Kubernetes is **a portable, extensible, open source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation**. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available

**CLUSTER = GROUP OF MACHINES.**

WORKER NODE

WORKER NODE

WORKER NODE

MASTER

**This is Cluster**

**Q- What we installed in this cluster?**

1. **On Each Machine**
2. **Kubelet**
3. **Kubeadm**
4. **Kubeproxy**
5. **container run time**

**On Master Machine**

1. **Kubeadm init**

**Kubeadm** is a tool to create and manage Kubernetes clusters.

**kubectl** is the command-line tool for working with a Kubernetes cluster.

**Kubelet** An agent that runs on each [node](https://kubernetes.io/docs/concepts/architecture/nodes/) in the cluster. It makes sure that [containers](https://kubernetes.io/docs/concepts/containers/) are running in a [Pod](https://kubernetes.io/docs/concepts/workloads/pods/).

**Container Runtime** The container runtime is the software that is responsible for running containers.

**Minikube** is a tool that lets you run Kubernetes locally. minikube runs a single-node Kubernetes cluster on your personal computer.

* Kubectl run webserver --image nginix --->making first pod
* Kubectl get pods
* Kubectl get pods -o wide --->to show all information about pod
* Kubectl run deepak --image httpd
* Kubectl delete pod deepak
* Kubectl --->press double tab to get all options
* Kubectl describe pod webserver --->to describe a particular pod
* Kubectl describe nods NODE\_NAME --->describe a particular worker node
* Kubectl logs webserver --->to show log messages
* Kubectl cp /deepak.txt webserver:/
* Kubectl exec webserver --ls / --->to run a command in pod

**Objects – we call objects these things in k8s like->Pods, Replication Controller, Replica Set, Deployment, Services, Config Maps, etc.**

**There are two methods making Objects in k8s**

1. **Command Line (This command line work, we have done on previous pages)**
2. **Menifest File/Config. File (use .yaml syntax file, Ex. Deepak.yml)**

**4 Main properties of Menifest file**

1. **apiVersion:**
2. **kind:**
3. **metadata:**
4. **spec:**

**POD**

A Pod usually has a one-to-one relationship with containers running your application.

* To scale up, you create a pod, and to scale down, you delete a pod.
* You do not add additional containers to an existing POD to scale your application.

#### **Kubernetes doesn’t deploy containers directly on the worker node.**

* The containers are encapsulated into a Kubernetes object called POD.
* A POD is a single instance of an application.
* A POD is the smallest object that you can create in Kubernetes

**MAKING PODS THROUGH MENIFEST FILE**

1. **Making a simpl pod by manifest file**

* **vim pods.yml**

**apiVersion: v1**

**kind: Pod**

**metadata:**

**name: httpdwebserver**

**labels:**

**app: webapp**

**type: frontend**

**spec:**

**containers:**

**- name: xyz**

**image: centos/httpd**

**NOTE – THIS UPER FILE WILL MAKE ONLY SINGLE PODE.**

* kubectl create -f pods.yml
* kubectl describe pod httpdwebserver
* kubectl replace -f pods.yml --->if you change anything in pod.yml file
* kubectl delete -f pods.yml

OR

* kubectl delete pod httpdwebserver --->that we made through pod.yml file

**THIS POD pods.yml IS NOT HIGHLY AVAILABLE THAT’S WHY WE USE KUBE CONTROLLERS – REPLICATION CONTROLLER, REPLICA-SET, DEPLOYMENTS**

**WHAT THESE CONTROLLERS DO – EX.- IF YOU SET 5 PODS TO RUN THEN THEY WILL ALWAYS RUN IF A SINGLE POD GET DELETE THEN AUTOMATICALLY A NEW POD WILL BE GENERATE.**

1. **Replication Controller**

A ReplicationController ensures that a specified number of pod replicas are running at any one time. In other words, a ReplicationController makes sure that a pod or a homogeneous set of pods is always up and available.

How a ReplicationController Works

If there are too many pods, the ReplicationController terminates the extra pods. If there are too few, the ReplicationController starts more pods. Unlike manually created pods, the pods maintained by a ReplicationController are automatically replaced if they fail, are deleted, or are terminated. For example, your pods are re-created on a node after disruptive maintenance such as a kernel upgrade. For this reason, you should use a ReplicationController even if your application requires only a single pod. A ReplicationController is similar to a process supervisor, but instead of supervising individual processes on a single node, the ReplicationController supervises multiple pods across multiple nodes.

