此文章已于 13:29:48 2017/12/8 重新发布到 u013042928的博客

如何在k8s集群中安装nvidia、cuda并使用GPU进行训练

**如何在k8s集群中安装nvidia、cuda并使用GPU进行训练**

在写具体步骤前，发表一下自己的感想体会，因为在这过程中，踩过很多坑，对于像我们这种小白，踩坑真的是无处不在，真的很感谢网上一些大神细心地帮助，也感谢其它大神分享的博客。所以自己实现之后也想把这过程中遇到的坑和解决的方法总结总结，希望对大家有用。

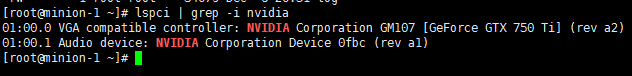
本文主要是针对已经部署好了k8s集群的，基于centos系统，截止k8s 1.8版本，对GPU的使用支持还是停留在实验阶段，只支持nvidia的GPU，在k8s上使用GPU，先要装好几样东西。

安装流程：

**1、关闭UEFI。**这个很重要，因为如果不关闭可能会导致安装nvidia驱动的时候无法加载kernel模块，我就在这浪费了好多时间。具体做法是开机按F2进入BIOS设置，通常在BOOT或Secure选项卡上，把安全启动[Secure BOOT]，改为[Disable]，如果不是的话自己去搜搜相应的。

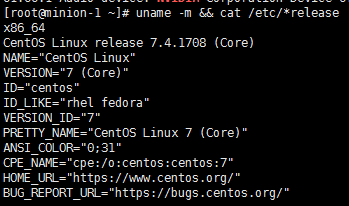
### *2、确认自己主机有nvidia的显卡。*

输入命令lspci | grep -i nvidia 查看nvidia显卡情况。



### 3、检查系统版本，确保系统支持(需要Linux-64bit系统)

#uname -m && cat /etc/\*release



### 4、根据nvidia显卡的信息和系统下载相应的安装包。

[NVIDIA-Linux-x86\_64-381.22.run](http://www.nvidia.cn/Download/index.aspx?lang=cn) # 最新显卡驱动

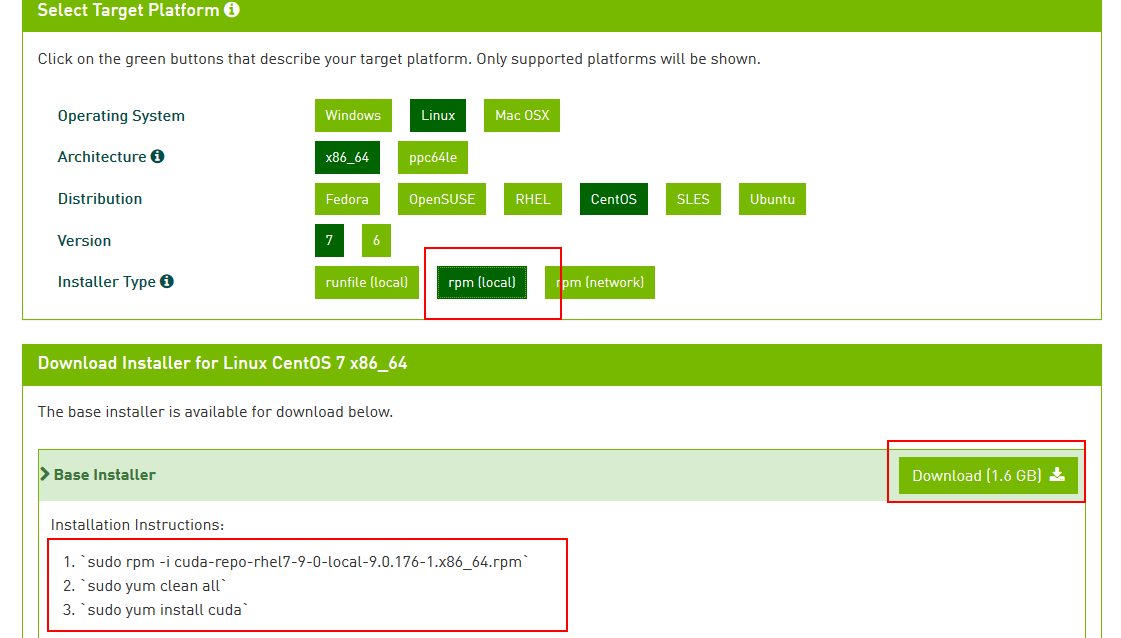
[cuda\_9.0.61\_375.26\_linux.run](https://developer.nvidia.com/cuda-downloads) # 最新CUDA安装包

[cudnn-8.0-linux-x64-v6.0.tgz](https://developer.nvidia.com/rdp/cudnn-download) # cudnn库v6.0

