Using Docker in BioHPC Cloud

Jaroslaw Pillardy
Bioinformatics Facility

Part 3 – presentation

May 26 2020

Docker vs Conda

Docker	Conda
Fully isolated, no direct interaction with host software	Partially isolated, can use software installed outside of Conda
Full copy of Linux OS, can start from scratch and work as an administrator.	Add-on to current operating system, works in user space
Almost any software can be installed and deployed	Only programs that work in user space can be installed
A separate environment, Docker needs to be preinstalled. Need to deploy images and containers before using.	Easy to start and use, part of user environment
Portable – images can be saved and moved easily to new machines, even outside BioHPC.	Hard to replicate to the servers outside BioHPC. Can be used inside BioHPC from home directory.

Building images from dockerfiles

Docker images can be built using a list of commands stored in file called *dockerfile*

Dockerfile below will create a CentOS 7 image with htop installed, same as in our “htop example” in Part 1.

```
FROM centos:7
RUN yum -y install epel-release
RUN yum -y install htop
```

Building images from dockerfiles – htop example

```
[jarekp@cbsum1c1b009 jarekp]$ pwd
/workdir/jarekp

[jarekp@cbsum1c1b009 jarekp]$ mkdir htop_build

[jarekp@cbsum1c1b009 jarekp]$ echo "FROM centos:7" > htop_build/dockerfile
[jarekp@cbsum1c1b009 jarekp]$ echo "RUN yum -y install epel-release" >> htop_build/dockerfile
[jarekp@cbsum1c1b009 jarekp]$ echo "RUN yum -y install htop" >> htop_build/dockerfile

[jarekp@cbsum1c1b009 jarekp]$ cat htop_build/dockerfile
FROM centos:7
RUN yum -y install epel-release
RUN yum -y install htop
[jarekp@cbsum1c1b009 jarekp]$
```

Building images from dockerfiles

To build the image you need to use **docker1 build** command, with full path to the dockerfile and dockerfile directory. The dockerfile can only be under /workdir/labid . Docker build command points to directory where dockerfile resides, if multiple dockerfiles are present additional -f options specifies which one to use.

```
[jarekp@cbsum1c1b009 jarekp]$ docker1 build -t my_centos7_htop /workdir/jarekp/htop_build
Sending build context to Docker daemon 2.048 kB
Step 1/3 : FROM centos:7
 ---> b5b4d78bc90c
Step 2/3 : RUN yum -y install epel-release
 ---> Running in cd0a3e93fc9d
[…]
Removing intermediate container a4f92ce8a07c
Successfully built 867b5b2a41c7
[jarekp@cbsum1c1b009 jarekp]$
```

Building images from dockerfiles – htop example

```
[jarekp@cbsum1c1b009 jarekp]$ docker1 images
REPOSITORY                      TAG           IMAGE ID            CREATED             SIZE
biohpc_jarekp/my_centos7_htop   latest        867b5b2a41c7        2 minutes ago       384 MB


[jarekp@cbsum1c1b009 jarekp]$ docker1 save -o my_centos7_htop.tar
biohpc_jarekp/my_centos7_htop
[jarekp@cbsum1c1b009 jarekp]$
```

Docker components

dockerfile – a text file with a list of instructions to create an image with docker build command.

image - a Docker template than can be loaded into Docker and executed. Image contains many files and can be stored in a hub/registry on in a tar file archive.

container - a running instance of Docker image – actual Docker “machine”.

Building images from dockerfiles

Dockerfiles seem to be a great way to store images in an easy and human readable way.

In reality there are a lot of problems. Each time you create an image from dockerfile it will install software from repositories, which may not be available anymore, or software versions may have changed, so they don't work together anymore.

If you have a good working image, save it! Dockerfile is a good way to document how it was created.

