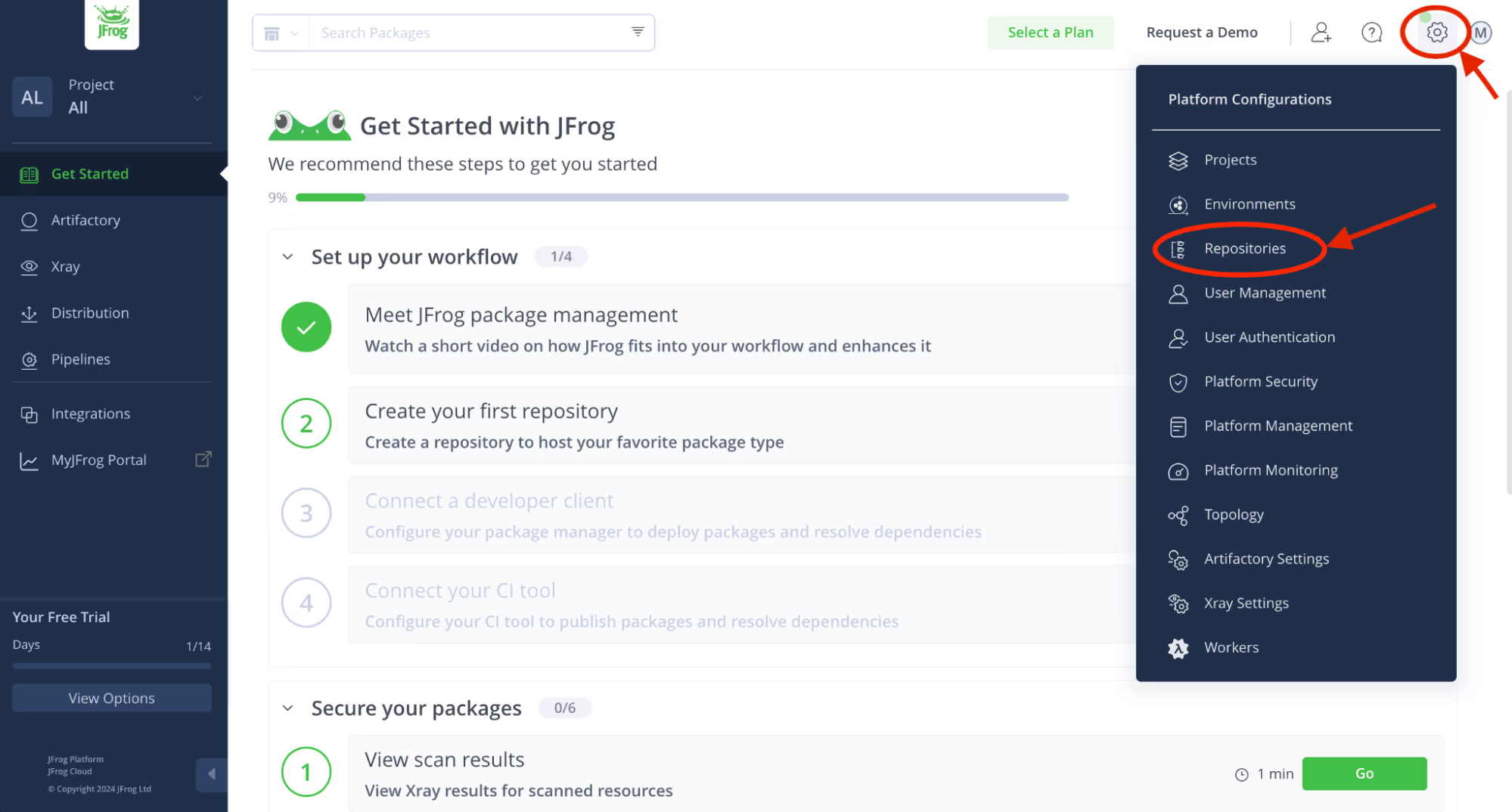
How to set up a Private, Remote and Virtual Docker Registry

