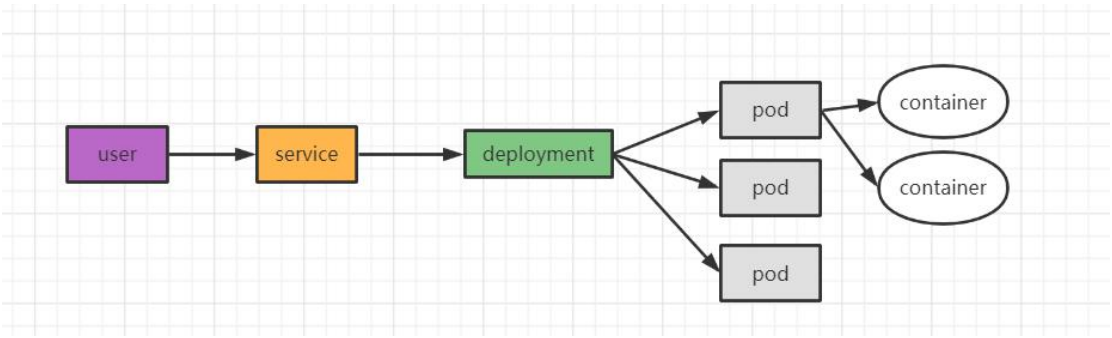
一、发布流程图

1、回忆下 k8s 的结构



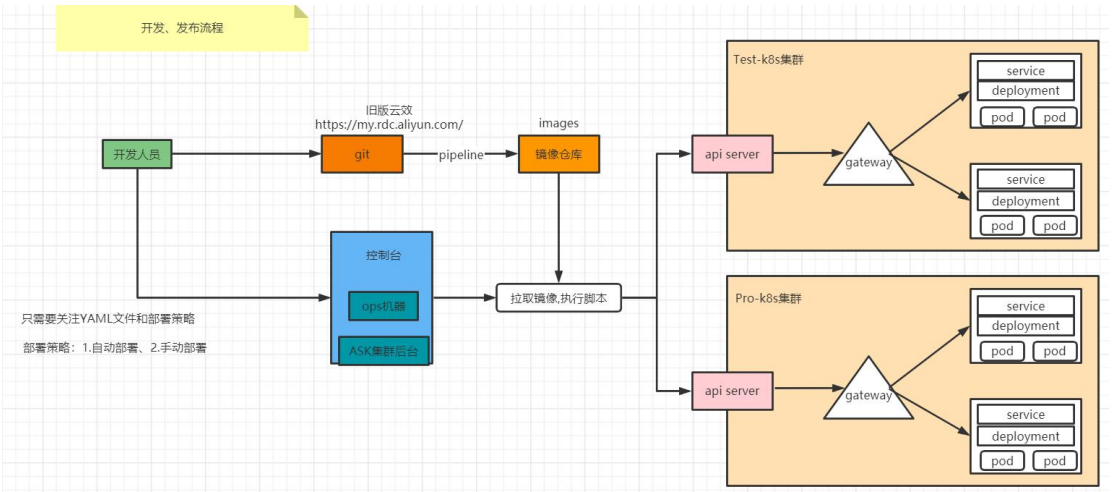
k8s 中任何东西都是资源对象，通过 yaml 创建资源的时候需要指定 kind，kind 常见类型 service、deployment 等

container ： docker 容器，运行的是业务代码+环境
pod ： k8s 的最小单元；容器组
https://www.cnblogs.com/kevingrace/p/11309409.html
kubernetes 为什么使用 pod 作为最小单元，而不是 container?
kubernetes 为什么允许一个 pod 里有多个容器?
deployment 与 service ：
service 是比 deployment 更高一层的抽象层。deployment 是通过 ReplicaSet 来管理一批功能相同的 pod，如果有 pod 数量很多的话，我们一个个去访问很不方便。 因此，k8s 将 deployment 下的一批 pod 应用抽象出来作为一个抽象层，并自动进行负载均衡，这样我们需要访问 pod 应用时只需访问 service 抽象层即可。

