

Docker Private registry

The Registry is a stateless, highly scalable server side application that stores and lets you distribute Docker images. The Registry is open-source, under the permissive Apache license.

USES

You should use the Registry if you want to:

1. tightly control where your images are being stored
2. fully own your images distribution pipeline
3. integrate image storage and distribution tightly into your in-house development workflow

Requirements

Start your registry here. **--restart=always** means (If you want to use the registry as part of your permanent infrastructure, you should set it to restart automatically when Docker restarts or if it exits)

```
$ docker run -d -p 5000:5000 --restart=always --name registry registry:2
```

```
[ec2-user@ip-172-31-18-218 ~]$ docker run -d -p 5000:5000 --name registry registry:2
Unable to find image 'registry:2' locally
2: Pulling from library/registry
c87736221ed0: Pull complete
1cc8e0bb44df: Pull complete
54d33bcb37f5: Pull complete
e8afc091c171: Pull complete
b4541f6d3db6: Pull complete
Digest: sha256:3b00e5438ebd8835bcfa7bf5246445a6b57b9a50473e89c02ecc8e575be3ebb5
Status: Downloaded newer image for registry:2
7e1a2b24326414394b1cbcfc9e51efb7e986edf2da5d45e10831b80c00b679a5
```

Check the registry container or not using below command.

```
$ docker ps
```

```
[ec2-user@ip-172-31-18-218 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS            PORTS              NAMES
7e1a2b243264        registry:2         "/entrypoint.sh /etc..."   About a minute ago
Up About a minute   0.0.0.0:5000->5000/tcp      registry
```

Pull (or build) some image from the hub

```
$ docker pull ubuntu
```

Tag the image so that it points to your registry

```
$ docker image tag ubuntu localhost:5000/my-ubuntu
```

push it your private registry

```
$ docker push localhost:5000/my-ubuntu
```

remove your local image and try to pull from local repository.

```
$ docker image remove ubuntu  
$ docker image remove localhost:5000/my-ubuntu
```

```
$ docker pull localhost:5000/my-ubuntu
```

Now stop your registry and remove all data

```
$ docker container stop registry && docker container rm -v registry
```

Ref:

<https://docs.docker.com/registry/deploying/>

Customize the storage location

By default, your registry data is persisted as a docker volume on the host filesystem. If you want to store your registry contents at a specific location on your host filesystem, such as if you have an SSD or SAN mounted into a particular directory, you might decide to use a bind mount instead. A bind mount is more dependent on the filesystem layout of the Docker host, but more performant in many situations. The following example bind-mounts the host directory **/mnt/registry** into the registry container at **/var/lib/registry/**.

```
$ docker run -d \  
-p 5000:5000 \  
--restart=always \  
--name registry \  
-v /mnt/registry:/var/lib/registry \  
registry:2
```

get the list of images in my private registry.

```
$ curl -X GET http://localhost:5000/v2/\_catalog
```

If you want get the all tags form a particular image.

```
curl -X GET http://localhost:5000/v2/my-ubuntu/tags/list
```

if you want delete a tag first we need to get a catalog and tags and then use bellow command

```
curl -v --silent -H "Accept: application/vnd.docker.distribution.manifest.v2+json" -X GET http://localhost:5000/v2/my-ubuntu/manifests/latest 2>&1 | grep Docker-Content-Digest | awk '{print ($3)}'
```

```
[ec2-user@ip-172-31-18-218 ~]$ curl -v --silent -H "Accept: application/vnd.docker.distribution.manifest.v2+json" -X GET http://localhost:5000/v2/my-ubuntu/manifests/latest 2>&1 | grep Docker-Content-Digest | awk '{print ($3)}'
```

