Readiness Health and Liveness Health

[Edition 1]

[Last Update 200831]

For any issues/help contact : ***support@k21academy.com***

Contents

1 INTRODUCTION

- Kubernetes makes sure the readiness probe passes before allowing a service to send traffic to the pod. If a readiness probe starts to fail, Kubernetes stops sending traffic to the pod until it passes.
- Liveness probes let Kubernetes know if your app is alive or dead. If you app is alive, then Kubernetes leaves it alone. If your app is dead, Kubernetes removes the Pod and starts a new one to replace it.

This guide Covers:

1. Configuring Readiness Probes for Container Health Check

    - Create a Pod with readiness probe health and expose it with a service.
    - Simulating Readiness Probe failure.
    - Delete all the resources created in this lab exercise.

2. Configuring Liveness Probes for Container Health Check

    - Create a Pod yaml with liveness probe.
    - Simulating Liveness Probe failure.
    - Delete all the resources created in this lab exercise.

2 DOCUMENTATION

2.1 Kubernetes Documentation

1. Readiness Health and Liveness Health
   https://www.magalix.com/blog/kubernetes-and-containers-best-practices-health-probes
2. Pod Readiness Probe
   https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/
3. Simulating Readiness Probe
   https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/troubleshooting-kubeadm/
4. Pod yaml with liveness probe
   https://blog.colinbreck.com/kubernetes-liveness-and-readiness-probes-how-to-avoid-shooting-yourself-in-the-foot/

2.2 Linux Commands and VIM Commands

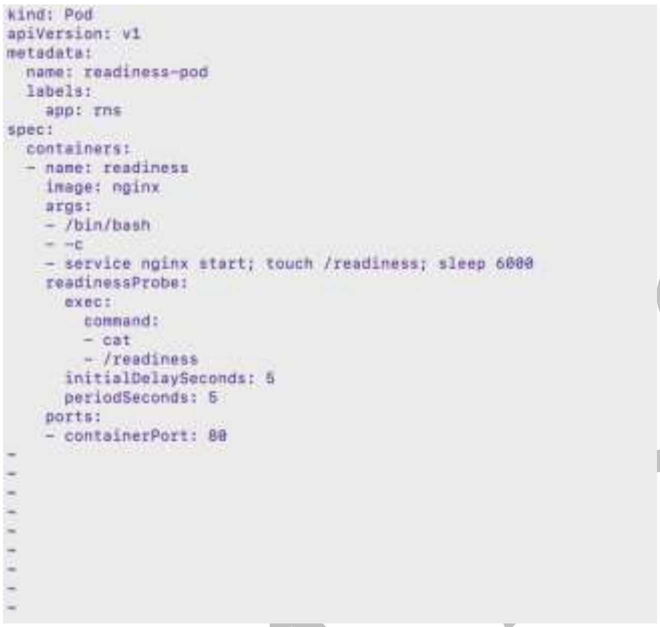
1. Basic Linux Commands
   https://maker.pro/linux/tutorial/basic-linux-commands-for-beginners
   https://www.hostinger.in/tutorials/linux-commands
2. Basic VIM Commands
   https://coderwall.com/p/adv71w/basic-vim-commands-for-getting-started
3. Popular VIM Commands
   https://www.keycdn.com/blog/vim-commands

3 CONFIGURING READINESS PROBES FOR CONTAINER HEALTH CHECK

3.1 Create a Pod with readiness probe and expose it with a service

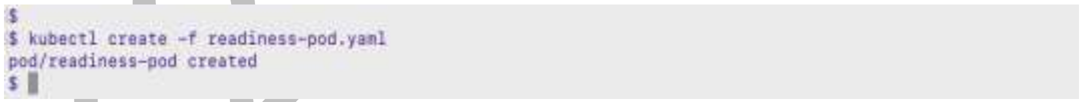
1. Create a Pod yaml and enter the content given below

```
$ vi readiness-pod.yaml
```



```
kind: Pod
apiVersion: v1
metadata:
  name: readiness-pod
  labels:
    app: rns
spec:
  containers:
  - name: readiness
    image: nginx
    args:
    - /bin/bash
    - -c
    - service nginx start; touch /readiness; sleep 6000
    readinessProbe:
      exec:
        command:
        - cat
        - /readiness
      initialDelaySeconds: 5
      periodSeconds: 5
    ports:
    - containerPort: 80
~
~
~
~
~
~
~
~
~
```

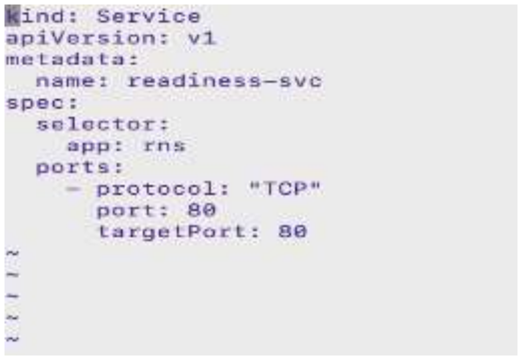
```
$ kubectl create -f readiness-pod.yaml
```



