

Docker

The way of containerizing. It's been a thing for a while now, and the simplicity of it is really a life changer. Well, maybe not a life changer, but definitely a mental method twist.

- The Load
 - Real Quick
 - Install
- What Now
 - Docker Hub
 - A Quicky (portainer.io)
 - Media Management Apps

The Load

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

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
# available to the operating system (so I can access them). "Isof" is a handy tool, so I am
# including it.
dnf -y install Isof 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
```

What Now

Soooo... what to do now?

What Now

Docker Hub

Docker Hub is a repository for Docker images, that you can use to deploy into your Docker environment. There are a bunch!

There are other repositories out there, but a lot of them feed back into the Docker Hub in one way or another. You could also start developing or turning apps/code/etc into containers and pushing them up to the hub. Me just saying that should give you an idea of how massive the Docker Hub is.

Explore The Hub

Docker Hub

There are a lot of images, and a lot of them has different ways to get them into your system and functional. I am not going to get into all of those ways, but I am going to show how a few simple apps (with big functionality) are installed.

Explore and search through the hub. There are tons of images that will really get your brain juices flowing.

What Now

A Quicky (portainer.io)

Here is a quick example of a image container install. I am actually going to write up a complete install that would be used to manage media (movies, tv shows, etc). I know, it has been done before, I get it. But I would like to write it up in such a way, that I don't have to fight through everyone else's advertisements, and pop-ups, and stupid garbage just to get the little piece of code I was looking for.

This example will be on how to install Portainer. Portainer is a cool web driven tool, that allows you to manage your containers. Why would I not start with this?

portainer.io

Portainer needs a spot to install its database. It's called a volume location.

```
docker volume create portainer_data
```

Now we install the Portainer container.

```
# I like pulling the images down first. It's kind of a way for me to see
# if it's even available before I actually use it to install with.
docker pull portainer/portainer-ce:latest

# Deploy the container
docker run -d \
  --name portainer \
  --restart=always \
  -p 8000:8000 -p 9443:9443 \
  -v /var/run/docker.sock:/var/run/docker.sock \
  -v portainer_data:/data \
  portainer/portainer-ce:latest
```

Check and see if it's running.

```
docker ps
```

Should be good to go.

You can now access the new Portainer web application. Swap out "localhost" for the IP (or name) of your Docker server.

Easy yea? Now, go play with that for a little bit. ;)

Media Management Apps

OK. Here is a collection of tools that can be used to manage a complete media library. When I say manage, I mean that these tools allow for requesting movies or tv shows, downloading the request, and placing the request in your library in whatever fashion, folder, or naming scheme you want. Having said that you should know, that my explanation is way over simplified. You will have some configurations on your hands, and will need some outside resources (indexers) to search from and maybe a couple sources (news servers and/or torrent sites) to pull from.

That is not part of this how-to.

Here I will describe installing some docker containers that would help facilitate the above idea. There are a bunch of folks that containerize these tools, but I am only going to show one particular offering. <https://hotio.dev/>

He/She/They have compiled a bunch of media management tools, so one stop shopping. The tools are also available through the docker hub.

docker search hotio

You can skip down to the code below if you just want to get rolling. If you want a bit more information, then here is the list of the tools we will load, their function, and a bit of explanation on the directory volume mounts we will create.

NZBGet - This is the core downloading client. Sonarr and Radarr will sent the request to NZBGet. Once the request is downloaded it will place the download in a categorized folder (sonarr or radarr respectively). Sonarr/Radarr will monitor the categorized folder for the download to appear.

Sonarr & Radarr - Sonarr is for TV shows, and Radarr is for movies. These applications will manage your media library, by forwarding your requests on to the downloader, and monitoring for it to complete. Once it appears in the categorized folder location, Sonarr/Radarr will take the download, rename it, create a folder for it, and drop it into your media collection. Sonarr/Radarr will also keep track of the quality of your media. If they detect a better quality has become available, they will automatically send a request to download it and then replace the current copy.

Overseerr - This application ties itself into your library and into Sonarr and Radarr. Overseerr gives a single pain of glass for you to make requests for media. When a request is made it forwards that on to Sonarr/Radarr. Once that request has hit your media library, it will mark it as available. Overseerr will scan your media library and notate your collection as already available, making it easy to recognize already downloaded or available media so that duplicate requests are not made. Overseerr also has the ability to launch the media using your desired media server.

Media Server - Plex, Emby, JellyFin to name a few. These systems make your media library available on your TV in a pretty, remote control, friendly way. They also provide the ability to access your media while remote (away from home). The backend of these systems control the streaming process and adjust transcoding (if required) in order to make your media stream-able while on various bandwidth handicapped networks. They also categorize your media into easily searchable areas.

Here is a sample media server folder structure. Please keep in mind, that this layout is for simplicity of this how-to. Your media could reside on some other storage system, and these dockers could be on different systems as well. You are not locked into a single source system. If using separate systems, you will need to map your media locations to the various tools, so that they have access. For this how-to, I am going to give a single source layout.

Imagine the following directory structure...

```
/opt/nzbget
/opt/radarr
/opt/sonarr
/opt/overseerr
/opt/utility
  /opt/utility/nzbget
  /opt/utility/radarr
  /opt/utility/sonarr
/opt/media
  /opt/media/movies
  /opt/media/tv
```

Our docker images are build with all of their configuration files inside of a /config folder. We will install our images to the /opt directory and map a path to those /config folders, that will be available to us on the outside of the container. This mapping (or mount point as is called in the docker world) would look like this (using nzbget as the example)...

