在k8s部署istio

```
$ mkdir istio
$ cd istio
$ wget https://github.com/istio/istio/releases/download/1.4.2/istio-1.4.2-linux.tar.gz
$ tar zxvf istio-1.4.2-linux.tar.gz
$ cd istio-1.4.2
$ mv bin/istioctl /usr/bin
#查看istio当前支持安装模式
$ istioctl profile list
Istio configuration profiles:
    default                                                    #默认安装
    demo                                                       #完全安装
    minimal                                                    #最小安装
    remote
    sds
$ istioctl manifest apply --set profile=demo
$ kubectl get pods -n istio-system
$ kubectl get svc -n istio-system
```

卸载：

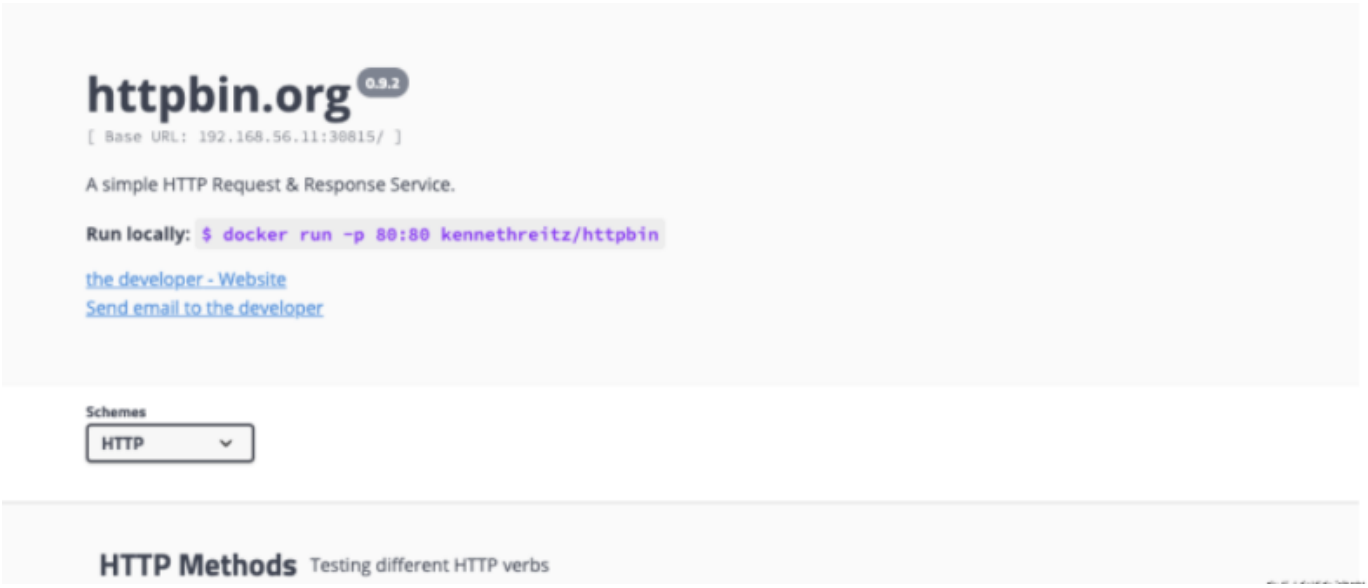
```
istioctl manifest generate --set profile=demo | kubectl delete -f -
```

部署httpbin web案例

安装httpbin

```
$ cd istio-1.4.2/samples/httpbin
$ kubectl apply -f httpbin-nodeport.yaml    #nodeport服务，默认是在default ns
$ kubectl get pod,svc
```

访问测试：http://10.4.7.110:30815/

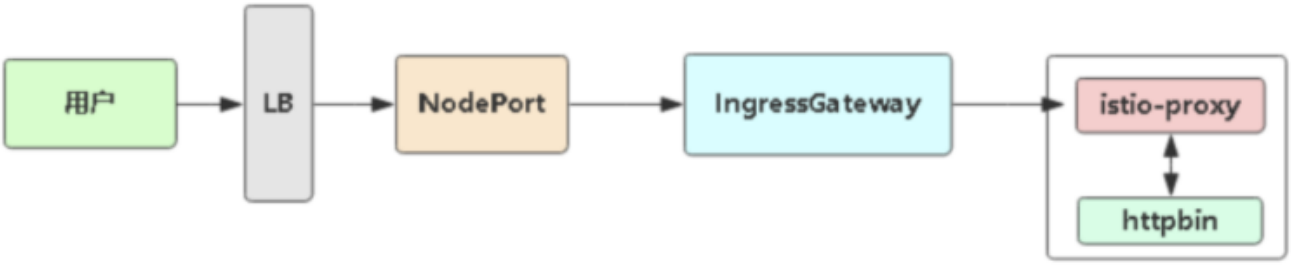


手动将httpbin服务注册到sidecar

```
$ kubectl apply -f <(istioctl kube-inject -f httpbin-nodeport.yaml)
# 或者
$ istioctl kube-inject -f httpbin-nodeport.yaml |kubectl apply -f -
$ kubectl get pod
NAME                                 READY   STATUS            RESTARTS   AGE
httpbin-77bfc6b755-k8pjb             1/1     Running           0          8m3s
httpbin-79bf7bbcd4-mjmjc             0/2     PodInitializing   0          11s
```

