# Cluster Server Agent for Docker Installation and Configuration Guide

Linux

8.0

March 2023

VERITAS™

# Cluster Server Agent for Docker Installation and Configuration Guide



Agent Version: 8.0

Document version: 8.0 Rev 1

## Legal Notice

Veritas Technologies LLC
500 E Middlefield Road
Mountain View, CA 94043

http://www.veritas.com

# Technical Support

Technical Support maintains support centers globally. Technical Support's primary role is to respond to specific queries about product features and functionality. The Technical Support group also creates content for our online Knowledge Base. The Technical Support group works collaboratively with the other functional areas within the company to answer your questions in a timely fashion.

Our support offerings include the following:

- A range of support options that give you the flexibility to select the right amount of service for any size organization
- Telephone and/or Web-based support that provides rapid response and up-to-the-minute information
- Upgrade assurance that delivers software upgrades
- Global support purchased on a regional business hours or 24 hours a day, 7 days a week basis
- Premium service offerings that include Account Management Services

For information about our support offerings, you can visit our website at the following URL:

www.veritas.com/support

All support services will be delivered in accordance with your support agreement and the then-current enterprise technical support policy.

## Contacting Technical Support

Customers with a current support agreement may access Technical Support information at the following URL:

www.veritas.com/support

Before contacting Technical Support, make sure you have satisfied the system requirements that are listed in your product documentation. Also, you should be at the computer on which the problem occurred, in case it is necessary to replicate the problem.

When you contact Technical Support, please have the following information available:

- Product release level
- Hardware information
- Available memory, disk space, and NIC information

- Operating system
- Version and patch level
- Network topology
- Router, gateway, and IP address information
- Problem description:
  - Error messages and log files
  - Troubleshooting that was performed before contacting Technical Support
  - Recent software configuration changes and network changes

## Licensing and registration

If your product requires registration or a license key, access our technical support Web page at the following URL:

www.veritas.com/support

## Customer service

Customer service information is available at the following URL:

www.veritas.com/support

Customer Service is available to assist with non-technical questions, such as the following types of issues:

- Questions regarding product licensing or serialization
- Product registration updates, such as address or name changes
- General product information (features, language availability, local dealers)
- Latest information about product updates and upgrades
- Information about upgrade assurance and support contracts
- Advice about technical support options
- Nontechnical presales questions
- Issues that are related to CD-ROMs, DVDs, or manuals

## Support agreement resources

If you want to contact us regarding an existing support agreement, please contact the support agreement administration team for your region as follows:

| | |
|---|---|
| Worldwide (except Japan) | CustomerCare@veritas.com |
| Japan | CustomerCare_Japan@veritas.com |

# Contents

Chapter 1

# Introducing the agents for Docker

This chapter includes the following topics:



## About the Cluster Server agents for Docker Daemon and Docker Container

The Cluster Server (VCS) agents monitor specific resources within an enterprise application. They determine the status of resources and start or stop them according to external events.

The Cluster Server agents for Docker Daemon and Docker Container provides high availability for Docker Daemon and Docker Containers or Podman Daemon and Podman container in cluster. The Docker daemon provides the base infrastructure for creating and hosting Docker containers. The Docker Container agent monitors the Docker Container instances while they are online and offline. If the system fails, the agent detects failure and takes the container instances offline. Cluster Server

initiates failover to another system in the cluster and the agent brings the container instances online.

## Supported software

For information on the software versions that the Cluster Server agents for Docker support, see the Veritas Services and Operations Readiness Tools (SORT) site: http://sort.veritas.com/agents.

# How the agents makes Docker highly available

The Cluster Server agent for Docker Daemon continuously monitors the Docker Daemon process to check the status of the Docker Service. The agent for Docker Daemon is Intelligent Monitoring Framework (IMF) aware and uses Asynchronous Monitoring Framework (AMF) kernel driver for IMF notification.

The Cluster Server agent for Docker Container monitors the configured container using the `docker inspect` command. The agent for Docker Container is IMF aware and uses AMF kernel driver for IMF notification.

# Features of the agents

The Cluster Server agents for Docker Daemon and Docker Container has the following features:

- Support for First Failure Data Capture (FFDC)
  In case of a fault, the Docker Daemon and Docker Container agents generate debug logs that enable troubleshooting of the fault.
- Support for intelligent resource monitoring and poll-based monitoring
  The Docker Daemon and Docker Container agents support the VCS Intelligent Monitoring Framework feature. IMF allows the agent to register the resources to be monitored with the IMF notification module to receive immediate notification of resource state changes without having to periodically poll the resources.
- Support for Docker container failover
  The Docker Container agent supports Docker container failover.
  Data volumes attached to the containers can also be failed over, if they are configured on the shared storage that is accessible to all the nodes.

# Docker Daemon agent functions

The agent consists of resource type declarations and agent executables. The agent executables are organized into online, offline, monitor, and clean functions.

## Online

The online function performs the following tasks:

- Verifies that the Docker daemon is not already stopped. If the daemon is already stopped, it exits immediately.
- Verifies that the required attributes are set correctly.
- Attempts to start the specified Docker daemon using the start command specified as part of the DaemonStartCommand attribute. The default start command is `/usr/sbin/service docker start`.

## Offline

The offline function performs the following tasks:

- Verifies that the Docker daemon is not already stopped. If the daemon is already stopped, it exits immediately.
- Verifies that the required attributes are set correctly.
- Attempts to start the specified Docker daemon using the stop command specified as part of the DaemonStopCommand attribute. The default start command is `/usr/sbin/service docker stop`.

## Monitor

The monitor function monitors the state of the Docker daemon on all nodes in the cluster. The function performs the following tasks:

- The first-level check searches for the Docker daemon process. The Docker daemon process pattern can be specified as part of the DaemonProcPattern attribute. Perl regular expression can be specified to match the Docker daemon process pattern.
- The second-level check for the Docker daemon runs the `service docker status` command. If the Docker daemon is in the ‘Runnning’ state, the agent reports the resource as online.

## Clean

The clean function performs the following tasks:

- If a graceful shutdown was not attempted earlier by the agent, the agent attempts to stop the Docker daemon using the graceful shutdown command.
- If the Docker daemon process is still running, even after a graceful shutdown attempt, the agent kills the Docker daemon process.

### imf_init

This function initializes the Docker Daemon agent to interface with the AMF kernel driver, which is the IMF notification module for the agent for Docker Daemon. This function runs when the agent starts up.

### imf_getnotification

This function gets notifications about resource state changes. This function runs after the agent initializes with the AMF kernel module. This function continuously waits for notification and takes action on the resource upon notification.

### imf_register

This function registers or unregisters resource entities with the AMF kernel module. This function runs for each resource after the resource goes into a steady online or offline state.

# Docker Container agent functions

The agent consists of resource type declarations and agent executables. The agent executables are organized into online, offline, monitor, and clean functions.

