安装手册

目 录

[需求一：构建Kubernetes集群 1](#_Toc507860151)

[1、首先创建一个基础的temple机 1](#_Toc507860152)

[2、克隆temple机 5](#_Toc507860153)

[3、安装dashboard-UI 8](#_Toc507860154)

[需求二 将 TensorFlow 部署到 Kubernetes 集群里面 9](#_Toc507860155)

[1、下载tensorflow镜像 9](#_Toc507860156)

[2、编写配置文件 9](#_Toc507860157)

[3、执行命令 11](#_Toc507860158)

[4、查看容器情况 12](#_Toc507860159)

[需求三: 通过TensorFlow深度学习平台上测试识别数字算法， 完成对于样本数据的训练以及测试功能。 13](#_Toc507860160)

[1、打开4个终端，分别进入4个Pod。 13](#_Toc507860161)

[2、将MNIST\_data、MNIST\_softmax.py 部署到4个pod 13](#_Toc507860162)

[3、在参数服务器容器执行： 13](#_Toc507860163)

[4、在计算服务器容器执行： 14](#_Toc507860164)

[6、结果（设置总共迭代200次） 14](#_Toc507860165)

[需求四：prometheus对kubernetes集群进行监控 18](#_Toc507860166)

[1、 安装heapster插件 18](#_Toc507860167)

[2、安装prometheus插件 18](#_Toc507860168)

[3、运行Grafana 20](#_Toc507860169)

# 

# 需求一：构建Kubernetes集群

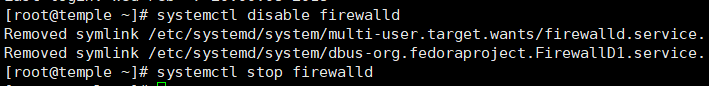
本需求，我们选择在虚拟机里面搭建Kubernetes集群，并且通过Kubernetes自带的dashboard-UI能够管理Kubernetes里面的资源，这里我们用一台master机和两台node机进行测试。

