Using Docker V1

Overview

This page describes how to use Artifactory with the Docker V1 Registry API. If you are using the Docker V2 Registry API, please refer to Docker Registry.

For general information on using Artifactory with Docker, please refer to Artifactory as a Docker Registry.

Getting Started with Artifactory and Docker

Artifactory supports Docker transparently, meaning you can point the Docker client at Artifactory and issue push, pull and other commands in exactly the same way that you are used to when working directly with a private registry or Docker Hub.

To get started using Docker with Artifactory you need to execute the following steps:

1. Set up a web server as a reverse proxy
2. Create a local repository
3. Set up authentication
4. Push and pull images

The screencast at the end of this section provides a demonstration.

1. Setting up NGINX as a Reverse Proxy

Artifactory can only be used with Docker through a reverse proxy due to the following limitations of the Docker client:

1. You cannot provide a context path when providing the registry path (e.g *`localhost:8081/artifactory`* is not valid)
2. Docker will only send basic HTTP authentication when working against an HTTPS host

For Artifactory to work with Docker, the preferred web server is **NGINX v1.3.9** and above configured as a reverse proxy.

For other supported web servers, please refer to Alternative Proxy Servers.

Below is a sample configuration for NGINX which configures SSL on port 443 to a specific local repository in Artifactory (named `docker-local`) on a server called *`artprod.company.com`*.

Using Docker v1, Docker client v1.10 and Artifactory 4.4.3 known issue.

To avoid incompatibility when using Docker V1 with Docker 1.10, use the NGINX configuration displayed below and not the NGINX configuration generated by Artifactory v4.4.3.

This code requires NGINX to support chunked transfer encoding which is available from NGINX v1.3.9.

```
 [...]

http {

    ##
    # Basic Settings
    ##
    [...]

    server {
        listen 443;
        server_name artprod.company.com;

        ssl on;
        ssl_certificate /etc/ssl/certs/artprod.company.com.crt;
        ssl_certificate_key /etc/ssl/private/artprod.company.com.key;

        access_log /var/log/nginx/artprod.company.com.access.log;
        error_log /var/log/nginx/artprod.company.com.error.log;

        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Original-URI $request_uri;
        proxy_read_timeout 900;

        client_max_body_size 0; # disable any limits to avoid HTTP 413 for
large image uploads

        # required to avoid HTTP 411: see Issue #1486 (https://github.com/docker
/docker/issues/1486)
        chunked_transfer_encoding on;

        location /v1 {
            proxy_pass http://artprod.company.com:8081/artifactory/api/docker
/docker-local/v1;
        }
    }
}
```

Multiple Docker repositories and port bindings

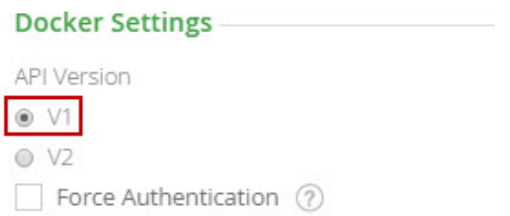
If you want to use multiple Docker repositories, you need to copy this configuration and bind different ports to each local repository in Artifactory. For details, please refer to Port Bindings.

Repository URL prefix

When accessing a Docker repository through Artifactory, the repository URL must be prefixed with **api/docker** in the path. For details, please refer to Docker Repository Path and Domain.

2. Creating a Local Docker Repository

This is done in the same way as when configuring a local repository to work with Docker V2, however, in the Docker Settings section, you should make sure to select V1 as the Docker API version.



Page Contents

Working with Artifactory SaaS

Due to limitations of the Docker client, in Artifactory SaaS there is a special configuration for each server with a sub-domain.

You need to create a new Docker enabled local repository named ***`docker-local.`***

Then, use the following address when working with the Docker client: ***`"${account_name}.jfrog.io"`***

3. Setting Up Authentication

When using Artifactory with Docker V1, you need to set your credentials manually by adding the following section to your `~/.docker/config.json` file.

~/.docker/config.json

```
{
        "auths" :{
                "https://artprod.company.com" : {
                        "auth": "<USERNAME>:<PASSWORD> (converted to base 64)",
                                "email": "youremail@email.com"
                },
                "https://artdev.company.com" : {
                        "auth": "<USERNAME>:<PASSWORD> (converted to base 64)",
                                "email": "youremail@email.com"
                }
        }
}
```

4. Pushing and Pulling Images

Pushing and pulling images when using Docker V1 is done in the same way as when using Docker V2. Please refer to Pushing and Pulling Images under the Docker Repositories page.

