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

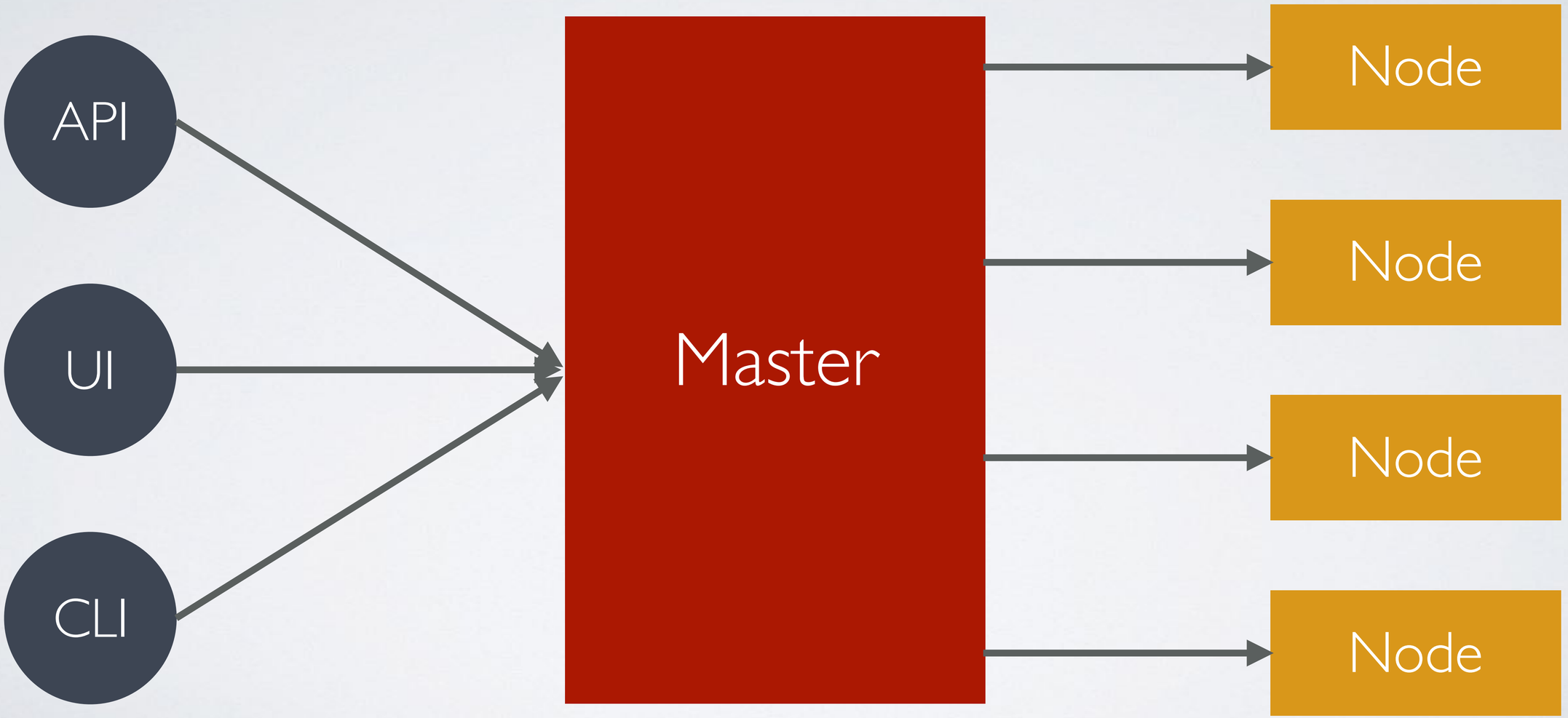
https://cloud.google.com/container-registry/

```
$ 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
```

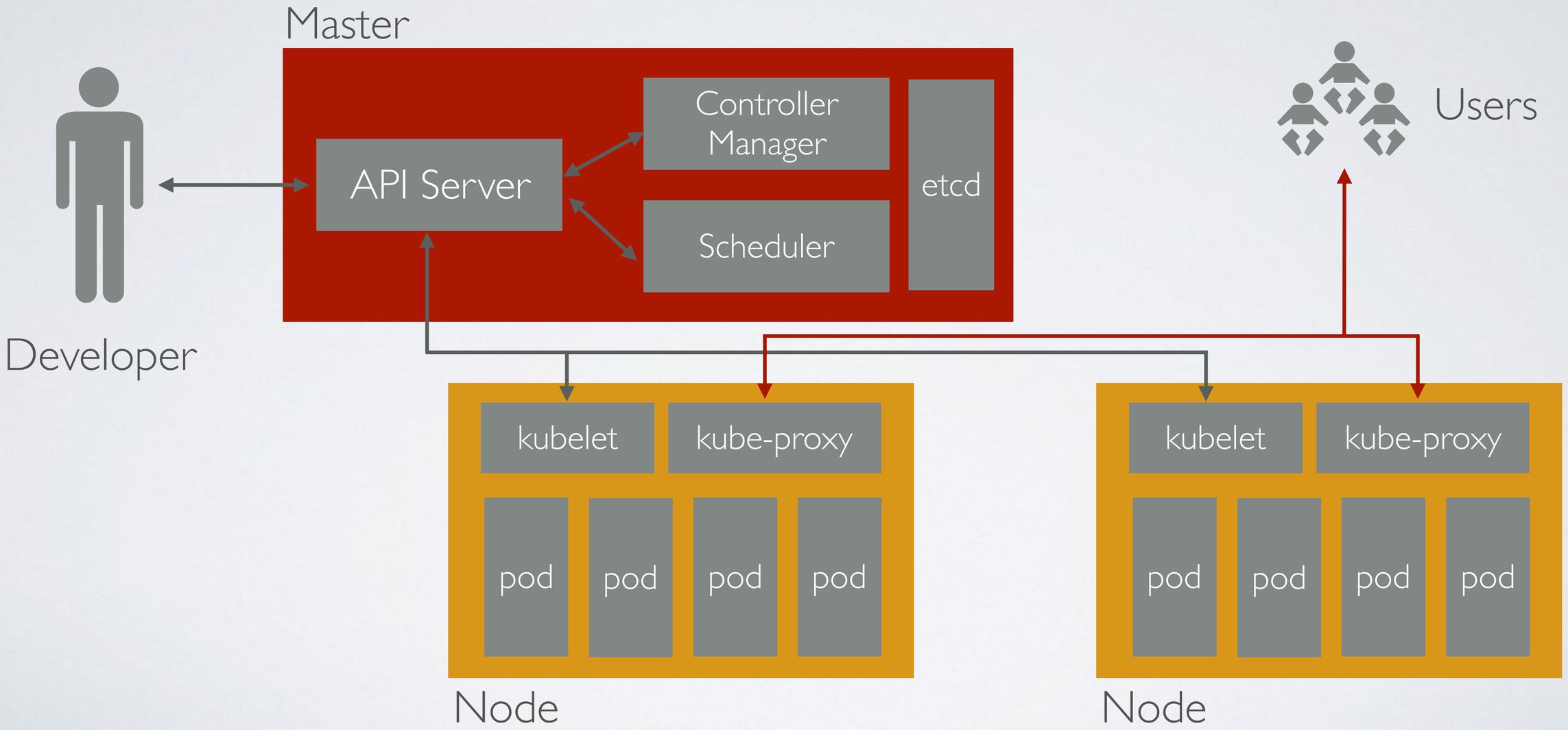
```
$ docker login -u _json_key -p "$(cat keyfile.json)" https://gcr.io
$ docker push gcr.io/myproject/backend:1.0.0
$ docker pull gcr.io/myproject/backend:1.0.0
```

