SUSE

使用 docker-compose 管理多容器应用程序

解释

使用 docker-compose 可以定义和管理多容器应用程序。该工具通过易用的定义文件简化了此类应用程序堆栈的部署。

原因

本文介绍如何使用 docker-compose 创建多容器应用程序。

工作量

读完本文大约需要 20 分钟。

目标

您可以创建自己的基于容器的应用程序堆栈。

要求

用于构建应用程序的应用程序容器映像或关联的源文件。

出版日期：2025 年 12 月 11 日

目录

1 用于管理多容器应用程序的工具 3

2 创建多容器应用程序 4

3 管理多容器应用程序 9

4 法律声明 10

A GNU 自由文档许可证 10

1 用于管理多容器应用程序的工具

docker-compose 是负责创建多容器应用程序的工具。默认情况下，它使用 SLE Micro 中未随附的 Docker。要绕过 Docker 并改用 Podman，可以使用 `podman-docker` 脚本。这样，就无需将现有脚本更改为使用 Podman 而不是 Docker。以下各节提供了这些工具的详细说明。

1.1 关于 Podman

Podman 是 Pod Manager 工具的简称。它是一个无守护程序容器引擎，可让您使用容器和容器映像来运行和部署应用程序。Podman 提供了一个命令行界面来管理容器。

由于 Podman 没有守护程序，因此它提供了与 systemd 的集成。这样就可以通过 systemd 单元来控制容器。您可为现有容器创建这些单元，以及生成可启动容器的单元（如果系统中不存在这些单元）。Podman 可以在容器内部运行 systemd。

Podman 支持将容器整理为 Pod。Pod 共享相同的网络接口和资源。将一组容器组织成 Pod 的典型用例包括运行数据库的容器，以及带有访问数据库的客户端的容器。

1.1.1 安装 Podman

默认情况下，Podman 已包含在 SLE Micro 中。但是，如果缺少 Podman，您可以如下所述安装它：

1. 运行以下命令：

```
> sudo transactional-update pkg install podman*
```

2. 重启动系统以引导进入新快照。

1.2 关于 `podman-docker`

`podman-docker` 是一个 bash 脚本，用于将您运行的任何 **`docker`** 命令更改为使用相同传递参数的相应 **`podman`** 命令。因此，您无需进行任何修改即可使用所有 Docker 脚本。

1.2.1 安装 podman-docker

默认情况下，podman-docker 未安装在 SLE Micro 中。要安装它，请执行以下步骤：

1. 运行以下命令安装 podman-docker 软件包:

```
transactional-update pkg install podman-docker
```

2. 重引导系统以切换到最新快照。

1.3 关于 docker-compose

docker-compose 是用于管理多容器应用程序的工具。docker-compose 可让您在单个主机上创建多个隔离的环境，它还支持在环境之间使用变量。使用 docker-compose，可以做到只重新创建那些已更改的容器，而不必破坏整个多容器应用程序。

重要：docker 已由 podman 命令取代

在 SLE Micro 中，每当您运行 docker-compose 时，就会使用 podman-docker 脚本来调用 Podman，因为 SLE Micro 中没有 Docker。

1.3.1 安装 docker-compose

如果您的系统上没有 docker-compose，可以按照以下步骤安装它：

1. 运行以下命令：

```
> sudo transactional-update pkg install docker-compose
```

2. 安装完成后，重引导系统以引导进入新快照。

2 创建多容器应用程序

要创建多容器应用程序，请执行以下步骤：

1. 创建配置文件 `compose.yml`。有关细节，请参见第 2.1 节 “创建 `compose.yml`”。
2. 准备所需的目录结构。我们建议将 `compose.yml` 文件放在工作目录的最顶层。
3. 如果需要，请专门为容器化应用程序使用的服务编写您自己的容器文件。例如，要部署某个 Go 应用程序，请为该应用程序创建一个包含所需配置和依赖项的容器文件。
   我们建议在工作目录中为每个服务创建一个子目录，并将专用于服务的文件放入其中。
4. 部署多容器应用程序。有关细节，请参见第 2.2 节 “部署多容器应用程序”。

