理解Pod

Pod是kubernetes中你可以创建和部署的最小也是最简的单位。一个Pod代表着集群中运行的一个进程。
Pod中封装着应用的容器（有的情况下是好几个容器），存储、独立的网络IP，管理容器如何运行的策略选项。Pod代表着部署的一个单位：kubernetes中应用的一个实例，可能由一个或者多个容器组合在一起共享资源。

在Kubrenetes集群中Pod有如下两种使用方式：

- **一个Pod中运行一个容器。**“每个Pod中一个容器”的模式是最常见的用法；在这种使用方式中，你可以把Pod想象成是单个容器的封装，kuberentes管理的是Pod而不是直接管理容器。
- **在一个Pod中同时运行多个容器。**一个Pod中也可以同时封装几个需要紧密耦合互相协作的容器，它们之间共享资源。这些在同一个Pod中的容器可以互相协作成为一个service单位——一个容器共享文件，另一个“sidecar”容器来更新这些文件。Pod将这些容器的存储资源作为一个实体来管理。

网络

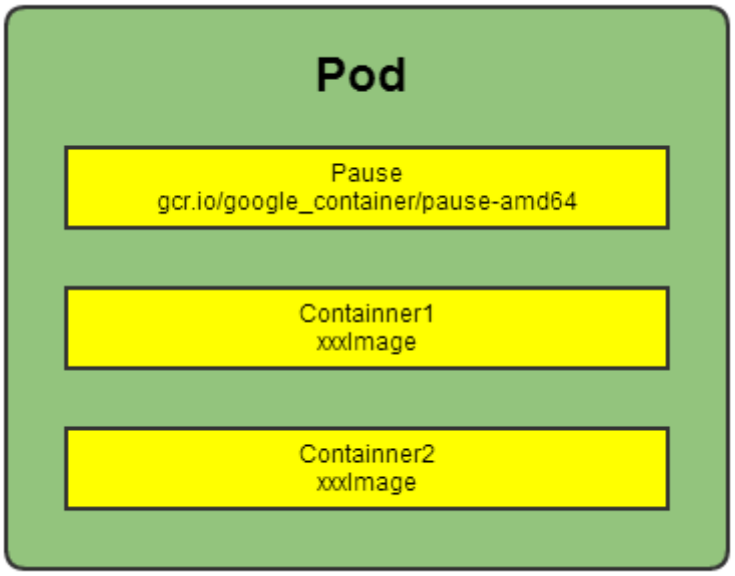
kubernetes为每个pod都分配了一个唯一的IP地址，称之为pod ip，一个pod里的多个容器共享pod ip地址。kubernetes要求底层网络支持集群内任意两个pod之间的tcp/ip直接通信，通常采用虚拟二层网络技术来实现，例如：flannel、open vswitch等，需要记住的一点：在kubernetes里，一个pod里的容器与另外主机上的pod容器能够直接通信。

存储

可以Pod指定多个共享的Volume。Pod中的所有容器都可以访问共享的volume。Volume也可以用来持久化Pod中的存储资源，以防容器重启后文件丢失。

Pause容器

Pause容器，又叫Infra容器，下图看到每个pod都有一个特殊的被称为"根容器"的pause容器。pause容器对应的镜像属于kubernetes平台的一部分，除了pause容器，每个pod还包含一个或多个紧密相关的用户业务容器



kubernetes中的pause容器主要为每个业务容器提供以下功能：

- 在pod中担任Linux命名空间共享的基础；
- 启用pid命名空间，开启init进程。

```
[root@vlnx251105 ~]# docker run -d --name pause -p 8080:80 k8s.gcr.io/pause
```

```
[root@vlnx251105 ~]# cat <<EOF > nginx.conf
error_log stderr;
```

```
events { worker_connections  1024; }
http {
    access_log /dev/stdout combined;
    server {
        listen       80;
        server_name  localhost;
        location / {
            proxy_pass http://127.0.0.1:8000;
        }
    }
}
EOF

[root@vlnx251105 ~]# docker run -d --name nginx-proxy -v
`pwd`/nginx.conf:/etc/nginx/nginx.conf --
net=container:pause --ipc=container:pause --
pid=container:pause nginx

[root@vlnx251105 ~]# cat <<EOF > nginx2.conf
error_log stderr;
events { worker_connections  1024; }
http {
    access_log /dev/stdout combined;
    server {
        listen       8000;
        server_name  localhost;
        root         /tmp;
        location / {
        }
    }
}
```

```
EOF
```

```
[root@vlnx251105 ~]# docker run -d --name nginx -v `pwd`/nginx2.conf:/etc/nginx/nginx.conf -v `pwd`/index.html:/tmp/index.html --net=container:pause --ipc=container:pause --pid=container:pause nginx
```

```
[root@vlnx251105 ~]# curl 127.0.0.1:8080
pause test
```

