**Challenges in Standalone Pod:**

* Pod is a basic entity to run applications on the kubernetes cluster. If you are trying to deploy on a cluster the first thing needed to learn is pod. Any object of kubernetes like Deployment, Daemonset, Replication controller or Replica set will create the pod. Hence it is the actual entity to learn about the pod.
* Pod works as a standalone application in the kubernetes. If something goes wrong with the pod, we manually have to delete the pod and create another one to replace it, by default kubernetes will not create the pod hence it is not recommended to create the standalone pod.
* Pod is not binded to a specific node unless we use Nodename, Nodeselector and NodeAffinity. Here when pods get deleted, the scheduler will create the other pod that may not be on the same specified node where the previous pod got deleted. Thus standalone pods are binded to a particular node only when we use Nodename, Nodeselector and NodeAffinity.
* Pod lifecycle is not managed by the kubernetes cluster; when something goes wrong with the pod, we manually have to delete the pod and recreate another one to replace it.
* It is not recommended to use Pod objects for an application which is to be deployed under high availability rather we can opt for Deployment, Daemonset, Replication controller or Replica set.
* In real time environments of kubernetes cluster, we do not deploy an application as a standalone pod.
* We should always define containers running in a pod as application specific. Though there is a possibility of creating multiple containers inside a single pod, we cannot create containers which are not dependent on the application running inside a pod and all containers inside one pod are also not meaningful. Containers of each application must be isolated in a single pod. For example, if you are running a nginx container then it is necessary to create containers dependent on that nginx application, since we could not manage to create mysql or any other containers irrelevant to nginx.
* Failed pods should be deleted by us through kubectl or API. Let us assume, we are trying to deploy a pod but it happened to be in a failed state or exited with crash loopbackoff back error then we can see those pods through kubectl get pod status command until we use the condition terminated-pod-gc-threshold, which means Pod has some grace period to get deleted by the kube controller from the node, in this grace period pods will be present in garbage collection. These leftover pods will occupy resources unnecessarily thus failed pods have to get deleted.
* If a node expires, the pods scheduled to that node will be deleted after a grace time period(default 30sec) but not soon after nodes go down.