拉勾教育
—互联网人实战大学—

《Kubernetes 原理剖析与实战应用》

正范

— 拉勾教育出品 —

25｜稳定基石：带你剖析容器运行时以及 CRI 原理

前言

Pod 在 Kube-APIServer 中被创建出来后

会被调度器调度，然后确定一个合适的节点

最终被这个节点上的 Kubelet 拉起，**以容器状态运行**

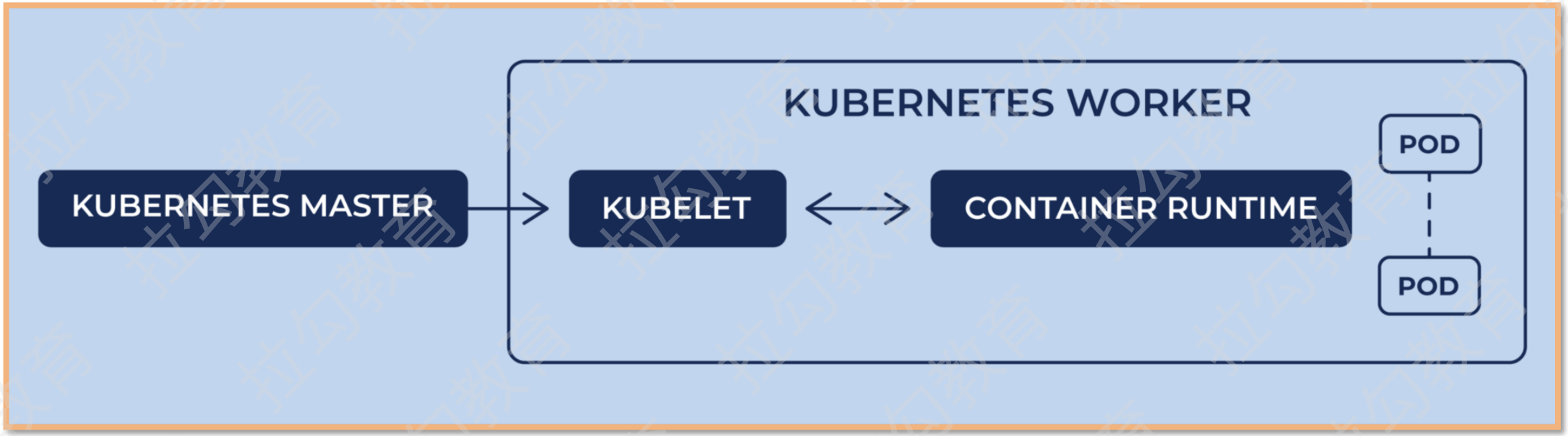
容器运行时 （Container Runtime）

Kubelet

负责运行具体的 Pod，并维护其整个生命周期，为 Pod 提供存储、网络等必要的资源

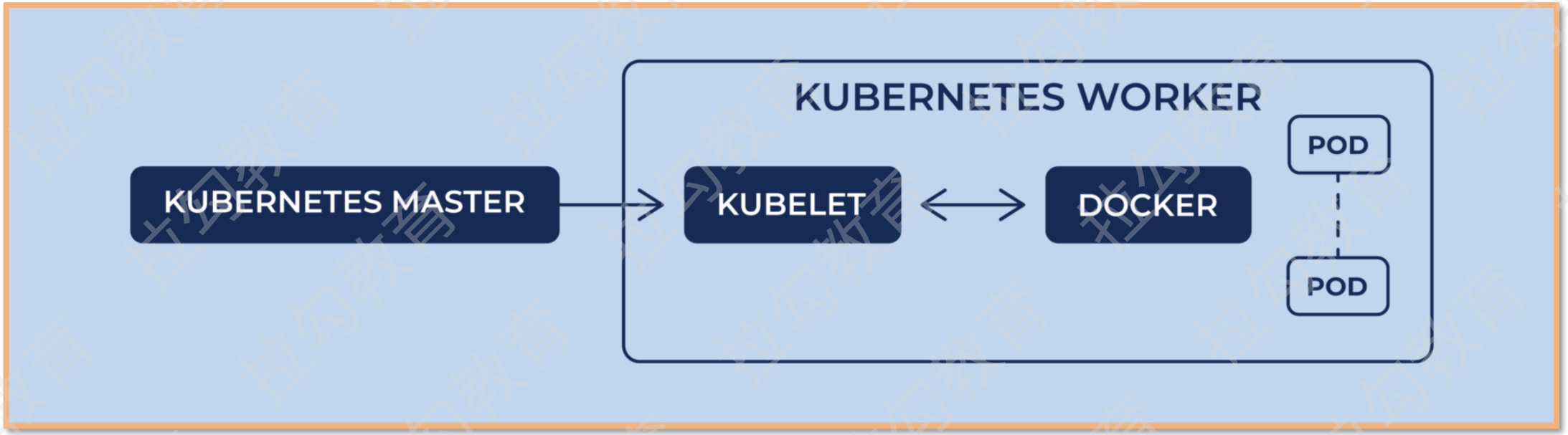
不负责真正的容器创建和逻辑管理

容器运行时 （Container Runtime）



Kubelet 跟容器运行的交互图

容器运行时 （Container Runtime）



使用 Docker 作为容器的运行

容器运行时 （Container Runtime）

在 Kubernetes v1.5 之前，Kubelet 内置了对 rkt 的支持

从 v1.5 版本开始，社区引入 CRI（Container Runtime Interface）

容器运行时 （Container Runtime）

CRI 接口两个好处

将 Kubelet 与容器运行时进行解耦
容器运行时进行更新升级等操作
不需要对 Kubelet 做任何的更改

解放 Kubelet，减少 Kubelet 负担
保证 Kubernetes 代码质量和整个系统的稳定性

CRI


```
// Runtime service defines the public APIs for remote container runtimes
service RuntimeService {
  // Version returns the runtime name, runtime version, and runtime API version.
  rpc Version(VersionRequest) returns (VersionResponse) {}

  // RunPodSandbox creates and starts a pod-level sandbox. Runtimes must ensure
  // the sandbox is in the ready state on success.
  rpc RunPodSandbox(RunPodSandboxRequest) returns (RunPodSandboxResponse) {}
  // Start a sandbox pod which was forced to stop by external factors.
  // Network plugin returns same IPs when input same pod names and namespaces
  rpc StartPodSandbox(StartPodSandboxRequest) returns (StartPodSandboxResponse) {}
  // StopPodSandbox stops any running process that is part of the sandbox and
  // reclaims network resources (e.g., IP addresses) allocated to the sandbox.
