So this type of Service works only when the cloud provider has Kubernetes support and provisions a load balancer.

We can create a Service with a load balancer by specifying the type `LoadBalancer`. Kubernetes then will add IP addresses to the `.spec` and `.status` fields, as shown in Example 12-7.

Example 12-7. Service of type LoadBalancer

```
apiVersion: v1
kind: Service
metadata:
  name: random-generator
spec:
  type: LoadBalancer
  clusterIP: 10.0.171.239                    ❶
  loadBalancerIP: 78.11.24.19
  selector:
    app: random-generator
  ports:
  - port: 80
    targetPort: 8080
    protocol: TCP
status:                                      ❷
  loadBalancer:
    ingress:
    - ip: 146.148.47.155
```

❶ Kubernetes assigns `clusterIP` and `loadBalancerIP` when they are available.

❷ The `status` field is managed by Kubernetes and adds the Ingress IP.

With this definition in place, an external client application can open a connection to the load balancer, which picks a node and locates the Pod. The exact way that load-balancer provisioning is performed and service discovery varies among cloud providers. Some cloud providers will allow defining the load-balancer address, and some will not. Some offer mechanisms for preserving the source address, and some replace that with the load-balancer address. You should check the specific implementation provided by your cloud provider of choice.

Yet another type of Service is available: *headless* services, for which you don't request a dedicated IP address. You create a headless service by specifying `clusterIP: None` within the Service's `spec:` section. For headless services, the backing Pods are added to the internal DNS server and are most useful for implementing Services to StatefulSets, as described in detail in Chapter 11, *Stateful Service*.