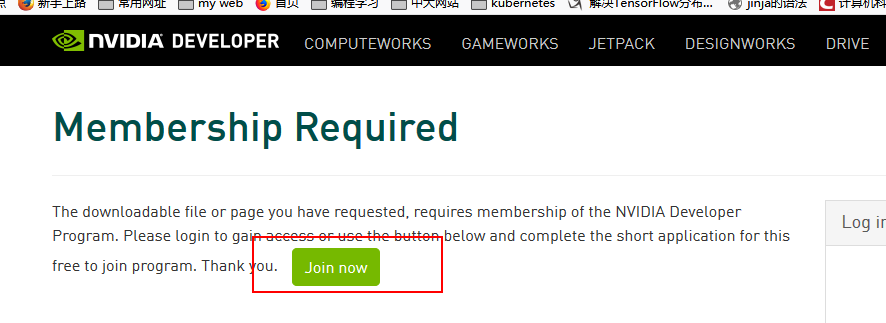
（1）登录NVIDIA官网<http://www.geforce.cn/drivers>设置驱动检索条件（注意尽量设置语言英文）：



（2）官网下载cuda-rpm包<https://developer.nvidia.com/cuda-downloads>，一定要对应自己的版本。



（3）下载cudnn需要注册<https://developer.nvidia.com/rdp/cudnn-download>，注意有些浏览器是注册不了的，我就踩了这个坑。



### 5、安装gcc gcc-c++

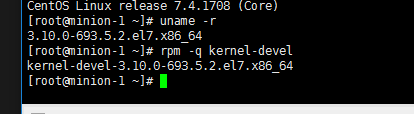
#yum install gcc gcc-c++

### 6、查看内核是否一致。

#uname -r

#rpm -q kernel-devel

#rpm -q kernel-headers



如果一致的话这步可以跳过，输入rpm -q kernel-devel时有可能是not found，说明没有安装，有显示说明已安装。

安装kernel-devel和kernel-headers。

#sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r)

或者

#yum install kernel-devel

#yum install kernel-headers

如果安装完发现不一致的话需要 yum -y update

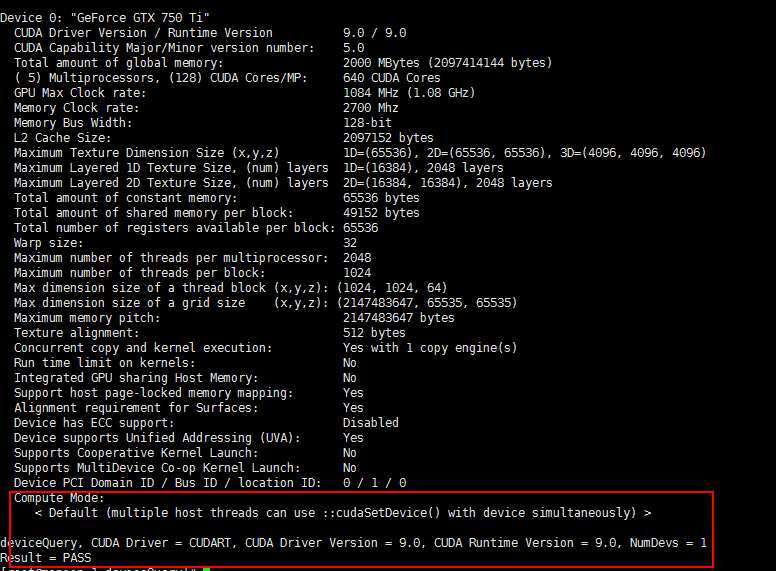
**7、安装cuda**(最好是先安装好这个再去安装驱动，不然会nvidia的安装不会很顺利，我就踩了这个坑。)

#sudo rpm -i cuda-repo-rhel7-8-0-local-ga2-8.0.61-1.x86\_64.rpm

#sudo yum clean all

#sudo yum install cuda

可能会报错，



（1）原因是缺少2个包，装第一个：

#sudo vim /etc/yum.repos.d/linuxtech.testing.repo

输入：

[linuxtech-testing]