### Online

The online function performs the following tasks:

- Verifies that the container with the name specified in the ContainerName attribute is not already running. If the container with the name specified in the ContainerName attribute is running, it exits immediately and resource is reported online.
- If a container with the given name does not exists, the online entry point creates a container using the ImageName, DockerRunOptions, and ContainerInitCommand attributes.
- If a container with the given name exists, the online operation attempts to start the Docker Container using the `docker start` command.

## Offline

The offline function performs the following tasks:

- Verifies that the container is not already stopped. If the container is not running, the agent exits immediately and reports the resource as offline.
- Attempts to stop the specified Docker Container using the `docker stop --time=<OfflineTimeout>` command. If the application does not handle the SIGTERM signal, the stop command will wait till the OfflineTimeout seconds before killing the container process forcefully using the SIGKILL signal.

**Note:** OfflineTimeout is a type-level attribute for the Docker Container agent.

## Monitor

The monitor function monitors the state of the Docker container on all nodes in the cluster. The function performs the following tasks:

- If a container with a given name does not exist, the monitor entry points report the container as offline.
- If a container with a given name exists, the monitor entry point runs the `docker inspect` command to get the state of container.
- Checks for the state of the container. If the state of the container is paused or restarting, it reports the state as unknown.
- Reports the state of the container as online, if the container is in the running state.

## Clean

The clean function performs the following tasks:

- Attempts to gracefully shut down the Docker container using the `docker stop` command.
- If an attempt to stop the container gracefully fails, it kills the container PID that is retrieved using the `docker inspect` command on the container.

  .

### imf_init

This function initializes the Docker Container agent to interface with the AMF kernel driver, which is the IMF notification module for the agent for Docker Container. This function runs when the agent starts up.

### imf_getnotification

This function gets notifications about resource state changes. This function runs after the agent initializes with the AMF kernel module. This function continuously waits for notification and takes action on the resource upon notification.

### imf_register

This function registers or unregisters resource entities with the AMF kernel module. This function runs for each resource after the resource goes into a steady online or offline state.

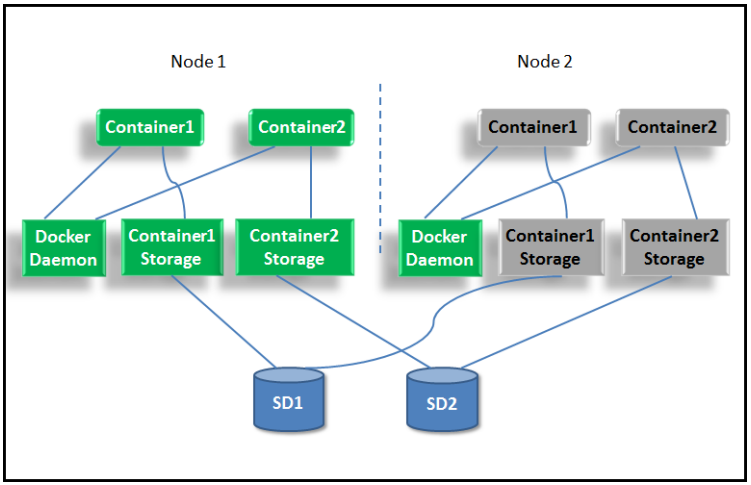
# Typical Docker configuration in a VCS cluster

A typical Docker configuration in a VCS cluster has the following characteristics:

- VCS is installed and configured.
- The Docker binaries are installed locally on all nodes.
- The Docker daemon data directory is on the shared storage.
- The Cluster Server agents for Docker are installed on all nodes.

Figure 1-1 depicts a typical Docker configuration.

**Figure 1-1** Docker Configuration




# Setting up Docker in a VCS cluster

Follow the steps below to set up Docker in a cluster:

- Set up a VCS cluster.
- Install the Cluster Server agent for Docker Daemon and Docker Container.
  See “Installing the agents ” on page 21.
- Configure the Docker Daemon and Docker Container agents for high availability.
- Configure the service groups for Docker Daemon and Docker Container.
  See “About configuring service groups for Docker” on page 37.

Chapter 2

# Introducing Docker and Docker Daemon process

This chapter includes the following topics:



## About Docker

Docker is an open platform that allows programmers and system administrators to develop, ship, and run distributed applications. It allows programmers to build applications with Docker containers using any language and tools. Docker provides instant application portability. The applications can be assembled quickly, shipped faster, and scaled on thousands of hosts. The Docker applications can also be moved between data centers and clouds and updated with zero downtime.

Docker consists of the following components:

- Docker daemon - A portable and lightweight application runtime and packaging tool. The Docker daemon manages containers and a client, which controls the daemon.
- Docker containers - A container that comprises the applications and its dependencies. Each container is created from a Docker image. It runs as an isolated process in the user space on the host operating system and shares the kernels with other containers.
- Docker images - A read-only template that is used to create Docker containers. Docker provides a simple way to create, update, and download images from the Docker registries.

## Docker Daemon process

The Docker Daemon agent uses the Docker daemon process. The Docker daemon process creates, starts, stops, modifies, and deletes Docker containers in the cluster.

# Recommendations for configuring Docker

This section lists the recommendations and guidelines for configuring Docker Container resources in a VCS environment.

- Specify all attribute values to ensure high availability of the containers.
- Specify long running processes before creating containers using the `docker run` command.
- Ensure that the Docker images are pulled locally on all of the cluster nodes. If the container does not exist, the agent creates container during the online operation. If image is not available, it would be pulled from the Docker registry, which may consume more time. The OnlineTimeOut attribute value should be set appropriately.
- If the application running in the Docker container needs to update data persistently, ensure that the data resides on the shared storage. The container data volume should be shared across the cluster nodes.

Chapter 3

# Installing and removing the agents for Docker

This chapter includes the following topics:



## Before you install the Cluster Server agents for Docker

You must install the Cluster Server agents for Docker on all the systems that will host Docker service groups.

Ensure that you meet the following prerequisites to install the agents for Docker.

- Install and configure Cluster Server.
  For more information about installing and configuring Cluster Server, refer to the Cluster Server installation and configuration guides.
- Install Docker binaries on each node in the cluster from https://docs.docker.com/installation/rhel/.

# About the ACC library

The operations of a VCS agent depend on a set of Perl modules known as the ACC library. The library must be installed on each system in the cluster that runs the agent. The ACC library contains common, reusable functions that perform tasks, such as process identification, logging, and system calls.

Instructions to install or remove the ACC library on a single system in the cluster are given in the following sections. The instructions assume that the ACCLib tar file has already been extracted.

---

**Note:** The LogDbg attribute should be used to enable debug logs for the ACCLib-based agents when the ACCLib version is 6.2.0.0 or later and VCS version is 6.2 or later.

---

# Installing the ACC library

Install the ACC library on each node in the cluster that runs the agent.

**To install the ACC library**

1. Log in as a superuser.
2. Download the ACC library from the Veritas Services and Operations Readiness Tools (SORT) site (http://sort.veritas.com/agents).

   You can download either the complete Agent Pack tar file or the individual ACCLib tar file.
3. If you downloaded the complete Agent Pack tar file, navigate to the Linux directory.

