# VMware vSphere Integrated Containers Engine Installation

vSphere Integrated Containers Engine 0.8

vmware

# Table of Contents

# vSphere Integrated Containers Engine Installation

*vSphere Integrated Containers Engine Installation* provides information about how to install and configure VMware vSphere® Integrated Containers™ Engine.

**Product version: 0.8**

## Intended Audience

This information is intended for anyone who wants to install, configure, and get started with using vSphere Integrated Containers Engine. The information is written for experienced VMware vSphere® administrators who are familiar with virtual machine technology and datacenter operations. Knowledge of container technology and Docker is useful.

## Send Documentation Feedback

Help us to improve the vSphere Integrated Containers documentation.

- Send doc feedback to VMware
- Submit a doc issue in Github

---



**VMware, Inc.** 3401 Hillview Ave. Palo Alto, CA94304

www.vmware.com

# Overview of vSphere Integrated Containers Engine for vSphere Administrators

vSphere Integrated Containers enables IT teams to run traditional and container workloads side-by-side on existing infrastructure seamlessly. With vSphere Integrated Containers Engine, containers are provisioned as virtual machines, offering the same security and functionality of virtual machines in VMware ESXi™ hosts or vCenter Server® instances.

This overview is intended for vSphere Administrators who must use vSphere Integrated Containers to manage container workloads in their vSphere environment.



## Introduction to Containers, Images and Volumes

The word "container" is an overloaded one these days. When understanding containers and how they relate to vSphere Integrated Containers, it is helpful to distinguish the *runtime* aspect from the *packaging* aspect.

### Runtime

At its most basic, a container is simply a sandbox in which a process can run. The sandbox isolates the process from other processes that are running on the same system. A container has a lifecycle which is typically tied to the lifecycle of the process that it is designed to run. If you start a container, it starts its main process and when that process ends, the container stops. The container might have access to some storage. It typically has an identity on a network.

Conceptually, a container represents many of the same capabilities as a VM. The main difference between the two is the abstraction layer:

- A software container is a sandbox within a guest OS and it is up to the guest to provide the container with its dependencies and to enforce isolation. Multiple containers share the guest kernel, networking, and storage. A container does not boot. It is simply a slice of an already-running OS. The OS running the container is called its *host*.
- In contrast, a VM is a sandbox within a hypervisor. It is the hypervisor that provides a VM with its dependencies, such as virtual disks and NICs. A VM has to boot an OS and its lifecycle is typically tied to that of the OS rather than to that of any one process. By design, a VM is strongly isolated from other VMs and its host.

One of the most interesting facets of containers is how they deal with state. Any data that a container writes is non-persistent by default and is lost when that container is deleted. State, however, can persist beyond the lifespan of a container by attaching a *volume* to it or by sending it over a network. Binary dependencies that the container needs,

such as OS libraries or application binaries, are encapsulated in *images* . Images are immutable.

### Packaging

One of the most significant benefits of containers is that they allow you to package up the entire environment that an application needs and run it anywhere. You can go to Docker Hub, select from hundreds of thousands of applications and run that application anywhere that you have installed Docker on a compatible OS. The packaging encapsulates the binary dependencies, environment variables, volumes, and even the network configuration.

The format of this packaging is called an *image*. An image is a template from which many containers can instantiate. The Docker image format allows for images to be composed in a parent-child relationship, just like a disk snapshot. This image hierarchy allows containers to share common dependencies. For example, you might have a Debian 8 image that has a child image with Java installed. That Java image might have a child with Tomcat installed. The Debian 8 image might have other children, such as PHP, Python, and so on.

The immutability of the image format means that you never modify an image, you always create a new one. The layered nature of the image format means that you can cache commonly-used layers so that you only need to download or upload the layers that you do not already have. It also means that if you want to patch a particular image, you create a new image and then rebuild all of its children.

The main advantage of the image format is its portability. As long as you have a destination that is running a container engine, for example Docker, you can download and run an image on it. This portability is facilitated by a *registry*. Aregistry is a service that indexes and stores images. You can run your own private image registry that forms part of a development pipeline. You can *push* images to the registry from development, *pull* them into a test environment for verification, and then *pull* them into a production environment.

## What is vSphere Integrated Containers?

vSphere Integrated Containers comprises the following major components:

- **vSphere Integrated Containers Engine**: Acontainer engine that is designed to integrate of all the packaging and runtime benefits of containers with the enterprise capabilities of your vSphere environment.
- **vSphere Integrated Containers Registry**: ADocker image registry with additional capabilities such as role-based access control (RBAC), replication, and so on.

Both components currently support the Docker image format. vSphere Integrated Containers is entirely Open Source and free to use. Support for vSphere Integrated Containers is included in the vSphere Enterprise Plus license.

vSphere Integrated Containers is designed to solve many of the challenges associated with putting containerized applications into production. It directly uses the clustering, dynamic scheduling, and virtualized infrastructure in vSphere and bypasses the need to maintain discrete Linux VMs as container hosts.

vSphere Integrated Containers Engine allows you, the vSphere administrator, to provide a container management endpoint to a user as a service. At the same time, you remain in complete control over the infrastructure that the container management endpoint service depends on. The main differences between vSphere Integrated Containers Engine and a classic container environment are the following:

- vSphere, not Linux, is the container host:
  - Containers are deployed *as* VMs, not *in* VMs.
  - Every container is fully isolated from the host and from the other containers.
  - vSphere provides per-tenant dynamic resource limits within a vCenter Server cluster
- vSphere, not Linux, is the infrastructure:
  - You can select vSphere networks that appear in the Docker client as container networks.

    - Images, volumes, and container state are provisioned directly to VMFS.
- vSphere is the control plane:
    - Use the Docker client to directly control selected elements of vSphere infrastructure.
    - A container endpoint Service-as-a-Service presents as a service abstraction, not as IaaS.

vSphere Integrated Containers Engine is designed to be the fastest and easiest way to provision any Linux-based workload to vSphere, if that workload can be serialized as a Docker image.

# What Does vSphere Integrated Containers Engine Do?

vSphere Integrated Containers Engine gives you, the vSphere administrator, the tools to easily make your vSphere infrastructure accessible to users so that they can provision container workloads into production.

**Scenario 1: A Classic Container Environment**

In a classic container environment:

- A user raises a ticket and says, "I need Docker".
- You provision a large Linux VM and send them the IP address.
- The user installs Docker, patches the OS, configures in-guest network and storage virtualization, secures the guest, isolates the containers, packages the containers efficiently, and manages upgrades and downtime.

In this scenario, what you have provided is similar to a nested hypervisor that they have to manage and which is opaque to you. If you scale that up to one large Linux VM per tenant, you end up creating a large distributed silo for containers.

**Scenario 2: vSphere Integrated Containers Engine**

With vSphere Integrated Containers Engine:

- A user raises a ticket and says, "I need Docker".
- You identify datastores, networking, and compute on your cluster that the user can use for their Docker environment.
- You use a utility called `vic-machine` to install a small appliance. The appliance represents an authorization to use the infrastructure that you have identified, into which the user can self-provision container workloads.
- The appliance runs a secure remote Docker API, that is the only access that the user has to the vSphere infrastructure.
- Instead of sending your user a Linux VM, you send them the IP address of the appliance, the port of the remote Docker API, and a certificate for secure access.

In this scenario, you have provided the user with a service portal. This is better for the user because they do not have to worry about isolation, patching, security, backup, and so on. It is better for you because every container that the user deploys is a container VM. You can perform vMotion and monitor container VMs just like all of your other VMs.

If the user needs more compute capacity, in Scenario 1, the pragmatic choice is to power down the VM and reconfigure it, or give the user a new VM and let them deal with the clustering implications. Both of these solutions are disruptive to the user. With vSphere Integrated Containers Engine in Scenario 2, you can reconfigure the VCH in vSphere, or redeploy it with a new configuration in a way that is completely transparent to the user.

vSphere Integrated Containers Engine allows you to select and dictate the appropriate infrastructure for the task in hand:

- Networking: You can select multiple port groups for different types of network traffic, ensuring that all of the containers that a user provisions get the appropriate interfaces on the right networks.
- Storage: You can select different vSphere datastores for different types of state. For example, container state is

ephemeral and is unlikely to need to be backed up, but volume state almost certainly should be backed up. vSphere Integrated Containers automatically ensures that state gets written to the appropriate datastore when the user provisions a container.

To summarize, vSphere Integrated Containers Engine gives you a mechanism that allows users to self-provision VMs as containers into your virtual infrastructure.

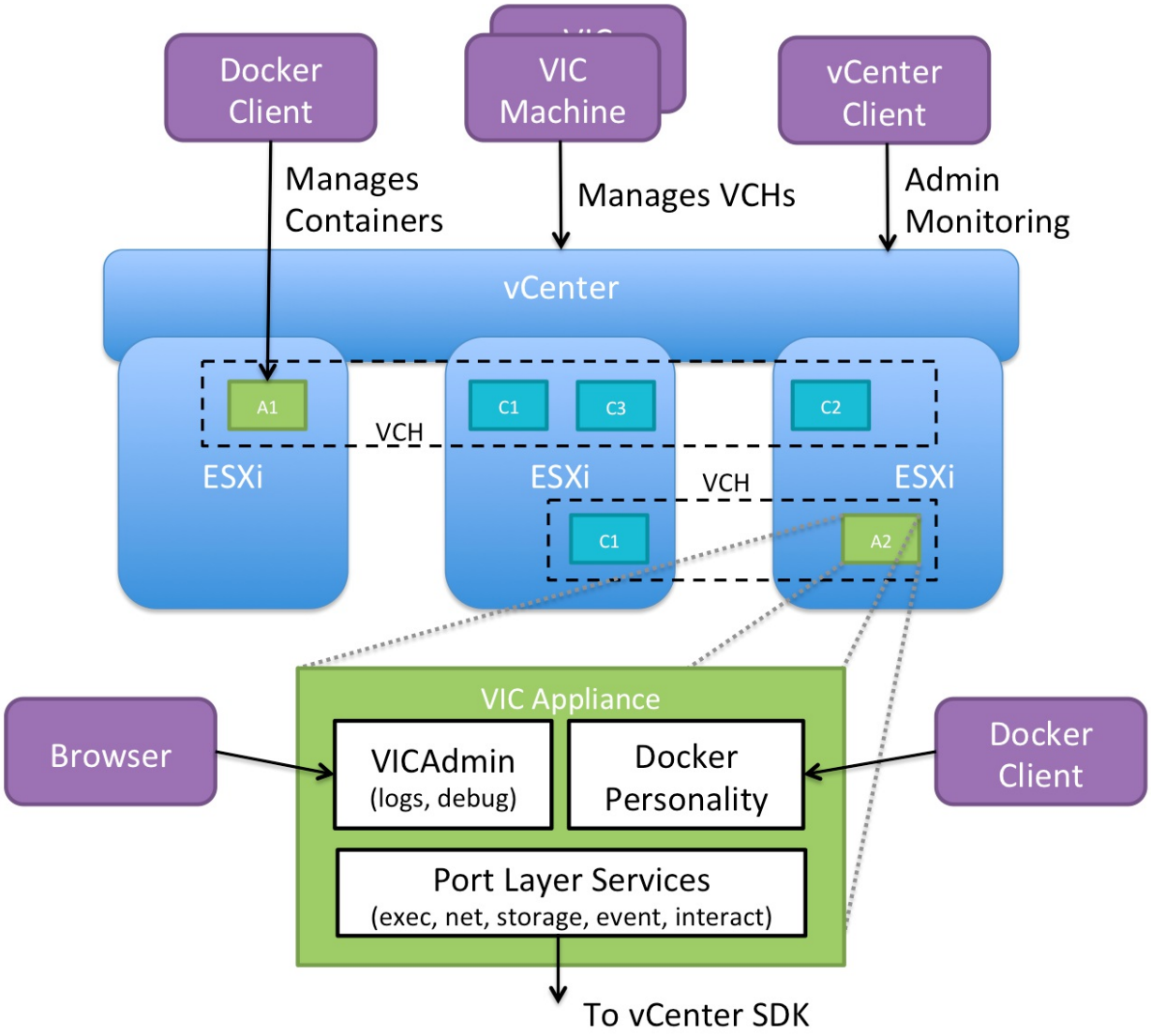
# What Is vSphere Integrated Containers Engine For?

vSphere Integrated Containers Engine currently offers a subset of the Docker API. It is designed to specifically address the provisioning of containers into production, solving many of the problems highlighted in What Does vSphere Integrated Containers Engine Do?.

vSphere Integrated Containers Engine exploits the portability of the Docker image format to present itself as an enterprise deployment target. Developers build containers on one system and push them to a registry. Containers are tested by another system and are approved for production. vSphere Integrated Containers Engine can then pull the containers out of the registry and deploy them to vSphere.

# vSphere Integrated Containers Engine Concepts

If you consider a Venn diagram with "What vSphere Does" in one circle and "What Docker Does" in another, the overlap is significant. The objective of vSphere Integrated Containers Engine is to take as much of vSphere as possible and layer whatever Docker capabilities are missing on top, reusing as much of Docker's own code as possible. The result should not sacrifice the portability of the Docker image format and should be completely transparent to a Docker client. The following sections describe key concepts and components that make this possible.




## Container VMs

The container VMs that vSphere Integrated Containers Engine creates have all of the characteristics of software containers:

- An ephemeral storage layer with optionally attached persistent volumes.
- A custom Linux guest OS that is designed to be "just a kernel" and that needs images to be functional.
- A mechanism for persisting and attaching read-only binary image layers.
- A PID 1 guest agent *tether* extends the control plane into the container VM.
- Various well-defined methods of configuration and state ingress and egress
- Automatically configured to various network topologies.

The provisioned container VM does not contain any OS container abstraction.

- The container VM boots from an ISO that contains the Photon Linux kernel. Note that container VMs do not run the full Photon OS.
- The container VM is configured with a container image that is mounted as a disk.
- Container image layers are represented as a read-only VMDK snapshot hierarchy on a vSphere datastore. At the top of this hierarchy is a read-write snapshot that stores ephemeral state.
- Container volumes are formatted VMDKs that are attached as disks and indexed on a datastore.
- Networks are distributed port groups that are attached as vNICs.

## Virtual Container Hosts

A virtual container host (VCH) is the functional equivalent of a Linux VM that runs Docker, but with some significant benefits. A VCH represents the following elements:

- A clustered pool of resource into which to provision container VMs.
- A single-tenant container namespace.
- An isolated Docker API endpoint.
- Authorization to use and configure pre-approved virtual infrastructure.
- A private network that containers are attached to by default.

If you deploy a VCH in a vCenter Server cluster it spans all of the hosts in the cluster, providing the same flexibility and dynamic use of host resources as is the norm.

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

- It naturally encapsulates clustering and dynamic scheduling by provisioning to vSphere targets.
- The resource constraints are dynamically configurable with no impact on the containers.
- Containers do not share a kernel.
- There is no local image cache. This is kept on a datastore in the cluster that you specify when you deploy a VCH.
- There is no read-write shared storage

A VCH is a multi-functional appliance that you deploy as a vApp in a vCenter Server cluster or as a resource pool on an ESXi host. The vApp or resource pool provides a useful visual parent-child relationship in the vSphere Web Client so that you can easily identify the container VMs that are provisioned into a VCH. You can also specify resource limits on the vApp. You can provision multiple VCHs onto a single ESXi host, into a vSphere resource pool, or into a vCenter Server cluster.

## The VCH Endpoint VM

The VCH endoint VM is the VM that runs inside the VCH vApp or resource pool. There is a 1:1 relationship between a VCH and a VCH endpoint VM. The VCH endpoint VM provides the following functions:

- Runs the services that a VCH requires.
- Provides a secure remote API to a client.
- Receives Docker commands and translates those commands into vSphere API calls and vSphere infrastructure constructs.
- Provides network forwarding so that ports to containers can be opened on the VCH endoint VM and the containers can access a public network.
- Manages the lifecycle of the containers, the image store, the volume store, and the container state
- Provides logging and monitoring of its own services and of its containers.

The lifecycle of the VCH endpoint VM is managed by a utility called `vic-machine`.

## The `vic-machine` Utility

The `vic-machine` utility is a binary for Windows, Linux, and OSX that manages the lifecycle of VCHs. `vic-machine` has been designed for use by vSphere administrators. 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 the prerequisites for VCH deployment are met on the cluster or host, namely that the firewall, licenses, and so on are configured correctly.
- Configures existing VCHs for debugging.

- Lists and deletes VCHs.

# Networks Used by vSphere Integrated Containers Engine

You can configure networks on a virtual container host (VCH) that are tied into the vSphere infrastructure. You define which networks are available to a VCH when you deploy the VCH.

