Kubernetes is depricating Docker. What this mean?

A couple of days ago a statement is heard, "Docker support in the Kubernetes is now deprecated and will be removed in a future release" and people are just getting panicked as How is this possible as Docker is the tool which made containers popular in the first place.

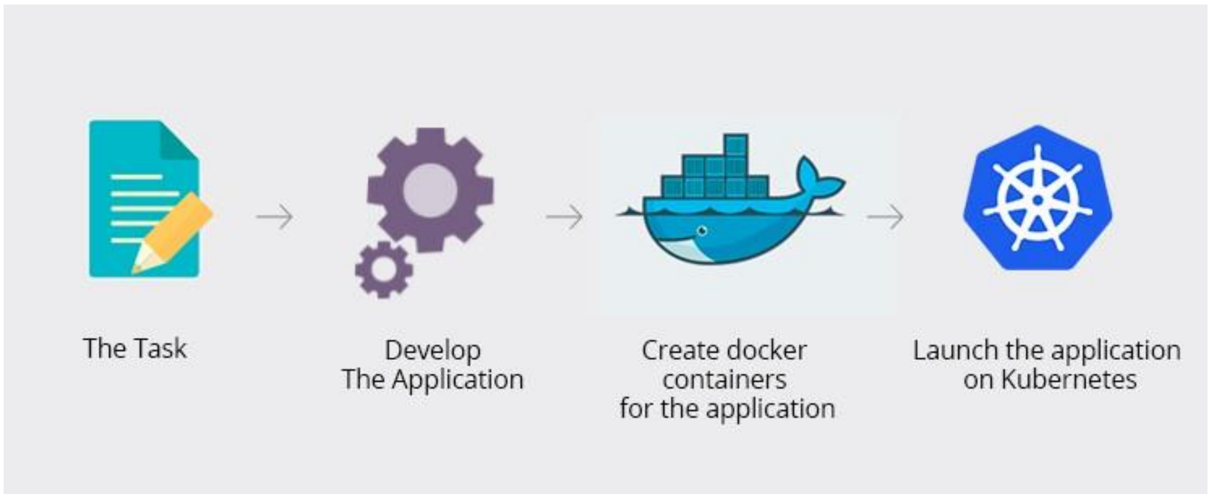
It's nothing to panic about. Our docker is not going anywhere.

The updates made has its own benefits. Let's understand what this statement, "Kubernetes deprecating docker" means, and how they affect developers, DevOps Engineer, etc. working on Docker or Kubernetes.

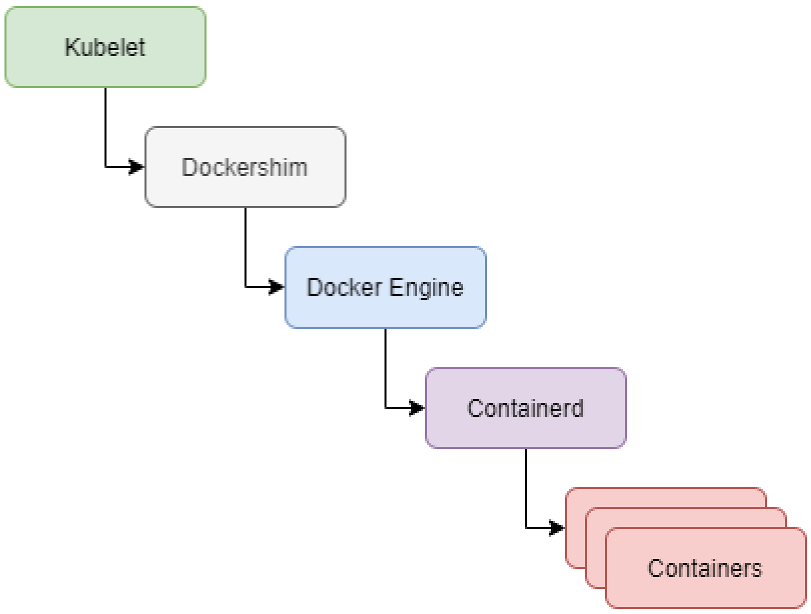
As we know Kubernetes supports different containers runtime in which docker is one of them and most popular one. Kubernetes will still run Docker because Kubernetes uses open container initiation. We can still use docker to create containers for Kubernetes as docker images are fully capable with no matter how you build them and where they are hosted. Kubernetes might use other ways to run container in cluster but docker images will continue to run just fine in Kubernetes Cluster.