## 1、首先创建一个基础的temple机

### 1.1、关闭防火墙

systemctl disable firewalld

systemctl stop firewalld

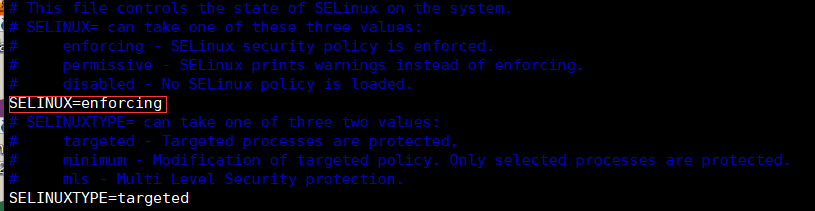


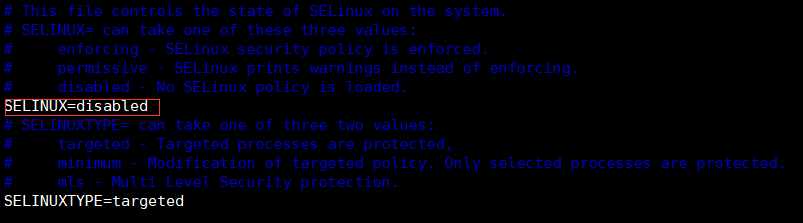
### 1.2、关闭selinux

临时关闭：setenforce 0

永久关闭：vim /etc/selinux/config

将selinux=enforcing改为selinux=disabled



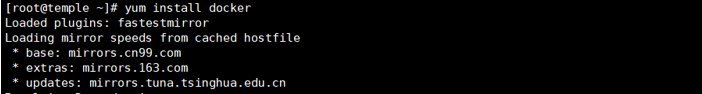


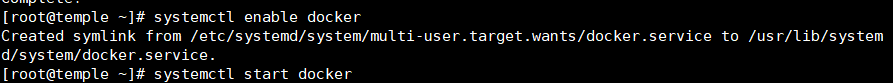
### 1.3、安装docker

yum install docker

systemctl enable docker

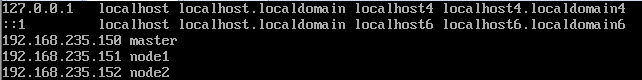
systemctl start docker





### 1.4、配置hosts文件

vim /etc/hosts

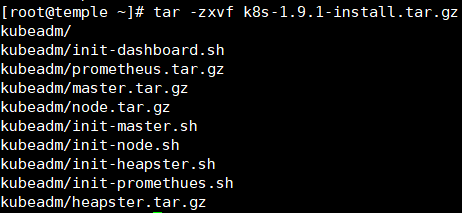


### 1.5、将k8s-1.9.1-install.tar.gz 文件包导入机子

通过WinSCP这个软件将k8s-1.9.1-install.tar.gz导入temple中的root目录下

### 1.6、解压k8s-1.9.1-install.tar.gz文件包

tar -zxvf k8s-1.9.1-install.tar.gz



注释：我们的k8s-1.9.1-install.tar.gz文件包中主要有如下的内容及其命令

[rootanasterl kubeadml# tree 
heapster 
t 
heapster.tar 
manifests 
grafana.yaml 
heapster-rbac.yaml 
heapster. yaml 
influxdb.yaml 
init -dashboard .sh 
init -heapster.sh 
init -master. sh 
init-node. sh 
init -p romethues. sh 
master 
l€-kubeadm . conf 
calico . yaml 
dashboard -admin. yaml 
kubeadm 
kubectl 
kubelet 
kubelet . service 
kubernetes -dashboa rd. yaml 
master. tar 
node 
l€-kubeadm . con f 
kubeadm 
kubectl 
kubelet 
kubelet.service 
p rometheus 
build .sh 
con figs 
alertmanager-tenplates 
t 
default . tinpl 
slack. tmpl 
grafana 
grafana -net -2 -dashboa rd . json 
grafana -net -737 -dashboard .json 
p rometheus -datasou rce. json 
prometheus 
t 
p rometheus . yaml 
rules 
cpu -usage. rules 
instance-availability. rules 
low-disk-space. rules 
man -usage. rules 
development .md 
docs 
grafana_cluster_overview.png 
g ra .png 
index .md 
p rometheus_alerts .png 

p rom etheus_ta rgets. png 
manifests 
€-namespace.yaml 
alertmanager 
alertmanager -tenplates. yaml 
configmap . yaml 
deployment . yaml 
service.yaml 
grafana 
deployment . yaml 
import-dashboards 
t 
configmap . yaml 
job . yaml 
ing ress . yaml 
service. yaml 
p rometheus 
configmap . yaml 
deployment . yaml 
kube-state-metrics 
deployment . yaml 
rbac . yaml 
service. yaml 
node-directo ry -size-met rics 
daanonset.yaml 
node-exporter 
t 
daanonset.yaml 
service. yaml 
p rometheus - rules. yaml 
rbac . yaml 
service. yaml 
manifests -au . yaml 
promethues.tar 
19 directories, 64 files 
[rootanasterl kubeadml# C] 

