DOCKER 101

Charlie Dey, Director of Training and Professional Development
Marjo Poindexter, Cloud and API developer
Ritu Arora, Research Scientist

WHAT IS DOCKER?

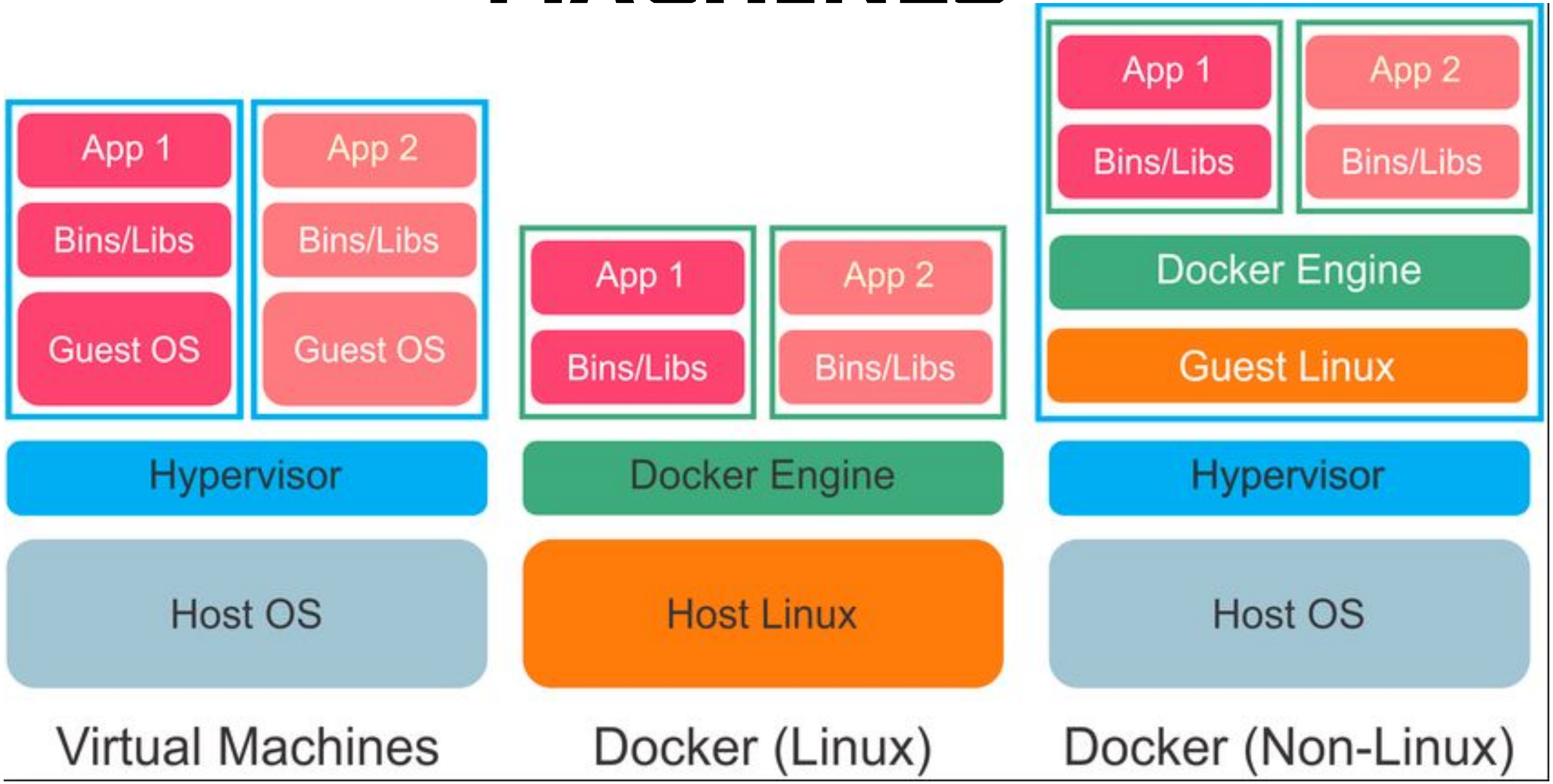
- An open-source project that can be used for creating, deploying, and running applications as containers

- These **containers** are portable, self-sufficient and can be run on any infrastructure in the cloud or on-premises on which Docker is installed/supported

WHAT IS A CONTAINER?

- Containers are used to package an application along with all its dependencies (such as runtime system, libraries, and data) into a single unit

CONTAINER VERSUS VIRTUAL MACHINES



Source: GUIdock: Using Docker Containers with a Common Graphics User Interface to Address the Reproducibility of Research - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/A-comparison-of-the-architecture-of-virtual-machines-and-Docker-software_fig4_299771559 [accessed 24 Sep, 2018]

WHY USE DOCKER?

- Helps in **deploying future-proof applications** by creating packages that are almost self-contained

- Makes it **convenient to distribute production and trial versions of the code** that can run on the customers' devices without requiring an application-specific installation and configuration, hence, helps in developing **a scalable software distribution model**

- **Saves time in complicated installs** - build a Docker application - push it to Docker Hub - and **reuse** it on any number of systems as you desire

- **Mitigates the portability issues related to the applications** - Dockerized applications can be ported conveniently across different cloud computing service providers - of course, you may need to install Docker and additional tools to run the Docker container depending upon the system that you are on

- **Helps in doing reproducible science**

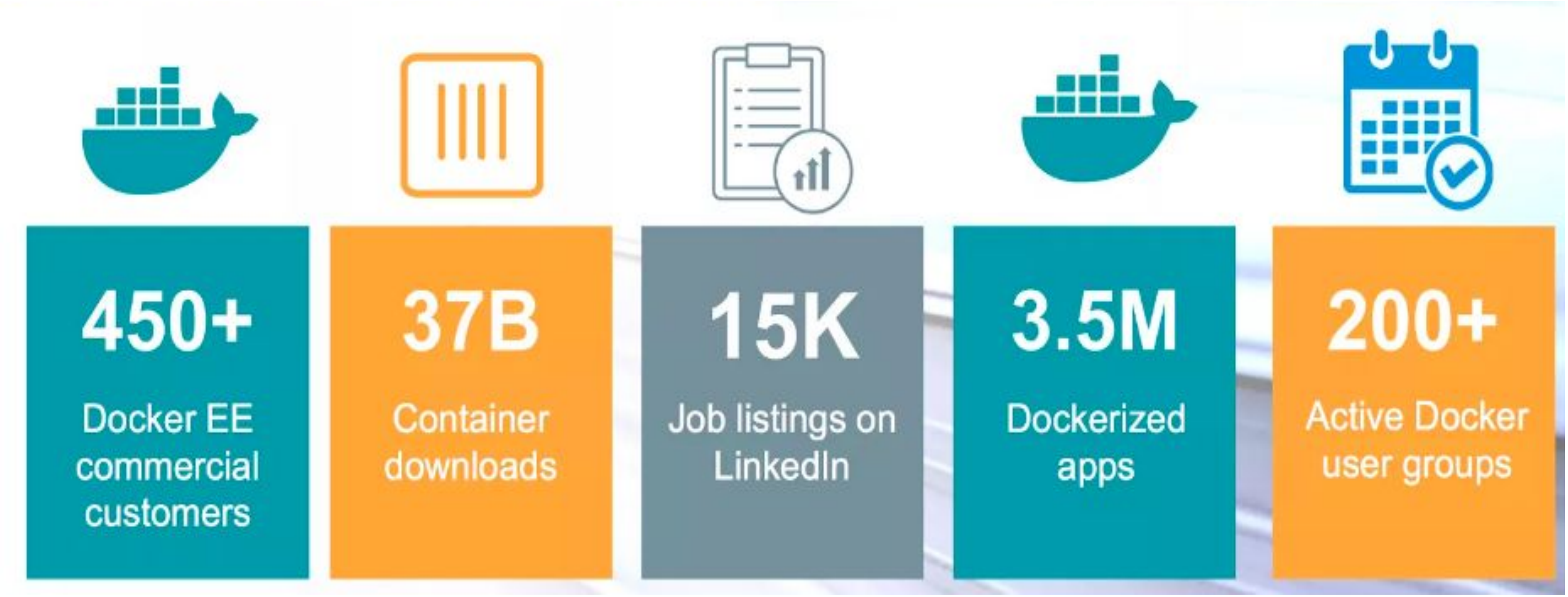
WHAT IS THE PROCESS FOR BUILDING DOCKER IMAGES?

1. Install Docker
2. Write a Dockerfile
   + **Dockerfile** is a text file that contains instructions, using which, Docker can build images automatically
   + Name of the file: Dockerfile
3. Build the Docker image and tag it
   + A file containing the snapshot of a container is known as a Docker image
   + It is created using the build command, and produces a container when it starts running
   + You can add tags to the images during the build step or while saving it to a repository - the default tag is "latest"
4. Run the image that you built
5. Register on DockerHub - use the credentials to push the image
   + Images are stored in a Docker registry such as registry.hub.docker.com
   + Pull Images later on any system that you want to run the application on

HOW CAN CONTAINERS BE CREATED FROM DOCKER IMAGES?

- You can use Docker commands such as “docker pull” or “docker run”
  - “Pull” will always fetch the latest version of the image from Docker Hub before running
  - “Run” will search for an image locally and run it, and if there is nothing available\ locally, it will go to Docker Hub.
- We will learn more about this topic during the hands-on session

HOW POPULAR IS DOCKER?



