

【Docker 实战】 - Registry & Portus 搭建详解

前言：

首先，Docker Hub 是一个很好的用于管理公共镜像的地方，我们可以在上面找到想要的镜像（Docker Hub 的下载量已经达到数亿次）；而且我们也可以把自己的镜像推送上去。但是，有的时候，使用场景需要我们有一个私有的镜像仓库用于管理自己的镜像，这个时候我们就通过 Registry 来实现此目的。本文详细介绍了本地镜像仓库 Docker Registry 以及 Portus 的搭建过程（Portus 是一个带 UI 管理的仓库管理软件），对于文中细节有兴趣或有疑问的朋友欢迎加群讨论。

注：文章由云舒网络原创，转载请注明出处。



了解最新云计算资讯，关注云舒网络官方微信

目录

● [Registry 搭建篇](#)

一. [Docker & Registry 环境准备](#)

1. [Docker 安装](#)

2. [Registry 安装](#)

二. [Registry 原理](#)

三. [Registry 配置](#)

1. [启动 Registry 容器](#)

2. [客户机访问 Registry](#)

3. [客户端向 Registry 存放镜像](#)

4. [客户端向仓库 Pull 镜像](#)

● [Portus 搭建篇](#)

一. [Portus 安装](#)

1. [安装 docker-compose](#)

2. [从 git 到 portus 的代码搭建](#)

3. [安装 Portus 程序](#)

4. [修改 Docker 配置文件](#)

二. [Portus 配置与验证](#)

1. [登录配置程序](#)

2. [上传镜像测试](#)

3. [下载镜像测试](#)

● [总结](#)

● Registry 搭建篇

一 . Docker & Registry 环境准备

在谈到 Registry 的部署之前，我们首先要考虑设计一个什么样 Registry 的仓库环境，是部署测试环境，还是生产环境。如果是用于生产环境发布 Registry，必须考虑如下因素：

- a. 应在何处存储镜像？
- b. 用户的权限是否受控？
- c. 当发生问题，如何解决？日志是否可以查看？
- d. 如何快速提取镜像？（注：这是至关重要的，如果依赖镜像进行构建测试环境、生产环境部署或自动化系统，这是取决仓库是否有生命力的最重要指标。）

本实例把存储镜像的路径放置到宿主机的文件路径下。例如：/opt/myregistry 目录中。这样即使宿主机中 Docker Registry 出现问题，我们再重创建一个，把此路径添加到新 Registry Server 中，那么之前上传的镜像文件仍然存在，这就更方便大家使用。下面是安装之前进行的一些准备工作：

1.Docker 安装：

首先，选择一个合适的 PC 机做宿主机，其配置参数如下：

CPU：Intel E8400 Duo CPU 3.0GHz RAM: 2G disk:300G

CPU 查看命令: `more /proc/cpuinfo |grep "model name"`

Mem 查看命令: `grep MemTotal /proc/meminfo`

```
[root@bogon ~]# docker version
Client:
 Version:      1.10.2-rc1
 API version:  1.22
 Go version:   gol.5.3
 Git commit:   89dafc4
 Built:        Sat Feb 20 04:46:05 2016
 OS/Arch:      linux/amd64

Server:
 Version:      1.10.2-rc1
 API version:  1.22
 Go version:   gol.5.3
 Git commit:   89dafc4
 Built:        Sat Feb 20 04:46:05 2016
 OS/Arch:      linux/amd64
```

