

给Kubernetes集群下的容器配置内核参数

在Kubernetes集群下运行的容器的内核参数是默认的，但是对于某型类型的应用如Nginx Ingress controller而言，默认的内核参数配置是不够的，需要做出调整，例如somaxconn是限制了接收新 TCP 连接侦听队列的大小，它的默认值是128，但是对于反向代理的服务器而言，这个配置实在是太小了。那么我们自然想到需要去调整这个应用的容器的内核配置参数。

解决之道

Docker Daemon的处理方式

对于Docker引擎而言，可是使用`--sysctl` 运行参数来设定需要更改的内核参数，例如：

```
docker run -it --sysctl net.core.somaxconn=65535 busybox
```

#在容器看看是否配置成功：

```
cat /proc/sys/net/core/somaxconn
```

然后我可以看看容器的详情：



可以看到，Docker引擎对容器进行了相关的配置，而无需使用特权模式来设置内核参数

Kubernetes的处理之道

在Kubernetes里，Kubernetes可以利用Docker引擎的这个`--sysctl`的能力，也可以利用`privilege init container`的方式。目前`sysctls`的能力还依然在alpha阶段，选择时需要注意。

Kubernetes Sysctls

具体可以参考: [Using Sysctls in a Kubernetes Cluster](#)

实践过程如下:

- 需要在kubelet启动参数上配置对应的开关 `--experimental-allowed-unsafe-sysctls`。例如在阿里云的kubernetes服务, 可以在node节点, 修改 `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf`, 增加对应的配置并重新加载新的配置 `systemctl daemon-reload`, 然后重启kubelet。如下是允许配置和net相关的内核参数:

```
[Install]
WantedBy=multi-user.target
[root@iZwz9ik7jd1mbaf6fclqZ ~]# cat /etc/systemd/system/kubelet.service
kubelet.service kubelet.service.d/
[root@iZwz9ik7jd1mbaf6fclqZ ~]# cat /etc/systemd/system/kubelet.service.d/10-kubeadm.conf
[Service]
Environment="KUBELET_KUBECONFIG_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/kubelet.conf"
Environment="KUBELET_SYSTEM_POOLS_ARGS=--pod-manifest-path=/etc/kubernetes/manifests --allow-privileged=true"
Environment="KUBELET_NETWORK_ARGS=--network-plugin=cni --cni-conf-dir=/etc/cni/net.d --cni-bin-dir=/opt/cni/bin"
Environment="KUBELET_DNS_ARGS=--cluster-dns=172.19.0.10 --pod-infra-container-image=registry.vpc.cn-shenzhen.aliyuncs.com/acs/pause-amd64:3.0"
enable-controller-attach-detach=false --cluster-domain=cluster.local --cloud-provider=external --hostname-override=cn-shenzhen.i-zwz9ik7jd1mbaf6fclqZ
Environment="KUBELET_AUTHZ_ARGS=--authorization-mode=Webhook --client-ca-file=/etc/kubernetes/pki/ca.crt"
Environment="KUBELET_CADVISOR_ARGS=--cadvisor-port=0"
Environment="KUBELET_CGROUP_ARGS=--cgroup-driver=systemd"
Environment="KUBELET_CERTIFICATE_ARGS=--rotate-certificates=true --cert-dir=/var/lib/kubelet/pki"
Environment="KUBELET_UNSAFE_SYSCTLS=--experimental-allowed-unsafe-sysctls='net.*'"
ExecStart=
ExecStart=/usr/bin/kubelet $KUBELET_KUBECONFIG_ARGS $KUBELET_SYSTEM_POOLS_ARGS $KUBELET_NETWORK_ARGS $KUBELET_DNS_ARGS $KUBELET_AUTHZ_ARGS $KUBELET_CADVISOR_ARGS
$KUBELET_CGROUP_ARGS $KUBELET_CERTIFICATE_ARGS $KUBELET_EXTRA_ARGS $KUBELET_UNSAFE_SYSCTLS
```

Kubernetes允许配置的内核参数如下:

```
kernel.shm*,
kernel.msg*,
kernel.sem,
fs.mqueue.*,
net.*.
```

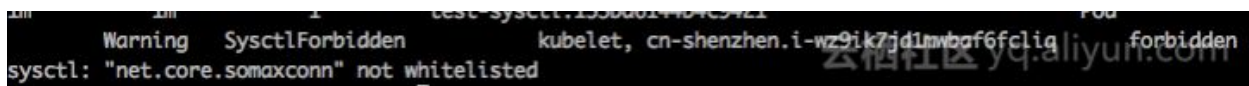
- 启动Pod的时候设置对应的annotation, 申明需要修改的内核参数, 以启动一个nginx为测试例子

```
apiVersion: v1
kind: Pod
metadata:
  name: test-sysctl
  annotations:
    security.alpha.kubernetes.io/unsafe-sysctls:
      net.core.somaxconn=65535
spec:
```

```
containers:
- image: nginx
  name: nginx
  ports:
  - containerPort: 80
    protocol: TCP
nodeSelector:
  kubernetes.io/hostname: cn-shenzhen.i-xxxxx
```

注意：对于需要变更内核参数的应用，建议部署到特定的机器上。为了方便，我是使用了node selector。

如果对应的机器的kubelet没有打开这个对应的开关，那么pod是部署不成功的。我们可以通过`kubectl get event`来看看对应的日志：



```
Warning SysctlForbidden kubelet, cn-shenzhen.i-wz9ik7jd1mwbaf6fclig forbidden
sysctl: "net.core.somaxconn" not whitelisted
```

那么这个方式背后的原理是什么呢？其实就是docker的`--sysctl`，我们可以登录到pod对应的node节点去`docker inspect`看看。但是奇怪的是，`docker inspect`看不到任何`sysctl`的迹象，是不是那里有误？

其实不是的，因为真正执行`sysctl`是kubernetes的`pause container`也叫做`infra container`，我们找到对应的这个容器，再`docker inspect`看看：



```
"UsernsMode": "",
"ShmSize": 67108864,
"Sysctls": {
  "net.core.somaxconn": "65535"
},
"Runtime": "runc",
"ConsoleSize": [
  0,
  0
]
```

这个时候，我们发现，这个pause容器配置了sysctls。

Kubernetes Init Container

Init container的用法可以参考：[Init Containers](#)

使用init container的好处是，无需去改变kubelet的配置，但是需要给这个init container配置成`privilege`的权限。

以下是一个启动Pod的例子：

```
apiVersion: v1
kind: Pod
metadata:
  name: test-sysctl-init
  namespace: default
spec:
  containers:
    - image: nginx
      imagePullPolicy: Always
      name: nginx
      ports:
        - containerPort: 80
          protocol: TCP
      initContainers:
        - image: busybox
          command:
            - sh
            - -c
            - echo 10000 > /proc/sys/net/core/somaxconn
          imagePullPolicy: Always
          name: setsysctl
          securityContext:
            privileged: true
```

至于选择那种方式，可以自行选择。不过后续需要关注kubernetes关于sysctls的演进。