Getting Started with Docker and Artifactory

Overview

There are three main ways you can use Docker with Artifactory and this document describes how to get started with each way.

Please review the brief summary below to decide which is the best way for you to use Docker with Artifactory.

Artifactory SaaS

The easiest way to start using Docker with Artifactory is through an Artifactory SaaS account.

In this mode, since Artifactory is a hosted service, you do not need to set up a reverse proxy and can create your Docker repositories and start pushing and pulling Docker images.

For more details, please refer to Getting Started with Artifactory SaaS.

Artifactory as a Docker Container - 1 Minute Setup

Artifactory is also available as a Docker image that is ready to run in a container.

The Artifactory Registry Docker image comes pre-configured with NGINX as a reverse proxy and local, remote and virtual repositories built-in so you are ready to start pushing and pulling images as soon as your container is running.

For more details, please refer to Getting Started with Artifactory as a Docker Container.

Artifactory On-Prem

You can setup your on-prem installation of Artifactory Pro to work with Docker.

Since the Docker client requires a different hostname for each registry you will need to configure a reverse proxy when using this method.

For more details, please refer to Getting Started with Artifactory Pro On-Prem.

Page Contents

Getting Started with Artifactory SaaS

Using Docker repositories with Artifactory SaaS is quick and easy to use.

Since, with Artifactory SaaS, you are using Artifactory as a hosted service, there is no need to configure Artifactory with a reverse proxy.

The example at the end of this section shows a complete process of creating a Docker repository, logging in, pulling an image and pushing an image.

Using Docker Client with Artifactory SaaS

To use the Docker client with one of your Artifactory SaaS Docker repositories, you can use the native Docker client to login to each Docker repository, pull, and push images as shown in the following example:

```
// Login to your repository use the following command with your Artifactory SaaS credentials
docker login ${server-name}-{repo-name}.jfrog.io

// Pull an image using the following command
docker pull ${server-name}-{repo-name}.jfrog.io/<image name>

// To push an image, first tag it and then use the push command
docker tag <image name> ${server-name}-{repo-name}.jfrog.io/<image name>
docker push ${server-name}-{repo-name}.jfrog.io/<image name>
```

Test Your Setup

You can test your setup with this example that assumes you are using an Artifactory SaaS server named "**acme**".z

The scenario it demonstrates is:

- Pulling the "hello-world" Docker image
- Logging into your local Docker repository
- Retagging the "hello-world" image, and the pushing it into your local Docker repository

Start by creating a local Docker repository called `dockerv2-local`.

```
// Pull the "hello-world" image
docker pull hello-world

// Login to repository dockerv2-local
docker login acme-dockerv2-local.jfrog.io

// Tag the "hello-world" image
docker tag hello-world acme-dockerv2-local.jfrog.io/hello-world

// Push the tagged "hello-world" image to dockerv2-local
docker push acme-dockerv2-local.jfrog.io/hello-world
```

Getting Started with Artifactory as a Docker Container

Artifactory may be pulled from Bintray as Docker image, this is the easiest way to use Artifactory as a Docker repository on-prem.

This installation comes bundled with an NGINX proxy server that uses a self-signed certificate and is configured for access using the subdomain method.

In addition, it comes pre-configured with repositories *`docker-dev-local2`*, *`docker-prod-local2`* , *`docker-remote`*, and *`docker-virtual`* out-of-the-box with docker-dev-local2 preset as the Default Deployment Repository.

An example of the Dockerfile used to create this image along with the NGINX configuration can be found in JFrog artifactory-docker-builder repository in GitHub.

Subdomain Method

This is the recommended method since you will not need to change the reverse proxy configuration for each new Docker repository created.

Out-of-the-box, Artifactory Pro Registry is also pre-configured with the domain **`*.art.local`** and the relevant self-signed certificate.

Prerequisites

1. You need to add the following to your DNS or */etc/hosts* file:

   ```
   <ip-address> docker-virtual.art.local docker-dev-local2.art.local docker-prod-local2.art.local
   ```

2. Since the certificate is self-signed, you need to configure the Docker client to work with an insecure registry by adding the following line to your */etc/default/docker* file (you may need to create the file if it does not already exist):

```
DOCKER_OPTS="$DOCKER_OPTS --insecure-registry docker-virtual.art.local --insecure-registry docker-dev-local2.art.local --insecure-registry docker-prod-local2.art.local --insecure-registry docker-remote.art.local"
```

3. Restart your Docker engine.

To start your Artifactory registry container run the following command:

```
docker run -d --name artifactory -p 80:80 -p 8081:8081 -p 443:443 jfrog-docker-reg2.bintray.io/jfrog/artifactory-registry:latest
```

Once the container is running you will need to activate Artifactory using your license. If you do not have a license you can get a free 30 day Trial license.

