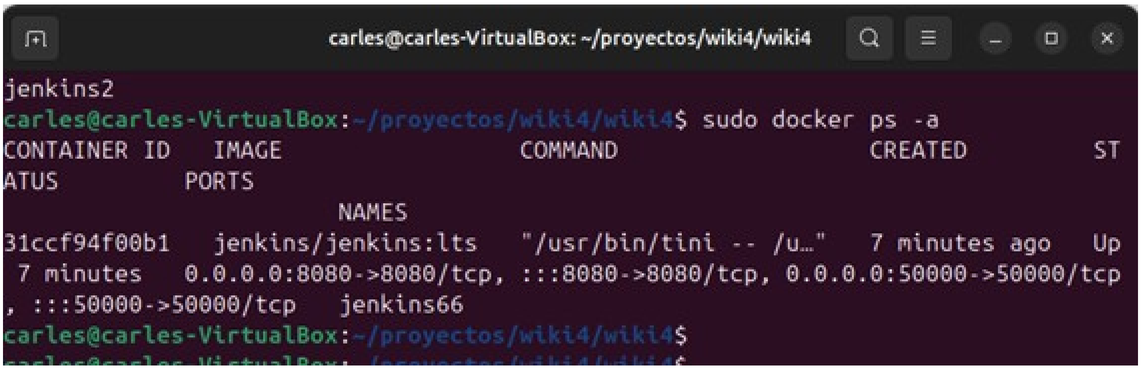
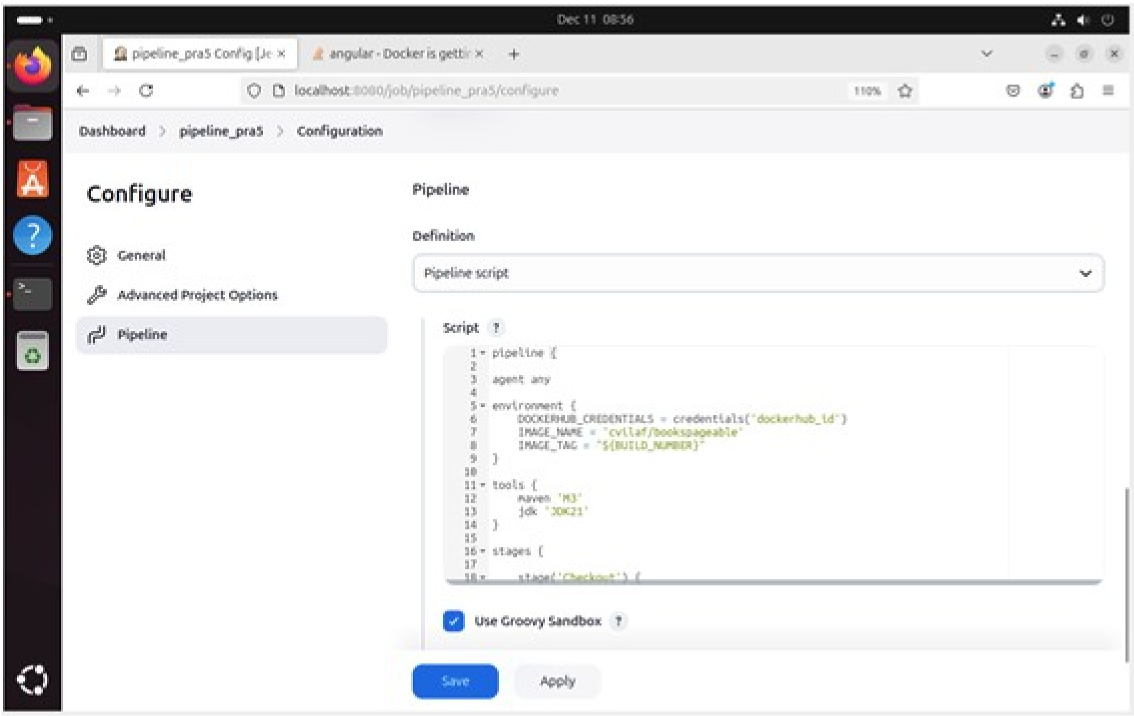
Finalment hem pogut solucionar el problema del docker dins el Jenkins afegint un volum amb la ruta del dimoni de docker.

I hem conseguit que funcioni la pipeline de Jenkins.



```
jenkins2
carles@carles-VirtualBox:~/proyectos/wiki4/wiki4$ sudo docker ps -a
CONTAINER ID   IMAGE                 COMMAND                  CREATED         ST
ATUS          PORTS
                                   NAMES
31ccf94f00b1   jenkins/jenkins:lts   "/usr/bin/tini -- /u…"   7 minutes ago   Up
 7 minutes    0.0.0.0:8080->8080/tcp, :::8080->8080/tcp, 0.0.0.0:50000->50000/tcp
, :::50000->50000/tcp   jenkins66
carles@carles-VirtualBox:~/proyectos/wiki4/wiki4$
carles@carles-VirtualBox:~/proyectos/wiki4/wiki4$
```

