a **protocol** that establishes an **encrypted session** between two computers on the Internet. It verifies the identity of the server and prevents hackers from intercepting any data.

TLS/SSL CERTIFICATE

What is a TLS certificate?

- **Digital certificates**, also known as identity certificates or public key certificates, are **digital files** that are used to **certify the ownership of a public key.**
- TLS certificates are a type of digital certificate, issued by a **Certificate Authority (CA)**.
- **The CA signs the certificate**, certifying that they have verified that it belongs to the owners of the domain name which is the subject of the certificate.

How does a TLS certificate work?

1. We request a CSR(Certificate Signing Request) from any CA (Certificate Authority).
2. CA validate our request.
3. CA send signed certificate to us.

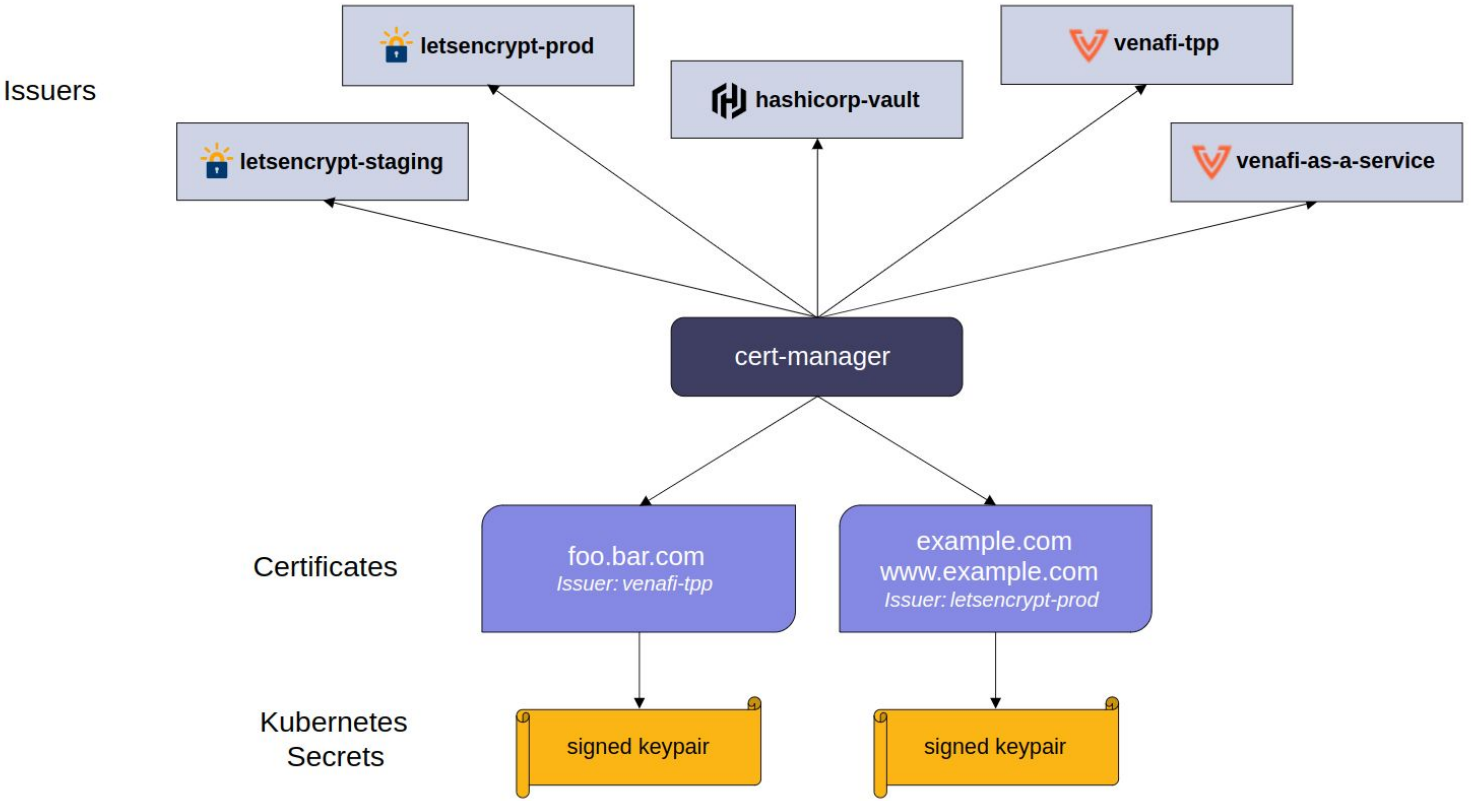
How does a browser know TLS certificate is valid?

TLS/SSL CERTIFICATE

TLS certificates usually contain the following information:

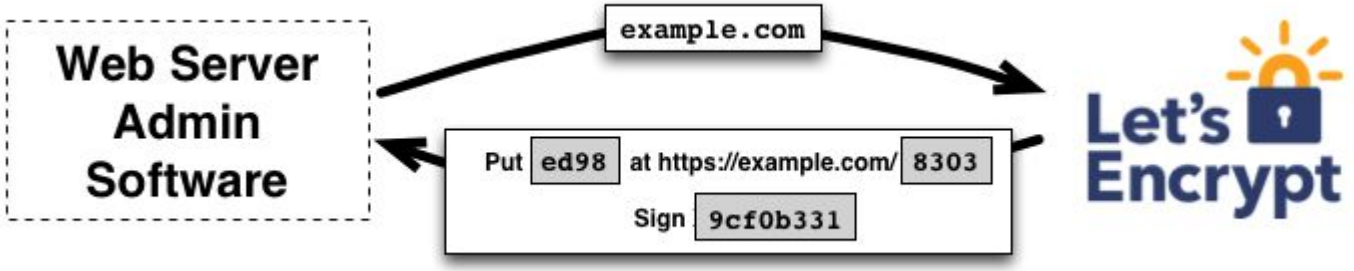
- The subject domain name
- The subject organization
- The name of the issuing CA
- Additional or alternative subject domain names, including subdomains, if any
- Issue date
- Expiry date
- The public key (The private key, however, is a secret.)
- **The digital signature by the CA**

TLS/SSL CERTIFICATE



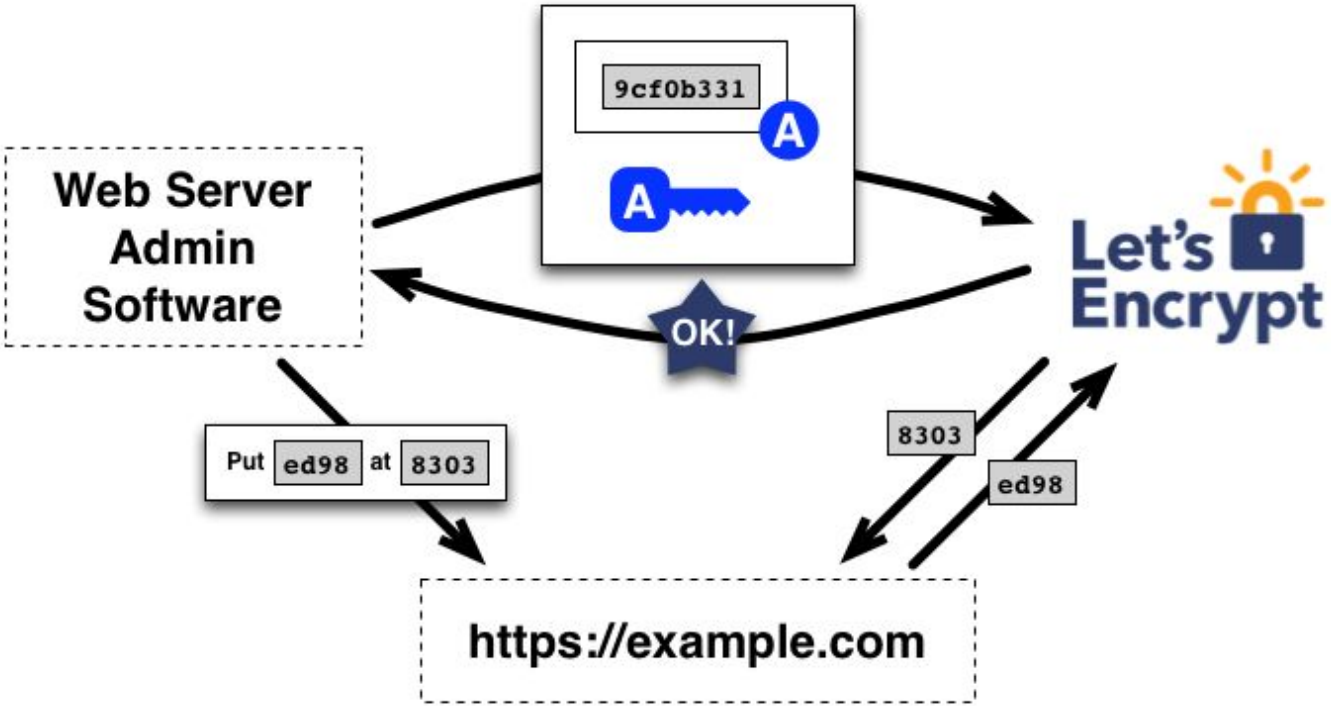
TLS/SSL CERTIFICATE

Applicant **Certificate Authority (CA)**



TLS/SSL CERTIFICATE

Applicant **Certificate Authority (CA)**



THANKS!

Any questions?

You can find me at:

- james@clarusway.com

TLS/SSL CERTIFICATE

TLS/SSL CERTIFICATE

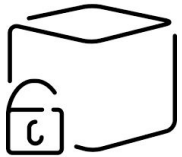
How does a TLS certificate work?

- When a user tries to connect to a server, the server sends them its TLS certificate.
- The user then verifies the server's certificate using CA certificates that are present on the user's device to establish a secure connection.
- This verification process uses public key cryptography, such as RSA or ECC, to prove the CA signed the certificate.
- As long as you trust the CA, this demonstrates you are communicating with the server certificate's subject

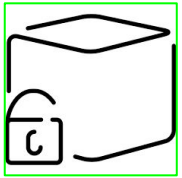
How does a TLS certificate work?

1. We request a CSR(Certificate Signing Request) from any CA (Certificate Authority).
2. CA validate our request.
3. CA send signed certificate to us.

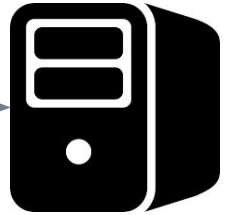
How does a browser know TLS certificate is valid?



CA public Key



CA public Key



Browsers has built-in CA's public keys.

