Kubernetes Readiness and Liveness probes

Uses & How to not shoot yourself in the foot!

by

Muhammad Kamran Azeem

https://github.com/Praqma/k8s-probes-demo

$ whoami

Kamran Azeem

Masters degree in IT
CCNA, OCP (DBA), RHCE, CEH, CISSP, CKA

- Consultant
- Trainer
- Speaker
- Author

Agenda

- readinessProbe
- livenessProbe
- startupProbe
- Troublemaker - the simulation tool
- Demo
- Discussion, additional questions and answers - hopefully!

! The slides are densely packed with information because of the nature of the topic being discussed.

What are you expected to know already?

- Basic knowledge of the architecture of Kubernetes is expected
- Basic knowledge of various common Kubernetes objects is expected such as:
  - Deployments,
  - Replica Sets,
  - Pods,
  - Services,
  - EndPoints, etc.

Readiness Probe

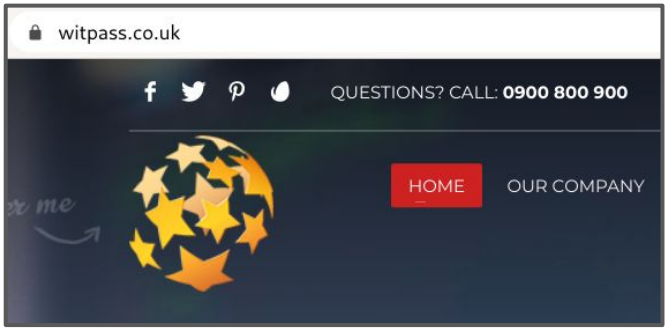
Whats wrong with having no Readiness probes?

```
$ cat site-deployment.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  name: witpass-co-uk
  labels:
    app: witpass-co-uk
spec:
  selector:
    matchLabels:
      app: witpass-co-uk
  template:
    metadata:
      labels:
app: witpass-co-uk
    spec:
      containers:
      - name: witpass-co-uk
image: eu.gcr.io/witline/witpass-co-uk
ports:
- containerPort: 80
  name: http
```

```
$ kubectl apply -f  site-deployment.yaml

$ kubectl get pods
witpass-co-uk-df8c8cbd8-gnfns  1/1  Running  0  5sec
```

10 seconds later

1 minute later



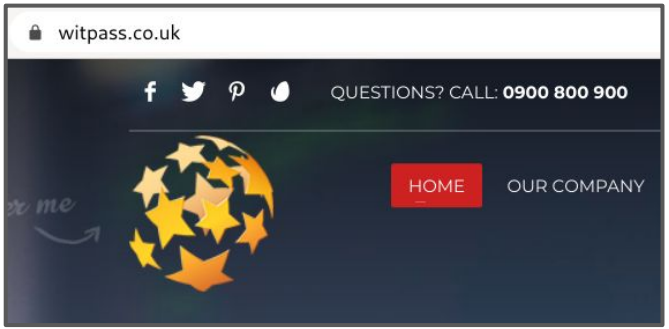
What is happening?

- The container in this example does some "*prep-work*" before the web-server process (apache) actually started.
- The prep-work takes somewhere between 20-50 seconds.
- Any attempt to access the service during the "prep-work" phase results in "Bad Gateway" (Error 502); though, Kubernetes declared the container "*Ready/Running*" as soon as deployment was created!
- Not ideal!

```
$ kubectl apply -f  site-deployment.yaml

$ kubectl get pods
witpass-co-uk-df8c8cbd8-gnfns  1/1  Running  0  5sec
```

10 seconds later

1 minute later



About deployment, service and endpoint

```
$ kubectl apply -f nginx.yaml
deployment.apps/nginx created
service/nginx created

$ kubectl get pods -w
NAME                     READY   STATUS    RESTARTS   AGE
nginx-5cf8cf645d-h9k4b   0/1     ContainerCreating   0          0s
nginx-5cf8cf645d-h9k4b   1/1     Running             0          18s

$ kubectl get svc -w
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
nginx        ClusterIP   10.104.74.63   <none>        80/TCP    0s

$ kubectl get endpoints -w
NAME         ENDPOINTS             AGE
nginx        <none>                0s
nginx        172.17.0.2:80         18s
```

The nginx service gets an endpoint as soon as the container is Running/Ready

Example of a Wordpress deployment

```
[kamran@kworkhorse k8s-probes-demo]$ kubectl apply -f wordpress-example.yaml
deployment.apps/wordpress-example created

[kamran@kworkhorse k8s-probes-demo]$ kubectl get pods -w
NAME                                READY   STATUS    RESTARTS   AGE
wordpress-example-7465474d98-64nsj   0/1     ContainerCreating   0          0s
wordpress-example-7465474d98-64nsj   1/1     Running             0          19s

[kamran@kworkhorse k8s-probes-demo]$ kubectl get svc -w
NAME                TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
wordpress-example   ClusterIP   10.104.243.107   <none>        80/TCP    0s

[kamran@kworkhorse k8s-probes-demo]$  kubectl get endpoints -w
NAME         ENDPOINTS             AGE
wordpress-example   <none>                1s
wordpress-example   172.17.0.2:80         19s

[kamran@kworkhorse k8s-probes-demo]$ kubectl logs -f wordpress-example-7465474d98-64nsj
WordPress not found in /var/www/html - copying now...
Complete! WordPress has been successfully copied to /var/www/html
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2.
Set the 'ServerName' directive globally to suppress this message
```

The service gets endpoint as soon as the container is Running/Ready.
(No readiness probes configured), even though the process responsible for serving incoming requests starts later

Copy operation takes about 1-2 minutes before Apache starts up , and starts serving requests

Kubernetes readinessProbe:

- With "readinessProbe", you only declare the container "Ready" when it is *really ready* to serve requests.
- A pod may be a single-container, or multi-container, but when readinessProbe fails for any single container inside a pod, the entire pod is considered "Not Ready".
- Examples:
  - Ready: 0/1 or 1/1
  - Ready: 0/2 or 1/2 or 2/2
- When a Pod is not ready, it is removed from its Service load balancers.
- Don't use small values for "**periodSeconds**", else your logs will fill up rapidly with useless "***probe***" entries.

```
. . .
readinessProbe:
  httpGet:
    path: /readinesscheck.txt
    port: 80

  # Wait time before the readinessProbe
  #   is tried for the first time.
  initialDelaySeconds: 5

  # Retry every X seconds (frequency):
  periodSeconds: 2

  # Number of times this probe can fail before k8s
  #   gives up and marks container as "Not Ready".
  # After 30 tries (or, 2x30=60 seconds),
  #   the container is marked "Unready".
  failureThreshold: 30

  # Mark container as "Ready", if readinessProbe has
  #   been successful at least once.
  successThreshold: 1
. . .
```

```
$ cat readiness-simple.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  name: readiness-simple
  labels:
    app: readiness-simple
spec:
  selector:
    matchLabels:
      app: readiness-simple
  template:
    metadata:
      labels:
        app: readiness-simple
    spec:
      containers:
      - name: readiness-simple
        image: kamranazeem/k8s-probes-demo:latest
        env:
        - name: START_DELAY
          value: "30"
        ports:
        - name: http
          containerPort: 80
```

A “feature” in this image - helps the container to start with a delay.

```
. . .
        resources:
          limits:
            cpu: 10m
            memory: 20Mi
          requests:
            cpu: 5m
            memory: 20Mi
        readinessProbe:
          httpGet:
            path: /readinesscheck.txt
            port: 80
          initialDelaySeconds: 5
          periodSeconds: 2
          failureThreshold: 30
          successThreshold: 1
```

This image, already contains this file.

