主讲老师：Fox

> 有道笔记地址：https://note.youdao.com/s/Dz279tTP

1. Nacos最佳实践

1.1 基于SPI机制实现Nacos插件开发

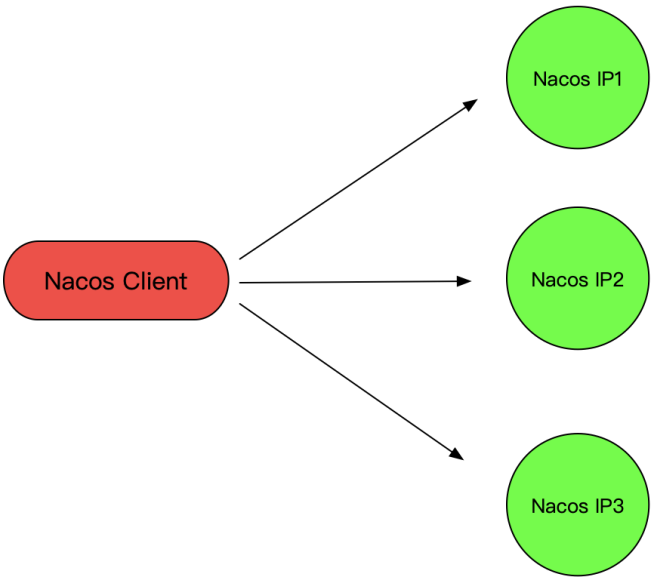
nacos插件：https://nacos.io/zh-cn/docs/v2/plugin/auth-plugin.html

1.2 Nacos 客户端接入集群最佳实践

Nacos 支持两种部署模式：单机模式和集群模式。在实践中，我们往往习惯用单机模式快速构建一个 Nacos 开发/测试环境，而在生产中，出于高可用的考虑，一定需要使用 Nacos 集群部署模式。一个多节点的 Nacos 集群启动之后，我们的应用如何很好地连接到 Nacos 集群，常用有三种方案：直连模式、 VIP 模式和地址服务器模式，并对它们进行对比。

直连模式

直连模式是部署上最简单，也是最容易理解的一种模式



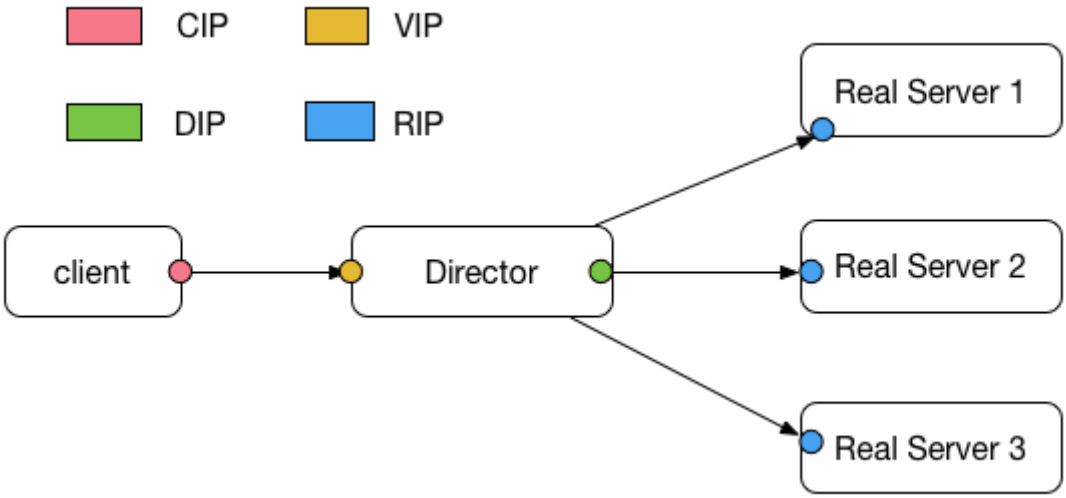
采用直连模式后，典型的开发场景配置如下：

```
Properties properties = new Properties();
properties.setProperty(PropertyKeyConst.SERVER_ADDR, "192.168.0.1:8848,192.168.0.2:8848,192.168.0.3:8848");
NamingService namingService = NacosFactory.createNamingService(properties);
```

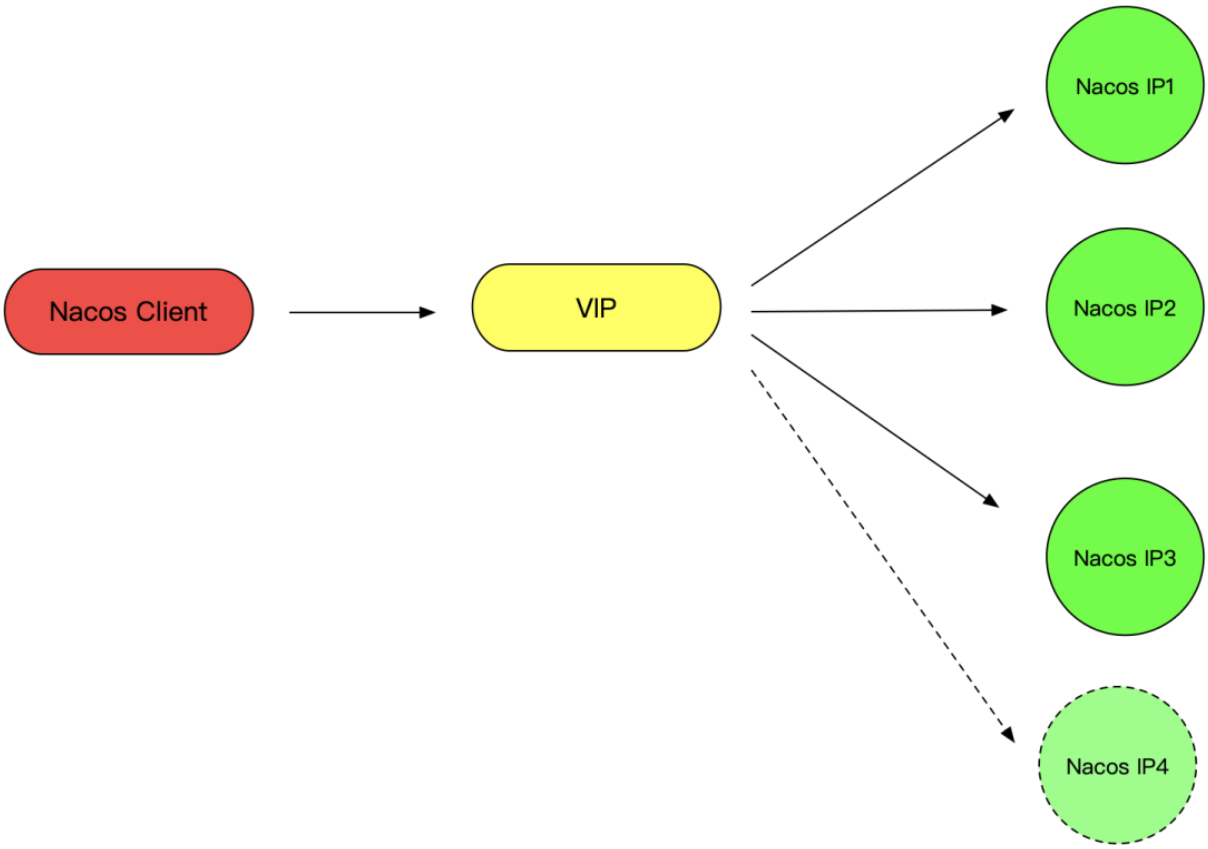
如果有一天，Nacos 的 IP 变了，例如扩缩容，机器置换，集群迁移等场景，所有的应用都需要修改，这样的方式并不灵活。所以这种模式并不是生产推荐的模式。

VIP 模式

VIP（Virtual IP） 模式可以很好的解决直连模式 IP 变化所带来的应用批量修改的问题。什么是 VIP 呢?



