

Kubernetes 练习

一、概述

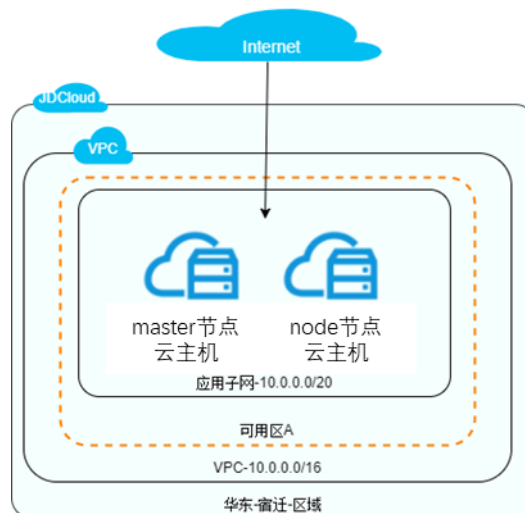
本练习使用京东智联云主机和 kubeadm 创建一个最小化的 Kubernetes 集群。集群包括两个节点，一个是 master 节点，一个是 node 节点。

在集群创建成功之后，使用 kubectl 在该集群上运行简单的应用。

二、配置信息

序号	资源类型	规格	数量
1	VPC	N/A	1
2	Subnet	N/A	1
3	公网 IP	带宽 1M	1
4	云主机	2 核 4GB	2

三、系统架构



四、操作步骤

1. 登录京东智联云控制台领取云资源

1.1 登录京东智联云账号

【如有登录有问题，请检查浏览器，一定设置 Chrome 为默认浏览器】

体验登录网址: <https://console.jdcloud.com/>



如您为新用户, 请注册后登录。如您为老用户, 请直接登录。



1.2 领取云资源

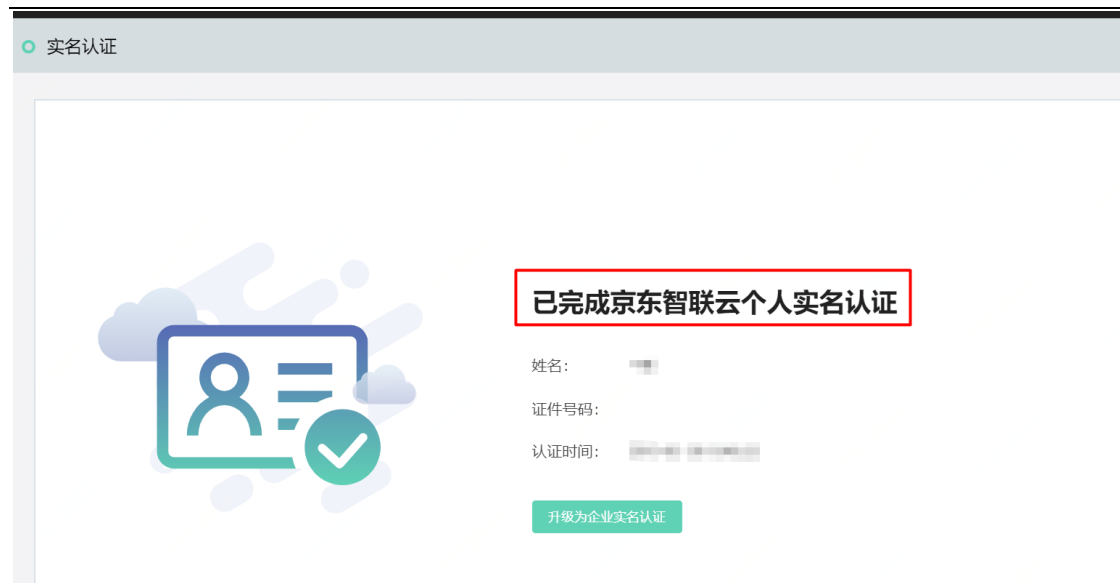
点击控制台->账户管理->实名认证->个人实名认证



点击立即认证->从下方三种方式选择认证，建议选择实名手机认证，可快速通过认证



认证成功



添加小助手 (ID: `jdcloud_dev`) 回复公开课, 进入交流群领取云资源包



(提示: 账号实名认证后方可领取)

点击费用管理->代金券管理->余额, 查看代金券是否到账。后续可进行动手实操



登录成功后进入控制台后看到如下界面。



2. 创建虚拟专用网 VPC

点击云服务->私有网络，**选择华北-北京**，点击**创建**。



输入名称，点击**确定**。



3. 创建子网

点击私有网络->子网，选择华北-北京，点击**创建**。



输入名称，选择刚刚创建的私有网络 VPC-1，点击确定。

新建子网

地域: 华北-北京 华南-广州 华东-宿迁 华东-上海 ?

