Barcelona GSE Data Science Center

Getting Started with Docker for Data Science

September 2018

Reproducible environments

Setting up a server requires lots of programs.

Every version of a program is different: software is alive!

How can you ensure your code will run on a new server? Or someone else's computer?

Cloud Images

Cloud providers usually provide a way to make an “image” of your server.

AWS calls these AMI (Amazon Machine Image).

This can be a great solution if you do all your work on AWS servers.

Barcelona GSE Data Science Center

Docker

Docker is the name of a popular containerization software.

Containers are like little mini computers that run inside your computer.

If you've heard of virtual machines, it's like that, but without the CPU overhead.

Docker

Demo:

(Install Docker via apt on Ubuntu VM)

https://docs.docker.com/install/linux/docker-ce/ubuntu/

Barcelona GSE Data Science Center

Giving Docker to our user

To avoid having to use “sudo” every time we use docker, we can add the current user (ubuntu) to the docker group:

```
sudo usermod -aG docker $USER
```

USER is an environment variable that contains your username, and “docker” is the group we are adding the user to, a group which was created by the docker install process.

Barcelona GSE Data Science Center

Docker Hello World

Let's run our first container:

```
docker run hello-world
```

Images and repositories

A container is created from an “image”. Images are hosted in repositories. All containers created from the same image will be exactly the same.

That’s the point!

“hello-world” is the name of an image. It is hosted in the official docker repository, Docker hub.

Images

When we ran hello-world, Docker downloaded the image from the repository (‘pull’) and then launched the container. This created a container, but it did something else: it ran a command that output text.

Barcelona GSE Data Science Center

Commands

A container consists of an operating system and any applications that the creator wished to install. Often, however, the purpose of the container is to run *particular* application, with everything else being there to support it.

Thus, when the container runs, it runs the command to start that particular application.

That's what the hello-world container does, it's application is printing out "hello", and everything else just exists to make that possible.

Barcelona GSE Data Science Center

Custom commands

Let's try running another container:

`docker run busybox`

What happened? Try: `docker ps -a`

ps

`ps` is a command for running processes. It works in Unix systems and Docker has copied the name. In Docker it lets you see the running docker containers.

`ps` shows you the containers that are currently running.

`ps -a` shows you all containers, including those that are “stopped”.

Barcelona GSE Data Science Center

Stopped containers

Stopped containers aren't deleted automatically. They sit there. This can be useful if you want to restart them or read their logs.

To delete the container, you "remove" it:

```
docker rm [CONTAINER NAME]
```

Try deleting that busybox container we ran (Hint: you will see the container name from the `ps` command).

Barcelona GSE Data Science Center

Named containers

Docker gives your container random names, but it can be a bit annoying trying to delete containers when you don't know what they are called.

```
docker run --name bboxking busybox
```

Now it is easier to remove! But we still haven't gotten the container to do anything.

Custom commands

The busybox container is unusual, it doesn't have a default command that it runs, it expects you to give it a custom command. Custom commands come after the name of the container:

```
docker run --name bboxking busybox echo "foo"
```

Barcelona GSE Data Science Center

Auto-removal

It’s a bit annoying to have to remove our container after using it, when we are doing simple things like echoing “foo.” We can tell docker to remove the container automatically when it stops:

```
docker run --name bboxking --rm busybox echo "foo"
```

Interactive

Also, sometimes we want the container to run an application that we can *interact* with. Sometimes, we want to interact by attaching our terminal to the application running inside the docker container. We can do that by adding two flags: `-i` and `-t`, which we often write together:

```
docker run --name bboxking --rm -it busybox /bin/sh
```

Note: you're now using a shell *inside* the docker container! Try looking around and creating a file. You can leave with `ctrl+d`. Try repeating the command, is the file you created still there?

Barcelona GSE Data Science Center

Volumes

In general, we like starting “from scratch” each time with a new container. That’s what makes it so reliable! There’s no *state* left over from last time to screw things up. It’s *deterministic*: if it works once, it will work twice!

This is what makes Docker containers so great.

But sometimes, you want to keep some state, or output a file:

```
docker run --name bboxking --rm -it -v $PWD:/home
busybox /bin/sh
```

Jupyter Docker Stacks

The folk behind Jupyter have created a set of Docker images that are extremely useful for data science:

https://github.com/jupyter/docker-stacks

We can run the “datascience” image with the following command:

```
sudo docker run -d --name notebook -v $PWD:/home/jovyan/wo
-p 8888:8888 jupyter/datascience-notebook
```

Even More Docker options

-d Run the container “in the background,” so you can do other things in your terminal, or close it, without the container exiting.

-p Just as computers have numbered ports, so do containers. This option creates a mapping from the host computers port to the containers port. For example, “707:8888” would map the port 707 on the host computer to the port 8888 in the container.

Barcelona GSE Data Science Center

Getting logs

Because we ran the container “in the background” via the -d command, we have to look a bit to find the token that jupyter creates for us:

```
docker logs notebook
```

This will display all the logs that the container named “notebook” has created.

Look at the website for the Jupyter Docker Stacks to see more about all the options they provide! Including setting your own password, and using Jupyter Lab.