第 8 章　深度解读 Istio 微服务治理

本节所讲内容：

8.1 带你认识微服务
8.2　微服务架构发展进程
8.3　微服务框架对比分析
8.4　使用微服务需要解决的问题
8.5 Istio 概念、架构、组件详细介绍
8.6 为什么要用 Istio?
8.7 Istio 如何与 k8s 完美结合?
8.8 在 k8s 平台安装 Istio
实战 1：通过 Istio 部署在线书店

实验环境：k8s 集群：　k8s 的控制节点
ip：192.168.1.63
主机名：xuegod63
配置：64vCPU/6Gi 内存
k8s 的工作节点：
ip：192.168.1.64
主机名：xuegod64
配置：4vCPU/8Gi 内存

8.1　带你认识微服务

Istio 称的上是最新一代微服务，目前社区活跃度极高，跟 k8s 有很好的互补，那什么是微服务呢?

8.1.1　什么是微服务?

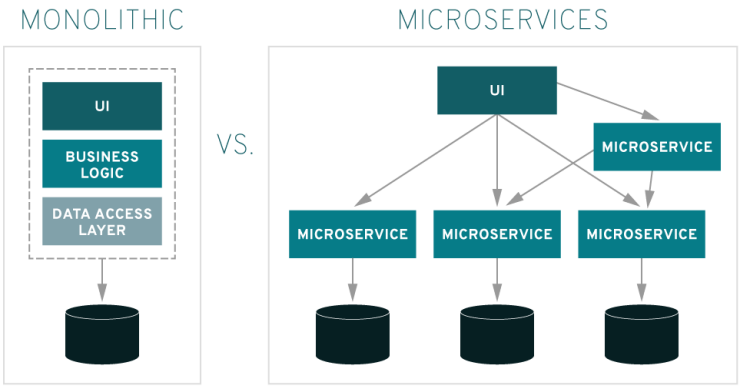
微服务可以从两个方面理解：

微：小的意思，大家可以搜索下 2 pizza（两个披萨原则），这个 2 pizza 原则最早是亚马逊 CEO 贝索斯提出来的，他认为如果两个披萨不足以喂饱一个项目团队，那么这个团队可能就显得太大了，因为人数过多的项目会议将不利于决策的形成，而让一个小团队在一起做项目、开会讨论，则更有利于达成共识，并能有效促进企业内部创新。

服务：相对较小且独立的功能单元

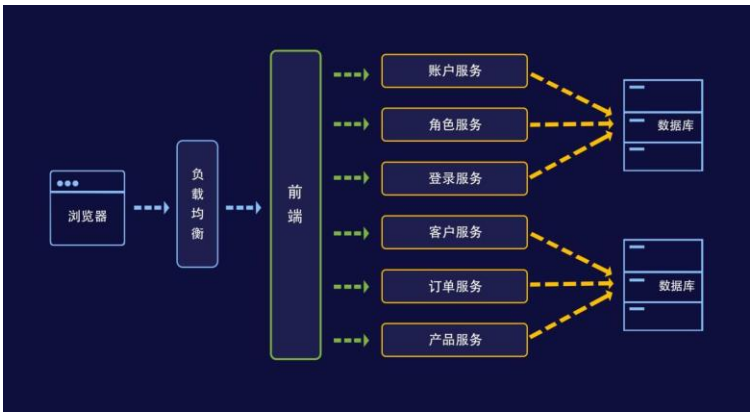
官方解释：

微服务是用于构建应用程序的架构风格，一个大的系统可由一个或者多个服务组成，微服务架构可将应用拆分成多个核心功能，每个功能都被称为一项服务，可以单独构建和部署，这意味着各项服务在工作和出现故障的时候不会相互影响。简单来说，微服务架构是把一个大的系统按照不同的业务单元分解成多个职责单一的小系统，并利用简单的方法使多个小系统相互协作，组合成一个大系统，各个小的系统是独立部署的，它们之间是松耦合的。

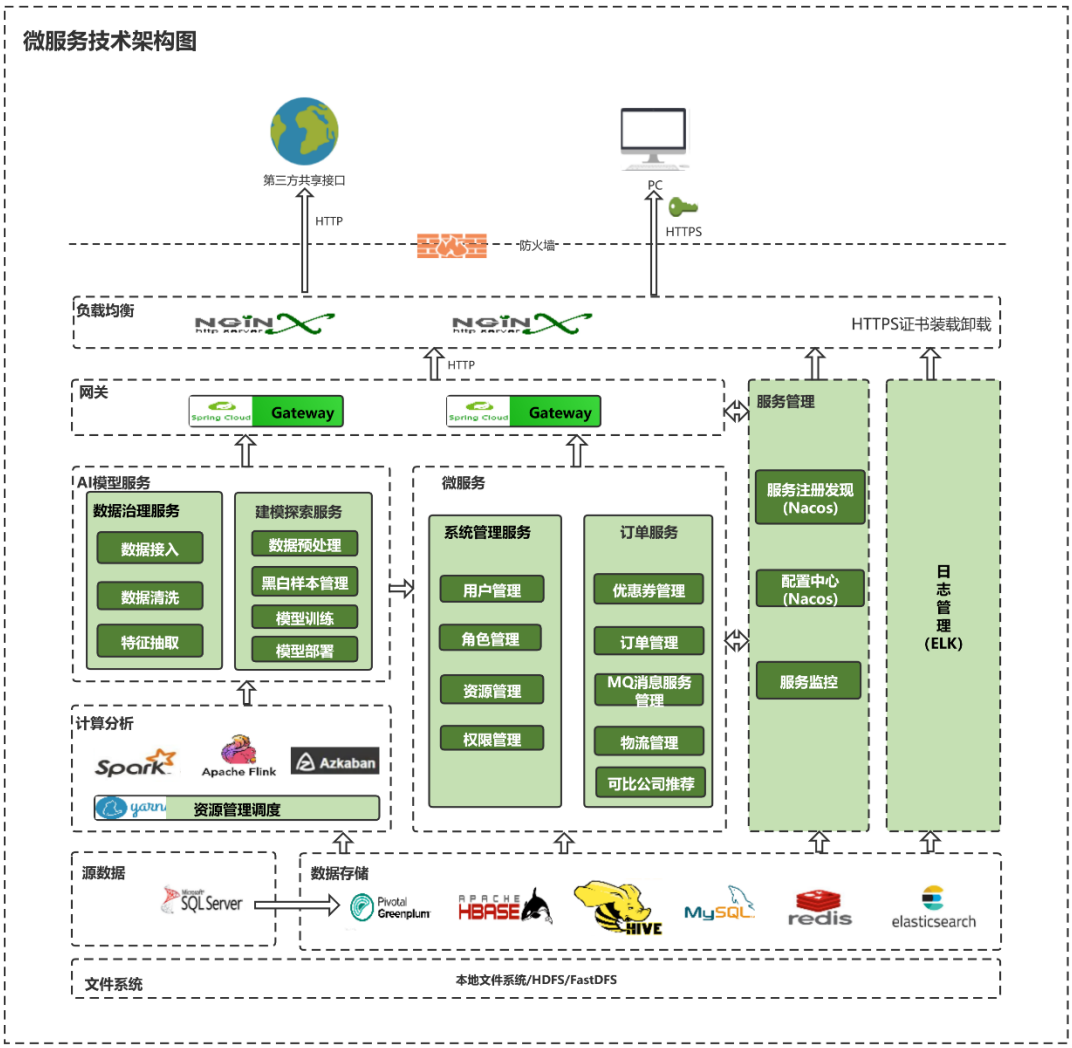


互动：回想一下在线购物时的情景

在使用网站上的搜索栏浏览产品时，这个搜索功能就是一项服务。
可以看到相关产品推荐，这些推荐项均来自数据库，这也是一项服务。
将商品添加到在线购物车中，这又是另一项服务。



