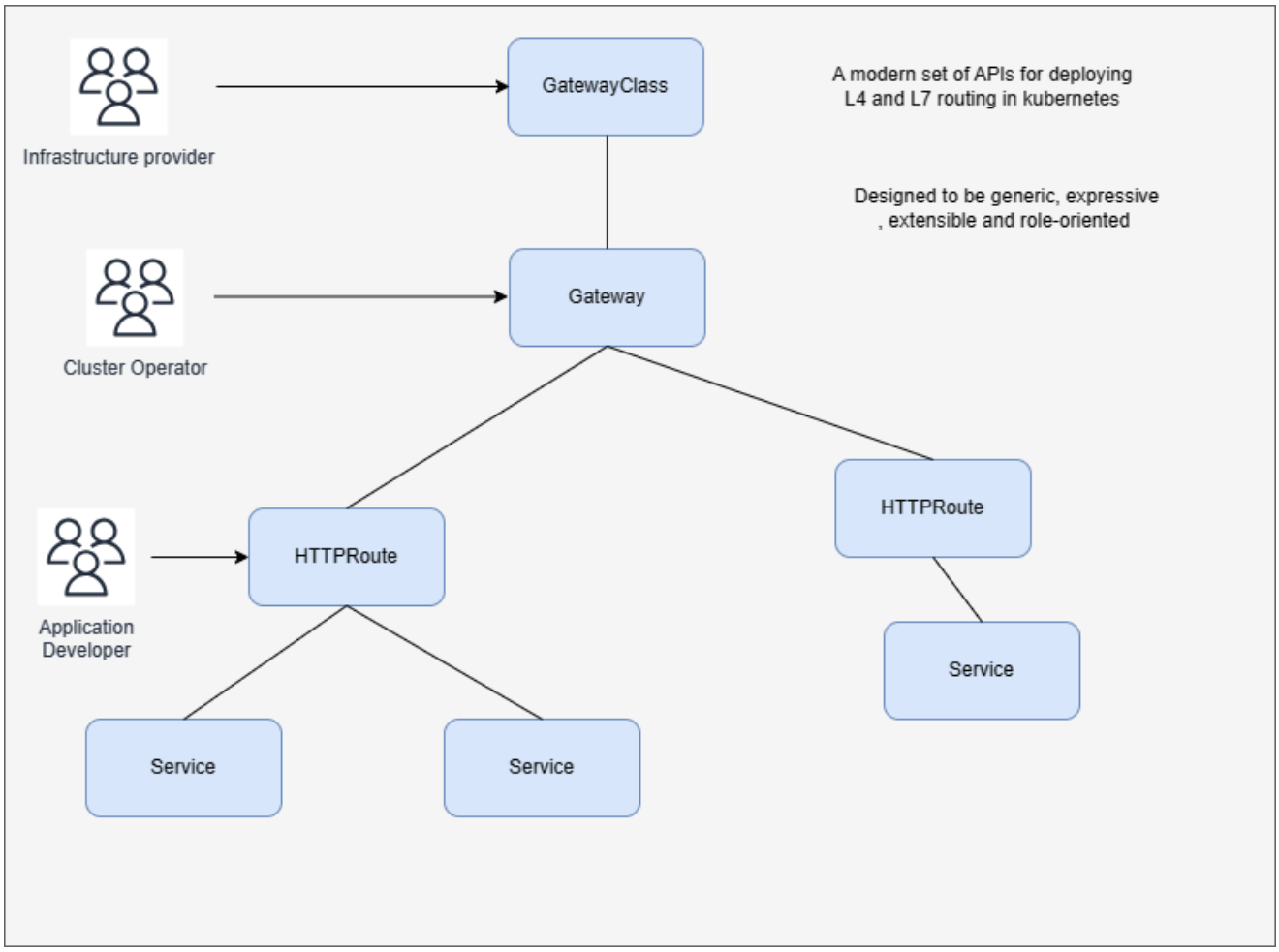
KUBERNETES API GATEWAY WITH ISTIO

Version	Description	Revision date	Amendment/ Modification/ Deletion	Reviewer Name	Approver Name
1.0	Initial Version	17/12/23	Amendment	Ashish Kapoor	Himanshu

