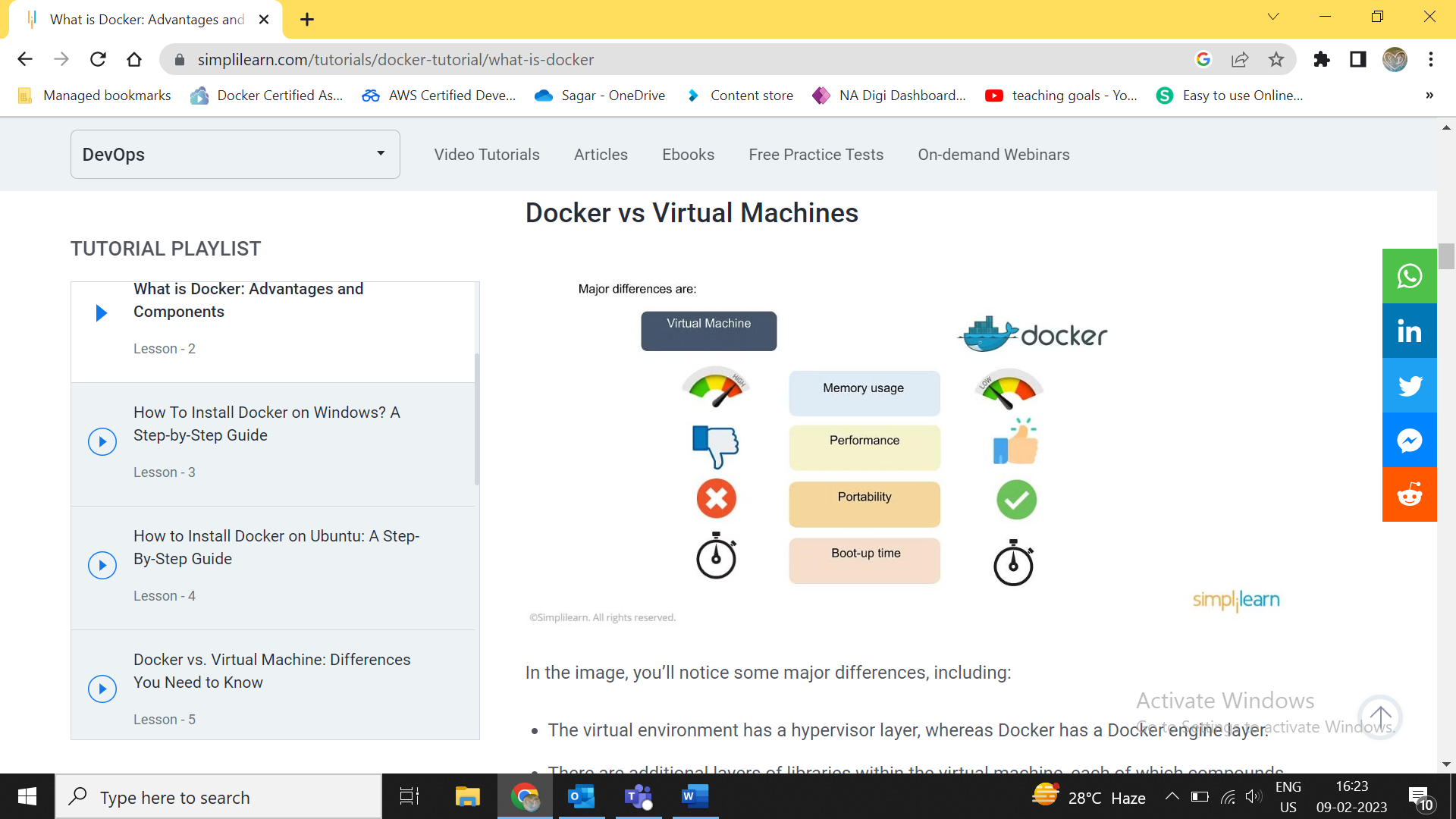
**Docker Advantages and Disadvantages**

**Docker vs Virtual machine**



**The virtual environment has a hypervisor layer, whereas Docker has a Docker engine layer.**

**There are additional layers of libraries within the virtual machine, each of which compounds and creates very significant differences between a Docker environment and a virtual machine environment.**

