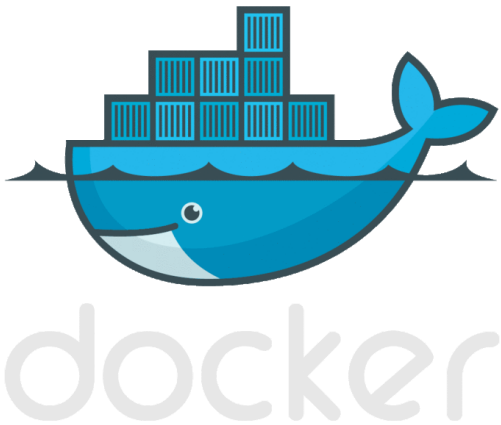


How to deploy Java apps with Docker (a quick tutorial)

By Atlassian (https://www.atlassian.com/blog/author/atlassian)

January 13, 2017

For those new to Docker, let me say "Welcome to the party!" It's an easy way to deploy, run, and manage applications using vm-like containers that are independent

of elements like hardware and language, which makes these containers highly portable. And it's all the rage.

So how do you deploy Java apps using Docker? You're in the right place. I'll walk through the process step by step – from installing Docker, to installing Java inside a Docker container, to deploying and running an app.

Setting up for the tutorial

Docker supports just about any OS you'd care to use:

- Linux
- Mac
- Windows
- Windows Server
- AWS
- Azure

I'll be using Linux for this demonstration. For the app, I'll use Bitbucket Server (https://www.atlassian.com/software/bitbucket/server) (Atlassian's behind-your-firewall Git repository manager), but you can substitute any other Java application you like.

On any new box I need vim, curl, and Git to deploy my .dotfiles – otherwise I can't function properly.

```
sudo apt-get install vim curl git
curl -Lks http://j.mp/durdn-cfg | bash
```

> Speaking of Git, did you know we offer a free Git tutorial site? It's true! Check it out. (http://www.atlassian.com/git)

Step 1: install Docker

Installing Docker is easy. First we install some kernel extensions needed for it to run:

```
sudo apt-get install linux-image-extra-$(uname -r)
```

Then we install *software-properties-common* which provides us with *add-apt-repository*:

```
sudo apt-get install software-properties-common
```

Add the dotcloud Personal Package Archive (PPA):

```
sudo add-apt-repository ppa:dotcloud/lxc-docker
sudo apt-get update
```

And finally install Docker with:

```
sudo apt-get install lxc-docker
```

Now we are ready to pull a base image which will be the foundation of all our work: `docker pull base`

This will output:

```
Pulling repository base from https://index.docker.io/v1
Pulling image 27cf784147099545 () from base
Pulling 27cf784147099545 metadata
Pulling 27cf784147099545 fs layer
Downloading 94863360/? (n/a)
Pulling image b750fe7[...]2b4accb2c21d589ff2f5f2dc (ubuntu-quantl) from base
Pulling b750fe79269d2[...]f05b433b1d1a02a62b4accb2c21d589ff2f5f2dc metadata
Pulling b750fe79269d2[...]3ef05b4332b1d1a02a62b4accb2c289ff2f5f2dc fs layer
Downloading 10240/? (n/a)
```

Note that any Docker command requires that your machine is running the Docker daemon. To run the Docker daemon in the background in case it's not already, simply type:

```
sudo docker -d &
```

You can pull any public image published on the Docker index or publish your own. And you can inspect all the images that Docker has at its disposal with:

```
docker images
```

This will list:

```
REPOSITORY TAG ID CREATED
base ubuntu-quantl b750fe79269d 10 weeks ago
base latest b750fe79269d 10 weeks ago
base ubuntu-quantal b750fe79269d 10 weeks ago
base ubuntu-12.10 b750fe79269d 10 weeks ago
```

Which means we have a base image based on Ubuntu 12.10 ready for our perusal.

Step 2: install Java inside Docker

Now we're on our way to creating an image and customizing it to our needs. The first requirement is to install Java. Let's spin up a shell in the base container:

```
docker run -i -t base /bin/bash
```

This starts a new container, gives it a unique id, assigns it an IP address and setups networking to it. We're greeted with a root shell:

```
root@298af82e71ef:/#
```

Now in this container – which is a lightweight and sealed vm-like thing separate from all the rest – we can install our dependencies:

```
apt-get install software-properties-common
```

Add the PPA that will allow us to install Java:

```
add-apt-repository ppa:webupd8team/java
apt-get update
```

Then finally, install Java:

```
apt-get install git curl oracle-java7-installer
```

(*Note that you may have to accept the license manually.)

Ok. Now we can create a commit to save the state of this container in an image.

