kubernetes

Pod Replication

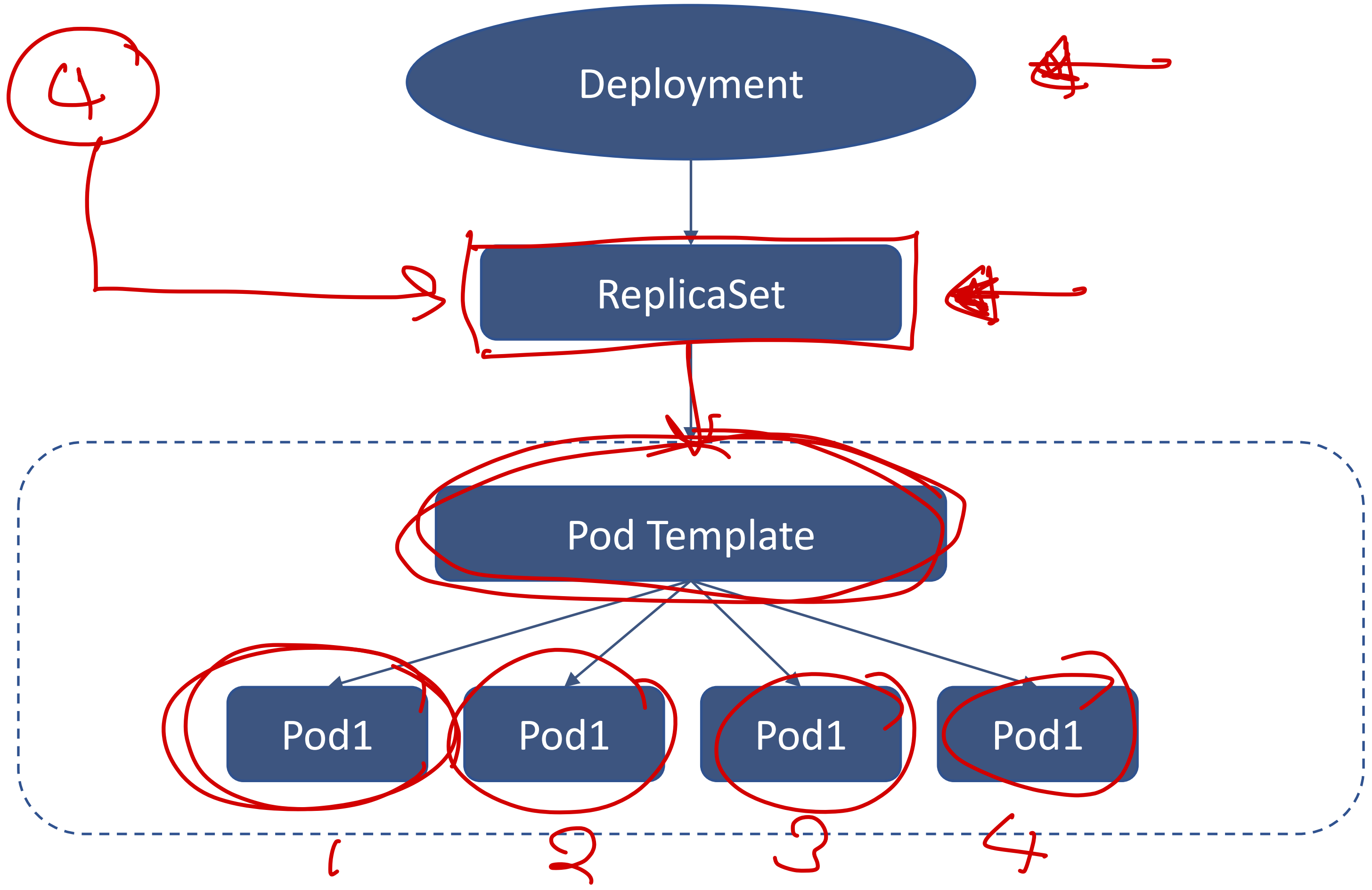
Replication Controller & ReplicaSet

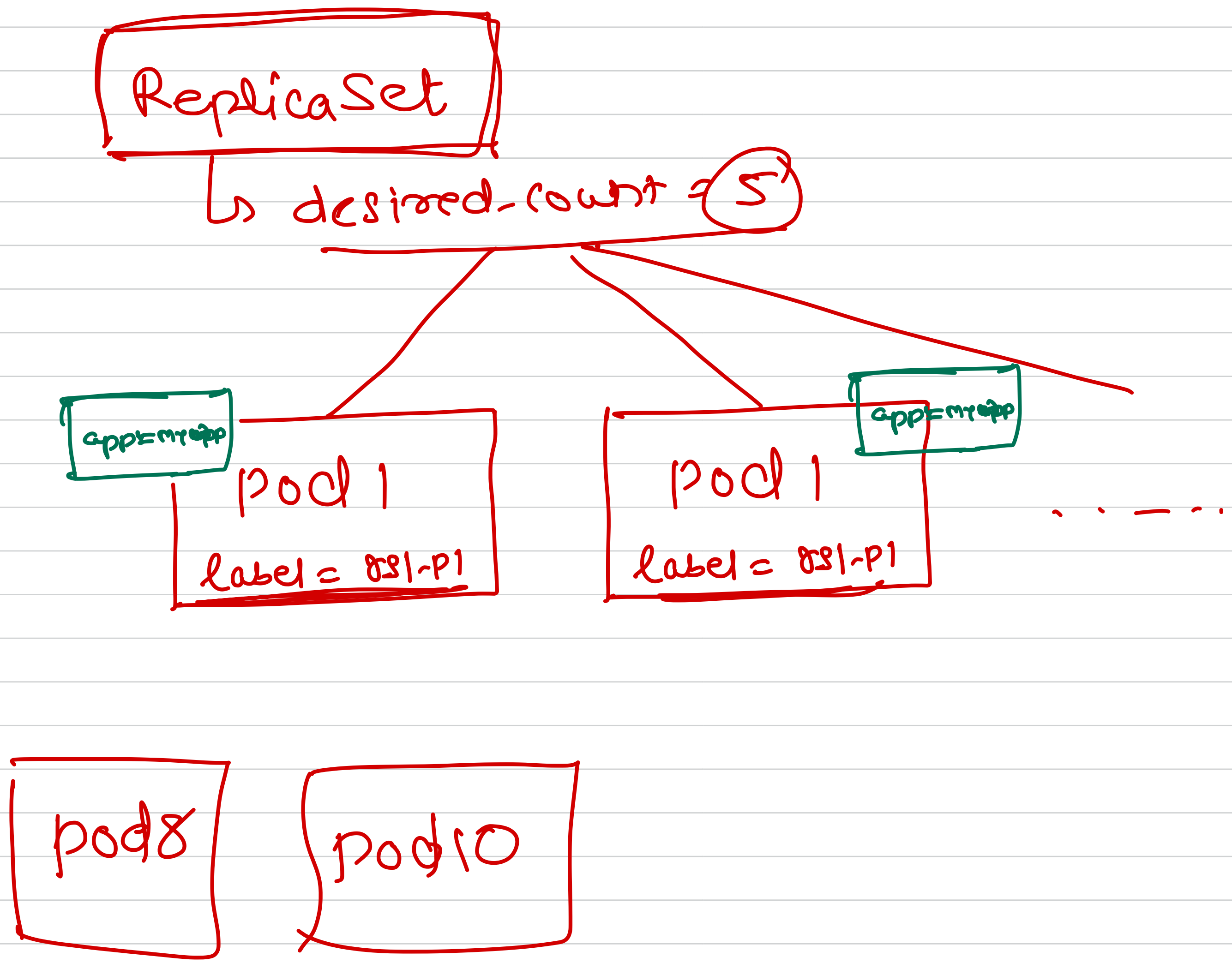
- Both of them do the similar job
- You specify the number of pods (desired count) you want to run, and RC or RS will make sure that those many pods are running at any given time
- If pod crashes RC/RS will start new pods to match the desired count
- To create the new pod, RC/RS will use the template specified in the yml definition
- Replication Controller supports only equality based selectors
- ReplicaSet supports both equality based as well as set based selectors

Deployments

- Describes the desired state of a component of the application
- In a simplest case, deployment involves one more more ReplicaSets to create pod replicas
- User does not require to use ReplicaSet separately as deployment will use it internally
- A single deployment is similar to a single microservice in the application
- Result of each deployment is a ReplicaSet which then manage the pods in the microservice
- You can create deployment
  - Using command
  - Using yml definition file

Deployment


4
Deployment
4
ReplicaSet
Pod Template
Pod1
Pod1
Pod1
Pod1
1
2
3
4


ReplicaSet
desired_count = 5
app=myapp
pod 1
label = rs1-p1
app=myapp
pod 1
label = rs1-p1
pod8
pod10