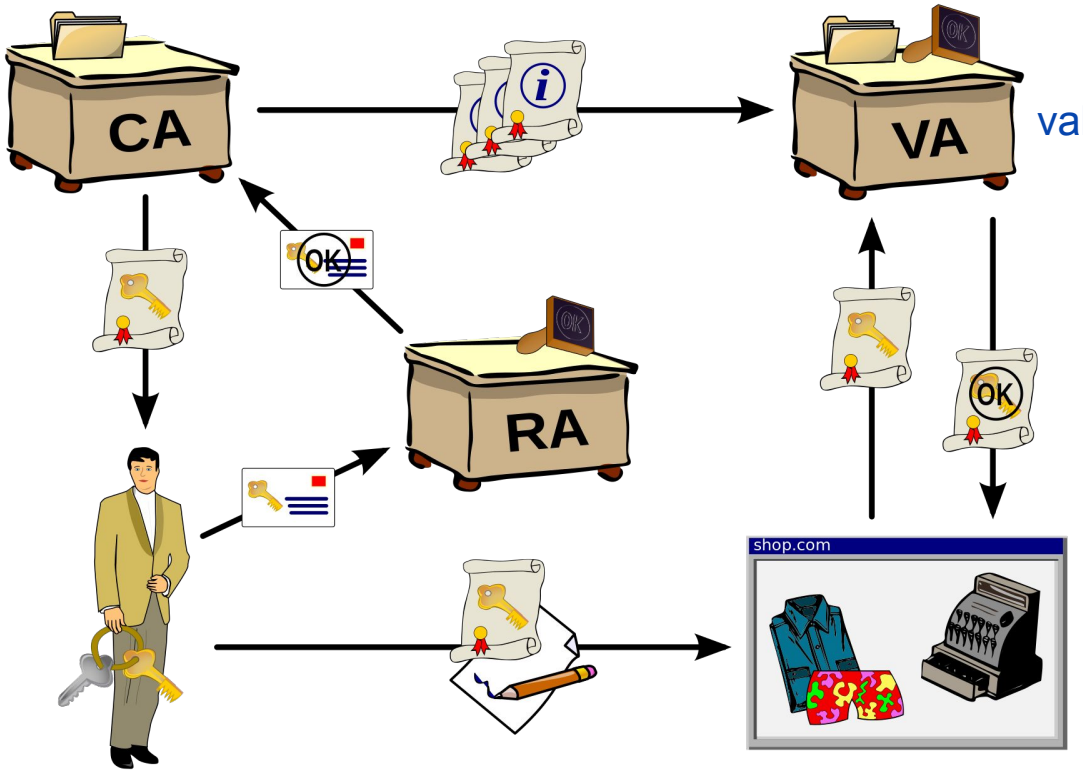
http://clarus-commerce.com

A public key infrastructure (PKI)

A public key infrastructure (PKI) is *a set of roles, policies, hardware, software and procedures needed to create, manage, distribute, use, store and revoke digital certificates and manage public-key encryption*.

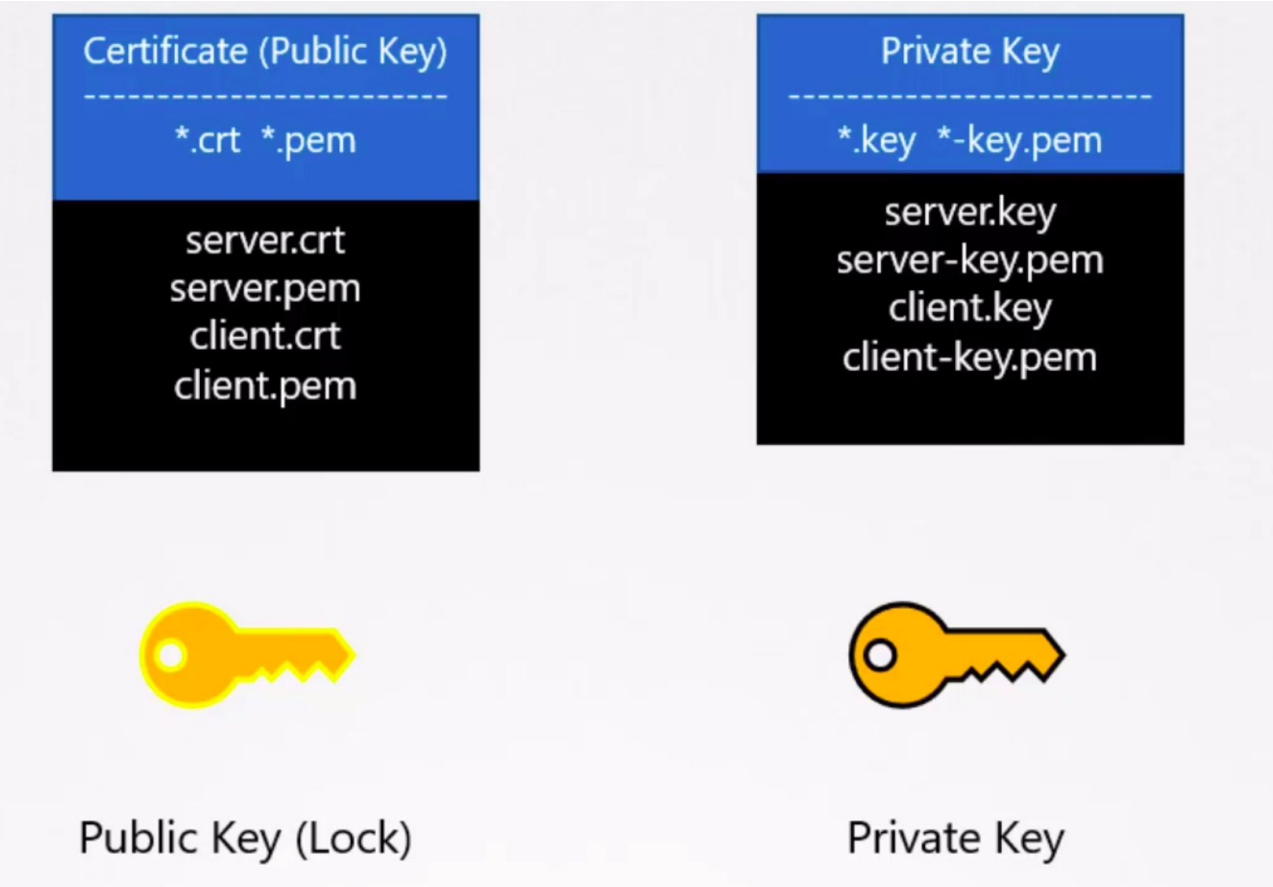
The purpose of a PKI is to facilitate the secure electronic transfer of information for a range of network activities such as e-commerce, internet banking and confidential email. It is required for activities where simple passwords are an inadequate authentication method and more rigorous proof is required to confirm the identity of the parties involved in the communication and to validate the information being transferred.

A public key infrastructure (PKI)

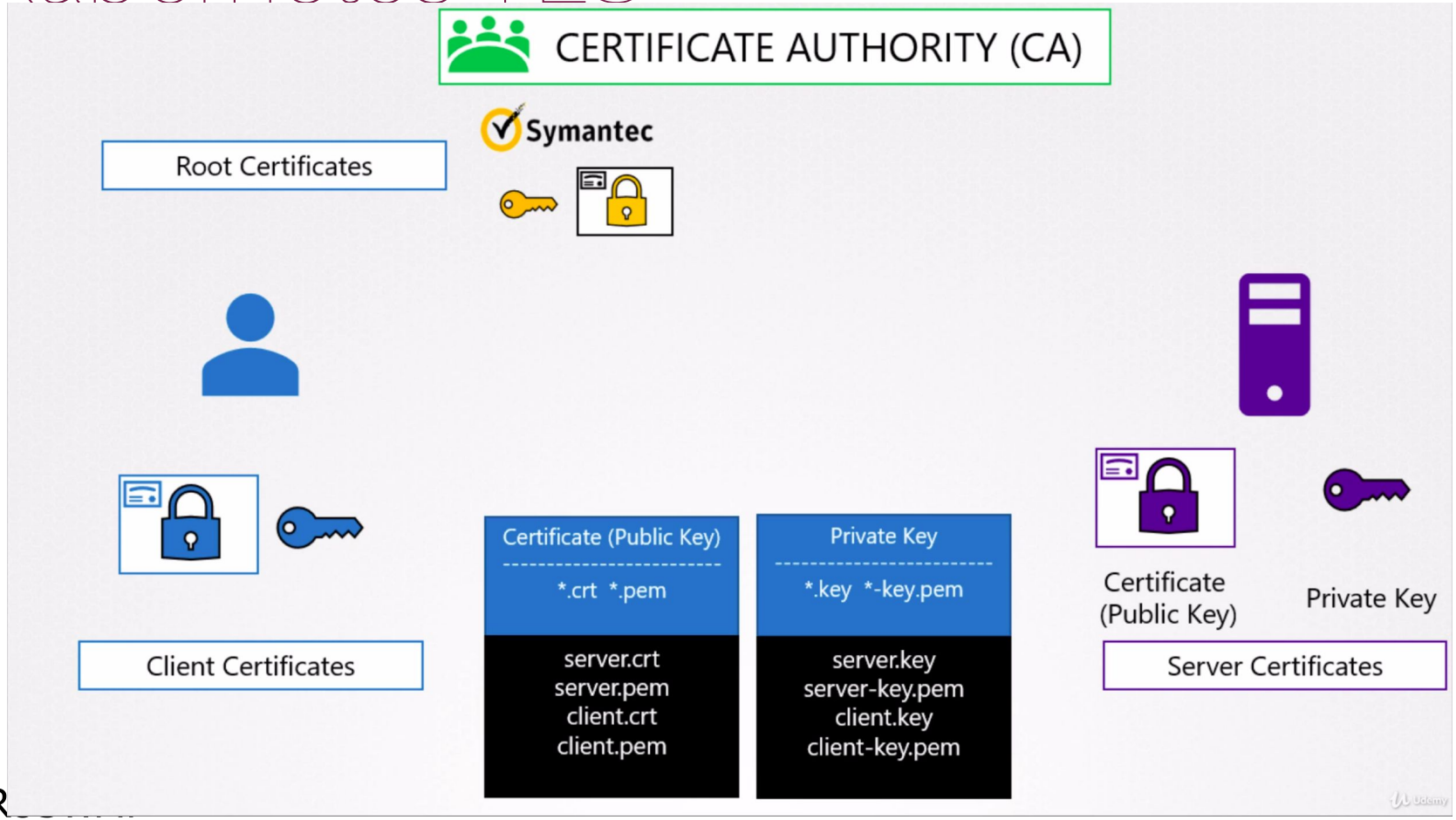


validation authority (VA)

