Kubernetes

acoshift (2017)

DAY 1

YAML

- stands for “YAML Ain’t Markup Language”
- is a human friendly data serialization standard for all programming language

YAML

```yaml
name: Courses
list:
- name: Go for Beginner
  price: 600
- name: Redis Fundamental
  price: 300
- name: RxJS for Beginner
  price: 500
```

JSON

```json
{
  "name": "Courses",
  "list": [
    {
      "name": "Go for Beginner",
      "price": 600
    },
    {
      "name": "Redis Fundamental",
      "price": 300
    },
    {
      "name": "RxJS for Beginner",
      "price": 500
    }
  ]
}
```

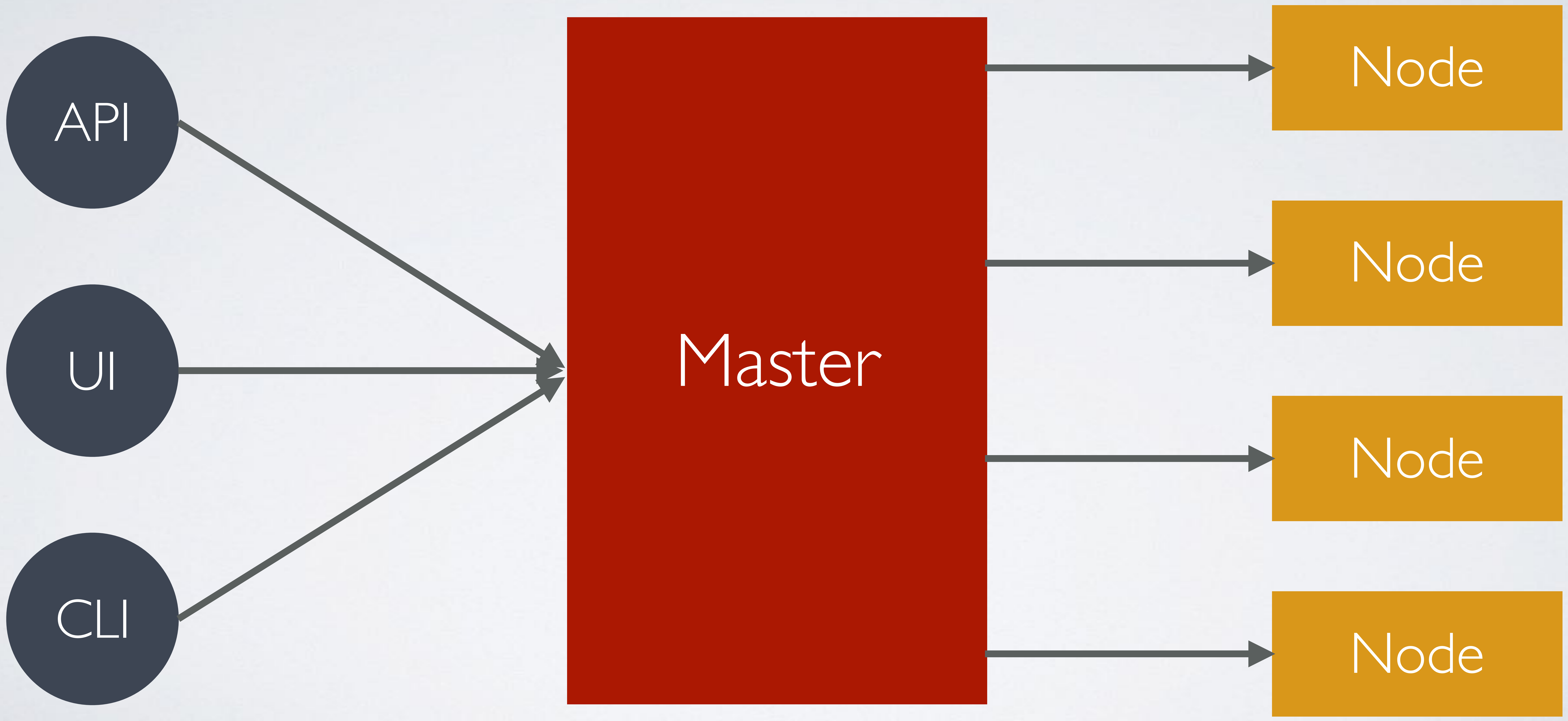
Google Container Registry

https://cloud.google.com/container-registry/

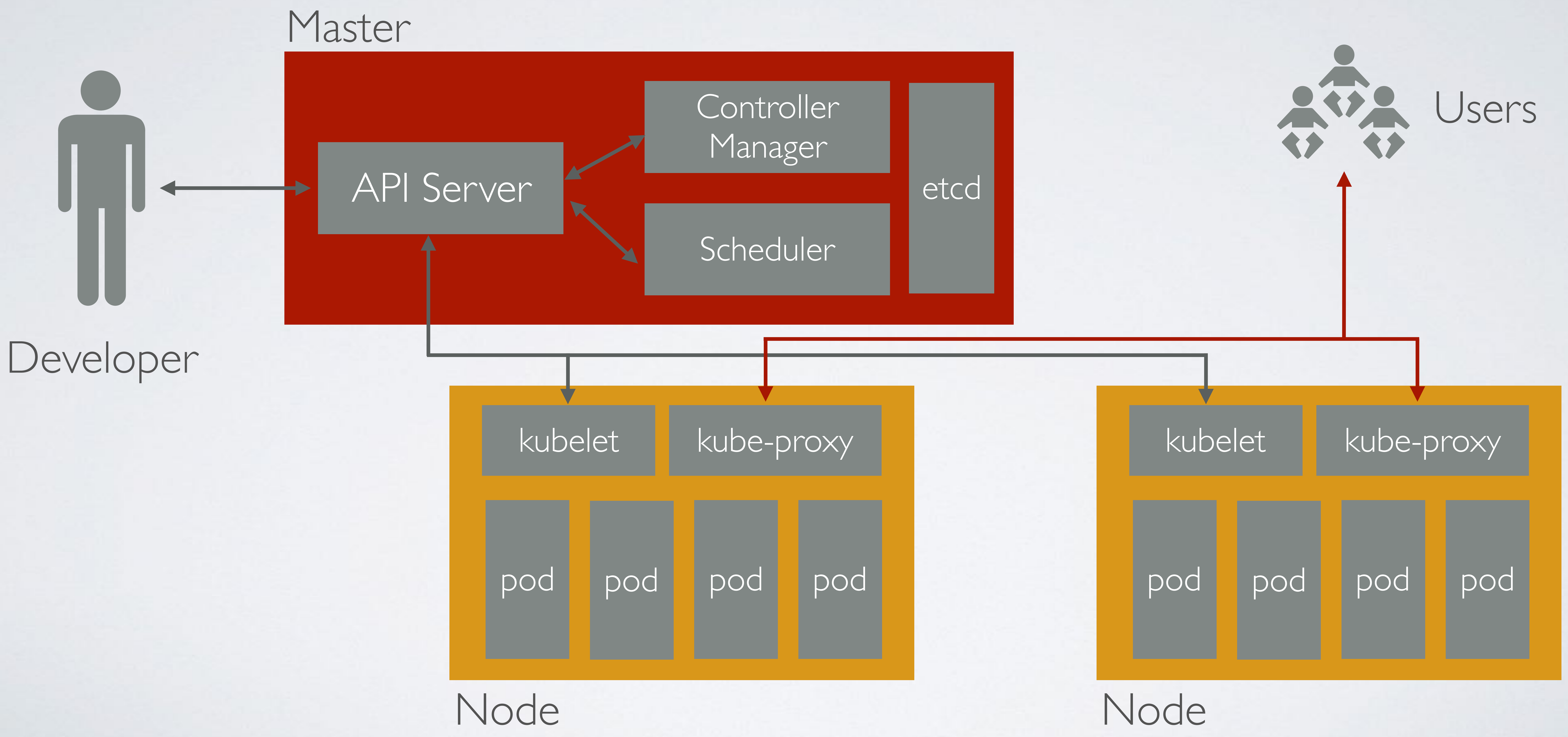
```
$ docker push acoshift/backend:1.0.0
$ gcloud docker -- push gcr.io/myproject/backend:1.0.0

$ docker pull acoshift/backend:1.0.0
$ gcloud docker -- pull gcr.io/myproject/backend:1.0.0
$ docker login -u _json_key -p "$(cat keyfile.json)" https://gcr.io
$ docker push gcr.io/myproject/backend:1.0.0
$ docker pull gcr.io/myproject/backend:1.0.0
```

https://console.cloud.google.com/gcr/images/google-containers/GLOBAL

Kubernetes Architecture



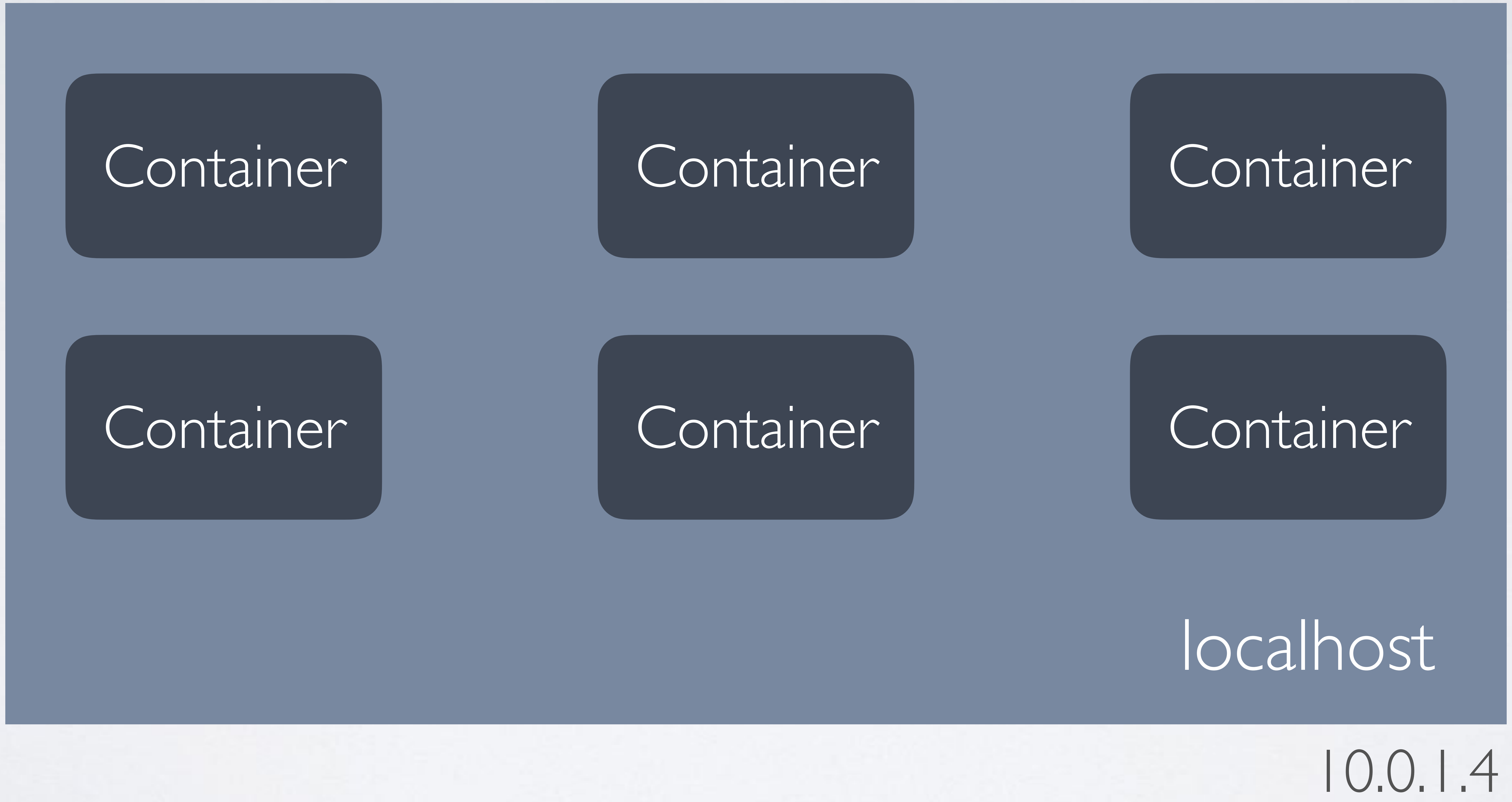
Kubernetes Architecture



Pods (po)

(pod of whales / pea pod)

Pod


Container
Container
Container
Container
Container
Container
localhost
10.0.1.4

```yaml
kind: Pod
apiVersion: v1
metadata:
  name: echoserver
spec:
  containers:
  - name: echoserver
    image: gcr.io/google-containers/echoserver:1.6
    ports:
    - containerPort: 8080
```

just additional information

all ports listening on 0.0.0.0 will be accessible from network

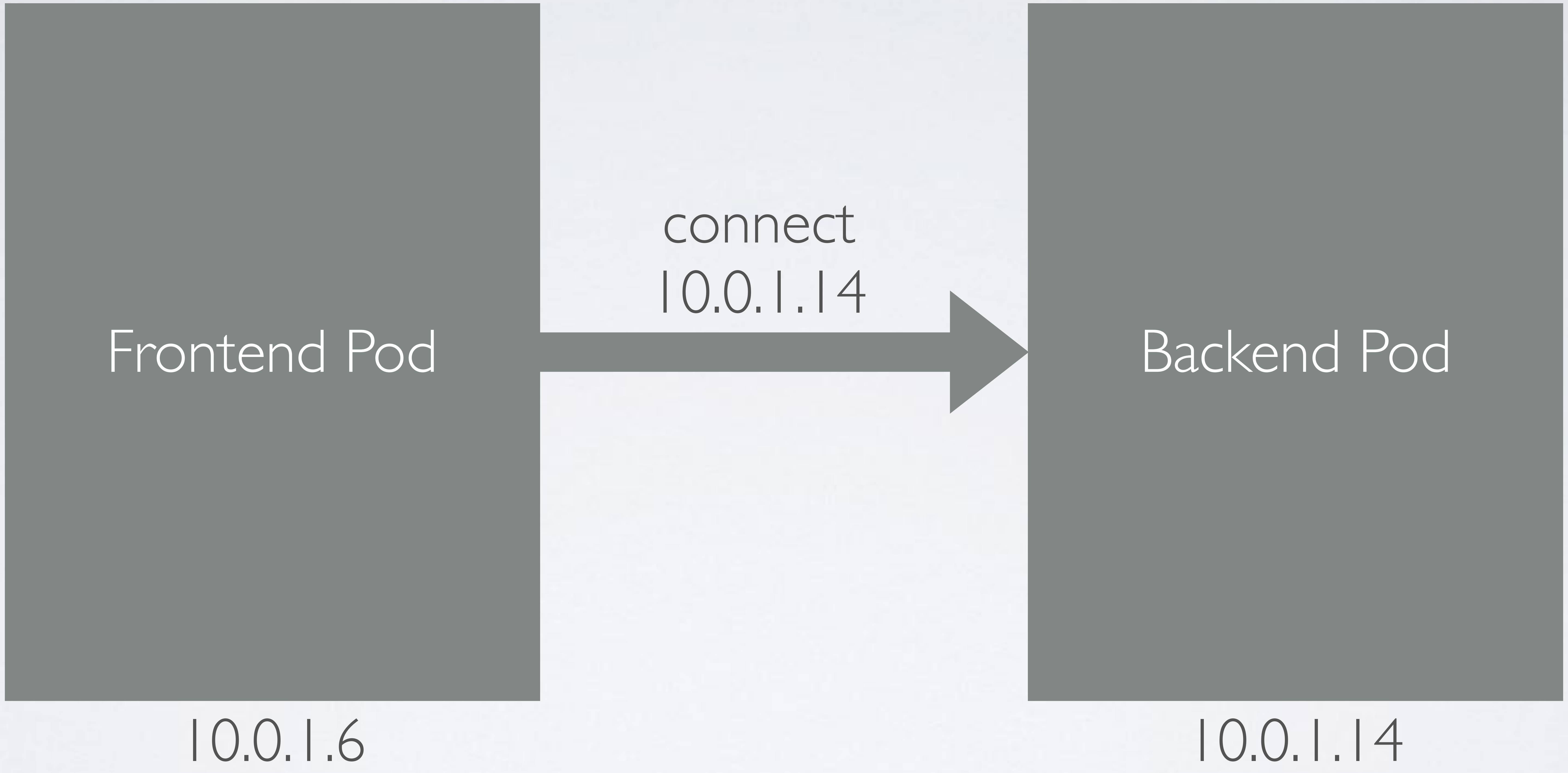
```
$ kubectl create -f 01-pod.yaml
pod "echoserver" created
```