   Linux
   ```
   cd1/linux/generic/vcs/application/acc_library/
   version_library/rpms
   ```
4. If you downloaded the individual ACCLib tar file, navigate to the rpms directory.
5. Install the package. Enter **Yes** if asked to confirm overwriting of files in the existing package.

   Linux
   ```
   # rpm -ivh \
   VRTSacclib-VersionNumber-GENERIC.noarch.rpm
   ```

# Installing the agents

Install the agents for Docker on each node in the cluster.

**To install the agent in a VCS environment**

1. Download the agent from the Veritas Services and Operations Readiness Tools (SORT) site: http://sort.veritas.com/agents.

   You can download either the complete Agent Pack tar file or an individual agent tar file.

2. Uncompress the file to a temporary location, say `/tmp`.

3. If you downloaded the complete Agent Pack tar file, navigate to the Linux directory.

   | | |
   |---|---|
   | Linux | `cd1/linux/generic/vcs/application/docker_agent/`*`vcs_version`*`/`*`version_agent`*`/rpms` |

4. Log in as a superuser.

5. Install the package.

   | | |
   |---|---|
   | Linux | `# rpm -ihv \`<br>`VRTSvcsdocker-6.2.0.0-GENERIC.noarch.rpm` |

   The agents for the Docker Daemon and the Docker Container are installed.

6. After installing the agent package, you must import the agent type configuration file. See “Importing the agent types files in a VCS environment” on page 24.

# Removing the agents

You must uninstall the agents for Docker from a cluster while the cluster is active.

**To uninstall the agents in a VCS environment**

1. Log in as a superuser.

2. Set the cluster configuration mode to read-write by running the following command from any node in the cluster:

   ```
   # haconf -makerw
   ```

3 Remove all Docker resources from the cluster. Run the following commands to verify that all resources have been removed:

```
# hares -list Type=DockerDaemon
# hares -list Type=DockerContainer
```

4 Remove the agent type from the cluster configuration by running the following commands from any node in the cluster:

```
# hatype -delete DockerDaemon
# hatype -delete DockerContainer
```

Removing the agent's type file from the cluster removes the include statement for the agent from the `main.cf` file, but the agent's type file is not removed from the cluster configuration directory. You can remove the agent's type file later from the cluster configuration directory.

5 Save these changes. Then set the cluster configuration mode to read-only by running the following command from any node in the cluster:

```
# haconf -dump -makero
```

6 Use the native software management program to remove the agents for Docker from each node in the cluster.

Run the following command to uninstall the agents:

Linux `# rpm -e VRTSvcsdocker`

# Removing the ACC library

Perform the following steps to remove the ACC library.

**To remove the ACC library**

1 Ensure that all agents that use ACC library are removed.

2 Run the following command to remove the ACC library package:

Linux `# rpm -e VRTSacclib`

Chapter 4

# Configuring the agents for Docker

This chapter includes the following topics:



## About configuring the Cluster Server agents for Docker

After installing the Cluster Server agents for Docker, you must import the agent type configuration file. After importing this file, review the attributes table that describes the resource type and its attributes, and then create and configure Docker resources.

To view the sample agent type definition and service groups configuration:

See “About sample configurations for the agents for Docker” on page 47.

# Importing the agent types files in a VCS environment

To use the agents for Docker, you must import the agent types file into the cluster.

You can import the agent types file using the Cluster Manager (Java Console) or using the command line interface (CLI).

**To import the agent types file using Cluster Manager (Java Console)**

1. Start the Cluster Manager and connect to the cluster on which the agent is installed.
2. Click **File > Import Types**.
3. In the **Import Types** dialog box, select the following file:

   ```
   /etc/VRTSagents/ha/conf/DockerDaemon/DockerTypes.cf
   ```

4. Click **Import**.
5. Save the VCS configuration.

   The Docker agent types file is now imported to the VCS engine.

   You can now create Docker resources. For additional information about using the Cluster Manager (Java Console), refer to the *Cluster Server Administrator's Guide*.

**To import the agent types file using the CLI:**

1. Log on to any one of the systems in the cluster as the superuser.
2. Run the following command:

   ```
   /etc/VRTSagents/ha/conf/DockerDaemon/DockerTypes.cmd
   ```

   The Docker agent type is now imported to the VCS engine.

   You can now create Docker resources.

# Docker Daemon agent attributes

**Table 4-1** Required attributes for configuring the Docker Daemon agent

| Required attributes | Description |
|---|---|
| DaemonStartCommand | The command and the arguments that start the Docker daemon. Ensure that the Docker daemon starts outside the VCS control using the specified command.<br>Type and dimension: string-scalar<br>Default: `/usr/sbin/service docker start`<br>Example 1: `/usr/sbin/service docker start`<br>Example 2: `/usr/bin/systemctl start docker`<br>Example 3: `/usr/bin/docker -d --selinux-enabled -g /cfs_mount &`<br>Example 4: `/usr/sbin/service podman start`<br>Example5: `/usr/bin/systemctl start podman` |
| DaemonProcPattern | The process pattern that uniquely identifies the Docker daemon process. To identify the Docker daemon process, the agent uses this value as a filter in the output of the `/bin/ps ww -eo uid,pid,ppid,args` command.<br>You can specify the complete process pattern as displayed in the `ps -ef` command. Alternatively, you can specify the regular expression to uniquely identify the Docker daemon process.<br>Ensure that the agent is able to uniquely identify the Docker daemon process using this pattern. Additionally, you can specify the hostname or port details in the DaemonProcPattern attribute to uniquely identify the Docker daemon process.<br>Type and dimension: string-scalar<br>Default: `\\bdocker .*-d`<br>Example: `\\bdocker -d --selinux-enabled -g /cfs_mount` |

**Table 4-2** Optional attributes for configuring the Docker Daemon agent

| Optional attribute | Description |
|---|---|
| ResLogLevel | The logging detail performed by the agent for the resource. Valid values are:<br>■ ERROR: Only logs error messages.<br>■ WARN: Logs above plus warning messages.<br>■ INFO: Logs above plus informational messages.<br>■ TRACE: Logs above plus trace messages. TRACE is very verbose and should only be used during initial configuration or for troubleshooting and diagnostic operations.<br>Type and dimension: string-scalar<br>Default: INFO<br>Example: TRACE<br>**Note:** The use of the ResLogLevel attribute is deprecated from VCS version 6.2 onwards. You must use the LogDbg attribute instead of the ResLogLevel attribute to enable debug logs for the ACCLib-based agents, when the ACCLib version is 6.2.0.0 or later. The agent captures the first failure data of the unexpected events and automatically logs debug messages in their respective agent log files. |
| LogDbg | For ACCLib-based agents, you must use the LogDbg resource type attribute to enable the debug logs when the ACCLib version is 6.2.0.0 or later and the VCS version is 6.2 or later.<br>Set the LogDbg attribute to DBG_5 to enable debug logs for the ACCLib-based agent. By default, setting the LogDbg attribute to DBG_5 enables debug logs for all Docker resources in the cluster. If debug logs must be enabled for a specific Docker resource, override the LogDbg attribute.<br>For more information on how to enable debug logs, See "To enable debug logs for all resources of type Docker" on page 46.<br>Type and dimension: keylist<br>Default: No default value<br>For more information on how to use the LogDbg attribute, refer to the *Cluster Server Administrator's Guide*. |

