### [Kubernetes](https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/) is an open-source, production ready system for container orchestration – i.e. automating deployment, scaling, and management of containerized applications. Developed by Google, after Borg and Omega. K is open source whereas the other two are google internal container management systems.

Kubernetes clusters : Kubernetes coordinates a highly available cluster of computers that are connected to work as a single unit. Kubernetes allow you to deploy containerized applications to a cluster without tying them specifically to individual machines. Kubernetes automates the distribution and scheduling of application containers across a cluster in a more efficient way.

A K cluster consists of two types of resources (1) A **master** coordinates the cluster and (2) the **Nodes**, which are the worker which runs the applications.

The Master is responsible for managing the cluster. The master coordinates all activities in your cluster, such as scheduling applications, maintaining applications' desired state, scaling applications, and rolling out new updates. A node is a VM or a physical computer that serves as a worker machine in a Kubernetes cluster. Each node has a **Kubelet**, which is an agent for managing the node and communicating with the Kubernetes master. The node should also have tools for handling container operations, such as [Docker](https://www.docker.com/) or [rkt](https://coreos.com/rkt/)

. A Kubernetes cluster that handles production traffic should have a minimum of three nodes. Masters manage the cluster and the nodes are used to host the running applications.

[When you deploy applications on Kubernetes, you tell the master to start the application containers. The master schedules the containers to run on the cluster's nodes. The nodes communicate with the master using the Kubernetes API, which the master exposes. End users can also use the Kubernetes API directly to interact with the cluster.](https://coreos.com/rkt/)

[Minikube is a lightweight Kubernetes implementation that creates a VM on your local machine and deploys a simple cluster containing only one node.](https://coreos.com/rkt/)

[*minikube version*](https://coreos.com/rkt/)

[*minikube start*](https://coreos.com/rkt/)

[*kubectl version*](https://coreos.com/rkt/)

[*kubectl get nodes*](https://coreos.com/rkt/)

[*kubectl cluster-info*](https://coreos.com/rkt/)

[To do the deployment of the containerized app on the cluster, you create a Kubernetes Deployment configuration. The Deployment instructs Kubernetes how to create and update instances of your application. Once you've created a Deployment, the Kubernetes master schedules mentioned application instances onto individual Nodes in the cluster.](https://coreos.com/rkt/)

[Once the application instances are created, a Kubernetes Deployment Controller continuously monitors those instances. If the Node hosting an instance goes down or is deleted, the Deployment controller replaces it. This provides a self-healing mechanism to address machine failure or maintenance.](https://coreos.com/rkt/)

[*kubectl run kubernetes-bootcamp --image=docker.io/jocatalin/kubernetes-bootcamp:v1 --port=8080*](https://coreos.com/rkt/)

[*kubectl get deployments*](https://coreos.com/rkt/)

[*kubectl proxy*](https://coreos.com/rkt/)

[*On another terminal:*](https://coreos.com/rkt/)

[*export POD\_NAME=$(kubectl get pods -o go-template --template '{{range .items}}{{.metadata.name}}{{"\n"}}{{end}}') echo Name of the Pod: $POD\_NAME*](https://coreos.com/rkt/)

[*curl* [*http://localhost:8001/api/v1/proxy/namespaces/default/pods/$POD\_NAME/*](http://localhost:8001/api/v1/proxy/namespaces/default/pods/$POD_NAME/)](https://coreos.com/rkt/)

[A kubernetes Pod : When we deploy an app, it creates a Pod. Pods are the atomic unit in the kubernetes platform and represents a group of one or more application containers, and the shared resources like (1) shared storage as Volumes (2) shared network, as a unique cluster IP address and (3) info about how to run each container, as image version and ports to use. A Pod can contain multiple container, each share the same IP address and port space and run in shared context in the same node. Each Pod is tied to a node where it’s scheduled. Containers should be scheduled together in a single POD if they are tightly coupled and share the same context (need to share resources such as disk).](http://localhost:8001/api/v1/proxy/namespaces/default/pods/$POD_NAME/)

[Nodes: A Pod always runs on a node. A node is a worker machine, can be either a VM or physical. A node will have at least (1) kubelet (2) a Container runtime, like docker or rkt.](http://localhost:8001/api/v1/proxy/namespaces/default/pods/$POD_NAME/)

[*kubectl action resource 🡪 command format.*](http://localhost:8001/api/v1/proxy/namespaces/default/pods/$POD_NAME/)

[*kubectl get 🡪 list resources*](http://localhost:8001/api/v1/proxy/namespaces/default/pods/$POD_NAME/)

[*kubectl describe 🡪 show detailed info about a resource*](http://localhost:8001/api/v1/proxy/namespaces/default/pods/$POD_NAME/)

[*kubectl logs 🡪 show logs from a container from a Pod.*](http://localhost:8001/api/v1/proxy/namespaces/default/pods/$POD_NAME/)

[*kubectl exec 🡪 execute a command on a container in a Pod.*](http://localhost:8001/api/v1/proxy/namespaces/default/pods/$POD_NAME/)

[Kubernetes [Pods](https://kubernetes.io/docs/concepts/workloads/pods/pod-overview/) are mortal. Pods in fact have a [lifecycle](https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/). When a worker node dies, the Pods running on the Node are also lost. A [ReplicationController](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller) might then dynamically drive the cluster back to desired state via creation of new Pods to keep your application running.](http://localhost:8001/api/v1/proxy/namespaces/default/pods/$POD_NAME/)