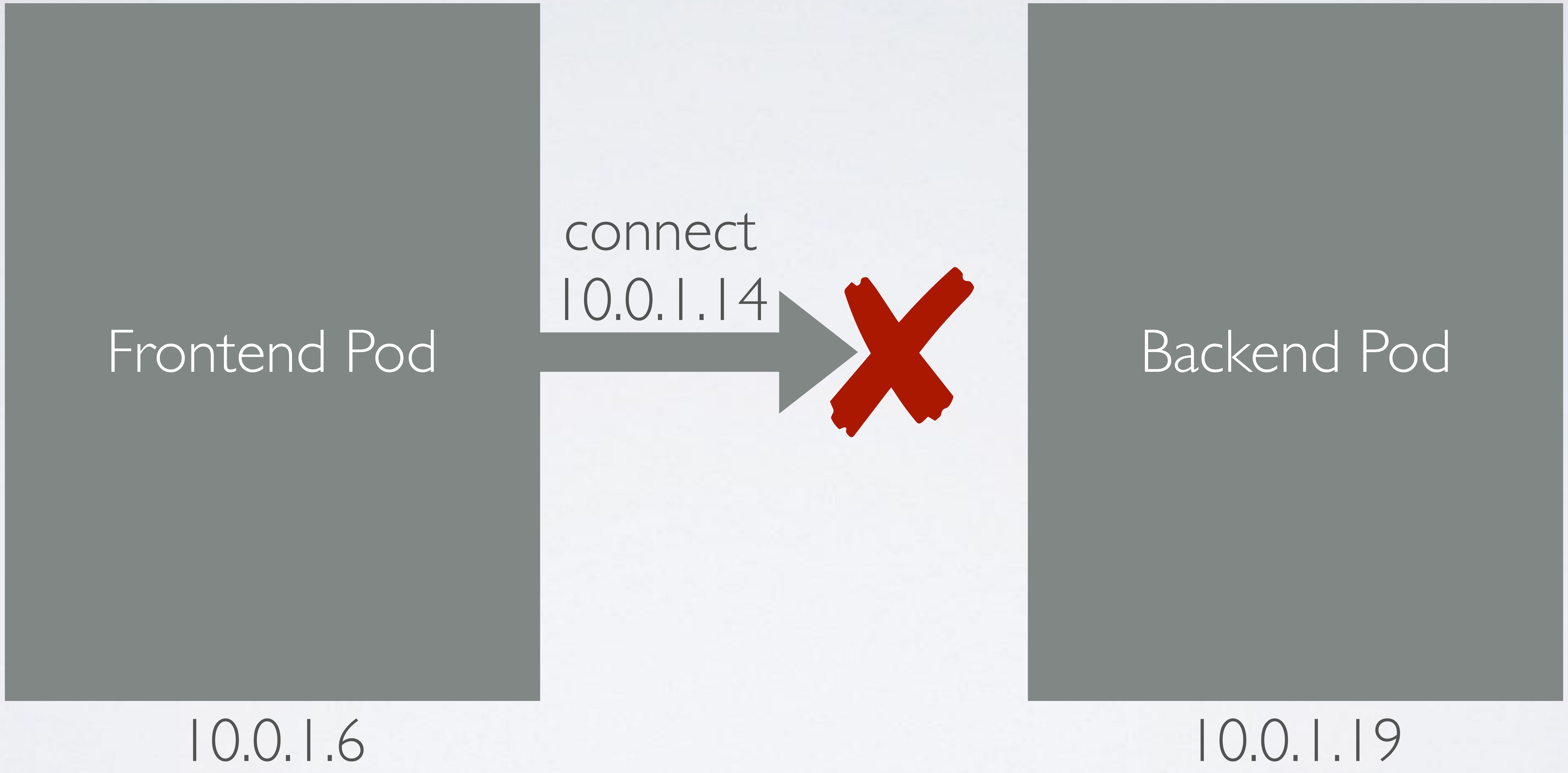
```
$ kubectl get pods
NAME            READY     STATUS    RESTARTS   AGE
echoserver      1/1       Running   0          4m
```

```
$ kubectl port-forward echoserver 9000:8080
Forwarding from 127.0.0.1:9000 -> 8080
Forwarding from [::1]:9000 -> 8080
```

```
$ curl localhost:9000
Hostname: echoserver

Pod Information:
  -no pod information available-

Server values:
  server_version=nginx: 1.13.1 - lua: 10008

Request Information:
  client_address=127.0.0.1
  method=GET
  real path=/
  query=
  request_version=1.1
  request_uri=http://localhost:8080/

Request Headers:
  accept=*/*
  host=localhost:9000
  user-agent=curl/7.51.0

Request Body:
  -no body in request-
```

```
$ kubectl delete pod echoserver
pod "echoserver" deleted
```


Frontend Pod
connect
10.0.1.14
Backend Pod
10.0.1.6
10.0.1.14


Frontend Pod
connect
10.0.1.14
Backend Pod
10.0.1.6
10.0.1.19

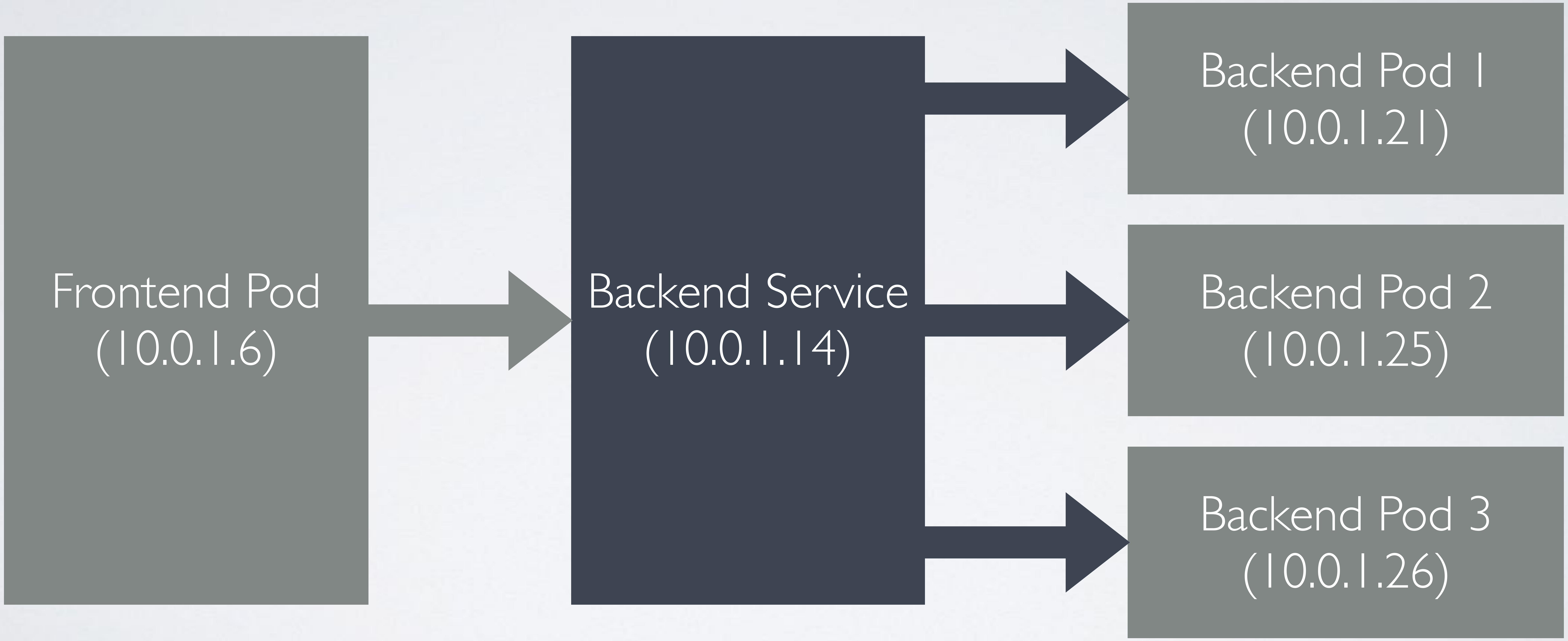
Services (svc)

an abstraction which defines a logical set of Pods and a policy by which to access them

Service Types

- ClusterIP
- NodePort
- LoadBalancer
- ExternalName

ClusterIP


Frontend Pod
(10.0.1.6)
Backend Service
(10.0.1.14)
Backend Pod 1
(10.0.1.21)
Backend Pod 2
(10.0.1.25)
Backend Pod 3
(10.0.1.26)

```yaml
kind: Pod
apiVersion: v1
metadata:
  name: echoserver
  labels:
    app: echoserver
spec:
  containers:
  - name: echoserver
    image: gcr.io/google-containers/echoserver:1.6
    ports:
    - containerPort: 8080
```

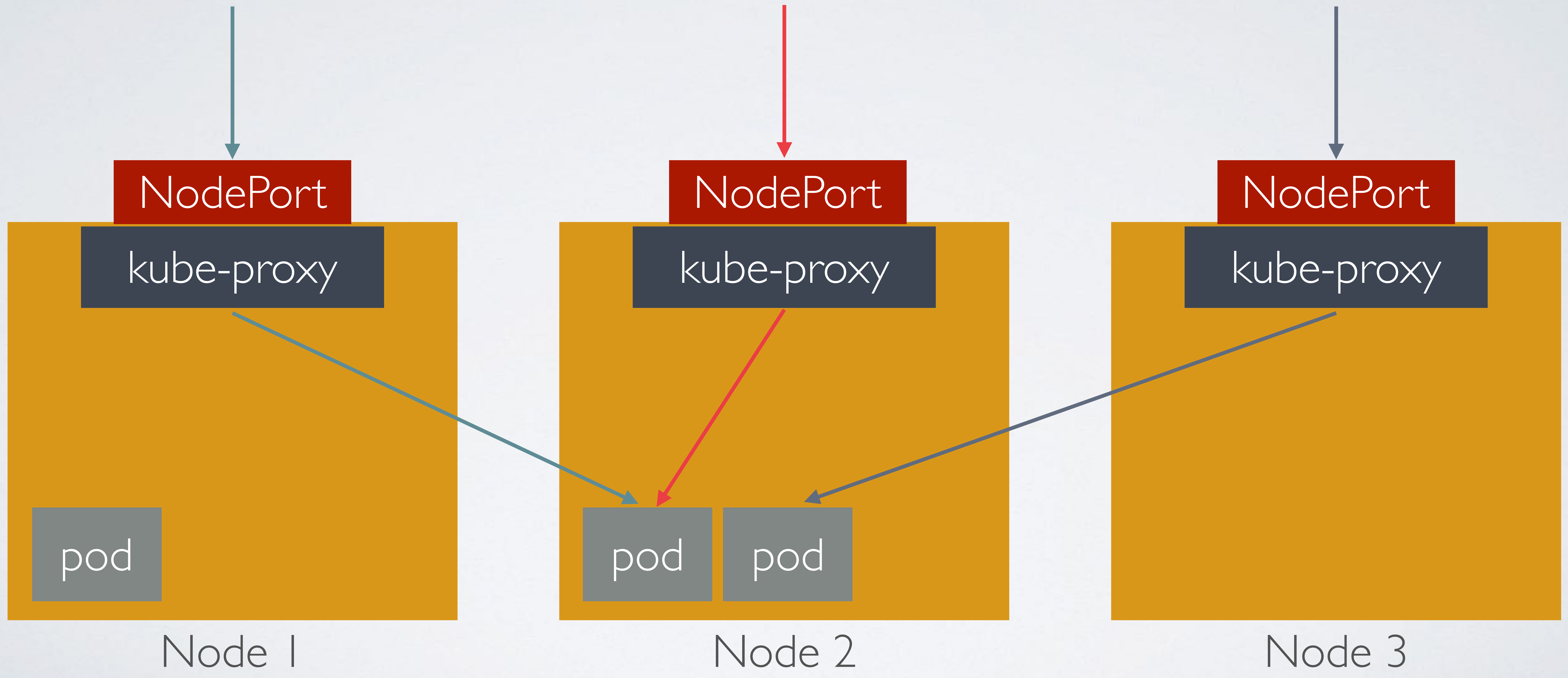
```yaml
kind: Service
apiVersion: v1
metadata:
  name: echoserver
spec:
  selector:
    app: echoserver
  ports:
  - port: 80
    targetPort: 8080
```

```
$ kubectl create -f 02-service.yaml
pod "echoserver" created
service "echoserver" created
```

```
$ kubectl get services
NAME          CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
echoserver    10.3.248.15   <none>        80/TCP    11s
```

```
$ kubectl run -i -t --rm busybox --image=busybox
$ wget -O - http://echoserver
$ kubectl delete -f 02-service.yaml
```

