could also do something more sophisticated like forward the data to a full logging infrastructure. For the REST service, it doesn't matter what happens to the log data, and you can easily exchange the *Ambassador* by reconfiguring the Pod without touching the main container.

Example 17-1. Ambassador processing log output

```
apiVersion: v1
kind: Pod
metadata:
  name: random-generator
  labels:
    app: random-generator
spec:
  containers:
  - image: k8spatterns/random-generator:1.0                ❶
    name: main
    env:
    - name: LOG_URL                                        ❷
      value: http://localhost:9009
    ports:
    - containerPort: 8080
      protocol: TCP
  - image: k8spatterns/random-generator-log-ambassador ❸
    name: ambassador
```

❶ Main application container providing a REST service for generating random numbers

❷ Connection URL for communicating with the *Ambassador* via localhost

❸ Ambassador running in parallel and listening on port 9009 (which is not exposed to the outside of the Pod)

Discussion

At a higher level, the *Ambassador* is a *Sidecar* pattern. The main difference between *Ambassador* and *Sidecar* is that an *Ambassador* does not enhance the main application with additional capability. Instead, it acts merely as a smart proxy to the outside world, where it gets its name from (this pattern is sometimes referred to as the *Proxy* pattern as well).