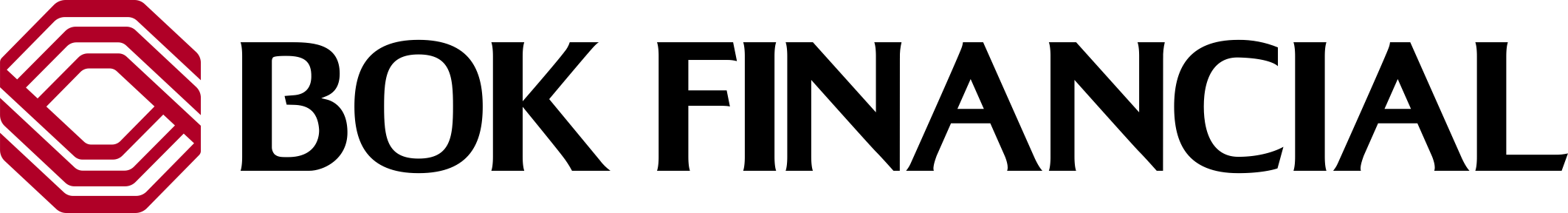
Docker Integration with JFrog Artifactory Guide

Document Subtitle



**REVISION RECORD**

|  |  |  |
| --- | --- | --- |
| **Revision Description** | **Author** | **Date** |
| Initial Draft | Sogo Alonge | 02/28/2025 |
|  |  |  |

Table Of Contents

1. [Introduction 3](#_Toc191388670)
   1. [System Requirements 4](#_Toc191388671)
   2. [Required Software And Tools 4](#_Toc191388672)
2. [Setting Up Artifactory 4](#_Toc191388673)
   1. [Installation Steps 4](#_Toc191388674)
   2. [Initial Configuration 4](#_Toc191388675)
3. [Configuring Docker To Use Artifactory 4](#_Toc191388676)
   1. [Setting Up Docker Repositories In Artifactory 4](#_Toc191388677)
   2. [Configuring Docker Clients To Use Artifactory 4](#_Toc191388678)
4. [Forcepoint Proxy Configuration 4](#_Toc191388679)
   1. [Overview Of Forcepoint Proxy 4](#_Toc191388680)
   2. [Steps To Configure Docker With Forcepoint Proxy 4](#_Toc191388681)
5. [Managing Docker Images 4](#_Toc191388682)
   1. [Pushing Docker Images To Artifactory 4](#_Toc191388683)
   2. [Pulling Docker Images From Artifactory 4](#_Toc191388684)
   3. [Version Control And Tagging 4](#_Toc191388685)
6. [Security And Access Control 4](#_Toc191388686)
7. [CI/CD Integration 4](#_Toc191388687)
8. [Best Practices 4](#_Toc191388688)
9. [Troubleshooting 4](#_Toc191388689)
10. [Conclusion 5](#_Toc191388690)

# Introduction

The Docker Integration with JFrog Artifactory Guide explains how to integrate Docker, a platform for building and distributing containerized applications, with JFrog Artifactory, a powerful repository manager. By connecting Docker to Artifatcory, you can centralize Docker image storage, manage image versions, and enforce access control to ensure secure distribution.

This guide provides detailed step-by-step instructions for setting up and configuring JFrog Artifactory as a Docker registry. It covers the following setups:

* **Self-hosted JFrog Artifactory**: Instructions for configuring and managing Artifactory when running it locally on your company laptop.
* **Cloud-based JFrog Artifactory**: Instructions for configuring Artifactory in the cloud environment, accessed via Okta Single Sign-On (SSO) for seamless authentication.

You will learn how to:

* Create and manage local, remote, and virtual Docker repositories.
* Set up Docker clients to authenticate with Artifactory.
* Integrate Artifactory with your CI/CD pipelines to automate Docker image builds and deployments.
* Secure access with API keys, access tokens, Role-Based Access Control (RBAC), and Forcepoint Proxy.
* Whether you're using a self-hosted or cloud-based setup, this guide will ensure you can effectively manage Docker images, integrate with your workflows, and maintain secure access.
* Additionally, this guide includes security best practices to ensure integrity of your artifacts, and troubleshooting tips to resolve common issues.