## 2、克隆temple机

对temple机进行克隆，克隆出三个机子分别命名为master机，node1机和node2机

### 2.1、开机，配置每台机子的hostname

hostnamectl set-hostname master

hostnamectl set-hostname node1

hostnamectl set-hostname node2

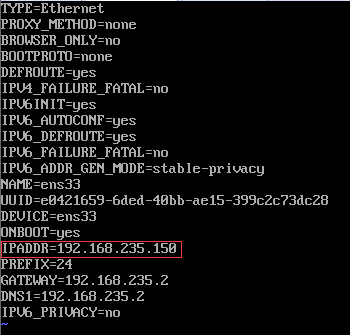
### 2.2、然后将本机IP配置进去

vim /etc/sysconfig/network-scripts/ifcfg-ens33

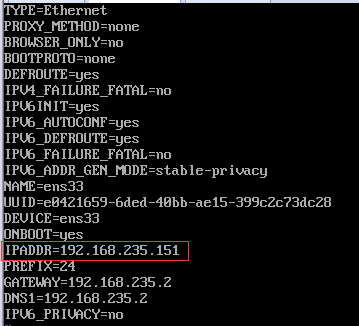
对应/etc/hosts 文件的IP及hostname

完成后重启

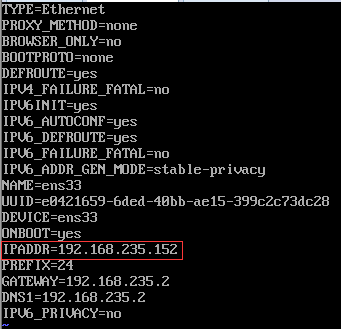
最开始的内容：



mster机修改后的内容：



node1机修改后的内容：

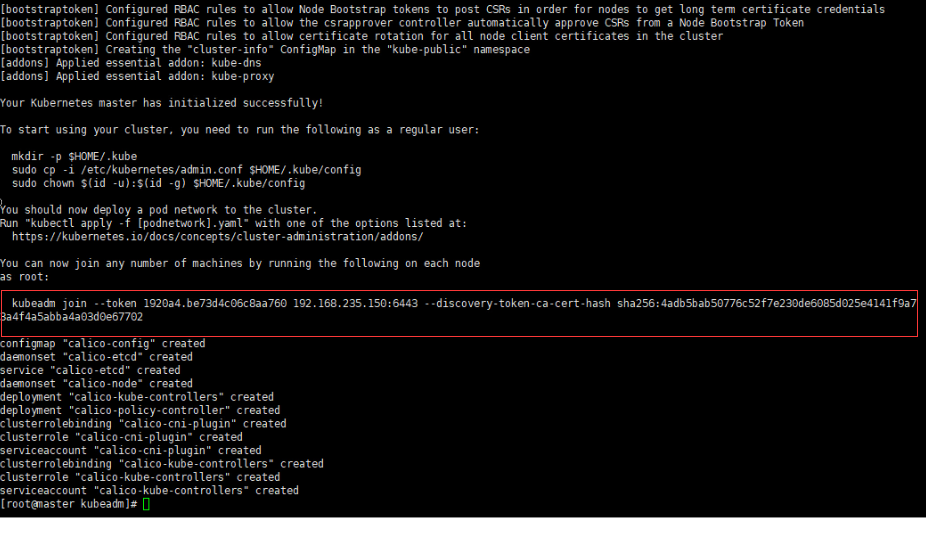
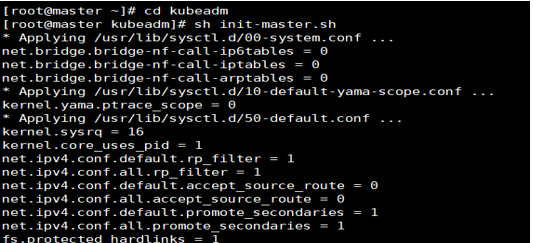


