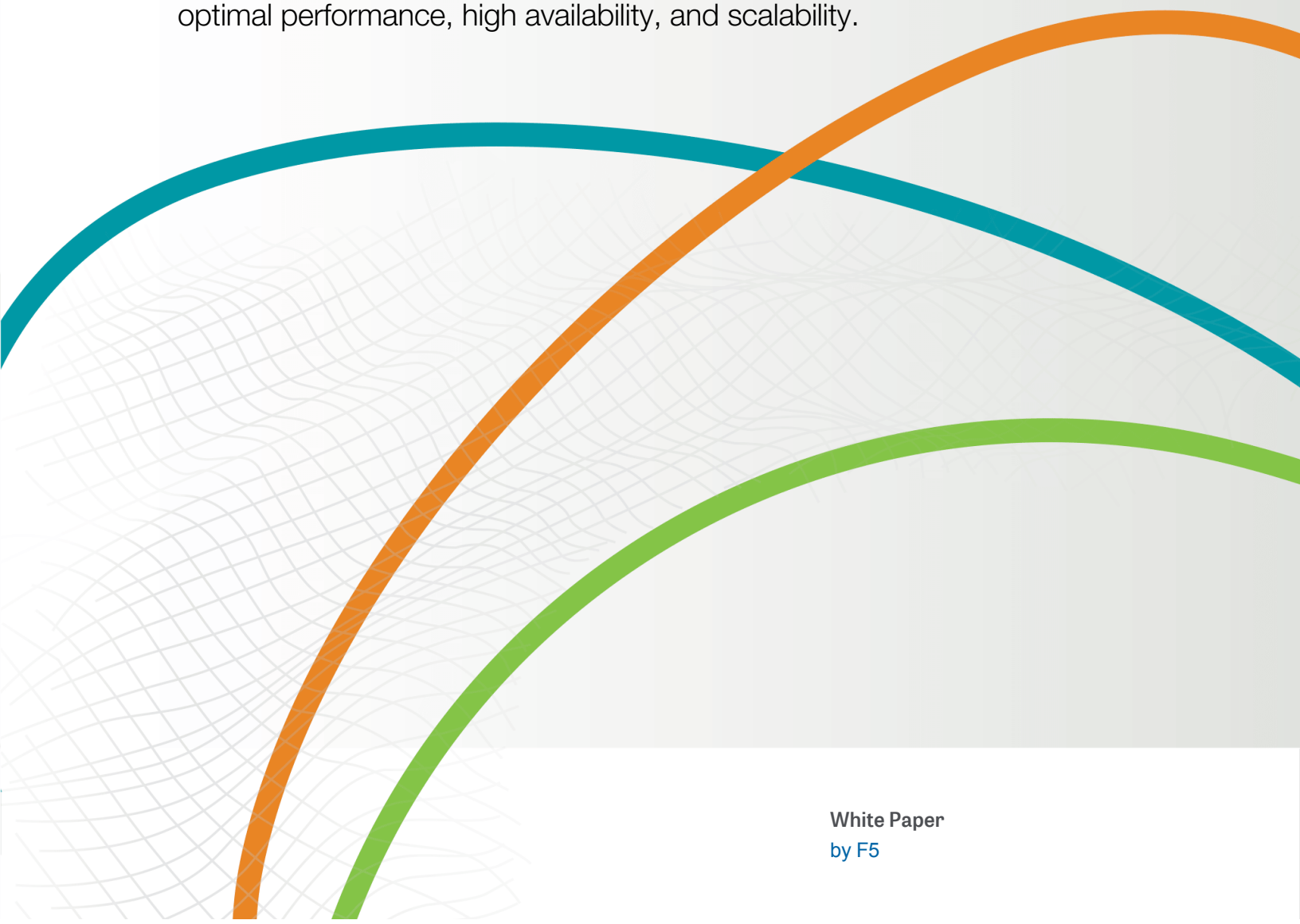
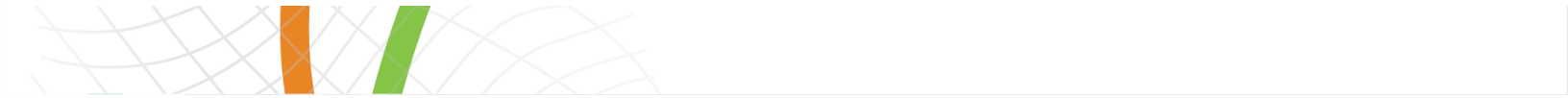
Using Docker Container Technology with F5 Products and Services

