Extending Kubernetes LoadBalancer Using CRDs

IBM **Developer**

Wei Huang (IBM, wei.huang1@ibm.com)

Srini Brahmaroutu (IBM, srbrahma@us.ibm.com)

- sig-scheduling

- sig-testing
- sig-storage

Agenda

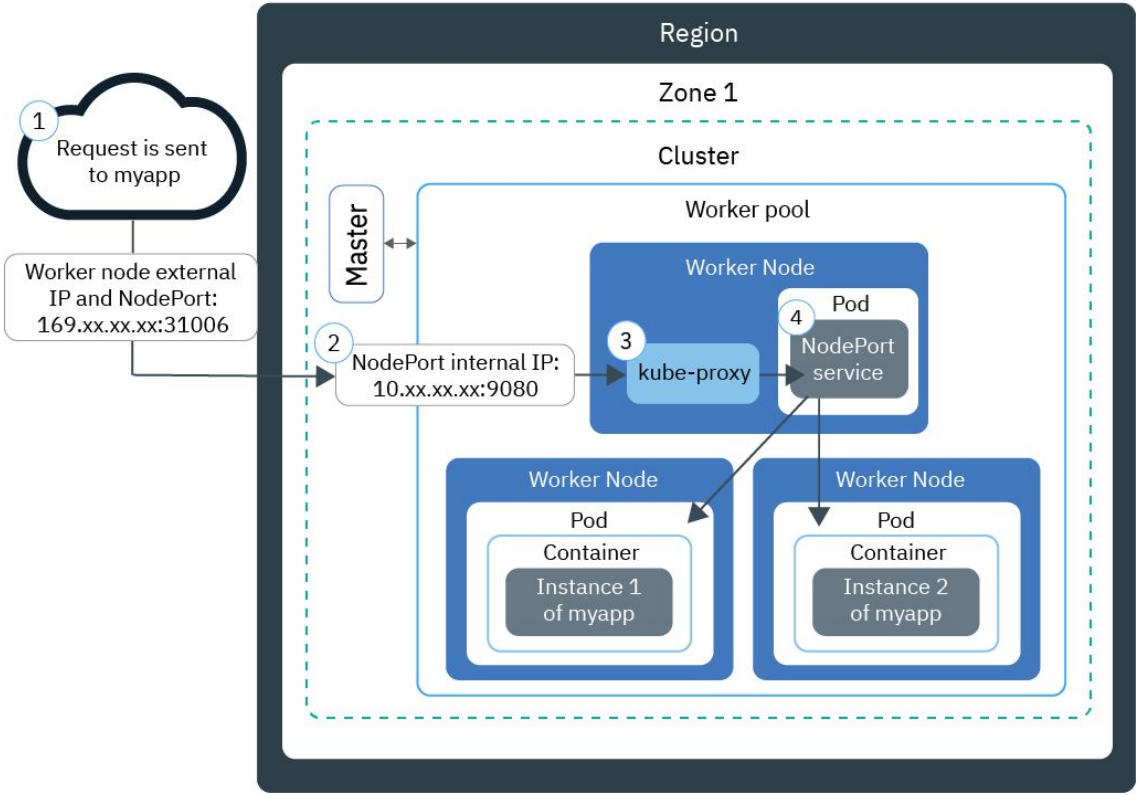
- How to Expose Kubernetes Workloads Externally
- Background / Motivations
- Shared LoadBalancer
- Demos
- Design / Implementation Details

