

# REBUILD

Simplifying Embedded and IoT Development  
Using Linux Containers

Yan Vugenfirer - [yan@daynix.com](mailto:yan@daynix.com)

Dmitry Fleytman, PhD - [dmitry@daynix.com](mailto:dmitry@daynix.com)

We feel developers' pain



# Build environments frustration



# Build environments frustration

It works on my machine!!!



# Solution

## REBUILD

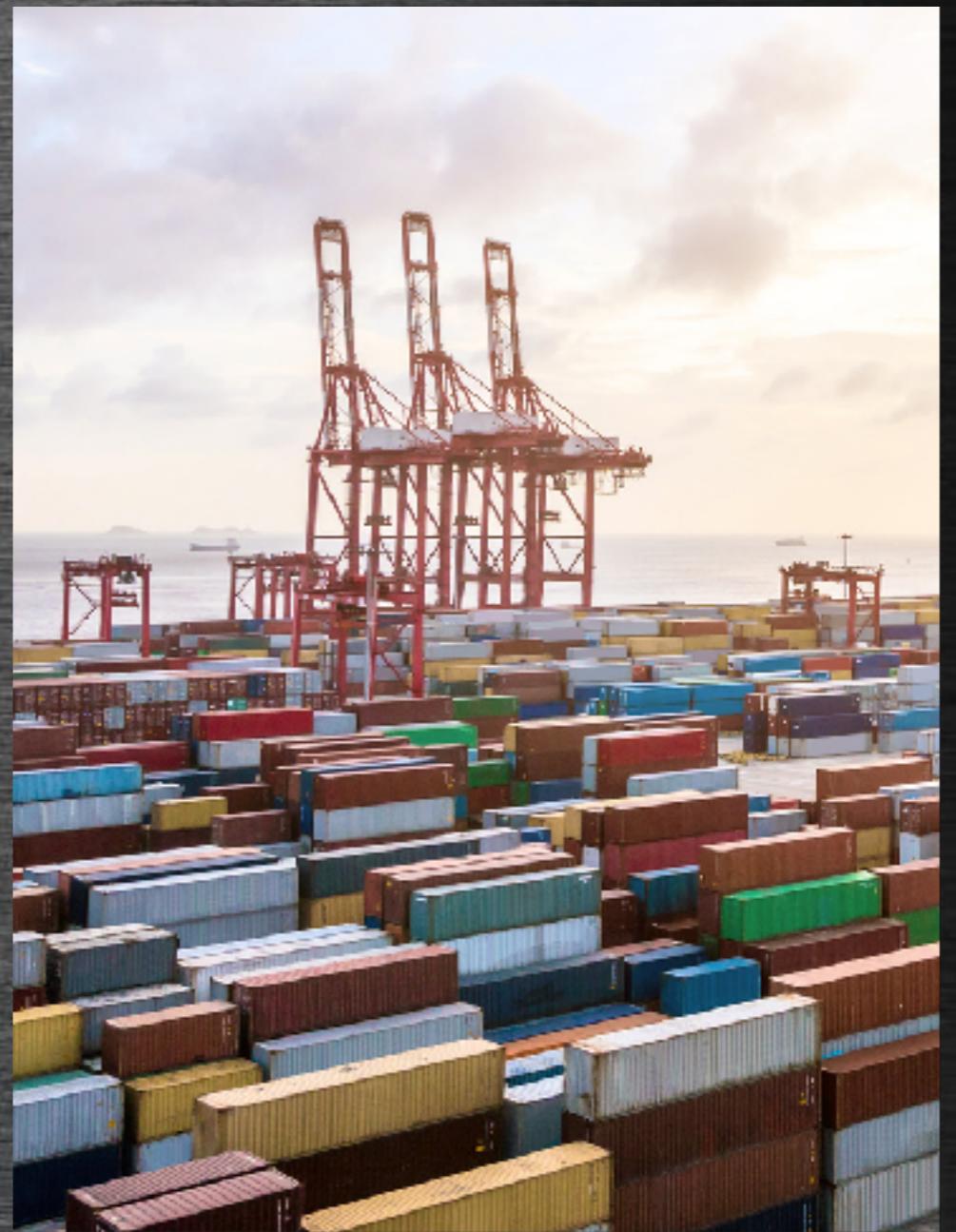
- Leverages containers technology
- Seamless usage of multiple environments
- Environments are easily created and shared
- No more “works on my machine”

DEMO

Am  
Sp

# Underlying magic

- Leverages Linux containers
- Enables correct file permission and ownership
- Works on any modern OS

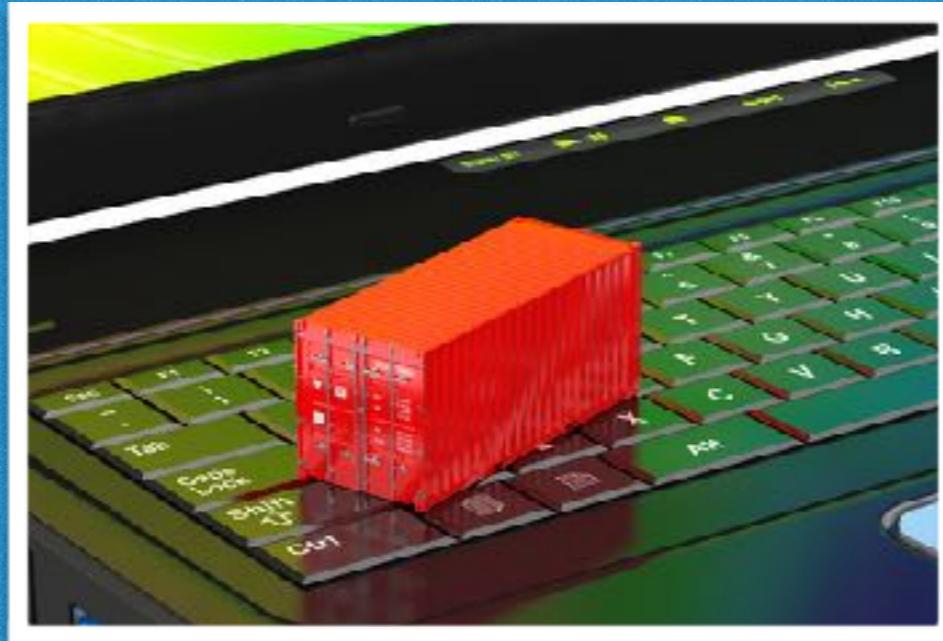


# What are containers?



Linux Containers - is an operating-system-level virtualization method for running multiple isolated Linux systems (containers) on a control host using a single Linux kernel.

# What is container's image?



A lightweight, stand-alone, executable package that includes everything needed to run a piece of software, including the code, a runtime, libraries, environment variables and config files.



<https://github.com/rblד/>  
rebuild

# Architecture

Rebuild CLI

Docker Engine

Environments  
registry

Docker private  
registry

Rebuild native  
registry

DockerHub

# Rebuild CLI

```
Last login: Sun Apr 23 12:14:20 on ttys000
Yans-MacBook-Pro-2:~ yanvugenfirer$ 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

Yans-MacBook-Pro-2:~ yanvugenfirer$
```

# CLI concepts

- Seamless usage for developer
- No knowledge of Docker, Docker files or other container technologies is needed

# CLI concepts: Manage local environments

- Manage environments on local machine
  - List - lists local environments
  - Remove (rm) - deletes local environment
  - Save\load - saves or loads to\from container image

# CLI concepts: Environment names and versioning

- name:tag
  - Name - environment ID
  - Tag - should be used for tracking the version of the environment

# CLI concepts: Deploy

- Deploy environment from remote registry to local machine
  - Search repository for environments
  - Need to be done once
  - Environment can be instantly used after deployment

# CLI concepts: Run

- Run 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

# CLI concepts: Create

- Create new environment
  - From base image from DockerHub
  - From archive of a file system

# CLI concepts: Modify, status and commit

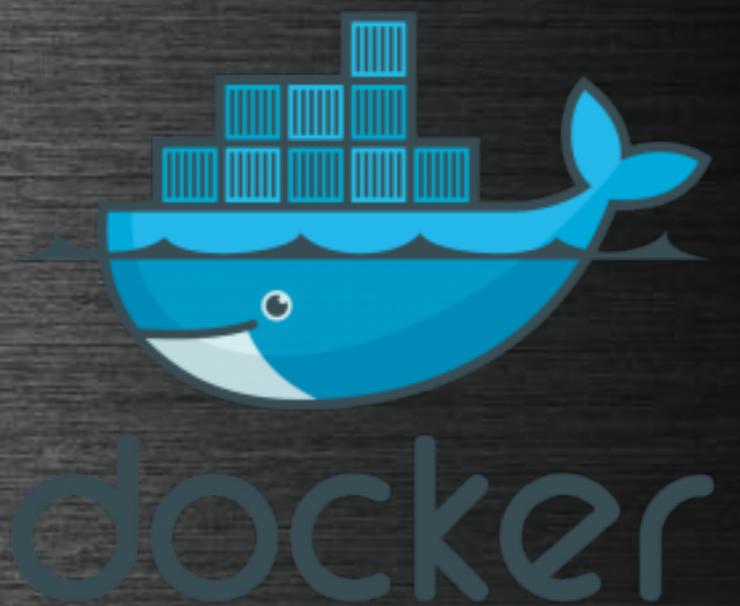
- Change existing environment
  - Modify - instructs rebuild to track changes in existing environment
  - Status - shows environments' status
  - Commit - saves the changes and create new tag for the environment

# CLI concepts: Publish

- Share your work
- “Pushes” the environment to repository

# 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 organisation
- Might need additional configurations



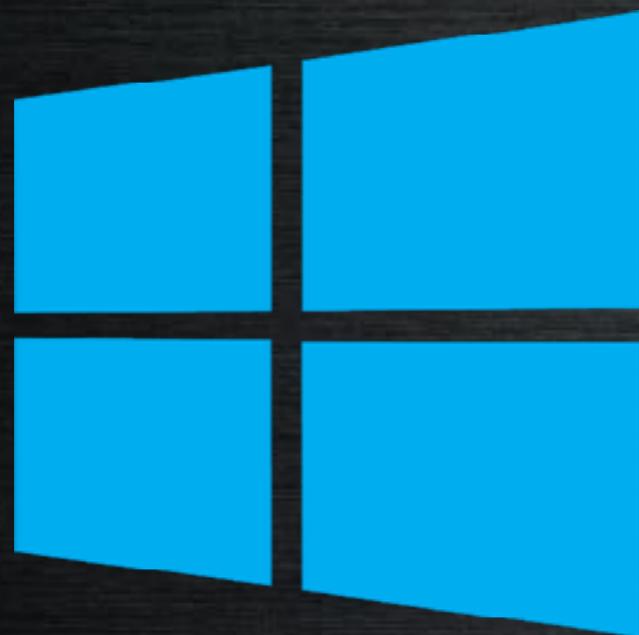
PRIVATE DOCKER CONTAINER IMAGES

# Registry: Rebuild Native registry

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



# Supported OSes



OSX

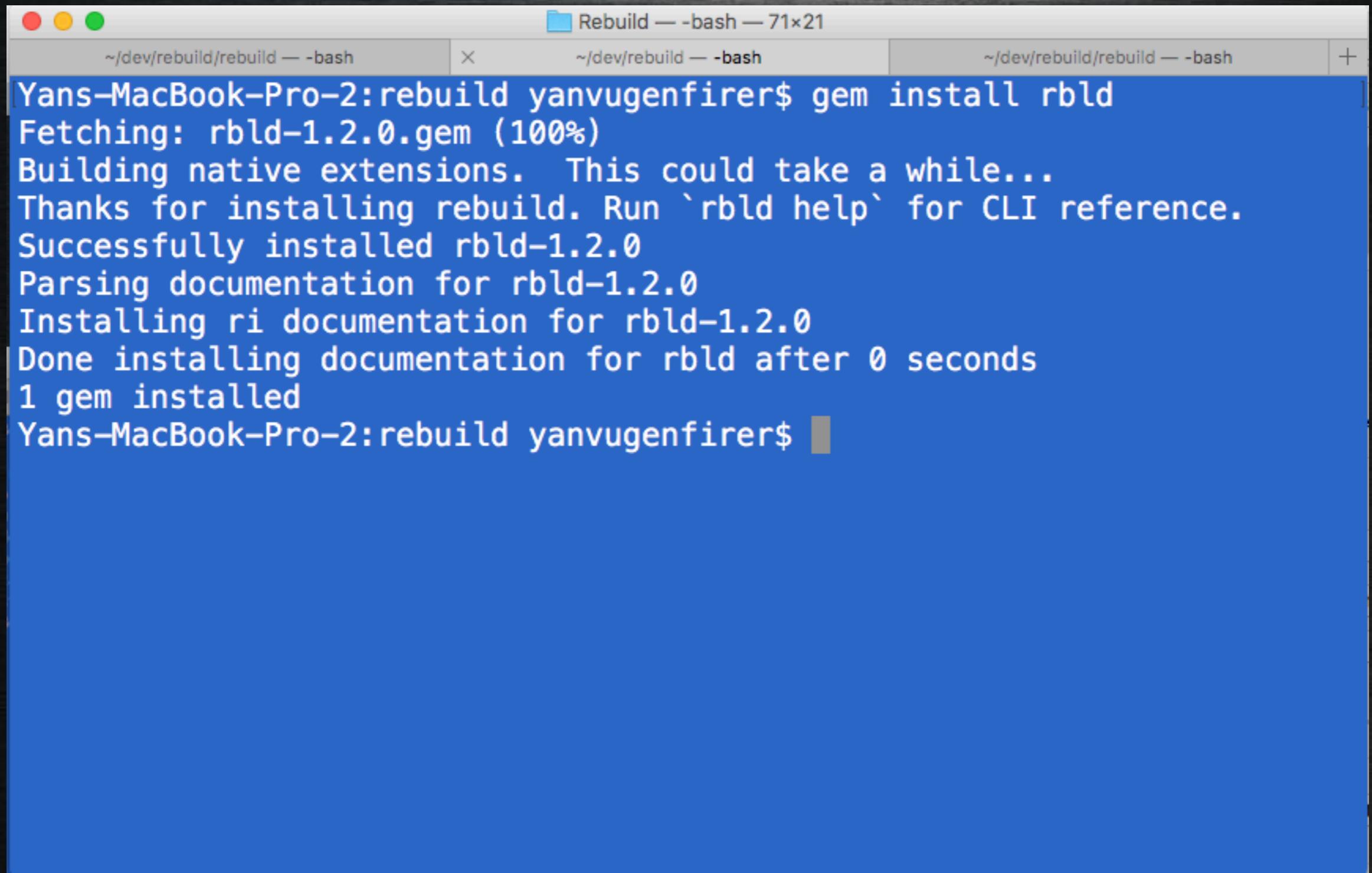
# Installation



# Installation - dependencies

- ❖ Docker engine
- ❖ Ruby 2.0+

# Installation



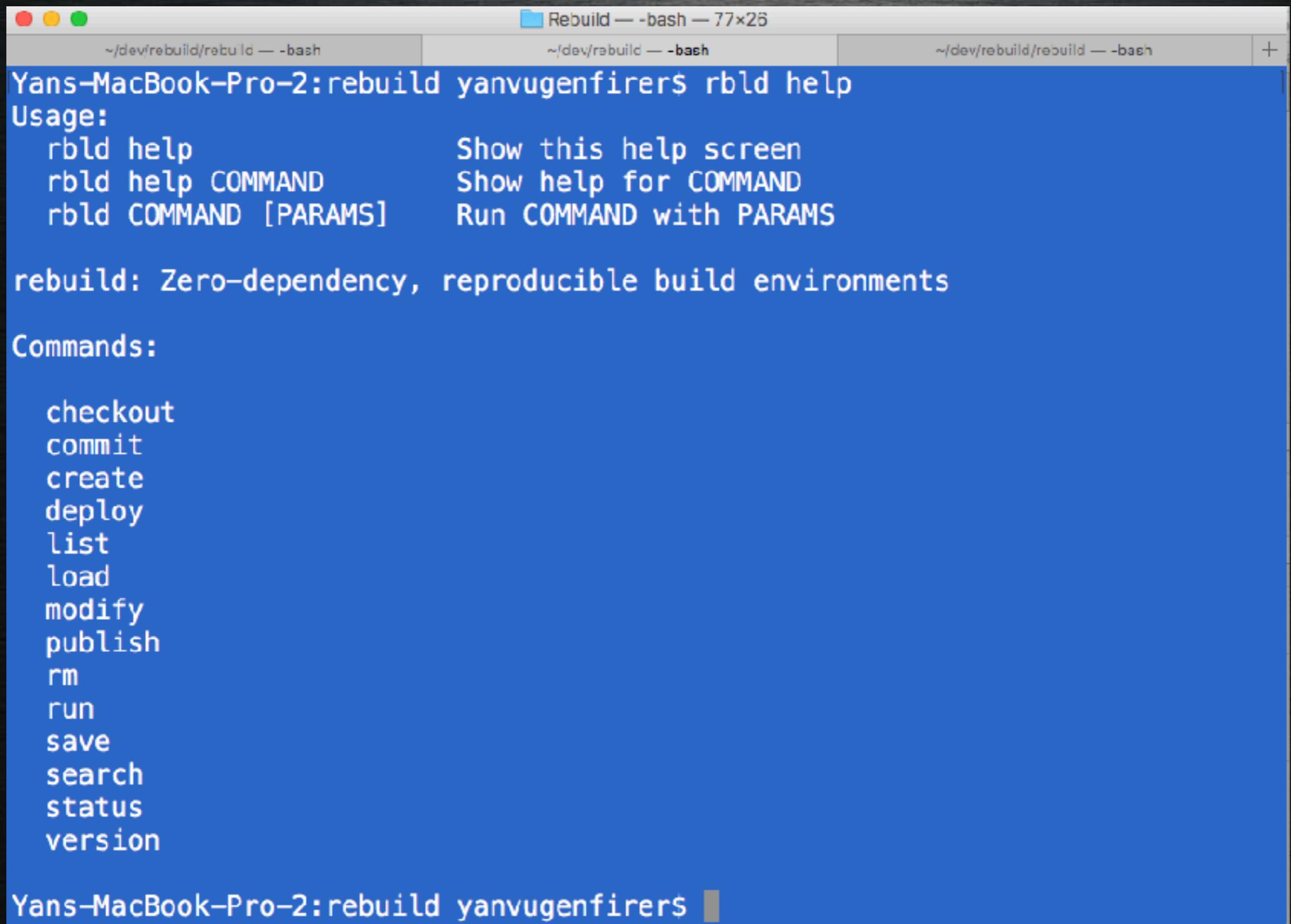
A screenshot of a macOS terminal window titled "Rebuild — -bash — 71x21". The window contains the following text output from a terminal session:

```
Yans-MacBook-Pro-2:rebuild yanvugenfirer$ gem install rbld
Fetching: rbld-1.2.0.gem (100%)
Building native extensions. This could take a while...
Thanks for installing rebuild. Run `rbld help` for CLI reference.
Successfully installed rbld-1.2.0
Parsing documentation for rbld-1.2.0
Installing ri documentation for rbld-1.2.0
Done installing documentation for rbld after 0 seconds
1 gem installed
Yans-MacBook-Pro-2:rebuild yanvugenfirer$
```

# Quick start

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

# Get Help



A screenshot of a macOS terminal window titled "Rebuild — bash — 77x26". The window contains the output of the command "rbld help". The output is organized into sections: "Usage:", "Commands:", and a summary of "rebuild".

```
~/dev/rebuild/rebuild — bash
~/dev/rebuild — bash
~/dev/rebuild/rebuild — bash +1

Yans-MacBook-Pro-2:rebuild yanvugenfirer$ 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

Yans-MacBook-Pro-2:rebuild yanvugenfirer$
```

# Configuration: DockerHub

- » \$HOME/.rbl/rbld/rebuild.conf

```
#DockerHub
REMOTE_NAME=origin
REMOTE_TYPE_origin="dockerhub"
REMOTE_origin="<NAMESPACE>/<REPOSITORY>"
```

# Configuration: Rebuild registry

- \$HOME/.rbl/rbl/rebuild.conf

```
#Rebuild registry
REMOTE_NAME=origin
REMOTE_TYPE_origin="rebuild"
REMOTE_origin="<ABSOLUTE PATH TO
REGISTRY ROOT DIRECTORY>"
```

# Configuration: Docker Registry

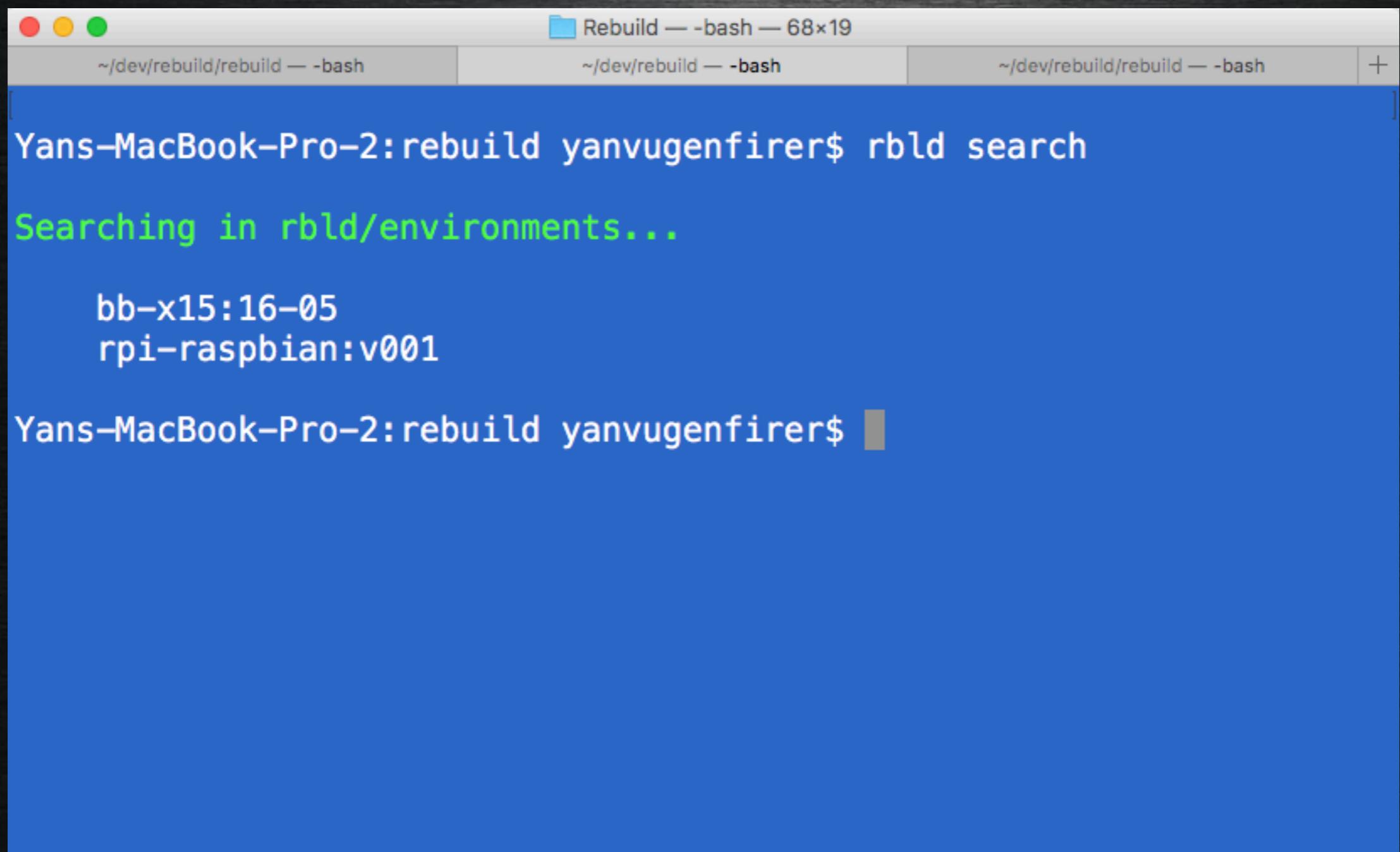
- \$HOME/.rbl/rbld/rebuild.conf

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

# Environment deployment



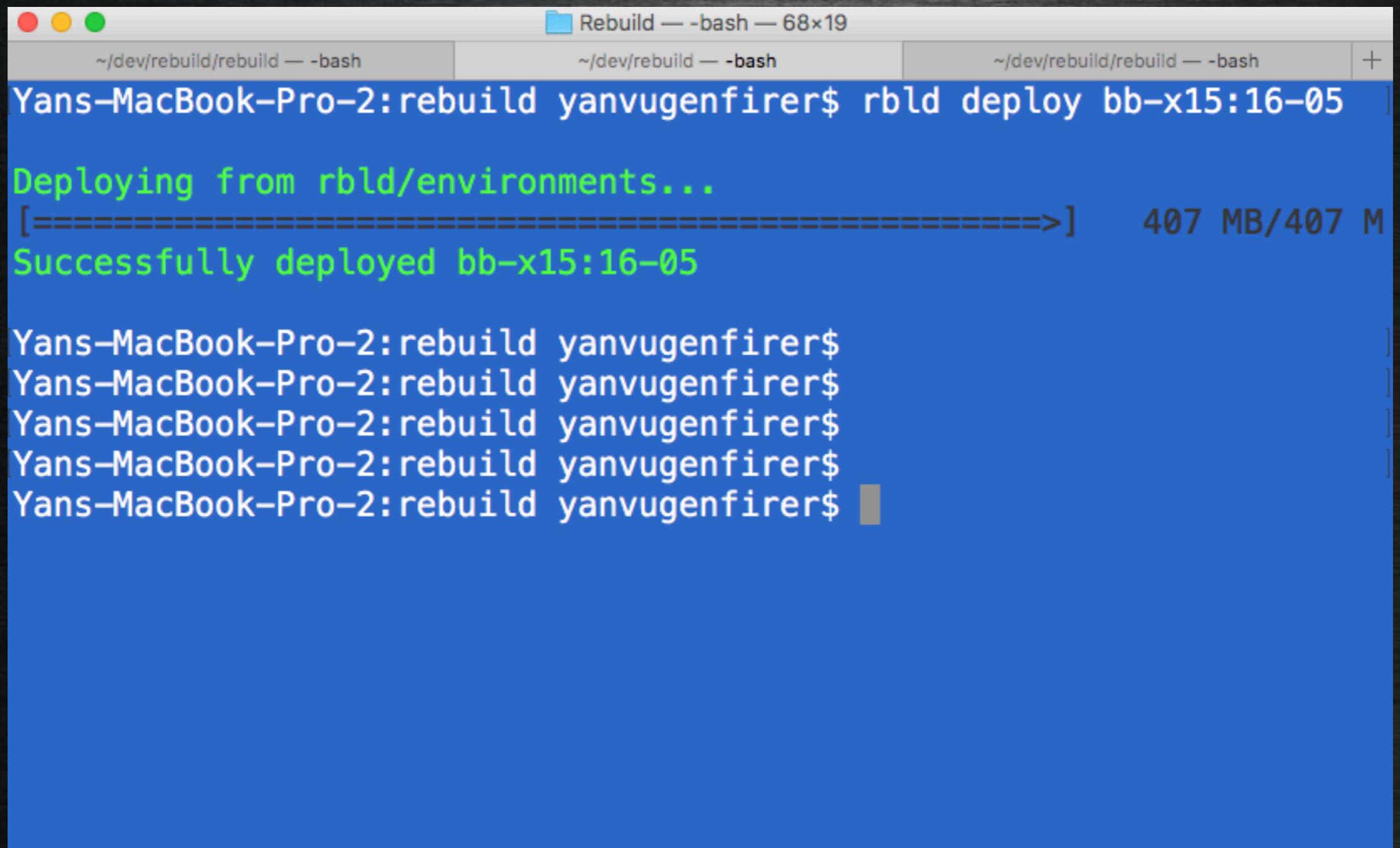
# Environment deployment - search in registry



A screenshot of a terminal window titled "Rebuild — -bash — 68x19". The window has three tabs at the top: "~/dev/rebuild/rebuild — -bash", "~/dev/rebuild — -bash", and "~/dev/rebuild/rebuild — -bash". The main pane shows the command "rbld search" being run, followed by the output "Searching in rbld/environments...". The results listed are "bb-x15:16-05" and "rpi-raspbian:v001". The terminal has a dark background with light blue text.

```
Yans-MacBook-Pro-2: rebuild yanvugenfirer$ rbld search
Searching in rbld/environments...
bb-x15:16-05
rpi-raspbian:v001
Yans-MacBook-Pro-2: rebuild yanvugenfirer$
```

# Environment deployment



A screenshot of a macOS terminal window titled "Rebuild — -bash — 68x19". The window contains three tabs, all showing the same directory: "~/dev/rebuild/rebuild — -bash". The central tab is active and displays the following command and its execution:

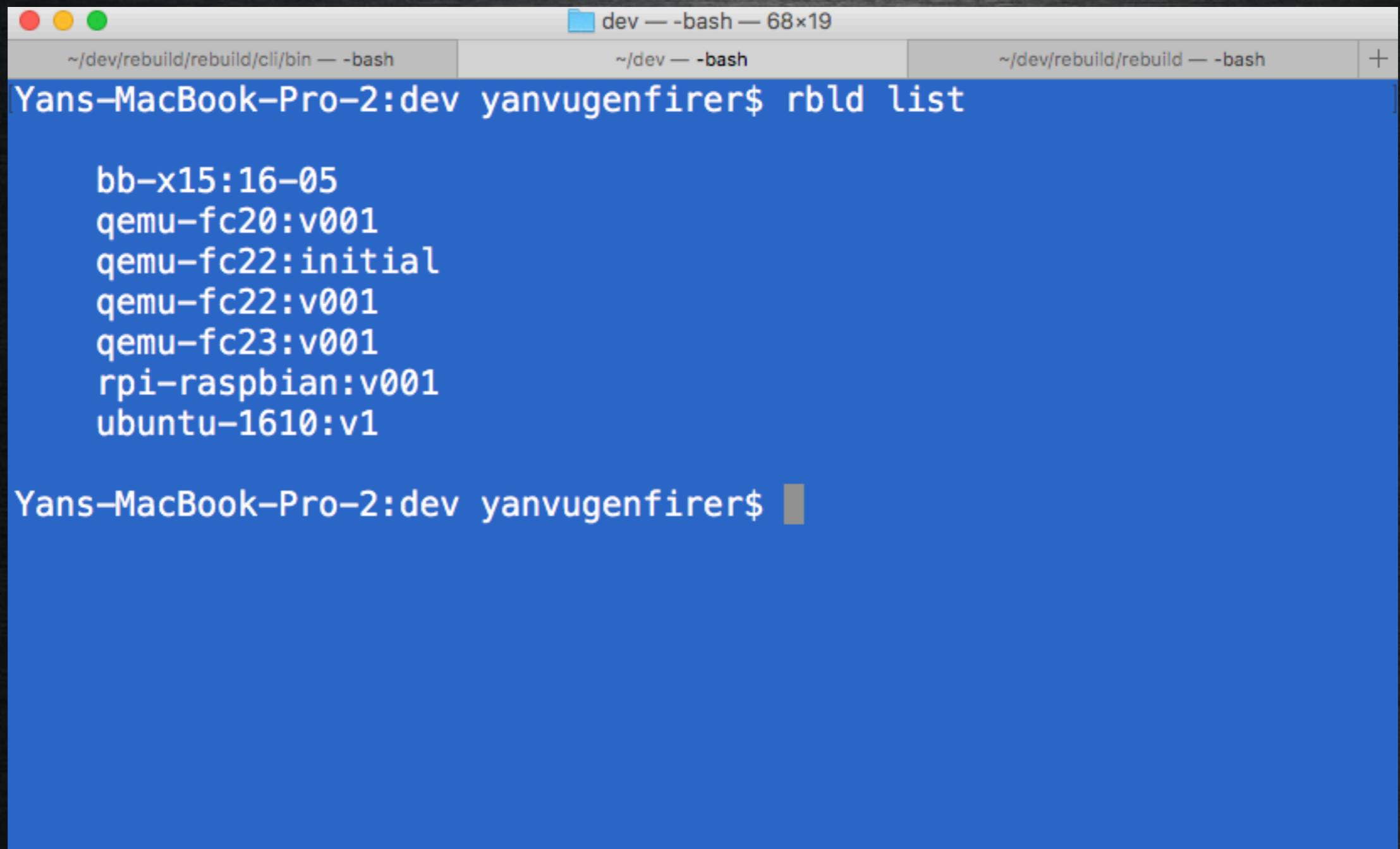
```
Yans-MacBook-Pro-2:rebuild yanvugenfirer$ rbld deploy bb-x15:16-05
```

The output shows the deployment process:

```
Deploying from rbld/environments...
[=====] 407 MB/407 M
Successfully deployed bb-x15:16-05
```

Below this, five blank command lines are visible, indicating the user has typed commands but not yet pressed enter.

# Environment deployment



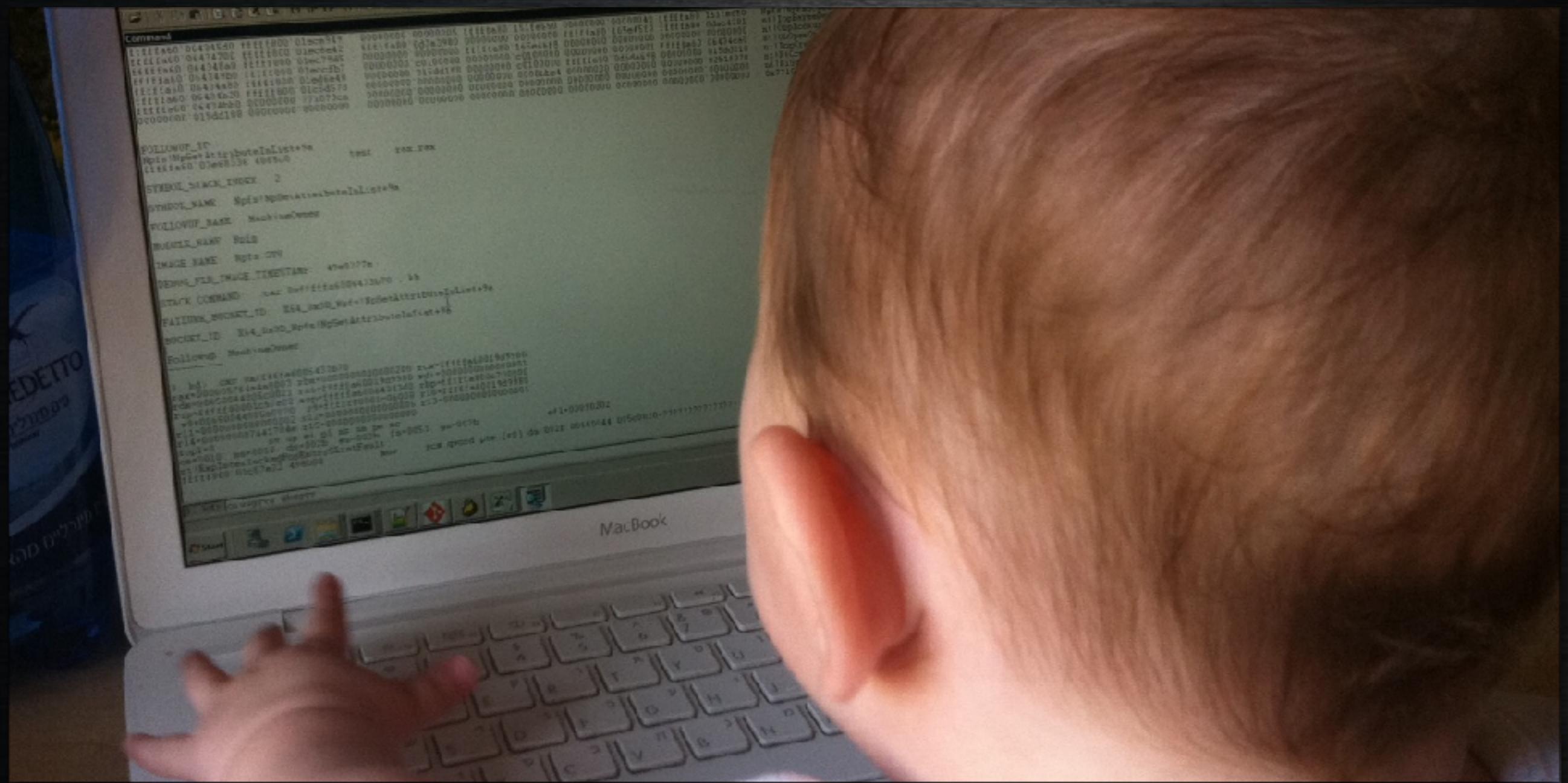
The image shows a macOS terminal window with three tabs. The active tab is titled 'dev — -bash — 68x19' and contains the command 'rbld list' followed by a list of environment configurations. The other two tabs are labeled '~dev/rebuild/rebuild/cli/bin — -bash' and '~dev/rebuild/rebuild — -bash'. The background of the terminal is blue.

```
Yans-MacBook-Pro-2:dev yanvugenfirer$ rbld list

bb-x15:16-05
qemu-fc20:v001
qemu-fc22:initial
qemu-fc22:v001
qemu-fc23:v001
rpi-raspbian:v001
ubuntu-1610:v1

Yans-MacBook-Pro-2:dev yanvugenfirer$
```

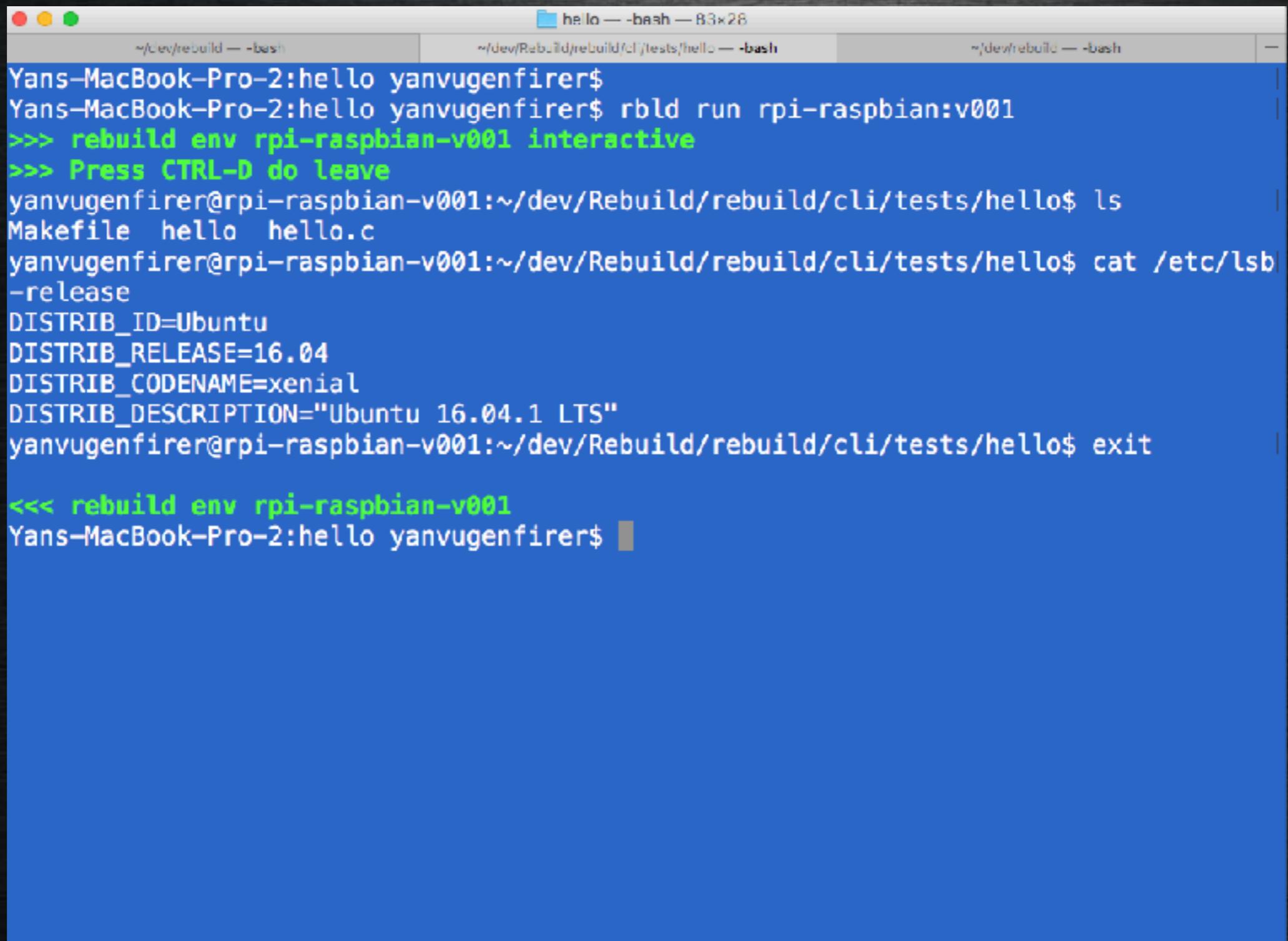
# Usage



# Usage - single command

```
Yans-MacBook-Pro-2:hello yanvugenfirer$ rbld run bb-x15:16-05 -- make
>>> rebuild env bb-x15-16-05
>>> make
<<< rebuild env bb-x15-16-05
Yans-MacBook-Pro-2:hello yanvugenfirer$ ls -la
total 40
drwxr-xr-x  5 yanvugenfirer  staff   170 Mar 20 16:43 .
drwxr-xr-x  3 yanvugenfirer  staff   102 Mar 20 14:53 ..
-rw-r--r--  1 yanvugenfirer  staff   113 Mar 20 14:53 Makefile
-rwxr-xr-x  1 yanvugenfirer  staff  9952 Mar 20 16:43 hello
-rw-r--r--  1 yanvugenfirer  staff   188 Mar 20 14:53 hello.c
Yans-MacBook-Pro-2:hello yanvugenfirer$
```

