Lesson 05 Demo 02

Configuring the DNS for Kubernetes Services and Pods

Objective: To configure the domain name system (DNS) for Kubernetes services and pods to ensure proper network resolution and connectivity

Tools required: kubeadm, kubectl, kubelet, and containerd

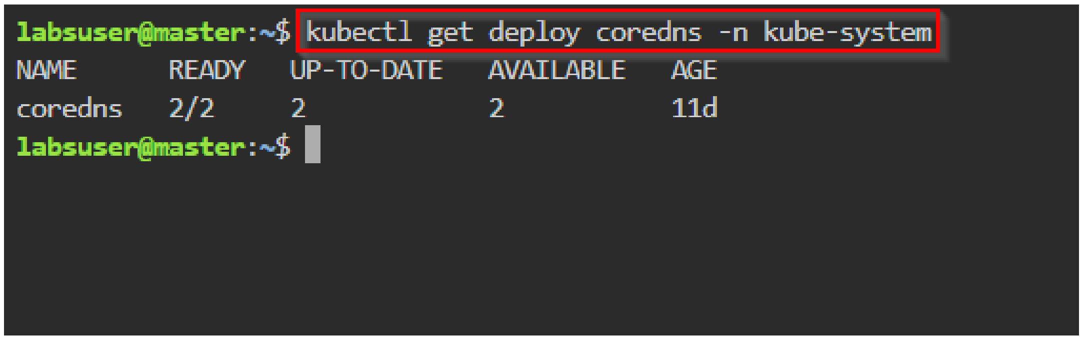
Prerequisites: A Kubernetes cluster (refer to Demo 01 from Lesson 01 for setting up a cluster)

Steps to be followed:

1. Determine the default DNS in the cluster
2. Execute the DNS query
3. Configure the DNS policy
4. Create a custom DNS configuration

Step 1: Determine the default DNS in the cluster

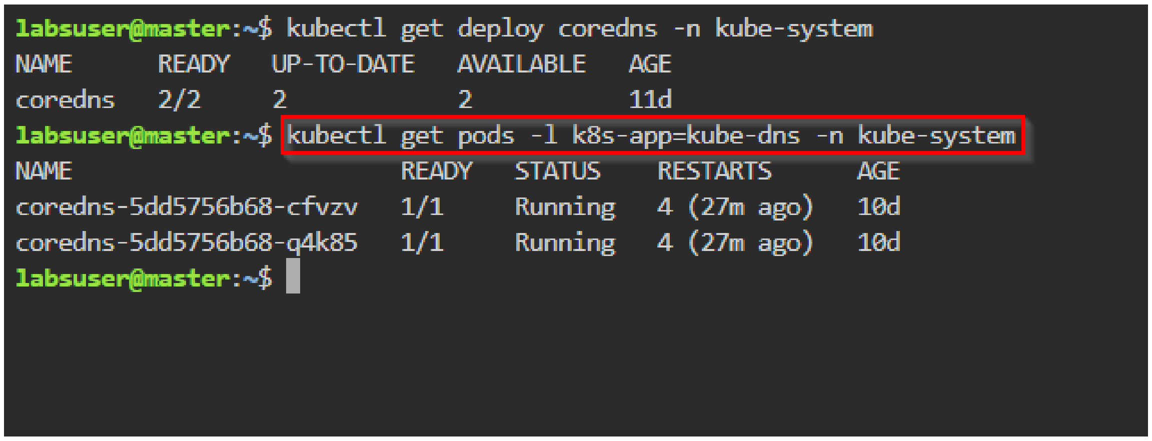
1.1 To identify the core DNS deployment, execute the following command:
kubectl get deploy coredns -n kube-system



```
labsuser@master:~$ kubectl get deploy coredns -n kube-system
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
coredns   2/2     2            2           11d
labsuser@master:~$
```

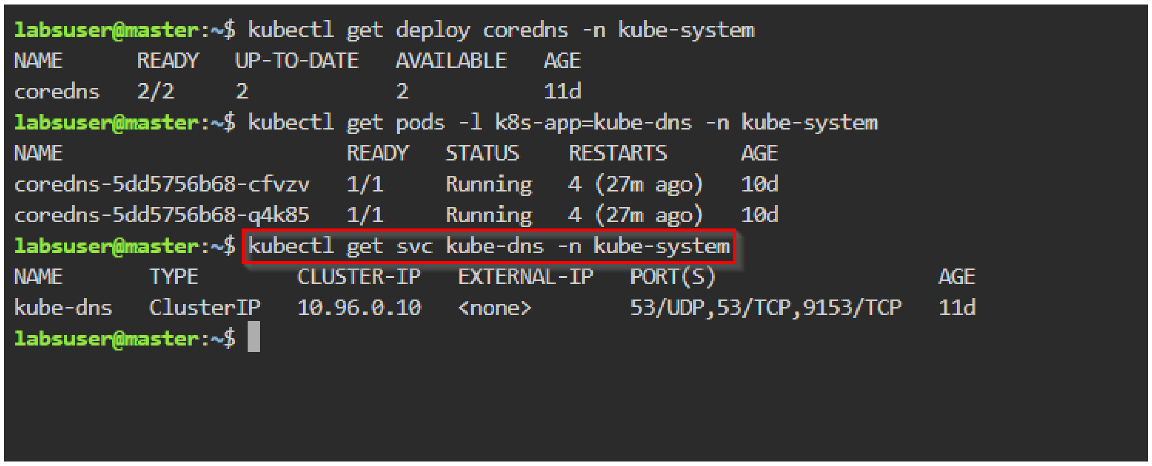
Kubernetes creates a default DNS in the **kube-system** namespace.

1.2 To identify the **coredns** pods using the selector, run the following command:
kubectl get pods -l k8s-app=kube-dns -n kube-system



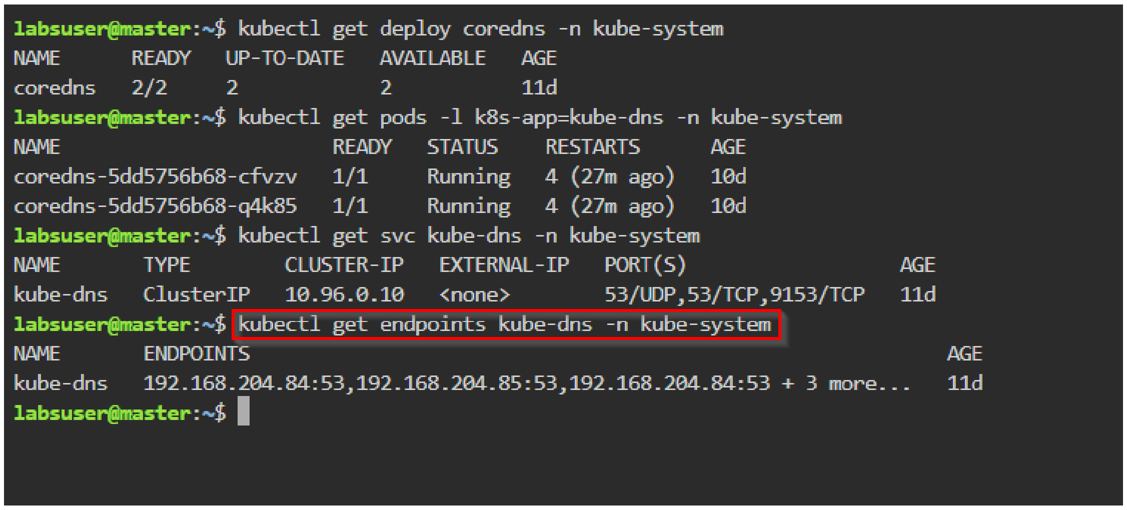
```
labsuser@master:~$ kubectl get deploy coredns -n kube-system
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
coredns   2/2     2            2           11d
labsuser@master:~$ kubectl get pods -l k8s-app=kube-dns -n kube-system
NAME                       READY   STATUS    RESTARTS      AGE
coredns-5dd5756b68-cfvzv   1/1     Running   4 (27m ago)   10d
coredns-5dd5756b68-q4k85   1/1     Running   4 (27m ago)   10d
labsuser@master:~$
```

