Kubernetes 资源控制管理

·三台机器先安装 k8s 环境，以 kubeadm 的方式安装 1.17.0 版本
注意：yum -y install kubelet-1.17.0 kubeadm-1.17.0 kubectl-1.17.0
镜像也要是 1.17.0 的镜像，
最后按照配置 flannel，确保 master 节点和 node 节点连接成功

一、Kubectl 命令工具

Kubectl 是一个用于操作 Kubernetes 集群的命令行工具，利用 Kubectl 的各种子命令可以实现各种功能，在管理 Kubernetes 集群过程中 kubectl 是非常实用的工具。

Kube-apiserver 是整个 Kubernetes 集群管理的入口。API Server 运行在 Kubernetes 集群的主管理节点（Master）上，用户需要通过 API Server 配置和组织集群，同时集群中各节点与 Etcd 存储的交互也是通过 API Server 来进行的。

API Server 实现了一套 RESTfull 的接口，用户可以直接使用 API 与 API Server 进行交互。另外官方还提供了一个客户端 kubectl 工具集，可直接通过 kubectl 命令行的方式与集群进行交互。

（一）kubectl 命令行的语法

Kubectl [command] [TYPE] [NAME][flags]

·command：子命令，用于操作 kubernetes 集群资源对象的命令。例如：create，delete、describe，get，apply 等。

·TYPE：资源对象的类型，区分大小写，能以单数、复数或者简写形式表示。例如：kubectl get pod pod1、kubectl get pods pod1，kubectl get po pod1 三种 TYPE 是等价的。

·NAME：资源对象名称，区分大小写。如果不指定名称，系统则返回属于 TYPE 的全部对象列表，例如，执行 kubectlget pods 命令即可返回所有 Pod 的列表。

·flags：kubectl 子命令的可选参数，例如使用"-s"指定 API Server 的 URL 地址而不用默认值。

kubectl 的子命令非常丰富，涵盖了对 Kubernetes 集群的主要操作，包括资源对象的创建、删除、查看、修改、配置、运行等操作。

（二）kubectl 命令列表

类型	命令	描述
基础命令	·create	通过文件名或标准输入创建资源
	·expose	将一个资源公开为一个新的 service（暴露端口）
	·run	在集群中运行一个特定的镜像

	·set	在对象上设置特定的功能
	·get	**显示一个或多个资源**
	·explain	文档参考资料
	·edit	使用默认的编辑器编辑一个资源
	·delete	**通过文件名、标准输入、资源名称或标准选择器来删除资源**
部署命令	·rollout	管理资源的发布
	·rolling-update	**对给定的复制控制器滚动更新**
	·scale	**扩容或缩容 pod 数量，deployment、replicaset、RC 或 job**
	·autoscale	创建一个自动选择扩容或缩容并设置 pod 数量
集群管理命令	·certificate	修改证书资源
	·cluster-info	**显示集群信息**
	·top	**显示资源（CPU/Memory/Storage）使用。需要 Heapster 运行**
	·cordon	标记节点不可调度
	·uncordon	标记节点可调度
	·drain	驱逐节点上的应用，准备下线维护
	·taint	修改节点 taint 标记
故障诊断和调试命令	**·describe**	**显示特定资源或资源组的详细信息**
	·logs	**在一个 Pod 中打印一个容器日志。如果 Pod 只有一个容器容器名称是可选的**
	·attach	**附力到一个运行的容器**
	·exec	**执行命令到容器**
	·port-forward	转发一个或多个本地端口到一个 pod
	·proxy	运行一个 proxy 到 Kubernetes API server
	·cp	拷贝文件或目录到容器中
	·auth	检查授权
高级命令	**·apply**	**通过文件名或标准输入对资源应用配置**
	·patch	使用不定修改、更新资源的字段
	·replace	通过文件名或标准输入替换一个资源
	·convert	不同的 API 版本之间转换配置文件
设置命令	**·lable**	**更新资源上的标签**
	·annotate	更新资源上的注释
	·completion	用于实现 kubectl 工具自动补全
其它命令	·api-versions	打印受支持的 API 版本
	·config	修改 Kubeconfig 文件（用于访问 API，比如配置认证信息）
	·help	所有帮助命令
	·plugin	运行一个命令行插件
	·version	打印客户端和服务版本信息