- RS（Real Server）： 处理实际请求的后端服务器节点。
- DS（Director Server）： 指的是负载均衡器节点，负责接收客户端请求，并转发给 RS。
- VIP（Virtual IP）： DS 用于和客户端通信的 IP 地址，作为客户端请求的目标 IP 地址。
- DIP（Directors IP）： DS 用于和内部 RS 通信的 IP 地址。
- RIP（Real IP）： 后端服务器的 IP 地址。
- CIP（Client IP）： 客户端的 IP 地址。



VIP 帮助 Nacos Client 屏蔽了后端 RIP，相对于 RIP 而言，VIP 很少会发生变化。以扩容场景为例，只需要让 VIP 感知到即可，Nacos Client 只需要关注 VIP，避免了扩容引起的代码改造。

只要是具备负载均衡能力的组件，均可以实现 VIP 模式，例如开源的 Nginx 以及阿里云负载均衡 SLB。

采用 VIP 模式后，代码不需要感知 RIP，典型的开发场景配置如下：

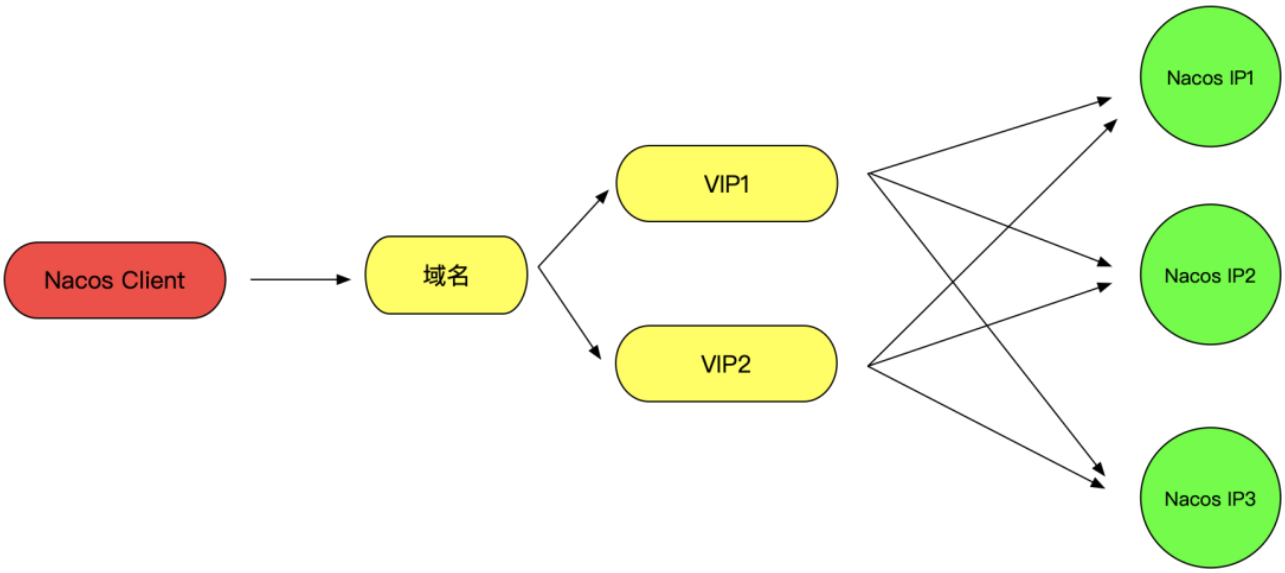
```
Properties properties = new Properties();
properties.setProperty(PropertyKeyConst.SERVER_ADDR, "{VIP}:8848");
NamingService namingService = NacosFactory.createNamingService(properties);
```

域名配置

VIP 模式和直连模式都不具备可读性，所以在实际生产中，往往还会给 VIP 挂载一个域名。

域名背后甚至可以挂载 2 个 VIP 用作高可用，路由到相同的RS；同时域名的存在也让 VIP 的置换变得更加灵活，当其中一台出现问题后，域名的 DNS 解析只会路由到另外一个正常的 VIP 上，为平滑置换预留了足够的余地。

VIP 模式的最终生产高可用版架构便产生了：



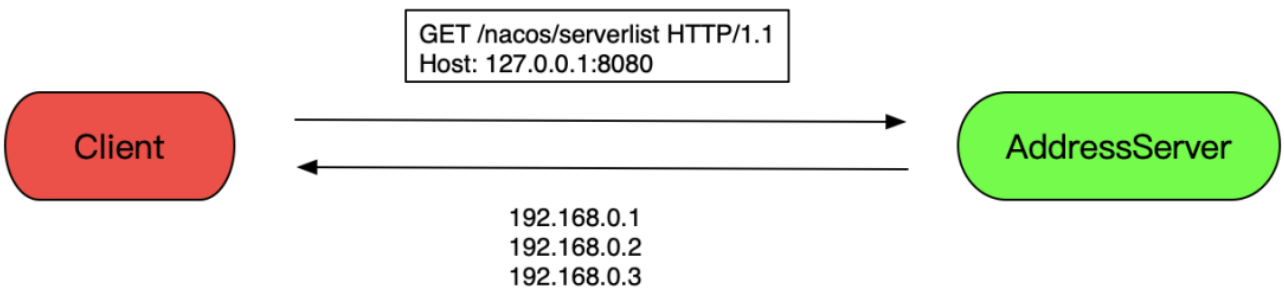
典型的开发场景配置只需要将 VIP 替换为域名即可

```
Properties properties = new Properties();
properties.setProperty(PropertyKeyConst.SERVER_ADDR, "mse-abc123qwe-nacos.mse.aliyuncs.com:8848");
NamingService namingService = NacosFactory.createNamingService(properties);
```

地址服务器模式

说起地址服务器，可能大家对这个词会感到陌生，因为地址服务器的概念主要在阿里内部比较普及，也是阿里中间件使用的最广的一种地址寻址模式。但是在开源领域，鲜有人会提及，但对于 Nacos 部署模式而言，地址服务器模式是除了 VIP 模式之外，另外一个生产可用的推荐部署方式。

地址服务器是什么？顾名思义，是用来寻址地址的服务器，发送一个请求，返回一串地址列表。尽管在阿里内部使用的真实地址服务器比这复杂一些，但下图这个简单交互逻辑，几乎涵盖了地址服务器 90% 的内容。



实现一个简易版本的地址服务器并不困难，推荐使用 nginx 搭建一个静态文件服务器管理地址， 当然你可以使用 Java!

```
@Controller
```

