zenoss
Own IT.

Control Center Installation Guide for High-Availability Deployments

Release 1.5.1

Zenoss, Inc.

www.zenoss.com

Control Center Installation Guide for High-Availability Deployments

Part Number: 1922.18.158.34

Zenoss, Inc.
11305 Four Points Drive
Bldg 1 - Suite 300
Austin, Texas 78726

Contents

About this guide

Control Center Installation Guide for High-Availability Deployments provides detailed procedures for installing and configuring a Control Center in a high-availability deployment.

Supported clients and browsers

The following table identifies the supported combinations of client operating systems and web browsers.

Client OS	Tested browsers
Windows 7, 10	Internet Explorer 11[*]
	Firefox 56 and later
	Chrome 61 and later
macOS 10.12.3, 10.13	Firefox 56 and later
	Chrome 61 and later
Ubuntu 14.04 LTS	Firefox 56 and later
	Chrome 61 and later

Related publications

Title	Description
Control Center Release Notes	Describes known issues, fixed issues, and late-breaking information not included in other publications.
Control Center Installation Guide	Provides detailed procedures for installing and configuring Control Center.
Control Center Installation Guide for High-Availability Deployments	Provides detailed procedures for installing and configuring Control Center in a high-availability deployment.
Control Center Reference Guide	Provides information and procedures for managing Control Center. This information is also available as online help in the Control Center browser interface.
Control Center Upgrade Guide	Provides detailed procedures for updating a Control Center deployment to the latest release.
Control Center Upgrade Guide for High-Availability Deployments	Provides detailed procedures for updating a high-availability deployment of Control Center to the latest release.

Documentation feedback

To provide feedback about this document, or to report an error or omission, please send an email to `docs@controlcenter.io`. In the email, please include the document title (*Control Center Installation Guide for High-Availability Deployments*) and part number (1922.18.158.34) and as much information as possible about the context of your feedback.

[*] Enterprise mode only; compatibility mode is not tested.

Change history

The following list associates document part numbers and the important changes to this guide since the previous release. Some of the changes involve features or content, but others do not. For information about new or changed features, refer to the *Control Center Release Notes*.

1922.18.158.34 (1.5.1)

Add a section about setting the connection timeout on resource pools to the end of the delegate configuration chapter.

1922.17.349 (1.5.0)

Specify Docker version in `yum install` command.

Reinstate steps to disable firewall.

1922.17.331 (1.5.0)

Replace Leapfile.net with delivery.zenoss.com.

Add steps for importing the Zenoss GPG key.

1922.17.311 (1.5.0)

Add RHEL/CentOS 7.4; remove RHEL/CentOS 7.1.

Add the specific port numbers required for DRBD to the Planning chapter.

Add a step for configuring SELinux policies to the DRBD chapter.

Remove a duplicate NTP configuration step from the delegate installation procedure.

Remove the configuration of *SERVICED_OUTBOUND_IP* from the delegate host configuration chapter.

1322.17.268

Update release number (1.4.1).

1322.17.242

Add a new storage requirement, for audit logs. For more information, see *Master host storage requirements* on page 10.

Change the `on` stanza in the DRBD configuration file (`/etc/drbd.d/global_common.conf`) to use hostnames instead of IP addresses.

Remove the `-y` parameter from all `yum` command invocations.

1322.17.206

Update the *SERVICED_DOCKER_REGISTRY* configuration steps to ensure the correct value is set.

1322.17.171

Update release number (1.3.3)

Replace Docker 1.12.1 with Docker CE 17.03.1.

Remove step for disabling SELinux.

1322.17.122

Update release number (1.3.2)

Move the step for loading Docker images to the next procedure.

1322.17.100

Initial release (1.3.1).

Planning a high-availability deployment 1

This chapter provides information about planning a high-availability deployment of Control Center, and about preparing to create a high-availability deployment. Before proceeding, please read the *Resource Manager Planning Guide*.

For optimal results, review the contents of this guide thoroughly before performing an installation.

Note The procedures in this guide describe how to configure a deployment that does not have internet access. You may create a deployment that does have internet access; the procedures assume that the deployment does not have access.

About high-availability deployments

Control Center can be deployed in a high-availability cluster to minimize downtime caused by the failure of hardware components or operating system services in a Control Center master host.

A high-availability cluster can be configured in a variety of ways, including the following:

- Active-Active, non geo-diverse
- Active-Passive, geo-diverse
- Active-Passive, non geo-diverse (works with Control Center)

The Active-Passive configuration that works with Control Center:

- uses Pacemaker, Corosync, and Distributed Replicated Block Device (DRBD) to manage the cluster
- includes two or more identical master hosts in the cluster—one primary node and one or more secondary nodes, which take over as the Control Center master host if the primary node becomes unavailable
- requires a minimum of two master hosts and two delegate hosts
- provides no protection against a facility catastrophe (the cluster nodes are located in the same facility)

Note Zenoss has tested DRBD 8.4, not DRBD 9.0. The procedures in this guide install the latest version of release 8.4.

High-availability cluster configuration options

The recommended deployment architecture requires two identical dual-NIC machines for optimal disk synchronization and network responsiveness. However, you can deploy a high-availability cluster with single-NIC servers if two identical dual-NIC machines are not available.

Requirements for two identical dual-NIC servers

- Master hosts: In a separate resource pool, you need two identical hosts in the role of Control Center master host; one host serves as the primary node and the other as the secondary node.
- Provide two NICs on the cluster primary node and two on the secondary node. On each node, dedicate one NIC to

  - network traffic that is required for disk synchronization via DRBD.
  - Control Center and application traffic.

  Route traffic for each NIC through separate subnets.
- Delegate hosts: You need *N*+1 identical hosts to serve as delegate hosts, where *N* is the number of hosts needed to satisfy the performance and scalability requirements of the tenant. They do not need to be members of the cluster because, if a host becomes unavailable, Control Center restarts their services on other hosts.
- Deploy application services on dedicated Control Center delegate hosts outside the cluster.

Requirements for two single-NIC servers

- Master host: Configure two hosts for the Control Center master host in an active/passive cluster.

  - Use the two hosts only to run Control Center. Do not use them to run application services.
  - For primary and secondary nodes that contain only one network-interface card (NIC), the network they use to communicate must support multicast.
- Delegate hosts: You need *N*+1 identical hosts to serve as delegate hosts, where *N* is the number of hosts needed to satisfy the performance and scalability requirements of the pool. (Only the Control Center master hosts must be configured in an active/passive cluster.)

Fencing recommendations

Fencing is an automated means of isolating a node that appears to be malfunctioning, used to protect the integrity of the DRBD volumes. In a *test* deployment of a Control Center high-availability cluster, fencing is not necessary. However, on *production* clusters, fencing is a critical consideration.

Work with your IT department to implement the best fencing solution for your infrastructure. Employ a technique that ensures that a failed node in the cluster is completely stopped to avoid application conflicts or conflicts with the cluster management software.

When fencing is employed in the high-availability cluster, use two NICs per node.

Before you configure and enable fencing in your production environment:

- Ensure that all components are deployed.
- Verify operation of the application that Control Center is managing.
- In a controlled scenario, confirm basic cluster failover.

  If a fencing method is not defined, when the cluster attempts to fail over to the backup node, the following error results:

  ```
  no method defined
  ```

Place the fencing device on the public network. (Passing heartbeat communication through a private network interface is not recommended. Doing so requires a complex fencing system that is prone to issues. For more information, see Quorum Disk documentation on the Red Hat website.)

Using a public network interface enables a healthy node to fence the unhealthy node, and prevents the unhealthy node from fencing the healthy node. If heartbeat communications pass through the public network and the link for a node goes down, the node with the down public network link cannot communicate with the fencing device.

Master host storage requirements

The following table identifies the minimum local storage requirements of Control Center master host nodes in high-availability deployments.

Purpose		Minimum size	Description
1	Root (`/`)	2-10GB (required)	Local, high-performance storage (XFS)
2	Swap	12-16GB (required)	Local, high-performance storage
3	Docker temporary	10GB (required)	Local, high-performance storage (XFS)
4	Control Center audit logging	10GB (configurable)	Remote, XFS-compatible storage
5	Docker data	50GB (required)	Local, high-performance storage
6	Control Center internal services data	50GB (required)	Local, high-performance storage (XFS)
7	Application data	200GB (suggested)	Local, high-performance storage
8	Application metadata	1GB (required)	Local, high-performance storage (XFS)
9	Application data backups	150GB (suggested)	Remote, XFS-compatible storage.

Areas 1-3 can be combined in a single filesystem when the operating system is installed. Areas 5-8 must be separate real or virtual devices. This guide includes procedures for preparing areas 5-8.

The suggested minimum sizes for application data (7) and application data backups (9) should be replaced with sizes that meet your application requirements. To calculate the appropriate sizes for these storage areas, use the following guidelines:

- Application data storage includes space for both data and snapshots. The default base size for data is 100GB, and the recommended space for snapshots is 100% of the base size. Adding the two yields the suggested minimum size of 200GB.
- For application data backups, the recommended space is 150% of the base size for data.

The suggested minimum size for application data backups is 150GB. For improved reliability and performance, Zenoss strongly recommends using shared remote storage or network-attached storage for application data backups. This guide does not include instructions for mounting remote storage, but does include a step for creating mount points.

Master host resource requirements

The default recommendation for multi-host deployments is to use a master host for Control Center services only. In high-availability deployments, some application services perform more reliably when they run on master host nodes. In these cases, master host nodes require additional RAM and CPU resources.

Specifically, Zenoss applications include a database service that performs best on master host nodes. For more information, please contact your Zenoss representative.

Port requirements

In addition to the ports that are required for all Control Center deployments, which are described in the *Resource Manager Planning Guide*, the master host nodes of a high-availability deployment require the following ports:

INPUT chain

`tcp`: 7789, 2224

OUTPUT chain

`tcp`: 7789, 2224

`udp`: 5405

Configuring SELinux

Control Center is tested with SELinux in enforcing mode. However, the default configuration of SELinux prevents DRBD from operating correctly. This guide includes an optional procedure for creating policies that permit DRBD when SELinux is used in enforcing mode.

Key variables used in this guide

The following tables associate key features of a high-availability deployment with variables used in this guide.

Feature	**Variable Name**	
	Primary Node	**Secondary Node**
Public IP address of master node (static; known to all Control Center machines)	*Primary-Public-IP*	*Secondary-Public-IP*
Public hostname of master node (returned by `uname`; resolves to the public IP address)	*Primary-Public-Name*	*Secondary-Public-Name*
Private IP address of master node (static; dual-NIC systems only)	*Primary-Private-IP*	*Secondary-Private-IP*
Private hostname of master node (resolves to the private IP address; dual-NIC systems only)	*Primary-Private-Name*	*Secondary-Private-Name*

Feature	**Variable Name**
Virtual IP address of the high-availability cluster (static; known enterprise-wide)	*HA-Virtual-IP*
Virtual hostname of the high-availability cluster (known enterprise-wide)	*HA-Virtual-Name*
Mirrored storage for Control Center internal services data	*Isvcs-Storage*
Mirrored storage for application metadata	*Metadata-Storage*
Mirrored storage for application data	*App-Data-Storage*

Downloading and staging required files 2

This chapter describes how to download and install or stage Control Center software and its operating system dependencies. The procedures in this chapter are required to perform an installation.

The following table identifies where to perform each procedure in this chapter.

Procedure	**Where to perform**
Downloading Control Center files on page 12	A workstation with internet access
Installing the repository mirror on page 13	All Control Center hosts
Staging Docker image files on page 15	Both Control Center master host nodes
Staging a Docker image file on ZooKeeper ensemble nodes on page 15	Delegate hosts that are ZooKeeper ensemble nodes
Downloading and staging cluster software on page 15	An RHEL/CentOS system with internet access and the same operating system release and kernel as the master host nodes

Downloading Control Center files

To perform this procedure, you need:

- A workstation with internet access.
- Permission to download files from *delivery.zenoss.com*. Customers can request permission by filing a ticket at the *Zenoss Support* site.
- A secure network copy program.

Use this procedure to

- download the required files to a workstation
- copy the files to the hosts that need them

Perform these steps:

1 In a web browser, navigate to the download site, and then log in.

The download site is *delivery.zenoss.com*.

2 Download the self-installing Docker image files.

```
install-zenoss-serviced-isvcs-v61.run
install-zenoss-isvcs-zookeeper-v10.run
```

3 Download the Control Center RPM file.

```
serviced-1.5.1-1.x86_64.rpm
```

4 Identify the operating system release on Control Center hosts.

Enter the following command on each Control Center host in your deployment, if necessary. All Control Center hosts should be running the same operating system release and kernel.

```
cat /etc/redhat-release
```

5 Download the RHEL/CentOS repository mirror file for your deployment.

The download site provides a repository mirror file for each tested release of RHEL/CentOS. Each mirror file contains the release-specific packages that Control Center requires.

```
yum-mirror-centos7.2-1511-serviced-1.5.1.x86_64.rpm
yum-mirror-centos7.3-1611-serviced-1.5.1.x86_64.rpm
yum-mirror-centos7.4-1708-serviced-1.5.1.x86_64.rpm
```

6 Download the Pacemaker resource agents for Control Center.

```
serviced-resource-agents-1.1.0-1.x86_64.rpm
```

7 Optional: Download the Zenoss GNU Privacy Guard (GPG) key, if desired.

You can use the Zenoss GPG key to verify Zenoss RPM files and the `yum` metadata of the repository mirror.

a Download the key.

```
curl --location -o /tmp/tmp.html \
'http://keys.gnupg.net/pks/lookup?op=get&search=0xED0A5FD2AA5A1AD7'
```

b Determine whether the download succeeded.

```
grep -Ec '^\-\-\-\-\-BEGIN PGP' /tmp/tmp.html
```

- If the result is `0`, return to the previous substep.
- If the result is `1`, proceed to the next substep.

c Extract the key.

```
awk '/^-----BEGIN PGP.*$/,/^-----END PGP.*$/' \
  /tmp/tmp.html > ./RPM-GPG-KEY-Zenoss
```

8 Use a secure copy program to copy the files to Control Center hosts.

- Copy all files to both master nodes.
- Copy the RHEL/CentOS RPM file, the Control Center RPM file, and the Zenoss GPG key file to all delegate hosts.
- Copy the Docker image file for ZooKeeper to delegate hosts that are ZooKeeper ensemble nodes.

Installing the repository mirror

Use this procedure to install the Zenoss repository mirror on a Control Center host. The mirror contains packages that are required on all Control Center hosts.

1 Log in to the target host as `root`, or as a user with superuser privileges.

2 Move the RPM files and the Zenoss GPG key file to `/tmp`.

3 Install the repository mirror.

```
yum install /tmp/yum-mirror-*.rpm
```

The `yum` command copies the contents of the RPM file to `/opt/zenoss-repo-mirror`.

4 Optional: Install the Zenoss GPG key, and then test the package files, if desired.

- **a** Move the Zenoss GPG key to the mirror directory.

  ```
  mv /tmp/RPM-GPG-KEY-Zenoss /opt/zenoss-repo-mirror
  ```

- **b** Install the key.

  ```
  rpm --import /opt/zenoss-repo-mirror/RPM-GPG-KEY-Zenoss
  ```

- **c** Test the repository mirror package file.

  ```
  rpm -K /tmp/yum-mirror-*.rpm
  ```

  On success, the result includes the file name and the following information:

  ```
  (sha1) dsa sha1 md5 gpg OK
  ```

- **d** Test the Control Center package file.

  ```
  rpm -K /tmp/serviced-1.5.1-1.x86_64.rpm
  ```

5 Optional: Update the configuration file of the Zenoss repository mirror to enable GPG key verification, if desired.

- **a** Open the repository mirror configuration file (`/etc/yum.repos.d/zenoss-mirror.repo`) with a text editor, and then add the following lines to the end of the file.

  ```
  repo_gpgcheck=1
  gpgkey=file:///opt/zenoss-repo-mirror/RPM-GPG-KEY-Zenoss
  ```

- **b** Save the file, and then close the editor.
- **c** Update the `yum` metadata cache.

  ```
  yum makecache fast
  ```

  The cache update process includes the following prompt:

  ```
  Retrieving key from file:///opt/zenoss-repo-mirror/RPM-GPG-KEY-
  Zenoss
  Importing GPG key 0xAA5A1AD7:
   Userid     : "Zenoss, Inc. <dev@zenoss.com>"
   Fingerprint: f31f fd84 6a23 b3d5 981d a728 ed0a 5fd2 aa5a 1ad7
   From       : /opt/zenoss-repo-mirror/RPM-GPG-KEY-Zenoss
  Is this ok [y/N]:
  ```

  Enter `y`.

6 Move the Control Center package file to the mirror directory.

```
mv /tmp/serviced-1.5.1-1.x86_64.rpm /opt/zenoss-repo-mirror
```

7 Move the Pacemaker resource agents for Control Center to the mirror directory.

```
mv /tmp/serviced-resource-agents-1.1.0-1.x86_64.rpm \
  /opt/zenoss-repo-mirror
```

8 Optional: Delete the mirror package file, if desired.

```
rm /tmp/yum-mirror-*.rpm
```

Staging Docker image files

Before performing this procedure, verify that approximately 640MB of temporary space is available on the file system where `/root` is located.

Use this procedure to copy Docker image files to a Control Center host. The files are used when Docker is fully configured.

1 Log in to the master host as `root`, or as a user with superuser privileges.
2 Copy or move the archive files to `/root`.
3 Add execute permission to the files.

```
chmod +x /root/*.run
```

Staging a Docker image file on ZooKeeper ensemble nodes

Before performing this procedure, verify that approximately 170MB of temporary space is available on the file system where `/root` is located.

Use this procedure to add a Docker image file to the Control Center delegate hosts that are ZooKeeper ensemble nodes. Delegate hosts that are not ZooKeeper ensemble nodes do not need the file.

1 Log in to a delegate host as `root`, or as a user with superuser privileges.
2 Copy or move the `install-zenoss-isvcs-zookeeper-v10.run` file to `/root`.
3 Add execute permission to the file.

```
chmod +x /root/*.run
```

Downloading and staging cluster software

To perform this procedure, you need:

- An RHEL/CentOS system with internet access and the same operating system release and kernel as the master host nodes.
- A secure network copy program.

Use this procedure to download packages for Distributed Replicated Block Device (DRBD) and Pacemaker/Corosync, and to bundle them for installation on master host nodes.

1 Log in to a compatible host that is connected to the internet as `root`, or as a user with superuser privileges.

  The host must have the same operating system (RHEL or CentOS) and release installed, and the same version of the Linux kernel, as the master host nodes.
2 Install `yum` utilities, if necessary.

a Determine whether the `yum` utilities package is installed.

```
rpm -qa | grep yum-utils
```

- If the command returns a result, the package is installed. Proceed to the next step.
- If the command does not return a result, the package is not installed. Perform the following substep.

b Install the `yum-utils` package.

```
yum install yum-utils
```

3 Add the Enterprise Linux packages repository (ELRepo), if necessary.

a Determine whether the ELRepo repository is available.

```
yum repolist | grep elrepo
```

- If the command returns a result, the repository is available. Proceed to the next step.
- If the command does not return a result, the repository is not available. Perform the following substeps.

b Import the public key for the repository.

```
rpm --import https://www.elrepo.org/RPM-GPG-KEY-elrepo.org
```

c Add the repository to the download host.

```
rpm -Uvh \
  http://www.elrepo.org/elrepo-release-7.0-2.el7.elrepo.noarch.rpm
```

d Clean and update the `yum` caches.

```
yum clean all && yum makecache fast
```

4 Download the required packages and their dependencies, and then create a `tar` archive of the package files.

a Create a temporary directory for the packages.

```
mkdir /tmp/downloads
```

b Download the DRBD packages to the temporary directory.

```
repotrack -a x86_64 -r elrepo -p /tmp/downloads kmod-drbd84
```

c Download the Corosync/Pacemaker packages to the temporary directory.

```
repotrack -a x86_64 -p /tmp/downloads pcs
```

d Create a `tar` archive of the temporary directory.

```
cd /tmp && tar czf ./downloads.tgz ./downloads
```

5 Use a secure copy program to copy the packages archive to the `/tmp` directory of each master host node.

Installing a master host 3

This chapter describes how to install Control Center on a Red Hat Enterprise Linux (RHEL) or CentOS host. The candidate host must have the CPU, RAM, and storage resources required to serve as a Control Center master host node. Perform the procedures in this chapter on both of the candidate master host nodes of a high-availability cluster.

Verifying candidate host resources

Use this procedure to determine whether the hardware resources and installed operating system of a host are sufficient to serve as a Control Center master host.

1. Log in to the candidate host as `root`, or as a user with superuser privileges.
2. Verify that the host implements the 64-bit version of the x86 instruction set.

   ```
   uname -m
   ```

   - If the output is `x86_64`, the architecture is 64-bit. Proceed to the next step
   - If the output is `i386/i486/i586/i686`, the architecture is 32-bit. Stop this procedure and select a different host.
3. Determine whether the installed operating system release is one of the releases that has been tested with Control Center.

   ```
   cat /etc/redhat-release
   ```

   - If the result includes `7.2`, `7.3`, or `7.4` proceed to the next step.
   - If the result does not include `7.2`, `7.3`, or `7.4`, select a different host, and then start this procedure again.
4. Determine whether the CPU resources are sufficient.
   a. Display the total number of CPU cores.

      ```
      grep -Ec '^core id' /proc/cpuinfo
      ```

   b. Compare the available resources with the requirements for a Control Center master host.

      For more information, refer to the *Resource Manager Planning Guide* or the *Zenoss Community Edition (Core) Planning Guide*.

5 Determine whether the CPU resources support the AES instruction set.

```
grep -Ec '^flags.*aes' /proc/cpuinfo
```

For optimal performance, the result of the preceding commands must match the total number of CPU resources available on the host. If the result is 0, performance is severely degraded.

If the result is 0 and the candidate host is a virtual machine, the managing hypervisor may be configured in Hyper-V compatibility mode. Check the setting and disable it, if possible, or select a different host.

6 Determine whether the available memory and swap is sufficient.

a Display the available memory.

```
free -h
```

b Compare the available memory and swap space with the amount required for a master host in your deployment.

