Networking

Service discovery

- Kubernetes is quite dynamic with placing pods on nodes, horizontal autoscalers etc.
- Service discovery helps us find the services in the cluster
- Should resolve quickly and reliably
-

The Service Object

A Service Object is a way to create a named label selector

```
# Creates a service:
kubectl expose deployment nginx

# Show including the selector:
kubectl get services -o wide
```

```
PS >kubectl get services -o wide
NAME                  TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE     SELECTOR
default-http-backend  ClusterIP   10.152.183.94   <none>        80/TCP     45h     app=default-http-backend
kubernetes            ClusterIP   10.152.183.1    <none>        443/TCP    4d20h   <none>
webapp                ClusterIP   10.152.183.44   <none>        8080/TCP   44h     run=webapp
```

Port-forward to service

```
# Connect to the pod
$NGINX_POD=$(kubectl get pods -l run=webapp \
-o jsonpath='{.items[0].metadata.name}')
```

Service DNS

- Kubernetes provides a DNS service exposed to Pods
- System component running in and managed by K8s

`kubectl exec -ti busybox -- nslookup webapp`

```
PS >kubectl exec -ti busybox -- nslookup webapp
Server:    10.152.183.10
Address 1: 10.152.183.10 kube-dns.kube-system.svc.cluster.local

Name:      webapp
Address 1: 10.152.183.44 webapp.default.svc.cluster.local
```

- Kubectl edit service webapp
- Change type to NodePort

- Kubectl describe service webapp

```
PS >kubectl describe service webapp
Name:                     webapp
Namespace:                default
Labels:                   run=webapp
Annotations:              <none>
Selector:                 run=webapp
Type:                     NodePort
IP:                       10.152.183.44
Port:                     <unset>  8080/TCP
TargetPort:               8080/TCP
NodePort:                 <unset>  30430/TCP
Endpoints:                10.1.1.26:8080
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```

LoadBalancer

- Only available if cloud supports it
- Builds on NodePorts
- Creates a loadbalancer in the cloud and direct it at nodes in cluster

- Kubectl edit service webapp
- Change spec.type to LoadBalancer
- Kubectl get services now gets an external ip (pending)

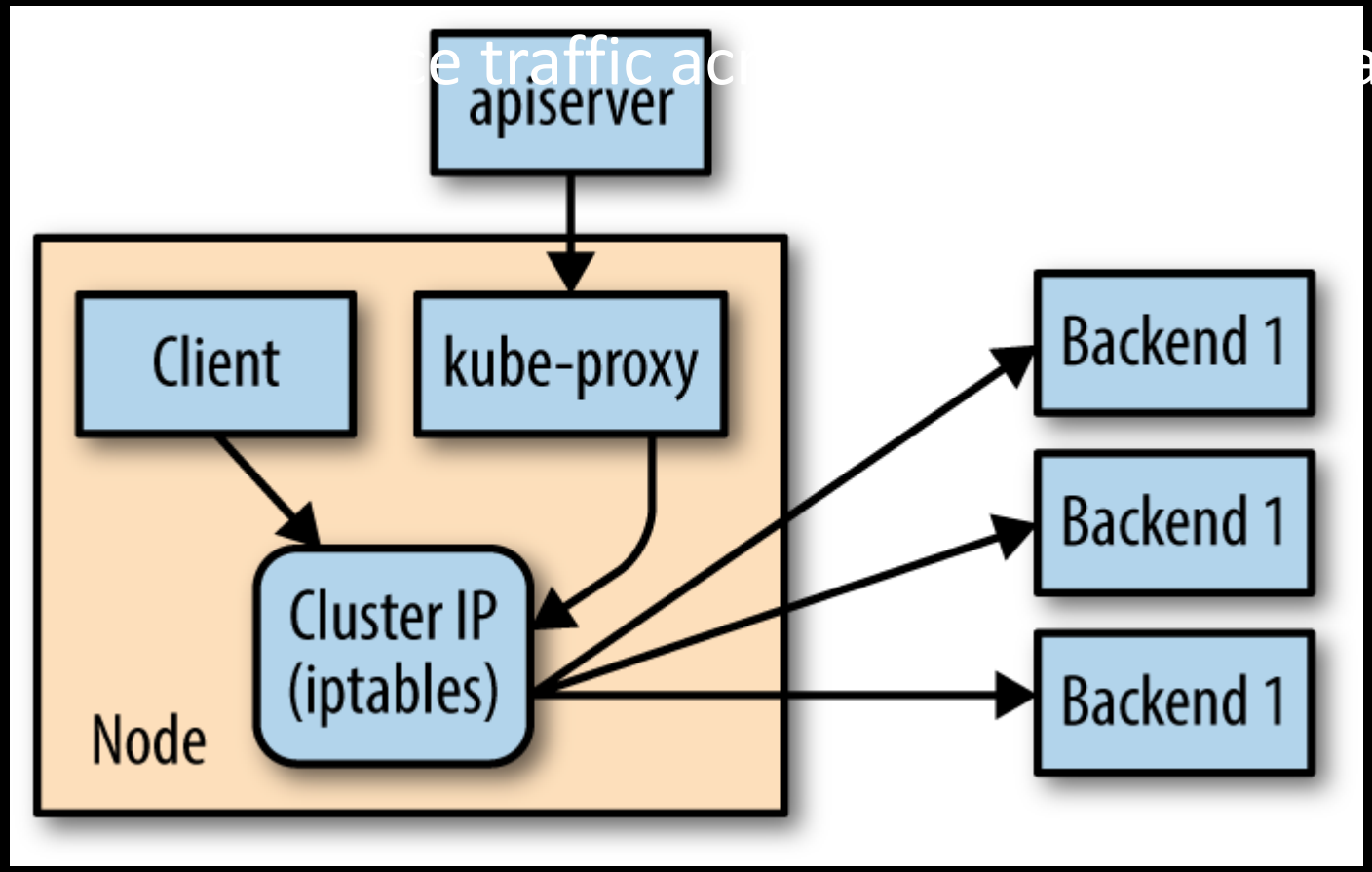
Endpoints

- Endpoints are created as a pair with services
- Kubectl describe endpoints webapp

