Pycharm是支持Docker的，我们可以使用Dockers容器来调试我们的程序，当然，如果我们采用Docker调试的话就需要忍受一下Docker的一些缺点。从我的使用情况来看，Docker有以下缺点：

1. 需要重复构建Image：当我们每增加一个Python依赖包时，我们就需要重构一次我们的Image，这需要我们对Docker有一定的了解。
2. 启动/停止速度慢：主要是Container的启动占用了时间，我们的程序真正启动的时间只比平常多了大概20%。
3. Container的重构：该操作是由Pycharm在自动完成的，我们知道Docker是会在每次未指定name创建Container时是会给我们一个名字的。显然Pycharm是记录了这个名字的，但Pycharm会在“程序/Image/系统状态”发生了更改后，比如我们重启了、我们给我们的程序添加了一个.py文件，这种情况下Pycharm就会重构我们的Containers，而且不会删除原来的。

（虽然我们不必在意这些，多了就手动全部删除就行，但看着不爽----处女座慎看）

1. Container系统组件的缺失：如果你是做运维开发的，或者你涉及到了系统操作，请不要用Docker，除非你确定你的Image包含了你想要的。

以上就是我总结的Docker的缺点。

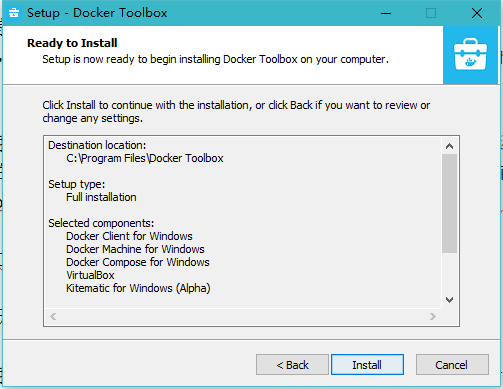
当然，我想Docker在生产中更多的是用在不影响现有生产环境的背景下。

由于我是在Windows下，我采用的是Docker Toolbox环境，因为我还要用Vmware和Virtualbox，所谓的“原生”Docker会打开Hyper-V，相信玩虚拟机的朋友对它深恶痛绝。而且所谓的原生Docker其实也是在Hyper-V中建立一个基于Debian的Docker虚拟机。

DaoIO上说不要用Docker Toolbox，我是赞成的。

正式开始：

1. 我们去Docker官网下载Docker Toolbox，然后下一步下一步的安装就可以了



1. 看下我们的Docker虚拟机

Docker Toolbox是利用virtualbox帮助我们创建了一个名叫default的基于debian的虚拟机，并且帮我们做了一些处理。我们可以修改内存，修改CPU分配数量。（这些在所谓的Windows原生 Docker中是无法做到的，只有2G内存，1核，无法更改）



我们需要关注的地方 ：Docker Toolbox 默认将Users文件夹按照virtualbox的共享方式共享给了Docker虚拟机，也就是说我们的工程其实必须在Users目录下，否则的话是找不到的。

