We feel the developer’s pain
Frustration with build environments

- Versions
- Packages
- Conflicts
- Updates
- Compilers
- Toolchains
Frustration with build environments

But it works on my machine!!!
What would we like to have?

- Build for multiple platforms on the same machine
- Have immutable build environments
- Use exactly the same environment in CI and on local machine
What would we like to have?

- Share between team members
- Enable unified experience
- Track environment’s versions
What would we like to have?

- Think about the future
- Being able to reproduce build environment in several years
- Certification audits
Solution

The solution leverages container technology, providing seamless usage of multiple environments. It enables easy creating and sharing of environments, stopping the phrase “but it works on my machine.”
Underlying magic

- Leverages Linux containers
- Enables correct file permission and ownership
- Works on any modern OS
https://github.com/rbld/rebuild
Architecture

- Rebuild CLI
- Docker Engine
- Environments
  - registry
  - Docker private registry
  - Rebuild native registry
  - DockerHub
[rbld@rebuild ~]$ rbld help

Usage:
rbld help Show this help screen
rbld help COMMAND Show help for COMMAND
rbld COMMAND [PARAMS] Run COMMAND with PARAMS

rebuild: Zero-dependency, reproducible build environments

Commands:
checkout
commit
create
deploy
list
load
modify
publish
rm
run
save
search
status
version
CLI Concepts

- Seamless usage for developer
- Knowledge of Docker, Docker files or other container technologies not required
CLI Concepts: Environment names and versioning

name:tag

- Name - environment ID
- Tag - should be used for tracking the version of the environment
Registry: DockerHub

- Images saved to public cloud
- Images can be shared with everybody
- Images from official repositories can be used as base images for new environments
Registry: Docker private registry

- Docker enterprise or community addition of private registry
- Internal to your organization
- Might need additional configurations
Registry: Rebuild Native registry

- Easy to deploy and use in production
- Lightweight
- Internal for your organization
- No configuration is needed
Supported OSes

- Windows
- OS X
Installation
Installation - dependencies

- Docker
- Ruby 2.0+
Installation

[rbl@rebuild ~]$ gem install rbld
Fetching: rbld-1.3.7.gem (100%)
Building native extensions. This could take a while...
Thanks for installing rebuild. Run `rbld help` for CLI reference.
Successfully installed rbld-1.3.7
Parsing documentation for rbld-1.3.7
Installing ri documentation for rbld-1.3.7
Done installing documentation for rbld after 2 seconds
1 gem installed
Quick start

- Run: rbld help
- Already configured to deploy environments from Rebuild DockerHub repositories
- Can create environments based on DockerHub
Get Help

[rbld@rebuild ~]$ rbld help

Usage:
rbld help				Show this help screen
rbld help COMMAND				Show help for COMMAND
rbld COMMAND [PARAMS]		Run COMMAND with PARAMS

rebuild: Zero-dependency, reproducible build environments

Commands:

checkout
commit
create
deploy
list
load
modify
publish
rm
run
save
search
status
version
Configuration: DockerHub

- $HOME/.rbld/rebuild.conf

#DockerHub
REMOTE_NAME=origin
REMOTE_TYPE_origin="dockerhub"
REMOTE_origin="<NAMESPACE>/<REPOSITORY>"
Configuration: Rebuild registry

- $HOME/.rbld/rebuild.conf

#Rebuild registry
REMOTE_NAME=origin
REMOTE_TYPE_origin="rebuild"
REMOTE_origin="<ABSOLUTE PATH TO REGISTRY ROOT DIRECTORY>"
Configuration: Docker Registry

- $HOME/.rbld/rebuild.conf

#Docker registry
REMOTE_NAME=origin
REMOTE_TYPE_origin="docker"
REMOTE_origin="<REGISTRY IP>:<PORT>"
Workflow: basic usage

Search: rbld search
Deploy: rbld deploy
Run: rbld run
Workflow: basic usage

Search  rbld search

Deploy  rbld deploy

Deploy environment from remote registry to local machine

- Only needs to be done once
- Environment can be instantly used after deployment
Workflow: basic usage

[test@rebuild-fedora23-1 ~]$ rbld search
Searching in /Users/yanvugenfirer/dev/rebuild/repository...
  rpi-raspbian:v001

[test@rebuild-fedora23-1 ~]$ rbld deploy rpi-raspbian:v001
Working: |===================================================|
Successfully deployed rpi-raspbian:v001

[test@rebuild-fedora23-1 ~]$ rbld run rpi-raspbian:v001 -- make
>>> rebuild env rpi-raspbian-v001
>>> make
/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/bin/arm-linux-gnueabihf-gcc hello.c -o hello -DENV_NAME="rpi-raspbian-v001"
<<< rebuild env rpi-raspbian-v001
Usage - interactive

[yanvugenfirer@rebuild ~]$ rbld run rpi-raspbian:v001

>>> rebuild env rpi-raspbian-v001 interactive
>>> Press CTRL-D do leave
yanvugenfirer@rpi-raspbian-v001:~/dev/Rebuild/rebuild/cli/tests/hello$ ls
Makefile  hello.c
yanvugenfirer@rpi-raspbian-v001:~/dev/Rebuild/rebuild/cli/tests/hello$ cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=16.04
DISTRIB_CODENAME=xenial
DISTRIB_DESCRIPTION="Ubuntu 16.04.1 LTS"
yanvugenfirer@rpi-raspbian-v001:~/dev/Rebuild/rebuild/cli/tests/hello$ exit