For more information, see *Master host storage requirements* on page 10.

If the result does not meet minimum requirements, stop this procedure and select a different host.

7 Ensure the host has a persistent numeric ID.

Skip this step if you are installing a single-host deployment.

Each Control Center host must have a unique host ID, and the ID must be persistent (not change when the host reboots).

```
test -f /etc/hostid || genhostid ; hostid
```

Record the ID for comparison with other Control Center hosts.

8 Verify that name resolution works on this host.

```
hostname -i
```

If the result is not a valid IPv4 address, add an entry for the host to the network nameserver, or to `/etc/hosts`.

9 Add an entry to `/etc/hosts` for localhost, if necessary.

a Determine whether `127.0.0.1` is mapped to `localhost`.

```
grep 127.0.0.1 /etc/hosts | grep localhost
```

If the preceding commands return no result, perform the following substep.

b Add an entry to `/etc/hosts` for `localhost`.

```
echo "127.0.0.1 localhost" >> /etc/hosts
```

10 Update the Linux kernel, if necessary.

a Determine which kernel version is installed.

```
uname -r
```

If the result is lower than `3.10.0-327.22.2.el7.x86_64`, perform the following substep.

b Update the kernel, and then restart the host.

The following commands require internet access or a local mirror of operating system packages.

```
yum makecache fast && yum update kernel && reboot
```

11 Display the available block storage on the candidate host.

```
lsblk -ap --output=NAME,SIZE,TYPE,FSTYPE,MOUNTPOINT
```

Compare the results with the storage requirements described in *Master host storage requirements* on page 10.

Preparing the master host operating system

Use this procedure to prepare a RHEL/CentOS host as a Control Center master host.

1 Log in to the candidate master host as `root`, or as a user with superuser privileges.

2 Disable the firewall, if necessary.

This step is required for installation but not for deployment. For more information, refer to the *Resource Manager Planning Guide* or the *Zenoss Community Edition (Core) Planning Guide*.

a Determine whether the `firewalld` service is enabled.

```
systemctl status firewalld.service
```

- If the result includes `Active: inactive (dead)`, the service is disabled. Proceed to the next step.
- If the result includes `Active: active (running)`, the service is enabled. Perform the following substep.

b Disable the `firewalld` service.

```
systemctl stop firewalld && systemctl disable firewalld
```

On success, the preceding commands display messages similar to the following example:

```
rm '/etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service'
rm '/etc/systemd/system/basic.target.wants/firewalld.service'
```

3 Optional: Enable persistent storage for log files, if desired.

By default, RHEL/CentOS systems store log data only in memory or in a ring buffer in the `/run/log/journal` directory. By performing this step, log data persists and can be saved indefinitely, if you implement log file rotation practices. For more information, refer to your operating system documentation.

Note The following commands are safe when performed during an installation, before Docker or Control Center are installed or running. To enable persistent log files after installation, stop Control Center, stop Docker, and then enter the following commands.

```
mkdir -p /var/log/journal && systemctl restart systemd-journald
```

4 Enable and start the Dnsmasq package.

The package facilitates networking among Docker containers.

```
systemctl enable dnsmasq && systemctl start dnsmasq
```

If name resolution in your environment relies solely on entries in `/etc/hosts`, configure `dsnmasq` so that containers can use the file:

a Open `/etc/dnsmasq.conf` with a text editor.

b Locate the line that starts with `#domain-needed`, and then make a copy of the line, immediately below the original.

- **c** Remove the number sign character (`#`) from the beginning of the line.
- **d** Locate the line that starts with `#bogus-priv`, and then make a copy of the line, immediately below the original.
- **e** Remove the number sign character (`#`) from the beginning of the line.
- **f** Locate the line that starts with `#local=/localnet/`, and then make a copy of the line, immediately below the original.
- **g** Remove `net`, and then remove the number sign character (`#`) from the beginning of the line.
- **h** Locate the line that starts with `#domain=example.com`, and then make a copy of the line, immediately below the original.
- **i** Replace `example.com` with `local`, and then remove the number sign character (`#`) from the beginning of the line.
- **j** Save the file, and then close the editor.
- **k** Restart the `dnsmasq` service.

```
systemctl restart dnsmasq
```

5 Add the required hostnames and IP addresses of both the primary and the secondary node to the `/etc/hosts` file.

For a dual-NIC system, replace each variable name with the values designated for each node, and replace `example.com` with the domain name of your organization:

```
echo "Primary-Public-IP Primary-Public-Name.example.com \
  Primary-Public-Name" >> /etc/hosts
echo "Primary-Private-IP Primary-Private-Name.example.com \
  Primary-Private-Name" >> /etc/hosts
echo "Secondary-Public-IP Secondary-Public-Name.example.com \
  Secondary-Public-Name" >> /etc/hosts
echo "Secondary-Private-IP Secondary-Private-Name.example.com \
  Secondary-Private-Name" >> /etc/hosts
```

For a single-NIC system, replace each variable name with the values designated for each node, and replace `example.com` with the domain name of your organization:

```
echo "Primary-Public-IP Primary-Public-Name.example.com \
  Primary-Public-Name" >> /etc/hosts
echo "Secondary-Public-IP Secondary-Public-Name.example.com \
  Secondary-Public-Name" >> /etc/hosts
```

6 Create a mount point for application data backups.

The default mount point is `/opt/serviced/var/backups`. You can change the default by editing the *SERVICED_BACKUPS_PATH* variable in the Control Center configuration file.

```
mkdir -p /opt/serviced/var/backups
```

7 Create a mount point for Control Center internal services data.

The default mount point is `/opt/serviced/var/isvcs`. You can change the default by editing the *SERVICED_ISVCS_PATH* variable in the Control Center configuration file.

```
mkdir -p /opt/serviced/var/isvcs
```

8 Create a mount point for Control Center audit logs.

The default mount point is `/var/log/serviced`. You can change the default by editing the *SERVICED_LOG_PATH* variable in the Control Center configuration file.

```
mkdir -p /var/log/serviced
```

9 Remove file system signatures from the required storage areas.

   Replace each variable name with the path of each storage area:

```
wipefs -a Isvcs-Storage
wipefs -a Metadata-Storage
wipefs -a App-Data-Storage
```

10 Reboot the host.

```
reboot
```

Configuring a private master NTP server

Control Center requires a common time source. The following procedure configures a private master *NTP* server to synchronize the system clocks of all Control Center hosts.

Note VMware vSphere guest systems can synchronize their system clocks with the host system. If that feature is enabled, it must be disabled to configure a private master NTP server. For more information, refer to the VMware documentation for your version of vSphere.

Installing Docker

Use this procedure to install Docker.

1. Log in to the host as `root`, or as a user with superuser privileges.
2. Install Docker CE 17.09.0 from the local repository mirror.
   a. Install Docker CE.

      ```
      yum install --enablerepo=zenoss-mirror docker-ce-17.09.0.ce
      ```

      If `yum` returns an error due to dependency issues, see *Resolving package dependency conflicts* on page 93 for potential resolutions.
   b. Enable automatic startup.

      ```
      systemctl enable docker
      ```

Installing Control Center

Use this procedure to install Control Center.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Install Control Center 1.5.1 from the local repository mirror.
   a. Clean the `yum` cache and update repository metadata.

      ```
      yum clean all && yum makecache fast
      ```

b Install Control Center.

```
yum install --enablerepo=zenoss-mirror \
  /opt/zenoss-repo-mirror/serviced-1.5.1-1.x86_64.rpm
```

3 Disable automatic startup of Control Center.

The cluster management software controls the `serviced` service.

```
systemctl disable serviced
```

4 Make a backup copy of the Control Center configuration file.

a Make a copy of `/etc/default/serviced`.

```
cp /etc/default/serviced /etc/default/serviced-1.5.1-orig
```

b Set the backup file permissions to read-only.

```
chmod 0440 /etc/default/serviced-1.5.1-orig
```

Configuring Docker and loading images

Use this procedure to configure Docker and load images in to the local repository.

1 Log in to the master host as `root`, or as a user with superuser privileges.

2 Create a symbolic link for the Docker temporary directory.

Docker uses its temporary directory to spool images. The default directory is `/var/lib/docker/tmp`. The following command specifies the same directory that Control Center uses, `/tmp`. You can specify any directory that has a minimum of 10GB of unused space.

a Create the `docker` directory in `/var/lib`.

```
mkdir /var/lib/docker
```

b Create the link to `/tmp`.

```
ln -s /tmp /var/lib/docker/tmp
```

3 Create a `systemd` drop-in file for Docker.

a Create the override directory.

```
mkdir -p /etc/systemd/system/docker.service.d
```

b Create the unit drop-in file.

```
cat <<EOF > /etc/systemd/system/docker.service.d/docker.conf
[Service]
TimeoutSec=300
EnvironmentFile=-/etc/sysconfig/docker
ExecStart=
ExecStart=/usr/bin/dockerd \$OPTIONS
TasksMax=infinity
EOF
```

  c. Reload the `systemd` manager configuration.

  ```
  systemctl daemon-reload
  ```

4. Create an LVM thin pool for Docker data.

   For more information about the `serviced-storage` command, see *serviced-storage* on page 89.

   To use an entire block device or partition for the thin pool, replace *Device-Path* with the device path:

   ```
   serviced-storage create-thin-pool docker Device-Path
   ```

   On success, the result is the device mapper name of the thin pool, which always starts with `/dev/mapper`.

5. Configure and start the Docker service.

   a. Create a variable for the name of the Docker thin pool.

      Replace *Thin-Pool-Device* with the name of the thin pool device created in the previous step:

      ```
      myPool="Thin-Pool-Device"
      ```

   b. Create variables for adding arguments to the Docker configuration file. The `--exec-opt` argument is a workaround for *a Docker issue* on RHEL/CentOS 7.x systems.

      ```
      myDriver="--storage-driver devicemapper"
      myLog="--log-level=error"
      myFix="--exec-opt native.cgroupdriver=cgroupfs"
      myMount="--storage-opt dm.mountopt=discard"
      myFlag="--storage-opt dm.thinpooldev=$myPool"
      ```

   c. Add the arguments to the Docker configuration file.

      ```
      echo 'OPTIONS="'$myLog $myDriver $myFix $myMount $myFlag'"' \
        >> /etc/sysconfig/docker
      ```

   d. Start or restart Docker.

      ```
      systemctl restart docker
      ```

      The startup may take up to a minute, and may fail. If startup fails, repeat the `restart` command.

6. Configure name resolution in containers.

   Each time it starts, `docker` selects an IPv4 subnet for its virtual Ethernet bridge. The selection can change; this step ensures consistency.

   a. Identify the IPv4 subnet and netmask `docker` has selected for its virtual Ethernet bridge.

      ```
      ip addr show docker0 | grep inet
      ```

   b. Open `/etc/sysconfig/docker` in a text editor.

   c. Add the following flags to the end of the *OPTIONS* declaration.

      Replace *Bridge-Subnet* with the IPv4 subnet `docker` selected for its virtual bridge:

      ```
      --dns=Bridge-Subnet --bip=Bridge-Subnet/16
      ```

      For example, if the bridge subnet is 172.17.0.1, add the following flags:

      ```
      --dns=172.17.0.1 --bip=172.17.0.1/16
      ```

Note Use a space character ( ) to separate flags, and make sure the double quote character (`"`) delimits the declaration of *OPTIONS*.

---

d Save the file, and then close the editor.

e Restart the Docker service.

```
systemctl restart docker
```

7 Import the Control Center images into the local Docker repository.

The images are contained in the self-extracting archive files that are staged in `/root`.

a Change directory to `/root`.

```
cd /root
```

b Extract the images.

```
for image in install-zenoss-*.run
do
  /bin/echo -n "$image: "
  ./$image
done
```

Image extraction begins when you press **y**. If you press **y** and then **Enter**, the current image is extracted, but the next one is not.

c Optional: Delete the archive files, if desired.

```
rm -i ./install-zenoss-*.run
```

8 Stop and disable the Docker service.

The cluster management software controls the Docker service.

```
systemctl stop docker && systemctl disable docker
```

Configuring DRBD 4

The procedures in this chapter configure LVM and DRBD for a two-node high-availability cluster.

The following list identifies the assumptions that inform the DRBD resource definition for Control Center:

- Each node has either one or two NICs. In dual-NIC hosts the private IP/hostnames are reserved for DRBD traffic. This is recommended configuration, which enables real-time writes for disk synchronization between the active and passive nodes, and no contention with application traffic. However, it is possible to use DRBD with a single NIC.
- All volumes should synchronize and failover together. This is accomplished by creating a single resource definition.
- DRBD stores its metadata on each volume (`meta-disk/internal`), so the total amount of space reported on the logical device `/dev/drbd`*`n`* is always less than the amount of physical space available on the underlying primary partition.
- The `syncer/rate` key controls the rate, in bytes per second, at which DRBD synchronizes disks. Set the rate to 30% of the available replication bandwidth, which is the slowest of either the I/O subsystem or the network interface. The following example assumes 100MB/s available for total replication bandwidth (0.30 * 100MB/s = 30MB/s).

Installing cluster management

Perform this procedure to install the cluster management software.

1. Log in to the primary node as `root`, or as a user with superuser privileges.
2. In a separate window, log in to the secondary node as `root`, or as a user with superuser privileges.
3. On both nodes, extract and install the cluster management packages.
   a. Extract the packages.

      ```
      cd /tmp && tar xzf ./downloads.tgz
      ```

   b. Install the packages.

      ```
      yum install ./downloads/*.rpm
      ```

4. On both nodes, install the Pacemaker resource agent for Control Center.

Pacemaker uses resource agents (scripts) to implement a standardized interface for managing arbitrary resources in a cluster. Zenoss provides a Pacemaker resource agent to manage the Control Center master host.

```
yum install \
  /opt/zenoss-repo-mirror/serviced-resource-agents-1.1.0-1.x86_64.rpm
```

5 On both nodes, start and enable the PCS daemon.

```
systemctl start pcsd.service && systemctl enable pcsd.service
```

6 On both nodes, set the password of the `hacluster` account.

The Pacemaker package creates the `hacluster` user account, which must have the same password on both nodes.

```
passwd hacluster
```

Configuring Logical Volume Manager

Control Center application data is managed by a device mapper thin pool created with Logical Volume Manager (LVM). This procedure adjusts the LVM configuration for mirroring by DRBD.

Perform this procedure on the primary node and on the secondary node.

1 Log in to the host as `root`, or as a user with superuser privileges.
2 Create a backup copy of the LVM configuration file.

```
cp -p /etc/lvm/lvm.conf /etc/lvm/lvm.conf.bak
```

3 Open `/etc/lvm/lvm.conf` with a text editor.
4 Edit the `devices/filter` configuration option to exclude the partition for Control Center application data.

   a Search for the following text, which marks the beginning of the section about the `devices/filter` configuration option:

   ```
   # Configuration option devices/filter
   ```

   b At the end of section, remove the comment character (`#`) from the beginning of the following line:

   ```
   # filter = [ "a|.*/|" ]
   ```

   The line to edit is about 30 lines below the beginning the section.

   c Exclude the partition for Control Center application data.
   Replace *App-Data-Storage* with the path of the block storage designated for Control Center application data:

   ```
   filter = ["r|App-Data-Storage|"]
   ```

   For example, if the value of *App-Data-Storage* in your environment is `/dev/sdd`, the result should look like the following line:

   ```
   filter = ["r|/dev/sdd|"]
   ```

5 Edit the `devices/write_cache_state` configuration option to disable caching.

- **a** Search for the following text, which marks the beginning of the section about the `devices/write_cache_state` configuration option:

  ```
  # Configuration option devices/write_cache_state
  ```

- **b** Set the value of the `write_cache_state` option to 0.
  The result should look like the following line:

  ```
  write_cache_state = 0
  ```

6 Edit the `global/use_lvmetad` configuration option to disable the metadata daemon.

- **a** Search for the following text, which marks the beginning of the section about the `global/use_lvmetad` configuration option:

  ```
  # Configuration option global/use_lvmetad
  ```

  The line to edit is about 27 lines below the beginning the section.
- **b** Set the value of the `use_lvmetad` option to 0.
  The result should look like the following line:

  ```
  use_lvmetad = 0
  ```

7 Save the file and close the text editor.

8 Delete any stale cache entries.

```
rm -f /etc/lvm/cache/.cache
```

9 Restart the host.

```
reboot
```

Configuring DRBD

This procedure configures DRBD for deployments with either one or two NICs in each node.

1 Log in to the primary node as `root`, or as a user with superuser privileges.

2 In a separate window, log in to the secondary node as `root`, or as a user with superuser privileges.

3 On both nodes, identify the storage areas to use.

```
lsblk --output=NAME,SIZE
```

Record the paths of the storage areas in the following table. The information is needed in subsequent steps and procedures.

Node	***Isvcs-Storage***	***Metadata-Storage***	***App-Data-Storage***

4 On both nodes, edit the DRBD configuration file.

- **a** Open `/etc/drbd.d/global_common.conf` with a text editor.

b Add the following values to the `global` and `common/net` sections of the file.

```
global {
  usage-count yes;
}
common {
  net {
    protocol C;
  }
}
```

c Save the file, and then close the editor.

5 Optional: On both nodes, add SELinux policies for DRBD, if necessary.

Perform this step only if you are using SELinux in enforcing mode.

a Determine whether DRBD security failures are present in the audit log.

```
grep drbd /var/log/audit/audit.log
```

If the command returns results similar to the following example, continue with the remaining substeps.

```
type=AVC msg=audit(1316737884.896:27): avc:  denied
  { module_request } for  pid=1253 comm="drbdsetup"
 kmod="hmac(sha1)" scontext=system_u:system_r:drbd_t:s0
 tcontext=system_u:system_r:kernel_t:s0 tclass=system
```

b Determine whether `auditd` is running.

```
ps -ef | grep auditd | grep -v grep
```

c Create policies based on the security failures in the log file.
If `auditd` is running, enter the following command:

```
grep drbd /var/log/audit/audit.log | audit2allow -a -l -M localdrbd
```

If `auditd` is not running, enter the following command:

```
grep drbd /var/log/audit/audit.log | audit2allow -d -l -M localdrbd
```

d Make the new policy active.

```
semodule -i localdrbd.pp
```

6 On both nodes, create a resource definition for Control Center.

a Open `/etc/drbd.d/serviced-dfs.res` with a text editor.

b **For a dual-NIC system**, add the following content to the file.

Replace the variables in the content with the actual values for your environment:

```
resource serviced-dfs {
   volume 0 {
      device /dev/drbd0;
      disk Isvcs-Storage;
      meta-disk internal;
   }
   volume 1 {
      device /dev/drbd1;
      disk Metadata-Storage;
```

```
         meta-disk internal;
      }
      volume 2 {
         device /dev/drbd2;
         disk App-Data-Storage;
         meta-disk internal;
      }
      syncer   {
         rate 30M;
      }
      net      {
         after-sb-0pri discard-zero-changes;
         after-sb-1pri discard-secondary;
      }
      on Primary-Public-Name {
         address Primary-Private-IP:7789;
      }
      on Secondary-Public-Name {
         address Secondary-Private-IP:7789;
      }
   }
```

c **For a single-NIC system**, add the following content to the file.

Replace the variables in the content with the actual values for your environment:

```
resource serviced-dfs {
   volume 0 {
      device /dev/drbd0;
      disk Isvcs-Storage;
      meta-disk internal;
   }
   volume 1 {
      device /dev/drbd1;
      disk Metadata-Storage;
      meta-disk internal;
   }
   volume 2 {
      device /dev/drbd2;
      disk App-Data-Storage;
      meta-disk internal;
   }
   syncer   {
      rate 30M;
   }
   net      {
      after-sb-0pri discard-zero-changes;
      after-sb-1pri discard-secondary;
   }
   on Primary-Public-Name {
      address Primary-Public-IP:7789;
   }
   on Secondary-Public-Name {
      address Secondary-Public-IP:7789;
   }
}
```

d Save the file, and then close the editor.

7 On both nodes, create device metadata and enable the new DRBD resource.

```
drbdadm create-md all && drbdadm up all
```

Initializing DRBD

Perform this procedure to initialize DRBD and the mirrored storage areas.

Note Unlike the preceding procedures, most of the steps in this procedure are performed on the primary node only.

1. Log in to the primary node as `root`, or as a user with superuser privileges.
2. Synchronize the storage areas of both nodes.
   a. Start the synchronization.

      ```
      drbdadm primary --force serviced-dfs
      ```

      The command may return right away, while the synchronization process continues running in the background. Depending on the sizes of the storage areas, this process can take several hours.
   b. Monitor the progress of the synchronization.

      ```
      drbd-overview
      ```

      Do not proceed until the status is `UpToDate/UpToDate`, as in the following example output:

      ```
      0:serviced-dfs/0 Connected Primary/Secondary UpToDate/UpToDate
      1:serviced-dfs/1 Connected Primary/Secondary UpToDate/UpToDate
      2:serviced-dfs/2 Connected Primary/Secondary UpToDate/UpToDate
      ```

      The `Primary/Secondary` values show that the command was run on the primary node; otherwise, the values are `Secondary/Primary`. Likewise, the first value in the `UpToDate/UpToDate` field is the status of the node on which the command is run, and the second value is the status of the remote node.
3. Format the block storage for Control Center internal services data and for Control Center metadata.

   The following commands use the paths of the DRBD devices defined previously, not the block storage paths.

   ```
   mkfs.xfs /dev/drbd0
   mkfs.xfs /dev/drbd1
   ```

   The commands create XFS filesystems on the primary node, and DRBD mirrors the filesystems to the secondary node.
4. Create a device mapper thin pool for Control Center application data.

   The following command uses the path of the DRBD device defined previously, not the block storage path.

   ```
   serviced-storage -v create-thin-pool serviced /dev/drbd2
   ```

   On success, DRBD mirrors the device mapper thin pool to the secondary node.
5. Identify the size of the thin pool for application data.

   The size is required to set an accurate value for the *SERVICED_DM_BASESIZE* variable.

   ```
   lvs --options=lv_name,lv_size | grep serviced-pool
   ```
6. Configure Control Center storage variables.
   a. Open `/etc/default/serviced` in a text editor.

