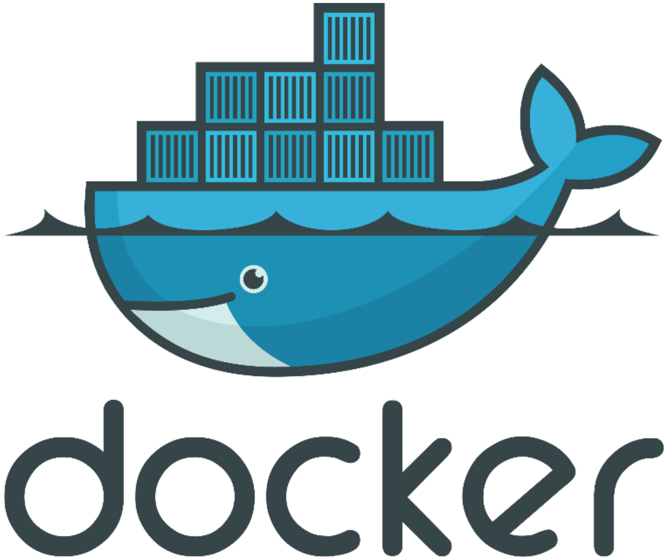
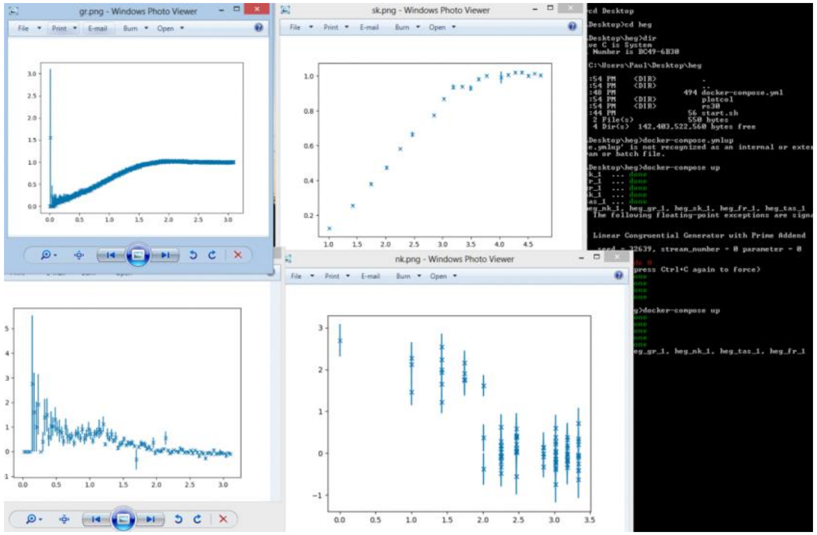
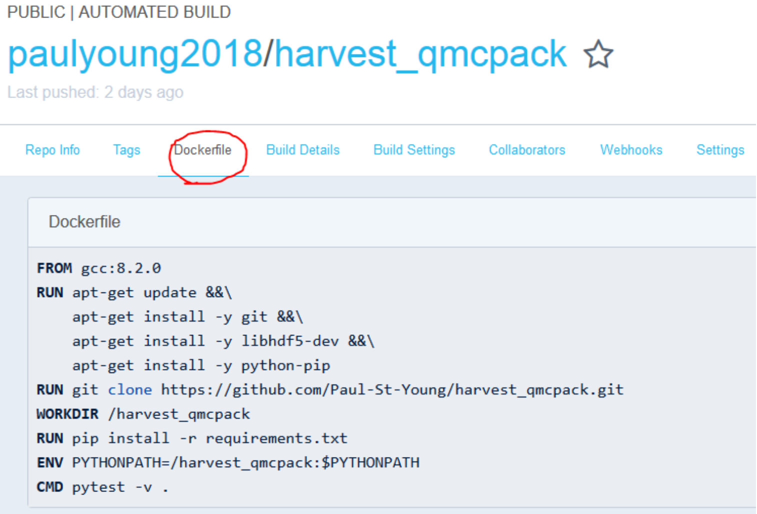
Docker Tutorial