Docker is an emerging technology that promises a more optimal application delivery lifecycle and lower overhead in the data center and public or private cloud. Learn the concepts behind Docker and how F5 solutions integrate to provide secure application delivery, optimal performance, high availability, and scalability.



White Paper

by F5



Executive Summary

The evolving needs of IT and the advent of agile development and deployment strategies has led to the emergence of "containerization," an alternative to full machine virtualization in which an application is encapsulated in a container with its own operating environment. Containerization is an attractive solution that enables developers to iterate faster. It also offers additional benefits that address the overhead associated with virtual machines, allowing for higher utilization of resources in the software-defined data center (SDDC).

Although containerization isn't a new concept, Docker, developed by Docker, Inc., has been widely cited as the implementation of choice due to its broad industry support, standardization, and comprehensive breadth of capability. In the company's words, Docker is "an open platform for building, shipping, and running distributed applications. It gives programmers, development teams, and operations engineers the common toolbox they need to take advantage of the distributed and networked nature of modern applications." As such, Docker simplifies application lifecycle management from development to deployment and enables application portability. This simplification is critical for enterprises, considering that there are multiple hosting options for an application, either in the public cloud or private cloud infrastructure.

This paper outlines F5's direction on using containers within F5 technology and for supporting Docker for application delivery and security. Before we discuss this strategy, it is important to recognize data center pain points and why these technologies are critical for the next generation enterprise application delivery.

Note: This document is meant for IT decision makers, architects, and developers. It is assumed that the reader has prior knowledge of virtualization technology, software development, and the release lifecycle process.

Data Center Infrastructure Challenges

Several recent studies on data center infrastructure pain points have identified a consistent set of needs for evolving the data center:

- Faster application deployment methods
- An improved workflow management process
- Increased availability and utilization of compute resources
- Improved agility to move workloads as needed

As new trends emerge in application development, enterprise customers are shifting their view of the application lifecycle management model to the following:

- Next-generation applications are increasingly built for "cloud first."
- Linux has become the de-facto operating system (OS) for cloud development.
- Next generation cloud applications:
  - Are designed to be stateless and utilize a loosely coupled micro service architecture.
  - Utilize frameworks that allow services to be independently built without software version or service dependencies.
- Enterprises are increasingly adopting automation by using configuration management tools (such as Puppet, Chef, and Ansible) and DevOps orchestration to increase agility to release software.
- As enterprises develop new and migrate existing applications into public clouds, portability is key to avoiding vendor lock-in. Although virtual machine (VM) technology provides a level of abstraction, each hypervisor implements its environment differently and is therefore not fully portable.

Docker attempts to address these challenges and has therefore emerged as both a leading and compelling technology for virtualizing the infrastructure.

Docker Overview

Containers enable virtualization at the OS level by isolating each application as an OS process. The concept has been around in many forms in operating systems such as BSD with Jails, in Oracle Solaris with Zones, and most recently in Linux with LXC. Docker builds on LXC and has added the "easy button" to enable developers to build, package, and deploy applications across cloud infrastructure without requiring a hypervisor.

The following features differentiate Docker from other container technologies:

- A lightweight abstraction layer (the Docker engine) on top of the OS to manage isolation and networking between applications.
- A documented application programming interface (API) to make Linux-based application deployment simpler.
- The Docker Registry for sharing applications with other users and developers.
- Application portability between Docker-enabled hosts, whether physical, virtual, or cloud-hosted.
- A union file system exposing a common file system to all Docker containers.
- An ecosystem of partner companies providing value-added services and software, enabling Docker to integrate well into a broad variety of development workstyles.

