下载1.8.3的tar.gz 包

然后到bin目录下

#查看可以进行部署的形式，线上环境默认default

./istioctl profile list

#查看里面的配置yaml文件

./istioctl profile dump default

#进行部署

./istioctl install #默认default

卸载：

#istioctl manifest generate : 获取当前集群istio yaml文件清单

istioctl manifest generate | kubectl delete -f -

部署完成后看到有2个部署

[istiod] 是控制平面

[istio-ingressgateway] 是负责接收服务网格流量

然后部署

kubectl apply -f samples/httpbin/httpbin-nodeport.yaml

手动注入Sidecar

#这里会自动重启，注入Sidecar容器

kubectl apply -f <(istioctl kube-inject -f httpbin-nodeport.yaml) #自动重启

```
apiVersion: v1
kind: Service
metadata:
  name: httpbin
  labels:
    app: httpbin
    service: httpbin
spec:
  type: NodePort
  ports:
  - name: http
    port: 8000
    targetPort: 80
  selector:
    app: httpbin
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: httpbin
spec:
  replicas: 1
  selector:
    matchLabels:
      app: httpbin
      version: v1
  template:
    metadata:
      labels:
        app: httpbin
        version: v1
    spec:
      containers:
      - image: docker.io/kennethreitz/httpbin
        imagePullPolicy: IfNotPresent
```

```
        name: httpbin
        ports:
        - containerPort: 80
```

#我们进入到Sidecar容器里面可以看到两个进程，

pilont-agent： 负责下面envcy程序的热更新，如果用户配置了新的网络控制规则，会被pilont-agent感知到,将这些配置转换 Envoy配置项，传输给envcy程序完成更新。

envcy：这个是代理服务间通信的进行，流量控制规则。

自动注入（给命名空间打指定标签，启用自动注入）

kubectl label namespace default istio-injection=enabled

#如果容器在自动注入命名空间前已经启动，容器不会像手动注入Sidecar时自动重启，需要删除容器或者重新部署才能够启动Sidecar容器。如果一个命名空间有很多存在的容器，你可以挨个调试来把他们接入到istio服务网格中。 就不必大规模来进行重启了。（例如：kuick业务线）

istio-ingressgateway 访问

如果要暴露出来一个在istio服务网格中的应用的话，需要创建一个ingressgateway规则，istio一个名为的gateway的CRM资源类，与ingress资源类类似。

网络入口：

```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: httpbin-gateway
spec:
  selector:
    istio: ingressgateway #选择器指向istio-ingressgateway容器标签,代表通过istio-ingressgateway来设置网络
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
```

```
    hosts:
    - "*" # '*' 代表是所有的ip 域名都能访问进来，比如你这个服务配置了域名，你通过网页域名如：（httpbin.kaikekba.com）来进行访问，istio网格这边就通过，然后给你给你返回数据，如果你要是通过非法渠道来进行访问，他就不放行，你就访问不到。
```

虚拟主机：

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService #配置具体关联到的后端服务 ，相当于nginx中的location，相当于proxy字段后的upsterm主机
metadata:
  name: httpbin
spec:
  hosts:
  - "*" #和Gateway中 的hosts同步
  gateways:
  - httpbin-gateway #这里关联着Gateway中同名的name
  http:
  - route:
    - destination:
        host: httpbin #这里host指定的是svc名称
        port:
          number: 8000 #svc端口
```

上面两个资源类设置就会组成一组路由转发的规则

查看网页

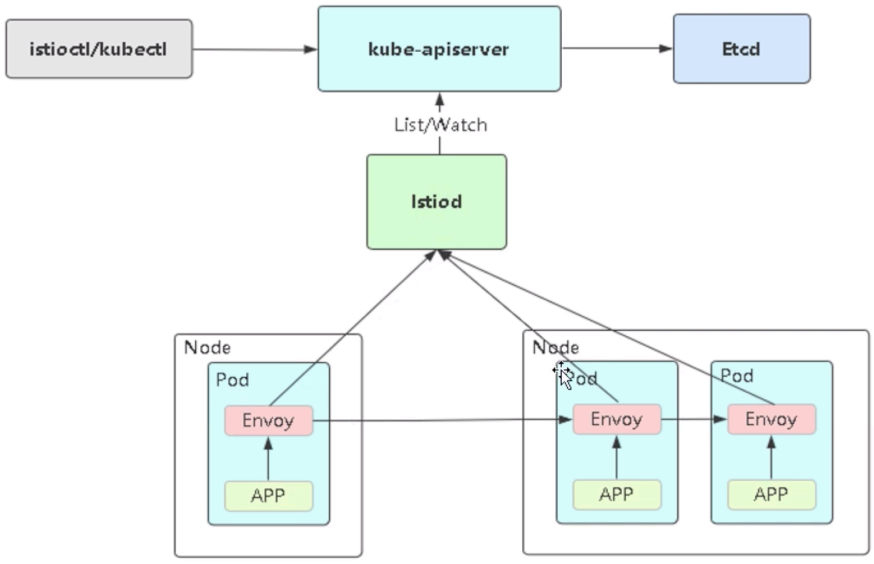
部署上面两个资源类后，我们就可以通过istio-ingressgateway 来进行访问了,本地转发istio-ingressgateway端口

```
kubectl port-forward services/istio-ingressgateway 10002:80 -n istio-system
```

#（下图为其他讲师图，需要自己重新画图）

访问流程图

#流程图（istiod会监听kube-apiserver是否有新建立的自身资源CRD，因为apiserver已经被istioctl注入了CRD资源如：Gateway等等istioCRD），然后找到新增CRD中后端对应的proxy和svc



Istio与K8s集成图