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

Kubernetes Architecture



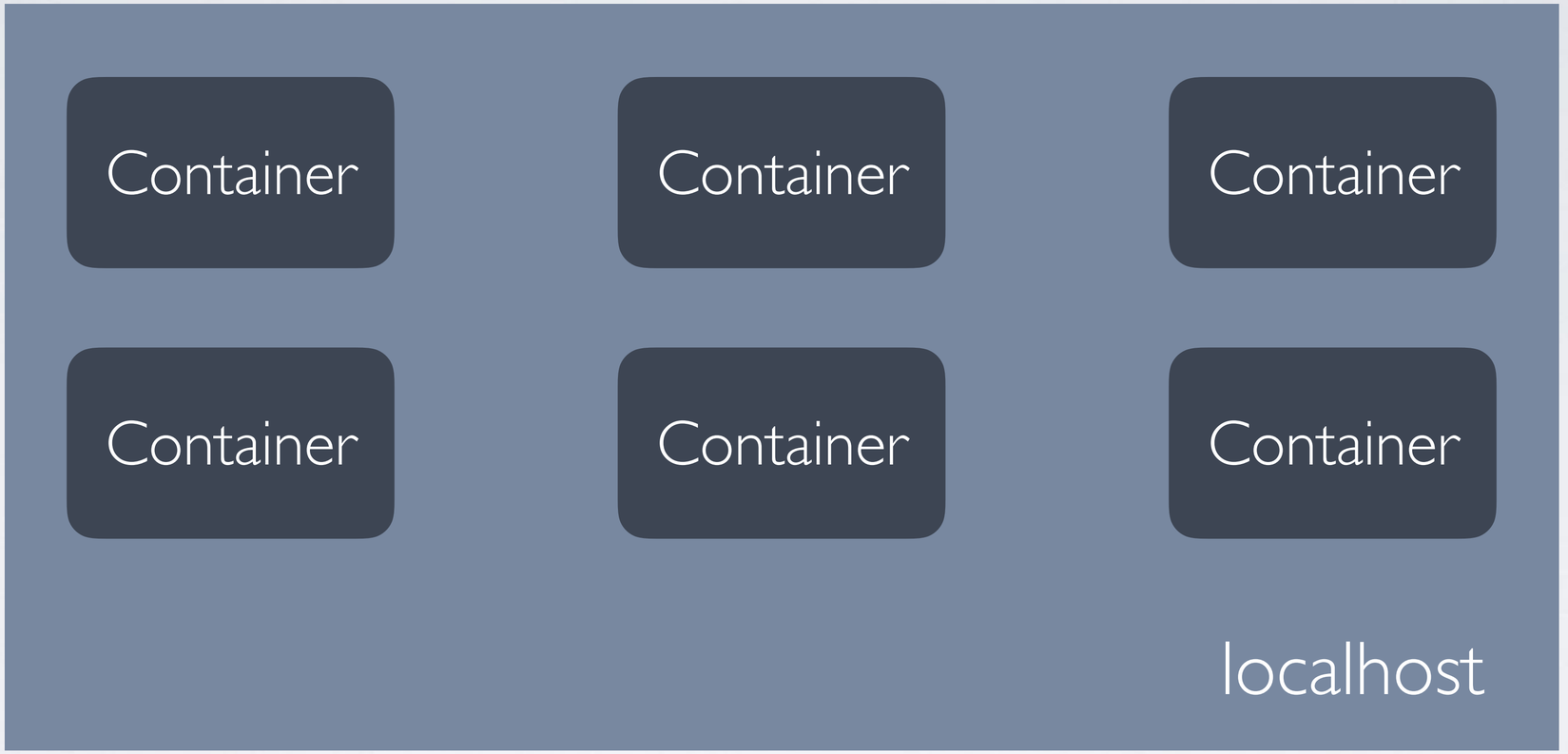
Kubernetes Architecture



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

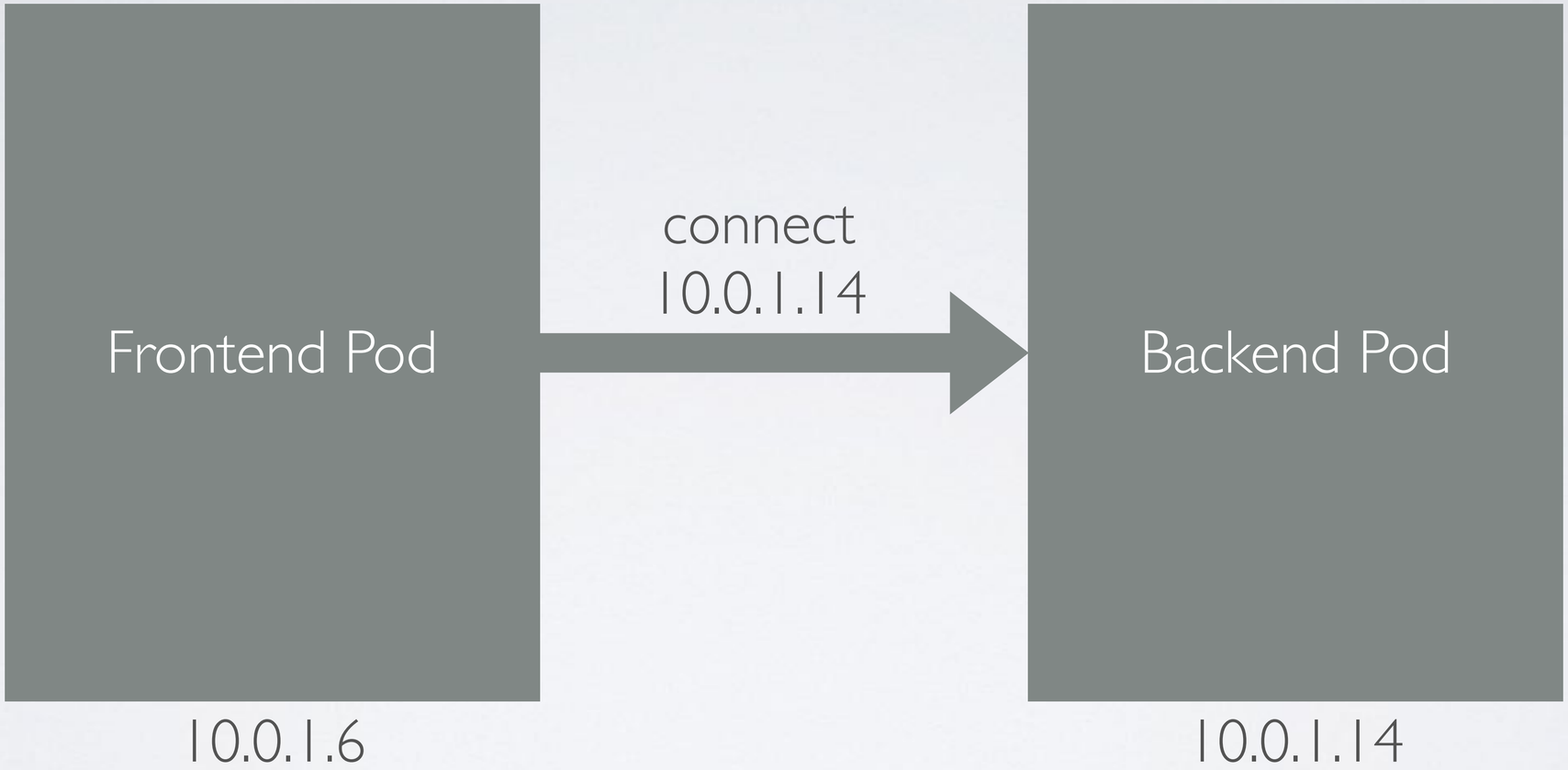
```
$ kubectl create -f 01-pod.yaml
pod "echoserver" created
```

