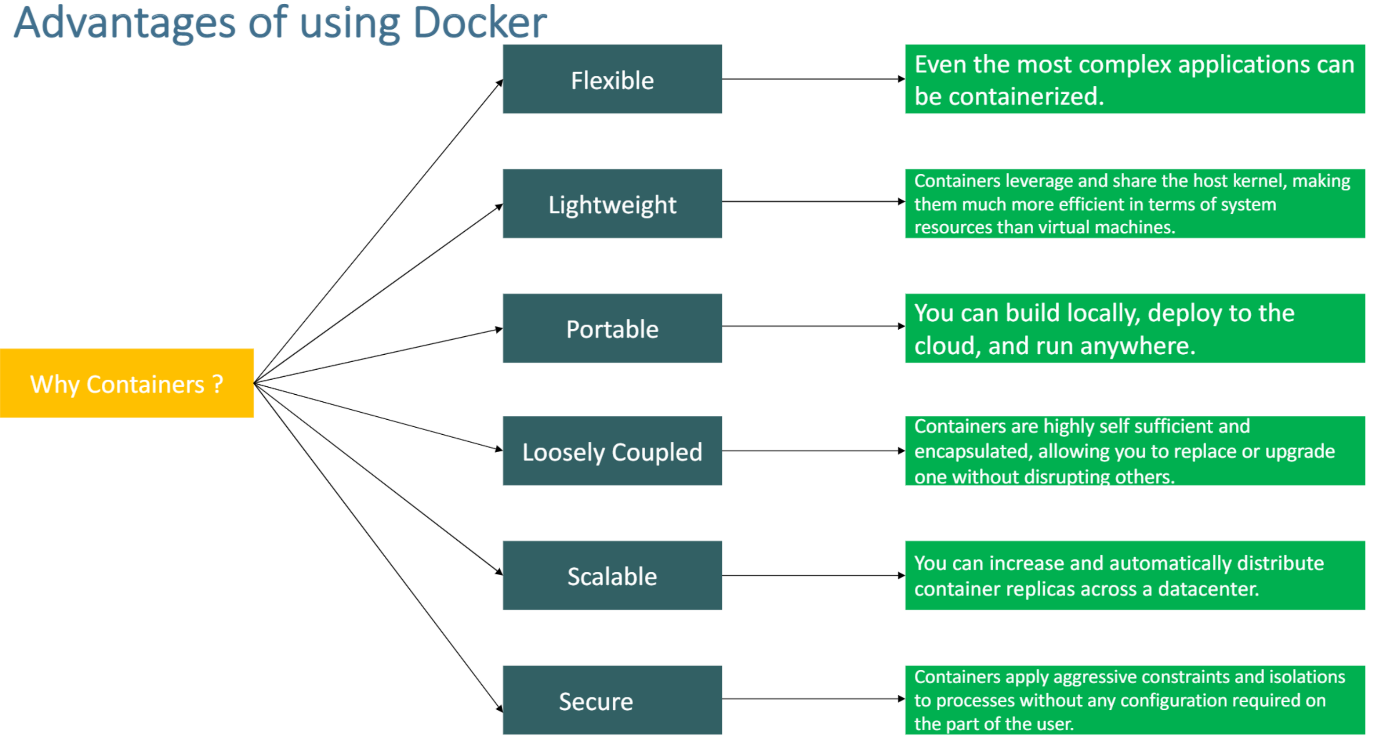
# Docker and Kubernetes

# 

1. **Docker Fundamental**

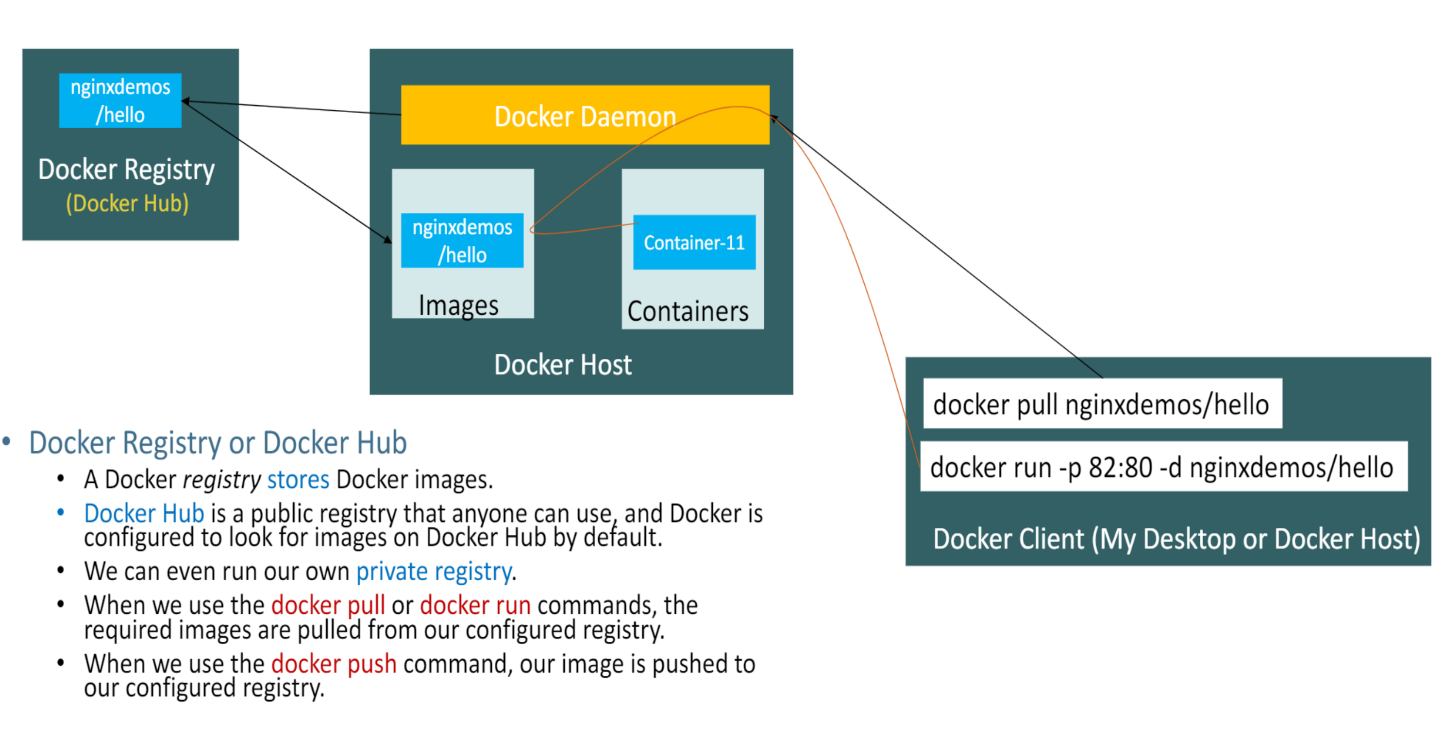
[stacksimplify/docker-fundamentals: Docker Fundamentals (github.com)](https://github.com/stacksimplify/docker-fundamentals)

****

What is docker Daemon? The Docker daemon is a service that runs on your host operating system. It currently only runs on Linux because it depends on a number of Linux kernel features, but there are a few ways to run Docker on MacOS and Windows too.