**Table 4-2** Optional attributes for configuring the Docker Daemon agent *(continued)*

| Optional attribute | Description |
|---|---|
| MonitorProgram | The absolute path name of an external, user-supplied monitor executable.<br><br>When this attribute is specified, the monitor entry point executes the MonitorProgram file to perform an additional instance state check. There are no restrictions on the actions that the external monitor program performs to determine the state of the Docker Daemon instance. However, the external monitor program must return one of the following integer values:<br><br>■ 0: instance is online<br>■ 110: instance is online<br>■ 100: instance is offline<br>■ 1: instance is offline<br>■ Any other value: instance is unknown.<br><br>Veritas recommends storing the external monitor utility on the shared disk directory to ensure that the file is always available on the online system.<br><br>Arguments are supported.<br><br>Type and dimension: string-scalar<br><br>Default: No default value<br><br>Example 1: `$ServerRoot/bin/myMonitor.pl`<br><br>Example 2: `$ServerRoot/bin/myMonitor.sh arg1 arg2` |
| IMF | Determines if the agent must perform the intelligent resource monitoring. You can also override the value of this attribute at the resource level. |
| IMFRegList | Specifies the ordered list of attributes whose values are registered with the IMF notification module. The attribute values can be overridden at the resource level. |

**Table 4-2** Optional attributes for configuring the Docker Daemon agent *(continued)*

| Optional attribute | Description |
|---|---|
| EnvFile | The complete file path name required to set the environment before executing the Docker daemon commands. The shell environments supported are ksh, sh, and csh.<br>Type and dimension: string-scalar<br>Default: No default value<br>Example: `/test/bin/envvars` |
| User | The operating system username under which the agent executes programs to manage the Docker daemon. By default, the Docker daemon is started as a user root.<br>Type and dimension: string-scalar<br>Default: root<br>Example: root |
| DaemonStopCommand | The command and arguments that stop the Docker daemon. Ensure that the Docker daemon stops outside the VCS control using the specified command.<br>Type and dimension: string-scalar<br>Default: `/usr/sbin/service docker stop`<br>Example 1: `/usr/sbin/service docker stop`<br>Example 2: `/usr/bin/systemctl stop docker`<br>Example 3: `/usr/sbin/service podman stop`<br>Example 4: `/usr/bin/systemctl stop podman` |

## About the keys of the IMF attribute

The IMF type-level attribute uses the following keys:

**Table 4-3** IMF attribute keys

| Key | Description |
|---|---|
| Mode | Define this attribute to enable or disable intelligent resource monitoring. Valid values are as follows:<br>■ 0: Does not perform intelligent resource monitoring<br>■ 2: Performs intelligent resource monitoring for online resources and performs poll-based monitoring for offline resources<br>**Note:** The agent for Docker supports intelligent resource monitoring for online resources only. Hence, Mode should be set to either 0 or 2.<br>Default: 2 |
| MonitorFreq | The frequency at which the agent invokes the monitor agent function. The value of this key is an integer.<br>You can set this key to a non-zero value for cases where the agent requires to perform both poll-based and intelligent resource monitoring.<br>If the value is 0, the agent does not perform poll-based process check monitoring.<br>After the resource registers with the AMF kernel driver, the agent calls the monitor agent function as follows:<br>■ After every (MonitorFreq x MonitorInterval) number of seconds for online resources<br>■ After every (MonitorFreq x OfflineMonitorInterval) number of seconds for offline resources<br>Default: 5 |
| RegisterRetryLimit | If you enable intelligent resource monitoring, the agent invokes the imf_register agent function to register the resource with the AMF kernel driver.<br>The value of the RegisterRetryLimit key determines the number of times the agent must retry registration for a resource. If the agent cannot register the resource within the limit that is specified, then intelligent monitoring is disabled until the resource state changes or the value of the Mode key changes.<br>Default: 3 |

# Enabling the Docker Daemon agent to support IMF

By default, the Docker Daemon agent is enabled to support IMF. If the IMF support for Docker Daemon agent is disabled by a user, you can enable the IMF support by setting the IMF Mode attribute to 2.

## Disabling intelligent resource monitoring

**To disable intelligent resource monitoring for Docker Daemon**

1. Make the VCS configuration writable.

   ```
   # haconf -makerw
   ```

2. To disable intelligent resource monitoring for all the resources of a certain type, run the following command:

   ```
   # hatype -modify DockerDaemon IMF -update Mode 0
   ```

3. To disable intelligent resource monitoring for a specific resource, run the following command:

   ```
   # hares -override resource_name IMF
   ```

   ```
   # hares -modify resource_name IMF -update Mode 0
   ```

4. Save the VCS configuration.

   ```
   # haconf -dump -makero
   ```

# Docker Container agent attributes

**Table 4-4** Required attributes for configuring the Docker Container agent

| Required attributes | Description |
|---|---|
| ContainerName | The unique name of the Docker container.<br>To get a container name, run the `docker ps -a` command.<br>Type and dimension: string-scalar<br>Default: No default value<br>Example 1: evil_ptolemy<br>Example 2: mysql_container_1 |

**Table 4-4** Required attributes for configuring the Docker Container agent *(continued)*

| Required attributes | Description |
|---|---|
| ImageName | The Docker image name. This attribute is required if the Docker container is to be created during an online operation.<br>ImageID can also be specified as part of the ImageName attribute.<br>Type and dimension: string-scalar<br>Default: No default value<br>Example 1: Ubuntu<br>Example 2: 8dffsd933n3<br>Example 3: `Docker.io/Ubuntu:latest` |

**Table 4-5** Optional attributes for configuring the Docker Container agent

| Optional attribute | Description |
|---|---|
| ResLogLevel | The logging detail performed by the agent for the resource. Valid values are:<br>■ ERROR: Only logs error messages.<br>■ WARN: Logs above plus warning messages.<br>■ INFO: Logs above plus informational messages.<br>■ TRACE: Logs above plus trace messages. TRACE is very verbose and should only be used during initial configuration or for troubleshooting and diagnostic operations.<br>Type and dimension: string-scalar<br>Default: INFO<br>Example: TRACE<br>**Note:** The use of the ResLogLevel attribute is deprecated from VCS version 6.2 onwards. You must use the LogDbg attribute instead of the ResLogLevel attribute to enable debug logs for the ACCLib-based agents, when the ACCLib version is 6.2.0.0 or later. The agent captures the first failure data of the unexpected events and automatically logs debug messages in their respective agent log files. |

