kubectl管理KS对象

在K8S中，对象的管理呢主要是通过Kubectl来进行的。还是改不了老一套啊，没有什么创新精神。
依然是老一套通过CS客户端发送REST API请求到K8S的kube-apiserver.所以如果要知道K8S如何管理对象，
那么还是要先看kubectl。万变不离其宗。

命令操作方式 - Imperative command

K8S为用户制定了不少场景，比如当你很紧急的需要做某些事情的时候，那么
用户可以直接通过紧急命令来直接操作集群中的对象。
但是要注意的是：这种方法直接操作集群中的对象，操作的历史记录对下一次的维护来说是不可见的。

比如：

```
kubectl run nginx --image nginx
kubectl create deployment nginx --image nginx
```

优缺点

先说优点：
    命令简单，易学容易上手
    只需要一个命令一个步骤就可以对集群进行操作

缺点：
    命令执行的方式没有经过change review process
    命令方式没有办法提供此次操作变更的审计记录
    命令不提供记录源，除了活动的记录
    命令不提供用于创建新对象的模板

命令式配置操作对象 - Imperative object configuration

依然是通过语句去执行，但是与上面直接敲命令不同点在于：
    这种方式中需要运维提供一个包含操作对象的配置文件。

整个执行语句由以下几部分拼接而成，注意字里行间体会处理的产品的设计思想：
    operation - 由create, replace, delete等操作种类组成

option - kubectl提供多种选项
file name - 要操作的对象描述，要么是yaml，要么是Json

更多关于操作的API，比如有哪些操作项，有哪些资源类型(Workloads, Discovery&LB, Config, Cluster, Metadata)： https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.15/

kubectl replace -f nginx.yaml - 通过新的nginx.yaml中的内容覆盖老的对象
kubectl delete -f nginx.yaml -f redis.yaml - 删除定义在nginx.ymal和redis.yaml中的定义的所有的对象
kubectl create -f nginx.yaml - 根据nginx.yml中的定义来创建相应的对象

优缺点

优点：
紧急操作对象的配置文件可以版本化管理
在push提交紧急操作执行，或者在审计之前，可以先与其他流程集成，比如review审核操作是否可行
这种方式提供模板，每次创建新的对象时可以通过调用模板的方式来创建对象
缺点：
需要对对象的一些定义有一些最基本的了解，如果不了解那么你可能书写object的时候就会出错
执行命令之前，需要额外写yaml来配置对象，是额外的effort

与之前命令方式的比较

较之优点：
配置先行，较为简单和理解
随着K8S版本更迭，这种方式越来越成熟
较之缺点：
对于文件的操作，擅长，但是对于目录的操作，不是最佳应用场景
如果第一次通过配置方式来做更新，那么下一次必须也是通过配置方式进行，否则就丢了

声明式(陈述式)管理 - Declarative object configuration

首先呢，K8S是基于声明式来设计的。记得之前我们提到过，在K8S中，每个object都有两种属性内容：spec和status。所谓声明式呢，就是告诉K8S你最终想要的东西，而任何时刻某个对象都会有一个现实中的实际状态，这就是status.

K8S不断的扫描比较现实和理想中的差异，以此来为用户服务，到底是创建还是更新。

那么与之前命令式的区别在于：

```
    声明式不会有操作符，比如create, replace, delete, read等

但是配置文件该有的，还是要有的
另外一个区别是：
    命令式对于目录的操作不够好，而对于声明式操作，运维可以把所有要操作的描述内容，全部放在
    某个目录下，然后通过kubectl调用，让其自动扫描该目录，然后完成所有必须的操作。


kubectl diff -f configs/
kubectl apply -f configs/

循环更新目录以及其子目录，一直到最终没有子目录为止
kubectl diff -R -f configs/
kubectl apply -R -f configs/
```

优缺点

```
优点：
    直接对活动对象所做的更改将保留，即使它们没有合并回配置文件中。
    声明性对象配置更好地支持在目录上操作和自动检测每个对象的操作类型（创建、修补、删除）

缺点：
    当结果出乎意料时，声明性对象配置很难调试和理解结果。 - 因为变得复杂了，都是描述性的东西，你无法根据内容就直接看到最终结尾
    使用diff的部分更新创建复杂的合并和修补操作
```

有用链接

```
一定要看：
    大神详解 Kubernetes 配置管理最佳方法
        https://blog.csdn.net/qq_34463875/article/details/78390111
```

下一步分叉

```
K8S Tasks:
    https://kubernetes.io/docs/tasks/manage-kubernetes-objects/declarative-config/
```