istio架构及各个组件介绍 - itanony - 博客园

cnblogs.com/itanony/p/11976340.html

istio架构及各个组件介绍

istio 整体架构图

Istio 服务网格从逻辑上分为数据平面和控制平面。

- 数据平面由一组智能代理（Envoy）组成，被部署为 sidecar。这些代理通过一个通用的策略和遥测中心（Mixer）传递和控制微服务之间的所有网络通信。
- 控制平面管理并配置代理来进行流量路由。此外，控制平面配置 Mixer 来执行策略和收集遥测数据。

※什么是sidecar？
sidecar 的中文意思为边车，可以参考下面这张图。
试想， 我们部署一个web 应用的服务，除了应用本身， 可能还需要监控，日志采集等功能，如果将这些功能统一放到一个容器中，web应用的体积会变得很大。 不便于维护(业务更新升级过程中，可能监控和日志采集并不需要更新)，这种情况下。 我们可以将监控和日志采集功能单独部署在这个Pod 中的另外container（容器）中， 这种将应用程序的功能划分为单独的进程可以被视为 Sidecar 模式

---

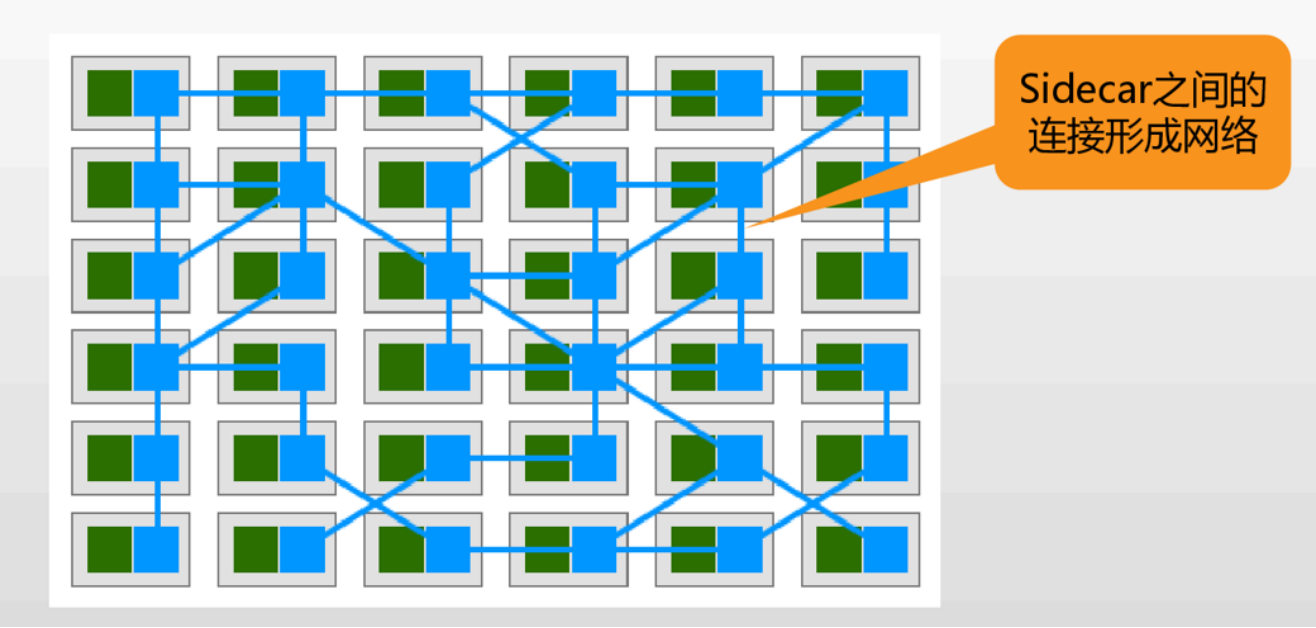
istio各个组件

部署完istio后， 我们可以在istio命名空间下找到istio 的各个组件，我们接下来会依次介绍这几个组件的功能和工作方式。

```
grafana-5d9dc9b755-ch7qf                  1/1       Running   0          4d
istio-citadel-668bc9c9d8-bs275            1/1       Running   0          7d
istio-egressgateway-6f9444c7b-rvlg8       1/1       Running   0          7d
istio-galley-74d4b75775-xpb6q             1/1       Running   0          8h
