

Continuous Delivery of embedded firmware using Docker and Jenkins

#TorinoTech Night, 2016-04-01

Gianpaolo Macario

<https://gmacario.github.io/>

(C) 2016 Gianpaolo Macario - License: [CC BY-SA 4.0](#)

The context

Firmware images of recent embedded devices are produced through a complex and time-consuming process

A few examples:

- In-Vehicle Infotainment
- Mobile Phone
- Home Gateway
- Wireless Router/Access Point
- etc

Example: GENIVI Demo Platform

```
$ slccount gdp_ivi9_beta/workspace
```

Totals grouped by language (dominant language first):

ansic:	46260693 (69.83%)
cpp:	9910207 (14.96%)
asm:	2006557 (3.03%)
sh:	1575972 (2.38%)
perl:	1276566 (1.93%)
python:	1163866 (1.76%)
xml:	1054193 (1.59%)

...

Total Physical Source Lines of Code (SLOC) = 66,247,653

Example: Android 5.1.1

```
$ slccount build_android_udooneo/workspace
```

Totals grouped by language (dominant language first):

ansic:	28909496	(48.66%)
cpp:	14820059	(24.95%)
xml:	6894345	(11.61%)
java:	5651249	(9.51%)
asm:	1105027	(1.86%)
python:	1061272	(1.79%)
sh:	434135	(0.73%)
perl:	197439	(0.33%)

...

Total Physical Source Lines of Code (SLOC) = 59,405,160

Build Host Configuration

Varies greatly between projects

Some examples:

- Android 1.5-2.2.x: Ubuntu 10.04, Oracle JDK 5
- Android 2.3.x-4.4.x: Ubuntu 12.04, Oracle JDK 6
- Android 5.x: Ubuntu 12.04, Oracle JDK 7
- Android 6.0: Ubuntu 14.04, Oracle JDK 7
- Android N: Ubuntu 14.04, OpenJDK 8
- Commercial IVI product: Ubuntu 10.04, CodeSourcery GCC 2014.05
- CommonAPI C: Fedora 23, systemd, libsystem-devel
- GENIVI Demo Platform: Ubuntu 14.04, gcc 4.8, make 4.22, python 2.7

Build host configuration (cont.)

- **Depends upon each embedded project**
 - Each developer may need to work on several projects with incompatible host requirements
 - The configuration of the host may affect the generated firmware image (host cross contamination)
 - Developers do not always follow instructions...
- **Should be maintained along the project lifetime**
 - We cannot assume that developers will not change the software configuration on their machines for years!

How can we ensure that each build is reproducible?