自动注入

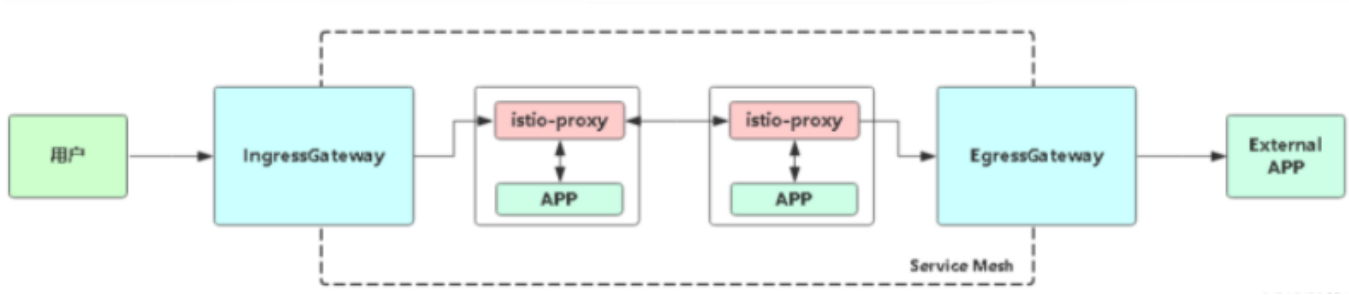
```
$ kubectl label namespace default istio-injection=enabled
$ kubectl apply -f httpbin-gateway.yaml
$ kubectl get gateway
NAME              AGE
httpbin-gateway   3m21s
```



```
$ kubectl get svc -n istio-system
istio-ingressgateway     LoadBalancer   10.0.0.114   <pending>
15020:32361/TCP,80:31929/TCP,443:31088/TCP,15029:31493/TCP,15030:32677/TCP,15031:30048/
TCP,15032:32207/TCP,15443:30034/TCP   75m
```

访问K8SNODEIP:31929 即可访问，此线路是走的SideCar的gateway

服务网关： gateway

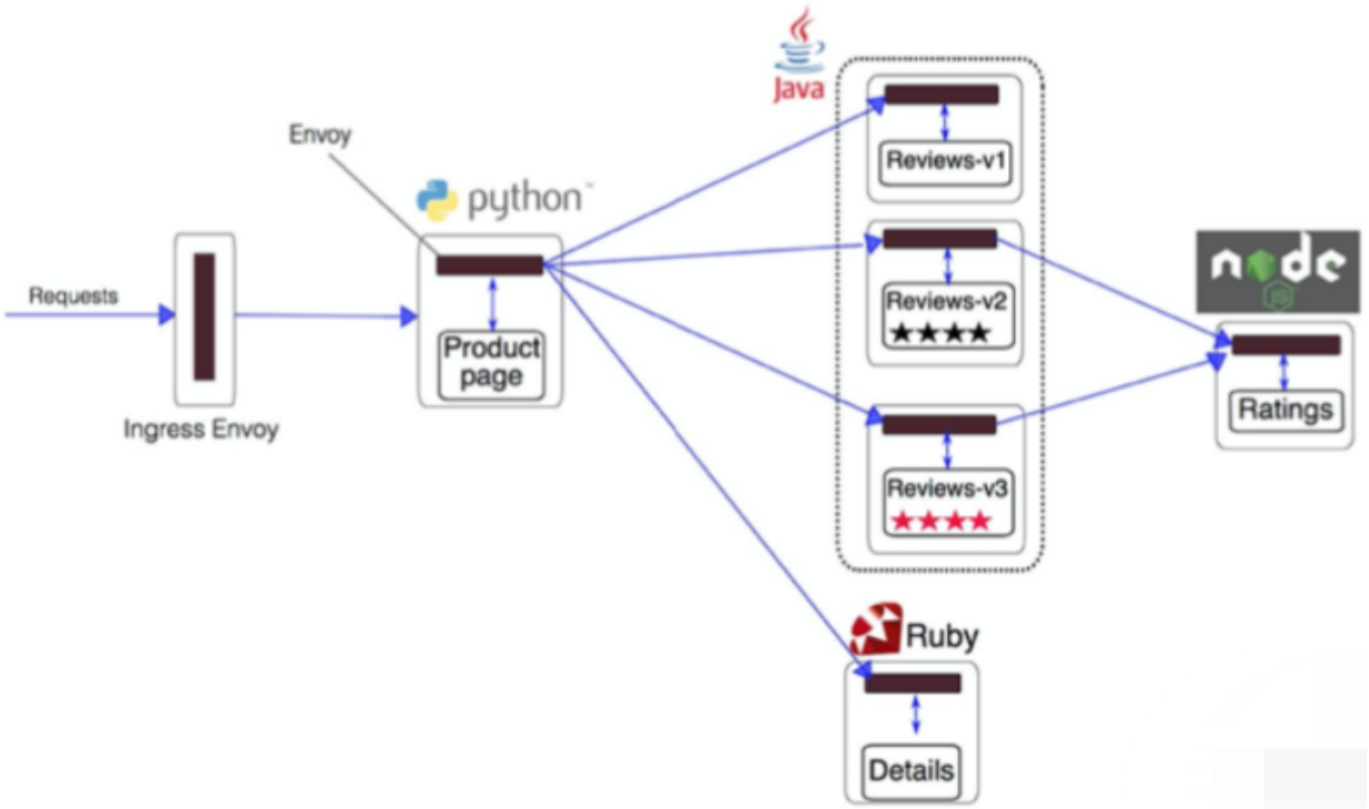


- Gateway为网格内服务提供负载均衡器，提供以下功能: •
  - L4-L7的负载均衡
  - 对外的mTLS
- Gateway根据流入流出方向分为:
  - IngressGateway:接收外部访问，并将流量转发到网格内的服务。
  - EgressGateway:网格内服务访问外部应用

部署Bookinfo案例

介绍

- Bookinfo 应用分为四个单独的微服务:
  - productpage :productpage 微服务会调用 details 和 reviews 两个微服务，用来生成页面。
  - details :这个微服务包含了书籍的信息。
  - reviews :这个微服务包含了书籍相关的评论。它还会调用 ratings 微服务。
  - ratings :ratings 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:
  - v1 版本不会调用 ratings 服务。
  - v2 版本会调用 ratings 服务，并使用 5个黑色五角星来显示评分信息。
  - v3 版本会调用 ratings 服务，并使用5个红色五角星 来显示评分信息。