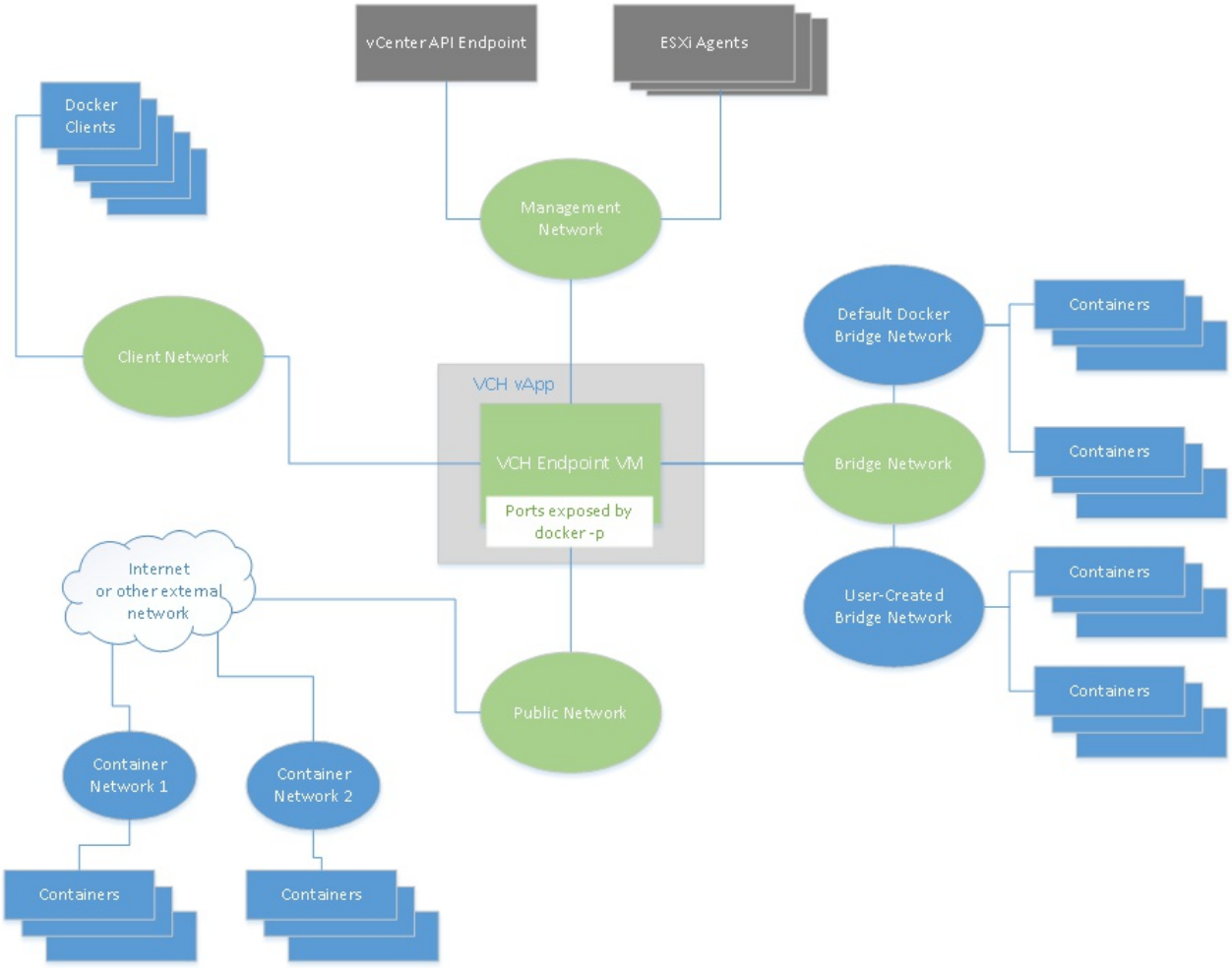
Each network that a VCH uses is a port group on either a vCenter Server instance or ESXi host.

This topic provides an overview of the different network types that virtual container hosts use.



## High-Level View of VCH Networking

The image below shows a high-level view of the networks that a VCH uses and how they connect to your vSphere environment and to the Docker environment that container developers use.



The following sections describe each of the VCH network types, that are represented in the image by green elipses. The blue shapes represent Docker objects, and the gray shapes represent vSphere.

**IMPORTANT**: A VCH supports a maximum of 3 distinct network interfaces. The bridge network requires its own port group, at least two of the public, client, and management networks must share a network interface and therefore a port group. Container networks do not go through the VCH, so they are not subject to this limitation. This limitation will be removed in a future release.

# Management Network

The network for communication between the VCH and vCenter Server and ESXi hosts. The VCH uses this network to provide the `attach` function of the Docker API.

**IMPORTANT**: Because the management network provides access to your vSphere environment, and because container VMs use this network to communicate with the VCH, always use a secure network for the management network. Ideally, use separate networks for the management network and the container networks.

You define the management network by setting the `--management-network` option when you run `vic-machine create`. For more detailed information about management networks, see the section on the `--management-network` option in VCH Deployment Options.

# Public Network

The network that container VMs use to connect to the internet. Ports that containers expose with `docker create -p` when connected to the default bridge network are made available on the public interface of the VCH endpoint VM via network address translation (NAT), so that containers can publish network services.

You define the public network by setting the `--public-network` option when you run `vic-machine create`. For more detailed information about management networks, see the section on the `--public-network` option in VCH Deployment Options.

# Client Network

The network on which the VCH endpoint VM makes the Docker API available to Docker clients. The client network isolates the Docker endpoints from the public network.

You define the Docker management endpoint network by setting the `--client-network` option when you run `vic-machine create`. For more detailed information about Docker management endpoint networks, see the section on the `--client-network` option in VCH Deployment Options.

# Bridge Network

The network or networks that container VMs use to communicate with each other. Each VCH requires a unique bridge network. The bridge network is a port group on a distributed virtual switch.

**IMPORTANT**: Do not use the bridge network for any other VM workloads, or as a bridge for more than one VCH.

You define the bridge networks by setting the `--bridge-network` option when you run `vic-machine create`. For more detailed information about bridge networks, see the section on the `--bridge-network` option in VCH Deployment Options.

Container application developers can also use `docker network create` to create additional bridge networks. These networks are represented by the User-Created Bridge Network in the image above. Additional bridge networks are created by IP address segregation and are not new port groups. You can define a range of IP addresses that additional bridge networks can use by defining the `bridge-network-range` option when you run `vic-machine create`. For more detailed information about how to set bridge network ranges, see the section on the `bridge-network-range` option.

# Container Networks

Container networks allow the vSphere administrator to make vSphere networks directly available to containers. This is done during deployment of a VCH by providing a mapping of the vSphere network name to an alias that is used inside the VCH endpoint VM. The mapped networks are then listed as available by the Docker API. Running `docker network ls` shows these networks, and container developers can attach them to containers in the normal way by using commands such as `docker run` or `create`, with the `--network=_mapped-network-name_` or `docker network connect`. The containers connected to container networks are connected directly to these networks, and traffic does not route though the VCH endpoint VM using NAT.

You can share one network alias between multiple containers. For more detailed information about setting up container networks, see the sections on the `container-network-xxx` options in Virtual Container Host Deployment Options.

# Download vSphere Integrated Containers

You can download different versions of vSphere Integrated Containers, that have different levels of stability and support.

## Official Releases

To obtain the latest official release of vSphere Integrated Containers, go to http://www.vmware.com/go/download-vic. This page includes the donwloads for both vSphere Integrated Containers Engine and vSphere Integrated Containers Registry.

Full support of vSphere Integrated Containers requires the vSphere Enterprise Plus license. To make a support request, contact VMware Global Support.

## Open Source Builds of vSphere Integrated Containers

You can obtain open source builds of vSphere Integrated Containers Engine and vSphere Integrated Containers Registry that have different levels of stability.

**IMPORTANT**: Open source builds are not supported by VMware Global Support. You can obtain community support for open source builds by reporting bugs and creating issues on Github.

- Tagged versions of vSphere Integrated Containers that have been tested and released to the community, but that might not reflect the most up-to-date version of the code:
  - vSphere Integrated Containers Engine: https://github.com/vmware/vic/releases
  - vSphere Integrated Containers Registry (Harbor): https://github.com/vmware/harbor/releases
- Latest built binaries of vSphere Integrated Containers Engine are available at https://bintray.com/vmware/vic-repo/build. Builds usually happen after every successful merge into the source code. These builds have been minimally tested for integration.
- Build the latest source version of vSphere Integrated Containers:
  - vSphere Integrated Containers Engine: https://github.com/vmware/vic/blob/master/README.md#building.
  - vSphere Integrated Containers Registry (Harbor): https://github.com/vmware/harbor/blob/master/README.md

# Contents of the vSphere Integrated Containers Engine Binaries

After you download and unpack a vSphere Integrated Containers Engine binary bundle, you obtain following files:

| File | Description |
|---|---|
| `appliance.iso` | The Photon based boot image for the virtual container host (VCH) endpoint VM |
| `bootstrap.iso` | The Photon based boot image for the container VMs. |
| `ui/` | A folder that contains the files and scripts for the deployment of the vSphere Web Client Plug-in for vSphere Integrated Containers Engine. |
| `vic-machine-darwin` | The OSX command line utility for the installation and management of VCHs. |
| `vic-machine-linux` | The Linux command line utility for the installation and management of VCHs. |
| `vic-machine-windows.exe` | The Windows command line utility for the installation and management of VCHs. |
| `vic-ui-darwin` | The OSX executable for the deployment of the vSphere Web Client Plug-in for vSphere Integrated Containers Engine.<br><br>**NOTE**: Do not run this executable directly.(1) |
| `vic-ui-linux` | The Linux executable for the deployment of the vSphere Web Client Plug-in for vSphere Integrated Containers Engine.<br><br>**NOTE**: Do not run this executable directly.(1) |
| `vic-ui-windows.exe` | The Windows executable for the deployment of the vSphere Web Client Plug-in for vSphere Integrated Containers Engine.<br><br>**NOTE**: Do not run this executable directly.(1) |
| `README` | Contains a link to the vSphere Integrated Containers Engine repository on GitHub. |
| `LICENSE` | The license file for vSphere Integrated Containers Engine |

(1) For information about how to install the vSphere Integrated Containers Engine client plug-in, see Installing the vSphere Web Client Plug-in for vSphere Integrated Containers Engine.

# Deploy Virtual Container Hosts

You use the vSphere Integrated Containers Engine `vic-machine` utility to deploy virtual container hosts.

# Environment Prerequisites for VCH Deployment

Before you install vSphere Integrated Containers Engine, you must ensure that your vSphere infrastructure meets the requirements.



## Supported Platforms for `vic-machine`

The vSphere Integrated Containers Engine installation and management utility, `vic-machine`, has been tested and verified on the following 64-bit Windows, Mac OS, and Linux OS systems.

| Platform | Supported Versions |
|---|---|
| Windows | 7, 10 |
| Mac OS X | 10.11 (El Capitan) |
| Linux | Ubuntu 16.04 LTS |

Other recent 64-bit OS versions should work but are untested.

## Supported vSphere Configurations

You can install vSphere Integrated Containers Engine in the following vSphere setups:

- vCenter Server 6.0 or 6.5, managing a cluster of ESXi 6.0 or 6.5 hosts, with VMware vSphere Distributed Resource Scheduler™ (DRS) enabled.
- vCenter Server 6.0 or 6.5, managing one or more standalone ESXi 6.0 or 6.5 hosts.
- Standalone ESXi 6.0 or 6.5 host that is not managed by a vCenter Server instance.

Caveats and limitations:

- VMware does not support the use of nested ESXi hosts, namely running ESXi in virtual machines. Deploying vSphere Integrated Containers Engine to a nested ESXi host is acceptable for testing purposes only.
- If you deploy a virtual container host (VCH) onto an ESXi host that is not managed by vCenter Server, and you then move that host into a cluster, the VCH might not function correctly.

## License Requirements

vSphere Integrated Containers Engine requires a vSphere Enterprise Plus license.

All of the ESXi hosts in a cluster require an appropriate license. Installation fails if your environment includes one or more ESXi hosts that have inadequate licenses.

# ESXi Host Firewall Requirements

To be valid targets for VCHs and container VMs, ESXi hosts must have the following firewall configuration:

- Allow outbound TCP traffic to port 2377 on the endpoint VM, for use by the interactive container shell.
- Allow inbound HTTPS/TCP traffic on port 443, for uploading to and downloading from datastores.

These requirements apply to standalone ESXi hosts and to ESXi hosts in vCenter Server clusters.

For information about how to open ports on ESXi hosts, see VCH Deployment Fails with Firewall Validation Error.

# ESXi Host Storage Requirements for vCenter Server Clusters

ESXi hosts in vCenter Server clusters must meet the following storage requirements in order to be usable by a VCH:

- Be attached to the datastores that you will use for image stores and volume stores.
- Have access to shared storage to allow VCHs to use more than one host in the cluster.

For information about image stores and volumes stores, see the Datastore Options section of *VCH Deployment Options*.

# General Network Requirements

The following network requirements apply to deployment of VCHs to standalone ESXi hosts and to vCenter Server:

- Use a trusted network for the deployment and use of vSphere Integrated Containers Engine.
- Use a trusted network for the management network.
- Connections between Docker clients and the VCH are encrypted via TLS unless you explicitly disable TLS. The client network does not need to be trusted.
- Each VCH requires an IPv4 address on each of the networks that it is connected to. The bridge network is handled internally, but other interfaces must have a static IP configured on them, or be able to acquire one via DHCP.
- Each VCH requires access to at least one network, for use as the public network. You can share this network between multiple VCHs. The public network does not need to be trusted.
- Do not share the bridge network interface with with any other network, unless you ensure that the bridge IP ranges do not conflict with other VCHs or VMs on that network. It is highly recommended that a bridge network be solely for use by only one VCH.

# vCenter Server Network Requirements

The following network requirements apply to the deployment of VCHs to vCenter Server:

- Create a distributed virtual switch with a port group for each VCH, for use as the bridge network. You can create multiple port groups on the same distributed virtual switch, but each VCH requires its own port group for the bridge network.
- Optionally create port groups for use as mapped container networks.
- All hosts in a cluster must be attached to the port groups that you will use for the VCH bridge network and for any mapped container networks.
- Isolate the bridge network and any mapped container networks. You can isolate networks by using a separate VLAN for each network.

For information about bridge networks and container networks, see the `--bridge-network` and `--container-network` options in *VCH Deployment Options*.

For information about how to create a distributed virtual switch and a port group, see Create a vSphere Distributed Switch in the vSphere documentation.

For information about how to add hosts to a distributed virtual switch, see Add Hosts to a vSphere Distributed Switch in the vSphere documentation.

For information about how to assign a VLAN ID to a port group, see VMware KB 1003825. For more information about private VLAN, see VMware KB 1010691.

# Deploy a VCH to an ESXi Host with No vCenter Server

This topic provides instructions for deploying a virtual container host (VCH) to an ESXi host that is not managed by vCenter Server. This is the most straightforward way to deploy a VCH, and is ideal for testing.

**Prerequisites**

- Download and unpack the vSphere Integrated Containers Engine bundle. For information about where to obtain vSphere Integrated Containers Engine, see Download vSphere Integrated Containers.
- Create or obtain an ESXi host with the following configuration:
  - One datastore
  - The VM Network is present
  - You can use a nested ESXi host for this example
- Verify that the ESXi host meets the requirements in Environment Prerequisites for vSphere Integrated Containers Engine Installation.
- Familiarize yourself with the vSphere Integrated Containers Engine binaries, as described in Contents of the vSphere Integrated Containers Engine Binaries.
- Familiarize yourself with the options of the `vic-machine create` command described in VCH Deployment Options.

**Procedure**

1. Open a terminal on the system on which you downloaded and unpacked the vSphere Integrated Containers Engine binary bundle.
2. Navigate to the directory that contains the `vic-machine` utility:
3. Run the `vic-machine create` command.

   Wrap any option arguments that include spaces or special characters in quotes. Use single quotes if you are using `vic-machine` on a Linux or Mac OS system and double quotes on a Windows system. In these examples, the password is wrapped in quotes because it contains `@`.

   - Linux OS:

     ```
     $ vic-machine-linux create
     --target esxi_host_address
     --user root
     --password 'esxi_host_p@ssword'
     --no-tlsverify
     --force
     ```

   - Windows:

     ```
     $ vic-machine-windows create
     --target esxi_host_address
     --user root
     --password "esxi_host_p@ssword"
     --no-tlsverify
     --force
     ```

   - Mac OS:

```
$ vic-machine-darwin create
--target esxi_host_address
--user root
--password 'esxi_host_p@ssword'
--no-tlsverify
--force
```

The `vic-machine create` command in this example specifies the minimum information required to deploy a VCH to an ESXi host:

- The address of the ESXi host on which to deploy the VCH, in the `--target` option.
- The ESXi host `root` user and password in the `--user` and `--password` options.
- Disables the verification of clients that connect to this VCH by specifying the `--no-tlsverify` option.
- Disables the verification of the ESXi host certificate by specifying the `--force` option.

Because the ESXi host only has only one datastore and uses the VM Network network, `vic-machine create` automatically detects and uses those resources.

When deploying to an ESXi host, `vic-machine create` creates a standard virtual switch and a port group for use as the container bridge network, so you do not need to specify any network options if you do not have specific network requirements.

This example deploys a VCH with the default name `virtual-container-host`.

**Result**

At the end of a successful installation, `vic-machine` displays information about the new VCH:

```
Initialization of appliance successful
VCH Admin Portal:
https://vch_address:2378
Published ports can be reached at:
vch_address
Docker environment variables:
DOCKER_HOST=vch_address:2376
Environment saved in virtual-container-host/virtual-container-host.env
Connect to docker:
docker -H vch_address:2376 --tls info
Installer completed successfully
```

**What to Do Next**

To test your VCH, see Verify the Deployment of a VCH.