Exit the container (with *exit* or *CTRL-d*) and list the containers – dead or alive with *docker ps -a*, which outputs:

```
ID IMAGE COMMAND CREATED STATUS PORTS
8e07a84ea97a base:latest /bin/bash 12 minutes ago Exit 0
fecada4ce303 base:latest /bin/bash 17 minutes ago Exit 0
9cb541022c5b base:latest /bin/bash 25 minutes ago Exit 127
a2914a38394d durdn/base:latest /bin/bash 26 minutes ago Exit 0
6fa304872025 durdn/base:latest /bin/bash 30 minutes ago Exit 0
3e0241227129 durdn/base:latest /bin/bash 30 minutes ago Exit 0
98b400fcb5dc durdn/base:latest /bin/bash 31 minutes ago Exit 0
88a113234c47 base:latest /bin/bash 36 minutes ago Exit 0
```

From the list of recent containers we take the most recent, where we installed all the dependencies. Now we can commit a new snapshot/image with *docker commit*:

```
docker commit 8e07a84ea97a durdn/java7
```

And as you can see, it is now listed if I type *docker images*:

```
REPOSITORY TAG ID CREATED
durdn/java7 latest ab6396541f9a 2 hours ago
base ubuntu-quantal b750fe79269d 10 weeks ago
base ubuntu-quantl b750fe79269d 10 weeks ago
base latest b750fe79269d 10 weeks ago
base ubuntu-12.10 b750fe79269d 10 weeks ago
```

Step 3: install the app in your Docker container

Now we can install our Java application – in this case Bitbucket Server (https://www.atlassian.com/software/bitbucket/server) – in our newly created *durdn/java7* image.

We can add content to an image in several ways: the *docker insert* command is one, but we can also use *curl*, *wget*, etc.

Spin up a shell in the newly created *durdn/java7* image opening a mirror port *7990* from the container to the host and with a persistent home where the data will be stored:

```
docker run -i -t -p :7990 -v /opt/stash-home durdn/java7 /bin/bash
root@298af82e71ef:/#
```

Download Bitbucket Server. (Note that when this post was originally published, Bitbucket Server was called "Stash". You'll see that here in some of the file and directory names, as well as the screenshot below

```
root@298af82e71ef:/# curl -Lks
https://www.atlassian.com/software/stash/downloads/binary/atlassian-
bitbucket-4.13.0.tar.gz -o /root/stash.tar.gz
```

Unpack stash, create and export *STASH_HOME* folder:

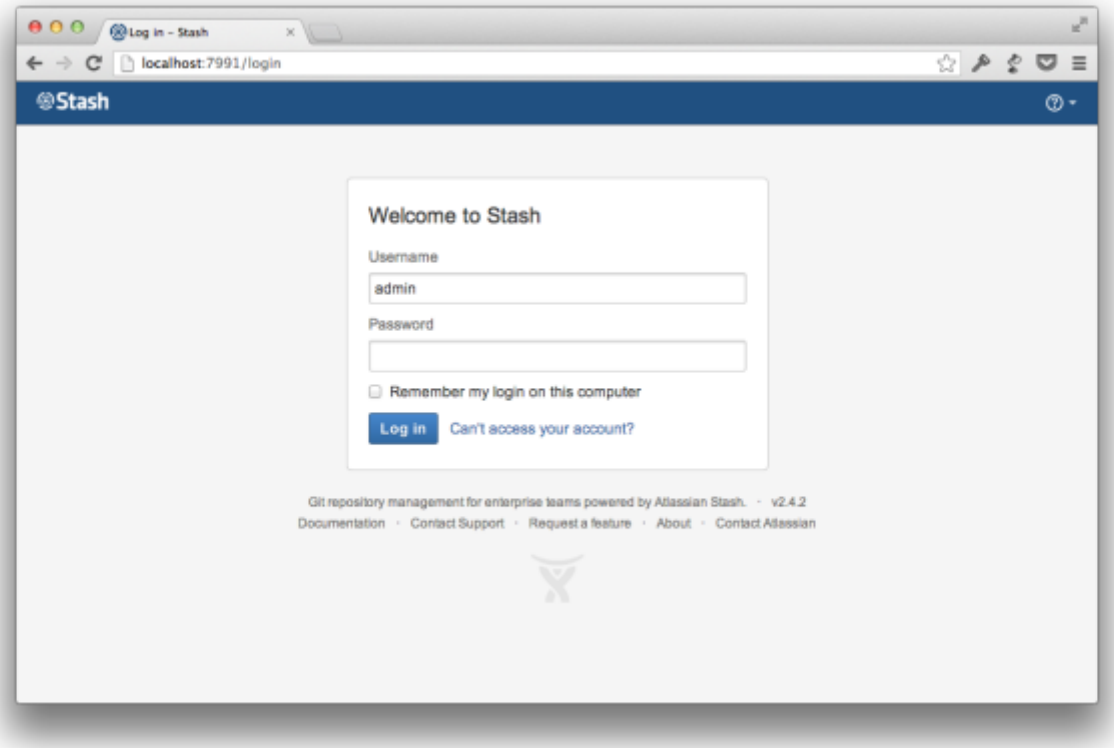
```
root@298af82e71ef:/# mkdir /opt/stash
root@298af82e71ef:/# tar zxf /root/stash.tar.gz --strip=1 -C /opt/stash
root@298af82e71ef:/# mkdir /opt/stash-home
root@298af82e71ef:/# export STASH_HOME=/opt/stash-home
```

