Docker今年应该算是很热门的技术了，之前一直没怎么真正地去了解、接触。通过翻译这篇简单的教程，我同时也对Dcoker有了入门的了解。Docker和我目前使用的Vagrant有些类似之处，都是通过类似镜像的文件创建孤立的系统环境。只是Docker容器中已经安装了所需的包，而我目前所了解的Vagrant box还没有这个功能。当然，应该是我使用程度不深的原因。不管怎么说，Docker确实是跨平台开发的利器。译文如果有什么不准确的地方，请大家指正。

数据科学开发环境配置起来让人头疼，会碰到包版本不一致、错误信息不熟悉和编译时间漫长等问题。这很容易让人垂头丧气，也使得迈入数据科学的这第一步十分艰难。而且这也是一个完全不常见的准入门槛。

还好，过去几年中出现了能够通过搭建孤立的环境来解决这个问题的技术。本文中我们就要介绍的这种技术名叫Docker。Docker能让开发者简单、快速地搭建数据科学开发环境，并支持使用例如Jupyter notebooks等工具进行数据探索。

要使用Docker，我们要先下载含有相关包（package）和数据科学工具的镜像文件。之后，我们可以通过该镜像，在数秒之内就启动一个数据科学开发环境，免去了手动安装包的麻烦。这个环境，也被成为Docker容器（container）。容器解除了配置的问题——当你启动一个Docker容器后，它就已经处于了良好的状态，所有的包都是可以正常运转的。

除了降低进入数据科学的门槛之外，Docker还可以让我们快速搭建拥有不同Python版本和安装了不同包的孤立环境，不像虚拟环境（virtualenv）那样还要重新安装包。

在本文中，我们将介绍Docker的基础知识，如何安装Docker以及如何利用Docker容器快速地在本地机器上搭建数据科学环境。

虚拟机

能够创建虚拟机的软件已经问世数十年，可以让你在本地电脑上模拟其他的系统环境。举个例子，即使你的电脑运行的是Windows操作系统，你仍可以通过虚拟机运行Linux系统。这可以让你在不重装系统的前提下，使用Linux——也就是说，Linux系统是虚拟化运行的，所以你可以从Windows系统访问虚拟机。基本上，你可以在点击该软件的程序图标之后，看到弹出的窗口中乃是一个Linux系统桌面。而虚拟机需要镜像来启动，也就是你必须先拥有一个目标系统的镜像，才能启动相应的虚拟机。如果你想使用Linux，你使用的镜像就得包含创建Linux环境所必须的全部文件。

容器

尽管虚拟机有诸多好处，例如能够使在Windows平台进行Linux开发成为现实，但是也有着自身的缺陷。首先，虚拟机的启动时间很长，要消耗大量的系统资源。另外，在利用镜像创建完虚拟机中，很难在安装完所需要的包后，再将这个镜像保存，创建为新的镜像。而Docker提供的Linux容器，则通过让多个孤立环境在同一台机器上运行，解决了这个问题。你可以把容器看作是一种更快、更简单地使用虚拟机的方法。

但是，容器的使用却有一点麻烦，而且管理和发布容器镜像也不容易。作为开发人员，我们希望能够快速下载并启动一个拥有指定包和工具配置的数据科学环境。例如，你肯定会希望能快速启动一个安装了Jupyter notebook、spark和pandas的容器。

Docker

Docker容器的里层包裹的是Linux容器（a layer over Linux containers），可以支持更简单地对容器进行管理和发布。使用Docker，可以很容易地下载具备相应包的镜像，并且快速启动。另外，Docker是跨平台的，支持包括Mac、Windows和Linux等系统。

作为创建孤立Python环境的另一种方式，虚拟环境（virtual environment）也有这些优势。但是Docker相较于虚拟环境的主要优势有：

- 能够快速启动。如果你想马上就开始进行数据分析，使用Docker就免去了你等待各种包进行安装的时间。
- 配置测试无误。很多时候，要正常安装Python包会，需要以安装某些系统包为前提，并只有在进行相应设置后才能正常使用。如果设置不当，会引起一些很奇怪的错误。但是使用Docker后，这些包就已经配置好了，可以立即使用。
- 跨平台一致性。Python中的包是可以跨平台使用的，但是在Windows和Linux平台下有些不同，而且还有部分依赖包无法在Windows中安装。但是由于Docker容器运行的都是Linux环境，所以它们是高度一致的。
- 能够设置checkpoint并且进行恢复。你可以往Docker镜像中安装包，然后将那个checkpoint下的环境创建成一个新的镜像。这让你能够快速撤销或者回滚配置。

运行一个Docker镜像，就相当于创建了一个Docker容器。在本文中，我们在容器中运行一个Jupyter notebook，然后通过浏览器界面来处理数据。

安装Docker

第一步就是安装Docker。Docker官方为Windows和Mac用户提供了一个简便安装过程的图形界面安装器。下面是每个操作系统的安装指南。

- Mac OS
- Linux
- Windows

在安装时，你需要使用shell命令提示符（shell prompt）。shell命令提示符（shell prompt）也被称为终端或命令行，是在你的机器上通过文本界面而非图形界面运行命令的一种方式。例如，你可以在Windows系统中双击记事本就可以打开一个文本编辑器，也可以在Linux终端中输入nano达到这个目的。Docker提供了一个预先配置好的shell，可以用来运行Docker命令。请按照下面的方法操作：

- Mac OS —— 从Launchpad中打开Docker Quickstart Terminal程序。详情见本篇文章。
- Linux —— 打开任意bash终端，就可以使用docker命令。
- Windows —— 双击桌面上的Docker QuickstartTerminal程序的图标。详情见本篇文章。

下文在提到需要运行Docker命令或输入某个命令时，你都需要使用这个shell命令提示符。

下载镜像

下一步是下载你需要的镜像。下面是我们网站(dataquestio)目前提供的数据科学开发专用镜像：

`dataquestio/python3-starter` —— 这个镜像已经安装好了Python 3, Jupyter notebook和许多其他流行的数据科学库，包括numpy，pandas，scipy，scikit-learn和nltk。

`dataquestio/python2-starter` —— 这个镜像已经安装好了Python 2, Jupyter notebook和许多其他流行的数据科学库，包括numpy，pandas，scipy，scikit-learn和nltk。

你可以通过输入`docker pull IMAGE_NAME`命令，下载相应的镜像。如果你想下载`dataquestio/python3-starter`这个镜像，那么你需要在终端输入`docker pull dataquestio/python3-starter`命令。输入这段命令后，程序会自动从Docker Hub下载镜像。Docker Hub与Github类似，不过却是Docker镜像的一个中枢。它会将相应的镜像文件下载至你的本地机器，这样你才能利用该镜像创建容器。

新建一个文件夹

在本地创建一个文件夹，用于存放notebooks。这个文件夹中将储存你所有的工作文件，并会持续存在于你的机器中，即使是你销毁了docker容器。在这里，我们将创建下面这个文件夹，`/home/vik/notebooks`。

运行镜像

镜像下载完成后，你可以通过`docker run`运行该镜像。我们还需要传入一些选项，确保镜像配置正确。

-p 选项用于设置虚拟机的端口，让我们可以在本地访问Jupyter notebook服务器。

-d 选项用于以detached模式运行容器，也就是作为背景进程运行。

-v 选项让我们指定在本地机器中使用哪个文件夹存储notebook。

完整的运行命令是类似这样的：`docker run -d -p 8888:8888 -v /home/vik/notebooks:/home/ds/notebooks dataquestio/python3-starter`。

你应该将`/home/vik/notebooks`更改为你用于存储文件的地址。另外，应该把`dataquestio/python3-starter`更改为自己喜欢的docker镜像。

执行`docker run`命令将会创建一个Docker容器。这是与你的本地机器相隔绝的，也可以把它看作是一台单独的电脑。在容器内部，会运行一个Jupyter notebook服务器，并可以让我们使用许多数据科学工具包。

另外，docker run命令也会在终端打印出容器的编码（container id），在通过其他docker容器对该容器进行修改时，就必须要使用这个编码。在下文中我们称该编码为容器编码。

查看notebook服务器

如果你的系统是Linux，那么下一步非常简单——只需要在浏览器中打开localhost:8888，之后应该就能看到运行中的notebook。如果你使用的是Windows或OSX，之前也按照Docker官方安装指南进行了操作，并且安装过程中使用了docker-machine，那么本地机器的名称是default，运行docker-machine ip default命令就可以得知docker容器的ip。如果使用了其他的名字，例如dev，那在命令中将default替换为dev即可。接下来，在浏览器中访问CONTAINER_IP:8888就可以看到notebook（将CONTAINER_IP替换为你的容器编码）。

创建一个notebook

到了这一步，你可以创建一个新的Jupyter notebook测试下这个孤立的开发环境。试试输入下面这个scikit-learn的例子：

fromsklearnimportdatasetsfromsklearn.cross_validationimportcross_val_predictfromsklearnimportlinear_modelimportmatplotlib.pyplotasplt[y.min(),y.max()],'k--',lw=4)ax.set_xlabel('Measured')ax.set_ylabel('Predicted')plt.show()