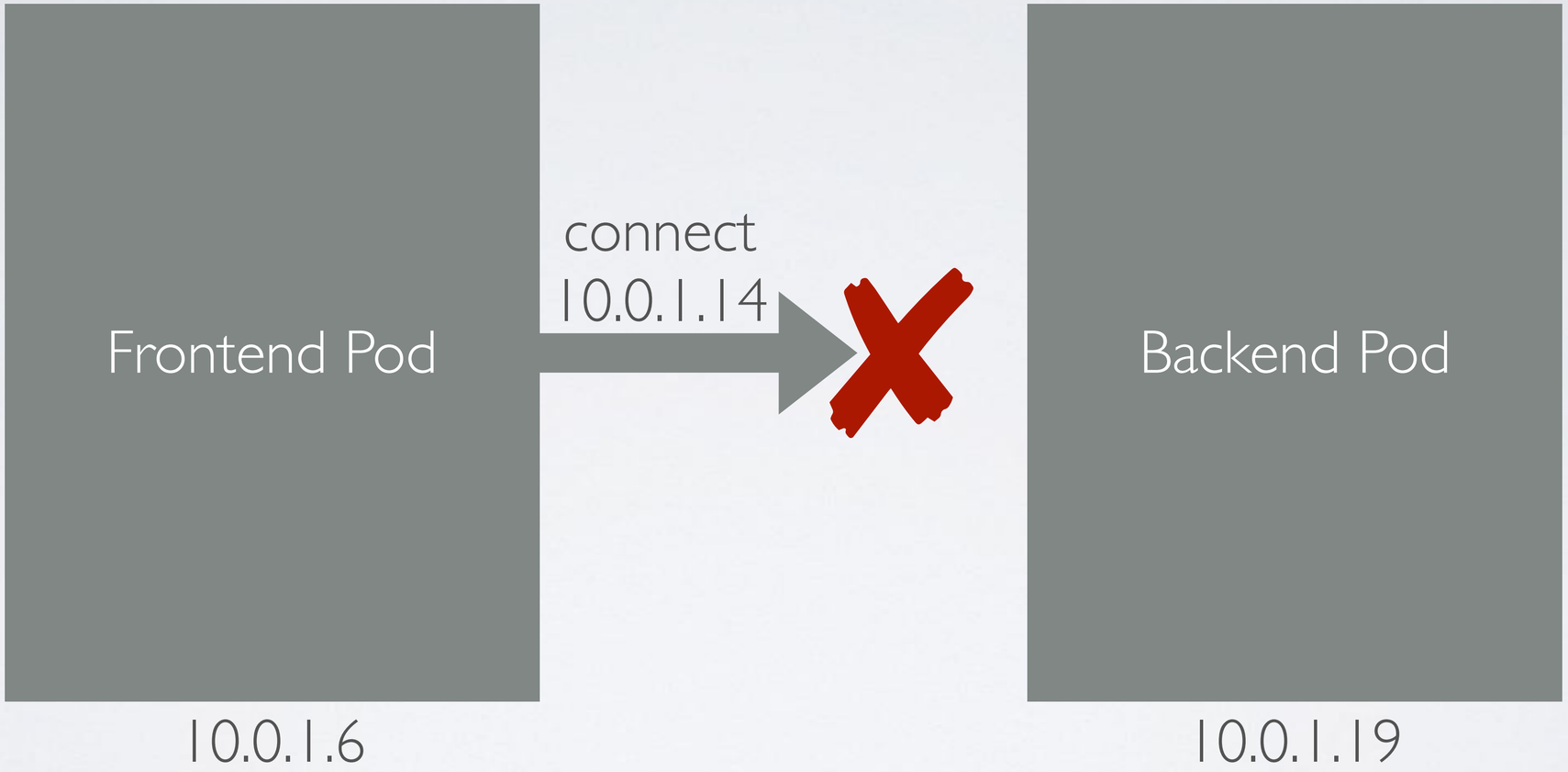
```
$ kubectl get pods
NAME              READY     STATUS    RESTARTS   AGE
echoserver        1/1       Running   0          4m
```

```
$ 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

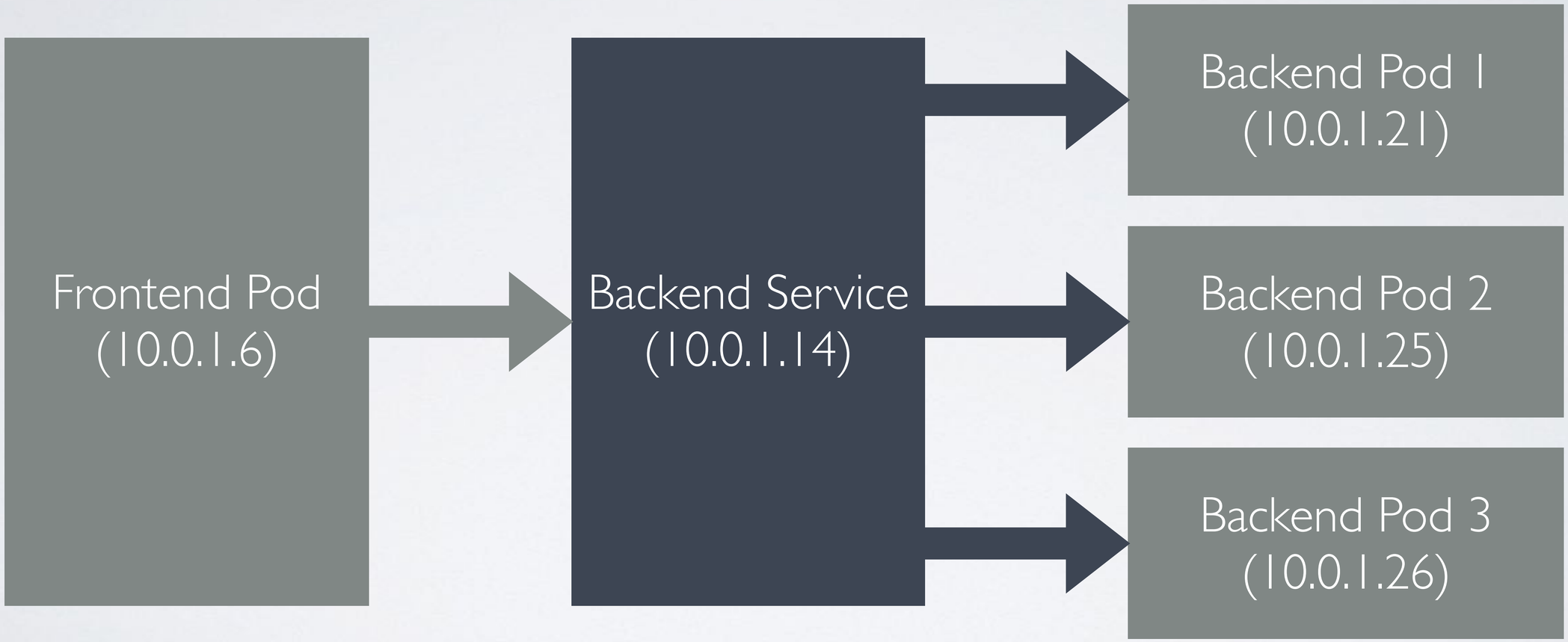
Services (svc)

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
```

