Make sure the following ports are enabled in your server firewall to accept connections from Jenkins master.

|  |  |
| --- | --- |
| Docker Remote API port | 4243 |
| Docker Hostport Range | 32768 to 60999 |

32768 to 60999 is used by Docker to assign a host port for Jenkins to connect to the container. Without this connection, the build slave would go in a pending state.

Lets get started,

**Step 1:**Spin up a VM, and install docker on it. You can follow the official[documentation for installing docker.](https://docs.docker.com/engine/installation/)based on the Linux distribution you use. Make sure the docker service is up and running.

**Step 2: L**og in to the server and open the docker service file /lib/systemd/system/docker.service. Search for ExecStart and replace that line with the following.

ExecStart=/usr/bin/dockerd -H tcp://0.0.0.0:4243 -H unix:///var/run/docker.sock

**Step 3:**Reload and restart docker service.

sudo systemctl daemon-reload

sudo service docker restart

Step 4: Validate API by executing the following curl commands. Replace 54.221.134.7 with your host IP.

curl http://localhost:4243/version

curl http://54.221.134.7:4243/version

Check the [docker remote API](https://scriptcrunch.com/enable-docker-remote-api/" \t "_blank) article for a detailed explanation of Docker API.

Once you enabled and tested the API, you can now start building the docker slave image.

Create a Jenkins Agent Docker Image

I have created a [Jenkins docker image for maven](https://hub.docker.com/r/bibinwilson/jenkins-slave/). You can use this image or use its [Dockerfile](https://github.com/bibinwilson/jenkins-docker-slave" \t "_blank) as a reference for creating your own.

If you are creating the image on your own, its image should contain the following minimum configurations to act as a slave.

1. sshd service running on port 22.
2. Jenkins user with password.
3. All the required application dependencies for the build. For example, for a java maven project, you need to have git, java, and maven installed on the image.

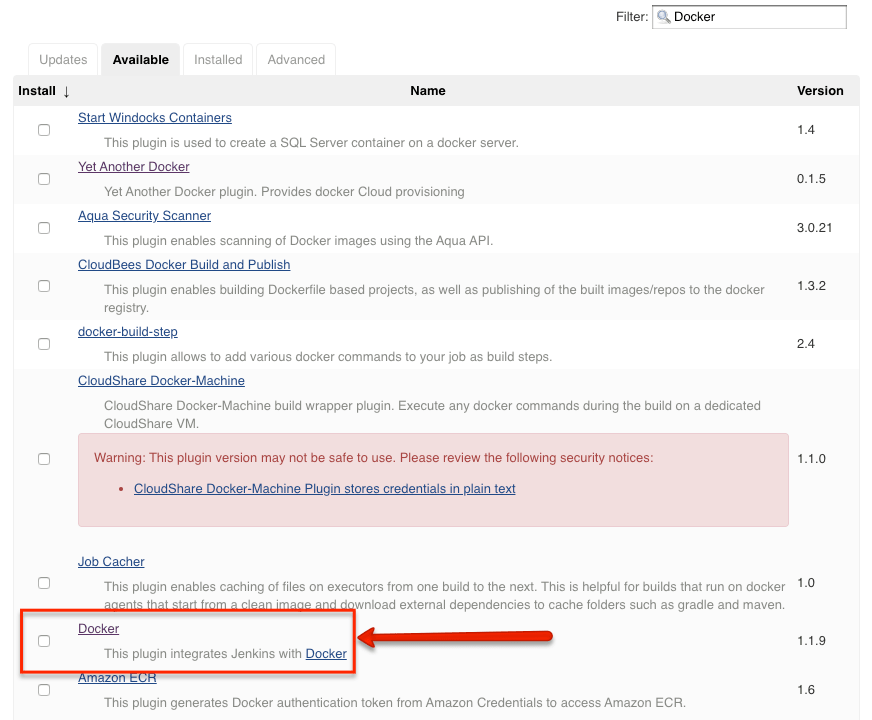
Make sure sshd service is running and can be logged into the containers using a username and password. Otherwise, Jenkins will not be able to start the build process.

Note: The default ssh username is **jenkins** and password is also **jenkins** as per the given Dockerfile. You will have to use these credentials in the below configuration.