What is a union file system?

A union file system allows each container to provide its own services specific to that container, even if the underlying path and filename collide with an underlying file. For example, one container might need version 2.6 of a Python library whereas another might require a later version. The underlying file system might provide version 2.6, in which case one container is satisfied. However, the container requiring the later version can supply this as part of its container image. This leads to a lower footprint for container images since they need only contain what is strictly necessary for them to run.

Summary of Docker containerization and virtual machine technology

The diagram in Figure 1 illustrates the components used in VM and Docker application deployments. Note that in this example, the VM approach has two guest operating systems to support two applications. By comparison, Docker only requires one host OS to achieve the same application density but, of course, it has a lower overhead to do so.

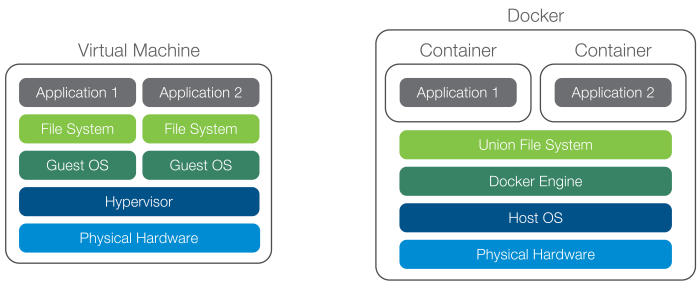


Figure 1: A comparison of virtual machines and Docker containers on a single host.

The following table shows the comparison between VM and Docker capabilities.

	VM	Docker
Application storage overhead	Gigabytes of OS overhead per application.	One common OS for all containers.Small Docker engine overhead (megabytes).
Instantiation	Boot-up time of OS and application.	Application initiation time only.
Resource allocation	Rigid and monolithic. Virtual CPUs are typically allocated to physical CPU cores or hyper threads. Disk space is typically pre-allocated to a VM host.	Flexible. Docker containers can be allocated CPU limits and can share physical host CPU cores very efficiently. Docker memory usage may be limited if desired, but memory that is used can be efficiently allocated among processes on the host and its containers. Disk is shared via the union file system.
Security	Excellent. VMs live in completely separate worlds with little sharing between them unless deliberately permitted by the hosting environment.	Good. The OS kernel prevents containers from accessing each other's memory space. The union file system provides each container a read-only view of the shared container. When a container modifies anything, it is given a container-specific copy of that data, which is seen only by that container.

Docker on virtual machines

As previously mentioned, the primary goal of Docker is to simplify application lifecycle management. While Docker on bare metal is certainly a compelling option, there are benefits to Docker running on hypervisors. These include the ability to snapshot and allow for live migrations of an entire guest—both of which might be key requirements for disaster recovery without losing in-flight state.

Leading infrastructure management and orchestration solutions such as VMware vRealize Suite, OpenStack, and public clouds such as AWS and Azure all support Docker on a given flavor of hypervisor but they expose a common environment to the Docker container, allowing for application portability regardless of environment. This type of heterogeneous deployment model allows customers to start using Docker and gain the benefits of being able to iterate more quickly without having to change the underlying infrastructure.

By moving to a single VM and OS per host, customers can also gain resourcing benefits since the VM do not have to contend for resources. This increase in efficiency is due to the fact that memory and local disk can be allocated to that single OS while the hypervisor no longer must arbitrate between multiple operating systems.

Docker Networking Technical Overview

Networking allows you to create virtual networks in Docker Engine that span multiple hosts. Containers can be attached to these networks wherever they are, providing complete control over the network topology and which containers can talk to which other network elements. Not only that, but the system powering networks can be swapped out with a plug-in, allowing you to integrate with any desired networking system without having to modify the application.

To accommodate high densities of containers on any given host, it is important to understand the mechanism by which each container joins the network. Out of the box, Docker provides each container a private address that is reachable directly only from another container that resides on the same host.

In order for services to reach a container from another host, they must be routed to through Docker's iptables-based Network Address Translation (NAT) function. An example is shown in Figure 2.

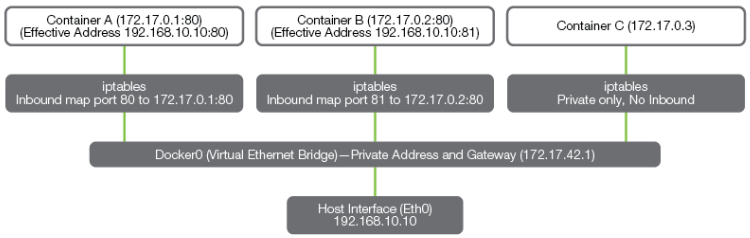


Figure 2: Services are routed through Docker's iptables-based NAT function to reach a container on another host.

The host interface (Eth0) is exposed using another address (in this case, another RFC1918 address, 192.168.10.10). Each Docker container is assigned an address in the 172.x.x/16 space automatically when it is started. In order for a container to communicate to entities outside of its host in a bidirectional fashion, it must be assigned an explicit set of rules through iptables.

In the example shown in Figure 2, the rules have been configured such that the containers may communicate through an IP and port mapping, exposing container A as 192.168.10.10/port 80 and container B as 192.168.10.10/port 81. However, container C can only communicate with the other two containers using the 172.17.0.x addressing.

Docker also supports IPv6 and permits the use of fully routable addresses. This enables containers to communicate with others on different hosts without the need for address mapping. However, this will only work for IPv6, so it may have limited applicability for some environments.

SDN and Docker

Many software-defined data centers use the concept of software-defined networking (SDN) to flexibly deploy their guests. SDN allows isolated network tunnels to be configured for independent tenants on the same physical hardware. It can also be useful to provide tunneled layer 2 inside a cloud data center that would otherwise be fully routed. Docker networking is built around the concept of the Docker Bridge, which may be attached to an Open vSwitch to enable interoperability with technologies such as VXLAN or GRE.

Using Open vSwitch in this manner allows for layer 2 network segregation for multi-tenancy as well as for options to connect to other virtualized environments. For example, it is likely that a data center utilizing Docker will still use virtual machines for key services for which known dedicated resources should be reserved. These might be application delivery services or high performance resources such as databases and processing nodes. These resources may be connected to the network via technologies like VXLAN or GRE so traffic from one tenant is not visible to another.

Scaling applications in this type of environment requires ADC services that can also participate natively in the tunneling protocols. F5 offers multi-tenant VXLAN and GRE capabilities so that functions such as load balancing, SSL offload, firewalling, application security, NAT, and DNS services can be served to clients on the network through a tunnel. Furthermore, F5 provides interoperability between tunnel encapsulation types, including 802.1Q VLANs.

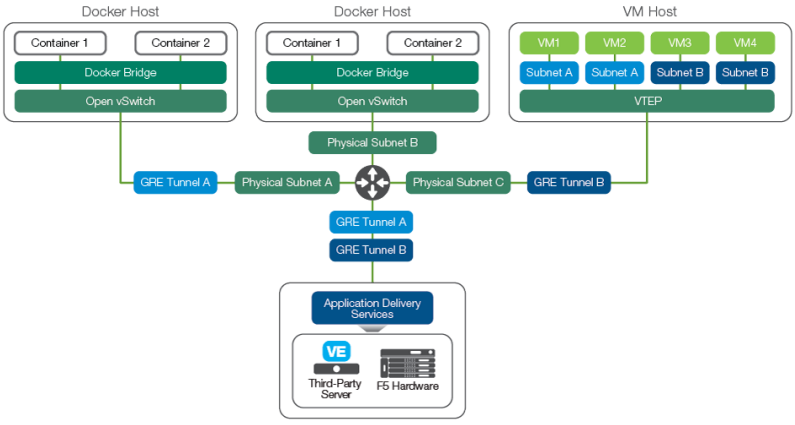


Figure 3: F5 provides interoperability between tunnel encapsulation types, including 802.1Q VLANs.

