**1. Health Checks in Kubernetes**

Kubernetes provides two types of probes to check the health of containers:

1. **Liveness Probe** - Determines if the application inside a container is alive. If it fails, Kubernetes will restart the container.
2. **Readiness Probe** - Checks if the application is ready to accept traffic. If it fails, Kubernetes will remove the pod from service discovery until it becomes ready.
3. **Startup Probe** - Used for slow-starting applications. It ensures the container has started successfully before running other probes.

**Example 1: HTTP Liveness and Readiness Probe**

apiVersion: v1

kind: Pod

metadata:

name: health-check-pod

spec:

containers:

- name: my-app

image: nginx

ports:

- containerPort: 80

livenessProbe:

httpGet:

path: /

port: 80

initialDelaySeconds: 5

periodSeconds: 10

readinessProbe:

httpGet:

path: /

port: 80

initialDelaySeconds: 3

periodSeconds: 5

**Explanation:**

* **Liveness Probe**: Checks if the container is alive by sending an HTTP GET request to / on port 80 every 10 seconds. If it fails, Kubernetes restarts the container.
* **Readiness Probe**: Determines if the container is ready to serve requests. If the check fails, the pod is removed from the service endpoint.

**Commands to Apply and Check**

kubectl apply -f health-check.yaml

kubectl describe pod health-check-pod

kubectl get pod health-check-pod -o jsonpath='{.status.conditions}'

**Example 2: TCP Socket Liveness Probe**

apiVersion: v1

kind: Pod

metadata:

name: tcp-health-check

spec:

containers:

- name: my-app

image: nginx

ports:

- containerPort: 80

livenessProbe:

tcpSocket:

port: 80

initialDelaySeconds: 5

periodSeconds: 10

**Explanation:**

* Instead of HTTP, this probe checks if a TCP connection to port 80 is successful.

**Commands**

kubectl apply -f tcp-health-check.yaml

kubectl get pods

kubectl describe pod tcp-health-check

**Example 3: Command-based Liveness Probe**

apiVersion: v1

kind: Pod

metadata:

name: cmd-health-check

spec:

containers:

- name: my-app

image: busybox

args:

- /bin/sh

- -c

- touch /tmp/healthy; sleep 30; rm -f /tmp/healthy; sleep 600

livenessProbe:

exec:

command:

- cat

- /tmp/healthy

initialDelaySeconds: 5

periodSeconds: 10

**Explanation:**

* This probe runs cat /tmp/healthy inside the container. If the file is missing, the probe fails and Kubernetes restarts the pod.

**Commands**

kubectl apply -f cmd-health-check.yaml

kubectl describe pod cmd-health-check

**2. Monitoring in Kubernetes**

Kubernetes provides multiple ways to monitor workloads:

1. **Using kubectl commands**
2. **Using Metrics Server**
3. **Using Prometheus & Grafana**
4. **Using Logging Solutions (Fluentd, Elasticsearch, Kibana)**

**Example 4: Monitoring Resource Usage with kubectl top**

Install the **Metrics Server** if not already installed:

kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml

Check node and pod resource usage:

kubectl top nodes

kubectl top pods --all-namespaces

**Example 5: Prometheus and Grafana for Advanced Monitoring**

**Step 1: Deploy Prometheus**

kubectl create namespace monitoring

kubectl apply -f https://github.com/prometheus-operator/prometheus-operator/releases/latest/download/bundle.yaml

**Step 2: Deploy Grafana**

kubectl apply -f https://raw.githubusercontent.com/grafana/grafana/main/deploy/kubernetes/grafana-deployment.yaml

**Step 3: Access Grafana**

kubectl port-forward svc/grafana 3000:80 -n monitoring

Open http://localhost:3000 and configure Prometheus as a data source.

**Example 6: Logging with Fluentd and Elasticsearch**

Deploy Fluentd DaemonSet:

kubectl apply -f https://raw.githubusercontent.com/fluent/fluentd-kubernetes-daemonset/master/fluentd-daemonset-elasticsearch-rbac.yaml

To view logs:

kubectl logs -f <pod-name>

**Summary of Key kubectl Commands**

| **Command** | **Description** |
| --- | --- |
| kubectl get pods | List all pods |
| kubectl describe pod <pod-name> | Detailed information about a pod |
| kubectl logs <pod-name> | Fetch logs from a pod |
| kubectl top nodes | Monitor node CPU/memory usage |
| kubectl top pods | Monitor pod CPU/memory usage |
| kubectl port-forward svc/grafana 3000:80 -n monitoring | Access Grafana dashboard |