  // If there are any running containers in the sandbox, they must be forcibly
  // terminated.
  // This call is idempotent, and must not return an error if all relevant
  // resources have already been reclaimed. kubelet will call StopPodSandbox
  // at least once before calling RemovePodSandbox. It will also attempt to
  // reclaim resources eagerly, as soon as a sandbox is not needed. Hence,
```

CRI


```
// multiple StopPodSandbox calls are expected.
rpc StopPodSandbox(StopPodSandboxRequest) returns (StopPodSandboxResponse) {}
// RemovePodSandbox removes the sandbox. If there are any running containers
// in the sandbox, they must be forcibly terminated and removed.
// This call is idempotent, and must not return an error if the sandbox has
// already been removed.
rpc RemovePodSandbox(RemovePodSandboxRequest) returns (RemovePodSandboxResponse) {}
// PodSandboxStatus returns the status of the PodSandbox. If the PodSandbox is not
// present, returns an error.
rpc PodSandboxStatus(PodSandboxStatusRequest) returns (PodSandboxStatusResponse) {}
// ListPodSandbox returns a list of PodSandboxes.
rpc ListPodSandbox(ListPodSandboxRequest) returns (ListPodSandboxResponse) {}

// CreateContainer creates a new container in specified PodSandbox
rpc CreateContainer(CreateContainerRequest) returns (CreateContainerResponse) {}
// StartContainer starts the container.
rpc StartContainer(StartContainerRequest) returns (StartContainerResponse) {}
// StopContainer stops a running container with a grace period (i.e., timeout).
// This call is idempotent, and must not return an error if the container has
```

