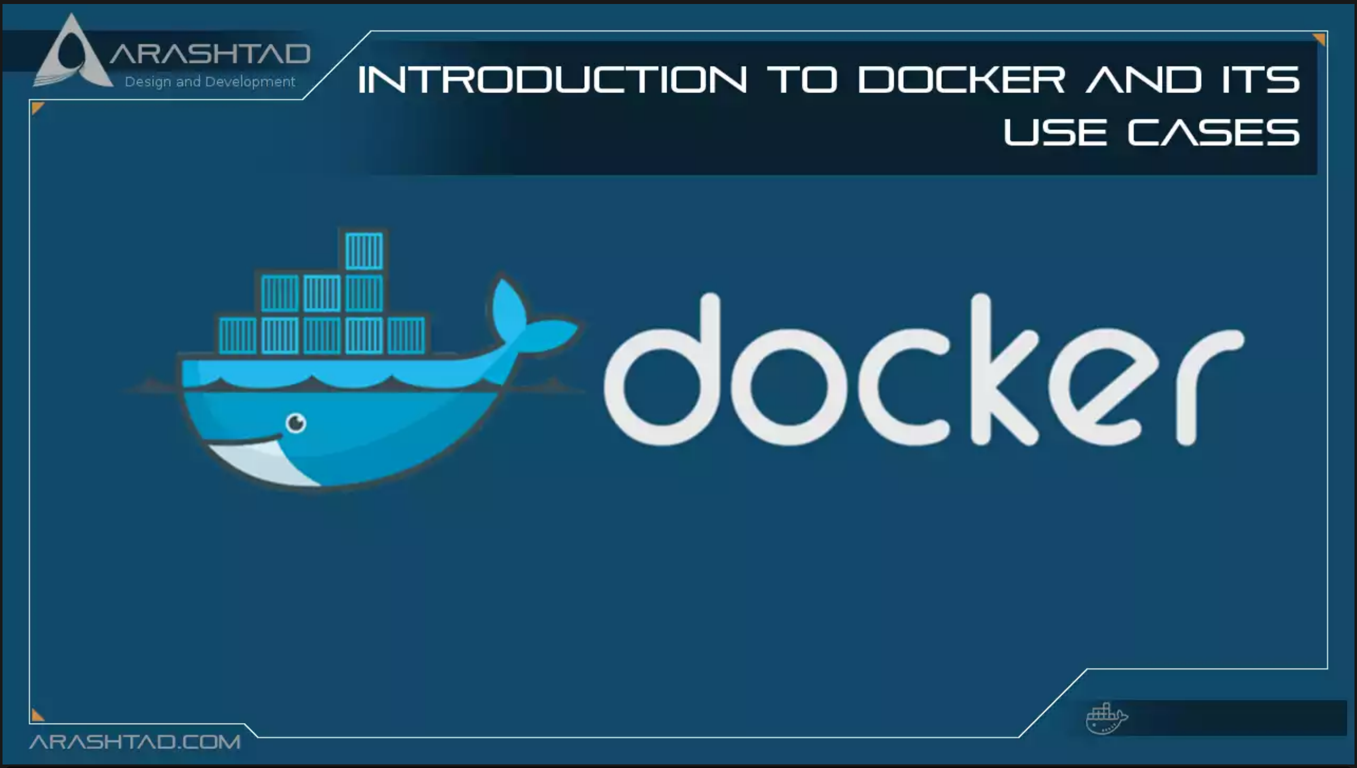
Introduction to Docker and its Use Cases

No comments



Several development teams are focusing on Docker as one of their favorite container-based platforms. It is becoming increasingly popular due to its reliability, performance, and functionality. The open-source containerization software and its underlying components must therefore be understood. In this article, we will focus on Docker, containers, use cases of Docker, its advantages, and its differences from Virtual Machines.

What is Docker?

The Docker containerization platform provides a lightweight virtualized environment called a container that allows developers to develop, deploy, and manage applications. It serves primarily as a development platform for distributed applications that can operate in a variety of environments. By making the software system agnostic, developers don't have to worry about compatibility issues. Since Docker utilizes virtualization to create containers to store apps, the concept may seem similar to virtual machines. Also, packaging apps into isolated environments (containers) makes it easier to develop, deploy, maintain, and use applications. In spite of the fact that both containers and virtual machines are isolated virtual environments used for software development, there are important differences between them. The most important difference is the Docker container's lighter, faster, and more resource-efficient nature.

What are Containers?

A Docker container is a lightweight virtualized runtime environment that can be used to run applications. Each container contains the necessary code, tools, runtime, libraries, dependencies, and configuration files to run a specific application. They are independent of the host and from all other instances running on the host. In Docker, containers are built by running an image on the Docker Engine. As these are the most common terms, you should be aware of the difference between Docker images and Docker containers. Unlike virtual machines, containers virtualize at the application level. In this way, the OS kernel is shared with the host and an operating system is virtualized over it, so resources are spared and lightweight virtual environments can be quickly and easily configured.

