Control Center Upgrade Guide

Release 1.4.1

Zenoss, Inc.

www.zenoss.com

Control Center Upgrade Guide

Part Number: 1300.17.268

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

Contents

About this guide.....7
Supported operating systems and browsers..... 7
Documentation feedback.....7
Change history..... 7

Chapter 1: Documented upgrade paths.....10
Release dates.....10
Upgrade paths included in this document.....11

Chapter 2: Downloading and staging required files.....13
Downloading required files..... 13
Installing the repository mirror.....15
Staging Docker image files on the master host..... 15
Staging a Docker image file on ZooKeeper ensemble nodes..... 16

Part I: Upgrading 1.3.x or 1.4.0 to 1.4.1.....17

Chapter 3: Before upgrading from 1.3.x or 1.4.0 to 1.4.1.....18
New features that affect upgrades.....18
Upgrade best practices.....19

Chapter 4: Stopping a Control Center deployment..... 20
Stopping Control Center (single-host deployment).....20
Stopping Control Center (multi-host deployment).....21

Chapter 5: Upgrading a master host from 1.3.x or 1.4.0 to 1.4.1.....25
Identifying storage for audit logging.....25
Updating to Docker CE.....25
Loading image files..... 26
Updating Control Center on the master host..... 27

Chapter 6: Upgrading delegates from 1.3.x or 1.4.0 to 1.4.1..... 29
Updating to Docker CE.....29
Updating Control Center on delegate hosts..... 30
Updating the ZooKeeper image on ensemble nodes.....31

Chapter 7: Starting the ZooKeeper ensemble..... 33

Chapter 8: After upgrading from 1.3.x or 1.4.0 to 1.4.1.....35
Removing unused images.....35

Part II: Upgrading 1.2.x to 1.4.1 36

Chapter 9: Before upgrading from 1.2.x to 1.4.1 37
New features that affect upgrades from release 1.2.x 37
Upgrade best practices 38

Chapter 10: Stopping a Control Center deployment 39
Stopping Control Center (single-host deployment) 39
Stopping Control Center (multi-host deployment) 40

Chapter 11: Upgrading a master host from 1.2.x to 1.4.1 44
Identifying storage for audit logging 44
Updating to Docker CE 44
Loading image files 45
Updating Control Center on the master host 46

Chapter 12: Upgrading delegates from 1.2.x to 1.4.1 48
Updating to Docker CE 48
Updating Control Center on delegate hosts 49
Updating the ZooKeeper image on ensemble nodes 50

Chapter 13: Starting the ZooKeeper ensemble 52

Chapter 14: After upgrading from 1.2.x to 1.4.1 54
Removing unused images 54

Part III: Upgrading 1.1.x to 1.4.1 55

Chapter 15: Before upgrading from 1.1.x to 1.4.1 56
New features that affect upgrades from 1.1.x 56
Upgrade best practices 57

Chapter 16: Stopping a Control Center deployment 59
Stopping Control Center (single-host deployment) 59
Stopping Control Center (multi-host deployment) 60

Chapter 17: Upgrading a master host from 1.1.x to 1.4.1 64
Identifying storage for audit logging 64
Updating to Docker CE 64
Configuring Docker 66

Optional: Changing the local Docker registry endpoint 67
Loading image files 68
Updating Control Center on the master host 69

Chapter 18: Upgrading delegates from 1.1.x to 1.4.1 71
Updating to Docker CE 71
Configuring Docker 72
Changing the local Docker registry endpoint 74
Updating Control Center on delegate hosts 75
Updating the ZooKeeper image on ensemble nodes 76

Chapter 19: Starting an upgraded deployment 77
Starting and registering the master host (single-host deployment) 77
Starting and registering the master host (multi-host deployment) 78
Updating delegate hosts with authentication 79
Starting the ZooKeeper ensemble 82

Chapter 20: After upgrading from 1.1.x to 1.4.1 84
Updating resource pool permissions 84
Setting the connection timeout of a resource pool 85
Removing unused images 85
Removing orphaned snapshot devices 86
Updating the OpenTSDB time-to-live value 87
Managing maintenance scripts 88

Appendix A: Control Center application data storage requirements 90
Examining application data storage status 90
Adding space to the metadata area of a Control Center thin pool 91
Adding space to the data area of a Control Center thin pool 92
Adding space to a tenant volume 92

Appendix B: Storage management utility 94
serviced-storage 94

Appendix C: User access control 98
Adding users to the default browser interface access group 98
Configuring a regular group as the Control Center browser interface access group 99
Enabling use of the command-line interface 100

Appendix D: Configuring a private master NTP server 101
Configuring an NTP master server 101
Configuring NTP clients 102

Appendix E: Resolving package dependency conflicts 104
Resolving device mapper dependency conflicts 104
Resolving other dependency conflicts 105

Appendix F: Control Center releases and images..... 107
Releases and image tags..... 107
Identifying installed Docker images..... 108
Removing unused images..... 108

Appendix G: Control Center configuration variables..... 109
Removed variables (at version 1.2.0)..... 109
New variables (at version 1.2.0)..... 110
Master host configuration variables..... 111
Delegate host configuration variables..... 114
Universal configuration variables..... 116
Best practices for configuration files..... 118
Control Center configuration file..... 118

About this guide

The *Control Center Upgrade Guide* provides detailed instructions for performing the following upgrades:

- Upgrade from version 1.3.x to version 1.4.1
- Upgrade from version 1.2.x to version 1.4.1
- Upgrade from version 1.1.x to version 1.4.1

Zenoss customers: This guide does not include procedures for upgrading a high-availability deployment. For more information, refer to the *Control Center Upgrade Guide for High-Availability Deployments*.

Supported operating systems and browsers

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

Client OS	Supported Browsers
Windows 7 and 8.1	Internet Explorer 11*
Windows 10	Internet Explorer 11*
	Firefox 50 and later
	Chrome 54 and later
	Microsoft Edge
Windows Server 2012 R2	Firefox 30
	Chrome 36
Macintosh OS/X 10.9	Firefox 30 and above
	Chrome 36 and above
Ubuntu 14.04 LTS	Firefox 30 and above
	Chrome 37 and above
Red Hat Enterprise Linux 6.5, CentOS 6.5	Firefox 30 and above
	Chrome 37 and above

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 Upgrade Guide*) and part number (1300.17.268) and as much information as possible about the context of your feedback.

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*.

* Enterprise mode only; compatibility mode is not supported.

1300.17.268

Update release number (1.4.1).

