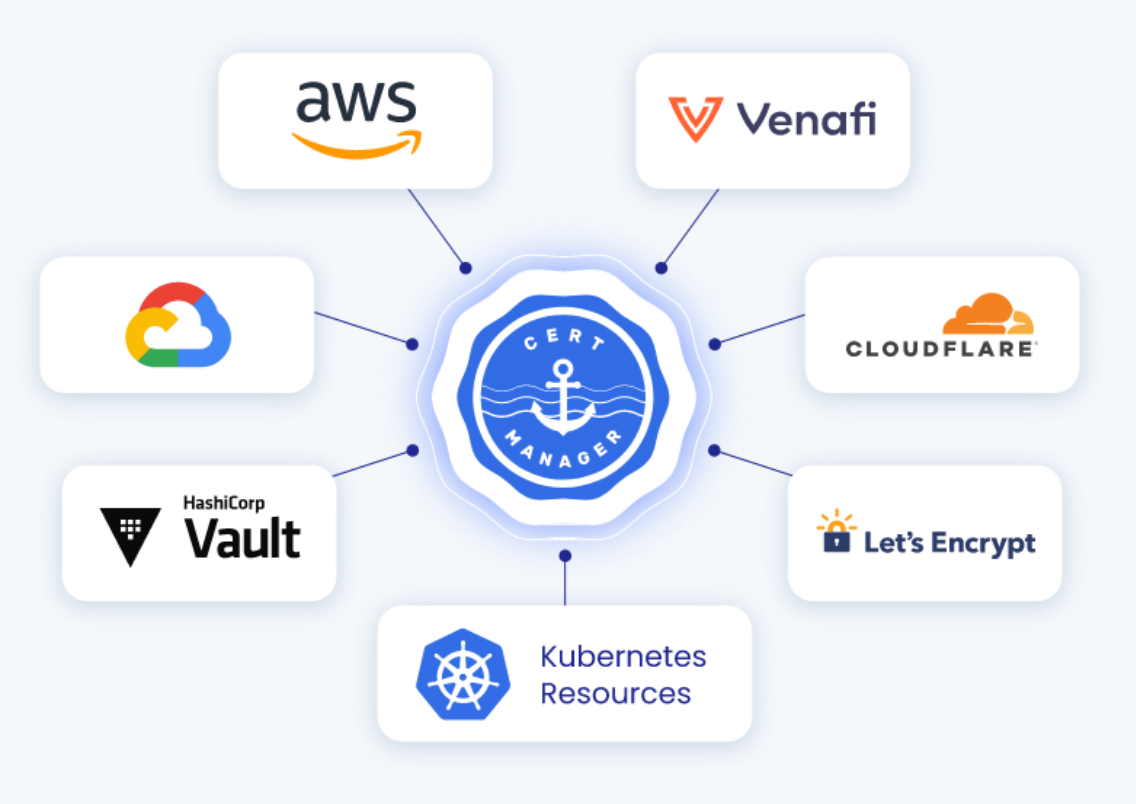
Lab 18 - Day 2 Certificates

Gloo and Istio heavily rely on TLS certificates to facilitate safe and secure communication. Gloo Platform uses mutual tls authentication for communication between the Server and the Agents. Istio requires an Intermediate Signing CA so that it can issue workload certificates to each of the mesh enabled services. These workload certificates encrypt and authenticate traffic between each of your microservices.

It is important to design and implement a secure and reliable Public Key Infrastructure (PKI) that Gloo and Istio can rely on which is why we chose cert-manager as the PKI due to its versatility and reliability for managing certificates.

Cert Manager

Not only is cert-manager the most widely used Kubernetes based solution, it natively integrates with a number of different issuing systems such as AWS Private CA, Google Cloud CA and Vault. Finally, cert-manager also creates certificates in the form of kubernetes secrets which are compatible with both Istio and Gloo Platform. It also has the ability to automatically rotate them when they are nearing their end of life.



