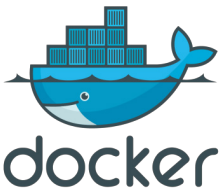
SOLUTION BRIEF

NetApp Docker Volume Plugin*
Intel® Xeon® Processors

Merging Enterprise Applications with Docker* Container Technology

Enabling Scale-out Solutions for a Changing World



Intel Enables NetApp* Storage Security and Performance

NetApp scale-out storage solutions built on the Intel® Xeon® processor E5 v4 and Intel® Xeon® processor D families give IT organizations the tools that they need to meet increasing storage and security demands.

NetApp storage systems rely on the Intel® Data Protection Technology with Advanced Encryption Standard New Instructions (Intel® AES-NI) feature in Intel Xeon processors to provide non-disruptive volume-level encryption. Intel AES-NI accelerates volume encryption by providing AES-specific instructions on the silicon itself.

In today's rapidly evolving business world, organizations are increasingly turning to cloud solutions to help accelerate the time-to-market of new web-based software and services. Yet software development teams can struggle with legacy, monolithic applications that were never designed for cloud architectures. IT operations teams might also have difficulty maintaining service level agreements (SLAs) for legacy applications as the frequency of deployments increases.

Docker* containers have emerged as a way to provide the agility that development teams need while delivering the stability and reliability required by IT operations. Developers can bring existing applications into the Docker ecosystem while building new applications using microservice design principles. IT operations teams can benefit from reduced complexity and faster deployment of containerized applications.

However, Docker containers do not retain data after being restarted or retired. A container's ephemeral nature presents challenges when container-generated data must be retained, or when a container must share its data across multiple Docker hosts. Docker and NetApp solve these issues and more with the NetApp Docker Volume Plugin* (nDVP*).

The nDVP adds persistent storage capabilities to Docker containers, which gives developers the ability to enable data persistence in applications that use Docker container technology. In addition, the nDVP brings the advanced storage capabilities of NetApp storage systems, including NetApp ONTAP*, SolidFire*, and NetApp E-Series platforms*, to Docker deployments. These storage platforms provide features such as transparent scalability, high availability, encryption, data deduplication, and seamless replication.

Docker software's speed and flexibility, combined with NetApp's storage expertise, can help organizations create a development and deployment environment that can enable a rapid response to constantly changing business requirements.

Docker Fundamentals

The Docker technology borrows concepts from hardware virtualization, but it takes application abstraction a step further. Hardware virtualization has led to a revolution in computing by abstracting the underlying hardware from the operating system. This approach makes it possible to run multiple operating systems and applications in virtual machines on a single physical host. Virtualization also enables users to easily provision a complete application stack, including the operating system, applications, and services that an application relies upon.

Yet while virtualization has proven itself a valuable technology for server consolidation and application flexibility, it does have limitations that appear as organizations push to further increase application densities. Virtualization still requires CPU and memory overhead because virtual machines must contain complete operating systems. This need can result in a significant investment in patching and lifecycle management.

Docker software removes some of the limitations of virtualization to help reduce CPU and memory overhead and increase application density per host. Systems architects and engineers familiar with virtual infrastructure might think that Docker software is a new twist on virtualization. However, Docker software is not a virtualization technology; rather, it is an application delivery technology that abstracts an application and its dependencies from the operating system. Instead of running an application within a full virtual machine, Docker software lets developers package an application with a file system, libraries, and any other dependencies that the application needs within a Docker container, which can run on a bare-metal server or in a virtual machine. The difference is that Docker containers use a shared operating system kernel model on the host instead of relying on isolation through hardware virtualization. Only the application and its dependencies are deployed within the container.

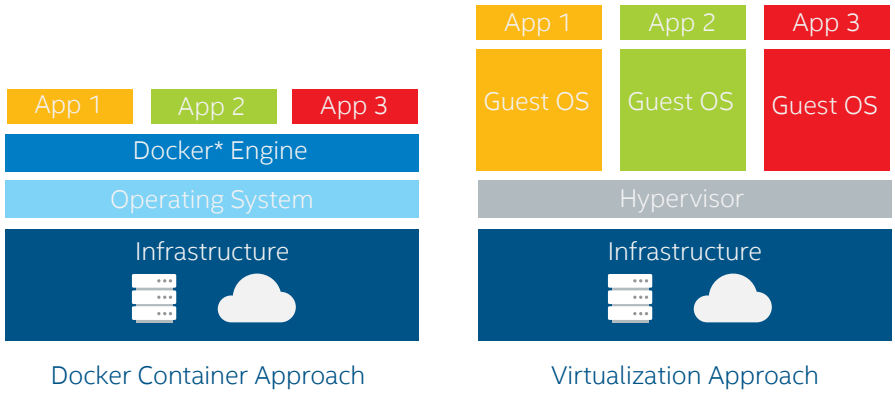


Figure 1. The Docker* architecture helps increase application density by sharing the operating system among multiple containers, which removes the additional guest operating system overhead present with traditional virtualization technologies

