

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

```
$ sloccount 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

```
$ sloccount 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

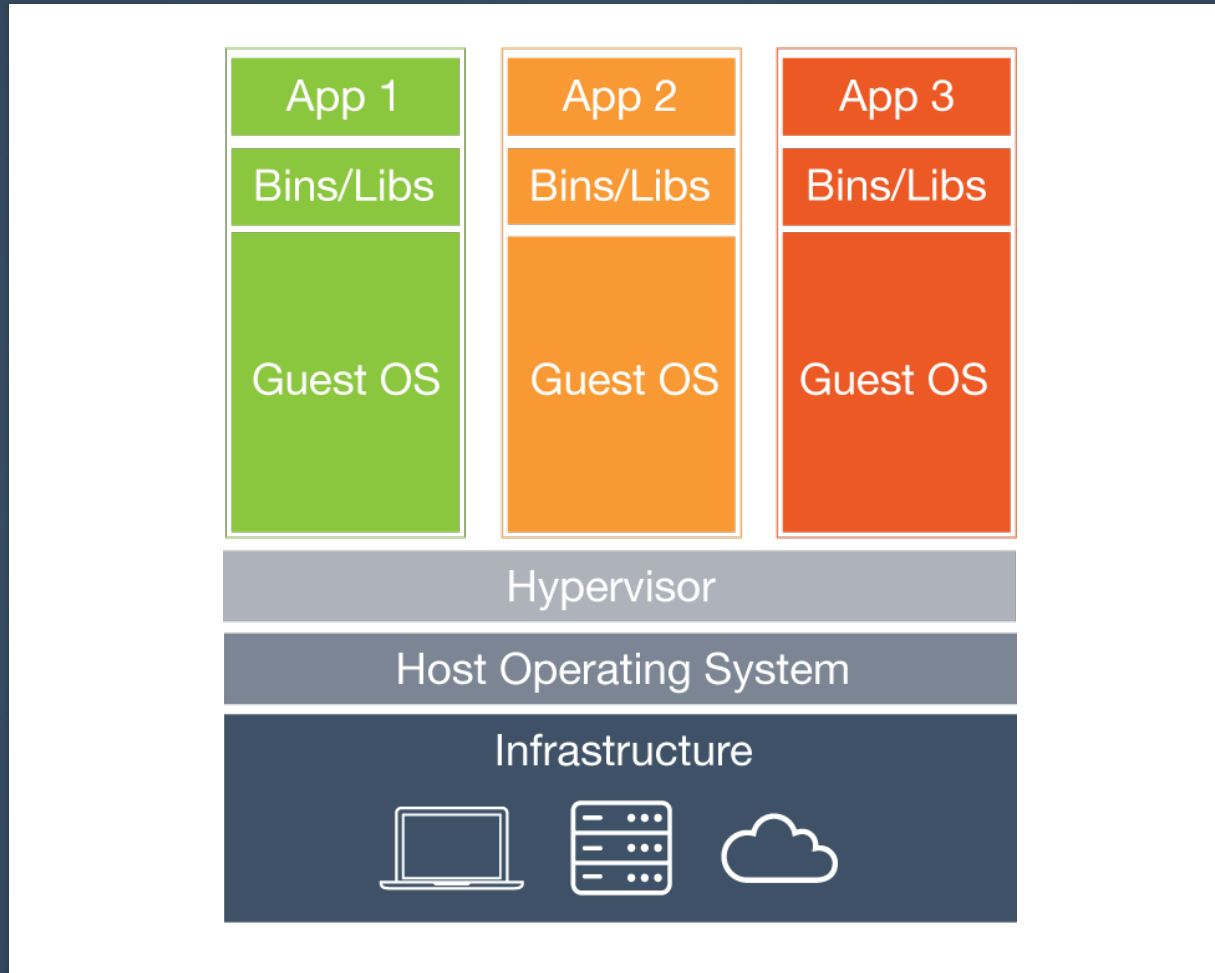
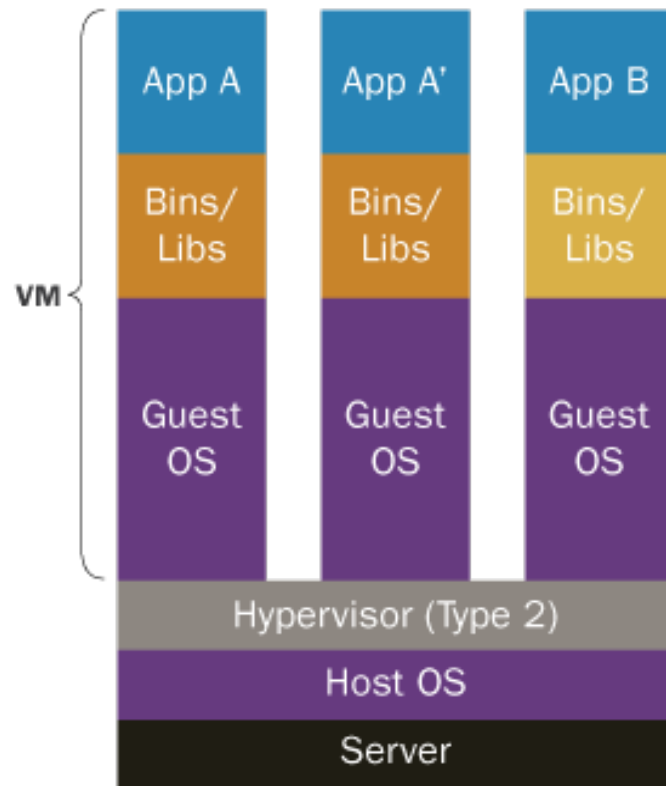


