Docker

References

- https://docs.docker.com/engine/reference/builder/
- https://docs.docker.com/get-started/part2/#define-a-container-with-a-dockerfile
- https://en.wikipedia.org/wiki/Separation_of_concerns
- https://hub.docker.com/explore/
- https://store.docker.com
- https://github.com/wsargent/docker-cheat-sheet
- https://askubuntu.com/questions/334994/which-one-is-better-using-or-to-execute-multiple-commands-in-one-line
- https://pypi.python.org/pypi/virtualenv

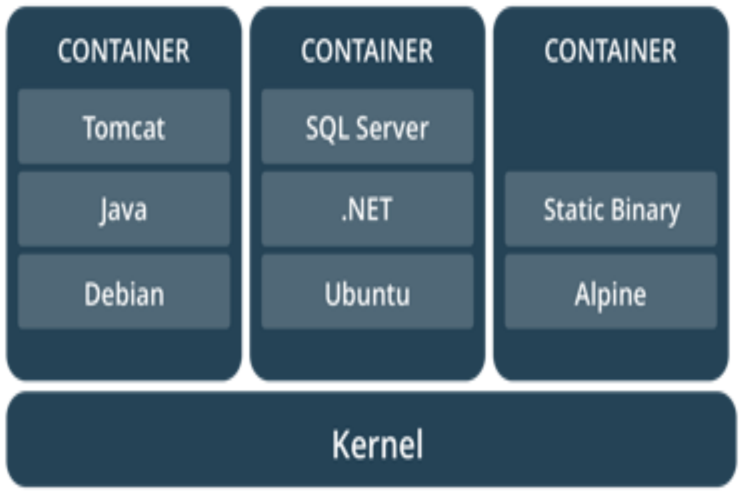
Docker

- Let's review what a container is
- A container makes use of the host environments OS, while providing an isolated environment in which software can be run
- A container does not need a guest OS.

Docker

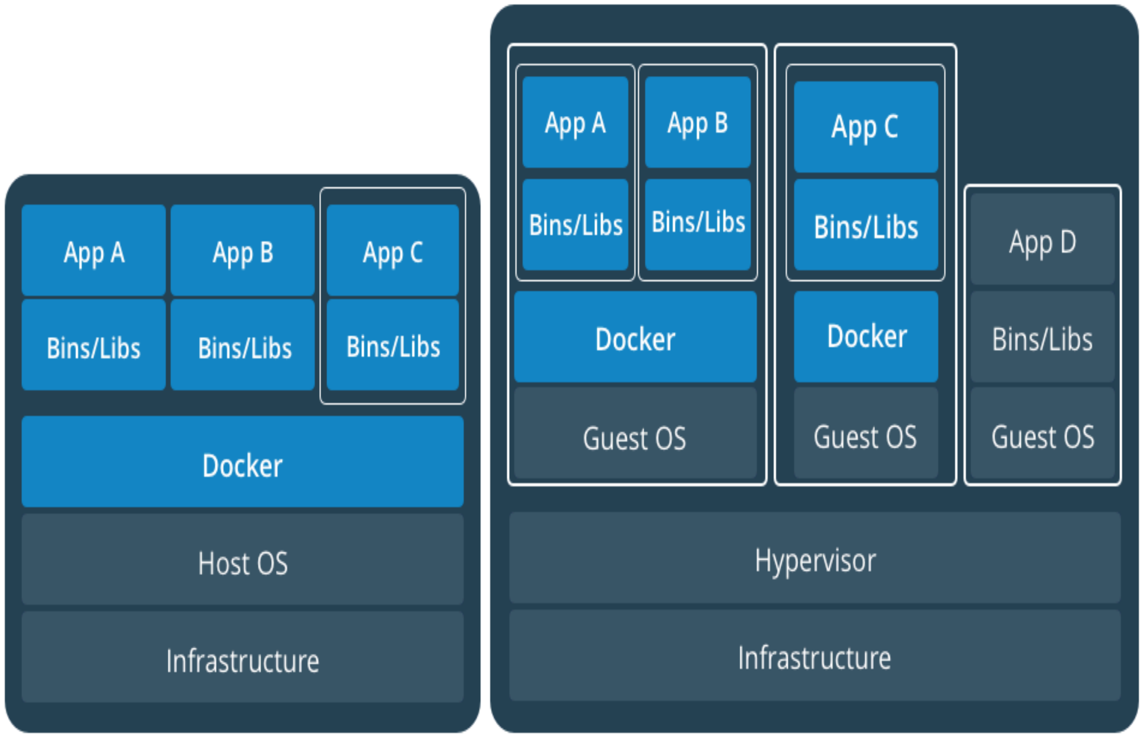
- Basically − a container only contains what it needs to run the applications embedded within it. This typically includes:
  - the applications
  - libraries and framework used by the applications
  - unlike a VM deployment, a container does NOT need an (guest) OS

Docker



From https://www.docker.com/what-container - each container contains the software it needs to run while utilizing the host environment's kernel

Docker



From https://www.docker.com/what-container#/virtual_machines: a containerized environment vs. a VM environment

Docker

- Docker's standard practices guidance recommends using a "separation of concerns" approach.

- From the Wiki:

  In computer science, **separation of concerns** (**SoC**) is a design principle for separating a computer program into distinct sections, such that each section addresses a separate concern

Docker

- An informal way to look at SoC with respect to Docker is:

  instead of using one container with all of your server-side components

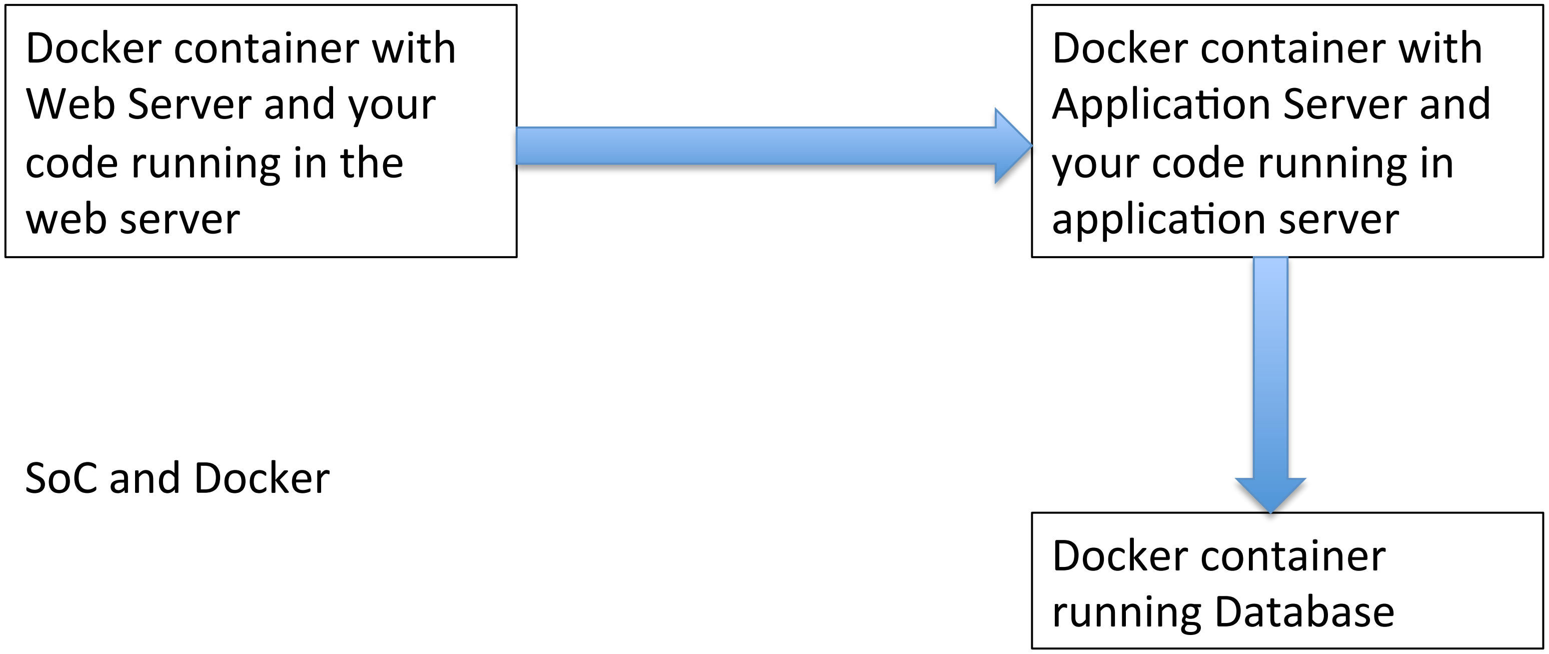
  several several containers, where each containers contains one component

Docker

- For example, if you have a web server running code, an application server running code, and a DB the application server accesses

  use 3 containers: one for the web server, one for the application server , and one for the DB

Docker SoC


Docker container with Web Server and your code running in the web server
Docker container with Application Server and your code running in application server
Docker container running Database

SoC and Docker

Docker and Microservices

- Using a Microservices approach (e.g. consider Microservices to be a finer-grained approach to Soc), we would break our Web Server, Application Server, DB into even more containers.

- We will put Microservices together with Docker later on

Dockerfile

- A Docker file specifies the environment to setup and configure inside your container.

- Docker guidelines state a container and its contents should be ephemeral − meaning everything in it is transient − not long lived.

Dockerfile

- Instead of saving the state of the container, you stop it and destroy it − and build a new one when you need it

  using as minimum as possible a set of setup and configuration parameters

Dockerfile

- Standard Docker practices suggest putting each Docker file in a different directory.
- If you Docker image needs external files that are not downloaded as part of the Docker setup, you can place these files in this directory.

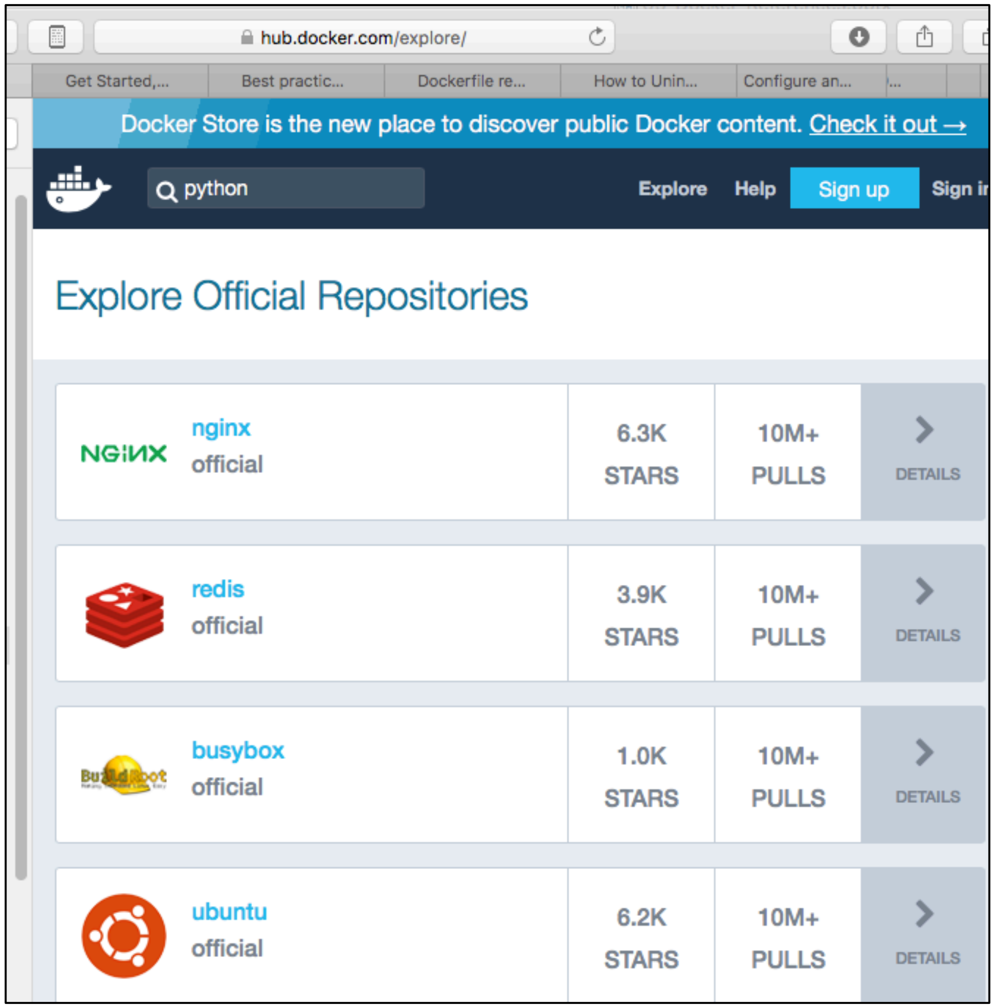
Dockerfile

- If you need to place a file in that Docker directory that you do not want in the Docker image, you can use a .dockerignore file

Docker Hub

- Like Vagrant, Docker has a repository of Docker images you can use.
- The following slides show Docker Hub

Docker Hub



hub.docker.com/explore/

Get Started,... | Best practic... | Dockerfile re... | How to Unin... | Configure an... | ...

