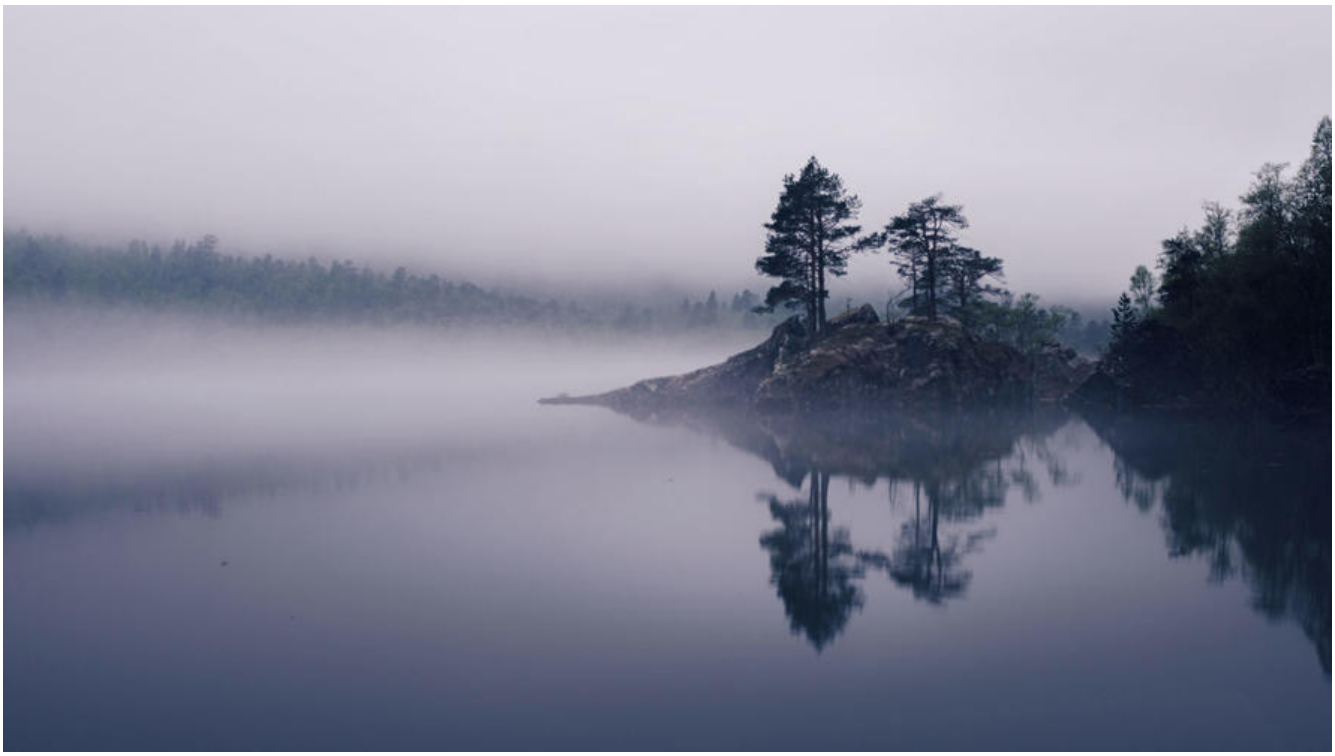
链滴

将 Kube-dns 部署到群集

作者：569707532

原文链接：https://ld246.com/article/1552618976497

来源网站：链滴



Kube-dns是一种易于使用的解决方案，用于在Kubernetes集群中提供DNS服务。本文章将指导您完在群集中安装kube-dns的过程，以及测试DNS设置以确保其正常工作。完成后，您应该在群集中安有效的kube-dns，并且pod应该能够成功使用您的DNS。

