Components in K8’s

Master

Worker

[Kubernetes Components | Kubernetes](https://kubernetes.io/docs/concepts/overview/components/)

**ETCD – key value store**

Key value store the stores the data in the form of documents or pages

**Controller manager**

controller is a process that continuously monitors the state of various components in system and works towards bringing the whole system to desired functioning state.

**Scheduler**

**Kube proxy**

**Yaml in Kubernetes**

[How to write YAML files for Kubernetes | Eskala](https://eskala.io/tutorial/how-to-write-yaml-files-for-kubernetes/)

**replica sets**

**replication controllers vs replica sets**

**deployments**

**services**

**labels and selectors**

**Scheduling**

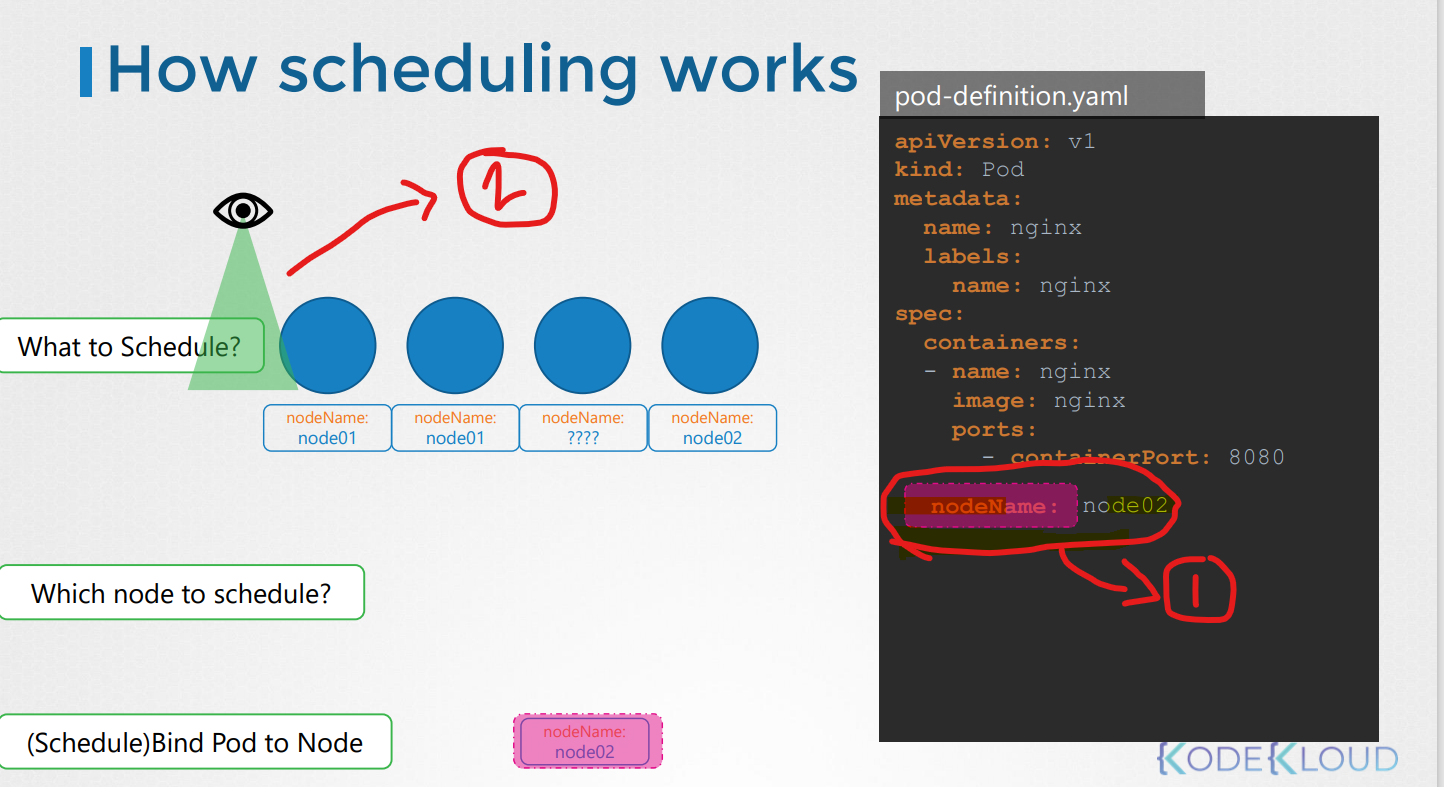
**Manual scheduling**

For an example let us assume we don’t have scheduler and we want to assign pods to node manually

, lets start with a simple pod definition file, every pod will have default nodename filed which is not set **(1),** we generally don’t specify the nodename field when we write pod manifest file instead k8’s adds it automatically.