Declare the container as “NotReady”, if readinessProbe does not succeed for 5s + (2s x 30) = 65s (seconds)

readinessProbe in action:

This container becomes **ready** after about 30 seconds have passed.

```
$ kubectl apply -f readiness-simple.yaml
deployment.apps/readiness-simple created

$ kubectl get pods -w
NAME                                 READY   STATUS              RESTARTS   AGE
readiness-simple-7d8d8f76bc-vw7f5    0/1     ContainerCreating   0          2s
readiness-simple-7d8d8f76bc-vw7f5    0/1     Running             0          6s
readiness-simple-7d8d8f76bc-vw7f5    1/1     Running             0          37s
```

```
$ kubectl logs -f readiness-simple-7d8d8f76bc-vw7f5
Container started at: 30-06-2020_09:47:42

START_DELAY is set - Simulating a slow-start container by
sleeping for 30 seconds ...

Web service started at: 30-06-2020_09:48:12
Exec-uting: nginx -g daemon off;

10.44.0.1 - - [30/Jun/2020:09:48:13 +0000] "GET
/probecheck.txt HTTP/1.1" 200 11 "-" "kube-probe/1.16+" "-"
```

```
$ kubectl get ep -l app=readiness-simple -w

NAME               ENDPOINTS       AGE
readiness-simple   <none>          0s
readiness-simple                   6s
readiness-simple   10.44.0.14:80   37s
```

More about readinessProbe:

- **readinessProbe** has been available in Kubernetes since version 1.0
- readinessprobe runs on the container during its entire lifecycle.
- There are different types of “readinessProbe”:
  - HTTPGet , TCPSocket, Exec
- Containers needing prep-work before they start serving, should employ readinessProbe.
- **Use cases:** Statefulsets scale one at the time, controlled by readiness. When no readiness probes are present, it will scale really fast, and might break the application. e.g. Scaling Confluence from 2 to 5 replicas. Same is true for increasing replicas of a deployment, or upgrading deployments. If no readiness probes are configured you risk exposing a service with **“non-ready”** endpoints.

```
readinessProbe:
  httpGet:
    path: /healthy
    port: 80
  initialDelaySeconds: 5
  periodSeconds: 5
```

```
readinessProbe:
  tcpSocket:
    port: 3306
  initialDelaySeconds: 5
  periodSeconds: 5
```

```
readinessProbe:
  exec:
    command:
    - cat
    - /tmp/healthy
  initialDelaySeconds: 5
  periodSeconds: 5
```

https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes

A little about this Docker image

Dockerfile

```
$ cat Dockerfile

From nginx:alpine

RUN echo "ready"  > /usr/share/nginx/html/readinesscheck.txt \
 && echo "alive"  > /usr/share/nginx/html/livenesscheck.txt  \
 && echo "started"  > /usr/share/nginx/html/startupcheck.txt


COPY troublemaker.sh /troublemaker.sh

COPY entrypoint.sh /entrypoint.sh

ENTRYPOINT ["/entrypoint.sh"]

CMD ["nginx", "-g", "daemon off;"]
```

Files used for readiness and liveness probes when this image is used.

They show up in container's web access logs.

Contains functions to continuously mess with readiness and liveness probes, with random intervals.

Simulates a really buggy/faulty application.

This script is used as "**entrypoint**" when in "**troublemaker**" mode.

The default "entrypoint". Contains logic to start a simple nginx web server with a configurable "delay".

Also creates a custom index.html file.

entrypoint.sh

```
$ cat entrypoint.sh
#!/bin/sh
. . .

if [ "${START_DELAY}" != "" ]; then
  echo "${TIMESTAMP} - START_DELAY is set - Simulating a slow-starting container by sleeping for
${START_DELAY} seconds ..."
  sleep ${START_DELAY}
else
  echo "${TIMESTAMP} - START_DELAY was not set, or was set to zero - not sleeping ..."
fi

. . .

TIMESTAMP=$(date +%d-%m-%Y_%T)
MESSAGE="${TIMESTAMP} - Kubernetes probes demo - Web service started"

# Run whatever was passed in CMD:
echo "Exec-uting: $@"
exec "$@"
```

Sleep for defined time interval

CMD ["nginx", "-g", "daemon off;"]

Liveness Probe

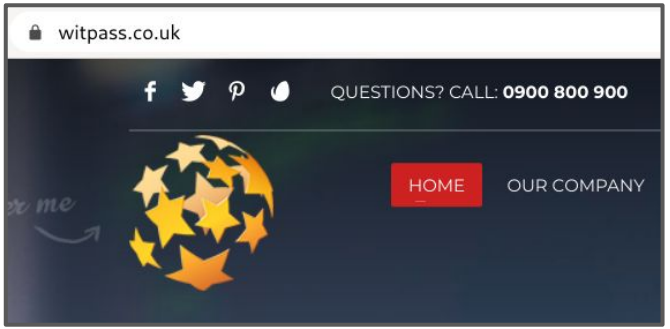
Whats wrong with having no Liveness probes?

```
$ cat site-deployment.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  name: witpass-co-uk
  labels:
    app: witpass-co-uk
spec:
  selector:
    matchLabels:
      app: witpass-co-uk
  template:
    metadata:
      labels:
app: witpass-co-uk
    spec:
      containers:
      - name: witpass-co-uk
image: eu.gcr.io/witline/witpass-co-uk
ports:
- containerPort: 80
  name: http
```

```
$ kubectl apply -f  site-deployment.yaml

$ kubectl get pods
witpass-co-uk-df8c8cbd8-gnfns  1/1  Running  0  3d
```

Kubernetes thinks the pod is healthy/ready.

1 minute later



3 days later

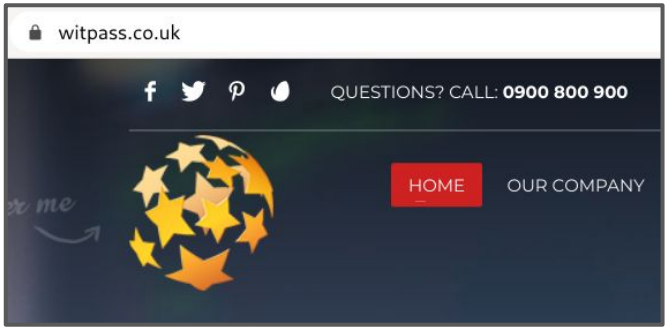
What is happening?

- (No readinessProbe/livenessProbe configured)
- Three days ago, the container in this example started properly, and started serving web requests.
- Something started leaking memory, and/or the main process hung up for some reason, or some deadlock occurred.
- Kubernetes still thinks it is “Ready/Running” and continues to send traffic to the pod, whereas the pod does not serve any incoming requests.
- Not ideal!

Kubernetes thinks the pod is healthy/ready.

```
$ kubectl apply -f  site-deployment.yaml

$ kubectl get pods
witpass-co-uk-df8c8cbd8-gnfns  1/1  Running  0  3d
```

1 minute later



3 days later

Kubernetes livenessProbe:

- The “livenessProbe” monitors *some aspect/component* of a running container to ensure it is doing what it is supposed to do. If not - it restarts it.
- When livenessProbe fails for any single container inside a pod, the entire pod is considered “Not Ready”, until the restarted container becomes “Ready”.
- The “Restart” shows how many times a container was restarted. (next slide)
- In real world the initialDelaySeconds for livenessProbe should be large enough to allow enough time for container to boot fully.
- The frequency (periodSeconds) should be a fairly large value - *unless you are running a really buggy/faulty application; then, you have a different problem to solve*.
- Also, the container will be restarted when it exceeds it’s CPU/mem resources. (not related to livenessProbe)

In this case, this could be a simple TCP check on port 80 too

```
. . .

livenessProbe:
  httpGet:
    path: /livenesscheck.txt
    port: 80

  # Waiting-time before the livenessProbe
  #   is tried for the first time.
  initialDelaySeconds: 5

  # Retry the probe every X seconds (frequency):
  periodSeconds: 3

  # Number of times this probe can fail before
  #   k8s gives up and "restarts" the
  #   container:
  failureThreshold: 1

. . .
```

$ cat liveness-simple.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: liveness-simple
  labels:
    app: liveness-simple
spec:
  selector:
    matchLabels:
      app: liveness-simple
  template:
    metadata:
      labels:
        app: liveness-simple
    spec:
      containers:
      - name: liveness-simple
        image: kamranazeem/k8s-probes-demo:latest
        env:
        - name: START_DELAY
          value: "30"
        ports:
        - name: http
          containerPort: 80
```

```
. . .

        resources:
          limits:
            cpu: 10m
            memory: 20Mi
          requests:
            cpu: 5m
            memory: 20Mi

        livenessProbe:
          httpGet:
            path: /livenesscheck.txt
            port: 80
          initialDelaySeconds: 5
          periodSeconds: 3
          failureThreshold: 1
```

Kubernetes reboots the container, if livenessProbe fails even once - after 5s + 3s = 8s (seconds)

livenessProbe in action:

Absence of readinessProbe results in Kubernetes putting it in **ready** state as soon as it is (re-)started.

Continuously restarting!

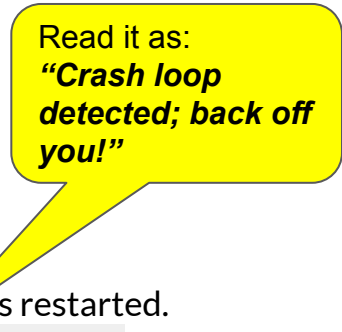
```
$ kubectl apply -f  liveness-simple.yaml
deployment.apps/liveness-simple created
service/liveness-simple created

$ kubectl get pods -w
NAME                                READY   STATUS             RESTARTS   AGE
liveness-simple-56b6bc5b64-4j6xd    1/1     Running            0          8s
liveness-simple-56b6bc5b64-4j6xd    1/1     Running            1          47s
liveness-simple-56b6bc5b64-4j6xd    1/1     Running            2          86s
liveness-simple-56b6bc5b64-4j6xd    1/1     Running            3          2m5s
liveness-simple-56b6bc5b64-4j6xd    1/1     Running            4          2m44s
liveness-simple-56b6bc5b64-4j6xd    0/1     CrashLoopBackOff   4          3m20s
liveness-simple-56b6bc5b64-4j6xd    1/1     Running            5          4m5s
```

- Above, the container's main process starts after 30 seconds. There is no readinessProbe, so every time kubernetes starts it, it thinks container is *Ready*.
- The livenessProbe kicks in after 5 seconds of container is declared *Ready*, probe gets a non-zero exit-code (as HTTP is not responding yet), and the container is restarted.
- This setup results in the container never able to start fully!

```
$ kubectl get ep -l app=liveness-simple -w
NAME               ENDPOINTS   AGE
liveness-simple    <none>      0s
liveness-simple    10.44.0.16:80   7s
liveness-simple                    3m20s
liveness-simple    10.44.0.16:80   4m5s
liveness-simple                    5m20s
```

More about livenessProbe:

- livenessProbe has been available in Kubernetes since version 1.0
- livenessProbe runs on the container during its entire lifecycle.
- There are different types of “livenessProbe”:
  - HTTPGet , TCPSocket, Exec
- If livenessProbe returns any non-zero exit code, the container is restarted.
- If the HTTP livenessProbe’s return-code is between 200-400, the container is healthy, else it is restarted.
- If kubelet sees that the containers is restarting too many times, the pod goes into "**CrashLoopBackOff**" state. i.e. scheduler applies time constraints for repeatedly trying again.
- **Use case:** Containers which do not have internal mechanism to handle crashes and deadlocks, or which have defects (bugs) in them, should employ livenessProbe.




```
livenessProbe:
  httpGet:
    path: /healthy
    port: 80
  initialDelaySeconds: 5
  periodSeconds: 5
```

```
livenessProbe:
  tcpSocket:
    port: 3306
  initialDelaySeconds: 5
  periodSeconds: 10
```

```
livenessProbe:
  exec:
    command:
    - cat
    - /tmp/healthy
  initialDelaySeconds: 5
  periodSeconds: 5
```

https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes

Using Readiness and Liveness Probes together

Use readiness and liveness probes together:

- readinessProbe simply attaches or removes the pod from it's service.
- The "livenessProbe" restarts a non-responsive container.
- Use two in combination so the service only starts accepting connections when the pod is ***really* Ready**; and the container is restarted if it becomes non-responsive, etc.
- readinessProbe and livenessProbe ***can*** perform two different types of checks.
- set initialDelaySeconds of livenessProbe to large enough value, to allow the container to be ready first, otherwise you will have **timing conflict**.