Image credit: <https://www.docker.com>

A better solution: Linux Containers

Containers vs. VMs



Containers are isolated, but share OS and, where appropriate, bins/libraries

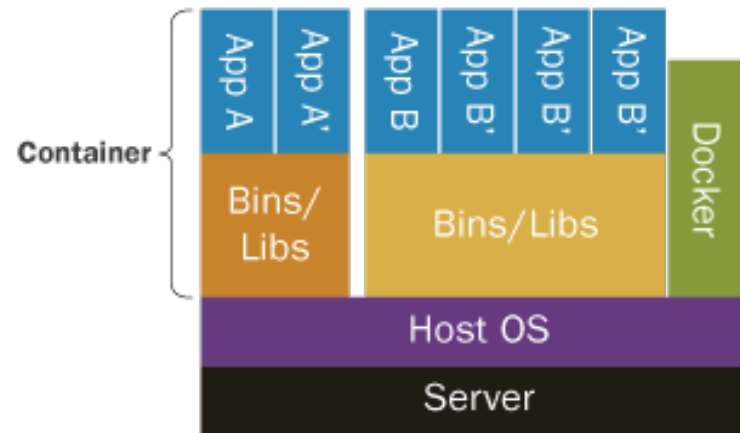
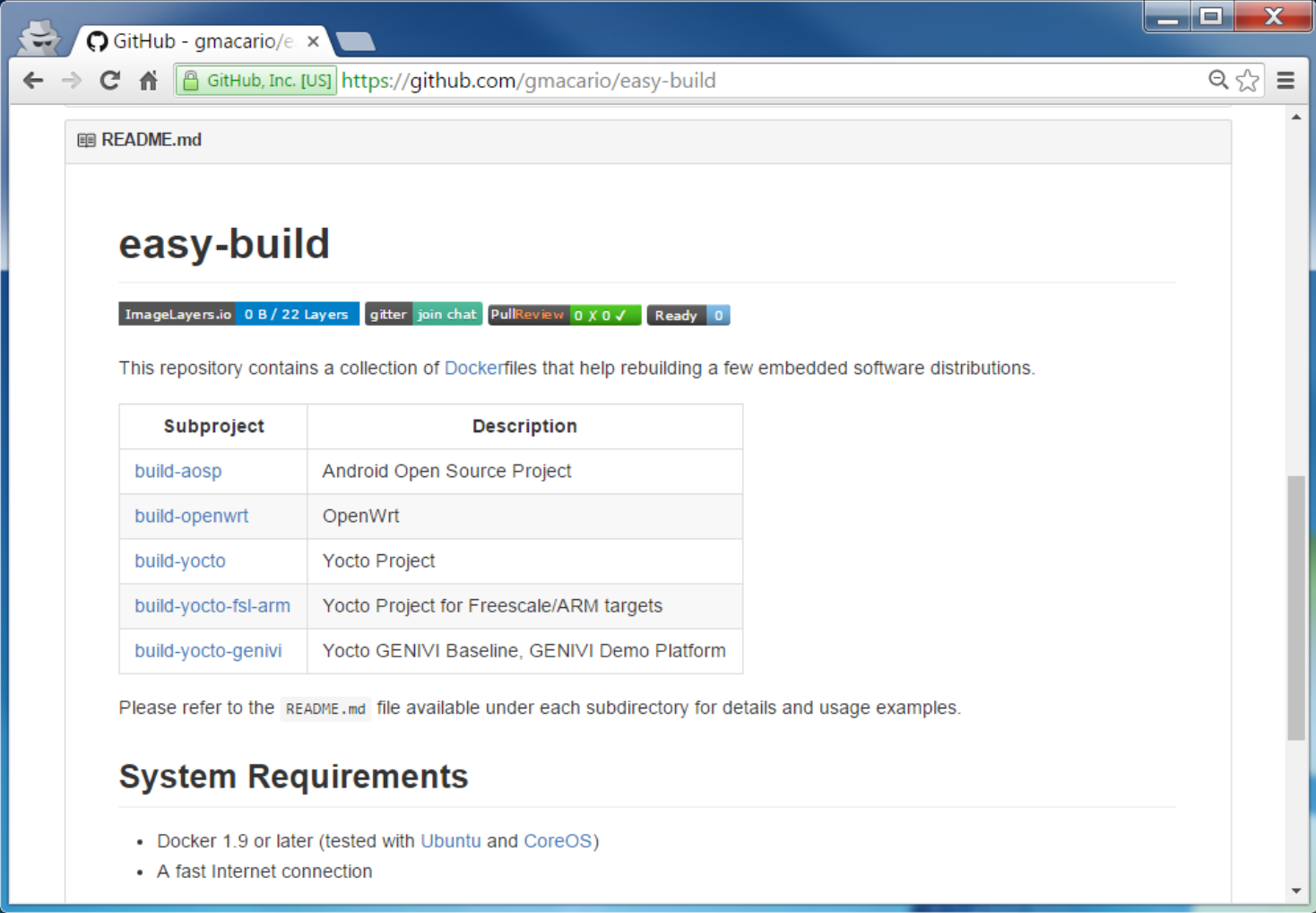


