VMware vSphere Integrated Containers for vSphere Administrators

vSphere Integrated Containers

Table of Contents

vSphere Integrated Containers for vSphere Administrators

vSphere Integrated Containers for vSphere Administrators provides information about how to install and configure VMware vSphere Integrated Containers.

Product version 0.3.0

NOTE This book is a work in progress.

Intended Audience

This information is intended for vSphere® Administrators who must manage a vSphere Integrated Containers implementation in their vSphere environment. The information is written for experienced vSphere administrators who are familiar with virtual machine technology and datacenter operations. Knowledge of container technology and Docker is assumed.

VMware, Inc. 3401 Hillview Ave. Palo Alto, CA 94304

www.vmware.com

vSphere Integrated Containers Architecture

vSphere Integrated Containers exists in a vSphere environment, allowing you to use virtual machines like containers. The architecture consists of these components:

- vCenter management tools: monitor and manage virtual machines as well as container virtual machines.
- vCenter Server: manage a single ESXi host or cluster of ESXi hosts with DRS enabled. Specify and deploy datastores and network paths, define clusters, resource pools, and port groups.
- Trusted networks: Deploy and use vSphere Integrated Containers and connections between Docker clients and virtual container hosts.
- Virtual SAN datastores: specify a datastore and top level directory with the name of the virtual container host.
- Docker API appliance virtual machine: The installer deploys this appliance virtual machine, which is also referred to as the virtual container host. You set the docker client to this appliance.
- Docker container virtual machines: Using vSphere Instant Clone and Photon OS technology, you can create and provision multiple container virtual machines directly from a template. The Docker daemon runs outside the container virtual machine. The container is a x86 hardware virtualized virtual machine with a process ID, container interfaces and mounts.

Virtual Container Host

The virtual container host appliance is backed by a Photon OS kernel that provides a virtual container endpoint backed by a vSphere resource pool that allows you to control and consume container services.

You can access a Docker API endpoint for development and map ports for client connections to run containers as required.

vSphere resource management handles container placement within the virtual container host, so that a virtual Docker host can serve as an entire vSphere cluster or a fraction of the same cluster. The only resource consumed by a container host in the cluster is the resource consumed by running containers.

You can reconfigure the virtual container host with no impact to containers running in it. The virtual container host is not limited by the kernel version or by the operating system the containers are running.

You can deploy multiple virtual container hosts in an environment, depending on your business needs, including allocating separate resources for development, testing, and production.

You can also nest virtual container hosts, giving your team access to a large virtual container host, or sub-allocate smaller virtual container hosts for individuals.

Each virtual container host maintains a cache of container images, which you download from either the public Docker Hub or a private registry.

The virtual container host maintains filesystem layers inherent in container images by mapping to discrete VMDK files, all of which are housed in vSphere datastores on VSAN, NFS, or local disks.

You deploy a virtual container host using the CLI installer, then access Virtual Container Host endpoints remotely through a Docker command line interface or other API client.

vSphere Web Client Plugin

You can monitor and manage containers using the vSphere Integrated Containers plugin in the vSphere Web Client.

The plugin allows you to create virtual container hosts, perform administrative tasks on containers such as resource allocation, port mapping, and manage communications between administrators, developers, and application owners during troubleshooting.

You can create, run, stop, and delete containers using standard docker commands in a command line interface and verify these actions in the vSphere Web Client.

Docker Client

Docker clients communicate with the virtual container host, not each container, so you can see aggregated pools of vSphere resources, including storage and memory allocations.

You can pull standard container images from the Docker hub or private registry.

You can create, run, stop, and delete containers using standard docker commands and verify these actions in the vSphere Web Client.

vSphere Integrated Containers Interoperability with Other VMware Software

IT administrators use vCenter Server to view and manage containers. vSphere Integrated Containers work seamlessly with VMware products.

vRealize Suite

The vRealize Suite is available for health monitoring, performance analysis, and compliance across private and public clouds to move businesses faster.

vSphere High Availability, Fault Tolerance, and vSphere vMotion

IT teams can assure service-level agreements for container workloads with VMware vSphere Distributed Resource Scheduler as well as reduce planned and unplanned downtime with VMware vSphere High Availability and VMware vSphere vMotion.