Yubo “Paul” Yang, THW-IL, 2018/12/05




gr.png - Windows Photo Viewer
sk.png - Windows Photo Viewer
nk.png - Windows Photo Viewer
File
Print
E-mail
Burn
Open

```
cd Desktop

Desktop>cd heg

Desktop\heg>dir
ve C is System
 Number is BC49-6830

C:\Users\Paul\Desktop\heg

:54 PM    <DIR>          .
:54 PM    <DIR>          ..
:48 PM               494 docker-compose.yml
:54 PM    <DIR>          plotcol
:54 PM    <DIR>          rs30
:44 PM                56 start.sh
 2 File(s)           550 bytes
 4 Dir(s)  142,403,522,560 bytes free

Desktop\heg>docker-compose.ymlup
e.ymlup' is not recognized as an internal or exte
am or batch file.

Desktop\heg>docker-compose up
k_1   ... done
r_1   ... done
r_1   ... done
k_1   ... done
as_1  ... done
eg_nk_1, heg_gr_1, heg_sk_1, heg_fr_1, heg_tas_1
 The following floating-point exceptions are signa

 Linear Congruential Generator with Prime Addend

  seed = 22639, stream_number = 0 parameter = 0

press Ctrl+C again to force)
one
one
one
one

g>docker-compose up
one
one
one
one
one
eg_gr_1, heg_nk_1, heg_tas_1, heg_fr_1
```



PUBLIC | AUTOMATED BUILD

paulyoung2018/harvest_qmcpack ☆

Last pushed: 2 days ago

Dockerfile

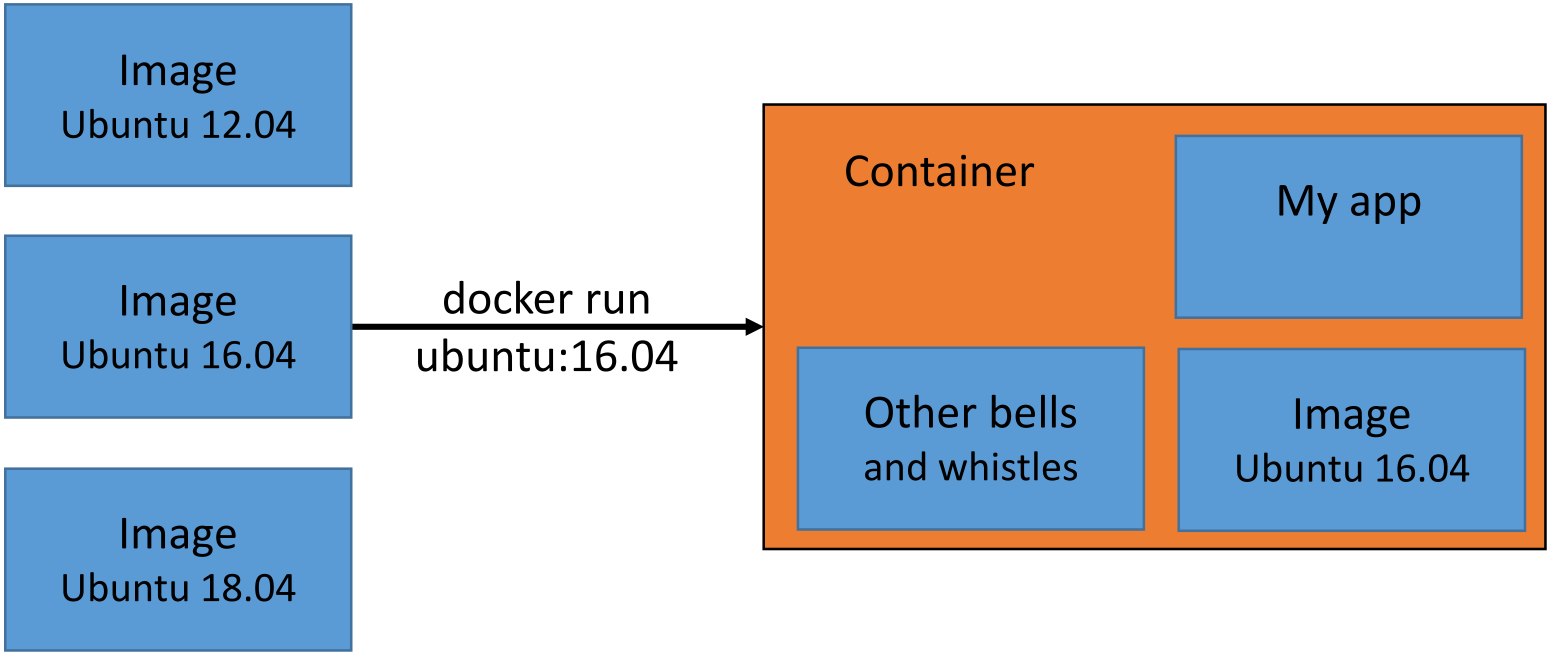
```
FROM gcc:8.2.0
RUN apt-get update &&\
    apt-get install -y git &&\
    apt-get install -y libhdf5-dev &&\
    apt-get install -y python-pip
RUN git clone https://github.com/Paul-St-Young/harvest_qmcpack.git
WORKDIR /harvest_qmcpack
RUN pip install -r requirements.txt
ENV PYTHONPATH=/harvest_qmcpack:$PYTHONPATH
CMD pytest -v .
```

