```
controlplane ~ ➜ kubectl get services
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.43.0.1    <none>        443/TCP   9m24s
```

That is a default service created by Kubernetes at launch.

What is the type of the default kubernetes service? Clusterip

What is the targetPort configured on the kubernetes service? 6443
Run the command: kubectl describe service and look at TargetPort.

```
controlplane ~ ➜ kubectl describe service
Name:                     kubernetes
Namespace:                default
Labels:                   component=apiserver
                          provider=kubernetes
Annotations:              <none>
Selector:                 <none>
Type:                     ClusterIP
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.43.0.1
IPs:                      10.43.0.1
Port:                     https  443/TCP
TargetPort:               6443/TCP
Endpoints:                192.4.244.9:6443
Session Affinity:         None
Events:                   <none>
```

How many labels are configured on the kubernetes service? 2

How many Endpoints are attached on the kubernetes service? 1

How many Deployments exist on the system now?

```
controlplane ~ ➜ kubectl get deployments
NAME                     READY   UP-TO-DATE   AVAILABLE   AGE
simple-webapp-deployment   4/4     4            4           17s
```

What is the image used to create the pods in the deployment?
Run the command: kubectl describe deployment and look under the containers section.

```
controlplane ~ ➜ kubectl describe deployments
Name:                   simple-webapp-deployment
Namespace:              default
CreationTimestamp:      Tue, 25 Apr 2023 05:15:24 +0000
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               name=simple-webapp
Replicas:               4 desired | 4 updated | 4 total | 4 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  name=simple-webapp
  Containers:
   simple-webapp:
    Image:        kodekloud/simple-webapp:red
    Port:         8080/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   simple-webapp-deployment-c7c68b6f4 (4/4 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  60s   deployment-controller  Scaled up replica set simple-webapp-deployment-c7c68b6f4 to 4
```

