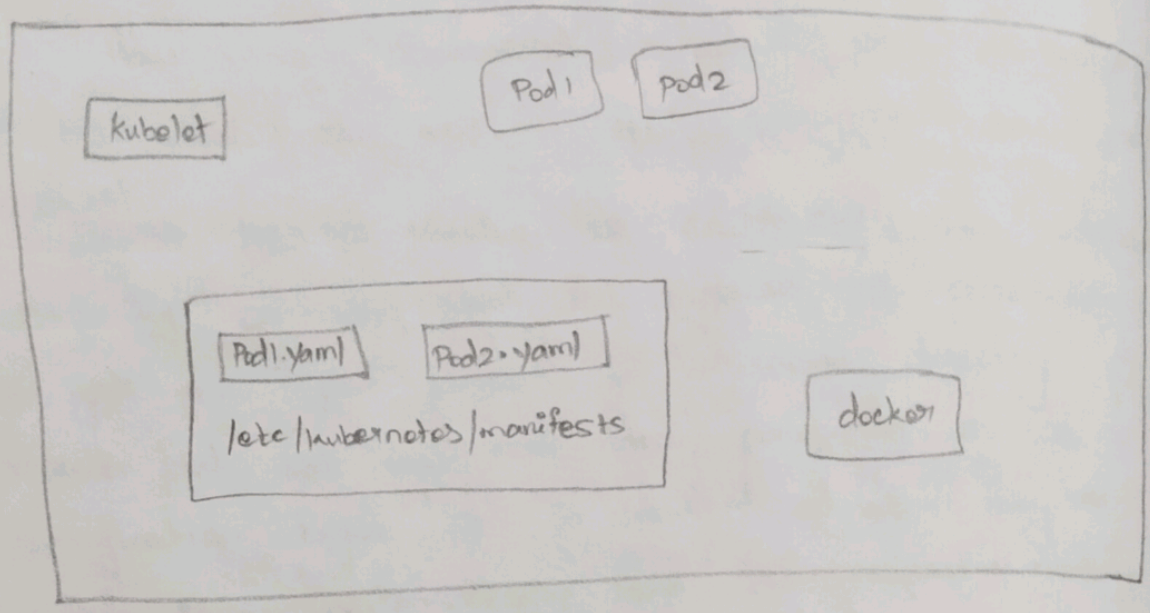
73 Static pods

* "control plane" node is usually created using static pods



2:39 what if there is "no master" & "no other node" & there is "only one node" that has only 'kubelet' installed but which is not part of any cluster

⇒ Is there anything "kubelet" can do alone, can it operate as independent node, if so, who would provide the instructions required to create those pods

⇒ Kubelet can manage the node independently, if it has "docker" to run containers. There is no K8S cluster [no master & also other worker nodes]

⇒ The only thing that the kubelet knows to do is create pods, but we don't have an API server here to provide Pod details, we know that "to create pods" we need the details of the pod in a "pod definition file".

⇒ But how do we provide "pod-definition file" to the 'kubelet' without a "kube-apiserver"

⇒ We can configure the "kubelet" to read the "pod definition files" from a directory on the server designated to store information about Pods.

⇒ The kubelet periodically checks this directory for files, reads these files, & creates pods on the host.

⇒ Not only it creates the pod, it can ensure that the "Pod stays alive", if the application crashes, the "kubelet" attempts to restart it.

⇒ If you make a change to any of the files within this directory, the kubelet recreates the Pod for those changes to take effect

⇒ If you remove a file from this directory, the pod is deleted automatically.

⇒ These pods created by its own (kubelet) without the intervention from the API server or rest of the k8s cluster components are known as "static pods"

Limitations:-

⇒ We can only create pods this way, you cannot create "replica sets" or "deployments" or "services", by placing a "definition file" in the designated directory They're all concepts part of the whole k8s architecture that require other cluster plane components like the "replication" & "deployment controllers" etc.,

⇒ kubelet only works on pod level & can only understand pods & which is why it is able to create static Pods this way

Static pod

1) pod-manifest-path
2) config option

⇒ kubelet.service

manifest file

3:16
So what is that designated folder & how do you configure it

⇒ It could be any directory on the host of the location of that directory is passed into the kubelet as an option while running the service.

