VSPHERE INTEGRATED CONTAINERS 1.2
Architecture Overview
Table of Contents

vSphere Integrated Containers Overview ......................................................... 4
vSphere Integrated Containers Design Objectives ........................................... 4
vSphere Integrated Containers Architecture .................................................. 5
  vSphere Integrated Containers Components ................................................. 6
  vSphere Integrated Containers Deployment Options .................................... 7
  Virtual Container Hosts .................................................................................. 8
  Docker Container Hosts .................................................................................. 8
The vSphere Integrated Containers Container Runtime ................................... 8
  vSphere Integrated Containers Engine ......................................................... 8
  vSphere Integrated Containers Engine Concepts ....................................... 10
  The Virtual Container Host (VCH) ................................................................. 10
  The Virtual Container Host API End-point .................................................. 12
  The vic-machine Utility ........................................................................... 12
  The Docker Personality ............................................................................... 13
  The ContainerVM, OS and Tether ............................................................... 13
  Docker Container Host .............................................................................. 13
  Virtual Container Hosts vs. Docker Container Hosts .................................. 14
The vSphere Integrated Containers Management Portal and Registry .......... 16
  vSphere Integrations .................................................................................... 17
    Control Plane ............................................................................................. 17
    Compute ...................................................................................................... 18
    Networking .................................................................................................. 18
    Storage ......................................................................................................... 19
Benefits of the containerVM model .............................................................. 20
Security ........................................................................................................... 23
Installation and Configuration ......................................................................... 24
Use Cases ......................................................................................................... 24
  Application Repackaging ........................................................................... 24
  Developer Sandbox ..................................................................................... 25
Summary .......................................................................................................... 25
Learn More ........................................................................................................ 25
vSphere Integrated Containers Overview

VMware vSphere® Integrated Containers is a comprehensive container solution built on VMware’s industry leading virtualization platform – vSphere, which enables customers to run both modern and traditional workloads in production on their existing Software-Defined Data Center (SDDC) infrastructure today with enterprise-grade networking, storage, security, performance and visibility.

vSphere Integrated Containers offers the quickest and easiest way for vSphere customers to start using containers today without additional capital or labor investment. Its tight integration with the entire VMware SDDC environment, as well as its support of leading container technologies such as Docker, makes it a great solution for a seamless transition to container adoption. vSphere Integrated Containers enables enterprises to leverage the benefits of containers today for enhanced developer productivity, business agility, and fast time-to-market.

vSphere Integrated Containers is available to all vSphere 6.0 and above Enterprise Plus customers and vSphere with Operations Management Enterprise Plus customers. There is no additional license subscription required to use vSphere Integrated Containers and support is included for customers with an active vSphere SnS.

This solution guide examines the architecture and capabilities of vSphere Integrated Containers when deployed as an integrated system with VMware vSphere, NSX, vSAN, and various external system interfaces such as: Microsoft Active Directory, Docker Hub, and others.

vSphere Integrated Containers Design Objectives

Before diving into the details of the system architecture, let’s briefly review the system’s design objectives. vSphere Integrated Containers concentrates on addressing the following objectives common across many customers:

- Enable a universal platform for transitioning to modern development practices
- Enable the infrastructure to support the coexistence of both traditional and modern application designs on common, existing hardware and software
- Improve developer agility, shorten time to market, and maximize application resiliency
- Developers need an environment where they can build, test, and run their applications using native container tools with minimal involvement from IT
- Support a standard framework for orchestrating the deployment of cloud native applications and automating management of application availability in operation
- Provide integration with the enterprise-grade capabilities of VMware infrastructure
- Provide security and availability of application when running in production
- Increase visibility into container deployments using standard VMware tools for better operability
- Streamline the development teams’ access to tools and infrastructure resources
- Eliminate extensive approval processes for acquisition and manual provisioning of infrastructure, frequently results in developers pursuing alternative paths of less resistance such as rogue IT or public offerings.

vSphere Integrated Containers Architecture

vSphere Integrated Containers is a product designed to tightly integrate container workflow, lifecycle and provisioning with the vSphere SDDC. It provides a container management portal, an enterprise-class registry, and a container runtime for vSphere fully integrated into a commercial distribution.