Services

Need of Services

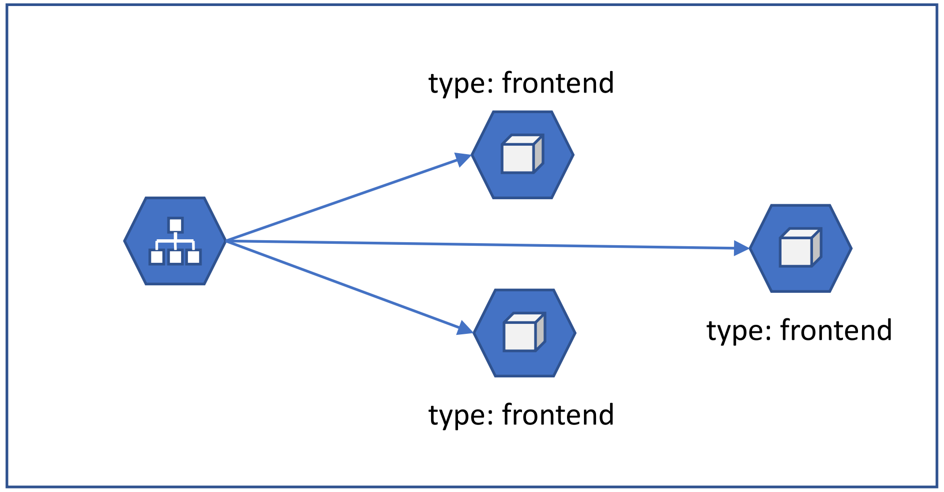
- Kubernetes pods are mortal, means they can die because of any reason
- Kubernetes provide different controllers to manage the lifecycle of these pods.
- Kubernetes gives a pod its own unique IP address, but if the pod dies then the IP address changes
- It becomes very difficult to keep track of which IP address to connect to access the application
- The problem increases in multi-tier applications

Service

- Service provides an abstraction for a set of Pods and a policy to access them
- The set of Pods targeted by a service are determined by selector
- Service is used to expose an application running in set of pods
- It provides a single DNS name and can load balance across them
- Represented by Kubernetes API object and it is namespaced

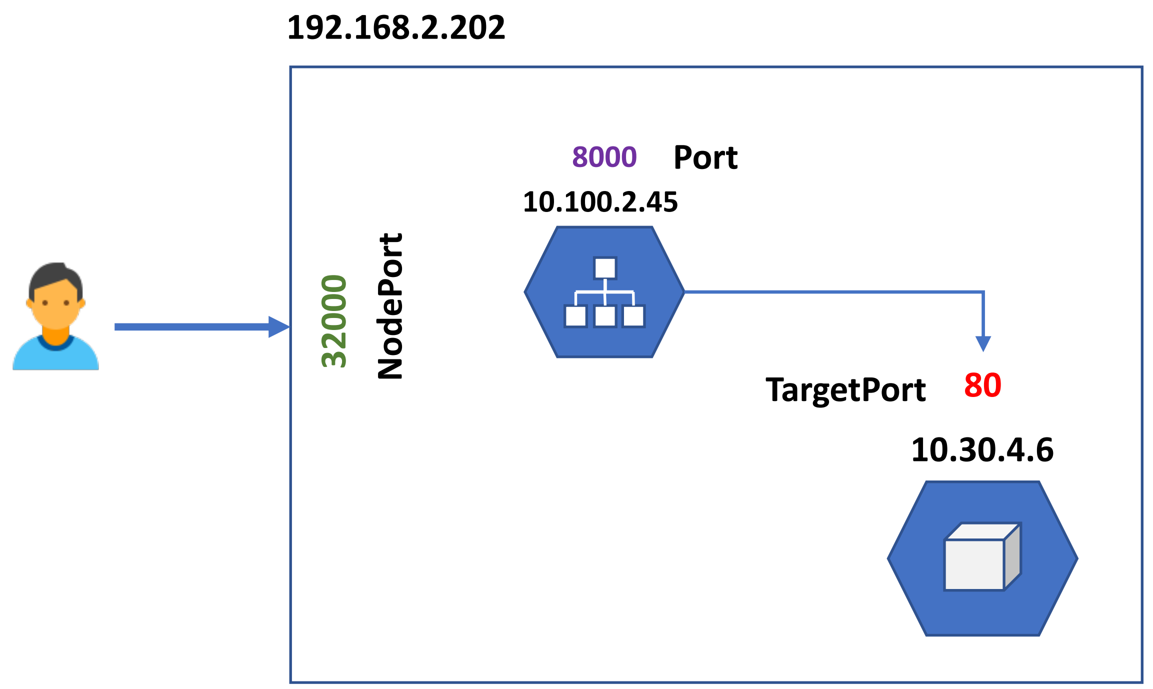
Service Type: ClusterIP

- Exposes the service within the cluster
- It uses cluster internal IP which is not reachable from external network
- Service maps any incoming port to a targetPort



Service Type: NodePort

- Exposed on each node’s IP address
- NodePort service can be accessible from external network
- Service provides the NodePort on which the application is accessible
- ClusterIP service gets created automatically



Service Types

- LoadBalancer
  - Exposes the service externally using a cloud provider's load balancer
  - Along with the LoadBalancer, ClusterIP and NodePort services are created automatically
  - External load balancer routes the traffic through these pods
- ExternalName
  - The service does not contain the selectors, instead it uses DNS names
  - It maps the pod to the external name like test.sunbeaminfo.com
  - Mainly used to expose and object using cname record