Introduction to Docker container syntax and environment

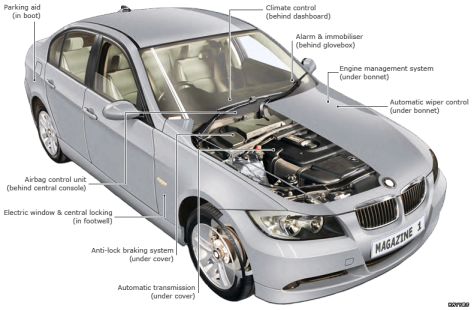
Or… “What the heck is a container?”

What is a computer?

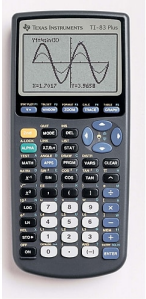
- Dr Wikipedia says:
  - A **computer** is a machine that can be instructed to carry out sequences of arithmetic or logical operations automatically via computer programming

What is a computer?

- Dr Wikipedia says:



All of these are computers:



What is a computer?

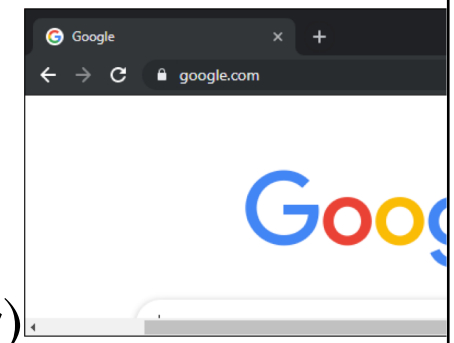
- Dr Wikipedia says:
  - A **computer** is a machine that can be instructed to carry out sequences of arithmetic or logical operations automatically via computer programming

We really only care about these kinds for this tutorial:

What is hardware? What is software?

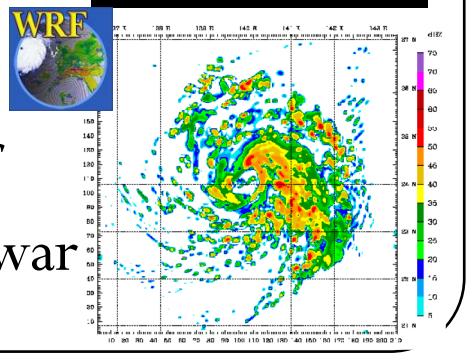
- Hardware is the physical metal, glass, and silicon that makes up your computer

- Software is programs running on the hardware
  - Google Chrome (web browser)



  - Snapchat (application)

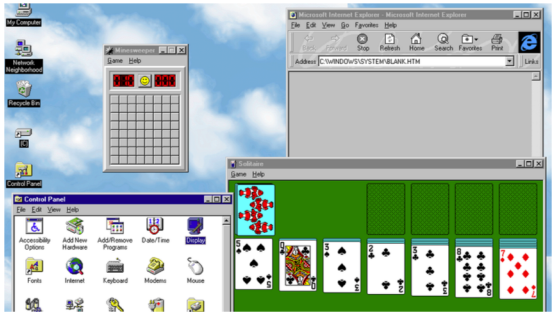
  - A WRF simulation of Typhoon Mawar



What is an operating system?

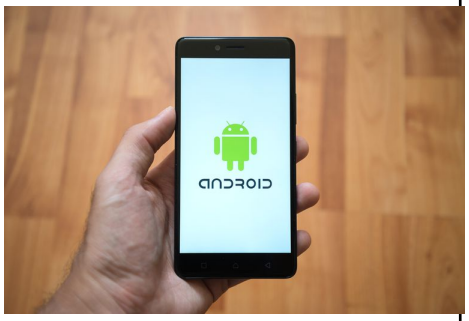
- The operating system is a piece of software that makes it easy for programs and other software to communicate with and make use of hardware
- Examples of operating systems:

MacOS



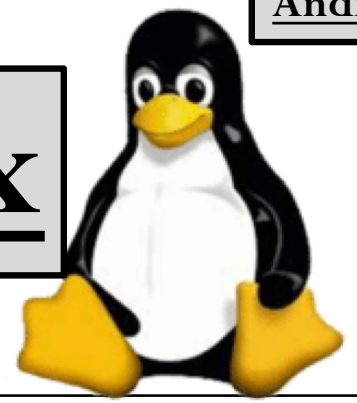
Windows

iOS



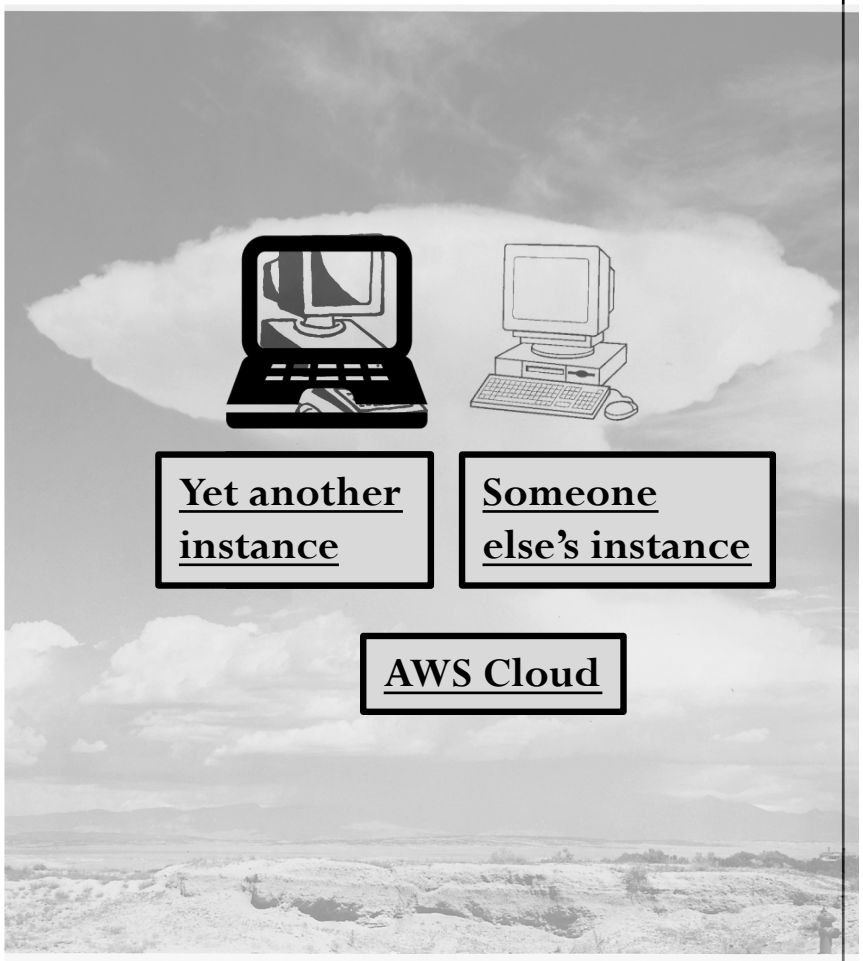
Android

And of course, Linux



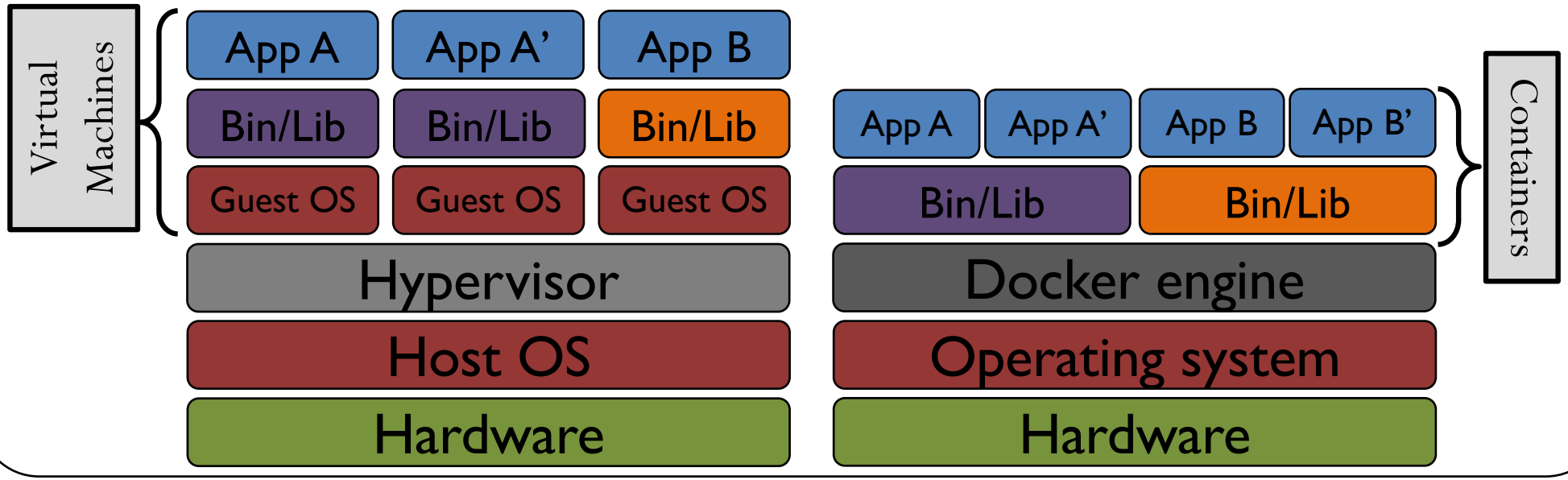
What is a virtual machine?