The Docker container approach provides a number of benefits for both developers and IT operations teams:

- **Simplicity:** Docker containers remove a level of complexity from the development and deployment process because developers only need to package an application and its dependencies. IT operations benefit from the simplicity of containers because Docker containers run independently in Docker environments, regardless of the underlying operating system configurations.
- **Speed:** Docker containers share the kernel of the underlying host operating system, but the Docker architecture protects containers from each other by isolating container processes. Each container application views itself as running within its own environment—similar to how an application runs in a virtual machine—but the container doesn't require a separate operating system. This lightweight environment relieves the application of operating system and hypervisor overhead. Development and IT operations teams can create and deploy new containers quickly, which can help reduce development and deployment time and increase team productivity.
- **Interoperability:** Development teams can link multiple Docker containers or services to create multi-tiered application stacks. If an individual container within the stack needs to be upgraded, a developer can swap out the container without affecting the rest of the containers within the stack.

- **Portability:** Docker provides a standardized method to deploy containers to any number of Docker environments, whether an environment is located on a development workstation, a private cloud infrastructure, or a public cloud service. This portability simplifies development and deployment by removing host-configuration dependencies from the process.
- **Density:** Docker containers provide a lightweight runtime environment for applications, which can lead to increased application density per host. In addition, Docker containers let developers create "microservice architectures" easily, where a single service runs within a single container without interfering with other services running in other containers.

Docker also provides Docker Datacenter* (DDC), an integrated, end-to-end platform that enables agile application development and management. DDC integrates with existing enterprise infrastructure to power container-as-a-service (CaaS) solutions that can extend from the data center to the cloud.

Docker Datacenter includes:

- **Docker Universal Control Plane (UCP)** with embedded Docker Swarm* for integrated management and orchestration of the Docker environment
- **Docker Trusted Registry (DTR)** for Docker image management, security, and collaboration
- **A commercially supported Docker engine** that provides a robust container runtime
- **Docker API** support for seamless integration with Docker's command-line interface (CLI), Docker Compose, and Docker Swarm

The Docker application-container approach, combined with the CaaS capabilities of DDC, lets developers create and deploy applications using self-service tools, which can lead to increased developer productivity and rapid application delivery.

Docker Architecture

Docker uses a client/server architecture that is composed of five elements:

- **The Docker client** communicates with a Docker server. The client can reside on the same host as the server or on another host.
- **The Docker engine** processes commands from the client and does the work of building, deploying, and running Docker containers. In a data center environment, this software typically runs on hardware powered by Intel® Xeon® processors.
- **Docker images** are the layered basic building block of a Docker container. Docker image layers can contain multiple components of an application stack. For example, a Docker image might contain operating system libraries not available on the host in one layer, an application server in another layer, and application binaries in a third layer.
- **Docker registries** are repositories for Docker images. These registries are used to store and distribute Docker images.
- **Docker containers** are the execution and storage environment for applications. Containers use Docker images as a foundation, and they contain all of the libraries needed for an application to run, in addition to a file system.

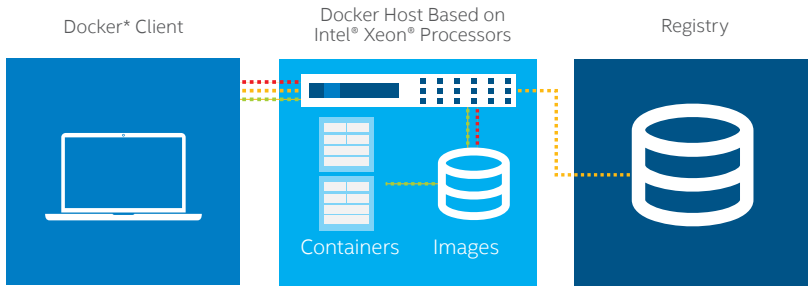


Figure 2. The core components of a Docker* infrastructure

Docker images and containers are central components of the Docker architecture. Images share similarities to base or "gold" virtual machine images in a virtual infrastructure. A gold virtual machine image typically consists of a virtual disk containing a file system, a full operating system, and the components needed to run an application. Docker images also contain these components, but they differ in that they share the operating system kernel with the host.