NodePort

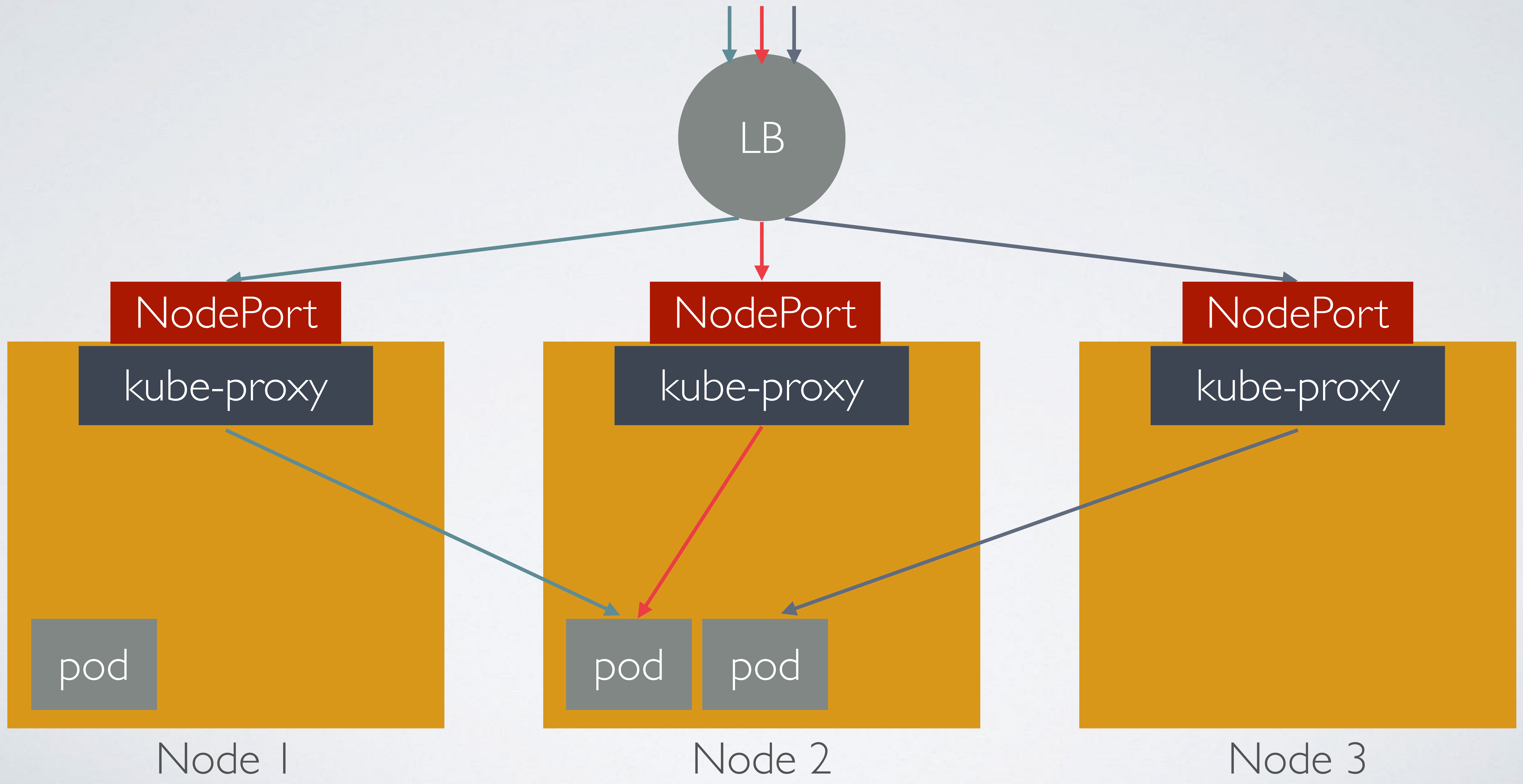

NodePort
kube-proxy
pod
Node 1
NodePort
kube-proxy
pod
pod
Node 2
NodePort
kube-proxy
Node 3

```yaml
kind: Service
apiVersion: v1
metadata:
  name: echoserver
spec:
  type: NodePort
  selector:
    app: echoserver
  ports:
  - port: 80
    targetPort: 8080
    nodePort: 31000
```

valid port:
30000-32767

```
$ curl http://serverIP:31000
```

LoadBalancer

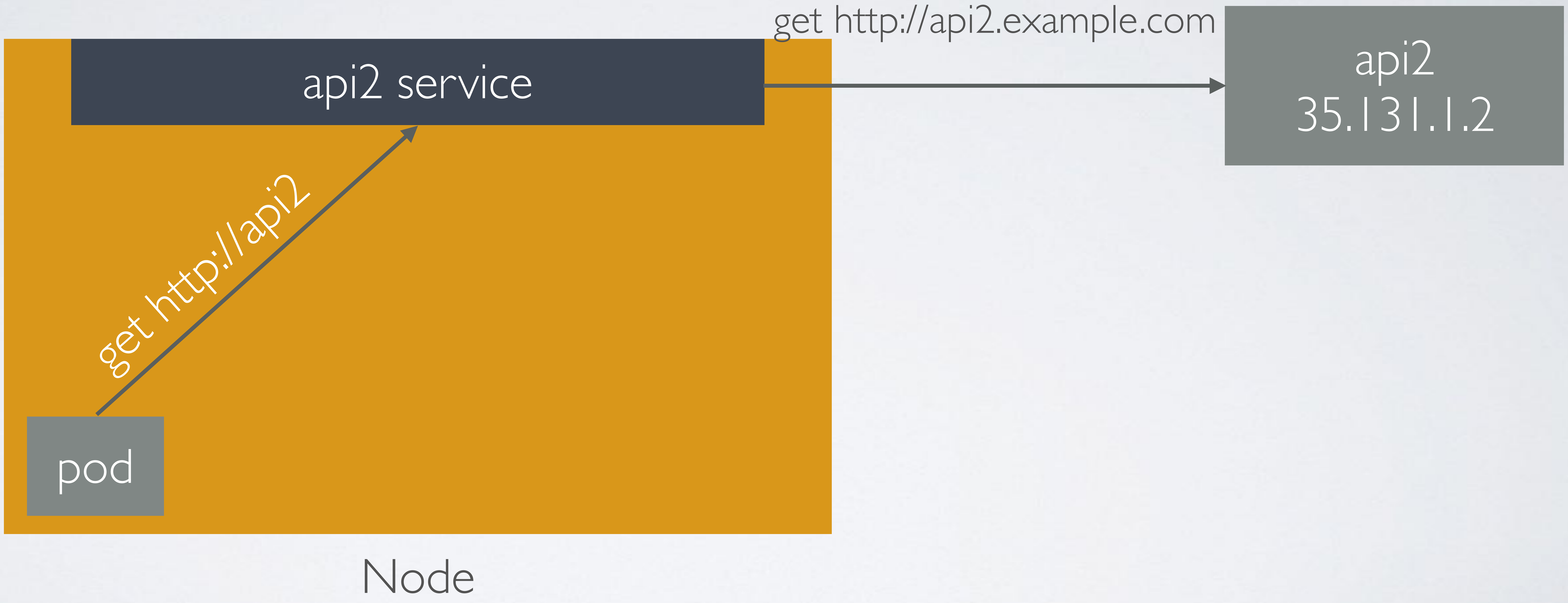


```yaml
kind: Service
apiVersion: v1
metadata:
  name: echoserver
spec:
  type: LoadBalancer
  selector:
    app: echoserver
  ports:
  - port: 80
    targetPort: 8080
  loadBalancerIP: 35.185.1.1
```

optional static ip

```
$ curl http://loadbalcnerIP
```

ExternalName



```yaml
kind: Service
apiVersion: v1
metadata:
  name: google
spec:
  type: ExternalName
  externalName: google.com
```

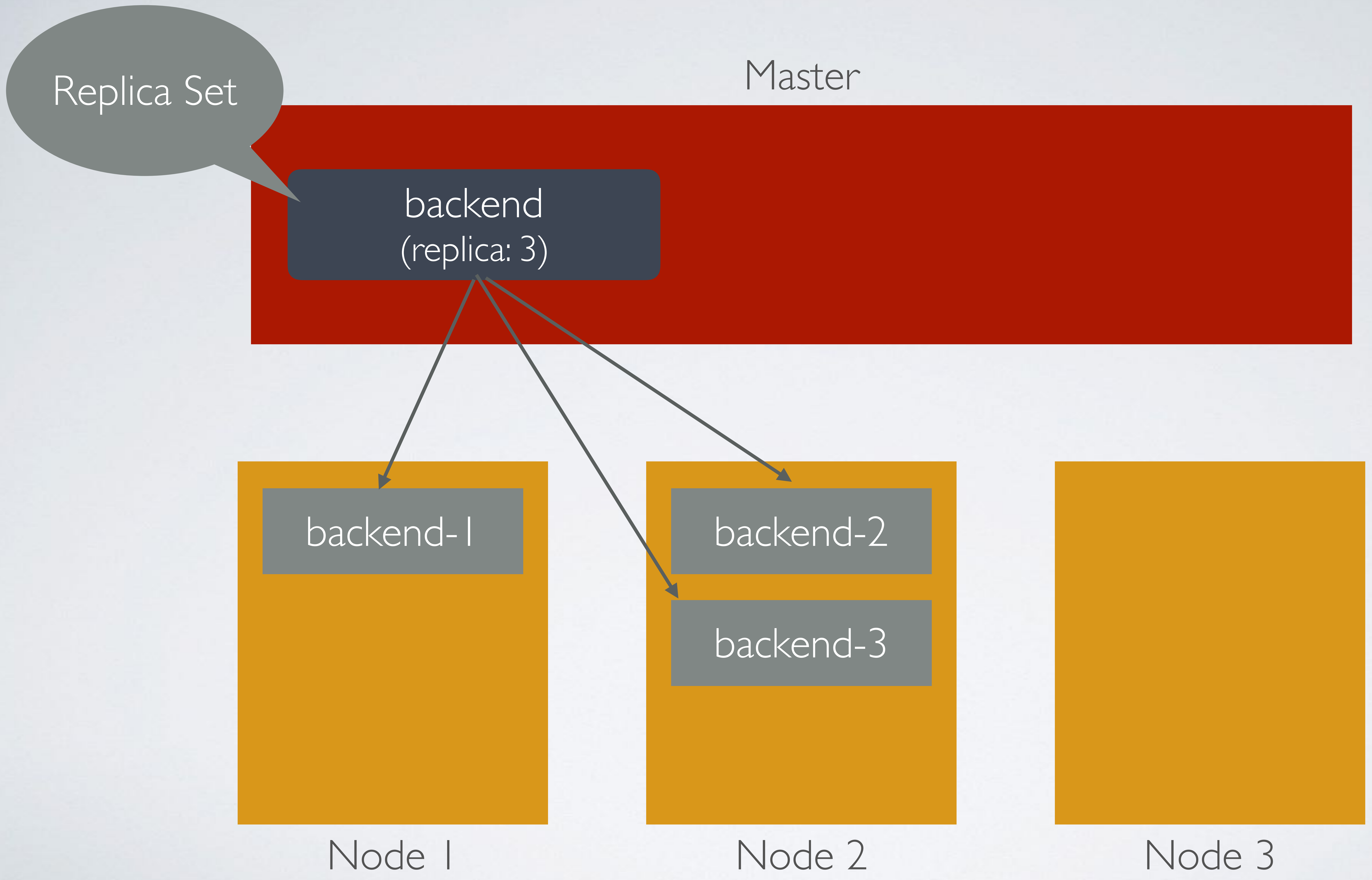
```
$ kubectl run -i -t --rm busybox --image=busybox
$ wget -O - --header="Host: www.google.com" http://google
```

~~Replication Controller (rc)~~

Replica Sets (rs)

the next-generation Replication Controller

ensures that a specified number of pod “replicas” are running at any given time

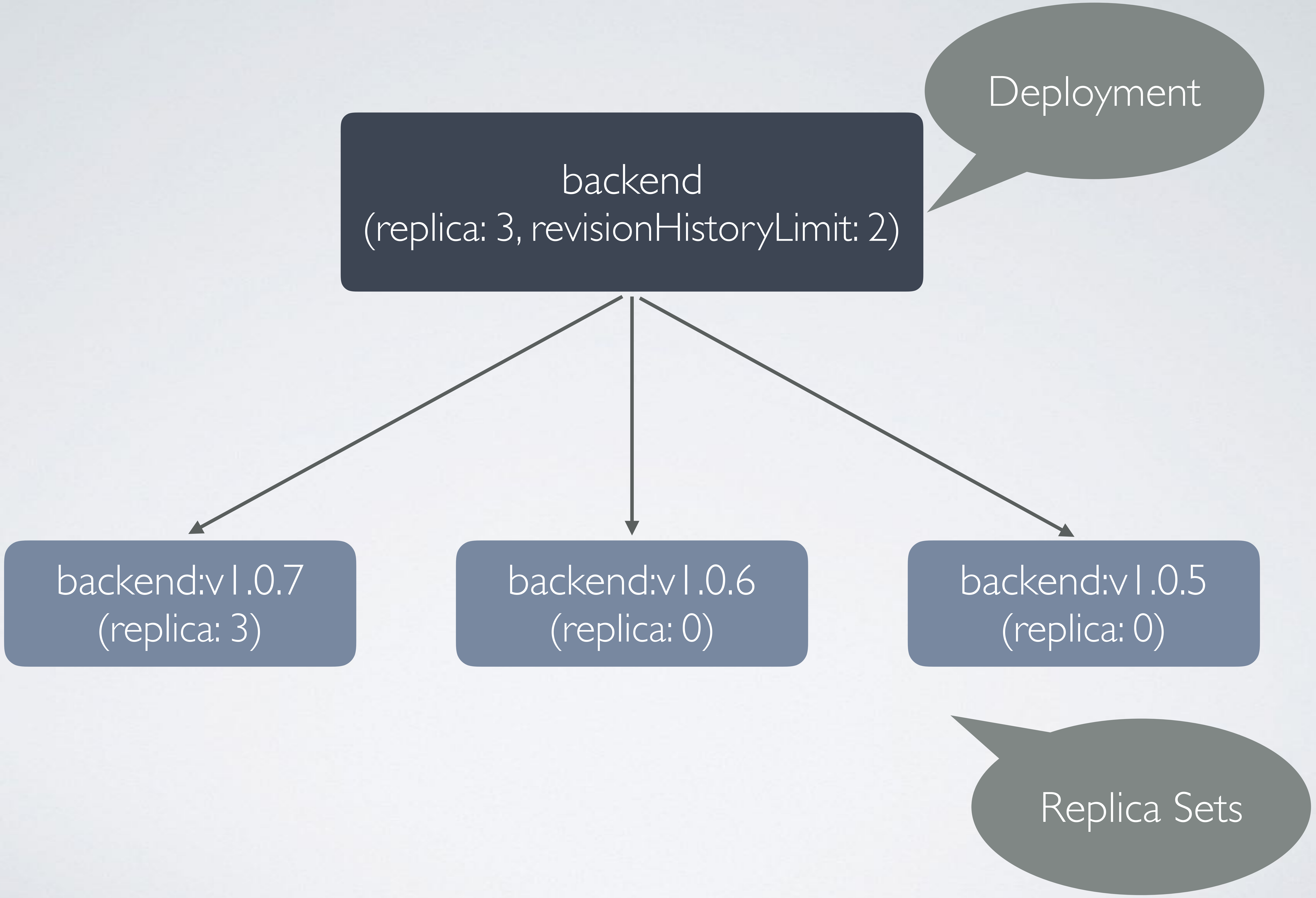

Replica Set
Master
backend
(replica: 3)
backend-1
backend-2
backend-3
Node 1
Node 2
Node 3

```yaml
kind: ReplicaSet
apiVersion: extensions/v1beta1
metadata:
  name: echoserver
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: echoserver
    spec:
      containers:
      - name: echoserver
        image: gcr.io/google-containers/echoserver:1.6
        ports:
        - containerPort: 8080
```

Pod

Deployments (deploy)

provides declarative updates for Pods and ReplicaSets


Deployment
backend
(replica: 3, revisionHistoryLimit: 2)
backend:v1.0.7
(replica: 3)
backend:v1.0.6
(replica: 0)
backend:v1.0.5
(replica: 0)
Replica Sets

```yaml
kind: Deployment
apiVersion: apps/v1beta1
metadata:
  name: echoserver
spec:
  replicas: 3
  revisionHistoryLimit: 2
  template:
    metadata:
      labels:
        app: echoserver
    spec:
      containers:
      - name: echoserver
        image: gcr.io/google-containers/echoserver:1.1
        ports:
        - containerPort: 8080
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
      maxSurge: 1
```

Strategy

Default

- RollingUpdate — updates one pod at a time
  - Max Unavailable — maximum number of Pods that can be unavailable during the update process
  - Max Surge — maximum number of Pods that can be created above the desired number of Pods
- Recreate — All existing Pods are killed before new ones are created

```
$ kubectl create -f 07-deployment.yaml --record=true
deployment "echoserver" created

$ kubectl set image deployment/echoserver echoserver=gcr.io/google-containers/echoserver:1.2
deployment "echoserver" image updated

$ kubectl rollout status deployment/echoserver
Waiting for rollout to finish: 1 out of 3 new replicas have been updated...
Waiting for rollout to finish: 1 out of 3 new replicas have been updated...
Waiting for rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for rollout to finish: 1 old replicas are pending termination...
Waiting for rollout to finish: 1 old replicas are pending termination...
deployment "echoserver" successfully rolled out
```

```
$ kubectl set image deployment/echoserver echoserver=gcr.io/google-containers/echoserver:1.3
deployment "echoserver" image updated

$ kubectl rollout history deployment/echoserver
deployments "echoserver"
REVISION        CHANGE-CAUSE
1               kubectl create --filename=07-deployment.yaml --record=true
2               kubectl set image deployment/echoserver echoserver=gcr.io/google-containers/echoserver:1.2
3               kubectl set image deployment/echoserver echoserver=gcr.io/google-containers/echoserver:1.3
```

```
$ kubectl rollout history deployment/echoserver --revision=2
deployments "echoserver" with revision #2
Pod Template:
  Labels:       app=echoserver
        pod-template-hash=1885346732
  Annotations:  kubernetes.io/change-cause=kubectl set image deployment/echoserver echoserver=gcr.io/google-containers/echoserver:1.2
  Containers:
   echoserver:
    Image:      gcr.io/google-containers/echoserver:1.2
    Port:       8080/TCP
    Environment:        <none>
    Mounts:     <none>
  Volumes:      <none>
```