- Just like any other piece of software, an operating system can run another operating system: this is known as a Virtual Machine
- Incredibly useful for a lot of applications, two major ones are
  - Software developers who have to work on multiple types of hardware and operating systems
  - Cloud computing!
    - Each AWS instance is a virtual machine: it runs much the same as your laptop or desktop, but is just one of many "virtual" machines running on a large rack of hardware, managed by a shell operating system that is invisible to you
    - When your instance is terminated, that same hardware will instantiate someone else's requested instance, probably very different from yours!

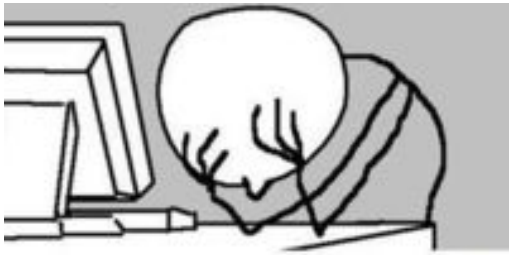


What is a software container?

- A container is a self-contained "box" that allows you to build software once in a custom, portable environment, and then take that "box" and copy it to other machines and run it, so long as you can run the software that runs that "box"
- Similar to a virtual machine, but much more lightweight and portable

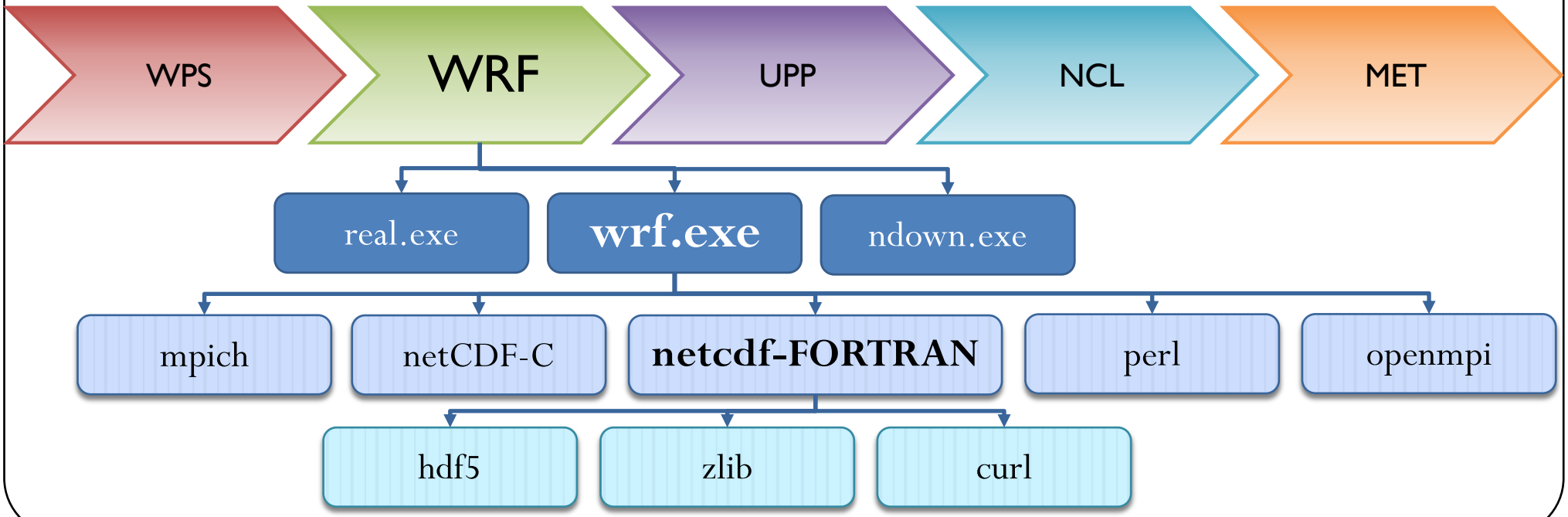


Why use containers for NWP?

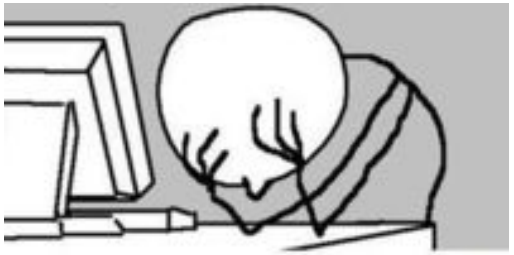


Stick figure trying to compile WRF, c. 2017

- Numerical weather prediction systems are *really* complicated
  - Many different components
  - Most components have multiple programs
  - Each of those programs depend on many *other* programs or software libraries
  - Compiling and setting up any one of these components has a chance to go horribly wrong

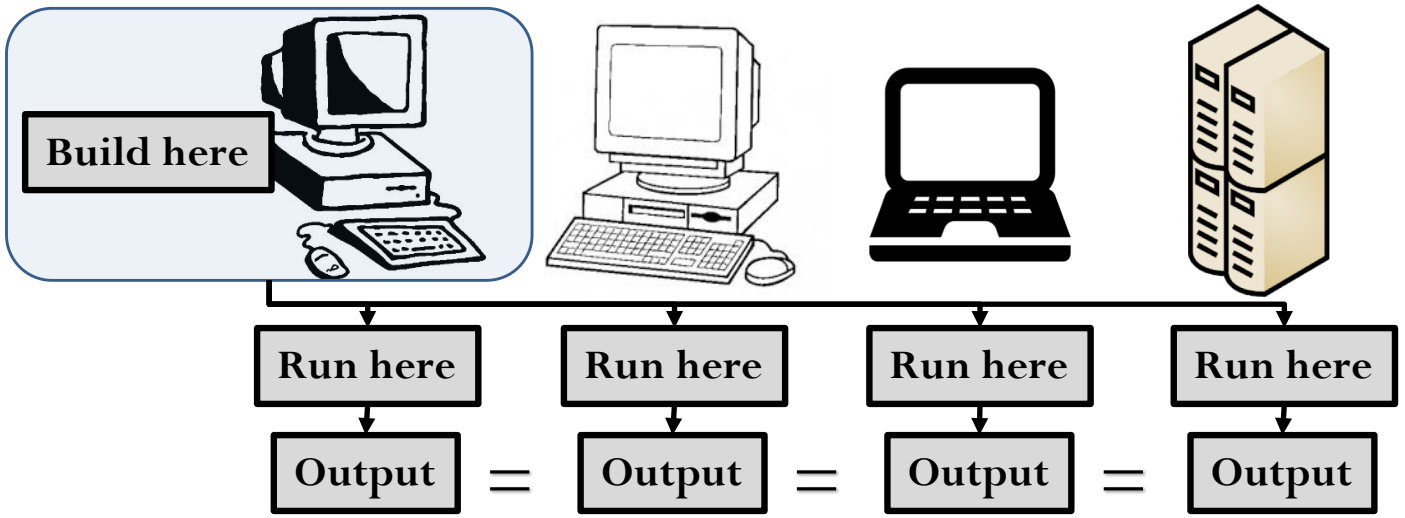


Why use containers for NWP?