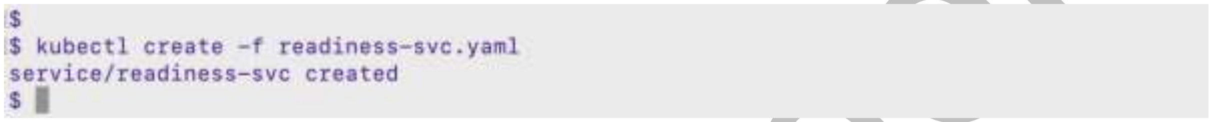
```
$
$ kubectl create -f readiness-pod.yaml
pod/readiness-pod created
$ 
```

2. Create a service yaml and enter the contents given below

```
$ vi readiness-svc.yaml
```

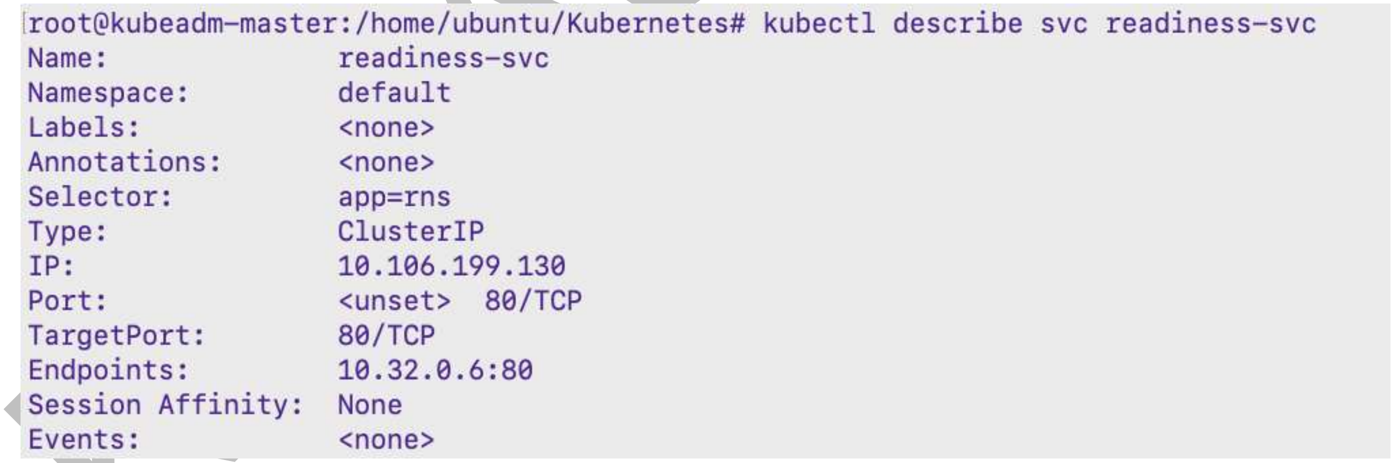

```
kind: Service
apiVersion: v1
metadata:
  name: readiness-svc
spec:
  selector:
    app: rns
  ports:
    - protocol: "TCP"
      port: 80
      targetPort: 80
~
~
~
~
~
```

```
$ kubectl create -f readiness-svc.yaml
```


```
$
$ kubectl create -f readiness-svc.yaml
service/readiness-svc created
$
```

