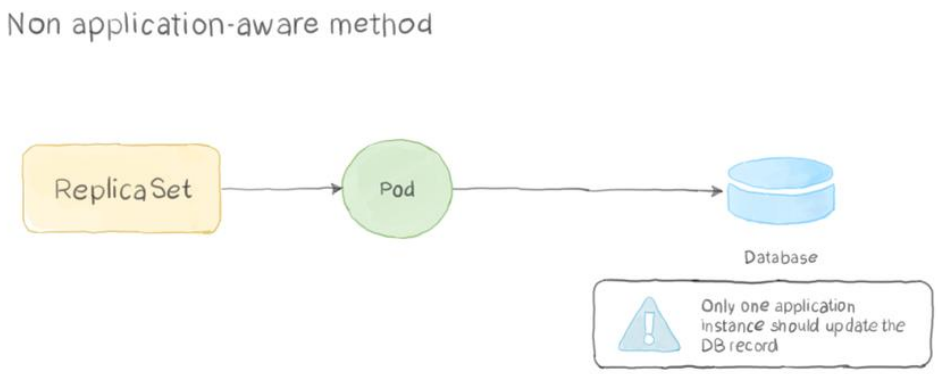
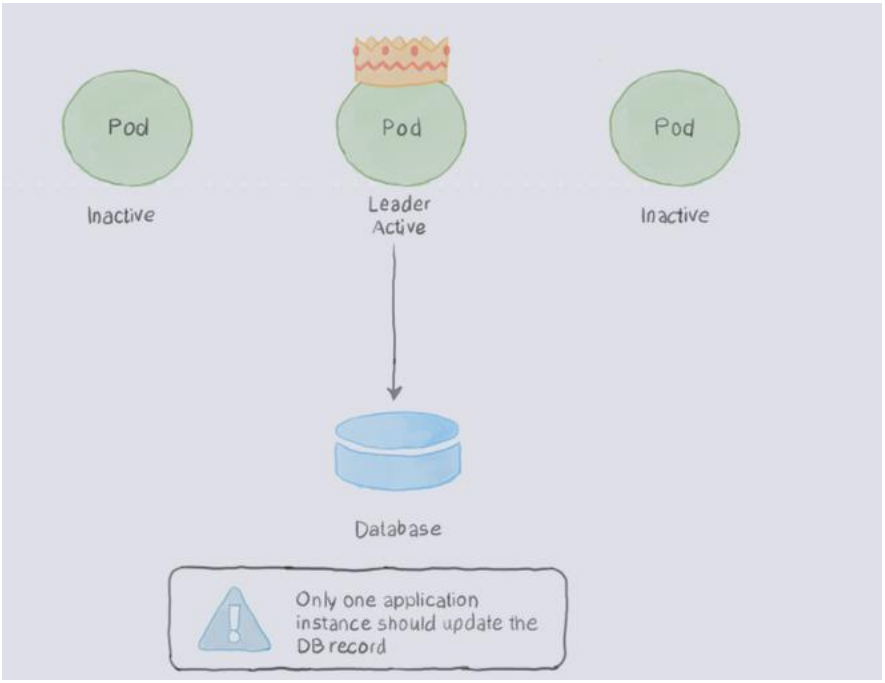
Implementing the Singleton Pattern in Kubernetes

This is primarily for Developer where they use ETCD API.

In a Kubernetes cluster, the default behaviour is to run and maintain several versions (replicas) of the application for high availability. A web application that runs on only one Nginx instance is vulnerable to downtime if this Nginx went down or got restarted. However, sometimes this may not best-serve your environment needs. In the microservices architecture, an application may be running on more than one component. If the application is hosted on Kubernetes, some of those components may need to follow the Singleton pattern when they run.

For example, a web application that needs to consume a message from a message queue in a sequential manner should not have more than one instance running at a time. Let's see how we can implement the Singleton pattern in Kubernetes using the two methods that we described earlier: from within the application and from outside the application.





Singleton Service

- Overview: Singleton Pattern ensures only one instance of an application is active at a time and is highly available
- Problem:
  - In some cases, only one instance of the service is allowed to run
- Solution:
  - Out of Application Locking:
    - Stateful set or replica set with replicas = 1
  - In-Application Locking
    - In a distribute environment one way to control the service instance count is through a distributed lock
    - We can implement leader election using kubernetes api's. For example apache camel has a K8s connector that also provides leader election and Singleton capabilities
    - This connector access the etcd api directly and k8s api to use Config Maps to acquire a distributed lock Refer Here
    - Refer Here for sample leader election in k8s.

