**Replicaset:**

* A ReplicaSet is a Kubernetes object that ensures that a specified number of replicas of a pod are running at any given time.
* It manages the lifecycle of pods and provides a way to scale and maintain the desired state of the application.
* The ReplicaSet is also responsible for creating and managing pods based on a template specification.
* It creates new replicas of a pod when necessary and removes old ones when they’re no longer needed.
* It also provides a mechanism for rolling updates and rollbacks of the application by creating new replicas with updated configurations and terminating the old ones.

**How a ReplicaSet works:**

* In replicasets there is a lot of properties which helps replicasets to work properly.
* There is selector properties which helps replicasets to recreate the pods based on label when pod failure happen, also helped to group the same label pods.
* There is one more properties called template that help RS to create pod according to the given pod template
* Replicas properties is used to provide the number of desired pods to create

**When to use a ReplicaSet:**

* A ReplicaSet ensures that a specified number of pod replicas are running at any given time.
* However, a Deployment is a higher-level concept that manages ReplicaSets and provides declarative updates to Pods along with a lot of other useful features. Therefore, we recommend using Deployments instead of directly using ReplicaSets, unless you require custom update orchestration or don't require updates at all.
* This actually means that you may never need to manipulate ReplicaSet objects: use a Deployment instead, and define your application in the spec section

**Key difference:**

|  |  |
| --- | --- |
| **Deployment** | **Replicaset** |
| **High-level abstractions that manage replica sets.**  It provides additional features such as rolling updates, rollbacks, and versioning of the application. | **A lower-level abstraction that manages the desired number of replicas of a pod.**  Additionally, it provides basic scaling and self-healing mechanisms. |
| Deployment **manages a template of pods** and uses replica sets to ensure that the specified number of replicas of the pod is running. | ReplicaSet only **manages the desired number of replicas of a pod.** |
| **Deployment provides a mechanism for rolling updates and rollbacks of the application**, enabling seamless updates and reducing downtime. | Applications must be manually updated or rolled back. |
| **It provides versioning of the application, allowing us to manage multiple versions of the same application.** It also makes it easy to roll back to a previous version if necessary. | ReplicaSet doesn’t provide this feature. |

**Replication controller:**

* If there are too many pods, the Replication Controller terminates the extra pods. If there are too few, the Replication Controller starts more pods
* A relocation controller is an object that enable you to easily create multiple pods, then make sure that number of pods always exists.
* if a pod get crashed, terminated then RC will automatically recreate the new pod with similar configuration.
* RC is recommended if you just want to make sure 1 pod is always running, even after system reboot.
* You can run the RC with 1 replica & the RC make sure that pod is always running.

**Example:**

apiVersion: apps/v1

kind: ReplicaSet

metadata:

name: frontend

labels:

app: guestbook

tier: frontend

spec:

# modify replicas according to your case

replicas: 3

selector:

matchLabels:

tier: frontend

template:

metadata:

labels:

tier: frontend

spec:

containers:

- name: php-redis

image: gcr.io/google\_samples/gb-frontend:v3