FLOWX.AI

PLATFORM OVERVIEW / Frameworks and standards / Event-driven architecture frameworks / intro-to-kubernetes

Contents

PLATFORM OVERVIEW / Frameworks and standards / Event-driven architecture frameworks / Intro to Kubernetes

What is Kubernetes?

Kubernetes is an open-source container orchestration platform that automates many of the manual processes involved in containerized application deployment, management, and scaling.

The purpose of Kubernetes is to orchestrate containerized applications to run on a cluster of hosts. **Containerization** enables you to deploy multiple applications using the same operating system on a single virtual machine or server.

Kubernetes, as an open platform, enables you to build applications using your preferred programming language, operating system, libraries, or messaging bus. To schedule and deploy releases, existing continuous integration and continuous delivery (CI/CD) tools can be integrated with Kubernetes.

Benefits of using Kubernetes

- A proper way of managing containers

- High availability
- Scalability
- Disaster recovery

Key Kubernetes Concepts

Node & PODs

A Kubernetes node is a machine that runs containerized workloads as part of a Kubernetes cluster. A node can be a physical machine or a virtual machine, and can be hosted on-premises or in the cloud.

A pod is composed of one or more containers that are colocated on the same host and share a network stack as well as other resources such as volumes. Pods are the foundation upon which Kubernetes applications are built.

Kubernetes uses pods to run an instance of your application. A pod represents a single instance of your application.

Pods are typically ephemeral, disposable resources. Individually scheduled pods miss some of the high availability and redundancy Kubernetes features. Instead, pods are deployed and managed by Kubernetes *Controllers*, such as the Deployment Controller.

Service & Ingress

Service is an abstraction that defines a logical set of pods and a policy for accessing them. In Kubernetes, a Service is a REST object, similar to a pod. A Service definition, like all REST objects, can be POSTed to the API server to

create a new instance. A Service object's name must be a valid RFC 1035 label name.

Ingress is a Kubernetes object that allows access to the Kubernetes services from outside of the Kubernetes cluster. You configure access by writing a set of rules that specify which inbound connections are allowed to reach which services. This allows combining all routing rules into a single resource.

Ingress controllers are pods, just like any other application, so they're part of the cluster and can see and communicate with other pods. An Ingress can be configured to provide Services with externally accessible URLs, load balance traffic, terminate SSL / TLS, and provide name-based virtual hosting. An Ingress controller is in charge of fulfilling the Ingress, typically with a load balancer, but it may also configure your edge router or additional frontends to assist with the traffic.

FlowX.AI offers a predefined NGINX setup as Ingress Controller. The NGINX Ingress Controller works with the NGINX web server (as a proxy). For more information, check the below sections:

» Intro to NGINX

» Designer setup guide

ConfigMap & Secret

ConfigMap is an API object that makes it possible to store configuration for use by other objects. A ConfigMap, unlike most Kubernetes objects with a spec, has `data` and `binaryData` fields. As values, these fields accept key-value pairs. The `data` field and `binaryData` are both optional. The data field is intended to hold UTF-8 strings, whereas the `binaryData` field is intended to hold binary data as base64-encoded strings.

> **INFO**
>
> The name of a ConfigMap must be a valid DNS subdomain name.

Secret represents an amount of sensitive data, such as a password, token, or key. Alternatively, such information could be included in a pod specification or a container image. Secrets are similar to ConfigMaps but they are designed to keep confidential data.

Volumes

A Kubernetes volume is a directory in the orchestration and scheduling platform that contains data accessible to containers in a specific pod. Volumes serve as a plug-in mechanism for connecting ephemeral containers to persistent data stores located elsewhere.

Deployment

A deployment is a collection of identical pods that are managed by the Kubernetes Deployment Controller. A deployment specifies the number of pod replicas that will be created. If pods or nodes encounter problems, the Kubernetes Scheduler ensures that additional pods are scheduled on healthy nodes.

Typically, deployments are created and managed using `kubectl create` or `kubectl apply`. Make a deployment by defining a manifest file in YAML format.

Kubernetes Architecture

Kubernetes architecture consists of the following main parts:

- Control Plane (master)
  - kube-apiserver
  - etcd
  - kube-scheduler
  - kube-controller-manager
  - cloud-controller-manager
- Node components
  - kubelet
  - kube-proxy
  - Container runtime

Install tools

kubectl

`kubectl` makes it possible to run commands against Kubernetes clusters using the `kubectl` command-line tool. `kubectl` can be used to deploy applications, inspect and manage cluster resources, and inspect logs. See the `kubectl` reference documentation for more information.

kind

`kind` command makes it possible to run Kubernetes on a local machine. As a prerequisite, Docker needs to be installed and configured. What `kind` is doing is to run local Kubernetes clusters using Docker container “nodes”.

In depth docs

» Kubernetes documentation

Was this page helpful?