- **b** Locate the line for the *SERVICED_FS_TYPE* variable, and then make a copy of the line, immediately below the original.
- **c** Remove the number sign character (#) from the beginning of the line.
- **d** Locate the line for the *SERVICED_DM_THINPOOLDEV* variable, and then make a copy of the line, immediately below the original.
- **e** Remove the number sign character (#) from the beginning of the line.
- **f** Set the value to the device mapper name of the thin pool for application data.

  ```
  SERVICED_DM_THINPOOLDEV=/dev/mapper/serviced-serviced--pool
  ```
- **g** Locate the line for the *SERVICED_DM_BASESIZE* variable, and then make a copy of the line, immediately below the original.
- **h** Remove the number sign character (#) from the beginning of the line.
- **i** Change the value, if necessary.
  Replace *Fifty-Percent* with the value that is less than or equal to 50% of the size of the thin pool for application data. Include the symbol for gigabytes, `G`:

  ```
  SERVICED_DM_BASESIZE=Fifty-PercentG
  ```
- **j** Save the file, and then close the editor.

7 In a separate window, log in to the secondary node as `root`, or as a user with superuser privileges, and then replicate the Control Center configuration on the secondary node.

Use a utility like `sum` to compare the files and ensure their contents are identical.

8 On the primary node, monitor the progress of the synchronization.

```
drbd-overview
```

Note Do not proceed until synchronization is complete.

9 On the primary node, deactivate the `serviced` volume group.

```
vgchange -an serviced
```

10 On both nodes, stop DRBD.

```
drbdadm down all
```

Configuring Control Center on master host nodes 5

This chapter includes the procedures for configuring Control Center on master host nodes, and describes the configuration options that apply to master hosts.

Many configuration choices depend on application requirements. Please review your application documentation before configuring Control Center.

Note Perform the procedures in this chapter on the primary node and the secondary node.

This chapter includes synopses of the configuration variables that affect the master host. For more information about a variable, see *Control Center configuration variables* on page 96.

Control Center maintenance scripts on the master host

The scripts in the following list are installed when Control Center is installed, and are started either daily or weekly by `anacron`.

`/etc/cron.hourly/serviced`

This script invokes `logrotate` hourly, to manage the files in `/var/log/serviced`.

This script is required on the master host only.

`/etc/cron.weekly/serviced-fstrim`

This script invokes `fstrim` weekly, to reclaim unused blocks in the application data thin pool.

The life span of a solid-state drive (SSD) degrades when `fstrim` is run too frequently. If the block storage of the application data thin pool is an SSD, you can reduce the frequency at which this script is invoked, as long as the thin pool never runs out of free space. An identical copy of this script is located in `/opt/serviced/bin`.

This script is required on the master host only.

`/etc/cron.d/cron_zenossdbpack`

This script invokes `/opt/serviced/bin/serviced-zenossdbpack`, the database maintenance script for a Zenoss application, every Sunday at midnight. If the Zenoss application is not installed or is offline, the command fails. You can change the day of the week and time of day when the maintenance script is invoked by editing `/etc/cron.d/cron_zenossdbpack`.

This script is required on one host in the resource pool in which the Zenoss database services run. For more information, refer to the configuration guide for your Zenoss application.

User access control

Control Center provides a browser interface and a command-line interface.

To gain access to the Control Center browser interface, users must have login accounts on the Control Center master host. In addition, users must be members of the Control Center browser interface access group, which by default is the system group, `wheel`. To enhance security, you may change the browser interface access group from `wheel` to any other group.

To use the Control Center command-line interface (CLI) on a Control Center host, a user must have login account on the host, and the account must be a member of the `serviced` group. The `serviced` group is created when the Control Center RPM package is installed.

Note You can use two different groups to control access to the browser interface and the CLI. You can enable access to both interfaces for the same users by choosing the `serviced` group as the browser interface access group.

Pluggable Authentication Modules (PAM) has been tested and is recommended for enabling access to both the browser interface and the command-line interface. However, the PAM configuration must include the `sudo` service. Control Center relies on the host's `sudo` configuration, and if no configuration is present, PAM defaults to the configuration for `other`, which is typically too restrictive for Control Center users. For more information about configuring PAM, refer to your operating system documentation.

Adding users to the default browser interface access group

Use this procedure to add users to the default browser interface access group of Control Center, `wheel`.

Note Perform this procedure or the next procedure, but not both.

1. Log in to the host as `root`, or as a user with superuser privileges.
2. Add a user to the `wheel` group.

   Replace *User* with the name of a login account on the master host.

   ```
   usermod -aG wheel User
   ```

   Repeat the preceding command for each user to add.

Configuring a regular group as the Control Center browser interface access group

Use this procedure to change the default browser interface access group of Control Center from `wheel` to a non-system group.

The following Control Center variables are used in this procedure:

SERVICED_ADMIN_GROUP

Default: `wheel`

The name of the Linux group on the `serviced` master host whose members are authorized to use the `serviced` browser interface. You may replace the default group with a group that does not have superuser privileges.

SERVICED_ALLOW_ROOT_LOGIN

Default: `1` (true)

Determines whether the `root` user account on the `serviced` master host may be used to gain access to the `serviced` browser interface.

Note Perform this procedure or the previous procedure, but not both.

1. Log in to the host as `root`, or as a user with superuser privileges.
2. Create a variable for the group to designate as the administrative group.

   In this example, the group is `ccuser`. You may choose a different group, or choose the `serviced` group. (Choosing the serviced group allows all browser interface users to use the CLI.)

   ```
   myGROUP=ccuser
   ```
3. Create a new group, if necessary.

   ```
   groupadd $myGROUP
   ```
4. Add one or more existing users to the group.

   Replace *User* with the name of a login account on the host:

   ```
   usermod -aG $myGROUP User
   ```

   Repeat the preceding command for each user to add.
5. Specify the new administrative group in the `serviced` configuration file.
   a. Open `/etc/default/serviced` in a text editor.
   b. Locate the line for the *SERVICED_ADMIN_GROUP* variable, and then make a copy of the line, immediately below the original.
   c. Remove the number sign character (`#`) from the beginning of the line.
   d. Change the value from `wheel` to the name of the group you chose earlier.
   e. Save the file, and then close the editor.
6. Optional: Prevent the `root` user from gaining access to the Control Center browser interface, if desired.
   a. Open `/etc/default/serviced` in a text editor.
   b. Locate the line for the *SERVICED_ALLOW_ROOT_LOGIN* variable, and then make a copy of the line, immediately below the original.
   c. Remove the number sign character (`#`) from the beginning of the line.
   d. Change the value from `1` to `0`.
   e. Save the file, and then close the editor.

Enabling use of the command-line interface

Use this procedure to enable a user to perform administrative tasks with the Control Center command-line interface.

1. Log in to the host as `root`, or as a user with superuser privileges.
2. Add a user to the `serviced` group.

   Replace *User* with the name of a login account on the host.

   ```
   usermod -aG serviced User
   ```

   Repeat the preceding command for each user to add.

Setting the host role to master

Use this procedure to configure a host as the master host. The following configuration variable assigns the host role:

SERVICED_MASTER

Default: `1` (true)

Assigns the role of a `serviced` instance, either master or delegate. The master runs the application services scheduler and other internal services. Delegates run the application services assigned to the resource pool to which they belong.

Only one `serviced` instance can be the master; all other instances must be delegates. The default value assigns the master role. To assign the delegate role, set the value to `0` (false). This variable must be explicitly set on all Control Center hosts.

Perform these steps:

1. Log in to the host as `root`, or as a user with superuser privileges.
2. Edit the Control Center configuration file.
   a. Open `/etc/default/serviced` in a text editor.
   b. Locate the line for the *SERVICED_MASTER* variable, and then make a copy of the line, immediately below the original.
   c. Remove the number sign character (`#`) from the beginning of the line.
   d. Save the file, and then close the editor.
3. Verify the settings in the `serviced` configuration file.

```
grep -E '^\b*[A-Z_]+' /etc/default/serviced
```

Configuring the local Docker registry

Use this procedure to configure the endpoint of the local Docker registry.

The following configuration variable identifies the local Docker registry endpoint:

SERVICED_DOCKER_REGISTRY

Default: `localhost:5000`

The endpoint of the local Docker registry, which `serviced` uses to store internal services and application images.

If the default value is changed, the host's Docker configuration file must include the `--insecure-registry` flag with the same value as this variable.

The safest replacement for `localhost` is the IPv4 address of the registry host. Otherwise, the fully-qualified domain name of the host must be specified.

Perform these steps:

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Edit the Control Center configuration file.
   a. Open `/etc/default/serviced` in a text editor.
   b. Locate the line for the *SERVICED_DOCKER_REGISTRY* variable, and then make a copy of the line, immediately below the original.
   c. Remove the number sign character (`#`) from the beginning of the line.
   d. Replace `localhost` with the virtual IP address of the high-availability cluster (*HA-Virtual-IP*).

The new variable value must include the delimiter (`:`) followed by the port number.

e Save the file, and then close the editor.

3 Verify the settings in the `serviced` configuration file.

```
grep -E '^\b*[A-Z_]+' /etc/default/serviced
```

4 Add the insecure registry flag to the Docker configuration file.

a Open `/etc/sysconfig/docker` in a text editor.

b Add the local Docker registry endpoint to the end of the *OPTIONS* declaration.

Replace *Registry-Endpoint* with the same endpoint that is the value of the *SERVICED_DOCKER_REGISTRY* variable:

```
--insecure-registry=Registry-Endpoint
```

Note Use a space character ( ) to separate flags, and make sure the double quote character (`"`) delimits the declaration of *OPTIONS*.

c Save the file, and then close the editor.

Configuring endpoints

Use this procedure to customize the internal services endpoints of Control Center. The following configuration variables specify the endpoints:

SERVICED_ZK

Default: (none)

The list of endpoints in the `serviced` ZooKeeper ensemble, separated by the comma character (`,`). Each endpoint identifies an ensemble node. Each Control Center server and in-container proxy uses *SERVICED_ZK* to create a randomized, round-robin list, and cycles through the list when it attempts to establish a connection with the lead ZooKeeper host.

SERVICED_ENDPOINT

Default: `{{SERVICED_MASTER_IP}}:4979`

The endpoint of the `serviced` RPC server. Replace `{{SERVICED_MASTER_IP}}` with the IP address or hostname of the `serviced` master host. The port number of this endpoint must match the value of the *SERVICED_RPC_PORT* variable defined on the `serviced` master host.

SERVICED_LOG_ADDRESS

Default: `{{SERVICED_MASTER_IP}}:5042`

The endpoint of the logstash service. Replace `{{SERVICED_MASTER_IP}}` with the IP address or hostname of the `serviced` master host.

SERVICED_LOGSTASH_ES

Default: `{{SERVICED_MASTER_IP}}:9100`

The endpoint of the Elasticsearch service for logstash. On delegate hosts, replace `{{SERVICED_MASTER_IP}}` with the IP address or hostname of the Elasticsearch host, which by default is the `serviced` master host.

SERVICED_STATS_PORT

Default: `{{SERVICED_MASTER_IP}}:8443`

The endpoint of the `serviced` metrics consumer service. Replace `{{SERVICED_MASTER_IP}}` with the IP address or hostname of the `serviced` master host.

Perform these steps:

1 Log in to the host as `root`, or as a user with superuser privileges.
2 Edit the Control Center configuration file.
   a Open `/etc/default/serviced` in a text editor.
   b For each endpoint variable, locate the line that sets the variable, and then make a copy of the line, immediately below the original.
   c Remove the number sign character (`#`) from the beginning of the line.
   d Replace `{{SERVICED_MASTER_IP}}` with the virtual IP address of the high-availability cluster (*HA-Virtual-IP*).
   e Save the file, and then close the editor.
3 Verify the settings in the `serviced` configuration file.

```
grep -E '^\b*[A-Z_]+' /etc/default/serviced
```

Configuring the cluster virtual IP address

Use this procedure to configure a host with the IPv4 address of the high-availability cluster (*HA-Virtual-IP*). The following configuration variable specifies the address:

SERVICED_OUTBOUND_IP

Default: (none)

The IPv4 address that delegates use to connect to the master host. When no address is specified, `serviced` attempts to discover its public IP address by pinging `google.com`.

This variable must be set on all Control Center hosts in either of the following scenarios:

- Control Center is deployed behind a firewall and `google.com` is not reachable. Set the value to the IPv4 address of the master host.
- Control Center is deployed in a high-availability cluster. Set the value to the virtual IPv4 address of the high-availability cluster (*HA-Virtual-IP*).

Note Setting the Docker *HTTP_PROXY* or *HTTPS_PROXY* environment variables prevents access to the IP address defined with this variable. To enable access, unset the Docker variables, and then reboot the host.

Perform these steps:

1 Log in to the host as `root`, or as a user with superuser privileges.
2 Edit the Control Center configuration file.
   a Open `/etc/default/serviced` in a text editor.
   b Locate the line for the *SERVICED_OUTBOUND_IP* variable, and then make a copy of the line, immediately below the original.
   c Remove the number sign character (`#`) from the beginning of the line.
   d Change the value to the virtual IP address of the high-availability cluster (*HA-Virtual-IP*).
   e Save the file, and then close the editor.
3 Verify the settings in the `serviced` configuration file.

```
grep -E '^\b*[A-Z_]+' /etc/default/serviced
```

Master host configuration variables

The tables in this section provide an overview of the `serviced` configuration variables that apply to the Control Center master host. Set these variables as required for your environment or applications.

Best practices for configuration files

The Control Center configuration file, `/etc/default/serviced`, contains Bash environment variables that are read by the `serviced` daemon startup script. The following list describes recommended best practices for its use and maintenance:

1. When in doubt, make a backup. Before editing, making a backup copy of the configuration file is always the safest choice.
2. Copy a variable, then edit the copy. If you need to revert a variable to its default value, you don't have to leave the file to look it up.
3. Copy and edit a variable only if the default value needs to be changed. It's easier to troubleshoot problems when only non-default variables are copied and edited.
4. Put the first character of the variable declaration in the first column of its line. It's easier to `grep` for settings when each one starts a line.
5. Add customizations to the top of the file. Customizations at the end of the file or scattered throughout the file may be overlooked.
6. In high-availability deployments, the contents of `/etc/default/serviced` on the master nodes must be identical. Use a utility like `sum` to compare the files quickly.

RPC service variables

The variables in the following table must be set identically on all Control Center hosts, except:

- *SERVICED_RPC_PORT*, set only on the master
- *SERVICED_MAX_RPC_CLIENTS*, set only on delegates

By default, `serviced` uses TLS to encrypt all RPC traffic. The *SERVICED_KEY_FILE* and *SERVICED_CERT_FILE* variables identify the digital certificate used for RPC, mux, and HTTP traffic.

The *SERVICED_AUTH_TOKEN_EXPIRATION* variable affects RPC, mux, and internal services endpoint traffic.

Variable	Where to set
SERVICED_ENDPOINT	Master, delegates
SERVICED_MAX_RPC_CLIENTS	Delegates
SERVICED_RPC_PORT	Master
SERVICED_RPC_CERT_VERIFY	Master, delegates
SERVICED_RPC_DISABLE_TLS	Master, delegates
SERVICED_RPC_TLS_MIN_VERSION	Master, delegates
SERVICED_RPC_TLS_CIPHERS	Master, delegates
SERVICED_KEY_FILE	Master
SERVICED_CERT_FILE	Master
SERVICED_RPC_DIAL_TIMEOUT	Master, delegates

Variable	**Where to set**
SERVICED_AUTH_TOKEN_EXPIRATION	Master

Multiplexer variables

The variables in the following table must be set identically on all Control Center hosts.

By default, `serviced` uses TLS to encrypt all mux traffic. The *SERVICED_KEY_FILE* and *SERVICED_CERT_FILE* variables identify the digital certificate used for RPC, mux, and HTTP traffic.

The *SERVICED_AUTH_TOKEN_EXPIRATION* variable affects RPC, mux, and internal services endpoint traffic.

Variable	**Where to set**
SERVICED_MUX_PORT	Master, delegates
SERVICED_MUX_DISABLE_TLS	Master, delegates
SERVICED_MUX_TLS_MIN_VERSION	Master, delegates
SERVICED_MUX_TLS_CIPHERS	Master, delegates
SERVICED_KEY_FILE	Master
SERVICED_CERT_FILE	Master
SERVICED_AUTH_TOKEN_EXPIRATION	Master

HTTP server variables

The variables in the following table are set only on the master host, except the *SERVICED_UI_PORT* variable, which must be set identically on all Control Center hosts.

By default, `serviced` uses TLS to encrypt all HTTP traffic. The *SERVICED_KEY_FILE* and *SERVICED_CERT_FILE* variables identify the digital certificate used for RPC, mux, and HTTP traffic.

Variable	**Description**
SERVICED_UI_PORT	The port on which the HTTP server listens for requests.
SERVICED_TLS_MIN_VERSION	The minimum version of TLS that `serviced` accepts for HTTP traffic.
SERVICED_TLS_CIPHERS	The list TLS ciphers that `serviced` accepts for HTTP traffic.
SERVICED_KEY_FILE	The path of a digital certificate key file.
SERVICED_CERT_FILE	The path of a digital certificate file.

Browser interface variables (master host only)

The variables in the following table are set only on the master host.

Variable	**Description**
SERVICED_UI_POLL_FREQUENCY	The number of seconds between polls from browser interface clients.
SERVICED_SVCSTATS_CACHE_TIMEOUT	The number of seconds to cache statistics about services.
SERVICED_ADMIN_GROUP	The group on the master host whose members can use the browser interface.
SERVICED_ALLOW_ROOT_LOGIN	Determines whether `root` on the master host can use the browser interface.

Tuning variables (master host only)

Variable	**Description**
SERVICED_ES_STARTUP_TIMEOUT	The number of seconds to wait for the Elasticsearch service to start.
SERVICED_MASTER_POOLID	The name of the default resource pool. This variable is only used the first time `serviced` is started.

Universal configuration variables

The tables in this section provide an overview of the `serviced` configuration variables that apply to all Control Center hosts. Set these variables as required for your environment or applications.

Browser interface variable (all hosts)

Variable	**Description**
SERVICED_UI_PORT	The port on which the HTTP server listens for requests.

Networking variables

Variable	**Description**
SERVICED_STATIC_IPS	A list of one or more static IP addresses for IP assignment.
SERVICED_OUTBOUND_IP	The IP address of the network interface for `serviced` to use. When this variable is not set, `serviced` uses the IP address of the default network interface and assumes it has internet access. To prevent `serviced` from assuming it has internet access, set this variable.
SERVICED_VIRTUAL_ADDRESS_SUBNET	The private network for containers that use virtual IP addresses. The default is `10.3.0.0/16`, and the network can be unique on each host. A `/29` network is sufficient.
SERVICED_DOCKER_DNS	A list of one or more DNS servers. The list is injected into all Docker containers.

Debugging variables

Variable	Description
SERVICED_LOG_LEVEL	The log level `serviced` uses when writing to the system log.
SERVICED_DEBUG_PORT	The port on which `serviced` listens for HTTP requests for the *Go profiler*.
SERVICED_DOCKER_LOG_DRIVER	The log driver for all Docker container logs.
SERVICED_DOCKER_LOG_CONFIG	Docker `--log-opt` options.

Tuning variables (all Control Center hosts)

Variable	Description
GOMAXPROCS	The maximum number of CPU cores that `serviced` uses.
SERVICED_MAX_CONTAINER_AGE	The number of seconds `serviced` waits before removing a stopped container.
SERVICED_ISVCS_ENV_[0-9]+	Startup arguments to pass to specific internal services.
SERVICED_SERVICE_MIGRATION_TAG	Overrides the default value for the service migration image.
SERVICED_OPTS	Startup arguments for `serviced`.
SERVICED_CONTROLLER_BINARY	The path of the `serviced-controller` binary.
SERVICED_HOME	The path of the home directory for `serviced`.
SERVICED_ETC_PATH	The path of the directory for `serviced` configuration files.
SERVICED_VHOST_ALIASES	A list of hostname aliases for a host; for example, `localhost`.
SERVICED_ZK_CONNECT_TIMEOUT	The number of seconds Control Center waits for a connection to the lead ZooKeeper host.
SERVICED_ZK_PER_HOST_CONNECT_DELAY	The number of seconds Control Center waits before attempting to connect to the next host in its round-robin list of ZooKeeper hosts.
SERVICED_ZK_RECONNECT_START_DELAY, *SERVICED_ZK_RECONNECT_MAX_DELAY*	These are used together when Control Center is unable to re-establish a connection with the lead ZooKeeper host.

Cluster management software 6

Pacemaker is an open source cluster resource manager, and Corosync is a cluster infrastructure application for communication and membership services. The Pacemaker/Corosync daemon (`pcs.d`) communicates across nodes in the cluster. When `pcs.d` is installed, started, and configured, the majority of PCS commands can be run on either node in the cluster.

Creating the cluster in standby mode

Perform this procedure to create the high-availability cluster in standby mode.

1. Log in to the primary node as `root`, or as a user with superuser privileges.
2. Authenticate the nodes.

   ```
   pcs cluster auth Primary-Public-Name Secondary-Public-Name
   ```

   When prompted, enter the password of the `hacluster` account.
3. Generate and synchronize an initial (empty) cluster definition.

   ```
   pcs cluster setup --name serviced-ha \
     Primary-Public-Name Secondary-Public-Name
   ```
4. Start the PCS management agents on both nodes in the cluster.

   The cluster definition is empty, so starting the cluster management agents has no side effects.

   ```
   pcs cluster start --all
   ```

   The cluster management agents start, on both nodes.
5. Check the status.

   ```
   pcs cluster status
   ```

   The expected result is `Online`, for both nodes.
6. Put the cluster in standby mode.

   Pacemaker begins monitoring and managing the different resources as they are defined, which can cause problems; standby mode prevents the problems.

   ```
   pcs cluster standby --all
   ```
7. Configure cluster services to start when the node starts.

For more information about cluster startup options, refer to the *Pacemaker documentation*.

```
systemctl enable corosync; systemctl enable pacemaker
```

8. Replicate the configuration on the secondary node.
   a. In a separate window, log in to the secondary node as `root`, or as a user with superuser privileges.
   b. Configure cluster services to start when the node starts.