Watch the Screencast

Once you have completed the above setup you should beableuse the Docker client to transparently push images to and pull them from Docker repositories in Artifactory. You can see this in action in the screencast below.

Browsing Docker Repositories

Artifactory stores docker images in a layout that is made up of 2 main directories:

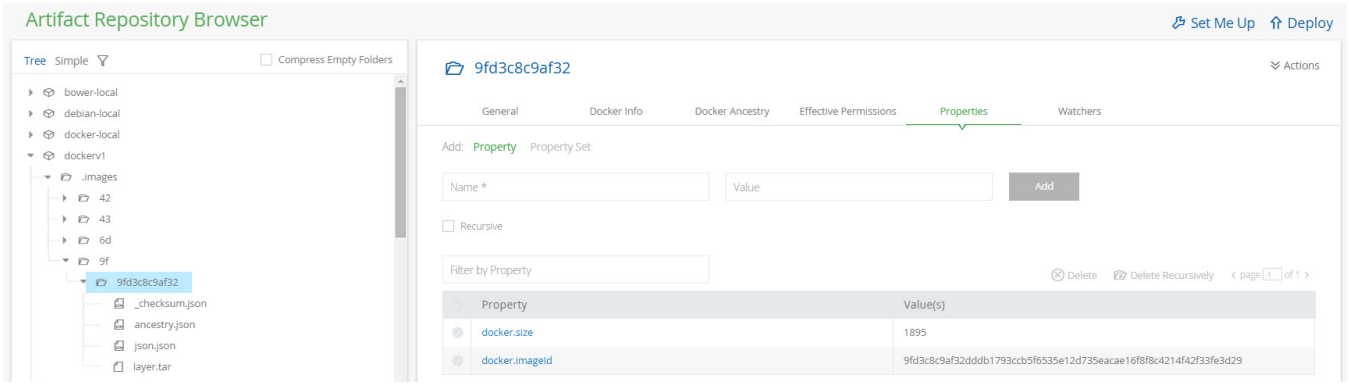
- **.images:** Stores all the flat docker images.
- **repositories:** Stores all the repository information with tags (similar to how repositories are stored in the Docker Hub).

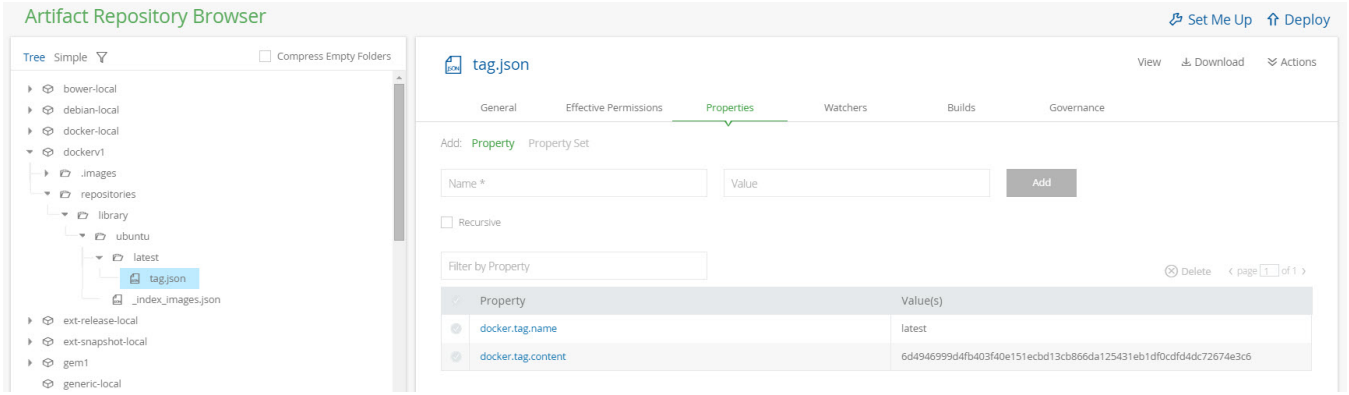
In addition, Artifactory annotates each deployed docker image with two properties:

- **docker.imageId:** The image id
- **docker.size:** The size of the image in bits

Deployed tags are also annotated with two properties:

- **docker.tag.name:** The tag name
- **docker.tag.content:** The id of the image that this tag points to