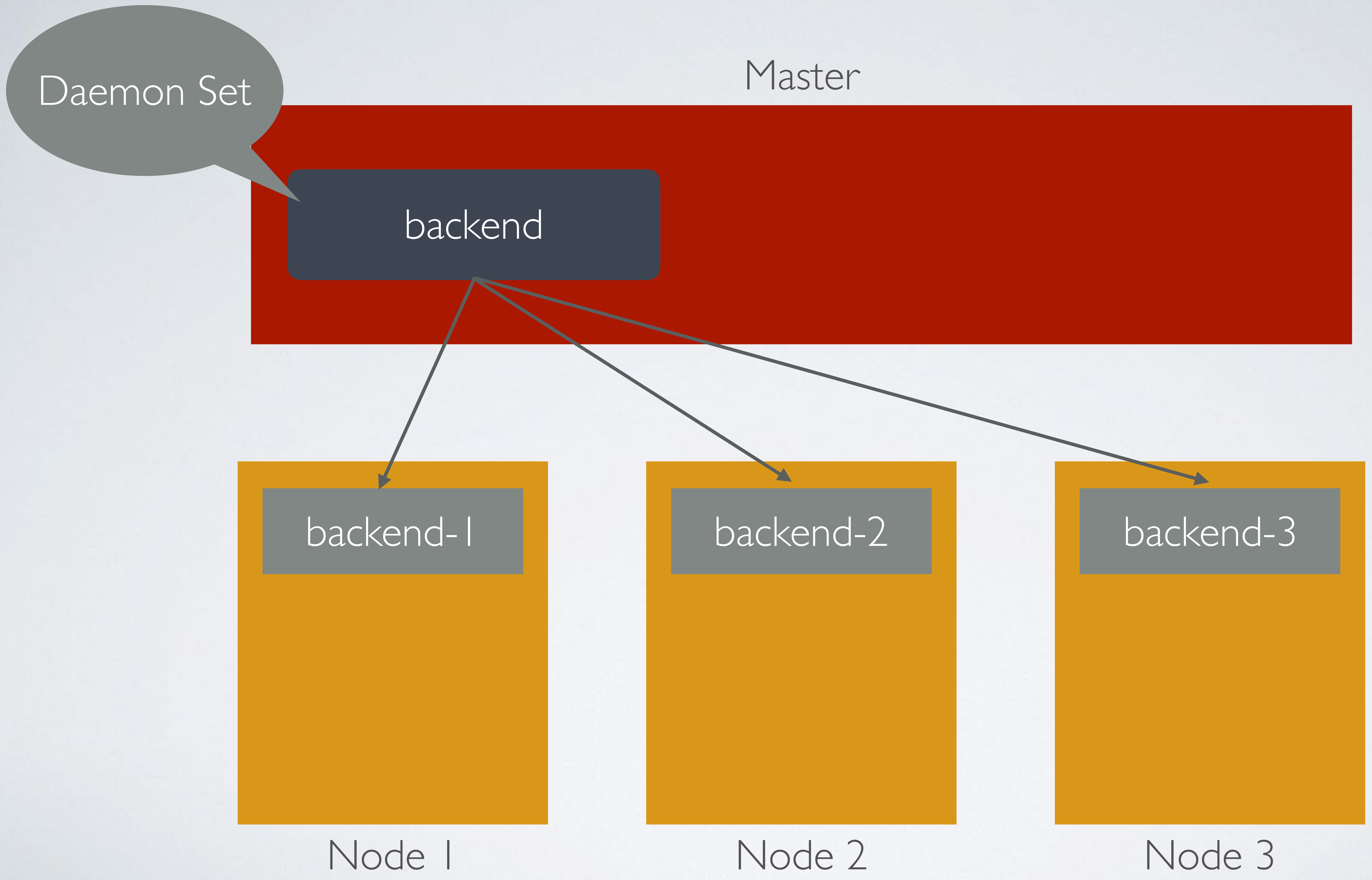
```
$ kubectl rollout undo deployment/echoserver
deployment "echoserver" rolled back

$ kubectl rollout history deployment/echoserver
deployments "echoserver"
REVISION        CHANGE-CAUSE
1               kubectl create --filename=07-deployment.yaml --record=true
3               kubectl set image deployment/echoserver echoserver=gcr.io/google-containers/echoserver:1.3
4               kubectl set image deployment/echoserver echoserver=gcr.io/google-containers/echoserver:1.2
```

```
$ kubectl rollout undo deployment/echoserver --to-revision=1
deployment "echoserver" rolled back

$ kubectl rollout history deployment/echoserver
deployments "echoserver"
REVISION        CHANGE-CAUSE
3               kubectl set image deployment/echoserver echoserver=gcr.io/google-containers/echoserver:1.3
4               kubectl set image deployment/echoserver echoserver=gcr.io/google-containers/echoserver:1.2
5               kubectl create --filename=07-deployment.yaml --record=true
```

```
$ kubectl scale deployment/echoserver --replicas 6
deployment "echoserver" scaled

$ kubectl get deployment/echoserver
NAME         DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
echoserver   6         6         6            6           11m
```

Daemon Sets (ds)

ensures that all (or some) nodes run a copy of a pod


Daemon Set
Master
backend
backend-1
backend-2
backend-3
Node 1
Node 2
Node 3

```yaml
kind: DaemonSet
apiVersion: extensions/v1beta1
metadata:
  name: echoserver
spec:
  template:
    metadata:
      labels:
        app: echoserver
    spec:
      containers:
      - name: echoserver
        image: gcr.io/google-containers/echoserver:1.6
        ports:
        - containerPort: 8080
  updateStrategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
```

Strategy

Default

- OnDelete — new DaemonSet pods will only be created when you manually delete old DaemonSet pods
- RollingUpdate

Resource Quotas (quota)

limit aggregate resource consumption

```yaml
kind: Deployment
apiVersion: apps/v1beta1
metadata:
  name: echoserver
spec:
  replicas: 3
  revisionHistoryLimit: 2
  template:
    metadata:
      labels:
        app: echoserver
    spec:
      containers:
      - name: echoserver
        image: gcr.io/google-containers/echoserver:1.6
        ports:
        - containerPort: 8080
        resources:
          requests:
            cpu: 200m
            memory: 300Mi
          limits:
            cpu: 1
            memory: 1Gi
```

Health Check

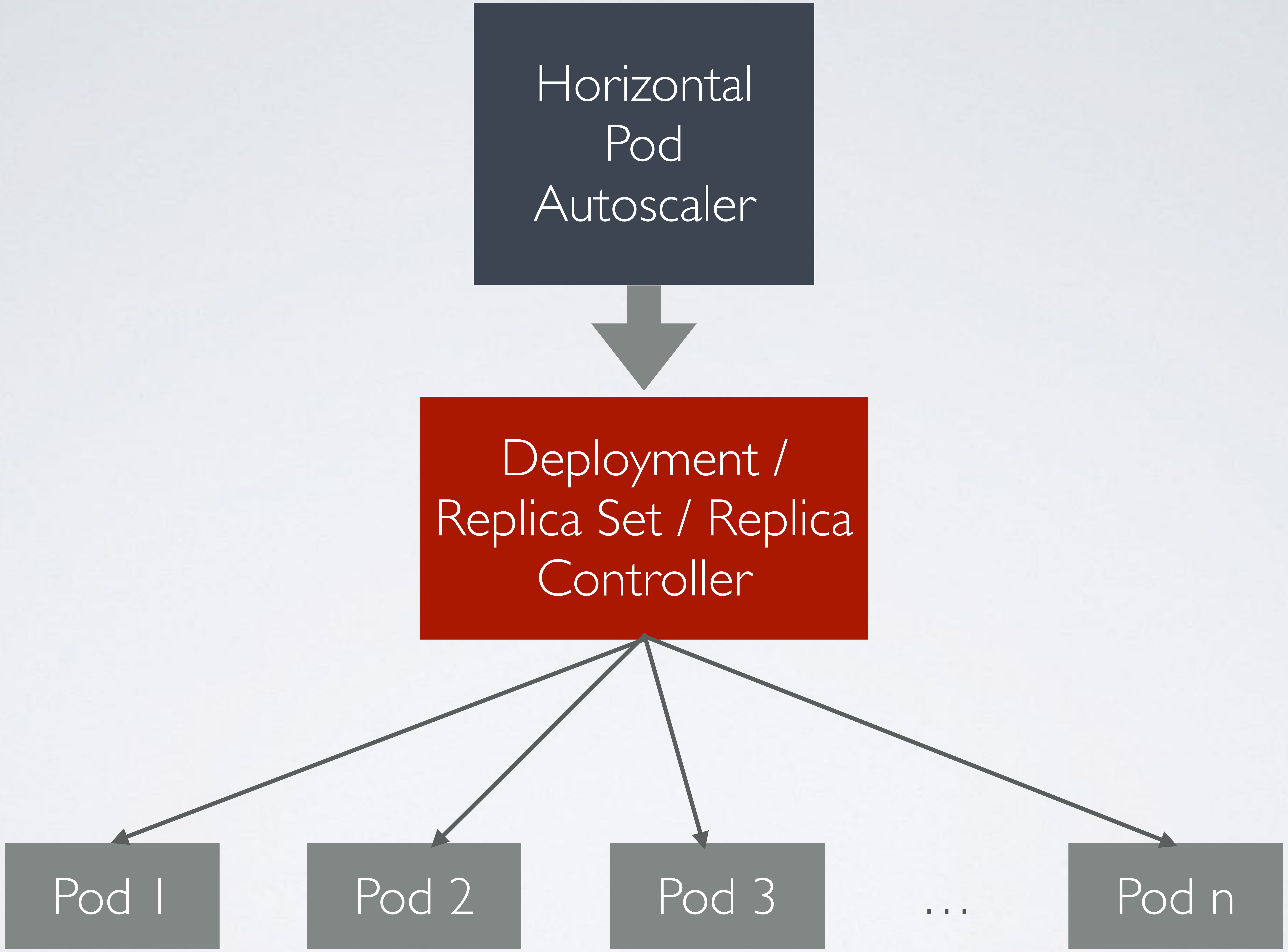
Health Check

- Liveness Probes — know when to restart a Container
- Readiness Probes — don't send requests until application started

```yaml
kind: Deployment
apiVersion: app/v1beta1
metadata:
  name: default-http-backend
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: default-http-backend
    spec:
      containers:
      - name: default-http-backend
        image: gcr.io/google-containers/defaultbackend:1.3
        ports:
        - containerPort: 8080
        readinessProbe:
          httpGet:
            path: /healthz
            port: 8080
            scheme: HTTP
          initialDelaySeconds: 30
          timeoutSeconds: 5
          periodSeconds: 10
          successThreshold: 1
          failureThreshold: 3
        livenessProbe:
          httpGet:
            path: /healthz
            port: 8080
            scheme: HTTP
          initialDelaySeconds: 30
          timeoutSeconds: 5
          periodSeconds: 10
          successThreshold: 1
          failureThreshold: 3
```

Horizontal Pod Autoscaler (hpa)

automatically scales the number of pods in
a replication controller, deployment or replica set


Horizontal Pod Autoscaler
Deployment / Replica Set / Replica Controller
Pod 1
Pod 2
Pod 3
...
Pod n

```yaml
kind: Deployment
apiVersion: apps/v1beta1
metadata:
  name: hpa-example
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: hpa-example
    spec:
      containers:
      - name: hpa-example
        image: gcr.io/google-containers/hpa-example
        ports:
        - containerPort: 80
        resources:
          requests:
            cpu: 100m
```

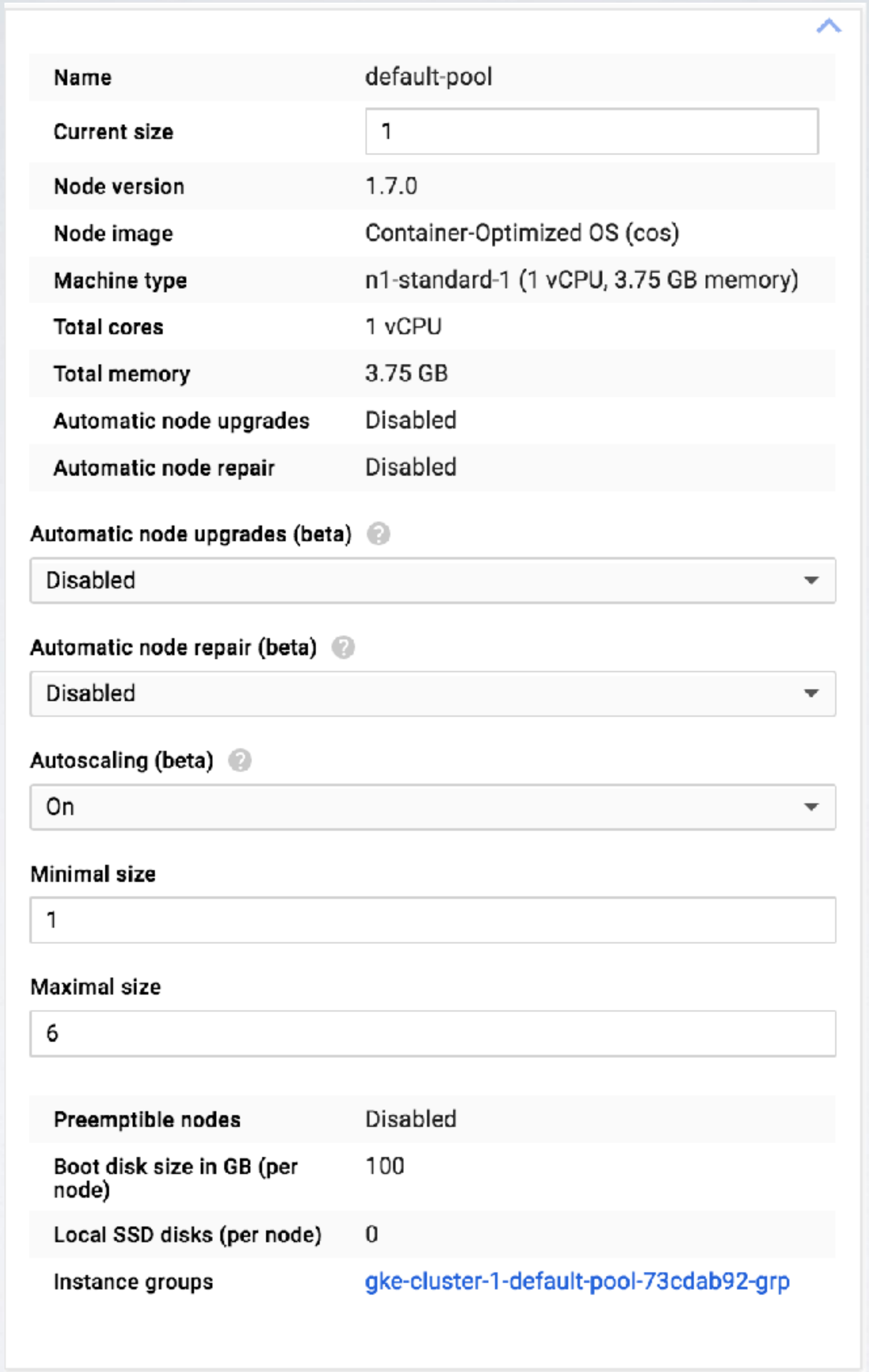
```yaml
apiVersion: v1
kind: Service
metadata:
  name: hpa-example
spec:
  selector:
    app: hpa-example
  ports:
  - port: 80
```

```yaml
kind: HorizontalPodAutoscaler
apiVersion: autoscaling/v1
metadata:
  name: hpa-example
spec:
  scaleTargetRef:
    apiVersion: apps/v1beta1
    kind: Deployment
    name: hpa-example
  minReplicas: 1
  maxReplicas: 6
  targetCPUUtilizationPercentage: 50
```

```
$ kubectl get hpa --watch
NAME          REFERENCE                TARGETS     MINPODS   MAXPODS   REPLICAS   AGE
hpa-example   Deployment/hpa-example     0% / 50%   1         6         1         12m
hpa-example   Deployment/hpa-example   522% / 50%   1         6         1         12m
hpa-example   Deployment/hpa-example   522% / 50%   1         6         1         12m
hpa-example   Deployment/hpa-example   941% / 50%   1         6         1         13m
hpa-example   Deployment/hpa-example   941% / 50%   1         6         4         13m
hpa-example   Deployment/hpa-example   362% / 50%   1         6         4         14m
hpa-example   Deployment/hpa-example   362% / 50%   1         6         4         14m
hpa-example   Deployment/hpa-example    12% / 50%   1         6         4         15m
hpa-example   Deployment/hpa-example    12% / 50%   1         6         4         15m
hpa-example   Deployment/hpa-example     0% / 50%   1         6         4         16m
```

