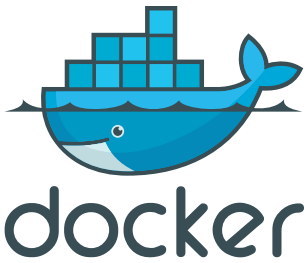
The Definitive Guide to Docker



Executive Summary

Enterprise teams are in the midst of digital transformation. Driven by the business' need to become more agile, deploy applications more frequently, and ensure security, combined with the need to reduce costs associated with legacy apps, enterprises are now changing not only the technology but also the way their personnel interact with each other. They're slowly moving away from bare metal servers and giant monolithic applications, and are now adopting hybrid cloud strategies, refactoring monolithic apps into microservice based applications and bringing IT Ops and developer teams closer by going DevOps.

Docker is the containerization technology enabling these initiatives. But what is a container and what is the value to enterprise teams?

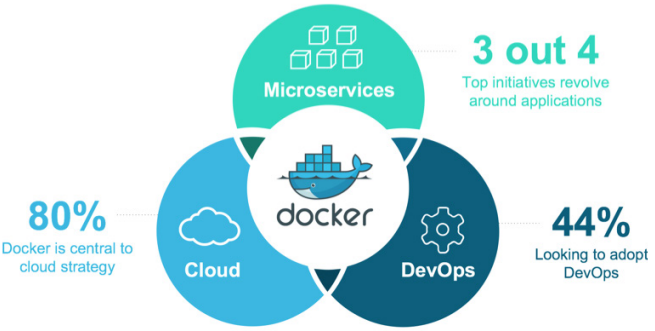
This guide provides an introduction to Docker technology, containerization, and how it's driving enterprise initiatives forward.

Intended for IT professionals new to Docker, this paper will discuss the following topics:

- Key enterprise technology initiatives underway
- Overview of Docker technology
- Paths to getting started with Docker Containers

Key enterprise initiatives call for Docker containers

In early 2016 we commissioned a survey of over 500 IT operations and developers to understand how applications are being built and unveil the top initiatives of enterprise application teams. We found that the three major enterprise initiatives underway include: cloud adopting, microservices and going "DevOps."



State of App Development Survey: Q1 2016

Cloud Adoption - Migrating application workloads to the cloud has been an important enterprise initiative in recent years. Cloud infrastructure enables teams to become more scalable and frees them from having to maintain their own servers that drive up CapEx and OpEx costs. Because of this, cloud service providers like AWS, Azure and Google Cloud Enterprise have all exploded in popularity, giving enterprise teams the ability to run their applications within their own private cloud, and scale them on an as needed basis, without having to maintain servers in house. But migrating applications to the cloud requires portability.

Microservices - Fueled by a desire to become more agile and innovate faster, organizations are moving away from waterfall style development practices traditionally linked to monolithic applications in favor of distributed applications and DevOps methodologies. With traditional applications, each additional line of code introduced grew the QA cycles exponentially and ultimately slowed innovation in favor of testing and bug fixing. The promise of distributed applications is that a collection of micro services can be developed independently, frequently and freely. This does introduce some complexity though as maintaining order becomes a challenge as a new matrix headache emerges.

What is a distributed or micro services application?

Unlike monolithic apps that a single application server containing an ever-expanding code base, microservice applications are a collection of loosely coupled smaller applications.

DevOps - This new approach to the way applications are being built has also led to teams rethinking how IT operations and developers teams interact with each other. Today 44% of enterprise teams are looking to break down the traditional barriers that have existed between IT Operators and Developers. This cultural shift has been deemed "going DevOps." This requires a platform that enables IT to manage and secure the environment, while enabling developers to build applications in a self service manner.

Enter Docker, a platform that leverages a new approach to applications delivering agility, portability and control by packaging up the application and all of its dependencies into a standardized unit of software – the Docker container.

Docker Container Technology & Key Terminology

Enterprise teams have turned to Docker to help drive their initiatives forward. Docker is the world's #1 containerization platform in which developers use to create applications, and IT operations teams use to secure and manage their application environment. Here are some key Docker terms you should know:

Dockerfile - This is the starting point of the dockerization process. The Dockerfile details the configuration of an application and specifies resources needed. This tells the image builder (i.e Jenkins) what the image should look like.

Docker Image - Created via the dockerfile. The image is a snapshot of the application. These artifacts are stored and managed in a Docker registry (i.e Docker Trusted Registry).

