# **JFrog Artifactory and Docker Integration Guide**

## **Table of Contents**

1. **Introduction .......................................................................................**
2. **System Requirements ......................................................................**
3. **Required Software and Tools ...........................................................**
4. **Setting Up Artifactory .......................................................................**
   * Installation Steps .................................................................
   * Initial Configuration ..............................................................
5. **Configuring Docker to Use Artifactory** ........................................
   * Setting Up Docker Repositories in Artifactory .....................
   * Configuring Docker Clients to Use Artifactory .....................
6. **Forcepoint Proxy Configuration ......................................................**
   * Overview of Forcepoint Proxy .............................................
   * Steps to Configure Docker with Forcepoint Proxy ..............
7. **Managing Docker Images** .............................................................
   * Pushing Docker Images to Artifactory .................................
   * Pulling Docker Images from Artifactory ...............................
   * Version Control and Tagging ...............................................
8. **Security and Access Control ...........................................................**
9. **CI/CD Integration ............................................................................**
10. **Best Practices .................................................................................**
11. **Troubleshooting ...............................................................................**
12. **Conclusion ......................................................................................**

## 

## 

## 

## **1. Introduction**

This guide provides step-by-step instructions for setting up and using JFrog Artifactory as a Docker registry. It covers creating local, remote, and virtual Docker repositories, configuring Docker clients, integrating with Forcepoint Proxy, and managing Docker images. Additionally, it includes best practices and troubleshooting tips.

## **2. System Requirements.**

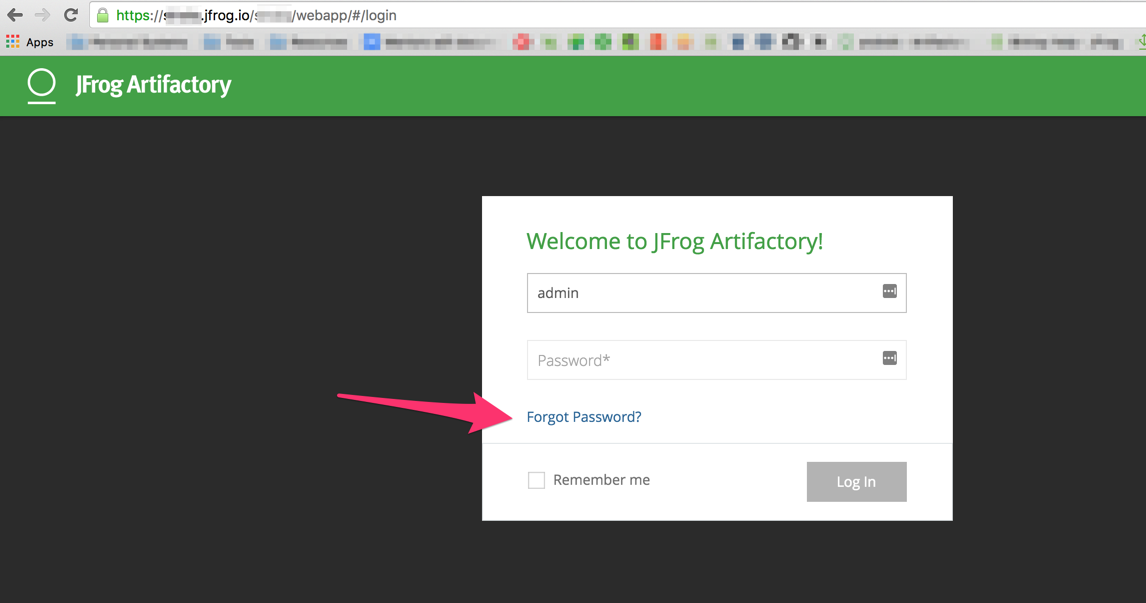
* **CPU**: Minimum 4 cores (Recommended 8+ cores).
* **RAM**: Minimum 8GB (Recommended 16GB+).
* **Storage**: Minimum 100GB (Recommended 500GB+, depends on artifact size).
* **Disk Type**: SSD (Recommended NVMe for high performance).
* Access to a JFrog Artifactory instance.
* Docker installed on your local machine or CI/CD environment.
* Network access to DockerHub and JFrog Artifactory.
* Credentials for JFrog Artifactory (retrieved via CyberArk if applicable).

