Setting Up Docker Build-info

General

To enable fully traceable Docker builds, the Jenkins Artifactory Plugin can collect build info for your Docker builds in Jenkins by setting up an internal proxy server through which the Docker daemon routes its traffic.

Important

Since version 2.14.0 of the Artifactory Plugin, docker build info creation no longer requires setting up the Build-Info Proxy.
Therefore the Pipeline API which uses the Build-Info Proxy is now deprecated.

Here's the deprecated API:
def rtDocker = Artifactory.docker [credentialsId: 'credentialsId'], [host: 'tcp://daemon IP:daemon port']
Please use the new API, which receives an Artifactory server as an argument, instead of credentialsId:
def rtDocker = Artifactory.docker server: server, [host: 'tcp://daemon IP:daemon port']

If you're moving from the old API to the new one, please make sure to remove the HTTP Proxy redirection in your docker configuration.

Page Contents

System Requirements

- Jenkins build agents running on OSX, Ubuntu 14.x, 16.x or Centos 7.
- Jenkins 2.x.
- Jenkins Artifactory Plugin 2.14.0 or above.
- Artifactory configured as a Docker registry.

Collecting and Publishing Docker Build-Info

To set up Jenkins to collect Docker build info, carefully execute the following steps:

1. Install Docker on all Jenkins build agents
2. Setup Artifactory as a Docker registry
3. Ensure Docker is working correctly with Artifactory
4. Make sure Artifactory reverse proxy is trusted by Jenkins
5. Make sure you have the correct version of the Jenkins Artifactory Plugin
6. For OSX Agents, Run a Socat Container
7. Test your setup

Installing Docker on the Jenkins Build Agent

To install Docker on all the Jenkins build agents, follow the instructions in the Docker Documentation.

Make Sure Artifactory is Set Up as a Docker Registry

Make sure Artifactory can be used as docker registry and is used by the Docker daemon from all Jenkins build agents. If you need to configure Artifactory as a Docker registry, please refer to Getting Started with Docker and Artifactory in the JFrog Artifactory User Guide.

Make Sure Docker Works With the Artifactory Docker Registry

To ensure that Docker and your Artifactory Docker registry are correctly configured to work together, run the following code snippet from all Jenkins build agents:

```
docker pull hello-world
docker tag hello-world:latest <artifactoryDockerRegistry>/hello-world:latest
docker login <artifactoryDockerRegistry>
docker push <artifactoryDockerRegistry>/hello-world:latest
```

If everything is configured correctly, pushing any image including the hello-world image should be successfully uploaded to Artifactory.

Make Sure the Artifactory Reverse Proxy is Trusted by Jenkins

If you are using a self-signed SSL certificate to access your Artifactory reverse proxy, your Jenkins build agents need to trust the reverse proxy domain. If you're not using a self-signed certificate or your Jenkins builds can already access Artifactory successfully using SSL, it means that your reverse proxy is already trusted by your Jenkins agents, and you can, therefore, skip this section.

To add your reverse proxy's certificate to your JRE trust store so your Jenkins build agents trust the Artifactory reverse proxy domain, execute the following steps on each Jenkins agent:

1. Find the JAVE_HOME directory. Make sure that this is the Java distribution used by your Jenkins agent. The following sections refer to the JAVA_HOME directory as $JAVE_HOME.
2. To see if your Artifactory reverse proxy domain is already trusted, list all the certificates that are in your JRE trust store using the following command:

```
sudo $JAVA_HOME/bin/keytool -v -list -keystore $JAVA_HOME/jre/lib/security/cacerts -storepass changeit
```

1. If your domain is not listed, obtain your Artifactory's proxy domain self-signed certificate file and add it to the trust store using the following command:

```
sudo keytool -import -alias <alias> -file <reverseProxyCertificateFile> -keystore $JAVA_HOME/jre/lib/security
/cacerts -storepass changeit
```

Make Sure You Have the Correct Version of Jenkins Artifactory Plugin

To collect Docker build information, you need to have Jenkins Artifactory Plugin 2.14.0 and above installed.

For OSX Agents, Run a Socat Container

If your Jenkins agent runs on an OSX machine, run the following command, to startup a Socat container. This container needs to be up and running for being able to push images to Artifactory using the Artifactory Pipeline API.

docker run -d -v /var/run/docker.sock:/var/run/docker.sock -p 127.0.0.1:1234:1234 bobrik/socat TCP-LISTEN:1234,fork UNIX-CONNECT:/var/run/docker.sock

Test Your Setup

Create a new Jenkins Pipeline job with the following script and run it. If everything is setup correctly, the build should publish build info to Artifactory.

```
node() {
    // Step 1: Obtain an Artifactiry instance, configured in Manage Jenkins --> Configure System:
    def server = Artifactory.server '<ArtifactoryServerID>'

    // Step 2: Create an Artifactory Docker instance:
    def rtDocker = Artifactory.docker server: server
    // Or if the docker daemon is configured to use a TCP connection:
    // def rtDocker = Artifactory.docker server: server, host: "tcp://<docker daemon host>:<docker daemon
port>"
    // If your agent is running on OSX:
    // def rtDocker= Artifactory.docker server: server, host: "tcp://127.0.0.1:1234"

    // Step 3: Push the image to Artifactory.
    // Make sure that <artifactoryDockerRegistry> is configured to reference <targetRepo> Artifactory
repository. In case it references a different repository, your build will fail with "Could not find manifest.
json in Artifactory..." following the push.
    def buildInfo = rtDocker.push '<artifactoryDockerRegistry>/hello-world:latest', '<targetRepo>'

    // Step 4: Publish the build-info to Artifactory:
    server.publishBuildInfo buildInfo
}
```