# **DaemonSet Controller**

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 **logs** **collection daemon** on every node, such as:
  + **fluentd**: Fluentd is an open source data collector, which lets you unify the data collection and consumption for a better use and understanding of data
  + **logstash**: Logstash is a lightweight, open-source, server-side data processing pipeline that allows you to collect data from various sources, transform it on the fly, and send it to your desired destination. It is most often used as a data pipeline for Elasticsearch, an open-source analytics and search engine
* Running a node **monitoring daemon** on every node, such as: -
  + **Prometheus** **Node** **Exporter**: It is an agent that gathers system metrics and exposes them in a format which can be ingested by Prometheus.
  + **collectd**: It is a Unix daemon written in C language, and it's designed to collect metrics pertaining to system resources and application performance.
  + **Datadog** **agent**: Provides an observability service for cloud-scale applications, providing monitoring of servers, databases, tools, and services, through a SaaS-based data analytics platform.
* Running a cluster **storage daemon** on every node, such as:
  + **glusterd**: Gluster is a scalable, distributed file system that aggregates disk storage resources from multiple servers into a single global namespace.
  + **Ceph**: Ceph makes it possible to decouple data from physical storage hardware using software abstraction layers, which provides unparalleled scaling and fault management capabilities.

**This is how the master pods on the worker nodes run, such as kube-proxy and kubelet.**

apiVersion: apps/v1

kind: **DaemonSet**

metadata:

  name: prometheus-daemonset

spec:

  selector:

    matchLabels:

      tier: monitoring

      name: prometheus-exporter

  template:

    metadata:

      labels:

        tier: monitoring

        name: prometheus-exporter

    spec:

      containers:

      - name: prometheus

**image: prom/node-exporter**

        ports:

        - containerPort: 80

**Updating DaemonSet**

**updateStrategy**

* **RollingUpdate**: Gradual, rolling update where Pods are updated one at a time (or more based on maxUnavailable setting). Default strategy.
* **OnDelete**: No automatic updates. Pods are updated only when manually deleted. Gives manual control over when updates happen.

**Delete the DaemonSet but retain the Pod**

Kubernetes maintains the relationship between Pods and controllers, but it lets you break that relationship with non-cascading deletes.

* kubectl delete ds <dsname> --cascade=false
* kubectl get ds
* kubectl get po -l app=nginx