

记录时间：2019.10.28

公司名称：火鹰科技有限公司

学生姓名：曾伟涛

记录类型：学习笔记

记录阶段：2019.10.28-2019.11.03

docker

前言：学习springBoot为何突然转向学Linux？越学越回去了？我确切的回答你们的疑惑，没有，作为后端开发人员，你将需要将你的文件打包部署到服务器，而如今的服务器大部分都是Linux，那跟docker有什么关系呢，学习Linux的命令不就可以了么？为何还要用大篇幅去描述这两个呢？看到这里的你们，不用但心，往下看，你就会明为什么了。

一：什么是docker

Docker 是一个开源的应用容器引擎，基于 [Go 语言](#) 并遵从 Apache2.0 协议开源。

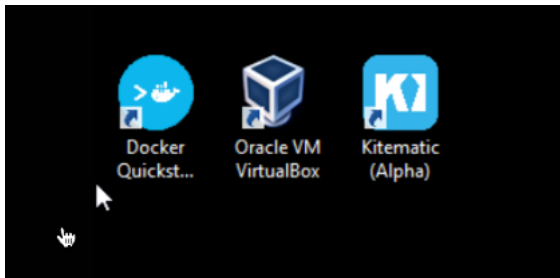
Docker 可以让开发者打包他们的应用以及依赖包到一个轻量级、可移植的容器中，然后发布到任何流行的 Linux 机器上，也可以实现虚拟化。

二：docker的安装

1.win7 win8的安装：

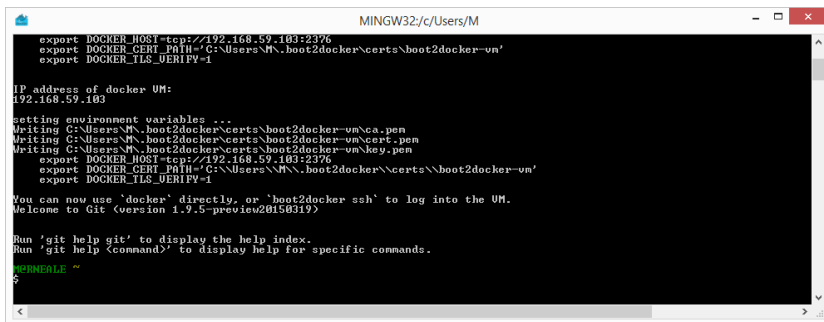
下载dockerToolbox: <http://mirrors.aliyun.com/docker-toolbox/windows/docker-toolbox/>

下载完成之后直接点击安装，安装成功后，桌边会出现三个图标，入下图所示：



点击 Docker QuickStart 图标来启动 Docker Toolbox 终端。

如果系统显示 User Account Control 窗口来运行 VirtualBox 修改你的电脑，选择 Yes。



\$ 符号那你可以输入以下命令来执行。

```
1 $ docker run hello-world
2 Unable to find image 'hello-world:latest' locally
3 Pulling repository hello-world
4 91c95931e552: Download complete
5 a8219747be10: Download complete
6 Status: Downloaded newer image for hello-world:latest
7 Hello from Docker.
```

```
8 This message shows that your installation appears to be working correctly.
9
10 To generate this message, Docker took the following steps:
11 1. The Docker Engine CLI client contacted the Docker Engine daemon.
12 2. The Docker Engine daemon pulled the "hello-world" image from the Docker Hub.
13    (Assuming it was not already locally available.)
14 3. The Docker Engine daemon created a new container from that image which runs the
15    executable that produces the output you are currently reading.
16 4. The Docker Engine daemon streamed that output to the Docker Engine CLI client, which
17    then printed it to your terminal.
18
19 To try something more ambitious, you can run an Ubuntu container with:
20 $ docker run -it ubuntu bash
21
22 For more examples and ideas, visit:
23 https://docs.docker.com/userguide/
```