One solution: Virtual Machines

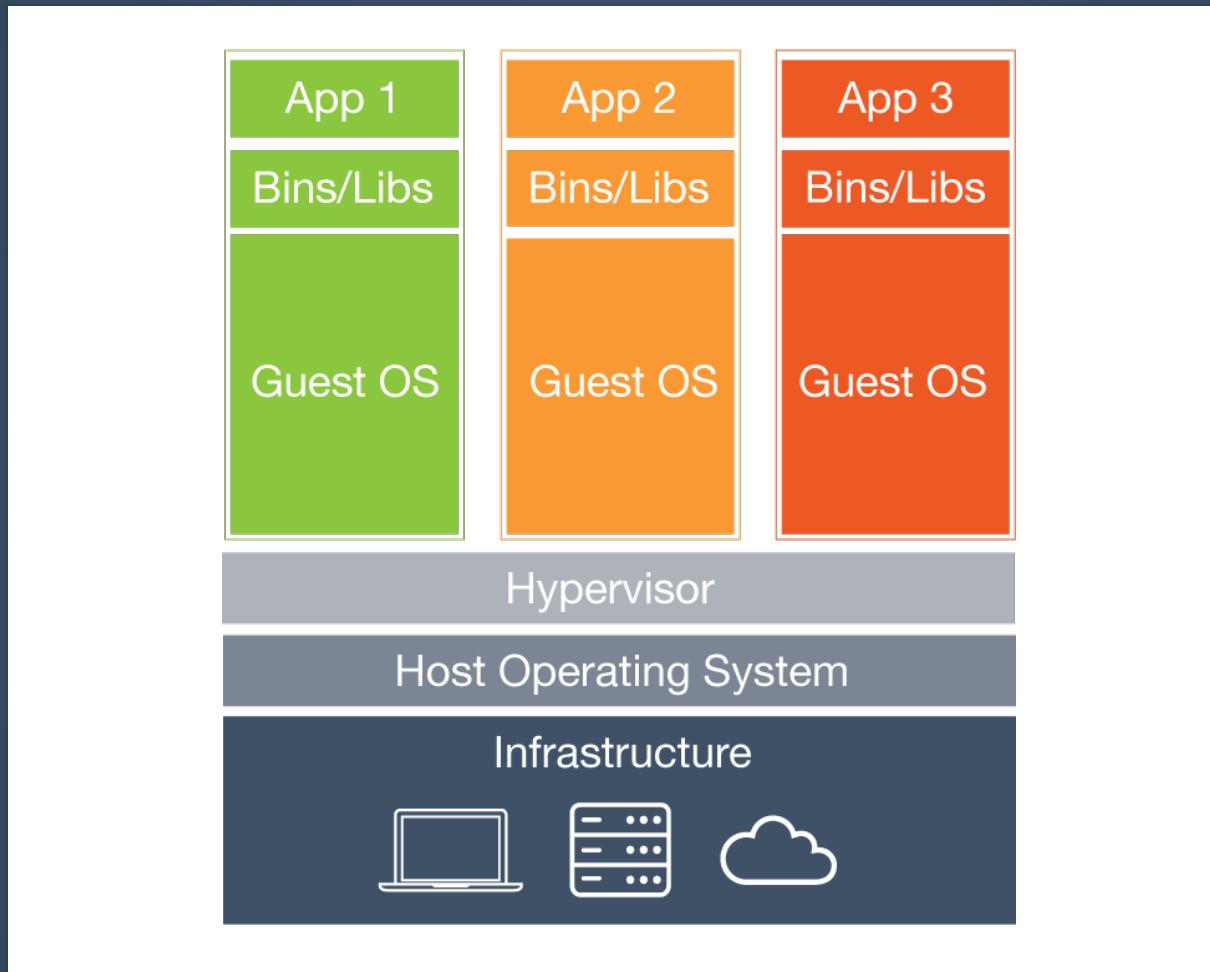


Image credit: <https://www.docker.com>

A better solution: Linux Containers

Containers vs. VMs

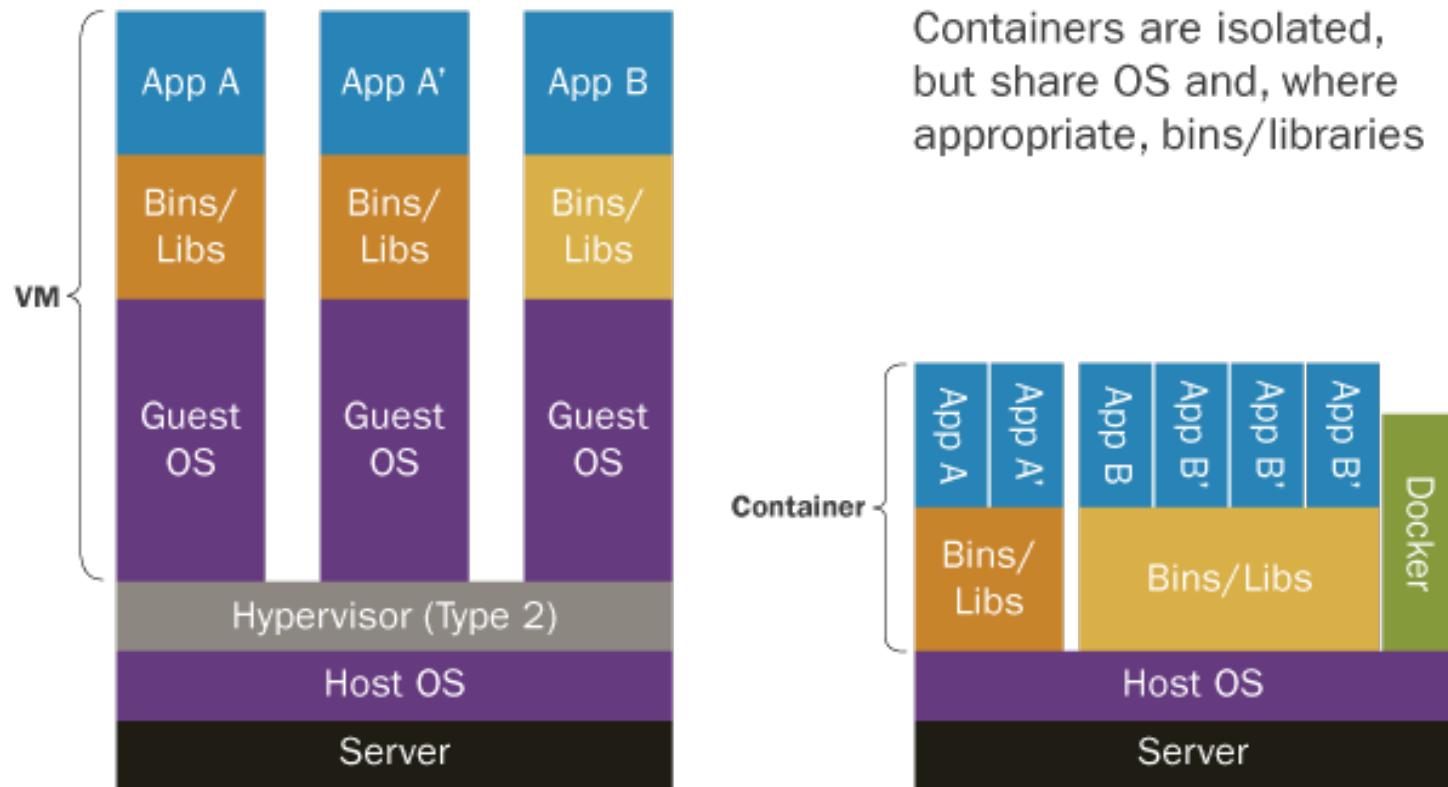


Image credit: <https://insights.sei.cmu.edu/devops/2015/01/devops-and-docker.html>

easy-build

The screenshot shows a web browser window displaying the GitHub README.md page for the `easy-build` repository. The title bar indicates the URL is `https://github.com/gmacario/easy-build`. The page content includes:

- ## easy-build
- Build status indicators: ImageLayers.io (0 B / 22 Layers), gitter (join chat), PullReview (0 X 0 ✓), Ready (0).
- A descriptive text block: "This repository contains a collection of Dockerfiles that help rebuilding a few embedded software distributions."
- A table listing subprojects and their descriptions:

Subproject	Description
build-aosp	Android Open Source Project
build-openwrt	OpenWrt
build-yocto	Yocto Project
build-yocto-fsl-arm	Yocto Project for Freescale/ARM targets
build-yocto-genivi	Yocto GENIVI Baseline, GENIVI Demo Platform

- A note: "Please refer to the `README.md` file available under each subdirectory for details and usage examples."
- ## System Requirements

 - Docker 1.9 or later (tested with Ubuntu and CoreOS)
 - A fast Internet connection