      ```
      systemctl enable corosync; systemctl enable pacemaker
      ```

Property and resource options

Pacemaker provides options to support cluster configurations from small and simple to and large and complex. The following list identifies the options that support the two-node, active/passive configuration for Control Center.

`resource-stickiness=100`

Keep all resources bound to the same host.

`no-quorum-policy=ignore`

Pacemaker supports the notion of a voting quorum for clusters of three or more nodes. However, with just two nodes, if one fails, it does not make sense to have a quorum of one, therefore we disable quorums.

`stonith-enabled=false`

Fence or isolate a failed node. (The string "stonith" is an acronym for "shoot the other node in the head".) **This option should only be set to `true` when fencing is implemented.** For more information about fencing, see *Planning a high-availability deployment* on page 8.

If fencing is implemented, set this option to `false` during the initial setup and testing period. For production use, set it to `true`.

Setting resource and property defaults

Perform this procedure to set resource and property defaults for the high-availability cluster.

1. Log in to the primary node as `root`, or as a user with superuser privileges.
2. Set resource and property defaults.

   ```
   pcs resource defaults resource-stickiness=100
   pcs property set no-quorum-policy=ignore
   pcs property set stonith-enabled=false
   ```
3. Check resource defaults.

   ```
   pcs resource defaults
   ```
   Example result:

   ```
   resource-stickiness: 100
   ```
4. Check property defaults.

   ```
   pcs property
   ```

Example result:

```
Cluster Properties:
cluster-infrastructure: corosync
cluster-name: serviced-ha
dc-version: 1.1.12-a14efad
have-watchdog: false
no-quorum-policy: ignore
stonith-enabled: false
```

Defining resources

This procedure defines the following logical resources required for the cluster:

- DRBD Master/Secondary DFS set
- Two mirrored filesystems running on top of DRBD:
  - `/opt/serviced/var/isvcs`
  - `/opt/serviced/var/volumes`
- `serviced` logical volume group running on top of DRBD
- Manage `serviced` storage
- The floating virtual IP address of the cluster (*HA-Virtual-IP*), which the management software assigns to the active node
- Docker
- NFS
- Control Center

1. Log in to the primary node as `root`, or as a user with superuser privileges.
2. In a separate window, log in to the secondary node as `root`, or as a user with superuser privileges.
3. Define a resource for the DRBD device, and a clone of that resource to act as the master.
   a. On the primary node, define a resource for the DRBD device.

   ```
   pcs resource create DFS ocf:linbit:drbd \
     drbd_resource=serviced-dfs \
     op monitor interval=30s role=Master \
     op monitor interval=60s role=Slave
   ```
   b. On the primary node, define a clone of that resource to act as the master.

   ```
   pcs resource master DFSMaster DFS \
     master-max=1 master-node-max=1 \
     clone-max=2 clone-node-max=1 notify=true
   ```

   For a master/slave resource, Pacemaker requires separate monitoring intervals for the different roles. In this case, Pacemaker checks the master every 30 seconds and the slave every 60 seconds.
4. Define the filesystems that are mounted on the DRBD devices.
   a. On the primary node, define a resource for Control Center internal services data.

   ```
   pcs resource create serviced-isvcs Filesystem \
     device=/dev/drbd/by-res/serviced-dfs/0 \
     directory=/opt/serviced/var/isvcs fstype=xfs
   ```

b On the primary node, define a resource for Control Center metadata.

```
pcs resource create serviced-volumes Filesystem \
  device=/dev/drbd/by-res/serviced-dfs/1 \
  directory=/opt/serviced/var/volumes fstype=xfs
```

In the preceding definitions, `serviced-dfs` is the name of the DRBD resource defined previously, in `/etc/drbd.d/serviced-dfs.res`.

5 On the primary node, define the logical volume for `serviced` that is backed by a DRBD device.

```
pcs resource create serviced-lvm ocf:heartbeat:LVM volgrpname=serviced
```

6 On the primary node, define the storage resource for `serviced`, to ensure that the device mapper device is deactivated and unmounted properly.

```
pcs resource create serviced-storage ocf:zenoss:serviced-storage
```

7 On the primary node, define the resource that represents the floating virtual IP address of the cluster.

For dual-NIC deployments, the definition includes the `nic` key-value pair, which specifies the name of the network interface that is used for all traffic except the private DRBD traffic between the primary and seconday nodes. For single-NIC deployments, omit `nic` key-value pair.

For dual-NIC deployments, replace *HA-Virtual-IP* with the floating virtual IP address of the cluster, and replace *HA-Virtual-IP-NIC* with the name of the network interface that is bound to *HA-Virtual-IP*:

```
pcs resource create VirtualIP ocf:heartbeat:IPaddr2 \
  ip=HA-Virtual-IP nic=HA-Virtual-IP-NIC \
  cidr_netmask=32 op monitor interval=30s
```

For single-NIC deployments, replace *HA-Virtual-IP* with the floating virtual IP address of the cluster:

```
pcs resource create VirtualIP ocf:heartbeat:IPaddr2 \
  ip=HA-Virtual-IP cidr_netmask=32 op monitor interval=30s
```

8 Define the Docker resource.

a On the primary node, define the resource.

```
pcs resource create docker systemd:docker
```

b On both nodes, ensure that the automatic startup of Docker by `systemd` is disabled.

```
systemctl stop docker && systemctl disable docker
```

9 Define the NFS resource.

Control Center uses NFS to share configuration in a multi-host deployment, and failover will not work properly if NFS is not stopped on the failed node.

a On the primary node, define the resource.

```
pcs resource create nfs systemd:nfs
```

b On the primary node, disable Pacemaker monitoring of NFS health.

During normal operations, Control Center occasionally stops and restarts NFS, which could be misinterpreted by Pacemaker and trigger an unwanted failover.

```
pcs resource op remove nfs monitor interval=60s timeout=100
pcs resource op add nfs monitor interval=0s timeout=100
```

c On both nodes, ensure that the automatic startup of NFS by `systemd` is disabled.

```
systemctl stop nfs && systemctl disable nfs
```

10 Define the Control Center resource.

a On the primary node, define the resource.

```
pcs resource create serviced ocf:zenoss:serviced
```

b On both nodes, ensure that the automatic startup of `serviced` by `systemd` is disabled.

```
systemctl stop serviced && systemctl disable serviced
```

Pacemaker uses the default timeouts defined by the Pacemaker resource agent for Control Center to decide if `serviced` is unable to start or shutdown correctly. Starting with version 0.0.5 of the Pacemaker resource agent for Control Center, the default values for the start and stop timeouts are 360 and 130 seconds respectively.

The default startup and shutdown timeouts are based on the worst case scenario. In practice, Control Center typically starts and stops in much less time. However, this does not mean that you should decrease these timeouts. There are potential edge cases, especially for startup, where Control Center may take longer than usual to start or stop. If the start/stop timeouts for Pacemaker are set too low, and Control Center encounters one of those edge cases, then Pacemaker takes unnecessary or incorrect actions. For example, if the startup timeout is artificially set too low, 2.5 minutes for example, and Control Center startup encounters an unusual case where it requires at least 3 minutes to start, then Pacemaker initiates failover prematurely.

Defining the Control Center resource group

The resources in a resource group are started in the order they appear in the group, and stopped in the reverse order they appear in the group. The start order is:

1. Mount the filesystems (`serviced-isvcs` and `serviced-volumes`)
2. Start the `serviced` logical volume.
3. Manage `serviced` storage.
4. Enable the virtual IP address of the cluster.
5. Start Docker.
6. Start NFS.
7. Start Control Center.

In the event of a failover, Pacemaker stops the resources on the failed node in the reverse order they are defined before starting the resource group on the standby node.

1. Log in to the primary node as `root`, or as a user with superuser privileges.
2. Create the Control Center resource group.

```
pcs resource group add serviced-group \
  serviced-isvcs serviced-volumes \
  serviced-lvm serviced-storage \
  VirtualIP docker nfs \
  serviced
```

3 Define constraints for the Control Center resource group.

Pacemaker resource constraints control when and where resources are deployed in a cluster.

a Ensure that `serviced-group` runs on the same node as `DFSMaster`.

```
pcs constraint colocation add serviced-group with DFSMaster \
  INFINITY with-rsc-role=Master
```

b Ensure that `serviced-group` is only started after `DFSMaster` is started.

```
pcs constraint order promote DFSMaster then \
  start serviced-group
```

Verification procedures 7

The cluster is created in standby mode while various configurations are created. Perform the procedures in the following sections to review the configurations and make adjustments as necessary.

Verifying the DRBD configuration

This procedure reviews the DRBD configuration.

1. Log in to the primary node as `root`, or as a user with superuser privileges.
2. In a separate window, log in to the secondary node as `root`, or as a user with superuser privileges.
3. On the primary node, display the full DRBD configuration.

   ```
   drbdadm dump
   ```
   The result should be consistent with the configuration created previously. For more information, see *Configuring DRBD* on page 27.
4. On both nodes, start DRBD.

   ```
   drbdadm up all
   ```
5. On the primary node, start the synchronization.

   ```
   drbdadm primary --force serviced-dfs
   ```
6. On the primary node, display the synchronization status of mirrored storage areas.

   ```
   drbd-overview
   ```
   Do not proceed until the synchronization is complete. The process is complete when the status of the devices is `UpToDate/UpToDate`.
7. On both nodes, stop DRBD.

   ```
   drbdadm down all
   ```

Verifying the Pacemaker configuration

This procedure reviews the resource and property defaults for Pacemaker.

1 Log in to the primary node as `root`, or as a user with superuser privileges.

2 Check resource defaults.

```
pcs resource defaults
```

Example result:

```
resource-stickiness: 100
```

3 Check property defaults.

```
pcs property
```

Example result:

```
Cluster Properties:
 cluster-infrastructure: corosync
 cluster-name: serviced-ha
 dc-version: 1.1.12-a14efad
 have-watchdog: false
 no-quorum-policy: ignore
 stonith-enabled: false
```

Note Set the `stonith-enabled` option to `true` only if fencing is implemented. For more information about fencing, see *Planning a high-availability deployment* on page 8.

4 Review the resource constraints.

The ordering constraint should show that `serviced-group` starts after `DFSMaster` (the DRBD master). The colocation constraint should show that `serviced-group` resource and `DFSMaster` are on the same active cluster node.

```
pcs constraint
```

Example result:

```
Location Constraints:
Ordering Constraints:
  promote DFSMaster then start serviced-group (kind:Mandatory)
Colocation Constraints:
  serviced-group with DFSMaster (score:INFINITY) (with-rsc-
role:Master)
```

5 Review the ordering of the `serviced-group` resource group.

```
pcs resource show --full
```

The resources in a resource group are started in the order they appear in the group, and stopped in the reverse order they appear in the group. The correct start order is:

1. serviced-isvcs
2. serviced-volumes
3. serviced-lvm
4. serviced-storage
5. VirtualIP
6. Docker

7 nfs
8 serviced

Verifying the Control Center configuration

This procedure verifies that the Control Center configuration is identical on both nodes.

1 Log in to the primary node as `root`, or as a user with superuser privileges.
2 In a separate window, log in to the secondary node as `root`, or as a user with superuser privileges.
3 On both nodes, compute the checksum of the Control Center configuration file.

```
cksum /etc/default/serviced
```

- If the result is identical on both nodes, the configurations are identical. Do not perform the next step.
- If the result is not identical on both nodes, there may be a difference in their configurations; proceed to the next step.

4 Optional: On both nodes, display the customized variables, if necessary.

```
grep -E '^\b*[A-Z_]+' /etc/default/serviced | sort
```

Example result:

```
SERVICED_DM_THINPOOLDEV=/dev/mapper/serviced-serviced--pool
SERVICED_DOCKER_REGISTRY=HA-Virtual-IP:5000
SERVICED_ENDPOINT=HA-Virtual-IP:4979
SERVICED_FS_TYPE=devicemapper
SERVICED_LOG_ADDRESS=HA-Virtual-IP:5042
SERVICED_MASTER=1
SERVICED_OUTBOUND_IP=HA-Virtual-IP
SERVICED_STATS_PORT=HA-Virtual-IP:8443
SERVICED_ZK=HA-Virtual-IP:2181
```

Note There may only be insignificant differences between the files, such as an extra space at the beginning of a variable definition.

Verifying cluster startup

This procedure verifies the initial configuration by attempting to start the resources on one node only. With the other node in standby mode, Pacemaker does not automatically fail over to the other node.

1 Log in to the primary node as `root`, or as a user with superuser privileges.
2 In a separate window, log in to the secondary node as `root`, or as a user with superuser privileges.
3 On the primary node, determine which node is the primary DRBD node.

```
pcs status
```

Example result:

```
Cluster name: serviced-ha
Last updated: Mon Feb 22 11:37:58 2016  Last change: Mon Feb 22
 11:35:19 2016 by root via crm_attribute on Secondary-Public-Name
Stack: corosync
Current DC: Primary-Public-Name (version 1.1.13-a14efad) - partition
 with quorum
```

```
2 nodes and 10 resources configured

Node Primary-Public-Name: standby
Node Secondary-Public-Name: standby

Full list of resources:

 Master/Slave Set: DFSMaster [DFS]
 Stopped: [ Primary-Public-Name Secondary-Public-Name ]
 Resource Group: serviced-group
     serviced-isvcs  (ocf::heartbeat:Filesystem):  Stopped
     serviced-volumes  (ocf::heartbeat:Filesystem):  Stopped
     serviced-lvm  (ocf::heartbeat:LVM):  Stopped
     serviced-storage  (ocf::zenoss:serviced-storage):  Stopped
     VirtualIP  (ocf::heartbeat:IPaddr2):  Stopped
     docker  (systemd:docker):  Stopped
     nfs  (systemd:nfs):  Stopped
     serviced  (ocf::zenoss:serviced):  Stopped

PCSD Status:
  Primary-Public-Name: Online
  Secondary-Public-Name: Online

Daemon Status:
  corosync: active/disabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

The line that begins with `Current DC` identifies the primary node. Review all of the command output for errors.

4 Start DRBD.

a On the secondary node, enter the following command:

```
drbdadm up all
```

b On the primary node, enter the following commands:

```
drbdadm up all && drbdadm primary serviced-dfs
```

5 Start cluster resources.

You can run `pcs` commands on either node.

```
pcs cluster unstandby Primary-Public-Name
```

6 Monitor the status of cluster resources.

```
watch pcs status
```

Monitor the status until all resources report `Started`. Resolve any issues before continuing.

Verifying cluster failover

This procedure simulates a failover.

1 Log in to the primary node as `root`, or as a user with superuser privileges.

2 Enable the DRBD secondary node.

a Take the secondary node out of standby mode.

Replace *Secondary-Public-Name* with the public hostname of the secondary node:

```
pcs cluster unstandby Secondary-Public-Name
```

b Monitor the status of the secondary node.

```
pcs status
```

Do not continue until the status of the secondary node is `Online`.

3 Verify that DRBD has completely synchonized all three volumes on the secondary node.

```
drbd-overview
```

Example result:

```
0:serviced-dfs/0  Connected Primary/Secondary UpToDate/UpToDate
1:serviced-dfs/1  Connected Primary/Secondary UpToDate/UpToDate
2:serviced-dfs/2  Connected Primary/Secondary UpToDate/UpToDate
```

4 Force a failover.

Pacemaker initiates a failover when the primary node is put in standby mode.

Replace *Primary-Public-Name* with the public hostname of the primary node:

```
pcs cluster standby Primary-Public-Name
```

5 Monitor the cluster status.

```
pcs status
```

Repeat the preceding command until all resources report a status of `Started`. Resolve any issues before continuing.

6 Restore the cluster.

Replace *Primary-Public-Name* with the public hostname of the primary node:

```
pcs cluster unstandby Primary-Public-Name
```

Configuring authentication on master host nodes 8

Creating new resource pools

This procedure creates a new resource pool for the Control Center master nodes, and one or more resource pools for other hosts.

1. Log in to the primary node of the high-availability cluster as `root`, or as a user with superuser privileges.

   To ensure you are logging in to the primary node, use the virtual hostname (*HA-Virtual-Name*) or virtual IP address (*HA-Virtual-IP*) of the high-availability cluster.
2. Create a new resource pool named `master`.
   a. Create the pool.

      ```
      serviced pool add master
      ```
   b. Set permissions on the pool.

      ```
      serviced pool set-permission --dfs=true --admin=true master
      ```
3. Optional: Create additional resource pools, if desired.

   At least one additional resource pools is recommended. Many users find it useful to have pool names such as `infrastructure` and `collector-`*n* for groups of delegate hosts.
   a. Display the names of the existing resource pools.

      ```
      serviced pool list
      ```
   b. Create a new resource pool.
      Replace *Resource-Pool* with the name of a new resource pool:

      ```
      serviced pool add Resource-Pool
      ```

      Repeat the preceding command as desired.
   c. Optional: Set permissions on new resource pools, if desired.
      Replace *Resource-Pool* with the name of the new resource pool:

      ```
      serviced pool set-permission --dfs=true --admin=true Resource-Pool
      ```

Repeat the preceding command as desired.

Adding master nodes to their resource pool

This procedure adds the Control Center master nodes to their resource pool, named `master`. The master nodes are added to the resource pool with their public hostnames, so that you can easily see which node is active when you log in to the Control Center browser interface.

1. Use the virtual hostname (*HA-Virtual-Name*) or virtual IP address (*HA-Virtual-IP*) of the high-availability cluster to start a Bash shell on the Control Center master host as `root`, or as a user with superuser privileges.
2. Display the public hostname of the current node.

   ```
   uname -n
   ```
   The result is either *Primary-Public-Name* or *Secondary-Public-Name*.
3. Add the current node to the `master` resource pool.

   Replace *Node-Hostname* with the public hostname of the current node:

   ```
   serviced host add --register Node-Hostname:4979 master
   ```
4. Force a failover.

   Replace *Node-Hostname* with the public hostname of the current node:

   ```
   pcs cluster standby Node-Hostname
   ```
5. Monitor the cluster status.

   ```
   watch pcs status
   ```
   Do not proceed until all resources report a status of `Started`.
6. Use the virtual hostname (*HA-Virtual-Name*) or virtual IP address (*HA-Virtual-IP*) of the high-availability cluster to start a Bash shell on the Control Center master host as `root`, or as a user with superuser privileges.
7. Display the public hostname of the current node.

   ```
   uname -n
   ```
8. Add the current node to the `master` resource pool.

   Replace *Node-Hostname* with the public hostname of the current node:

   ```
   serviced host add --register Node-Hostname:4979 master
   ```
9. Restore the cluster.

   Replace *Standby-Node-Hostname* with the public hostname of the node that is in standby mode:

   ```
   pcs cluster unstandby Standby-Node-Hostname
   ```

Installing delegate hosts 9

This chapter describes how to install Control Center on a Red Hat Enterprise Linux (RHEL) or CentOS host. The candidate host must have the CPU, RAM, and storage resources required to serve as a Control Center delegate host. Delegate hosts are not members of the high-availability cluster.

Control Center delegate hosts run the application services scheduled for the resource pool to which they belong, and for which they have sufficient RAM and CPU resources. In a high-availability deployment, a delegate host may belong to any resource pool other than `master`.

Repeat the procedures in this chapter on each host to add to the Control Center deployment.

Verifying candidate host resources

Use this procedure to determine whether a host's hardware resources and operating system are sufficient to serve as a Control Center delegate host. Perform this procedure on each candidate delegate host in your deployment.

1. Log in to the candidate host as `root`, or as a user with superuser privileges.
2. Verify that the host implements the 64-bit version of the x86 instruction set.

   ```
   uname -m
   ```

   - If the output is `x86_64`, the architecture is 64-bit. Proceed to the next step
   - If the output is `i386/i486/i586/i686`, the architecture is 32-bit. Stop this procedure and select a different host.
3. Verify that the host has adequate storage space on the root filesystem.

   ```
   df -h /
   ```

   If the result does not include a minimum of 35GB of available space, stop this procedure and select a different host.
4. Determine whether the available memory and swap is sufficient.
   a. Display the available memory.

      ```
      free -h
      ```

   b. Compare the available memory with the amount required for a delegate host in your deployment.

      For more information, refer to the *Resource Manager Planning Guide* or the *Zenoss Community Edition (Core) Planning Guide*.

If the result does not meet minimum requirements, stop this procedure and select a different host.

5 Determine whether the installed operating system release is one of the releases that has been tested with Control Center.

```
cat /etc/redhat-release
```

- If the result includes `7.2`, `7.3`, or `7.4` proceed to the next step.
- If the result does not include `7.2`, `7.3`, or `7.4`, select a different host, and then start this procedure again.

6 Determine whether the CPU resources are sufficient.

a Display the total number of CPU cores.

```
grep -Ec '^core id' /proc/cpuinfo
```

b Compare the available resources with the requirements for a Control Center master host.

For more information, refer to the *Resource Manager Planning Guide* or the *Zenoss Community Edition (Core) Planning Guide*.

7 Determine whether the CPU resources support the AES instruction set.

```
grep -Ec '^flags.*aes' /proc/cpuinfo
```

For optimal performance, the result of the preceding commands must match the total number of CPU resources available on the host. If the result is 0, performance is severely degraded.

If the result is 0 and the candidate host is a virtual machine, the managing hypervisor may be configured in Hyper-V compatibility mode. Check the setting and disable it, if possible, or select a different host.

8 Ensure the host has a persistent numeric ID.

Each Control Center host must have a unique host ID, and the ID must be persistent (not change when the host reboots).

```
test -f /etc/hostid || genhostid ; hostid
```

Record the ID for comparison with other Control Center hosts.

9 Verify that name resolution works on this host.

```
hostname -i
```

If the result is not a valid IPv4 address, add an entry for the host to the network nameserver, or to `/etc/hosts`.

10 Add an entry to `/etc/hosts` for localhost, if necessary.

a Determine whether `127.0.0.1` is mapped to `localhost`.

```
grep 127.0.0.1 /etc/hosts | grep localhost
```

If the preceding commands return no result, perform the following substep.

b Add an entry to `/etc/hosts` for `localhost`.