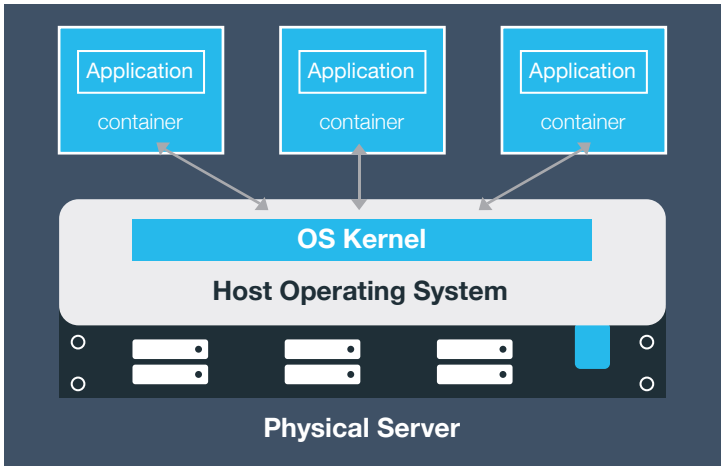
Docker Container - The standard and isolated unit in which the application is packaged together with all of its libraries and binaries. At run time, the engine reads the image and spins up a container

Docker Engine - The container runtime with built in orchestration, networking and security that installs on any physical, virtual (VM) or cloud host (AWS, Azure, Google Cloud Enterprise etc).y.The lightweight runtime installs directly on the host OS i.e Windows Server 2016, Ubuntu, CentOS, RHEL OpenSUSE.

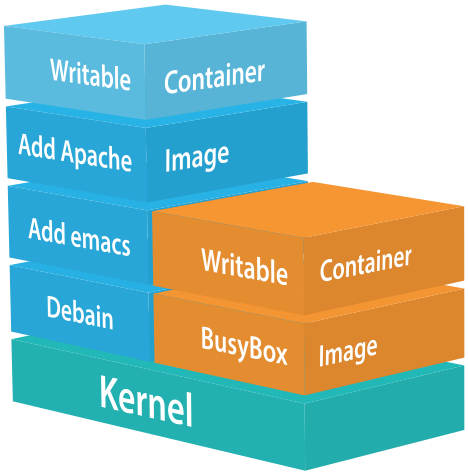
Docker Registry – A service where Docker images are stored, secured and managed. Multiple versions of an image (application) called "tags" can be stored within repositories (folders) within a registry. This enables teams to easily update applications by tweaking the image, creating a new tag, then storing the new version within the registry.

Understanding Docker Containers

A Docker container wraps up a piece of software in a complete filesystem that contains everything it needs to run: code, runtime, system tools, system libraries – anything you can install on a server. By encapsulating and isolating everything in a container, this guarantees that the container will always run the same, regardless of the environment it is running in.



Containers are built from a Docker image. The Docker image uses union filesystems and is comprised of multiple layers.

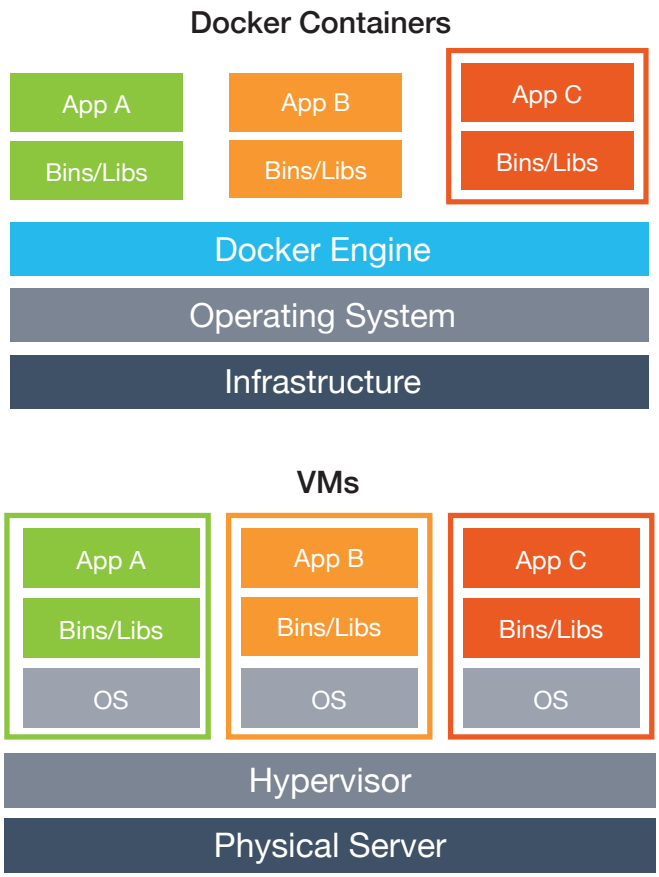


Once given the "docker run" command the Docker Engine spins up a container from the defined image (it can also spin them down). Every command is executed to the Docker engine including: creating new containers, scaling existing containers, stopping, removing and much more.

