Set up is running with one master and one worker node

```
[root@master ~]# kubectl get nodes -o wide
NAME     STATUS   ROLES                  AGE   VERSION   INTERNAL-IP       EXTERNAL-IP   OS-IMAGE                KERNEL-VERSION                CONTAINER-RUNTIME
master   Ready    control-plane,master   78m   v1.20.1   192.168.148.107   <none>        CentOS Linux 7 (Core)   3.10.0-1160.11.1.el7.x86_64   docker://19.3.11
worker   Ready    worker                 49m   v1.20.1   192.168.148.110   <none>        CentOS Linux 7 (Core)   3.10.0-1160.11.1.el7.x86_64   docker://19.3.11
[root@master ~]# kubectl get pods -A
NAMESPACE     NAME                                READY   STATUS    RESTARTS   AGE
kube-system   coredns-74ff55c5b-bqrmd             1/1     Running   1          77m
kube-system   coredns-74ff55c5b-bwcqj             1/1     Running   1          77m
kube-system   etcd-master                         1/1     Running   1          77m
kube-system   kube-apiserver-master               1/1     Running   1          77m
kube-system   kube-controller-manager-master      1/1     Running   1          77m
kube-system   kube-flannel-ds-8v7tz               1/1     Running   1          49m
kube-system   kube-flannel-ds-lfmn6               1/1     Running   1          77m
kube-system   kube-proxy-hbxl8                    1/1     Running   1          77m
kube-system   kube-proxy-x96t7                    1/1     Running   1          49m
kube-system   kube-scheduler-master               1/1     Running   1          77m
[root@master ~]#
```

- Create the 2 namespaces for this lab we are taking dev and prod namespaces
  - Current list of namespaces
  - Create two names space yaml as shown
    [root@master ~]# cat << EOF > dev_ns.yml
    apiVersion: v1
    kind: Namespace
    metadata:
     name: dev
    EOF

    [root@master ~]# cat << EOF > prod_ns.yml
    apiVersion: v1
    kind: Namespace
    metadata:
     name: prod
    EOF

```
[root@master ~]# kubectl get ns
NAME              STATUS   AGE
default           Active   80m
kube-node-lease   Active   80m
kube-public       Active   80m
kube-system       Active   80m
[root@master ~]# cat dev_ns.yml
apiVersion: v1
kind: Namespace
metadata:
  name: dev
[root@master ~]# cat prod_ns.yml
apiVersion: v1
kind: Namespace
metadata:
  name: prod
[root@master ~]#
```

  - Create two names space using the yaml

```
[root@master ~]# kubectl create -f dev_ns.yml
namespace/dev created
k[root@master ~]# kubectl create -f prod_ns.yml
namespace/prod created
[root@master ~]# kubectl get ns
NAME              STATUS   AGE
default           Active   82m
dev               Active   11s
kube-node-lease   Active   82m
kube-public       Active   82m
kube-system       Active   82m
prod              Active   6s
[root@master ~]#
```

- Now create the deployment yaml each in both the namespaces created above
  [root@master ~]# cat << EOF > yaml_files/app1-dev.yml
  > apiVersion: apps/v1
  > kind: Deployment
  > metadata:
  > name: hello
  > namespace: dev
  > spec:
  > replicas: 1
  > selector:
  > matchLabels:
  > app: hello
  > template:
  > metadata:

```
>      labels:
>        app: hello
>    spec:
>      containers:
>      - name: hello-world
>        image: paulbouwer/hello-kubernetes:1.5
>        ports:
>        - containerPort: 8080
> EOF
[root@master ~]# cat << EOF > yaml_files/app2-prod.yml
> apiVersion: apps/v1
> kind: Deployment
> metadata:
>   name: hello
>   namespace: prod
> spec:
>   replicas: 1
>   selector:
>     matchLabels:
>       app: hello
>   template:
>     metadata:
>       labels:
>         app: hello
>     spec:
>       containers:
>       - name: hello-world
>         image: paulbouwer/hello-kubernetes:1.5
>         ports:
>         - containerPort: 8080
> EOF
```