### 2.3、进入kubeadm目录

cd kubeadm

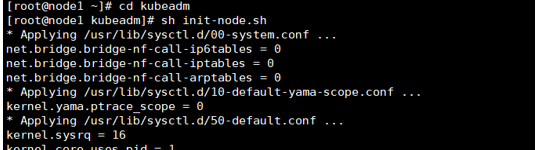
### 2.4、master机上执行sh init-master.sh

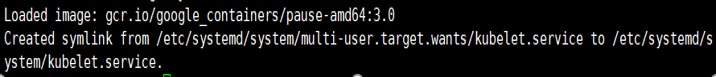
master机执行：



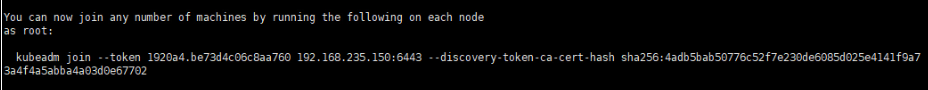
### 2.5、node机执行 sh init-node.sh

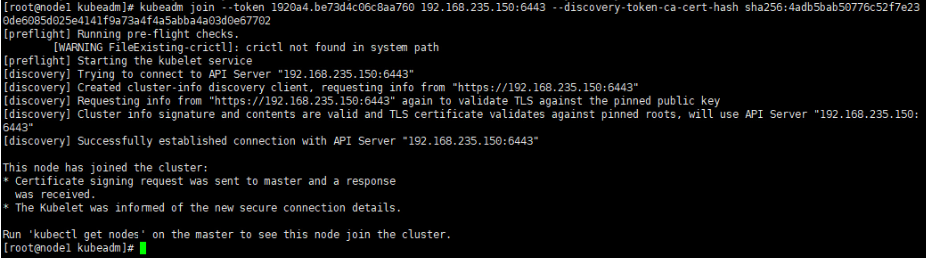
node1、2执行：

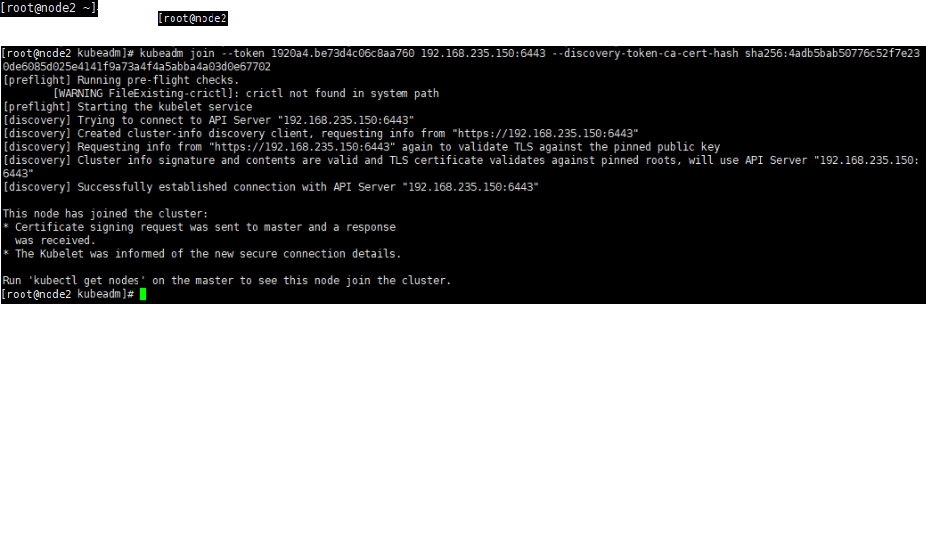




### 2.6、将master机上得出的 kubeadm join 命令 在node机上执行







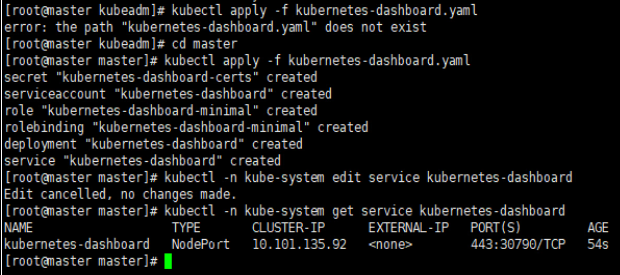
## 3、安装dashboard-UI

### 3.1、安装 dashboard 在master机上执行 sh init-dashboard.sh

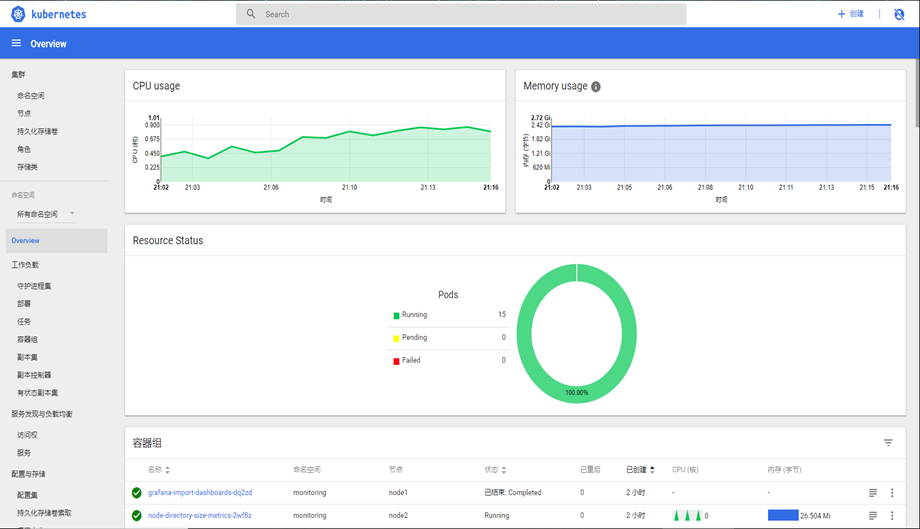
init-dashboard.sh中的命令

![C:\Users\王旭辉\Documents\Tencent Files\1247479570\Image\C2C\L_CN76OV0(_N4G6)OC}](I3.png](data:image/png;base64,)

执行init-dashboard.sh中的命令



最后我们出来的监控画面如下：



# 需求二 将 TensorFlow 部署到 Kubernetes 集群里面

## 1、下载tensorflow镜像

docker run -it -p 8888:8888 tensorflow/tensorflow

## 2、编写配置文件

### 2.1 配置参数服务器部署(deployment)文件，命名tf-ps-deployment.yaml

### #vim tf-ps-deployment.yaml

### apiVersion: extensions/v1beta1

### kind: Deployment

### metadata:

### name: tensorflow-ps2

### spec:

### replicas: 2

### template:

### metadata:

### labels:

### name: tensorflow-ps2

### role: ps

### spec:

### containers:

### - name: ps

### image: tensorflow/tensorflow

### ports:

### - containerPort: 