Figure 1: vSphere Integrated Containers Open Source Components

With these capabilities, vSphere Integrated Containers enables VMware customers to deliver a production-ready container solution to their developers and DevOps teams. By leveraging their existing SDDC, customers can run container-based applications alongside existing virtual
machine based workloads in production without having to build out a separate, specialized container infrastructure stack.

As an added benefit for customers and partners, vSphere Integrated Containers is modular. So, for example, if your organization already has a container registry in production, you can use that registry with vSphere Integrated Containers Engine and vSphere Integrated Containers Management Portal.

vSphere Integrated Containers Components

vSphere Integrated Containers is built on these major open source products:

1. **vSphere Integrated Containers Engine** is a container runtime for vSphere that enables the provisioning and management of applications packaged as Docker images into vSphere clusters. With the vSphere Integrated Containers Engine Developers can deploy container images alongside traditional workloads on vSphere clusters. The vSphere Integrated Containers engine gives developers the agility and speed they need, while allowing operations to reuse the tools, processes and people they’ve already invested in.

2. **Harbor** is an enterprise-class private container registry that stores and distributes container images. It extends the Docker Distribution open source project by adding the functionalities that enterprises require, such as security, auditing and identity management.

3. **Admiral** is a container management portal. It provides a GUI for DevOps teams to provision and manage containers, and includes the ability to obtain statistics and information about container instances. It provides both Docker compose and a proprietary application definition through templating to combine different containers into an application. It also supports containers scaling in and out. Advanced capabilities, such as approval workflows, are available when integrated with vRealize Automation.

4. **Photon OS** is a minimal Linux container host, optimized to run on VMware platforms. It is used throughout the vSphere Integrated Containers product wherever a Linux guest kernel is required.

The core SDDC infrastructure subsystems, vSphere, NSX, and vSAN complement vSphere Integrated Containers by extending trusted capabilities such as:

- Distributed Resource Scheduling (DRS)
- vMotion
- High-Availability (HA)
- Secure isolation, micro segmentation, and RBAC
- SSO via PSC with extension to external identity sources such as Active Directory/LDAP
- Granular monitoring and logging visibility via vCenter, vRealize Operations, and VRNI
- vSAN / iSCSI / NFS shared storage
- Direct deployment to Distributed vSwitch and NSX Logical Switches, and integration with NSX virtual network infrastructure components
- Unified, full-stack monitoring and logging visibility

vSphere Integrated Containers Deployment Options

vSphere Integrated Containers supports multiple ways to deploy and run containers. Its deep integration with your existing VMware SDDC environment provides the best of both worlds for your developers and IT staff, while supporting a variety of container use cases for the modern enterprise.

Small businesses to large enterprises can leverage the capabilities of vSphere Integrated Containers as it deploys to vSphere infrastructures of various sizes. In the diagram below (Figure 2) we are overlaying typical management and control plane components along with the two container deployment model options over two vSphere clusters of ESXi hosts. Although not illustrated, traditional VMs could also exist on a common cluster with the containers.

![Figure 2: Container Deployment Models](image-url)
Virtual Container Hosts
vSphere Integrated Containers leverages the native constructs of vSphere for provisioning container-based applications. IT admins can deliver a production-ready container solution to their developers and app teams without having to build out a separate, specialized container infrastructure stack. By deploying each container image as a vSphere Virtual Machine (VM), vSphere Integrated Containers allows these container workloads to leverage critical vSphere application security, isolation, availability and performance features – VMware HA, vMotion and Distributed Resource Scheduler. vSphere Integrated Containers provides these features while still presenting a Docker API to developers of container-based applications to consume.

Docker Container Hosts
vSphere Integrated Containers also supports running native Docker container hosts on vSphere. It allows developers to self-provision Docker container hosts for use as a development sandbox, a build server or a swarm cluster. Now you can treat a Docker host as ephemerally as a container.