微服务架构图



8.1.2 微服务架构的优势

1、可以让产品更快的上市

由于开发周期缩短，微服务架构有助于实现更加敏捷的部署和更新。

2、高度可扩展

随着某些服务的不断扩展，你可以跨多个服务器和基础架构进行部署，充分满足自身需求。

3、出色的弹性

只要确保正确构建，这些独立的服务就不会彼此影响。这意味着，一个服务出现故障不会导致整个应用下线。

4、易于部署

相对于传统的单体式应用，基于微服务的应用更加模块化且小巧，所以无需为它们的部署操心。

5、易于访问

由于大型应用被拆分成了多个小型服务，所以开发人员能够更加轻松地理解、更新和增强这些服务，从而缩短开发周期。

6、更加开放

由于使用了多语言 API，所以开发人员可以根据需要实现的功能，自由选用最适合的语言和技术。

8.2 微服务架构发展进程

第一代微服务框架

SpringCloud

springCloud 为开发者提供了快速构建分布式系统的通用模型的工具，主要是针对 java 的开发框架

第二代微服务框架

dubbo

Dubbo 是一个阿里巴巴开源出来的一个分布式服务框架，致力于提供高性能和透明化的 RPC 远程服务调用方案，以及 SOA 服务治理方案

第三代微服务框架

1.Service Mesh（服务网格）

istio 是开源的 Service Mesh（服务网格），Service Mesh 翻译成中文就是服务网格

8.3 微服务框架对比分析

主流微服务框架：SpringCloud、Dubbo

新锐微服务框架：k8s+istio

1、框架背景对比

（1）Spring Cloud，来源于 SpringSource ，具有 Spring 社区的强大背景支持，还有 Netflix 强大的后盾与技术输出。

扩展：Netflix 是什么?

Netflix 作为一家成功实践微服务架构的互联网公司，在几年前就把几乎整个微服务框架开源贡献给了社区，这些框架开源的整套微服务架构套件是 SpringCloud 的核心。包括：

Eureka：服务注册发现框架；

Zuul：服务网关；

Karyon：服务端框架；

Ribbon：客户端框架；

Hystrix：服务容错组件；

Archaius：服务配置组件；

Servo： Metrics 组件；

Blitz4j：日志组件。

（2）Dubbo 是一个分布式服务框架，是国内互联网公司阿里开放的微服务化治理框架，致力于提供高性能和透明化的 RPC 远程过程调用方案，以及 SOA 服务治理方案。 其核心部分包含）：

集群容错：提供基于接口方法的透明远程过程调用，包括多协议支持，以及软负载均衡，失败容错，地址路由，动态配置等集群支持；

自动发现：基于注册中心目录服务，使服务消费方能动态的查找服务提供方，使地址透明，使服务提供方可以平滑增加或减少机器。

扩展：RPC（(Remote Procedure Call)）？

简单的理解是一个节点请求另一个节点提供的服务

扩展：SOA（面向服务的架构）

面向服务的架构（SOA）是一个组件模型，它将应用程序的不同功能单元（称为服务）进行拆分，并通过这些服务之间定义良好的接口和协议联系起来。接口是采用中立的方式进行定义的，它应该独立于实现服务的硬件平台、操作系统和编程语言。这使得构建在各种各样的系统中的服务可以以一种统一和通用的方式进行交互。

（3）Istio 作为用于微服务服务聚合层管理的新锐项目，是 Google、IBM、Lyft（海外共享出行公司、Uber 劲敌） 首个共同联合开源的项目，提供了统一的连接，安全，管理和监控微服务的方案。

2、开源社区活跃度对比

开源社区情况：现如今企业在采用云计算首选开源，而选择一个开源框架，社区的活跃度将作为重要参考选项。

查看下在 Github 上的更新时间

Spring Cloud ： Spring Cloud • GitHub → 所有项目均更新于『1 小时』内。

Dubbo ： Dubbo • GitHub → 核心项目最近更新于『一个月乃至数月』前。

istio：istio • GitHub → 所有项目均更新于『30 分钟』内。

总结：结合项目背景、提供功能、社区更新活跃度，SpringCloud 是目前阶段发展最早的微服务框架方案，Istio 作为 Kubernetes 的优先支持来讲，也是一个值得关注的方案，而且发展潜力巨大，相信不久的将来 90%+的 k8s 用户都会使用 istio。

8.4 使用微服务需要解决的问题

8.4.1 配置中心

微服务为什么要解决配置中心问题?

在企业中，一般都有 2-4 种环境：开发，测试，交付，生产这四种，这几种环境的配置也有所变化，我们在部署的时候通常不能一个配置文件部署四个环境，这些环境的配置是不通用的。这就要用到 k8s 原生的配置中心 configMap 就是用解决这个问题的

常见的配置中心有哪些：Nacos、Apollo、SpringCloud Config、k8s configmap

配置管理流程：

配置的权限管理、灰度发布、版本管理、格式检验和安全配置等一系列的配置管理相关的特性也是配置中心不可获取的一部分。

1）权限管理

配置的变更和代码变更都是对应用运行逻辑的改变，重要的配置变更常常会带来核弹的效果， 通过项目的维度来对配置进行权限管理，一个项目的 owner 可以授权给其他用户配置的修改发布权限。

