V2 Lab 5 - Steps 2A, 2B, 2C & 2D

Like | Updated 8 December 2017 by Roy Mitchley | Tags: *None*

Atom	Anything in this box needs to be edited in Atom Text

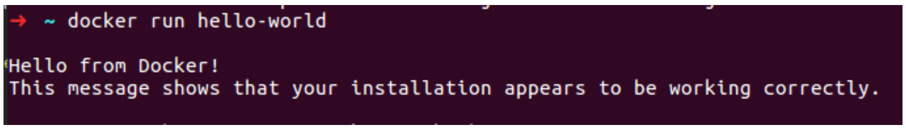
Terminal	This box contains lists of commands that should executed in order on the Terminal

(A) Creating slaves using Docker

The objective of this lab is to be able to scale up the number of Jenkins executors. This will be done using Docker slaves and by creating static ssh servers for jenkins to run builds on. In the real world these slaves could be dynamic and would be hosted on another machine but to keep things simple and contained within the VM, they will be deployed locally.

- On the terminal verify the Docker daemon is running

```
docker run hello-world
```

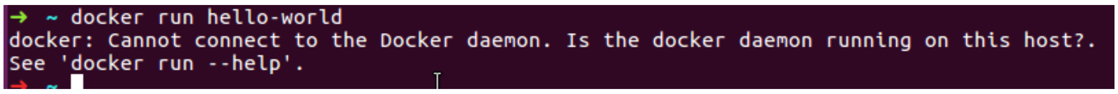


```
→ ~ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

If you see an error such as this, then run

```
sudo service docker start
```



```
→ ~ docker run hello-world
docker: Cannot connect to the Docker daemon. Is the docker daemon running on this host?.
See 'docker run --help'.
```

- Open the `jenkins-node-slave` file located in the `/share/lab-material/docker` folder with Atom

```
atom /share/lab-material/docker
```

- The dockerfile is mostly filled out, but some minor configuration will be added to it. In the top section(#1. Setup the core os packages), add the `openssh-server, git` and `openjdk-7-jre` packages to our core os as shown below. There should be no trailing spaces after any of the back slashes (\).

```
RUN apt-get install -y \
    curl \
    jq \
    wget \
    software-properties-common \
    openssh-server \
    git \
    openjdk-7-jre
```

- In the third section (`#3. Add user jenkins to the image`) create a new jenkins user and generate a password for him. Add the following lines to create our jenkins user and establish some basic config

```
openssl passwd -crypt <PASSWORD>
```
where <PASSWORD> is your new password

```
# password is jenkins
RUN useradd -p dW7IzpQGfPQtY -ms /bin/bash jenkins
ENV JENKINS_HOME /home/jenkins
RUN mkdir -p ${JENKINS_HOME}/.ssh && \
    chown -R jenkins:jenkins ${JENKINS_HOME}
```

NOTE - to change the jenkins user password (string after the -p flag), generate a new one by running `openssl passwd -crypt <PASSWORD>` where <PASSWORD> is your new password and paste in the new encrypted string.

– In the node js (`#5. Install nodejs and deps for builds`) section add the lines below to install nodejs and then all the dependencies needed by our nodejs build (`grunt-cli grunt bower npm-cache istanbul`)

```
RUN curl -sL https://deb.nodesource.com/setup_4.x | bash
RUN apt-get install -y nodejs libfontconfig
RUN npm install -g grunt-cli grunt bower npm-cache istanbul
```

– Finally, at the end of the file (`#7. COMMAND`), append the cmd to start the sshd in the container and save the changes.