CRI


```
// already been stopped.
// TODO: what must the runtime do after the grace period is reached?
rpc StopContainer(StopContainerRequest) returns (StopContainerResponse) {}
// RemoveContainer removes the container. If the container is running, the
// container must be forcibly removed.
// This call is idempotent, and must not return an error if the container has
// already been removed.
rpc RemoveContainer(RemoveContainerRequest) returns (RemoveContainerResponse) {}
// PauseContainer pauses the container.
rpc PauseContainer(PauseContainerRequest) returns (PauseContainerResponse) {}
// UnpauseContainer unpauses the container.
rpc UnpauseContainer(UnpauseContainerRequest) returns (UnpauseContainerResponse) {}
// ListContainers lists all containers by filters.
rpc ListContainers(ListContainersRequest) returns (ListContainersResponse) {}
// ContainerStatus returns status of the container. If the container is not
// present, returns an error.
rpc ContainerStatus(ContainerStatusRequest) returns (ContainerStatusResponse) {}
// UpdateContainerResources updates ContainerConfig of the container.
rpc UpdateContainerResources(UpdateContainerResourcesRequest) returns
```

CRI


```
(UpdateContainerResourcesResponse) {}
  // ReopenContainerLog asks runtime to reopen the stdout/stderr log file
  // for the container. This is often called after the log file has been
  // rotated. If the container is not running, container runtime can choose
  // to either create a new log file and return nil, or return an error.
  // Once it returns error, new container log file MUST NOT be created.
  rpc ReopenContainerLog(ReopenContainerLogRequest) returns (ReopenContainerLogResponse)
{}

  // ExecSync runs a command in a container synchronously.
  rpc ExecSync(ExecSyncRequest) returns (ExecSyncResponse) {}
  // Exec prepares a streaming endpoint to execute a command in the container.
  rpc Exec(ExecRequest) returns (ExecResponse) {}
  // Attach prepares a streaming endpoint to attach to a running container.
  rpc Attach(AttachRequest) returns (AttachResponse) {}
  // PortForward prepares a streaming endpoint to forward ports from a PodSandbox.
  rpc PortForward(PortForwardRequest) returns (PortForwardResponse) {}

  // ContainerStats returns stats of the container. If the container does not
```