## Step 1: Login to your environment and navigate to the Platform Configurations screen

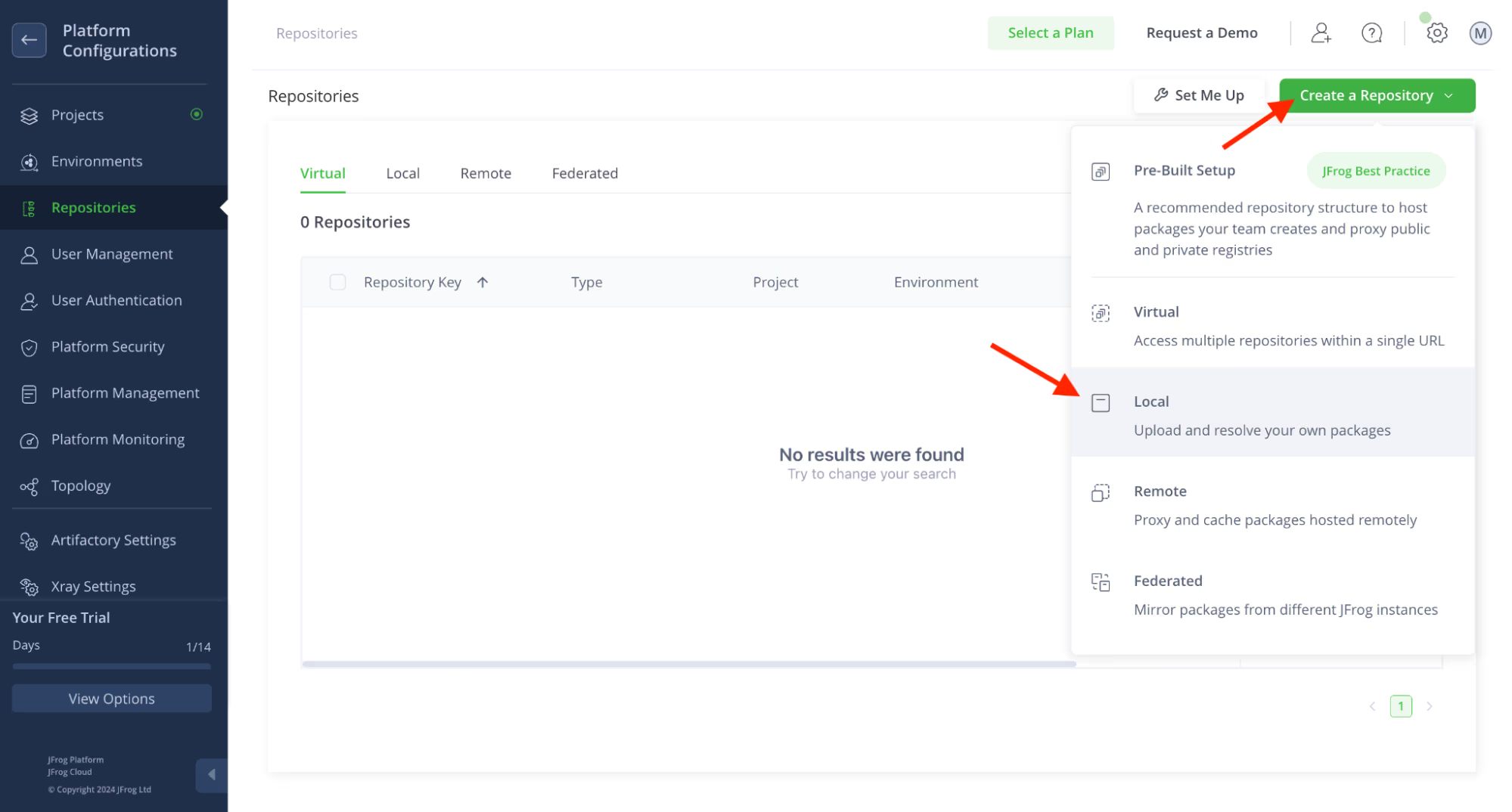
After signing up for a cloud trial, you should have received an email with your environment url and username. The password would have been created during the signup process. Login to your environment, click the Platform Configurations screen gear icon in the upper left corner, and select the *Repositories* menu item.



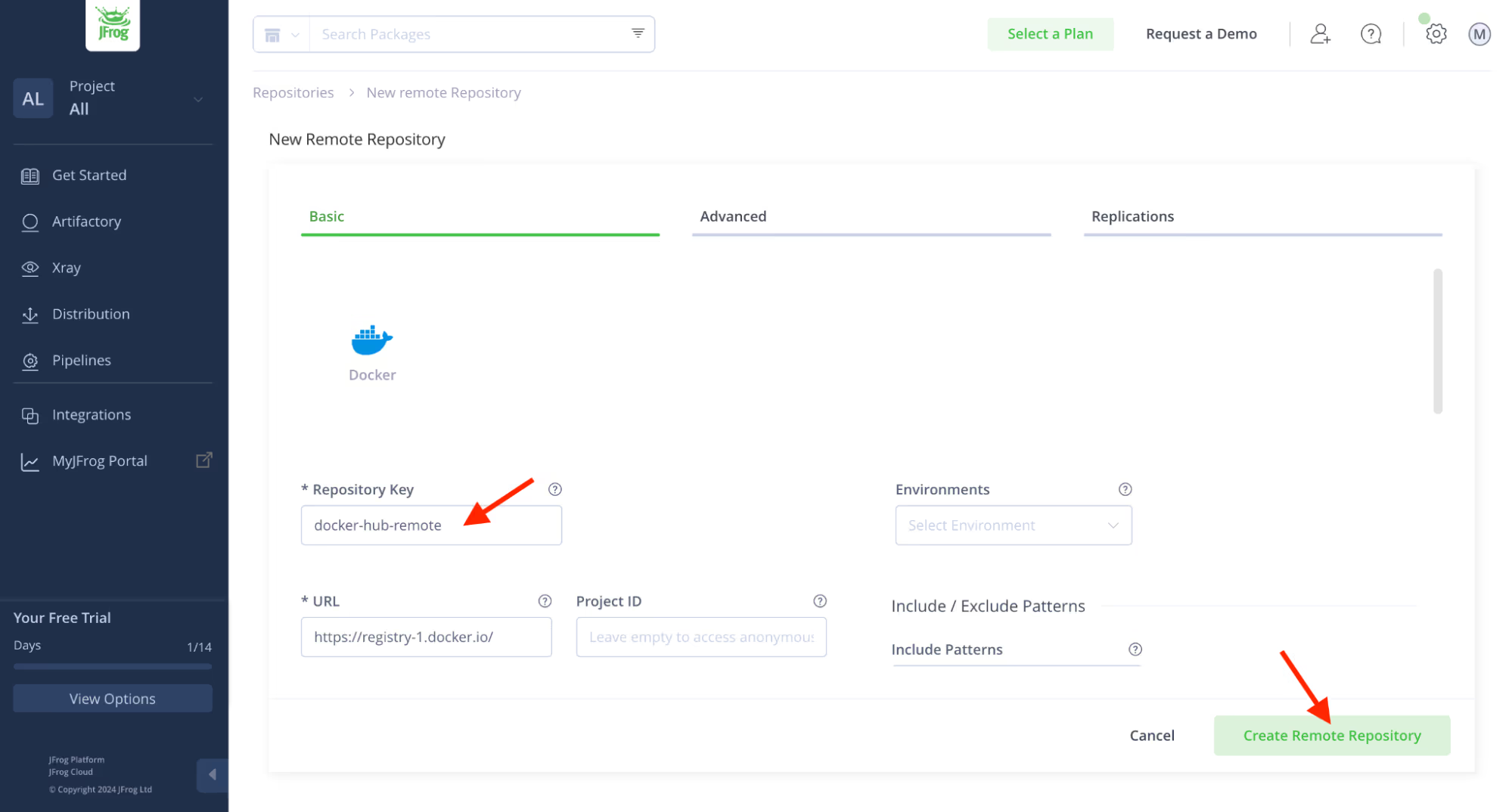
## **Step 2: Add local Docker repository**