2.1 创建 `compose.yml`

要创建多容器应用程序，需要创建 `compose.yml` 文件，并最好将其放置在工作目录中。您可以创建单个文件，也可以通过片段和扩展名来使用更精细的方法。还可以合并多个 docker-compose 文件来定义整个应用程序模型。

文件 `compose.yml` 定义应用程序。可在其中包含以下部分。

服务

服务是应用程序的计算组件。有关定义的细节，请参见第 2.1.1 节 “服务定义”。

网络

可以使用 `network` 语句来定义自定义网络，并将特定的服务指派到自定义网络。有关细节，请参见第 2.1.2 节 “网络定义”。

卷

由容器引擎管理的目录，服务在其中存储和共享数据。

环境变量

您可能还需要使用传递给服务的环境变量列表。有关细节，请参见环境变量参考信息 (https://docs.docker.com/compose/environment-variables/)。

配置文件

必须在 `configs` 部分声明服务所需的所有配置文件。有关细节，请参见 `configs` 定义 (https://docs.docker.com/compose/compose-file/08-configs/)。

2.1.1 服务定义

定义服务时，需要指定所要使用的容器映像，或提供用于构建服务的源文件。

要从映像创建服务容器，请使用 `image` 语句：

```
services:
       db:
         image: database
```

Podman 将检查 `compose.yml` 文件中声明的映像名称是否已在本地容器存储中提供。如果未提供，Podman 将从一个已配置的注册表中提取映像。

要从源文件构建服务，请在同一目录中提供源文件并创建容器文件。然后在 `compose.yml` 文件中使用 `build` 语句：

```
      services:
        db:
          build: PATH_TO_SOURCE_FILES
```

如果某个特定服务必须在另一个服务之后启动，您可以使用 `depends_on` 语句：

```
      services:
       db:
         image: database
         depends_on:
          system:
            condition: SERVICE_CONDITION
```

`SERVICE_CONDITION` 可以是 `service_started`、`service_healthy` 或 `service_completed_successfully`。

有关 `services` 定义的详细信息，请参见 `services` 规范 (https://docs.docker.com/compose/compose-file/05-services/)。

2.1.2 网络定义

默认情况下，docker-compose 会创建默认网络，应用程序堆栈中的每个容器将包含在该网络中。不必在 `compose.yml` 文件中声明默认网络，因为 docker-compose 会自动创建默认网络。

您还可以定义自定义网络，并将特定服务指派到其中。例如，要创建 network1 和 network2 这两个网络，请添加以下代码段：

```
networks:
  network1:
    # Use a custom driver
    driver: custom-driver-1
  network2:
    # Use a custom driver and name the network
    driver: custom-driver-2
    name: custom_network
```

也可以使用现有网络。在这种情况下，请将网络标记为外部网络：

```
networks:
  network1:
    name: network1
    external: true
```

有关完整的 networks 规范，请参见 networks 规范 (https://docs.docker.com/compose/compose-file/06-networks/)↗。

2.1.3 compose.yml 示例

以下 compose.yml 示例定义一个使用 Prometheus 监控系统和 Grafana 分析系统的应用程序堆栈。

```
services:
  prometheus:
    image: prom/prometheus
    container_name: prometheus
    command:
      - '--config.file=/etc/prometheus/prometheus.yml'
    ports:
      - 9090:9090
    restart: unless-stopped
    volumes:
```

```
      - ./prometheus:/etc/prometheus
      - prom_data:/prometheus
  grafana:
    image: grafana/grafana
    container_name: grafana
    ports:
      - 3000:3000
    restart: unless-stopped
    environment:
      - GF_SECURITY_ADMIN_USER=admin
      - GF_SECURITY_ADMIN_PASSWORD=grafana
    volumes:
      - ./grafana:/etc/grafana/provisioning/datasources
volumes:
  prom_data:
```

