

Introduction to containers

Paul Whitford

Northeastern University

Center for Theoretical Biological Physics Tech Talk

October 11, 2023

Disclaimer

- This is *NOT* a comprehensive description of everything one may want to know about containers
- This is an introduction that describes the absolute basics of using and generating containers. The container framework is very powerful, and there are many considerations one may explore when building and distributing containers.

Caution!!

- When trying to run commands, you should always type them manually. Copy/Paste can result in hidden characters being copied, and some commands may fail.

Today's Topics

- Discuss VMs, Containers, Environments and Modules
- Using Docker Containers
- Generating your own Docker container
- Compare Singularity and Docker containers strategies
- Generating a Singularity container
 - From scratch
 - From a Docker image

VMs, Containers, modules and Environments

So many options. What's the point?

Enable reproducible calculations and easily distribute/use software

How can we accomplish this?

- Virtual Machines – Run a complete OS inside of your machine, so that you can have the exact same environment for performing a calculation
- environments (conda, pip, etc) – A locally-stored set of libraries and tools, where the precise configuration can be saved and re-loaded at will
- modules – A way to load a single pre-configured tool (usually in a shared computing environment)
- containers – Like a VM, but smaller. Like an environment, or module, but more portable and secure. The new kid in town.

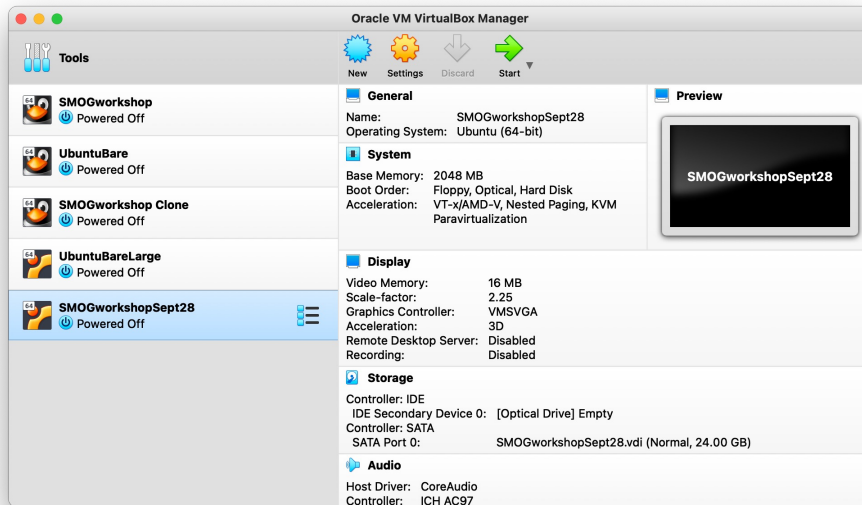
Virtual Machines

- You load an image of an entirely separate operating system that you run “on top” of your OS. It is common that Windows users will run a Linux VM. This way, all software is operating in Linux, and the file systems of the VM and the machine OS are separated
- Pros
 - Portable
 - The VM looks exactly the same, regardless of where you run it
 - Expanded utility
 - Can use Linux tools in a Linux VM, even if your physical machine run Windows
- Cons
 - VM images are typically very large (5-10 Gb), since they contain the entire OS of the virtualized machine
 - Interfacing between the VM and the host OS can be clumsy. Shared file access can be tricky.
 - VM clients are not always free