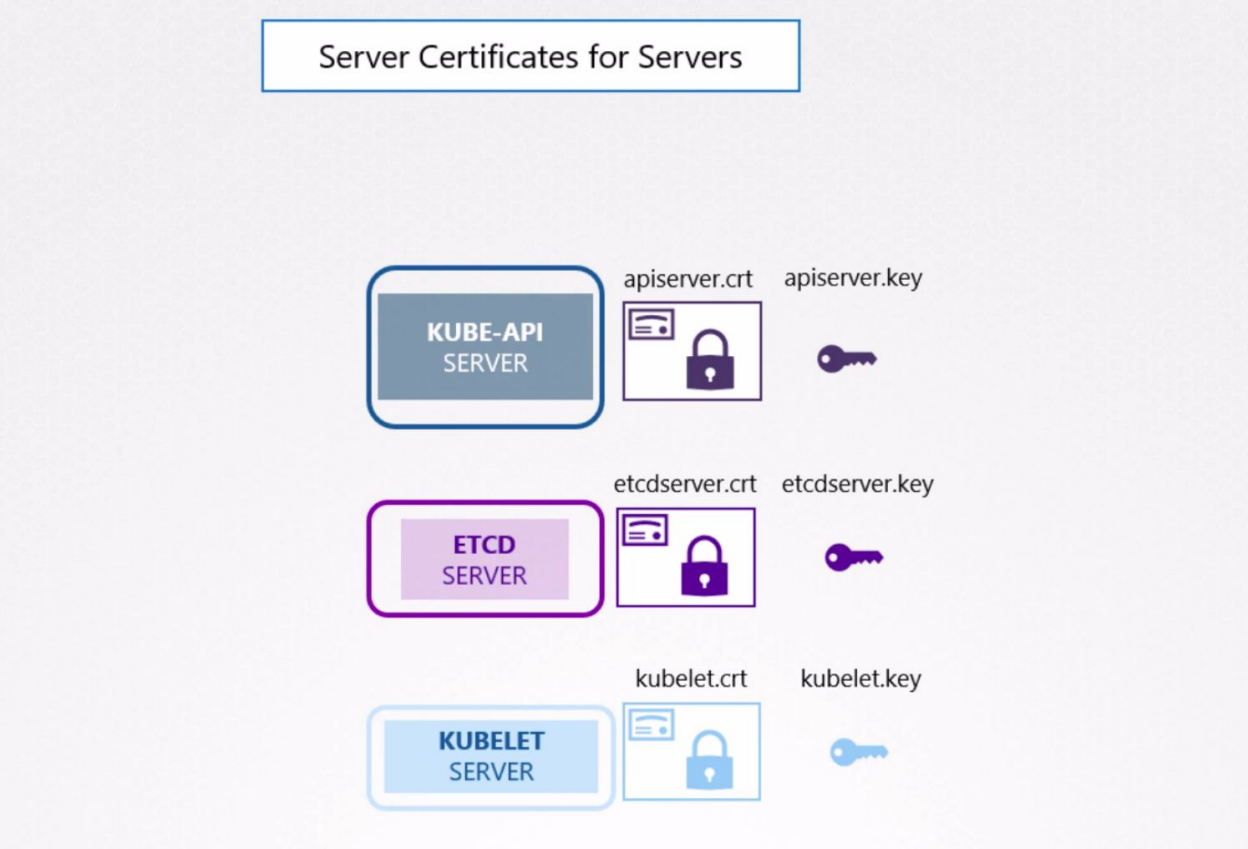
registration authority (RA)


Certificate (Public Key)
*.crt *.pem
server.crt
server.pem
client.crt
client.pem
Private Key
*.key *-key.pem
server.key
server-key.pem
client.key
client-key.pem
Public Key (Lock)
Private Key