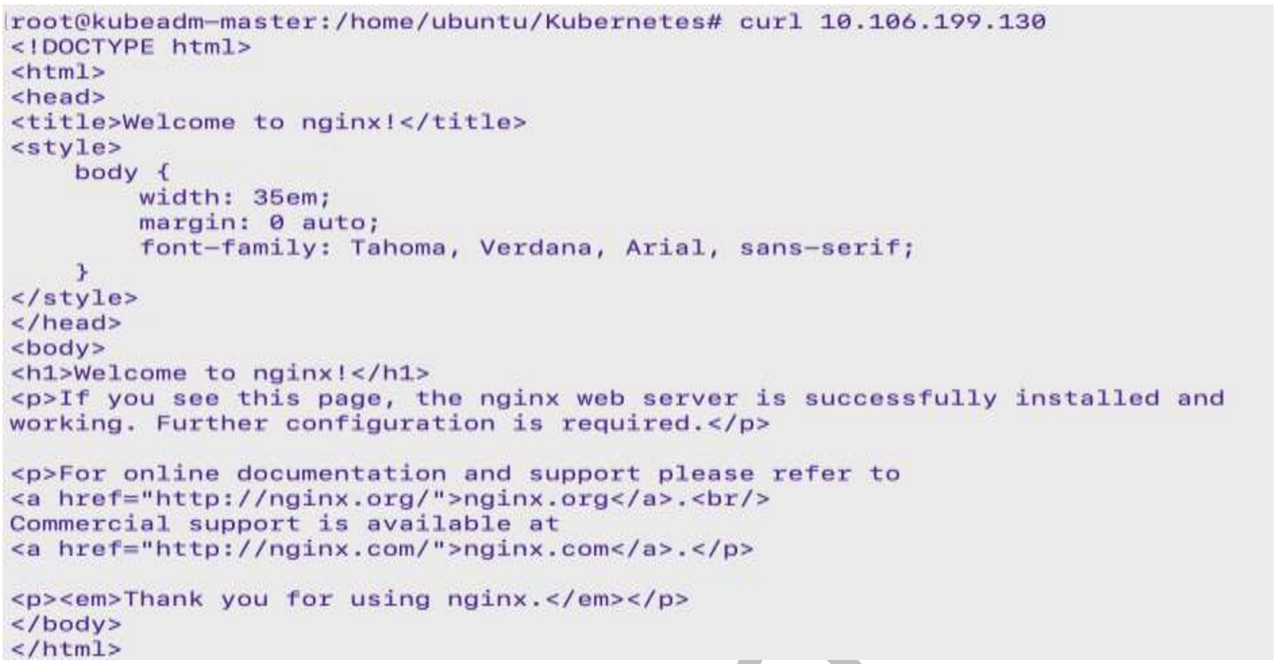
3. Describe the service and verify the pod endpoint is populated

```
$ kubectl describe svc readiness-svc
```


```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl describe svc readiness-svc
Name:              readiness-svc
Namespace:         default
Labels:            <none>
Annotations:       <none>
Selector:          app=rns
Type:              ClusterIP
IP:                10.106.199.130
Port:              <unset>  80/TCP
TargetPort:        80/TCP
Endpoints:         10.32.0.6:80
Session Affinity:  None
Events:            <none>
```

4. Curl on the service IP Address and see if you are able to reach the nginx web server in the conatiner

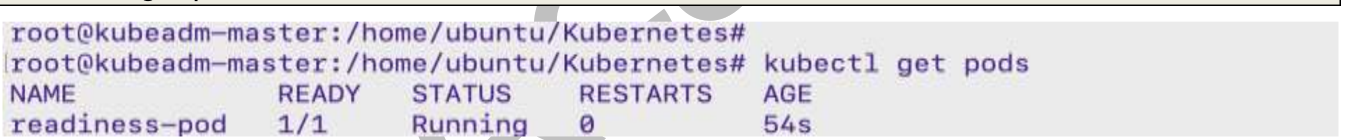
```
$ curl  10.106.199.130
```



```
root@kubeadm-master:/home/ubuntu/Kubernetes# curl 10.106.199.130
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    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>
```

5. Verify the status of the pod it should be in ready and running state

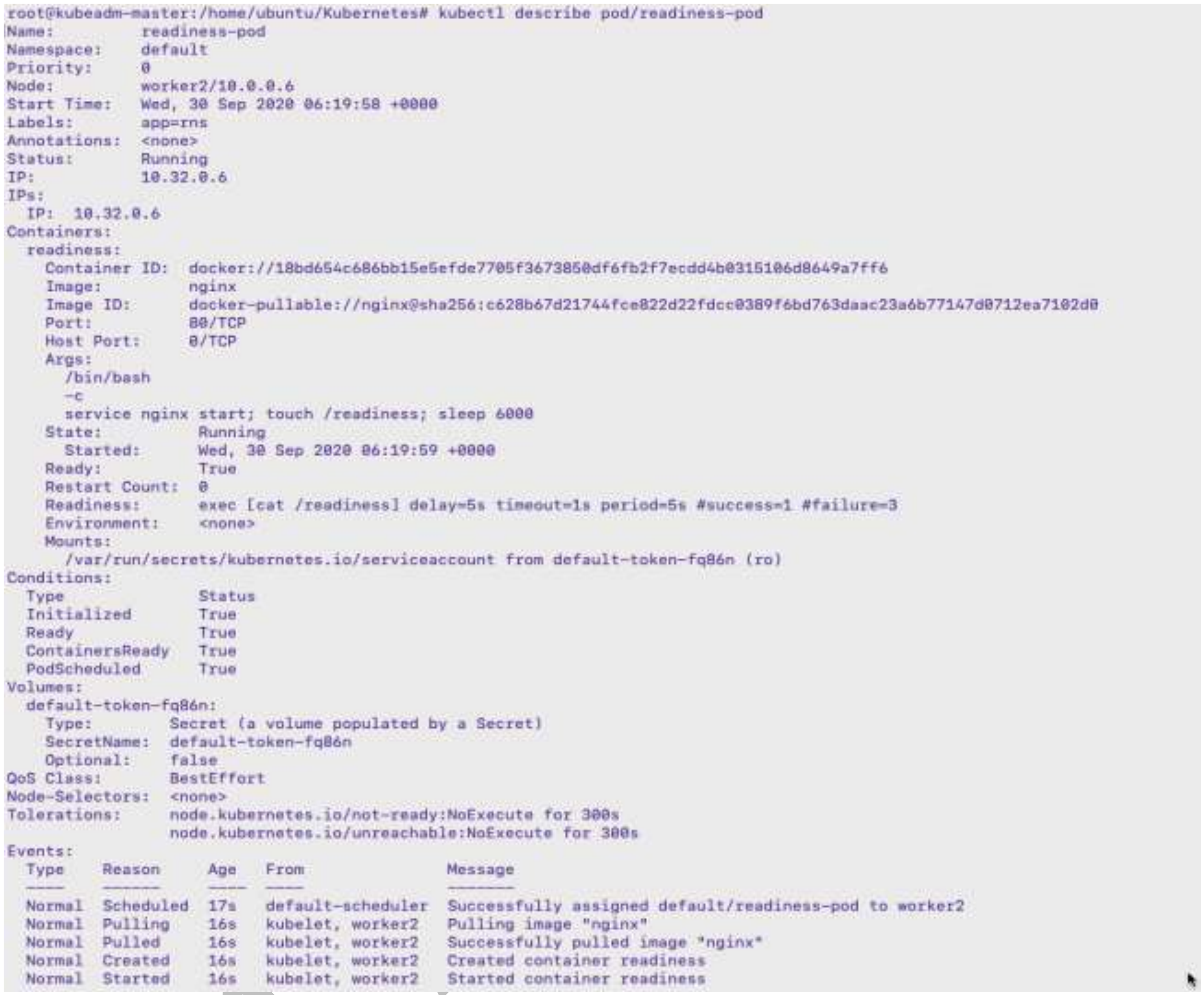
```
$ kubectl get pods
```



```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get pods
NAME            READY   STATUS    RESTARTS   AGE
readiness-pod   1/1     Running   0          54s
```