* **vim rc.yml**

**apiVersion: v1**

**kind: ReplicationController**

**metadata:**

**name: deepak**

**labels:**

**app: webapp**

**spec:**

**replicas: 5**

**template:**

**metadata:**

**name: sharma**

**labels:**

**app: webapp**

**spec:**

**containers:**

**- name: krishna**

**image: centos/httpd**

* Kubectl create -f rc.yml
* Kubectl get pods
* Kubectl delete rc deepak

OR

* kubectl delete -f rc.yml
* Kubectl get rc

**If you delete 1 pod from these 5 pods then automatically 1 pod will be generate.**

**If you make another pod from another menifest file then that pod also automatically get deleted because you set 5 pods in “rc.yml” file.**

**If labels match in two files then another files’s pods will be terminated. If labels are different then you can make both files pods.**

**IF YOU WANT TO SCALE UP PODS. THEN THERE ARE TWO METHODS.(2 & 3 are same)**

1. **EDIT IN MENIFEST FILE AND THEN REPLACE WITH OLD FILE.**
   * **Kubectl replace -f rc.yml**
2. **RUN THIS COMMAND**
   * **Kubectl scale rc --replicas=15 deepak**
3. **RUN THIS COMMAND**
   * **Kubectl scale --replicas=50 -f rc.yml**

**NOTE- IF YOU SCALEUP THROUGH COMMAND LINE THAT NUMBER WILL NOT UPDATE IN MENIFEST FILE.**

**IQ – DIFFERENCE BETWEEN REPLICATION CONTROLLER AND REPLICA SET?**

**ANS - Replica Set is the next generation of Replication Controller. Replication controller is kinda imperative, but replica sets try to be as declarative as possible.**

1. **Replica Set**

A ReplicaSet's purpose is to maintain a stable set of replica Pods running at any given time. As such, it is often used to guarantee the availability of a specified number of identical Pods.

## How a ReplicaSet works

A ReplicaSet is defined with **fields**, including a **selector** that specifies how to identify Pods it can acquire, a number of replicas indicating how many Pods it should be maintaining, and a pod template specifying the data of new Pods it should create to meet the number of replicas criteria. A ReplicaSet then fulfills its purpose by creating and deleting Pods as needed to reach the desired number. When a ReplicaSet needs to create new Pods, it uses its Pod template.

## When to use a ReplicaSet

A ReplicaSet ensures that a specified number of pod replicas are running at any given time. However, a Deployment is a higher-level concept that manages ReplicaSets and provides declarative updates to Pods along with a lot of other useful features. Therefore, we recommend using Deployments instead of directly using ReplicaSets, unless you require custom update orchestration or don't require updates at all.

This actually means that you may never need to manipulate ReplicaSet objects: use a Deployment instead, and define your application in the spec section.

### **Pod Selector**

The .spec.selector field is a [label selector](https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/). As discussed [earlier](https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/#how-a-replicaset-works) these are the labels used to identify potential Pods to acquire. In our frontend.yaml example, the selector was:

In the ReplicaSet, .spec.template.metadata.labels must match spec.selector, or it will be rejected by the API.

**Note:** For 2 ReplicaSets specifying the same .spec.selector but different .spec.template.metadata.labels and .spec.template.spec fields, each ReplicaSet ignores the Pods created by the other ReplicaSet.

### **Replicas**

You can specify how many Pods should run concurrently by setting .spec.replicas. The ReplicaSet will create/delete its Pods to match this number.

If you do not specify .spec.replicas, then it defaults to 1.

* **vim rs.yml**

**apiVersion: apps/v1**

**kind: ReplicaSet**

**metadata:**

**name: myrs**

**labels:**

**course: redhat**

**spec:**

**replicas: 10**

**selector:**

**matchLabels:**

**app: webapp**

**template:**

**metadata:**

**name: xyz**

**labels:**

**app: webapp**

**spec:**

**containers:**

**- name: lsd**

**image: centos/httpd**

* Kubectl create -f rs.yml
* Kubectl get pods
* Kubectl delete rs myrs

OR

* kubectl delete -f rs.yml
* Kubectl get rs

1. **DEPLOYMENTS**

Deployment provides rolling update and roll back facility.

A Deployment provides declarative updates for [Pods](https://kubernetes.io/docs/concepts/workloads/pods/) and [ReplicaSets](https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/).

You describe a desired state in a Deployment, and the Deployment [Controller](https://kubernetes.io/docs/concepts/architecture/controller/) changes the actual state to the desired state at a controlled rate. You can define Deployments to create new ReplicaSets, or to remove existing Deployments and adopt all their resources with new Deployments.

## Use Case

The following are typical use cases for Deployments:

* [Create a Deployment to rollout a ReplicaSet](https://kubernetes.io/docs/concepts/workloads/controllers/deployment/#creating-a-deployment). The ReplicaSet creates Pods in the background. Check the status of the rollout to see if it succeeds or not.
* [Declare the new state of the Pods](https://kubernetes.io/docs/concepts/workloads/controllers/deployment/#updating-a-deployment) by updating the PodTemplateSpec of the Deployment. A new ReplicaSet is created and the Deployment manages moving the Pods from the old ReplicaSet to the new one at a controlled rate. Each new ReplicaSet updates the revision of the Deployment.
* [Rollback to an earlier Deployment revision](https://kubernetes.io/docs/concepts/workloads/controllers/deployment/#rolling-back-a-deployment) if the current state of the Deployment is not stable. Each rollback updates the revision of the Deployment.
* [Scale up the Deployment to facilitate more load](https://kubernetes.io/docs/concepts/workloads/controllers/deployment/#scaling-a-deployment).
* [Pause the rollout of a Deployment](https://kubernetes.io/docs/concepts/workloads/controllers/deployment/#pausing-and-resuming-a-deployment) to apply multiple fixes to its PodTemplateSpec and then resume it to start a new rollout.
* [Use the status of the Deployment](https://kubernetes.io/docs/concepts/workloads/controllers/deployment/#deployment-status) as an indicator that a rollout has stuck.
* [Clean up older ReplicaSets](https://kubernetes.io/docs/concepts/workloads/controllers/deployment/#clean-up-policy) that you don't need anymore.
* **vim deployment.yml**