For more options on how to run Artifactory as a Docker container, please refer to **Running with Docker**.

Test Your Setup

You can test your setup with this example .

The scenario it demonstrates is:

- Pulling the "hello-world" Docker image
- Logging into your virtual Docker repository
- Retagging the "hello-world" image, and the pushing it into your virtual Docker repository

The Artifactory Docker registry is already configured with a virtual repository called docker-virtual

```
// Pull the "hello-world" image
docker pull hello-world

// Login to repository docker-virtual
docker login docker-virtual.art.local

// Tag the "hello-world" image
docker tag hello-world docker-virtual.art.local/hello-world

// Push the tagged "hello-world" image to docker-virtual
docker push docker-virtual.art.local/hello-world
```

Getting Started with Artifactory Pro On-Prem

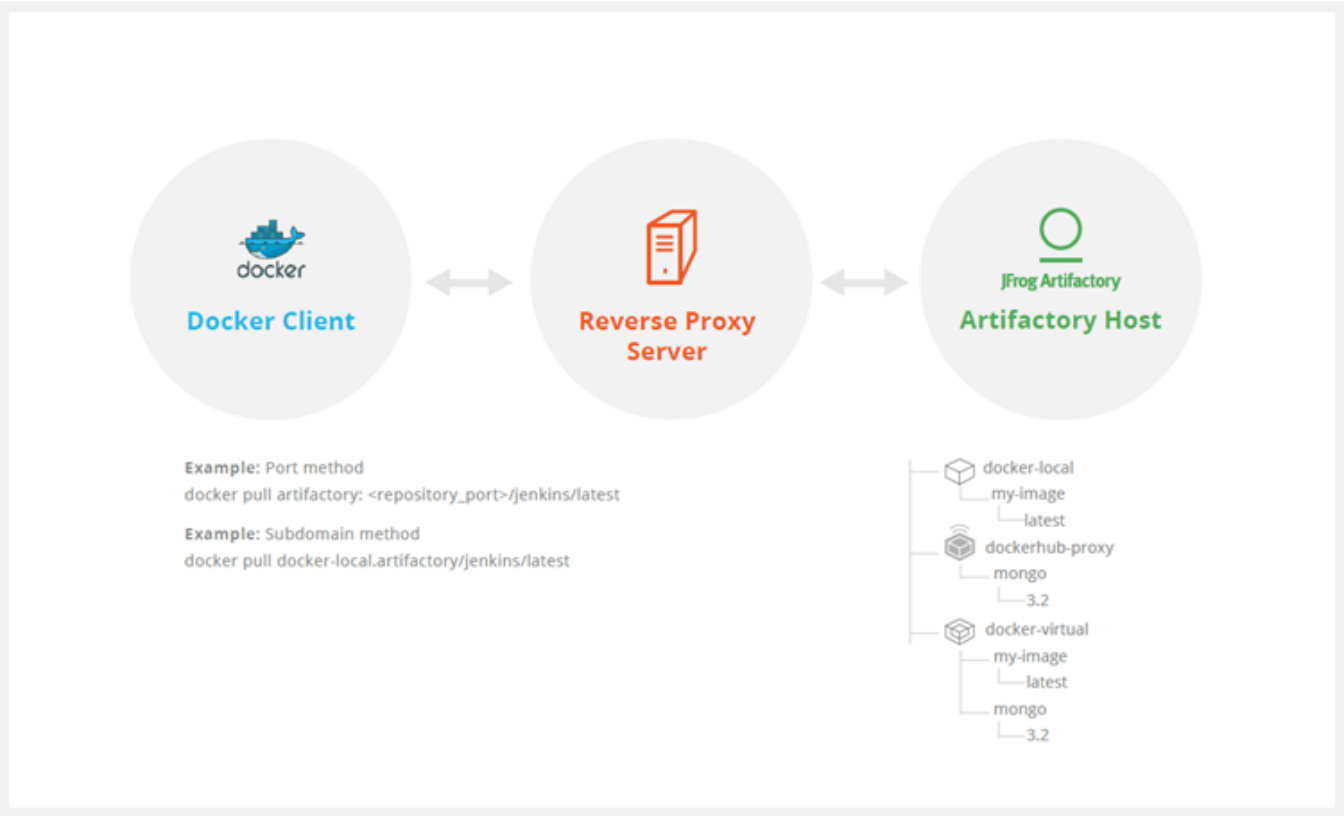
Using Artifactory Pro On-Prem with Docker requires a reverse proxy due to the following limitations of the Docker client:

1. You cannot use a context path when providing the registry path (e.g *localhost:8080/artifactory* is not valid)
2. Docker will only send basic HTTP authentication when working against an HTTPS host

Therefore, you need a reverse proxy to map Docker commands to Docker registries in Artifactory using either the **subdomain method** or the **ports method**.

Testing or evaluating?

If you are currently only testing or evaluating using Artifactory with Docker, we recommend running Artifactory as a Docker container which is easily installed and comes with a proxy server and Docker registries pre-configured out-of-the-box. You can be up and running in minutes.



With the ports method, a port number mapped to each Artifactory Docker registry. While this is an easy way to get started, you will need to modify your reverse proxy configuration and add a new mapping for each new Docker registry you define in Artifactory. In addition, firewalls and other restrictions by your IT department may restrict port numbers making the ports method not feasible.

With the subdomain method, you only need to configure your reverse proxy once, and from then on, the mapping from Docker commands to Docker registries in Artifactory is dynamic and requires no further modification of your reverse proxy configuration.

We recommend to use the subdomain method since it will require one time effort.

The Subdomain Method

Getting started with Docker and your on-prem Artifactory Pro installation using the subdomain method involves two basic steps:

1. Configuring Artifactory and your reverse proxy.
2. Configuring your Docker client.

Configuring Artifactory and Your Reverse Proxy

To configure Artifactory and your reverse proxy using the subdomain method, carry out the following steps:

1. Make sure Artifactory is up and running, and is activated with a valid license.
2. Create your local Docker repository. In our example below we will use a repository named **docker-local**.
3. Make sure you have a reverse proxy server up and running.
4. Obtain a **wildcard** SSL certificate or use a wildcard self-signed certificate.

   > ⚠ Make sure your certificate matches the **Artifactory hostname** used in your reverse proxy configuration. In our example below we will use **art.local**.

5. Configure your reverse proxy. Artifactory's Reverse Proxy Configuration Generator can generate your complete reverse proxy configuration file for supported servers. All you need to do is fill in the fields in according to how your reverse proxy is set up while making sure to:

   a. Use the correct **Artifactory hostname** in the **Public Server Name** field (in our example this will be **art.local**)
   b. Select **Subdomain** as the **Reverse Proxy Method** under **Docker Reverse Proxy Settings**

   NGINX
   Copy the code snippet generated by the configuration generator into your *artifactory-nginx.conf* file, and place it in your */etc/nginx/sites-available* directory.
   Create the following symbolic link.

```
sudo ln -s /etc/nginx/sites-available/artifactory-nginx.conf /etc/nginx/sites-enabled/artifactory-nginx.conf
```

Apache HTTPD

Copy the code snippet generated by the configuration generator into your *artifactory-apache.conf* file and place it in you */etc/apache2/sites-available* directory.

Create the following symbolic link:

```
sudo ln -s /etc/apache2/sites-available/artifactory-apache.conf /etc/apache2/sites-enabled/artifactory-apache.conf
```

Configuring Your Docker Client

To configure your Docker client, carry out the following steps

1. Add the following to your DNS or to the client's */etc/hosts* file:

```
<ip-address> docker-local.art.local
```

2. If you are using a Self-Signed Certificate you need to configure the Docker client to work with an insecure registry by adding the following line to your */etc/default/docker* file (you may need to create the file if it does not already exist):

```
DOCKER_OPTS="$DOCKER_OPTS --insecure-registry docker-local.art.local"
```

3. Restart your Docker engine.

Test Your Setup

To verify your reverse proxy is configured correctly, run the following command:

```
// Make sure the following results in return code 200
curl -I -k -v https://<artifactory url>
```

Run the following commands to ensure your proxy configuration is functional and can communicate with Artifactory:

```
// Pull the "hello-world" image
docker pull hello-world

// Login to repository docker-local
docker login docker-local.art.local

// Tag the "hellow-world" image
docker tag hello-world docker-local.art.local/hello-world

// Push the tagged "hello-world" image to docker-local
docker push docker-local.art.local/hello-world
```