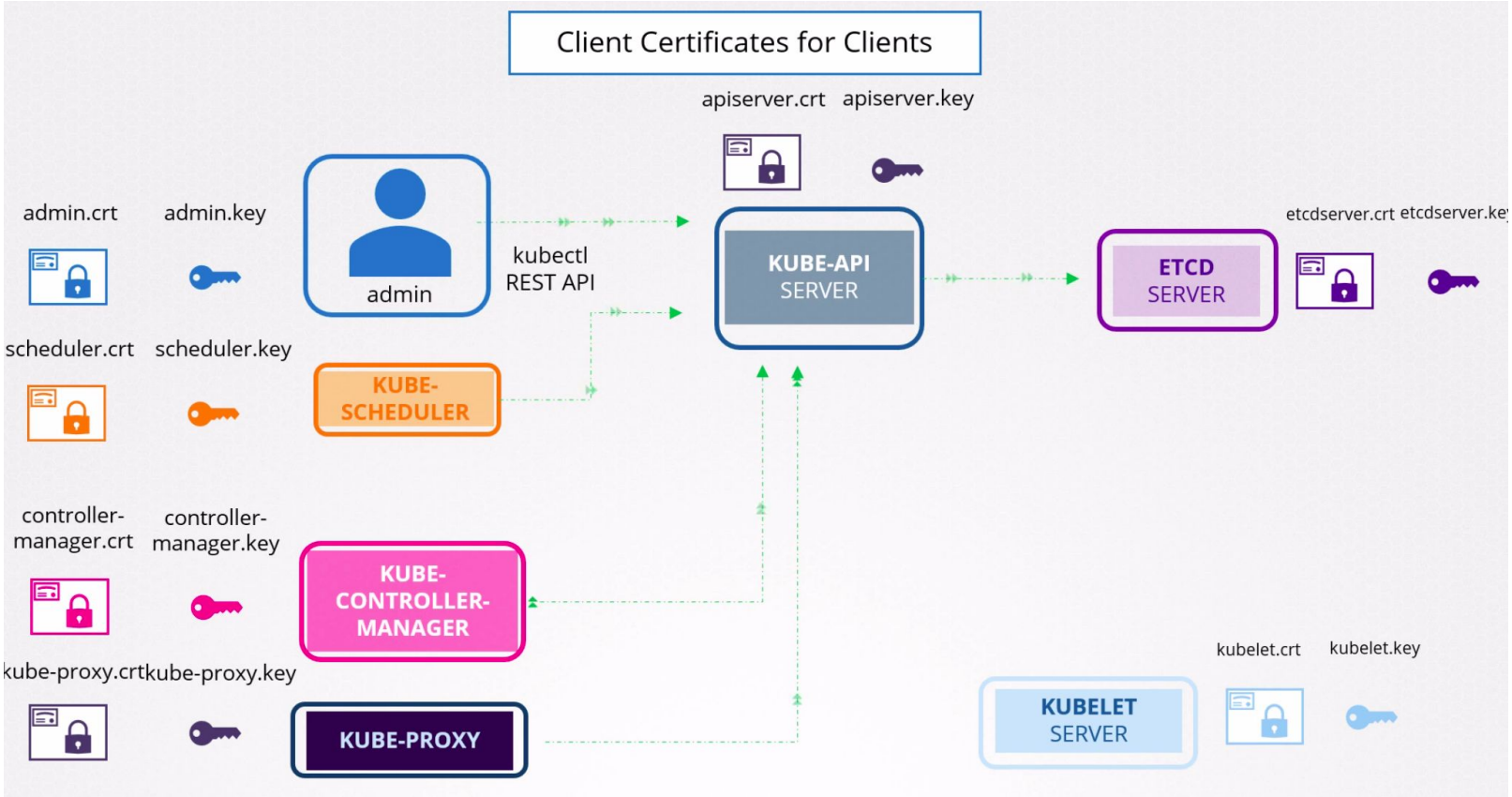
Kubernetes TLS



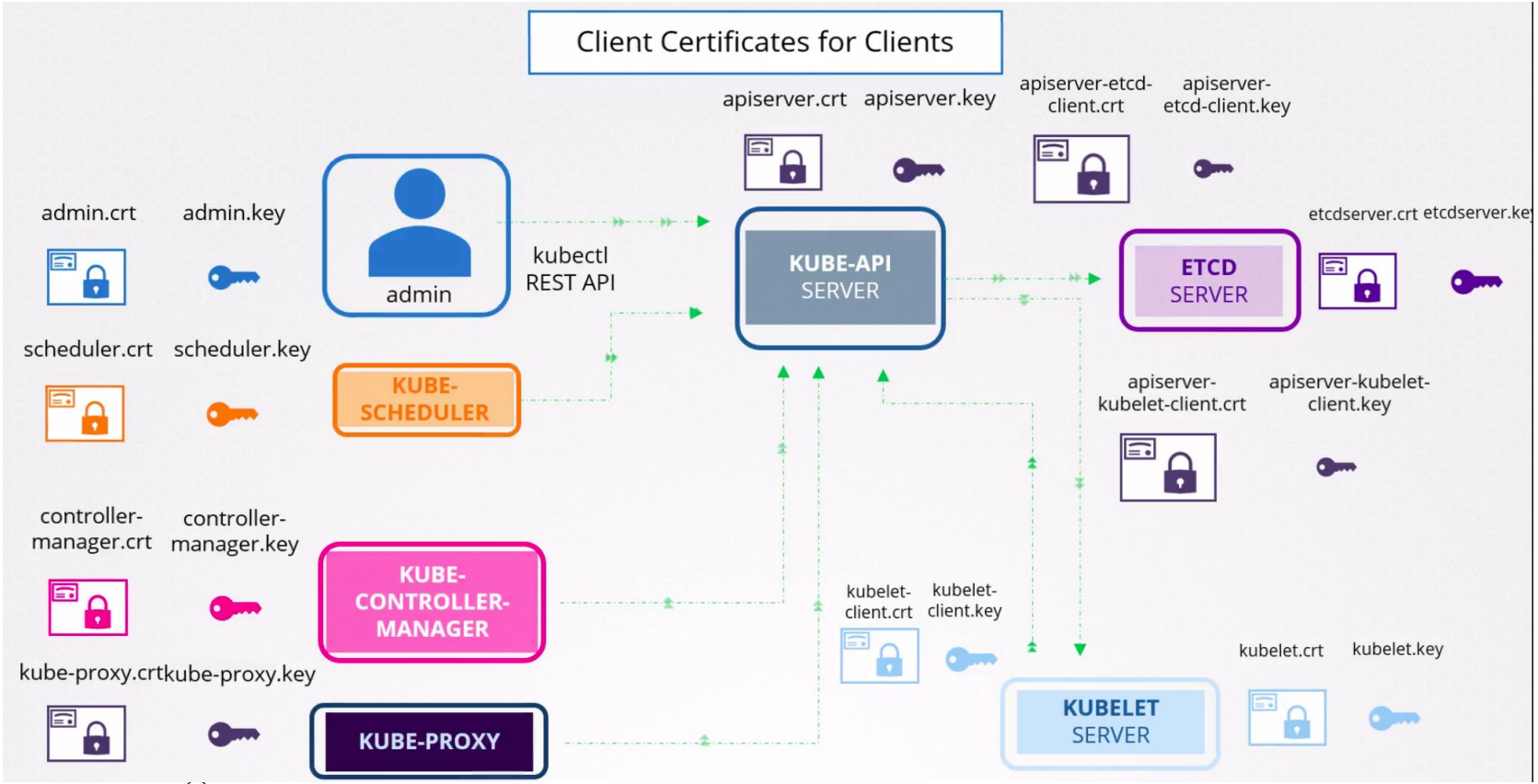
Kubernetes TLS



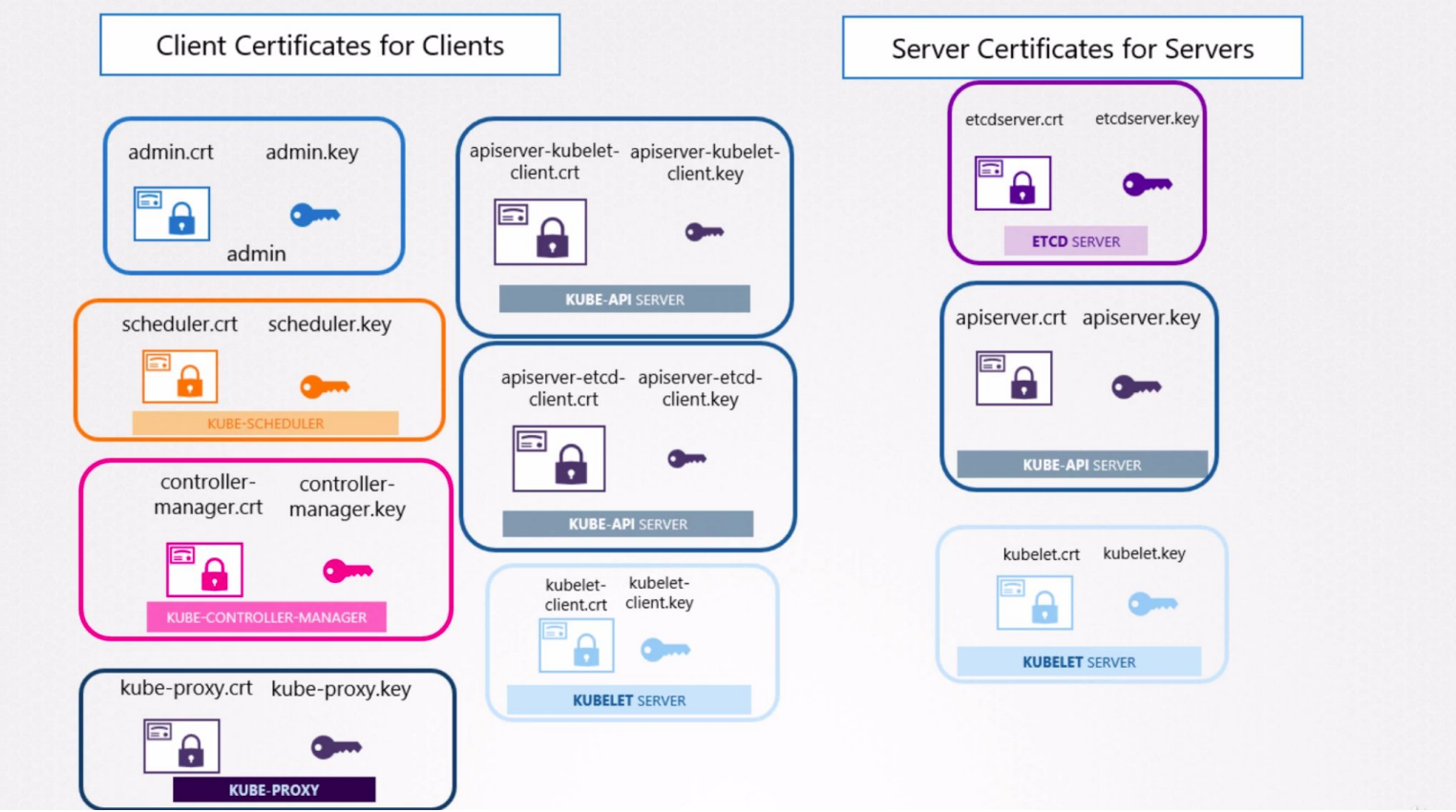
Kubernetes TLS

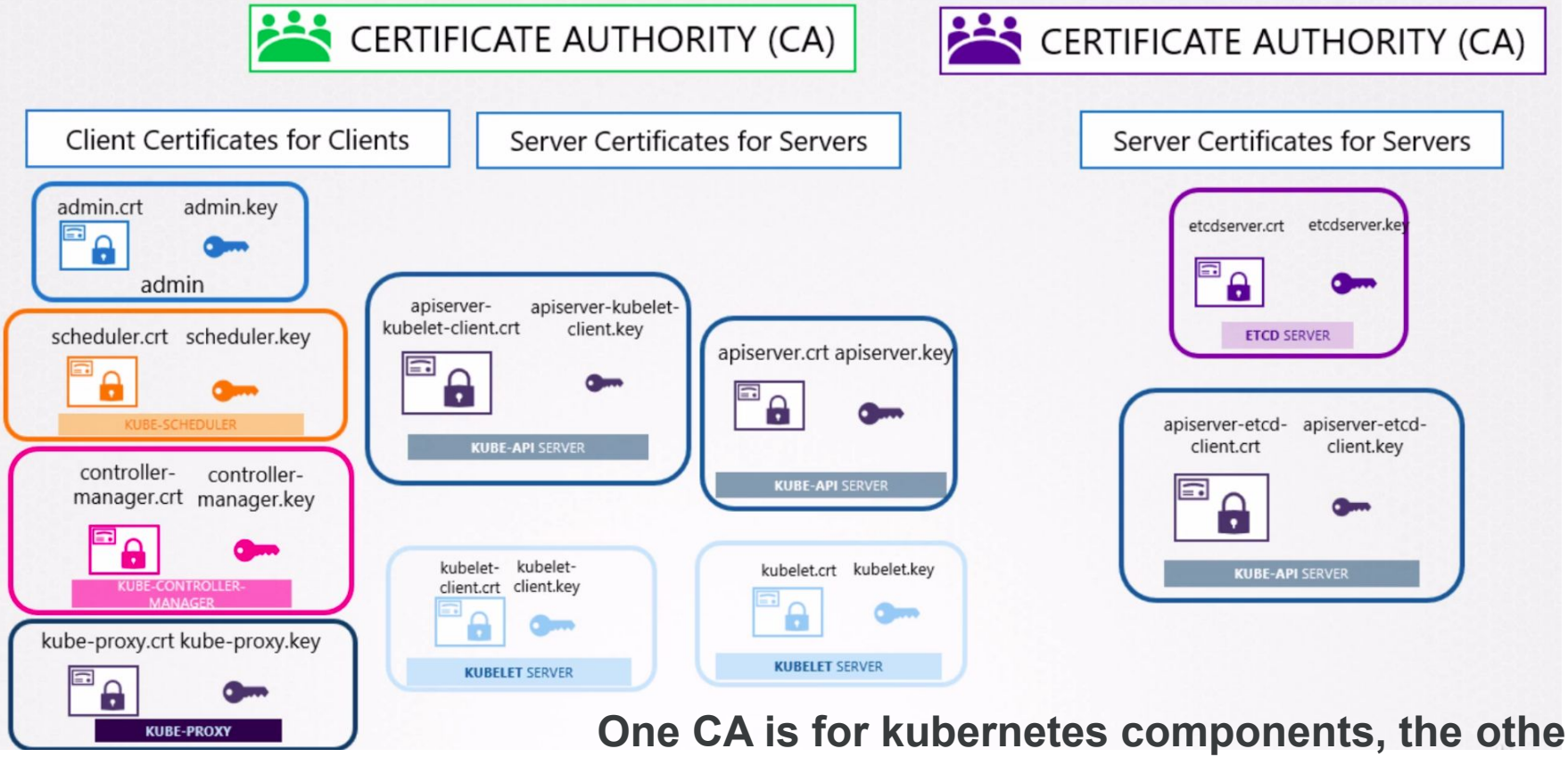


Kubernetes TLS



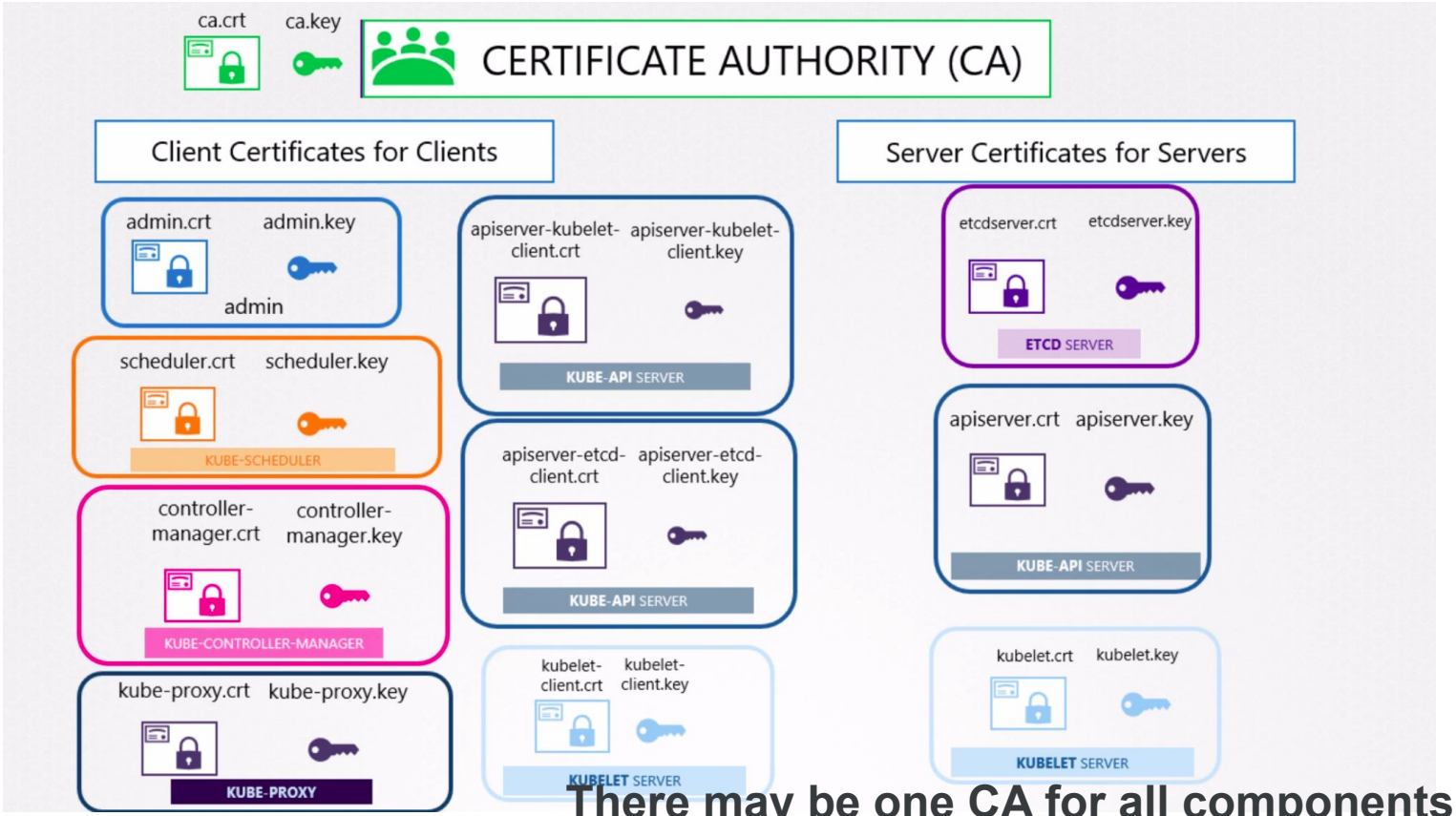
Kubernetes TLS





One CA is for kubernetes components, the other CA is just for ETCD.

Kubernetes TLS



There may be one CA for all components.

