# Understanding services

Kubernetes Pods are mortal. They are born and die.

Replication Controller maintains the desired count of Pods all the time.

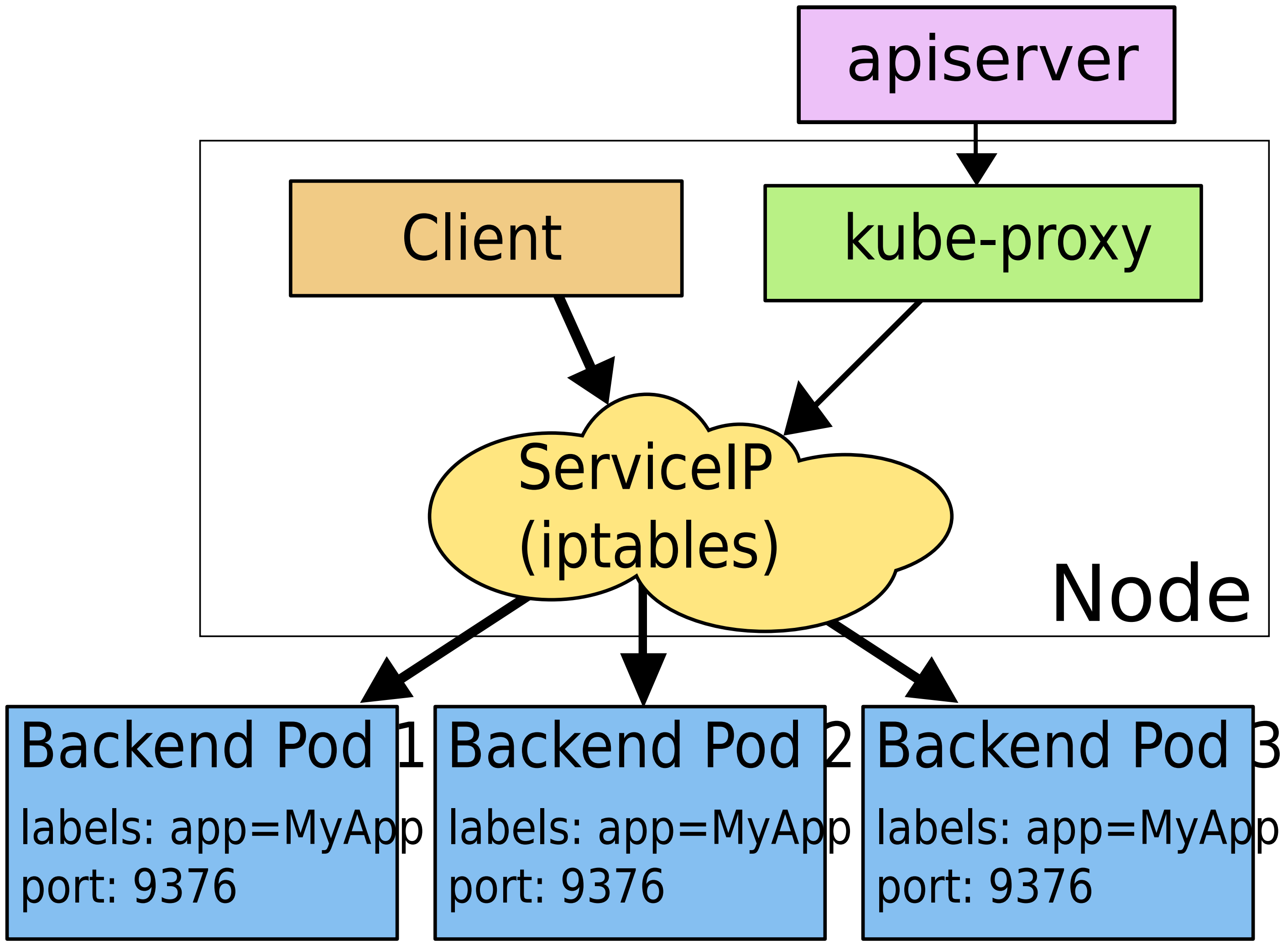
Pod IP address may change during its lifetime.

## Understanding services

Every node in a Kubernetes cluster runs a kube-proxy. kube-proxy is responsible for implementing a form of virtual IP for Services

kube-proxy watches the Kubernetes master for the addition and removal of Service and Endpoints objects. For each Service it installs iptables rules which capture traffic to the Service’s clusterIP

## Understanding services



## Publishing services - service types

- ClusterIP: Exposes the service on a cluster-internal IP. Choosing this value makes the service only reachable from within the cluster. This is the default ServiceType.

- NodePort: Exposes the service on each Node’s IP at a static port (the NodePort). A ClusterIP service, to which the NodePort service will route, is automatically created. You’ll be able to contact the NodePort service, from outside the cluster, by requesting <NodeIP>:<NodePort>.

## Publishing services - service types

- LoadBalancer: Exposes the service externally using a cloud provider’s load balancer. NodePort and ClusterIP services, to which the external load balancer will route, are automatically created.

- ExternalName: Maps the service to the contents of the externalName field (e.g. foo.bar.example.com), by returning a CNAME record with its value. No proxying of any kind is set up. This requires version 1.7 or higher of kube-dns.

