A Deep Dive into Cilium Gateway API: The Future of Ingress Traffic Routing

蒋兴彦 (DaoCloud 道客)

Context

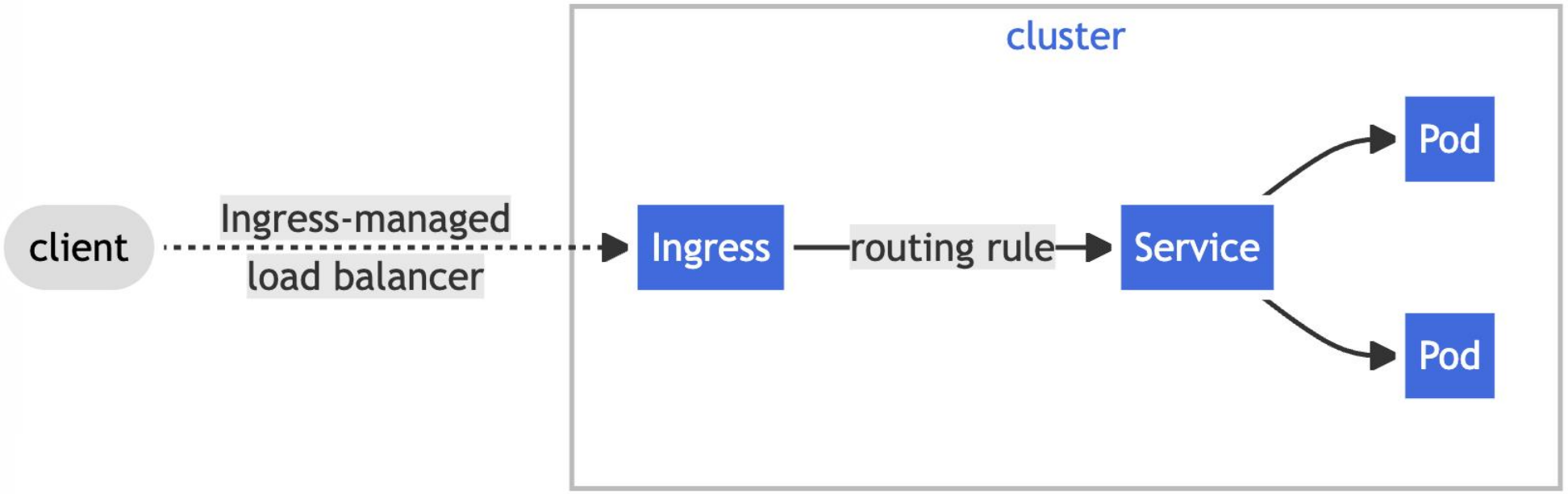
目录

Part 01

什么是Gateway API

回顾什么是 Ingress

Ingress 提供从集群外部到集群内服务的 HTTP 和 HTTPS 路由。 流量路由 由 Ingress 资源所定义的规则来控制。



Refer: https://kubernetes.io/zh-cn/docs/concepts/services-networking/ingress/

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: minimal-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx-example
  rules:
  - http:
      paths:
      - path: /testpath
        pathType: Prefix
        backend:
          service:
            name: test
            port:
              number: 80
```

这个 Ingress 资源将外部流量从 /testpath 转发到名为 "test" 的 Service 的 80 端口。

Refer: https://kubernetes.io/zh-cn/docs/concepts/services-networking/ingress/

什么是Gateway API？

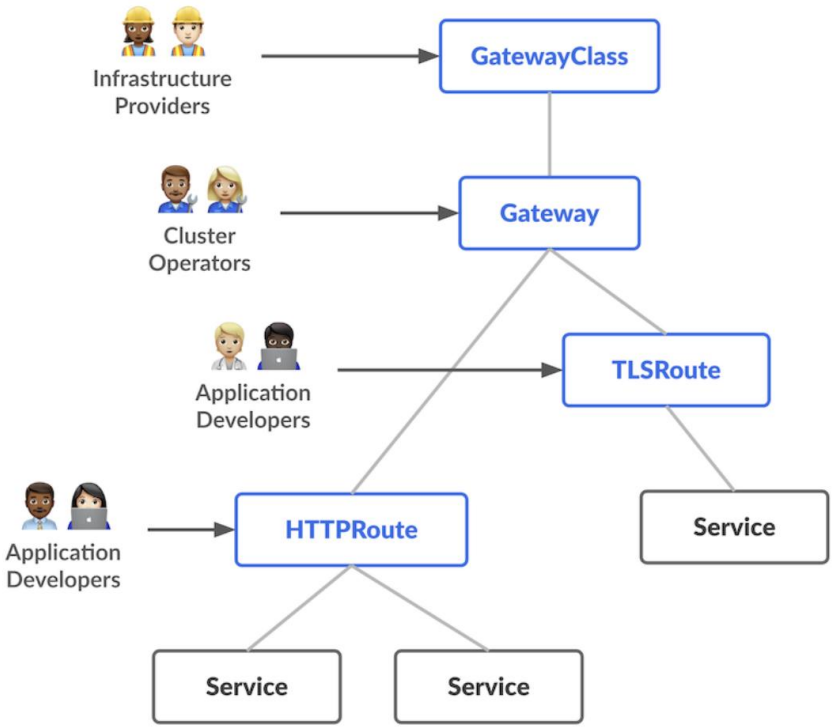
Gateway API 通过使用可扩展的、角色导向的、 协议感知的配置机制来提供网络服务。
它是一个附加组件， 可提供动态基础设施配置和高级流量路由的 API 。

设计原则
Gateway API 的设计和架构遵从以下原则：

- 角色导向： Gateway API 类别是基于负责管理 Kubernetes 服务网络的组织角色建模。
- 可移植： Gateway API 规范用自定义资源来定义， 并受到许多实现的支持。
- 表达能力强：Gateway API 类别支持常见流量路由场景的功能，
  例如基于Header的匹配、流量加权以及其他只能在 Ingress 中使用自定义注解才能实现的功能。
- 可扩展：Gateway 允许在 API 的各个层链接自定义资源。这使得在 API 结构内的适当位置进行精细定制成为可能。

Refer: https://gateway-api.sigs.k8s.io

什么是Gateway API？



Infrastructure Providers