450+
Docker EE commercial customers

37B
Container downloads

15K
Job listings on LinkedIn

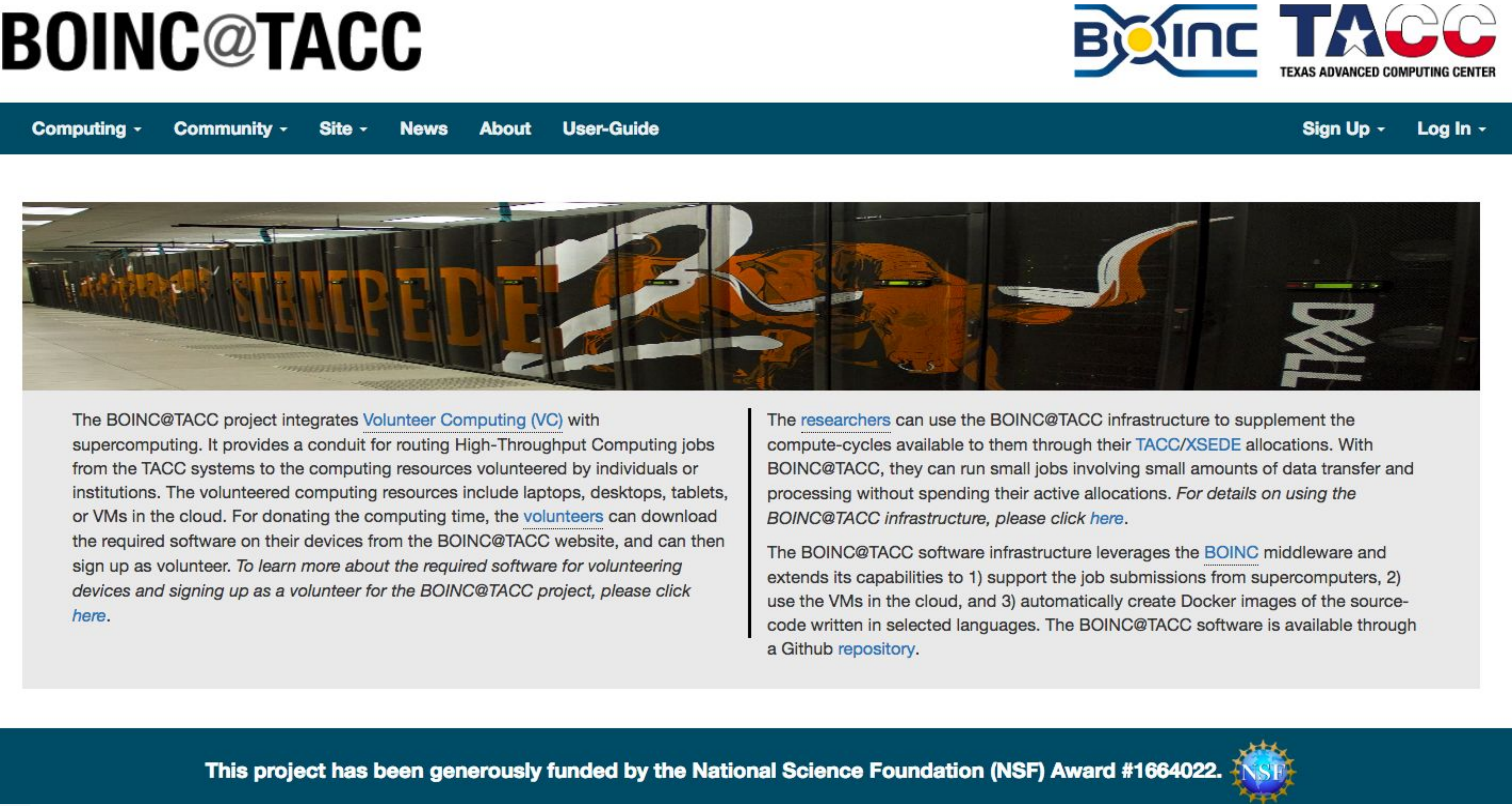
3.5M
Dockerized apps

200+
Active Docker user groups

PHOTO CREDIT: DAVID MESSINA, DOCKER INC.

Source: https://blog.docker.com/2018/03/5-years-later-docker-journey/ [accessed 24 Sep, 2018]

HOW ARE WE USING DOCKER? (1)



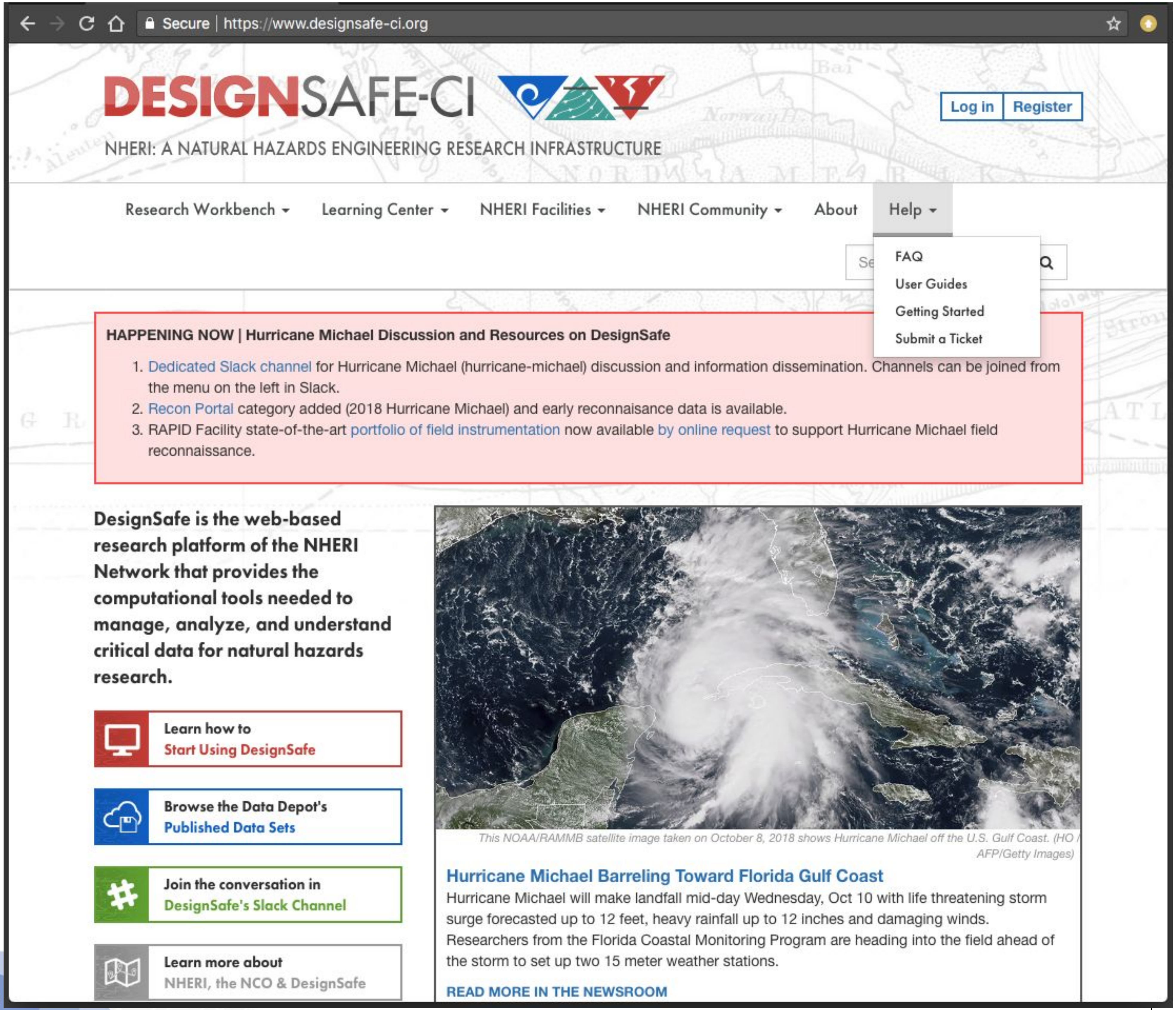
The BOINC@TACC project integrates Volunteer Computing (VC) with supercomputing. It provides a conduit for routing High-Throughput Computing jobs from the TACC systems to the computing resources volunteered by individuals or institutions. The volunteered computing resources include laptops, desktops, tablets, or VMs in the cloud. For donating the computing time, the volunteers can download the required software on their devices from the BOINC@TACC website, and can then sign up as volunteer. *To learn more about the required software for volunteering devices and signing up as a volunteer for the BOINC@TACC project, please click here*.

The researchers can use the BOINC@TACC infrastructure to supplement the compute-cycles available to them through their TACC/XSEDE allocations. With BOINC@TACC, they can run small jobs involving small amounts of data transfer and processing without spending their active allocations. *For details on using the BOINC@TACC infrastructure, please click here*.

The BOINC@TACC software infrastructure leverages the BOINC middleware and extends its capabilities to 1) support the job submissions from supercomputers, 2) use the VMs in the cloud, and 3) automatically create Docker images of the source-code written in selected languages. The BOINC@TACC software is available through a Github repository.

This project has been generously funded by the National Science Foundation (NSF) Award #1664022.

HOW ARE WE USING DOCKER? (1)



HAPPENING NOW | Hurricane Michael Discussion and Resources on DesignSafe

1. Dedicated Slack channel for Hurricane Michael (hurricane-michael) discussion and information dissemination. Channels can be joined from the menu on the left in Slack.
2. Recon Portal category added (2018 Hurricane Michael) and early reconnaisance data is available.
3. RAPID Facility state-of-the-art portfolio of field instrumentation now available by online request to support Hurricane Michael field reconnaissance.