The vSphere Integrated Containers Container Runtime
Mentioned earlier, vSphere Integrated Containers comprises four major components. Let’s identify each component’s functional significance to the system design and the capabilities it provides.

vSphere Integrated Containers Engine
The vSphere Integrated Containers Engine is a container runtime for vSphere. It enables the provisioning and management of VMs into vSphere clusters using the Docker binary image format. It enables vSphere admins to pre-allocate certain amounts of compute, networking and storage and provide that to developers as a self-service portal exposing a familiar Docker-compatible API. It allows developers who are familiar with Docker to develop in containers and deploy them alongside traditional VM-based workloads on vSphere clusters. VMs provisioned using vSphere Integrated Containers take advantage of many of the benefits of vSphere including DRS, clustering, vMotion, HA, distributed port groups and shared storage.

Using the native constructs of vSphere, IT admins can deliver a production-ready container solution to their developers and app teams without having to build out a separate, specialized container infrastructure stack.

vSphere Integrated Containers engine brings many of the value propositions of containers and container images directly to vSphere infrastructure. vSphere Integrated Containers engine turns container images into objects that look just like containers when viewed from a developer’s perspective and look just like a virtual machine when viewed from an operator’s perspective. Since vSphere Integrated Containers expose the Docker API, it is easy to integrate with developer tools, scripts and processes. And since they behave just like virtual machines, vCenter, NSX, vRealize Operations, vSAN, vMotion and other familiar technologies are just as relevant and valuable for container workloads.

The VMs created by vSphere Integrated Containers engine have all of the characteristics of software containers:

- Ephemeral storage layer with optionally attached persistent “volumes”
- Custom Linux guest - designed to be “just a kernel” - needs “images” to be functional
- Automatically configured to various network topologies

“ContainerVMs” are provisioned into a “Virtual Container Host” which represents a clustered pool of resource, a single-tenant container namespace and an API endpoint. A VCH is not a literal host, rather it is akin to a vSphere resource pool in that it transparently provides clustering, scheduling, vMotion and HA to containers running in it. A VCH is represented in vSphere as a vApp construct.

All of the basic capabilities for creating VMs with these characteristics along with the ability to configure the necessary networking, compute and storage to support them is encapsulated in the “Port Layer” (4.4) service. The Port Layer also adds capabilities to listen for events and interact with the containers.

The scope of vSphere Integrated Containers engine is limited to being an endpoint that runs production container workloads. There is no native support for building images directly on the engine. However, vSphere Integrated Containers can be used by developers to provision native Docker Host VMs to be used for development and build in that manner. The fact that a native Docker Host is controlled from the exact same client as a VCH makes the experience relatively seamless.

vSphere Integrated Containers is optimized for container uptime and availability. Upgrading vSphere Integrated Containers momentarily impacts endpoint availability, but not the containers. Modifying per-tenant compute limits is completely transparent. Upgrading ESXi is also transparently handled with vMotion.
vSphere Integrated Containers Engine Concepts

If we consider a Venn diagram of “What vSphere Does” in one circle and “What Docker Does” in another, the intersection is not insignificant. vSphere Integrated Containers takes as much of vSphere as possible and layers on whatever Docker capabilities are missing. Below are the key concepts and components that make this possible.

The Virtual Container Host (VCH)

A VCH is the virtual functional equivalent of a Linux VM running Docker. From a Docker client point of view, the Virtual Container Host looks very similar to a native Docker host. Hence, there are differences between a native Docker host and a Virtual Container Host (VCH), and between a Linux container and a container VM. Some of those differences are intentional design constraints, such as there being no such thing as a “privileged” container in vSphere Integrated Containers. Some are because of a lack of functional completeness, some are outside of the existing scope of the product - such as native support for docker build.

To use the standard command line tools, simply set DOCKER_HOST to point to your virtual Docker host. Commands like docker run, docker volume and docker net will work similarly as they do with a standard Docker host. However, in the contest of vSphere Integrated Containers, the docker run command creates and provisions a containerVM. Docker network commands allow container workloads to be connected to vSphere networks. Docker volume commands allow for the creation and lifecycle management of disks on vSphere storage.