For examples of commands to deploy a VCH in various other vSphere configurations, see Advanced Examples of Deploying a VCH.

# Deploy a VCH to a Basic vCenter Server Cluster

This topic provides instructions for deploying a virtual container host (VCH) in a very basic vCenter Server environment. This basic deployment allows you to test vSphere Integrated Containers Engine with vCenter Server before attempting a more complex deployment that corresponds to your real vSphere environment.

The vCenter Server instance to which you deploy the VCH must match the specifications listed in the prerequisites.

**Prerequisites**

- Download and unpack the vSphere Integrated Containers Engine bundle. For information about where to obtain vSphere Integrated Containers Engine, see Download vSphere Integrated Containers.
- Create or obtain a vCenter Server instance with the following configuration:
  - One datacenter
  - One cluster with two ESXi hosts and DRS enabled. You can use nested ESXi hosts for this example.
  - A shared datastore, that is accessible by both of the ESXi hosts.
  - The VM Network is present
  - One distributed virtual switch with one port group named `vic-bridge`
- Verify that your vCenter Server instance and both of the ESXi hosts in the cluster meet the requirements in Environment Prerequisites for vSphere Integrated Containers Engine Installation.
- Familiarize yourself with the vSphere Integrated Containers Engine binaries, as described in Contents of the vSphere Integrated Containers Engine Binaries.
- Familiarize yourself with the options of the `vic-machine create` command described in VCH Deployment Options.

**Procedure**

1. Open a terminal on the system on which you downloaded and unpacked the vSphere Integrated Containers Engine binary bundle.
2. Navigate to the directory that contains the `vic-machine` utility:
3. Run the `vic-machine create` command.

   Wrap any option arguments that include spaces or special characters in quotes. Use single quotes if you are using `vic-machine` on a Linux or Mac OS system and double quotes on a Windows system. In these examples, the user name is wrapped in quotes because it contains `@`.

   - Linux OS:

     ```
     $ vic-machine-linux create
     --target vcenter_server_address
     --user 'Administrator@vsphere.local'
     --password vcenter_server_password
     --bridge-network vic-bridge
     --image-store shared_datastore_name
     --no-tlsverify
     --force
     ```

   - Windows:

```
$ vic-machine-windows create
--target vcenter_server_address
--user "Administrator@vsphere.local"
--password vcenter_server_password
--bridge-network vic-bridge
--image-store shared_datastore_name
--no-tlsverify
--force
```

- Mac OS:

```
$ vic-machine-darwin create
--target vcenter_server_address
--user 'Administrator@vsphere.local'
--password vcenter_server_password
--bridge-network vic-bridge
--image-store shared_datastore_name
--no-tlsverify
--force
```

The `vic-machine create` command in this example specifies the minimum information required to deploy a VCH to vCenter Server:

- The address of the vCenter Server instance on which to deploy the VCH, in the `--target` option.
- The vCenter Single Sign-On user and password in the `--user` and `--password` options.
- The port group named `vic-bridge`, for use as the container bridge network.
- The name of the shared datastore to use as the image store, in which to store container images.
- Disables the verification of clients that connect to this VCH by specifying the `--no-tlsverify` option.
- Disables the verification of the vCenter Server certificate by specifying the `--force` option.

Because the vCenter Server instance only has one datacenter and one cluster, and uses the VM Network network, `vic-machine create` automatically detects and uses these resources.

This example deploys a VCH with the default name `virtual-container-host`.

**Result**

At the end of a successful installation, `vic-machine` displays information about the new VCH:

```
Initialization of appliance successful
VCH Admin Portal:
https://vch_address:2378
Published ports can be reached at:
vch_address
Docker environment variables:
DOCKER_HOST=vch_address:2376
Environment saved in virtual-container-host/virtual-container-host.env
Connect to docker:
docker -H vch_address:2376 --tls info
Installer completed successfully
```

**What to Do Next**

To test your VCH, see Verify the Deployment of a VCH.

For examples of commands to deploy a VCH in various other vSphere configurations, see Advanced Examples of Deploying a VCH.

# Verify the Deployment of a VCH

After you have deployed a virtual container host (VCH), you can verify the deployment by connecting a Docker client to the VCH and running Docker operations. You can check the results in the vSphere Client or vSphere Web Client.

**IMPORTANT**: Do not use the vSphere Client or vSphere Web Client to perform operations on VCH appliances or container VMs. Specifically, using the vSphere Client or vSphere Web Client to power off, power on, or delete VCH appliances or container VMs can cause vSphere Integrated Containers Engine to not function correctly. Always use `vic-machine` to perform operations on VCHs. Always use Docker commands to perform operations on containers.

**Prerequisites**

- You followed the instructions in Deploy a VCH to an ESXi Host with No vCenter Server or Deploy a VCH to a Basic vCenter Server Cluster to deploy a VCH to either an ESXi host or to a vCenter Server instance.
- You ran `vic-machine create` with the `--no-tlsverify` option.
- You have installed a Docker client.
- If you deployed the VCH to vCenter Server, connect a vSphere Web Client to that vCenter Server instance.
- If you deployed the VCH to an ESXi host, connect a vSphere Client to that host.

**Procedure**

1. View the VCH appliance in the vSphere Web Client or vSphere Client.
   - vCenter Server: Go to **Hosts and Clusters** in the vSphere Web Client and select the cluster or host on which you deployed the VCH. You should see a vApp with the name that you set for the VCH.
   - ESXi host: Go to **Inventory** in the vSphere Client and select the host on which you deployed the VCH. You should see a resource pool with the name that you set for the VCH.

   The vApp or resource pool contains the VCH endpoint VM.
2. Run the `docker info` command to confirm that you can connect to the VCH.

   ```
   docker -H vch_address:2376 --tls info
   ```

   You should see confirmation that the Storage Driver is `vSphere Integrated Containers Backend Engine`. If the connection fails with a Docker API version error, see Docker Commands Fail with a Docker API Version Error.
3. Pull a Docker container image into the VCH, for example, the `BusyBox` container.

   ```
   docker -H vch_address:2376 --tls pull busybox
   ```

4. View the container image files in the vSphere Web Client or vSphere Client.
   - vCenter Server: Go to **Storage**, select the datastore that you designated as the image store, and click **Manage** > **Files**.
   - ESXi host: Click the **Summary** tab for the ESXi host, right-click the datastore that you designated as the image store, and select **Browse Datastore**.

     vSphere Integrated Containers Engine creates a folder a folder that has the same name as the VCH, that contains a folder named `VIC` in which to store container image files.
5. Expand the `VIC` folder to navigate to the `images` folder. The `images` folder contains a folder for every container image that you pull into the VCH. The folders contain the container image files.
6. In your Docker client, run the Docker container that you pulled into the VCH.

```
docker -H vch_address:2376 --tls run --name test busybox
```

7. View the container VMs in the vSphere Web Client or vSphere Client.
   - vCenter Server: Go to **Hosts and Clusters** and expand the VCH vApp.
   - ESXi host: Go to **Inventory** and expand the VCH resource pool.

   You should see a VM for every container that you run, including a VM named `test-container_id`.
8. View the container VM files in the vSphere Web Client or vSphere Client.
   - vCenter Server: Go to **Storage** and select the datastore that you designated as the image store.
   - ESXi host: Click the **Summary** tab for the ESXi host, right-click the datastore that you designated as the image store, and select **Browse Datastore**.

   At the top-level of the datastore, you should see a folder for every container that you run. The folders contain the container VM files.

# VCH Deployment Options

The command line utility for vSphere Integrated Containers Engine, `vic-machine`, provides a `create` command with options that allow you to customize the deployment of virtual container hosts (VCHs) to correspond to your vSphere environment.

- vSphere Target Options
- Security Options
- Private Registry Options
- Datastore Options
- Networking Options
- Additional Deployment Options

To allow you to fine-tune the deployment of VCHs, `vic-machine create` provides Advanced Options.

- Advanced Security Options
- Options for Specifying a Static IP Address for the VCH Endpoint VM
- Options for Configuring a Non-DHCP Network for Container Traffic
- Options to Configure VCHs to Use Proxy Servers
- Advanced Resource Management Options
- Other Advanced Options

# vSphere Target Options

The `create` command of the `vic-machine` utility requires you to provide information about where in your vSphere environment to deploy the VCH and the vCenter Server or ESXi user account to use.

### `--target`

Short name: `-t`

The IPv4 address, fully qualified domain name (FQDN), or URL of the ESXi host or vCenter Server instance on which you are deploying a VCH. This option is always **mandatory**.

To facilitate IP address changes in your infrastructure, provide an FQDN whenever possible, rather than an IP address.

- If the target ESXi host is not managed by vCenter Server, provide the address of the ESXi host.

  ```
  --target esxi_host_address
  ```

- If the target ESXi host is managed by vCenter Server, or if you are deploying to a cluster, provide the address of vCenter Server.

  ```
  --target vcenter_server_address
  ```

- You can include the user name and password in the target URL.

  ```
  --target vcenter_or_esxi_username:password@vcenter_or_esxi_address
  ```

Wrap the user name or password in single quotes (Linux or Mac OS) or double quotes (Windows) if they include special characters.

```
'vcenter_or_esxi_usern@me':'p@ssword'@vcenter_or_esxi_address
```

If you do not include the user name in the target URL, you must specify the `user` option. If you do not specify the `password` option or include the password in the target URL, `vic-machine create` prompts you to enter the password.

- If you are deploying a VCH on a vCenter Server instance that includes more than one datacenter, include the datacenter name in the target URL. If you include an invalid datacenter name, `vic-machine create` fails and suggests the available datacenters that you can specify.

  ```
  --target vcenter_server_address/datacenter_name
  ```

  Wrap the datacenter name in single quotes (') on Mac OS and Linux and in double quotes (") on Windows if it includes spaces.

  ```
  --target vcenter_server_address/'datacenter name'
  ```

### `--user`

Short name: `-u`

The username for the ESXi host or vCenter Server instance on which you are deploying a VCH.

If you are deploying a VCH on vCenter Server, specify a username for an account that has the Administrator role on that vCenter Server instance.

```
--user esxi_or_vcenter_server_username
```

Wrap the user name in single quotes (') on Mac OS and Linux and in double quotes (") on Windows if it includes special characters.

```
--user 'esxi_or_vcenter_server_usern@me'
```

You can also specify the username in the URL that you pass to `vic-machine create` in the `target` option, in which case the `user` option is not required.

### `--password`

Short name: `-p`

The password for the user account on the vCenter Server on which you are deploying the VCH, or the password for the ESXi host if you are deploying directly to an ESXi host. If not specified, `vic-machine` prompts you to enter the password during deployment.

```
--password esxi_host_or_vcenter_server_password
```

Wrap the password in single quotes (') on Mac OS and Linux and in double quotes (") on Windows if it includes special characters.

```
--password 'esxi_host_or_vcenter_server_p@ssword'
```

You can also specify the username and password in the URL that you pass to `vic-machine create` in the `target` option, in which case the `password` option is not required.

## `--compute-resource`

Short name: `-r`

The relative path to the host, cluster, or resource pool in which to deploy the VCH.

If the vCenter Server instance on which you are deploying a VCH only includes a single instance of a standalone host or a cluster, `vic-machine create` automatically detects and uses those resources. In this case, you do not need to specify a compute resource when you run `vic-machine create`. If you are deploying to an ESXi host and you do not specify `--compute-resource`, `vic-machine create` automatically uses the default resource pool.

You specify the `--compute-resource` option in the following circumstances:

- A vCenter Server instance includes multiple instances of standalone hosts or clusters, or a mixture of standalone hosts and clusters.
- You want to deploy the VCH to a specific resource pool in your environment.

If you do not specify the `--compute-resource` option and multiple possible resources exist, or if you specify an invalid resource name, `vic-machine create` fails and suggests valid targets for `--compute-resource` in the failure message.

- To deploy to a specific resource pool on an ESXi host, specify the name of the resource pool:

  ```
  --compute-resource  resource_pool_name
  ```

- To deploy to a vCenter Server instance that has more than one standalone host that are not part of a cluster, specify the IPv4 address or fully qualified domain name (FQDN) of the target host:

  ```
  --compute-resource host_address
  ```

- To deploy to a vCenter Server with more than one cluster, specify the name of the target cluster:

  ```
  --compute-resource cluster_name
  ```

- To deploy to a specific resource pool on a standalone host that is managed by vCenter Server, specify the IPv4 address or FQDN of the target host and name of the resource pool:

  ```
  --compute-resource host_name/resource_pool_name
  ```

- To deploy to a specific resource pool in a cluster, specify the names of the target cluster and the resource pool:

  ```
  --compute-resource cluster_name/resource_pool_name
  ```

- Wrap the resource names in single quotes (') on Mac OS and Linux and in double quotes (") on Windows if they include spaces:

  ```
  --compute-resource 'cluster name'/'resource pool name'
  ```

## `--thumbprint`

Short name: None

The thumbprint of the vCenter Server or ESXi host certificate. Specify this option if your vSphere environment uses untrusted, self-signed certificates. If your vSphere environment uses trusted certificates that are signed by a known Certificate Authority (CA), you do not need to specify the `--thumbprint` option.

**NOTE** If your vSphere environment uses untrusted, self-signed certificates, you can run `vic-machine create` without the `--thumbprint` option by using the `--force` option. However, running `vic-machine create` with the `--force` option rather than providing the certificate thumbprint is not recommended, because it permits man-in-the-middle attacks to go undetected.

To obtain the thumbprint of the vCenter Server or ESXi host certificate, run `vic-machine create` without the specifying the `--thumbprint` or `--force` options. The deployment of the VCH fails, but the resulting error message includes the required certificate thumbprint. You can copy the thumbprint from the error message and run vic-machine create again, including the `--thumbprint` option. If you obtain the thumbprint by other means, use upper-case letters and colon delimitation rather than space delimitation when you specify `--thumbprint`.

```
--thumbprint certificate_thumbprint
```

# Security Options

As a convenience, `vic-machine create` provides the option of generating a client certificate, server certificate, and certificate authority (CA) as appropriate when you deploy a VCH. The generated certificates are functional, but they do not allow for fine control over aspects such as expiration, intermediate certificate authorities, and so on. To exercise fine control over the certificates used, obtain or generate custom certificates yourself before deploying a VCH. Use the `--key`, `--cert`, and `--tls-ca` options to pass the custom certificates to `vic-machine create`.

## Restrict access to the Docker API

vSphere Integrated Containers Engine authenticates Docker API clients by using client certificates. This configuration is commonly referred to as `tlsverify` in documentation about containers and Docker. A client certificate is accepted if it is signed by a CA that you provide by specifying one or more instances of the `--tls-ca` option. In the case of the certificates that `vic-machine create` generates, `vic-machine create` creates a CA and uses it to create and sign a single client certificate.

When using the Docker client, the client validates the server either by using CAs that are present in the root certificate bundle of the client system, or that are provided explicitly by using the `--tlscacert` option when running Docker commands. As a part of this validation, the server certificate must explicitly state at least one of the following, and must match the name or address that the client uses to access the server:

- The FQDN used to communicate with the server
- The IP address used to communicate with the server
- A wildcard domain that matches all of the FQDNs in a specific subdomain. For an example of a domain wildcard, see https://en.wikipedia.org/wiki/Wildcard_certificate#Example.

### `--tls-cname`

Short name: None

The FQDN or IP address to embed in the generated server certificate. Specify an FQDN, IP address, or a domain wildcard. If you provide a custom server certificate by using the `--cert` option, you can use `--tls-cname` as a sanity check to ensure that the certificate is valid for the deployment.

If you do not specify `--tls-cname` but you do set a static address for the VCH on the client network interface, `vic-machine create` uses that address for the Common Name, with the same results as if you had specified `--tls-cname=x.x.x.x`. For information about setting a static IP address on the client network, see Options for Specifying a Static IP Address for the VCH Endpoint VM.

When you specify the `--tls-cname` option, `vic-machine create` performs the following actions during the deployment of the VCH:

- Checks for an existing certificate in either a folder that has the same name as the VCH that you are deploying, or in a location that you specify in the `--cert-path` option. If a valid certificate exists that includes the same Common Name attribute as the one that you specify in `--tls-cname`, `vic-machine create` reuses it. Reusing certificates allows you to delete and recreate VCHs for which you have already distributed the certificates to container developers.
- If certificates are present in the certificate folder that include a different Common Name attribute to the one that you specify in `--tls-cname`, `vic-machine create` fails.
- If a certificate folder does not exist, `vic-machine create` creates a folder with the same name as the VCH, or creates a folder in the location that you specify in the `--cert-path` option.
- If valid certificates do not already exist, `vic-machine create` creates the following trusted CA, server, and client certificate/key pairs in the certificate folder:
  - `ca.pem`
  - `ca-key.pem`
  - `cert.pem`
  - `key.pem`
  - `server-cert.pem`
  - `server-key.pem`
- Creates a browser-friendly PFX client certificate, `cert.pfx`, to use to authenticate connections to the VCH Admin portal for the VCH.

**NOTE**: The folder and file permissions for the generated certificate and key are readable only by the user who created them.

