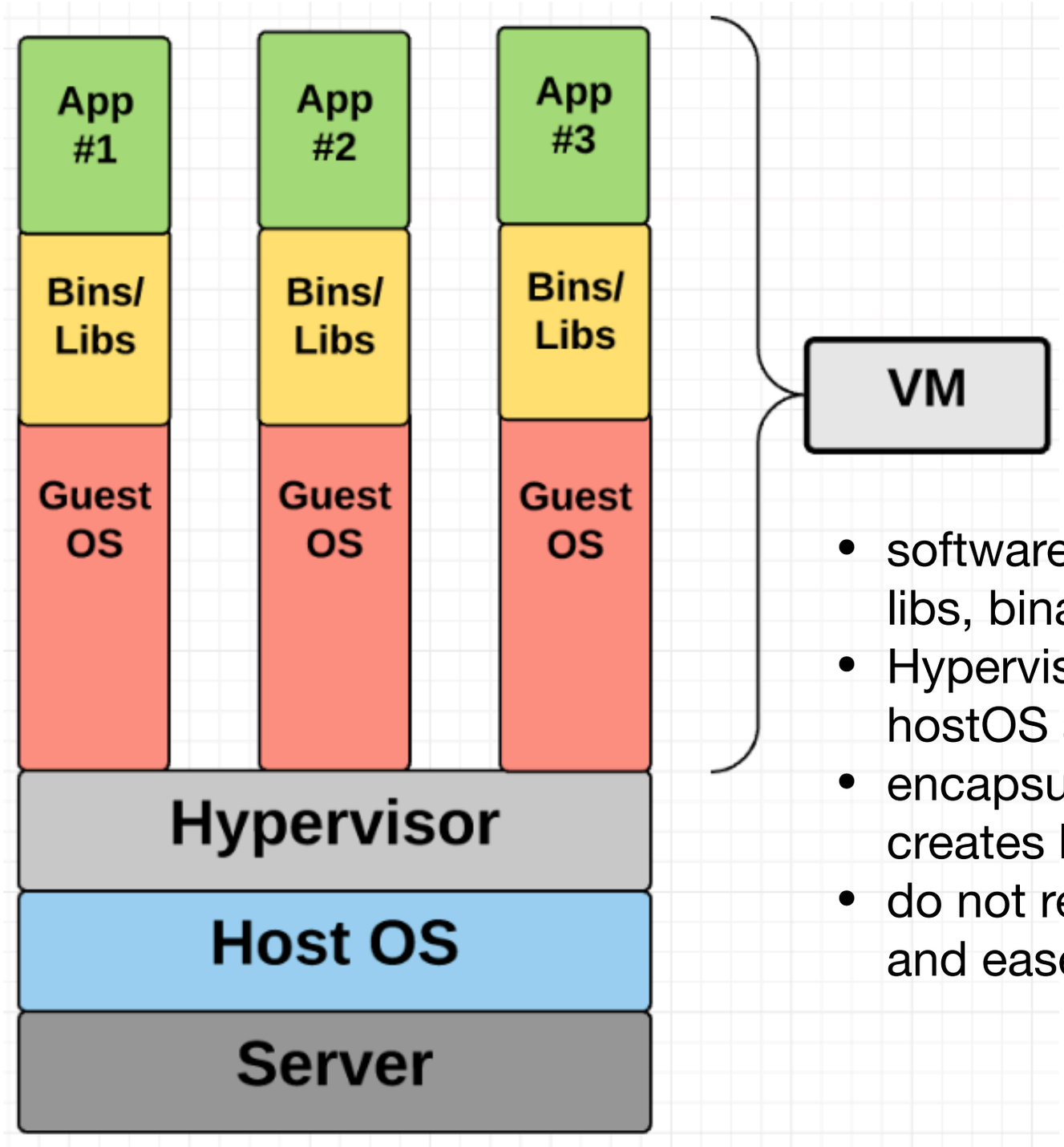
Introduction to docker

Chris Stricker, stricker@genetics-network.ch

- Software needs to interact with other soft- and hardware to successfully run on a computer
  - we want to access the hard disk for file I/O, the printer, libraries for scientific computing, …
  - interactions between software and/or software and hardware
  - dependencies may be different between different software products, creating incompatibilities (DLL incompatibilities in my old windows days)
  - even worse: we want our software to run on different hardware, different operating systems

- What if we could encapsulate the software and isolate it as much as possible from other software?
  - virtual machines have been around for almost 20 years now
  - “computer within a computer”
  - provide maximum isolation at the expense of overhead



- software is encapsulated with specific libs, binaries and a guest OS
- Hypervisor as additional layer between hostOS and guestOS
- encapsulating software with guest OS creates large overhead
- do not really contribute to portability and ease of maintenance

◦ https://www.aquasec.com/wiki/display/containers/Docker+Containers+vs.+Virtual+Machines

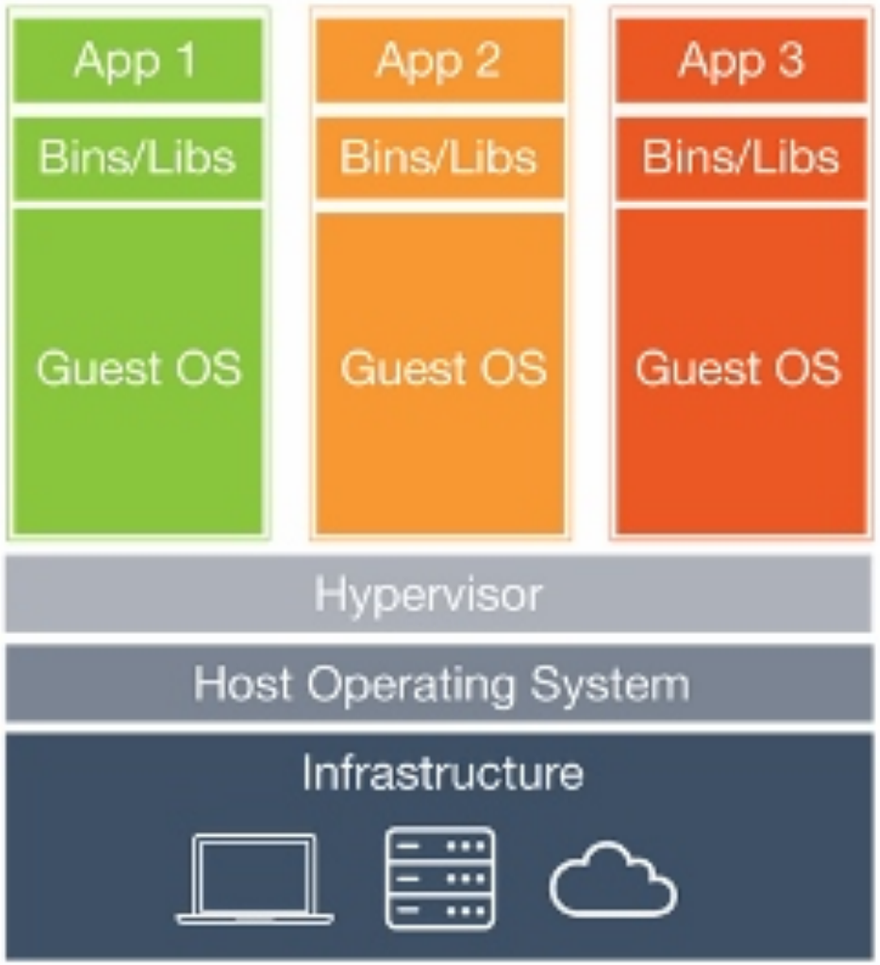
- What if we could encapsulate the software and isolate it as much as possible from other software …

  - but can bundle it only with the necessary components, i.e. an ideal balance between isolation and overhead