How to Expose Kubernetes Workloads Externally

Kubernetes Basics - NodePort Service

NodePort

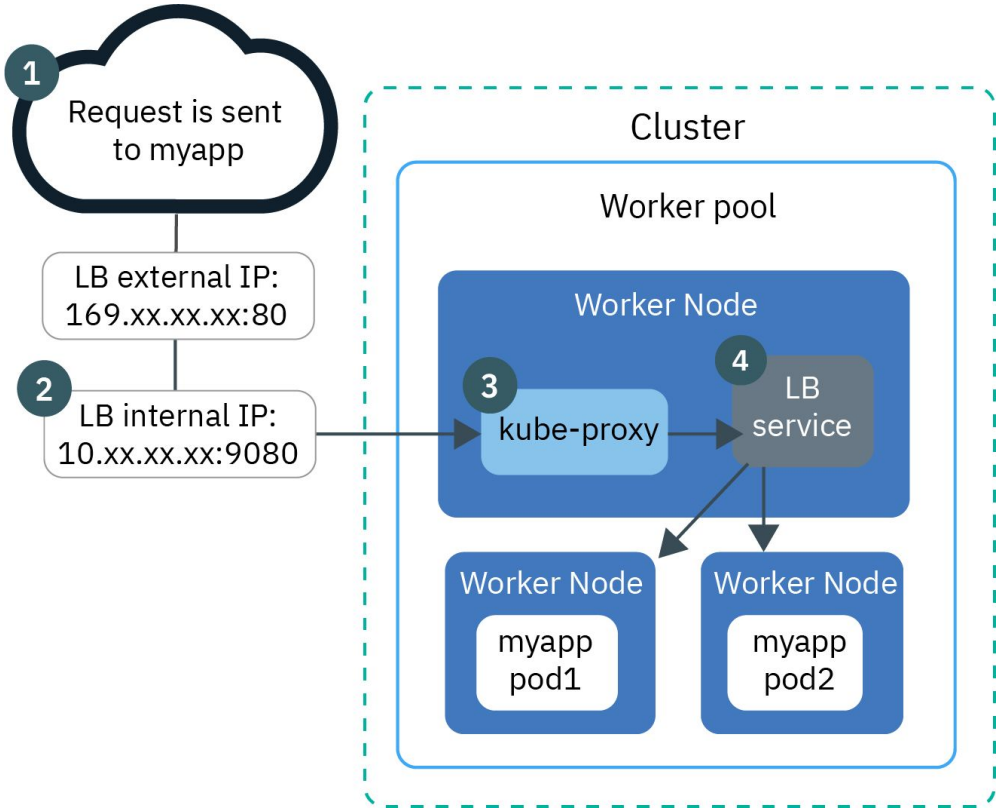
- Worker nodes needs to have a public/external IP
- Ports opened on *all* worker nodes
- Ports range from 30000 to 32767



