replify

FLEXIBLE NETWORK ACCELERATION

The Journey from Virtual Machines to Docker with Replify

With release 5.1 of Replify, the Replify Virtual Appliance is now available as a Docker container and fully supported by Replify. In this whitepaper, learn about what Docker means for you and how Replify and Docker could be the alternative for you to the traditional Virtual Machine deployment.

The Business

Over Subscription

The business demands of most Service Providers require the over subscription of many resources. Internet Service Providers strive to oversubscribe internet bandwidth 8:1 in order to compete. Managed Service Providers are continuing to oversubscribe human resources with RMM (Remote Monitoring & Management) tools in order to meet growing user expectations. The Data Centers serving both is experiencing yet another bubble in growth to accommodate higher application density in shrinking real estate. The cost of expansion is capital intensive and doing more with less is what drives the industry while trying to stay profitable.

Traditional Virtualization

In the early 2000s, the time required to rack and stack a single physical server was an order of magnitude greater than anything today's engineer expects. An entire day could be spent downloading the latest operating system, running power and ethernet, just to wait longer for patches and software to be loaded. Today, the expectation is to spin up a Virtual Machine, Instance, or Droplet within one minute. That is incredible! So why isn't this enough?

Modern Containerization

Several notable business goals drive the Data Center manager. Right as the economics of the cloud and shared resources start to yield a profit, the CFO sends out the Quarterly Profit and Loss and Balance Sheet. Sales orders are coming in, the numbers are looking good, and then your engineering team literally hit the wall. There is no more capacity, no more walls to knock down, and a build out will require another round of funding. How can you squeeze more applications into your facilities? Docker!

docker

Governance and Control

The dynamic nature of cloud infrastructure has introduced growing complexity with integrated networks, versioning, and rollback procedures. With Docker, the entire environment (multiple containers, micro services, the network, etc.) can be rolled out with a single command that minimizes human error and provides better business continuity.

User Experience

All of the systems we roll out and the networks we design are necessary to support the applications we build. The application is where a company gets differentiated and functionality has value to the end user. To deploy a container in a standardized way exponentially increases the ability to support and assure the experience of our end users.

Team Efficiency

Development, Quality Assurance, and Operations traditionally have different skillsets which present a challenge for diverse teams to coordinate effectively. When changes to the entire environment can be tracked and rolled out in the lab and production, the same issues are isolated much quicker and communication between these groups are standardized.

79% of those using containers chose Docker as their primary container technology in 2017

Source: Portworx Annual Container Adoption Survey 2017

Orchestration and Scale

Unlike Virtual Machines that have a fixed allocation of resources (like RAM), a Docker container allows caps to be placed on application consumption. As demand increases, the cap can be lifted allowing more sessions or traffic through a docker container versus re-provisioning an entire operating system required with a Virtual Machine.

Infrastructure Consolidation

A physical server that hosts Virtual Machines (Xen, KVM, VMWare, etc.) can increase the application density by 4 to 6 times if using Docker.

The Details: Container Technology

While there are other container technologies like LFC and CoreOS, Docker has surpassed both in functionality, documentation, commercial support if required, and adoption. In the last year, multiple inquiries have been made to Replify regarding Docker support. We have heard the clear choice for a container technology by our customers and now have official support for Replify software deployed with Docker!

Docker The first step to working with Docker is understanding the vocabulary around containers.

Docker Engine The Docker Engine is a lightweight daemon that provides a powerful set of technology to build, ship, and deploy applications in an isolated container.

Docker Image A Docker Image is the resulting file that the Docker Engine can use to launch a container (or multiple containers) and has a known starting point with application defaults and environment details necessary to get up and running.

Docker Container A Docker Container is launched from an image and creates a new read/write layer on start where configuration changes, variable data, and logs can be stored.

Docker Storage Legacy code residing directly on the same system as the Operating System with packages installed are written to the same file system. Docker fundamentally changes how data is stored and persisted to disk by implementing Layers.