The Ports Method

Getting started with Docker and your on-prem Artifactory Pro installation using the ports method involves two basic steps:

1. Configuring Artifactory and your reverse proxy.
2. Configuring your Docker client.

Configuring Artifactory and Your Reverse Proxy

To configure Artifactory and your reverse proxy using the ports method, carry out the following steps:

1. Make sure Artifactory is up and running, and is activated with a valid license.
2. Create your local Docker repository. In our example below we will use a repository named **docker-local**.
3. Make sure you have a reverse proxy server up and running.
4. Obtain an SSL certificate or use a Self-Signed certificate that can be generated following this example.

Make sure your certificate matches the **Artifactory hostname** used in your reverse proxy configuration. In our example below we will use **art.local**.

5. Configure your reverse proxy. Artifactory's **Reverse Proxy Configuration Generator** can generate your complete reverse proxy configuration file for supported servers. All you need to do is fill in the fields in according to how your reverse proxy is set up while making sure to:
    a. Use the correct **Artifactory hostname** in the **Public Server Name** field
    b. Select **Ports** as the **Reverse Proxy Method** under **Docker Reverse Proxy Settings.** In the example below, we will use port **5001** to bind repository **docker-local.**

NGINX

For Artifactory to work with Docker, the preferred web server is **NGINX v1.3.9** and above.
First, you need to create a self-signed certificate for NGINX as described here for Ubuntu.
Then use Artifactory's **Reverse Proxy Configuration Generator** to generate the configuration code snippet for you.
Copy the code snippet into your *artifactory-nginx.conf* file and place it in your */etc/nginx/sites-available* directory.
Finally, create the following symbolic link:

```
sudo ln -s /etc/nginx/sites-available/artifactory-nginx.conf /etc/nginx/sites-enabled/artifactory-nginx.
conf
```

Apache HTTPD

Install Apache HTTP server as a reverse proxy and then install the required modules.

Create the following symbolic link:

```
sudo ln -s /etc/apache2/mods-available/slotmem_shm.load /etc/apache2/mods-enabled/slotmem_shm.load
```

Similarly, create corresponding symbolic links for:

- headers
- proxy_balancer
- proxy_load
- proxy_http
- proxy_connect
- proxy_html
- rewrite.load
- ssl.load
- lbmethod_byrequests.load

Then use Artifactory's **Reverse Proxy Configuration Generator** to generate the configuration code snippet for you.
Copy the code snippet into your *artifactory.conf* file and place it in your */etc/apache2/sites-available* directory.

HAProxy

First, you need to create a self-signed certificate for HAProxy as described here for Ubuntu.

Then, copy the code snippet below into your */etc/haproxy/haproxy.cfg* file. After editing the file as described in the snippet, you can test your configuration using the following command:

```
haproxy -f /etc/haproxy/haproxy.cfg -c
```