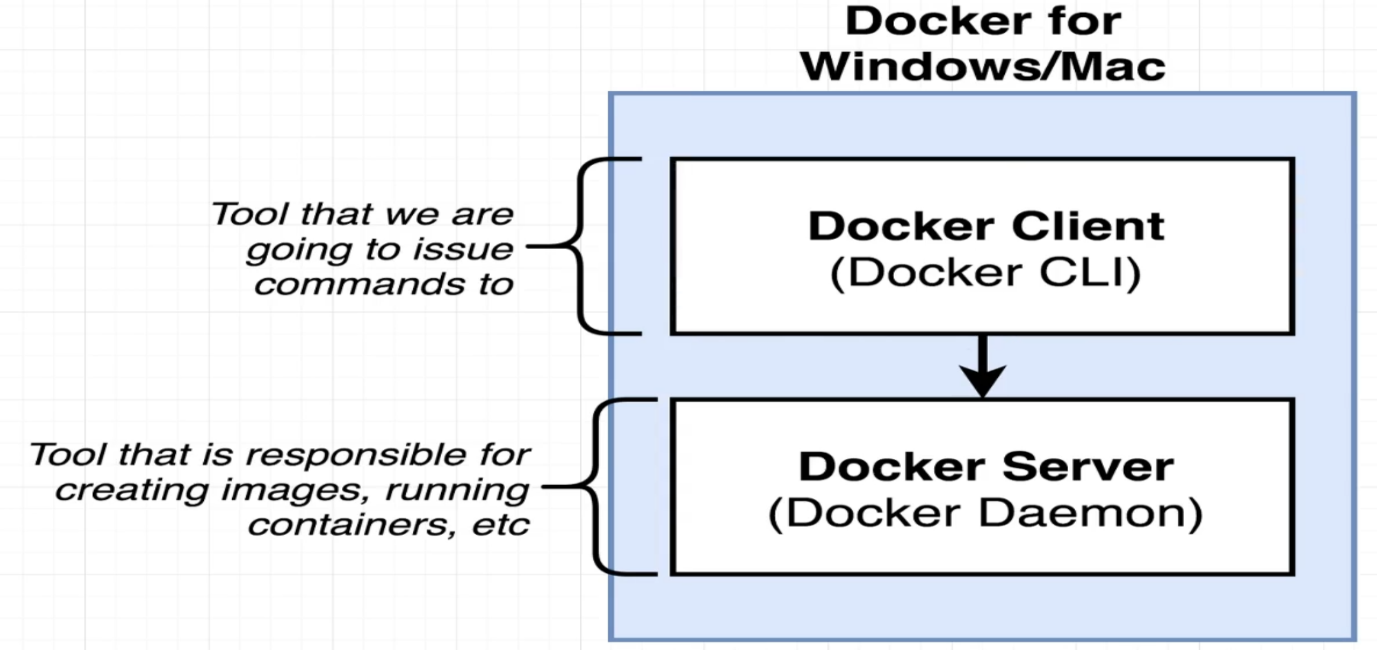
What is Docker Client? The Docker client is the primary way with which user can interact with Docker. When we run any docker command -> docker client send it to Daemon to perform action.

1. **Docker Terminology**



So, when we do -> docker pull -> it connects to Docker Demon API and route to Docker Registry to pull the image into Docker Host

Finally, when we do -> docker run -p 82:80 -d nginxdemos/hello -> it run the image in the form of container into docker host.

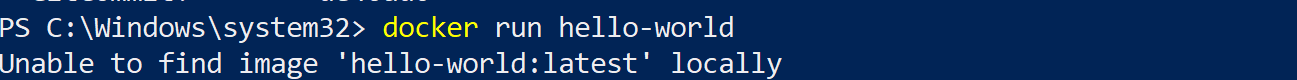


Two very important tools that we're going to be making use of throughout this course.

