```
    name: random-generator
    volumeMounts:
    - mountPath: "/logs"
      name: log-volume
  volumes:
  - name: log-volume
    persistentVolumeClaim:                    ❶
      claimName: random-generator-log
```

❶ Dependency of a PVC to be present and bound

The scheduler evaluates the kind of volume a Pod requires, which affects where the Pod gets placed. If the Pod needs a volume that is not provided by any node on the cluster, the Pod is not scheduled at all. Volumes are an example of a runtime dependency that affects what kind of infrastructure a Pod can run and whether the Pod can be scheduled at all.

A similar dependency happens when you ask Kubernetes to expose a container port on a specific port on the host system through `hostPort`. The usage of a `hostPort` creates another runtime dependency on the nodes and limits where a Pod can be scheduled. `hostPort` reserves the port on each node in the cluster and limit to maximum one Pod scheduled per node. Because of port conflicts, you can scale to as many Pods as there are nodes in the Kubernetes cluster.

A different type of dependency is configurations. Almost every application needs some configuration information and the recommended solution offered by Kubernetes is through ConfigMaps. Your services need to have a strategy for consuming settings—either through environment variables or the filesystem. In either case, this introduces a runtime dependency of your container to the named ConfigMaps. If not all of the expected ConfigMaps are created, the containers are scheduled on a node, but they do not start up. ConfigMaps and Secrets are explained in more details in Chapter 19, *Configuration Resource*, and Example 2-2 shows how these resources are used as runtime dependencies.

Example 2-2. Dependency on a ConfigMap

```
apiVersion: v1
kind: Pod
metadata:
  name: random-generator
spec:
  containers:
  - image: k8spatterns/random-generator:1.0
    name: random-generator
    env:
    - name: PATTERN
      valueFrom:
```