In the example shown in Figure 3, core application tiers such as a database may be located in a different part of the data center than the resources used to host Docker instances. In such a case, the tenant network might make use of GRE or VXLAN to isolate and join the two otherwise physically distinct subnets.

A BIG-IP solution can be seamlessly inserted into the network at the tenant level by creating a VXLAN tunnel endpoint (VTEP) on the BIG-IP instance. It then becomes part of the tenant network with connectivity to the Docker and virtual machine instances.

Beginning in version 1.7, Docker began to offer some experimental features that extend the base Docker networking capabilities with SDN concepts. The plug-in architecture provides an exciting opportunity to allow F5 network and application delivery services to be inserted for a variety of new use cases, including next-generation firewalling with application-fluent containers, container flow analysis and policy enforcement, and per-flow traffic management.

F5 Direction on Docker Containerization

F5 offers a range of products to enable virtualization. As the ADC market leader with the broadest portfolio of L4–L7 application delivery and security services in the industry, F5 is constantly exploring innovative technologies and extending these technologies across BIG-IP platforms, since they all share a common underlying framework. Figure 4 shows the range of F5's product offerings, from custom hardware to complete cloud-based as-a-service offerings for L4-L7 services. The BIG-IP platform is well positioned to support applications running on Docker containers.

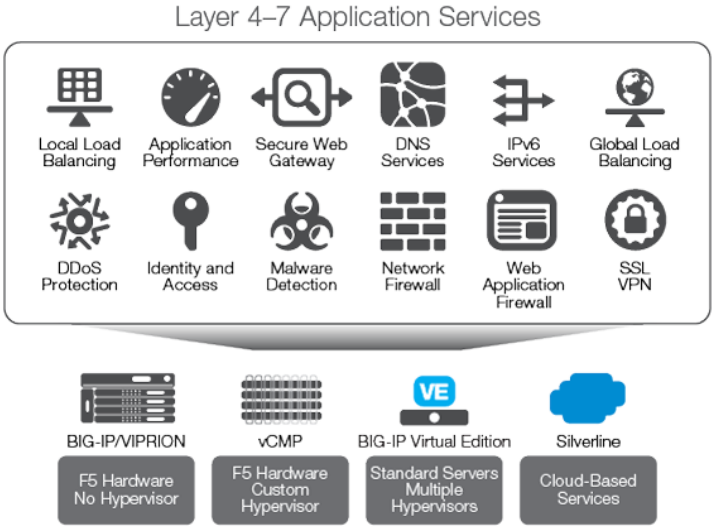


Figure 4: F5 provides a full range of application delivery and security services across a unified platform.

F5 platform evolution with Docker

As mentioned above, F5 recognizes the benefits of Docker in different use cases. However, deployment in containerized infrastructure is still nascent, and multiple options exist for service discovery (such as Consul, etcd, and Mesos-DNS), orchestration, and managing the life cycle of container services (which include Mesos, OpenStack, Kubernetes, Docker, and Cloud Foundry). While networking services are a critical aspect of this ecosystem, it is important to have tight integration with the orchestration environment to ensure seamless deployment. Such integration is also vital to make L4-L7 services available and transparent to users deploying microservices-based applications in a containerized infrastructure. The F5 approach is geared toward providing a consistent experience in services as well as visualization across bare metal, virtualized, or containerized deployments.

The first part of solving the problem is to provide networking services for north-south (N-S) traffic (that is, traffic for the "client-side facing microservices.") Most of the leading orchestration platforms handle deploying, scaling, and exposing of connectivity from containerized services to the outside world. By enabling tight integration with leading container orchestration platforms, F5 ensures that N-S microservices can be automatically discovered by BIG-IP systems, which can then manage traffic for those services. For example, Mesosphere Marathon and Kubernetes provide the option to "label" services. These labels could be used to discover client-side facing services (spin up, tear down, or scaled) and can be automatically added as pool members to the BIG-IP system.