管理使用多个独立集群为多个租户提供服务的基础设施，例如，云提供商。

Cluster Operators

管理集群，通常关注策略、网络访问、应用程序权限等。

Application Developers

管理在集群中运行的应用程序，
通常关注应用程序级配置和 Service 组合。

Refer: https://gateway-api.sigs.k8s.io

什么是Gateway API？


```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: prod-web
spec:
  gatewayClassName: example
  listeners:
  - protocol: HTTP
    port: 80
    name: prod-web-gw
    allowedRoutes:
      namespaces:
        from: Same
```

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: foo
spec:
  parentRefs:
  - name: prod-web
  rules:
  - backendRefs:
    - name: foo-svc
      port: 8080
```

Refer: https://gateway-api.sigs.k8s.io

Part 02

Gateway API vs Ingress

Gateway API vs Ingress

- **协议支持**: Ingress 仅适用于第 7 层协议,如 HTTP 和 HTTPS。非第 7 层支持需要使用自定义控制器扩展。
  Gateway API 原生支持第 4 层协议(如 TCP 和 UDP)以及第 7 层协议(如 HTTP)。

- **流量管理**: Ingress 内置的高级流量管理功能有限,如 A/B 测试或请求镜像,需要依赖供应商扩展和定制。
  Gateway API 提供内置支持,包括流量拆分、镜像、注入和细粒度指标。

- **可移植性**: Ingress 定义依赖于特定供应商,每个实现都有自己的语法和扩展能力。
  Gateway API 建立了一个通用标准,可在所有符合要求的控制器实现之间保持一致。

- **资源对象定义**: Ingress 规范中没有引入新的资源对象。
  Gateway API 引入了 GatewayClass、Gateway、HTTPRoute 等对象,
  用于定义功能、实例化和HTTP流量规则等。

- **路由自定义**: Ingress 只支持基于路径或主机的路由规则。
  Gateway API 允许基于任意头字段以及路径和主机进行路由自定义。

- **扩展功能**: 为 Ingress 添加身份验证策略或速率限制等功能需要依赖特定供应商的自定义注解和扩展。
  这些功能在 Gateway API 中都作为整体规范的一部分内置实现。