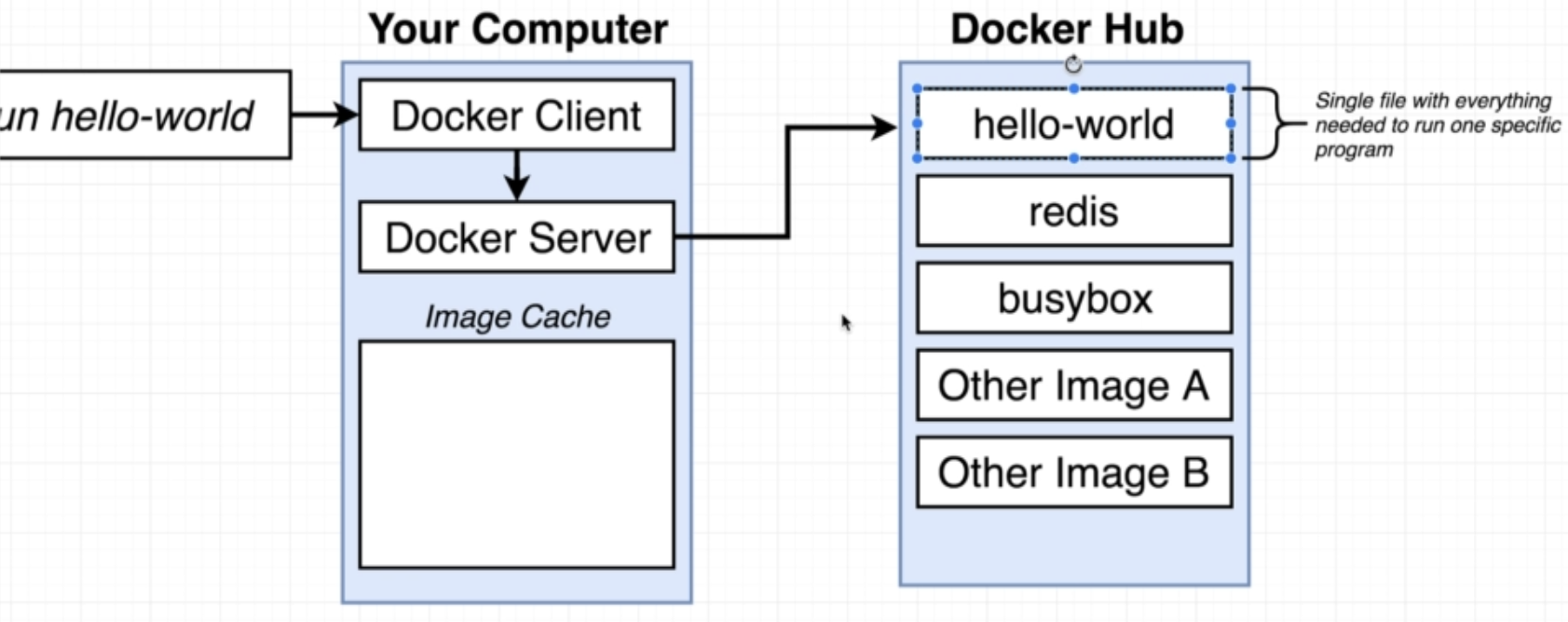
* Docker Client (Also known as Docker CLI) -> Enter commands to our terminal, issue them to Docker client. It's going to take our commands and figure out what to do with them, and help us interact with another piece of software known as Docker Server.