部署bookinfo实例

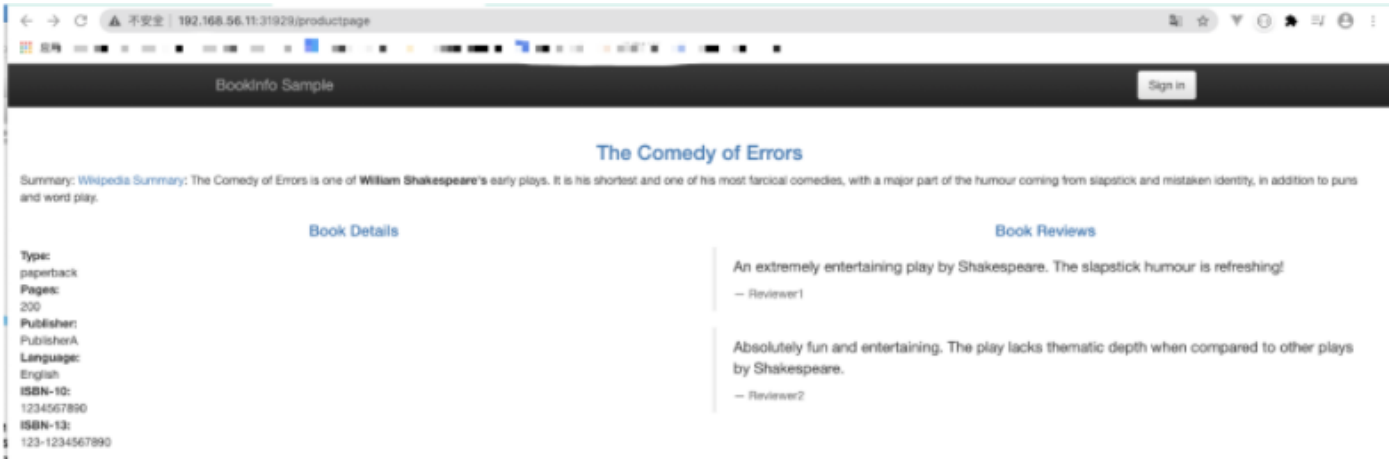
```
kubectl create ns bookinfo
kubectl label namespace bookinfo istio-injection=enabled   #sidecar自动注入开启
cd istio-1.4.2/samples/bookinfo/
kubectl apply -f platform/kube/bookinfo.yaml -n bookinfo
kubectl apply -f networking/bookinfo-gateway.yaml -n bookinfo
kubectl get svc -n istio-system| grep ingress  #查到80对应31929的端口暴露
```

访问http://192.168.56.11:31929/productpage

使用LB+NGINX实现域名反代

- 准备： 找k8s其中的一台节点，安装nginx（192.168.56.12）

```
$ yum install epel-release -y
$ yum install nginx
```

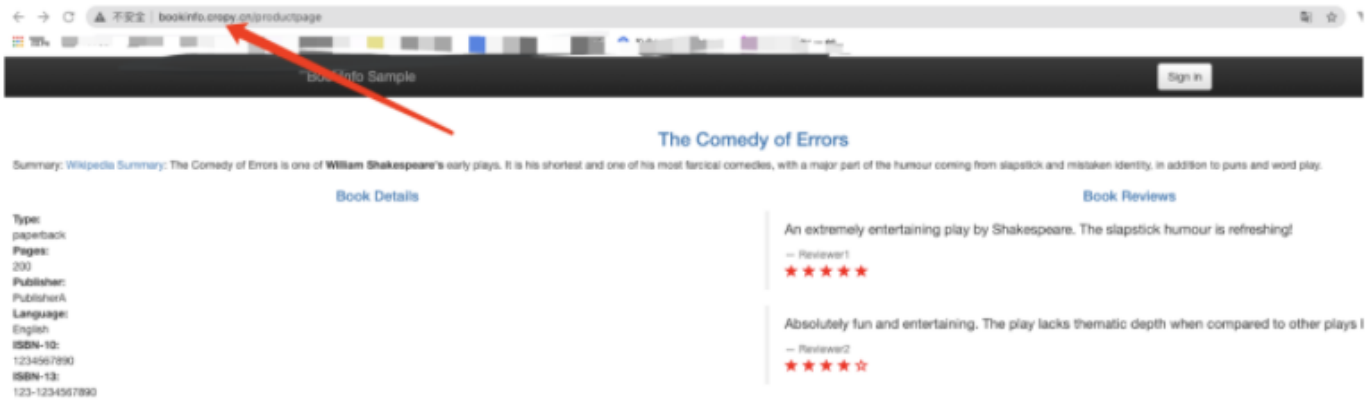


- 配置nginx LB

```
$ vim /etc/nginx/nginx.conf
   # 增加这里
   upstream  ingressgateway {
           server     192.168.56.11:31929;
           server     192.168.56.12:31929;
           server     192.168.56.13:31929;

    }
    server {
        listen       80 default_server;
        server_name  _;
        root         /usr/share/nginx/html;

        # Load configuration files for the default server block.
        include /etc/nginx/default.d/*.conf;

        location / {
            proxy_pass    http://ingressgateway;  # 增加这里
            proxy_set_header  Host $host;
            proxy_http_version  1.1;
        }
        .....

$ nginx -t
$ nginx
```

- 配置域名解析测试

```
$ vim /etc/hosts
192.168.56.12 bookinfo.jettjia.com
```



- bookinfo gateway 配置(其实也就是要配置hosts与LB保持一致)

```
$ cd ~/istio-1.4.2/samples/bookinfo
$ vim networking/bookinfo-gateway.yaml
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: bookinfo-gateway
spec:
  selector:
    istio: ingressgateway # use istio default controller
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
    hosts:
    - "bookinfo.jettjia.com" # 修改这里
---
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: bookinfo
spec:
  hosts:
  - "bookinfo.jettjia.com" # 修改这里
  gateways:
  。。。。

$ kubectl apply -f networking/bookinfo-gateway.yaml -n bookinfo
```