win10的安装：

开启 Hyper-V



程序和功能

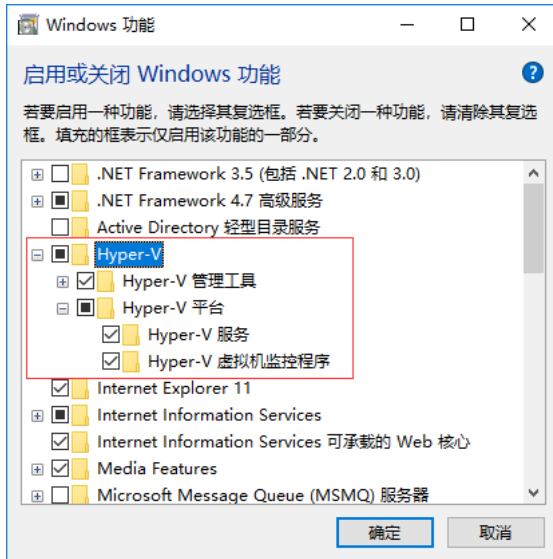
相关设置

程序和功能

启用或关闭Windows功能

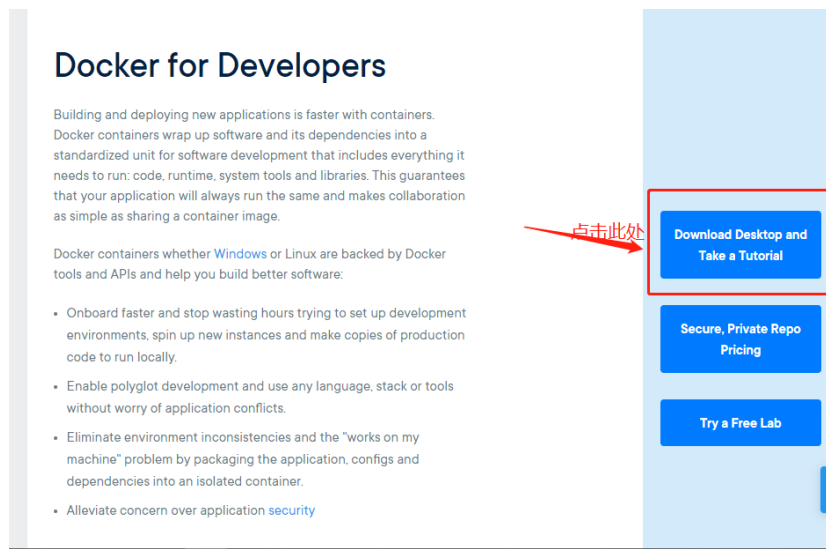


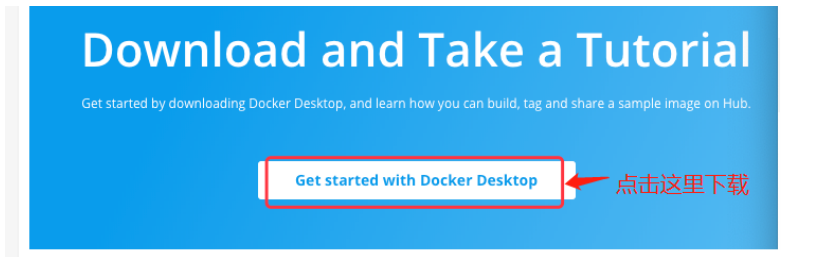
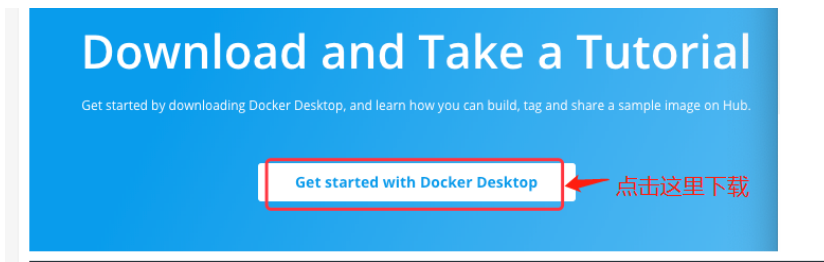
选中Hyper-V