Auto-scale Node on Container Engine

```
$ gcloud alpha container clusters update cluster-1 \
  --enable-autoscaling \
  --min-nodes=2 \
  --max-nodes=6 \
  --zone=asia-southeast1-b \
  --node-pool=default-pool
```



Name	default-pool
Current size	1
Node version	1.7.0
Node image	Container-Optimized OS (cos)
Machine type	n1-standard-1 (1 vCPU, 3.75 GB memory)
Total cores	1 vCPU
Total memory	3.75 GB
Automatic node upgrades	Disabled
Automatic node repair	Disabled

Automatic node upgrades (beta): Disabled

Automatic node repair (beta): Disabled

Autoscaling (beta): On

Minimal size: 1

Maximal size: 6

Preemptible nodes	Disabled
Boot disk size in GB (per node)	100
Local SSD disks (per node)	0
Instance groups	gke-cluster-1-default-pool-73cdab92-grp

Q&A

DAY 2

Persistent Disk (pd)

Create Persistent Disk (pd) on GCP

```
$ gcloud compute disks create --size=20GB --zone=asia-southeast1-b --project=acoshift-k8s mysql-disk
Created [https://www.googleapis.com/compute/v1/projects/acoshift-k8s/zones/asia-southeast1-b/disks/mysql-disk].
NAME        ZONE               SIZE_GB  TYPE         STATUS
mysql-disk  asia-southeast1-b  20       pd-standard  READY
```

```yaml
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: mysql
spec:
  replicas: 1
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
      - name: mysql
        env:
        - name: MYSQL_ROOT_PASSWORD
          value: mysqlpassword1234
        image: mysql:5.6.36
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 3306
        volumeMounts:
        - mountPath: /var/lib/mysql
          name: mysql-disk
      volumes:
      - name: mysql-disk
        gcePersistentDisk:
          pdName: mysql-disk
          fsType: ext4
```

```
$ kubectl create -f 14-pv.yaml
deployment "mysql" created

$ kubectl get po
NAME                     READY     STATUS    RESTARTS   AGE
mysql-1398320157-mgf6c   1/1       Running   0          3m

$ kubectl port-forward mysql-1398320157-mgf6c 3306:3306
$ mysql -u root -p -h 127.0.0.1

mysql> create database db1;
Query OK, 1 row affected (0.05 sec)

mysql> use db1;
Database changed

mysql> create table users (
    -> id int auto_increment,
    -> name varchar(255) not null,
    -> created_at timestamp not null default now(),
    -> primary key (id)
    -> );
Query OK, 0 rows affected (0.08 sec)

mysql> insert into users (name) values ('acoshift'), ('user1'), ('user2');
Query OK, 3 rows affected (0.08 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select * from users;
+----+----------+---------------------+
| id | name     | created_at          |
+----+----------+---------------------+
|  1 | acoshift | 2017-07-15 14:46:04 |
|  2 | user1    | 2017-07-15 14:46:04 |
|  3 | user2    | 2017-07-15 14:46:04 |
+----+----------+---------------------+
3 rows in set (0.03 sec)

mysql> exit
Bye
```

```
$ kubectl get po
NAME                     READY     STATUS    RESTARTS   AGE
mysql-1398320157-mgf6c   1/1       Running   0          19m

$ kubectl delete po/mysql-1398320157-mgf6c
pod "mysql-1398320157-mgf6c" deleted

$ kubectl get po
NAME                     READY     STATUS              RESTARTS   AGE
mysql-1398320157-d0scs   0/1       ContainerCreating   0          30s
```

Pods

	Name	Status	Restarts	Age	CPU (cores)	Memory (bytes)
!	mysql-1398320157-d0scs	Waiting: Containe...	0	41 seconds	-	-

```
AttachVolume.Attach failed for volume "mysql-disk" : googleapi: Error 400: The disk resource 'projects/acoshift-k8s/zones/asia-southeast1-b/disks/mysql-disk' is already being used by 'projects/acoshift-k8s/zones/asia-southeast1-b/instances/gke-cluster-1-default-pool-73cdab92-hhk2'
```

```
$ kubectl get po
NAME                         READY     STATUS    RESTARTS   AGE
mysql-1398320157-d0scs   1/1       Running   0          6m

$ kubectl port-forward mysql-1398320157-d0scs 3306:3306
Forwarding from 127.0.0.1:3306 -> 3306
Forwarding from [::1]:3306 -> 3306

$ mysql -u root -p -h 127.0.0.1

mysql> use db1;
Database changed

mysql> select * from users;
+----+-----------+---------------------+
| id | name      | created_at          |
+----+-----------+---------------------+
|  1 | acoshift | 2017-07-15 14:46:04 |
|  2 | user1     | 2017-07-15 14:46:04 |
|  3 | user2     | 2017-07-15 14:46:04 |
+----+-----------+---------------------+
3 rows in set (0.04 sec)

mysql> exit
Bye
```

```
$ kubectl delete -f 14-pd.yaml
deployment "mysql" deleted

$ gcloud compute disks delete --zone=asia-southeast1-b --project=acoshift-k8s mysql-disk
The following disks will be deleted:
 - [mysql-disk] in [asia-southeast1-b]

Do you want to continue (Y/n)?  Y

Deleted [https://www.googleapis.com/compute/v1/projects/acoshift-k8s/zones/asia-southeast1-b/disks/mysql-disk].
```

Persistent Volumes (pv)

a piece of storage in the cluster that has been provisioned by an administrator

Persistent Volume Claims (pvc)

a request for storage by a user

StorageClasses

a way for administrators to describe the “classes” of storage they offer

Provisioning

- Static
- Dynamic

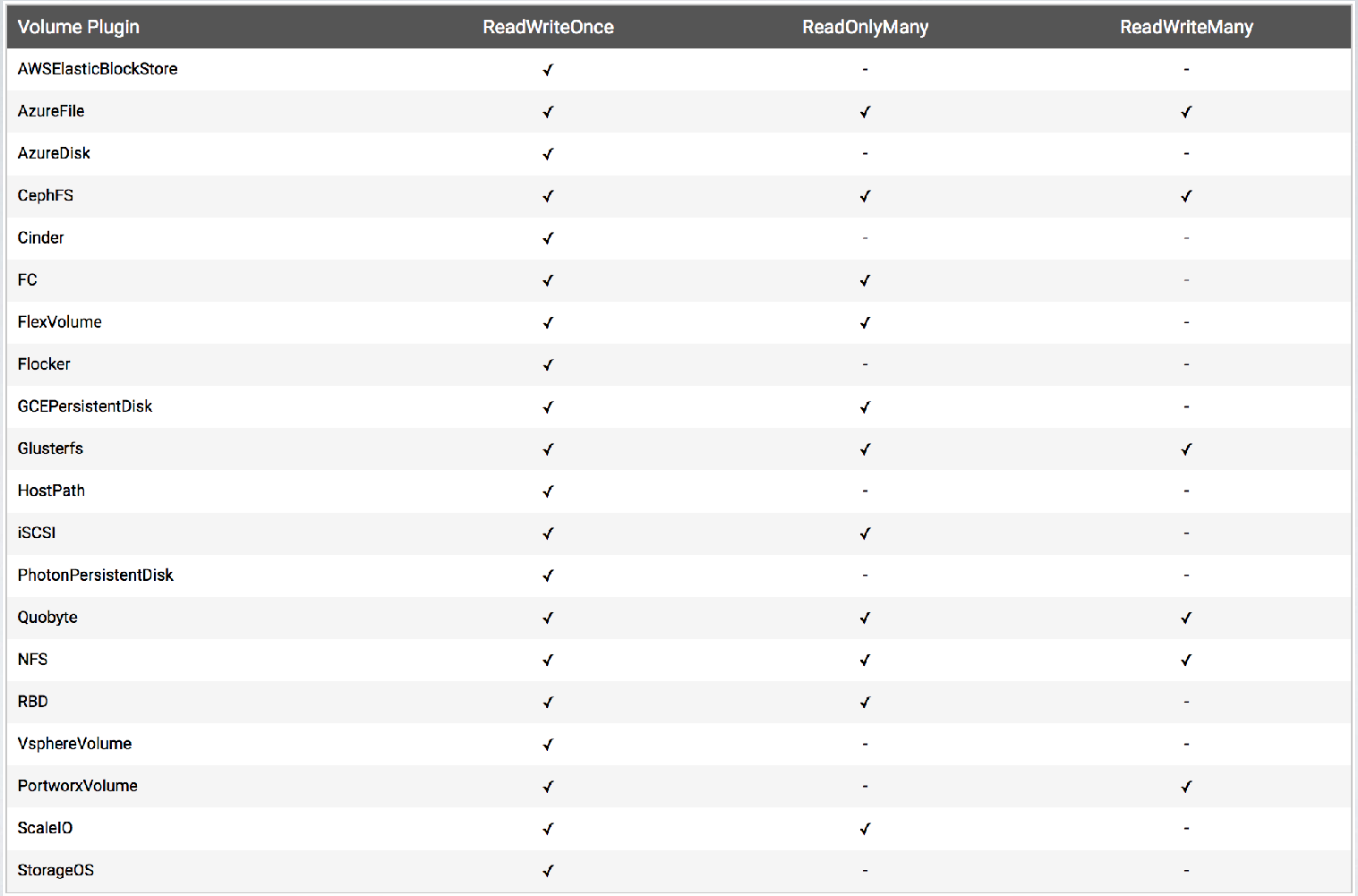
```
$ kubectl get storageclass
NAME                 TYPE
standard (default)   kubernetes.io/gce-pd

$ kubectl describe storageclass standard
Name:           standard
IsDefaultClass: Yes
Annotations:    storageclass.beta.kubernetes.io/is-default-class=true
Provisioner:    kubernetes.io/gce-pd
Parameters:     type=pd-standard
Events:         <none>
```

```yaml
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: ssd
provisioner: kubernetes.io/gce-pd
parameters:
  # type: pd-standard
  type: pd-ssd
  zone: asia-southeast1-b
```

```
$ kubectl create -f 15-storageclass.yaml
storageclass "ssd" created

$ kubectl get storageclass
NAME                    TYPE
ssd                     kubernetes.io/gce-pd
standard (default)      kubernetes.io/gce-pd
```

Access Modes

- ReadWriteOnce — the volume can be mounted as read-write by a single node
- ReadOnlyMany — the volume can be mounted read-only by many nodes
- ReadWriteMany — the volume can be mounted as read-write by many nodes



Volume Plugin	ReadWriteOnce	ReadOnlyMany	ReadWriteMany
AWSElasticBlockStore	✓	-	-
AzureFile	✓	✓	✓
AzureDisk	✓	-	-
CephFS	✓	✓	✓
Cinder	✓	-	-
FC	✓	✓	-
FlexVolume	✓	✓	-
Flocker	✓	-	-
GCEPersistentDisk	✓	✓	-
Glusterfs	✓	✓	✓
HostPath	✓	-	-
iSCSI	✓	✓	-
PhotonPersistentDisk	✓	-	-
Quobyte	✓	✓	✓
NFS	✓	✓	✓
RBD	✓	✓	-
VsphereVolume	✓	-	-
PortworxVolume	✓	-	✓
ScaleIO	✓	✓	-
StorageOS	✓	-	-

https://kubernetes.io/docs/concepts/storage/persistent-volumes/

Reclaim Policy

- Retain
- Recycle
- Delete

Default for static provisioning

Default for dynamic provisioning

```yaml
kind: PersistentVolume
apiVersion: v1
metadata:
  name: disk-1
  annotations:
    volume.beta.kubernetes.io/mount-options: discard
spec:
  storageClassName: standard
  capacity:
    storage: 10Gi
  accessModes:
  - ReadWriteOnce
  gcePersistentDisk:
    fsType: ext4
    pdName: disk-1
```

```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: mysql-pvc
spec:
  storageClassName: standard
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
```

```
$ gcloud compute disks create --size=10GB --zone=asia-southeast1-b --project=acoshift-k8s disk-1
Created [https://www.googleapis.com/compute/v1/projects/acoshift-k8s/zones/asia-southeast1-b/disks/disk-1].
NAME    ZONE               SIZE_GB  TYPE         STATUS
disk-1  asia-southeast1-b  10       pd-standard  READY

$ kubectl create -f 16-pv.yaml
persistentvolume "disk-1" created

$ kubectl get pv
NAME      CAPACITY   ACCESSMODES   RECLAIMPOLICY   STATUS      CLAIM     STORAGECLASS   REASON    AGE
disk-1    10Gi       RWO           Retain          Available                                      12s

$ kubectl create -f 16-pvc.yaml
persistentvolumeclaim "mysql-pvc" created

$ kubectl get pv
NAME      CAPACITY   ACCESSMODES   RECLAIMPOLICY   STATUS    CLAIM               STORAGECLASS   REASON    AGE
disk-1    10Gi       RWO           Retain          Bound     default/mysql-pvc   standard                 1m

$ kubectl get pvc
NAME        STATUS    VOLUME    CAPACITY   ACCESSMODES   STORAGECLASS   AGE
mysql-pvc   Bound     disk-1    10Gi       RWO           standard       34s
```

```yaml
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: mysql
spec:
  replicas: 1
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
      - name: mysql
        env:
        - name: MYSQL_ROOT_PASSWORD
          value: mysqlpassword1234
        image: mysql:5.6.36
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 3306
        volumeMounts:
        - mountPath: /var/lib/mysql
          name: mysql-disk
      volumes:
      - name: mysql-disk
        persistentVolumeClaim:
          claimName: mysql-pvc
```

```
$ kubectl create -f 16-mysql.yaml
deployment "mysql" created

$ kubectl get po
NAME                   READY     STATUS    RESTARTS   AGE
mysql-68058648-d3m8l   1/1       Running   0          48s

$ kubectl delete po/mysql-68058648-d3m8l
pod "mysql-68058648-d3m8l" deleted

$ kubectl get po
NAME                   READY     STATUS    RESTARTS   AGE
mysql-68058648-rsqk1   1/1       Running   0          4s
```

```
$ kubectl delete deploy/mysql pvc/mysql-pvc pv/disk-1
deployment "mysql" deleted
persistentvolumeclaim "mysql-pvc" deleted
persistentvolume "disk-1" deleted

$ gcloud compute disks delete --zone=asia-southeast1-b --project=acoshift-k8s disk-1
The following disks will be deleted:
 - [disk-1] in [asia-southeast1-b]

Do you want to continue (Y/n)?  Y

Deleted [https://www.googleapis.com/compute/v1/projects/acoshift-k8s/zones/asia-
southeast1-b/disks/disk-1].
```

