Kubernetes-java-client

结构

Client-java-api

--api:对外调用接口

--common:公共方法

--services:二次封装

Client-java-http:工具封装类

--auth

--http

Client-java-model:实体类

--constant 常量类

--custom 公共数据

--model kubernetes实体类

概念：

1. Namespaces :命名空间，一个虚拟的独立空间，k8s的所有服务、节点等资源都需要建立在这个虚拟空间上，并且各命名空间之间的资源相互独立。系统的命名空间为kube-system。  
   相对于我们的环境来说，一个产品：比如宁家鲜生、招商系统，都分别对应自己的namespaces，其相对的模块及服务都部署在自己的namespaces上.  
   对应model中的com.suneee.kubernetes.model.namespace.V1Namespace
2. Pod：节点，简单看可对应为docker容器，但同时也是配置最多的模组，基本配置信息包含：命名空间，名称，镜像名，(端口名称，端口号)(多个)，限定cpu资源，限定memory资源，环境变量设置(map)等。  
   相对于当前环境就是一个rest模块、一个provider模块都是一个Pod。  
   对应的model为com.suneee.kubernetes.model.pod.V1Pod
3. Deployment：部署，对pod节点进行负载管理，即可以控制同时起多少个pod，更新策略等功能。  
   相对于生产环境上一个rest模块会在不同的服务器上部署多个节点进行负载，这里就由deployment来进行控制。  
   对应的model为com.suneee.kubernetes.model.deployment.AppsV1beta1Deployment
4. Service：服务，为一组相同的pod（deployment控制）提供统一的入口地址，同命名空间下的通过服务名即可互相访问，外部访问需要声明nodePort端口。  
   基本配置信息包含：命名空间，名称，(端口名称，端口号，输出外部端口号:可选)(多个，且对应pod配置)  
   对应的model为com.suneee.kubernetes.model.service.V1Service
5. Ingress：nginx服务，提供域名及内部服务的映射功能。  
   基本配置信息包括：命名空间，名称，域名，端口号，服务名  
   相对于当前环境的nginx配置，只不过这里由k8s自动完成  
   对应的model为com.suneee.kubernetes.model.ingress.V1beta1Ingress

接口调用，在com.suneee.kubernetes.api包下：

1. NamespaceApi：对namespace进行控制的接口
2. DeploymentApi：对Deployment进行控制
3. ServiceApi：对Service进行控制
4. IngressApi：对Ingress进行控制

P.S.:当前条件下，我们所需要使用到的也就是以上这些模块，由于新的域名策略，Ingress暂时不用

接口调用：

ApiClient apiClient = ApiClient.getApiClient();

apiClient.setBasePath(ip); ip: https://10.6.250.253:6443

ApiKeyAuth BearerToken = (ApiKeyAuth) apiClient.getAuthentication("BearerToken");

BearerToken.setApiKey(bearerValue);

bearerValue: “Bearer eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJlLXN5c3RlbSIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJhZG1pbi11c2VyLXRva2VuLWtyYm1xIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQubmFtZSI6ImFkbWluLXVzZXIiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC51aWQiOiJiOTU3MzQ0Mi05M2M1LTExZTgtOWEwMy1mYTE2M2UxMTdjN2MiLCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6a3ViZS1zeXN0ZW06YWRtaW4tdXNlciJ9.mhIGMjSzoIBEmy66XY39fq2E75j7KkpXQxQXauMZVHFTyBa92mUwQdf3WRleEvSbknjKaqe6h\_5z8J7RLc1T8vJeD--qtB86V9lijKac9kOYcY0-NQ352li42wsTTyh1KwJGpHpqBrVvxX36Lyg7cmEpJ1ql3sIDN3PJz-4KX6KndfVkYtTtmFWucJnalYgRO2KaK4OWSYpiHPlTtNzNnmAjeWhaoB9KaBV1fERU63IiRAFSk8dp8VviZTvlsJa6Yq5QywxARCx96PspEKYacWbzRnQl5At3AJ45OHeGEw9eO\_SwwwFl2KWc\_yLvTcAWc\_7lEEmJluRr2dTekxsf7g”

以上初始化一次即可，ApiClient为单例。

范例：

1. Namespace：
2. 创建：com.suneee.kubernetes.api.NamespaceApi.createNamespaceDeployment(String namespace)  
   参数1：命名空间名称
3. 取得命名空间列表：

com.suneee.kubernetes.api.NamespaceApi.getNamespaces()