Prerequisites

## System Requirements

* **CPU**: Minimum 4 cores (Recommended: 8+ cores for optimal performance).
* **RAM**: Minimum 8GB (Recommended: 16GB+ for larger builds or more concurrent tasks).
* **Storage**: Minimum 100GB (Recommended: 500GB+, depending on artifact size and usage).
* **Disk Type**: SSD (Recommended: NVMe for better performance, especially with large Docker images).
* **Access**: Access to a JFrog Artifactory instance.
* **Docker**: Docker installed on your local machine or within your CI/CD environment.
* **Network**: Network access to DockerHub and JFrog Artifactory for image pulling/pushing.
* **Credentials**: Valid credentials for JFrog Artifactory (can be retrieved via CyberArk if applicable).

## Required Software And Tools

* **JFrog Artifactory**: A repository manager for managing Docker images and other artifacts.
* **Docker**: A containerization platform for building, pushing, and pulling Docker images.
* **CyberArk**: Used for securely managing and retrieving JFrog Artifactory credentials (if applicable).
* **Forcepoint Proxy**: Used for network-level proxy configuration when accessing JFrog Artifactory through a proxy (if applicable).

# Setting Up Artifactory

**Self-Hosted JFrog Artifactory Integration**

## Installation Steps

1. **Download JFrog Artifactory:**
   * Go to the [JFrog website](https://www.googleadservices.com/pagead/aclk?sa=L&ai=DChcSEwiVyOTK7uWLAxWsRv8BHRpDCtAYABAAGgJtZA&co=1&gclid=EAIaIQobChMIlcjkyu7liwMVrEb_AR0aQwrQEAAYASAAEgKCVfD_BwE&ohost=www.google.com&cid=CAASEuRoAop_CS87C3vFvBC-TMkPaw&sig=AOD64_2vY998FCPSTnjWg2aMwUXrgD1lkQ&q&adurl&ved=2ahUKEwjIjt7K7uWLAxWrnokEHVcROBMQ0Qx6BAgKEAE) (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. Make sure to store or **remember this password.**
   * Click **"Save & Finish"** or the equivalent button to complete the setup.

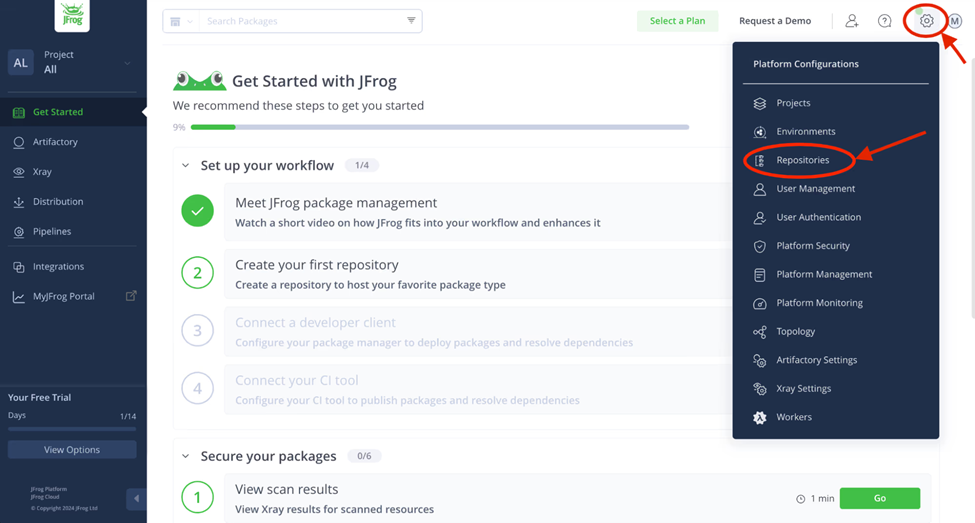
**A screenshot of a computer

Description automatically generated**

## Initial Configuration

1. **Log in to Artifactory:**
   * After setting your admin credentials, you should be redirected to the Artifactory login page.
   * Enter the **admin** username and the password you just set.
   * Click "**Log In**” to access the Artifactory web UI.
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 usually located in the top right corner or within a side menu.
   * Click on the "**Admin**" section to open the administration panel.
   * In the **Admin** section, locate 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 this option to proceed to the configuration settings
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 where you can configure Docker and other repositiories as in the screenshot below.