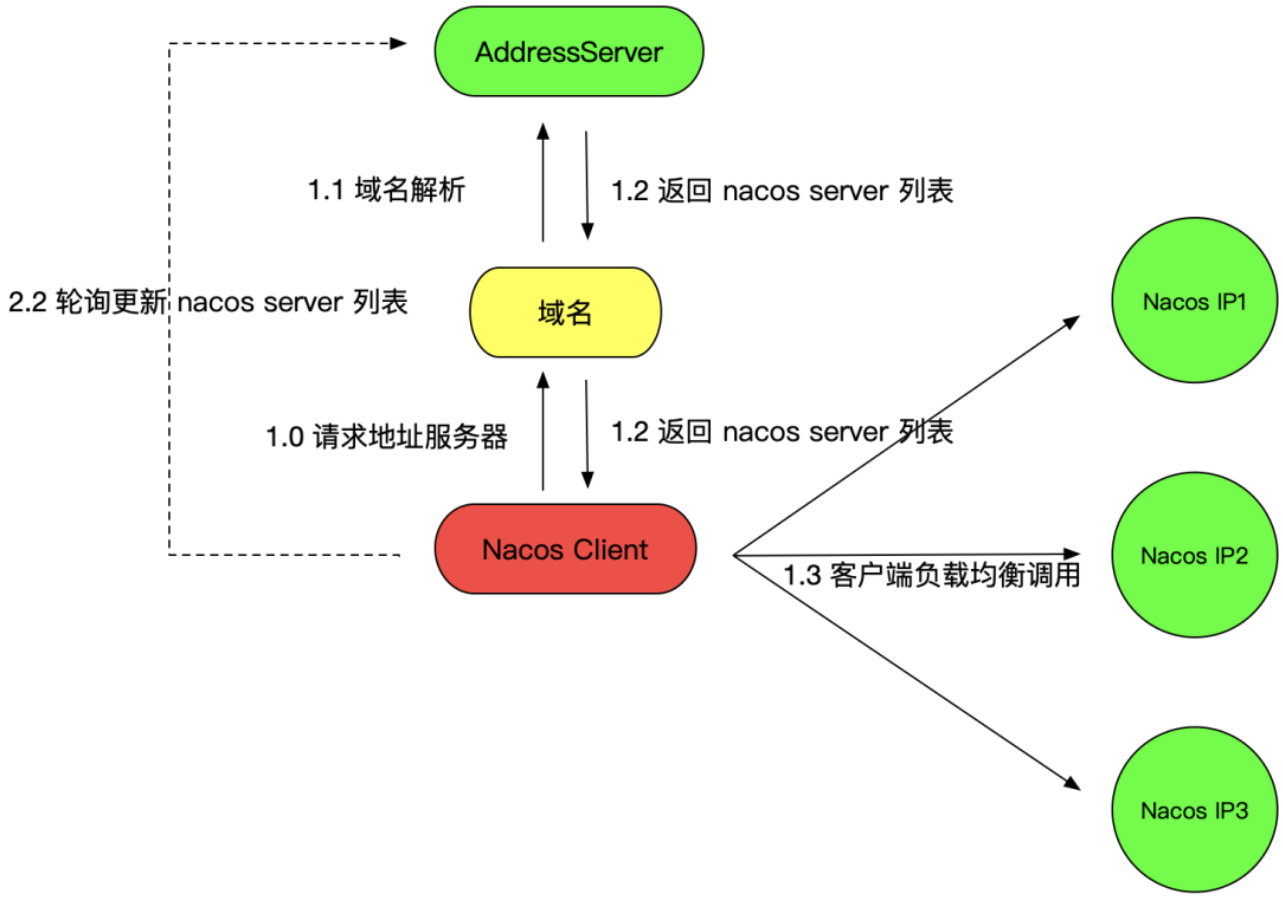
```java
public class AddressServerController {

    @RequestMapping("/nacos/serverlist")
    public ResponseEntity<String> serverlist() {
        return ResponseEntity.ok().
            header("Content-Type", "text/plain").
            body("192.168.0.1:8848\r\n" +
                    "192.168.0.2:8848\r\n" +
                    "192.168.0.3:8848\r\n"
            );
    }

}
```

使用地址服务器可以完成集群地址和客户端配置的解耦，解决直连模式中无法动态感知集群节点变化的问题。客户端根据地址服务器返回的列表，随后采取直连模式连接；并且在客户端启动后，会启动一个定时器，轮询感知 AddressServer 的变化，进而及时更新地址列表。

并且地址服务器建议配置域名，增加可读性。所以最后的部署交互架构是这样的：



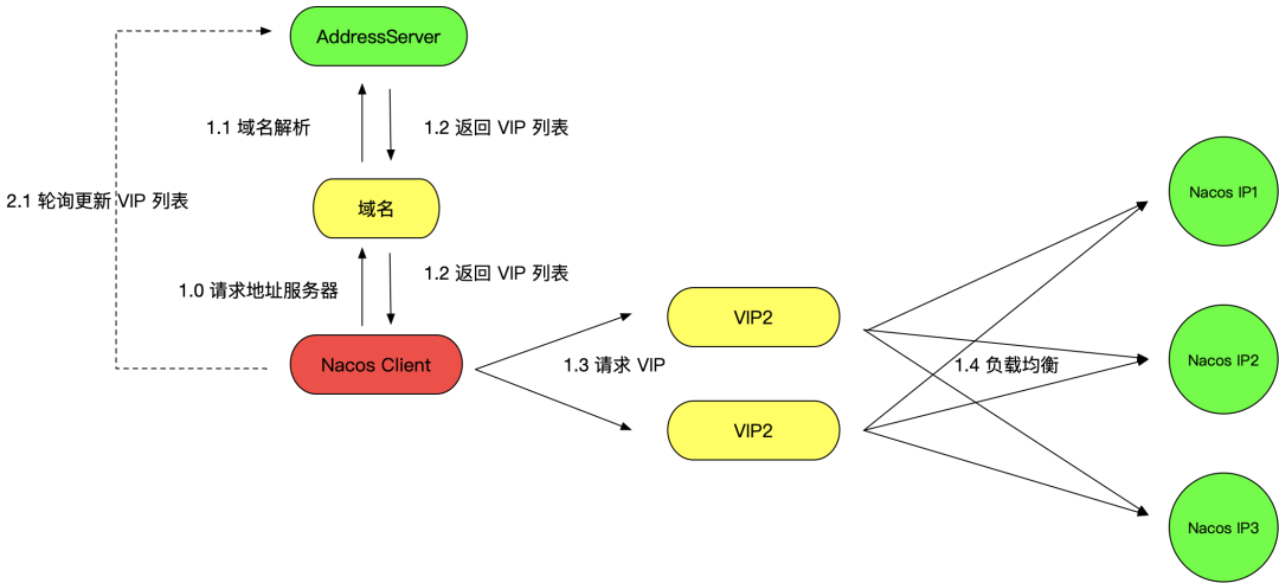
nacos-client 的源码专门适配了地址服务器模式，我们只需要配置好 addressServer 的 endpoint 即可

```java
Properties properties = new Properties();
properties.setProperty(PropertyKeyConst.ENDPOINT, "{addressServerDomain}");
properties.setProperty(PropertyKeyConst.ENDPOINT_PORT, "8080");
NamingService namingService = NacosFactory.createNamingService(properties);
```

部署模式对比

	直连模式	VIP 模式	地址服务器模式
转发模式	直连	代理（网络多一跳）	直连
高可用	弱，代码配置不灵活，节点故障时无法批量变更	强	强
可伸缩性	弱	强	强
部署成本	无	负载均衡组件运维成本高	地址服务器运维成本低
负载均衡模式	nacos-sdk 客户端负载均衡	负载均衡组件提供负载均衡能力	nacos-sdk 客户端负载均衡
开源接受度	高	高	低，地址服务器模式在开源领域不太普遍
企业级能力	不方便	灵活	灵活
跨网络	内网环境，平坦网络	VIP 模式灵活地支持反向代理、安全组、ACL 等特性，可以很好的工作在内/外网环境中，使得应用服务器和 Nacos Server 可以部署在不同的网络环境中，借助 VIP 打通	内网环境，平坦网络
推荐使用环境	开发测试环境	生产环境，云环境	生产环境

