Kubernetes安装部署操作手册

作 者 姓 名 ： 杨遥

项 目 组 ： 中科软

制 作 日 期 ： 2019 年 01 月 07 日

Kubernetes 安装部署操作手册

摘要

Kubernetes 单词起源于希腊语, 是“舵手”或者“领航员”的意思，是“管理者”和“控制论”的根源。 K8s 是把用 8 代替 8 个字符“ubernete”而成的缩写。首先，他是一个全新的基于容器技术的分布式架构领先方案。Kubernetes(k8s)是 Google 开源的容器集群管理系统（谷歌内部:Borg）。在 Docker 技术的基础上，为容器化的应用提供部署运行、资源调度、服务发现和动态伸缩等一系列完整功能，提高了大规模容器集群管理的便捷性。

Kubernetes 是一个完备的分布式系统支撑平台，具有完备的集群管理能力，多扩多层次的安全防护和准入机制、多租户应用支撑能力、透明的服务注册和发现机制、内建智能负载均衡器、强大的故障发现和自我修复能力、服务滚动升级和在线扩容能力、可扩展的资源自动调度机制以及多粒度的资源配额管理能力。

关键词：Kubernetes；自我修复；滚动升级；动态伸缩

目录

总页数：19 页

1. 初识 kubernetes

1.1kubernetes 简介

Kubernetes 是一个全新的基于容器技术的分布式架构领先方案。是 Google 内部集群管理系统 Borg 的一个开源版本。直到 2015 年 4 月，随着论文发布，才被众人熟知。Kubernetes 是一个开放的开发平台。不局限于任何一种语言，没有限定任何编程接口。是一个完备的分布式系统支撑平台。它构建在 docker 之上，提供应用部署、维护、扩展机制等功能，利用 Kubernetes 能方便地管理跨机器运行容器化的应用。

1.2kubernetes 特征

1. 自主的管理容器，保证云平台中的容器按照用户的期望状态运行着。
2. 自动扩容，弹性伸缩。
3. 自动监控，删除出故障的应用。
4. 容器编排成组，并提供容器间的负载均衡。
5. 自我调度，自我管理。

1.3 核心组件

kubectl:客户端命令行工具，作为整个系统的操作入口。

kube-apiserver:以 REST API 服务形式提供接口，作为整个系统的控制入口。

kube-controller-manager:执行整个系统的后台任务，包括节点状态状况、Pod 个数、Pods 和 Service 的关联等。

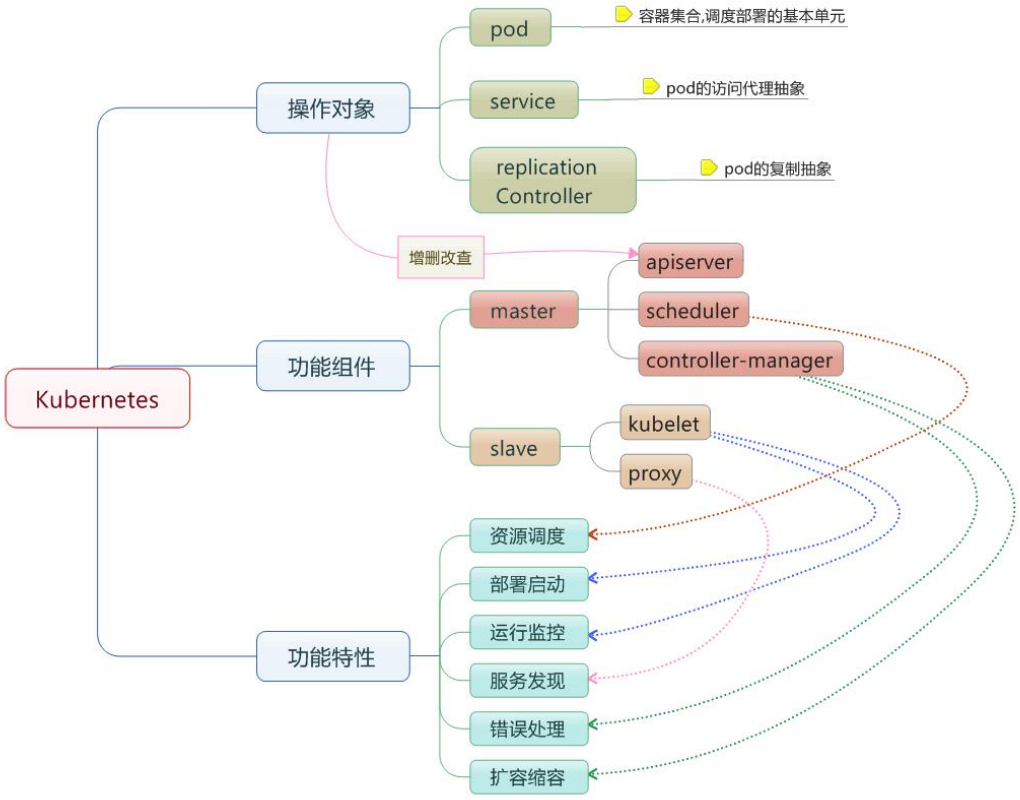
kube-scheduler:负责节点资源管理，接收来自 kube-apiserver 创建 Pods 任务，并分配到某个节点。