这就是Docker Toolbox的麻烦之处，“显式的虚拟机”，你需要自己处理很多 问题

1. 使用QuickStart或直接在Virtualbox里启动我们的Dockers

。。。此处省略一万行

。。。此处省略两万行

（不会就去看文档去）

如果是初次使用，我们要先Pull一个镜像下来，还需要到修改下我们的加速器，我们可不想忍受那50K的小水管。具体方法参见Daocloud（免费的）

<https://www.daocloud.io/mirror#accelerator-doc>

然后我们先pull一个镜像下来

docker pull python

我一般都采用ubuntu原始image来生成我想到的镜像

1. 初始化我们的Image

。。。此处省略一万行

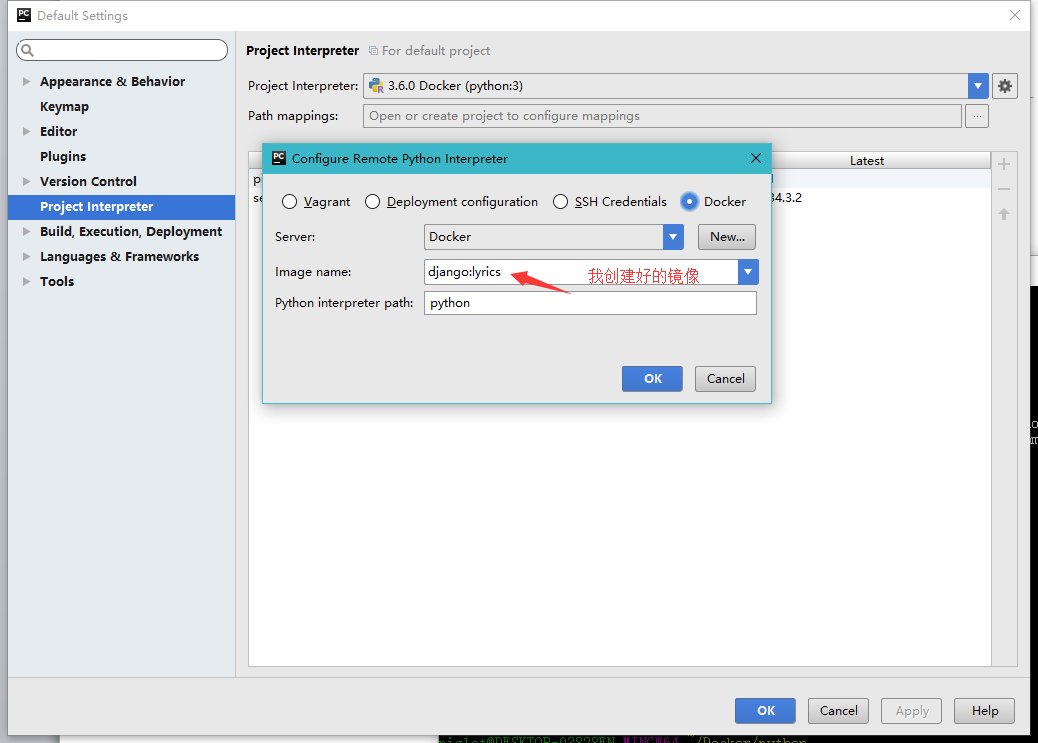
。。。此处省略两万行

（不会就去看文档去）

我已经做好一个用于Django的Image，基于Ubuntu的

1. 创建Pycharm到Docker的连接

打开Pycharm的Interpreter，添加我们的Docker服务

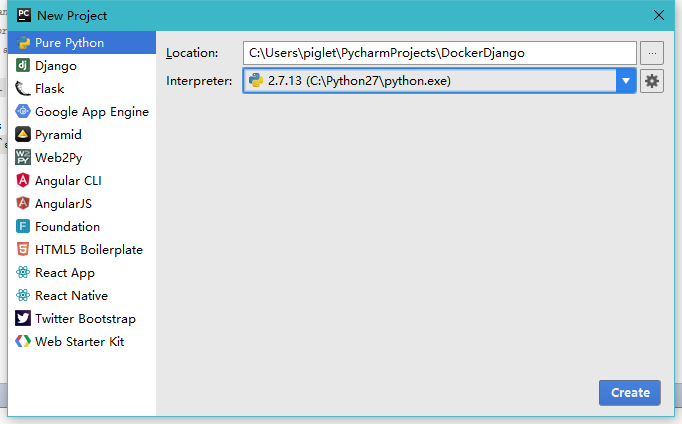
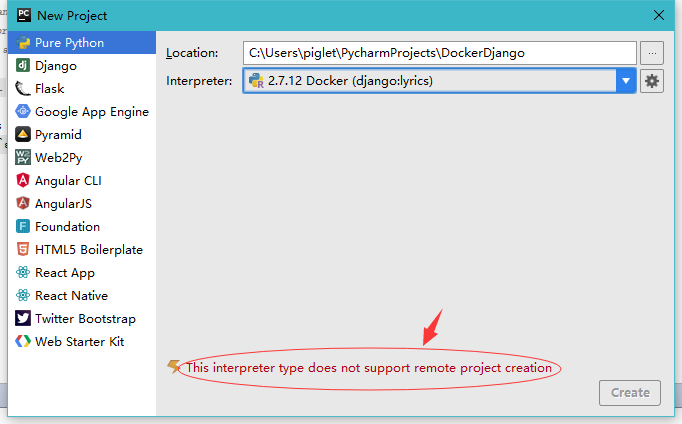