**Table 4-5** Optional attributes for configuring the Docker Container agent *(continued)*

| Optional attribute | Description |
|---|---|
| LogDbg | For ACCLib-based agents, you must use the LogDbg resource type attribute to enable the debug logs when the ACCLib version is 6.2.0.0 or later and the VCS version is 6.2 or later.<br>Set the LogDbg attribute to DBG_5 to enable debug logs for the ACCLib-based agent. By default, setting the LogDbg attribute to DBG_5 enables debug logs for all Docker resources in the cluster. If debug logs must be enabled for a specific Docker resource, override the LogDbg attribute.<br>For more information on how to enable debug logs, See "To enable debug logs for all resources of type Docker" on page 46.<br>Type and dimension: keylist<br>Default: No default value<br>For more information on how to use the LogDbg attribute, refer to the *Cluster Server Administrator's Guide*. |
| ContainerInitCommand | The command executed inside the container. Specify full path for the command and specify any arguments to be passed to the `container init` command. Value of this attribute is used, if the container is created during an online operation.<br>Type and dimension: string-scalar<br>Default: No default value<br>Example 1: `/shared_storage/my_application.pl`<br>Example 2: `/abc/xyz arg1 arg2` |
| DockerBinaryPath | The full path to the Docker binary.<br>Type and dimension: string-scalar<br>Default: `/usr/bin/docker`<br>Example 1: `/path/to/docker`<br>Example 2: `/usr/bin/podman` |

**Table 4-5** Optional attributes for configuring the Docker Container agent *(continued)*

| Optional attribute | Description |
|---|---|
| DockerRunOptions | The options passed during the `docker run` command. The `-d` option can be used to run the container in the detached mode, which is the default value of the DockerRunOptions attribute. Agent passes the `--name=$ContainerName` option, if the container is to be created during an online operation of the agent. Additional options can be passed as part of this attribute.<br>Type and dimension: integer-scalar<br>Default: `-d`<br>Example: `-d -it -v /my_shared_storage/C1:/C1` |
| EnvFile | The complete file path name to source in order to set the environment before executing the `docker` command.<br>The Docker client will honor the DOCKER_HOST environment variable to set the `-H` flag for the client.<br>You can specify the following line in the environment file:<br>`export DOCKER_HOST="tcp://0.0.0.0:2375"`<br>Type and dimension: string-scalar<br>Default: No default value<br>Example: `/test/bin/envvars` |
| IMF | Determines if the agent must perform the intelligent resource monitoring. You can also override the value of this attribute at the resource level. |
| IMFRegList | Specifies the ordered list of attributes whose values are registered with the IMF notification module. The attribute values can be overridden at the resource level. |

**Table 4-5** Optional attributes for configuring the Docker Container agent *(continued)*

| Optional attribute | Description |
|---|---|
| MonitorProgram | The absolute path name of an external, user-supplied monitor executable.<br><br>If specified, the monitor entry point will execute this file to perform an additional server state check. There are no restrictions for what actions the external monitor program performs to determine the state of a Docker Container instance server. The only constraint is that the external monitor program must return one of the following integer values:<br><br>■ 0: server is online<br>■ 110: server is online<br>■ 100: server is offline<br>■ 1: server is offline<br>■ Any other value: state is unknown.<br><br>Veritas recommends storing the external monitor utility on the shared disk directory to ensure the file is always available on the online system.<br><br>Type and dimension: string-scalar<br><br>Default: No default value<br><br>Example 1: `$ServerRoot/bin/myMonitor.pl`<br><br>Example 2: `$ServerRoot/bin/myMonitor.sh arg1 arg2` |
| User | The account name under which the agent executes programs to manage the Docker container. If unspecified, the Docker daemon is started as user root.<br><br>Type and dimension: string-scalar<br><br>Default: root<br><br>Example: root |

## About the keys of the IMF attribute

The IMF type-level attribute uses the following keys:

**Table 4-6** IMF attribute keys

| Key | Description |
|---|---|
| Mode | Define this attribute to enable or disable intelligent resource monitoring. Valid values are as follows:<br>■ 0: Does not perform intelligent resource monitoring<br>■ 2: Performs intelligent resource monitoring for online resources and performs poll-based monitoring for offline resources<br>**Note:** The agent for Docker supports intelligent resource monitoring for online resources only. Hence, Mode should be set to either 0 or 2.<br>Default: 2 |
| MonitorFreq | This key value specifies the frequency at which the agent invokes the monitor agent function. The value of this key is an integer.<br>You can set this key to a non-zero value for cases where the agent requires to perform both poll-based and intelligent resource monitoring.<br>If the value is 0, the agent does not perform poll-based process check monitoring.<br>After the resource registers with the AMF kernel driver, the agent calls the monitor agent function as follows:<br>■ After every (MonitorFreq x MonitorInterval) number of seconds for online resources<br>■ After every (MonitorFreq x OfflineMonitorInterval) number of seconds for offline resources<br>Default: 5 |
| RegisterRetryLimit | If you enable intelligent resource monitoring, the agent invokes the imf_register agent function to register the resource with the AMF kernel driver.<br>The value of the RegisterRetryLimit key determines the number of times the agent must retry registration for a resource. If the agent cannot register the resource within the limit that is specified, then intelligent monitoring is disabled until the resource state changes or the value of the Mode key changes.<br>Default: 3 |

# Enabling the Docker Container agent to support IMF

By default, the Docker Container agent is enabled to support IMF. If the IMF support for Docker Container agent is disabled by a user, you can enable the IMF support by setting the IMF Mode attribute to 2.

## Disabling intelligent resource monitoring

**To disable intelligent resource monitoring for Docker Container**

1. Make the VCS configuration writable.

   ```
   # haconf -makerw
   ```

2. To disable intelligent resource monitoring for all the resources of a certain type, run the following command:

   ```
   # hatype -modify DockerContainer IMF -update Mode 0
   ```

3. To disable intelligent resource monitoring for a specific resource, run the following command:

   ```
   # hares -override resource_name IMF
   ```

   ```
   # hares -modify resource_name IMF -update Mode 0
   ```

4. Save the VCS configuration.

   ```
   # haconf -dump -makero
   ```

# Limitations of VCS Docker agents

The following are the limitations of the Docker agents.

- The container rename operation may result in an unexpected behavior when the container is configured under the VCS control.
- The `-d` option is required in the detached mode.
  The DockerRunOptions attribute takes `-d` as its default value. This option is required because it allows to run a container in the detached mode.

Chapter 5

# Configuring service groups for Docker

This chapter includes the following topics:



## About configuring service groups for Docker

Configuring the Docker service group involves creating the Docker service group, its resources, and defining attribute values for the configured resources. You must have administrator privileges to create and configure a service group.

## Before configuring the service groups for Docker

Before you configure the Docker service group, you must:

- Verify that Cluster Server is installed and configured on all nodes in the cluster where you will configure the service group.
  For more information about installing and configuring Cluster Server, refer to the Cluster Server installation and configuration guides.
- Verify that the Cluster Server agents for Docker is installed on all nodes in the cluster.
  See “Installing the agents ” on page 21.

- Verify that the type definition for the Cluster Server agents for Docker is imported into the VCS engine.
  See "Importing the agent types files in a VCS environment" on page 24.