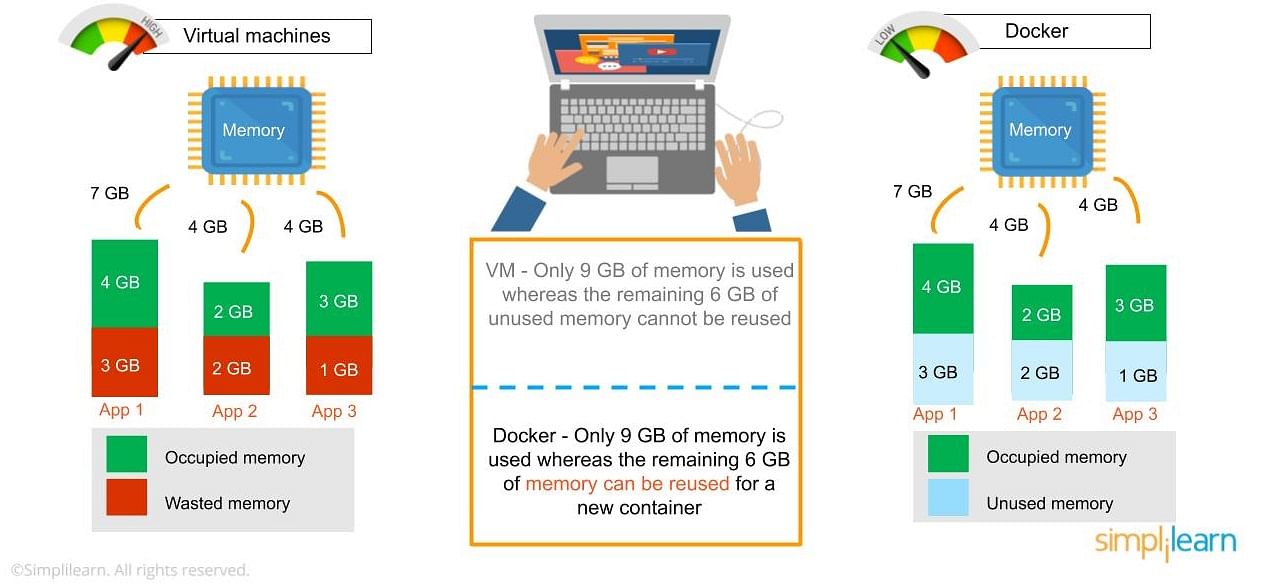
**With a virtual machine, the memory usage is very high, whereas, in a Docker environment, memory usage is very low.**

**In terms of performance, when you start building out a virtual machine, particularly when you have more than one virtual machine on a server, the performance becomes poorer. With Docker, the performance is always high because of the single Docker engine.**

**In terms of portability, virtual machines just are not ideal. They’re still dependent on the host operating system, and a lot of problems can happen when you use virtual machines for portability. In contrast, Docker was designed for portability. You can actually build solutions in a Docker container, and the solution is guaranteed to work as you have built it no matter where it’s hosted.**

**The boot-up time for a virtual machine is fairly slow in comparison to the boot-up time for a Docker environment, in which boot-up is almost instantaneous.**

**Docker vs Virtual Machines - Challenges**



**One of the other challenges of using a virtual machine is that if you have unused memory within the environment, you cannot reallocate it. If you set up an environment that has 9 gigabytes of memory, and 6 of those gigabytes are free, you cannot do anything with that unused memory. With Docker, if you have free memory, you can reallocate and reuse it across other containers used within the Docker environment.**

**Running multiples of them in a single environment can lead to instability and performance issues. Docker, on the other hand, is designed to run multiple containers in the same environment—it actually gets better with more containers run in that hosted single Docker engine.**

**Virtual machines have portability issues; the software can work on one machine, but if you move that virtual machine to another machine, suddenly some of the software won’t work, because some dependencies will not be inherited correctly. Docker is designed to be able to run across multiple environments and to be deployed easily across systems.**

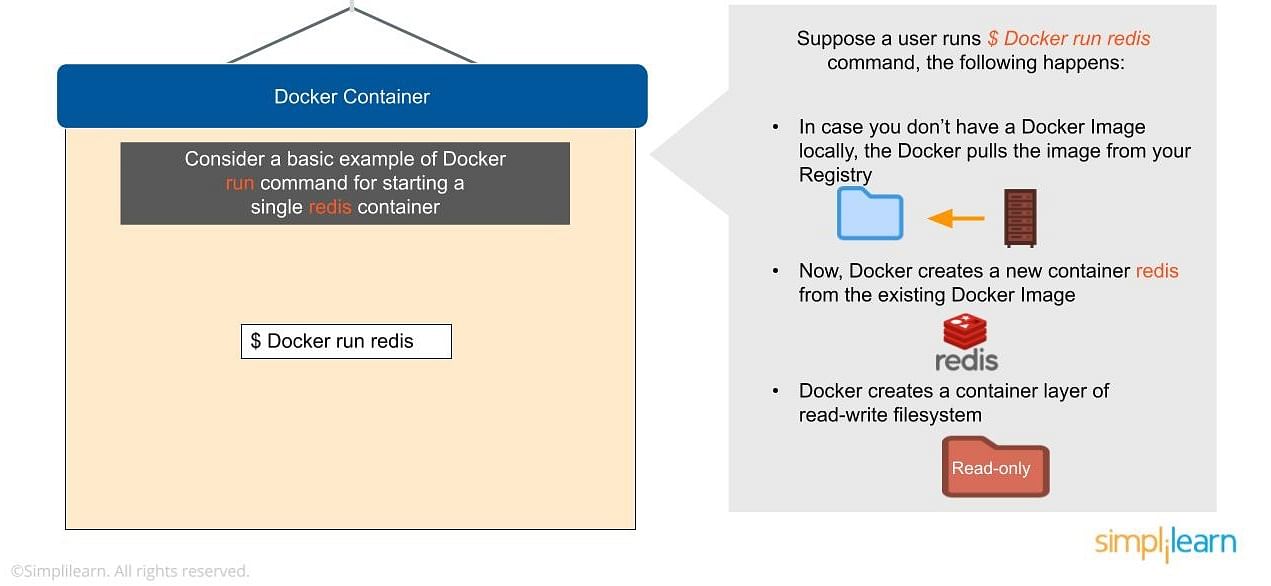
**The boot-up time for a virtual machine is about a few minutes, in contrast to the milliseconds it takes for a Docker environment to boot up.**