****

**Cloud-Based JFrog Artifactory Integration via Okta**

Login to BOKF Cloud Environment via Okta

**Access JFrog Artifactory via Okta**:

* Navigate to the Okta dashboard.
* **Log in to Okta** using your BOKF credentials.
* Once logged in, locate the JFrog Artifactory application in the Okta menu or dashboard.
* Click on the JFrog Artifactory application to be redirected to yBOKF cloud-based Artifactory instance. This will automatically log you in using Single Sign-On (SSO).

**Verify Login**:

* After clicking on the Artifactory application, you should be taken to the JFrog Artifactory dashboard without needing to manually enter a username or password, as Okta manages your credentials.
* If prompted, confirm that you have access to the necessary repositories and tools within Artifactory.

**Navigate to Platform Configurations**

* After logging in to JFrog Artifactory, you will be directed to the cloud-based Artifactory dashboard.
* Click the gear icon (settings) in the top-left corner to open the Platform Configurations screen.
* In the Platform Configurations menu, select the Repositories option to access the repository management page.

**Working with Cloud-Based Artifactory**

From this point, you can configure repositories and manage Docker images as you would with a self-hosted instance.

The main difference is that you will use BOKF cloud-based Artifactory URL, which will look something like:

https://<company-name>.jfrog.io/artifactory/

# Configuring Docker To Use Artifactory

#### ~~Setting Up Docker Repositories In JFrog Artifactory~~

#### Add Local Docker Repository

This repository will store your custom Docker images that you will create in a later step.

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

## A screenshot of a computer Description automatically generated

Enter the Repository Key as "docker-dev-local" and leave the other settings as default.

Click the "Create Local Repository" button to complete this step.

A screenshot of a computer

Description automatically generated

## You should now see your new local Docker repository in the Local repository list.

#### Add Remote Docker Repository

This repository will act as a caching proxy for storing third-party Docker images, such as those from Docker Hub or other external registries.

## Similar to the previous step, expand the Create a Repository menu, but this time select the Remote menu item. Again, choose the Docker package type.

## Enter the Repository Key “docker-hub-remote” and keep the rest of the default settings.

## A screenshot of a computer Description automatically generated

After creation, you should now see your new remote Docker repository in the Remote repository list as in the above screenshot.

#### Add Virtual Docker Repository

This repository will be set up to allow you to push your custom Docker images to your local repository and pull from either the local or remote repositories.

Expand the **Create a Repository** menu and select the **Virtual** menu item. Select the Docker package type.

You will need to do the following:

1. **Enter the Repository Key** as “**docker**”.

A red arrow pointing to a key

Description automatically generated

1. Scroll down the settings page to the **Available Repositories** and select the local (**docker-dev-local**) and remote (**docker-hub-remote**) Docker repositories (from steps 2 and 3) to the list. The order of these repositories in the list will determine the order used to resolve the dependencies required for building your Docker image.

Screens screenshot of a computer

Description automatically generated

1. Set the **Default Deployment Repository**: Select the local repository you created in step 2 (i.e., docker-dev-local) as the Default Deployment Repository. This repository will receive the Docker images you push. Keep the rest of the default settings.

A close-up of a computer screen

Description automatically generated

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

Configuring Docker clients to use JFrog Artifactory requires setting up authentication so that Docker client can push and pull images from Artifactory. The below steps shows how to configure Docker client to use JFrog artifactory.

## Update Docker’s configuration File (~/.docker/config.json):

* Open or create the Docker config file located at **~/.docker/config.json** on your local machine.
* Add the following JSON snippet to configure Docker to authenticate with your Artifactory instance. Replace **<your-jfrog-repo**> with your JFrog Artifactory domain and <**base64-encoded-username:api-key**> with your **base64-encoded** Artfiactory credentials (username and API key) as in the below.