Pycharm会自动配置的，等待我们的只是OK

如果你需要多个，那就按照这种方法去创建多个基于不同Image的Interpreter就行，Pycharm一个Interpreter对应一个Docker Image

1. 创建我们的工程

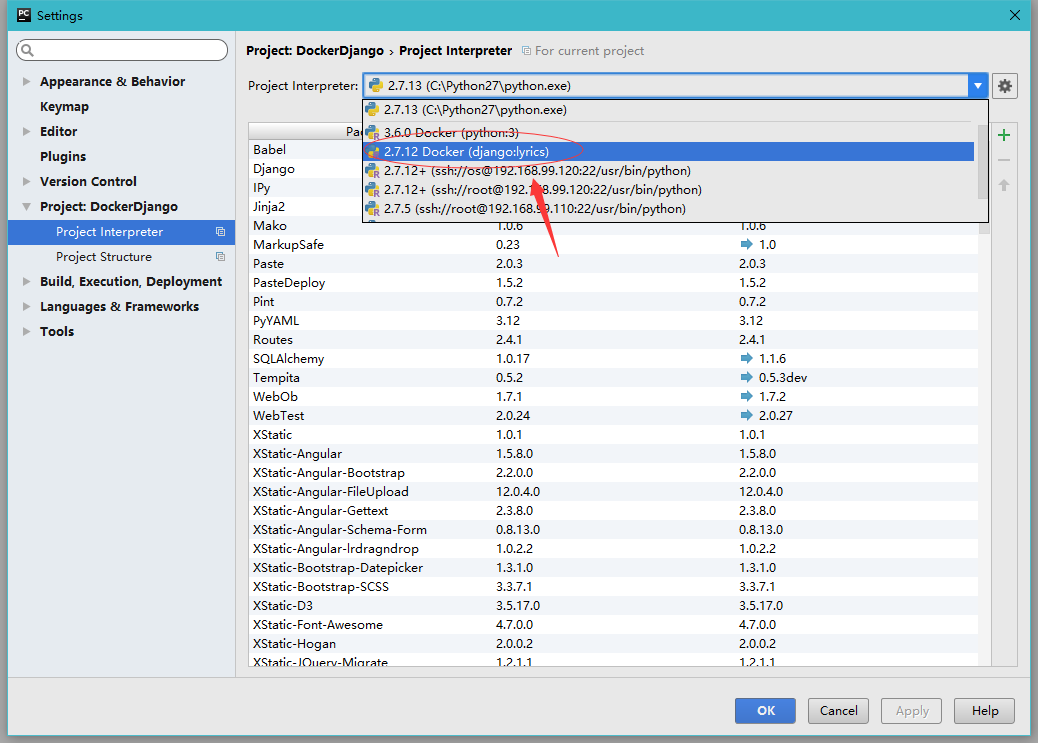
现在我们来创建一个工程 ，还以Django为例（因为Pycharm支持的很好），如前面所说，我们的工程必须创建在Users目录下，否则的话会找不到，用pycharm默认目录就行。

