

**Replication controller (RC) is the first object introduced in K8 to achieve pod replication, allowing K8 to run multiple replicas of your container - Pod, we all understand the value of RC realizing reliability, load balancing and scalability of all workloads running on K8.**

**Replica Set (RS) is the successor of the Replication controller and is meant to do exactly the same, So which one we use and what is the main difference between both?**

**Replication Controller (RC)**

**A ReplicationController ensures that a specified number of replicas of a Pod are running at all times. If Pods exit or are deleted, the ReplicationController acts to instantiate more up to the defined number. Likewise, if there are more running than desired, it deletes as many as necessary to match the defined amount.**

## Components of RC

**A replicationController consists of:**

* **The number of replicas desired (which can be adjusted at runtime).**
* **A Pod definition to use when creating a replicated Pod.**
* **A selector for identifying managed Pods.**

## Replica Set (RS) - next-generation Replication Controller

**RS ensures a specified number of pod replicas are running at any given time.**

## RC vs RS

**The difference between a ReplicaSet and a ReplicationController is that a ReplicaSet supports set-based selector, while replication controller only supports equality-based selector requirements. Same time, RC supports Rolling updates while RS doesn’t.**

**Note:**

**Having said that RC is supporting more deployment strategies - specifically rolling updates - doesn’t mean we are happy with current rolling update implementation! HOW?**

**Kubernetes users always had their own concerns about rolling updates and the way they are implemented because of the following:**

1. **Rather than describing the desired end state, kubectl rolling-update issues commands to get the system into the desired state.**
2. **The whole orchestration logic for replacing the containers and the Replication‐ Controllers is performed by kubectl, which monitors and interacts with the API Server behind the scenes while the update process happens, moving an inherent server-side responsibility to the client.**
3. **You may need more than one command to get the system into the desired state. These commands must be automated and repeatable in different environments.**
4. **Somebody else may override your changes with time.**
5. **The update process has to be documented and kept up-to-date while the service evolves.**
6. **The only way to find out what we have deployed is by checking the condition of the system. Sometimes the state of the current system might not be the desired state, in which case we have to correlate it with the deployment documentation.**

**Therefore, There has to be a better solution replacing RC without losing any of the features including rolling-update.**

**The answer is “Deployment” object.**

## Deployment:

**RS is rarely used separately to manage pods, in most cases K8 users use it internally with another Object for better pod management, “Deployment” is the most commonly used object in this context, Deployment is intended to replace RC on the long term, It complements RS by providing all the missing features including the rolling-update.**

**What is the deal, both are allowing rolling-update!**

**Don’t forget While RC rolling update will need a creation of new RC and occurs on the client side using kubectl, IT IS NOT THE CASE in “Deployment” rolling-update!!!**

**First happening via client is a big violation of separation of concerns as It is pure server responsibility, and because of that also the rolling-update becomes vulnerable because of any network isolation during the execution of the rolling-update command. While in “Deployment” case It is all about the flexibility of RS newer set-based label selectors that allows K8 users to roll-update on server side with full history and audit that helps later to rollback.**

**Just relax and avoid this hassle by switching to “RS’ and “Deployment” Objects.**

**Finally, this is not the only difference but in my opinion is the most important one, for more details review K8 official documentation**