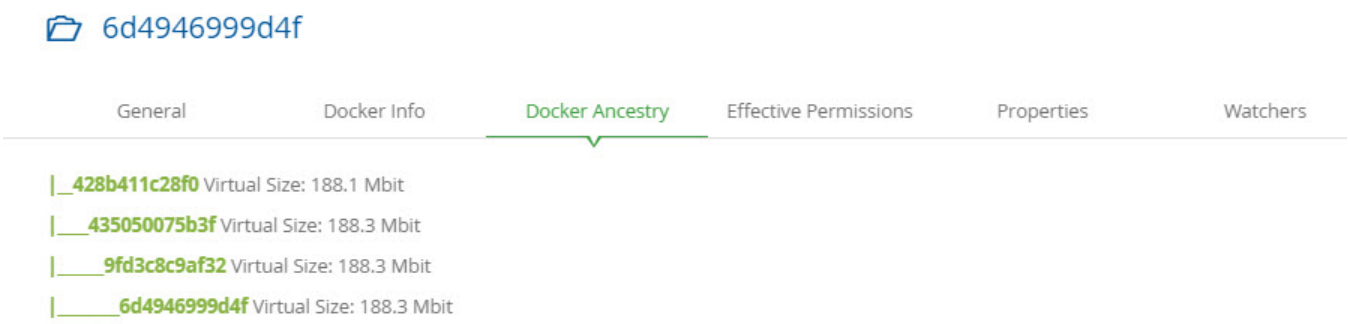




Viewing the Docker Images Tree

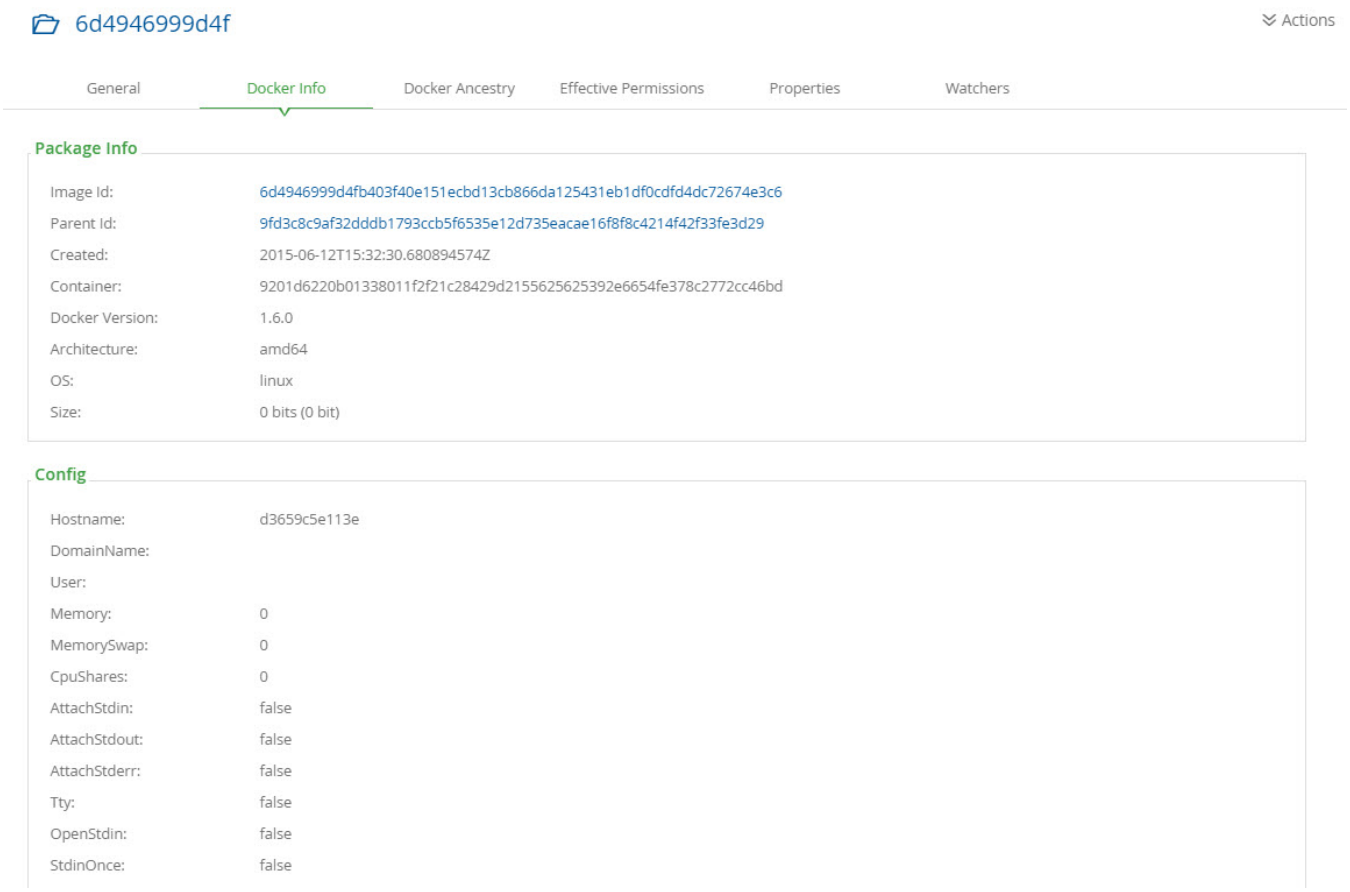
Artifactory lets you view the complete images tree for a specific image directly from the UI in a similar way to what you would get from the `docker images --tree` command.