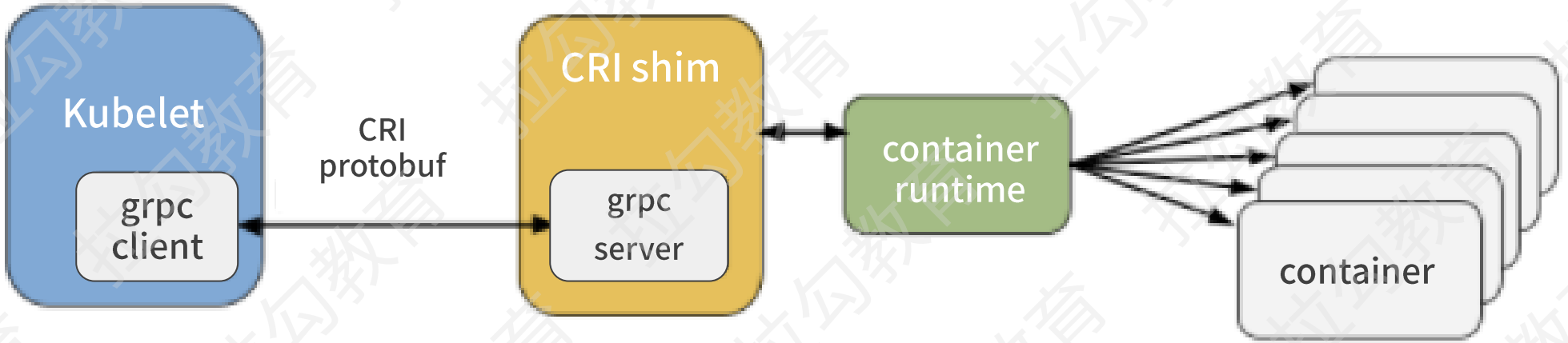
CRI

```
rpc Exec(ExecRequest) returns (ExecResponse) {}
// Attach prepares a streaming endpoint to attach to a running container.
rpc Attach(AttachRequest) returns (AttachResponse) {}
// PortForward prepares a streaming endpoint to forward ports from a PodSandbox.
rpc PortForward(PortForwardRequest) returns (PortForwardResponse) {}

// ContainerStats returns stats of the container. If the container does not
// exist, the call returns an error.
rpc ContainerStats(ContainerStatsRequest) returns (ContainerStatsResponse) {}
// ListContainerStats returns stats of all running containers.
rpc ListContainerStats(ListContainerStatsRequest) returns (ListContainerStatsResponse) {}

// UpdateRuntimeConfig updates the runtime configuration based on the given request.
rpc UpdateRuntimeConfig(UpdateRuntimeConfigRequest) returns
(UpdateRuntimeConfigResponse) {}

// Status returns the status of the runtime.
rpc Status(StatusRequest) returns (StatusResponse) {}
```

CRI

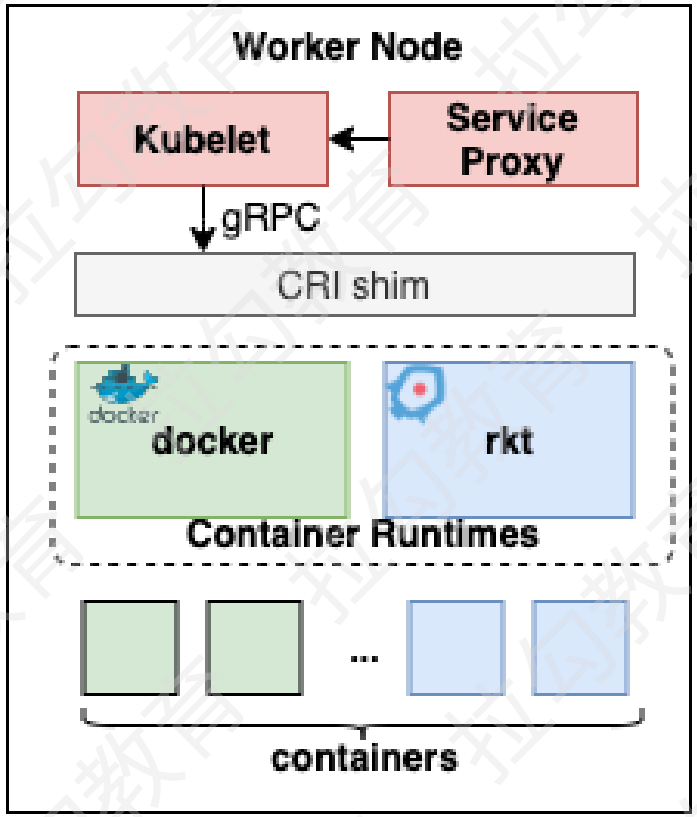

```
// ImageService defines the public APIs for managing images.
service ImageService {
    // ListImages lists existing images.
    rpc ListImages(ListImagesRequest) returns (ListImagesResponse) {}
    // ImageStatus returns the status of the image. If the image is not
    // present, returns a response with ImageStatusResponse.Image set to
    // nil.
    rpc ImageStatus(ImageStatusRequest) returns (ImageStatusResponse) {}
    // PullImage pulls an image with authentication config.
    rpc PullImage(PullImageRequest) returns (PullImageResponse) {}
    // RemoveImage removes the image.
    // This call is idempotent, and must not return an error if the image has
    // already been removed.
    rpc RemoveImage(RemoveImageRequest) returns (RemoveImageResponse) {}
    // ImageFSInfo returns information of the filesystem that is used to store images.
    rpc ImageFsInfo(ImageFsInfoRequest) returns (ImageFsInfoResponse) {}
}
```