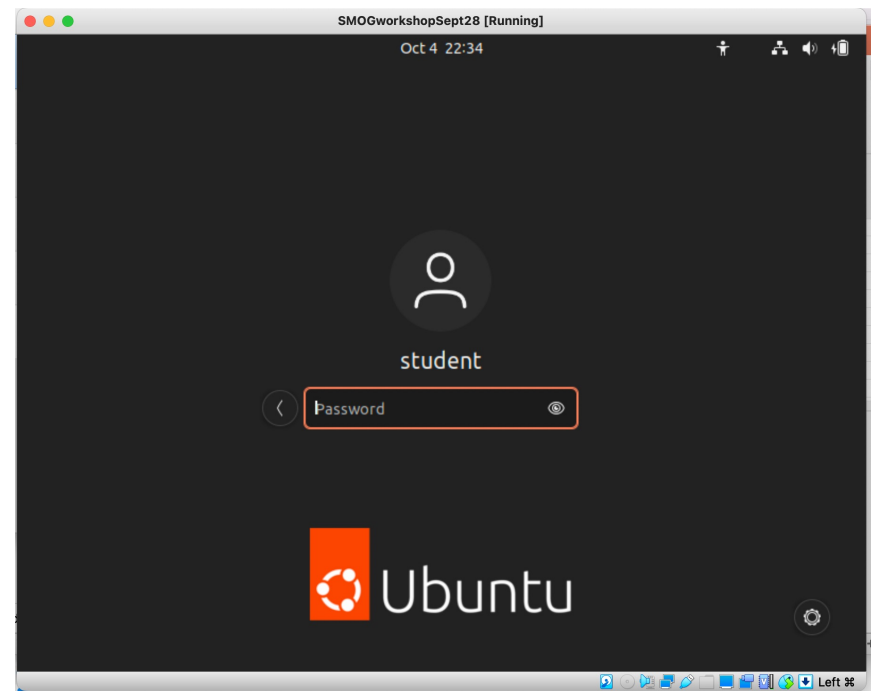
What the VM experience looks like

Launch a VM Client, such as VirtualBox

Can be running on any OS



When VM starts, looks like a new machine inside of a single window



environments

- An environment is a collection of software packages and dependencies that are accessible when performing a calculation
- Some types of environments
 - bash - When you open a terminal, you often have a bash environment, which is initialized by the instructions listed in the .bashrc file.
 - conda – you can create and customize an environment that you can activate/deactivate. e.g. environment1 has compiler v1.0 and software v2.3. environment2 has compiler v1.2 and software v2.4
 - pip virtual environments – package manager is used to install and customize an environment that is specific to a version of python
- Pro
 - You can easy (usually) customize and control the exact tools that are available in the environment
 - Many tools are available through conda and pip, making installation straightforward (usually)
- Limitations
 - There can be dependencies, such as OS-related tools, that are not defined by the environment. If dependencies are updated, there is potential for the environment to not function properly.
 - It is not common to package and send entire environments to different users. You typically don't transfer "the environment" to someone else and have them use it on their machine.

modules

- The “module” tool allows one to dynamically add/remove specific packages from your current working environment.
- This is very common on shared computing resources.
- Pro
 - Easy to use
 - A sysadmin will typically create and manage them. If created correctly, the calculations will work properly.
- Cons
 - All the same general issues one can have with any environment

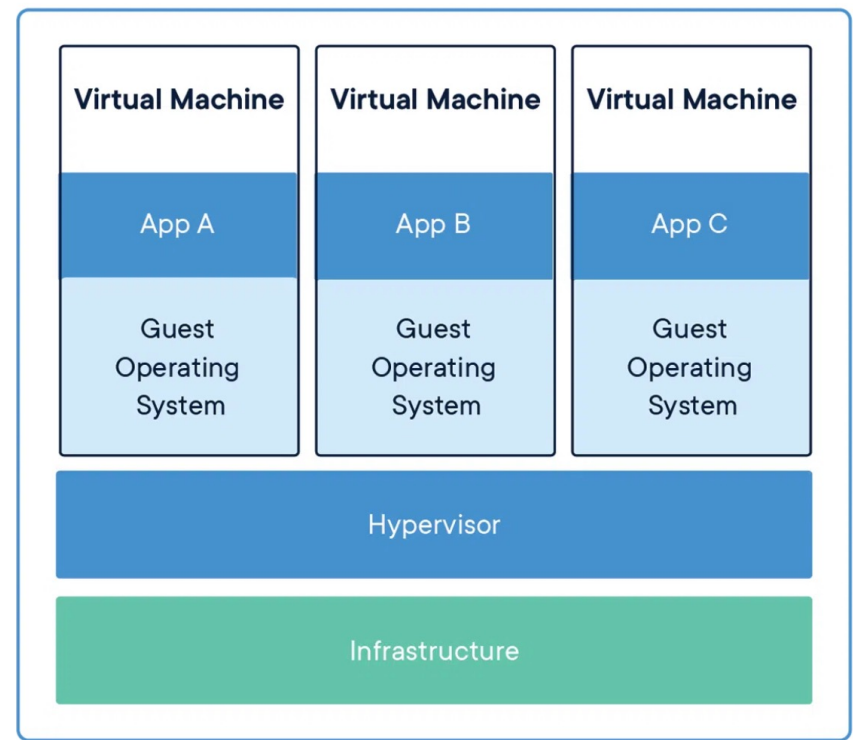
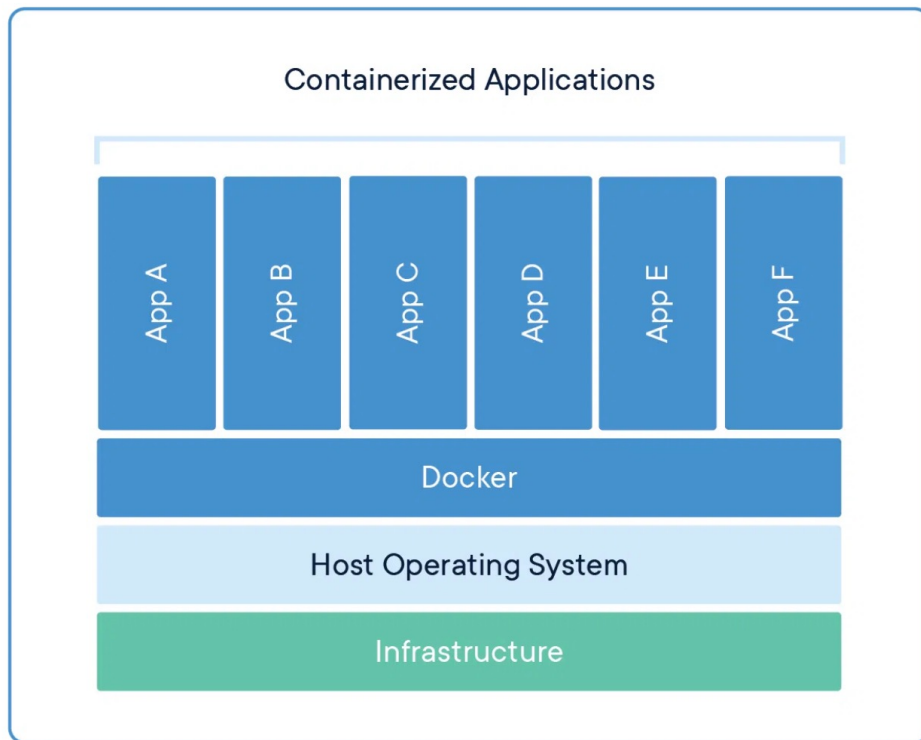
What is a container?

- “A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.” – docker.com

Why use containers?

- Portable
 - May use on Linux, Windows, Mac, VMs, personal or shared resources
 - You can share your containers with anyone
- Reproducible
 - Software behaves identically* on different machines.
 - *as possible
- Application isolation
 - Virtualize CPU, memory, storage and network resources at the OS level. In other words, your application will “see” the same OS and dependencies, regardless of what machine you are running on.
- Lightweight
 - While a VM contains an entire OS in the image, a container only contains “enough” of the OS. The container software ensure that the container can use the host machine, to some extent, thereby reducing the weight of the container

Containers vs. VMs



Notice that each VM has a complete OS

Using a Docker container

- You need Docker installed on your machine.
- When configured you can call docker from the command line
- Some CTBP-generated images can be found at:
<https://hub.docker.com/u/smogserver>
- Let's pull an image used in CTBP
 - >\$ docker pull smogserver/smog2:stable
- If the pull was successful, you can verify it is on your machine with the following command:
 - >\$ docker images

```
whitford@Pauls-MacBook-Pro-3 ~ % docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
moby/buildkit	buildx-stable-1	6b5b3e76f5a6	6 weeks ago	168MB
smogserver/smog2	latest	0f501509d34f	4 months ago	997MB
→ smogserver/smog2	stable	0f501509d34f	4 months ago	997MB

Running a Docker container

- Several approaches
 - Work completely inside the container, with no access to external files
 - Work in the container, but access files that are external to the container
 - Launch the container non-interactively and run a single command

Work completely inside the container, with no access to external files

- `>$ docker run -it --rm smogserver/smog2:stable`
- Explainer
 - `run` : launch the container, based on the image listed
 - `<docker user>/<image name>:<version>`
 - `-it` : run interactively
 - `--rm` removes the container (not the image) upon exit
- If successful, you will see:

```
whitford@Pauls-MacBook-Pro-3 ~ % docker run -it --rm smogserver/smog2:stable
```

```
        Welcome to the SMOG2 Docker Container
You may find more information about the container with the command: smoginfo
Please send any questions to the smog-server team at info@smog-server.org
        This container was built on: Tue May 23 05:41:57 EEST 2023
```

```
smoguser@c10184456522:/workdir$ █
```

From inside the container

```
whitford@Pauls-MacBook-Pro-3 ~ % docker run -it --rm smogserver/smog2:stable
```

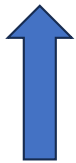
```
Welcome to the SMOG2 Docker Container
```