1300.17.229

Update release number (1.4.0).

Correct a previous modification for configuring Docker. The `--insecure-registry` flag must be set when the value of *SERVICED_DOCKER_REGISTRY* is not `localhost:5000`.

1300.17.187

Modify the `yum` command used to install Docker CE 17.03.1.

1300.17.172

Update release number (1.3.3).

Add support for Docker CE 17.03.1.

Add ZooKeeper variables for tuning TCP/IP communications with resource pools.

Remove step for disabling SELinux.

1300.17.122

Update release number (1.3.2)

1300.17.100

Update release number (1.3.1).

Add a part for upgrading 1.3.x systems to the latest release.

Move thin pool resize procedures to an appendix.

Modify the step for configuring Docker on master hosts. The `--insecure-registry` flag is only needed on delegate hosts.

Move information about Docker images to an appendix.

1300.17.076

Update release number (1.3.0).

Change space requirements for the master.

1300.17.058

Update release number (1.2.3).

Remove `TLS_RSA_WITH_RC4_128_SHA` from list of ciphers associated with *SERVICED_TLS_CIPHERS*.

Remove section for configuring `dnsmasq`.

1300.17.024

Update release number (1.2.2).

1300.16.351

Add sections for configuring `dnsmasq` and removing consistent network device naming.

Simplify Docker install and configuration steps.

Add sections about the maintenance scripts that are installed.

1300.16.350

Add a chapter of details about releases and upgrades.

Create a part for the minor release upgrade chapters.

Add a part and new chapters for micro release upgrades.

Add steps to download and use the `serviced` RPM file (Zenoss customers only).

Add a procedure for updating ZooKeeper images on offline nodes (Zenoss customers only).

Clarify the procedures for stopping a cluster.

Update release number (1.2.1).

1300.16.327

Add a description of a new feature, setting the connection timeout value of a resource pool, to the pre-upgrade chapter.

Add a procedure for the new feature to the post-upgrade chapter.

1300.16.322

Initial release (1.2.0).

1 Documented upgrade paths

This chapter includes the dates of Control Center releases and the upgrade paths that are documented in this guide.

Release dates

Table 1: Release 1.4

Release	Date
Control Center 1.4.1	25 Sep 2017
Control Center 1.4.0	17 Aug 2017

Table 2: Release 1.3

Release	Date
Control Center 1.3.3	20 Jun 2017
Control Center 1.3.2	03 May 2017
Control Center 1.3.1	13 Apr 2017
Control Center 1.3.0	09 Mar 2017

Table 3: Release 1.2

Release	Date
Control Center 1.2.3	27 Feb 2017
Control Center 1.2.2	25 Jan 2017
Control Center 1.2.1	16 Dec 2016
Control Center 1.2.0	14 Nov 2016

Table 4: Release 1.1

Release	Date
Control Center 1.1.10	23 Nov 2016
Control Center 1.1.9	17 Oct 2016
Control Center 1.1.8	20 Sep 2016
Control Center 1.1.7	20 Jul 2016
Control Center 1.1.6	28 Jun 2016
Control Center 1.1.5	01 Jun 2016
Control Center 1.1.4	24 May 2016
Control Center 1.1.3	20 Apr 2016
Control Center 1.1.2	04 Mar 2016
Control Center 1.1.1	29 Feb 2016

Table 5: Release 1.0

Release	Date
Control Center 1.0.10	20 Feb 2016
Control Center 1.0.9	02 Dec 2015
Control Center 1.0.8	16 Nov 2015
Control Center 1.0.7	10 Oct 2015
Control Center 1.0.6	14 Sep 2015
Control Center 1.0.5	05 Aug 2015
Control Center 1.0.4	10 Jul 2015
Control Center 1.0.3	27 May 2015
Control Center 1.0.2	20 Apr 2015
Control Center 1.0.1	03 Apr 2015
Control Center 1.0.0	24 Feb 2015

Upgrade paths included in this document

Release 1.3.x and 1.4.0 upgrades

From	To
Control Center 1.4.0	Control Center 1.4.1
Control Center 1.3.3	Control Center 1.4.1
Control Center 1.3.2	Control Center 1.4.1
Control Center 1.3.1	Control Center 1.4.1

From	To
Control Center 1.3.0	Control Center 1.4.1

Release 1.2.x upgrades

From	To
Control Center 1.2.3	Control Center 1.4.1
Control Center 1.2.2	Control Center 1.4.1
Control Center 1.2.1	Control Center 1.4.1
Control Center 1.2.0	Control Center 1.4.1

Release 1.1.x upgrades

From	To
Control Center 1.1.10	Control Center 1.4.1
Control Center 1.1.9	Control Center 1.4.1
Control Center 1.1.8	Control Center 1.4.1
Control Center 1.1.7	Control Center 1.4.1
Control Center 1.1.5	Control Center 1.4.1
Control Center 1.1.4	Control Center 1.4.1
Control Center 1.1.3	Control Center 1.4.1
Control Center 1.1.2	Control Center 1.4.1
Control Center 1.1.1	Control Center 1.4.1

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 upgrade.

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

Procedure	Where to perform
Downloading required files on page 13	A workstation with internet access
Installing the repository mirror on page 15	All Control Center cluster hosts
Staging Docker image files on the master host on page 15	The master host
Staging a Docker image file on ZooKeeper ensemble nodes on page 16	Delegate hosts that are ZooKeeper ensemble nodes

Downloading required files

Perform one of the procedures in this section.

Downloading Control Center files (Zenoss Resource Manager users)

To perform this procedure, you need:

- A workstation with internet access.
- Permission to download files from the *File Portal - Download Zenoss Enterprise Software* site. Zenoss 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 *File Portal - Download Zenoss Enterprise Software* site.
2. Log in with the account provided by Zenoss Support.
3. Download the self-installing Docker image files.

```
install-zenoss-serviced-isvcs-v60.run
install-zenoss-isvcs-zookeeper-v10.run
```

4 Download the Control Center RPM file.

```
serviced-1.4.1-1.x86_64.rpm
```

5 Download a RHEL/CentOS repository mirror file.

The download site provides a repository mirror file for each supported release of RHEL/CentOS. Each file contains the Control Center dependencies.