## **3. Required Software and Tools**

* JFrog Artifactory: For managing Docker repositories.
* Docker: For building, pushing, and pulling Docker images.
* CyberArk: For securely retrieving JFrog credentials (if applicable).
* Forcepoint Proxy: For network-level proxy configuration (if applicable).

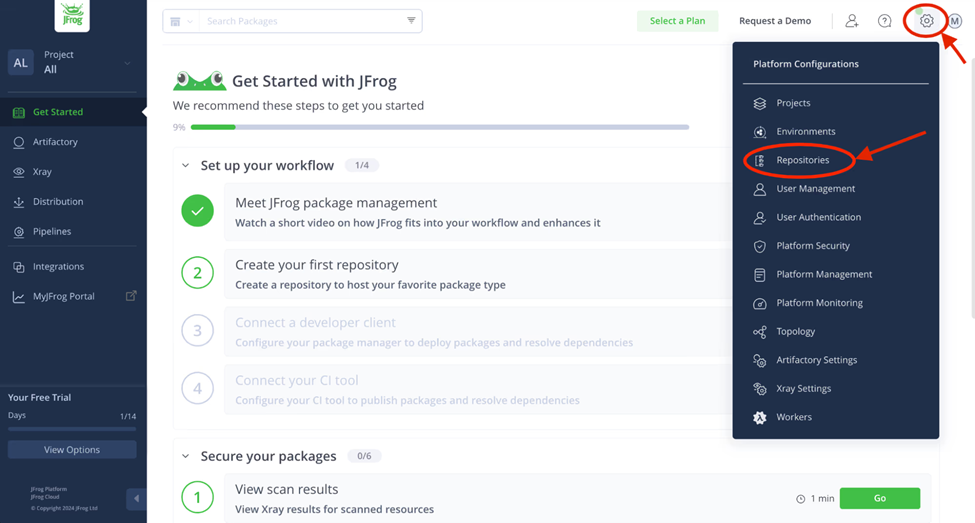
## **4. Setting Up Artifactory**

**Installation Steps**

1. **Download JFrog Artifactory:**
   * Go to the JFrog website (jfrog.com) and navigate to the Artifactory download page.
   * Choose the appropriate version (e.g., Community Edition, Pro, Enterprise) and operating system.
   * Select the installation package type (ZIP, installer, etc.). A ZIP file is common for manual installations.
2. **Install JFrog Artifactory:**
   * **ZIP Installation:** Extract the ZIP file to a directory of your choice (e.g., /opt/artifactory on Linux, C:\artifactory on Windows). This will be your Artifactory home directory.
   * **Installer Installation:** Run the installer and follow the on-screen instructions. The installer will typically ask you to choose an installation directory.
3. **Start JFrog Artifactory:**
   * Navigate to the bin directory within your Artifactory installation directory.
   * Run the appropriate startup script:
     + **Linux/macOS:** ./artifactory.sh start (or sh artifactory.sh start)
     + **Windows:** artifactory.bat start
4. **Configure Initial Admin Credentials:**
   * Open a web browser and go to **http://<your-server-ip>:<port>**. Replace **<your-server-ip>** with the IP address of the machine where you installed Artifactory. The default port is usually 8081. If Artifactory is running on your local machine, you can use **http://localhost:8081.**
   * You should see the Artifactory welcome screen.
   * Enter a username **(usually admin).**
   * Enter a strong password and confirm it. **Remember this password!**
   * Click **"Save & Finish"** or the equivalent button.
   * ****

**Initial Configuration**

1. **Log in to Artifactory:**
   * After setting the admin credentials, you should be redirected to the Artifactory login page.
   * Enter the admin username and the password you just set.
   * Click "Log In."
2. **Navigate to Platform Configurations:**
   * Once logged in, you'll be in the Artifactory web UI.
   * Look for an "Admin" or "Administration" section. It's often in the top right corner or in a side menu.
   * Click on the "Admin" section.
   * In the Admin section, find a link or menu item labeled "Platform Configuration," "General Configuration," or something similar. The exact wording might vary slightly depending on the Artifactory version.
   * Click on it.
3. **Select Repositories from the menu:**
   * In the Platform Configuration section, you'll see a menu or list of configuration options.
   * Find and click on "Repositories." This will take you to the repository management page.