A VCH is deployed as a vApp in a vCenter Server cluster backed by a resource pool. The vApp provides a useful visual parent-child relationship in the vSphere Client so that you can easily identify the containerVMs that are provisioned into a VCH. The first VM deployed inside the vApp provides a secure Docker compatible API endpoint and other services for the VCH to run.

A VCH is functionally distinct from a traditional container host in the following ways:

- It naturally encapsulates clustering and scheduling by provisioning to vSphere targets
- The resource constraints are dynamically configurable with no impact to the containers
- The containers don’t share a kernel. They could in theory run different kernels
- There is no local image cache. This is kept on a vSphere datastore

The VCH includes all containerVMs instantiated via docker run command and provides networking, storage, clustering, scheduling, vMotion, host evacuation and HA capabilities.

A single ESXi host can have multiple VCHs, each of which with different resources and different users. Similarly, a single VCH can expose the entire capacity of a vSphere cluster of ESXi hosts. It all depends on your own use case and requirements.

The lifecycle of the VCH is managed by a binary called vic-machine which installs, upgrades, deletes and enables debugging for the VCH.

The following diagram illustrates a typical user interaction with the VCH deployment model and container VMs.
The Virtual Container Host API End-point

There is a 1:1 relationship between a VCH and a VCH API end-point. It is built off the same Photon Linux kernel as the containerVMs and is stateless. It has the following functions:

- Run the Core services, Docker personality and image resolution services
- Provide a secure remote Docker API
- Port mapping and routing - When a container port is mapped to a host port, the Virtual Container Host is responsible for listening on that port and routing traffic to the corresponding containerVM
- Manage the lifecycle of containerVMs, image store, volume store and container state
- Provide logging and monitoring of its own services and of its containers

The VCH VM is completely stateless. The state is either on datastores, in VMX guestinfo or in vCenter itself. This makes upgrade a simple process of power down, swap ISO, power up, rediscover.

Security of the VCH VM is an important consideration. It is isolated from the containers, isolated from the vSphere management network and there is no ability to get a remote shell into it without explicit configuration.

The vic-machine Utility

The vic-machine utility is a binary built for Windows, Linux and OSX that manages the lifecycle of VCHs. The vic-machine has been designed to be used by vSphere admins. It takes pre-existing compute, network, storage and a vSphere user as input and creates a VCH as output. It has the following additional functions:

- Creates certificates for Docker client TLS authentication
- Checks that prerequisites have been met on the cluster (firewall, licenses, etc.)
- Assists in configuring ESXi host firewalls
- Configures a running VCH for debugging
- Lists, reconfigure, upgrades/downgrades and deletes VCHs.

The vSphere Integrated Containers machine requires a vSphere admin user for the installation, but takes a separate “proxy” user for client operations. Operations from each VCH can then be audited under the name of the proxy user.

The Docker Personality

vSphere Integrated Containers engine supports version 1.25 of the Docker API, however not all commands and options are implemented. This is because the main target use case for vSphere Integrated Containers Engine is to run applications vs. build applications. The Docker client will report “not implemented” for anything the engine doesn’t support.

The ContainerVM, OS and Tether

As already stated, a containerVM is a VM with all the characteristics of a container. To be clear, the provisioned VM does not contain any OS container abstraction. The VM boots from an ISO containing the Photon Linux kernel and is configured with container images mounted as a disk. Container image layers are represented as a read-only VMDK snapshot hierarchy on VMFS. At the top of this hierarchy is a read-write snapshot that stores ephemeral state. Container volumes are formatted VMDKs attached as disks and indexed on VMFS. Networks are distributed port groups attached as vNICs.

When the VM powers on, it boots from the ISO, chroots into the container filesystem on the attached disk, sets up any internal state such as environment variables and then starts the container process.