* 名称: subnet-10
名称不可为空，只支持中文、数字、大小写字母、英文下划线 "_" 及中划线 "-", 且不能超过32字符

私有网络: VPC-1(10.0.0.0/16) 新建私有网络

VPC CIDR: 10.0.0.0/16

* IPv4 CIDR: 10 . 0 . 0 . 0 / 20

* 路由表: 默认路由表

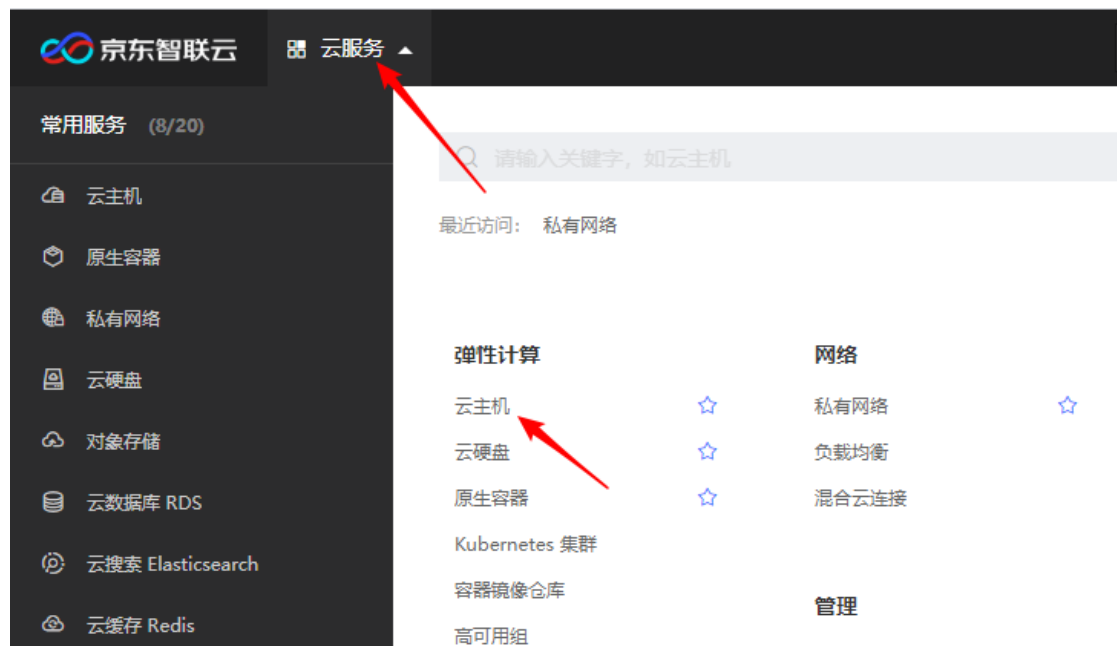
描述:
描述不能超过256个字符

点击确定

取消 确定

4. 创建云主机

点击云产品->云主机



点击“创建”



点击 按配置，地域域选择华北-北京，可用区 B



选择本地系统盘

镜像选择 Ubuntu Ubuntu 18.04 64 位

规格选择 计算优化 标准型 c.n2.large 2 核 4GB (如果显示规格售罄, 请在上面更换)

可用区后重试)

镜像

类型: **官方** 私有 共享 镜像市场

分类: 云硬盘系统盘 **本地盘系统盘**

本地数据盘为临时存储盘，有丢失数据的风险（比如发生迁移或宿主机宕机等情况），不适用于应用层没有数据冗余架构的使用场景，建议您使用云硬盘存储重要数据。

官方镜像: Ubuntu Ubuntu 18.04 64位

规格

分类: 全部 通用型 **计算优化** 内存优化 高频计算 GPU

规格类型: 全部 vCPU: 请选择 内存: 全部

实例规格: 例如g.n2.medium ☐ 裸金属 **查询**

规格类型	规格	vCPU	内存
<input checked="" type="radio"/> 计算优化 标准型	c.n2.large	2核	4GB
<input type="radio"/> 计算优化 标准型	c.n2.xlarge	4核	8GB
<input type="radio"/> 计算优化 标准型	c.n2.2xlarge	8核	16GB