To download the correct repository mirror file, match the operating system release number in the file name with the version of RHEL/CentOS installed on all of the hosts in your Control Center cluster.

```
yum-mirror-centos7.centos7.1-1503-serviced-1.4.1.x86_64.rpm
yum-mirror-centos7.centos7.2-1511-serviced-1.4.1.x86_64.rpm
yum-mirror-centos7.centos7.3-1611-serviced-1.4.1.x86_64.rpm
```

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

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

Downloading Control Center files (Zenoss Core users)

To perform this procedure, you need:

- A workstation with internet access.
- An account on the *Zenoss Community* 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 *Zenoss Community* site, and then log in with your account.
2 Download the self-installing Docker image files.

```
install-zenoss-serviced-isvcs-v60.run
install-zenoss-isvcs-zookeeper-v10.run
```

3 Download the Control Center RPM file.

```
serviced-1.4.1-1.x86_64.rpm
```

4 Download a RHEL/CentOS repository mirror file.

The download site provides a repository mirror file for each supported release of RHEL/CentOS. Each file contains the Control Center dependencies.

To download the correct repository mirror file, match the operating system release number in the file name with the version of RHEL/CentOS installed on all of the hosts in your Control Center cluster.

```
yum-mirror-centos7.centos7.1-1503-serviced-1.4.1.x86_64.rpm
yum-mirror-centos7.centos7.2-1511-serviced-1.4.1.x86_64.rpm
yum-mirror-centos7.centos7.3-1611-serviced-1.4.1.x86_64.rpm
```

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

- Copy all files to the master host.
- Copy the RHEL/CentOS RPM file and the Control Center RPM 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 cluster hosts.

1 Log in to the target host as `root`, or as a user with superuser privileges.
2 Move the RPM files to `/tmp`.
3 Optional: Remove the existing repository mirror, if necessary.
   a Search for the mirror.

```
yum list --disablerepo=* | awk '/^yum-mirror/ { print $1}'
```

   b Remove the mirror.
   Replace *Old-Mirror* with the name of the Zenoss repository mirror returned in the previous substep:

```
yum remove Old-Mirror
```

4 Install the new repository mirror.

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

The `yum` command copies the contents of the RPM file to `/opt/zenoss-repo-mirror`.
5 Move the Control Center RPM file to the mirror directory.

```
mv /tmp/serviced-1.4.1-1.x86_64.rpm \
  /opt/zenoss-repo-mirror
```

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

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

Staging Docker image files on the master host

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 add Docker image files to the Control Center master 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. The files are used when the ZooKeeper ensemble is configured.

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
```

Part I: Upgrading 1.3.x or 1.4.0 to 1.4.1

The chapters in this part provide instructions for upgrading Control Center from version 1.3.x or 1.4.0 to 1.4.1.

The following table identifies the upgrades of Control Center that are documented in this part.

From	To
Control Center 1.4.0	Control Center 1.4.1
Control Center 1.3.3	Control Center 1.4.1
Control Center 1.3.2	Control Center 1.4.1
Control Center 1.3.1	Control Center 1.4.1
Control Center 1.3.0	Control Center 1.4.1

Before upgrading from 1.3.x or 1.4.0 to 1.4.1 3

This chapter provides information and procedures to prepare a Control Center deployment for an upgrade from 1.3.x or 1.4.0 to 1.4.1.

New features that affect upgrades

This release includes new features and new requirements that affect the upgrade process. The following list provides an overview of the changes that are addressed during this upgrade.

- The upgrade (and install) process now requires downloading packages from Zenoss manually. The required software and images are no longer available from the online Zenoss repository or Docker Hub. For more information, see *Downloading and staging required files* on page 13.

  The packages to download include a `yum` repository mirror that contains the required dependencies of Docker CE and Control Center. An operating system or kernel upgrade can install newer versions of the dependencies included in the mirror, and when you attempt to upgrade Docker CE or Control Center, `yum` will stop when it finds the newer versions. A new appendix in this document includes workarounds for the most common dependency conflicts.
- This release includes a new feature, `serviced` and application audit logging. **By default, audit logging requires 10GB of storage on the master host.** The upgrade process includes a procedure for adding space to the master host, if necessary. For more information about audit logging, refer to the *Control Center Reference Guide*.
- A new configuration variable, *SERVICED_LOG_PATH*, sets the location for audit logs. The default location is `/var/log/serviced`.
- This release replaces Docker 1.12.1 with Docker Community Edition (CE) 17.03.1 in Control Center 1.3.0, 1.3.1, and 1.3.2.
- The following new configuration variables are available in `/etc/default/serviced`, for tuning TCP/IP communications between ZooKeeper ensemble hosts and Control Center:

  SERVICED_ZK_CONNECT_TIMEOUT
  SERVICED_ZK_PER_HOST_CONNECT_DELAY
  SERVICED_ZK_RECONNECT_START_DELAY
  SERVICED_ZK_RECONNECT_MAX_DELAY

  For more information, see *Control Center configuration file* on page 118.
- The Control Center RPM package includes a script that adds the `serviced` user group to a host, if necessary. Users must be members of the `serviced` group to use the command-line interface. For more information, see *User access control* on page 98.

For more information about this release, refer to the *Control Center Release Notes*.

Upgrade best practices

The following list outlines recommended best practices for upgrading Control Center deployments:

1. Download a copy of the *Control Center Release Notes* for this release and review its contents. The latest information is included in that document.
2. Compare the Docker images that accompany this release with the images that accompany the installed release, and determine whether the image files need to be downloaded and installed. For more information, see *Releases and image tags* on page 107.
3. On delegate hosts, most of the upgrade steps are identical. Use `screen`, `tmux` or a similar program to establish sessions on each delegate host and perform the steps at the same time.
4. Review and verify the settings in delegate host configuration files (`/etc/default/serviced`) before starting the upgrade. Ideally, the settings on all delegate hosts are identical, except on ZooKeeper nodes and delegate hosts that do not mount the DFS.
5. Review the procedures in this guide before performing them. Every effort is made to avoid mistakes and anticipate needs; nevertheless, the instructions may be incorrect or inadequate for some requirements or environments.
6. Download and stage the required files for your upgrade. For more information, see *Downloading and staging required files* on page 13.

Stopping a Control Center deployment 4

This chapter includes procedures for stopping both single-host and multi-host Control Center deployments.

Note The procedures in this chapter assume that Control Center is the only source of Docker containers that are run on Control Center cluster hosts.

Stopping Control Center (single-host deployment)

Use this procedure to stop the Control Center service (`serviced`) in a single-host deployment.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. 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.
3. Stop the Control Center service.

   ```
   systemctl stop serviced
   ```

4. 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, stop. This procedure is complete.
- 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
```

c Display the identifiers of all containers, running and exited.

```
docker ps -qa
```

- If the command returns no result, stop. This procedure is complete.
- If the command returns a result, perform the remaining substeps.

d Disable the automatic startup of `serviced`.

```
systemctl disable serviced
```

e Reboot the host.

```
reboot
```

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

g Enable the automatic startup of `serviced`.

```
systemctl enable serviced
```

Stopping Control Center (multi-host deployment)

To stop Control Center in a multi-host deployment, perform the procedures in this section, in order.

