

# Nexus 3 Docker倉庫（hosted、proxy、group）模式解說

出處：<https://www.cnblogs.com/yjbjingcha/p/8308973.html>

Nexus作為私庫管理最受歡迎的工具之一，用於套件的管理和Docker映像管理的私庫管理場景中非經常用。Easypack利用最新版本號的oss版Nexus作為基礎鏡像來提供相似服務。

本文將同一時候給出具體步驟結合最新發行的Docker-CE版本號實現鏡像私庫的管理。

## Why Nexus 3

這裡整理了為什麼使用Nexus 3的一些理由，在做選型的時候能夠做一個簡單參照。

專案	具體
為什麼要使用Nexus 3	<a href="http://blog.csdn.net/liumiaocn/article/details/62050525">http://blog.csdn.net/liumiaocn/article/details/62050525</a>

## docker版本號

本次使用的docker的版本號為17.03.0-ce，可是應該不限於此版本號，其它版本號未作驗證。

```
[root@liumiaocn ~]# docker version
Client:
Version:      17.03.0-ce
API version:  1.26
Go version:   go1.7.5
Git commit:   3a232c8
Built:        Tue Feb 28 08:10:07 2017
OS/Arch:      linux/amd64

Server:
Version:      17.03.0-ce
API version:  1.26 (minimum version 1.12)
Go version:   go1.7.5
Git commit:   3a232c8
Built:        Tue Feb 28 08:10:07 2017
OS/Arch:      linux/amd64
Experimental: false
[root@liumiaocn ~]#
```

## 下載鏡像

```
[root@liumiaocn ~]# docker pull liumiaocn/nexus
Using default tag: latest
latest: Pulling from liumiaocn/nexus
Digest: sha256:b93f9a6bba2b35ada33c324cd06bd2c732fc1bed352df186af1a013e228af8d8
Status: Image is up to date for liumiaocn/nexus:latest
[root@liumiaocn ~]#
```