2222

### 

### 2.2 配置参数服务器服务(Service)文件，命名tf-ps-service.yaml

### #vim tf-ps-service.yaml

### apiVersion: v1

### kind: Service

### spec:

### ports:

### - {port: 2222, targetPort: 2222}

### selector: {name: tensorflow-ps2}

### metadata:

### labels: {name: tensorflow, role: service}

### name: tensorflow-ps2-service

### 

### 2.3 配置计算服务器部置文件，命名tf-worker-deployment.yaml

### #vim tf-worker-deployment.yaml

### apiVersion: extensions/v1beta1

### kind: Deployment

### metadata: {name: tensorflow-worker2}

### spec:

### replicas: 2

### template:

### metadata:

### labels:

### name: tensorflow-worker2

### role: worker

### spec:

### containers:

### - name: worker

### image: tensorflow/tensorflow

### ports:

### - containerPort: 2222

### 

### 2.4 配置计算服务器服务文件，命名tf-worker-service.yaml

### #vim tf-worker-service.yaml

### apiVersion: v1

### kind: Service

### spec:

### ports:

### - {port: 2222, targetPort: 2222}

### selector: {name: tensorflow-worker2}

### metadata:

### labels: {name: tensorflow-worker2, role: service}

### name: tensorflow-wk2-service

## 3、执行命令

kubectl create -f tf-ps-deployment.yaml

kubectl create -f tf-ps-service.yaml

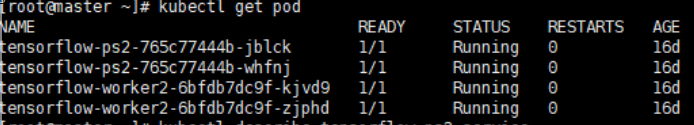
kubectl create -f tf-worker-deployment.yaml