  - containers

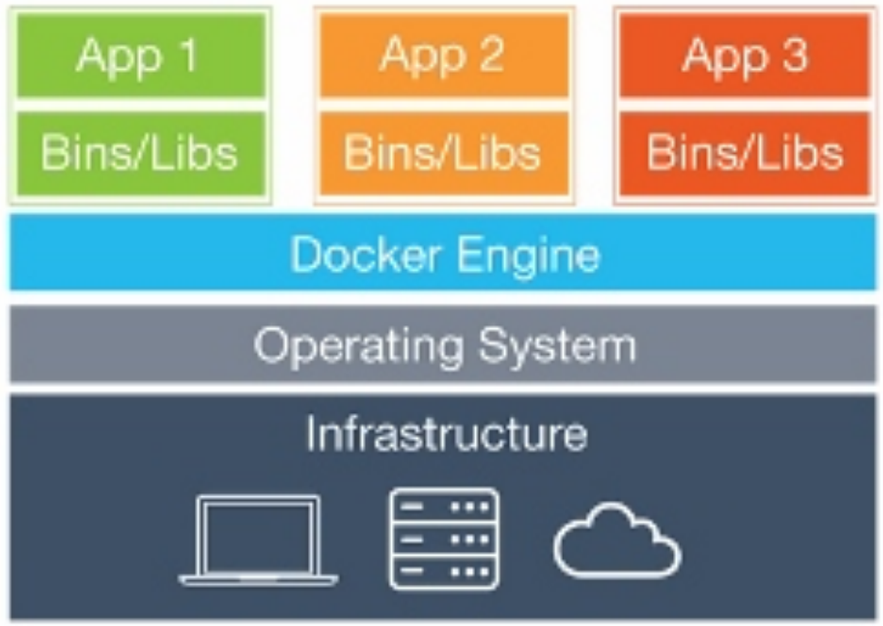
  - balance between isolation and overhead

  - containers share segments of the hostOS



Virtual Machines

- software encapsulated with specific libs, binaries and a guest OS
- Hypervisor as additional layer between hostOS and guestOS
- encapsulating software with guest OS creates large overhead
- do not really contribute to portability and ease of maintenance



Containers

- software encapsulated with specific libs, binaries
- container daemon interacts with kernel
- container isolated form OS
- portability and ease of maintenance
- stateless
- immutable
-

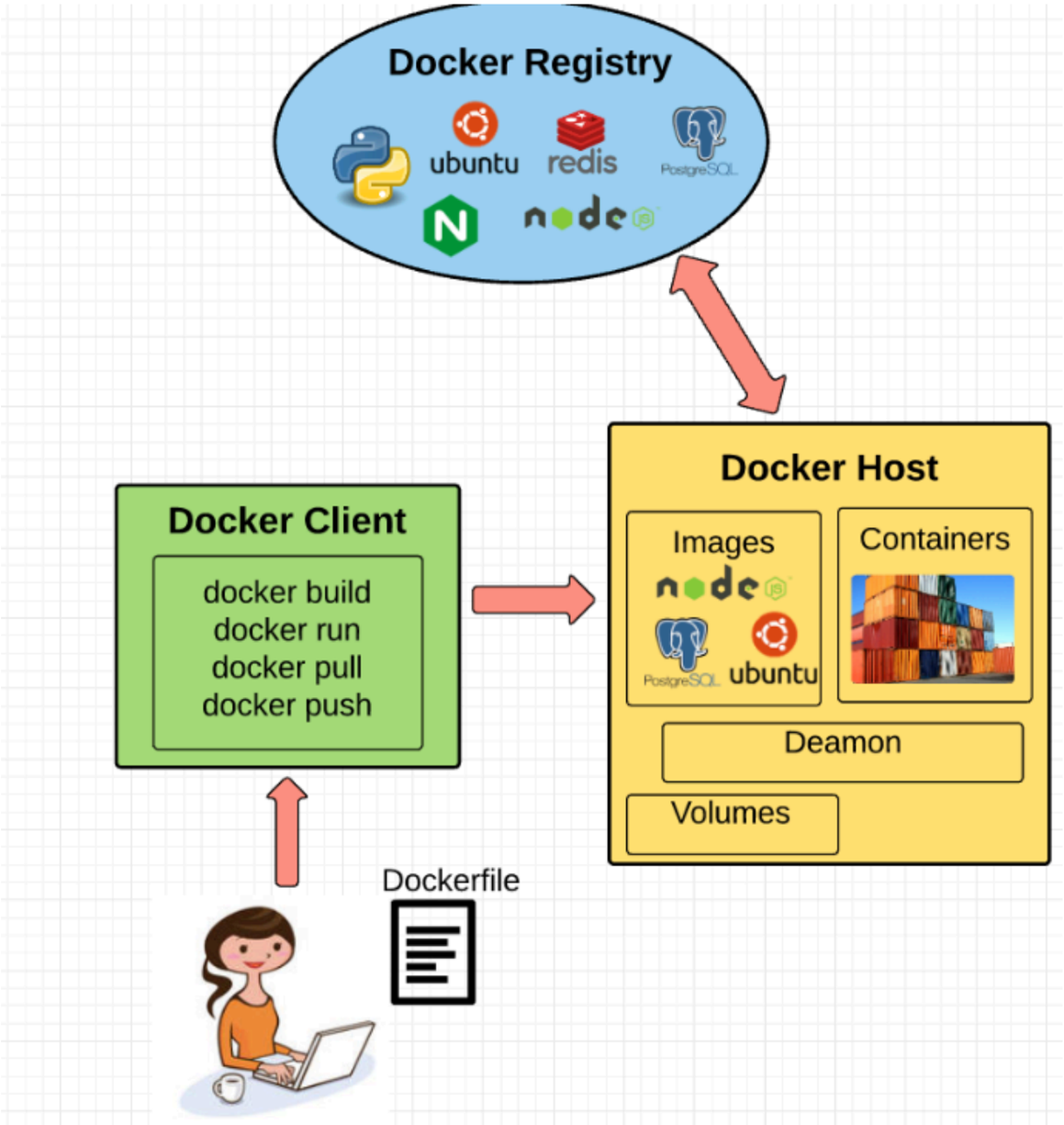
Getting started with Docker

- you downloaded the docker client from https://www.docker.com/get-started and installed it

```
[al5:~] Chris% docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
d1725b59e92d: Pull complete
Digest: sha256:0add3ace90ecb4adbf7777e9aacf18357296e799f81cabc9fde470971e499788
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Fundamental Concepts

- docker client is the user interface
- docker daemon runs on host, executes the command coming form the client
- the docker file contains instructions to build a docker image
- a docker image is the blueprint of a container, it become a container, when the client requests to start the image. A container is an instance of an image.
- software is wrapped into a container with a file system, libs, tools, etc.
- images are read only, so read-write volumes are added also
- see https://medium.freecodecamp.org/a-beginner-friendly-introduction-to-containers-vms-and-docker-79a9e3e119b



Available docker images

- what containers are currently on your system?

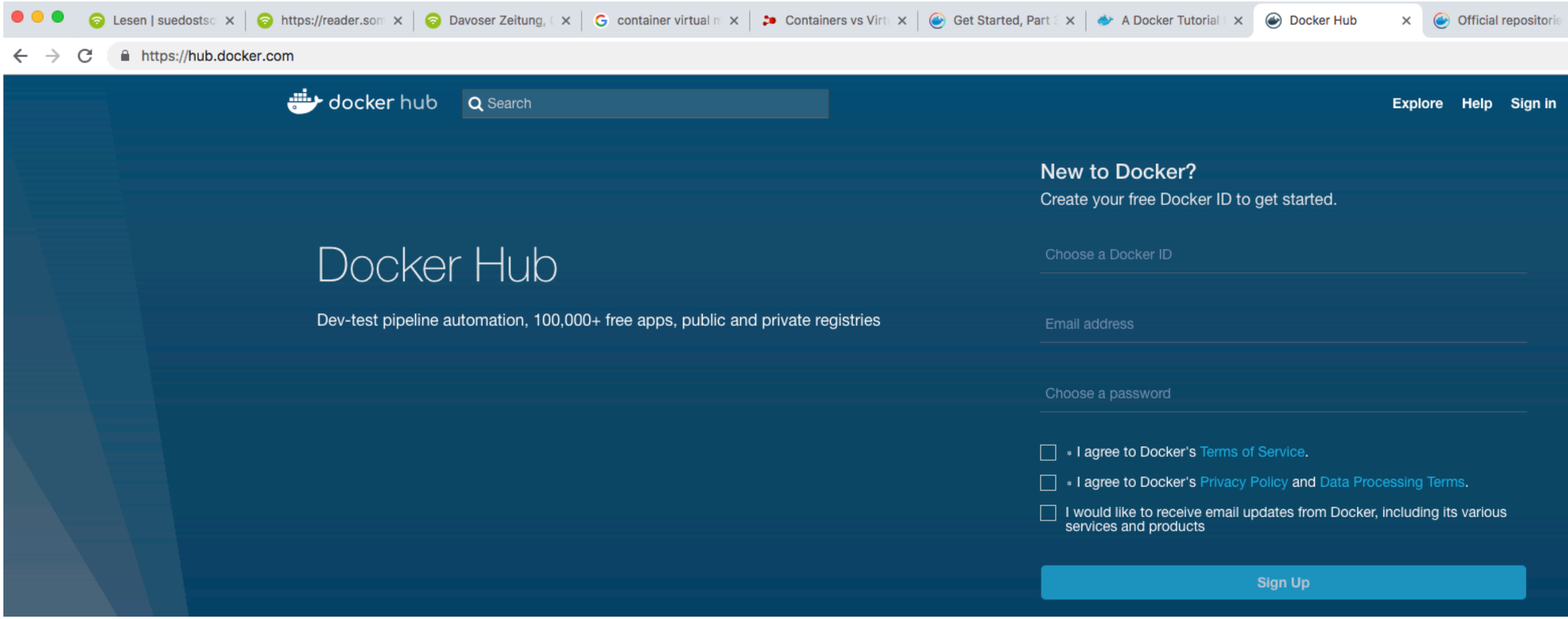
```
[al5:~] Chris% docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS               NAMES
08b7d4fec6a9        hello-world         "/hello"            17 hours ago        Exited (0) 17 hours ago                         quirky_tesla
```

- what images are currently on your system?

```
[al5:~] Chris% docker images
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
hello-world             latest              4ab4c602aa5e        2 months ago        1.84kB
jwas                    latest              a0b8b84eb779        4 months ago        5GB
qtlrocks/jwas-docker    latest              6d66776c87a7        5 months ago        999MB
```

Available docker images

- what images are available on the docker hub?
  - https://hub.docker.com



https://hub.docker.com/search/?isAutomated=0&isOfficial=0&page=1&pullCount=0&q=ubuntu&starCount=1

ubuntu

Dashboard Explore Organizations Create cstricker

Repositories (52973)

Stars

ubuntu official	8.7K STARS	10M+ PULLS	DETAILS
phusion/baseimage public	1.5K STARS	5M+ PULLS	DETAILS
alexeiled/docker-oracle-xe-11g public | automated build	276 STARS	100K+ PULLS	DETAILS
dorowu/ubuntu-desktop-lxde-vnc public | automated build	241 STARS	1M+ PULLS	DETAILS
williamyeh/ansible public | automated build	186 STARS	1M+ PULLS	DETAILS
rastasheep/ubuntu-sshd public | automated build	180 STARS	1M+ PULLS	DETAILS

```
[al5:~] Chris% docker run ubuntu
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
473ede7ed136: Pull complete
c46b5fa4d940: Pull complete
93ae3df89c92: Pull complete
6b1eed27cade: Pull complete
Digest: sha256:29934af957c53004d7fb6340139880d23fb1952505a15d69a03af0d1418878cb
Status: Downloaded newer image for ubuntu:latest
[al5:~] Chris%
```

image started as container but immediately exited

```
[al5:~] Chris% docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED              STATUS
3cbbb3db66fa        ubuntu              "/bin/bash"         About a minute ago   Exited (0) 59 seconds ago
[al5:~] Chris%
```

- now try

```
[al5:~] Chris% docker run -itd ubuntu
c866e65f29788ef0247bb79ca67392800def276bebd17114d431c9a9c87be04b
[al5:~] Chris% docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
c866e65f2978        ubuntu              "/bin/bash"         10 seconds ago      Up 8 seconds
3cbbb3db66fa        ubuntu              "/bin/bash"         2 minutes ago       Exited (0) 2 minutes ago
[al5:~] Chris%
```

  - spawns a new ubuntu container from the ubuntu image, keeps it alive (-i), connects it to a terminal (-t) but detaches it from the current terminal (-d).

- Now login to that container and do something…

```
[al5:~] Chris% docker exec -it c866e65f2978 bash
root@c866e65f2978:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@c866e65f2978:/# mkdir MYDIR
root@c866e65f2978:/# ls
MYDIR  bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@c866e65f2978:/# exit
exit
[al5:~] Chris% docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS
c866e65f2978        ubuntu              "/bin/bash"         2 minutes ago       Up 2 minutes
3cbbb3db66fa        ubuntu              "/bin/bash"         4 minutes ago       Exited (0) 4 minutes ago
[al5:~] Chris% docker exec -it c866e65f2978 bash
root@c866e65f2978:/# ls
MYDIR  bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@c866e65f2978:/# exit
exit
[al5:~] Chris%
```

Changes are persistent, as long as container exists!

- can verify this, stop and restart the container

```
[al5:~] Chris% docker stop c866e65f2978
c866e65f2978
[al5:~] Chris% docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS
c866e65f2978        ubuntu              "/bin/bash"         4 minutes ago       Exited (0) 6 seconds ago
3cbbb3db66fa        ubuntu              "/bin/bash"         7 minutes ago       Exited (0) 7 minutes ago
```

- Now start it again…

```
[al5:~] Chris% docker start -i c866e65f2978
root@c866e65f2978:/# ls
MYDIR  bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@c866e65f2978:/#
```

Changes are persistent, as long as container exists!
However, starting a new container from the ubuntu image will not have MYDIR in it

- create a new ubuntu container

```
root@c866e65f2978:/# exit
exit
[al5:~] Chris% docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS
c866e65f2978        ubuntu              "/bin/bash"         17 minutes ago      Exited (0) 13 seconds ago
3cbbb3db66fa        ubuntu              "/bin/bash"         19 minutes ago      Exited (0) 19 minutes ago
[al5:~] Chris% docker run -itd ubuntu
ad96279dd9423e3cd8548e05418b87f20adfbfed86a33c955392fbb2f0fc9df9
[al5:~] Chris% docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS
ad96279dd942        ubuntu              "/bin/bash"         14 seconds ago      Up 12 seconds
c866e65f2978        ubuntu              "/bin/bash"         17 minutes ago      Exited (0) 50 seconds ago
3cbbb3db66fa        ubuntu              "/bin/bash"         20 minutes ago      Exited (0) 20 minutes ago
```

- see whether it has MYDIR?

```
[al5:~] Chris% docker exec -it ad96279dd942 bash
root@ad96279dd942:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@ad96279dd942:/# exit
exit
[al5:~] Chris% docker rm ad96279dd942
Error response from daemon: You cannot remove a running container ad96279dd9423e3cd8548e05418b87f20adfbfed
[al5:~] Chris% docker stop ad96279dd942
ad96279dd942
[al5:~] Chris% docker rm ad96279dd942
ad96279dd942
[al5:~] Chris% docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
c866e65f2978        ubuntu              "/bin/bash"         19 minutes ago      Exited (0) 2 minutes ago
3cbbb3db66fa        ubuntu              "/bin/bash"         21 minutes ago      Exited (0) 21 minutes ago
[al5:~] Chris%
```

Changes are persistent, as long as container exists!
However, starting a new container from the ubuntu image will not have MYDIR in it

—> Containers are immutable & stateless

- equivalently, try

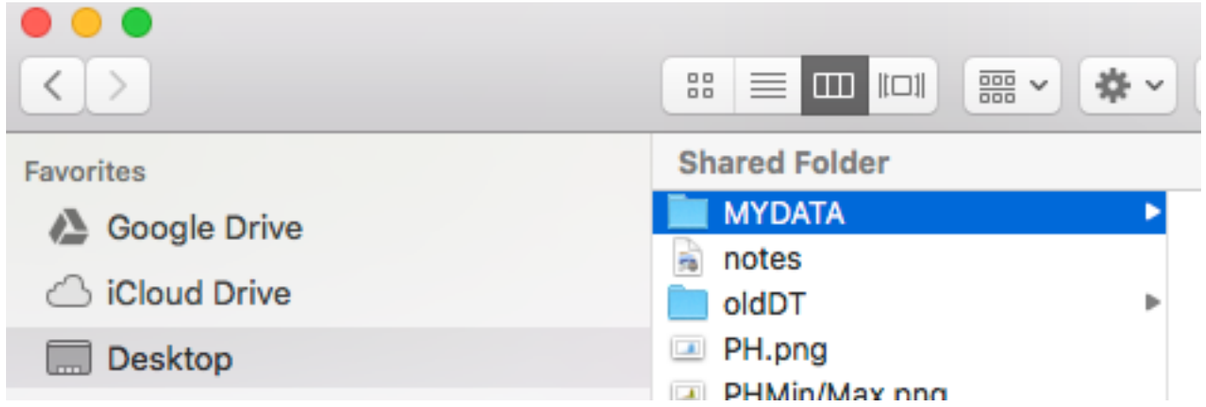
```
[al5:~] Chris% docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS
c866e65f2978        ubuntu              "/bin/bash"         26 minutes ago      Exited (0) 9 minutes ago
3cbbb3db66fa        ubuntu              "/bin/bash"         28 minutes ago      Exited (0) 28 minutes ago
[al5:~] Chris% docker run --rm -it ubuntu bash
root@91ec62e5458f:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@91ec62e5458f:/# mkdir MYDIR
root@91ec62e5458f:/# ls
MYDIR  bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@91ec62e5458f:/# exit
exit
[al5:~] Chris% docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS
c866e65f2978        ubuntu              "/bin/bash"         27 minutes ago      Exited (0) 10 minutes ago
3cbbb3db66fa        ubuntu              "/bin/bash"         30 minutes ago      Exited (0) 30 minutes ago
[al5:~] Chris%
```

Changes are lost, when the container is removed!
—> Container images are immutable & stateless

- how can we save our work, the data we created?

```
[al5:~] Chris% docker run --rm -it -v ~/Desktop/MYDATA/:/MYDir ubuntu bash
root@bcb6276780df:/# ls
MYDir  bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
```

- the new dir MYDir in the container is linked to MYDATA on the local desktop




```
root@bcb6276780df:/# ls -al > MYDir/listing
root@bcb6276780df:/# cat MYDir/listing
total 72
drwxr-xr-x   1 root root 4096 Nov 16 11:19 .
drwxr-xr-x   1 root root 4096 Nov 16 11:19 ..
-rwxr-xr-x   1 root root    0 Nov 16 11:19 .dockerenv
drwxr-xr-x   3 root root  102 Nov 16 09:49 MYDir
drwxr-xr-x   2 root root 4096 Oct 18 21:03 bin
drwxr-xr-x   2 root root 4096 Apr 24  2018 boot
drwxr-xr-x   5 root root  360 Nov 16 11:19 dev
drwxr-xr-x   1 root root 4096 Nov 16 11:19 etc
drwxr-xr-x   2 root root 4096 Apr 24  2018 home
drwxr-xr-x   8 root root 4096 Oct 18 21:02 lib
drwxr-xr-x   2 root root 4096 Oct 18 21:02 lib64
drwxr-xr-x   2 root root 4096 Oct 18 21:02 media
drwxr-xr-x   2 root root 4096 Oct 18 21:02 mnt
drwxr-xr-x   2 root root 4096 Oct 18 21:02 opt
dr-xr-xr-x 317 root root    0 Nov 16 11:19 proc
drwx------   2 root root 4096 Oct 18 21:03 root
drwxr-xr-x   1 root root 4096 Oct 19 00:47 run
drwxr-xr-x   1 root root 4096 Oct 19 00:47 sbin
drwxr-xr-x   2 root root 4096 Oct 18 21:02 srv
dr-xr-xr-x  13 root root    0 Nov 15 12:14 sys
drwxrwxrwt   2 root root 4096 Oct 18 21:03 tmp
drwxr-xr-x   1 root root 4096 Oct 18 21:02 usr
drwxr-xr-x   1 root root 4096 Oct 18 21:03 var
```

- after exiting the container…

```
root@bcb6276780df:/# exit
exit
[al5:~] Chris% cat ~/Desktop/MYDATA/listing
total 72
drwxr-xr-x   1 root root 4096 Nov 16 11:19 .
drwxr-xr-x   1 root root 4096 Nov 16 11:19 ..
-rwxr-xr-x   1 root root    0 Nov 16 11:19 .dockerenv
drwxr-xr-x   3 root root  102 Nov 16 09:49 MYDir
drwxr-xr-x   2 root root 4096 Oct 18 21:03 bin
drwxr-xr-x   2 root root 4096 Apr 24  2018 boot
drwxr-xr-x   5 root root  360 Nov 16 11:19 dev
drwxr-xr-x   1 root root 4096 Nov 16 11:19 etc
drwxr-xr-x   2 root root 4096 Apr 24  2018 home
drwxr-xr-x   8 root root 4096 Oct 18 21:02 lib
drwxr-xr-x   2 root root 4096 Oct 18 21:02 lib64
drwxr-xr-x   2 root root 4096 Oct 18 21:02 media
drwxr-xr-x   2 root root 4096 Oct 18 21:02 mnt
drwxr-xr-x   2 root root 4096 Oct 18 21:02 opt
dr-xr-xr-x 317 root root    0 Nov 16 11:19 proc
drwx------   2 root root 4096 Oct 18 21:03 root
```

Where are we?

- we learned:
  - docker is a way to isolate software into a self-contained box, a container
  - there are basic containers available on a docker hub
  - can download and run such a container
  - can add functionality to a container
- next: we want to save those changes and upload such a container

- Note that by

```
[al5:~] Chris% docker run --rm -it -v ~/Desktop/MYDATA/:/MYDir ubuntu bash
root@bcb6276780df:/# ls
MYDir  bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
```

- we were spawning a new container from the ubuntu image, created the new directory “MYDIR” in the container and linked it to a directory on our desktop.

- However, the container was deleted after exiting. Therefore, let’s do it again, but this time without ‘—rm’…