You can apply vSphere High Availability and Fault Tolerance to both the container VMs and the virtual container host, so that containers and the virtual container host can power on or off independently of each other.

You can also restart or upgrade the virtual container host without needing to take the containers offline during the process.

VMware NSX

IT administrators can apply fine-grained network micro-segmentation and policy-based security to cloud-native applications with VMware NSX and also obtain greater visibility into the behavior of containers. VMware NSX integrates containers with the rest of the datacenter, and can be connected to various quarantine, forensics, and monitoring networks for additional monitoring and troubleshooting. VMware NSX is platform agnostic.

VMware Virtual SAN

The virtual container host maintains filesystem layers inherent in container images by mapping to discrete VMDK files, all of which are housed in vSphere datastores on VSAN, NFS, or local disks.

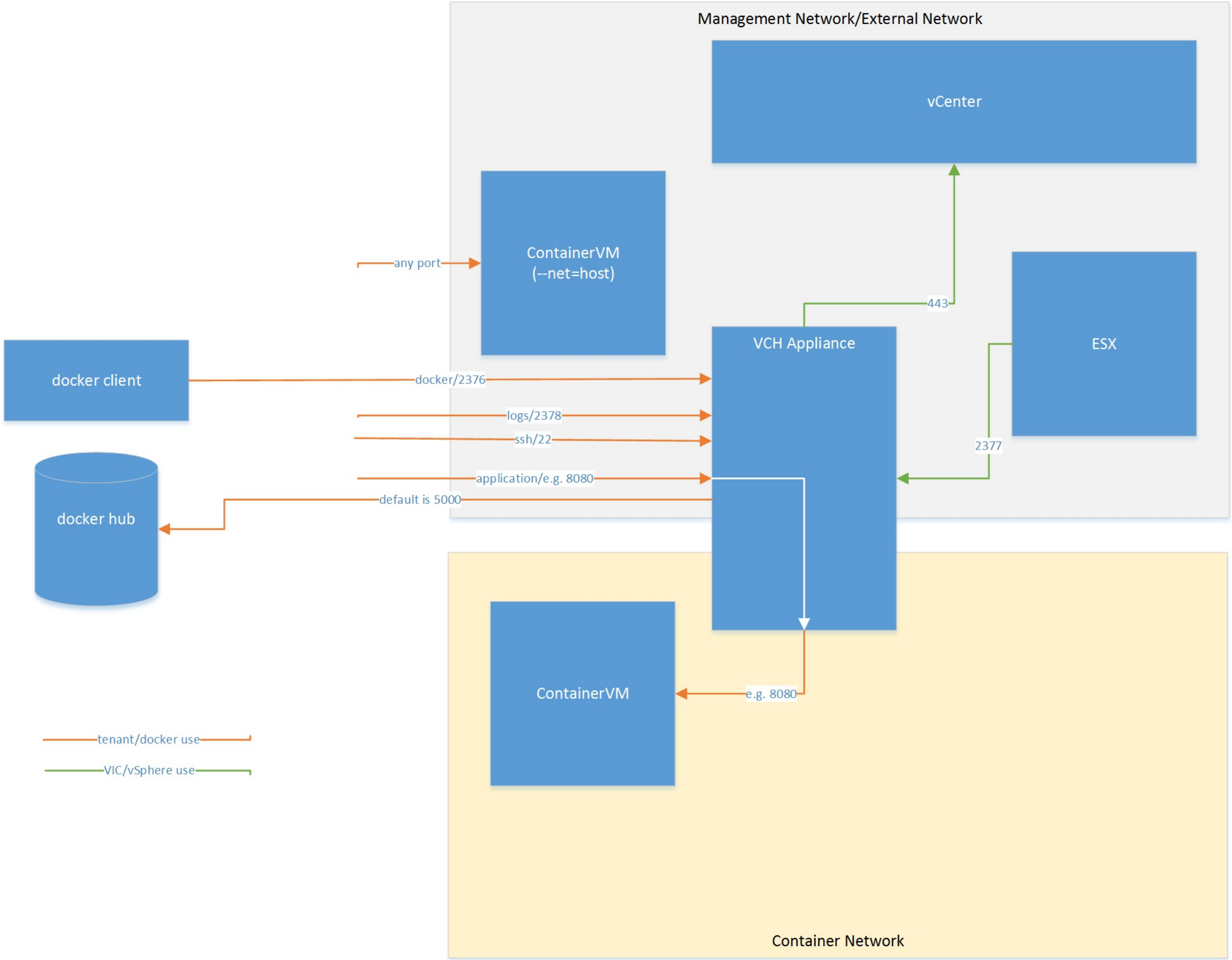
vSphere Instant Clone

vSphere Integrated Containers allows you to create and run multiple containers rapidly with minimal overhead using vSphere 6 Instant Clone technology, which provisions child VMs forked directly from a parent VM template running a Linux kernel. vSphere Integrated Containers creates the kernel and a few supporting resources to run containers using Photon OS technology.

vSphere Integrated Containers Network Overview

vSphere Container Host connects to four network types:

- vSphere Management Network: to communicate with vCenter and ESXi hosts. This network also serves as a tether within the containers to communicate with the vSphere Container Host.
- Docket Management Endpoint Network: to connect to Docker clients and isolate the Docker endpoints from the more public external network.
- External Network: to connect to the internet. Containers can use this external network to publish network services. After defining the external network, you can deploy containers directly on the external interface.
- Container Network: to allow containers to communicate with each other.



The Port Layer

You can configure networks that are tied into the vSphere infrastructure. Pre-configured networks available to a vSphere Container Host are determined by the networks that are part of the provisioning or added when you reconfigure the vSphere Container Host.

The port layer augments vSphere API with low level, platform-specific primitives to allow you to implement a simple container engine:

- Port Layer Execution: Handles container management, such as create, start, and stop.
- Port Layer Interaction: Handles interaction with a running container.

- Port Layer Networking: Handles specific vSphere NSX network mappings into the Docker network namespace as well as mapping existing network entities such as database servers into the Docker container namespace with defined aliases.
