

عنوان مستند:

تنظیم Nexus به عنوان Docker Repo

نسخه 0.9

اسفند 1397

فهرست مطالب

1.....	مقدمه	1
1.....	استفاده از Nexus به عنوان Proxy Repository به Docker-Hub	2
5.....	استفاده از Nexus به عنوان Docker Repository	3
7.....	تست عملکرد Repository های ایجاد شده	4
7.....	تنظیم Docker Client برای دسترسی به Nexus Repository	4.1
10.....	لاگین کردن به Repository های ساخته شده	4.2
10.....	Pull کردن Image از طریق پروکسی ساخته شده	4.3
12.....	Push کردن Image به Host Repository ساخته شده در Nexus	4.4
12.....	Pull کردن Image از Host repository	4.5

1 مقدمه

ابزار Nexus یکی از معروفترین ابزارهای Repository Management است که در چرخه توسعه نرم افزار بکار گرفته می شود. همانگونه که میدانیم برای کار کردن با تکنولوژی داکر به یک Repository نیاز است که می توان به صورت Local آن را تعریف نموده و یا از Docker-Hub استفاده کرد که یک Repository عمومی است. همچنین Image رسمی مربوط به ابزارهای مختلف که در توسعه نرم افزار نیاز است در Repository رسمی داکر (Docker-Hub) قرار دارد بنابراین به هریک از این Image ها که نیاز باشد باید آن را از Docker-Hub دانلود کرد. در یک پروژه نرم افزاری که توسعه، ساخت، تست و عملیات بر روی چندین سیستم مختلف انجام می شود جهت صرفه جویی در مصرف پهنای باند و بهبود سرعت و فرآیند توسعه می توان از ابزار Nexus به دو صورت زیر استفاده کرد:

1. استفاده از Nexus به عنوان Proxy Repository به Docker-Hub

2. استفاده از Nexus به عنوان Docker Repository

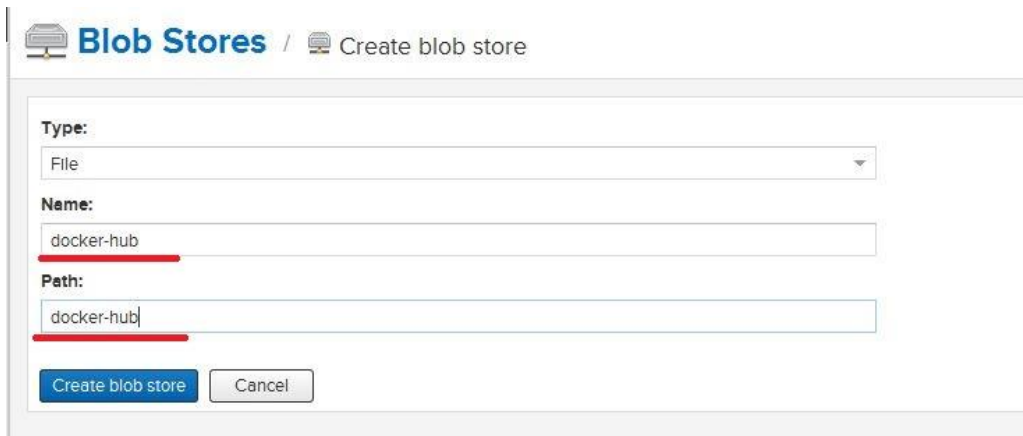
در ادامه هر کدام از این رویکردها و نحوه پیاده سازی آنها توضیح داده می شود.