- 安装kube-dns
- 验证kube-dns pod是否正确启动
- 测试kube-dns
- 故障排错

安装kube-dns

- 准备kube-dns.yaml配置文件

```
# Copyright 2016 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.

apiVersion: v1
kind: Service
```

```
metadata:
  name: kube-dns
  namespace: kube-system
  labels:
    k8s-app: kube-dns
    kubernetes.io/cluster-service: "true"
    addonmanager.kubernetes.io/mode: Reconcile
    kubernetes.io/name: "KubeDNS"
spec:
  selector:
    k8s-app: kube-dns
  clusterIP: 10.32.0.10
  ports:
  - name: dns
    port: 53
    protocol: UDP
  - name: dns-tcp
    port: 53
    protocol: TCP
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: kube-dns
  namespace: kube-system
  labels:
    kubernetes.io/cluster-service: "true"
    addonmanager.kubernetes.io/mode: Reconcile
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: kube-dns
  namespace: kube-system
  labels:
    addonmanager.kubernetes.io/mode: EnsureExists
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: kube-dns
  namespace: kube-system
  labels:
    k8s-app: kube-dns
    kubernetes.io/cluster-service: "true"
    addonmanager.kubernetes.io/mode: Reconcile
spec:
  # replicas: not specified here:
  # 1. In order to make Addon Manager do not reconcile this replicas parameter.
  # 2. Default is 1.
  # 3. Will be tuned in real time if DNS horizontal auto-scaling is turned on.
  strategy:
    rollingUpdate:
      maxSurge: 10%
```

```
    maxUnavailable: 0
selector:
  matchLabels:
    k8s-app: kube-dns
template:
  metadata:
    labels:
      k8s-app: kube-dns
    annotations:
      scheduler.alpha.kubernetes.io/critical-pod: ''
  spec:
    tolerations:
    - key: "CriticalAddonsOnly"
      operator: "Exists"
    volumes:
    - name: kube-dns-config
      configMap:
        name: kube-dns
        optional: true
    containers:
    - name: kubedns
      image: gcr.io/google_containers/k8s-dns-kube-dns-amd64:1.14.7
      resources:
        # TODO: Set memory limits when we've profiled the container for large
        # clusters, then set request = limit to keep this container in
        # guaranteed class. Currently, this container falls into the
        # "burstable" category so the kubelet doesn't backoff from restarting it.
        limits:
          memory: 170Mi
        requests:
          cpu: 100m
          memory: 70Mi
      livenessProbe:
        httpGet:
          path: /healthcheck/kubedns
          port: 10054
          scheme: HTTP
        initialDelaySeconds: 60
        timeoutSeconds: 5
        successThreshold: 1
        failureThreshold: 5
      readinessProbe:
        httpGet:
          path: /readiness
          port: 8081
          scheme: HTTP
        # we poll on pod startup for the Kubernetes master service and
        # only setup the /readiness HTTP server once that's available.
        initialDelaySeconds: 3
        timeoutSeconds: 5
      args:
      - --domain=cluster.local.
      - --dns-port=10053
      - --config-dir=/kube-dns-config
```

```
    - --v=2
    env:
    - name: PROMETHEUS_PORT
      value: "10055"
    ports:
    - containerPort: 10053
      name: dns-local
      protocol: UDP
    - containerPort: 10053
      name: dns-tcp-local
      protocol: TCP
    - containerPort: 10055
      name: metrics
      protocol: TCP
    volumeMounts:
    - name: kube-dns-config
      mountPath: /kube-dns-config
  - name: dnsmasq
    image: gcr.io/google_containers/k8s-dns-dnsmasq-nanny-amd64:1.14.7
    livenessProbe:
      httpGet:
        path: /healthcheck/dnsmasq
        port: 10054
        scheme: HTTP
      initialDelaySeconds: 60
      timeoutSeconds: 5
      successThreshold: 1
      failureThreshold: 5
    args:
    - -v=2
    - -logtostderr
    - -configDir=/etc/k8s/dns/dnsmasq-nanny
    - -restartDnsmasq=true
    - --
    - -k
    - --cache-size=1000
    - --no-negcache
    - --log-facility=-
    - --server=/cluster.local/127.0.0.1#10053
    - --server=/in-addr.arpa/127.0.0.1#10053
    - --server=/ip6.arpa/127.0.0.1#10053
    ports:
    - containerPort: 53
      name: dns
      protocol: UDP
    - containerPort: 53
      name: dns-tcp
      protocol: TCP
    # see: https://github.com/kubernetes/kubernetes/issues/29055 for details
    resources:
      requests:
        cpu: 150m
        memory: 20Mi
    volumeMounts:
```

```
      - name: kube-dns-config
        mountPath: /etc/k8s/dns/dnsmasq-nanny
    - name: sidecar
      image: gcr.io/google_containers/k8s-dns-sidecar-amd64:1.14.7
      livenessProbe:
        httpGet:
          path: /metrics
          port: 10054
          scheme: HTTP
        initialDelaySeconds: 60
        timeoutSeconds: 5
        successThreshold: 1
        failureThreshold: 5
      args:
      - --v=2
      - --logtostderr
      - --probe=kubedns,127.0.0.1:10053,kubernetes.default.svc.cluster.local,5,SRV
      - --probe=dnsmasq,127.0.0.1:53,kubernetes.default.svc.cluster.local,5,SRV
      ports:
      - containerPort: 10054
        name: metrics
        protocol: TCP
      resources:
        requests:
          memory: 20Mi
          cpu: 10m
  dnsPolicy: Default  # Don't use cluster DNS.
  serviceAccountName: kube-dns
```