It is recommended to download the images, else the `docker run` command pulls images from the registry, which is time consuming. The container start operation may fail. You can set the value of the OnlineTimeOut attribute appropriately to avoid failures.

# Configuring service groups for Docker Daemon

While setting up a cluster, you must always ensure that the cluster has some spare capacity to handle the Docker failover scenarios. For example, in case of any container failure, the cluster must be able to run another container instance in conjunction with other running applications.

The cluster should be able to provide application failover by encapsulating the resources required for an application into a service group. A service group is a virtualized application that can switch between the cluster nodes. It contains a set of dependent resources, such as disk groups, disk volumes, file systems, IP addresses, NIC cards, and dependent application processes. It also includes logic about the dependencies between the application components.

These service groups should thus be configured such that the cluster can start, stop, monitor, and switch the service groups between the nodes, depending upon the server faults or resource faults. An administrator should also be proactively able to move a service group between cluster nodes to perform preventative maintenance or apply patches.

**Perform the following steps to add a service group for Docker Daemon**

1. Create a service group for Docker Daemon.

   ```
   # hagrp -add <daemon_servicegroup_name>
   ```

   For more details on creating a service group refer to, *Cluster Server Administrator's Guide*.

2. Modify the SystemList attribute for the group, to add systems.

   For example,

   ```
   # hagrp -modify <daemon_servicegroup_name> SystemList systemA 0
   systemB 1
   ```

3. Create resource for Docker Daemon in the service group.

   For example,

   ```
   # hares -add <resource_name> DockerDaemon
   <daemon_servicegroup_name>
   ```

   For more details on creating and modifying resource attributes for DiskGroup, Volume, and Mount refer to, *Cluster Server Bundled Agents Reference Guide*.

   Based on the Docker Daemon instance in your cluster, modify the resource attributes.

# Configuring service groups for Docker Container

**Perform the following steps to add a service group for Docker Container**

1. Create a service group for Docker Container.

   ```
   # hagrp -add <container_servicegroup_name>
   ```

   For more details on creating a service group refer to, *Cluster Server Administrator's Guide*.

2. Modify the SystemList attribute for the group, to add systems.

   For example,

   ```
   # hagrp -modify <container_servicegroup_name> SystemList systemA
   0 systemB 1
   ```

3 Create resources for the linked Docker Containers in the service group.

For example,

```
# hares -add <docker_container1> DockerContainer
<container_servicegroup_name>
```

```
# hares -add <docker_container2> DockerContainer
<container_servicegroup_name>
```

For more details on creating and modifying resource attributes for NIC, IP, DiskGroup, Volume, and Mount refer to, *Cluster Server Bundled Agents Reference Guide*.

Based on the Docker Container instance in your cluster, modify the resource attributes.

Perform steps 4 through 6 to mount data volume inside the container.

4 Add the file system to respective agent service group using the Mount, DiskGroup, and Volume resources.

Create Mount, DiskGroup, and Volume resources.

For example,

```
# hares -DCM652-CS_mnt Mount <container_servicegroup_name>
```

```
# hares -add DCM652-CS_dg DiskGroup <container_servicegroup_name>
```

Based on the Docker Container instance in your cluster, modify the resource attributes of the Mount, DiskGroup, and Volume resources.

5 Create links between the Mount, DiskGroup, and Volume resources.

For example,

```
# hares -link DCM652-CS_mnt DCM652-CS_vol
```

```
# hares -link -DCM652-CS_vol DCM652-CS_dg
```

6 Verify the final resource dependencies for <container_servicegroup_name> server group.

For example,

```
# hares -dep
```

```
Group                          Parent          Child
<container_servicegroup_name>  DCM652-CS_cs    DCM652-CS _mnt
<container_servicegroup_name>  DCM652-CS_mnt   DCM652-CS_vol
<container_servicegroup_name>  DCM652-CS_vol   DCM652-CS_dg
```

7 Create dependency between the Docker Daemon service group and the Docker Container service group.

```
# hagrp -link <daemon_servicegroup_name>
<container_servicegroup_name> -type online local hard
```

8 Create dependency between the Docker Container service group and the storage service group.

```
# hagrp -link <container_servicegroup_name>
<storage_servicegroup_name> -type online local firm
```

Chapter 6

# Troubleshooting the agents for Docker

This chapter includes the following topics:



## Preliminary troubleshooting checks

If you face problems with the Cluster Server agents for Docker, perform the following checks before further investigation:

- Use the correct software and operating system versions.
  Ensure that no issues arise due to incorrect software and operating system versions. For information on the software versions that the agents for Docker supports, see the Veritas Services and Operations Readiness Tools (SORT) site: http://sort.veritas.com/agents.
- Meet prerequisites.
  Before installing the agents for Docker, ensure that all the prerequisites are met. For example, you must install the ACC library on VCS before installing the agent for Docker.
- Configure the Docker resources correctly.

Before using the Docker resources, ensure that you configure the resources properly. For a list of attributes used to configure all Docker resources, refer to the agent attributes.

# Starting the Docker Daemon instance outside a cluster

If you face problems while working with a resource, you must disable the resource within the cluster framework. A disabled resource is not under the control of the cluster framework, and so you can test the Docker Daemon instance independent of the cluster framework. Refer to the cluster documentation for information about disabling a resource.

You can then restart the Docker Daemon instance outside the cluster framework.

---

**Note:** While restarting the Docker Daemon instance outside the cluster framework, use the same parameters as that configured for the VCS resource.

---

**To restart the Docker Daemon instance outside the cluster framework**

1. Log in as a superuser.
2. Ensure that the Docker Daemon is up and running. Refer to the Docker documentation for more information.
3. Run the following command to restart the Docker Daemon instance.

   ```
   service docker restart
   ```

4. Ensure that the Docker Daemon instance is running successfully by running the grep command for Docker Daemon.

   For example,

   ```
   $ ps -ef | grep docker
   ```

# Starting the Docker Container instance outside a cluster

If you face problems while working with a resource, you must disable the resource within the cluster framework. A disabled resource is not under the control of the cluster framework, and so you can test the Docker Container instance independent of the cluster framework.

You can then restart the Docker Container instance outside the cluster framework.

**Note:** While restarting the Docker Container instance outside the cluster framework, use the same parameters as that configured for the VCS resource.

**To restart the Docker Container instance outside the cluster framework**

1. Log in as a superuser.
2. Ensure that the Docker Container is up and running. Refer to the Docker documentation for more information.
3. Run the following command to restart the Docker Container instance.

   ```
   docker restart <container_name>
   ```

4. Ensure that the Docker Container instance is running successfully by running the ps command for Docker Container.

   For example,

   ```
   $ docker ps
   ```

# Reviewing log files

If you face problems while using Docker or the agent for Docker, use the log files described in this section to investigate the problems.

