LAB

Docker Images Part 1

Lab Objectives

This lab demonstrates how to search through the repositories located on Docker Hub using the web browser as well as the Command Line Interface. This lab then walks through the steps to deploy WordPress All-In-One and add the WordPress CLI.

Lab Structure - Overview

1. Search Docker Hub for Images
2. Deploy WordPress All-In-One
3. Add WordPress CLI

Lab Overview

Conventions

Lab Guide Conventions

`reboot`	Any text a student needs to enter is printed like this.
`<your.ip>`	Any time a student needs to insert their own value, the text has brackets.
File	User Interface (UI) buttons and objects are bold.
Special Font	Unusual or important words or phrases are marked with italics.

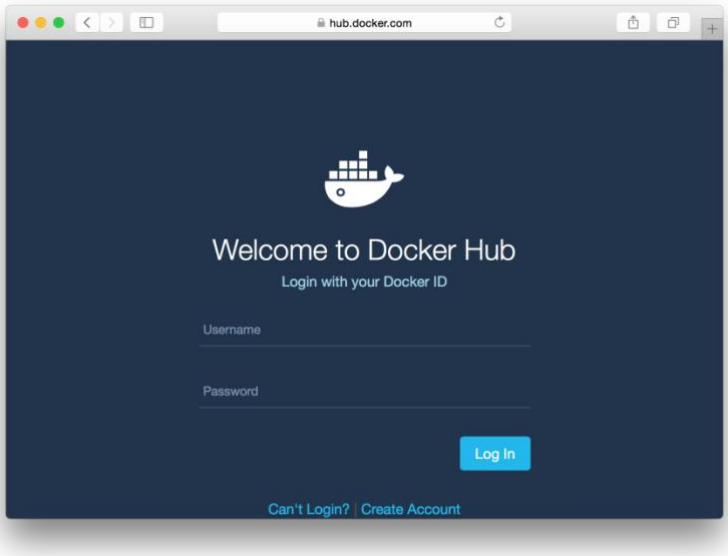
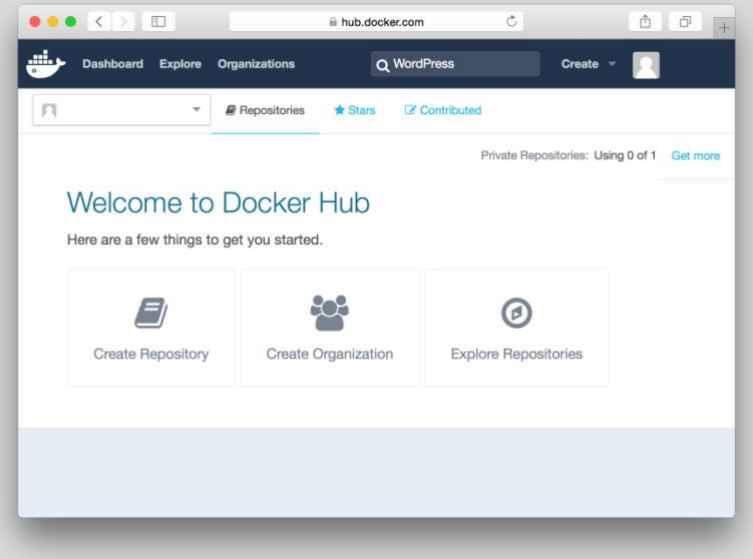
Code Blocks