```
[al5:~] Chris% docker run  -it -v ~/Desktop/MYDATA/:/MYDir ubuntu bash
root@b18e135cbd91:/# ls
MYDir  bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@b18e135cbd91:/# ls MYDir/
listing
root@b18e135cbd91:/# exit
exit
[al5:~] Chris%
```

- now the new container is still available

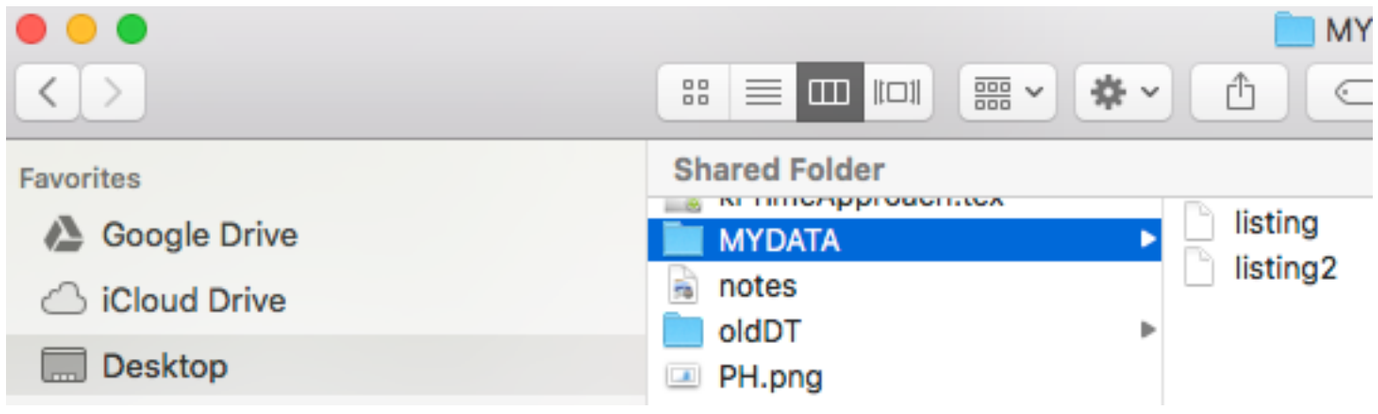
```
[al5:~] Chris% docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS
b18e135cbd91        ubuntu              "bash"              27 minutes ago      Exited (0) 4 minutes ago
c866e65f2978        ubuntu              "/bin/bash"         2 days ago          Exited (0) 27 minutes ago
3cbbb3db66fa        ubuntu              "/bin/bash"         2 days ago          Exited (0) 2 days ago
[al5:~] Chris%
```

- and the directory MYDir has persisted and is linked to to MYDATA on the desktop!

- and the directory MYDir has persisted and is linked to MYDATA on the desktop!

```
[al5:~] Chris% docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS
b18e135cbd91        ubuntu              "bash"              3 hours ago         Exited (0) 3 seconds ago
c866e65f2978        ubuntu              "/bin/bash"         3 days ago          Exited (0) 3 hours ago
3cbbb3db66fa        ubuntu              "/bin/bash"         3 days ago          Exited (0) 3 days ago
[al5:~] Chris% docker start -i b18e135cbd91
[root@b18e135cbd91:/# ls
MYDir  bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
[root@b18e135cbd91:/# ls >MYDir/listing2
[root@b18e135cbd91:/# cat MYDir/listing2
MYDir
bin
boot
dev
etc
home
```

- and MYDATA on my Desktop looks like



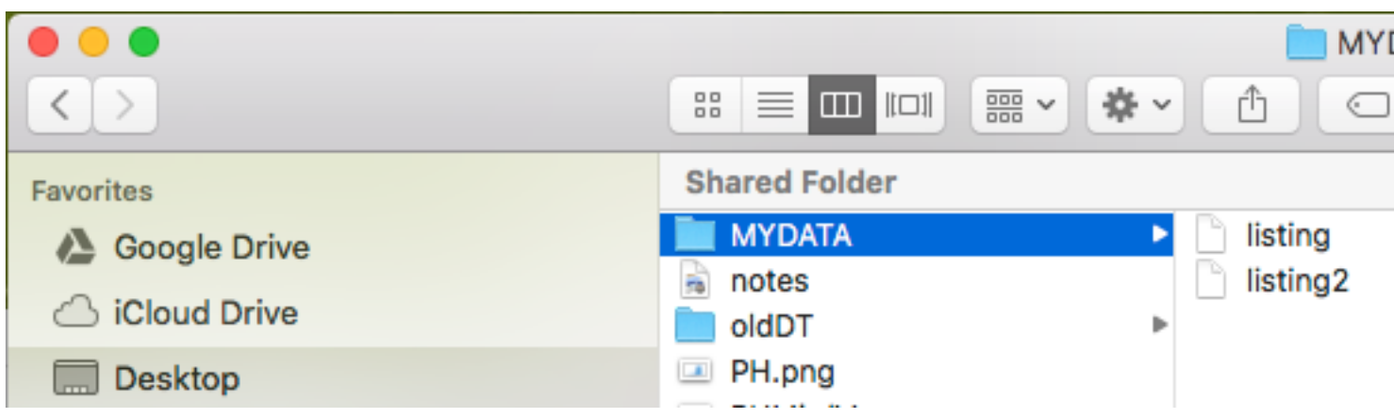
- now let’s commit those changes to a new image

- now let’s commit those changes to a new image

```
[al5:~] Chris% docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                          PORTS
b18e135cbd91        ubuntu              "bash"              3 hours ago         Exited (0) 9 minutes ago
c866e65f2978        ubuntu              "/bin/bash"         3 days ago          Exited (0) 3 hours ago
3cbbb3db66fa        ubuntu              "/bin/bash"         3 days ago          Exited (0) 3 days ago
[al5:~] Chris% docker images
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
ubuntu                  latest              ea4c82dcd15a        4 weeks ago         85.8MB
hello-world             latest              4ab4c602aa5e        2 months ago        1.84kB
jwas                    latest              a0b8b84eb779        5 months ago        5GB
qtlrocks/jwas-docker    latest              6d66776c87a7        5 months ago        999MB
[al5:~] Chris% docker commit b18e135cbd91 chris-ubuntu
sha256:0bce066c853a720c7a4e3799cc0c2a64db1e0b415329465632de674360db3ae1
[al5:~] Chris% docker run -it chris-ubuntu
[root@9c24685c01c5:/# ls
MYDir  bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
```

- spawning a new container from the modified images reveals that the changes are still there (dir MYDir), BUT

```
[root@9c24685c01c5:/# cd MYDir/
[root@9c24685c01c5:/MYDir# ls -al
total 8
drwxr-xr-x 2 root root 4096 Nov 19 09:34 .
drwxr-xr-x 1 root root 4096 Nov 19 12:29 ..
[root@9c24685c01c5:/MYDir# cd ..
[root@9c24685c01c5:/# ls /usr > MYDir/listing3
[root@9c24685c01c5:/# ls MYDir/
listing3
```



- the link between the dir MYDir in the container and our local MYDATA-directory is not saved with to the newly committed image.

- we need to re-establish that link upon spawning the new container

```
[al5:~] Chris% docker run -it -v ~/Desktop/MYDATA/:/MYDir chris-ubuntu
[root@50b58ccf6732:/# ls MYDir/
 listing  listing2
 root@50b58ccf6732:/#
```

- now we can push our modified image to the docker hub for others to download

- now we can push our modified image to the docker hub for others to download

```
[al5:~] Chris% docker push chris-ubuntu
The push refers to repository [docker.io/library/chris-ubuntu]
3ba479ced88d: Preparing
76c033092e10: Preparing
2146d867acf3: Preparing
ae1f631f14b7: Preparing
102645f1cf72: Preparing
denied: requested access to the resource is denied
```

- we need to tag the image and login to docker hub

```
[al5:~] Chris% docker tag chris-ubuntu cstricker/chris-ubuntu
[al5:~] Chris% docker images
REPOSITORY                TAG                 IMAGE ID            CREATED             SIZE
chris-ubuntu              latest              0bce066c853a        About an hour ago   85.8MB
cstricker/chris-ubuntu    latest              0bce066c853a        About an hour ago   85.8MB
ubuntu                    latest              ea4c82dcd15a        4 weeks ago         85.8MB
hello-world               latest              4ab4c602aa5e        2 months ago        1.84kB
jwas                      latest              a0b8b84eb779        5 months ago        5GB
qtlrocks/jwas-docker      latest              6d66776c87a7        5 months ago        999MB