What is Docker?

Docker is *conceptually* a virtual machine image manager.

All dependencies are included in a single portable container.

Build in layers: modular and extendable.
(Like initial laptop setup. Only minimal, automated, and repeatable.)



Why use Docker?

Docker cleanses us of our sins and prepares us for the **heavens**

Photo by
Christopher Burns
on Unsplash

- Break out of dependency hell

- Easy deployment to the cloud

Photo by
Jeremy Perkins
on Unsplash

How to use Docker?

Level 0: Running one container – “docker –i –t –p –v”

Level 1: Building one image – “dockerfile”

Level 2: Modify container and image – “docker exec, commit”

Level 3: Using multiply containers – “docker-compose up”

Level 4: Run your application on the cloud – “docker swarm” & ECS on AWS

Great reference: PyCon 2016 slides
https://us.pycon.org/2016/site_media/media/tutorial_handouts/DockerSlides.pdf

Reference bookmarks

Great reference: PyCon 2016 slides
https://us.pycon.org/2016/site_media/media/tutorial_handouts/DockerSlides.pdf

docker ps	p 73
docker logs	p 79
^P^Q then docker attach	p 84
docker images and containers	p 90
docker commit and tag	p107
docker build and tag – figlet example	p125
find IP of container	p173
ping container	p183
compose	p202
write better dockerfile	p272-p296

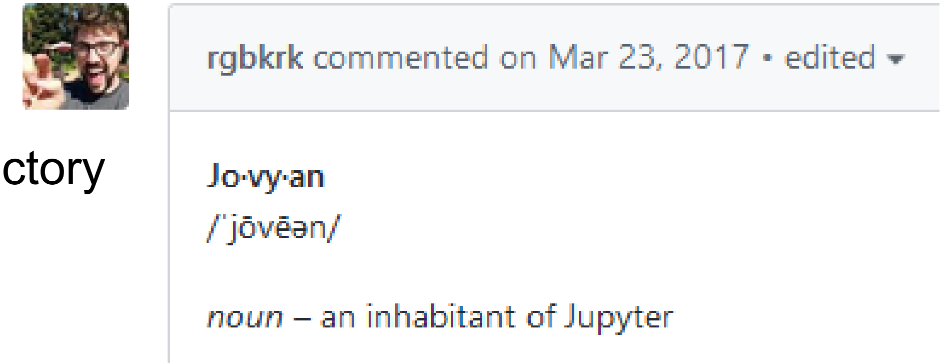
Level 0: Run one container

Ubuntu base build: “-i” connect to standard-in; “-t” open pseudo terminal

```
docker run –it ubuntu
# echo “hello from Ubuntu”
```

Web server: “-p” pass host port to container port i.e. [host port]:[container port]

```
docker run –p 200:80 nginx
firefox localhost:200
```



rgbkrk commented on Mar 23, 2017 • edited

Jo·vy·an
/ˈjōvēən/

noun – an inhabitant of Jupyter

Isolated Dev. Env.: “-v” mounts a local directory

```
docker run –v /soft/myapp/src:/src –it gcc
# cd /src; make
```

Do Data Science Anywhere:

```
docker run –v /data:/data –p 8888:8888 – – name mynb jupyter/datascience–notebook
firefox localhost:8888/?token=[really long token for security]
docker cp /data/mybigdata.json mynb:/home/jovyan/work
```

Level 1: Build one container

docker build and tag – figlet example p125

Introducing: “dockerfile”

A list of commands to build your image.
(preferably from an official image as base)

```
FROM ubuntu
RUN apt-get update
RUN apt-get install –y figlet
```

Build the image: the natural way

```
docker build .
```

```
docker image ls
```

```
REPOSITORY          TAG          IMAGE ID          CREATED              SIZE
<none>              <none>       1b402d6bbb5e      About a minute ago   112MB
```

```
docker tag 1b402 figlet
```

Build the image: the right way

```
docker build –t figlet .
```

Level 1: Build one container

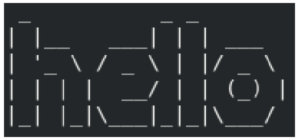
docker build and tag – figlet example

p125

Use interactively

```
FROM ubuntu
RUN apt-get update
RUN apt-get install –y figlet
```

```
docker run –it figlet
# figlet hello
```

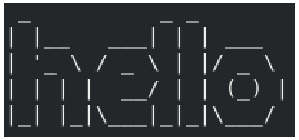


```
 _          _ _       
| |__   ___| | | ___  
| '_ \ / _ \ | |/ _ \ 
| | | |  __/ | | (_) |
|_| |_|\___|_|_|\___/ 
```

Single-purpose application: CMD

```
FROM ubuntu
RUN apt-get update
RUN apt-get install –y figlet
CMD figlet hello
```

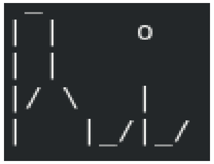
```
docker run figlet
```



```
 _          _ _       
| |__   ___| | | ___  
| '_ \ / _ \ | |/ _ \ 
| | | |  __/ | | (_) |
|_| |_|\___|_|_|\___/ 
```

override

```
docker run figlet figlet –f script hi
```

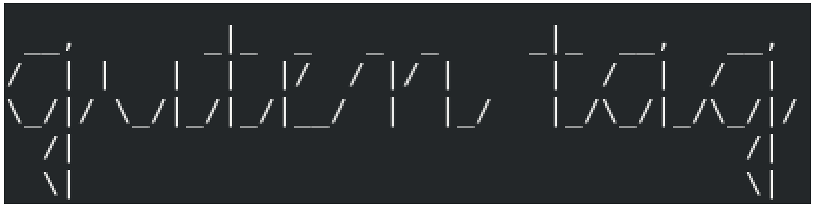


```
 _
| |    o
| |
|/ \   |
|   |_/|_/
```

Multi-purpose application: ENTRYPOINT

```
FROM ubuntu
RUN apt-get update
RUN apt-get install –y figlet
ENTRYPOINT [“figlet”, “-f”, “script”]
```

```
docker run figlet guten tag
```



```
            |                 |
 __,       _|_  _   _  _     _|_  __,   __,
/  | |   |  |  |/  / |/ |     |  /  |  /  |
\_/|/ \_/|_/|_/|__/  |  |_/   |_/\_/|_/\_/|/
  /|                                     /|
  \|                                     \|
```

Level 1: Build one container

docker build and tag – figlet example p125

Publish container

```
docker tag figlet paulyoung2018/figlet:script
docker login --user paulyoung2018
docker push paulyoung2018/figlet:script
```

Tell your friends!