- Add the cert-manager helm chart

```
helm repo add jetstack https://charts.jetstack.io
helm repo update
```

- Install cert-manager using helm

```
helm install cert-manager jetstack/cert-manager \
  --namespace cert-manager \
  --create-namespace \
  --version v1.12.2 \
  --kube-context management \
  -f data/cert-manager-values.yaml

helm install cert-manager jetstack/cert-manager \
  --namespace cert-manager \
  --create-namespace \
  --version v1.12.2 \
  --kube-context cluster-1 \
  -f data/cert-manager-values.yaml

helm install cert-manager jetstack/cert-manager \
  --namespace cert-manager \
  --create-namespace \
  --version v1.12.2 \
  --kube-context cluster-2 \
  -f data/cert-manager-values.yaml
```

- Wait for deployments to become healthy

```
kubectl wait deployment --for condition=Available=True --all --context management -n
cert-manager
kubectl wait deployment --for condition=Available=True --all --context cluster-1 -n
cert-manager
kubectl wait deployment --for condition=Available=True --all --context cluster-2 -n
cert-manager
```

Self Signed Root Cert

This lab will use a self signed root certificate for all relay and workload certificates. This is not recommended in production as the Root CA key is stored in a kubernetes secret. Instead an external 3rd party PKI is recommended like Vault or Venafi.

- Create the self-signed secret

```
kubectl create secret generic issuer-ca --from-file=tls.key=data/root-key.pem --
from-file=tls.crt=data/root-cert.pem --context management -n cert-manager
kubectl create secret generic issuer-ca --from-file=tls.key=data/root-key.pem --
from-file=tls.crt=data/root-cert.pem --context cluster-1 -n cert-manager
kubectl create secret generic issuer-ca --from-file=tls.key=data/root-key.pem --
from-file=tls.crt=data/root-cert.pem --context cluster-2 -n cert-manager
```

- Create a ClusterIssuer for the root secret

```
kubectl apply --context management -n cert-manager -f data/secret-issuer.yaml
kubectl apply --context cluster-1 -n cert-manager -f data/secret-issuer.yaml
```

```
kubectl apply --context cluster-2 -n cert-manager -f data/secret-issuer.yaml
```

- Verify the issuers

```
kubectl get clusterissuer self-signed-issuer -o jsonpath='{.status}' --context
management -n cert-manager
kubectl get clusterissuer self-signed-issuer -o jsonpath='{.status}' --context
cluster-1 -n cert-manager
kubectl get clusterissuer self-signed-issuer -o jsonpath='{.status}' --context
cluster-2 -n cert-manager
```

Cluster: management Configuration

The Gloo Platform server and Telemetry Gateway will need mTLS server certificates. The following commands generate the two certificates to allow the applications to receive connections from the workload clusters. * Create certificates for Gloo Management Server and Telemetry Gateway ```shell kubectl apply --context management -f - <<EOF apiVersion: cert-manager.io/v1 kind: Certificate metadata: name: gloo-mgmt-server namespace: gloo-mesh spec: commonName: gloo-mgmt-server dnsNames: - "*.gloo-mesh" duration: 8760h0m0s ### 1 year life renewBefore: 8736h0m0s issuerRef: kind: ClusterIssuer name: self-signed-issuer secretName: gloo-mgmt-server-tls usages: - server auth - client auth privateKey: algorithm: "RSA" size: 4096

apiVersion: cert-manager.io/v1 kind: Certificate metadata: name: gloo-telemetry-gateway namespace: gloo-mesh spec: commonName: gloo-mgmt-server dnsNames: - "*.gloo-mesh" duration: 8760h0m0s ### 1 year life renewBefore: 8736h0m0s issuerRef: kind: ClusterIssuer name: self-signed-issuer secretName: gloo-telemetry-gateway usages: - server auth - client auth privateKey: algorithm: "RSA" size: 4096 EOF

```
* Verify certificates were created
```shell
kubectl get certificates --context management -n gloo-mesh
```

Note if certificates were not generated it may be beneficial to look at the cert manager logs.

```
kubectl logs deploy/cert-manager --context management -n cert-manager
```

- Cleanup old Gloo certificates and allow cert-manager to replace them

```
kubectl delete secret relay-server-tls-secret --context management -n gloo-mesh
kubectl delete secret relay-tls-signing-secret --context management -n gloo-mesh
```
```

```
kubectl delete secret relay-root-tls-secret --context management -n gloo-mesh
```

- Update the Gloo Platform to use the new certificates

```
helm upgrade --install gloo-platform gloo-platform/gloo-platform \
  --version=2.3.9 \
  --namespace=gloo-mesh \
  --kube-context management \
  --reuse-values \
  -f data/gloo-mgmt-values.yaml
```

Cluster: cluster-1 Configuration

The workload clusters will need 2-3 certificates depending on your environment. The Gloo Platform Agent will require a client mTLS certificate for communicating with the Gloo Platform Server. Likewise the Telemetry Collector will also require an mTLS certificate to communicate with the Telemetry Gateway.