2）灰度发布

配置的灰度发布是配置中心比较重要的功能，当配置的变更影响比较大的时候，需要先在部分应用实例中验证配置的变更是否符合预期，然后再推送到所有应用实例。

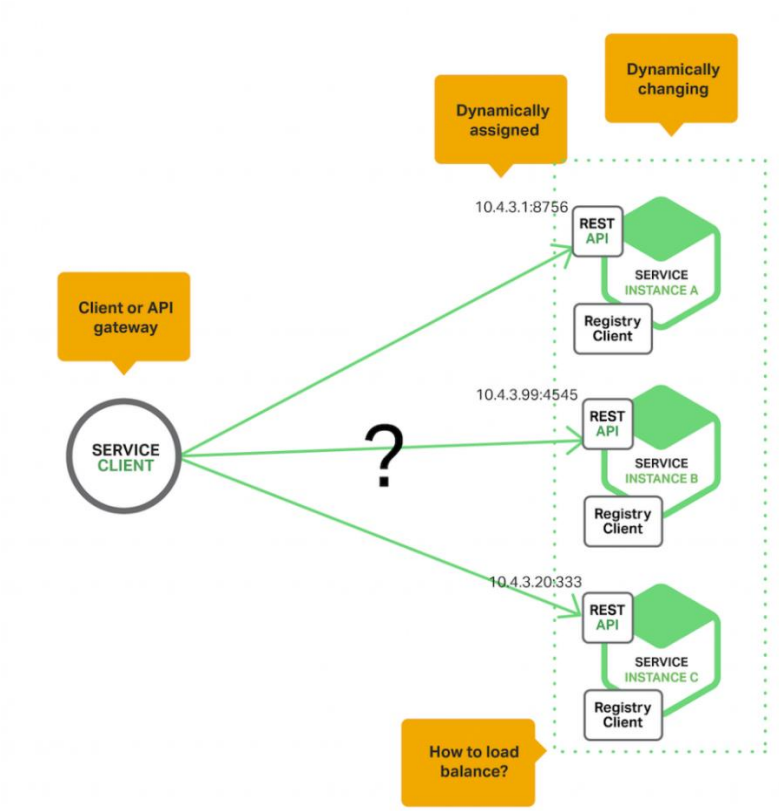
3）版本管理&回滚

当配置变更不符合预期的时候，需要根据配置的发布版本进行回滚。

4）配置格式校验

应用的配置数据存储在配置中心一般都会以一种配置格式存储，比如 Properties、Json、Yaml 等，如果配置格式错误，会导致客户端解析配置失败引起生产故障，配置中心对配置的格式校验能够有效防止人为错误操作的发生，是配置中心核心功能中的刚需。

8.4.2 服务发现



假设我们写的代码要调用 REST API 服务，代码需要知道服务实例的网络位置（IP 地址和端口）。运行在物理硬件上的传统应用中，服务实例的网络位置是相对固定的。如果服务部署到 k8s 或者容器中，服务地址是由集群系统动态分配的。这时我们需要访问这个服务时，如何确定它的地址呢？这时就需要服务发现（Service Discovery）了。

以 k8s 中部署应用为例：

Pod 是由生命周期的，如果 Pod 重启 IP 会发生变化。

如果我们的服务都是将 Pod 的 IP 地址写死，Pod 的挂掉或者重启，后端其他服务也将会不可用，那我们只能通过手动修改如 nginx 的 upstream 配置来改变后端的服务 IP。在我们可以通过，Consul，ZooKeeper 还有我们熟悉的 etcd 等工具，有了这些工具过后我们就可以只需要把我们的服务注册到这些服务发现中心去就可以。

Kubernetes 集群就为我们提供了这样的一个对象 - Service，Service 是一种抽象的对象，它定义了一组 Pod 的逻辑集合和一个用于访问它们的策略，其实这个概念和微服务非常类似。一个 Serivce 下面包含的 Pod 一般是由 Label Selector 来决定的。

我们这样就可以不用去管后端的 Pod 如何变化，只需要指定 Service 的地址就可以了，因为我们在中间添加了一层服务发现的中间件，Pod 销毁或者重启后，把这个 Pod 的地址注册到这个服务发现中心去。Service 的这种抽象就可以帮我们达到这种解耦的目的。

8.5 Istio 概念、架构、组件详细介绍

8.5.1 Istio 是什么?

Istio 是一个与 Kubernetes 紧密结合的适用于云原生场景下用于服务治理的开放平台。

服务治理包括:

连接（Connect)、安全（Secure)、策略执行（Control）和可观察性（Observe)。

1）连接:

Istio 通过集中配置的流量规则控制服务间的流量和调用，实现负载均衡、熔断、故障注入、重试、重定向等服务治理功能。

2）安全:

Istio 提供透明的认证机制、通道加密、服务访问授权等安全能力，可增强服务访问的安全性。

3）策略执行:

Istio 通过可动态插拔、可扩展的策略实现访问控制、速率限制、配额管理、服务计费等能力。

4）可观察性:

动态获取服务运行数据和输出，提供强大的调用链、监控和调用日志收集输出的能力。配合可视化工具，可方便运维人员了解服务的运行状态，发现并解决问题。

5）策略与遥测:

常常需要为服务设置一定的授权策略，比如限制流量的速率、设置黑名单等。另外，遥测（Telemetry）也是一个很重要的功能，可以通过分析收集到的指标（Metric）来监控系统的状态。在 Istio 中，策略设定和遥测都是通过 Mixer 组件完成的（mixer 在 1.5+已经被废弃了，整合到 istiod 中了)

8.5.2 Istio 核心特性

1、流控(traffic management)

断路器(circuit breakers)、超时、重试、多路由规则、AB 测试、灰度发布、按照百分比分配流量等。

2、安全(security)

加密、身份认证、服务到服务的权限控制、K8S 里容器到容器的权限控制等。

3、可观察(observability)

追踪、监控、数据收集，通过控制后台全面了解上行下行流量，服务链路情况，服务运行情况，系统性能情况，国内微服务架构体系，这一块做得比较缺乏。

4、平台无关系(platform support)

K8s，物理机，自己的虚机都没问题。

5、集成与定制(integration and customization)

可定制化扩展功能。

1.2.1 断路器

互动 1：举个生活中的例子解释断路器

当电路发生故障或异常时，伴随着电流不断升高，并且升高的电流有可能能损坏电路中的某些重要

器件，也有可能烧毁电路甚至造成火灾。若电路中正确地安置了保险丝，那么保险丝就会在电流异常升高到一定的高度和热度的时候，自身熔断切断电流，从而起到保护电路安全运行的作用。

很多技术都是来源生活的，随着社会进步，科技发展

断路器也称为服务熔断，在多个服务调用的时候，服务 A 依赖服务 B，服务 B 依赖服务 C，如果服务 C 响应时间过长或者不可用，则会让服务 B 占用太多系统资源，而服务 A 也依赖服 B，同时也在占用大量的系统资源，造成系统雪崩的情况出现。 Istio 断路器通过网格中的边车对流量进行拦截判断处理，避免了在代码中侵入控制逻辑，非常方便的就实服务熔断的能力。

在微服务架构中，在高并发情况下，如果请求数量达到一定极限（可以自己设置阈值），超出了设置的阈值，断路器会自动开启服务保护功能，然后通过服务降级的方式返回一个友好的提示给客户端。假设当 10 个请求中，有 10%失败时，熔断器就会打开，此时再调用此服务，将会直接返回失败，不再调远程服务。直到 10s 钟之后，重新检测该触发条件，判断是否把熔断器关闭，或者继续打开。

互动 2：服务降级（提高用户体验效果）

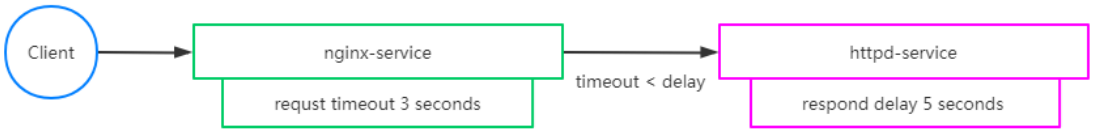
比如电商平台，在针对 618、双 11 的时候会有一些秒杀场景，秒杀的时候请求量大，可能会返回报错标志“当前请求人数多，请稍后重试”等，如果使用服务降级，无法提供服务的时候，消费者会调用降级的操作，返回服务不可用等信息，或者返回提前准备好的静态页面写好的信息。