此示例中的项目结构必须如下所示：

```
.
├── compose.yaml
├── grafana
│   └── datasource.yml
├── prometheus
│   └── prometheus.yml
└── README.md
```

2.2 部署多容器应用程序

创建正确的目录结构和 `compose.yml` 文件后，便可以部署多容器应用程序：

1. 校验您要运行的容器是否尚不存在：

   ```
   > podman ps --all
   ```

   如果需要，请去除特定的容器：

   ```
   > podman rm -fCONTAINER_ID
   ```

2. 通过从 `compose.yml` 所在的目录运行以下命令来启动多容器应用程序：

```
> docker compose up -d
```

docker-compose 将为多容器应用程序创建单独的网络。

3. 可以通过列出正在运行的容器，来校验容器是否正在运行，以及端口是否已映射：

```
> podman ps
```

3 管理多容器应用程序

创建多容器应用程序后，可以使用 **`docker-compose`** 命令执行管理操作。命令语法如下：

```
> docker compose[OPTIONS]COMMAND
```

从您要管理的多容器应用程序的 `compose.yml` 文件所在的同一目录运行该命令。或者，使用 `-f, --file` 选项提供 `compose.yml` 文件的路径。例如，要退出并去除多容器应用程序，请运行以下命令：

```
> docker compose -f ./test/compose.yml down
```

其他有用命令：

images

列出多容器应用程序中的容器使用的所有映像。

```
> docker compose images
```

pause

暂停所有容器。

```
> docker compose pause[SERVICE]
```

ps

列出多容器应用程序中的容器。

```
> docker compose ps
```

rm

去除已停止的容器。

```
> docker compose rm
```

start/stop

启动或停止容器。

```
> docker compose stop[SERVICE]
```

要显示完整的选项和命令列表，请运行：

```
> docker-compose --help
```

4 法律声明

A GNU 自由文档许可证

0. 导言

此许可证的目的是赋予手册、教科书或其他功能性的和有用的文档以“自由”：即保证每个人都有复制和再分发此类文档的有效自由，无论是否进行修改，也无论将其用于商业或非商业用途。其次，此许可证为作者和出版者保留了因工作获得声誉但不视为对他人所做修改负责的方式。

本许可证是一种“非盈利版权”，这意味着从该文档衍生的作品也必须是以同一方式自由的。它补充了 GNU 通用公共许可证（为自由软件设计的非盈利版权许可证）。

我们设计此许可证旨在将其用于免费软件的手册，因为免费软件需要自由文档：免费程序所附手册应具有与软件本身同样的自由。但是此许可证不限于软件手册；它可以用于任何文本作品，无论主题如何或它是否作为印刷书籍出版。建议本许可证主要用于目的是指导或参考的作品。

1. 适用性和定义

此许可证适用的对象：由版权所有者在其中明确声明可按照此许可证条款以任何媒体分发的任何手册和其他作品。此类声明授予在此处所述的条款和条件下使用该作品的全球无限期无版权许可证。下述“文档”指任何此类手册或作品。任何公众成员都是一个被许可人，以下称为“您”。如果您以需要版权法许可的任何方式复制、修改或分发该作品，则表示您接受该许可证。

该文档的“修改版本”表示包含该文档或其一部分（或者逐字复制或者有修改和/或翻译为另一语言）的任何作品。

“次要章节”是该文档的命名附录或扉页章节，专门讲述该文档的出版者或作者与该文档整个主题（或相关问题）的关系，不包含与整个主题相关的内容。（因此，如果该文档是数学课本的一部分，则辅助部分可能不说明任何数学问题。）这种关系可以是与主题或相关问题的历史联系，或与它们相关的法律、商业、哲学、伦理或政治地位。

在该文档基于此许可证项发布的声明中，“固定章节”是将其标题指定为固定章节标题的一些辅助章节。如果一个章节不适用上述辅助章节的定义，则不允许将其指定为固定章节。该文档可能不包含固定章节。如果该文档不标识任何固定章节，则表示没有固定章节。

