DaemonSet:

A DaemonSet ensures that all (or some) Nodes run a copy of a Pod. As nodes are added to the cluster, Pods are added to them. As nodes are removed from the cluster, those Pods are garbage collected. Deleting a DaemonSet will clean up the Pods it created.

Some typical uses of a DaemonSet are:

- running a cluster storage daemon on every node
- running a logs collection daemon on every node
- running a node monitoring daemon on every node

In a simple case, one DaemonSet, covering all nodes, would be used for each type of daemon. A more complex setup might use multiple DaemonSets for a single type of daemon, but with different flags and/or different memory and cpu requests for different hardware types

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: my-ds
  namespace: my-ns
  labels:
    type: daemon
spec:
  selector:
    matchLabels:
      tier: middleWare
  template:
    metadata:
      labels:
        tier: middleWare
    spec:
      containers:
        - name: random
          image: busybox
          command: ['echo','Hello There']
          ports:
          - containerPort: 8080
```

How a Daemon Pods are Scheduled:

A DaemonSet can be used to ensure that all eligible nodes run a copy of a Pod. The DaemonSet controller creates a Pod for each eligible node and adds the spec.affinity.nodeAffinity field of the Pod to match the target host. After the Pod is created, the default scheduler typically takes over and then binds the Pod to the target host by setting the .spec.nodeName field. If the new Pod cannot fit on the node, the default scheduler may preempt (evict) some of the existing Pods based on the priority of the new Pod.

Commands

- kubectl create -f daemonsetDefinition.yaml
- kubectl delete -f daemonsetDefinition.yaml