2.Registry 安装：

由于 [Docker](#) 和 Registry 更新比较快，所以我们在此宿主机中进行安装的时候，选择 Registry2.1 版本进行安装。

```
yum -y install docker-registry
```

```
[root@bogon ~]# yum -y install docker-registry
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: centos.ustc.edu.cn
 * extras: centos.ustc.edu.cn
 * updates: centos.ustc.edu.cn
Resolving Dependencies
--> Running transaction check
--> Package docker-registry.x86_64 0:0.9.1-7.el7 will be installed
--> Processing Dependency: python-sqlalchemy >= 0.9.8-1 for package: docker-registry-0.9.1-7.el7.x86_64
--> Processing Dependency: python-backports-lzma >= 0.0.2-8 for package: docker-registry-0.9.1-7.el7.x86_64
--> Processing Dependency: python-gunicorn for package: docker-registry-0.9.1-7.el7.x86_64
--> Processing Dependency: python-gevent for package: docker-registry-0.9.1-7.el7.x86_64
--> Processing Dependency: python-flask for package: docker-registry-0.9.1-7.el7.x86_64
--> Processing Dependency: python-blinker for package: docker-registry-0.9.1-7.el7.x86_64
--> Running transaction check
--> Package python-backports-lzma.x86_64 0:0.0.2-8.el7 will be installed
--> Package python-blinker.noarch 0:1.3-2.el7 will be installed
--> Package python-flask.noarch 1:0.10.1-4.el7 will be installed
--> Processing Dependency: python-werkzeug for package: 1:python-flask-0.10.1-4.el7.noarch
--> Processing Dependency: python-jinja2 for package: 1:python-flask-0.10.1-4.el7.noarch
--> Processing Dependency: python-itsdangerous for package: 1:python-flask-0.10.1-4.el7.noarch
--> Package python-gevent.x86_64 0:1.0-2.el7 will be installed
--> Processing Dependency: python-greenlet for package: python-gevent-1.0-2.el7.x86_64
--> Processing Dependency: libev.so.4()(64bit) for package: python-gevent-1.0-2.el7.x86_64
--> Package python-gunicorn.noarch 0:18.0-2.el7 will be installed
--> Package python-sqlalchemy.x86_64 0:0.9.8-1.el7 will be installed
--> Running transaction check
--> Package libev.x86_64 0:4.15-6.el7 will be installed
--> Package python-greenlet.x86_64 0:0.4.2-3.el7 will be installed
--> Package python-itsdangerous.noarch 0:0.23-2.el7 will be installed
--> Package python-jinja2.noarch 0:2.7.2-2.el7 will be installed
--> Processing Dependency: python-babel >= 0.8 for package: python-jinja2-2.7.2-2.el7.noarch
--> Processing Dependency: python-markupsafe for package: python-jinja2-2.7.2-2.el7.noarch
--> Package python-werkzeug.noarch 0:0.9.1-2.el7 will be installed
--> Running transaction check
--> Package python-babel.noarch 0:0.9.6-8.el7 will be installed
--> Package python-markupsafe.x86_64 0:0.11-10.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

Package Arch Version Repository Size
Installing:
docker-registry x86_64 0.9.1-7.el7 extras 123 k
Installing for dependencies:
libev x86_64 4.15-6.el7 extras 44 k
python-babel noarch 0.9.6-8.el7 base 1.4 M
python-backports-lzma x86_64 0.0.2-8.el7 extras 25 k
```

```
192.168.4.38 192.168.4.121 192.168.0.70
(10/14): python-gunicorn-18.0-2.el7.noarch.rpm | 171 kB 00:00:00
(11/14): python-jinja2-2.7.2-2.el7.noarch.rpm | 515 kB 00:00:00
(12/14): python-markupsafe-0.11-10.el7.x86_64.rpm | 25 kB 00:00:00
(13/14): python-werkzeug-0.9.1-2.el7.noarch.rpm | 562 kB 00:00:01
(14/14): python-sqlalchemy-0.9.8-1.el7.x86_64.rpm | 2.9 MB 00:00:01
-----
Total 729 KB/s | 6.3 MB 00:00:08
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Installing : python-blinker-1.3-2.el7.noarch 1/14
Installing : python-werkzeug-0.9.1-2.el7.noarch 2/14
Installing : python-sqlalchemy-0.9.8-1.el7.x86_64 3/14
Installing : python-gunicorn-18.0-2.el7.noarch 4/14
Installing : libev-4.15-6.el7.x86_64 5/14
Installing : python-itsdangerous-0.23-2.el7.noarch 6/14
Installing : python-greenlet-0.4.2-3.el7.x86_64 7/14
Installing : python-gevent-1.0-2.el7.x86_64 8/14
Installing : python-babel-0.9.6-8.el7.noarch 9/14
Installing : python-backports-lzma-0.0.2-8.el7.x86_64 10/14
Installing : python-markupsafe-0.11-10.el7.x86_64 11/14
Installing : python-jinja2-2.7.2-2.el7.noarch 12/14
Installing : python-flask-0.10.1-4.el7.noarch 13/14
Installing : docker-registry-0.9.1-7.el7.x86_64 14/14
Verifying : python-gevent-1.0-2.el7.x86_64 1/14
Verifying : python-jinja2-2.7.2-2.el7.noarch 2/14
Verifying : python-flask-0.10.1-4.el7.noarch 3/14
Verifying : python-markupsafe-0.11-10.el7.x86_64 4/14
Verifying : python-backports-lzma-0.0.2-8.el7.x86_64 5/14
Verifying : python-babel-0.9.6-8.el7.noarch 6/14
Verifying : python-greenlet-0.4.2-3.el7.x86_64 7/14
Verifying : python-itsdangerous-0.23-2.el7.noarch 8/14
Verifying : libev-4.15-6.el7.x86_64 9/14
Verifying : python-gunicorn-18.0-2.el7.noarch 10/14
Verifying : docker-registry-0.9.1-7.el7.x86_64 11/14
Verifying : python-sqlalchemy-0.9.8-1.el7.x86_64 12/14
Verifying : python-werkzeug-0.9.1-2.el7.noarch 13/14
Verifying : python-blinker-1.3-2.el7.noarch 14/14
Installed:
docker-registry.x86_64 0:0.9.1-7.el7
Dependency Installed:
libev.x86_64 0:4.15-6.el7 python-babel.noarch 0:0.9.6-8.el7 python-backports-lzma.x86_64 0:0.0.2-8.el7 python-blinker.noarch 0:1.3-2.el7
python-flask.noarch 1:0.10.1-4.el7 python-gevent.x86_64 0:1.0-2.el7 python-greenlet.x86_64 0:0.4.2-3.el7 python-gunicorn.noarch 0:18.0-2.el7
python-itsdangerous.noarch 0:0.23-2.el7 python-jinja2.noarch 0:2.7.2-2.el7 python-markupsafe.x86_64 0:0.11-10.el7 python-sqlalchemy.x86_64 0:0.9.8-1.el7
python-werkzeug.noarch 0:0.9.1-2.el7
Complete!
[root@bogon ~]#
```

至此，Docker 和 Registry 的安装已经完成，下面开始进行配置和测试。

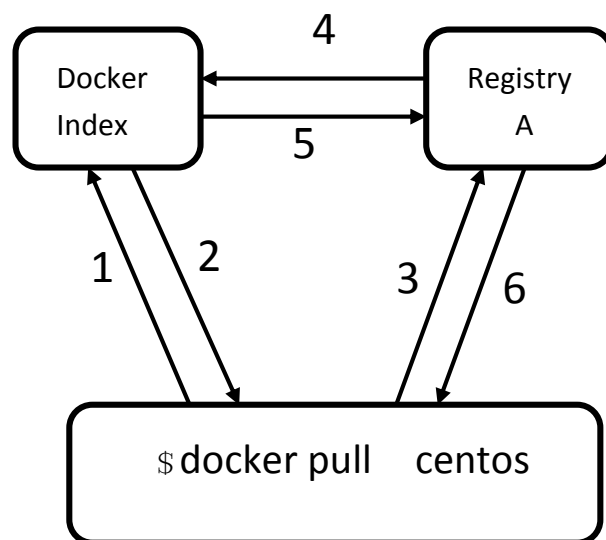
```
Last login: Tue Feb 23 13:46:00 2016
[root@bogon ~]# ls
anaconda-ks.cfg
[root@bogon ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
[root@bogon ~]# ls
anaconda-ks.cfg
[root@bogon ~]# cd /
[root@bogon /]# ls
bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys usr var
[root@bogon /]# cd opt/
[root@bogon opt]# ls
rh
[root@bogon opt]# mkdir myregistry
[root@bogon opt]# ls
myregistry rh
[root@bogon opt]# cd /
[root@bogon /]# systemctl enable docker
[root@bogon /]# systemctl enable docker-registry
[root@bogon /]# systemctl start docker
[root@bogon /]# systemctl start docker-registry
[root@bogon /]#
```

