30 Days of Kubernetes :-

#Day4

Various objects in Kubernetes :-

1) Pod
2) service
3) Replica set
4) Deployment
5) config Map
6) secret
7) Volumes

1) Pod :-

- smallest deployable units of computing that we can create & manage in K8s.
- consists of one or more containers with shared storage & Network resources.

simple pod definition file :-

pods/simple-pod.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

2) service :-

- A service is a method for exposing a Network application that is running as one / more pods in your cluster.
- Each service object defines logical set of endpoints (usually these endpoints are pods) along with policy about how to make these pods available.

3) Replica set :-

- A Replicaset's purpose is to maintain a stable set of Pods running at any given time. i.e it is used to guarantee the availability of a specified no. of identical pods.

4) Deployment :-

- Deployment provides declarative updates for Pods & Replica sets.
- you can describe state in deployment and the deployment controller changes actual state to desired state at controlled rate. you can define deployments to create new Replica sets or to remove existing deployments and adopt all their resources with new Deployments.

5) ConfigMaps :-

- It is an API object used to store key-value pairs which are Non-confidential.
- Pods can consume ConfigMaps as env. variables, CLI arguements & config files in Volumes.
- Name of the ConfigMap must be valid DNS sub domain name.

6) Secrets :-

- A secret is a object that contains a small amount of sensitive data such as password, a token, or a key.
- Using a secret means you don't need to use confidential data in your application code

7) Volumes :-

On Disk files in a container are ephemeral, when container terminates/crashes we lost all data. Multiple containers are running in a pod and needs to share files. It's challenging to establish shared FS among all of containers. K8's "Volume" abstraction solves both of these problems

- Kubernetes supports several types of volumes.