Docker Store is the new place to discover public Docker content. Check it out →

python

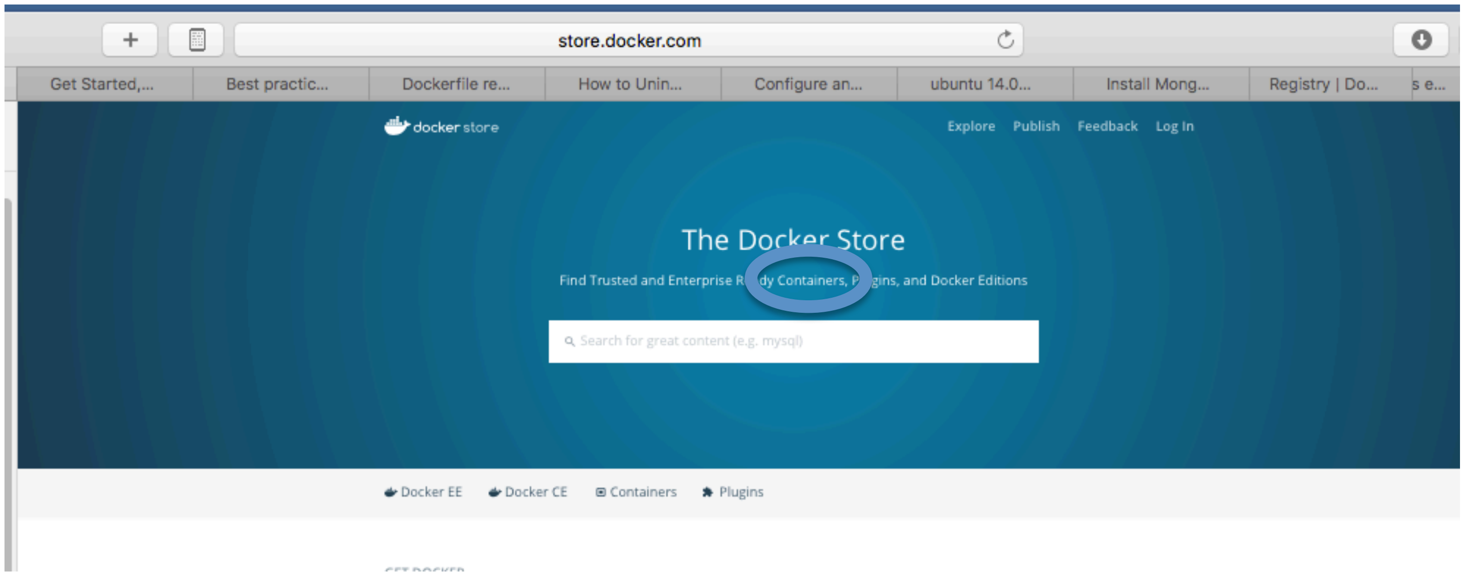
Explore Help Sign up Sign in

Explore Official Repositories

nginx official	6.3K STARS	10M+ PULLS	DETAILS
redis official	3.9K STARS	10M+ PULLS	DETAILS
busybox official	1.0K STARS	10M+ PULLS	DETAILS
ubuntu official	6.2K STARS	10M+ PULLS	DETAILS

https://hub.docker.com/explore/

Docker Store



The Docker Store

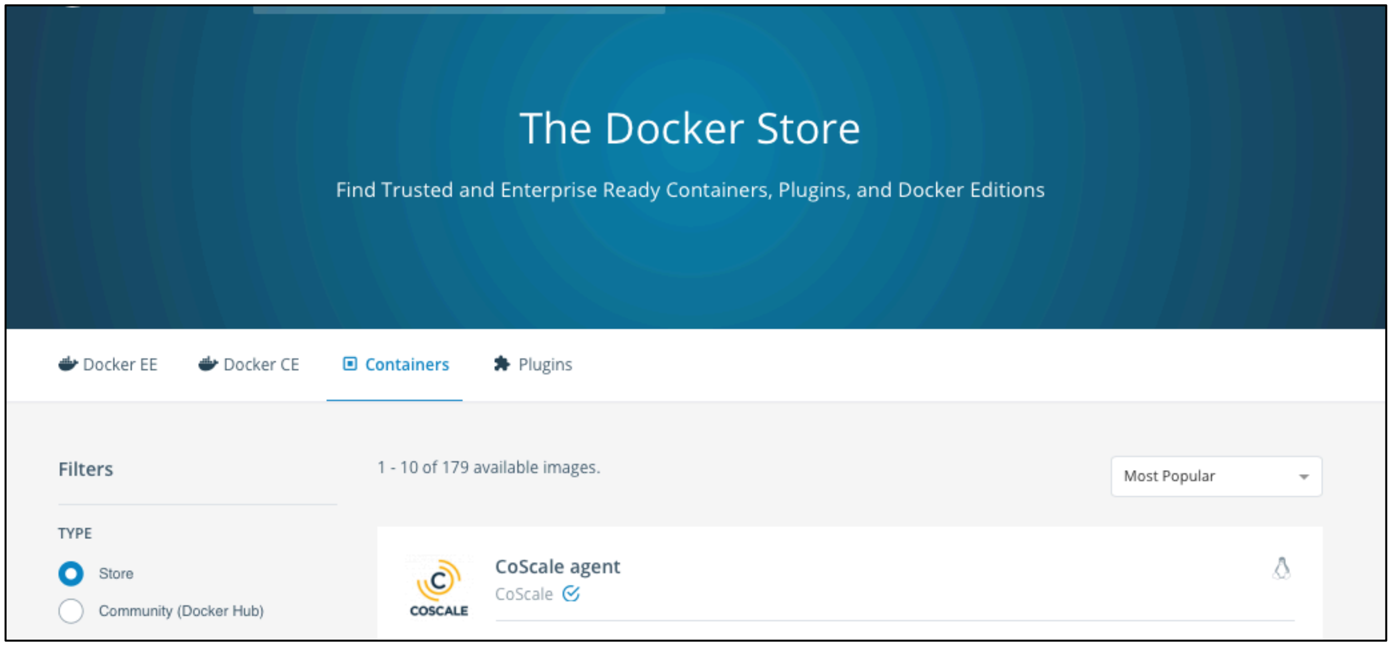
Find Trusted and Enterprise Ready Containers, Plugins, and Docker Editions

Search for great content (e.g. mysql)

https://store.docker.com

lets take the Containers link

Docker Store: Containers



The Docker Store

Find Trusted and Enterprise Ready Containers, Plugins, and Docker Editions

Filters

TYPE

[✓ Store] Community (Docker Hub)

1 - 10 of 179 available images.

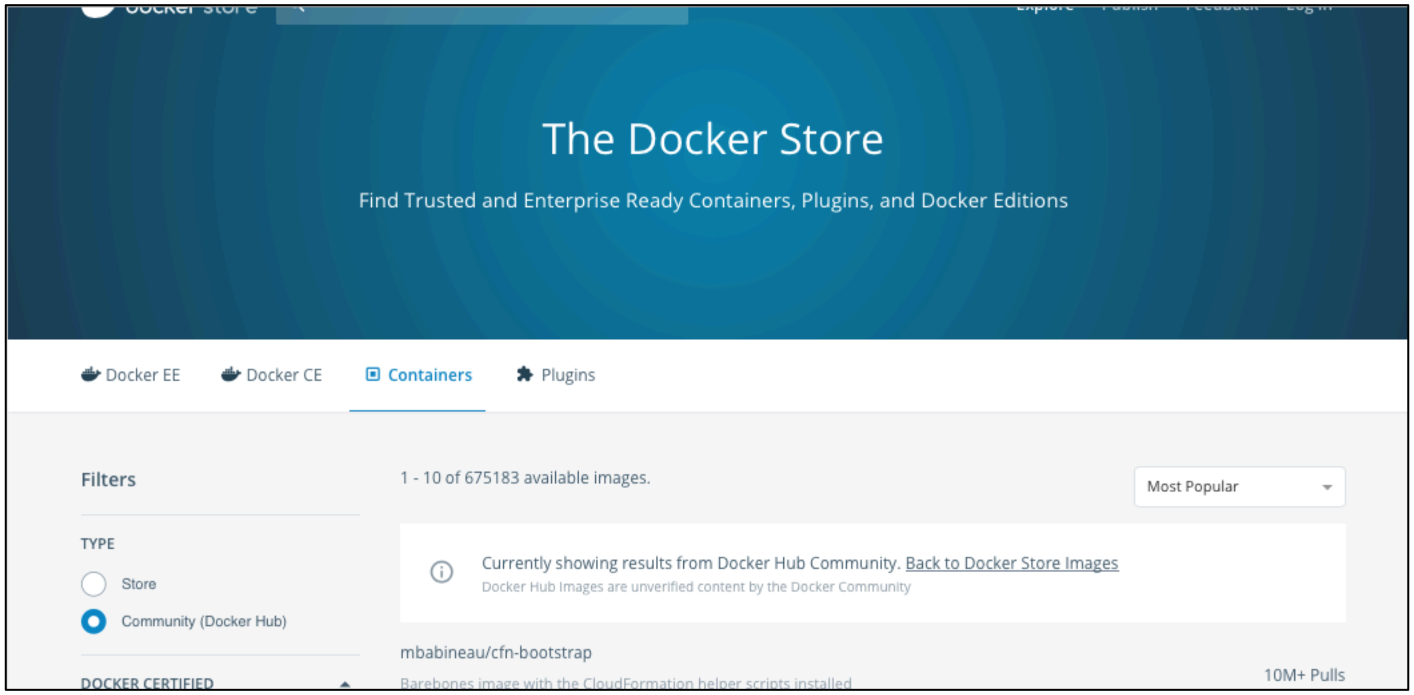
Most Popular

CoScale agent
CoScale

https://store.docker.com/search?q=&source=verified&type=image

let's select "Community (Docker Hub)"

Docker Store: Docker Hub



1 - 10 of 675183 available images.

Currently showing results from Docker Hub Community. Back to Docker Store Images
Docker Hub Images are unverified content by the Docker Community

mbabineau/cfn-bootstrap
Barebones image with the CloudFormation helper scripts installed
10M+ Pulls

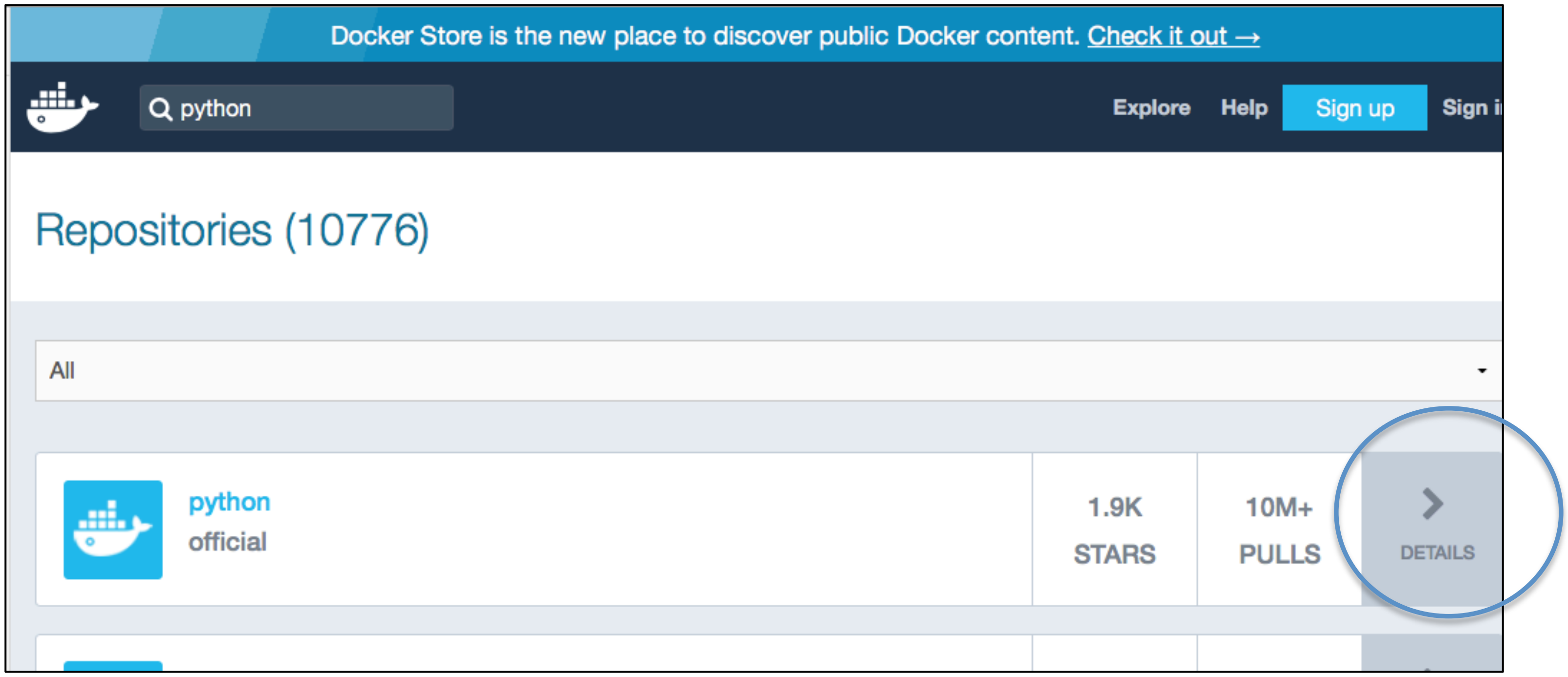
https://store.docker.com/search?q=&source=community&type=image

This is essentially another view of Docker Hub

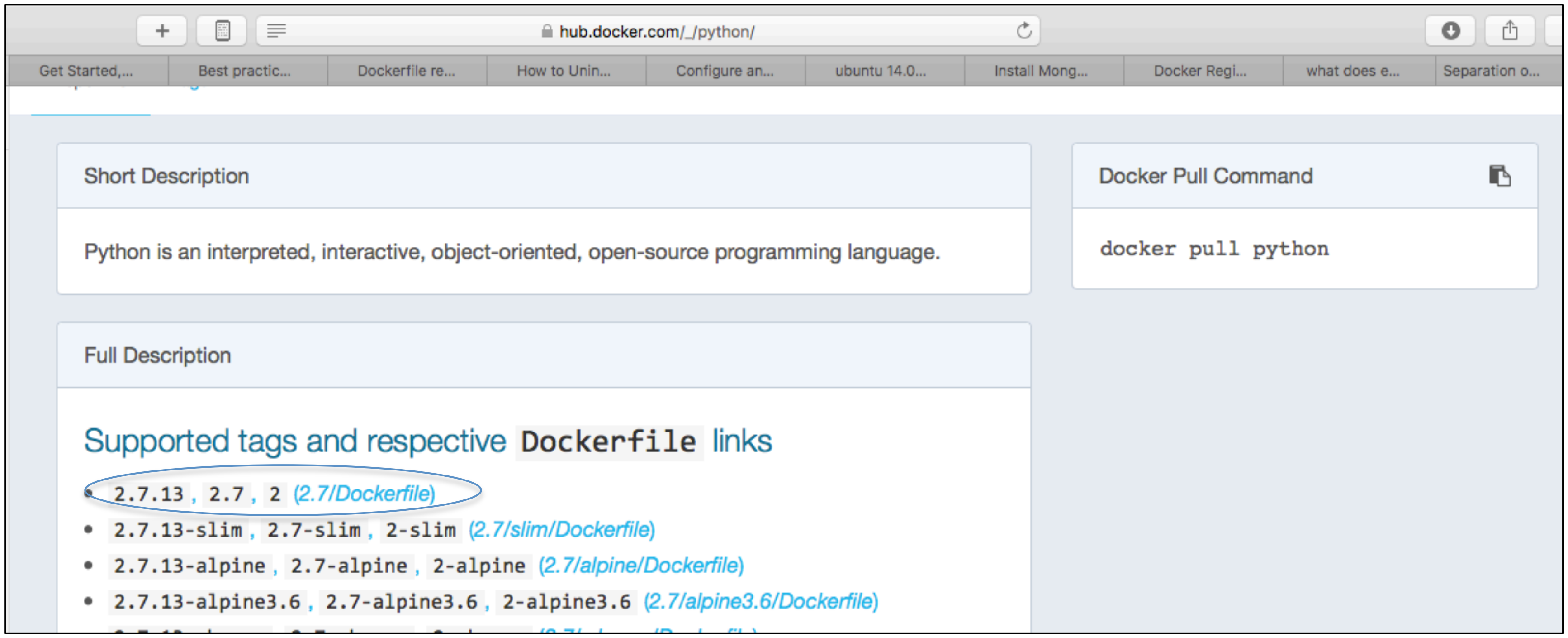
Docker Example py01

- In this example, we will create a Docker container with Python in it.
- Let's go back to hub.docker.com and search for Python

Docker Example py01



Docker Example py01



Short Description

Python is an interpreted, interactive, object-oriented, open-source programming language.

Docker Pull Command

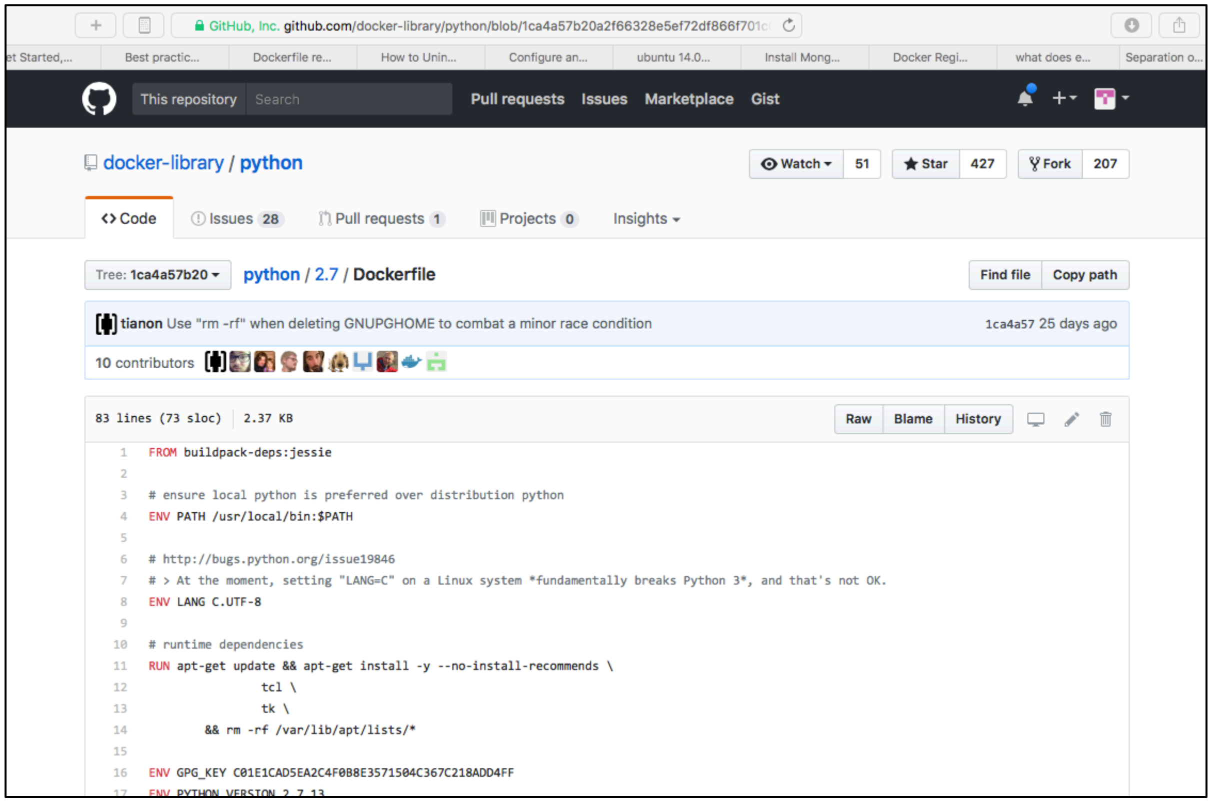
```
docker pull python
```

Full Description

Supported tags and respective `Dockerfile` links

- `2.7.13`, `2.7`, `2` (2.7/Dockerfile)
- `2.7.13-slim`, `2.7-slim`, `2-slim` (2.7/slim/Dockerfile)
- `2.7.13-alpine`, `2.7-alpine`, `2-alpine` (2.7/alpine/Dockerfile)
- `2.7.13-alpine3.6`, `2.7-alpine3.6`, `2-alpine3.6` (2.7/alpine3.6/Dockerfile)

Docker Example py01



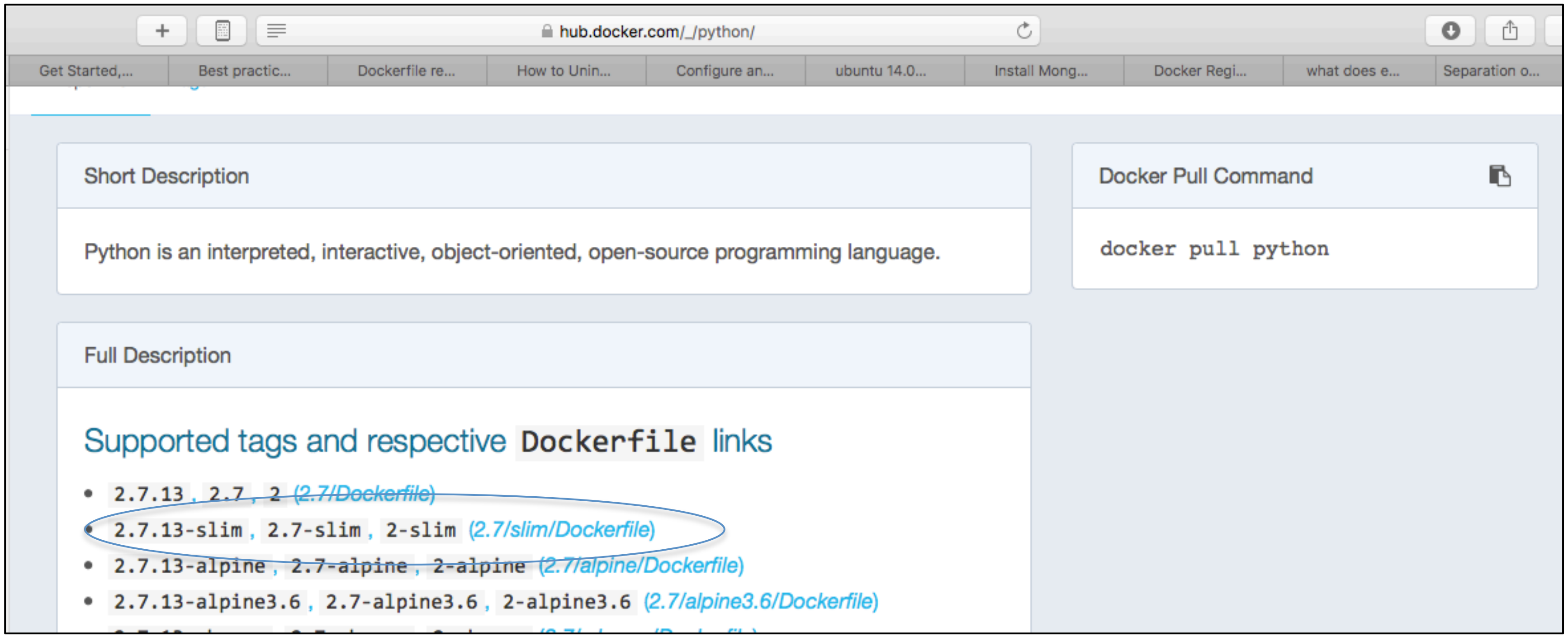
GitHub, Inc. github.com/docker-library/python/blob/1ca4a57b20a2f66328e5ef72df866f701c
et Started,... Best practic... Dockerfile re... How to Unin... Configure an... ubuntu 14.0... Install Mong... Docker Regi... what does e... Separation o...
This repository Search Pull requests Issues Marketplace Gist
docker-library / python
Watch 51 Star 427 Fork 207
Code Issues 28 Pull requests 1 Projects 0 Insights
Tree: 1ca4a57b20 python / 2.7 / Dockerfile
Find file Copy path
10 contributors
83 lines (73 sloc) 2.37 KB
Raw Blame History

tianon Use "rm -rf" when deleting GNUPGHOME to combat a minor race condition 1ca4a57 25 days ago

```
FROM buildpack-deps:jessie

# ensure local python is preferred over distribution python
ENV PATH /usr/local/bin:$PATH

# http://bugs.python.org/issue19846
# > At the moment, setting "LANG=C" on a Linux system *fundamentally breaks Python 3*, and that's not OK.
ENV LANG C.UTF-8

# runtime dependencies
RUN apt-get update && apt-get install -y --no-install-recommends \
		tcl \
		tk \
	&& rm -rf /var/lib/apt/lists/*

ENV GPG_KEY C01E1CAD5EA2C4F0B8E3571504C367C218ADD4FF
ENV PYTHON_VERSION 2.7.13
```

taking the link took us to a Dockerfile in GitHub

Docker Example py01



Short Description

Python is an interpreted, interactive, object-oriented, open-source programming language.

Docker Pull Command

```
docker pull python
```

Full Description

Supported tags and respective `Dockerfile` links

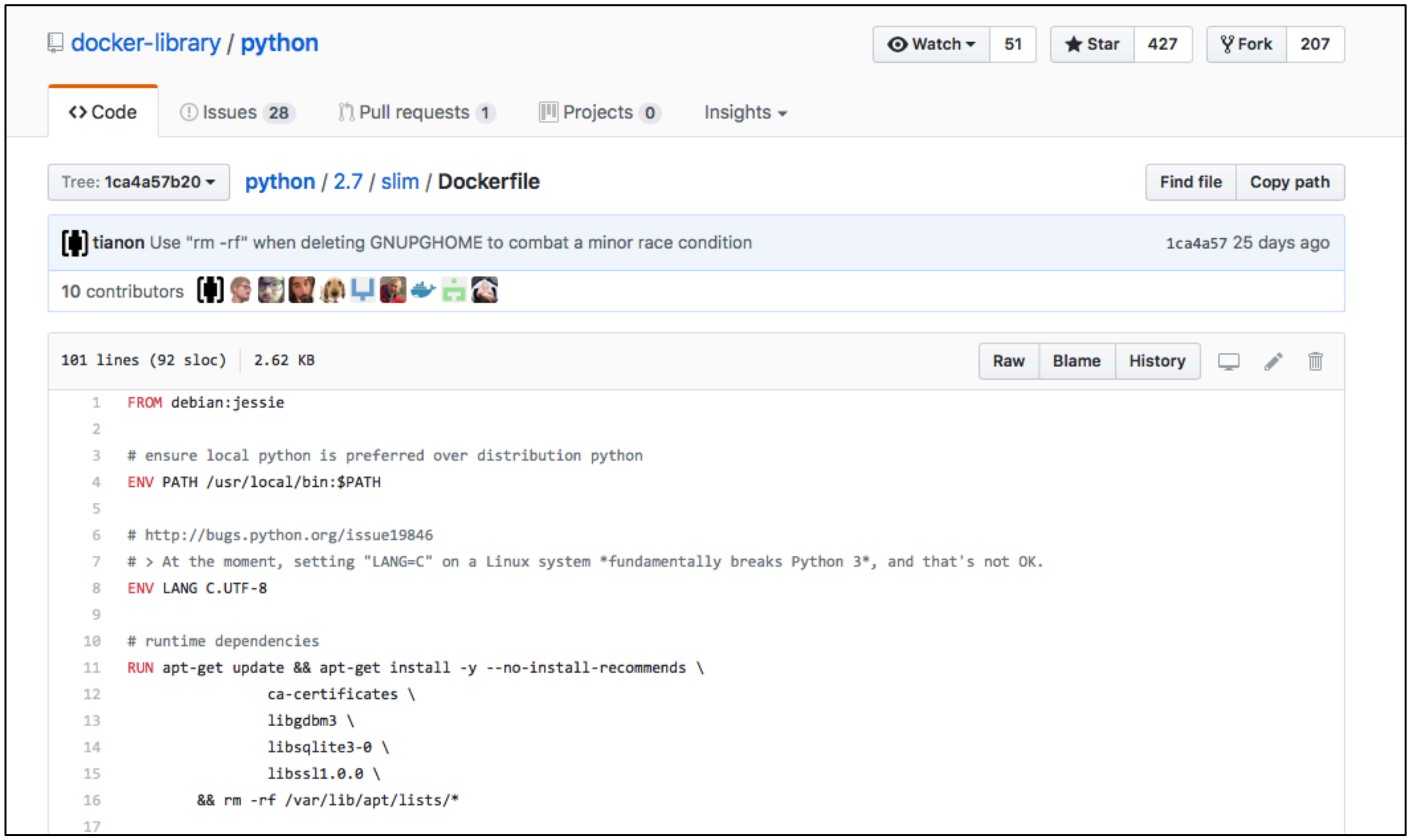
- `2.7.13`, `2.7`, `2` (2.7/Dockerfile)
- `2.7.13-slim`, `2.7-slim`, `2-slim` (2.7/slim/Dockerfile)
- `2.7.13-alpine`, `2.7-alpine`, `2-alpine` (2.7/alpine/Dockerfile)
- `2.7.13-alpine3.6`, `2.7-alpine3.6`, `2-alpine3.6` (2.7/alpine3.6/Dockerfile)

Let's look at the Dockerfile for Python slim – a smaller Docker image for Python.

The previous Dockerfile for a Python would produce a much bigger image.

In general, it is a good idea to use a smaller Docker image and add whatever else you need into the image

Docker Example py01



python / 2.7 / slim / Dockerfile

tianon Use "rm -rf" when deleting GNUPGHOME to combat a minor race condition — 1ca4a57 25 days ago

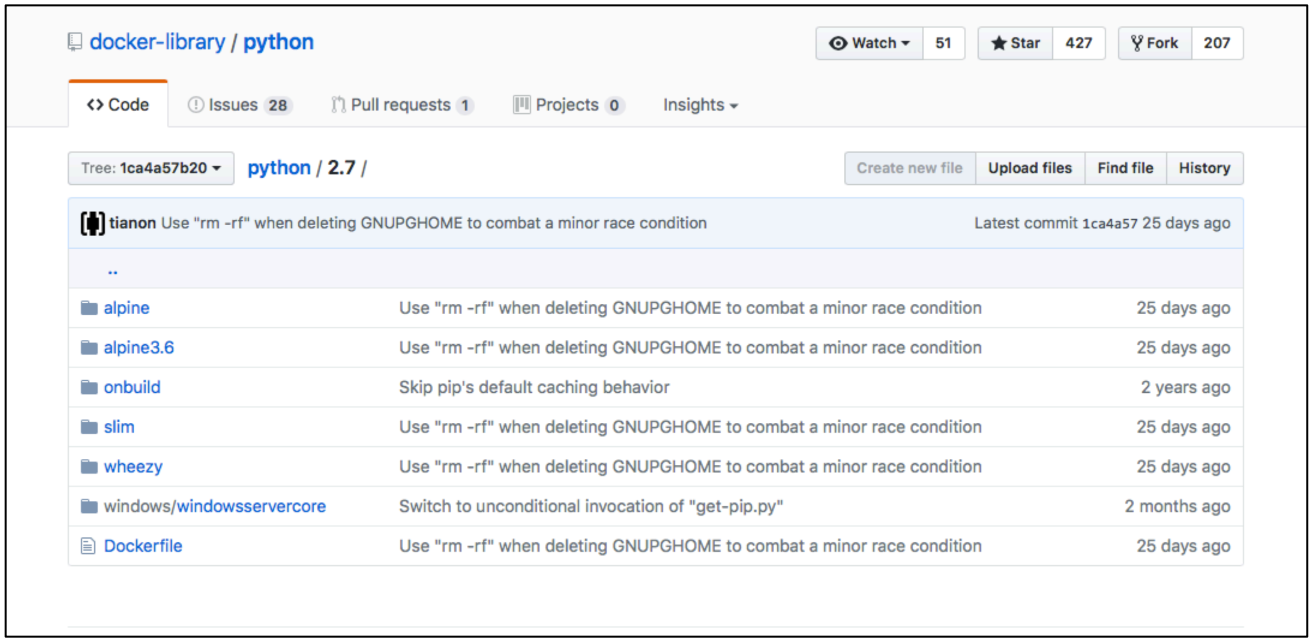
10 contributors

101 lines (92 sloc) | 2.62 KB

```
FROM debian:jessie

# ensure local python is preferred over distribution python
ENV PATH /usr/local/bin:$PATH

# http://bugs.python.org/issue19846
# > At the moment, setting "LANG=C" on a Linux system *fundamentally breaks Python 3*, and that's not OK.
ENV LANG C.UTF-8

# runtime dependencies
RUN apt-get update && apt-get install -y --no-install-recommends \
                ca-certificates \
                libgdbm3 \
                libsqlite3-0 \
                libssl1.0.0 \
        && rm -rf /var/lib/apt/lists/*

```

Again, we are at GitHub

Docker Example py01



docker-library / python

Watch 51 | Star 427 | Fork 207

<> Code | Issues 28 | Pull requests 1 | Projects 0 | Insights

Tree: 1ca4a57b20 python / 2.7 /

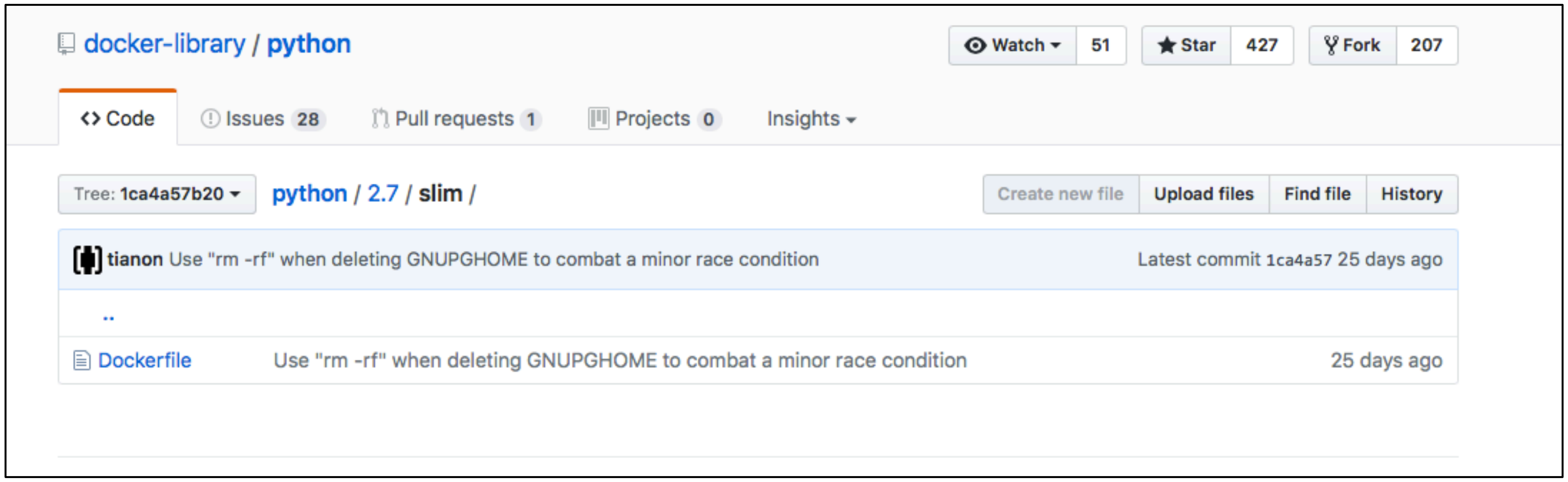
Create new file | Upload files | Find file | History

tianon Use "rm -rf" when deleting GNUPGHOME to combat a minor race condition — Latest commit 1ca4a57 25 days ago

..		
alpine	Use "rm -rf" when deleting GNUPGHOME to combat a minor race condition	25 days ago
alpine3.6	Use "rm -rf" when deleting GNUPGHOME to combat a minor race condition	25 days ago
onbuild	Skip pip's default caching behavior	2 years ago
slim	Use "rm -rf" when deleting GNUPGHOME to combat a minor race condition	25 days ago
wheezy	Use "rm -rf" when deleting GNUPGHOME to combat a minor race condition	25 days ago
windows/windowsservercore	Switch to unconditional invocation of "get-pip.py"	2 months ago
Dockerfile	Use "rm -rf" when deleting GNUPGHOME to combat a minor race condition	25 days ago

Here is the set of Docker Python images at GitHub

Docker Example py01



python / 2.7 / slim /

tianon Use "rm -rf" when deleting GNUPGHOME to combat a minor race condition — Latest commit 1ca4a57 25 days ago

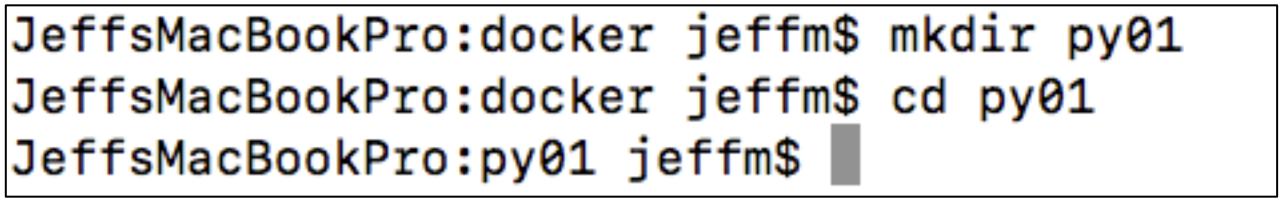
..		
Dockerfile	Use "rm -rf" when deleting GNUPGHOME to combat a minor race condition	25 days ago

Here is the Python 2.7 slim Dockerfile

Docker Example py01

In a terminal:

- create a directory named py01
- cd into py01

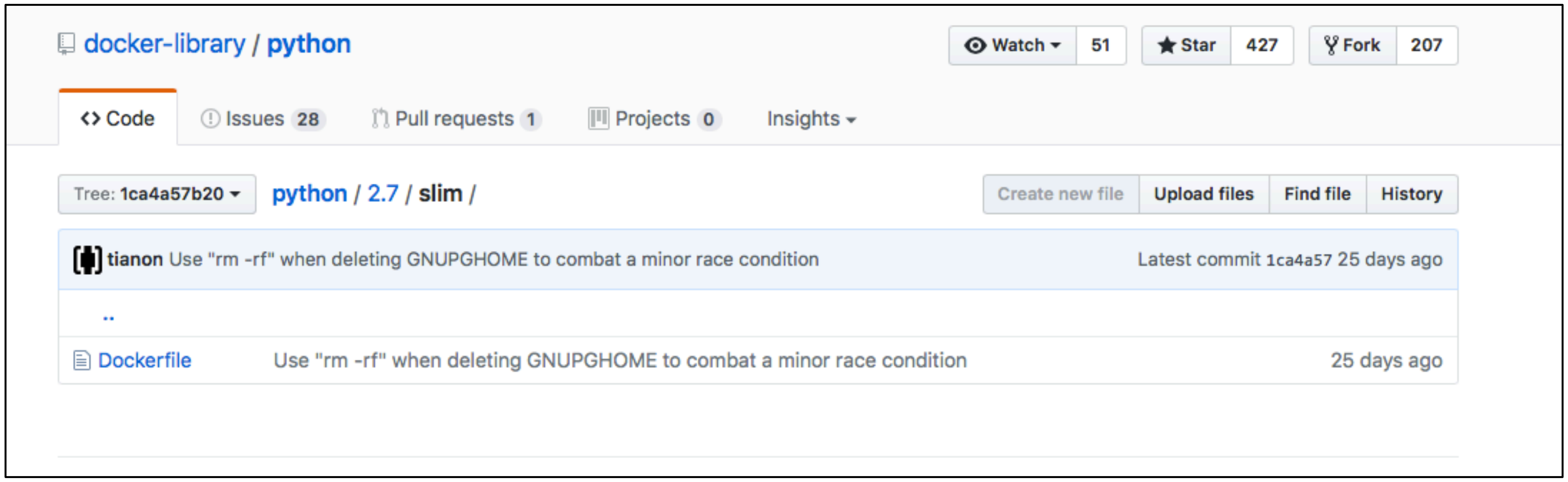


```
JeffsMacBookPro:docker jeffm$ mkdir py01
JeffsMacBookPro:docker jeffm$ cd py01
JeffsMacBookPro:py01 jeffm$ 
```

Docker Example py01

- Let's use **wget** to download the Dockerfile for Python 2.7 slim
- If you have not installed **wget** on your system yet, please do so now
- To do this we need the "raw" URL to the file.

Docker Example py01


docker-library / python

python / 2.7 / slim /

tianon Use "rm -rf" when deleting GNUPGHOME to combat a minor race condition — Latest commit 1ca4a57 25 days ago

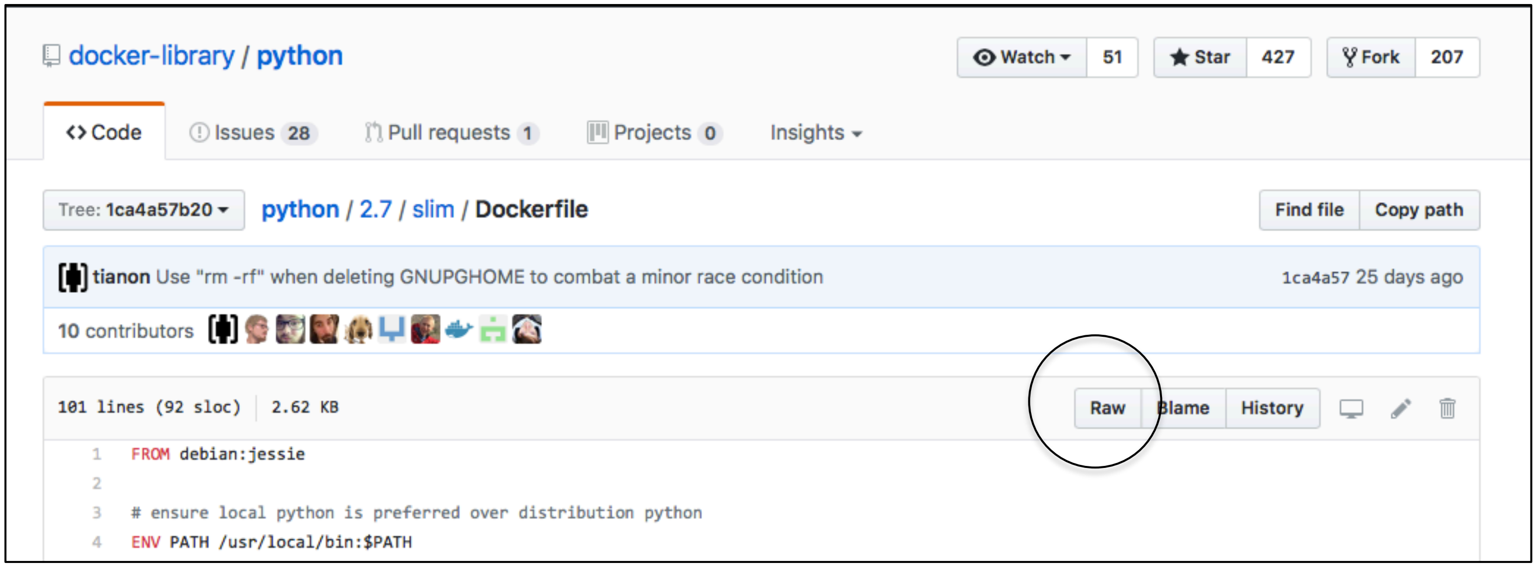
..