注：安装完成后需要对 Docker 和 Registry 进行 enable and start。

二 . Registry 原理

Docker 模型的核心部分是有效的利用分层镜像机制，镜像可以通过分层来进行继承，基于基础镜像，可以制作各种具体的应用镜像。不同的 Docker 容器可以共享一些基础的文件系统层，同时再加上自己独有的改动层，大大提高了存储的效率。由于最终镜像最终是以 tar.gz 的方式静态存储在服务器端，这种存储适用于对象存储而不是块存储。

一次 docker pull 发生的交互：



1. Client 向 Index 请求，知道从哪里下载 CentOS
2. Index 回复
3. CentOS 在 RegistryA
4. CentOS 的 Checksum，所有层的 Token
5. Client 向 Registry A 请求，CentOS 的所有层。Registry A 负责存储 CentOS，以及它所依赖的层
6. Registry A 向 Index 发起请求，验证用户 Token 的合法性
7. Index 返回这次请求是否合法
8. Client 从 Registry 下载所有的层
9. Registry 从后端存储中获取实际的文件数据，返给 Client

三 . Registry 配置

1 .启动 Registry 容器 :

```
sudo docker run -d -p 80:5000 --restart=always --name registry -v /opt/myregistry:/var/lib/registry registry:2
```

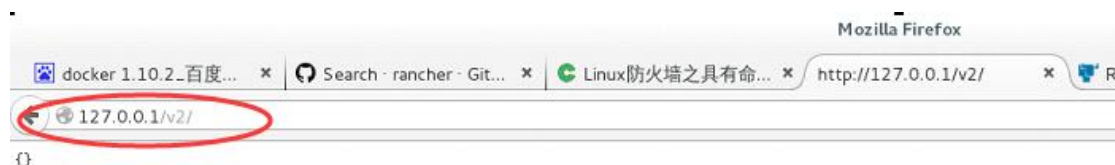
```
[root@bogon /]# docker run -d -p 80:5000 --restart=always --name registry -v /opt/myregistry:/var/lib/registry registry:2
Unable to find image 'registry:2' locally
2: Pulling from library/registry
7268d8f794c4: Pull complete
a3ed95caeb02: Pull complete
6915c066a24f: Pull complete
7f04289ca403: Pull complete
7539a42f6147: Pull complete
Digest: sha256:339d702cf9a4b0aa665269cc36255ee7ce424412d56bee9ad8a247afe8c49ef1
Status: Downloaded newer image for registry:2
aa9f9bdc8b1592358100b58c38054016f0c7242756b83b2ac0dfde4d4b37037e
[root@bogon /]#
```

Registry 服务默认会将上传的镜像保存在容器的/var/lib/registry，我们将主机的/opt/registry 目录挂载到该目录，即可实现将镜像保存到主机的/opt/registry 目录了。

运行 docker ps 看一下容器情况:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
aa9f9bdc8b15	registry:2	"/bin/registry /etc/d"	51 seconds ago	Up 49 seconds	0.0.0.0:80->5000/tcp	registry

说明我们已经启动了 Registry 服务，打开浏览器输入 <http://127.0.0.1:80/v2/>，出现下面情况说明 Registry 运行正常。



2 .客户机访问 Registry

对于需要访问 Registry 仓库的客户机来说，首先需要修改/etc/sysconfig/docker 的配置文件。添加 `--insecure-registry 192.168.0.70:80`



```
# /etc/sysconfig/docker

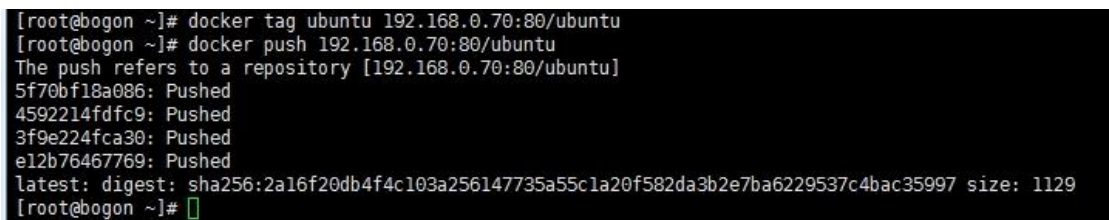
# Modify these options if you want to change the way the docker daemon runs
#OPTIONS='--selinux-enabled'
OPTIONS='--selinux-enabled --insecure-registry 192.168.0.70:80'

DOCKER_CERT_PATH=/etc/docker

# If you want to add your own registry to be used for docker search and docker
# pull use the ADD_REGISTRY option to list a set of registries, each prepended
# with --add-registry flag. The first registry added will be the first registry
# searched.
#ADD_REGISTRY='--add-registry registry.access.redhat.com'
```

3. 客户端向 Registry 存放镜像

首先，需要 docker tag 给需要上传的镜像文件打标。然后，再从本地上传镜像到仓库。



```
[root@bogon ~]# docker tag ubuntu 192.168.0.70:80/ubuntu
[root@bogon ~]# docker push 192.168.0.70:80/ubuntu
The push refers to a repository [192.168.0.70:80/ubuntu]
5f70bf18a086: Pushed
4592214fd9c9: Pushed
3f9e224fca30: Pushed
e12b76467769: Pushed
latest: digest: sha256:2a16f20db4f4c103a256147735a55c1a20f582da3b2e7ba6229537c4bac35997 size: 1129
[root@bogon ~]#
```