```
CMD ["/usr/sbin/sshd", "-D"]
```

– The Jenkins slave expects to be able to use the Jenkins id_rsa key and other things relative to it's home in the Dockerfile. To do so export Jenkins home in this terminal session and copy the key in place

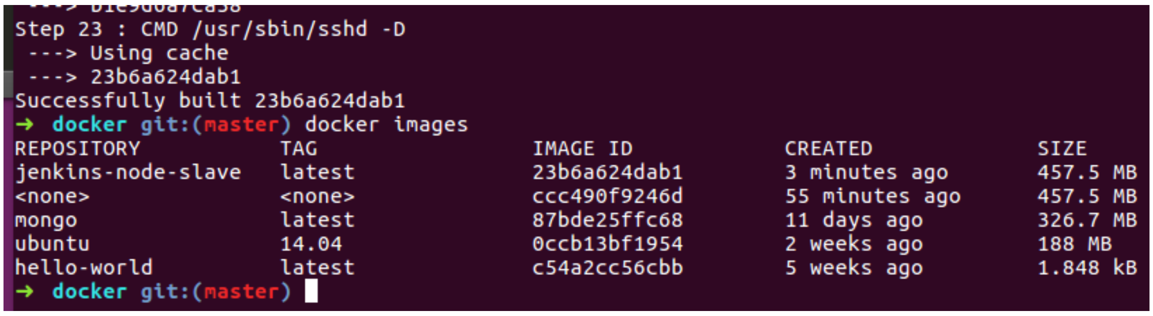
```
cd /share/lab-material/docker
export JENKINS_HOME='/var/lib/jenkins'
sudo cp ${JENKINS_HOME}/.ssh/id_rsa .
sudo chown devops:devops id_rsa
```

– Run the following to build the container. This process may take some time. The -t flag specifies the tag to give the image being created. The -f is for file and the . at the end is to set the context ie the directory to send to the docker daemon for building. If we wanted to add files into the image we would make sure they are in the directory specified by the context

```
cd /share/lab-material/docker
docker build -t jenkins-node-slave -f jenkins-node-slave .
```

– Once complete you should see output like this . Run `docker images` to see the list of local images including the one just built a moment ago

```
docker images
```

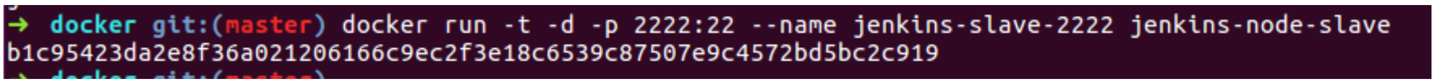


```
Step 23 : CMD /usr/sbin/sshd -D
 ---> Using cache
 ---> 23b6a624dab1
Successfully built 23b6a624dab1
➜ docker git:(master) docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
jenkins-node-slave  latest              23b6a624dab1        3 minutes ago       457.5 MB
<none>              <none>              ccc490f9246d        55 minutes ago      457.5 MB
mongo               latest              87bde25ffc68        11 days ago         326.7 MB
ubuntu              14.04               0ccb13bf1954        2 weeks ago         188 MB
hello-world         latest              c54a2cc56cbb        5 weeks ago         1.848 kB
➜ docker git:(master)
```

– Having built the image, we should now try to start and ssh to it. The -d flag is to run the container in detached mode ie backgrounded. The -p maps ports, we are mapping port 2222 in the VM to 22 in the container ie the sshd port in the container. The --name flag gives the container a meaningful name and the final argument is the name of the image we are about to run.

Terminal

```
docker run -t -d -p 2222:22 --name jenkins-slave-2222 jenkins-node-slave
```

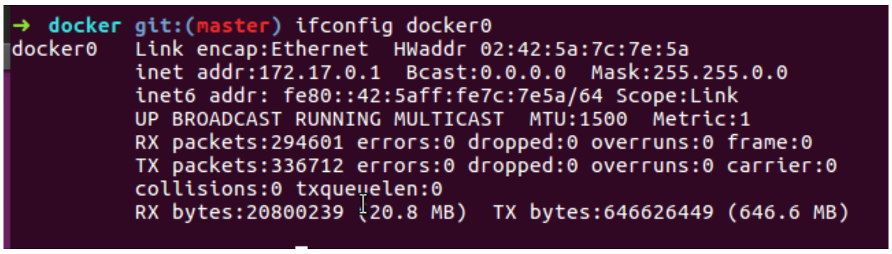


```
➜ docker git:(master) docker run -t -d -p 2222:22 --name jenkins-slave-2222 jenkins-node-slave
b1c95423da2e8f36a021206166c9ec2f3e18c6539c87507e9c4572bd5bc2c919
```

– To ssh to it run `ifconfig docker0` to see what port the docker host is running on. The `inet addr:<IP_OF_DOCKER>` field is what we are looking for so copy out it's value

Terminal