当然，组合使用地址服务器 + VIP 也是可以的，可以充分的融合两者的优势：



MSE Nacos 的实践

上述场景主要介绍了三种模式的具体部署方案，以及自建 Nacos 场景如何做到高可用，最后要介绍的是阿里云环境 MSE 是如何部署的。

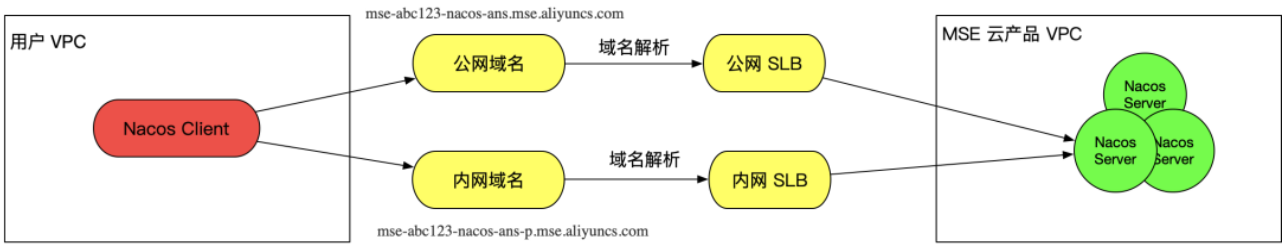
MSE（微服务引擎）提供了 Nacos 注册中心中心的全托管能力，除了要做上述提到的高可用、可伸缩、易用性，还要考虑以下的因素：

开源接受度。避免给用户带来太多理解成本，尽量做到对标开源，这样用户接受度才会高。

网络隔离。MSE 提供的是 BaaS 化的能力，Nacos Server 部署在云产品 VPC，与用户 VPC 是隔离的，需要解决网络隔离问题。

网络安全。MSE Nacos 是独享模式，网络上租户隔离是最基本的要求。除此之外企业级用户会对 MSE Nacos 提出安全组/ACL 控制的诉求，这些都需要考量。

综上，MSE Nacos 最终采用的是域名 + SLB 的 VIP 模式。



MSE Nacos 提供两个域名，其中公网域名可以用做本地开发测试，或者自建环境、混合云等场景的接入点，内网域名用做阿里云生产环境接入点。公网域名有带宽限制，需要在集群创建时根据场景选择合适的带宽，而内网域名则没有带宽限制。公网域名请注意添加 IP 访问白名单。

1.3 跨注册中心服务同步实践

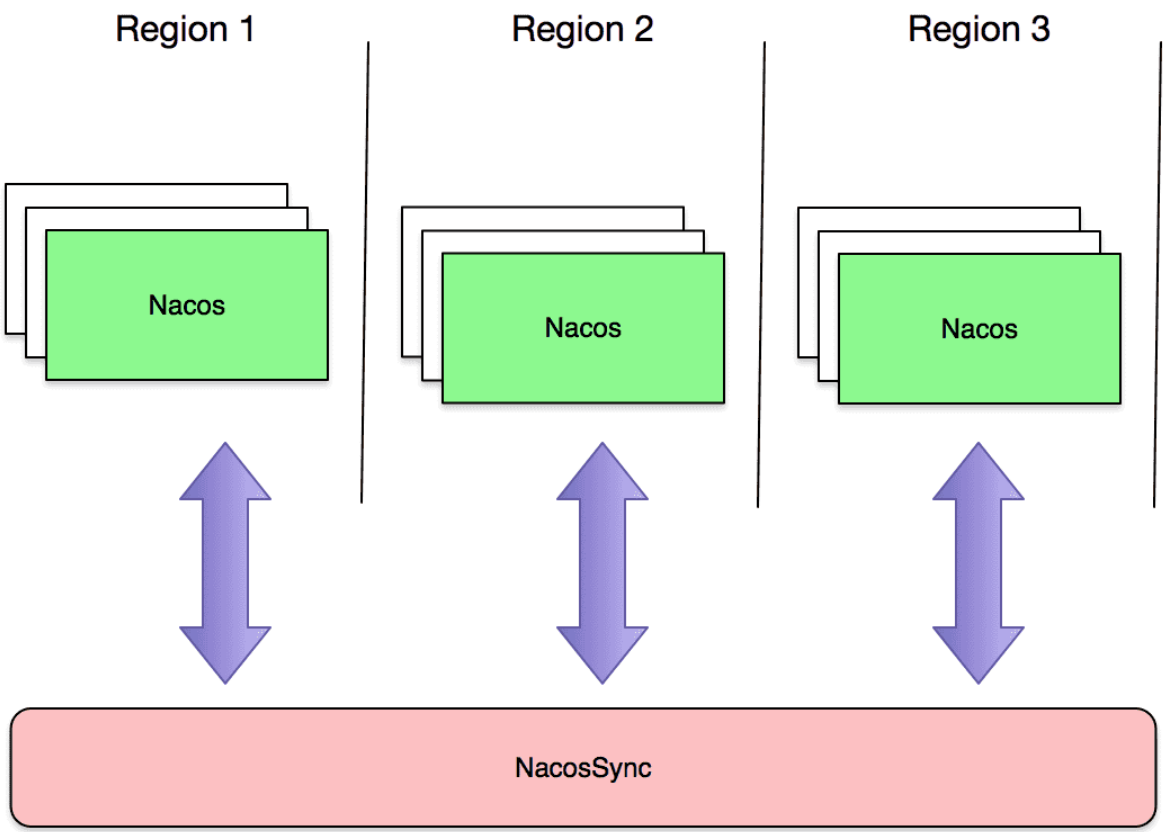
NacosSync是一个支持多种注册中心的同步组件，目前已支持的同步类型:

- Nacos数据同步到Nacos
- Zookeeper数据同步到Nacos
- Nacos数据同步到Zookeeper
- Eureka数据同步到Nacos
- Consul数据同步到Nacos

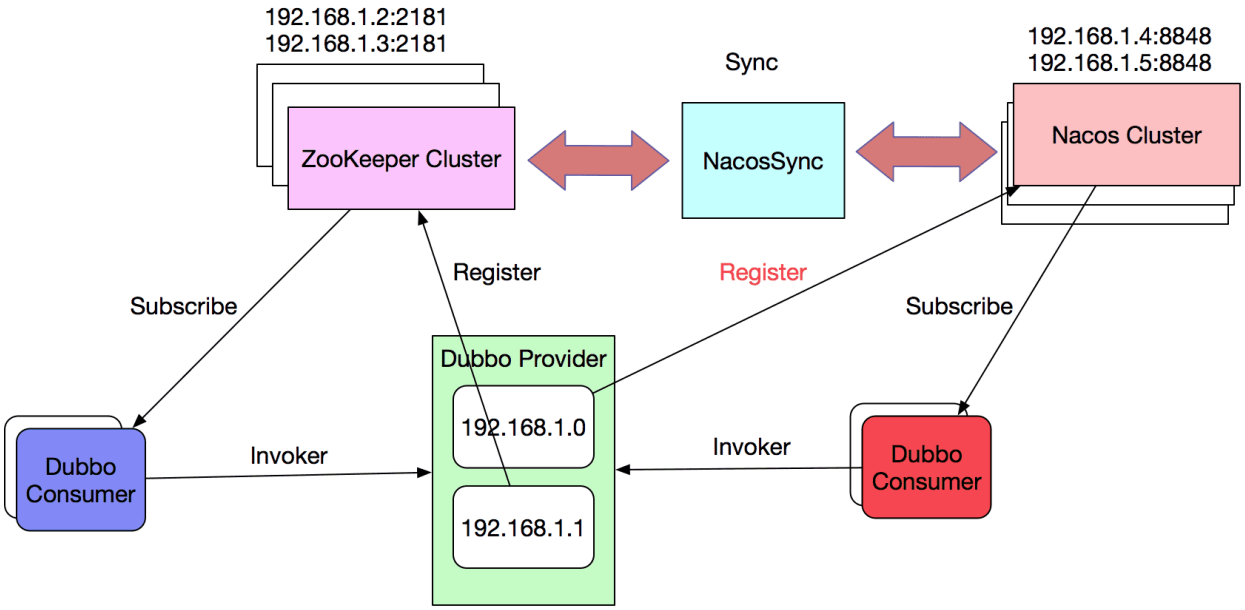
> https://nacos.io/zh-cn/docs/nacos-sync.html