1.2.2 超时

什么时候需要用到超时？

在生产环境中经常会碰到由于调用方等待下游的响应过长，堆积大量的请求阻塞了自身服务，造成雪崩的情况，通过超时处理来避免由于无限期等待造成的故障，进而增强服务的可用性。

通过例子来理解



nginx 服务设置了超时时间为 3 秒，如果超出这个时间就不在等待，返回超时错误

httpd 服务设置了响应时间延迟 5 秒，任何请求都需要等待 5 秒后才能返回

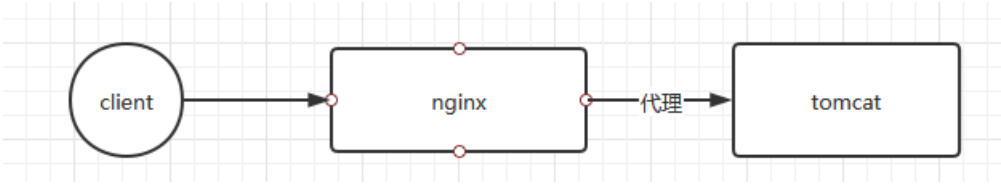
client 通过访问 nginx 服务去反向代理 httpd 服务，由于 httpd 服务需要 5 秒后才能返回，但 nginx 服务只等待 3 秒，所以客户端会提示超时错误。

1.2.3 重试

istio 重试机制就是如果调用服务失败，Envoy 代理尝试连接服务的最大次数。而默认情况下，

Envoy 代理在失败后并不会尝试重新连接服务。

举个例子：



客户端调用 nginx，nginx 将请求转发给 tomcat。tomcat 通过故障注入而中止对外服务，nginx 设置如果访问 tomcat 失败则会重试 3 次。

1.2.4 多路由规则

1、HTTP 重定向（HTTPRedirect）

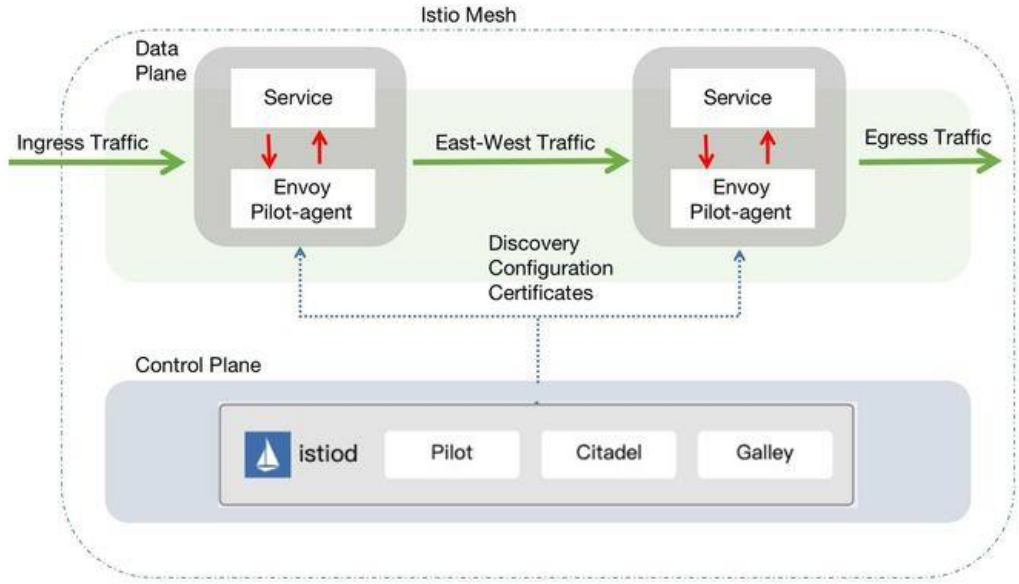
2、HTTP 重写（HTTPRewrite）

3、HTTP 重试（HTTPRetry）

4、HTTP 故障注入（HTTPFaultInjection）

5、HTTP 跨域资源共享（CorsPolicy）

8.5.3 Istio 架构



istio 服务网格从逻辑上分为数据平面和控制平面。

1、数据平面由部署为 Sidecar 的一组智能代理(Envoy+Polit-agent)组成。这些代理承载并控制微服务之间的所有网络通信，管理入口和出口流量，类似于一线员工。

2、控制平面管理并配置代理来进行流量路由。

3、istio1.5+中使用了一个全新的部署模式，重建了控制平面，将原有的多个组件整合为一个单体结构 istiod，这个组件是控制平面的核心，负责处理配置、证书分发、sidecar 注入等各种功能。istiod 是新版本中最大的变化，以一个单体组件替代了原有的架构，在降低复杂度和维护难度的同时，也让易用性得到提升。需要注意的一点是，原有的多组件并不是被完全移除，而是在重构后以模块的形式整合在一起组成了 istiod。

数据平面：

Envoy 和 pilot-agent 打在同一个镜像中，即 Proxy。

8.5.4 istio 组件

1、Envoy

Envoy 是什么?

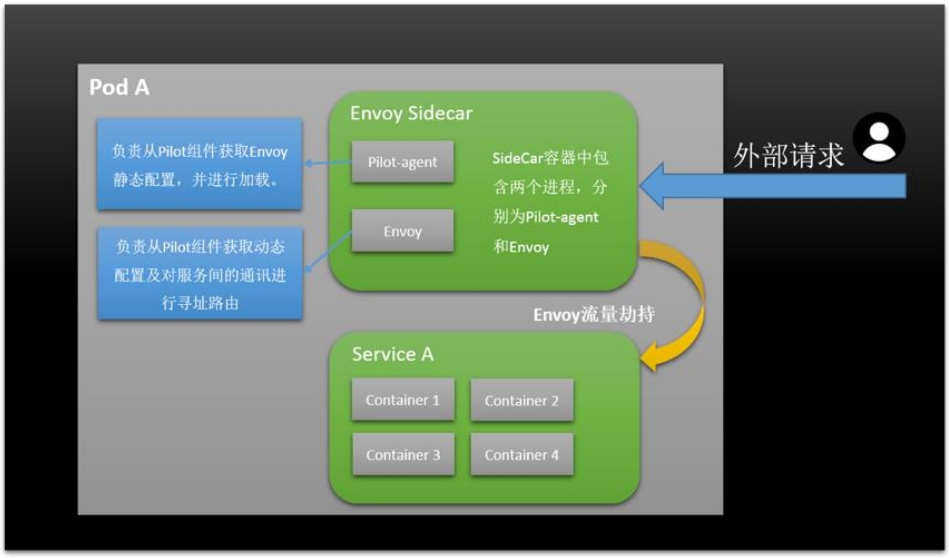
Envoy 是用 C++ 开发的高性能代理，用于协调服务网格中所有服务的入站和出站流量。

Envoy 有许多强大的功能，例如:

- 动态服务发现
- 负载均衡
- TLS 终端
- HTTP/2 与 gRPC 代理
- 熔断
- 健康检查
- 基于百分比流量拆分的灰度发布
- 故障注入
- 丰富的指标

Istio 中 Envoy 与服务什么关系?