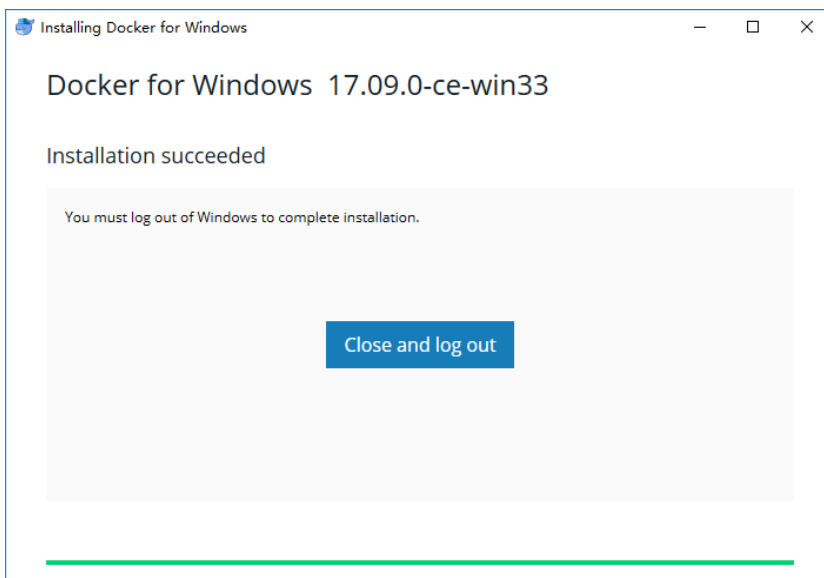
这里确定之后会重启电脑，重启之后步骤如下：

打开此链接：<https://www.docker.com/get-docker>





双击下载的 Docker for Windows Installer 安装文件，一路 Next，点击 Finish 完成安装。



安装完成后，Docker 会自动启动。通知栏上会出现个小鲸鱼的图标

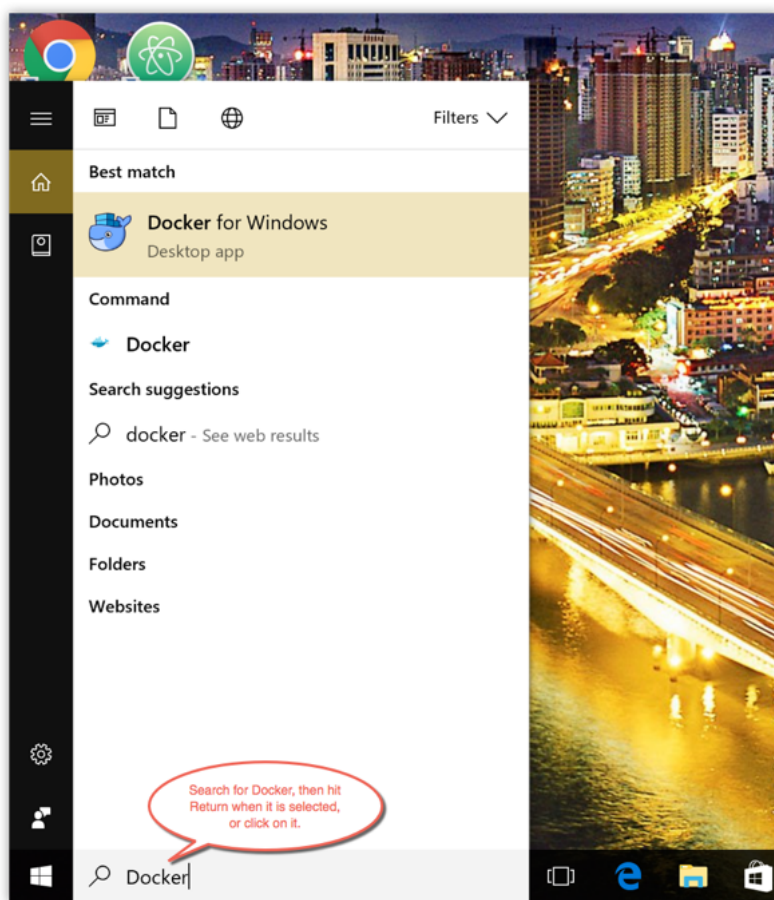


，这表示 Docker 正在运行。

桌边也会出现三个图标，入下图所示：

我们可以在命令行执行 `docker version` 来查看版本号，`docker run hello-world` 来载入测试镜像测试。

如果没启动，你可以在 Windows 搜索 Docker 来启动：



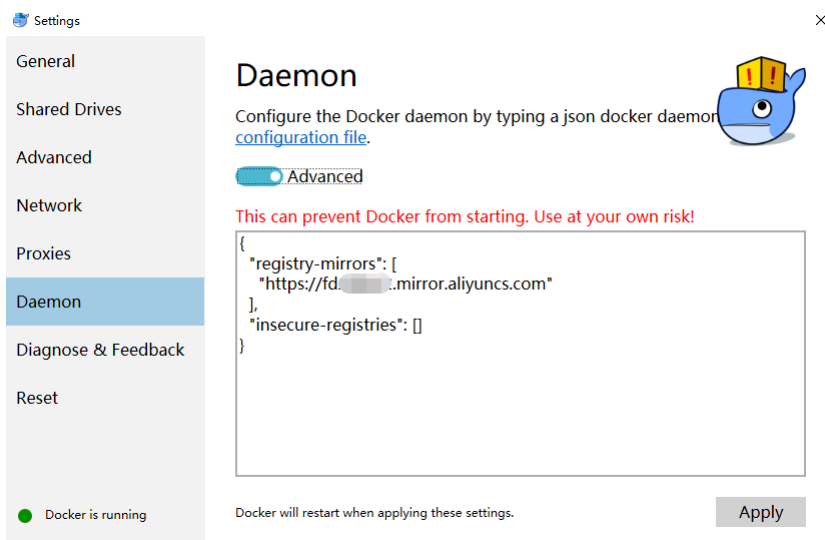
启动后，也可以在通知栏上看到小鲸鱼图标：



镜像加速

Windows 10

对于使用 Windows 10 的系统，在系统右下角托盘 Docker 图标内右键菜单选择 Settings，打开配置窗口后左侧导航菜单选择 Daemon。在 Registrymirrors 一栏中填写加速器地址 <https://registry.docker-cn.com>，之后点击 Apply 保存后 Docker 就会重启并应用配置的镜像地址了。



用win+x+i打开cmd，在cmd上输入以下代码：

```
1 docker run hello-world
2 Unable to find image 'hello-world:latest' locally
3 Pulling repository hello-world
4 91c95931e552: Download complete
5 a8219747be10: Download complete
6 Status: Downloaded newer image for hello-world:latest
7 Hello from Docker.