Gateway API vs Ingress

	Ingress	Gateway API
Protocol Support	HTTP/HTTPS only	L4 & L7 support
Traffic Management	Limited, vendor extensions required	Built-in advanced support
Portability	Vendor specific definitions	Standardized across implementations
Resource Objects	Ingress resource only	GatewayClass, Gateway, HTTPRoute, etc.
Routing Rules	Host/path-based only	Header-based also supported
Extending Capabilities	Custom annotations needed	Built-in advanced functionality

It's important to note that Ingress is now frozen, and all new features are being added to the Gateway API going forward.

Part 03

Cilium Gateway API 样例展示

Gateway API 样例展示

```
# kubectl apply -f https://raw.githubusercontent.com/istio/istio/release-1.11/samples/bookinfo/platform/kube/bookinfo.yaml
# kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
details-v1-5498c86cf5-kjzkj       1/1     Running   0          2m39s
productpage-v1-65b75f6885-ff59g   1/1     Running   0          2m39s
ratings-v1-b477cf6cf-kv7bh        1/1     Running   0          2m39s
reviews-v1-79d546878f-r5bjz       1/1     Running   0          2m39s
reviews-v2-548c57f459-pld2f       1/1     Running   0          2m39s
reviews-v3-6dd79655b9-nhrnh       1/1     Running   0          2m39s
```

Refer: https://docs.cilium.io/en/stable/network/servicemesh/gateway-api/http

Gateway API 样例展示

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: my-gateway
spec:
  gatewayClassName: cilium
  listeners:
  - protocol: HTTP
    port: 80
    name: web-gw
    allowedRoutes:
      namespaces:
        from: Same
```

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: http-app-1
spec:
  parentRefs:
  - name: my-gateway
    namespace: default
  rules:
  - matches:
    - path:
        type: PathPrefix
        value: /details
    backendRefs:
    - name: details
      port: 9080
```

Refer: https://docs.cilium.io/en/stable/network/servicemesh/gateway-api/http

Gateway API 样例展示

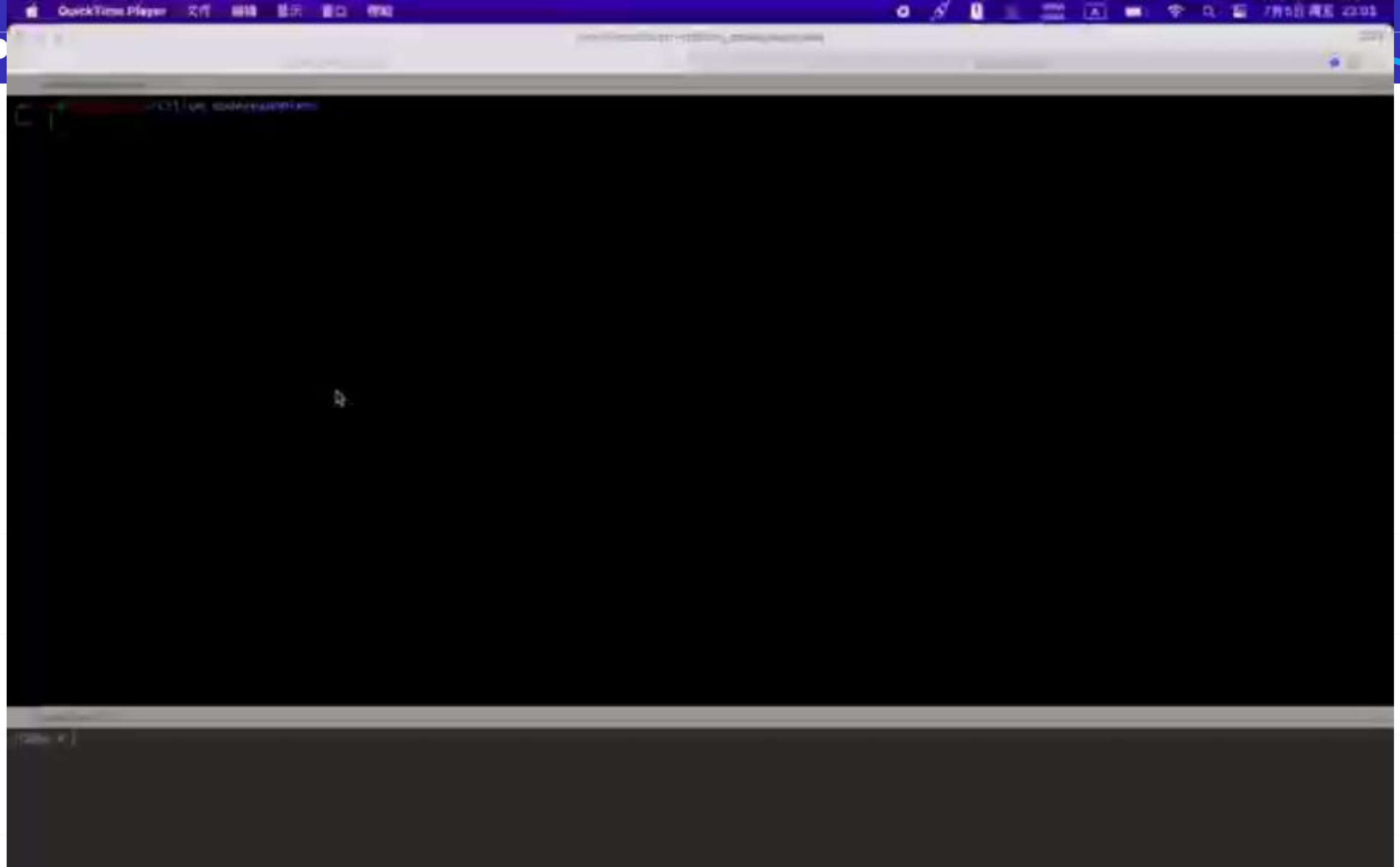
```
# kubectl get gateway
NAME           CLASS    ADDRESS         PROGRAMMED   AGE
my-gateway     cilium   172.18.0.131    True         7d23h
```