（三）使用 kubectl 工具容器资源

Kubectl 是管理 kubernetes 集群的命令行工具，通过生成的 json 格式传递给 API Server 进行创建、查看、管理的操作

使用 kubectl --help 命令可以查看 kubectl 帮助命令，其中包括基本命令、部署命令、群集管理命令、调试命令以及高级命令等。

```
[root@k8s-master ~]# kubectl --help
kubectl controls the Kubernetes cluster manager.

 Find more information at: https://kubernetes.io/docs/reference/kubectl/overview/

Basic Commands (Beginner):
 create       Create a resource from a file or from stdin.
 expose       使用 replication controller, service, deployment 或者 pod
并暴露它作为一个 新的 Kubernetes Service
 run          在集群中运行一个指定的镜像
 set          为 objects 设置一个指定的特征
```

在一个完整的项目周期中，包含**创建->发布->更新-回滚>删除**等过程，下面针对该过程依次进行操作命令的演示。

1、创建容器

Kubectl run 命令可以创建并运行一个或多个容器镜像，也可以创建一个 deployment 或 job 来管理容器。此命令和 docker run 相类似，也是实现容器镜像的创建，先从仓库中拉取基础镜像，然后对容器进行操作。kubectl run 的命令语法如下所示。

```
kubectl run NAME --image=image [--env="key=value"] [--port=port] [--replicas=replicas] [--dry-run=bool] [--overrides=inline-json] [--command] -- [COMMAND] [args...] [options]
```

各选项的作用分别如下所。

·NAME：指定容器运行的名称；

·Image：指定运行的基础镜像；

·env：指定在容器中设置的环境参数；

·port：指定容器暴露的端口；

·replicas：指定启动容器设置的副本数；（新版本中删除）

·dry-run：dry-run 焦如果为 true，则只打印要发送的对象，而不发送它；

·overrides：生成对象的内联 JSON 重写。如果非空，则用于覆盖生成的对象。要求对象提供有效的 apiVersion 字段。

通过 kubectrun 命令创建 Nginx 容器，指定名称为 nginx-deployment，指定基础镜像为 Nginx

目前最新版本，指定对外暴露的端口为 80 以及副本数为 3。Nginx 容器创建完成后使用 get pod 命令查看 Pod 的信息，可以发现确实有 3 容器资源，并且处于 Running 状态。还可以查看 deployment，也显示的是 3。

```
两个 node 节点上传 nginx 镜像//nginx-1.14.tar 和 nginx-1.19.tar
[root@k8s-node1 ~]# docker load < nginx-1.14.tar
[root@k8s-node1 ~]# docker load < nginx-1.19.tar
[root@k8s-node1 ~]# docker tag nginx nginx:1.19      //修改一下标签

使用命令行发布任务（适合临时的、一次性的）
[root@k8s-master ~]# kubectl run nginx-deployment --image=nginx:1.14 --port=80 --replicas=3
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future version.
Use kubectl run --generator=run-pod/v1 or kubectl create instead.
deployment.apps/nginx-depoloyment created
[root@k8s-master ~]# kubectl get pod
NAME                          READY   STATUS    RESTARTS   AGE
nginx-deployment-5f987ff8d8-67fvz   1/1     Running   0          55s
nginx-deployment-5f987ff8d8-q7cw8   1/1     Running   0          55s
nginx-deployment-5f987ff8d8-r8srd   1/1     Running   0          55s
```

2、发布任务

容器资源创建完成，就需要完成发布工作，确保 Pod 能够对外提供服务，保证客户端能够正常访问，使用 kubectlexpose 命令可以实现该目的。kubectlexpose 的命令语法如下所示。

```
kubectl expose (-f FILENAME | TYPE NAME) [--port=port] [--protocol=TCP|UDP] [--target -
port=number-or-name] [--name=name] [--external-ip=external-ip-of-service] [--type=type]
```