2、流程分几步

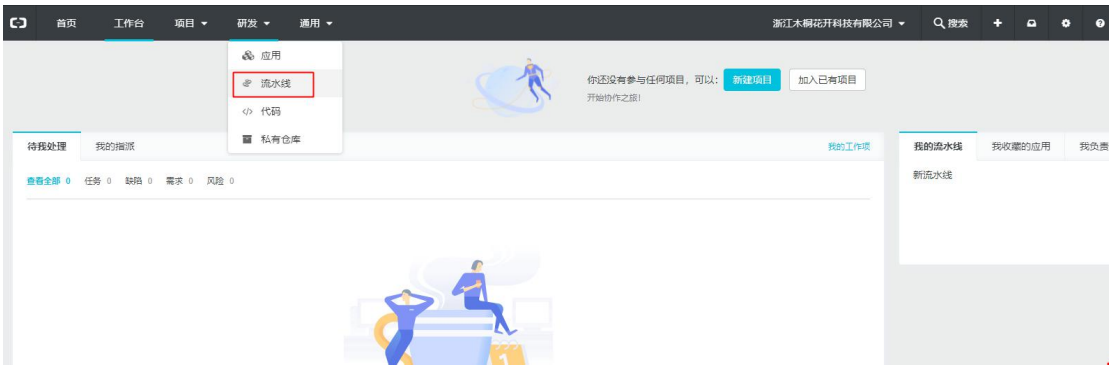
1、开发人员提交代码到 git
2、本地测试没问题，触发 pipeline 生成镜像到镜像仓库
3、关注需要发布的服务，执行相应的 yaml