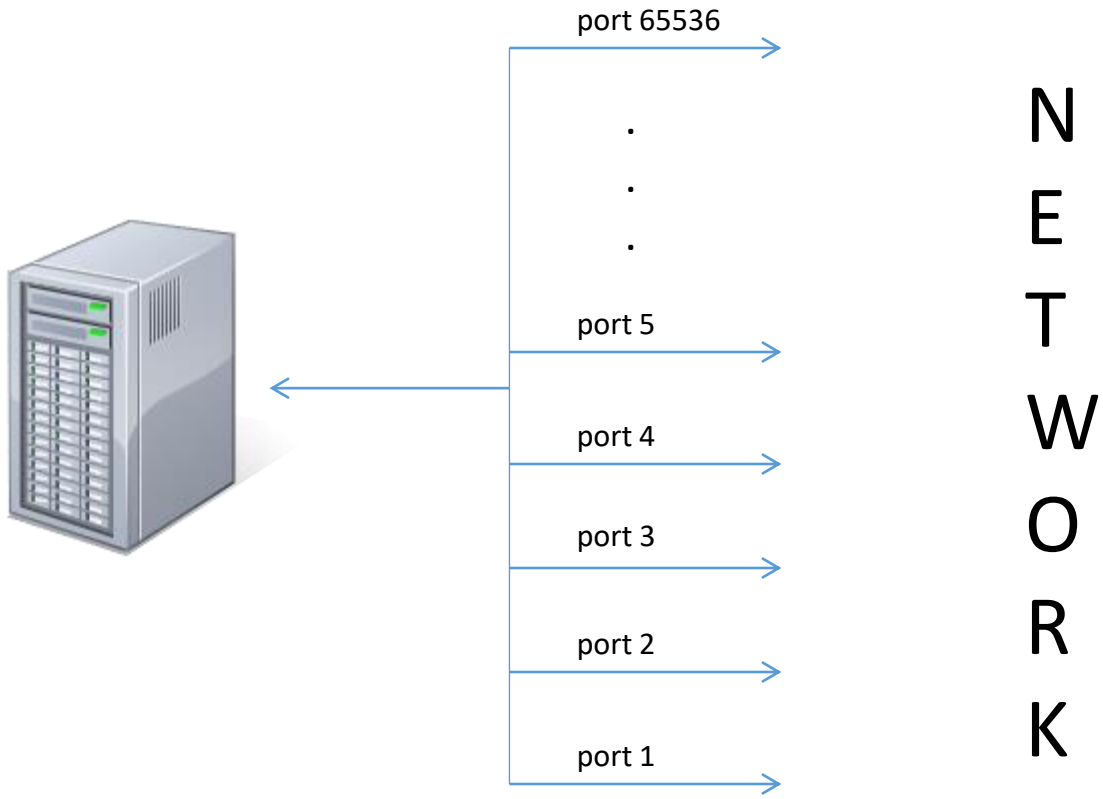
BioHPC Docker - network ports

Each service on the network is referenced by two values

1. Server address (i.e. IP number usually linked to a name)

2. Service port (a number referencing network socket to connect to).

BioHPC Docker - network ports

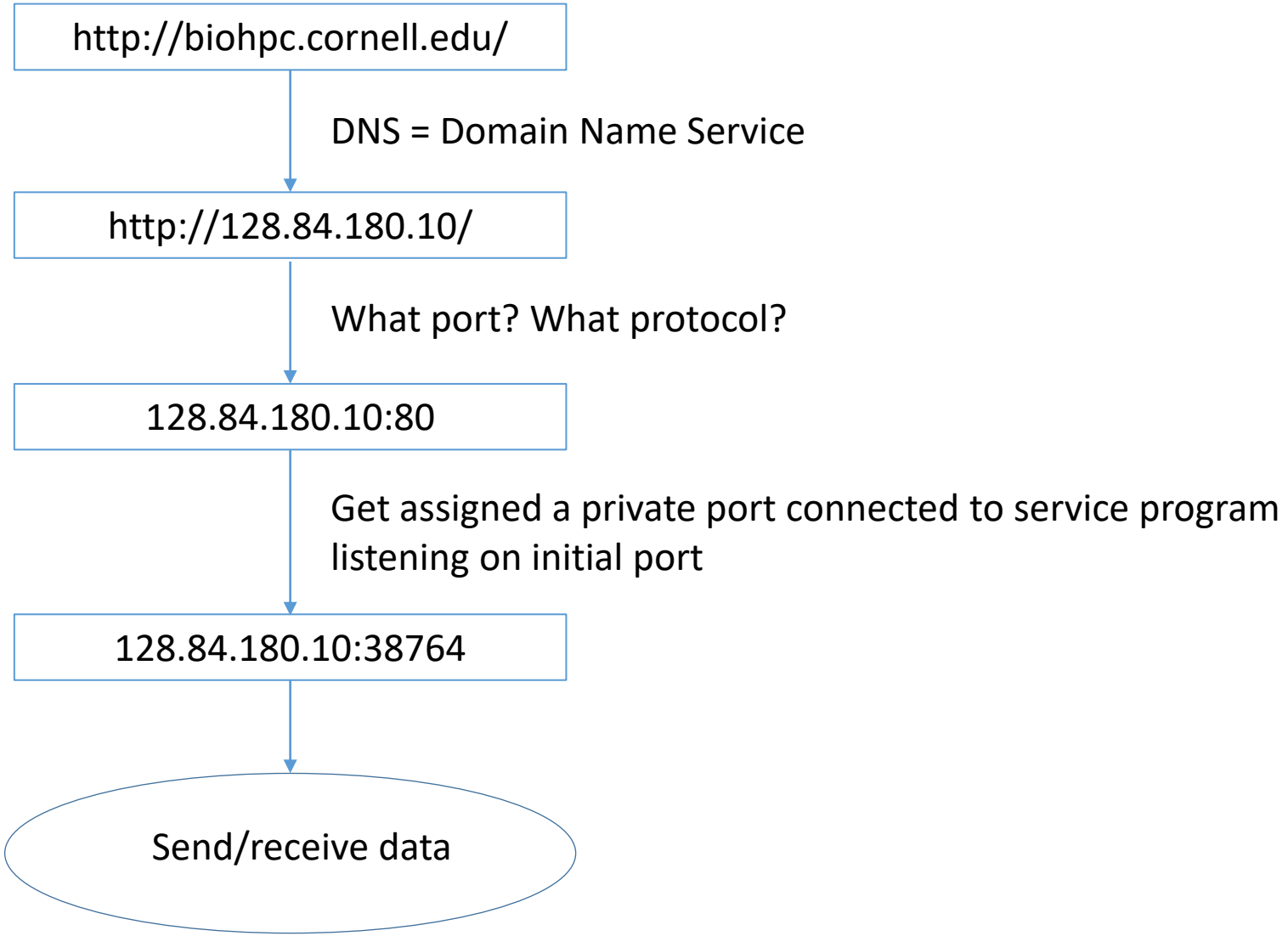


Computer full address: ip_number:port i.e. 128.8.3.22:22

BioHPC Docker - network ports

Service (protocol)	Port
FTP	20 and 21
TELNET	23
SSH	22
SMTP (mail service)	25
DNS (domain name system)	53
HTTP (www)	80
HTTPS (www secure)	443

BioHPC Docker - network ports



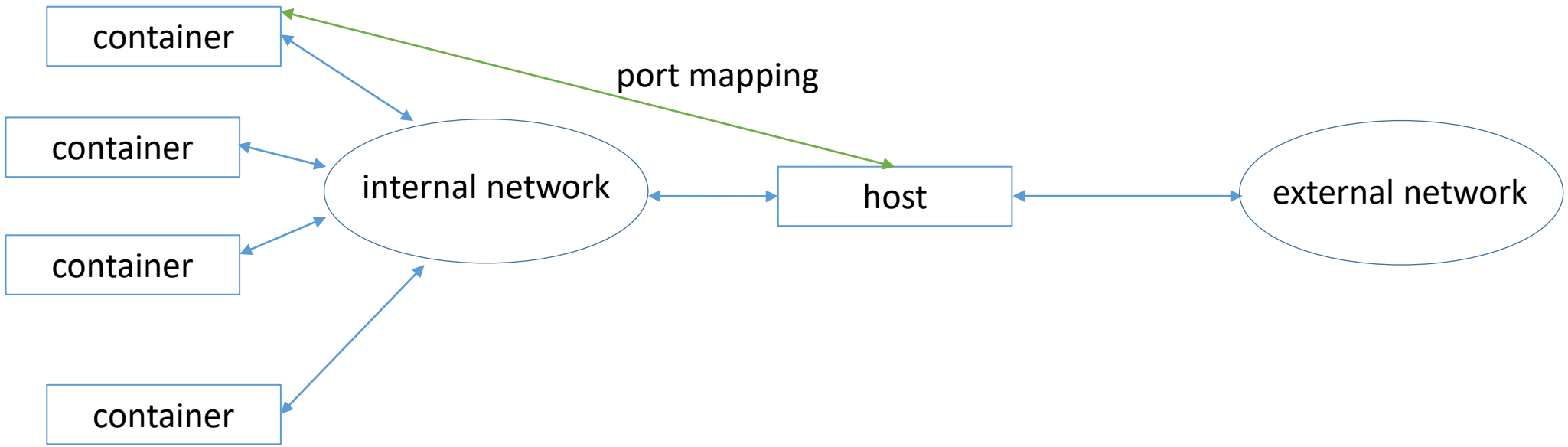
BioHPC Docker - network ports

Containers can offer network services, and you can connect to them if you know what is container IP and service port.