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: webapp**

**labels:**

**course: redhat**

**spec:**

**replicas: 30**

**selector:**

**matchLabels:**

**app: webapp**

**template:**

**metadata:**

**name: xyz**

**labels:**

**app: webapp**

**spec:**

**containers:**

**- name: lsd**

**image: rupeshsaini09/website**

**NOTE : Deployment apnea aap 1 Replica Set bna deta h**

* kubectl create -f deployment.yml
* kubectl get deployment
* kebectl get rs
* kubectl describe deployment webapp
* kubectl delete deployment webapp

OR

* kubectl delete -f deployment.yml

**Application ko update karne ki 2 streategies h**

* 1. **Recreate** (isme downtime face karna padta h ise lie hum ise use nhi karte)
  2. **Rolling Update** (isme hum Deployment use karte h)
* Kubectl rollout history deployment mywebapp --->ye revisions show karega
* Kubectl rollout undo deployment mywebapp --->previous version pe jane ke lie

**OR**

* **Kubectl rollout udo deployment mywebapp ---> specifie version pe jane ke lie**

**NOTE : Roll Update karne ke lie deployment file me bs desired/new image name dene ke baad “kubectl apply” command chla dijiye.**

1. **SERVICE**

An abstract way to expose an application running on a set of [Pods](https://kubernetes.io/docs/concepts/workloads/pods/) as a network service.With Kubernetes you don't need to modify your application to use an unfamiliar service discovery mechanism. Kubernetes gives Pods their own IP addresses and a single DNS name for a set of Pods, and can load-balance across them.

* 1. **Node Port**
  2. **Cluster Ip**
  3. **Load Balancer**
  4. **Node Port**

If you set the type field to NodePort, the Kubernetes control plane allocates a port from a range specified by --service-node-port-range flag (default: 30000-32767). Each node proxies that port (the same port number on every Node) into your Service. Your Service reports the allocated port in its .spec.ports[\*].nodePort field.

**NOTE : SERVICE SE PHLE DEPLOYMENT BANANA PADTA H**

* **Firstly run a Deployment file with containing website host Ex.- image: rupeshsaini09/website**
* vim svc.yml ---> making a service

apiVersion: v1

kind: Service

metadata:

name: mywebsite

labels:

app: website

spec:

type: NodePort

ports:

- targetPort: 80

port: 80

nodePort: 30080

selector:

app: webapp

* kubectl create -f svc.yml
* kubectl get svc

**Note : Ab hum website chlane ke lie kisi bhi worker node ka IP address use kr skte h**

* 1. **Cluster IP**

Cluster internally communicate with each other with single IP.

You can specify your own cluster IP address as part of a Service creation request. To do this, set the .spec.clusterIP field. For example, if you already have an existing DNS entry that you wish to reuse, or legacy systems that are configured for a specific IP address and difficult to re-configure.

The IP address that you choose must be a valid IPv4 or IPv6 address from within the service-cluster-ip-range CIDR range that is configured for the API server. If you try to create a Service with an invalid clusterIP address value, the API server will return a 422 HTTP status code to indicate that there's a problem.

* Firstly run a Deployment file with containing website host Ex.- image: rupeshsaini09/website
* **vim svc.yml**

**apiVersion: v1**

**kind: Service**

**metadata:**

**name: dbservice**

**labels:**

**app: website**

**spec:**

**type: ClusterIP**

**ports:**

**- targetPort: 80**

**port: 80**

**selector:**

**app: webapp**

* kubectl create -f svc.yml
* kubectl get service ---> Yhan aapko clustet IP show honge jinko aap within cluster hi use kr skte h browse pe nhi
  1. **Load Balancer**

Load balance ke lie kai algorithms h jinme se hum 3 Algorithms use kr rhe h request ko server pe forward krne ke lie.

1. **Round Robin(Mostly Used)** – isme request server pe one by one jayege phle 1st server pe fir 2nd pe for 3rd pe fir 1st pe then 2nd then 3rd esa chalta rhega.
2. **Random** – isme request randomly kisi bhi server pe ja skti h.
3. **Least Connection** – isme jis server pe sabse kam connection honge request uspe jayegi.

Sbse phle multiple nodes pe 1 website configure krenge. Deployment or Service ke through.

Fir is website ko Load Balancer ke through access krenge.

Load Balancer ke lie 1 ec2-instance banayenge or uspe load balancer config. Karenge.