etcd:负责节点间的服务发现和配置共享。

kube-proxy:运行在每个计算节点上，负责 Pod 网络代理。定时从 etcd 获取到 service 信息来做相应的策略。

kubelet:运行在每个计算节点上，作为 agent，接收分配该节点的 Pods 任务及管理容器，周期性获取容器状态，反馈给 kube-apiserver。

1.4 拓扑图



2. 集群

2.1 环境介绍

192.168.2.5 k8s-master 4G 2 核

192.168.2.6 k8s-node 4G 2 核

2.2 安装必要的组件

2.2.1 安装 docker

在每个节点执行

1.添加阿里 yum 源

sudo yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

2.生成缓存

yum makecache fast

安装 docker-ce

yum -y install docker-ce

3.启动 docker 并设置为开机启动

systemctl start docker && systemctl enable docker

2.2.2 安装集群组件

每个节点执行

1. 修改 hosts 文件

vim /etc/hosts 添加以下信息

192.168.2.5 k8s-master

182.168.2.6 k8s-node

2.分别在不同主机上设置 hostname

hostnamectl --static set-hostname k8s-master

hostnamectl --static set-hostname k8s-node

3.关掉 selinux

setenforce 0

sed -i "s/^SELINUX=enforcing/SELINUX=disabled/g" /etc/sysconfig/selinux

4.关掉防火墙

systemctl stop firewalld

systemctl disable firewalld

5.关闭 swap

swapoff -a

sed -i 's/.*swap.*/#&/' /etc/fstab

6.配置转发参数

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

sysctl --system

7.配置国内 yum 源并生成缓存

cat <<EOF > /etc/yum.repos.d/kubernetes.repo [kubernetes] name=Kubernetes baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64/ enabled=1 gpgcheck=1 repo_gpgcheck=1 gpgkey=https://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg https://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg EOF

yum makecache fast

8.安装 kubeadm,kubectl,kubelet,ntpdate

yum install -y kubeadm kubectl kubelet ntpdate

systemctl start kubelet && systemctl enable kubelet

2.2 使用 kubeadm 初始化集群

在 master 上面执行一下命令，初始化集群信息

kubeadm init --image-repository registry.aliyuncs.com/google_containers --pod-network-cidr=10.244.0.0/16 --service-cidr=10.96.0.0/12

等待初始化完成后，系统提示我们执行一下三步：

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

并且初始化完成后，我们将 token 拷贝下来保存，后来加入 node 使用

kubeadm join 192.168.2.5:6443 --token 6pq12p.qolwnkesr8nu3ger --discovery-token-ca-cert-hash sha256:eb1dcad8d31fd03c4cf9ab0476c360da13b2204e01885ca8f98eafd43345ee8d