Kubernetes Basics - LoadBalancer Service

LoadBalancer

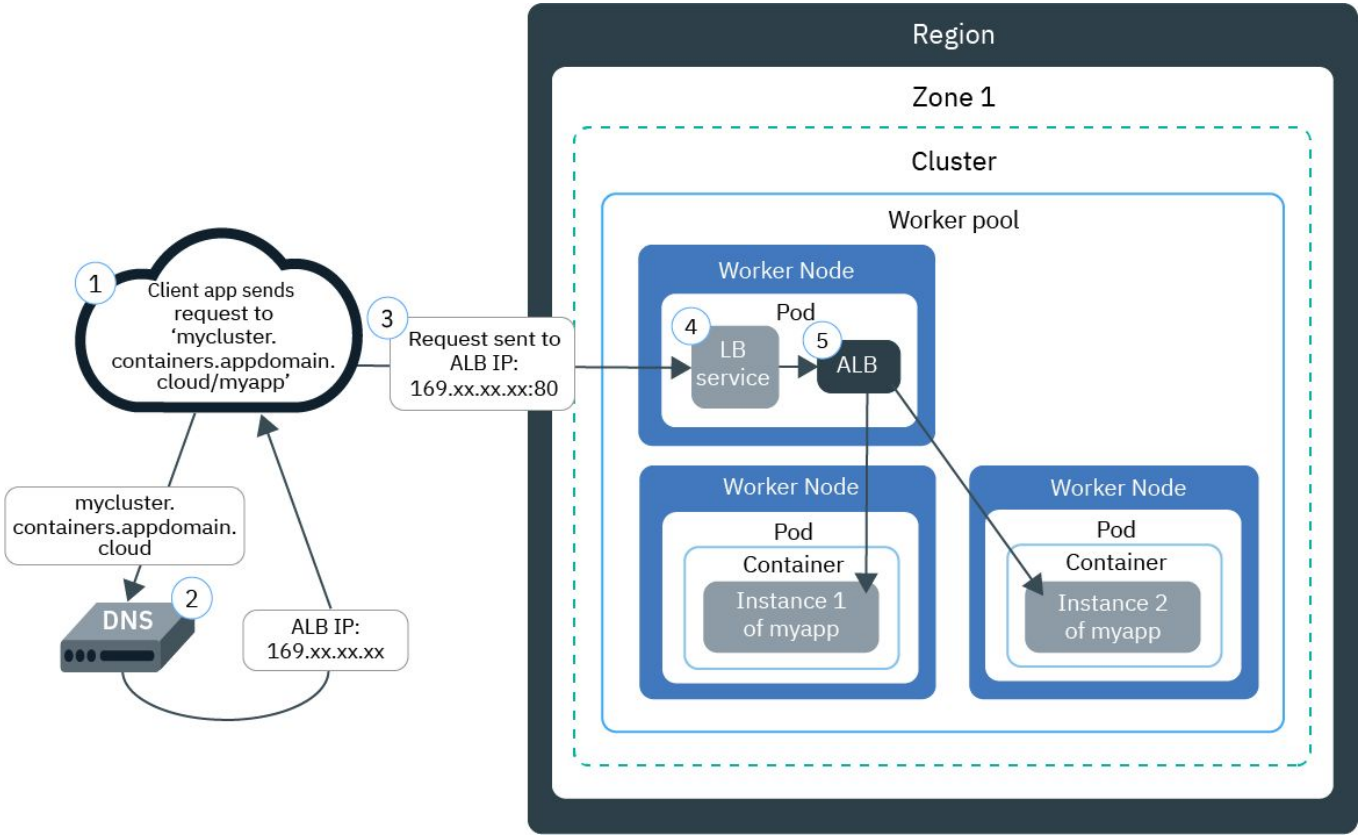
- EKS (Amazon)
- IKS (IBM)
- GKE (Google)
- AKS (Azure)
- ...



Kubernetes Basics - Ingress

Ingress

- L7
- Nginx/Envoy/...