[al5:~] Chris% docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over
Username: cstricker
Password:
Login Succeeded
```

- now, uploading works

```
[al5:~] Chris% docker push cstricker/chris-ubuntu
The push refers to repository [docker.io/cstricker/chris-ubuntu]
3ba479ced88d: Pushed
76c033092e10: Pushed
2146d867acf3: Pushed
ae1f631f14b7: Pushed
102645f1cf72: Pushed
latest: digest: sha256:2295a0d59e78a80bdee35a471adc52454eb1a05aaffe1551ac129e55381386ce size: 1357
```

- docker uploads layers of modifications, not the whole modified image
- can create a new image from scratch by a docker file
  - we may need to update the software in our container
  - cookbook of our image

Docker file for JWAS image

(https://github.com/reworkhow/JWAS-Docker/blob/master/dockerfile-jwas-large/Dockerfile)

```
ARG BASE_CONTAINER=jupyter/datascience-notebook
FROM $BASE_CONTAINER

LABEL maintainer="Hao Cheng <qtlcheng@ucdavis.edu>"

# Add JWAS, XSim and PyPlot packages.
RUN    julia -e 'using Pkg; Pkg.update()' && \
       julia -e 'using Pkg; Pkg.add("PyPlot")' && \
       julia -e 'using Pkg; Pkg.add("Plots")' && \
       julia -e 'using Pkg; Pkg.add("DataFrames")' && \
       julia -e 'using Pkg; Pkg.add("CSV")' && \
       julia -e 'using Pkg; Pkg.add("Distributions")' && \
       julia -e 'using Pkg; Pkg.add("JWAS")' && \
       julia -e 'using Pkg; Pkg.add("XSim")' && \
        # Precompile JWAS  \
       julia -e 'using Pkg; using JWAS' && \
       julia -e 'using Pkg; using XSim' && \
       julia -e 'using Pkg; using Plots' && \
       julia -e 'using Pkg; using CSV' && \
       julia -e 'using Pkg; using DataFrames' && \
       fix-permissions $JULIA_PKGDIR $CONDA_DIR/share/jupyter

ADD --chown=jovyan:users ./notebooks /home/jovyan/notebooks
```

what did we learn?

- docker is a way to isolate software into a self-contained box, a container
- there are basic containers available on a docker hub
- download and run such a container
- add functionality to a container
- save those changes to an image
- upload to docker hub
- create our own containers via docker files
- automated connection between docker hub and GitHub possible

what did we learn?

- most useful docker commands
  - docker ps -a
  - docker pull
  - docker build
  - docker run
  - docker logs
  - docker volume ls
  - docker rm
  - docker rmi
  - docker stop / start
- lots of tutorials online, e.g. https://docker-curriculum.com/

Using the JWAS docker container…

but Jupyter first….

Chris Stricker, stricker@genetics-network.ch

Pulling the container from docker hub

```
[al5:~] Chris% docker pull qtlrocks/jwas-docker
Using default tag: latest
latest: Pulling from qtlrocks/jwas-docker
a48c500ed24e: Already exists
1e1de00ff7e1: Already exists
0330ca45a200: Already exists
471db38bcfbf: Already exists
0b4aba487617: Already exists
27261a5cbf78: Pull complete
2773b009b29a: Pull complete
0915dfd6dae3: Pull complete
ee7dca77f9ed: Pull complete
cf5be5e11621: Pull complete
33aa38e94d28: Pull complete
68f4b2245d2a: Pull complete
f5ab1b831f77: Pull complete
7ba257ca073a: Pull complete
0eb8f950b3ea: Pull complete
7774c2ec4751: Pull complete
2b9f5766a263: Pull complete
e4052009a673: Pull complete
f9e7255ce189: Pull complete
267a2399516f: Pull complete
3b54695138ea: Pull complete
d23336e5273c: Pull complete
e612b1583510: Pull complete
35ccca5c04fe: Pull complete
89cf804a40ff: Pull complete
81cdf7ccfcd1: Pull complete
2d79f9fc672b: Pull complete
e68a6f6ec56b: Pull complete
889930c80fd2: Pull complete
4c9509f294c8: Pull complete
14e58f71fd50: Pull complete
Digest: sha256:7a6d83f972ed725385d146a765f816d0d02330917446becfe1fc9a00b13a4408
Status: Downloaded newer image for qtlrocks/jwas-docker:latest
[al5:~] Chris%
```

starting the container from the pulled image

```
[al5:~] Chris% docker images
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
chris-ubuntu            latest              0bce066c853a        22 hours ago        85.8MB
qtlrocks/jwas-docker    latest              96dc980a3a55        5 days ago          6.26GB
ubuntu                  latest              ea4c82dcd15a        4 weeks ago         85.8MB
hello-world             latest              4ab4c602aa5e        2 months ago        1.84kB
jwas                    latest              a0b8b84eb779        5 months ago        5GB
qtlrocks/jwas-docker    <none>              6d66776c87a7        5 months ago        999MB
[al5:~] Chris%
```

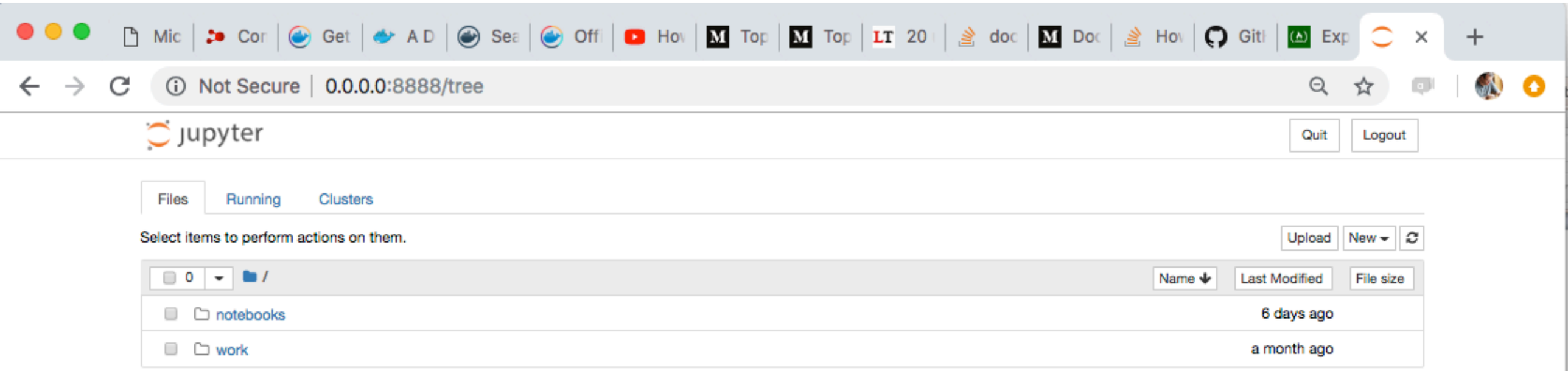
```
[al5:~] Chris% docker run -it --rm -p 8888:8888 -v /Users/Chris/agn/TUM/JuliaCourse/dataDay1:/home/jovyan/work qtlrocks/jwas-docker
Container must be run with group "root" to update passwd file
Executing the command: jupyter notebook
[I 13:23:20.023 NotebookApp] Writing notebook server cookie secret to /home/jovyan/.local/share/jupyter/runtime/notebook_cookie_secr
[I 13:23:20.380 NotebookApp] JupyterLab extension loaded from /opt/conda/lib/python3.6/site-packages/jupyterlab
[I 13:23:20.380 NotebookApp] JupyterLab application directory is /opt/conda/share/jupyter/lab
[I 13:23:20.392 NotebookApp] Serving notebooks from local directory: /home/jovyan
[I 13:23:20.393 NotebookApp] The Jupyter Notebook is running at:
[I 13:23:20.393 NotebookApp] http://(3a7300f0e64d or 127.0.0.1):8888/?token=86f15cc77730d704ac931052cc66cd5ef0a390064eb4a2e7
[I 13:23:20.393 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 13:23:20.394 NotebookApp]