加入 flannel:

kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

注意：

这里有一个镜像拉取不下来我们手动拉取国内，并给改镜像打一个标签

docker pull registry.cn-hangzhou.aliyuncs.com/kubernetes_containers/flannel:v0.10.0-amd64

docker tag registry.cn-hangzhou.aliyuncs.com/kubernetes_containers/flannel:v0.10.0-amd64 quay.io/coreos/flannel:v0.10.0-amd64

2.3 加入集群

在 node 上面执行之前生成的 token

kubeadm join 192.168.2.5:6443 --token 6pq12p.qolwnkesr8nu3ger --discovery-token-ca-cert-hash sha256:eb1dcad8d31fd03c4cf9ab0476c360da13b2204e01885ca8f98eafd43345ee8d

3. 验证

3.1 验证集群节点

上面执行完命令后，集群不会马上变成 ready 状态，因为系统需要去下载 docker 镜像，稍等片刻后我们可以执行一下命令验证集群状态。

Kubectl get node

或者

Kubectl get node -o wide 显示更新信息

```
[root@k8s-master ~]# kubectl get node
NAME         STATUS   ROLES    AGE   VERSION
k8s-master   Ready    master   40m   v1.13.1
k8s-node     Ready    <none>   28m   v1.13.1
[root@k8s-master ~]# kubectl get node -o wide
NAME         STATUS   ROLES    AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE                KERNEL-VERSION          CONTAINER-RUNTIME
k8s-master   Ready    master   40m   v1.13.1   192.168.2.5   <none>        CentOS Linux 7 (Core)   3.10.0-693.el7.x86_64   docker://18.9.0
k8s-node     Ready    <none>   28m   v1.13.1   192.168.2.6   <none>        CentOS Linux 7 (Core)   3.10.0-693.el7.x86_64   docker://18.9.0
```

当所有节点都变成 ready 后，表示集群搭建完成。

3.2 验证 pod 信息

kubectl get pod -n kube-system

```
[root@k8s-master ~]# kubectl get pod -n kube-system
NAME                                 READY   STATUS    RESTARTS   AGE
coredns-78d4cf999f-4r49q             1/1     Running   2          32m
coredns-78d4cf999f-q9h7s             1/1     Running   2          32m
etcd-k8s-master                      1/1     Running   2          32m
kube-apiserver-k8s-master            1/1     Running   2          32m
kube-controller-manager-k8s-master   1/1     Running   2          32m
kube-flannel-ds-amd64-rgpqq          1/1     Running   4          20m
kube-flannel-ds-amd64-vqnsf          1/1     Running   3          28m
kube-proxy-46dww                     1/1     Running   1          20m
kube-proxy-dcxfl                     1/1     Running   2          32m
kube-scheduler-k8s-master            1/1     Running   2          31m
```

当所有 pod 的 ready 都变成 1/1 就表示集群搭建成功。

4. 部署

4.1 deployment

一次 deployment 为一次部署，如图 Deployment 调用 ReplicaSet 创建多个 Pod 副本。而 ReplicaSet 不需要我们去管理，所以我们只需要创建一个 deployment 即可。我们编写一个 nginx-deployment.yaml。

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
        track: stable
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
~
```

apiVersion: 组名/版本号
Kind: 类型
Metadata: 元数据，下面 name 表示当前 deployment 名称为 nginx-deployment
Spec: 规格
Replicas:3 表示副本集 3 个
Spec.template.metadata.labels 自定义标签，一般配合 selector
spec.template.containers:可以有多个容器
- name 表示容器[1]的名字
Image: 使用镜像 nginx:1.7.9
containerPort: 容器端口 80

执行 kubectl apply -f nginx-deployment.yaml 即可创建 deployment
查看 deployment:
kubectl get deploy/deployment/deployments
查看 pod：
Kubectl get pod
此时 pod 显示正在创建，因为镜像需要去 pull 下来。