kubectl create -f tf-worker-service.yaml

## 4、查看容器情况

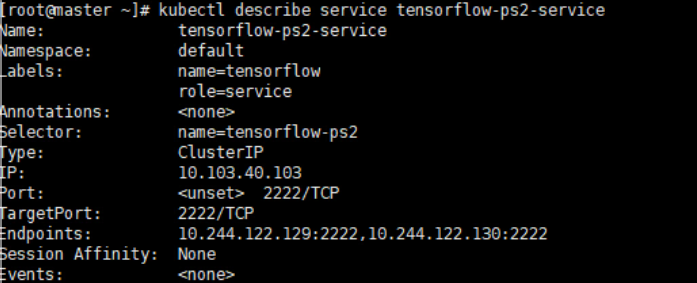
执行如下命令查看：

kubectl get pod



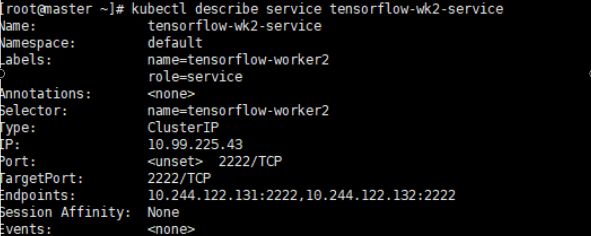
查看2台ps\_host的IP地址：

kubectl describe service tensorflow-ps2-service



查看2台worker\_host的IP地址：

kubectl describe service tensorflow-worker2-service



需求三: 通过TensorFlow深度学习平台上测试识别数字算法， 完成对于样本数据的训练以及测试功能。

1、打开4个终端，分别进入4个Pod。

kubectl exec –ti tensorflow-ps2-765c77444b-jblck /bin/bash

kubectl exec –ti tensorflow-ps2-765c77444b-whfnj /bin/bash

kubectl exec –ti tensorflow-worker2-6bfdb7dc9f-kjvd9 /bin/bash

kubectl exec –ti tensorflow-worker2-6bfdb7dc9f-zjphd /bin/bash

2、将MNIST\_data、MNIST\_softmax.py 部署到4个pod

kubectl cp MNIST\_data tensorflow-ps2-765c77444b-jblck:/notebooks

kubectl cp MNIST\_data tensorflow-ps2-765c77444b-whfnj:/notebooks

kubectl cp MNIST\_data tensorflow-worker2-6bfdb7dc9f-kjvd9:/notebooks

kubectl cp MNIST\_data tensorflow-worker2-6bfdb7dc9f-zjphd:/notebooks

kubectl cp MNIST\_softmax.py tensorflow-ps2-765c77444b-jblck:/notebooks

kubectl cp MNIST\_softmax.py tensorflow-ps2-765c77444b-whfnj:/notebooks

kubectl cp MNIST\_softmax.py tensorflow-worker2-6bfdb7dc9f-kjvd9:/notebooks

kubectl cp MNIST\_softmax.py tensorflow-worker2-6bfdb7dc9f-zjphd:/notebooks

3、在参数服务器容器执行：

python mnist\_replica.py --ps\_hosts=10.244.122.129:2222, 10.244.122.130:2222 --worker\_hosts=172.17.0.3:2222,172.17.0.8: 2222 --job\_name="ps" --task\_index=0

python mnist\_replica.py --ps\_hosts=10.244.122.129:2222, 10.244.122.130:2222 --worker\_hosts=172.17.0.3:2222,172.17.0.8: 2222 --job\_name="ps" --task\_index=1

4、在计算服务器容器执行：

python mnist\_replica.py --ps\_hosts=10.244.122.131:2222, 10.244.122.132:2222 --worker\_hosts=172.17.0.3:2222,172.17.0.8: 2222 --job\_name="worker" --task\_index=0

python mnist\_replica.py --ps\_hosts=10.244.122.131:2222, 10.244.122.132:2222 --worker\_hosts=172.17.0.3:2222,172.17.0.8: 2222 --job\_name="worker" --task\_index=1

6、结果（设置总共迭代200次）

工作节点1执行了4次迭代，输出如下：

job name = worker

task index = 0

2018-02-27 09:04:29.769414: I tensorflow/core/platform/cpu\_feature\_guard.cc:137] Your CPU supports instructions that this TensorFlow binary was not compiled to use: SSE4.1 SSE4.2 AVX AVX2 FMA

2018-02-27 09:04:29.771434: I tensorflow/core/distributed\_runtime/rpc/grpc\_channel.cc:215] Initialize GrpcChannelCache for job ps -> {0 -> localhost:2222}

2018-02-27 09:04:29.771518: I tensorflow/core/distributed\_runtime/rpc/grpc\_channel.cc:215] Initialize GrpcChannelCache for job worker -> {0 -> localhost:2223, 1 -> localhost:2224}

2018-02-27 09:04:29.772636: I tensorflow/core/distributed\_runtime/rpc/grpc\_server\_lib.cc:324] Started server with target: grpc://localhost:2223

WARNING:tensorflow:From MNIST\_softmax.py:171: \_\_init\_\_ (from tensorflow.python.training.supervisor) is deprecated and will be removed in a future version.