istio-ingressgateway-558d965db4-dbxq2     1/1       Running   0          4d
istio-pilot-7964cb6958-qqkcn              2/2       Running   0          7d
istio-policy-965b7db48-rmf9s              2/2       Running   0          7d
istio-sidecar-injector-6f96c6f496-4vpbn   1/1       Running   0          7d
istio-telemetry-56cbfcfc85-jvcjq          2/2       Running   0          7d
istio-tracing-7d76cf8d76-n2bnp            1/1       Running   0          7d
kiali-7b46bf9f99-2z6kp                    1/1       Running   0          4d
prometheus-8689fbf8bf-wdwcm               1/1       Running   0          4d
```

istio-proxy（envoy)

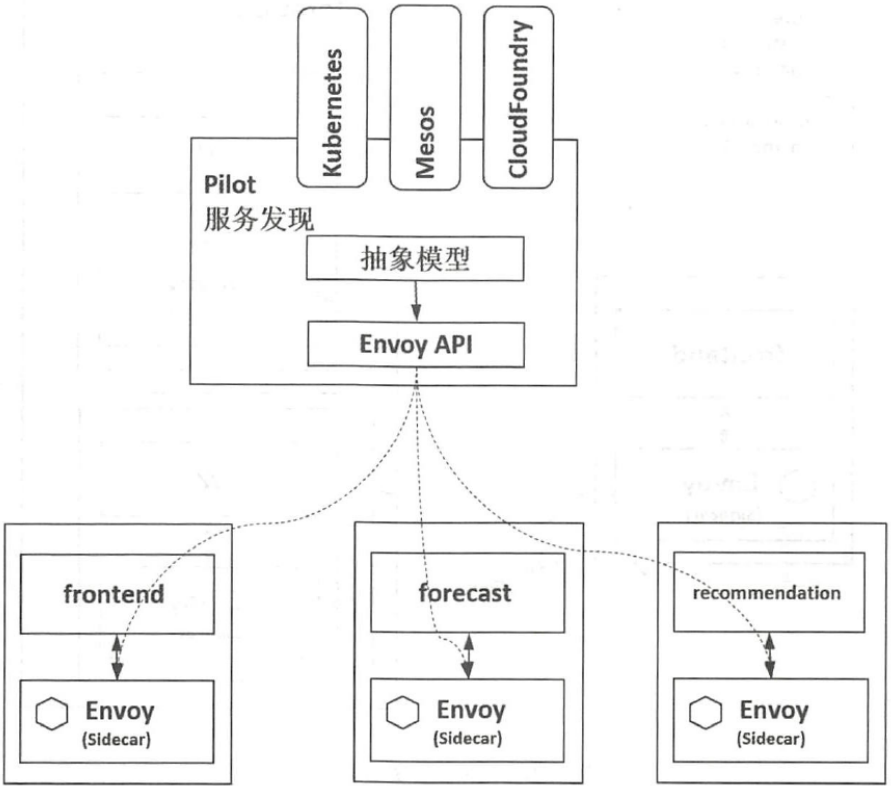
istio 的sidecar（istio-proxy）是开源项目envoy的扩展版，Envoy是用C＋＋开发的非常有影响力的轻量级高性能开源服务代理。作为服务网格的数据面，是istio架构中唯一的数据面组件， Envoy 提供了动态服务发现、负载均衡、TLS , HTTP/2 及gRPC 代理、熔断器、健康检查、流量拆分、灰度发布、故障注入等功能。在istio-proxy容器中除了有Envoy ，还有一个pilot-agent的守护进程。