Stateful Sets

provides guarantees about the ordering of deployment and scaling

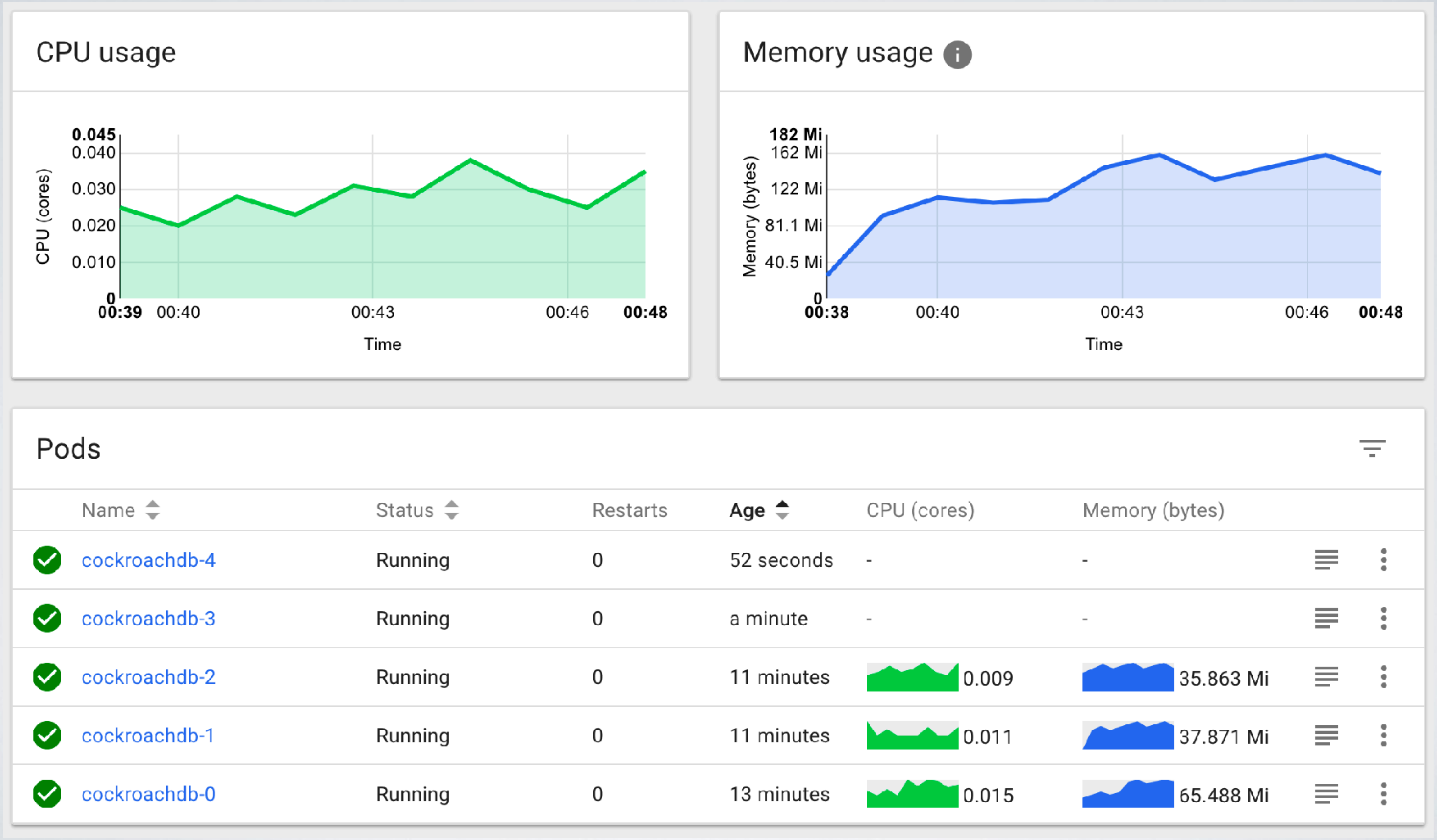
Stateful Sets

- Stable, unique network identifier
- Stable, persistent storage
- Ordered, graceful deployment and scaling
- Ordered, graceful deletion and termination
- Ordered, automated rolling updates

```
$ kubectl create -f https://raw.githubusercontent.com/cockroachdb/cockroach/master/cloud/kubernetes/
cockroachdb-statefulset.yaml
service "cockroachdb-public" created
service "cockroachdb" created
poddisruptionbudget "cockroachdb-budget" created
statefulset "cockroachdb" created

$ kubectl get po
NAME                READY     STATUS    RESTARTS   AGE
cockroachdb-0       1/1       Running   0          10m
cockroachdb-1       1/1       Running   0          9m
cockroachdb-2       1/1       Running   0          8m

$ kubectl port-forward cockroachdb-0 8080
```

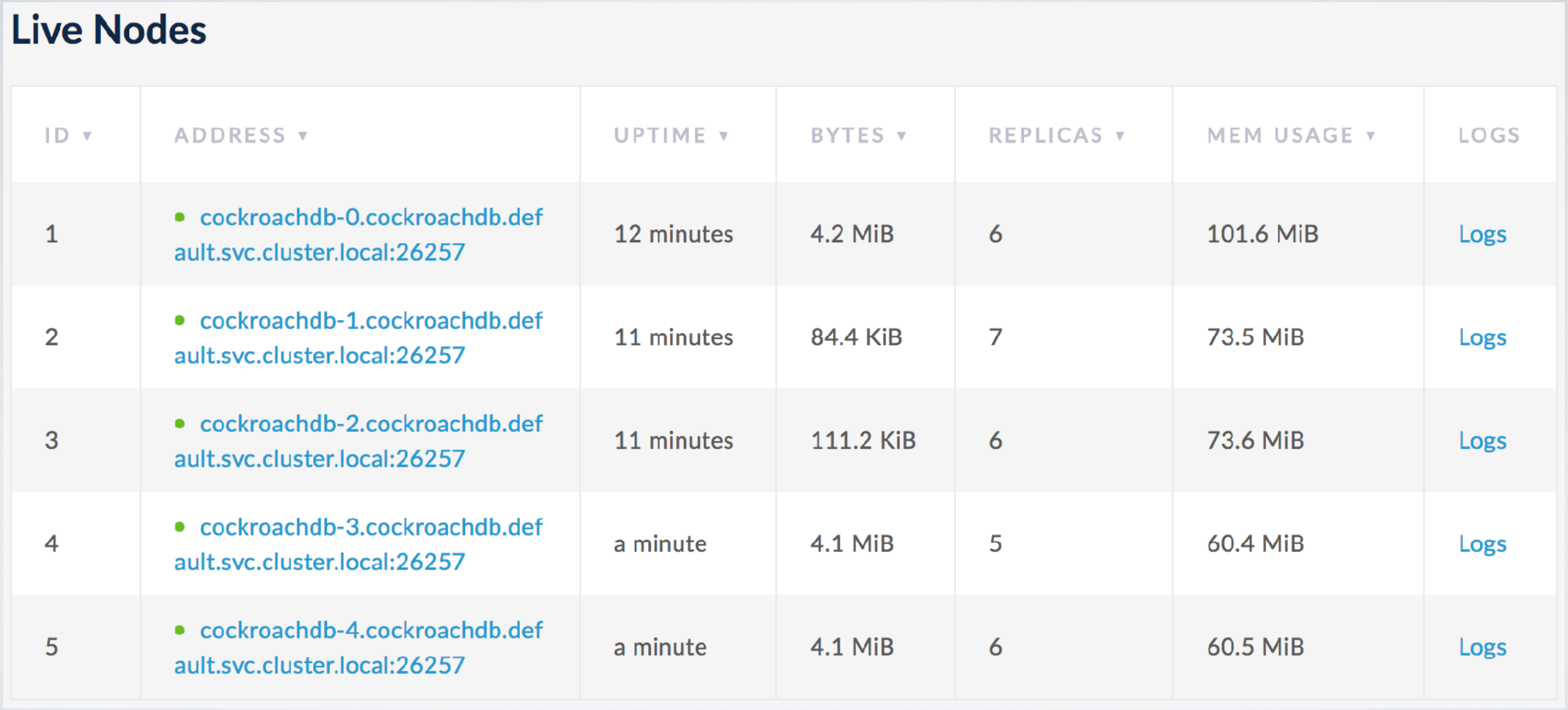
Live Nodes

ID ▾	ADDRESS ▾	UPTIME ▾	BYTES ▾	REPLICAS ▾	MEM USAGE ▾	LOGS
1	● cockroachdb-0.cockroachdb.default.svc.cluster.local:26257	10 minutes	3.2 MiB	10	95.6 MiB	Logs
2	● cockroachdb-1.cockroachdb.default.svc.cluster.local:26257	9 minutes	3.3 MiB	10	71.5 MiB	Logs
3	● cockroachdb-2.cockroachdb.default.svc.cluster.local:26257	9 minutes	3.3 MiB	10	70.4 MiB	Logs



Pods

	Name	Status	Restarts	Age	CPU (cores)	Memory (bytes)
✓	cockroachdb-4	Running	0	52 seconds	-	-
✓	cockroachdb-3	Running	0	a minute	-	-
✓	cockroachdb-2	Running	0	11 minutes	0.009	35.863 Mi
✓	cockroachdb-1	Running	0	11 minutes	0.011	37.871 Mi
✓	cockroachdb-0	Running	0	13 minutes	0.015	65.488 Mi



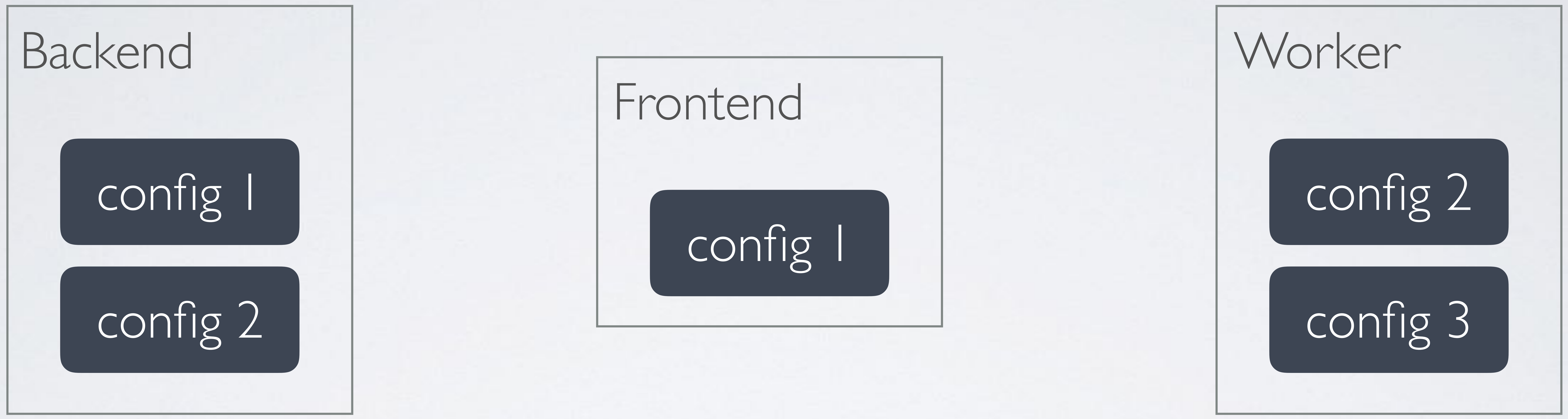
Live Nodes

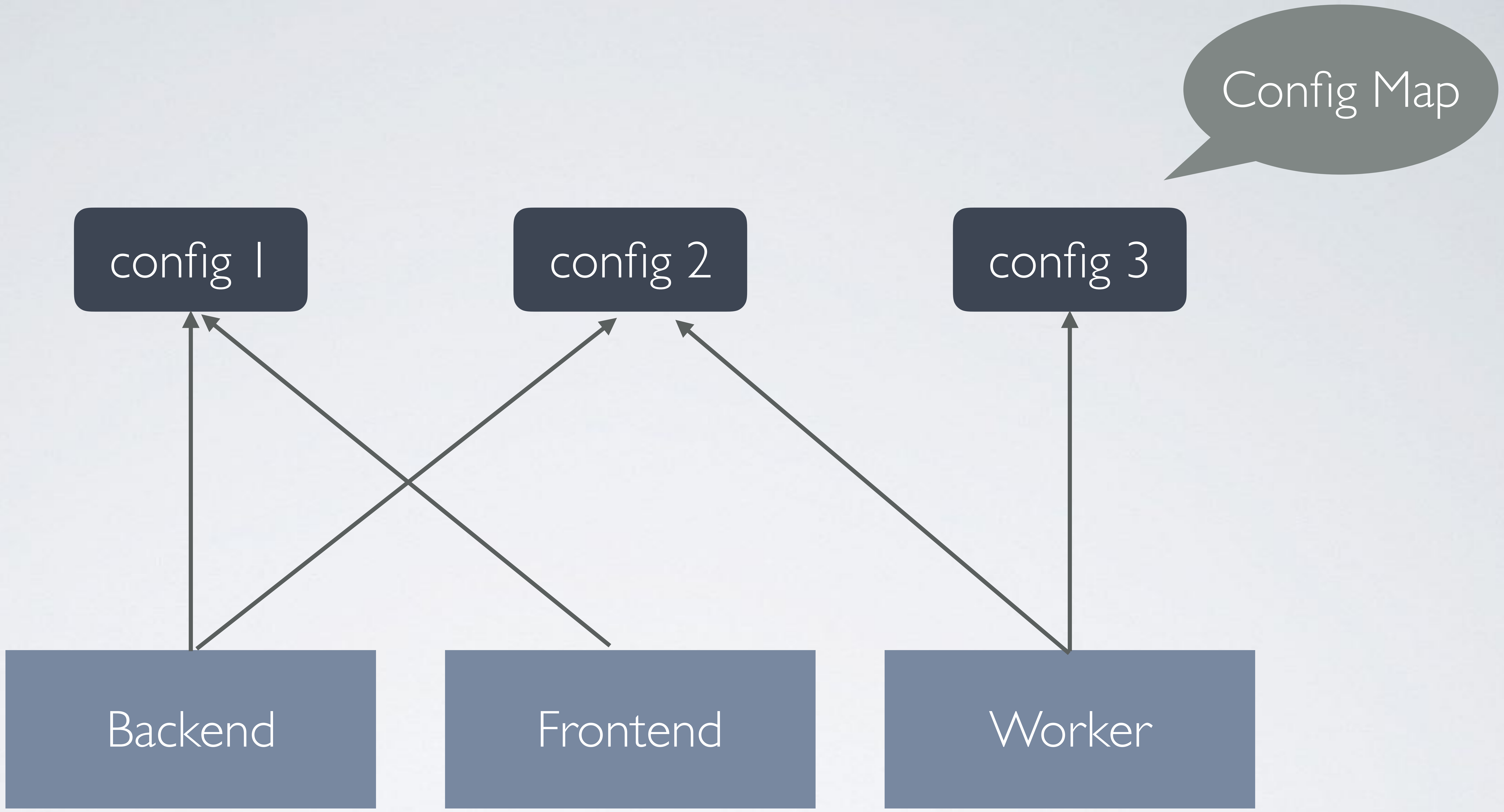
ID ▾	ADDRESS ▾	UPTIME ▾	BYTES ▾	REPLICAS ▾	MEM USAGE ▾	LOGS
1	● cockroachdb-0.cockroachdb.default.svc.cluster.local:26257	12 minutes	4.2 MiB	6	101.6 MiB	Logs
2	● cockroachdb-1.cockroachdb.default.svc.cluster.local:26257	11 minutes	84.4 KiB	7	73.5 MiB	Logs
3	● cockroachdb-2.cockroachdb.default.svc.cluster.local:26257	11 minutes	111.2 KiB	6	73.6 MiB	Logs
4	● cockroachdb-3.cockroachdb.default.svc.cluster.local:26257	a minute	4.1 MiB	5	60.4 MiB	Logs
5	● cockroachdb-4.cockroachdb.default.svc.cluster.local:26257	a minute	4.1 MiB	6	60.5 MiB	Logs

```
$ kubectl run -it --rm cockroach-client --image=cockroachdb/cockroach --restart=Never --command -- ./cockroach sql --host
cockroachdb-public --insecure

root@cockroachdb-public:26257/> create database db1;
CREATE DATABASE

root@cockroachdb-public:26257/> set database = db1;
SET

root@cockroachdb-public:26257/db1> create table users (
                                -> id serial,
                                -> name string not null default '',
                                -> created_at timestamp not null default now(),
                                -> primary key (id)
                                -> );
CREATE TABLE

root@cockroachdb-public:26257/db1> insert into users (name) values ('acoshift'), ('user1'), ('user2');
INSERT 3

root@cockroachdb-public:26257/db1> select * from users;
+--------------------+----------+----------------------------------+
|         id         |   name   |            created_at            |
+--------------------+----------+----------------------------------+
| 262376372306051076 | acoshift | 2017-07-15 17:55:14.366042+00:00 |
| 262376372306247684 | user1    | 2017-07-15 17:55:14.366042+00:00 |
| 262376372306345988 | user2    | 2017-07-15 17:55:14.366042+00:00 |
+--------------------+----------+----------------------------------+
(3 rows)
```