使用场景

- 多个网络互通的Region之间服务共享,打破Region之间的服务调用限制



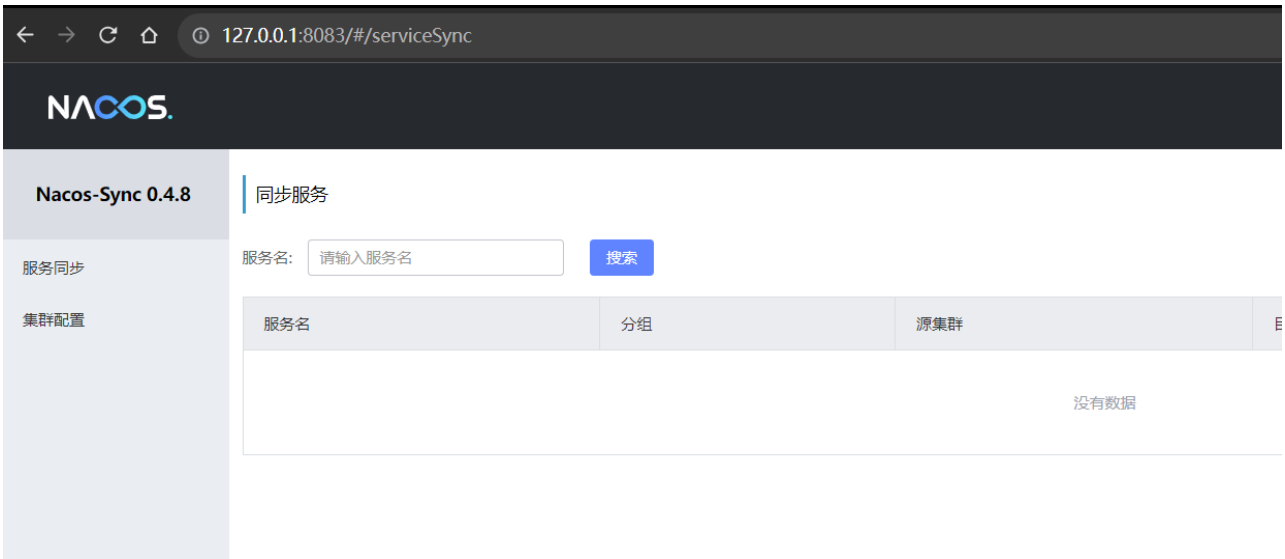
- 双向同步功能,支持Dubbo+Zookeeper服务平滑迁移到Dubbo+Nacos,享受Nacos更加优质的服务



安装NacosSync

> https://nacos.io/zh-cn/docs/v2/ecology/use-nacos-sync.html

启动服务后，访问控制台http://127.0.0.1:8083/#/serviceSync

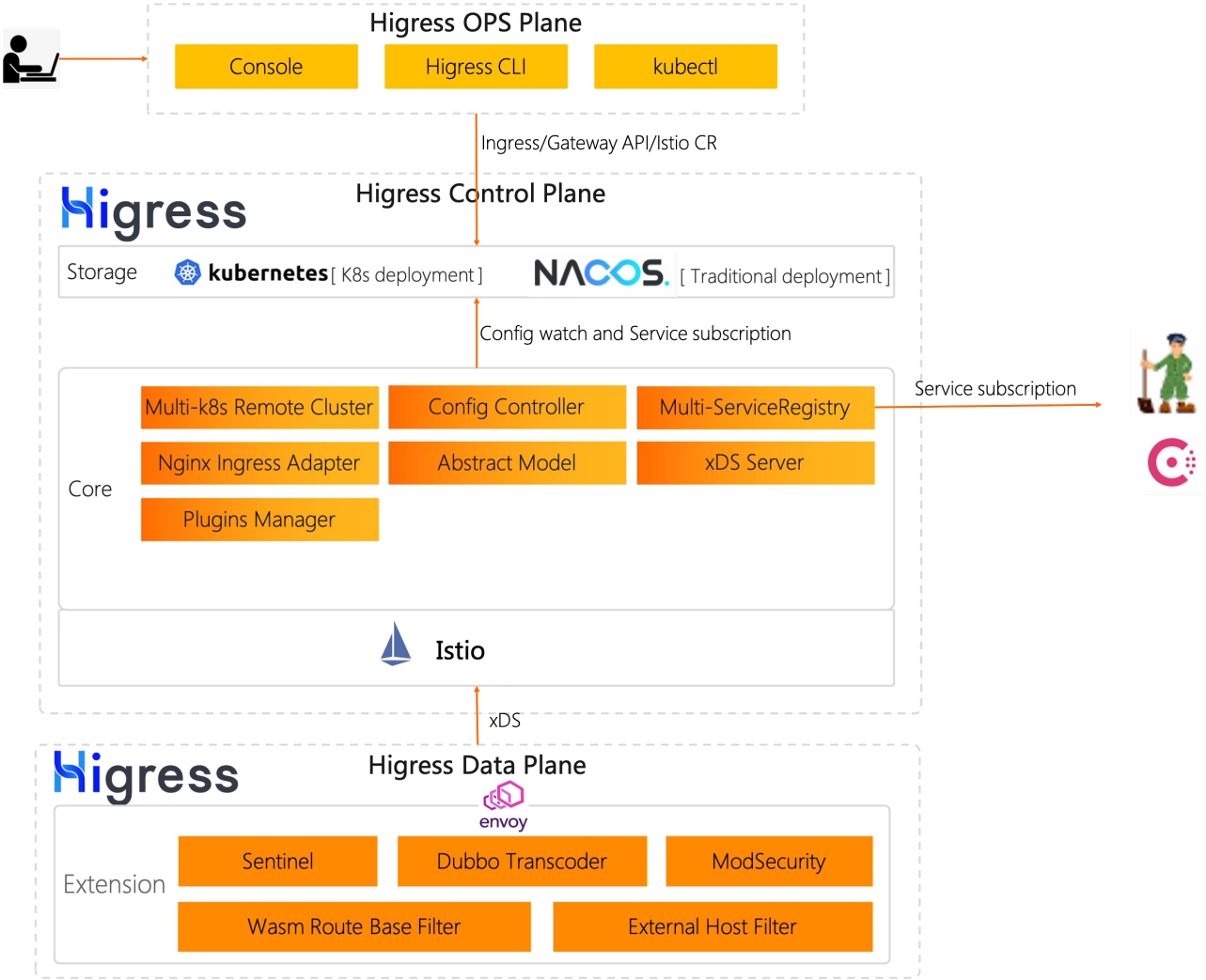


测试： 跨nacos注册中心服务同步

2. 云原生网关Higress实战

2.1 Higress是什么

Higress是基于阿里内部的Envoy Gateway实践沉淀、以开源Istio + Envoy为核心构建的下一代云原生网关，实现了流量网关 + 微服务网关 + 安全网关三合一的高集成能力，深度集成Dubbo、Nacos、Sentinel等微服务技术栈，能够帮助用户极大的降低网关的部署及运维成本且能力不打折；在标准上全面支持Ingress与Gateway API，积极拥抱云原生下的标准API规范；同时，Higress Controller也支持Nginx Ingress平滑迁移，帮助用户零成本快速迁移到Higress。



