Table of Contents

What Is Docker & How Is It Related to Containerization?

Running applications in containers instead of virtual machines is gaining momentum in the IT world. The technology is considered to be one of the fastest growing in recent the history of the software industry. At its heart lies Docker, a platform that allows users to easily pack, distribute and manage applications within containers. In other words, It is an open-source project that automates the deployment of applications inside software containers.

Docker really makes it easier to create, deploy, and run applications by using containers. And containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package. By doing so, the developer can be assured that the application will run on any other Linux machine regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code.

Docker: Statistics & Facts

- 2/3 of Companies that try using Docker, adopt it. Most companies who will adopt have already done so within 30

days of initial production usage, and almost all the remaining adopters convert within 60 days.

- Real Docker adoption is up 30% in one year.
- Adopters quickly increase their container. Docker adopters approximately quintuple the average number of running containers they have in production between their first and tenth month of usage.
- PHP, Ruby, Java and Node are the main programming frameworks used in containers1
- Top technologies run on Docker.

Popularity & Benefits of Using Docker

Why do large companies like ING, PayPal, ADP and Spotify keep using Docker? Why is the adoption of docker growing so fast? Let’s go over the top advantages of docker to better understand it.

1. Return on Investment & Cost Savings

The first advantage of using Docker is the ROI. The biggest driver of most management decisions when selecting a new product is the return on investment. The more a solution can drive down costs while raising profits, the better the solution is, especially for large, established companies, that need to generate steady revenue on the long term.

In this sense, Docker can help facilitate this type of savings by dramatically reducing infrastructure resources. The nature of Docker is that fewer resources are necessary to run the same application. Because of the reduced infrastructure requirements that Docker has, organizations are able to save on everything from server costs to the employees needed to maintain them. Docker allows engineering teams to be smaller and more effective.

2. Standardization & Productivity

Docker containers ensure consistency across multiple development, release cycles and standardising your environment. One of the biggest advantages to a Docker-based architecture is actually standardization. Docker provides repeatable development, build, test, and production environments. Standardizing service infrastructure across the entire pipeline allows every team member to work on a production parity environment. By doing this, engineers are more equipped to efficiently analyze and fix bugs within the application. This reduces the amount of time wasted on defects and increases the amount of time available for feature development.

As we mentioned, Docker containers allow you to commit changes to your Docker images and version control them. For example, if you perform a component upgrade that breaks your whole environment, it is very easy to rollback to a previous version of your Docker image. This whole process can be tested in a few minutes. Docker is fast, allowing you to quickly make replications

and achieve redundancy. Also, launching Docker images is as fast as running a machine process.

Docker enables you to build a container image and use that same image across every step of the deployment process. A huge benefit of this is the ability to separate non-dependent steps and run them in parallel. The length of time it takes from build to production can be sped up notably.

4. Compatibility & Maintainability

Eliminate the “it works on my machine” problem once and for all. One of the benefits that the entire team will appreciate is parity. Parity, in terms of Docker, means that your images run the same no matter which server or whose laptop they are running on. For your developers, this means less time spent setting up environments, debugging environment-specific issues, and a more portable and easy-to-set-up codebase. Parity also means your production infrastructure will be more reliable and easier to maintain.

5. Simplicity & Faster Configurations

One of the key benefits of Docker is the way it simplifies matters. Users can take their own configuration, put it into code and deploy it without any problems. As Docker can be used in a wide variety of environments, the requirements of the infrastructure are no longer linked with the environment of the application.

6. Rapid Deployment

Docker manages to reduce deployment to seconds. This is due to the fact that it creates a container for every process and does not boot an OS. Data can be created and destroyed without worry that the cost to bring it up again would be higher than affordable.

7. Continuous Deployment & Testing

Docker ensures consistent environments from development to production. Docker containers are configured to maintain all configurations and dependencies internally. So, you can use the same container from development to production making sure there are no discrepancies or manual intervention.

If you need to perform an upgrade during a product’s release cycle, you can easily make the necessary changes to Docker containers, test them, and implement the same changes to your existing containers. This sort of flexibility is another key advantage of using Docker. Docker really allows you to build, test and release images that can be deployed across multiple servers. Even if a new security patch is available, the process remains the same. You can apply the patch, test it and release it to production.

8. Multi-Cloud Platforms

This is possibly one of Docker’s greatest benefits. Over the last few years, all major cloud computing providers, including Amazon Web Services (AWS) and Google Compute Platform (GCP), have embraced Docker’s availability and added individual support.

