| #Install VirtualBox and docker:  sudo apt update -y  sudo apt-get install docker.io -y  systemctl start docker  sudo usermod -aG docker $(whoami)  sudo apt install virtualbox -y  sudo apt-get install conntrack -y  #Install kubectl:  sudo apt-get update && sudo apt-get install -y apt-transport-https gnupg2  curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -  echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list  sudo apt-get update  sudo apt-get install -y kubectl  #Install Minikube:  curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64 && chmod +x minikube && sudo mv minikube /usr/local/bin/  #start minikube  apt install jq -y  minikube start --kubernetes-version=v1.22.0 --driver=none  #enable ingress add on minikube  minikube addons enable ingress  #Install helm  curl https://baltocdn.com/helm/signing.asc | gpg --dearmor | sudo tee /usr/share/keyrings/helm.gpg > /dev/null  sudo apt-get install apt-transport-https --yes  echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/helm.gpg] https://baltocdn.com/helm/stable/debian/ all main" | sudo tee /etc/apt/sources.list.d/helm-stable-debian.list  sudo apt-get update  sudo apt-get install helm |
| --- |

Vault Install

| helm repo add hashicorp https://helm.releases.hashicorp.com  helm repo update  helm install vault hashicorp/vault --set "injector.enabled=false"  # Get vault pod  kubectl get po vault-0  # Create Vault unseal key  kubectl exec vault-0 -- vault operator init -key-shares=1 -key-threshold=1 -format=json > /tmp/demo-keys.json  cat /tmp/demo-keys.json | jq -r ".unseal\_keys\_b64[]"    # Unseal Vault  VAULT\_UNSEAL\_KEY=$(cat /tmp/demo-keys.json | jq -r ".unseal\_keys\_b64[]")  kubectl exec vault-0 -- vault operator unseal $VAULT\_UNSEAL\_KEY    # Get Vault token from our file  VAULT\_TOKEN=$(cat /tmp/demo-keys.json | jq -r ".root\_token")  # Get a shell in vault-0 pod with VAULT\_TOKEN as environment variable  kubectl exec vault-0 -ti -- /bin/sh -c "VAULT\_TOKEN=$VAULT\_TOKEN sh"  # Vault CLI example  vault token lookup  # Exit pod  exit |
| --- |

Kubernetes Auth Method

| # Prepare Vault CLI in `vault-0` pod  VAULT\_TOKEN=$(cat /tmp/demo-keys.json | jq -r ".root\_token")  kubectl exec vault-0 -ti -- /bin/sh -c "VAULT\_TOKEN=$VAULT\_TOKEN sh"  # Enable Kubernetes Auth Method  vault auth enable kubernetes  vault write auth/kubernetes/config \  token\_reviewer\_jwt="$(cat /var/run/secrets/kubernetes.io/serviceaccount/token)" \  kubernetes\_host="https://$KUBERNETES\_PORT\_443\_TCP\_ADDR:443" \  kubernetes\_ca\_cert=@/var/run/secrets/kubernetes.io/serviceaccount/ca.crt  # Policy pki: read access to pki paths  vault policy write pki - <<EOF  path "pki\*" { capabilities = ["read", "list"] }  path "pki/roles/example-dot-com" { capabilities = ["create", "update"] }  path "pki/sign/example-dot-com" { capabilities = ["create", "update"] }  path "pki/issue/example-dot-com" { capabilities = ["create"] }  EOF  # Create Kubernetes Auth Method role: Grant service account `issuer` access via the `pki` policy  vault write auth/kubernetes/role/issuer \  bound\_service\_account\_names=issuer \  bound\_service\_account\_namespaces=default \  policies=pki  # Exit pod  exit |
| --- |

Webapp with ingress

| # Deploy webapp  kubectl create deployment web --image=gcr.io/google-samples/hello-app:1.0  # Create service  kubectl expose deployment web --port=8080  kubectl get service web  # Create ingress  cat <<EOF | kubectl apply -f -  apiVersion: networking.k8s.io/v1  kind: Ingress  metadata:  name: example-ingress  annotations:  kubernetes.io/ingress.class: "nginx"  spec:  tls:  - hosts:  - demo.example.com  secretName: demo-example-com-tls  rules:  - host: demo.example.com  http:  paths:  - path: /  pathType: Prefix  backend:  service:  name: web  port:  number: 8080  EOF    Add host entry in your laptop |
| --- |