4. 客户端向仓库 Pull 镜像

在另外一台主机上使用 pull 从 192.168.0.70 的仓库中把镜像给 Pull 下来。


```
1 192.168.4.38 x 2 192.168.4.121 x 3 192.168.0.70 x 4 192.168.0.24 x 5 192.168.4.202 x +
[root@bogon ~]# docker pull 192.168.0.70:80/ubuntu
Using default tag: latest
latest: Pulling from ubuntu
6edcc89ed412: Pull complete
bdf37643ee24: Pull complete
ea0211d47051: Pull complete
a3ed95caeb02: Pull complete
Digest: sha256:2a16f20db4f4c103a256147735a55c1a20f582da3b2e7ba6229537c4bac35997
Status: Downloaded newer image for 192.168.0.70:80/ubuntu:latest
[root@bogon ~]#
```

```
[root@bogon ~]# docker pull 192.168.0.70:80/ubuntu
Using default tag: latest
latest: Pulling from ubuntu
6edcc89ed412: Pull complete
bdf37643ee24: Pull complete
ea0211d47051: Pull complete
a3ed95caeb02: Pull complete
Digest: sha256:2a16f20db4f4c103a256147735a55c1a20f582da3b2e7ba6229537c4bac35997
Status: Downloaded newer image for 192.168.0.70:80/ubuntu:latest
[root@bogon ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
rancher/agent        v0.9.2             f8c9e328157e       2 weeks ago        356.8 MB
rancher/agent-instance v0.8.0             f4c7ed487037       3 weeks ago        316.1 MB
centos                latest              9fd593454178       8 weeks ago        196.6 MB
192.168.0.70:80/ubuntu latest              d59bdb51bb5c       10 weeks ago       187.9 MB
[root@bogon ~]# docker run -it 192.168.0.70:80/ubuntu /bin/bash
root@414726f070fe:/# uname -a
Linux 414726f070fe 3.10.0-327.10.1.el7.x86_64 #1 SMP Tue Feb 16 17:03:50 UTC 2016 x86_64 x86_64 x86_64 GNU/Linux
root@414726f070fe:/#
```

通过以上操作就可以看到我们已经 Pull 的镜像，然后可以运行此镜像，开始你的应用之旅。到此应该说 Registry 基本搭建完成，但是仍然不适合实际的使用。因为不方便管理和查看到上传的镜像和权限设置。接下来我们介绍一个带 UI 管理的仓库管理软件——

【Portus】

● Portus 搭建篇

注：请勿将本篇作为上篇 Registry 安装内容延续，以下内容全新环境下的 Portus 搭建

一 . Portus 安装

Portus(by SUSE)是用于 Docker Registry API (v2) 的开源前端和授权工具，最低要求注册表版本是 2.1。它可以作为授权服务器和用户界面，用于新一代的 Docker Registry。具有以下优点：

1.) 安全：Portus 实现了最新的 Docker Registry 中定义的新的授权方案。它允许对你所有的镜像进行细颗粒度控制，你可以决定哪个用户和团队可 push/pull 镜像。

2.) 轻松管理用户：在 Portus 映射你的公司，可以定义任意数量的 Team，并从 Team 添加和移除用户。Team 有三种类型的用户：Viewers，只能 pull 镜像；Contributors，可以 push/pull 镜像；Owners，类似 contributors，但可以从 team 添加或移除用户。

3.) 搜索：Portus 提供你的私人注册表的内容的预览，同时有一个快速搜索镜像的功能。

4.) 审计：用户的所有相关事件都会被 Portus 自动记录，并可被管理员进行用户分析。

1.安装 Docker-Compose

首先，yum 添加源

```
[root@bogon]# yum -y install epel-release
```

安装 python-pip

```
[root@ bogon]# yum -y install python-pip
```

安装 docker-compose

```
pip install -U docker-compose
```

```
[root@bogon docker-compose]# pip install docker-compose
Collecting docker-compose
  Using cached docker-compose-1.6.2.tar.gz
Collecting cached-property<2,>=1.2.0 (from docker-compose)
  Using cached cached-property-1.3.0-py2.py3-none-any.whl
Collecting docopt<0.7,>=0.6.1 (from docker-compose)
  Using cached docopt-0.6.2.tar.gz
Requirement already satisfied (use --upgrade to upgrade): PyYAML<4,>=3.10 in /usr/lib64/python2.7/site-packages (from docker-compose)
Collecting requests<2.7.0-py2.py3-none-any.whl
  Using cached requests-2.7.0-py2.py3-none-any.whl
Collecting texttable<0.9,>=0.8.1 (from docker-compose)
  Using cached texttable-0.8.4.tar.gz
Collecting websocket-client<1.0,>=0.32.0 (from docker-compose)
  Downloading websocket-client-0.35.0.tar.gz (193kB)
  100% |#####| 196kB 604kB/s
Collecting docker-py<2,>=1.7.0 (from docker-compose)
  Using cached docker-py-1.7.2.tar.gz
Collecting dockertpy<0.5,>=0.4.1 (from docker-compose)
  Using cached dockertpy-0.4.1.tar.gz
Requirement already satisfied (use --upgrade to upgrade): six<2,>=1.3.0 in /usr/lib/python2.7/site-packages (from docker-compose)
Collecting jonschema<3,>=2.5.1 (from docker-compose)
  Downloading jonschema-2.5.1-py2.py3-none-any.whl
Requirement already satisfied (use --upgrade to upgrade): enum34<2,>=1.0.4 in /usr/lib/python2.7/site-packages (from docker-compose)
Requirement already satisfied (use --upgrade to upgrade): backports.ssl-match-hostname in /usr/lib/python2.7/site-packages (from websocket-client<1.0,>=0.32.0->docker-compose)
Collecting funtools32 (from jonschema<3,>=2.5.1->docker-compose)
  Downloading funtools32-3.2.3-2.tar.gz
Building wheels for collected packages: docker-compose, docopt, texttable, websocket-client, docker-py, dockertpy, funtools32
Running setup.py bdist_wheel for docker-compose ... done
Stored in directory: /root/.cache/pip/wheels/26/a6/b9/666219945995fabe85916c238491804b7776802945a18b92ca
Running setup.py bdist_wheel for docopt ... done
Stored in directory: /root/.cache/pip/wheels/0d/5c/a7/cb986749520c1956217b5d8405def5c18541322dbc41a80d1
Running setup.py bdist_wheel for texttable ... done
Stored in directory: /root/.cache/pip/wheels/ab/74/7f/2b7fc041492ad26d80779955f8fbc596c948f13e2dbb33bf6d
Running setup.py bdist_wheel for websocket-client ... done
Stored in directory: /root/.cache/pip/wheels/56/d3/af/529917d2f31df813fb0c5445ff88abcd8ee6e450c192b2ed21
Running setup.py bdist_wheel for docker-py ... done
Stored in directory: /root/.cache/pip/wheels/0a/26/d0/6acfa391aeb1824475d9125c88d075ae9b66e16107961f6678
Running setup.py bdist_wheel for dockertpy ... done
Stored in directory: /root/.cache/pip/wheels/05/8c/84/a3085d476576abe6b39eadd7886f14eea7f346321a98e6c3fd
Running setup.py bdist_wheel for funtools32 ... done
Stored in directory: /root/.cache/pip/wheels/30/c6/c7/eel7acd621120c302e25c2fa8b3a0b235d5d1137c6ab4c972b
Successfully built docker-compose docopt texttable websocket-client docker-py dockertpy funtools32
Installing collected packages: cached-property, docopt, requests, texttable, websocket-client, docker-py, dockertpy, funtools32, jonschema, docker-compose
Found existing installation: requests 2.6.0
DEPRECATION: Uninstalling a distutils installed project (requests) has been deprecated and will be removed in a future version. This is due to the fact that uninstalling a distutils project will only partially uninstall the project.
Uninstalling requests-2.6.0:
  Successfully uninstalled requests-2.6.0
Successfully installed cached-property-1.3.0 docker-compose-1.6.2 docker-py-1.7.2 dockertpy-0.4.1 docopt-0.6.2 funtools32-3.2.3-2 jonschema-2.5.1 requests-2.7.0 texttable-0.8.4 websocket-client-0.35.0
[root@bogon docker-compose]#
```