In the **Artifacts** module **Tree Browser,** drill down to select the image you want to inspect. The metadata is displayed in the **Docker Ancestry** tab.



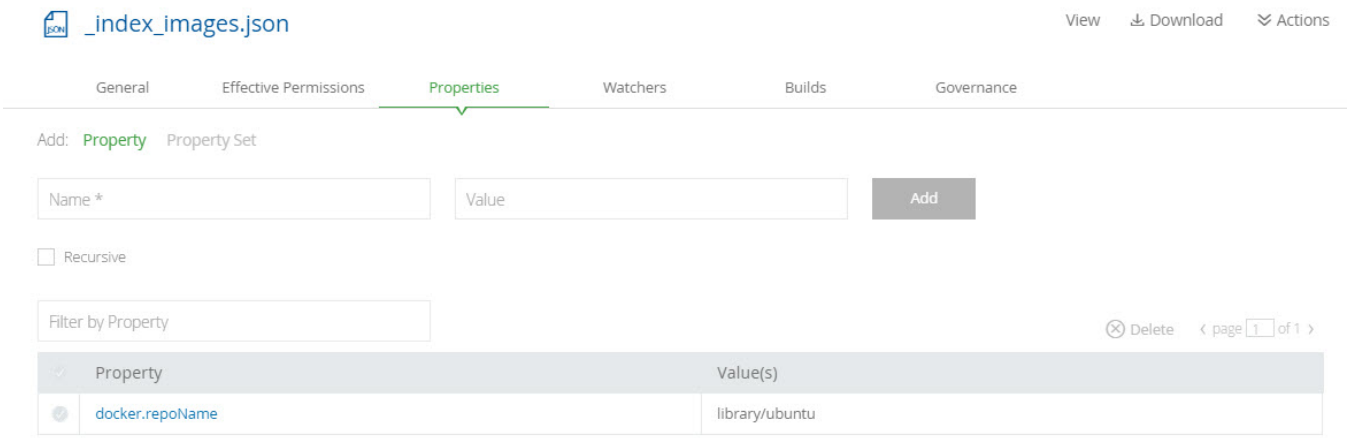
Viewing Individual Docker image Information

In the **Artifacts** module **Tree Browser,** drill down to select image you want to inspect. The metadata is displayed in the **Docker Info** tab.



Searching for Docker Images

In addition to other properties related to Docker repositories, you can also search for repositories using a property called `docker.repoName`, which represents the repository name (e.g., "library/ubuntu").



Promoting Docker Images with V1

Promoting Docker images with Docker V1 is done in exactly the same way as when Promoting Images with Docker V2.

Migrating a V1 repository to V2

We recommend using Docker V2 repositories when possible (provided your Docker client is version 1.6 and above).