Running `vic-machine create` with the `--tls-cname` option also creates an environment file named `vch_name.env`, that contains Docker environment variables that container developers can use to configure their Docker client environment:

- Activates TLS client verification.

  ```
  DOCKER_TLS_VERIFY=1
  ```

- The path to the client certificates.

  ```
  DOCKER_CERT_PATH=path_to_certs
  ```

- The address of the VCH.

  ```
  DOCKER_HOST=vch_address:2376
  ```

You must provide copies of the `cert.pem` and `key.pem` client certificate files and the environment file to container developers so that they can connect Docker clients to the VCH. If you deploy the VCH with the `--tls-cname` option, container developers must configure the client appropriately with one of the following options:

- By using the following `tlsverify`, `tlscert`, and `tlskey` Docker options, adding `tlscacert` if a custom CA was used to sign the server certificate.
- By setting `DOCKER_CERT_PATH=/path/to/client/cert.pem` and `DOCKER_TLS_VERIFY=1`.

```
--tls-cname vch-name.example.org
```

```
--tls-cname *.example.org
```

### `--tls-ca`

Short name: `--ca`

You can specify `--tls-ca` multiple times, to point `vic-machine create` to a file that contains the public portion of a CA. `vic-machine create` uses these CAs to validate client certificates that are offered as credentials for Docker API access. This does not need to be the same CA that you use to sign the server certificate.

```
--tls-ca path_to_ca_file
```

**NOTE**: The `--tls-ca` option appears in the extended help that you see by running `vic-machine-os create --extended-help` or `vic-machine-os create -x`.

## Unrestricted access to the Docker API

To deploy a VCH that does not restrict access to the Docker API, use the `--no-tlsverify` option.

### `--no-tlsverify`

Short name: `--kv`

The `--no-tlsverify` option prevents the use of CAs for client authentication. You still require a server certificate if you use `--no-tlsverify`. You can supply a custom server certificate by using the `--cert` and `--key` options. If you do not use `--cert` and `--key` to supply a custom server certificate, `vic-machine create` generates a self-signed server certificate.

When you specify the `--no-tlsverify` option, `vic-machine create` performs the following actions during the deployment of the VCH.

- Generates a self-signed server certificate if you do not specify `--cert` and `--key`.
- Creates a folder with the same name as the VCH in the location in which you run `vic-machine create`.
- Creates an environment file named `vch_name.env` in that folder, that contains the `DOCKER_HOST=vch_address` environment variable, that you can provide to container developers to use to set up their Docker client environment.

If you deploy a VCH with the `--no-tlsverify` option, container developers run Docker commands with the `--tls` option, and the `DOCKER_TLS_VERIFY` environment variable must not be set. Note that setting `DOCKER_TLS_VERIFY` to 0 or `false` has no effect.

The `--no-tlsverify` option takes no arguments.

```
--no-tlsverify
```

# Private Registry Options

If container developers need to access Docker images that are stored in private registry servers, you must configure VCHs to allow them to connect to the private registry servers when you deploy the VCHs. VCHs can connect to both secure and insecure private registry servers.

### `--registry-ca`

Short name: `--rc`

The path to a CAcertificate that can validate the registry's server certificate. You can specify `--registry-ca` multiple times to specify multiple CAcertificates for different registries. The use of these certificates is independent of the client security options that you specify. For example, it is possible to disable TLS for client authentication by using `--no-tls`, and to use `--registry-ca` tospecify CAcertificates to validate a private registry.

```
--registry-ca path_to_ca_cert_1
--registry-ca path_to_ca_cert_2
```

**NOTE**: The `--registry-ca` option appears in the extended help that you see by running `vic-machine-os create --extended-help` or `vic-machine-os create -x`.

### `--insecure-registry`

Short name: `--dir`

An insecure private registry server is a private registry server for Docker images that does not provide TLS. The VCH cannot confirm the identity of the remote system that it is pulling images from and the communication is not encrypted. Insecure private registries are not recommended in production environments.

If you authorize a VCH to connect to an insecure private registry server, the VCH attempts to access the registry server via HTTP if access via HTTPS fails. VCHs always use HTTPS when connecting to registry servers for which you have not authorized insecure access.

To permit pulling images from an insecure private registry, use the `--insecure-registry` option. You can specify `--insecure-registry` multiple times if multiple insecure registries are permitted. If the registry server listens on a specific port, add the port number to the URL

```
--insecure-registry registry_URL_1
--insecure-registry registry_URL_2:port_number
```

## Datastore Options

The `vic-machine` utility allows you to specify the datastore in which to store container image files, container VM files, and the files for the VCH. You can also specify datastores in which to create container volumes.

- vSphere Integrated Containers Engine fully supports VMware vSAN datastores.
- vSphere Integrated Containers Engine supports all alphanumeric characters, hyphens, and underscores in datastore paths and datastore names, but no other special characters.
- If you specify different datastores in the different datastore options, and if no single host in a cluster can access all of those datastores, `vic-machine create` fails with an error.

  ```
  No single host can access all of the requested datastores.
  Installation cannot continue.
  ```

- If you specify different datastores in the different datastore options, and if only one host in a cluster can access all of them, `vic-machine create` succeeds with a warning.

  ```
  Only one host can access all of the image/container/volume datastores.
  This may be a point of contention/performance degradation and HA/DRS
  may not work as intended.
  ```

- VCHs do not support datastore name changes. If a datastore changes name after you have deployed a VCH that uses that datastore, that VCH will no longer function.

## `--image-store`

Short name: `-i`

The datastore in which to store container image files, container VM files, and the files for the VCH. The `--image-store` option is **mandatory** if there is more than one datastore in your vSphere environment. If there is only one datastore in your vSphere environment, the `--image-store` option is not required.

If you do not specify the `--image-store` option and multiple possible datastores exist, or if you specify an invalid datastore name, `vic-machine create` fails and suggests valid datastores in the failure message.

If you are deploying the VCH to a vCenter Server cluster, the datastore that you designate in the `image-store` option must be shared by at least two ESXi hosts in the cluster. Using non-shared datastores is possible, but limits the use of vSphere features such as vSphere vMotion® and VMware vSphere Distributed Resource Scheduler™ (DRS).

To specify a whole datastore as the image store, specify the datastore name in the `--image-store` option:

```
--image-store datastore_name
```

If you designate a whole datastore as the image store, `vic-machine` creates the following set of folders in the target datastore:

- `datastore_name/VIC/vch_uuid/images`, in which to store all of the container images that you pull into the VCH.
- `datastore_name/vch_name`, that contains the VM files for the VCH.
- `datastore_name/vch_name/kvstores`, a key-value store folder for the VCH.

You can specify a datastore folder to use as the image store by specifying a path in the `--image-store` option</code>:

```
--image-store datastore_name/path
```

If the folder that you specify in `/path` does not already exist, `vic-machine create` creates it. Wrap the datastore name and path in single quotes (') on Mac OS and Linux and in double quotes (") on Windows if they include spaces:

```
--image-store 'datastore name'/'datastore path'
```

If you designate a datastore folder as the image store, `vic-machine` creates the following set of folders in the target datastore:

- `datastore_name/path/VIC/vcu_uuid/images`, in which to store all of the container images that you pull into the VCH.
- `datastore_name/vch_name`, that contains the VM files for the VCH. This is the same as if you specified a datastore as the image store.
- `datastore_name/vch_name/kvstores`, a key-value store folder for the VCH. This is the same as if you specified a datastore as the image store.

By specifying the path to a datastore folder in the `--image-store` option, you can designate the same datastore folder as the image store for multiple VCHs. In this way, `vic-machine create` creates only one `VIC` folder in the datastore, at the path that you specify. The `VIC` folder contains one `vch_uuid/images` folder for each VCH that you deploy. By creating one `vch_uuid/images` folder for each VCH, vSphere Integrated Containers Engine limits the potential for conflicts of image use between VCHs, even if you share the same image store folder between multiple hosts.

When container developers create containers, vSphere Integrated Containers Engine stores the files for container VMs at the top level of the image store, in folders that have the same name as the containers.

### `--volume-store`

Short name: `--vs`

The datastore in which to create volumes when container developers use the `docker volume create` or `docker create -v` commands. When you specify the `volume-store` option, you provide the name of the target datastore and a label for the volume store. You can optionally provide a path to a specific folder in the datastore in which to create the volume store. If the folders that you specify in the path do not already exist on the datastore, `vic-machine create` creates the appropriate folder structure.

The `vic-machine create` creates command creates the `volumes` folder independently from the folders for VCH files so that you can share volumes between VCHs. If you delete a VCH, any volumes that the VCH managed will remain available in the volume store unless you specify the `--force` option when you delete the VCH. You can then assign an existing volume store that already contains data to a newly created VCH.

**IMPORTANT**: If multiple VCHs will use the same datastore for their volume stores, specify a different datastore folder for each VCH. Do not designate the same datastore folder as the volume store for multiple VCHs.

If you are deploying the VCH to a vCenter Server cluster, the datastore that you designate in the `volume-store` option should be shared by at least two ESXi hosts in the cluster. Using non-shared datastores is possible and `vic-machine create` succeeds, but it issues a warning that this configuration limits the use of vSphere features such as vSphere vMotion and DRS.

The label that you specify is the volume store name that Docker uses. For example, the volume store label appears in the information for a VCH when container developers run `docker info`. Container developers specify the volume store label in the `docker volume create --opt VolumeStore=volume_store_label` option when they create a volume.

If you specify an invalid datastore name, `vic-machine create` fails and suggests valid datastores.

**IMPORTANT** If you do not specify the `volume-store` option, no volume store is created and container developers cannot use the `docker volume create` or `docker create -v` commands.

- If you only require one volume store, you can set the volume store label to `default`. If you set the volume store label to `default`, container developers do not need to specify the `--opt VolumeStore=volume_store_label` option when they run `docker volume create`.

  **NOTE**: If container developers intend to use `docker create -v` to create containers that are attached to anonymous or named volumes, you must create a volume store with a label of `default`.

  ```
  --volume-store datastore_name:default
  ```

- If you specify the target datastore and the volume store label, `vic-machine create` creates a folder named `VIC/volumes` at the top level of the target datastore. Any volumes that container developers create will appear in the `VIC/volumes` folder.

  ```
  --volume-store datastore_name:volume_store_label
  ```

- If you specify the target datastore, a datastore path, and the volume store label, `vic-machine create` creates a folder named `volumes` in the location that you specify in the datastore path. Any volumes that container developers create will appear in the `path/volumes` folder.

  ```
  --volume-store datastore_name/datastore_path:volume_store_label
  ```

- Wrap the datastore name and path in single quotes (') on Mac OS and Linux and in double quotes (") on Windows if they include spaces. The volume store label cannot include spaces.

  ```
  --volume-store 'datastore name'/'datastore path':volume_store_label
  ```

- You can specify the `volume-store` option multiple times, to create multiple volume stores for the VCH.

  ```
  --volume-store datastore_name/path:volume_store_label_1
  --volume-store datastore_name/path:volume_store_label_2
  [...]
  --volume-store datastore_name/path:volume_store_label_n
  ```

# Networking Options

The `vic-machine create` utility allows you to specify different networks for the different types of traffic between containers, the VCH, the external internet, and your vSphere environment. For information about the different networks that VCHs use, see Networks Used by vSphere Integrated Containers Engine.

**IMPORTANT**: A VCH supports a maximum of 3 distinct network interfaces. Because the bridge network requires its own port group, at least two of the public, client, and management networks must share a network interface and therefore a port group. Container networks do not go through the VCH, so they are not subject to this limitation. This limitation will be removed in a future release.

By default, `vic-machine create` obtains IP addresses for VCH endpoint VMs by using DHCP. For information about how to specify a static IP address for the VCH endpoint VM on the client, public, and management networks, see Specify a Static IP Address for the VCH Endpoint VM in Advanced Options.

If your network access is controlled by a proxy server, see Options to Configure VCHs to Use Proxy Servers in Advanced Options.

When you specify different network interfaces for the different types of traffic, `vic-machine create` checks that the firewalls on the ESXi hosts allow connections to port 2377 from those networks. If access to port 2377 on one or more ESXi hosts is subject to IP address restrictions, and if those restrictions block access to the network interfaces that you specify, `vic-machine create` fails with a firewall configuration error:

```
Firewall configuration incorrect due to allowed IP restrictions on hosts:
"/ha-datacenter/host/localhost.localdomain/localhost.localdomain"
Firewall must permit dst 2377/tcp outbound to the VCH management interface
```

### `--bridge-network`

Short name: `-b`

A port group that container VMs use to communicate with each other.

The `bridge-network` option is **mandatory** if you are deploying a VCH to vCenter Server.

In a vCenter Server environment, before you run `vic-machine create`, you must create a distributed virtual switch and a port group. You must add the target ESXi host or hosts to the distributed virtual switch, and assign a VLAN ID to the port group, to ensure that the bridge network is isolated. For information about how to create a distributed virtual switch and port group, see vCenter Server Network Requirements in *Environment Prerequisites for vSphere Integrated Containers Engine Installation*.

You pass the name of the port group to the `bridge-network` option. Each VCH requires its own port group.

**IMPORTANT**

- Do not assign the same `bridge-network` port group to multiple VCHs. Sharing a port group between VCHs might result in multiple container VMs being assigned the same IP address.
- Do not use the `bridge-network` port group as the target for any of the other `vic-machine create` networking options.

If you specify an invalid port group name, `vic-machine create` fails and suggests valid port groups.

The `bridge-network` option is **optional** when you are deploying a VCH to an ESXi host with no vCenter Server. In this case, if you do not specify `bridge-network`, `vic-machine` creates a virtual switch and a port group that each have the same name as the VCH. You can optionally specify this option to assign an existing port group for use as the bridge network for container VMs. You can also optionally specify this option to create a new virtual switch and port group that have a different name to the VCH.

```
--bridge-network distributed_port_group_name
```

Wrap the port group name in single quotes (') on Mac OS and Linux and in double quotes (") on Windows if it includes spaces.

```
--bridge-network 'port group name'
```

For information about how to specify a range of IP addresses for additional bridge networks, see `bridge-network-range` in Advanced Networking Options.

## `--client-network`

Short name: `--cln`

A port group on which the VCH will make the Docker API available to Docker clients. Docker clients use this network to issue Docker API requests to the VCH.

If not specified, the VCH uses the public network for client traffic. If you specify an invalid port group name, `vic-machine create` fails and suggests valid port groups.

```
--client-network port_group_name
```

Wrap the port group name in single quotes (') on Mac OS and Linux and in double quotes (") on Windows if it includes spaces.

```
--client-network 'port group name'
```

## `--public-network`

Short name: `--pn`

A port group for containers to use to connect to the Internet. VCHs use the public network to pull container images, for example from https://hub.docker.com/. Containers that use use port mapping expose network services on the public interface.

**NOTE**: vSphere Integrated Containers Engine adds a new capability to Docker that allows you to directly map containers to a network by using the `--container-network` option. This is the recommended way to deploy container services.

If not specified, containers use the VM Network for public network traffic. If you specify an invalid port group name, `vic-machine create` fails and suggests valid port groups.

```
--public-network port_group
```

Wrap the network name in single quotes (') on Mac OS and Linux and in double quotes (") on Windows if it includes spaces.

```
--public-network 'port group name'
```

## `--management-network`

Short name: `--mn`

A port group that the VCH uses to communicate with vCenter Server and ESXi hosts. Container VMs use this network to communicate with the VCH.

**IMPORTANT**: Because the management network provides access to your vSphere environment, and because container VMs use this network to communicate with the VCH, always use a secure network for the management network.

When you create a VCH, `vic-machine create` checks that the firewall on ESXi hosts allows connections to port 2377 from the management network of the VCH. If access to port 2377 on ESXi hosts is subject to IP address restrictions, and if those restrictions block access to the management network interface, `vic-machine create` fails with a firewall configuration error:

```
Firewall configuration incorrect due to allowed IP restrictions on hosts:
"/ha-datacenter/host/localhost.localdomain/localhost.localdomain"
Firewall must permit dst 2377/tcp outbound to the VCH management interface
```

**NOTE**: If the management network uses DHCP, `vic-machine` checks the firewall status of the management network before the VCH receives an IP address. It is therefore not possible to fully assess whether the firewall permits the IP address of the VCH. In this case, `vic-machine create` issues a warning.

```
Unable to fully verify firewall configuration due to DHCP use on management network
VCH management interface IP assigned by DHCP must be permitted by allowed IP settings
Firewall allowed IP configuration may prevent required connection on hosts:
"/ha-datacenter/host/localhost.localdomain/localhost.localdomain"
Firewall must permit dst 2377/tcp outbound to the VCH management interface
```

If not specified, the VCH uses the public network for management traffic. If you specify an invalid port group name, `vic-machine create` fails and suggests valid port groups.

```
--management-network port_group_name
```

Wrap the network name in single quotes (') on Mac OS and Linux and in double quotes (") on Windows if it includes spaces.

```
--management-network 'port group name'
```

## `--container-network`

Short name: `--cn`

A port group for container VMs to use for external communication when container developers run `docker run` or `docker create` with the `--net` option.

You can optionally specify one or more container networks. Container networks allow containers to directly attach to a network without having to route through the VCH via network address translation (NAT). Container networks that you add by using the `--container-network` option appear when you run the `docker network ls` command. These networks are available for use by containers. Containers that use these networks are directly attached to the container network, and do not go through the VCH or share the public IP of the VCH.

