**SERVICES**

* Service provides a single point of entry for accessing one or more Pods
* Since Pods live and die, we cant rely on their IP, thats why we need Services
  + Pods are mortal and may only live a short time
  + You cant rely on a Pod IP address staying the same
  + Pods can be horizontally scaled so each Pod gets its own IP address
  + A Pod gets an IP address after it has been scheduled (no way for clients to know IP ahead of time)
* **A screenshot of a cell phone

  Description automatically generated**
* **A screenshot of a cell phone

  Description automatically generated**
* **A screenshot of a cell phone

  Description automatically generated**
* with browser is a little bit unique, they use the same connection over and over for the request to the server, that will be respected by Kubernetes and that one connection would keep hitting the same pod
* Services can be defined in different way:
  + **ClusterIP –** expose the service on cluster-internal IP (default)
  + **NodePort –** expose the service on each Node's IP at a static port
  + **LoadBalancer –** provision an external IP to act as load balancer for the service
  + **ExternalName –** maps a service to a DNS name
* **CLUSTERIP SERVICE**

**A screenshot of a cell phone

Description automatically generated**

* **NODEPORT SERVICE**
* This one is very useful for few reason. Number one, we could set this up so an external caller could call in and thats useful, but as a developer, oftentimes we might want to run some type of an external call for debugging purpose or maybe performance reasons. So knowing how we can proxy into a given pod using NodePort Service is really valuable

**A screenshot of a cell phone

Description automatically generated**

* **LOADBALANCER SERVICE**
* Load balancers are important, because as call comes in, some nodes are busier than others and we might want to route to different nodes based on the traffic

**A screenshot of a cell phone

Description automatically generated**

* **EXTERNALNAME SERVICE**
* This is just an alias. So we know a service sticks around and we can count on the service's IP address, but wat if really what we are afte ris we want to call some other domain, maybe within the company, maybe outside the company, or IP address, but lets say that domain or IP keeps changing on us and we now have to keep updating the containers in the pods that want to talk to it. This would allows us to define a service once that can then proxy request to this external service and that way if the external service DNS or IP address changes, we could just update this service. That way client stay uneffected. So in essence, the external service is just hiding the details from the actual cluster of what we are actually calling out to externally.

**A screenshot of a social media post

Description automatically generated**

* **kubectl port-forward service/service\_name 8080.**

**A screenshot of a cell phone

Description automatically generated**

* when the metadata:name property is set, then other services can call that service by name:port, fe. backend:80
* how can you quickly test if a Serviec and Pod is working? Use **kubectl exec** to shell into a Pod/Container.
* **kubectl exec pod-name –curl -s** [**http://podIP**](http://podIP)
* if curl not installed -> **kubectl exec pod-name -it sh, -> apk add curl,**
* **curl -s** [**http://podIP**](http://podIP)
* enter in pod via **kubectl exec pod\_name -it sh,** install curl with **apk add curl,** then run **curl** [**http://IP**](http://IP) **or** [**http://dns\_name:port**](http://dns_name:port)**,** afterwards, you should see nginx html content in your terminal which has been received by another service/pod => **this is related to CLUSTERIP type of service**

**A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated**