链滴

推荐 5 款超好用的开源 Docker 工具!

作者：shealtiel
原文链接：https://ld246.com/article/1591089106694
来源网站：链滴

导读

Docker 社区已经创建了许多开源工具，它们能帮我们处理各种用例。作者在本文中推荐了 5 款认为有用的 Docker 工具，分别是 Watchtower（自动更新 Docker 容器）、docker-gc（容器和镜像的圾回收）、docker-slim（容器瘦身）、 rocker：突破 Dockerfile 的限制，以及 ctop（容器的类顶接口）。

Docker 社区已经创建了许多开源工具，它们所能帮你处理的用例甚至会超出你的想象。

你可以在网上找到很多酷炫的 Docker 工具，其中大部分是开源的，都可以在 Github 上找到。在过的两年里，我非常热衷于 Docker，在大多数开发项目中都使用了它。当你开始使用 Docker 后，你发现它适用的场景比你最初预想的还更多。你会希望 Docker 尽可能为你多做一点事，而它不会让你望的！

Docker 社区非常活跃，每天都会出现许多有用的工具，时时关注社区中发生的所有创新是很困难的为了帮助你，我收集了一些我在日常工作中使用的又有趣又实用的 Docker 工具，这些工具提升了我工作效率，减少了原本需要手工完成的工作。

watchtower（自动更新 Docker 容器）

Watchtower 监视运行容器并监视这些容器最初启动时的镜像有没有变动。当 Watchtower 检测到个镜像已经有变动时，它会使用新镜像自动重新启动相应的容器。我想在我的本地开发环境中尝试最的构建镜像，所以使用了它。

Watchtower 本身被打包为 Docker 镜像，因此可以像运行任何其他容器一样运行它。要运行 Watchower，你需要执行以下命令：

```
docker run -d --name watchtower --rm -v /var/run/docker.sock:/var/run/docker.sock  v2tec/atchtower --interval 3
```

在上面的命令中，我们使用一个挂载文件 /var/run/docker.sock 启动了 Watchtower 容器。这么做

有必要的，为的是使 Watchtower 可以与 Docker 守护 API 进行交互。我们将 30 秒传递给间隔选项 nterval。此选项定义了 Watchtower 的轮询间隔。Watchtower 支持更多的选项，你可以根据文档的描述来使用它们。

我们现在启动一个 Watchtower 可以监视的容器。

```
docker run -p 4000:80 --name friendlyhello shekhargulati/friendlyhello:latest
```

现在，Watchtower 将开始温和地监控这个 friendlyhello 容器。当我将新镜像推送到 Docker Hub，Watchtower 在接下来的运行中将检测到一个新的可用的镜像。它将优雅地停止那个容器并使用这新镜像启动容器。它将传递我们之前传递给这条 run 命令的选项。换句话说，该容器将仍然使用 4000 80 发布端口来启动。

默认情况下，Watchtower 将轮询 Docker Hub 注册表以查找更新的镜像。通过传递环境变量 REPO USER 和 REPO_PASS 中的注册表凭据，可以将 Watchtower 配置为轮询私有注册表。

要了解更多 Watchtower 的相关信息，建议你阅读 Watchtower 文档

https://github.com/v2tec/watchtower/blob/master/README.md

GitHub 地址：https://github.com/v2tec/watchtower

docker-gc（容器和镜像的垃圾回收）

Docker-gc 工具通过删除不需要的容器和镜像来帮你清理 Docker 主机。它会删除存在超过一个小时所有容器。此外，它还删除不属于任何留置容器的镜像。

你可以将 docker-gc 作为脚本和容器来使用。我们将以容器的形式运行 docker-gc。若要使用 docke -gc 来查找所有可以删除的容器和镜像，命令如下：

```
docker run --rm -v /var/run/docker.sock:/var/run/docker.sock -e DRY RUN=1 spotify/docker-c
```

上述命令中，我们加载了 docker.sock 文件，以便 docker-gc 能够与 Docker API 交互。我们传递一个环境变量 DRY_RUN=1 来查找将被删除的容器和镜像。如果不提供该参数，docker-gc 会删除有容器和镜像。最好事先确认 docker-gc 要删除的内容。上述命令的输出如下所示：

```
\[2017-04-28T06:27:24\] \[INFO\] : The following container would have been removed 0c1b3 0972bb792bee508 60c35a4 bc08ba32b527d53eab173d12a15c28deb931/vibrant_ yonath\[20 7-04-28T06:27:24\] \[INFO\] : The following container would have been removed 2a72d41e4b 5e2782f7844e188643e395650a9ecca660e7a0dc2b7989e5acc28 /friendlyhello_ web\[2017-04 28T06:27:24\] \[INFO\] : The following image would have been removed sha256:00f017a8c2a e1 fe2f fd05c281 f27d069d2a99323a8cd514dd35f228ba26d2ff\[busybox: latest\]\[2017-04-28 06:27:24\] \[ INFO\] : The following image would have been removed sha256 :4a323b466a5ac ce6524 8dd970b538922c54e535700cafe9448b52a3094483ea\[hello-world:latest\]\[2017-04-2 T06:27:24\] \[INFO\] : The following image would have been removed sha256:4a323b4 66a5a 4ce65248dd970b538922c54e535700cafe9448b52a3094483ea\[python:2.7-slim\]
```

如果你认同 docker-gc 清理方案， 可以不使用 DRY_RUN 再次运行 docker-gc 执行清空操作。

```
docker run --rm -v /var/run/docker.sock:/var/run/docker.sock spotify/docker-gc
```

docker-gc 还支持一些其他的选项。建议你阅读 docker-gc 文档以了解更多相关信息：

https://github.com/spotify/docker-gc/blob/master/README.md

GitHub 地址：https://github.com/spotify/docker-gc

docker-slim（面向容器的神奇减肥药）

如果你担心你的 Docker 镜像的大小，docker-slim 可以帮你排忧解难。

docker-slim 工具使用静态和动态分析方法来为你臃肿的镜像瘦身。要使用 docker-slim，可以从 Github 下载 Linux 或者 Mac 的二进制安装包。成功下载之后，将它加入到你的系统变量 PATH 中。

为举例需要，我参考 Docker 官方文档创建了一个名为 friendlyhello 的 Docker 镜像，该镜像大小为 194MB（如下所示）：

```
friendlyhello                  latest              390075b080ad        53
minutes ago       194 MB
```

你可以看到，对于一个简单的应用程序，我们必须下载 194 MB 的数据。让我们用 docker-slim 来看它能减掉多少脂肪。

docker-slim build --http-probe friendlyhello

docker-slim 工具对胖镜像进行一系列的检查、测量，最终创建一个瘦版本的镜像。让我们看看这个过肥的大小吧。

```
REPOSITORY                     TAG                 IMAGE ID
CREATED              SIZE
friendlyhello.slim             latest              e547885c0b02        7
seconds ago        24.9 MB
```

正如你所看到的，镜像大小被减少到 24.9 MB。你可以启动这个容器，它将以同样的方式运行。docker-slim 工具支持 Java、Python、Ruby 和 Node.js 应用。

你自己试试，看看能减下来多少。在我的个人项目中，我发现它在大多数情况下都适用。你可以从其档中了解更多关于 docker-slim 的信息：

https://github.com/docker-slim/docker-slim/blob/master/README.md

GitHub 地址：https://github.com/docker-slim/docker-slim

rocker（突破 Dockerfile 的限制）

大多数使用 Docker 的开发人员都使用 Dockerfile 来构建镜像。Dockerfile 是一种声明式的方法，于定义用户可以在命令行上调用的所有命令，从而组装镜像。

Rocker（https://github.com/grammarly/rocker）为 Dockerfile 指令集增加了新的指令。Grammarly 为了解决他们遇到的 Dockerfile 格式的问题，创建了 Rocker。Grammarly 团队写了一篇深入的客，解释他们创建它的原因。我建议你读一读，以更好地了解 Rocker。他们在博文中强调了两个问：

Docker 镜像的大小。

缓慢的构建速度。

该博客还提到了 Rocker 加入的一些新指令。参考 Rocker 文档，了解 Rocker 支持的所有指令：

https://github.com/grammarly/rocker/blob/master/README.md

MOUNT 用于在构建之间共享卷，以便能够被依赖项管理工具重用。

在 Dockerfile 中原本已有 FROM 指令。而 Rocker 使我们可以添加一条以上的 FROM 指令。这意着你可以通过单个 Rockerfile 创建多个镜像。第一批指令用于构建产品所有的依赖；第二批指令用于建产品；这能够极大地降低镜像大小。

TAG 用于在构建的不同阶段标识镜像，这意味着你不必手动为每个镜像打标签。

PUSH 用于将镜像推送到镜像仓库。

ATTACH 使你能够交互式地运行中间步骤。这一点对于调试非常有用。

要使用 Rocker，首先必须在你的机器上安装。对 Mac 用户来说，就是简单地运行几条 brew 命令：

```
brew tap grammarly/tap$ brew install grammarly/tap/rocker
```

一旦完成安装，你就可以通过传递 Rockerfile 使用 Rocker 来构建镜像了：

```
FROM python:2.7-slimWORKDIR /appADD . /appRUN pip install -r requirements. txtEXPOSE 8
ENV NAME WorldCMD \["python","app.Py"\]TAG shekhargulati/ friendlyhello:{{ .VERSION }}P
SH shekhargulati/friendlyhello:{{ .VERSION }}
```

若要构建一个镜像并将其推送到 Docker Hub，你可以运行以下命令：

```
rocker d build --push -var VERSION-1.0
```

GitHub 地址：https://github.com/grammarly/rocker

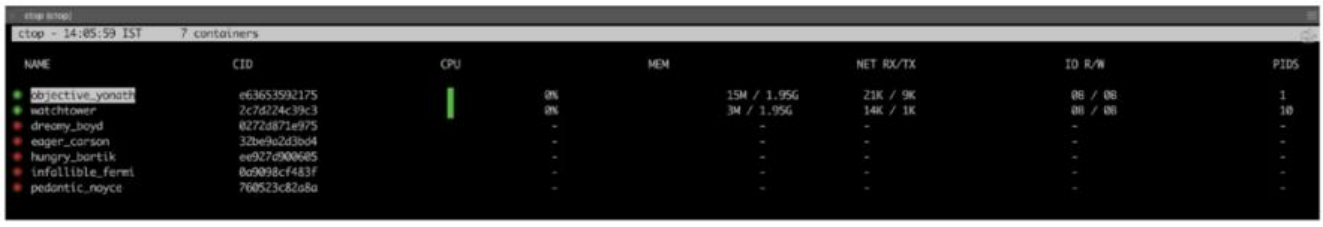
ctop（容器的类顶层接口）

ctop 是我最近开始使用的一个工具，它能够提供多个容器的实时指标视图。如果你是一个 Mac 用户可以使用 brew 安装，如下所示：

```
brew install ctop
```

一旦完成安装，就可以开始使用 ctop 了。现在，你只需要配置 DOCKER_HOST 环境变量。

你可以运行 ctop 命令，查看所有容器的状态。



若只想查看正在运行的容器，可以使用 ctop -a 命令。

ctop 是一个简单的工具，对于了解在你的主机上运行的容器很有帮助。你可以在 ctop 文档中了解更相关信息：

https://github.com/bcicen/ctop/blob/master/README.md

GitHub 地址：https://github.com/bcicen/ctop

以上是我发现的很有用的 5 款 Docker 工具。你在日常工作中使用 Docker 工具吗？希望这些工具能为你带来帮助，也欢迎在评论中推荐你觉得有用的工具。

原文：dzone.com/articles/5-docker-utilities-you-should-know