**IMPORTANT**: For security reasons, whenever possible, use separate port groups for the container network and the management network.

To specify a container network, you provide the name of a port group for the container VMs to use, and an optional descriptive name for the container network for use by Docker. If you do not specify a descriptive name, Docker uses the vSphere network name. If you specify an invalid port group name, `vic-machine create` fails and suggests valid port groups.

- You can specify a vSphere network as the container network.
- The port group must exist before you run `vic-machine create`.
- You cannot use the same port group as you use for the bridge network.
- You can create the port group on the same distributed virtual switch as the port group that you use for the bridge network.
- If the port group that you specify in the `container-network` option does not support DHCP, see Options for Configuring a Non-DHCP Network for Container Traffic in Advanced Options.
- The descriptive name appears under `Networks` when you run `docker info` or `docker network ls` on the deployed VCH.
- Container developers use the descriptive name in the `--net` option when they run `docker run` or `docker create`.

You can specify `--container-network` multiple times to add multiple vSphere networks to Docker.

If you do not specify `--container-network`, or if you deploy containers that do not use a container network, the containers' network services are still be available via port mapping through the VCH, by using NAT through the public interface of the VCH.

```
--container-network port_group_name:container_port _group_name
```

Wrap the port group name in single quotes (') on Mac OS and Linux and in double quotes (") on Windows if it includes spaces. The descriptive name cannot include spaces.

```
--container-network 'port group name':container port group name
```

# Additional Deployment Options

The `vic-machine` utility provides options to customize the VCH.

### `--name`

Short name: `-n`

Aname for the VCH. If not specified, `vic-machine` sets the name of the VCH to `virtual-container-host`. If a VCH of the same name exists on the ESXi host or in the vCenter Server inventory, or if a folder of the same name exists in the target datastore, `vic-machine create` creates a folder named `vch_name_1`

```
--name vch_name
```

Wrap the name in single quotes (') on Mac OS and Linux and in double quotes (") on Windows if it includes spaces.

```
--name 'vch name'
```

### `--memory`

Short name: `--mem`

Limit the amount of memory that is available for use by the VCH vApp in vCenter Server, or for the VCH resource pool on an ESXi host. This limit also applies to the container VMs that run in the VCH vApp or resource pool. Specify the memory limit value in MB. If not specified, `vic-machine create` sets the limit to 0 (unlimited).

```
--memory 1024
```

### `--cpu`

Short name: None

Limit the amount of CPU capacity that is available for use by the VCH vApp in vCenter Server, or for the VCH resource pool on an ESXi host. This limit also applies to the container VMs that run in the VCH vApp or resource pool. Specify the CPU limit value in MHz. If not specified, `vic-machine create` sets the limit to 0 (unlimited).

```
--cpu 1024
```

### `--force`

Short name: `-f`

Forces `vic-machine create` to ignore warnings and non-fatal errors and continue with the deployment of a VCH. Errors such as an incorrect compute resource still cause the installation to fail.

If your vSphere environment uses untrusted, self-signed certificates, you can use the `--force` option to deploy a VCH without providing the thumbprint of the vCenter Server or ESXi host in the `thumbprint` option.

**IMPORTANT** Running `vic-machine create` with the `--force` option rather than providing the certificate thumbprint is not recommended, because it permits man-in-the-middle attacks to go undetected.

```
--force
```

### `--timeout`

Short name: none

The timeout period for uploading the vSphere Integrated Containers Engine files and ISOs to the ESXi host, and for powering on the VCH. Specify a value in the format `XmYs` if the default timeout of 3m0s is insufficient.

```
--timeout 5m0s
```

# Advanced Options

The options in this section are exposed in the `vic-machine create` help if you run `vic-machine-operating_system create --extended-help`, or `vic-machine-operating_system create -x`.

# Advanced Security Options

The advanced security options allow you to customize the authentication of connections from Docker clients to VCHs.

- Add optional information to auto-generated trusted TLS certificates by specifying the `--certificate-key-size`, and `--organization` options.
- Use custom server certificates by using the `--cert` and `--key` options.
- Disable TLS authentication completely by using the `--no-tls` option.

### `--certificate-key-size`

Short name: `--ksz`

The size of the key for `vic-machine create` to use when it creates auto-generated trusted certificates. If not specified, `vic-machine create` creates keys with default size of 2048 bits. It is not recommended to use key sizes of less than 2048 bits.

```
--certificate-key-size 3072
```

### `--organization`

Short name: None

Alist of identifiers to record in certificates generated by vic-machine. If not specified, `vic-machine create` uses the name of the VCH as the organization value.

**NOTE**: The `client-ip-address` is used for `CommonName` but not for `Organisation`.

```
--organization organization_name
```

### `--cert`

Short name: none

The path to a custom X.509 server certificate. This certificate identifies the VCH endpoint VM both to Docker clients and to browsers that connect to the VCH Admin portal.

- This certificate should have the following certificate usages:
  - `KeyEncipherment`
  - `DigitalSignature`
  - `KeyAgreement`
  - `ServerAuth`
- This option is mandatory if you use custom TLS certificates, rather than auto-generated certificates.
- Use this option in combination with the `--key` option, that provides the path to the private key file for the custom certificate.
- Include the names of the certificate and key files in the paths.
- If you use trusted custom certificates, container developers run Docker commands with the `--tlsverify`, `--tlscacert`, `--tlscert`, and `--tlskey` options.

```
--cert path_to_certificate_file/certificate_file_name.pem
--key path_to_key_file/key_file_name.pem
```

Wrap the folder names in the paths in single quotes (Linux or Mac OS) or double quotes (Windows) if they include spaces.

```
--cert 'path to certificate file'/certificate_file_name.pem
--key 'path to key file'/key_file_name.pem
```

## `--key`

Short name: none

The path to the private key file to use with a custom server certificate. This option is mandatory if you specify the `--cert` option, that provides the path to a custom X.509 certificate file. Include the names of the certificate and key files in the paths.

```
--cert path_to_certificate_file/certificate_file_name.pem
--key path_to_key_file/key_file_name.pem
```

Wrap the folder names in the paths in single quotes (Linux or Mac OS) or double quotes (Windows) if they include spaces.

```
--cert 'path to certificate file'/certificate_file_name.pem
--key 'path to key file'/key_file_name.pem
```

## `--cert-path`

Short name: none

By default `--cert-path` is a folder in the current directory, that takes its name from the VCH name that you specify in the `--name` option. `vic-machine create` checks in `--cert-path` for existing certificates with the standard names and uses those certificates if they are present:

- `server-cert.pem`

- `server-key.pem`
- `ca.pem`

If `vic-machine create` does not find existing certificates with the standard names in `--cert-path`, or if you do not specify certificates directly by using the `--cert`, `--key`, and `--tls-ca` options, `vic-machine create` generates certificates. Generated certificates are saved in the `--cert-path` folder with the standard names listed. `vic-machine create` additionally generates other certificates:

- `cert.pem` and `key.pem` for client certificates, if required.
- `ca-key.pem`, the private key for the certificate authority.

```
--cert-path 'path_to_certificate_folder'
```

## `--no-tls`

Short name: `-k`

Disables TLS authentication of connections between the Docker client and the VCH.

Set the `no-tls` option if you do not require TLS authentication between the VCH and the Docker client. Any Docker client can connect to the VCH if you disable TLS authentication.

If you use the `no-tls` option, container developers connect Docker clients to the VCH via port 2375, instead of via port 2376.

```
--no-tls
```

## `--ops-user`

Short name: None

AvSphere user account with which the VCH runs after deployment. Because deploying a VCH requires greater levels of permissions than running a VCH, you can configure a VCH so that it uses different user accounts for deployment and for operation. In this way, you can limit the day-to-day operation of a VCH to an account that does not have full administrator permissions on the target vCenter Server.

If not specified, the VCH runs with the credentials with which you deploy the VCH, that you specify in either `--target` or `--user`.

```
--ops-user user_name
```

Wrap the user name in single quotes (') on Mac OS and Linux and in double quotes (") on Windows if it includes special characters.

```
--ops-user 'user_n@me'
```

## `--ops-password`

Short name: None

The password or token for the operations user that you specify in `--ops-user`.

If not specified, the VCH runs with the credentials with which you deploy the VCH, that you specify in either `--target` or `--user`.

```
--ops-password password
```

Wrap the password in single quotes (') on Mac OS and Linux and in double quotes (") on Windows if it includes special characters.

```
--ops-password 'p@ssword'
```

# Options for Specifying a Static IP Address for the VCH Endpoint VM

You can specify a static IP address for the VCH endpoint VM on each of the client, public, and management networks. DHCP is used for the endpoint VM for any network on which you do not specify a static IP address.

To specify a static IP address for the endpoint VM on the client, public, or management network, you provide an IP address in the `client/public/management-network-ip` option. If you set a static IP address, you must also provide a gateway address. You can optionally specify one or more DNS server addresses.

### `--dns-server`

Short name: None

ADNS server for the VCH endpoint VM to use on the client, public, or management networks. You can specify `dns-server` multiple times, to configure multiple DNS servers.

- If you specify `dns-server`, `vic-machine create` always uses the `--dns-server` setting for all three of the client, public, and management networks.
- If you do not specify `dns-server` and you specify a static IP address for the endpoint VM on all three of the client, public, and management networks, `vic-machine create` uses the Google public DNS service.
- If you do not specify `dns-server` and you use a mixture of static IP addresses and DHCP for the client, public, and management networks, `vic-machine create` uses the DNS servers that DHCP provides.
- If you do not specify `dns-server` and you use DHCP for all of the client, public, and management networks, `vic-machine create` uses the DNS servers that DHCP provides.

```
--dns-server=172.16.10.10
--dns-server=172.16.10.11
```

### `--client-network-ip`, `--public-network-ip`, `--management-network-ip`

Short name: None

Astatic IP address for the VCH endpoint VM on the public, client, or management network.

You specify a static IP address for the endpoint VM on the public, client, or management networks by using the `--public/client/management-network-ip` options. If you set a static IP address for the endpoint VM on any of the networks, you must specify a corresponding gateway address by using the `--public/client/management-network-gateway` option.

- You can only specify one static IP address on a given port group. If more than one of the client, public, or management networks
- s a port group, you can only specify a static IP address on one of those networks. All of the networks that share that port group use the IP address that you specify.
- If either of the client or management networks shares a port group with the public network, you can only specify

a static IP address on the public network.

- If either or both of the client or management networks do not use the same port group as the public network, you can specify a static IP address for the endpoint VM on those networks by using `--client-network-ip` or `--management-network-ip`, or both. In this case, you must specify a corresponding gateway address by using `client/management-network-gateway`.
- If the client and management networks both use the same port group, and the public network does not use that port group, you can set a static IP address for the endpoint VM on either or both of the client and management networks.
- If you assign a static IP address to the VCH endpoint VM on the client network by setting the `--client-network-ip` option, and you do not specify one of the TLS options, `vic-machine create` uses this address as the Common Name with which to auto-generate trusted CA certificates. If you do not specify `--tls-cname`, `--no-tls` or `--no-tlsverify`, two-way TLS authentication with trusted certificates is implemented by default when you deploy the VCH with a static IP address on the client network. If you assign a static IP address to the endpoint VM on the client network, `vic-machine create` creates the same certificate and environment variable files as described in the `--tls-cname` option.

  **IMPORTANT**: If the client network shares a port group with the public network you cannot set a static IP address for the endpoint VM on the client network. To assign a static IP address to the endpoint VM you must set a static IP address on the public network by using the `--public-network-ip` option. In this case, `vic-machine create` uses the public network IP address as the Common Name with which to auto-generate trusted CA certificates, in the same way as it would for the client network.

- If you do not specify an IP address for the endpoint VM on a given network, `vic-machine create` uses DHCP to obtain an IP address for the endpoint VM on that network.
- When you specify an address, `vic-machine create` uses the netmask from the gateway.

You can specify addresses as IPv4 addresses. Do not use CIDR notation.

```
--public-network-ip 192.168.X.N
--management-network-ip 192.168.Y.N
--client-network-ip 192.168.Z.N
```

You can also specify addresses as resolvable FQDNs.

```
--public-network-ip=vch27-team-a.internal.domain.com
--management-network-ip=vch27-team-b.internal.domain.com
--client-network-ip=vch27-team-c.internal.domain.com
```

## `--client-network-gateway`, `--public-network-gateway`, `--management-network-gateway`

Short name: None

The gateway to use if you use `--public/client/management-network-ip` to specify a static IP address for the VCH endpoint VM on the public, client, or management networks. If you specify a static IP address on any network, you must specify a gateway by using the `--public/client/management-network-gateway` options.

You specify the public network gateway address in CIDR format.

```
--public-network-gateway 192.168.X.1/24
```

**IMPORTANT**: Assigning the same subnet to multiple port groups can cause routing problems. If `vic-machine create` detects that you have assigned the same subnet to multiple port groups, it issues a warning.

The default route for the VCH endpoint VM is always on the public network. As a consequence, if you specify a static IP address on either of the management or client networks, you must specify the routing destination for those networks in the `--management-network-gateway` and `--client-network-gateway` options. You specify the routing destination or destinations in a comma-separated list, with the address of the gateway separated from the routing destinations by a colon (:). You specify the gateway addresses in CIDR format:

```
--management-network-gateway routing_destination_1/subnet_mask,
routing_destination_2/subnet_mask:
gateway_address/subnet_mask
```

```
--client-network-gateway routing_destination_1/subnet_mask,
routing_destination_2/subnet_mask:
gateway_address/subnet_mask
```

In the following example, `--management-network-gateway` informs the VCH that it can reach all of the vSphere management endoints that are in the ranges 192.168.3.0-255 and 192.168.128.0-192.168.131.255 by sending packets to the gateway at 192.168.2.1. Ensure that the address ranges that you specify include all of the systems that will connect to this VCH instance.

```
--management-network-gateway 192.168.3.0/24,192.168.128.0/22:192.168.2.1/24
```

# Options for Configuring a Non-DHCP Network for Container Traffic

If the network that you specify in the `container-network` option does not support DHCP, you must specify the `container-network-gateway` option. You can optionally specify one or more DNS servers and a range of IP addresses for container VMs on the container network.

For information about the container network, see the section on the `container-network` option.

## `--container-network-gateway`

Short name: `--cng`

The gateway for the subnet of the container network. This option is required if the network that you specify in the `--container-network` option does not support DHCP. Specify the gateway in the format `container_network:subnet`. If you specify this option, it is recommended that you also specify the `--container-network-dns` option.

When you specify the container network gateway, you must use the port group that you specify in the `--container-network` option. If you specify `--container-network-gateway` but you do not specify `--container-network`, or if you specify a different port group to the one that you specify in `--container-network`, `vic-machine create` fails with an error.

```
--container-network-gateway port_group_name:gateway_ip_address/subnet_mask
```

Wrap the port group name in single quotes (Linux or Mac OS) or double quotes (Windows) if it includes spaces.

```
--container-network-gateway 'port group name':gateway_ip_address/subnet_mask
```

### --container-network-dns

Short name: `--cnd`

The address of the DNS server for the container network. This option is recommended if the network that you specify in the `--container-network` option does not support DHCP.

When you specify the container network DNS server, you must use the port group that you specify in the `--container-network` option. You can specify `--container-network-dns` multiple times, to configure multiple DNS servers. If you specify `--container-network-dns` but you do not specify `--container-network`, or if you specify a different port group to the one that you specify in `--container-network`, `vic-machine create` fails with an error.

```
--container-network-dns port_group_name:8.8.8.8
```

Wrap the port group name in single quotes (Linux or Mac OS) or double quotes (Windows) if it includes spaces.

```
--container-network-dns 'port group name':8.8.8.8
```

### --container-network-ip-range

Short name: `--cnr`

The range of IP addresses that container VMs can use if the network that you specify in the `container-network` option does not support DHCP. If you specify `--container-network-ip-range`, VCHs manage the addresses for containers within that range. The range that you specify must not be used by other computers or VMs on the network. If you specify `container-network-gateway` but do not specify `--container-network-ip-range`, the IP range for container VMs is the entire subnet that you specify in `--container-network-gateway`.

When you specify the container network IP range, you must use the port group that you specify in the `--container-network` option. If you specify `--container-network-ip-range` but you do not specify `--container-network`, or if you specify a different port group to the one that you specify in `--container-network`, `vic-machine create` fails with an error.

```
--container-network-ip-range port_group_name:192.168.100.2-192.168.100.254
```

You can also specify the IP range as a CIDR.

```
--container-network-ip-range port_group_name:192.168.100.0/24
```

Wrap the port group name in single quotes (Linux or Mac OS) or double quotes (Windows) if it includes spaces.

```
--container-network-ip-range 'port group name':192.168.100.0/24
```

# Options to Configure VCHs to Use Proxy Servers

If access to the Internet or to your private image registries requires the use of a proxy server, you must configure a VCH to connect to the proxy server when you deploy it. The proxy is used only when pulling images, and not for any other purpose.

**IMPORTANT**: Configuring a VCH to use a proxy server does not configure proxy support on the containers that this VCH runs. Container developers must configure proxy servers on containers when they create them.