```
[root@k8s-master kubernetes]# kubectl get pod
NAME                                READY   STATUS              RESTARTS   AGE
nginx-deployment-79694b8c57-bwxhs   0/1     ContainerCreating   0          97s
nginx-deployment-79694b8c57-cjg2k   0/1     ContainerCreating   0          97s
nginx-deployment-79694b8c57-ts6ht   0/1     ContainerCreating   0          97s
[root@k8s-master kubernetes]#
```

等待一段时间后，我们的 pod 已经跑起来了。

```
[root@k8s-master kubernetes]# kubectl get pod
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-79694b8c57-bwxhs   1/1     Running   0          3m13s
nginx-deployment-79694b8c57-cjg2k   1/1     Running   0          3m13s
nginx-deployment-79694b8c57-ts6ht   1/1     Running   0          3m13s
[root@k8s-master kubernetes]#
```

此时我们使用-o wide 查看 ip,此时的应用只有集群内部能访问，外部暂时不能访问。使用一个 ip 在任何一个节点都可以访问。

```
[root@k8s-master kubernetes]# kubectl get pod -o wide
NAME                                READY   STATUS    RESTARTS   AGE    IP           NODE       NOMINATED NODE   READINESS GATES
nginx-deployment-79694b8c57-bwxhs   1/1     Running   0          4m5s   10.244.1.4   k8s-node   <none>           <none>
nginx-deployment-79694b8c57-cjg2k   1/1     Running   0          4m5s   10.244.1.2   k8s-node   <none>           <none>
nginx-deployment-79694b8c57-ts6ht   1/1     Running   0          4m5s   10.244.1.3   k8s-node   <none>           <none>
[root@k8s-master kubernetes]# curl 10.244.1.4
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

4.2 Service

Service 定义了一个服务的访问入口地址，前端的应用通过这个入口地址访问其背后的一组由 Pod 副本组成的集群实例，来自外部的访问请求被负载均衡到后端的各个容器应用上。Service 与其后端 Pod 副本集群之间则是通过 Label Selector 来实现对接的.

nginx-svc.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
  labels:
    app: nginx
spec:
  type: NodePort
  ports:
  - port: 80
    targetPort: 80
  selector:
    app: nginx
```

这里我们讲一下 selector,选择的是之前 deployment，labels 定义的标签。

创建 Service:kubectl apply -f nginx-svc.yaml

查看 service:kubectl get svc/service

```
[root@k8s-master kubernetes]# kubectl get svc
NAME            TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes      ClusterIP   10.96.0.1       <none>        443/TCP        77m
nginx-service   NodePort    10.106.49.182   <none>        80:30565/TCP   7s
[root@k8s-master kubernetes]#
```

因为我们的 type 指定的是 NodePort 所以我们现在可以直接在浏览器使用任意节点 ip+30565 访问。并且我们指定的 pod 有 3 个，他每次会根据负载策略去访问三个 pod 中的任意一个。

5. 升级

5. 1 查看当前镜像版本

Kubectl get pod

Kubectl describe pod podName

看了看到 image: nginx:1.7.9

```
[root@k8s-master kubernetes]# kubectl get pod
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-79694b8c57-bwxhs   1/1     Running   0          17m
nginx-deployment-79694b8c57-cjg2k   1/1     Running   0          17m
nginx-deployment-79694b8c57-ts6ht   1/1     Running   0          17m
[root@k8s-master kubernetes]# kubectl describe pod nginx-deployment-79694b8c57-bwxhs
Name:               nginx-deployment-79694b8c57-bwxhs
Namespace:          default
Priority:           0
PriorityClassName:  <none>
Node:               k8s-node/192.168.2.6
Start Time:         Fri, 04 Jan 2019 16:52:36 +0800
Labels:             app=nginx
                    pod-template-hash=79694b8c57
                    track=stable
Annotations:        <none>
Status:             Running
IP:                 10.244.1.4
Controlled By:      ReplicaSet/nginx-deployment-79694b8c57
Containers:
  nginx:
    Container ID:   docker://de24c5c58361619ca0fe609d2994a2135cf9c9aa698302f9cebc7ef450c59955
    Image:          nginx:1.7.9
    Image ID:       docker-pullable://nginx@sha256:e3456c851a152494c3e4ff5fcc26f240206abac0c9d794affb40e0714846c451
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Fri, 04 Jan 2019 16:54:40 +0800
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-9r5ws (ro)
Conditions:
```

5. 2 更新镜像

Kubectl set image deploy/nginx-deployment nginx=nginx:1.10 --record