2 استفاده از Nexus به عنوان Proxy Repository به Docker-Hub

همانگونه که گفته شد در یک پروژه نرم افزاری به ابزارهای پیش نیازی برای توسعه نیاز است، این Dependency ها را می توان به عنوان Docker Image از Docker-Hub دانلود کرد و استفاده نمود، جهت جلوگیری از دانلود مجدد Image هایی که قبلا دانلود شده اند می توان آنها را در یک Repository ذخیره کرد و در مراحل مختلف تولید نرم افزار از آن ها استفاده کرد. ابزار Nexus به خوبی وظیفه Repository Management را برای ما انجام می دهد و

می توان به عنوان یک Proxy Repository از آن استفاده کرد. در این حالت زمانی که درخواستی برای دانلود یک Dependency به Nexus می رسد، ابتدا Local Repository خود را بررسی می نماید و اگر موجود باشد دیگر نیازی به دانلود نیست و Dependency را از Local Repository خود در اختیار درخواست کننده قرار می دهد و در صورتی که Dependency مربوطه در Local Repository نباشد آن را دانلود می نماید و برای درخواست های بعدی نیز Dependency را Cache می کند بنابراین این ابزار با مدیریت Repository ها در صرفه جویی پهنای باند و سرعت توسعه، نقش مهمی را ایفا می کند. با توجه به توضیحاتی که داده شد می توانیم از Nexus به عنوان Docker Proxy استفاده نماییم تا نقش یک Proxy به Docker-Hub را داشته باشد و Image هایی که از آن دانلود می شوند را مدیریت نماید تا از دانلود مجدد Image ها جلوگیری شود. در ادامه نحوه انجام این کار را توضیح می دهیم:

ابتدا به عنوان Admin به سرور Nexus لاگین نموده و سپس یک Blob Store برای ذخیره Image های دانلود شده ایجاد می نماییم که در شکل 1 زیر قابل مشاهده است.



شکل 1 ایجاد Blob Store

سپس یک Repository از نوع Docker (Proxy) ایجاد می نماییم و تنظیمات را مطابق شکل های 1-2 تا 4-2 انجام می دهیم:

Repositories

Select Recipe

Create Repository: docker (proxy)

Name:

A unique identifier for this repository

docker-hub

Online:

☒ If checked, the repository accepts incoming requests

Repository Connectors

Connectors allow Docker clients to connect directly to hosted registries, but are not always required. Consult our [documentation](#) for which connector is appropriate for your use case.

HTTP:

Create an HTTP connector at specified port. Normally used if the server is behind a secure proxy.

☒ 8082

HTTPS:

Create an HTTPS connector at specified port. Normally used if the server is configured for https.

☐

Force basic authentication:

☒ Disable to allow anonymous pull (Note: also requires Docker Bearer Token Realm to be activated)

Docker Registry API Support

Enable Docker V1 API:

☐ Allow clients to use the V1 API to interact with this Repository

Proxy

شکل 1-2 ایجاد Proxy Repository

Proxy

Remote storage:

Location of the remote repository being proxied

https://registry-1.docker.io

Use the Nexus truststore:

☐ Use certificates stored in the Nexus truststore to connect to external systems

View certificate

Docker Index:

☐ Use proxy registry (specified above)
 ☒ Use Docker Hub
 ☐ Custom Index

Location of Docker index

https://index.docker.io/

Use the Nexus truststore:

☐ Use

Blocked:

☐ Block outbound connections on the repository

Auto blocking enabled:

☒ Auto-block outbound connections on the repository if remote peer is detected as unreachable/unresponsive

Maximum component age:

شکل 2-2 ایجاد Proxy Repository

Maximum component age:
How long (in minutes) to cache artifacts before rechecking the remote repository. Release repositories should use -1.

1440

Maximum metadata age:
How long (in minutes) to cache metadata before rechecking the remote repository.

1440

Storage

Blob store:
Blob store used to store asset contents

docker-hub

Strict Content Type Validation:

☒ Validate that all content uploaded to this repository is of a MIME type appropriate for the repository format

Negative Cache

Not found cache enabled:

☒ Cache responses for content not present in the proxied repository

Not found cache TTL:
How long to cache the fact that a file was not found in the repository (in minutes)

1440

شکل 3-2 ایجاد Proxy Repository

Negative Cache

Not found cache enabled:

☒ Cache responses for content not present in the proxied repository

Not found cache TTL:
How long to cache the fact that a file was not found in the repository (in minutes)

