



The Dockerfile explosion

Gareth Rushgrove

Senior Software Engineer
Puppet

dockercon 16

The Dockerfile explosion and the need for higher level tools

Introductions

Who am I and what am I doing here

@garethr





puppet

**The shortest path
to better software.**

Built the Puppet Docker module

The screenshot shows the Puppet Forge page for the 'gareth/docker' module. The page layout includes a navigation bar with links for Home, Forge, Docs, Learn, Support & Services, and Contact Us. A search bar is present with the text 'Search from 4,249 modules' and a 'Find' button. The main content area features a profile for 'gareth/docker' by Gareth Rushgrove, with a description 'Module for installing and managing docker'. It lists compatibility with Puppet Enterprise versions (2016.1.x, 2015.3.x, 2015.2.x, 3.8.x, 3.7.x, 3.3.x, 3.2.x) and Puppet >= 3.4.0, along with supported operating systems (RedHat, Ubuntu, Debian, CentOS, Gentoo, Archlinux, Fedora). A 'Quality Score' of 5.0 is shown with a green bar and a note '58% improvement with last release'. A 'Community Rating' of 4.3 is also displayed. A 'Tell us about this module' section contains several survey questions with radio button options. A code block shows the command 'puppet module install gareth-docker' and a 'download latest tar.gz' button. The page also indicates 'Version 5.2.0 released Mar 30th 2016' and has a 'README' tab selected.

gareth/docker · Puppet Fo x

https://forge.puppet.com/gareth/docker

Home Forge Docs Learn Support & Services Contact Us

puppetforge Search from 4,249 modules Find

Puppet Publish Module Sign Out

gareth/docker by: Gareth Rushgrove [Project URL](#) [Report issues](#) [RSS Feed](#)

Module for installing and managing docker

Latest version is compatible with:

- Puppet Enterprise 2016.1.x, 2015.3.x, 2015.2.x, 3.8.x, 3.7.x, 3.3.x, 3.2.x
- Puppet >= 3.4.0
- RedHat, Ubuntu, Debian, CentOS, Gentoo, Archlinux, Fedora

Tags: [ubuntu](#), [lxc](#), [redhat](#), [centos](#), [docker](#)

Use this command to install the latest compatible version:

```
puppet module install gareth-docker
```

[Learn about installing and upgrading modules](#) [download latest tar.gz](#) **348,099** Latest version: 29,387

Quality Score **5.0**
58% improvement with last release [details](#)

Community Rating **4.3**
Based on 57 questions answered [details](#)

Tell us about this module

How helpful are the docs? not at all helpful

How easy to use? hard easy

Does what it promises? Y N

Works without changes? Y N

Used in production? Y N

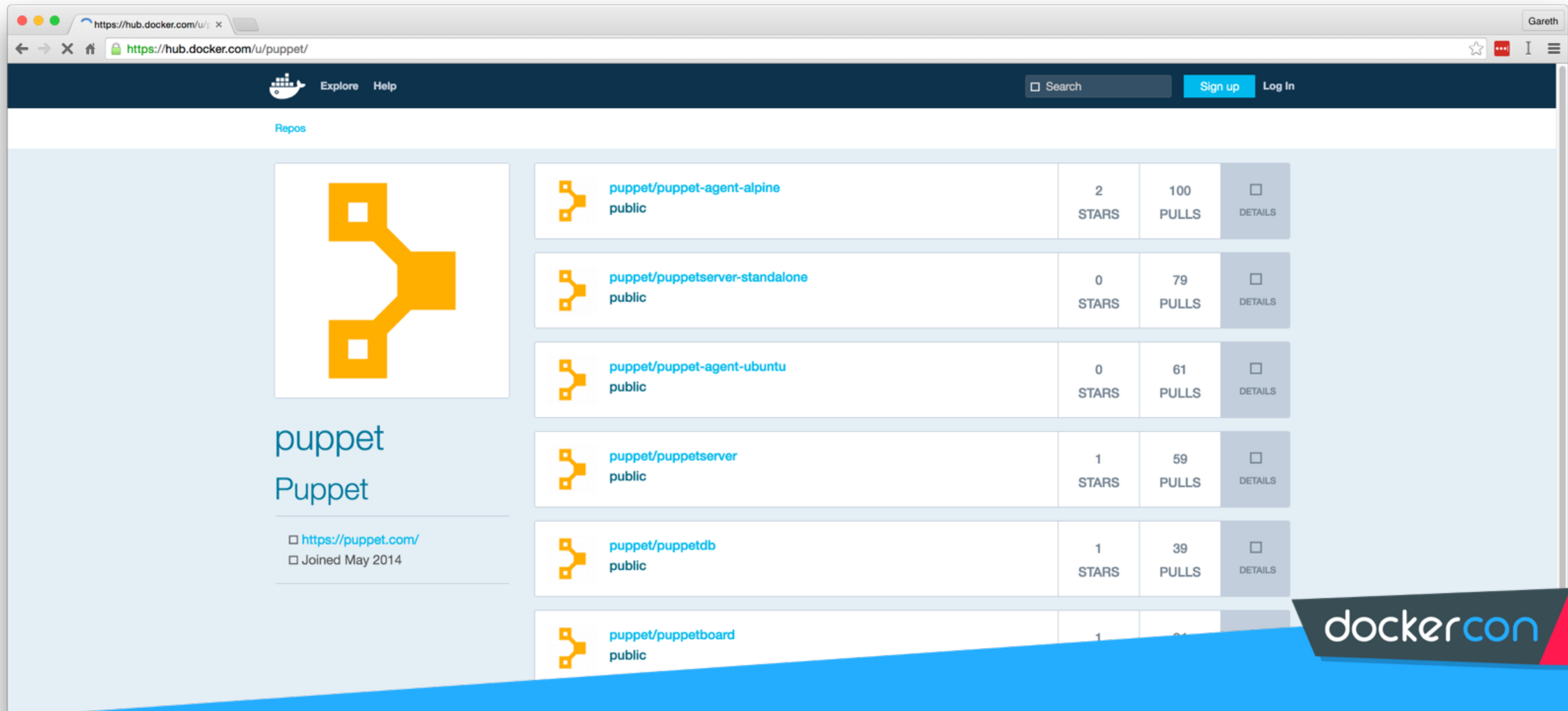
Select another release: **5.2.0** [Download](#)

Version **5.2.0** released **Mar 30th 2016**

[README](#) [Changelog](#) [Dependencies](#) [Compatibility](#) [License](#) [Scores](#) [Issues \(25\)](#)

Puppet module for installing, configuring and managing Docker from the official repositories and EPEL on RedHat based distributions

Maintain the Puppet images



The screenshot shows the Docker Hub user profile for 'puppet'. The profile includes a repository icon, the name 'puppet', the description 'Puppet', a link to 'https://puppet.com/', and the text 'Joined May 2014'. A list of repositories is displayed with columns for repository name, stars, pulls, and a details link.

Repository	Stars	Pulls	Details
puppet/puppet-agent-alpine public	2	100	DETAILS
puppet/puppetserver-standalone public	0	79	DETAILS
puppet/puppet-agent-ubuntu public	0	61	DETAILS
puppet/puppetserver public	1	59	DETAILS
puppet/puppetdb public	1	39	DETAILS
puppet/puppetboard public	1	31	DETAILS

Obsessed with metadata

The screenshot shows a web browser window displaying a Speaker Deck presentation. The browser's address bar shows the URL: <https://speakerdeck.com/garethr/shipping-manifests-bill-of-lading-and-docker-metadata-and-container>. The page header includes the Speaker Deck logo, a search bar, and links for 'Sign Up' and 'Sign In'. The main content area features a video player with a blue and red graphic background. The video title is 'Shipping Manifests, Bill of Lading and Docker Metadata and Container' by Gareth Rushgrove, Senior Software Engineer at Puppet Labs. The video player includes a progress bar and a 'share' button. To the right of the video player, there is a profile card for Gareth Rushgrove, showing 45 presentations, 11 stars, and 1,513 views. Below the profile card, there are options to 'Share' (Twitter, Facebook), 'Embed', 'Direct Link', and 'Download PDF'. The bottom of the page features a blue banner with the 'dockercon 16' logo.

Shipping Manifests, Bill of Lading and Docker - Metadata and Container
by Gareth Rushgrove

dockercon 16

A brief history of Dockerfile

Docker can build images automatically by reading the instructions from a **Dockerfile**

From the official docs at <https://docs.docker.com/engine/reference/builder/>

A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.

From the official docs at <https://docs.docker.com/engine/reference/builder/>

A simple Dockerfile

```
FROM ubuntu
# Install vnc, xvfb in order to create a 'fake' display and fire
RUN apt-get update && apt-get install -y x11vnc xvfb firefox
RUN mkdir ~/.vnc
# Setup a password
RUN x11vnc -storepasswd 1234 ~/.vnc/passwd
# Autostart firefox (might not be the best way, but it does the
RUN bash -c 'echo "firefox" >> ~/.bashrc'
EXPOSE 5900
CMD ["x11vnc", "-forever", "-usepw", "-create"]
```

Dockerfile reference

The screenshot shows a web browser window displaying the Dockerfile reference page. The browser's address bar shows the URL `https://docs.docker.com/engine/reference/builder/`. The page features the Docker logo and a navigation menu with links for Support, Training, Docs, Blog, Docker Hub, and a Get Started button. Below the navigation, there are links for Products, Customers, Community, Partners, Company, and Open Source. The main content area is titled "Dockerfile reference" and includes an introductory paragraph about Dockerfiles, a "Usage" section, and a terminal snippet. A sidebar on the left contains a table of contents for the Docker Engine documentation, and a "On this page:" section on the right lists the page's contents.

Dockerfile reference

Docker can build images automatically by reading the instructions from a **Dockerfile**. A **Dockerfile** is a text document that contains all the commands a user could call on the command line to assemble an image. Using **docker build** users can create an automated build that executes several command-line instructions in succession.

This page describes the commands you can use in a **Dockerfile**. When you are done reading this page, refer to the **Dockerfile Best Practices** for a tip-oriented guide.

Usage

The **docker build** command builds an image from a **Dockerfile** and a *context*. The build's context is the files at a specified location **PATH** or **URL**. The **PATH** is a directory on your local filesystem. The **URL** is the location of a Git repository.

A context is processed recursively. So, a **PATH** includes any subdirectories and the **URL** includes the repository and its submodules. A simple build command that uses the current directory as context:

```
$ docker build .  
Sending build context to Docker daemon 6.51 MB  
...
```

On this page:

- Dockerfile reference
- Usage
- Format
 - Environment replacement
 - .dockerignore file
- FROM
- MAINTAINER
- RUN
 - Known issues (RUN)
- CMD
- LABEL
- EXPOSE
- ENV
- ADD
- COPY
- ENTRYPOINT
 - Exec form ENTRYPOINT example
 - Shell form ENTRYPOINT example
 - Understand how CMD and ENTRYPOINT interact
- VOLUME
- USER

Commands you know

```
MAINTAINER <name>
RUN <command>
CMD ["executable", "param1", "param2"]
EXPOSE <port> [<port>...]
ADD <src>... <dest>
ENV <key> <value>
WORKDIR /path/to/workdir
USER daemon
VOLUME ["/data"]
ENTRYPOINT ["executable", "param1", "param2"]
COPY <src> <dest>
```

Commands you don't know

```
ONBUILD [INSTRUCTION]
```

```
STOPSIGNAL signal
```

```
ARG <name>[=<default value>]
```

```
LABEL <key>=<value> <key>=<value> <key>=<value> ...
```

```
HEALTHCHECK [OPTIONS] CMD command
```

```
SHELL ["executable", "parameters"]
```

Close ALL the issues



jfrazelle commented on Jul 10, 2015



Hello!