6. Lets describe the pod and look for the readiness probe parameters which got configured for the conatiner

```
$ kubectl describe pod readiness-pod
```

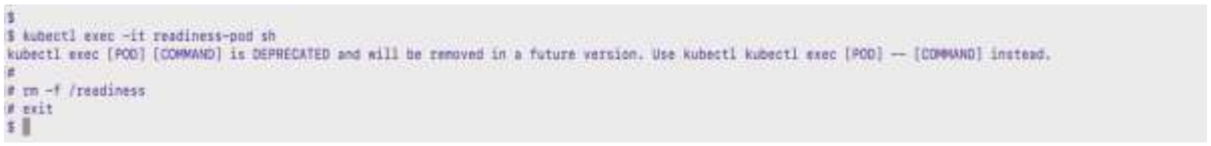


```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl describe pod/readiness-pod
Name:         readiness-pod
Namespace:    default
Priority:     0
Node:         worker2/10.0.0.6
Start Time:   Wed, 30 Sep 2020 06:19:58 +0000
Labels:       app=rns
Annotations:  <none>
Status:       Running
IP:           10.32.0.6
IPs:
  IP:  10.32.0.6
Containers:
  readiness:
    Container ID:  docker://18bd654c686bb15e5efde7705f3673850df6fb2f7ecdd4b0315106d8649a7ff6
    Image:         nginx
    Image ID:      docker-pullable://nginx@sha256:c628b67d21744fce822d22fdcc0389f6bd763daac23a6b77147d0712ea7102d0
    Port:          80/TCP
    Host Port:     0/TCP
    Args:
      /bin/bash
      -c
      service nginx start; touch /readiness; sleep 6000
    State:          Running
      Started:      Wed, 30 Sep 2020 06:19:59 +0000
    Ready:          True
    Restart Count:  0
    Readiness:      exec [cat /readiness] delay=5s timeout=1s period=5s #success=1 #failure=3
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-fq86n (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-fq86n:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-fq86n
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  17s   default-scheduler  Successfully assigned default/readiness-pod to worker2
  Normal  Pulling    16s   kubelet, worker2   Pulling image "nginx"
  Normal  Pulled     16s   kubelet, worker2   Successfully pulled image "nginx"
  Normal  Created    16s   kubelet, worker2   Created container readiness
  Normal  Started    16s   kubelet, worker2   Started container readiness
```