存储使用默认配置，网络选择刚刚创建的 **VPC 和子网**，安全组选择 **默认安全组开放全部端口**（此安全组仅用于调试、学习，请不要在生产环境使用该安全组），带宽使用默认配置。

存储

系统盘: 类型: 本地盘 容量: 40 GB 快照: bm-centos7...

设备名: /dev/vda ☐ 加密 ☒ 随云主机释放

本地数据盘为临时存储盘，有丢失数据的风险（比如发生迁移或宿主机宕机等情况），不适用于应用层没有数据冗余架构的使用场景，建议您使用云硬盘存储重要数据。不同镜像分类支持不同系统盘类型，若需更换系统盘类型请先更换镜像分类，更换后下方配置的数据盘信息将会清空

数据盘: 添加一块云硬盘，一共可增加8块（数据盘+系统盘） 恢复镜像默认配置

网络

私有网络: VPC-1 subnet-10 新建私有网络 新建子网

该子网下还可以创建 4092 台云主机。

内网IP: 自动分配 系统将自动分配

将在10.0.0.0/20区间内自动分配内网IP

安全组: **默认安全组开放全部端口** 新建安全组

安全组是一种分布式、有状态的虚拟防火墙，用于实现云主机/容器的网络访问控制，[操作指南](#)

风险提示: 当前选择的安全组会开放所有端口，容易遭受攻击者/扫描器直接攻击；为了保证业务的安全性，建议合理设置安全组的开放端口。

带宽

带宽计费: ☒ 按固定带宽 ☐ 按使用流量 [?](#)

线路: ☒ BGP [?](#)

带宽上限: 1 Mbps

1 200

支持暂不购买公网IP, 可云主机创建完成后另行购买再绑定, [点击暂不购买](#)。

输入名称 **k8s**, 设置密码, 请牢记密码用于后续步骤中登录云主机。

基本信息

* 名称:

不为空且只允许中文、数字、大小写字母、英文下划线“_”、中划线“-”及点“.”, 不能以“.”作为首尾, 不超过128字符

描述:

描述不能超过256个字符

标签: [+](#) 添加 [?](#)

登录信息

设置密码: ☒ 立即设置 ☐ 暂不设置 [?](#)

* 密码:

* 确认密码:

修改购买数量为 2, 点击立即购买

描述不能超过256个字符

标签: [+](#) 添加 [?](#)

登录信息

设置密码: ☒ 立即设置 ☐ 暂不设置 [?](#)

* 密码:

* 确认密码:

高级选项

自定义数据: ☐ [?](#)

购买量

数量: [?](#)

基于以上配置, 可批量创建云主机的数量。若选择自定义内网IP, 暂不支持批量创建云主机。

已选配置

地域: 华北-北京
可用区: 可用区8
镜像: CentOS --
规格: g.n2.medium(1核 4GB 通用 标...
系统盘: 本地盘(40GB)
数据盘: --
私有网络: VPC-1
子网: subnet-10
公网IP: BGP 按带宽1Mbps
购买数量: 2
费用: **¥1.06/小时**
(合计 ¥763.20/月)

[立即购买](#)

当前配置包年包月仅售 **¥38.00/月**

点击 **已阅读并同意，点击立即开通。**

基础云 - 公网IP	地域: 华北-北京 线路: BGP 带宽上限: 按带宽 1Mbps	1	按配置	¥ 0.06/小时
基础云 - 公网IP	地域: 华北-北京 线路: BGP 带宽上限: 按带宽 1Mbps	1	按配置	¥ 0.06/小时
基础云 - 云主机	地域: 华北-北京 可用区: 可用区B 名称: k8s 镜像: CentOS 7.6 64位 规格: g.n2.medium 1核 4G 系统盘: 40GB 本地盘 子网: subnet-10	1	按配置	¥ 0.47/小时
基础云 - 云主机	地域: 华北-北京 可用区: 可用区B 名称: k8s 镜像: CentOS 7.6 64位 规格: g.n2.medium 1核 4G 系统盘: 40GB 本地盘 子网: subnet-10	1	按配置	¥ 0.47/小时