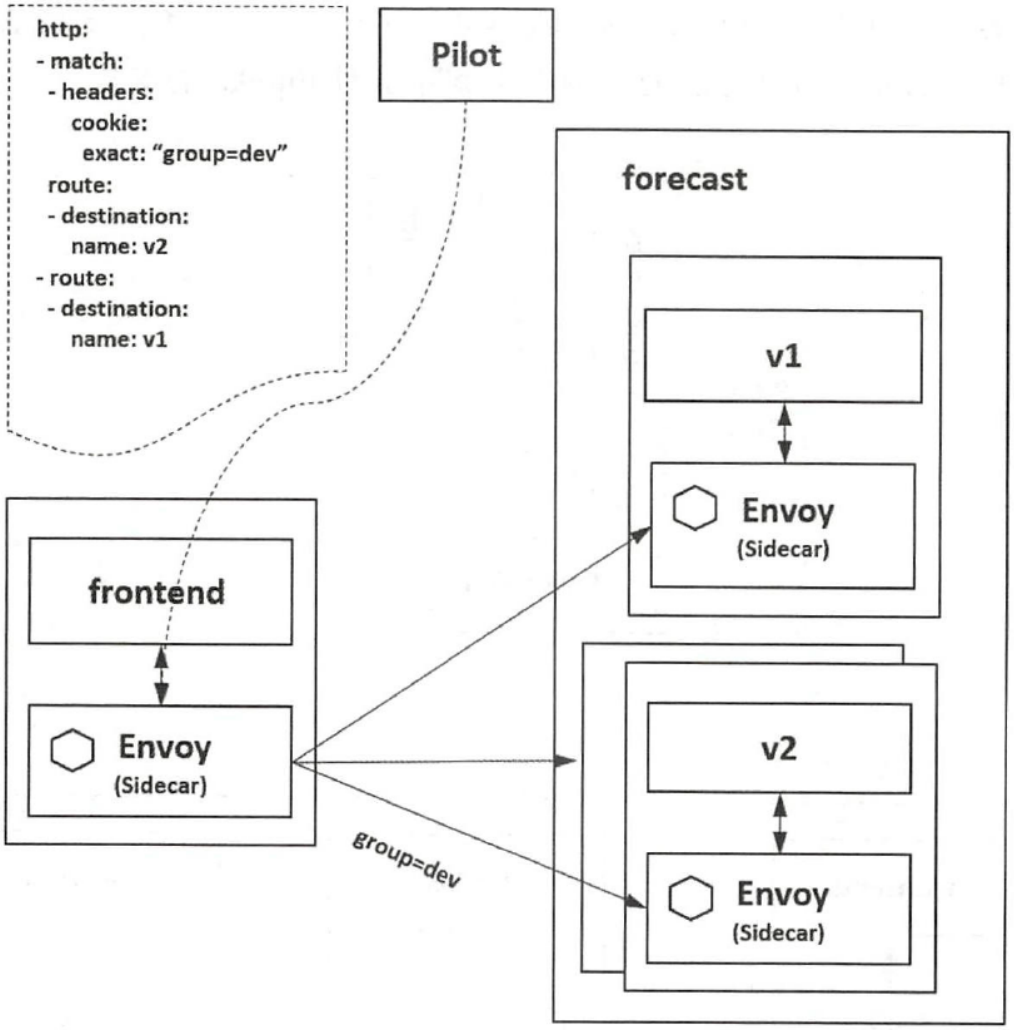
pilot

服务列表中的istio-pilot是istio的控制中枢Pilot服务。如果把数据面的envoy 也看作一种agent， 则Pilot 类似传统C /S 架构中的服务端Master，下发指令控制客户端完成业务功能。和传统的微服务架构对比， Pilot 至少涵盖服务注册中心和Config Server等管理组件的功能。

如下图所示：pilot直接从运行平台(kubernetes,consul)提取数据并将其构造和转换成istio的服务发现模型， 因此pilot只有服务发现功能，无须进行服务注册。这种抽象模型解耦Pilot和底层平台的不同实现，可支持kubernetes，consul等平台