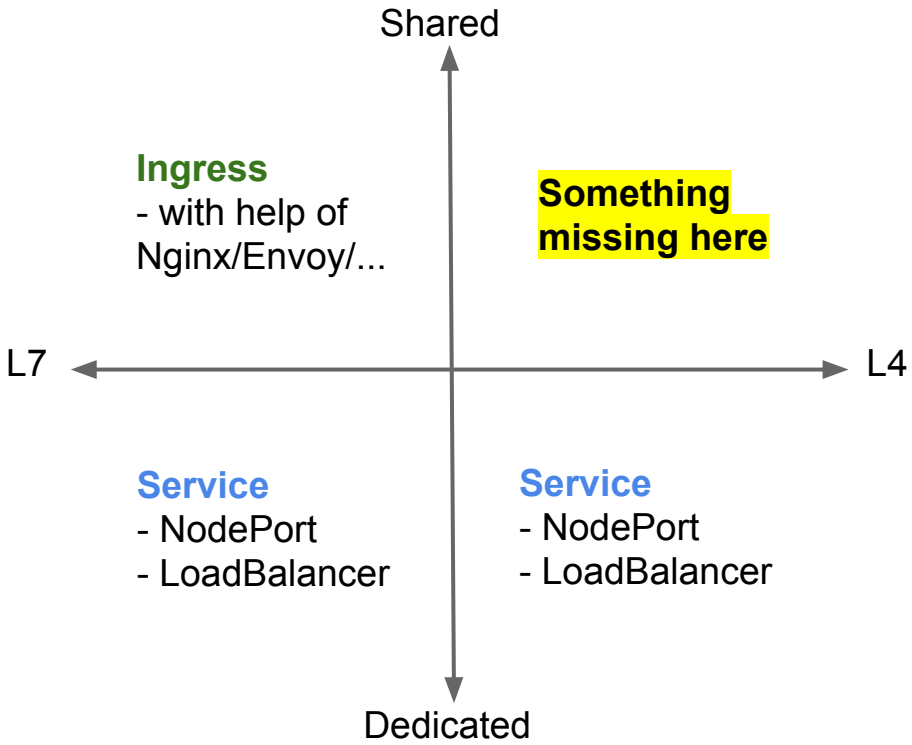
Ways to Expose K8s Apps Externally

Service - for both L4/L7 traffic

- Type NodePort
- Type LoadBalancer

Ingress - for L7 traffic

- Shared via ingress controller

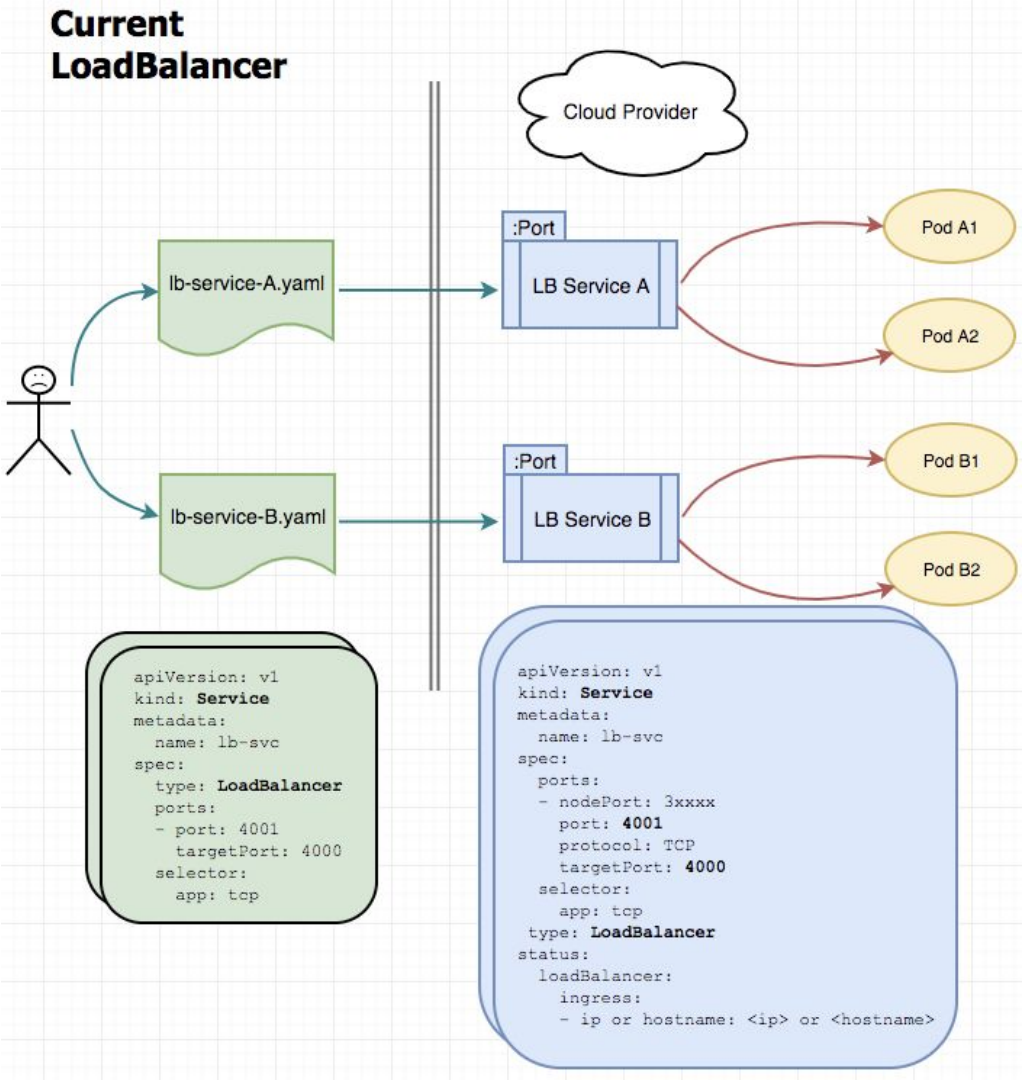


Background / Motivations

Background

An internal business requirement from an internal team.

