

Liveness, Readiness, Volume

Цели работы:

- изучить концепции Liveness и Readiness probes
- освоить различные типы проверок: exec, httpGet, TCP
- научиться настраивать Volume для хранения данных
- понять разницу между Liveness и Readiness probes
- освоить диагностику состояния приложений

LivenessProbe

- назначение: определение необходимости перезапуска контейнера
- типы: exec, httpGet, TCP
- параметры: initialDelaySeconds, periodSeconds, timeoutSeconds

ReadinessProbe

- назначение: определение готовности принимать трафик
- отличие от Liveness: не перезапускает, а исключает из балансировки

Volume

- назначение: хранение данных между перезапусками контейнеров
- типы: emptyDir, hostPath, configMap, secret

Все эти элементы помогают управлять состоянием приложений и их жизненным циклом в контейнеризированной среде

a) LivenessProbe (Проверки живучести) предназначены для определения, находится ли контейнер в состоянии, когда его следует перезапустить. Если проверка живучести не проходит, Kubernetes перезапускает контейнер. Это полезно, например, в случаях, когда приложение зависло или перестало отвечать. Есть 3 механизма проверок: exec, http get, TCP.

Параметры

initialDelaySeconds: 5 # кол-во сек от старта контейнера до запуска LivenessProbe
periodSeconds: 5 # кол-во секунд между пробями
timeoutSeconds: 1 # кол-во секунд ожидания пробы
successThreshold: 1 # мин кол-во проверок, чтобы проба считалась неудачной

failureThreshold: 3 # сколько раз сделает команду, чтобы считать контейнер умершим и перезапустить

Пример 1. Exec Liveness Probe

Проверка здоровья через выполнение команды в контейнере. Команда `cat /tmp/healthy` проверяет существование файла. В результате контейнер перезапускается при удалении файла → `CrashLoopBackOff`

01-liveness-exec.yaml – Блокнот

Файл Правка Формат Вид Справка

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: liveness-exec
spec:
  selector:
    matchLabels:
      app: liveness-exec
  template:
    metadata:
      labels:
        app: liveness-exec
    spec:
      containers:
        - name: app
          image: alpine
          command: ["/bin/sh"]
          args: ["-c", "touch /tmp/healthy; sleep 30; rm -f /tmp/healthy; sleep 600"]
          livenessProbe:
            exec:
              command:
                - cat
                - /tmp/healthy
              initialDelaySeconds: 5
              periodSeconds: 5
```

```
C:\Temp>kubectl apply -f 01-liveness-exec.yaml
deployment.apps/liveness-exec created
```

```
C:\Temp>kubectl get pods --watch
```

NAME	READY	STATUS	RESTARTS	AGE
liveness-exec-5dd6cf654d-pnhm4	0/1	ContainerCreating	0	8s
liveness-exec-5dd6cf654d-pnhm4	0/1	ErrImagePull	0	16s
liveness-exec-5dd6cf654d-pnhm4	0/1	ImagePullBackOff	0	31s
liveness-exec-5dd6cf654d-pnhm4	1/1	Running	0	43s
liveness-exec-5dd6cf654d-pnhm4	1/1	Running	1 (26s ago)	2m21s

При вызове `kubectl describe pod`

```

    touch /tmp/healthy; sleep 30; rm -f /tmp/healthy; sleep 600
    State:          Running
    Started:        Thu, 16 Oct 2025 23:04:19 +0300
    Last State:     Terminated
    Reason:         Error
    Exit Code:      137
    Started:        Thu, 06 Oct 2024 23:02:15 +0300
    Finished:       Thu, 06 Oct 2024 23:03:25 +0300
    Ready:          True
    Restart Count:  4
    Liveness:       exec [cat /tmp/healthy] delay=5s timeout=1s period=5s #success=1 #failure=3
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-5mzw4 (ro)
Conditions:

```

