The Load

Some thought about how Docker is powerful, and getting it on your system.

- Real Quick
- Install

Real Quick

In my opinion (oh snap, here we go, right?),

If every operating system in the world was the same, then there would be little to no hassle with how an application was installed, used and supported. Everything that got installed on your machine, would be the same exact way it was installed on every other machine out there. Well, that is not the case. There are so many flavors of operating systems, that it has actually become very difficult to find a decent answer online when you run into issues. A single application has to be built and released in so many different formats just to appease the masses. That is a lot of work for application developers, and a lot of environments they need to know their way around in order for those releases to actually be a thing. Some developers have just said "screw that", and they only release a single variant. If your not using a system that works with that variant, well, your out of luck. There are some of us that actually determine a applications worth or worthiness, by seeing how many different operating systems their software can run on. A developer that has taken the time to make sure their product can work on just about anything, is a developer I might pay a bit more attention to, or throw my money at.

Then. Some group of devs, come out with this bright idea, to make something that had the potential to solve a world wide non-universal problem... with a universal solution. **Docker**.

They have spent the time to make sure that this product can be installed on all (almost all) operating systems. That speaks to the worth of it. They have also made the product Open Source! That alone speaks volumes!! A world changing construct. Available so anyone can see how it's built, so anyone can help support it, so anyone can modify it how every they wish. Available to the world for free!

To help try and paint a picture of what this product is. Imagine taking all the most common operating systems in the world, that people use for work or pleasure. Then on those operating systems, installing a product that takes some of the space on that machine and creates a small bubble. Inside that bubble are empty book shelves all lined up. The bubble itself is capable of looking at its surrounding area (the operating system) and determining how much processing power it has to work with, how much memory it has to work with, how much disk space it has to work with, and what kind of network it is working with. The bubble takes that information and makes it available to the book shelves inside. This same "infrastructure" (if you will) is the same setup that every one gets when they install Docker onto their operating system. It is exactly the same across the board.
Now a developer can take a single application and format it to fit and work inside this bubble. The application gets installed in the bubble and put on the shelf. Because the application was designed to work inside this bubble, this makes the application universal, and able to function inside any system that has Docker (a bubble) installed on it. Now anyone can pick up a application off the shelf and just start using it. They don't need to worry about a ton of requirements in order to make

that application function. As long as your Docker environment was installed without errors (and the install does not seem very error prone), and the application was built to work within Docker, then anyone can run a Docker built application without a hitch.

Awesome!

Now that scenario is a very basic idea. There are many options that can be thrown at this system to form it to everyone's liking, or to be able to wrap some security around it. But the plane Jane out of the box Docker load, will work as described. The application developers also build in options for the user. So when your installing the application or before installing, you can decide and configure what folders you want it too look at or what folders it has access to (outside the bubble), or what name and/or IP you want it to have, or who can access this new application, etc..

It was a well thought out plan... and it is sweeping the nation! Or rather, it has swept the nation!

Install

I run Rocky Linux 8 and also have some Red Hat 8 builds (feeling out some 9 right now too). So these instructions will use the package manager that comes with those operating systems. You may have to tweak these commands to fit your package manager, if not using the same style systems. However, the simplicity of it should still be the same.

Install The Docker Repository (Docker Documentation)

```
dnf -y config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

Update The System
I usually build a system using the bare minimums, so I am also going to add a couple items before just going for it.

```
# Update the entire system (which will also bring in the new Docker repo)
dnf -y update

# I need to add iptables to my system so Docker can use it to make it's own internal networks
# avaialbe to the operating system (so I can access them).  "lsof" is a handy tool, so I am
# including it.
dnf -y install lsof iptables iptables-services iptables-utils
```

Install Docker
This is the super hard part.

```
# This will install the Docker Community Edition Engine and CLI.
# I am also including the Docker Compose Plugin and ContainerD.io
dnf -y install docker-ce docker-ce-cli docker-compose-plugin containerd.io
```

Enable and Start Docker

```
systemctl enable --now docker
```

That's it.

Docker is installed and ready to use!

Here is all of the above in a nice copy/paste string. *(It's me, not you.)* ;)

```
dnf -y config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo && \
dnf -y update && \
dnf -y install lsof iptables iptables-services iptables-utils && \
dnf -y install docker-ce docker-ce-cli docker-compose-plugin containerd.io && \
systemctl enable --now docker
```