****

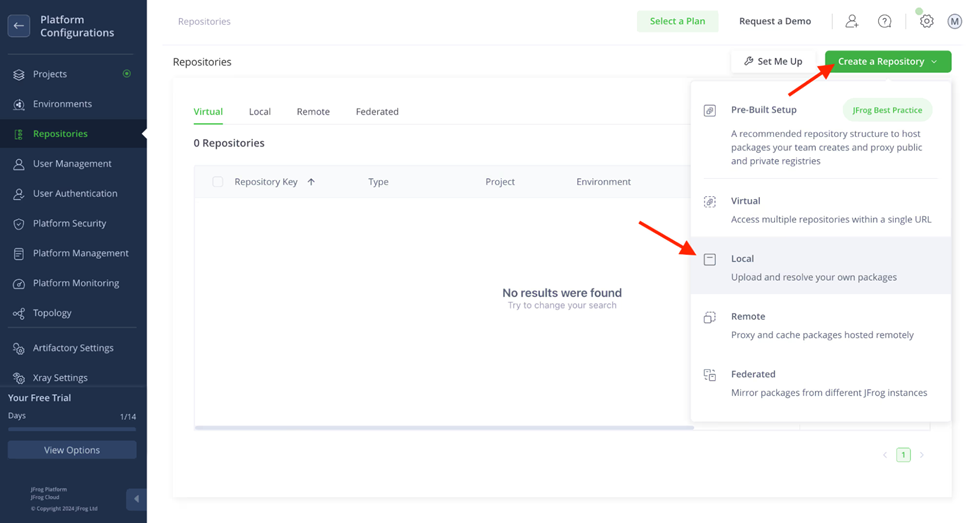
( These enhanced steps provide a more complete guide. Remember to replace placeholders like <your-server-ip> and <port> with your actual values. )

## **5. Configuring Docker to Use Artifactory**

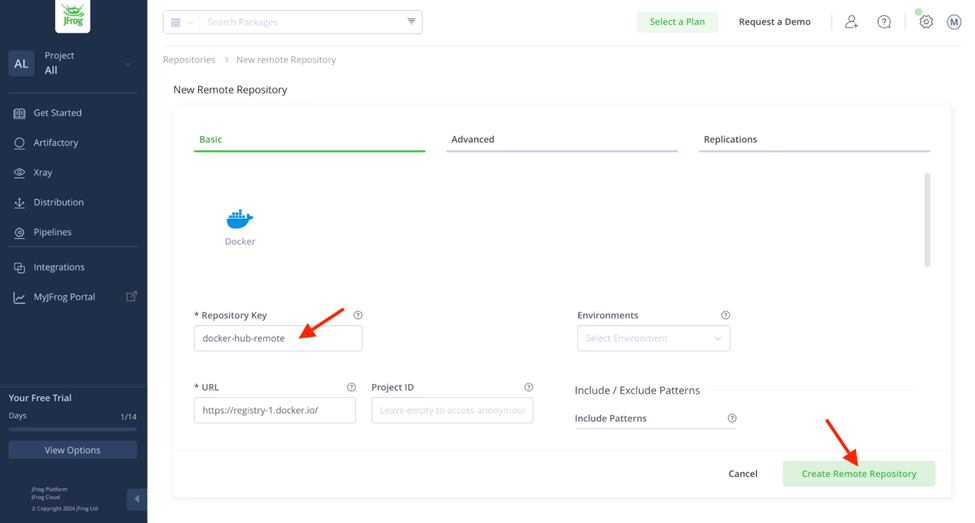
#### **Add Local Docker Repository**

Used to store your custom Docker image you will create in a later step.

Expand the Create a Repository menu and select the Local menu item. You will be presented with a number of different choices for a package type. Select the Docker package type.