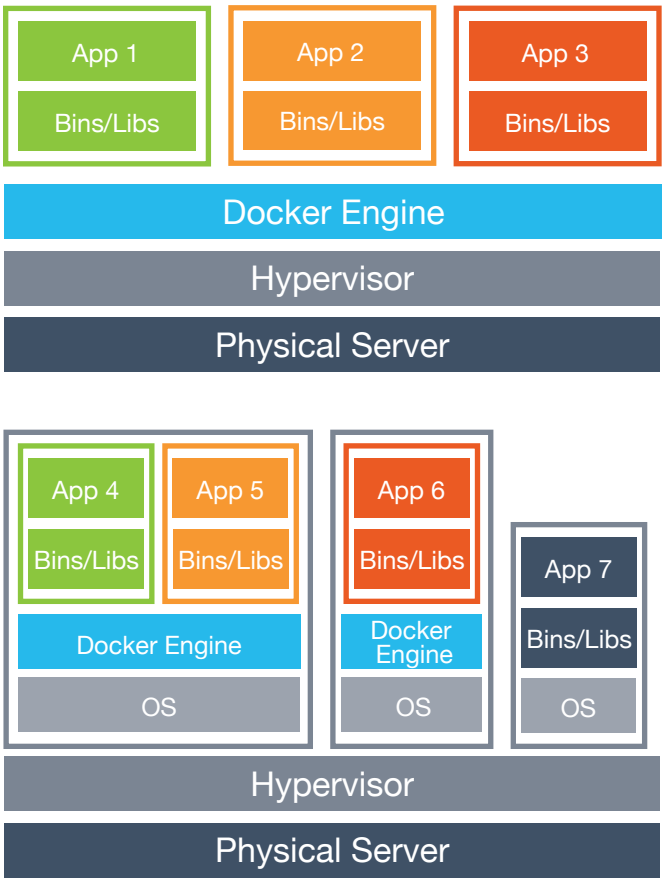
```
Examples
docker run ubuntu:14.04 echo "Hello World"
docker run ubuntu ps ax
```

What about VMs?

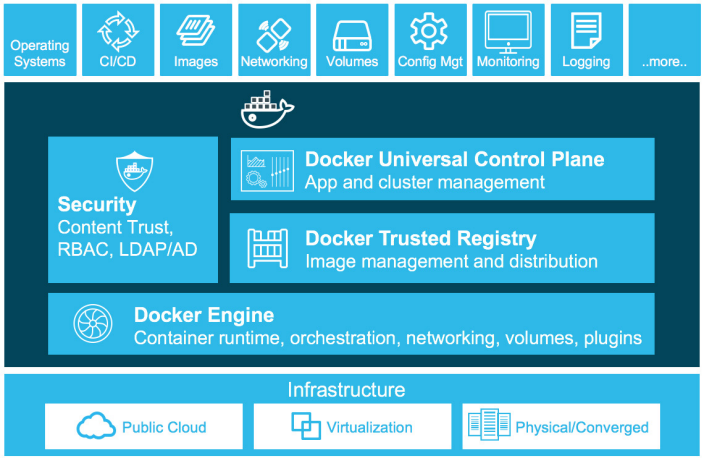
For years enterprise teams have relied on virtualization technology to help them optimize their environment. Virtual machines were seen as the status quo for packaging and running applications, but they're heavyweight and require an entire OS within them. Docker containers are not VMs nor even lightweight VMs. Their architectures are completely different. The image below displays the key differences between Docker containerization and virtualization. Docker containers share the OS kernel on the host, while with virtualization each VM has a full copy of an OS inside the VM.



This does not mean these two models are mutually exclusive. Docker containers run anywhere a Docker Engine is installed, and the Docker Engine runs anywhere i.e on bare metal, in VMs (vSphere, Hyper-V) and clouds (AWS, Azure, Google and more). This also means that Docker containers are portable from any environment to the another without having to recode the application. Additionally many users add containers into existing virtual infrastructure to increase the density of workloads possible per VM. Some enterprise teams have realized a 20x increase in infrastructure optimization.



Docker containers are part of the overall Docker platform and lifecycle. Docker Datacenter, our commercial solution, includes container management, registry and the commercially supported Docker engine, delivering Container as a Service (CaaS) to the enterprise. Below is a quick look at the Docker Datacenter platform.

