6. Pod Probes

In Kubernetes, you can define liveness and readiness probes to check the health of your app.

Probes

--------------------------------------------script starts here----------------------------------------------------------

apiVersion: apps/v1

kind: Deployment

metadata:

name: myapp

spec:

replicas: 1

selector:

matchLabels:

app: myapp

template:

metadata:

labels:

app: myapp

spec:

containers:

- name: myapp-container

image: myapp\_image:latest

env:

- name: DB\_HOST

value: "localhost" # If DB\_HOST needs to be dynamic, consider storing it in a secret or ConfigMap

- name: DB\_PORT

value: "5432" # Same as above for DB\_PORT

- name: DB\_USER

valueFrom:

secretKeyRef:

name: db-credentials # Referencing the db-credentials secret

key: username # Fetching the DB username from the secret

- name: DB\_PASSWORD

valueFrom:

secretKeyRef:

name: db-credentials # Referencing the db-credentials secret

key: password # Fetching the DB password from the secret

- name: DB\_NAME

value: "myapp\_db" # You can leave this in the deployment if it doesn't change dynamically

ports:

- containerPort: 8080

livenessProbe:

httpGet:

path: /health

port: 8080

initialDelaySeconds: 5 # Wait 5 seconds before starting liveness probe

periodSeconds: 5 # Run the probe every 5 seconds

timeoutSeconds: 3 # Timeout after 3 seconds if the probe doesn't return in time

failureThreshold: 3 # After 3 failures, the container will be restarted

successThreshold: 1 # After 1 successful check, the container is considered healthy

readinessProbe:

httpGet:

path: /db-health

port: 8080

initialDelaySeconds: 10 # Wait 10 seconds before starting readiness probe

periodSeconds: 5 # Run the probe every 5 seconds

timeoutSeconds: 3 # Timeout after 3 seconds

failureThreshold: 3 # After 3 failures, the pod is considered not ready

successThreshold: 1 # After 1 successful check, the pod is considered ready

---

apiVersion: v1

kind: Service

metadata:

name: myapp-service

spec:

ports:

- port: 8080

selector:

app: myapp

--------------------------------------------script ends here----------------------------------------------------------

**Key Updates and Explanations:**

1. **Liveness Probe**:
   * The liveness probe is configured to check the /health endpoint on port 8080. This endpoint checks the basic health of the web application, and it doesn’t specifically verify the database connection.
   * initialDelaySeconds: This gives the container 5 seconds after it starts before the liveness probe begins.
   * periodSeconds: The probe will run every 5 seconds to ensure the container is alive.
   * timeoutSeconds: If the probe takes longer than 3 seconds, it will be considered a failure.
   * failureThreshold: The container will be restarted after 3 consecutive failures of the liveness probe.
   * successThreshold: After 1 successful liveness check, the container is considered alive again.
2. **Readiness Probe**:
   * The readiness probe is configured to check the /db-health endpoint on port 8080. This endpoint specifically checks the database connection and ensures the web application can connect to the PostgreSQL database.
   * initialDelaySeconds: This delay gives the application time to establish the database connection. The readiness probe starts after 10 seconds.
   * periodSeconds: The probe runs every 5 seconds to verify the database connectivity.
   * timeoutSeconds: If the probe takes longer than 3 seconds, it will be considered a failure.
   * failureThreshold: The pod will be considered not ready after 3 consecutive failures.
   * successThreshold: After 1 successful readiness check, the pod is considered ready.

**How Environment Variables Work with the Probes:**

* **Liveness Probe**: Since /health just verifies that the app is running, it doesn't depend on the database credentials. So, there is no need to alter the probe based on database credentials.
* **Readiness Probe**: This probe calls /db-health to check if the application can successfully connect to the database using the credentials provided by Kubernetes secrets. The DB\_USER, DB\_PASSWORD, DB\_HOST, DB\_PORT, and DB\_NAME environment variables will be automatically populated by Kubernetes using the secretKeyRef for DB\_USER and DB\_PASSWORD, and the hardcoded values for DB\_HOST, DB\_PORT, and DB\_NAME.

**Conclusion:**

The pod probes (livenessProbe and readinessProbe) are already properly configured to check the application's health and database connectivity using the endpoints /health and /db-health. Since the environment variables (including database credentials) are already injected into the container via Kubernetes secrets, no further changes are necessary for the probes themselves.