```
ifconfig docker0
```



```
➜ docker git:(master) ifconfig docker0
docker0   Link encap:Ethernet  HWaddr 02:42:5a:7c:7e:5a
          inet addr:172.17.0.1  Bcast:0.0.0.0  Mask:255.255.0.0
          inet6 addr: fe80::42:5aff:fe7c:7e5a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:294601 errors:0 dropped:0 overruns:0 frame:0
          TX packets:336712 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:20800239 (20.8 MB)  TX bytes:646626449 (646.6 MB)
```

– For simplicity and future use, add a meaningful name to the IP by adding this to our `/etc/hosts` file. The `<IP_OF_DOCKER>` should match the one in running the command above. **Be sure to use a double chevron (>>) in this command otherwise it will erase the file and just add your entry.**

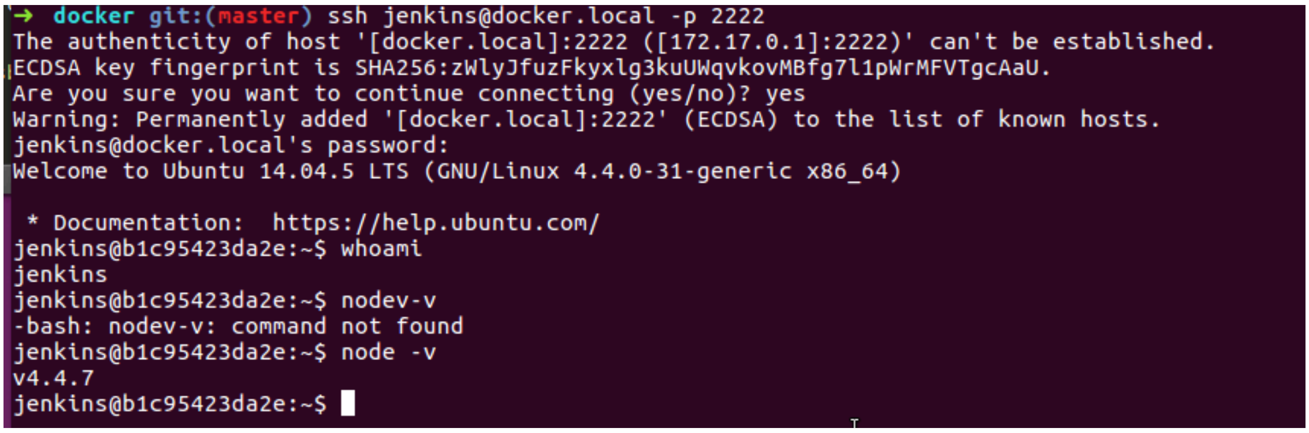
Terminal

```
sudo sh -c "echo <IP_OF_DOCKER>  docker.local >> /etc/hosts"
```

– Verify you can access the container by running sshing to it and typing yes and inputting the password. Type `exit` to get out of the ssh session

Terminal

```
ssh jenkins@docker.local -p 2222
exit
```

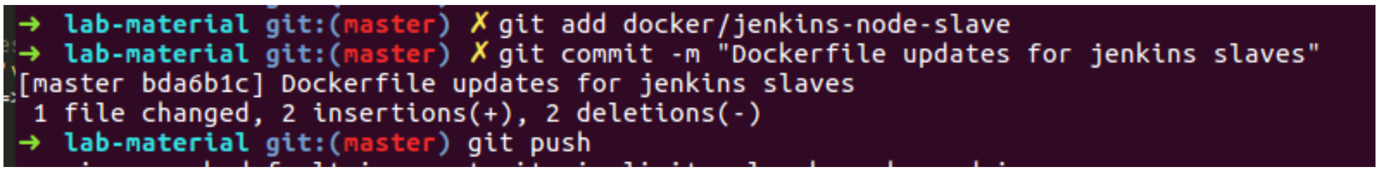