Configure Jenkins Server

**Step 1**: Head over to Jenkins Dashboard –> Manage Jenkins –> Manage Plugins.

**Step 2**: Under the Available tab, search for “Docker” and install the docker cloud plugin and restart Jenkins. Here is the [official plugin site](https://plugins.jenkins.io/docker-plugin). Make sure you install the right plugin as shown below.



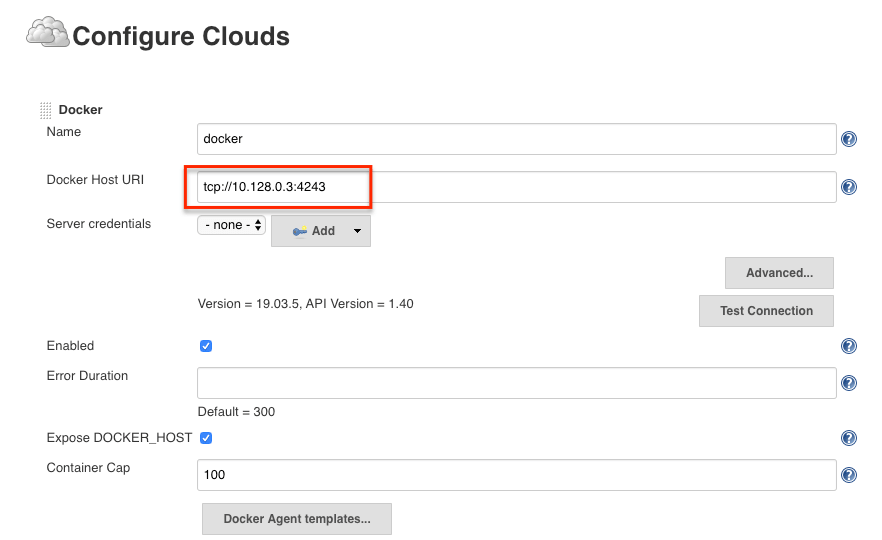
**Step 3:**Once installed, head over to Jenkins Dashboard –> Manage Jenkins –>Configure system.

**Step 4**: Under “Configure System”, if you scroll down, there will be a section named “**cloud**” at the last. There you can fill out the docker host parameters for spinning up the slaves.

**Note**: In Jenkins versions 2.200 or later you will find dedicated cloud configuration under Manage Jenkins –> Manage Nodes and Clouds

**Step 5:** Under docker, you need to fill out the details as shown in the image below.

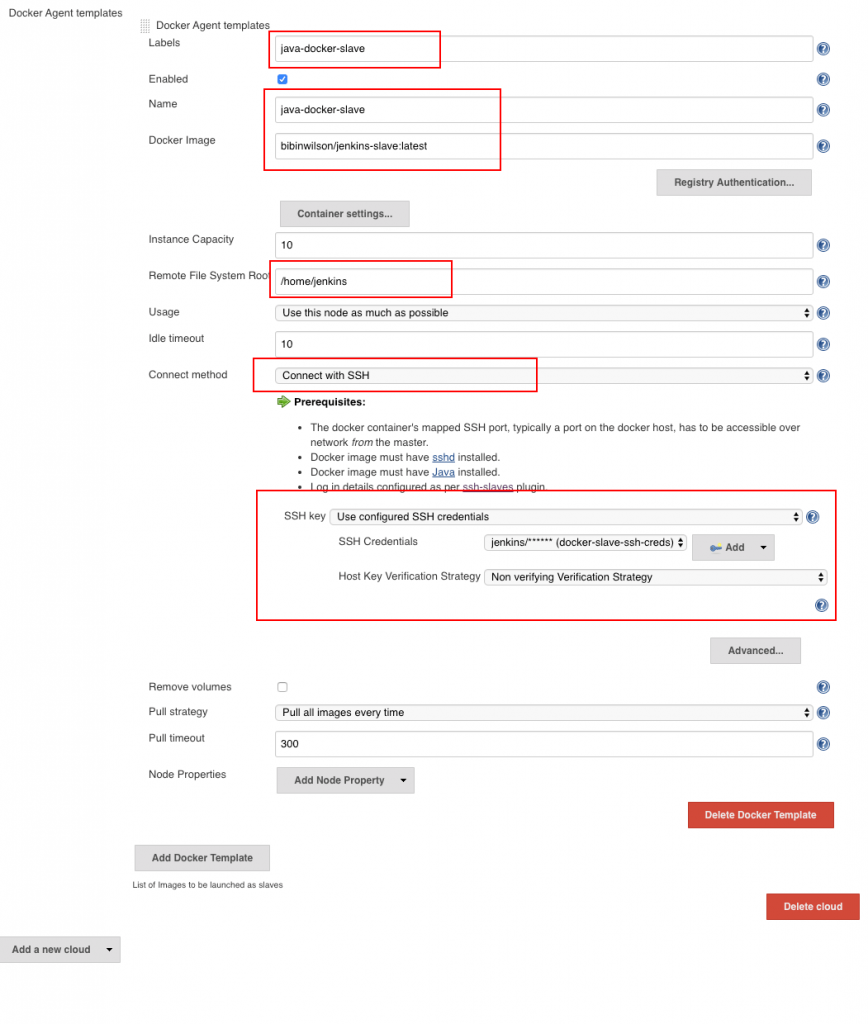
**Note**: Replace “Docker URI” with your docker host IP. For example, tcp://10.128.0.3:4243 You can use the “Test connection” to test if Jenkins is able to connect to the Docker host.



