Introduction to Docker

Hersh Gupta
Data Scientist, Department of Human Services, DC
Fellow, The Lab @ DC

Agenda

Docker Overview

What is Docker and how does it work?

Using Docker

How can I integrate Docker into my workflow?

Extending Docker

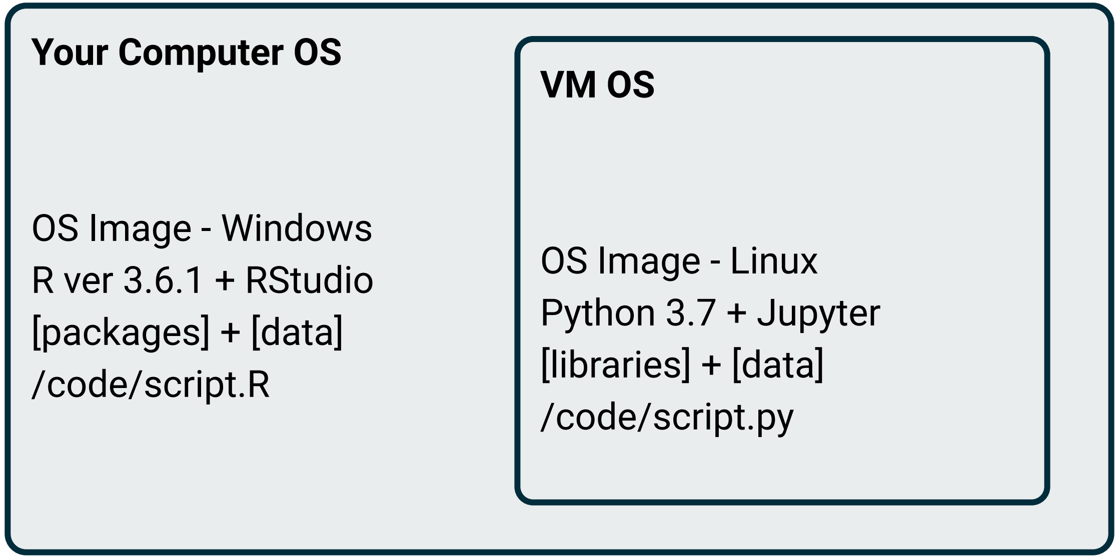
How do I use additional Docker utilities?

What is Docker?

Challenge: You want your code output to be entirely **reproducible**.

However, others running your code are doing so with **different operating systems, different versions of software, different utilities/packages**.

You can't hand your computer over to someone and run the code for them...**or can you?**