Are you able to accesss the Web App UI? No

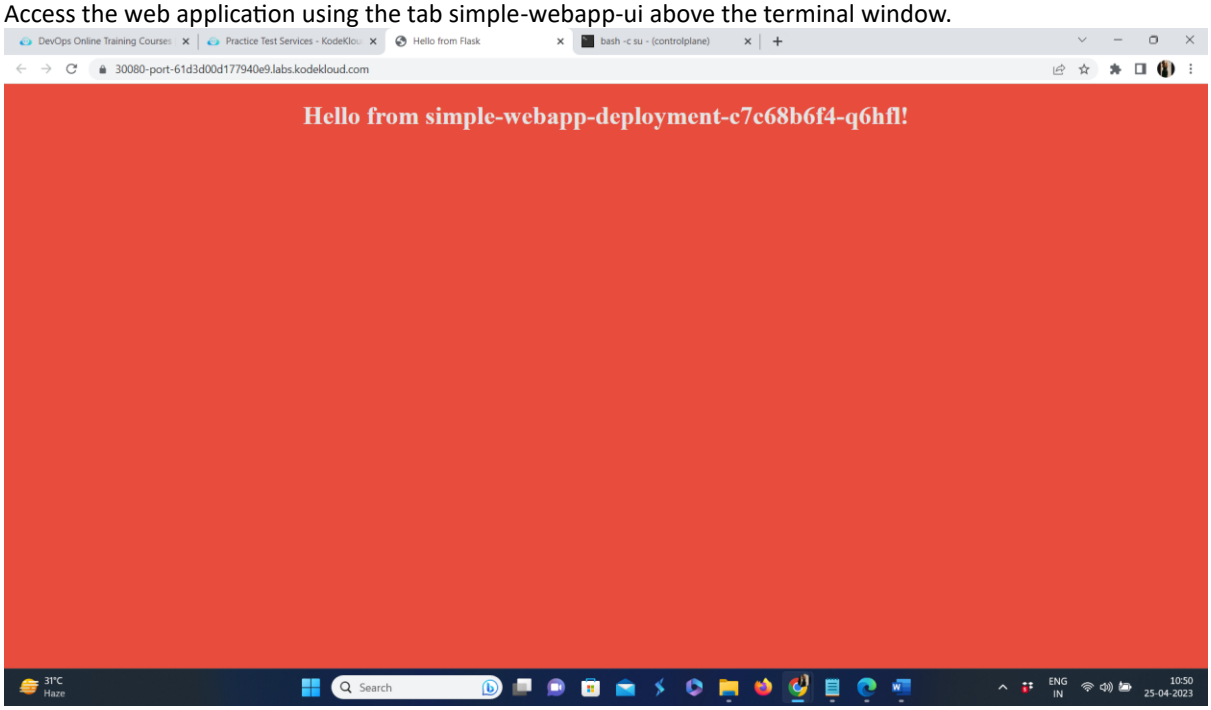


Create a new service to access the web application using the service-definition-1.yaml file.
Name: webapp-service
Type: NodePort
targetPort: 8080
port: 8080
nodePort: 30080
selector:
  name: simple-webapp

```
controlplane ~ ➜ cat service-definition-1.yaml
---
apiVersion: v1
kind: Service
metadata:
  name: webapp-service
  namespace: default
spec:
  ports:
  - nodePort: 30080
    port: 8080
    targetPort: 8080
  selector:
    name: simple-webapp
  type: NodePort

controlplane ~ ➜ kubectl apply -f service-definition-1.yaml
service/webapp-service created
```

Access the web application using the tab simple-webapp-ui above the terminal window.



Namespaces :
How many Namespaces exist on the system?

```
controlplane ~ ➜ kubectl get ns
NAME              STATUS   AGE
default           Active   9m57s
kube-system       Active   9m57s
kube-public       Active   9m57s
kube-node-lease   Active   9m57s
finance           Active   23s
marketing         Active   23s
dev               Active   23s
prod              Active   23s
manufacturing     Active   23s
research          Active   23s

controlplane ~ ➜ kubectl get ns | wc -l
11

controlplane ~ ➜ kubectl get ns --no-headers
default           Active   10m
kube-system       Active   10m
kube-public       Active   10m
kube-node-lease   Active   10m
finance           Active   55s
marketing         Active   55s
dev               Active   55s
prod              Active   55s
manufacturing     Active   55s
research          Active   55s

controlplane ~ ➜ kubectl get ns --no-headers | wc -l
10
```

How many pods exist in the research namespace?
Run the command to get exact the number of pods in the research namespace kubectl -n research get pods --no-headers | wc -l

```
controlplane ~ ✖ kubectl -n research get pods
NAME    READY   STATUS             RESTARTS      AGE
dna-1   0/1     CrashLoopBackOff   4 (72s ago)   2m49s
dna-2   0/1     CrashLoopBackOff   4 (64s ago)   2m48s

controlplane ~ ➜ kubectl -n research get pods --no-headers
dna-1   0/1   CrashLoopBackOff   4 (76s ago)   2m53s
dna-2   0/1   CrashLoopBackOff   4 (68s ago)   2m52s

controlplane ~ ➜ kubectl -n research get pods --no-headers | wc -l
2
```

Create a POD in the finance namespace.
Use the spec given below.
Run the command: kubectl run redis --image=redis -n finance