1.3 To identify the **coredns** service, execute the following command:
kubectl get svc kube-dns -n kube-system



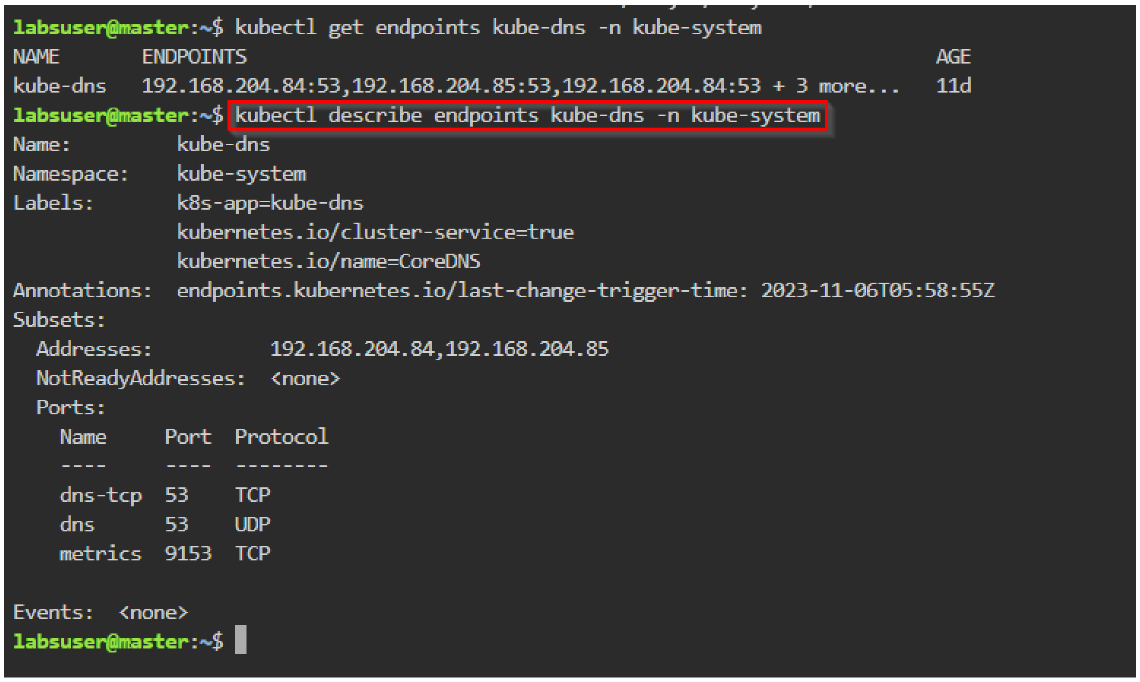
```
labsuser@master:~$ kubectl get deploy coredns -n kube-system
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
coredns   2/2     2            2           11d
labsuser@master:~$ kubectl get pods -l k8s-app=kube-dns -n kube-system
NAME                       READY   STATUS    RESTARTS      AGE
coredns-5dd5756b68-cfvzv   1/1     Running   4 (27m ago)   10d
coredns-5dd5756b68-q4k85   1/1     Running   4 (27m ago)   10d
labsuser@master:~$ kubectl get svc kube-dns -n kube-system
NAME       TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)                  AGE
kube-dns   ClusterIP   10.96.0.10   <none>        53/UDP,53/TCP,9153/TCP   11d
labsuser@master:~$
```

1.4 Use the following command to get the service endpoints:
kubectl get endpoints kube-dns -n kube-system



```
labsuser@master:~$ kubectl get deploy coredns -n kube-system
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
coredns   2/2     2            2           11d
labsuser@master:~$ kubectl get pods -l k8s-app=kube-dns -n kube-system
NAME                       READY   STATUS    RESTARTS      AGE
coredns-5dd5756b68-cfvzv   1/1     Running   4 (27m ago)   10d
coredns-5dd5756b68-q4k85   1/1     Running   4 (27m ago)   10d
labsuser@master:~$ kubectl get svc kube-dns -n kube-system
NAME       TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)                  AGE
kube-dns   ClusterIP   10.96.0.10   <none>        53/UDP,53/TCP,9153/TCP   11d
labsuser@master:~$ kubectl get endpoints kube-dns -n kube-system
NAME       ENDPOINTS                                                     AGE
kube-dns   192.168.204.84:53,192.168.204.85:53,192.168.204.84:53 + 3 more...   11d
labsuser@master:~$
```

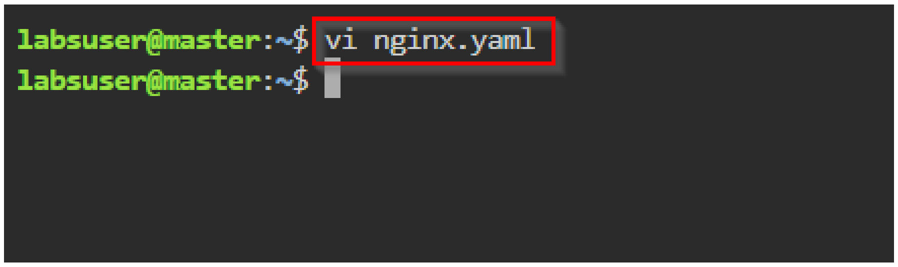
1.5 Run the following command to describe the endpoints:
kubectl describe endpoints kube-dns -n kube-system



```
labsuser@master:~$ kubectl get endpoints kube-dns -n kube-system
NAME       ENDPOINTS                                                     AGE
kube-dns   192.168.204.84:53,192.168.204.85:53,192.168.204.84:53 + 3 more...   11d
labsuser@master:~$ kubectl describe endpoints kube-dns -n kube-system
Name:         kube-dns
Namespace:    kube-system
Labels:       k8s-app=kube-dns
              kubernetes.io/cluster-service=true
              kubernetes.io/name=CoreDNS
Annotations:  endpoints.kubernetes.io/last-change-trigger-time: 2023-11-06T05:58:55Z
Subsets:
  Addresses:          192.168.204.84,192.168.204.85
  NotReadyAddresses:  <none>
  Ports:
    Name     Port  Protocol
    ----     ----  --------
    dns-tcp  53    TCP
    dns      53    UDP
    metrics  9153  TCP

Events:  <none>
labsuser@master:~$
```

