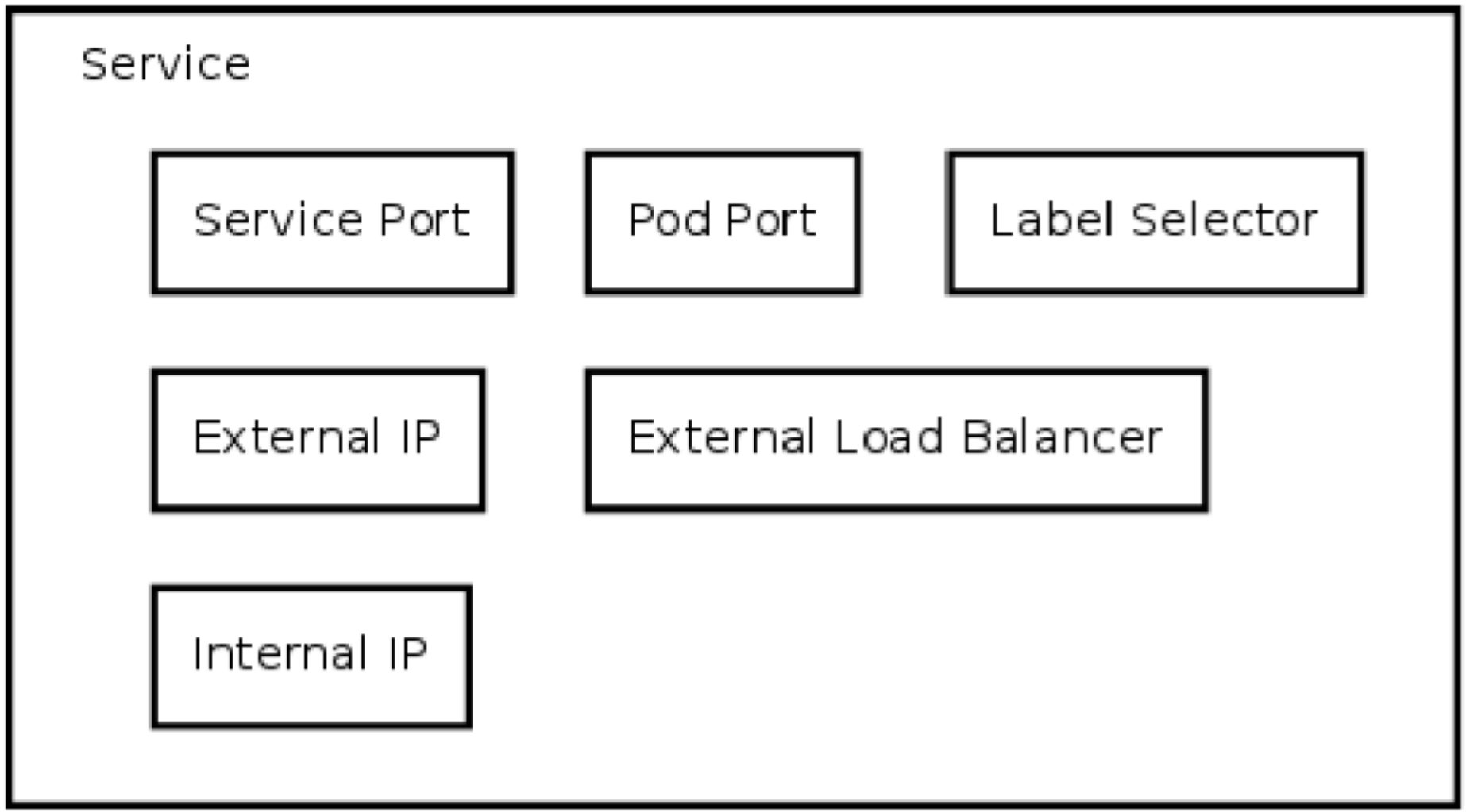
Services

- An abstraction to define a logical set of Pods bound by a policy by to access them
- Services are exposed through internal and external endpoints
- Services can also point to non-Kubernetes endpoints through a Virtual-IP-Bridge
- Supports TCP and UDP
- Interfaces with kube-proxy to manipulate iptables
- Service can be exposed internal or external to the cluster

Services

- How 'stuff' finds pods which could be anywhere
- Define:
  - What port in the container
  - Labels on pods which should respond to this type of request
- Can define:
  - What the 'internal' IP should be
  - What the 'external' IP should be
  - What port the service should listen on

Services


Service
Service Port
Pod Port
Label Selector
External IP
External Load Balancer
Internal IP

Services

- **Pods** are very **dynamic**, they come and go on the kubernetes cluster
  - When using a **replication-controller**, pods are **terminated** and created during scaling operations
  - When using **Deployments**, when **updating** the image version, pods are **terminated** and new pods take the place of older pods
- That's why pods should never be accessed directly, but always through a **Service**
- A Service is **logical bridge** between the "mortal" pods and other **services** or **end-users**

Services

- When using the "kubectl expose" command earlier, you created a new service for your pod, so it could be accessed externally
- Create a service will create an endpoints for your pod(s)
  - A **ClusterIP**: a virtual IP address only reachable from within the cluster. (this is default)
  - A **Nodeport**:   a port that is the same on each node that is also reachable externally
  - A **LoadBalancer**: a LoadBalancer created by the cloud provider that will route the external traffic to every node on the NodePort (ELB on AWS)

Services

- The option just shown only allow you to create **virtual IPs** or **Ports**
- There is also a possibilities to use **DNS Name**
  - **ExternalName** can provide a DNS name for the service
  - e.g for service discovery using DNS
  - This only works when the **DNS add-on** is enabled

Services

- This is an example of a Service definition (also created using kubectl expose):

```
-----------------------------------------
apiVersion: v1
kind: Service
metadata:
  name: helloworld-service
spec:
  ports:
  - port: 31001
    nodePort: 31001
    targetPort: nodejs-port
    protocol: TCP
  selector:
    app: helloworld
  type: NodePort
-----------------------------------------
```

- **Note:** By default service can run only between port 30000 - 32767, but you could change this behavior by adding the --service-node-port-range= argument to the kube-apiserver (in the init scripts)

Demo Placeholder

- A new service

Exposing Services