```
PS >kubectl describe endpoints webapp
Name:         webapp
Namespace:    default
Labels:       run=webapp
Annotations:  <none>
Subsets:
  Addresses:          10.1.1.26
  NotReadyAddresses:  <none>
  Ports:
    Name     Port  Protocol
    ----     ----  --------
    <unset>  8080  TCP
```

Kube-proxy and Cluster IPs

- Cluster Ips are stable virtual IPs,



- kubectl exec -it $nginx_pod -- env | grep WEBAPP_

```
PS> kubectl exec -it $nginx_pod -- env | grep WEBAPP_
WEBAPP_PORT=tcp://10.152.183.44:8080
WEBAPP_SERVICE_HOST=10.152.183.44
WEBAPP_PORT_8080_TCP_PROTO=tcp
WEBAPP_PORT_8080_TCP_ADDR=10.152.183.44
WEBAPP_SERVICE_PORT=8080
WEBAPP_PORT_8080_TCP=tcp://10.152.183.44:8080
WEBAPP_PORT_8080_TCP_PORT=8080
```

- kubectl delete services,deployments -l run=webapp

Ingress

Ingress

- Exposing service with http or https
- Other services use:
  - NodePort
  - LoadBalancer
- Uses an ingress controller
  - Nginx-ingress
  - HAProxy, Traefik, Istio, Kong,

Single service

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: test-ingress
spec:
  backend:
    serviceName: testsvc
    servicePort: 80
```

Ingress

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: test-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /testpath
        backend:
          serviceName: test
          servicePort: 80
```

Enable Ingress

```
multipass exec microk8s-vm --
/snap/bin/microk8s.enable ingress
```

```
Enabling Ingress
deployment.extensions/default-http-backend unchanged
service/default-http-backend unchanged
serviceaccount/nginx-ingress-microk8s-serviceaccount unchanged
clusterrole.rbac.authorization.k8s.io/nginx-ingress-microk8s-clusterrole unchanged
role.rbac.authorization.k8s.io/nginx-ingress-microk8s-role unchanged
clusterrolebinding.rbac.authorization.k8s.io/nginx-ingress-microk8s unchanged
rolebinding.rbac.authorization.k8s.io/nginx-ingress-microk8s unchanged
configmap/nginx-load-balancer-microk8s-conf unchanged
daemonset.apps/nginx-ingress-microk8s-controller unchanged
Ingress is enabled
```

DaemonSets

- Ensures a copy of a Pod is running across a set of nodes in a Kubernetes cluster
- Typically used for log collectors, monitoring tools etc.
- Like ReplicaSets. DaemonSets run exactly 1 copy per node.
- Add additional capabilities and features to the Kubernetes cluster itself instead of running normal services

DaemonSet definition

```
apiVersion: extensions/v1beta1
kind: DaemonSet
metadata:
  name: fluentd
  namespace: kube-system
  labels:
    app: fluentd
spec:
  template:
    metadata:
      labels:
        app: fluentd
    spec:
      containers:
      - name: fluentd
        image: fluent/fluentd:v0.14.10
```

Node selectors with DaemonSets

- spec:
- nodeSelector:
- ssd: "true"

Jobs

- So far we've looked at long running processes
- Jobs are short-lived, typically one-off tasks
- A Job creates Pods that run until successful termination
- Pods restart even when successful terminated

Job Types

Type	Use case	Behavior	completions	parallelism
One shot	Database migrations	A single pod running once until successful termination	1	1
Parallel fixed completions	Multiple pods processing a set of work in parallel	One or more Pods running one or more times until reaching a fixed completion count	1+	1+
Work queue: parallel Jobs	Multiple pods processing from a centralized work queue	One or more Pods running once until successful termination	1	2+

- kubectl run -i oneshot \
- --image=<some-image:latest> \
- --restart=OnFailure

- kubectl delete jobs oneshot

- kubectl get pod -a -l job-name=oneshot
- Describe, edit etc are available

```
apiVersion: batch/v1
kind: Job
metadata:
  name: oneshot
  labels:
    chapter: jobs
spec:
  template:
    metadata:
      labels:
        chapter: jobs
    spec:
      containers:
      - name: my-job-container
        image: <some-image>
        imagePullPolicy: Always
        args:
restartPolicy: OnFailure
```

Parallel jobs

- Configure the following variables.
- completions: 10
  - Number of jobs to run
- parallelism: 5
  - Max. parallel executions (same time)

```
apiVersion: batch/v1
kind: Job
metadata:
  name: parallel
  labels:
    chapter: jobs
spec:
  parallelism: 5
  completions: 10
  template:
    metadata:
      labels:
        chapter: jobs
    spec:
      containers:
      - name: kuard
        image: <some-image>
        imagePullPolicy: Always
        args:
restartPolicy: OnFailure
```