If you have an existing Docker V1 repository, you can migrate its content into a V2 repository using the following endpoint with cURL:

```
POST api/docker/<repoKey>/v1/migrate
{
    "targetRepo" : "<targetRepo>",
    "dockerRepository" : "<dockerRepository>",
    "tag" : "<tag>"
}
```

where:

<repoKey>	Source repository key (For example, *docker-local* as used in this page)
<targetRepo>	The target Docker V2 repository to migrate to (For example, *docker-local2* as used in this page). The repository should be created before running the `migrate` endpoint.
<dockerRepository>	An optional docker repository name to migrate, if null - the entire source repository will be migrated. Default: ""
<tag>	An optional tag name to promote, if null - the entire docker repository will be promoted. Default: ""

An example for migrating the docker image *"jfrog/ubuntu"* with all of it's tags from *docker-local* to *docker-local2* using cURL would be:

```
curl -i -uadmin:password -X POST "http://localhost:8081/artifactory/api/docker/docker-local/v1/migrate" -H
"Content-Type: application/json" -d
'{"targetRepo":"docker-local2","dockerRepository":"jfrog/ubuntu"}'
```

Deletion and Cleanup

Artifactory natively supports removing tags and repositories and complies with the Docker Hub Spec.

Deletion of Docker tags and repositories automatically cleans up any orphan layers that are left (layers not used by any other tag/repository).

Currently, the Docker client does not support DELETE commands, but deletion can be triggered manually using cURL. Here are some examples:

Removing repositories and tags

```
//Removing the "jfrog/ubuntu" repository
 curl -uadmin:password -X DELETE "https://artprod.company.com/v1/repositories/jfrog/ubuntu"

//Removing the "12.04" tag from the "jfrog/ubuntu" repository
 curl -uadmin:password -X DELETE "https://artprod.company.com/v1/repositories/jfrog/ubuntu/tags/12.04"
```


Empty Directories

Any empty directories that are left following removal of a repository or tag will automatically be removed during the next folder pruning job (which occurs every 5 minutes by default).

Advanced Topics

Using a Self-signed SSL Certificate

From Docker version 1.3.1, you can use self-signed SSL certificates with `docker push/pull` commands, however for this to work, you need to specify the `--insecure-registry` daemon flag for each insecure registry.

For full details please refer to the Docker documentation.

For example, if you are running Docker as a service, edit the */etc/default/docker* file, and append the `--insecure-registry` flag with your registry URL to the DOCKER_OPTS variable as in the following example:

Edit the DOCKER_OPTS variable

```
DOCKER_OPTS="-H unix:///var/run/docker.sock --insecure-registry artprod.company.com"
```

For this to take effect, you need to restart the Docker service.

If you are using **Boot2Docker**, please refer to the **Boot2Docker** documentation for Insecure Registry.

If you do not make the required modifications to the `--insecure-registry` daemon flag, you should get the following error:

Error message

```
Error: Invalid registry endpoint https://artprod.company.com/v1/: Get https://artprod.company.com/v1/_ping:
x509: certificate signed by unknown authority.
```

Using previous versions of Docker

In order to use self-signed SSL certificates with previous versions of Docker, you need to manually install the certificate into the OS of each machine running the Docker client (see Issue 2687).

Alternative Proxy Servers

In addition to NGINX, you can setup Artifactory to work with Docker using Apache.

Apache Configuration

The sample configuration below configures SSL on port 443 and a server name of *`artprod.company.com`*.

```
<VirtualHost *:443>
  ServerName artprod.company.com

  ErrorLog ${APACHE_LOG_DIR}/error.log
  CustomLog ${APACHE_LOG_DIR}/access.log combined

  SSLEngine on
  SSLCertificateFile/etc/ssl/certs/artprod.company.com.pem
  SSLCertificateKeyFile   /etc/ssl/private/artprod.company.com.key

  ProxyRequests off
  ProxyPreserveHost on

  ProxyPass         / http://artprod.company.com:8080/artifactory/api/docker/docker-local/
  ProxyPassReverse  / http://artprod.company.com:8080/artifactory/api/docker/docker-local/
</VirtualHost>
```

Port Bindings

If you want to use multiple repositories, you need to copy the NGINX configuration and bind different ports to each local repository in Artifactory.

When binding a port other than 443, note that the configuration for the proxy header must be appended with the port number on the `proxy_set_header` line.

For example, for a server running on port 444 you should write `proxy_set_header Host $host:444`.

Docker Repository Path and Domain

When accessing a Docker repository through Artifactory, the repository URL must be prefixed with **api/docker** in the path.

You can copy the full URL from the UI using **Set Me Up** when the repository is selected in the Tree Browser.

For example, if you are using Artifactory standalone or as a local service, you would access your Docker repositories using the following URL:

*`http://localhost:8081/artifactory/`**`api/docker/`**`<repository key>`*

Also, the domain of your Docker repository must be expressed as an explicit IP address. The only exception is when working locally, you can use the *localhost* domain name as the proxy pass.

Support Matrix

Please refer to the support matrix under Docker Repositories.