在该文档基于此许可证项发布的声明中，“封页文本”是作为封面文本或封底文本列出的简短文本段落。封面文本最多 5 个单词，封底文本最多 25 个单词。

文档的“透明”副本是一个机器可读的副本，使用公众可以得到其规范的格式表达，这样的副本适合于使用通用文本编辑器、（对于像素构成的图像）通用绘图程序、（对于绘制的图形）广泛使用的绘画编辑器直接修改文档，也适用于输入到文本格式处理程序或自动翻译成各种适于适用于输入到文本格式处理程序的格式。一个用其他透明文件格式表示的副本，如果该格式的标记（或缺少标记）已经构成了对读者的后续修改的障碍，那么就是不透明的。表示实质性数量的文本的图像格式都是不透明的。不“透明”的副本称为“不透明”。

适于作为透明副本的格式的示例有：没有标记的纯 ASCII 文本、Texinfo 输入格式、LaTeX 输入格式、使用公共可用 DTD 的 SGML 或 XML，符合标准的简单 HTML、可以人为修改的 PostScript 或 PDF。透明图像格式的示例有 PNG、XCF 和 JPG。不透明的格式包括：仅可以被私有版权的字处理软件使用的私有版权格式、所用的 DTD 和/或处理工具不是广泛可用的的 SGML 或 XML、机器生成的 HTML、一些字处理器生成的只用于输出目的的 PostScript 或 PDF。

对于印刷书籍，“扉页”就是扉页本身以及随后的一些用于补充的页，显然本许可资料需要出现在扉页上。对于那些没有扉页的作品形式，“扉页”代表接近作品最突出标题的、在文本正文之前的文本。

“命名为 XYZ”的章节表示文档的一个特定的子单元，其标题就是 XYZ 或在括号中包含 XYZ 且后跟 XYZ 的其他语言翻译文本。（这里 XYZ 代表下面提及的特定章节名称，比如“致谢”、“题献”、“签名”或“历史”。）要在修改文档时对这类章节“保留标题”就是依据此定义保持这样一个“命名为 XYZ”的章节。

文档可能在文档遵照此许可证的声明后面包含免责声明。这些免责声明应作为参考信息包含在此许可证中，但是只能将其视作免责声明：这些免责声明暗指的任何其他含义均无效，且对此许可证的含义不产生任何影响。

2. 逐字复制

您可以用任何媒体复制并分发文档，无论是出于商业还是非商业目的，只要保证此许可证、版权声明和声称此许可证应用于文档的声明都完整地、无任何附加条件地存在于所有副本中。不能使用任何技术手段阻碍或控制您制作或发布的副本的阅读或再次复制。不过您可以在副本交易中得到报酬。如果发布足够多的副本，则您必须遵循下面第三节中的条件。

您也可以在如上的条件下出租副本和向公众放映副本。

3. 大量复制

如果您出版的文档印刷版副本（或是有印制封页的其他媒体副本）多于 100 份，而文档的许可证声明中要求有封页文本，则您必须将它清晰地置于封页之上，封面文本在封面上，封底文本在封底上。封面和封底上还必须标明您是这些副本的出版者。封面必须同等显著地完整展现标题的所有文字。您可以在封页上加入其他资料。改动仅限于封页的复制，只要保持文档的标题不变并满足这些条件，可以在其他方面被视为逐字复制。

如果需要加上的文本对于封面或封底过多，无法明显地表示，您应该在封页上列出前面的（在合理的前提下尽量多），把其他的放在邻近的页面上。

如果您出版或分发了超过 100 份文档的不透明副本，则必须在每个不透明副本中包含一份计算机可读的透明副本，或是在每个不透明副本中给出一个计算机网络地址，通过这个地址，网络公共用户可以使用标准网络协议下载文档的无任何附加资料的完整透明副本。如果您选择后者，则必须在开始大量分发非透明副本的时候采用相当谨慎的步骤，保证透明副本在其所给出的位置在（直接或通过代理和零售商）分发最后一次该版本的非透明副本的时间之后一年之内始终是有效的。