添加数据文件

如果你想往开发环境中添加数据文件，你有三个选择。第一个选择，就是将文件放在你之前创建用来存放notebook的文件夹中。你放那里的任何文件将可以自动通过Jupyter notebook中访问。

第二种选择就是使用docker cp命令。docker cp可以从本地机器复制文件至容器中，反之亦然。假设你想拷贝/hom/vik/data.csv文件至一个id为4greg24134的容器中，你可以输入下面的命令：docker cp /home/vik/data.csv 4greg24134:/home/ds/notebooks。这会将data.csv文件拷贝到容器中用于存放notebook的文件夹中。当然，你可以选择将文件放到容器中的任何地方，但是把它们放在存放notebook的文件夹忠厚，你就可以轻松地通过Jupyter notebook访问这些文件了。

第三个选择就是使用Jupyter notebook首页右上方的upload按钮。这可以让你选择一个文件，并上传到容器中用于存放notebook的文件夹中。

不管你使用哪种方法，要想在Jupyter notebook中加载文件，需要按照类似下面的方式进行：

importpandasdata=pandas.read_csv("data.csv")

复制容器中的数据文件

你可能会需要从容器中拷贝文件至本地机器。最容易的办法就是把文件放置在/home/ds/notebooks文件夹中，这样的话这些文件就会自动映像到本地机器。

另一种方法也就是利用docker cp命令。假设你想从id为4greg24314的容器中，把/home/ds/notebooks/data.csv文件拷贝至本地机器的/home/vik/文件夹中，你可以输入下面的命令：cp 4greg24134:/home/ds/notebooks/data.csv /home/vik/data.csv。

最后一种方法就是使用Jupyter界面中的download选项。在网页模式下点击一个不是notebook的文件，将会将其下载至本地。如果你已经打开了一个notebook，那么可以先点击File，然后选中download as就可以下载至本地。

安装更多的工具包

如果你想在容器中安装更多的工具包，你可以通过正常的bash命令行命令就可以实现。要想在容器中执行这些命令，你需要运行docker exec命令。这个命令接受容器的id作为参数，以及一个期望运行的命令。例如输入docker exec -it 4greg24134 /bin/bash将会在编码为4greg24134的容器中开启一个shell命令提示符（shell prompt）。-it选项确保我们在容器中打开了一个输入会话，并且可以输入命令。

在运行docker exec命令之后，你就会看到容器中的shell命令提示符（shell prompt）出现。容器此时正通过一个名为ds的虚拟环境运行Python程序，这个虚拟环境已经是处于激活状态的。

接下来，只需要输入`pip install PACKAGE_NAME`就可以安装其他的工具包。例如，你可以使用`pip install requests`来安装requests。

当你希望退出容器的shell终端时，只需要输入`exit`即可。

关闭docker容器

在完成数据处理工作之后，你就可以通过`docker rm -f CONTAINER_ID`来停止docker容器。你应该输入之前获得的容器编码。如果你忘了，你可以运行`docker ps`查看。容器停止运行之后，notebooks会继续存放在你本地用于存放的文件夹中。

更进一步

Docker镜像是通过Dockerfile创建的。Dockerfile指定了镜像中应该安装的包和工具。通过修改Dockerfile，你就可以改变镜像默认按照的包和工具。

如果你想在本文中所使用的镜像基础上做一定修改，可以向我们的Github仓库提交PR，这个仓库中包含了镜像的Dockerfile。我们欢迎大家参与改善当前的镜像，或是添加其他安装了非Python包和工具的镜像。

更多Docker相关教程见以下内容:

Docker安装应用(CentOS 6.5_x64) http://www.linuxidc.com/Linux/2014-07/104595.htm

Ubuntu 14.04安装Docker http://www.linuxidc.com/linux/2014-08/105656.htm

Ubuntu使用VNC运行基于Docker的桌面系统 http://www.linuxidc.com/Linux/2015-08/121170.htm

阿里云CentOS 6.5 模板上安装 Docker http://www.linuxidc.com/Linux/2014-11/109107.htm

Ubuntu 15.04下安装Docker http://www.linuxidc.com/Linux/2015-07/120444.htm

在Ubuntu Trusty 14.04 (LTS) (64-bit)安装Docker http://www.linuxidc.com/Linux/2014-10/108184.htm

在 Ubuntu 15.04 上如何安装Docker及基本用法 http://www.linuxidc.com/Linux/2015-09/122885.htm

Docker 的详细介绍: 请点这里

Docker 的下载地址: 请点这里

本文永久更新链接地址: http://www.linuxidc.com/Linux/2015-11/125542.htm