到这里 docker-compse 就完成了。

2. 从 Git 到 Portus 的代码搭建

正常安装方法：需要到 Git Clone <https://github.com/SUSE/Portus.git> 上获取 Portus 的源码。下载包 Portus_git.tar.gz，解压此源码包，并修改 Gemfile.lnk 的第一行：~~"https://rubygems.org"~~ 修改为 ~~"http://rubygems.org"~~。运行 compose-setup.sh -e server IP。进行构建安装。(此处省略略)

```
[root@bogon Portus]# ./compose-setup.sh -e 192.168.0.70

#####
# WARNING #
#####

This deployment method is intended for testing/development purposes.
To deploy Portus on production please take a look at: http://port.us.org/documentation.html

The setup will destroy the containers used by Portus, removing also their volumes.
Are you sure to delete all the data? (Y/N) [Y] y
No stopped containers
Pulling db (library/mariadb:10.0.23)...
10.0.23: Pulling from library/mariadb
7268d8f794c4: Pull complete
a3ed95caeb02: Pull complete
e5a99361f38c: Pull complete
20b20853e29d: Pull complete
9dbc63cf121f: Pull complete
fdebb5c64c6c: Pull complete
38152cd1ae2a: Pull complete
d7f1267eb179: Pull complete
cb5b075a9692: Pull complete
d65a44f4573e: Pull complete
Digest: sha256:508d6866998ce604ca0a9560da342baf3c67e613372c1431397b12d7d7a59c47
Status: Downloaded newer image for mariadb:10.0.23
Creating portus_db_1
Building web
Step 1 : FROM library/rails:4.2.2
4.2.2: Pulling from library/rails
df22f9f3e4ec: Downloading [====>] 4.187 MB/51.36 MB
a3ed95caeb02: Download complete
a2f74b08a06b: Downloading [=====>] 4.498 MB/18.54 MB
29b84dd39cd5: Downloading [====>] 3.429 MB/42.34 MB
a85bd624bab4: Waiting
a3ebd79fa0b0: Waiting
556000417e4a: Waiting
378ba1495d4a: Waiting
f46fbc0aa189: Waiting
ac45525c1e54: Waiting
7cfa69356b17: Waiting
7cfa69356b17: Pulling fs layer
```

由于 Portus 在安装过程中，需要下载几个依赖的镜像包，例如：Portus 安装依赖 MariaDB, portus_web.tar、rails4.2.2tar,Registry2.1.1.tar 安装过程中下载比较慢。我们先下载了再进行安装过程。

```
[root@bogon opt]# cd Portus\ Registry的搭建/
[root@bogon Portus Registry的搭建]# ls
get-pip.py mariadb10.0.23.tar Portus_git.tar.gz portus_web.tar Portus搭建.docx rails4.2.2.tar registry2.1.1.tar suse的portus的搭建过程 - 晓明.txt
[root@bogon Portus Registry的搭建]# tar -zxvf Portus_git.tar.gz
Portus/app/
Portus/app/assets/
Portus/app/assets/images/
Portus/app/assets/images/layout/
Portus/app/assets/images/layout/.portus-logo-dashboard.png
Portus/app/assets/images/layout/bg.png
Portus/app/assets/images/layout/company-panel-bg.jpg
Portus/app/assets/images/layout/portus-error.png
```

由于我先把这几个已经下载完成，所以首先拷贝到宿主机的目录中了，需要直接解压。

```
[root@bogon Portus Registry的搭建]# ls
get-pip.py mariadb10.0.23.tar Portus Portus_git.tar.gz portus_web.tar Portus搭建.docx rails4.2.2.tar registry2.1.1.tar suse的portus的搭建过程 - 晓明.txt
[root@bogon Portus Registry的搭建]# docker load < mariadb10.0.23.tar
[root@bogon Portus Registry的搭建]# docker load < portus_web.tar
[root@bogon Portus Registry的搭建]# docker load < rails4.2.2.tar
[root@bogon Portus Registry的搭建]# cd Portus/
[root@bogon Portus]# ls
app          compose      config.ru    deploy       docker-compose.yml  Gemfile.lock  log          Rakefile    tmp          vendor
bin          compose-setup.sh  CONTRIBUTING.md  doc          Dockerfile         Gemfile       packaging    README.md   vagrant     VERSION
CHANGELOG.md  config         db           docker-compose_bak.yml  Lib              LICENSE       public       spec        Vagrantfile
```

3. 安装 portus 程序

安装之前需要把拷贝到宿主机中的几个依赖镜像给 load 到 Images 中。

```
get-pip.py mariadb10.0.23.tar Portus Portus_git.tar.gz portus_web.tar Portus搭建.docx rails4.2.2.tar registry2.1.1.tar suse的portus的搭建过程 - 说明.txt
[root@bogon Portus Registry的搭建]# docker load < mariadb10.0.23.tar
[root@bogon Portus Registry的搭建]# docker load < portus_web.tar
[root@bogon Portus Registry的搭建]# docker load < rails4.2.2.tar
[root@bogon Portus Registry的搭建]# docker load < registry2.1.1.tar
[root@bogon Portus Registry的搭建]#
```