在重新大量发布副本之前，请您（但不是必须）与文档的作者联系，以便他们可以有机会向您提供文档的更新版本。

4. 修改

在上述第 2、3 节的条件下，您可以复制和分发文档的修改版本，前提是严格按照此许可证发布修改后的文档，将修改版本用作文档，从而允许任何拥有此修改版副本的人执行分发或修改。另外，在修改版中，您需要做到如下几点：

A. 用于与文档以及以前各个版本（如果有，应该列在文档的“历史”章节中）显著不同的扉页（和封页，如果有）。如果那个版本的原始发行者允许的话，您可以使用和以前版本相同的标题。

B. 与作者一样，在扉页上列出承担修改版本中的修改的作者责任的一个或多个人或实体和至少五个文档的原作者（如果原作者不足五个就全部列出），除非他们免除了您的这个责任。

C. 与原来的发行者一样，在扉页上列出修改版的发行者的姓名。

D. 保持该文档的全部版权声明不变。

E. 在与其他版权声明邻近的位置加入恰当的针对您的修改的版权声明。

F. 在紧接着版权声明的位置加入许可声明，按照下面附录中给出的形式，以本许可证给公众授于是用修订版本的权利。

G. 保持原文档的许可声明中的全部不可变章节、封面文字和封底文字的声明不变。

H. 包含一份未作任何修改的本协议的副本。

I. 保持命名为“历史”的章节不变，保持它的标题不变，并在其中加入一项，至少声明扉页上的已修改版本的标题、年份、新作者和出版者。如果文档中没有命名为“历史”章节，则请新建它，并加入一项以声明原文档扉页上所列的标题、年份、作者与出版者，再在其后加入如上所说的描述修改版本的项。

J. 如果文档中有用于公共用户访问的文档透明副本的网址，则保持网址不变，并同样提供它所基于的以前文档版本的网址。这些网址可以放在“历史”章节。您可以不给出那些在原文档发行之前已经发行至少四年的版本给出的网址，或者该版本的发行者授权不列出网址。

K. 对于任何命名为“致谢”或“题献”的章节，保持其标题不变，并保持其全部内容以及对每位贡献者致谢和/或题献的语气不变。

L. 保持文档的所有固定章节不变，不改变它们的标题和内容。章节的编号或相当的内容不被认为是章节标题的一部分。

M. 删除命名为“签名”的章节。这样的章节不可以被包含在修改后的版本中。

N. 不要把任何现有章节重命名为“签名”或与任何不可变章节相冲突的标题。

O. 保持任何免责声明不变。

如果修改版本加入了新的符合次要章节定义的引言或附录章节，并且不含有从原文档中复制的内容，您可以选择将其标记为固定。如果需要这样做，则将它们的标题加入修改版本许可声明的不可变章节列表之中。这些标题必须和其他章节的标题相区分。

您可以加入一个命名为“签名”的章节，只要它只包含对您的修改版本由不同的各方给出的签名，例如书评或是声明文本已经被一个组织认定为一个标准的权威定义。

您可以加入一个最多 5 个字的段落作为封面文本和一个最多 25 个字的段落作为封底文本，将它们加入修改版本的封页文本列表末端。一个实体只可以添加（或编排）一段封面和一段封底文本。如果原文档已经为该封页（封面或封底）包含了封页文本，由您或您所代表的实体先前加入或排列的文本，不能再新加入一个，但您可以在原来的发行者的明确许可下替换掉原来的那个。

作者和发行者不能通过本许可证授权公众使用他们的名字推荐或暗示认可任何一个修改版本。

5. 组合文档

遵照第 4 节所说的修改版本的规定，您以将文档和其他文档合并并以本许可证发布，只要您在合并结果中包含原文档的所有不可变章节，对它们不加以任何改动，并在合并结果的许可声明中将它们全部列为不可变章节，而且维持原作者的免责声明不变。