You can also map container port to host port. Then you can connect to your host and reference assigned port number to access container service.

BioHPC Docker - network ports

Docker maintains an internal network inside host usually 172.17.0.*. This network is NOT accessible from the outside of the server, but it is used for communication between containers and the host server.



BioHPC Docker - network ports

What is container IP?

=> use **docker1 inspect**

BioHPC Docker - network ports

```
[jarekp@cbsum1c1b009 jarekp]$ docker1 inspect 25eaa875f6b9
[
    {
        "Id": "25eaa875f6b95e69a3152dccf02a79166555a4507fa4311727cf2d978db47e1a",
        "Created": "2020-05-18T21:23:11.505755196Z",
        "Path": "/bin/bash",
        "Args": [],

… lines skipped …

        "NetworkSettings": {
            "Bridge": "",

… lines skipped …

            "Gateway": "172.17.0.1",
            "GlobalIPv6Address": "",
            "GlobalIPv6PrefixLen": 0,
            "IPAddress": "172.17.0.2",
```

BioHPC Docker – setting up a web server

1. Google about “web server” on CentOS Linux – it is Apache
2. Find out how to install Apache on CentOS 7 – plenty examples
3. Do it – pull CentOS 7 image and install Apache
4. How to run Apache in container – search that too, it may be different than in non-Docker CentOS
5. Prepare test HTML file, startup script and run it!
6. Map ports so you can enjoy your new web server from campus network

BioHPC Docker – setting up a web server

pull CentOS 7 and install Apache inside

```
[jarekp@cbsum1c1b009 ~]$ docker1 run -it -d centos:7 /bin/bash
Unable to find image 'centos:7' locally
Trying to pull repository dtr.cucloud.net/centos ...
Trying to pull repository docker.io/library/centos ...
sha256:4a701376d03f6b39b8c2a8f4a8e499441b0d567f9ab9d58e4991de4472fb813c: Pulling from
docker.io/library/centos
ab5ef0e58194: Pull complete
Digest: sha256:4a701376d03f6b39b8c2a8f4a8e499441b0d567f9ab9d58e4991de4472fb813c
Status: Downloaded newer image for docker.io/centos:7
8e7a04aac719de0f8e553e820dc1df7ac8537c6b7f418d7d219eded11141b665
[jarekp@cbsum1c1b009 ~]$ docker1 ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS               NAMES
8e7a04aac719        centos:7            "/bin/bash"         2 minutes ago       Up 2 minutes
jarekp__biohpc_1
[jarekp@cbsum1c1b009 ~]$ docker1 exec -it 8e7a04aac719 /bin/bash
[root@8e7a04aac719 /]# yum -y install httpd
Loaded plugins: fastestmirror, ovl
Determining fastest mirrors
 * base: mirror.siena.edu
 * extras: mirror.mi.incx.net
 * updates: mirror.metrocast.net
```

BioHPC Docker – setting up a web server

How to start a service inside container?

Apache service name is ‘httpd’ and normally it is started on CentOS with command ‘systemctl start httpd’

Won’t work in a container, it requires ‘systemd’ service, not present in Docker.

Instead search online “how to start httpd in CentOS Docker”

BioHPC Docker – setting up a web server

/usr/bin/env bash -c 'exec /usr/sbin/apachectl -DFOREGROUND’ &

In the container create a file /start.sh that will be starting our web server – this way you won't need to start it manually each time

```
--------
#!/bin/bash

rm -rf /run/httpd/* /tmp/httpd*
/usr/bin/env bash -c 'exec /usr/sbin/apachectl -DFOREGROUND' &

/bin/bash
--------
```

BioHPC Docker – setting up a web server

Also create a test HTML file /var/www/html/test.htm so we can see something in browser

---

<h1>Hello from Docker</h1

```
[root@8e7a04aac719 /]# vi /var/www/html/test.htm
[root@8e7a04aac719 /]# vi /start.sh
[root@8e7a04aac719 /]# chmod a+x /start.sh
[root@8e7a04aac719 /]# exit
```

BioHPC Docker – setting up a web server

Our web server is almost ready. We need to save it as an image so we can start it later and execute /start.sh to run our web server program.

```
[jarekp@cbsum1c1b009 ~]$ docker1 ps -a
CONTAINER ID        IMAGE      COMMAND        CREATED         STATUS          PORTS NAMES
8e7a04aac719        centos:7   "/bin/bash"    30 minutes ago Up 30 minutes  jarekp__biohpc_1
[jarekp@cbsum1c1b009 ~]$ docker1 commit 8e7a04aac719 webserver
sha256:0b7cd07abff03b034c3ba8146e3ae9f620e1ca1a41a5700e8635965ecd9569d7
[jarekp@cbsum1c1b009 ~]$ docker1 images
REPOSITORY                    TAG         IMAGE ID            CREATED              SIZE
biohpc_jarekp/webserver       latest      0b7cd07abff0        11 seconds ago       348 MB
[jarekp@cbsum1c1b009 ~]$ docker1 save -o webserver.tar biohpc_jarekp/webserver
```

New image is stored on this server only!

