# Jenkinsfile-Dockerfile-Jenkins-Integration

## **options**

The options directive allows you to configure various settings for the pipeline, such as timeouts, retry policies, and build discarding policies. Here are some common options:

options {

timeout (time: 1, unit: 'HOURS') // Timeout the entire pipeline after 1 hour

retry (3) // Retry failed stages up to 3 times

buildDiscarder(logRotator(numToKeepStr: '10')) // Keep the last 10 builds skipDefaultCheckout() // Skip the default checkout

}

## **tools**

The tools directive is used to specify versions of tools that should be installed and available on the agent running the pipeline. Commonly used tools include JDK, Maven, and Gradle.

tools {

jdk 'JDK 1.17' *// Use JDK 1.17 installed in jenkins*

//jenkins->tools->we can either use local jdk/install jdk in jenkinsgradle 'GRADLE\_HOME' *// Use Gradle 6.5 installed on the agent*

// we use gradle installed in local system

}

## **environment**

The environment directive is used to define environment variables that will be available to all stages in the pipeline. These variables can be set to static values or dynamically using scripts.

environment {

DOCKER\_CREDENTIALS\_ID = 'your-docker-credentials-id'

DOCKER\_IMAGE = 'your-docker-image-name'

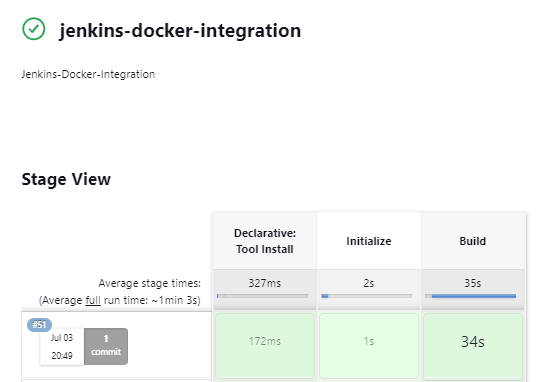
DOCKER\_REGISTRY = 'your-docker-registry-url'

PATH = "$PATH:/usr/local/bin" // Add a directory to the PATH

}

### **Stage view plugin**

If we have to view the stage like below, we have to add the above plugin.



### **Difference between env and def variables:**

* Environment variables are accessible throughout the entire pipeline, across multiple stages and steps. It’s like a global variable.
* def **Variables**: Limited to the script block or function where they are defined.

## **How the jar name is getting decided in gradle file?**

rootProject.name = '02-jenkins-docker'

version = '0.0.1-SNAPSHOT'

02-jenkins-docker-0.0.1-SNAPSHOT.jar

**Note: sdp.yml file can it accessible in Jenkinsfile, Dockerfile as well using environment variable.**

## Reading Version from sdp.yml and populating it to build jar and images:

Read the version from properties file(sdp.yml)

env.*APP\_VERSION* = config.appVersion.toLowerCase()

Add the version name while building jar:

bat "gradle clean build -Pversion=${version}"

Pass the version from jenkins file to docker file

docker build --build-arg VERSION=${env.APP\_VERSION} -t ${env.IMAGE\_NAME}-${env.BUILD\_ID} .

Accept the argument in Dockerfile:

ARG *VERSION*

COPY build/libs/02-jenkins-docker-${*VERSION*}.jar app.jar

## **Execution Of jenkins File:**

* + First It will build the jar file
  + When we run the docker build command in Jenkinsfile to create an image, basically It call docker file to create a build.
  + We pass image name while running docker build command.
  + Inside Dockerfile, we rename original jar with app.jar name and when image will run basically It will execute entry point command.