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.

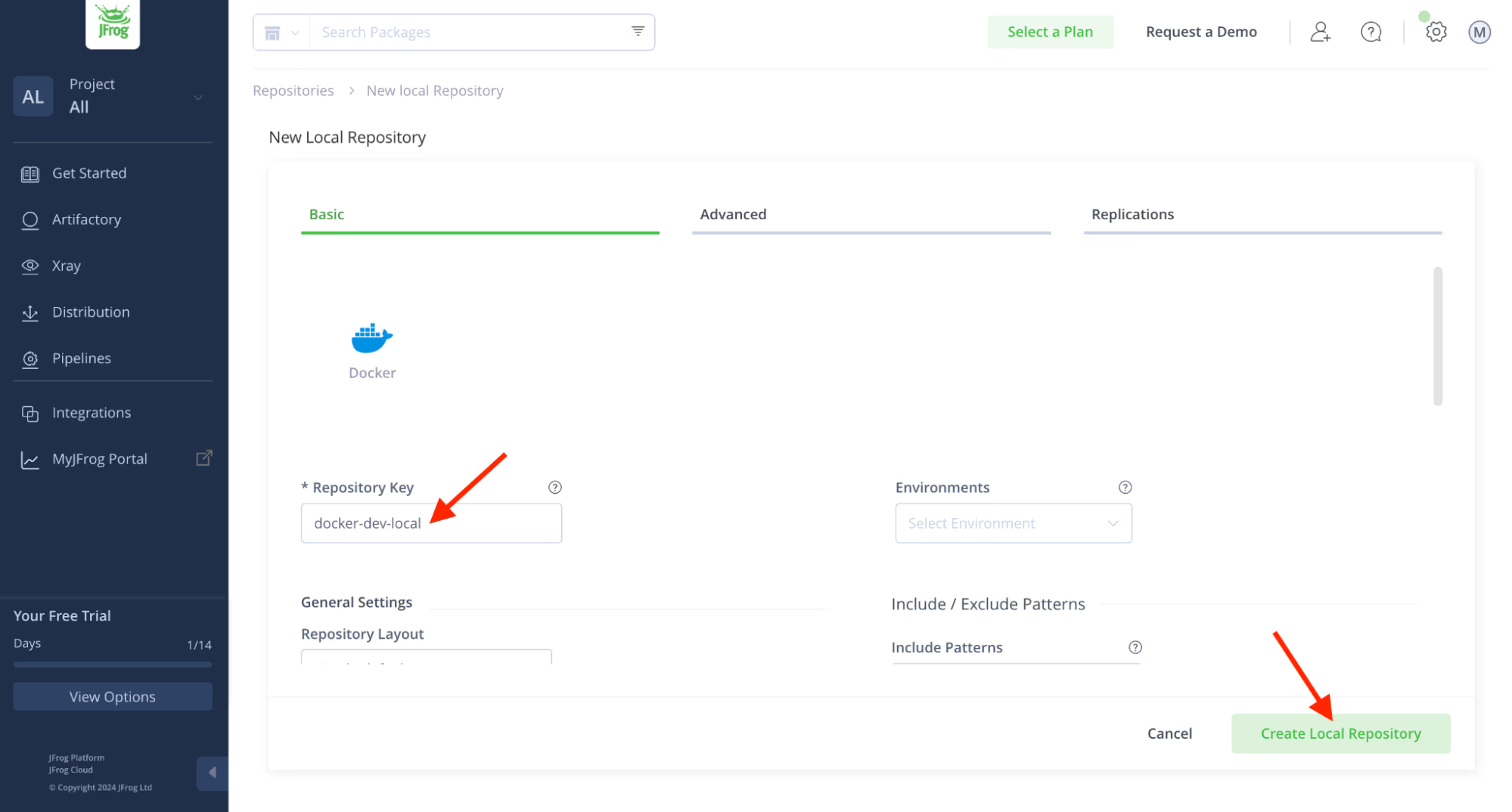


You should now see your new local Docker repository in the Local repository list.

## **Step 3: 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.



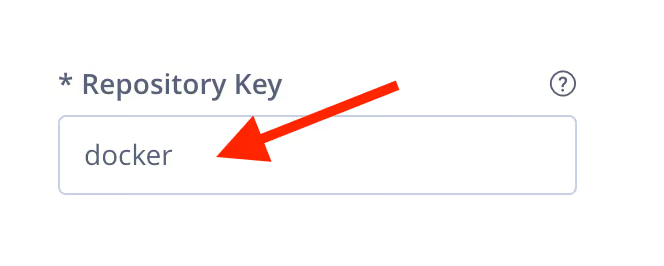
You should now see your new remote Docker repository in the Remote repository list.