1440

HTTP

☐ Authentication

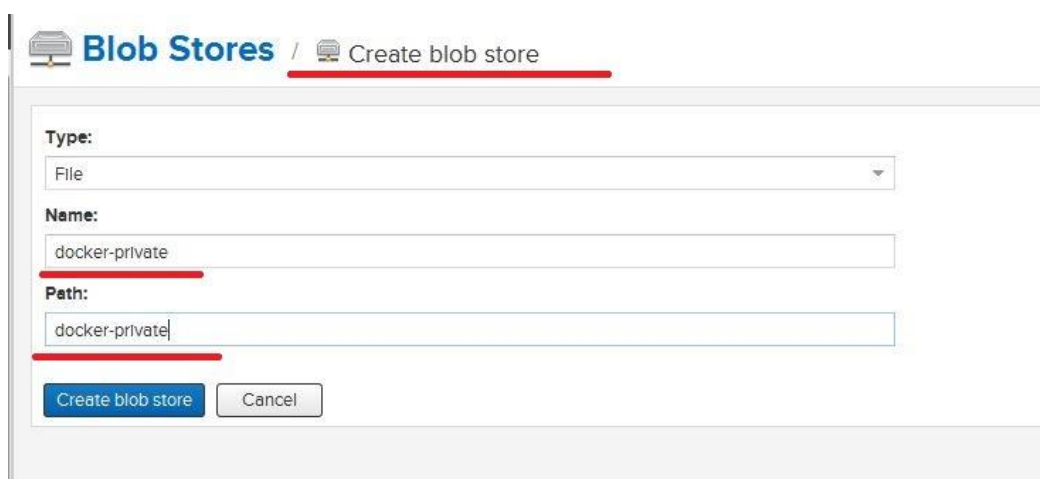
☐ HTTP request settings

Create repository Cancel

شکل 4-2 ایجاد Proxy Repository

3 استفاده از *Nexus* به عنوان *Docker Repository*

برای ذخیره Image های Docker یا باید از Repository عمومی Docker-Hub استفاده کرد و یا از Repository اختصاصی و Local استفاده نمود. از ابزار Nexus می توانیم به عنوان یک Repository اختصاصی جهت ذخیره Image های اپلیکیشن های توسعه داده شده استفاده کرد تا آن ها را مدیریت کرد و به راحتی آن ها را از سیستم های مختلف درون شبکه و در مراحل مختلف بازیابی نمود. در این راستا از Nexus به عنوان یک Host Repository استفاده می نماییم. ابتدا یک Blob Store برای ذخیره Image ها ایجاد می نماییم که در شکل 3 نشان داده شده است:



شکل 3 ایجاد Blob Store

سپس یک Repository از نوع Docker (Hosted) ایجاد نموده و تنظیمات آن را مطابق شکل های 1-4 و 2-4 انجام می دهیم:

Repositories

Select Recipe

Create Repository: docker (hosted)

Name:

A unique identifier for this repository

docker-private

Online:

☒ If checked, the repository accepts incoming requests

Repository Connectors

Connectors allow Docker clients to connect directly to hosted registries, but are not always required. Consult our [documentation](#) for which connector is appropriate for your use case.

HTTP:

Create an HTTP connector at specified port. Normally used if the server is behind a secure proxy.

☒ 8083

HTTPS:

Create an HTTPS connector at specified port. Normally used if the server is configured for https.

☐

Force basic authentication:

☒ Disable to allow anonymous pull (Note: also requires Docker Bearer Token Realm to be activated)

Docker Registry API Support

Enable Docker V1 API:

☒ Allow clients to use the V1 API to interact with this Repository

Storage

شكل 1-4 ايجاد Host Repository

Storage

Blob store:

Blob store used to store asset contents

docker-private

Strict Content Type Validation:

☒ Validate that all content uploaded to this repository is of a MIME type appropriate for the repository format

Hosted

Deployment policy:

Controls if deployments of and updates to artifacts are allowed

Allow redeploy

Create repository

Cancel

شكل 2-4 ايجاد Host Repository

4 تست عملکرد *Repository* های ایجاد شده

در قسمت های قبلی *Repository* های مورد نیاز را ساختیم، حال می خواهیم از طریق یک *Client* که داکر بر روی آن نصب است و در شبکه سرور *Nexus* قرار دارد، آن ها را تست نموده تا از عملکرد صحیحشان مطمئن شویم.