Using the above approach with BIG-IP hardware or virtual editions (VEs) permits centralization of critical functions with hardware acceleration such as:

- SSL offload with centralized certificate management.
- Acceleration such as compression, TCP optimization, and SPDY/HTTP2.
- Sophisticated firewalling with DoS protection.
- Application fluency to thwart application layer attacks.
- Visibility for all incoming and outgoing network connections.
- Much improved debugging through intelligent logging and anomaly detection.

F5 and IPv4 to IPv6 translation with DNS support

F5 solutions provide the ability to scale containerized applications as well as to perform IPv4 to IPv6 and DNS translation between the Docker infrastructure and the external network. To utilize a fully routable Docker container infrastructure, organizations will require not just an efficient IPv4 to IPv6 network function but also support for translating DNS requests. The Docker container infrastructure can operate purely in IPv6, completely isolated from IPv4 and yet, at the same time, maintain a seamless pathway to IPv4 connectivity.

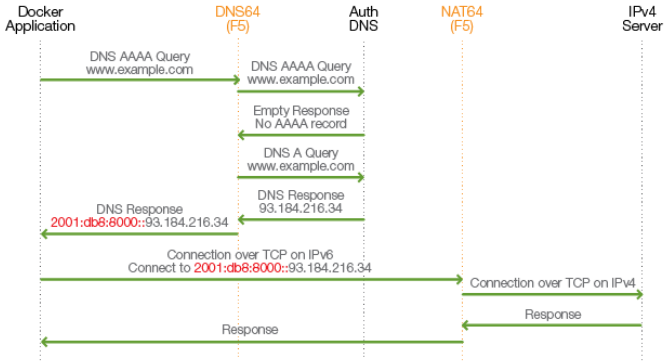


Figure 5: BIG-IP systems perform both DNS64 and NAT64 to allow for IPv6 to IPv4 connectivity.

In the example shown in Figure 5, NAT64 and DNS64 services have been provisioned (again, in any form, physical or virtual). The Docker container attempts a connection to www.example.com, for which, in this example, no IPv6 address exists.

The BIG-IP system is configured to be the DNS resolver for the Docker platform installation. It is configured with an IPv6 address for the DNS resolver itself as well as a special IPv6 prefix address (shown in red) for IPv4 to IPv6 translation.

Once the BIG-IP device has received the IPv6 DNS query, it first performs a recursive operation to see if an IPv6 address is available. However, in this example, the authoritative DNS server for www.example.com responds with an empty record for the AAAA request. The BIG-IP device then performs an IPv4 query, for which it receives a DNS A record. It prepends the special prefix address onto the IPv4 address and sends this back to the Docker client.

The Docker client now has its address resolved and initiates a TCP connection. Because Docker is using the special prefix, the NAT64 function recognizes that IPv6 to IPv4 translation is required.

The NAT64 function creates a binding for the connection between the Docker IPv6 address, the specially prefixed NAT64 address for this IPv4 server, and the IPv4 server. The connection request is sent to the IPv4 server. All responses from that server, which responds via IPv4, are translated by the NAT64 function for connectivity between the Docker container and the IPv4 server.

F5 and east-west traffic flows

The next critical step for creating tight integration is providing services for east-west (E-W) traffic—that is, data passing between microservices that require ADC services. Considering the need to spin up services in seconds and the ephemeral nature of the microservices, the F5 approach is to enable a light-weight ADC. (For advanced services such as authentication or L7, application-level protection, the traffic will be redirected to the BIG-IP instance on the N-S edge.)

The number of containerized services in a microservices architecture will be much higher than in a traditional architecture, with multiple paths of communication between microservices. The likely side effect of this architecture may be complexity, making it harder to troubleshoot performance issues. For example, if an application is showing poor performance, it is important to have end-to-end visibility across different services all the way from the N-S edge. Therefore, a centralized visualization approach that can correlate the traffic patterns between the BIG-IP system in the N-S domain and the light-weight ADC in the E-W domain is extremely critical for troubleshooting.