Kubernetes TLS

Certificate Authority (CA)

1. Create a ca.key.
2. Create a ca.csr from ca.key.
3. Create a ca.crt from ca.csr and ca.key.

CERTIFICATE AUTHORITY (CA)

Generate Keys

ca.key

```
openssl genrsa -out ca.key 2048
ca.key
```

Certificate Signing Request

ca.csr

```
openssl req -new -key ca.key -subj  "/CN=KUBERNETES-CA" -out ca.csr
ca.csr
```

Sign Certificates

ca.crt

```
openssl x509 -req -in ca.csr -signkey ca.key -out ca.crt
ca.crt
```

Kubernetes TLS

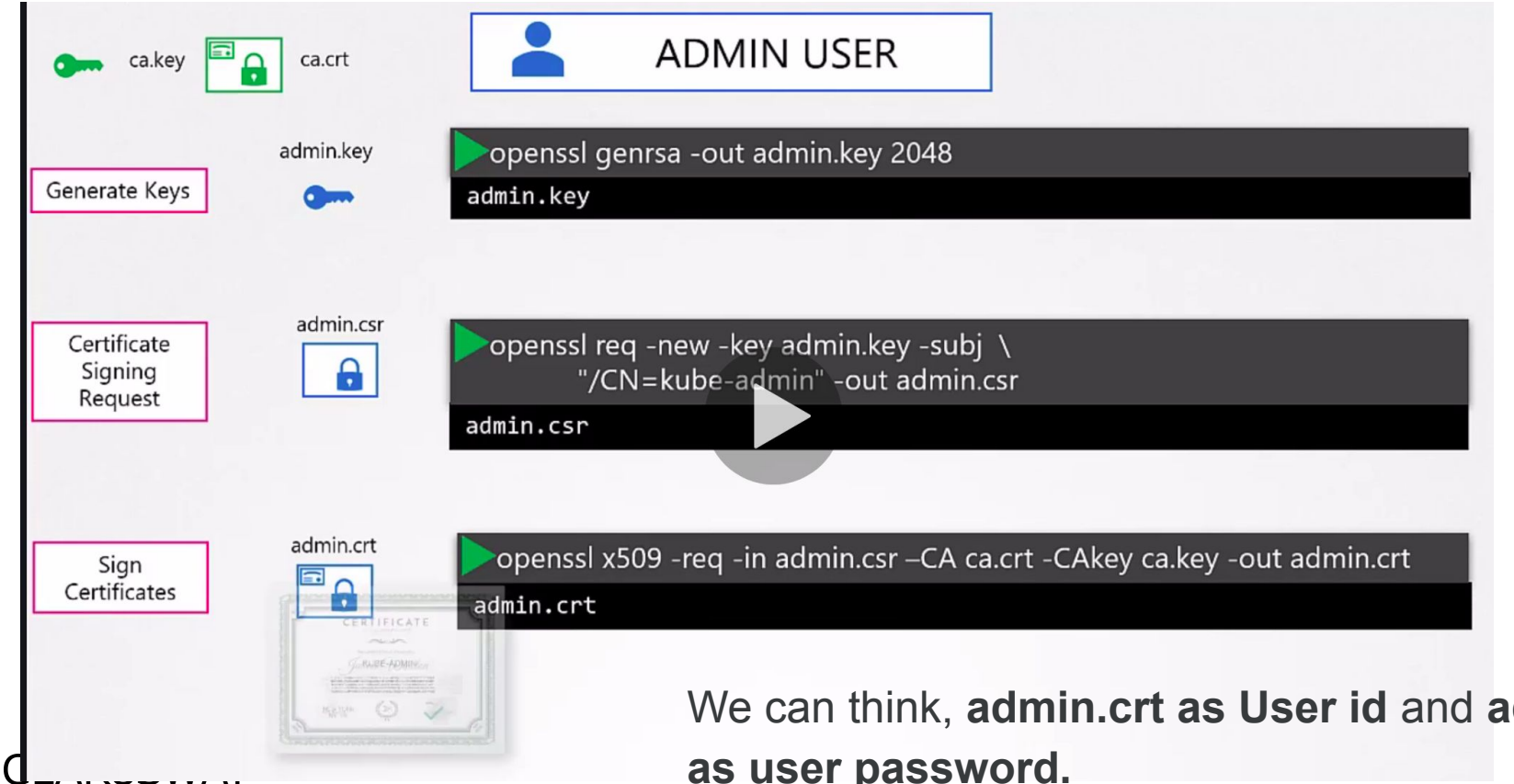
Certificate Authority (CA)

1. Create a ca.key.
2. Create a ca.csr from ca.key.
3. Create a ca.crt from ca.csr and ca.key.

Admin User

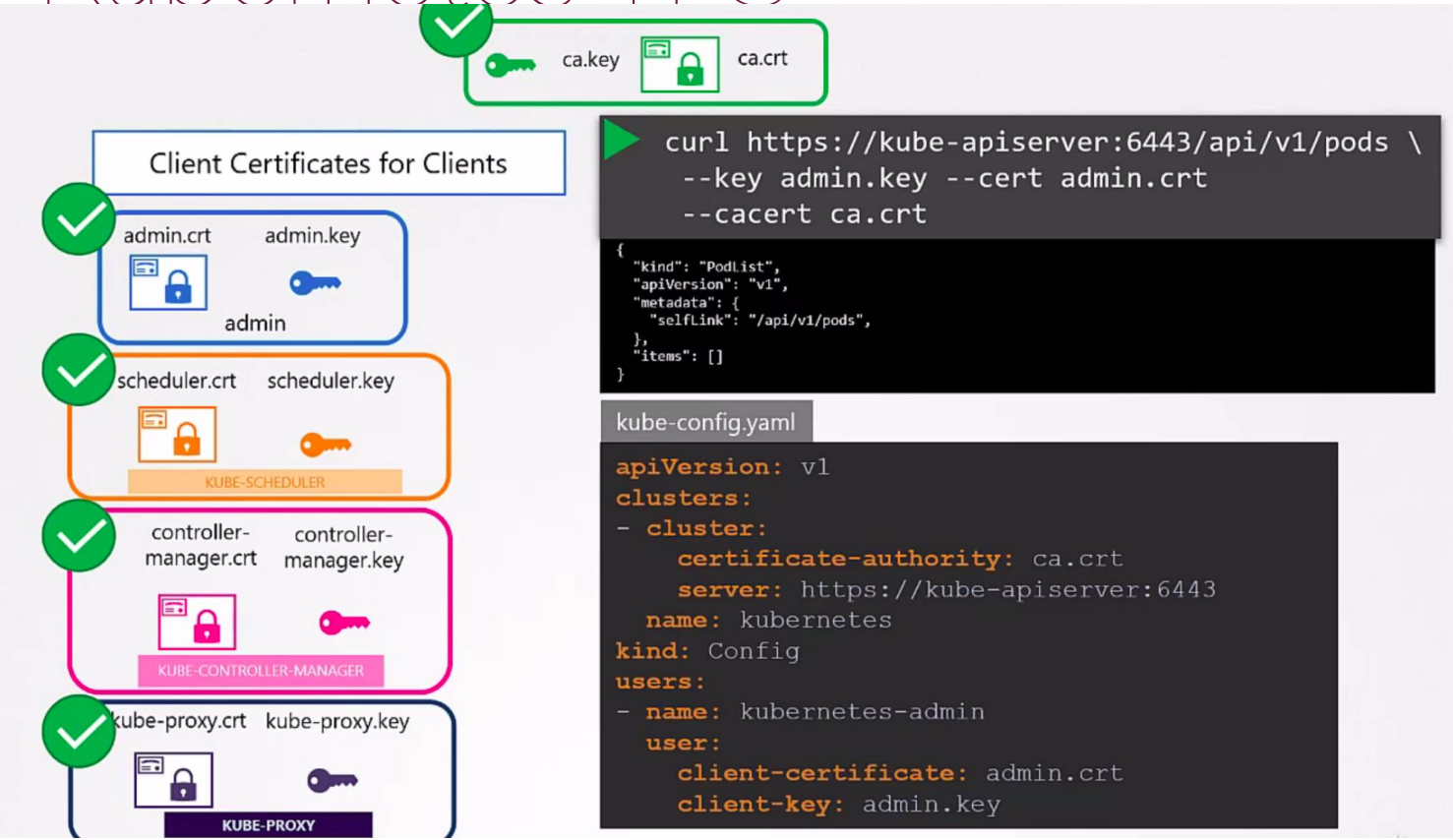
4. Create an admin.key.
5. Create an admin.csr from admin.key.
6. Create a admin.crt from admin.csr, ca.crt and ca.key.

Kubernetes TLS



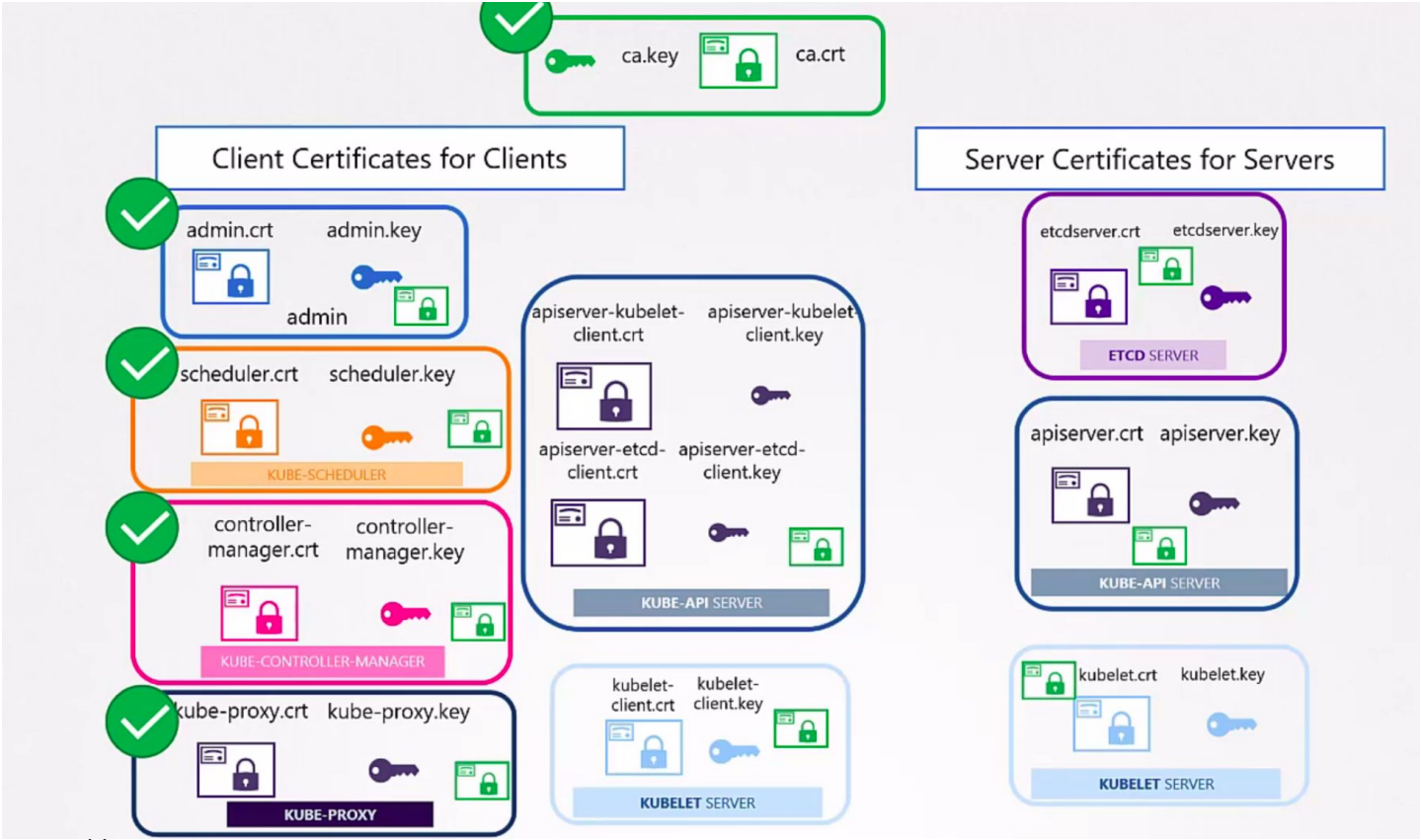
We can think, **admin.crt as User id** and **admin.key as user password.**