## {

## "auths": {

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

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

## }

## }

## }

Note: You can generate the base64-encoded credentials by using the following format.

*echo -n “username:api-key” | base64*

## Verify Authentication:

## After updating the config file, verify that Docker can authenticate with Artifactory by running the following command:

## *docker login https://<your-jfrog-repo>.jfrog.io*

# If everything is set up correctly, you should receive successful login message from Docker, confirming the connection to your Artifactory instance.

# Forcepoint Proxy Configuration

**Overview of Forcepoint Proxy**

The **Forcepoint Proxy** acts as an intermediary between between Docker and external services, such as Docker Hub or private registries. It enhances security and performance by offering the following benefits:

* **Content Filtering**: Inspects and blocks 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**

* + - 1. **Obtain Force Point Proxy Information**
* Get the Forcepoint Proxy URL and port from your network administrator.
* If proxy authentication is required, obtain the **username** and **password**.

1. **Set Environment Variables (for Linux):**

* Open a terminal & edit your shell configuration file(e.g., **~bashrc** or **~/.bash\_profile**.
* Add the following lines to set the proxy environment variables:

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

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

export NO\_PROXY=localhost,127.0.0.1,.local

* If proxy authentication is required, use the following format:

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

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

* Save the file and reload it by running:

source ~/.bashrc # or source ~/.bash\_profile

1. **Configure Docker Daemon Proxy Settings**

Create or edit the Docker daemon proxy configuration file

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

Add these following lines to configure Docker to use the Forcepoint proxy :

[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"

1. **Reload the Docker daemon and Restart Docker:**

* After saving the configuration, reload the Docker daemon and restart Docker for the changes to take effect:

sudo systemctl daemon-reload

sudo systemctl restart docker

For Docker Desktop (Windows/Mac):

1. Open Docker Desktop.
2. Go to Settings > Resources > Proxies.
3. Enable Manual proxy configuration.
4. 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

5. Click Apply & Restart to save the settings.

Verify Proxy Configuration:

To verify that Docker is correctly configured to use the Forcepoint proxy, run the following command:

docker pull hello-world

If the proxy configuration is correct, Docker will pull the hello-world image successfully.

A screenshot of a computer screen

Description automatically generated

# Managing Docker Images

**Pushing Docker Images to Artifactory**

To push a Docker image to Artifactory, you can use either your IDE or terminal. Follow these steps to build, tag, and push your custom Docker image to Artifactory:

1. **Log in to your Artifactory virtual repository**:

* Run the following command to authenticate Docker with your Artifactory instance:

docker login SERVER\_NAME.jfrog.io

Replace SERVER\_NAME with your actual Artifactory server name.

1. **Build the Docker image**:

* Use the docker build command to create your Docker image and tag it for your Artifactory repository:

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

Replace SERVER\_NAME with your Artifactory server name and VIRTUAL\_REPO\_NAME with your virtual repository name.

1. **Push the image to Artifactory**:

* Once the image is built, push it to your Artifactory repository:

docker push SERVER\_NAME.jfrog.io/VIRTUAL\_REPO\_NAME/my-docker-image:latest

This will upload the Docker image to Artifactory for use by other developers or systems.

A screenshot of a computer

Description automatically generated

**Pulling Docker Images from Artifactory**

To pull an image from Artifactory, follow these steps:

**Copy the repository name**:

* Identify the Docker image you want to pull and copy the repository name (e.g., **my-docker-image**).

**Run the docker pull command**:

* Pull the image using the following command:

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

Replace:

* <your-jfrog-repo> with your JFrog Artifactory repository URL (e.g., SERVER\_NAME.jfrog.io).
* <dockerhub-image>:<tag> with the image name and version tag from Artifactory (e.g., my-docker-image:latest).

A screenshot of a computer

Description automatically generated

**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.

It’s important to maintain consistent versioning for your Docker images. Here are some best practices:

**Use semantic versioning**: Tag your Docker images with version numbers that follow semantic versioning (e.g., v1.0.0, v1.0.1, latest).