## **Step 4: Add virtual Docker repository**

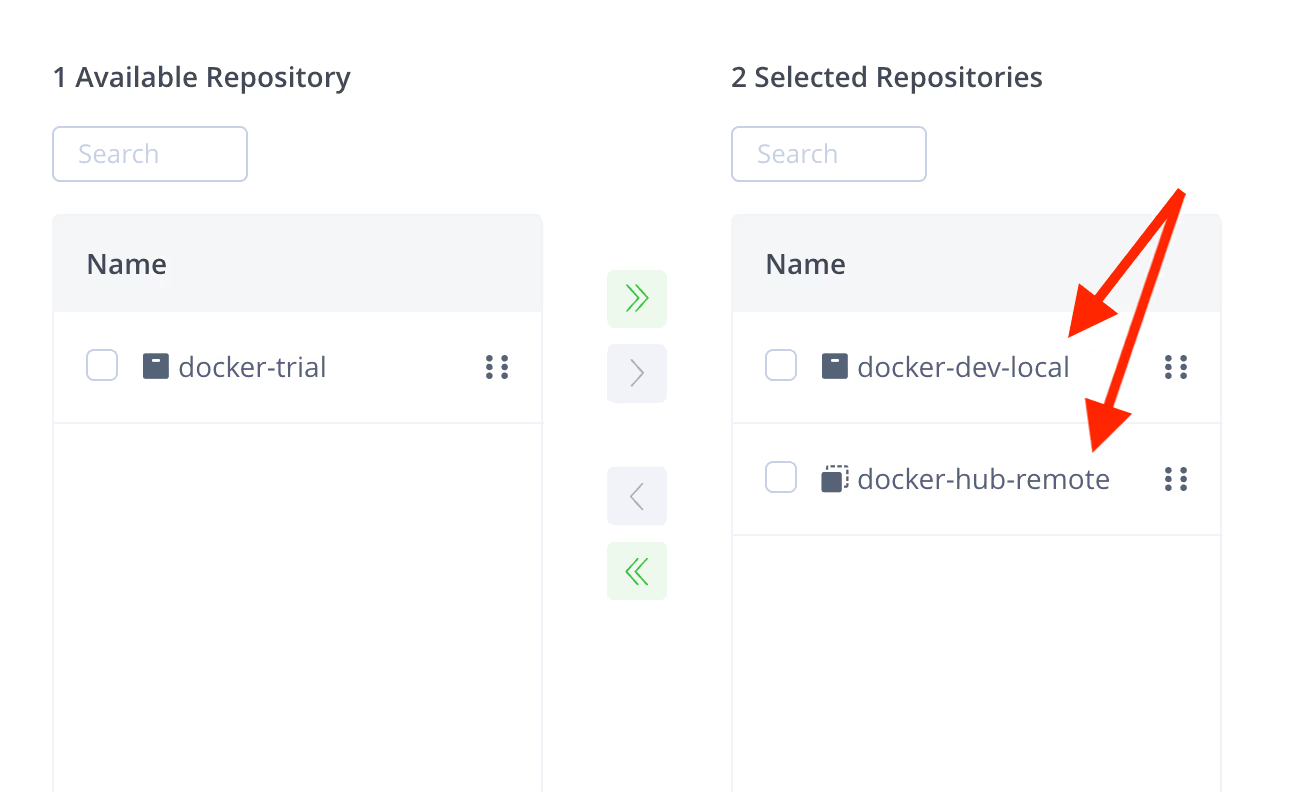
Used when creating your custom Docker image. This repository will be set up so that you can push to your local Docker repository and pull from either your local or remote Docker repository.

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:

1. 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.

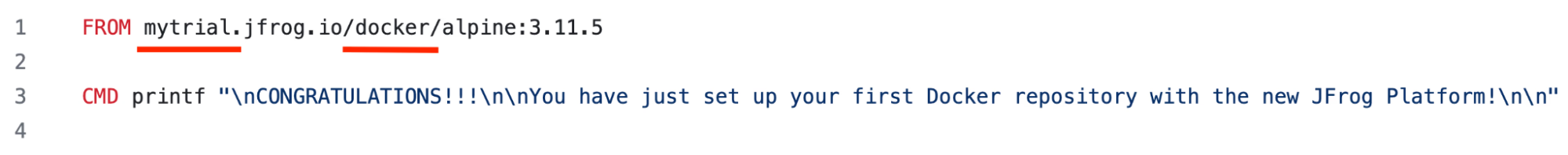
## **Step 5: Update the base image reference**

Clone your repository and using an IDE or a simple text editor on your machine, update the FROM line of the Dockerfile to reference your virtual Docker repository.

FROM ${SERVER\_NAME}.jfrog.io/${VIRTUAL\_REPO\_NAME}/alpine:3.11.5

The SERVER\_NAME is the first part of the URL given to you for your environment: *https://SERVER\_NAME.jfrog.io*

The VIRTUAL\_REPO\_NAME is the name “*docker*” that you assigned to your virtual repository in Step 4.