[A service is used to expose your application. Service defines a logical group of Pods and the policy to access them. Service enables loose coupling between dependent Pods. A service is defined using YAML or JSON. A set of Pods targeted by a service is defined by using a Label and Selector, and the services are exposed in different ways by specifying a type in the spec. Labels are key/value pairs attached to objects.](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[ClusterIP (default) - Exposes the Service on an internal IP in the cluster. This type makes the Service only reachable from within the cluster.](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[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 :. Superset of ClusterIP.](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[LoadBalancer - Creates an external load balancer in the current cloud (if supported) and assigns a fixed, external IP to the Service. Superset of NodePort.](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[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.](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[*kubectl get services*](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[*kubectl expose deployment/kubernetes-bootcamp --type="NodePort" --port 8080*](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[*kubectl get services*](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[*kubectl describe services/kubernetes-bootcamp*](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[*export NODE\_PORT=$(kubectl get services/kubernetes-bootcamp -o go-template='{{(index .spec.ports 0).nodePort}}') echo NODE\_PORT=$NODE\_PORT*](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[*curl host01:$NODE\_PORT*](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[*kubectl describe deploymen*](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[*kubectl get pods -l run=kubernetes-bootcamp (The deployment will have the default label “run=<deployment name>”*](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[*kubectl get services -l run=kubernetes-bootcamp (Similarly the service also has the default label same as the deployment)*](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[*export POD\_NAME=$(kubectl get pods -o go-template --template '{{range .items}}{{.metadata.name}}{{"\n"}}{{end}}') echo Name of the Pod: $POD\_NAME*](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[*kubectl label pod $POD\_NAME app=v1*](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[*kubectl describe pods $POD\_NAME*](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[*kubectl get pods -l app=v1*](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[*curl host01:$NODE\_PORT 🡪 This will show the output*](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[*kubectl delete service -l run=kubernetes-bootcamp*](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[*curl host01:$NODE\_PORT 🡪 this will show error since the service has been now removed and the application is no longer exposed.*](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[*kubectl get services 🡪 will show the service is deleted*](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[*kubectl exec -ti $POD\_NAME curl localhost:8080 🡪 This will work.*](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[When the traffic increases, we will need to scale the application to keep up with the user demand. Scaling is accomplished by changing the replicas in a deployment. Scaling up will create new Pods, and scaling down will destroy some of the Pods. Scaling to zero is possible. Running multiple instances will require a way to distribute the traffic and the integrated load balancer within the service will ensure this. It will keep monitoring the running Pod endpoints so that the traffic is routed only to the available Pods.](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[kubectl get deployments](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[The DESIRED state is showing the configured number of replicas](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[The CURRENT state show how many replicas are running now](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[The UP-TO-DATE is the number of replicas that were updated to match the desired (configured) state](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[The AVAILABLE state shows how many replicas are actually AVAILABLE to the users](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[kubectl scale deployments/kubernetes-bootcamp --replicas=4](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[kubectl get pods -o wide](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[kubectl describe deployments/kubernetes-bootcamp](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[kubectl describe services/kubernetes-bootcamp](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[export NODE\_PORT=$(kubectl get services/kubernetes-bootcamp -o go-template='{{(index .spec.ports 0).nodePort}}') echo NODE\_PORT=$NODE\_PORT](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[curl host01:$NODE\_PORT 🡪 we will see that every time we enter this command, the different Pods are getting invoked in a RR fashion.](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[kubectl scale deployments/kubernetes-bootcamp --replicas=2 🡪 thjis will scale down.](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[kubectl get deployments 🡪 this will show 2](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[kubectl get pods -o wide 🡪 this will show that 2 of the 4 are terminating status for a while and then disappear.](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[Rolling update :](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[Rolling update allows deployment updates to take place with zero downtime, by incrementally updating Pods instances with new ones. The new Pods will scheduled on nodes with available resources. The application should have multiple instances for the rolling update to happen. The maximum number of Pods that can be unavailable and the maximum number of Pods that can be created, by default, is ONE; but it’s configurable either in numbers or in percentages.](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[If the deployment is exposed via a service, the service will load balance the traffic only to the available Pods during the update.](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[kubectl describe pods 🡪 you can see the current version of the images.](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[kubectl set image deployments/kubernetes-bootcamp kubernetes-bootcamp=jocatalin/kubernetes-bootcamp:v2 🡪 set the image version to 2. This will notify the deployment to use the new image and start a rolling update.](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[kubectl get pods 🡪 you will see that some of the Pods are getting terminated and new ones are coming up.](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[kubectl describe pods 🡪 you will see that image version has changed.](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[kubectl describe services/kubernetes-bootcamp 🡪 you can see all the Pods.](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[export NODE\_PORT=$(kubectl get services/kubernetes-bootcamp -o go-template='{{(index .spec.ports 0).nodePort}}') echo NODE\_PORT=$NODE\_PORT](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[curl host01:$NODE\_PORT 🡪 every time you run this, it’s hitting different Pods and each will show the new image version.](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[kubectl rollout status deployments/kubernetes-bootcamp 🡪 this command will also show the status of the rollout.](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[Rollback:](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[kubectl set image deployments/kubernetes-bootcamp kubernetes-bootcamp=jocatalin/kubernetes-bootcamp:v10 🡪 try to update to a non-existing version.](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[kubectl get deployments 🡪 you will see some of them are failing to start.](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[kubectl rollout status deployments/kubernetes-bootcamp 🡪 it will show that waiting for rollout to finish.](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)

[kubectl rollout undo deployments/kubernetes-bootcamp 🡪 undo the rollout, i.e. roll back the changes.](https://kubernetes.io/docs/user-guide/replication-controller/#what-is-a-replicationcontroller)