除了服务发现， Pilot 更重要的一个功能是向数据面下发规则，包括VirtualService 、DestinationRule 、Gateway 、ServiceEntry 等流量治理规则，也包括认证授权等安全规则。Pilot 负责将各种规则转换成Envoy 可识别的格式，通过标准的XDS 协议发送给Envoy,指导Envoy 完成功作。在通信上， Envoy 通过gRPC 流式订阅Pilot 的配置资源。如下图所示，Pilot 将VirtualService 表达的路由规则分发到Evnoy 上， Envoy 根据该路由规则进行流量转发。



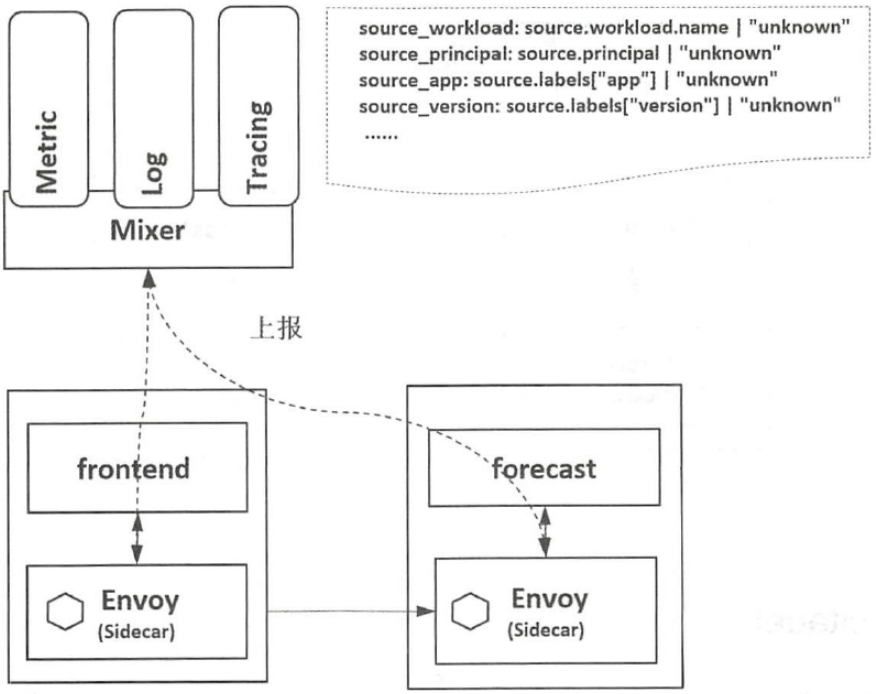
mixer

Istio 控制面部署了两个Mixer 组件： istio-telemetry 和istio-policy ，分别处理遥测数据的收集和策略的执行。查看两个组件的Pod 镜像会发现，容器的镜像是相同的，都是"/istio/mixer"

Mixer 是Istio 独有的一种设计,不同于Pilot ，在其他平台上总能找到类似功能的服务组件。从调用时机上来说,Pilot 管理的是配置数据，在配置改变时和数据面交互即可；然而，对于Mixer 来说，在服务间交互时Envoy 都会对Mixer 进行一次调用，因此这是一种实时管理。当然，在实现上通过在Mixer 和Proxy 上使用缓存机制，可保证不用每次进行数据面请求时都和Mixer 交互。