**List Of Top Software Load Balancers**

1. Nginx
2. Avi Vantage Software Load Balancer
3. HAProxy
4. Kemp LoadMaster
5. Loadbalancer.org

Load Balance ke lie yhan hum HAPROXY Software use krenge

Installation

* yum install haproxy -y
* rpm -qi haproxy

HAPROXY ki config file **/etc/haproxy/haproxy.cfg**

Is file me setting kese kare

---> Fruntend main me “bind \*:5000” ki jagha 80 port number use karenge “bind \*:80”.

--->Fir Backend app me tino woker nodes machines ki entry karenge jese

Balance roundrobin

Server Hostname IP:PortNo check

Ex- Server servera 10.20.30.40:30080 check

# **Kubernetes Scheduler**

In Kubernetes, scheduling refers to making sure that [Pods](https://kubernetes.io/docs/concepts/workloads/pods/) are matched to [Nodes](https://kubernetes.io/docs/concepts/architecture/nodes/) so that [Kubelet](https://kubernetes.io/docs/reference/generated/kubelet) can run them.

## Scheduling overview

A scheduler watches for newly created Pods that have no Node assigned. For every Pod that the scheduler discovers, the scheduler becomes responsible for finding the best Node for that Pod to run on. The scheduler reaches this placement decision taking into account the scheduling principles described below.

If you want to understand why Pods are placed onto a particular Node, or if you're planning to implement a custom scheduler yourself, this page will help you learn about scheduling.

## kube-scheduler

[kube-scheduler](https://kubernetes.io/docs/reference/command-line-tools-reference/kube-scheduler/) is the default scheduler for Kubernetes and runs as part of the [control plane](https://kubernetes.io/docs/reference/glossary/?all=true#term-control-plane). kube-scheduler is designed so that, if you want and need to, you can write your own scheduling component and use that instead.

For every newly created pod or other unscheduled pods, kube-scheduler selects an optimal node for them to run on. However, every container in pods has different requirements for resources and every pod also has different requirements. Therefore, existing nodes need to be filtered according to the specific scheduling requirements.

In a cluster, Nodes that meet the scheduling requirements for a Pod are called feasible nodes. If none of the nodes are suitable, the pod remains unscheduled until the scheduler is able to place it.

The scheduler finds feasible Nodes for a Pod and then runs a set of functions to score the feasible Nodes and picks a Node with the highest score among the feasible ones to run the Pod. The scheduler then notifies the API server about this decision in a process called binding.

Factors that need to be taken into account for scheduling decisions include individual and collective resource requirements, hardware / software / policy constraints, affinity and anti-affinity specifications, data locality, inter-workload interference, and so on.

### **Node selection in kube-scheduler**

kube-scheduler selects a node for the pod in a 2-step operation:

1. Filtering
2. Scoring

The filtering step finds the set of Nodes where it's feasible to schedule the Pod. For example, the PodFitsResources filter checks whether a candidate Node has enough available resource to meet a Pod's specific resource requests. After this step, the node list contains any suitable Nodes; often, there will be more than one. If the list is empty, that Pod isn't (yet) schedulable.

In the scoring step, the scheduler ranks the remaining nodes to choose the most suitable Pod placement. The scheduler assigns a score to each Node that survived filtering, basing this score on the active scoring rules.

Finally, kube-scheduler assigns the Pod to the Node with the highest ranking. If there is more than one node with equal scores, kube-scheduler selects one of these at random.

There are two supported ways to configure the filtering and scoring behavior of the scheduler:

1. [Scheduling Policies](https://kubernetes.io/docs/reference/scheduling/policies) allow you to configure Predicates for filtering and Priorities for scoring.
2. [Scheduling Profiles](https://kubernetes.io/docs/reference/scheduling/config/#profiles) allow you to configure Plugins that implement different scheduling stages, including: QueueSort, Filter, Score, Bind, Reserve, Permit, and others. You can also configure the kube-scheduler to run different profiles.

**Default Scheduler**

Default Scheduler apne hisab se pod ko kisi bhi worker node pe run kr skta h.

**Manual Scheduling**

Manual Scheduling me hum apne hisab se apne pod ko node assign kra skte h deployment file me is line ko add kra ke “nodeName: servera.example.com”

* **vim deployment.yml**

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: webapp**

**labels:**

**course: redhat**

**spec:**

**replicas: 30**

**selector:**

**matchLabels:**

**app: webapp**

**template:**

**metadata:**

**name: xyz**

**labels:**

**app: webapp**

**spec:**

**containers:**

**- name: lsd**

**image: rupeshsaini09/website**

**nodeName: servera.example.com**

**NAME SPACE**

# **Namespaces**

In Kubernetes, namespaces provides a mechanism for isolating groups of resources within a single cluster. Names of resources need to be unique within a namespace, but not across namespaces. Namespace-based scoping is applicable only for namespaced objects (e.g. Deployments, Services, etc) and not for cluster-wide objects (e.g. StorageClass, Nodes, PersistentVolumes, etc).

## When to Use Multiple Namespaces

Namespaces are intended for use in environments with many users spread across multiple teams, or projects. For clusters with a few to tens of users, you should not need to create or think about namespaces at all. Start using namespaces when you need the features they provide.

Namespaces provide a scope for names. Names of resources need to be unique within a namespace, but not across namespaces. Namespaces cannot be nested inside one another and each Kubernetes resource can only be in one namespace.