☒ 已阅读并同意 《云主机服务条款》 《公网IP服务条款》

应付款: ¥ 1.06

立即开通

点击刷新按钮，直到云主机进入**运行中**状态，并且获得公网 IP 地址。名称为 k8s1 的节点将作为 master，k8s2 将作为 node。

云主机								
<div> 华北-北京 华南-广州 华东-宿迁 华东-上海 创建 </div> <div> 主机名称 <input type="text"/> 请输入名称进行搜索 </div>								
ID/名称	可用区	主IP地址	私有网络/子网	状态	规格类型	配置	计费信息	操作
i-4g7vpq3awk		116.196.122.192	VPC-1					
k8s2 node	可用区B	116.196.122.192 (内)	subnet-10	运行中	g.n2.medium 通用 标准型	CPU: 1% 1核 内存: 1.5% 4GB 带宽: 1Mbps	按配置 2020-03-23 09:45:42 创建	远程连接 停止 更多
i-47mzmt33t		116.196.120.122	VPC-1					
k8s1 master	可用区B	116.196.120.122 (内)	subnet-10	运行中	g.n2.medium 通用 标准型	CPU: 0% 1核 内存: 1.5% 4GB 带宽: 1Mbps	按配置 2020-03-23 09:45:43 创建	远程连接 停止 更多

5. 使用 kubeadm 安装单节点 kubernetes 集群

写在前面：

- 本操作用例参考 kubernetes 官方文档 *Creating a single control-plane cluster with kubeadm* <https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/>
- 本操作用例创建一个单工作节点集群，仅用于学习、实验目的，不能用于生产环境。
- 使用 kubeadm 创建集群步骤比较简单，但是由于众所周知的原因，某些在需要在 google 网站 (google.com, gcr.io) 上下载的资源无法直接下载。

本操作例的一部分内容用于绕过需要从 google 网站上下载的内容。通过比较本文档与 google 官方文档，您可以大致区别出来哪些内容是为了避免直接从 google 网站上下载而特别增加的。作者也会尽力在文档中指出。

- 大部分操作需要在 master 和 node 节点上分别执行一次，文档中会标明该操作需要在哪个节点上执行。

5.1 登录云主机控制台

点“云产品”-“云主机”，进入云主机列表页：



ID/名称	可用区	主IP地址	私有网络/子网	状态	规格类型	配置	计费信息	操作
i-3wd0y2k9a k8s2	可用区B	116.196.100.95 10.0.0.7 (内)	VPC-1 subnet-10	运行中	c.n2.large 计算优化 标准型	CPU: 3% 2核 内存: 10% 4GB 带宽: 1Mbps	按配置 2020-03-23 10:56:26 创建	远程连接 停止 更多
i-pi2usc2mh1 k8s1	可用区B	116.196.101.87 10.0.0.8 (内)	VPC-1 subnet-10	运行中	c.n2.large 计算优化 标准型	CPU: 0% 2核 内存: 23% 4GB 带宽: 1Mbps	按配置 2020-03-23 10:56:57 创建	远程连接 停止 更多

在列表页找到 master (k8s1)和 node (k8s2)的公网 IP 地址。如果云主机的 IP 地址未显示，您可以点击刷新按钮。您可能需要 1 到 2 分钟以等待云主机各种资源准备完毕。

请使用 SSH 客户端（推荐[下载 mobaXterm](#)）远程连接云主机。

```
[root@k8s2 ~]# ssh root@116.196.100.95
Last failed login: Mon Mar 23 11:34:04 CST 2020 from 92.63.194.105 on ssh:notty
There were 6 failed login attempts since the last successful login.
Last login: Mon Mar 23 11:10:19 2020 from 124.126.2.222
Welcome to JDCLLOUD Elastic Compute Service
[root@k8s2 ~]#
```

以下操作均在 ssh 客户端远程控制台完成。

5.2 安装 Docker

执行节点: k8s1 + k8s2

开始安装 Docker 软件包: 输入 `apt update && apt install -y docker.io` 如下图:

```
root@k8s2:~# apt update && apt install -y docker.io
```

5.3 安装 kubernetes 组件和 kubeadm

[由于无法从 google 下载软件包, 本操作将软件仓库地址改为国内地址]