* Docker Server (Also known as Docker Demon) -> software that is responsible for creating container's images, maintaining containers, uploading images and doing just about everything you can possibly imagine around the world of Docker.

1. **Using Docker Client**

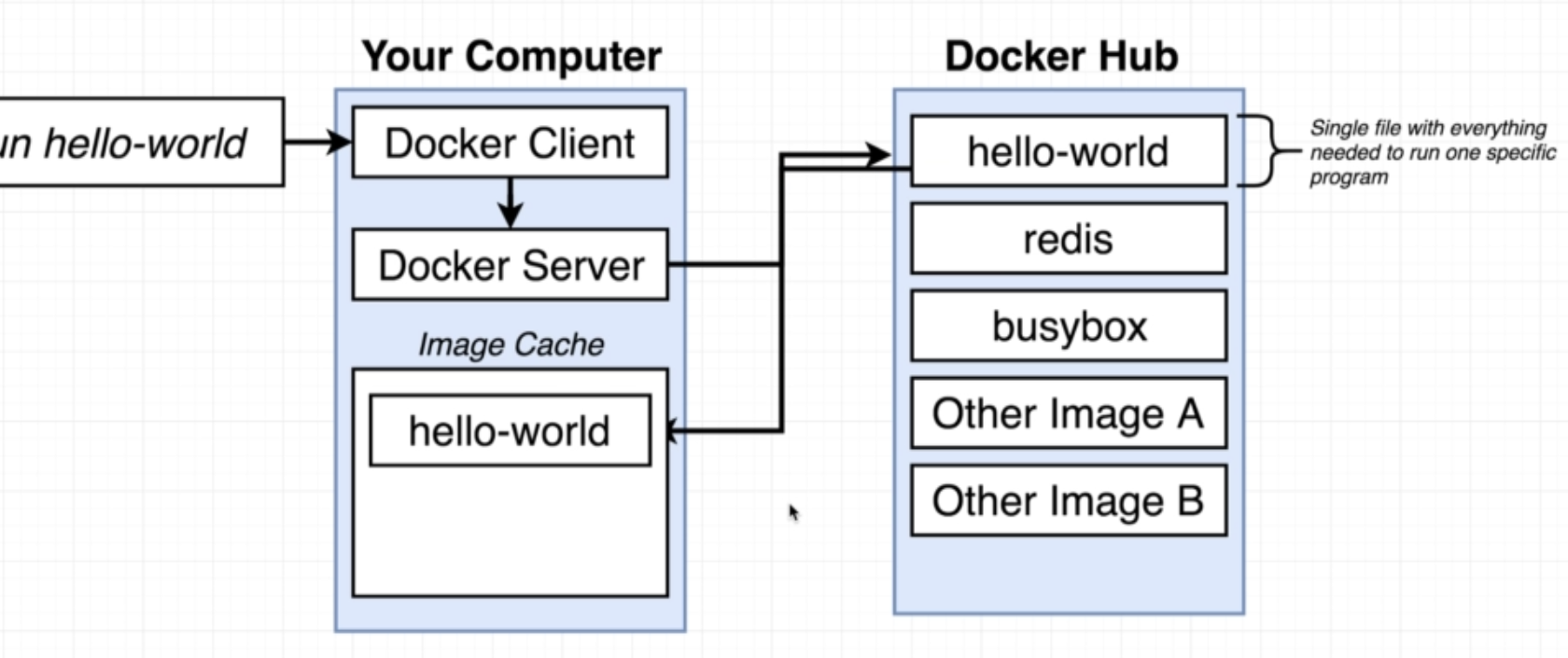
When we run any command in Docker CLI like below