直接访问 gateway的地址。

```
#  curl 172.18.0.131/details/1 |jq
{
  "id": 1,
  "author": "William Shakespeare",
  "year": 1595,
  "type": "paperback",
  "pages": 200,
  "publisher": "PublisherA",
  "language": "English",
  "ISBN-10": "1234567890",
  "ISBN-13": "123-1234567890"
}
```

Refer: https://docs.cilium.io/en/stable/network/servicemesh/gateway-api/http

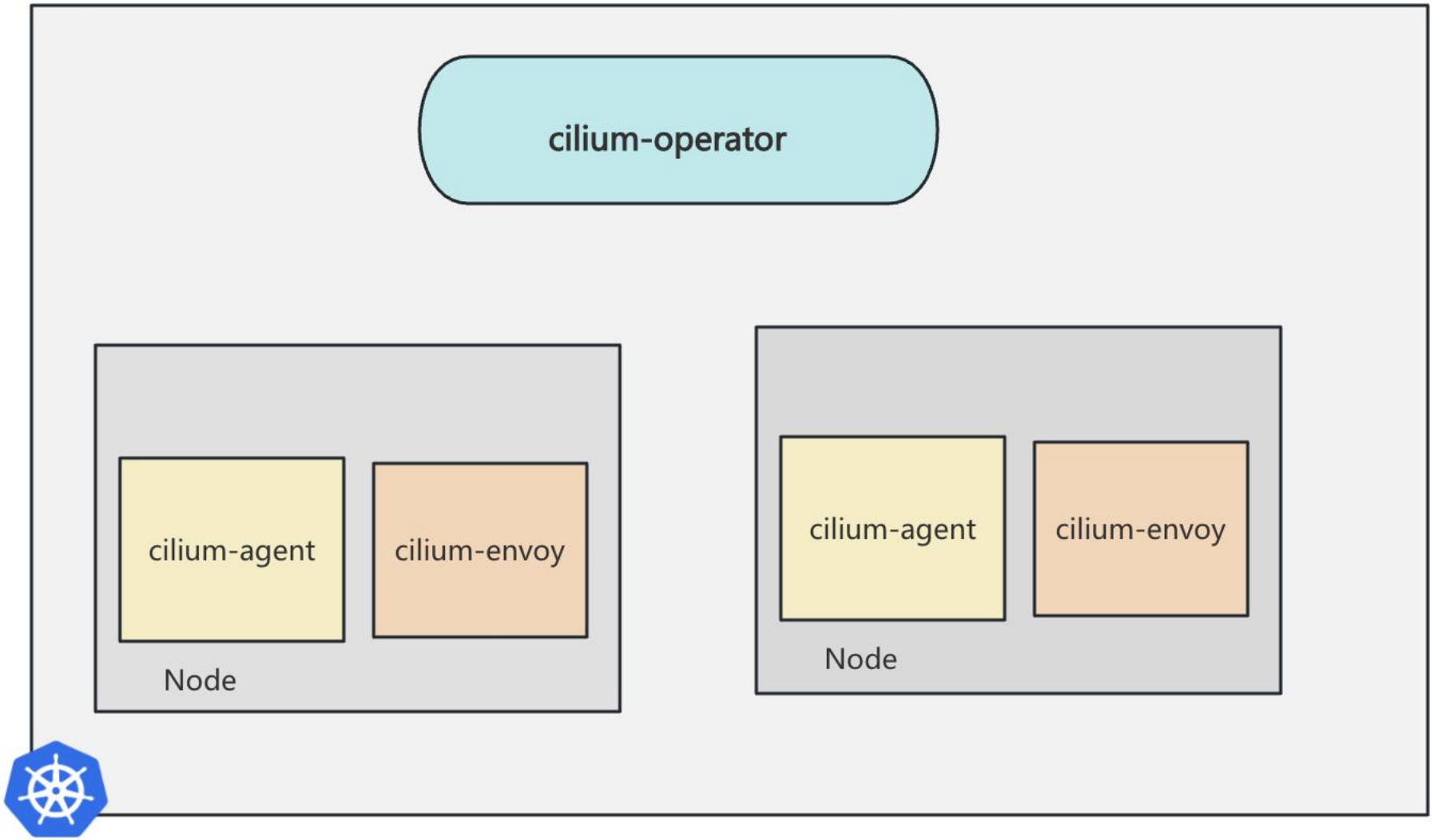
Demo

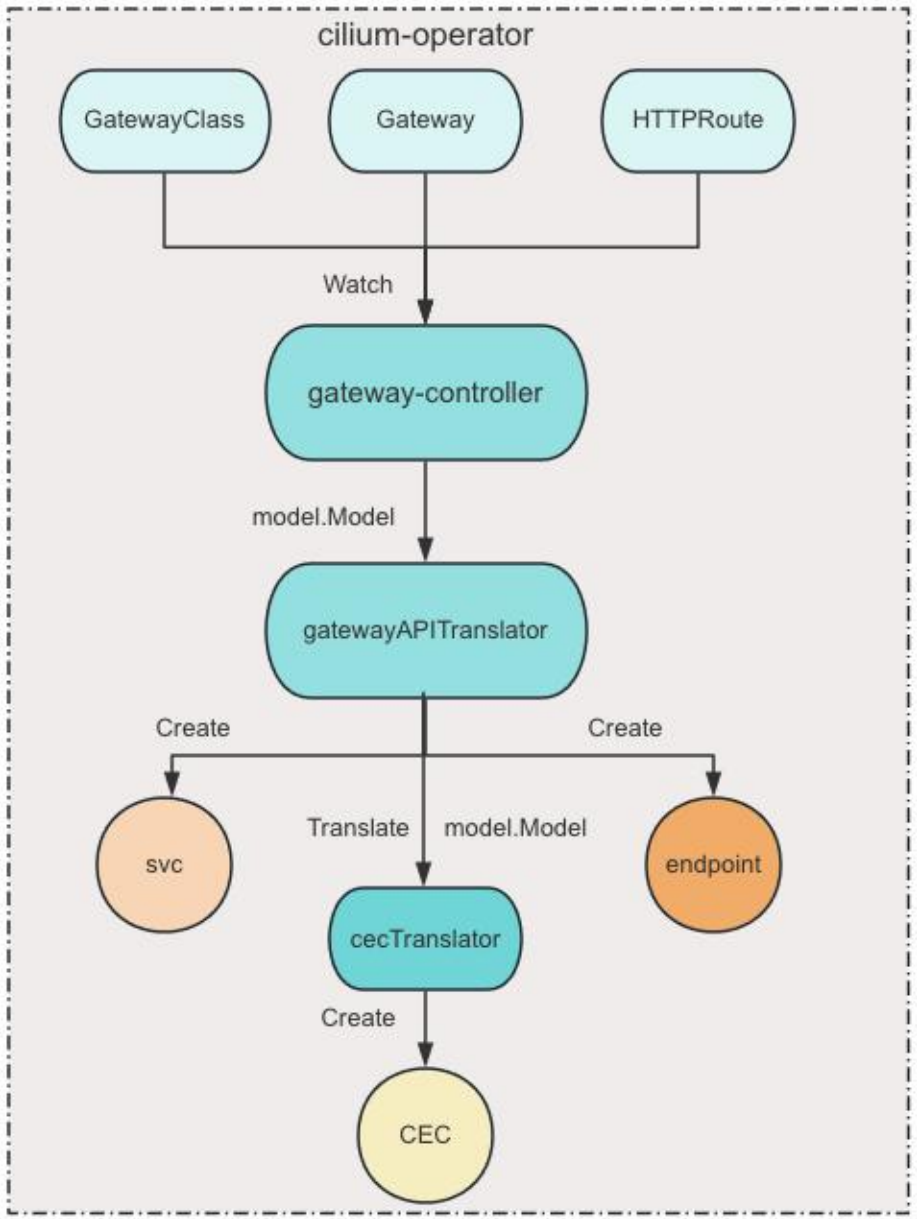


Part 04

Cilium Gateway API 原理解析

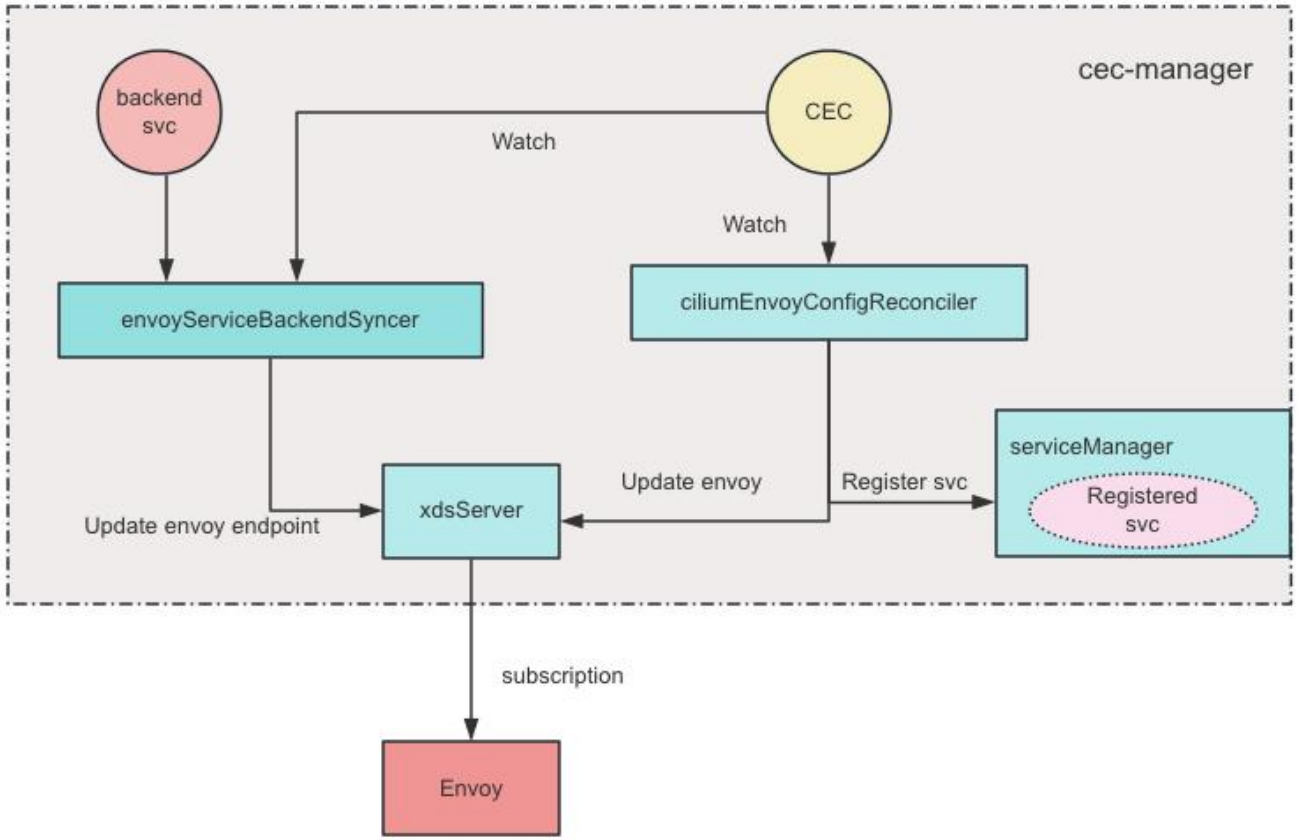
cilium 架构


cilium-operator
cilium-agent
cilium-envoy
Node
cilium-agent
cilium-envoy
Node



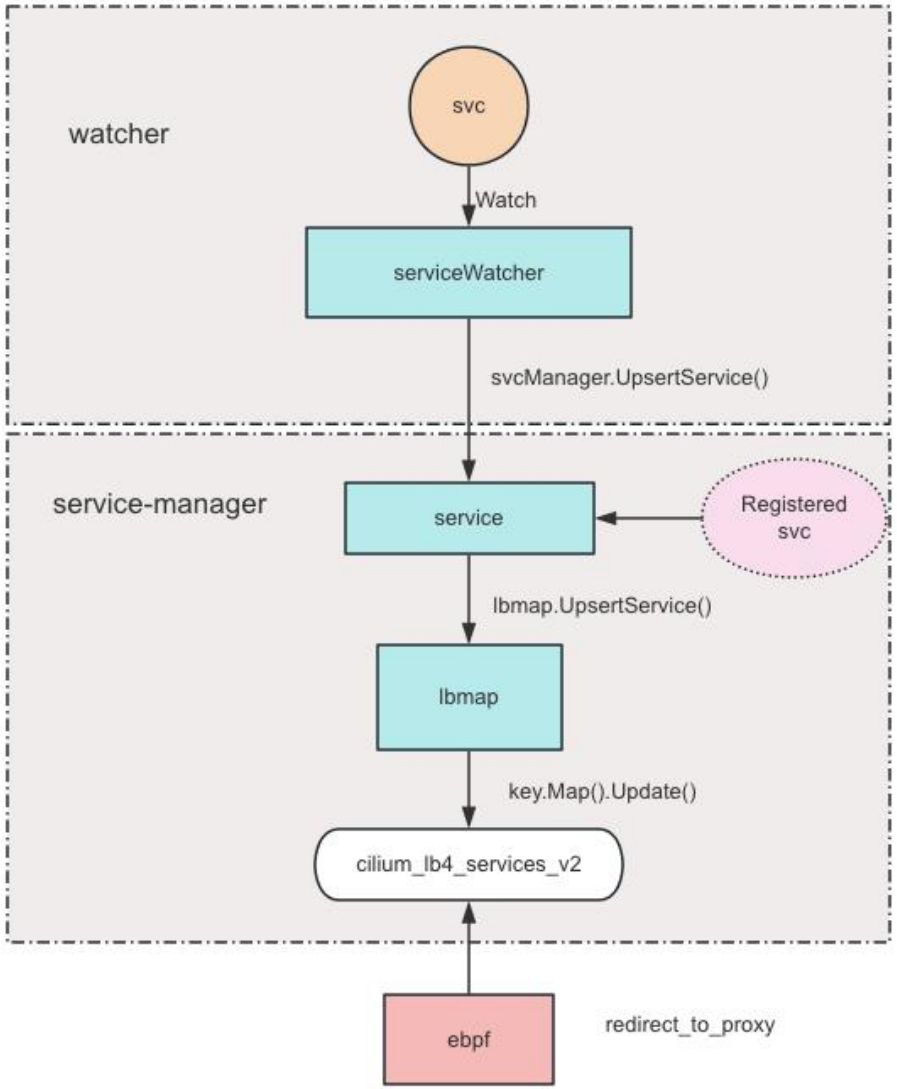

```
# kubectl get svc
NAME                          TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S
cilium-gateway-my-gateway     LoadBalancer   10.96.1.25      172.18.0.131    80:301
```

```
# kubectl get gateway
NAME           CLASS    ADDRESS         PROGRAMMED   AGE
my-gateway     cilium   172.18.0.131    True         7d23h
```




```yaml
apiVersion: cilium.io/v2
kind: CiliumEnvoyConfig
metadata:
  name: cilium-gateway-my-gateway
spec:
  backendServices:
  - name: details
    namespace: default
    number:
    - "9080"
  resources:
  - '@type': type.googleapis.com/envoy.config.listener.v3.Listener
```