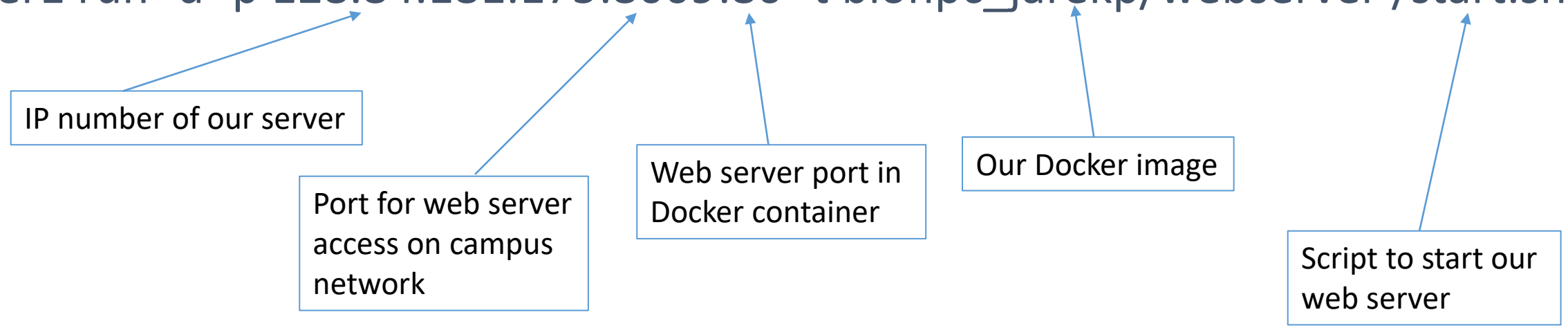
MUST be saved to reuse on other servers.

BioHPC Docker - setting up a web server

You can map container ports to external host ports, but they need to be opened in the host firewall to be accessible.

We keep ports 8009 – 8019 open for campus access.

```
docker1 run -d -p 128.84.181.175:8009:80 -t biohpc_jarekp/webserver /start.sh
```



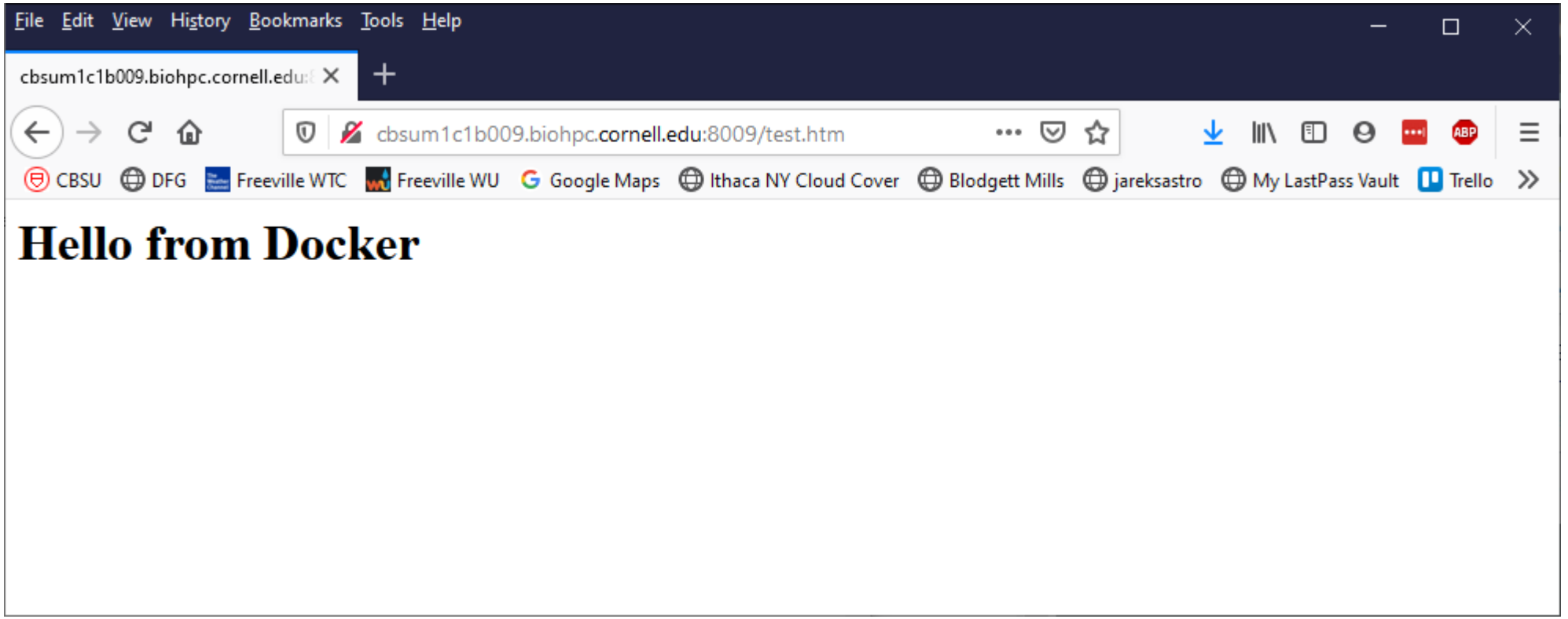
BioHPC Docker – setting up a web server

```
[jarekp@cbsum1c1b009 ~]$ ping cbsum1c1b009
PING cbsum1c1b009 (128.84.181.175) 56(84) bytes of data.
64 bytes from cbsum1c1b009 (128.84.181.175): icmp_seq=1 ttl=64 time=0.050 ms
^C
--- cbsum1c1b009 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.050/0.050/0.050/0.000 ms

[jarekp@cbsum1c1b009 ~]$ docker1 run -d -p 128.84.181.175:8009:80 -t biohpc_jarekp/webserver /start.sh
909f4daec197c17b81e5ba59476634065cbfb16cd85efb317593a96b67993e48

[jarekp@cbsum1c1b009 ~]$ docker1 ps -a
CONTAINER ID   IMAGE                    COMMAND      CREATED         STATUS         PORTS                          NAMES
909f4daec197   biohpc_jarekp/webserver "/start.sh" 13 seconds ago  Up 11 seconds  128.84.181.175:8009->80/tcp   jarekp__biohpc_2
```

Easy way to find IP number of our server

NOTE: several users can run containers on your server. Run the command above to check if port 8009 is free. If not use other port between 8009-8019. If you try to use port that has been allocated you will get an error message “port is already allocated”.

BioHPC Docker – setting up a web server

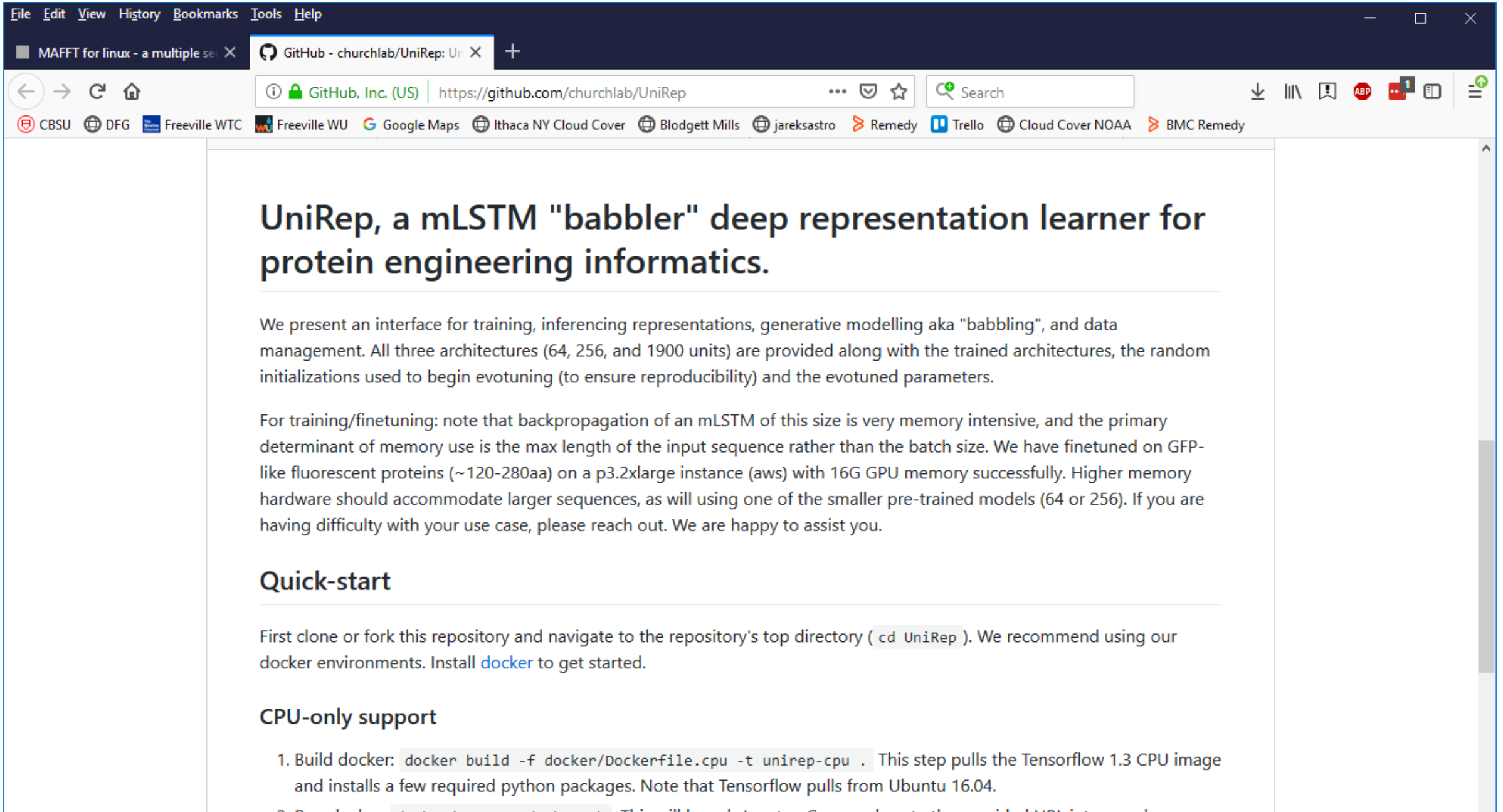


Our URL is
http://cbsum1c1b009.biohpc.cornell.edu:8009/test.htm

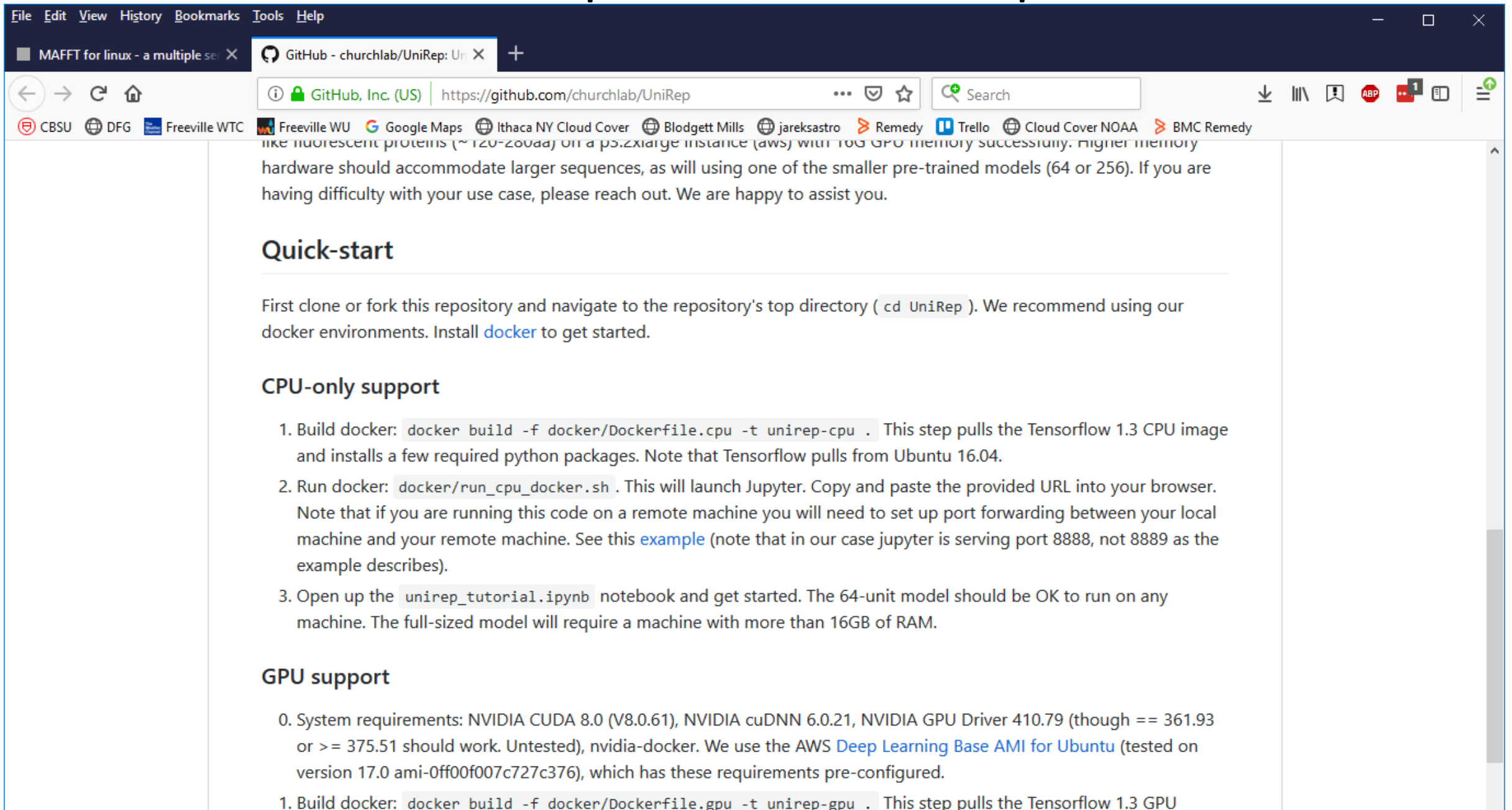
BioHPC Docker Example – install UniRep from dockerfile