Docker containers are similar to cloned virtual machines that use a gold virtual machine image as a template. A gold virtual machine image remains static, while the virtualization software writes changes to a cloned virtual machine elsewhere. This method allows many copies of a virtual machine to run using a single gold image, which significantly decreases the amount of disk space required by the cloned virtual machines. Docker uses a similar method. When a developer instantiates a Docker container, the Docker host adds a writeable container layer to the Docker image. As the container runs, the Docker image remains static. No data is written to the Docker image, but the Docker host writes any changes made to the container to the container layer itself. This read/write method is often known as copy-on-write. The benefit of this technology is that because the image never changes, multiple containers can be run using a single image.

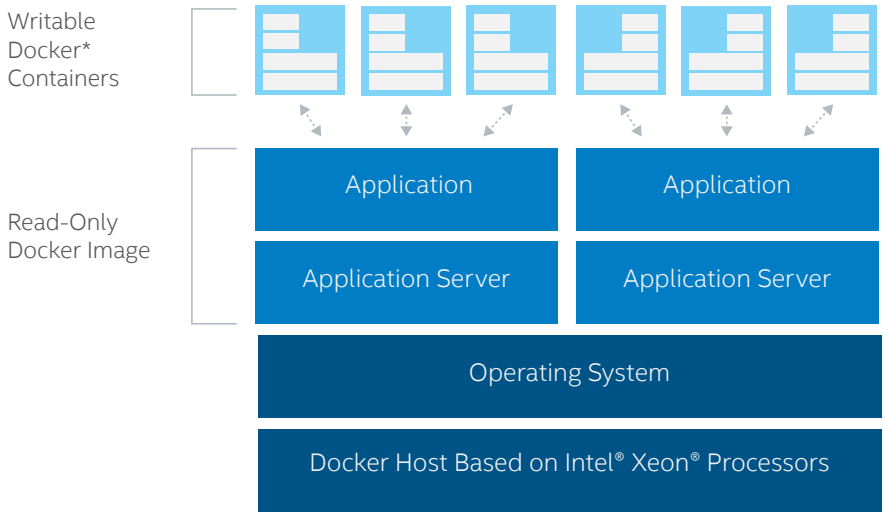


Figure 3. Docker* software uses application images to enable fast spin-up of writable containers, which helps reduce disk usage

Docker Container and Data Persistence

Many traditional enterprise applications—including applications from leading enterprise vendors and applications built in-house—were developed using monolithic application standards. The software is typically a single, complex application that combines code for data access, business logic, security, integration, and user-interface components into a set of large binaries. Building and maintaining these applications can be slow and difficult, as the code for the entire application might need to be recompiled and deployed even when developers make small changes. In addition, monolithic applications often can only scale up instead of scaling out, which makes taking advantage of on-demand

NetApp and Docker Collaborate to Expand Docker* Ecosystem Capabilities

As a Docker Ecosystem Technology Partner, NetApp produced the NetApp Docker Volume Plugin* (nDVP*), which provides enterprise-grade persistent storage to Docker environments. The nDVP tool enables datavolume provisioning and management within popular container-management tools, such as Docker Swarm* and Docker Datacenter*, while also giving organizations the ability to embrace cloud-native strategies across both private cloud and public cloud technologies.

cloud services difficult. A microservice architecture can help overcome these complex application challenges, but the effort and cost required to rewrite legacy applications often must be spread over a long period of time.

Organizations can take advantage of a hybrid approach, where developers "containerize" existing monolithic applications within a Docker container while pursuing a microservices approach. This can help organizations take advantage of Docker container benefits while paving the way for rewriting the application using modern design methodologies.

Traditional applications or applications built using certain types of microservices, such as MySQL*, PostgreSQL*, or WordPress*, require data to persist. For example, a traditional application might write state data to the local file system, or a microservice such as MySQL might write customer data to its local database. Yet by default, Docker containers are ephemeral in nature in that they don't persist application data. While a Docker container is running, Docker software writes all application-specific data to the container's copy-on-write file system. When a container no longer needs to run and is exited, the Docker engine removes the copy-on-write layer. This removal results in all application data written to the container file system being lost. Any modified files within the container revert to their pre-startup state. This ephemeral nature can present challenges for both hybrid and microservice architecture approaches.

Docker data volumes can help mitigate this condition. A Docker data volume is a directory within a container file system that can persist data regardless of the container's lifecycle. Developers can define data volumes when creating a new container, or they can configure containers to use any folder—whether local or shared through mechanisms such as Network File System (NFS) or Internet Small Computer System Interface (iSCSI)—as a persistent location. Docker containers can use data volumes to retain data across container restarts, share data with the host operating system, or share data between multiple containers, but the storage must be provisioned and managed outside of the Docker ecosystem.