https://www.processon.com/view/link/61273218e0b34d14f763e74e

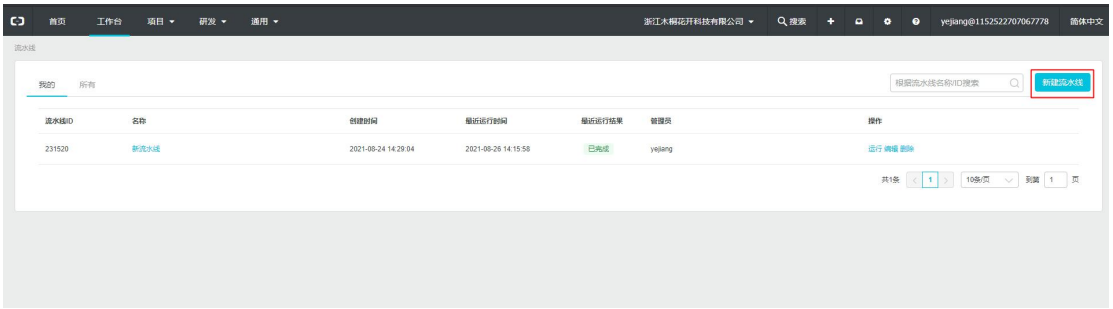


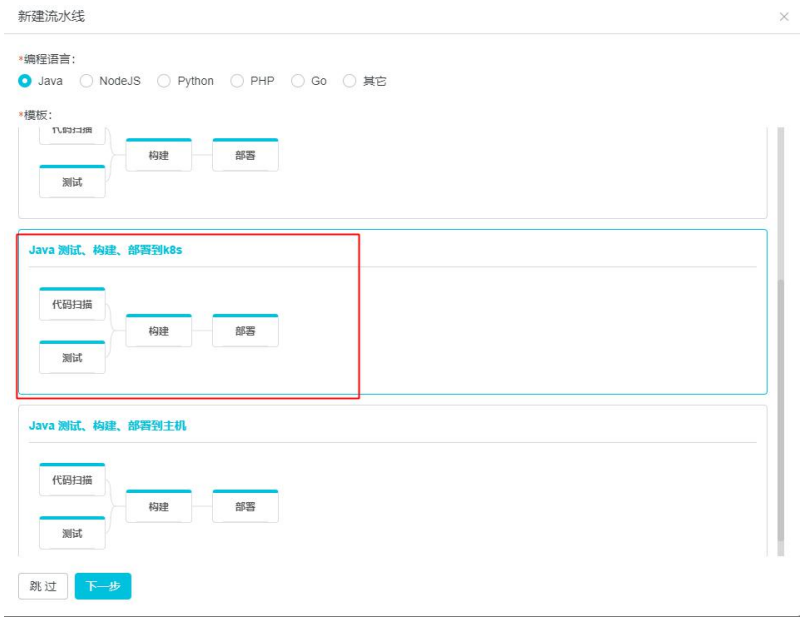
二、pipeline 的使用

1、入口：研发->流水线

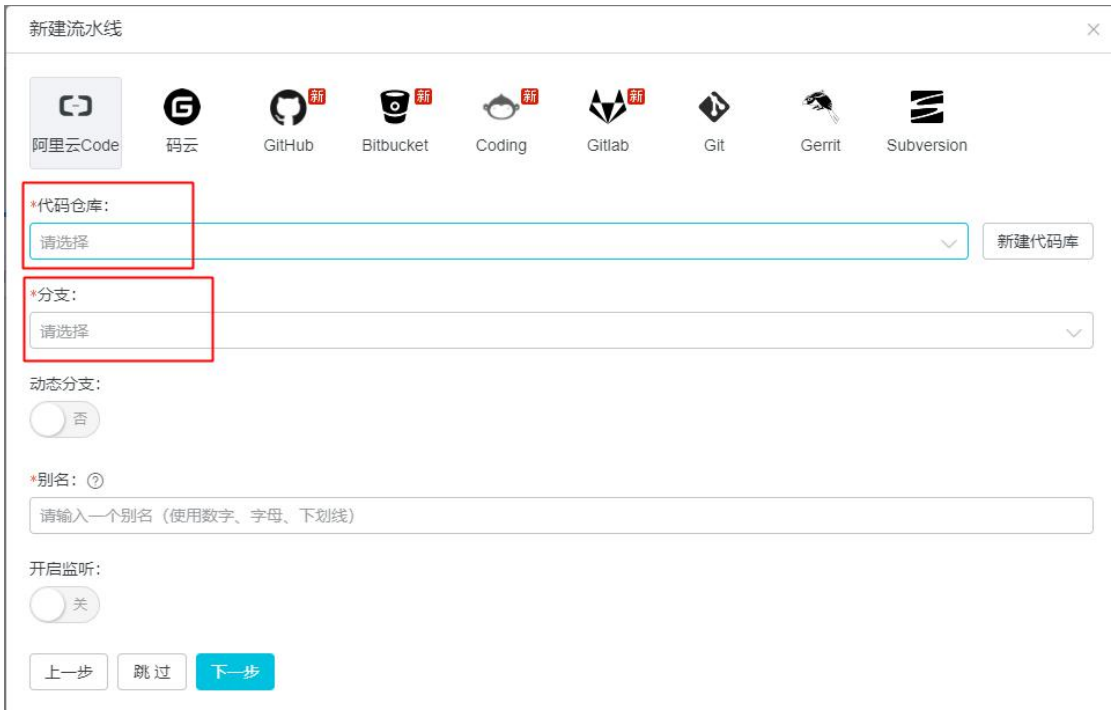


2、点击创建流水线，选择“Java 测试、构建、部署到 k8s”

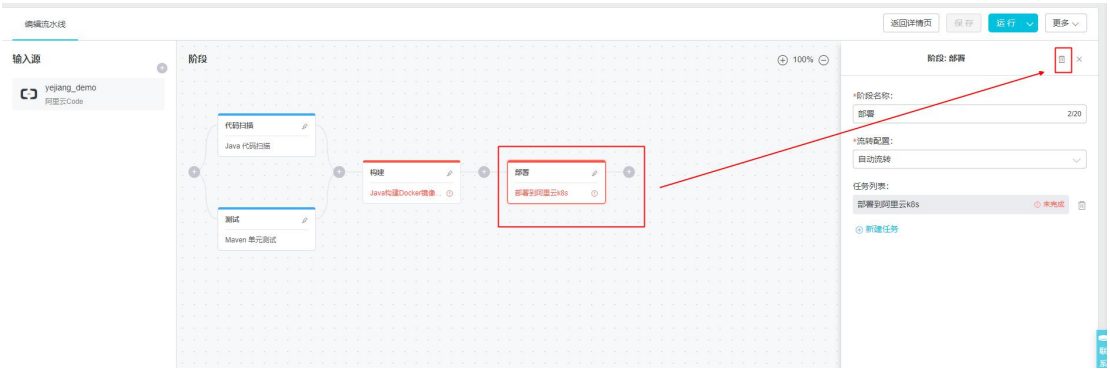




3、点击下一步，可选择代码仓库和分支



4、生成流水线后，我们可以删掉“部署”，我们通过脚本方式部署到 K8S



5、设置构建 docker 参数

点击构建填写右边菜单

点击任务列表修改参数：

step1、不需要修改按默认的即可

step2、填写构建 docker 的参数

区域填写的是自己镜像仓库的区域

镜像仓库是阿里云镜像服务里的地址

标签填写的是生成 docker 镜像的标签，图里${version}使用的是配置的全局变量

Dockerfile 路径是指当前打包模块 Dockerfile 存放位置

ContextPath 路径是指当前模块是在当前代码仓库的位置

构建步骤：

*步骤类型

Java 构建

*步骤名称

step1-Java 构建 13/20

*请选择Java版本：

jdk1.8

*构建命令：

```
# maven build default command
mvn -B clean package -Dmaven.test.skip=true
    -Dautoconfig.skip

# gradle build default command
# ./gradlew build

# ant build default command
```

*步骤类型

Docker镜像构建上传

*步骤名称

step2-Docker镜像构建上传 18/20

*区域:

华北2（北京）

*仓库:

registry.cn-beijing.aliyuncs.com/edipao/docker...