2.2 Higress快速开始

基于docker compose安装Higress

使用独立部署的 Nacos

```
curl -fsSL https://higress.io/standalone/get-higress.sh | bash -s -- -c
nacos://192.168.65.174:8848  -p admin
```

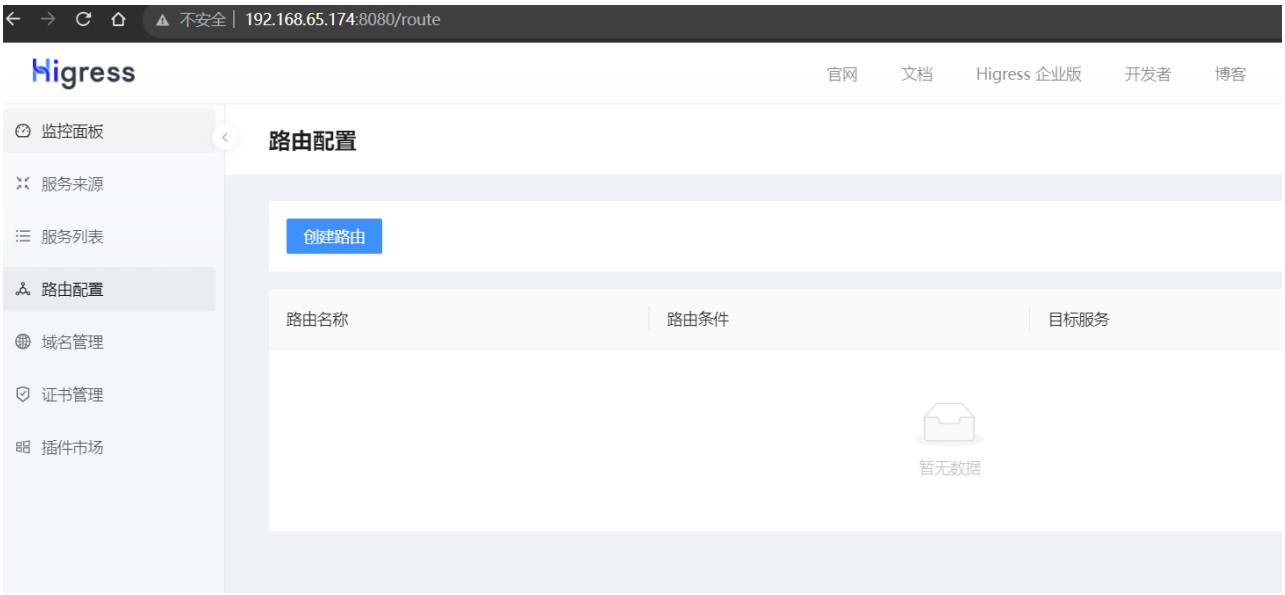
启动成功后，本机端口占用情况如下：

- 80端口：Higress 暴露，用于 HTTP 协议代理
- 443端口：Higress 暴露，用于 HTTPS 协议代理

- 15020端口：Higress 暴露，用于暴露 Prometheus 指标
- 8080端口：Higress 控制台 暴露，（admin/123456）

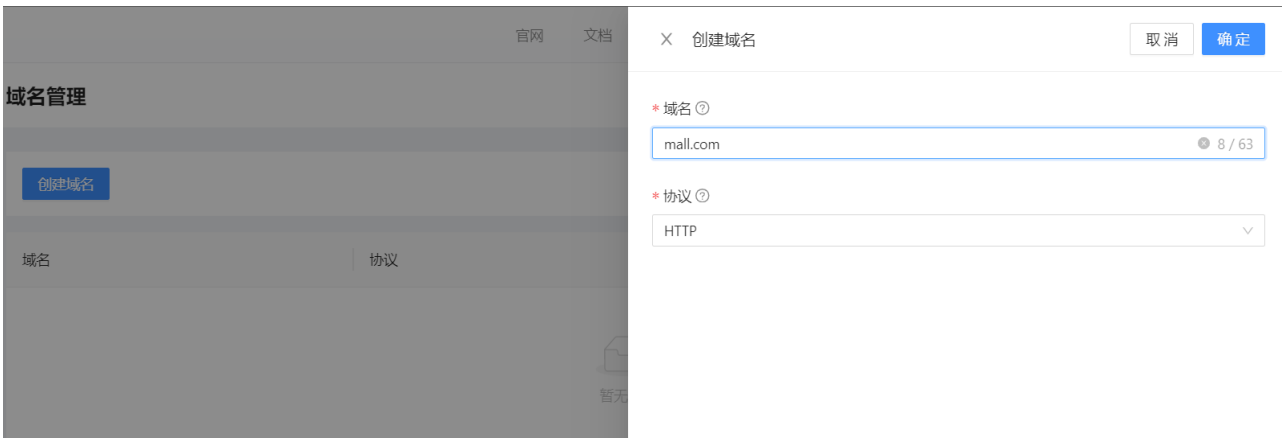
```
[root@hadoop01 higress]# curl -fsSL https://higress.io/standalone/get-higress.sh | bash -s -- -c nacos://192.168.65.174:8848  -p
admin
Downloading https://higress.io/standalone/higress-v0.7.0.tar.gz...
==== Build Configurations ====
[+] Building 0.0s (0/0)
[+] Creating 1/0
 ✔ Network higress_higress-net  Created                                                                          0.0s
[+] Running 4/4
 ✔ initializer 3 layers [▒▒▒]      0B/0B      Pulled                                                             10.6s
   ✔ 3153aa388d02 Pull complete                                                                                  8.7s
   ✔ 41c105be7bce Pull complete                                                                                  8.8s
   ✔ 521ca7ce1993 Pull complete                                                                                  9.3s
[+] Building 0.0s (0/0)
Nacos is ready.
Initializing Nacos server...
  Creating namespace higress-system...
Initializing API server configurations...
  Generating CA certificate...
  Generating server certificate...
  Generating data encryption key...
  Generating client certificate...
  Generating kubeconfig...
Initializing controller configurations...
```