Automate Certificate Renewal

Enable PKI in Vault

| # Prepare Vault CLI in `vault-0` pod  VAULT\_TOKEN=$(cat /tmp/demo-keys.json | jq -r ".root\_token")  kubectl exec vault-0 -ti -- /bin/sh -c "VAULT\_TOKEN=$VAULT\_TOKEN sh"  # Enable PKI secret engine  vault secrets enable pki  # Generate root CA (and write to file)  vault write -format=json pki/root/generate/internal \  common\_name="Demo Root Certificate Authority" > /tmp/demo-root-ca.json  cat /tmp/demo-root-ca.json  # Configure pki api endpoints  vault write pki/config/urls \  issuing\_certificates="http://vault.default:8200/v1/pki/ca" \  crl\_distribution\_points="http://vault.default:8200/v1/pki/crl"  # Create `pki` role to create certificates for example.com and all subdomains and with a maximum time to live of 1 day.  vault write pki/roles/example-dot-com \  allowed\_domains=example.com \  allow\_subdomains=true \  max\_ttl=1d  # Exit pod  exit |
| --- |

Cert-Manager

| kubectl create namespace cert-manager  helm repo add jetstack https://charts.jetstack.io  helm repo update  # Install Cert-Manager  helm install cert-manager jetstack/cert-manager \  --namespace cert-manager \  --version v1.3.0 \  --set installCRDs=true  # View pods  kubectl get pods --namespace cert-manager  # View CRD's  kubectl get customresourcedefinitions.apiextensions.k8s.io |
| --- |

| # Create service account  kubectl create serviceaccount issuer  # Token of service account `issuer`  ISSUER\_SECRET\_REF=$(kubectl get serviceaccount issuer -o json | jq -r ".secrets[].name")  # Define issuer  cat <<EOF | kubectl apply -f -  apiVersion: cert-manager.io/v1alpha2  kind: Issuer  metadata:  name: vault-issuer  namespace: default  spec:  vault:  server: http://vault.default  path: pki/sign/example-dot-com  auth:  kubernetes:  mountPath: /v1/auth/kubernetes  role: issuer  secretRef:  name: $ISSUER\_SECRET\_REF  key: token  EOF  # Show issuer  kubectl get issuer |
| --- |

| # Define certificate  cat <<EOF | kubectl apply -f -  apiVersion: cert-manager.io/v1alpha2  kind: Certificate  metadata:  name: demo-example-com  namespace: default  spec:  secretName: demo-example-com-tls  issuerRef:  name: vault-issuer  commonName: demo.example.com  dnsNames:  - demo.example.com  EOF  # View certificate CRD  kubectl get certificate |
| --- |

# See the request in the events

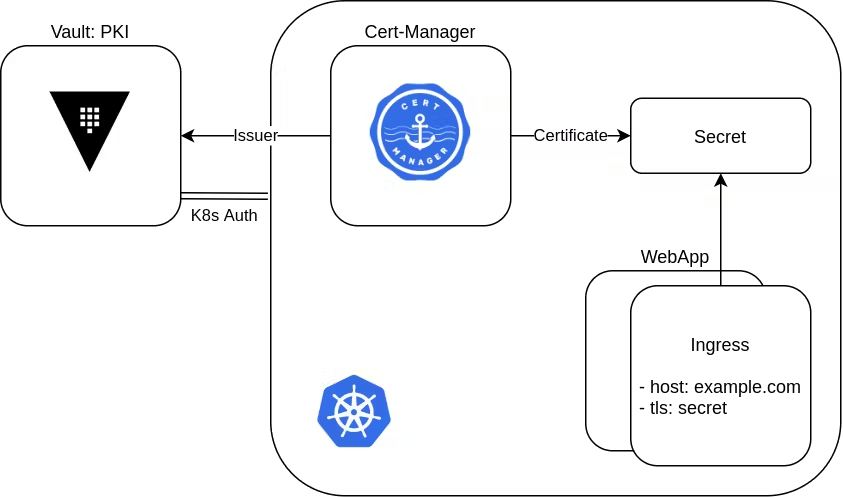
kubectl describe certificate demo-example-com