name=LinuxTECH Testing

baseurl=http://pkgrepo.linuxtech.net/el6/testing/

enabled=0

gpgcheck=1

gpgkey=http://pkgrepo.linuxtech.net/el6/release/RPM-GPG-KEY-LinuxTECH.NET

退出。

#sudo yum --enablerepo=linuxtech-testing install libvdpau

（2）第二个，尤其是dkms，安装驱动的时候会用到这个模块。

#yum -y install epel-release

#yum -y install --enablerepo=epel dkms

（3）好了，再次执行安装cuda的步骤。

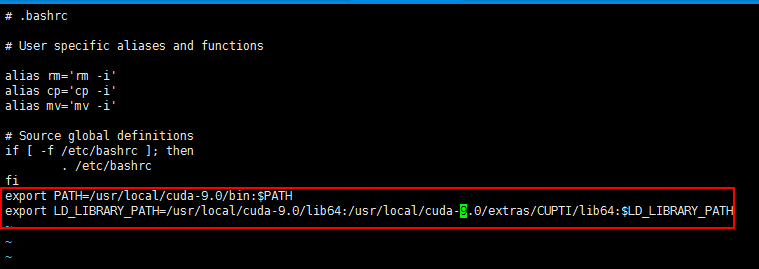
（4）验证安装结果。

添加环境变量，在 ~/.bashrc的最后面添加下面两行。

#vi ~/.bashrc

export PATH=/usr/local/cuda-8.0/bin:$PATH

export LD\_LIBRARY\_PATH=/usr/local/cuda-8.0/lib64:/usr/local/cuda-8.0/extras/CUPTI/lib64:$LD\_LIBRARY\_PATH

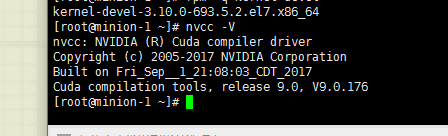


使生效

# source ~/.bashrc

验证安装结果

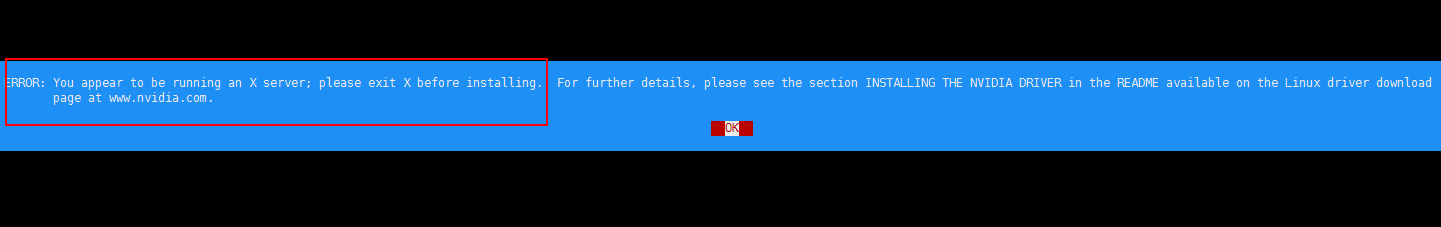
#nvcc -V



### 8、安装显卡驱动

安装NVIDIA驱动是很重要的步骤，该步成功了，后面也就基本上一马平川了。

(1)、关闭X server，不然在安装过程中会出现以下错误。



#sudo init 3（建议用这个）

或者

#systemctl stop gdm.service

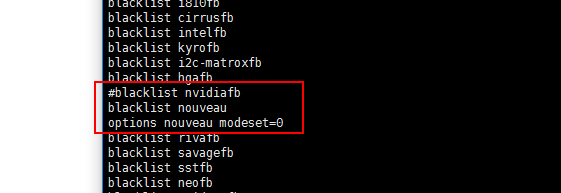
（2）禁用nouveau（因为它是一般linxu系统自带的显卡驱动，会和nvidia冲突，所以必须要关掉）

使用su命令切换到root用户下: su root

#vi /lib/modprobe.d/dist-blacklist.conf

将nvidiafb注释掉。  
#blacklist nvidiafb

然后添加以下语句：  
blacklist nouveau  
options nouveau modeset=0



（3）重建initramfs image步骤

备份 initramfs 文件

# sudo mv /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r).img.bak

重建 initramfs 文件

# sudo dracut -v /boot/initramfs-$(uname -r).img $(uname -r)

（4）修改运行级别为文本模式

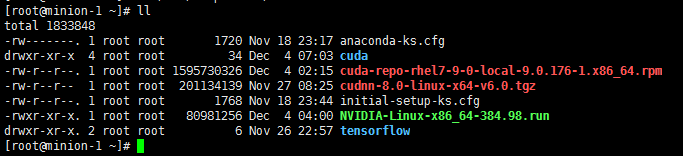
systemctl set-default multi-user.target

（5）关机重启#reboot

（6）查看nouveau是否已经禁用，没有输出就表示禁用了。

# lsmod | grep nouveau

（7）开始安装nvidia驱动（在驱动的目录下）



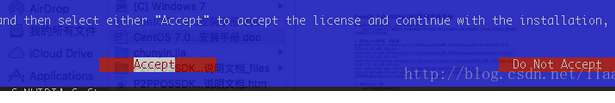
#chmod +x NVIDIA-Linux-x86\_64-384.98.run

