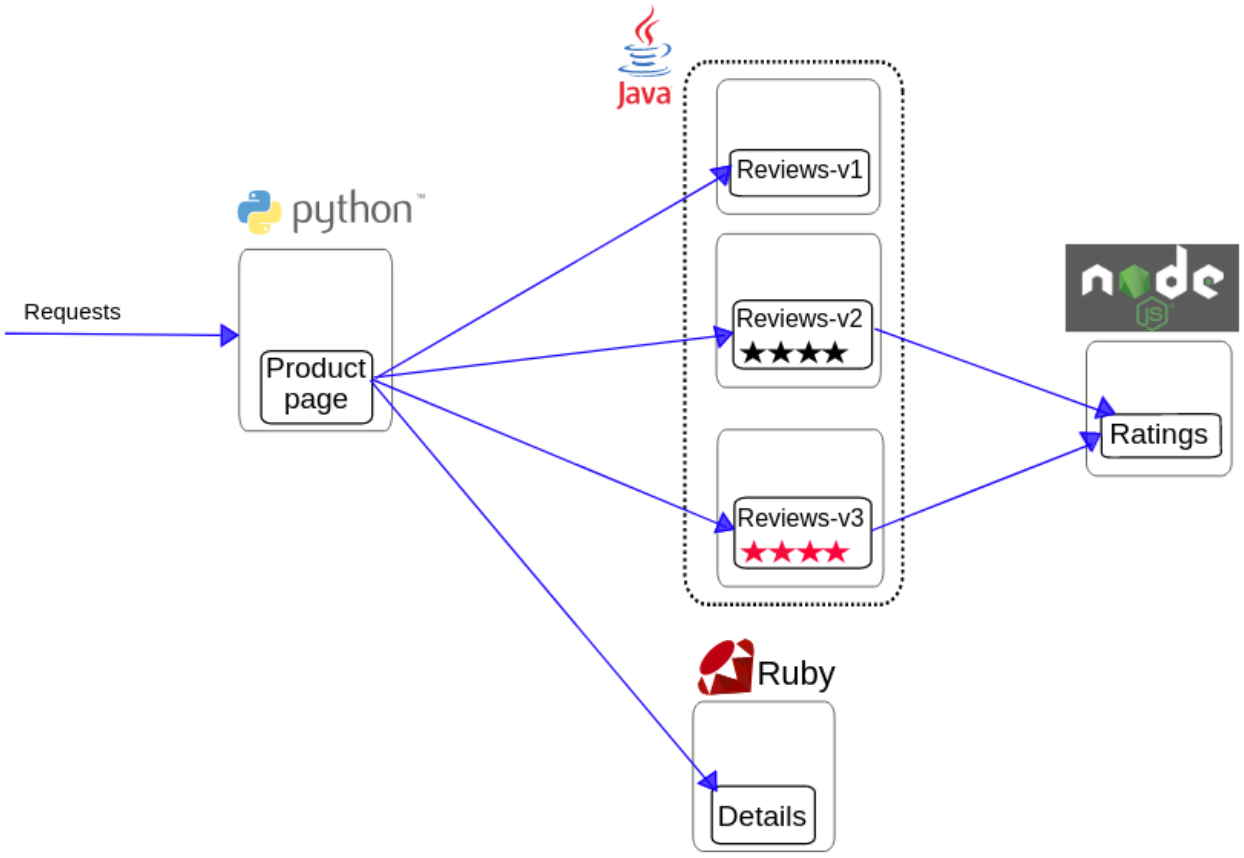
istio 官方示例

bookinfo



配置sidecar注入

```
istioctl kube-inject --debug -f samples/bookinfo/kube/bookinfo.yaml > bookinfo2.yaml
```

使用kubernetes创建示例

```
kubernetes create -f bookinfo2.yaml
```

配置gateway

```
export GATEWAY_URL=$(kubectl get po -l istio=ingress -n istio-system -o 'jsonpath={.items[0].status.hostIP}'):$(kubectl get svc istio-ingress -n istio-system -o 'jsonpath={.spec.ports[0].nodePort}')
```