Using containers, we can isolate certain kernel processes and trick them into thinking they're the only ones running on a completely new computer. unlike virtual machines, containers can share the kernel of the operating system with only their respective binaries/libraries loaded. Thus, you don't have to install a whole separate OS (called a guest OS) inside your host OS. You can run multiple containers inside one operating system.

What are the Benefits of Using Docker?

1. New Developers can quickly Get started With the Team's Project

When a new developer starts to work on your product, he still has to install some local servers, set up a database, install libraries, integrate third-party software, and configure it all. How long does it take? From a few hours to many days. Even if your project has an excellent onboarding manual that explains all the steps, there is a high probability that it won't work for every laptop and system configuration. A new developer's onboarding is often a collaborative effort by the whole team that guides him or her through the installation of all missing parts and resolving system problems. The Docker container automates all these manual installation and configuration steps, so the developer only has to launch Docker and run a single command (docker-compose up), and that does all the work for him. No matter if he uses Docker on a macOS, Windows, or Linux computer.

2. You can Test Your App in all Environments:

You must have heard of a famous developer's excuse “It worked on my machine”. Whether something works on your server or on the developer's computer does not guarantee that it will work. Additionally, the software versions may not match the missing libraries or tools. Besides, there are many different computers and servers out there, and their configuration may differ in so many ways. if you do not use Docker, you have to configure them manually, which can be time-consuming and prone to error.

In order to solve this problem, Docker bundles not just the code but also all the components that need to run an application or website. These components are inside Docker containers, which are isolated and independent of the outside world. This allows them to run in a predictable and consistent manner everywhere. This means developers spend less time fixing issues and more time delivering new features.

3. No vendor Lock-in When it comes to Hosting

The Docker container is similar to the cargo container. Docker containers, just like cargo containers, are standardized, so most computers, servers, and cloud computing (such as container ships, freight trains, and trucks) can handle them. It gives you great flexibility in terms of your hosting technology and provider. You can start with a small, cheap server and scale it into a larger, more expensive one when necessary. And what's more, you won't be stuck with just one host.

4. Flexible scaling of your app or website

While Docker won't make your website or web application scalable out of the box, it may be a key component of software scalability. especially if you use AWS or Google Cloud to host your website. The right containers can be launched in many copies to handle the growing number of users. In the cloud, this scaling can be automated, so if more people use your site, more containers will be launched. The same applies to downsizing, fewer users mean a smaller hosting bill.

5. Keep track of all changes made to your application components

The Docker container enables developers to easily define a programmable environment for the code to run in. This eliminates the need for written or spoken instructions to be executed by humans. No more manual installation or configuration. Docker converts all these manual processes into code, so there will be no more misunderstandings when a developer installs a library without notifying others that it is required to launch the application. An automated installation and configuration process that is automatically performed on every developer's computer, as well as on the server. And this saves a lot of time during development.

6. Makes your application easier to compose with third-party apps

It has already been mentioned that Docker helps you bundle your code with third-party software inside closed containers. However, it also provides a vast repository (Docker Hub) where developers can find templates (called Docker Images) for creating containers for their website or web application. Using the configuration already created by the Docker community reduces the development time since no new configuration needs to be created, and the development team will save time by not starting from scratch.

Differences between Docker and Virtual Machines:

Both Docker Containers and Virtual Machines are containers that can store code and mix it with additional software, files, and configuration. At first glance, these two may seem similar. Virtual machines require a separate operating system (like Linux or Windows) to be installed inside of them. and operating systems are not the lightest pieces of software. In other words, if you have a lot of virtual machines, it will get very heavy (both in terms of disk space and computing power).

Docker containers, instead of having their own operating system, use the operating system of the computer/server where Docker is installed. In the Docker engine, computer resources can be accessed by the containers in a way that makes them feel like they have their own operating system.

Conclusion:

In this article, you got familiar with Containers, Docker, its benefits and use cases, and its differences from Virtual Machines. In summary, with Docker, you can easily build, test, and deploy applications. Docker packages software into standardized units called containers that contain libraries, system tools, code, and runtime so it can run quickly. You can deploy and scale applications in any environment with Docker and know that the code will run.

Join Arashtad Community

Follow Arashtad on Social Media

We provide variety of content, products, services, tools, tutorials, etc. Each social profile according to its features and purpose can cover only one or few parts of our updates. We can not upload our videos on SoundCloud or provide our eBooks on Youtube. So, for not missing any high quality original content that we provide on various social networks, make sure you follow us on as many social networks as you're active in. You can find out Arashtad's profiles on different social media services.

Get Even Closer!

Did you know that only one universal Arashtad account makes you able to log into all Arashtad network at once? Creating an Arashtad account is free. Why not to try it? Also, we have regular updates on our newsletter and feed entries. Use all these benefitial free features to get more involved with the community and enjoy the many products, services, tools, tutorials, etc. that we provide frequently.

SIGN UP NEWSLETTER RSS FEED