

Jenkins with Docker Containers as Build Agents

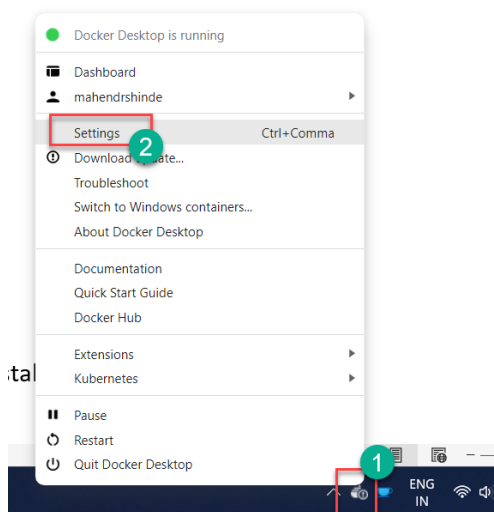
Pre-requisites:

1. Docker Desktop
2. Jenkins CI

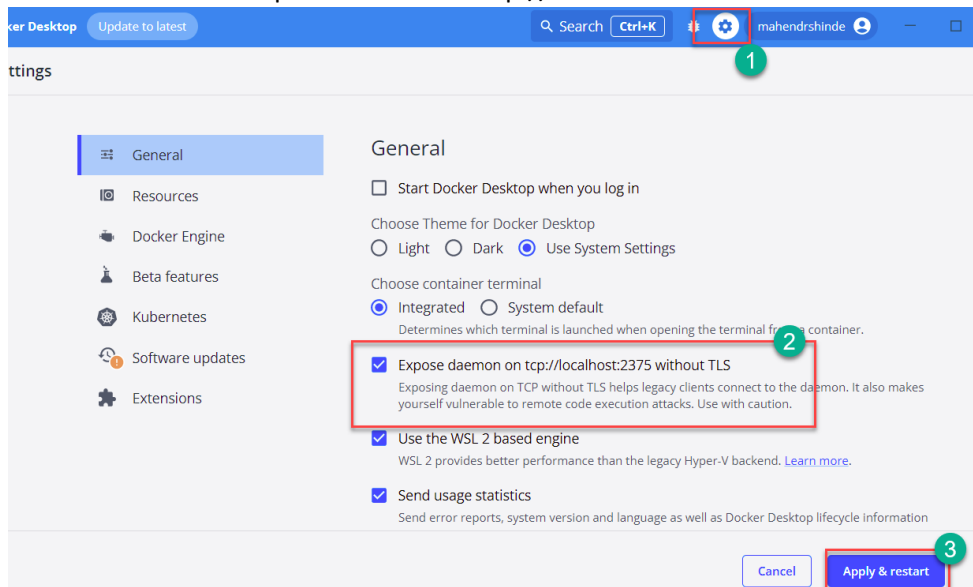
Part 1 : Configuration

1. Launch Docker Desktop Settings (Use Docker Icon in System Tray)

Docker Desktop Must be installed on your System



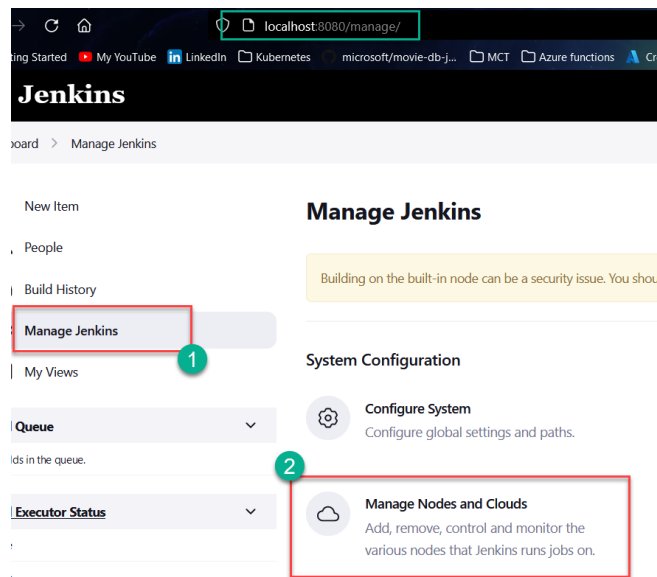
2. Switch the feature “Expose Daemon on tcp://localhost:2375 without TLS”



