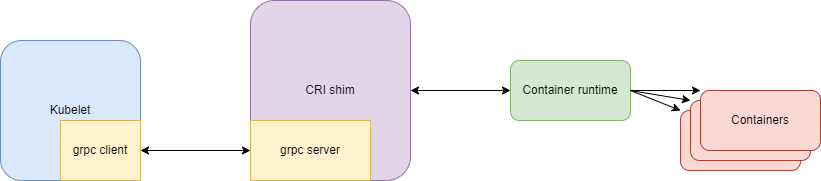
## **Why are we using containerd over docker in kubernetes?**

## 

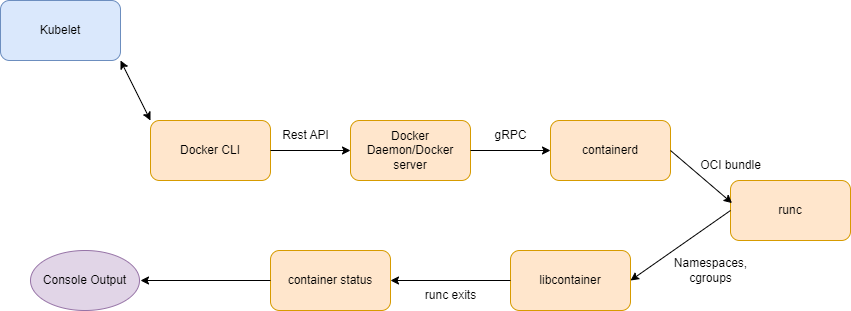
Kubernetes has deprecated docker as a container runtime since version *V.120.* This is due to the reason that using docker leads to more latency compared to any other. There are a few other options such as cri-o, rkt container engine.



Here, kubelet acts as our client which sends requests if we want to deploy pods/containers. CRI is a layer that acts between container runtime and kubelet. Container runtime is what we know as docker, containerd, cri-o, rkt, etc.

Between kubelet and CRI a grcp protocol is formed. Once, CRI sends this request to container runtime, it creates the necessary containers and informs kubelet.

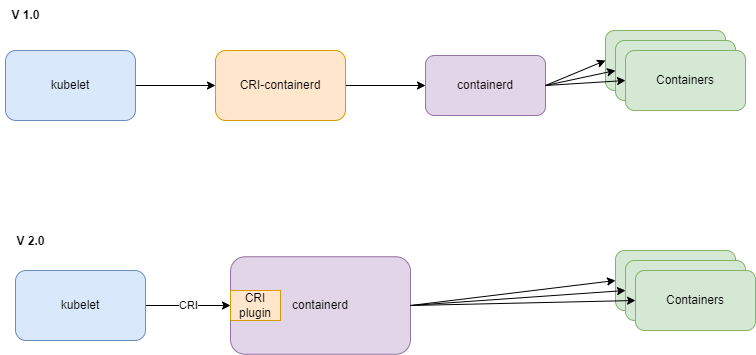
Whereas if we look at the container creation workflow while using docker, it looks like this:



Here, docker also uses containerd in the backend to create a container. Since if we are using docker, it first has to go through the docker CLI and then to the docker daemon and finally to containerd, kubernetes has decided to use containerd by default as it is much faster than using a middleman in docker.

Note: We still have a great use for docker as creating custom images and adding them to the docker registry is much easier and also using a docker engine in our local system is easier than creating a whole cluster like kubernetes.

In containerd, there has been a few changes from v1.0 to v2.0:



The intermediate stem in V1.0 where it is passed through the CRI layer has been modified as CRI is now available as a plugin in containerd from v2.0.