3.2 Simulating Readiness Probe failure

1. Get shell access to the container and delete the /readiness file

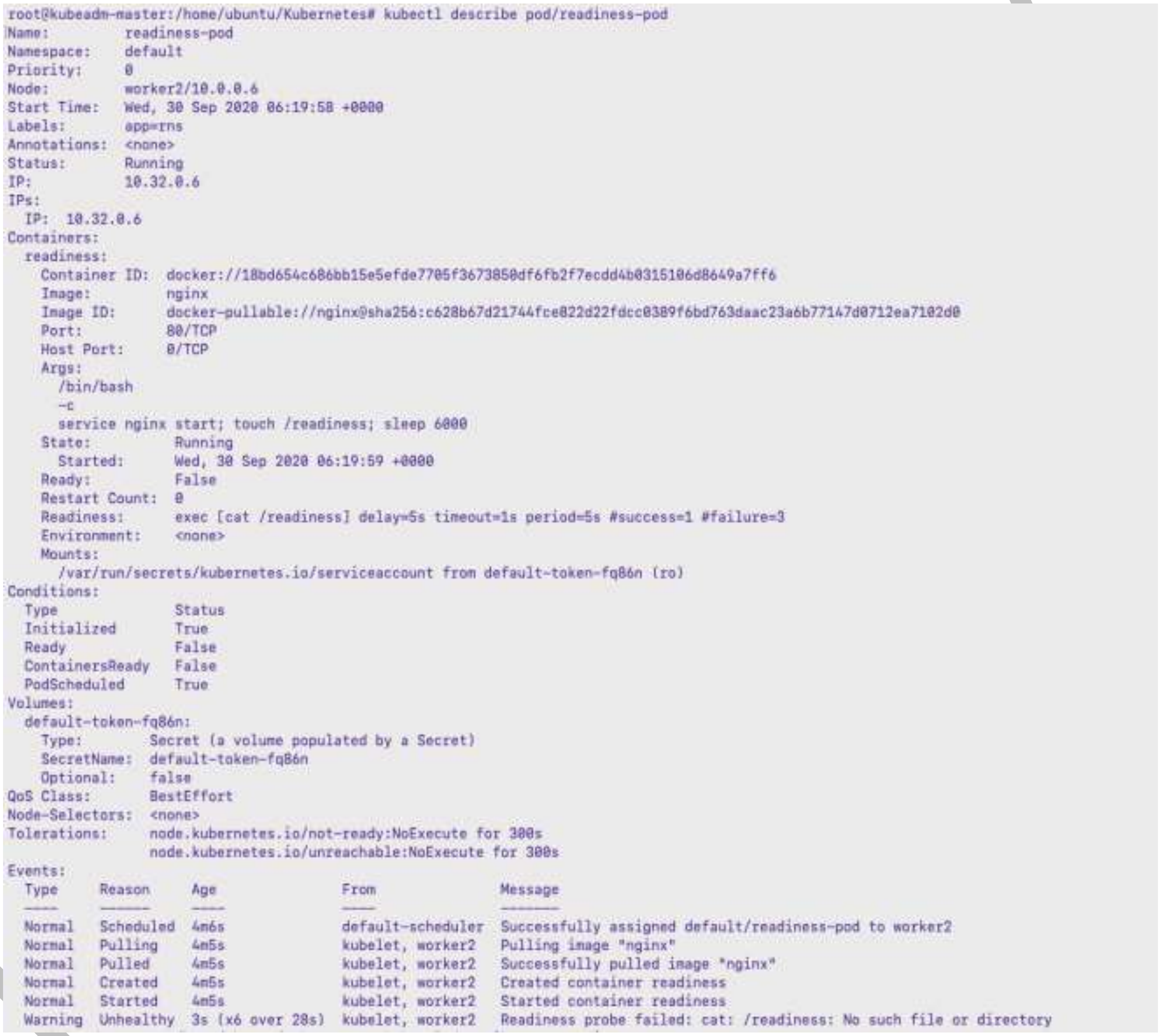
```
$ kubectl exec -it readiness-pod sh
# rm -f /readiness
  # exit
```



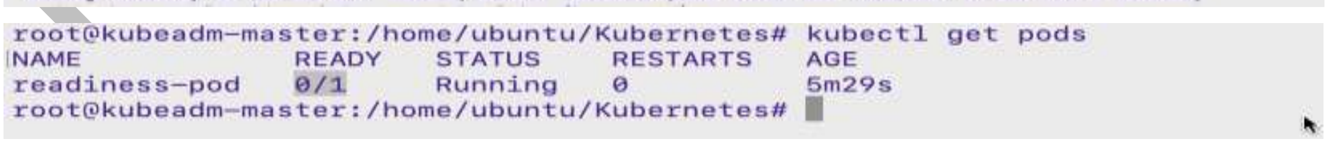
```
$
$ kubectl exec -it readiness-pod sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl kubectl exec [POD] -- [COMMAND] instead.
#
# rm -f /readiness
# exit
$
```

2. View pod events and see that the readiness probe has failed and also check pod health status

```
$ kubectl describe pod readiness-pod

$ kubectl get pods
```