DNS For Services and Pods

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deployment-example
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14
        ports:
          - containerPort: 80
            name: http
```

```yaml
kind: Service
apiVersion: v1
metadata:
  name: service-example
spec:
  ports:
    # Accept traffic sent to port 80
    - name: http
      port: 80
      targetPort: 80
  selector:
    app: nginx
  type: ClusterIP
```

```yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: test
spec:
  containers:
    - image: alpine
      name: alpine
      args: ["sleep", "1d"]
```

```
PS D:\khajaclassroom\ExpertK8s\DNS> kubectl apply -f .\nginx-deploy.yaml
deployment.apps/deployment-example created
PS D:\khajaclassroom\ExpertK8s\DNS> kubectl get deploy
NAME                 READY   UP-TO-DATE   AVAILABLE   AGE
PS D:\khajaclassroom\ExpertK8s\DNS> kubectl get rs
NAME                            DESIRED   CURRENT   READY   AGE
deployment-example-94f57779c    2         2         2       14s
PS D:\khajaclassroom\ExpertK8s\DNS> kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
deployment-example-94f57779c-bc8wt    1/1     Running   0          22s
deployment-example-94f57779c-mwvjx    1/1     Running   0          22s
PS D:\khajaclassroom\ExpertK8s\DNS> kubectl apply -f .\nginx-svc.yaml
service/service-example created
PS D:\khajaclassroom\ExpertK8s\DNS> kubectl get svc
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
kubernetes        ClusterIP   10.79.240.1     <none>        443/TCP   6m13s
service-example   ClusterIP   10.79.246.136   <none>        80/TCP    6s
```

Lets create the test pod

```
PS D:\khajaclassroom\ExpertK8s\DNS> kubectl apply -f .\testpod.yaml
pod/test created
PS D:\khajaclassroom\ExpertK8s\DNS> kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
deployment-example-94f57779c-bc8wt    1/1     Running   0          2m47s
deployment-example-94f57779c-mwvjx    1/1     Running   0          2m47s
test                                  1/1     Running   0          7s
PS D:\khajaclassroom\ExpertK8s\DNS>
```

Make a note of service names and ip addresses of pods

```
PS D:\khajaclassroom\ExpertK8s\DNS> kubectl get svc
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
kubernetes        ClusterIP   10.79.240.1     <none>        443/TCP   8m57s
service-example   ClusterIP   10.79.246.136   <none>        80/TCP    2m50s
PS D:\khajaclassroom\ExpertK8s\DNS> kubectl get pods -o wide
NAME                                  READY   STATUS    RESTARTS   AGE     IP           NODE
                NOMINATED NODE   READINESS GATES
deployment-example-94f57779c-bc8wt    1/1     Running   0          3m45s   10.76.0.11   gke-hello-cluster-default-pool
-c0ab9be8-4dl9   <none>           <none>
deployment-example-94f57779c-mwvjx    1/1     Running   0          3m45s   10.76.0.10   gke-hello-cluster-default-pool
-c0ab9be8-4dl9   <none>           <none>
test                                  1/1     Running   0          65s     10.76.0.12   gke-hello-cluster-default-pool
-c0ab9be8-4dl9   <none>           <none>
PS D:\khajaclassroom\ExpertK8s\DNS>
```

Now login into test pod

kubectl exec -it test -- /bin/sh

Look into resolve.conf

