Das ist toll, aber was, wenn Sie haufenweise Pods haben? Sie werden diese vermutlich basierend auf ihren Labels filtern wollen, die als Teil des Deployments zugewiesen wurden. Machen wir das nur für die `alpaca`-Anwendung:

```
$ kubectl get pods -o wide --selector=app=alpaca

NAME                          ... IP          ...
alpaca-prod-3408831585-bpzdz ... 10.112.1.54 ...
alpaca-prod-3408831585-kncwt ... 10.112.2.84 ...
alpaca-prod-3408831585-l9fsq ... 10.112.2.85 ...
```

Das sind schon mal Grundlagen für ein Service-Discovery! Wir können immer Labels nutzen, um die Pods herauszufinden, an denen wir interessiert sind, und dann deren IP-Adressen auslesen. Aber es kann knifflig sein, die richtigen Labels synchron zu halten. Daher wurde das Service-Objekt geschaffen.

7.5.3 kube-proxy und Cluster-IPs

Cluster-IPs sind stabile virtuelle IPs, über die der Netzverkehr per Load Balancing auf die Endpunkte eines Service verteilt wird. Diese Zauberei geschieht durch eine Komponente, die auf jedem Knoten im Cluster läuft und den Namen `kube-proxy` trägt (Abb. 7–1).

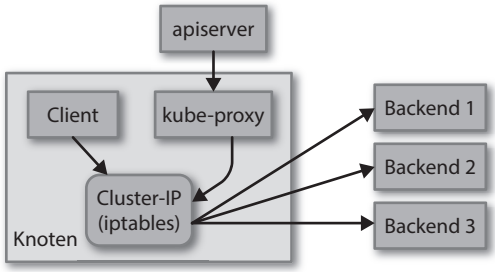

Abb. 7–1 *Konfigurieren und Verwenden einer Cluster-IP*

In Abbildung 7–1 lauscht der `kube-proxy` über den API-Server auf neue Services im Cluster. Dann programmiert er einen Satz von `iptables`-Regeln im Kernel dieses Hosts, um die Ziele der Pakete umzuschreiben, sodass sie auf einen der Endpunkte für diesen Service umgeleitet werden. Wenn sich die Endpunkte für einen Service ändern (weil Pods kommen und gehen oder aufgrund eines fehlgeschlagenen Readiness-Checks), werden die `iptables`-Regeln umgeschrieben.

Die Cluster-IP selbst wird normalerweise beim Erstellen des Service durch den API-Server zugewiesen. Aber beim Erstellen kann der Benutzer auch eine bestimmte Cluster-IP angeben. Einmal gesetzt, lässt sie sich nicht mehr verändern, ohne das Service-Objekt zu löschen und neu anzulegen.