上述命令语法中，各选项的作用分别如下所示。

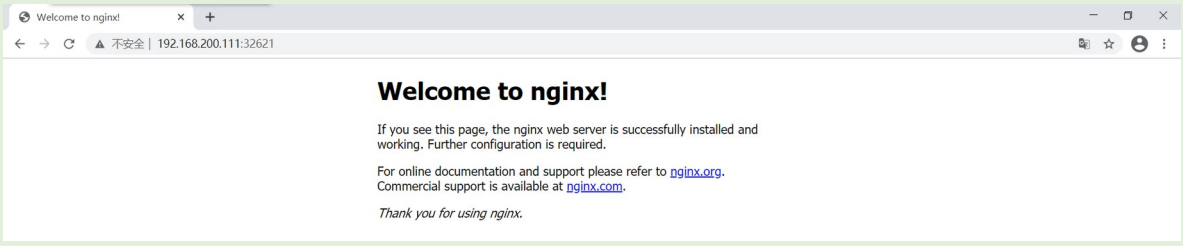
- **-f：**标识公开服务的资源的文件的文件名，目录或 URL；
- **TYPE NAME：**指定 deployment 名称；
- **port：**指定内部通信端口；
- **protocol：**指定网络协议，tcp 或者 udp；
- **target-port：**指定暴露在外部的端口；
- **name：**指定最新创建的对象的名称；
- **external-ip：**为 service 的外部 IP 地址；
- **type：**指定此服务的类型 ClusterIP，NodePort，LoadBalancer 或 ExternalName，默认 ClusterIP

通过 kubectlexpose 命令创建 Service，设置内部通信端口和外部暴露的端口均为 80、名称为 nginx-service、类型是 NodePorto 创建 Service 完后，就可以通过 kubectl get svc 命令查看到对外暴露的端口是 32621，内部通信的地址是 10.96.59.78

```
[root@k8s-master ~]# kubectl expose deployment nginx-deployment --port=80 --target-port=80 --name=nginx-service --type=NodePort
service/nginx-service exposed

[root@k8s-master ~]# kubectl get svc
NAME         TYPE       CLUSTER-IP    EXTERNAL-IP   PORT(S)      AGE
kubernetes   ClusterIP  10.96.0.1     <none>        443/TCP      4h48m
nginx-service NodePort  10.96.59.78   <none>        80:32621/TCP 35s  //对外发布端口
```

浏览器网页测试 192.168.200.111:32621



执行 kubectl get endpoints **查看容器自带的负载均衡**。从执行结果可以得知容器自带的负载均衡分别是 10.244.1.5，10.244.2.6、10.244.2.7

```
[root@k8s-master ~]# kubectl get endpoints
NAME         ENDPOINTS                                        AGE
kubernetes   192.168.200.111:6443                             4h52m
nginx-service 10.244.1.5:80,10.244.2.6:80,10.244.2.7:80       4m
```

执行 kubectl get pods 命令时加参数-o wide 可以查看到 Node 真实节点上的分布。

```
[root@k8s-master ~]# kubectl get pods -o wide
NAME                               READY   STATUS    RESTARTS   AGE    IP           NODE        NOMINATED NODE   READINESS GATES
nginx-deployment-6f8754f958-hx2wh  1/1     Running   0          5m39s  10.244.2.6   k8s-node2   <none>           <none>
nginx-deployment-6f8754f958-hx725  1/1     Running   0          5m39s  10.244.1.5   k8s-node1   <none>           <none>
nginx-deployment-6f8754f958-lpl9t  1/1     Running   0          5m39s  10.244.2.7   k8s-node2   <none>           <none>
```

完成上述步骤就已经完成了发布，并且可以进行对其访问。

3、版本更新

一般来说，生产环境中的线上项目会随着时间的推进不断地进行更新、维护、版本升级、兼容老版本。而此时如果需要对 Nginx 更换版本，就需要滚动更新。

执行以下命令查看 nginx 版本

```
[root@k8s-master ~]# kubectl get pod
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-6f8754f958-hx2wh   1/1     Running   0          7m46s
nginx-deployment-6f8754f958-hx725   1/1     Running   0          7m46s
nginx-deployment-6f8754f958-lpl9t   1/1     Running   0          7m46s
[root@k8s-master ~]# kubectl exec nginx-deployment-6f8754f958-hx2wh -- nginx -v
nginx version: nginx/1.14.2
```

