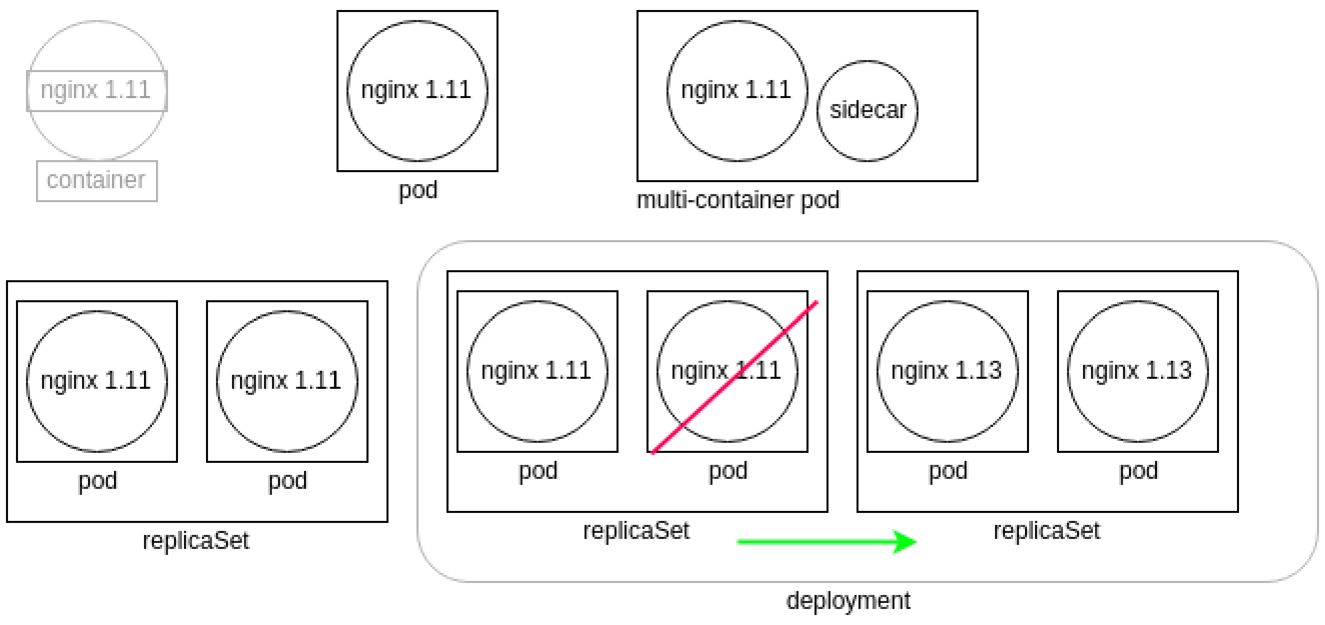
MANAGING STATE WITH DEPLOYMENTS

Managing State with Deployments

Object Relationship

Here you can see the relationship between objects from the container, which Kubernetes does not directly manage, up to the deployment.



Nested Objects

The boxes and shapes are logical, in that they represent the controllers, or watch loops, running as a thread of the kube-controller-manager. Each controller queries the kube-apiserver for the current state of the object they track. The state of each object on a worker node is sent back from the local kubelet.

The graphic in the upper left represents a container running nginx 1.11. Kubernetes does not directly manage the container. Instead, the kubelet daemon checks the pod specifications by asking the container engine, which could be Docker or cri-o, for the current status. The graphic to the right of the container shows a pod which represents a watch loop checking the container status. kubelet compares the current pod spec against what the container engine replies and will terminate and restart the pod if necessary.

A multi-container pod is shown next. While there are several names used, such as *sidecar* or *ambassador*, these are all multi-container pods. The names are used to indicate the particular reason to have a second container in the pod, instead of denoting a new kind of pod.

On the lower left we see a **`replicaSet`**. This controller will ensure you have a certain number of pods running. The pods are all deployed with the same **`podSpec`**, which is why they are called replicas. Should a pod terminate or a new pod be found, the replicaSet will create or terminate pods until the current number of running pods matches the specifications. Any of the current pods could be terminated should the spec demand fewer pods running.

The graphic in the lower right shows a deployment. This controller allows us to manage the versions of images deployed in the pods. Should an edit be made to the deployment, a new **`replicaSet`** is created, which will deploy pods using the new **`podSpec`**. The deployment will then direct the old **`replicaSet`** to shut down pods as the new **`replicaSet`** pods become available. Once the old pods are all terminated, the deployment terminates the old **`replicaSet`** and the deployment returns to having only one **`replicaSet`** running.