```
echo "127.0.0.1 localhost" >> /etc/hosts
```

11 Update the Linux kernel, if necessary.

a Determine which kernel version is installed.

```
uname -r
```

If the result is lower than `3.10.0-327.22.2.el7.x86_64`, perform the following substep.

b Update the kernel, and then restart the host.

The following commands require internet access or a local mirror of operating system packages.

```
yum makecache fast && yum update kernel && reboot
```

Delegate host storage requirements

This section provides a quick reference to the block storage requirements of Control Center delegate hosts. For more information, refer to the *Resource Manager Planning Guide* or the *Zenoss Community Edition (Core) Planning Guide*.

Control Center hosts need either unformatted block storage devices or partitions, or free space in one or more LVM volume groups.

- Enter the following command to display information about block storage:

```
lsblk -ap --output=NAME,SIZE,TYPE,FSTYPE,MOUNTPOINT
```

- Enter the following command to display information about LVM volume groups:

```
vgdisplay
```

The following storage configuration is recommended for delegate hosts:

- A root filesystem with a minimum of 35GB of storage, formatted with XFS.
- A dedicated swap area.
- One or more block devices or partitions, or one or more LVM physical volumes, with a total of 50GB of space. A step in this chapter configures the space as an LVM thin pool for Docker data.

Configuring NTP clients

Use this procedure to configure a delegate hosts to synchronize its clocks with the NTP server on the Control Center master host. Perform this procedure only if the hosts do not have internet access. Repeat this procedure on each Control Center delegate host.

Note On VMware vSphere guests, disable time synchronization between guest and host operating systems before performing this procedure.

1 Log in to the Control Center delegate host as `root`, or as a user with superuser privileges.
2 Create a backup of the NTP configuration file.

```
cp -p /etc/ntp.conf /etc/ntp.conf.orig
```

3 Edit the NTP configuration file./

a Open `/etc/ntp.conf` with a text editor.

b Replace all of the lines in the file with the following lines:

```
# Point to the master time server
server MASTER_ADDRESS

restrict default ignore
restrict 127.0.0.1
restrict MASTER_ADDRESS mask 255.255.255.255 nomodify notrap noquery

driftfile /var/lib/ntp/drift
```

c Replace both instances of `MASTER_ADDRESS` with the virtual IP address (*HA-Virtual-IP*) of the high-availability cluster.

d Save the file and exit the editor.

4 Synchronize the clock with the master server.

```
ntpd -gq
```

5 Enable and start the NTP daemon.

a Enable the `ntpd` daemon.

```
systemctl enable ntpd
```

b Configure `ntpd` to start when the system starts.

Currently, an unresolved issue associated with NTP prevents `ntpd` from restarting correctly after a reboot, and the following commands provide a workaround to ensure that it does.

```
echo "systemctl start ntpd" >> /etc/rc.d/rc.local
chmod +x /etc/rc.d/rc.local
```

c Start `ntpd`.

```
systemctl start ntpd
```

Preparing a delegate host

Use this procedure to prepare a RHEL/CentOS host as a Control Center delegate host.

1 Log in to the candidate delegate host as `root`, or as a user with superuser privileges.

2 Disable the firewall, if necessary.

This step is required for installation but not for deployment. For more information, refer to the *Resource Manager Planning Guide* or the *Zenoss Community Edition (Core) Planning Guide*.

a Determine whether the `firewalld` service is enabled.

```
systemctl status firewalld.service
```

- If the result includes `Active: inactive (dead)`, the service is disabled. Proceed to the next step.
- If the result includes `Active: active (running)`, the service is enabled. Perform the following substep.

b Disable the `firewalld` service.

```
systemctl stop firewalld && systemctl disable firewalld
```

On success, the preceding commands display messages similar to the following example:

```
rm '/etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service'
rm '/etc/systemd/system/basic.target.wants/firewalld.service'
```

3 Optional: Enable persistent storage for log files, if desired.

By default, RHEL/CentOS systems store log data only in memory or in a ring buffer in the `/run/log/journal` directory. By performing this step, log data persists and can be saved indefinitely, if you implement log file rotation practices. For more information, refer to your operating system documentation.

Note The following commands are safe when performed during an installation, before Docker or Control Center are installed or running. To enable persistent log files after installation, stop Control Center, stop Docker, and then enter the following commands.

```
mkdir -p /var/log/journal && systemctl restart systemd-journald
```

4 Enable and start the Dnsmasq package.

The package facilitates networking among Docker containers.

```
systemctl enable dnsmasq && systemctl start dnsmasq
```

If name resolution in your environment relies solely on entries in `/etc/hosts`, configure `dsnmasq` so that containers can use the file:

a Open `/etc/dnsmasq.conf` with a text editor.
b Locate the line that starts with `#domain-needed`, and then make a copy of the line, immediately below the original.
c Remove the number sign character (`#`) from the beginning of the line.
d Locate the line that starts with `#bogus-priv`, and then make a copy of the line, immediately below the original.
e Remove the number sign character (`#`) from the beginning of the line.
f Locate the line that starts with `#local=/localnet/`, and then make a copy of the line, immediately below the original.
g Remove `net`, and then remove the number sign character (`#`) from the beginning of the line.
h Locate the line that starts with `#domain=example.com`, and then make a copy of the line, immediately below the original.
i Replace `example.com` with `local`, and then remove the number sign character (`#`) from the beginning of the line.
j Save the file, and then close the editor.
k Restart the `dnsmasq` service.

```
systemctl restart dnsmasq
```

5 Reboot the host.

```
reboot
```

Installing Docker

Use this procedure to install Docker.

1. Log in to the host as `root`, or as a user with superuser privileges.
2. Install Docker CE 17.09.0 from the local repository mirror.
   - **a** Install Docker CE.

     ```
     yum install --enablerepo=zenoss-mirror docker-ce-17.09.0.ce
     ```

     If `yum` returns an error due to dependency issues, see *Resolving package dependency conflicts* on page 93 for potential resolutions.
   - **b** Enable automatic startup.

     ```
     systemctl enable docker
     ```

Installing Control Center

Use this procedure to install Control Center.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Install Control Center 1.5.1 from the local repository mirror.
   - **a** Install Control Center.

     ```
     yum install --enablerepo=zenoss-mirror \
       /opt/zenoss-repo-mirror/serviced-1.5.1-1.x86_64.rpm
     ```

     If `yum` returns an error due to dependency issues, see *Resolving package dependency conflicts* on page 93 for potential resolutions.
   - **b** Enable automatic startup.

     ```
     systemctl enable serviced
     ```
3. Make a backup copy of the Control Center configuration file.
   - **a** Make a copy of `/etc/default/serviced`.

     ```
     cp /etc/default/serviced /etc/default/serviced-1.5.1-orig
     ```
   - **b** Set the backup file permissions to read-only.

     ```
     chmod 0440 /etc/default/serviced-1.5.1-orig
     ```

Configuring Docker

Use this procedure to configure Docker.

1. Log in to the delegate host as `root`, or as a user with superuser privileges.
2. Create a symbolic link for the Docker temporary directory.

   Docker uses its temporary directory to spool images. The default directory is `/var/lib/docker/tmp`. The following command specifies the same directory that Control Center uses, `/tmp`. You can specify any directory that has a minimum of 10GB of unused space.

a Create the `docker` directory in `/var/lib`.

```
mkdir /var/lib/docker
```

b Create the link to `/tmp`.

```
ln -s /tmp /var/lib/docker/tmp
```

3 Create a `systemd` drop-in file for Docker.

a Create the override directory.

```
mkdir -p /etc/systemd/system/docker.service.d
```

b Create the unit drop-in file.

```
cat <<EOF > /etc/systemd/system/docker.service.d/docker.conf
[Service]
TimeoutSec=300
EnvironmentFile=-/etc/sysconfig/docker
ExecStart=
ExecStart=/usr/bin/dockerd \$OPTIONS
TasksMax=infinity
EOF
```

c Reload the `systemd` manager configuration.

```
systemctl daemon-reload
```

4 Create an LVM thin pool for Docker data.

For more information about the `serviced-storage` command, see *serviced-storage* on page 89.

To use an entire block device or partition for the thin pool, replace *Device-Path* with the device path:

```
serviced-storage create-thin-pool docker Device-Path
```

On success, the result is the device mapper name of the thin pool, which always starts with `/dev/mapper`.

5 Configure and start the Docker service.

a Create a variable for the name of the Docker thin pool.

Replace *Thin-Pool-Device* with the name of the thin pool device created in the previous step:

```
myPool="Thin-Pool-Device"
```

b Create a variable for the master host.

Replace *Master* with the virtual hostname (*HA-Virtual-Name*) or IPv4 address (*HA-Virtual-IP*) of the high-availability cluster:

```
myCluster="Master"
```

The value of this variable must match the value of the *SERVICED_DOCKER_REGISTRY* variable in `/etc/default/serviced`.

c Create variables for adding arguments to the Docker configuration file. The `--exec-opt` argument is a workaround for *a Docker issue* on RHEL/CentOS 7.x systems.

```
myDriver="--storage-driver devicemapper"
myLog="--log-level=error"
myFix="--exec-opt native.cgroupdriver=cgroupfs"
myMount="--storage-opt dm.mountopt=discard"
myFlag="--storage-opt dm.thinpooldev=$myPool"
myReg="--insecure-registry=$myCluster:5000"
```

d Add the arguments to the Docker configuration file.

```
echo 'OPTIONS="'$myLog $myDriver $myFix $myMount $myFlag $myReg'"' \
  >> /etc/sysconfig/docker
```

e Start or restart Docker.

```
systemctl restart docker
```

The startup may take up to a minute, and may fail. If startup fails, repeat the `restart` command.

6 Configure name resolution in containers.

Each time it starts, `docker` selects an IPv4 subnet for its virtual Ethernet bridge. The selection can change; this step ensures consistency.

a Identify the IPv4 subnet and netmask `docker` has selected for its virtual Ethernet bridge.

```
ip addr show docker0 | grep inet
```

b Open `/etc/sysconfig/docker` in a text editor.

c Add the following flags to the end of the *OPTIONS* declaration.

Replace *Bridge-Subnet* with the IPv4 subnet `docker` selected for its virtual bridge:

```
--dns=Bridge-Subnet --bip=Bridge-Subnet/16
```

For example, if the bridge subnet is 172.17.0.1, add the following flags:

```
--dns=172.17.0.1 --bip=172.17.0.1/16
```

Note Use a space character ( ) to separate flags, and make sure the double quote character (`"`) delimits the declaration of *OPTIONS*.

d Save the file, and then close the editor.

e Restart the Docker service.

```
systemctl restart docker
```

Configuring NFS 4.0

Use this procedure to configure NFS 4.0 on delegate hosts if the operating system release is 7.4. There may be a file locking defect in NFS 4.1 with RHEL/CentOS 7.4.

1 Log in to the host as `root`, or as a user with superuser privileges.

2 Determine which release is installed.

```
cat /etc/redhat-release
```

- If the result includes `7.4`, perform the remaining steps of this procedure.
- If the result includes `7.2` or `7.3`, continue to the next procedure.

3 Change the NFS configuration file.

 a Open `/etc/nfsmount.conf` with a text editor.
 b Locate the `Defaultvers` directive.
 c Remove the number sign character (`#`) from the beginning of the line.
 d Change the value from `4` to `4.0`.
 The line should appear as follows:

```
Defaultvers=4.0
```

 e Save the file, and then close the editor.

4 Restart the NFS server.

```
systemctl restart nfs-server
```

Importing the ZooKeeper image for Docker

Use this procedure to import the Docker image for the ZooKeeper service from a ZooKeeper image file. Repeat this procedure on each node in the ZooKeeper ensemble.

1 Log in to the ZooKeeper ensemble node as `root`, or as a user with superuser privileges.
2 Change directory to `/root`.

```
cd /root
```

3 Extract the ZooKeeper image.

```
./install-zenoss-isvcs-zookeeper-v*.run
```

Image extraction begins when you press the **y** key.

4 Optional: Delete the archive file, if desired.

```
rm -i ./install-zenoss-isvcs-zookeeper-v*.run
```

Proceed to the next chapter and configure the host.

Configuring and starting delegate hosts 10

This chapter provides the procedures that are required to configure a delegate host, describes the configuration options that apply to delegate hosts, and includes steps for starting a delegate for the first time. Perform the procedures in this chapter on each Control Center delegate host.

Note If you are installing a single-host deployment, skip this chapter.

This chapter includes synopses of the configuration variables that affect delegate hosts. For more information about a variable, see *Control Center configuration variables* on page 96.

Control Center maintenance scripts on delegate hosts

The scripts in the following list are installed when Control Center is installed, and are started either daily or weekly by `anacron`.

Of these scripts, only the first is required on delegate hosts. The others should be removed.

`/etc/cron.hourly/serviced`

This script invokes `logrotate` hourly, to manage the `/var/log/serviced-audit.log` and `/var/log/application-audit.log` files.

This script should be removed from delegate hosts.

`/etc/cron.weekly/serviced-fstrim`

This script should be removed from delegate hosts.

`/etc/cron.d/cron_zenossdbpack`

This script is required on one host in the resource pool in which the Zenoss database services run. For more information, refer to the configuration guide for your Zenoss application.

Enabling use of the command-line interface

Use this procedure to enable a user to perform administrative tasks with the Control Center command-line interface.

1 Log in to the host as `root`, or as a user with superuser privileges.
2 Add a user to the `serviced` group.

Replace *User* with the name of a login account on the host.

```
usermod -aG serviced User
```

Repeat the preceding command for each user to add.

Setting the host role to delegate

Use this procedure to configure a host as a delegate host. The following configuration variable assigns the host role:

SERVICED_MASTER

Default: `1` (true)

Assigns the role of a `serviced` instance, either master or delegate. The master runs the application services scheduler and other internal services. Delegates run the application services assigned to the resource pool to which they belong.

Only one `serviced` instance can be the master; all other instances must be delegates. The default value assigns the master role. To assign the delegate role, set the value to `0` (false). This variable must be explicitly set on all Control Center hosts.

Perform these steps:

1. Log in to the host as `root`, or as a user with superuser privileges.
2. Edit the Control Center configuration file.
   a. Open `/etc/default/serviced` in a text editor.
   b. Locate the line for the *SERVICED_MASTER* variable, and then make a copy of the line, immediately below the original.
   c. Remove the number sign character (`#`) from the beginning of the line.
   d. Change the value from `1` to `0`.
   e. Save the file, and then close the editor.
3. Verify the settings in the `serviced` configuration file.

```
grep -E '^\b*[A-Z_]+' /etc/default/serviced
```

Configuring the local Docker registry

Use this procedure to configure the endpoint of the local Docker registry.

The following configuration variable identifies the local Docker registry endpoint:

SERVICED_DOCKER_REGISTRY

Default: `localhost:5000`

The endpoint of the local Docker registry, which `serviced` uses to store internal services and application images.

If the default value is changed, the host's Docker configuration file must include the `--insecure-registry` flag with the same value as this variable.

The safest replacement for `localhost` is the IPv4 address of the registry host. Otherwise, the fully-qualified domain name of the host must be specified.

Perform these steps:

1. Log in to the delegate host as `root`, or as a user with superuser privileges.
2. Edit the Control Center configuration file.
   a. Open `/etc/default/serviced` in a text editor.
   b. Locate the line for the *SERVICED_DOCKER_REGISTRY* variable, and then make a copy of the line, immediately below the original.
   c. Remove the number sign character (`#`) from the beginning of the line.
   d. Replace `localhost` with the virtual IP address of the high-availability cluster (*HA-Virtual-IP*).

      The new variable value must include the delimiter (`:`) followed by the port number.
   e. Save the file, and then close the editor.
3. Verify the settings in the `serviced` configuration file.

```
grep -E '^\b*[A-Z_]+' /etc/default/serviced
```

4. Add the insecure registry flag to the Docker configuration file.
   a. Open `/etc/sysconfig/docker` in a text editor.
   b. Add the local Docker registry endpoint to the end of the *OPTIONS* declaration.
      Replace *Registry-Endpoint* with the same value used for the *SERVICED_DOCKER_REGISTRY* variable:

```
--insecure-registry=Registry-Endpoint
```

Note Use a space character (` `) to separate flags, and make sure the double quote character (`"`) delimits the declaration of *OPTIONS*.

   c. Save the file, and then close the editor.
5. Restart the Docker service.

```
systemctl restart docker
```

Configuring endpoints

Use this procedure to customize the internal services endpoints of Control Center. The following configuration variables specify the endpoints:

SERVICED_ZK

Default: (none)

The list of endpoints in the `serviced` ZooKeeper ensemble, separated by the comma character (`,`). Each endpoint identifies an ensemble node. Each Control Center server and in-container proxy uses *SERVICED_ZK* to create a randomized, round-robin list, and cycles through the list when it attempts to establish a connection with the lead ZooKeeper host.

SERVICED_ENDPOINT

Default: `{{SERVICED_MASTER_IP}}:4979`

The endpoint of the `serviced` RPC server. Replace `{{SERVICED_MASTER_IP}}` with the IP address or hostname of the `serviced` master host. The port number of this endpoint must match the value of the *SERVICED_RPC_PORT* variable defined on the `serviced` master host.

SERVICED_LOG_ADDRESS

Default: `{{SERVICED_MASTER_IP}}:5042`

The endpoint of the logstash service. Replace `{{SERVICED_MASTER_IP}}` with the IP address or hostname of the `serviced` master host.

SERVICED_LOGSTASH_ES

Default: `{{SERVICED_MASTER_IP}}:9100`

The endpoint of the Elasticsearch service for logstash. On delegate hosts, replace `{{SERVICED_MASTER_IP}}` with the IP address or hostname of the Elasticsearch host, which by default is the `serviced` master host.

SERVICED_STATS_PORT

Default: `{{SERVICED_MASTER_IP}}:8443`

The endpoint of the `serviced` metrics consumer service. Replace `{{SERVICED_MASTER_IP}}` with the IP address or hostname of the `serviced` master host.

Perform these steps:

1. Log in to the host as `root`, or as a user with superuser privileges.
2. Edit the Control Center configuration file.
   a. Open `/etc/default/serviced` in a text editor.
   b. For each endpoint variable, locate the line that sets the variable, and then make a copy of the line, immediately below the original.
   c. Remove the number sign character (`#`) from the beginning of the line.
   d. Replace `{{SERVICED_MASTER_IP}}` with the virtual IP address of the high-availability cluster (*HA-Virtual-IP*).
   e. Save the file, and then close the editor.
3. Verify the settings in the `serviced` configuration file.

```
grep -E '^\b*[A-Z_]+' /etc/default/serviced
```

Delegate host configuration variables

The tables in this section provide an overview of the `serviced` configuration variables that apply to Control Center delegate hosts. Set these variables as required for your environment or applications.

Delegate variables

The following miscellaneous variables apply only to delegate hosts.

Variable	Description
SERVICED_ZK	The list of hosts in the ZooKeeper ensemble.
SERVICED_STATS_PERIOD	The frequency at which delegates gather metrics to send to the master host.
SERVICED_IPTABLES_MAX_CONNECTIONS	The maximum number of open connections to allow on a delegate. The number increases when the master is unavailable and decreases when the master returns to service.

RPC service variables

The variables in the following table must be set identically on all Control Center hosts, except:

- *SERVICED_RPC_PORT*, set only on the master

- *SERVICED_MAX_RPC_CLIENTS*, set only on delegates

By default, `serviced` uses TLS to encrypt all RPC traffic. The *SERVICED_KEY_FILE* and *SERVICED_CERT_FILE* variables identify the digital certificate used for RPC, mux, and HTTP traffic.

The *SERVICED_AUTH_TOKEN_EXPIRATION* variable affects RPC, mux, and internal services endpoint traffic.

Variable	Where to set
SERVICED_ENDPOINT	Master, delegates
SERVICED_MAX_RPC_CLIENTS	Delegates
SERVICED_RPC_PORT	Master
SERVICED_RPC_CERT_VERIFY	Master, delegates
SERVICED_RPC_DISABLE_TLS	Master, delegates
SERVICED_RPC_TLS_MIN_VERSION	Master, delegates
SERVICED_RPC_TLS_CIPHERS	Master, delegates
SERVICED_KEY_FILE	Master
SERVICED_CERT_FILE	Master
SERVICED_RPC_DIAL_TIMEOUT	Master, delegates
SERVICED_AUTH_TOKEN_EXPIRATION	Master

Multiplexer variables

The variables in the following table must be set identically on all Control Center hosts.

By default, `serviced` uses TLS to encrypt all mux traffic. The *SERVICED_KEY_FILE* and *SERVICED_CERT_FILE* variables identify the digital certificate used for RPC, mux, and HTTP traffic.

The *SERVICED_AUTH_TOKEN_EXPIRATION* variable affects RPC, mux, and internal services endpoint traffic.

Variable	Where to set
SERVICED_MUX_PORT	Master, delegates
SERVICED_MUX_DISABLE_TLS	Master, delegates
SERVICED_MUX_TLS_MIN_VERSION	Master, delegates
SERVICED_MUX_TLS_CIPHERS	Master, delegates
SERVICED_KEY_FILE	Master
SERVICED_CERT_FILE	Master
SERVICED_AUTH_TOKEN_EXPIRATION	Master

Universal configuration variables

The tables in this section provide an overview of the `serviced` configuration variables that apply to all Control Center hosts. Set these variables as required for your environment or applications.

Role variable

Variable	Description
SERVICED_MASTER	Assigns the role of a `serviced` instance, either master or delegate. The master runs the application services scheduler and other internal services. Delegates run the application services assigned to the resource pool to which they belong.

Browser interface variable (all hosts)

Variable	Description
SERVICED_UI_PORT	The port on which the HTTP server listens for requests.

Networking variables

Variable	Description
SERVICED_STATIC_IPS	A list of one or more static IP addresses for IP assignment.
SERVICED_OUTBOUND_IP	The IP address of the network interface for `serviced` to use. When this variable is not set, `serviced` uses the IP address of the default network interface and assumes it has internet access. To prevent `serviced` from assuming it has internet access, set this variable.
SERVICED_VIRTUAL_ADDRESS_SUBNET	The private network for containers that use virtual IP addresses. The default is `10.3.0.0/16`, and the network can be unique on each host. A `/29` network is sufficient.
SERVICED_DOCKER_DNS	A list of one or more DNS servers. The list is injected into all Docker containers.

Debugging variables

Variable	Description
SERVICED_LOG_LEVEL	The log level `serviced` uses when writing to the system log.
SERVICED_DEBUG_PORT	The port on which `serviced` listens for HTTP requests for the *Go profiler*.
SERVICED_DOCKER_LOG_DRIVER	The log driver for all Docker container logs.
SERVICED_DOCKER_LOG_CONFIG	Docker `--log-opt` options.

Tuning variables (all Control Center hosts)

Variable	Description
GOMAXPROCS	The maximum number of CPU cores that `serviced` uses.
SERVICED_MAX_CONTAINER_AGE	The number of seconds `serviced` waits before removing a stopped container.
SERVICED_ISVCS_ENV_[0-9]+	Startup arguments to pass to specific internal services.
SERVICED_SERVICE_MIGRATION_TAG	Overrides the default value for the service migration image.
SERVICED_OPTS	Startup arguments for `serviced`.
SERVICED_CONTROLLER_BINARY	The path of the `serviced-controller` binary.
SERVICED_HOME	The path of the home directory for `serviced`.
SERVICED_ETC_PATH	The path of the directory for `serviced` configuration files.
SERVICED_VHOST_ALIASES	A list of hostname aliases for a host; for example, `localhost`.
SERVICED_ZK_CONNECT_TIMEOUT	The number of seconds Control Center waits for a connection to the lead ZooKeeper host.
SERVICED_ZK_PER_HOST_CONNECT_DELAY	The number of seconds Control Center waits before attempting to connect to the next host in its round-robin list of ZooKeeper hosts.
SERVICED_ZK_RECONNECT_START_DELAY, *SERVICED_ZK_RECONNECT_MAX_DELAY*	These are used together when Control Center is unable to re-establish a connection with the lead ZooKeeper host.

Starting Control Center

Use this procedure to start `serviced` on a delegate host for the first time.

1. Log in to the delegate host as `root`, or as a user with superuser privileges.
2. Verify the settings in the `serviced` configuration file.