通过 set 选项将 Nginx 版本换成 1.19，再监使用-w 先处于监听状态进行听，更新速度快。如果不使用 Ctrl+c 中断监听，会一直持续。

```
[root@k8s-master ~]# kubectl get deploy,pod
NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deployment   3/3     3            3           11m

NAME                                   READY   STATUS    RESTARTS   AGE
pod/nginx-deployment-6f8754f958-hx2wh   1/1     Running   0          11m
pod/nginx-deployment-6f8754f958-hx725   1/1     Running   0          11m
pod/nginx-deployment-6f8754f958-lpl9t   1/1     Running   0          11m
[root@k8s-master ~]# kubectl set image deployment.apps/nginx-deployment nginx-deployment=nginx:1.19
deployment.apps/nginx-deployment image updated
[root@k8s-master ~]# kubectl get pod        //看到名字跟之前的已经不一样了
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-76775dd78f-9jljw   1/1     Running   0          65s
nginx-deployment-76775dd78f-mpf9m   1/1     Running   0          63s
nginx-deployment-76775dd78f-zsbsw   1/1     Running   0          62s
[root@k8s-master ~]# kubectl exec nginx-deployment-76775dd78f-9jljw -- nginx -v
nginx version: nginx/1.19.6          //版本已经升级成功
```

4、版本回滚

若想回滚到上一个版本可以使用 rollout 参数

通过 history 查看历史信息

```
[root@k8s-master ~]# kubectl rollout history deployment.apps/nginx-deployment
deployment.apps/nginx-deployment
REVISION  CHANGE-CAUSE
1         <none>
2         <none>
```

通过 undo 执行回滚

```
[root@k8s-master ~]# kubectl rollout undo deployment.apps/nginx-deployment     //进行回滚
deployment.apps/nginx-deployment rolled back
```

通过 status 查看回滚状态

```
[root@k8s-master ~]# kubectl rollout status deployment.apps/nginx-deployment
deployment "nginx-deployment" successfully rolled out
```

```
[root@k8s-master ~]# kubectl get pod
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-6f8754f958-gdfl9   1/1     Running   0          104s
nginx-deployment-6f8754f958-h9dt6   1/1     Running   0          105s
nginx-deployment-6f8754f958-qxdz6   1/1     Running   0          106s
[root@k8s-master ~]# kubectl exec nginx-deployment-6f8754f958-gdfl9 -- nginx -v
nginx version: nginx/1.14.2       //版本回滚成功
```

5、删除资源

通过以上步骤，完成了创建、发布、更新、回滚操作，接下来进行删除操作。通过 kubect get deploy 查看 deployment 信息，并且使用 kubectl delete 删除 deployment；通过 kubectl get svc 查看 sevice 信息，并且使用 kubectl delete 删除 sevice。

```
[root@k8s-master ~]# kubectl get deploy
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3/3     3            3           23m
[root@k8s-master ~]# kubectl delete deployment/nginx-deployment    //删除 deployment
deployment.apps "nginx-deployment" deleted

[root@k8s-master ~]# kubectl get deploy
No resources found in default namespace.

[root@k8s-master ~]# kubectl get pod
No resources found in default namespace.

[root@k8s-master ~]# kubectl get svc
NAME            TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
kubernetes      ClusterIP   10.96.0.1      <none>        443/TCP        6h13m
nginx-service   NodePort    10.96.59.78    <none>        80:32621/TCP   85m

[root@k8s-master ~]# kubectl delete svc/nginx-service        //删除 service
```

```
service "nginx-service" deleted
[root@k8s-master ~]# kubectl get svc
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   6h15m
```

6、查看资源使用情况

通过 describe 可以查看较为详细的容器资源使用情况，包括一些错误信息都能检测出来。

```
[root@k8s-master ~]# kubectl run nginx --image=nginx1.14 --port=80 --replicas=3   //运行容器
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future version.
Use kubectl run --generator=run-pod/v1 or kubectl create instead.
deployment.apps/nginx created

[root@k8s-master ~]# kubectl get pod
NAME                     READY   STATUS    RESTARTS   AGE
nginx-59d795d786-bvndq   1/1     Running   0          8s
nginx-59d795d786-hh566   1/1     Running   0          8s
nginx-59d795d786-m42h2   1/1     Running   0          8s
[root@k8s-master ~]# kubectl describe pod nginx-59d795d786-bvndq   //查看 pod 详细信息
Name:         nginx-59d795d786-bvndq
Namespace:    default
Priority:     0
Node:         k8s-node2/192.168.200.113
Start Time:   Mon, 29 Mar 2021 23:06:10 +0800
Labels:       pod-template-hash=59d795d786
              run=nginx
Annotations:  <none>
Status:       Running
IP:           10.244.2.16
IPs:
  IP:           10.244.2.16
Controlled By:  ReplicaSet/nginx-59d795d786
Containers:
  nginx:
    Container ID:   docker://9174aca89ab6d6869a60caffd47ce8ec7d9f0c41360b9f4ee4ba175f74533a88
    Image:          nginx:1.14
                                                                    Image                       ID:
docker://sha256:295c7be079025306c4f1d65997fcf7adb411c88f139ad1d34b537164aa060369
    Port:           80/TCP
```

```
    Host Port:     0/TCP
    State:         Running
      Started:     Mon, 29 Mar 2021 23:06:11 +0800
    Ready:         True
    Restart Count:  0
    Environment:   <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-hlqtd (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-hlqtd:
    Type:       Secret (a volume populated by a Secret)
    SecretName:  default-token-hlqtd
    Optional:   false
QoS Class:      BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  66s   default-scheduler  Successfully assigned default/nginx-59d795d786-bvndq to
k8s-node2
  Normal  Pulled     65s   kubelet, k8s-node2  Container image "nginx:1.14" already present on machine
  Normal  Created    65s   kubelet, k8s-node2  Created container nginx
  Normal  Started    65s   kubelet, k8s-node2  Started container nginx
```

