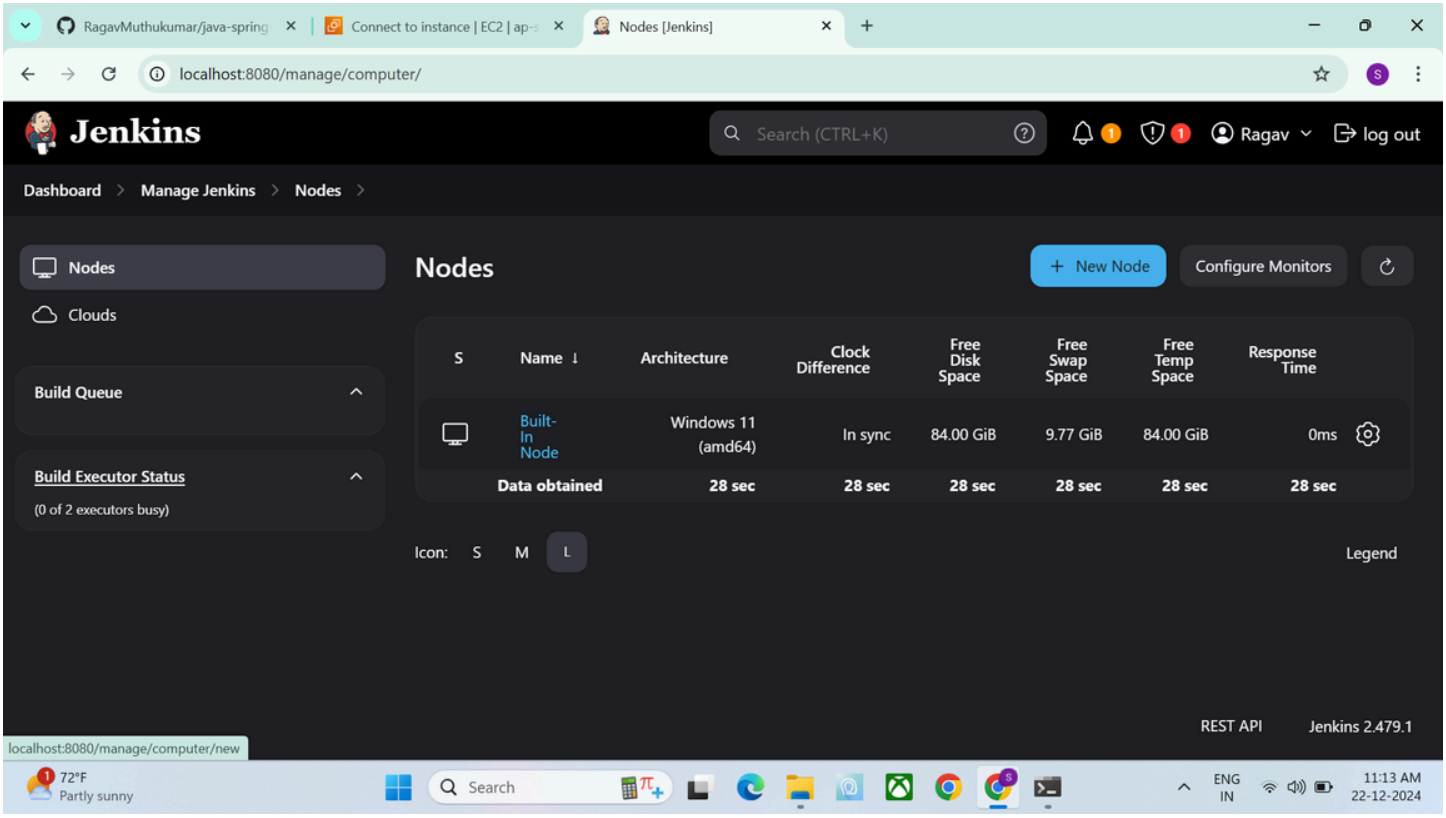
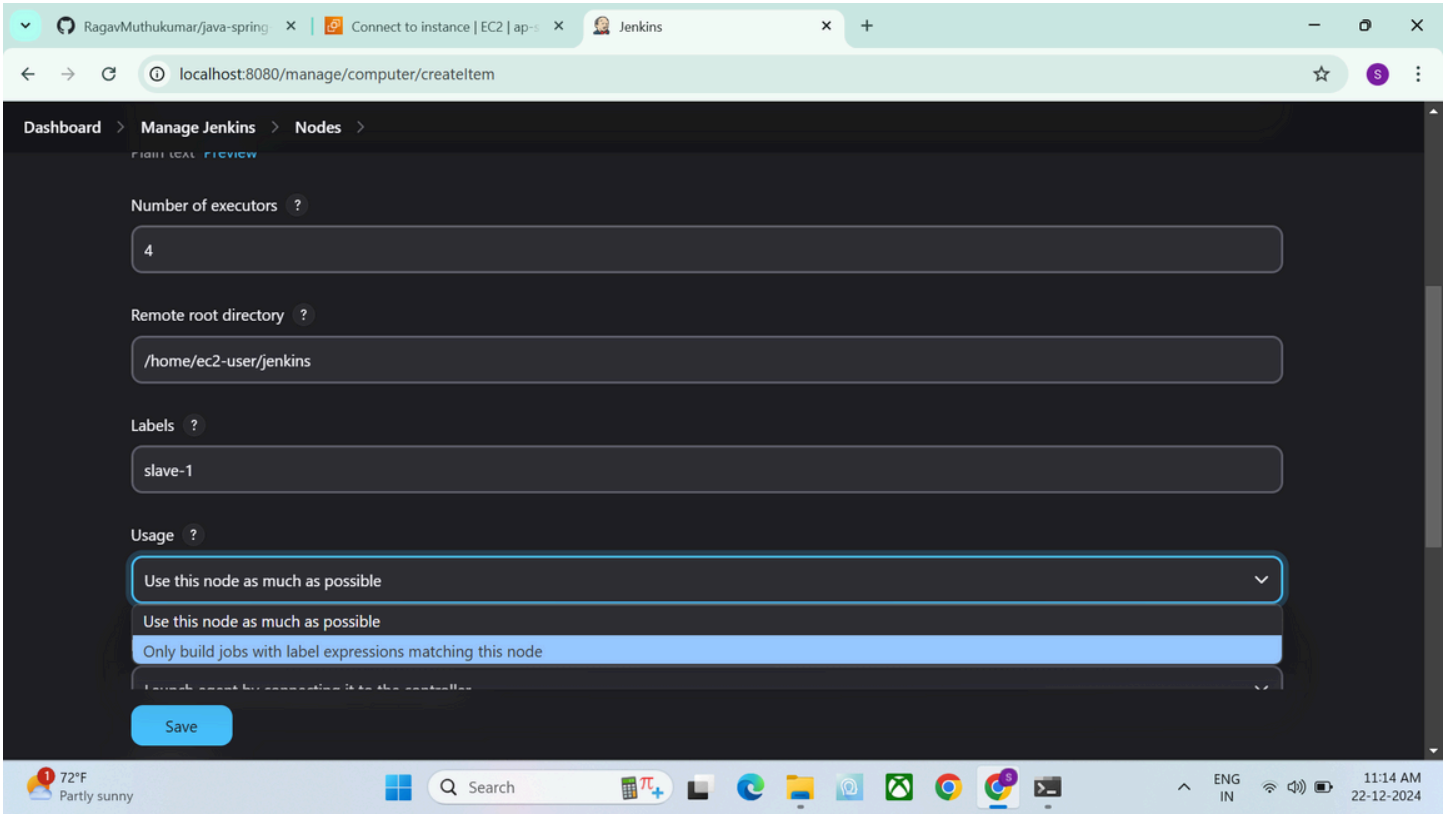
Jenkins Documentation for Node Creation and Pipeline Job Docker Setup