Layers With Docker, each step of the build process results in a new read-only layer that is added and isolated from other layers. The base files that make up the operating system, the code installed from packages, and the files copied into the image are all isolated into different layers.

Persistence When a container is started, a read/write layer is added on top of the read/only image layers. Any changes to content in image layers (log files, configuration files, etc.) result in a new layer that is a copy on write. This methodology increases speed and minimizes storage requirements. Compared to the time it takes to clone a Virtual Machine, a new container can be spun up in less than 500ms.

Storage Drivers Docker utilizes storage drivers to talk to the underlying file system. While it ships with a default, it supports all of the storage drivers (OverlayFS, Btrfs, VFS, and ZFS) that ship with supported Linux Distributions. Considering the way Docker interfaces with the host, storage access times are optimal compared to writing through the abstraction layers required in a full Virtual Machine.

Volumes A Docker volume resides on the Docker host as a simple directory and exists outside of the storage driver system. This volume is mapped to the appropriate mount points inside the container and read/writes operate at host speeds! Volumes persist after a container is removed for data retention and later analysis.

Understanding the architecture of Docker is very important to developing a clear picture of how resources are more efficiently utilized. Many layers can exist and provide isolation between the different stages of building an environment for the application. While they do not need to be managed individually, this allows global access by application developers to well-known layers and reuse of common layers. Rather than spending time installing the Operating System, it's details are fetched and built upon. While Developers see great value here, Operations and downstream Customers have reported over a **60% reduction** in Continuous Integration efforts working with their application Vendor!

As we have been working to dockerize our application stack, several features have captured our own attention as having significant value. Many of customers are familiar with our use of Erlang in much of our code base. This provides carrier grade resilience to application faults that can be restarted gracefully and this is mission critical for our partners. In a traditional Virtual Machine, other mechanisms were required for managing the health of the VM and restarting it if there were operating system or dependency issues.

Fault Tolerance

With Docker, a fault can be detected, reported, and automatically restarted to mitigate issues of that nature. Docker comes with the ability to define restart policies and what action to take on subsequent failures. This increases the resilience that ships with our product.

Memory Management

The memory footprint of our stack can be heavily dependent on the number of clients and the cache supporting each. In a VM, there is not an easy way to contain the memory usage of processes in general.

With Docker, a cap is defined and can be adjusted without re-provisioning. This contains the application to the allocated resources and allows for triage to occur at the application level, not the OS level.

Dependency Control

There are many open source libraries that Replify leverages in order to best support user needs. One in particular is OpenSSL. The last couple of years has revealed some significant security issues that required immediate attention.

While this makes sense for the community at large, a more methodical rollout during maintenance windows is expected in carrier environments. The community can rollout new libraries overnight. This has the potential to become service impacting where rollbacks aren't ideal and both teams scramble to patch in new support. With Docker, the libraries on one image is contained from the next.

No more dependency conflicts, no more unexpected incompatibilities, less midnight alerts, higher uptime!

Distribution Support

The Operating System of choice for the Replify team has been Debian for some time. As our customer base has grown, we have expanded our efforts to support our software on FreeBSD, RHEL, CentOS, OpenWRT, Raspbian, and other Linux variants. For a Systems Administrator, the differences may seem trivial.

For a Developer, compiling software with the complexity of our own can be daunting and time consuming. This can be frustrating for new releases, library management, documentation management, etc. Likewise, customers that prefer and officially support one distribution have increase support costs and training to manage our software on Debian. Docker eliminates this altogether.

While yes, the Docker Image uses Debian as a base, the Docker Container can be spun up on a CentOS Docker Host with no differences. Delivering and supporting a Docker Image brings teams on both sides closer together and makes the promise of Continuous Integration much more achievable.