Config Maps (cm)

decouple configuration artifacts from image content
to keep containerized applications portable


Backend
config 1
config 2
Frontend
config 1
Worker
config 2
config 3


Config Map
config 1
config 2
config 3
Backend
Frontend
Worker

```yaml
kind: ConfigMap
apiVersion: v1
metadata:
  name: redis-config
data:
  redis.conf: |
    databases 1
    save ""
    appendonly no
    maxmemory 2mb
    maxmemory-policy allkeys-lru
---
kind: Service
apiVersion: v1
metadata:
  name: redis
spec:
  selector:
    app: redis
  ports:
  - port: 6379
```

```yaml
kind: StatefulSet
apiVersion: apps/v1beta1
metadata:
  name: redis
spec:
  serviceName: redis
  replicas: 1
  template:
    metadata:
      labels:
        app: redis
    spec:
      containers:
      - name: redis
        image: redis:3.2.9
        ports:
        - containerPort: 6379
        volumeMounts:
        - mountPath: /usr/local/etc/redis
          name: config
        command:
        - redis-server
        - /usr/local/etc/redis/redis.conf
      volumes:
      - name: config
        configMap:
          name: redis-config
          items:
          - key: redis.conf
            path: redis.conf
```

```
$ kubectl create -f 21-cm.yaml
configmap "redis-config" created
service "redis" created
statefulset "redis" created

$ kubectl run -it --rm redis-client --image=redis --restart=Never --command -- bash
root@redis-client:/data# redis-cli -h redis

redis:6379> config get save
1) "save"
2) ""
```

Secrets

hold sensitive information

```
$ echo -n "testuser" | base64
dGVzdHVzZXI=
$ echo -n "testpassword" | base64
dGVzdHBhc3N3b3Jk
```

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-secret
data:
  username: dGVzdHVzZXI=
  password: dGVzdHBhc3N3b3Jk
```

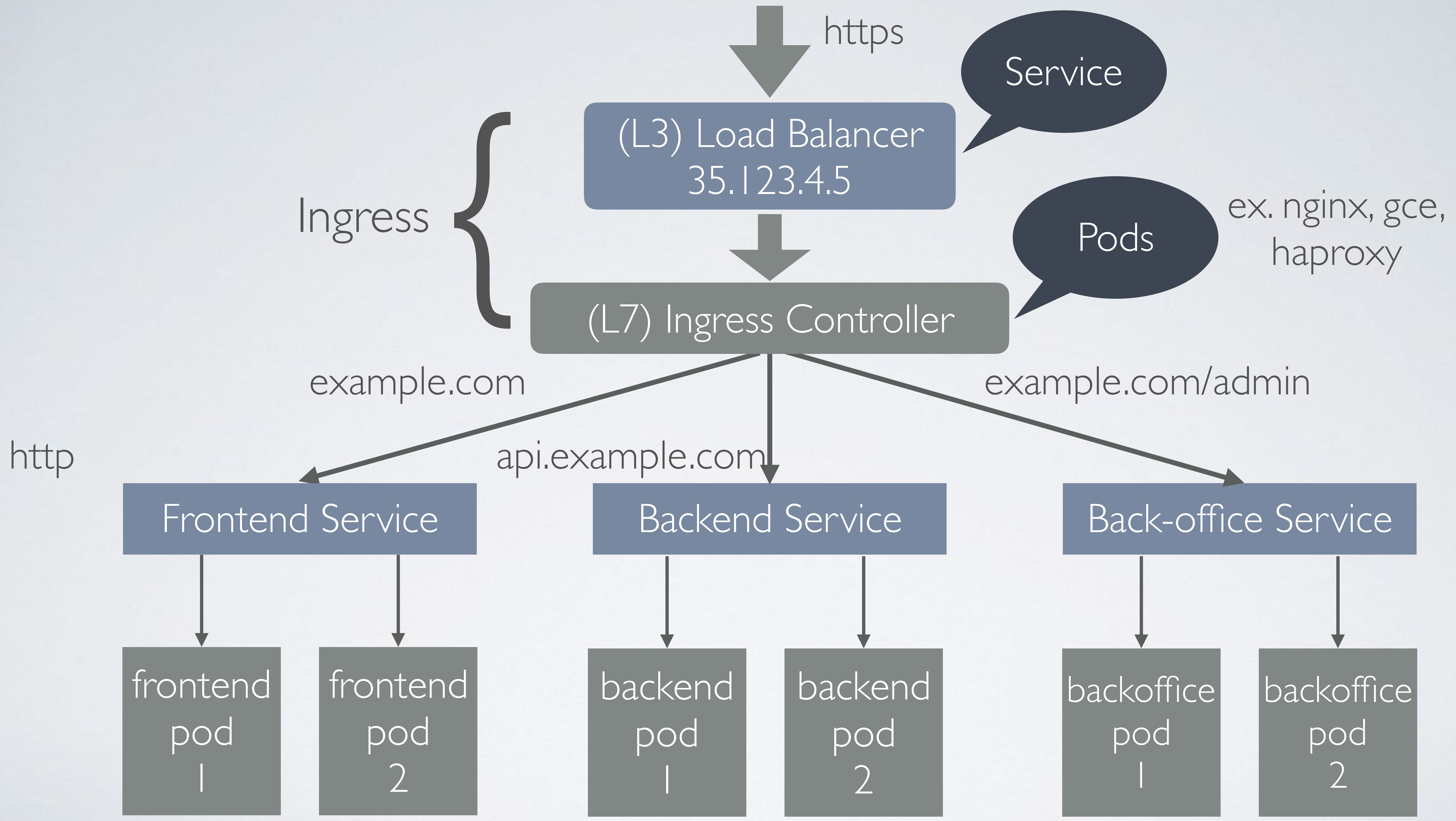
```yaml
kind: Deployment
apiVersion: apps/v1beta1
metadata:
  name: nginx
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: gcr.io/google-containers/nginx-slim:0.8
        ports:
        - containerPort: 80
        volumeMounts:
        - name: www
          mountPath: /usr/share/nginx/html
      volumes:
      - name: www
        secret:
          secretName: backend-secret
          defaultMode: 0666
```

```
$ kubectl create -f 22-secret.yaml

$ kubectl get po
NAME                         READY     STATUS    RESTARTS   AGE
nginx-1183500012-cjcpq       1/1       Running   0          15s

$ kubectl port-forward nginx-1183500012-cjcpq 8080:80

$ curl localhost:8080/username
testuser

$ curl localhost:8080/password
testpassword
```

```
$ openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout tls.key -out tls.crt -subj "/CN=echoserver.acoshift.com"
Generating a 2048 bit RSA private key
.....................................+++
...+++
writing new private key to 'tls.key'
-----

$ kubectl create secret tls echoserver-acoshift-com-tls --key=tls.key --cert=tls.crt
secret "echoserver-acoshift-com-tls" created
```

Ingresses (ing)

a collection of rules that allow inbound connections to reach the cluster services


https
Service
(L3) Load Balancer
35.123.4.5
Ingress
Pods
ex. nginx, gce, haproxy
(L7) Ingress Controller
example.com
example.com/admin
http
api.example.com
Frontend Service
Backend Service
Back-office Service
frontend pod 1
frontend pod 2
backend pod 1
backend pod 2
backoffice pod 1
backoffice pod 2

```yaml
kind: Deployment
apiVersion: apps/v1beta1
metadata:
  name: echoserver
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: echoserver
    spec:
      containers:
      - name: echoserver
        image: gcr.io/google-containers/echoserver:1.6
        ports:
        - containerPort: 8080
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: echoserver
spec:
  ports:
  - port: 80
    targetPort: 8080
  selector:
    app: echoserver
```

```yaml
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: default-http-backend
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: default-http-backend
    spec:
      containers:
      - name: default-http-backend
        image: gcr.io/google-containers/defaultbackend:1.0
        livenessProbe:
          httpGet:
            path: /healthz
            port: 8080
            scheme: HTTP
          initialDelaySeconds: 30
          timeoutSeconds: 5
        ports:
        - containerPort: 8080
        resources:
          limits:
            cpu: 10m
            memory: 20Mi
          requests:
            cpu: 10m
            memory: 20Mi
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: default-http-backend
spec:
  ports:
  - port: 80
    targetPort: 8080
  selector:
    app: default-http-backend
```

Nginx Ingress Controller

```yaml
kind: ConfigMap
apiVersion: v1
metadata:
  name: nginx-config
data:
  client-max-body-size: 20m
  hsts: "false"
  keep-alive: "30"
  proxy-body-size: 20m
  server-tokens: "false"
  use-gzip: "true"
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-ingress
spec:
  type: LoadBalancer
  selector:
    app: nginx-ingress
  ports:
  - name: http
    port: 80
  - name: https
    port: 443
```

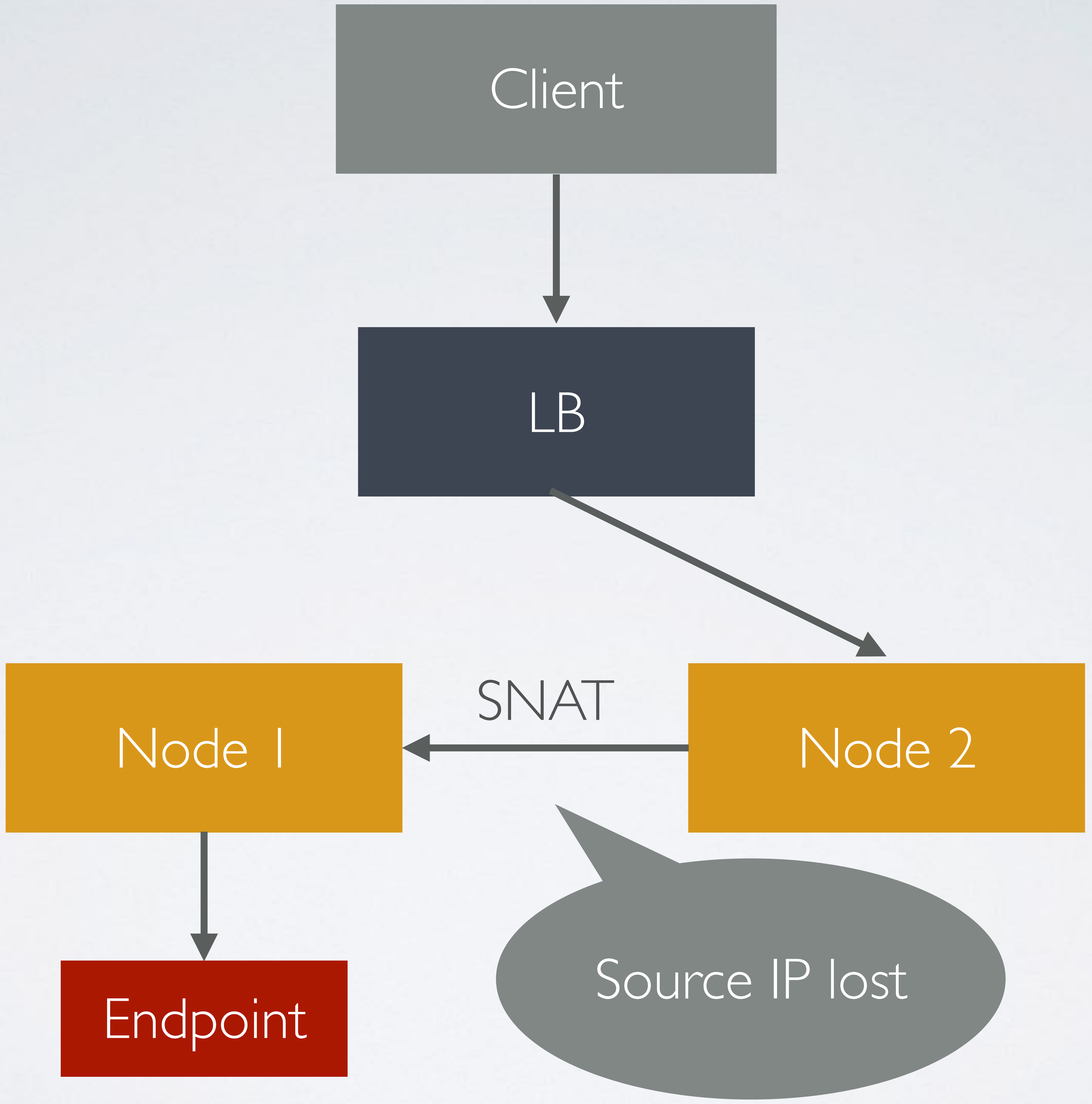
```yaml
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: nginx-ingress
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx-ingress
    spec:
      containers:
      - name: nginx-ingress-controller
        image: gcr.io/google-containers/nginx-ingress-controller:0.9.0-beta.10
        imagePullPolicy: Always
        ports:
        - containerPort: 80
        - containerPort: 443
        env:
        - name: POD_NAME
          valueFrom:
            fieldRef:
              apiVersion: v1
              fieldPath: metadata.name
        - name: POD_NAMESPACE
          valueFrom:
            fieldRef:
              apiVersion: v1
              fieldPath: metadata.namespace
        args:
        - /nginx-ingress-controller
        - --default-backend-service=$(POD_NAMESPACE)/default-http-backend
        - --configmap=$(POD_NAMESPACE)/nginx-config
        - --publish-service=$(POD_NAMESPACE)/nginx-ingress
```

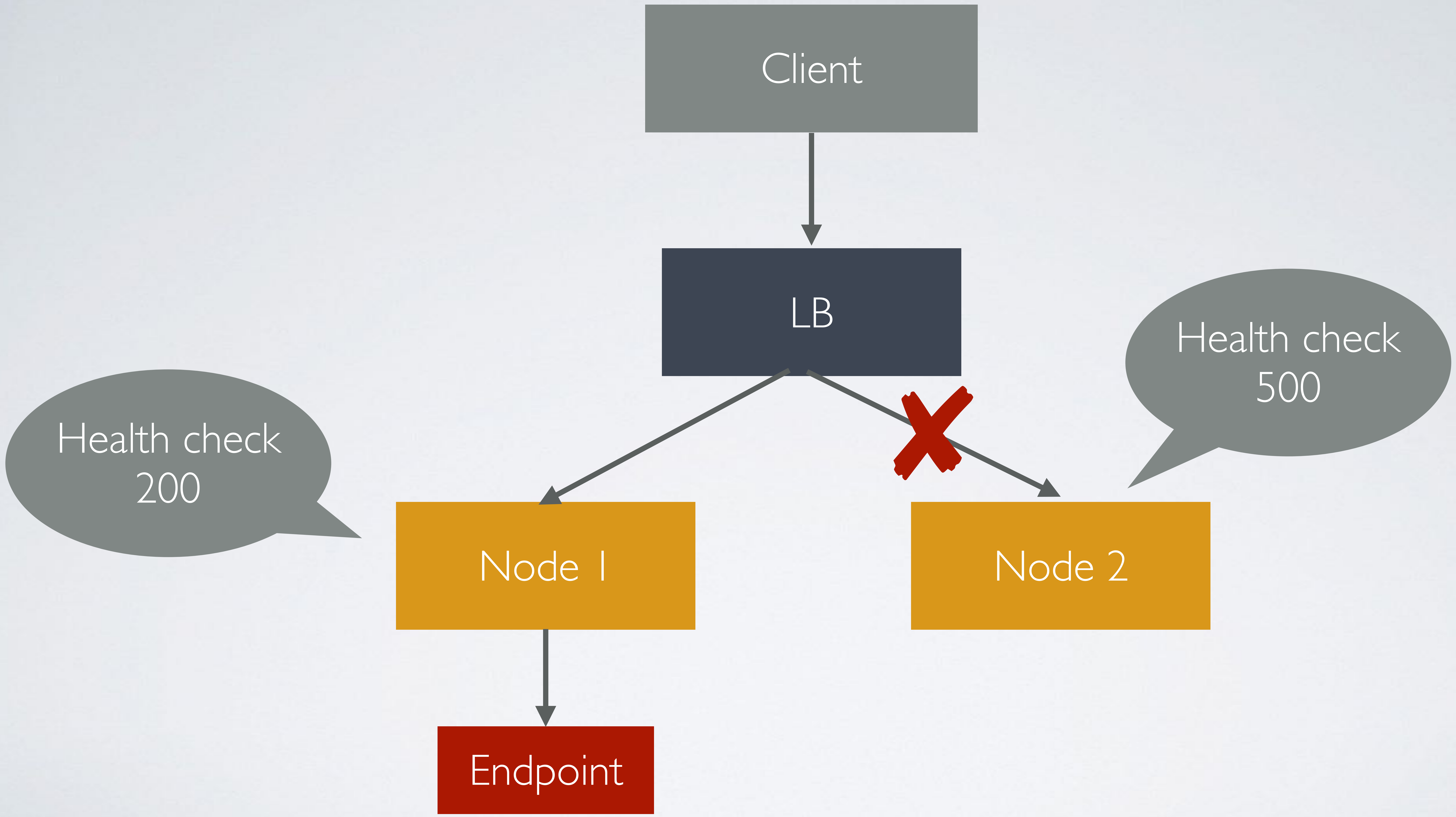
```yaml
livenessProbe:
  failureThreshold: 3
  httpGet:
    path: /healthz
    port: 10254
    scheme: HTTP
  initialDelaySeconds: 10
  periodSeconds: 10
  successThreshold: 1
  timeoutSeconds: 5
readinessProbe:
  failureThreshold: 3
  httpGet:
    path: /healthz
    port: 10254
    scheme: HTTP
  periodSeconds: 10
  successThreshold: 1
  timeoutSeconds: 1
```

Ingress