浏览器访问：bookinfo.jettjia.com

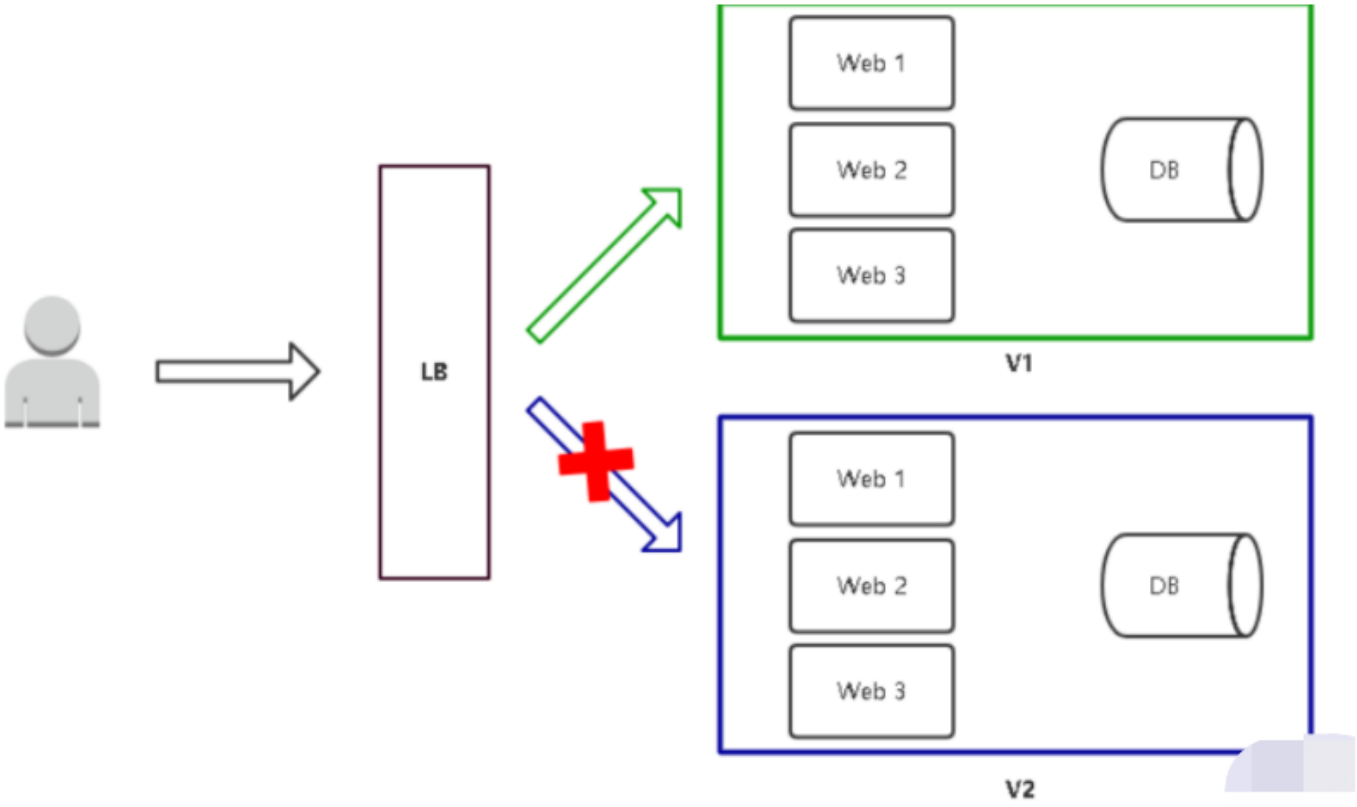
Istio 实现灰度发布

- 主流发布方案
  - 蓝绿发布
  - 滚动发布
  - 灰度发布
  - A/B Test

蓝绿发布

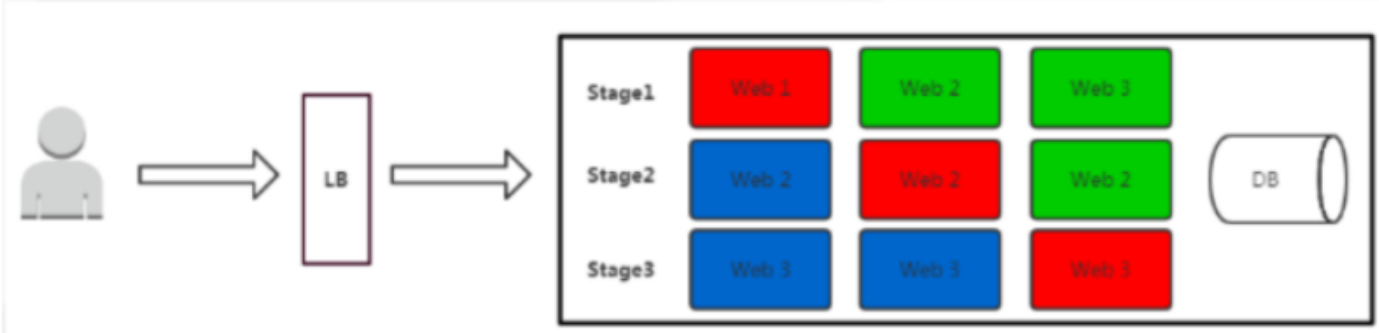
项目逻辑上分为AB组，在项目升级时，首先把A组从负载均衡中摘除，进行新版本的部署。B组仍然继续提供服务。A组升级完成上线，B组从负载均衡中摘除。

- 特点:
  - 策略简单
  - 升级/回滚速度快
  - 用户无感知，平滑过渡
- 缺点:
  - 需要两倍以上服务器资源
  - 短时间内浪费一定资源成本
  - 有问题影响范围大