⇒ The option is named pod-manifest-path and here it is set to /etc/kubernetes/manifests folder

kubelet.service

```
ExecStart = /usr/local/bin/kubelet \\
  --container-runtime = remote \\
  --container-runtime-endpoint = unix:// /var/run/containerd/
                                  containerd.sock \\
  --pod-manifest-path = /etc/kubernetes/manifests \\
  --kubeconfig = /var/lib/kubelet/kubeconfig \\
  --network-plugin = cni \\
  --register-node = true \\
  --v=2
```

⇒ There is also another way to configure this, instead of specifying the option directly in the kubelet.service file,

⇒ you could provide a path to another config file using the config option & define the directory path as staticPodPath in that file

⇒ Clusters set up by the "kube admin" tool uses this approach.

kubelet.service

ExecStart = /usr/local/bin/kubelet \\

--container-runtime = remote \\

--container-runtime-endpoint = unix:///var/run/containerd/containerd.sock \\

--config = KubeConfig.yaml \\

--KubeConfig = /var/lib/kubelet/KubeConfig \\

--network-plugin = cni \\

--register-node = true \\

--v = 2

KubeConfig.yaml

staticPodPath: /etc/kubernetes/manifest

If you're inspecting an existing cluster, you should inspect this option of the kubelet to identify the path to the directory. You will then know where to place the definition file for your static pods

We should know how to "view" & "configure" this option, irrespective of the method used to setup the cluster

1) First, check the option "pod-manifest-path" in the "kube-let" service file. If its not there, then

2) Look for the config option & identify the file used as the config file & then within the config file, look for the "staticPodPath" option

Either of this should give right path

⇒ Once the static pods are created, you can view them by running the `docker ps` command

⇒ why not "kubectl command", because we don't have the rest of the k8s cluster yet. so the kubectl utility works with the kube-apiserver, since we don't have an API server here, no kubectl utility. which is why we're using docker command.

So how does this work, when the node is part of the cluster

when there is an API server requesting the kubelet to create pods. Can kubelet create both kinds of pods at the same time?

The way the kubelet works is that it can take in requests for creating pods from different inputs

1) First, is through the pod definition files from the static pods folder, as we just saw

2) It is through an HTTP API endpoint, & that's how the "kube-apiserver" provides input to kubelet

The kube-let can provide both kinds of pods

* Static pods &
* Ones from the API server at the same time

Well in that case, is the API server aware of the static pods created by the kubelet, Yes it is

If we run,

kubectl get pods on master node, the static pods will be listed as any other pods

How is this Happening

when the kubelet creates static pod, if it is a part of a cluster, it also creates a mirror object in the "kube-apiserver"

what we see from the "kube-api server" is just a read-only mirror of the pod, you can view the details of the pod but you cannot edit or delete it like the usual pods

we can only delete them by modifying the files from the nodes manifest folder

Note that the name of the pod is automatically appended with the node name,

In this case node01

kubectl get pods

NAME	READY	STATUS	RESTARTS	AGE
static-web-node01	0/1	ContainerCreating	0	29s

So, why we have to use static pods

Since, static pods are not dependant on the K8S control plane, you can use static pods to deploy the control plane components itself as Pods on a Node

well, start by installing "kubelet" on all "master nodes", then create "pod definition files" that uses docker images of the various control plane components such as the API Server, Controller, Etcd, etc.,

⇒ place the "manifest file" in the designated manifest folder & the kubelet takes care of deploying the control plane components themselves as pods on the cluster

⇒ This way, you don't have to download binaries, configure services, or worry about the services crashing

⇒ If any of these services were to crash, since it is a static pod, it'll be automatically restarted by the kubelet

⇒ This is how the kubeadmin tool sets up a k8s cluster. which is why when you list the pods in the kube-system ns, you'll see control plane components as Pods in a cluster setup by the "kube admin" tool

```
kubectl get pods -n kube-system
```

diff b/w static pods & Daemonsets.

Static pods	Daemonsets
1) Created by the kubelet	created by Kube-API server (Daemonset Controller)
2) Deploy Control plane Component as "Static" Pods.	Deploy Monitoring Agents, Logging Agents on nodes
Ignored by the kube-scheduler	