- A Label can indicate a logical grouping of set of Pods and give an application identity to them.
- In addition to the preceding typical use cases, labels can be used to store metadata. It may be difficult to predict what a label could be used for, but it is best to have enough labels to describe all important aspects of the Pods. For example, having labels to indicate the logical group of an application, the business characteristics and criticality, the specific runtime platform dependencies such as hardware architecture, or location preferences are all useful.

  Later, these labels can be used by the scheduler for more fine-grained scheduling, or the same labels can be used from the command line for managing the matching Pods at scale. However, you should not go overboard and add too many labels in advance. You can always add them later if needed. Removing labels is much riskier as there is no straight-forward way of finding out what a label is used for, and what unintended effect such an action may cause.

Annotations

Another primitive very similar to labels is called *annotations*. Like labels, annotations are organized as a map, but they are intended for specifying nonsearchable metadata and for machine usage rather than human.

The information on the annotations is not intended for querying and matching objects. Instead, it is intended for attaching additional metadata to objects from various tools and libraries we want to use. Some examples of using annotations include build IDs, release IDs, image information, timestamps, Git branch names, pull request numbers, image hashes, registry addresses, author names, tooling information, and more. So while labels are used primarily for query matching and performing actions on the matching resources, annotations are used to attach metadata that can be consumed by a machine.

Namespaces

Another primitive that can also help in the management of a group of resources is the Kubernetes *namespace*. As we have described, a namespace may seem similar to a label, but in reality, it is a very different primitive with different characteristics and purpose.

Kubernetes namespaces allow dividing a Kubernetes cluster (which is usually spread across multiple hosts) into a logical pool of resources. Namespaces provide scopes for Kubernetes resources and a mechanism to apply authorizations and other policies to a subsection of the cluster. The most common use case of namespaces is representing different software environments such as development, testing, integration testing, or production. Namespaces can also be used to achieve multitenancy, and provide isola-