- Cluster log files
  The engine log file is located at `/var/VRTSvcs/log/engine_A.log`. For a long running cluster, the log files are rotated as `engine_B.log`, `engine_C.log`, and so on. The most recent engine logs are present in the `engine_A.log` file.
- Docker agent log files
  The Docker Daemon agent log file is located at `/var/VRTSvcs/log/DockerDaemon_A.log` and the Docker Container agent log file is located at `/var/VRTSvcs/log/DockerContainer_A.log`.

## Using trace level logging

The ResLogLevel attribute controls the level of logging that is written in a cluster log file for each Docker resource. You can set this attribute to TRACE, which enables very detailed and verbose logging.

If you set the ResLogLevel attribute to TRACE, a very high volume of messages are produced. Veritas recommends that you localize the ResLogLevel attribute for a particular resource.

---

**Warning:** You may consider temporarily increasing the timeout values for Docker for debugging purposes. After the debugging process is complete, you can revert back to the original timeout values.

---

The LogDbg attribute should be used to enable the debug logs for the ACCLib-based agents when the ACCLIB version is 6.2.0.0 or later and the VCS version is 6.2 or later.

**To localize ResLogLevel attribute for a resource**

1. Identify the resource for which you want to enable detailed logging.
2. Localize the ResLogLevel attribute for the identified resource:

   ```
   # hares -local Resource_Name ResLogLevel
   ```

3. Set the ResLogLevel attribute to TRACE for the identified resource:

   ```
   # hares -modify Resource_Name ResLogLevel TRACE -sys SysA
   ```

4. Note the time before you begin to operate the identified resource.
5. Test the identified resource. The function reproduces the problem that you are attempting to diagnose.
6. Note the time when the problem is reproduced.
7. Set the ResLogLevel attribute back to INFO for the identified resource:

   ```
   # hares -modify Resource_Name ResLogLevel INFO -sys SysA
   ```

8. Save the configuration changes.

   ```
   # haconf -dump
   ```

9. Review the contents of the log file. Use the time noted in Step 4 and Step 6 to diagnose the problem.

   You can also contact Veritas support for more help.

**To enable debug logs for all resources of type Docker**

- Enable the debug log.

  For Docker Daemon, run the following command:

  ```
  # hatype -modify DockerDaemon LogDbg DBG_5
  ```

  For Docker Container, run the following command:

  ```
  # hatype -modify DockerContainer LogDbg DBG_5
  ```

**To override the LogDbg attribute at resource level**

- Override the LogDbg attribute at the resource level and enable the debug logs for the specific resource.

  ```
  # hares -override hh LogDbg
  # hares -modify hh LogDbg DBG_5
  ```

# Troubleshooting the agent

If you face problems with the Docker agent, consider the following:

- Container creation might fail on an SE Linux-enabled host, if you bind mount vxfs volume inside the container.
  The container creation fails when SE Linux is enabled in the enforcing mode and you try to bind mount vxfs volume inside the container.
  Workaround: Run the following command:

  ```
  setenforce 0
  ```

  If you now try to bind mount vxfs volume inside the container, the container creation succeeds.

Appendix A

# Sample Configurations

This appendix includes the following topics:



## About sample configurations for the agents for Docker

The sample configuration graphically depicts the resource types, resources, and resource dependencies within the service group. Review these dependencies carefully before configuring the agents for Docker. For more information about these resource types, refer to the *Cluster Server Bundled Agents Reference Guide*.

## Sample agent type definition for Docker Daemon

After importing the agent type file into the cluster, if you save the configuration on your system disk using the `haconf -dump` command, you can find the `DockerTypes.cf` file in the `/etc/VRTSvcs/conf/config` cluster configuration directory.

An excerpt from this file is as follows:

```
type DockerDaemon (
 static int IMF{} = { Mode=2, MonitorFreq=5, RegisterRetryLimit=3 }
 static str IMFRegList[] = { DaemonProcPattern, User }
```

```
 static str AgentDirectory = "/opt/VRTSagents/ha/bin/DockerDaemon"
 static str AgentFile = "/opt/VRTSvcs/bin/Script60Agent"
 static keylist LogDbg = { DBG_1, DBG_2, DBG_3, DBG_4, DBG_5, DBG_6 }
 static int RestartLimit = 1
 static str ArgList[] = { ResLogLevel, State, IState, User, EnvFile,
                          DaemonStartCommand, DaemonStopCommand,
                          DaemonProcPattern, MonitorProgram }
 static boolean AEPTimeout = 1
 str DaemonStartCommand = "/usr/sbin/service docker start"
 str DaemonStopCommand = "/usr/sbin/service docker stop"
 str DaemonProcPattern = "\\bdocker .*-d"
 str EnvFile
 str MonitorProgram
 str ResLogLevel = INFO
 str User = root
)
```

# Sample agent type definition for Docker Container

After importing the agent type file into the cluster, if you save the configuration on your system disk using the `haconf -dump` command, you can find the `DockerTypes.cf` file in the `/etc/VRTSvcs/conf/config` cluster configuration directory.

An excerpt from this file is as follows:

```
type DockerContainer (
 static int IMF{} = { Mode=2, MonitorFreq=5, RegisterRetryLimit=3 }
 static str IMFRegList[] = { ContainerName, User }
 static boolean IntentionalOffline = 0
 static str AgentDirectory = "/opt/VRTSagents/ha/bin/DockerContainer"
 static str AgentFile = "/opt/VRTSvcs/bin/Script60Agent"
 static int LevelTwoMonitorFreq = 1
 static keylist LogDbg = { DBG_5 }
 static str ArgList[] = { ResLogLevel, State, IState, User,
                          EnvFile, ContainerName, ContainerInitCommand,
                          DockerBinaryPath, DockerRunOptions,
                          ImageName, MonitorProgram }
 static boolean AEPTimeout = 1
 str ResLogLevel = INFO
 str User = root
 str EnvFile
```

```
 str ContainerName
 str ContainerInitCommand
 str DockerBinaryPath = "/usr/bin/docker"
 str DockerRunOptions = "-d"
 str ImageName
 str MonitorProgram
)
```

# Sample Docker resource configuration

This section contains sample configurations for Docker Daemon and Docker Container.