# New secret

kubectl get secrets demo-example-com-tls

**Certificate renewal**

* Vault exposing an pki api
* Cert-Manager using this pki api to request a certificate
* the certificate is stored as a Kubernetes secret demo-example-com-tls
* Ingress gets demo-example-com-tls and uses it to enable TLS on our webapp



**Import the certificate in google chrome**

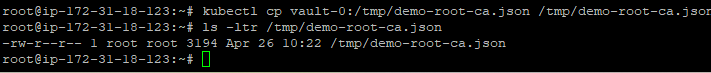
# Copy root CA json from container to local machine

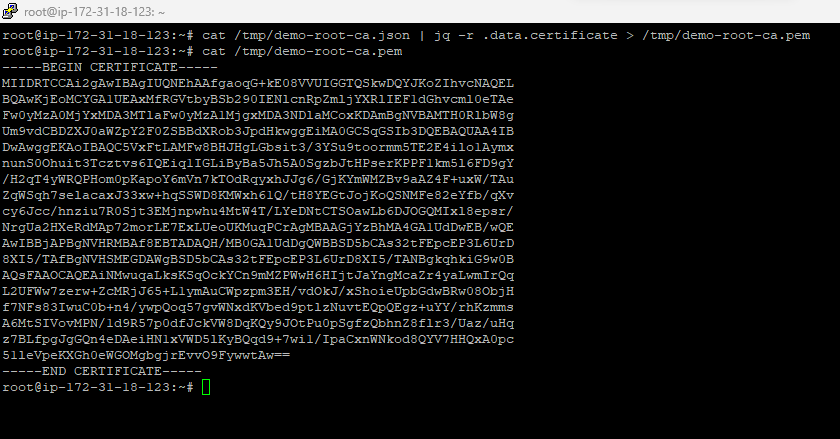
kubectl cp vault-0:/tmp/demo-root-ca.json /tmp/demo-root-ca.json

