# Setting up a test job

To test that your Jenkins server and slaves are running correctly and able to run Docker containers we will set up a basic job to test the base functionality

First from the Jenkins home page click new item

Select freestyle project and give it an appropriate name (I used test-job)

This job will not require much configuration so on the next page scroll to build click add build step and execute shell

Paste the following into the command box

*docker -v*

*docker run ubuntu /bin/echo hello world*

*docker stop $(docker ps -a -q)*

*docker rm $(docker ps -a -q)*

*docker rmi $(docker images -q)*

this shell script will print to the console the version of docker your running then start a basic Ubuntu image which prints hello world then stops all running containers, removes them and finally removes all the docker images on the slave (do not use this once you have more jobs set up as you will not want to be removing all the containers encase they are being used by other tests)

next click to save the job and click build now to test your setup

if you want you can click the arrow next to the running job and click console output to see live what is happening on the slave and check that it is running as expected (you will need to do this once the job has finished running anyway)

once the job has finished running and you are viewing the console output check that you can see in the logs the docker version, the hello world command and finally the stopping and clean-up of containers

if this all ran as expected then you have proven that you are able to pull and run docker images on your slave

once you have this job passing it is recommended to make some modifications to it to allow you to run it to test your slaves without interfering with other jobs you can do this by modifying the shell command and giving the container a name then only removing the named container and the specific image that you created an example of this is below

*docker -v*

*docker run -d --name testUbuntu ubuntu /bin/echo hello worlddocker stop $(docker ps -a -q)*

*docker logs testUbuntu*

*dockrr stop testUbuntu*

*docker rm testUbuntu*

*docker rmi Ubuntu*

if you make these changes make sure to rerun the job to ensure that is still passes

# Pulling from a git repo

To create a project based on a git repo it is similar to the test job that we set up, first create an appropriately named freestyle job but this time under source code management select git (if you cannot see git in the list of providers you will want to check that the git plugin has been successfully installed and resolve any dependency errors that it might have)

Next input the url of the git project you wish to clone. If your project requires you to be logged in to download it then you create a credentials object using the dropdown below the url box. And specifying the details of the user you want Jenkins to use (it is recommended that you create Jenkins its own user for this purpose)

In the next box, you can specify what branch you want to check out when the git repository is cloned

To check this is able to pull down the repo correctly just add a shell script to the build commands that’s runs the command ‘ls’ this way when you run the job you can check in the console that you see the files for the repo that you expected

# Building a Docker image

# Running unit tests

# Pushing a Docker image to repository

# Deploying a image to a rancher server

# Running database migrations

# Running acceptance tests

# Running accessibility tests

# Running performance tests