We are no longer accepting patches to the Dockerfile syntax as you can read about here:

<https://github.com/docker/docker/blob/master/ROADMAP.md#22-dockerfile-syntax>

Mainly:

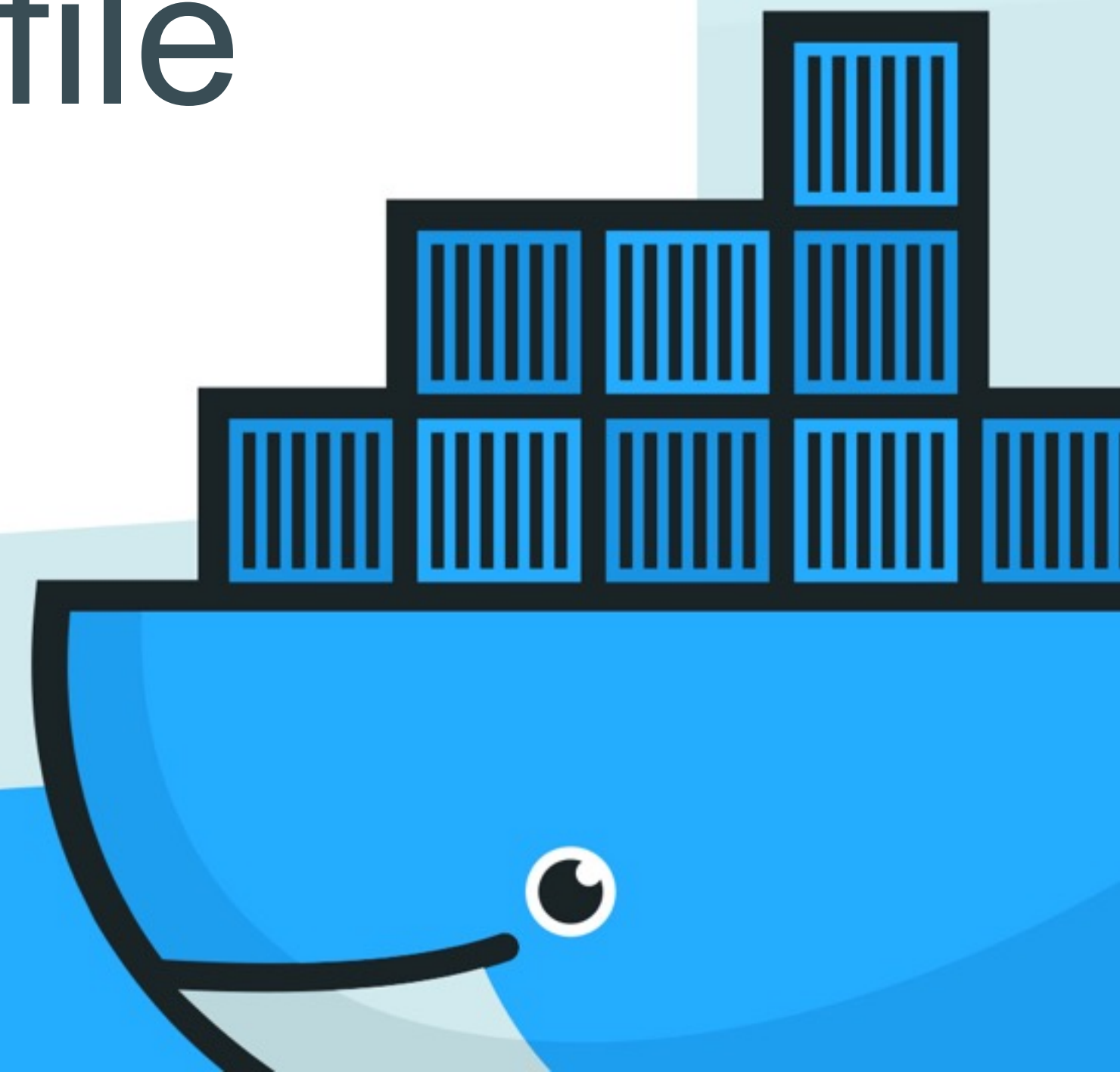
Allowing the Builder to be implemented as a separate utility consuming the Engine's API will open the door for many possibilities, such as offering alternate syntaxes or DSL for existing languages without cluttering the Engine's codebase

Then from there, patches/features like this can be re-thought. Hope you can understand.

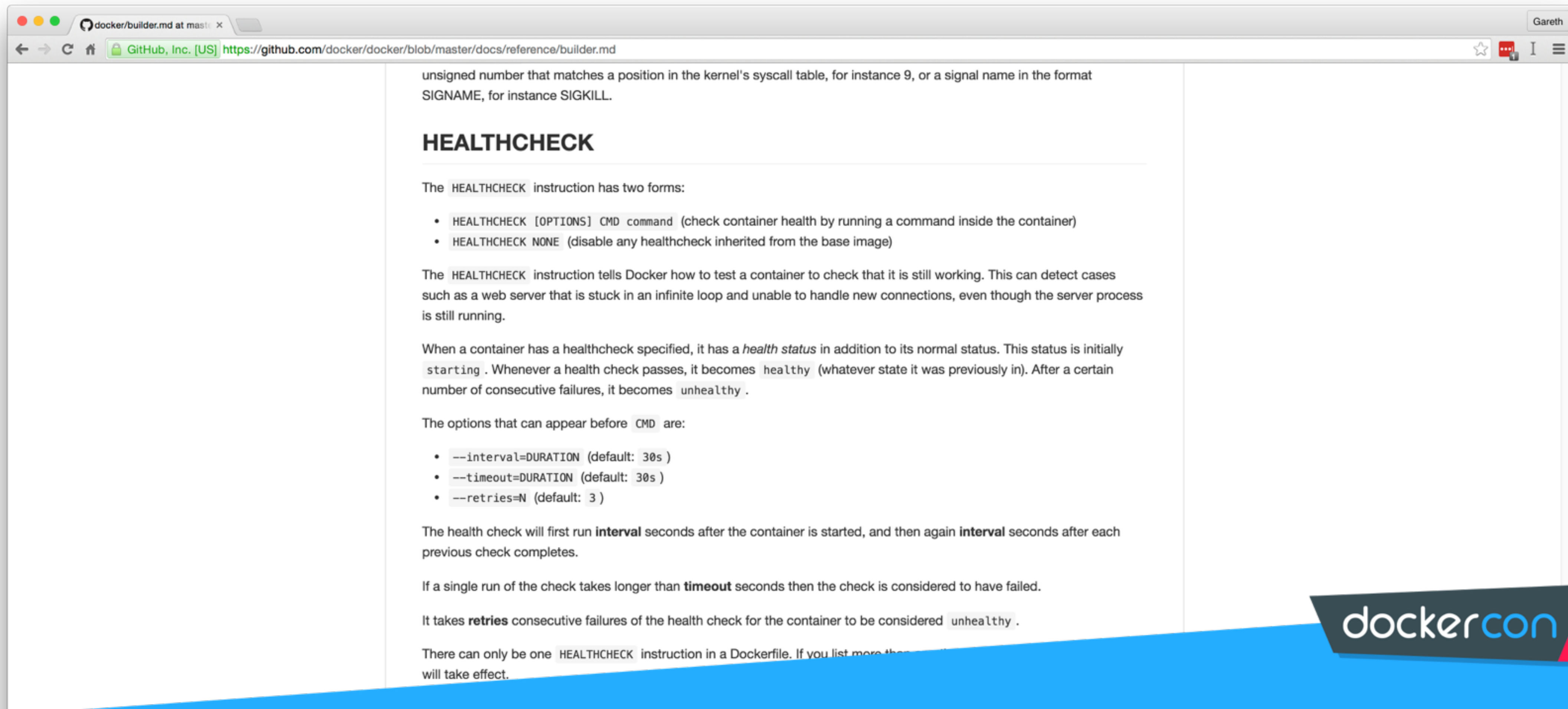


jfrazelle closed this on Jul 10, 2015

Although this is not a definitive move, we temporarily won't accept more patches to the Dockerfile syntax for several reasons



HEALTHCHECK coming in 1.12



docker/builder.md at master · Docker · GitHub

GitHub, Inc. [US] <https://github.com/docker/docker/blob/master/docs/reference/builder.md>

unsigned number that matches a position in the kernel's syscall table, for instance 9, or a signal name in the format `SIGNAME`, for instance `SIGKILL`.

HEALTHCHECK

The `HEALTHCHECK` instruction has two forms:

- `HEALTHCHECK [OPTIONS] CMD command` (check container health by running a command inside the container)
- `HEALTHCHECK NONE` (disable any healthcheck inherited from the base image)

The `HEALTHCHECK` instruction tells Docker how to test a container to check that it is still working. This can detect cases such as a web server that is stuck in an infinite loop and unable to handle new connections, even though the server process is still running.

When a container has a healthcheck specified, it has a *health status* in addition to its normal status. This status is initially `starting`. Whenever a health check passes, it becomes `healthy` (whatever state it was previously in). After a certain number of consecutive failures, it becomes `unhealthy`.

The options that can appear before `CMD` are:

- `--interval=DURATION` (default: 30s)
- `--timeout=DURATION` (default: 30s)
- `--retries=N` (default: 3)

The health check will first run **interval** seconds after the container is started, and then again **interval** seconds after each previous check completes.

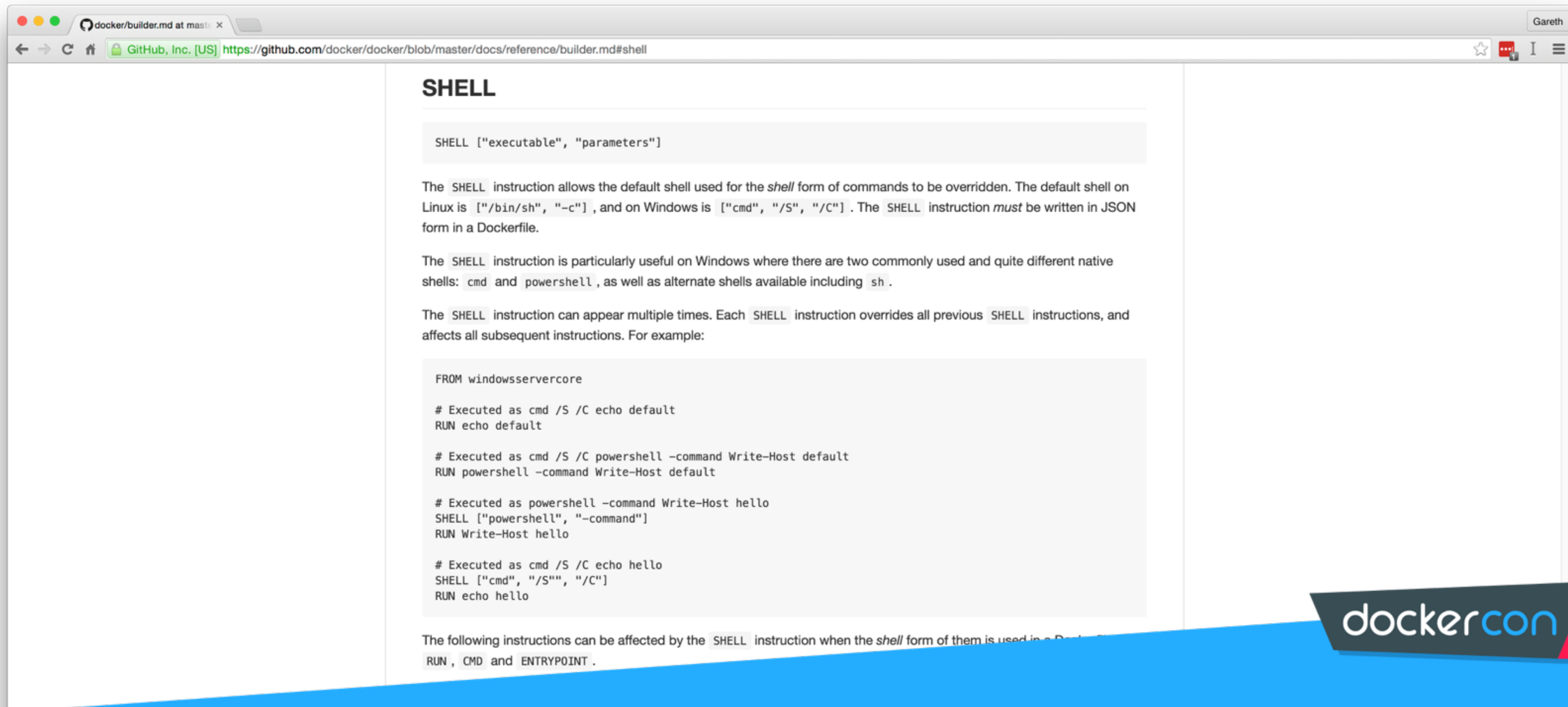
If a single run of the check takes longer than **timeout** seconds then the check is considered to have failed.

It takes **retries** consecutive failures of the health check for the container to be considered `unhealthy`.

There can only be one `HEALTHCHECK` instruction in a Dockerfile. If you list more than one, only the last one will take effect.

Gareth

SHELL coming in 1.12



The screenshot shows a web browser window displaying the Dockerfile documentation for the SHELL instruction. The browser's address bar shows the URL: `https://github.com/docker/docker/blob/master/docs/reference/builder.md#shell`. The page title is "SHELL".

```
SHELL ["executable", "parameters"]
```

The SHELL instruction allows the default shell used for the *shell* form of commands to be overridden. The default shell on Linux is `["/bin/sh", "-c"]`, and on Windows is `["cmd", "/S", "/C"]`. The SHELL instruction *must* be written in JSON form in a Dockerfile.

The SHELL instruction is particularly useful on Windows where there are two commonly used and quite different native shells: `cmd` and `powershell`, as well as alternate shells available including `sh`.

The SHELL instruction can appear multiple times. Each SHELL instruction overrides all previous SHELL instructions, and affects all subsequent instructions. For example:

```
FROM windowsservercore

# Executed as cmd /S /C echo default
RUN echo default

# Executed as cmd /S /C powershell -command Write-Host default
RUN powershell -command Write-Host default

# Executed as powershell -command Write-Host hello
SHELL ["powershell", "-command"]
RUN Write-Host hello

# Executed as cmd /S /C echo hello
SHELL ["cmd", "/S", "/C"]
RUN echo hello
```

The following instructions can be affected by the SHELL instruction when the *shell* form of them is used in a Dockerfile: `RUN`, `CMD` and `ENTRYPOINT`.

Why Dockerfiles are great

Simplicity

```
FROM scratch  
COPY hello /  
CMD ["/hello"]
```

Multi-platform support

```
PS> Install-PackageProvider ContainerImage -Force
```

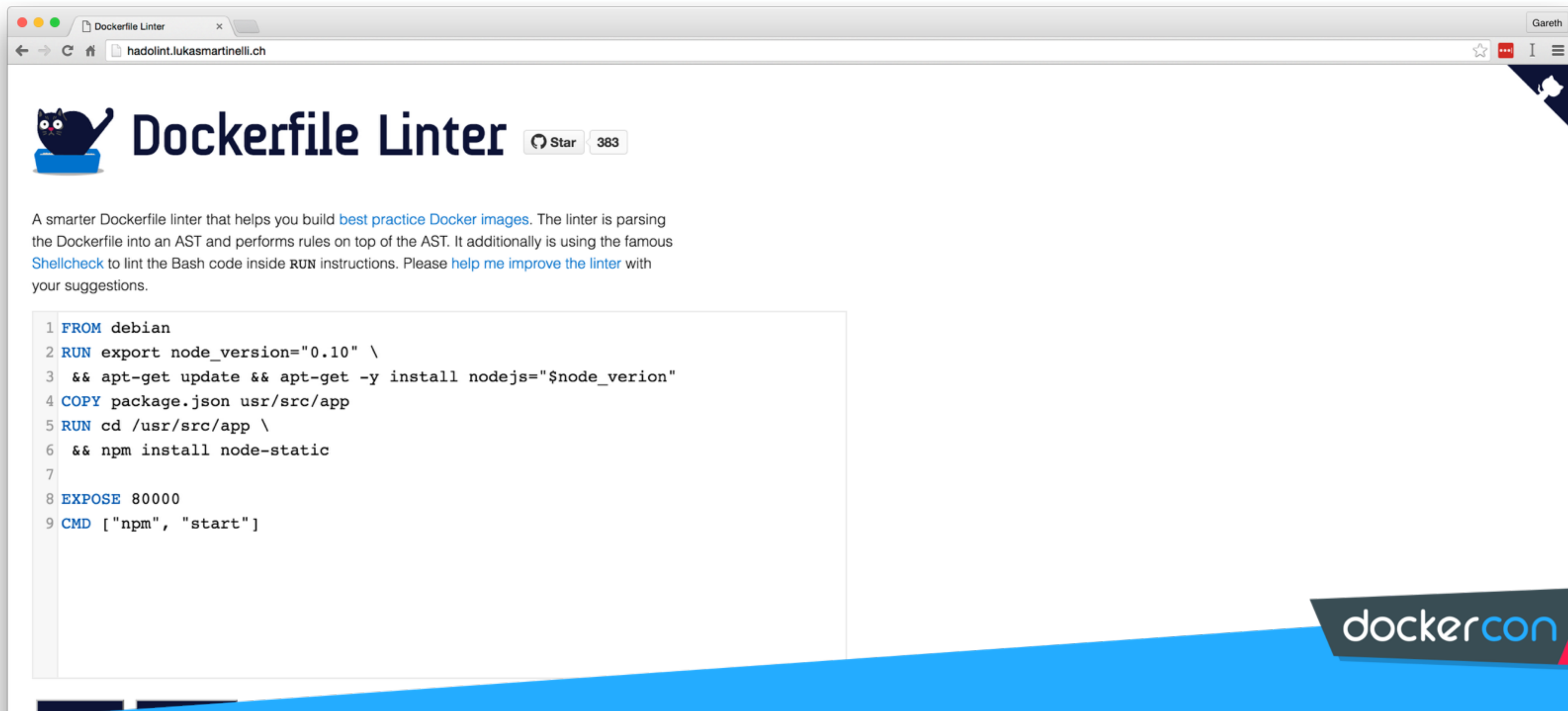
```
PS> Install-ContainerImage -Name WindowsServerCore
```

```
PS> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED
windowsservercore	10.0.14300.1000	dbfee88ee9fd	7 weeks ago

Emerging tooling

Linting



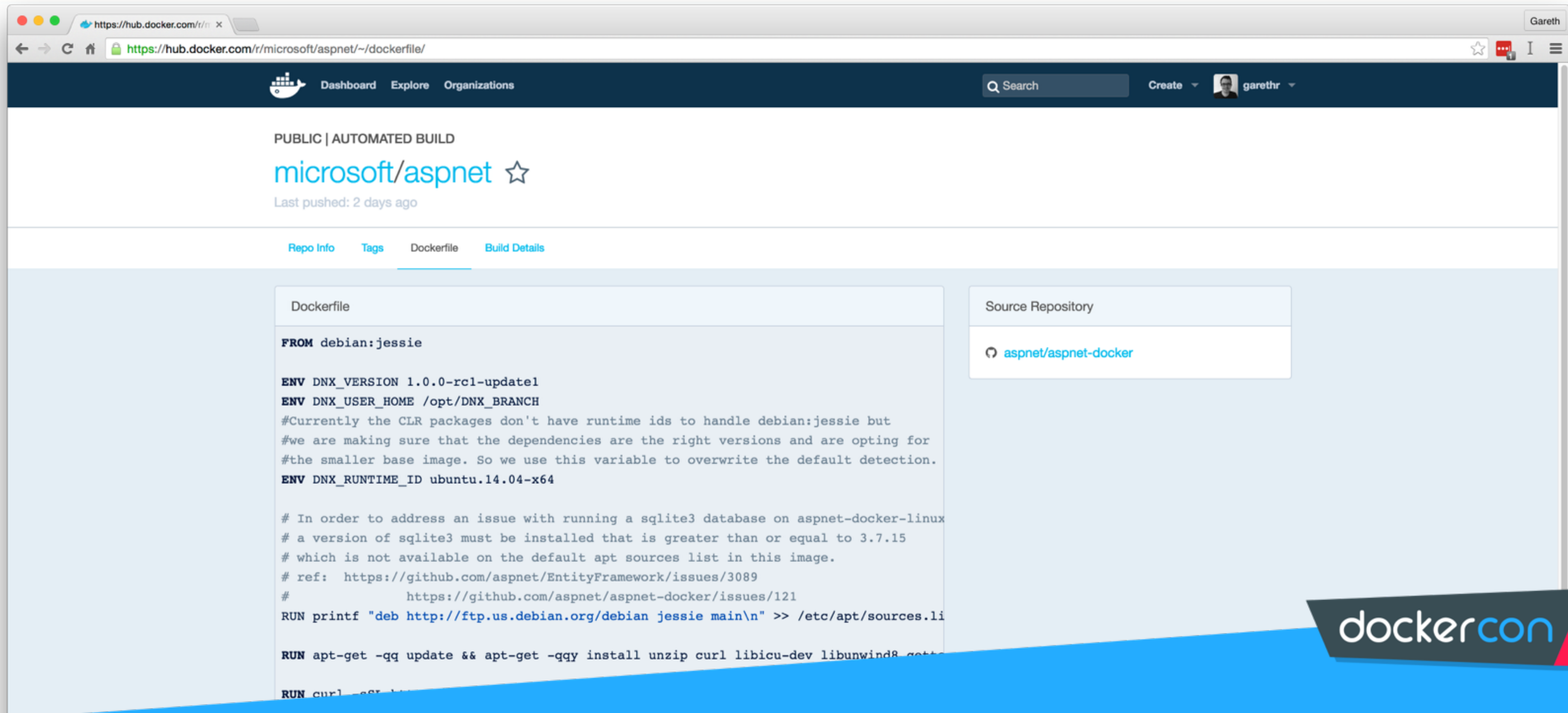
The screenshot shows a web browser window with the URL `hadolint.lukasmartinelli.ch`. The page features the Dockerfile Linter logo, which is a black cat sitting on a blue box. The title "Dockerfile Linter" is displayed in a large, bold font, followed by a GitHub Star button showing 383 stars. Below the title, there is a paragraph of text: "A smarter Dockerfile linter that helps you build [best practice Docker images](#). The linter is parsing the Dockerfile into an AST and performs rules on top of the AST. It additionally is using the famous [Shellcheck](#) to lint the Bash code inside `RUN` instructions. Please [help me improve the linter](#) with your suggestions." Below this text is a code editor with a light blue background and a line number column on the left. The code is as follows:

```
1 FROM debian
2 RUN export node_version="0.10" \
3   && apt-get update && apt-get -y install nodejs="$node_verion"
4 COPY package.json usr/src/app
5 RUN cd /usr/src/app \
6   && npm install node-static
7
8 EXPOSE 80000
9 CMD ["npm", "start"]
```


Editor support

```
1. tmux
1 FROM alpine:3.3
2 MAINTAINER Gareth Rushgrove "gareth@puppet.com"
3
4 ENV PUPPET_EXPLORER_VERSION="2.0.0"
5
6 LABEL com.puppet.version=$PUPPET_EXPLORER_VERSION com.puppet.git.repo="https://github.com/puppetlabs/dockerfiles" com.puppet.git.sha="ca4b5726dd0e244b04ed7f84d6568f799">
7
8 RUN apk add --no-cache --update ca-certificates && \
9     rm -rf /var/cache/apk/*
10
11 RUN wget "https://caddyserver.com/download/build?os=linux&arch=amd64&features=cors,jsonp,prometheus,realip" -O - | tar -xz --no-same-owner -C /usr/bin/ caddy
12
13 RUN wget https://github.com/spotify/puppetexplorer/releases/download/"$PUPPET_EXPLORER_VERSION"/puppetexplorer-"$PUPPET_EXPLORER_VERSION".tar.gz -O - | tar -xz && \
14     ln -s puppetexplorer-"$PUPPET_EXPLORER_VERSION" /puppetexplorer
15
16 # This patch fixes https://github.com/spotify/puppetexplorer/issues/56 until a new release of puppetexplorer is made
17 RUN sed -i -e 's/puppetlabs\.puppetdb\.query\.population/puppetlabs\.puppetdb\.population/g' -e 's/type=default,//g' /puppetexplorer/app.js
18
19 COPY Caddyfile /etc/caddy/Caddyfile
20 COPY config.js /puppetexplorer
21
22 EXPOSE 80
23
24 WORKDIR /etc/caddy
25
26 CMD ["/usr/bin/caddy"]
27
28 COPY Dockerfile /
~
~
~
```

Cross platform



The screenshot shows a web browser displaying the Docker Hub page for the repository `microsoft/aspnet`. The page is titled "PUBLIC | AUTOMATED BUILD" and shows the repository name with a star icon and the text "Last pushed: 2 days ago". The navigation tabs include "Repo Info", "Tags", "Dockerfile", and "Build Details". The "Dockerfile" tab is selected, showing the following content:

```
Dockerfile

FROM debian:jessie

ENV DNX_VERSION 1.0.0-rc1-update1
ENV DNX_USER_HOME /opt/DNX_BRANCH
#Currently the CLR packages don't have runtime ids to handle debian:jessie but
#we are making sure that the dependencies are the right versions and are opting for
#the smaller base image. So we use this variable to overwrite the default detection.
ENV DNX_RUNTIME_ID ubuntu.14.04-x64

# In order to address an issue with running a sqlite3 database on aspnet-docker-linux
# a version of sqlite3 must be installed that is greater than or equal to 3.7.15
# which is not available on the default apt sources list in this image.
# ref: https://github.com/aspnet/EntityFramework/issues/3089
# https://github.com/aspnet/aspnet-docker/issues/121
RUN printf "deb http://ftp.us.debian.org/debian jessie main\n" >> /etc/apt/sources.li

RUN apt-get -qq update && apt-get -qqy install unzip curl libicu-dev libunwind8-gett

RUN curl -sSL https://raw.githubusercontent.com/aspnet/aspnet-docker/1.0.0-rc1-update1/
```

On the right side, the "Source Repository" section shows a link to `aspnet/aspnet-docker`.

Why Dockerfiles are problematic

Complexity

```
RUN apt-get update && \  
  apt-get install -y wget=1.17.1-1ubuntu1 && \  
  wget https://apt.example.com/release-"$UBUNTU_CODENAME".deb \  
  dpkg -i release-"$UBUNTU_CODENAME".deb && \  
  rm release-"$UBUNTU_CODENAME".deb && \  
  apt-get update && \  
  apt-get install --no-install-recommends -y package=0.1.2 && \  
  apt-get clean && \  
  rm -rf /var/lib/apt/lists/*
```

Dockerfile proliferation

language:Dockefile maintainer

Search

138,062

Only two approaches to reuse

Inheritance

```
FROM debian:jessie
```

Ctrl

C

Ctrl

V

Dockerfile is not the source of truth for your image



nathan leclaire

I care, I share, I'm Nathan LeClaire.

[About](#) [Archives](#) [RSS](#)

nathan.leclaire@gmail.com

Get my essays about tech delivered
to your inbox

The Dockerfile is not the source of truth for your image

29 Sep 2014

As Docker grows in popularity we at Docker Inc. are very pleased and one of the things we are trying to encourage the most is the clearing up of misconceptions in the community. Things move rapidly in the open source world, so we do our best to educate those who are willing to listen. On that note, there's a few thoughts about Dockerfiles that I want to share.

The Dockerfile is a wonderful creation - it allows you to automate the otherwise tedious process of creating Docker images. A bit of review for those of you who might be scratching your heads right now:

- Docker provides process, network, etc. isolation and a “chroot on steroids” from a given filesystem state.
- You have to get that initial filesystem state somehow.
- You could either roll your own (any Docker on ARM people out there?) from scratch, or use the images provided by a registry. Docker Hub is one such registry.
- You can also create images interactively using a base image and `docker commit`.

`docker commit` is the operation which creates a new image layer in Docker's layered union filesystem (AUFS by default on Debian-based systems).

You can actually see the changes which will be committed with `docker diff`:

```
$ docker run -it ubuntu bash
root@b3a195b117aa:/# mkdir /data
root@b3a195b117aa:/# cd /data
root@b3a195b117aa:/data# touch a.java b.java
root@b3a195b117aa:/data# exit
exit
$ docker diff $(docker ps -lq)
A /data
A /data/a.java
A /data/b.java
C /root
A /root/.bash_history
```

Similar to what you may be familiar with

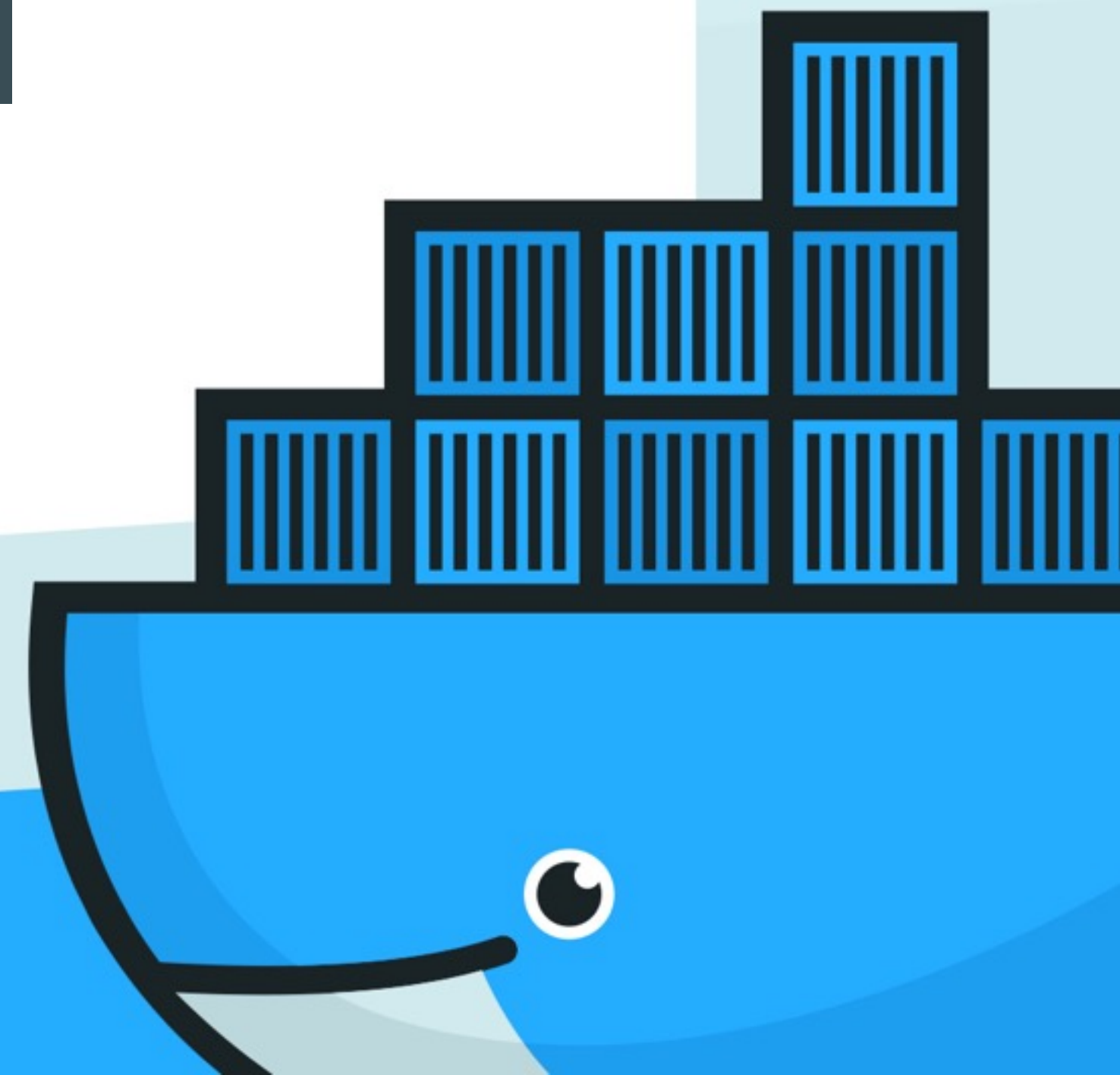
The Dockerfile generally works
beautifully for the class of
problem for which it was designed

Nathan Leclair, Docker Inc



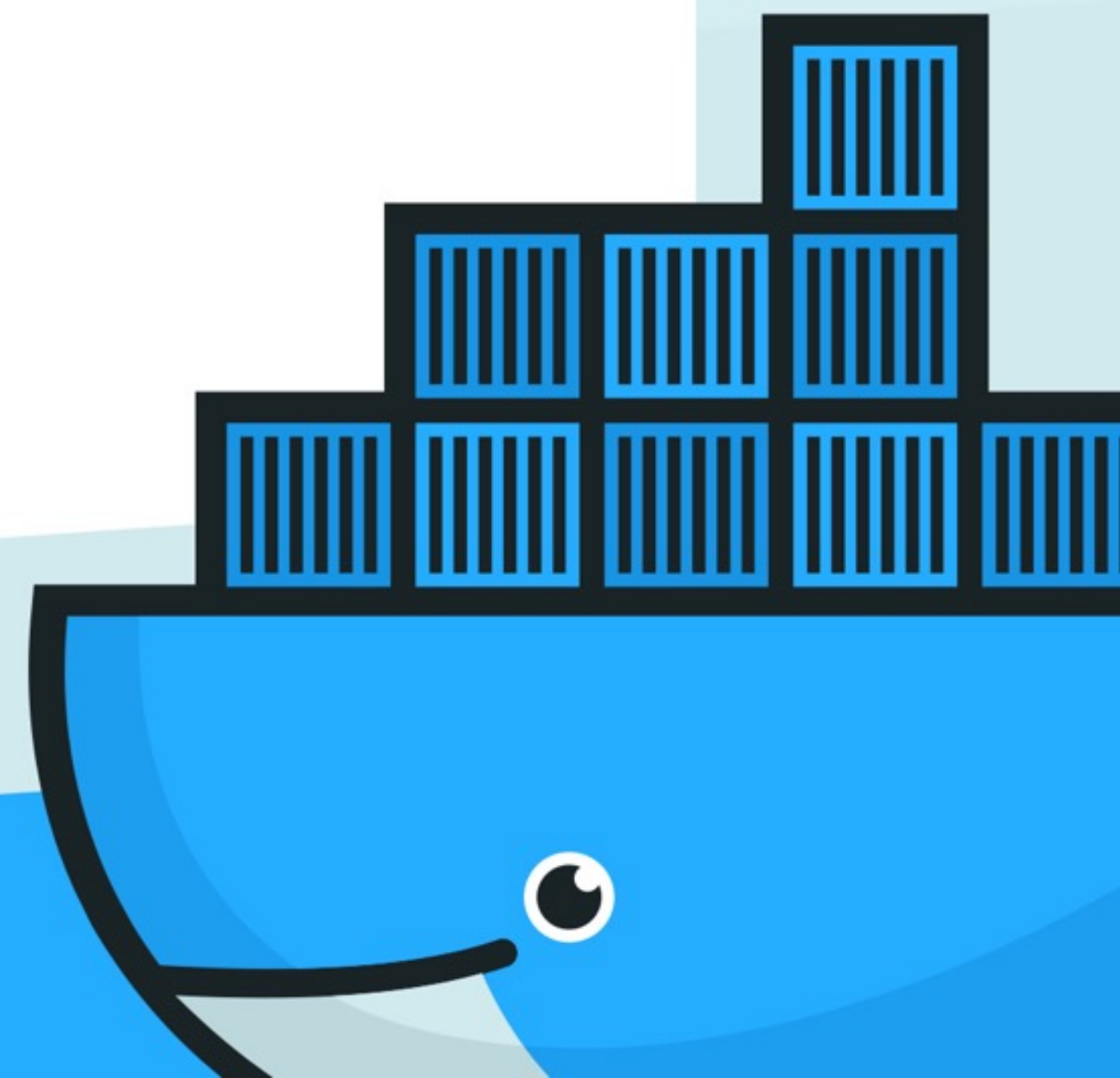
The Dockerfile is a tool for creating images, but it is not the only weapon in your arsenal

Nathan Leclair, Docker Inc

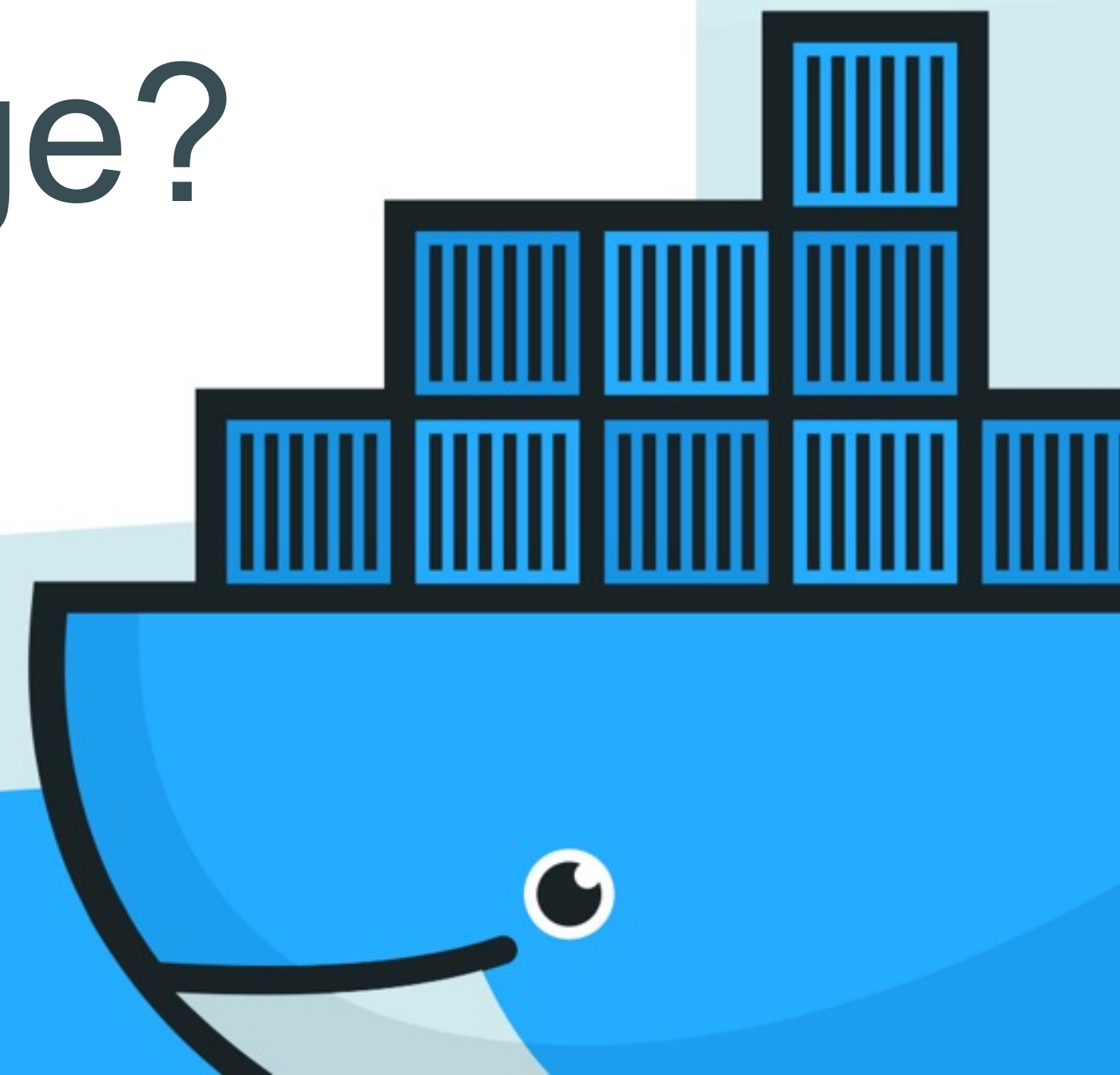


Putting the problems in context

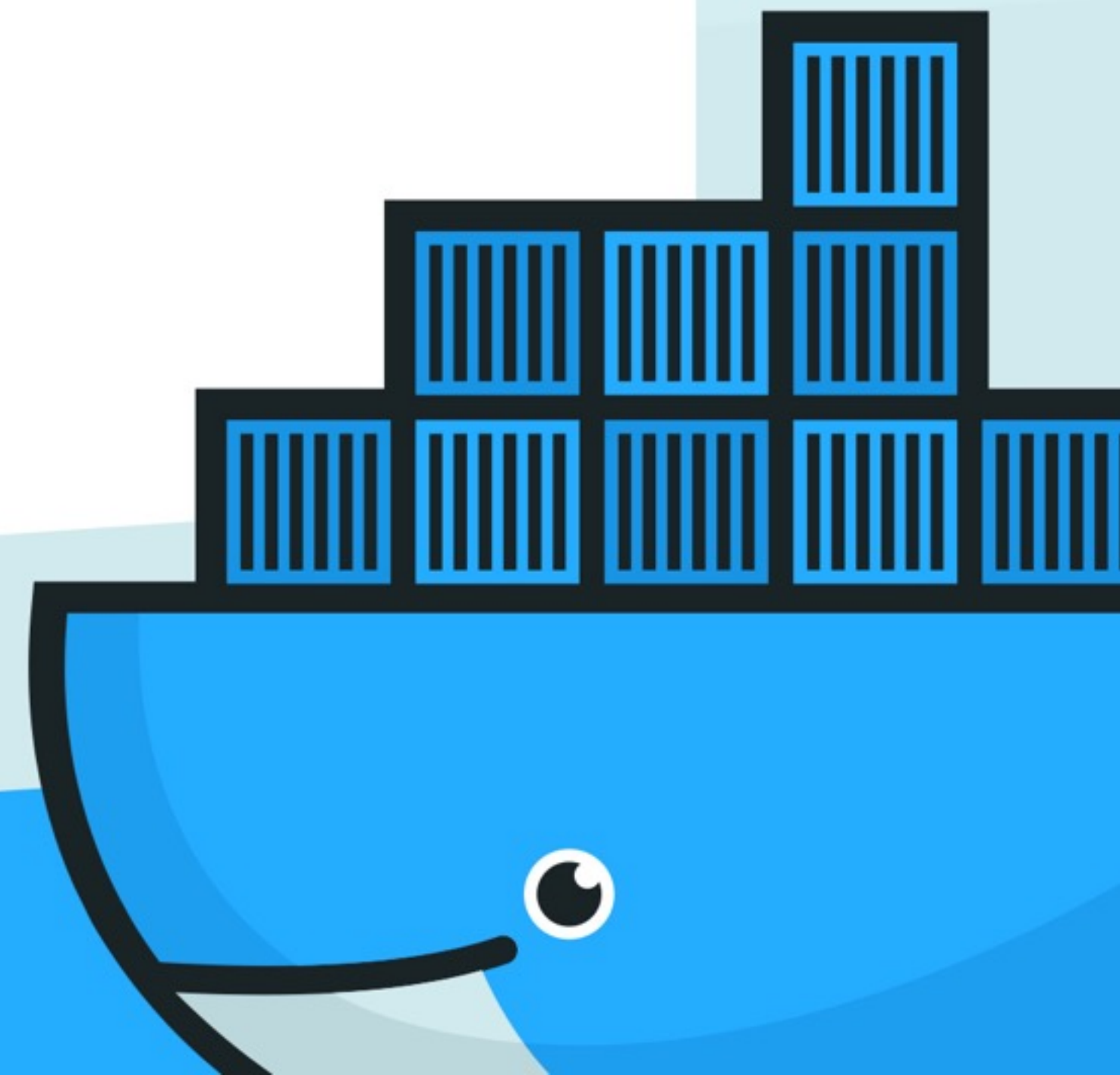
If we dockerize all of our applications how many **Dockerfiles** is that?



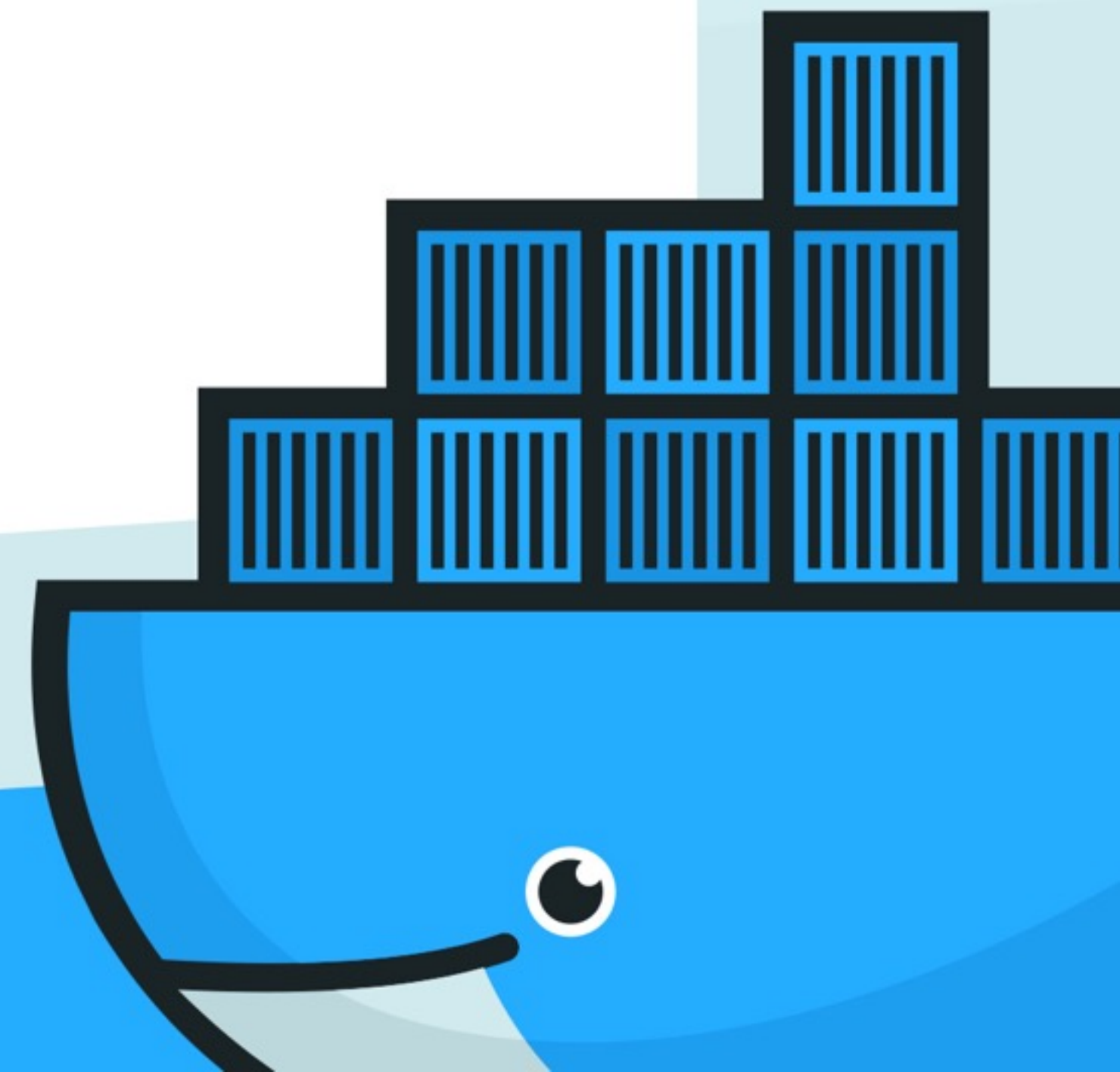
If we build a complex hierarchy of **Dockerfiles**, how quickly can we trace/rebuild a specific image?



As best-practices develops how
can we refactor our **Dockfiles**
with confidence?



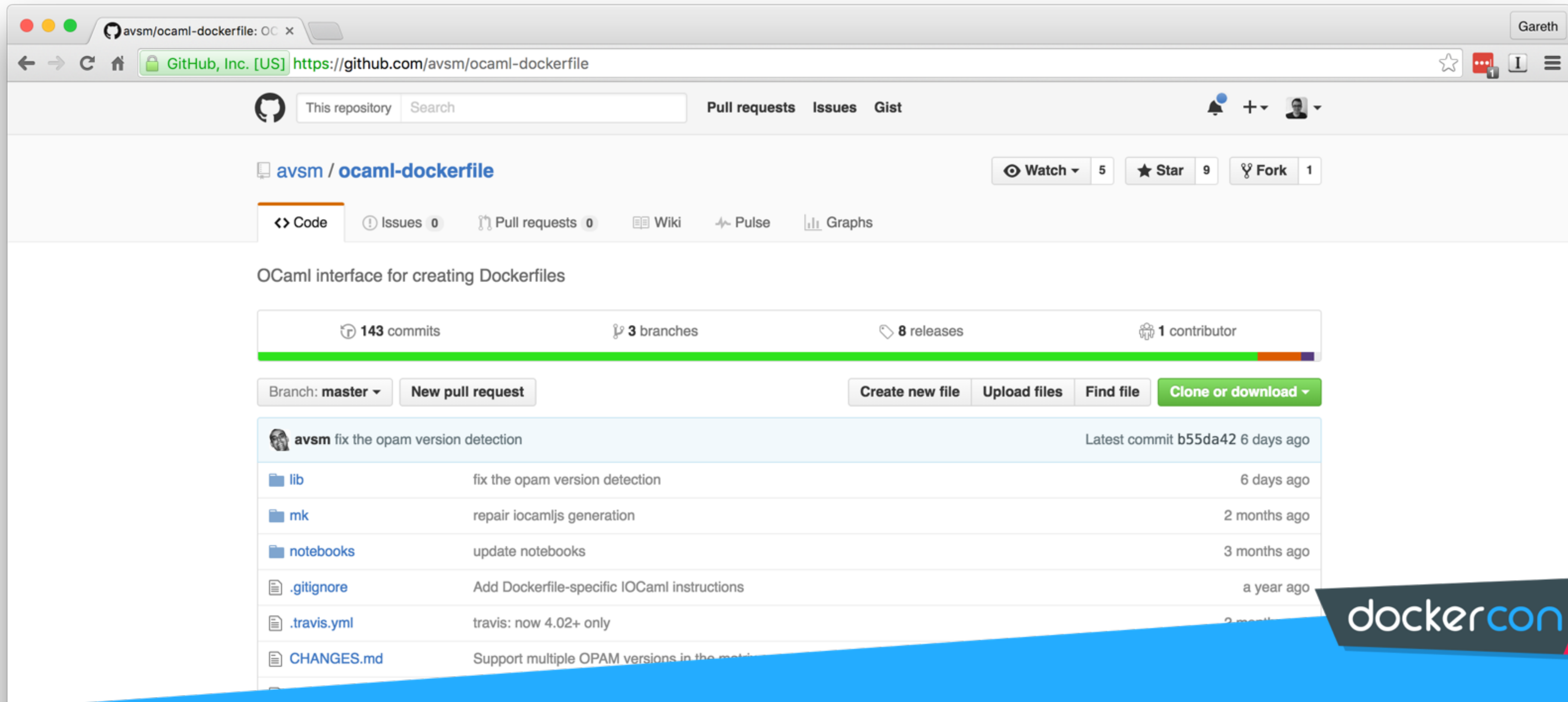
Are **Dockerfiles** best managed centrally or on a team-by-team basis?



Some community ideas

Generate Dockerfiles

Build Dockerfiles with OCAML



The screenshot shows a web browser window displaying the GitHub repository page for `avsm/ocaml-dockerfile`. The browser's address bar shows the URL `https://github.com/avsm/ocaml-dockerfile`. The repository page includes a search bar, navigation links for Pull requests, Issues, and Gist, and a header with the repository name and statistics: 5 Watchers, 9 Stars, and 1 Fork. Below the header, there are tabs for Code, Issues (0), Pull requests (0), Wiki, Pulse, and Graphs. The main content area features a description: "OCaml interface for creating Dockerfiles". A summary bar indicates 143 commits, 3 branches, 8 releases, and 1 contributor. A row of buttons includes "Branch: master", "New pull request", "Create new file", "Upload files", "Find file", and "Clone or download". A commit history table is visible, showing the latest commit by `avsm` titled "fix the opam version detection" 6 days ago. Other commits include "repair iocamljs generation" (2 months ago), "update notebooks" (3 months ago), "Add Dockerfile-specific IOcaml instructions" (a year ago), "travis: now 4.02+ only" (2 months ago), and "Support multiple OPAM versions in the meti".

avsm/ocaml-dockerfile: OC x

GitHub, Inc. [US] <https://github.com/avsm/ocaml-dockerfile>

This repository Search

Pull requests Issues Gist

avsm / **ocaml-dockerfile** Watch 5 Star 9 Fork 1

Code Issues 0 Pull requests 0 Wiki Pulse Graphs

OCaml interface for creating Dockerfiles

143 commits 3 branches 8 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

avsm fix the opam version detection Latest commit b55da42 6 days ago

lib	fix the opam version detection	6 days ago
mk	repair iocamljs generation	2 months ago
notebooks	update notebooks	3 months ago
.gitignore	Add Dockerfile-specific IOcaml instructions	a year ago
.travis.yml	travis: now 4.02+ only	2 months ago
CHANGES.md	Support multiple OPAM versions in the meti	

OCAML example

```
let base =  
  let email = "anil@recoil.org" in  
  comment "Generated by OCaml Dockerfile" @@  
  from "ubuntu" ~tag:"trusty" @@  
  maintainer "Anil Madhavapeddy <%s>" email
```

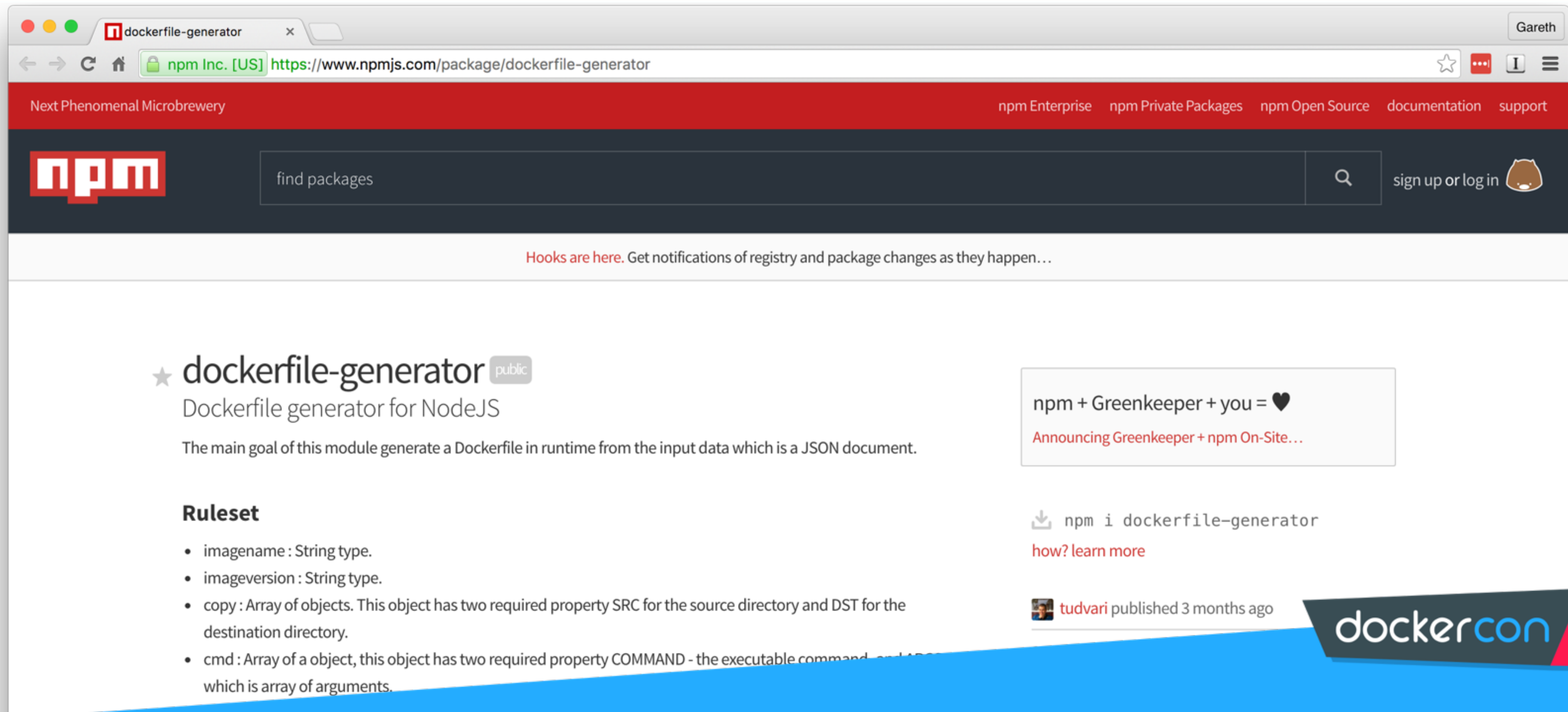
```
let ocaml_ubuntu_image =  
  base @@  
  run "apt-get -y -qq update" @@  
  run "apt-get -y install ocaml ocaml-native-compiler-libs ocaml-ex  
  onbuild (run "apt-get -y -qq update")
```

With Gradle

The screenshot shows a web browser window displaying the GitHub repository page for `Transmode/gradle-docker`. The browser's address bar shows the URL `https://github.com/Transmode/gradle-docker`. The repository page includes a search bar, navigation links for Pull requests, Issues, and Gist, and a user profile for Gareth. The repository name `Transmode / gradle-docker` is prominently displayed, along with statistics: 45 Watchers, 255 Stars, and 55 Forks. Below this, there are tabs for Code, Issues (36), Pull requests (8), Wiki, Pulse, and Graphs. A description of the repository reads: "A Gradle plugin to build Docker images from the build script." A summary bar shows 168 commits, 2 branches, 4 releases, and 18 contributors. Action buttons include "New pull request", "Create new file", "Upload files", "Find file", and "Clone or download". A commit history table is visible, with the latest commit by `mattgruter` on Feb 23. The commit history table contains the following data:

Commit	Description	Time
<code>1241c24</code>	Reverting documentation until we release"	Latest commit
<code>.idea/runConfigurations</code>	IntelliJ test run configurations	8 months ago
<code>codequality</code>	Initial public release 1.0	3 years ago
<code>examples</code>	Use new dockerfile DSL for exposing ports in example project	4 months ago
<code>gradle</code>	Upgrading gradle to version 2.10	4 months ago
<code>src</code>	Upgrade docker-java to 2.2.0	3 months ago
<code>.drone.yml</code>	Drone CI: publish to local maven repo before testing example	
<code>.gitignore</code>		

Or Javascript



The screenshot shows a web browser window with the URL `https://www.npmjs.com/package/dockerfile-generator`. The page features the npm logo, a search bar, and navigation links. The main content area displays the package name `dockerfile-generator` with a 'public' badge, a description: 'Dockerfile generator for NodeJS', and a brief overview: 'The main goal of this module generate a Dockerfile in runtime from the input data which is a JSON document.' Below this is a 'Ruleset' section with a bulleted list of configuration options: `imagename`, `imageversion`, `copy`, and `cmd`. On the right side, there is a promotional box for 'npm + Greenkeeper + you = ❤️' and a code snippet `npm i dockerfile-generator` with a 'how? learn more' link. At the bottom right, a user profile for 'tudvari' is shown with the text 'published 3 months ago'. The browser's address bar and tabs are visible at the top.

Next Phenomenal Microbrewery

npm Enterprise npm Private Packages npm Open Source documentation support

npm find packages sign up or log in

Hooks are here. Get notifications of registry and package changes as they happen...

★ **dockerfile-generator** public
Dockerfile generator for NodeJS
The main goal of this module generate a Dockerfile in runtime from the input data which is a JSON document.

Ruleset

- `imagename` : String type.
- `imageversion` : String type.
- `copy` : Array of objects. This object has two required property SRC for the source directory and DST for the destination directory.
- `cmd` : Array of a object, this object has two required property COMMAND - the executable command, and ARGV which is array of arguments.

npm + Greenkeeper + you = ❤️
Announcing Greenkeeper + npm On-Site...

npm i dockerfile-generator
how? learn more

tudvari published 3 months ago

dockercon 16

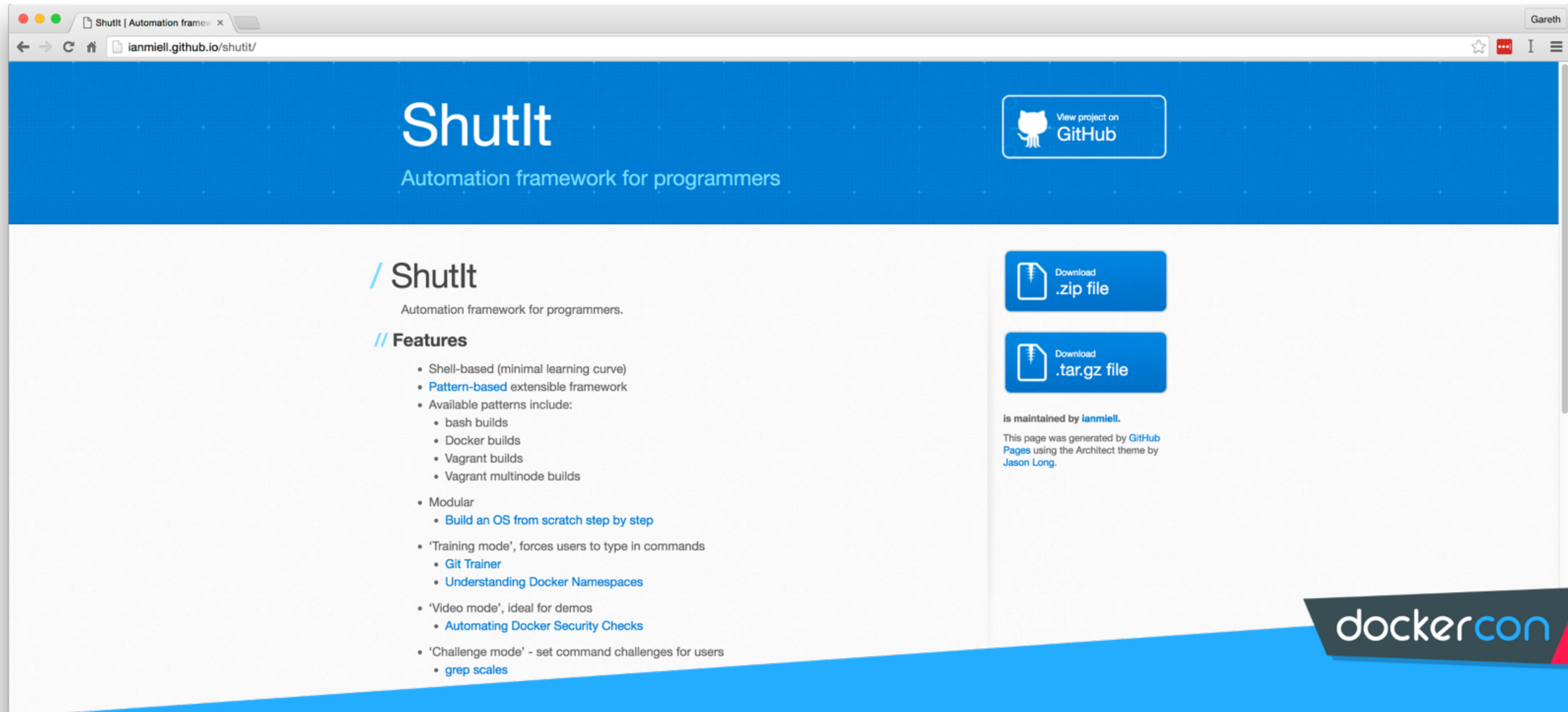
Or Scala and SBT

The screenshot shows a web browser window with the following elements:

- Browser Tab:** Docker Plugin — sbt-native x
- Address Bar:** www.scala-sbt.org/sbt-native-packager/formats/docker.html
- Page Header:** sbt-native-packager 1.1
- Search Bar:** Search docs
- Navigation Menu (Left):**
 - Introduction
 - Getting Started
 - Formats
 - Universal Plugin
 - Linux Plugin
 - Debian Plugin
 - Rpm Plugin
 - Docker Plugin**
 - Requirements
 - Build
 - Configuration
 - Settings
 - Tasks
 - Customize
 - Windows Plugin

- Breadcrumbs:** Docs » Packaging Formats » Docker Plugin
- Page Actions:** View page source
- Section Header:** Docker Plugin
- Text:** Docker images describe how to set up a container for running an application, including what files are present, and what program to run.
- Text:** <https://docs.docker.com/introduction/understanding-docker/> provides an introduction to Docker. <https://docs.docker.com/reference/builder/> describes the Dockerfile; a file which describes how to set up the image.
- Text:** sbt-native-packager focuses on creating a Docker image which can “just run” the application built by SBT.
- Note:** The docker plugin depends on the [Universal Plugin](#).
- Section Header:** Requirements

Or with Python



The screenshot shows a web browser window displaying the ShutIt website. The browser's address bar shows the URL `ianmiell.github.io/shutit/`. The website has a blue header with the title "ShutIt" and the subtitle "Automation framework for programmers". A button in the header says "View project on GitHub". The main content area is white and contains the following text:

/ ShutIt
Automation framework for programmers.

// Features

- Shell-based (minimal learning curve)
- **Pattern-based** extensible framework
- Available patterns include:
 - bash builds
 - Docker builds
 - Vagrant builds
 - Vagrant multinode builds
- Modular
 - [Build an OS from scratch step by step](#)
- 'Training mode', forces users to type in commands
 - [Git Trainer](#)
 - [Understanding Docker Namespaces](#)
- 'Video mode', ideal for demos
 - [Automating Docker Security Checks](#)
- 'Challenge mode' - set command challenges for users
 - [grep scales](#)

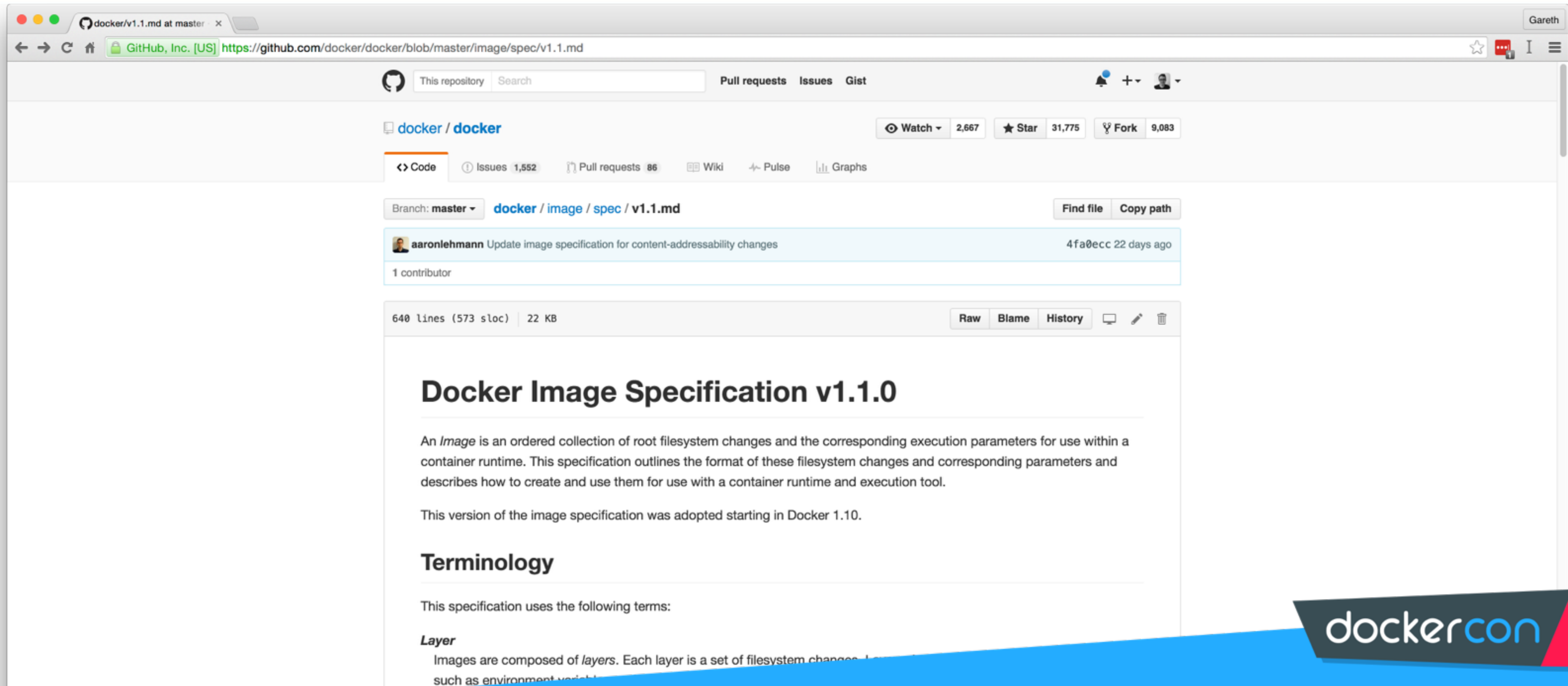
On the right side of the page, there are two blue buttons for downloading the project:

- Download .zip file
- Download .tar.gz file

Below the buttons, it says "is maintained by [ianmiell](#)." and "This page was generated by [GitHub Pages](#) using the [Architect](#) theme by [Jason Long](#)."

**No Dockerfile
to be seen**

Docker Image Specification



The screenshot shows a web browser displaying the Docker Image Specification v1.1.0 page on GitHub. The page is titled "docker / docker" and shows the file path "docker / image / spec / v1.1.md". The commit history shows a commit by aaronlehmann titled "Update image specification for content-addressability changes" with the commit hash 4fa0ecc, made 22 days ago. The file size is 22 KB and it contains 640 lines (573 sloc). The page content includes the title "Docker Image Specification v1.1.0" and a description: "An *Image* is an ordered collection of root filesystem changes and the corresponding execution parameters for use within a container runtime. This specification outlines the format of these filesystem changes and corresponding parameters and describes how to create and use them for use with a container runtime and execution tool. This version of the image specification was adopted starting in Docker 1.10." The page also includes a "Terminology" section with the heading "This specification uses the following terms:" and a sub-section for "Layer" which states "Images are composed of *layers*. Each layer is a set of filesystem changes, such as environment variables".

docker/v1.1.md at master · docker / docker

GitHub, Inc. [US] <https://github.com/docker/docker/blob/master/image/spec/v1.1.md>

This repository Search Pull requests Issues Gist

docker / docker Watch 2,667 Star 31,775 Fork 9,083

Code Issues 1,552 Pull requests 86 Wiki Pulse Graphs

Branch: master docker / image / spec / v1.1.md Find file Copy path

aaronlehmann Update image specification for content-addressability changes 4fa0ecc 22 days ago

1 contributor

640 lines (573 sloc) 22 KB Raw Blame History

Docker Image Specification v1.1.0

An *Image* is an ordered collection of root filesystem changes and the corresponding execution parameters for use within a container runtime. This specification outlines the format of these filesystem changes and corresponding parameters and describes how to create and use them for use with a container runtime and execution tool.

This version of the image specification was adopted starting in Docker 1.10.

Terminology

This specification uses the following terms:

Layer

Images are composed of *layers*. Each layer is a set of filesystem changes, such as environment variables.

opencontainers/image-spec

GitHub, Inc. [US] https://github.com/opencontainers/image-spec

This repository Search Pull requests Issues Gist

opencontainers / image-spec Watch 42 Star 77 Fork 21

Code Issues 18 Pull requests 3 Pulse Graphs

OCI Image Format <https://www.opencontainers.org/>

131 commits 2 branches 2 releases 12 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

request #110 from RobDolinMS/patch-2 Latest commit b72e75a a day ago

*: add license header checking and tweak Makefile	15 days ago
layout: implement unpacking and validation	7 days ago
img: move files in	8 days ago
test: improve examples ha	11 days ago
layout: implement unpacking and validation	7 days ago
*: add license header checking and tweak Makefile	15 days ago
Fix the PullApprove approve_regex to cover more cases	2 days ago
travis: use a `make .gitvalidation` target	2 days ago
*: add common files from runtime-spec	2 months ago
MAINTAINERS: initial commit	2 months ago
travis: use a `make .gitvalidation` target	2 days ago
[ReadMe] remove "should" from FAQ question	2 days ago
[Manifest] Mark digest as required	2 days ago
*: fixup the docs output	2 days ago
*: add common files from runtime-spec	2 months ago
serialization: remove combined spec	7 days ago

.pullapprove.yml
.travis.yml
LICENSE
MAINTAINERS
Makefile
README.md
manifest.md
media-types.md
project.md
serialization.md
README.md

Open Container Initiative Image Format



OPEN CONTAINER INITIATIVE

Packer

The screenshot shows a web browser window with the URL `https://www.packer.io/docs/builders/docker.html`. The page title is "Docker Builder - Packer by HashiCorp". The navigation menu includes "Intro", "Docs", "Community", "Download", and "GitHub". The main content area is titled "DOCKER BUILDER" and describes the builder's type as "docker". It explains that the builder starts a Docker container, runs provisioners, and exports the container. It also notes that Packer builds Docker containers without Dockerfiles, using portable scripts or configuration management systems. A sidebar on the left lists "DOCS" (Installation, Terminology) and "COMMAND-LINE" (Introduction, Build, Fix).

Docker Builder - Packer by HashiCorp

Intro Docs Community Download GitHub

DOCS

- Installation
- Terminology

COMMAND-LINE

- Introduction
- Build
- Fix

DOCKER BUILDER

Type: `docker`

The `docker` Packer builder builds Docker images using Docker. The builder starts a Docker container, runs provisioners within this container, then exports the container for reuse or commits the image.

Packer builds Docker containers *without* the use of Dockerfiles. By not using Dockerfiles, Packer is able to provision containers with portable scripts or configuration management systems that are not tied to Docker in any way. It also has a simpler mental model: you provision containers much the same way you provision a normal virtualized or dedicated server. For more information, read the section on Dockerfiles.

The Docker build

Packer example