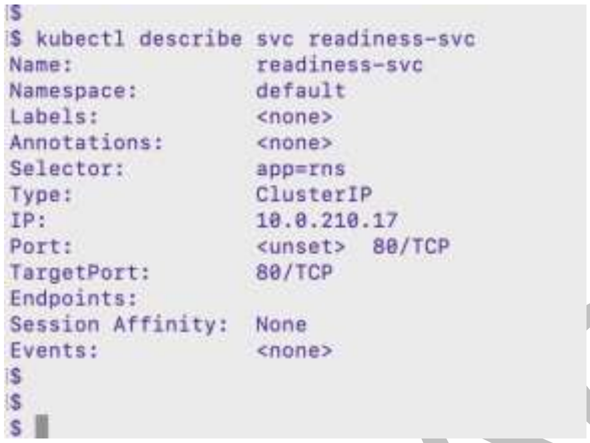
```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl describe pod/readiness-pod
Name:         readiness-pod
Namespace:    default
Priority:     0
Node:         worker2/10.0.0.6
Start Time:   Wed, 30 Sep 2020 06:19:58 +0000
Labels:       app=rns
Annotations:  <none>
Status:       Running
IP:           10.32.0.6
IPs:
  IP:  10.32.0.6
Containers:
  readiness:
    Container ID:  docker://18bd654c686bb15e5efde7705f3673850df6fb2f7ecdd4b0315106d8649a7ff6
    Image:         nginx
    Image ID:      docker-pullable://nginx@sha256:c628b67d21744fce822d22fdcc0389f6bd763daac23a6b77147d0712ea7102d0
    Port:          80/TCP
    Host Port:     0/TCP
    Args:
      /bin/bash
      -c
      service nginx start; touch /readiness; sleep 6000
    State:          Running
      Started:      Wed, 30 Sep 2020 06:19:59 +0000
    Ready:          False
    Restart Count:  0
    Readiness:      exec [cat /readiness] delay=5s timeout=1s period=5s #success=1 #failure=3
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-fq86n (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             False
  ContainersReady   False
  PodScheduled      True
Volumes:
  default-token-fq86n:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-fq86n
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s
Events:
  Type     Reason     Age                From               Message
  ----     ------     ----               ----               -------
  Normal   Scheduled  4m6s               default-scheduler  Successfully assigned default/readiness-pod to worker2
  Normal   Pulling    4m5s               kubelet, worker2   Pulling image "nginx"
  Normal   Pulled     4m5s               kubelet, worker2   Successfully pulled image "nginx"
  Normal   Created    4m5s               kubelet, worker2   Created container readiness
  Normal   Started    4m5s               kubelet, worker2   Started container readiness
  Warning  Unhealthy  3s (x6 over 28s)   kubelet, worker2   Readiness probe failed: cat: /readiness: No such file or directory
```



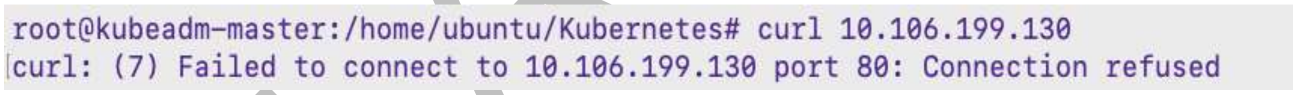
```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get pods
NAME            READY   STATUS    RESTARTS   AGE
readiness-pod   0/1     Running   0          5m29s
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

3. Describe service and see that the endpoints field it not populated meaning the traffic is not being routed to the Pod. Verify that by trying to reach the web server using service IP address

```
$ kubectl describe svc readiness-svc
$ curl <service Ip Address>
```



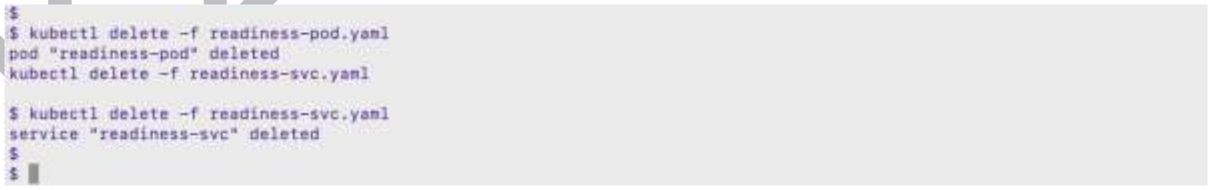
```
$
$ kubectl describe svc readiness-svc
Name:              readiness-svc
Namespace:         default
Labels:            <none>
Annotations:       <none>
Selector:          app=rns
Type:              ClusterIP
IP:                10.0.210.17
Port:              <unset>  80/TCP
TargetPort:        80/TCP
Endpoints:
Session Affinity:  None
Events:            <none>
$
$
$
```



```
root@kubeadm-master:/home/ubuntu/Kubernetes# curl 10.106.199.130
curl: (7) Failed to connect to 10.106.199.130 port 80: Connection refused
```

4. Delete all the resources created in this lab exercise

```
$ kubectl delete -f readiness-pod.yaml
$ kubectl delete -f readiness-svc.yaml
```



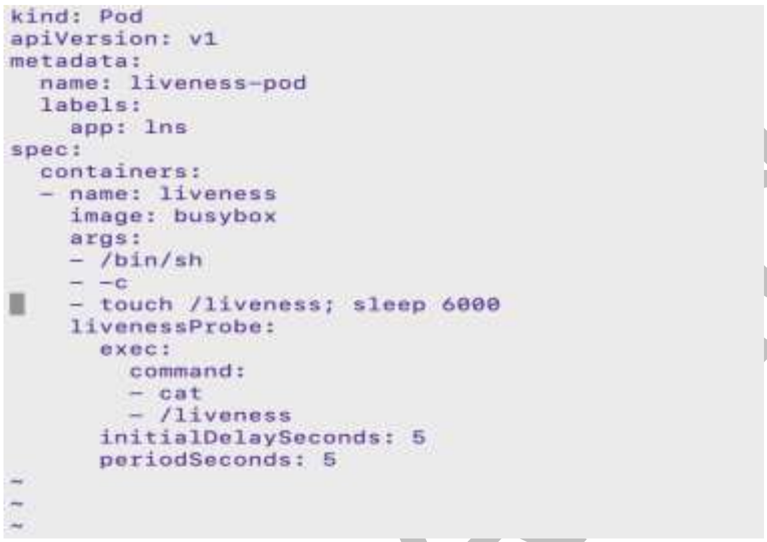
```
$
$ kubectl delete -f readiness-pod.yaml
pod "readiness-pod" deleted
kubectl delete -f readiness-svc.yaml