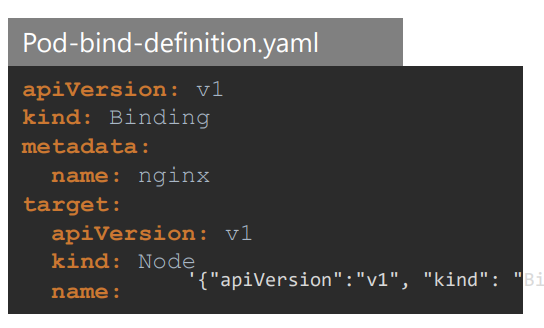
What is scheduler - The scheduler goes to all pods and looks for those that does not have this property set**(2)** , those are candidates for scheduling. Once it identifies it then identifies right node for pod by running an scheduling algorithm. Once identified it schedules pod on node by setting the node name property to the name of pod by creating bndign object**.**

So what is scheduler is not there in cluster , then all pods will stay in pending state. So here we can go for manually adding node name in pod manifest file while creating the pod.



We can only specify the node name while creating the pod name in pod manifest file.

If pod is already running then k8’s will not allow us to assign node to pod so another way to assign if pos is already running state is to create a binding object and send a post request to pods binding api.

****

**Labels and selectors :** Labels and selectors are standard method to group things together.

Labels are properties attached to each item for their class

Selectors help to filter these items.

**Kubectl label nodes <node-name> <label-key>:<label-value>**

**Eg – Kubectl label nodes node1 size=large**

Labels

Labels are key-value pairs which are attached to pods, replication controller and services. They are used as identifying attributes for objects such as pods and replication controller. They can be added to an object at creation time and can be added or modified at the run time.

Selectors

Labels do not provide uniqueness. In general, we can say many objects can carry the same labels. Labels selector are core grouping primitive in Kubernetes. They are used by the users to select a set of objects.

Kubernetes API currently supports two type of selectors −

* Equality-based selectors
* Set-based selectors

Equality-based Selectors

They allow filtering by key and value. Matching objects should satisfy all the specified labels.

Set-based Selectors

Set-based selectors allow filtering of keys according to a set of values.

K8’s objects use labels and selectors internally to connect different objects together. For example to create replica set consisting of three different pods , we first label the pod definition and use selector in replica set to group the pods.

ApiVersion: apps/v1

kind: ReplicaSet

metadata:

**labels: ---🡪1**

app: webapp

tier: frontend

name: webapp

spec:

replicas: 3

selector:

matchLabels:

tier: frontend

template:

metadata:

**labels: 🡪2**

tier: frontend

spec:

containers:

-

image: "webapp:2.0"

name: webapp

In replica set definition we will see labels defines in two places, 1 is for labels for replica set and 2 is for labels used for pods

**Taints and Tolerations**

Taints and tolerances are nothing to with the security and intrusion on the cluster instead these are used to set restrictions on what pod can scheduled on a node.

[Kubernetes - Taint and Toleration - GeeksforGeeks](https://www.geeksforgeeks.org/kubernetes-taint-and-toleration/)

Taints are set on node and tolerations are set on pod

There are three taint effects 1)noscheduler – which means pods will not schedule on node

2)PreferNoScheduler – will try to avoid placing a pod on node

3)noexecute – new pods will not scheduled on the node and if there any existing pods they will be removed if the do not tolerate

**Kubectl taint nodes node-name key-value:taint-effect**

Eg – Kubectl taint nodes node1 app-blue:NoScheduler

Spec:

Container:

-name: container1

Image:nginx

Tolerations:

-key:”app”

Operator:”equlal”

Value:”blue”

Effect:”noschedule”

**Node Selectors –** we can set limitations on pod to run only on particular nodes. There are two ways to do this , one is easiest way Node selector by adding nodeSelector property on pod definition file.

Nodeselector:

Size:Large (large is label which is already defined on node)

It has limitations such as we cannot keep conditions for complex situations. In above example it easy to select large or small but what if our pod re

**Node Affinity**

**Commands**

**Selector – kubectl get pods –selector app=app1**

**Taints -** Kubectl taint nodes node1 app-blue:NoScheduler