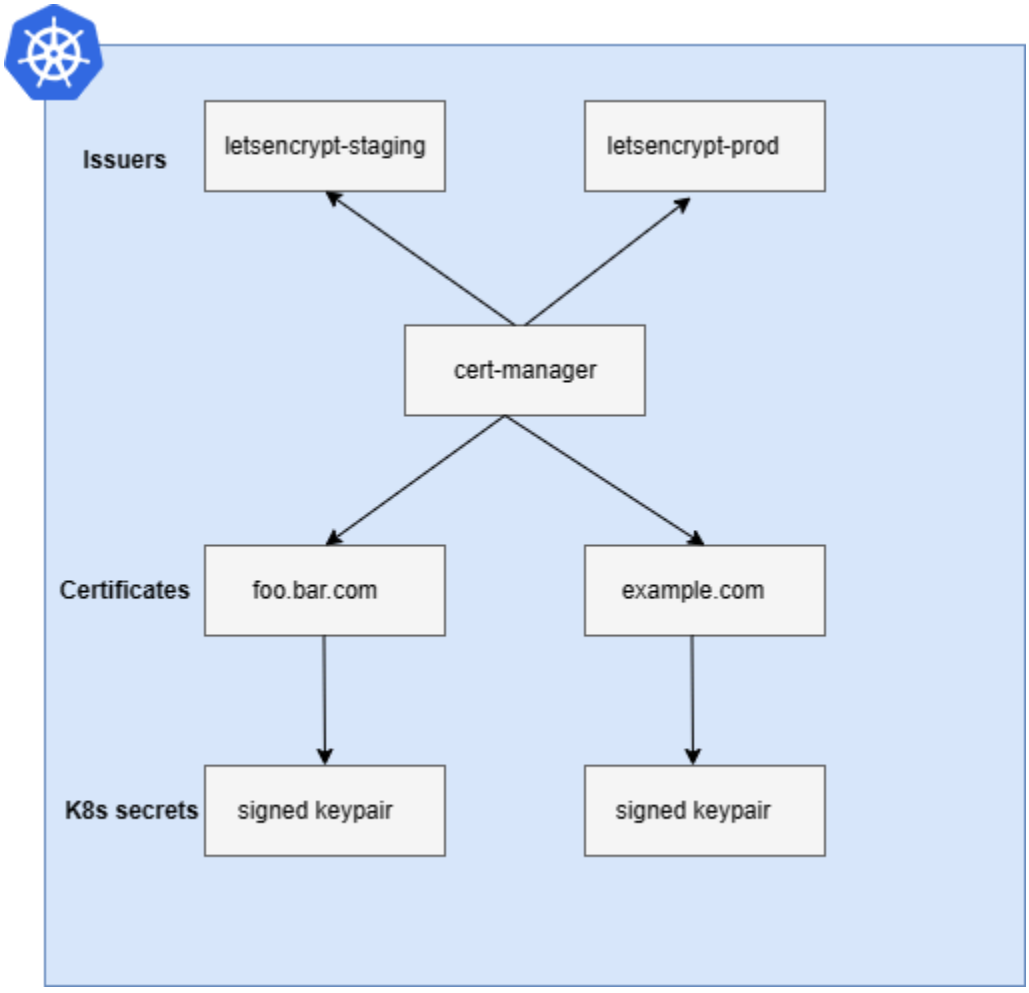
Introduction:

Gateway API is an open source project managed by the SIG-NETWORK community. It is an API (collection of resources) that model service networking in Kubernetes. These resources - GatewayClass, Gateway, HTTPRoute, TCPRoute, etc., as well as the Kubernetes Service resource - aim to evolve Kubernetes service networking through expressive, extensible, and role-oriented interfaces that are implemented by many vendors and have broad industry support.



CertManager :

cert-manager adds certificates and certificate issuers as resource types in Kubernetes clusters, and simplifies the process of obtaining, renewing and using those certificates.



Prerequisites:

Before you begin the Istio deployment process, make sure you have the following prerequisites:

- AWS CLI and kubectl installed
- Helm
- Amazon EKS cluster

n

Install cert manager

Add helm repo

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

Update your local Helm chart repository cache:

```
helm repo update
```

Install *CustomResourceDefinitions*

```
kubectl apply -f
https://github.com/cert-manager/cert-manager/releases/download/v1.13.
3/cert-manager.crds.yaml
```

Install cert-manager

```
helm install   cert-manager jetstack/cert-manager   --namespace
cert-manager   --create-namespace   --version v1.13.2   --set
```

```
"featureGates={UseCertificateRequestBasicConstraints=true}"
```

Configuring DNS01 Challenge Provider

ROUTE 53 :

Set up an IAM Role

cert-manager needs to be able to add records to Route53 in order to solve the DNS01 challenge. To enable this, create a IAM policy with the following permissions:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "route53:GetChange",
      "Resource": "arn:aws:route53:::change/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "route53:ChangeResourceRecordSets",
        "route53:ListResourceRecordSets"
      ],
      "Resource": "arn:aws:route53:::hostedzone/*"
    },
    {
      "Effect": "Allow",
      "Action": "route53:ListHostedZonesByName",
```

```
      "Resource": "*"
    }
  ]
}
```

IAM role trust policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Principal": {
        "Federated":
"arn:aws:iam::<aws-account-id>:oidc-provider/oidc.eks.<aws-region>.am
azonaws.com/id/<eks-hash>"
      },
      "Condition": {
        "StringEquals": {
          "oidc.eks.<aws-region>.amazonaws.com/id/<eks-hash>:sub":
"system:serviceaccount:<namespace>:<service-account-name>"
        }
      }
    }
  ]
}
```

Service annotation

Annotate the ServiceAccount created by cert-manager:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  annotations:
    warn:aws:iam::XXXXXXXXXXX:role/cert-manager
```

ISSUER

Issuers, and ClusterIssuers, are Kubernetes resources that represent certificate authorities (CAs) that are able to generate signed certificates by honoring certificate signing requests.