```
[root@k8s-master kubernetes]# kubectl set image deploy/nginx-deployment nginx=nginx:1.10 --record
deployment.extensions/nginx-deployment image updated
[root@k8s-master kubernetes]# kubectl get pod
NAME                                READY   STATUS              RESTARTS   AGE
nginx-deployment-79694b8c57-cjg2k   1/1     Running             0          22m
nginx-deployment-79694b8c57-ts6ht   1/1     Running             0          22m
nginx-deployment-86bc9bd78b-bpw6x   0/1     ContainerCreating   0          5s
nginx-deployment-86bc9bd78b-xqxpb   0/1     ContainerCreating   0          5s
[root@k8s-master kubernetes]#
```

Kubectl get pod 可以看到之前的 pod 并不会被马上终止，而是一步步新运行一个，终止一个，达到灰度发布，不重启服务器更新的目的。稍等片刻后我们就可以看到之前的 3 个 pod 全部被替换。

```
[root@k8s-master kubernetes]# kubectl get pod
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-86bc9bd78b-9twrv   1/1     Running   0          109s
nginx-deployment-86bc9bd78b-bpw6x   1/1     Running   0          3m21s
nginx-deployment-86bc9bd78b-xqxpb   1/1     Running   0          3m21s
[root@k8s-master kubernetes]# kubectl describe pod nginx-deployment-86bc9bd78b-9twrv
Name:               nginx-deployment-86bc9bd78b-9twrv
Namespace:          default
Priority:           0
PriorityClassName:  <none>
Node:               k8s-node/192.168.2.6
Start Time:         Fri, 04 Jan 2019 17:16:09 +0800
Labels:             app=nginx
                    pod-template-hash=86bc9bd78b
                    track=stable
Annotations:        <none>
Status:             Running
IP:                 10.244.1.7
Controlled By:      ReplicaSet/nginx-deployment-86bc9bd78b
Containers:
  nginx:
    Container ID:   docker://dccffaba37090bc79a52d2ece92d9df6dbc0882b358a8cce1164c13787fb7061
    Image:          nginx:1.10
    Image ID:       docker-pullable://nginx@sha256:6202beb06ea61f44179e02ca965e8e13b961d12640101fca213efbfd145d75
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Fri, 04 Jan 2019 17:16:10 +0800
    Ready:          True
    Restart Count:  0
    Environment:    <none>
```

紧着我们查看 pod 信息，运行正常，版本也成功更新到 1.10

6. 回滚

6.1 查看历史版本

Kubectl rollout history deploy/nginx-deployment

```
[root@k8s-master kubernetes]# kubectl rollout history deploy/nginx-deployment
deployment.extensions/nginx-deployment
REVISION  CHANGE-CAUSE
1         <none>
2         kubectl set image deploy/nginx-deployment nginx=nginx:1.10 --record=true

[root@k8s-master kubernetes]#
```

可以看到以上有 1 和 2 两个版本，1 表示之前的，2 表示当前的，可能有 2,3,4 那么最新的就表示当前版本。接着我们回滚一下。

6.2 回滚版本

回滚到上一个版本：

Kubectl rollout undo deploy/nginx-deployment

回滚到指定版本：

Kubectl rollout undo deploy/nginx-deployment --to-revision=2

我们回滚到上一个版本试试，可以看到我们镜像版本已经回退到之前的 1.7.9 了。也是逐个替换，滚动更新。