    Copy/paste this URL into your browser when you connect for the first time,
    to login with a token:
        http://(3a7300f0e64d or 127.0.0.1):8888/?token=86f15cc77730d704ac931052cc66cd5ef0a390064eb4a2e7
```

- exposing port 8888 to local host
- mounting local dir to docker dir
- container executes jupyter notebook mapped to port 8888

* paste

  http://127.0.0.1:8888?token=86f15cc77730d704ac931052cc66cd5ef0a390064eb4a2e7

  in browser



* we interact with the running container by a jupyter notebook through the exposed port 8888

But what is jupyter notebook?

- loose acronym meaning **Ju**lia, **Pyt**hon and **R**
- a web-based frontend to create, run, document, plot and manage source code in various programming languages (Julia, Python, R, C/C++, Fortran,…..)
- it comes free with the anaconda distribution
- Anaconda is an open source distribution for python and R. It includes packages for data management and large scale data processing and scientific computing. It has its own package manager.

Components of jupyter

- ***notebook document ("notebooks"):***
  - your source code including the description of the analysis (markdown language) and the results of your analysis
- ***notebook app:***
  - server-client application that allows editing and running notebooks via web browser. Can run locally or remotely
- ***kernel:***
  - "computational engine" executing the code, a notebook is associated with a kernel
- ***dashboard***
  - jupyters "file manager"

Let's start

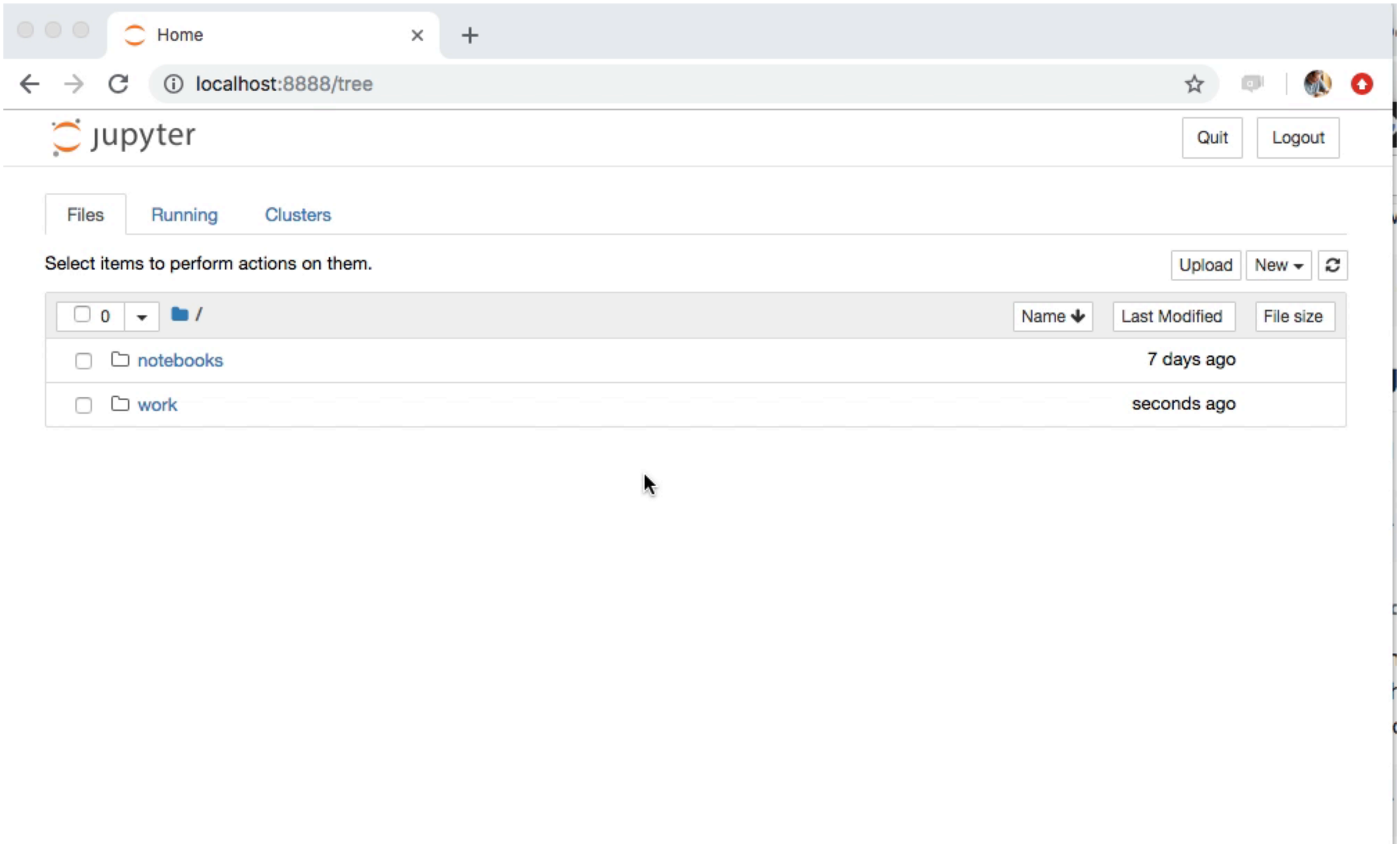
`jupyter notebook`

- without our docker container, after having installed Julia, Anaconda and IJulia, then the above command will start the jupyter server
- our docker-jwas container has all necessary software installed and starts the server automatically which the cmd

```
docker run -it --rm -p 8888:8888 -v /Users/Chris/agn/TUM/JuliaCourse/dataDay1:/home/ubuntu/work qtlrocks/jwas-docker