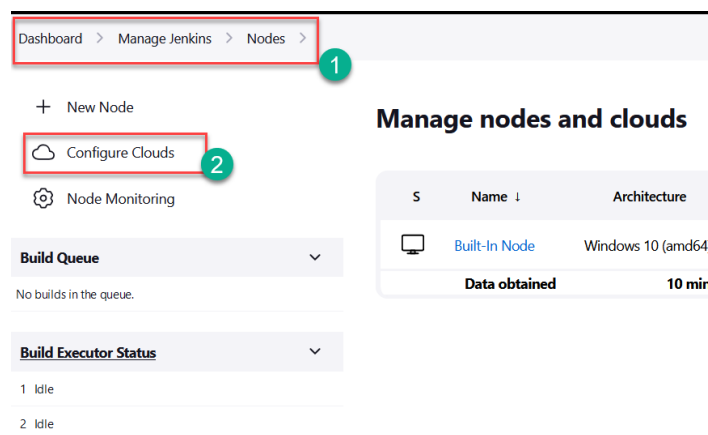
3. Click “Apply And Restart” button.
4. Launch Jenkins, from Manage Jenkins > Plugin Manager install following Plugins (Without Restart)
 - a. Git

- b. Docker Pipeline
- c. Pipeline Plugin (Suite of plugins)

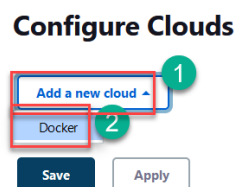
3. From “Manage Jenkins” click on “Manage Nodes and Clouds”



4. Then click on “Configure Cloud” button.



5. Click on “Add new cloud” drop-down and then choose “docker” (It should be only choice)



6. Click on “Cloud Details” button

Configure Clouds

Docker

Name ?
docker

Docker Cloud details...

Docker Agent templates...

Add a new cloud +

Save Apply

7. Now, Enter docker host URI which should be “tcp://localhost:2375” and then click “Test Connection” button, It should display docker API and Daemon version. Click “Save “

Docker

Name ?
docker

Docker Host URI ?
tcp://localhost:2375

Server credentials
- none -

+ Add

Advanced...

Version = 20.10.21, API Version = 1.41

Test Connection

☒ Enabled ?

Save Apply

Part 2 : The Project

1. Create a new Project with Kind “Pipeline”

Enter an item name

Test2 1

» Required field

Freestyle project
 This is the central feature of Jenkins. Jenkins will build your project, combining something other than software build.

Pipeline 2
 Orchestrates long-running activities that can span multiple build agents. Suitable for organizing complex activities that do not easily fit in free-style job type.

Folder
 Creates a container that stores nested items in it. Useful for grouping things to separate namespaces, so you can have multiple things of the same name as local

Multibranch Pipeline 3
 Creates a set of Pipeline projects according to detected branches in one SCM

OK

Organization Folder

2. Set the Git Repository in “Source Code Management”

Configure

Source Code Management

General 1

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

☐ None

☒ Git 2

Repositories 3

Repository URL 2

<https://github.com/mahendra-shinde/sample-library-api>

Credentials 3

- none -

+ Add

Advanced 3

3. Now, add a new “Build Step” -> Build / Publish Docker Image

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment**
- Build Steps
- Post-build Actions

- ☐ Add timestamps to the Console Output
- ☐ Inspect build log for published build scans
- ☐ Terminate a build if it's stuck
- ☐ With Ant ?

Build Steps

Add build step ^

Filter

Add a new template to all docker clouds

Build / Publish Docker Image

Execute Windows batch command

Execute shell

Invoke Ant

Invoke Gradle script

4. Choose "Docker" from dropdown and then enter new image name. Click Save to finish creating new job.

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps**
- Post-build Actions

Advanced ^

Cloud ?

Cloud to use to build image

docker

Image ?

library-app1

Build Args ?

☐ Push image ?

Registry Credentials ?

none

Save

Apply