```
You may find more information about the container with the command: smoginfo
```

```
Please send any questions to the smog-server team at info@smog-server.org
```

```
This container was built on: Tue May 23 05:41:57 EEST 2023
```

```
smoguser@c10184456522:/workdir$ █
```



The username
when inside



The "machine"
name in the
container



Default working directory in
the container

Run some commands

```
smoguser@c10184456522:/workdir$ pwd
/workdir
smoguser@c10184456522:/workdir$ ls
smoguser@c10184456522:/workdir$ smog2
*****
***** SMOG v2.4.5 *****
```

Thank you for using the Structure-based Model (SMOG) software

This package is the product of contributions from a number of people, including:
Jeffrey Noel, Mariana Levi, Antonio Oliveira, Vinícius Contessoto,
Esteban Doderó-Rojas, Mohit Raghunathan, Joyce Yang, Prasad Bandarkar,
Udayan Mohanty, Ailun Wang, Heiko Lammert, Ryan Hayes,
Jose Onuchic & Paul Whitford

In this example, SMOG 2 is available, but we don't have any files in /workdir
At this point, we would need to be able to access some files to do anything
Sometimes, one does not need external files, and working in the container is sufficient.
We will want to access external files, to let's get out of the container
To exit, type "exit" or use Ctrl+D

Work in the container, but access files that are external to the container

- Before running Docker, download the file 2ci2.pdb from the workshop Github repo
- Make sure 2ci2.pdb is in your current directory
- Launch the container
 - `$> docker run -it --rm -v $(pwd):/workdir smogserver/smog2:stable`
 - Explainer:
 - `$(pwd):/workdir` – mount your current directory (pwd) to /workdir
 - This will allow you to read and edit files from within the container
 - /workdir will allow access to all files and directories within \$pwd
- When Docker launches, it will look the same as before. Now try “ls” and see what is available. You should see 2ci2.pdb
 - Caution: you are seeing files on your machine. That means you could overwrite/delete/corrupt them while in the container.

Access external files to run something

- Run some stuff
 - `smog_adjustPDB -i 2ci2.pdb -removewater`
 - `smog2 -i adjusted.pdb -AA`
- After they run, let's exit the container and see the generated files

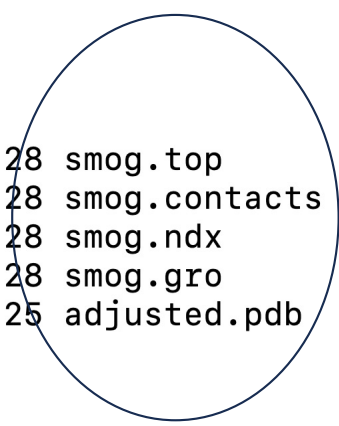
```
smoguser@fd502569d525:/workdir$
```

```
exit
```

```
whitford@Pauls-MacBook-Pro-3 ~ % ls -lt|more
```

```
total 6515168
```

-rw-r--r--	1	whitford	staff	194630	Oct 10 10:28	smog.top
-rw-r--r--	1	whitford	staff	6968	Oct 10 10:28	smog.contacts
-rw-r--r--	1	whitford	staff	1983	Oct 10 10:28	smog.ndx
-rw-r--r--	1	whitford	staff	23542	Oct 10 10:28	smog.gro
-rw-r--r--	1	whitford	staff	70215	Oct 10 10:25	adjusted.pdb



Execute a single command non-interactively

- You don't need to run the container interactively. You can pass the command to be run inside of the container, and then exit the container afterwards.
- We will run the same command as before, but call our files different names:

```
$>docker run --rm -v $(pwd):/workdir smogserver/smog2:stable \
smog2 -AA -i adjusted.pdb -dname singecall
```

- Everything on the continuation line is a single command that is run inside.

Let's create our own container

- To build a container, you need to prepare a Dockerfile
 - Dockerfile – text document that contains all commands one would give to assemble an image
- You can then build the image locally with a single command:
 - `$> docker build -t smogserver/smog2:example -f Dockerfile.SMOG2 .`
- After it is done building, you can use it locally
 - `$> docker run -it smogserver/smog2:example`
- If you want to share it with the world, you can push it to your DockerHub account
 - `$> docker push smogserver/smog2:example`

What was going on during the build?

- Open the file Dockerfile.SMOG2
 - FROM : Pull a base Docker image (ubuntu:22.04)
 - ENV : Set the environment
 - RUN : Give instructions needed to build your tools
 - USER : the default used in the container

In this example, we pulled SMOG2 from a git version, and then checked out v2.4.5. We then deleted .git so we don't have to save the full history

More examples available in the SMOG 2 Github repository