Interaction with a running containerVM is managed by a “Tether” init process that runs as PID 1 inside the containerVM. It is responsible for intermediating interaction (streaming stderr, tty etc) between the container and the client. It also manages the lifecycle of container processes and publishes the exit code when it terminates. The tether communicates with the vSphere Integrated Containers appliance via a virtual serial port concentrator.
Docker Container Host
As mentioned earlier, vSphere Integrated Containers Engine also supports running native Docker container hosts on vSphere. It allows developers to self-provision Docker container hosts, and then use native Docker commands to build and run applications inside those Docker hosts.

In the following diagram (Figure 5), we illustrate typical user interaction with the Docker Container Hosts deployment model.

The following table summarizes the differences between the two deployment options:

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>VIRTUAL CONTAINER HOST</th>
<th>DOCKER CONTAINER HOST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Docker Client Tools</td>
<td>Partial Compatibility (optimized for Run stage)</td>
<td>Full Compatibility (Optimized for Build stage)</td>
</tr>
<tr>
<td>Provisioning Process</td>
<td>VI Admin provisions the VCH; Developers provision containers as VMs</td>
<td>VI Admin provisions the VCH; Developers provision Docker Container Hosts and containers inside the DCH</td>
</tr>
<tr>
<td>Speed of Container Deployment</td>
<td>Around 10 secs</td>
<td>Around 2 secs</td>
</tr>
</tbody>
</table>

You can look at the Docker Container Host as a containerVM that delivers a particular use case. Instead of instantiating, as a containerVM, a Docker image that represents an application, you are instantiating, as a containerVM, a Docker image that represents a Docker host.

The vSphere Integrated Containers Management Portal and Registry
Cloud admins and developers can manage and provision container-based applications through the vSphere Integrated Containers management portal. Integrated with VMware Identity Access Management, customers are able to provide local and LDAP-based authentication and authorization to their teams and project-level content trust and notary services for container images in their private registries. Manual and automated container image vulnerability scanning is also included to avoid running images with known vulnerabilities in your data centers.

The management portal provides a UI for DevOps teams to provision and manage containers, including the ability to obtain statistics and information about container instances. Cloud administrators can manage container hosts and apply governance to their usage, including capacity quotas. Administrators can manage identity sources (local and/or external), users and groups, roles, and other credentials.

You can look at the Docker Container Host as a containerVM that delivers a particular use case. Instead of instantiating, as a containerVM, a Docker image that represents an application, you are instantiating, as a containerVM, a Docker image that represents a Docker host.
The vSphere Integrated Containers Management Portal and Registry

Cloud admins and developers can manage and provision container-based applications through the vSphere Integrated Containers management portal. Integrated with VMware Identity Access Management, customers are able to provide local and LDAP-based authentication and authorization to their teams and project-level content trust and notary services for container images in their private registries. Manual and automated container image vulnerability scanning is also included to avoid running images with known vulnerabilities in your data centers.

The management portal provides a UI for DevOps teams to provision and manage containers, including the ability to obtain statistics and information about container instances. Cloud administrators can manage container hosts and apply governance to their usage, including capacity quotas. Administrators can manage identity sources (local and/or external), users and groups, roles, and other credentials.

The Management portal also provides:

- Rule-based resource management, allowing DevOps administrators to set deployment preferences for container placement
- Live state updates that provide a live view of the container system
- Multi-container template management, that enables logical multi-container application deployments
- Basic scale in and scale out of number of containers in a multi-container app template
- Enterprise-class private container registry

The container registry stores and distributes container images. Through the Management Portal DevOps administrators can organize image repositories in projects, and to set up role-based access control to those projects to define which users can access which repositories. The registry also provides rule-based replication of images between registries, implements Docker Content Trust, and provides detailed logging for project and user auditing. It extends the Docker Distribution open source project by adding the functionalities that an enterprise requires, such as security, identity and management.

In addition to user authentication and RBAC, vSphere Integrated Containers Registry enables other security controls, namely:

- Content trust - Image signing and verification to content and version for ensuring security and auditability when running containers in production
- Vulnerability scanning -Traditionally, operating systems have been managed (specifically, patched on a regular basis) by ops personnel with developers providing only the application-level executables. However, containers often use base images like Ubuntu and CentOS from DockerHub, which contain hundreds of features, each of which is susceptible to vulnerabilities. Since container images are essentially opaque to ops personnel, having vulnerability scanning helps IT Ops to prevent exploitation of known vulnerabilities when deploying these applications to production

vSphere Integrations

This section takes a concise deep dive into how vSphere Integrated Containers interacts with vSphere.