注意修改你的这个参数　clusterIP: 10.32.0.10

- 安装kube-dns

```
kubectl create -f kube-dns.yaml
```

验证kube-dns pod是否正确启动

```
kubectl get pods -l k8s-app=kube-dns -n kube-system
```

- 你应该得到显示kube-dns pod的输出。它应该看起来像这样：

```
NAME                    READY     STATUS    RESTARTS   AGE
kube-dns-598d7bf7d4-spbmj   3/3       Running   0          36s
```

确保3/3容器已准备就绪，并且pod的状态为Running,可能需要一段时间才能完全启动并运行，因此果READY首先不是3/3，请稍后再检查。

测试kube-dns

- 首先，我们需要启动一个可用于测试的pod：

```
kubectl run busybox --image=busybox:1.28 --command -- sleep 3600
```

```
POD_NAME=$(kubectl get pods -l run=busybox -o jsonpath="{.items[0].metadata.name}")
```

- 接下来，从busybox容器中运行一个nslookup

```
kubectl exec -ti $POD_NAME -- nslookup kubernetes
```

- 您应该得到如下所示的输出：

```
Server:    10.32.0.10
Address 1: 10.32.0.10 kube-dns.kube-system.svc.cluster.local
Name:      kubernetes
Address 1: 10.32.0.1 kubernetes.default.svc.cluster.local
```

- 如果nslookup成功，那么你的kube-dns安装正在运行！

故障排错

- 我在做最后一步测试的时候，nslookup的时候，报错了，解析失败了

```
[root@k8s-master example]# kubectl exec -ti busybox -- nslookup kubernetes.default
Server:    10.254.0.2
Address 1: 10.254.0.2

nslookup: can't resolve 'kubernetes.default'
command terminated with exit code 1
```

查看kube-dns的相关信息，找出了错误的所在，实在粗心大意

```
[root@k8s-master example]# kubectl exec busybox cat /etc/resolv.conf
nameserver 10.254.0.11
search default.svc.cluster.local svc.cluster.local cluster.local nhcdru4itrfetijnzscjvd1jna.ax.inter
al.chinacloudapp.cn
options ndots:5
[root@k8s-master example]# kubectl get pod -n kube-system
NAME                       READY     STATUS    RESTARTS   AGE
kube-dns-6db447cbfc-2zmrr   3/3       Running   0          23m
[root@k8s-master example]# kubectl get ep kube-dns --namespace=kube-system
NAME       ENDPOINTS                     AGE
kube-dns   172.17.0.2:53,172.17.0.2:53   25m
[root@k8s-master example]# kubectl get svc --namespace=kube-system
NAME       TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)         AGE
kube-dns   ClusterIP   10.254.0.2   <none>        53/UDP,53/TCP   25m
```

- 发现容器里面指定的nameserver为10.254.0.11，但是我的dns服务的ip地址是10.254.0.2，这样不致肯定解析失败了。最终通过修改kube-dns.yaml，中的clusterIP: 参数，修改为10.254.0.11，并重构建了kube-dns服务，最终顺利解析。