DesignSafe is the web-based research platform of the NHERI Network that provides the computational tools needed to manage, analyze, and understand critical data for natural hazards research.

Learn how to
Start Using DesignSafe

Browse the Data Depot's
Published Data Sets

Join the conversation in
DesignSafe's Slack Channel

Learn more about
NHERI, the NCO & DesignSafe

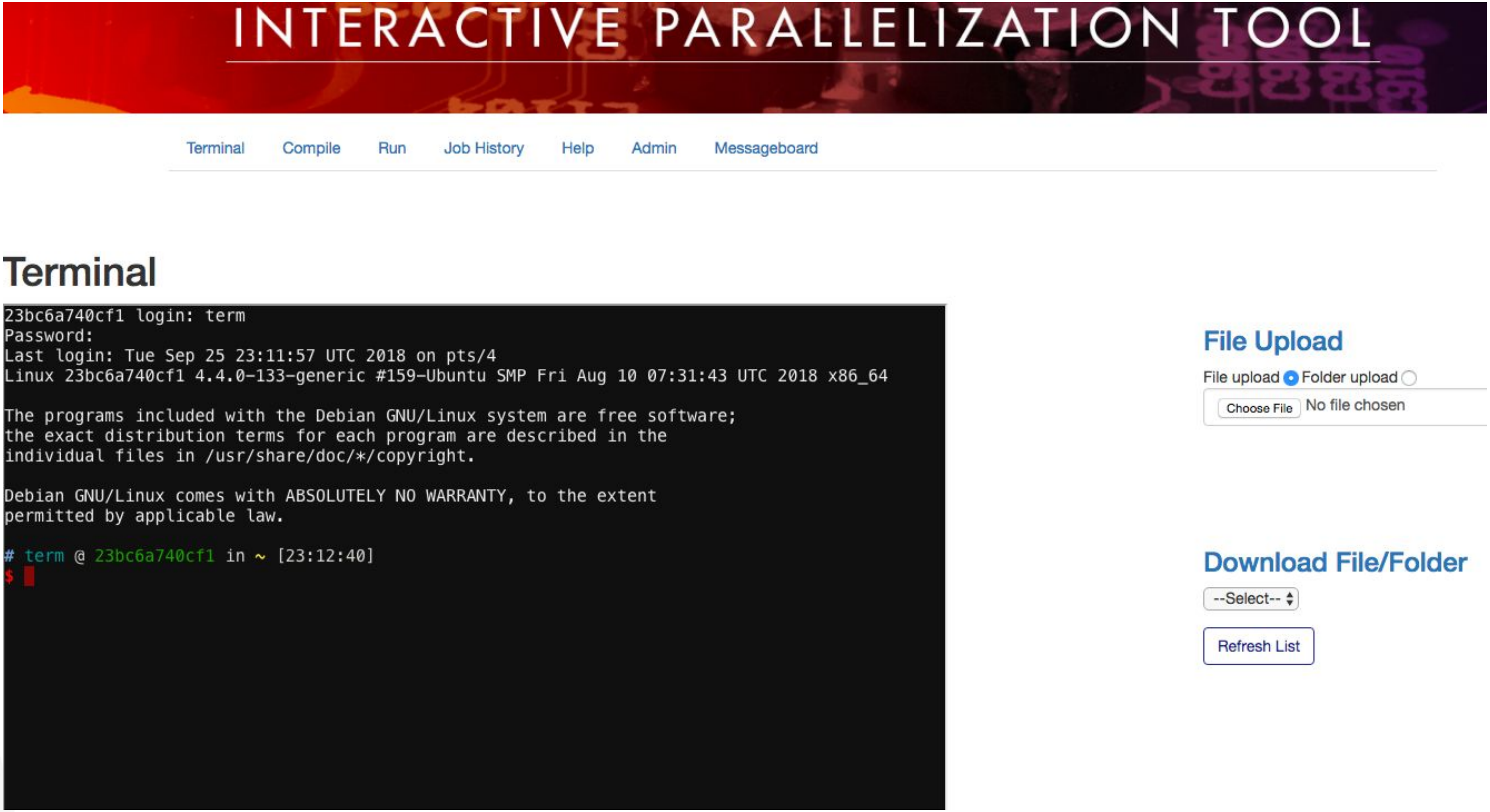
This NOAA/RAMMB satellite image taken on October 8, 2018 shows Hurricane Michael off the U.S. Gulf Coast. (HO / AFP/Getty Images)

Hurricane Michael Barreling Toward Florida Gulf Coast

Hurricane Michael will make landfall mid-day Wednesday, Oct 10 with life threatening storm surge forecasted up to 12 feet, heavy rainfall up to 12 inches and damaging winds. Researchers from the Florida Coastal Monitoring Program are heading into the field ahead of the storm to set up two 15 meter weather stations.

READ MORE IN THE NEWSROOM

HOW ARE WE USING DOCKER? (2)



```
23bc6a740cf1 login: term
Password:
Last login: Tue Sep 25 23:11:57 UTC 2018 on pts/4
Linux 23bc6a740cf1 4.4.0-133-generic #159-Ubuntu SMP Fri Aug 10 07:31:43 UTC 2018 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

# term @ 23bc6a740cf1 in ~ [23:12:40]
$ 
```

Hands-On Session

STEP 0: SET-UP

Create a Docker ID: https://docs.docker.com/docker-id/#log-in

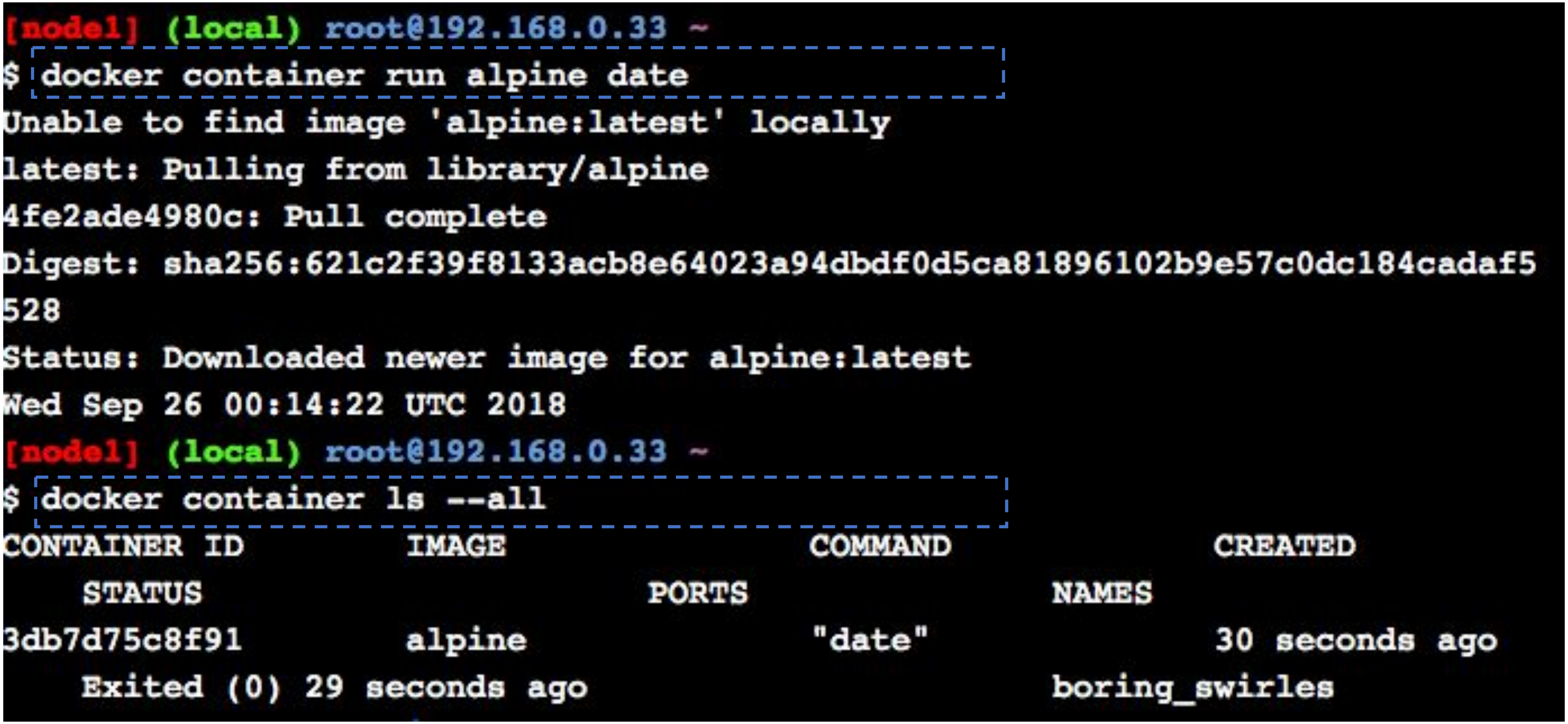
Pick one of the following options for proceeding:

Option # 1: Use a VM on a cloud computing system named Jetstream - please see the login details on the sheets handed out and also the next slide

Option # 2: Use Docker's online platform for learning about Docker: https://tinyurl.com/y9sg7eq7

STEP 1: RUNNING PRE-BUILT DOCKER CONTAINERS (1)