* That means we wanted to start up a new container using the image with the name of Hello World. As we run this command -a series of actions very quickly occurred in the background.
* The Docker server saw that we were trying to start up a new container using an image called Hello World.
* The first thing that the Docker server did was check to see if it already had a local copy, like a copy on your personal machine of the Hello World Image or that hello world file. So the Docker server looked into something called the image cache.
* If no image of Hello World found -> Docker server decided to reach out to a free service called Docker Up. The Docker hub is a repository of free public images so you can freely download and run on your personal computer.



* So, Docker Server downloaded this hello world file and stored it on your personal computer in this image cache where it can now be reran at some point in the future very quickly without having to download it from the Docker hub.

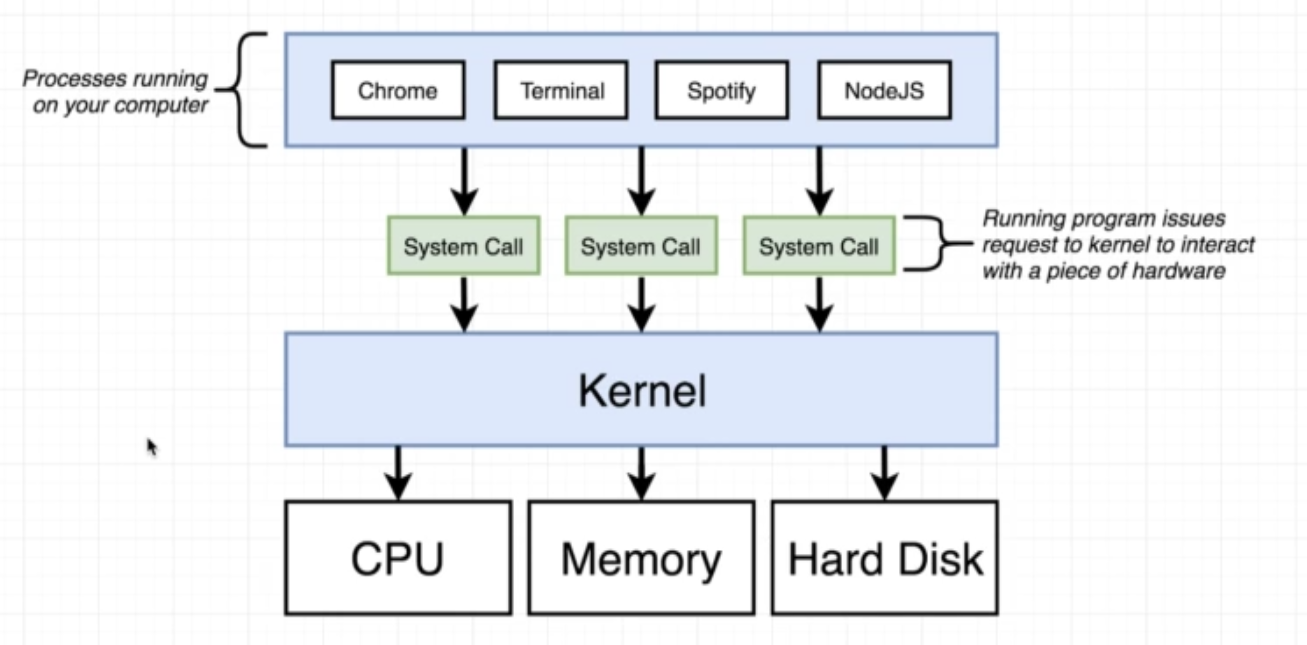


* The Docker server then said, OK, great, I've got this image and now it's time to use it to create an instance of a container.
* The Docker server then essentially took that single file, loaded it up into memory, created a container out of it, and then ran a single program inside of it.

1. **Basic of How System Works:**

So, in this section, I'm going to give you a behind the scenes look at what a container is and how it is created on your machine.

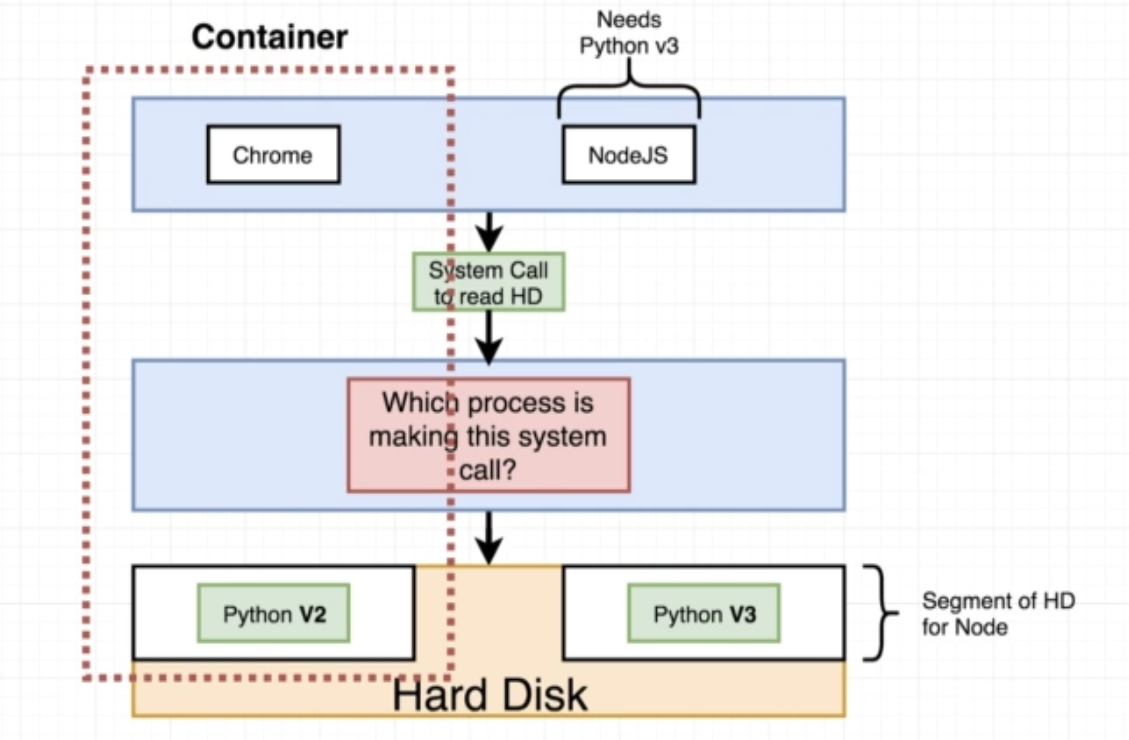
First need to have a little bit of background on exactly how your operating system runs on your computer.



* Operating systems have something called a kernel -> This kernel is a running software process that governs access between all the programs that are running on your computer and all the physical hardware that is connected to your computer as well.
* So the kernel is always kind of this intermediate layer that governs access between these programs and your actual hard drive.