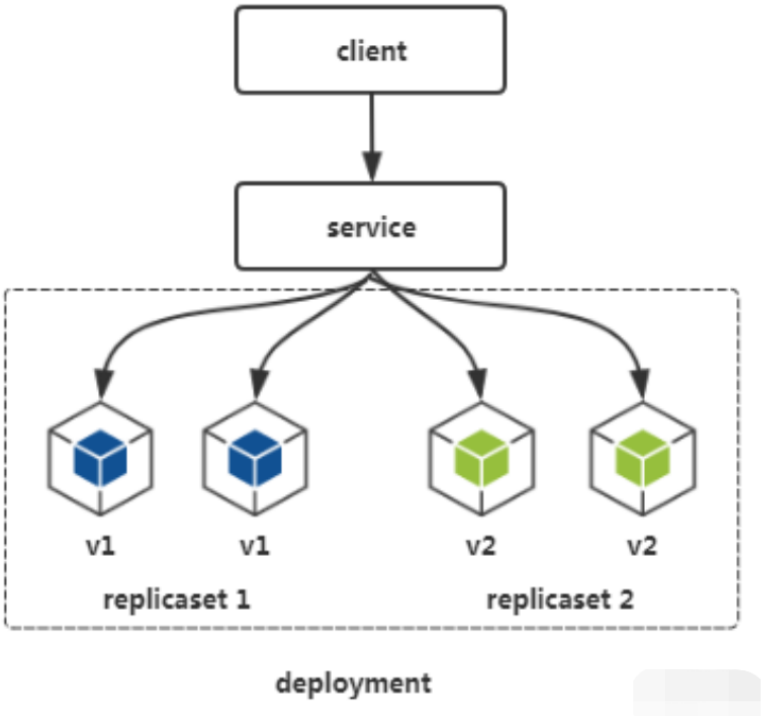


滚动发布

每次只升级一个或多个服务，升级完成后加入生产环境，不断执行这个过程，直到集群中的全部旧版升级新版本。Kubernetes的默认发布策略。



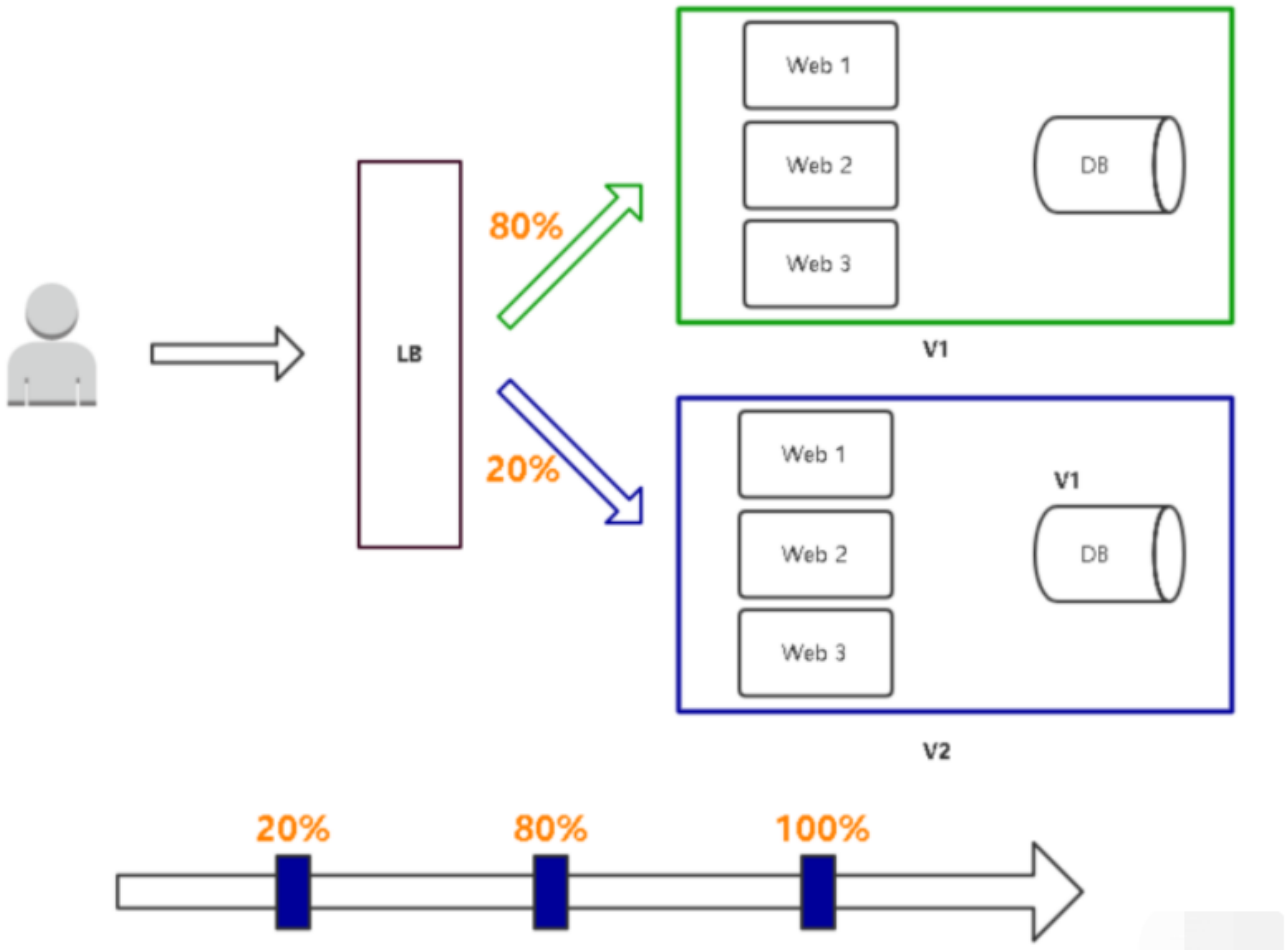
- 特点:
  - 用户无感知，平滑过渡
- 缺点:
  - 部署周期长
  - 发布策略复杂
  - 不易回滚
  - 有影响范围较大



灰度发布(金丝雀发布)