Namespaces are a way to divide cluster resources between multiple users (via [resource quota](https://kubernetes.io/docs/concepts/policy/resource-quotas/)).

It is not necessary to use multiple namespaces to separate slightly different resources, such as different versions of the same software: use [labels](https://kubernetes.io/docs/concepts/overview/working-with-objects/labels) to distinguish resources within the same namespace.

* Kubectl get namespace --->to show name spaces
* Kubectl get ns --->to show name spaces
* Kubectl get pods ---> ye only mete name spaces ke pods show krega by default
* Kubectl get pods –namespace=kube-system --->kisi particular namespace ke pods kekhne ke lie
* Kubectl create -f pods.yml --namespace=kube-system --->kisi particular namespace me pod bnane ke lie. Ab “kubectl get pods” command run krne se new pod vhan show nhi hoga ab vo “kube-system” namespace me show hoga.
* Kubectl delete pod PODNAME --namespace=kube-system
* Kubectl create namespace deepak --->to create a namespace

**Hum manifest file bnate time bhi namespace define kr skte h jis namespace me hme pods banvane h. “namespace: deepak” metadata section me .**

* Kubectl config set-context $(kubectl config current-context) –namespace=deepak --->permanentally namespace save krne ke lie iske bad sare pods isi namespace me banenge.
* Kubectl config get-contexts --->default namespace check krne ke lie.

**MENIFEST FILE SE NAMESPACE BANANA**

* vim ns.yml

apiVersion: v1

kind: Namespace

metadata:

name: deepak

* kubectl create -f ns.yml
* kubectl get ns
* kubectl delete ns deepak
* kubectl get pods--all-namespaces --->it will show all pods from all namespaces

**IMPERATIVE APPROACH and DECLARATIVE APPROACH**

इंपरेटिव में ईच एंड एवरी स्टेप हम खुद कॉन्फ़िगर करते हैं, हम बताते ह सिस्टम को की उसे करना क्या ह। जैसे हम डेफिनिशन फाइल बनाते ह और उसमे हर एक स्टेप डिफाइन करते ह की उसमे काम कैसे करना ह, और फिर उस फाइल से हम काम करते ह मतलब रिसोर्सेज क्रिएट करते ह kubectl create command से।

Declarative way में हम अपनी requirment डिफाइन करते ह और सिस्टम अपने according स्कैन कर लेता ह की उसे करना क्या है।

अगर हम kubectl create command चलाएंगे तो फाइल के अकॉर्डिंग जो भी कोड लिखा हुआ ह उसके अकॉर्डिंग रिसोर्स क्रिएट होगा। और अगर वो रिसोर्स आलरेडी क्रिएटेड होगा तो हमारे सामने error आ जाएगी।

वहीं अगर kubectl apply command रन करते ह किसी फाइल के साथ तो अगर वो रिसोर्स आलरेडी क्रिएटेड नहीं ह तो वो बन जायेगा और अगर आलरेडी क्रिएटेड ह तो एरर नहीं आएगी, अगर हमने फाइल में कुछ चेंज किया ह तो वो अपडेट हो जाएगी।

kubectl apply command अपने आप चेक कर लेती ह की वो रिसोर्स बनाना ह या उसमे कुछ चेंज करना ह।

Imperative में हमें every स्टेप बताना होता ह जबकि Declaritive में वो काम अपने आप हो जाता ह।

**IMPERATIVE APPROACH**

Isme hme car wale ko rasta hme btana hota h jese left, right to home.

* kubectl create -f pods.yml --->example of imperative approach

**DECLARATIVE APPROACH**

Isme hum cab book kr lete h hme rasta btane ki jarurat nhi hoti.

Examples:- Ansible, Cheff, Puppet, Terraform

* kubectl apply -f pods.yml --->example of declarative approach

**IQ – DIFFERENCE BETWEEN kubectl create and kubectl apply COMMAND?**

12-04-22

**Taint and Toleration**

Jab hum chahte h ki hamare kuch specific pods kisi 1 particular worker node pe deploy ho to hum taint and toleration use karte h.

Mana ki hume kisi worker node “A” pe pods banane h to hum us worker node pe taint laga denge or pod banate time pod ki definition me tolerate karva denge to ab jo bhi A machine ke taint ko tolerate karega pod usi pe jake Banega.

Tolerations are applied to pods, and allow (but do not require) the pods to schedule onto nodes with matching taints.

Taints and tolerations work together to ensure that pods are not scheduled onto inappropriate nodes. One or more taints are applied to a node; this marks that the node should not accept any pods that do not tolerate the taints.

**Taint** = Taint lagta h machine/worker noder pr.

**Toleration** = Toleration lagta h pod par.

* Kubectl describe nodes serverb.example.com ---> you can see taint detail in these details.

Agar abhi aap deployment me bina tolerate lagaye pod banayenge to vo “serverb.example.com” ko chod kar baki sabhi worker node pe hi banenge.