Instances of F5 BIG-IP solutions can also be inserted between applications to provide load balancing or security services, addressing the security concerns for E-W traffic. For example, a Docker host can be configured to direct traffic from one container to traverse a BIG-IP system for analysis before it enters another container. This can be performed using BIG-IP Application Security Manager™ (ASM), which is application-fluent and can detect whether the container in question is under an attack such as exploitation of a vulnerability.

F5's phased approach to Docker service delivery

Many successful deployments by F5 customers today utilize containerization at massive scale. These organizations span many vertical markets, including financial, telecommunications, and SaaS providers. The role of F5 solutions in these environments currently includes providing ADC service into these environments to ensure fast, secure, and available applications. These services can be integrated to enable self-service for application owners or DevOps—or to orchestrate those services via the container management system.

F5 plans to extend these services into the containerized infrastructure itself to address service discovery, E-W traffic management, and security. F5 also plans to leverage containerization within its product suite, taking advantage of the technology on physical platforms, offering software services in containers, and using the containerized architecture within the F5 Silverline® cloud-base services platform.

The table below provides a glimpse into directions F5 is exploring or actively developing:

Available Today	Near Term	Mid Term	Future
BIG-IP products ensure high availability and scalability of container applications through VIP to an L4 port and IP mapping, with full REST API for orchestration integration. The full range of availability, acceleration, caching, and DNS functions is deployable for Docker environments, along with leading F5 security protection and mitigation capabilities. Additionally, F5 offers plug-ins to allow all BIG-IP form factors to operate in Docker environments utilizing OpenStack.	The F5 Silverline platform will use the elastic compute capability for advanced behavioral DDoS functionality as part of the subscription service. The new networking capabilities of Docker will allow for insertion of new services for advanced E-W traffic profiling, policy enforcement, and security analysis, with traffic inspection and visibility functionality.	F5 is actively exploring new capabilities for F5 vCMP® technology to allow for high VM density and to lay the foundation for vCMP to take advantage of new deployment models, including Docker.	F5 is looking to expand the footprint of an elastic computing manager for customer use, allowing BIG-IP solutions in any format to harness containerized compute functionality for demanding workloads. F5 supports an open container standard (OCS) to enable F5 virtualization services to run across multiple container formats.

Conclusion

Docker presents clear opportunities to improve data center efficiency, whether physical or cloud-based. At the same time, Docker adopters can be more confident that their applications are portable to new environments. Critically, Docker allows application developers to become more agile and deliver applications to market faster. When evolving their DevOps to the Docker model, customers often take the opportunity to introduce new workflows for self-service based around smaller standardized services on which developers can place their applications.

Docker allows for applications to scale rapidly through lightweight container instantiation, and F5 application delivery products fully support such environments. Using F5 BIG-IP solutions, customers can orchestrate the full lifecycle of an application. This can be done through comprehensive REST APIs for critical operations, such as for the creation and maintenance of VIPs, centralized SSL/certificate management, firewall services, and application security with high availability in a multi-tenant architecture.

Docker can be utilized in a variety of models, including public and private cloud deployments. F5 is at the forefront for providing interoperability and support for these environments, offering key functionality that specifically targets OpenStack, VMware, and major cloud providers such as Amazon AWS and Microsoft Azure.

Customers moving to an evolved DevOps model in which Docker is a major component recognize that the operational improvements that can be potentially gained are dependent upon a platform that scales, is secure, highly available, and is as agile as new workflows demand. F5 products and services are designed to work with the broadest set of technologies and technology partners in the industry to deliver on the promise of the Docker vision. F5's commitment to Docker is backed up by solid roadmap investments and continuous product improvement to ensure the success for what will become one of the dominant deployment models for the software-defined data center.

F5 Networks, Inc.
401 Elliott Avenue West, Seattle, WA 98119
888-882-4447 f5.com

Americas
info@f5.com

Asia-Pacific
apacinfo@f5.com

Europe/Middle-East/Africa
emeainfo@f5.com

Japan
f5j-info@f5.com