访问

```
http://$GATEWAY_URL/productpage
```

便可以看到example的界面了。

测试：动态路由请求

bookinfo示例中有三个Reviews v1,v2,v3服务，没有配置默认的使用版本，Productpage的服务请求会被istio随机的到3个版本的服务上。
1、配置所有的微服务使用v1版本
以下yaml中destination代表查找目的service的name，route指定了只要pod有label标签version=v1被访问。
route-rule-all-v1.yaml

```
apiVersion: config.istio.io/v1alpha2
kind: RouteRule
metadata:
  name: productpage-default
spec:
  ## Used by services inside the Kubernetes cluster
  destination:
    name: productpage
  precedence: 1
  route:
  - labels:
      version: v1
---
apiVersion: config.istio.io/v1alpha2
kind: RouteRule
metadata:
  name: reviews-default
spec:
  destination:
    name: reviews
  precedence: 1
  route:
```

```
  - labels:
      version: v1
---
apiVersion: config.istio.io/v1alpha2
kind: RouteRule
metadata:
  name: ratings-default
spec:
  destination:
    name: ratings
  precedence: 1
  route:
  - labels:
      version: v1
---
apiVersion: config.istio.io/v1alpha2
kind: RouteRule
metadata:
  name: details-default
spec:
  destination:
    name: details
  precedence: 1
  route:
  - labels:
      version: v1
```

创建路由规则

```
istioctl create -f samples/bookinfo/kube/route-rule-all-v1.yaml
```

istio流量管理

istio有三种流量管理的规则类型：Route Rules, Destination Policies, Egress Rules。

Route Rules

route rules可以配置request路由到service的不同版本，可以基于source和destination，HTTP header字段做路由，可以配置单个service 版本的流量权重和优先级。

route rules的destination

1、配置规则的destination

```
destination:
  name: reviews
```

name要配置成

```
FQDN = name + "." + namespace + "." + domain
```

例如

```
destination:
  name: reviews
  namespace: default
  domain: svc.cluster.local
```

2、配置规则的source，headers

1）配置服务的caller（source），调用的服务为reviews。source的name与destination一样，都要满足

```
FQDN = name + "." + namespace + "." + domain
```

```
apiVersion: config.istio.io/v1alpha2
kind: RouteRule
metadata:
  name: reviews-to-ratings
spec:
  destination:
    name: ratings
  match:
    source:
      name: reviews
  ...
```

2）还可以限定source 服务的版本，例如限定reviews的版本为v2

```
apiVersion: config.istio.io/v1alpha2
kind: RouteRule
metadata:
  name: reviews-v2-to-ratings
spec:
  destination:
    name: ratings
  match:
    source:
      name: reviews
      labels:
        version: v2
  ...
```

3）配置HTTP headers，例如要求incoming的流量cookie头包含字串“user=jason”

```
apiVersion: config.istio.io/v1alpha2
kind: RouteRule
metadata:
  name: ratings-jason
spec:
  destination:
    name: reviews
  match:
    request:
      headers:
        cookie:
          regex: "^(.*?;)?(user=jason)(;.*)?$"
  ...
```

4）也可以同时配置source和http header

```
apiVersion: config.istio.io/v1alpha2
kind: RouteRule
metadata:
```

```
  name: ratings-reviews-jason
spec:
  destination:
    name: ratings
  match:
    source:
      name: reviews
      labels:
        version: v2
    request:
      headers:
        cookie:
          regex: "^(.*?;)?(user=jason)(;.*)?$"
  ...
```

5）配置service versions进行分流

```
apiVersion: config.istio.io/v1alpha2
kind: RouteRule
metadata:
  name: reviews-v2-rollout
spec:
  destination:
    name: reviews
  route:
  - labels:
      version: v2
    weight: 25
  - labels:
      version: v1
    weight: 75
```

6）配置http超时和重试次数，默认的http超时为15秒

```
apiVersion: config.istio.io/v1alpha2
kind: RouteRule
metadata:
```

```
  name: ratings-timeout
spec:
  destination:
    name: ratings
  route:
  - labels:
      version: v1
  httpReqTimeout:
    simpleTimeout:
      timeout: 10s
  httpReqRetries:
    simpleRetry:
      attempts: 3
```

7）故障注入

可以在request path上进行故障注入，故障可以是延迟（delays）和退出（aborts）

```
apiVersion: config.istio.io/v1alpha2
kind: RouteRule
metadata:
  name: ratings-delay-abort
spec:
  destination:
    name: ratings
  match:
    source:
      name: reviews
      labels:
        version: v2
  route:
  - labels:
      version: v1
  httpFault:
    delay:
      fixedDelay: 5s
    abort:
      percent: 10
```

```
      httpStatus: 400
```