在浏览器中输入http://192.168.65.174:8080/，使用用户名 admin 和安装时设置的密码登录 Higress 控制台



配置 higress 域名

配置域名解析至 higress 所在机器的 80/443 端口

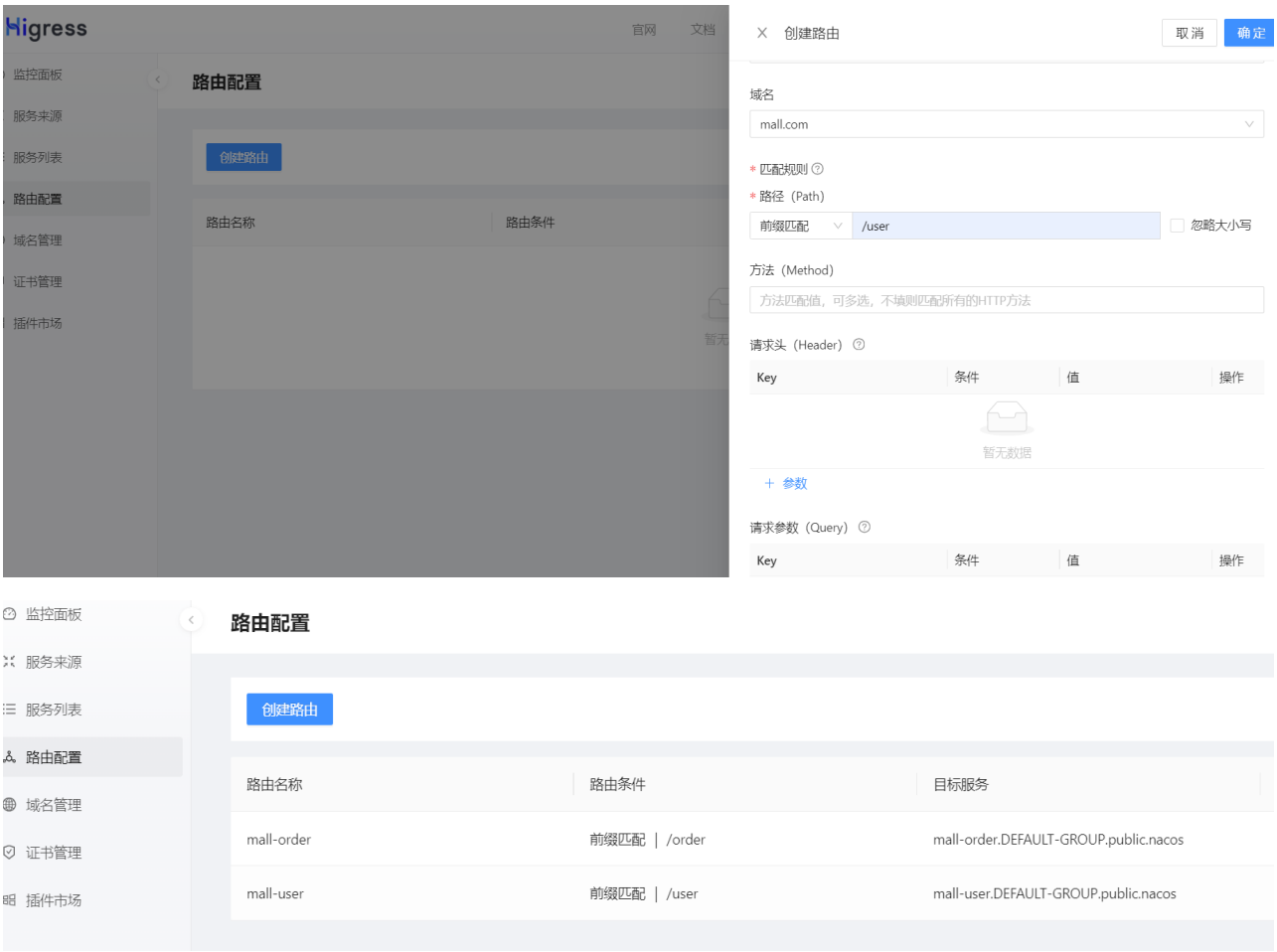


服务列表

服务列表自动同步指向的 nacos 中已注册服务列表

配置路由转发

在路由管理中创建两条新的路由，根据路由前缀将其转发至相应的微服务



测试

访问，验证测试路由是否可以正常工作

```
curl localhost/user/findOrderByUserId/1 -H 'host: mall.com'
```

基于 K8s 集群安装Higress

> 利用sealos快速安装kubernetes集群：https://note.youdao.com/s/M2z4OzsL

安装Higress

```
# 使用Helm 安装Higress
helm repo add higress.io https://higress.io/helm-charts
helm install higress -n higress-system higress.io/higress --create-namespace --render-subchart-notes --set higress-console.domain=console.higress.io
```

注意：安装完成后会输出一段文本，其中包含获取控制台登录信息的命令。请执行该命令并记录用户名和密码。

```
NOTES:
1. Get the application URL by running these commands:
  export POD_NAME=$(kubectl get pods --namespace higress-system -l "app.kubernetes.io/name=higress-console,app.kubernetes.io/instance=higress" -o jsonp
ath="{.items[0].metadata.name}")
  export CONTAINER_PORT=$(kubectl get pod --namespace higress-system $POD_NAME -o jsonpath="{.spec.containers[0].ports[0].containerPort}")
  echo "Visit http://127.0.0.1:8080 to use your application"
  kubectl --namespace higress-system port-forward $POD_NAME 8080:$CONTAINER_PORT
2. Use following commands to get the credential and login:
  export ADMIN_USERNAME=$(kubectl get secret --namespace higress-system higress-higress-console -o jsonpath="{.data.adminUsername}" | base64 -d)
  export ADMIN_PASSWORD=$(kubectl get secret --namespace higress-system higress-higress-console -o jsonpath="{.data.adminPassword}" | base64 -d)
  echo -e "Username: ${ADMIN_USERNAME}\nPassword: ${ADMIN_PASSWORD}"
  NOTE: If this is an upgrade release, your current password won't be changed.
3. If you'd like to change the credential, you can edit this secret with new values: higress-system/higress-higress-console
Higress successfully installed!
```

例如安装在 higress-system 命名空间下时，执行下面命令获取用户名密码：

```
  export ADMIN_USERNAME=$(kubectl get secret --namespace higress-system higress-console -o jsonpath="{.data.adminUsername}" | base64 -d)
  export ADMIN_PASSWORD=$(kubectl get secret --namespace higress-system higress-console -o jsonpath="{.data.adminPassword}" | base64 -d)
  echo -e "Username: ${ADMIN_USERNAME}\nPassword: ${ADMIN_PASSWORD}"
```

```
[root@k8s-master01 ~]# export ADMIN_USERNAME=$(kubectl get secret --namespace higress-system higress-console -o jsonpath="{.data.
adminUsername}" | base64 -d)
}" | base64 -d)
  echo -e "Username: ${ADMIN_USERNAME}\nPassword: ${ADMIN_PASSWORD}"[root@k8s-master01 ~]#   export ADMIN_PASSWORD=$(kubectl get
a.adminPassword}" | base64 -d)tem higress-console -o jsonpath="{.dat
[root@k8s-master01 ~]#   echo -e "Username: ${ADMIN_USERNAME}\nPassword: ${ADMIN_PASSWORD}"
Username: admin
Password: GGR0EL51
```

获取 Higress Gateway 的 LoadBalancer IP，并记录下来。后续可以通过该 IP 的 80 和 443 端口访问 Higress Gateway

```
[root@k8s-master01 ~]# kubectl get svc -n higress-system
NAME                TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)                                                      AGE
higress-console     ClusterIP      10.96.0.191    <none>        8080/TCP                                                     22h
higress-controller  ClusterIP      10.96.3.186    <none>        8888/TCP,15051/TCP,15010/TCP,15012/TCP,443/TCP,15014/TCP   22h
higress-gateway     LoadBalancer   10.96.1.19     <pending>     80:30332/TCP,443:31448/TCP                                   22h
```