To avoid startup errors we must add the unique id of this host *298af82e71ef* to the */etc/hosts* directory.

Now we can start the app:

```
root@298af82e71ef:/# /opt/stash/bin/start-stash.sh -fg
```

Check that Stash Bitbucket Server is running in the container by accessing *http://localhost:7990/stash*. It works:



We can now exit this container and commit it so that we can reuse the work we did:

```
docker commit aec2feb8cdea durdn/stash
effd5d47b34f
```

Where *aec2feb8cdea* was the ID of the last modified container in *docker ps -a*. The result is that we have a new image called *durdn/stash* with Bitbucket Server installed:

```
REPOSITORY TAG ID CREATED
durdn/java7 latest ab6396541f9a 2 hours ago
durdn/stash latest effd5d47b34f 3 seconds ago
base ubuntu-quantal b750fe79269d 10 weeks ago
base ubuntu-quantl b750fe79269d 10 weeks ago
base latest b750fe79269d 10 weeks ago
base ubuntu-12.10 b750fe79269d 10 weeks ago
```

A note about ephemeral containers and persistent storage

Containers are ephemeral. Once torn down, they will be reset entirely to their snapshot state!

This means your app installation will be reset every time you stop the container. To make sure data is preserved between runs we can use *volumes*, which will be shared and persisted between containers. We do this simply by reusing the volumes from older containers using the option *-volumes-from*.

Let's start Bitbucket in the container interactively (*-i* runs the command interactively and *-t* attaches a tty to it):

```
docker run -i -t -p :7990 -volumes-from aec2feb8cdea durdn/stash /bin/bash -c 'STASH_HOME=/opt/stash-home /opt/stash/bin/start-stash.sh -fg'
```

This command will start the application in foreground. If you want the container to run in the background, use the *-d* flag:

```
docker run -d -p :7990 -volumes-from aec2feb8cdea durdn/stash /bin/bash -c 'STASH_HOME=/opt/stash-home /opt/stash/bin/start-stash.sh -fg'
```

Voilá! Your application instance – with data persisted – is now running in the background as you can see by running *docker ps*.

More Docker goodies to explore

What I like most about Docker is how responsive and quick it is, and the instant repeatability it offers. Spinning up a new container takes literally the same time it takes to run the command on bare metal. It's a joy to behold! Since this post was originally published in 2013, my fellow Atlassians and I have written several more articles about Docker. What can I say?… we're in love. You might find some of them useful:

- Docker all the things: automation and wiring (https://blogs.atlassian.com/2013/11/docker-all-the-things-at-atlassian-automation-and-wiring/)
- Building web apps with Docker and Bamboo (https://blogs.atlassian.com/2015/09/bamboo-docker-building-web-apps)
- Docker containers, Bamboo, and winning at continuous delivery (https://blogs.atlassian.com/2015/06/docker-containers-bamboo-winning-continuous-delivery/)
- Common Dockerfile mistakes (https://developer.atlassian.com/blog/2016/06/common-dockerfile-mistakes/)
- Docker clusters from the ground up: front-end reverse proxy (https://developer.atlassian.com/blog/2016/01/docker-cluster-reverse-proxy-1/)

And for the truly container-addicted, there's Bitbucket Pipelines – the easiest way to build, test, and deploy from your Bitbucket repo. Builds run in isolated Docker containers on infrastructures that Atlassian manages, so there's no need to set up and configure your own build agents. I highly recommend it.

Take a tour of Bitbucket Pipelines (https://bitbucket.org/product/features/pipelines)

READ MORE IN:

Git (Https://Www.Atlassian.Com/Blog/Git), Software (Https://Www.Atlassian.Com/Blog/Software-Teams)

READERS ALSO CHECKED OUT:

BITBUCKET

Stash on Docker

(https://www.atlassian.com/blog/bitbucket/stash-docker)

ARCHIVES

Docker all the things at Atlassian: automation and wiring

(https://www.atlassian.com/blog/archives/docker-all-the-things-at-atlassian-automation-and-wiring)

ARCHIVES

Announcing Docker Automated Builds on Bitbucket

(https://www.atlassian.com/blog/archives/docker-automated-builds-bitbucket)