sha256:f2557f94cac1cc4509d0483cb6e302da841ecd6f82eb2e91dc7ba6cfd0c580ab

Run the command given below with manifest value:

```
curl -v --silent -H "Accept: application/vnd.docker.distribution.manifest.v2+json" -X DELETE http://127.0.0.1:5000/v2/my-ubuntu/manifests/sha256:f2557f94cac1cc4509d0483cb6e302da841ecd6f82eb2e91dc7ba6cfd0c580ab
```

if you face below error

```
* Connected to 127.0.0.1 (127.0.0.1) port 5000 (#0)
> DELETE /v2/my-ubuntu/manifests/sha256:f2557f94cac1cc4509d0483cb6e302da841ecd6f82eb2e91dc7ba6cfd0c580ab HTTP/1.1
> Host: 127.0.0.1:5000
> User-Agent: curl/7.61.1
> Accept: application/vnd.docker.distribution.manifest.v2+json
>
< HTTP/1.1 405 Method Not Allowed
< Content-Type: application/json; charset=utf-8
< Docker-Distribution-API-Version: registry/2.0
< X-Content-Type-Options: nosniff
< Date: Tue, 23 Apr 2019 13:29:37 GMT
< Content-Length: 78
<
{"errors":[{"code":"UNSUPPORTED","message":"The operation is unsupported."}]}
* Connection #0 to host 127.0.0.1 left intact
```

if you want to delete the images we must set the environment variable called

`REGISTRY_STORAGE_DELETE_ENABLED=true`

```
docker run -d -p 5000:5000 -e REGISTRY_STORAGE_DELETE_ENABLED=true -v /mnt/registry:/var/lib/registry --restart=always --name registry registry:2
```

```
[ec2-user@ip-172-31-18-218 ~]$ curl -v --silent -H "Accept: application/vnd.docker.distribution.manifest.v2+json" -X DELETE http://127.0.0.1:5000/v2/my-ubuntu/manifests/sha256:f2557f94cac1cc4509d0483cb6e302da841ecd6f82eb2e91dc7ba6cfd0c580ab
* Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to 127.0.0.1 (127.0.0.1) port 5000 (#0)
> DELETE /v2/my-ubuntu/manifests/sha256:f2557f94cac1cc4509d0483cb6e302da841ecd6f82eb2e91dc7ba6cfd0c580ab HTTP/1.1
> Host: 127.0.0.1:5000
> User-Agent: curl/7.61.1
> Accept: application/vnd.docker.distribution.manifest.v2+json
>
< HTTP/1.1 202 Accepted
< Docker-Distribution-API-Version: registry/2.0
< X-Content-Type-Options: nosniff
< Date: Tue, 23 Apr 2019 13:42:32 GMT
< Content-Length: 0
<
* Connection #0 to host 127.0.0.1 left intact
```

Install Jenkins using docker

```
$ docker run -p 8080:8080 -p 50000:50000 jenkins
```

This will store the workspace in /var/jenkins_home. All Jenkins data lives in there - including plugins and configuration. You will probably want to make that a persistent volume (recommended):

```
docker run -d -p 8080:8080 -p 50000:50000 -u root -v /Jenkins_home:/var/jenkins_home jenkins/jenkins:its
```

```
docker run --restart=always -d -p 8080:8080 -p 50000:50000 -u root -v /Jenkins_home:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock --name jenkins jenkins/jenkins:its
```

You can run builds on the master (out of the box) but if you want to attach build slave servers: make sure you map the port: -p 50000:50000 - which will be used when you connect a slave agent.

```
docker run -d --restart=always -u root -p 8080:8080 -p 50000:50000 -v /Jenkins_home:/var/jenkins_home -v $(which docker):/usr/bin/docker -v /var/run/docker.sock:/var/run/docker.sock --name jenkins jenkins/jenkins:its
```

```
docker run -d --restart=always -u root -p 8080:8080 -p 50000:50000 -v /Jenkins_home:/var/jenkins_home -v $(which docker):/usr/bin/docker -v /var/run/docker.sock:/var/run/docker.sock -v /home/ec2-user/.docker/config.json:/root/.docker/config.json --name jenkins jenkins/jenkins:its
```

Docker Private Repository as remote system

1. Install docker and run registry container in new machine.
2. go to docker host machine where we are building images and create a daemon.json file.

