Kubernetes

App Development

GOLDEN GUIDE TO CERTIFIED KUBERNETES APPLICATION DEVELOPMENT

by Matthew Palmer

Overview

Kubernetes is an operating system for running web services. You tell it to run a process, the characteristics of that process, and the resources that process needs. It will run that process—but unlike a normal operating system, it can run it across an entire cluster of computers.

Fundamentally, Kubernetes lets you run processes on a cluster of computers without caring about the cluster itself. As a result, you get access to some powerful Kubernetes features:

- guarantee the process is always running, and if it stops, restart it, potentially on a different machine
- make sure there are a certain number of the process running
- run the process at a certain time, and check that it's successful

You don't have to worry about any details of the machines your process is running on. Kubernetes manages the machine itself, watches out for failures, tracks resource usage, and configures networking rules.

Kubernetes Objects

Your application environment is defined by a collection of objects. Kubernetes provides many types of objects, all with unique characteristics, that you combine to perform specific tasks.

The two most basic attributes of any Kubernetes object are the Kubernetes API version it's from and the kind of object it is.

An object also has metadata. Like metadata in other contexts, it stores meta-information about the object itself. In the case of Kubernetes objects, this is the object's name, its labels, and its annotations.

The two most important properties of an object are spec and status. spec is how you define your object

A Kubernetes Object

Version	*v1, v1beta1, ...*
Kind	*Pod, Service, ...*

Metadata
Name
Labels
Annotations

Spec
Your desired state

Status
Kubernetes' actual state

—it tells Kubernetes the desired state of your object. status is reality—it's where Kubernetes stores how your object is actually running. If the actual state (status) differs from the ideal state (spec), Kubernetes will act to get that object to its ideal state.

You might find this description very high level and abstract. However, this pattern appears repeatedly in Kubernetes—use it as a scaffold to which you attach your learning in the next chapters.

As a concrete example, here's an annotated YAML configuration file that defines a Pod running nginx.

```yaml
# Which version of Kubernetes this object comes from
apiVersion: v1

# What type of object am I?
kind: Pod

#
# Metadata -  Meta-information about the object itself
#
metadata:
  name: nginx-pod

  # (Labels and metadata will be explained later!)
  labels:
    key: value
  metadata:
    otherkey: othervalue

#
# Spec - what you want your object to be
#
spec:
  containers:
  - name: nginx
    image: nginx:1.7.9
    ports:
    - containerPort: 80
```