4.1 تنظیم *Docker Client* برای دسترسی به *Nexus Repository*

ارتباط *Docker Client* بصورت پیش فرض از طریق پروتکل *HTTPS* است، اما ارتباط سرور *Nexus* ما از طریق پروتکل *HTTP* است. بنابراین باید در *Daemon* داکر سیستم مربوطه، پارامتر *insecure-registry* را اضافه نماییم. با توجه به سیستم عاملی که داکر بر روی آن کار می کند، انجام این کار متفاوت خواهد بود.

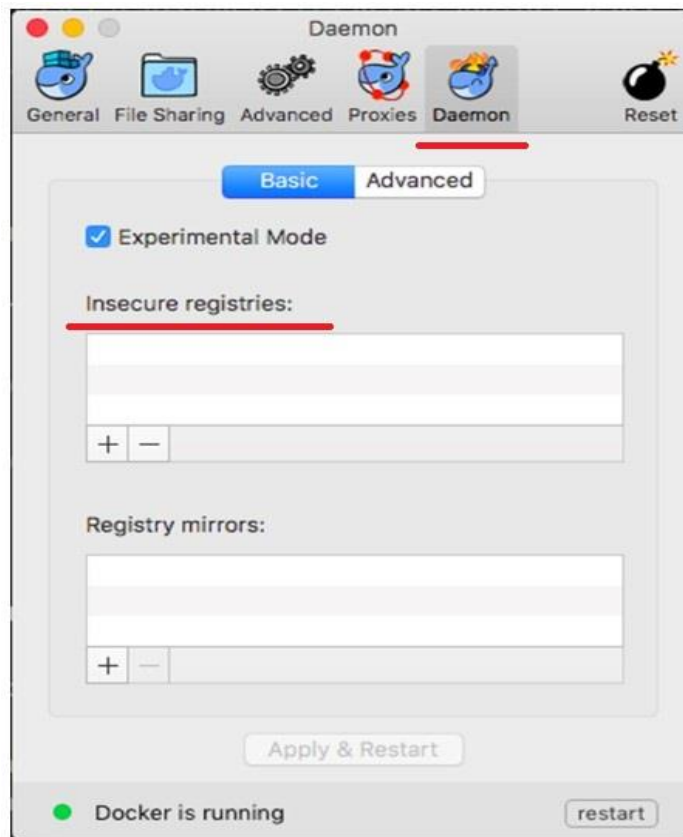
- اگر سیستم عامل ما لینوکس باشد باید در مسیر `/etc/docker/` فایل `daemon.json` را ساخته (در صورتی که موجود نباشد) و پارامتر زیر را به آن اضافه نماییم:

```
{
  "insecure-registries": [
    "192.168.253.10:8082",
    "192.168.253.10:8083"
  ],
}
```

آدرس 192.168.253.10 مربوط به سرور *Nexus* است و پورت 8082 پورتهای است که برای *docker proxy* انتخاب کردیم، همچنین 8083 نیز پورتهای است که برای *docker hosted* انتخاب نمودیم.

سپس با دستور `sudo systemctl restart docker` داکر را *restart* می نماییم.

- اگر از *windows Server* استفاده می نماییم تغییرات را در مسیر `C:\ProgramData\docker\config\daemon.json` انجام می دهیم.
- اگر از *Docker Desktop for Mac* و یا *Docker Desktop for Windows* استفاده می کنیم، با کلیک روی آیکون *Docker* و انتخاب *Preferences* و سپس *Daemon* + مطابق شکل 5 باکسی باز می شود که می توانیم پارامترها را اضافه نماییم. (لازم به ذکر است که *Docker Desktop* فقط بر روی *Windows 10* کار می کند و در نسخه های قدیمی تر باید از *Docker toolbox* استفاده نماییم).