The Main purpose of deprecating docker is that we don't need to install docker which is a whole application include tools for containers. Let's say we deployed docker engine on Kubernetes worker node. Docker Engine allows developing, assemble, ship and run application using Docker Engine Components CLI, API and Server.

Server has a couple of components and features like Runtime Containers which is responsible for managing the whole cycle likes starting, stopping containers, pulling and running container images and Build Images.


The Task
Develop The Application
Create docker containers for the application
Launch the application on Kubernetes

The only part Kubernetes need to run the containers inside Cluster is the Container Runtime of docker. So why should Kubernetes install whole docker application with the features Kubernetes don't need because Kubernetes has its own features like Kubernetes Volume, Networks, CLI as well as Container network interface. So, it is logical not to deploy the whole docker with all the features Kubernetes doesn't need and instead of just the runtime component. If containers run in Kubernetes cluster will save resources like RAM, CPU and Storage. This also reduce the security risk as the fewer components you have more secure you are. To use container runtime component Kubernetes need to interact with docker and Kubernetes uses docker shim for the interaction.



This flow chart explains the workflow of using dockers by Kubernetes. To use Containerd we need to install and srtup whole Docker.

What is Dockershim ?

Dockershim implements CRI (Container Runtime Interface) support for Docker and maintenance issues noticed by Kubernetes Community. Dockershim is updated and maintained by Kubernetes till now. The part of code in Kubernetes, talks to docker is actually Kubernetes is deprecating and removing from code i.e. DOCKERSHIM. Main reason is to stop is that to stop maintaining Dockershim. As to use one thing i.e. containerd we need to use another tool which is Dockershim. So if it's possible to include containerd in docker stack why does Kubernetes will use another tool dockershim.

Containers Runtime Alternatives

Instead of installing whole docker, we can use many alternatives which are lightweight purpose build for runtime container as they don't do other things like build containers image or fancy CLI.

1. Containerd — A part of Docker Daemon designed to be used by Docker and Kubernetes or any other containers platform to abstract away syscall or OS functionality to run container in different Operating Systems.

2. Cri-O — Lightweight container runtime to Kubernetes used by OpenShift.

3. Buildah — Buildah is a Command-Line tool for building Open Container which is compatible with both Docker and Kubernetes. So Buildah is better choice for Building Containers

4. Podman — Podman is a demon less container engine for developing, managing, and running OCI Containers on Linux System and a better alternatives of docker for running container locally.

What this Update means for Developers and everyone working on Kubernetes ?

The changes done will is going to affect some but not everyone and finally when people will understand exactly what changed they will realize that this change is good.

For an end-user of Kubernetes

- For the DevOps Engineer responsible for installing resources on an existing Kubernetes Cluster doesn’t change anything and no action required from your side. They can continue to work as they are.
- If using Docker for building container images and within your CI/CD pipeline, we can continue to use it

As Kubernetes Administrator

These can be DevOps Engineer, Sys Admin or Cloud Engineer who sets up Kubernetes Cluster and configure Kubernetes Cluster on cloud in AWS, Google Cloud so, you also don't need to take any action because Cloud providers are already managing that for you.

If using docker as container runtime

You are impacted If using docker as container runtime within Kubernetes cluster. You have to replace with another container runtime like Containerd or CRI-O

If you have built cluster

You need to work if you have built cluster or have an on-premise setup because you need to replace your container runtime and need to start again after modification.