

Exposing Containers



- `kubectl expose` creates a **service** for existing pods
- A **service** is a stable address for pod(s)
- If we want to connect to pod(s), we need a **service**
- CoreDNS allows us to resolve **services** by name
- There are different types of **services**
 - ClusterIP
 - NodePort
 - LoadBalancer
 - ExternalName



Basic Service Types

- **ClusterIP** (default)
 - Single, internal virtual IP allocated
 - Only reachable from within cluster (nodes and pods)
 - Pods can reach service on apps port number
- **NodePort**
 - High port allocated on each node
 - Port is open on every node's IP
 - Anyone can connect (if they can reach node)
 - Other pods need to be updated to this port
- **These services are always available in Kubernetes**



More Service Types

- **LoadBalancer**
 - Controls a LB endpoint external to the cluster
 - Only available when infra provider gives you a LB (AWS ELB, etc)
 - Creates NodePort+ClusterIP services, tells LB to send to NodePort
- **ExternalName**
 - Adds CNAME DNS record to CoreDNS only
 - Not used for Pods, but for giving pods a DNS name to use for something outside Kubernetes
- **Kubernetes Ingress:** We'll learn later



Creating a ClusterIP Service

- Open two shell windows so we can watch this
 - > `kubectl get pods -w`
- In second window, lets start a simple http server using sample code
 - > `kubectl create deployment httpenv --image=bretfisher/httpenv`
- Scale it to 5 replicas
 - > `kubectl scale deployment/httpenv --replicas=5`
- Let's create a ClusterIP service (default)
 - > `kubectl expose deployment/httpenv --port 8888`

Inspecting ClusterIP Service



- Look up what IP was allocated
 - > `kubectl get service`
- Remember this IP is cluster internal only, how do we curl it?
- If you're on Docker Desktop (Host OS is not container OS)
 - > `kubectl run --generator=run-pod/v1 tmp-shell --rm -it --image bretfisher/netshoot -- bash`
 - > `curl httpenv:8888`
- If you're on Linux host
 - > `curl [ip of service]:8888`

Cleanup



- Leave the deployment there, we'll use it in the next Lecture

Create a NodePort Service



- Let's expose a NodePort so we can access it via the host IP (including localhost on Windows/Linux/macOS)
 - > `kubectl expose deployment/httpenvv --port 8888 --name httpenvv-np --type NodePort`
- Did you know that a NodePort service also creates a ClusterIP?
- These three service types are additive, each one creates the ones above it:
 - ClusterIP
 - NodePort
 - LoadBalancer

Add a LoadBalancer Service



- If you're on Docker Desktop, it provides a built-in LoadBalancer that publishes the `--port` on localhost
 - > `kubectl expose deployment/httpenv --port 8888 --name httpenv-lb --type LoadBalancer`
 - > `curl localhost:8888`
- If you're on kubeadm, minikube, or microk8s
 - No built-in LB
 - You can still run the command, it'll just stay at "pending" (but its NodePort works)

Cleanup



- Let's remove the Services and Deployment
 - > `kubectl delete service/httpenv service/httpenv-np`
 - > `kubectl delete service/httpenv-lb deployment/httpenv`



Kubernetes Services DNS

- Starting with 1.11, internal DNS is provided by CoreDNS
- Like Swarm, this is DNS-Based Service Discovery
- So far we've been using hostnames to access Services
 - > `curl <hostname>`
- But that only works for Services in the same Namespace
 - > `kubectl get namespaces`
- Services also have a FQDN
 - > `curl <hostname>.<namespace>.svc.cluster.local`

Assignment: Explore run get and logs



- Dry Run

- > `kubectl create deployment nginx --image nginx --dry-run`

- Run does different things based on options

- > `kubectl create deployment nginx --image nginx --dry-run --port 80 --expose`

- Only create a simple Pod, not a Deployment, ReplicaSet, etc.

- > `kubectl run nginx-pod --generator=run-pod/v1 --image nginx`

- Get a shell in new Pod, remove on exit

- > `kubectl run shell --generator=run-pod/v1 --rm -it --image busybox`

Assignment: Explore run get and logs



- Create a Deployment and ClusterIP Service in one line
 - > `kubectl run nginx2 --image nginx --replicas 2`
- Get multiple resources in one line
 - > `kubectl get deploy,pods`
- Get all pods, in wide format (gives more info)
 - > `kubectl get pods -o wide`
- Get all pods and show labels
 - > `kubectl get pods --show-labels`
 -

Assignment: Explore run get and logs



- Better log viewing with stern
 - github.com/wercker/stern
 - > `kubectl run mydate --image bretfisher/date --replicas 3`
 - > `kubectl logs deployment/mydate`
 - > `stern mydate`

Cleanup



- Let's remove everything but the service/kubernetes
 - > kubectl get all
 - > kubectl delete deployment/nginx2 pod/nginx-pod