



kubernetes

A DEMO OF BASICS

by
Kishore Potla

Agenda

- Introduction – What is Kubernetes
- Kubernetes Cluster
- Kubernetes Components
- Kubernetes Cluster Configuration
- Kubectl basic commands
- Application Deployment
- Kubernetes pods
- Executing on Containers
- Kubernetes Services.

What is Kubernetes

- A highly available cluster of computers that are connected to work as a single unit
- Service for Container Cluster Management
- Kubernetes is an open-source platform by Google
- *K8s* is an abbreviation derived by replacing the 8 letters with “8” from Kubernetes
- Used to manage Docker containers as a default implementation

Kubernetes Cluster

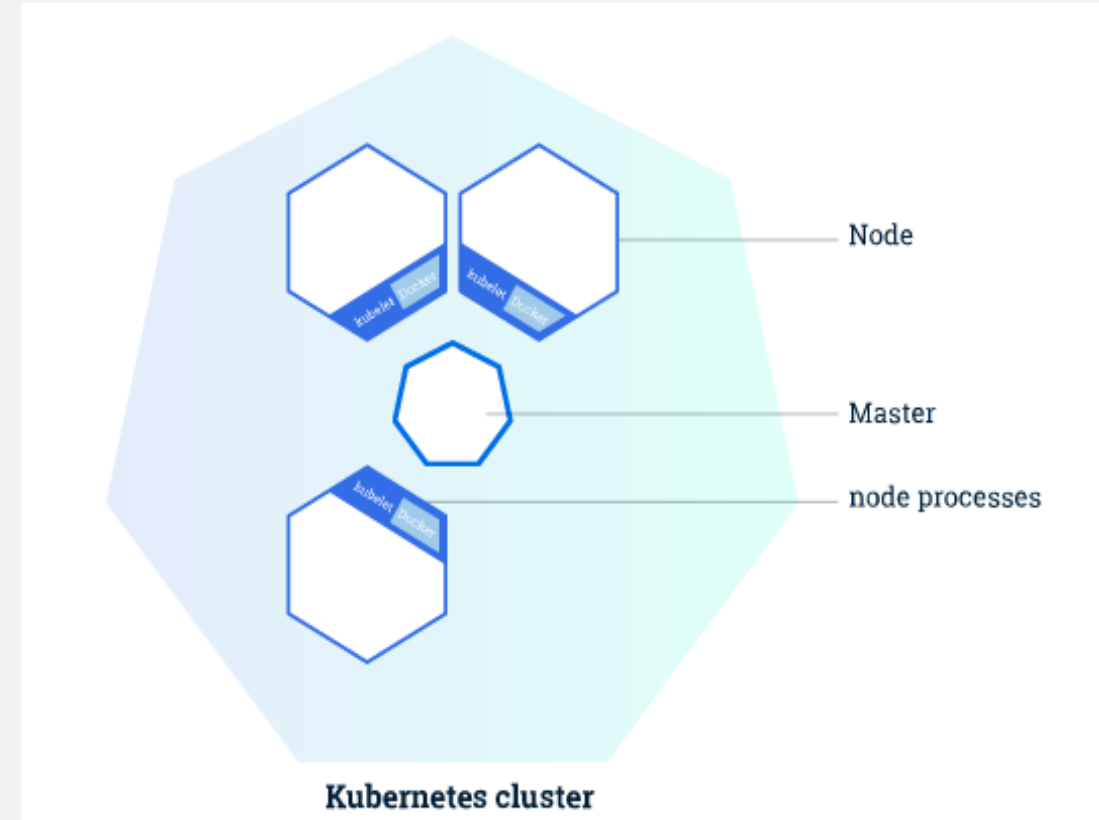
Kubernetes cluster consists of 2 types of resources

1. Master

- Master is responsible for managing the cluster
- Coordinates cluster activities like scheduling, maintaining, scaling of applications and rolling out new updates

2. Nodes

- Node is a VM or a physical computer that serves as a worker machine in Kubernetes cluster
- Each node has a **Kubelet**, which is an agent for managing the node and communicating with the Kubernetes master
- The nodes communicate with the master using the **Kubernetes API**, which the master exposes
- Master maintains the state of the Kubernetes in the **etcd** backend.