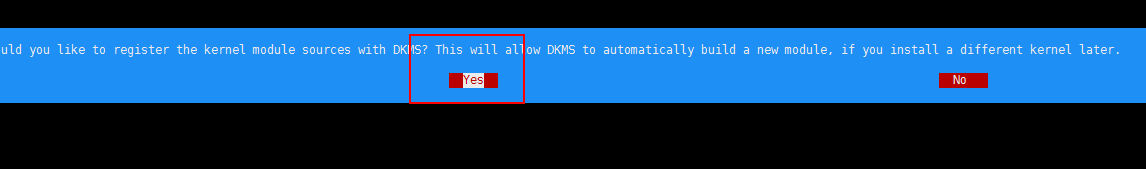
#sh NVIDIA-Linux-x86\_64-384.98.run

开始安装

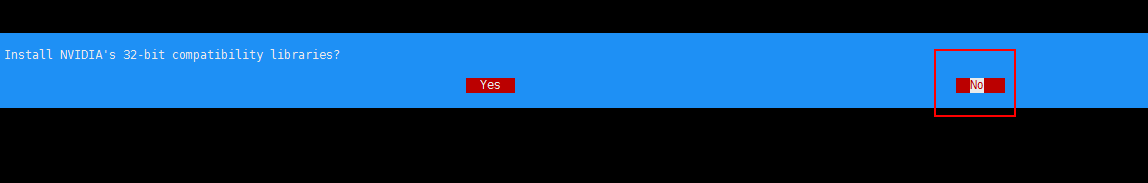


**选择Accept**

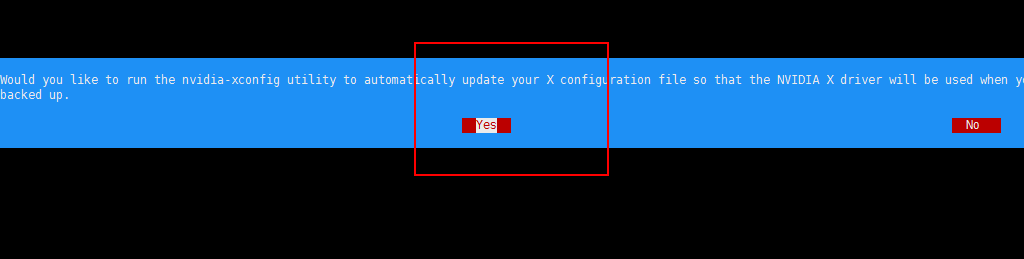




32bit兼容包选择, 这里要注意选择**NO**,不然后面就会出错。

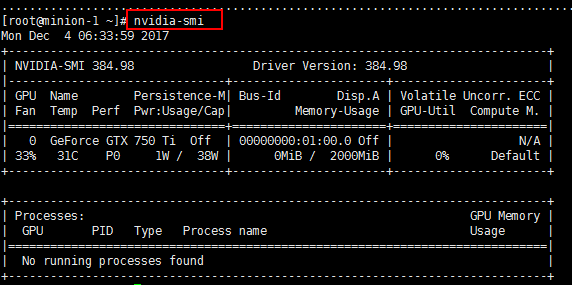


X-configurtion的选择页面**YES**



后面的都选择默认即可。

安装完之后输入nvidia-smi，能显示说明安装正确。



### 9、安装cuDNN

# tar -xvzf cudnn-8.0-linux-x64-v6.0.tgz

# cp -P cuda/include/cudnn.h /usr/local/cuda-9.0/include

# cp -P cuda/lib64/libcudnn\* /usr/local/cuda-9.0/lib64

# chmod a+r /usr/local/cuda-9.0/include/cudnn.h /usr/local/cuda-9.0/lib64/libcudnn\*

### 10、安装nvidia-docker and nvidia-docker-plugin

#wget -P /tmp https://github.com/NVIDIA/nvidia-docker/releases/download/v1.0.1/nvidia-docker-1.0.1-1.x86\_64.rpm