8 This message shows that your installation appears to be working correctly.
9
10 To generate this message, Docker took the following steps:
11 1. The Docker Engine CLI client contacted the Docker Engine daemon.
12 2. The Docker Engine daemon pulled the "hello-world" image from the Docker Hub.
13    (Assuming it was not already locally available.)
14 3. The Docker Engine daemon created a new container from that image which runs the
15    executable that produces the output you are currently reading.
16 4. The Docker Engine daemon streamed that output to the Docker Engine CLI client, which
17    then printed it to your terminal.
18
19 To try something more ambitious, you can run an Ubuntu container with:
20 $ docker run -it ubuntu bash
21
22 For more examples and ideas, visit:
23 https://docs.docker.com/userguide/
```

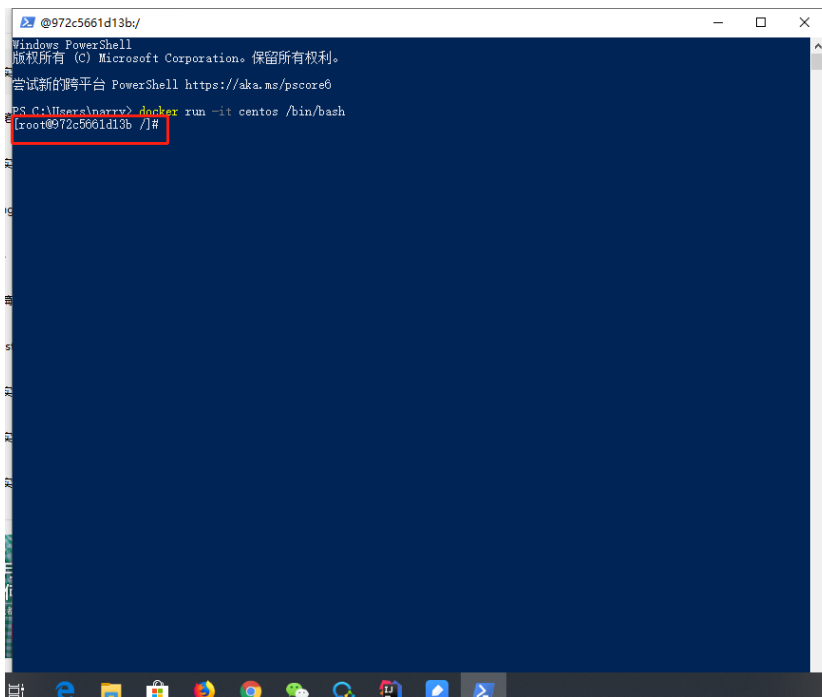
若出现以上这样的代码，则视为成功。

2.进入Linux系统：我以windows 10 为例：

在cmd输入以下代码：

```
1 docker run -it centos /bin/bash
2 或
3 docker run -it ubuntu /bin/bash
```