```
→ docker git:(master) ssh jenkins@docker.local -p 2222
The authenticity of host '[docker.local]:2222 ([172.17.0.1]:2222)' can't be established.
ECDSA key fingerprint is SHA256:zWlyJfuzFkyxlg3kuUWqvkovMBfg7l1pWrMFVTgcAaU.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[docker.local]:2222' (ECDSA) to the list of known hosts.
jenkins@docker.local's password:
Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 4.4.0-31-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
jenkins@b1c95423da2e:~$ whoami
jenkins
jenkins@b1c95423da2e:~$ nodev-v
-bash: nodev-v: command not found
jenkins@b1c95423da2e:~$ node -v
v4.4.7
jenkins@b1c95423da2e:~$ 
```

– Commit your changes to the dockerfile.

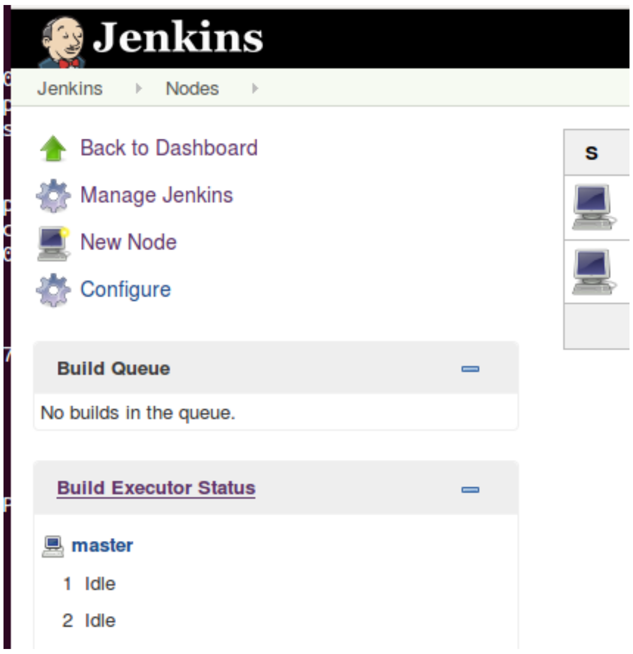
Terminal	git add jenkins-node-slave git commit -m "Dockerfile updates for Jenkins" git push