查看 deployment 资源

```
[root@k8s-master ~]# kubectl get deploy
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   3/3     3            3           3m32s
[root@k8s-master ~]# kubectl describe deployment/nginx     //查看 deployment 详细信息
Name:                   nginx
Namespace:              default
```

```
CreationTimestamp:      Mon, 29 Mar 2021 23:06:10 +0800
Labels:                 run=nginx
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               run=nginx
Replicas:               3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  run=nginx
  Containers:
   nginx:
    Image:        nginx:1.14
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type          Status  Reason
  ----          ------  ------
  Available     True    MinimumReplicasAvailable
  Progressing   True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   nginx-59d795d786 (3/3 replicas created)
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  2m51s  deployment-controller  Scaled up replica set nginx-59d795d786 to 3
```

7、执行容器命令

使用 exec 可以进入 pod，完成相关操作

```
[root@k8s-master ~]# kubectl get pod
NAME                READY   STATUS    RESTARTS   AGE
nginx-59d795d786-bvndq   1/1     Running   0          5m4s
nginx-59d795d786-hh566   1/1     Running   0          5m4s
nginx-59d795d786-m42h2   1/1     Running   0          5m4s
[root@k8s-master ~]# kubectl exec -it nginx-59d795d786-bvndq bash
```

```
root@nginx-59d795d786-bvndq:/# /etc/init.d/nginx status
[ ok ] nginx is running.
root@nginx-59d795d786-bvndq:/# nginx -v
nginx version: nginx/1.14.2
root@nginx-59d795d786-bvndq:/# exit
exit
```

二、yaml 资源清单

在 Kubernetes 中可以使用 YAML 格式的文件来创建符合预期期望的 Pod，这样的 YAML 文件一般称之为资源清单。

（一）yaml 语言

YAML 语言是一个可读性高，用来表达数据序列的语言格式。YAML 是"YAMLAin't a Markuplanguage"（YAML 不是一种标记语言）的递归缩写。在开发这个语言时，YAML 的意思其实是："yetAnother Markup Language"（仍是一种标记语言），但为了强调这种语言以数据做为中心，而不是以标记语言为重点，而用反向缩略语重命名。

1、基本语法

·缩进时不允许使用 Tab 键，只允许使用空格；
·缩进的空格数目不重要，只要相同层级的元素左侧对齐即可；
·#用于标识注释，从这个字符一直到行尾，都会被解释器忽略。

2、支持的数据结构

·**对象：**键值对的集合，又称为映射（mapping）/哈希（hashes）/字典（dictionary）；
·**数组：**一组按次序排列的值，又称为序列（sequence）/列表（list）；
·**纯量（scalars）：**单个的、不可再分的值字符串。包括布尔值、整数、浮点数、Null、时间、日期。

（二）通过资源清单管理容器资源

前面介绍了使用 kubectl 命令创建容器资源方法。基于这种命令方式创建容器资源，优点在于简单直观快捷、上手比较快，适合临时测试或实验。除了 kubectl 命令方式创建资源之外，还可以通过 YAML 配置文件来创建容器资源。