```
[root@k8s-master kubernetes]# kubectl rollout undo deploy/nginx-deployment
deployment.extensions/nginx-deployment rolled back
[root@k8s-master kubernetes]# kubectl get pod
NAME                                READY   STATUS        RESTARTS   AGE
nginx-deployment-79694b8c57-5sk58   1/1     Running       0          10s
nginx-deployment-79694b8c57-pcng2   1/1     Running       0          10s
nginx-deployment-79694b8c57-srskc   1/1     Running       0          7s
nginx-deployment-86bc9bd78b-bpw6x   0/1     Terminating   0          8m49s
[root@k8s-master kubernetes]# kubectl get pod
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-79694b8c57-5sk58   1/1     Running   0          20s
nginx-deployment-79694b8c57-pcng2   1/1     Running   0          20s
nginx-deployment-79694b8c57-srskc   1/1     Running   0          17s
[root@k8s-master kubernetes]# kubectl describe pod nginx-deployment-79694b8c57-5sk58
Name:               nginx-deployment-79694b8c57-5sk58
Namespace:          default
Priority:           0
PriorityClassName:  <none>
Node:               k8s-node/192.168.2.6
Start Time:         Fri, 04 Jan 2019 17:23:15 +0800
Labels:             app=nginx
                    pod-template-hash=79694b8c57
                    track=stable
Annotations:        <none>
Status:             Running
IP:                 10.244.1.8
Controlled By:      ReplicaSet/nginx-deployment-79694b8c57
Containers:
  nginx:
    Container ID:   docker://e2b12bcdcdb123d58501e71241ef7cd7402e07cf48ac8361c21db872d507cc89
    Image:          nginx:1.7.9
    Image ID:       docker-pullable://nginx@sha256:e3456c851a152494c3e4ff5fcc26f240206abac0c9d794affb40e0714846c451
    Port:           80/TCP
    Host Port:      0/TCP
```

7. Dashborad

7.1 镜像准备

因为 k8s 镜像都是需要从国外下载，我们需要先准备国内镜像。

注意：每个节点，包括 master 和 node 节点。

vim dashborad-image.sh

```
#/bin/bash
docker pull registry.cn-shanghai.aliyuncs.com/qubit/kubernetes-dashboard-amd64:v1.10.0
docker tag registry.cn-shanghai.aliyuncs.com/qubit/kubernetes-dashboard-amd64:v1.10.0 k8s.gcr.io/kubernetes-dashboard-amd64:v1.10.1
docker image rm registry.cn-shanghai.aliyuncs.com/qubit/kubernetes-dashboard-amd64:v1.10.0
chmod +x image.sh ; sh image.sh
```

7.2 编写 dashborad.yaml

```
# Copyright 2017 The Kubernetes Authors.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#        http://www.apache.org/licenses/LICENSE-2.0
#
```

```
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

# ------------------- Dashboard Secret ------------------- #

apiVersion: v1
kind: Secret
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard-certs
  namespace: kube-system
type: Opaque

---
# ------------------- Dashboard Service Account ------------------- #

apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kube-system

---
# ------------------- Dashboard Role & Role Binding ------------------- #

kind: Role
apiVersion: rbac.authorization.k8s.io/v1
```

```
    metadata:
      name: kubernetes-dashboard-minimal
      namespace: kube-system
    rules:
      # Allow Dashboard to create 'kubernetes-dashboard-key-holder' secret.
    - apiGroups: [""]
      resources: ["secrets"]
      verbs: ["create"]
      # Allow Dashboard to create 'kubernetes-dashboard-settings' config map.
    - apiGroups: [""]
      resources: ["configmaps"]
      verbs: ["create"]
      # Allow Dashboard to get, update and delete Dashboard exclusive secrets.
    - apiGroups: [""]
      resources: ["secrets"]
      resourceNames: ["kubernetes-dashboard-key-holder",
"kubernetes-dashboard-certs"]
      verbs: ["get", "update", "delete"]
      # Allow Dashboard to get and update 'kubernetes-dashboard-settings' config
map.
    - apiGroups: [""]
      resources: ["configmaps"]
      resourceNames: ["kubernetes-dashboard-settings"]
      verbs: ["get", "update"]
      # Allow Dashboard to get metrics from heapster.
    - apiGroups: [""]
      resources: ["services"]
      resourceNames: ["heapster"]
      verbs: ["proxy"]
    - apiGroups: [""]
      resources: ["services/proxy"]
      resourceNames: ["heapster", "http:heapster:", "https:heapster:"]
      verbs: ["get"]
```