```

Events:
  Type     Reason      Age      From          Message
  ----     -
  Normal   Scheduled   4m4s     default-scheduler   Successfully assigned default/liveness-exec-5dd6cf654d-pnh
  Warning  Failed      3m49s    kubelet        Failed to pull image "alpine": Error response from daemon:
  failed to resolve reference "docker.io/library/alpine:latest": failed to authorize: failed to fetch anonymous token: Get
  "https://auth.docker.io/token?scope=repository%3Alibrary%2Falpine%3Apull&service=registry.docker.io": dialing auth.docker
  .io:443 container via direct connection because static system has no HTTPS proxy: connecting to auth.docker.io:443: d
  ial tcp: lookup auth.docker.io: no such host
  Warning  Failed      3m49s    kubelet        Error: ErrImagePull
  Warning  Failed      3m48s    kubelet        Error: ImagePullBackOff
  Normal   BackOff     3m48s    kubelet        Back-off pulling image "alpine"
  Normal   Pulled      3m22s    kubelet        Successfully pulled image "alpine" in 10.732s (10.732s inclu
  ding waiting). Image size: 3813273 bytes.
  Normal   Pulled      103s     kubelet        Successfully pulled image "alpine" in 25.41s (25.41s inclu
  ding waiting). Image size: 3813273 bytes.
  Warning  Unhealthy   59s (x6 over 2m49s)  kubelet        Liveness probe failed: cat: can't open '/tmp/healthy': No
  such file or directory
  Normal   Killing     59s (x2 over 2m39s)  kubelet        Container app failed liveness probe, will be restarted
  Normal   Pulling     29s (x4 over 4m4s)  kubelet        Pulling image "alpine"
  Normal   Pulled      22s     kubelet        Successfully pulled image "alpine" in 7.194s (7.194s inclu
  ding waiting). Image size: 3813273 bytes.
  Normal   Created     21s (x3 over 3m22s)  kubelet        Created container: app
  Normal   Started     21s (x3 over 3m22s)  kubelet        Started container app

```

Что произошло:

1. Конфигурация Liveness Probe:

```
Liveness: exec [cat /tmp/healthy] delay=5s timeout=1s period=5s
#success=1 #failure=3
```

Показывает параметры проверки

2. События провала пробы:

```
Warning Unhealthy 59s (x3 over 69s) Liveness probe failed: cat:
can't open '/tmp/healthy': No such file or directory
```

Проба не нашла файл /tmp/healthy

3. Реакция Kubernetes:

```
Normal Killing 59s Container app failed liveness probe, will be
restarted
```

Kubernetes убивает контейнер из-за провала пробы

4. Состояние контейнера:

```
Last State: Terminated
Reason:      Error
```

Exit Code: 137

LivenessProbe работает корректно:

- через 5 секунд начал проверять файл
- после 3 неудачных проверок (15 секунд) решил перезапустить контейнер
- Kubernetes выполнил перезапуск при провале health check
- итоговое состояние Pod — CrashLoopBackOff, что является ожидаемым поведением для постоянно падающего контейнер

4.b) TCP проверка подключения. Если подключение успешно установлено - то все хорошо. Здесь специально в livenessProbe идет ошибочный порт 6380 (а не 6379), чтобы показать, что контейнер перезагрузится.

Пример 2. TCP Liveness Probe

Проверка здоровья через TCP подключение к неправильному порту. Redis слушает порт 6379. LivenessProbe проверяет порт 6380. В результате: connection refused → постоянные перезапуски → CrashLoopBackOff

```
03-liveness-tcp.yaml – Блокнот
Файл  Правка  Формат  Вид  Справка
apiVersion: apps/v1
kind: Deployment
metadata:
  name: liveness-tcp
spec:
  selector:
    matchLabels:
      app: liveness-tcp
  template:
    metadata:
      labels:
        app: liveness-tcp
    spec:
      containers:
      - name: app
        image: redis:7-alpine
        ports:
        - containerPort: 6379
        livenessProbe:
          tcpSocket:
            port: 6380 # ← Неправильный порт!
          initialDelaySeconds: 15
          periodSeconds: 20
```

```
C:\Temp>kubectl apply -f 03-liveness-tcp.yaml
deployment.apps/liveness-tcp configured

