containers:

kubectl explain pod.spec.containers 查看containers可选配置项

```
# 返回的重要属性
KIND:     Pod
VERSION:  v1
RESOURCE: containers <[]Object>   # 数组，代表可以有多个容器FIELDS:
  name  <string>     # 容器名称
  image <string>     # 容器需要的镜像地址
  imagePullPolicy  <string> # 镜像拉取策略
  command  <[]string> # 容器的启动命令列表，如不指定，使用打包时使用的启动命令
  args   <[]string> # 容器的启动命令需要的参数列表
  env    <[]Object> # 容器环境变量的配置
  ports  <[]Object>  # 容器需要暴露的端口号列表
  resources <Object> # 资源限制和资源请求的设置
```

imagePullPolicy：

用于设置镜像拉取的策略，kubernetes支持配置三种拉取策略：

- Always：总是从远程仓库拉取镜像（一直远程下载）。
- IfNotPresent：本地有则使用本地镜像，本地没有则从远程仓库拉取镜像（本地有就用本地，本地没有就使用远程下载）。
- Never：只使用本地镜像，从不去远程仓库拉取，本地没有就报错（一直使用本地，没有就报错）。

默认值说明：

- 如果镜像tag为具体的版本号，默认策略是IfNotPresent。
- 如果镜像tag为latest（最终版本），默认策略是Always。

command，args：

用于在Pod中的容器初始化完毕之后执行一个命令。

特别说明：command已经可以完成启动命令和传递参数的功能，为什么还要提供一个args选项，用于传递参数？其实和Docker有点关系，kubernetes中的command和args两个参数其实是为了实现覆盖Dockerfile中的ENTRYPOINT的功能：

- 如果command和args均没有写，那么用Dockerfile的配置。
- 如果command写了，但是args没有写，那么Dockerfile默认的配置会被忽略，执行注入的command。

- 如果command没有写，但是args写了，那么Dockerfile中配置的ENTRYPOINT命令会被执行，使用当前args的参数。
- 如果command和args都写了，那么Dockerfile中的配置会被忽略，执行command并追加上args参数。

env:
环境变量，用于在Pod中的容器设置环境变量。可以通过kubectl exec -it pod 进入查看
不推荐使用，推荐将这些配置单独存储在配置文件中。

ports:
访问Pod中的容器中的程序使用的是PodIp:containerPort。

kubectl explain pod.spec.containers.ports 查看

```
KIND:     Pod
VERSION:  v1
RESOURCE: ports <[]Object>
FIELDS:
  name <string> # 端口名称，如果指定，必须保证name在pod中是唯一的
  containerPort <integer> # 容器要监听的端口(0<x<65536)
  hostPort <integer> # 容器要在主机上公开的端口，如果设置，主机上只能运行容器的一个副本(一般省略）
  hostIP <string>  # 要将外部端口绑定到的主机IP(一般省略)
  protocol <string>  # 端口协议。必须是UDP、TCP或SCTP。默认为“TCP”
```

访问容器中的程序需要使用的是 podIp:containerPort

resources:
resources有两个子选项：limit、requests分别对应上限和下限

limit:
用于限制运行时容器的最大占用资源，当容器的资源超过limit时会被终止，并进行重启

requests:
用于设置容器需要的最小的资源，如果资源不够，则无法启动
参数：
- CPU:core数，可以为整数或者小数
- memory:内存大小，单位Gi，Mi,G，M等形式