k3s汇报及迁移方案

卜一珂

K3s 是一个轻量级的 Kubernetes 发行版，有以下增强功能：

打包为单个二进制文件；

使用基于 sqlite3 的轻量级存储后端作为默认存储机制，同时支持使用 etcd3、MySQL 和 PostgreSQL 作为存储机制；

封装在简单的启动程序中，通过该启动程序处理很多复杂的 TLS 和选项；

添加了简单但功能强大的batteries-included功能，例如：本地存储提供程序，服务负载均衡器，Helm controller 和 Traefik Ingress controller；

所有 Kubernetes control-plane 组件的操作都封装在单个二进制文件和进程中，使 K3s 具有自动化和管理包括证书分发在内的复杂集群操作的能力；

最大程度减轻了外部依赖性，K3s 仅需要 kernel 和 cgroup 挂载。 K3s 软件包需要的依赖项包括：

containerd

Flannel

CoreDNS

CNI

主机实用程序（iptables、socat 等）

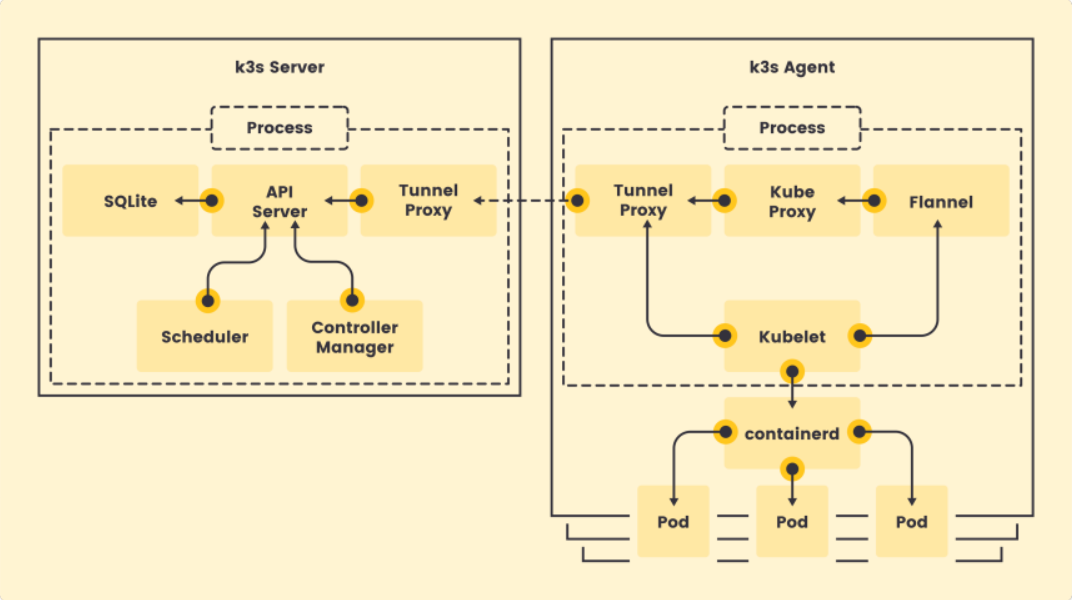
Ingress controller（Traefik）

嵌入式服务负载均衡器（service load balancer）

嵌入式网络策略控制器（network policy controller）

1. k3s架构

1.1整体架构



1.1.1 部署架构

k3s Server：一个进程，包含了API Server, Scheduler, Controller Manager, 以及代替了etcd 的SQLite, 和k3s Agent通信的Tunnel Proxy。

K3s Agent：两个进程体，第一个进程包含Kube Proxy, Kubelet, Flannel, Tunnel Proxy; 第二个进程是containerd，负责管理运行容器。

1.1.2 代码组织架构

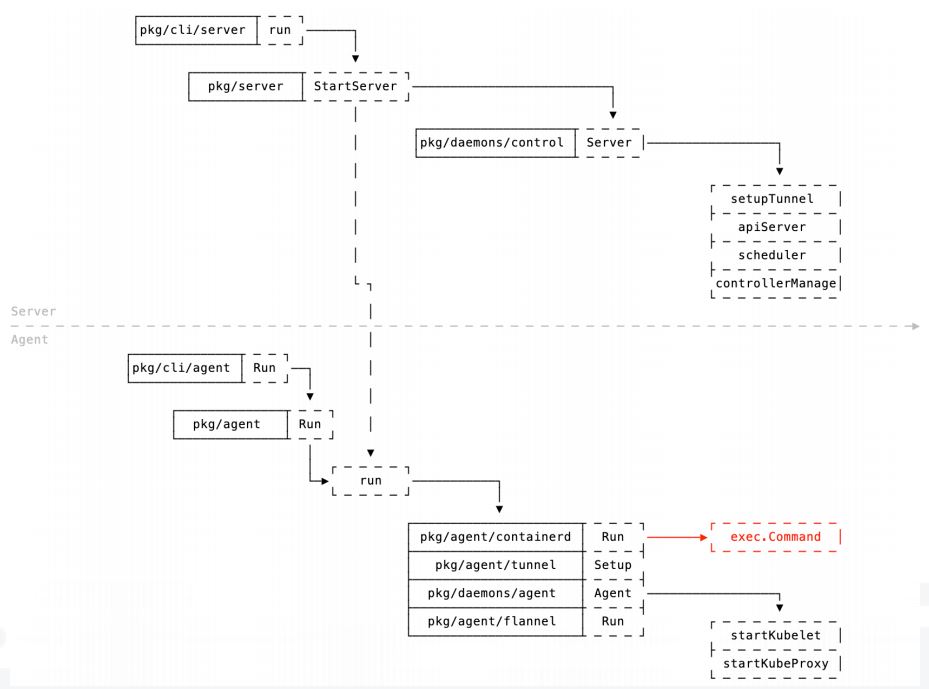
1.1.2.1 轻量化

删除了一些旧的非必要的代码；将一些组件打包到一起（一个进程）；使用containerd作为容器运行时；引入SQLite作为可选的存储源。

删除了：旧的、非必要的、Alpha的功能，大部分in-tree插件（云提供商和存储插件），将其用附加组件进行替换，Docker(可选）。

增加了：简易化安装，SQLite3支持，TLS管理，自动化的 manifest 和 Helm Chart 管理，Containerd、CoreDns、Flannel。