- Two JDBC services (TCP)
- One data transferring service (UDP)
- One web console service (HTTP)

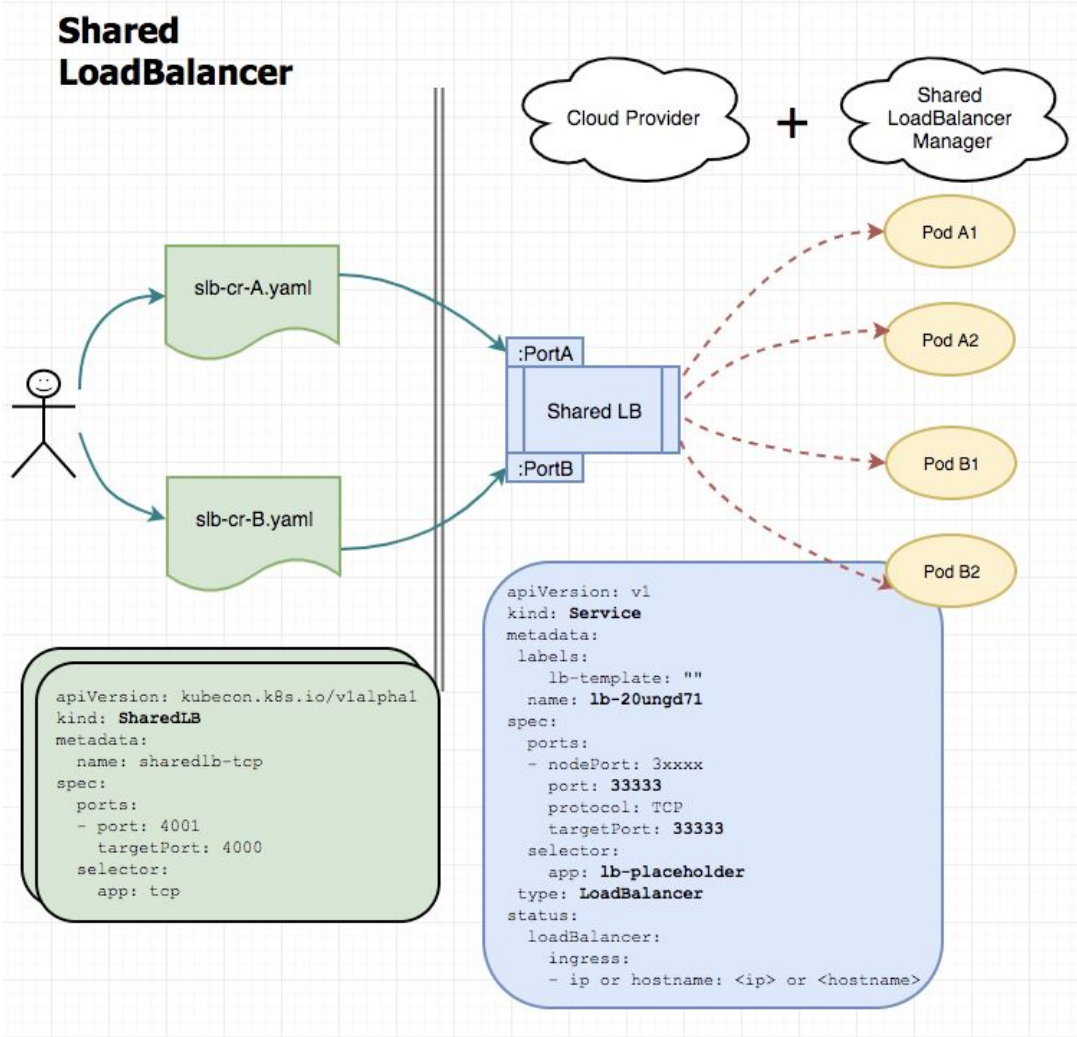


Expected Goal

```
⇒  k create -f crs
sharedlb.kubecon.k8s.io/sharedlb-tcp1 created
sharedlb.kubecon.k8s.io/sharedlb-tcp2 created
sharedlb.kubecon.k8s.io/sharedlb-tcp3 created
sharedlb.kubecon.k8s.io/sharedlb-tcp4 created
wei.huang1@wei-mbp:~/gospace/src/github.com/Huang-Wei/shared-loadbalancer
⇒  k get slb
NAME            EXTERNAL-IP      PORT   PROTOCOL   REF
sharedlb-tcp1   169.62.88.170    4001   TCP        default/lb-z5lrv7he
sharedlb-tcp2   169.62.88.170    4002   TCP        default/lb-z5lrv7he
sharedlb-tcp3   169.62.88.170    4003   TCP        default/lb-z5lrv7he
sharedlb-tcp4   169.62.88.170    4004   TCP        default/lb-z5lrv7he
```