شکل 5 اضافه کردن *insecure registry*

- اگر از Docker Toolbox استفاده می نمایم هنگام ساختن Docker-Machine باید پارامتر **Insecure-Registry** را به آن اضافه نمایم. این کار مطابق با شکل 6 قابل انجام است:

```

HMAHMOUDI-S-PC+admin@HMAhmoUDI-S-PC MINGW64 /c/Program Files/Docker Toolbox
$ docker-machine create --driver virtualbox --engine-insecure-registry 192.168.253.10:8082 --engine-insecure-registry 192.168.253.10:8083 TestNexus
Running pre-create checks...
<TestNexus> Unable to get the latest Boot2Docker ISO release version: Get https://api.github.com/repos/boot2docker/boot2docker/releases/latest: dial tcp: lookup api.github.com: getaddrinfo: This is usually a temporary error during hostnam
e resolution and means that the local server did not receive a response from an
authoritative server.
Creating machine...
<TestNexus> Unable to get the latest Boot2Docker ISO release version: Get https://api.github.com/repos/boot2docker/boot2docker/releases/latest: dial tcp: lookup
api.github.com: getaddrinfo: This is usually a temporary error during hostnam
e resolution and means that the local server did not receive a response from an
authoritative server.
<TestNexus> Copying C:\Users\admin\.docker\machine\cache\boot2docker.iso to C:\U
sers\admin\.docker\machine\machines\TestNexus\boot2docker.iso...
<TestNexus> Creating VirtualBox VM...
<TestNexus> Creating SSH key...
<TestNexus> Starting the VM...
<TestNexus> Check network to re-create if needed...
<TestNexus> Windows might ask for the permission to configure a dhcp server. Som
etimes, such confirmation window is minimized in the taskbar.
<TestNexus> Waiting for an IP...
Waiting for machine to be running, this may take a few minutes...
Detecting operating system of created instance...
Waiting for SSH to be available...
Detecting the provisioner...
Provisioning with boot2docker...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on this vi
rtual machine, run: C:\Program Files\Docker Toolbox\docker-machine.exe env TestN
exus
HMAHMOUDI-S-PC+admin@HMAhmoUDI-S-PC MINGW64 /c/Program Files/Docker Toolbox

```

شکل 6 ساخت docker machine

لازم به ذکر است اگر بخواهیم پارامترها را به Docker-Machine ی که قبلا ساخته ایم اضافه کنیم باید از طریق دستورات زیر فایل Profile را ادیت نماییم (TestNexus اسم ماشینی است که می خواهیم پارامترها را به آن اضافه نماییم).

```

$ docker-machine ssh TestNexus
$ sudo vi /var/lib/boot2docker/profile

```

و مقادیر زیر را به آن اضافه نماییم:

```

EXTRA_ARGS="
--insecure-registry 192.168.253.10:8082
--insecure-registry 192.168.253.10:8083
"

```

سپس با دستور زیر ماشین را Restart می نماییم:

```
$ docker-machine restart {machineName}
```

پس از انجام این تغییرات حال می توان به Nexus لاگین نموده و از Repository ها استفاده نماییم.

4.2 لاگین کردن به Repository های ساخته شده

برای لاگین کردن کافیه که از دستور لاگین استفاده نماییم و Username و Password ادمین Nexus را وارد نماییم که در شکل 7 نشان داده شده است.

```
HMAHMOUDI-S-PC+admin@HMahmoudi-S-PC MINGW64 /c/Program Files/Docker Toolbox
$ docker login -u admin -p admin123 192.168.253.10:8082
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
Login Succeeded

HMAHMOUDI-S-PC+admin@HMahmoudi-S-PC MINGW64 /c/Program Files/Docker Toolbox
$ docker login -u admin -p admin123 192.168.253.10:8083
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
Login Succeeded

HMAHMOUDI-S-PC+admin@HMahmoudi-S-PC MINGW64 /c/Program Files/Docker Toolbox
```