kubernetes里的所有资源对象都可以采用yaml或者json格式的文件来定义或描述

```
apiVersion: v1
kind: Pod
metadata:
  name: myweb
  labels:
    name: myweb
spec:
  containers:
  - name: myweb
    image: kubeguide/tomcat-app:v1
    ports:
    - containerPort: 8080
    env:
    - name: MYSQL_SERVICE_HOST
```

```
      value: 'mysql'
    - name: MYSQL_SERVICE_PORT
      value: '3306'
```

kind为pod表明这是个pod的定义，metadata里的name属性为pod的名字，metadata里还能定义资源对象的 label，这里声明myweb拥有一个name=myweb的 label。pod里所包含的容器组的定义则在spec中声明，这里定义了一个名字为myweb、对应镜像为kubeguide/tomcat-app:v1 的容器， 该容器注入了名为 MYSQL_SERVICE_HOST='mysql'和MYSQL_SERVICE_PORT='3306'的环境变量（env），并且在8080 containerPort 上启动容器进程。pod的ip加上 containerPort，就组成了一个新的概念--endpoint，它代表此pod里的一个服务进程的对外通信地址。一个pod也存在着具有多个endpoint的情况，比如当把tomcat定义为一个pod时，可以对外暴露管理端口和服务端口这两个endpoint。
我们所熟悉的docker volume在kubernetes里也有对应的概念--pod volume，后者有一些扩展，比如可以用分布式文件系统glusterfs实现后端存储功能；pod volume是定义在pod之上，然后被各个容器挂载到自己的文件系统中的。
kubernetes的event是一个事件的记录，记录了事件的最早产生时间、最后重现时间、重复次数、发起者、类型、以及导致此时间的原因等众多信息。event通常会关联到某个具体的资源对象上，是排查故障的重要参考信息，node的描述信息包括了event，而pod同样有event记录，当我们发现某个pod迟迟无法创建时，可以用 kubectl decribe pod <port name> 来查看他的描述信息，用来定位问题的原因。

资源限制

每个pod都可以对其能使用的服务器上的计算资源设置限额，当前可以设置限额的计算资源有CPU与memory两种，其中cpu资源单位为 cpu core的数量，是一个绝对值而非相对值。

一个cpu的配额对于绝大多数容器来说是相当大的一个资源配额，所以，在kubernetes里，通常以千分之一的cpu配额为最小单位，用m来表示。通常一个容器的cpu配额被定义为100～300m，即占用0.1～0.3个cpu。由于cpu配额是一个绝对值，所以无论在拥有一个core的机器上，还是拥有48个core的机器上，100m这个配额所代表的cpu的使用量都是一样的。与cpu配额类似，memory配额也是一个绝对值，它的单位是内存字节数。

在kubernetes里，一个计算资源进行配额限定需要设定以下两个参数

- requests：该资源的最小申请量，系统必须满足要求，
- limits：该资源最大允许使用的量，不能被突破，当容器试图使用超过这个量的资源时，可能会被kubernetes kill并重启

通常requests会设置为一个较小的数值，符合容器平时的工作负载情况下的资源需求，而把limits设置为峰值负载情况下资源占用的最大量。比如下面定义，mysql容器申请最少0.25个cpu及64MiB内存，在运行过程中mysql容器所能使用的资源配额为 0.5 个cpu 以及 128MiB内存。

```
spec:
  containers:
  - name: db
    image: mysql
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "128Mi"
        cpu: "500m"
```