Step 2: Execute the DNS query

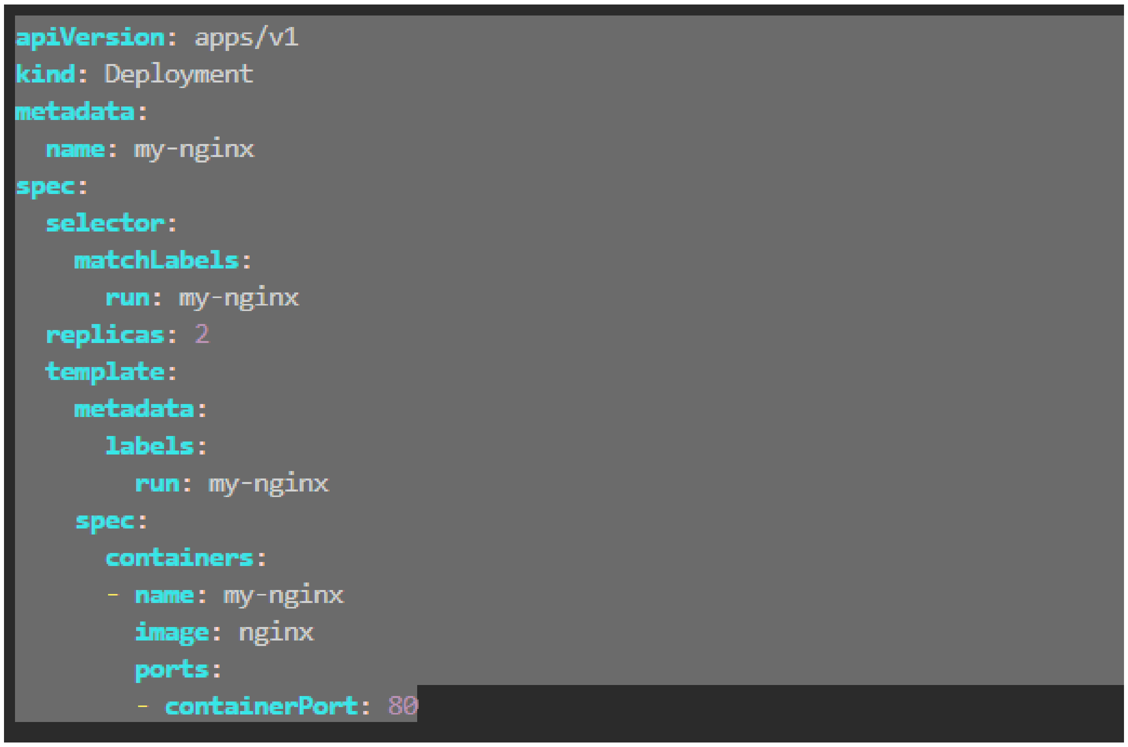
2.1 Execute the following command to create an nginx deployment file:
vi nginx.yaml



```
labsuser@master:~$ vi nginx.yaml
labsuser@master:~$
```

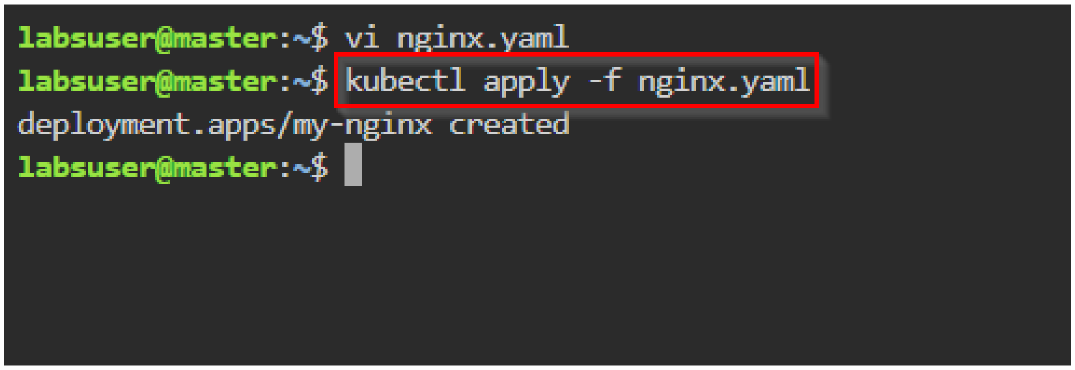
2.2 Enter the following YAML code in the **nginx.yaml** file to define a Kubernetes deployment with two replicas, each running an nginx container on port 80:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-nginx
spec:
  selector:
    matchLabels:
      run: my-nginx
  replicas: 2
  template:
    metadata:
      labels:
        run: my-nginx
    spec:
      containers:
      - name: my-nginx
        image: nginx
        ports:
        - containerPort: 80
```



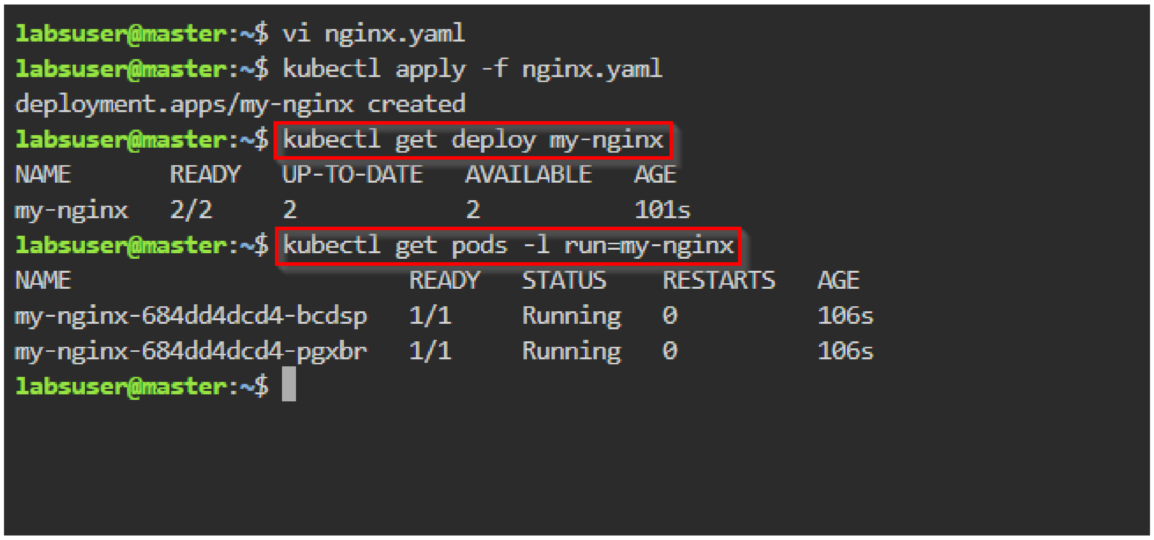
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-nginx
spec:
  selector:
    matchLabels:
      run: my-nginx
  replicas: 2
  template:
    metadata:
      labels:
        run: my-nginx
    spec:
      containers:
      - name: my-nginx
        image: nginx
        ports:
        - containerPort: 80
```

2.3 Run the following command to apply the configuration specified in the **nginx.yaml** file to create Kubernetes resources:
kubectl apply -f nginx.yaml