基于 YAML 配置文件创建容器资源的方式，优点在于配置文件提供了创建资源的模板，能够重复部署，可以像管理代码一样管理部署，适合正式的、跟环境的、规模化部署。

YAML 语法格式：

·缩进标识层级关系；
·不支持制表符（Tab 键）缩进，使用空格缩进；

·通常开头缩进两个空格；
·字符后缩进一个空格，如冒号、逗号等；
·"---"表示 YAML 格式，一个文件的开始；
·"#"表示注释。

通过 kubectl api-versions 查看以 group/version 的格式显示服务器所支持的 API 版本

```
[root@k8s-master ~]# kubectl api-versions
admissionregistration.k8s.io/v1
admissionregistration.k8s.io/v1beta1
apiextensions.k8s.io/v1
apiextensions.k8s.io/v1beta1
apiregistration.k8s.io/v1
apiregistration.k8s.io/v1beta1
apps/v1
authentication.k8s.io/v1
authentication.k8s.io/v1beta1
authorization.k8s.io/v1
authorization.k8s.io/v1beta1
autoscaling/v1
autoscaling/v2beta1
autoscaling/v2beta2
batch/v1
batch/v1beta1
certificates.k8s.io/v1beta1
coordination.k8s.io/v1
coordination.k8s.io/v1beta1
discovery.k8s.io/v1beta1
events.k8s.io/v1beta1
extensions/v1beta1
networking.k8s.io/v1
networking.k8s.io/v1beta1
node.k8s.io/v1beta1
policy/v1beta1
rbac.authorization.k8s.io/v1
rbac.authorization.k8s.io/v1beta1
scheduling.k8s.io/v1
scheduling.k8s.io/v1beta1
storage.k8s.io/v1
storage.k8s.io/v1beta1
```

v1

在创建 Deployment 资源清单之前先创建 demo 目录，用于存放资源清单的文件。在创建的 nginx-deployment.yaml 资源清单中，定义以下信息。

·kind：指定资源类型为 deployment；

·metadata：为元数据对象；

·name：指定资源 deployment 的名称；

·labels：为标签，定义标签选择器为 app:nginx，和 template 定义的遥相呼应，目的在于明确标识对象的属性。启动 Pod 的时候，通过 nodeselector 指定要调度到的 node 节点的标签；

·spec：为详细定义对象；

·replicas：定义副本数为 3；

·matchLabels：用于定义一组 Label，与直接写在 Selector 中作用相同。

·matchExpression 用于定义一组基于集合的筛选条件，可用的条件运算符包括：In.Notln Exists 和 DoesNotExist

·template：为模板，这里是 Pod 的定义。定义标签选择器为 app:nginx，容器名称为 nginx 为镜像 1.19.6，对外暴露的端口为 80。

```
[root@k8s-master ~]# cat nginx-deployment.yaml
apiVersion: apps/v1           //指定 API 版本
kind: Deployment              //指定资源类型是 deployment
metadata:                     //指定属性
 name: nginx-deployment       //指定名称
 labels:                      //定义标签
  app: nginx
spec:                         //定义详细信息
 replicas: 3                  //定义副本数量
 selector:                    //定义选择器信息
  matchLabels:
   app: nginx
 template:                    //定义模板
  metadata:
   labels:
    app: nginx
  spec:                       //定义容器信息
   containers:
   - name: nginx
    image: nginx:1.19
    ports:
    - containerPort: 80
```

```
[root@k8s-master ~]# mkdir demo
[root@k8s-master ~]# cd demo/
[root@k8s-master demo]# vim nginx-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx-deployment
 labels:
   app: nginx
spec:
 replicas: 3
 selector:
   matchLabels:
    app: nginx
 template:
   metadata:
    labels:
      app: nginx
   spec:
    containers:
    - name: nginx
      image: nginx:1.19
      ports:
      - containerPort: 80
```

创建完 Deployment 的资源清单之后，使用 create 执行资源清单来创建 pods 可以查看到 Pod 容器资源已经自动创建完成。

```
[root@k8s-master demo]# kubectl create -f nginx-deployment.yaml    //根据 yaml 文件部署容器
deployment.apps/nginx-deployment created
[root@k8s-master demo]# kubectl get pod       //部署成功
NAME                          READY   STATUS    RESTARTS   AGE
nginx-deployment-687548bb8c-9548v   1/1     Running   0          3s
nginx-deployment-687548bb8c-ctc5x   1/1     Running   0          3s
nginx-deployment-687548bb8c-t6frt   1/1     Running   0          3s
```