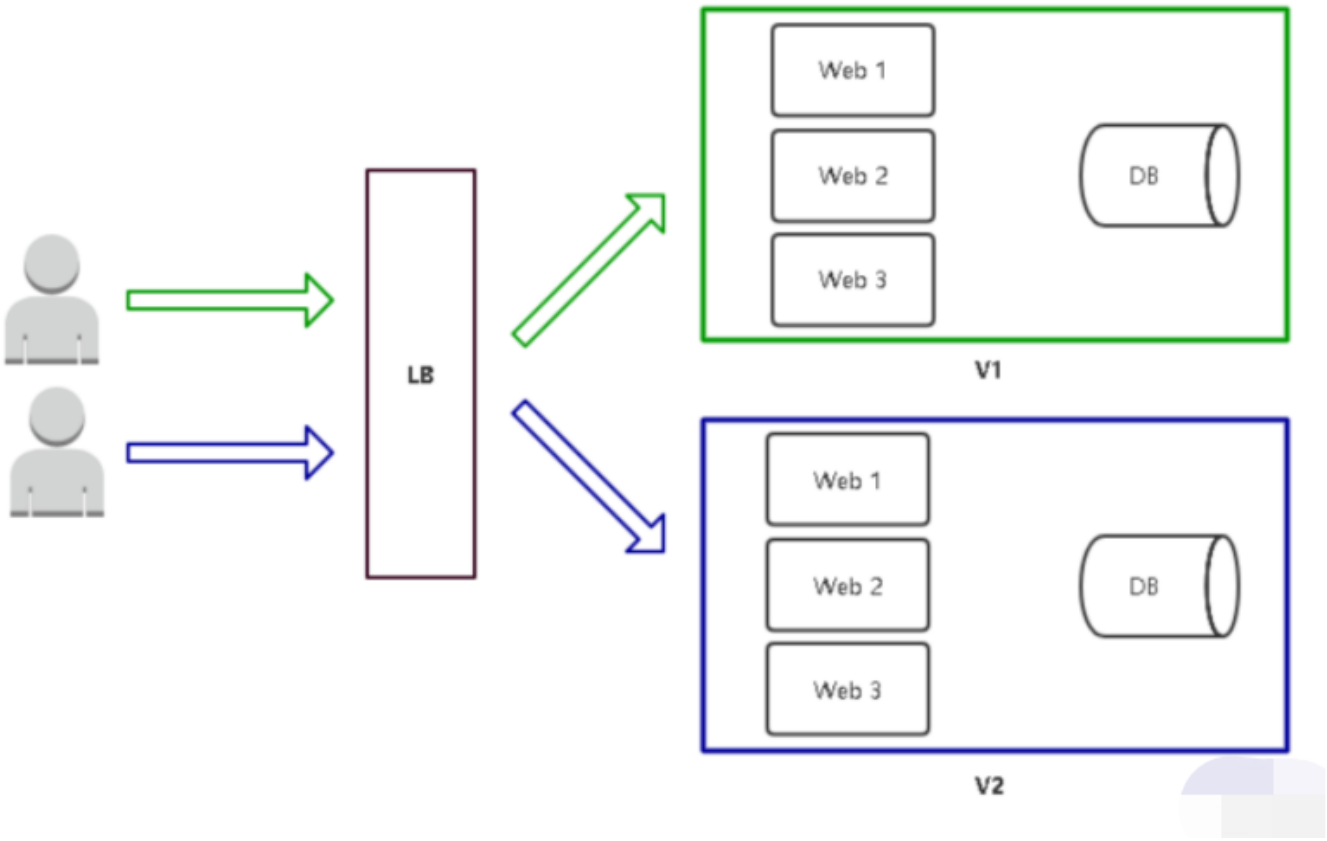
只升级部分服务，即让一部分用户继续用老版本，一 部分用户开始用新版本，如果用户对新版本没有什么 意见，那么逐步扩大范围，把所有用户都迁移到新版 本上面来。