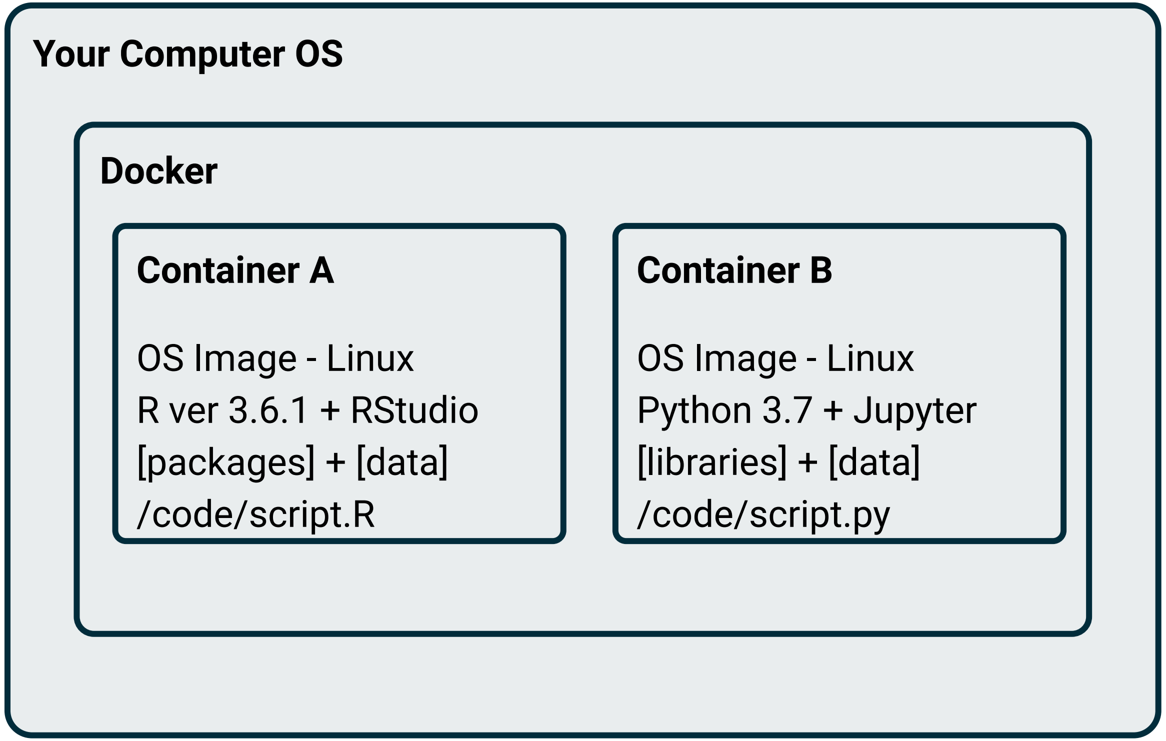


Option 1: Virtual Machines

Each VM runs in its own OS

Allocates own required memory

Can take up a lot of system resources



Option 2: Docker

All containers share the host OS

Requires less memory space

Lightweight/Startup time in milliseconds

Spin up multiple containers quickly

docker

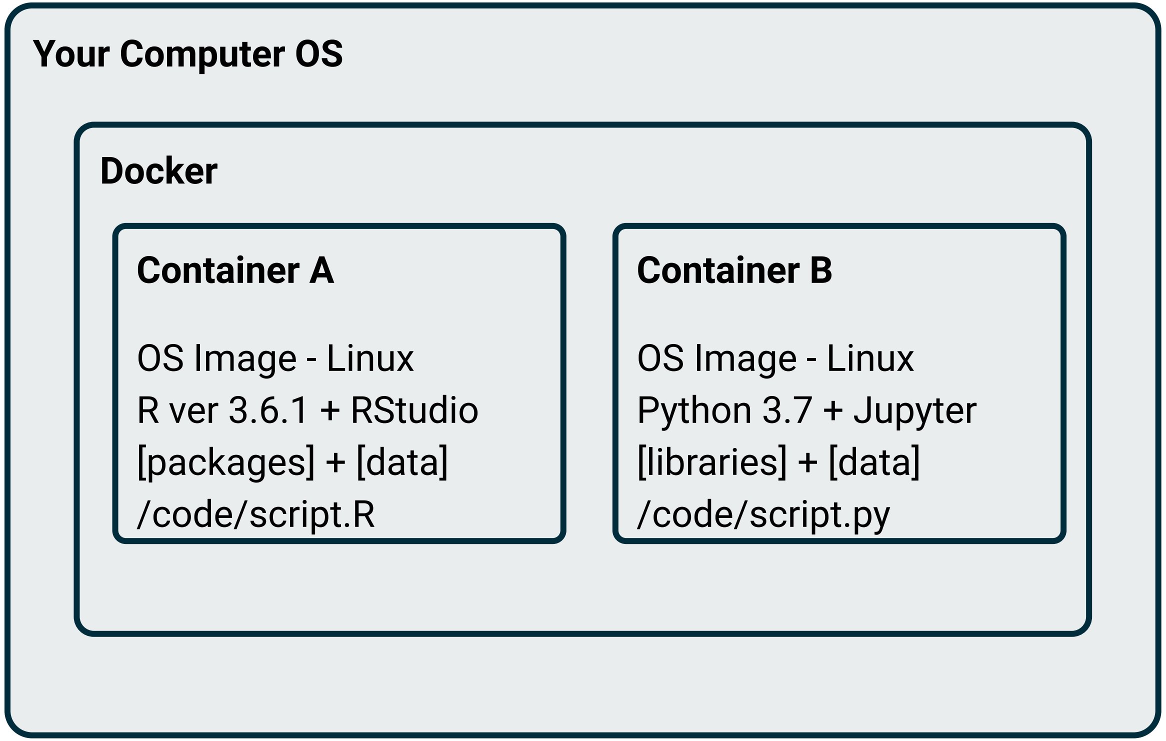
Docker is a program that allows one to:

manipulate (launch and stop)

multiple operating systems (in *containers*)

on your machine (i.e. the *host*).