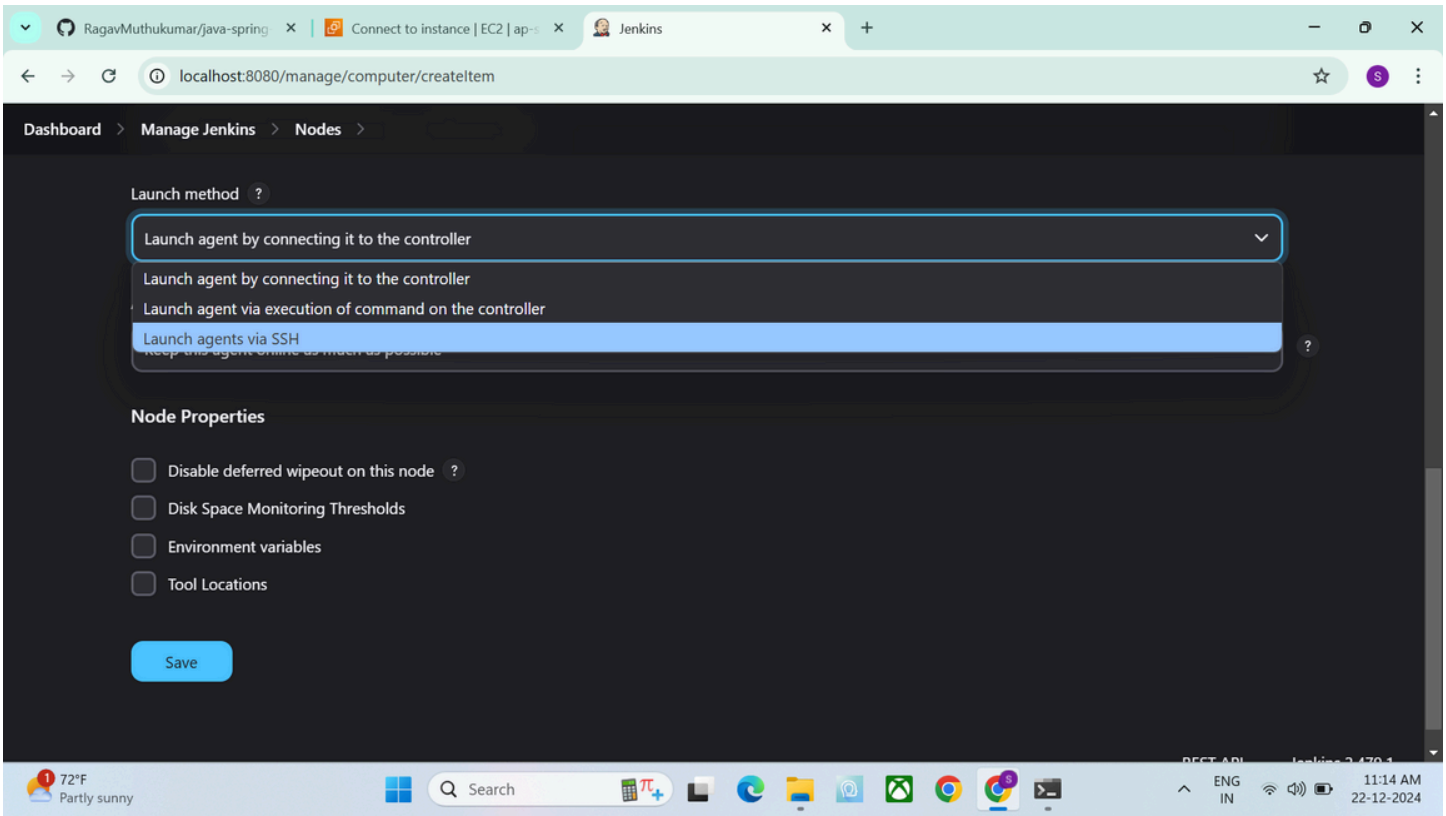
Step 1: Configure Jenkins Node

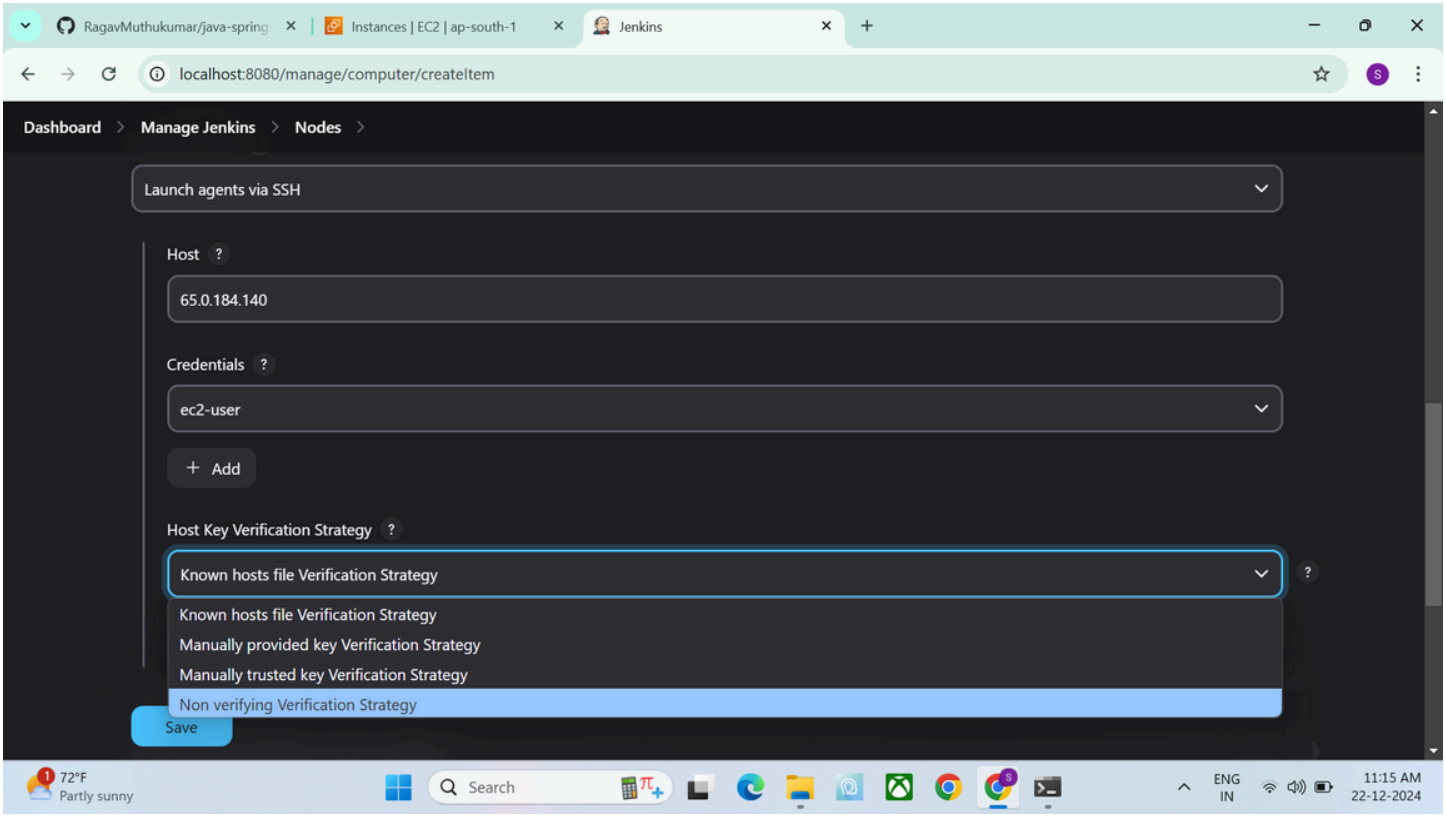
On Jenkins Master:

1. Create a New Node:
   - Go to Manage Jenkins > Manage Nodes and Clouds > New Node.
   - Enter the name (e.g., slave-1) and select "Permanent Agent."
   - Click OK.
2. Configure Node Settings:
   - Enter details such as:
     - Remote root directory: /home/ec2-user/jenkins
     - Labels: slave-1
     - Usage: Use this node as much as possible.
   - Save the configuration.