## Sample configuration for Docker

A sample excerpt from the `main.cf` file for the Docker resource configuration is shown below:

```
include "types.cf"
include "CFSTypes.cf"
include "CVMTypes.cf"
include "DockerTypes.cf"

cluster vcs_docker (
 UserNames = { a = dqqK }
 Administrators = { a }
 )

system SystemA (
 )

system SystemB (
 )

group CVM_Mount (
 SystemList = { SystemA = 0, SystemB = 1 }
 AutoFailOver = 0
 Parallel = 1
 AutoStartList = { SystemA, SystemB }
 )

 CFSMount cvm_mount (
  MountPoint = "/cfs_mount"
```

```
 BlockDevice = "/dev/vx/dsk/docker_dg/docker_volume"
 )

CVMVolDg cvmdg (
 CVMDiskGroup = docker_dg
 CVMVolume = { docker_volume }
 CVMActivation = sw
 )

requires group cvm online local firm
cvm_mount requires cvmdg


// resource dependency tree
//
// group CVM_Mount
// {
// CFSMount cvm_mount
//     {
//     CVMVolDg cvmdg
//     }
// }

group applicationContainer (
 SystemList = { SystemA = 0, SystemB = 1 }
 )

 DockerContainer app_res (
  Critical = 0
  ResLogLevel = TRACE
  ContainerName = app_container
  ContainerInitCommand = "/dv_c1/myapp.pl"
  DockerRunOptions = "-d -v /cfs_mount/dv_c1/:/dv_c1 "
  ImageName = ubuntu
  )

 requires group docker_daemon_sg online local firm
 requires group CVM_Mount online local firm

 // resource dependency tree
 //
 // group applicationContainer
 // {
```

```
 // DockerContainer app_res
 // }

group cvm (
 SystemList = { SystemA = 0, SystemB = 1 }
 AutoFailOver = 0
 Parallel = 1
 AutoStartList = { SystemA, SystemB }
 )

 CFSfsckd vxfsckd (
  )

 CVMCluster cvm_clus (
  CVMClustName = vcs_docker
  CVMNodeId = { SystemA = 0, SystemB = 1 }
  CVMTransport = gab
  CVMTimeout = 200
  )

 CVMVxconfigd cvm_vxconfigd (
  Critical = 0
  CVMVxconfigdArgs = { syslog }
  )

 ProcessOnOnly vxattachd (
  Critical = 0
  PathName = "/bin/sh"
  Arguments = "- /usr/lib/vxvm/bin/vxattachd root"
  RestartLimit = 3
  )

 cvm_clus requires cvm_vxconfigd
 vxfsckd requires cvm_clus


 // resource dependency tree
 //
 // group cvm
 // {
 // ProcessOnOnly vxattachd
 // CFSfsckd vxfsckd
 //     {
```

```
 //     CVMCluster cvm_clus
 //          {
 //          CVMVxconfigd cvm_vxconfigd
 //          }
 //     }
 // }

group databaseContainer (
 SystemList = { SystemA = 0, SystemB = 1 }
 )

 DockerContainer db_res (
  Critical = 0
  ResLogLevel = TRACE
  ContainerName = database_container
  DockerRunOptions = "-d -it "
  ImageName = ubuntu
  )

 requires group docker_daemon_sg online local firm

 // resource dependency tree
 //
 // group databaseContainer
 // {
 // DockerContainer db_res
 // }

group docker_daemon_sg (
 SystemList = { SystemA = 0, SystemB = 1 }
 Parallel = 1
 )

 DockerDaemon daemon_res (
  RestartLimit = 1
  )

 // resource dependency tree
 //
 // group docker_daemon_sg
 // {
 // DockerDaemon daemon_res
 // }
```

```
group webServerContainer (
 SystemList = { SystemA = 0, SystemB = 1 }
 )

 DockerContainer web_res (
  Critical = 0
  ResLogLevel = TRACE
  ContainerName = web_container
  DockerRunOptions = "-d -it "
  ImageName = ubuntu
  )

 requires group docker_daemon_sg online local firm

 // resource dependency tree
 //
 // group webServerContainer
 // {
 // DockerContainer web_res
 // }
```

## Sample configuration for Podman

A sample excerpt from the `main.cf` file for the Podman resource configuration is shown below:

```
group Podman_daemon (
        SystemList = { inaqalnx070 = 0, inaqalnx073 = 1 }
        Parallel = 1
        )
        DockerDaemon podman_daemon_res (
                DaemonStartCommand = "/usr/bin/systemctl start
                podman.service"
                DaemonStopCommand = "/usr/bin/systemctl stop
                podman.service"
                DaemonProcPattern = "/usr/bin/podman --log-level=info
                system service"
                )
group Container_sg (
        SystemList = { inaqalnx070 = 0, inaqalnx073 = 1 }
        )
        DockerContainer ubuntu_res (
```

```
ContainerName = my_ubuntu
DockerBinaryPath = "/usr/bin/podman"
DockerRunOptions = "-d -it"
ImageName = "docker.io/library/ubuntu"
)
```

# Sample service group dependency

This section depicts various ways to configure the Docker resources.

Figure A-1 depicts the Docker resource configuration when the container is created with a data volume, which is mounted on the CFS/CVM storage.

**Figure A-1** Container dependency on CFS/CVM

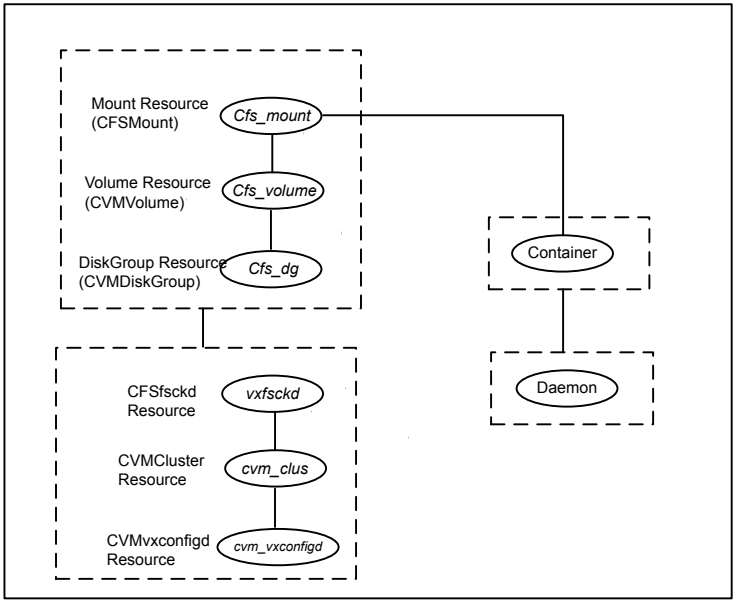
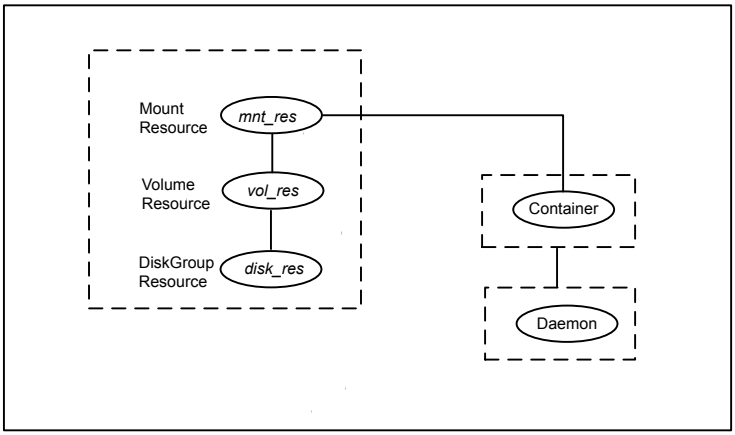



Figure A-2 depicts the Docker resource configuration when the container is created with a data volume, which is mounted on the VXVM storage.

**Figure A-2** Container dependency on VXVM



# Index

## E



## L



## R



## S



## T



## U



## V