C:\Temp>kubectl get pods
NAME                                READY   STATUS             RESTARTS   AGE
liveness-exec-5dd6cf654d-pnhm4      0/1     CrashLoopBackOff   11 (4m42s ago)  38m
liveness-http-5dc5b9bd9b-p44j9      0/1     CrashLoopBackOff   9 (73s ago)     16m
liveness-tcp-6946484545-mffhg       1/1     Running            0             33s

C:\Temp>kubectl get pods
NAME                                READY   STATUS             RESTARTS   AGE
liveness-exec-5dd6cf654d-pnhm4      0/1     CrashLoopBackOff   11 (4m50s ago)  38m
liveness-http-5dc5b9bd9b-p44j9      0/1     CrashLoopBackOff   9 (81s ago)     17m
liveness-tcp-6946484545-mffhg       1/1     Running            0             41s

C:\Temp>kubectl get pods --watch
NAME                                READY   STATUS             RESTARTS   AGE
liveness-exec-5dd6cf654d-pnhm4      1/1     Running            12 (5m28s ago)  39m
liveness-http-5dc5b9bd9b-p44j9      0/1     CrashLoopBackOff   9 (119s ago)    17m
liveness-tcp-6946484545-mffhg       1/1     Running            1 (19s ago)     79s
```

События провала пробы

```
Events:
  Type    Reason      Age    From          Message
  ----    -
  Normal  Scheduled   2m4s   default-scheduler  Successfully assigned default/liveness-tcp-6946484545-mffhg to docker-desktop
  Normal  Pulled      4s (x3 over 2m4s)  kubelet            Container image "redis:7-alpine" already present on machine
  Normal  Created     4s (x3 over 2m3s)  kubelet            Created container app
  Warning Unhealthy   4s (x6 over 104s)  kubelet            Liveness probe failed: dial tcp 10.1.0.91:6380: connect: connection refused
  Normal  Killing     4s (x2 over 64s)  kubelet            Container app failed liveness probe, will be restarted
  Normal  Started     3s (x3 over 2m3s)  kubelet            Started container app
```

Финальный статус (через 5+ минут)

```
C:\Temp>kubectl get pods -l app=liveness-tcp
NAME                                READY   STATUS             RESTARTS   AGE
liveness-tcp-6946484545-mffhg       0/1     CrashLoopBackOff   6 (33s ago)  8m53s
```

Что произошло:

- начальное состояние - под запущен
- события провала - "connection refused"
- конфигурация пробы - неправильный порт 6380
- реальные порты - Redis на 6379
- результат - CrashLoopBackOff

Неправильная конфигурация → провал проб → перезапуски → CrashLoopBackOff.

4.c) HttpGet делает запрос на порт. Если сервис не отвечает или передает код с ошибкой - контейнер автоматически перезапустится. Делается GET запрос на /healthz. Если статус 200-399 - приложение здорово

Пример 3. HTTP Liveness Probe

Проверка здоровья через HTTP запрос. Запрос к несуществующему эндпоинту /healthz. В результате: 404 ошибка → перезапуск контейнера

```
02-liveness-http.yaml – Блокнот
Файл Правка Формат Вид Справка
apiVersion: apps/v1
kind: Deployment
metadata:
  name: liveness-http
spec:
  selector:
    matchLabels:
      app: liveness-http
  template:
    metadata:
      labels:
        app: liveness-http
    spec:
      containers:
      - name: app
        image: nginx:1.25
        ports:
        - containerPort: 80
        livenessProbe:
          httpGet:
            path: /healthz
            port: 80
            httpHeaders:
            - name: Custom-Header
              value: HealthCheck
          initialDelaySeconds: 10
          periodSeconds: 10
          timeoutSeconds: 5
```

```
C:\Temp>kubectl apply -f 02-liveness-http.yaml
deployment.apps/liveness-http created
```