```
FROM --platform=$BUILDPLATFORM ubuntu:22.04
CMD ["bash"]
ENV PATH=/opt/smog2/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
ENV TZ=Europe/Kiev
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone #
    buildkit
RUN apt-get update && \
    apt-get upgrade -y && \
    apt-get install -y --no-install-recommends cpanminus vim git-all patch m
ake cmake pdl default-jre libxml-simple-perl nano emacs && \
    cpanm String::Util Exporter XML::Validator::Schema && \
    rm -rf /root/.cpanm && \
    rm -rf /var/lib/apt/lists/* && \
    mkdir /opt/smog2 && \
    git clone -n https://github.com/smog-server/SMOG2.git /opt/smog2 && \
    cd /opt/smog2 && \
    git checkout tags/v2.4.5 && \
    rm -rf /opt/smog2/.git && \
    mkdir /workdir && \
    useradd -rm -d /home/smoguser -s /bin/bash -g root -G sudo -u 1001 smogu
ser -p "$(openssl passwd -1 smoguser)" && \
    chown -R smoguser /opt/*
USER smoguser
WORKDIR /workdir
RUN cd /opt/smog2 && perl4smog=/usr/bin/perl smog2dir=/opt/smog2 bash /opt/smog2
/configure.smog2
RUN echo "alias smoginfo=\"more /opt/smog2/docker/README.docker\"" >> ~/.bashrc
RUN echo "echo \"\n                               Welcome to the SMOG2 Docker Container\"" >
> ~/.bashrc
RUN echo "echo \"You may find more information about the container with the comm
and: smoginfo\"" >> ~/.bashrc
```

Containers with Singularity/Apptainer

- Used to be called singularity, but for legal reasons changed to Apptainer - it's the same thing
- While Docker is typically used on a personal machine (or cloud resources), Singularity/Apptainer is more common on HPC resources
- Why singularity vs. Docker on clusters?
 - Docker can elevate privileges of a given user, which can effectively allow users to access each other's data.
 - Recall the “smoguser” user in our Docker example? Singularity doesn't have a container-specific user. When using S/A, you are you, whether you are in the container, or not.

Using a S/A container

- Pull the container to your machine
 - `$> singularity pull --arch amd64 library://smog-server/library/smog2:stable`
 - Unlike Docker images, the S/A image will appear as a file (smog2_stable.sif)
- Run the container
 - Interactively: `$> singularity shell smog2_stable.sif`
 - Single-command (two ways):
 - `singularity exec smog2_stable.sif smog2 -AA -i 2ci2.pdb`
 - `./smog2_stable.sif smog2 -AA -i 2ci2.pdb`
 - Run a script
 - `singularity exec smog2_stable.sif ./myscript.sh`

Building a singularity image

- Specify a definition file and build an image from scratch
 - Build on your own machine can be a mess (due to root issues). So, you will probably want to use a build service, such as sylabs
- Pull and convert a Docker image to a S/A image
 - This can be done locally.

Building from scratch (the hard way)

Example def file

Bootstrap : where to retrieve the base image

From : the specific image to retrieve

Here, osperl is an image in the sylabs
library for user smog-server

Labels : a name

Environment : obvious

Post: build your software

Runscript : something that gets called when
you launch the container

```
Bootstrap: library
From: smog-server/library/osperl

%labels
    SMOG v2.4.5

%environment
    PATH=/opt/smog2/bin:$PATH
    export LC_ALL=C
    export perl4smog=/usr/bin/perl
    export smog2dir=/opt/smog2

%post
    git clone -n https://github.com/smog-server/SMOG2.git /opt/smog2
    cd /opt/smog2
    git checkout tags/v2.4.5
    rm -rf .git
    cd /opt/smog2
    sed -i '5,6d' configure.smog2
    perl4smog=/usr/bin/perl smog2dir=/opt/smog2 bash /opt/smog2/configure.smog2

%runscript
    bash

%help
```

Building from Docker (easy)

- If you have singularity installed, you can retrieve a Docker image and convert it to S/A image in one command.
- e.g. `$> singularity pull smogopensmog.2023.sif \`
`docker://smogserver/smogopensmog:hip-git`

Good luck!