#sudo rpm -i /tmp/nvidia-docker\*.rpm && rm /tmp/nvidia-docker\*.rpm

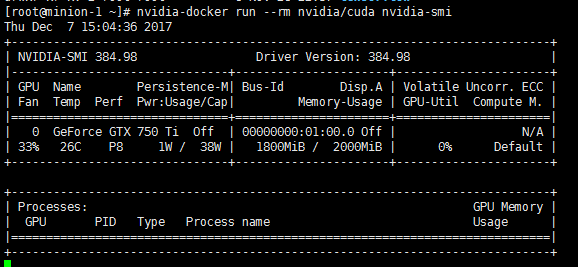
启动nvidia-docker

#sudo systemctl start nvidia-docker

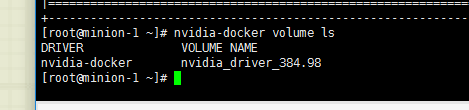
为了确认 nvidia-docker 是否安装成功，运行

#nvidia-docker run --rm nvidia/cuda nvidia-smi

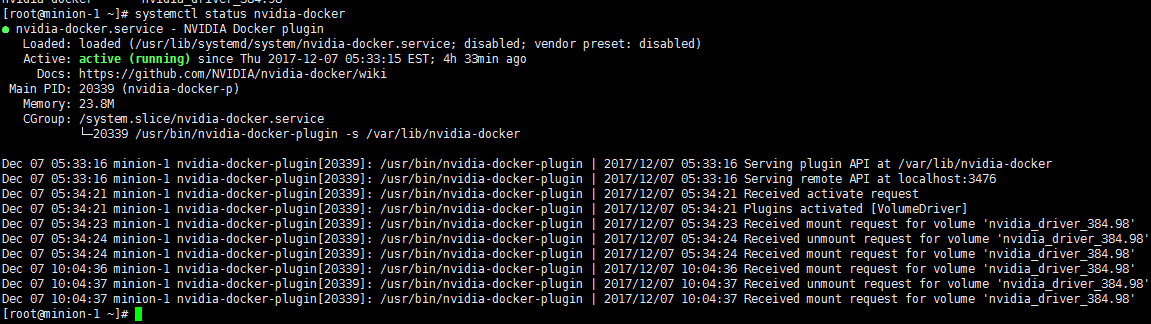
如果出现下面这些，说明正确（这过程可能有些久，因为要pull镜像）

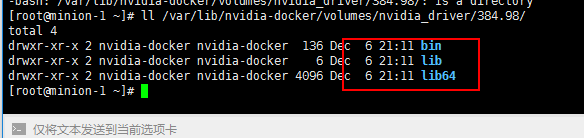


另外通过使用一些命令也可以查到相应的信息。比如：  
#nvidia-docker volume ls，查看卷



#systemctl status nvidia-docker，查看nvidia-docker运行状态，途中绿色的点说明在运行。记得一定要启动nvidia-docker。



#ll /var/lib/nvidia-docker/volumes/nvidia\_driver/384.98/,查看安装了nvidia-docker后映射的nvidia驱动和cuda库（注意后面的384.98文件夹因版本可能不一，要记住这个路径，因为用yaml文件创建pod的时候需要挂载这个文件夹）

11、至此，nvidia的驱动和cuda以及cudnn和nvidia-docker已经安装完成。可以用nvidia-docker run来跑个容器来试试看看。

#nvidia-docker run -itd -p 8888:8888 --name ten tensorflow/tensorflow:0.11.0rc0-gpu

等容器创建完后在浏览器可以用8888端口访问，可以访问jupyter，然后输入如下信息，如果出现GPU，也进一步证明nvidia等是安装正确的。

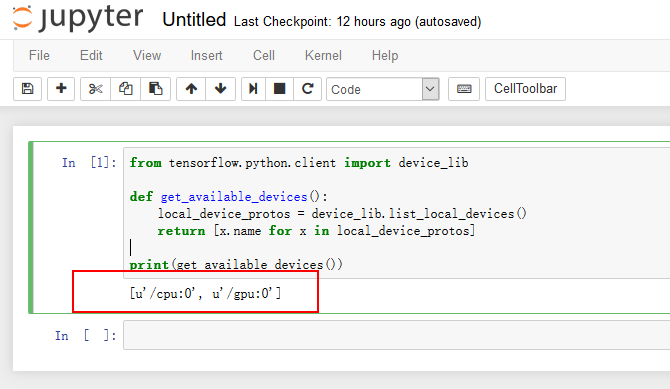
from tensorflow.python.client import device\_lib

def get\_available\_devices():

local\_device\_protos = device\_lib.list\_local\_devices()

return [x.name for x in local\_device\_protos]

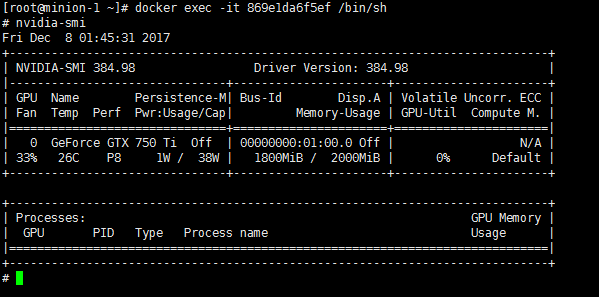
print(get\_available\_devices())