运行后若本身有其镜像，则立马进入linux，若没有则在安装其镜像



这里就是进入到centos的终端了。

三：docker安装并部署多环境：

1.安装Nginx

打开cmd：win+x+i： 输入以下代码：

```
1 docker search nginx
2 //会出现一大串nginx的镜像
3 //.....此处忽略....//
4 docker pull nginx
5 //默认情况下是最早版本的，若要换版本，在nginx输入:版本号
6 Using default tag: latest
7 //报错，失败了？ 安装不成功，不不不，再走一遍
8 Error response from daemon: Get https://registry-1.docker.io/v2/: net/http: request canceled
9 //再走一遍
10 docker pull nginx
11 Using default tag: latest
12 latest: Pulling from library/nginx
13 8d691f585fa8: Pull complete
14 Status: Downloaded newer image for nginx:latest
15 docker.io/library/nginx:latest
16 //检验是否已经安装了
17 docker images nginx
18 REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
19 nginx                latest             540a289bab6c       8 days ago         126MB
20 //配置启动
21 docker run --name runoob-nginx-test -p 8081:80 -d nginx
22 //返回一大串像是他的序列号
23 //...此处省略...//
```

```
24 //查看是否已经在运行了
25 docker ps
26 CONTAINER ID        IMAGE               ...          PORTS          NAMES
27 6dd4380ba708        nginx              ...          0.0.0.0:8081->80/tcp  runoob-nginx-test
```

在浏览器输入：http://127.0.0.1: 8081会出现以下效果：



nginx 的部署

创建目录，用于存放后面的相关东西：代码如下：

```
1 mkdir -p ~/nginx/www ~/nginx/logs ~/nginx/conf
2 //以上写法你们觉得可行吗？？？（可行进行下一步）
3 //拷贝容器内nginx默认配置文件到本地路径下
4 docker cp [容器ID]:/etc/nginx/nginx.conf ~/nginx/conf
5 //容器ID是什么，怎么查？这样的写法又是否可行？（可行进行下一步）
6 //nginx的部署
7 docker run -d -p 8082:80 --name [自定义: eg: nginx-test] -v ~/nginx/www:/usr/share/nginx/html
8 //成功会出现一大串字符，如下例子：
9 b7cbf48997af6c671b2b36fc482bad787cc7b51b740008447c715a916bd0fb62
```

在nginx的www目录内写一个html文件，然后在浏览器地址栏输入：http://127.0.0.1:8082会出现以下效果：（以我自己为例）



第一次使用docker部署nginx

看到便是成功了！

解码：

1.第一行代码是否可行？----不行

正确写法：

```
1 mkdir -p ~/nginx/www, ~/nginx/logs, ~/nginx/conf
```

2.容器ID怎么查？

```
1 docker ps
2 CONTAINER ID        IMAGE               COMMAND          CREATED          ST
3 0ec94331ed90        nginx              "nginx -g 'daemon of..."  About an hour ago  Up
```


这里的container id便是容器ID

3.写法哪里不行? ,正确写法如下:

```
1 docker cp [容器ID]:/etc/nginx/nginx.conf nginx/conf
```

2.安装PHP

```
1 docker search php
2 //*****此处省略搜索出来的php版本*****//
3 docker pull php:5.6-fpm
4 //最后三条出现如下类似的,即为成功
5 Digest: sha256:4f070f1b7b93cc5ab364839b79a5b26f38d5f89461f7bc0cd4bab4a3ad7d67d7 9.981M
6 Status: Downloaded newer image for php:5.6-fpm
7 docker.io/library/php:5.6-fpm
8 //检验是否确实成功
9 docker images php
```