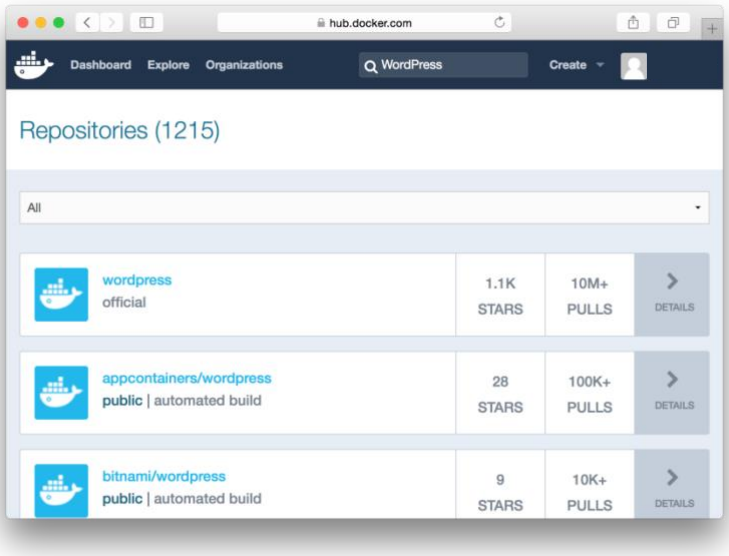
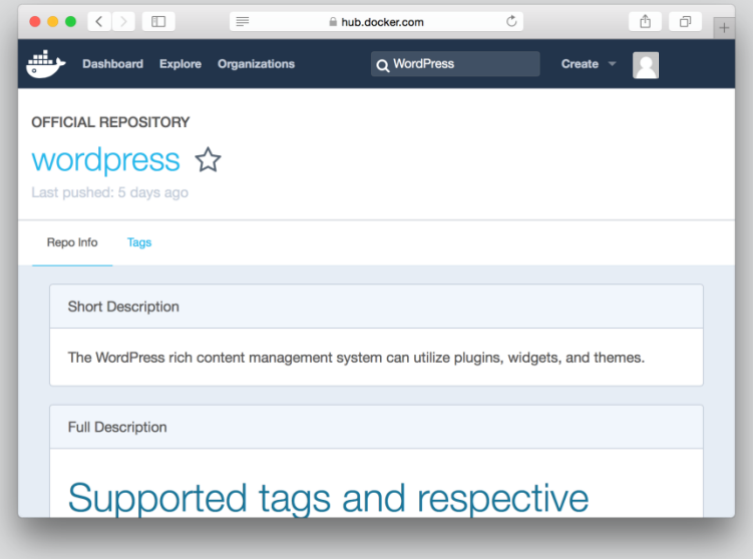
Blocks of sample code are set apart from the body and marked accordingly. It is recommended that students do not copy/paste text from the lab into their files. Extra formatting is often transferred in this process and can result in failed operations.

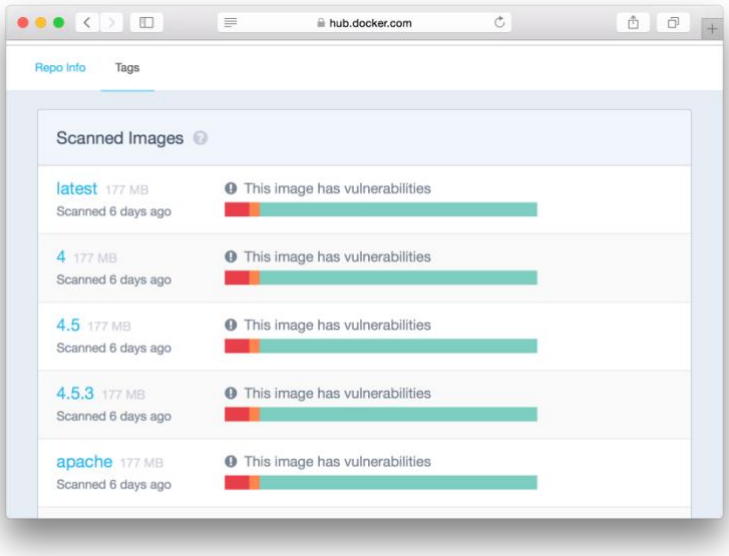
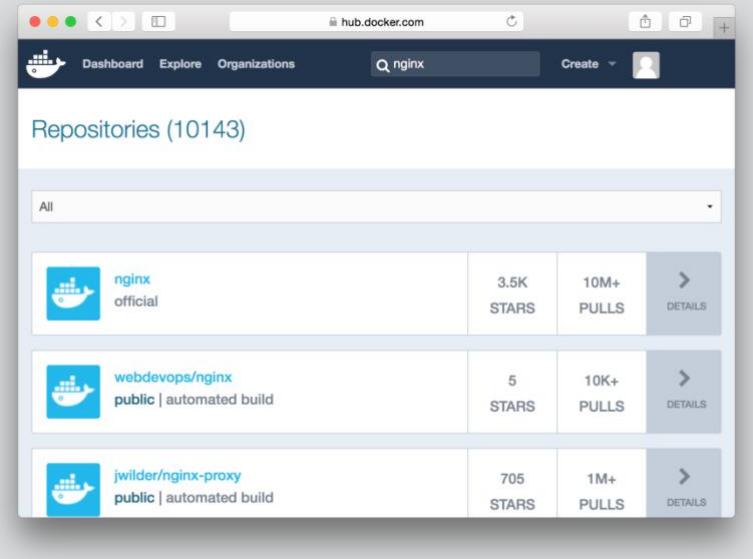
```
# ls -l /var/www/html/index.html
-rw-rw-r--  1 root root 1872 Jun 21 09:33 /var/www/html/index.html
# date
Wed Jun 21 09:33:42 EDT 200
```