1. 删除命名空间：删除命名空间会导致该命名空间下的所有资源均被删除，所以不予提供。
2. Service：
3. 外部服务：外部服务指服务部署在非k8s集群中的服务（如postgresql，rocketmq等）  
   一：创建：使用封装好的com.suneee.kubernetes.services.AppApi.createEndpointService(String namespace,String name,String addr,Integer port)  
   参数1:命名空间名称;参数2:服务名称(服务调用地址写的就是该名称);参数3:服务地址(服务的ip地址);参数4:端口，外部服务的端口同时也是内部服务调用的端口

二：删除：使用封装好的

com.suneee.kubernetes.services.AppApi. deleteEndpointService(String namespace,String name)  
参数1:命名空间名称;参数2:服务名称

1. 对外服务：部署在k8s集群中，对外提供服务，如tomcat、rest  
   一：创建：

①：先new一个服务对象：com.suneee.kubernetes.model.service.V1Service

V1Service service = new V1Service(namespace,name);

②：添加端口：service.addPort(portName,port, targetPort, nodePort)

port 是服务自身的端口，targetPort是暴露给k8s集群内其他服务的端口,nodePort是暴露给集群外部调用的端口。

p.s.:一个服务可添加多个端口，addPort有多个重载方法。

注意：service是需要为pod容器服务的，pod容器有多少个端口，就需要添加多少个端口，其中：port需要和pod容器内的对应，在我们现在的应用中，port和targetport一样即可，nodePort必须保证全局唯一。

③：添加deployment对应：service.setSelector("app",name)

name为deployment的name

④：创建：ServiceApi serviceApi = new ServiceApi();

serviceApi.createService(namespace,service)

二：删除：

serviceApi.deleteService(namespace,name);

三：获得service列表：

serviceApi.getServiceList()

四：获得service对象：

serviceApi.getServiceByName(String namespace,String name)

1. 对内服务：部署在k8s集群中，对内部提供服务，如provider  
   一：创建：（基本和对外服务一样）

②中添加端口：service.addPort(portName,port, targetPort, nodePort)无需nodePort

并且更改type：service.setType(ServiceType.ClusterIP)

1. Deployment：
2. 创建：

一：new Deployment对象:

com.suneee.kubernetes.model.deployment.AppsV1beta1Deployment deployment = new AppsV1beta1Deployment(namespace,name);

二：添加容器：

deployment.addContainer(name,imageName);imageName即为要发布的镜像名称。

deployment.addPort(port);为容器添加端口，可添加多个端口，对应产品模块暴露端口

deployment.addEnv(key,value);为容器添加环境变量，可添加多个

deployment.setResource(String limitsCpu,String limitsmem,String requestsCpu,String requestsmen);为容器设置资源，limitsCpu,limitsMem即为最多可用量，requestsCpu,requestmem为至少保证有多少资源:可以只设置上限setResource(String limitsCpu,String limitsmem)。cpu的数据格式为核数如“1”,“1.5”,“2”;men的数据格式为内存大小如“1Gi”,“500Mi”;

deployment.setReplicas(Integer);设置副本个数，即同时存在的pod数量，一般测试和预发布环境均为1，生产环境为2。

三：发布：

DeploymentApi deploymentApi = new DeploymentApi();

deploymentApi.createNamespaceDeployment(namespace,deployment);

1. 获取：

deployment = deploymentApi.getNamespaceDeploymentByName(namespace,name)

1. 更新：

更新要分为测试(只更新镜像)及其他情况：即镜像名是否改变(测试镜像tag为test不变，预发布及生产tag会产生相应版本号)

测试(只更新镜像)：AppApi appapi = new AppApi();

appapi.updateDeploymentNoChange(namespace,name)

其他情况：

如更新镜像名

①：先获取deployment对象，更新其镜像名：

deployment.updateImageName(imageName);

如更新端口：

deployment.addPort(port) or deployment.deletePort(port)

如更改pod副本数

deployment.setReplicas(replicas)

②：调用接口：

deploymentApi.updateNamespaceDeployment(namespace,name,deployment)

1. 删除：

deploymentApi.deleteNamespaceDeployment(namespace,name)

1. 获取deployment列表：

deploymentApi.getNamespaceDeploymentlist(namespace)

1. Pod：