1. **Container:**

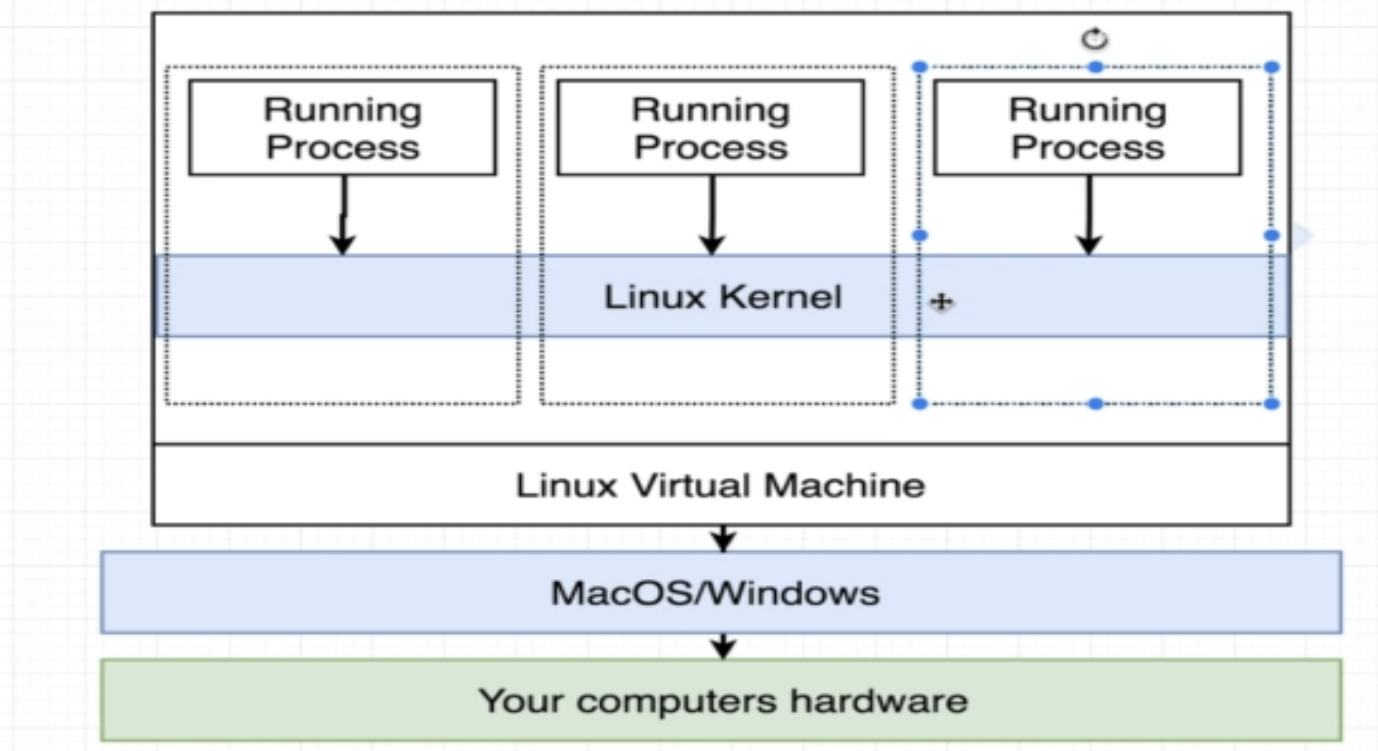
When people say, oh, yeah, I have a docker container, you really should not think of these as being like a physical construct that exists inside of your computer. Instead, a container is really a process or a set of processes that have a grouping of resources specifically assigned to it.



So when any process running into container -> The Process that sends a system call to a kernel. The kernel is going to look at that incoming system call and direct it to a very specific portion of the hard drive, the RAM CPU or whatever else it might need.

1. **How’s Docker Running in your Computer**

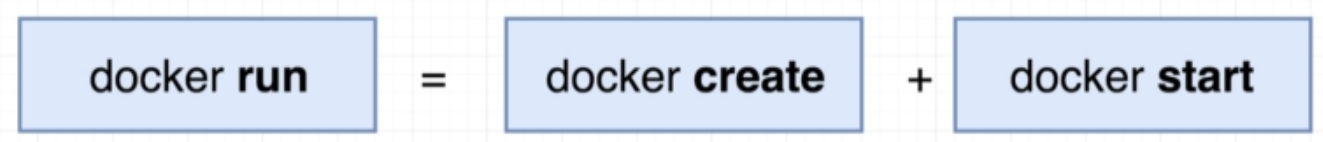
When you installed Docker for Windows -> you installed a Linux virtual machine. So so long as Docker up here is running, you technically have a Linux virtual machine running on your computer.