* Kubectl taint node serverb.example.com app=database:NoExecute --->worker node pe taint lagana
* vim deployment.yml

apiVersion: apps/v1

kind: Deployment

metadata:

name: webapp

labels:

course: redhat

spec:

replicas: 50

selector:

matchLabels:

app: webapp

template:

metadata:

name: xyz

labels:

app: webapp

spec:

containers:

- name: lsd

image: rupeshsaini09/website

tolerations:

- key: "app"

operator: "Equal"

value: "database"

effect: "NoExecute"

* kubectl apply -f deployment.yml

**HOW TO REMOVE TENT/UNTENT**

* Kubectl taint node serverb.example.com app=database:NoExecute- (there is – in the last)

NOTE – Ab kuch pod servera pe kuch serverb pe or kuch serverc pe launch honge. Agar serverb pe taint tolerate nhi hota to serverb pe 1 bhi pod nhi banta.

Machine me Taint is lie lagate h ki koi bhi unnecessary pod us machin pe na bane jispe taint lagaya h. Agar hum chahte h ki hamare pod taint vali machin pe launch ho to hum deployment file me tolerate laga denge. Iska ye matlab nhi h ki pod baki machine pe nhi banenge, vo vhan bhi ban skte h.

**IQ – WHEN YOU MAKE PODS THEN WHY THEY DON’T DEPLOY ON MASTER NODE?**

**ANS – BECAUSE THERE IS ALREADY A “NoSchedule” TENT IMPLEMENTED THAT’S WHY NO POD DEPLOY ON MASTER NODE. AND THIS IS AUTOMATICALLY APPLIED WHEN WE SETUP CLUSTER.**

**NODE SELECTOR**

nodeSelector is the simplest recommended form of node selection constraint. You can add the nodeSelector field to your Pod specification and specify the [node labels](https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node/#built-in-node-labels) you want the target node to have. Kubernetes only schedules the Pod onto nodes that have each of the labels you specify.

**Node selector ke lie phle node pe label lagana hoga fir deployment me node selector lagana hoga.**

* Kubectl label nodes servera.example.com machine=large ---> to set a label on worker node
* vim deployment.yml

apiVersion: apps/v1

kind: Deployment

metadata:

name: webapp

labels:

course: redhat

spec:

replicas: 50

selector:

matchLabels:

app: webapp

template:

metadata:

name: xyz

labels:

app: webapp

spec:

containers:

- name: lsd

image: rupeshsaini09/website

nodeSelector:

machine: large

**DAEMON SET**

DaemonSet make sure karta h ki har 1 node pe 1 pod up and running rahe. DaemonSet ko hum 1 example se samajh skte h.

Ex:- mana ki hamare pas 60 servers h or hum sabhi ke log msgs padne h to hum kese padenge. Islie hum 1 Log server config karenge or 1-1 pod sabhi nodes pe chla denge jise hme sare server machines ke log msgs 1 single machine pe show ho jaye.

A *DaemonSet* ensures that all (or some) Nodes run a copy of a Pod. As nodes are added to the cluster, Pods are added to them. As nodes are removed from the cluster, those Pods are garbage collected. Deleting a DaemonSet will clean up the Pods it created.

Some typical uses of a DaemonSet are:

* running a cluster storage daemon on every node
* running a logs collection daemon on every node
* running a node monitoring daemon on every node

In a simple case, one DaemonSet, covering all nodes, would be used for each type of daemon. A more complex setup might use multiple DaemonSets for a single type of daemon, but with different flags and/or different memory and cpu requests for different hardware types.

**Making a DaemonSet**

* vim daemonset.yml

apiVersion: apps/v1

kind: DaemonSet

metadata:

name: metrics-agent

spec:

selector:

matchLabels:

app: metrics

template:

metadata:

name: xyz

labels:

app: metrics

spec:

containers:

- name: lsd

image: httpd

* kubectl apply -f daemonset.yml

NOTE:- Is DaemonSet se har 1 node pe 1 pod baj jayega. Or ager hum same cluster me or nodes banate h to unpe bhi automatically ye same single pod chal jayega.

**STATIC POD**

* Agar aapki master node or baki worker note band ho jaye or aapke paas only 1 single worker node h to aap pod kese banayenge?
* Iske lie hum worker node ki **/etc/Kubernetes/manifest/pod.yml**  yhan is pod.yml file me pod banane ki enty kr denge. Jisse hi vim file me entry karake save karenge automatically pod ban jayenge.
* Kubelet regularly motitor karta rhata h is location ko ki yhan koi new definition file h ki nhi pod banane ke lie.jese hi ise kuch new file milegi vo pod bana dega.
* Yhan kubectl command kam nhi karegi is lie hum yhan “docker container ls” command use karenge.

### **Requests and Limits**

256 MIB RAM

.5 CPU

Requests and limits are the mechanisms Kubernetes uses to control resources such as CPU and memory. Requests are what the container is guaranteed to get. If a container requests a resource, Kubernetes will only schedule it on a node that can give it that resource. **Limits, on the other hand**, make sure a container never goes above a certain value. The container is only allowed to go up to the limit, and then it is restricted.

It is important to remember that the limit can never be lower than the request. If you try this, Kubernetes will throw an error and won’t let you run the container.