****

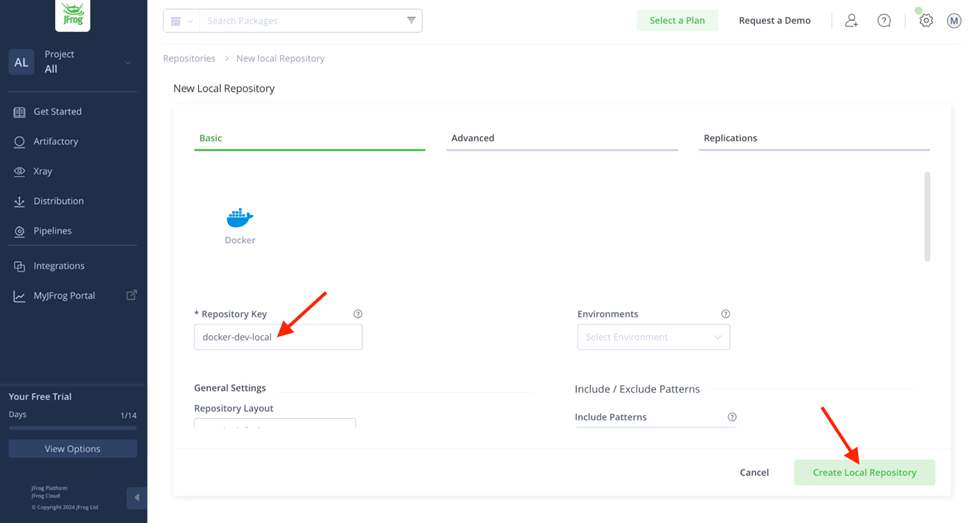
Enter the Repository Key “docker-dev-local” and keep the rest of the default settings. Click the Create Local Repository button.



#### **Add Remote Docker Repository**

Used as a caching proxy, to store 3rd party images from Docker Hub or any other external registries.

As in the previous step, expand the Create a Repository menu, but this time select the Remote menu item. Again, select the Docker package type. Enter the Repository Key “docker-hub-remote” and keep the rest of the default settings.



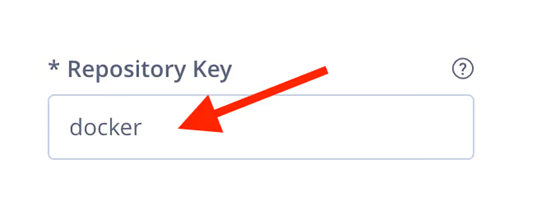
You should now see your new remote Docker repository in the Remote repository list.

#### **Add Virtual Docker Repository**

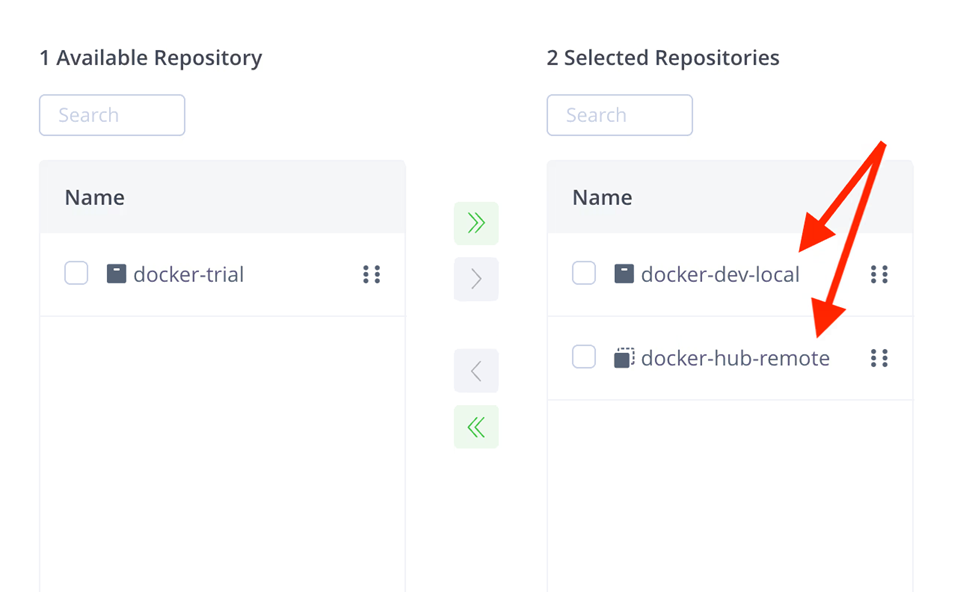
Used when creating your custom Docker image. This repository will be set up so that you can push to your local Docker repository and pull from either your local or remote Docker repository.

Expand the Create a Repository menu and select the Virtual menu item. Select the Docker package type. There are three things you will need to do here:

Enter the Repository Key “docker”



2. Scroll down the settings page and add the local and remote docker repositories you created in Steps 2 and 3 (move them from Available Repositories to Selected Repositories using the arrow buttons). The order of these repositories in the list will determine the order used to resolve the dependencies required for building your docker image.