Kubernetes Components

Master Components

Master components provide the cluster's control plane

- **Kube-apiserver:** A component on the master that exposes the Kubernetes API. It is the front end for the Kubernetes control plane
- **Etc**d: Consistent and highly-available key value store used as Kubernetes' backing store for all cluster data
- **Kube-scheduler:** Component on the master that watches newly created pods that have no node assigned, and selects a node for them to run on
- **Kube-controller-manager:** Component on the master that runs controllers
 - Node Controller: Responsible for noticing and responding when nodes go down.
 - Replication Controller: Responsible for maintaining the correct number of pods
 - Endpoints Controller: Populates the Endpoints object (that is, joins Services & Pods).
 - Service Account & Token Controllers: Create default accounts and API access tokens for new namespaces
- **Cloud-controller-manager:** It runs controllers that interact with the underlying cloud providers.

Kubernetes Components (Conti..)

Node Components

Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment.

- **kubelet:** An agent that runs on each node in the cluster. It makes sure that containers are running in a pod.
- **Kube-proxy:** enables the Kubernetes service abstraction by maintaining network rules on the host and performing connection forwarding
- **Container Runtime:** The container runtime is the software that is responsible for running containers. Kubernetes supports several runtimes: [Docker](#), [rkt](#)

Kubernetes Cluster Configuration

Three different ways to configure a Kubernetes Cluster.

1. Minikube: Minikube is a tool that makes it easy to run Kubernetes **locally**. Minikube runs a **single-node** Kubernetes cluster inside a VM on your laptop for users looking to try out Kubernetes or develop with it day-to-day.

Installation Reference: <https://kubernetes.io/docs/getting-started-guides/minikube/>

Interactive tutorial using Minikube: <https://kubernetes.io/docs/tutorials/kubernetes-basics/>

2. Kubeadm: kubeadm is a toolkit that helps you bootstrap a best-practice Kubernetes cluster in an easy, reasonably secure and extensible way.

Installing Kubeadm: <https://kubernetes.io/docs/setup/independent/install-kubeadm/>

Creating cluster using Kubeadm: <https://kubernetes.io/docs/setup/independent/create-cluster-kubeadm/>

3. Creating custom cluster: In case of specific IaaS, networking, configuration management, or operating system requirements then this will provide an outline of the steps you need to take. Note that it requires considerably more effort than using one of the pre-defined guides.

Reference: <https://kubernetes.io/docs/getting-started-guides/scratch/>

Kubectl – basic commands

To interact with Kubernetes we'll use the command line interface, kubectl.

version: To check if kubectl is installed you can run the *kubectl version* command. we can see both the version of the client and as well as the server. The client version is the kubectl version; the server version is the Kubernetes version installed on the master

```
[root@ip-172-30-1-112:~# kubectl version
Client Version: version.Info{Major:"1", Minor:"10", GitVersion:"v1.10.1", GitCommit:"d4ab47518836c750f9949b9e0d387f20fb92260b", GitTreeState:"clean", BuildDate:"2018-04-12T14:26:04Z", GoVersion:"go1.9.3", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"10", GitVersion:"v1.10.1", GitCommit:"d4ab47518836c750f9949b9e0d387f20fb92260b", GitTreeState:"clean", BuildDate:"2018-04-12T14:14:26Z", GoVersion:"go1.9.3", Compiler:"gc", Platform:"linux/amd64"}
```

Cluster-info: To get cluster details.

```
[root@ip-172-30-1-112:~# kubectl cluster-info
Kubernetes master is running at https://172.30.1.112:6443
KubeDNS is running at https://172.30.1.112:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

Get nodes: Shows all nodes that can be used to host our applications