```
. . .

readinessProbe:
  httpGet:
    path: /license.txt
    port: 80
  initialDelaySeconds: 5
  periodSeconds: 2
  failureThreshold: 10
  successThreshold: 1

livenessProbe:
  tcpSocket:
    port: 80
  initialDelaySeconds: 30
  periodSeconds: 5
  failureThreshold: 1

. . .
```

A file which already exists in your htdocs, e.g. wordpress

Check if the server responds on port 80 or not

readinessProbe + livenessProbe - in action:

```
$ kubectl apply -f readiness-liveness.yaml
deployment.apps/readiness-liveness created
service/readiness-liveness created

$ kubectl get pods -w

NAME                                   READY   STATUS              RESTARTS   AGE
readiness-liveness-65896c78d5-b8gf6   0/1     ContainerCreating   0          0s
readiness-liveness-65896c78d5-b8gf6   0/1     Running             0          16s
readiness-liveness-65896c78d5-b8gf6   1/1     Running             0          28s
```

```
$ kubectl get svc -l app=readiness-liveness -w

NAME                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
readiness-liveness   ClusterIP   10.98.120.110   <none>        80/TCP    0s
```

```
$ kubectl get endpoints \
  -l app=readiness-liveness -w

NAME                ENDPOINTS          AGE
readiness-liveness   <none>             0s
readiness-liveness                      16s
readiness-liveness   172.17.0.2:80      28s
```

Check logs for more understanding:

```
$ kubectl logs -f readiness-liveness-65896c78d5-b8gf6

17-09-2020_10:43:07 - Container started
17-09-2020_10:43:07 - START_DELAY is set to 10 - Simulating a slow-starting container by sleeping for 10 seconds
...
17-09-2020_10:43:18 - Kubernetes probes demo - Web service started
Exec-uting: nginx -g daemon off;
172.17.0.1 - - [17/Sep/2020:10:43:19 +0000] "GET /readinesscheck.txt HTTP/1.1" 200 6 "-" "kube-probe/1.18" "-"
172.17.0.1 - - [17/Sep/2020:10:43:21 +0000] "GET /readinesscheck.txt HTTP/1.1" 200 6 "-" "kube-probe/1.18" "-"
172.17.0.1 - - [17/Sep/2020:10:43:23 +0000] "GET /readinesscheck.txt HTTP/1.1" 200 6 "-" "kube-probe/1.18" "-"
172.17.0.1 - - [17/Sep/2020:10:43:25 +0000] "GET /readinesscheck.txt HTTP/1.1" 200 6 "-" "kube-probe/1.18" "-"
172.17.0.1 - - [17/Sep/2020:10:43:27 +0000] "GET /readinesscheck.txt HTTP/1.1" 200 6 "-" "kube-probe/1.18" "-"
172.17.0.1 - - [17/Sep/2020:10:43:29 +0000] "GET /readinesscheck.txt HTTP/1.1" 200 6 "-" "kube-probe/1.18" "-"
172.17.0.1 - - [17/Sep/2020:10:43:31 +0000] "GET /readinesscheck.txt HTTP/1.1" 200 6 "-" "kube-probe/1.18" "-"
172.17.0.1 - - [17/Sep/2020:10:43:33 +0000] "GET /readinesscheck.txt HTTP/1.1" 200 6 "-" "kube-probe/1.18" "-"
172.17.0.1 - - [17/Sep/2020:10:43:35 +0000] "GET /readinesscheck.txt HTTP/1.1" 200 6 "-" "kube-probe/1.18" "-"
172.17.0.1 - - [17/Sep/2020:10:43:37 +0000] "GET /livenesscheck.txt HTTP/1.1" 200 6 "-" "kube-probe/1.18" "-"
172.17.0.1 - - [17/Sep/2020:10:43:38 +0000] "GET /readinesscheck.txt HTTP/1.1" 200 6 "-" "kube-probe/1.18" "-"
172.17.0.1 - - [17/Sep/2020:10:43:39 +0000] "GET /readinesscheck.txt HTTP/1.1" 200 6 "-" "kube-probe/1.18" "-"
172.17.0.1 - - [17/Sep/2020:10:43:41 +0000] "GET /readinesscheck.txt HTTP/1.1" 200 6 "-" "kube-probe/1.18" "-"
172.17.0.1 - - [17/Sep/2020:10:43:42 +0000] "GET /livenesscheck.txt HTTP/1.1" 200 6 "-" "kube-probe/1.18" "-"
172.17.0.1 - - [17/Sep/2020:10:43:43 +0000] "GET /readinesscheck.txt HTTP/1.1" 200 6 "-" "kube-probe/1.18" "-"
172.17.0.1 - - [17/Sep/2020:10:43:45 +0000] "GET /readinesscheck.txt HTTP/1.1" 200 6 "-" "kube-probe/1.18" "-"
```

Readiness and Liveness probes with troublemaker

```
$ cat readiness-liveness-with-troublemaker.yaml
apiVersion: apps/v1
kind: Deployment
. . .
    spec:
      volumes:
      - name: shared-directory
        emptyDir: {}
      containers:
      - name: troublemaker
        image: kamranazeem/k8s-probes-demo:latest
        command: ["/troublemaker.sh"]
        args: ["nginx", "-g", "daemon off;"]
        env:
        - name: ROLE
          value: "TROUBLEMAKER"
        volumeMounts:
        - name: shared-directory
          mountPath: /shared
. . .
      - name: probes-demo
        image: kamranazeem/k8s-probes-demo:latest
        env:
        - name: START_DELAY
          value: "10"
```

The "troublemaker"

```
. . .
        readinessProbe:
          httpGet:
            path: /readinesscheck.txt
            port: 80
          initialDelaySeconds: 5
          periodSeconds: 2
          failureThreshold: 10
          successThreshold: 1

        livenessProbe:
          httpGet:
            path: /livenesscheck.txt
            port: 80
          initialDelaySeconds: 30
          periodSeconds: 5
          failureThreshold: 1