3. Select your local repository that you created in Step 2 as the Default Deployment Repository. The Default Deployment Repository is the repository that the docker image you build will be pushed to. Keep the rest of the default settings.



Click the Create Virtual Repository button and skip through the client setup prompt. You should now see your new virtual Docker repository in the Virtual repository list.

### **Configuring Docker Clients to Use Artifactory**

1. Update Docker’s config file (~/.docker/config.json):

{

"auths": {

"https://<your-jfrog-repo>.jfrog.io": {

"auth": "<base64-encoded-username:api-key>"

}

}

}

1. Run *docker login https://<your-jfrog-repo>.jfrog.io* to verify.

## **6. Forcepoint Proxy Configuration**

**Overview of Forcepoint Proxy**

Forcepoint proxy acts as an intermediary between Docker and the outside world (like Docker Hub or your private registries). It enhances security by:

* Content Filtering: Inspecting and blocking potentially malicious images or downloads.
* Authentication and Authorization: Controlling which users or systems can access Docker registries.
* Caching: Storing frequently accessed images to improve performance.
* Logging and Reporting: Tracking Docker activity for auditing and analysis.

**Steps to Configure Forcepoint Proxy for Docker**

* Get the Forcepoint Proxy URL and port from your network administrator.
* If authentication is required, obtain the username and password.
* Open a terminal & Edit the shell configuration file.
* Set Environment Variables : For Linux

export HTTP\_PROXY=http://<proxy-url>:<port>

export HTTPS\_PROXY=http://<proxy-url>:<port>

export NO\_PROXY=localhost,127.0.0.1,.local

* If authentication required then

export HTTP\_PROXY=http://<username>:<password>@<proxy-url>:<port>

export HTTPS\_PROXY=http://<username>:<password>@<proxy-url>:<port>

* Configure Docker Daemon Proxy Settings

sudo nano /etc/systemd/system/docker.service.d/http-proxy.conf

Add these lines :

[Service]

Environment="HTTP\_PROXY=http://<forcepoint-proxy-url>:<port>"

Environment="HTTPS\_PROXY=http://<forcepoint-proxy-url>:<port>"

Environment="NO\_PROXY=localhost,127.0.0.1,.local"

* Reload the Docker daemon and restart Docker:
* Docker Desktop (Windows/Mac):

Open Docker Desktop.

Go to Settings > Resources > Proxies.

Enable Manual proxy configuration.

Enter the Forcepoint Proxy details:

HTTP Proxy: http://<forcepoint-proxy-url>:<port>

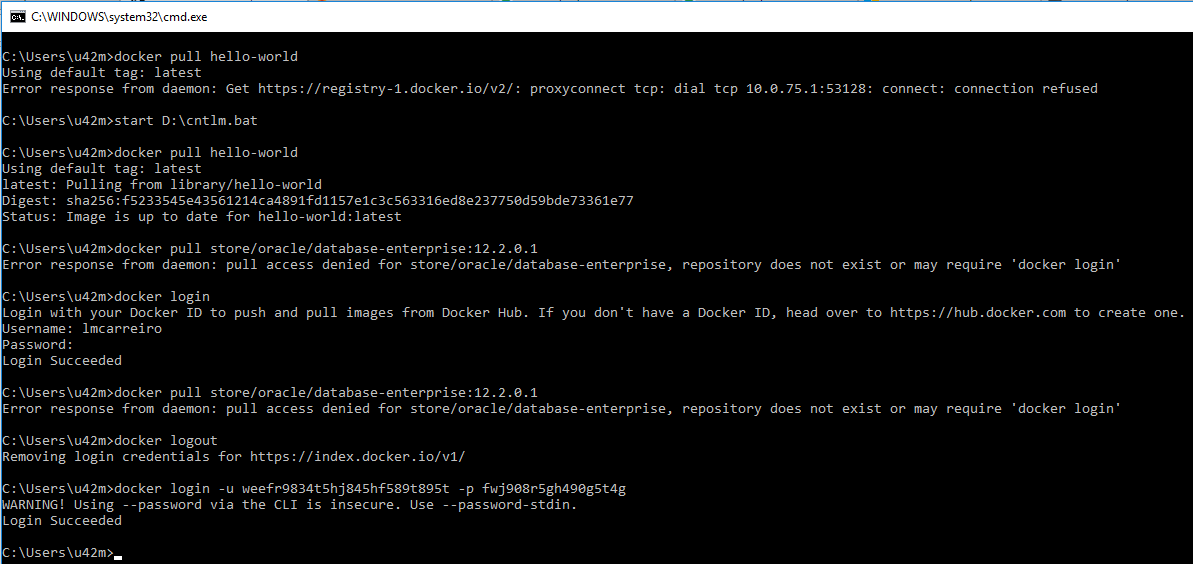
HTTPS Proxy: http://<forcepoint-proxy-url>:<port>