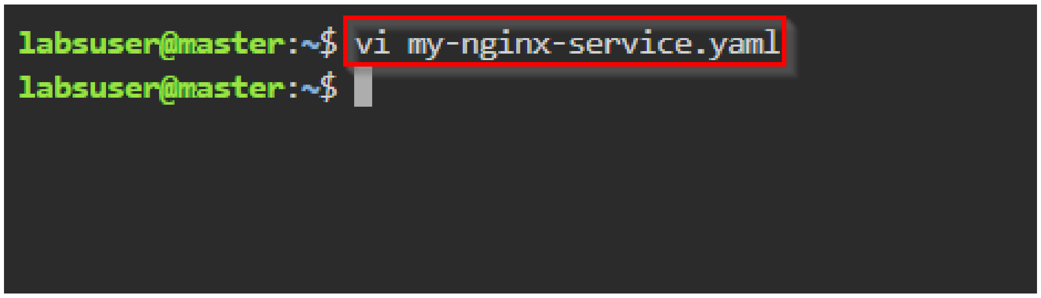
```
labsuser@master:~$ vi nginx.yaml
labsuser@master:~$ kubectl apply -f nginx.yaml
deployment.apps/my-nginx created
labsuser@master:~$
```

2.4 Run the following commands to get the status of the **my-nginx** deployment and list all pods with the **run=my-nginx** label:
kubectl get deploy my-nginx
kubectl get pods -l run=my-nginx



```
labsuser@master:~$ vi nginx.yaml
labsuser@master:~$ kubectl apply -f nginx.yaml
deployment.apps/my-nginx created
labsuser@master:~$ kubectl get deploy my-nginx
NAME       READY   UP-TO-DATE   AVAILABLE   AGE
my-nginx   2/2     2            2           101s
labsuser@master:~$ kubectl get pods -l run=my-nginx
NAME                        READY   STATUS    RESTARTS   AGE
my-nginx-684dd4dcd4-bcdsp   1/1     Running   0          106s
my-nginx-684dd4dcd4-pgxbr   1/1     Running   0          106s
labsuser@master:~$
```

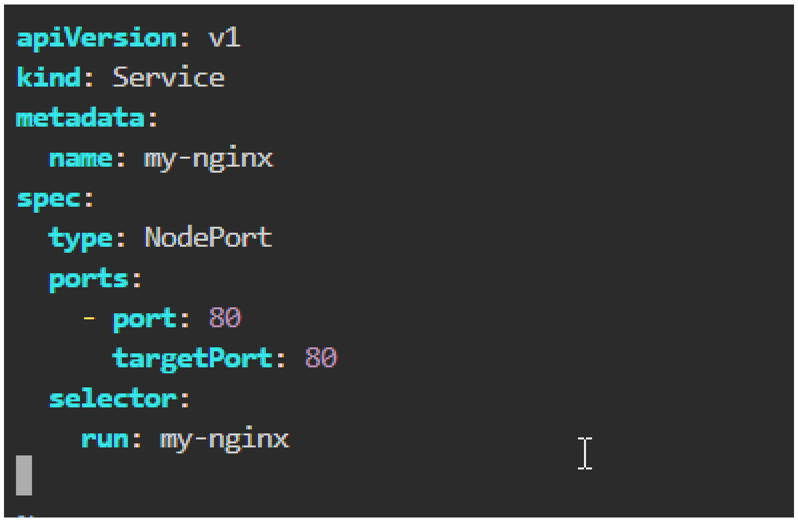
2.5 Enter the following command to create the **my-nginx-service.yaml** file:
vi my-nginx-service.yaml



```
labsuser@master:~$ vi my-nginx-service.yaml
labsuser@master:~$
```

2.6 Add the following code to the YAML file:

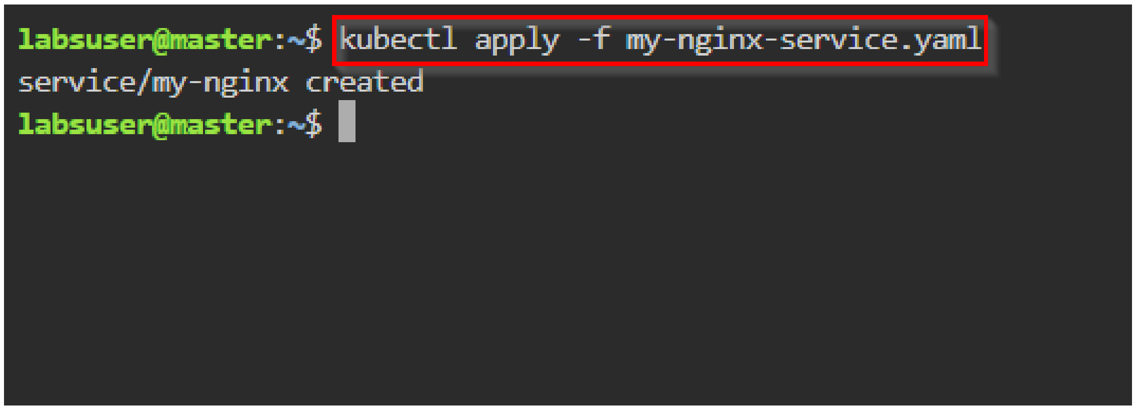
```
apiVersion: v1
kind: Service
metadata:
  name: my-nginx
spec:
  type: NodePort
  ports:
    - port: 80
      targetPort: 80
  selector:
    run: my-nginx
```



```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-nginx
spec:
  type: NodePort
  ports:
    - port: 80
      targetPort: 80
  selector:
    run: my-nginx
```

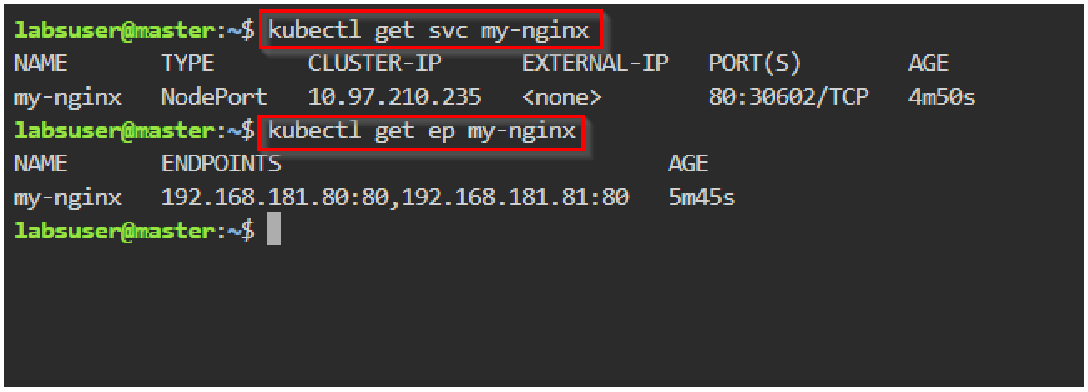
This YAML configuration defines a Kubernetes service named **my-nginx** of the type **NodePort**. It exposes port 80 and directs traffic to it on pods labeled with **run: my-nginx.**

2.7 Run the following command to apply the configurations of the **my-nginx-service.yaml** file:
kubectl apply -f my-nginx-service.yaml