Stopping a master host (multi-host deployment)

Use this procedure to stop the Control Center service (`serviced`) on the master host in a multi-host deployment.

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

3 Stop the Control Center service.

```
systemctl stop serviced
```

4 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, stop. This procedure is complete.
- 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
```

c Display the identifiers of all containers, running and exited.

```
docker ps -qa
```

- If the command returns no result, stop. This procedure is complete.
- If the command returns a result, perform the remaining substeps.

d Disable the automatic startup of `serviced`.

```
systemctl disable serviced
```

e Reboot the host.

```
reboot
```

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

g Enable the automatic startup of `serviced`.

```
systemctl enable serviced
```

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
```

Upgrading a master host from 1.3.x or 1.4.0 to 1.4.1

5

Use this chapter to upgrade a Control Center 1.3.x or 1.4.0 master host to 1.4.1. Before performing the procedures in this chapter, download required software. See *Downloading and staging required files* on page 13.

Identifying storage for audit logging

The default configuration of Control Center audit logging requires 10GB of storage in `/var/log/serviced` on the master host. Use this procedure to determine whether sufficient space is available.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Display the amount of space available in `/var/log`.

   The `/var/log/serviced` directory does not exist until Control Center is upgraded.

   ```
   df -h /var/log
   ```

   Typically, `/var/log` is mounted on the root filesystem, `/`.
3. Display the amount of space available in `/tmp`.

   Docker requires 10GB of storage for temporary files, and the *Control Center Installation Guide* includes instructions to link the Docker temporary directory to `/tmp`.

   ```
   df -h /tmp
   ```

   Like `/var/log`, `/tmp` is typically mounted on the root filesystem, `/`.

If `/var/log` and `/tmp` each have 10GB of available storage, for a combined total of 20GB, then the master host has sufficient space for audit logging. Otherwise, you must choose one of the following alternatives:

- After upgrading Control Center, mount the `serviced` audit log directory on a larger local or remote file system, modify the settings in the `/opt/serviced/etc/logrotate.conf` configuration file.
- Use a `cron` job to copy the files to a larger local or remote file system.
- Forward the log files to a log management application.

Updating to Docker CE

Use this procedure to update Docker to Docker Community Edition (Docker CE) version 17.03.1.

Note If you are upgrading Control Center 1.3.3 or 1.4.0, skip this procedure. Docker CE is already installed.

1. Log in as `root`, or as a user with superuser privileges.
2. Remove the Docker repository description file, if present.
   Upgrades no longer require external repositories.

   ```
   rm -f /etc/yum.repos.d/docker.repo
   ```
3. 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 substeps.
   b. Disable automatic start of `serviced`.

      ```
      systemctl disable serviced
      ```
   c. 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
      ```
   d. Log in as `root`, or as a user with superuser privileges.
   e. Enable automatic start of `serviced`.

      ```
      systemctl enable serviced
      ```
4. Stop the Docker service.

   ```
   systemctl stop docker
   ```
5. Remove Docker 1.12.1.
   a. Remove without checking dependencies.

      ```
      rpm -e --nodeps docker-engine-1.12.1
      ```
   b. Clean the `yum` databases.

      ```
      yum clean all
      ```
6. Install Docker CE 17.03.1.

   ```
   yum install --enablerepo=zenoss-mirror docker-ce-17.03.1.ce
   ```
   If `yum` returns an error due to dependency issues, see *Resolving package dependency conflicts* on page 104 for potential resolutions.
7. Start the Docker service.

   ```
   systemctl start docker
   ```

Loading image files

Perform the steps in *Downloading and staging required files* on page 13 before performing this procedure.

Use this procedure to load images into the local Docker registry.

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

```
cd /root
```

3. Extract the images.

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

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

4. List the images in the registry.

```
docker images
```

The result should show one image for each archive file.

5. Optional: Delete the archive files.

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

Updating Control Center on the master host

Use this procedure to update Control Center on the master host to version 1.4.1.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Save the current `serviced` configuration file as a reference.
   a. Rename the file.

```
mv /etc/default/serviced /etc/default/serviced-pre-1.4.1
```

   b. Set permissions to read-only.

```
chmod 0440 /etc/default/serviced-pre-1.4.1
```

3. Remove the external Zenoss repository package.

   Upgrades no longer require external repositories.

```
yum remove zenoss-repo-1-1.x86_64
```

4. Install the new version of Control Center.

```
yum install --enablerepo=zenoss-mirror \
  /opt/zenoss-repo-mirror/serviced-1.4.1-1.x86_64.rpm
```

If `yum` returns an error due to dependency issues, see *Resolving package dependency conflicts* on page 104 for potential resolutions.

5. Make a backup copy of the new configuration file.
   a. Copy the file.

```
cp /etc/default/serviced /etc/default/serviced-1.4.1-orig
```

b Set permissions to read-only.

```
chmod 0440 /etc/default/serviced-1.4.1-orig
```

6 Compare the new configuration file with the configuration file of the previous release.

a Identify the configuration files to compare.

```
ls -l /etc/default/serviced*
```

The original versions of the configuration files should end with `orig`, but you may have to compare the dates of the files.

b Compare the new and previous configuration files.
Replace *New-Version* with the name of the new configuration file, and replace *Previous-Version* with the name of the previous configuration file:

```
diff New-Version Previous-Version
```

For example, to compare versions 1.2.0 and 1.4.1, enter the following command:

```
diff /etc/default/serviced-1.2.0-orig \
  /etc/default/serviced-1.4.1-orig
```

- If the command returns no result, restore the backup of the previous configuration file.

```
cp /etc/default/serviced-pre-1.4.1 \
  /etc/default/serviced && chmod 0644 /etc/default/serviced
```

- If the command returns a result, restore the backup of the previous configuration file, and then optionally, use the results to edit the restored version.

```
cp /etc/default/serviced-pre-1.4.1 \
  /etc/default/serviced && chmod 0644 /etc/default/serviced
```

For more information about configuring the master host, see *Control Center configuration variables* on page 109.