Image credit: <https://insights.sei.cmu.edu/devops/2015/01/devops-and-docker.html>

easy-build



The screenshot shows a web browser window displaying the GitHub repository page for 'easy-build' by user 'gmacario'. The browser's address bar shows the URL 'https://github.com/gmacario/easy-build'. The repository page features a 'README.md' tab, a title 'easy-build', and a statistics bar indicating 0 B / 22 Layers, 0 gitter, 0 join chat, 0 PullReview, 0 X 0 ✓, and 0 Ready. A description states: 'This repository contains a collection of Dockerfiles that help rebuilding a few embedded software distributions.' Below this is a table with two columns: 'Subproject' and 'Description'. The table lists six subprojects: 'build-aosp' (Android Open Source Project), 'build-openwrt' (OpenWrt), 'build-yocto' (Yocto Project), 'build-yocto-fsl-arm' (Yocto Project for Freescale/ARM targets), and 'build-yocto-genivi' (Yocto GENIVI Baseline, GENIVI Demo Platform). A note below the table says: 'Please refer to the README.md file available under each subdirectory for details and usage examples.' The 'System Requirements' section lists two items: 'Docker 1.9 or later (tested with Ubuntu and CoreOS)' and 'A fast Internet connection'.

GitHub - gmacario/easy-build

← → ↺ ⬆ 📁 GitHub, Inc. [US] https://github.com/gmacario/easy-build 🔍 ☆ ☰

📖 README.md

easy-build

ImageLayers.io 0 B / 22 Layers gitter join chat PullReview 0 X 0 ✓ Ready 0

This repository contains a collection of [Dockerfiles](#) that help rebuilding a few embedded software distributions.

