
docker

Do any of these sound familiar

- It worked fine on my machine
- I set that up months ago but can't remember the details
- I installed so much stuff trying to get it to work I can't really say which are actually required
- It only works on a linux machine
- I just want to test it without installing
- I want to test/use different versions
- Set-up is too complicated to explain in a paper

Possible solutions

- Hand holding support
- Very detailed documentation
- Virtual machine
- Docker

Virtual Machines

- "is an emulation of a particular computer system"[1]
- Completely separate
- Full set of resources (or as much as possible)
- Current set-up can be saved, copied and shared
- No central repositories of images
- Black-box on how it was set-up
- Full GUI support
- Ideal for working on a host operating system/ controlled set-up
  - ex. Word on a Linux or doing a training where everyone has the same set-up

[1]https://en.wikipedia.org/w/index.php?titlatorle=Virtual_machine&oldid=669500805

How Docker describes itself

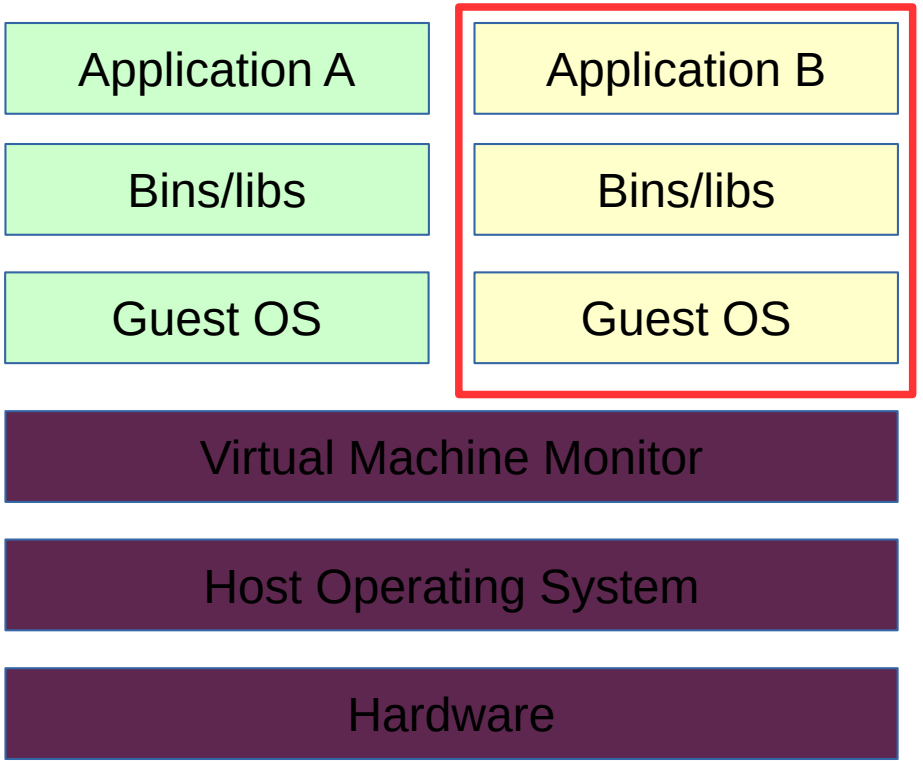
- Build, Ship, Run
- An open platform for distributed applications for developers and sysadmins
- Ship Applications Faster and Easier
- Application Portability and Infrastructure Flexibility
- Dynamically Update, Change and Scale Apps

Docker

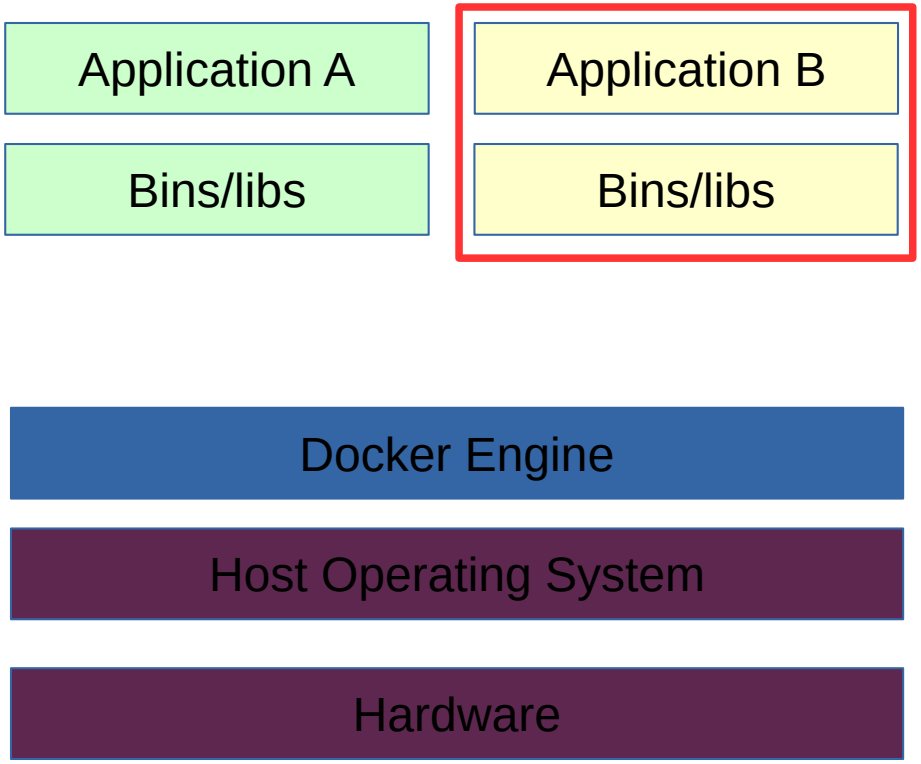
- Uses LinuX Containers (LXC)
- Sharing resources
- Central repositories of images
- Current set-up can be saved, copied and shared
- Dockerfile showing exact set-up
- Typically no GUI support
- Ideal for running a single application or service
- Many Docker images can be run side by side