7 Update the log file management configuration file, if necessary.

The default configuration of the `/opt/serviced/etc/logrotate.conf` file now uses file size rather than file age to determine when to rotate logs. If you customized `/opt/serviced/etc/logrotate.conf` for your environment, your customizations are saved in `/opt/serviced/etc/logrotate.conf.bak`. Compare the new and old versions, and update the new one as desired.

- If you are upgrading a multi-host deployment, continue to the next procedure.
- If you are upgrading a single-host deployment, start Control Center.

```
systemctl daemon-reload && systemctl start serviced
```

Upgrading delegates from 1.3.x or 1.4.0 to 1.4.1

6

Use this chapter to upgrade Control Center 1.3.x or 1.4.0 delegate hosts to 1.4.1. Before performing the procedures in this chapter, download required software. See *Downloading and staging required files* on page 13.

Updating to Docker CE

Use this procedure to update Docker to Docker Community Edition (Docker CE) version 17.03.1.

Note If you are upgrading Control Center 1.3.3 or 1.4.0, skip this procedure. Docker CE is already installed.

1. Log in as `root`, or as a user with superuser privileges.
2. Remove the Docker repository description file, if present.

   Upgrades no longer require external repositories.

   ```
   rm -f /etc/yum.repos.d/docker.repo
   ```
3. 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 substeps.
   b. Disable automatic start of `serviced`.

      ```
      systemctl disable serviced
      ```
   c. 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
      ```
   d. Log in as `root`, or as a user with superuser privileges.
   e. Enable automatic start of `serviced`.

      ```
      systemctl enable serviced
      ```

4 Stop the Docker service.

```
systemctl stop docker
```

5 Remove Docker 1.12.1.

a Remove without checking dependencies.

```
rpm -e --nodeps docker-engine-1.12.1
```

b Clean the `yum` databases.

```
yum clean all
```

6 Install Docker CE 17.03.1.

```
yum install --enablerepo=zenoss-mirror docker-ce-17.03.1.ce
```

If `yum` returns an error due to dependency issues, see *Resolving package dependency conflicts* on page 104 for potential resolutions.

7 Start the Docker service.

```
systemctl start docker
```

Updating Control Center on delegate hosts

This procedure updates Control Center on delegate hosts to version 1.4.1.

Perform this procedure on each delegate host in your deployment.

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

2 Save the current `serviced` configuration file as a reference.

a Rename the file.

```
mv /etc/default/serviced /etc/default/serviced-pre-1.4.1
```

b Set permissions to read-only.

```
chmod 0440 /etc/default/serviced-pre-1.4.1
```

3 Remove the external Zenoss repository package.

Upgrades no longer require external repositories.

```
yum remove zenoss-repo-1-1.x86_64
```

4 Install the new version of Control Center.

```
yum install --enablerepo=zenoss-mirror \
  /opt/zenoss-repo-mirror/serviced-1.4.1-1.x86_64.rpm
```

If `yum` returns an error due to dependency issues, see *Resolving package dependency conflicts* on page 104 for potential resolutions.

5 Make a backup copy of the new configuration file.

a Copy the file.

```
cp /etc/default/serviced /etc/default/serviced-1.4.1-orig
```

b Set permissions to read-only.

```
chmod 0440 /etc/default/serviced-1.4.1-orig
```

6 Compare the new configuration file with the configuration file of the previous release.

a Identify the configuration files to compare.

```
ls -l /etc/default/serviced*
```

The original versions of the configuration files should end with `orig`, but you may have to compare the dates of the files.

b Compare the new and previous configuration files.
Replace *New-Version* with the name of the new configuration file, and replace *Previous-Version* with the name of the previous configuration file:

```
diff New-Version Previous-Version
```

For example, to compare versions 1.2.0 and 1.4.1, enter the following command:

```
diff /etc/default/serviced-1.2.0-orig \
  /etc/default/serviced-1.4.1-orig
```

- If the command returns no result, restore the backup of the previous configuration file.

```
cp /etc/default/serviced-pre-1.4.1 \
  /etc/default/serviced && chmod 0644 /etc/default/serviced
```

- If the command returns a result, restore the backup of the previous configuration file, and then optionally, use the results to edit the restored version.

```
cp /etc/default/serviced-pre-1.4.1 \
  /etc/default/serviced && chmod 0644 /etc/default/serviced
```

For more information about configuring a delegate host, see *Control Center configuration variables* on page 109.

Updating the ZooKeeper image on ensemble nodes

Perform the steps in *Downloading and staging required files* on page 13 before performing this procedure.

Use this procedure to install a new Docker image for ZooKeeper on ZooKeeper ensemble nodes.

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

2 Identify the hosts in the ZooKeeper ensemble.

```
grep -E '^\b*SERVICED_ZK=' /etc/default/serviced
```

The result is a list of 3 or 5 hosts, separated by the comma character (`,`). The master host is always a node in the ZooKeeper ensemble.

3 Log in to a ZooKeeper ensemble node as `root`, or as a user with superuser privileges.

4 Change directory to `/root`.

```
cd /root
```

5 Extract the ZooKeeper image.

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

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

6 Optional: Delete the archive file.

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

7 Repeat the preceding four steps on each delegate that is a node in the ZooKeeper ensemble.

7 Starting the ZooKeeper ensemble

Use this procedure to start the ZooKeeper ensemble.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Determine whether `serviced` is configured to start when the system starts.

```
systemctl is-enabled serviced
```

   - If the result is `enabled`, proceed to the next step.
   - If the result is `disabled`, enter the following command:

```
systemctl enable serviced
```

3. Identify the hosts in the ZooKeeper ensemble.

```
grep -E '^\b*SERVICED_ZK=' /etc/default/serviced
```

   The result is a list of 3 or 5 hosts, separated by the comma character (`,`). The master host is always a node in the ZooKeeper ensemble.
4. 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.
5. On all 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 daemon-reload && systemctl start serviced
```

6. 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 /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 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 After upgrading from 1.3.x or 1.4.0 to 1.4.1

Perform the procedures in this chapter after Control Center is upgraded.

Removing unused images

Use this procedure to identify and remove unused Control Center images.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Identify the images associated with the installed version of `serviced`.

   ```
   serviced version | grep Images
   ```

   Example result:

   ```
   IsvcsImages: [zenoss/serviced-isvcs:v60 zenoss/isvcs-zookeeper:v10]
   ```

3. Display the `serviced` images in the local repository.

   ```
   docker images | awk '/REPO|isvcs/'
   ```

   Example result (edited to fit):

   ```
   REPOSITORY                      TAG       IMAGE ID
   zenoss/serviced-isvcs           v40       88cd6c24cc82
   zenoss/serviced-isvcs           v60       0aab5a2123f2
   zenoss/isvcs-zookeeper          v3        46fa0a2fc4bf
   zenoss/isvcs-zookeeper          v10       0ff3b3117fb8
   ```

   The example result shows the current versions and one set of previous versions. Your result may include additional previous versions and will show different images IDs.
4. Remove unused images.
   Replace *Image-ID* with the image ID of an image for a previous version.

   ```
   docker rmi Image-ID
   ```

   Repeat this command for each unused image.

Part II: Upgrading 1.2.x to 1.4.1

The chapters in this part provide instructions for upgrading Control Center from version 1.2.x to 1.4.1.

The following table identifies the upgrades of Control Center that are documented in this part.

From	To
Control Center 1.2.3	Control Center 1.4.1
Control Center 1.2.2	Control Center 1.4.1
Control Center 1.2.1	Control Center 1.4.1
Control Center 1.2.0	Control Center 1.4.1

9 Before upgrading from 1.2.x to 1.4.1

This chapter provides information and procedures to prepare a Control Center deployment for an upgrade from 1.2.x to 1.4.1.

New features that affect upgrades from release 1.2.x

This release includes new features and new requirements that affect the upgrade process. The following list provides an overview of the changes that are addressed during this upgrade.

- The upgrade (and install) process now requires downloading packages from Zenoss manually. The required software and images are no longer available from the online Zenoss repository or Docker Hub. For more information, see *Downloading and staging required files* on page 13.

  The packages to download include a `yum` repository mirror that contains the required dependencies of Docker CE and Control Center. An operating system or kernel upgrade can install newer versions of the dependencies included in the mirror, and when you attempt to upgrade Docker CE or Control Center, `yum` will stop when it finds the newer versions. A new appendix in this document includes workarounds for the most common dependency conflicts.
- This release includes a new feature, `serviced` and application audit logging. **By default, audit logging requires 10GB of storage on the master host.** The upgrade process includes a procedure for adding space to the master host, if necessary. For more information about audit logging, refer to the *Control Center Reference Guide*.
- A new configuration variable, *SERVICED_LOG_PATH*, sets the location for audit logs. The default location is `/var/log/serviced`.
- When storage becomes critically low, Control Center initiates an *emergency shutdown* of applications and services while sufficient resources remain to take action to avoid data loss. Support for this feature requires specific minimum amounts of space in the Control Center thin pool for application data.

  Note Before upgrading, determine whether the thin pool is adequate and if necessary, resize it. For more information, see *Control Center application data storage requirements* on page 90.
- This release replaces Docker 1.12.1 with Docker Community Edition (CE) 17.03.1.
- The following new configuration variables are available in `/etc/default/serviced`, for tuning TCP/IP communications between ZooKeeper ensemble hosts and Control Center:

  SERVICED_ZK_CONNECT_TIMEOUT
  SERVICED_ZK_PER_HOST_CONNECT_DELAY
  SERVICED_ZK_RECONNECT_START_DELAY

SERVICED_ZK_RECONNECT_MAX_DELAY

For more information, see *Control Center configuration file* on page 118.

- The Control Center RPM package includes a script that adds the `serviced` user group to a host, if necessary. Users must be members of the `serviced` group to use the command-line interface. For more information, see *User access control* on page 98.

For more information about this release, refer to the *Control Center Release Notes*.

Upgrade best practices

The following list outlines recommended best practices for upgrading Control Center deployments:

1. Download a copy of the *Control Center Release Notes* for this release and review its contents. The latest information is included in that document.
2. Compare the Docker images that accompany this release with the images that accompany the installed release, and determine whether the image files need to be downloaded and installed. For more information, see *Releases and image tags* on page 107.
3. On delegate hosts, most of the upgrade steps are identical. Use `screen`, `tmux` or a similar program to establish sessions on each delegate host and perform the steps at the same time.
4. Review and verify the settings in delegate host configuration files (`/etc/default/serviced`) before starting the upgrade. Ideally, the settings on all delegate hosts are identical, except on ZooKeeper nodes and delegate hosts that do not mount the DFS.
5. Review the procedures in this guide before performing them. Every effort is made to avoid mistakes and anticipate needs; nevertheless, the instructions may be incorrect or inadequate for some requirements or environments.
6. Download and stage the required files for your upgrade. For more information, see *Downloading and staging required files* on page 13.

10 Stopping a Control Center deployment

This chapter includes procedures for stopping both single-host and multi-host Control Center deployments.

Note The procedures in this chapter assume that Control Center is the only source of Docker containers that are run on Control Center cluster hosts.

Stopping Control Center (single-host deployment)

Use this procedure to stop the Control Center service (`serviced`) in a single-host deployment.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. 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.
3. Stop the Control Center service.

   ```
   systemctl stop serviced
   ```

4. 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, stop. This procedure is complete.
- 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
```

c Display the identifiers of all containers, running and exited.

```
docker ps -qa
```

- If the command returns no result, stop. This procedure is complete.
- If the command returns a result, perform the remaining substeps.

d Disable the automatic startup of `serviced`.

```
systemctl disable serviced
```

e Reboot the host.

```
reboot
```

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

g Enable the automatic startup of `serviced`.

```
systemctl enable serviced
```

Stopping Control Center (multi-host deployment)

To stop Control Center in a multi-host deployment, perform the procedures in this section, in order.

Stopping a master host (multi-host deployment)

Use this procedure to stop the Control Center service (`serviced`) on the master host in a multi-host deployment.

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

3 Stop the Control Center service.

```
systemctl stop serviced
```

4 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, stop. This procedure is complete.
- 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
```

c Display the identifiers of all containers, running and exited.

```
docker ps -qa
```

- If the command returns no result, stop. This procedure is complete.
- If the command returns a result, perform the remaining substeps.

d Disable the automatic startup of `serviced`.

```
systemctl disable serviced
```

e Reboot the host.

```
reboot
```

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

g Enable the automatic startup of `serviced`.

```
systemctl enable serviced
```

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
```

Upgrading a master host from 1.2.x to 1.4.1 11

Use this chapter to upgrade a Control Center 1.2.x master host to 1.4.1. Before performing the procedures in this chapter, download required software. See *Downloading and staging required files* on page 13.