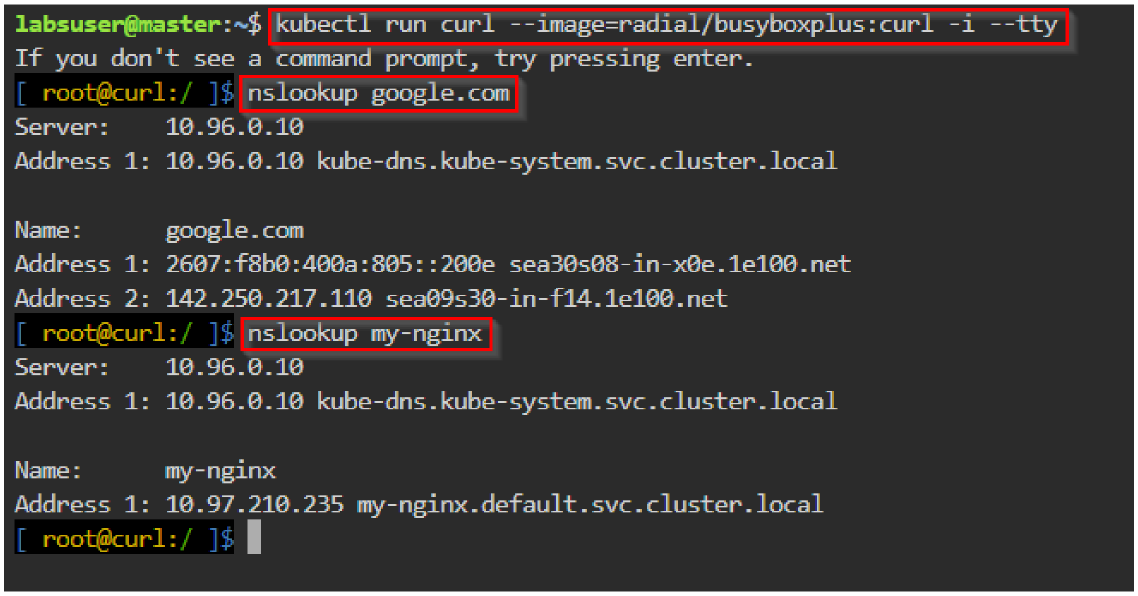
```
labsuser@master:~$ kubectl apply -f my-nginx-service.yaml
service/my-nginx created
labsuser@master:~$
```

2.8 Run the following commands to retrieve the status and details of the **my-nginx** service and its associated endpoints:
kubectl get svc my-nginx
kubectl get ep my-nginx



```
labsuser@master:~$ kubectl get svc my-nginx
NAME       TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
my-nginx   NodePort   10.97.210.235   <none>        80:30602/TCP   4m50s
labsuser@master:~$ kubectl get ep my-nginx
NAME       ENDPOINTS                             AGE
my-nginx   192.168.181.80:80,192.168.181.81:80   5m45s
labsuser@master:~$
```

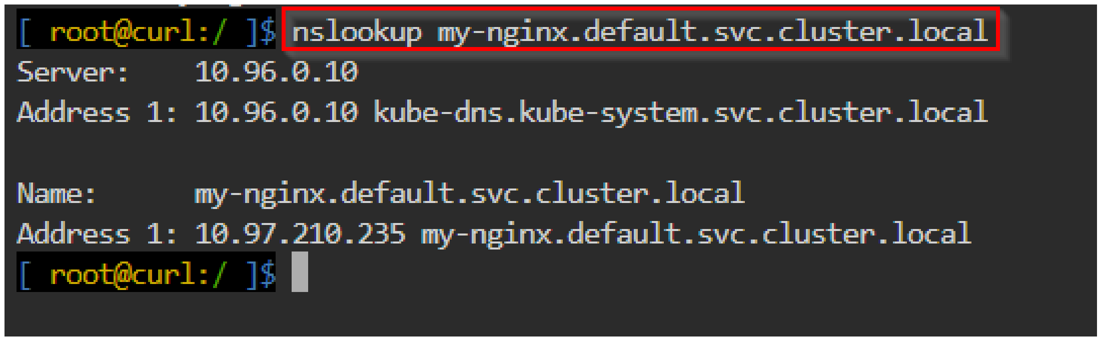
2.9 To create a curl pod to perform a DNS query, run the following commands:
kubectl run curl --image=radial/busyboxplus:curl -i –tty
nslookup google.com
nslookup my-nginx



```
labsuser@master:~$ kubectl run curl --image=radial/busyboxplus:curl -i --tty
If you don't see a command prompt, try pressing enter.
[ root@curl:/ ]$ nslookup google.com
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      google.com
Address 1: 2607:f8b0:400a:805::200e sea30s08-in-x0e.1e100.net
Address 2: 142.250.217.110 sea09s30-in-f14.1e100.net
[ root@curl:/ ]$ nslookup my-nginx
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      my-nginx
Address 1: 10.97.210.235 my-nginx.default.svc.cluster.local
[ root@curl:/ ]$
```

Note: Create a pod using the **radial/busyboxplus:curl** image. This image has network tools pre-installed, which helps perform DNS queries.

2.10 Run the following command to create a local cluster:
nslookup my-nginx.default.svc.cluster.local

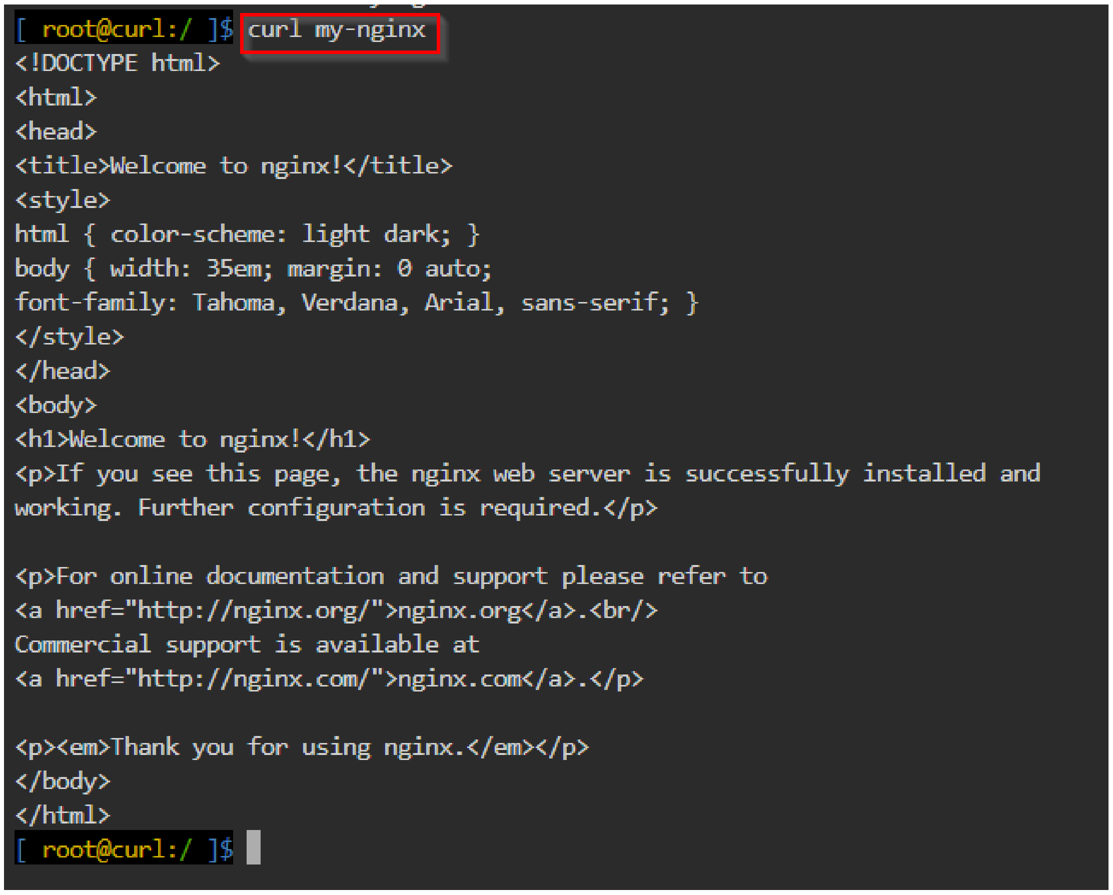


```
[ root@curl:/ ]$ nslookup my-nginx.default.svc.cluster.local
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      my-nginx.default.svc.cluster.local
Address 1: 10.97.210.235 my-nginx.default.svc.cluster.local
[ root@curl:/ ]$
```