HAProxy v1.5 Configuration

```
# haproxy server configuration
# version 1.0
# History
# ------------------------------------------------------------------------
# Features enabled by this configuration
# HA configuration
# port 80, 443  Artifactory GUI/API
#
# This uses ports to distinguish artifactory docker repositories
# port 443  docker-virtual (v2) docker v1 is redirected to docker-dev-local.
# port 5001 docker-prod-local (v1); docker-prod-local2 (v2)
# port 5002 docker-dev-local (v1); docker-dev-local2 (v2)
```

```
#
# Edit this file with required information enclosed in <...>
# 1. certificate and key
# 2. artifactory-host
# 3  replace the port numbers if needed
# ---------------------------------------------------------------------
global
        log 127.0.0.1   local0
        chroot /var/lib/haproxy
        maxconn 4096
        user haproxy
        group haproxy
        daemon
        tune.ssl.default-dh-param 2048
        stats socket /run/haproxy/admin.sock mode 660 level admin
defaults
        log     global
        mode    http
        option  httplog
        option  dontlognull
        option  redispatch
        option  forwardfor
        option  http-server-close
        maxconn 4000
        timeout connect 5000
        timeout client 50000
        timeout server 50000
        errorfile 400 /etc/haproxy/errors/400.http
        errorfile 403 /etc/haproxy/errors/403.http
        errorfile 408 /etc/haproxy/errors/408.http
        errorfile 500 /etc/haproxy/errors/500.http
        errorfile 502 /etc/haproxy/errors/502.http
        errorfile 503 /etc/haproxy/errors/503.http
        errorfile 504 /etc/haproxy/errors/504.http
frontend normal
         bind *:80
         bind *:443 ssl crt </etc/ssl/certs/server.bundle.pem>
         mode http
         option forwardfor
         reqirep ^([^\ :]*)\ /v2(.*$) \1\ /artifactory/api/docker/docker-virtual/v2\2
         reqadd X-Forwarded-Proto:\ https if { ssl_fc }
         option forwardfor header X-Real-IP
         default_backend normal

# if only need to access the docker-dev-local2 then skip this section. Docker-virtual can be configured
to deploy to docker-dev-local2 frontend dockerhub
         bind *:5000 ssl crt </etc/ssl/certs/server.bundle.pem>
         mode http
         option forwardfor
         option forwardfor header X-Real-IP
         reqirep ^([^\ :]*)\ /v2(.*$) \1\ /artifactory/api/docker/docker-remote/v2\2
         reqadd X-Forwarded-Proto:\ https if { ssl_fc }
         default_backend normal

# if only need to access the docker-dev-local2 then skip this section. Docker-virtual can be configured
to deploy to docker-dev-local2 frontend dockerprod
         bind *:5001 ssl crt </etc/ssl/certs/server.bundle.pem>
         mode http
         option forwardfor
         option forwardfor header X-Real-IP
             reqirep ^([^\ :]*)\ /v1(.*$) \1\ /artifactory/api/docker/docker-prod-local/v1\2
             reqirep ^([^\ :]*)\ /v2(.*$) \1\ /artifactory/api/docker/docker-prod-local2/v2\2
         reqadd X-Forwarded-Proto:\ https if { ssl_fc }
         default_backend normal

# if only need to access the docker-dev-local2 then skip this section. Docker-virtual can be configured
to deploy to docker-dev-local2 frontend dockerdev
         bind *:5002 ssl crt </etc/ssl/certs/server.bundle.pem>
         mode http
         option forwardfor
         option forwardfor header X-Real-IP
```

```
            reqirep ^([^\ :]*)\ /v1(.*$) \1\ /artifactory/api/docker/docker-dev-local/v1\2
            reqirep ^([^\ :]*)\ /v2(.*$) \1\ /artifactory/api/docker/docker-dev-local2/v2\2
        reqadd X-Forwarded-Proto:\ https if { ssl_fc }
        default_backend normal

# Artifactory Non HA Configuration
# i.e server artifactory 198.168.1.206:8081
#
backend normal
        mode http
        server <artifactory-host> <artifactory-host ip address>:<artifactory-host port>

#
# Artifactory HA Configuration
# Using default failover interval - rise = 2; fall =3 3; interval - 2 seconds
# backend normal
#        mode http
#        balance roundrobin
#        option httpchk OPTIONS /
#        option forwardfor
#        option http-server-close
#        appsession JSESSIONID len 52 timeout 3h
#        server <artifactory-host-ha1> <artifactory-host ip address>:<artifactory-host port>
#                  server <artifactory-host-ha2> <artifactory-host ip address>:<artifactory-host port>
```

Configuring Your Docker Client

To configure your Docker client, carry out the following steps

1. Add the following to your DNS or to the client's */etc/hosts* file:

```
<ip-address> art.local
```

2. If you are using a Self-Signed Certificate you need to configure the Docker client to work with an insecure registry by adding the following line to your */etc/default/docker* file (you may need to create the file if it does not already exist):

```
DOCKER_OPTS="$DOCKER_OPTS --insecure-registry art.local:5001"
```

3. Restart your Docker engine.

Test Your Setup

To verify your reverse proxy is configured correctly, run the following command:

```
// Make sure the following results in return code 200
curl -I -k -v https://<artifactory url>
```

Run the following commands to ensure your proxy configuration is functional and can communicate with Artifactory. In this example, we will pull down a Docker image, tag it and then deploy it to our our **docker-local** repository that is bound to **port 5001**:

```
// Pull the "hello-world" image
docker pull hello-world

// Login to repository docker-local
docker login art-local:5001

// Tag the "hello-world" image
docker tag hello-world art-local:5001/hello-world

// Push the tagged "hello-world" image to docker-local
docker push art-local:5001/hello-world
```

Testing With a Self-signed Certificate

To test your setup with a self-signed certificate, add the certificate to the */etc/docker/certs.d/* directory as described in the Docker documentation.

Repeat the tests above.