Docker containers can be run inside an Amazon EC2 instance, Google Compute Engine instance, Rackspace server or VirtualBox, provided that the host OS supports Docker. If this is the case, a container running on an Amazon EC2 instance can easily be ported between environments, for example to VirtualBox, achieving similar consistency and functionality. Also, Docker works very well with other providers like Microsoft Azure, and OpenStack, and can be used with various configuration managers like Chef, Puppet, and Ansible,etc.

9. Isolation

Docker ensures your applications and resources are isolated and segregated. Docker makes sure each container has its own resources that are isolated from other containers. You can have various containers for separate applications running completely different stacks. Docker helps you ensure clean app removal since each application runs on its own container. If you no longer need an application, you can simply delete its container. It won’t leave any temporary or configuration files on your host OS.

On top of these benefits, Docker also ensures that each application only uses resources that have been assigned to them. A particular application won’t use all of your available resources, which would normally lead to performance degradation or complete downtime for other applications.

10. Security

And the last benefit of using docker is security. From a security point of view, Docker ensures that applications that are running on containers are completely segregated and isolated from each other, granting you complete control over traffic flow and management. No Docker container can look into processes running inside another container. From an architectural point of view, each container gets its own set of resources ranging from processing to network stacks.

Docker Success Stories

Let’s look at success stories of well-known companies, which implemented Docker and are very happy with the results.

ADP Case Study

ADP is one of those companies that keep using Docker to better manage their application infrastructure. ADP is the largest global provider of cloud-based human resources services. From payroll to benefits, ADP handles HR for more than 600,000 clients, which caused a challenge in terms of security and scalability. To solve the security issue, ADP uses Docker Datacenter. Docker Content Trust enables their IT ops team to sign images and ensure that only signed binary will run in production. They also perform automated container scanning. Using multiple Docker Trusted Registries enables them to build a progressive trust workflow for their applications development process.

To solve the scalability issue, the company relies on Universal Control Plane/Swarm. Swarm gives their team the ability to first start small and have each application made up of many small Docker engine swarms instead of one swarm per application. Then the swarms will merge over time, becoming larger and in the end each application will have its own swarm. One day, a swarm could potentially span across public and private infrastructure and across applications. This will enable the business to make the best financial decision for the company. With Docker containers, ADP plans to containerize the most dynamic parts of their applications first making it easier to change and re-deploy them moving forward, while leaving the other areas of the application for a later time. Containerizing with Docker enables ADP to have a hybrid strategy. They will have a mix of big and small containers for any application, which creates an evolutionary path forward to microservices

The vision and goal of ADP is to get to microservices but the reality is that no company will get there overnight. Not all applications will be refactored at the same rate and the platform needs to be flexible to accommodate a variety of application architectures. Now, by slowly isolating services into separate containers, ADP is able to slowly grow into a microservices architecture using Docker, rather than doing it all overnight.

Spotify Case Study

A digital music service with millions of users is running a microservices architecture with as many as 300 servers for every

engineer on staff. The biggest pain point Spotify experienced managing such a large number of microservices was the deployment pipeline. With Docker, Spotify was able to pass the same container all the way through their CI/CD pipeline.

From build to test to production, they were able to ensure that the container that passed the build and test process was the exact same container that was on production.

Now the company can guarantee that all of their services remain up and running, providing a great user experience for their customers. They also built a new platform called Helios based on Docker containers to deploy their containers across their entire fleet or servers and maintain their development ecosystem.

ING Case Study

As one of the top 10 financial services companies in the world, ING operates on global scale. The IT organization in the Netherlands alone comprised of 1,800 people creating unique challenges of coordinating change across large groups of people, processes, and technology and it leads to poor quality software. Now, ING is able to move faster with their CD pipeline running in Docker containers. Key areas accelerated are provisioning build servers, provisioning and publishing tests, deployment automation, and in the functional integration testing environment across their 180 teams. Additionally, the increasing levels of automation was starting to strain their infrastructure resources and Docker helped to greatly reduce that utilization and

ultimately hard costs, especially within some of their biggest development efforts.

Conclusion

To conclude I want to say that Docker containers share their operating system, so they run as isolated processes regardless of the host operating system. As Docker proudly admits, this means that its containers can “run on any computer, on any infrastructure and in any cloud.” The portability, flexibility and simplicity that this enables, is a key reason why Docker has been able to generate such strong momentum. We are big fans of using Docker and we believe that it will continue growing.