The NetApp Docker Volume Plugin*

A core principle that guides Docker software's development is "batteries included, but replaceable." This principle has led to an architecture that provides certain capabilities out of the box, but also provides a plug-in mechanism that lets technology providers replace key components.

From a storage perspective, Docker's architecture provides a storage-volume framework that enables volume management, regardless of the underlying storage system. NetApp and Docker have worked together to create nDVP, a plug-in that enables integration between Docker and NetApp storage systems, including ONTAP, SolidFire, and NetApp E-Series platforms.

The nDVP brings storage management capabilities to Docker environments that rely on the advanced storage capabilities of NetApp systems. Large Docker software deployments typically require enterprise class storage systems that provide high availability, fault tolerance, efficient management of disk space, and feature-rich management tools. The nDVP helps provide and manage these enterprise class features by delivering an integrated, easy-to-use mechanism for managing containerized application-data storage.

The nDVP uses the Docker software's storage-volume framework to give Docker software users the ability to manage storage for Docker containers across a NetApp storage fabric. A Docker host running one or more nDVP instances can manage volumes across multiple NetApp storage platforms. This capability lets Docker containers take advantage of enterprise-class storage features provided by NetApp storage platforms, such as data protection, high availability, resiliency, and storage efficiency. The nDVP also lets Docker users consume disparate NetApp storage devices with varying capabilities and costs, which provides greater control over the type of storage used by applications. For example, application containers requiring low latency and high density can consume storage from flash-based SolidFire, NetApp EF-Series*, and NetApp All Flash FAS* (AFF) storage systems.

NetApp storage platforms also provide multiprotocol support for data-volume storage using NFS and iSCSI. Docker software can use data volumes stored on NFS to share data among multiple Docker containers simultaneously. Data sharing gives organizations greater flexibility when deploying multi-tiered application stacks that use services across multiple Docker containers.

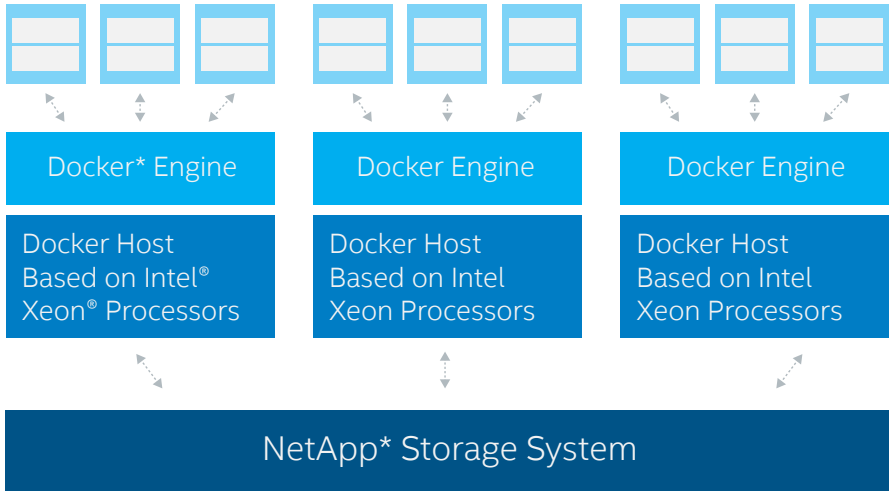


Figure 4. NetApp* technologies brings enterprise-grade storage features to the Docker* ecosystem

The capabilities provided by NetApp storage systems and the nDVP help ease the transition between running traditional applications within Docker containers and developing new applications around a microservices architecture. Software development teams can take advantage of the persistence features provided by the nDVP, while IT operations can keep vital data generated by containerized applications highly available and secure using the mature storage capabilities of NetApp systems.

FIND OUT MORE

NetApp storage solutions: **netapp.com**

Intel: **intel.com**

Docker: **docker.com/enterprise**

NetApp and Docker: Enabling Scale-out Solutions for a Changing World

Docker technology running on servers powered by Intel Xeon processors provides a powerful foundation for rapid development and deployment of web-based services. As demand for these services increases, NetApp and Docker can provide growing Docker deployments with enterprise-class storage and scalability. With the nDVP, developers can build applications that easily persist data across Docker container instances while taking advantage of the scale-out and high-availability capabilities of NetApp storage systems.

Printed in USA 0317/JS/PRW/PDF Please Recycle 335097-002US