Kubernetes TLS

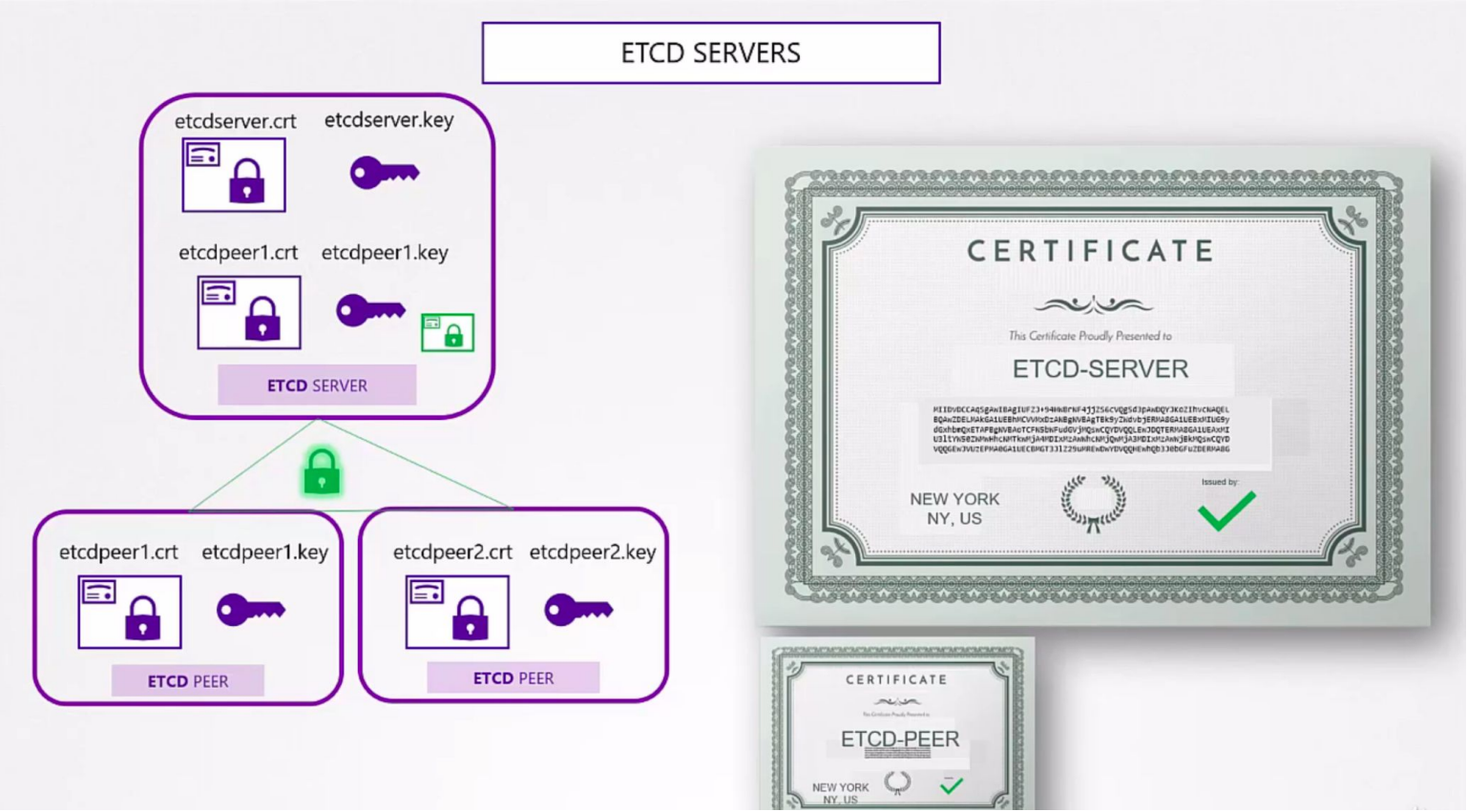


To connect kupe-apiserver, we need **admin.key, admin.crt and ca.crt.**

Kubernetes TLS

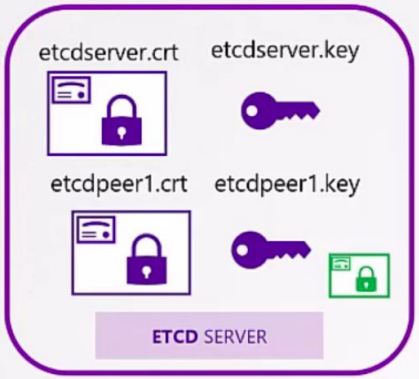


Kubernetes TLS



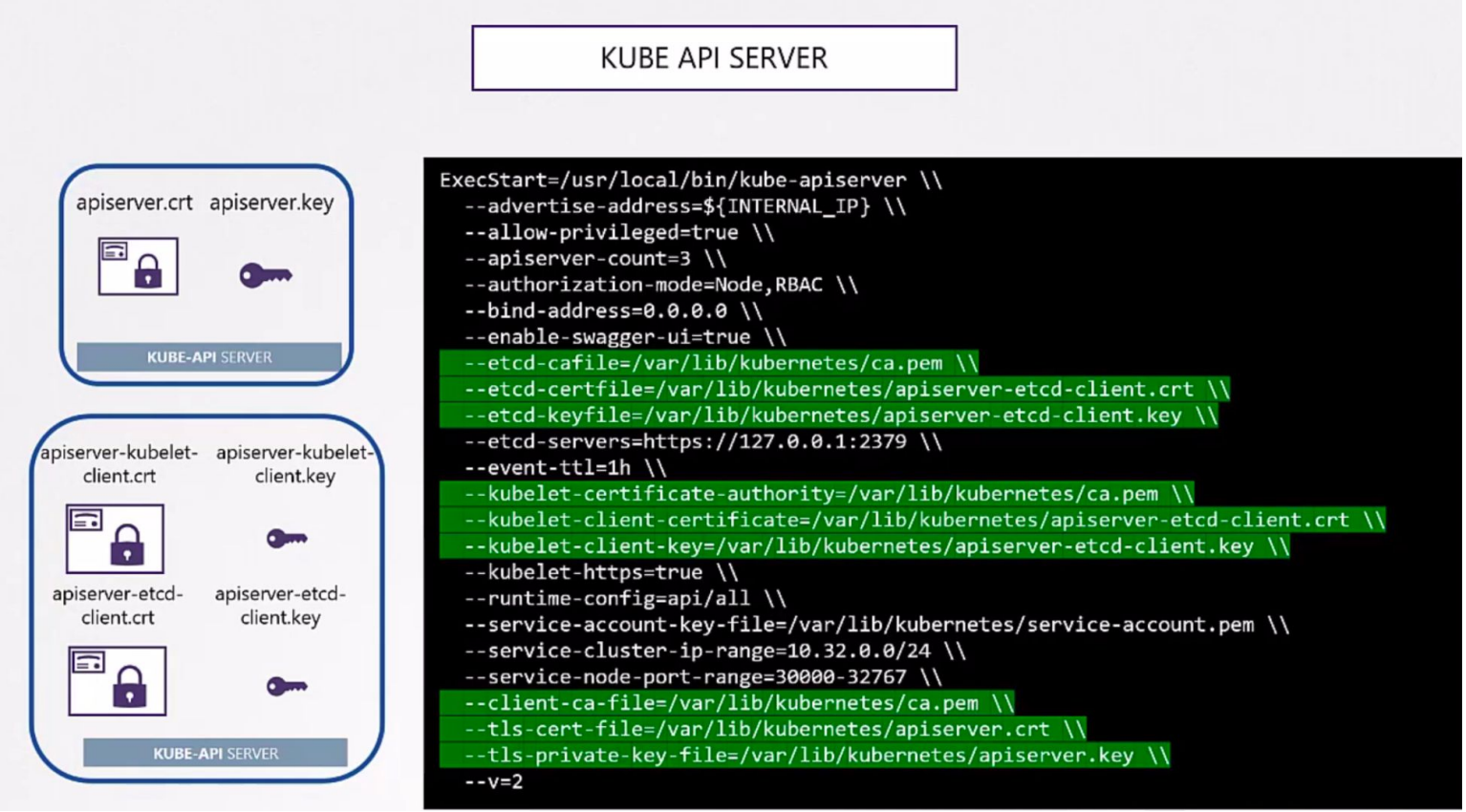
Kubernetes TLS

ETCD SERVERS




```
cat etcd.yaml

- etcd
    - --advertise-client-urls=https://127.0.0.1:2379
    - --key-file=/path-to-certs/etcdserver.key
    - --cert-file=/path-to-certs/etcdserver.crt
    - --client-cert-auth=true
    - --data-dir=/var/lib/etcd
    - --initial-advertise-peer-urls=https://127.0.0.1:2380
    - --initial-cluster=master=https://127.0.0.1:2380
    - --listen-client-urls=https://127.0.0.1:2379
    - --listen-peer-urls=https://127.0.0.1:2380
    - --name=master
    - --peer-cert-file=/path-to-certs/etcdpeer1.crt
    - --peer-client-cert-auth=true
    - --peer-key-file=/etc/kubernetes/pki/etcd/peer.key
    - --peer-trusted-ca-file=/etc/kubernetes/pki/etcd/ca.crt
    - --snapshot-count=10000
    - --trusted-ca-file=/etc/kubernetes/pki/etcd/ca.crt
```