*标签:

provider-${version}

docker镜像Tag

*Dockerfile路径:

provider/Dockerfile

ContextPath:

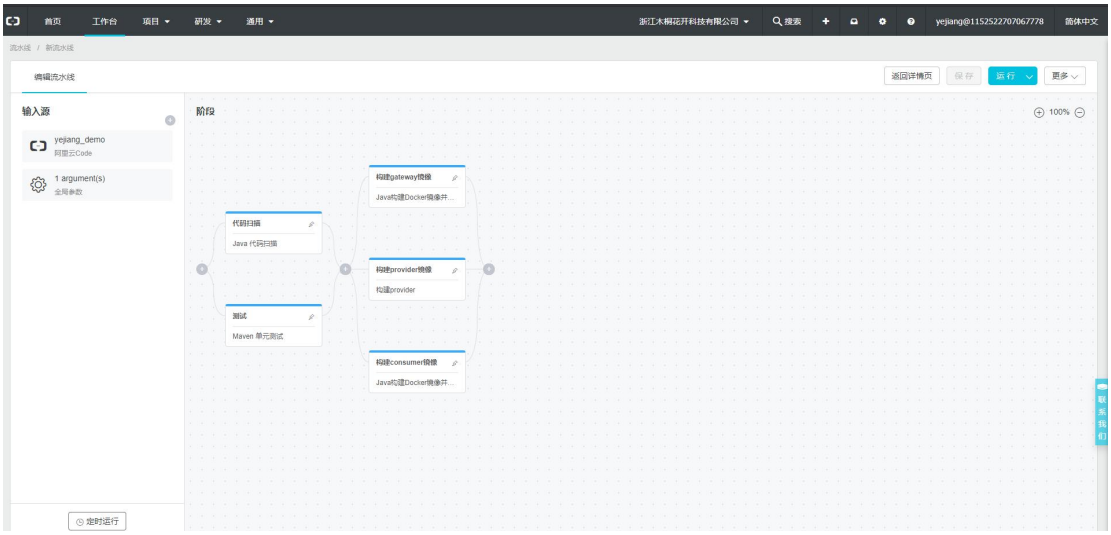
provider

不使用缓存:

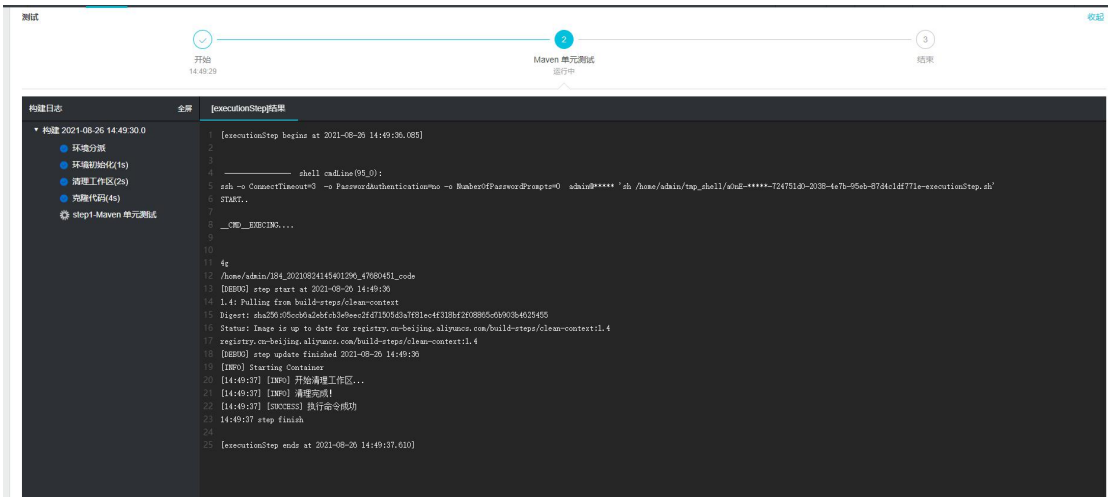
构建参数:

添加参数

6、多模块的话，可以设置多个镜像构建，步骤和上面一样，只需在阶段图添加构建即可



7、运行 pipeline



三、K8S 服务发现

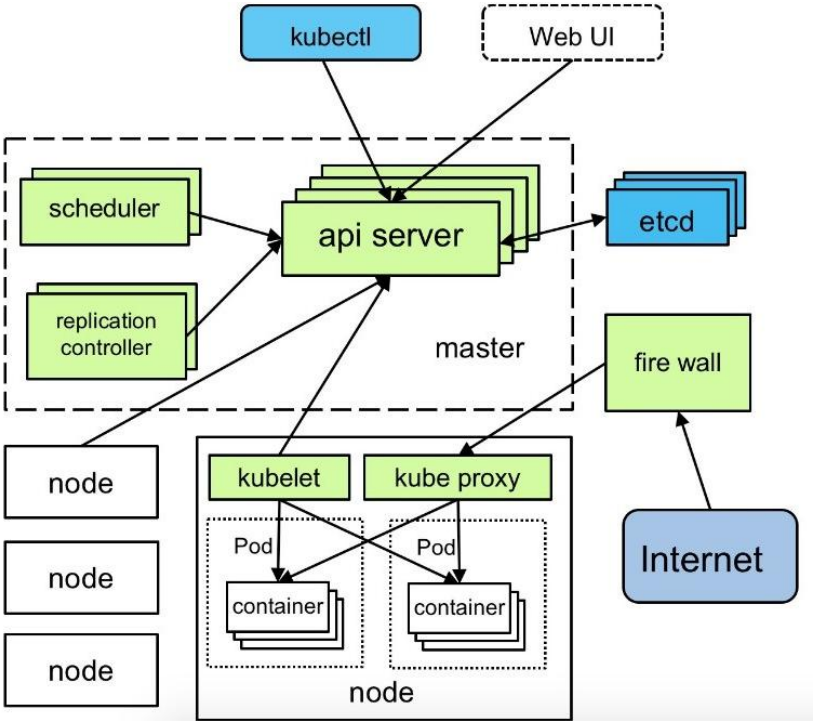
1、项目地址

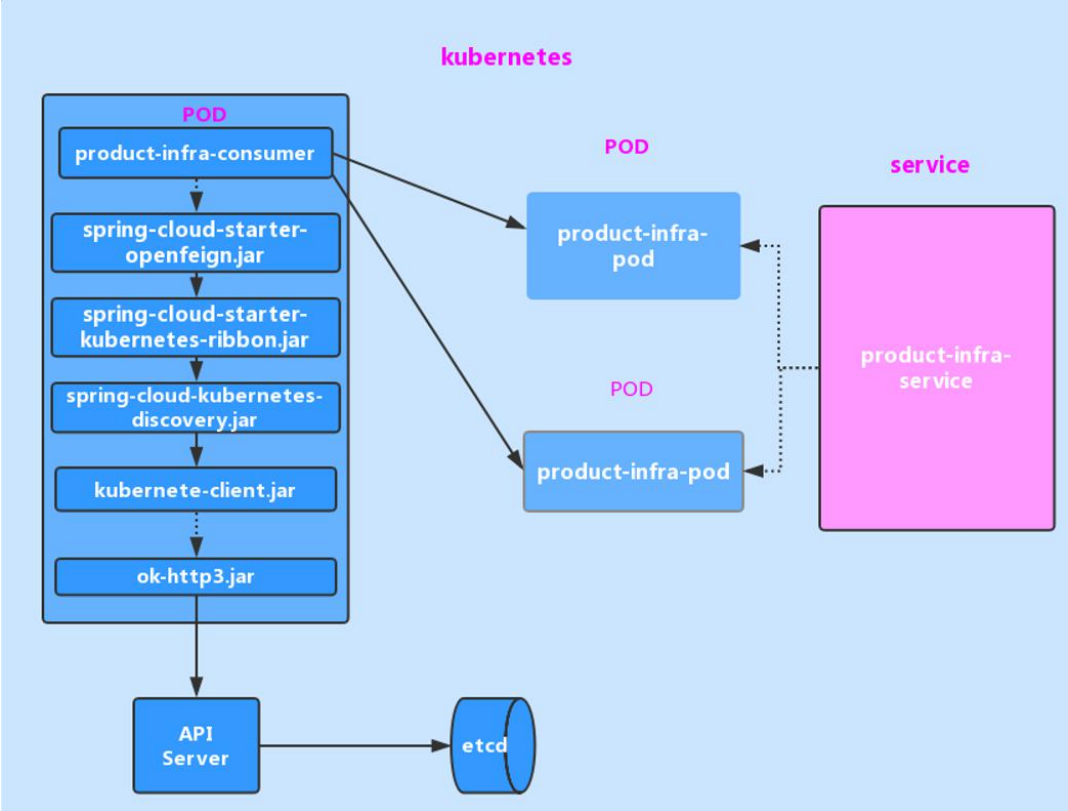
https://github.com/MuyerJ/springcloud-k8s