关于 Registry 的存储路径修改，请在安装前先到 compose 中修改 docker-compose.yml.template 文件。这里面包含 web 端口和挂载 volumes 等参数。然后，再执行安装脚本 ./compose-setup.sh -e 192.168.0.70。（由于本实例重新安装了 Registry，所以，此处宿主机和容器的端口都是 5000。）


```
1 192.168.4.38 x 2 192.168.0.70 x 3 192.168.0.24 x 4 192.168.0.70 x
web:
  build: .
  command: puma -b tcp://0.0.0.0:3000 -w 3
  environment:
    - PORTUS_MACHINE_FQDN_VALUE=EXTERNAL_IP
    - PORTUS_DB_HOST=portus_db_1
  volumes:
    - ../portus
  ports:
    - 3000:3000
  links:
    - db

crono:
  image: portus_web
  entrypoint: bin/crono
  environment:
    - PORTUS_MACHINE_FQDN=EXTERNAL_IP
    - PORTUS_DB_HOST=portus_db_1
  volumes:
    - ../portus
  links:
    - db

db:
  image: library/mariadb:10.0.23
  environment:
    MYSQL_ROOT_PASSWORD: portus

registry:
  image: library/registry:2.1.1
  volumes:
    - /opt/myregistry/registry_data
    - ../compose/registry/portus.crt:/etc/docker/registry/portus.crt:ro
    - ../compose/registry/config.yml:/etc/docker/registry/config.yml:ro
  ports:
    - 5000:5000
    - 5001:5001 # required to access debug service
  links:
    - web
```

然后到 Portus 源代码文件中执行./compose-setup.sh。

```

1 192.168.4.38 x 2 192.168.4.121 x 3 192.168.0.70 x 4 192.168.0.24 x 5 192.168.4.202
[root@bogon Portus]# ./compose-setup.sh -e 192.168.0.70

#####
# WARNING #
#####

This deployment method is intended for testing/development purposes.
To deploy Portus on production please take a look at: http://port.us.org/documentation.html

The setup will destroy the containers used by Portus, removing also their volumes.
Are you sure to delete all the data? (Y/N) [Y] y
Killing portus_web_1 ... done
Killing portus_db_1 ... done
Going to remove portus_registry_1, portus_web_1, portus_db_1
Removing portus_registry_1 ... done
Removing portus_web_1 ... done
Removing portus_db_1 ... done
Creating portus_db_1
Creating portus_web_1
Creating portus_registry_1
Creating portus_crono_1
Waiting for mariadb to be ready in 5 seconds
Portus: configuring database... [SUCCESS]

#####
# SUCCESS #
#####

Make sure port 3000 and 5000 are open on host 192.168.0.70

Open http://192.168.0.70:3000 with your browser and perform the following steps:

1. Create an admin account
2. You will be redirected to a page where you have to register the registry. In this form:
   - Choose a custom name for the registry.
   - Enter 192.168.0.70:5000 as the hostname.
   - Do *not* check the "Use SSL" checkbox, since this setup is not using SSL.

Perform the following actions on the docker hosts that need to interact with your registry:

- Ensure the docker daemon is started with the '--insecure-registry 192.168.0.70:5000'
- Perform the docker login.

To authenticate against your registry using the docker cli do:

$ docker login -u <portus username> -p <password> -e <email> 192.168.0.70:5000

To push an image to the private registry:

$ docker pull busybox
$ docker tag busybox 192.168.0.70:5000/<username>busybox
$ docker push 192.168.0.70:5000/<username>busybox

```

到此，Portus 的安装已经完成。

4.修改 Docker 配置文件

此时，我们还需要修改 `vi /lib/systemd/system/docker.service` 把 `--insecure-registry 192.168.0.70:5000` 这句加到 `ExecStart=/usr/bin/docker` 这个配置项后


```
[Unit]
Description=Docker Application Container Engine
Documentation=http://docs.docker.com
After=network.target
Wants=docker-storage-setup.service

[Service]
Type=notify
EnvironmentFile=-/etc/sysconfig/docker
EnvironmentFile=-/etc/sysconfig/docker-storage
EnvironmentFile=-/etc/sysconfig/docker-network
Environment=GOTRACEBACK=crash
ExecStart=/usr/bin/docker daemon --insecure-registry 192.168.0.70:5000 $OPTIONS \
    $DOCKER_STORAGE_OPTIONS \
    $DOCKER_NETWORK_OPTIONS \
    $ADD_REGISTRY \
    $BLOCK_REGISTRY \
    $INSECURE_REGISTRY
LimitNOFILE=1048576
LimitNPROC=1048576
LimitCORE=infinity
MountFlags=slave
TimeoutStartSec=1min
Restart=on-failure

[Install]
WantedBy=multi-user.target
~
~
~
```

重启 Docker

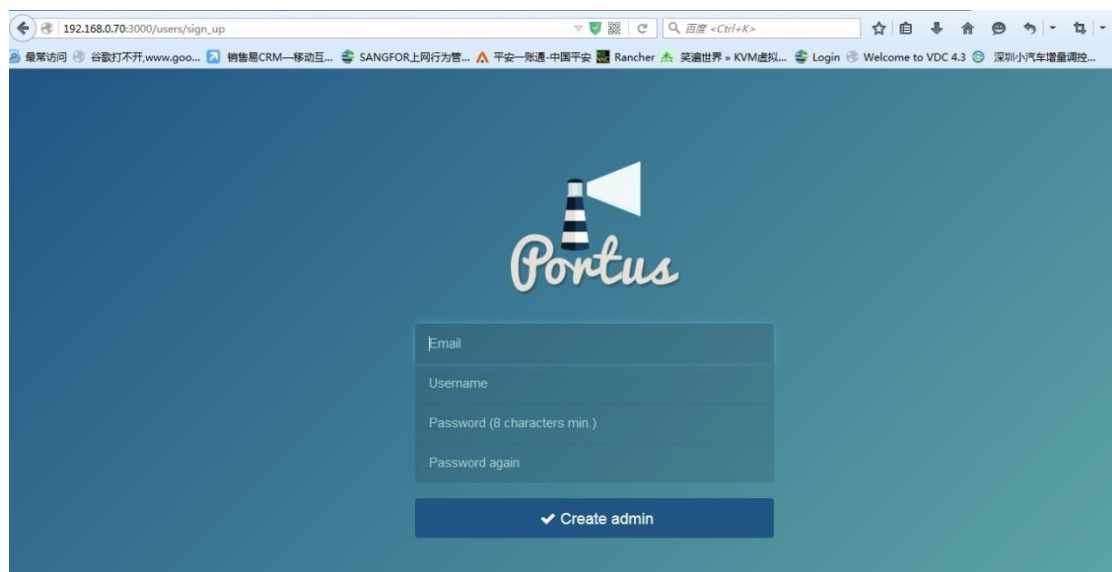
```
systemctl daemon-reload
```

```
systemctl restart docker
```