1. Go to UniRep website and read instructions.
2. Download appropriate dockerfile
3. Build image from dockerfile
4. Save image for future use

BioHPC Docker Example – install UniRep from dockerfile



BioHPC Docker Example – install UniRep from dockerfile



like fluorescent proteins (~120-280aa) on a p3.2xlarge instance (aws) with 16G GPU memory successfully. Higher memory hardware should accommodate larger sequences, as will using one of the smaller pre-trained models (64 or 256). If you are having difficulty with your use case, please reach out. We are happy to assist you.

Quick-start

First clone or fork this repository and navigate to the repository's top directory (`cd UniRep`). We recommend using our docker environments. Install docker to get started.

CPU-only support

1. Build docker: `docker build -f docker/Dockerfile.cpu -t unirep-cpu .` This step pulls the Tensorflow 1.3 CPU image and installs a few required python packages. Note that Tensorflow pulls from Ubuntu 16.04.
2. Run docker: `docker/run_cpu_docker.sh` . This will launch Jupyter. Copy and paste the provided URL into your browser. Note that if you are running this code on a remote machine you will need to set up port forwarding between your local machine and your remote machine. See this example (note that in our case jupyter is serving port 8888, not 8889 as the example describes).
3. Open up the `unirep_tutorial.ipynb` notebook and get started. The 64-unit model should be OK to run on any machine. The full-sized model will require a machine with more than 16GB of RAM.

GPU support

0. System requirements: NVIDIA CUDA 8.0 (V8.0.61), NVIDIA cuDNN 6.0.21, NVIDIA GPU Driver 410.79 (though == 361.93 or >= 375.51 should work. Untested), nvidia-docker. We use the AWS Deep Learning Base AMI for Ubuntu (tested on version 17.0 ami-0ff00f007c727c376), which has these requirements pre-configured.
1. Build docker: `docker build -f docker/Dockerfile.gpu -t unirep-gpu .` This step pulls the Tensorflow 1.3 GPU

BioHPC Docker Example – install UniRep from dockerfile

```
cd /workdir/jarekp
git clone https://github.com/churchlab/UniRep.git
cd UniRep
```

Command suggested in instructions:

```
docker build -f docker/Dockerfile.cpu -t unirep-cpu .
```

However, docker1 requires FULL PATH to dockerfile and dockerfile directory! Our command:

```
docker1 build -f /workdir/jarekp/UniRep/docker/Dockerfile.cpu -t unirep-cpu /workdir/jarekp/UniRep
```

Command suggested to run is docker/run_cpu_docker.sh . What is inside?

```
docker run -it -p 8888:8888 -p 6006:6006 -v `pwd`:/notebooks  unirep-cpu:latest
```

=> Won't work! Need to change *docker* to *docker1*. Also need to change image name

BioHPC Docker Example – install UniRep from dockerfile

```
docker1 images
```

```
REPOSITORY                      TAG                 IMAGE ID            CREATED             SIZE
biohpc_jarekp/unirep-cpu        latest              08bfdf512403        5 minutes ago       1.37 GB
```

Our command will be:

It will only work if I am in UniRep directory

```
docker1 run -it -p 8888:8888 -p 6006:6006 -v `pwd`:/notebooks  biohpc_jarekp/unirep-cpu
```

After changing docker to docker1 and image name it is a good idea to save both the image and all the files. I use vi to edit, you can use your favorite Linux editor

```
vi docker/run_cpu_docker.sh
cd ..
docker1 save -o /home/jarekp/UniRep_image.tar biohpc_jarekp/unirep-cpu
tar -cf /home/jarekp/UniRep_files.tar UniRep
```

Docker @ BioHPC

Often there is a Docker image on a public hub with a program you may want to run.

