Client. Zudem gibt es eine Obergrenze für die Art und Menge an Informationen, die auf eine typische DNS-Abfrage hin zurückgegeben werden können. Ab etwa 20 bis 30 A-(Address-)Records für einen einzelnen Namen wird es kritisch. SRV-(Service-) Records lösen manche Probleme, aber es ist häufig schwer, mit ihnen umzugehen. Und schließlich reagieren Clients auf mehrere IPs in einem DNS-Record im Allgemeinen so, dass sie die erste IP-Adresse nutzen und sich darauf verlassen, dass der DNS-Server die Reihenfolge der Einträge zufällig zurückgibt oder ein Round-Robin-Verfahren nutzt. Das ist aber kein Ersatz für ein eher zweckbestimmtes Load Balancing.

7.2 Das Service-Objekt

Eine echte Service-Discovery beginnt in Kubernetes mit einem Service-Objekt. Ein solches Service-Objekt ist eine Möglichkeit, einen benannten Label-Selektor zu erstellen. Wie Sie sehen werden, bietet uns das Objekt auch noch ein paar andere nette Dinge.

So wie wir mit `kubectl run` einfach ein Kubernetes-Deployment erzeugen können, schaffen wir mit `kubectl expose` einen Service. Wir werden in Kapitel 10 genauer auf Deployments eingehen, hier reicht es aus, sie als Instanz eines Microservice zu betrachten. Erstellen wir also ein paar Deployments und Services, um zu sehen, wie diese funktionieren:

```
$ kubectl create deployment alpaca-prod \
  --image=gcr.io/kuar-demo/kuard-amd64:blue \
  --port=8080
$ kubectl scale deployment alpaca-prod --replicas 3
$ kubectl expose deployment alpaca-prod
$ kubectl create deployment bandicoot-prod \
  --image=gcr.io/kuar-demo/kuard-amd64:green \
  --port=8080
$ kubectl scale deployment bandicoot-prod --replicas 2
$ kubectl expose deployment bandicoot-prod
$ kubectl get services -o wide

NAME             CLUSTER-IP    ... PORT(S)  ... SELECTOR
alpaca-prod      10.115.245.13 ... 8080/TCP ... app=alpaca-prod
bandicoot-prod   10.115.242.3  ... 8080/TCP ... app=bandicoot-prod
kubernetes       10.115.240.1  ... 443/TCP  ... <none>
```

Nach dem Ausführen dieser Befehle haben wir drei Services. Die eben erstellten sind `alpaca-prod` und `bandicoot-prot`. Der `kubernetes`-Service wird automatisch für Sie erstellt, sodass Sie innerhalb der Anwendung die Kubernetes-API finden und mit ihr kommunizieren können.

Schauen wir uns die Spalte `SELECTOR` an, sehen wir, dass der Service `alpaca-prod` einfach einem Selektor einen Namen gibt und festlegt, über welche Ports mit diesem Service gesprochen werden kann. Der Befehl `kubectl expose` holt sich dabei den Label-Selektor und die relevanten Ports (in diesem Fall 8080) praktischerweise direkt aus der Deployment-Definition.