**Step 6:** Now, from “**Docker Agent Template**” dropdown, click th**e**“Add **Docker template**” and fill in the details based on the explanation and the image given below and save the configuration.

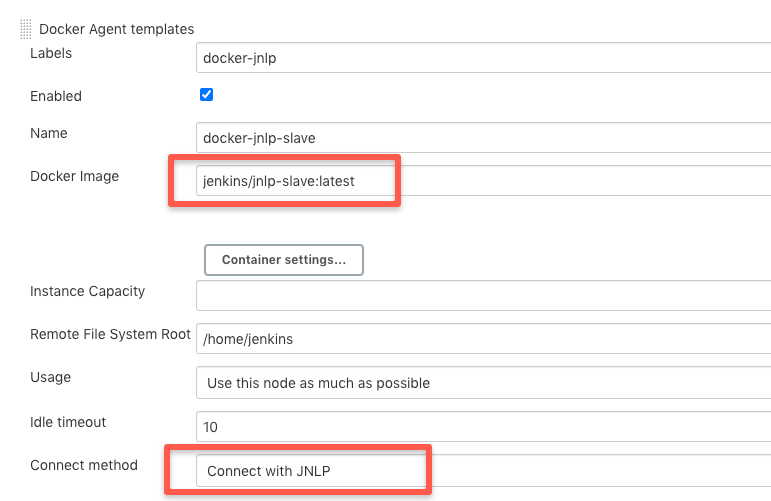
1. **Labels** – Identification for the docker host. It will be used in the Job configuration. Here we usejava-docker-slave
2. **Name:** Name of the docker template. Here we use the same name as label ie, java-docker-slave
3. **Docker Image** –bibinwilson/jenkins-slave:latest or the image that you created for the slave.
4. **Remote Filing System Root** – Home folder for the user you have created. In our case, it’s /home/jenkins
5. **Credentials** – click add and enter the SSH username and password that you have created for the docker image. Leave the rest of the configuration as shown in the image below and click save. If you are using my Docker image, the user will be jenkins & password is also jenkins.

**Note:** There are additional configurations like registry authentication and container settings that you might have to use when configuring this set up in the corporate network.



You can also use JNLP based slave agents. For this, the configurations need a little change as shown below. Primarily the docker image name and the connect method.

**Note:** For JNLP to work, you need to enable the JNLP connection port (50000) in Jenkins’s global security configuration (TCP port for inbound agents). Also, the Jenkins master firewall should be able to accept this connection form the docker host.