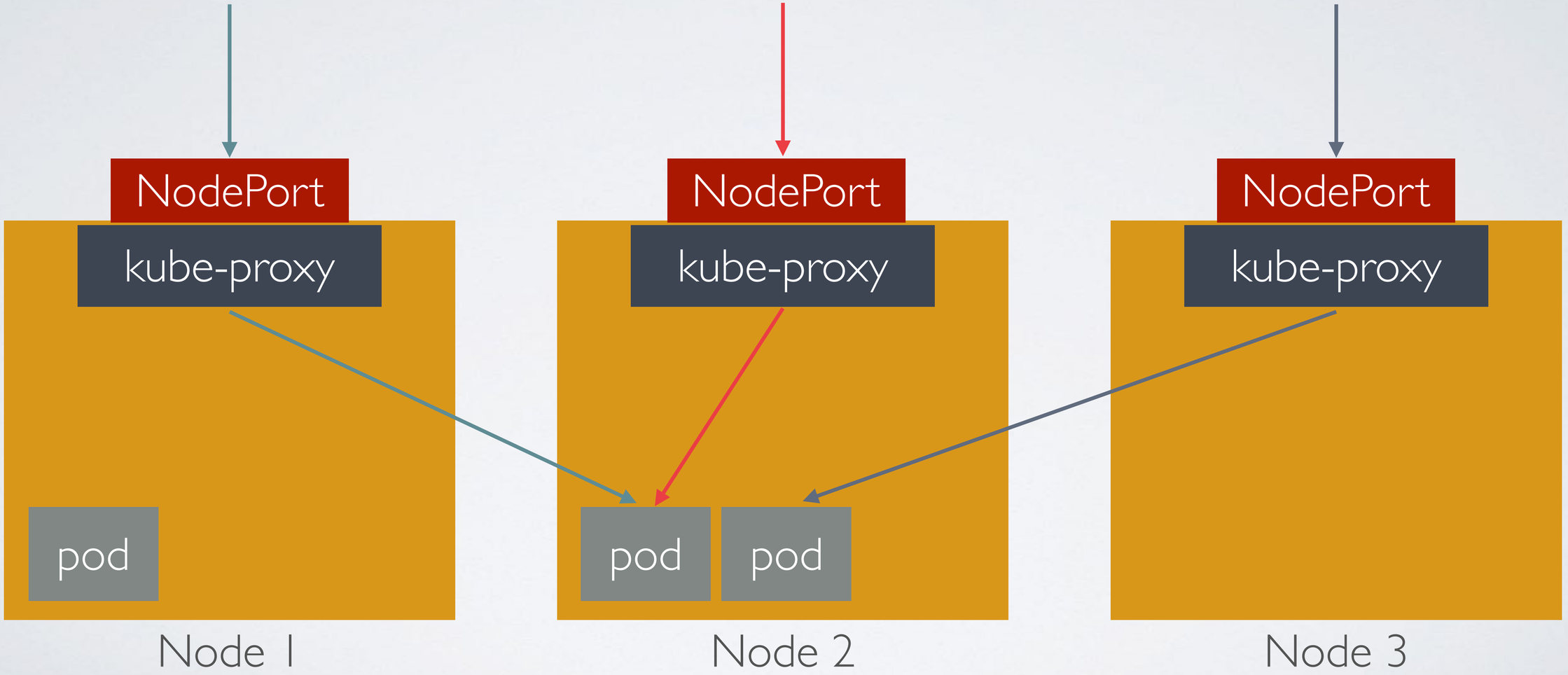
```
$ kubectl create -f 02-service.yaml
pod "echoserver" created
service "echoserver" created
```

```
$ 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
```