创建 service 资源清单

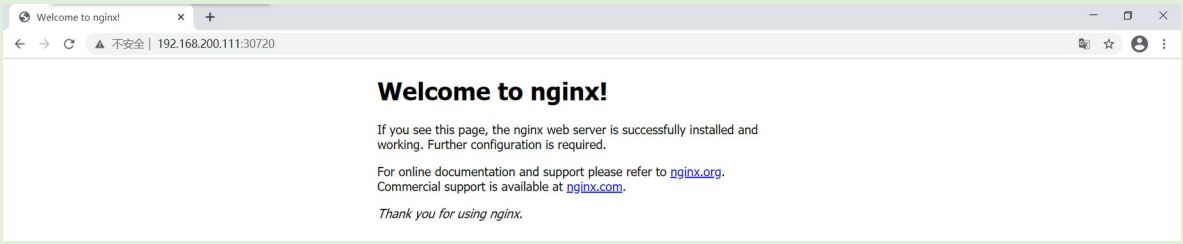
在创建的 nginx-service 资源清单中，定义名称为 nginx-service 的 Service、标签选择器为 app:nginx，type 为 NodePort 指明外部流量可以访问内部容器。在 ports 中定义暴露的端只库号烈

表，对外暴露访问的端口是 80，容器内部的端口也是 80

```
[root@k8s-master demo]# vim nginx-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
  labels:
    app: nginx
spec:
  type: NodePort
  ports:
    - port: 80
      targetPort: 80
  selector:
    app: nginx

[root@k8s-master demo]# kubectl create -f nginx-service.yaml
service/nginx-service created
[root@k8s-master demo]# kubectl get svc
NAME         TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)      AGE
kubernetes   ClusterIP  10.96.0.1      <none>        443/TCP      7h10m
nginx-service  NodePort  10.96.181.117  <none>        80:30720/TCP  10s
```

浏览器访问 192.168.200.111:30720



（三）使用 kubectl 命令创建 yaml 模板

创建 deployment 模板

```
[root@k8s-master ~]# kubectl run nginx-deployment --image=nginx:1.14 --port=80 -o yaml --dry-run > nginx-deployment.yaml
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future version. Use kubectl run --generator=run-pod/v1 or kubectl create instead.
[root@k8s-master ~]# ls      //看到生成了 yaml 文件
  nginx-deployment.yaml
```

```
[root@k8s-master ~]# vim nginx-deployment.yaml       //要进行简单的修改，或直接执行
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    run: nginx-deployment
  name: nginx-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      run: nginx-deployment
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        run: nginx-deployment
    spec:
      containers:
      - image: nginx:1.14
        name: nginx-deployment
        ports:
        - containerPort: 80
        resources: {}
status: {}

[root@k8s-master ~]# kubectl create -f nginx-deployment.yaml
[root@k8s-master ~]# kubectl get pod
NAME                          READY   STATUS    RESTARTS   AGE
nginx-deployment-687548bb8c-9548v   1/1     Running   0          25m
nginx-deployment-687548bb8c-ctc5x   1/1     Running   0          25m
nginx-deployment-687548bb8c-t6frt   1/1     Running   0          25m
```

创建 service 模板

```
[root@k8s-master ~]# kubectl expose deployment nginx-deployment --port=80 --target-port=80 --name=nginx-service --type=NodePort -o yaml --dry-run > nginx-service.yaml
```

编辑 service 模板

```
[root@k8s-master ~]# vim nginx-service.yaml
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels:
    app: nginx
  name: nginx-service
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: NodePort
status:
  loadBalancer: {}

[root@k8s-master ~]# kubectl create -f nginx-service.yaml
[root@k8s-master ~]# kubectl get svc
NAME         TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)      AGE
kubernetes   ClusterIP  10.96.0.1      <none>        443/TCP      7h28m
nginx-service  NodePort  10.96.181.117  <none>        80:30720/TCP  17m
```

浏览器访问测试 192.168.200.111:30720

直接运行 service 也是可以的，但是需要在网上拉取镜像，之前学习的 harbor 镜像仓库，可以直接将 harbor 与 k8s 进行整合，将 service 中的 image 镜像的路径改成 harbor 仓库的路径即可，从 harbor 镜像仓库中拉取镜像