```yaml
kind: Ingress
apiVersion: extensions/v1beta1
metadata:
  name: nginx-ingress
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  rules:
  - host: echoserver.acoshift.com
    http:
      paths:
      - path: /
        backend:
          serviceName: echoserver
          servicePort: 80
  - host: echoserver.acoshift.me
    http:
      paths:
      - path: /
        backend:
          serviceName: echoserver
          servicePort: 80
  tls:
  - secretName: echoserver-acoshift-com-tls
    hosts:
    - echoserver.acoshift.com
```

Source IP

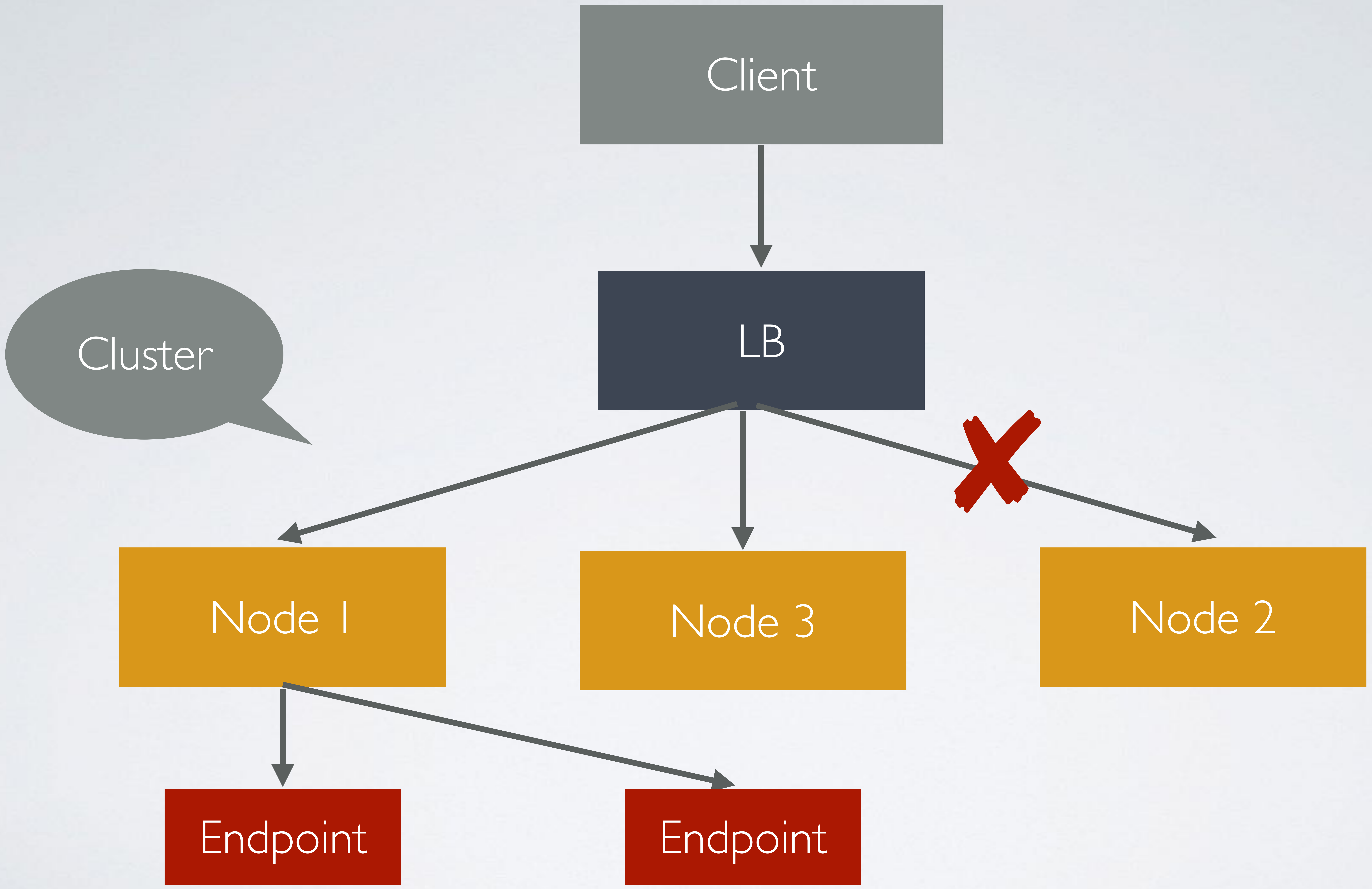

Client
LB
Node 1
SNAT
Node 2
Endpoint
Source IP lost

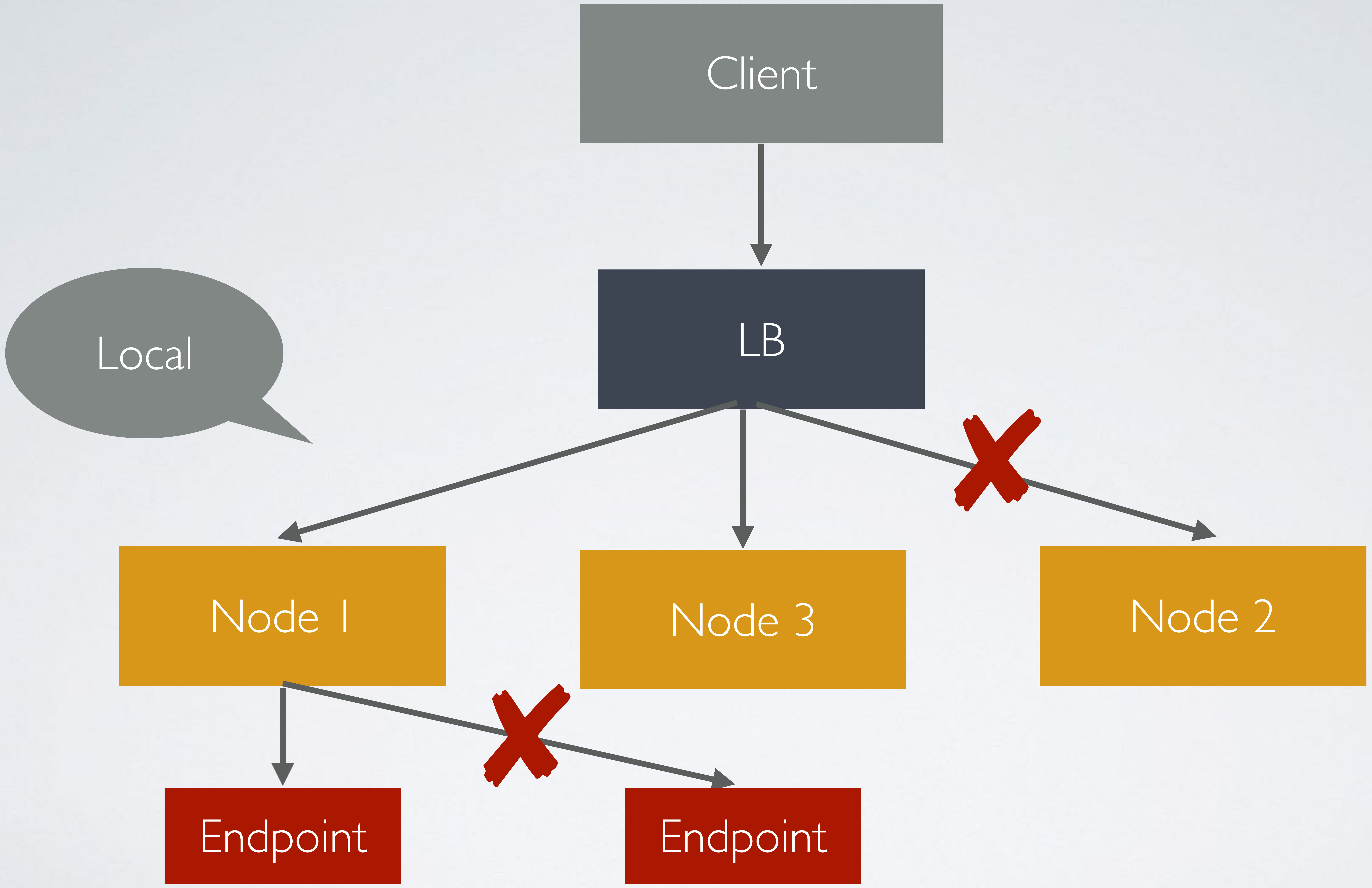

Client
LB
Health check 500
Health check 200
Node 1
Node 2
Endpoint

Source IP Type

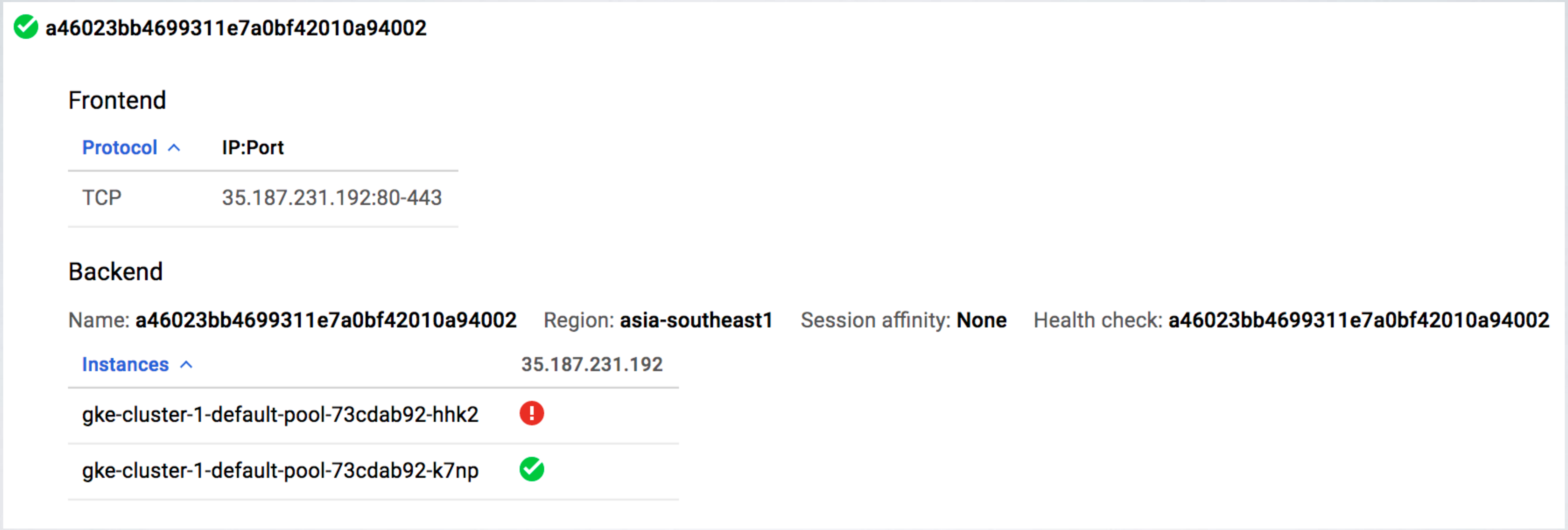
Default

- Cluster
- Local


Client
LB
Cluster
Node 1
Node 3
Node 2
Endpoint
Endpoint


Client
LB
Local
Node 1
Node 3
Node 2
Endpoint
Endpoint

```yaml
kind: Service
apiVersion: v1
metadata:
  name: nginx-ingress
spec:
  type: LoadBalancer
  externalTrafficPolicy: Local
  selector:
    app: nginx-ingress
...
```


✓ a46023bb4699311e7a0bf42010a94002

Frontend

Protocol ^	IP:Port
TCP	35.187.231.192:80-443

Backend

Name: **a46023bb4699311e7a0bf42010a94002** Region: **asia-southeast1** Session affinity: **None** Health check: **a46023bb4699311e7a0bf42010a94002**

Instances ^	35.187.231.192
gke-cluster-1-default-pool-73cdab92-hhk2	!
gke-cluster-1-default-pool-73cdab92-k7np	✓

kube-lego

automatically requests certificates for Kubernetes Ingress resources from Let's Encrypt

Q&A