Expected Goal (cont.)

```
➜  k create -f crs
sharedlb.kubecon.k8s.io/sharedlb-tcp1 created
sharedlb.kubecon.k8s.io/sharedlb-tcp2 created
sharedlb.kubecon.k8s.io/sharedlb-tcp3 created
sharedlb.kubecon.k8s.io/sharedlb-tcp4 created
wei.huang1@wei-mbp:~/gospace/src/github.com/Huang-Wei/shared-loadbalancer
➜  k get slb
NAME            EXTERNAL-IP      PORT   PROTOCOL   REF
sharedlb-tcp1   169.62.88.170    4001   TCP        default/lb-z5lrv7he
sharedlb-tcp2   169.62.88.170    4002   TCP        default/lb-z5lrv7he
sharedlb-tcp3   169.62.88.170    4003   TCP        default/lb-z5lrv7he
sharedlb-tcp4   169.62.88.170    4004   TCP        default/lb-z5lrv7he
```



Motivations

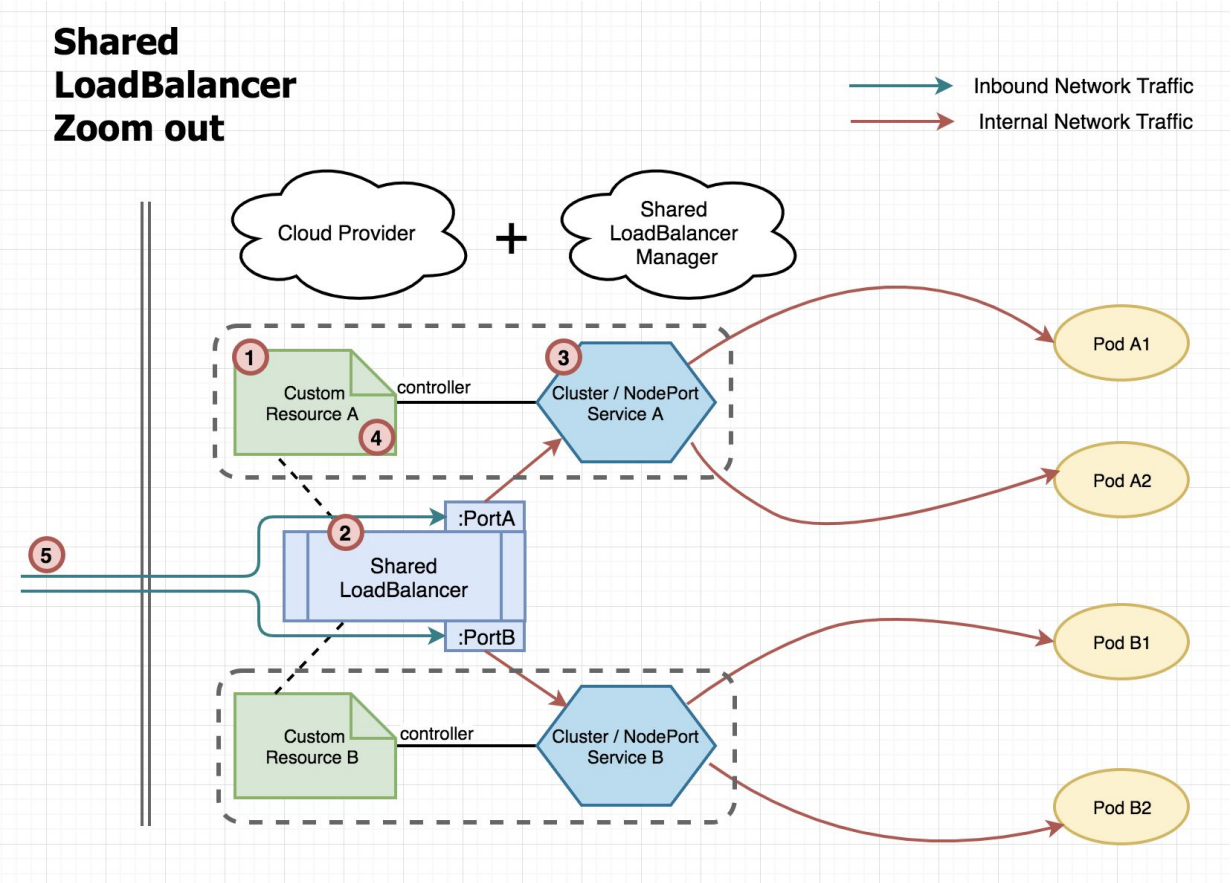
- Cost effective
- User friendly
- Reusing existing Kubernetes assets (don’t reinvent wheel)
- Minimum operation efforts
- Consistent with Kubernetes roadmap

