Docker 版本及内...

这是本专栏的第一部分：Docker 入门，共 3 篇，帮助大家进入 Docker 的世界。上一篇，我带大家了解了 Docker 入门的基础知识，知道了如何运行和操作容器。下面我们一起进入第三篇的内容。

Docker 的版本历程

快速迭代期

在第一篇，我们已经知道 Docker 是在 2013 年首次亮相，在 2014 年 6 月 9 日发布了 1.0.0 版本；**直到 2017 年，Docker 的版本号都是 X.Y.Z 这样的格式。**差不多每 2~3 个月会升一次 Y 的版本号，比如：1.3.0 是 2014 年 10 月发布的，1.4.0 则是 2014 年 12 月发布。

2017 年 2 月 8 日 Docker 发布了 1.13.1 版本，这是 **最后一个 以 X.Y.Z 格式命名的版本**了。

在这个阶段，Docker 在主流 Linux 发行版上的安装包一般叫做 docker、docker-engine 或 docker.io，所以现在如果你需要安装新版本 Docker 的话，如果你已经安装了上述包中的任何一个，一般会建议你先删除掉。

同年 Docker 宣布将原 GitHub 上开源的 Docker 项目 更名为 Moby，这次更名之后，产生了几个重要的变化：

- Moby 项目将作为上游的开源开发项目
- Docker CE 是基于 Moby 的免费发布版本（也是现在大家最常用的）
- Docker EE 是基于 Docker CE 的商业产品版本

现在对于我们而言，可能上游项目叫 Docker 或者叫 Moby 已经不会太在意了，但实际上这些改动带来的影响是将原本统一的项目分拆成了很多不同的组件，之后再进行组织，这当然也会为理解 Docker 代码等带来一定的复杂度。在本次专栏的最后一部分，我会与你分享 Docker 上游项目的组织方式，以及如何参与到 Docker 项目中去。

蓬勃发展期

到 2017 年时，Docker 已经面世 4 年左右的时间，这期间 Docker 迅速攀升成了新时代的技术趋势。而 Docker 的主体功能也已经相对完善。随着 Docker Inc. 有了企业级产品，这时候 Docker 的版本号命名规则也随之发生了变化。

2017 年 3 月，Docker 发布了 Docker CE 17.03.0-ce 和 Docker EE 17.03.0-ee-1。从此之后，Docker 开始了蓬勃发展期，**发布周期改成了每月一次，同时版本命名也换成了 YY.MM.<patch> 的形式** 当然也为了考虑到社区用户在实际使用中可能不会很频繁的升级，所以提供过了两种渠道：

- 月度更新：这便是我刚才所说的那种方式，每月更新，但是每月更新的这个版本只会在下个版本发布之前得到安全更新和错误修正；在下个版本发布后，便不再更新了。例如：17.04.0-ce 在 17.05.0-ce 发布前都能得到安全更新和错误修正的支持，一旦 17.05.0-ce 发布后，17.04.0-ce 就将结束生命周期，不再更新。
- 季度更新：指的是每个季度的首个版本，在发布之后，在 4 个月内都将收到安全更新和错误修正。例如：17.06.0-ce 在 17.09.0-ce 发布之前，这 4 个月内都可以得到安全更新和错误修正，一旦 17.09.0-ce 发布后，17.06.0-ce 就将结束生命周期，不再更新。

所以**在这个阶段，我会建议你选择季度更新的版本**，因为其能得到的支持时间最长，也不至于总是需要更新。

稳步提升期

之后时间线到了 2018 年，此时 Docker 由于之前几年的蓬勃发展，容器化已成为主流；加上 Docker 自身也日益成熟，成为了功能完备的“容器平台”， **Docker 现在需要提供更加稳定可靠的版本。**

所以在 2018 年 6 月 Docker 公司宣布，将 Docker 的发布周期修改为半年更新。Docker 18.06 CE 版本将会是最后一个有 4 个月维护周期的版本，下一个版本 Docker 18.09 CE 会有 7 个月的维护期。之后在 2019 年 7 月份发布了 Docker 19.03，在这个版本正式发布之前，经历了 5 个 beta 版本、3 个 rc 版本，可以看到经过这次改变后，Docker 发布新版本是非常慎重的。

你可能会有几个困惑：

1. 为什么版本号是 19.03 而它却在 7 月发布?