شکل 7 لاگین به Repository ها

4.3 Pull کردن Image از طریق پروکسی ساخته شده

پس از لاگین به پروکسی به راحتی می توان از طریق آن Image های داکر را Pull نمود که در شکل 8 طریقه انجام این کار نشان داده شده است.

```
HMAHMOUDI-S-PC+admin@HMahmoudi-S-PC MINGW64 /c/Program Files/Docker Toolbox
$ docker run 192.168.253.10:8082/hello-world
Unable to find image '192.168.253.10:8082/hello-world:latest' locally
latest: Pulling from hello-world
1b930d010525: Pull complete
Digest: sha256:92c7f9c92844b5d0a101b22f7c2a7949e40f8ea90c8b3bc396879d95e899a
Status: Downloaded newer image for 192.168.253.10:8082/hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

HMAHMOUDI-S-PC+admin@HMahmoudi-S-PC MINGW64 /c/Program Files/Docker Toolbox
```

شکل 8 pull کردن Image از طریق پروکسی

طی مراحل زیر از این بابت مطمئن خواهیم شد زمانی که یک Image را مجدداً از Proxy Repository خواهیم Pull نماییم، Nexus دوباره آن را دانلود نخواهد کرد و از Repository خود آن را در اختیار Client قرار می دهد. در شکل 9 این مراحل نشان داده شده است.

- ابتدا لیست Image های Client را مشاهده می کنیم. همانگونه که مشخص است 192.168.253.10:8082/tomcat یکبار از پروکسی گرفته شده است (پروکسی یکبار آن را دانلود کرده است).
- 192.168.253.10:8082/tomcat را از لیست image های کلاینت پاک می کنیم.
- Tomcat را مجدداً از پروکسی Pull می کنیم. همانگونه که ملاحظه می شود، Image دیگر دانلود نمی شود و از پروکسی دریافت می شود.

```

HMAHMOUDI-S-PC+admin@HMahmoudi-S-PC MINGW64 /c/Program Files/Docker Toolbox
$ docker images
REPOSITORY              TAG                IMAGE ID            CREATE
D                        SIZE
192.168.253.10:8082/tomcat latest            168588387c68        2 week
s ago                   463MB
192.168.253.10:8083/tomi latest            168588387c68        2 week
s ago                   463MB
192.168.253.10:8082/hello-world latest           fce289e99eb9        8 week
s ago                   1.84kB
192.168.253.10:8083/hell latest            fce289e99eb9        8 week
s ago                   1.84kB

HMAHMOUDI-S-PC+admin@HMahmoudi-S-PC MINGW64 /c/Program Files/Docker Toolbox
$ docker rmi 192.168.253.10:8082/tomcat
Untagged: 192.168.253.10:8082/tomcat:latest
Untagged: 192.168.253.10:8082/tomcat@sha256:751898078f660f2570d65b2c55f6a3f71f3944d5f716b43b82372db9927ba4bc

HMAHMOUDI-S-PC+admin@HMahmoudi-S-PC MINGW64 /c/Program Files/Docker Toolbox
$ docker images
REPOSITORY              TAG                IMAGE ID            CREATE
D                        SIZE
192.168.253.10:8083/tomi latest            168588387c68        2 week
s ago                   463MB
192.168.253.10:8082/hello-world latest           fce289e99eb9        8 week
s ago                   1.84kB
192.168.253.10:8083/hell latest            fce289e99eb9        8 week
s ago                   1.84kB

HMAHMOUDI-S-PC+admin@HMahmoudi-S-PC MINGW64 /c/Program Files/Docker Toolbox
$ docker pull 192.168.253.10:8082/tomcat
Using default tag: latest
latest: Pulling from tomcat
Digest: sha256:751898078f660f2570d65b2c55f6a3f71f3944d5f716b43b82372db9927ba4bc
Status: Downloaded newer image for 192.168.253.10:8082/tomcat:latest

HMAHMOUDI-S-PC+admin@HMahmoudi-S-PC MINGW64 /c/Program Files/Docker Toolbox
$ docker images
REPOSITORY              TAG                IMAGE ID            CREATE
D                        SIZE
192.168.253.10:8083/tomi latest            168588387c68        2 week
s ago                   463MB
192.168.253.10:8082/tomcat latest            168588387c68        2 week
s ago                   463MB
192.168.253.10:8082/hello-world latest           fce289e99eb9        8 week
s ago                   1.84kB
192.168.253.10:8083/hell latest            fce289e99eb9        8 week
s ago                   1.84kB

HMAHMOUDI-S-PC+admin@HMahmoudi-S-PC MINGW64 /c/Program Files/Docker Toolbox
$