Dockerfile — Use "rm -rf" when deleting GNUPGHOME to combat a minor race condition — 25 days ago

To get the raw url to a file – Dockerfile for this example:

- click the file

Docker Example py01



python / 2.7 / slim / Dockerfile

tianon Use "rm -rf" when deleting GNUPGHOME to combat a minor race condition   1ca4a57 25 days ago

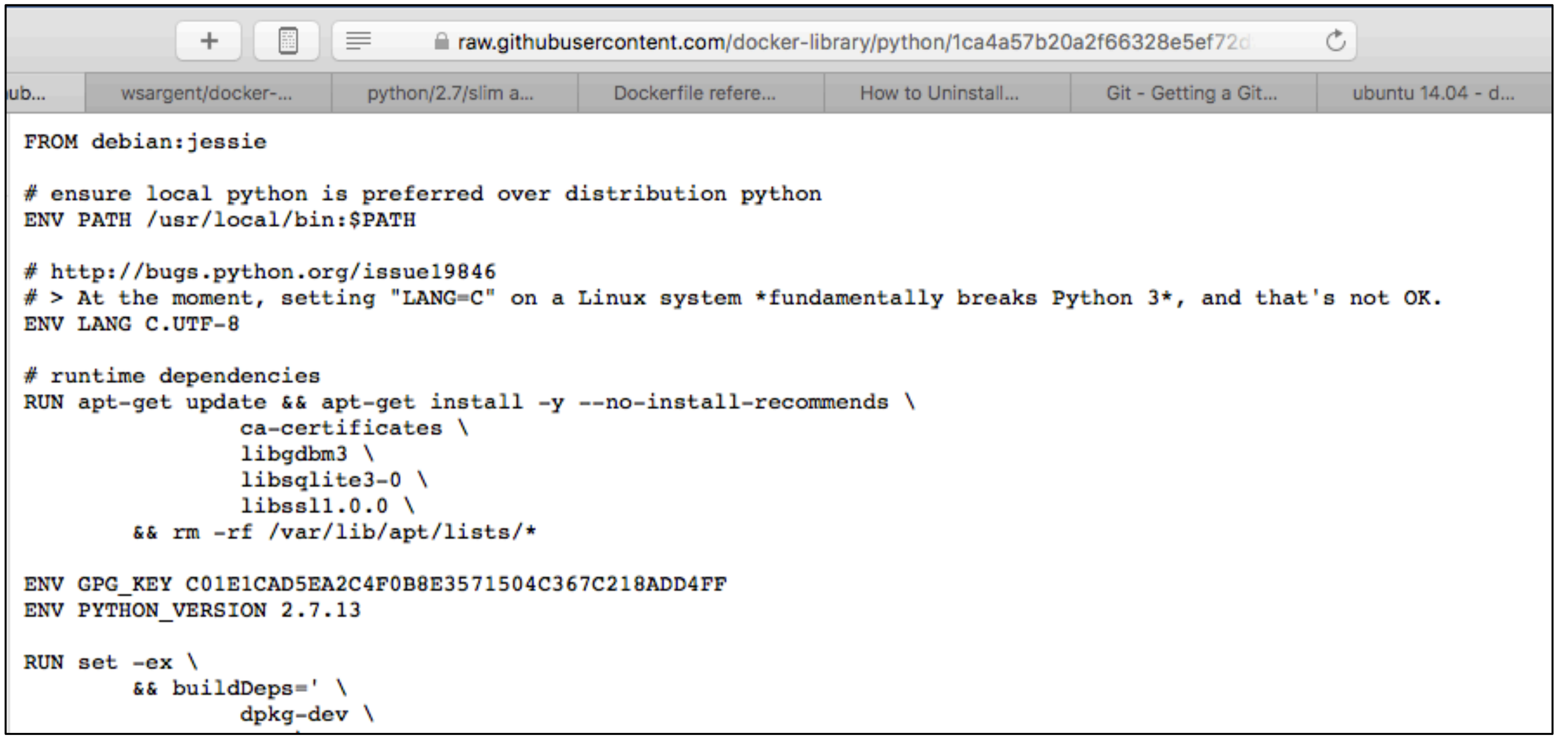
10 contributors

101 lines (92 sloc) | 2.62 KB

```
FROM debian:jessie

# ensure local python is preferred over distribution python
ENV PATH /usr/local/bin:$PATH
```

Click the Raw button

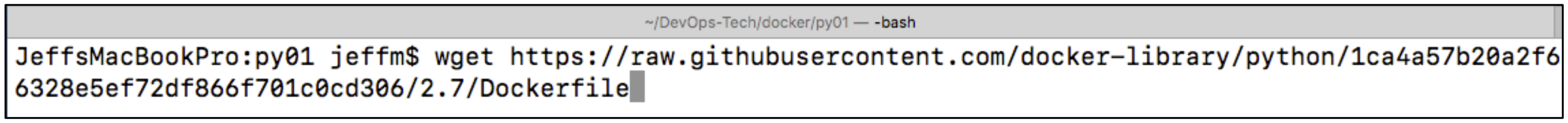
Docker Example py01



```
FROM debian:jessie

# ensure local python is preferred over distribution python
ENV PATH /usr/local/bin:$PATH

# http://bugs.python.org/issue19846
# > At the moment, setting "LANG=C" on a Linux system *fundamentally breaks Python 3*, and that's not OK.
ENV LANG C.UTF-8

# runtime dependencies
RUN apt-get update && apt-get install -y --no-install-recommends \
                ca-certificates \
                libgdbm3 \
                libsqlite3-0 \
                libssl1.0.0 \
        && rm -rf /var/lib/apt/lists/*

ENV GPG_KEY C01E1CAD5EA2C4F0B8E3571504C367C218ADD4FF
ENV PYTHON_VERSION 2.7.13

RUN set -ex \
        && buildDeps=' \
                dpkg-dev \
```

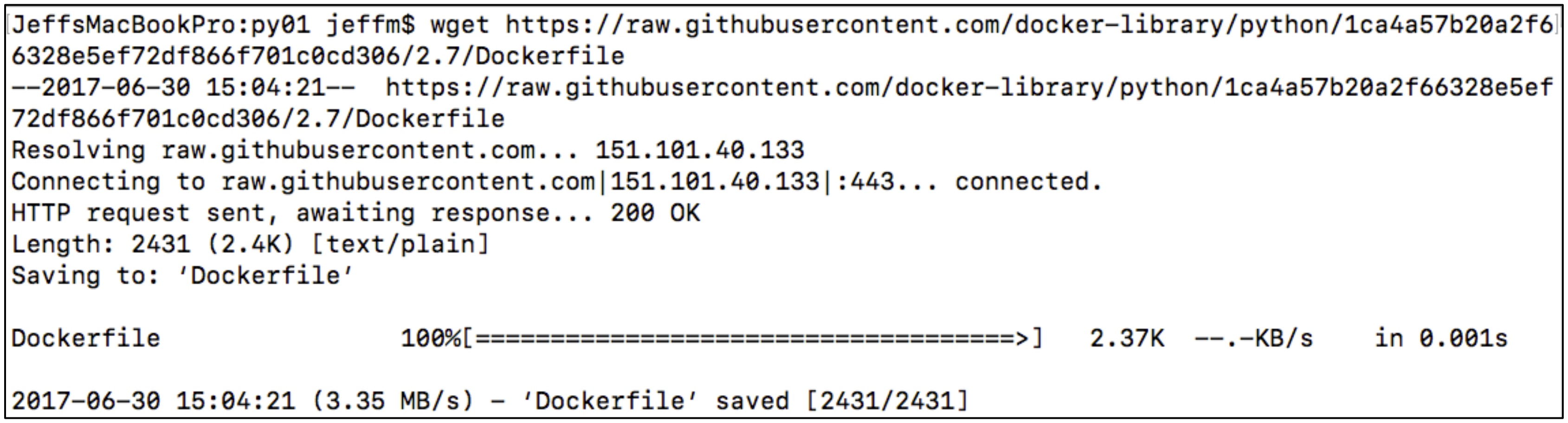
Copy the Url from the browser. We will use this URL on the command line with wget

Docker Example py01

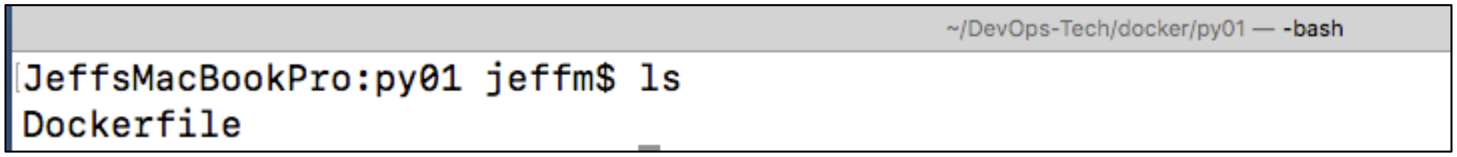


~/DevOps-Tech/docker/py01 — -bash

```
JeffsMacBookPro:py01 jeffm$ wget https://raw.githubusercontent.com/docker-library/python/1ca4a57b20a2f6
6328e5ef72df866f701c0cd306/2.7/Dockerfile
```



```
JeffsMacBookPro:py01 jeffm$ wget https://raw.githubusercontent.com/docker-library/python/1ca4a57b20a2f6
6328e5ef72df866f701c0cd306/2.7/Dockerfile
--2017-06-30 15:04:21--  https://raw.githubusercontent.com/docker-library/python/1ca4a57b20a2f66328e5ef
72df866f701c0cd306/2.7/Dockerfile
Resolving raw.githubusercontent.com... 151.101.40.133
Connecting to raw.githubusercontent.com|151.101.40.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2431 (2.4K) [text/plain]
Saving to: ‘Dockerfile’

Dockerfile                100%[=====================================>]   2.37K  --.-KB/s    in 0.001s

2017-06-30 15:04:21 (3.35 MB/s) - ‘Dockerfile’ saved [2431/2431]
```



~/DevOps-Tech/docker/py01 — -bash

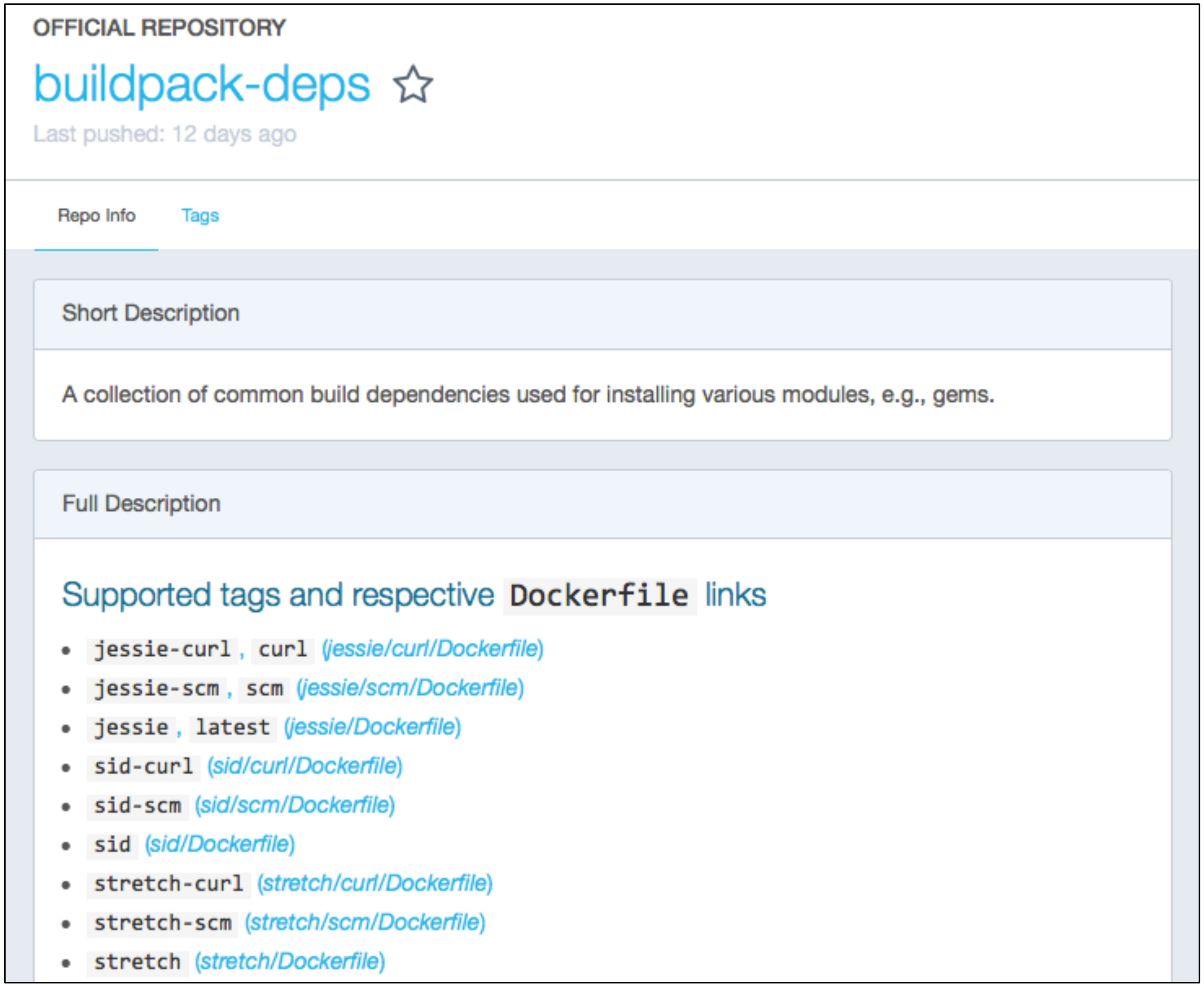
```
JeffsMacBookPro:py01 jeffm$ ls
Dockerfile
```