**Docker Registry**

**The Docker registry is where you would host various types of images and where you would distribute the images from. The repository itself is just a collection of Docker images, which are built on instructions written in YAML and are very easily stored and shared. You can give the Docker images name tags so that it’s easy for people to find and share them within the Docker registry. One way to start managing a registry is to use the publicly accessible Docker hub registry, which is available to anybody. You can also create your own registry for your own use internally.**

**The registry that you create internally can have both public and private images that you create. The commands you would use to connect the registry are Push and Pull. Use the Push command to push a new container environment you’ve created from your local manager node to the Docker registry, and use a pull command to retrieve new clients (Docker image) created from the Docker registry. Again, a Pull command pulls and retrieves a Docker image from the Docker registry, and a Push command allows you to take a new command that you’ve created and pushed it to the registry, whether it’s Docker hub or your own private registry.**

**Docker Container**



**Docker Compose**

**Docker-compose is designed for running multiple containers as a single service. It does so by running each container in isolation but allowing the containers to interact with one another. As noted earlier, you would write the compose environments using YAML.**

**So in what situations might you use Docker compose? An example would be if you are running an Apache server with a single database and you need to create additional containers to run additional services without having to start each one separately. ou would write a set of files using Docker compose to do that.**

**Docker Swarm**

**Docker swarm is a service for containers that allows IT administrators and developers to create and manage a cluster of swarm nodes within the Docker platform. Each node of Docker swarm is a Docker daemon, and all Docker daemons interact using the Docker API. A swarm consists of two types of nodes: a manager node and a worker node. A manager node maintains cluster management tasks. Worker nodes receive and execute tasks from the manager node.**

**Advantages of Docker**

**Rapid deployment**

**You can do rapid deployment using Docker. The environment itself is highly portable and was designed with efficiencies that allow you to run multiple Docker containers in a single environment, unlike traditional virtual machine environments.**

**Configuration files**

**The configuration itself can be scripted through a language called YAML, which allows you to describe the Docker environment you want to create. This, in turn, allows you to scale your environment quickly. But probably the most critical advantage these days is security.**

**Security**

**You have to ensure that the environment you’re running is highly secure yet highly scalable, and Docker takes security very seriously. You’ll see it as one of the key components of the agile architecture of the system you’re implementing.**

**Docker is fast**

**When I was a kid, and until I started using Docker containers in my late twenties, virtualization was synonymous with lethargy. Running something inside a virtual environment meant waiting for a virtual machine to boot up, then waiting even longer to start whichever particular application I needed to run. Even emulating DOS applications in DOSBox took a while.**

**With Docker, all of that waiting is a distant memory. By cutting out the need to boot up a full virtual machine, Docker lets me start apps in a virtual, software-defined environment much more quickly.**

**Docker is well documented**

**Docker’s feature set changes rapidly. The Docker team churns out new releases at a dizzying pace. Each release tends to add new features, and deprecate old ones.**

**Fortunately, the Docker team also does a nice job of documenting everything. The Docker documentation is reliably up-to-date. The docs usually make it very clear if information applies only to specific versions of Docker.**

**Docker’s solid documentation merits praise because many other software projects do a poorer job in this respect—especially ones that evolve as rapidly as Docker.**

**Docker has public container registries**

**One of the coolest things about Docker that people tend to overlook, I think, is the way it has made public repositories the go-to way to distribute and install software. I’m referring to Docker Hub, which hosts thousands of container images that anyone can grab in just a single command.**

**Through Docker Hub, Docker brings turn-key software distribution and installation to a new level. Repositories are no longer something you use just for source code or on Linux. With Docker, they become the default way to install software almost anywhere.**

**Disadvantages Of Docker**

**Of course, not all is perfect with Docker. Here are some features and characteristics that I dislike.**

**Docker storage is still hard**

**Better storage options for Docker containers are on the horizon. But the fact remains that today, there is no really seamless way to connect containers to storage. Docker Data Volumes require a lot of provisioning on the host and manual configuration. They solve the storage dilemma, but not in a really user-friendly or efficient way.**

**I have yet to try out most of the third-party storage solutions for Docker containers. I’m hoping they’ll smooth things over. But what I’d really like to see is a way to do storage with Docker itself that “just works.”**

**Docker has poor monitoring**

**Basically, the only type of monitoring solution that Docker offers is the stats command. That works fine if you need very basic information about your containers. But for more advanced monitoring, you’re out of luck.**

**Docker is platform-dependent**

**Docker now advertises itself as supporting Windows and Mac OS X as well as Linux. But it actually uses virtual machines to run on non-Linux platforms. At the end of the day, Docker is still Linux-only.**