```
C:\Temp>kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
liveness-exec-5dd6cf654d-pnhm4	0/1	CrashLoopBackOff	7 (4m35s ago)	22m
liveness-http-5dc5b9bd9b-p44j9	1/1	Running	1 (11s ago)	51s

liveness-exec - постоянно перезапускается (CrashLoopBackOff)

liveness-http - уже был 1 перезапуск и сейчас работает

```
Events:
  Type     Reason      Age          From          Message
  ----     -
Normal    Scheduled   2m43s       default-scheduler Successfully assigned default/liveness-http-5dc5b9bd9b-p44j9 to docker-desktop
Warning   Unhealthy   3s (x12 over 2m23s) kubelet        Liveness probe failed: HTTP probe failed with statuscode: 404
Normal    Killing     3s (x4 over 2m3s) kubelet        Container app failed liveness probe, will be restarted
Normal    Pulled      2s (x5 over 2m42s) kubelet        Container image "nginx:1.25" already present on machine
Normal    Created     2s (x5 over 2m42s) kubelet        Created container: app
Normal    Started     2s (x5 over 2m42s) kubelet        Started container app
```

Что произошло с liveness-http:

- Nginx запустился, но эндпоинт /healthz не существует
- LivenessProbe получал 404 ошибку
- после нескольких фейлов контейнер перезапустился
- статус Running может быть обманчив. Pod находится в процессе перезапуска или между пробам. Liveness Probe продолжает опрашивать несуществующий эндпоинт, и после очередной серии неудач (failureThreshold) контейнер будет перезапущен снова.

d) Readiness - используются для определения, готово ли приложение принимать трафик. Если проверка готовности не проходит, Kubernetes не отправляет на этот контейнер запросы от сервиса, то есть идентично прошлому, только liveness отвечает за живучесть контейнера (просто его перезапускает), то Readiness понимает, когда регистрировать порт к сервису, чтобы начать принимать трафик.

Пример 4. Комбинированные Probes

Одновременное использование Liveness и Readiness проб: Liveness: /health (фейлится), Readiness: / (работает). В результате: Под "живой" но "не готов" → нет трафика + перезапуски



04-liveness-combined.yaml – Блокнот

Файл Правка Формат Вид Справка

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: liveness-combined
spec:
  selector:
    matchLabels:
      app: liveness-combined
  template:
    metadata:
      labels:
        app: liveness-combined
    spec:
      containers:
        - name: app
          image: nginx:1.25
          ports:
            - containerPort: 80
          livenessProbe:
            httpGet:
              path: /health
              port: 80 # ← Исправил порт
            initialDelaySeconds: 30
            periodSeconds: 10
          readinessProbe:
            httpGet:
              path: /
              port: 80 |
            initialDelaySeconds: 5
            periodSeconds: 5
```

При вызове `kubectl describe pod`

```
Events:
  Type       Reason      Age          From          Message
  ----       -
  Normal    Scheduled   6m4s        default-scheduler   Successfully assigned default/liveness-combined-565f7cd74b-tmf6r to docker-desktop
  Normal    Pulled      64s (x6 over 6m4s)   kubelet            Container image "nginx:1.25" already present on machine
  Normal    Created     64s (x8 over 6m4s)   kubelet            Created container: app
  Normal    Started     64s (x6 over 6m4s)   kubelet            Started container app
  Warning   Unhealthy   5s (x18 over 5m25s)  kubelet            Liveness probe failed: HTTP probe failed with statuscode: 404
  Normal    Killing     5s (x6 over 5m5s)    kubelet            Container app failed liveness probe, will be restarted
  Warning   BackOff     2s (x3 over 4s)     kubelet            Back-off restarting failed container app in pod liveness-combined-565f7cd74b-tmf6r
3dfb-b018-435b-87bc-55fdb3976c9)

C:\Temp>kubectl get endpoints -l app=liveness-combined
No resources found in default namespace.

