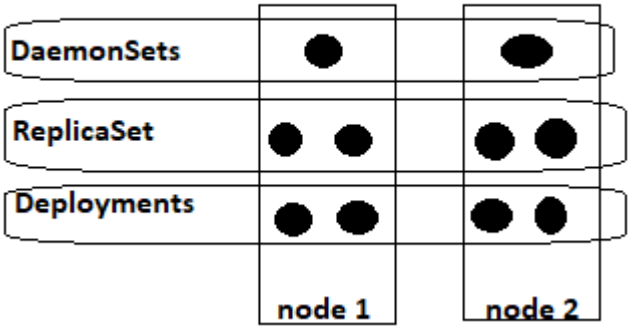
DaemonSets

DaemonSets are like replica sets, as in it helps you to deploy multiple instances of pod. But it runs one copy of your pod on each node in your cluster.



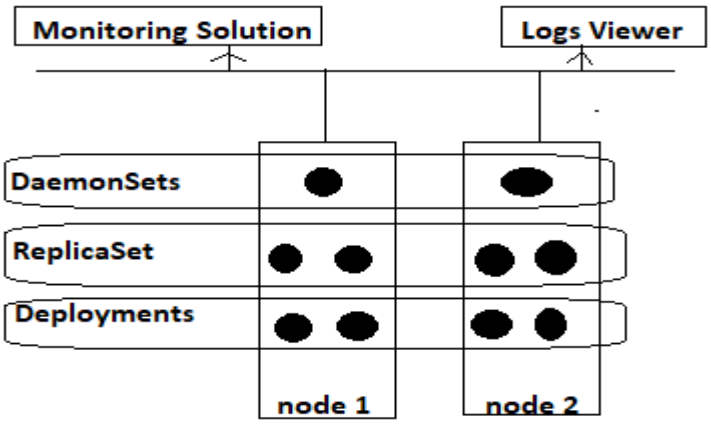
Whenever a new role is added to the cluster a replica of the pod is automatically added to that node and when a node is removed the pod is automatically removed.

The DaemonSets ensure that one copy of the pod is always present in all nodes in the cluster.

Use cases:

1) If you would like to deploy a monitoring agent or log collector on each of your nodes in the cluster. So, you can monitor your cluster better.

DaemonSet is a perfect for that as it can deploy monitoring agent in the form if a pod in all the nodes in your cluster. Then you don't have to worry about adding/removing monitoring agents. From these nodes where there are changes in your cluster.



2) The kuberproxy component can be deployed as a DaemonSet in the cluster.

3) Networking solutions like weave net requires an agent to be deployed on each node in the cluster.

- **How does DaemonSet works? and How does it schedule pods on each node?**

  We could set the nodeName property on the pod to bypass the schedular and get the pod placed on a node directly. On each pod set the nodeName property in its specification. Before it is created and when they are created, they automatically land on the respective nodes.

  So that's how it used to be until kubernetes version v1.12.

Note: From v1.12 onwards the DaemonSets uses DefaultSchedular and Node Affinity.

Example:

elasticsearch.yml

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  creationTimestamp: null
  labels:
    app: elasticsearch
  name: elasticsearch
  namespace: kube-system
spec:
  selector:
    matchLabels:
      app: elasticsearch
  template:
    metadata:
      labels:
        app: elasticsearch
    spec:
      containers:
      - image: k8s.gcr.io/fluentd-elasticsearch:1.20
        name: fluentd-elasticsearch
```

```
controlplane $ kubectl apply -f elasticsearch.yml
daemonset.apps/elasticsearch created
controlplane $ kubectl -n kube-system get ds elasticsearch
NAME            DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
elasticsearch   1         1         0       1            0           <none>          5s
```