## **Step 6: Push custom image to your Docker repository**

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

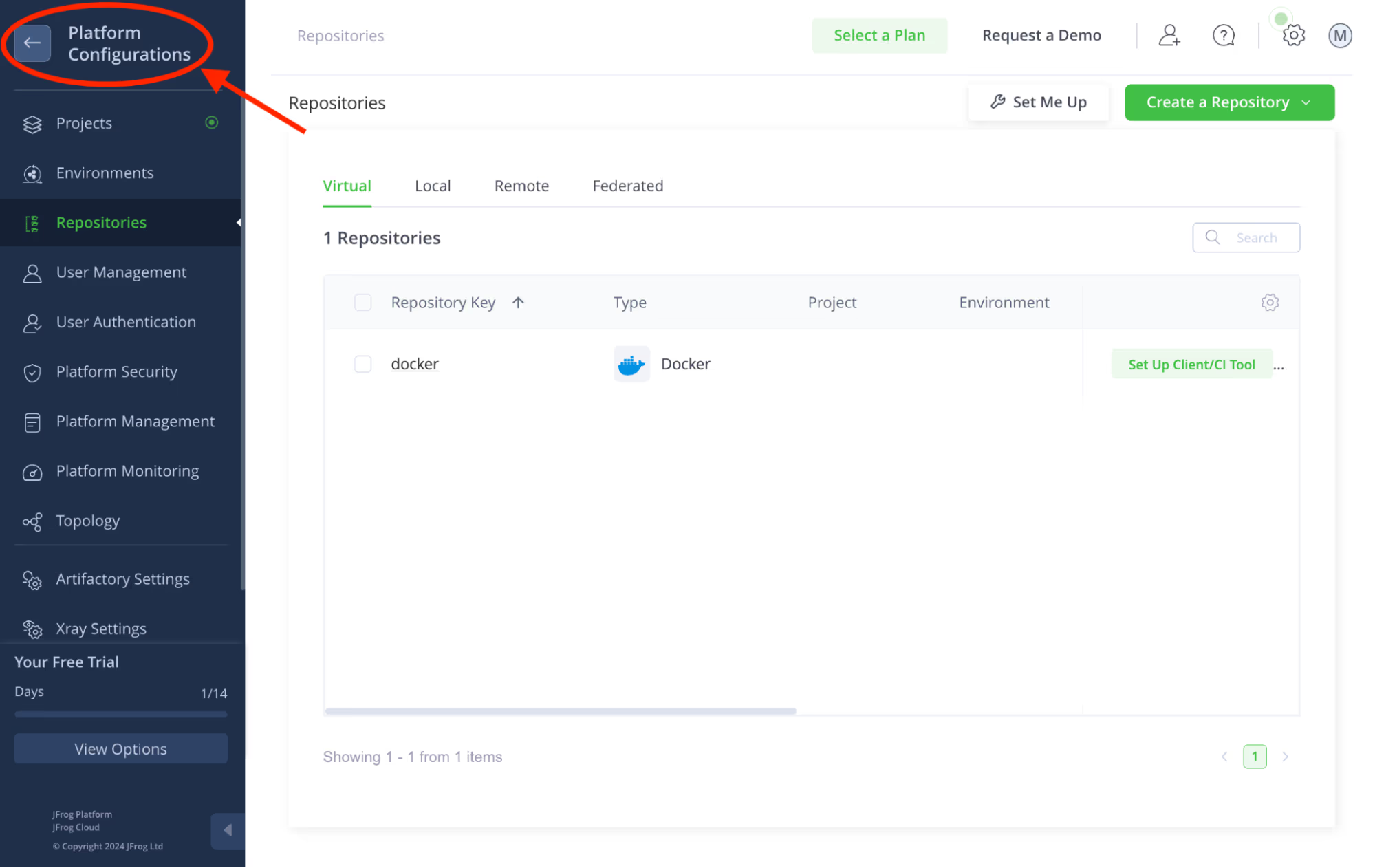
> docker build --tag SERVER\_NAME.jfrog.io/VIRTUAL\_REPO\_NAME/my-docker-image:latest .