From this curl pod, you can access the **my-nginx** service.

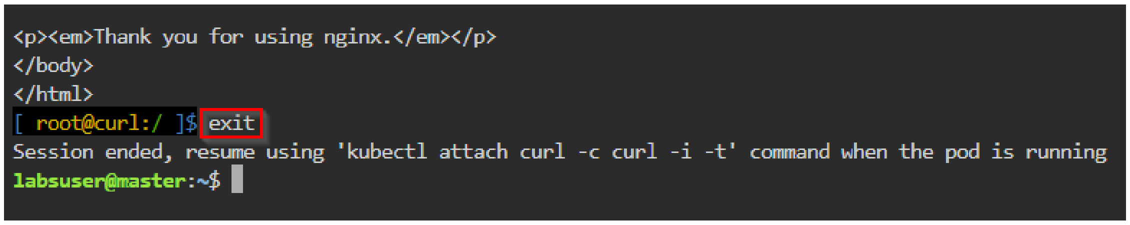
Note: Use this format to run the local cluster:
<service-name>.<namespace>.svc.cluster.local.

2.11 Run the following command to access the **my-nginx** file:
curl my-nginx



```
[ root@curl:/ ]$ curl my-nginx
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
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>
[ root@curl:/ ]$
```

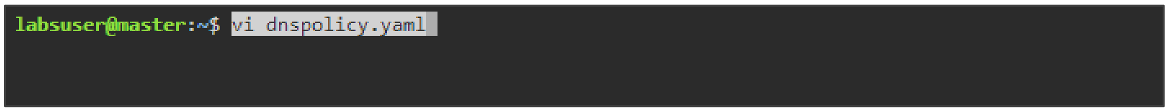
2.12 Enter the following command to exit the root directory:
exit



```
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
[ root@curl:/ ]$ exit
Session ended, resume using 'kubectl attach curl -c curl -i -t' command when the pod is running
labsuser@master:~$
```

Step 3: Configure the DNS policy

3.1 In the master node, create a configuration file that defines the DNS policy for a Kubernetes pod using the following command:
vi dnspolicy.yaml

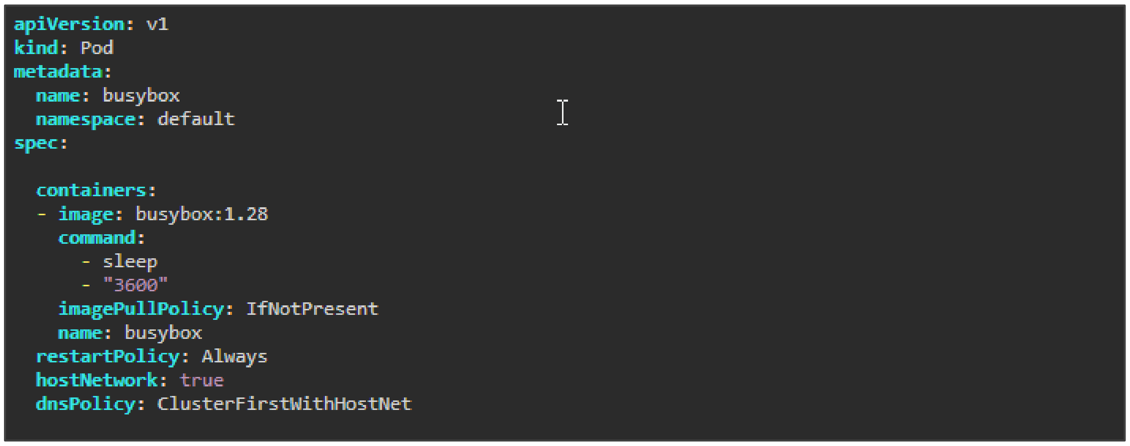


```
labsuser@master:~$ vi dnspolicy.yaml
```

The YAML file will be created and opened in the **vi** editor.

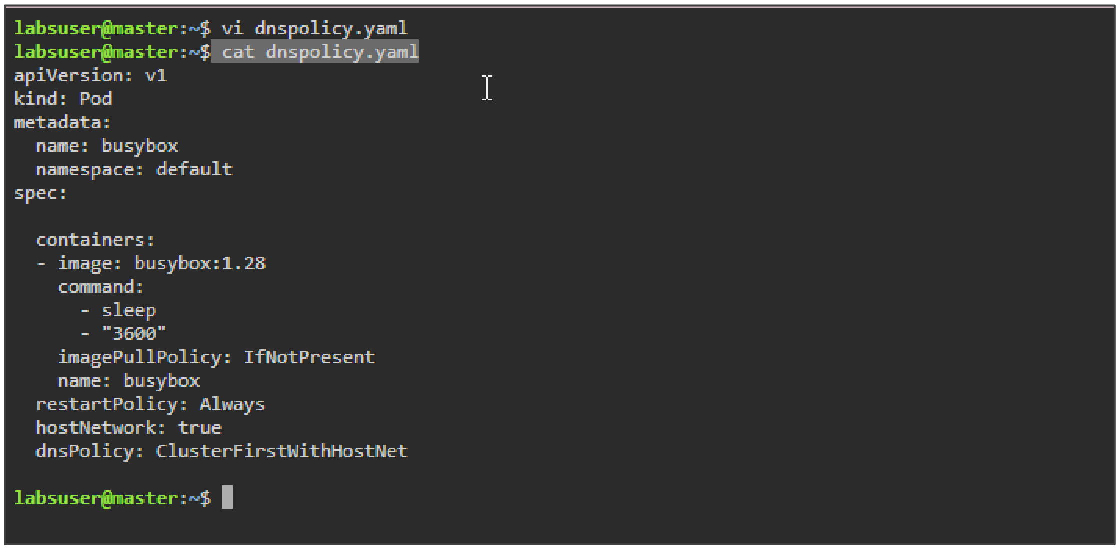
3.2 Add the following YAML code inside the **dnspolicy.yaml** file to configure the DNS policy:

```
apiVersion: v1
kind: Pod
metadata:
  name: busybox
  namespace: default
spec:
  containers:
  - image: busybox:1.28
    command:
      - sleep
      - "3600"
    imagePullPolicy: IfNotPresent
    name: busybox
  restartPolicy: Always
  hostNetwork: true
  dnsPolicy: ClusterFirstWithHostNet
```



