Title: How to cache local Maven repository using Docker with Pipelines?

Post Body:

I want to run my Maven builds in a Docker container. I don't want to upload all depenedencies with every build, so I tried to mount host's local Maven repository as documented at Using Docker with Pipeline:

> **Caching data for containers**
>
> [...]
>
> Pipeline supports adding custom arguments which are passed to Docker, allowing users to specify custom Docker Volumes to mount, which can be used for caching data on the agent between Pipeline runs. The following example will cache ~/.m2 between Pipeline runs utilizing the maven container, thereby avoiding the need to re-download dependencies for subsequent runs of the Pipeline.
>
> ```
> pipeline {     agent {         docker {             image 'maven:3-alpine'             args '-v $HOME/.m2:/root/.m2'
> ```

Code

```
pipeline {     agent {         docker {             image 'maven:3-alpine'             args '-v /home/jenkins/.m2:/root/.m2'
```

Log

```
Running in Durability level: MAX_SURVIVABILITY [Pipeline] Start of Pipeline [Pipeline] node Running on jenkins-docker in /home
```

Problem

After building the `~/m2` directory is empty, no file/directory was added. All files were added under `/home/jenkins/workspace/Test/Docker Test@2/?/.m2` (*Test* is the name of the folder, *Docker Test* is the name of the pipline).

The problem is that this directory is only used for this particular pipeline not for other pipelines, so I could not share local Maven repository with different pipelines/jobs.

Also my `settings.xml` is not used, because it is saved under `~/m2`.

Is there any solution for sharing local Maven repository and Maven settings with different pipelines using Docker?

Accepted Answer:

You can see my answer related but using Gradle configuration.

As you said, in my base image, Jenkins runs the Docker container with the user 1002 and there is no user defined. You have to configure the Maven variable `user.home` in order to put the dependencies there. You can do it by including `user.home` in the `JAVA_OPTIONS` as an environment variable in your pipeline. Also `MAVEN_CONFIG` should be included:

```
environment {   JAVA_TOOL_OPTIONS = '-Duser.home=/var/maven'   SETTINGS = credentials('your-secret-file') }
```

and create a volume to cache the dependencies:

```
docker {     image 'maven:3.3.9-jdk-8-alpine'     args '-v $HOME:/var/maven'     reuseNode true }
```

UPDATE: forgot to tell you that you can put your `settings.xml` in a secret file in order to use a 'least exposure principle' to limit credentials exposure in the Jenkins pipeline. Also we are configuring personal credentials and this is the way we are configuring for instance Nexus credentials per user. Check the Jenkins documentation on how to upload your secret file in your credentials:

```
sh 'mvn -s $SETTINGS -B clean verify'
```

UPDATE2: I'm not using declarative pipeline, so my pipeline looks like:

```
            withCredentials([                 file(credentialsId: 'settings-xml', variable: 'SETTINGS')]) {
```

It seems it can also be used in declarative pipelines but I did not test it myself.

Highest Rated Answer:

I found a work-around, see Local settings.xml not picked up by Jenkins agent:

> The issue is related to the `-u uid:gid` that jenkins uses to run the container. As you may know the image you are running only has the user `root` created, so when jenkins pass its own uid and gid , there is no entry for the user and consequentially no `$HOME` declared for it.
>
> If you only want to run the build independently of the user, you can use the follow as agent:
>
> ```
> agent {         docker {             image 'maven:3-alpine'             args '-v $HOME/.m2:/root/.m2:z -u root'
> ```
>
> A few notes:

1. if you notice the volume I am using with the flag `z`, as I am going to build with root, I need to tell docker that this volume will be shared among another containers, and then preventing access denied from my jenkins container (running with the user jenkins not root)
2. I tell jenkins to reuseNode, so any other stage using the same image, will be executing on the same container (it is just to speed up the provisioning time)

Log

```
[DEBUG] Reading global settings from /usr/share/maven/conf/settings.xml [DEBUG] Reading user settings from /root/.m2/settings.
```

Unfortunately the files in local repository `/home/jenkins/.m2` are now owned by user `root` instead of user `jenkins`. That could cause other problems.