# Discovering services - Env Vars

- Kubernetes creates Docker Link compatible environment variables in all Pods
- Containers can use the environment variable to talk to the service endpoint

{SVCNAME}_SERVICE_HOST and {SVCNAME}_SERVICE_PORT variables, where the Service name is upper-cased and dashes are converted to underscores.

For example, the Service "redis-master" which exposes TCP port 6379 and allocated cluster IP address 10.0.0.11

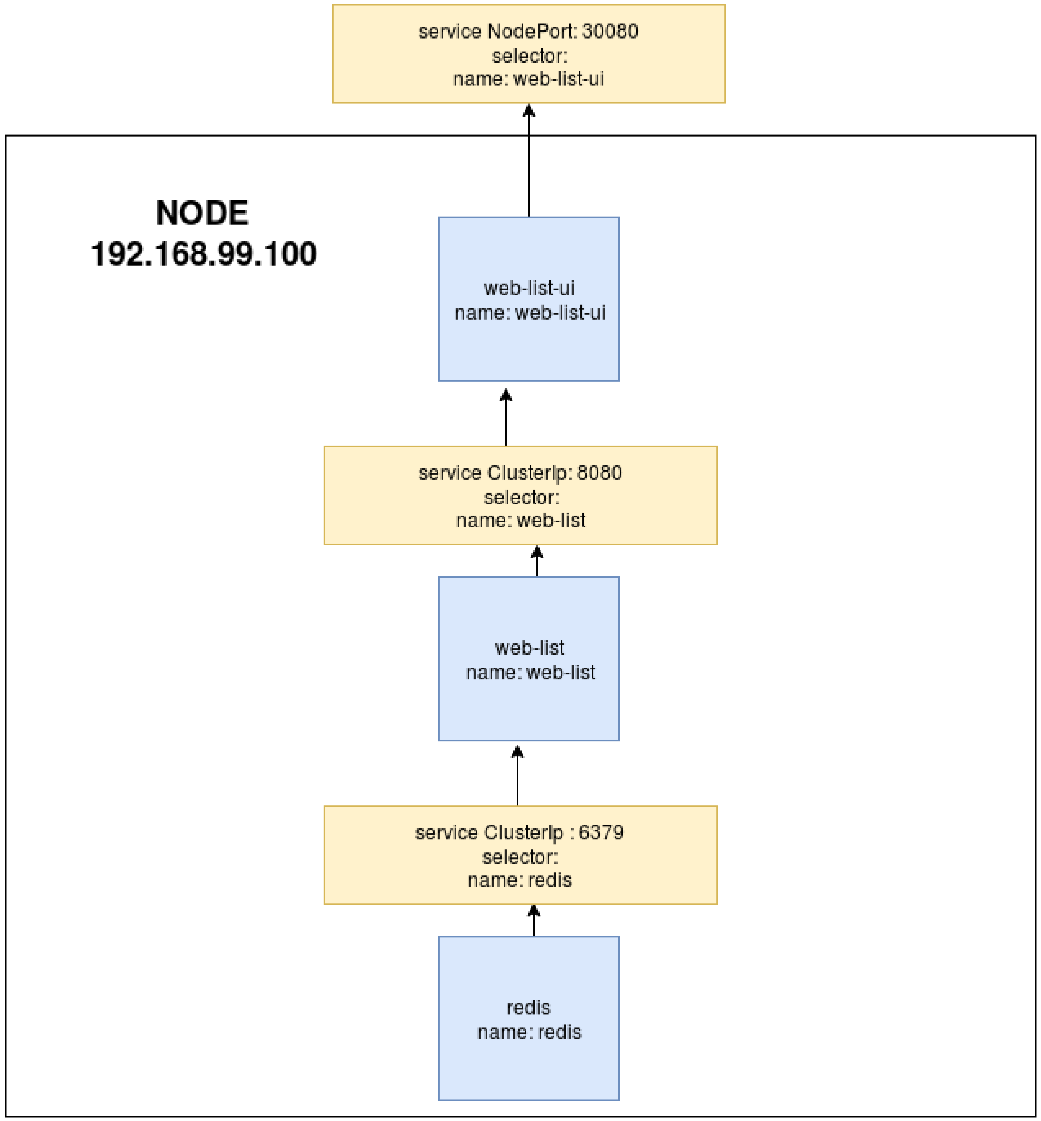
REDIS_MASTER_SERVICE_HOST=10.0.0.11

REDIS_MASTER_SERVICE_PORT=6379

## Discovering services - DNS

- The DNS server watches Kubernetes API for new Services
- The DNS server creates a set of DNS records for each Service
- Services can be resolved by the name within the same namespace
- Pods in other namespaces can access the Service by adding the
- namespace to the DNS path : my-service.my-namespace

# Demo



service NodePort: 30080
selector:
name: web-list-ui
NODE
192.168.99.100
web-list-ui
name: web-list-ui
service ClusterIp: 8080
selector:
name: web-list
web-list
name: web-list
service ClusterIp : 6379
selector:
name: redis
redis
name: redis