/etc/docker/daemon.json

```
{  
  
  "insecure-registries" : ["publicip:5000"]  
  
}
```

After adding json file must be restart docker service.

```
$ sudo service docker restart
```

Add the 5000 port in security group in registry server.

```
$ docker tag dbtest:v1 13.127.205.10:5000/dbtest:v1
```

```
$ docker push 13.127.205.10:5000/dbtest:v1
```

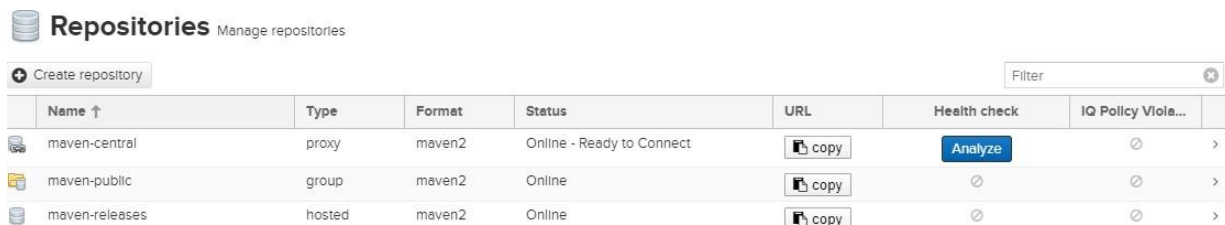
```
ubuntu@ip-172-31-32-157:/etc/docker$ docker tag dbtest:v1 13.127.205.10:5000/dbtest:v1
ubuntu@ip-172-31-32-157:/etc/docker$ docker push 13.127.205.10:5000/dbtest:v1
The push refers to repository [13.127.205.10:5000/dbtest]
060a4b458b2b: Pushed
f405467a84b0: Pushed
dc317a2d2b52: Pushed
969fd6ad73dc: Pushed
v1: digest: sha256:4c37f58747bc1dd7b97e5084fac96118c17c5e761fbd0fc5710759eb76ddc067 size: 1165
```

No we are able to push the images successfully.

CREATE NEXUS DOCKER PRIVATE REGISTRY












1. Install the nexus in your machine as per our nexus documentation.

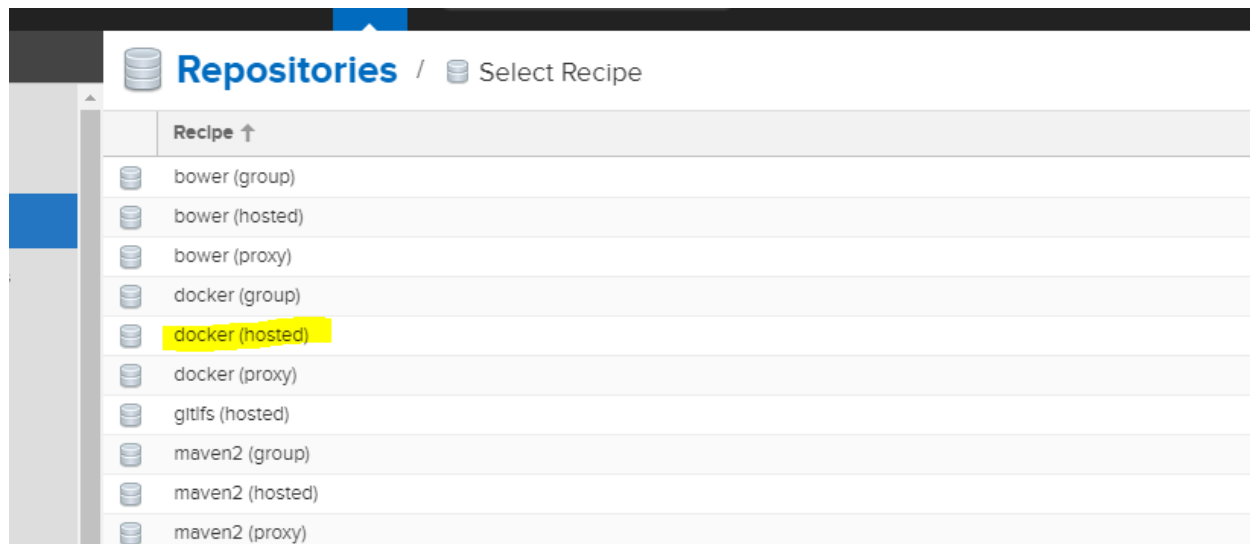
2. create docker repo server administration and configuration → Repositories → new Repository → select docker hosted