```
[root@ip-172-30-1-112:~# kubectl get nodes
```

| NAME | STATUS | ROLES | AGE | VERSION |
|-----------------|--------|--------|-----|---------|
| ip-172-30-0-105 | Ready | <none> | 22h | v1.10.1 |
| ip-172-30-0-75 | Ready | <none> | 22h | v1.10.1 |
| ip-172-30-1-112 | Ready | master | 1d | v1.10.1 |

Application Deployment

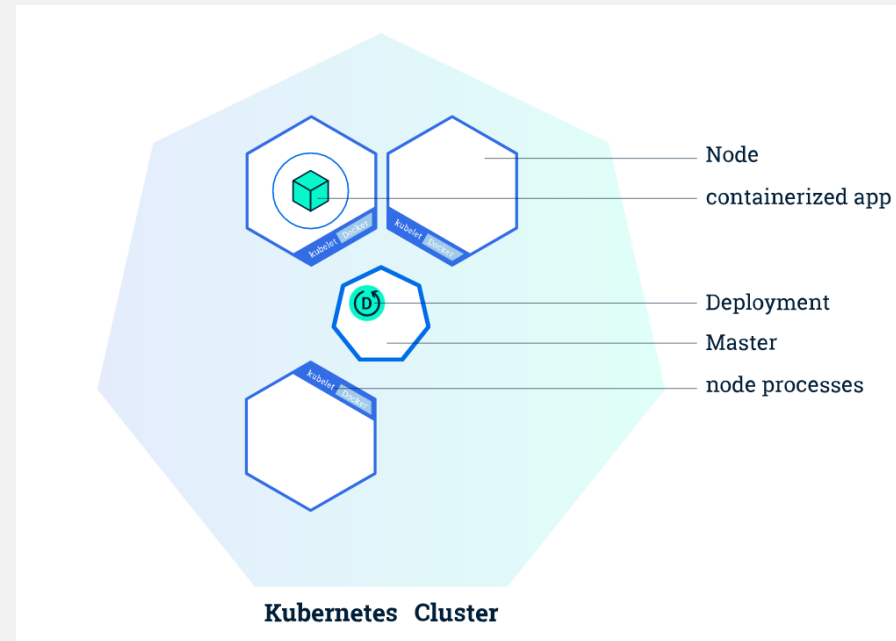
Kubernetes Deployments

- Create a Kubernetes **Deployment** configuration which instructs Kubernetes how to create and update instances of your application
- Kubernetes master schedules mentioned application instances onto individual Nodes in the cluster
- If any instance goes down or is deleted, the Deployment controller replaces it. This provides a **self-healing mechanism** to address machine failure or maintenance.

Deploying first app on Kubernetes

In deployment configuration file you need to mention:

- Container image for application
- Number of replicas
- Container information
- Labels and Selectors
- Controller



Application Deployment (Conti...)

Kubernetes Deployment Configuration file:

nginx-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

In this example:

- A Deployment named **nginx-deployment** is created, indicated by the ***metadata: name*** field.
- The Deployment creates **3 replicated Pods**, indicated by the replicas field.
- The selector field defines how the Deployment finds which Pods to manage. In this case, we simply select on one label defined in the Pod template (app: nginx).
- The Pod template's specification, or ***template: spec*** field, indicates that the Pods run one container, nginx, which runs the nginx Docker Hub image at version 1.7.9.
- The `template` field contains the following instructions:
 - The Pods are labeled `app: nginx`
 - Create one container and name it `nginx`.
 - Run the `nginx` image at version `1.7.9`.
 - Open port `80` so that the container can send and accept traffic.