Docker Example py01

- Now that we have downloaded a Dockerfile for python/2.7/slim, let's take a look inside of it.
  - NOTE: we will NOT look at every line

Docker Example py01

```
FROM buildpack-deps:jessie

# A Dockerfile file must start with a FROM instruction
# The FROM instruction specifies the Base Image from which you are building.

# buildpack-deps refers to  location buildpack-deps in the Docker hub:
# https://hub.docker.com/_/buildpack-deps/

# is a comment line
```


OFFICIAL REPOSITORY

buildpack-deps

Last pushed: 12 days ago

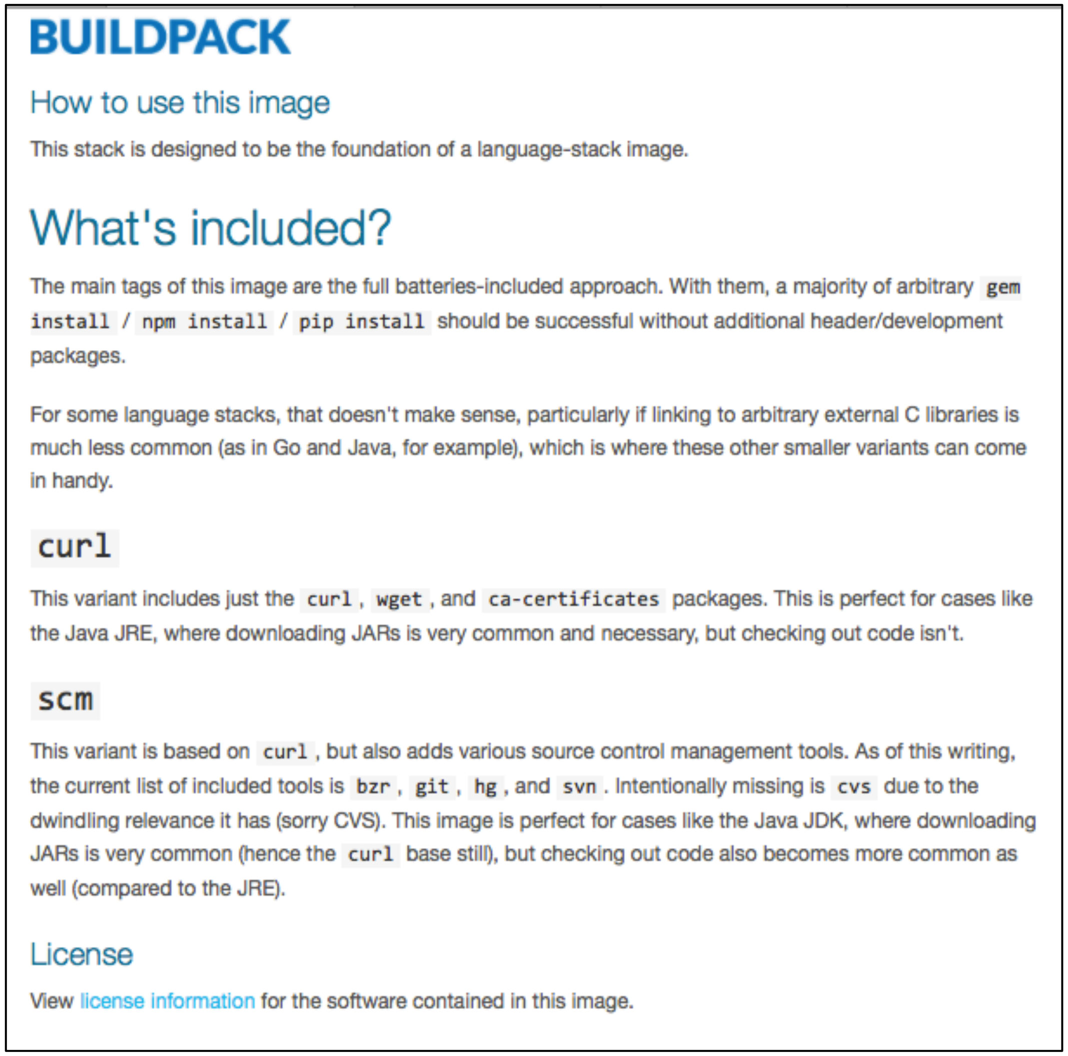
Short Description

A collection of common build dependencies used for installing various modules, e.g., gems.

Full Description

Supported tags and respective `Dockerfile` links

- `jessie-curl`, `curl` (jessie/curl/Dockerfile)
- `jessie-scm`, `scm` (jessie/scm/Dockerfile)
- `jessie`, `latest` (jessie/Dockerfile)
- `sid-curl` (sid/curl/Dockerfile)
- `sid-scm` (sid/scm/Dockerfile)
- `sid` (sid/Dockerfile)
- `stretch-curl` (stretch/curl/Dockerfile)
- `stretch-scm` (stretch/scm/Dockerfile)
- `stretch` (stretch/Dockerfile)



BUILDPACK

How to use this image

This stack is designed to be the foundation of a language-stack image.

What's included?

The main tags of this image are the full batteries-included approach. With them, a majority of arbitrary `gem install` / `npm install` / `pip install` should be successful without additional header/development packages.

For some language stacks, that doesn't make sense, particularly if linking to arbitrary external C libraries is much less common (as in Go and Java, for example), which is where these other smaller variants can come in handy.

`curl`

This variant includes just the `curl`, `wget`, and `ca-certificates` packages. This is perfect for cases like the Java JRE, where downloading JARs is very common and necessary, but checking out code isn't.

`scm`

This variant is based on `curl`, but also adds various source control management tools. As of this writing, the current list of included tools is `bzr`, `git`, `hg`, and `svn`. Intentionally missing is `cvs` due to the dwindling relevance it has (sorry CVS). This image is perfect for cases like the Java JDK, where downloading JARs is very common (hence the `curl` base still), but checking out code also becomes more common as well (compared to the JRE).

License

View license information for the software contained in this image.

continued:

https://hub.docker.com/_/buildpack-deps/

Docker Example py01

```
ENV PATH /usr/local/bin:$PATH
ENV LANG C.UTF-8

# ENV declares an environment variable that will reside within the Docker image

# There are 2 forms of ENV:

# ENV key value −
# in this example: environment variable LANG will be set to C.UTF-*
#
# The 2nd form of the ENV statement is: ENV key=value
# in this example the PATH environment variable puts /usr/local/bin at
# the front of the PATH in the Docker image

#Like in Bash, the value of an environment variable is accessed using:
# $variable-name or ${variable-name} − e.g. $PATH or ${PATH}
```

Docker Example py01

```
RUN apt-get update && apt-get install -y --no-install-recommends \
          tcl \
          tk \
     && rm -rf /var/lib/apt/lists/*
```

The RUN command executes the commands within the current Docker image and, updates the Docker image with the results created by executing the RUN commands

Run has two syntactical forms:

RUN command

RUN executable-to-run

Docker Example py01

```
RUN apt-get update && apt-get install -y --no-install-recommends \
            tcl \
            tk \
      && rm -rf /var/lib/apt/lists/*
```

The && shell operator − runs two commands − e.b. cmd-A && cmd-B

- cmd-B will be run if cmd-A was successful

In the Bash shell other multiple command constructs include:

```
A; B        #run B regardless of wheather A is successful
A || B      #run B if A fails
A &         #run A in the background
```

The '\' character is the line continuation character

Docker Example py01

```
RUN apt-get update && apt-get install -y --no-install-recommends \
            tcl \
            tk \
      && rm -rf /var/lib/apt/lists/*
```

In the example we are running 3 commands:

apt-get update

apt-get install −y −no-install-recommends tcl tk

rm −rf /var/;ib/apt/lists/*

Docker Example py01

```
RUN set -ex \
        && buildDeps=' \
                dpkg-dev \
                tcl-dev \
                tk-dev \
        ' \
        && apt-get update && apt-get install -y $buildDeps --no-install-recommends && rm -rf /var/lib/apt/lists/* \
        \
        && wget -O python.tar.xz "https://www.python.org/ftp/python/${PYTHON_VERSION%%[a-z]*}/
Python-$PYTHON_VERSION.tar.xz" \
```

set −ex e causes the script to exit if an error occurs, x tells the shell to print each line it executes with a "+"

buildDeps='…**'** sets a variable used elsewhere in the script

apt-get … updates the Docker image

wget … gets python

Docker Example py01

```
# install "virtualenv", since the vast majority of users of this image will want it
RUN pip install --no-cache-dir virtualenv

CMD ["python2"]
```

RUN pip install --no-cache-dir virtualenv − uses the Python package manager (pip) to get the.

virtualenv is a tool to create isolated Python environments (on the same machine or in the same VM or container). This is useful if you need to work with Python projects that use different versions of the same libraries.

Docker Example py01

```
# install "virtualenv", since the vast majority of users of this image will want it
RUN pip install --no-cache-dir virtualenv

CMD ["python2"]
```

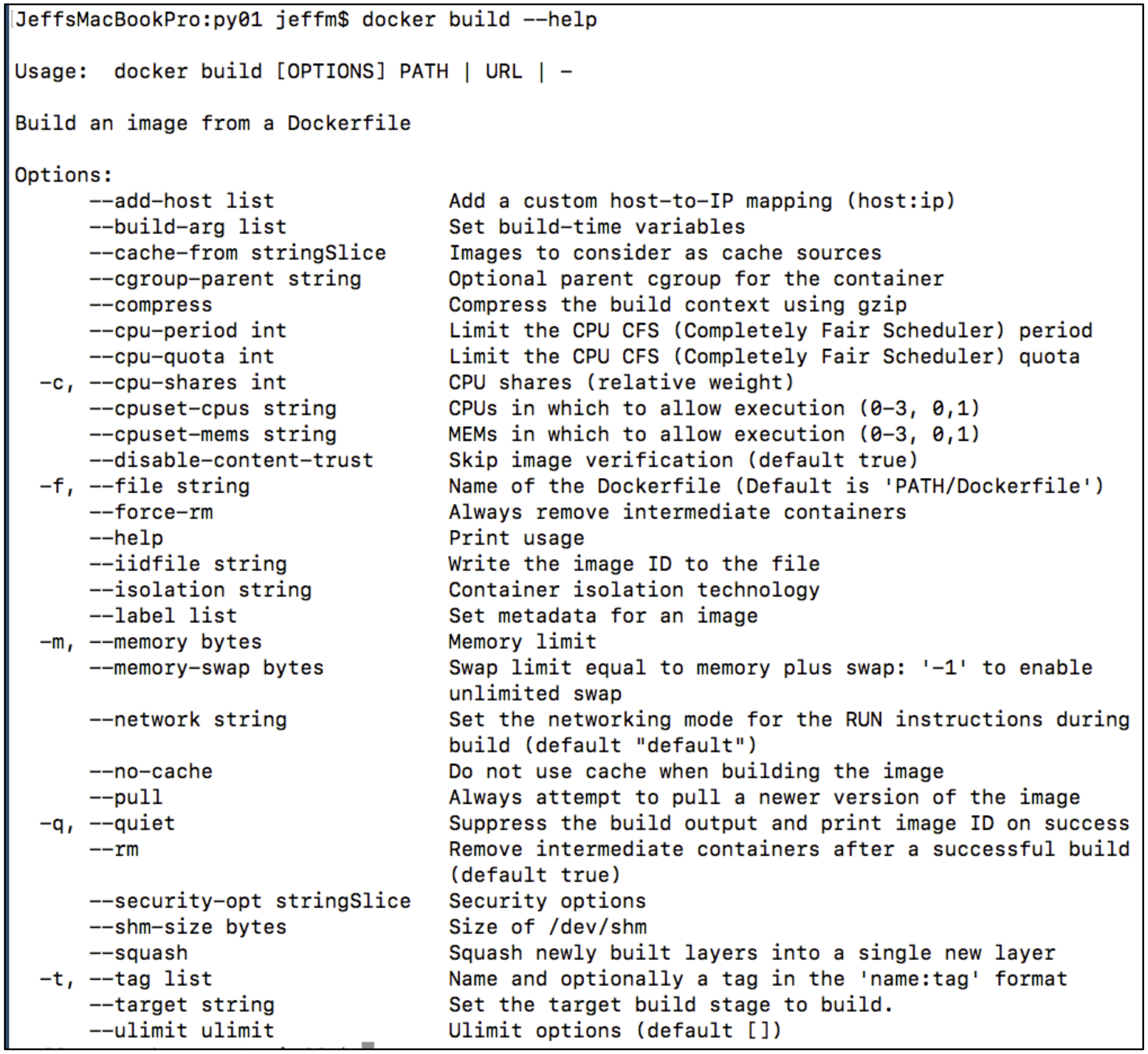
CMD ["python2"] − for this example a parameter − python2 − to pass into the container when it is started.

There can be only 1 CMD in a Dockerfile.

We will discuss CMD , along with ENTRYPOINT in a subsequent section.