### `--https-proxy`

Short name: `--sproxy`

The address of the HTTPS proxy server through which the VCH accesses image registries when using HTTPS. Specify the address of the proxy server as either an FQDN or an IP address.

```
--https-proxy https://proxy_server_address:port
```

### `--http-proxy`

Short name: `--hproxy`

The address of the HTTP proxy server through which the VCH accesses image registries when using HTTP. Specify the address of the proxy server as either an FQDN or an IP address.

```
--http-proxy http://proxy_server_address:port
```

## Advanced Resource Management Options

You can set limits on the memory and CPU shares and reservations on the VCH. For information about memory and CPU shares and reservations, see Allocate Memory Resources, and Allocate CPU Resources in the vSphere documentation.

### `--memory-reservation`

Short name: `--memr`

Reserve a quantity of memory for use by the VCH vApp in vCenter Server, or for the VCH resource pool on an ESXi host. This limit also applies to the container VMs that run in the VCH vApp or resource pool. Specify the memory reservation value in MB. If not specified, `vic-machine create` sets the reservation to 1.

```
--memory-reservation 1024
```

### `--memory-shares`

Short name: `--mems`

Set memory shares on the VCH vApp in vCenter Server, or on the VCH resource pool on an ESXi host. This limit also applies to the container VMs that run in the VCH vApp or resource pool. Specify the share value as a level or a number, for example `high`, `normal`, `low`, or `163840`. If not specified, `vic-machine create` sets the share to `normal`.

```
--memory-shares low
```

### `--cpu-reservation`

Short name: `--cpur`

Reserve a quantity of CPU capacity for use by the VCH vApp in vCenter Server, or for the VCH resource pool on an ESXi host. This limit also applies to the container VMs that run in the VCH vApp or resource pool. Specify the CPU reservation value in MHz. If not specified, `vic-machine create` sets the reservation to 1.

```
--cpu-reservation 1024
```

### `--cpu-shares`

Short name: `--cpus`

Set CPU shares on the VCH vApp in vCenter Server, or on the VCH resource pool on an ESXi host. This limit also applies to the container VMs that run in the VCH vApp or resource pool. Specify the share value as a level or a number, for example `high`, `normal`, `low`, or `163840`. If not specified, `vic-machine create` sets the share to `normal`.

```
--cpu-shares low
```

### `--appliance-cpu`

Short name: none

The number of virtual CPUs for the VCH endpoint VM. The default is 1. Set this option to increase the number of CPUs in the VCH endpoint VM.

**NOTE** Always use the `--cpu` option instead of the `--appliance-cpu` option to increase the overall CPU capacity of the VCH vApp, rather than increasing the number of CPUs on the VCH endpoint VM. The `--appliance-cpu` option is mainly intended for use by VMware Support.

```
--appliance-cpu number_of_CPUs
```

### `--appliance-memory`

Short name: none

The amount of memory for the VCH endpoint VM. The default is 2048MB. Set this option to increase the amount of memory in the VCH endpoint VM if the VCH will pull large container images.

**NOTE** With the exception of VCHs that pull large container images, always use the `--memory` option instead of the `--appliance-memory` option to increase the overall amount of memory for the VCH vApp, rather than on the VCH endpoint VM. Use `docker create -m` to set the memory on container VMs. The `--appliance-memory` option is mainly intended for use by VMware Support.

```
--appliance-memory amount_of_memory
```

## Other Advanced Options

### `--bridge-network-range`

Short name: `--bnr`

The range of IP addresses that additional bridge networks can use when container application developers use `docker network create` to create new bridge networks. If you do not specify the `bridge-network-range` option, the IP range for bridge networks is 172.16.0.0/12.

When you specify the bridge network IP range, you specify the IP range as a CIDR. The smallest subnet that you can specify is /16.

```
--bridge-network-range 192.168.100.0/16
```

### `--base-image-size`

Short name: None

The size of the base image from which to create other images. You should not normally need to use this option. Specify the size in `GB` or `MB`. The default size is 8GB. Images are thin-provisioned, so they do not usually consume 8GB of space.

```
--base-image-size 4GB
```

### `--container-store`

Short name: `--cs`

The `container-store` option is not enabled. Container VM files are stored in the datastore that you designate as the image store.

### `--appliance-iso`

Short name: `--ai`

The path to the ISO image from which the VCH appliance boots. Set this option if you have moved the `appliance.iso` file to a folder that is not the folder that contains the `vic-machine` binary or is not the folder from which you are running `vic-machine`. Include the name of the ISO file in the path.

**NOTE**: Do not use the `--appliance-iso` option to point `vic-machine` to an `--appliance-iso` file that is of a different version to the version of `vic-machine` that you are running.

```
--appliance-iso path_to_ISO_file/appliance.iso
```

Wrap the folder names in the path in single quotes (Linux or Mac OS) or double quotes (Windows) if they include spaces.

```
--appliance-iso 'path to ISO file'/appliance.iso
```

### `--bootstrap-iso`

Short name: `--bi`

The path to the ISO image from which to boot container VMs. Set this option if you have moved the `bootstrap.iso` file to a folder that is not the folder that contains the `vic-machine` binary or is not the folder from which you are running `vic-machine`. Include the name of the ISO file in the path.

**NOTE**: Do not use the `--bootstrap-iso` option to point `vic-machine` to a `--bootstrap-iso` file that is of a different version to the version of `vic-machine` that you are running.

```
--bootstrap-iso path_to_ISO_file/bootstrap.iso
```

Wrap the folder names in the path in single quotes (Linux or Mac OS) or double quotes (Windows) if they include spaces.

```
--bootstrap-iso 'path to ISO file'/bootstrap.iso
```

### `--use-rp`

Short name: none

Deploy the VCH appliance to a resource pool on vCenter Server rather than to a vApp. If you specify this option, `vic-machine create` creates a resource pool with the same name as the VCH.

```
--use-rp
```

### `--debug`

Short name: `-v`

Deploy the VCH with a more verbose level of logging, for troubleshooting purposes. Specifying the `--debug` option increases the verbosity of the logging for all aspects of VCH operation, not just deployment. For example, by setting `--debug`, you increase the verbosity of the logging for VCH initialization, VCH services, container VM initialization, and so on. If not specified, the `debug` value is set to 0 and verbose logging is disabled. Provide a value of 1 or greater to increase the verbosity of the logging. Note that setting debug to a value greater than 1 can affect the behavior of `vic-machine create`. For example, setting `--debug` to 3 suppresses the restart of failed components.

Additionally, deploying a VCH with a `debug` value of 3 or higher enables access to the VCH endpoint VM console by default, with a root password of `password`. This functionality enables you to perform targeted interactive diagnostics in environments in which a VCH endpoint VM failure occurs consistently and in a fashion that prevents `vic-machine debug` from functioning.

**IMPORTANT**: There is no provision for persistently changing the default root password. This configuration should not be used other than for debugging in a secured environment

```
--debug 1
```

# Advanced Examples of Deploying a VCH

This topic provides examples of the options of the `vic-machine create` command to use when deploying virtual container hosts (VCHs) in various vSphere configurations.



For simplicity, these examples use the `--force` option to disable the verification of the vCenter Server certificate, so the `--thumbprint` option is not specified. Similarly, all examples that do not relate explicitly to certificate use specify the `--tls-noverify` option.

For detailed descriptions of all of the `vic-machine create` options, see VCH Deployment Options.

# General Deployment Examples

The examples in this section demonstrate the deployment of VCHs in different vSphere environments.

## Deploy to a vCenter Server Cluster with Multiple Datacenters and Datastores

If vCenter Server has more than one datacenter, you specify the datacenter in the `--target` option.

If vCenter Server manages more than one cluster, you use the `--compute-resource` option to specify the cluster on which to deploy the VCH.

When deploying a VCH to vCenter Server, you must use the `--bridge-network` option to specify an existing port group for container VMs to use to communicate with each other. For information about how to create a distributed virtual switch and port group, see vCenter Server Network Requirements in *Environment Prerequisites for vSphere Integrated Containers Engine Installation*.

This example deploys a VCH with the following configuration:

- Provides the vCenter Single Sign-On user and password in the `--target` option. Note that the user name is wrapped in quotes, because it contains the `@` character. Use single quotes if you are using `vic-machine` on a Linux or Mac OS system and double quotes on a Windows system.
- Deploys a VCH named `vch1` to the cluster `cluster1` in datacenter `dc1`.
- Uses a port group named `vic-bridge` for the bridge network.

- Designates `datastore1` as the datastore in which to store container images, the files for the VCH appliance, and container VMs.

```
vic-machine-operating_system create
--target 'Administrator@vsphere.local':password@vcenter_server_address/dc1
--compute-resource cluster1
--image-store datastore1
--bridge-network vch1-bridge
--name vch1
--force
--no-tlsverify
```

## Deploy to a Specific Standalone Host in vCenter Server

If vCenter Server manages multiple standalone ESXi hosts that are not part of a cluster, you use the `--compute-resource` option to specify the address of the ESXi host to which to deploy the VCH.

This example deploys a VCH with the following configuration:

- Specifies the user name, password, image store, bridge network, and name for the VCH.
- Deploys the VCH on the ESXi host with the FQDN `esxihost1.organization.company.com` in the datacenter `dc1`. You can also specify an IP address.

```
vic-machine-operating_system create
--target 'Administrator@vsphere.local':password@vcenter_server_address/dc1
--image-store datastore1
--bridge-network vch1-bridge
--compute-resource esxihost1.organization.company.com
--name vch1
--force
--no-tlsverify
```

## Deploy to a Resource Pool on an ESXi Host

To deploy a VCH in a specific resource pool on an ESXi host that is not managed by vCenter Server, you specify the resource pool name in the `--compute-resource` option.

This example deploys a VCH with the following configuration:

- Specifies the user name and password, image store, and a name for the VCH.
- Designates `rp 1` as the resource pool in which to place the VCH. Note that the resource pool name is wrapped in quotes, because it contains a space. Use single quotes if you are using `vic-machine` on a Linux or Mac OS system and double quotes on a Windows system.

```
vic-machine-operating_system create
--target root:password@esxi_host_address
--compute-resource 'rp 1'
--image-store datastore1
--name vch1
--force
--no-tlsverify
```

### Deploy to a Resource Pool in a vCenter Server Cluster

To deploy a VCH in a resource pool in a vCenter Server cluster, you specify the names of the cluster and resource pool in the `compute-resource` option.

This example deploys a VCH with the following configuration:

- Specifies the user name, password, datacenter, image store, bridge network, and name for the VCH.
- Designates `rp 1` in cluster `cluster 1` as the resource pool in which to place the VCH. Note that the resource pool and cluster names are wrapped in quotes, because they contain spaces. Use single quotes if you are using `vic-machine` on a Linux or Mac OS system and double quotes on a Windows system.

```
vic-machine-operating_system create
--target 'Administrator@vsphere.local':password@vcenter_server_address/dc1
--compute-resource 'cluster 1'/'rp 1'
--image-store datastore1
--bridge-network vch1-bridge
--name vch1
--force
--no-tlsverify
```

### Set Limits on Resource Use

To limit the amount of system resources that the container VMs in a VCH can use, you can set resource limits on the VCH vApp.

This example deploys a VCH with the following configuration:

- Specifies the user name, password, image store, cluster, bridge network, and name for the VCH.
- Sets resource limits on the VCH by imposing memory and CPU reservations, limits, and shares.

```
vic-machine-operating_system create
--target 'Administrator@vsphere.local':password@vcenter_server_address/dc1
--compute-resource cluster1
--image-store datastore1
--bridge-network vch1-bridge
--memory 1024
--memory-reservation 1024
--memory-shares low
--cpu 1024
--cpu-reservation 1024
--cpu-shares low
--name vch1
--force
--no-tlsverify
```

For more information about setting resource use limitations on VCHs, see the Advanced Deployment Options and Advanced Resource Management Options sections in VCH Deployment Options.

# Networking Examples

The examples in this section demonstrate how to direct traffic to and from VCHs and the other elements in your environment, how to set static IPs, how to configure container VM networks, and how to configure a VCH to use a proxy server.

## Specify Public, Management, and Client Networks

In addition to the mandatory bridge network, if your vCenter Server environment includes multiple networks, you can direct different types of traffic to different networks.

- You can direct the traffic between the VCH and the Internet to a specific network by specifying the `--public-network` option. Any container VM traffic that routes through the VCH also uses the public network. If you do not specify the `--public-network` option, the VCH uses the VM Network for public network traffic.
- You can direct traffic between ESXi hosts, vCenter Server, and the VCH to a specific network by specifying the `--management-network` option. If you do not specify the `--management-network` option, the VCH uses the public network for management traffic.
- You can designate a specific network for use by the Docker API by specifying the `--client-network` option. If you do not specify the `--client-network` option, the Docker API uses the public network.

**IMPORTANT**: A VCH supports a maximum of 3 distinct network interfaces. Because the bridge network requires its own port group, at least two of the public, client, and management networks must share a network interface and therefore a port group. Container networks do not go through the VCH, so they are not subject to this limitation. This limitation will be removed in a future release.

This example deploys a VCH with the following configuration:

- Specifies the user name, password, datacenter, cluster, image store, bridge network, and name for the VCH.
- Directs public and management traffic to network 1 and Docker API traffic to network 2. Note that the network names are wrapped in quotes, because they contain spaces. Use single quotes if you are using `vic-machine` on a Linux or Mac OS system and double quotes on a Windows system.

```
vic-machine-operating_system create
--target 'Administrator@vsphere.local':password@vcenter_server_address/dc1
--compute-resource cluster1
--image-store datastore1
--bridge-network vch1-bridge
--public-network 'network 1'
--management-network 'network 1'
--client-network 'network 2'
--name vch1
--force
--no-tlsverify
```

For more information about the networking options, see the Networking Options section in VCH Deployment Options.

## Set a Static IP Address for the VCH Endpoint VM on the Different Networks

If you specify networks for any or all of the public, management, and client networks, you can deploy the VCH so that the VCH endpoint VM has a static IP address on one or more of those networks.

This example deploys a VCH with the following configuration:

- Specifies the user name, password, datacenter, cluster, image store, bridge network, and name for the VCH.
- Directs public and management traffic to network 1 and Docker API traffic to network 2. Note that the network

names are wrapped in quotes, because they contain spaces. Use single quotes if you are using `vic-machine` on a Linux or Mac OS system and double quotes on a Windows system.

- Sets a DNS server for use by the public, management, and client networks.
- Sets a static IP address for the VCH endpoint VM on the public and client networks. Because the management network shares a network with the public network, you only need to specify the public network IP address. You cannot specify a management IP address because you are sharing a port group between the management and public network.
- Specifies the gateway for the public network. If you set a static IP address on the public network, you must also specify the gateway address.
- Specifies a gateway for the client network. The `--client-network-gateway` option specifies the routing destination for client network traffic through the VCH endpoint VM, as well as the gateway address. The routing destination informs the VCH that it can reach all of the Docker clients at the network addresses in the ranges that you specify in the routing destinations by sending packets to the specified gateway.
- Because this example specifies a static IP address for the VCH endpoint VM on the client network, `vic-machine create` uses this address as the Common Name with which to create auto-generated trusted certificates. Full TLS authentication is implemented by default, so no TLS options are specified.

```
vic-machine-operating_system create
--target 'Administrator@vsphere.local':password@vcenter_server_address/dc1
--compute-resource cluster1
--image-store datastore1
--bridge-network vch1-bridge
--public-network 'network 1'
--public-network-ip 192.168.1.10
--public-network-gateway 192.168.1.1/24
--management-network 'network 1'
--client-network 'network 2'
--client-network-ip 192.168.3.10
--client-network-gateway 192.168.3.0/24,192.168.128.0/22:192.168.2.1/24
--dns-server dns_server_address
--force
--name vch1
```

For more information about setting static IP addresses, see the Options for Specifying a Static IP Address for the VCH Endpoint VM in VCH Deployment Options.

## Configure a Non-DHCP Network for Container VMs

You can designate a specific network for container VMs to use by specifying the `--container-network` option. Containers use this network if the container developer runs `docker run` or `docker create` specifying the `--net` option with one of the specified container networks when they run or create a container. This option requires a port group that must exist before you run `vic-machine create`. You cannot use the same port group that you use for the bridge network. You can provide a descriptive name for the network, for use by Docker. If you do not specify a descriptive name, Docker uses the vSphere network name. For example, the descriptive name appears as an available network in the output of `docker info` and `docker network ls`.

If the network that you designate as the container network in the `--container-network` option does not support DHCP, you can configure the gateway, DNS server, and a range of IP addresses for container VMs to use.

This example deploys a VCH with the following configuration:

- Specifies the user name, password, datacenter, cluster, image store, bridge network, and name for the VCH.

- Uses the VM Network for the public, management, and client networks.
- Designates a port group named `vic-containers` for use by container VMs that are run with the `--net` option.
- Gives the container network the name `vic-container-network`, for use by Docker.
- Specifies the gateway, two DNS servers, and a range of IP addresses on the container network for container VMs to use.

```
vic-machine-operating_system create
--target 'Administrator@vsphere.local':password@vcenter_server_address/dc1
--compute-resource cluster1
--image-store datastore1
--bridge-network vch1-bridge
--container-network vic-containers:vic-container-network
--container-network-gateway vic-containers:gateway_ip_address/24
--container-network-dns vic-containers:dns1_ip_address
--container-network-dns vic-containers:dns2_ip_address
--container-network-ip-range vic-containers:192.168.100.0/24
--name vch1
--force
--no-tlsverify
```

For more information about the container network options, see the `--container-network` and Options for Configuring a Non-DHCP Network for Container Traffic sections in VCH Deployment Options.