- Port Layer Storage: Provides storage manipulation, including container image storage, layering with volume creation and manipulation. imagec, the docker registry client library, uses this component to translate registry images into a layered format that VMDK disk chains can use directly.

Tether Process

The tether process is a minimal agent in the container VM that starts and stops processes and provides monitoring statistics.

Use Cases

These are some use cases of containers using network ports to communicate with each other.

Container with a Published Port

Launch a container and expose a port: `run -p`

Connect the container with the external mapped port on the external surface of the vSphere Container Host.

```
$ docker run -p 8080:80 --name test1 my_container my_app
```

Outcome

You can access Port 80 on test1 from the external network interface on the vSphere Container Host at port 8080.

Simple Bridge Network

Create a new non-default bridge network and set up two containers on the network. Verify that the containers can locate and communicate with each other.

```
$ docker network create -d bridge my-bridge-network
$ docker network ls
...
NETWORK ID          NAME                DRIVER
615d565d498c        my-bridge-network   bridge
...
$ docker run -d --net=my-bridge-network \
                --name=server my_server_image server_app
$ docker run -it --name=client --net=my-bridge-network busybox
/ # ping server
PING server (172.18.0.2): 56 data bytes
64 bytes from 172.18.0.2: seq=0 ttl=64 time=0.073 ms
64 bytes from 172.18.0.2: seq=1 ttl=64 time=0.092 ms
64 bytes from 172.18.0.2: seq=2 ttl=64 time=0.088 ms
```

Outcome

Server and Client can ping each other by name.

Bridged Containers with Exposed Port

Connect two containers on a bridge network and set up one of the containers to publish a port via the vSphere Container Host. Assume server_app binds to port 5000.

```
$ docker network create -d bridge my-bridge-network
$ docker network ls
...
NETWORK ID          NAME                DRIVER
615d565d498c        my-bridge-network   bridge
...
$ docker run -d -p 5000:5000 --net=my-bridge-network \
                --name=server my_server_image server_app
$ docker run -it --name=client --net=my-bridge-network busybox
/ # ping -c 3 server
PING server (172.18.0.2): 56 data bytes
64 bytes from 172.18.0.2: seq=0 ttl=64 time=0.073 ms
64 bytes from 172.18.0.2: seq=1 ttl=64 time=0.092 ms
64 bytes from 172.18.0.2: seq=2 ttl=64 time=0.088 ms
/ # telnet server 5000
GET /

Hello world!Connection closed by foreign host
$ telnet vch_external_interface 5000
Trying 192.168.218.137...
Connected to 192.168.218.137.
Escape character is '^]'.
GET /

Hello world!Connection closed by foreign host.
```

Outcome

Server and Client can ping each other by name. You can connect to the server on port 5000 from the client container and to port 5000 on the vSphere Container Host external interface.