- endpoint_config
  - @type : type.googleapis.com/envoy.config.endpoint.v3.ClusterLoadAssignment
  - cluster_name : default/details:9080
  - endpoints :
    - 0
      - locality
      - lb_endpoints :
        - 0
          - endpoint
            - address
              - socket_address
                - address : 10.244.1.235
                - port_value : 9080
            - health_check_config
          - health_status : HEALTHY
          - load_balancing_weight : 1
      - load_balancing_weight : 0
  - policy



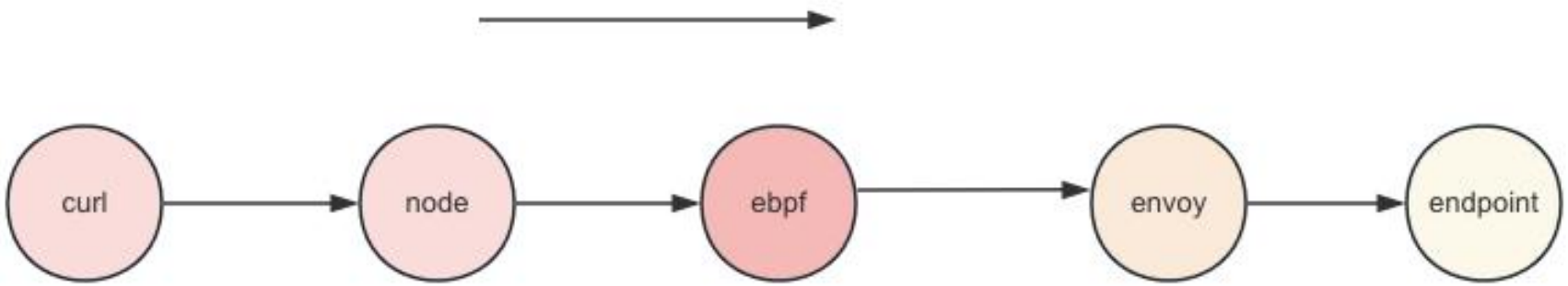

```
# cilium-dbg map get cilium_lb4_services_v2
Key                     Value                       State   Error
...
172.18.0.131:80 (0)     1580 0 (63) [0x60 0x4]      sync
...
```

```
ret = lb4_extract_tuple(ctx, ip4, ETH_HLEN, &l4_off, &tuple);
...
lb4_fill_key(&key, &tuple);

svc = lb4_lookup_service(&key, is_defined(ENABLE_NODEPORT));
if (svc) {
#if defined(ENABLE_L7_LB)
  if (lb4_svc_is_l7loadbalancer(svc)) {
    proxy_port = (__u16)svc->l7_lb_proxy_port;
    goto skip_service_lookup;
  }
```

```
# cilium-dbg bpf lb list
SERVICE ADDRESS          BACKEND ADDRESS (REVNAT_ID) (SLOT)
...
172.18.0.131:80 (                                    r] (L7LB Proxy Port: 11270)
...
```


overprovisioning_factor : 140
3
@type : type.googleapis.com/envoy.admin.v3.ListenersConfigDump
version_info : 6
dynamic_listeners :
0
1
2
filter_chains :



感谢观看

云原生社区
Cloud Native Community