```
grep -E '^\b*[A-Z_]+' /etc/default/serviced
```

3. Start the Control Center service (`serviced`).

```
systemctl start serviced
```

To monitor progress, enter the following command:

```
journalctl -flu serviced -o cat
```

Delegate host authentication

Control Center uses RSA key pairs to create the authentication tokens that are required for all delegate communications. When you add a host to a resource pool, the `serviced` instance on the master host creates a private key for the delegate and bundles it with its own public key. The `serviced` instance on the delegate host uses the bundle to sign messages with its unique tokens.

Key bundles are installed by using an SSH connection or a file.

- The command to add a host to a pool can initiate an SSH connection with the delegate and install the key bundle. This option is the most secure, because no file is created. However, it requires either public key authentication or password authentication between the master and delegate hosts.
- When no SSH connection is requested, the command to add a host to a pool creates a file containing the key bundle. You can move the key bundle file to the delegate host with any file transfer method, and then install it on the delegate.

The following procedures demonstrate how to add a host to a resource pool and install its key bundle.

Adding a delegate host through an SSH connection

To succeed, the following statements about the login account used to perform this procedure must be true:

- The account exists on both the master host and on the delegate host.
- The account has `serviced` CLI privileges.
- The account has either public key authentication or password authentication enabled on the master host and on the delegate host.

Use this procedure to add a delegate host to a resource pool through an SSH connection.

1. Log in to the Control Center master host as a user with `serviced` CLI privileges.
2. Optional: Create a new resource pool:
   - **a** Display the names of the existing resource pools.

     ```
     serviced pool list
     ```
   - **b** Create a new resource pool.
     Replace *Resource-Pool* with the name of a new resource pool:

     ```
     serviced pool add Resource-Pool
     ```
   - **c** Optional: Set permissions on the new resource pool, if desired.
     Replace *Resource-Pool* with the name of the new resource pool:

     ```
     serviced pool set-permission --dfs=true --admin=true Resource-Pool
     ```
3. Add a delegate host to a resource pool.

   If the master and delegate host are configured for key-based access, the following command does not prompt you to add the delegate to the list of known hosts or to provide the password of the remote user account.

   Use the hostname or IP address to identify a Control Center host. If you use a hostname, all Control Center hosts must be able to resolve it, either through an entry in `/etc/hosts` or through a nameserver on the network. In the following example, replace *Hostname-Or-IP* with the hostname or IP address of a delegate host, and replace *Resource-Pool* with the name of a resource pool.

If the host is behind a router or firewall for network address translation (NAT), include the option `--nat-address` to specify the NAT device's hostname or IP address and port of the delegate host.

```
serviced host add --register Hostname-Or-IP:4979 Resource-Pool \
    --nat-address==NAT-Hostname-Or-IP:NAT-Port
```

Adding a delegate host using a file

Use this procedure to add a delegate host to a resource pool by using a key bundle file.

1. Log in to the Control Center master host as a user with `serviced` CLI privileges.
2. Optional: Create a new resource pool, if desired.
   a. Display the names of the existing resource pools.

      ```
      serviced pool list
      ```
   b. Create a new resource pool.
      Replace *Resource-Pool* with the name of a new resource pool:

      ```
      serviced pool add Resource-Pool
      ```
   c. Optional: Set permissions on the new resource pool, if desired.
      Replace *Resource-Pool* with the name of the new resource pool:

      ```
      serviced pool set-permission --dfs=true --admin=true Resource-Pool
      ```
3. Add a delegate host to a resource pool.

   Use the hostname or IP address to identify a Control Center host. If you use a hostname, all Control Center hosts must be able to resolve it, either through an entry in `/etc/hosts` or through a nameserver on the network. In the following example, replace *Hostname-Or-IP* with the hostname or IP address of a delegate host, and replace *Resource-Pool* with the name of a resource pool.

   If the host is behind a router or firewall for network address translation (NAT), include the option `--nat-address` to specify the NAT device's hostname or IP address and port of the delegate host.

   ```
   serviced host add Hostname-Or-IP:4979 Resource-Pool \
       --nat-address==NAT-Hostname-Or-IP:NAT-Port
   ```
   The command creates a unique key bundle file in the local directory.
4. Use a file transfer utility such as `scp` to copy the key bundle file to the delegate host.

   Once copied to the delegate host, the key bundle file is not needed on the master host and can be deleted.
5. Log in to the Control Center delegate host as a user with `serviced` CLI privileges.
6. Install the key bundle.
   Replace *Key-Bundle-Path* with the pathname of the key bundle file:

   ```
   serviced host register Key-Bundle-Path
   ```
7. Delete the key bundle file.

   The file is no longer needed on the delegate host.

   Replace *Key-Bundle-Path* with the pathname of the key bundle file:

   ```
   rm Key-Bundle-Path
   ```

Setting the connection timeout of a resource pool

Use this procedure to set the length of time the services scheduler waits for a disconnected delegate host to reconnect before moving its services to a different host in the resource pool. This affects all multi-host resource pools. Zenoss recommends using a minimum value of 15 seconds for all multi-host pools. For resource pools that are connected through high-latency, wide-area networks, Zenoss recommends values greater than 15 seconds.

1. Log in to the master host as a user with `serviced` CLI privileges.
2. Display the list of resource pools and their connection timeout values.

   ```
   serviced pool list -v | grep -E 'ID|ConnectionTimeout'
   ```
3. Optional: Set the connection timeout value of a resource pool.

   This command accepts the following units identifiers:

   `ms` (milliseconds)

   `s` (seconds)

   `m` (minutes)

   `h` (hours)

   Replace *Pool-ID* with a resource pool identifier, and replace *Timeout+Units* with an integer followed by a units identifier:

   ```
   serviced pool set-conn-timeout Pool-ID Timeout+Units
   ```

Configuring a ZooKeeper ensemble 11

This chapter describes how to create a ZooKeeper ensemble (cluster) for a multi-host Control Center deployment that includes a minimum of three hosts. If your deployment includes just one host or two hosts, skip this chapter.

ZooKeeper and Control Center

Control Center relies on *Apache ZooKeeper* to distribute and manage application services. ZooKeeper maintains the definitions of each service and the list of services assigned to each host. The scheduler, which runs on the master host, determines assignments and sends them to the ZooKeeper node that is serving as the ensemble leader. The leader replicates the assignments to the other ensemble nodes, so that the other nodes can assume the role of leader if the leader node fails.

All Control Center hosts retrieve assignments and service definitions from the ZooKeeper ensemble leader and then start services in Docker containers as required. So, the Control Center configuration files of all Control Center hosts must include a definition for the *SERVICED_ZK* variable, which specifies the ZooKeeper endpoints of the ensemble nodes. Additional variables are required on ensemble nodes.

A ZooKeeper ensemble requires a minimum of three nodes, which is sufficient for most environments. An odd number of nodes is recommended and an even number of nodes is strongly discouraged. A five-node ensemble improves failover protection during maintenance windows but larger ensembles yield no benefits.

The Control Center master host is always an ensemble node. All ensemble nodes should be on the same subnet.

Understanding the configuration process

The procedures in this chapter instruct you to create temporary variables that are used as building blocks, to construct Control Center configuration variables accurately. You append the Control Center variables to `/etc/default/serviced`, and then edit the file to move the variables to more appropriate locations.

The most important temporary variables specify the IP address or hostname of each host in the ZooKeeper ensemble. The following table identifies these important variables, the names and values of which must be identical on every Control Center host.

Variable name	Placeholder value	Actual value
`node1`	*HA-Virtual-IP* or *HA-Virtual-Name*	Virtual IP address or hostname of the high-availability cluster
`node2`	*Delegate-A*	IP address or hostname of delegate host A

Variable name	Placeholder value	Actual value
`node3`	*Delegate-B*	IP address or hostname of delegate host B

Note All ensemble hosts should be on the same subnet.

ZooKeeper variables

The variables in the following table are set only on ZooKeeper ensemble nodes, except *SERVICED_ZK*, which must be identical on all Control Center hosts.

Variable	Where to set
SERVICED_ISVCS_START	ZooKeeper ensemble nodes
SERVICED_ISVCS_ZOOKEEPER_ID	ZooKeeper ensemble nodes
SERVICED_ISVCS_ZOOKEEPER_QUORUM	ZooKeeper ensemble nodes
SERVICED_ZK	All Control Center hosts
SERVICED_ZK_SESSION_TIMEOUT	ZooKeeper ensemble nodes

Example multi-host ZooKeeper configuration

This example shows the ZooKeeper variables in the `/etc/default/serviced` configuration file of each host in a 4-node Control Center deployment. For convenience, the relevant settings for each node or host are also included in subsequent procedures.

Note The value of the *SERVICED_ISVCS_ZOOKEEPER_QUORUM* variable is formatted to fit the available space. In the configuration file, the variable and value are on the same line.

Master host and ZooKeeper ensemble node, 198.51.100.135:

```
SERVICED_ISVCS_ZOOKEEPER_ID=1
SERVICED_ZK=198.51.100.135:2181,198.51.100.136:2181,198.51.100.137:2181
SERVICED_ISVCS_ZOOKEEPER_QUORUM=1@0.0.0.0:2888:3888,\
  2@198.51.100.136:2888:3888,3@198.51.100.137:2888:3888
SERVICED_ZK_SESSION_TIMEOUT=15
```

Delegate host and ZooKeeper ensemble node, 198.51.100.136:

```
SERVICED_ISVCS_START=zookeeper
SERVICED_ISVCS_ZOOKEEPER_ID=2
SERVICED_ZK=198.51.100.135:2181,198.51.100.136:2181,198.51.100.137:2181
SERVICED_ISVCS_ZOOKEEPER_QUORUM=1@198.51.100.135:2888:3888,\
  2@0.0.0.0:2888:3888,3@198.51.100.137:2888:3888
SERVICED_ZK_SESSION_TIMEOUT=15
```

Delegate host and ZooKeeper ensemble node, 198.51.100.137:

```
SERVICED_ISVCS_START=zookeeper
SERVICED_ISVCS_ZOOKEEPER_ID=3
SERVICED_ZK=198.51.100.135:2181,198.51.100.136:2181,198.51.100.137:2181
SERVICED_ISVCS_ZOOKEEPER_QUORUM=1@198.51.100.135:2888:3888,\
  2@198.51.100.136:2888:3888,3@0.0.0.0:2888:3888
SERVICED_ZK_SESSION_TIMEOUT=15
```

Delegate host, 198.51.100.138:

```
SERVICED_ZK=198.51.100.135:2181,198.51.100.136:2181,198.51.100.137:2181
```

Configuring master host nodes as ZooKeeper node

Use this procedure to configure each master host node in a high-availability deployment as a node in a ZooKeeper ensemble. Perform this procedure on the primary node and on the secondary node.

1. Log in to a master host node as `root`, or as a user with superuser privileges.
2. Define the IP address variables for each node in the ZooKeeper ensemble.
   Replace *Master* with the IP address or hostname of the Control Center master host, and replace *Delegate-A* and *Delegate-B* with the IP addresses or hostnames of the delegate hosts to include in the ensemble:

   ```
   node1=Master
   node2=Delegate-A
   node3=Delegate-B
   ```

   Use the virtual IP address or hostname of the high-availability cluster as the value for `node1`.
3. Set the ZooKeeper node ID to `1`.

   ```
   echo "SERVICED_ISVCS_ZOOKEEPER_ID=1" >> /etc/default/serviced
   ```
4. Specify the nodes in the ZooKeeper ensemble.
   You can copy the following text and paste it in your console:

   ```
   echo "SERVICED_ZK=${node1}:2181,${node2}:2181,${node3}:2181" \
     >> /etc/default/serviced
   ```
5. Specify the nodes in the ZooKeeper quorum.

   ZooKeeper requires a unique quorum definition for each node in its ensemble. To achieve this, replace the IP address or hostname of the master host with `0.0.0.0`.

   You can copy the following text and paste it in your console:

   ```
   q1="1@0.0.0.0:2888:3888"
   q2="2@${node2}:2888:3888"
   q3="3@${node3}:2888:3888"
   echo "SERVICED_ISVCS_ZOOKEEPER_QUORUM=${q1},${q2},${q3}" \
     >> /etc/default/serviced
   ```
6. Specify the timeout for inactive connections.
   You can copy the following text and paste it in your console:

   ```
   echo "SERVICED_ZK_SESSION_TIMEOUT=15" >> /etc/default/serviced
   ```
7. Clean up the Control Center configuration file.
   a. Open `/etc/default/serviced` in a text editor.
   b. Navigate to the end of the file, and cut the line that contains the *SERVICED_ZK* variable declaration at that location.
   c. Locate the original *SERVICED_ZK* variable declaration, and then paste the cut line immediately below it.

- **d** Navigate to the end of the file, and cut the line that contains the *SERVICED_ISVCS_ZOOKEEPER_ID* variable declaration at that location.
- **e** Locate the original *SERVICED_ISVCS_ZOOKEEPER_ID* variable declaration, and then paste the cut line immediately below it.
- **f** Navigate to the end of the file, and cut the line that contains the *SERVICED_ISVCS_ZOOKEEPER_QUORUM* variable declaration at that location.
- **g** Locate the original *SERVICED_ISVCS_ZOOKEEPER_QUORUM* variable declaration, and then paste the cut line immediately below it.
- **h** Navigate to the end of the file, and cut the line that contains the *SERVICED_ZK_SESSION_TIMEOUT* variable declaration at that location.
- **i** Locate the original *SERVICED_ZK_SESSION_TIMEOUT* variable declaration, and then paste the cut line immediately below it.
- **j** Save the file, and then close the editor.

8 Verify the ZooKeeper environment variables.

```
grep -E '^\b*SERVICED' /etc/default/serviced | grep -E '_Z(OO|K)'
```

The following example shows the environment variables for a master host with IP address 198.51.100.135.

Note The value of the *SERVICED_ISVCS_ZOOKEEPER_QUORUM* variable is formatted to fit the available space. The result of the `grep` command shows the variable and value on the same line.

```
SERVICED_ZK=198.51.100.135:2181,198.51.100.136:2181,198.51.100.137:2181
SERVICED_ISVCS_ZOOKEEPER_ID=1
SERVICED_ISVCS_ZOOKEEPER_QUORUM=1@0.0.0.0:2888:3888,\
  2@198.51.100.136:2888:3888,3@198.51.100.137:2888:3888
SERVICED_ZK_SESSION_TIMEOUT=15
```

Repeat this procedure on the other master host node.

Configuring delegate host A as a ZooKeeper node

Use this procedure to configure the delegate host designated as *Delegate-A* as a ZooKeeper node.

1 Log in to the delegate host as `root`, or as a user with superuser privileges.

2 Define the IP address variables for each node in the ZooKeeper ensemble.
Replace *Master* with the IP address or hostname of the Control Center master host, and replace *Delegate-A* and *Delegate-B* with the IP addresses or hostnames of the delegate hosts to include in the ensemble:

```
node1=Master
node2=Delegate-A
node3=Delegate-B
```

Use the virtual IP address or hostname of the high-availability cluster as the value for `node1`.

3 Set the ID of this node in the ZooKeeper ensemble.

```
echo "SERVICED_ISVCS_ZOOKEEPER_ID=2" >> /etc/default/serviced
```

4 Specify the nodes in the ZooKeeper ensemble.
You can copy the following text and paste it in your console:

```
echo "SERVICED_ZK=${node1}:2181,${node2}:2181,${node3}:2181" \
  >> /etc/default/serviced
```

5 Specify the nodes in the ZooKeeper quorum.

ZooKeeper requires a unique quorum definition for each node in its ensemble. To achieve this, replace the IP address or hostname of delegate host A with `0.0.0.0`.

You can copy the following text and paste it in your console:

```
q1="1@${node1}:2888:3888"
q2="2@0.0.0.0:2888:3888"
q3="3@${node3}:2888:3888"
echo "SERVICED_ISVCS_ZOOKEEPER_QUORUM=${q1},${q2},${q3}" \
  >> /etc/default/serviced
```

6 Specify the timeout for inactive connections.
You can copy the following text and paste it in your console:

```
echo "SERVICED_ZK_SESSION_TIMEOUT=15" >> /etc/default/serviced
```

7 Configure Control Center to start the ZooKeeper service.
You can copy the following text and paste it in your console:

```
echo "SERVICED_ISVCS_START=zookeeper" >> /etc/default/serviced
```

8 Clean up the Control Center configuration file.

- **a** Open `/etc/default/serviced` in a text editor.
- **b** Navigate to the end of the file, and cut the line that contains the *SERVICED_ZK* variable declaration at that location.
- **c** Locate the original *SERVICED_ZK* variable declaration, and then paste the cut line immediately below it.
- **d** Comment the original *SERVICED_ZK* declaration, which references only the master host.

  Insert the number sign character (`#`) immediately in front of the original *SERVICED_ZK* variable.
- **e** Navigate to the end of the file, and cut the line that contains the *SERVICED_ISVCS_ZOOKEEPER_ID* variable declaration at that location.
- **f** Locate the original *SERVICED_ISVCS_ZOOKEEPER_ID* variable declaration, and then paste the cut line immediately below it.
- **g** Navigate to the end of the file, and cut the line that contains the *SERVICED_ISVCS_ZOOKEEPER_QUORUM* variable declaration at that location.
- **h** Locate the original *SERVICED_ISVCS_ZOOKEEPER_QUORUM* variable declaration, and then paste the cut line immediately below it.
- **i** Navigate to the end of the file, and cut the line that contains the *SERVICED_ZK_SESSION_TIMEOUT* variable declaration at that location.
- **j** Locate the original *SERVICED_ZK_SESSION_TIMEOUT* variable declaration, and then paste the cut line immediately below it.
- **k** Navigate to the end of the file, and cut the line that contains the *SERVICED_ISVCS_START* variable declaration at that location.
- **l** Locate the original *SERVICED_ISVCS_START* variable declaration, and then paste the cut line immediately below it.
- **m** Save the file, and then close the editor.

9 Verify the ZooKeeper environment variables.

```
grep -E '^\b*SERVICED' /etc/default/serviced \
  | grep -E '(CS_ZO|_ZK|CS_ST)'
```

The following example shows the environment variables for a delegate host with IP address 198.51.100.136.

Note The value of the *SERVICED_ISVCS_ZOOKEEPER_QUORUM* variable is formatted to fit the available space. The result of the `grep` command shows the variable and value on the same line.

```
SERVICED_ZK=198.51.100.135:2181,198.51.100.136:2181,198.51.100.137:2181
SERVICED_ISVCS_START=zookeeper
SERVICED_ISVCS_ZOOKEEPER_ID=2
SERVICED_ISVCS_ZOOKEEPER_QUORUM=1@198.51.100.135:2888:3888,\
  2@0.0.0.0:2888:3888,3@198.51.100.137:2888:3888
SERVICED_ZK_SESSION_TIMEOUT=15
```

Configuring delegate host B as a ZooKeeper node

Use this procedure to configure the delegate host designated as *Delegate-B* as a ZooKeeper node.

1 Log in to the delegate host as `root`, or as a user with superuser privileges.

2 Define the IP address variables for each node in the ZooKeeper ensemble.
Replace *Master* with the IP address or hostname of the Control Center master host, and replace *Delegate-A* and *Delegate-B* with the IP addresses or hostnames of the delegate hosts to include in the ensemble:

```
node1=Master
node2=Delegate-A
node3=Delegate-B
```

Use the virtual IP address or hostname of the high-availability cluster as the value for `node1`.

3 Set the ID of this node in the ZooKeeper ensemble.