配置Higress

假设在 default 命名空间下已经部署了一个名为 foo 的服务，而我们希望创建一个对应 http://foo.bar.com/foo 的路由指向该服务。

如果需要的话，各位可以使用下方 YAML 来创建对应的测试服务。

```
kind: Pod
apiVersion: v1
metadata:
  name: foo-app
  labels:
    app: foo
spec:
```

```
  containers:
  - name: foo-app
    image: higress-registry.cn-hangzhou.cr.aliyuncs.com/higress/http-echo:0.2.4-alpine
    args:
    - "-text=foo"
---
kind: Service
apiVersion: v1
metadata:
  name: foo-service
spec:
  selector:
    app: foo
  ports:
  # Default port used by the image
  - port: 5678
```

使用 Ingress CRD 进行路由配置，编写foo-ingress.yaml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: foo
spec:
  ingressClassName: higress
  rules:
  - host: foo.bar.com
    http:
      paths:
      - pathType: Prefix
        path: "/foo"
        backend:
          service:
            name: foo-service
            port:
              number: 5678
```

执行如下命令生效规则：

```
kubectl apply -f foo-ingress.yaml
```

测试

执行以下命令，验证测试路由可以正常工作：

```
curl http://10.96.1.19/foo -H 'host: foo.bar.com'
```

```
[root@k8s-master01 ~]# curl http://10.96.1.19/foo -H 'host: foo.bar.com'
foo
```

2.3 电商项目接入Higress

以微服务tulingmall-product为例

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tulingmall-product-deployment
  labels:
    app: tulingmall-product
spec:
  replicas: 2
  selector:
    matchLabels:
      app: tulingmall-product
  template:
    metadata:
      labels:
        app: tulingmall-product
    spec:
      hostNetwork: true  # 主机模式，初学者建议使用此模式
      imagePullSecrets:
```

```
      - name: myregistrykey  #私有仓库的secret
      containers:
        - name: tulingmall-product
          image: registry.cn-hangzhou.aliyuncs.com/fox666/tulingmall-product:0.0.5
          imagePullPolicy: Always
          ports:
            - containerPort: 8866
          env:
            - name: TZ
              value: Asia/Shanghai
            - name: spring.cloud.nacos.config.server-addr
              value: 192.168.65.174:8848
            - name: LOG_FILE
              value: /var/logs
          volumeMounts:
            - mountPath: /var/logs
              name: log-volume
      volumes:
        - name: log-volume
          hostPath:
            path: /mydata/k8s-app/tulingmall-product/logs

---

apiVersion: v1
kind: Service
metadata:
  name: tulingmall-product-service
spec:
  type: NodePort
  selector:
    app: tulingmall-product
  ports:
  - name: http
    protocol: TCP
    port: 8866
    targetPort: 8866
```

```
[root@k8s-master01 tulingmall]# kubectl get svc
NAME                            TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)          AGE
foo-service                     ClusterIP   10.96.1.177   <none>        5678/TCP         22h
kubernetes                      ClusterIP   10.96.0.1     <none>        443/TCP          23h
tulingmall-authcenter-service   NodePort    10.96.1.254   <none>        9999:32629/TCP   21h
tulingmall-gateway-service      NodePort    10.96.0.200   <none>        8888:30630/TCP   21h
tulingmall-member-service       NodePort    10.96.1.147   <none>        8877:31766/TCP   21h
tulingmall-product-service      NodePort    10.96.1.76    <none>        8866:32718/TCP   21h
```

编写路由配置higress-tulingmall.yml

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: product-route
spec:
  ingressClassName: higress
  rules:
  - host: product.tulingmall.com
    http:
      paths:
      - pathType: Prefix
        path: "/pms"
        backend:
          service:
            name: tulingmall-product-service
            port:
              number: 8866
```

执行如下命令生效规则：

```
kubectl apply -f higress-tulingmall.yml
```

查看生效的higress规则

```
[root@k8s-master01 tulingmall]# kubectl get ing
NAME            CLASS     HOSTS                   ADDRESS   PORTS   AGE
foo             higress   foo.bar.com                       80      3h54m
product-route   higress   product.tulingmall.com            80      3h12m
```

测试

```
curl http://10.96.1.19/pms/productInfo/29 -H 'host: product.tulingmall.com'
```

```
[root@k8s-master01 tulingmall]# curl http://10.96.1.19/pms/productInfo/29 -H 'host: product.tulingmall.com'
{"code":200,"message":"操作成功","data":{"id":29,"brandId":51,"productCategoryId":19,"feightTemplateId":0,"productAttributeCatego
ryId":3,"name":"Apple iPhone 8 Plus 64GB 红色特别版 移动联通电信4G手机","pic":"http://macro-oss.oss-cn-shenzhen.aliyuncs.com/mall
/images/20180615/5acc5248N6a5f81cd.jpg","productSn":"7437799","deleteStatus":0,"publishStatus":1,"newStatus":0,"recommandStatus":
0,"verifyStatus":0,"sort":0,"sale":0,"price":5499.00,"promotionPrice":null,"giftGrowth":5499,"giftPoint":5499,"usePointLimit":0,"
subTitle":"【限时限量抢购】Apple产品年中狂欢节，好物尽享，美在智慧！速来 >> 勾选[保障服务][原厂保2年]，获得AppleCare+全方位服务计
划，原厂延保售后无忧。 ","originalPrice":5499.00,"stock":100,"lowStock":0,"unit":"","weight":0.00,"previewStatus":0,"serviceIds":"
","keywords":"","note":"","albumPics":"","detailTitle":"","promotionStartTime":null,"promotionEndTime":null,"promotionPerLimit":0
,"promotionType":0,"brandName":"苹果","productCategoryName":"手机通讯","description":"","detailDesc":"","detailHtml":"","detailMo
bileHtml":"","productLadderList":[{"id":67,"productId":29,"count":0,"discount":0.00,"price":0.00}],"productFullReductionList":[{"
id":62,"productId":29,"fullPrice":0.00,"reducePrice":0.00}],"memberPriceList":[{"id":200,"productId":29,"memberLevelId":3,"member
Price":null,"memberLevelName":"钻石会员"},{"id":199,"productId":29,"memberLevelId":2,"memberPrice":null,"memberLevelName":"白金会
员"},{"id":198,"productId":29,"memberLevelId":1,"memberPrice":null,"memberLevelName":"黄金会员"}],"skuStockList":[{"id":106,"prod
uctId":29,"skuCode":"201808270029001","price":5499.00,"stock":100000,"lowStock":null,"sp1":"金色","sp2":"32G","sp3":null,"pic":nu
ll,"sale":null,"promotionPrice":null,"lockStock":null},{"id":107,"productId":29,"skuCode":"201808270029002","price":6299.00,"stoc
k":100,"lowStock":null,"sp1":"金色","sp2":"64G","sp3":null,"pic":null,"sale":null,"promotionPrice":null,"lockStock":null},{"id":1
08,"productId":29,"skuCode":"201808270029003","price":5499.00,"stock":100,"lowStock":null,"sp1":"银色","sp2":"32G","sp3":null,"pi
c":null,"sale":null,"promotionPrice":null,"lockStock":null},{"id":109,"productId":29,"skuCode":"201808270029004","price":6299.00,
"stock":100,"lowStock":null,"sp1":"银色","sp2":"64G","sp3":null,"pic":null,"sale":null,"promotionPrice":null,"lockStock":null}],"
productAttributeValueList":[{"id":227,"productId":29,"productAttributeId":48,"value":"1960ml"},{"id":226,"productId":29,"productA
ttributeId":47,"value":"IOS"},{"id":225,"productId":29,"productAttributeId":46,"value":"4G"},{"id":224,"productId":29,"productAtt
ributeId":45,"value":"4.7"},{"id":223,"productId":29,"productAttributeId":43,"value":"金色,银色"}],"flashPromotionPrice":4999.00,
```