Instructions for updating:

Please switch to tf.train.MonitoredTrainingSession

Worker 0: Initializing session...

2018-02-27 09:05:06.774554: I tensorflow/core/distributed\_runtime/master\_session.cc:1017] Start master session f45b3b4f211ea180 with config: device\_filters: "/job:ps" device\_filters: "/job:worker/task:0" allow\_soft\_placement: true

Worker 0: Session initialization complete.

Training begins @ 1519740312.522976

1519740392.448870: Worker 0: training step 1 done (global step: 163)

1519740392.468708: Worker 0: training step 2 done (global step: 165)

1519740392.891714: Worker 0: training step 3 done (global step: 167)

1519740393.279025: Worker 0: training step 4 done (global step: 201)

Training ends @ 1519740393.279250

Training elapsed time: 80.756274 s

After 200 training step(s), validation cross entropy = 923.842

工作节点2执行了196次迭代，输出如下：

job name = worker

task index = 1

2018-02-27 09:04:37.440751: I tensorflow/core/platform/cpu\_feature\_guard.cc:137] Your CPU supports instructions that this TensorFlow binary was not compiled to use: SSE4.1 SSE4.2 AVX AVX2 FMA

2018-02-27 09:04:38.244376: I tensorflow/core/distributed\_runtime/rpc/grpc\_channel.cc:215] Initialize GrpcChannelCache for job ps -> {0 -> localhost:2222}

2018-02-27 09:04:38.244497: I tensorflow/core/distributed\_runtime/rpc/grpc\_channel.cc:215] Initialize GrpcChannelCache for job worker -> {0 -> localhost:2223, 1 -> localhost:2224}

2018-02-27 09:04:38.245910: I tensorflow/core/distributed\_runtime/rpc/grpc\_server\_lib.cc:324] Started server with target: grpc://localhost:2224

WARNING:tensorflow:From MNIST\_softmax.py:171: \_\_init\_\_ (from tensorflow.python.training.supervisor) is deprecated and will be removed in a future version.

Instructions for updating:

Please switch to tf.train.MonitoredTrainingSession

Worker 1: Waiting for session to be initialized...

2018-02-27 09:05:06.774558: I tensorflow/core/distributed\_runtime/master\_session.cc:1017] Start master session 8d45a5db42536158 with config: device\_filters: "/job:ps" device\_filters: "/job:worker/task:1" allow\_soft\_placement: true

Worker 1: Session initialization complete.

Training begins @ 1519740307.661162

1519740311.406557: Worker 1: training step 1 done (global step: 0)

1519740311.587401: Worker 1: training step 2 done (global step: 1)

1519740311.599175: Worker 1: training step 3 done (global step: 2)

1519740311.607491: Worker 1: training step 4 done (global step: 3)

1519740311.615004: Worker 1: training step 5 done (global step: 4)

#......中间略去

1519740392.783947: Worker 1: training step 192 done (global step: 196)

1519740392.794592: Worker 1: training step 193 done (global step: 197)

1519740392.803964: Worker 1: training step 194 done (global step: 198)

1519740392.814065: Worker 1: training step 195 done (global step: 199)

1519740392.824238: Worker 1: training step 196 done (global step: 200)

Training ends @ 1519740392.824369

Training elapsed time: 85.163207 s

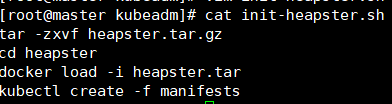
After 200 training step(s), validation cross entropy = 910.885