# Extract the root CA in pem format

cat /tmp/demo-root-ca.json | jq -r .data.certificate > /tmp/demo-root-ca.pem

cat /tmp/demo-root-ca.pem



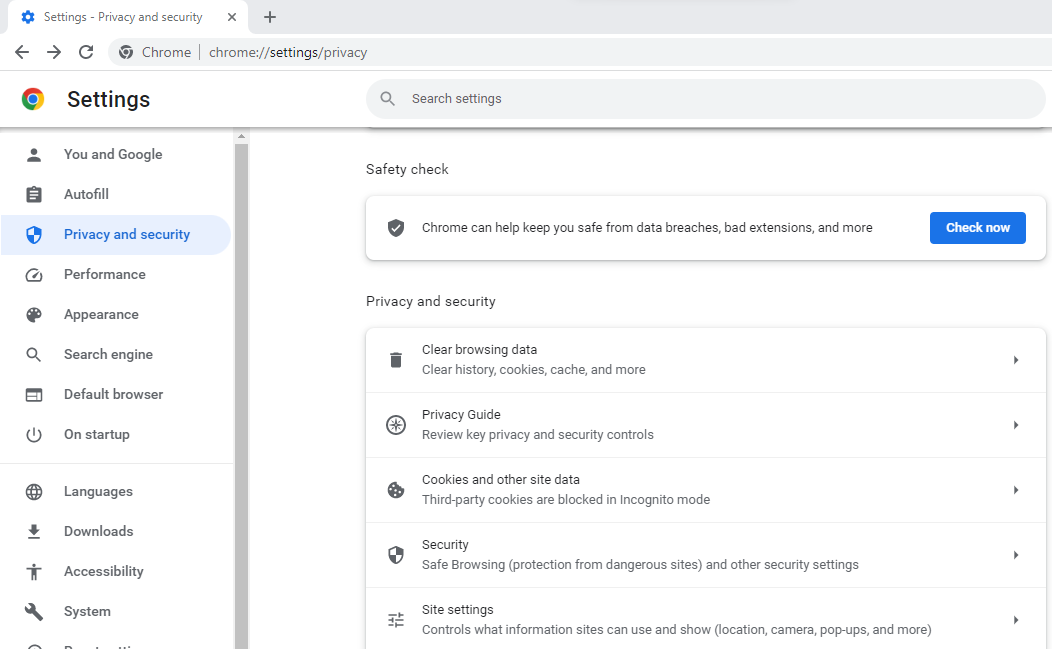


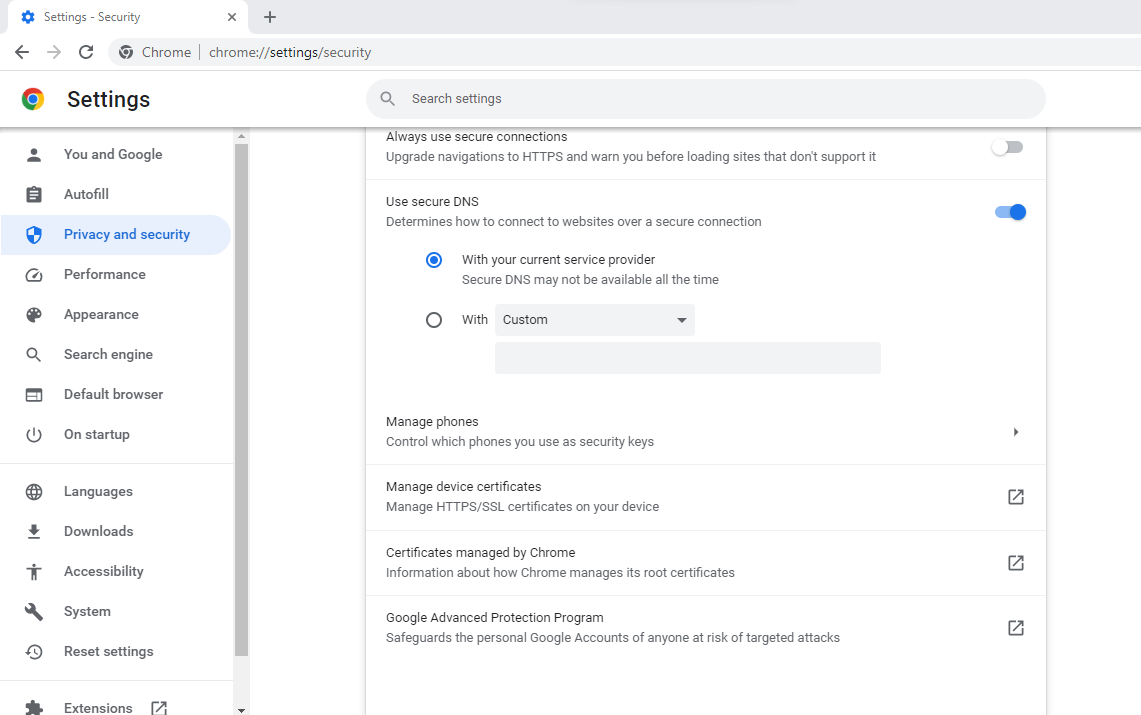
Then import it into our browser.