1.1.2.2 代码路径图



1.1.3注意事项

containerd 是一个独立的运行进程；

Tunnel Proxy负责维护k3s Server和k3s Agent之间的连接，采用Basic Auth的方式进行认证；

k3s Server是一个进程，包含多个线程。其中端口号6443是k3s-server用，6444是api-server用。

K3s Agent有两个进程，它的进程树是在k3s-agent进程下，有一个子进程containerd。

1.1.4参考链接

k3s代码组织结构分析参考： https://mp.weixin.qq.com/s/1o9X0Dlv2WhUS6iC9P-KQA

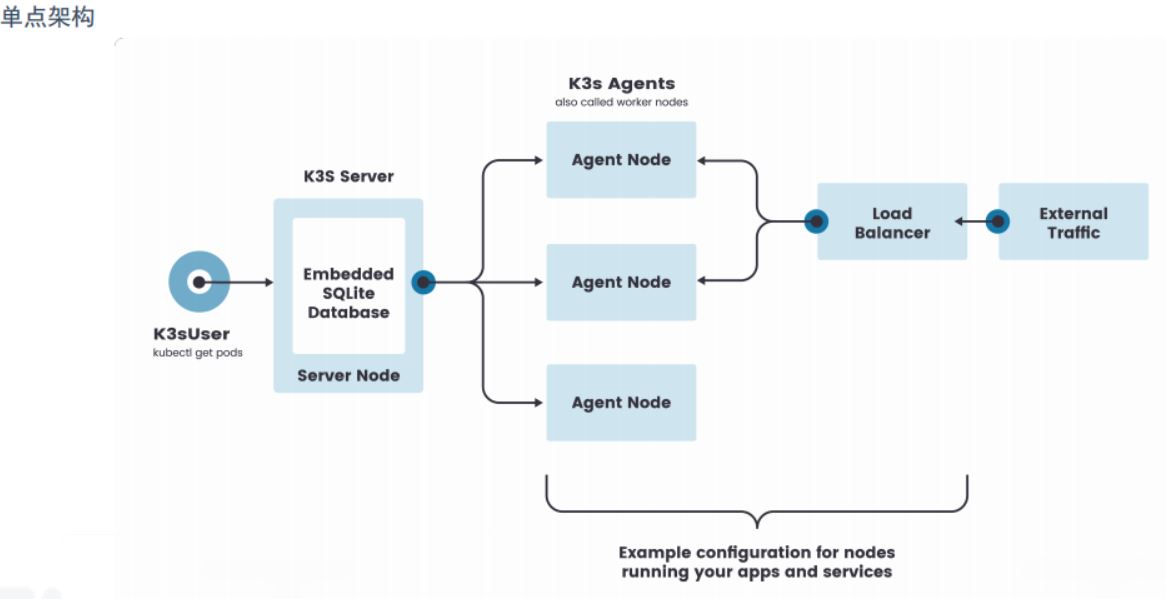
k3s部署参与： https://mp.weixin.qq.com/s/V-VyWrCZux5WXxD\_\_QpEWQ

k3s为何轻量： https://mp.weixin.qq.com/s/5aprEfYSWJyVoW4trDp4Hw

1.2高可用

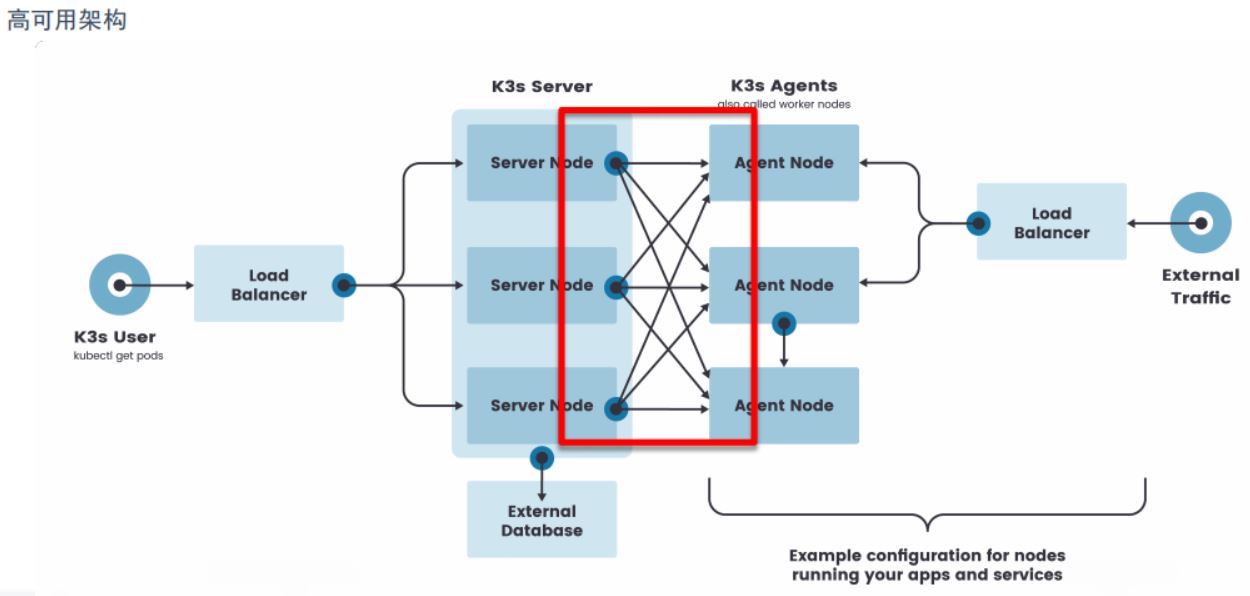
1.2.1从单点架构到高可用架构

单点架构k3s-server内置数据库SQLite，面临的风险是当k3s-server宕机之后，整个集群可能会停止工作，同时k3s-server会丢失节点数据。



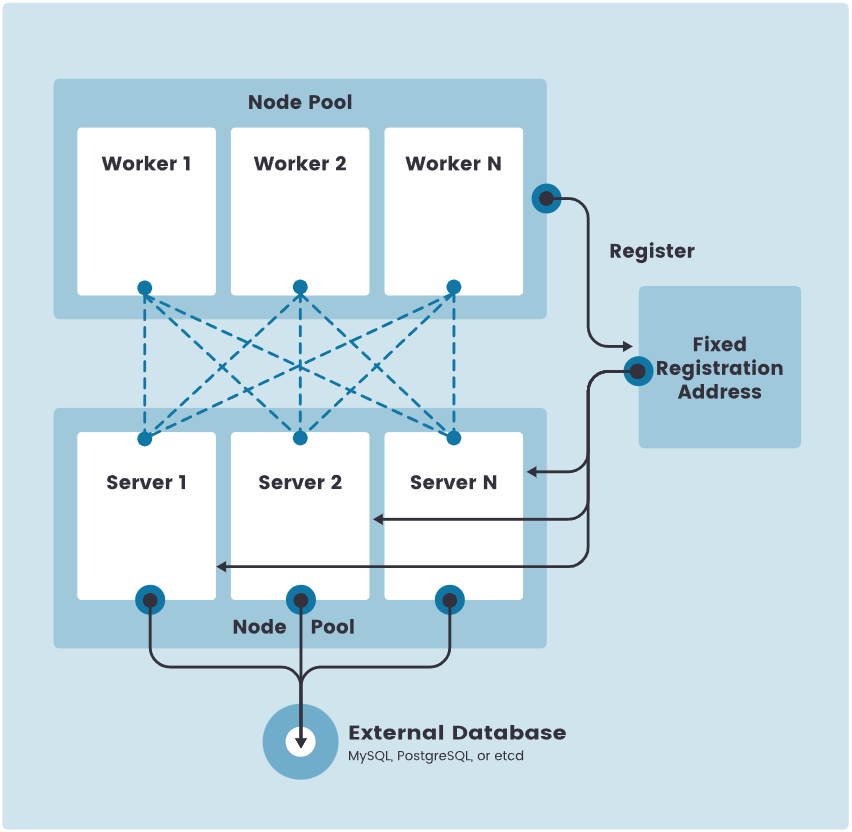
高可用架构采取多个k3s-server对多个agent的结构，可以采取L4层负载均衡、Round-robin DNS 或者弹性IP等方法，将server的ip地址固定，然后agent访问该ip即可。并采用外置数据库或者分布式数据库，避免数据因为节点宕机而丢失。