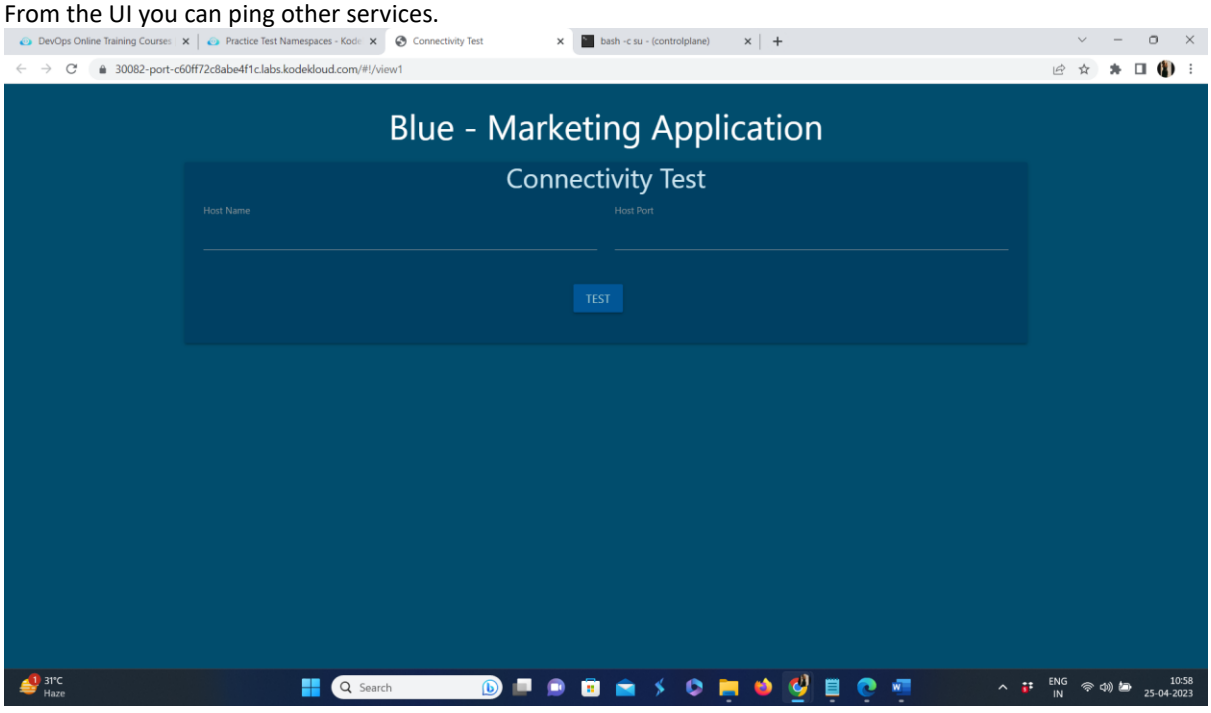
```
controlplane ~ ➜ kubectl run redis --image=redis -n finance
pod/redis created
```

Which namespace has the blue pod in it?
Run the command kubectl get pods --all-namespaces | grep blue