Docker for Reproducible Research



You can bundle your code with a *Dockerfile*

Dockerfiles are instructions for setting up each container

They specify the OS image, the software, and can specify all packages/libraries to include

Using the Dockerfile, anyone can run your code in a *fixed working environment*

Demo with R

Git Repository

```
repo-name
├── Code/
│    └── 01 Code.R
├── Data/
│    └── data.csv
├── Dockerfile
├── README
└── LICENSE
```

Anatomy of a (simple) Dockerfile

`FROM rocker/r-ver:3.6.0`	Pulls an existing OS image from DockerHub, which sets up Debian Linux, R Version 3.6.0
`MAINTAINER your.email@dc.gov`	Specify who's responsible for this Dockerfile
`RUN Rscript -e "install.packages('tidyverse')"`	Include any additional setup commands sent to container
`COPY /data/data.csv`	Add data from repository to container
`RUN Rscript /code/01_Analysis.R`	Run your analysis in the container

Workflow

1. **Install Docker, if not already installed**
2. **Write your code, make your git commits**
3. **Create Dockerfile, and build image (`docker build -t name`)**
4. **Make docker commits**

Workflow (Alternative)

1. Install Docker, if not already installed
2. Create container, write your code in container
3. Commit your container image
4. Run your image (locally/remotely), which runs your analysis

Demo

- Make sure git and Docker is installed
- Clone github repo hguptadc/DockerTest
- Modify code in Code/ directory
- Examine Dockerfile
- Build local image
- Run analysis in linked container

Download + Install Docker

A. Search for “download docker”, and download the installer (https://www.docker.com/products/docker-desktop)

B. Run the installer with the default configurations

C. Share your drive with docker (click whale icon, shared drive)

D. Create a DockerHub account to upload your images to a central repository

Clone github repo hguptadc/DockerTest

A. Make sure you have git installed and a github account

B. Navigate to a directory where you want to create the project folder

C. Run `git clone git@github.com:hguptadc/DockerTest.git` (if using SSH) from the command line

D. Check the directory to see if all the files are there

Run the Code Locally/Make Modifications

A. Open /Code/01 Code.R file and examine code

B. Run it locally and see if it gives you the results you want

C. Make any changes

D. Save your code

Examine Dockerfile

A. Open Dockerfile

B. Examine the layers in the Dockerfile

C. Understand what it’s trying to do

Build Docker Image

A. Build the Docker image using the instructions in the Dockerfile

B. Enter the following code in the command line/terminal of the project directory: `docker build -t [imagename] .`

C. `-t` is a flag for the name, `imagename` is a name you assign, and the path, `.`, is the location of the directory of the Dockerfile

Run analysis in linked container

A. Run the image you built: enter the following code in the command line/terminal of the project directory: `docker run -it --rm -v ~/Documents/DockerTest/Output:/DockerTest/Output imagename`

B. `-it`: connect container to terminal, `--rm`: remove the container once it's done running, `-v` link a local directory to the container directory

Push image to DockerHub

A. **Name the image you built: use** `docker tag [imagename] [username]/[public imagename]:[version]`

   `imagename` **is the name you assigned on build**
   `username` **is your DockerHub account**
   `public imagename` **is the name you want people to see**
   `version` **is the version number, i.e. 1.0**

B. **Push your image to DockerHub so anyone can use it:** `docker push [username]/[public imagename]:[version]`

Docker Utilities

DockerHub

Directory of already built images

You can pull the image using a simple docker `user/imagename` command

You can build on top of those images

Build Your Code Remotely

You can offload your code execution on a remote server using Docker

Do this for:

- **Unit tests, integration tests**
- **Analysis that would take a long time to run**
- **Continuous integration, i.e. push to a repository, automate testing, and build the report/site/app/etc**

Deploy Your Models into Production

Deploy your machine learning models into production

How to:

- **Build your model and model API locally, add docker file**
- **Set up container on remote server, connect to data source(s)**
- **Run model on container, linking output to additional containers/machines**

Deploy Your Applications into Production

Deploy your applications into production, aka “Microservice architecture”

How to:

- **Build your application locally, add docker file**
- **Set up container on remote server, connect to data source(s)**
- **Run application on container, linking to additional containers/machines**

Useful Docker Commands

- **View containers:** `docker container ls -a` **or** `docker ps -a`
- **View images:** `docker images -a`
- **Remove image:** `docker image rm [image id]`
- **Stop container:** `docker container stop [containername]`
- **Remove container:** `docker container rm [containername]`

Docker Workflow Summarized

Write/edit dockerfile

Build image: `docker build -t [imagename]:[imageversion]`

Run image in container: `docker container run [imagename]`

Tag image: `docker tag [imagename] [username]/[public imagename]:[ver]`

Push image to registry: `docker push [username]/[public imagename]:[ver]`

Pull image from registry: `docker pull [username]/[public imagename]:[ver]`

Real World Examples

- Docker container for a web service deploying a Tensorflow model:

  https://github.com/nolis-llc/DeepMoji-docker

- Docker container for a web service deploying a Keras model:

  https://github.com/nolis-llc/pet-names

- FWE for Civis Analytics:

  https://hub.docker.com/r/civisanalytics/datascience-r/dockerfile

Questions?