Docker compared to VMs

Virtual Machines



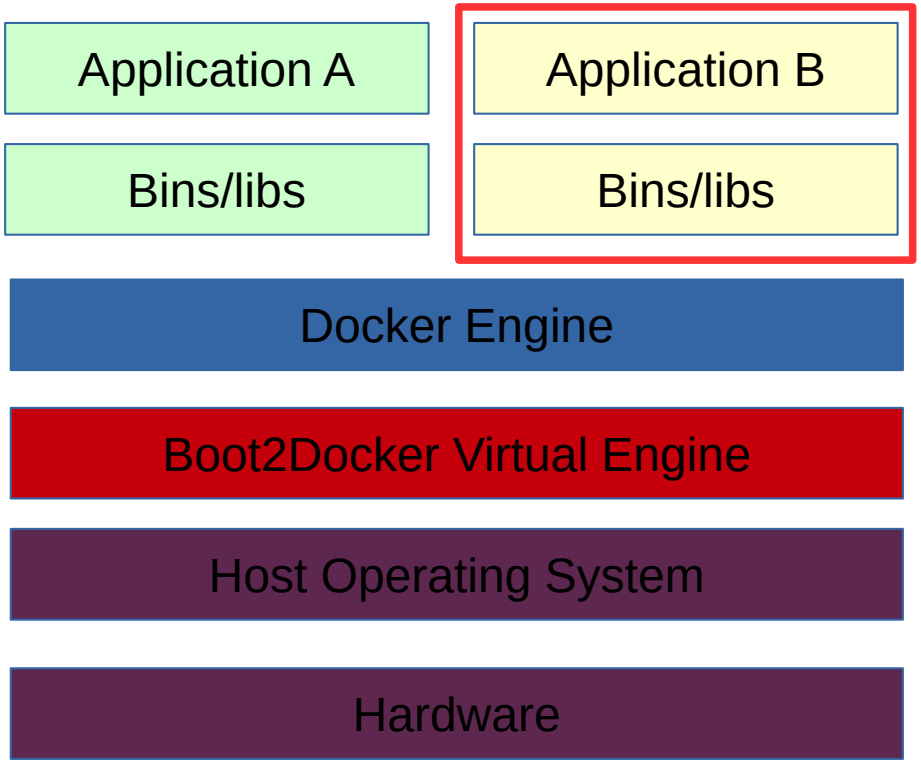
Docker Images



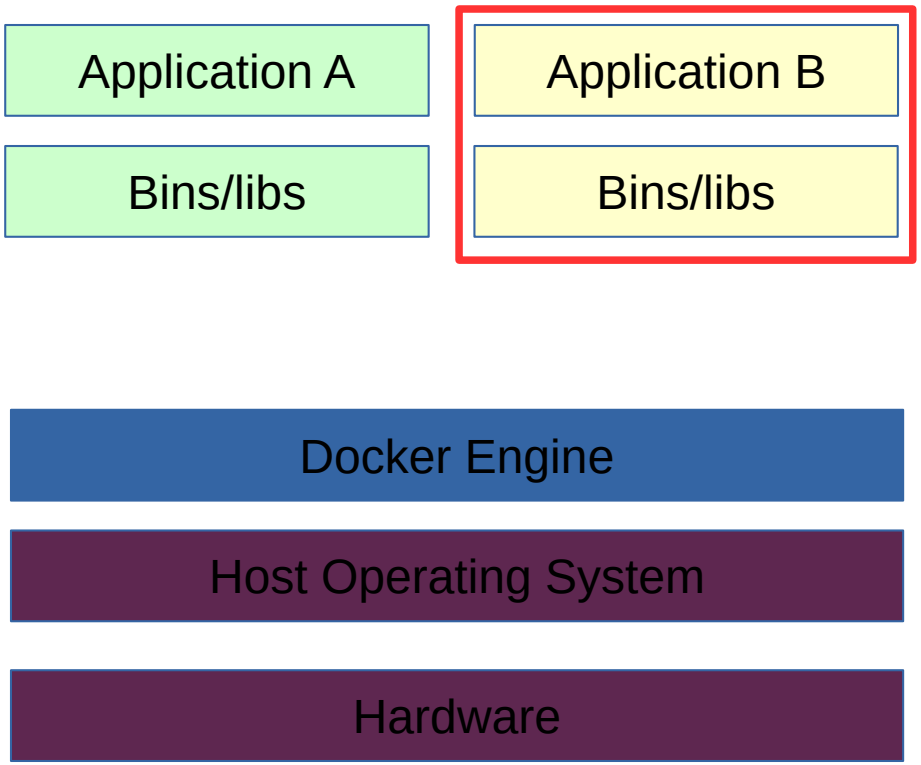
http://www.jayway.com/wp-content/uploads/2015/03/vm-vs-docker.png

Docker Linux vs Windows

Windows/ Mac OS



Linux



http://www.jayway.com/wp-content/uploads/2015/03/vm-vs-docker.png

A few Run Examples

- docker run docker/whalesay:latest cowsay Hi bioinference group

See: http://docs.docker.com/linux/started/

Parts of docker command

- docker : Starts the docker application
- run : docker command to run an image
- docker/whalesay : image to run
  - docker : owner of the repository
  - whalesay : image to run
  - :latest :tag of image to run (:latest is the default
- cowsay : Application inside the image to run
- Hi Bioinference group: parameters for application