- 特点:
  - 保证系统整体稳定性
  - 用户无感知，平滑过渡
- 缺点:
  - 自动化要求高



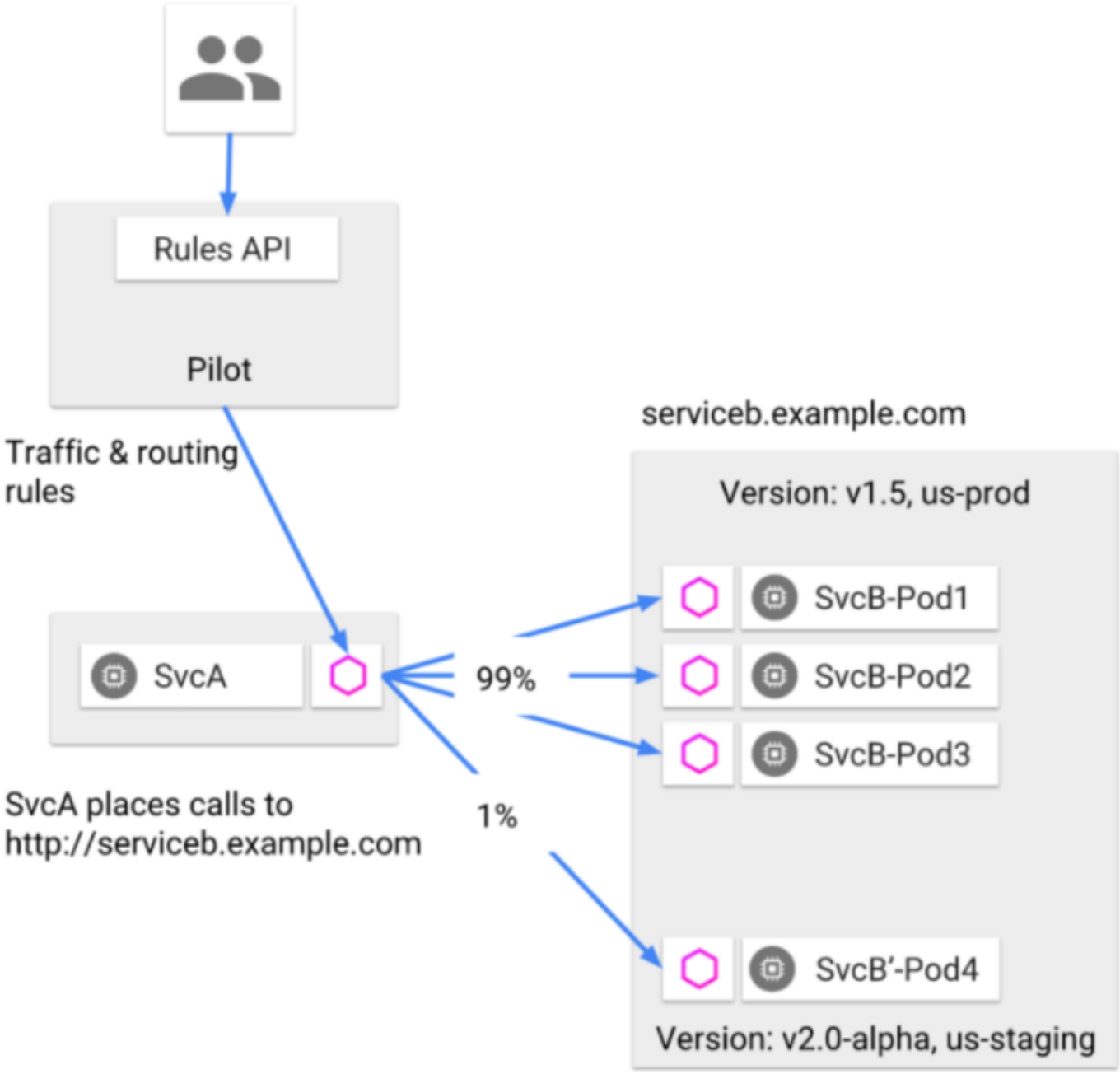
A/B Test

灰度发布的一种方式，主要对特定用户采样后，对收 集到的反馈数据做相关对比，然后根据比对结果作出 决策。用来测试应用功能表现的方法, 侧重应用的可用性，受欢迎程度等，最后决定是否升级。



基于权重的路由(金丝雀发布)

1. 流量全部发送到reviews v1版本(不带五角星)
2. 将90%的流量发送到reviews v1版本，另外10%的流量发送到reviews v2版本(5个黑色五星)，最后完全切换到v2版本
3. 将50%的流量发送到v2版本，另外50%的流量发送到v3版本(5个红色五角星)



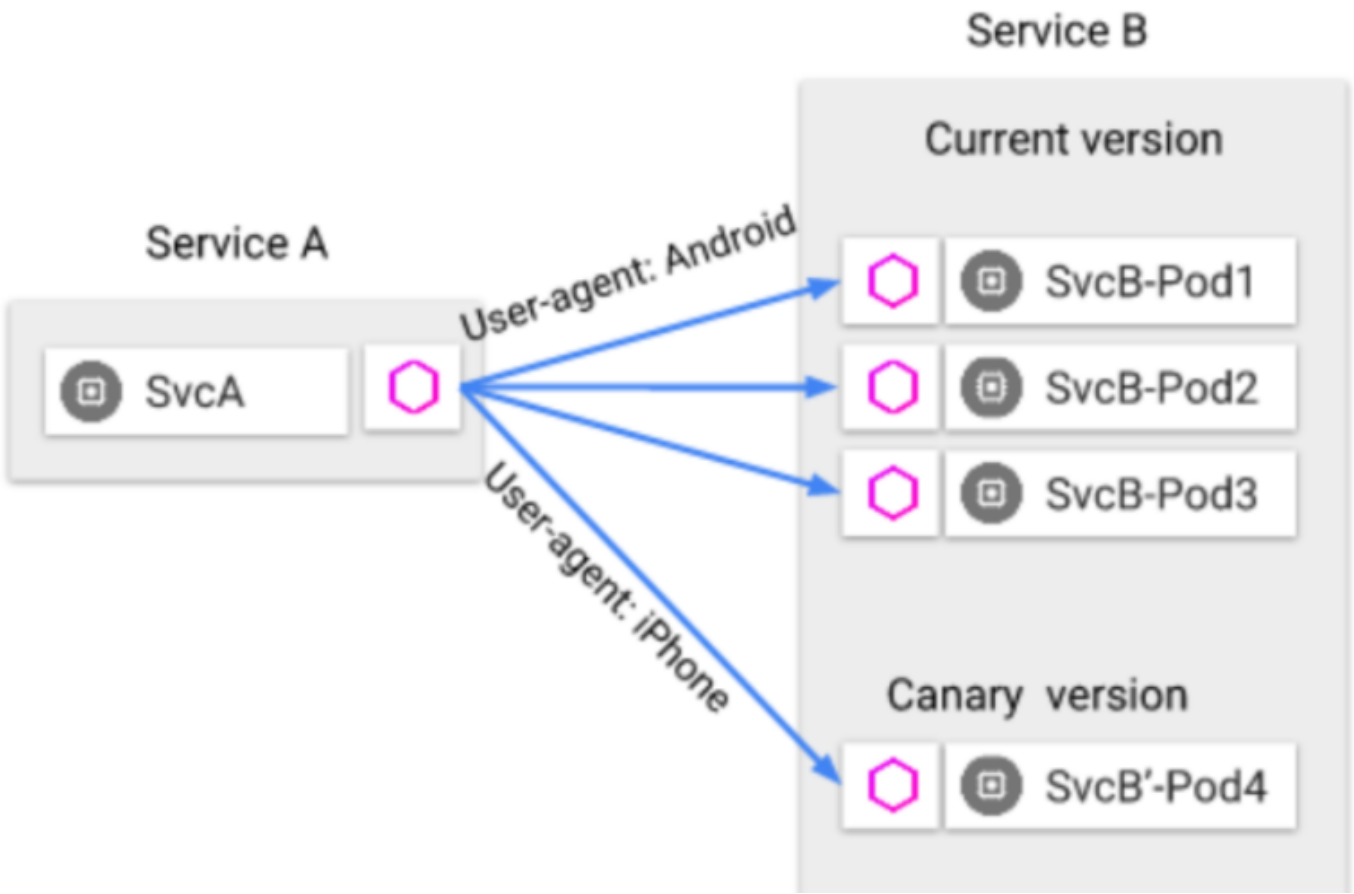

```
$ cd ~/istio-1.4.2/samples/bookinfo
# 发布的内容，流量全部在v1上， 浏览器访问测试；发现页面都 不带五角星
$ kubectl apply -f networking/virtual-service-all-v1.yaml -n bookinfo
$ kubectl apply -f networking/destination-rule-all.yaml -n bookinfo

# 发布的内容，会有 10%到v2版本，90%到v1版本
$ kubectl apply -f networking/virtual-service-reviews-90-10.yaml -n bookinfo


$ kubectl apply -f networking/virtual-service-reviews-v2-v3.yaml -n bookinfo
```