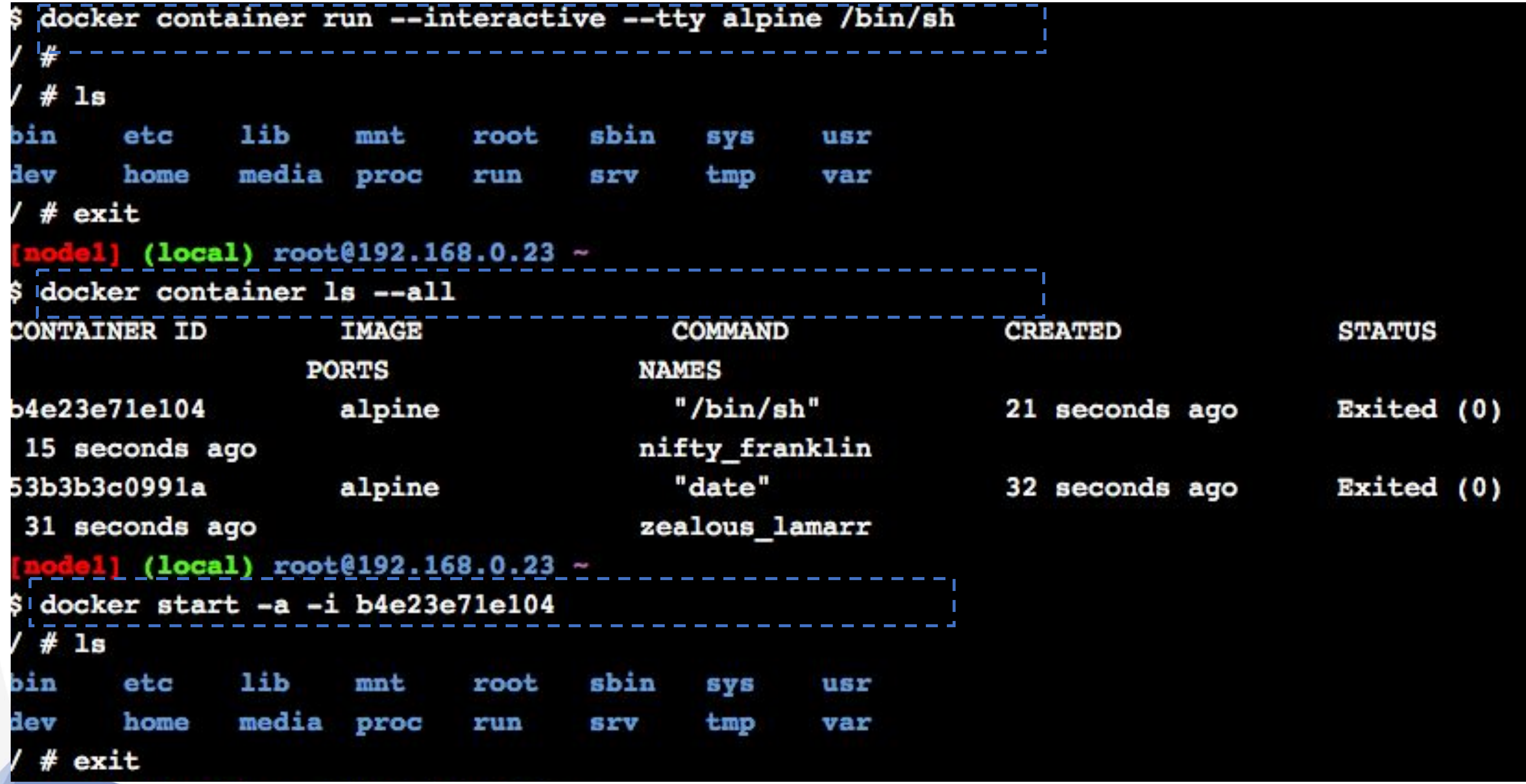
Please go to "Step 1" at the following link: https://tinyurl.com/yatewkx2



```
[node1] (local) root@192.168.0.33 ~
$ docker container run alpine date
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
4fe2ade4980c: Pull complete
Digest: sha256:621c2f39f8133acb8e64023a94dbdf0d5ca81896102b9e57c0dc184cadaf5
528
Status: Downloaded newer image for alpine:latest
Wed Sep 26 00:14:22 UTC 2018
[node1] (local) root@192.168.0.33 ~
$ docker container ls --all
CONTAINER ID        IMAGE               COMMAND             CREATED
    STATUS                      PORTS               NAMES
3db7d75c8f91        alpine              "date"              30 seconds ago
    Exited (0) 29 seconds ago                       boring_swirles
```

STEP 1: RUNNING PRE-BUILT DOCKER CONTAINERS (2)

Please go to "Step 1" at the following link:

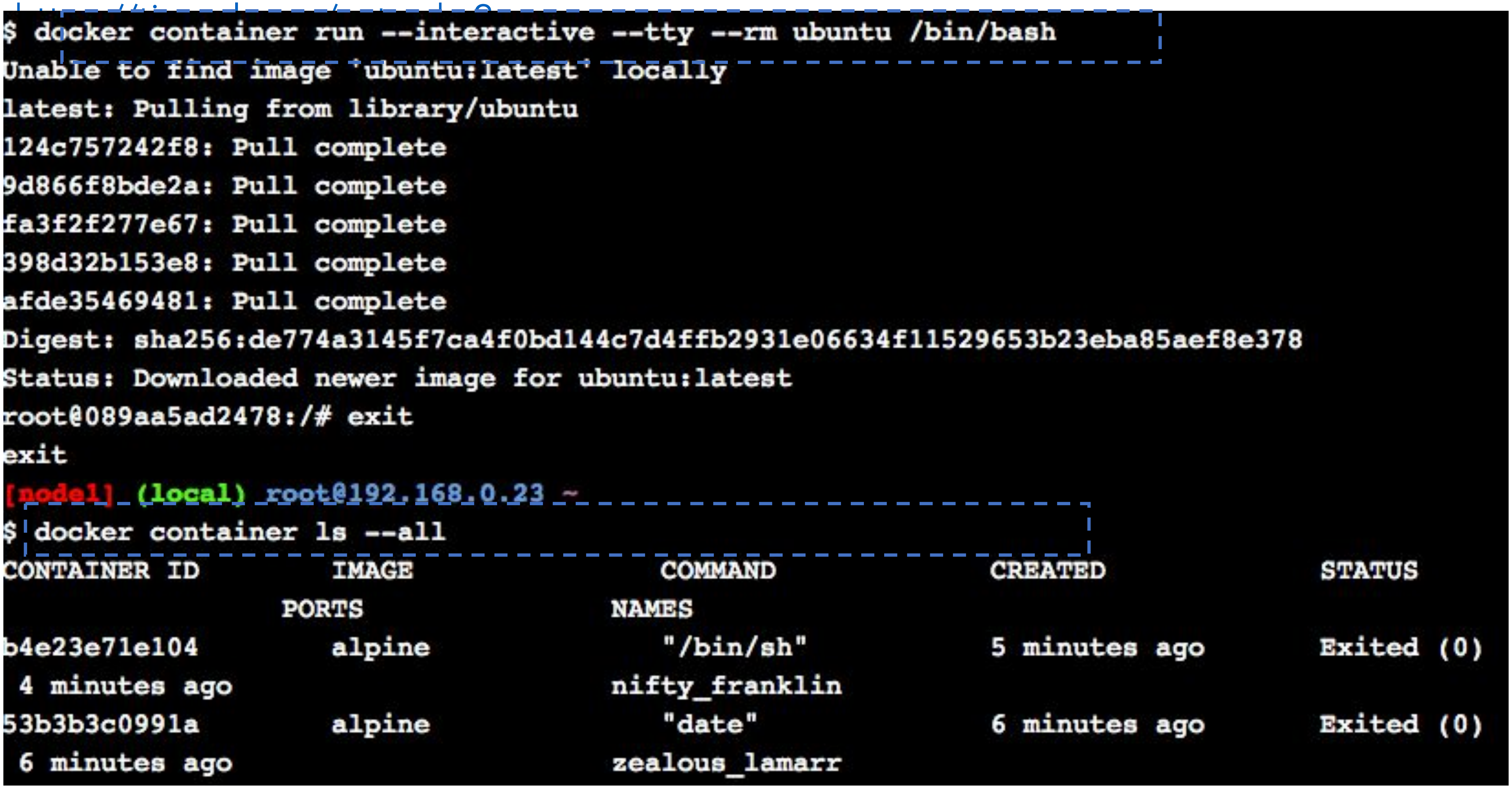


```
$ docker container run --interactive --tty alpine /bin/sh
/ #
/ # ls
bin     etc     lib     mnt     root    sbin    sys     usr
dev     home    media   proc    run     srv     tmp     var
/ # exit
[node1] (local) root@192.168.0.23 ~
$ docker container ls --all
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
            PORTS               NAMES
b4e23e71e104        alpine              "/bin/sh"           21 seconds ago      Exited (0)
 15 seconds ago                     nifty_franklin
53b3b3c0991a        alpine              "date"              32 seconds ago      Exited (0)
 31 seconds ago                     zealous_lamarr
[node1] (local) root@192.168.0.23 ~
$ docker start -a -i b4e23e71e104
/ # ls
bin     etc     lib     mnt     root    sbin    sys     usr
dev     home    media   proc    run     srv     tmp     var
/ # exit
```

STEP 1: RUNNING PRE-BUILT DOCKER CONTAINERS (3)

Please go to “Step 1” at the following link:

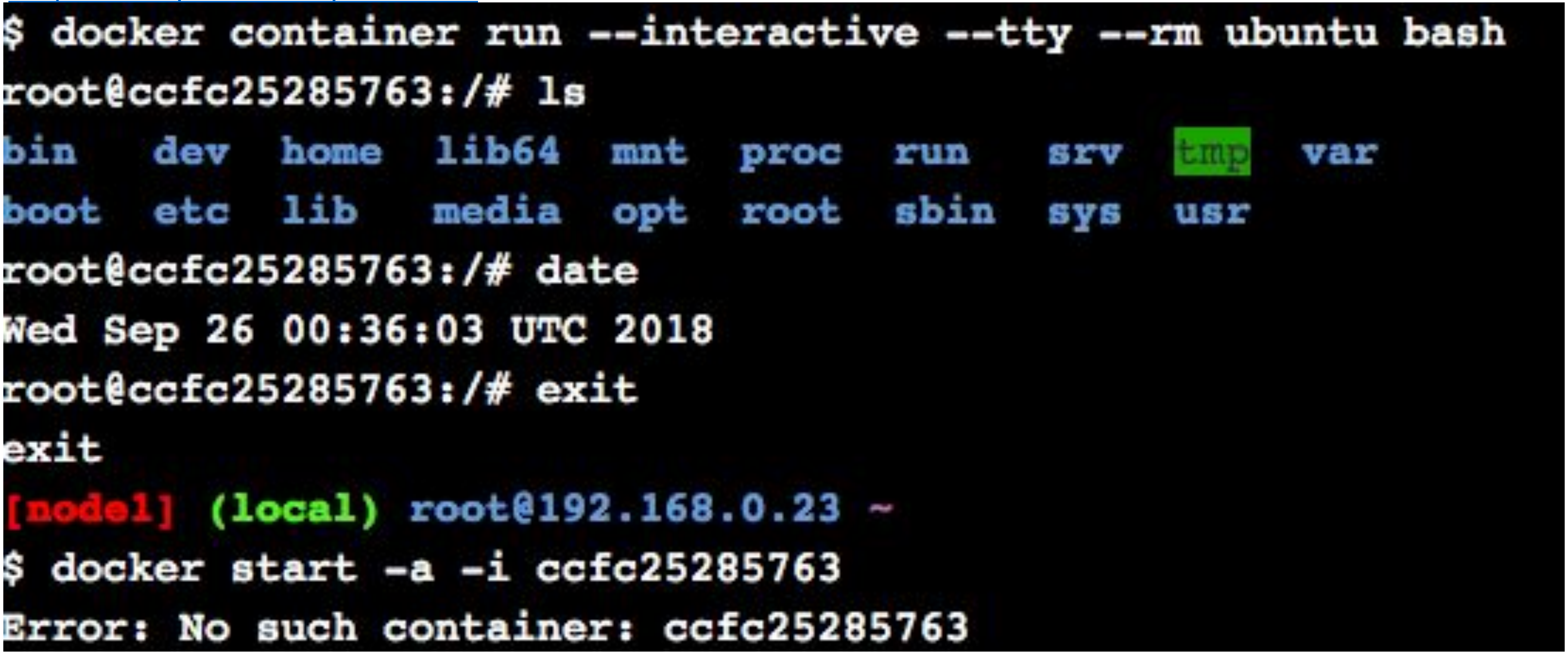
[illegible]



```
$ docker container run --interactive --tty --rm ubuntu /bin/bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
124c757242f8: Pull complete
9d866f8bde2a: Pull complete
fa3f2f277e67: Pull complete
398d32b153e8: Pull complete
afde35469481: Pull complete
Digest: sha256:de774a3145f7ca4f0bd144c7d4ffb2931e06634f11529653b23eba85aef8e378
Status: Downloaded newer image for ubuntu:latest
root@089aa5ad2478:/# exit
exit
[node1] (local) root@192.168.0.23 ~
$ docker container ls --all
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
                 PORTS               NAMES
b4e23e71e104        alpine              "/bin/sh"           5 minutes ago       Exited (0)
 4 minutes ago                       nifty_franklin
53b3b3c0991a        alpine              "date"              6 minutes ago       Exited (0)
 6 minutes ago                       zealous_lamarr
```

STEP 1: RUNNING PRE-BUILT DOCKER CONTAINERS (4)

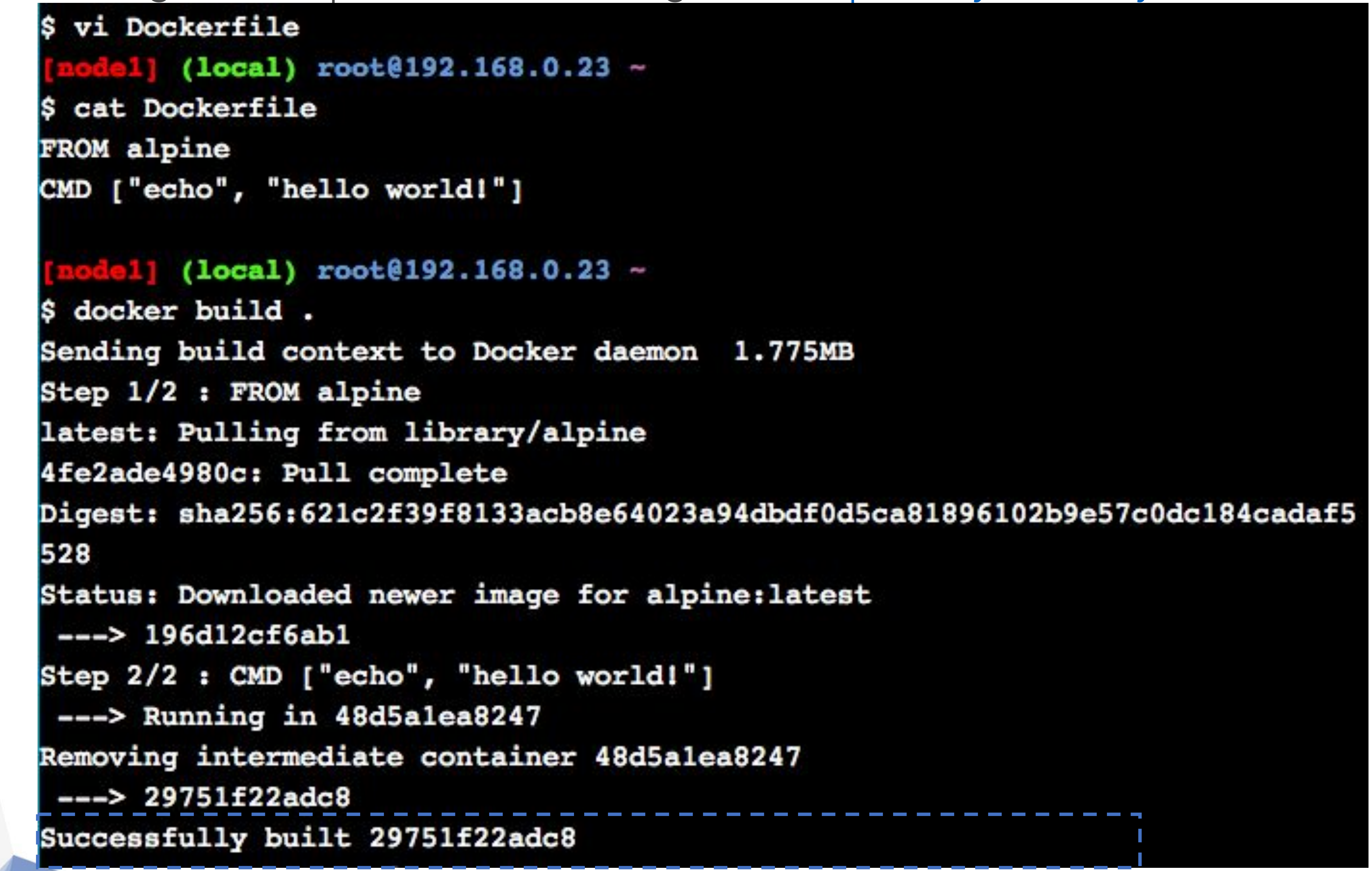
Please go to “Step 1” at the following link:
https://tinyurl.com/yatewkx2



```
$ docker container run --interactive --tty --rm ubuntu bash
root@ccfc25285763:/# ls
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
root@ccfc25285763:/# date
Wed Sep 26 00:36:03 UTC 2018
root@ccfc25285763:/# exit
exit
[node1] (local) root@192.168.0.23 ~
$ docker start -a -i ccfc25285763
Error: No such container: ccfc25285763
```

STEP 2: PACKAGE AND RUN AN APPLICATION USING DOCKER (1)

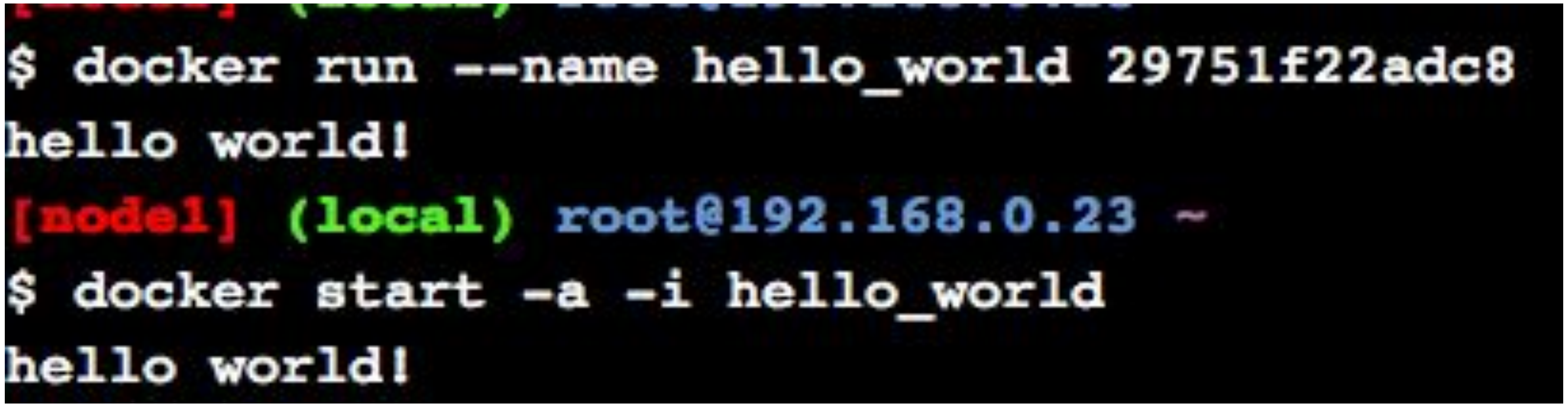
Please go to “Step 2” at the following link: https://tinyurl.com/yatewkx2