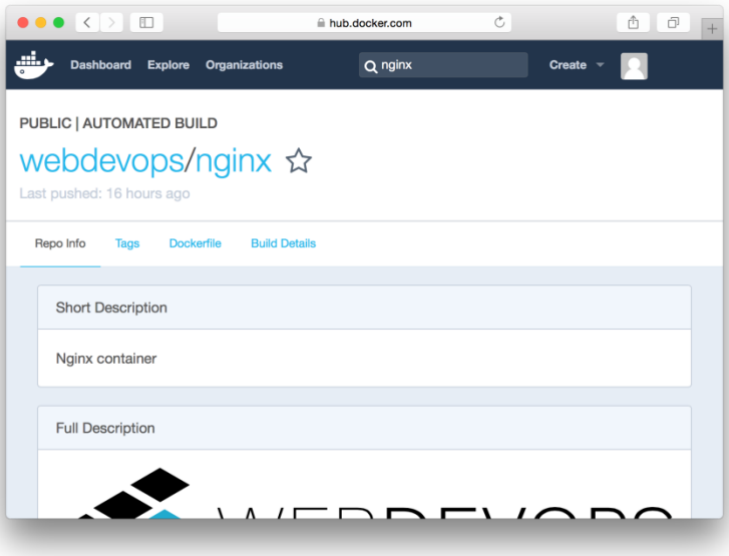
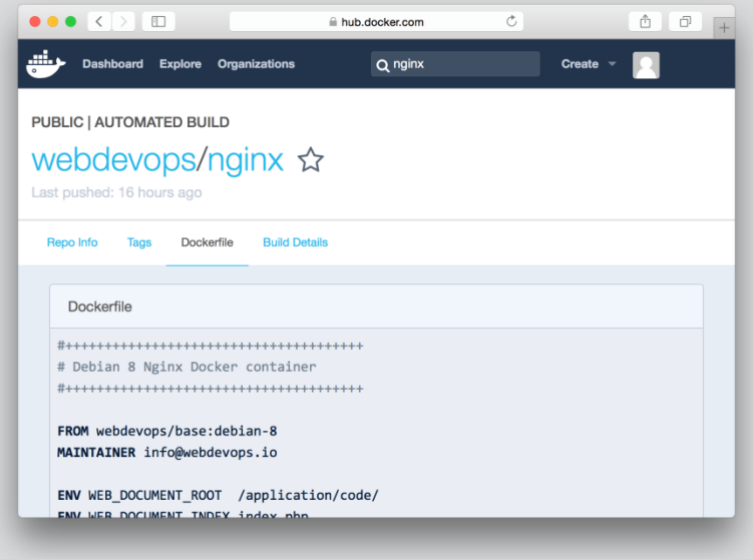
1. Search Docker Hub for Images

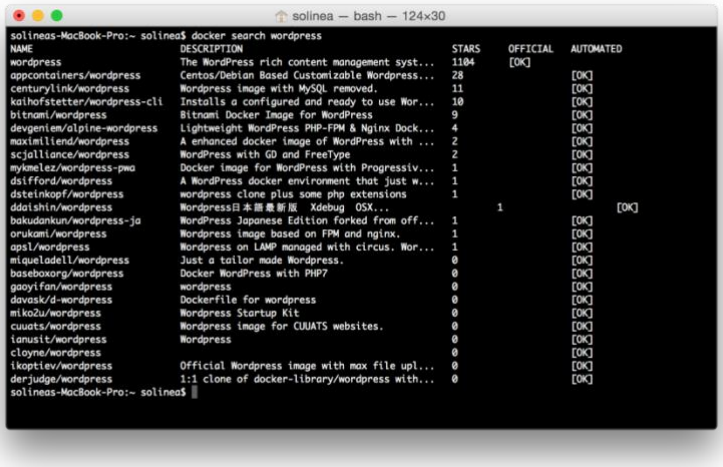
Step by Step Guide

Step	Action
1.	In a web browser, navigate to http://hub.docker.com and log in to Docker Hub. Creating a Docker Hub account is free and simple if you do not have one already. 
2.	Once logged in to Docker Hub, use the **Search** bar at the top to locate the official **WordPress** image. 