### Configure a Proxy Server

If your network access is controlled by a proxy server, you must configure a VCH to connect to the proxy server when you deploy it, so that it can pull images from external sources.

This example deploys a VCH with the following configuration:

- Specifies the user name, password, image store, cluster, bridge network, and name for the VCH.
- Configures the VCH to access the network via an HTTPS proxy server.

```
vic-machine-operating_system create
--target 'Administrator@vsphere.local':password@vcenter_server_address/dc1
--compute-resource cluster1
--image-store datastore1
--bridge-network vch1-bridge
--https-proxy https://proxy_server_address:port
--name vch1
--force
--no-tlsverify
```

## Specify Volume Stores

If container application developers will use the `docker volume create` command to create containers that use volumes, you must create volume stores when you deploy VCHs. You specify volume stores in the `--volume-store` option. You can specify `--volume-store` multiple times to create multiple volume stores.

When you create a volume store, you specify the name of the datastore to use and an optional path to a folder on that datastore. You also specify a descriptive name for that volume store for use by Docker.

This example deploys a VCH with the following configuration:

- Specifies the user name, password, datacenter, cluster, bridge network, and name for the VCH.
- Specifies the `volumes` folder on `datastore 1` as the default volume store. Creating a volume store named `default` allows container application developers to create anonymous or named volumes by using `docker create -v`.
- Specifies a second volume store named `volume_store_2` in the `volumes` folder on `datastore 2`.
- Note that the datastore names are wrapped in quotes, because they contain spaces. Use single quotes if you are using `vic-machine` on a Linux or Mac OS system and double quotes on a Windows system.

```
vic-machine-operating_system create
--target 'Administrator@vsphere.local':password@vcenter_server_address/dc1
--compute-resource cluster1
--bridge-network vch1-bridge
--image-store 'datastore 1'
--volume-store 'datastore 1'/volumes:default
--volume-store 'datastore 2'/volumes:volume_store_2
--name vch1
--force
--no-tlsverify
```

For more information about volume stores, see the volume-store section in VCH Deployment Options.

# Security Examples

The examples in this section demonstrate how to configure a VCH to use Certificate Authority (CA) certificates to enable `TLSVERIFY` in your Docker environment, and to allow access to insecure registries of Docker images.

## Use Auto-Generated Trusted CA Certificates

You can deploy a VCH that implements two-way authentication with trusted auto-generated TLS certificates that are signed by a CA.

To automatically generate a server certificate that can pass client verification, you must specify the Common Name (CN) for the certificate by using the `--tls-cname` option. The CN should be the FQDN or IP address of the server, or a domain with a wildcard. The CN value must match the name or address that clients will use to connect to the server. You can use the `--organization` option to add basic descriptive information to the server certificate. This information is visible to clients if they inspect the server certificate.

If you specify an existing CA file with which to validate clients, you must also provide an existing server certificate that is compatible with the `--tls-cname` value or the IP address of the client interface.

This example deploys a VCH with the following configuration:

- Specifies the user, password, datacenter, image store, cluster, bridge network, and name for the VCH.
- Provides a wildcard domain `*.example.org` as the FQDN for the VCH, for use as the Common Name in the certificate. This assumes that there is a DHCP server offering IP addresses on VM Network, and that those addresses have corresponding DNS entries such as `dhcp-a-b-c.example.com`.
- Specifies a folder in which to store the auto-generated certificates.

```
vic-machine-operating_system create
--target 'Administrator@vsphere.local':password@vcenter_server_address/dc1
--compute-resource cluster1
--image-store datastore1
--bridge-network vch1-bridge
--tls-cname *.example.org
--cert-path path_to_cert_folder
--force
--name vch1
```

The Docker API for this VCH will be accessible at `https://dhcp-a-b-c.example.com:2376`.

For more information about using auto-generated CA certificates, see the Security Options section in VCH Deployment Options.

## Use Custom Server Certificates

You can create a VCH that uses a custom server certificate, for example a server certificate that has been signed by Verisign or another public root. You use the `--cert` and `--key` options to provide the paths to a custom X.509 certificate and its key when you deploy a VCH. The paths to the certificate and key files must be relative to the location from which you are running `vic-machine create`.

This example deploys a VCH with the following configuration:

- Specifies the user name, password, image store, cluster, bridge network, and name for the VCH.
- Provides the paths relative to the current location of the `*.pem` files for the custom server certificate and key files.

```
vic-machine-operating_system create
--target 'Administrator@vsphere.local':password@vcenter_server_address/dc1
--compute-resource cluster1
--image-store datastore1
--bridge-network vch1-bridge
--cert ../some/relative/path/certificate_file.pem
--key ../some/relative/path/key_file.pem
--name vch1
--force
```

For more information about using custom server certificates, see the Advanced Security Options section in VCH Deployment Options.

## Specify Different User Accounts for VCH Deployment and Operation

When you deploy a VCH, you can use different vSphere user accounts for deployment and for operation. This allows you to run VCHs with lower levels of privileges than are required for deployment.

This example deploys a VCH with the following configuration:

- Specifies the image store, cluster, bridge network, and name for the VCH.
- Specifies *vsphere_admin* in the `--target` option, to identify the user account with which to deploy the VCH.
- Specifies *vsphere_user* and its password in the `--ops-user` and `--ops-password` options, to identify the user account with which the VCH runs.

```
vic-machine-operating_system create
--target vsphere_admin:vsphere_admin_password@vcenter_server_address/dc1
--compute-resource cluster1
--image-store datastore1
--bridge-network vch1-bridge
--name vch1
--ops-user vsphere_user
--ops-password vsphere_user_password
--force
--no-tlsverify
```

For more information about using custom server certificates, see --ops-user in VCH Deployment Options.

## Authorize Access to an Insecure Private Registry Server

To authorize connections from a VCH to an insecure private registry server, set the `insecure-registry` option. You can specify `insecure-registry` multiple times to allow connections from the VCH to multiple insecure private registry servers.

This example deploys a VCH with the following configuration:

- Specifies the user name, password, image store, cluster, bridge network, and name for the VCH.
- Authorizes the VCH to pull Docker images from the insecure private registry servers located at the URLs *registry_URL_1* and *registry_URL_2*.
- The registry server at *registry_URL_2* listens for connections on port 5000.

```
vic-machine-operating_system create
--target 'Administrator@vsphere.local':password@vcenter_server_address/dc1
--compute-resource cluster1
--image-store datastore1
--bridge-network vch1-bridge
--insecure-registry registry_URL_1
--insecure-registry registry_URL_2:5000
--name vch1
--force
--no-tlsverify
```

For more information about configuring VCHs to connect to insecure private registry servers, see the section on the `insecure-registry` option in VCH Deployment Options.

# Deploy a VCH with vSphere Integrated Containers Registry (Harbor)

This example uses vSphere Integrated Containers Engine 0.8.0, Harbor 0.5.0, and Ubuntu on the user machine. Harbor requires 60GB or more free space on your datastore.

If no server certificate and private key are provided during installation, Harbor will create a self-generated CA (certificate authority) certificate, a server certificate, and a server private key. The self-generated CAcertificate will be available for download from the Harbor web client.

See the OVAinstallation guide for Harbor Harbor docs. Harbor requires both an IP address and FQDN (fully qualified domain name) for the server. ADHCP install method is available for debugging purposes, but it is not a recommended production deployment model.

This example assumes a Harbor instance was installed without a server certificate or a private key and the CAcert downloaded using the Harbor instructions. For Harbor to work with vSphere Integrated Containers Engine you need to update standard docker with the Harbor CAcert and deploy a new VCH with the CAcert.

## Update the User Working Machine's Standard Docker with the Harbor CA Certificate

Update the machine with standard Docker so it recognizes the CAcertificate. Docker can look for additional CA certificates outside of the operating system's CAbundle folder if the new CAcertificates are in the correct location. See Verify repository client with certificates.

Create the necessary folder, copy the CAcert file to the folder. Restart Docker, then verify that you can log onto the Harbor server.

**Note** This example has Ubuntu-specific commands.

```
user@Devbox:~/mycerts$ sudo su
[sudo] password for user:
root@Devbox:/home/user/mycerts# mkdir -p /etc/docker/certs.d/<Harbor FQDN>
root@Devbox:/home/user/mycerts# mkdir -p /etc/docker/certs.d/<Harbor IP>
root@Devbox:/home/user/mycerts# cp ca.crt /etc/docker/certs.d/<Harbor FQDN>/
root@Devbox:/home/user/mycerts# cp ca.crt /etc/docker/certs.d/<Harbor IP>/
root@Devbox:/home/user/mycerts# exit
exit
user@Devbox:~/mycerts$ sudo systemctl daemon-reload
user@Devbox:~/mycerts$ sudo systemctl restart docker

user@Devbox:~$ docker logout <Harbor FQDN>
Remove login credentials for <Harbor FQDN>
user@Devbox:~$ docker logout <Harbor IP>
Remove login credentials for <Harbor IP>

user@Devbox:~$ docker login <Harbor FQDN>
Username: user
Password:
Login Succeeded

user@Devbox:~$ docker login <Harbor IP>
Username: user
Password:
Login Succeeded

user@Devbox:~$ docker logout <Harbor FQDN>
Remove login credentials for <Harbor FQDN>

user@Devbox:~$ docker logout <Harbor IP>
Remove login credentials for <Harbor IP>
```

This example creates folders for both FQDN and IP in the docker cert folder and copies the CAcert to both folders. You can then log into Harbor from Docker using both FQDN and IP address.

# Install a VCH with the New CA Certificate

Deploy a VCH and specify the CAcert with `--registry-ca` parameter in vic-machine. This parameter is a list, and you can easily add multiple CAcerts by specifying multiple `--registry-ca` parameters.

For simplicity, this example installs a VCH with the `--no-tls` flag, so you do not need TLS from a docker CLI to the VCH. However, it does not imply that access to Harbor is performed without TLS.

```
./vic-machine-linux create
--target=
--image-store="vsanDatastore"
--name=vic-docker
--user=root
--password=
--compute-resource="/dc1/host/cluster1/Resources"
--bridge-network DPortGroup
--force
--no-tls
--registry-ca=ca.crt

 WARN[2016-11-11T11:46:37-08:00] Configuring without TLS
- all communications will be insecure
 ...

 INFO[2016-11-11T11:47:57-08:00] Installer completed successfully
```

# Installing the vSphere Web Client Plug-In for vSphere Integrated Containers Engine

You can install a plug-in that adds information about virtual container hosts (VCHs) and container VMs in the vSphere Web Client.

You can install the plug-in for vSphere Integrated Containers Engine either on a vCenter Server instance that runs on Windows, or on a vCenter Server Appliance.

**NOTE**: If you deployed the VCH to a vCenter Server 6.5 instance, use the Flash-based vSphere Web Client to view the vSphere Web Client plug-in for vSphere Integrated Containers Engine. vSphere Integrated Containers Engine does not currently provide a plug-in for the new HTML5 vSphere Client.

Information about VCHs and container VMs appears in the **Summary** tabs for those VMs.

- Install the vSphere Integrated Containers Engine Plug-In on vCenter Server For Windows by Using a Web Server
- Install the vSphere Integrated Containers Engine Plug-In on vCenter Server For Windows Without Access to a Web Server
- Install the vSphere Integrated Containers Engine Plug-In on a vCenter Server Appliance by Using a Web Server
- Install the vSphere Integrated Containers Engine Plug-In on a vCenter Server Appliance Without Access to a Web Server
- Verify the Deployment of the vSphere Integrated Containers Engine Plug-In

# Install the vSphere Integrated Containers Engine Plug-In on vCenter Server For Windows by Using a Web Server

If your vCenter Server instance runs on Windows, you can use a Web server to host the vSphere Web Client plug-in for vSphere Integrated Containers Engine.

**Prerequisites**

- You deployed at least one virtual container host (VCH) to a vCenter Server instance that runs on Windows.
- You are running a Web server that your vCenter Server instance can access.
- You must use a Windows system to run the script to install the plug-in on a vCenter Server that runs on Windows. If you used a Linux or Mac OS system to deploy the VCH, download and unpack the vSphere Integrated Containers Engine package on a Windows system. For example, download the package to the system on which vCenter Server is running.
- If you deployed the VCH to a vCenter Server 6.5 instance, use the Flash-based vSphere Web Client to view the vSphere Web Client plug-in for vSphere Integrated Containers Engine. vSphere Integrated Containers Engine does not currently provide a plug-in for the new HTML5 vSphere Client.

**Procedure**

1. On the Windows system on which you have downloaded and unpacked vSphere Integrated Containers Engine, navigate to the folder that contains the `vic-machine` utility and open the `ui` folder.
2. Upload the plug-in bundle to your Web server.

   ```
   unpack_dir\vic\ui\vsphere-client-serenity\com.vmware.vicui.Vicui-version.zip
   ```

3. On the `vic-machine` system, open the `vic_unpack_dir\vic\ui\vCenterForWindows\configs` file in a text editor.
4. Enter the IPv4 address or FQDN of the vCenter Server instance on which to install the plug-in.

   ```
   SET target_vcenter_ip=vcenter_server_address
   ```

5. Enter the path to the folder on your Web server that contains the `com.vmware.vicui.Vicui-version.zip` file.

   ```
   SET vic_ui_host_url="vicui_zip_location"
   ```

6. (Optional) If you used an HTTPS address in `vic_ui_host_url`, provide the SHA-1 thumbprint of the Web server.

   ```
   SET vic_ui_host_thumbprint="thumbprint"
   ```

   **NOTE**: Use colon delimitation in the thumbprint. Do not use space delimitation.
7. Save and close the `configs` file.
8. Open a command prompt, navigate to `vic_unpack_dir\vic\ui\vCenterForWindows`, and run the installer.

   ```
   install.bat
   ```

9. Enter the user name and password for the vCenter Server administrator account.
10. When installation finishes, if you are logged into the vSphere Web Client, log out then log back in again.

**What to Do Next** Check that the deployment has succeeded by following the procedure in Verify the Deployment of the vSphere Integrated Containers Engine Plug-In.

# Install the vSphere Integrated Containers Engine Plug-In on vCenter Server for Windows Without Access to a Web Server

You can install the vSphere Web Client plug-in for vSphere Integrated Containers Engine on a vCenter Server instance for Windows that does not have access to a Web Server.

**Prerequisites**

- You deployed at least one virtual container host (VCH) to a vCenter Server instance that runs on Windows.
- You must use a Windows system to run the script to install the plug-in on a vCenter Server that runs on Windows. If you used a Linux or Mac OS system to deploy the VCH, download and unpack the vSphere Integrated Containers Engine package on a Windows system. For example, download the package to the system on which vCenter Server is running.
- If you deployed the VCH to a vCenter Server 6.5 instance, use the Flash-based vSphere Web Client to view the vSphere Web Client plug-in for vSphere Integrated Containers Engine. vSphere Integrated Containers Engine does not currently provide a plug-in for the new HTML5 vSphere Client.

**Procedure**

1. On the Windows system on which you have downloaded and unpacked vSphere Integrated Containers Engine, navigate to the folder that contains the `vic-machine` utility and open the `ui` folder.
2. Copy the `com.vmware.vicui.Vicui-version` folder into the folder on the vCenter Server system that contains the vSphere Web Client packages.
   - Source location on `vic-machine` system:

     ```
     vic_unpack_dir\vic\ui\vsphere-client-serenity
     ```

   - Destination location on vCenter Server Windows system:

     ```
     instl_dir\vCenterServer\cfg\vsphere-client\vc-packages\vsphere-client-serenity
     ```

     `instl_dir` is the location in which vCenter Server is installed. If the `vc-packages\vsphere-client-serenity` folders do not exist under the `vsphere-client` folder, create them manually.
3. On the `vic-machine` system, open the `vic_unpack_dir\vic\ui\vCenterForWindows\configs` file in a text editor.
4. Enter the IPv4 address or FQDN of the vCenter Server instance on which to install the plug-in.

   ```
   SET target_vcenter_ip=vcenter_server_address
   ```

5. Save and close the `configs` file.
6. Open a command prompt, navigate to `vic_unpack_dir\vic\ui\vCenterForWindows`, and run the installer.

   ```
   install.bat
   ```

7. Enter the user name and password for the vCenter Server administrator account.
8. When installation finishes, if you are logged into the vSphere Web Client, log out then log back in again.

**What to Do Next** Check that the deployment has succeeded by following the procedure in Verify the Deployment of the vSphere Integrated Containers Engine Plug-In.

# Install the vSphere Integrated Containers Engine Plug-In on a vCenter Server Appliance by Using a Web Server

If you are running the vCenter Server Appliance, you can use a Web server to host the vSphere Web Client plug-in for vSphere Integrated Containers Engine.

**Prerequisites**

- You deployed at least one virtual container host (VCH) to a vCenter Server Appliance instance.
- You are running a Web server that the vCenter Server Appliance can access.
- If you deployed the VCH to a vCenter Server 6.5 instance, use the Flash-based vSphere Web Client to view the vSphere Web Client plug-in for vSphere Integrated Containers Engine. vSphere Integrated Containers Engine does not currently provide a plug-in for the new HTML5 vSphere Client.