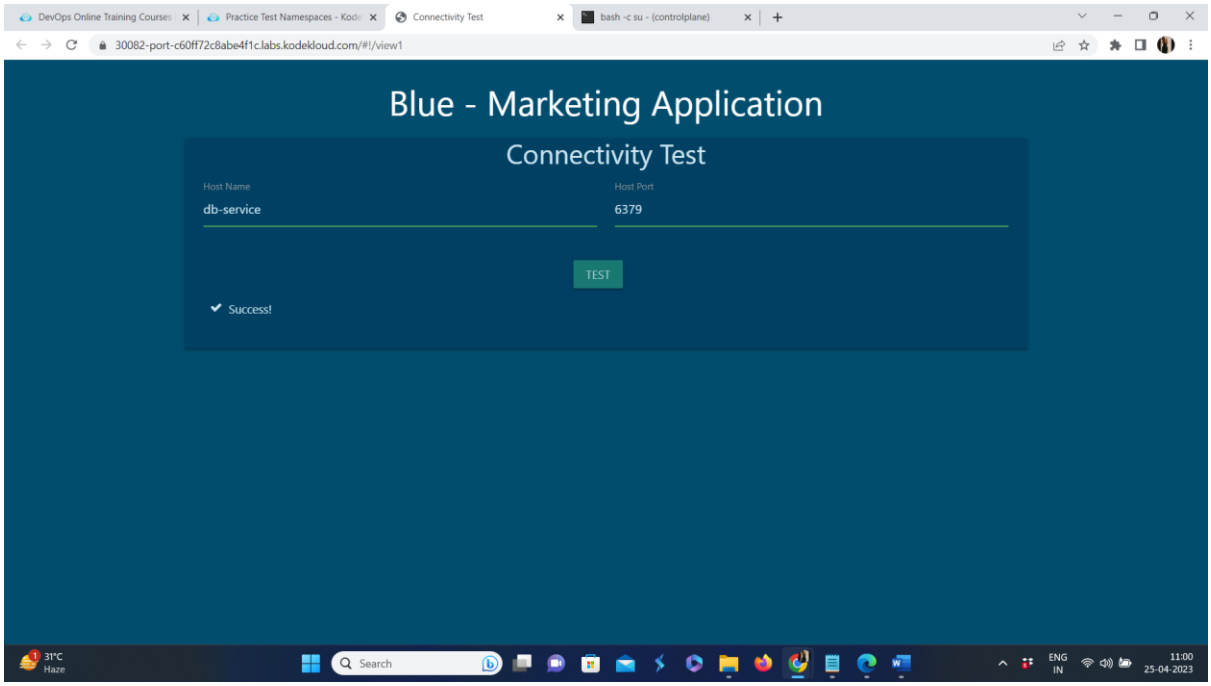
```
controlplane ~ ➜ kubectl get pods --all-namespaces
NAMESPACE       NAME                                      READY   STATUS             RESTARTS        AGE
kube-system     local-path-provisioner-5d56847996-sczbk   1/1     Running            0               14m
kube-system     coredns-5c6b6c5476-jlt8x                  1/1     Running            0               14m
kube-system     helm-install-traefik-crd-ztrrf            0/1     Completed          0               14m
kube-system     metrics-server-7b67f64457-m5f7z           1/1     Running            0               14m
kube-system     helm-install-traefik-q27n6                0/1     Completed          2               14m
kube-system     svclb-traefik-36294407-zqhxl              2/2     Running            0               13m
kube-system     traefik-56b8c5fb5c-klzt7                  1/1     Running            0               13m
marketing       redis-db                                  1/1     Running            0               4m46s
dev             redis-db                                  1/1     Running            0               4m46s
finance         payroll                                   1/1     Running            0               4m46s
marketing       blue                                      1/1     Running            0               4m46s
manufacturing   red-app                                   1/1     Running            0               4m46s
research        dna-1                                     0/1     CrashLoopBackOff   5 (106s ago)    4m47s
research        dna-2                                     0/1     CrashLoopBackOff   5 (91s ago)     4m46s
finance         redis                                     1/1     Running            0               42s

controlplane ~ ➜ kubectl get pods --all-namespaces | grep blue
marketing       blue                                      1/1     Running            0               4m54s
```

Access the Blue web application using the link above your terminal!!
From the UI you can ping other services.



