
kubernetes
DAEMONSET

❖ DAEMONSET:

- A DaemonSet is a type of Kubernetes API object that replicates identical Pods across the Nodes in your cluster.
- It 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.
- DaemonSets are often used to run long-lived background services such as Node monitoring systems and log collection agents.
- DaemonSets differ from any of these other Kubernetes workload types because they have unique scheduling behavior.
- Pods, ReplicaSets, and Deployments schedule to available cluster Nodes automatically, until the requested number of replicas is running. Unless you set affinity rules, you can't know which Nodes will be selected to run a Pod. DaemonSets, however, ensure every Node runs a replica of the Pod.

USE CASES:

RUNNING NODE MONITORING AGENTS:

- In-cluster services that collect metrics data from your Nodes need to reliably deploy a Pod on each one. DaemonSets implement this behavior without requiring any special configuration.

COLLECTING LOGS FROM NODES:

- Similarly, collecting the contents of Node-level logs (such as Kubelet and kernel logs) helps you audit your environments and troubleshoot problems. Deploying your logging service as a DaemonSet ensures all your Nodes will be included.

BACKING UP NODE DATA:

- Backups are another good candidate for DaemonSets. Using a DaemonSet ensures all your Nodes will be included in your backups without making you scale or reconfigure your backup service when Nodes change.
- If some Nodes don't need backups, you can customize your DaemonSet so that only relevant Nodes are covered

CREATE A DAEMONSET:

- The simple manifest for a DaemonSet that runs the **Fluentd logging system** on each of your cluster's Nodes:

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluentd
spec:
  selector:
    matchLabels:
      name: fluentd
  template:
    metadata:
      labels:
        name: fluentd
    spec:
      containers:
        - name: fluentd-elasticsearch
          image: quay.io/fluentd_elasticsearch/fluentd:latest
```

→ **Use Kubectl to apply it to your cluster:**

$kubectl apply -y <file-name>

→ **To get Daemonsets:**

$kubectl get daemonset

$kubectl get pods -o wide

SCOPING DAEMONSETS TO SPECIFIC NODES:

- We can configure DaemonSets with a **nodeSelector** and **affinity rules** to run Pods on only some of your cluster's Nodes. These constraints are set using the DaemonSet's **spec.template.spec.nodeSelector** and **spec.template.spec.affinity fields**, respectively.

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluentd
spec:
  selector:
    matchLabels:
      name: fluentd
  template:
    metadata:
      labels:
        name: fluentd
    spec:
      nodeSelector:
        - log-collection-enabled: "true"
      containers:
        - name: fluentd-elasticsearch
          image: quay.io/fluentd_elasticsearch/fluentd:latest
```

→ **Before applying, set the** log-collection-enabled: true **label on one of your Nodes:**

$kubectl label node Node2 log-collection-enabled=true

$ kubectl apply -f fluentd.yaml

$ kubectl get daemonsets

→ **Pod list will confirm that the Pod is running on the labelled Node2:**

$ kubectl get pod -o wide