- Once YAML files are ready apply the deployments

```
[root@master ~]# kubectl get pods -n dev
No resources found in dev namespace.
[root@master ~]# kubectl get pods -n prd
No resources found in prd namespace.
[root@master ~]# kubectl apply -f yaml_files/app1-dev.yml
deployment.apps/hello created
[root@master ~]# kubectl apply -f yaml_files/app2-prod.yml
deployment.apps/hello created
[root@master ~]# kubectl get pods -n dev
NAME                     READY   STATUS              RESTARTS   AGE
hello-565c6dd64b-mrmj4   0/1     ContainerCreating   0          9s
[root@master ~]# kubectl get pods -n prd
No resources found in prd namespace.
[root@master ~]# kubectl get pods -n prod
NAME                     READY   STATUS              RESTARTS   AGE
hello-565c6dd64b-vzsvw   0/1     ContainerCreating   0          13s
[root@master ~]#
```

- Wait till both the pods turns running

```
[root@master ~]# kubectl get pods -A |egrep 'dev|prod'
dev           hello-565c6dd64b-mrmj4           1/1     Running   0          62s
prod          hello-565c6dd64b-vzsvw           1/1     Running   0          58s
```

- Now define the service for both the dev and prod deployments
  - Define the service for dev deployment

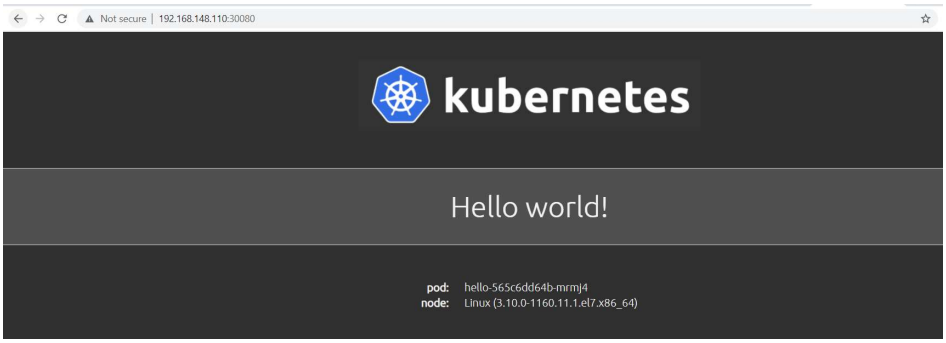
```
[root@master ~]# cat << EOF > yaml_files/app1-devsvc.yml
> apiVersion: v1
> kind: Service
> metadata:
>   name: hello
>   namespace: dev
> spec:
>   type: NodePort
>   ports:
>   - port: 80
>     targetPort: 8080
>     nodePort: 30080
>   selector:
>     app: hello
> EOF
```

  - Create the service

```
[root@master ~]# kubectl apply -f yaml_files/app1-devsvc.yml
service/hello created
[root@master ~]# kubectl get svc -n dev
NAME    TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
hello   NodePort   10.108.108.78   <none>        80:30080/TCP   7s
[root@master ~]#
```

  - Now verify the application availability

```
[root@master ~]# kubectl describe svc -n dev
Name:                     hello
Namespace:                dev
Labels:                   <none>
Annotations:              <none>
Selector:                 app=hello
Type:                     NodePort
IP Families:              <none>
IP:                       10.108.108.78
IPs:                      10.108.108.78
Port:                     <unset>  80/TCP
TargetPort:               8080/TCP
NodePort:                 <unset>  30080/TCP
Endpoints:                192.168.1.6:8080
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```




```
[root@worker ~]# netstat -ntpl |grep 30080
tcp        0      0 0.0.0.0:30080           0.0.0.0:*               LISTEN      1665/kube-proxy
[root@worker ~]# curl http://localhost:30080 |grep world
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   654  100   654    0     0  33285      0 --:--:-- --:--:-- --:--:-- 34421
  Hello world!
[root@worker ~]# curl http://192.168.148.110:30080 |grep world
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   654  100   654    0     0  24078      0 --:--:-- --:--:-- --:--:-- 25153
  Hello world!
[root@worker ~]#
```

- Now repeat the same process for Prod deployment and use different port (we will use 30081)

```
[root@master ~]# cat << EOF > yaml_files/app1-prodsvc.yml
> apiVersion: v1
> kind: Service
> metadata:
>   name: hello
>   namespace: prod
> spec:
>   type: NodePort
>   ports:
>   - port: 80
>     targetPort: 8080
>     nodePort: 30081
>   selector:
>     app: hello
> EOF
```