Repositories Manage repositories

+ Create repository Filter

Name ↑	Type	Format	Status	URL	Health check	IQ Policy Viola...	
 maven-central	proxy	maven2	Online - Ready to Connect	 copy	Analyze		>
 maven-public	group	maven2	Online	 copy			>
 maven-releases	hosted	maven2	Online	 copy			>



Enter the repo name http port which used to push images and select and enable docker v1 api and then create repository.

The screenshot shows the 'Create Repository: docker (hosted)' form. The breadcrumb trail is 'Repositories / Select Recipe / Create Repository: docker (hosted)'. The form includes the following sections:

- Name:** A unique identifier for this repository. The input field contains 'my_repo'.
- 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. For information on scaling the repositories see our [scaling documentation](#).*
 - HTTP:** Create an HTTP connector at specified port. Normally used if the server is behind a secure proxy. ☒ 8085
 - HTTPS:** Create an HTTPS connector at specified port. Normally used if the server is configured for https. ☐
- Allow anonymous docker pull:** ☐ Allow anonymous docker pull (DockerBearerToken Realm required)
- Docker Registry API Support**
 - Enable Docker V1 API:** ☒ Allow clients to use the V1 API to interact with this Repository

Repositories <small>Manage repositories</small>							
Create repository						Filter	
	Name ↑	Type	Format	Status	URL	Health check	IQ Policy Viola...
	maven-central	proxy	maven2	Online - Ready to Connect	copy	Analyze	
	maven-public	group	maven2	Online	copy		
	maven-releases	hosted	maven2	Online	copy		
	maven-snapshots	hosted	maven2	Online	copy		
	mydockerrepo	hosted	docker	Online	copy		
	my_repo	hosted	docker	Online	copy		
	nuget-group	group	nuget	Online	copy		
	nuget-hosted	hosted	nuget	Online	copy		
	nuget.org-proxy	proxy	nuget	Online - Ready to Connect	copy	Analyze	

Sonatype Nexus Repository Manager
OSS 3.33.1-01

Search components

Administration

- Repository
 - Repositories
 - Blob Stores
 - Cleanup Policies
 - Content Selectors
 - Proprietary Repositories
 - Routing Rules
- Security
 - Privileges
 - Roles
 - Users
 - Anonymous Access
 - LDAP
 - Realms**
 - SSL Certificates
 - IQ Server
- Support
- System

Realms Manage the active security realms and their order

Active realms:

Available

Filter

- Conan Bearer Token Realm
- Default Role Realm
- LDAP Realm
- npm Bearer Token Realm
- NuGet API-Key Realm
- Rut Auth Realm

Active

Local Authenticating Realm
 Local Authorizing Realm
Docker Bearer Token Realm

Save

Discard

add the 8085 ports in to security group of nexus server.