```
PS D:\khajaclassroom\ExpertK8s\DNS> kubectl exec -it test -- /bin/sh
/ # cat /etc/resolv.conf
search default.svc.cluster.local svc.cluster.local cluster.local us-central1-a.c.expertkubernetes.internal c.expertku
bernetes.internal google.internal
nameserver 10.79.240.10
options ndots:5
/ #
```

What objects in k8s gets DNS records
Services and Pods

DNS A record Sample: Google.com = 8.8.8.8

The 'A' stands for 'address' and this is the most fundamental type of DNS record: it indicates the IP address of a given domain. For example, if you pull the DNS records of cloudflare.com, the A record currently returns an IP address of: 104.17.210.9.

A record only hold IPv4 addresses. If a website has an IPv6 address, it will instead use an 'AAAA' record.

Here is an example of an A record:

example.com	record type:	value:	TTL
@	A	192.0.2.1	14400

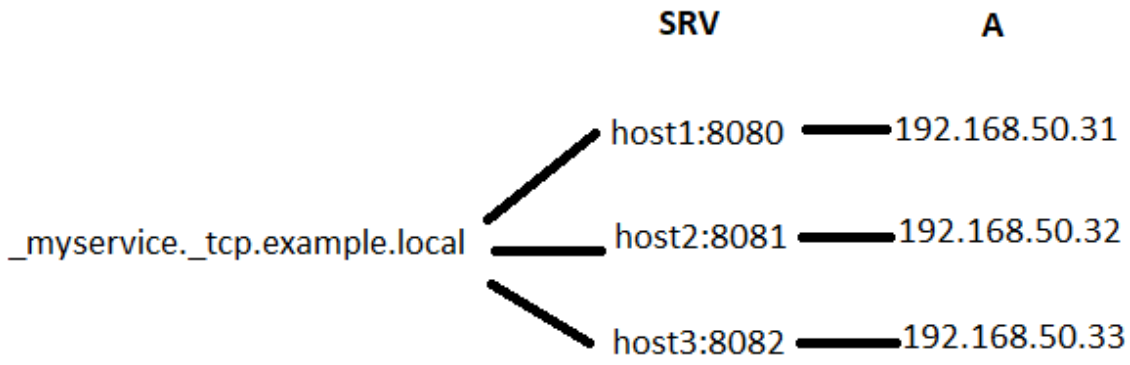
DNS CNAME (Here alias name is also specified like- fb.com, facebook.com)

Example of a CNAME record:

blog.example.com	record type:	value:	TTL
@	CNAME	is an alias of example.com	32600

What is a DNS SRV record?

The DNS 'service' (SRV) record specifies a host and port for specific services such as voice over IP (VoIP), instant messaging, and so on. Most other DNS records only specify a server or an IP address, but SRV records include a port at that IP address as well. Some Internet protocols require the use of SRV records in order to function.



- K8s Services:
  - A/AAAA records
    - Normal Services are assigned a DNS A record depending on the IP family of the service for the name in the form of my-svc.my-namespace.svc.cluster-domain.example
  - SRV records:
    - SRV records are created for each named ports that are part of normal or Headless Services. For each name port the SRV record would have the form _my-port-name._my-port-protocol.my-svc.my-namespace.svc.cluster.local
- K8S Pods
  - A/AAAA records:
    - Pod has the following dns resolution
  - pod-ip-address.my-namespace.pod.cluster.local

- In kubernetes in addition to DNS records k8s create environmental variables with service information

```
KUBERNETES_PORT='tcp://10.79.240.1:443'
KUBERNETES_PORT_443_TCP='tcp://10.79.240.1:443'
KUBERNETES_PORT_443_TCP_ADDR='10.79.240.1'
KUBERNETES_PORT_443_TCP_PORT='443'
KUBERNETES_PORT_443_TCP_PROTO='tcp'
KUBERNETES_SERVICE_HOST='10.79.240.1'
KUBERNETES_SERVICE_PORT='443'
KUBERNETES_SERVICE_PORT_HTTPS='443'
LINENO=''
OPTIND='1'
PATH='/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin'
PPID='0'
PS1='\w \$ '
PS2='> '
PS4='+ '
PWD='/'
SERVICE_EXAMPLE_PORT='tcp://10.79.246.136:80'
SERVICE_EXAMPLE_PORT_80_TCP='tcp://10.79.246.136:80'
SERVICE_EXAMPLE_PORT_80_TCP_ADDR='10.79.246.136'
SERVICE_EXAMPLE_PORT_80_TCP_PORT='80'
SERVICE_EXAMPLE_PORT_80_TCP_PROTO='tcp'
SERVICE_EXAMPLE_SERVICE_HOST='10.79.246.136'
SERVICE_EXAMPLE_SERVICE_PORT='80'
SERVICE_EXAMPLE_SERVICE_PORT_HTTP='80'
SHLVL='1'
TERM='xterm'
_='10-76-0-11.default.pod.cluster.local'
```

```
/ # ping service-example.default.svc.cluster.local
PING service-example.default.svc.cluster.local (10.79.246.136): 56 data bytes
```

```
PS D:\khajaclassroom\ExpertK8s\DNS> kubectl get svc -w
NAME              TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes        ClusterIP      10.79.240.1     <none>        443/TCP        35m
service-example   LoadBalancer   10.79.246.136   <pending>     80:32672/TCP   29m
service-example   LoadBalancer   10.79.246.136   34.132.135.129   80:32672/TCP   30m
```