        volumeMounts:
        - name: shared-directory
          mountPath: /usr/share/nginx/html
```

Notice large value for **initialDelaySeconds** for **livenessProbe**

readinessProbe + livenessProbe + troublemaker - in action:

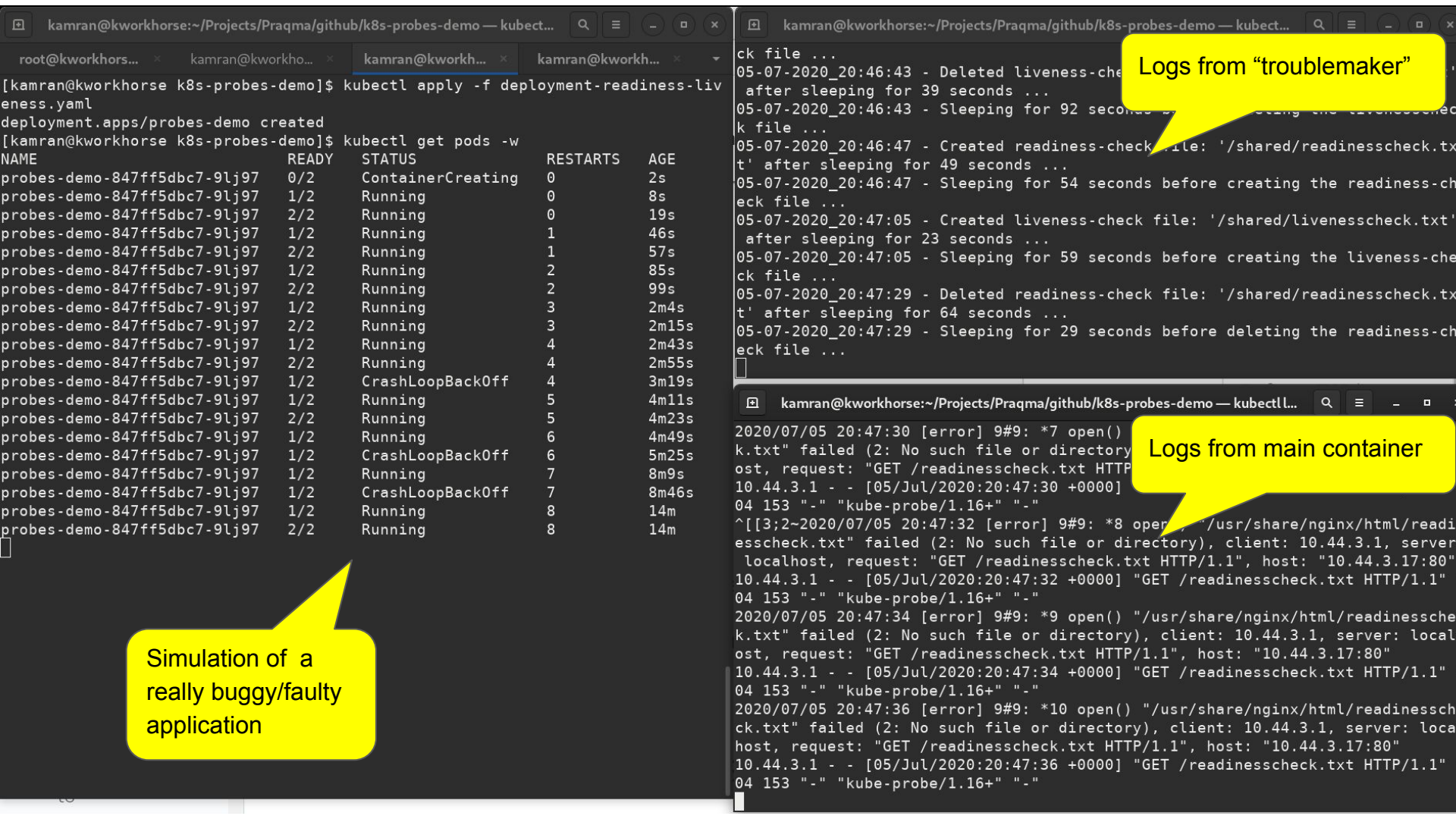
```
[kamran@kworkhorse k8s-probes-demo]$ kubectl apply -f readiness-liveness-with-troublemaker.yaml
deployment.apps/readiness-liveness-with-troublemaker created
service/readiness-liveness-with-troublemaker created

[kamran@kworkhorse k8s-probes-demo]$ kubectl get pods -w
NAME                                                READY   STATUS              RESTARTS   AGE
readiness-liveness-with-troublemaker-64bf55dffd-tk4sb   0/2     ContainerCreating   0          2s
readiness-liveness-with-troublemaker-64bf55dffd-tk4sb   1/2     Running             0          20s
readiness-liveness-with-troublemaker-64bf55dffd-tk4sb   2/2     Running             0          31s
readiness-liveness-with-troublemaker-64bf55dffd-tk4sb   1/2     Running             1          60s
readiness-liveness-with-troublemaker-64bf55dffd-tk4sb   2/2     Running             1          93s
readiness-liveness-with-troublemaker-64bf55dffd-tk4sb   1/2     Running             2          2m5s
readiness-liveness-with-troublemaker-64bf55dffd-tk4sb   2/2     Running             2          2m17s
```

Simulation of a really buggy/faulty application

```
$ kubectl get endpoints -l app=readiness-liveness-with-troublemaker -w
NAME                                  ENDPOINTS   AGE
readiness-liveness-with-troublemaker   <none>      0s
readiness-liveness-with-troublemaker               20s
readiness-liveness-with-troublemaker   172.17.0.2:80   31s
readiness-liveness-with-troublemaker               60s
readiness-liveness-with-troublemaker   172.17.0.2:80   93s
readiness-liveness-with-troublemaker               2m5s
readiness-liveness-with-troublemaker   172.17.0.2:80   2m17s
```

Each time the container restarts, it is removed from the service; and each time the readinessProbe succeeds, (after restart), it is **ready**, thus it is connected to the service.

```
[kamran@kworkhorse k8s-probes-demo]$ kubectl get svc -l app=readiness-liveness-with-troublemaker -w