```
→ lab-material git:(master) ✗ git add docker/jenkins-node-slave
→ lab-material git:(master) ✗ git commit -m "Dockerfile updates for jenkins slaves"
[master bda6b1c] Dockerfile updates for jenkins slaves
 1 file changed, 2 insertions(+), 2 deletions(-)
→ lab-material git:(master) git push
```

(B) Telling Jenkins about the slaves and configuring builds to use them.

– Having verified the container is running, we can now connect jenkins up to the node. On Jenkins console, hit *Build Executor Status* followed by *New Node*

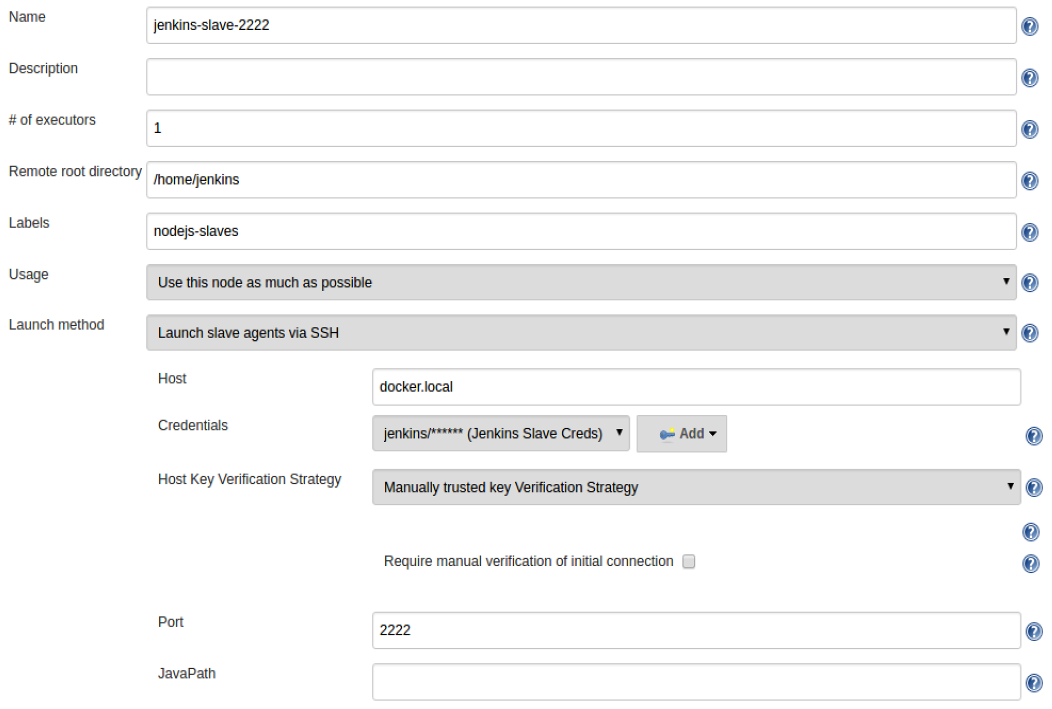


– Name the new node `nodejs-slave-2222`, click the "Permanent Agent" radio button and hit ok. The 2222 is used to to denote the ssh port of the running container - this is for our own ref and has no meaning in Jenkins land.

– On the configuration page, add `/home/jenkins` to the Remote root directory field. Create a sensible label name for our slave label eg `nodejs-slaves`. This will be used to group slaves later and run multiple builds across many slaves of the same namespace.

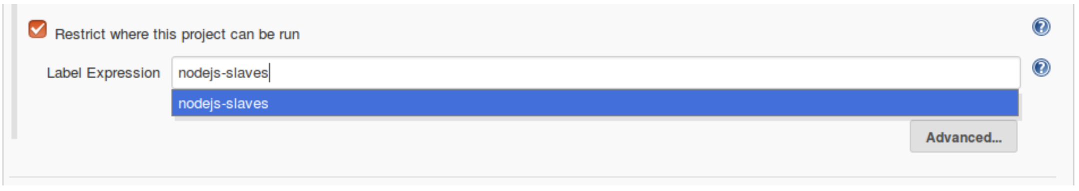
– Set the Launch method to `Launch slaves via SSH`

- Set the hostname to `docker.local`
- On the advanced tab, set the port to `2222`
- `Add` new credentials (username and password) for access to the slaves ie : jenkins / jenkins
- Change 'Host Key Verification Strategy' to `Manually trusted key Verification Strategy`



Name	jenkins-slave-2222
Description	
# of executors	1
Remote root directory	/home/jenkins
Labels	nodejs-slaves
Usage	Use this node as much as possible
Launch method	Launch slave agents via SSH
Host	docker.local
Credentials	jenkins/****** (Jenkins Slave Creds)
Host Key Verification Strategy	Manually trusted key Verification Strategy
Require manual verification of initial connection	☐
Port	2222
JavaPath	

- Hit save and the slave should launch automatically. you can view the information about it coming online by selecting the *nodejs-slave-2222* slave on the dashboard of jenkins then *log* on the left hand side.
- Once the slave has launched, configure the "hello-world" job to use this slave by going to the job configuration. From the 'hello-world', hit *configure* and on the general tab checking the box *Restrict where this project can be run*. Enter the label for the slave you created in the box

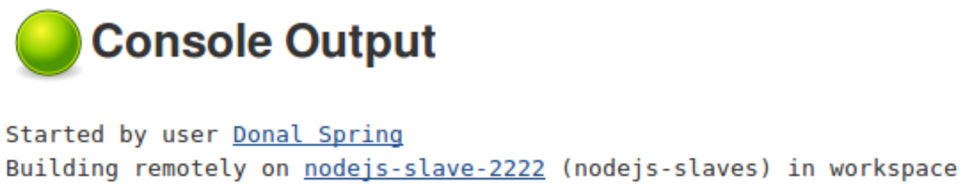


☑ Restrict where this project can be run

Label Expression: nodejs-slaves

- You will notice a "feature" of Jenkins that has been there for as long as I can remember - when you select the label name from the dropdown, you will get an error telling you that there is no such label. this is because there is a space added to the item in the list. Backspace to remove it and click off the widget and all is well.
- Run your build to see it working. The console output for the job will now show at the top which slave was used to run the build. NOTE - the first build may fail because it cannot find `git.server` Run through the next section where the hostname gets injected to the container OR relaunch your container with the flag below, where `<IP_OF_DOCKER>` is the value previously added to the `/etc/hosts`

