**Kube-controller-manager is a collection of different Kubernetes controllers**, all of them included in a binary and running permanently in a loop. Its **main task is to watch for changes in the state of the objects**, and make sure that the actual state converges towards the new desired state. In summary, it is **responsible for the reconciling tasks** around the state of Kubernetes objects.

Controllers running in the kube-controller-manager use a **watch mechanism to get notifications** every time there are changes on resources. Each controller acts accordingly to what is required from this notification (create/delete/update).

There are [multiple controllers](https://github.com/kubernetes/kubernetes/tree/master/pkg/controller) running in the kube-controller-manager. Each one of them is responsible for reconciling its own kind of Object. Let’s talk about some of them:

* ReplicaSet controller: This controller **watches the desired number of replicas for a ReplicaSet**and compares this number **with the Pods matching its Pod selector**. If the controller is informed via the watching mechanism of changes on the desired number of replicas, it acts accordingly, via the Kubernetes API. If the controller needs to **create a new Pod because the actual number of replicas is lower than desired**, it creates the new Pod manifests and posts them to the API server.
* Deployment controller: It takes care of **keeping the actual Deployment state in sync with the desired state**. When there is a change in a Deployment, this controller performs a **rollout of a new version**. As a consequence, a new ReplicaSet is created, **scaling up the new Pods and scaling down the old ones**. How this is performed depends on the strategy specified in the Deployment.
* Namespace controller: When a **Namespace is deleted,** all the **objects belonging to it must be deleted**. The Namespace controller is responsible for completing these deletion tasks.
* ServiceAccount controller: Every time a namespace is created, the **ServiceAccount controller ensures a default ServiceAccount is created for that namespace**. Along with this controller, a Token controller is run at the same time, acting asynchronously and watching ServiceAccount creation and deletion to create or delete its corresponding token. It applies for ServiceAccount secret creation and deletion.
* Endpoint controller: This controller is responsible for **updating and maintaining the list of Endpoints in a Kubernetes cluster**. It watches both Services and Pod resources. When Services or Pods are added, deleted, or updated, it selects the Pods matching the Service Pod criteria (selector) and their IPs and ports to the Endpoint object. When a Service is deleted, the controller deletes the dependent Endpoints of that service.
* PersistentVolume controller: When a user creates a PersistentVolumeClaim (PVC), Kubernetes **must find an appropriate Persistent Volume to satisfy this request** and bind it to this claim. When the **PVC is deleted, the volume is unbound and reclaimed** according to its reclaim policy. The PersistentVolume controller is responsible for such tasks.

In summary, **kube-controller-manager and its controllers reconcile the actual state with the desired state**, writing the new actual state to the resources’ status section. Controllers don’t talk to each other, they always talk directly to the Kubernetes API server.