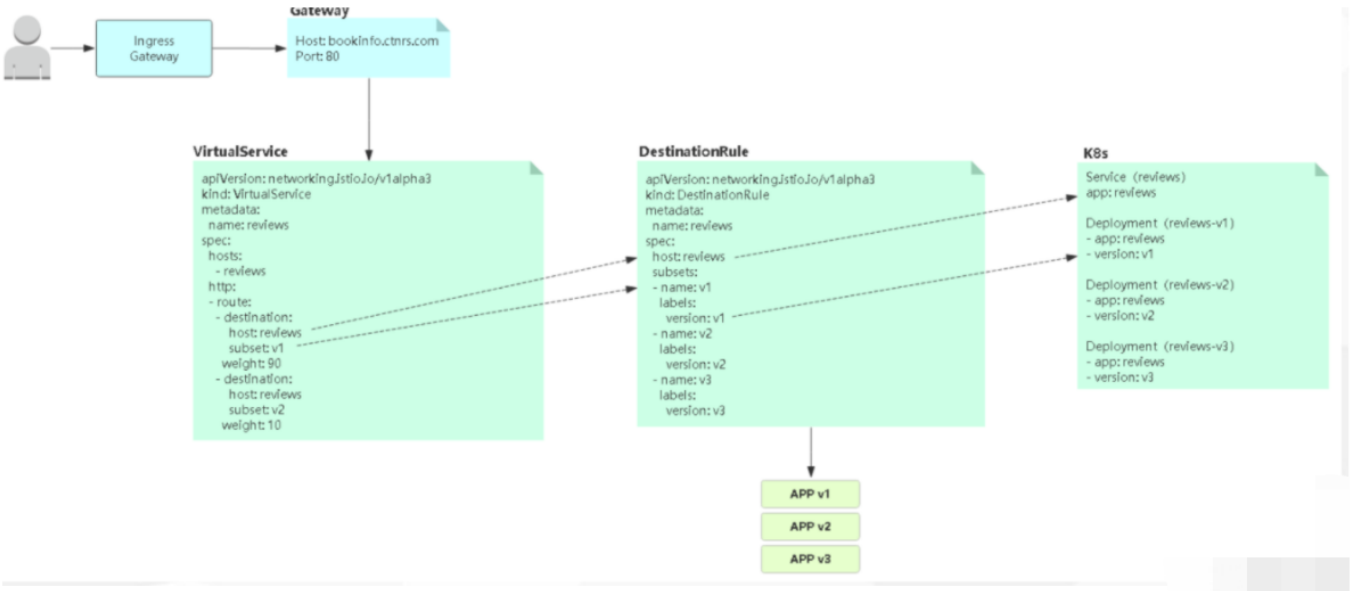
基于请求内容的路由(A/B Test)

任务:

1. 将特定用户的请求发送到reviews v2版本(5个黑色五 角星)，其他用户则不受影响(v3)

```
$ kubectl apply -f networking/virtual-service-reviews-jason-v2-v3.yaml
```





微服务可视化监控

安装监控

监控指标（ Grafana）

网格可视化（Kiai）

调用链跟踪（ Jaeger）

```
[root@k8s-m1 bookinfo]# cat monitor-gateway.yaml
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: grafana-gateway
  namespace: istio-system
spec:
  selector:
    istio: ingressgateway
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
    hosts:
    - "*"
---
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: grafana
spec:
  hosts:
  - "grafana.jettjia.com"
  gateways:
  - grafana-gateway
  http:
  - route:
    - destination:
        host: grafana
        port:
          number: 3000
---
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: kiali-gateway
```

```
  namespace: istio-system
spec:
  selector:
    istio: ingressgateway
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
    hosts:
    - "*"
---
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: kiali
spec:
  hosts:
  - "kiali.jettjia.com"
  gateways:
  - kiali-gateway
  http:
  - route:
    - destination:
        host: kiali
        port:
          number: 20001
---
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: tracing-gateway
  namespace: istio-system
spec:
  selector:
    istio: ingressgateway
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
    hosts:
    - "*"
---
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: tracing
spec:
```

```
  hosts:
  - "tracing.jettjia.com"
  gateways:
  - tracing-gateway
  http:
  - route:
    - destination:
        host: tracing
        port:
          number: 80
```

创建

```
$ kubectl apply -f monitor-gateway.yaml -n istio-system
```

检验

0. 本地host配置

```
10.4.7.110 grafana.jettjia.com kiali.jettjia.com  tracing.jettjia.com
```

1） 浏览器访问：grafana.jettjia.com

模拟产生数据

```
$ for i in {1...100}; do curl -I http://10.4.7.110/productpage -H
"Host:bookinfo.jettjia.com";sleep 1; done
```

2） kiali.jettjia.com/kiali/

admin/admin

3. tracing.jettjia.com

介绍可视化监控

grafana

1、请求错误率

2、请求时延（响应时间）

kiali

3、链路调用拓扑图

4、RPS（每秒请求），也有请求错误率

5、请求/响应数据包大小

6、查看Pod日志

7、istio配置资源在线编辑

Jeager

8、一个服务涉及的调用情况

9、分析数据包中具体请求/响应信息

18、也有响应时间

主要针对流量获取

详细的规划,人力的支持

性能

service 四层

user 访问NodePort节点 <NodeIP：PORT> 作为流量的入口

-->ipvs 做分发

--> 后端的endpoint PodIP:Port

占用端口

Ingress 七层

user 访问ingress-pod所在的节点 该节点的 IP:80 作为流量入口

-->ingress 内部基于域名的分发

-->分发后端的服务

某个节点作为流量入口，采用nginx进行负载均衡

Istio

user 访问位于每个节点的istio映射端口 NodeIP:Port 该节点做了路由分发。 真正做到负载均衡

-->istio-ingressgateway(每个节点都有监听端口，接受流量)

-->virtualservice（基于域名的分发。有点像ingress。将流量进行分发） 七层的负载均衡

-->应用服务资源端口