为了便于理解 Istio 中 Envoy 与服务的关系，下图为 Envoy 的拓扑图，如图所示：



Envoy 和 Service A 同属于一个 Pod，共享网络和命名空间，Envoy 代理进出 Pod A 的流量，并将流量按照外部请求的规则作用于 Service A 中。

Pilot-agent 是什么?

Envoy 不直接跟 k8s 交互，通过 pilot-agent 管理的

Pilot-agent 进程根据 K8S APIserver 中的配置信息生成 Envoy 的配置文件，并负责启动 Envoy 进程。

Envoy 由 Pilot-agent 进程启动，启动后，Envoy 读取 Pilot-agent 为它生成的配置文件，然后根据该文件的配置获取到 Pilot 的地址，通过数据面从 pilot 拉取动态配置信息，包括路由（route），监听器（listener），服务集群（cluster）和服务端点（endpoint）。

2、Pilot

1）Pilot 为 Envoy 提供服务发现

2）提供流量管理功能（例如，A/B 测试、金丝雀发布等）以及弹性功能（超时、重试、熔断器等）；

3）生成 envoy 配置

4）启动 envoy

5）监控并管理 envoy 的运行状况，比如 envoy 出错时 pilot-agent 负责重启 envoy，或者 envoy 配置变更后 reload envoy

3、Citadel

负责处理系统上不同服务之间的 TLS 通信。 Citadel 充当证书颁发机构(CA)，并生成证书以允许在数据平面中进行安全的 mTLS 通信。

Citadel 通过内置的身份和证书管理，可以支持强大的服务到服务以及最终用户的身份验证。

可以使用 Citadel 来升级服务网格中的未加密流量。

4、Galley

Galley 是 istio 的配置验证、提取、处理和分发组件。Galley 提供配置管理的服务。实现原理是通过 k8s 提供的 ValidatingWebhook 对配置进行验证。比如我们在部署 istio 的时候，会部署如下的 ValidatingWebhook ：

```
apiVersion: admissionregistration.k8s.io/v1beta1
kind: ValidatingWebhookConfiguration
metadata:
  name: istiod-istio-system
  labels:
    app: istiod
    release: istio
    istio: istiod
webhooks:
  - name: validation.istio.io
    clientConfig:
      service:
        name: istiod
        namespace: istio-system
        path: "/validate"
      caBundle: "" # patched at runtime when the webhook is ready.
    rules:
      - operations:
```

```
    - CREATE
    - UPDATE
    apiGroups:
    - config.istio.io
    - security.istio.io
    - authentication.istio.io
    - networking.istio.io
    apiVersions:
    - "*"
    resources:
    - "*"
  # Fail open until the validation webhook is ready. The webhook controller
  # will update this to `Fail` and patch in the `caBundle` when the webhook
  # endpoint is ready.
  failurePolicy: Ignore
  sideEffects: None
  admissionReviewVersions: ["v1beta1", "v1"]
```

Galley 使 Istio 可以与 Kubernetes 之外的其他环境一起工作，因为它可以将不同的配置数据转换为 Istio 可以理解的通用格式。

8.6 为什么要用 Istio?

Istio 作为目前众多 Service Mesh 中最闪耀的新星，他到底有哪些闪光点和功能？我们又为什么要选择使用它呢?

1）流量管理:

Istio 通过 Pilot 所提供的 API 动态地配置所有 Pod 中 Sidecar 的路由规则，进而控制服务间的流量和 API 调用。Istio 简化了断路器、超时和重试等服务级别属性的配置，并且可以轻松设置 A/B 测试、金丝雀部署和基于百分比的流量分割的分阶段部署等重要任务。

2）安全:

Istio 提供给开发人员应用程序级别的安全性。Istio 提供底层安全通信信道，并大规模管理服务通信的认证、授权和加密。使用 Istio ，服务通信在默认情况下是安全的，它允许跨多种协议和运行时一致地实施策略——所有这些都很少或根本不需要应用程序更改。将 Istio 与 Kubernetes 的网络策略结合使用，其优势会更大，包括在网络和应用层保护 Pod 间或服务间通信的能力。

3）可观察性:

Istio 的 Mixer 组件负责策略控制和遥测收集。通过 Istio 的监控功能，可以了解服务性能如何影响上游和下游的功能；其自定义仪表板可以提供对所有服务性能的可视化，从而了解性能如何影响其他进程。

4）平台独立: Istio 是独立于平台的，旨在运行在各种环境中，包括跨云、内部部署、Kubernetes、Mesos 等。可以在 Kubernetes 上部署 Istio 或具有 Consul 的 Nomad 上部署。

5）集成和定制: 策略执行组件可以扩展和定制，以便与现有的 ACL、日志、监控、配额、审计等方案集成。

6）istio 为可扩展性而设计，可以满足不同的部署需求。

8.7 Istio 如何与 k8s 完美结合?

8.7.1 k8s 和 istio 各自功能解读

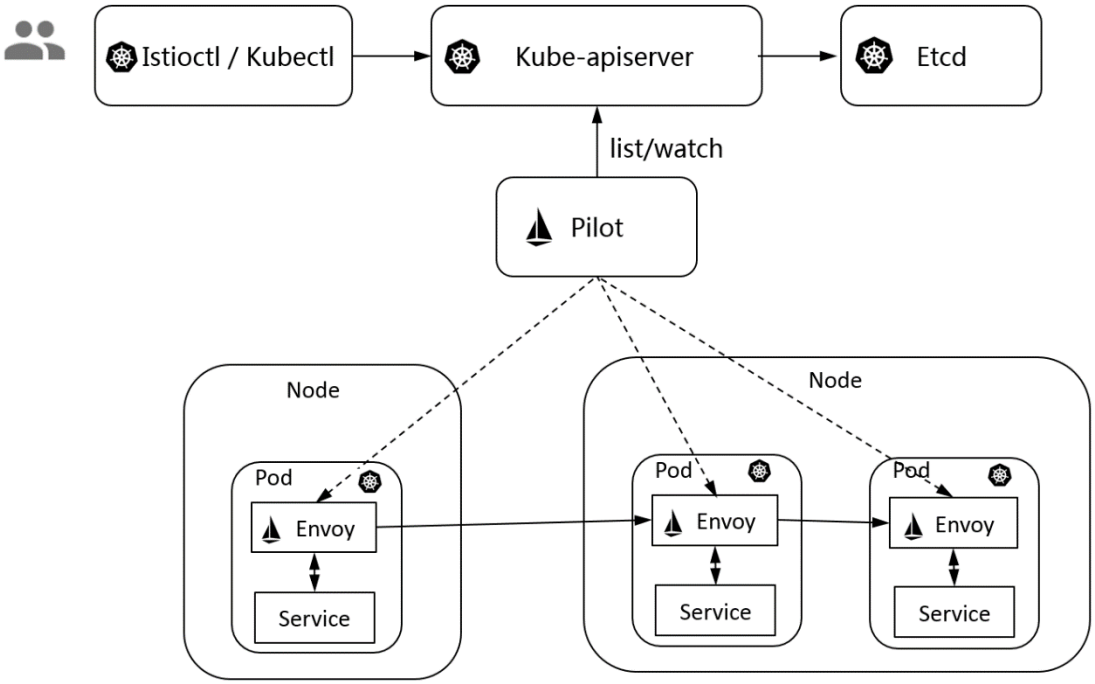
Kubernetes 提供了部署、升级、扩缩容和有限的流量管理等能力；利用 service 的机制来做服务注册和发现，通过 kube-proxy 实现转发和负载均衡。但并不具备上层如熔断、限流降级、动态路由、调用链治理等能力。