* vim pods.yml

apiVersion: v1

kind: Pod

metadata:

name: webserver

labels:

app: webapp

type: frontend

spec:

containers:

- name: xyz

image: httpd

resources:

requests:

memory:"4G"

cpu: 4

limits:

memory:”2G”

cpu:2

* kubectl apply -f pods.yml

NOTE: JAB AAP ISE DESCRIBE KARENGE TO EVENTS ME ISKI DETAILS DEKH SAKTE H.

**SIDECAR CONTAINER/ HELPER CONTAINER**

Sidecar containers are **containers that are needed to run alongside the main container**. The two containers share resources like pod storage and network interfaces. The sidecar containers can also share storage volumes with the main containers, allowing the main containers to access the data in the sidecars.

* vim deployment.yml

apiVersion: v1

kind: Pod

metadata:

name: webserver

labels:

app: webapp

type: frontend

spec:

containers:

- name: xyz

image: httpd

- name: abc

image: nginx

**CONFIG MAPS**

**Environment Variable Set karne ke lie** Config Maps use karte h.

A ConfigMap is **an API object that lets you store configuration for other objects to use**. Unlike most Kubernetes objects that have a spec , a ConfigMap has data and binaryData fields. These fields accept key-value pairs as their values.

A ConfigMap is an API object used to store non-confidential data in key-value pairs. [Pods](https://kubernetes.io/docs/concepts/workloads/pods/) can consume ConfigMaps as environment variables, command-line arguments, or as configuration files in a [volume](https://kubernetes.io/docs/concepts/storage/volumes/).

A ConfigMap allows you to decouple environment-specific configuration from your [container images](https://kubernetes.io/docs/reference/glossary/?all=true#term-image), so that your applications are easily portable.

**Caution:** ConfigMap does not provide secrecy or encryption. If the data you want to store are confidential, use a [**Secret**](https://kubernetes.io/docs/concepts/configuration/secret/) rather than a **ConfigMap**, or use additional (third party) tools to keep your data private.

**NOTE: ConfigMap me hum environments me hum password plain text me save karte h.**

**TWO WAYS OF MAKING CONFIGMAPING**

* 1. **Command line Config Maping**
  + Kubectl create configmap deepak --from-literal name=deepak --from-literal course=linux ---> ConfigMap Creation through command line
  + Kubectl get configmaps --->to list configmaps
  + Kubectl describe configmaps deepak --->to describe configmap
  1. **Menifast File se Config Maping**
* vim myconfigmap.yml

apiVersion: v1

kind: ConfigMap

metadata:

name: wordpress

data:

WORDPRESS\_DB\_HOST: "1.1.1.1"

WORDPRESS\_DB\_USER: deepak

WORDPRESS\_DB\_PASSWORD: deepak@sharma.com

WORDPRESS\_DB\_NAME: sharmadeepak

NAME: "deepak sharma"

MOB: "4582675239"

* + kubectl create -f myconfigmap.yml
  + kubectl get configmaps
  + Kubectl describe configmaps wordpress --->to describe configmap

**NOTE: Config maping karne ke baad hum pod se variable hata denge or pod kuch is tarah se config karenge.**

* + vim pods.yml

apiVersion: v1

kind: Pod

metadata:

name: webserver

labels:

app: webapp

type: frontend

spec:

containers:

- name: xyz

image: wordpress

envFrom:

- configMapRef:

name: wordpress

* + kubectl create -f pods.yml
  + kubectl get configmaps

**SECRETS**

A Secret is an object that contains a small amount of sensitive data such as a password, a token, or a key. Such information might otherwise be put in a [Pod](https://kubernetes.io/docs/concepts/workloads/pods/) specification or in a [container image](https://kubernetes.io/docs/reference/glossary/?all=true#term-image). Using a Secret means that you don't need to include confidential data in your application code.

Secrets are similar to [ConfigMaps](https://kubernetes.io/docs/concepts/configuration/configmap/" \o "" \t "_blank) but are specifically intended to hold confidential data.

* vim mysecret.yml

apiVersion: v1

kind: Secret

metadata:

name: dbcreate

data:

MYSQL\_ROOT\_PASSWORD: this password will be in encrpted form(encryption method explained bellow)

**Password encryption method**

* echo “deepak” | base64 ---> aap apna password deepak ki jagha kuch bhi de sakte h or jo output milega use password ki jagha dal sakte h.
* kubectl create -f mysecret.yml
* kubectl get secrets
* kubectl describe secrets dbcreate --->yhan aapko password ki value nhi show hogi
* kubectl get secrets dbcreate -o yml --->yhan aapko value show hogi lekin fir bhi actual password show nhi hoga.

Actual Password dekhne ke lie upar vali command se password ki value copy kr ke is command ke sath run kara denge.

* echo “Encrypted value” | base64 --decode ---> you will get your actual password.

**INIT CONTAINERS**

This page provides an overview of init containers: specialized containers that run before app containers in a [Pod](https://kubernetes.io/docs/concepts/workloads/pods/). Init containers can contain utilities or setup scripts not present in an app image.