Application Deployment (Conti...)

Kubectl Deployment commands:

- 1) **Create Deployment:** create command creates a new deployment. You need to provide the deployment configuration filename.
- 2) **Get Deployments:** “get deployments” command provides the details about the deployments
- 3) **Delete Deployment:** Deletes the deployment by name.

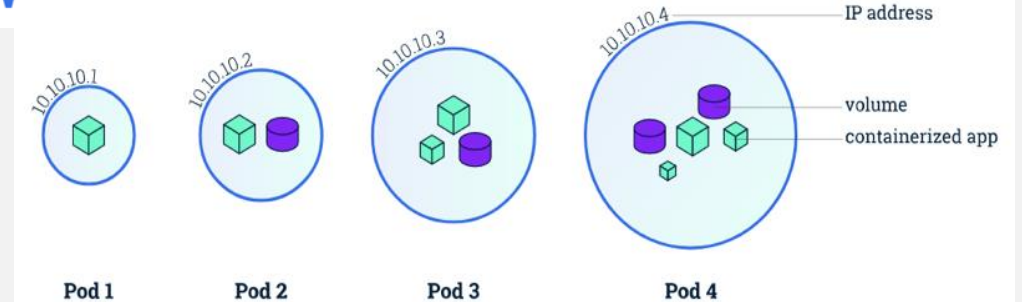
```
[root@ip-172-30-1-112:~#  
[root@ip-172-30-1-112:~# kubectl create -f Deployment_nginx.yaml  
deployment.apps "nginx-deployment" created  
[root@ip-172-30-1-112:~#  
[root@ip-172-30-1-112:~# kubectl get deployments  
NAME                DESIRED    CURRENT    UP-TO-DATE    AVAILABLE    AGE  
nginx-deployment    3          3          3             3            14s  
[root@ip-172-30-1-112:~#  
[root@ip-172-30-1-112:~#  
[root@ip-172-30-1-112:~#  
[root@ip-172-30-1-112:~# kubectl delete deployment nginx-deployment  
deployment.extensions "nginx-deployment" deleted  
[root@ip-172-30-1-112:~#
```

Kubernetes Pods

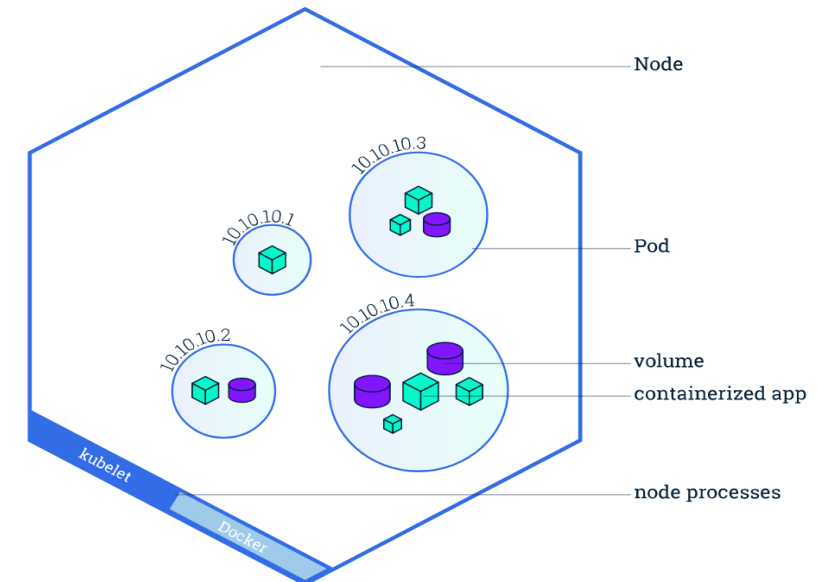
Pods:

- Deployment creates pod to host your application instance.
- A Pod is a Kubernetes abstraction that represents a group of one or more application containers and some shared resources for those containers.
- Those shared resources include:
 - Shared storage, as Volumes
 - Networking, as a unique cluster IP address
 - Information about how to run each container, such as the container image version or specific ports to use

Pods overview



Node overview



Every Kubernetes Node runs at least:

- **Kubelet** - A process responsible for communication between the Kubernetes Master and the Node; it manages the Pods and the containers running on a machine.
- **A container runtime** (like Docker, rkt) responsible for pulling the container image from a registry, unpacking the container, and running the application.

Kubernetes Pods (Conti...)

Pods commands:

kubectl get pods ↵

