# POD详解一

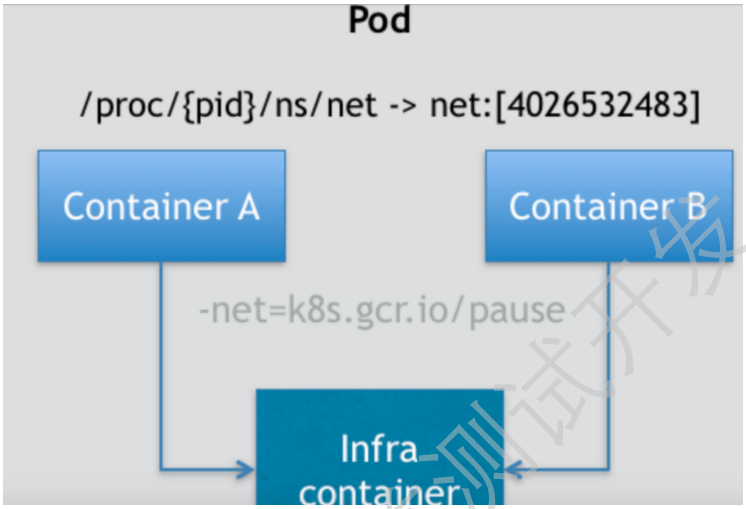
## 前言

有一些关于docker的前置知识需要了解一下，大家可以根据下面的链接阅读。尤其要明白什么是linux的名称空间以及docker的4种网络模式。 这里要注意的是linux的namespace和k8s的namespace是完全两个不同的东西。

docker网络原理
docker4种网络模式

## 什么是POD

首先，关于 Pod 最重要的一个事实是：它只是一个逻辑概念。也就是说，Kubernetes 真正处理的，还是宿主机操作系统上 Linux 容器的 Namespace 和 Cgroups，而并不存在一个所谓的 Pod 的边界或者隔离环境。
那么POD究竟是什么呢，它又是被怎么创造出来的呢？答案是：Pod，其实是一组共享了某些资源的容器。具体的说：Pod 里的所有容器，共享的是同一个Network Namespace，并且可以声明共享同一个Volume。那么POD究竟是如何实现的呢。
好记的我们之前的帖子中说明的docker的4中网络模式么。 其中有一种模式是container模式，它能够让很多容器共享一个网络名称空间，具体的原理是先使用briage模式启动第一个容器， 之后启动的其他容器纷纷使用container模式将网络环境绑定到这第一个容器上。这样这些容器的网络就连接到了一起，他们互相可以使用localhost这种方式进行网络通信。 如下图：




如上图所示，这个 Pod 里有两个用户容器 A 和 B，还有一个infra container， 它也叫做pause容器，也被称为sandbox，
意思是沙箱，这个沙箱为其他容器提供共享的网络和文件挂载资源。 这也是我们刚才说的在container模式中的第一个容器。而当这个容器被创建出来并hold住Network Namespace之后，其他由用户自己定义的容器就可以通过container模式加入到这个容器的Network Namespace中。这也就意味着，对于在一个POD中的容器A和容器B来说，他们拥有相同的IP地址，可以通过localhost进行互相通信。

## sidecar模式

那么网络共享了之后呢，要解决文件的共享就容易很多。有一个概念大家一定要记清楚就是。POD是k8s调度的最小逻辑单位，一个POD中的所有容器都必须被调度到同一台机器上， 所以在容器之间共享文件和目录就变成了一件很简单的事情。
只要在宿主机上开辟出一个空目录，然后把这个目录地址挂载到pod中的各个容器中就可以了。
而在k8s为POD做的声明式API中，专门有一个empty dir的volume类型。
就是专门在POD中开辟一个临时的可共享的空目录。这样在一个POD中的多个容器就可以通过这个目录进行文件的共享。
一个典型的例子就是日志收集。 在我们的产品之前的部署方式中，就是使用这种方式来做日志收集的。
在一个pod中启动一个服务容器，然后再启动一个filebeat容器，这两个容器通过这个 empty dir 共享了日志的目录。
服务容器把日志写到这个目录中， filebeat容器就可以同步的收集这些日志并发送到ES中了。而这种模式在k8s中有一个专门的术语叫做sidecar。sidecar 指的就是我们可以在一个 Pod 中，启动一个辅助容器，来完成一些独立于主进程（主容器）之外的工作。
同时我们有时候也会叫它伴生容器，因为往往它与主容器伴生伴死。 这种sidecar模式有一个缺点就是由于这种伴生伴死的特性来的。
如果辅助容器出现了以外，比如oom，或者用来做健康检查的探针探测失败。
在POD层面都会触发重启，那么在重启结束前会导致主容器的不可用。
所以这也是我们产品在一些关键的服务上抛弃了sidecar模式的原因。PS：关于探针的内容我之后会讲到。

```
apiVersion: v1
kind: Pod
metadata:
  name: javaweb-2
spec:
  initContainers:
  - image: geektime/sample:v2
    name: war
    command: ["cp", "/sample.war", "/app"]
    volumeMounts:
    - mountPath: /app
      name: app-volume
  containers:
  - image: geektime/tomcat:7.0
    name: tomcat
    command: ["sh","-c","/root/apache-tomcat-7.0.42-v2/bin/start.sh"]
    volumeMounts:
    - mountPath: /root/apache-tomcat-7.0.42-v2/webapps
      name: app-volume
    ports:
    - containerPort: 8080
      hostPort: 8001
  volumes:
  - name: app-volume
    emptyDir: {}
```