```
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: tea
spec:
  acme:
    server: https://acme-staging-v02.api.letsencrypt.org/directory
    privateKeySecretRef:
      name: teacerts
    solvers:
    - selector:
        dnsZones:
          - "teacoffe.api.trialanderror.in.net"
      dns01:
        route53:
          region: us-east-1
          hostedZoneID: <HSER4954309F> # optional, see policy above
          role: arn:aws:iam::0123456:role/xyz
```

- ClusterIssuer Definition:

- Configures a cert-manager ClusterIssuer named "tea" for managing TLS certificates in a Kubernetes cluster.
- Uses the ACME protocol with a Let's Encrypt staging server for testing.
- Specifies a Kubernetes Secret ("teacerts") for storing the private key.

- ACME Configuration:

- Utilizes the ACME protocol for automated certificate management.
- Sets the ACME server URL to Let's Encrypt staging server.
- References a Kubernetes Secret ("teacerts") to securely store the private key.
- DNS01 Challenge Solver Configuration:

- Implements DNS01 challenge solver for proving domain ownership during certificate issuance.
- Targets the DNS zone "teacoffe.api.trialanderror.in.net."
- Configures Amazon Route 53 as the DNS provider with specific settings like region, hosted zone ID, and IAM role.

Deploy sample *Httpbin.org* application

```
Kubectl apply -f sampleapp.yaml
```

GATEWAY

```
apiVersion: gateway.networking.k8s.io/v1beta1
kind: Gateway
metadata:
  name: gateway
  annotations:
    cert-manager.io/cluster-issuer: coffe
spec:
  gatewayClassName: nginx
  listeners:
  - name: http
    hostname: "teacoffe.api.trialanderror.in.net"
```

```
    port: 80
    protocol: HTTP
  - name: https
    hostname: "teacoffe.api.trialanderror.in.net"
    port: 443
    protocol: HTTPS
    tls:
      mode: Terminate
      certificateRefs:
      - kind: Secret
        name: cafe-secret
```

- Gateway Definition:

- Declares a Kubernetes resource of type Gateway in the gateway.networking.k8s.io/v1 API version.
- Names the Gateway resource "gateway" and associates it with the cert-manager ClusterIssuer named "coffe" using annotations.
- Specifies the gatewayClassName as "nginx" to indicate the class of the Ingress controller responsible for handling this Gateway.

- Listeners Configuration:

- Defines two listeners, one for HTTP on port 80 and another for HTTPS on port 443.
- The HTTP listener is named "http" and configured to handle traffic on the hostname "teacoffe.api.trialanderror.in.net" using the HTTP protocol.
- The HTTPS listener is named "https" and also handles traffic on the same hostname but using the HTTPS protocol.

- TLS Termination Configuration:

- For the HTTPS listener, specifies TLS termination using the tls section.
- mode: Terminate indicates that TLS termination should occur at the Gateway, and unencrypted traffic is forwarded internally.
- Refers to a TLS certificate stored in a Kubernetes Secret named "cafe-secret" using certificateRefs.

- The referenced Secret is expected to contain the necessary TLS certificate and private key for securing the HTTPS communication on the specified hostname

HTTPROUTE

```
---
apiVersion: gateway.networking.k8s.io/v1beta1
kind: HTTPRoute
metadata:
  name: api
spec:
  parentRefs:
  - name: gateway
  hostnames: ["teacoffe.api.trialanderror.in.net"]
  rules:
  - matches:
    - path:
        type: PathPrefix
        value: /
    backendRefs:
    - name: api
      port: 8000

---
apiVersion: gateway.networking.k8s.io/v1beta1
kind: HTTPRoute
metadata:
  name: http-filter-1
spec:
  parentRefs:
    - name: gateway
```

```
hostnames:
  - teacoffe.api.trialanderror.in.net
rules:
  - filters:
      - type: RequestRedirect
        requestRedirect:
          scheme: https
```

- HTTPRoute for API Endpoint:

- Defines a Kubernetes resource of type HTTPRoute in the gateway.networking.k8s.io/v1 API version.
- Names the HTTPRoute "api" and associates it with the previously defined Gateway named "gateway" using parentRefs.
- Specifies that this route is intended for the hostname "teacoffe.api.trialanderror.in.net."
- Configures a rule for matching any path (PathPrefix "/") and directs traffic to a backend service named "api" on port 8000.

- HTTPRoute for HTTPS Redirect:

- Declares another HTTPRoute named "http-filter-1" associated with the "gateway" resource.
- Targets the same hostname "teacoffe.api.trialanderror.in.net."
- Implements a rule with a filter of type RequestRedirect to enforce HTTPS redirection.
- The scheme: https directive ensures that any incoming HTTP requests to the specified hostname are redirected to HTTPS for enhanced security.

Access the application from the nlb created by service