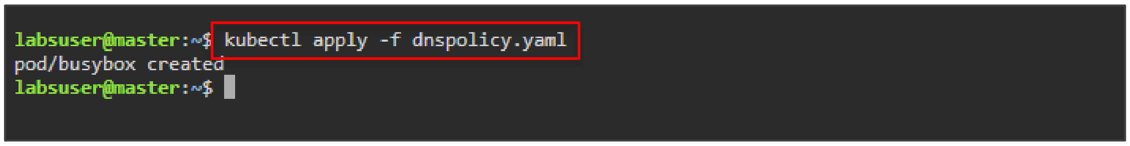
```
apiVersion: v1
kind: Pod
metadata:
  name: busybox
  namespace: default
spec:

  containers:
  - image: busybox:1.28
    command:
      - sleep
      - "3600"
    imagePullPolicy: IfNotPresent
    name: busybox
  restartPolicy: Always
  hostNetwork: true
  dnsPolicy: ClusterFirstWithHostNet
```

3.3 View the content of the **dnspolicy.yaml** file using the following command:
cat dnspolicy.yaml



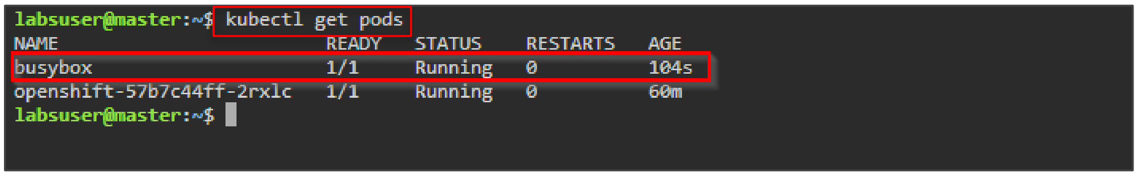
```
labsuser@master:~$ vi dnspolicy.yaml
labsuser@master:~$ cat dnspolicy.yaml
apiVersion: v1
kind: Pod
metadata:
  name: busybox
  namespace: default
spec:

  containers:
  - image: busybox:1.28
    command:
      - sleep
      - "3600"
    imagePullPolicy: IfNotPresent
    name: busybox
  restartPolicy: Always
  hostNetwork: true
  dnsPolicy: ClusterFirstWithHostNet

labsuser@master:~$
```

3.4 Create a Kubernetes pod using the following command:
kubectl apply -f dnspolicy.yaml



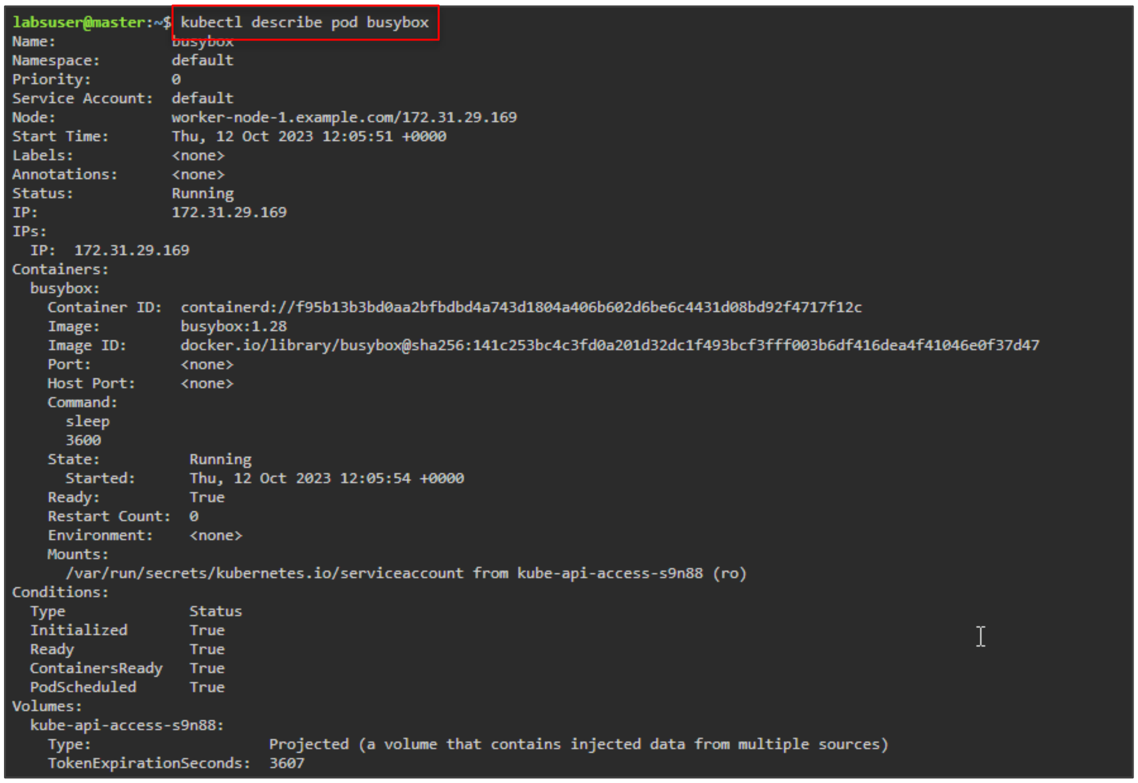
```
labsuser@master:~$ kubectl apply -f dnspolicy.yaml
pod/busybox created
labsuser@master:~$
```

3.5 Execute the following command to list the newly created pod:
kubectl get pods

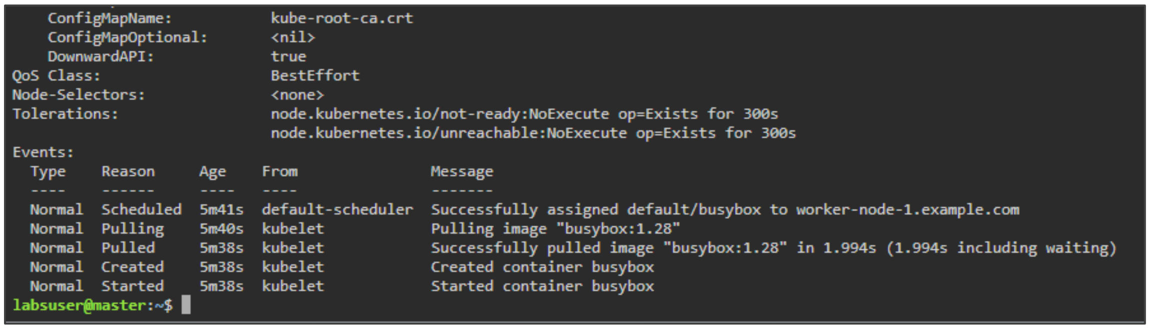


```
labsuser@master:~$ kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
busybox                     1/1     Running   0          104s
openshift-57b7c44ff-2rxlc   1/1     Running   0          60m
labsuser@master:~$
```

