



Creating a Docker Image with Ubuntu and Java

by Ivan K · Aug. 11, 15 · Cloud Zone

Deploy and scale data-rich applications in minutes and with ease. Mesosphere DC/OS includes everything you need to elastically run containerized apps and data services in production.

This short article shows how to create a Docker image with Ubuntu 15.04 and that has Java 7 or 8 installed. It will act as a base for some future articles that I have planned to write.

Yes, I am aware that there are several Docker images with Java 8 on Docker Hub.

Prerequisites

When writing this article, I have used Docker 1.7.1 and Docker-Machine 0.3.0. Instructions on how to install these can be found [here](#).

Docker File

The Docker documentation recommends that Docker files are to be located in a dedicated directory which only contains the files necessary for the creation of the Docker image.

I have thus created a directory named “krizsan-ubuntu1504java8” for the Docker image with Java 8 and another directory named “krizsan-ubuntu1504java7” for the image with Java 7. Each of these directories are to contain a file named “Dockerfile” (note: no file extension!).

The Docker file for Java 8 on Ubuntu 15.04 has the following contents:

```
1  # Ubuntu 15.04 with Java 8 installed.
2  # Build image with:  docker build -t krizsan/ubuntu1504java8:v1 .
3
4  FROM ubuntu:15.04
5  MAINTAINER Ivan Krizsan, https://github.com/krizsan
6  RUN apt-get update && \
7      apt-get upgrade -y && \
8      apt-get install -y  software-properties-common && \
9      add-apt-repository ppa:webupd8team/java -y && \
10     apt-get update && \
11     echo oracle-java7-installer shared/accepted-oracle-license-v1-1 select true | /usr/bin
12     apt-get install -y oracle-java8-installer && \
13     apt-get clean
```

The short version is that this Docker file will create a Docker image based on the ubuntu:15.04 Docker image in which Oracle's Java 8 is installed.

Long version, row-by-row:

- FROM ubuntu:15.04
States which Docker image this Docker image is to be based upon. Compare to parent class in object-oriented programming.
- MAINTAINER Ivan Krizsan, <https://github.com/krizsan>
Tells us who is maintaining this Docker image.
- The Docker RUN command executes shell commands to install programs, configure the Docker image etc.
Details follow below on the purpose of each shell command.
- apt-get update
Updates the list of available packages and their versions.
- apt-get upgrade
Installs any newer versions of currently installed packages.
- apt-get install -y software-properties-common
Installs software in order to allow for independent vendor software sources, in particular the "add-apt-repository" command that is used in the next line.
- add-apt-repository ppa:webupd8team/java -y
Adds the PPA repository that contains Oracle Java 8, assuming yes to all queries.
- apt-get update
Again, updates the list of available packages and their versions. The reason for running an update again is that a new repository was just added and we want to refresh the list of available packages and their versions, including those in the new repository.
- echo oracle-java7-installer shared/accepted-oracle-license-v1-1 select true | /usr/bin/debconf-set-selections
The Java installer require you to accept a license before the installation begins. This line automates accepting the license.
- apt-get install -y oracle-java8-installer
Finally Java 8 can be installed! The -y flag runs the installation in a non-interactive mode, assuming that all questions are to be answered "yes".
- apt-get clean
Cleans the local repository from retrieved package files.

The reason for having one single Docker RUN command and chaining the shell commands using && is that Docker will create one additional layer on the image for each RUN command. The fewer RUN commands that a Docker file contains, the smaller will the resulting image be.

The Docker file that will create an image with Java 7 on Ubuntu 15.04 looks like this:

```
1 # Ubuntu 15.04 with Java 7 installed.
2 # Build image with: docker build -t krizsan/ubuntu1504java7:v1 .
3
```

```

3
4 FROM ubuntu:15.04
5 MAINTAINER Ivan Krizsan, https://github.com/krizsan
6 RUN apt-get update && \
7     apt-get upgrade -y && \
8     apt-get install -y software-properties-common && \
9     add-apt-repository ppa:webupd8team/java -y && \
10    apt-get update && \
11    echo oracle-java7-installer shared/accepted-oracle-license-v1-1 select true | /usr/bin
12    apt-get install -y oracle-java7-installer && \
13    apt-get clean

```

The only difference compared to the Java 8 Docker file, apart from the comments, is the line second-to-last in which the Java 7 installer is used.

Build the Java 8 Docker image using the following steps:

- Open a terminal window.
- If you are not using a Linux operating system, create a Docker-Machine (if you don't already have one):
`docker-machine create dev`
- If you are not using a Linux operating system, start the Docker-Machine:
`docker-machine start dev`
- If you are not using a Linux operating system, ssh into the Docker-Machine:
`docker-machine ssh dev`
- Go to the directory containing the Docker-file which describes the image you want to build. In this example, I will build the Java 8 image.
- Build the Docker image:
`docker build -t krizsan/ubuntu1504java8:v1 .`
 Note the period at the end of the line!
 Linux users will need to add the sudo command before the docker command:
`sudo docker build -t krizsan/ubuntu1504java8:v1 .`
 You will see quite some console output that should end with something like this:

```

1  ...
2  update-alternatives: using /usr/lib/jvm/java-8-oracle/bin/xjc to provide /usr/bin/xjc (xjc
3  Oracle JDK 8 installed
4  ...
5  invoke-rc.d: policy-rc.d denied execution of start.
6  Setting up xfonts-encodings (1:1.0.4-2) ...
7  Setting up xfonts-utils (1:7.7+2) ...
8  Setting up gsfonts-x11 (0.22) ...
9  Processing triggers for libc-bin (2.21-0ubuntu4) ...
10 Processing triggers for systemd (219-7ubuntu6) ...
11 ---> d3c43c2de29e
12 Removing intermediate container f417de6f01c6