```
[root@ip-172-30-1-112:~# kubectl get pods
NAME                                READY    STATUS    RESTARTS   AGE
nginx-deployment-75675f5897-28jxm   1/1      Running   0           2m
nginx-deployment-75675f5897-qz698   1/1      Running   0           2m
nginx-deployment-75675f5897-wflwd   1/1      Running   0           2m
```

kubectl describe pods Command used to to view

- what containers are inside that Pod
- what images are used to build those containers.
- Also the details about the Pod's container: IP address, the ports used and a list of events related to the lifecycle of the Pod.

kubectl logs \$POD_NAME To view the container logs

Anything that the application would normally send to STDOUT becomes logs for the container within the Pod. We can retrieve these logs using the kubectl logs command

Executing Commands on Container

Kubectl exec: We can execute commands directly on the container once the Pod is up and running. For this, we use the exec command and use the name of the Pod as a parameter

```
[root@ip-172-30-1-112:~# kubectl exec nginx-deployment-75675f5897-28jxm env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=nginx-deployment-75675f5897-28jxm
KUBERNETES_SERVICE_PORT=443
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
KUBERNETES_SERVICE_HOST=10.96.0.1
NGINX_VERSION=1.7.9-1~wheezy
HOME=/root
```

List the environment variables:

Start a bash session in the Pod's container:

```
root@ip-172-30-1-112:~#
[root@ip-172-30-1-112:~# kubectl exec -ti nginx-deployment-75675f5897-28jxm bash
[root@nginx-deployment-75675f5897-28jxm:/#
[root@nginx-deployment-75675f5897-28jxm:/#
[root@nginx-deployment-75675f5897-28jxm:/# exit
exit
[root@ip-172-30-1-112:~# _
```

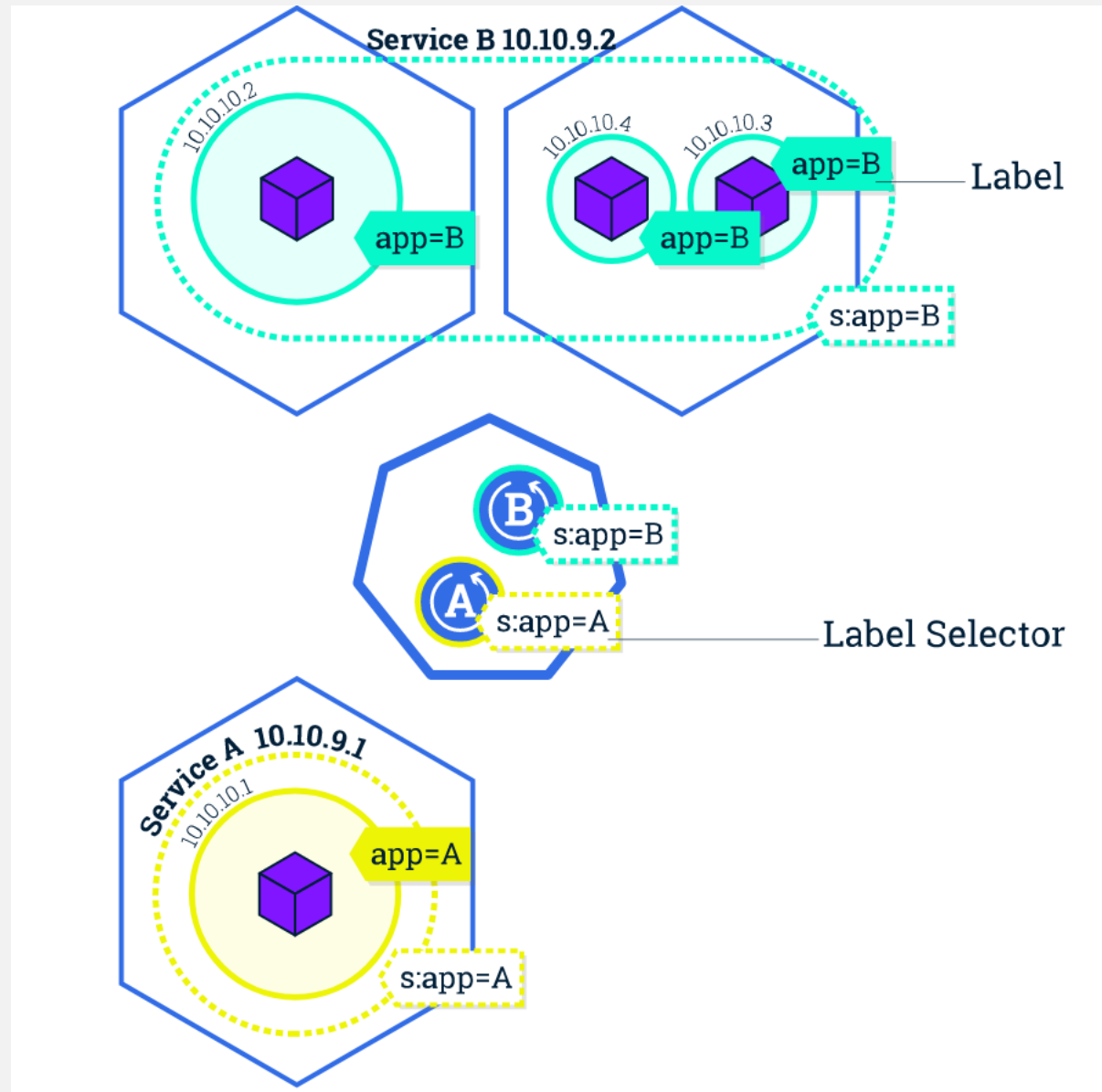
Exit from the container:

Kubernetes Services

Service in Kubernetes:

- A Service in Kubernetes is an abstraction which defines a **logical set of Pods and a policy** by which to access them
- Services enable a **loose coupling** between dependent Pods
- A Service is defined using YAML or JSON
- The set of Pods targeted by a Service is usually determined by a **LabelSelector**
- Although each Pod has a unique IP address, those **IPs are not exposed outside the cluster** without a Service
- Services can be exposed in different ways by specifying a **type** in the ServiceSpec:
 - ClusterIP** (default) - Exposes the Service on an internal IP in the cluster. This type makes the Service only reachable from within the cluster.
 - NodePort** - Exposes the Service on the same port of each selected Node in the cluster using NAT. Makes a Service accessible from outside the cluster using <NodeIP>:<NodePort>. Superset of ClusterIP.
 - LoadBalancer** - Creates an external load balancer in the current cloud (if supported) and assigns a fixed, external IP to the Service. Superset of NodePort.
 - ExternalName** - Exposes the Service using an arbitrary name (specified by externalName in the spec) by returning a CNAME record with the name. No proxy is used. This type requires v1.7 or higher of kube-dns.

Kubernetes Services (Conti...)



Kubernetes Services (Conti...)

Service example:

Kubectl expose command: To create and expose service.