Control Plane
The VCH VM acts as a proxy between the Docker client and the vSphere SDK and all of the control plane operations of a VCH are initiated by the vSphere user associated with it. As previously mentioned, the control plane is extended into containerVMs via the Tether process. The majority of control plane operations are VM creation, reconfiguration and deletion.

Given the multi-tenant nature of vSphere, it should be expected that there are multiple VCHs running concurrently in a vSphere cluster and multiple
Docker clients connected to each VCH. Most control plane operations that result in a container state transition are synchronous API calls. The VCH API end-point handles blocking and queuing of concurrent Docker clients. In terms of vSphere sessions, the VCH appliance keeps a single session open and multiple connections are made over that session for control plane operations. The vicadmin web UI opens an additional session as the user currently needs to authenticate to it using vSphere credentials.

**Compute**
Compute is limited at the container level by number of CPUs and memory. This can be set from the Docker client. Compute is limited at the macro level by memory and CPU limits either on the VCH or a Resource Pool it’s deployed into. One difference between vSphere Integrated Containers and regular Docker is that there’s no such thing as an “unlimited” container. A VM necessarily has to have limits. As such, there is a default containerVM configuration associated with a VCH.

**Networking**
vSphere Integrated Containers engine uses pre-configured vSphere port groups for its networking: either regular port groups, distributed port groups or logical switches created by NSX. It is designed to allow different types of traffic to be isolated on distinct networks. It is also designed to allow vSphere networks to be directly exposed to the Docker client for private container traffic. It is the use of distributed port groups that allows for containerVMs to be provisioned across multiple hosts and vMotioned.

Networks must be created ahead of VCH creation and are specified as input to vic-machine, vSphere networks are exposed as “container networks” in Docker. vSphere Integrated Containers does not attempt to create or configure networks in vSphere. It is possible to specify different networks for the following:
- Expose Docker API traffic from Client to VCH
- Container traffic bridged to the VCH appliance
- Public network for image downloading and uploading
- vSphere management traffic
- Exposure of vSphere networks directly to containers

![Figure 7: VCH – vApp](image)

All of these networks default to DHCP, but it is possible to specify IP ranges and gateways if required. Likewise, the appliance itself can be given a static IP address or use DHCP.

Networks created via the Docker client currently use IPAM segregation rather than full micro-segmentation.

Note: that there is no special in-guest networking integration required for vSphere Integrated Containers containers. The container process talks through interfaces directly corresponding to vNICs.

**Storage**
As previously stated, vSphere Integrated Containers uses VMDKs on VMFS for all container storage. It supports any VMFS datastore, including vSAN, iSCSI, or local datastores. And it provides shared storage between containers by using an NFS volume driver.

As input to vSphere Integrated Containers machine, a user can specify different datastores for different types of container state. That’s container ephemeral state, read-only image state and volume state. It is to be expected that different characteristics will be desirable for different kinds of state - for example, a customer is likely to want to back up their volumes, but not their container state.

When images are pulled from a Docker registry, they are extracted onto VMDK snapshots and indexed on a local datastore. Multiple containerVMs can share the same base images because they are immutable and mounted read-only.
Interaction with other vSphere features:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>vMotion</td>
<td>Yes</td>
</tr>
<tr>
<td>Cross Cluster vMotion</td>
<td>No</td>
</tr>
<tr>
<td>Distributed Resource Scheduler (DRS)</td>
<td>Yes</td>
</tr>
<tr>
<td>High Availability (HA)</td>
<td>Yes</td>
</tr>
<tr>
<td>Fault Tolerance (FT)</td>
<td>No</td>
</tr>
<tr>
<td>VMware vSAN</td>
<td>Yes</td>
</tr>
<tr>
<td>Virtual Volumes (VVOL)</td>
<td>No</td>
</tr>
<tr>
<td>Snapshot</td>
<td>No</td>
</tr>
<tr>
<td>Storage DRS</td>
<td>You cannot point to a storage DRS cluster but can consume individual datastores within it</td>
</tr>
</tbody>
</table>