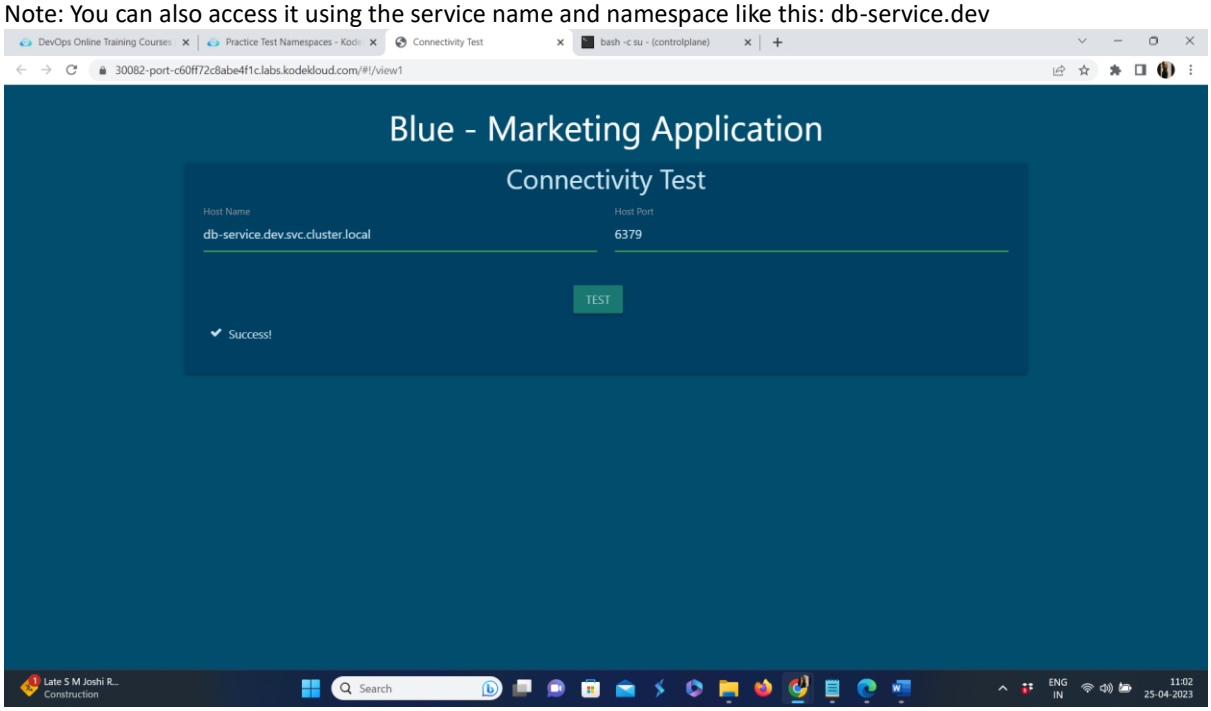

What DNS name should the Blue application use to access the database db-service in its own namespace - marketing?
You can try it in the web application UI. Use port 6379.
Since the blue application and the db-service are in the same namespace, we can simply use the service name to access the database.

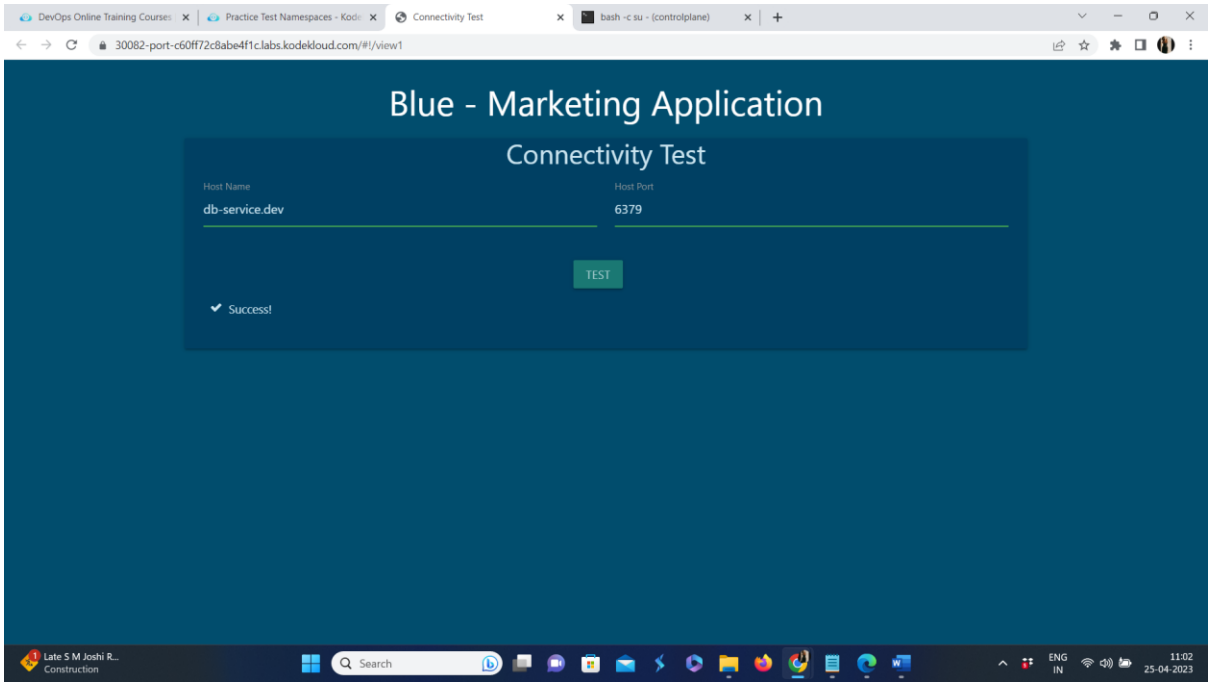



What DNS name should the Blue application use to access the database db-service in the dev namespace? You can try it in the web application UI. Use port 6379.