8）配置规则的权重

例如下面的规则，如果第一条的优先级precedence小的话，则所有的流量会路由到v1版本的reviews服务。

```
apiVersion: config.istio.io/v1alpha2
kind: RouteRule
metadata:
  name: reviews-foo-bar
spec:
  destination:
    name: reviews
  precedence: 2
  match:
    request:
      headers:
        Foo: bar
  route:
  - labels:
      version: v2
---
apiVersion: config.istio.io/v1alpha2
kind: RouteRule
metadata:
  name: reviews-default
spec:
  destination:
    name: reviews
  precedence: 1
  route:
  - labels:
      version: v1
    weight: 100
```

Destination policies

Destination policies可以配置指定版本的服务的多种策略，包括负载均衡，熔断，健康检查等
配置负载均衡，支持的负载均衡有
一致性哈希（根据http_header配置）

```
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: bookinfo-ratings
spec:
  name: ratings
  trafficPolicy:
    loadBalancer:
      consistentHash:
        http_header: Cookie
```

轮询（ROUND_ROBIN），最少连接数（LEAST_CONN），随机（RANDOM），直通（PASSTHROUGH，即不应用负载均衡，具体功能不详，为Envoy代理自带的Destination load balancer）

```
apiVersion: config.istio.io/v1alpha2
kind: DestinationPolicy
metadata:
  name: ratings-lb-policy
spec:
  source:
    name: reviews
    labels:
      version: v2
  destination:
    name: ratings
    labels:
      version: v1
  loadBalancing:
    name: ROUND_ROBIN #可取值为ROUND_ROBIN、LEAST_CONN、RANDOM、PASSTHROUGH
```

配置熔断，熔断可以最大连接数，requesst limits。

```
apiVersion: config.istio.io/v1alpha2
```

```
kind: DestinationPolicy
metadata:
  name: reviews-v1-cb
spec:
  destination:
    name: reviews
    labels:
      version: v1
  circuitBreaker:
    simpleCb:
       maxConnections: 100
```

Egress Rules

Egress Rules可以定义出口到外部的服务（不归于istio sidecar管理，可以是集群内或者集群外）
只能定义http，https到外部service的规则

```
apiVersion: config.istio.io/v1alpha2
kind: EgressRule
metadata:
  name: foo-egress-rule
spec:
  destination:
    service: *.foo.com
  ports:
    - port: 80
      protocol: http
    - port: 443
      protocol: https
```

定义了Egress rules后，可以创建目的地service与Egress rules目的地一样的Route Rules和Destination policies。
使用Egress rules与Route Rules和Destination policies结合可以对外部服务配置重试，超时，故障注入支持，但是不能配置外部服务的多个版本。

Istio Ingress

Istio可以定义ingress把service暴露到service mesh集群之外，一旦Istio Ingress定义了，进入集群的流量会通过istio-ingress service，istio的monitoring和route rule可以应用到集群入口流量。
istio ingress是基于标准的kubernetes ingress，它与kubernetes ingress语法有以下不同
1）istio ingress需要包含kubernetes.io/ingress.class: istio 的annotation。
2）所有其他的的annotation将会被忽略掉。
此外istio ingress有以下限制
1）paths中的正则表达式不被支持
2）故障注入特征不被支持
下面是创建istio ingress的example

```
cat <<EOF | kubectl create -f -
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: simple-ingress
  annotations:
    kubernetes.io/ingress.class: istio
spec:
  rules:
  - http:
      paths:
      - path: /status/.*
        backend:
          serviceName: httpbin
          servicePort: 8000
      - path: /delay/.*
        backend:
          serviceName: httpbin
          servicePort: 8000
EOF
```

istio ingress支持https，还可以配合Route Rules一起使用

```
cat <<EOF | istioctl create -f -
apiVersion: config.istio.io/v1alpha2
kind: RouteRule
metadata:
```

```
  name: status-route
spec:
  destination:
    name: httpbin
  match:
    # Optionally limit this rule to istio ingress pods only
    source:
      name: istio-ingress
      labels:
        istio: ingress
    request:
      headers:
        uri:
          prefix: /delay/ #must match the path specified in ingress spec
              # if using prefix paths (/delay/.*), omit the .*.
              # if using exact match, use exact: /status
```

可以使用route rule来配置多版本路由，基于http header路由，超时，重试等特征。（注意，故障注入不支持，request path不支持正则表达式）

istio cluster访问external services

有两种方法可以配置Istio cluster内的service访问external services

1、使用egress rule（只支持http和https协议）

2、在注入sidecar时加上参数–includeIPRanges=$IPRange 来允许该pod访问外部IP