Additional Kubernetes Features

In this lesson, we'll discuss some additional Kubernetes features.

We'll cover the following

- Monitoring with liveness and readiness probes
- Configuration
- Separating Kubernetes environments with namespaces
- Applications with state
  - Persistent volumes & stateful sets
  - Operators
- Extensions with Helm

Kubernetes is a powerful technology with many features. Here are some examples of additional Kubernetes features:

Monitoring with liveness and readiness probes

Kubernetes recognizes the failure of a pod via **Liveness Probes** (https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-probes/). A custom Liveness Probe can be used to determine **when a container is started anew** depending on the needs of the application.

A Readiness Probe (https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-probes/#define-readiness-probes)

container/configure-liveness-readiness-probes/#define-readiness-probes), however, **indicates whether a container can process requests or not.**

- For example, if the application is blocked by processing a large amount of data or has not yet started completely, the Readiness Probe can report this state to Kubernetes.
- In contrast to a Liveness Probe, the container is not restarted as a result of a failed Readiness Probe.
- Kubernetes assumes that after some time the pod will signal via the Readiness Probe that it can handle requests.

Configuration

The configuration of applications is possible with ConfigMaps (https://kubernetes.io/docs/tasks/configure-pod-container/configure-pod-configmap/). The configuration data is provided to the applications as values in environment variables.

Separating Kubernetes environments with namespaces

Kubernetes environments can be separated with Namespaces (https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/).

> **Namespaces** are virtual clusters so that services and deployments are completely separated.

Separation with namespaces allows **different environments to coexist**; it is possible for multiple teams to share a cluster and use namespaces to separate their environments.

Separation with namespaces also makes it possible to **separate the microservices from infrastructure** like databases or monitoring infrastructure. That way users only see the services and deployments they are interested in.

Applications with state#

Persistent volumes & stateful sets#

Kubernetes **can also handle applications that have state**. Applications without state can simply be restarted on another node in a cluster. This facilitates fail-safety and load balancing.

If the application has state and therefore requires certain data in a Docker volume, the required Docker volumes must be available on each node the application runs on. This makes the **handling of such applications more complex**.

Kubernetes offers persistent volumes (https://kubernetes.io/docs/concepts/storage/persistent-volumes/) and stateful sets (https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/#stable-storage) for dealing with this challenge.

Operators#

Another option for dealing with applications that have state are operators

(https://coreos.com/operators). They allow the automated installation of applications with state.

For example, there is the Prometheus operator (https://github.com/coreos/prometheus-operator) which installs the monitoring system Prometheus in a Kubernetes cluster. It introduces Kubernetes resources for Prometheus components such as `Prometheus`, `ServiceMonitor`, and `AlterManager`.

With the Prometheus operator, these keywords are used by the Kubernetes configuration instead of pods, services, or deployments. The operator also determines how Prometheus saves the monitoring data, and thus solves a key challenge.

Extensions with Helm

Kubernetes offers a complex ecosystem with numerous extensions. Helm (https://helm.sh/) offers the possibility to install extensions as Charts (https://github.com/kubernetes/charts/) and thereby assumes the functionality of a package manager for Kubernetes. This extensibility is an important advantage of Kubernetes.

Using a Helm chart, just the name of the microservice has to be defined. The Kubernetes configuration is generated with a template and the provided name. This makes the installation of the microservices much easier and more uniform.

Q U I

Z

1

What is the difference between a readiness probe and a liveliness probe?

A) A readiness probe indicates if a pod has failed whereas a liveliness probe indicates whether a pod is ready to process data or not.

B) A readiness probe indicates if a pod has failed and a liveliness probe indicates whether a pod is ready to process data or not. Also, the container is restarted in the case of a failed liveliness probe.

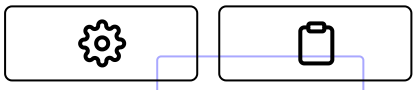
C) A liveliness probe indicates if a pod has failed and a readiness probe indicates whether a pod is ready to process data or not. Also, the container is restarted in the case of a failed liveliness probe.

Submit Answer

Question 1 of 3
0 attempted

Reset Quiz

In the next lesson, we'll look at some variations that can be conducted.



← Back

Next →

The Example in Detail

Variations

Mark as Completed

Report an Issue