Istio 则很好的补齐了 k8s 在微服务治理上的这部分能力，同时是基于 k8s 构建的，但不是像 SpringCloud Netflix 完全重新做一套。Istio 在谷歌微服务治理上已经落地实施

8.7.2 在 Kubernetes 上叠加 Istio

Kubernetes 的 Service 基于每个节点的 Kube-proxy 从 Kube-apiserver 上获取 Service 和 Endpoint 的信息，并将对 Service 的请求经过负载均衡转发到对应的 Endpoint 上。在服务健康检查上只提供了基本的探针机制，并不提供服务访问指标和调用链追踪这种应用的服务运行诊断能力。

Istio 复用了 Kubernetes Service 的定义。Istio 的服务发现就是从 Kube-apiserver 中获取 Service 和 Endpoint，然后将其转换成 Istio 服务模型的 Service，但是其数据面组件不再是 Kube-proxy，而是在每个 Pod 里部署了 Envoy 代理，通过拦截 Pod 的 Inbound 流量和 Outbound 流量，可以做更多更细粒度的服务治理、监控和安全等能力。



对于云原生应用，采用 Kubernetes 构建微服务部署和集群管理能力，采用 Istio 构建服务治理能力，这将成为微服务的主流方案。

8.7.3 Istio 在 AI 领域的应用

AI 领域中，深度学习模型的持续优化是生产应用重要挑战，它关乎到整体效率及资源利用率。比如电商平台，需要根据用户行为记录持续训练模型，才能更好地对用户偏好和消费热点进行有效地预测。同时一个新模型上线也需要一个安全可控的流程来验证新模型带来的改进。而基于 Istio，阿里云可以轻巧实现流量控制和模型版本的更新及回滚，以及适用于深度学习模型灰度发布能力。

在 AI 领域之外，目前阿里云已经支持游戏行业、自动化办公、在线教育、互联网广告的等数家客户基于 Istio 在 Kubernetes 集群上进行微服务应用的开发。

8.8 在 k8s 平台安装 Istio

8.8.1 准备安装 Istio 是要的压缩包

官网下载地址：

https://github.com/istio/istio

官方访问相对较慢，我在课件提供了压缩包，大家最好用我的压缩包，这样做实验才不会出问题

1、把压缩包上传到 k8s 的控制节点 xuegod63。手动解压：

[root@xuegod63 ~]# tar zxvf istio-1.8.1.tar.gz

2、切换到 istio 包所在目录下。Istio-1.8.1.tar.gz 解压的软件包包名是 istio-1.8.1，则：

[root@xuegod63 ~]# cd istio-1.8.1

安装目录包含如下内容：

1）samples/目录下，有示例应用程序

2）bin/目录下，包含 istioctl 的客户端文件。istioctl 工具用于手动注入 Envoy sidecar 代理。

3、将 istioctl 客户端路径增加到 path 环境变量中，macOS 或 Linux 系统的增加方式如下：

[root@xuegod63 istio-1.8.1]# export PATH=$PWD/bin:$PATH

4、把 istioctl 这个可执行文件拷贝到/usr/bin/目录

[root@xuegod63 istio-1.8.1]# cd /root/istio-1.8.1/bin/

[root@xuegod63 bin]# cp -ar istioctl /usr/bin/

8.8.2 安装 istio

1.下载镜像：

安装 istio 需要的镜像默认从官网拉取，但是官网的镜像我们拉取会有问题，可以从课件下载镜像，然后上传到自己 k8s 集群的各个节点，通过 docker load -i 手动解压镜像：

```
docker load -i  examples-bookinfo-details.tar.gz
docker load -i  examples-bookinfo-reviews-v1.tar.gz
docker load -i  examples-bookinfo-productpage.tar.gz
docker load -i  examples-bookinfo-reviews-v2.tar.gz
docker load -i  examples-bookinfo-ratings.tar.gz
docker load -i  examples-bookinfo-reviews-v3.tar.gz
docker load -i  pilot.tar.gz
docker load -i  proxyv2.tar.gz
docker load -i  httpbin.tar.gz
```

2.安装

在 k8s 的控制节点 xuegod63 操作

[root@xuegod63 ~]# istioctl install --set profile=demo -y

看到如下，说明 istio 初始化完成：

```
Detected that your cluster does not support third party JWT authentication. Falling
back to less secure first party JWT. See https://istio.io/docs/ops/best-
practices/security/#configure-third-party-service-account-tokens for details.
- Applying manifest for component Base...
✔ Finished applying manifest for component Base.
- Applying manifest for component Pilot...
✔ Finished applying manifest for component Pilot.
  Waiting for resources to become ready...
  Waiting for resources to become ready...
- Applying manifest for component EgressGateways...
- Applying manifest for component IngressGateways...
```

- Applying manifest for component AddonComponents...
✔ Finished applying manifest for component EgressGateways.
✔ Finished applying manifest for component IngressGateways.
✔ Finished applying manifest for component AddonComponents.

✔ Installation complete

3.验证 istio 是否部署成功
[root@xuegod63 ~]# kubectl get pods -n istio-system
显示如下，说明部署成功
istio-egressgateway-d84f95b69-5gtdc 1/1 Running 0 15h
istio-ingressgateway-75f6d79f48-fhxjj 1/1 Running 0 15h
istiod-c9f6864c4-nrm82 1/1 Running 0 15h

4.卸载 istio 集群，暂时不执行，记住这个命令即可
[root@xuegod63 ~]# istioctl manifest generate --set profile=demo | kubectl delete -f –