Step	Action
3.	Select the **wordpress** image with the official tag below it. 
4.	Here you can see all the information about this repository. Next, click on the **Tags** tab. 

Step	Action
5.	This tab shows a list of all the images that have been tagged in this repository. 
6.	Using the **Search** function, locate the list of **nginx** repositories. Select the **webdevops/nginx** repo to see more details. 

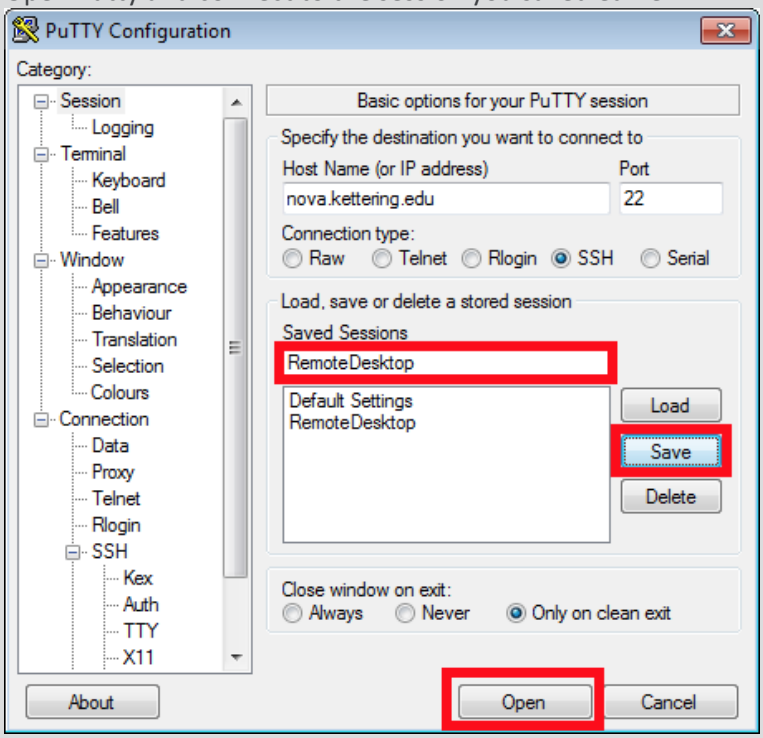
Step	Action
7.	Select the **Dockerfile** tab to view the Dockerfile for this build. 
8.	This is where the information about the base image and repository maintainer is located, as well as the build code. 

Step	Action
9.	You can also search for an image using the CLI. From a command line running Docker, enter `docker search wordpress` This will list all of the repositories within Docker Hub without having to go through the web browser. 

2. Deploy WordPress All-In-One and the WordPress CLI

Step by Step Guide

Step	Action
1.	Locate the IP address of the **Master** machine within the lab folder.

Step	Action
2.	If on a Mac, or using Linux: In a command line, enter `ssh -i /path/to/k8s-lab ubuntu@<MasterIP>` The **SSH** file will be provided by the instructor for this lab. This command will connect the console to the Docker machine. If using Windows: Open Putty and connect to the session you saved earlier. 
3.	In the command line, enter the following command to deploy the **WordPress All-In-One** container. In the following steps, this WordPress all-in-one will receive a WordPress tool called the "WordPress CLI." This tool allows WordPress administrators to interact with their WordPress deployment using the command-line. `docker run -d -P --name wpaio s5atrain/wordpress:aio` The output will be a container ID.