3. create a image with <serverpublicip>:port

```
docker tag myapp <publicip>:8085/myapp
```

4. should login with nexus credentials

```
docker login -u admin -p admin123 <publicip>:8085
```

Before pushing and login you must add the repo url in daemon.json file and then restart your docker daemon.

```
$ sudo service docker restart
```

```
/etc/docker/daemon.json
```

```
{  
  "insecure-registries" : ["publicip:8085"]  
}
```

5. push the image

```
Docker push <publicip>:8085/myapp
```

Push docker images to ECR repo

First need needs to setup aws cli

Create aws iam user → add administration policy → download AWS Access Key ID and AWS Secret Access Key.

If you are using ami linux2 we no need to install aws cli. If you are Ubuntu or some other os need to install aws cli refer below link or refer aws cli installation document.

<https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-linux.html#cliv2-linux-install>

you need to create repository **my-alpine** in aws ecr

```
$ aws ecr create-repository \  
  --repository-name my-alpine \  
  --image-tag-scheme latest
```

```
--image-scanning-configuration scanOnPush=true \
```

```
--region ap-south-1
```

create tag using below command

```
$ docker tag alpine 403959134869.dkr.ecr.ap-south-1.amazonaws.com/my-alpine
```

After you have installed and configured the AWS CLI, authenticate the Docker CLI to your default registry. That way, the **docker** command can push and pull images with Amazon ECR. The AWS CLI provides a **get-login** command to simplify the authentication process.

```
$ aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 403959134869.dkr.ecr.ap-south-1.amazonaws.com
```

You can see output like below

```
[ec2-user@ip-172-31-47-246 ~]$ aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 403959134869.dkr.ecr.ap-south-1.amazonaws.com
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

Then try to push docker image

```
$ docker push 403959134869.dkr.ecr.ap-south-1.amazonaws.com/my-alpine
```

```
[ec2-user@ip-172-31-47-246 ~]$ docker push 403959134869.dkr.ecr.ap-south-1.amazonaws.com/my-alpine
The push refers to repository [403959134869.dkr.ecr.ap-south-1.amazonaws.com/my-alpine]
50644c29ef5a: Pushed
latest: digest: sha256:a15790640a6690aa1730c38cf0a440e2aa44aaca9b0e8931a9f2b0d7cc90fd65 size: 528
```

Docker private repo ui third party lib

<https://hub.docker.com/r/joxit/docker-registry-ui>

<https://opensorcelibs.com/lib/joxit-docker-registry-ui>

<https://github.com/Joxit/docker-registry-ui/issues/25>

```
docker run -d -p 5000:5000 -e REGISTRY_STORAGE_DELETE_ENABLED=true -v  
`pwd`/config.yml:/etc/docker/registry/config.yml -v /mnt/registry:/var/lib/registry --  
restart=always --name registry registry:2.7.1
```

```
docker run -d -p 80:80 -e URL=http://127.0.0.1:5000 -e DELETE_IMAGES=true joxit/docker-registry-ui:1.5-static
```

config.yml

version: 0.1

log:

fields:

service: registry

storage:

delete:

enabled: true

cache:

blobdescriptor: inmemory

filesystem:

rootdirectory: /var/lib/registry

http:

addr: :5000

headers:

X-Content-Type-Options: [nosniff]

Access-Control-Allow-Origin: ['http://3.109.122.249']

Access-Control-Allow-Methods: ['HEAD', 'GET', 'OPTIONS', 'DELETE']

Access-Control-Allow-Headers: ['Authorization', 'Accept']

Access-Control-Max-Age: [1728000]

Access-Control-Allow-Credentials: [true]

Access-Control-Expose-Headers: ['Docker-Content-Digest']

```
docker run -d -p 5000:5000 -e REGISTRY_STORAGE_DELETE_ENABLED=true -v  
`pwd`/config.yml:/etc/docker/registry/config.yml -v /mnt/registry:/var/lib/registry --  
restart=always --name registry registry:2.7.1
```

```
docker run -d -p 80:80 -e URL=http://65.1.94.194:5000 -e DELETE_IMAGES=true --  
name=registry_ui joxit/docker-registry-ui:1.5-static
```