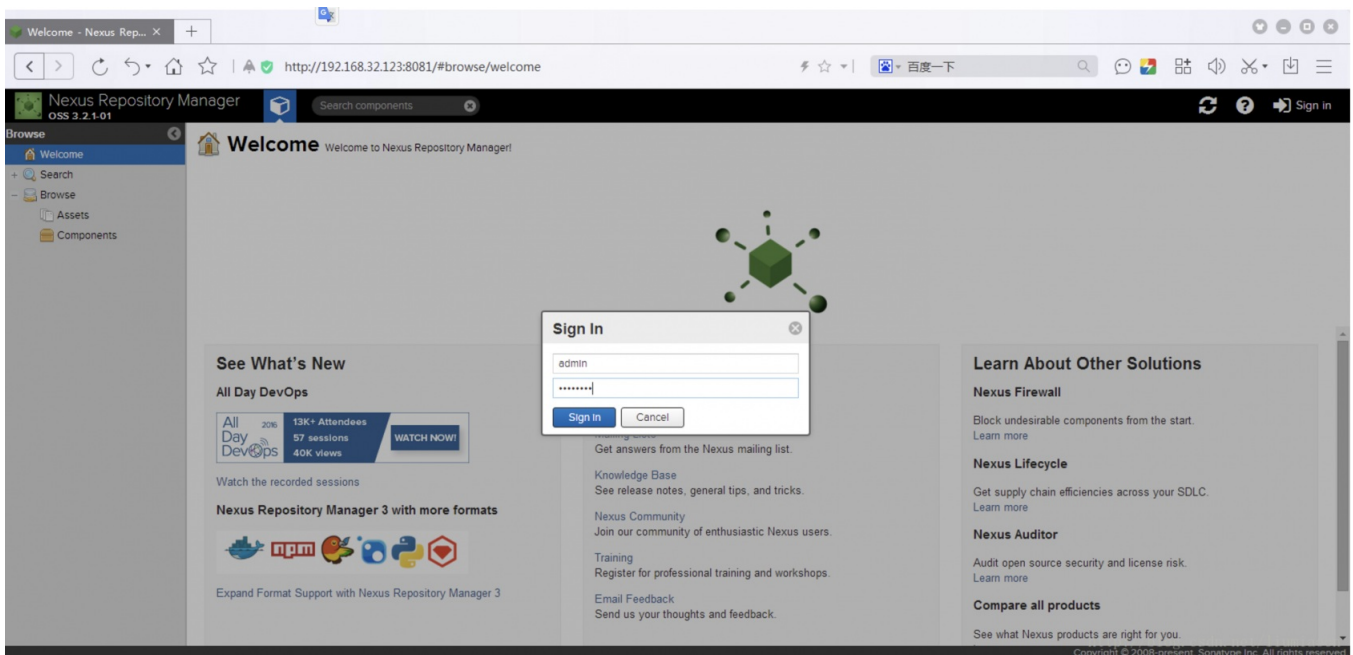
## 啟動Nexus

專案	具體
Nexus UI	8081
private repo	8082
proxy repo	8083
URL	<a href="http://192.168.32.123:8081/">http://192.168.32.123:8081/</a>

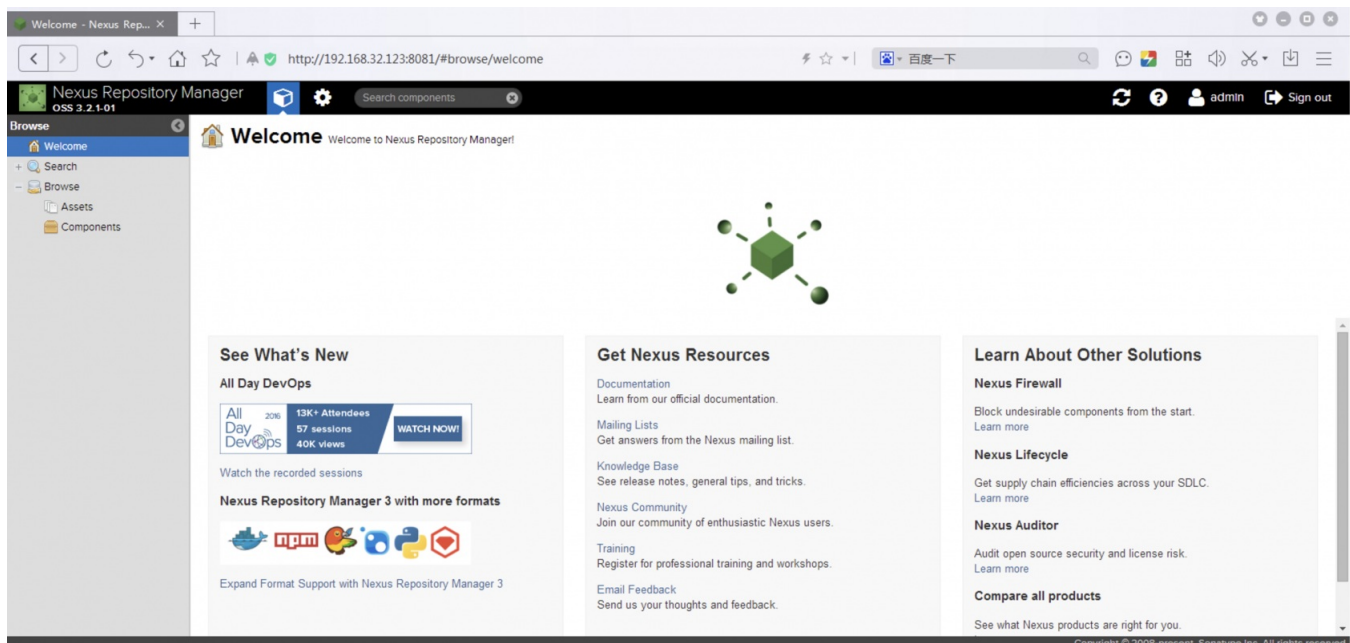
```
[root@liumiaocn ~]# docker run -d -p 8081:8081 -p 8082:8082 -p 8083:8083 --name nexus liumiaocn/nexus
222abae47fc9d32c821bff6426edd03f6757a3dd4cbe07517dada5d800e173f
[root@liumiaocn ~]#
```