**Procedure**

1. On the system on which you run `vic-machine`, navigate to the folder that contains the `vic-machine` utility and open the `ui` folder.
2. Upload the plug-in bundle to your Web server.

   ```
   vic_unpack_dir/vic/ui/vsphere-client-serenity/com.vmware.vicui.Vicui-version.zip
   ```

3. Open the `vic_unpack_dir/vic/ui/VCSA/configs` file in a text editor.
4. Enter the IPv4 address or FQDN of the vCenter Server instance on which to install the plug-in.

   ```
   VCENTER_IP="vcenter_server_address"
   ```

5. Enter the path to the folder on your Web server that contains the `com.vmware.vicui.Vicui-version.zip` file.

   ```
   VIC_UI_HOST_URL="vicui_zip_location"
   ```

6. (Optional) If you used an HTTPS address in `VIC_UI_HOST_URL`, provide the SHA-1 thumbprint of the Web server.

   ```
   VIC_UI_HOST_THUMBPRINT="thumbprint"
   ```

   **NOTE**: Use colon delimitation in the thumbprint. Do not use space delimitation.
7. Save and close the `configs` file.
8. (Optional) If you run `vic-machine` on a Windows system, open the `vic_unpack_dir/vic/ui/VCSA/install.sh` file in a text editor and point `PLUGIN_MANAGER_BIN` to the Windows UI executable.

   Before:

   ```
   if [[ $(echo $OS | grep -i "darwin") ]] ; then
      PLUGIN_MANAGER_BIN="../../vic-ui-darwin"
     else
      PLUGIN_MANAGER_BIN="../../vic-ui-linux"
   ```

   After:

```
if [[ $(echo $OS | grep -i "darwin") ]] ; then
   PLUGIN_MANAGER_BIN="../../vic-ui-darwin"
  else
   PLUGIN_MANAGER_BIN="../../vic-ui-windows"
```

9. Open a command prompt, navigate to `vic_unpack_dir/vic/ui/VCSA`, and run the installer.

   ```
   ./install.sh
   ```

   - Make sure that `install.sh` is executable by running `chmod` before you run it.
   - On Windows systems, run `install.sh` in a UNIX shell that supports SSH and SCP, for example Cygwyn or Git Bash. Do not use Windows 10 native Bash.
10. Enter the user name and password for the vCenter Server administrator account.
11. When installation finishes, if you are logged into the vSphere Web Client, log out then log back in again.

**What to Do Next** Check that the deployment has succeeded by following the procedure in Verify the Deployment of the vSphere Integrated Containers Engine Plug-In.

# Install the vSphere Integrated Containers Engine Plug-In on a vCenter Server Appliance Without Access to a Web Server

If you are running the vCenter Server Appliance and you do not have access to a Web server, you can manually install the vSphere Web Client plug-in for vSphere Integrated Containers Engine.

**Prerequisites**

- You deployed at least one virtual container host (VCH) to a vCenter Server Appliance instance.
- If you deployed the VCH to a vCenter Server 6.5 instance, use the Flash-based vSphere Web Client to view the vSphere Web Client plug-in for vSphere Integrated Containers Engine. vSphere Integrated Containers Engine does not currently provide a plug-in for the new HTML5 vSphere Client.

**Procedure**

1. On the system on which you run `vic-machine`, open the `vic_unpack_dir/vic/ui/VCSA/configs` file in a text editor.
2. Enter the IPv4 address or FQDN of the vCenter Server instance on which to install the plug-in.

   ```
   VCENTER_IP="vcenter_server_address"
   ```

3. Save and close the `configs` file.
4. (Optional) If you run `vic-machine` on a Windows system, open the `vic_unpack_dir/vic/ui/VCSA/install.sh` file in a text editor and point `PLUGIN_MANAGER_BIN` to the Windows UI executable.

   Before:

   ```
   if [[ $(echo $OS | grep -i "darwin") ]] ; then
       PLUGIN_MANAGER_BIN="../../vic-ui-darwin"
     else
       PLUGIN_MANAGER_BIN="../../vic-ui-linux"
   ```

   After:

   ```
   if [[ $(echo $OS | grep -i "darwin") ]] ; then
       PLUGIN_MANAGER_BIN="../../vic-ui-darwin"
     else
       PLUGIN_MANAGER_BIN="../../vic-ui-windows"
   ```

5. Open a command prompt, navigate to `vic_unpack_dir/vic/ui/VCSA`, and run the installer.

   ```
   ./install.sh
   ```

   - Make sure that `install.sh` is executable by running `chmod` before you run it.
   - On Windows systems, run `install.sh` in a UNIX shell that supports SSH and SCP, for example Cygwyn or Git Bash. Do not use Windows 10 native Bash.
6. Enter the user name and password for the vCenter Server administrator account.
7. Enter the root password for the vCenter Server Appliance.

   The installer requires the root password of the vCenter Server Appliance three times:

    - Once to check whether the Bash shell is enabled on the vCenter Server Appliance. If the Bash shell is not enabled, the installation fails and the installer provides remedial instructions.
    - Once to copy the files to the appliance over SSH.
    - Once to set the correct ownership on the files and folders.
8. When installation finishes, if you are logged into the vSphere Web Client, log out then log back in again.

**What to Do Next** Check that the deployment has succeeded by following the procedure in Verify the Deployment of the vSphere Integrated Containers Engine Plug-In.

# Verify the Deployment of the vSphere Integrated Containers Engine Plug-In

After you have installed the vSphere Web Client plug-in for vSphere Integrated Containers Engine, verify the deployment of the plug-in in the vSphere Web Client.

**Prerequisites**

- You deployed a virtual container host (VCH).
- You installed the vSphere Web Client plug-in for vSphere Integrated Containers Engine.
- You logged out of the vSphere Web Client after deploying the plug-in, and logged back in.
- If you deployed the VCH to a vCenter Server 6.5 instance, use the Flash-based vSphere Web Client to view the vSphere Web Client plug-in for vSphere Integrated Containers Engine. vSphere Integrated Containers Engine does not currently provide a plug-in for the new HTML5 vSphere Client.

**Procedure**

1. In the vSphere Web Client Home page, select **Hosts and Clusters**.
2. Expand the hierarchy of vCenter Server objects to navigate to the VCH vApp.
3. Expand the VCH vApp and select the VCH endpoint VM.
4. Click the **Summary** tab for the VCH endpoint VM and scroll down to the VCH portlet.

**Result**

Information about the VCH appears in the VCH portlet in the **Summary** tab:

- The address of the Docker API endpoint for this VCH
- Alink to the vic-admin portal for the VCH, from which you can obtain health information and download log bundles for the VCH.

**What to Do Next**

If the VCH portlet still does not appear in the **Summary** tab for the VCH endpoint VM, restart the vSphere Web Client service. For instructions about how to restart the vSphere Web Client service, see vSphere Integrated Containers Engine Plug-In Does Not Appear in the vSphere Web Client.

# Troubleshooting vSphere Integrated Containers Engine Installation

This information provides solutions for common problems that you might encounter when deploying virtual container hosts (VCHs).

# VCH Deployment Fails with a Certificate Verification Error

When you use `vic-machine create` to deploy a virtual container host (VCH), the installation fails with a certificate verification error, noting that it `failed to create validator`.

## Problem

Deployment of the VCH fails during the validation of the configuration that you provided:

```
Failed to verify certificate for target=vcenter_server_or_esxi_host
(thumbprint=vc_or_esxi_cert_thumbprint)
Create cannot continue: failed to create validator
vic-machine-platform.exe failed: x509: certificate signed by unknown authority
```

## Cause

The certificate on the vCenter Server or ESXi host that you specified in the `--target` option cannot be validated on the client system.

## Solution

If the certificate was signed by a certificate authority (CA), add that CA to the trusted roots for the client system.

If the CA should not be generally trusted, or the certificate is self-signed:

- If the server is trusted and you did not specify the certificate thumbprint when you ran `vic-machine create`, specify the `--thumbprint` option, using the thumbprint from the error message.
- If the thumbprint that you specified in `--thumbprint` does not match the server certificate reported in the error message:
  1. Remove the thumbprint from the `vic-machine create` command. **WARNING:** A thumbprint mismatch could mean the server you have connected to is not the intended target and might have be spoofed.
  2. Validate that the change in server certificate is legitimate
  3. Re-run `vic-machine create`, specifying the new thumbprint in the `--thumbprint` option.

# VCH Deployment Fails with Missing Common Name Error Even When TLS Options Are Specified Correctly

If you deploy a virtual container host (VCH) and you have specified one of the `vic-machine create --tls-cname`, `--no-tlsverify`, or `--no-tls` options, or you set a static IP address on the client network, the deployment fails with an error about the certificate Common Name being missing.

## Problem

Deployment fails during the validation of the configuration that you provided, even if you did specify a TLS option or you set a static IP address on the client network. For example:

```
$ vic-machine-windows create
--target 'Administrator@vsphere.local:password'@vcenter_server
--bridge-network vic bridge --no-tls
### Installing VCH ####
[...]
Common Name must be provided when generating certificates for client
authentication:
[...]
Create cannot continue: unable to generate certificates
--------------------
vic-machine-windows.exe failed: provide Common Name for server certificate
```

If you include a TLS option at the beginning of the `vic-machine create` command rather than the end, you see the following error:

```
$ vic-machine-windows create
--target 'Administrator@vsphere.local:password'@vcenter_server
 --no-tls
--bridge-network vic bridge
### Installing VCH ####
[...]
Unknown argument: bridge
--------------------
vic-machine-windows.exe failed: invalid CLI arguments
```

## Cause

String values that you provided for certain options contain spaces or special characters that you did not escape with quotations marks. The `vic-machine create` input validator validates the arguments that you provide only as far as the argument that includes the space or special character. If you specify the TLS option before the argument with the space or special character, `vic-machine create` throws the correct error message. However, if you specify the TLS option after the argument that includes the space or special character, the `vic-machine create` validator stops before it reaches the TLS option, and so throws the error about the missing Common Name.

# Solution

Wrap any arguments that contain spaces or special characters in single quotation marks (') on Mac OS and Linux and in double quotation marks (") on Windows.

Option arguments that might require quotation marks include the following:

- User names and passwords in `--target`, or in `--user` and `--password`
- Datacenter names in `--target`
- VCH names in `--name`
- Datastore names and paths in `--image-store` and `--volume-store`
- Network and port group names in all networking options.
- Cluster and resource pool names in `--compute-resource`
- Folder names in the paths for `--cert-path`, `--cert`, `--key`, `--appliance-iso`, and `--bootstrap-iso`

For information about when to use quotation marks for different options, see the descriptions of those options in VCH Deployment Options.

# VCH Deployment Fails with Firewall Validation Error

When you use `vic-machine create` to deploy a virtual container host (VCH), deployment fails because firewall port 2377 is not open on the target ESXi host or hosts.

## Problem

Deployment fails with a firewall error during the validation phase:

```
Firewall must permit dst 2377/tcp outbound to the VCH management interface
```

## Cause

ESXi hosts communicate with the VCHs through port 2377 via Serial Over LAN. For deployment of a VCH to succeed, port 2377 must be open for outgoing connections on all ESXi hosts before you run `vic-machine create`. Opening port 2377 for outgoing connections on ESXi hosts opens port 2377 for inbound connections on the VCHs.

## Solution

Set a firewall ruleset on the ESXi host or hosts. In test environments, you can disable the firewall on the hosts.

### Set a Firewall Ruleset Manually

In production environments, if you are deploying to a standalone ESXi host, set a firewall ruleset on that ESXi host. If you are deploying to a cluster, set the firewall ruleset on all of the ESXi hosts in the cluster.

**IMPORTANT**: Firewall rulesets that you set manually are not persistent. If you reboot the ESXi hosts, any firewall rules that you set are lost. You must recreate firewall rules each time you reboot a host.

1. Use SSH to log in to each ESXi host as `root` user.
2. Follow the instructions in VMware KB 2008226 to add the following rule after the last rule in the file `/etc/vmware/firewall/service.xml`.

```
<service id='id_number'>
<id>vicoutgoing</id>
<rule id='0000'>
 <direction>outbound</direction>
 <protocol>tcp</protocol>
 <port type='dst'>2377</port>
</rule>
<enabled>true</enabled>
<required>true</required>
</service>
```

In this example, *id_number* is the number of the preceding ruleset in `service.xml`, incremented by 1.

### Disable the Firewall

In test environments, you can disable the firewalls on the ESXi hosts instead of opening port 2377.

1. Use SSH to log in to each ESXi host as `root` user.
2. Run the following command:

```
$ esxcli network firewall set --enabled false
```

# VCH Deployment Fails with Certificate `cname` Mismatch

When you use `vic-machine create` to deploy a virtual container host (VCH), the deployment fails with an error about the certificate `cname` value.

## Problem

Deployment fails during the validation of the configuration that you provided:

```
Provided cname does not match that in existing server certificate: cname
Unable to load certificates: cname option doesn't match existing server certificate
in certificate path path_to_certificate
```

## Cause

`vic-machine create` attempts to re-use certificates that it finds in `--cert-path`. The default value of `--cert-path` derives from the value that you specify in `--name`. If you are deploying a VCH from the same location and with the same name as a previous VCH, `vic-machine create` reuses the old certificates. This behavior is intentional, to allow you to easily redeploy a VCH without requiring you to re-issue client certificates to users.

Before reusing the existing certificates, `vic-machine` confirms that the existing certificate is valid given the options supplied for the new deployment. The options that influence this in order of priority are:

- `--tls-cname` if specified, or
- `--client-ip-address`, or
- `--public-ip-address` if the client and public network roles share an interface.

The error message means that the existing certificate has a Common Name attribute that differs from the value derived from the options detailed above.

## Solution

- To reuse the certificates directly, change `--tls-cname`, `--client-ip-address`, or `--public-ip-address` to match the Common Name in the existing certificate.
- If you want to reuse the Certificate Authority so that client certificates remain valid, but you need to provide a different IP address:
  1. Manually generate the server certificates by using `openssl`, signing them with the existing CA.
  2. Use the `--cert` and `--key` options to pass the newly generated certificates to `vic-machine create`.
- If you do not want to reuse the certificates, choose one of the following options:
  - Change the location from which you run `vic-machine`. This alters the default `--cert-path`.
  - Change the value of `--name`. This alters the default `--cert-path`.
  - Specify `--cert-path` explicitly.
  - Delete the existing certificates from `--cert-path`

# vSphere Integrated Containers Engine Plug-In Does Not Appear in the vSphere Web Client

After you have installed the vSphere Web Client plug-in for vSphere Integrated Containers Engine, the plug-in does not appear in the vSphere Web Client.

## Problem

The UI plug-in installer reported success, but the virtual container host (VCH) portlet does not appear in the **Summary** tab for the VCH endpoint VM. Logging out of the vSphere Web Client and logging back in again does not resolve the issue.

## Cause

If a previous attempt at installing the vSphere Integrated Containers Engine plug-in failed, the failed installation state is retained in the vSphere Web Client cache.

## Solution

Restart the vSphere Web Client service.

### vCenter Server on Windows

1. Open Server Manager on the Windows system on which vCenter Server is running.
2. Select **Configuration** > **Services**.
3. Select **VMware vSphere Web Client** and click **Restart**.

### vCenter Server Appliance

1. Use SSH to log in to the vCenter Server Appliance as root.
2. Stop the vSphere Web Client service by running one of the following commands.
   - vCenter Server 6.0:

     ```
     service vsphere-client stop
     ```

   - vCenter Server 6.5:

     ```
     service-control --stop vsphere-client
     ```

3. Restart the vSphere Web Client service by running one of the following commands.
   - vCenter Server 6.0:

     ```
     service vsphere-client start
     ```

   - vCenter Server 6.5:

     ```
     service-control --start vsphere-client
     ```

# Docker Commands Fail with a Docker API Version Error

After a successful deployment of a vSphere Integrated Containers Engine virtual container host (VCH), attempting to run a Docker command fails with a Docker version error.

## Problem

When you attempt to run a Docker command from a Docker client that is connecting to a VCH, the command fails with the following error:

```
Error response from daemon: client is newer than server
(client API version: x.xx, server API version: 1.23)
```

## Cause

This version of vSphere Integrated Containers Engine supports Docker 1.11, that includes version 1.23 of the Docker API. You are using a more recent version of the Docker client, that includes a version of the Docker API that is incompatible.

## Solution

1. Open a terminal on the system on which you run the Docker client.
2. Set the Docker client API to the same version as the one that is used by vSphere Integrated Containers Engine.

   ```
   export DOCKER_API_VERSION=1.23
   ```

3. Check that your Docker client can now connect to the VCH by running a Docker command.
   - With TLS authentication:

     ```
     docker -H virtual_container_host_address:2376 --tls info
     ```

   - Without TLS authentication:

     ```
     docker -H virtual_container_host_address:2375 info
     ```

   The `docker info` command should succeed and you should see information about the VCH.

# Send Documentation Feedback

Help us to improve the vSphere Integrated Containers documentation.

- Send doc feedback to VMware
- Submit a doc issue in Github