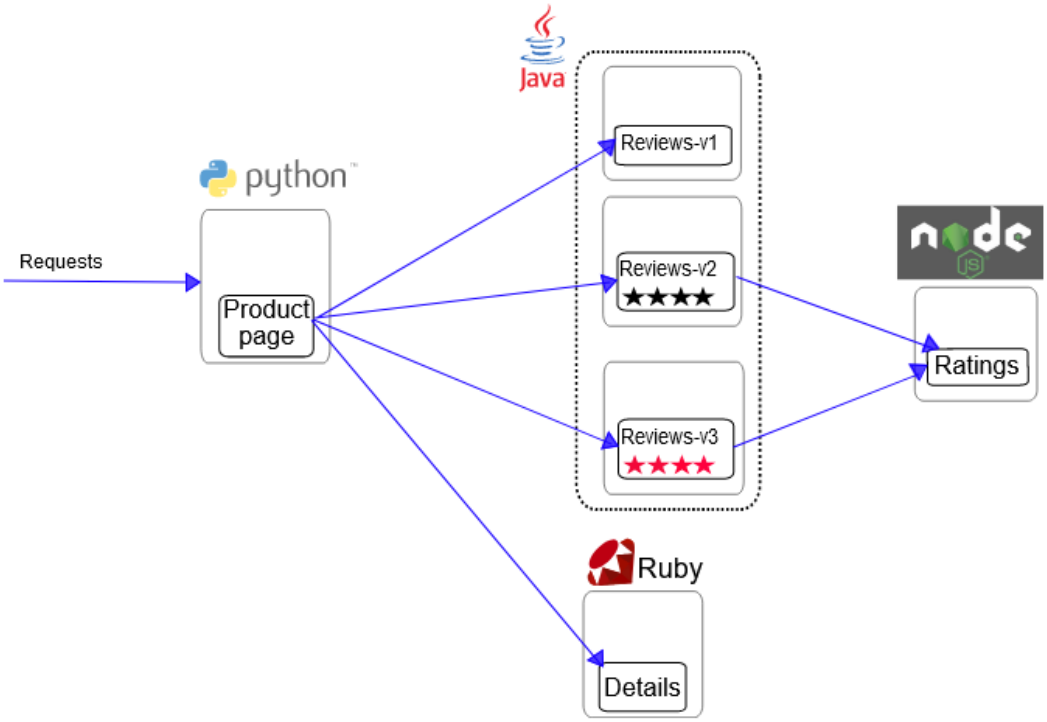
实战 1：通过 Istio 部署在线书店 bookinfo

1.在线书店功能介绍
在线书店-bookinfo
该应用由四个单独的微服务构成，这个应用模仿在线书店的一个分类，显示一本书的信息，页面上会显示一本书的描述，书籍的细节（ISBN、页数等），以及关于这本书的一些评论。

Bookinfo 应用分为四个单独的微服务
1）productpage 这个微服务会调用 details 和 reviews 两个微服务，用来生成页面；
2）details 这个微服务中包含了书籍的信息；
3）reviews 这个微服务中包含了书籍相关的评论，它还会调用 ratings 微服务；
4）ratings 这个微服务中包含了由书籍评价组成的评级信息。

reviews 微服务有 3 个版本
1）v1 版本不会调用 ratings 服务；
2）v2 版本会调用 ratings 服务，并使用 1 到 5 个黑色星形图标来显示评分信息；
3）v3 版本会调用 ratings 服务，并使用 1 到 5 个红色星形图标来显示评分信息。

下图展示了这个应用的端到端架构

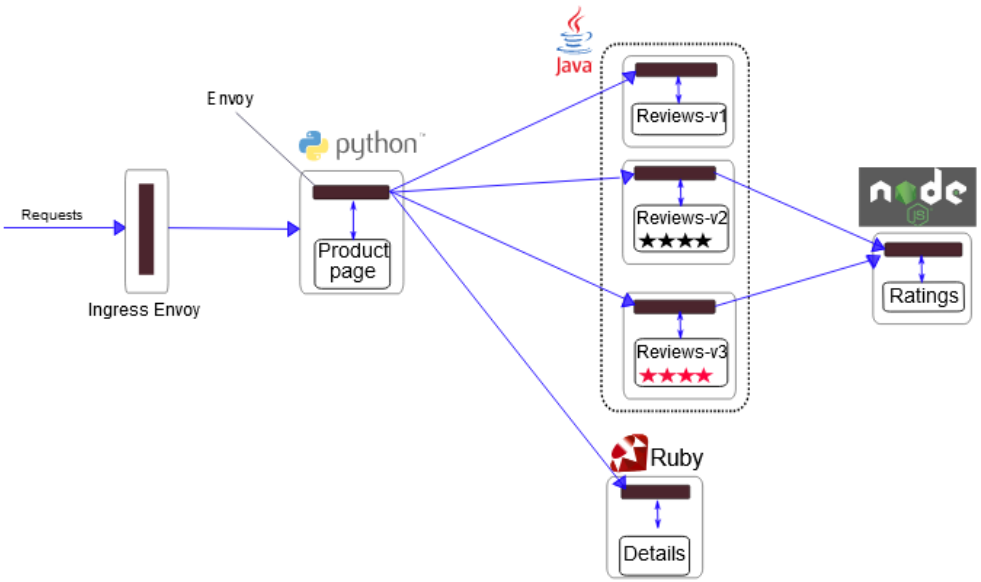


Bookinfo Application without Istio

Bookinfo 应用中的几个微服务是由不同的语言编写的。这些服务对 istio 并无依赖，但是构成了一个有代表性的服务网格的例子：它由多个服务、多个语言构成，并且 reviews 服务具有多个版本。

2.部署应用

要在 Istio 中运行这一应用，无需对应用自身做出任何改变。 只要简单的在 Istio 环境中对服务进行配置和运行，具体一点说就是把 Envoy sidecar 注入到每个服务之中。 最终的部署结果将如下图所示：



Bookinfo Application

所有的微服务都和 Envoy sidecar 集成在一起，被集成服务所有的出入流量都被 envoy sidecar 所劫持，这样就为外部控制准备了所需的 Hook，然后就可以利用 Istio 控制平面为应用提供服务路由、遥测数据收集以及策略实施等功能。

3.启动应用服务

1.进入 istio 安装目录。

2.istio 默认自动注入 sidecar，需要为 default 命名空间打上标签 istio-injection=enabled

[root@xuegod63 ~]# kubectl label namespace default istio-injection=enabled

3.使用 kubectl 部署应用

[root@xuegod63 ~]# cd istio-1.8.1

[root@xuegod63 istio-1.8.1]# kubectl apply -f samples/bookinfo/platform/kube/bookinfo.yaml

上面的命令会启动全部的四个服务，其中也包括了 reviews 服务的三个版本（v1、v2 以及 v3）。

4.确认所有的服务和 Pod 都已经正确的定义和启动：

[root@xuegod63 istio-1.8.1]# kubectl get services

显示如下

```
NAME                       TYPE          CLUSTER-IP        EXTERNAL-IP     PORT(S)
details                    ClusterIP     10.109.124.202    <none>          9080/TCP
productpage                ClusterIP     10.102.89.129     <none>          9080/TCP
ratings                    ClusterIP     10.101.97.75      <none>          9080/TCP
reviews                    ClusterIP     10.100.105.33     <none>          9080/TCP
```