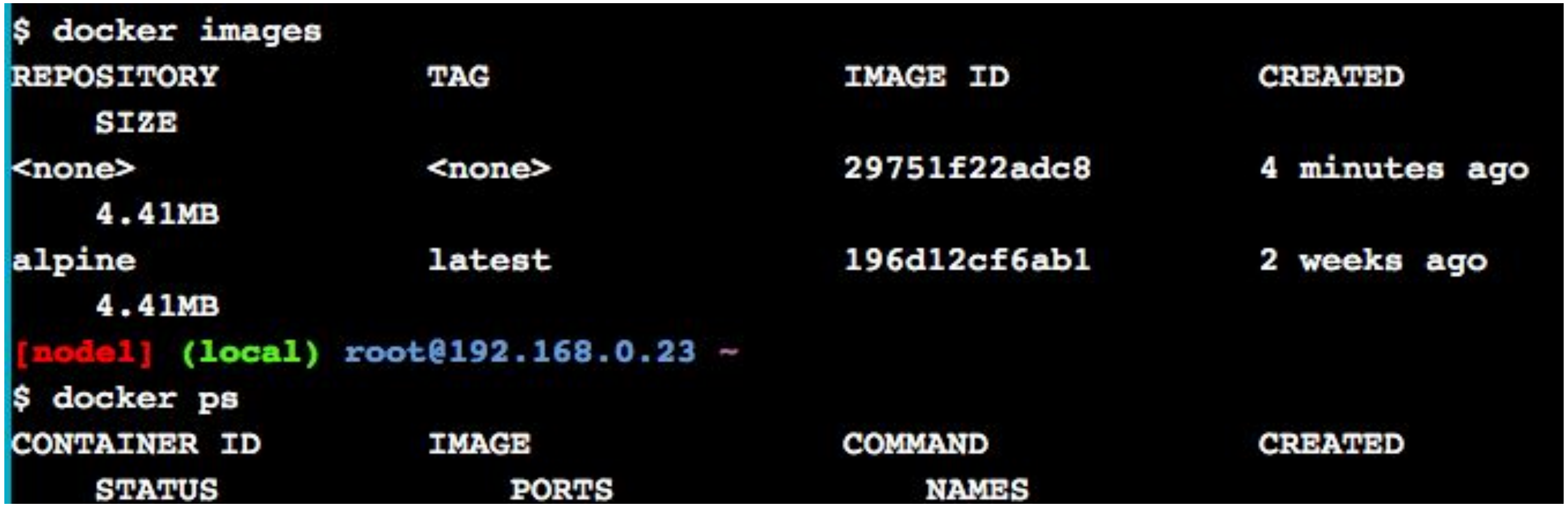
```
$ vi Dockerfile
[node1] (local) root@192.168.0.23 ~
$ cat Dockerfile
FROM alpine
CMD ["echo", "hello world!"]

[node1] (local) root@192.168.0.23 ~
$ docker build .
Sending build context to Docker daemon  1.775MB
Step 1/2 : FROM alpine
latest: Pulling from library/alpine
4fe2ade4980c: Pull complete
Digest: sha256:621c2f39f8133acb8e64023a94dbdf0d5ca81896102b9e57c0dc184cadaf5
528
Status: Downloaded newer image for alpine:latest
 ---> 196d12cf6ab1
Step 2/2 : CMD ["echo", "hello world!"]
 ---> Running in 48d5a1ea8247
Removing intermediate container 48d5a1ea8247
 ---> 29751f22adc8
Successfully built 29751f22adc8
```

STEP 2: PACKAGE AND RUN AN APPLICATION USING DOCKER (2)

Please go to “Step 2” at the following link: https://tinyurl.com/yatewkx2



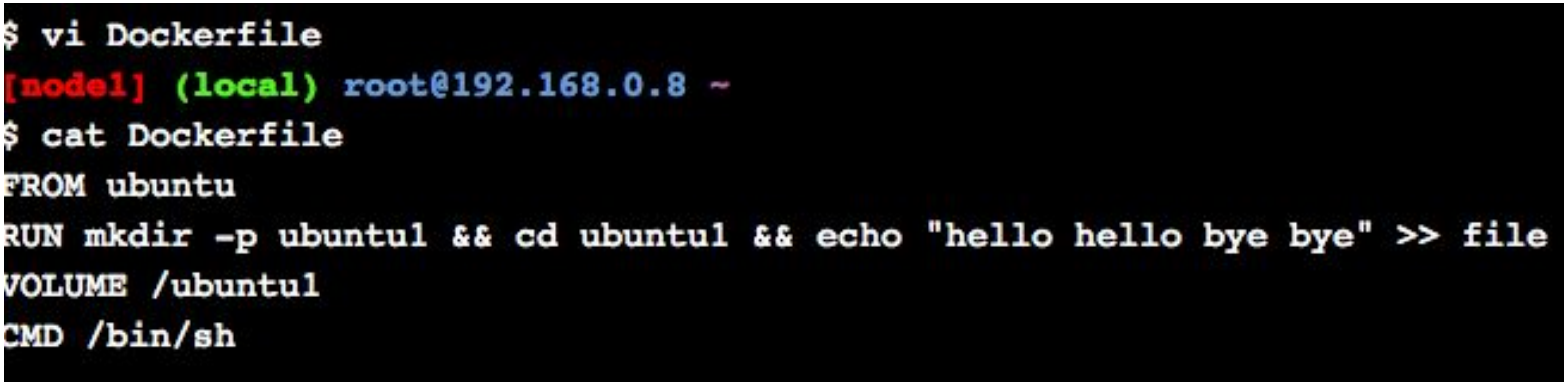
```
$ docker run --name hello_world 29751f22adc8
hello world!
[node1] (local) root@192.168.0.23 ~
$ docker start -a -i hello_world
hello world!
```



```
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
    SIZE
<none>              <none>              29751f22adc8        4 minutes ago
    4.41MB
alpine              latest              196d12cf6ab1        2 weeks ago
    4.41MB
[node1] (local) root@192.168.0.23 ~
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
    STATUS              PORTS               NAMES
```

STEP 3: ADDING VOLUME (1)

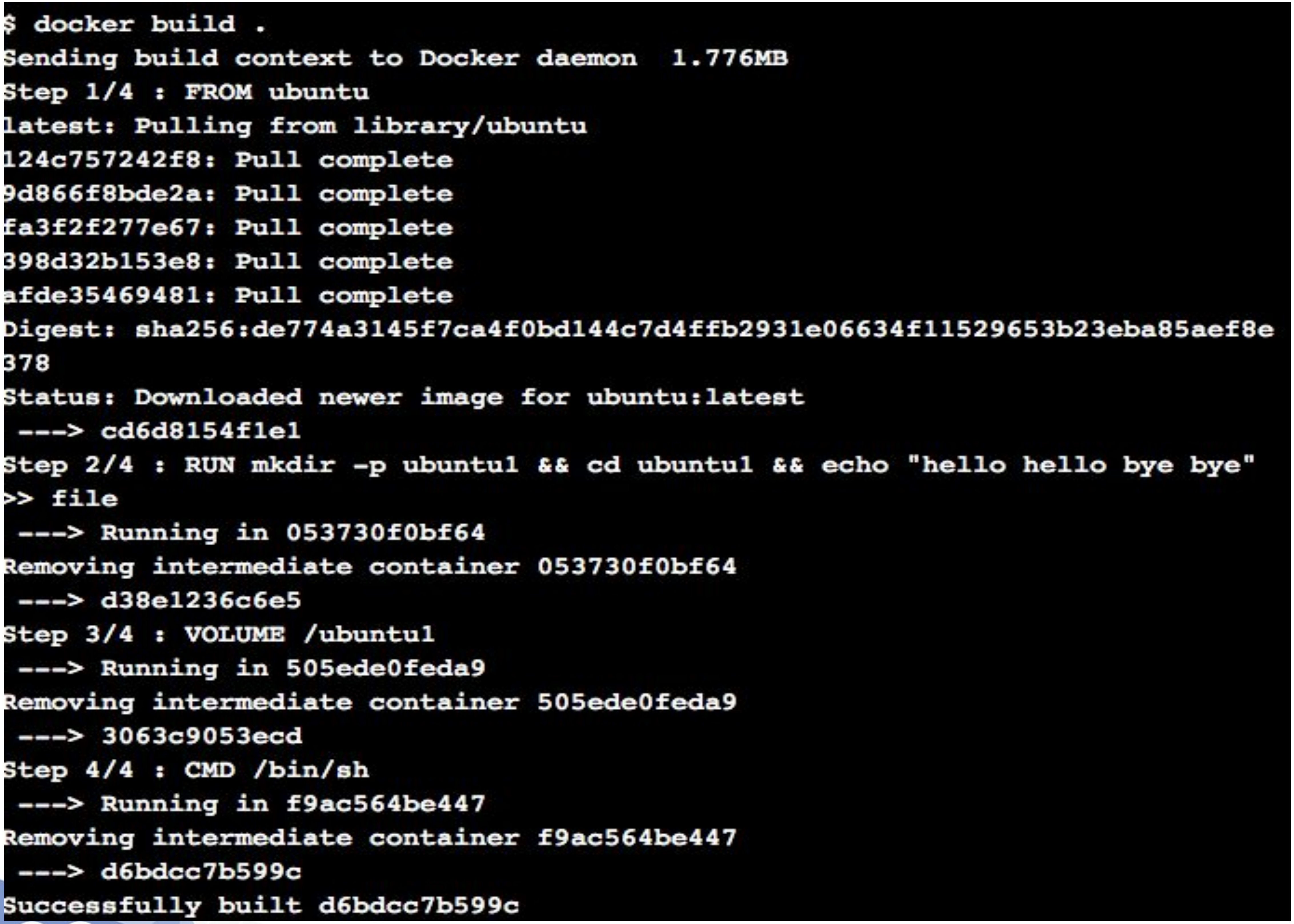
Please go to “Step 3” at the following link: https://tinyurl.com/yatewkx2



```
$ vi Dockerfile
[node1] (local) root@192.168.0.8 ~
$ cat Dockerfile
FROM ubuntu
RUN mkdir -p ubuntu1 && cd ubuntu1 && echo "hello hello bye bye" >> file
VOLUME /ubuntu1
CMD /bin/sh
```

