# Materials to learn from

* [youtube - TechWorld with Nana](https://www.youtube.com/watch?v=X48VuDVv0do) – 4h Kubernetes beginner tutorial.

# Concepts

## Node

Node is a single computer in Kubernetes cluster (a group of nodes).

## Resources

In Kubernetes we are creating resources which are objects with different purposes, which we manage and which helps us to manage our applications.

Those are pieces creating our application, for example we might have the following resources:

* Pod – containing a container running our app
* Service – Assigning a static IP address to the container running our app
* Volume – mounting external storage to the container running our app

More resources are mentioned and explained in further sections of this documentation.

## Custom resources

Standard resources and available by default in every Kubernetes cluster and Custom resources are additional ones added by using the CRD (Custom Resource Definition) which is a YAML file.

## Manifests

Manifests are conifguration files (mainly YAML) which defines what resources we want to create in Kubernetes cluster.

## Pod

It is a resource which is the smallest unit in Kubernetes. It is an abstraction of a container. Inside of a Pod we have a container running with app, and that Pod is running on one of the nodes.

We don’t work with the container itself, just with a Pod.

## Service

Service is a resource which we can assign to a Pod and it provides a static IP address to Pod s. Pods have its own IP address, which can change over time, and Service has a separate, static IP address and it is independent on the IP address of the Pod.

In Service we can also specify a domain name for a Pod.

It also acts as a load balancer. We can have two replicas (described in the next section) of the same app, both attached to the same service, and service will redirect a request to the replica which is the least busy.

## Replicas

We can have multiple replicas of the same app (Pod). That means that we have multiple copies of the same app running. It is useful because when one app goes down, then the other one can take its place in handling requests.

We can use a Service (described in the previous section) to create a static IP address, and each request made to that IP address will be redirected by a Service to a replica which is the least busy.

## Ingress

Ingress is a resource which we can use to specify which URL will be redirecting to which Service when making a HTTP request.

## ConfigMap

ConfigMap is a resource which we can use to provide an external configuration for an application running as a container in a Pod.

We can define there variables which will be used by our app running in Pod, and we can change them without changing the app itself.

## Secret

Secret is a resource like ConfigMap but for storing confidential data.

## Volumes and volumeMounts

Volumes are a resource which works like volumes in Docker. Volumes allow us to save data from a container outside of a container (on a server running a container). So container will be using this data and even after container restart, that data will be still accessible.

Volumes need to be used together with VolumeMounts which will specify the path in the container where data from a Volume will be accessible.

We can additionally use the hostPath parameter to specify where data from a Volume will be stored on a server running Kubernetes.

For example we might use Volumes in a YAML manifest in this way:

A screenshot of a computer

AI-generated content may be incorrect.

The other way of doing this is to create a separate Volume resource. Here is an example of Persistent Volume (it preserves data even after Pod is deleted):

A screen shot of a computer

AI-generated content may be incorrect.

And yet another option is that we don’t create a Volume resource on our own, we just create a volume claim (PersistentVolumeClaim) and it will create a volume resource automatically:

A screen shot of a computer

AI-generated content may be incorrect.

## Deployment

It is a resource which we create using a manifest. It is a blueprint, set of instructions for app Pods. It is abstraction of Pods. It specifies which app we want to run (which container), how many replicas of that Pod to create and more.

## StatefulSet

It is a resource which we create using a manifest. Is like a Deployment but specifically designed for applications which have a state. For example databases have a state which is their data.

Even if we are using volumes for our databases running in Pods, then still we should use the StatefulSet for them to guarantee data consistency.

## Processes runnin on worker nodes

Each worker node needs to have installed 3 tools:

* Container runtime
* Kubelet
* Kube proxy

### Container runtime

Each node needs to have installed a container runtime, like docker or containerd, which will be running containers.

### Kubelet

It is a process running containers on nodes. It interacts with a container and node. It creates containers based on provided configuration, like Deployment.

### Kube proxy

It forwards requests from Services to Pods.

## Processes running on master nodes

Each master node needs to have 4 processes running:

* API server
* Scheduler
* Controller manager

### API server

Users interacting with Kubernetes cluster are using the API server. It takes requests from users about what they want to do with a cluster, and then API server talks to other processes running on Kubernetes.

It is also used for authentication.

### Scheduler

API server talks to the Scheduler and Scheduler decides on which node to run requested Pods. It check how much resources (CPU, RAM) will be needed for running a Pod, and which nodes have those resources available.

Scheduler doesn’t run the app itself but it talks to the kubelet and kubelet runs an app.

### Controller manager

It monitors the state of Pods, if they are running correctly or if they have died. If a Pod died then Controller manager talks to the Scheduler in order to rerun a Pod.

### Etcd

Etcd Stores data about all the changes that happens in a Kubernetes cluster, for example when Pod dies, and when a new Pod gets created.

It provides data used by other servicer (API server, Scheduler, Controller manager).

## Kubectl

This is a tool for users to interact with a Kubernetes cluster. It allows for example to get list of Pods, create, update or delete resources.

## Namespace

Namespaces are a way for organizing resources in Kubernetes. They are a groups containing specified resources (different containers running different apps).

They don’t have separate resources (nodes, CPU, RAM). They share common, cluster’s resources. They just provide a way yo logically group resources.

## Helm

Helm is a package manager for Kubernetes that helps you define, install, and manage Kubernetes applications.

### Sharing Helm charts

Helm uses charts, which are packages of pre-configured Kubernetes resources written in YAML. They can be uploaded and downloaded from repositories.

### Templating YAML files

Helm also allows for creating YAML files which are using variables and we can put different values into them.

## Operators

Operators can manage applications (resources) in Kubernetes. For example, in case of PostreSQL it can:

* Create a PostgreSQL instance
* Monitor its health
* Automatically handle failures
* Manage backups

Or in case of Spark Operator:

* It watches for the SparkApplication custom resource
* Once that resource is created it creates a Spark Driver Pod

## CoreDNS

It is a DNS server. It translates domain (DNS) names into IP addresses inside of a cluster. That means that once we assign a DNS name to a Pod then every other Pod from a cluster can resolve it.

Every Pod get assigned a default DNS name like this:

* <pod-ip-address>.<namespace>.pod.cluster.local

We can also use a Service resource in order to assign a static IP address and our chosen domain name to a Pod. Then everytime we recreate that Pod it will have always the same domain name.

## The kube-system Namespace

This namespace contains all the system pods which are running required Kubernetes services like kube proxy, kube scheduler, api server etc.

## Webhooks

Webhooks is a tool used for validation or modification of other resources during their creation or update.

They are configured using the MutatingWebhookConfiguration and ValidatingWebhookConfiguration resources.

The Webhook server usually runs inside of its own Pod. Kubernetes communicates with that server through HTTPS.

It uses TLS certificates for secure HTTPS communication between Kubernetes API server and Webhook server. It is saved as files on the Webhook server and as the caBundle field in the MutatingWebhookConfiguration or ValidatingWebhookConfiguration.

# Taints

Taints are a way to control which Pods can be scheduled on specific Nodes.

We can create such a taint on a Node that every Pod that doesn’t tolerate it will not be able to be scheduled on that Node. But in a Pod’s manifest we can specify that it tolerates that taint and then it can be scheduled.

Every Taint consists of 3 parts: key, value and effect. Key and value identifies a Taint (unique identifier of a Taint) and effect defines what happnes when we try to schedule a Pod on a Node that has a Taint not tolerated by that Pod.