Benefits of the containerVM model

A containerVM is strongly isolated by design and benefits from vSphere enterprise features such as High Availability and vMotion. It is ideally suited to long-running containers or services with the following requirements:

- Strong isolation - a container VM has its own kernel and has no access to a shared filesystem or control plane
- High throughput - a container VM has its own guest buffer cache and can connect directly to a virtual network
- High availability - a container VM can be configured so it can run independent of the availability of the VCH and can benefit from vSphere HA and vMotion
- Persistent data - a container VM can persist its data to a volume disk that can be backed up completely independent of the VM

This means that it is not possible to deploy a container with access to the control plane. It is also impossible to mount parts of the host’s filesystem as shared read-write volumes into the container.

vSphere Integrated Containers containers are slower to start and use more memory resource than Linux containers. ContainerVMs have to be placed, configured and booted. However, in terms of runtime performance, vSphere Integrated Containers containers show improved throughput. Improved throughput is due to not having the additional layer of OS virtualization in the guest.

Running containers in containerVMs makes a lot of sense for long-running services. If the service fails, it should have no impact on any other services. Examples of a long-running service are a database, web server, key-value store etc.

A container VM is less well suited to containers that are transactional and have a very short lifespan, such as running a unit test. This is because the cost to boot the VM is high relative to the time spent running the test. A container VM however is very well suited to longer-running transactional workloads, such as builds. This is because vSphere resource is only consumed for the period of execution and is immediately freed up after. This can lead to a much more efficient use of virtual infrastructure than slave VMs that are up all the time waiting for jobs.

A container VM is also less well suited to containers that need to be weakly isolated by design, for example a logging or monitoring container that need access to the other processes in an application. This is also true of very small containers that together make up a single service or unit of scale. For this purpose, the VM is the ideal isolation domain for the service as a whole and the containers can be deployed inside the VM as software containers using a regular container engine.

vSphere Integrated Containers is a great way to manage regular container hosts, because the container VM abstraction allows you to treat them just as ephemerally as containers.

When deploying applications into production, it’s important to consider where the isolation boundaries should lie for your particular container, service or application. A VM is a natural isolation and failure domain and works well as a unit of scale.
Benefits of DCH over a Docker engine deployed in a VM
Deploying infrastructure to support application development is often cumbersome, error-prone, and time-consuming. As developers rush to build new apps, IT teams waste time with manual configuration, provisioning, and scripting. To improve productivity, they need, at the very least, to streamline the way they roll out and manage infrastructure for developers to use.

Modern developers need an environment where they can build and test their apps using native container technology with minimal involvement from IT. Today, they use their laptops or a VM with a Docker engine in it as the main tools to build containerized applications. However, trying to build an application that goes beyond a simple demo on a laptop or desktop can hit performance and memory constraints. And having developers requesting a VM with the Docker engine in it from IT is time consuming because all the burden of configuration management and network configuration is left to the IT team.

The key solution is providing developers with a secure sandbox so they can serve their own development needs by letting them create native Docker container hosts on demand on vSphere using the Docker CLI they love. By using the Docker container hosts (DCH) feature, developers can deploy Docker container hosts within a vSphere resource pool without having to file a ticket with IT. The create, run, stop, and delete operations are all handled using the native Docker CLI/API.

For example, developers can use DCH to integrate VIC into their CI/CD pipeline and use products like Jenkins to build applications on DCH and then push them to production using VCH. This allows build and test jobs to use vSphere infrastructure as completely ephemeral compute.

The DCH gives developers the Docker tools they need to build modern applications or repackage existing ones and IT teams governance and control over the infrastructure. vSphere administrators provision compute, networking, and storage resources and provide them to developers as a self-service portal that exposes the familiar Docker compatible API.

The DCH provisioned using vSphere Integrated Containers has also a much-reduced attack surface because no extra services besides the Docker daemon are installed and only access to the remote Docker API is provided.