NAME                                  TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
readiness-liveness-with-troublemaker   ClusterIP   10.105.132.24   <none>        80/TCP    0s
```


kamran@kworkhorse:~/Projects/Praqma/github/k8s-probes-demo — kubect...
root@kworkhors...
kamran@kworkho...
kamran@kworkh...
kamran@kworkh...
[kamran@kworkhorse k8s-probes-demo]$ kubectl apply -f deployment-readiness-liveness.yaml
deployment.apps/probes-demo created
[kamran@kworkhorse k8s-probes-demo]$ kubectl get pods -w
NAME READY STATUS RESTARTS AGE
probes-demo-847ff5dbc7-9lj97 0/2 ContainerCreating 0 2s
probes-demo-847ff5dbc7-9lj97 1/2 Running 0 8s
probes-demo-847ff5dbc7-9lj97 2/2 Running 0 19s
probes-demo-847ff5dbc7-9lj97 1/2 Running 1 46s
probes-demo-847ff5dbc7-9lj97 2/2 Running 1 57s
probes-demo-847ff5dbc7-9lj97 1/2 Running 2 85s
probes-demo-847ff5dbc7-9lj97 2/2 Running 2 99s
probes-demo-847ff5dbc7-9lj97 1/2 Running 3 2m4s
probes-demo-847ff5dbc7-9lj97 2/2 Running 3 2m15s
probes-demo-847ff5dbc7-9lj97 1/2 Running 4 2m43s
probes-demo-847ff5dbc7-9lj97 2/2 Running 4 2m55s
probes-demo-847ff5dbc7-9lj97 1/2 CrashLoopBackOff 4 3m19s
probes-demo-847ff5dbc7-9lj97 1/2 Running 5 4m11s
probes-demo-847ff5dbc7-9lj97 2/2 Running 5 4m23s
probes-demo-847ff5dbc7-9lj97 1/2 Running 6 4m49s
probes-demo-847ff5dbc7-9lj97 1/2 CrashLoopBackOff 6 5m25s
probes-demo-847ff5dbc7-9lj97 1/2 Running 7 8m9s
probes-demo-847ff5dbc7-9lj97 1/2 CrashLoopBackOff 7 8m46s
probes-demo-847ff5dbc7-9lj97 1/2 Running 8 14m
probes-demo-847ff5dbc7-9lj97 2/2 Running 8 14m
Simulation of a really buggy/faulty application
kamran@kworkhorse:~/Projects/Praqma/github/k8s-probes-demo — kubect...
Logs from “troublemaker”
ck file ...
05-07-2020_20:46:43 - Deleted liveness-che
after sleeping for 39 seconds ...
05-07-2020_20:46:43 - Sleeping for 92 seco
k file ...
05-07-2020_20:46:47 - Created readiness-check file: '/shared/readinesscheck.tx
t' after sleeping for 49 seconds ...
05-07-2020_20:46:47 - Sleeping for 54 seconds before creating the readiness-ch
eck file ...
05-07-2020_20:47:05 - Created liveness-check file: '/shared/livenesscheck.txt'
after sleeping for 23 seconds ...
05-07-2020_20:47:05 - Sleeping for 59 seconds before creating the liveness-che
ck file ...
05-07-2020_20:47:29 - Deleted readiness-check file: '/shared/readinesscheck.tx
t' after sleeping for 64 seconds ...
05-07-2020_20:47:29 - Sleeping for 29 seconds before deleting the readiness-ch
eck file ...
kamran@kworkhorse:~/Projects/Praqma/github/k8s-probes-demo — kubectl l...
Logs from main container
2020/07/05 20:47:30 [error] 9#9: *7 open()
k.txt" failed (2: No such file or directory
ost, request: "GET /readinesscheck.txt HTTP
10.44.3.1 - - [05/Jul/2020:20:47:30 +0000]
04 153 "-" "kube-probe/1.16+" "-"
^[[3;2~2020/07/05 20:47:32 [error] 9#9: *8 open
esscheck.txt" failed (2: No such file or directory), client: 10.44.3.1, server
localhost, request: "GET /readinesscheck.txt HTTP/1.1", host: "10.44.3.17:80"
10.44.3.1 - - [05/Jul/2020:20:47:32 +0000] "GET /readinesscheck.txt HTTP/1.1"
04 153 "-" "kube-probe/1.16+" "-"
2020/07/05 20:47:34 [error] 9#9: *9 open() "/usr/share/nginx/html/readinessche
k.txt" failed (2: No such file or directory), client: 10.44.3.1, server: local
ost, request: "GET /readinesscheck.txt HTTP/1.1", host: "10.44.3.17:80"
10.44.3.1 - - [05/Jul/2020:20:47:34 +0000] "GET /readinesscheck.txt HTTP/1.1"
04 153 "-" "kube-probe/1.16+" "-"
2020/07/05 20:47:36 [error] 9#9: *10 open() "/usr/share/nginx/html/readinessch
ck.txt" failed (2: No such file or directory), client: 10.44.3.1, server: loca
host, request: "GET /readinesscheck.txt HTTP/1.1", host: "10.44.3.17:80"
10.44.3.1 - - [05/Jul/2020:20:47:36 +0000] "GET /readinesscheck.txt HTTP/1.1"
04 153 "-" "kube-probe/1.16+" "-"

Startup Probes

Available since v1.16+

Ver 1.16.x - Ver 1.17.x : Alpha feature
Ver 1.18.x : Beta feature

Why use Startup probes?

- Readiness and liveness probes may interfere with startup of a slow starting legacy application.
- **Startup probe** disables liveness and readiness checks until it succeeds, making sure those probes don't interfere with the application startup.
- Set up a startup probe with the same (HTTP or TCP) check, with `failureThreshold x periodSeconds` long enough to cover the worse case startup time.
- Since startupProbes is an alpha feature, you can only test it/see it in action if you enable alpha features on your cluster. Otherwise startupProbe is ignored.

```
. . .
readinessProbe:
  httpGet:
    path: /license.txt
    port: 80
  initialDelaySeconds: 5
  periodSeconds: 2
  failureThreshold: 10
  successThreshold: 1

livenessProbe:
  tcpSocket:
    port: 80
  initialDelaySeconds: 10
  periodSeconds: 5
  failureThreshold: 1

startupProbe:
  tcpSocket:
    port: 80
  periodSeconds: 5
  failureThreshold: 24
. . .
```

Disable other probes for 5 x 24 = 120 seconds

startupProbe on a k8s 1.16+ cluster

```
$ kubectl version
Client Version: version.Info{Major:"1", Minor:"18", GitVersion:"v1.18.5",
GitCommit:"e6503f8d8f769ace2f338794c914a96fc335df0f", GitTreeState:"clean",
BuildDate:"2020-06-26T03:47:41Z", GoVersion:"go1.13.9", Compiler:"gc",
Platform:"linux/amd64"}

Server Version: version.Info{Major:"1", Minor:"16+", GitVersion:"v1.16.9-gke.6",
GitCommit:"14bc8ad5f8c245f1240a8e4eab128c4d51bfeffe", GitTreeState:"clean",
BuildDate:"2020-05-30T02:07:03Z", GoVersion:"go1.13.9b4", Compiler:"gc",
Platform:"linux/amd64"}
```

startupProbe not working

kamran@kworkhorse:~/Projects/Praqma/github/k8s-probes-demo — kubect...

root@kworkhors... | kamran@kworkho... | kamran@kworkh... | kamran@kworkh...

```
[kamran@kworkhorse k8s-probes-demo]$ kubectl apply -f deployment-startup-probe
-w-other-probes.yaml
deployment.apps/probes-demo created
[kamran@kworkhorse k8s-probes-demo]$ kubectl get pods -w
NAME                            READY   STATUS              RESTARTS   AGE
probes-demo-76f55d5947-58nk6    0/1     ContainerCreating   0          3s
probes-demo-76f55d5947-58nk6    0/1     Running             0          8s
probes-demo-76f55d5947-58nk6    0/1     Running             1          59s
probes-demo-76f55d5947-58nk6    0/1     Running             2          109s
```

kamran@kworkhorse:~/Projects/Praqma/github/k8s-probes-demo

```
[kamran@kworkhorse k8s-probes-demo]$ kubectl logs -f  probes-demo-76f55d5947-5
8nk6
Container started at: 05-07-2020_21:49:10
START_DELAY is set - Simulating a slow-start container by sleeping for 60 seco
nds ...
[kamran@kworkhorse k8s-probes-demo]$ kubectl logs -f  probes-demo-76f55d5947-5
8nk6
Container started at: 05-07-2020_21:50:01
START_DELAY is set - Simulating a slow-start container by sleeping for 60 seco
nds ...
[kamran@kworkhorse k8s-probes-demo]$
```

Evidence that startupProbe is being ignored

K8s 1.16+

kamran@kworkhorse:~/Projects/Praqma/github/k8s-probes-demo — kubect...

root@kworkhors... | kamran@kworkho... | kamran@kworkh... | kamran@kworkh...