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

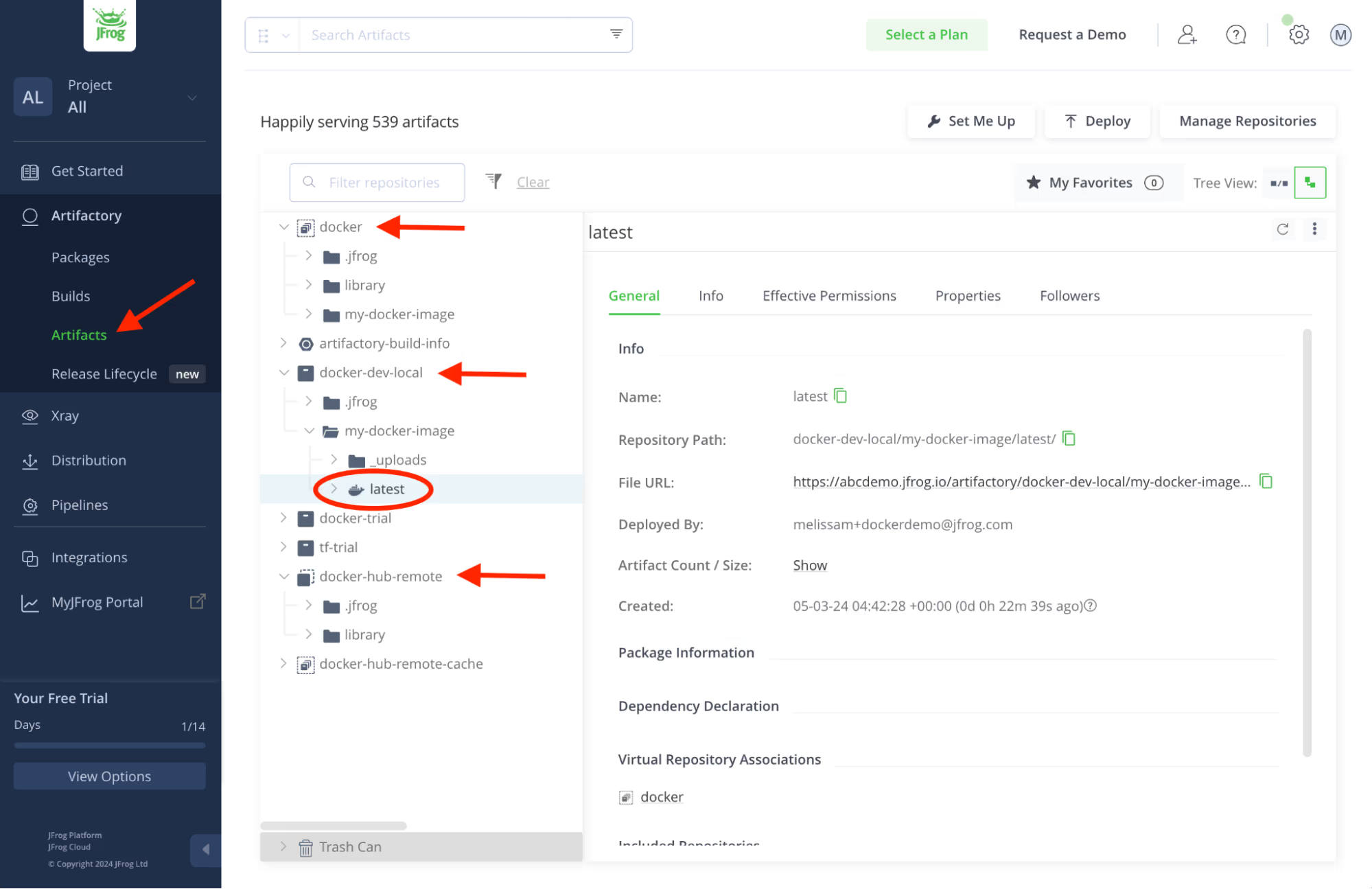
## 

## **Step 7: View the images in your Docker Registry**

Leave the Platform Configurations screen using the navigation at the top left.



Expand the Artifactory menu and click the Artifacts menu item. In the *Artifact Tree View*, expand the *docker*, the *docker-dev-local* and the *docker-hub-remote* repositories to see your new artifacts. Click on your image in your *docker-dev-local* repository to see its details.



**Step 1:How to request/get access to Jfrog?**

To request access to JFrog when your organization uses CyberArk for credential management.

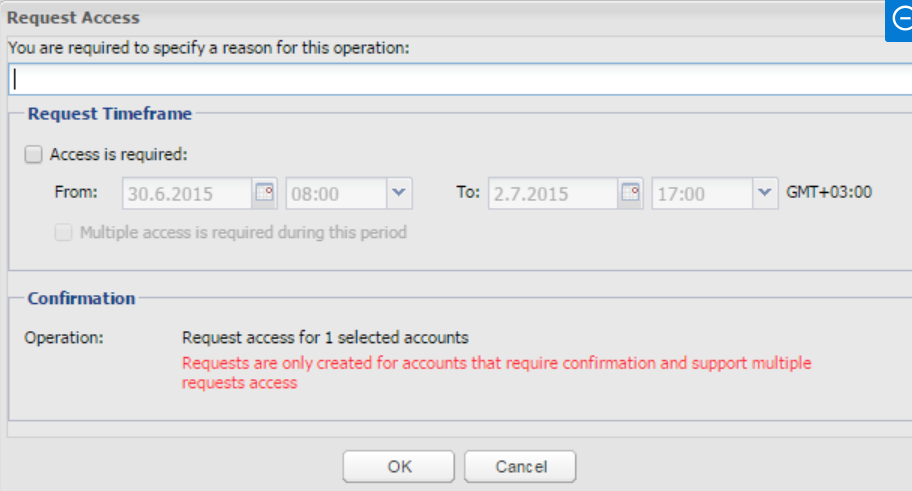
Submit a Request:

Contact your organization's IT or security team to request access to JFrog.

Once access is granted, retrieve the JFrog username and password from CyberArk.

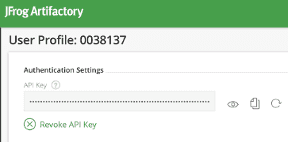
Access Jfrog :

Use the credentials retrieved from CyberArk to log in to your JFrog instance.

****

**Step 2 : How to Set Up JFrog Authentication in Your Docker Config File**

To authenticate Docker with JFrog using credentials retrieved from CyberArk, update your Docker configuration file (~/.docker/config.json).

Use CyberArk to retrieve the JFrog username and API key or have access can get api through CyberArk User profile.

Update Docker Config :

Open or create the ~/.docker/config.json file.

Update following line in config file.