或者使用docker进入容器，输入nvidia-smi能输出一下信息，说明nvidia-docker等是安装成功。

# docker exec -it 869e1da6f5ef(容器ID) /bin/sh

#nvidia-smi



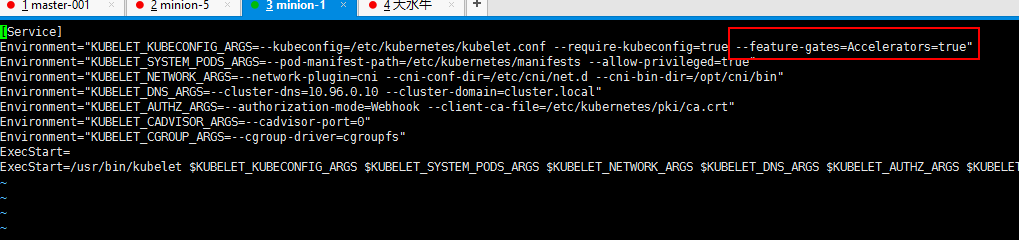
12、在kube-apiserver, kube-controller-manager, kube-scheduler和kubelet上开启--feature-gates="Accelerators=true"。

这个很重要，我之前就是因为没有开启这些，所以一直在困惑中度过，幸好有涛哥大神帮我一步步排查，最后才发现是没有加入启动参数。记得，kubelet启动参数一定要在所有的机子上加入，包括master和node，其它几个好像只有master上有。

(1)、在master上和node上的kubelet加入启动参数

#vi /etc/systemd/system/kubelet.service.d/10-kubeadm.conf

添加：--feature-gates=Accelerators=true



执行如下命令使新增参数生效

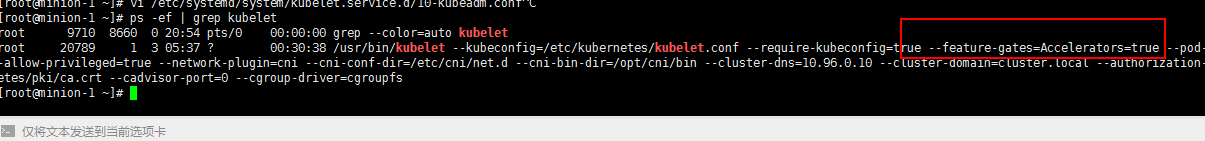
# systemctl stop kubelet

# systemctl daemon-reload

# systemctl start kubelet

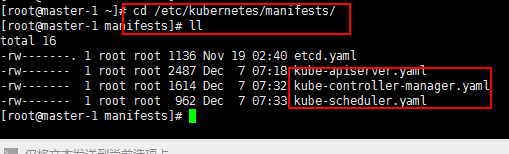
检查新增参数是否已经生效

# ps -ef | grep kubelet



（2）在master的kube-apiserver，kube-controller-manager, kube-scheduler 上参入启动参数。

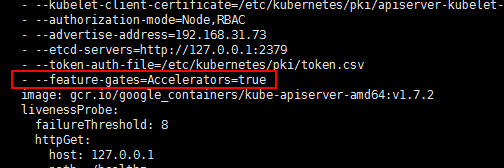
我的是在/etc/kubernetes/manifests/目录下，



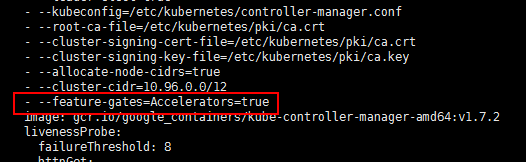
#cd /etc/kubernetes/manifests/

# vi kube-apiserver.yaml

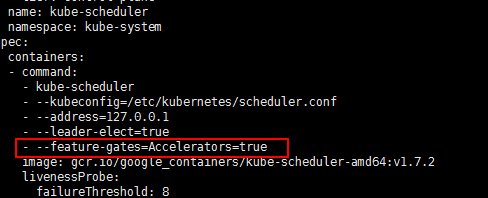
添加- --feature-gates=Accelerators=true



#vi kube-controller-manager.yaml



#vi kube-scheduler.yaml



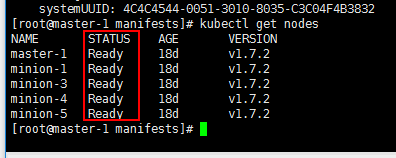
（3）重新启动docker和kubelet

#service docker restart

#systemctl restart kubelet

（4）检查集群是否已经重启和连上。

# kubectl get nodes



如果是NotReady说明（1）和（2）配置那里出错了，重新回去看看。

13、在master上查看node信息，看k8s是否能识别node的gpu.