Example tags: v1.0.0, v1.1.0, latest

**Push and pull specific versions**: You can use tags to push or pull specific versions of Docker images as needed. For example:

docker push SERVER\_NAME.jfrog.io/VIRTUAL\_REPO\_NAME/my-docker-image:v1.0.0

docker pull SERVER\_NAME.jfrog.io/VIRTUAL\_REPO\_NAME/my-docker-image:v1.0.0

# Security and Access Control

Setting up user permissions

Enabling vulnerability scanning

**1. API Keys and Tokens**

API keys and access tokens pallows for programmatic authentication to Artifactory, making them essential for automation, CI/CD pipelines, and other scenarios without exposing exposing your regular username and password.

**Generating API Keys (Older Artifactory Versions):**

1. Log in to Artifactory.
2. Go to your user profile (usually by clicking your username in the top right).
3. Look for "API Key" or "Generate API Key."
4. 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):**

1. Log in to Artifactory.
2. Go to your user profile.
3. Look for "Access Tokens" or "Generate Access Token."
4. 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.
5. 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 for whoch the token was generated.

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

RBAC allows you to define roles with specific permissions and assign them to users or groups. This provides fine-grained access management for Artifactory features and repositories.

* **Defining Roles:**
  1. In Artifactory, go to "**Admin**" -> "**Security**" -> "**Users & Groups**" -> "**Roles**."
  2. **Create new roles** (e.g., "**Developer**," "**Deployer**," "**Reader**").
  3. **Assign permissions** to each role. Permissions can include:
     + Repositories access (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.
  + This ensures that users have the necessary permissions based on their role.

# CI/CD Integration

**Integrate with CI/CD Tools**:

* Integrate **JFrog Artifactory** with popular CI/CD tools like **Jenkins**, **GitLab**, or **GitHub Actions** to automate the build, test, and deployment pipelines.
* This integration allows you to push and pull Docker images automatically based on triggers like code commits or pull requests.

**Use the JFrog CLI**:

* Leverage the **JFrog CLI** to automate Docker image operations such as building, pushing, and pulling images.
* The JFrog CLI simplifies these tasks with streamlined commands, making the CI/CD pipeline more efficient and easier to manage.

# Best Practices

To optimize your Docker image management in Artifactory, follow these best practices:

**Use Virtual Repositories**:

**Virtual repositories** simplify the management of Docker images by aggregating multiple local and remote repositories under a single URL. This reduces complexity and ensures a cleaner structure.

**Scan Images for Vulnerabilities**:

Regularly use **JFrog Xray** to scan Docker images for known vulnerabilities, ensuring that you maintain a secure environment by identifying potential threats before deployment.

**Use Semantic Versioning for Docker Image Tags**:

Adopt semantic versioning (e.g., v1.0.0, v1.1.0, latest) for your Docker image tags. This practice helps in managing versions effectively and allows for clear version control of your images.

# Troubleshooting

Common issues and solutions

Useful logs and diagnostics

* 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.

If you encounter any issues during the Docker-Artifact integration process, here are some common problems and solutions:

**Docker Login Fails**:

Problem: Docker fails to log in to Artifactory.

Solution: Verify your credentials and ensure the **~/.docker/config.json** file is correctly configured with the appropriate authentication token or API key.

**Proxy Issues**:

Problem: Unable to push or pull images due to proxy-related issues.

Solution: Check the Forcepoint Proxy settings and ensure that the environment variables (HTTP\_PROXY, HTTPS\_PROXY, NO\_PROXY) are correctly set in both the shell configuration file and Docker daemon settings.

**Image Push/Pull Fails**:

Problem: Docker fails to push or pull images from Artifactory.

Solution: Verify that the repository URL is correct and that you have the necessary permissions for the repository. Check that the repository exists and is accessible.

# 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 Artifactory with CI/CD pipelines, and ensure secure access using CyberArk for credentials management and Forcepoint Proxy for secure networking.

By implementing these practices and troubleshooting strategies, you can ensure a smooth integration and maintain an efficient and secure Docker image management workflow in your CI/CD environment.