*{*

*"auths": {*

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

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

*}*

*}*

*}*

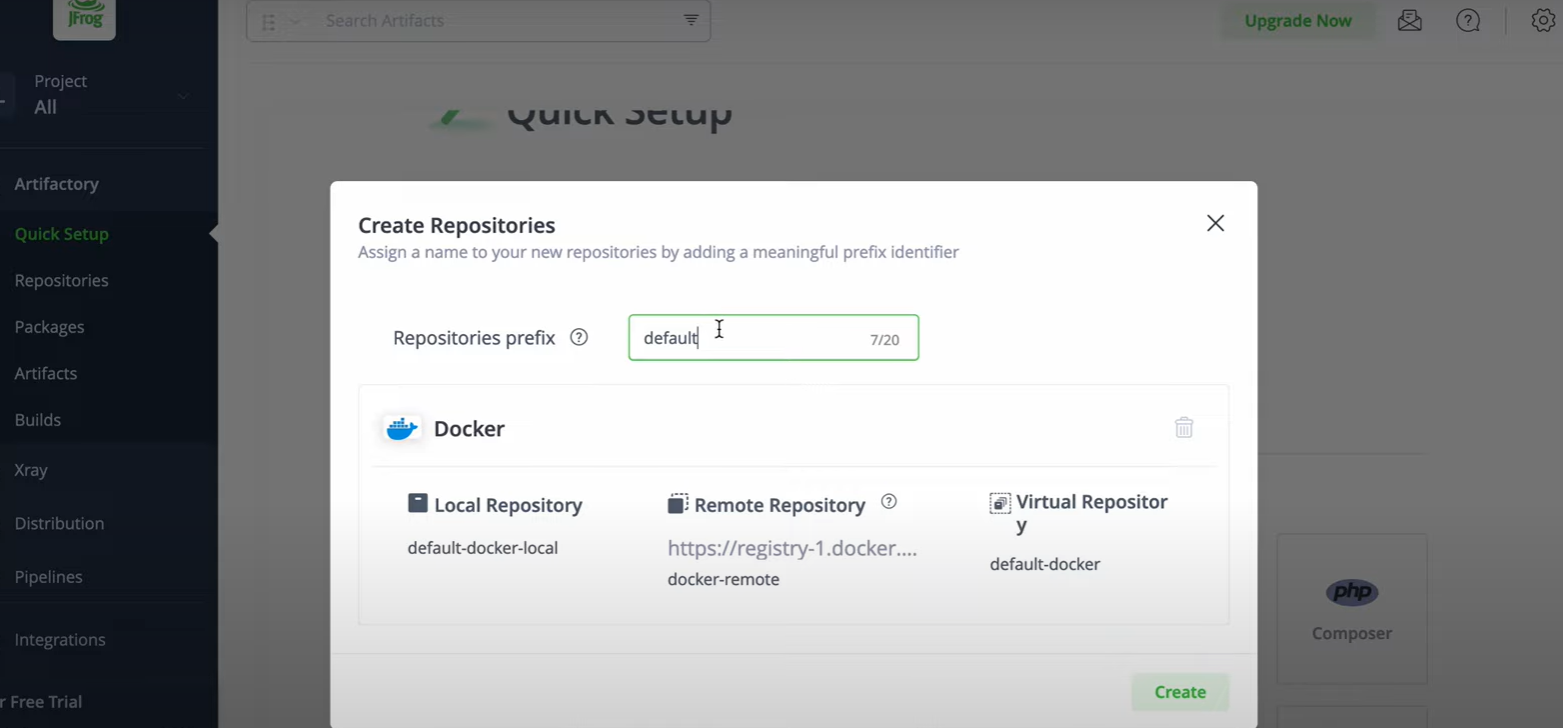
Replace <your-jfrog-repo> with your JFrog repository URL and API key.

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

**Step 3 : How to Pull a Docker Image from DockerHub Through JFrog Proxy**

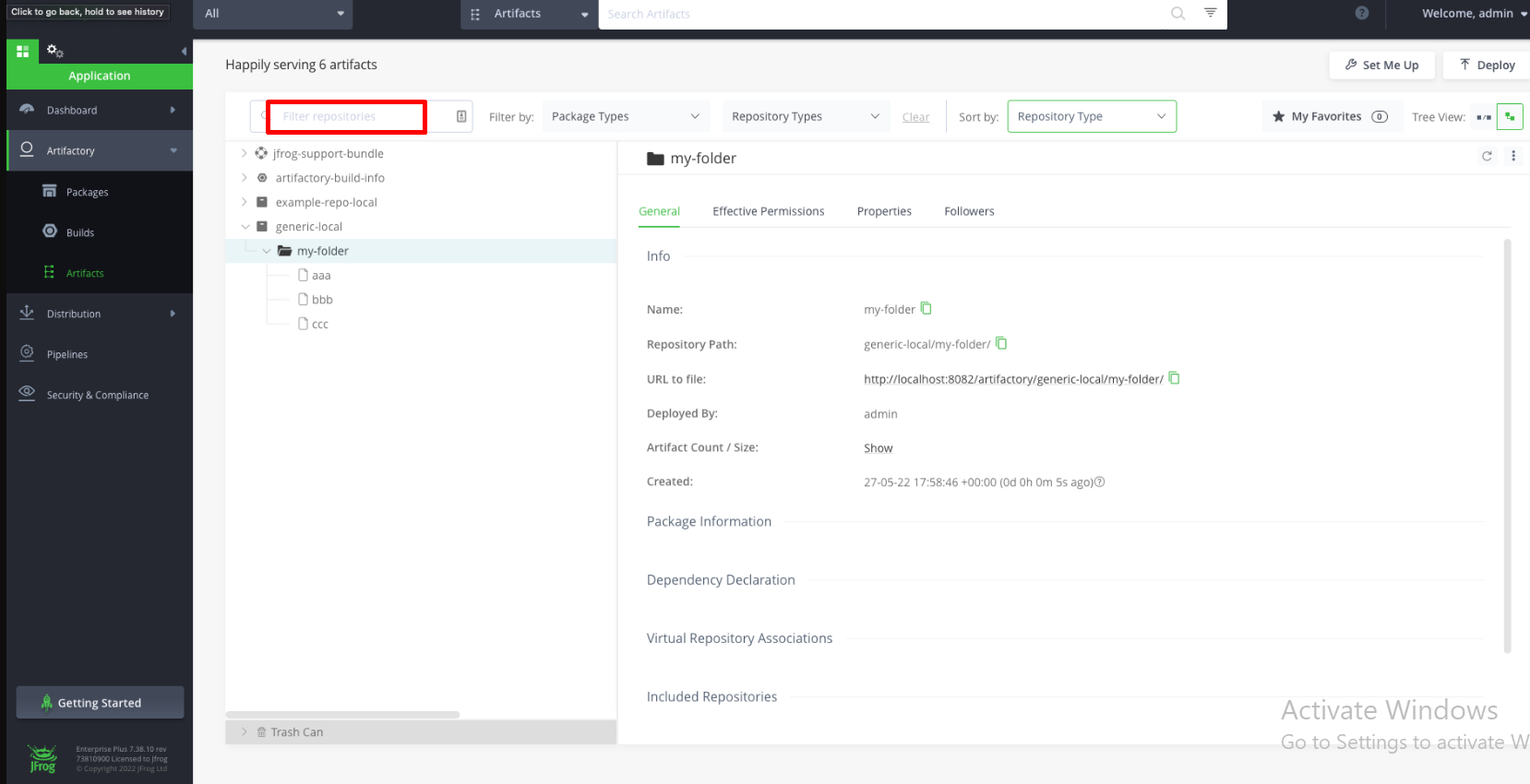
Ensure your JFrog instance has a remote repository configured to proxy DockerHub.

