Containers Are the New Hotness

Maybe you've heard of it, maybe not. What is Docker? Docker is a "container" virtualization technology. Containers are related to the full virtual machines you may already know but are quite a bit different. For instance, you can't run a GUI in a container — only a command line.

Containers are a bit like alternate dimensions inside a computer. The file system, networking, and memory all have special properties that allow you to cleanly package up and execute software applications separately from one another in a lightweight fashion. With containers it becomes quite simple to run Foo v5.1 right next to Foo v4.3 with no worry about library or driver versioning or cross-contaminated cache directories or any other such headaches.

Containers are one tool among many for DevOps where an application's production environment is the same as its development environment. A container is well managed and portable. It also happens that containers are an excellent vehicle for command line based build environments of any sort — even if they're never used to run software in production.

Terminology

- **Docker image:** A singular, static bundle of resources that the virtualization runtime executes. This bundle is built with Docker tools, contains resources you specify as its developer, and is versioned in a repository.
- **Docker container:** An executing (i.e. in memory) stack of virtualized resources.

Running a Container (help for your environment set up)

If a running container neatly segregates an executing application away from everything else, then how are you supposed to get your codebase inside? Good question. Lo and behold, Docker volumes…

With a Docker volume, we're able to map a piece of your local file system into the course container file system. This mapping allows the file system inside the container to magically refer to your local file system outside the container.

Launching a Docker container with volume mapping might look like this:
`docker run -it --rm -v $PWD:/lab throwtheswitch/drsurly-course2`

- `run` loads the image into execution.
- `-it` connects the container input and output streams to your local host command line.

- `--rm` releases the container's resources when it ceases to execute at `exit`.
- `-v` maps a volume.
    - The component before the colon in the `-v` parameter is your fully qualified local file system path.
    - The component after the colon is the path inside the container being mapped to.
    - `$PWD` is a Unix environment variable that contains the full path of the current working directory. In Windows land, it's `%cd%`.
- `throwtheswitch/drsurly-course2` is the image to load, available from a public repository.

Command line notes:

1. Find the full Docker `run` command documentation here.
2. Documentation for the `run` command's volume mapping option is here.
3. Docker requires fully qualified paths in its volume mapping command. Do not attempt to use `.`, the dot convention to represent the current working directory.
4. Some special directories on your system cannot be volume mapped within Docker. You're best off working in the normal user space of your file system.

Docker Notes & Resources

- In addition to its nifty volumes Docker also wraps up virtual networks that allow you to connect containers together so they can communicate. These abilities are not needed for your course development environment and are not discussed here.
- Docker is one name brand and the leading commercial implementation of container-based virtualization. Docker containers are free to create and use. Docker makes its money through ancillary services. This is a messy and emerging space. Other commercial and open source alternatives exist.
- *The Docker Book* is an excellent, in-depth guide to learning Docker. It costs $9.99. The e-book speaks to a DevOps audience developing for and deploying databases and web servers. That is, there's no discussion of creating Docker images for software development build environments. Nevertheless, this book can help you become quite proficient at assembling build environments for embedded software in addition to deploying servers and system software. (We have no connection to sales of this book.)
- As always, Windows is a bit of a special case. Core Docker is Linux-based and executes Linux applications (even *on* Windows). However, Docker for Windows applications also exists. In theory, you could, for example, assemble a Docker container that packages up the IAR command line executable toolchain for Windows. A bit more here.