2、创建步骤

1、创建 gatway 模块（参考 github）
2、创建 provider 模块（参考 github）
3、创建 consumer 模块（参考 github）
4、创建 k8s 资源（脚本执行）
   (1) 创建 configMap
   (2) 创建 ServiceAccount(给服务有访问 Api Server 的权限)
   (3) 创建 gateway、customers、provider

3、spring cloud kubernetes 调用过程





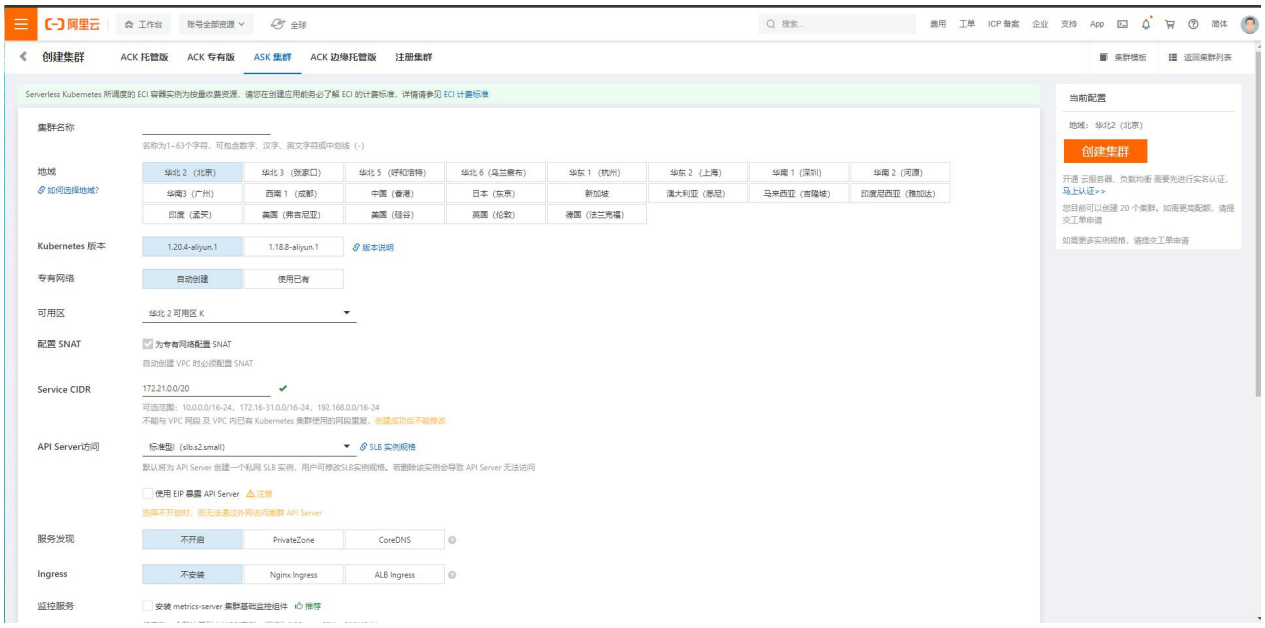
从上图可以看出 product-infra-consumer 在调用 product-infra-service 时，通过 FeignClient 组件拿到 service name 信息，最底层通过 ok-http3，根据 service name 调用 api server 获取该 service 下对应的 Pod 信息，拿到 Pod 信息后通过，轮询的方式向这些 pod 发送请求。

spring-cloud-starter-kubernetes-ribbon 组件中的 KubernetesServerList 继承了 ribbon-loaderbanlancer 组件中的 AbstractServerList 以及实现了 ServerList 类中的方法，并通过 KubernetesClient 提供的能力向 k8s api server 发送请求信息。

