Notes –

**Command to Forcefully Remove a Directory:**

rm -rf <directory\_name>

**Command to Give Full Access (rwx) to the Current Directory:**

bash

Copy code

chmod u+rwx .

This grants **read**, **write**, and **execute** permissions to the **owner** (which in this case is root).

**Give Full Access to All Users (user, group, others):**

If you want **all users** (not just root) to have **read**, **write**, and **execute** permissions:

bash

Copy code

chmod a+rwx .

This will apply **read**, **write**, and **execute** permissions to **user**, **group**, and **others**.

**Apply Permissions Recursively (for All Files/Subdirectories):**

If you want to give the same permissions to all files and subdirectories inside the current directory, use the **-R** flag:

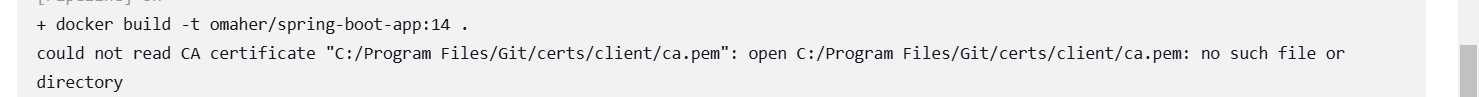
bash

Copy code

chmod -R a+rwx .

This will apply **read**, **write**, and **execute** permissions recursively to all files and directories under the current directory.

Error –



**Step 1: Install Docker on the Linux Node**

Ensure Docker is installed and running on your Jenkins controller's Linux node:

1. Install Docker:

bash

sudo apt-get update

sudo apt-get install -y docker.io

1. Add the Jenkins user to the Docker group to allow non-root access:

bash

sudo usermod -aG docker jenkins

Restart the system or log out and log back in to apply group changes.

1. Verify Docker installation:

bash

docker --version

1. Test running a container:

bash

docker run hello-world

**Step 2: Configure Docker in Jenkins**

1. Go to **Manage Jenkins** → **Global Tool Configuration**.
2. Scroll down to the **Docker** section and click **Add Docker**.
3. Set the following fields:
   * **Name**: docker (or any identifier for your Docker CLI installation).
   * **Install automatically**: Check this box if you want Jenkins to manage Docker installation.
4. Save the configuration.

**Step 3: Declarative Pipeline with Docker**

Here’s a Jenkins Declarative Pipeline that uses Docker on the Linux node:

groovy

pipeline {

agent {

label 'linux-node' // Use the Linux node where Docker is installed

}

tools {

docker 'docker' // Use the Docker tool configured in Jenkins

}

stages {

stage('Verify Docker Installation') {

steps {

sh 'docker --version' // Check if Docker CLI is accessible

}

}

stage('Build Docker Image') {

steps {

script {

def dockerImage = docker.build('my-app-image:latest', '.')

}

}

}

stage('Run Docker Container') {

steps {

script {

docker.image('my-app-image:latest').run('-d -p 8080:8080')

}

}

}

}

}

**Explanation**

1. **Agent Block**:
   * label 'linux-node': Ensures the pipeline runs on the Jenkins Linux node with Docker installed.
2. **Tools Directive**:
   * Configures the docker CLI tool defined in **Global Tool Configuration**.
3. **Stages**:
   * **Verify Docker Installation**: Checks if Docker CLI is installed and accessible.
   * **Build Docker Image**: Builds a Docker image using the docker.build() method.
   * **Run Docker Container**: Runs a container from the built image.

**Step 4: Test and Debug**

* Save the pipeline and run it in Jenkins.
* If you encounter permission issues with Docker commands, ensure the Jenkins user has access to the Docker group:

bash

sudo usermod -aG docker jenkins