Containers using External Network

Configure two external networks in vSphere: `default-external` is `10.2.0.0/16` with gateway `10.2.0.1` `vic-production` is `208.91.3.0/24` with gateway `208.91.3.1`

Associate a vSphere Container Host, then set up the vSphere Container Host to the default external network.

Attach the vSphere Container Host to the default-external network at 08.91.3.2.

`docker network ls` shows:

```
$ docker network ls
NETWORK ID          NAME                DRIVER
e2113b821ead        none                null
37470ed9992f        default-external    bridge
ea96a6b919de        vic-production      bridge
b7e91524f3e2        bridge              bridge
```

You have a container providing a web service to expose outside of the vSphere Integrated Containers environment.

Output of `docker network inspect default-external`:

```
[
    {
        "Name": "default-external",
        "Id": "37470ed9992f6ab922e155d8e902ca03710574d96ffbfde1b3faf541de2a701f",
        "Scope": "external",
        "Driver": "bridge",
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "10.2.0.0/16",
                    "Gateway": "10.2.0.1"
                }
            ]
        },
        "Containers": {},
        "Options": {}
    }
]
```

Output of `docker network inspect vic-production`:

```
[
    {
        "Name": "vic-production",
        "Id": "ea96a6b919de4ca2bd627bfdf0683ca04e5a2c3360968d3c6445cb18fab6d210",
        "Scope": "external",
        "Driver": "bridge",
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "208.91.3.0/24",
                    "Gateway": "208.91.3.1"
                }
            ]
        },
        "Containers": {},
        "Options": {}
    }
]
```

Set up a server on the vic-production network:

```
$ docker run -d --expose=80 --net=vic-production --name server my_webapp
$ docker inspect --format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' server
208.91.3.2
$ telnet 208.91.3.2 80
Trying 208.91.3.2...
Connected to 208.91.3.2.
Escape character is '^]'.
GET /

Hello world!Connection closed by foreign host.
```

NOTE: You can also use `-p 80` or `-p 80:80` instead of `--expose=80`. If you try to map to different ports with `-p`, you get a configuration error.

Outcome

The server container port is exposed on the external network vic-production.

Remove a Virtual Container Host

In the current builds of vSphere Integrated Containers, you remove virtual container hosts manually.

Removing a virtual container host involves several operations in the vSphere Web Client.

Prerequisites

- You have a virtual container host that you no longer require.
- Connect a vSphere Web Client to the vCenter Server instance on which you deployed the virtual container host.

Procedure

1. In the vSphere Web Client, go to **Hosts and Clusters** and select the cluster or host on which you deployed the virtual container host.
2. Select the resource pool for the virtual container host and click **Related Objects** > **Top Level Objects**.
3. Select the VM for the virtual container host and any container VMs, right-click, and select **Power** > **Power Off**.
4. Select the VM for the virtual container host and the container VMs, right-click, and select **Delete from Disk**.
5. Right-click the resource pool for the virtual container host and select **Delete**.
6. Go to **Storage**, select the datastore that you designated as the image datastore, and delete the folder for the virtual container host.

   If other virtual container hosts on this vCenter Server instance use this datastore, do not delete the `VIC` folder. The `VIC` folder contains container images that are shared between all virtual container hosts.

7. Select the datastore that you designated as the container datastore, and delete the folders for the container VMs that this virtual container host manages.

   In the current builds, container VM files are stored in the image datastore.

8. (Optional) Go to **Hosts and Clusters**, select the host on which you configured a private port group for the virtual container host, and select **Manage** > **Networking** > **Virtual switches**.
9. (Optional) Select the private port group and click the delete button.

   Do not delete the port group if other virtual container hosts on this vCenter Server instance use it.