Shared LoadBalancer (SLB)

Problem Analysis

1. How to open **additional** ports (and firewall rules) on the “Shared” LoadBalancer
2. How to **associate** the ports with backing pods
3. How to give **accessing info** back to end-user

Shared LoadBalancer Internals



Demos

Design / Implementation Details

Design Considerations

1. Using CRD as the facade to end-user, ~~instead of Service with annotation~~.
2. Namespaced CRD ~~vs. Clustered CRD~~.
3. Create real LB on demand~~, or prepare placeholder LBs in a pool~~.
4. Configure how many SharedLB CRs are mapped with one real LoadBalancer.
    a. It can be a static way like 100 SharedLB CRs => 1 real LoadBalancer.
    b. Or, can be done in a smart/dynamic way to take workload of each CR into consideration.
5. Support IKS, EKS, GKE and AKS

	EKS (Amazon)	IKS (IBM)	GKE (Google)	AKS (Azure)
Core Extension Solution	NodePort Service	Cluster Service with “externalIP”	NodePort Service	NodePort Service
SDK Authentication	aws_access_key_id aws_secret_access_key	APIKEY *(only needed when adding portable ip quota)*	oauth2 *(gcloud auth application-default login)*	Service principle and Role *(az ad sp create-for-rbac)*
Forward Rule Firewall Rule	Use SDK to operate	Auto Managed	Use SDK to operate	Use SDK to operate
Accessing method	<Hostname>:<Port>	<IP>:<Port>	<IP>:<Port>	<IP>:<Port>
Limitations	UDP not supported Latest version is 1.10	N/A	Incoming port limited to 30000~32767	N/A

Future Considerations

1. Get feedback (sig-cloudprovider, sig-network, internal & external users)
2. Support on more cloudproviders
3. Testings (unit tests, integration tests, e2e tests) and CI
4. Features (essentially it’s a scheduling problem)
    a. Ports (completed)
    b. Request/Limits
    c. LeastRequested vs. MostRequested
    d. {Anti}-Affinity, or Taint/Toleration
5. Thinking in Kubernetes

Thanks!

- github.com/Huang-Wei/shared-loadbalancer
- Github: @Huang-Wei / @brahmaroutu
- Slack: @Huang-Wei / @srbrahma
- Twitter: @hweicdl / @brahmaroutu

Backup: CRD Practices

1. Use CRD built-in features
    a. validation
    b. shortNames
    c. additionalPrinterColumns
    d. controllerRef
    e. finalizers
2. CRD controller
    a. kubebuilder
    b. internal cache
    c. reconcile upon IndexKey (namespace/name) of a changed object

Changes in User's View

Before/Input: **N** service yamls (with type LoadBalancer, and src/dst port info)

Before/Output: **N** publicly accessible {ip/hostname, port} pairs

After/Input: **N** custom resource yamls (w/ or w/o src port info)

After/Output: **1** publicly accessible {ip/hostname, (random) port} pair

(N is configurable)