<<< rebuild env rpi-raspbian-v001
Workflow: basic usage

Running the environment

- Environment doesn’t change in the process of running
- Can be used in interactive mode or to run just one command
- Changes to local files are preserved with correct permissions and ownership
Workflow: environment modification

Modify: rbld modify
Commit: rbld commit
Publish: rbld publish
Workflow: environment modification

- Local modification only

Modify: rbld modify
Commit: rbld commit
Workflow: environment modification

[test@rebuild-fedora23-1 ~]$ rbld modify ubuntu1510 -- \
  "sudo apt-get update && \
  sudo apt-get install -y gcc"

Initializing environment [.....]
>>> rebuild env ubuntu1510:initial-M
>>> sudo apt-get update && sudo apt-get install -y gcc
Hit http://archive.ubuntu.com wily InRelease

...

Processing triggers for libc-bin (2.21-0ubuntu4.3) ...
<<< rebuild env ubuntu1510:initial-M
Modify - interactive mode

[test@rebuild-fedora ~]$ rbld modify
ubuntu1510:initial

Initializing environment [.....]
>>> rebuild env ubuntu1510:initial-M interactive
>>> Press CTRL-D do leave
test@ubuntu1510:initial-M:~$
Get status

[test@rebuild-fedora23-1 ~]$ rbld status
modified: ubuntu1510:initial
Commit changes to environment

[test@rebuild-fedora ~]$ rbld commit ubuntu1510 -tag v001
Workflow: environment modification - reverting the changes

Modify: rbld modify
Revert: rbld checkout
Revert changes

[test@rebuild-fedora23-1 ~]$ rbld checkout ubuntu1510:initial
Workflows - environment creation

Create: `rbld create`
Modify: `rbld modify`
Commit: `rbld commit`
Publish: `rbld publish`
Workflows - environment creation

[test@rebuild-fedora23-1 ~]$ rbld create --base ubuntu:15.10 ubuntu1510

Downloading the base image...
Trying to pull repository docker.io/library/ubuntu ... 15.10: Pulling from library/ubuntu

...  

Successfully created ubuntu1510:initial
Workflows - environment creation

[test@rebuild-fedora23-1 ~]$ rbld create --basefile ubuntu1510.tar.gz rpi-raspbian

Building environment...

... 

Successfully created rpi-raspbian:initial
Workflows - environment creation

[test@rebuild-fedora23-1 ~]$ rblt publish qemu-fc22:v001

Checking for collisions...

Publishing at 10.0.110.110:5000...

... 

Successfully published qemu-fc22:v001
putting it all together

```
# Clone the rpi-tools repository
git clone git://github.com/raspberrypi/tools.git rpi-tools

# Create a new base for rpi-raspbian
rbld create --base ubuntu:16.04 rpi-raspbian

# Modify the initial configuration for rpi-raspbian
rbld modify rpi-raspbian:initial

# Update the package list
>> sudo apt-get update

# Install make
>> sudo apt-get install -y make

# Set the toolchain
>> TOOLCHAIN=gcc-linaro-arm-linux-gnueabihf-raspbian-x64

# Copy the toolchain
>> sudo cp -r rpi-tools/arm-bcm2708/$TOOLCHAIN /

# Append the export command to rebuild/rebuild.rc
>> echo export CC=/$TOOLCHAIN/bin/arm-linux-gnueabihf- | 
    sudo tee -a /rebuild/rebuild.rc
```
Workflows - local environments management

- List: `rbld list`
- Delete: `rbld rm`
- Save: `rbld save`
- Load: `rbld load`
Workflows - local environments management

[test@rebuild-fedora23-1 ~]$ rbld list

bb-x15:16-05
nrf5:13
qemu-fc20:v001
qemu-fc22:initial
qemu-fc22:v001
qemu-fc23:v001
rpi-raspbian:initial
rpi-raspbian:v001
ubuntu1604:v001
Sharing environments - saving and loading from a file

[test@rebuild-fedora23-1 ~]$ rbld save qemu-fc20:v001

Successfully saved environment qemu-fc20:v001 to qemu-fc20-v001.rbld

[test@rebuild-fedora23-1 ~]$ rbld load qemu-fc20-v001.rbld

Successfully loaded environment from qemu-fc20-v001.rbld
Environment-wide variables

[test@rebuild-fedora23-1 ~]$ rbld modify ubuntu1510:initial

Initializing environment [.....]
>>> rebuild env ubuntu1510:initial-M interactive
>>> Press CTRL-D do leave

# This is the rebuild environment definition file.
# Define required environment variables here.
#
# NOTE: Make sure to prefix definitions with
# 'export' directive, i.e.
#export MY_PATH=/path/to/some/location
export CC=clang

test@ubuntu1510:initial-M:$ exit
<<< rebuild env ubuntu1510:initial-M
Debugging

- Rebuild CLI
  - RBLD_LOG_LEVEL - info, warn, error and fatal
  - RBLD_LOG_FILE - save logs to file
- Environment bootstrap tracing
  - RBLD_BOOTSTRAP_TRACE - set to 1 to enable environment startup tracing for rbld run and rbld modify.
Summary

- Seamless usage
- Isolated environments
- Environments can be easily shared between team members
Future plans

- Role management for Rebuild Native Registry
- Support for multiple registries
- Support for additional environment registries
- IDE extensions
- USB redirection for non-Linux hosts
rbld@rbld.io

https://github.com/rbld/rebuild