# logon

專案	具體
URL	<a href="http://192.168.32.123:8081/">http://192.168.32.123:8081/</a>
username稱	admin
使用者password	admin123



登陸後



# 倉庫類型

具體倉庫類型主要分為hosted/proxy/group三種。

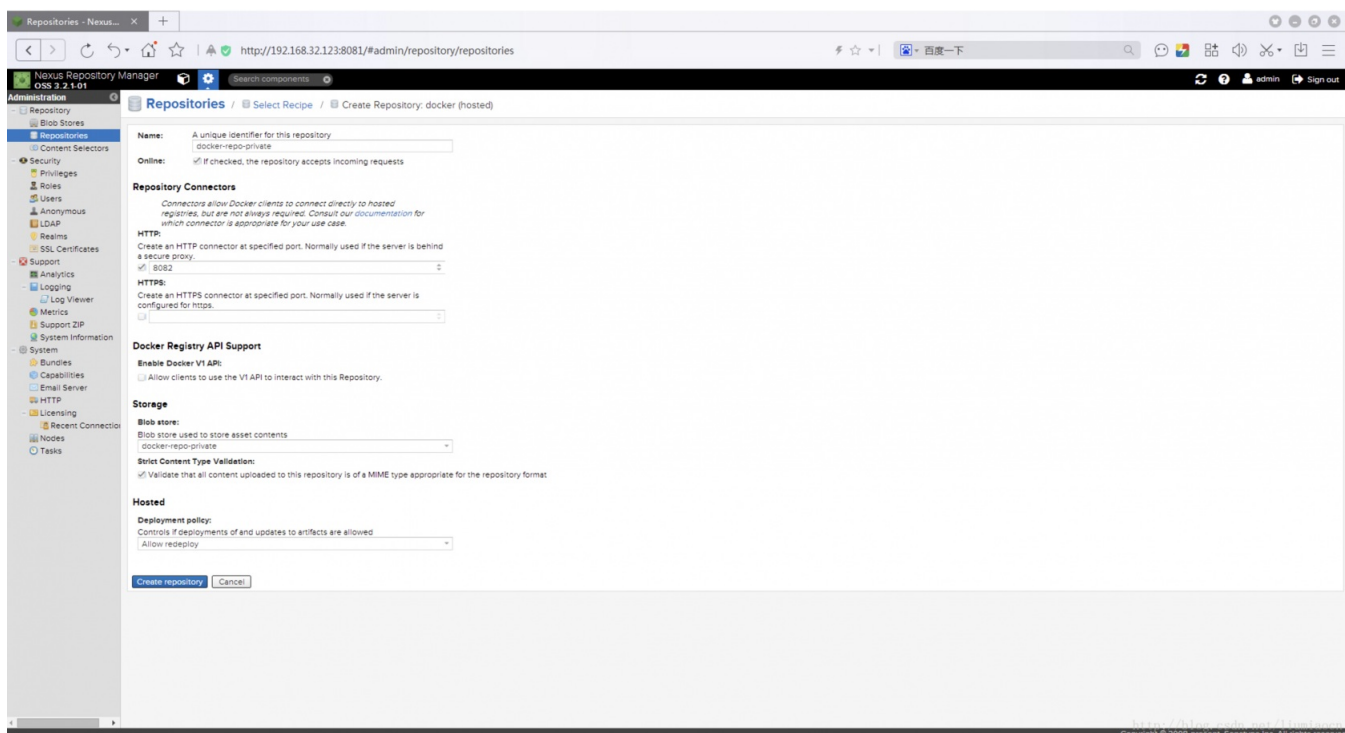
具體含義例如以下：

專案	具體說明
hosted	本地儲存。像官方倉庫一樣提供本地私庫功能
proxy	提供代理其它倉庫的類型
group	群組類型，能夠組合多個倉庫為一個位址提供服務