```
{
  "builders": [{
    "type": "docker",
    "image": "ubuntu",
    "export_path": "image.tar"
  }],
  "provisioners": [
    {
      "type": "shell",
      "inline": ["apt-get -y update; apt-get install -y net-tools"]
    },
    {
```


Source-to-Image

The screenshot shows a web browser window displaying the OpenShift Enterprise 3.0 documentation page for S2I Requirements. The browser's address bar shows the URL https://docs.openshift.com/enterprise/3.0/creating_images/s2i.html. The page header includes the OpenShift logo, a 'MENU' button, and navigation links for 'MY ACCOUNT' and 'SIGN UP FOR FREE'. The breadcrumb trail reads 'Documentation / OpenShift Enterprise 3.0 / Creating Images / S2I Requirements'. A search bar is located above the left-hand navigation menu. The left-hand navigation menu lists various documentation topics, with 'S2I Requirements' highlighted. The main content area is titled 'S2I Requirements' and includes links for 'Overview', 'Build Process', 'S2I Scripts', and 'Using Images with ONBUILD Instructions'. The 'Overview' section explains that Source-to-Image (S2I) is a framework for creating Docker images from source code. The 'Build Process' section states that it consists of three fundamental elements: sources and S2I scripts.

OpenShift Enterprise 3.0 | x Gareth

https://docs.openshift.com/enterprise/3.0/creating_images/s2i.html

MENU **OPENSHIFT** MY ACCOUNT **SIGN UP FOR FREE**

Documentation / OpenShift Enterprise 3.0 / Creating Images / S2I Requirements

Search

- > OpenShift Documentation
- > What's New?
- > Getting Started
- > Architecture
- > Installation and Configuration
- > Administration
- > CLI Reference
- > Developer Guide
- > Creating Images
 - Overview
 - Guidelines
 - Image Metadata
 - S2I Requirements**
 - Testing S2I Images
 - Custom Builder
 - Revision History
- > Using Images

S2I Requirements

- [Overview](#)
- [Build Process](#)
- [S2I Scripts](#)
- [Using Images with ONBUILD Instructions](#)

Overview

Source-to-Image (S2I) is a framework that makes it easy to write images that take application source code as an input and produce a new image that runs the assembled application as output.

The main advantage of using S2I for building reproducible Docker images is the ease of use for developers. As a builder image author, you must understand two basic concepts in order for your images to provide the best possible S2I performance: [the build process](#) and [S2I scripts](#).

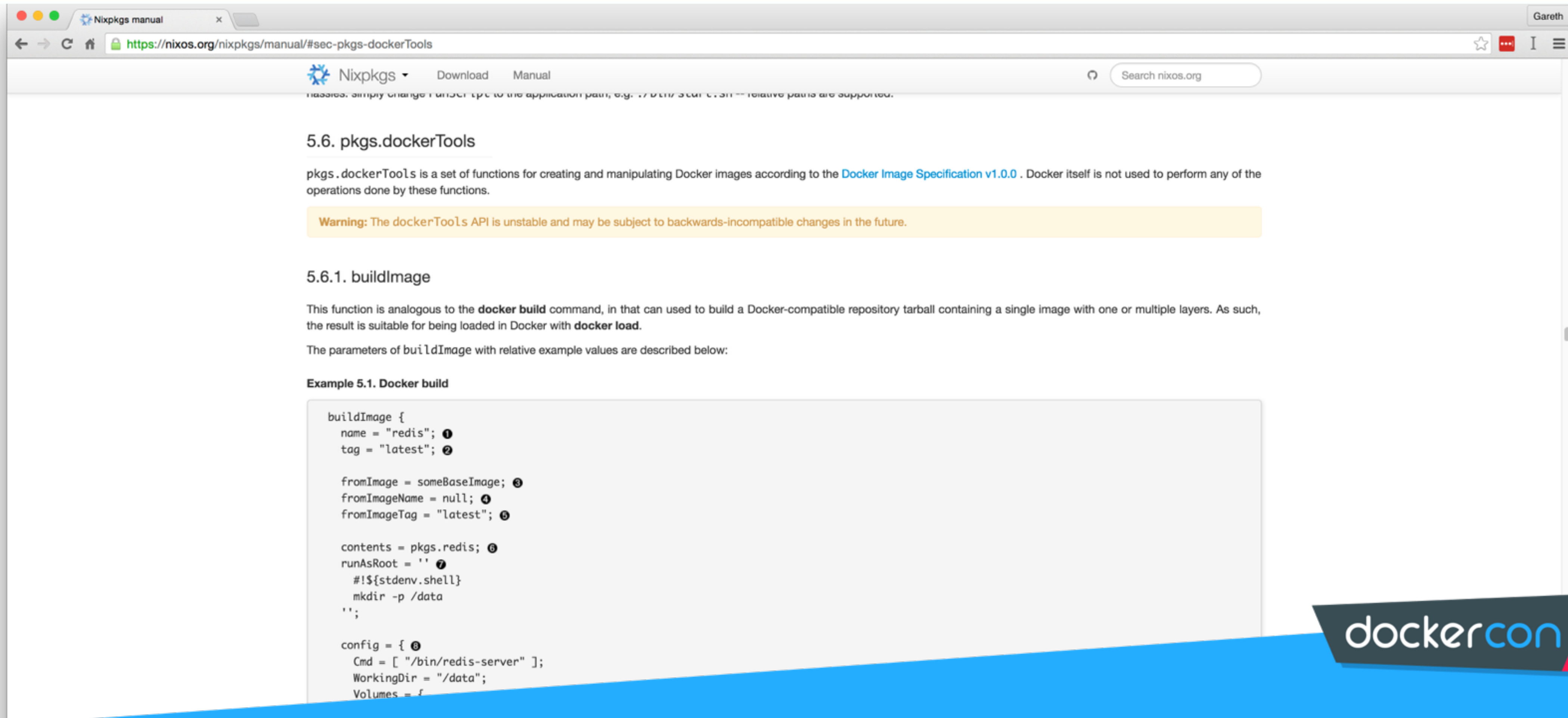
Build Process

The build process consists of the following three fundamental elements, which are combined into a final Docker image:

- sources
- S2I scripts

s2i example

```
$ s2i create <image name> <destination directory>  
$ s2i build <source location> <builder image> [<tag>] [flags]  
$ s2i rebuild <image name> [<new-tag-name>]  
$ s2i usage <builder image> [flags]  
  
$ s2i build ./sinatra-app openshift/ruby-20-centos7 ruby-app
```



The screenshot shows a web browser window displaying the Nixpkgs manual page for `pkg.dockerTools`. The browser's address bar shows the URL `https://nixos.org/nixpkgs/manual/#sec-pkgs-dockerTools`. The page header includes the Nixpkgs logo, navigation links for "Download" and "Manual", and a search bar. The main content area is titled "5.6. pkg.dockerTools" and contains a paragraph explaining that `pkg.dockerTools` is a set of functions for creating and manipulating Docker images according to the Docker Image Specification v1.0.0. A yellow warning box states: "Warning: The dockerTools API is unstable and may be subject to backwards-incompatible changes in the future." Below this, the sub-section "5.6.1. buildImage" is introduced, followed by a description of the function and its parameters. An example titled "Example 5.1. Docker build" shows a Nix expression for building a Redis image.

5.6. pkg.dockerTools

`pkg.dockerTools` is a set of functions for creating and manipulating Docker images according to the [Docker Image Specification v1.0.0](#). Docker itself is not used to perform any of the operations done by these functions.

Warning: The `dockerTools` API is unstable and may be subject to backwards-incompatible changes in the future.

5.6.1. buildImage

This function is analogous to the `docker build` command, in that can used to build a Docker-compatible repository tarball containing a single image with one or multiple layers. As such, the result is suitable for being loaded in Docker with `docker load`.

The parameters of `buildImage` with relative example values are described below:

Example 5.1. Docker build

```
buildImage {
  name = "redis"; ❶
  tag = "latest"; ❷

  fromImage = someBaseImage; ❸
  fromImageName = null; ❹
  fromImageTag = "latest"; ❺

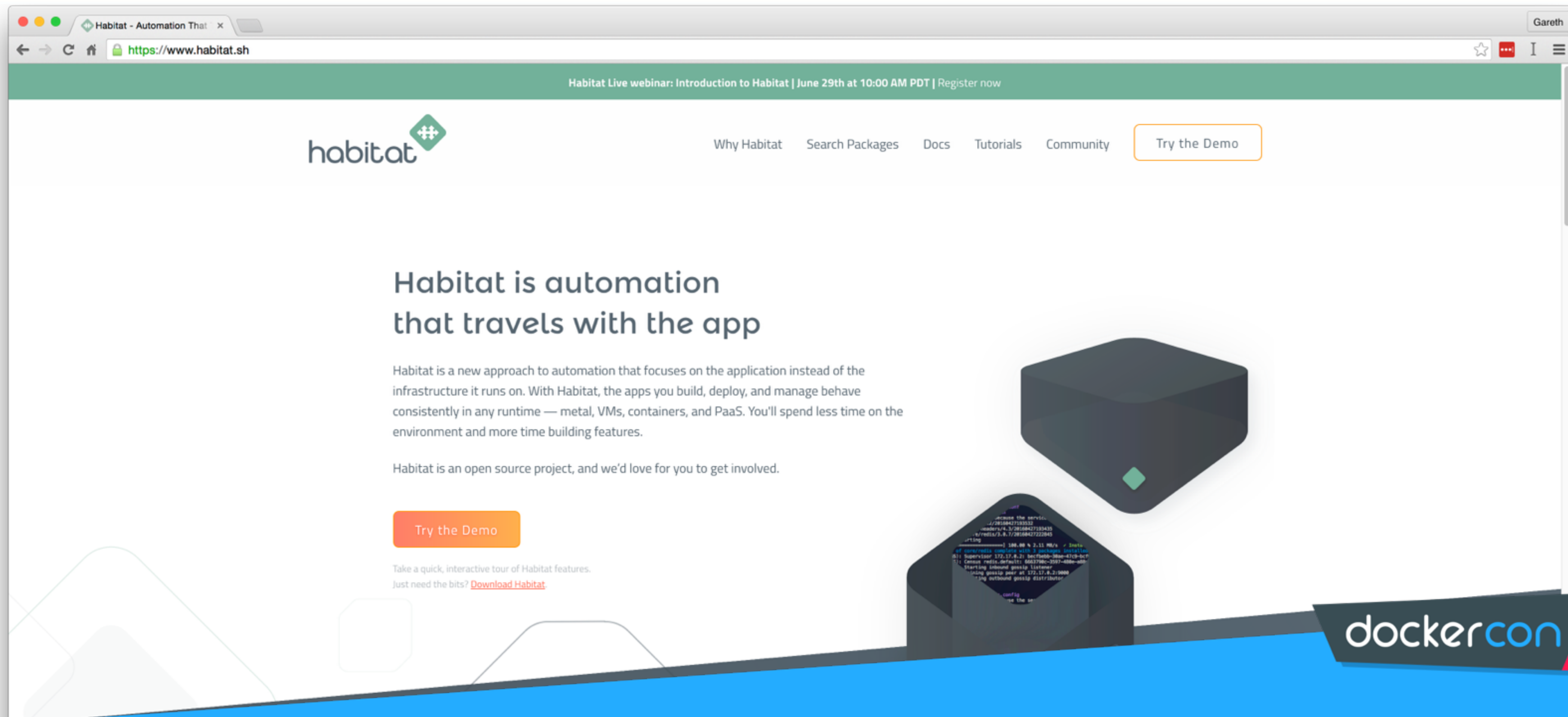
  contents = pkgs.redis; ❸
  runAsRoot = '' ❷
    #!${stdenv.shell}
    mkdir -p /data
  '';

  config = { ❸
    Cmd = [ "/bin/redis-server" ];
    WorkingDir = "/data";
    Volumes = {
```

Nix example

```
dockerTools.buildImage {  
  name = "redis";  
  runAsRoot = ''  
    #!${stdenv.shell}  
    ${dockerTools.shadowSetup}  
    groupadd -r redis  
    useradd -r -g redis -d /data -M redis  
    mkdir /data  
    chown redis:redis /data  
  '';  
}
```

Habitat



The image shows a browser window displaying the Habitat website. The browser's address bar shows the URL <https://www.habitat.sh>. The website has a green header with a navigation menu containing links for "Why Habitat", "Search Packages", "Docs", "Tutorials", "Community", and a "Try the Demo" button. The main content area features the Habitat logo, a headline "Habitat is automation that travels with the app", and a paragraph explaining that Habitat focuses on the application instead of the infrastructure. Below this is another "Try the Demo" button and a link to "Download Habitat". On the right side, there are 3D-style icons of a server rack and a terminal window showing code. A "dockercon 16" banner is visible in the bottom right corner of the browser window.

Habitat - Automation That x

<https://www.habitat.sh>

Habitat Live webinar: Introduction to Habitat | June 29th at 10:00 AM PDT | Register now

habitat

Why Habitat Search Packages Docs Tutorials Community Try the Demo

Habitat is automation that travels with the app

Habitat is a new approach to automation that focuses on the application instead of the infrastructure it runs on. With Habitat, the apps you build, deploy, and manage behave consistently in any runtime — metal, VMs, containers, and PaaS. You'll spend less time on the environment and more time building features.

Habitat is an open source project, and we'd love for you to get involved.

Try the Demo

Take a quick, interactive tour of Habitat features.
Just need the bits? [Download Habitat](#)

```
because the service
//2018042718335
waders/4.3/20180427183435
//2018/3.8.17/20180427222845
//1100
[100.00 % 2.15 MB/s / Sudo
01: Supervisor 172.17.0.2: 807900-80a-41c9-bcf
01: Command: /usr/bin/ls: /usr/bin/ls: /usr/bin/ls
Starting libcloud gossip listener
Using gossip peer at 172.17.0.2:8000
Using outbound gossip client/cluster
config
use the se
```

dockercon 16

Expand on Dockerfile

Rocker

The screenshot shows the GitHub repository page for 'grammarly/rocker'. The browser address bar shows the URL 'https://github.com/grammarly/rocker/'. The repository name is 'grammarly / rocker'. The page features a navigation bar with 'Pull requests', 'Issues', and 'Gist' links. Below the repository name, there are statistics for 'Watch' (48), 'Star' (292), and 'Fork' (12). The main content area displays the repository description: 'Rocker breaks the limits of Dockerfile.' Below this, there are statistics for '521 commits', '35 branches', '13 releases', and '7 contributors'. A navigation bar includes 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. The commit history shows a list of commits with their authors, descriptions, and dates.

grammarly/rocker: Rocker

GitHub, Inc. [US] https://github.com/grammarly/rocker/

This repository Search Pull requests Issues Gist

grammarly / rocker Watch 48 Star 292 Fork 12

Code Issues 4 Pull requests 0 Wiki Pulse Graphs

Rocker breaks the limits of Dockerfile.

521 commits 35 branches 13 releases 7 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

diunko update readme: remove not-supported note for ADD <url> Latest commit c2c56c6 15 days ago

Rockerfile.tmLanguage	fix #29 highlight INCLUDE in Rockerfile.tmLanguage	8 months ago
example	wipe history, make initial commit	9 months ago
rsync	wipe history, make initial commit	9 months ago
src	Close http.Response.Body in dockerclient to avoid	23 days ago
test	tests: use idiomatic time.Second	20 days ago
vendor	Update go-dockerclient to fetch upstream changes related to	
.dockerignore		

Rocker adds some crucial features that are missing from Dockerfile while keeping Docker's original design

Rockerfile example

```
FROM ubuntu:16.04
MAINTAINER Gareth Rushgrove "gareth@puppet.com"

ENV PUPPET_AGENT_VERSION="1.5.0" UBUNTU_CODENAME="xenial" PATH=/

LABEL com.puppet.version="0.1.0" com.puppet.dockerfile="/Dockerfile"

MOUNT /opt/puppetlabs /etc/puppetlabs /root/.gem

RUN apt-get update && \
    apt-get install
```

Includes new instructions

```
FROM ubuntu:16.04
MAINTAINER Gareth Rushgrove "gareth@puppet.com"

ENV PUPPET_AGENT_VERSION="1.5.0" UBUNTU_CODENAME="xenial" PATH=/

LABEL com.puppet.version="0.1.0" com.puppet.dockerfile="/Dockerfile"

MOUNT /opt/puppetlabs /etc/puppetlabs /root/.gem

RUN apt-get update && \
    apt-get install
```

More new instructions

```
rm -rf /var/lib/apt/lists/*
```

```
EXPOSE 80
```

```
CMD ["nginx"]
```

```
COPY Rockerfile /Dockerfile
```

```
TAG puppet/puppet-rocker-example
```

Dockramp

The screenshot shows a web browser window displaying the GitHub repository page for 'dockramp' by user 'jlhawn'. The browser's address bar shows the URL 'https://github.com/jlhawn/dockramp'. The repository page includes a search bar, navigation links for 'Pull requests', 'Issues', and 'Gist', and a header with the repository name and statistics: 15 watches, 217 stars, and 13 forks. Below the header, there are tabs for 'Code', 'Issues', 'Pull requests', 'Wiki', 'Pulse', and 'Graphs'. The main content area features a description 'A Client Driven Docker Image Builder' and a summary bar showing 19 commits, 1 branch, 1 release, and 2 contributors. A commit history table is visible, listing recent commits by user 'dmcgowan'.

Commit Message	Time Ago
Updated dependencies (#6)	a month ago
Initial working prototype	11 months ago
Updated dependencies (#6)	a month ago
[build] Move commit logic to separate file	11 months ago
Add a build process for Dockramp binaries	11 months ago
Add a build process for Dockramp binaries	11 months ago
Initial commit	a year ago
Add a build process for Dockramp binaries	11 months ago
Add a truck ramp picture to README	11 months ago
Initial working prototype	11 months ago
Initial working prototype	11 months ago
Add a build process for Dockramp binaries	11 months ago

Dockerfile pre-processors

The screenshot shows a GitHub repository page for 'garethr/dockerfilepp'. The repository has 1 commit, 1 branch, 0 releases, and 1 contributor. The files listed are .gitignore, Dockerfile, README.md, and main.go, all committed 3 minutes ago. The README.md content is as follows:

A *very much* proof-of-concept showing how simple it is to build Dockerfile pre-processors.

In this case `dockerfilepp` is a trivial go application which takes a Dockerfile on stdin and simply replaces some pre-defined values. The idea would be to make Dockerfile declarative again, making multiple Dockerfiles doing the same thing easier to maintain.

The examples centre around Puppet, but this is for demonstration purposes only. You could imagine building a library of DSL extensions in a similar way, or extending...

Domain-specific extensions

```
FROM ubuntu:16.04
MAINTAINER Gareth Rushgrove "gareth@puppet.com"

ENV PUPPET_AGENT_VERSION="1.5.0" R10K_VERSION="2.2.2" \
    UBUNTU_C

PUPPET_INSTALL
PUPPET_COPY_PUPPETFILE
PUPPET_COPY_MANIFESTS
PUPPET_RUN

EXPOSE 80
```

Simple expansion

```
$ cat Dockerfile | dockerfilepp
FROM ubuntu:16.04
MAINTAINER Gareth Rushgrove "gareth@puppet.com"

ENV PUPPET_AGENT_VERSION="1.5.0" R10K_VERSION="2.2.2" UBUNTU_CODEM

RUN apt-get update && \
    apt-get install -y wget=1.17.1-1ubuntu1 && \
    wget https://apt.puppetlabs.com/puppetlabs-release-pc1-"$UBUNTU_CODEM" \
    dpkg -i puppetlabs-release-pc1-"$UBUNTU_CODEM" \
    rm puppetlabs-rele
```

The future

Speculation and things I'd like to see

Formal specification for Dockerfile

RUN, FROM, COPY, etc.
as first class API primitives

Opinionated workflow tooling around image build

Shared libraries and support for pre-processors

Complementary tools that
take an organizational
view of image building

Conclusions

If all you take away is...

Dockerfile is a great starting point for many use cases

But we will need better tools for
managing many Dockerfiles

And Dockerfile is just one
interface to building images

We'll need different types of tools for different use cases

Questions?

And thanks for listening

Thank you!