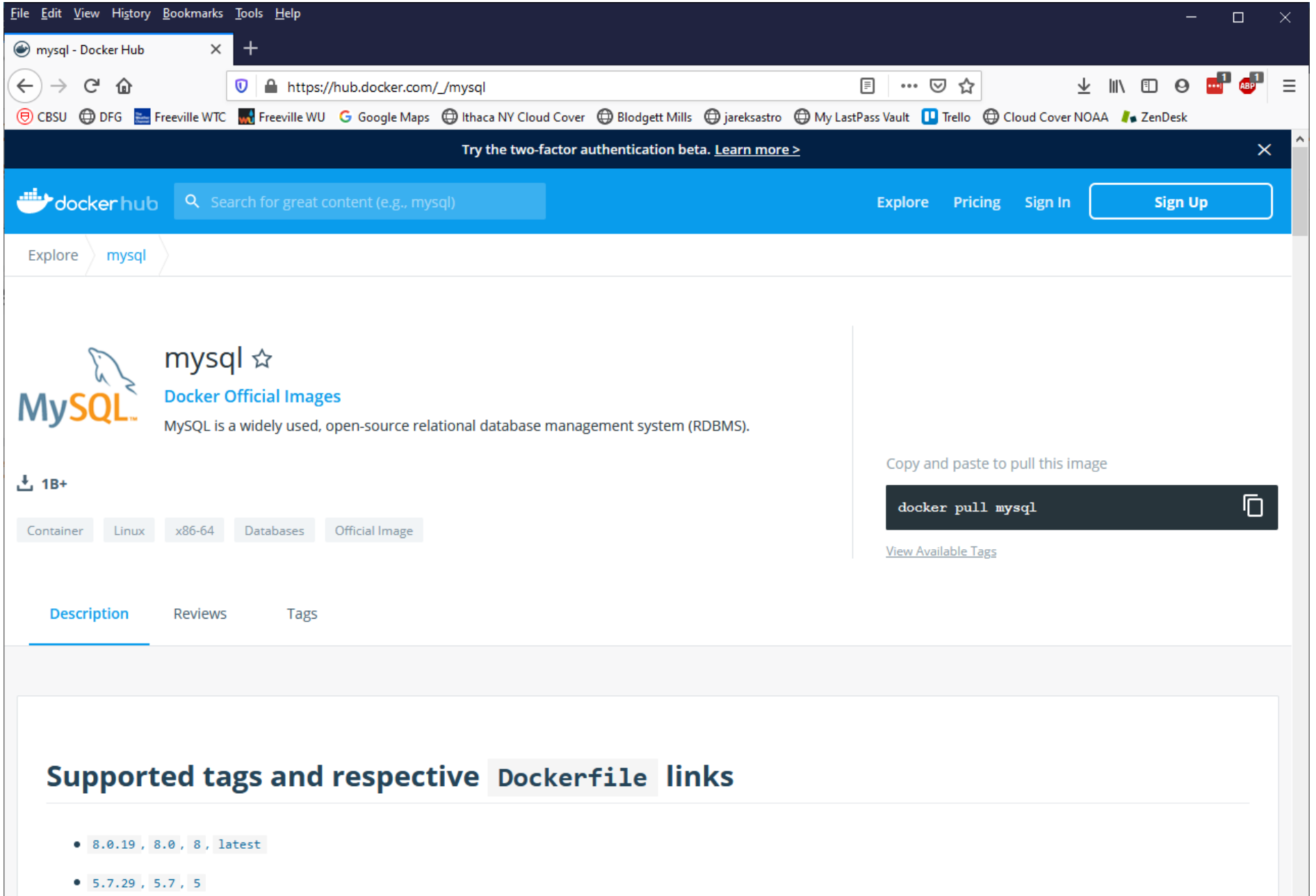
Sometimes such an image is too limited, and you will need to install it yourself anyway, but often it is useful

Usually worth a try.

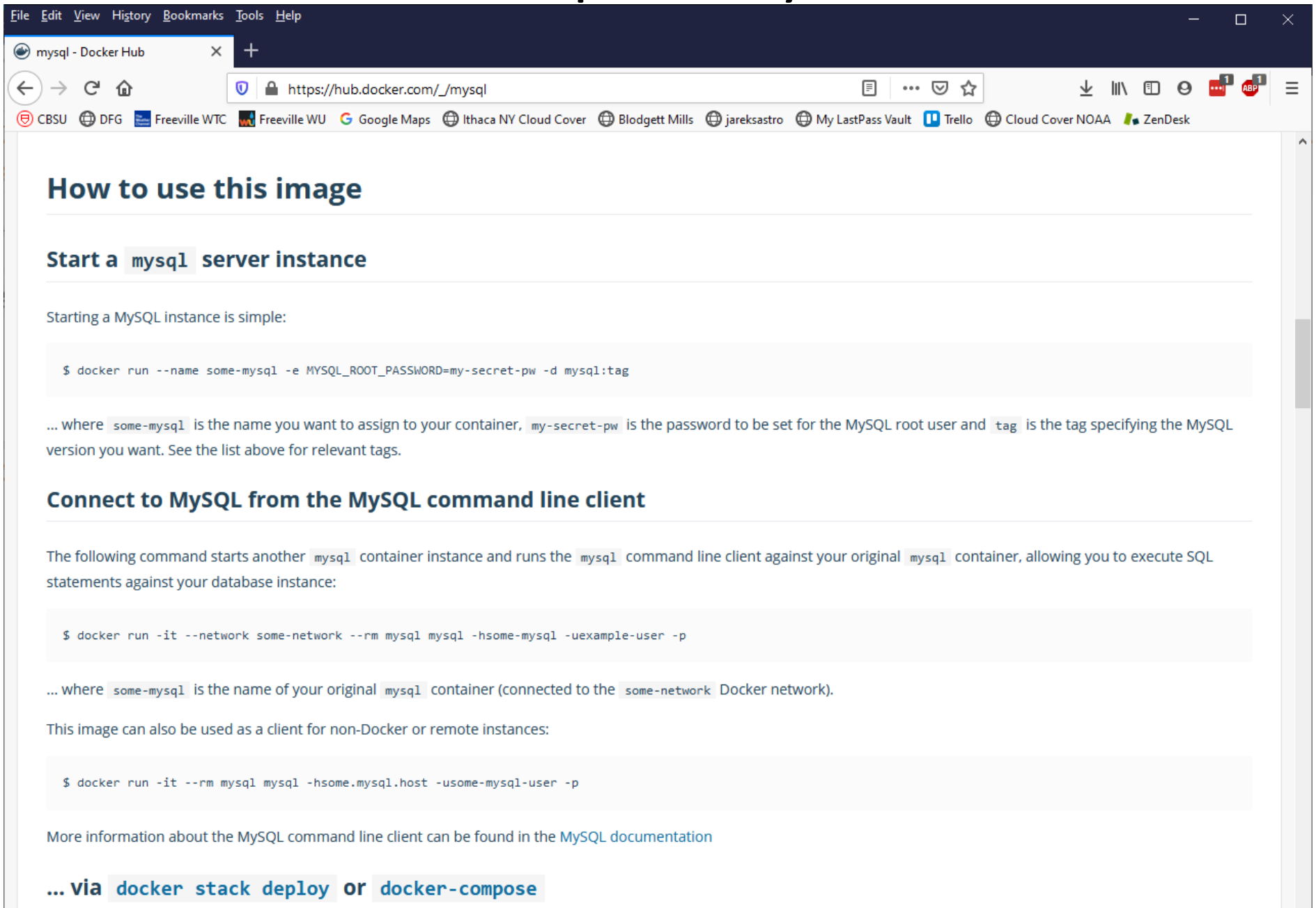
BioHPC Docker Example – MySQL Database Server

1. Search online for “MySQL and Docker”.
2. Turns out there is already a Docker image for MySQL
3. Follow instructions to pull and run the image
4. Customize it as needed
5. Test access form other applications.
6. Save image.