Run from anywhere

```
[not me]$ docker run paulyoung2018/figlet:script wuwuwuwuuuut!
```

Pull to update

```
[not me]$ docker pull paulyoung2018/figlet:script
```

Level 2: Modify container and image

Log in to a container: the natural way

```
docker run –it ––entrypoint=bash figlet
```

start a bash window in a new container

```
docker ps
```

list all running containers

```
CONTAINER ID      IMAGE            COMMAND              CREATED          STATUS           PORTS              NAMES
295aa35eaa82      figlet          "bash"            About a minute ago   Up About a minute
```

```
docker rename 295aa myfig
```

rename container to something readable

```
docker exec –it myfig bash
```

start a new bash window

Log in to a container: the right way

```
docker run –it ––name myfig ––entrypoint=bash figlet
```

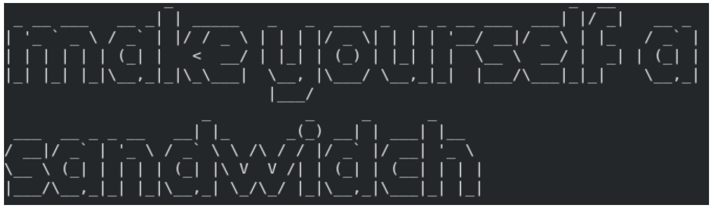
name container at start

Pop quiz: How to let figlet take a file’s content as input?

Level 2: Modify container and image

Pop quiz solution: mount host file; then use bash

```
echo “make me a sandwidch” > /tmp/figlet.in
docker run –v /tmp/figlet.in:/home/figlet.in figlet /bin/bash –c “cat
/home/figlet.in | sed ‘s/me/yourself/’ | figlet”
```



Modify ENTRYPOINT (-c or --change)

```
docker commit –c ‘ENTRYPOINT /bin/bash –c “cat /home/figlet.in | figlet”’ myfig
```

```
sha256:773ec79ad35a17c25b38cabf968c61915d02c605793394aa0d76be38047bd167
```

```
docker tag 773ec figlet:file
```

Revert modification

```
docker history figlet:file
```

```
IMAGE           CREATED             CREATED BY                                      SIZE          COMMENT
773ec79ad35a      4 minutes ago                                                    0B
ffe572d63dcf     About an hour ago   /bin/sh -c #(nop)  ENTRYPOINT ["figlet" "-f"...   0B
a32c5b019c62      About an hour ago   /bin/sh -c apt-get update && apt-get install...   25.4MB
```

```
docker tag ffe572d63dcf figlet:file
```

Level 2: Modify container and image

Publish new image

```
docker tag figlet:file paulyoung2018/figlet:file
```

```
docker push paulyoung2018/figlet:file
```

figlet:file is modified from figlet:script, so most layers are reused.

For more best practices regarding dockerfile. See official docs and

Great reference: PyCon 2016 slides
https://us.pycon.org/2016/site_media/media/tutorial_handouts/DockerSlides.pdf

write better dockerfile p272-p296

Level 3: Using multiple containers

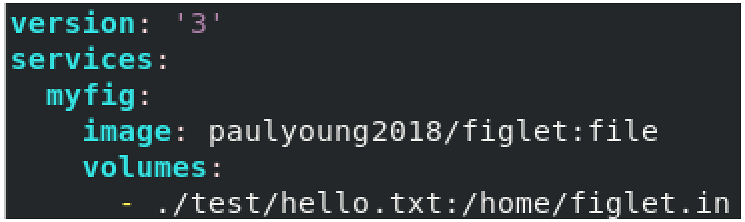
Introducing: “docker-compose.yml”

A list of containers and options to run.

Q/ How to do X with docker-compose?
A/ Use bash script and the docker API.

Basic usage 1: put docker API flags down in a file

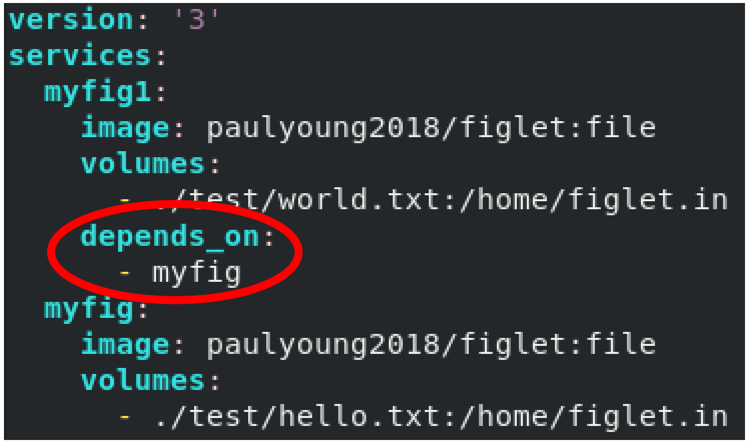
```
echo “make me a sandwidch” > /tmp/figlet.in
docker run −v /tmp/figlet.in:/home/figlet.in − − name myfig figlet
```



```
version: '3'
services:
  myfig:
    image: paulyoung2018/figlet:file
    volumes:
      - ./test/hello.txt:/home/figlet.in
```

Level 3: Using multiple containers

Basic usage 2: start multiple containers in order



```
version: '3'
services:
  myfig1:
    image: paulyoung2018/figlet:file
    volumes:
      - ./test/world.txt:/home/figlet.in
    depends_on:
      - myfig
  myfig:
    image: paulyoung2018/figlet:file
    volumes:
      - ./test/hello.txt:/home/figlet.in
```

More common useful examples compose:
Communicate between web front end interfaces and backend service

e.g. website, dockercoin, voting app

Level 4: Deploy applications to the cloud

Docker swarm: takes a cluster of machines and make them look like one big machine.

Kubernetes (predates docker swarm) is Google’s equivalent of docker swarm.

Elastic Container Services (ECS): scales the number of running containers & Amazon cloud servers based on load. “ecs-cli” is the command line tool.

Practical Example 1: Quantum Monte Carlo + Monitoring

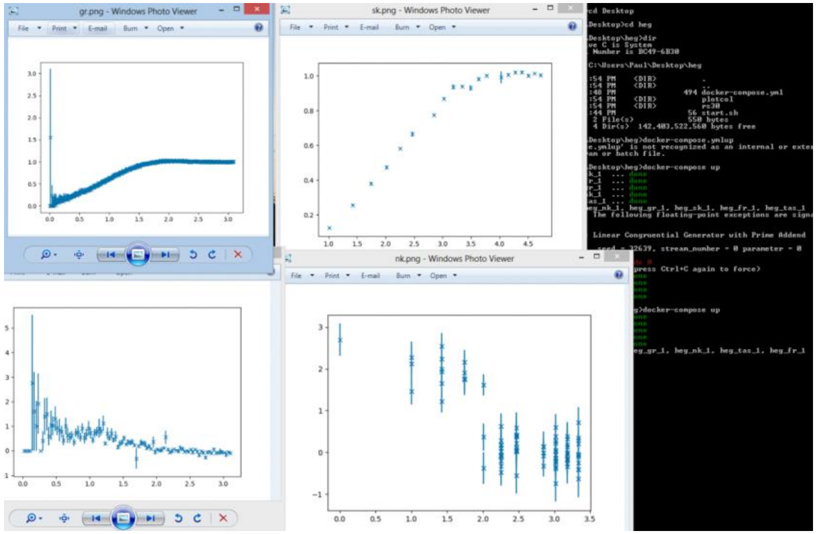
Step 1: compile code

Step 2: make image with reasonable entrypoint

Step 3: write analysis code

Step 4: write docker-compose.yml to orchestrate run and analysis

Step 5: go ham!