1. **istio-telemetry**

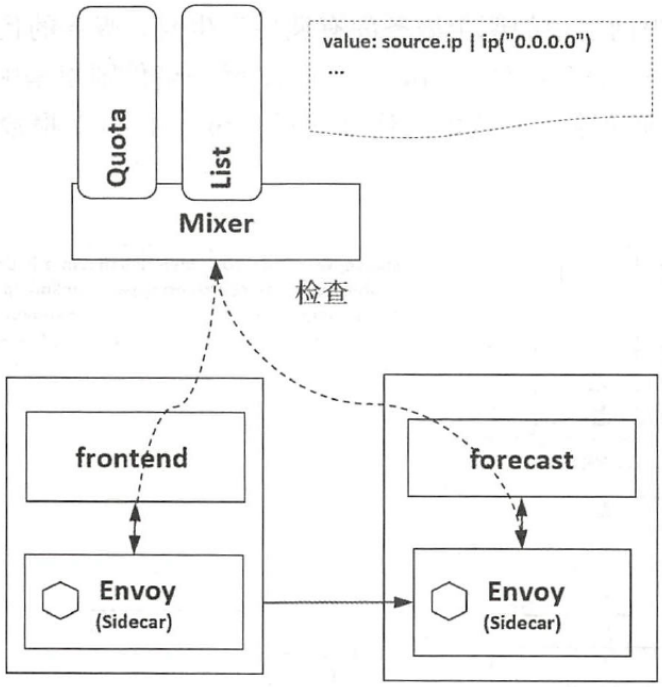
istio-telemetry是专门用于收集遥测数据的Mixer服务组件;如下图所示 所示，当网格中的两个服务间有调用发生时，服务的代理Envoy 就会上报遥测数据给istio-telemetry服务组件，istio-telemetry 服务组件则根据配置将生成访问Metric等数据分发给后端的遥测服务。数据面代理通过Report 接口上报数据时访问数据会被批
量上报。



2. **istio-policy**

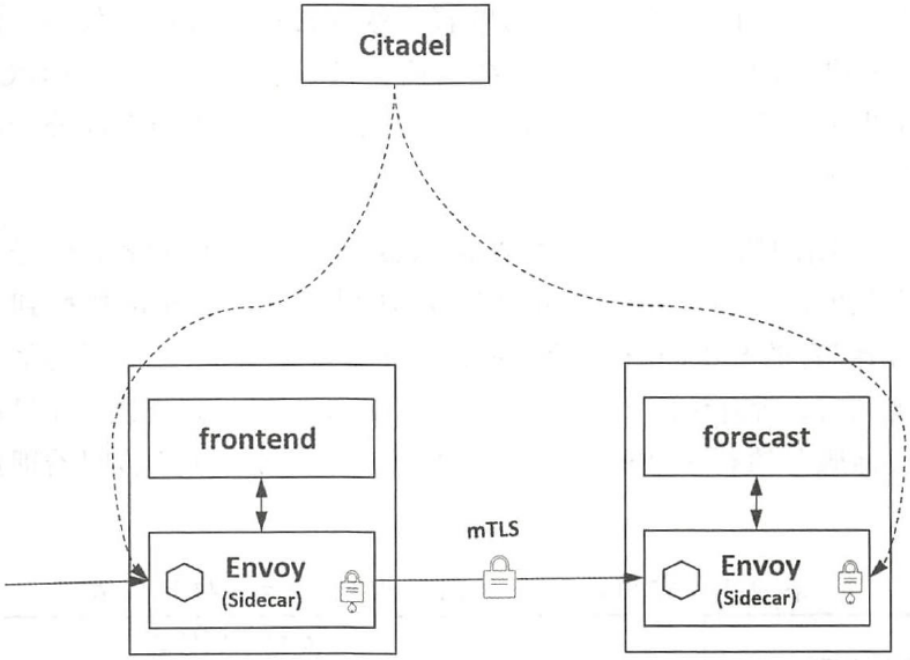
istio-policy 是另外一个Mixer 服务，和istio-telemetry 基本上是完全相同的机制和流程。

如图下图所示，数据面在转发服务的请求前调用istio-policy 的Check接口检查是否允许访问， Mixer 根据配置将请求转发到对应的Adapter 做对应检查，给代理返回允许访问还是拒绝。可以对接如配额、授权、黑白名单等不同的控制后端，对服务间的访问进行可扩展的控制。