合并作品仅需要包含一份此许可证，多个相同的固定章节可以由一个副本取代。如果有多个名称相同但内容不同的固定章节，通过在章节名称后面的括号中加上原作者或出版者的姓名（如果已知）来加以区别，或者使用唯一编号加以区别。并对合并作品许可声明中的固定章节列表中的章节标题做相同的调整。

在合并过程中，必须合并不同原始文档中任何命名为“历史”的章节，从而形成新的命名为“历史”的章节；类似地，还要合并命名为“致谢”和“题献”的章节。必须删除所有命名为“签名”的章节。

6. 文档的合集

您可以制作一个文档和其他文档的合集，在本许可证下发布，并在合集中将不同文档中的多个本许可证的副本以一个单独的副本来代替，只要您在文档的其他方面遵循本许可证的逐字复制的条款即可。

您可以从一个这样的合集中提取一个单独的文档，并将它在本许可证下单独发布，只要您想这个提取出的文档中加入一份本许可证的副本，并在文档的其他方面遵循本许可证的逐字复制的原则。

7. 独立作品的聚合体

将文档或其派生品以及其他独立和无关文档或作品编撰在一个储存卷中或分发媒体上，这称为文档的“聚合体”，前提是编撰成品的著作权对其使用者的法律权限的限制未超出各个独立作品的许可范围。当基于此许可证发布的文档包含在一个聚合体中时，此许可证不适用于聚合体中的本非该文档派生作品的其他作品。

如果第 3 节中的封页文本要求适用于这些文档的副本，则若文档在聚合体中所占的比重小于全作品的一半，文档的封页文本可以放置在聚合体内包含文档部分的封页上，或是电子文档中的等效部分。否则，它必须位于整个聚合体的印刷的封页上。

8. 翻译

翻译被视为一种修改，因此您可以根据第 4 节的条款分发文档的翻译。将固定章节替换为翻译内容需要经得其版权所有者的特别许可，但除了这些固定章节的原始版本之外，您还可以包含一部分或所有固定章节的翻译。您可以包含一个此许可证以及所有许可证声明和免责声明的翻译版本，前提是同时包含它们的原始英文版本。当翻译版本和英文版发生冲突的时候，原始版本有效。

如果在文档中有命名为“致谢”、“题献”或“历史”的章节，保持标题（第 1 节）的要求（第 4 节）恰恰需要更换实际的标题。

9. 终止

除非此许可证中有明确规定，否则您不能对该文档进行复制、修改、分授许可或分发。在此许可证规定外对该文档所进行的任何复制、修改、分授许可或分发都是无效的，并且将自动终止您在此许可证下所拥有的权利。但是，对于在此许可证的规定下从您这里获得副本或权利的各方，只要其完全遵守此许可证的规定，其许可证将不会被终止。

10. 本许可的未来修订版本

自由软件基金会有时会发布 GNU 自由文档许可证的新的修订版版本。这些新版本的主旨和精神与当前版本是一致的，但在解决新问题的具体细节方面可能有所不同。请参见 https://www.gnu.org/copyleft/。

许可证的每个版本都有一个不同的版本号。如果文档指定了适用于它的此许可证“或任何后续版本”的特定带编号版本，则您可以选择遵从指定版本或自由软件基金会发布的任何随后版本（非草稿）的条款和条件。如果文档没有指定此许可证的版本号，您可以选择自由软件基金会发布的任何许可证版本（非草稿）。

附录：如何针对您的文档使用此许可证

```
Copyright (c) YEAR YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled “GNU
Free Documentation License”.
```

如果您有固定章节、封面文本和封底文本，请将“with...Texts”部分替换为：

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

如果有不可变章节而没有封页文本，或这三种内容（不可变章节、封面文本、封底文本）的任何其他组合，请合并这两个备选项以适应您的情况。

如果您的文档包含不一般的程序代码示例，建议同时选择自由软件许可证（如 GNU 通用公共许可证）发布这些示例，以允许它们可以用于自由软件。