在高可用架构下，某一个server-node宕机后，整个集群不受影响，访问会转移到其他server-node上。



1.2.2 固定agent节点的注册地址

在高可用k3s server配置中，每个节点必须使用固定的注册地址向kubernetes API注册，注册后，agent节点直接与其中一个server节点建立连接。



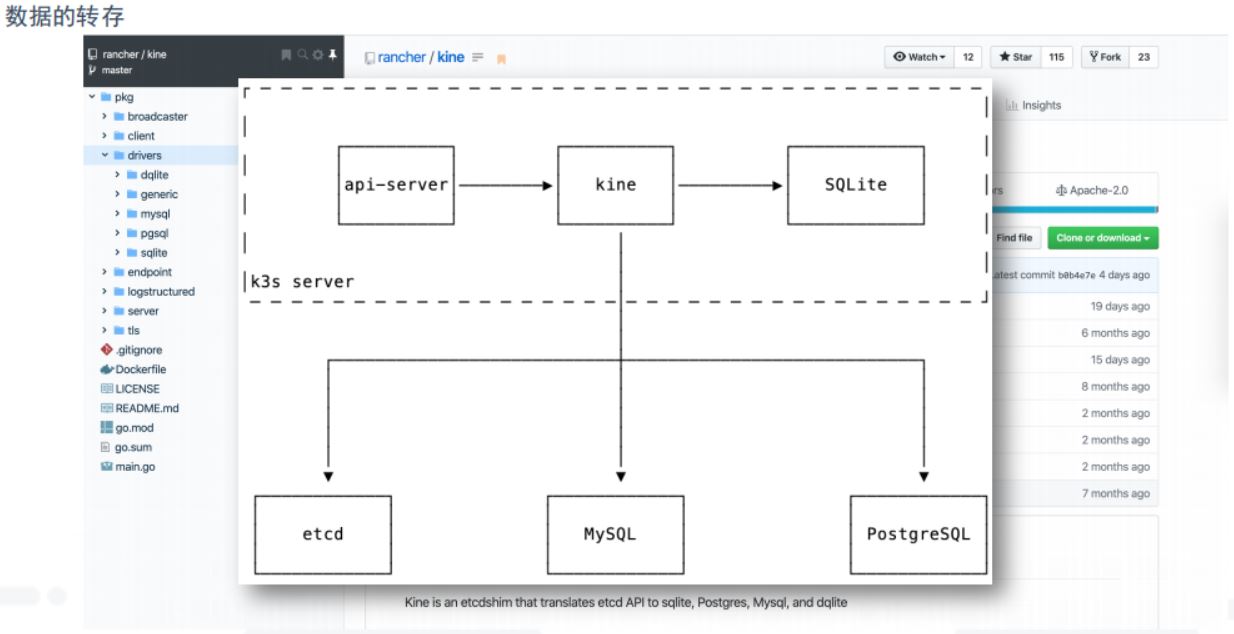
1.2.3使用外置数据库

外置数据源支持PostgreSQL (v10.7、v11.5)、MySQL (v5.7)、etcd (v3.3.15)等。

为了实现将数据从api-server转储到关系型数据库，rancher/Kine开源项目实现了etcd-v3的存储接口，并根据对应的数据源进行转存，kine可以理解为是一个适配器。

api-server通过和kine打交道，既可以向SQLite读写数据，也可以向外部的数据源读写数据。

至少应提供两个节点安置k3s server，还需要一个稳定的外置关系型数据库数据源。



演示build HA k3s with external db: https://asciinema.org/a/OFaQIHuye4AU5QBaEgzUFq9ql

1.2.4 高可用架构的局限性

需要额外的资源支持，比如购买公有云服务数据源；

需要运维人员对数据库进行维护；

可能面临不可以的情况，比如资源不足、数据库被删除等。

1.2.5 使用分布式数据库（试验性）

1.2.5.1 SQLite

SQLite的特点：

软件库，无独立进程：C 语言编写；（使用时通过动态或静态链接，kine采用静态链接使用SQLite软件库）

零管理配置：不需要服务管理进程；

事务安全：兼容ACID，可并发访问；

标准SQL支持；

单一磁盘文件；（k3s server节点中的 /var/lib/rancher/k3s/server/db/state.db）

1.2.5.2 Dqlite

Dqlite的特点：

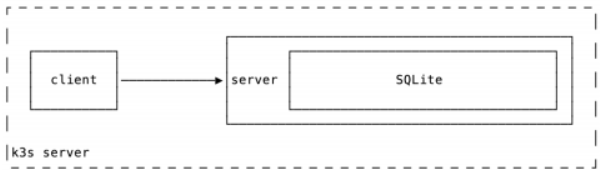
软件库，无独立进程：C 语言编写；

分布式一致性协议：C-Raft实现；（在SQLite基础上重新实现了一组链接库）

兼容SQLite；

Dqlite的原理：

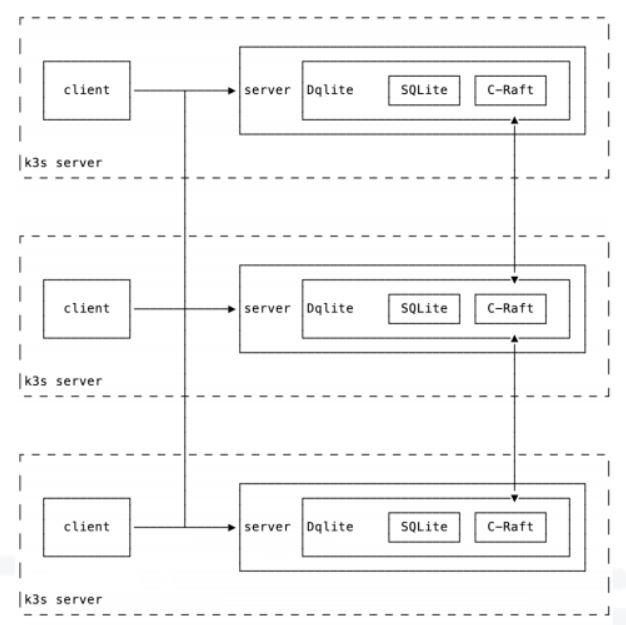
在使用SQLite时，一个k3s server进程内含SQL的client组件和server组件；client 仅连接一个server；server 链接 SQLite 软件库。



在使用Dqlite时，一个k3s server进程内含SQL的client组件和server组件，而server链接了Dqlite软件库，Dqlite软件库包含SQLite和C-Raft两个部分。C-Raft会进行分布式的通信，确保数据一致性。

Dqlite要求有奇数个k3s server进程，且client需要链接所有server。第二个及以后的server需要第一个server的token，被第一个server视为agent加到集群里。

当集群启动后，Server会通过C-Raft进行选主（默认是加进集群的第一个节点，当第一个节点‘死掉’后，server才会进行选主），client通过询问发现主server，之后client把请求都发送到主server，主server通过C-Raft给从server发送差分日志作为数据同步。