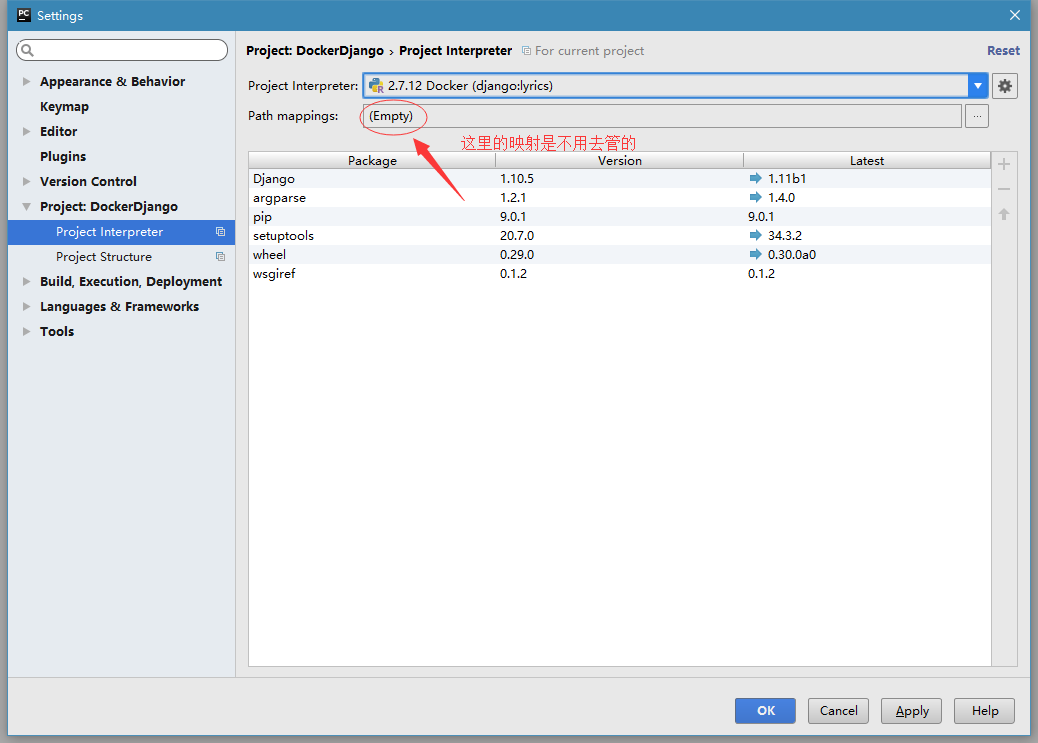


我们需要注意的是，当我们在创建工程时就采用Docker 的Interpreter，会出现如上上图所出现的错误，提示我们这个Interpreter不支持远程创建工程。没关系，我们先换成本地的Interpreter，先把工程创建起来，然后再修改。

1. 修改Interpreter

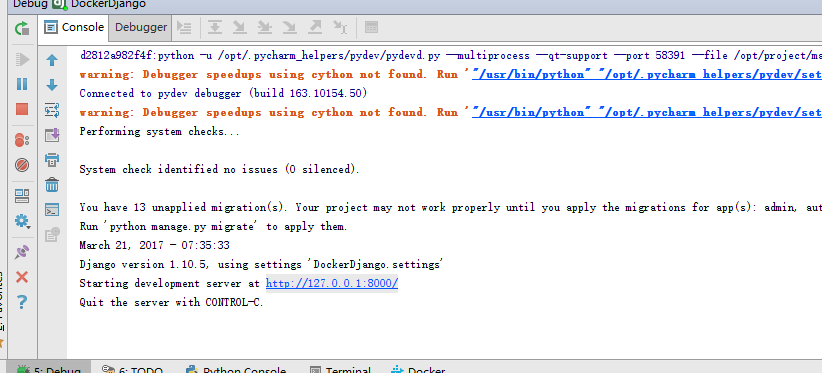
现在我们来把Local Interpreter修改成Docker Interpreter