#kubectl get node minion-1 -o yaml

或者

#kubectl describe node minion-1



如果未出现GPU或者GPU后面的数字是0，那就说明安装或者配置没有成功，回过头看看哪一步漏了。

至此，你的k8s集群能够使用GPU了。

14、用yaml创建pod来测试看看。

这里要说一说，创建yaml文件的时候kind类型可以直接用pod，也可使用Deployment，但使用ReplicationController的时候虽然也能创建容器，但却无法使用GPU，到现在我也不知道为什么。

#vi jupyter.yaml

---

apiVersion: extensions/v1beta1

kind: Deployment

metadata:

name: jupyter

spec:

replicas: 1

template:

metadata:

labels:

k8s-app: jupyter

spec:

containers:

- name: jupyter

image: tensorflow/tensorflow:0.11.0rc0-gpu

ports:

- containerPort: 8080

- containerPort: 8888

resources:

limits:

alpha.kubernetes.io/nvidia-gpu: 1

volumeMounts:

- mountPath: /usr/local/nvidia

name: nvidia

volumes:

- hostPath:

path: /var/lib/nvidia-docker/volumes/nvidia\_driver/384.98

name: nvidia

---

apiVersion: v1

kind: Service

metadata:

name: jupyter-svc

spec:

type: NodePort

ports:

- port: 8888

nodePort: 31001

name: jupyter

- port: 8080

nodePort: 31002

name: ssh

selector:

k8s-app: jupyter

Deployment配置：

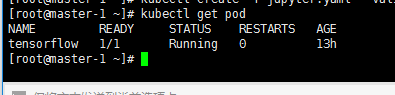
（1）alpha.kubernetes.io/nvidia-gpu 指定调用nvidia gpu的数量

（2）为了能让GPU容器运行起来，需要将Nvidia驱动和CUDA库文件指定到容器中。这里需要使用hostPath，您只需要将hostPath指定到/var/lib/nvidia-docker/volumes/nvidia\_driver/384.98即可，并不需要指定多个bin和lib目录。

创建pod的

# kubectl create -f jupyter.yaml --validate=false

#kubectl get pod



在浏览器输入31001端口访问jupyter

输入以下内容看是否能使用GPU

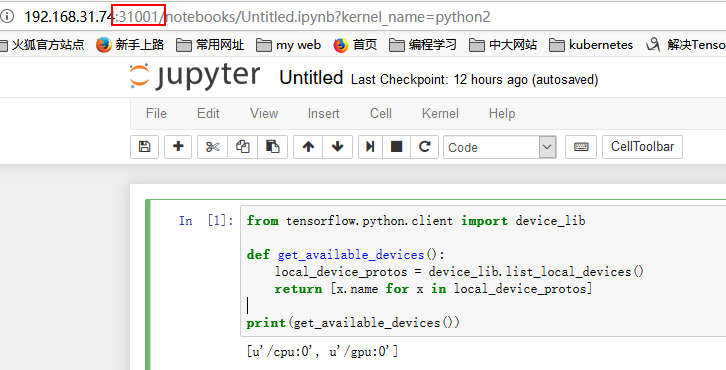
from tensorflow.python.client import device\_lib

def get\_available\_devices():

local\_device\_protos = device\_lib.list\_local\_devices()

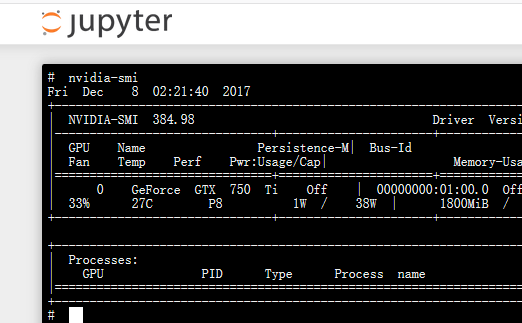
return [x.name for x in local\_device\_protos]

print(get\_available\_devices())



或者开启一个terminal

输入nvidia-smi



14、到这里，已经成功配置使用GPU了。

15、说一说在这过程中踩过的坑以及解决方法。

先奉上一些可能用到命令

（1）卸载NVIDIA驱动，前提是安装包还在哈。

#sh NVIDIA-Linux-x86\_64-384.98.run --uninstall

（2）卸载nvidia-docker(这是centos卸载方式)

#docker volume ls -q -f driver=nvidia-docker | xargs -r -I{} -n1 docker ps -q -a -f volume={} | xargs -r docker rm -f