```
[kamran@kworkhorse k8s-probes-demo]$ kubectl apply -f deployment-startup-probe
-w-other-probes.yaml
deployment.apps/probes-demo created
[kamran@kworkhorse k8s-probes-demo]$ kubectl get pods -w
NAME                           READY   STATUS              RESTARTS   AGE
probes-demo-76f55d5947-tmjbx   0/1     ContainerCreating   0          2s
probes-demo-76f55d5947-tmjbx   0/1     Running             0          7s
```

kamran@kworkhorse:~/Projects/Praqma/github/k8s-probes-demo — watch ...

```
Every 2.0s: kubectl d...  kworkhorse.oslo.praqma.com: Mon Jul  6 00:00:23 2020

Events:
  Type     Reason     Age               From
                      Message
  ----     ------     ----              ----
                      -------
  Normal   Scheduled  34s               default-scheduler
                      Successfully assigned default/probes-demo-76f55d5947-tmj
bx to gke-witline-production-default-pool-bb67b489-p7a0
  Normal   Pulling    31s               kubelet, gke-witline-production-defaul
t-pool-bb67b489-p7a0  Pulling image "kamranazeem/k8s-probes-demo:latest"
  Normal   Pulled     30s               kubelet, gke-witline-production-defaul
t-pool-bb67b489-p7a0  Successfully pulled image "kamranazeem/k8s-probes-demo:l
atest"
  Normal   Created    30s               kubelet, gke-witline-production-defaul
t-pool-bb67b489-p7a0  Created container probes-demo
  Normal   Started    28s               kubelet, gke-witline-production-defaul
t-pool-bb67b489-p7a0  Started container probes-demo
  Warning  Unhealthy  11s               kubelet, gke-witline-production-defaul
t-pool-bb67b489-p7a0  Liveness probe failed: Get http://10.44.0.26:80/liveness
check.txt: dial tcp 10.44.0.26:80: connect: connection refused
  Normal   Killing    11s               kubelet, gke-witline-production-defaul
t-pool-bb67b489-p7a0  Container probes-demo failed liveness probe, will be res
tarted
  Warning  Unhealthy  1s (x5 over 21s)  kubelet, gke-witline-production-defaul
t-pool-bb67b489-p7a0  Readiness probe failed: Get http://10.44.0.26:80/readine
sscheck.txt: dial tcp 10.44.0.26:80: connect: connection refused
```

Enable feature-gate for alpha components

Each Kubernetes component lets you enable or disable a set of feature gates that are relevant to that component. To set feature gates for a component, such as kubelet, use the `--feature-gates` flag assigned to a list of feature pairs. e.g. `--feature-gates="...,DynamicKubeletConfig=true"`

For K8s 1.16.x & 1.17.x

```
$  minikube start --feature-gates="StartupProbe=true" --vm-driver=kvm2
```

Example without using Startup probes

K8s 1.18.3 (minikube)

```
$ kubectl apply -f readiness-liveness-without-startup-probe.yaml ; kubectl get pods -w
NAME                                                     READY   STATUS              RESTARTS   AGE
readiness-liveness-without-startup-probe-56cbcb794f-dhbwn   0/1     ContainerCreating   0          3s
readiness-liveness-without-startup-probe-56cbcb794f-dhbwn   0/1     Running             0          28s
readiness-liveness-without-startup-probe-56cbcb794f-dhbwn   1/1     Running             0          64s
readiness-liveness-without-startup-probe-56cbcb794f-dhbwn   0/1     Running             1          86s
readiness-liveness-without-startup-probe-56cbcb794f-dhbwn   1/1     Running             1          2m4s
readiness-liveness-without-startup-probe-56cbcb794f-dhbwn   0/1     Running             2          2m25s
```

```
$ cat readiness-liveness-without-startup-probe.yaml
apiVersion: apps/v1
kind: Deployment
. . .
spec:
. . .
    spec:
      containers:
      - name: readiness-liveness-without-startup-probe
        image: kamranazeem/k8s-probes-demo:latest
        env:
        - name: START_DELAY
          value: "30"
        ports:
        - name: http
          containerPort: 80
```

```
. . .
        readinessProbe:
          httpGet:
            path: /readinesscheck.txt
            port: 80
          initialDelaySeconds: 5
          periodSeconds: 5
          failureThreshold: 24
          successThreshold: 1

        livenessProbe:
          httpGet:
            path: /livenesscheck.txt
            port: 80
          initialDelaySeconds: 10
          periodSeconds: 10
          timeoutSeconds: 1
          failureThreshold: 1
```

The “**timing conflict**” between readiness and liveness probes result in container restarting continuously.

Example using Startup probes

K8s 1.18.3 (minikube)

```
[kamran@kworkhorse k8s-probes-demo]$ kubectl apply -f readiness-liveness-with-startup-probe.yaml
deployment.apps/readiness-liveness-with-startup-probe created
[kamran@kworkhorse k8s-probes-demo]$ kubectl get pods -w
NAME                                                     READY   STATUS              RESTARTS   AGE
readiness-liveness-with-startup-probe-677d467bfd-b6h8v   0/1     ContainerCreating   0          6s
readiness-liveness-with-startup-probe-677d467bfd-b6h8v   0/1     Running             0          38s
readiness-liveness-with-startup-probe-677d467bfd-b6h8v   0/1     Running             0          75s
readiness-liveness-with-startup-probe-677d467bfd-b6h8v   1/1     Running             0          79s
```