这是因为 Docker 使用 YY.MM 的版本格式，主要是为了说明节奏，保证一般可用的“期望日期”，但并不会为了该日期而提前发布一个不可靠或者功能不完备的版本。

另外，新版本发布时，会尽可能将所有的依赖更新到最新的稳定版本，以保证在 Docker 的下一个版本发布之前，用户所用的 Docker 版本中不包含太落后的代码包。

2. 为什么 18.09 版本发布日期是 2018 年 11 月，但是直到 2019 年 8 月它仍然在更新小版本?

因为 Docker 的最新版本 19.03 是 2019 年 7 月发布的，Docker 通常会保持前一个版本与新版本的维护周期之间重合一个月左右，以便给用户足够多的时间来进行升级，以及可用于回归测试，让用户知道不会因为升级到下一个版本而造成什么严重问题。

小结

上面我们介绍了 Docker 版本的发展历程，如果你现在需要新安装 Docker 或是想要对现有 Docker 版本进行升级，我会强烈推荐你选择 Docker 19.03（截至当前最新的稳定版本）。

主要原因如下：

- 它是 2019 年 7 月份发布的，离社区结束维护周期还有很长时间。在此期间你遇到的各种问题均可反馈至社区，并会得到社区的帮助。

- 19.03 包含了更多特性，这些特性的介绍不是本篇的重点，我们在后续内容中会涉及并讨论。

内核兼容性

如果你是刚接触 Docker，并且要在一台自己可自由选择内核版本或系统版本的机器上安装 Docker，那你可以选择跳过本篇剩余的内容，直接看我这里给出的建议（剩余内容中会涉及一些 Docker 版本与内核相关的知识，现在跳过并不影响后续内容）：

尽可能地使用最新的稳定版内核和操作系统，以及安装最新且稳定的 Docker 版本。

这个建议中包含了两个信息：

- 选择最新且稳定的 Docker 版本，关于版本的问题前面已进行了说明，此处不再赘述；
- 使用最新的稳定版内核和操作系统。

> 注：这部分的讨论只涉及 Docker CE 与 Linux 系统，不讨论 Docker Desktop for Mac 或 Docker Desktop for Windows。

概览

这里以我们平时最常接触的 x86_64/amd64 的架构为例。

Docker 默认提供了 CentOS、Debian、Fedora、Ubuntu 等四种 Linux 发行版上的官方安装源，如果你是在这四种发行版上安装 Docker 我推荐你直接使用官方源，并利用系统上自带的包管理器进行安装，或者如同第二篇介绍的，直接使用 https://get.docker.com/ 提供的脚本进行安装。

官方文档中简要的写了一些安装 Docker 的必须项：

- 64 位操作系统
- 3.10 以上的内核
- iptables、Git、xz 或者挂载正确的 Cgroupfs

但写的其实很模糊，我们先看看**内核**相关的信息。

内核

为什么会选择 3.10 版本版本的内核呢？总的来说是因为 3.10 中包含了 Docker 所需的大多数特性，而且 3.10 是一个 LTS 版本，这里先不具体展开了，我们还是聚焦于内核兼容性的问题上。

Linux 3.10 版本是在 2013 年 6 月底发布的，我们来看看上述几个 Linux 发行版何时开始使用该内核版本的：

发行版	版本	发布日期	内核版本
CentOS	7.0-1406	2014 年 7 月	3.10.0-123
Debian	8.0	2015 年 4 月	Linux 3.16
Fedora	20	2013 年 12 月	3.11
Ubuntu	13.10	2013 年 10 月	Linux 3.11

可以看到，不同的发行版会选择不同版本的内核。我们来看看现在这些发行版所用的内核版本：

发行版	版本	发布日期	内核版本
CentOS	7.7-1908	2019 年 9 月	3.10.0-1062
CentOS	8.0-1905	2019 年 9 月	4.18.0-80
Debian	10.0	2019 年 7 月	Linux 4.19
Fedora	30	2019 年 4 月	5.0
Ubuntu	19.04	2019 年 4 月	Linux 5.0