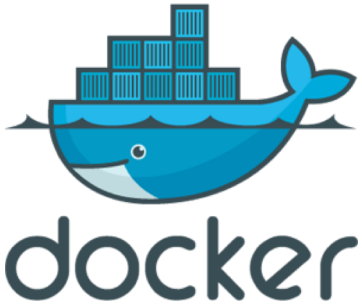
Stick figure trying to compile WRF, c. 2017

- Containers mean someone still has to do all the work to get all those things set up… but only once!
  - Everything required for NWP can be packaged into isolated components, ready for development, shipment, and deployment to many different computing environments
  - Software *should* always run the same, regardless of where it is deployed



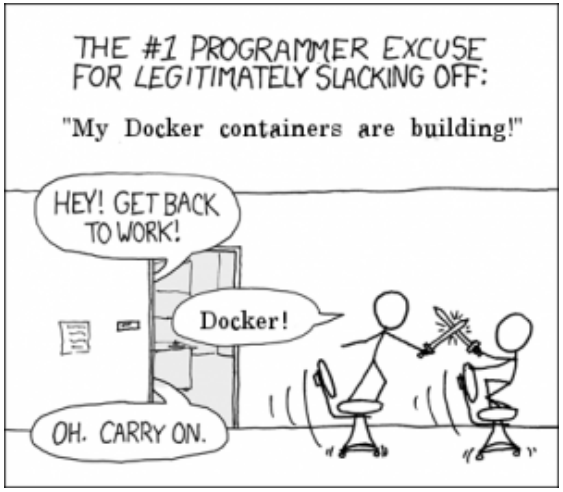
What is Docker?

- Docker is one of the leading software containerization platforms
  - Home page: **https://www.docker.com**
  - Documentation: **https://docs.docker.com**
- Works on Windows, Mac, and Linux machines



Understanding the lingo: Images vs. containers

- Image:
  - Inert, immutable snapshot
  - Created from a recipe file (Dockerfile) with the `docker build` command
  - Can build from scratch *(slower, but offers customization!)* or save to a tar file, which can then be loaded for faster deployment
  - Once an image is built, you can use that image to create a container
- Container:
  - Instance of an image created with the `docker run` command
  - Can be manipulated just like an operating system—data can be created, deleted, and modified—and data can be saved outside of the container with proper settings
  - Can have many running containers of the same image



"The image is the recipe, the container is the cake"
- some rando on the internet

Intro to docker commands

- Getting help:
  - **docker --help** : lists **all** Docker commands
  - **docker run --help** : lists all options for the docker run command specifically
- Building or loading images:
  - **docker build -t my-name .** : builds a Docker image from Dockerfile
    - All of the containers we will use in this class have been pre-built on the AMI for you, mainly for time constraints.
    - The online tutorial contains instructions on building each container from scratch (example: https://dtcenter.org/tutorial-version-3/software-containers/nwp-components)
  - **docker save my-name > my-name.tar.gz** : saves a Docker image to a tarfile
  - **docker load < my-name.tar.gz** : loads a Docker image from a tarfile

Intro to docker commands

- Listing images and containers:
  - **docker images** : lists the images that currently exist, including the image names and ID's
  - **docker ps -a** : lists the containers that currently exist, including the container names and ID's
- All of this and more (including solutions to common docker problems) can be found on the tutorial page: https://dtcenter.org/tutorial-version-3/introduction/docker-commands-tips

Try it yourself

- `docker images`

```
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
dtc-metviewer       latest              393d706ab6c2        5 days ago          1.43GB
mysql               5.7                 db39680b63ac        2 weeks ago         437MB
dtc-gsi             latest              ddaa4a2f0e3f        3 weeks ago         2.62GB
dtc-nwp-gsi_data    latest              5767d897c037        3 weeks ago         1.61GB
dtc-met             latest              2cead505a231        3 weeks ago         4.28GB
dtc-ncl             latest              ca23ab83fd60        3 weeks ago         2.97GB
dtc-upp             latest              67e16bf40248        3 weeks ago         2.56GB
dtc-wps_wrf         latest              a09c530d2d4d        3 weeks ago         2.95GB
dtc-nwp-derecho     latest              d1b991e8b35c        3 weeks ago         845MB
dtc-nwp-snow        latest              d819065690cd        3 weeks ago         1.2GB
dtc-nwp-sandy       latest              e8d3c9285d4e        3 weeks ago         803MB
dtc-nwp-wps_geog    latest              36bf70df233e        3 weeks ago         2.53GB
```