$ kubectl delete -f readiness-svc.yaml
service "readiness-svc" deleted
$
$
```

4 CONFIGURING LIVENESS PROBES FOR CONTAINER HEALTH CHECK

4.1 Create a Pod yaml with liveness probe

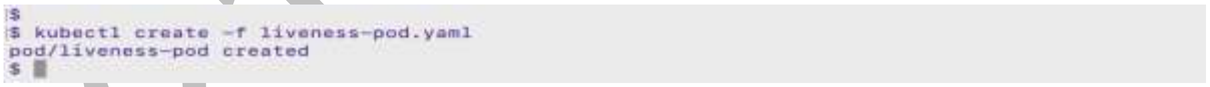
1. Create a Pod yaml and enter the content given below

```
$ vi liveness-pod.yaml
```



```
kind: Pod
apiVersion: v1
metadata:
  name: liveness-pod
  labels:
    app: lns
spec:
  containers:
  - name: liveness
    image: busybox
    args:
    - /bin/sh
    - -c
    - touch /liveness; sleep 6000
    livenessProbe:
      exec:
        command:
        - cat
        - /liveness
      initialDelaySeconds: 5
      periodSeconds: 5
~
~
~
```

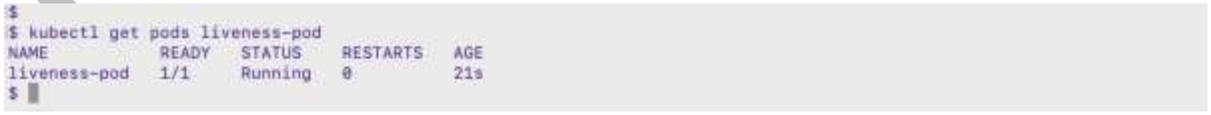
```
$ kubectl create -f liveness-pod.yaml
```



```
$
$ kubectl create -f liveness-pod.yaml
pod/liveness-pod created
$
```

2. Verfy that the Pod runs with no restarts

```
$ kubectl get pods liveness-pod
```

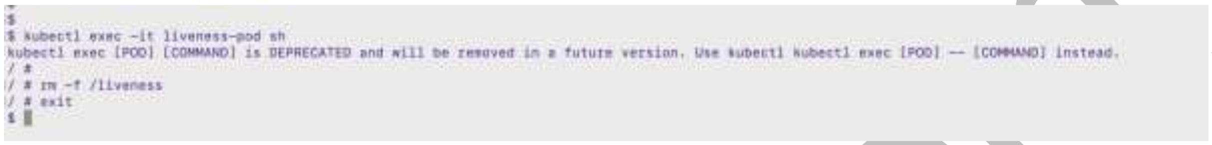


```
$
$ kubectl get pods liveness-pod
NAME           READY   STATUS    RESTARTS   AGE
liveness-pod   1/1     Running   0          21s
$
```

4.2 Simulating Liveness Probe failure

1. Get shell access to the container and delete the /liveness file

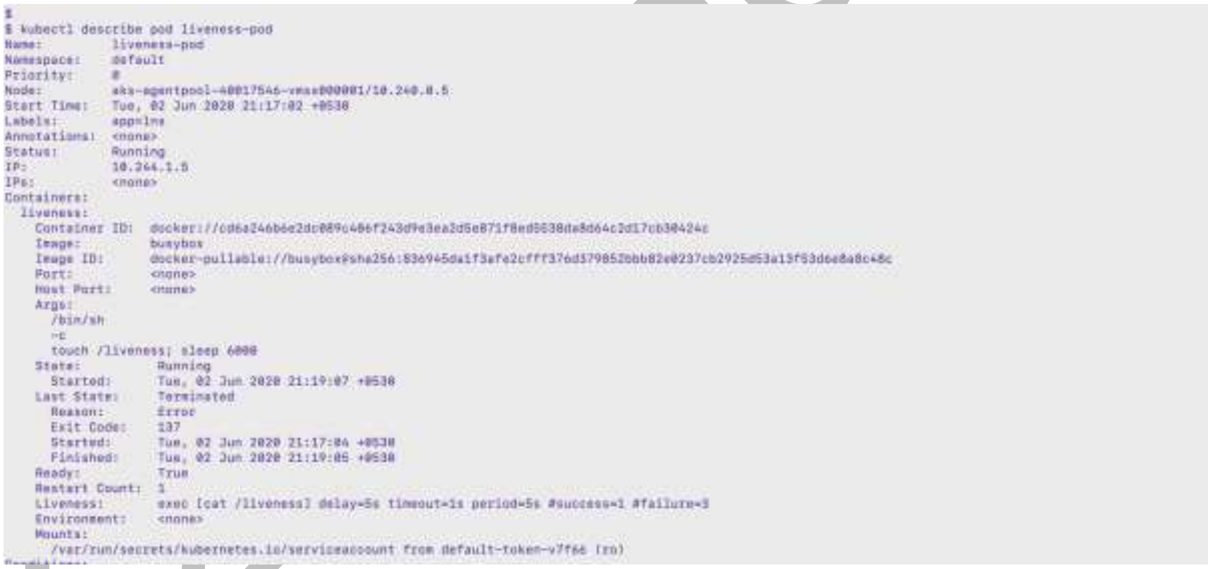
```
$ kubectl exec -it liveness-pod sh
# rm -f /liveness
# exit
```