Identifying storage for audit logging

The default configuration of Control Center audit logging requires 10GB of storage in `/var/log/serviced` on the master host. Use this procedure to determine whether sufficient space is available.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Display the amount of space available in `/var/log`.

   The `/var/log/serviced` directory does not exist until Control Center is upgraded.

   ```
   df -h /var/log
   ```

   Typically, `/var/log` is mounted on the root filesystem, `/`.
3. Display the amount of space available in `/tmp`.

   Docker requires 10GB of storage for temporary files, and the *Control Center Installation Guide* includes instructions to link the Docker temporary directory to `/tmp`.

   ```
   df -h /tmp
   ```

   Like `/var/log`, `/tmp` is typically mounted on the root filesystem, `/`.

If `/var/log` and `/tmp` each have 10GB of available storage, for a combined total of 20GB, then the master host has sufficient space for audit logging. Otherwise, you must choose one of the following alternatives:

- After upgrading Control Center, mount the `serviced` audit log directory on a larger local or remote file system, modify the settings in the `/opt/serviced/etc/logrotate.conf` configuration file.
- Use a `cron` job to copy the files to a larger local or remote file system.
- Forward the log files to a log management application.

Updating to Docker CE

Use this procedure to update Docker to Docker Community Edition (Docker CE) version 17.03.1.

1. Log in as `root`, or as a user with superuser privileges.
2. Remove the Docker repository description file.

Upgrades no longer require external repositories.

```
rm -f /etc/yum.repos.d/docker.repo
```

3 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 substeps.

b Disable automatic start of `serviced`.

```
systemctl disable serviced
```

c 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
```

d Log in as `root`, or as a user with superuser privileges.

e Enable automatic start of `serviced`.

```
systemctl enable serviced
```

4 Stop the Docker service.

```
systemctl stop docker
```

5 Remove Docker 1.12.1.

a Remove without checking dependencies.

```
rpm -e --nodeps docker-engine-1.12.1
```

b Clean the `yum` databases.

```
yum clean all
```

6 Install Docker CE 17.03.1.

```
yum install --enablerepo=zenoss-mirror docker-ce-17.03.1.ce
```

If `yum` returns an error due to dependency issues, see *Resolving package dependency conflicts* on page 104 for potential resolutions.

7 Start the Docker service.

```
systemctl start docker
```

Loading image files

Perform the steps in *Downloading and staging required files* on page 13 before performing this procedure.

Use this procedure to load images into the local Docker registry.

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

2 Change directory to `/root`.

```
cd /root
```

3 Extract the images.

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

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

4 List the images in the registry.

```
docker images
```

The result should show one image for each archive file.

5 Optional: Delete the archive files.

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

Updating Control Center on the master host

Use this procedure to update Control Center on the master host to version 1.4.1.

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

2 Save the current `serviced` configuration file as a reference.

a Rename the file.

```
mv /etc/default/serviced /etc/default/serviced-pre-1.4.1
```

b Set permissions to read-only.

```
chmod 0440 /etc/default/serviced-pre-1.4.1
```

3 Remove the external Zenoss repository package.

Upgrades no longer require external repositories.

```
yum remove zenoss-repo-1-1.x86_64
```

4 Install the new version of Control Center.

```
yum install --enablerepo=zenoss-mirror \
  /opt/zenoss-repo-mirror/serviced-1.4.1-1.x86_64.rpm
```

If `yum` returns an error due to dependency issues, see *Resolving package dependency conflicts* on page 104 for potential resolutions.

5 Make a backup copy of the new configuration file.

a Copy the file.

```
cp /etc/default/serviced /etc/default/serviced-1.4.1-orig
```

b Set permissions to read-only.

```
chmod 0440 /etc/default/serviced-1.4.1-orig
```

6 Compare the new configuration file with the configuration file of the previous release.

a Identify the configuration files to compare.

```
ls -l /etc/default/serviced*
```

The original versions of the configuration files should end with `orig`, but you may have to compare the dates of the files.

b Compare the new and previous configuration files.
Replace *New-Version* with the name of the new configuration file, and replace *Previous-Version* with the name of the previous configuration file:

```
diff New-Version Previous-Version
```

For example, to compare versions 1.2.0 and 1.4.1, enter the following command:

```
diff /etc/default/serviced-1.2.0-orig \
  /etc/default/serviced-1.4.1-orig
```

- If the command returns no result, restore the backup of the previous configuration file.

```
cp /etc/default/serviced-pre-1.4.1 \
  /etc/default/serviced && chmod 0644 /etc/default/serviced
```

- If the command returns a result, restore the backup of the previous configuration file, and then optionally, use the results to edit the restored version.

```
cp /etc/default/serviced-pre-1.4.1 \
  /etc/default/serviced && chmod 0644 /etc/default/serviced
```

For more information about configuring the master host, see *Control Center configuration variables* on page 109.

7 Update the log file management configuration file, if necessary.

The default configuration of the `/opt/serviced/etc/logrotate.conf` file now uses file size rather than file age to determine when to rotate logs. If you customized `/opt/serviced/etc/logrotate.conf` for your environment, your customizations are saved in `/opt/serviced/etc/logrotate.conf.bak`. Compare the new and old versions, and update the new one as desired.

- If you are upgrading a multi-host deployment, continue to the next procedure.
- If you are upgrading a single-host deployment, start Control Center.

```
systemctl daemon-reload && systemctl start serviced
```

12 Upgrading delegates from 1.2.x to 1.4.1

Use this chapter to upgrade Control Center 1.2.x delegate hosts to 1.4.1. Before performing the procedures in this chapter, download required software. See *Downloading and staging required files* on page 13.

Updating to Docker CE

Use this procedure to update Docker to Docker Community Edition (Docker CE) version 17.03.1.

1. Log in as `root`, or as a user with superuser privileges.
2. Remove the Docker repository description file.

   Upgrades no longer require external repositories.

   ```
   rm -f /etc/yum.repos.d/docker.repo
   ```
3. 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 substeps.
   b. Disable automatic start of `serviced`.

      ```
      systemctl disable serviced
      ```
   c. 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
      ```
   d. Log in as `root`, or as a user with superuser privileges.
   e. Enable automatic start of `serviced`.

      ```
      systemctl enable serviced
      ```
4. Stop the Docker service.

   ```
   systemctl stop docker
   ```
5. Remove Docker 1.12.1.

a Remove without checking dependencies.

```
rpm -e --nodeps docker-engine-1.12.1
```

b Clean the `yum` databases.