RagavMuthukumar/java-spring
Connect to instance | EC2 | ap-s
Jenkins
localhost:8080/manage/computer/createItem
Dashboard
Manage Jenkins
Nodes
Number of executors
4
Remote root directory
/home/ec2-user/jenkins
Labels
slave-1
Usage
Use this node as much as possible
Use this node as much as possible
Only build jobs with label expressions matching this node
Save
72°F
Partly sunny
Search
ENG
IN
11:14 AM
22-12-2024


RagavMuthukumar/java-spring
Connect to instance | EC2 | ap-s
Jenkins
localhost:8080/manage/computer/createItem
Dashboard
Manage Jenkins
Nodes
Launch method
Launch agent by connecting it to the controller
Launch agent by connecting it to the controller
Launch agent via execution of command on the controller
Launch agents via SSH
Node Properties
Disable deferred wipeout on this node
Disk Space Monitoring Thresholds
Environment variables
Tool Locations
Save
72°F
Partly sunny
Search
ENG
IN
11:14 AM
22-12-2024



On AWS EC2 Instance (Slave):

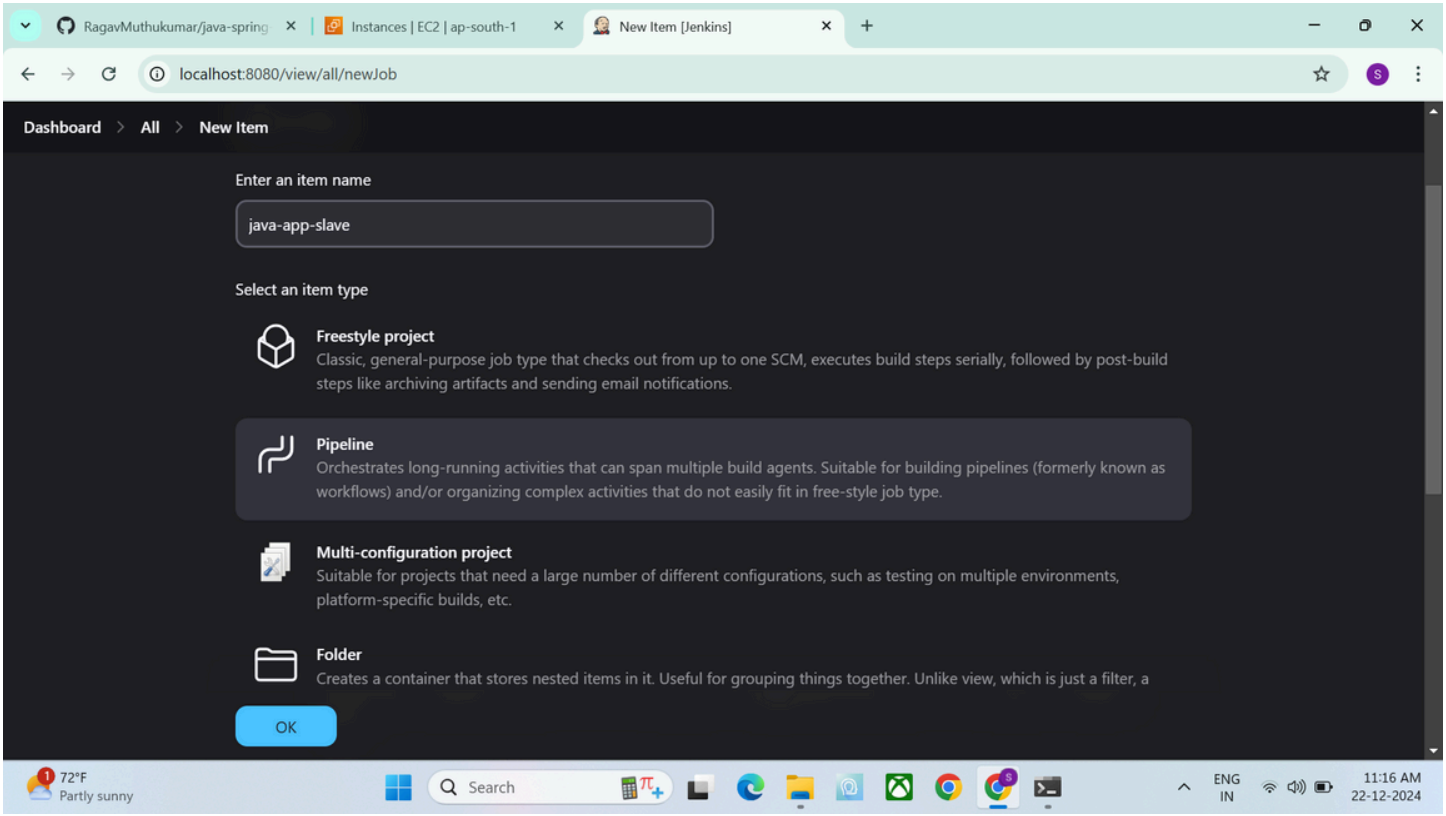
1. Install Java and Create Jenkins Directory:
2. sudo yum install java-17
3. mkdir jenkins
4. Generate SSH Key Pair:(if not already done)
5. ssh-keygen
6. Copy Public Key to Jenkins Master:
   - Add the public key (~/.ssh/id_rsa.pub) to the authorized keys of the Jenkins master.