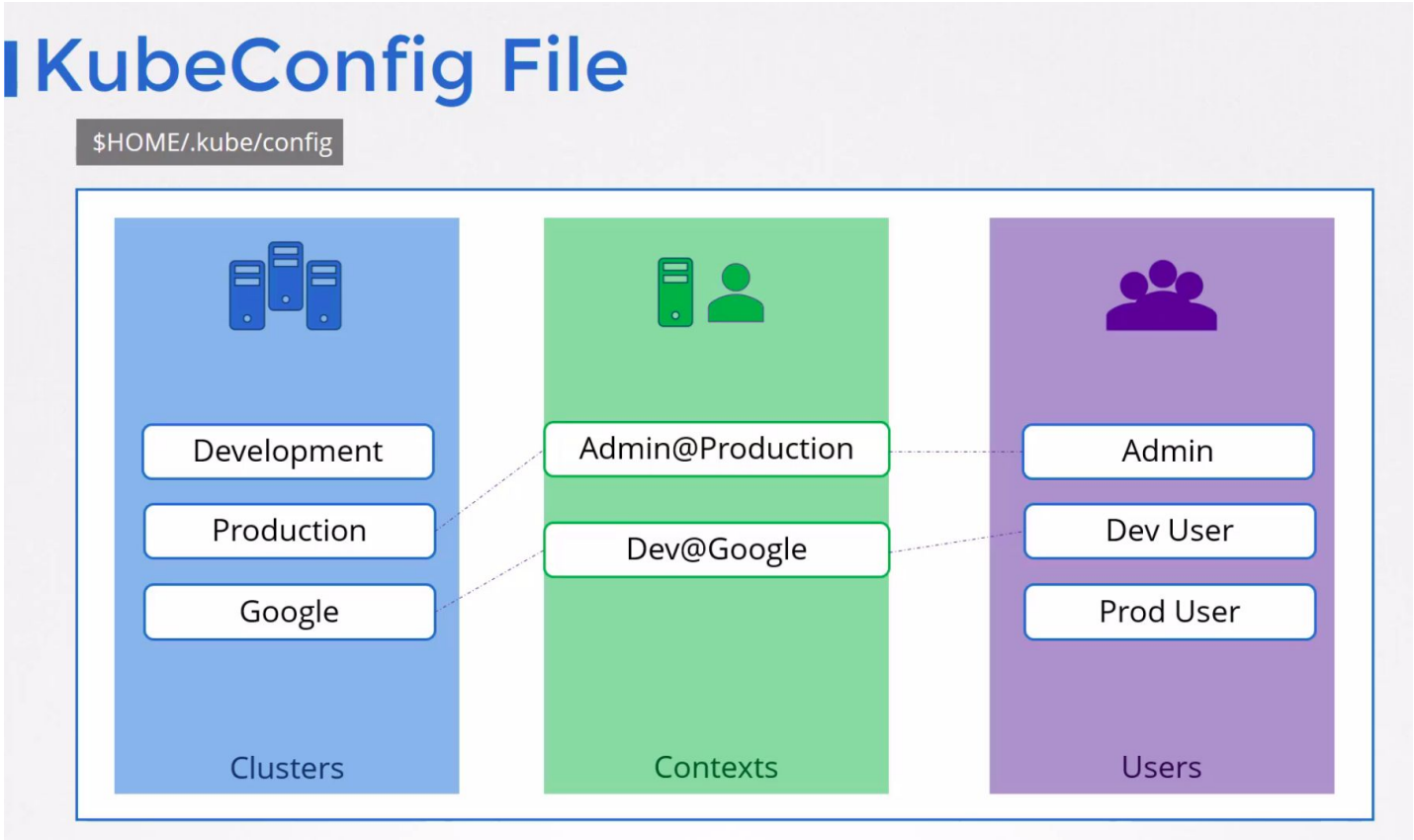
Kubernetes TLS



3

KubeConfig

KubeConfig

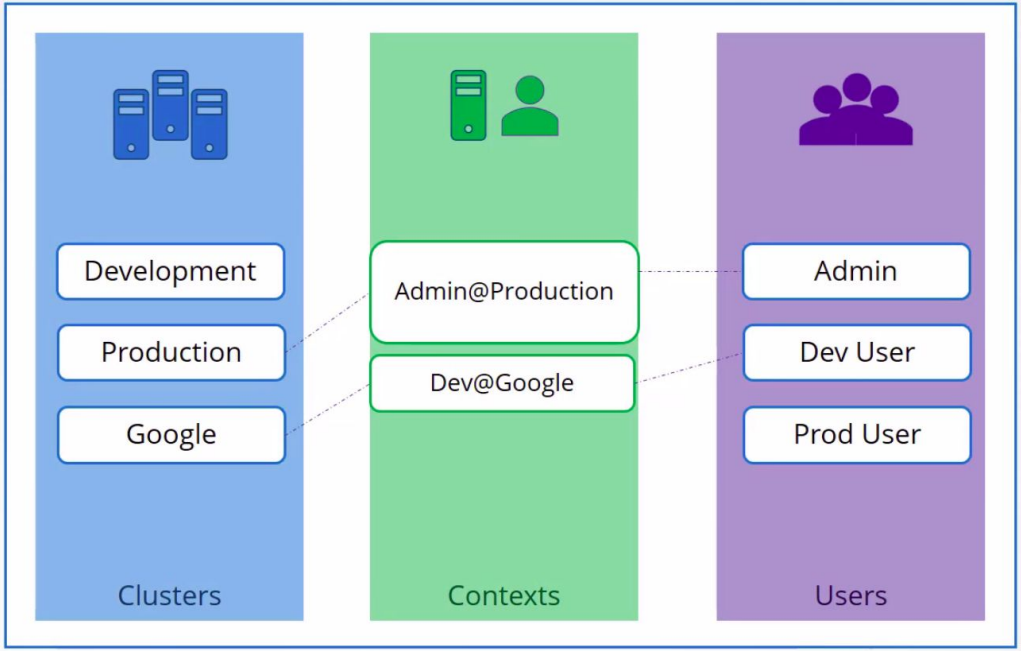

KubeConfig File
$HOME/.kube/config
Development
Production
Google
Clusters
Admin@Production
Dev@Google
Contexts
Admin
Dev User
Prod User
Users

KubeConfig

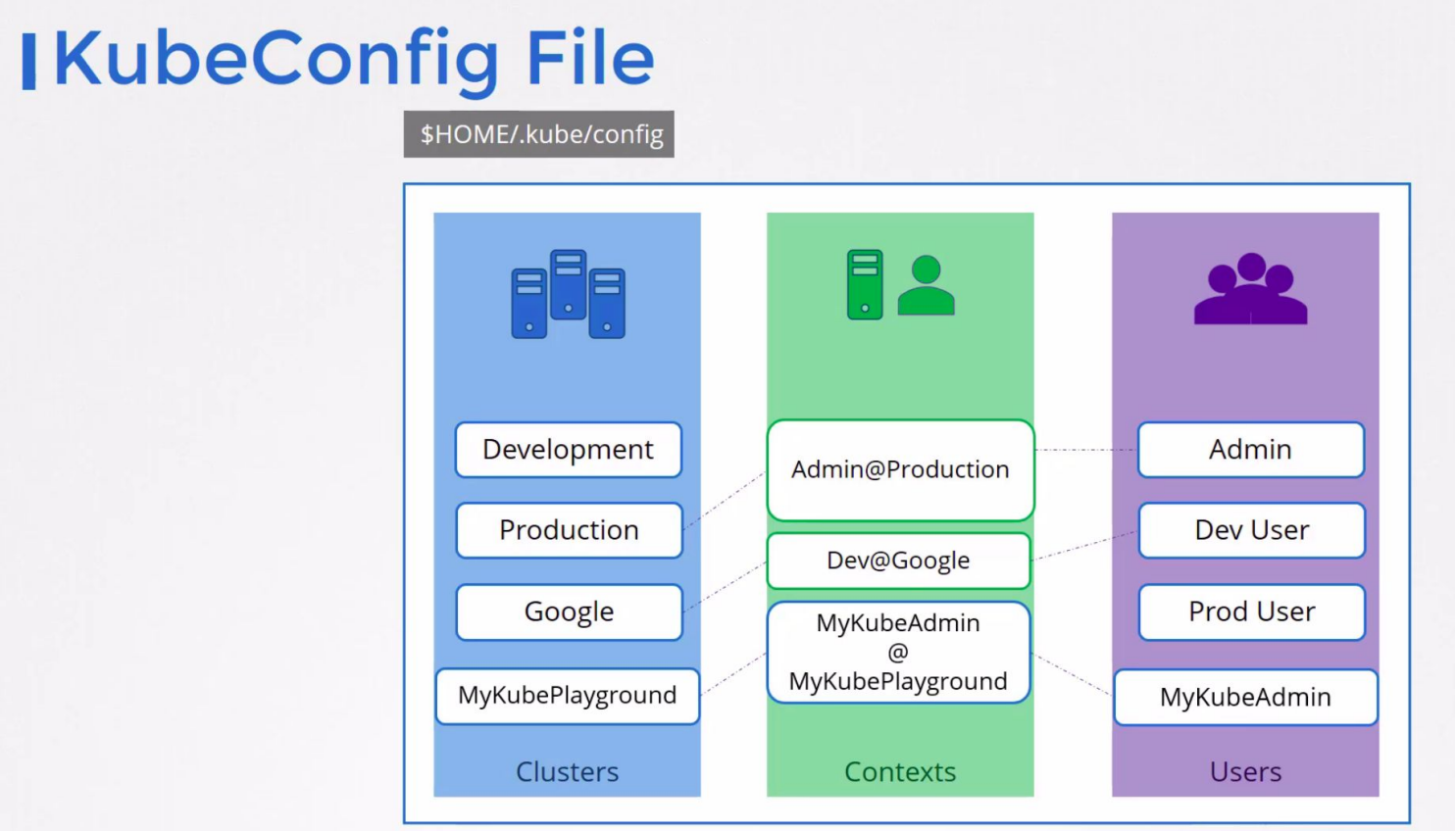
KubeConfig File

$HOME/.kube/config

```
--server my-kube-playground:6443
--client-key admin.key
--client-certificate admin.crt
--certificate-authority ca.crt
```



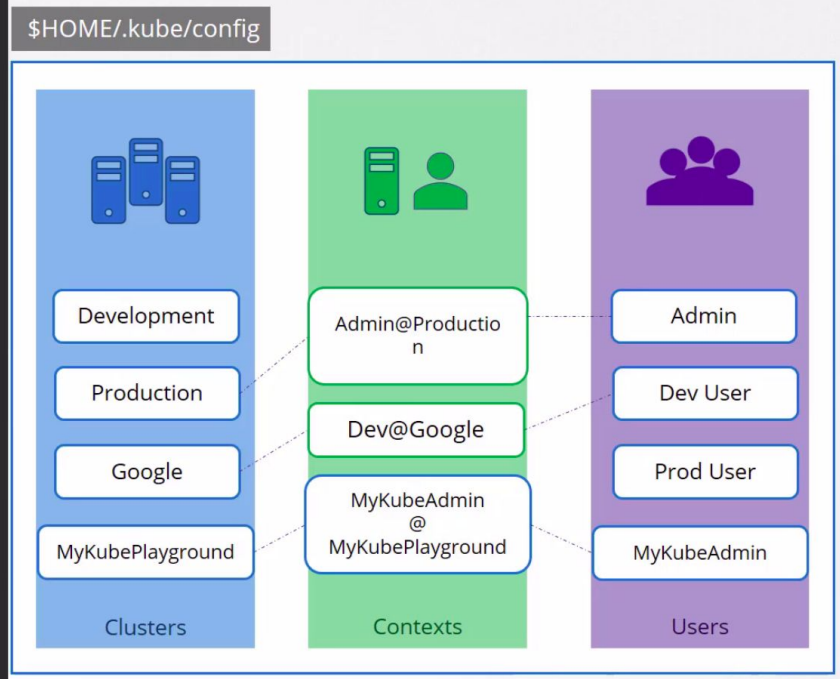
KubeConfig


KubeConfig File
$HOME/.kube/config
Development
Production
Google
MyKubePlayground
Clusters
Admin@Production
Dev@Google
MyKubeAdmin
@
MyKubePlayground
Contexts
Admin
Dev User
Prod User
MyKubeAdmin
Users