4.	Run the following command to validate that WordPress CLI is not installed.
	`docker exec wpaio wp theme list --allow-root --path='/var/www/html'`
	The output will contain an error message. This is because the **WordPress CLI** is not yet installed.
	`"rpc error: code = 2 desc = oci runtime error: exec failed: exec: "wp": executable file not found in $PATH`

Step	Action

5.	Run the docker port command to get the mapped ports of the WordPress all-in-one container.
	`docker port wpaio`
	Two ports will be listed:
	`3306/tcp -> 0.0.0.0:32769`
	`80/tcp -> 0.0.0.0:32770`

6.	In a web browser, navigate to http://<MASTER_IP>:port and the port mapped to 80/tcp from the previous step (i.e. 32770).
7.	Configure **WordPress** with the following credentials: Username: root Password: root
8.	Use the **docker exec** command to gain remote access to the WordPress container and run some configuration commands. `docker exec -it wpaio /bin/bash`

9.	Once inside the **WordPress** container, enter the following commands in following sequence to install the WordPress CLI tool.
	`apt-get update`
	`apt-get install wget -y`
	`wget https://raw.githubusercontent.com/wp-cli/builds/gh-pages/phar/wp-cli.phar`
	`mv wp-cli.phar /usr/local/bin/wp && chmod +x /usr/local/bin/wp`
	`cd /var/www/html`
	`wp theme list --allow-root`
	The command line will return the following output and validate that the WordPress CLI has been installed on the running container.
	`No value for $TERM and no -T specified`
	`+----------+----------+----------+---------+`
	`| name | status | update | version |`
	`+----------+----------+----------+---------+`
	`| twentyeleven | inactive | available | 2.3 |`
	`| twentyfifteen | active | available | 1.4 |`
	`| twentyfourteen | inactive | available | 1.6 |`
	`| twentyten | inactive | none | 2.1 |`
	`| twentythirteen | inactive | available | 1.8 |`
	`| twentytwelve | inactive | available | 1.9 |`
	`+----------+----------+----------+---------+`

10.	Now exit the **WordPress** container: `exit`

3. Test the WordPress CLI Install

Step by Step Guide

This process will take approximately 5 minutes.

Step	Action

1.	Run the command from the earlier step and validate that the WordPress CLI is executable using the docker exec command.
	docker exec wpaio wp theme list --allow-root --path='/var/www/html'
	If it is working correctly, it will output this:
	`name status update version`
	`twentyeleven inactive available 2.3`
	`twentyfifteen active available 1.4`
	`twentyfourteen inactive available 1.6`
	`twentyten inactive none 2.1`
	`twentythirteen inactive available 1.8`
	`twentytwelve inactive available 1.9`

2.	Now commit the **WordPress CLI** container to an image. This will commit the read/write layer of the running container to an image in the local image cache. **docker commit -m "added wpcli" wpaio <dockerhubusername>/wordpress-cli:aio-manual** The output will be a new image ID.

3.	Log in to your account on Docker Hub, so that you can access the registry:
	docker login
	You will need to enter the username and password you used when you created your account, as follows:
	`docker login`
	`Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.`
	`Username: <dockerhubusername>`
	`Password: *****`
	`Login Succeeded`

4.	Using the new image created in the previous step, enter **docker push <dockerhubusername>/wordpress-cli:aio-manual**

Step	Action
5.	Deploy a container from the newly created image, this container will create a read/write layer above the previous layer which carries the installed WordPress CLI. `docker run -d -P --name wpaio2 <dockerhubusername>/wordpress-cli:aio-manual`

6.	Run the following to get the dynamic port mapping of the newly deployed container.
	`docker port wpaio2`
	Two ports will be listed:
	`3306/tcp -> 0.0.0.0:32769`
	`80/tcp -> 0.0.0.0:32770`

7.	In a web browser, navigate to http://<MASTER_IP>:port and the port mapped to 80/tcp from the previous step (i.e. 32770).
8.	Configure WordPress again with the same credentials as before: Username: root Password: root

9.	Test the install again with the same execute command as before:
	`docker exec wpaio2 wp theme list --allow-root --path='/var/www/html'`
	The output should remain the same:
	`name twentyeleven status inactive update available version 2.3`
	`twentyfifteen active available 1.4`
	`twentyfourteen inactive available 1.6`
	`twentyten inactive none 2.1`
	`twentythirteen inactive available 1.8`
	`twentytwelve inactive available 1.9`
	This validates that the **WordPress CLI** is already installed on the image. The image wordpress-cli:aio-manual was created from the read/write layer of the previous container wpaio; which, had the WordPress CLI installed on it.

10.	Return to view your Docker Hub account in a web browser. Find the repository for the image you pushed, and view the information you can now edit. What statistics is Docker Hub reporting for the repository?

Lab Complete!