Step 2: Create a Pipeline Job

1. **Create a New Pipeline Job:**
   - Go to the Jenkins dashboard.
   - Click New Item.
   - Enter a name for the job (e.g., Java Spring Boot Pipeline).
   - Select Pipeline and click OK.
2. **Define the Pipeline Script:**
   - In the Pipeline section, select Pipeline scriptand paste the following:
   - pipeline {
   - agent { label 'slave-1' }
   -
   - environment {
   - REPO_URL = 'https://github.com/RagavMuthukumar/java-spring-boot.git'
   - IMAGE_NAME = 'java-app:latest'
   - }
   -

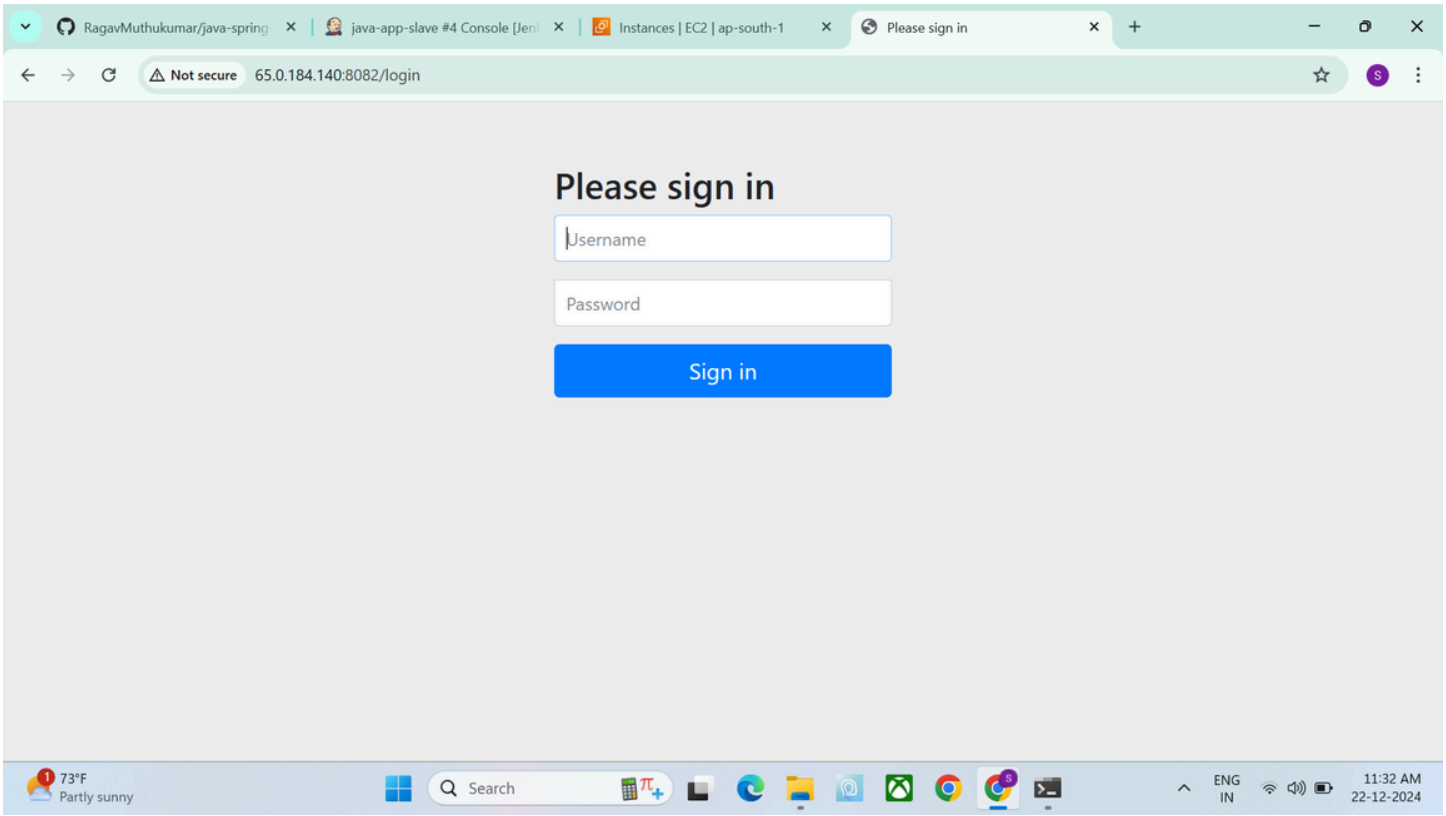
```
    stages {
      stage('Clone Repository') {
        steps {
          git branch: 'master', url: "${REPO_URL}"
        }
      }

      stage('Build Application') {
        steps {
          sh 'mvn clean package'
        }
      }

      stage('Build Docker Image') {
        steps {
          sh '''
          sudo docker build -t ${IMAGE_NAME} .
          '''
        }
      }

      stage('Run Docker Container') {
        steps {
          sh '''
          sudo docker run -d --name java-app-container -p 8082:8080 ${IMAGE_NAME}
          '''
        }
      }
    }
}
```

3. **Save and Run the Job:**
   - Click Save and then Build Now.
   - Monitor the build progress in the Console Output.



Step 3: Verify the Application

1. Once the job is successfully built and the Docker container is running, access the application:
2. http://<AWS_EC2_Public_IP>:8082
3. Replace <AWS_EC2_Public_IP> with the public IP of your slave EC2 instance.



Troubleshooting Tips

1. **Node Connection Issues:**
   - **Verify SSH connectivity between the Jenkins master and slave.**
   - **Ensure the slave's firewall allows connections.**
2. **Pipeline Failures:**
   - **Check the console logs for errors.**
   - **Ensure Docker and Maven are installed and configured properly on the slave.**
3. **Port Accessibility:**
   - **Open port 8082 in the security group of your AWS EC2 instance.**

By following these steps, you will have successfully set up a Jenkins pipeline to build and deploy a Java Spring Boot application using a Jenkins slave on AWS.