No Proxy: localhost,127.0.0.1,.local

Click Apply & Restart.

⇒ Verify Proxy Configuration

docker pull hello-world



## **7. Managing Docker Images**

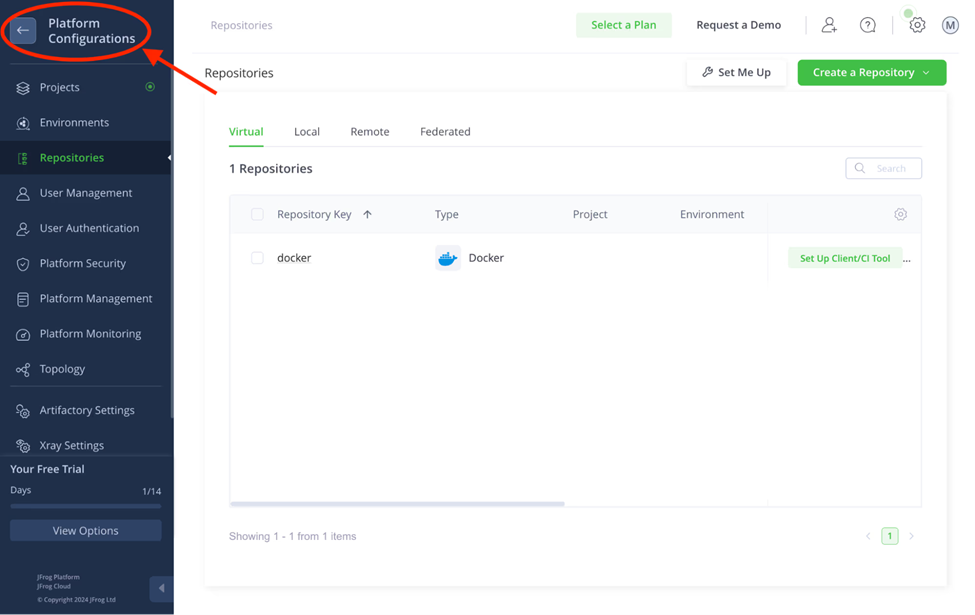
### **Pushing Docker Images to Artifactory**

Using either your IDE or your terminal, login to your virtual repository, build, tag and push your custom image with the following commands (as in Step 6, use your SERVER\_NAME and VIRTUAL\_REPO\_NAME):

> docker login SERVER\_NAME.jfrog.io

> docker build --tag SERVER\_NAME.jfrog.io/VIRTUAL\_REPO\_NAME/my-docker-image:latest .

> docker push SERVER\_[NAME.jfrog.io/VIRTUAL\_REPO\_NAME/my-docker-image:latest](http://name.jfrog.io/VIRTUAL_REPO_NAME/my-docker-image:latest)