执行节点: k8s1 + k8s2

```
apt-get update && apt-get install -y apt-transport-https curl
cat <<EOF | tee /etc/apt/sources.list.d/kubernetes.list
deb http://mirrors.ustc.edu.cn/kubernetes/apt kubernetes-xenial main
EOF
gpg --keyserver keyserver.ubuntu.com --recv-keys 6A030B21BA07F4FB
gpg --export --armor 6A030B21BA07F4FB | apt-key add -
apt-get update
apt-get install -y kubelet kubeadm kubectl
```

在本操作之后, kubelet 服务可能无法运行, 这是正常的

5.4 下载关键镜像

[由于 kubernetes 本身需要一些在 gcr.io 上存放的镜像, 这些镜像国内无法正常下载, 本操作从国内仓库预先下载这些镜像。这种方式仅用于学习和调试, 正式生产环境不能用这种方式]

执行节点: k8s1 + k8s2

```
#pull-image.sh
```

```
MY_REGISTRY=gcr.azk8s.cn/google_containers

images=$(kubeadm config images list | grep ^k8s.gcr.io | cut -d '/' -f2)

for imageName in ${images[@]} ; do

    docker pull gcr.azk8s.cn/google_containers/$imageName

    docker tag gcr.azk8s.cn/google_containers/$imageName k8s.gcr.io/$imageName

done
```

5.5 修改 iptable 配置

执行节点：k8s1 + k8s2

```
cat <<EOF > /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
sysctl --system
```

该配置使 iptable 可以作用于 bridge，使 kube-proxy 可以正常工作。

5.6 初始化 cluster

执行节点：k8s1

在 master 节点上运行：

```
kubeadm init
```

执行成功后会看到如下提示：

```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 10.0.0.10:6443 --token fphruy.p2gmcib0f5n8rbqe \
  --discovery-token-ca-cert-hash sha256:eef8fe91749060a59d938ff39e468890a73c13e1224c8207747971af10da7451
```

拷贝内容在k8s2执行

执行节点: k8s1

在 master 节点执行:

```
mkdir -p $HOME/.kube
cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
chown $(id -u):$(id -g) $HOME/.kube/config
```

该命令用于配置 kubectl 客户端所需的必要信息。

执行节点: k8s2

拷贝第二个红框的内容, 按照提示, 在 node 节点执行:

```
kubeadm join [master ip]:6443 --token [example] \
  --discovery-token-ca-cert-hash [example]
```

成功执行后, 将看到如下提示:

```
This node has joined the cluster:
* Certificate signing request was sent to apiservert and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

5.7 安装 cni 网络插件

至此集群已经成功创建, 我们在 master 节点可以使用 kubectl 查看集群状态:

执行节点: k8s1

在 master 节点上执行: kubectl get node

```
[root@k8s1 ~]# kubectl get node
NAME      STATUS    ROLES    AGE     VERSION
k8s1      NotReady  master   5m4s    v1.17.4
k8s2      NotReady  <none>   2m3s    v1.17.4
```

我们可以看到集群有两个节点，其中一个是一个 master，一个是 node。但是两个节点都是 NotReady 状态，这是因为还没有安装网络插件。（通过 `kubectl describe node k8s1` 可以查看节点 NotReady 的详细原因）

在 master 节点上执行以下命令安装网络插件 (calico):

```
kubectl apply -f https://docs.projectcalico.org/v3.11/manifests/calico.yaml
```

```
[root@k8s1 ~]# kubectl apply -f https://docs.projectcalico.org/v3.11/manifests/calico.yaml
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
daemonset.apps/calico-node created
serviceaccount/calico-node created
deployment.apps/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
```

5.8 安装完成

执行节点: k8s1

在 master 节点上执行: `kubectl get node`

```
[root@k8s1 ~]# kubectl get node
NAME      STATUS    ROLES    AGE     VERSION
k8s1      Ready     master   14m     v1.17.4
k8s2      Ready     <none>   11m     v1.17.4
```

请注意，镜像下载需要一定时间。您可能需要 10 分钟以上才能看到集群的

两个节点都处于 Ready 状态。

6. 创建应用

6.1 简单应用

执行节点: k8s1

在 master 节点上执行: `kubectl run --generator run-pod/v1 bb --image busybox -- sh -c "echo hello world && sleep 1d"`