Copy/paste this URL into your browser when you connect for the first time, to login with a token:
        http://(80b2e434cd65 or 127.0.0.1):8888/?token=e672dec4624214a8a2da817d541c9b5ebef5f27fef0af73b
```

Using jupyter notebook

creating a new notebook from the dashboard

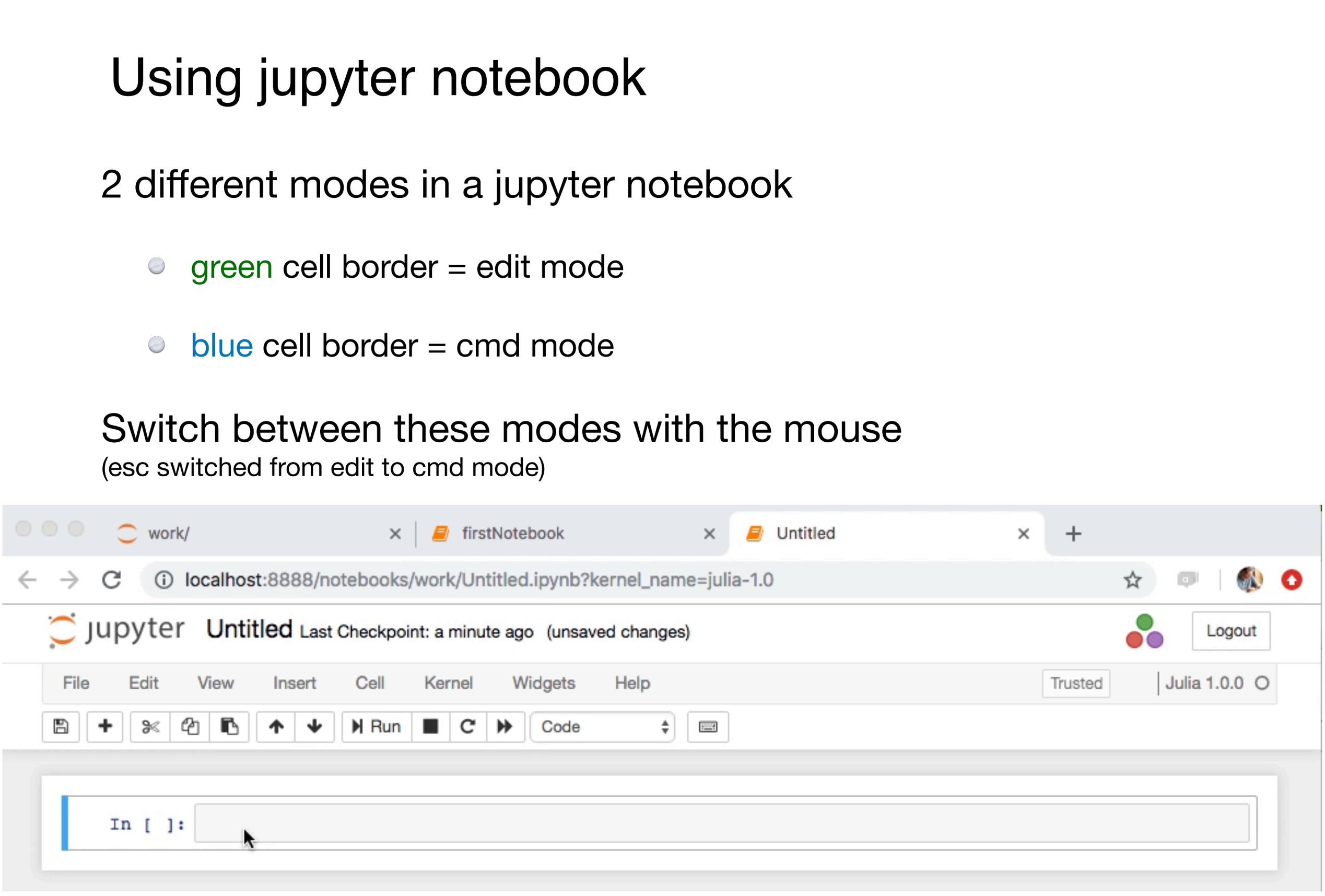


Using jupyter notebook

2 different modes in a jupyter notebook

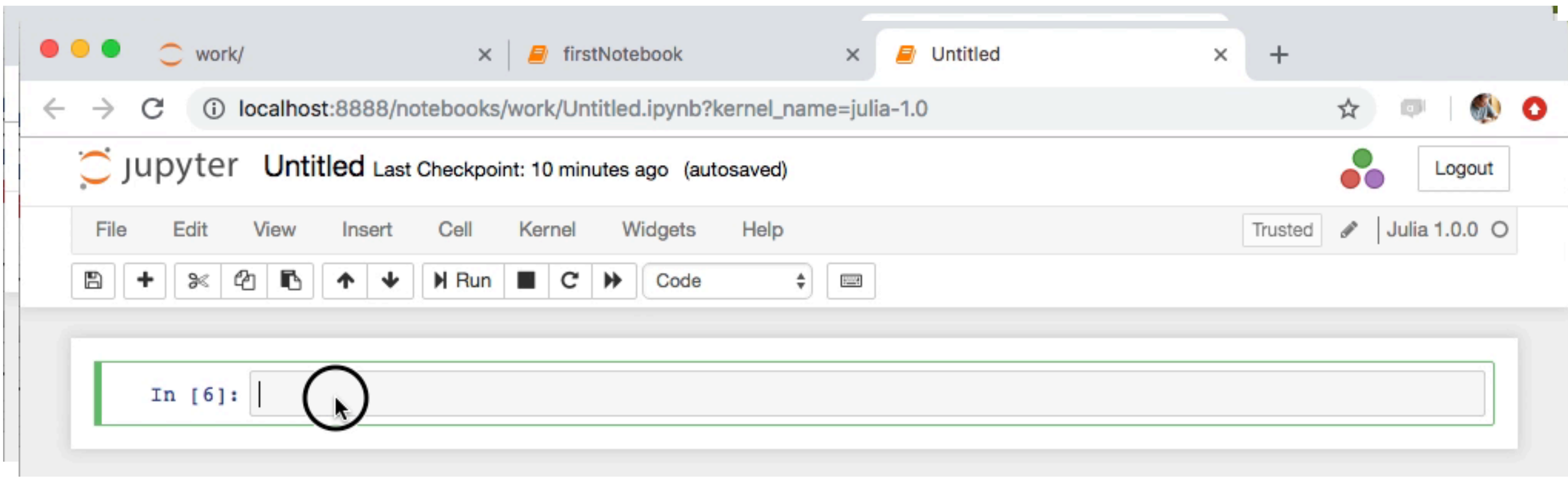
- green cell border = edit mode
- blue cell border = cmd mode

Switch between these modes with the mouse
(esc switched from edit to cmd mode)



Using jupyter notebook

- switch between edit and cmd mode
- execute a cell by dropdown menu, crtl-return or shift-return
- execute several cells via drop-down menu
- by default, output from last line of a cell is printed to stdout (; at the end suppresses that)



Shortcuts in jupyter notebook

(try this in notebook_JupyterCMDs)

- **ESC** to go into cmd mode
- **M** changes cell to Markdown, **Y** back to code
- **X** or **DD** deletes a cell, **a** adds above **b** adds below
- **Shift+Tab** will show short info on method on that line
- **?<cmd>** will show help for <cmd>
- **Crtl+Shift+-** split cell at cursor pos
- **Shift+down/up** to select cells down/upwards, then **Shift+M** to merge
- **ESC+O** to toggle cell output
- https://jupyter-notebook.readthedocs.io/en/stable/notebook.html

- no need to declare type of variable explicitly

Basics in Julia

- # is used for comments, #= ….=# for mulit-line comments
- simple math is straightforward (+-*/^%)
- strings are declared by “string” or “””string”””
  - can be concatenated, indexed like arrays
- Test this with the notebook “notebookJupyterCMDs”

In [ ]:
```
myFirstName = "Chris"
myLastName = "Stricker"
myCountry ="Switzerland"
numberParticipants=28
numberInstructors=3;
```

In [ ]:
```
# $ is used for string interpolation
println("My name is $myFirstName, and I live in $myCountry.")
```

In [ ]:
```
# without string interpolation
println("My name is ",myFirstName," and I live in ", myCountry,".")
```

In [ ]:
```
# Interpolating other types of variables works too
println("There are currently $(numberParticipants+numberInstructors) people in this room.")
```

String concatenation and indexing

In [ ]:
```
string(myFirstName," ",myLastName)
```

In [ ]:
```
myFirstName*" "*myLastName
```

In [ ]:
```
"$myFirstName $myLastName"
```

In [ ]:
```
myFirstName[1]
```

In [ ]:
```
myFirstName[end]
```

In [ ]:

Basics in Julia (https://julialang.org/)

- File I/O *(notebook_IO_DataFrames)*
- adding packages
  - Pkg.add(), Pkg.update()
  - using <pkgName>
- DataFrames *(notebook_IO_DataFrames)*
- strings, arrays, dictionaries, tuples *(notebook_StringsArraysDictPlotting)*
- plotting *(notebook_StringsArraysDictPlotting)*
  - *many plotting packages available (PyPlot, Gadfly, Plots, Winston, StatPlots, Vega, Plotly, Plotlyjs,….), not all of them ported to Julia 1.0 yet*
  - *will show how to use plots and gadfly by example*

References

- https://docs.julialang.org/en/v1/index.html
- https://juliabyexample.helpmanual.io/
- https://syl1.gitbook.io/julia-language-a-concise-tutorial/
- https://jupyter-notebook.readthedocs.io/en/stable/notebook.html