Moreover, the DCH can take advantage of many of the benefits of vSphere, including Distributed Resource Scheduler, clustering, VMware vSphere vMotion®, VMware vSphere High Availability (HA), distributed port groups, and shared storage making it a very robust development infrastructure.

Developers and IT teams need not worry about patching, security, isolation, of the Docker hosts. Those functions are completely automated by how DCHs are deployed as part of VIC.

The outcome is a win-win situation for both developers and administrators: The vSphere administrator gets visibility into and control over the virtual infrastructure, while developers can self-provision Docker container hosts and work with them by using a Docker client.

Security
Security and isolation is one of the biggest differentiators of vSphere Integrated Containers.

Here is a high-level list of security features:

- Docker client authenticates with VCH using a certificate by default
- Network isolation is achieved through multiple port groups
- vSphere Integrated Containers appliance is locked down by default
- VM isolation, every container is fully isolated from the host and from other containers
- Containers are completely isolated from each other and the ESXi hosts
- vSphere Integrated Containers supports authentication with a secure registry
- vSphere Integrated Containers supports strong identity and access management (IAM) with LDAP and Active Directory services
- vSphere Integrated Containers enables admins to control access at the project level, ensuring granular security across teams
- vSphere credentials persisted in ExtraConfig so not visible to the appliance guest
- vSphere Integrated Containers offers an enterprise private container registry with advanced security features such as identity management, LDAP integration, role based access control, trusted content/notary, and vulnerability scanning to ensure the highest security for container images
Installation and Configuration
You install vSphere Integrated Containers by deploying an OVA appliance. The OVA appliance provides access to all of the vSphere Integrated Containers components.

The installation process involves several steps.
1. Download the OVA from VMware web site
2. Deploy the OVA, providing configuration information for vSphere Integrated Containers. The OVA deploys an appliance VM that provides the following services:
   - Runs vSphere Integrated Containers Management Portal and Registry
   - Makes the vSphere Integrated Containers Engine binaries available for download
   - Hosts the vSphere Client plug-in packages for vCenter Server
3. Run the scripts to install the vSphere Client plug-ins on vCenter Server
4. Run the command line utility, vic-machine, to deploy and manage virtual container hosts

Use Cases
A comprehensive container infrastructure solution, vSphere Integrated Containers supports two primary enterprise container use cases. Both use cases are common across enterprises whether they are just starting or advancing through their cloud-native journey.

Application Repackaging
Some organizations prefer to take monolithic applications that traditionally run in VMs and repackage them in containers without refactoring the app. This approach expedites development workflow, simplifies application deployment, and makes app maintenance such as upgrading and patching much easier. vSphere Integrated Containers provides a way to run these containerized workloads in production on vSphere by deploying container images as VMs alongside existing VM workloads. You can now manage these container workloads using existing tooling of your vSphere environment to achieve production-grade availability, security, and visibility.

Developer Sandbox
Developers need an environment in which they can build, test, and run their applications using native container tools with minimal involvement from IT. vSphere Integrated Containers offers on-demand, native Docker container hosts, so developers can use their familiar Docker tools to test and run applications without going through a lengthy IT process. This greatly enhances developer speed and agility, and shortens time to market.

Summary
VMware vSphere Integrated Containers is a comprehensive container solution built on VMware’s industry leading virtualization platform – vSphere. It enables customers to run both modern and traditional workloads in production on their existing SDDC infrastructure today with enterprise-grade networking, storage, security, performance and visibility. It offers the quickest and easiest way for vSphere customers to start using containers today without additional capital or labor investment.

Learn More
Visit vSphere Integrated Containers Product Page:

Download and Install vSphere Integrated Containers today at:

Try vSphere Integrated Containers Hands-On-Labs:

Follow vSphere Integrated Containers product news on our social media channels:
Twitter: https://twitter.com/cloudnativeapps
YouTube: https://www.youtube.com/c/vmwarecloudnativeapps
Slack: https://vmwarecode.slack.com/messages/C293W9V0A