四、ASK 的使用

1、创建集群

阿里云搜索 ASK，进入之后点击创建 ASK 集群，选择相应参数后点击创建集群即可



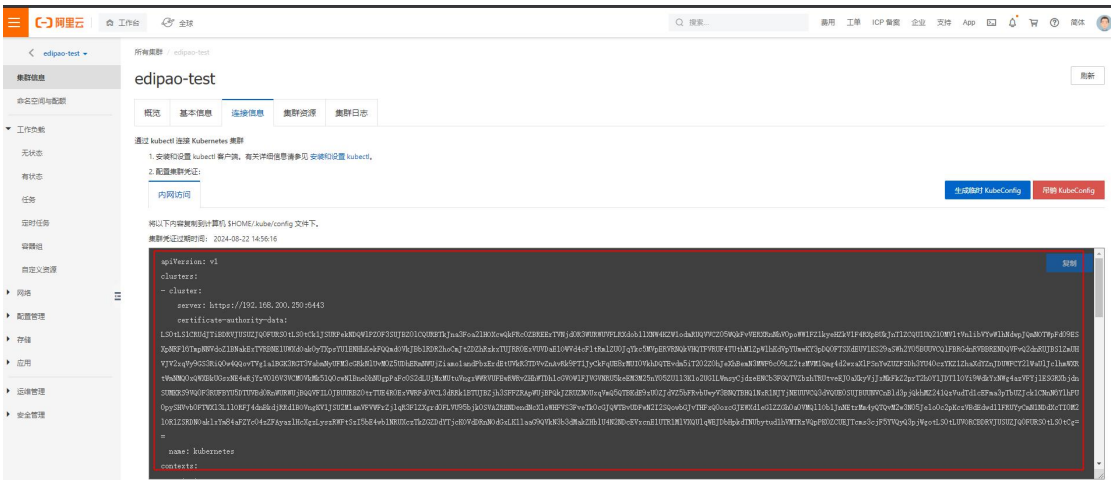
2. 通过其他计算机访问集群

前提：能够 ping 通集群

通过 kubectl 连接 Kubernetes 集群

1）安装和设置 kubectl 客户端。有关详细信息请参见 安装和设置 kubectl。

2）配置集群凭证：把凭证复制到计算机文件下 $HOME/.kube/config



连通后，我们就可以通过其他计算机进行发布、查询 k8s 集群了

3. 通过 yaml 创建发布服务

创建 gateway 服务

1、serviceAccoutName ： 指定服务账户信息，这个账户可以赋予权限，比如访问 Api-Server

2、image ： 填写当前服务在容器仓库版本

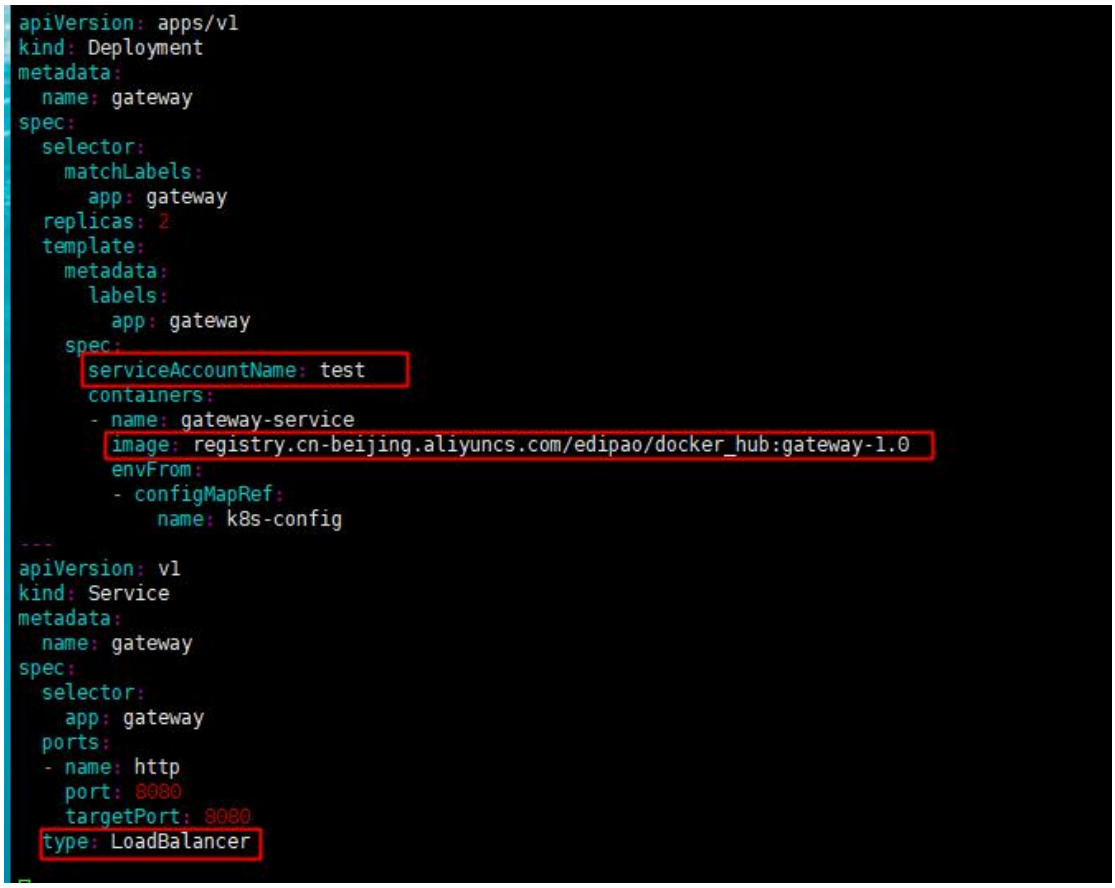
3、type ： 指定当前服务暴露 IP 的类型；

type 有三个取值:ClusterIP、NodePort 与 LoadBalance

http://www.dockerone.com/article/4884（参考理解）

ClusterIP 的方式只能在集群内部访问。

NodePort、LoadBalance 可以暴露给外部使用



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gateway
spec:
  selector:
    matchLabels:
      app: gateway
  replicas: 2
  template:
    metadata:
      labels:
        app: gateway
    spec:
      serviceAccountName: test
      containers:
      - name: gateway-service
        image: registry.cn-beijing.aliyuncs.com/edipao/docker_hub:gateway-1.0
        envFrom:
        - configMapRef:
            name: k8s-config
---
apiVersion: v1
kind: Service
metadata:
  name: gateway
spec:
  selector:
    app: gateway
  ports:
  - name: http
    port: 8080
    targetPort: 8080
  type: LoadBalancer
```

创建 provider 服务

1、type ： 选择的是 ClusterIP

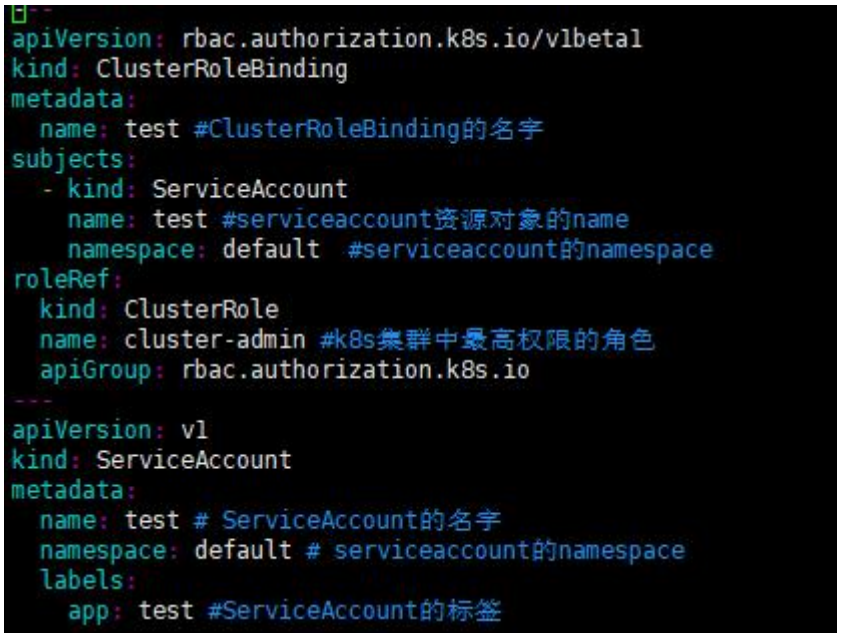