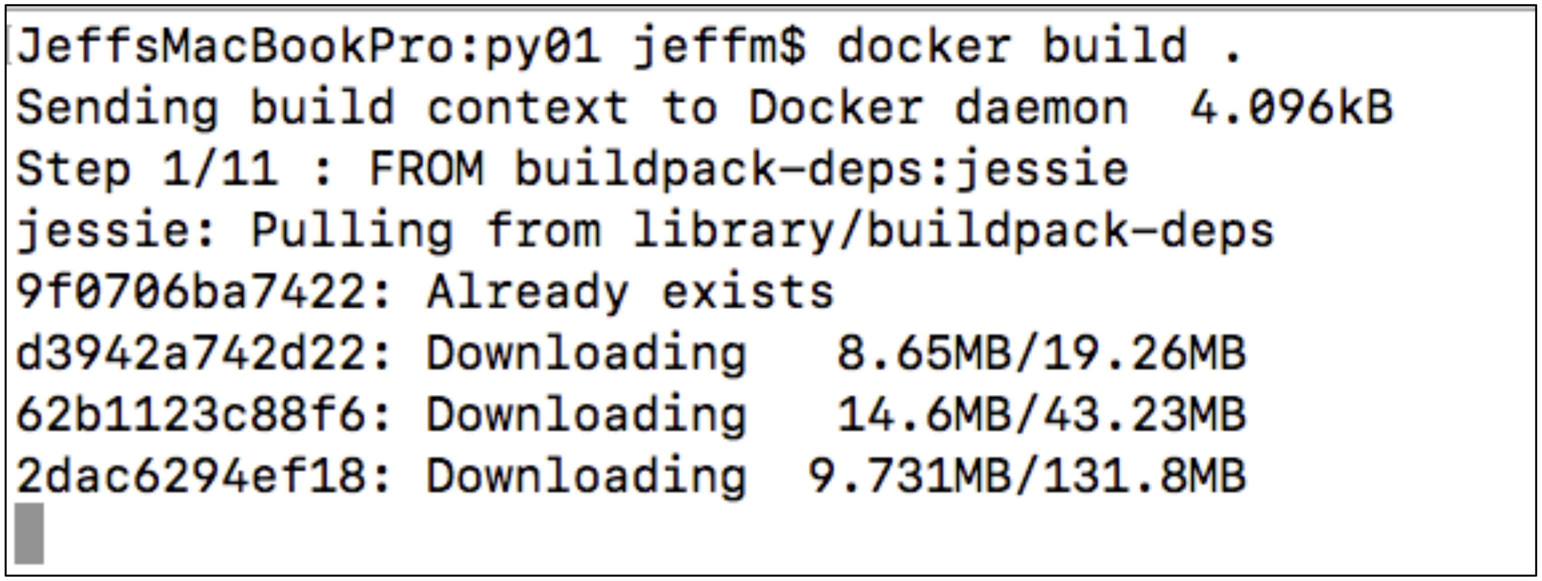
Docker Example py01

- OK – it's time to create our Docker image.
- To build an image from a Dockerfile use the "build" command.



```
JeffsMacBookPro:py01 jeffm$ docker build --help

Usage:  docker build [OPTIONS] PATH | URL | -

Build an image from a Dockerfile

Options:
      --add-host list           Add a custom host-to-IP mapping (host:ip)
      --build-arg list          Set build-time variables
      --cache-from stringSlice  Images to consider as cache sources
      --cgroup-parent string    Optional parent cgroup for the container
      --compress                Compress the build context using gzip
      --cpu-period int          Limit the CPU CFS (Completely Fair Scheduler) period
      --cpu-quota int           Limit the CPU CFS (Completely Fair Scheduler) quota
  -c, --cpu-shares int          CPU shares (relative weight)
      --cpuset-cpus string      CPUs in which to allow execution (0-3, 0,1)
      --cpuset-mems string      MEMs in which to allow execution (0-3, 0,1)
      --disable-content-trust   Skip image verification (default true)
  -f, --file string             Name of the Dockerfile (Default is 'PATH/Dockerfile')
      --force-rm                Always remove intermediate containers
      --help                    Print usage
      --iidfile string          Write the image ID to the file
      --isolation string        Container isolation technology
      --label list              Set metadata for an image
  -m, --memory bytes            Memory limit
      --memory-swap bytes       Swap limit equal to memory plus swap: '-1' to enable
                                unlimited swap
      --network string          Set the networking mode for the RUN instructions during
                                build (default "default")
      --no-cache                Do not use cache when building the image
      --pull                    Always attempt to pull a newer version of the image
  -q, --quiet                   Suppress the build output and print image ID on success
      --rm                      Remove intermediate containers after a successful build
                                (default true)
      --security-opt stringSlice  Security options
      --shm-size bytes          Size of /dev/shm
      --squash                  Squash newly built layers into a single new layer
  -t, --tag list                Name and optionally a tag in the 'name:tag' format
      --target string           Set the target build stage to build.
      --ulimit ulimit           Ulimit options (default [])
```

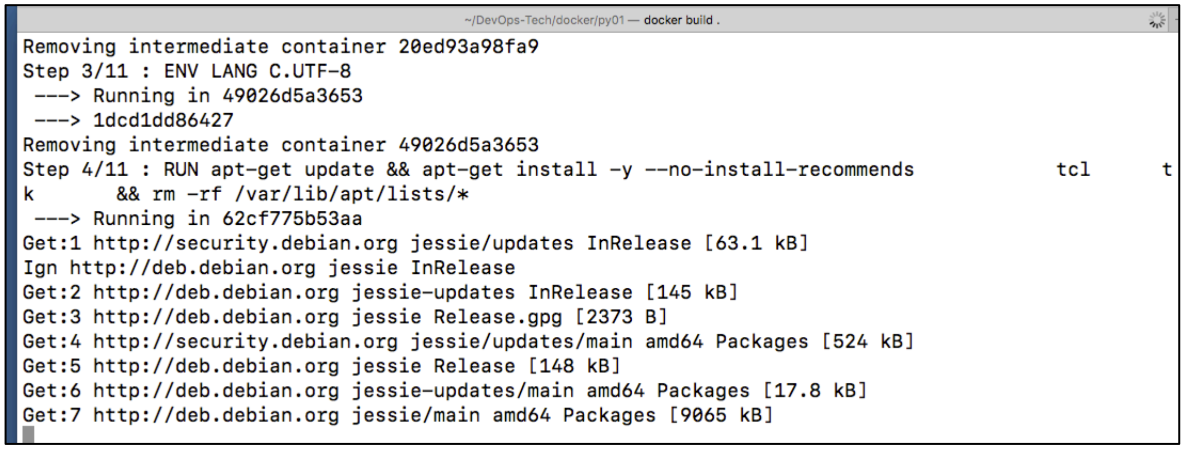
Docker Example py01



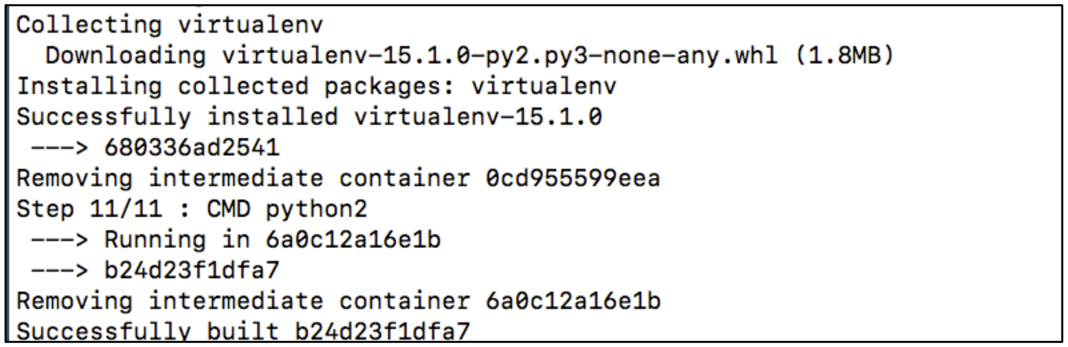
```
JeffsMacBookPro:py01 jeffm$ docker build .
Sending build context to Docker daemon  4.096kB
Step 1/11 : FROM buildpack-deps:jessie
jessie: Pulling from library/buildpack-deps
9f0706ba7422: Already exists
d3942a742d22: Downloading   8.65MB/19.26MB
62b1123c88f6: Downloading   14.6MB/43.23MB
2dac6294ef18: Downloading  9.731MB/131.8MB
```

docker build . directs docker to use the Dockerfile in the current directory

Docker Example py01



```
Removing intermediate container 20ed93a98fa9
Step 3/11 : ENV LANG C.UTF-8
 ---> Running in 49026d5a3653
 ---> 1dcd1dd86427
Removing intermediate container 49026d5a3653
Step 4/11 : RUN apt-get update && apt-get install -y --no-install-recommends            tcl       t
k        && rm -rf /var/lib/apt/lists/*
 ---> Running in 62cf775b53aa
Get:1 http://security.debian.org jessie/updates InRelease [63.1 kB]
Ign http://deb.debian.org jessie InRelease
Get:2 http://deb.debian.org jessie-updates InRelease [145 kB]
Get:3 http://deb.debian.org jessie Release.gpg [2373 B]
Get:4 http://security.debian.org jessie/updates/main amd64 Packages [524 kB]
Get:5 http://deb.debian.org jessie Release [148 kB]
Get:6 http://deb.debian.org jessie-updates/main amd64 Packages [17.8 kB]
Get:7 http://deb.debian.org jessie/main amd64 Packages [9065 kB]
```

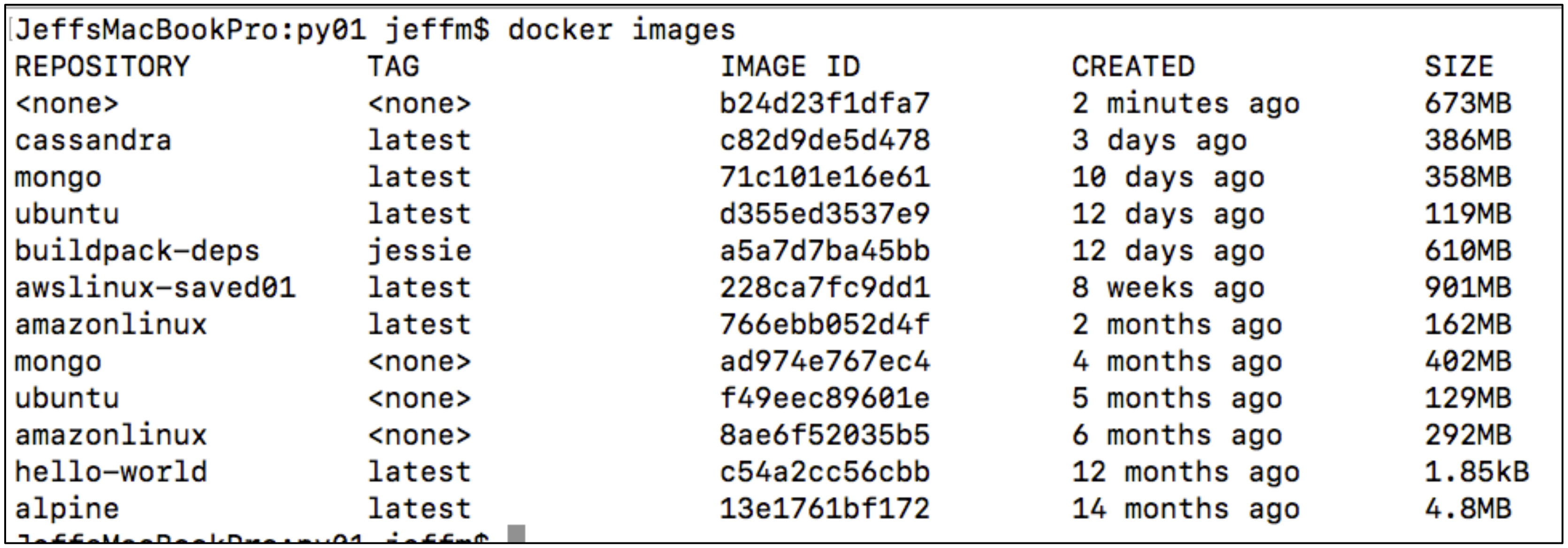


```
Collecting virtualenv
  Downloading virtualenv-15.1.0-py2.py3-none-any.whl (1.8MB)
Installing collected packages: virtualenv
Successfully installed virtualenv-15.1.0
 ---> 680336ad2541
Removing intermediate container 0cd955599eea
Step 11/11 : CMD python2
 ---> Running in 6a0c12a16e1b
 ---> b24d23f1dfa7
Removing intermediate container 6a0c12a16e1b
Successfully built b24d23f1dfa7
```

docker build . takes a while to download, extract, and build the image specified in the Dockerfile

Docker Example py01

Command − **docker images** − lists all Docker images on your machine

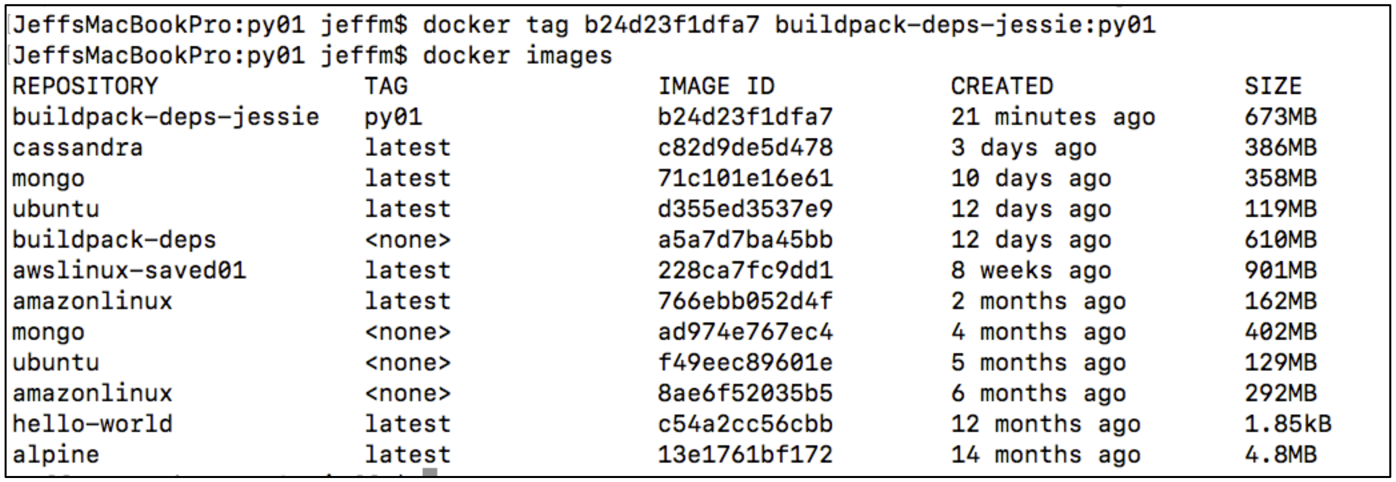


```
JeffsMacBookPro:py01 jeffm$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
<none>              <none>              b24d23f1dfa7        2 minutes ago       673MB
cassandra           latest              c82d9de5d478        3 days ago          386MB
mongo               latest              71c101e16e61        10 days ago         358MB
ubuntu              latest              d355ed3537e9        12 days ago         119MB
buildpack-deps      jessie              a5a7d7ba45bb        12 days ago         610MB
awslinux-saved01    latest              228ca7fc9dd1        8 weeks ago         901MB
amazonlinux         latest              766ebb052d4f        2 months ago        162MB
mongo               <none>              ad974e767ec4        4 months ago        402MB
ubuntu              <none>              f49eec89601e        5 months ago        129MB
amazonlinux         <none>              8ae6f52035b5        6 months ago        292MB
hello-world         latest              c54a2cc56cbb        12 months ago       1.85kB
alpine              latest              13e1761bf172        14 months ago       4.8MB
```

Notice the image we just had Docker build does NOT have a REPOSITORY or TAG setting.