```
--add-host="git.server:${<IP_OF_DOCKER>}"
```



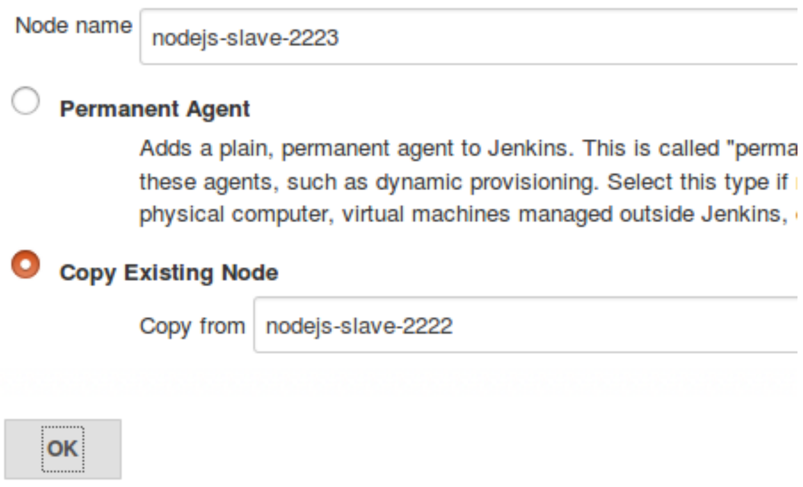
Console Output