# Usage - interactive



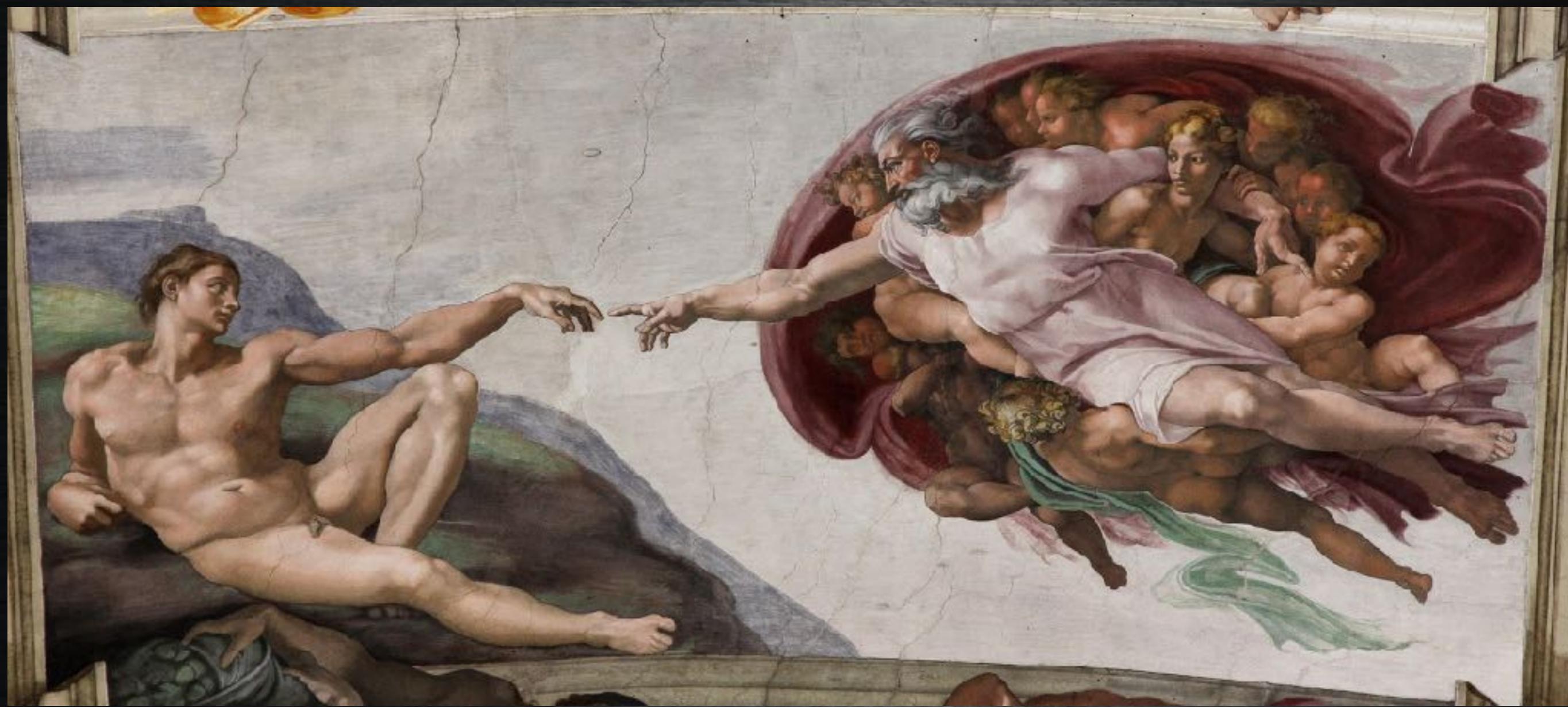
The screenshot shows a macOS terminal window with three tabs:

- Tab 1: ~/Dev/rebuild — bash
- Tab 2: ~/dev/Rebuild/rebuild/cli/tests/hello — bash
- Tab 3: ~/dev/rebuild — bash

The terminal output is as follows:

```
Yans-MacBook-Pro-2:hello yanvugenfirer$  
Yans-MacBook-Pro-2:hello yanvugenfirer$ rbl d 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 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  
Yans-MacBook-Pro-2:hello yanvugenfirer$
```

# Environment creation



# Environment creation

```
[test@rebuild-fedora23-1 ~]$ rblc 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
```

# Environment creation

```
[test@rebuild-fedora23-1 ~]$ rbld create --basefile  
alpine34.tar.gz alpine34
```

Building environment...

...

Successfully created alpine34:initial

# Environment modification



# Modify

```
[test@rebuild-fedora23-1 ~]$ rblt 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-fedora23-1 ~]$ rblt 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-fedora23-1 ~]$ rbld commit --tag v001
```

# Revert changes

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

# Environment-wide variables

```
[test@rebuild-fedora23-1 ~]$ rbld modify ubuntu1510:initial  
  
Initializing environment [.....]  
>>> rebuild env ubuntu1510:initial-M interactive  
>>> Press CTRL-D do leave  
test@ubuntu1510:initial-M:~$ sudo vi /rebuild/rebuild.rc  
test@ubuntu1510:initial-M:~$ cat /rebuild/rebuild.rc  
# 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
```

# Putting all together

```
git clone git://github.com/raspberrypi/tools.git rpi-tools  
  
rbld create --base ubuntu:16.04 rpi-raspbian  
  
rbld modify rpi-raspbian:initial  
  
>> sudo apt-get update  
>> sudo apt-get install -y make  
>> TOOLCHAIN=gcc-linaro-arm-linux-gnueabihf-raspbian-x64  
>> sudo cp -r rpi-tools/arm-bcm2708/$TOOLCHAIN /  
>> echo export CC=$TOOLCHAIN/bin/arm-linux-gnueabihf- |  
sudo tee -a /rebuild/rebuild.rc  
>> exit  
  
rbld commit rpi-raspbian --tag v001
```

# Sharing environments - publishing to registry

```
[test@rebuild-fedora23-1 ~]$ rbl d publish qemu-fc22:v001  
Checking for collisions...  
Publishing at 10.0.110.110:5000...  
...  
Successfully published qemu-fc22:v001
```

# Sharing environments - saving and loading from file

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

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

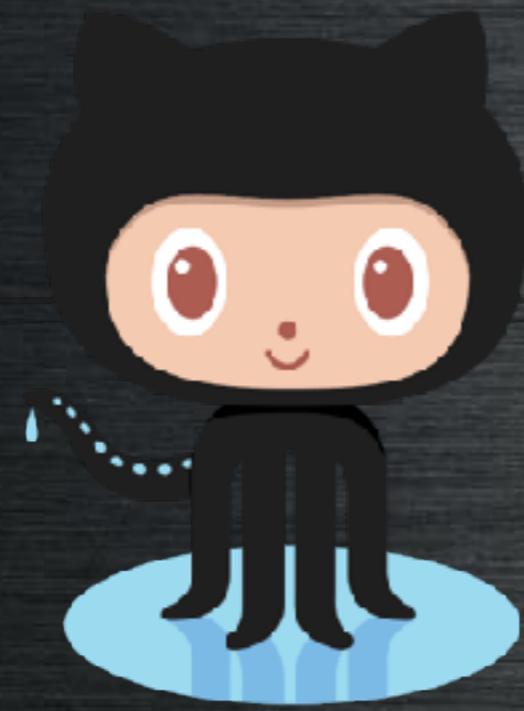
```
[test@rebuild-fedora23-1 ~]$ rbld load qemu-fc20-v001.rbld
```

```
Successfully loaded environment from qemu-fc20-v001.rbld
```

# 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.



<https://github.com/rblד/>  
rebuild

# Summary

- Seamless usage
- Isolated environments
- Easy to share the environments between team members

# Future plans

- Enterprise Dashboard with role management
- Multiple registries support
- Support for additional image registries
- USB redirection for non-Linux hosts

# Q&A



rbl<sup>d</sup>@rbl<sup>d</sup>.io

www.rbl<sup>d</sup>.io

<https://github.com/rbl<sup>d</sup>/rebuild>