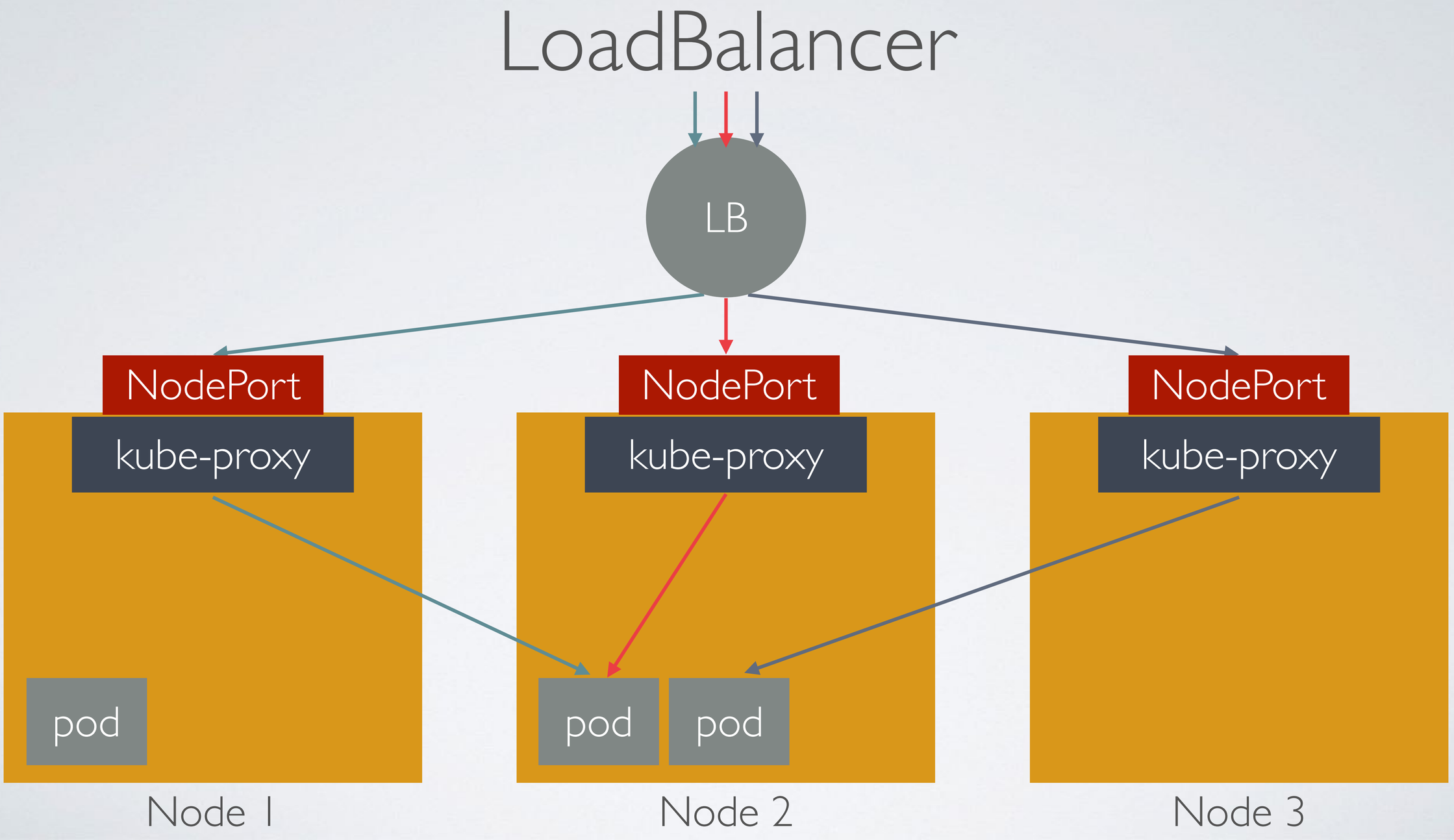
NodePort


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

```
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

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

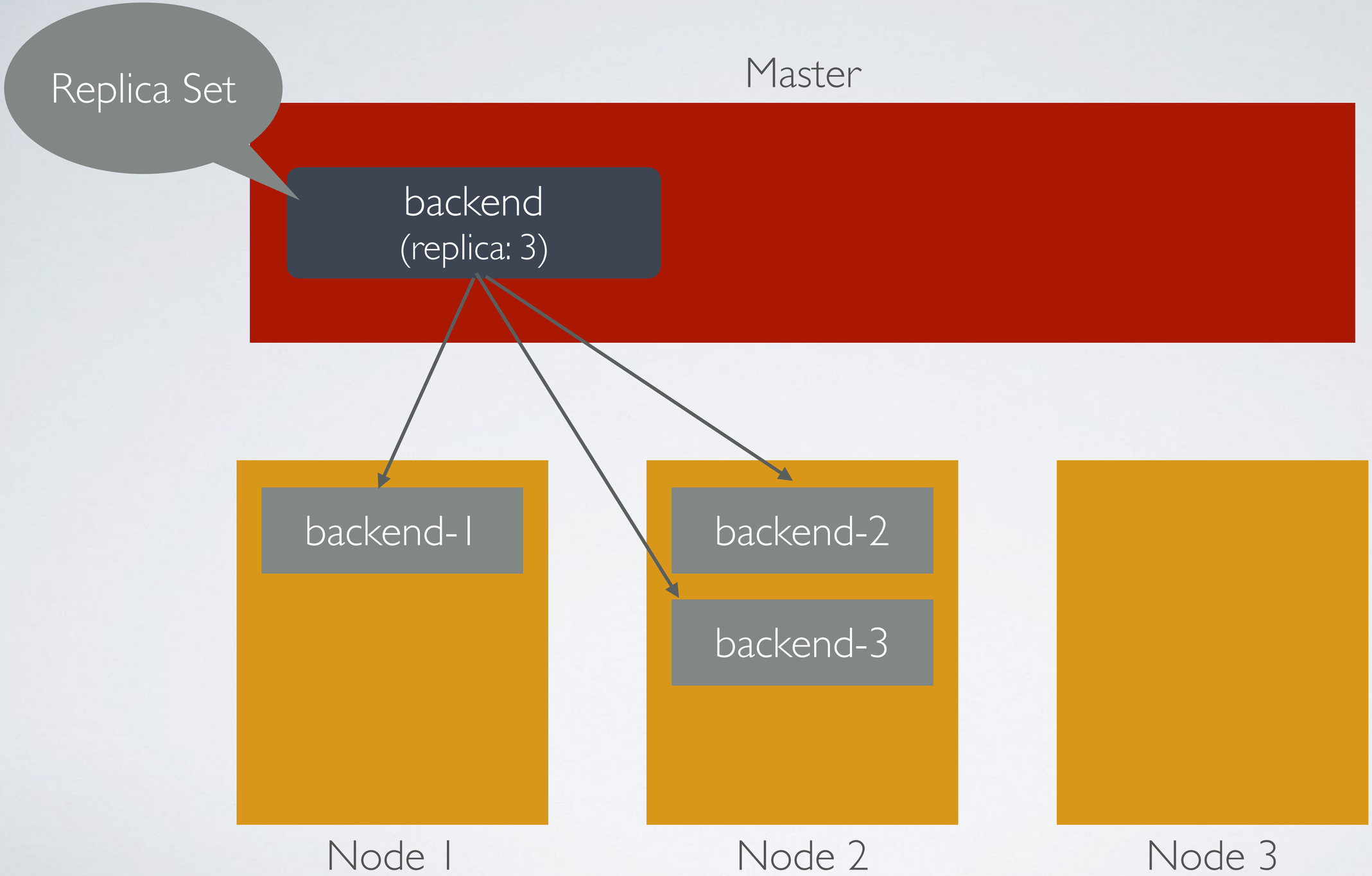
```