Since the blue application and the db-service are in different namespaces. In this case, we need to use the service name along with the namespace to access the database. The FQDN (fully Qualified Domain Name) for the db-service in this example would be db-service.dev.svc.cluster.local.

Note: You can also access it using the service name and namespace like this: db-service.dev






---

Imperative commands

In this lab, you will get hands-on practice with creating Kubernetes objects imperatively.
All the questions in this lab can be done imperatively. However, for some questions, you may need to first create the YAML file using imperative methods. You can then modify the YAML according to the need and create the object using kubectl apply -f command.

Deploy a pod named nginx-pod using the nginx:alpine image.
Use imperative commands only.

Run the command: kubectl run nginx-pod --image=nginx:alpine

```
controlplane ~ ➜ kubectl run nginx-pod --image=nginx:alpine
pod/nginx-pod created
```

Deploy a redis pod using the redis:alpine image with the labels set to tier=db.
Either use imperative commands to create the pod with the labels. Or else use imperative commands to generate the pod definition file, then add the labels before creating the pod using the file.

```
controlplane ~ ➜ kubectl run redis --image=redis:alpine --dry-run=client -o yaml > redis-pod.yaml

controlplane ~ ➜ vi redis-pod.yaml

controlplane ~ ➜ vi redis-pod.yaml

controlplane ~ ➜ cat redis-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: redis
    tier: db
  name: redis
spec:
  containers:
  - image: redis:alpine
    name: redis
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}

controlplane ~ ➜ kubectl create -f redis-pod.yaml
pod/redis created

controlplane ~ ➜ kubectl run redisnew -l tier=db --image=redis:alpine
pod/redisnew created
```

Create a service redis-service to expose the redis application within the cluster on port 6379.
Use imperative commands.
Run the command: kubectl expose pod redis --port=6379 --name redis-service

```
controlplane ~ ➜ kubectl expose pod redis --port=6379 --name redis-service
service/redis-service exposed
```

Create a deployment named webapp using the image kodekloud/webapp-color with 3 replicas.
Try to use imperative commands only. Do not create definition files.
Run the command: kubectl create deployment webapp --image=kodekloud/webapp-color --replicas=3

```
controlplane ~ ➜ kubectl create deployment webapp --image=kodekloud/webapp-color --replicas=3
deployment.apps/webapp created
```

Create a new pod called custom-nginx using the nginx image and expose it on container port 8080.
Run the command: kubectl run custom-nginx --image=nginx --port=8080

```
controlplane ~ ➜ kubectl run custom-nginx --image=nginx --port=8080
pod/custom-nginx created
```

Create a new namespace called dev-ns.
Use imperative commands.
Run the command: kubectl create namespace dev-ns or kubectl create ns dev-ns

```
controlplane ~ ➜ kubectl create ns dev-ns
namespace/dev-ns created

controlplane ~ ➜
```

Create a new deployment called redis-deploy in the dev-ns namespace with the redis image. It should have 2 replicas.
Use imperative commands.
Run the command: kubectl create deployment redis-deploy --image=redis --replicas=2 -n dev-ns

```
controlplane ~ ➜ kubectl create deployment redis-deploy --image=redis --replicas=2 -n dev-ns
deployment.apps/redis-deploy created
```

Create a pod called httpd using the image httpd:alpine in the default namespace. Next, create a service of type ClusterIP by the same name (httpd). The target port for the service should be 80.
Try to do this with as few steps as possible.
Run the command: kubectl run httpd --image=httpd:alpine --port=80 –expose

```
controlplane ~ ➜ kubectl run httpd --image=httpd:alpine --port=80 --expose
service/httpd created
pod/httpd created
```