KubeConfig

KubeConfig File

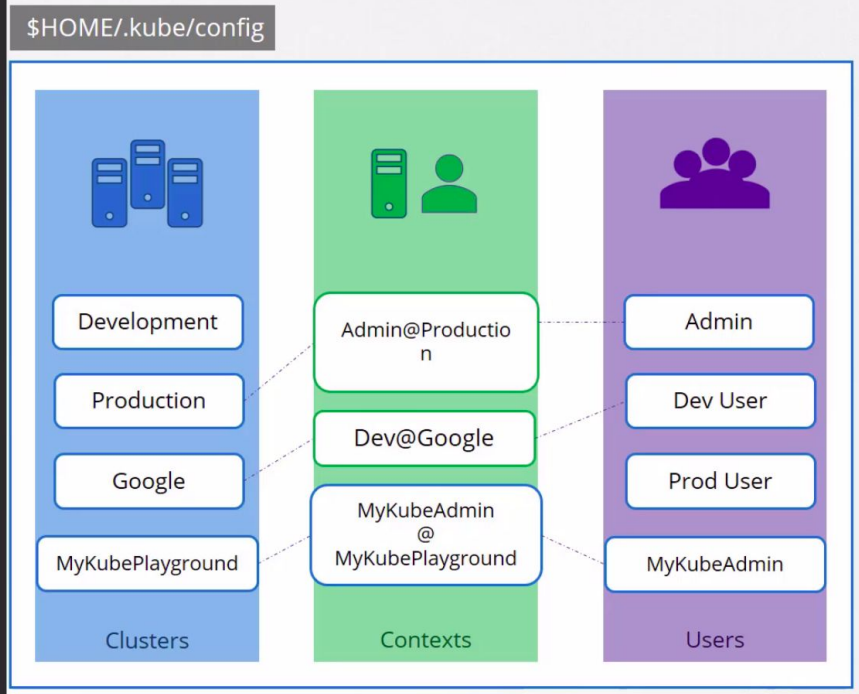
```
apiVersion: v1
kind: Config

clusters:




contexts:




users:
```



KubeConfig

KubeConfig File

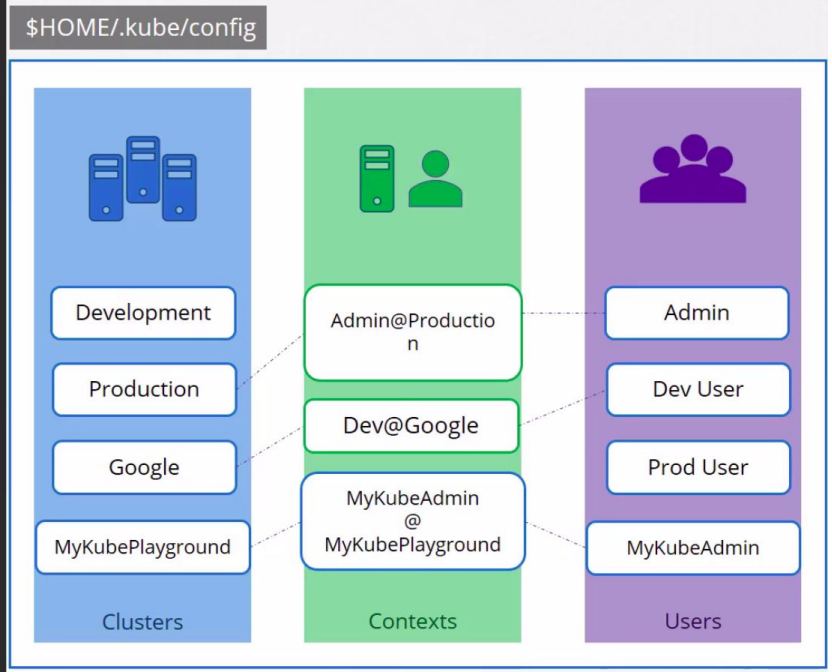
```
apiVersion: v1
kind: Config

clusters:

- name: my-kube-playground
  cluster:
    certificate-authority: ca.crt
    server: https://my-kube-playground:6443


contexts:

- name: my-kube-admin@my-kube-playground
  context:
    cluster: my-kube-playground
    user: my-kube-admin



users:

- name: my-kube-admin
  user:
    client-certificate: admin.crt
    client-key: admin.key
```



KubeConfig

KubeConfig File

```
apiVersion: v1
kind: Config
current-context: dev-user@google

clusters:
- name: my-kube-playground   (values hidden…)
- name: development
- name: production
- name: google

contexts:
- name: my-kube-admin@my-kube-playground
- name: dev-user@google
- name: prod-user@production

users:
- name: my-kube-admin
- name: admin
- name: dev-user
- name: prod-user
```



Kubectl config

```
kubectl config view
```

```
apiVersion: v1

kind: Config
current-context: my-kube-admin@my-kube-playground


clusters:
- name: my-kube-playground
- name: development
- name: production


contexts:
-  name: my-kube-admin@my-kube-playground
-  Name: prod-user@production


users:
- name: my-kube-admin
- name: prod-user
```

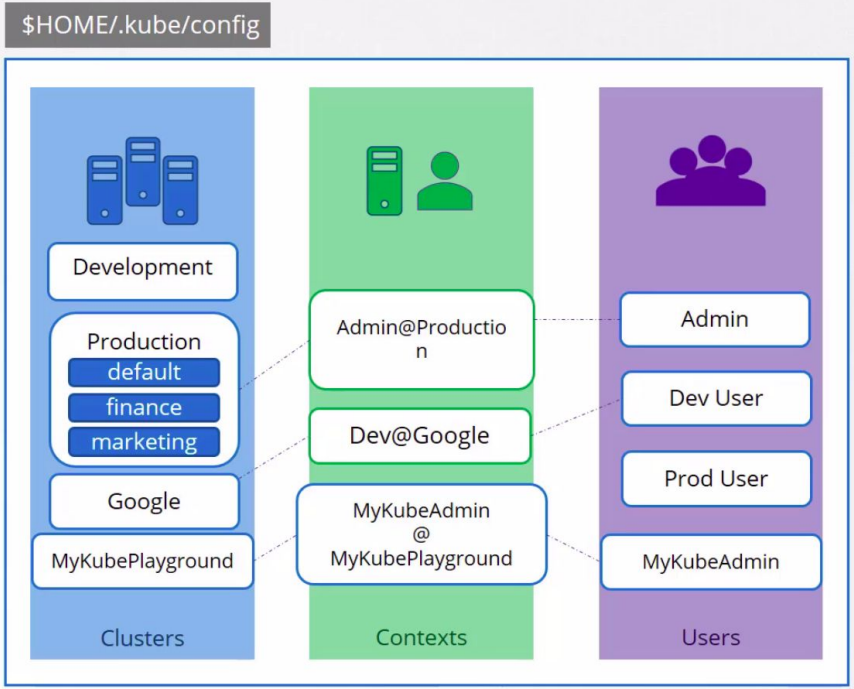
```
kubectl config use-context prod-user@producti
```

```
apiVersion: v1

kind: Config
current-context: prod-user@production


clusters:
- name: my-kube-playground
- name: development
- name: production


contexts:
-  name: my-kube-admin@my-kube-playground
-  Name: prod-user@production


users:
- name: my-kube-admin
- name: prod-user
```

KubeConfig

Namespaces

```
apiVersion: v1
kind: Config

clusters:
- name: production
  cluster:
    certificate-authority: ca.crt
    server: https://172.17.0.51:6443

contexts:
- name: admin@production
  context:
    cluster: production
    user: admin
    namespace: finance

users:
- name: admin
  user:
    client-certificate: admin.crt
    client-key: admin.key
```



KubeConfig

Certificates in KubeConfig

```
apiVersion: v1
kind: Config

clusters:
- name: production
  cluster:
    certificate-authority: /etc/kubernetes/pki/ca.crt
    server: https://172.17.0.51:6443

contexts:
- name: admin@production
  context:
    cluster: production
    user: admin
    namespace: finance

users:
- name: admin
  user:
    client-certificate: /etc/kubernetes/pki/users/admin.crt
    client-key: /etc/kubernetes/pki/users/admin.key
```

KubeConfig

Certificates in KubeConfig

```yaml
apiVersion: v1
kind: Config

clusters:
- name: production
  cluster:
    certificate-authority: /etc/kubernetes/pki/ca.crt

    certificate-authority-data: LS0tLS1CRUdJTiBDRVJU
                      SUZJQ0FURSBSRVFVRVNULS0tLS0KTUlJ
                      Q1dEQ0NBVUFDQVFBd0V6RVJNQThHQTFV
                      RUF3d0libVYzTFhWelpYSXdnZ0VpTUEw
                      R0NTcUdTSWIzRFFFQgpBUVVBQTRJQkR3
                      QXdnZ0VLQW9JQkFRRE8wV0pXK0RYc0FK
                      U0lyanBObzV2UklCcGxuemcrNnhjOStV
                      VndrS2kwCkxmQzI3dCsxZUVuT041TXVx
                      OTlOZXZtTUVPbnJ
```

```
-----BEGIN CERTIFICATE -----
MIICWDCCAUACAQAwEzERMA8GA1UEAwwIbmV3LXVzZXIwggEiMA0GC
AQUAA4IBDwAwggEKAoIBAQDO0WJW+DXsAJSIrjpNo5vRIBplnzg+6
LfC27t+1eEnON5Muq99NevmMEOnrDUO/thyVqP2w2XNIDRXjYyF46
y3BihhB93MJ7Oql3UTvZ8TELqyaDknRl/jv/SxgXkok0ABUTpWMx4
IF5nxAttMVkDPQ7NbeZRG43b+QWlVGR/z6DWOfJnbfezOtaAydGLT
EcCXAwqChjBLkz2BHPR4J89D6Xb8k39pu6jpyngV6uP0tIbOzpqNv
j2qEL+hZEWkkFz80lNNtyT5LxMqENDCnIgwC4GZiRGbrAgMBAAGgA
9w0BAQsFAAOCAQEAS9iS6C1uxTuf5BBYSU7QFQHUzalNxAdYsaORF
hOK4a2zyNyi44OOijyaD6tUW8DSxkr8BLK8Kg3srREtJql5rLZy9L
P9NL+aDRSxROVSqBaB2nWeYpM5cJ5TF53lesNSNMLQ2++RMnjDQJ7
Wr2EUM6UawzykrdHImwTv2mlMY0R+DNtV1Yie+0H9/YElt+FSGjh5
4l3E/y3qL71WfAcuH3OsVpUUnQISMdQs0qWCsbE56CC5DhPGZIpUb
vwQ07jG+hpknxmuFAeXxgUwodALaJ7ju/TDIcw==
-----END CERTIFICATE -----
```

```
cat ca.crt | base64
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSBSRVFVRVN
tLS0KTUlJQ1dEQ0NBVUFDQVFBd0V6RVJNQThHQT
F3d0libVYzTFhWelpYSXdnZ0VpTUEwR0NTcUdTS
FFFQgpBUVVBQTRJQkR3QXdnZ0VLQW9JQkFRRE8w
K0RYc0FKU0lyanBObzV2UklCcGxuemcrNnhjOSt
rS2kwCkxmQzI3dCsxZUVuT041TXVxOTlOZXZtTU
J
```