Separate Run environment

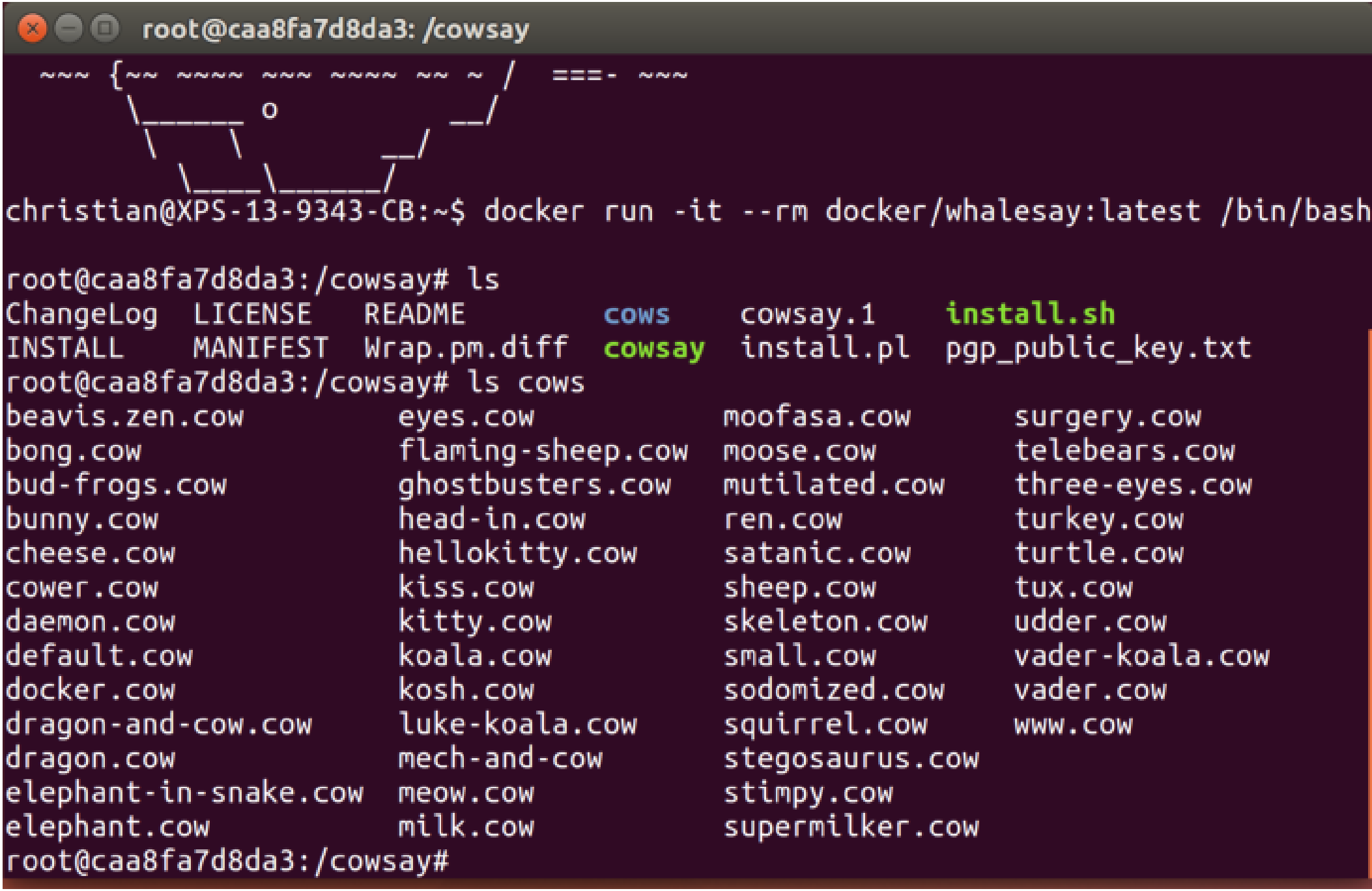
- docker run -i -t --rm docker/whalesay

  -i = Keep STDIN open even if not attached

  -t = Allocate a pseudo-TTY

  --rm = Automatically remove the container when it exits

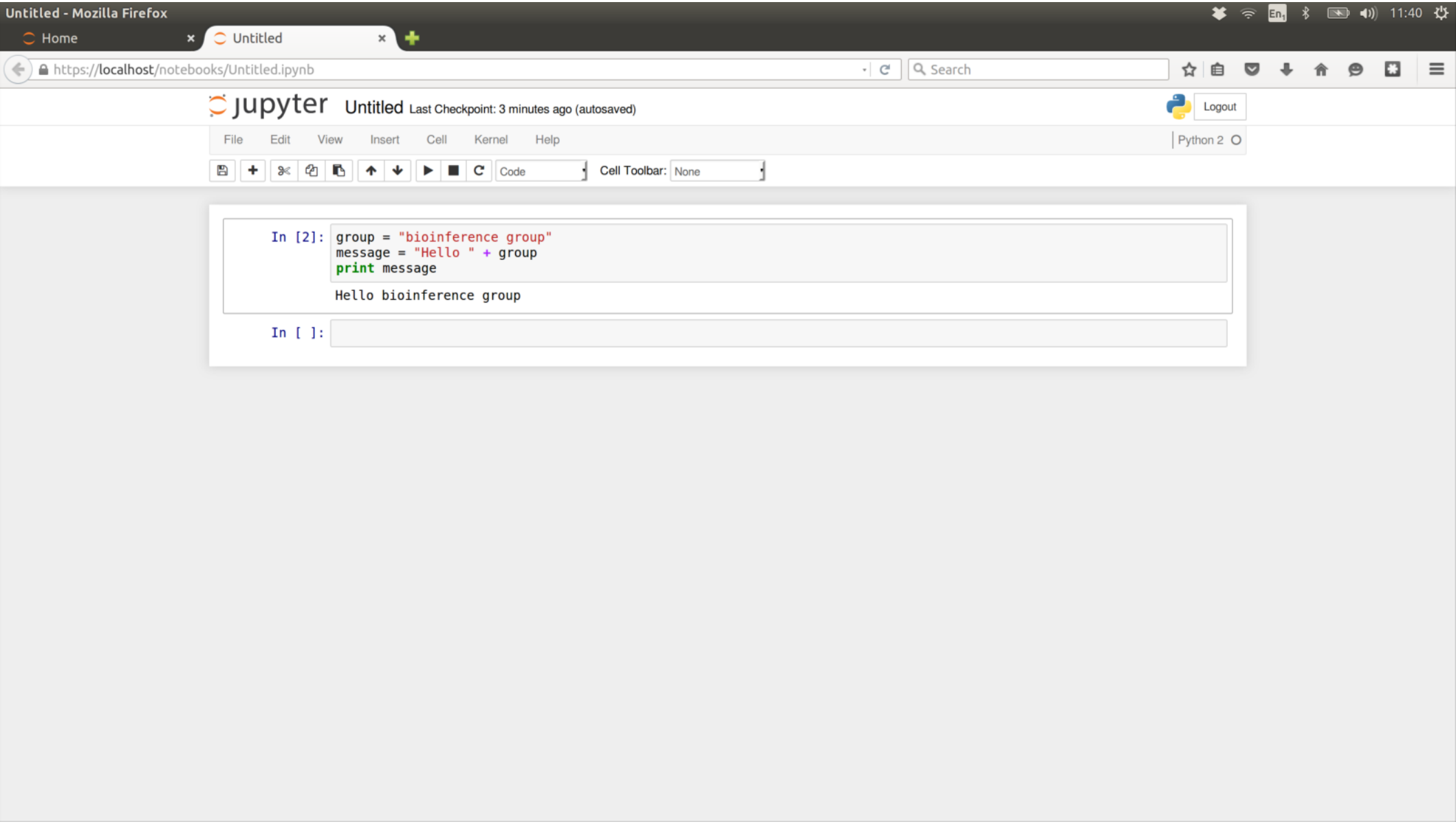
- Open a bin/bash terminal
- ls
  - Cows directory , cowsay progam


root@caa8fa7d8da3: /cowsay
christian@XPS-13-9343-CB:~$ docker run -it --rm docker/whalesay:latest /bin/bash
root@caa8fa7d8da3:/cowsay# ls
ChangeLog LICENSE README cows cowsay.1 install.sh
INSTALL MANIFEST Wrap.pm.diff cowsay install.pl pgp_public_key.txt
root@caa8fa7d8da3:/cowsay# ls cows
beavis.zen.cow eyes.cow moofasa.cow surgery.cow
bong.cow flaming-sheep.cow moose.cow telebears.cow
bud-frogs.cow ghostbusters.cow mutilated.cow three-eyes.cow
bunny.cow head-in.cow ren.cow turkey.cow
cheese.cow hellokitty.cow satanic.cow turtle.cow
cower.cow kiss.cow sheep.cow tux.cow
daemon.cow kitty.cow skeleton.cow udder.cow
default.cow koala.cow small.cow vader-koala.cow
docker.cow kosh.cow sodomized.cow vader.cow
dragon-and-cow.cow luke-koala.cow squirrel.cow www.cow
dragon.cow mech-and-cow stegosaurus.cow
elephant-in-snake.cow meow.cow stimpy.cow
elephant.cow milk.cow supermilker.cow
root@caa8fa7d8da3:/cowsay#

Ipython example

- docker run -d -p 443:8888 -e "PASSWORD=test" --name iserver ipython/scipyserver
- docker ps
- https://0.0.0.0/tree (use password entered in run command)
- If using boot2docker
  - boot2docker ip (to get ip address vm uses)
  - https://*.*.*.*/tree
- https://www.ibm.com/developerworks/community/blogs/jfp/entry/using_ipython_notebooks_in_docker_containers_on_windows?lang=en

Docker ipython



Ipython continued

- -d
  - Run container in background and print container ID
- -p 443:8888
  - Publish a container's port(s) to the host
- -e "PASSWORD=test"
  - Set environment variables
- --name iserver
  - Assign a name to the container
- ipython/scipyserver
  - Name of the image

Container

- docker ps
  - CONTAINER ID b61cf298f297
  - IMAGE ipython/scipyserver
  - COMMAND "/notebook.sh"
  - CREATED 28 minutes ago
  - STATUS Up 28 minutes
  - PORTS 0.0.0.0:443->8888/tcp
  - NAMES iserver
- Docker ps -a
- docker rm `docker ps --no-trunc -aq`

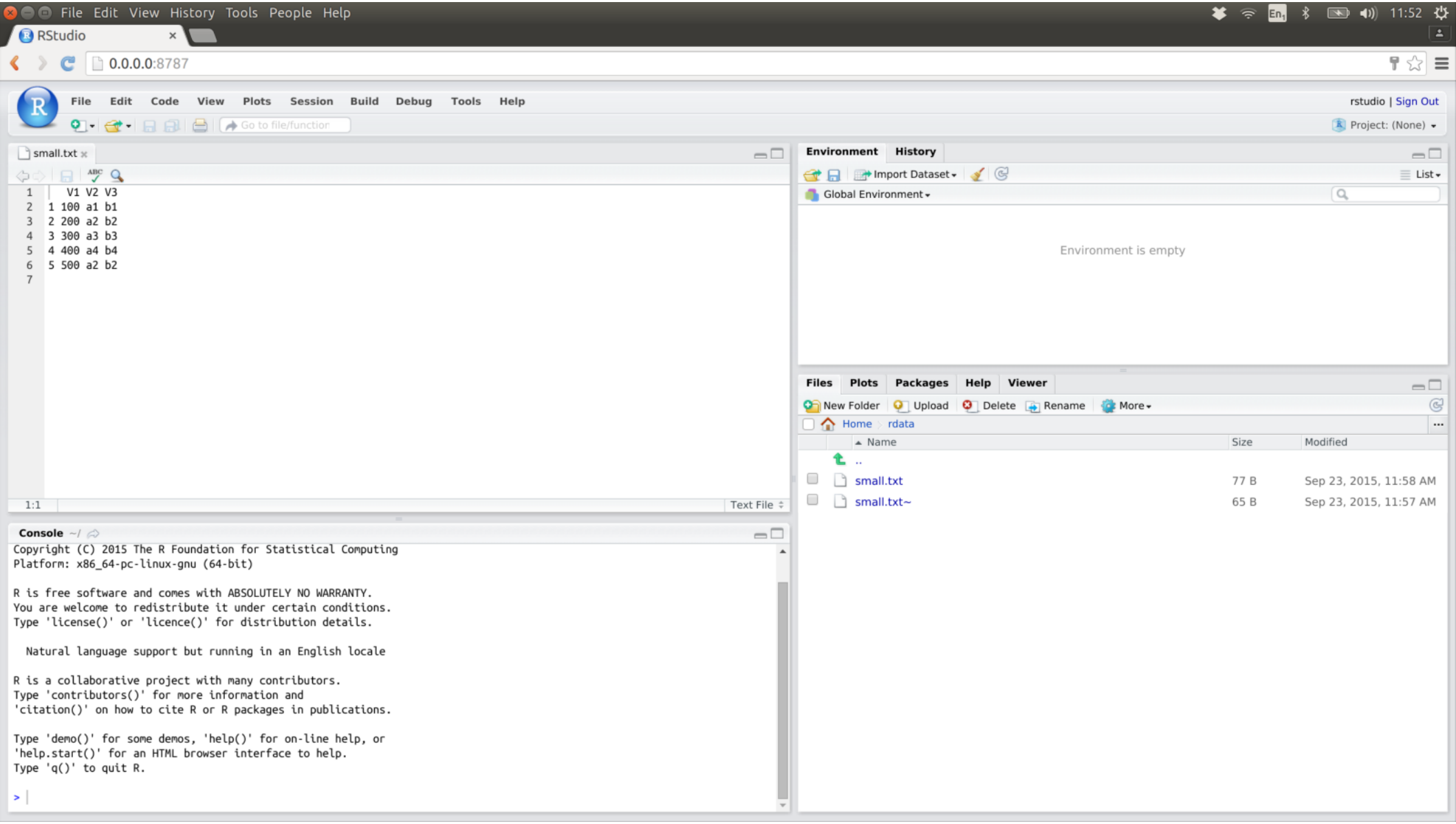
Container start and start

- At https://0.0.0.0/tree
  - New Python 2
  - print “hello world”
  - Run Button
- Close and reopen Jupiter
- docker stop iserver
- See https://0.0.0.0/tree fails
- docker start iserver
- See https://0.0.0.0/tree saves still there

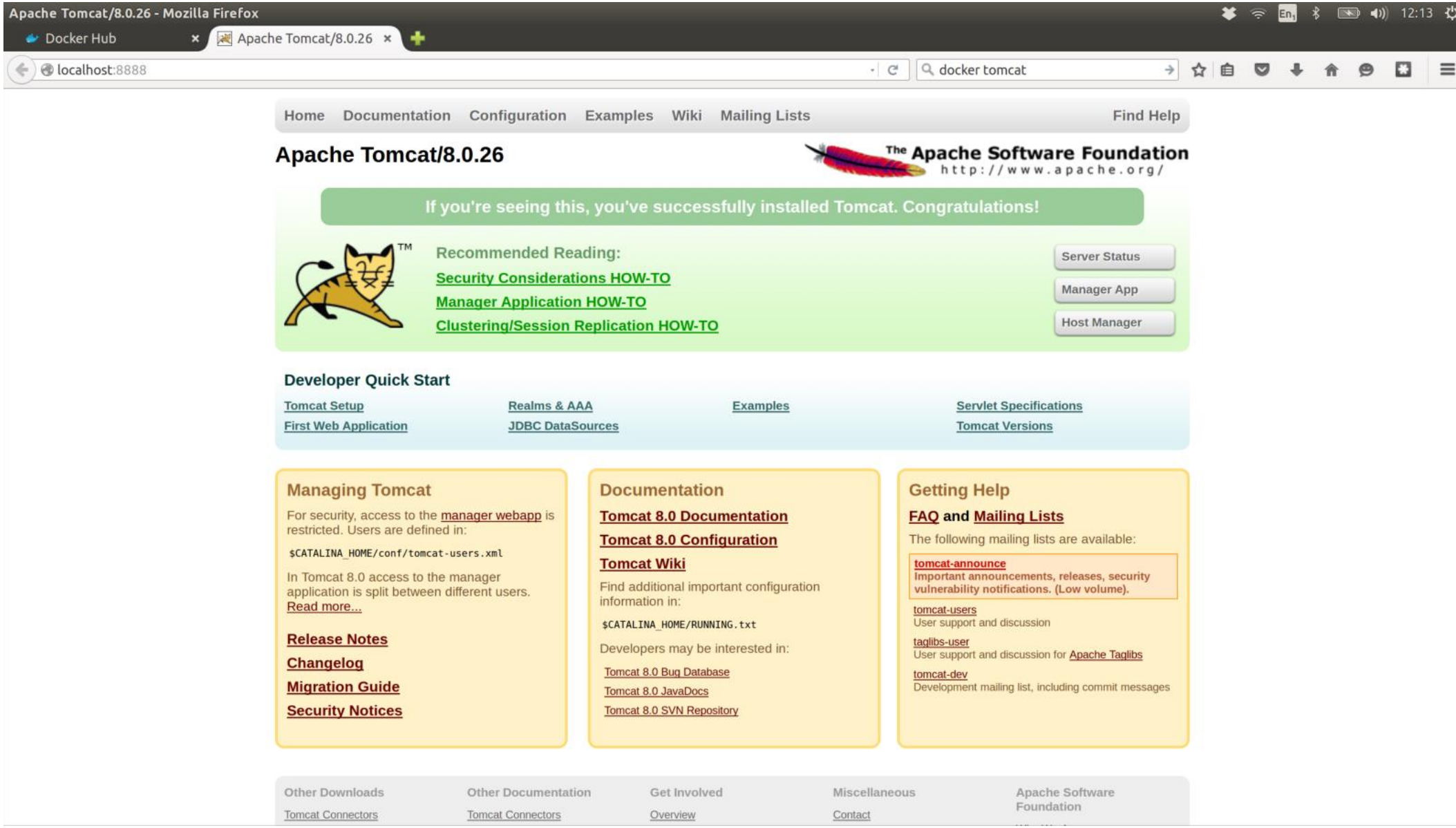
RStudio

- docker run -d -p 8787:8787 -v /home/christian/docker/rdata/:/home/rstudio/rdata --name=rstudio -e USER=rstudio -e PASSWORD=rstudio rocker/rstudio
  - v maps a directory into the docker container
- http://0.0.0.0:8787/
- Outside changes to ../rdata are visible in rstudio
- see https://github.com/rocker-org/rocker/wiki/Using-the-RStudio-image

Docker RStudio

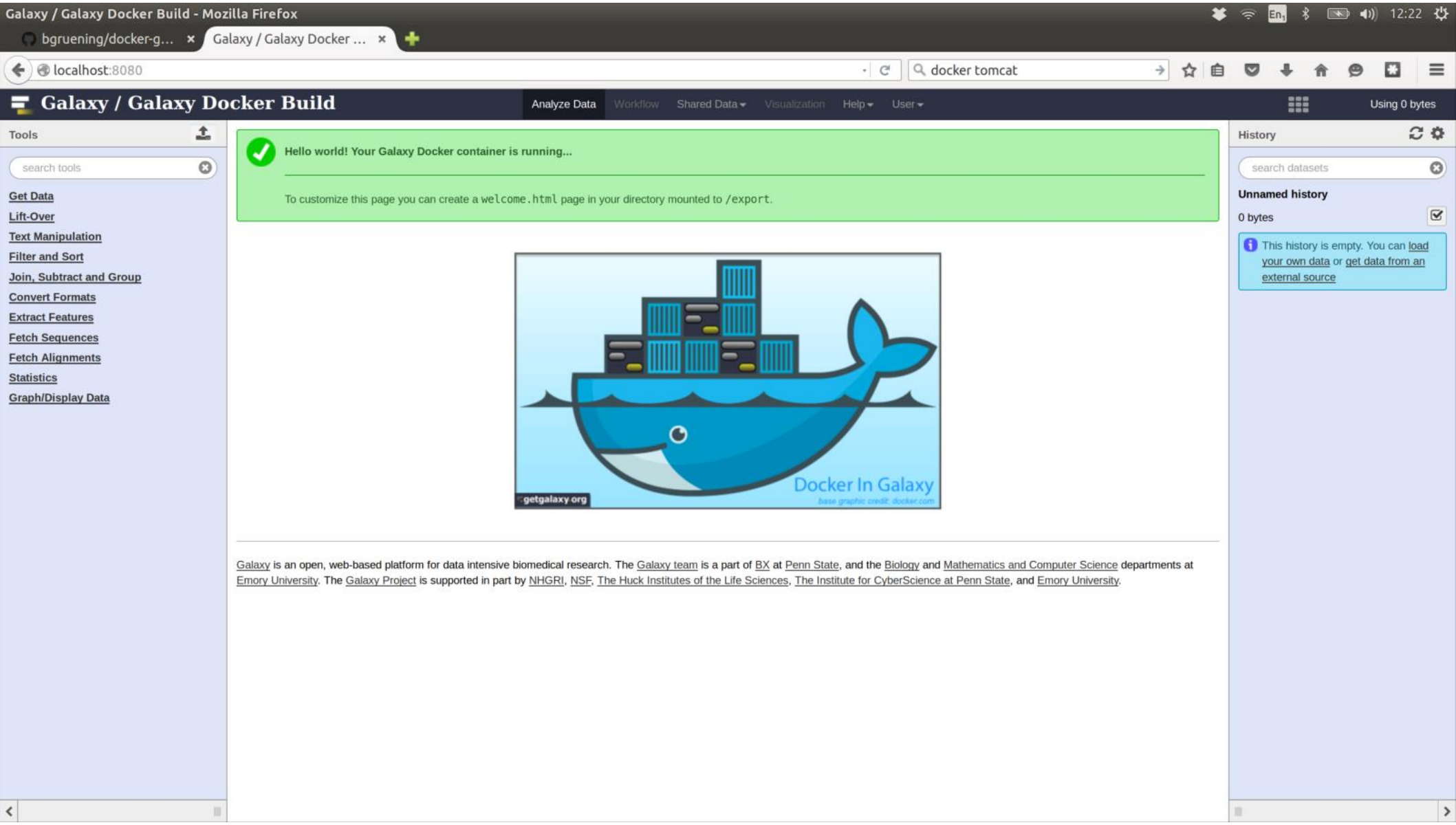


docker run -it --rm -p 8888:8080 tomcat:8.0

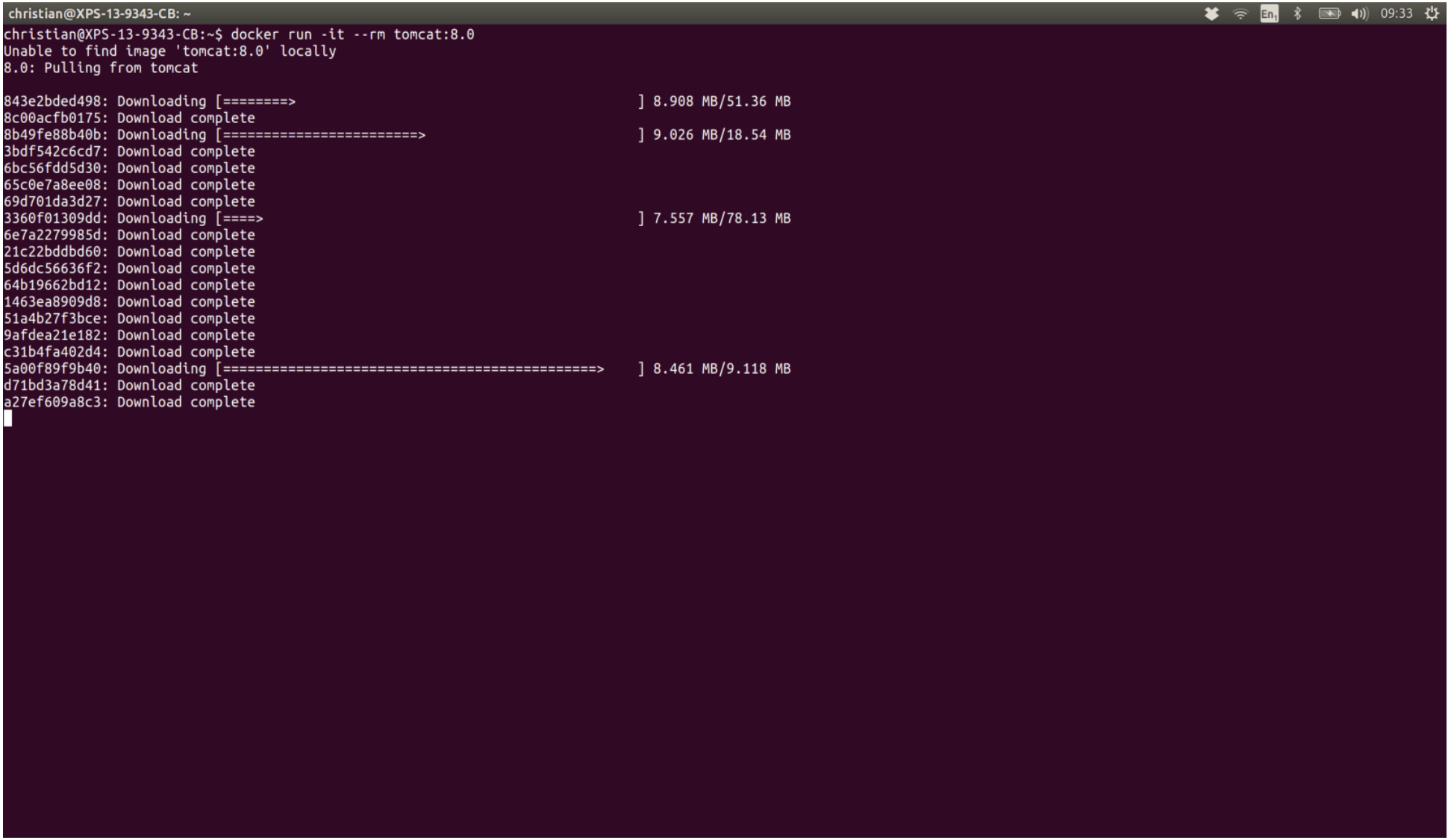


- https://hub.docker.com/_/tomcat/

docker run -d -p 8080:80 -p 8021:21 bgruening/galaxy-stable



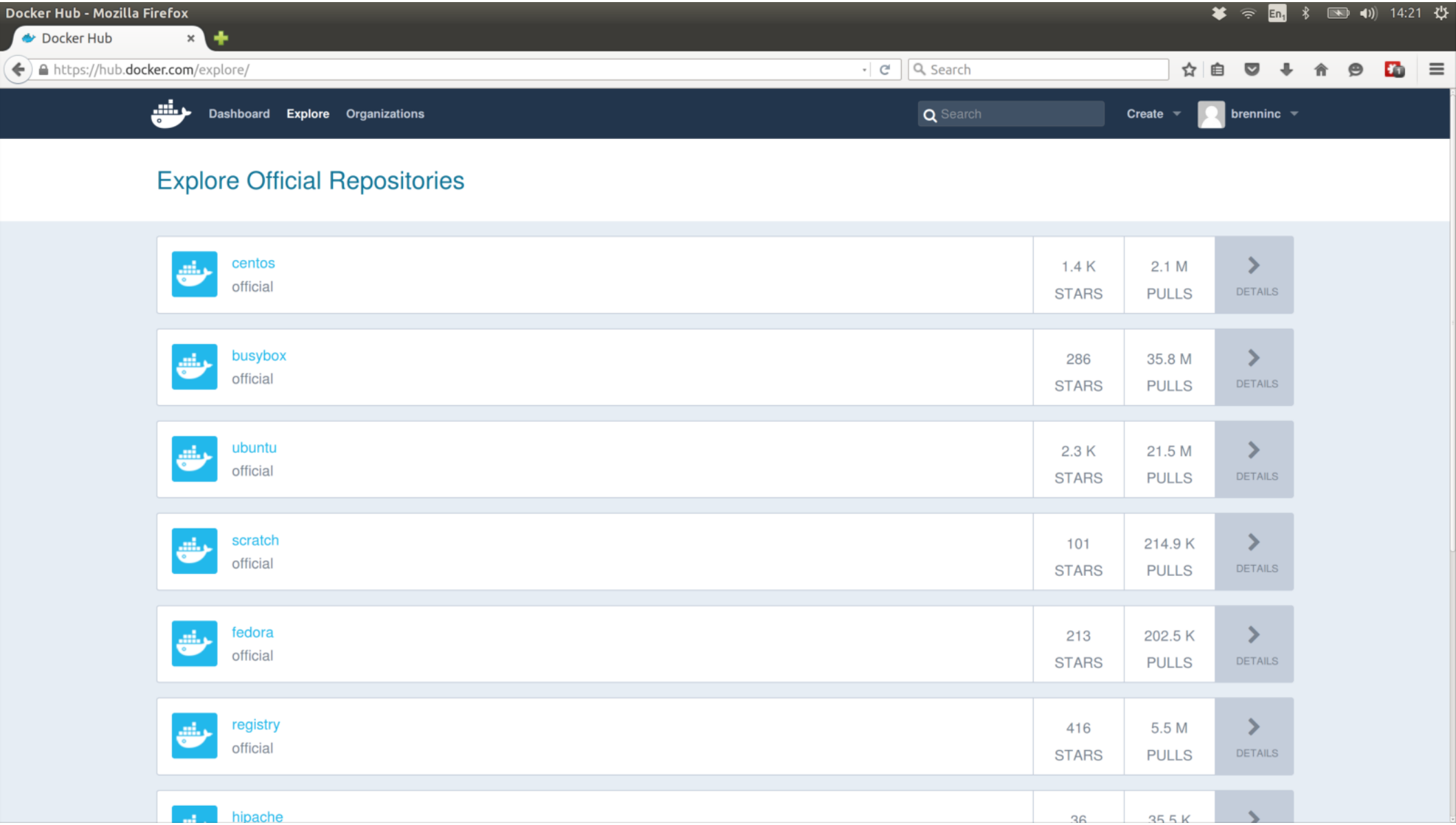
Docker downloads the first time



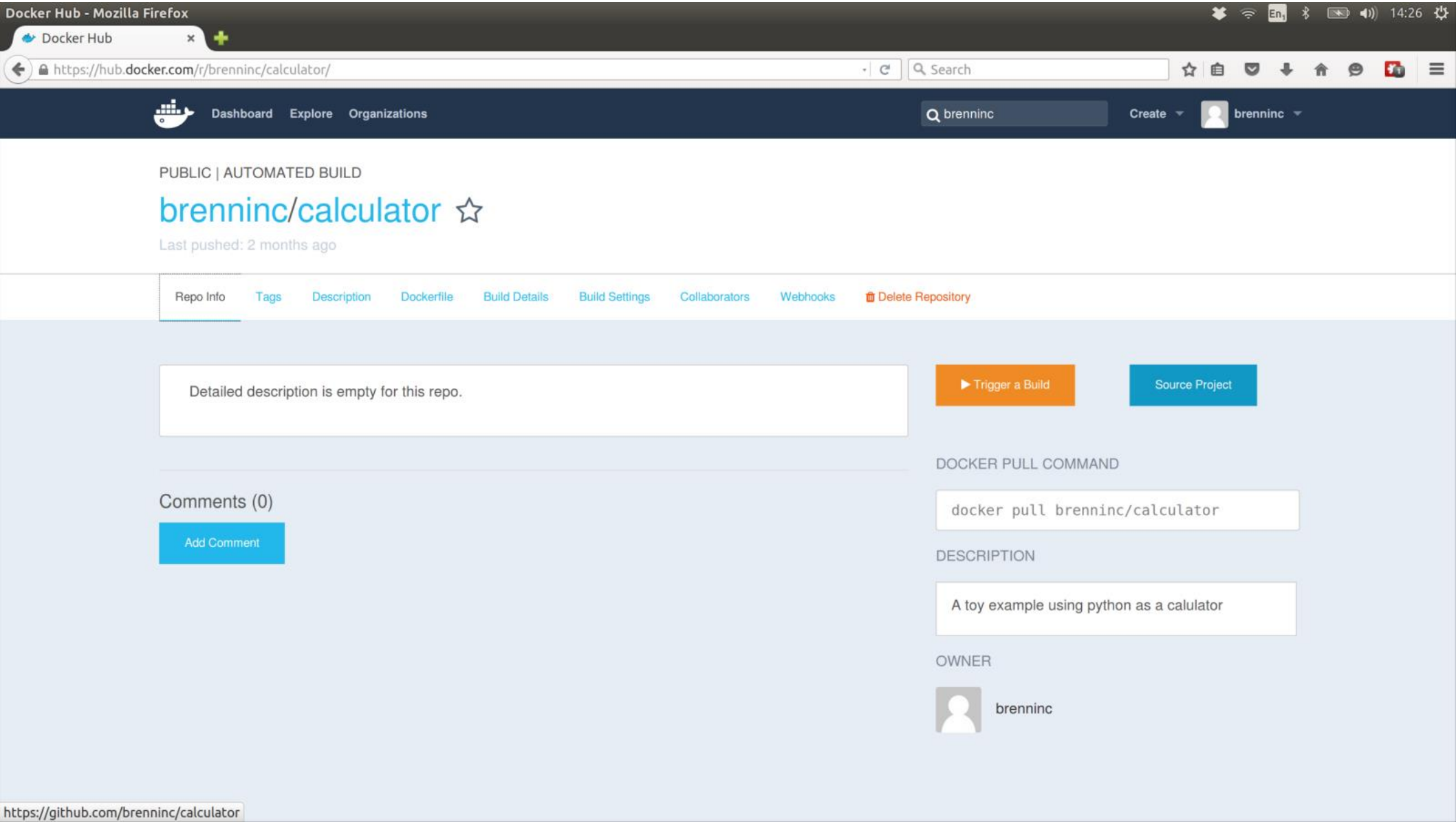
Ship

- https://hub.docker.com
  - Images that can be downloaded
- docker pull xyz (gets an image and its parents)
- docker run xyz (pulls if required)
- Many images linked to a github account
  - Dockerfile
  - Extra files
  - Info files
- Automatically built so you know exactly what you get

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



https://hub.docker.com/r/brenninc/calculator/



Build (the BAD WAY)

- docker run -i -t --name=bad ubuntu:14.04
  - curl --version

    curl: command not found
  - sudo apt-get install curl
  - curl --version
    - curl 7.35.0 .....
  - exit
- docker run -i -t --rm ubuntu:14.04
  - curl --version

    curl: command not found
- docker start -i bad
  - curl --version
    - curl 7.35.0 .....