上面的图中，标记为红色框内的内容就是在一个POD中使用empty
dir这种volume类型来达到容器中的文件共享的目的。之后我们还会介绍更多的volume类型的。

## Pod的一些基本概念和操作

在K8S中一切皆资源，而每个资源都可以根据用户的需要来自定义一些meta信息。 而label就是其中一个，label是标签的意思，
我们可以根据自己的喜好给K8S中的任何资源打上我们需要的标签。 比如，Node(集群中的节点) 也是k8s中的资源，我们可以通过kubecl get node来查看当前集群中的节点信息。 而我们就可以给这些Node打上我们需要的label来供我们后续调度的使用。 比如我们可以使用kubecl label node env=test 这样一个命令kubectl label nodes = 来给node打标签。 其中labe是一个key：value的格式。 比如我们用 kubectl label nodes qa-test001 env=test 这个命令， 就是给集群中一个名字叫qa-test001的节点上打上env=test的标签。
那么这个标签有什么用呢， 我们看下面一个yaml的定义。

```
apiVersion: v1
kind: Pod
...
spec:
nodeSelector:
env: test
```

上面我们隐去了不必要的细节只留下了spec字段中关于nodeSelector的定义。
这个nodeSelector的意思就是在部署pod的时候选择node中有env=test标签的节点进行部署。这就是k8s中根据label进行调度的机制。

k8s中任何资源都可以拥有label，
而在k8s中大部分资源对象在调度的时候都会有各种各样基于label的selector用来或调度，或匹配不同的资源对象。
在我们这个例子中，pod就是使用这种机器来选择部署到哪个节点上。
比如我们有些特别消耗IO的服务(ES,ETCD,Mysql)需要部署在有ssd硬盘的节点上。那我们就可以在有ssd硬盘的节点打上相应的label。

接下来我们看看下面的定义：

```
apiVersion: v1
kind: Pod
metadata:
name: test
labels:
run: test
spec:
nodeSelector:
env: test
containers:
- name: selenium-node-chrome
image: registry.4paradigm.com/chrome_debug
imagePullPolicy: IfNotPresent
ports:
- containerPort: 5900
env:
- name: HUB_PORT_4444_TCP_ADDR
value: "selenium-hub"
- name: HUB_PORT_4444_TCP_PORT
value: "4444"
- name: NODE_MAX_INSTANCES
value: "30"
- name: NODE_MAX_SESSION
value: "30"
- name: NODE_REGISTER_CYCLE
value: "5000"
- name: DBUS_SESSION_BUS_ADDRESS
value: "/dev/null"
```

下面我们来看看这里面的定义，首先我们在contianers里面定义在POD中都启动哪些容器，这里我们就不使用sidecar模式了，就只启动一个容器看看。

- image: 来指定容器使用的镜像。
- imagePullPolicy 则定义了我们更新镜像的策略。
  这里一共我们有3个选择，一个是always(不管什么情况只要启动容器就尝试拉取镜像，如果节点中有镜像会覆盖)，一个是Nerver(不论如何都不拉取镜像，如果节点上没有镜像会抛出错误)以及我们这里使用的IfNotPresent(如果节点中存在此镜像就不再拉取，如果不存在就拉取)。
- ports：定义容器对外暴露的端口号
- env：定义为容器指定的环境变量

以上是一个POD最基本的一些字段， 我们可以通过上面的定义使用kubectl create -f 的方式把这个pod部署起来了。