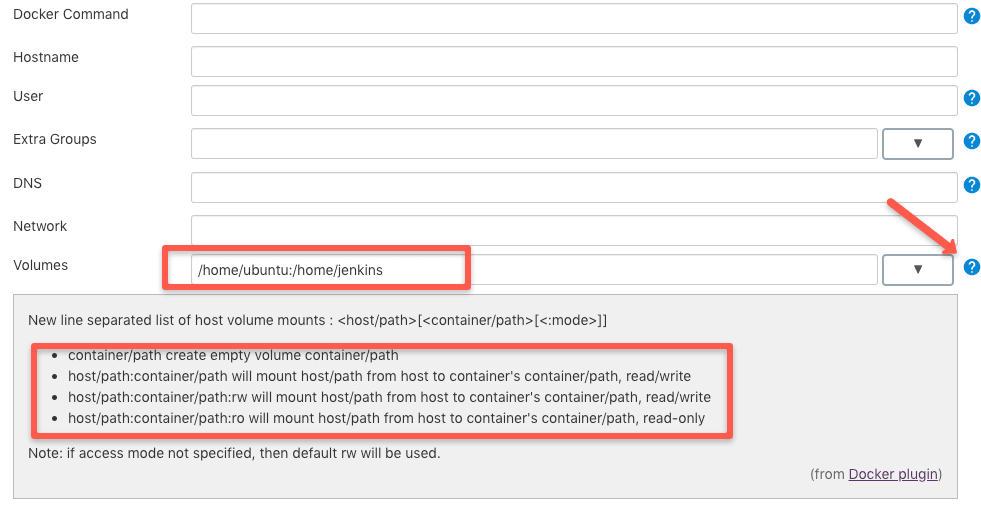


By default, the workspace will not be persisted in the host. However, if you want the workspace to be persistent, add a host volume path under container settings.

For example, if you want the workspace to be available at /home/ubuntu, you can add the volume path as shown below./home/jenkins is the path inside the container.

/home/ubuntu:/home/jenkins

Towards the right of the Volumes option, if you click the question mark, it will show you additional volume options as shown below.

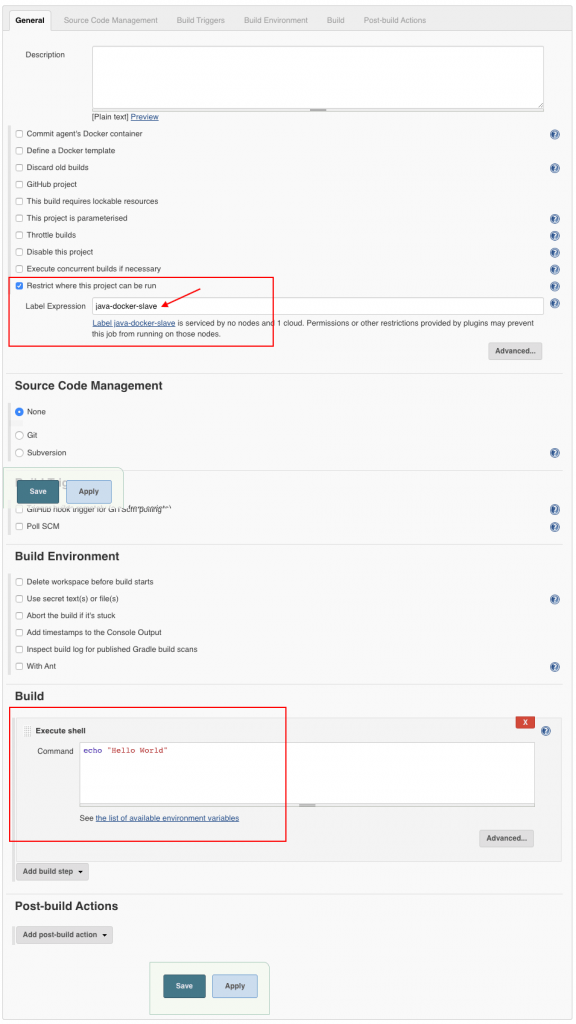


If you are planning to run docker in docker for your CI process, you can mount the host docker.sock as volume to execute docker commands. Check out my [article on running docker in docker](https://devopscube.com/run-docker-in-docker/) to know more about it.

Test Docker Slaves Using FreeStyle Job

Now that you have the slave configurations ready,

1. Create a freestyle job, select “Restrict where this project can be run” option and select the docker host as a slave using the label.
2. Add a shell build step which echoes a simple “Hello World“



If you have done all the configurations right, Jenkins will spin up a container, builds the project and destroys the container once the build is done.

First you will see a pending notification as Jenkins tries to deploy a container on run time and establishes an SSH connection. After a few seconds, your job will start building.



You can check the build logs in your jobs console output as shown below.



Also, you can check out the video explaining the whole process.