If you are relying on Istio's CA issuer functionality, you will also need to issue Istio a CA certificate to issue workload certificates to the dataplane. If your security posture does not allow for CA certificates to be stored on the workload clusters, ask your Solo.io Representative about `istio-csr`.

- Verify issuers is correctly setup

```
kubectl get clusterissuer --context cluster-1 -n cert-manager
```

- Create certificates for Gloo Agent, Telemetry Gateway and Istio if needed

```
kubectl apply --context cluster-1 -f - <<EOF
kind: Certificate
apiVersion: cert-manager.io/v1
metadata:
  name: gloo-agent
  namespace: gloo-mesh
spec:
  commonName: gloo-agent
  dnsNames:
    # Must match the cluster name used in the install
    - "cluster-1"
  duration: 8760h0m0s   ### 1 year life
  renewBefore: 8736h0m0s
  issuerRef:
    kind: ClusterIssuer
    name: self-signed-issuer
  secretName: relay-client-tls-secret
  usages:
    - digital signature
    - key encipherment
    - client auth
    - server auth
  privateKey:
```

```
    algorithm: "RSA"
    size: 4096
---
kind: Certificate
apiVersion: cert-manager.io/v1
metadata:
  name: gloo-telemetry-collector
  namespace: gloo-mesh
spec:
  commonName: gloo-telemetry-collector
  dnsNames:
    - "cluster-1-gloo-telemetry-collector"
  duration: 8760h0m0s   ### 1 year life
  renewBefore: 8736h0m0s
  issuerRef:
    kind: ClusterIssuer
    name: self-signed-issuer
  secretName: gloo-telemetry-collector
  usages:
    - digital signature
    - key encipherment
    - client auth
    - server auth
  privateKey:
    algorithm: "RSA"
    size: 4096
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: istio-cacerts
  namespace: istio-system
spec:
  secretName: cacerts
  duration: 720h # 30d
  renewBefore: 360h # 15d
  commonName: cluster-1.demo.example.com
  isCA: true
  usages:
    - digital signature
    - key encipherment
    - cert sign
  dnsNames:
    - cluster-1.demo.example.com
  issuerRef:
    kind: ClusterIssuer
    name: self-signed-issuer
EOF
```

- Verify certificates were created

```
kubectl get certificates --context cluster-1 -n gloo-mesh
kubectl get certificates --context cluster-1 -n istio-system
```

Note if certificates were not generated it may be beneficial to look at the cert manager logs.

```
kubectl logs deploy/cert-manager --context cluster-1 -n cert-manager
kubectl logs deploy/cert-manager-istio-csr --context cluster-1 -n cert-manager
```

- Cleanup old Gloo certificates and allow cert-manager to replace them

```
kubectl delete secret relay-client-tls-secret --context cluster-1 -n gloo-mesh
kubectl delete secret relay-root-tls-secret --context cluster-1 -n gloo-mesh
kubectl delete secret relay-identity-token-secret --context cluster-1 -n gloo-mesh
```

- Update the Gloo Platform to use the new certificates

```
helm upgrade --install gloo-agent gloo-platform/gloo-platform \
  --version=2.3.9  \
  --namespace gloo-mesh \
  --kube-context cluster-1 \
  --reuse-values \
  -f data/gloo-agent-values.yaml
```

- Verify Gloo Agent connectivity

```
kubectl logs deploy/gloo-mesh-agent --context cluster-1 -n gloo-mesh
```

- Update Istio to use new CA certificate

```
kubectl delete secret cacerts --context cluster-1 -n istio-system
```

- Verify new cacerts is generated

```
kubectl get secret cacerts --context cluster-1 -n istio-system
```

- Restart Istiod to pick up new certificate

```
kubectl rollout restart deploy --context cluster-1 -n istio-system
```

- Verify new certificate is picked up by istiod

```
kubectl logs -l app=istiod --tail 500 --context cluster-1 -n istio-system| grep x509
```

- Restart Gateways

```
kubectl rollout restart deploy --context cluster-1 -n istio-ingress
kubectl rollout restart deploy --context cluster-1 -n istio-eastwest
kubectl rollout restart deploy --context cluster-1 -n gloo-platform-addons
```

- Restart workloads

```
kubectl rollout restart deploy -n online-boutique --context cluster-1
```

Cluster: cluster-2 Configuration

- Verify issuers is correctly setup

```
kubectl get clusterissuer --context cluster-2 -n cert-manager
```