File System Expansion

As a customer grows, we hope they monitor resources like disk space accordingly. We have monitoring information available on our dashboard but the demands of a data center often distract Operations from the most basics aspects of running a healthy environment. It happens.

The disk space allocation of a VM has to be rigidly defined. Using thick provisioning performs better when the cache grows, and thin provisioning can help prevent over allocation of the disk subsystem where one VM does not need to grow to its full capacity immediately.

Replify has addressed this in the past by requiring a second disk of the customers choosing for storing the variable data such as cache. The Docker storage driver does not require fixed disk space and Docker containers can grow as required and attention can be focused on the Docker Host.

Now in a Docker environment, the Replify software can take advantage of the Docker Host disk subsystem with minimal overhead and less maintenance.

Micro Services

When integrated into core service offerings, the Replify software can still benefit from other technologies residing locally. For instance, we see more providers leaning on Public DNS infrastructure from Google, Level 3, Verisign, OpenDNS, and many more. Sure, this has eliminated some maintenance and security obligations at the local ISP, but there is a cost. Users still have some latency to get to these Tier 1 Data Centers and 50ms adds up.

One micro service Replify has chosen to integrate with our Docker Image is dnsmasq. DNS caching is critical to a well-designed Wan Optimization solution. Now it's included!

Other micro services may be of value to you and now there is a framework to capture joint effort in code that we can help deploy.

Disaster Recovery

On a traditional Virtual Machine, any one of many configuration files for basic Operating System functionality could be fat fingered. We've all done it. Open a file in vi and the lack of caffeine invokes a paste where you meant a copy. Before you can back out, the damage is done.

Wouldn't it be easier to just restart the application and restore the one file from a backup versus figuring out why the Virtual Machine won't boot now? Yes it is! Docker's architecture results in an Image that is made up of read-only layers. You can literally restart the container, and optionally purge your changes, to get back up and running in seconds, not minutes.

What this all boils down to is increased efficiency, lower cost of ownership, and simply better support. The goal with our software has been to improve the user experience. This now allows us to improve the admin experience. All parties involved can continue to strive towards meeting the growing expectations of our users regarding new features and support at today's pace.

For OEM Partners

Replify prides itself in its people and the in-house talent to provide OEM partners with a solution to productize as their own. In the Embedded or Reseller markets, there is often a significant expense to talent acquisition and retention. A business often suffers if it can no longer support the Intellectual Property developed during the tenure of their staff. Not only does our application stack have tremendous value, so does the resulting work product of our partners.

To ensure DevOps agility and consistent support across the entire organization (Development, QA, and Operations), creativity and configuration management can be preserved and propagated into the next release and managed as code itself. This leads to a more organized team that spans companies or departments and reduces the knowledge gap to support a product in diverse deployments.

Replify has a long heritage of securing lasting OEM partnerships that have resulted in enhancements to the Replify product that make integrating world-class WAN optimization with a wide range of existing software, on a wide range of devices easier than ever.

For Resellers

Some deployments do not require the complexity of a productized solution when our software fits the requirements. We recognize the value of resellers in understanding where Software Defined Networking and WAN Optimization solutions are required. They are the evangelists of our core technology and have existing relationships with Customers who need a turn-key solution.

It is incredibly common for a reseller to have deep knowledge of the End User and associated applications, yet lack some in-house expertise or time to support our stack on a new Virtual Machine or an unfamiliar distribution of Linux. With Docker, our image can be deployed on existing systems (Windows or Linux) as another application without the depth of training previously required to adopt our technology.

See it in Action

Want to give a Replify Docker container a try? Visit the Replify Ltd repository on DockerHub:

`http://hub.docker.com/u/replifyltd`

As with all Replify software, visit the Replify website and request a trial to get your free trial license key:

`https://www.replify.com`

Get in Touch

If you'd like to discuss becoming an OEM or reseller, or you have a use for Replify for your business, please don't hesitate to contact us at the following email address:

`contact@replify.com`