```

شکل 9 استفاده از قابلیت proxy Repository

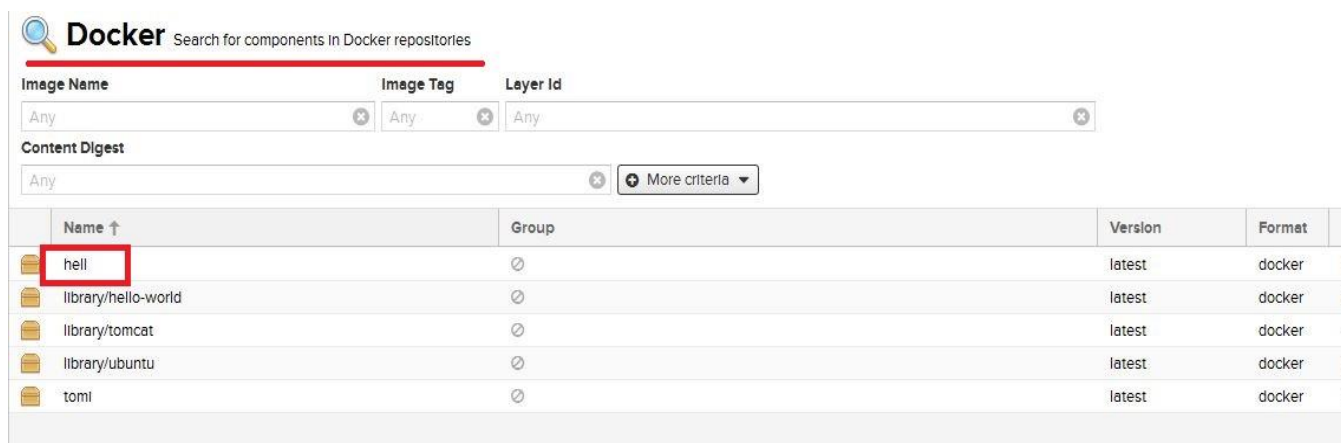
4.4 Push کردن Image به Host Repository ساخته شده در Nexus

برای Push کردن Image باید از دستورات Tag و Push استفاده نماییم در شکل 10-1 اجرای دستورات نشان داده شده است و در شکل 10-2 ملاحظه می شود که Image به Repository Host اضافه شده است.

```
HMAHMOUDI-S-PC+admin@HMAhmoudi-S-PC MINGW64 /c/Program Files/Docker Toolbox
$ docker tag 192.168.253.10:8082/hello-world 192.168.253.10:8083/hell

HMAHMOUDI-S-PC+admin@HMAhmoudi-S-PC MINGW64 /c/Program Files/Docker Toolbox
$ docker push 192.168.253.10:8083/hell
The push refers to repository [192.168.253.10:8083/hell]
af0b15c8625b: Pushed
latest: digest: sha256:92c7f9c92844bbbb5d0a101b22f7c2a7949e40f8ea90c8b3bc396879d95e899a size: 524
```

شکل 10-1 Push کردن Image به Host Repository



The screenshot shows the Docker Hub search interface. At the top, there is a search bar with the text "Docker Search for components in Docker repositories". Below the search bar, there are filters for "Image Name", "Image Tag", and "Layer Id", all set to "Any". There is also a "Content Digest" filter set to "Any". Below the filters, there is a table of search results. The table has columns for "Name", "Group", "Version", and "Format". The first row in the table is "hell", which is highlighted with a red box. The other rows are "library/hello-world", "library/tomcat", "library/ubuntu", and "toml".