2、configMapRef： 可作为我们的配置中心，里面可以配置例如数据库信息等

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: providers
spec:
  selector:
    matchLabels:
      app: providers
  replicas: 2
  template:
    metadata:
      labels:
        app: providers
    spec:
      serviceAccountName: test
      containers:
      - name: providers
        image: registry.cn-beijing.aliyuncs.com/edipao/docker_hub:provider-1.0
        envFrom:
        - configMapRef:
            name: k8s-config
---
apiVersion: v1
kind: Service
metadata:
  name: providers
spec:
  selector:
    app: providers
  ports:
  - name: http
    port: 8080
    targetPort: 8080
  type: ClusterIP
```

创建 ConfigMap

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: k8s-config
data:
  SERVER_PORT: "8080"
  PROVIDER_ENDPOINT: http://providers:8080
  CUSTOMER_ENDPOINT: http://customers:8080
```

创建 ServiceAccount


---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
name: test #ClusterRoleBinding的名字
subjects:
- kind: ServiceAccount
name: test #serviceaccount资源对象的name
namespace: default #serviceaccount的namespace
roleRef:
kind: ClusterRole
name: cluster-admin #k8s集群中最高权限的角色
apiGroup: rbac.authorization.k8s.io
---
apiVersion: v1
kind: ServiceAccount
metadata:
name: test # ServiceAccount的名字
namespace: default # serviceaccount的namespace
labels:
app: test #ServiceAccount的标签