```
$
$ kubectl exec -it liveness-pod sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl kubectl exec [POD] -- [COMMAND] instead.
/ #
/ # rm -f /liveness
/ # exit
$
```

2. View container events and see that the liveness probe has failed

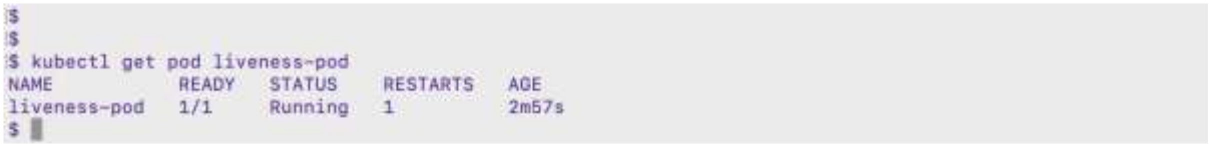
```
$ kubectl describe pod liveness-pod
```



```
$
$ kubectl describe pod liveness-pod
Name:         liveness-pod
Namespace:    default
Priority:     0
Node:         aks-agentpool-40017546-vmss000001/10.240.0.5
Start Time:   Tue, 02 Jun 2020 21:17:02 +0530
Labels:       app=lns
Annotations:  <none>
Status:       Running
IP:           10.244.1.5
IPs:          <none>
Containers:
  liveness:
    Container ID:  docker://cd6a246b6e2dc089c406f243d9e3ea2d5e871f8ed5638da8d64c2d17cb30424c
    Image:         busybox
    Image ID:      docker-pullable://busybox@sha256:836945da1f3afe2cfff376d379852bb82e0237cb2925d53a13f53d6e8a8c48c
    Port:          <none>
    Host Port:     <none>
    Args:
      /bin/sh
      -c
      touch /liveness; sleep 6000
    State:          Running
      Started:      Tue, 02 Jun 2020 21:19:07 +0530
    Last State:     Terminated
      Reason:       Error
      Exit Code:    137
      Started:      Tue, 02 Jun 2020 21:17:04 +0530
      Finished:     Tue, 02 Jun 2020 21:19:05 +0530
    Ready:          True
    Restart Count:  1
    Liveness:       exec [cat /liveness] delay=5s timeout=1s period=5s #success=1 #failure=3
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-v7f66 (ro)
```

3. View container status and see that it has restarted due to failed liveness probe

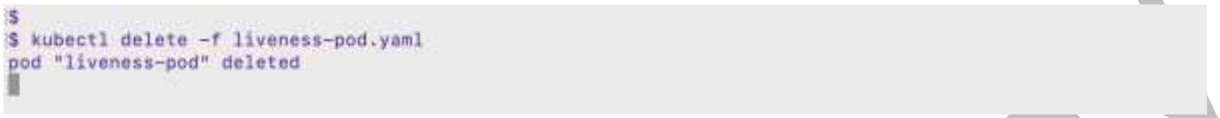
```
$ kubectl get pod liveness-pod
```



```
$
$
$ kubectl get pod liveness-pod
NAME           READY   STATUS    RESTARTS   AGE
liveness-pod   1/1     Running   1          2m57s
$
```

4.3 Delete all the resources created in this lab exercise

```
$ kubectl delete -f liveness-pod.yaml
```



```
$
$ kubectl delete -f liveness-pod.yaml
pod "liveness-pod" deleted
```

5 SUMMARY

In this guide we Covered:

1. Configuring Readiness Probes for Container Health Check

- Create a Pod with readiness probe and expose it with a service.
- Simulating Readiness Probe failure.
- Delete all the resources created in this lab exercise.

2. Configuring Liveness Probes for Container Health Check

- Create a Pod yaml with liveness probe.
- Simulating Liveness Probe failure.
- Delete all the resources created in this lab exercise.