So inside the virtual machine, we have a Linux kernel and that Linux kernel is going to be hosting running processes inside of containers. So, I'm running a Linux virtual machine and that's what's being used to host all these different containers

1. **Container Lifecycle:**

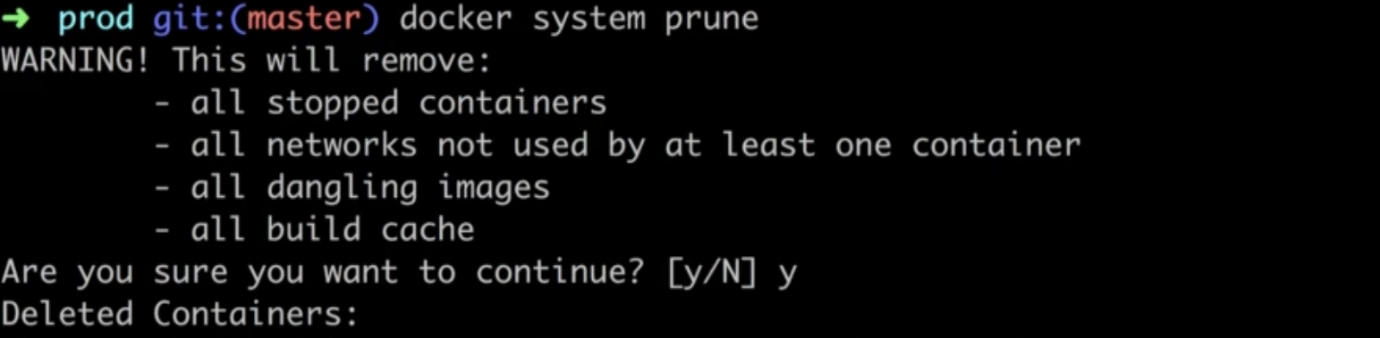
Docker Run is identical to running two other commands together. First, Docker create and then Docker start.



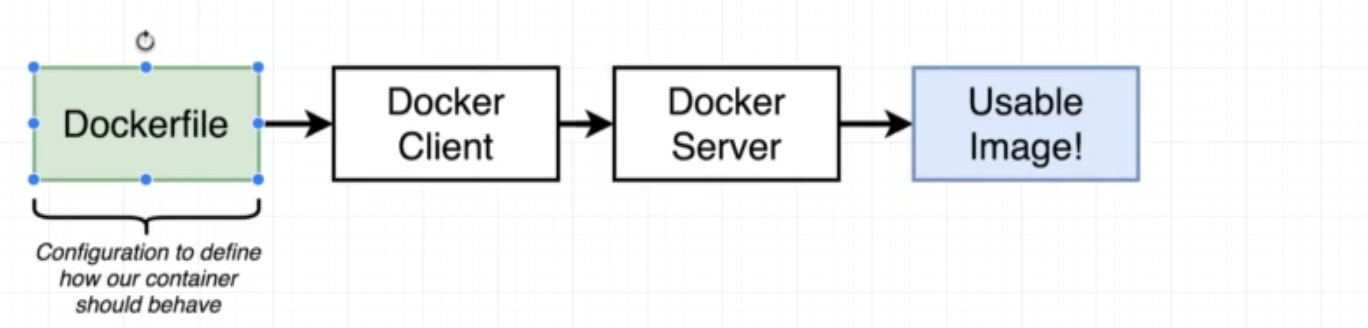
Docker ps –all -> command used to check how many images running



To remove container:



1. **Creating Image:**

****

* Firstly, we create something called a docker file, a darker file as essentially a plain text file that is going to have a couple of lines of configuration placed inside of it. This configuration is going to define how our container behaves or more specifically, what different programs it's going to contain and what it does when it starts up as a container.
* Once we create that docker file will then pass it off to the Docker client, which you'll recall is the Docker CLI that we've been using at our terminal.
* Docker server then going to take the doctor file, look at all the lines of configuration that we have inside of it, and then build a usable image that can then be used to start up a new container.