当你换好以后，Pycharm的右下角会做它该做的事，等待它，而不是停止它。

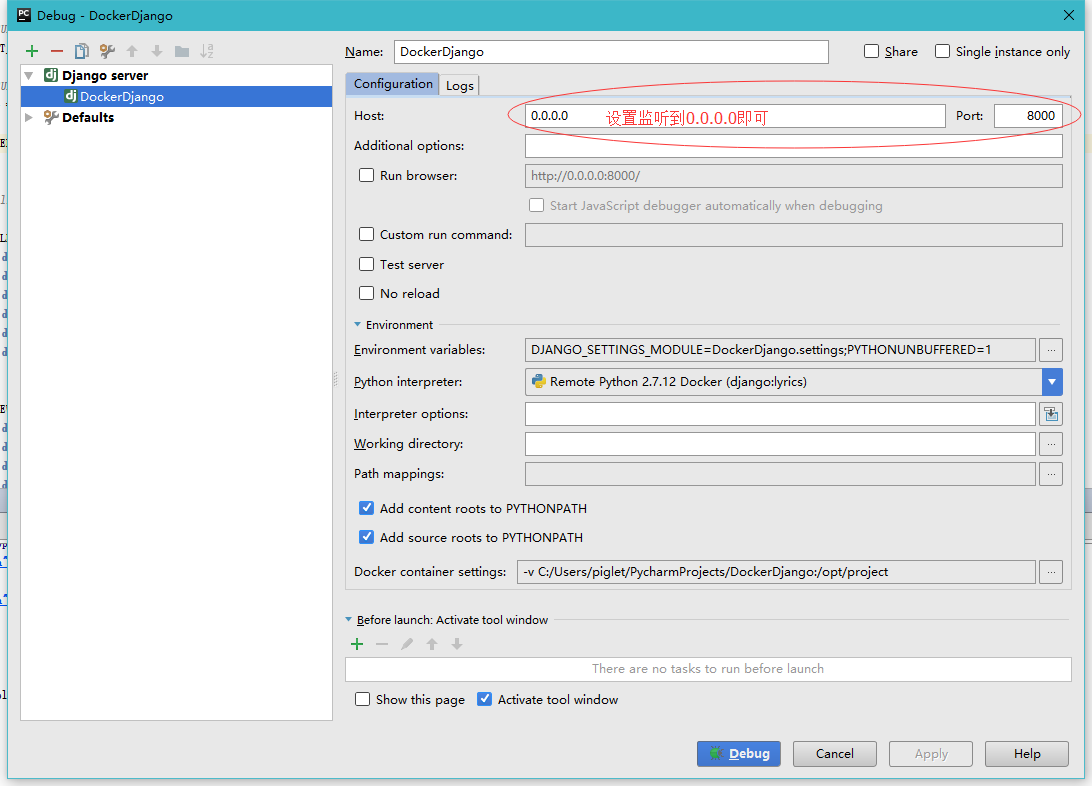
1. Debug我们的工程



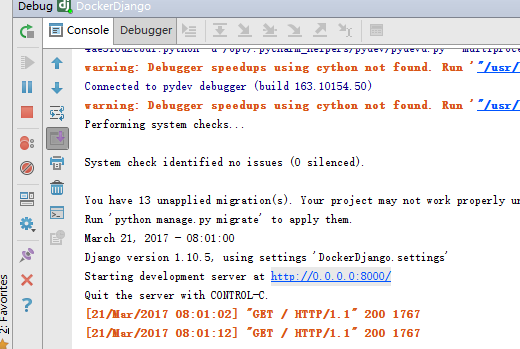
当然，我们现在点击http://127.0.0.1:8000是完全没有反应的，因为我们还没有做端口映射

1. 端口映射

Docker内部映射，将我们的程序映射到Docker虚拟机的IP上（默认是192.168.99.100），我们需要修改我们的debug配置



（某些Django版本需要设置allow\_hosts）



这时候程序已经可以从192.168.99.100:8000端口上访问了，你点击0.0.0.0：8000时pycharm会帮你自动跳转到192.168.99.100:8000地址上的，很牛X，有懂的亲可以告诉我介是个什么原理。

关于端口映射的问题，其实还是有点问题的，我们知道Pycharm是支持Django和Flask的，所以对Django的端口映射能自动建立 ，但对于一些小众的或者自己写的wsgi程序，支持起来就麻烦了，下一篇以pecan为例，帮大家撸一下，有点绕。

写在后面：倒腾了两个晚上，终于写出来了，今天贴出来，原创不易，如有错别字，还请多多包涵。