可以看到内核版本更新最慢的是 CentOS，而一个很值得注意的点是，Linux 3.10 截至 Linux 3.10.108 就已经 EOL（End Of Life）了，**CentOS 使用的内核其实移植了很多高版本 Linux 中的特性。**

其实 Docker 和内核兼容性问题是一个很大的话题，参考上面提供的表格，Debian、Fedora、Ubuntu 等发行版的内核更新都相对及时，只要你不选择太过时或者太旧的版本，Docker 在发布之前，都会在这几个发行版的最新版上经过大量测试，若有问题也会尽早修复的。

我们来单独说下 CentOS。**如果你使用 CentOS，请务必选择 CentOS 7.6 及以上版本，或是将内核升级到 3.10.0-957 及更高版本**，为什么会有这个建议呢？

我们举个实际的例子：Docker 的存储驱动，在早先版本中其实对于 CentOS 系统上一直是默认使用 Devicemapper 的存储驱动，但最近的 Docker 版本中已经将 Devicemapper 标记为了废弃。所以我们推荐选择 Overlay2 存储驱动，关于这个存储驱动，在之后的存储篇我们会详细介绍。

Docker 的 Overlay2 存储驱动需要内核的 Multiple lower layers 支持，而 OverlayFS 是在 Linux 3.18 时合并进入内核的，但在 Linux 3.19 版本时才添加了 Multiple lower layers 的支持。

Linux 3.19 恰好是 3.x 系列的最后一个版本，两个月后 4.0 便发布了，通常情况下，人们会更推荐采用 4.0。所以官方文档上对于使用 Overlay2 存储驱动的建议是使用 Linux 4.0 及更高版本的内核。

而 CentOS 的内核中存在大量的反向移植，Mmultiple lower layers 的特性也被移植到了 CentOS 7.4 的内核之上。所以如果你只是考虑使用 Overlay2 作为存储驱动的话，则至少需要选择 CentOS 7.4 内核版本为 3.10.0-693 或者更新版本的内核。

> 备注：CentOS 7.x 与 8.x 区别较多，所以单独将 CentOS 的 7 和 8 版本列了出来。

总结

通过本篇，我们介绍了 Docker 版本及内核兼容性相关的内容。Docker 在现阶段使用 YY.MM 形式的版本号，一年发布两次版本，通常具有 7 个月左右的维护周期。如果在进行 Docker 版本的选择，我推荐你选择最新的稳定版。

关于内核方面，其实是一个很大的话题，这里只是简单的介绍了一下。而且考虑内核兼容性时不仅需要考虑 Docker 自身相关的兼容性，还需要考虑 Docker 依赖的相关底层组件与内核的兼容性问题。

比如，Docker 底层的容器运行时 RunC。2019 年 2 月，有一个关于 RunC 的提权漏洞 CVE-2019-5736，由于是一个紧急的安全漏洞，所以 RunC 的维护者进行了快速修复，而当时的修复方式是增加了两个系统调用 memfd_create(2) 和 fcntl(2)。而这两个的系统调用是在 Linux 3.17 时被加入内核中的。

造成的影响就是凡是内核不支持这两个系统调用的，均无法正常运行。但也有部分例外，比如 CentOS 的 3.10.0-514 内核反向移植了此特性。

不过上述提到的修正，在后来的版本中修改了实现方式，所以兼容性也得到了改善。关于这些底层组件，我在后续的架构篇中也都会有介绍。

所以如果你对内核方面感兴趣，可以多进行此方面的研究和积累，但如果对这些内容不甚了解或者没什么兴趣，那我推荐你**尽可能地选择最新的稳定版内核和系统**，可以免除很多因为内核兼容性方面造成的异常。

另外：Docker 也提供了一个脚本用于检查系统的配置，如果不确定自己当前系统的配置是否正确，可先运行此脚本进行检查。

相关内容：

- A new upstream project to break up Docker into independent components