- Take one of the repository names you see above (I arbitrarily choose `dtc-wps_wrf`), and run it this way:

  ```
  docker run --rm -it dtc-wps_wrf /bin/bash
  ```

- You can do all sorts of things: create a file, run commands, delete everything under /comsoftware (seriously, try it!)
- Once you are done, you can exit the container by typing "exit". If you want to re-create the same container from the original image, just type your original "docker run" command again…and the container will be recreated in its original state!

But wait...

- If all that I've done inside my container disappears when I type "exit", what good is it?
  - We instruct you to use the "--rm" flag; this actually removes the container when you exit
  - If you omit this flag, the container remains in the background, and can be restarted
- Even better: we can map directories inside the container to the world outside of the container
  - The "-v" flag is a very powerful option
    - It allows us to mount data containers
    - It allows us to mount a "local" directory (outside of the container) into the system inside the container ("bind mount")
  - Files in a bind-mounted volume that are changed inside the container will be changed outside of the container, and vice-versa
  - This is the way that we can get data into and out of containers

Try it yourself...again

- In your home directory (type `cd` to get there), make a new directory and create a new file in it

```
mkdir /home/ec2-user/magic_portal
touch magic_portal/new_file_outside_container
```

- Run the same container you did earlier, but with an extra option:

```
docker run --rm -it -v /home/ec2-user/magic_portal:/home/other_portal
dtc-wps_wrf /bin/bash
```

- Now in your container, you should see a directory /home/other_portal. If you create a file in here, the same file will appear outside the container

```
bash-4.2$ ls /home/other_portal/
new_file_outside_container
bash-4.2$ touch /home/other_portal/new_file_inside_container

cannot touch '/home/other_portal/new_file_inside_container':
                        Permission denied
```

The possible problems and pitfalls of permissions

- As "magic" as containers seem to be at times, they are constrained by all the same rules imposed on other software by the operating system
  - This includes file and directory attributes such as user (UID) group (GID) and associated file permissions
  - A file or directory mounted into a container will have the same UID and GID as it did outside of the container: these numbers can't be changed
  - On different platforms, this can present a problem if you are not careful: since the user properties such as UID are set when you build the container, moving to a different machine with a different UID can present problems for getting files into and out of containers
- We have taken care of these potential problems for you, but they are worth keeping in mind if you want to use containers for your own purposes

Try it yourself...a third time

- In your home directory, we have already created our example directory and file

```
[ec2-user@ip-172-31-20-202 ~]$ ls magic_portal/
new_file_outside_container
```

- Run the same container you did in the last step, but with yet another extra option:

```
docker run --rm -it -e LOCAL_USER_ID=`id -u $USER` -v /home/ec2-
user/magic_portal:/home/other_portal dtc-wps_wrf /bin/bash
```

- Now in your container, as before, you should see a directory /home/other_portal. If you create a file in here, the same file will appear outside the container

```
bash-4.2$ ls /home/other_portal/
new_file_outside_container
bash-4.2$ touch /home/other_portal/new_file_inside_container
```

- Now exit the container by typing "exit". This time, if you look at the contents of the magic_portal directory, you should see another file in there!

```
bash-4.2$ exit
exit
[ec2-user@ip-172-31-46-86 ~]$ ls magic/
new_file_inside_container new_file_outside_container
```

What is in the DTC containers?

- DTC containers package everything that is needed to build and run the WRF model and produce graphics and verification
  - Repository: https://github.com/NCAR/container-dtc-nwp/
  - Components included: WPS, GSI, WRF, UPP, NCL, MET, and METviewer
  - Components can be run individually or as part of an entire workflow
  - Uses open source software such as GNU compilers; can be run serially or with distributed memory
- README files and online tutorial with explicit instructions for building and running
- Necessary namelist and configuration files
  - Vtable.GFS
  - namelist.wps and namelist.input
  - MET configuration files
- Case-specific data
  - GFS files for ICs/LBCs
  - Observation data for data assimilation and gridded and point verification