$ kubectl run -i -t --rm busybox --image=busybox
$ wget -O - --header="Host: www.google.com" http://google
```

~~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

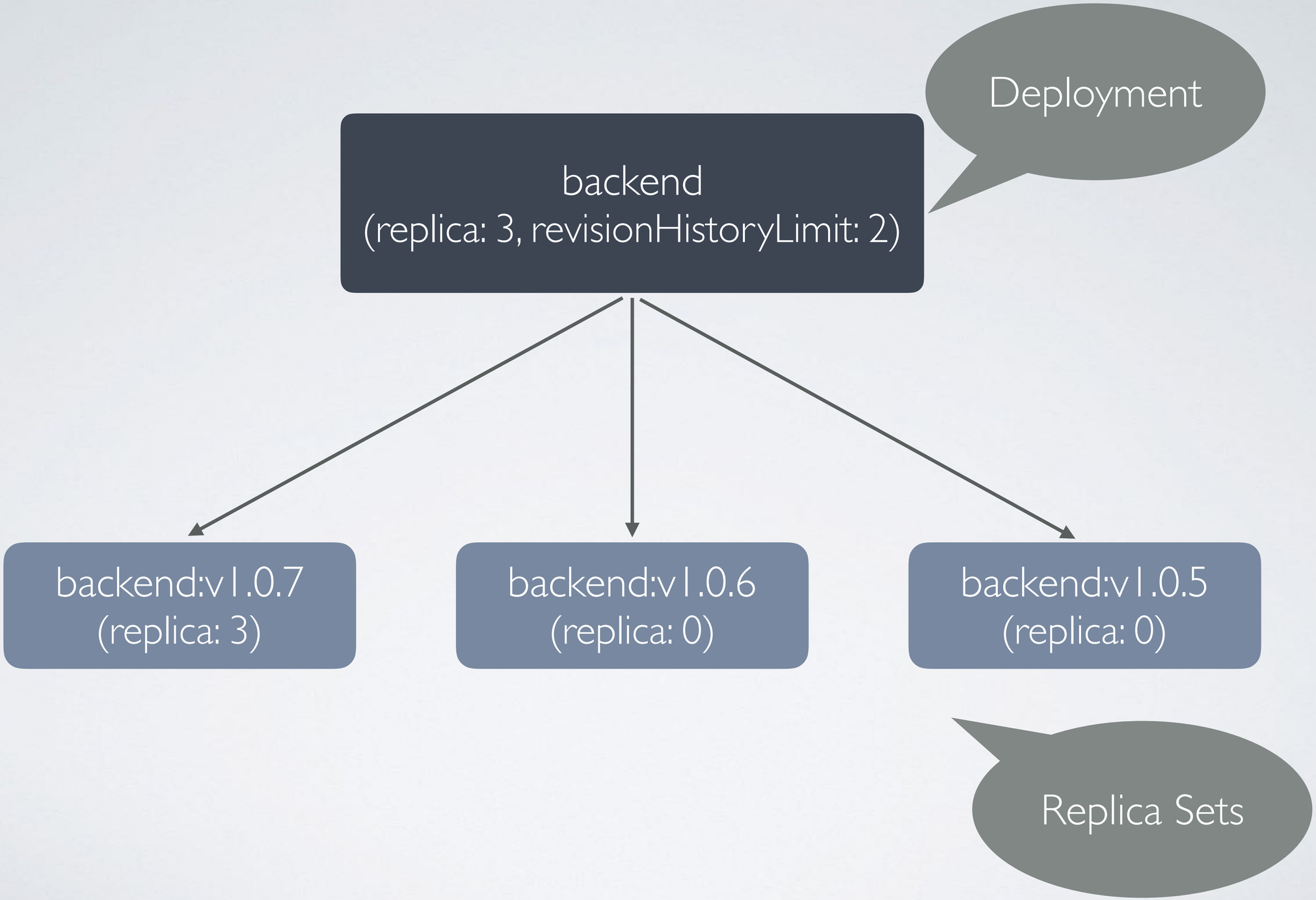

Replica Set
Master
backend
(replica: 3)
backend-1
backend-2
backend-3
Node 1
Node 2
Node 3