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

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

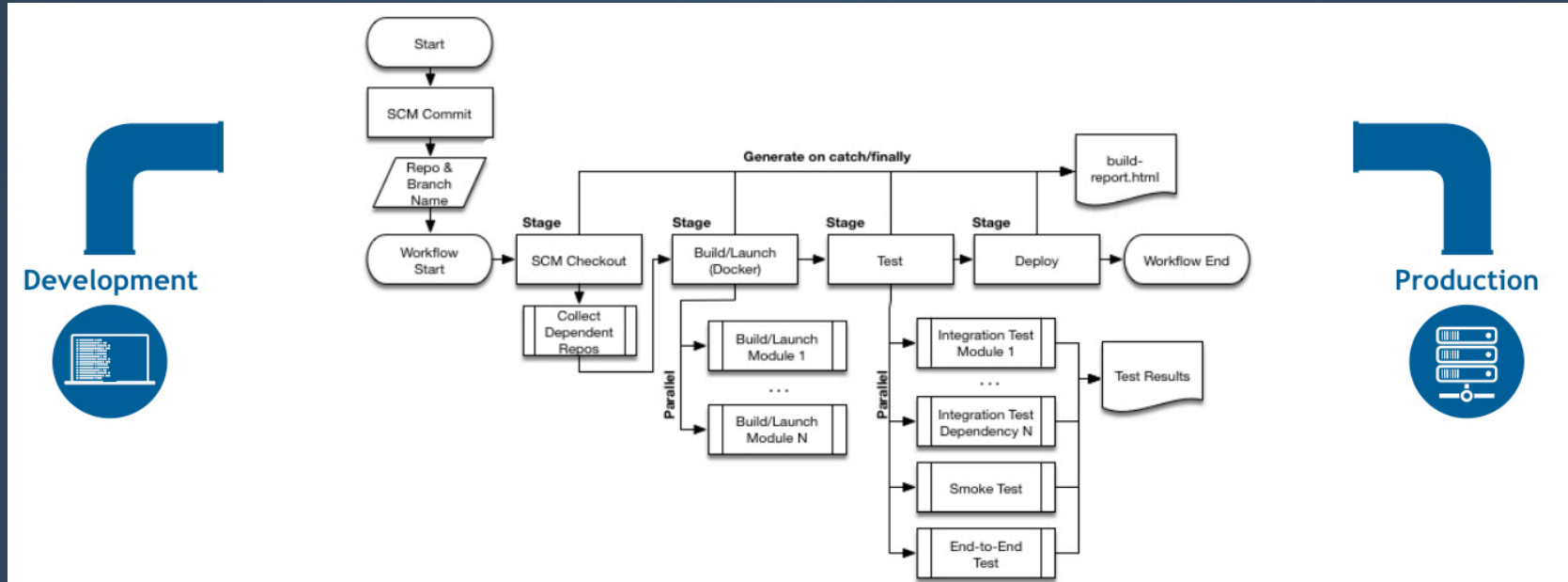
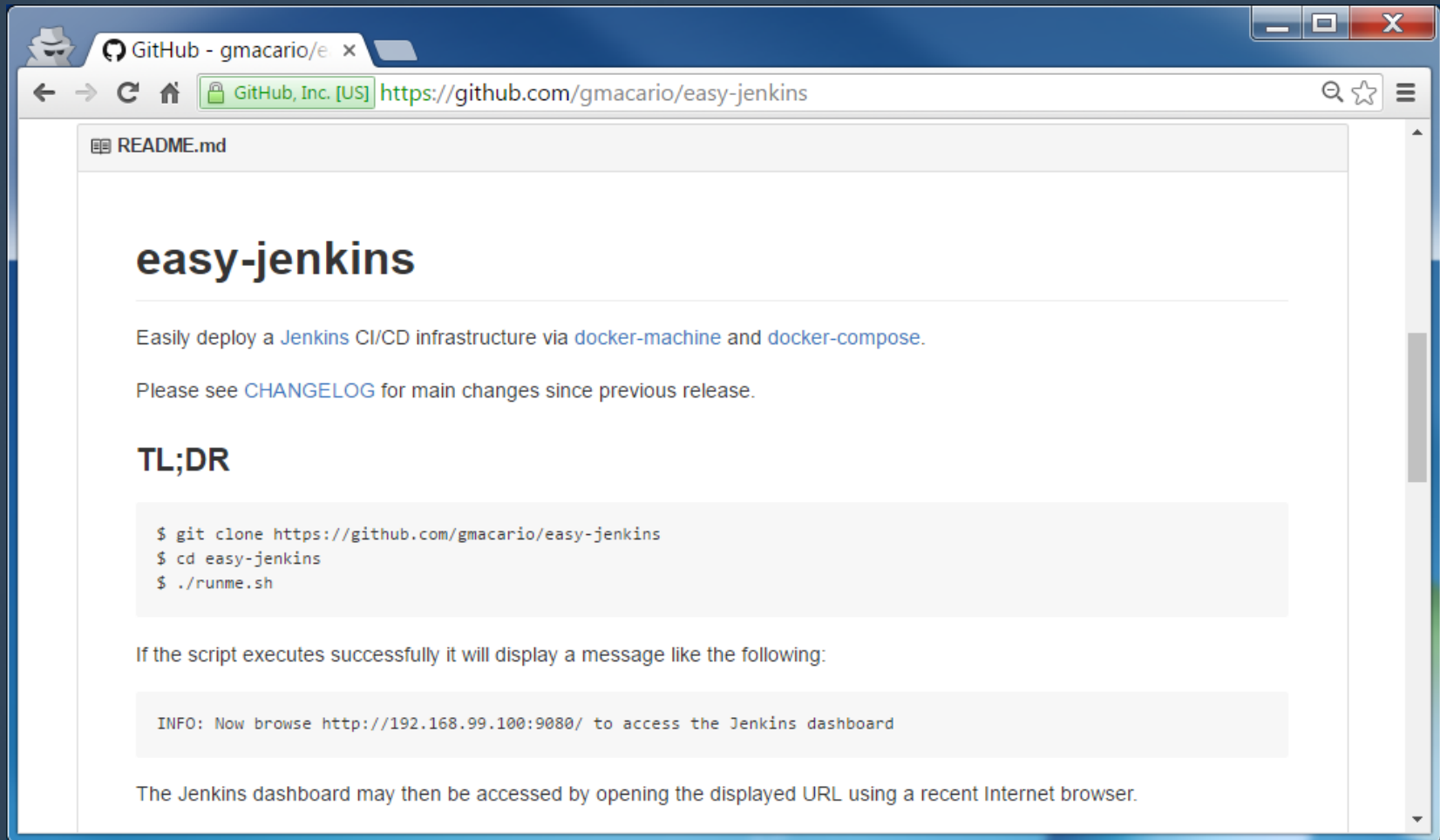