STEP 3: ADDING VOLUME (2)

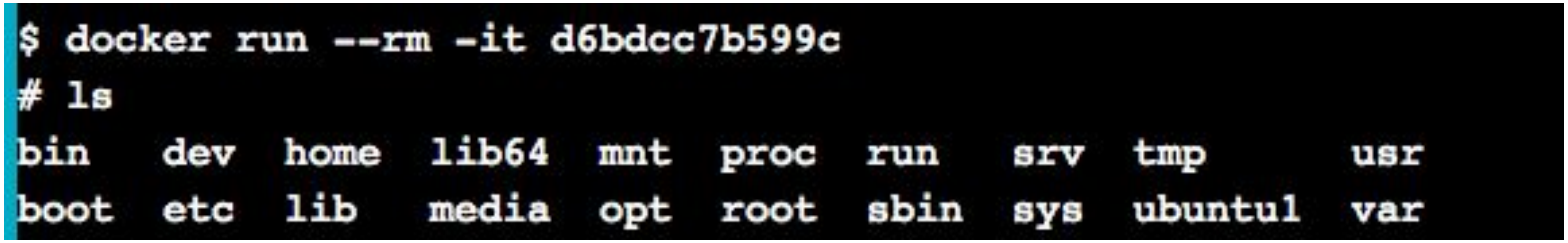
Please go to "Step 3" at the following link: https://tinyurl.com/yatewkx2



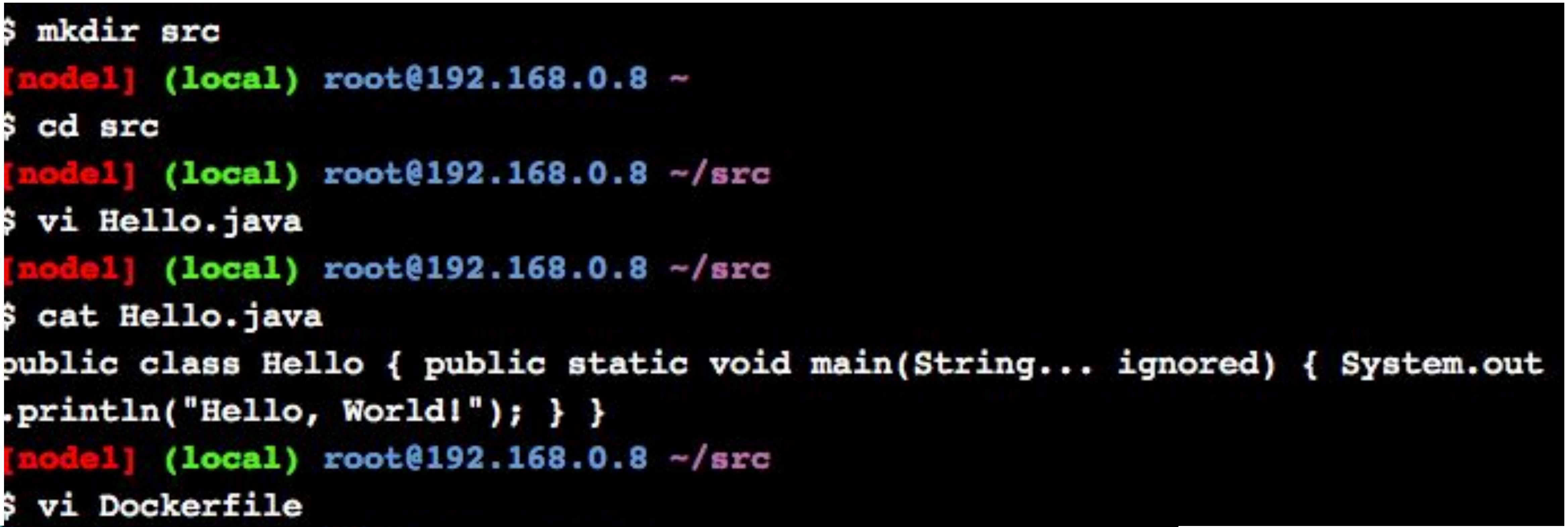
```
$ docker build .
Sending build context to Docker daemon  1.776MB
Step 1/4 : FROM ubuntu
latest: Pulling from library/ubuntu
124c757242f8: Pull complete
9d866f8bde2a: Pull complete
fa3f2f277e67: Pull complete
398d32b153e8: Pull complete
afde35469481: Pull complete
Digest: sha256:de774a3145f7ca4f0bd144c7d4ffb2931e06634f11529653b23eba85aef8e
378
Status: Downloaded newer image for ubuntu:latest
 ---> cd6d8154f1e1
Step 2/4 : RUN mkdir -p ubuntu1 && cd ubuntu1 && echo "hello hello bye bye"
>> file
 ---> Running in 053730f0bf64
Removing intermediate container 053730f0bf64
 ---> d38e1236c6e5
Step 3/4 : VOLUME /ubuntu1
 ---> Running in 505ede0feda9
Removing intermediate container 505ede0feda9
 ---> 3063c9053ecd
Step 4/4 : CMD /bin/sh
 ---> Running in f9ac564be447
Removing intermediate container f9ac564be447
 ---> d6bdcc7b599c
Successfully built d6bdcc7b599c
```

STEP 3: ADDING VOLUME (3)

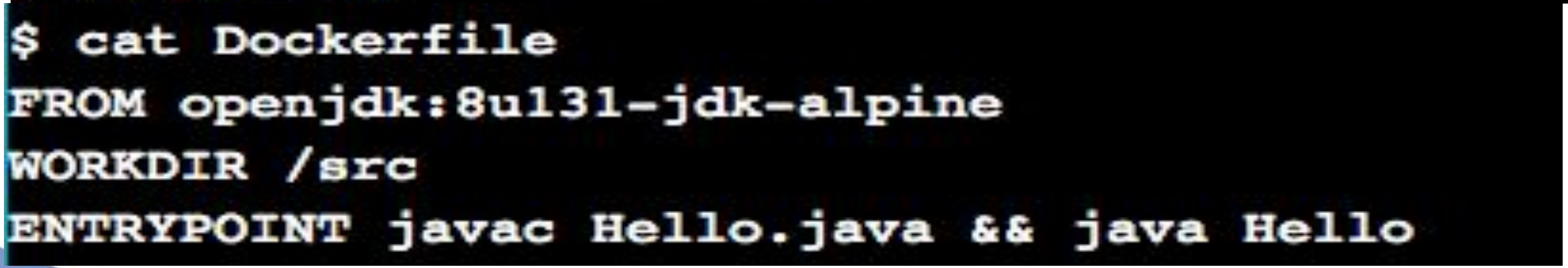
Please go to “Step 3” at the following link: https://tinyurl.com/yatewkx2



```
$ docker run --rm -it d6bdcc7b599c
# ls
bin    dev   home  lib64  mnt  proc  run   srv  tmp      usr
boot   etc   lib   media  opt  root  sbin  sys  ubuntu1  var
```



