# Cloud Provider:AWS
# Server:Ubuntu 18.04 x6
# Server List: 2 Controller(Master), 2(Worker), 1 Nginx(Reverse Proxy), 1 Ubuntu 18.04(local)
# Deploying Kube-dns to the Cluster

**// Kube-dns is an easy-to-use solution for providing DNS service in a Kubernetes cluster. This lesson guides you through the process of installing kube-dns in your cluster, as well as testing your DNS set up to make sure that it is working. After completing this lesson, you should have a working kube-dns installation in your cluster, and pods should be able to successfully use your DNS.**

**// To install and test kube-dns, you will need to use kubectl. To connect with kubectl, you can either log into one of the control nodes and run kubectl there, or open an SSH tunnel for port 6443 to the load balancer server and use kubectl locally.**

**// You can open an SSH tunnel by running this in a separate terminal. Leave the session open while you are working to keep the tunnel active.**
$ ssh -L 6443:localhost:6443 ubuntu@<your load balancer cloud server public IP>

// You can install kube-dns like so:
$ kubectl create -f https://storage.googleapis.com/kubernetes-the-hard-way/kube-dns.yaml

// Verify that the kube-dns pod starts up correctly:
$ kubectl get pods -l k8s-app=kube-dns -n kube-system

**// You should get an output showing the kube-dns pod. It should look something like this:**

```
ubuntu@ip-10-100-0-97:~$ kubectl get pods -l k8s-app=kube-dns -n kube-system
NAME                        READY     STATUS    RESTARTS   AGE
kube-dns-598d7bf7d4-pd22k   3/3       Running   1          42m
```

**// Make sure that 3/3 containers are ready, and that the pod has a status of running. It may take a moment for the pods to be fully up and running, so if READY is not 3/3 at first, check again after a few moments.**

**// Now let's test out kube-dns installation by doing a DNS lookup from within a pod. First, we need to start up a pod that we can use for testing:**
$ kubectl run busybox --image=busybox:1.28 --command -- sleep 3600
$ POD_NAME=$(kubectl get pods -l run=busybox -o jsonpath="{.items[0].metadata.name}")

**// Next, run an nslookup from inside the busybox container:**
$ kubectl exec -ti $POD_NAME -- nslookup kubernetes

**// You should get an output that looks something like this**

```
ubuntu@ip-10-100-0-97:~$ kubectl exec $POD_NAME -- nslookup kubernetes
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
```

**// If nslookup succeeds, then your kube-dns installation is working.**

**// Once you are done, it's probably a good idea to clean up the objects that were created for testing:**

$ kubectl delete deployment busybox