The build is just one step

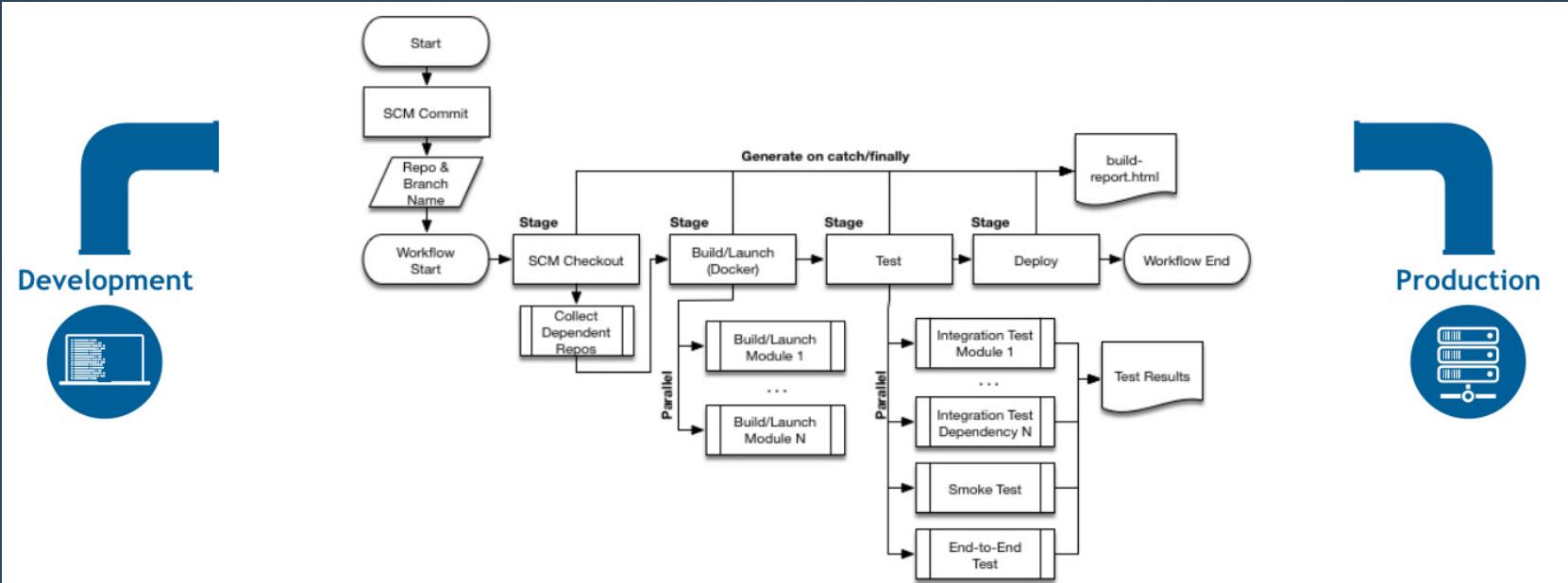


Image credit: <https://jenkins.io/doc/pipeline/>

Introducing Jenkins



A screenshot of a web browser displaying the Jenkins homepage at <https://jenkins.io>. The page features a large blue header with the Jenkins logo and navigation links for Downloads, Participate, Use-cases, Blog, Documentation, Wiki, Issues, Press, and Contribute. A red banner on the right says "Fork me on GitHub". The main content area has a teal background. On the left is a cartoon illustration of a man in a tuxedo holding a coffee cup. To his right, the word "Jenkins" is written in large white letters, followed by the tagline "Build great things at any scale". Below that is a description: "The leading open source automation server, Jenkins provides hundreds of plugins to support building, deploying and automating any project." A large red button with white text says "Download Jenkins". At the bottom, there's a link to "Get 1.642.3 LTS .war or the latest 1.654 weekly release".

easy-jenkins

The screenshot shows a web browser window with the GitHub URL <https://github.com/gmacario/easy-jenkins>. The page displays the contents of the README.md file. The title "easy-jenkins" is prominently displayed, followed by a description: "Easily deploy a Jenkins CI/CD infrastructure via docker-machine and docker-compose." It also links to the CHANGELOG. A "TL;DR" section provides a command to clone and run the repository:

```
$ git clone https://github.com/gmacario/easy-jenkins  
$ cd easy-jenkins  
$ ./runme.sh
```

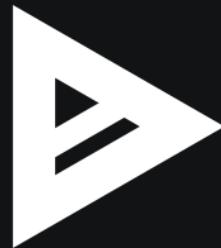
If the script executes successfully, it will display a message like the following:

```
INFO: Now browse http://192.168.99.100:9080/ to access the Jenkins dashboard
```

The Jenkins dashboard may then be accessed by opening the displayed URL using a recent Internet browser.

Running easy-jenkins

```
root@ies-genbld01-vm:~# # This session explains how to get started with easy-jen  
kins  
root@ies-genbld01-vm:~#
```



easy-jenkins in action

The screenshot shows the Jenkins web interface with the title "Build History of Jenkins". The left sidebar includes links for New Item, People, Build History (which is selected), Project Relationship, Check File Fingerprint, Manage Jenkins, Credentials, Job Config History, and Scriptler. Below these are sections for Build Queue (empty) and Build Executor Status, which lists two builds: "build_android_udoneo #3" and "part of GENIVI > yocto-baseline-next #6". The main content area features a timeline from March 29 to April 2, with a 3hr to 7hr scale below it. A table titled "Export as plain XML" lists ten build entries with columns for Build, Time Since, and Status. Most builds are green (stable), except for "build_android_udoneo #2" which is red (broken for a long time) and "build_android_udoneo #1" which is red (broken since this build). At the bottom, there are icons for S, M, L, and links for RSS feeds.

Build	Time Since ↑	Status
GENIVI > yocto-baseline-next #6	19 sec	?
build_android_udoneo #3	1 min 18 sec	?
build_android_udoneo #2	1 hr 50 min	broken for a long time
GENIVI > yocto-baseline-next #5	12 hr	stable
build_android_udoneo #1	13 hr	broken since this build
build.automotivelinux.org > 00-TEST-SNAPSHOT-AGL-master #1	22 hr	stable
seed-agl #1	22 hr	stable
GENIVI > build_gdp_ivi9_beta #1	1 day 19 hr	stable
GENIVI > common-api-c #1	1 day 19 hr	stable
test_docker #1	1 day 19 hr	stable

Icon: S M L Legend: [RSS for all](#) [RSS for failures](#) [RSS for just latest builds](#)

[Help us localize this page](#) Page generated: Apr 1, 2016 9:53:49 AM [REST API](#) [Jenkins ver 1.642.2](#)

Summary

- Working on an embedded project requires
 - Setting up a customized build environment
 - Maintaining it during the whole project lifetime
- Create a reproducible build environment using Docker
 - Develop your own image, or start from `easy-build`
- Jenkins can automate the complete delivery pipeline
- Use `easy-jenkins` together with `easy-build`
 - Setup your CD infrastructure in minutes, not days
 - Upgrade your infrastructure config with a `git commit`

Thanks!



Gianpaolo Macario
<https://gmacario.github.io/>