70. Daemon Sets

* "Daemon Sets", are like "Replicasets", * Replicaset helps you deploy multiple instances of pods but Daemonset runs one copy of our pod on each node in your cluster whenever a new node 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 DaemonSet ensures that one copy of the pod is always present in all nodes in the cluster.

Usecases of DaemonSets :-

⇒ Say 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, a DaemonSet is perfect for that, as it can deploy your monitoring agent in the form of a pod in all the nodes in your cluster

⇒ Then we dont have to worry about adding or removing monitoring agents from those nodes when there're changes in your cluster as the "DaemonSet" will take care of that for you.

Daemon Sets - Use Case - Kube-proxy

⇒ Earlier, we've learned that one of the worker node components that is required on every node in the cluster is a "kube-proxy". This is one good use case of "DaemonSets"

⇒ The "kube-proxy" component can be deployed as a DaemonSet in the cluster

Another Use case "networking" - weave-net

* "weave-net" requires an agent to be deployed on each node in the cluster

"Daemonset" manifest file & "Replicaset" manifest file looks similar except the "kind"

daemon-set-definition.yaml

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: monitoring-daemon
spec:
  selector:
    matchLabels:
      app: monitoring-agent
  template:
    metadata:
      labels:
        app: monitoring-agent
    spec:
      containers:
        - name: monitoring-agent
          image: monitoring-agent
```

Replicaset - definition. yaml

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
    name: monitoring-daemon
spec:
    selector:
        matchLabels:
            app: monitoring-agent

    template:
        metadata:
            labels:
                app: monitoring-agent
        spec:
            containers:
            - name: monitoring-agent
              image: monitoring-agent
```

To create daemon-set

```
> kubectl create -f daemon-set-definition.yaml
```

View daemon-set

```
>kubectl get daemonsets
```

To view daemon-set with more details

```
>kubectl describe daemonsets <monitoring-daemon>
```

How does DaemonSet work

How does "DaemonSet" schedule pods on each node & how does it ensure that every node has a pod?

Until/Before k8s 1.12 version, we could set the nodeName property on the pod to bypass the scheduler & get the pod placed on a node directly [Each pod set the node name property in its specification before it is created & when they're created they automatically land on respective node]

From V1.12 uses NodeAffinity and default Scheduler

From V1.12 onwards ~~that~~ ~~uses~~ ~~use~~ the "DaemonSet" uses the "default scheduler" & "node affinity" rules that we learned in one of the previous lectures to schedule pods on nodes