istio-citadel

服务列表中的istio-citadel 是Istio 的核心安全组件，提供了自动生成、分发、轮换与撤销密钥和证书功能。Citadel 一直监听Kube-apiserver ，以Secret 的形式为每个服务都生成证书密钥，并在Pod 创建时挂载到Pod 上，代理容器使用这些文件来做服务身份认证，进而代理两端服务实现双向TLS认证、通道加密、访问授权等安全功能，这样用户就不用在代码里面维护证书密钥了。如下图 所示，frontend 服务对forecast 服务的访问用到了HTTP 方式，通过配置即可对服务增加认证功能， 双方的Envoy 会建立双向认证的TLS 通道，从而在服务间启用双向认证的HTTPS 。



istio-galley

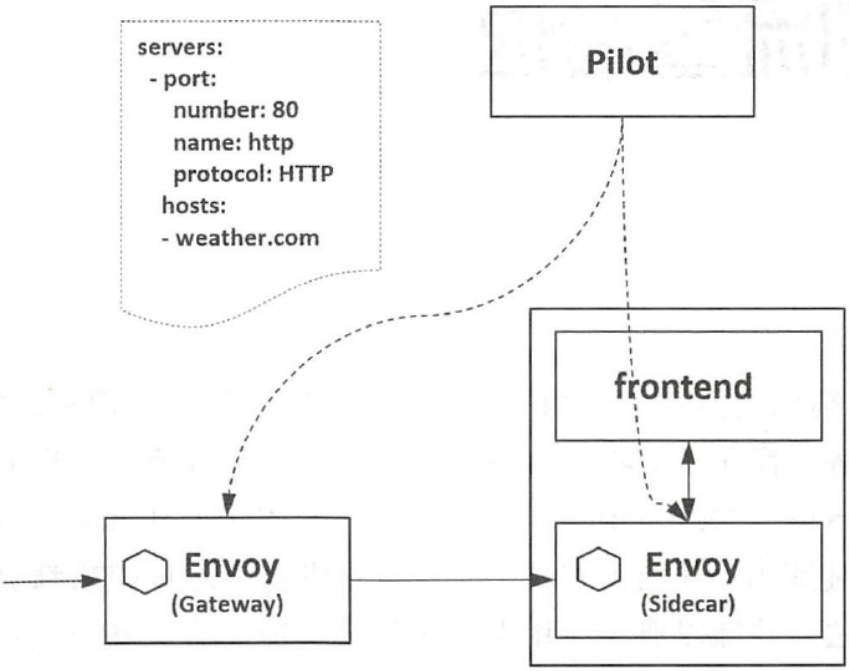
istio-galley 并不直接向数据面提供业务能力，而是在控制面上向其他组件提供支持。Galley 作为负责配置管理的组件，验证配置信息的格式和内容的正确性，并将这些配置信息提供给管理面的Pilot和Mixer服务使用，这样其他管理面组件只用和Galley 打交道，从而与底层平台解耦。在新的版本中Galley的作用越来越核心。

istio-sidecar-injector

istio-sidecar-inj ector 是负责向动注入的组件，只要开启了自动注入，在Pod 创建时就会自动调用istio-sidecar-injector 向Pod 中注入Sidecar 容器。
在Kubernetes环境下，根据自动注入配置， Kube-apiserver 在拦截到Pod 创建的请求时，会调用自动注入服务istio-sidecar-injector生成Sidecar 容器的描述并将其插入原Pod的定义中，这样，在创建的Pod 内除了包括业务容器，还包括Sidecar 容器。这个注入过程对用户透明，用户使用原方式创建工作负载。

istio-ingressgateway

istio-ingressgateway 就是入口处的Gateway ，从网格外访问网格内的服务就是通过这个Gateway 进行的。istio-ingressgateway 比较特别， 是一个Loadbalancer 类型的Service,不同于其他服务组件只有一两个端口,istio-ingressgateway 开放了一组端口，这些就是网格内服务的外部访问端口.如下图 所示，网格入口网关istio-ingressgateway 的负载和网格内的Sidecar 是同样的执行体，也和网格内的其他Sidecar 一样从Pilot处接收流量规则并执行。 。Istio 通过一个特有的资源对象Gateway 来配置对外的协议、端口等。



作者：轻易科技
出处：OPS