```
-v /opt/nzbget:/config
```

-v indicates a "mount" is being defined. Left of the colon is our path, and right of the colon is inside the container.

Are you tracking?

OK. So here are a couple more. Let me explain these with a reference to what I said previously about the applications. NZBGet is going to get a request from Sonarr or Radarr. When that request is made, those apps tell NZBGet to please pull down this request, and if you don't mind,

please put it in my folder. Sonarr's folder that it will monitor for the request is /opt/utility/sonarr and same goes for Radarr /opt/utility/radarr. (this is all configurable)

So we need to make that location available to NZBGet and we need to make it available to the requesting applications.

NZBGet:

-v /opt/utility:/utility

Sonarr:

-v /opt/utility:/utility

Radarr:

-v /opt/utility:/utility

Here we just open up the /opt/utility directory to all three apps. Then during the configuration of the application you would select your respective directory.

Alright. I hope that makes sense.

Let me go ahead and drop the complete setup and image loads for these applications.

```
# make directory structure
mkdir -p /opt/{nzbget,radarr,sonarr,overseerr}
mkdir -p /opt/utility/{nzbget,radarr,sonarr}
mkdir -p /opt/media/{movies,tv}

# pull down the images
docker pull hotio/nzbget
docker pull hotio/radarr
docker pull hotio/sonarr
docker pull hotio/overseerr

# list out the images docker now holds
docker images

# build nzbget container
docker run -d --restart=unless-stopped \
  --name nzbget \
  -p 6789:6789 \
  -e UMASK=002 -e TZ="America/Denver" \
  -v /opt/nzbget:/config \
```

```
-v /opt/utility:/utility \  
hotio/nzbget:latest
```

```
# build sonarr container
```

```
docker run -d --restart=unless-stopped \  
  --name sonarr \  
  -p 8989:8989 \  
  -e UMASK=002 -e TZ="America/Denver" \  
  -v /opt/sonarr:/config \  
  -v /opt/utility:/utility \  
  -v /opt/media:/media \  
hotio/sonarr:latest
```

```
# build radarr container
```

```
docker run -d --restart=unless-stopped \  
  --name radarr \  
  -p 7878:7878 \  
  -e UMASK=002 -e TZ="America/Denver" \  
  -v /opt/radarr:/config \  
  -v /opt/utility:/utility \  
  -v /opt/media:/media \  
hotio/radarr:latest
```

```
# build overseerr container
```

```
docker run -d --restart=unless-stopped \  
  --name overseerr \  
  -p 5055:5055 \  
  -e UMASK=002 -e TZ="America/Denver" \  
  -v /opt/overseerr:/config \  
  -v /opt/media:/media \  
hotio/overseerr:latest
```

```
# check docker processes and see if our containers are running  
docker ps
```

```
# some non-pro notes
```

```
#
```

```
# docker start <name or container id>
```

```
# docker stop <name or container id>
```

```
# docker rm <name or container id>
```

```
# docker image rm <image name or id>
```

```
#
```

All the applications should be running.

That was easy, yea? :)

It usually takes longer to explain than it does just doing the work.

You should be able to reach each of those applications by referencing the docker server and the port you specified.

<http://localhost:6789> --> nzbget

<http://localhost:7878> --> radarr

<http://localhost:8989> --> sonarr

<http://localhost:5055> --> overseerr

Soooo.... remember I was saying that when creating the mount points, the content to the left of the colon is your side and the content to the right of the colon is inside the container?

Well that goes for ports as well.

So you want to change Overseerr to appear as the main web page for the server?

```
# change the line that references the port
```

```
-p 80:5055
```

Oh you want to upgrade or add another mount point to your NZBGet container?

```
# remove container and image - your settings were saved in the configured "config" location
```

```
# those will NOT be removed with the container
```

```
docker stop nzbget
```

```
docker rm nzbget
```

```
docker image rm hotio/nzbget
```

```
# add new mount location (media) then deploy nzbget container again
```

```
# as the container is built it will look for updates as well
```

```
docker run -d --restart=unless-stopped \
```

```
  --name nzbget \
```

```
  -p 6789:6789 \
```

```
  -e UMASK=002 -e TZ="America/Denver" \
```

```
  -v /opt/nzbget:/config \
```

```
  -v /opt/utility:/utility \
```

```
-v /opt/media:/media \  
hotio/nzbget:latest
```

You would like to install a different version? Perhaps you need to know what version you have right now if

"docker images" only shows "latest".

```
# docker images only shows "latest" under TAGS
```

```
docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hotio/overseerr	latest	e5a53c0a5afb	3 days ago	1.32GB
hotio/radarr	latest	696871c89ef3	5 days ago	247MB
hotio/sonarr	latest	bdd448f1c3ae	5 days ago	374MB
hotio/nzbget	latest	a592e2f8f7b2	5 days ago	104MB

```
# use "docker inspect <image name or id> and grep for "version"
```

```
docker inspect nzbget | grep version
```

```
"org.opencontainers.image.version": "21.1"
```

```
# you want to install the "testing" TAG version "21.2-testing-r2333"
```

```
docker stop nzbget
```

```
docker rm nzbget
```

```
docker image rm hotio/nzbget
```

```
# validate the image has been removed
```

```
docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hotio/overseerr	latest	e5a53c0a5afb	3 days ago	1.32GB
hotio/radarr	latest	696871c89ef3	5 days ago	247MB
hotio/sonarr	latest	bdd448f1c3ae	5 days ago	374MB

```
# on the last line, replace "latest" with the TAG you want and build it
```

```
docker run -d --restart=unless-stopped \  
--name nzbget \  
-p 6789:6789 \  
-e UMASK=002 -e TZ="America/Denver" \  
-v /opt/nzbget:/config \  
hotio/nzbget:testing
```



```
-v /opt/utility:/utility \  
-v /opt/media:/media \  
hotio/nzbget:testing
```

I hope this was helpful!

:)