需求四：prometheus对kubernetes集群进行监控

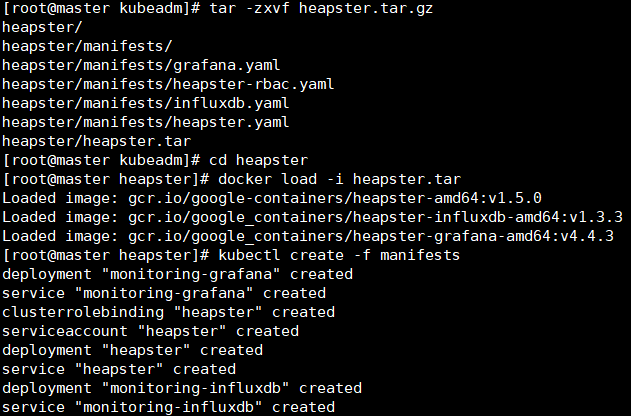
1. 安装heapster插件

在master机上执行 init-heapster.sh

init-heapster.sh中的命令



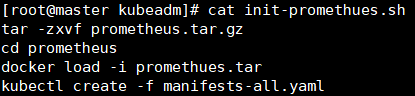
执行init-heapster.sh中的命令



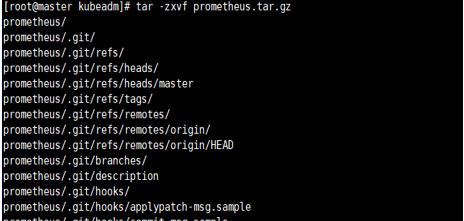
2、安装prometheus插件

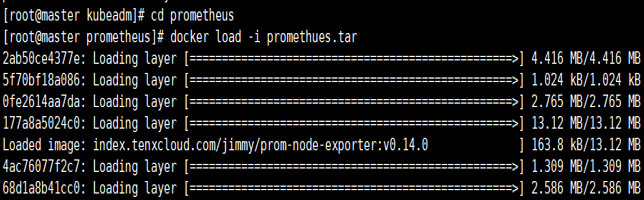
在master机上执行 init-prometheus.sh

init-prometheus.sh中的命令

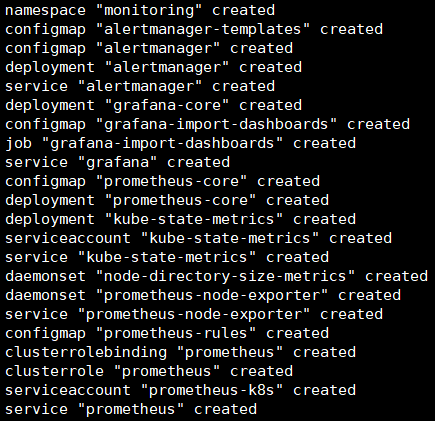


执行init-prometheus.sh中的命令

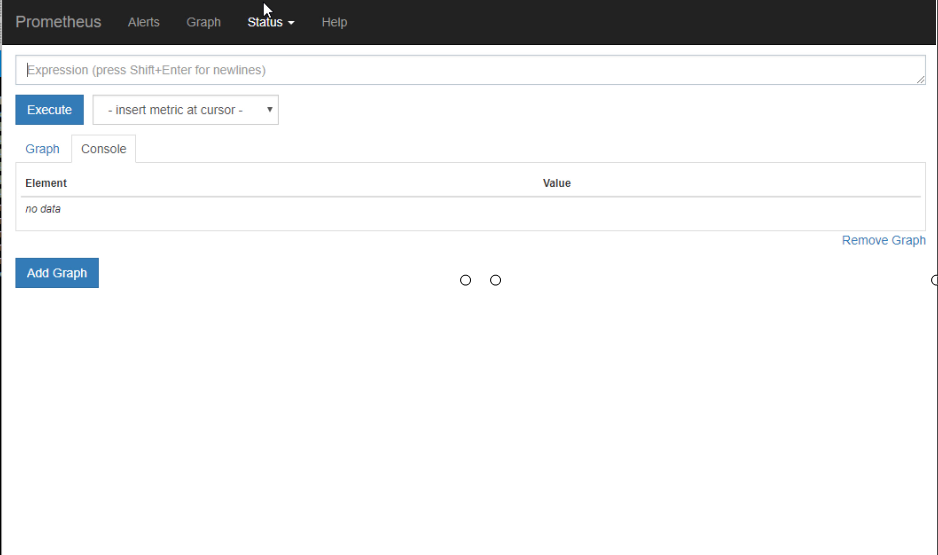




Kubectl create –f manifests-all.yaml



在浏览器中打开：http://localhost:9090



1. 运行Grafana

http://localhost:3000