# 建立private倉庫

建立一個Hosted的倉庫作為private倉庫，具體設定資訊例如以下：

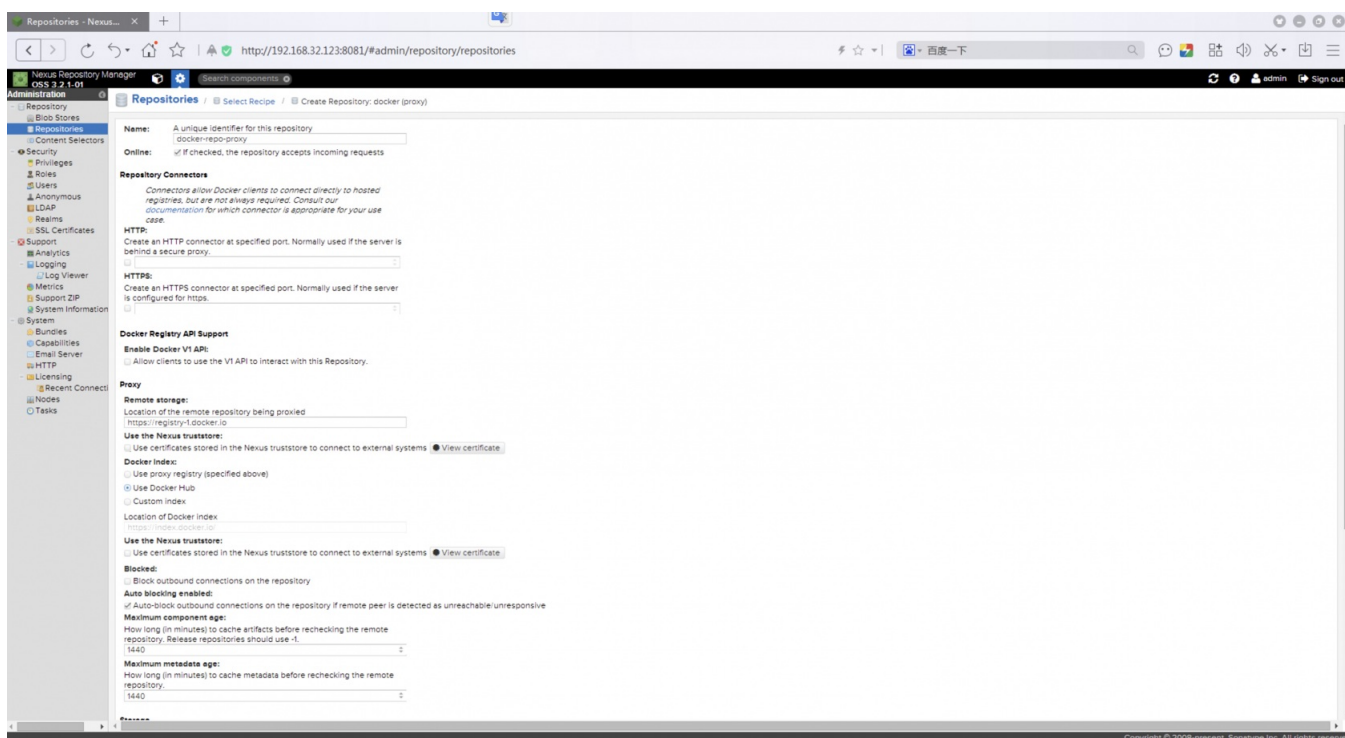
專案	具體說明
類型	docker (hosted)
Name	docker-repo-private
HTTP Port	8082
Blob store	docker-repo-private
Deployment policy	Allow redeploy