C:\Temp>kubectl get pods -l app=liveness-combined -w
NAME                                READY    STATUS      RESTARTS   AGE
liveness-combined-565f7cd74b-tmf6r  0/1     CrashLoopBackOff  5 (48s ago)  6m49s
liveness-combined-565f7cd74b-tmf6r  0/1     Running         6 (92s ago)  7m33s
liveness-combined-565f7cd74b-tmf6r  1/1     Running         6 (99s ago)  7m40s
liveness-combined-565f7cd74b-tmf6r  0/1     CrashLoopBackOff  6 (1s ago)   8m32s
```

Warning Unhealthy 5s (x18 over 5m25s) Liveness probe failed: HTTP probe failed with statuscode: 404
 Normal Killing 5s (x6 over 5m5s) Container app failed liveness probe, will be restarted

Показывает: Liveness probe постоянно фейлится → контейнер перезапускается

No resources found in default namespace.

Показывает: Readiness probe не проходит → под НЕ добавляется в endpoints → НЕ получает трафик

NAME	READY	STATUS	RESTARTS	AGE
liveness-combined-565f7cd74b-tmf6r	0/1	CrashLoopBackOff	5 (48s ago)	6m49s
liveness-combined-565f7cd74b-tmf6r	0/1	Running	6 (92s ago)	7m33s
liveness-combined-565f7cd74b-tmf6r	1/1	Running	6 (99s ago)	7m40s

Показывает:

- READY: 0/1 ← Readiness probe не прошла
- CrashLoopBackOff + RESTARTS: 6 ← Liveness probe вызывает перезапуски
- Моментальное изменение 0/1 → 1/1 → 0/1 ← борьба проб

Liveness Probe работает:

- Видит 404 на /health
- После 3 фейлов убивает контейнер
- Вызывает перезапуски (6 раз на скриншоте)
- Приводит к CrashLoopBackOff
-

Readiness Probe работает:

- Не пропускает под в endpoints
- Делает под недоступным для трафика
- Показывает READY: 0/1

Вывод: Обе пробы работают корректно, но с разными последствиями:

- **Liveness** → перезапускает контейнер
- **Readiness** → блокирует трафик

е) Volume: предоставляют способ хранения данных, который переживает перезапуск контейнеров и обеспечивает совместное использование данных между контейнерами.

Volume в Kubernetes — это абстракция для хранения данных, которая позволяет сохранять информацию независимо от жизненного цикла контейнера. В отличие от файловой системы контейнера, данные в Volume сохраняются при перезапуске Pod. Данные в `emptyDir` сохраняются только на время жизни Pod. При удалении Pod все данные в `emptyDir` будут безвозвратно утеряны. Этот тип тома не подходит для постоянных данных

Пример 5. Volume `emptyDir`

Общее хранилище данных между контейнерами. Два контейнера работают с одним файлом. В результате: обмен данными + сохранение при перезапусках контейнеров.



06-volume-emptydir.yaml – Блокнот

—



Файл Правка Формат Вид Справка

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: volume-demo
spec:
  selector:
    matchLabels:
      app: volume-demo
  template:
    metadata:
      labels:
        app: volume-demo
    spec:
      containers:
        - name: writer
          image: alpine
          command: ["/bin/sh"]
          args: ["-c", "while true; do echo $(date) | tee -a /data/log.txt; sleep 5; done"]
          volumeMounts:
            - name: shared-data
              mountPath: /data

        - name: reader
          image: alpine
          command: ["/bin/sh"]
          args: ["-c", "tail -f /data/log.txt"]
          volumeMounts:
            - name: shared-data
              mountPath: /data

      volumes:
        - name: shared-data
          emptyDir: {}
```

```
C:\Temp>kubectl apply -f 06-volume-emptydir.yaml
deployment.apps/volume-demo created

C:\Temp>kubectl get pods -l app=volume-demo
NAME                                READY   STATUS    RESTARTS   AGE
volume-demo-756bf6985c-z5pt2       2/2     Running   0           28s

C:\Temp>kubectl logs -l app=volume-demo -c writer
Thu Oct 16 21:52:10 UTC 2025
Thu Oct 16 21:52:15 UTC 2025
Thu Oct 16 21:52:20 UTC 2025
Thu Oct 16 21:52:25 UTC 2025
Thu Oct 16 21:52:30 UTC 2025
Thu Oct 16 21:52:35 UTC 2025

C:\Temp>kubectl logs -l app=volume-demo -c reader
Thu Oct 16 21:52:10 UTC 2025
Thu Oct 16 21:52:15 UTC 2025
Thu Oct 16 21:52:20 UTC 2025
Thu Oct 16 21:52:25 UTC 2025
Thu Oct 16 21:52:30 UTC 2025
Thu Oct 16 21:52:35 UTC 2025
Thu Oct 16 21:52:40 UTC 2025
```

Volume emptyDir успешно работает:

- Writer контейнер пишет данные в /data/log.txt каждые 5 секунд
- Reader контейнер читает те же данные в реальном времени
- оба контейнера имеют доступ к общему файлу
- данные идентичны в обоих контейнерах

```
C:\Temp>kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
volume-demo-756bf6985c-z5pt2       2/2     Running   0           12m

C:\Temp>kubectl exec pod/volume-demo-756bf6985c-z5pt2 -c reader -- ls -i /data/log.txt
7931 /data/log.txt

C:\Temp>kubectl exec pod/volume-demo-756bf6985c-z5pt2 -c writer -- ls -i /data/log.txt
7931 /data/log.txt
```

Задания для самостоятельного решения

Задание 6.1. Базовые LivenessProbes

Создайте Deployment с разными типами LivenessProbe:

- exec: проверка существования файла
- httpGet: проверка HTTP эндпоинта (убедитесь, что Pod с httpGet пробой переходит в состояние CrashLoopBackOff)
- TCP: проверка доступности порта

Проверочные команды

1. Общий мониторинг состояния подов
2. Детальная информация о пробах
3. Проверка событий провала проб
4. Логи контейнера для анализа причин падения
5. Проверка истории перезапусков

Задание 6.2. ReadinessProbe

Создайте приложение, которое:

- имеет длительную инициализацию (30+ секунд)
- использует ReadinessProbe для исключения из балансировки во время старта
- демонстрирует разницу между Liveness и Readiness
- используйте sleep 40 в команде инициализации приложения. С помощью kubectl get pods -w наблюдайте, как статус меняется с 0/1 на 1/1 только после окончания инициализации

Проверочные команды

1. Наблюдение за изменением статуса Ready
2. Проверка endpoints сервиса до и после готовности
3. Сравнение Liveness и Readiness конфигурации
4. Логи инициализации приложения в реальном времени
5. Время перехода в Ready состояние
6. Анализ временной шкалы событий запуска

Задание 6.3. Volume emptyDir

Создайте многоконтейнерный Pod с:

- контейнером-генератором данных
- контейнером-читателем данных
- общим emptydir volume для обмена данными

Проверочные команды

1. Проверка состояния многоконтейнерного пода
2. Проверка монтирования volume в контейнерах
3. Сравнение inode файлов в разных контейнерах
4. Логи генератора данных

5. Логи читателя данных

Задание 6.4. Диагностика probes

Создайте проблемные сценарии и научитесь их диагностировать:

- неправильные параметры таймаутов
- некорректные пути/порты
- анализ событий и логов
- создайте сценарий, где Probe имеет `initialDelaySeconds: 2`, но приложение стартует 10 секунд. Объясните, что произойдет

Проверочные команды

1. Комплексная диагностика проблемного пода
2. Детальный анализ событий
3. Логи предыдущего контейнера после перезапуска
4. Проверка доступности портов внутри контейнера
5. Тестирование endpoints изнутри контейнера

Контрольные вопросы для всех заданий

В чем основное отличие `LivenessProbe` от `ReadinessProbe`?

Какие типы проверок поддерживают probes и когда каждый из них следует использовать?

Что произойдет, если не настроить `ReadinessProbe` для приложения с долгим стартом?

Какой смысл параметров `initialDelaySeconds`, `periodSeconds` и `timeoutSeconds`?

Что означает `failureThreshold` и `successThreshold`?

Что такое `emptyDir` и для каких сценариев он предназначен?

Какие три команды вы бы использовали для диагностики проблем с `LivenessProbe`?

Что делать, если контейнер постоянно перезапускается (`CrashLoopBackOff`)?

Как узнать, сколько раз контейнер перезапускался из-за провала `LivenessProbe`?

Какие события в `kubectl describe pod` указывают на проблемы с probes?

Как проверить, что `ReadinessProbe` корректно исключает под из балансировки?

Как доказать, что два контейнера используют один физический файл?

Какой тип Probe (Liveness/Readiness) должен сработать, если ваше приложение полностью функционирует, но временно не может обрабатывать новые запросы из-за перегрузки базы данных?

Что произойдет с данными в emptyDir, если узел (node), на котором работает Pod, перезагрузится?