重启 portus 容器

```
docker start portus_db_1 portus_web_1 portus_crono_1 portus_registry_1
```

可以在浏览器中打开登录窗口。



在客户机中需要修改 /etc/sysconfig/docker 下的文件。



```
# /etc/sysconfig/docker

# Modify these options if you want to change the way the docker daemon runs
#OPTIONS='--selinux-enabled'
OPTIONS='--selinux-enabled --insecure-registry 192.168.0.70:5000'

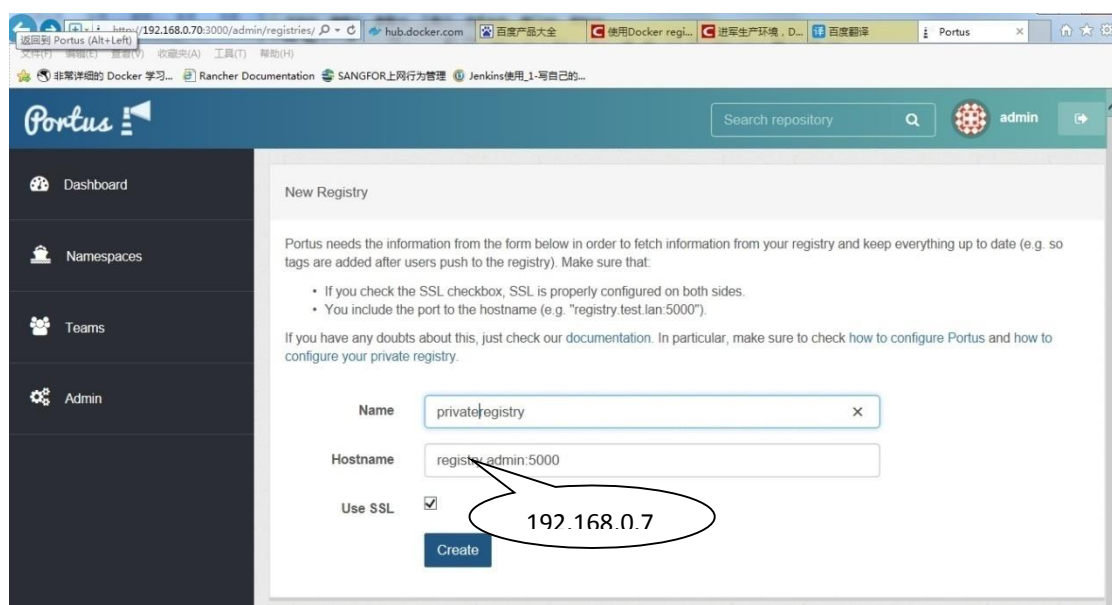
DOCKER_CERT_PATH=/etc/docker

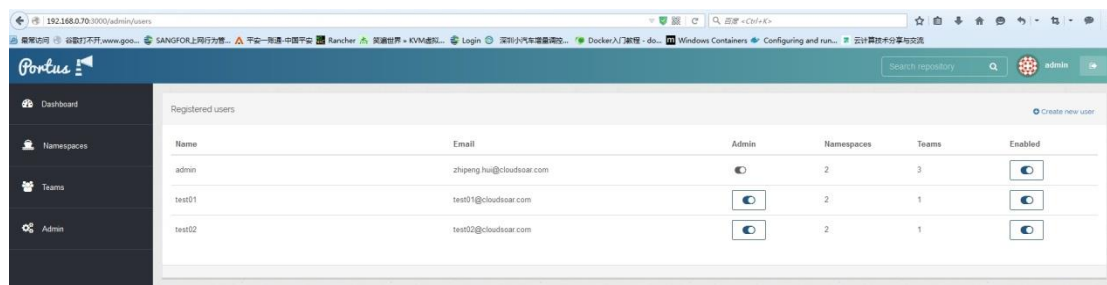
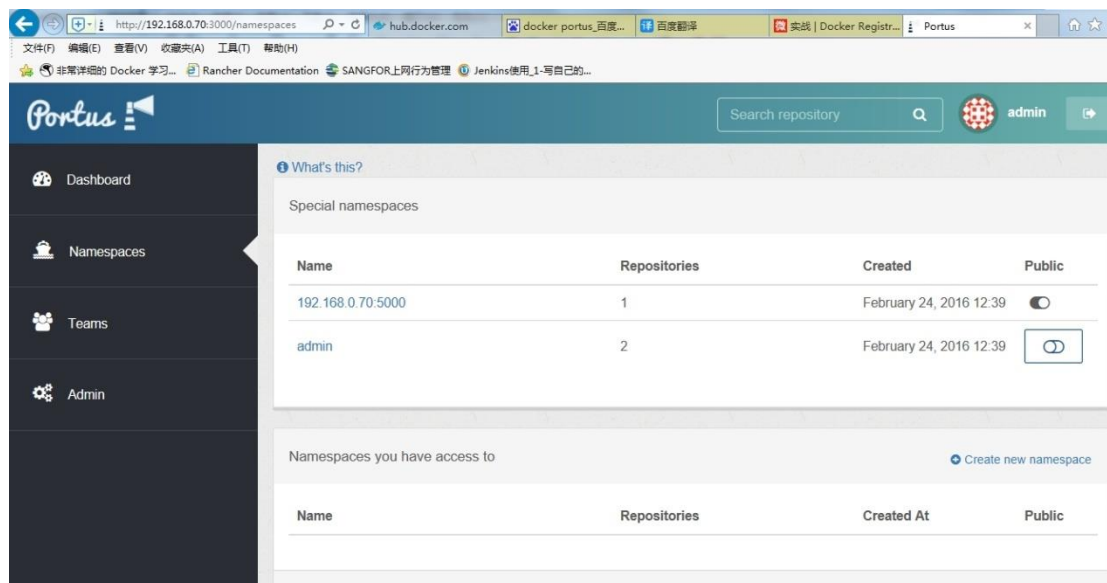
# If you want to add your own registry to be used for docker search and docker
# pull use the ADD_REGISTRY option to list a set of registries, each prepended
# with --add-registry flag. The first registry added will be the first registry
# searched.
#ADD_REGISTRY='--add-registry registry.access.redhat.com'
```