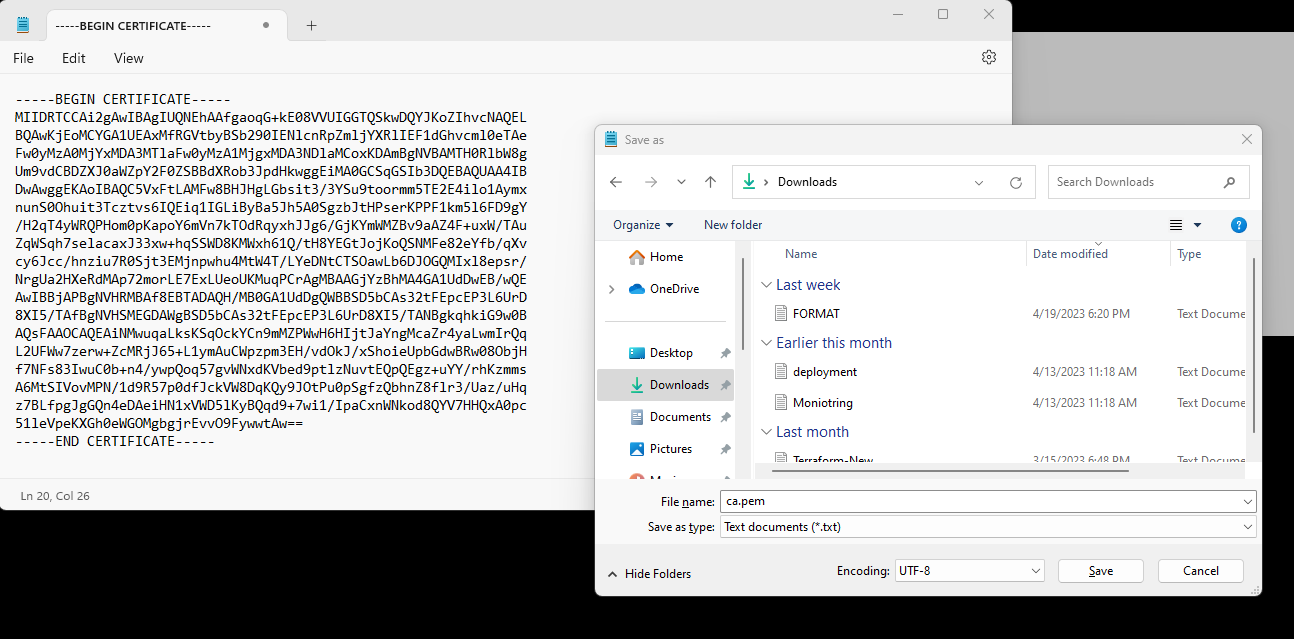
Go to Settings

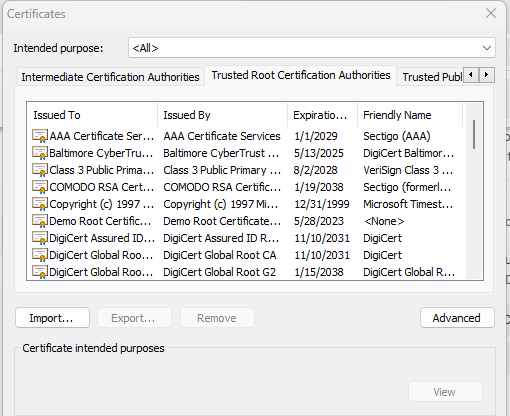
Go to Privacy and security. Select security

And then select manage device certificates and import the certificate from above screenshot. You can save to your local and import it

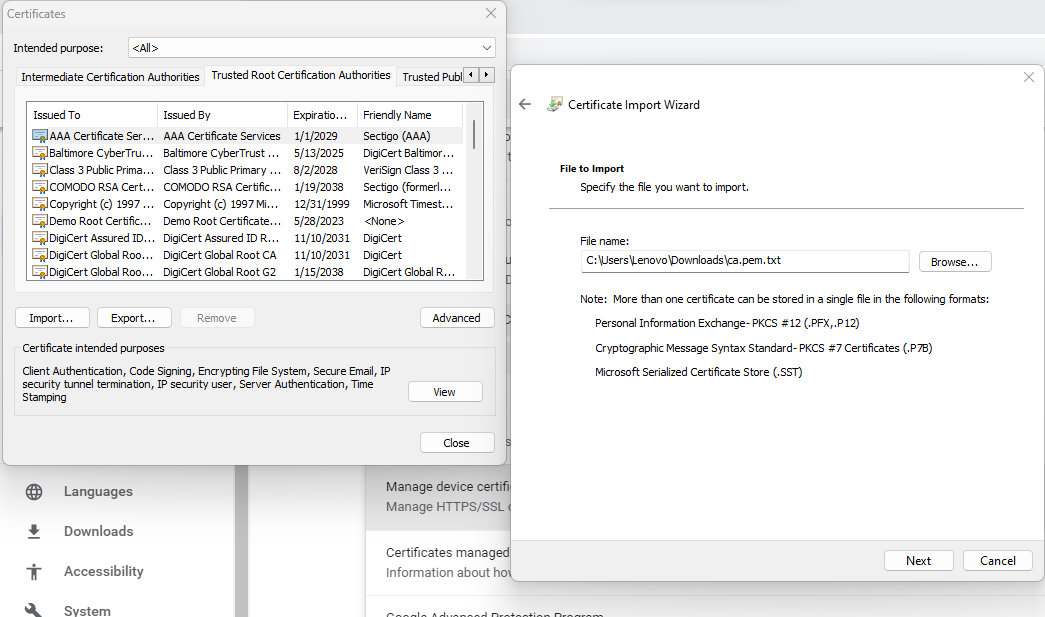








Click on import and the select the file which we saved earlier



Now you can see certificate is valid and it will automatically renew in one day

