Title: How does the Jenkins config file provider plugin work?

Post Body:

How does the config file provider plugin work?

I have a Jenkinsfile for a declarative pipeline (multi-branch build) that contains:

```
configFileProvider([configFile(fileId: 'maven-settings', variable: 'MAVEN_SETTINGS')]) {     sh 'mvn -B -s $MAVEN_SETTINGS -Dw
```

I have tried running this on two different Jenkins installations one which is installed directly and runs as daemon and another running as a container (jenkinsci/blueocean).

When run on the direct the Jenkins Config File Provider plugin is able to provide the required settings.

```
provisioning config files... copy managed file [Maven settings] to file:/var/lib/jenkins/workspace/redacted@tmp/config89893541
```

When run under the jenkinsci/blueocean container it fails with:

```
provisioning config files... not able to provide the file [ManagedFile: id=maven-settings, targetLocation=null, variable=MAVEN
```

I have created a managed maven-settings.xml file with id maven-settings for blue/ocean but it is not being picked up. I've also tried copying it to ~/.m2/settings.xml

By contrast the working installation does not have any managed files (settings.xml or otherwise) and I am unable to locate any maven settings file in the workspace. I'm not sure what the @tmp directory is. It is deleted by the time a build finishes.

So my question is:

Where should I put the settings to make the configFileProvider pass them on for the jenkinsci/blueocean build job?

How does the config file provider plugin work?

I have no idea what its doing so its hard to debug. The source is here but Java, Maven & Jenkins are not my main area.

What differences are there when Jenkins itself run as a container?

This answer suggests the the config file provider is unnecessary.

There is a similar question which is unanswered but it relates to a maven plugin.

Accepted Answer: None

Highest Rated Answer:

I have part of the answer. My maven-settings file wasn't being picked up because I was using the file name rather than the file **Id** which is different.

The remaining part of the question is how is the original Jenkins instance able to generate this file without it being listed as a managed file.