```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

- 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>
```

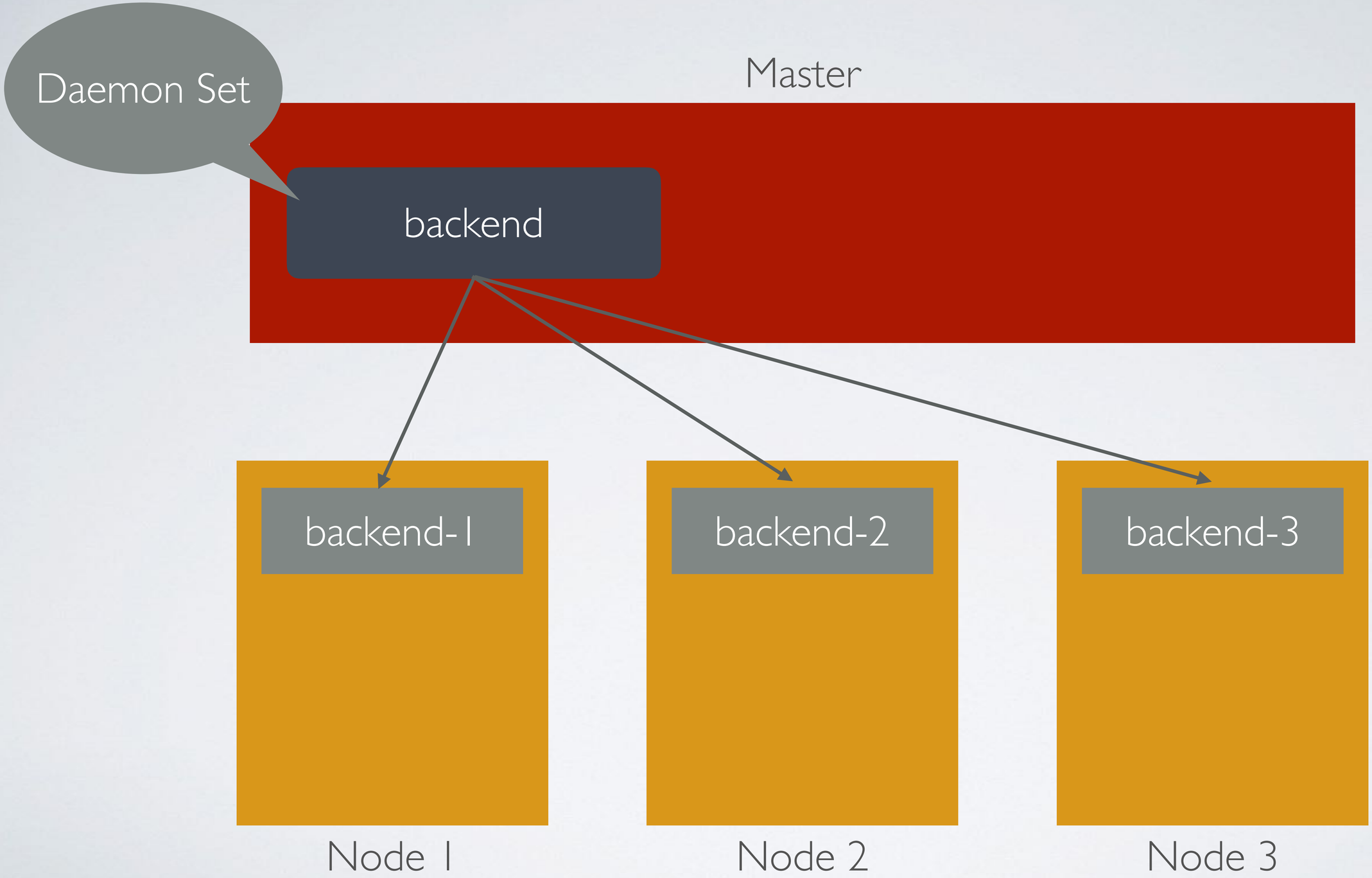
```
$ kubectl rollout undo deployment/echoserver
deployment "echoserver" rolled back