```
Started by user Donal Spring
Building remotely on nodejs-slave-2222 (nodejs-slaves) in workspace
```

(C) DevOps for your DevOps

The objective of this lab is to be able to scale up the number of Jenkins executors and have Jenkins admin the container builds and restarts. A second slave will also be added

– Create a new slave on Jenkins by copying the config of the previous one. Go to *Node Executor Status* > *New Node.* Enter the name and copy from a previous node as shown below.

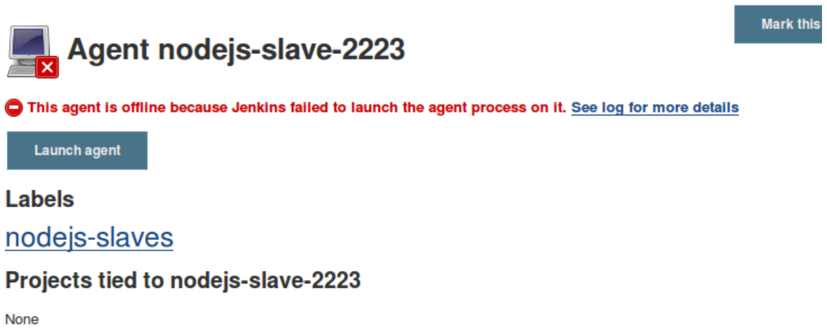


Node name: nodejs-slave-2223

Permanent Agent
Adds a plain, permanent agent to Jenkins. This is called "perma
these agents, such as dynamic provisioning. Select this type if
physical computer, virtual machines managed outside Jenkins,

[✓ Copy Existing Node]
Copy from: nodejs-slave-2222

– On the advanced tab select the new port of 2223. Save and you should see the slave cannot launch.



Agent nodejs-slave-2223

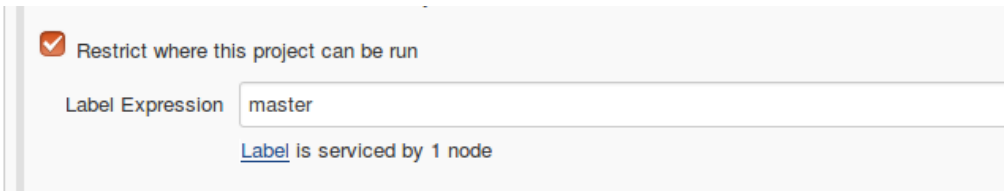
This agent is offline because Jenkins failed to launch the agent process on it. See log for more details

Labels
nodejs-slaves

Projects tied to nodejs-slave-2223
None

– Back on Jenkins home page create a *New Item* > *Freestyle Project* job. Give it a name like `'docker-nodejs-slave-build'`.

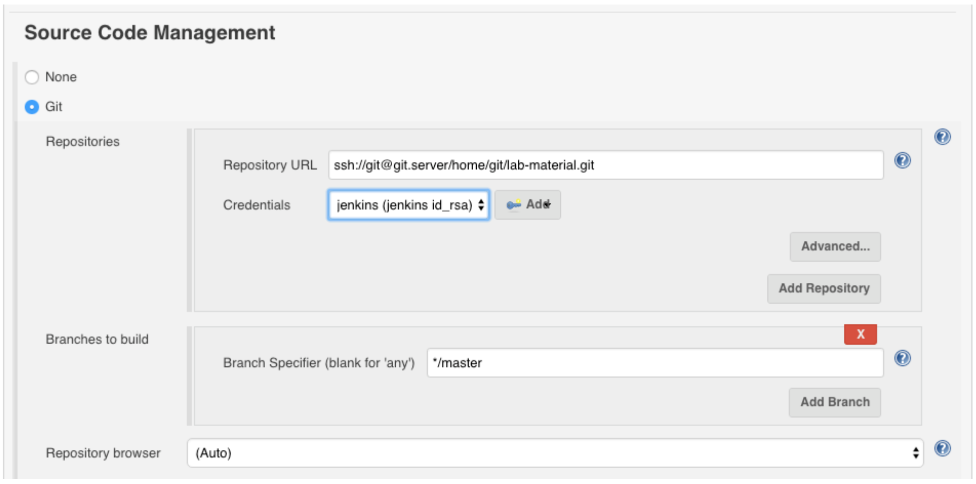
– Restrict this project to the master node as you cannot build docker images inside of a docker container



☑ Restrict where this project can be run
Label Expression: master
Label is serviced by 1 node

– In the *Source Code Management* tab enter the url `ssh://git@git.server/home/git/lab-material.git`

You will probably see an error at this point if you have not yet added the keys to jenkins. If you have added a key previously select it from the dropdown list. It will be the key called `jenkins (jenkins id_rsa)`


Source Code Management

None
Git

Repositories
Repository URL: ssh://git@git.server/home/git/lab-material.git
Credentials: jenkins (jenkins id_rsa)

Branches to build
Branch Specifier (blank for 'any'): */master

Repository browser: (Auto)

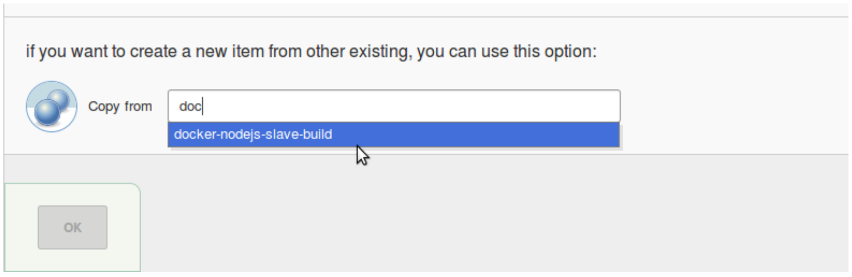
– Move to the *Build* tab and *Add Build Step* > *Execute shell.* In the box add

```
cd docker
cp ~/.ssh/id_rsa .
docker build -t jenkins-node-slave -f jenkins-node-slave  .
rm id_rsa
```

The `id_rsa` private key is added to the docker image so that the docker containers are still authorized to git clone.

– Save and run your job to rebuild your slave image.

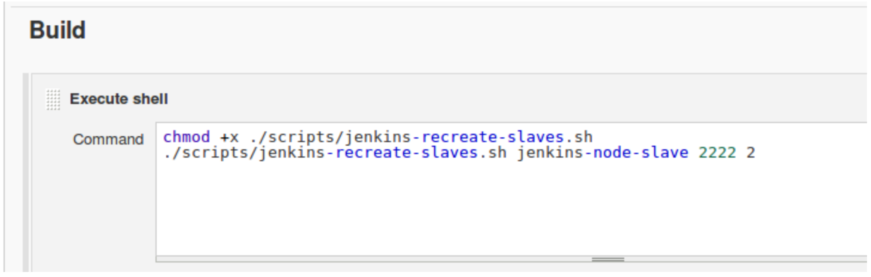
– Next build a container deploy job to trash our containers and redeploy them from jenkins UI. Create a new Item called `docker-nodejs-slave-deploy` which copies the configuration from the previously created `docker-*` job for speed.


if you want to create a new item from other existing, you can use this option:

Copy from: doc
docker-nodejs-slave-build

– Move to execute shell section and put in the following