gr.png - Windows Photo Viewer
sk.png - Windows Photo Viewer
nk.png - Windows Photo Viewer
File
Print
E-mail
Burn
Open
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
0.2
0.4
0.6
0.8
−1
0
1
2
3
4
5

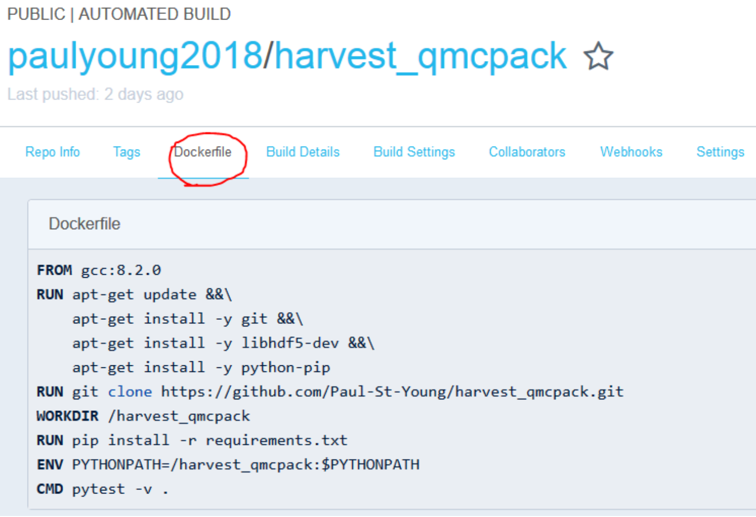
```
cd Desktop

Desktop>cd heg

Desktop\heg>dir
ve C is System
 Number is BC49-6B30

 C:\Users\Paul\Desktop\heg

:54 PM    <DIR>          .
:54 PM    <DIR>          ..
:48 PM               494 docker-compose.yml
:54 PM    <DIR>          plotcol
:54 PM    <DIR>          rs30
:44 PM                56 start.sh
 2 File(s)            550 bytes
 4 Dir(s)  142,403,522,560 bytes free

Desktop\heg>docker-compose.ymlup
e.ymlup' is not recognized as an internal or exte
am or batch file.

Desktop\heg>docker-compose up
k_1  ... done
r_1  ... done
r_1  ... done
k_1  ... done
as_1 ... done
eg_nk_1, heg_gr_1, heg_sk_1, heg_fr_1, heg_tas_1
 The following floating-point exceptions are sign

 Linear Congruential Generator with Prime Addend

  seed = 32639, stream_number = 0 parameter = 0

press Ctrl+C again to force)
one
one
one
one

g>docker-compose up
one
one
one
one
one
eg_gr_1, heg_nk_1, heg_tas_1, heg_fr_1
```

Practical Example 2: Make your GitHub repo. eternal

Step 1: add Dockerfile to your repo

Step 2: sell your soul to Docker Hub

Step 3: wait for your automatically built image!