演示build HA k3s with dqlite: <https://asciinema.org/a/CtIdSei6XtO5x6DoLiJjvc5qe>

1.2.6参考链接

SQLite 基础命令： https://www.runoob.com/sqlite/sqlite-commands.html

SQLite Go语言驱动实现： https://github.com/mattn/go-sqlite3

CGO的动静态链接介绍： https://books.studygolang.com/advanced-go-programming-

book/ch2-cgo/ch2-06-static-shared-lib.html

Dqlite 讲解： https://fosdem.org/2020/schedule/event/dqlite/

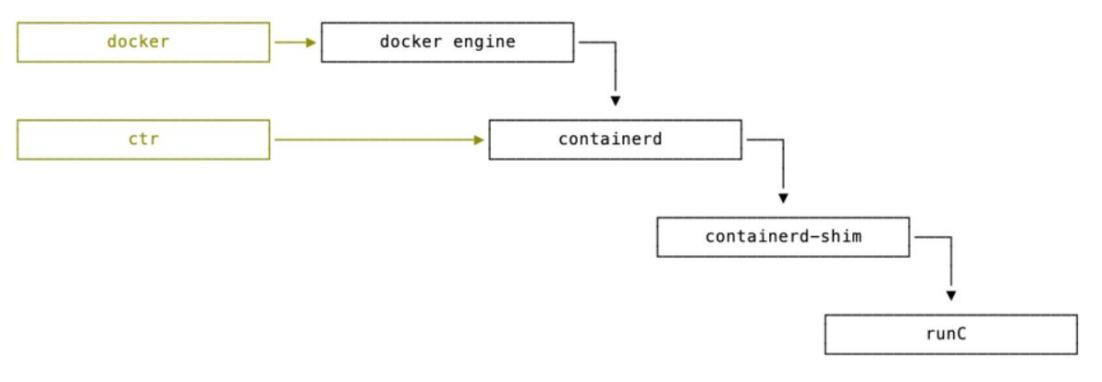
Raft 协议介绍： https://raft.github.io/

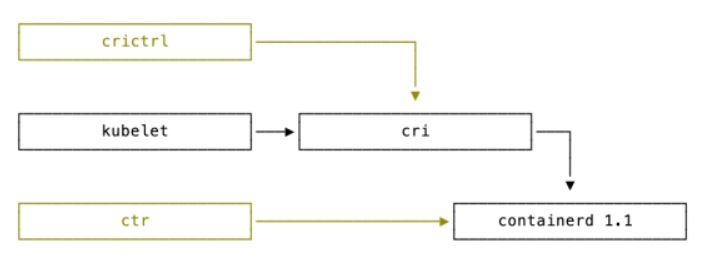
k3s 社区高可用介绍： https://mp.weixin.qq.com/s/3by95UIJ7v41KXElt7uvGA

1.3 Containerd

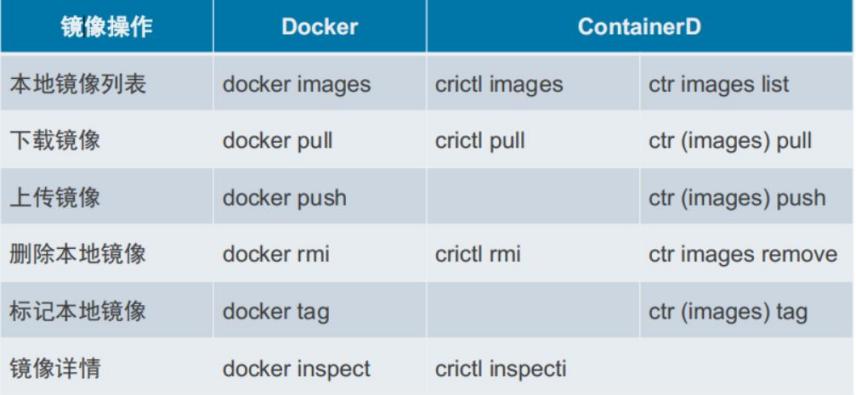
containerd被设计目的是嵌套到更大的系统里去，而不是被开发者或终端用户直接使用。

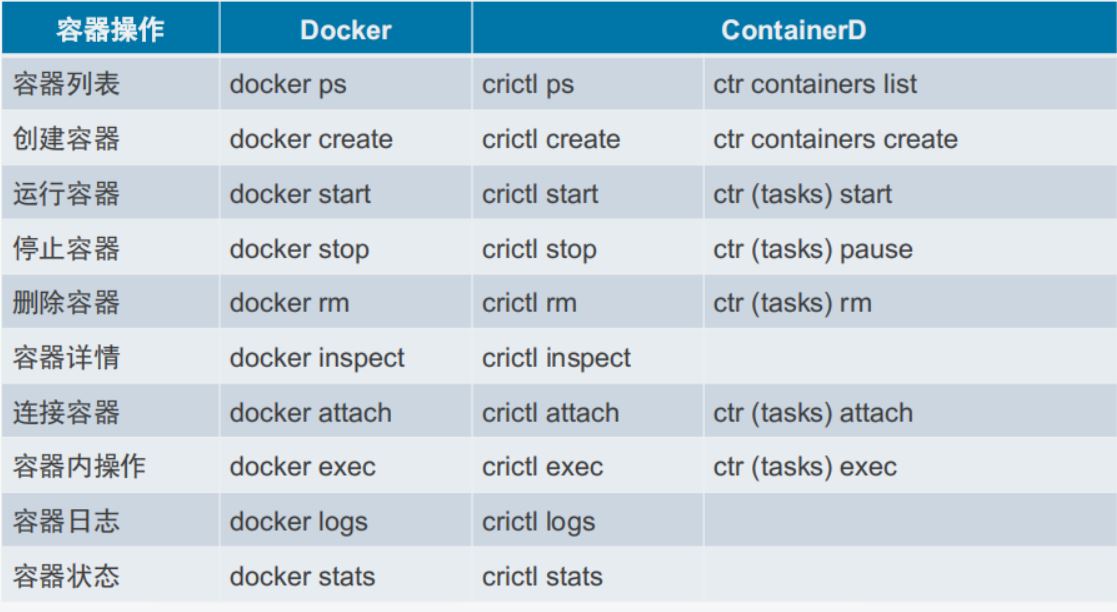
containerd调用路径:

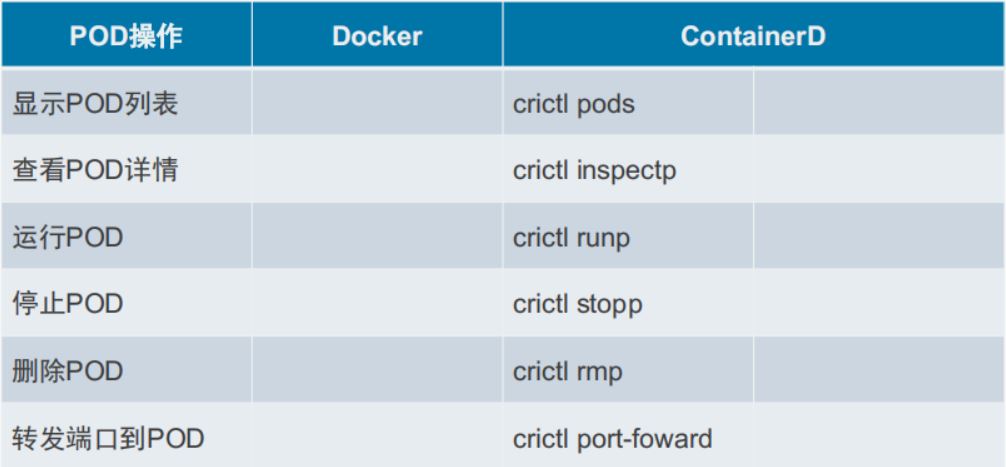




1.3.1操作（使用ctr和crictl）







1.3.2 参考链接

cri的演进史：https://zhuanlan.zhihu.com/p/87602649

runC的介绍：https://www.infoq.cn/article/docker-standard-container-execution-engine-runc

k3s 社区 Containerd 使用介绍： https://mp.weixin.qq.com/s/EJqS7G36f7\_srgxSZaCuPw

crictl 基本操作： https://github.com/kubernetes-sigs/cri-tools/blob/master/docs/crictl.md

Containerd CNI插件配置介绍： https://github.com/containerd/cri/blob/master/docs/config.md

1. 安装k3s

2.1.参考链接和安装选项

K3s安装文档：<https://docs.rancher.cn/docs/k3s/installation/_index>

K3s安装选项文档：<https://docs.rancher.cn/docs/k3s/installation/install-options/_index>

K3s资源分析文档：<https://docs.rancher.cn/docs/k3s/installation/installation-requirements/resource-profiling/_index> （具有工作负载的 K3s server、具有单个 agent 的 K3s 集群、K3s agent最低资源需求）

安装脚本：curl -sfL https://get.k3s.io | sh -

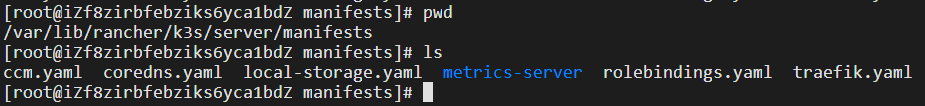
使用install.sh脚本安装了 K3s，在安装过程中会生成一个卸载脚本到节点。

从server节点卸载 K3s：/usr/local/bin/k3s-uninstall.sh

从agent节点卸载 K3s：/usr/local/bin/k3s-agent-uninstall.sh

（卸载 K3s 会删除集群数据和所有脚本。要使用不同的安装选项重新启动集群，需使用不同的选项重新运行安装脚本。）

默认的配置文件在/var/lib/rancher/k3s/server/manifests 目录下，该目录下的yaml文件会自动部署。



2.2 k3s网络选项

默认情况下，K3s 将以 flannel 作为 CNI 运行，使用 VXLAN 作为默认后端。（开放端口号UDP 8472）

安装时的网络选项： --flannel-backend value value值可以为'none', 'vxlan', 'ipsec', 'host-gw', 或 'wireguard'中的一个，默认为vxlan。使用值none时允许自定义CNI：

$ curl -sfL https://get.k3s.io | sh -s - --flannel-backend none

2.3 外加数据库的高可用k3s集群部署

2.3.1 k3s HA 集群组成

一个 K3s HA 集群由以下几个部分组成：

（1）两个或多个server节点，将为 Kubernetes API 提供服务并运行其他 control-plane 服务。

（2）零个或多个agent 节点，用于运行应用和服务。

（3）外部数据存储 。

（4）固定的注册地址，位于 server节点的前面，以允许agent节点向集群注册。

Agent 通过固定的注册地址进行注册，但注册后直接与其中一个 server节点建立连接。这是一个由k3s agent进程发起的 websocket 连接。

2.3.2 创建外部存储并启动k3s server节点

当在多个节点上运行k3s server命令时，必须设置datastore-endpoint参数，以便 K3s 知道如何连接到外部数据存储。启动server时，可以选择设置TLS证书（cafile、certfile、keyfile）。

K3s 支持以下数据存储选项：嵌入式 [SQLite](https://www.sqlite.org/index.html" \t "https://docs.rancher.cn/docs/k3s/installation/datastore/_blank)， [PostgreSQL](https://www.postgresql.org/" \t "https://docs.rancher.cn/docs/k3s/installation/datastore/_blank) (经过认证的版本：10.7 和 11.5)，[MySQL](https://www.mysql.com/" \t "https://docs.rancher.cn/docs/k3s/installation/datastore/_blank) (经过认证的版本：5.7)，[MariaDB](https://mariadb.org/" \t "https://docs.rancher.cn/docs/k3s/installation/datastore/_blank) (经过认证的版本：10.3.20)，[etcd](https://etcd.io/" \t "https://docs.rancher.cn/docs/k3s/installation/datastore/_blank) (经过认证的版本：3.3.15)，嵌入式 etcd 高可用。

最常见的 PostgreSQL 数据存储端点的参数格式：postgres://username:password@hostname:port/database-name

最常见的 MySQL 和 MariaDB 的datastore-endpoint参数格式如下：

mysql://username:password@tcp(hostname:3306)/database-name

最常见的 etcd （三节点 etcd 集群）的datastore-endpoint参数的格式如下：

https://etcd-host-1:2379,https://etcd-host-2:2379,https://etcd-host-3:2379

以连接外部mysql数据库为例：

curl -sfL https://get.k3s.io | sh -s - server \

--datastore-endpoint="mysql://username:password@tcp(hostname:3306)/database-name"

2.3.3 配置固定的注册地址

Agent 节点需要一个 URL 来注册。这可以是任何 server 节点的 IP 或主机名，但在许多情况下，这些节点可能会随着时间的推移而改变。

应该在 server 节点前面有一个稳定的端点，不会随时间推移而改变。可以使用许多方法来设置此端点，例如：一个 4 层（TCP）负载均衡器；轮询 DNS；虚拟或弹性 IP 地址。

2.3.4 Agent节点注册

 HA K3s server 集群的最小节点数是两个 server 节点和零个 agent 节点。要添加agent节点，需要指定 agent 应该注册到的 URL 和它应该使用的 token 。

K3S\_TOKEN=SECRET k3s agent --server <https://fixed-registration-address:6443>

2.3.5其他：嵌入式DB的高可用

从 v1.19.5+k3s1 版本开始，K3s 已添加了对嵌入式 etcd 的完全支持。要在这种模式下运行 K3s，必须有奇数的服务器节点（选主）。

（1）要开始运行，首先启动一个服务器节点，使用cluster-init标志来启用集群，并使用一个标记作为共享的密钥来加入其他服务器到集群中。

K3S\_TOKEN=SECRET k3s server --cluster-init

（2）启动第一台服务器后，使用共享密钥将第二台和第三台服务器加入集群。

K3S\_TOKEN=SECRET k3s server --server https://<ip or hostname of server1>:6443

2.3.6 关于禁用组件选项：

使用 --cluster-init 参数启动 K3s server 时，它会运行包括 API Server、Controller Manager、Scheduler 和 ETCD 在内的所有 controlplane 组件。