二 . Portus 配置与验证

1 .登录配置程序

Portus 的登录界面需要创建用户名和密码。然后进行 Registry 的设置。



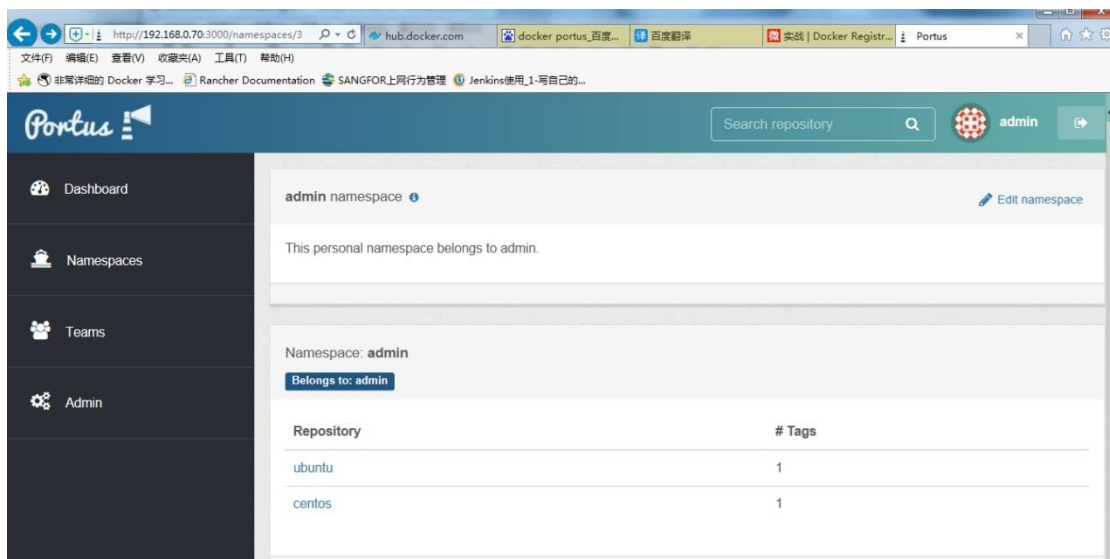


创建 test01 和 test02 帐户，并 Enabled 帐户，然后进行登录测试。

2. 上传镜像测试

```
[root@bogon ~]# docker tag centos 192.168.0.70:5000/admin/centos
[root@bogon ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
rancher/server      latest             7c0ed1de7188       3 weeks ago        845.3 MB
192.168.0.70:5000/admin/centos latest             9fd593454178       8 weeks ago        196.6 MB
centos               latest             9fd593454178       8 weeks ago        196.6 MB
192.168.0.70:5000/admin/ubuntu latest             d59bdb51bb5c       11 weeks ago       187.9 MB
192.168.0.70:5010/admin/ubuntu latest             d59bdb51bb5c       11 weeks ago       187.9 MB
192.168.0.70:80/ubuntu latest             d59bdb51bb5c       11 weeks ago       187.9 MB
ubuntu               latest             d59bdb51bb5c       11 weeks ago       187.9 MB
[root@bogon ~]# docker push 192.168.0.70:5000/admin/centos
The push refers to a repository [192.168.0.70:5000/admin/centos]
5f70bf18a086: Pushed
ee1dd2cb6df2: Pushed
latest: digest: sha256:dc15a34cab6ec6d386c39596dc82d3dc01d9003b5e4f49f5deac6f74e9473646 size: 3420
[root@bogon ~]#
```

```
[root@bogon ~]# docker login 192.168.0.70:5000
Username: admin
Password:
Email:
WARNING: login credentials saved in /root/.docker/config.json
Login Succeeded
[root@bogon ~]# docker push 192.168.0.70:5000/admin/ubuntu
The push refers to a repository [192.168.0.70:5000/admin/ubuntu]
5f70bf18a086: Pushed
4592214fdcf9: Pushed
3f9e224fca30: Pushed
e12b76467769: Pushed
latest: digest: sha256:4e769f546a9b5a245974379665ea2f88e93db13af3b2d0dc6adea2cec04f219a size: 4335
[root@bogon ~]#
```



3. 下载镜像测试

```
[root@bogon ~]# docker pull 192.168.0.70:5000/admin/centos
Using default tag: latest
Trying to pull repository 192.168.0.70:5000/admin/centos ... latest: Pulling from admin/centos
3690474eb5b4: Pull complete
6f16b97dbf97: Extracting [=====] 70.51 MB/70.51 MB
0d1bea7a25ee: Download complete
e2c799a75582: Download complete
e2c799a75582: Pulling fs layer
```

● 总结

Docker Registry 的创建私有仓库的方法有很多种。像京东的 Docker 镜像存储系统-Speedy，Registry+Nginx &SSL 等后续将进一步探究，实现开发、测试以及生产的一体化流程，敬请期待！

本文电子书下载：

云盘下载：<http://pan.baidu.com/s/1jHfP9Dc>

网页下载：<http://www.cloudsoar.com/down/ddoc/v1.1/>

博客期刊：<http://www.cloudsoar.com/about/BlogNews/content/0/133/v1.1/>

温馨提示：

云舒网络携手 Rancher Labs 推出【Rancher | 实战微信群】，在线为您分享 Docker 技术干货，更有往期回顾精选期刊等你拿！

本群汇集了 Rancher 中国最强技术精英团队及业内技术派高人，宗旨是为了大家拥有更专业的平台交流 Rancher 实战技术，实时与 Rancher 创始团队面对面！同时欢迎各位分享自己的经验、疑难问题，我们将定期邀请分享嘉宾做各类话题分享及回顾，共同实践研究 Docker 容器生态圈。

对 Rancher 和 Docker 技术感兴趣、或对本文中细节需继续探讨的朋友，欢迎加入本群参与讨论！

加微信群方法：

- 1.关注【云舒网络】公众号
- 2.留言“我要加群”

QQ 群号：216521218