Image credit: <https://jenkins.io/doc/pipeline/>

Introducing Jenkins



easy-jenkins



Running easy-jenkins

```
gmacario@alm-gm-ubu15:~$ # Getting started with easy-jenkins
```



easy-jenkins in action

The screenshot shows the Jenkins web interface. The browser address bar indicates the URL: `alm-gm-ubu15.solarma.it:9080/view/All/builds`. The Jenkins logo and name are at the top left. A sidebar on the left contains navigation links: New Item, People, Build History (selected), Project Relationship, Check File Fingerprint, Manage Jenkins, Credentials, Job Config History, and Scriptler. Below the sidebar, the 'Build Queue' section shows 'No builds in the queue.' The 'Build Executor Status' section shows two executors: 'build_android_udoooneo #3' and 'part of GENIVI » yocto-baseline-next #6'. The main content area is titled 'Build History of Jenkins' and features a calendar view showing builds from March 29 to April 2. Below the calendar is a table of build history.

Build	Time Since	Status
GENIVI » yocto-baseline-next #6	19 sec	?
build_android_udoooneo #3	1 min 18 sec	?
build_android_udoooneo #2	1 hr 50 min	broken for a long time
GENIVI » yocto-baseline-next #5	12 hr	stable
build_android_udoooneo #1	13 hr	broken since this build
build_automotivelinux.org » 00-TEST-SNAPSHOT-AGL-master #1	22 hr	stable
seed-agi #1	22 hr	stable
GENIVI » build_gdp_iv9_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

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/>