```

```
12 Removing intermediate container 1447de070100
13 Successfully built d3c43c2de29e
```

- Start a Docker container using the newly created Docker image:
`docker run -it krizsan/ubuntu1504java8:v1 bash`
 Again, Linux users need to insert the `sudo` command before the `docker` command.
- Verify the Java version installed:
`java -version`
 In my case I see that Java version 1.8.0_51 is installed.
- Exit the Docker container:
`exit`

Finally, while still being let's examine the size of the image that was created:

- List the locally available Docker images:

`docker images`

I see the following information about the Docker image we just created listed:

	REPOSITORY	TAG	IMAGE ID	CREATED	VIR
1	◀				▶
	krizsan/ubuntu1504java8	v1	3f1e02768a70	16 seconds ago	802
2	◀				▶

The size of the image is on the right, in the VIRTUAL SIZE column, and, as you can see is around 803MB.

This concludes this article and I am now ready to create further Docker images containing software that runs on Java.

Discover new technologies simplifying running containers and data services in production with this free eBook by O'Reilly. Courtesy of Mesosphere.

Like This Article? Read More From DZone



Deployment Automation of Docker WebLogic Cluster on Any Cloud



Java EE Deployment Scenarios for Docker Containers



Push Spring Boot Docker Images on ECR



**Free DZone Refcard
Getting To Know Google Compute Engine And How To Use It**

Topics: DOCKER , JAVA , CLOUD , CONTAINERS

Published at DZone with permission of Ivan K. [See the original article here.](#)

Opinions expressed by DZone contributors are their own.

Get the best of Cloud in your inbox.

Stay updated with DZone's bi-weekly Cloud Newsletter. SEE AN EXAMPLE

SUBSCRIBE

Cloud Partner Resources

Networking for Docker Containers: 3 Part Series

Mesosphere



Free report: How Mesos is enabling the adoption of containers and big data services

Mesosphere



Three key metrics for monitoring AWS SNS performance and usage

Site 24x7



Whitepaper: Site24x7: Powerful, Agile, Cost-Effective IT Management from the Cloud

Site 24x7



Configure Bitbucket Webhook to Trigger Jenkins Builds on AWS EC2

by Ajitesh Kumar MVB · Aug 04, 17 · Cloud Zone

Site24x7 - Full stack It Infrastructure Monitoring from the cloud. Sign up for free trial.

This article represents the steps required to configure BitBucket Webhooks to trigger Jenkins Builds on AWS EC2 based on code committed in the repository. This essentially means that a code commit in the BitBucket code repository would trigger a build on the Jenkins server running on a AWS EC2 machine. This forms the starting point of a continuous delivery pipeline. The Jenkins build, when triggered as a result of code push, could perform tasks such as some of the following:

- Run the build.
- Run tests.
- Publish build artifacts in artifactory.
- Deploy the build artifacts in different environments including QA, UAT, and production.

Please feel free to comment/suggest if I missed one or more important points.

Following are the steps which need to be taken to configure the push-based trigger of a Jenkins build:

- **Configure Inbound Rule on Jenkins Server on AWS EC2:** One needs to add the inbound rule on the Jenkins box on AWS EC2 to follow this IP: 104.192.143.192/28. This is required to allow requests coming from BitBucket to reach the Jenkins server on AWS EC2. This is the BitBucket cloud IP outbound address (for hooks like POST), while the inbound IP address for BitBucket is 104.192.143.1. The IP (104.192.143.1) can be figured out by pinging bitbucket.org.

The outbound IP address can be found on this page

THE OUTBOUND IP ADDRESS CAN BE FOUND ON THIS PAGE.

- **Configure Webhook in BitBucket:** One needs to add the following URL while creating the WebHook in BitBucket: `http://jenkins_server_uri/bitbucket-hook/`.
- **Configure SCM in Jenkins Project:** The next step is to configure Git as a source code management tool within the Jenkins project by clicking on the *Configure* link on the project's dashboard. Following is a sample of input parameters:
 - Repository_URL:
`https://<bitbucket_user>@bitbucket.org/<bitbucket_user>/<bitbucket_project_name>.git`
 - Credentials: Create a credential by providing a BitBucket username and password.
- **Setup Build Triggers in Jenkins Project:** Finally, one would need to setup build triggers for the Jenkins project by clicking on the *Configure* link on project's dashboard. Check the option "Build when a change is pushed to BitBucket."

Site24x7 - Full stack It Infrastructure Monitoring from the cloud. Sign up for free trial.

Like This Article? Read More From DZone



DevOps Tools for Continuous Delivery in Jelastic Private Cloud. Part 2



Yelp's Distributed System for Concurrent Task Execution




Spinnaker: Netflix's New Cloud Deployment Automation Tool



**Free DZone Refcard
Getting To Know Google Compute Engine And How To Use It**

Topics: CLOUD, AWS EC2, JENKINS, BITBUCKET, CONTINUOUS DELIVERY

Published at DZone with permission of Ajitesh Kumar, DZone MVB. [See the original article here.](#) 
Opinions expressed by DZone contributors are their own.