出现以下代码的即为成功:

```
PS C:\Users\parry> docker images php
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
php                  5.6-fpm     3458979c7744     9 months ago    344MB
```

创建目录,方便后面的操作:

```
1 mkdir ~/nginx/conf/conf.d
```

nginx部署php

```
1 docker run --name parry-php-nginx -p 8083:80 -d /
2 -v ~/nginx/www:/usr/share/nginx/html:ro /
3 -v ~/nginx/conf/conf.d:/etc/nginx/conf.d:ro /
4 --link parryzeng-php:php nginx
```


注意这里的"/ "坑,我是为了分行才这样写,你们复制的时候注意把/去掉

```
PS C:\Users\parry> docker run --name parry-php-nginx -p 8083:80 -d /
>> -v ~/nginx/www:/usr/share/nginx/html:ro /
>> -v ~/nginx/conf/conf.d:/etc/nginx/conf.d:ro /
>> --link parryzeng-php:php nginx
```

在www目录下写php,以我为例(index.php)

```
1 <?php
2 echo phpinfo();
3 ?>
```

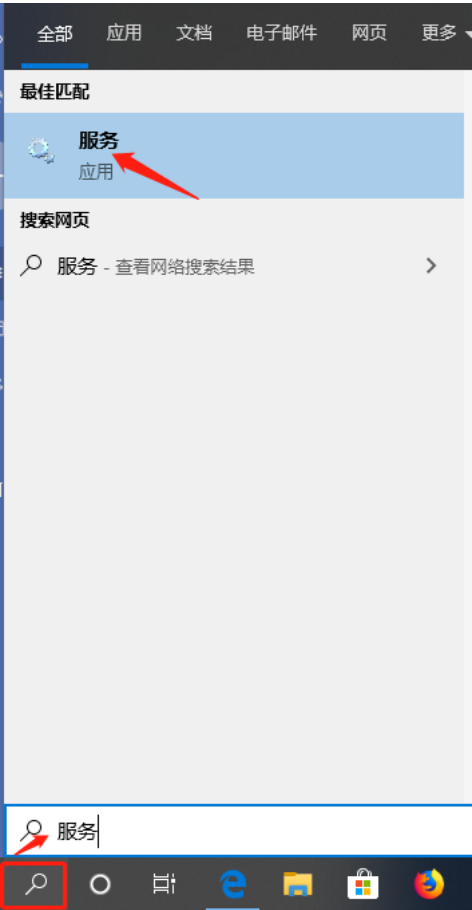
在浏览器地址栏输入"http://127.0.0.1:8083/index.php",显示的效果如下:

<div> <div>127.0.0.1:8083/index.php</div> <div> <div></div> <div>☆</div> </div> </div>	
<div> <div>PHP Version 5.6.40</div> <div>  </div> </div>	
System	Linux 77a6691164c 4.9.184-linuxkit #1 SMP Tue Jul 2 22:58:16 UTC 2019 x86_64
Build Date	Jan 23 2019 00:14:48
Configure Command	./configure '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--enable-option-checking=fatal' '--with-mhash' '--enable-ftp' '--enable-mbstring' '--enable-mysqlnd' '--with-curl' '--with-libedit' '--with-openssl' '--with-zlib' '--with-ibdir=/lib/x86_64-linux-gnu' '--enable-fpm' '--with-fpm-user=www-data' '--with-fpm-group=www-data' '--disable-cgi' 'build_alias=x86_64-linux-gnu' 'CFLAGS=-fstack-protector-strong -fPIC -pie -O2' 'LDFLAGS=-Wl,-O1 -Wl,-hash-style=both -pie' 'CXXFLAGS=-fstack-protector-strong -fPIC -pie -O2'
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	(none)
Scan this dir for additional .ini files	/usr/local/etc/php/conf.d
Additional .ini files parsed	(none)
PHP API	20131106
PHP Extension	20131226
Zend Extension	220131226
Zend Extension Build	API220131226.NTS
PHP Extension Build	API20131226.NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	disabled

证明已经部署成了！！

3.安装MySQL

在 安装和部署之前，请先停止本地本身的mysql服务，打开



关闭此服务

MySQL57

正在... 自动

网络服务

安装MySQL:

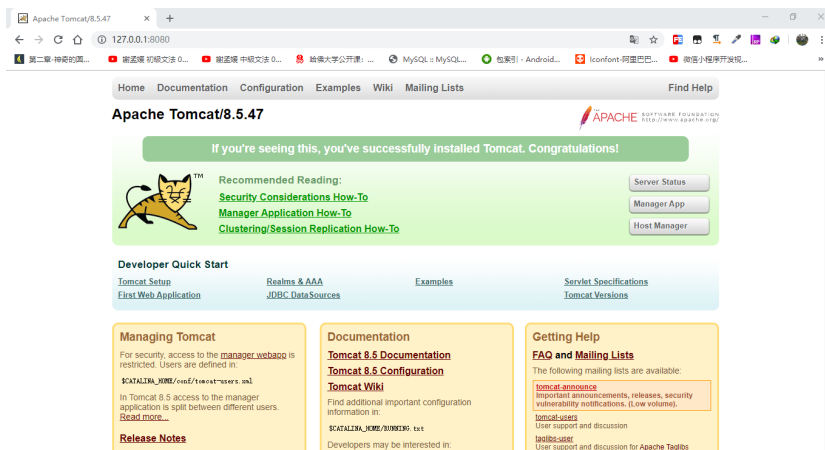
```
1 //查找镜像
2 docker search mysql
3 //拉取镜像
4 docker pull mysql:5.6
5 //创建目录
6 mkdir -p ~/nginx/mysql/data, ~/nginx/mysql/logs, ~/nginx/mysql/conf
7 //检验是否安装
8 docker images mysql
9 //启动mysql
10 docker run -p 3306:3306 --name [自定义]
11 -v $pwd/nginx/mysql/conf:/etc/mysql/conf.d
12 -v $pwd/nginx/mysql/logs:/logs
13 -v $pwd/data:/var/lib/mysql
14 -e MYSQL_ROOT_PASSWORD=password -d mysql:5.6
15 //若返回序列号则是安装成功
16 //检验是否启动成功
17 docker ps
```

注意: 第10到14在cmd输入是一行的, password是自定义输入。

4.安装tomcat

```
1 //查找镜像
2 docker search tomcat
3 //拉取镜像
4 docker pull tomcat
5 //创建目录
6 mkdir -p ~/nginx/tomcat/webapps/test, ~/nginx/tomcat/logs, ~/nginx/tomcat/conf
7 //检验是否安装
8 docker images tomcat
9 //启动tomcat
10 docker run -p 8080:8080 --name [自定义]
11 -v $pwd/test:/usr/local/tomcat/webapps/test -d tomcat
12 //若返回序列号则是安装成功
13 //检验是否启动成功
14 docker ps
```

在浏览器地址栏输入: 127.0.0.1: 8080, 效果如下:



5.安装python

```

1 //查找镜像
2 docker search python
3 //拉取镜像
4 docker pull python
5 //创建目录
6 mkdir -p ~/nginx/python/myapp
7 //检验是否安装
8 docker images python
9 //启动python, 启动之前, 请去myapp下创建helloworld.py
10 //helloworld.py
11 print("Hello,World!");
12 //启动python
13 docker run -v $pwd/nginx/python/myapp:/usr/src/myapp -w /usr/src/myapp python:3.5 python helloworld.py
14 //若返回序列号则是安装成功
15

```

效果如下:

```

Hello,World!

```

6.安装redis

```

1 //查找镜像
2 docker search redis
3 //拉取镜像
4 docker pull redis:3.2
5 //创建目录
6 mkdir -p ~/nginx/redis/data
7 //检验是否安装
8 docker images redis:3.2
9 //启动redis
10 docker run -p 6379:6379 -v $pwd/nginx/redis/data:/data -d redis:3.2 redis-server --appendonly yes
11 //检验是否启动
12 docker ps
13 //此行会显示redis相关信息, 记得复制其CONTAINER ID

```

```
14 //连接容器 redis
15 docker exec -it CONTAINER ID redis-cli
16 //显示redis信息
17 127.0.0.1: 6379>info
18
```

显示如下:

```
127.0.0.1:6379> info
# Server
redis_version:3.2.12
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:b0df607ad3315254
redis_mode:standalone
os:Linux 4.9.184-linuxkit x86_64
arch_bits:64
```

7.安装MongoDB 以及apache自行安装，写法步骤跟以上基本一样。