这个命令会生成一个只有 etcd 的 Server 节点：

curl -fL https://get.k3s.io | sh -s - server --cluster-init --disable-apiserver --disable-controller-manager --disable-scheduler

1. k3s网络部分

3.1 coreDNS

CoreDNS 是在agent节点启动时部署的。要禁用，需要运行--disable coredns选项。禁用coreDNS后需要自行安装一个集群DNS提供商。

3.2 Traefik Ingress Controller

[Traefik](https://traefik.io/" \t "https://docs.rancher.cn/docs/k3s/networking/_blank)是一个现代的 HTTP 反向代理和负载均衡器，它是为了轻松部署微服务而生的。在设计，部署和运行应用程序时，它简化了网络复杂性。

启动 server 时，默认情况下会部署 Traefik（通过Helm Controller定义）。默认配置文件在/var/lib/rancher/k3s/server/manifests/traefik.yaml中（/var/lib/rancher/k3s/server/ /manifests/目录下的yaml自动部署）。Traefik ingress controller 将使用主机上的80和443端口。

不应该手动编辑 traefik.yaml文件，因为 k3s 一旦重启就会再次覆盖它。可以通过在/var/lib/rancher/k3s/server/manifests中创建一个额外的HelmChartConfig清单来定制 Traefik。

从 v1.19.0+k3s1 开始的 K3s 版本支持通过 HelmChartConfig 资源自定义部署。

3.3 Service Load Balancer

K3s 提供了一个名为[Klipper Load Balancer](https://github.com/rancher/klipper-lb" \t "https://docs.rancher.cn/docs/k3s/networking/_blank)（https://github.com/rancher/klipper-lb）的负载均衡器，它可以使用可用的主机端口。K3s service LB 使得可以在没有云提供商的情况下使用 LB 服务。

[Klipper Service Load Balancer](https://github.com/rancher/klipper-lb" \t "https://docs.rancher.cn/docs/k3s/networking/_blank)：This is the runtime image for the integrated service load balancer in klipper. This works by using a host port for each service load balancer and setting up iptables to forward the request to the cluster IP. The regular k8s scheduler will find a free host port. If there are no free host ports, the service load balancer will stay in pending.

3.3.1 Service LB工作原理

对于每个 service load balancer，都会创建一个[DaemonSet](https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/" \t "https://docs.rancher.cn/docs/k3s/networking/_blank)。 DaemonSet 在每个节点上创建一个前缀为svc的 Pod。

Service LB 控制器会监听其他 Kubernetes Services。当它找到一个 Service 后，它会在所有节点上使用 DaemonSet 为该服务创建一个代理 Pod。这个 Pod 成为其他 Service 的代理，例如，来自节点上 8000 端口的请求可以被路由到端口 8888 上的工作负载。

如果 Service LB 运行在有外部 IP 的节点上，则使用外部 IP。

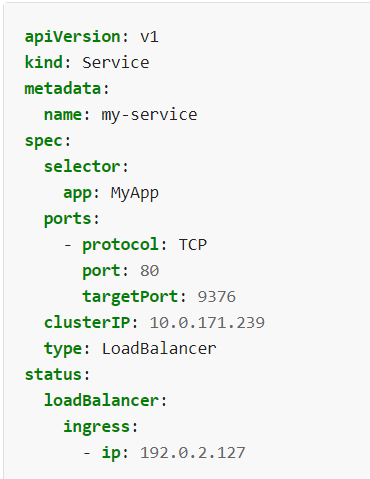
如果创建多个 Services，则为每个 Service 创建一个单独的 DaemonSet。

只要使用不同的端口，就可以在同一节点上运行多个 Services。

如果尝试创建一个在 80 端口上监听的 Service LB，Service LB 将尝试在集群中找到 80 端口的空闲主机。如果该端口没有可用的主机，LB 将保持 Pending 状态。

3.3.2 用法

在k3s中创建一个[LoadBalancer 类型的 Service](https://kubernetes.io/docs/concepts/services-networking/service/" \l "loadbalancer" \t "https://docs.rancher.cn/docs/k3s/networking/_blank)。



3.3.3 从节点中排除Service LB

如果使用标签，则 service load balancer 仅在标记的节点上运行。因此要在不应别排除的节点上增加标签：svccontroller.k3s.cattle.io/enablelb。

禁用Service LB需使用--disable servicelb选项运行 k3s server。

1. K3s Helm包管理工具（略）

Helm 是 Kubernetes 的首选包管理工具。Helm Chart 为 Kubernetes YAML 清单文件提供了模板化语法。通过 Helm可以创建可配置的部署。K3s 包括一个[Helm Controller](https://github.com/rancher/helm-controller/" \t "https://docs.rancher.cn/docs/k3s/helm/_blank)（ A simple way to manage helm charts (v2 and v3) with Custom Resource Definitions in k8s）（https://github.com/k3s-io/helm-controller）。

HelmChart字段定义（Helm CRD）（作为helm参数）：



1. 其他k3s 常见问题：

<https://docs.rancher.cn/docs/k3s/faq/_index>

5.1节点注册：

节点注册到 k3s 集群，会在节点的/etc/rancher/node/password生成一串随机的 password。如果 agent 首次注册，master 节点会把 agent 发送的 node-name 和 node-passwd 解析出来存储到/var/lib/rancher/k3s/server/cred/node-passwd 中。如果 agent 是非首次注册，k3s master 会结合 node-name 和 node-passwd 进行比对，如果信息不一致会拒绝添加节点请求。

如果用 k3s-agent-uninstall.sh 来清理安装过的 agent node，并不会删除 password 文件（/etc/rancher/node/password）。如果手动操作删除这个password文件，agent 再次注册时会重新生成 password，就导致了新的 password 和 k3s master 上原先存储的不一致。（原agent 节点node name 没有变）

手动在 agent 上创建 password，内容和 server 中存储保持一致；修改 server 中的原始内容，让 password 和 agent 上新生成的保持一致；可以试试 agent 注册时使用--with-node-id将 ID 附加到节点名称，这样 server 中认为这完全是新 node，不会用原始信息比对。

5.2 对Helm 、Chart的补充（略）

文档：<https://helm.sh/zh/docs/>

引用链接：<https://www.jianshu.com/p/aff467690d57>

chart结构示例：

$ helm create mongodb

$ tree mongodb

mongodb

├── Chart.yaml #Chart本身的版本和配置信息

├── charts #依赖的chart

├── templates #配置模板目录

│ ├── NOTES.txt #helm提示信息

│ ├── \_helpers.tpl #用于修改kubernetes objcet配置的模板

│ ├── deployment.yaml #kubernetes Deployment object

│ └── service.yaml #kubernetes Serivce

└── values.yaml #kubernetes object configuration

2 directories, 6 files

1. Rancher介绍

Rancher文档：<https://docs.rancher.cn/docs/rancher2.5/overview/_index>

6.1 什么是Rancher?