```
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: kubernetes-dashboard-minimal
  namespace: kube-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: kubernetes-dashboard-minimal
subjects:
- kind: ServiceAccount
  name: kubernetes-dashboard
  namespace: kube-system

---
# ------------------- Dashboard Deployment ------------------- #

kind: Deployment
apiVersion: apps/v1
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kube-system
spec:
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      k8s-app: kubernetes-dashboard
  template:
    metadata:
      labels:
```

```
        k8s-app: kubernetes-dashboard
    spec:
      containers:
      - name: kubernetes-dashboard
        image: k8s.gcr.io/kubernetes-dashboard-amd64:v1.10.1
        ports:
        - containerPort: 8443
          protocol: TCP
        args:
          - --auto-generate-certificates
          # Uncomment the following line to manually specify Kubernetes API
server Host
          # If not specified, Dashboard will attempt to auto discover the API
server and connect
          # to it. Uncomment only if the default does not work.
          # - --apiserver-host=http://my-address:port
        volumeMounts:
        - name: kubernetes-dashboard-certs
          mountPath: /certs
          # Create on-disk volume to store exec logs
        - mountPath: /tmp
          name: tmp-volume
        livenessProbe:
          httpGet:
            scheme: HTTPS
            path: /
            port: 8443
          initialDelaySeconds: 30
          timeoutSeconds: 30
      volumes:
      - name: kubernetes-dashboard-certs
        secret:
          secretName: kubernetes-dashboard-certs
      - name: tmp-volume
```

```
        emptyDir: {}
      serviceAccountName: kubernetes-dashboard
      # Comment the following tolerations if Dashboard must not be deployed on master
      tolerations:
      - key: node-role.kubernetes.io/master
        effect: NoSchedule

---
# ------------------- Dashboard Service ------------------- #

kind: Service
apiVersion: v1
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kube-system
spec:
  ports:
    - port: 443
      targetPort: 8443
  selector:
    k8s-app: kubernetes-dashboard
```

7.3 创建和修改

创建：Kubectl create -f dashborad.yaml

因为创建出来的 service 默认是 ClusterIP,只能集群内访问，修改一下类型需要改成 NodePort。

kubectl patch svc kubernetes-dashboard -p '{"spec":{"type":"NodePort"}}' -n kube-system

查看一下 service

Kubectl get svc -n kube-system

```
[root@k8s-master kubernetes]# kubectl get svc -n kube-system
NAME                   TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)         AGE
kube-dns               ClusterIP   10.96.0.10      <none>        53/UDP,53/TCP   2d18h
kubernetes-dashboard   NodePort    10.104.86.29    <none>        443:31141/TCP   38m
[root@k8s-master kubernetes]#
```

这里我们对外暴露的端口是 31141，此时我们可以直接用 https://master_ip:31141 来访问，这个端口是随机的默认是 30000-32767

注意：我们只能用 firefox 浏览器

此时还不能直接访问，我们还需要添加一个授权管理的用户。

7.4 创建 ServiceAccount

```
vim admin-user.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: admin
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin
  namespace: kube-system
```

创建：kubectl create -f admin-user.yaml

查看 svc:kubectl describe serviceaccount admin -n kube-system

此时会有一个 tokens 的描述

查看 token:kubectl describe secret tokens 描述 -n kube-system

此时我们拷贝生成的 token。

7.5 登录 UI

此时输入 https://master_ip:31141

此时我们选择高级，添加例外，然后选择 token，将上面的 token 黏贴后即可登录。