You can specify init containers in the Pod specification alongside the containers array (which describes app containers).

## Understanding init containers

A [Pod](https://kubernetes.io/docs/concepts/workloads/pods/) can have multiple containers running apps within it, but it can also have one or more init containers, which are run before the app containers are started.

Init containers are exactly like regular containers, except:

* Init containers always run to completion.
* Each init container must complete successfully before the next one starts.

If a Pod's init container fails, the kubelet repeatedly restarts that init container until it succeeds. However, if the Pod has a restartPolicy of Never, and an init container fails during startup of that Pod, Kubernetes treats the overall Pod as failed.

To specify an init container for a Pod, add the initContainers field into the [Pod specification](https://kubernetes.io/docs/reference/kubernetes-api/workload-resources/pod-v1/#PodSpec), as an array of container items (similar to the app containers field and its contents). See [Container](https://kubernetes.io/docs/reference/kubernetes-api/workload-resources/pod-v1/#Container) in the API reference for more details.

The status of the init containers is returned in .status.initContainerStatuses field as an array of the container statuses (similar to the .status.containerStatuses field).

**STORAGE**

**PV = Persistent Volume**

**PVC = Persistent Volume Claim**

* vim pv.yml

apiVersion: v1

kind: PersistentVolume

metadata:

name: mypv

spec:

accessMode:

- ReadWriteMany

capacity:

storage: 1G

hostpath:

path: /tmp/data

* kubectl create -f pv.yml

**INGRESS**

An API object that manages external access to the services in a cluster, typically HTTP.

Ingress may provide load balancing, SSL termination and name-based virtual hosting

**Ingress Controller** = Load Balancer

**Ingress Resources** = Rules

Ingress Resources ke rules Ingress Controller se match honge.

Ingress Controller pure cluster me hi deploy hota h.

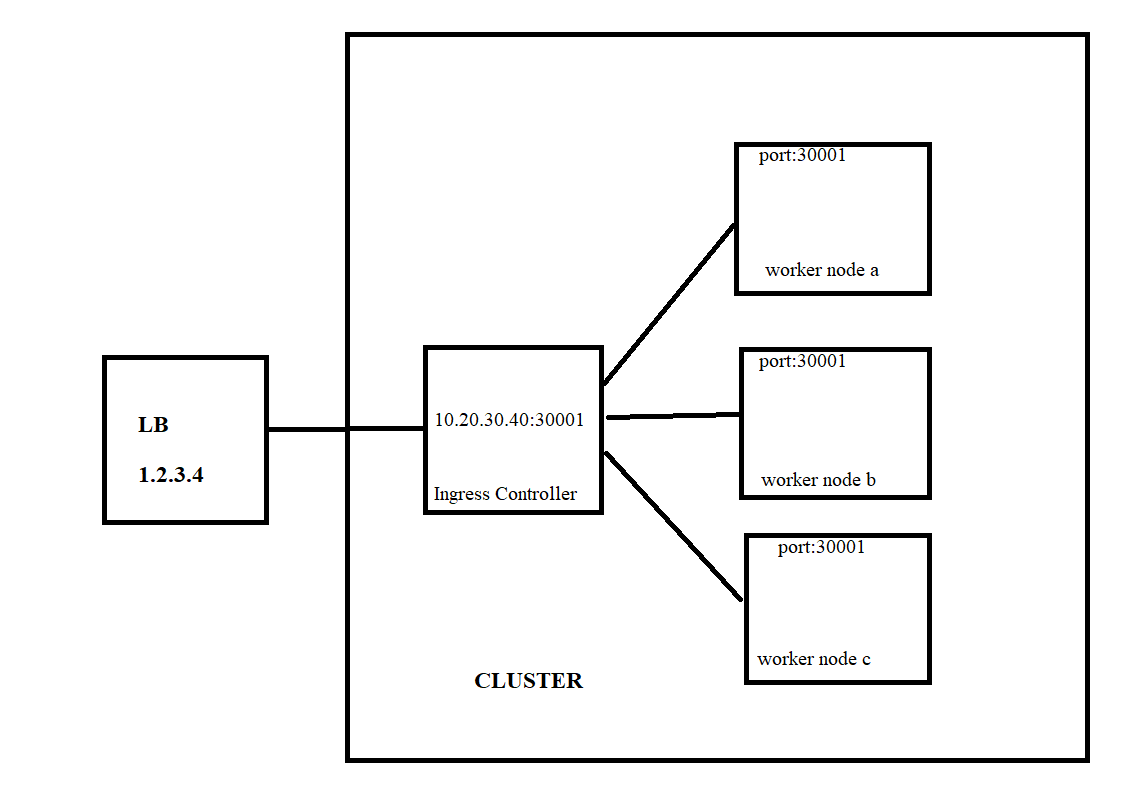
Ingress is an K8s object

Ingress controller 1 tarike se load balancer hi h

Ingress Controler aapko ye provide karvata h.

* virtual hosting
* ssl termination
* path based routing

Ingress Controller her 1 worker node ke kisi na kisi pod se maped h.



Search on google “top 5 ingress controller”

We will use nginix controller