[root@xuegod63 istio-1.8.1]# kubectl get pods

显示如下

```
NAME                                READY    STATUS     RESTARTS    AGE
details-v1-78d78fbddf-qssjb         2/2      Running    0           73m
productpage-v1-85b9bf9cd7-r699f     2/2      Running    0           73m
ratings-v1-6c9dbf6b45-77kv7         2/2      Running    0           73m
reviews-v1-564b97f875-2jtxq         2/2      Running    0           73m
reviews-v2-568c7c9d8f-f5css         2/2      Running    0           73m
reviews-v3-67b4988599-fxfzx         2/2      Running    0           73m
tomcat-deploy-59664bcb6f-5z4nn      1/1      Running    0           22h
tomcat-deploy-59664bcb6f-cgjbn      1/1      Running    0           22h
tomcat-deploy-59664bcb6f-n4tqq      1/1      Running    0           22h
```

5.确认 Bookinfo 应用是否正在运行，在某个 Pod 中用 curl 命令对应用发送请求，例如 ratings：

[root@xuegod63 istio-1.8.1]# kubectl exec -it $(kubectl get pod -l app=ratings -o jsonpath='{.items[0].metadata.name}') -c ratings -- curl productpage:9080/productpage | grep -o "<title>.*</title>"

显示如下：

```
<title>Simple Bookstore App</title>
```

6.确定 Ingress 的 IP 和端口

现在 Bookinfo 服务已经启动并运行，你需要使应用程序可以从 Kubernetes 集群外部访问，例如从浏览器访问，那可以用 Istio Gateway 来实现这个目标。

1）为应用程序定义 Ingress 网关：

[root@xuegod63 istio-1.8.1]# kubectl apply -f samples/bookinfo/networking/bookinfo-gateway.yaml

2）确认网关创建完成：

[root@xuegod63 istio-1.8.1]# kubectl get gateway

显示如下：

NAME AGE

bookinfo-gateway 2m18s

3）确定 ingress ip 和端口

执行如下指令，明确自身 Kubernetes 集群环境支持外部负载均衡：

[root@xuegod63 istio-1.8.1]# kubectl get svc istio-ingressgateway -n istio-system

```
[root@k8s-master istio-1.5.1]# kubectl get svc istio-ingressgateway -n istio-system
NAME                   TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)                                                                                                   AG
E
istio-ingressgateway   LoadBalancer   10.100.8.62    <pending>     15020:31144/TCP,80:30925/TCP,443:30282/TCP,15029:31571/TCP,15030:32362/TCP,15031:30052/TCP,15032:32110/TCP,31400:30367/TCP,15443:32221/TCP   2d
6h
```

如果 EXTERNAL-IP 值已设置，说明环境正在使用外部负载均衡，可以用其为 ingress gateway 提供服务。 如果 EXTERNAL-IP 值为<none>（或持续显示<pending>）， 说明环境没有提供外部负载均衡，无法使用 ingress gateway。在这种情况下，你可以使用服务的 NodePort 访问网关。

若自身环境未使用外部负载均衡器，需要通过 node port 访问。执行如下命令。设置 ingress 端口：

export INGRESS_PORT=$(kubectl -n istio-system get service istio-ingressgateway -o jsonpath='{.spec.ports[?(@.name=="http2")].nodePort}')

export SECURE_INGRESS_PORT=$(kubectl -n istio-system get service istio-ingressgateway -o jsonpath='{.spec.ports[?(@.name=="https")].nodePort}')

4）设置 GATEWAY_URL

[root@xuegod63 istio-1.8.1]# INGRESS_HOST=192.168.1.63

#192.168.1.63 是安装 istio 的机器，即 k8s 控制节点 xuegod63 的 ip

[root@xuegod63 istio-1.8.1]# export GATEWAY_URL=$INGRESS_HOST:$INGRESS_PORT

[root@xuegod63 istio-1.8.1]# echo $GATEWAY_URL 显示如下：

192.168.1.63:32109

确认可以从集群外部访问应用

可以用 curl 命令来确认是否能够从集群外部访问 Bookinfo 应用程序：

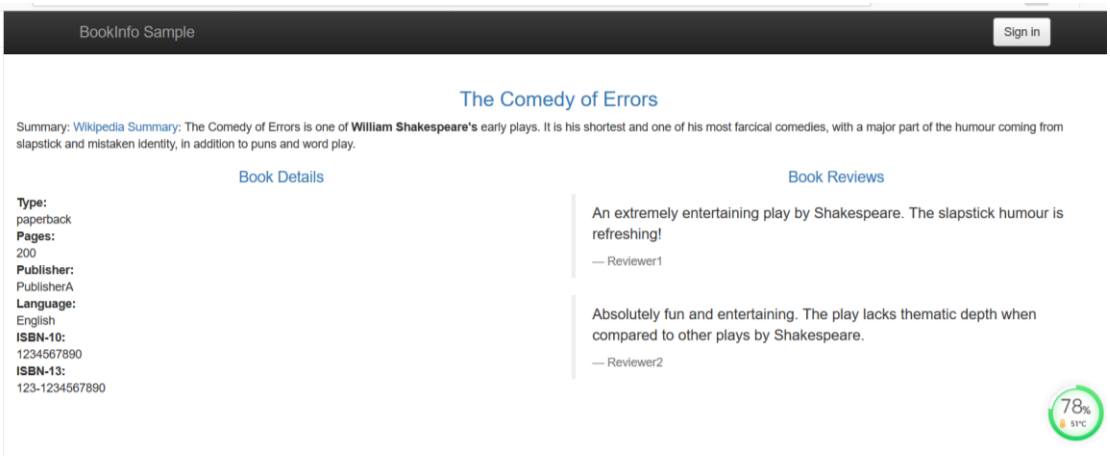
[root@xuegod63 istio-1.8.1]# curl -s http://${GATEWAY_URL}/productpage | grep -o "<title>.*</title>"

显示如下：

```
[root@k8s-master istio-1.5.1]# curl -s http://${GATEWAY_URL}/productpage | grep -o "<title>.
*</title>"
<title>Simple Bookstore App</title>
```

还可以用浏览器打开网址 http://$GATEWAY_URL/productpage，也就是 192.168.1.63:32109/productpage 来浏览应用的 Web 页面。如果刷新几次应用的页面，就会看到

productpage 页面中会随机展示 reviews 服务的不同版本的效果（红色、黑色的星形或者没有显示）。



4.卸载 bookinfo 服务

可以使用下面的命令来完成应用的删除和清理了：

1.删除路由规则，并销毁应用的 Pod

[root@xuegod63 istio-1.8.1]# sh samples/bookinfo/platform/kube/cleanup.sh

2.确认应用已经关停

kubectl get virtualservices #-- there should be no virtual services

kubectl get destinationrules #-- there should be no destination rules

kubectl get gateway #-- there should be no gateway

kubectl get pods #-- the Bookinfo pods should be deleted

总结

8.1 带你认识微服务

8.2 微服务架构发展进程

8.3 微服务框架对比分析

8.4 使用微服务需要解决的问题

8.5 Istio 概念、架构、组件详细介绍

8.6 为什么要用 Istio?

8.7 Istio 如何与 k8s 完美结合?

8.8 在 k8s 平台安装 Istio

实战 1：通过 Istio 部署在线书店