# 建立proxy倉庫

建立一個proxy倉庫。具體設定資訊例如以下：

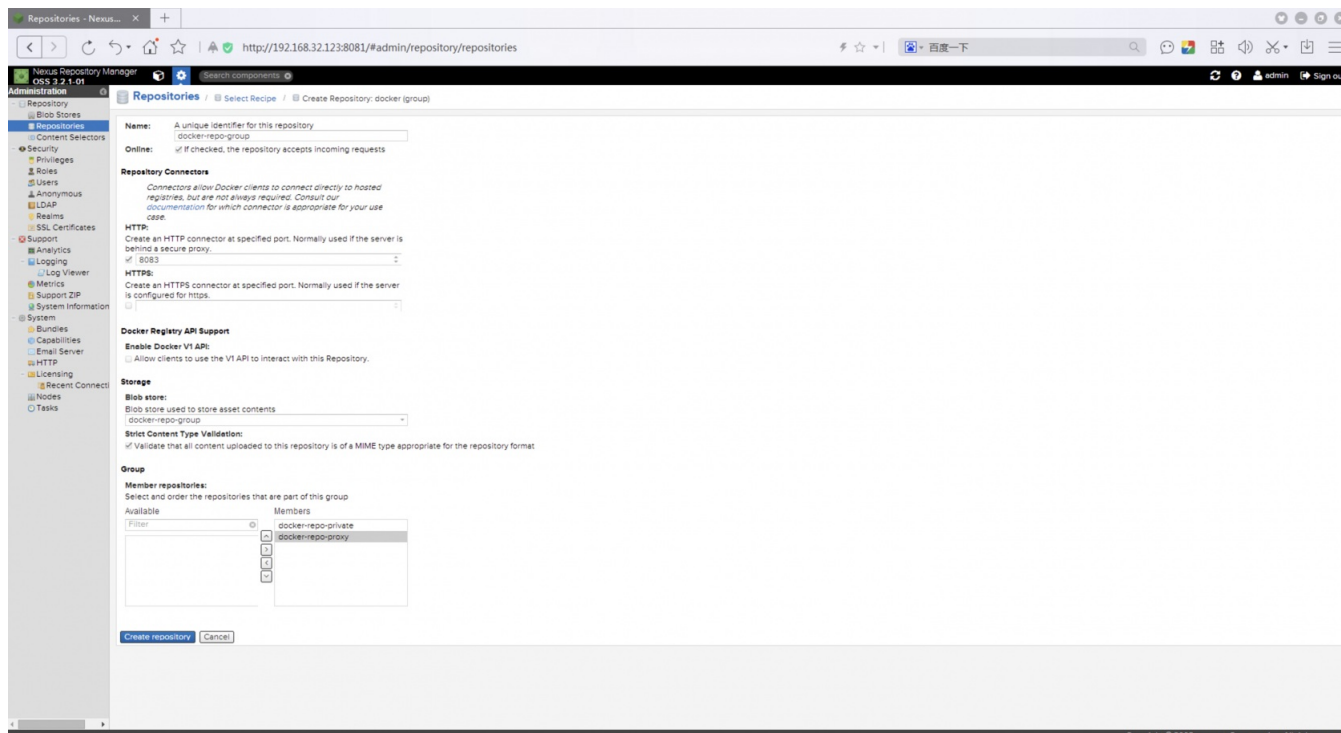
專案	具體說明
類型	docker (proxy)
Name	docker-repo-proxy
Location of the remote repository being proxied	<a href="https://registry-1.docker.io">https://registry-1.docker.io</a>
Docker Index	Use Docker Hub
Blob store	docker-repo-proxy



# 建立group倉庫

建立一個group倉庫。具體設定資訊例如以下：

專案	具體說明
類型	docker (group)
Name	docker-repo-group
HTTP Port	8083
Blob store	docker-repo-group
Member repositories	docker-repo-private
Member repositories	docker-repo-proxy



## docker設定

Docker的私庫能夠使用HTTP或HTTPS。Nexus 3都予以支持。本文的方式採用HTTP方式。因此須要設定docker。在docker啟動前設定例如以下資訊是須要的

專案	具體說明
設定物件文件	/etc/docker/daemon.json
設定內容	insecure-registries

設定具體內容。例如以下所看到的：

```
[root@liumiaocn ~]# cat /etc/docker/daemon.json
{
  "insecure-registries": [
    "192.168.32.123:8082",
    "192.168.32.123:8083"
  ],
  "disable-legacy-registry": true
}
[root@liumiaocn ~]#
```

# 重新啟動docker

```
[root@liumiaocn docker]# systemctl restart docker
[root@liumiaocn docker]#
```

## 啟動nexus

隨著docker的重新啟動。nexus的容器也須要啟動。具體例如以下所看到的：

```
[root@liumiaocn docker]# docker start nexus
nexus
[root@liumiaocn docker]#
```