```
$ cat readiness-liveness-without-startup-probe.yaml
apiVersion: apps/v1
kind: Deployment
. . .
      containers:
      - name: readiness-liveness-without-startup-probe
        image: kamranazeem/k8s-probes-demo:latest
        env:
        - name: START_DELAY
          value: "30"
. . .
        readinessProbe:
          httpGet:
            path: /readinesscheck.txt
            port: 80
          initialDelaySeconds: 5
          periodSeconds: 5
          failureThreshold: 24
          successThreshold: 1
```

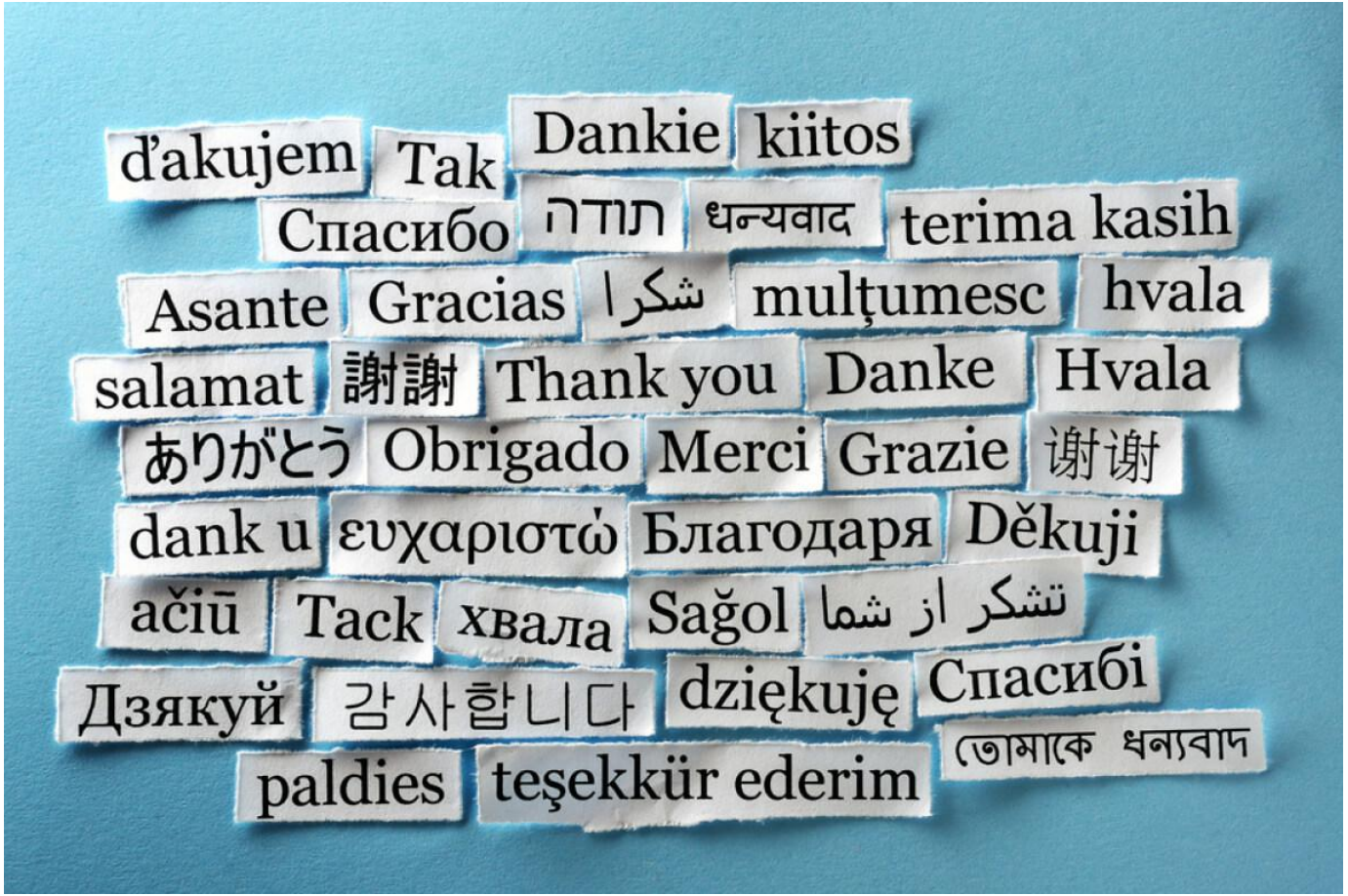
```
. . .

        livenessProbe:
          httpGet:
            path: /livenesscheck.txt
            port: 80
          initialDelaySeconds: 10
          periodSeconds: 10
          timeoutSeconds: 1
          failureThreshold: 1

        startupProbe:
          httpGet:
            path: /startupcheck.txt
            port: 80
          periodSeconds: 5
          failureThreshold: 12
          timeoutSeconds: 1
```

Startup probe disables readiness and liveness probes in the beginning, so their conflict does not prevent container to start properly.

Example using Startup probes

```
$ kubectl logs -f readiness-liveness-with-startup-probe-677d467bfd-b6h8v
Container started at: 16-09-2020_10:44:37
16-09-2020_10:44:37 - START_DELAY is set - Simulating a slow-start container by sleeping for 30 seconds ...
Web service started at: 16-09-2020_10:45:07
Exec-uting: nginx -g daemon off;
172.17.0.1 - - [16/Sep/2020:10:45:14 +0000] "GET /startupcheck.txt HTTP/1.1" 200 8 "-" "kube-probe/1.18" "-"
172.17.0.1 - - [16/Sep/2020:10:45:18 +0000] "GET /livenesscheck.txt HTTP/1.1" 200 14 "-" "kube-probe/1.18" "-"
172.17.0.1 - - [16/Sep/2020:10:45:18 +0000] "GET /readinesscheck.txt HTTP/1.1" 200 15 "-" "kube-probe/1.18" "-"
172.17.0.1 - - [16/Sep/2020:10:45:23 +0000] "GET /readinesscheck.txt HTTP/1.1" 200 15 "-" "kube-probe/1.18" "-"
172.17.0.1 - - [16/Sep/2020:10:45:28 +0000] "GET /livenesscheck.txt HTTP/1.1" 200 14 "-" "kube-probe/1.18" "-"
172.17.0.1 - - [16/Sep/2020:10:45:28 +0000] "GET /readinesscheck.txt HTTP/1.1" 200 15 "-" "kube-probe/1.18" "-"
172.17.0.1 - - [16/Sep/2020:10:45:33 +0000] "GET /readinesscheck.txt HTTP/1.1" 200 15 "-" "kube-probe/1.18" "-"
172.17.0.1 - - [16/Sep/2020:10:45:38 +0000] "GET /livenesscheck.txt HTTP/1.1" 200 14 "-" "kube-probe/1.18" "-"
172.17.0.1 - - [16/Sep/2020:10:45:38 +0000] "GET /readinesscheck.txt HTTP/1.1" 200 15 "-" "kube-probe/1.18" "-"
172.17.0.1 - - [16/Sep/2020:10:45:43 +0000] "GET /readinesscheck.txt HTTP/1.1" 200 15 "-" "kube-probe/1.18" "-"
```

Startup probe disables readiness and liveness probes in the beginning, so their conflict does not prevent container to start properly.


ďakujem
Tak
Dankie
kiitos
Спасибо
תודה
धन्यवाद
terima kasih
Asante
Gracias
شکرا
mulţumesc
hvala
salamat
謝謝
Thank you
Danke
Hvala
ありがとう
Obrigado
Merci
Grazie
谢谢
dank u
ευχαριστώ
Благодаря
Děkuji
ačiū
Tack
хвала
Sağol
تشکر از شما
Дзякуй
감사합니다
dziękuję
Спасибі
paldies
teşekkür ederim
তোমাকে ধন্যবাদ