#sudo yum remove nvidia-docker

（3）查看nvidia-docker状态等

systemctl status nvidia-docker

systemctl start nvidia-docker

systemctl stop nvidia-docker

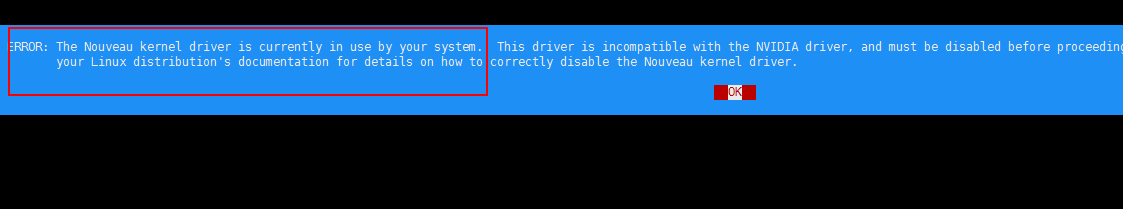
nvidia-docker volume ls

（4）重命名镜像

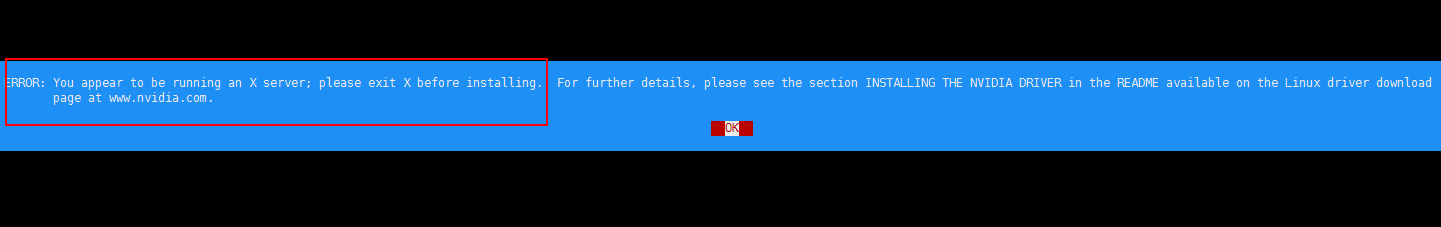
#docker tag IMAGEID(镜像id) REPOSITORY:TAG（仓库：标签）

下面是可能出现的问题（我踩过的坑）

（1）ERROR: The Nouveau kernel driver is currently in use by your system. This driver is incompatible with the NVIDIA driver。说明是Nouveau没有禁用，请看我上面写的“第8步内容：禁用nouveau”



（2）ERROR: You appear to be running an X server; please exit X before installing.是因为没有关闭X server，输入init 3。



（3）The driver installation is unable to locate the kernel source. Please make sure that the

 kernel source packages are installed and set up correctly.

If you know that the kernel source packages are installed and set up correctly, you may pass the location of thekernelsource with the '--kernel-source-path' flag.

解决方法都是安装下面这两个，这也就为什么最好先安装cuda先。

#sudo yum install epel-release

#sudo yum install --enablerepo=epel dkms

（4）ERROR: Unable to load the 'nvidia-drm' kernel module.



解决方法：

关闭BIOS中的UEFI，也就是最前面说的第一步。

（5）装完nvidia后输入nvidia-smi，出现下面这个：

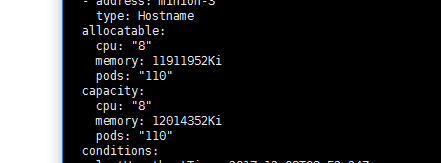
NVIDIA-SMI has failed because it couldn’t communicate with the NVIDIA driver. Make sure that the latest NVIDIA driver is installed and running

解决办法：

出现这个问题说明nvidia没装好，缺少东西，解决就是安装nvidia驱动前确保UEFI关了，还有明确 dkms等是否安装了，往文章上面看看哪些没做的。要么就是安装的kernel-devel版本不一样，需要yum -y update,记得重启电脑。

（6）kubectl get node minion-1 -o yaml显示

alpha.kubernetes.io/nvidia-gpu: "0"或者直接没有显示



说明驱动、nvidia-docker没装好，如果都装好了，用nvidia-docker run能跑容器，说明是kube-apiserver, kube-controller-manager, kube-scheduler, kubelet这些启动参数没配好，尤其是gpu显示是0的这种情况，非常有可能就是kubelet等启动参数没有配好，记得master和node上都要配，我就是在这费了好多时间。忘文章上面看看第11和12步是如何做的。

**16、在这分享几位大神的博客链接，感谢大神们无私的分享和教导。**

（1）、西电涛哥的博客：如何在Kubernetes集群中利用GPU进行AI训练。

<https://my.oschina.net/jxcdwangtao/blog/1574199>

（2）、kubernetes官网使用指南

<https://zhuanlan.zhihu.com/p/27376696>

（3）、nvidia-docker以及挂载路径的一些解析

<http://blog.csdn.net/qq_35254726/article/details/60326433>

（4）、cento7 安装GPU版TensorFlow教程

<http://blog.csdn.net/j790675692/article/details/78133551>

（5）、在CentOS7上安装NVIDIA CUDA 并在docker中使用CUDA

<http://blog.csdn.net/itaacy/article/details/72628792?utm_source=itdadao&utm_medium=referral>

(6)、阿里云部署集群GPU

<https://yq.aliyun.com/articles/203865>

(7)、centos7.3安装nvidia驱动、cuda、cudnn

<https://www.cnblogs.com/mar-q/p/7482720.html>