“Rancher 可以创建来自 Kubernetes 托管服务提供商的集群，自动创建节点并安装 Kubernetes 集群，或者导入任何已经存在的 Kubernetes 集群。

Rancher 通过支持集群的身份验证和基于角色的访问控制（RBAC），使系统管理员能够从一个位置控制全部集群的访问。Rancher 可以对集群及其资源进行详细的监控和并在需要时发送告警，也可以将容器日志发送给外部日志系统，并通过应用商店与 Helm 集成。如果您具有外部 CI/CD 流水线系统，则可以将其与 Rancher 对接，如果没有，Rancher 也提供了简单易用的流水线来帮助您自动部署和升级工作负载。除此之外，Rancher 还有很多开箱即用的功能来帮助您更好的管理集群和业务应用，例如多集群应用，全局 DNS，服务网格，安全扫描，集群模版和基于 OPA 的策略管理等功能。”

6.2 Rancher提供的功能

对于已有集群而言，启动集群的方法决定了可编辑的选项和设置。例如，只有通过 RKE 启动的集群才有可编辑的集群选项。



6.3 Rancher supported product version 对照表（支持维护条款）

<https://www.suse.com/suse-rancher/support-matrix/all-supported-versions/rancher-v2-6-2/>

1. Rancher产品架构

7.1 Rancher Server架构

Rancher Server 由认证代理（Authentication Proxy）、Rancher API Server、集群控制器（Cluster Controller）、etcd 节点和集群 Agent（Cluster Agent） 组成。除了集群 Agent 以外，其他组件都部署在 Rancher Server 中。

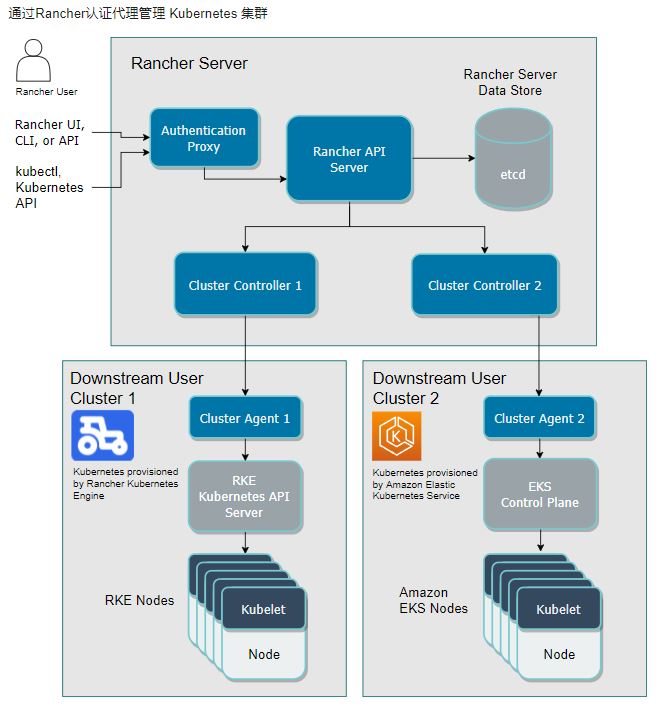
下图描述的是用户通过 Rancher Server 管控 Rancher 部署的 Kubernetes 集群（RKE 集群）和托管的 Kubernetes 集群的（EKS）集群的流程。以用户下发指令为例，指令的流动路径如下：

（1）首先，用户通过 Rancher UI（即 Rancher 控制台） Rancher 命令行工具（Rancher CLI）输入指令；直接调用 Rancher API 接口也可以达到相同的效果。

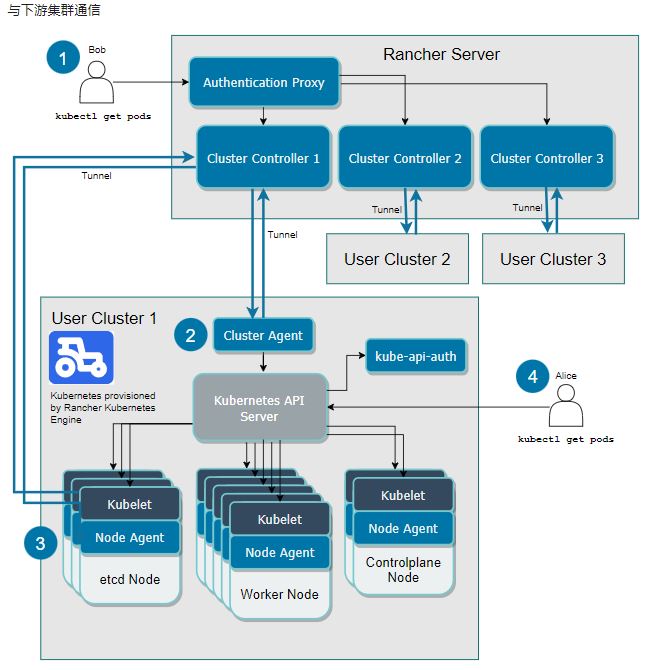
（2）用户通过 Rancher 的代理认证后，指令会进一步下发到 Rancher Server 。

（3）与此同时，Rancher Server 也会执行容灾备份，将数据备份到 etcd 节点。

（4）然后 Rancher Server 把指令传递给集群控制器。集群控制器把指令传递到下游集群的 Agent，最终通过 Agent 把指令下发到指定的集群中。



下图演示了集群控制器、集群 Agent 和 Node Agent 是如何允许 Rancher 控制下游集群的。



图中的数字和对应的描述如下：

（1）[认证代理](https://docs.rancher.cn/docs/rancher2.5/overview/architecture/_index" \l "%E8%AE%A4%E8%AF%81%E4%BB%A3%E7%90%86)

（2）[集群控制器和集群 Agent](https://docs.rancher.cn/docs/rancher2.5/overview/architecture/_index" \l "%E9%9B%86%E7%BE%A4%E6%8E%A7%E5%88%B6%E5%99%A8%E5%92%8C%E9%9B%86%E7%BE%A4-agent)

（3）[节点 Agents](https://docs.rancher.cn/docs/rancher2.5/overview/architecture/_index" \l "%E8%8A%82%E7%82%B9-agent)

（4）[授权集群端点](https://docs.rancher.cn/docs/rancher2.5/overview/architecture/_index" \l "%E6%8E%88%E6%9D%83%E9%9B%86%E7%BE%A4%E7%AB%AF%E7%82%B9)

7.1.1认证代理

图左上角一个叫做 Bob 的用户希望查看下游集群“User Cluster 1”里面正在运行的 pod。Bob 发起的请求会首先经过认证代理，通过认证之后，Rancher 的 认证代理才会把 API 调用命令转发到下游集群。

认证代理集成了多种认证方式，如本地认证、活动目录认证、GitHub 认证等。在发起每一个 Kubernetes API 调用请求的时候，认证代理会去确认请求方的身份。

Rancher 使用 [Service Account](https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/" \t "https://docs.rancher.cn/docs/rancher2.5/overview/architecture/_blank) （Service Accout 提供了一种方便的认证机制）和 Kubernetes 进行交互。

默认状态下，Rancher 生成一个包含认证信息的[kubeconfig](https://docs.rancher.cn/docs/rancher2.5/cluster-admin/cluster-access/kubectl/_index)文件，为 Rancher Server 和下游集群的 Kubernetes API Server 之间的通信提供认证。该文件包含了访问集群的所有权限。