```
[root@master ~]# kubectl apply -f yaml_files/app1-prodsvc.yml
service/hello created
[root@master ~]# kubectl get svc -n prod
NAME    TYPE       CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
hello   NodePort   10.100.227.123   <none>        80:30081/TCP   15s
[root@master ~]# kubectl describe svc -n prod
Name:                     hello
Namespace:                prod
Labels:                   <none>
Annotations:              <none>
Selector:                 app=hello
Type:                     NodePort
IP Families:              <none>
IP:                       10.100.227.123
IPs:                      10.100.227.123
Port:                     <unset>  80/TCP
TargetPort:               8080/TCP
NodePort:                 <unset>  30081/TCP
Endpoints:                192.168.1.7:8080
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
[root@master ~]#
```

```
[root@worker ~]# netstat -ntpl |grep 30081
tcp        0      0 0.0.0.0:30081           0.0.0.0:*               LISTEN      1665/kube-proxy
[root@worker ~]# curl http://192.168.148.110:30081 |grep world
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   654  100   654    0     0   7202      0 --:--:-- --:--:-- --:--:--  7517
  Hello world!
[root@worker ~]#
```

- Now create utility pod using busybox image which will be used for test the functionality

```
[root@master ~]# cat << EOF > yaml_files/utilitypod.yml
> apiVersion: v1
> kind: Pod
> metadata:
>   name: utility
>   namespace: default
> spec:
>   containers:
>   - name: busybox
>     image: busybox:1.28
>     command:
>       - sleep
>       - "3600"
>     imagePullPolicy: IfNotPresent
>   restartPolicy: Always
> EOF
[root@master ~]#
```

- Apply the configuration and wait until pod gets in to the running state

```
[root@master ~]# kubectl apply -f yaml_files/utilitypod.yml
pod/utility created
[root@master ~]# kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
utility    1/1     Running   0          7s
[root@master ~]#
```

- Capture the Name and IP-Addr details

```
[root@master ~]# kubectl describe pods -n dev |egrep 'Name:|IP:'
Name:         hello-565c6dd64b-mrmj4
IP:           192.168.1.6
  IP:           192.168.1.6
    SecretName:  default-token-ws6nv
[root@master ~]# kubectl describe pods -n prod |egrep 'Name:|IP:'
Name:         hello-565c6dd64b-vzsvw
IP:           192.168.1.7
  IP:           192.168.1.7
    SecretName:  default-token-lqrr2
[root@master ~]# kubectl describe pods|egrep 'Name:|IP:'
Name:         utility
IP:           192.168.1.8
  IP:  192.168.1.8
    SecretName:  default-token-m47cj
[root@master ~]#
```

- Now verify the DNS Names

```
[root@master ~]# kubectl exec -it utility -- nslookup 192.168.1.6
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      192.168.1.6
Address 1: 192.168.1.6 192-168-1-6.hello.dev.svc.cluster.local
[root@master ~]# kubectl exec -it utility -- nslookup 192.168.1.7
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      192.168.1.7
Address 1: 192.168.1.7 192-168-1-7.hello.prod.svc.cluster.local
[root@master ~]# kubectl exec -it utility -- nslookup hello.dev.svc.cluster.local
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      hello.dev.svc.cluster.local
Address 1: 10.108.108.78 hello.dev.svc.cluster.local
[root@master ~]# kubectl exec -it utility -- nslookup hello.prod.svc.cluster.local
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      hello.prod.svc.cluster.local
Address 1: 10.100.227.123 hello.prod.svc.cluster.local
[root@master ~]#
```

- Now also try to ping the pod IP Address

```
[root@master ~]# kubectl exec -it utility -- ping 192.168.1.7 -c 1
PING 192.168.1.7 (192.168.1.7): 56 data bytes
64 bytes from 192.168.1.7: seq=0 ttl=64 time=0.295 ms

--- 192.168.1.7 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.295/0.295/0.295 ms
[root@master ~]# kubectl exec -it utility -- ping 192.168.1.6 -c 1
PING 192.168.1.6 (192.168.1.6): 56 data bytes
64 bytes from 192.168.1.6: seq=0 ttl=64 time=0.308 ms

--- 192.168.1.6 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.308/0.308/0.308 ms
[root@master ~]#
```