CRI



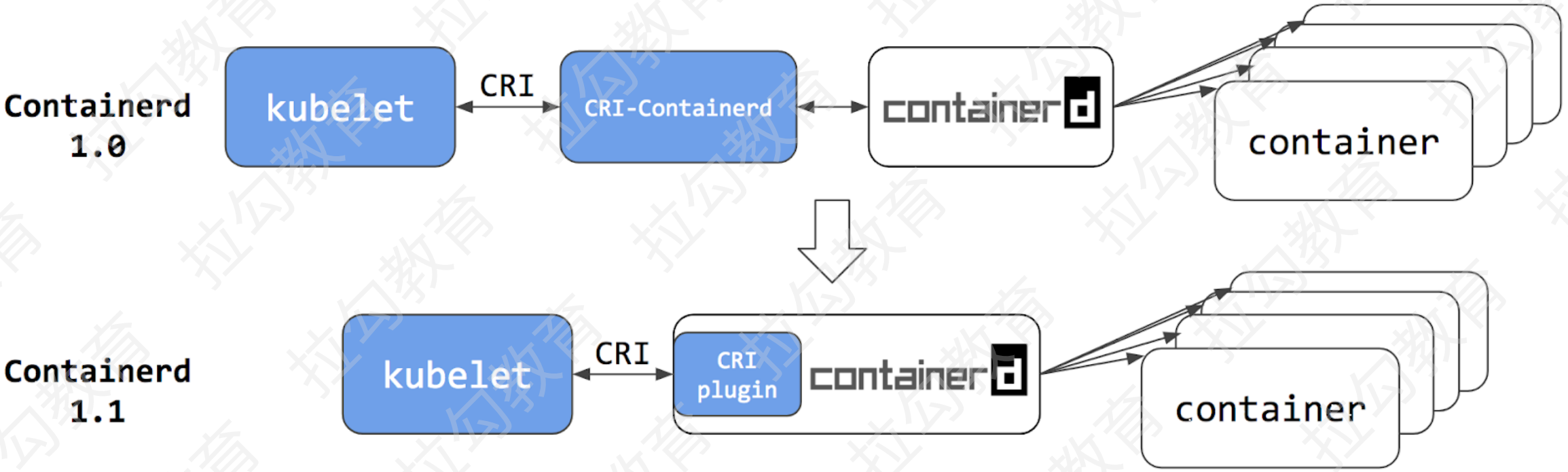
Kubelet 与容器运行时的交互

CRI



Kubelet 内置对 CRI shim 的实现

CRI



部署 containerd

写在最后

Kubernetes

- CRI 提供简单易用的扩展接口，极大地方便了用户进行定制化
- CRI 对容器运行时进行抽象，这极大地方便了开发者的对接，减少升级和维护成本
- Kubernetes 中可以为不同的 Pod 设置不同的容器运行时（Container Runtime）以提供性能与安全性之间的平衡

https://kubernetes.io/zh/docs/concepts/containers/runtime-class/

Next：《26 | 网络插件：Kubernetes 搞定网络原来可以如此简单？》

拉勾教育

—互联网人实战大学—

关注拉勾「教育公众号」
获取更多课程信息

L / A / G / O / U