3.6 Execute the following command to list all the details regarding the **busybox** pod:
kubectl describe pod busybox



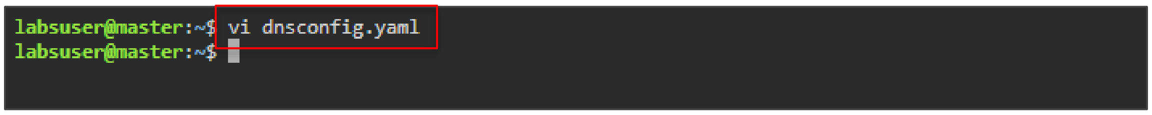
```
labsuser@master:~$ kubectl describe pod busybox
Name:             busybox
Namespace:        default
Priority:         0
Service Account:  default
Node:             worker-node-1.example.com/172.31.29.169
Start Time:       Thu, 12 Oct 2023 12:05:51 +0000
Labels:           <none>
Annotations:      <none>
Status:           Running
IP:               172.31.29.169
IPs:
  IP:  172.31.29.169
Containers:
  busybox:
    Container ID:  containerd://f95b13b3bd0aa2bfbdbd4a743d1804a406b602d6be6c4431d08bd92f4717f12c
    Image:         busybox:1.28
    Image ID:      docker.io/library/busybox@sha256:141c253bc4c3fd0a201d32dc1f493bcf3fff003b6df416dea4f41046e0f37d47
    Port:          <none>
    Host Port:     <none>
    Command:
      sleep
      3600
    State:          Running
      Started:      Thu, 12 Oct 2023 12:05:54 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-s9n88 (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  kube-api-access-s9n88:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
```



```
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age    From               Message
  ----    ------     ----   ----               -------
  Normal  Scheduled  5m41s  default-scheduler  Successfully assigned default/busybox to worker-node-1.example.com
  Normal  Pulling    5m40s  kubelet            Pulling image "busybox:1.28"
  Normal  Pulled     5m38s  kubelet            Successfully pulled image "busybox:1.28" in 1.994s (1.994s including waiting)
  Normal  Created    5m38s  kubelet            Created container busybox
  Normal  Started    5m38s  kubelet            Started container busybox
labsuser@master:~$
```

Step 4: Create a custom DNS configuration

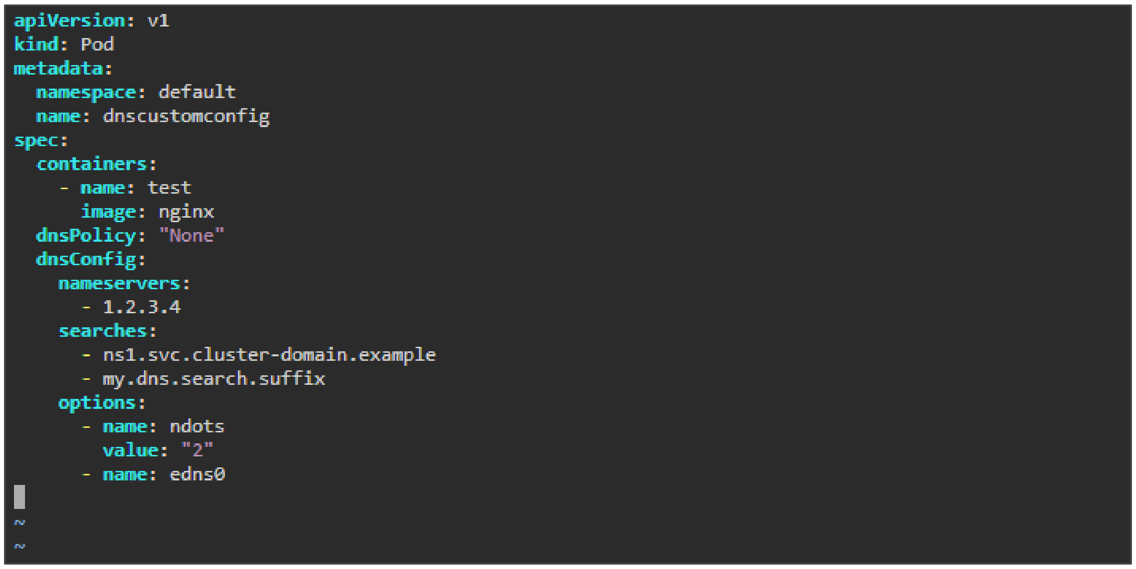
4.1 Create a DNS configuration YAML file using the following command:
vi dnsconfig.yaml



```
labsuser@master:~$ vi dnsconfig.yaml
labsuser@master:~$ 
```

4.2 Add the following YAML code in the **dnsconfig.yaml** file:

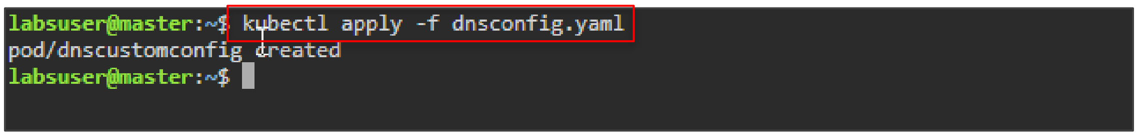
```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dnscustomconfig
spec:
  containers:
    - name: test
      image: nginx
  dnsPolicy: "None"
  dnsConfig:
    nameservers:
      - 1.2.3.4
    searches:
      - ns1.svc.cluster-domain.example
      - my.dns.search.suffix
    options:
      - name: ndots
        value: "2"
      - name: edns0
```



```
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dnscustomconfig
spec:
  containers:
    - name: test
      image: nginx
  dnsPolicy: "None"
  dnsConfig:
    nameservers:
      - 1.2.3.4
    searches:
      - ns1.svc.cluster-domain.example
      - my.dns.search.suffix
    options:
      - name: ndots
        value: "2"
      - name: edns0

~
~
```

4.3 Create another pod using the following command:

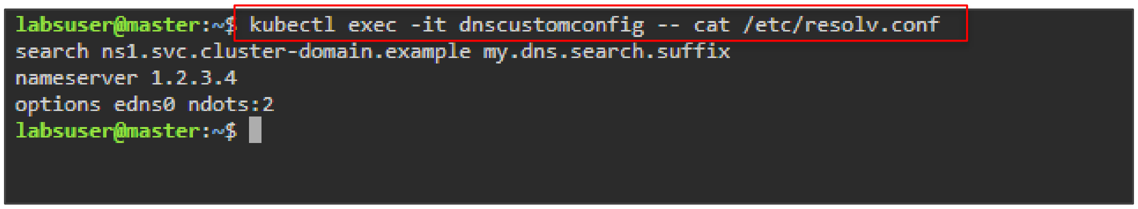
kubectl apply -f dnsconfig.yaml



```
labsuser@master:~$ kubectl apply -f dnsconfig.yaml
pod/dnscustomconfig created
labsuser@master:~$
```

4.4 Set up the IPv6 for the DNS connectivity using the following command:

kubectl exec -it dnscustomconfig -- cat /etc/resolv.conf



```
labsuser@master:~$ kubectl exec -it dnscustomconfig -- cat /etc/resolv.conf
search ns1.svc.cluster-domain.example my.dns.search.suffix
nameserver 1.2.3.4
options edns0 ndots:2
labsuser@master:~$
```

By following these steps, you have successfully configured the DNS for Kubernetes services and pods, ensuring efficient network resolution and seamless connectivity.