```
chmod +x ./scripts/jenkins-recreate-slaves.sh
./scripts/jenkins-recreate-slaves.sh jenkins-node-slave 2222 2
```


Build

Execute shell

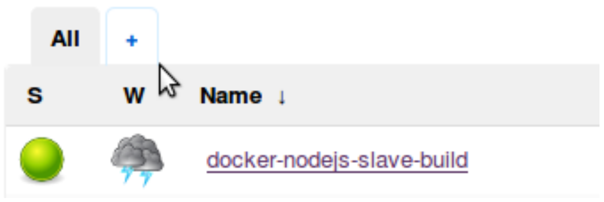
Command:
```
chmod +x ./scripts/jenkins-recreate-slaves.sh
./scripts/jenkins-recreate-slaves.sh jenkins-node-slave 2222 2
```

– Save this configuration and run the job. Once complete re-launch the slave created earlier *nodejs-slave-2223.* The slave should come online, install java into it along with the jenkins agent.

(D) Grouping jobs

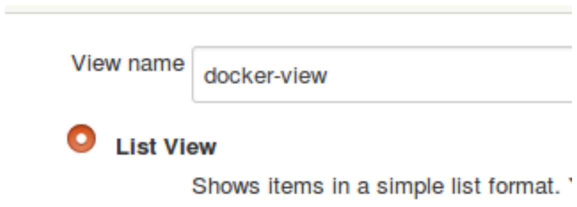
The objective of here is to create a grouping of the docker-* jobs for ease.

– On jenkins homepage click the + icon to create a new view



S	W	Name ↓
●	⛈	docker-nodejs-slave-build

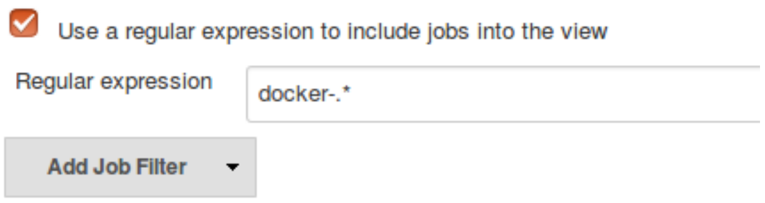
- Create a new list view with a sensible name like docker-view



View name: docker-view

List View
Shows items in a simple list format.

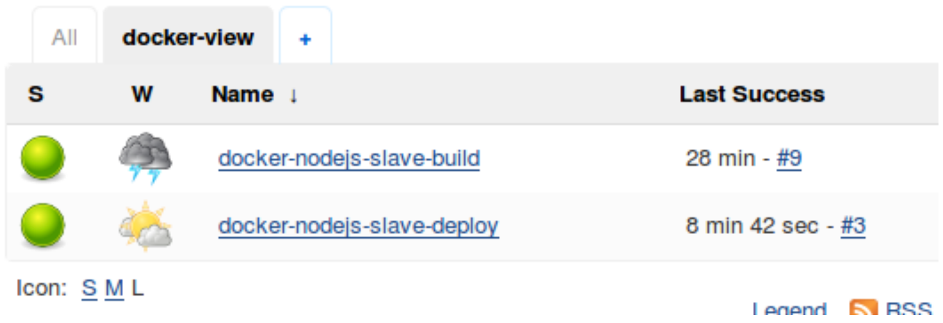
- On the config page, select regular expression and enter this `docker-.*` then save.



☑ Use a regular expression to include jobs into the view

Regular expression: docker-.*

The new view should show all the docker prefixed jobs



S	W	Name ↓	Last Success
●	⛈	docker-nodejs-slave-build	28 min - #9
●	⛅	docker-nodejs-slave-deploy	8 min 42 sec - #3

(E) Extension tasks

- Use the slave to run out the build job. NOTE you cannot build a docker image inside a docker container, so any job that uses the docker command will not run in the slaves

Comments

There are no comments.