## 確認

至此，Nexus的設定準備基本就緒。能夠確認結果了。

## docker login

為了進行操作。須事先進行docker login的操作

專案	具體說明
private倉庫	docker login -u admin -p admin123 192.168.32.123:8082
proxy倉庫	docker login -u admin -p admin123 192.168.32.123:8083

運行確認

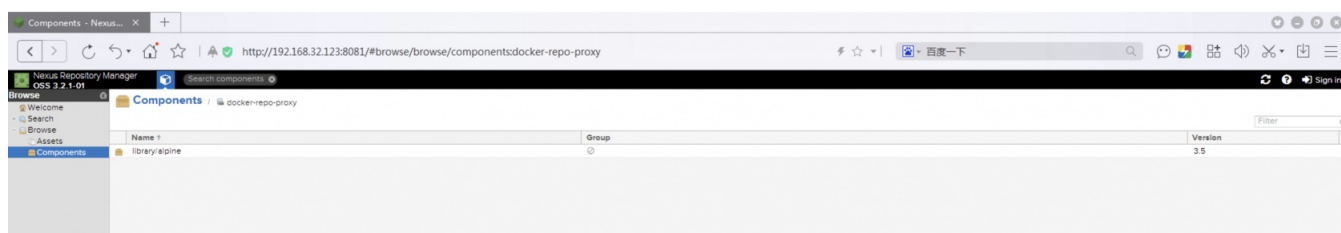
```
[root@liumiaocn ~]# docker login -u admin -p admin123 192.168.32.123:8082
Login Succeeded
[root@liumiaocn ~]# docker login -u admin -p admin123 192.168.32.123:8083
Login Succeeded
[root@liumiaocn ~]#
```

## proxy倉庫確認

從remote倉庫pull下來鏡像。然後確認是否在proxy倉庫中存在

```
[root@liumiaocn ~]# docker pull 192.168.32.123:8083/alpine:3.5
3.5: Pulling from alpine
627beaf3eaaf: Downloading
unknown blob
[root@liumiaocn docker]#
```

查了一下nexus的一些issue，發現有不少都是跟blob相關，看起來相關的小的問題另一些在不斷的收拾中，可是不影響結果。確認proxy倉庫，發現pull的alpine的3.5版本號已然在proxy倉庫中保存完成，所以上面的unknown blob也確實沒有影響結果。



# private倉庫確認

## 事前鏡像確認

```
[root@liumiaocn ~]# docker images
REPOSITORY          TAG             IMAGE ID         CREATED          SIZE
busybox              latest          00f017a8c2a6    2 days ago      1.11 MB
liumiaocn/maven      latest          833b66f10ce6    5 days ago      160 MB
liumiaocn/nexus       latest          932d715eb7e1    5 days ago      460 MB
liumiaocn/gitlab      latest          2462fb291203    5 days ago      1.21 GB
liumiaocn/jenkins     latest          6668ecd39e4f    5 days ago      293 MB
[root@liumiaocn ~]#
```

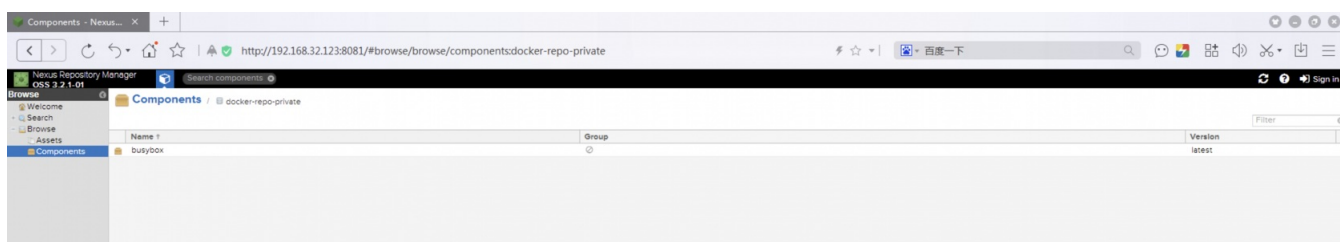
## tag busybox鏡像

```
[root@liumiaocn ~]# docker tag busybox 192.168.32.123:8082/busybox:latest
[root@liumiaocn ~]# docker images
REPOSITORY          TAG             IMAGE ID         CREATED          SIZE
192.168.32.123:8082/busybox latest          00f017a8c2a6    2 days ago      1.11 MB
busybox              latest          00f017a8c2a6    2 days ago      1.11 MB
liumiaocn/maven      latest          833b66f10ce6    5 days ago      160 MB
liumiaocn/nexus       latest          932d715eb7e1    5 days ago      460 MB
liumiaocn/gitlab      latest          2462fb291203    5 days ago      1.21 GB
liumiaocn/jenkins     latest          6668ecd39e4f    5 days ago      293 MB
[root@liumiaocn ~]#
```