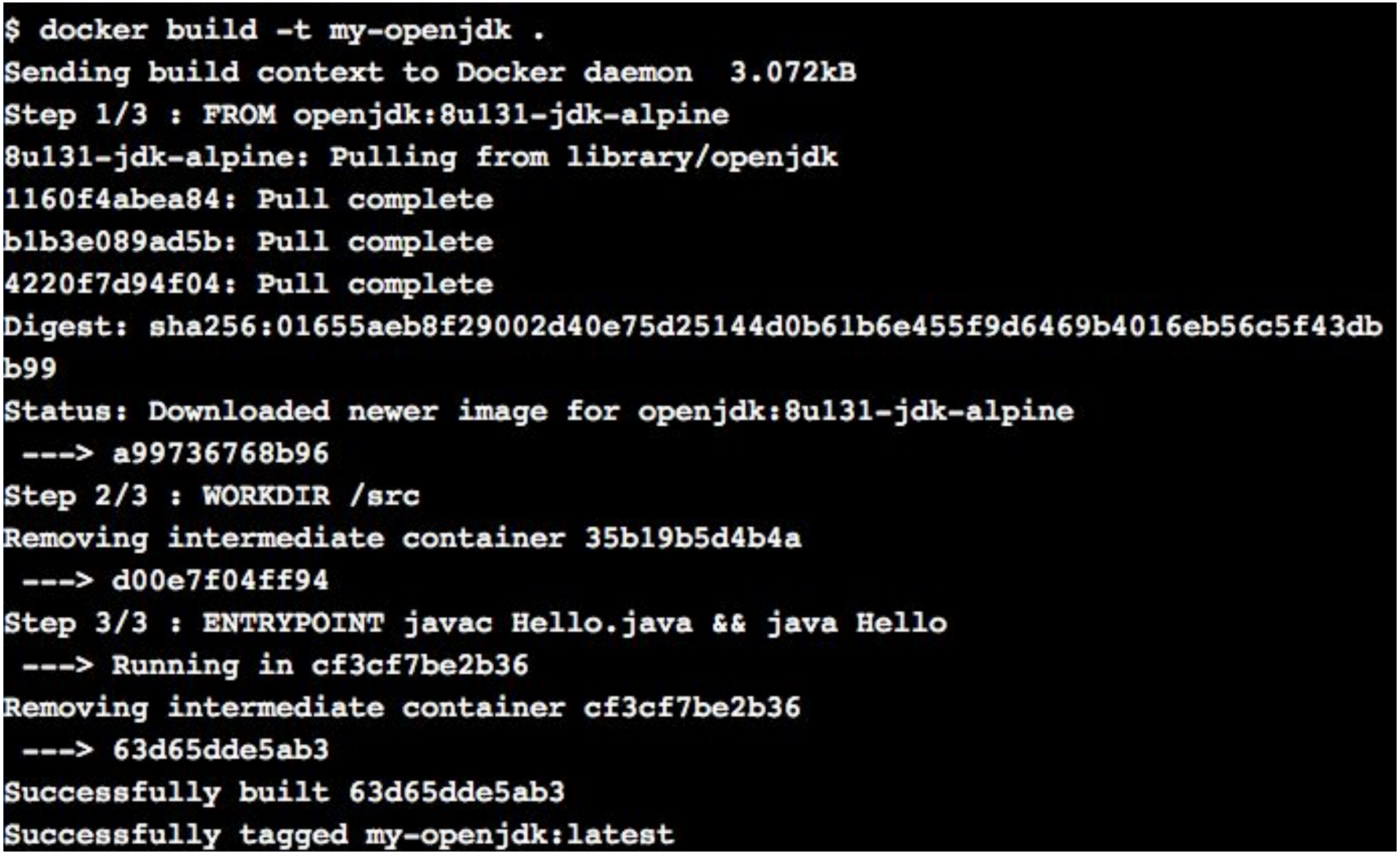
```
$ mkdir src
[node1] (local) root@192.168.0.8 ~
$ cd src
[node1] (local) root@192.168.0.8 ~/src
$ vi Hello.java
[node1] (local) root@192.168.0.8 ~/src
$ cat Hello.java
public class Hello { public static void main(String... ignored) { System.out
.println("Hello, World!"); } }
[node1] (local) root@192.168.0.8 ~/src
$ vi Dockerfile
```



```
$ cat Dockerfile
FROM openjdk:8u131-jdk-alpine
WORKDIR /src
ENTRYPOINT javac Hello.java && java Hello
```

STEP 3: ADDING VOLUME (4)

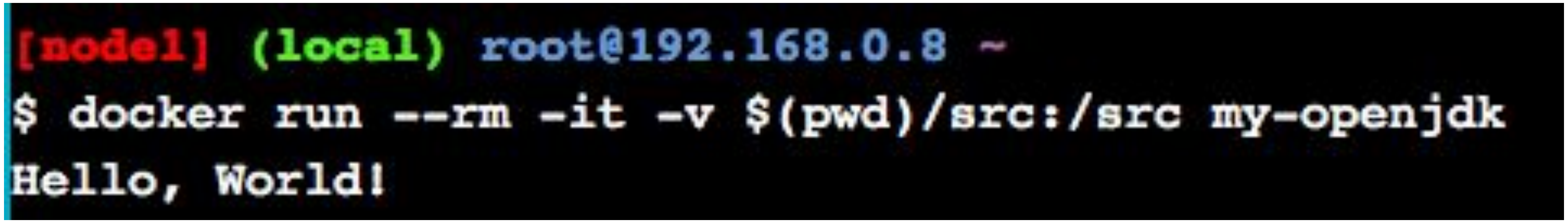
Please go to “Step 3” at the following link: https://tinyurl.com/yatewkx2



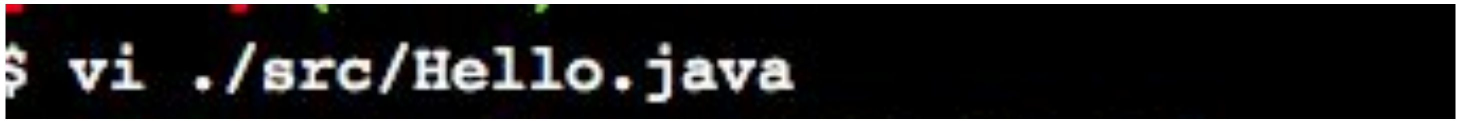
```
$ docker build -t my-openjdk .
Sending build context to Docker daemon  3.072kB
Step 1/3 : FROM openjdk:8u131-jdk-alpine
8u131-jdk-alpine: Pulling from library/openjdk
1160f4abea84: Pull complete
b1b3e089ad5b: Pull complete
4220f7d94f04: Pull complete
Digest: sha256:01655aeb8f29002d40e75d25144d0b61b6e455f9d6469b4016eb56c5f43db
b99
Status: Downloaded newer image for openjdk:8u131-jdk-alpine
 ---> a99736768b96
Step 2/3 : WORKDIR /src
Removing intermediate container 35b19b5d4b4a
 ---> d00e7f04ff94
Step 3/3 : ENTRYPOINT javac Hello.java && java Hello
 ---> Running in cf3cf7be2b36
Removing intermediate container cf3cf7be2b36
 ---> 63d65dde5ab3
Successfully built 63d65dde5ab3
Successfully tagged my-openjdk:latest
```

STEP 3: ADDING VOLUME (5)

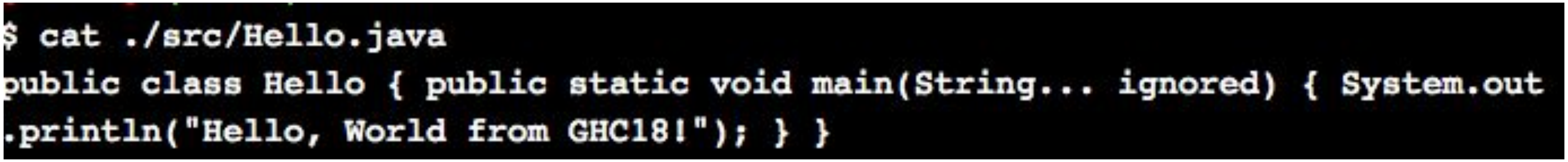
Please go to “Step 3” at the following link: https://tinyurl.com/yatewkx2



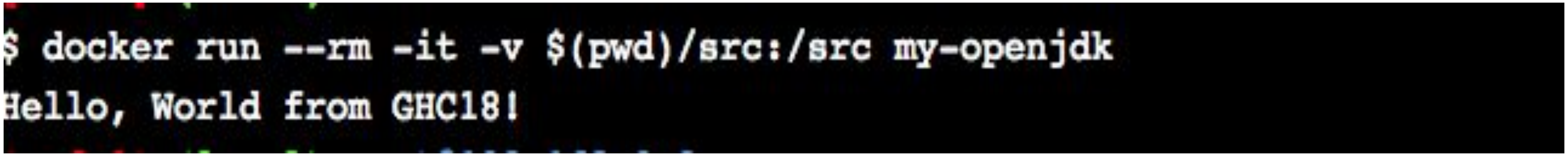
```
[node1] (local) root@192.168.0.8 ~
$ docker run --rm -it -v $(pwd)/src:/src my-openjdk
Hello, World!
```



```
$ vi ./src/Hello.java
```



```
$ cat ./src/Hello.java
public class Hello { public static void main(String... ignored) { System.out
.println("Hello, World from GHC18!"); } }
```



```
$ docker run --rm -it -v $(pwd)/src:/src my-openjdk
Hello, World from GHC18!
```

Additional Topics On Using Docker

DOCKER COMPOSE

- A tool for defining and running multi-container Docker applications

- Provides a configuration that can be used to bring up an application and the suite of services it depends on with just one command

- There is a notion of swarm managers that can manage a swarm, while standalone containers can be started on any daemon

- This tool should be installed separately

DOCKER SWARM

- A native clustering solution for Docker - it implements Docker’s orchestration layer
- This is a separate project, that is now a part of the Docker engine
- One of the key advantages of swarm services is that you can modify a service’s configuration, including the networks and volumes it is connected to, without the need to manually restart the service
- Docker will update the configuration, stop the service tasks with the out of date configuration, and create new ones matching the desired configuration

Conclusion

SUMMARY

- In this workshop we covered:
  - What is Docker
  - Why is it important?
  - How can we use Docker?
  - We built our own Docker images and pulled pre-built ones from Docker Hub and ran them as containers

NEXT?

Let us go over the exercises at the following link:

https://training.play-with-docker.com/beginner-linux/

Thanks!
Any Questions or Comments?

Email:
{charlie, mpoindexter, rauta}@tacc.utexas.edu