7.1.2 集群控制器和集群 Agent

每一个下游集群都有一个集群 Agent 保持下游集群的集群控制器与 Rancher Server 之间的信息畅通。默认状态下，集群控制器连接 Agent，Rancher 才可以与下游集群通信。

集群控制器具有以下功能：

（1）检测下游集群的资源变化，如内存使用率、CPU 使用率等

（2）把下游集群从“当前”状态变更到“目标”状态

（3）配置集群和项目的访问控制策略

（4）通过调用 Docker Machine 和 Kubernetes Engine，如 RKE 和 GKE，创建集群。

集群 Agent，也叫做“cattle-cluster-agent”，是在下游集群中运行的组件，它具有以下功能：

（1）连接使用 Rancher 部署的 Kubernetes 集群（RKE 集群）中的 Kubernetes API。

（2）管理集群内的工作负载，pod 创建和部署。

（3）根据每个集群的设置，配置 Role 和 RoleBindings

（4）实现集群和 Rancher Server 之间的消息传输，包括事件，指标，健康状况和节点信息等。

7.1.3节点 Agent

如果集群 Agent 不可用，下游集群中的其中一个节点 Agent 会创建一个通信管道，由节点 Agent 连接到集群控制器，实现下游集群和 Rancher 之间的通信。

部署节点 Agent 的方式有很多，可以使用[DaemonSet](https://kubernetes.io/docs/concepts/workloads/Controllers/daemonset/" \t "https://docs.rancher.cn/docs/rancher2.5/overview/architecture/_blank)部署节点 Agent ，这种方式可以确保下游集群内每个节点都成功运行节点 Agent。

7.1.4 授权集群端点

Rancher Server 和下游集群之间有明显的延迟，或 Rancher Server 不可用时，用户可以通过授权集群端点连接下游集群，实现 Rancher Server 和集群之间的通信，降低网络延迟。

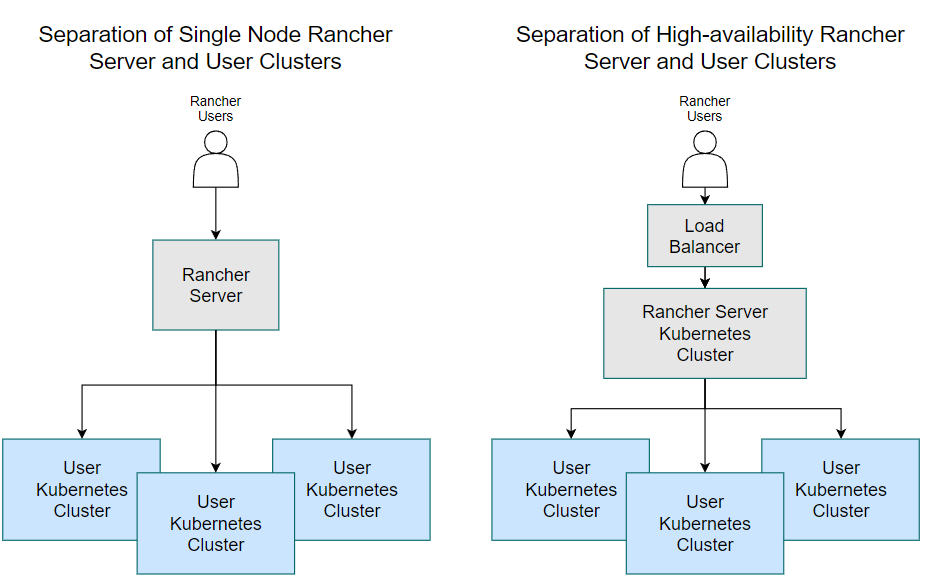
需要注意的是，只有 Rancher 部署的 Kubernetes 集群（RKE 集群）可以使用授权集群端点这个功能。

7.2高可用集群与Rancher

将 Rancher Server 安装在高可用的 Kubernetes 集群上，这可以保护 Rancher Server 的数据。在高可用安装中，负载均衡器充当客户端的单点入口，并在集群中的多台服务器之间分配网络流量，这有助于防止任何一台服务器成为单点故障。

Rancher 需要安装在高可用的 [RKE（Rancher Kubernetes Engine）](https://docs.rancher.cn/docs/rke/_index)Kubernetes 集群上，或高可用的[K3s (轻量级 Kubernetes)](https://docs.rancher.cn/docs/k3s/_index)Kubernetes 集群。 RKE 和 K3s 都是经过完全认证的 Kubernetes 发行版。

 K3s 架构的一个主要优点是，它允许使用外部数据库保存集群数据，从而可以将 K3s 服务器节点视为无状态的。在 K3s 集群上安装 Rancher 的功能是在 Rancher v2.4 中引入的。



7.3 为 Rancher 设置高可用 K3s Kubernetes 集群

7.3.1先决条件

要在高可用性 K3s 集群上安装 Rancher 管理服务器，我们建议设置以下基础设施。

两个 Linux 节点，通常是虚拟机，在您选择的基础设施提供商中。

一个外部数据库，用于存储集群数据。建议使用 MySQL。

一个负载均衡器，将流量引导到两个节点。

一个 DNS 记录，用于将一个 URL 映射到负载均衡器。这将成为 Rancher Server 的 URL，下游的 Kubernetes 集群需要访问这个 Rancher Server 地址。

7.3.2设置k3s集群（两个节点只允许server节点即可）

在两个Linux节点上分别运行

curl -sfL https://get.k3s.io | sh -s - server \

--datastore-endpoint="mysql://username:password@tcp(hostname:3306)/database-name"

这样就构建了一个k3s集群。运行以下命令验证，看到两个具有 master role 的节点说明k3s设置成功。

k3s kubectl get nodes

k3s kubectl get pods --all-namespaces

7.3.3保存并开始使用 kubeconfig 文件

当在每个 Rancher server 节点上安装 K3s 时，会在节点上创建了一个/etc/rancher/k3s/k3s.yaml的kubeconfig文件。这个文件包含了完全访问集群的凭证。

（1）安装[kubectl]（[https://kubernetes.io/docs/tasks/tools/install-kubectl/#install-kubectl）。](https://kubernetes.io/docs/tasks/tools/install-kubectl/" \l "install-kubectl%EF%BC%89%E3%80%82" \t "https://docs.rancher.cn/docs/rancher2.5/installation/resources/k8s-tutorials/ha-with-external-db/_blank)

（2）复制/etc/rancher/k3s/k3s.yaml处的文件，并保存到本地机器上的~/.kube/config目录下。

（3）在 kubeconfig 文件中，server指令定义为 localhost。将服务器配置为负载均衡器的 DNS，参考端口 6443。Kubernetes API 服务器将通过端口 6443 到达，而 Rancher 服务器将通过端口 80 和 443 到达）。

现在可以使用kubectl在本机来管理 K3s 集群。如果有多个的 kubeconfig 文件，可以在使用kubectl时通过传递文件的路径来指定你要使用的文件。

kubectl --kubeconfig ~/.kube/config/first\_k3s.yaml get pods --all-namespaces

7.3.4 在集群上安装 Rancher

在k3s集群上使用helm安装或者启动一个Pod（运行rancher的docker）即可。

1. Kubeedge 设备管理设计原理

<https://www.bilibili.com/video/BV1LJ411D7t1?p=4>