弃用 Dockershim 的常见问题

为什么弃用 dockershim

维护 dockershim 已经成为 Kubernetes 维护者肩头一个沉重的负担。 创建 CRI 标准就是为了减轻这个负担，同时也可以增加不同容器运行时之间平滑的互操作性。 但反观 Docker 却至今也没有实现 CRI，所以麻烦就来了。
Dockershim 向来都是一个临时解决方案（因此得名：shim）。
此外，与 dockershim 不兼容的一些特性，例如：控制组（cgoups）v2 和用户名字空间（user namespace），已经在新的 CRI 运行时中被实现。 移除对 dockershim 的支持将加速这些领域的发展。

在 Kubernetes 1.20 版本中，我还可以用 Docker 吗?

当然可以，在 1.20 版本中仅有的改变就是：如果使用 Docker 运行时，启动 `kubelet` 的过程中将打印一条警告日志。

什么时候移除 dockershim

考虑到此改变带来的影响，k8s 使用了一个加长的废弃时间表。 在 Kubernetes 1.22 版之前，它不会被彻底移除；换句话说，dockershim 被移除的最早版本会是 2021 年底发布的 1.23 版。

我现有的 Docker 镜像还能正常工作吗?

当然可以，`docker build` 创建的镜像适用于任何 CRI 实现。 所有你的现有镜像将和往常一样工作。

私有镜像呢?

当然可以。所有 CRI 运行时均支持 Kubernetes 中相同的拉取（pull）Secret 配置， 不管是通过 PodSpec 还是通过 ServiceAccount 均可。

Docker 和容器是一回事吗?

虽然 Linux 的容器技术已经存在了很久， 但 Docker 普及了 Linux 容器这种技术模式，并在开发底层技术方面发挥了重要作用。 容器的生态相比于单纯的 Docker，已经进化到了一个更宽广的领域。 像 OCI 和 CRI 这类标准帮助许多工具在 k8s 的生态中成长和繁荣，其中一些工具替代了 Docker 的某些部分，另一些增强了现有功能。

现在是否有在生产系统中使用其他运行时的例子?

Kubernetes 所有项目在所有版本中出产的工件（Kubernetes 二进制文件）都经过了验证。
此外，`kind` 项目使用 `containerd` 已经有年头了， 并且在这个场景中，稳定性还明显得到提升。 `Kind` 和 `containerd` 每天都会做多次协调，以验证对 `Kubernetes` 代码库的所有更改。 其他相关项目也遵循同样的模式，从而展示了其他容器运行时的稳定性和可用性。 例如，OpenShift 4.x 从 2019 年 6 月以来，就一直在生产环境中使用 `CRI-O` 运行时。

至于其他示例和参考资料，你可以查看 `containerd` 和 CRI-O 的使用者列表， 这两个容器运行时是云原生基金会（[CNCF]）下的项目。

- containerd
- CRI-O

人们总在谈论 OCI，那是什么?

OCI 代表开放容器标准， 它标准化了容器工具和底层实现（technologies）之间的大量接口。 他们维护了打包容器镜像（OCI image-spec）和运行容器（OCI runtime-spec）的标准规范。 他们还以 runc 的形式维护了一个 runtime-spec 的真实实现， 这也是 containerd 和 CRI-O 依赖的默认运行时。 CRI 建立在这些底层规范之上，为管理容器提供端到端的标准。

我应该用哪个 CRI 实现?

这是一个复杂的问题，依赖于许多因素。 在 Docker 工作良好的情况下，迁移到 containerd 是一个相对容易的转换，并将获得更好的性能和更少的开销。 然而，在条件允许的条件下建议你先探索 CNCF 全景图 提供的所有选项，以做出更适合你的环境的选择。

当切换 CRI 底层实现时，我应该注意什么?

Docker 和大多数 CRI（包括 containerd）的底层容器化代码是相同的，但其周边部分却存在一些不同。 迁移时一些常见的关注点是：

- 日志配置
- 运行时的资源限制
- 直接访问 docker 命令或通过控制套接字调用 Docker 的节点供应脚本
- 需要访问 docker 命令或控制套接字的 kubectl 插件
- 需要直接访问 Docker 的 Kubernetes 工具（例如：kube-imagepuller）
- 像 `registry-mirrors` 和不安全的注册表这类功能的配置
- 需要 Docker 保持可用、且运行在 Kubernetes 之外的，其他支持脚本或守护进程（例如：监视或安全代理）
- GPU 或特殊硬件，以及它们如何与你的运行时和 Kubernetes 集成

如果你只是用了 Kubernetes 资源请求/限制或基于文件的日志收集 DaemonSet，它们将继续稳定工作， 但是如果你用了自定义了 dockerd 配置，则可能需要为新容器运行时做一些适配工作。

另外还有一个需要关注的点，那就是当创建镜像时，系统维护或嵌入容器方面的任务将无法工作。 对于前者，可以用 `crictl` 工具作为临时替代方案 (参见 从 docker 命令映射到 crictl)； 对于后者，可以用新的容器创建选项，比如 img、 buildah、 kaniko、或 buildkit-cli-for-kubectl， 他们均不需要访问 Docker。