```
[root@k8s1 ~]# kubectl run --generator run-pod/v1 bb --image busybox -- sh -c "echo hello world && sleep 1d"
pod/bb created
[root@k8s1 ~]#
```

查看 POD 运行状态: `kubectl get pod`

```
[root@k8s1 ~]# kubectl get pod
NAME      READY   STATUS    RESTARTS   AGE
bb        1/1     Running   0           13s
```

查看 POD 输出: `kubectl logs bb`

```
[root@k8s1 ~]# kubectl logs bb
hello world
```

6.2 创建 nginx 应用及服务

执行节点: k8s1

在 master 节点上执行以下命令创建 nginx-deploy.yaml:

```
cat <<EOF > nginx-deploy.yaml

apiVersion: apps/v1

kind: Deployment

metadata:
  name: nginx-deployment
  labels:
    app: nginx

spec:
  replicas: 2
  selector:
```

```

    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
EOF

```

在 master 节点上执行以下命令创建一个 deployment:

```
kubectl apply -f nginx-deploy.yaml
```

```

[root@k8s1 ~]# kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-deployment created
[root@k8s1 ~]#

```

在 master 节点上执行以下命令查看 POD:

```
kubectl get pod
```

```

[root@k8s1 ~]# kubectl get pod

```

NAME	READY	STATUS	RESTARTS	AGE
bb	1/1	Running	0	7m45s
nginx-deployment-86c57db685-2shjd	0/1	ContainerCreating	0	5s
nginx-deployment-86c57db685-c4cwv	0/1	ContainerCreating	0	5s

注意：第一次下载镜像需要一些时间，POD 可能要等待 10 分钟以上才会进入 running 状态。

在 master 节点上执行以下命令创建 nginx-svc.yaml:

```

cat <<EOF > nginx-svc.yaml

apiVersion: v1

kind: Service

metadata:

  name: nginx-service

```

```
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      nodePort: 30303
  type: NodePort
EOF
```

在 master 节点上执行以下命令创建一个 service:

```
kubectl apply -f nginx-svc.yaml
```

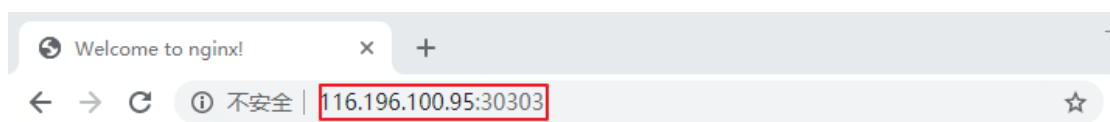
```
[root@k8s1 ~]# kubectl apply -f nginx-svc.yaml
service/nginx-service created
[root@k8s1 ~]#
```

在 master 节点上执行以下命令查看 service:

```
kubectl get svc
```

```
[root@k8s1 ~]# kubectl get svc
NAME                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
kubernetes          ClusterIP     10.96.0.1     <none>         443/TCP          3h14m
nginx-service       NodePort      10.96.65.186  <none>         80:30303/TCP     6s
```

使用浏览器访问 node 节点 (k8s2) 的 IP 地址和端口 30303:



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

7. 结束

恭喜完成所有任务！

请根据个人兴趣继续探索 kubernetes。

完毕后请及时释放不再继续使用的资源。

谢谢！

8. 上传练习截图，领取奖励

部署成功后，请将两个练习页面截图形式回复到文章评论区，有机会获得奖励。

链接地址：<https://developer.jdcloud.com/topics/912>

要求：

Kubernetes 集群作业：页面截图+账号名称（参考下图）

账户名称：jdcloudAI_dev

116.196.68.225:30303

欢迎来到nginx!

如果您看到此页面，则说明Nginx Web服务器已成功安装并正在运行。需要进一步的配置。

有关在线文档和支持，请访问 nginx.org。
可以在nginx.com上获得商业支持。

感谢您使用nginx。

奖励说明：活动时间内成功提交测试结果的，前**3**名部署成功者可以获得京东智联云音响移动电源套装，第**4**名-第**20**名部署成功者可以获得京东 Joy 公仔一只。



活动时间:

请于 2020 年 4 月 1 日晚 20 点前提交，2020 年 4 月 2 日公布结果和领奖方式。