- Create certificates for Gloo Agent, Telemetry Gateway and Istio if needed

```
kubectl apply --context cluster-2 -f - <<EOF
kind: Certificate
apiVersion: cert-manager.io/v1
metadata:
  name: gloo-agent
  namespace: gloo-mesh
spec:
  commonName: gloo-agent
  dnsNames:
    # Must match the cluster name used in the install
    - "cluster-2"
  duration: 8760h0m0s   ### 1 year life
  renewBefore: 8736h0m0s
  issuerRef:
    kind: ClusterIssuer
    name: self-signed-issuer
  secretName: relay-client-tls-secret
  usages:
    - digital signature
    - key encipherment
    - client auth
    - server auth
  privateKey:
    algorithm: "RSA"
    size: 4096
---
kind: Certificate
apiVersion: cert-manager.io/v1
metadata:
  name: gloo-telemetry-collector
  namespace: gloo-mesh
spec:
  commonName: gloo-telemetry-collector
```

```
  dnsNames:
    - "cluster-2-gloo-telemetry-collector"
  duration: 8760h0m0s   ### 1 year life
  renewBefore: 8736h0m0s
  issuerRef:
    kind: ClusterIssuer
    name: self-signed-issuer
  secretName: gloo-telemetry-collector
  usages:
    - digital signature
    - key encipherment
    - client auth
    - server auth
  privateKey:
    algorithm: "RSA"
    size: 4096
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: istio-cacerts
  namespace: istio-system
spec:
  secretName: cacerts
  duration: 720h # 30d
  renewBefore: 360h # 15d
  commonName: cluster-2.demo.example.com
  isCA: true
  usages:
    - digital signature
    - key encipherment
    - cert sign
  dnsNames:
    - cluster-2.demo.example.com
  issuerRef:
    kind: ClusterIssuer
    name: self-signed-issuer
EOF
```

- Verify certificates were created

```
kubectl get certificates --context cluster-2 -n gloo-mesh
kubectl get certificates --context cluster-2 -n istio-system
```

Note if certificates were not generated it may be beneficial to look at the cert manager logs.

```
kubectl logs deploy/cert-manager --context cluster-2 -n cert-manager
kubectl logs deploy/cert-manager-istio-csr --context cluster-2 -n cert-manager
```

- Cleanup old Gloo certificates and allow cert-manager to replace them

```
kubectl delete secret relay-client-tls-secret --context cluster-2 -n gloo-mesh
kubectl delete secret relay-root-tls-secret --context cluster-2 -n gloo-mesh
kubectl delete secret relay-identity-token-secret --context cluster-2 -n gloo-mesh
```

- Update the Gloo Platform to use the new certificates

```
helm upgrade --install gloo-agent gloo-platform/gloo-platform \
  --version=2.3.9  \
  --namespace gloo-mesh \
  --kube-context cluster-2 \
  --reuse-values \
  -f data/gloo-agent-values.yaml
```

- Verify Gloo Agent connectivity

```
kubectl logs deploy/gloo-mesh-agent --context cluster-2 -n gloo-mesh
```

- Update Istio to use new CA certificate

```
kubectl delete secret cacerts --context cluster-1 -n istio-system
```

- Verify new cacerts is generated

```
kubectl get secret cacerts --context cluster-2 -n istio-system
```

- Restart Istiod to pick up new certificate

```
kubectl rollout restart deploy --context cluster-2 -n istio-system
```

- Verify new certificate is picked up by istiod

```
kubectl logs -l app=istiod --tail 500 --context cluster-2 -n istio-system| grep x509
```

- Restart Gateways

```
kubectl rollout restart deploy --context cluster-2 -n istio-ingress
kubectl rollout restart deploy --context cluster-2 -n istio-eastwest
```

- Restart workloads

```
kubectl rollout restart deploy -n online-boutique --context cluster-2
kubectl rollout restart deploy -n checkout-apis --context cluster-2
```

- Verify that the Gloo UI appears to be healthy

- Open the Gloo UI and observe the agents are connected and service discovery is working

```
kubectl port-forward svc/gloo-mesh-ui 8090:8090 --context management -n gloo-mesh
echo "Gloo UI: http://localhost:8090"
```

- Verify Online Boutique is functioning as expected

```
export GLOO_GATEWAY_HTTPS=$(kubectl --context cluster-1 -n istio-ingress get svc -l
istio=ingressgateway -o jsonpath='{.items[0].status.loadBalancer.ingress[0].*}'):443

echo "SECURE Online Boutique available at https://$GLOO_GATEWAY_HTTPS"
```

- Optional curl

```
curl -k --write-out '%{http_code}' https://$GLOO_GATEWAY_HTTPS
```