```
echo "SERVICED_ISVCS_ZOOKEEPER_ID=3" >> /etc/default/serviced
```

4 Specify the nodes in the ZooKeeper ensemble.
You can copy the following text and paste it in your console:

```
echo "SERVICED_ZK=${node1}:2181,${node2}:2181,${node3}:2181" \
  >> /etc/default/serviced
```

5 Specify the nodes in the ZooKeeper quorum.

ZooKeeper requires a unique quorum definition for each node in its ensemble. To achieve this, replace the IP address or hostname of delegate host B with `0.0.0.0`.

You can copy the following text and paste it in your console:

```
q1="1@${node1}:2888:3888"
q2="2@${node2}:2888:3888"
q3="3@0.0.0.0:2888:3888"
echo "SERVICED_ISVCS_ZOOKEEPER_QUORUM=${q1},${q2},${q3}" \
  >> /etc/default/serviced
```

6 Specify the timeout for inactive connections.
You can copy the following text and paste it in your console:

```
echo "SERVICED_ZK_SESSION_TIMEOUT=15" >> /etc/default/serviced
```

7 Configure Control Center to start the ZooKeeper service.

You can copy the following text and paste it in your console:

```
echo "SERVICED_ISVCS_START=zookeeper" >> /etc/default/serviced
```

8 Clean up the Control Center configuration file.
   a Open `/etc/default/serviced` in a text editor.
   b Navigate to the end of the file, and cut the line that contains the *SERVICED_ZK* variable declaration at that location.
   c Locate the original *SERVICED_ZK* variable declaration, and then paste the cut line immediately below it.
   d Comment the original *SERVICED_ZK* declaration, which references only the master host.

      Insert the number sign character (#) immediately in front of the original *SERVICED_ZK* variable.
   e Navigate to the end of the file, and cut the line that contains the *SERVICED_ISVCS_ZOOKEEPER_ID* variable declaration at that location.
   f Locate the original *SERVICED_ISVCS_ZOOKEEPER_ID* variable declaration, and then paste the cut line immediately below it.
   g Navigate to the end of the file, and cut the line that contains the *SERVICED_ISVCS_ZOOKEEPER_QUORUM* variable declaration at that location.
   h Locate the original *SERVICED_ISVCS_ZOOKEEPER_QUORUM* variable declaration, and then paste the cut line immediately below it.
   i Navigate to the end of the file, and cut the line that contains the *SERVICED_ZK_SESSION_TIMEOUT* variable declaration at that location.
   j Locate the original *SERVICED_ZK_SESSION_TIMEOUT* variable declaration, and then paste the cut line immediately below it.
   k Navigate to the end of the file, and cut the line that contains the *SERVICED_ISVCS_START* variable declaration at that location.
   l Locate the original *SERVICED_ISVCS_START* variable declaration, and then paste the cut line immediately below it.
   m Save the file, and then close the editor.

9 Verify the ZooKeeper environment variables.

```
grep -E '^\b*SERVICED' /etc/default/serviced \
  | grep -E '(CS_ZO|_ZK|CS_ST)'
```

The following example shows the environment variables for a delegate host with IP address 198.51.100.137.

Note The value of the *SERVICED_ISVCS_ZOOKEEPER_QUORUM* variable is formatted to fit the available space. The result of the `grep` command shows the variable and value on the same line.

```
SERVICED_ZK=198.51.100.135:2181,198.51.100.136:2181,198.51.100.137:2181
SERVICED_ISVCS_START=zookeeper
SERVICED_ISVCS_ZOOKEEPER_ID=3
SERVICED_ISVCS_ZOOKEEPER_QUORUM=1@198.51.100.135:2888:3888,\
  2@198.51.100.136:2888:3888,3@0.0.0.0:2888:3888
SERVICED_ZK_SESSION_TIMEOUT=15
```

Importing the ZooKeeper image for Docker

Perform the steps in *Staging a Docker image file on ZooKeeper ensemble nodes* on page 15 before performing this procedure.

Use this procedure to import the Docker image for the ZooKeeper service from a ZooKeeper image file. Repeat this procedure on each node in the ZooKeeper ensemble.

1 Log in to the ZooKeeper ensemble node as `root`, or as a user with superuser privileges.

2 Change directory to `/root`.

```
cd /root
```

3 Extract the ZooKeeper image.

```
./install-zenoss-isvcs-zookeeper-v*.run
```

Image extraction begins when you press the **y** key.

4 Optional: Delete the archive file, if desired.

```
rm -i ./install-zenoss-isvcs-zookeeper-v*.run
```

Starting a ZooKeeper ensemble

Use this procedure to start a ZooKeeper ensemble.

The window of time for starting a ZooKeeper ensemble is relatively short. The goal of this procedure is to restart Control Center on each ensemble node at about the same time, so that each node can participate in electing the leader.

1 Log in to the primary node of the high-availability cluster as `root`, or as a user with superuser privileges.

To ensure you are logging in to the primary node, use the virtual hostname (*HA-Virtual-Name*) or virtual IP address (*HA-Virtual-IP*) of the high-availability cluster.

2 In a separate window, log in to the second node of the ZooKeeper ensemble (*Delegate-A*) as `root`, or as a user with superuser privileges.

3 In a different window, log in to the third node of the ZooKeeper ensemble (*Delegate-B*) as `root`, or as a user with superuser privileges.

4 On the primary node, put the high-availability cluster in standby mode.

```
pcs cluster standby --all
```

5 On the primary node, monitor the status of cluster resources.

```
watch pcs status
```

Monitor the status until all resources report `Stopped`.

6 Start the ZooKeeper ensemble nodes.

a On the primary node, start the high-availability cluster.

```
pcs cluster unstandby --all
```

b On delegate host that are ZooKeeper ensemble nodes, stop and start `serviced`.

```
systemctl stop serviced && systemctl start serviced
```

7 On the primary node, check the status of the ZooKeeper ensemble.

a Attach to the container of the ZooKeeper service.

```
docker exec -it serviced-isvcs_zookeeper /bin/bash
```

b Query the master host and identify its role in the ensemble.

Replace *Master* with the hostname or IP address of the master host:

```
{ echo stats; sleep 1; } | nc Master 2181 | grep Mode
```

The result includes `leader` or `follower`.

c Query delegate host A and identify its role in the ensemble.
Replace *Delegate-A* with the hostname or IP address of delegate host A:

```
{ echo stats; sleep 1; } | nc Delegate-A 2181 | grep Mode
```

d Query delegate host B and identify its role in the ensemble.
Replace *Delegate-B* with the hostname or IP address of delegate host B:

```
{ echo stats; sleep 1; } | nc Delegate-B 2181 | grep Mode
```

e Detach from the container of the ZooKeeper service.

```
exit
```

Updating delegate hosts

The default configuration of delegate hosts sets the value of the *SERVICED_ZK* variable to the master host only. Use this procedure to update the setting to include all of the hosts in the ZooKeeper ensemble. **Perform this procedure on each delegate host that is not a ZooKeeper ensemble node.**

1 Log in to the delegate host as `root`, or as a user with superuser privileges.

2 Define the IP address variables for each node in the ZooKeeper ensemble.
Replace *Master* with the IP address or hostname of the Control Center master host, and replace *Delegate-A* and *Delegate-B* with the IP addresses or hostnames of the delegate hosts to include in the ensemble:

```
node1=Master
node2=Delegate-A
node3=Delegate-B
```

3 Specify the nodes in the ZooKeeper ensemble.
You can copy the following text and paste it in your console:

```
echo "SERVICED_ZK=${node1}:2181,${node2}:2181,${node3}:2181" \
  >> /etc/default/serviced
```

4 Update the variable.

a Open `/etc/default/serviced` in a text editor.

b Navigate to the end of the file, and cut the line that contains the *SERVICED_ZK* variable declaration at that location.

The value of this declaration specifies three endpoints.

c Locate the *SERVICED_ZK* variable near the beginning of the file, and then delete the line it is on.

The value is just the master host endpoint.

d Paste the *SERVICED_ZK* variable declaration from the end of the file in the location of the just-deleted declaration.

e Save the file, and then close the editor.

5 Verify the setting.

```
grep -E '^\b*SERVICED_ZK' /etc/default/serviced
```

The following example shows the environment variable for a delegate host that is not a node in the ZooKeeper ensemble:

```
SERVICED_ZK=198.51.100.135:2181,198.51.100.136:2181,198.51.100.137:2181
```

6 Restart Control Center.

```
systemctl restart serviced
```

Starting and stopping Control Center deployments A

This appendix includes procedures for stopping and starting a high-availability Control Center deployment.

Note The procedures in this appendix assume that Control Center is the only source of Docker containers that are run on a host.

Stopping Control Center

To stop Control Center in a high-availability deployment, perform the procedures in this section, in order.

Stopping a master host node

Use this procedure to stop the Control Center service (`serviced`) on the master host in a high-availability deployment.

1 Use the virtual hostname or virtual IP address of the high-availability cluster to log in to the Control Center master node as `root`, or as a user with superuser privileges.
2 Display the public hostname of the current node.

```
uname -n
```

Make a note of which node (primary or secondary) is the current node, for use in a subsequent step.
3 Stop the top-level service `serviced` is managing, if necessary.
   a Show the status of running services.

   ```
   serviced service status
   ```

   The top-level service is the service listed immediately below the headings line.

   - If the status of the top-level service and all child services is `stopped`, proceed to the next step.
   - If the status of the top-level service and all child services is **not** `stopped`, perform the remaining substeps.

   b Stop the top-level service.
   Replace *Service* with the name or identifier of the top-level service:

   ```
   serviced service stop Service
   ```

c. Monitor the stop.

```
serviced service status
```

When the status of the top-level service and all child services is `stopped`, proceed to the next step.

4. Stop Control Center with the cluster management tool.

```
pcs cluster standby --all
```

5. Monitor the status of cluster resources.

```
watch pcs status
```

Monitor the status until all resources report `Stopped`. Resolve any issues before continuing.

6. Ensure that no containers remain in the local repository.

a. Start the Docker service.

```
systemctl start docker
```

b. Display the identifiers of all containers, running and exited.

```
docker ps -qa
```

If the command returns a result, enter the following command:

```
docker ps -qa | xargs --no-run-if-empty docker rm -fv
```

c. Stop the Docker service.

```
systemctl stop docker
```

7. To ensure that no containers remain in both Docker repositories, log in to the other master node as `root`, or as a user with superuser privileges, and then perform the preceding step.

Stopping a delegate host

Use this procedure to stop the Control Center service (`serviced`) on a delegate host in a multi-host deployment. Repeat this procedure on each delegate host in your deployment.

1. Log in to the delegate host as `root`, or as a user with superuser privileges.
2. Stop the Control Center service.

```
systemctl stop serviced
```

3. Ensure that no containers remain in the local repository.

a. Display the identifiers of all containers, running and exited.

```
docker ps -qa
```

- If the command returns no result, proceed to the next step.
- If the command returns a result, perform the following substeps.

b. Remove all remaining containers.

```
docker ps -qa | xargs --no-run-if-empty docker rm -fv
```

- If the remove command completes, proceed to the next step.
- If the remove command does not complete, the most likely cause is an NFS conflict. Perform the following substeps.

c Stop the NFS and Docker services.

```
systemctl stop nfs && systemctl stop docker
```

d Start the NFS and Docker services.

```
systemctl start nfs && systemctl start docker
```

e Repeat the attempt to remove all remaining containers.

```
docker ps -qa | xargs --no-run-if-empty docker rm -fv
```

- If the remove command completes, proceed to the next step.
- If the remove command does not complete, perform the remaining substeps.

f Disable the automatic startup of `serviced`.

```
systemctl disable serviced
```

g Reboot the host.

```
reboot
```

h Log in to the delegate host as `root`, or as a user with superuser privileges.

i Enable the automatic startup of `serviced`.

```
systemctl enable serviced
```

4 Dismount all filesystems mounted from the Control Center master host.

This step ensures no stale mounts remain when the storage on the master host is replaced.

a Identify filesystems mounted from the master host.

```
awk '/serviced/ { print $1, $2 }' < /proc/mounts \
  | grep -v '/opt/serviced/var/isvcs'
```

- If the preceding command returns no result, stop. This procedure is complete.
- If the preceding command returns a result, perform the following substeps.

b Force the filesystems to dismount.

```
for FS in $(awk '/serviced/ { print $2 }' < /proc/mounts \
  | grep -v '/opt/serviced/var/isvcs')
do
  umount -f $FS
done
```

c Identify filesystems mounted from the master host.

```
awk '/serviced/ { print $1, $2 }' < /proc/mounts \
  | grep -v '/opt/serviced/var/isvcs'
```

- If the preceding command returns no result, stop. This procedure is complete.

- If the preceding command returns a result, perform the following substeps.

d Perform a lazy dismount.

```
for FS in $(awk '/serviced/ { print $2 }' < /proc/mounts \
  | grep -v '/opt/serviced/var/isvcs')
do
  umount -f -l $FS
done
```

e Restart the NFS service.

```
systemctl restart nfs
```

f Determine whether any filesystems remain mounted from the master host.

```
awk '/serviced/ { print $1, $2 }' < /proc/mounts \
  | grep -v '/opt/serviced/var/isvcs'
```

- If the preceding command returns no result, stop. This procedure is complete.
- If the preceding command returns a result, perform the remaining substeps.

g Disable the automatic startup of `serviced`.

```
systemctl disable serviced
```

h Reboot the host.

```
reboot
```

i Log in to the delegate host as `root`, or as a user with superuser privileges.

j Enable the automatic startup of `serviced`.

```
systemctl enable serviced
```

Starting Control Center

Use this procedure to start Control Center in a high-availability deployment. The default configuration of the Control Center service (`serviced`) is to start when the host starts. This procedure is only needed after stopping `serviced` to perform maintenance tasks.

1 Log in to the primary node as `root`, or as a user with superuser privileges.

In this context, the primary node is the node that was the current node when you stopped Control Center.

2 Identify the hosts in the ZooKeeper ensemble.

```
grep -E '^\b*SERVICED_ZK=' /etc/default/serviced
```

The result is a list of 1, 3, or 5 hosts, separated by the comma character (`,`). The master host is always a node in the ZooKeeper ensemble.

3 In separate windows, log in to each of the delegate hosts that are nodes in the ZooKeeper ensemble as `root`, or as a user with superuser privileges.

4 Take the cluster out of standby mode.

```
pcs cluster unstandby --all
```

5 On the other ZooKeeper ensemble hosts, start `serviced`.

The window of time for starting a ZooKeeper ensemble is relatively short. The goal of this step is to start Control Center on each ensemble node at about the same time, so that each node can participate in electing the leader.

```
systemctl start serviced
```

6 Monitor the status of cluster resources.

```
watch pcs status
```

Monitor the status until all resources report `Started`. Resolve any issues before continuing.

7 On the master host, check the status of the ZooKeeper ensemble.

a Attach to the container of the ZooKeeper service.

```
docker exec -it serviced-isvcs_zookeeper bash
```

b Query the master host and identify its role in the ensemble.
Replace *Master* with the hostname or IP address of the master host:

```
{ echo stats; sleep 1; } | nc Master 2181 | grep Mode
```

The result includes `leader` or `follower`. When multiple hosts rely on the ZooKeeper instance on the master host, the result includes `standalone`.

c Query the other delegate hosts to identify their role in the ensemble.
Replace *Delegate* with the hostname or IP address of a delegate host:

```
{ echo stats; sleep 1; } | nc Delegate 2181 | grep Mode
```

d Detach from the container of the ZooKeeper service.

```
exit
```

If none of the nodes reports that it is the ensemble leader within a few minutes of starting `serviced`, reboot the ensemble hosts.

8 Log in to each of the delegate hosts that are not nodes in the ZooKeeper ensemble as `root`, or as a user with superuser privileges, and then start `serviced`.

```
systemctl start serviced
```

9 Optional: Monitor the startup, if desired.

```
journalctl -u serviced -f -o cat
```

Once Control Center is started, it is ready to start managing applications. For more information, refer to the documentation of your application.

Storage management utility B

This appendix includes a description of the `serviced-storage` command, the required utility for creating the Docker thin pool and creating and managing the Control Center application data thin pool.

serviced-storage

The `serviced-storage` command manages Control Center storage.

Use this command to create LVM thin pools for Docker and Control Center.

USAGE

```
serviced-storage  [-h|--help] [-o DeviceMapperOption=Value] \
  [-v] Command [CommandOptions]
```

GLOBAL OPTIONS

--help, -h

Shows the help information.

-o *DeviceMapperOption=Value*

A device mapper option. Applies only to device mapper drivers.

-v

Displays verbose logging.

COMMANDS

check

Check for orphaned devices.

create

Create a volume on a driver.

create-thin-pool

Create an LVM thin pool.

disable

Disable a driver.

init

Initialize a driver.

list
Print volumes on a driver.

mount
Mount an existing volume from a driver.

remove
Remove an existing volume from a driver.

resize
Resize an existing volume.

set
Set the default driver.

status
Print the driver status

sync
Sync data from a volume to another volume.

unset
Unset the default driver.

version
Print the version and exit.

serviced-storage check

The `serviced-storage check` command searches for orphaned snapshot devices in the `serviced` application data thin pool and removes them, if requested. This command requires the path of `serviced` tenant volumes, which is determined by the *SERVICED_VOLUMES_PATH* variable in `/etc/default/serviced`. The default path is `/opt/serviced/var/volumes`.

Syntax:

```
serviced-storage [GlobalOptions] check [-c|--clean] Path
```

Command options:

[-c|--clean]
Remove orphaned snapshot devices.

serviced-storage create-thin-pool

The `serviced-storage create-thin-pool` command creates an LVM thin pool either for Docker data or for Control Center application data. When devices are specified, the command creates an LVM volume group.

Syntax:

```
serviced-storage [GlobalOptions] create-thin-pool \
  [-s|--size]=[Value][G|%] [docker|serviced] \
  [DevicePath [DevicePath...]|VolumeGroupName]
```

Command options:

[-s|--size]=[*Value*][G|%]

The size of the thin pool to create. The size can be a fixed value (in gigabytes) or a relative value (a percentage) of the available storage. When this option is not used, the thin pool size defaults to 90% of the specified storage resource.

serviced-storage resize

The `serviced-storage resize` command increases the size of a `serviced` tenant device in its LVM thin pool. Like LVM thin pools, the size of a `serviced` tenant device can never decrease.

Syntax:

```
serviced-storage [GlobalOptions] resize \
  [-d|--driver]=Value TenantID NewSize
```

Command options:

[-d|--driver]=*Value*

The path of the tenant volume.

EXAMPLES

Create an LVM volume group named `zenoss` and use it for both thin pools:

```
vgcreate zenoss /dev/sdb /dev/sdc
serviced-storage create-thin-pool --size=50G docker zenoss
serviced-storage create-thin-pool --size=50% serviced zenoss
```

If you specify devices or partitions, `serviced-storage` creates an LVM volume group with the same name as the thin pool. The following example yields the same result as the previous, except the name of the volume group is `docker` instead of `zenoss`:

```
serviced-storage create-thin-pool docker /dev/sdb /dev/sdc
serviced-storage create-thin-pool serviced docker
```

Create thin pools on separate block devices:

```
serviced-storage create-thin-pool docker /dev/sdb
serviced-storage create-thin-pool serviced /dev/sdc
```

Create thin pools on separate partitions:

```
serviced-storage create-thin-pool docker /dev/sdb1
serviced-storage create-thin-pool serviced /dev/sdc3
```

Increase the size of the `serviced` LVM thin pool, and then increase the size of a `serviced` tenant device.

```
lvextend -L+300G zenoss/serviced-pool
serviced-storage -o dm.thinpooldev=/dev/mapper/zenoss-serviced--pool \
  resize -d /opt/serviced/var/volumes 58uuetj38draeu9alp6002b1y 200G
```

Identify the `serviced` application data thin pool, and then remove orphaned snapshot devices.

```
ls /dev/mapper | grep serviced
serviced-storage -o dm.thinpooldev=/dev/mapper/zenoss-serviced--pool \
  check -c /opt/serviced/var/volumes
```

Resolving package dependency conflicts C

This appendix includes procedures for resolving common Docker CE and Control Center dependency conflicts.

Resolving device mapper dependency conflicts

To perform this procedure, you need:

- An RHEL/CentOS system with internet access and the same operating system release and kernel as the Control Center hosts in your deployment.
- A secure network copy program.

Use this procedure to resolve dependency issues in which the installed versions of device mapper libraries are newer than the versions included in the Zenoss mirror. The following example shows a typical `yum` error of this type:

```
Error: Package: 7:device-mapper-event-1.02.107-5.el7.x86_64 (zenoss-
mirror)
Requires: device-mapper = 7:1.02.107-5.el7
Installed: 7:device-mapper-1.02.107-5.el7_2.5.x86_64 (@updates)
device-mapper = 7:1.02.107-5.el7_2.5
```

Follow these steps:

1 Display the version number of the installed device mapper package.

```
rpm -q device-mapper | cut -d - -f 3-
```

Example result:

```
1.02.135-1.el7_3.1.x86_64
```

Record the version number for subsequent use.

2 Log in to a compatible host that is connected to the internet as `root`, or as a user with superuser privileges.

The host must have the same operating system (RHEL or CentOS) and release installed as the Control Center hosts in your deployment.

3 Install `yum` utilities, if necessary.

a Determine whether the `yum` utilities package is installed.

```
rpm -qa | grep yum-utils
```

- If the command returns a result, the package is installed. Proceed to the next step.
- If the command does not return a result, the package is not installed. Perform the following substep.

b Install the `yum-utils` package.

```
yum install yum-utils
```

4 Download the required dependencies, and then create a `tar` archive of the files.

a Create a variable for the dependency version to download.
Replace *Device-Mapper-Version* with the version number displayed in a previous step:

```
myVersion=Device-Mapper-Version
```

b Create a temporary directory for the dependencies.

```
mkdir /tmp/downloads
```

c Download the dependencies to the temporary directory.

```
yum install --downloadonly --downloaddir=/tmp/downloads \
  device-mapper-event-$myVersion
```

The `yum` command downloads two package files.

d Create a `tar` archive of the temporary directory.

```
cd /tmp && tar czf ./downloads.tgz ./downloads
```

5 Use a secure copy program to copy the archive file to the `/tmp` directory of the Control Center host or hosts that need the dependencies.

6 Log in to the host as `root`, or as a user with superuser privileges.

7 Install the device mapper dependencies.

a Extract the packages from the `tar` archive.

```
cd /tmp && tar xzf ./downloads.tgz
```

b Install the dependencies.