BioHPC Docker Example – MySQL Database Server

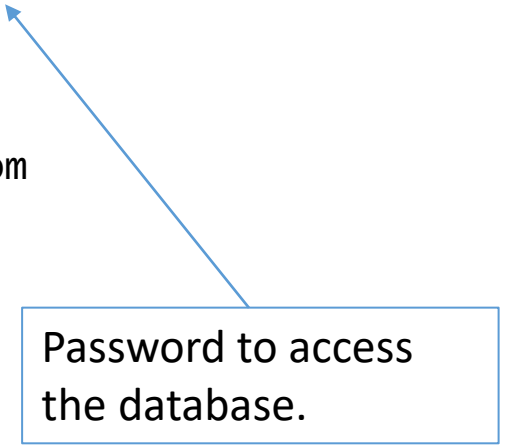


BioHPC Docker Example – MySQL Database Server



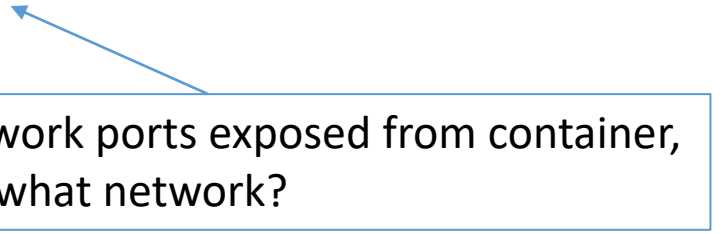
BioHPC Docker Example – MySQL Database Server

```
[jarekp@cbsum1c1b009 ~]$ docker1 run --name mysqlserver -e MYSQL_ROOT_PASSWORD=edeyewrg3 -d mysql
Unable to find image 'mysql:latest' locally
Trying to pull repository dtr.cucloud.net/mysql ...
Trying to pull repository docker.io/library/mysql ...
sha256:9643e9fbd6330d10686f8922292dcb20995e7b792c17d4e94ddf95255f1d5449: Pulling from
docker.io/library/mysql
54fec2fa59d0: Pull complete
bcc6c6145912: Pull complete
951c3d959c9d: Pull complete
05de4d0e206e: Pull complete
319f0394ef42: Pull complete
d9185034607b: Pull complete
013a9c64dadc: Pull complete
96d4c3d31f9f: Pull complete
785bc90808da: Pull complete
1339cf094729: Pull complete
beb8f531cc37: Pull complete
2b40c9f6a918: Pull complete
Digest: sha256:9643e9fbd6330d10686f8922292dcb20995e7b792c17d4e94ddf95255f1d5449
Status: Downloaded newer image for docker.io/mysql:latest
79b38279788c923b4aab7a3ae2830f28fa392f56e634714885482991a7c23177
[jarekp@cbsum1c1b009 ~]$
```



BioHPC Docker Example – MySQL Database Server

```
[jarekp@cbsum1c1b009 ~]$ docker1 ps -a
CONTAINER ID  IMAGE   COMMAND              CREATED          STATUS          PORTS                NAMES
79b38279788c  mysql   "docker-entryp.."   35 seconds ago   Up 34 seconds   3306/tcp, 33060/tcp  jarekp__biohpc_mysqlserver
[jarekp@cbsum1c1b009 ~]$
```



Before we can connect to our MySQL database remotely, we need to figure out what is our container IP address.

Docker maintains an internal network inside host usually 172.17.0.*. This network is NOT accessible from the outside of the server, but it is used for communication between containers and the host server.

Use command “docker1 inspect conatinerid” to find out more.

We have two ways to communicate with the database: get inside container or use network.

BioHPC Docker Example – MySQL Database Server

```
[jarekp@cbsum1c1b009 ~]$ docker1 exec -it jarekp__biohpc_mysqlserver /bin/bash
root@79b38279788c:/# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.19 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)
```

BioHPC Docker Example – MySQL Database Server

```
[jarekp@cbsum1c1b009 ~]$ docker1 inspect jarekp__biohpc_mysqlserver
[...]

"NetworkSettings": {
            "Bridge": "",
            "SandboxID": "b8e44e387c4e0cb7a63c4abe86386f72f0f1b1524854bdf570ac9921b541f63b",
            "HairpinMode": false,
            "LinkLocalIPv6Address": "",
            "LinkLocalIPv6PrefixLen": 0,
            "Ports": {
                "3306/tcp": null,
                "33060/tcp": null
            },
            "SandboxKey": "/var/run/docker/netns/b8e44e387c4e",
            "SecondaryIPAddresses": null,
            "SecondaryIPv6Addresses": null,
            "EndpointID": "7e64e46e4bb45e790114fff95e4c9ede3c2efc0d72f29e09363d76cdca21ee23",
            "Gateway": "172.17.0.1",
            "GlobalIPv6Address": "",
            "GlobalIPv6PrefixLen": 0,
            "IPAddress": "172.17.0.2",
            "IPPrefixLen": 16,
```

BioHPC Docker Example – MySQL Database Server

```
[jarekp@cbsum1c1b009 ~]$ mysql -u root -p -h 172.17.0.2
Enter password:
ERROR 2059 (HY000): Authentication plugin 'caching_sha2_password' cannot be loaded:
/usr/lib64/mysql/plugin/caching_sha2_password.so: cannot open shared object file: No such file or directory
[jarekp@cbsum1c1b009 ~]$
```

Something is wrong!

What happened is that the local “mysql” program uses different type of passwords than the container (legacy vs sha2)

If you search Internet you will find it is a common problem, and it has a common solution: modify password type :-)

BioHPC Docker Example – MySQL Database Server

```
[jarekp@cbsum1c1b009 ~]$ docker1 exec -it jarekp__biohpc_mysqlserver /bin/bash
root@79b38279788c:/# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.19 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> ALTER USER 'root'@'%' IDENTIFIED WITH mysql_native_password BY 'edeyewrg3';
Query OK, 0 rows affected (0.04 sec)

mysql> quit
Bye
root@79b38279788c:/# exit
exit
[jarekp@cbsum1c1b009 ~]$
```

BioHPC Docker Example – MySQL Database Server

```
[jarekp@cbsum1c1b009 ~]$ mysql -u root -p -h 172.17.0.2
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.19 MySQL Community Server - GPL

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)
```

BioHPC Docker Example – MySQL Database Server

You can run MySQL container with native port 3306 mapped to an external port from range 8009-8019. This would allow to connect to the database from anywhere on campus network. Similar as with our web server example.