Build (The bad way)

- These images can be uploaded to docker hub
- No Dockerfile will be available
- No Automatic build

- Would you trust someone else’s black box?

Build using Docker files

- Saved in a text file called Dockerfile
- Exact record of how the system was built
- Dockerfile can built upon other docker images
- Built up in layers
  - Max 128 layers
- Each command in a Dockerfile is a layer
- Docker file allow for “automatic builds” on Docker hub
- Docker files typically shared via github

Calculator Example

- docker run –rm brenninc/calculator 4+5*2
  - 4+5*2 = 14
- docker run –rm brenninc/calculator
  - 1 + 2 * 3 = 7

Dockerfile instructions

- FROM
- MAINTAINER
- LABEL
- RUN
- ENTRYPOINT
- CMD
- EXPOSE
- ENV
- COPY
- ADD
- VOLUME
- USER
- WORKDIR
- ONBUILD

From

- Base or parent image
- Can be an operating system
  - FROM ubuntu:14.04
  - FROM centos
  - FROM febora
- Only Linux family operating system
- Can be a base image
  - ipython/scipyserver
  - ipython/scipystack
  - ipython/ipython:3.x
  - …..
- Can be scratch
  - Root of operating system images

MAINTAINER

- A way of signalling who is responsible for the image

- MAINTAINER Christian Brenninkmeijer <Christian.Brenninkmeijer@manchester.ac.uk>
- MAINTAINER IPython Project <ipython-dev@scipy.org>

- Does count towards the 128 layer limit

LABEL

- key-value paired metadata

- LABEL com.example.label-with-value="foo"
- LABEL version="1.0"
- LABEL description="This text illustrates \
- that label-values can span multiple lines."

- Exposed vai
- docker inspect image_name
  - Includes other metadata
  - Includes info from MAINTAINER

RUN

- Executes command on base image and saves a new image
- apt-get Install stuff
- Download stuff
- Unzip stuff
- Create directories
- Run setup and config scripts
- Delete temporary files

Run examples

- RUN apt-get update && apt-get install -y python

- RUN curl -L http://downloads.sourceforge.net/project/libpng/libpng16/older-releases/1.6.7/libpng-1.6.7.tar.gz > libpng-1.6.7.tar.gz && \

  tar -xzf libpng-1.6.7.tar.gz &&

  rm libpng-1.6.7.tar.gz && \

  mkdir libpng && \

  cd libpng-1.6.7 && \

  ./configure --prefix=/libpng && \

  make && \

  make install && \

  cd / && \

  rm -r /libpng-1.6.7

Run notes

- Multiple command can be combined
  - These then count as one layer (out of 128 max)
- Temporary files must be removed in same layer as used or they stay in the image
  - Next image builds on previous
- cd (change directory) only effects that layer
  - Each new layer starts in home
- export only effects that layer
  - See ENV command

ENTRYPOINT and CMD

- Command to run then the image is run
- There can only be one of each
  - Earlier ones are ignored
- Both are optional and independent
- Various different formats possible

- Example:

  ENTRYPOINT ["python","calculator.py"]

  CMD ["1","+","2","*","3"]

  - Runs "python calculator.py 1+2*3

ENTRYPOINT

- Command part expected to be used every time
- Makes the image an executable file

- If docker run is provided arguments the ENTRYPOINT commands are still included

- Can be ignored with the docker run flag --entrypoint

CMD

- Default arguments for Docker run
- Ignored if any arguments are provided when docker images is run

docker run –rm brenninc/calculator 4+5*2

4+5*2 = 14

docker run –rm brenninc/calculator

1 + 2 * 3 = 7

EXPOSE

- “informs Docker that the container will listen on the specified network ports at runtime”
- Connects ports of any application/ service to be run to the outside of the docker
- Note requires the -p flag at runtime to expose it from docker to the host

ENV

- Sets Key value environment variable
- Persist on all future layers and runtime
- Can be overwritten

- ENV myName John Doe
- ENV myDog Rex The Dog
- ENV myCat fluffy

- ENV myName="John Doe" myDog=Rex\ The\ Dog \
- myCat=fluffy

COPY

- COPY source destination
- Copies local files or directories into the docker image
- Source must be in the same context as the Dockerfile
  - Files in the same context as the Docker file are only available in the image if copied in
- Multiple sources can be specified but then destination must be a folder

- COPY calculator.py calculator.py

ADD

- Similar to COPY but with extra functionality
  - Docker recommends using COPY when possible
- If source is a local tar archive in a recognized compression format (identity, gzip, bzip2 or xz) then it is unpacked as a directory.
- ADD can add data from URLS
  - Never unpacked

VOLUME

- Creates a mount point
  - Creates a directory in /var/lib/docker/volumes/
  - With a random name
- Used by containers that save data
  - Example Ipython
- Similar to the -v flag in docker run image
- Directory created when a container is created are not removed even if the container is

USER

- Allow you to run image as other than root user
- User must be created

WORKDIR

- Sets the working directory
- Should be an absolute directory
  - Absolute within docker image not the host
- Unlike cd persists between layers

ONBUILD

- Used in images that will be parents to other images
- Adds instructions to run then child image builds

- example

  RUN mkdir -p /usr/src/app

  WORKDIR /usr/src/app

  ONBUILD COPY Gemfile /usr/src/app/

  ONBUILD COPY Gemfile.lock /usr/src/app/

  ONBUILD RUN bundle install

  ONBUILD COPY . /usr/src/ap

Calculator Dockerfile

- FROM ubuntu:14.04
- MAINTAINER Christian Brenninkmeijer <Christian.Brenninkmeijer@manchester.ac.uk>
- LABEL "description"="An example docker app using python as a calculator"
- #Install python via apt-get
- RUN apt-get update && apt-get install -y python
- #copy in the code
- COPY calculator.py calculator.py
- ENTRYPOINT ["python","calculator.py"]
- CMD ["1","+","2","*","3"]

Calculator.py

- import parser
- import sys

- command = " ".join(sys.argv[1:])
- st = parser.expr(command)
- code = st.compile('file.py')
- print command,"=",eval(code)

Build Calculator

- docker build -t brenninc/calculator .

-t provides a tag (name) for your image

- Docker will reuse existing images layers wherever this is possible
- Automatically detecting the first layer that changed
  - Including if a file copied in has changed
- All subsequent layers are built

SHIP Calculator

- Source file uploaded to:https://github.com/brenninc/calculator
  - Dockerfile
  - Calculator.py
- Linked to:https://hub.docker.com/r/brenninc/calculator/
- Automatically built image (by docker hub)
- docker pull brenninc/calculator