If not, create a remote repository.



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.)

****

**Step 4 : How to Build a Docker Image from a Dockerfile and Push to a JFrog Repo**

To build a Docker image and push it to a JFrog repository:

Build the Docker Image  
Navigate to the directory containing your Dockerfile.

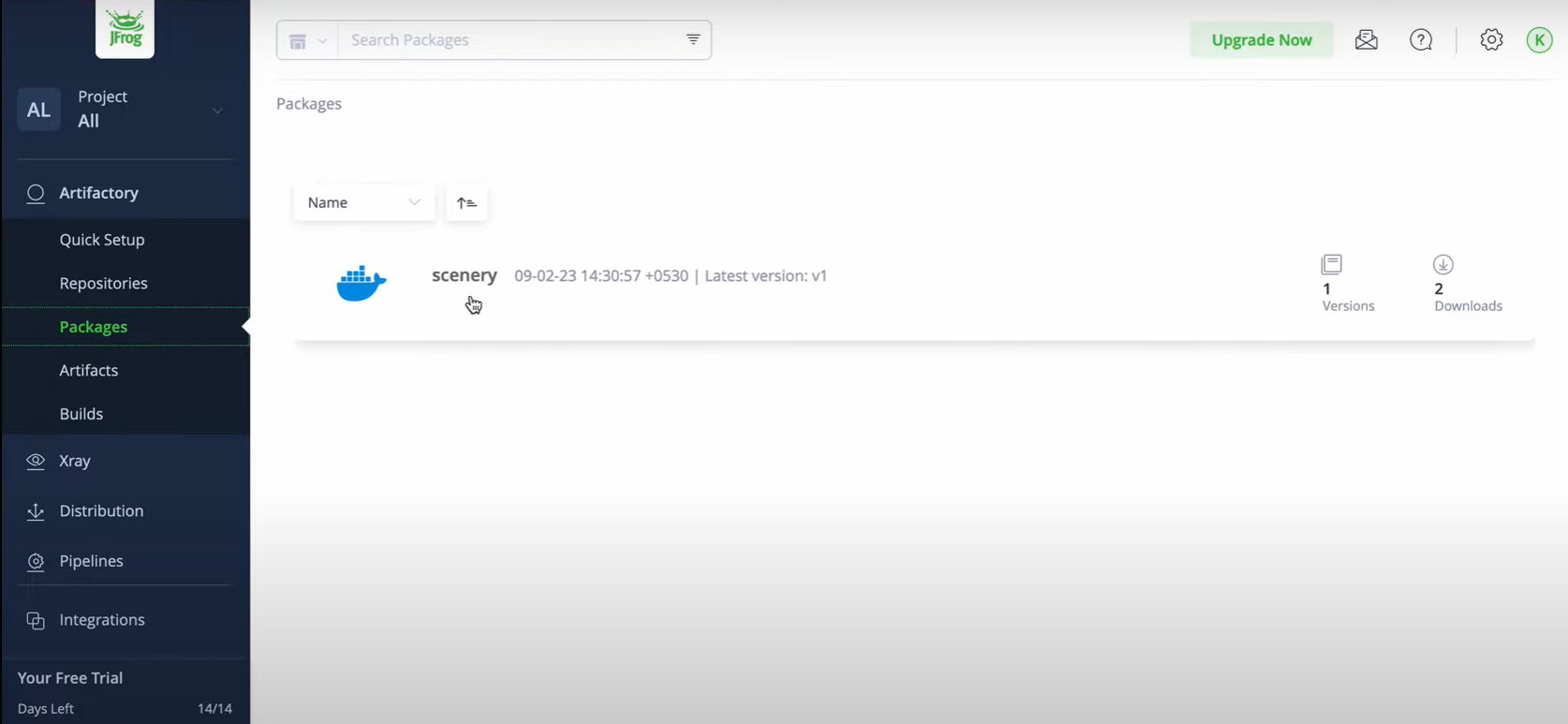
To Build the Image :

*docker build -t <your-jfrog-repo>.jfrog.io/<image-name>:<tag> .*(Replace <your-jfrog-repo>, <image-name>, and <tag>)

Push the Image to JFrog

To Push the Image :

docker push <your-jfrog-repo>.jfrog.io/<image-name>:<tag>



Now go to your JFrog Artifactory and navigate to the repository to verify the image has been uploaded.