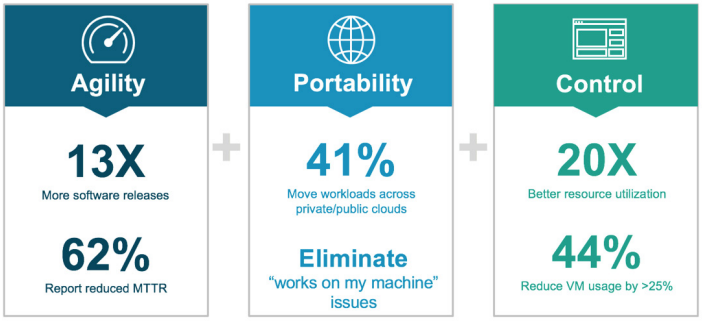


CaaS, delivered by Docker Datacenter, allows enterprise teams to have the agility, portability and control that IT operations and developer teams require.

Agility - Developer teams can quickly create and patch new microservices applications or containerize existing applications and hand them off to the IT operations team. The IT operations team can pull the image from the registry, spin up an application and quickly deploy the application to production. Teams can go from taking multiple months to deploy a single application to a streamlined DevOps workflow that has been proven to help teams deploy 13x more often.

Portability - Dockerized applications can run on any infrastructure and within any environment allowing teams to migrate applications to the cloud. IT operations teams take their portable containerized applications and run them within the cloud service provider (i.e AWS, Azure, Google Cloud etc.) of their choosing using Docker's container orchestration tools. They can also leverage multiple cloud service providers and move applications from private to public clouds.

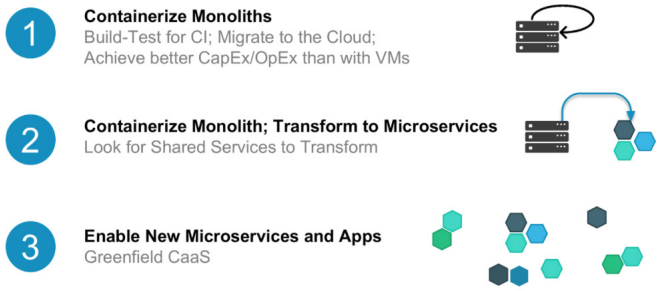
Control - Developers have the ability to define what their microservices and containerized legacy applications needs to run, and patch them quickly if bugs are found. IT operations teams can manage and deploy these distributed applications in production, cutting down on the management complexity that microservice applications introduce. They can also move containers to the cloud provider that makes the most business sense to help optimize infrastructure, reduce costs and improve the security of their environment.



The 2016 State of The Application Survey

Three Paths to Adopting Docker

When it comes to actually getting started with Docker containers, teams begin differently based on their application needs and their desired outcome. Here's a look at three of the most common ways:



1. Containerize existing apps - Using Docker containers to containerize existing legacy (monolithic) applications. We call this the "lift and shift" strategy. In this case, customers like Merck benefit from the portability that Docker containers provide. This portability enables them to streamline their software supply chain and develop a CI/CD strategy. They can then run these containers (their apps) within VMs, allowing them to optimize resources and spend less on CapEx and OpEx.

2. Containerize then refactor - Customers, like ADP for example, start out by containerizing their existing legacy applications and then overtime slowly refactor their applications into microservices. They often prioritize the refactoring of their dockerized apps from monolithic to microservices based on the application's importance to the organization and how often IT operators interact with it.

3. Build new microservice apps - Using Docker as the way to help teams build new microservices applications. Docker containers package up each of the distributed services and deploy them as a single running application in production. This is the case with DoubleDutch, an innovative start-up customer of ours located here in Silicon Valley.

At Docker our mission is to help teams embrace digital transformation. As you now know, when it comes to creating microservices, going DevOps and migrating to cloud it takes more than just containers. The real power lies within the Docker platform itself, and its ability to deliver a secure and managed application environment that gives teams agility, portability and control when developing and delivering applications. A platform built by Docker, for Docker, that comes complete with the tools you need to build, ship and run your applications, and with support from the Docker team.

Docker Datacenter delivers the platform enterprise teams need. Read about Docker Datacenter here: www.docker.com/ddc.

For more information take a look at the following resources:

- Docker for the Enterprise: www.docker.com/enterprise
- Docker Datacenter solution: www.docker.com/products/docker-datacenter
- Docker Self-paced Training: https://training.docker.com/self-paced-training
- Contact our sales team: www.docker.com/contact
- See Docker customers: www.docker.com/customers

www.docker.com