Name ↑	Group	Version	Format
hell		latest	docker
library/hello-world		latest	docker
library/tomcat		latest	docker
library/ubuntu		latest	docker
toml		latest	docker

شکل 10-2 چک کردن Docker Repository در Nexus

4.5 Pull کردن Image از Host repository

برای Pull کردن Image از Nexus Host Repository کافی است که آدرس repository را در دستور Pull وارد نماییم. صحت عملکرد Host Repository در شکل 11 نشان داده شده است.

- ابتدا لیست Image های Docker Machine را چک می کنیم، همانگونه که مشاهده می شود
 - سپس این Image را از Docker Machine حذف می کنیم.
- 192.168.253.10:8083/tomi را قبلا به Host repository ، Push کرده ایم و در شکل 10-2 نیز قابل مشاهده است.

- در نهایت آن را از Host Repository ، Pull می نماییم که نشان دهنده عملکرد درست Repository است.

```

HMAHMOUDI-S-PC+admin@HMAhmodi-S-PC MINGW64 /c/Program Files/Docker Toolbox
$ docker images
REPOSITORY                                TAG                                IMAGE ID                                CREATE
D                                         SIZE                                D                                         TIME
192.168.253.10:8082/tomcat                 latest                             168588387c68                           2 week
s ago                                     463MB
192.168.253.10:8083/tomi                   latest                             168588387c68                           2 week
s ago                                     463MB
192.168.253.10:8082/hello-world            latest                             fce289e99eb9                           8 week
s ago                                     1.84kB
192.168.253.10:8083/hell                   latest                             fce289e99eb9                           8 week
s ago                                     1.84kB

HMAHMOUDI-S-PC+admin@HMAhmodi-S-PC MINGW64 /c/Program Files/Docker Toolbox
$ docker rmi 192.168.253.10:8083/tomi
Untagged: 192.168.253.10:8083/tomi:latest
Untagged: 192.168.253.10:8083/tomi@sha256:751898078f660f2570d65b2c55f6a3f71f3944d5f716b43b82372db9927ba4bc

HMAHMOUDI-S-PC+admin@HMAhmodi-S-PC MINGW64 /c/Program Files/Docker Toolbox
$ docker images
REPOSITORY                                TAG                                IMAGE ID                                CREATE
D                                         SIZE                                D                                         TIME
192.168.253.10:8082/tomcat                 latest                             168588387c68                           2 week
s ago                                     463MB
192.168.253.10:8082/hello-world            latest                             fce289e99eb9                           8 week
s ago                                     1.84kB
192.168.253.10:8083/hell                   latest                             fce289e99eb9                           8 week
s ago                                     1.84kB

HMAHMOUDI-S-PC+admin@HMAhmodi-S-PC MINGW64 /c/Program Files/Docker Toolbox
$ docker pull 192.168.253.10:8083/tomi
Using default tag: latest
latest: Pulling from tomi
Digest: sha256:751898078f660f2570d65b2c55f6a3f71f3944d5f716b43b82372db9927ba4bc
Status: Downloaded newer image for 192.168.253.10:8083/tomi:latest

HMAHMOUDI-S-PC+admin@HMAhmodi-S-PC MINGW64 /c/Program Files/Docker Toolbox
$ docker images
REPOSITORY                                TAG                                IMAGE ID                                CREATE
D                                         SIZE                                D                                         TIME
192.168.253.10:8082/tomcat                 latest                             168588387c68                           2 week
s ago                                     463MB
192.168.253.10:8083/tomi                   latest                             168588387c68                           2 week
s ago                                     463MB
192.168.253.10:8082/hello-world            latest                             fce289e99eb9                           8 week
s ago                                     1.84kB
192.168.253.10:8083/hell                   latest                             fce289e99eb9                           8 week
s ago                                     1.84kB

HMAHMOUDI-S-PC+admin@HMAhmodi-S-PC MINGW64 /c/Program Files/Docker Toolbox
$

```

شکل 11 Pull کردن Image از Host Repository