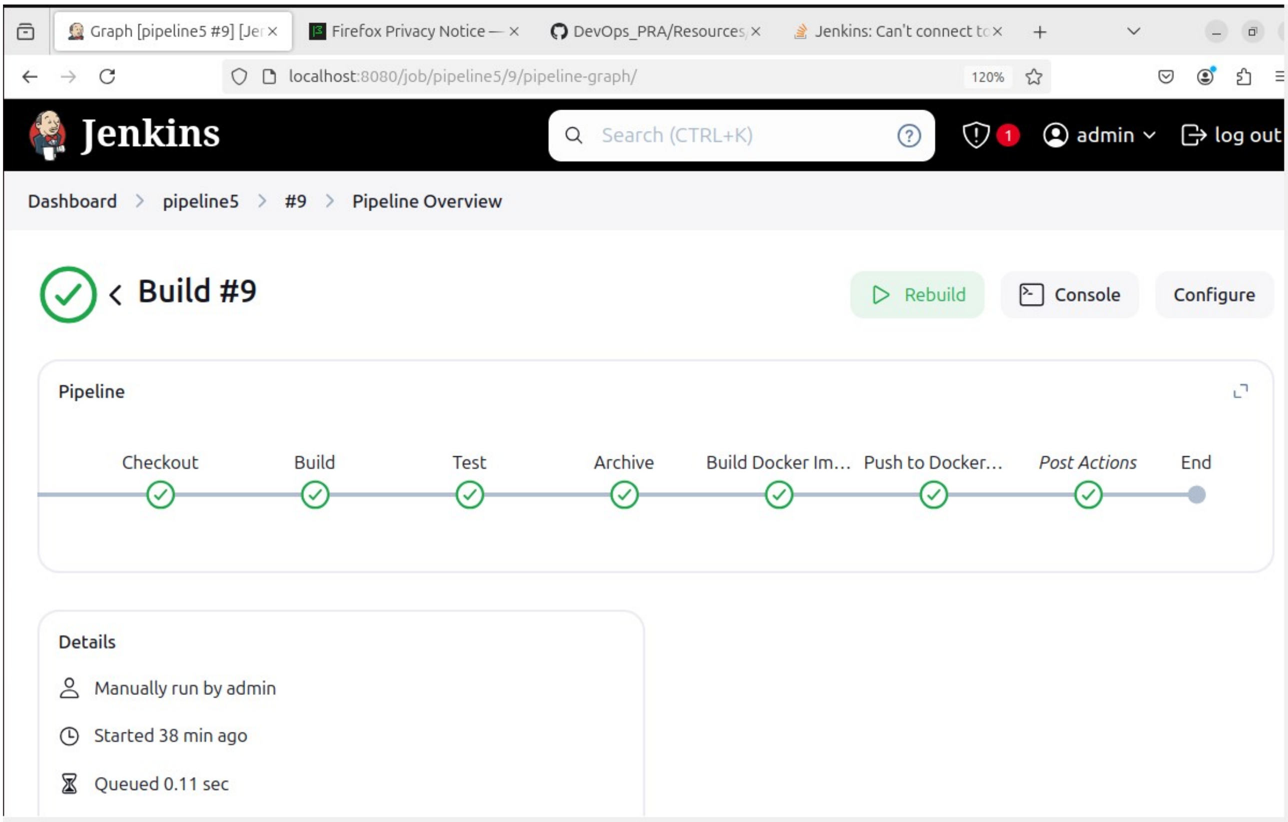


Definition: Pipeline script

Script

```
pipeline {

agent any

environment {
    DOCKERHUB_CREDENTIALS = credentials('dockerhub_id')
    IMAGE_NAME = 'cvllaf/bookspageable'
    IMAGE_TAG = "${BUILD_NUMBER}"
}

tools {
    maven 'M3'
    jdk 'JDK21'
}

stages {

    stage('Checkout') {
```

☑ Use Groovy Sandbox



Manually run by admin

Started 38 min ago

Queued 0.11 sec

Pipeline Jennnkins:

```
pipeline {
agent any
environment {
DOCKERHUB_CREDENTIALS = credentials('dockerhub_id')
IMAGE_NAME = 'cvilaf/bookspageable'
IMAGE_TAG = "${BUILD_NUMBER}"
}
tools {
maven 'M3'
jdk 'JDK21'
}
stages {
stage('prova1') {
steps {
sh 'docker --version'
}
}
stage('Checkout') {
steps {
```

```
git 'https://github.com/AlbertProfe/BooksPageable.git'
}
}

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

stage('Test') {
    steps {
        sh 'mvn test'
    }
}

stage('Archive') {
    steps {
        archiveArtifacts artifacts: '**/target/*.jar', fingerprint: true
    }
}

stage('Build Docker Image') {
    steps {
        script {
            // Build the Docker image and tag it with both BUILD_NUMBER and
latest
            sh "docker build -t ${IMAGE_NAME}:${IMAGE_TAG} -t $
{IMAGE_NAME}:latest ."
        }
    }
}

stage('Push to DockerHub') {
    steps {
        script {
            // Login to DockerHub
            sh "echo ${DOCKERHUB_CREDENTIALS_PSW} | docker login -u $
{DOCKERHUB_CREDENTIALS_USR} --password-stdin"

            // Push both the specific tag and the latest tag
            sh "docker push ${IMAGE_NAME}:${IMAGE_TAG}"
            sh "docker push ${IMAGE_NAME}:latest"
        }
    }
}

}
post {
always {
// Logout from DockerHub and remove images
sh "docker logout"
sh "docker rmi ${IMAGE_NAME}:${IMAGE_TAG} || true"
sh "docker rmi ${IMAGE_NAME}:latest || true"
}
}
}
```