$ 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
```

Q&A

DAY 2

Persistent Volumes (pv)

a piece of storage in the cluster that has been provisioned

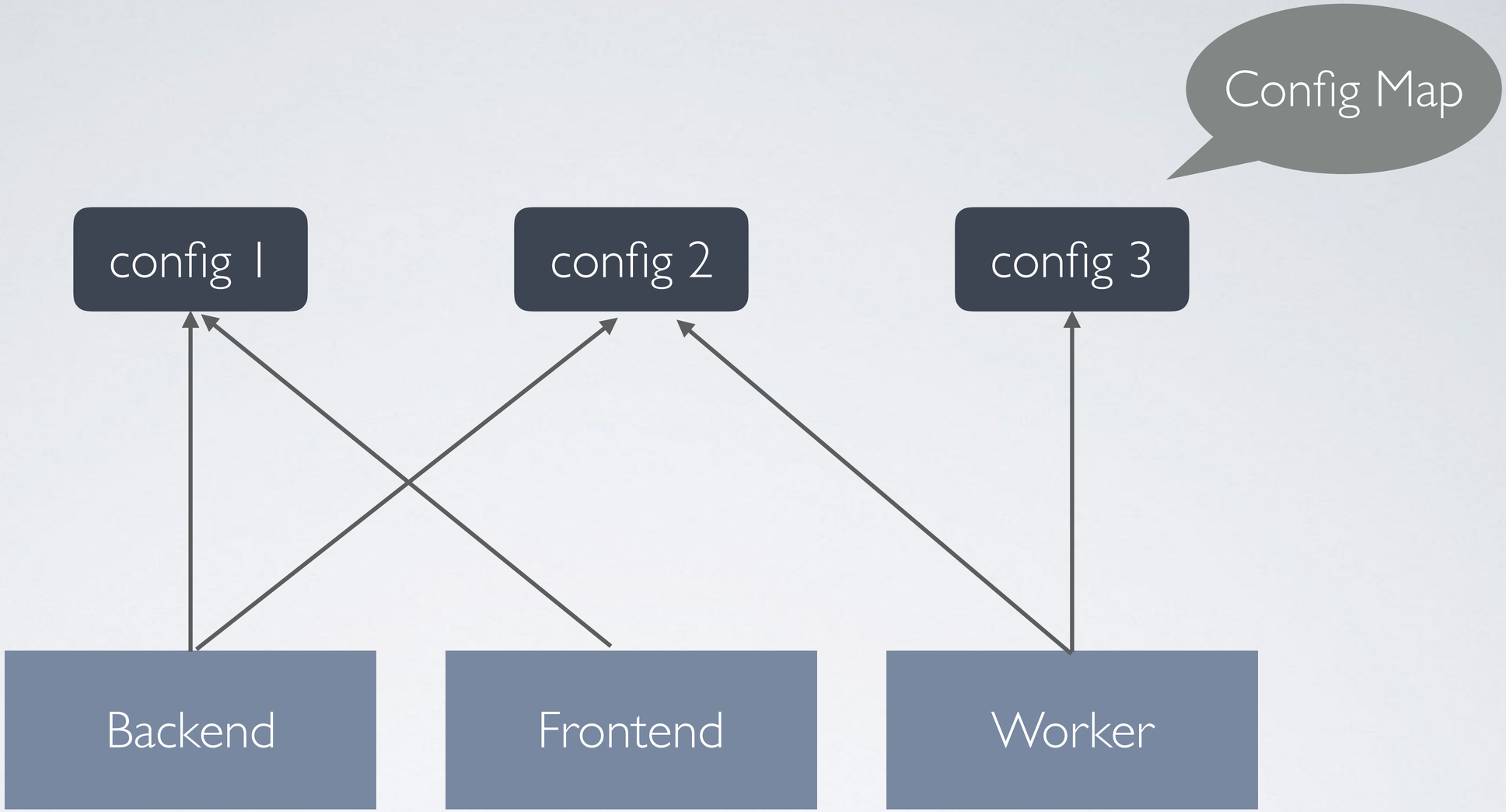
Persistent Volume Claim (pvc)

a request for storage

Config Maps (cm)

decouple configuration artifacts from image content
to keep containerized applications portable

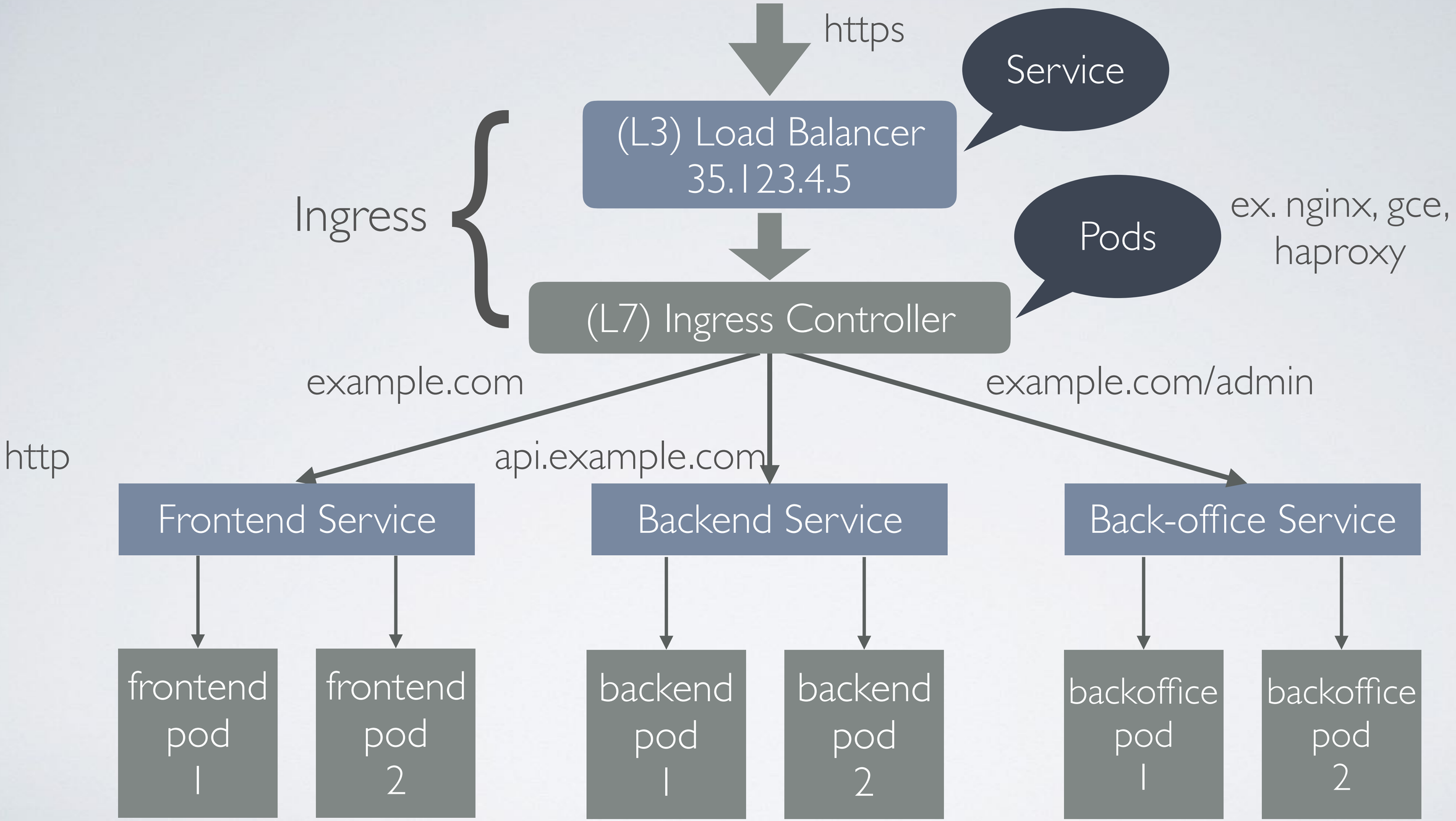

Backend
config 1
config 2
Frontend
config 1
Worker
config 2
config 3


Config Map
config 1
config 2
config 3
Backend
Frontend
Worker

Secrets

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

GCE Load-Balancer Controller (GLBC)

```yaml
kind: Ingress
apiVersion: extensions/v1beta1
metadata:
  name: glbc-ingress
spec:
  backend:
    serviceName: default-http-backend
    servicePort: 80
  tls:
  - secretName: tls-secret
```

```yaml
kind: Ingress
apiVersion: extensions/v1beta1
metadata:
  name: glbc-ingress
spec:
  rules:
  - host: echo.acoshift.com
    http:
      paths:
      - path: /*
        backend:
          serviceName: echoserver
          servicePort: 8080
  - host: echo.acoshift.me
    http:
      paths:
      - path: /*
        backend:
          serviceName: echoserver
          servicePort: 8080
  tls:
  - secretName: echo-acoshift-com-tls
    hosts:
    - echo.acoshift.com
```

Nginx Ingress Controller

Stateful Sets

Jobs

creates one or more pods and ensures that
a specified number of them successfully terminate

Cron Jobs

manages time based Jobs