## docker push

```
[root@liumiaocn ~]# docker push 192.168.32.123:8082/busybox:latest
The push refers to a repository [192.168.32.123:8082/busybox]
c0de73ac9968: Pushed
latest: digest: sha256:68effe31a4ae8312e47f54bec52d1fc925908009ce7e6f734e1b54a4169081c5 size: 527
[root@liumiaocn ~]#
```

結果確認：busybox鏡像已經被正常地push到了private倉庫中



## docker pull

為了確認docker pull的動作。事前先將先前的busybox先刪除，以確認確實下載了新的busybox鏡像到本地。

```
[root@liumiaocn ~]# docker rmi busybox
Untagged: busybox:latest
Untagged: busybox@sha256:32f093055929dbc23dec4d03e09dfe971f5973a9ca5cf059cbfb644c206aa83f
[root@liumiaocn ~]# docker rmi 192.168.32.123:8082/busybox
Untagged: 192.168.32.123:8082/busybox:latest
Untagged: 192.168.32.123:8082/busybox@sha256:68effe31a4ae8312e47f54bec52d1fc925908009ce7e6f734e1b54a4169081c5
Deleted: sha256:00f017a8c2a6e1fe2ffd05c281f27d069d2a99323a8cd514dd35f228ba26d2ff
Deleted: sha256:c0de73ac99683640bc8f8de5cda9e0e2fc97fe53d78c9fd60ea69b31303efbc9
[root@liumiaocn ~]# docker images
REPOSITORY          TAG             IMAGE ID         CREATED          SIZE
liumiaocn/maven      latest          833b66f10ce6    5 days ago      160 MB
liumiaocn/nexus       latest          932d715eb7e1    5 days ago      460 MB
```

liumiaocn/gitlab	latest	2462fb291203	5 days ago	1.21 GB
liumiaocn/jenkins	latest	6668ecd39e4f	5 days ago	293 MB
[root@liumiaocn ~]#				

docker pull操作:能夠看到確實是從private倉庫下載下來了鏡像，並且速度也明顯快了非常之多。

```
[root@liumiaocn ~]# docker pull 192.168.32.123:8082/busybox
Using default tag: latest
latest: Pulling from busybox
04176c8b224a: Pull complete
Digest: sha256:68effe31a4ae8312e47f54bec52d1fc925908009ce7e6f734e1b54a4169081c5
Status: Downloaded newer image for 192.168.32.123:8082/busybox:latest
[root@liumiaocn ~]# docker images
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
192.168.32.123:8082/busybox  latest      00f017a8c2a6     2 days ago      1.11 MB
liumiaocn/maven      latest      833b66f10ce6     5 days ago      160 MB
liumiaocn/nexus      latest      932d715eb7e1     5 days ago      460 MB
liumiaocn/gitlab     latest      2462fb291203     5 days ago      1.21 GB
liumiaocn/jenkins    latest      6668ecd39e4f     5 days ago      293 MB
[root@liumiaocn ~]#
```

# 總結

至此，如何使用Nexus管理Docker映像已然十分清楚，可是怎樣非常好的管理鏡像，備份機制/版本號控制/差分管理/定期cache/及時清理等方面還有非常多問題須要考慮。須要在實際的工作中不斷實踐總結。