apiversion- for each component there is different API version – specify which API version we are using for a component ,
kind – declaring what kind of component we want to create e.g., deployment , service .
3 parts of Kubernetes configuration file :

1) Metadata – name of component , label for component
2) Specification – here we put every basic type of configuration that we want to apply for the component
3) Status – Kubernetes will always compare actual and desired status of component , if both are not equal . Then Kubernetes know something needs to be fixed and fixes it , this is self-healing process of Kubernetes.

Attributes of spec are specific to kind .for each component there are different specs .Every component will have spec .
E.g., for deployment :
Replicas – number of replicas of pod we want for our deployment .
Selector – what should we select to create replica e.g., we can provide matchselector .
Template – a detailed plan for pod . in this template also we define metadata and spec . in spec we can define if we want container , image to be used for container , port for container .
We can create service to expose endpoint of our deployment .

For service kind – we can define nodeport in spec and add ports also – port – port on which pod will run , targetport – port for lb where all pod will send traffic
We don't define status when we create configuration file . But when we pass configuration file to api server status attribute is created by Kubernetes . when we define spec for component – we define desired state for component . Kubernets store desired state in etcd keystore and Kubernetes will understand desired state and create component based on that desired state and all the status will be stored .
Selector field is used to select set of pods that service should route traffic to .
Kubectl apply means kubernetes will act on status and update status continuously and if kubernetes find any mistake with state of replica then it will create new replica of deployment .

Status information comes from etcd . etcd is the cluster brain . it stores current status of any kubernetes component .

Yaml has strict indentation .
Deployment manage pods that are below them .
Whenever we want to create pod we will create deployment .

Pod should have its own configuration inside deployment .

Kubectl get deployment – to check all deployment
Kubectl get service – to list all service present
Kubectl apply -f deployment.yml – it will create deployment using deployment.yml file
Kubectl apply -f svc.yml – it will create service using svc.yml file
Kubectl describe service service_name – to checd details of service eg : pod port , name lb port , endpoint ip .
Kubectl get pods – to get list of pods
Kubectl get pods -o wide – to get details of pod .

Status will be created after we apply configuration so we can check that in yaml or json format :
Kubectl get deployment -o yaml : this will give the entire yaml file for deployment including all status also .
Status information will be stored in etcd key-store
If we add any new change to configuration then it will match with the status from etcd and apply new changes .
We can get external ip to open our service to be accessible from browser for pod using minikube
Kubectl get service
Minukube service service-name – this will craete endpoint for us and our application will be accessible .