```
yum install $(ls /tmp/downloads/*.rpm)
```

Return to the procedure you were performing before turning to this appendix and retry the `yum` install command that failed previously.

Resolving other dependency conflicts

Use this procedure to resolve dependency issues in which the installed versions of one or more dependencies are newer than the versions included in the Zenoss mirror. The following example shows a typical `yum` error of this type:

```
Error: Package: policycoreutils-python-2.5-9.el7.x86_64 (zenoss-mirror)
Requires: policycoreutils = 2.5-9.el7
```

```
Installed: policycoreutils-2.5-11.el7_3.x86_64 (@updates)
```

Follow these steps:

1 Install the older package.
Replace *Package-Name* with the name of the package displayed in the error message:

```
rpm -Uvh --oldpackage Package-Name
```

2 Clean all `yum` caches.

```
yum clean all
```

Return to the procedure you were performing before turning to this appendix and retry the `yum` install command that failed previously.

Control Center configuration variables D

This appendix includes recommended best practices for editing the `serviced` configuration file, and descriptions of the variables in the file.

Best practices for configuration files

The Control Center configuration file, `/etc/default/serviced`, contains Bash environment variables that are read by the `serviced` daemon startup script. The following list describes recommended best practices for its use and maintenance:

1. When in doubt, make a backup. Before editing, making a backup copy of the configuration file is always the safest choice.
2. Copy a variable, then edit the copy. If you need to revert a variable to its default value, you don't have to leave the file to look it up.
3. Copy and edit a variable only if the default value needs to be changed. It's easier to troubleshoot problems when only non-default variables are copied and edited.
4. Put the first character of the variable declaration in the first column of its line. It's easier to `grep` for settings when each one starts a line.
5. Add customizations to the top of the file. Customizations at the end of the file or scattered throughout the file may be overlooked.
6. In high-availability deployments, the contents of `/etc/default/serviced` on the master nodes must be identical. Use a utility like `sum` to compare the files quickly.

Control Center configuration file

The Control Center configuration file, `/etc/default/serviced`, contains Bash environment variables that are read by the `serviced` daemon startup script. The order of the following list matches the order of the variables in the file.

HOME

Default: (the value of shell variable *HOME*)

The path Docker clients use to locate the `.docker/config.json` authentication file, which contains Docker Hub credentials.

TMPDIR

Default: (the value of shell variable *TMPDIR*)

The path `serviced` uses for temporary files.

GOMAXPROCS

Default: `2`

The maximum number of CPU cores `serviced` uses.

SERVICED_MASTER

Default: `1` (true)

Assigns the role of a `serviced` instance, either master or delegate. The master runs the application services scheduler and other internal services. Delegates run the application services assigned to the resource pool to which they belong.

Only one `serviced` instance can be the master; all other instances must be delegates. The default value assigns the master role. To assign the delegate role, set the value to `0` (false). This variable must be explicitly set on all Control Center hosts.

SERVICED_MASTER_IP

Default: `127.0.0.1`

A convenience variable, for use in places where the IP address or hostname of the master host is required. This variable is unused unless it is both set here and referenced elsewhere. (For example, by replacing `{{SERVICED_MASTER_IP}}` with `$SERVICED_MASTER_IP`.)

SERVICED_MASTER_POOLID

Default: `default`

The name of the default resource pool. This variable is only used the first time `serviced` is started.

SERVICED_ZK

Default: (none)

The list of endpoints in the `serviced` ZooKeeper ensemble, separated by the comma character (`,`). Each endpoint identifies an ensemble node. Each Control Center server and in-container proxy uses *SERVICED_ZK* to create a randomized, round-robin list, and cycles through the list when it attempts to establish a connection with the lead ZooKeeper host.

SERVICED_DOCKER_REGISTRY

Default: `localhost:5000`

The endpoint of the local Docker registry, which `serviced` uses to store internal services and application images.

If the default value is changed, the host's Docker configuration file must include the `--insecure-registry` flag with the same value as this variable.

The safest replacement for `localhost` is the IPv4 address of the registry host. Otherwise, the fully-qualified domain name of the host must be specified.

SERVICED_OUTBOUND_IP

Default: (none)

The IPv4 address that delegates use to connect to the master host. When no address is specified, `serviced` attempts to discover its public IP address by pinging `google.com`.

This variable must be set on all Control Center hosts in either of the following scenarios:

- Control Center is deployed behind a firewall and `google.com` is not reachable. Set the value to the IPv4 address of the master host.
- Control Center is deployed in a high-availability cluster. Set the value to the virtual IPv4 address of the high-availability cluster (*HA-Virtual-IP*).

Note Setting the Docker *HTTP_PROXY* or *HTTPS_PROXY* environment variables prevents access to the IP address defined with this variable. To enable access, unset the Docker variables, and then reboot the host.

SERVICED_STATIC_IPS

Default: (none)

A list of one or more static IP addresses that are available for IP assignment. Use the comma character (`,`) to separate addresses.

SERVICED_ENDPOINT

Default: `{{SERVICED_MASTER_IP}}:4979`

The endpoint of the `serviced` RPC server. Replace `{{SERVICED_MASTER_IP}}` with the IP address or hostname of the `serviced` master host. The port number of this endpoint must match the value of the *SERVICED_RPC_PORT* variable defined on the `serviced` master host.

SERVICED_MAX_RPC_CLIENTS

Default: `3`

The preferred maximum number of simultaneous connections a `serviced` delegate uses for RPC requests. The value is used to create a pool of sockets, which are reused as needed. Increasing the value increases the number of open sockets and the use of socket-related operating system resources.

When the demand for connections exceeds the supply of open sockets, `serviced` opens more sockets. When demand eases, `serviced` reduces the number of open sockets to the preferred maximum.

SERVICED_RPC_PORT

Default: `4979`

The port on which the `serviced` RPC server listens for connections. The value of this variable must match the port number defined for the *SERVICED_ENDPOINT* variable on all `serviced` delegate hosts.

SERVICED_RPC_CERT_VERIFY

Default: `false`

Determines whether `serviced` performs TLS certificate verification for RPC connections. The certificate is defined by the *SERVICED_CERT_FILE* variable.

SERVICED_RPC_DISABLE_TLS

Default: `false`

Determines whether `serviced` encrypts RPC traffic with TLS.

SERVICED_RPC_TLS_MIN_VERSION

Default: `VersionTLS10`

The minimum version of TLS `serviced` accepts for RPC connections. Valid values include the default, `VersionTLS11`, and `VersionTLS12`.

SERVICED_RPC_TLS_CIPHERS

Default: (list of ciphers)

The list of TLS ciphers `serviced` prefers for RPC connections, separated by the comma character (`,`):

- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`
- `TLS_RSA_WITH_AES_128_CBC_SHA`
- `TLS_RSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`

Other ciphers are supported; the preceding ciphers provide strong security for relatively low processing overhead.

An instance of `serviced` is on both ends of an RPC connection, so both daemons use the first cipher in the list. To use a different cipher, put it first in the list, on all Control Center hosts.

SERVICED_UI_PORT

Default: `:443`

The port on which the `serviced` HTTP server listens for requests for its internal services and for tenant services. The value may be expressed as follows:

IP-Address:*Port-Number*
:*Port-Number*
Port-Number

Tenant applications can specify alternative ports with the port public endpoint feature.

The value of this variable must be identical on all Control Center hosts in a deployment.

SERVICED_UI_POLL_FREQUENCY

Default: `3`

The number of seconds between polls from Control Center browser interface clients. The value is included in a JavaScript library that is sent to the clients.

SERVICED_MUX_PORT

Default: `22250`

The port `serviced` uses for traffic among Docker containers.

SERVICED_MUX_DISABLE_TLS

Default: `0`

Determines whether inter-host traffic among Docker containers is encrypted with TLS. Intra-host traffic among Docker containers is not encrypted. To disable encryption, set the value to `1`.

SERVICED_MUX_TLS_MIN_VERSION

Default: `VersionTLS10`

The minimum version of TLS `serviced` accepts for mux traffic. Valid values include the default, `VersionTLS11`, and `VersionTLS12`.

SERVICED_MUX_TLS_CIPHERS

Default: (list of ciphers)

The list of TLS ciphers `serviced` prefers for mux traffic, separated by the comma character (`,`):

- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`
- `TLS_RSA_WITH_AES_128_CBC_SHA`
- `TLS_RSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`

Other ciphers are supported; the preceding ciphers provide strong security for relatively low processing overhead.

An instance of `serviced` is on both ends of a mux connection, so both daemons use the first cipher in the list. To use a different cipher, put it first in the list, on all Control Center hosts.

SERVICED_ISVCS_PATH

Default: `/opt/serviced/var/isvcs`

The location of `serviced` internal services data.

SERVICED_VOLUMES_PATH

Default: `/opt/serviced/var/volumes`

The location of `serviced` application data.

SERVICED_BACKUPS_PATH

Default: `/opt/serviced/var/backups`

The location of `serviced` backup files.

SERVICED_LOG_PATH

Default: `/var/log/serviced`

The location of `serviced` audit log files. Non-audit (operations) messages are written to `journald`.

SERVICED_KEY_FILE

Default: `$TMPDIR/zenoss_key.[0-9]+`

The path of a digital certificate key file. Choose a location that is not modified during operating system updates, such as `/etc`.

This key file is used for all TLS-encrypted communications (RPC, mux, and HTTP). The default, insecure key file is created when the `serviced` web server first starts, and is based on a public key that is compiled into `serviced`.

SERVICED_CERT_FILE

Default: `$TMPDIR/zenoss_cert.[0-9]+`

The path of a digital certificate file. Choose a location that is not modified during operating system updates, such as `/etc`. Certificates with passphrases are not supported.

This certificate file is used for all TLS-encrypted communications (RPC, mux, and HTTP). The default, insecure certificate file is created when the `serviced` web server first starts, and is based on a public certificate that is compiled into `serviced`.

SERVICED_TLS_MIN_VERSION

Default: `VersionTLS10`

The minimum version of TLS that `serviced` accepts for HTTP traffic. Valid values include the default, `VersionTLS11`, and `VersionTLS12`.

SERVICED_TLS_CIPHERS

Default: (list of ciphers)

The list of TLS ciphers that `serviced` accepts for HTTP traffic, separated by the comma character (`,`):

1. `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
2. `TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256`
3. `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
4. `TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384`
5. `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`
6. `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`
7. `TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA`
8. `TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA`
9. `TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA`
10. `TLS_RSA_WITH_AES_256_CBC_SHA`
11. `TLS_RSA_WITH_AES_128_CBC_SHA`
12. `TLS_RSA_WITH_3DES_EDE_CBC_SHA`
13. `TLS_RSA_WITH_AES_128_GCM_SHA256`
14. `TLS_RSA_WITH_AES_256_GCM_SHA384`

To disable support for most ciphers, you can remove them from the list. The following rules apply to the list:

- The first cipher, `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`, must always be present in the list of ciphers.
- The first four ciphers in the list must always precede any of the ciphers that appear after the first four. The first four ciphers are valid for HTTP/2, while the remaining ciphers are not.

SERVICED_FS_TYPE

Default: `devicemapper`

The driver to manage application data storage on the `serviced` master host. Only `devicemapper` is supported in production deployments.

The only supported storage layout for the `devicemapper` driver is an LVM thin pool. To create a thin pool, use the `serviced-storage` utility. To specify the name of the thin pool device, use the *SERVICED_DM_THINPOOLDEV* variable.

SERVICED_DM_ARGS

Default: (none)

Customized startup arguments for the `devicemapper` storage driver.

SERVICED_DM_BASESIZE

Default: `100G`

The base size of virtual storage devices for tenants in the application data thin pool, in gigabytes. The units symbol (`G`) is required. This variable is used when `serviced` starts for the first time, to set the initial size of tenant devices, and when a backup is restored, to set the size of the restored tenant device.

The base size device is sparse device that occupies at most 1MB of space in the application data thin pool; its size has no immediate practical impact. However, the application data thin pool should have enough space for twice the size of each tenant device it supports, to store both the data itself and snapshots of the data. Since the application data thin pool is an LVM logical volume, its size can be increased at any time. Likewise, the size of a tenant device can be increased, as long as the available space in the thin pool can support the larger tenant device plus snapshots.

SERVICED_DM_LOOPDATASIZE

Default: `100G`

Specifies the size of the data portion of the loop-back file. This setting is ignored when *SERVICED_ALLOW_LOOP_BACK* is `false`.

SERVICED_DM_LOOPMETADATASIZE

Default: `2G`

Specifies the size of the metadata portion of the loop-back file. This setting is ignored when *SERVICED_ALLOW_LOOP_BACK* is `false`.

SERVICED_DM_THINPOOLDEV

Default: (none)

The name of the thin pool device to use with the `devicemapper` storage driver.

SERVICED_STORAGE_STATS_UPDATE_INTERVAL

Default: `300` (5 minutes)

The number of seconds between polls of kernel statistics about the application data thin pool.

This setting is ignored when the operating system kernel version is less than 3.10.0-366.

SERVICED_ALLOW_LOOP_BACK

Default: `false`

Determines whether loop-back files can be used with the `devicemapper` storage driver. This option is not supported for production use.

SERVICED_MAX_CONTAINER_AGE

Default: `86400` (24 hours)

The number of seconds `serviced` waits before removing a stopped container.

SERVICED_VIRTUAL_ADDRESS_SUBNET

Default: `10.3.0.0/16`

The private subnet for containers that use virtual IP addresses on a host. This value may be unique on each Control Center host, if necessary.

RFC 1918 restricts private networks to the 10.0/24, 172.16/20, and 192.168/16 address spaces. However, `serviced` accepts any valid IPv4 address space.

Specify the value in CIDR notation. A `/29` network provides sufficient address space.

SERVICED_LOG_LEVEL

Default: `0`

The log level `serviced` uses when writing to the system log. Valid values are `0` (normal) and `2` (debug).

SERVICED_LOG_ADDRESS

Default: `{{SERVICED_MASTER_IP}}:5042`

The endpoint of the logstash service. Replace `{{SERVICED_MASTER_IP}}` with the IP address or hostname of the `serviced` master host.

SERVICED_LOGSTASH_ES

Default: `{{SERVICED_MASTER_IP}}:9100`

The endpoint of the Elasticsearch service for logstash. On delegate hosts, replace `{{SERVICED_MASTER_IP}}` with the IP address or hostname of the Elasticsearch host, which by default is the `serviced` master host.

SERVICED_LOGSTASH_MAX_DAYS

Default: `14`

The maximum number of days to keep application logs in the logstash database before purging them.

SERVICED_LOGSTASH_MAX_SIZE

Default: `10`

The maximum size of the logstash database, in gigabytes.

SERVICED_LOGSTASH_CYCLE_TIME

Default: `6`

The amount of time between logstash purges, in hours.

SERVICED_STATS_PORT

Default: `{{SERVICED_MASTER_IP}}:8443`

The endpoint of the `serviced` metrics consumer service. Replace `{{SERVICED_MASTER_IP}}` with the IP address or hostname of the `serviced` master host.

SERVICED_STATS_PERIOD

Default: `10`

The frequency, in seconds, at which delegates gather metrics to send to the `serviced` metrics consumer service on the master host.

SERVICED_SVCSTATS_CACHE_TIMEOUT

Default: `5`

The number of seconds to cache statistics about services. The cache is used by Control Center browser interface clients.

SERVICED_DEBUG_PORT

Default: `6006`

The port on which `serviced` listens for HTTP requests for the *Go profiler*. To stop listening for requests, set the value to `-1`.

SERVICED_ISVCS_ENV_[0-9]+

Default: (none)

Startup arguments to pass to internal services. You may define multiple arguments, each for a different internal service. The variables themselves, and their arguments, use the following syntax:

`SERVICED_ISVCS_ENV_%d`

Each variable name ends with a unique integer in place of *%d*.

`Service-Name`*`:`*`Key=Value`

The value of each variable includes the following elements, in order:

1. *Service-Name*, the internal service name. The following command returns the internal service names that may be used for *Service-Name*:

   ```
   docker ps | awk '/serviced-isvcs:/{print $NF}'
   ```
2. The colon character (**:**).
3. *Key*, a variable to pass to the internal service.
4. The equals sign character (=).
5. *Value*, the definition of the variable to pass to the internal service.

The following example variable passes `ES_JAVA_OPTS=-Xmx4g` to the Elasticsearch internal service.

```
SERVICED_ISVCS_ENV_0=serviced-isvcs_elasticsearch-
logstash:ES_JAVA_OPTS=-Xmx4g
```

SERVICED_ADMIN_GROUP

Default: `wheel`

The name of the Linux group on the `serviced` master host whose members are authorized to use the `serviced` browser interface. You may replace the default group with a group that does not have superuser privileges.

SERVICED_ALLOW_ROOT_LOGIN

Default: `1` (true)

Determines whether the `root` user account on the `serviced` master host may be used to gain access to the `serviced` browser interface.

SERVICED_IPTABLES_MAX_CONNECTIONS

Default: `655360`

The default value of this variable ensures that a `serviced` delegate does not run out of connections if the `serviced` master goes down. The connections are automatically cleaned up by the kernel soon after the `serviced` master comes back online.

SERVICED_SNAPSHOT_TTL

Default: `12`

The number of hours an application data snapshot is retained before removal. To disable snapshot removal, set the value to zero. The application data storage can fill up rapidly when this value is zero or too high.

SERVICED_NFS_CLIENT

Default: `1`

DEPRECATED: Prevent a delegate host from mounting the DFS.

SERVICED_SERVICE_MIGRATION_TAG

Default: `1.0.2`

Overrides the default value for the service migration image.

SERVICED_ISVCS_START

Default: (none)

Enables one or more internal services to run on a delegate host. Currently, only `zookeeper` has been tested.

SERVICED_ISVCS_ZOOKEEPER_ID

Default: (none)

The unique identifier of a ZooKeeper ensemble node. The identifier must be a positive integer.

SERVICED_ISVCS_ZOOKEEPER_QUORUM

Default: (none)

The comma-separated list of nodes in a ZooKeeper ensemble. Each entry in the list specifies the ZooKeeper ID, IP address or hostname, peer communications port, and leader communications port of a node in the ensemble. Each quorum definition must be unique, so the IP address or hostname of the "current" host must be 0.0.0.0.

The following example shows the syntax of a node entry:

```
ZooKeeper-ID@Host-IP-Or-Name:2888:3888
```

SERVICED_DOCKER_LOG_DRIVER

Default: `json-file`

The log driver for all Docker container logs, including containers for Control Center internal services. Valid values:

- `json-file`
- `syslog`
- `journald`
- `gelf`
- `fluentd`
- `none`

This is a direct port of the Docker `--log-driver` option.

SERVICED_DOCKER_LOG_CONFIG

Default: `max-file=5,max-size=10m`

A comma-separated list of Docker `--log-opt` options as *`key=value`* pairs. To specify the default values for a log driver, or for drivers that need no additional options, such as `journald`, use a single comma character (`,`) as the value of this variable.

SERVICED_DOCKER_DNS

Default: (empty)

The IP address of one or more DNS servers. The value of this variable is injected into each Docker container that `serviced` starts. Separate multiple values with the comma character (`,`).

SERVICED_OPTS

Default: (empty)

Special options for the `serviced` startup command.

SERVICED_SNAPSHOT_USE_PERCENT

Default: `20`

The amount of free space in the thin pool specified with *SERVICED_DM_THINPOOLDEV*, expressed as a percentage the total size. This value is used to determine whether the thin pool can hold a new snapshot.

SERVICED_ZK_SESSION_TIMEOUT

Default: `15`

The number of seconds the lead ZooKeeper host waits before flushing an inactive connection.

SERVICED_ZK_CONNECT_TIMEOUT

Default: `1`

The number of seconds Control Center waits for a connection to the lead ZooKeeper host.

SERVICED_ZK_PER_HOST_CONNECT_DELAY

Default: `0`

The number of seconds Control Center waits before attempting to connect to the next host in its round-robin list of ZooKeeper hosts. For more information about the round-robin list, see *SERVICED_ZK*.

SERVICED_ZK_RECONNECT_START_DELAY

Default: `1`

SERVICED_ZK_RECONNECT_START_DELAY and *SERVICED_ZK_RECONNECT_MAX_DELAY* are used together when Control Center is unable to re-establish a connection with the lead ZooKeeper host.

To prevent unnecessary spikes in TCP traffic, Control Center waits a randomized amount of time that is equal to plus or minus 20% of the value of *SERVICED_ZK_RECONNECT_START_DELAY*. If Control Center is unable to reconnect after contacting all of the hosts in its round-robin list of ZooKeeper hosts, the wait time is increased by a randomized value and the process of attempting to reconnect begins again. If the attempts fail again, the process repeats until the wait time reaches the value of *SERVICED_ZK_RECONNECT_MAX_DELAY*, and the wait time of subsequent reconnection attempts is capped at *SERVICED_ZK_RECONNECT_MAX_DELAY*. Once connection is re-established, the wait time is reset to *SERVICED_ZK_RECONNECT_START_DELAY*.

For more information about the round-robin list, see *SERVICED_ZK*.

SERVICED_ZK_RECONNECT_MAX_DELAY

Default: `1`

See *SERVICED_ZK_RECONNECT_START_DELAY*.

SERVICED_ES_STARTUP_TIMEOUT

Default: `240`

The number of seconds to wait for the Elasticsearch service to start.

SERVICED_MAX_DFS_TIMEOUT

Default: `300`

The number of seconds until a DFS snapshot attempt times out.

SERVICED_RPC_DIAL_TIMEOUT

Default: `30`

The number of seconds until an RPC connection attempt times out.

SERVICED_AUTH_TOKEN_EXPIRATION

Default: `3600` (1 hour)

The expiration time, in seconds, of delegate authentication tokens. This timeout affects RPC, mux, and `serviced` internal services endpoint communications.

SERVICED_CONTROLLER_BINARY

Default: `/opt/serviced/bin/serviced-controller`

The path of the `serviced-controller` binary, which runs in every container that `serviced` manages.

SERVICED_HOME

Default: `/opt/serviced`

The path of the home directory for `serviced`.

SERVICED_ETC_PATH

Default: `/opt/serviced/etc`

The path of the directory for `serviced` configuration files. The default is *`SERVICED_HOME`*`/etc`.

SERVICED_VHOST_ALIASES

Default: (none)

A list of hostname aliases for a host; for example, `localhost`. Separate multiple values with the comma character (`,`).