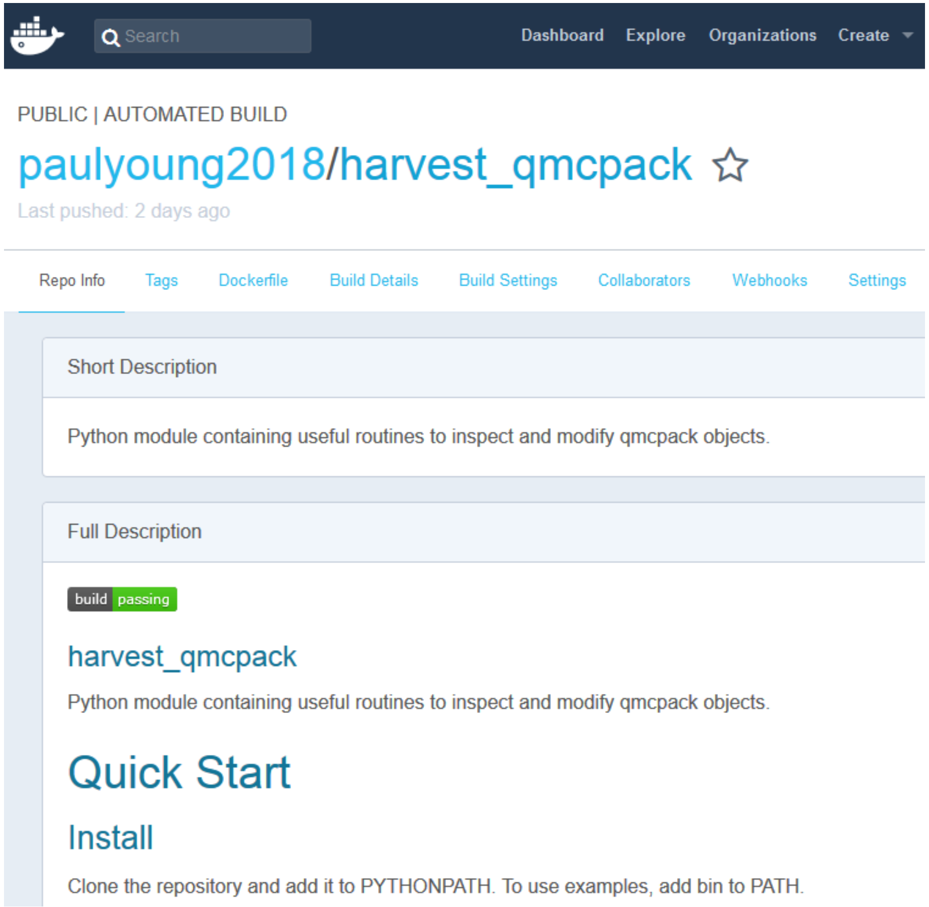
PUBLIC | AUTOMATED BUILD

paulyoung2018/harvest_qmcpack ☆

Last pushed: 2 days ago

Dockerfile

```
FROM gcc:8.2.0
RUN apt-get update &&\
    apt-get install -y git &&\
    apt-get install -y libhdf5-dev &&\
    apt-get install -y python-pip
RUN git clone https://github.com/Paul-St-Young/harvest_qmcpack.git
WORKDIR /harvest_qmcpack
RUN pip install -r requirements.txt
ENV PYTHONPATH=/harvest_qmcpack:$PYTHONPATH
CMD pytest -v .
```



PUBLIC | AUTOMATED BUILD

paulyoung2018/harvest_qmcpack ☆

Last pushed: 2 days ago

Short Description

Python module containing useful routines to inspect and modify qmcpack objects.

Full Description

harvest_qmcpack

Python module containing useful routines to inspect and modify qmcpack objects.

Quick Start

Install

Clone the repository and add it to PYTHONPATH. To use examples, add bin to PATH.

Last but not least: save your SSD

Docker keeps stopped containers and intermediate images on disk.
In English, Moby Dock will gobble up your disk like krill.

Solution is simple:

```
docker container prune
docker image prune
```

All named containers and tagged images will be left alone.

You may wish to do manual clean up on a production server.

Conclusion:

Docker container = *future of* virtual machine instance

Docker image = *futher of* virtual machine snapshot

Containerized applications require no setup on any platform.
Growing support on:

✓ Amazon Web Service (AWS)
✓ Microsoft Azure Cloud
✓ Google Cloud (Kubernets)

□ Run a single container using the docker API

```
docker run -it -p [host port]:[device port] --name me [owner/image:version]
```

□ Manage multiple containers using docker-compose.yml

```
docker-compose up
```

□ Write dockerfile to eternalize your GitHub repo.

□ Modify containers with exec and commit

□ Tag modified containers into images

□ Publish and be forever!