Let's set a Tag …

Docker Example py01



```
JeffsMacBookPro:py01 jeffm$ docker tag b24d23f1dfa7 buildpack-deps-jessie:py01
JeffsMacBookPro:py01 jeffm$ docker images
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
buildpack-deps-jessie   py01                b24d23f1dfa7        21 minutes ago      673MB
cassandra               latest              c82d9de5d478        3 days ago          386MB
mongo                   latest              71c101e16e61        10 days ago         358MB
ubuntu                  latest              d355ed3537e9        12 days ago         119MB
buildpack-deps          <none>              a5a7d7ba45bb        12 days ago         610MB
awslinux-saved01        latest              228ca7fc9dd1        8 weeks ago         901MB
amazonlinux             latest              766ebb052d4f        2 months ago        162MB
mongo                   <none>              ad974e767ec4        4 months ago        402MB
ubuntu                  <none>              f49eec89601e        5 months ago        129MB
amazonlinux             <none>              8ae6f52035b5        6 months ago        292MB
hello-world             latest              c54a2cc56cbb        12 months ago       1.85kB
alpine                  latest              13e1761bf172        14 months ago       4.8MB
```

docker tag Image-ID Repository:Tag – tags an image

NOTE: you can create multiple tags on an image. Each time your create a different tag, you get another image listed in "docker images".

Command **docker –rmi Repository:Tag -** removes an image

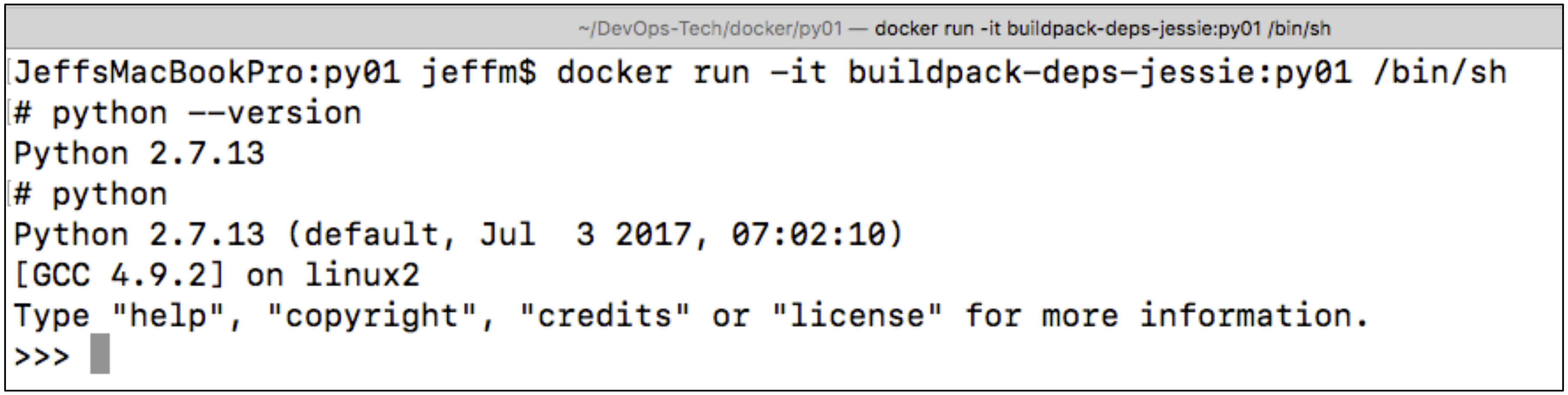
Be careful using "docker rmi" – if you only have one image, it will remove that image.

Docker Example py01

- Let's run our Docker image:

```
docker run -it buildpack-deps-jessie:py01 /bin/sh
```

Docker Example py01



```
~/DevOps-Tech/docker/py01 — docker run -it buildpack-deps-jessie:py01 /bin/sh
JeffsMacBookPro:py01 jeffm$ docker run -it buildpack-deps-jessie:py01 /bin/sh
# python --version
Python 2.7.13
# python
Python 2.7.13 (default, Jul  3 2017, 07:02:10)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

The docker run command:

-it   **-i** is run the Docker container interactively, **-t** is allocate a terminal

buildpack-deps-jessie:py01 – is Repository:Tag

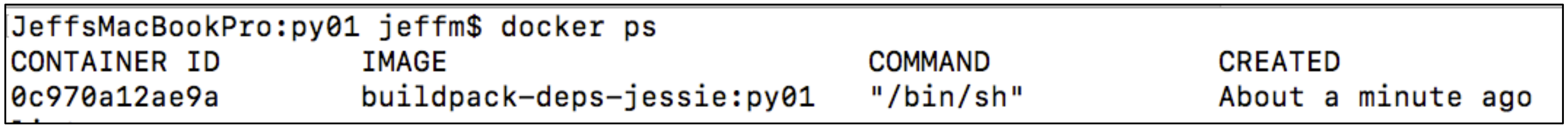
/bin/sh – is the command we had the container run when it started. For this example we ran a shell

Docker Example py01

While our Docker container is still running:

open up another terminal window and run command:

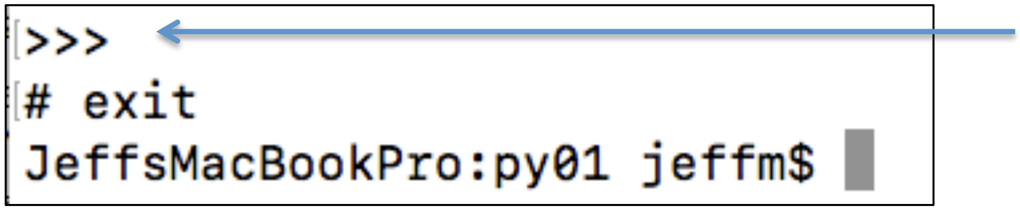
docker ps



```
JeffsMacBookPro:py01 jeffm$ docker ps
CONTAINER ID        IMAGE                         COMMAND             CREATED
0c970a12ae9a        buildpack-deps-jessie:py01    "/bin/sh"           About a minute ago
```

Docker Example py01

Now, let's exit the Docker container: from Python hit Control-D to exit Python, from the shell type exit:



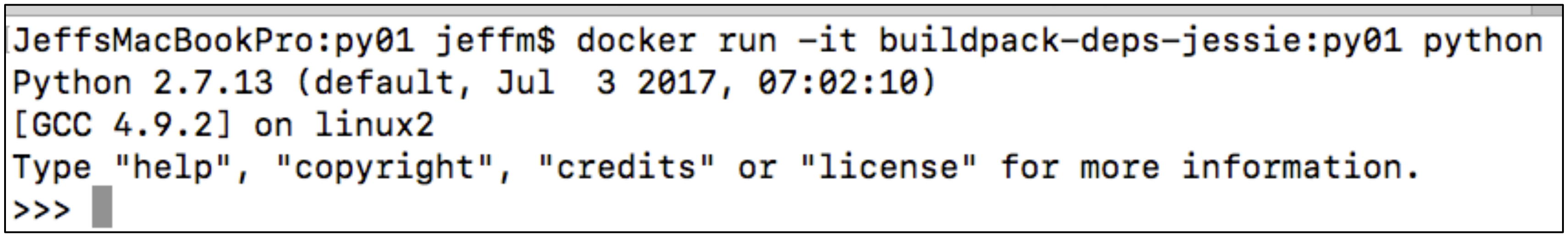
```
>>>
# exit
JeffsMacBookPro:py01 jeffm$
```

control-D hit here

Docker Example py01

Now let's rerun the container with a different command:

docker run -it buildpack-deps-jessie:py01 python

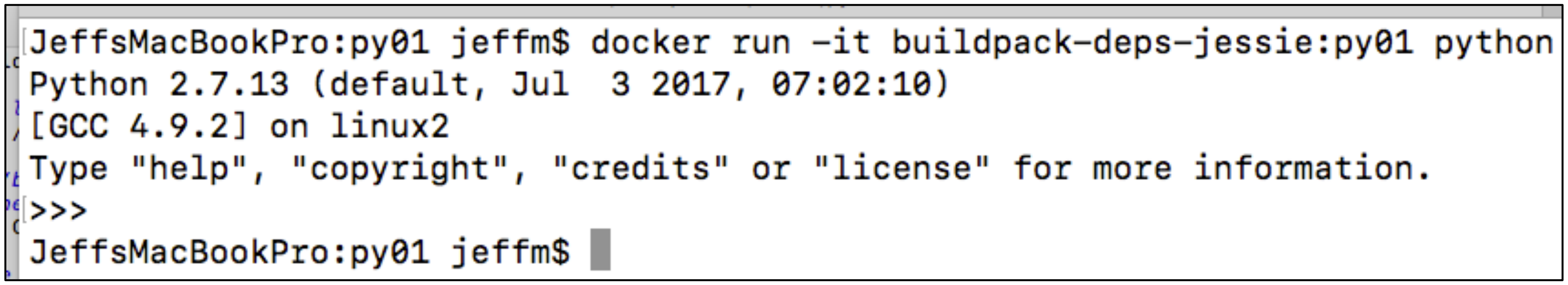


```
JeffsMacBookPro:py01 jeffm$ docker run -it buildpack-deps-jessie:py01 python
Python 2.7.13 (default, Jul  3 2017, 07:02:10)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

Docker Example py01

This time, when the container came up , Python was immediate executed.

Now, let's exit Python with a control-D again:

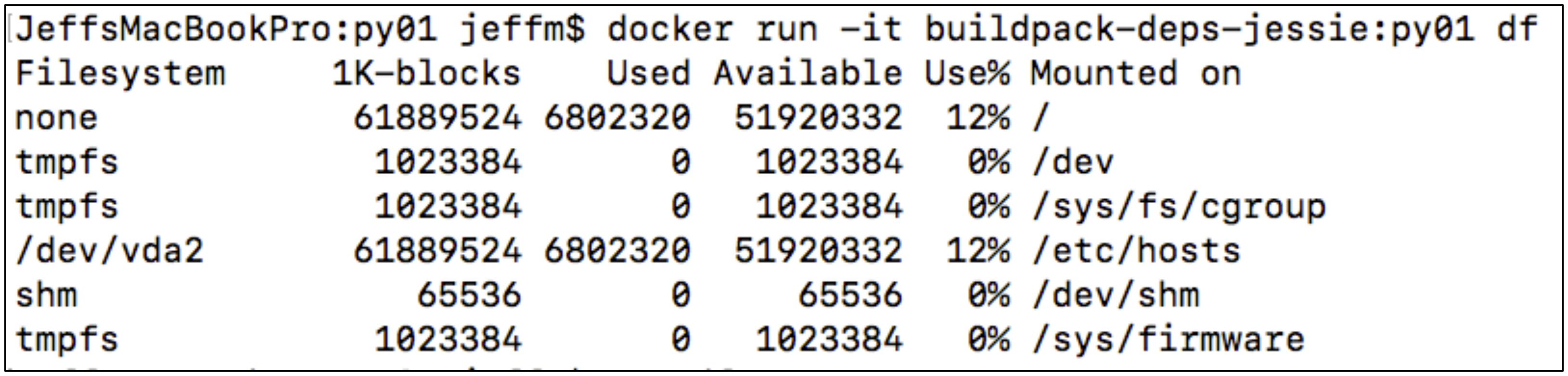


```
JeffsMacBookPro:py01 jeffm$ docker run -it buildpack-deps-jessie:py01 python
Python 2.7.13 (default, Jul  3 2017, 07:02:10)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
JeffsMacBookPro:py01 jeffm$
```

This time, when we exited Python, the Docker container exited.
When the command passed into "docker run" terminates,
Docker terminates the image.

Docker Example py01

Let's run our Docker container with a different command:



```
JeffsMacBookPro:py01 jeffm$ docker run -it buildpack-deps-jessie:py01 df
Filesystem     1K-blocks    Used Available Use% Mounted on
none            61889524 6802320  51920332  12% /
tmpfs            1023384       0   1023384   0% /dev
tmpfs            1023384       0   1023384   0% /sys/fs/cgroup
/dev/vda2       61889524 6802320  51920332  12% /etc/hosts
shm                65536       0     65536   0% /dev/shm
tmpfs            1023384       0   1023384   0% /sys/firmware
```

This time we ran the **df** command which display disk space usage