```
[root@ip-172-30-1-112:~# kubectl expose -f Service_nginx.yaml
service "nginx-svc" exposed
```

kubectl get services: To list available services.

```
[root@ip-172-30-1-112:~# kubectl get services
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP  PORT(S)    AGE
kubernetes    ClusterIP     10.96.0.1     <none>       443/TCP    1d
nginx-svc     ClusterIP     10.102.224.24 <none>       8080/TCP   10s
[root@ip-172-30-1-112:~#
```


Kubectl describe service/<service name>: To find out what port was opened externally

```
[root@ip-172-30-1-112:~# kubectl describe services/nginx-svc
Name:          nginx-svc
Namespace:     default
Labels:        app=nginx
Annotations:    <none>
Selector:      app=nginx
Type:          ClusterIP
IP:            10.102.224.24
Port:          <unset> 8080/TCP
TargetPort:    8080/TCP
Endpoints:     192.168.226.70:8080,192.168.226.71:8080,192.168.72.7:8080
Session Affinity: None
Events:        <none>
root@ip-172-30-1-112:~# █
```

Service_nginx.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-svc

  labels:
    app: nginx
spec:
  selector:
    app: nginx
  ports:
    - port: 8080
      protocol: TCP
      nodePort: 30062
  type: LoadBalancer
```



Kubernetes Services (Conti...)

Using labels:

Query list of pods with label:

```
[root@ip-172-30-1-112:~# kubectl get pods -l app=nginx
NAME                                READY    STATUS    RESTARTS   AGE
nginx-deployment-75675f5897-28jxm   1/1      Running   0           2h
nginx-deployment-75675f5897-qz698   1/1      Running   0           2h
nginx-deployment-75675f5897-wflwd   1/1      Running   0           2h
```


Query list of services with label:

```
[root@ip-172-30-1-112:~# kubectl get services -l app=nginx
NAME      TYPE        CLUSTER-IP    EXTERNAL-IP  PORT(S)    AGE
nginx-svc ClusterIP    10.102.224.24 <none>       8080/TCP   11m
```

Service_nginx.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-svc

  labels:
    app: nginx
spec:
  selector:
    app: nginx
  ports:
    - port: 8080
      protocol: TCP
      nodePort: 30062
  type: LoadBalancer
```



THANK YOU

QUESTIONS?