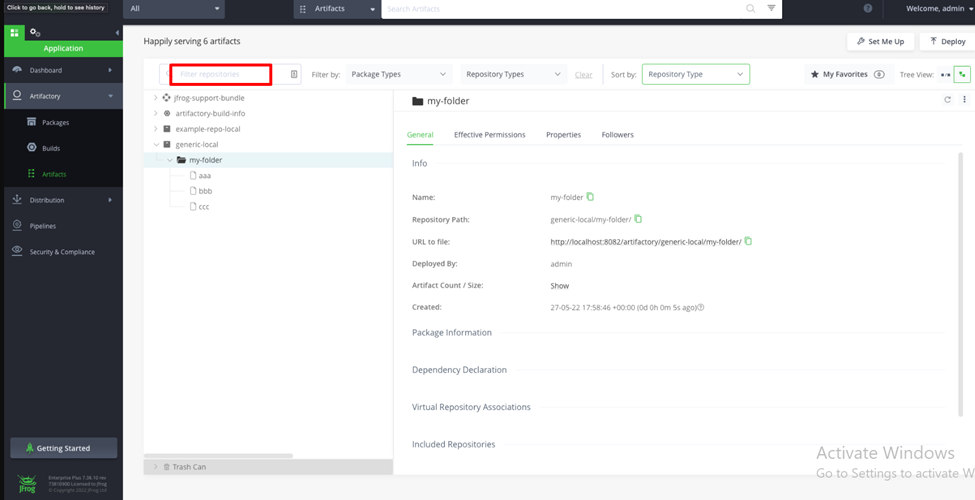


### **Pulling Docker Images from Artifactory**

Select and copy repo name,

Pull the Image : docker pull <your-jfrog-repo>.jfrog.io/<dockerhub-image>:<tag>

(Replace <your-jfrog-repo> with your JFrog repository URL and <dockerhub-image>:<tag> with the DockerHub image name and tag.)



### **Version Control and Tagging**

Use semantic versioning (e.g., v1.0.0, latest) to tag your Docker images.

Push and pull specific versions as needed.

## **8. Security and Access Control**

**1. API Keys and Tokens**

API keys and access tokens provide a way to authenticate to Artifactory programmatically without using your regular username and password. This is crucial for automation, CI/CD pipelines, and any other situation where you don't want to expose your credentials.

* **Generating API Keys (Older Artifactory Versions):**
  + Log in to Artifactory.
  + Go to your user profile (usually by clicking your username in the top right).
  + Look for "API Key" or "Generate API Key."
  + Artifactory will generate a key that you can then copy and use. **Store this key securely!** You won't be able to see it again.
* **Generating Access Tokens (Recommended - Newer Artifactory Versions):**
* Log in to Artifactory.
* Go to your user profile.
* Look for "Access Tokens" or "Generate Access Token."
* You can typically specify the scope (permissions) and expiry time for the token. This is a significant advantage over API keys, as you can grant more granular access and revoke tokens when necessary.
* Store the generated token securely!
* **Using API Keys/Tokens:**
* API keys and access tokens are typically used in the Authorization header of HTTP requests:  
   Authorization: Bearer <your-api-key-or-access-token>
* For Docker, you would use the token as the password when logging in:  
   Bash  
  docker login <your-artifactory-server>:<port> -u <your-username> -p <your-access-token>

(You can use any username, but it is best practice to use the username the token was generated for)

**2. Role-Based Access Control (RBAC)**

RBAC allows you to define roles with specific permissions and then assign those roles to users or groups. This makes it much easier to manage access to Artifactory's features and repositories.

* **Defining Roles:**
  + In Artifactory, go to "Admin" -> "Security" -> "Users & Groups" -> "Roles."
  + Create new roles (e.g., "Developer," "Deployer," "Reader").
  + Assign permissions to each role. Permissions can include:
    - Access to specific repositories (read, write, delete).
    - Management of repositories.
    - System administration tasks.
  + Artifactory provides a wide range of granular permissions.
* **Assigning Roles to Users and Groups:**
  + Go to "Admin" -> "Security" -> "Users & Groups" -> "Users" or "Groups."
  + Edit a user or group and assign the appropriate roles.

## **9. CI/CD Integration**

* Integrate JFrog Artifactory with CI/CD tools like Jenkins, GitLab, or GitHub Actions.
* Use the JFrog CLI to automate Docker image builds, pushes, and pulls.

## **10. Best Practices**

* Use virtual repositories to simplify Docker image management.
* Regularly scan images for vulnerabilities using JFrog Xray.
* Use semantic versioning for Docker image tags.

## **11. Troubleshooting**

* Docker Login Fails: Verify your credentials and ensure the ~/.docker/config.json file is correctly configured.
* Proxy Issues: Check your Forcepoint Proxy settings and environment variables.
* Image Push/Pull Fails: Ensure the repository URL and permissions are correct.

## **12. Conclusion**

This guide provides a comprehensive overview of setting up and using JFrog Artifactory as a Docker registry. By following these steps, you can efficiently manage Docker images, integrate with CI/CD pipelines, and ensure secure access using CyberArk and Forcepoint Proxy.