```
yum clean all
```

6 Install Docker CE 17.03.1.

```
yum install --enablerepo=zenoss-mirror docker-ce-17.03.1.ce
```

If `yum` returns an error due to dependency issues, see *Resolving package dependency conflicts* on page 104 for potential resolutions.

7 Start the Docker service.

```
systemctl start docker
```

Updating Control Center on delegate hosts

This procedure updates Control Center on delegate hosts to version 1.4.1.

Perform this procedure on each delegate host in your deployment.

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

2 Save the current `serviced` configuration file as a reference.

a Rename the file.

```
mv /etc/default/serviced /etc/default/serviced-pre-1.4.1
```

b Set permissions to read-only.

```
chmod 0440 /etc/default/serviced-pre-1.4.1
```

3 Remove the external Zenoss repository package.

Upgrades no longer require external repositories.

```
yum remove zenoss-repo-1-1.x86_64
```

4 Install the new version of Control Center.

```
yum install --enablerepo=zenoss-mirror \
  /opt/zenoss-repo-mirror/serviced-1.4.1-1.x86_64.rpm
```

If `yum` returns an error due to dependency issues, see *Resolving package dependency conflicts* on page 104 for potential resolutions.

5 Make a backup copy of the new configuration file.

a Copy the file.

```
cp /etc/default/serviced /etc/default/serviced-1.4.1-orig
```

b Set permissions to read-only.

```
chmod 0440 /etc/default/serviced-1.4.1-orig
```

6 Compare the new configuration file with the configuration file of the previous release.

a Identify the configuration files to compare.

```
ls -l /etc/default/serviced*
```

The original versions of the configuration files should end with `orig`, but you may have to compare the dates of the files.

b Compare the new and previous configuration files.
Replace *New-Version* with the name of the new configuration file, and replace *Previous-Version* with the name of the previous configuration file:

```
diff New-Version Previous-Version
```

For example, to compare versions 1.2.0 and 1.4.1, enter the following command:

```
diff /etc/default/serviced-1.2.0-orig \
  /etc/default/serviced-1.4.1-orig
```

- If the command returns no result, restore the backup of the previous configuration file.

```
cp /etc/default/serviced-pre-1.4.1 \
  /etc/default/serviced && chmod 0644 /etc/default/serviced
```

- If the command returns a result, restore the backup of the previous configuration file, and then optionally, use the results to edit the restored version.

```
cp /etc/default/serviced-pre-1.4.1 \
  /etc/default/serviced && chmod 0644 /etc/default/serviced
```

For more information about configuring a delegate host, see *Control Center configuration variables* on page 109.

Updating the ZooKeeper image on ensemble nodes

Perform the steps in *Downloading and staging required files* on page 13 before performing this procedure.

Use this procedure to install a new Docker image for ZooKeeper on ZooKeeper ensemble nodes.

1 Log in to the master host as `root`, or as a user with superuser privileges.
2 Identify the hosts in the ZooKeeper ensemble.

```
grep -E '^\b*SERVICED_ZK=' /etc/default/serviced
```

The result is a list of 3 or 5 hosts, separated by the comma character (`,`). The master host is always a node in the ZooKeeper ensemble.

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

```
cd /root
```

5 Extract the ZooKeeper image.

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

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

6 Optional: Delete the archive file.

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

7 Repeat the preceding four steps on each delegate that is a node in the ZooKeeper ensemble.

13 Starting the ZooKeeper ensemble

Use this procedure to start the ZooKeeper ensemble.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Determine whether `serviced` is configured to start when the system starts.

```
systemctl is-enabled serviced
```

   - If the result is `enabled`, proceed to the next step.
   - If the result is `disabled`, enter the following command:

```
systemctl enable serviced
```

3. Identify the hosts in the ZooKeeper ensemble.

```
grep -E '^\b*SERVICED_ZK=' /etc/default/serviced
```

   The result is a list of 3 or 5 hosts, separated by the comma character (`,`). The master host is always a node in the ZooKeeper ensemble.
4. 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.
5. On all 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 daemon-reload && systemctl start serviced
```

6. 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 /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 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.

14 After upgrading from 1.2.x to 1.4.1

Perform the procedures in this chapter after Control Center is upgraded.

Removing unused images

Use this procedure to identify and remove unused Control Center images.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Identify the images associated with the installed version of `serviced`.

   ```
   serviced version | grep Images
   ```

   Example result:

   ```
   IsvcsImages: [zenoss/serviced-isvcs:v60 zenoss/isvcs-zookeeper:v10]
   ```

3. Display the `serviced` images in the local repository.

   ```
   docker images | awk '/REPO|isvcs/'
   ```

   Example result (edited to fit):

   ```
   REPOSITORY                      TAG       IMAGE ID
   zenoss/serviced-isvcs           v40       88cd6c24cc82
   zenoss/serviced-isvcs           v60       0aab5a2123f2
   zenoss/isvcs-zookeeper          v3        46fa0a2fc4bf
   zenoss/isvcs-zookeeper          v10       0ff3b3117fb8
   ```

   The example result shows the current versions and one set of previous versions. Your result may include additional previous versions and will show different images IDs.
4. Remove unused images.
   Replace *Image-ID* with the image ID of an image for a previous version.

   ```
   docker rmi Image-ID
   ```

   Repeat this command for each unused image.

Part III: Upgrading 1.1.x to 1.4.1

The chapters in this part provide instructions for upgrading Control Center from version 1.1.x to 1.4.1.

The following table identifies the upgrades of Control Center that are documented in this part.

From	To
Control Center 1.1.10	Control Center 1.4.1
Control Center 1.1.9	Control Center 1.4.1
Control Center 1.1.8	Control Center 1.4.1
Control Center 1.1.7	Control Center 1.4.1
Control Center 1.1.5	Control Center 1.4.1
Control Center 1.1.4	Control Center 1.4.1
Control Center 1.1.3	Control Center 1.4.1
Control Center 1.1.2	Control Center 1.4.1
Control Center 1.1.1	Control Center 1.4.1

Before upgrading from 1.1.x to 1.4.1

15

This chapter provides information and procedures to prepare a Control Center deployment for an upgrade from 1.1.x to 1.4.1.

New features that affect upgrades from 1.1.x

This release includes new features and new requirements that affect the upgrade process. The following list provides an overview of the changes that are addressed during this upgrade.

Note Zenoss applications may require additional steps to prepare to use this release. For more information, refer to the documentation for your application.

- The upgrade (and install) process now requires downloading packages from Zenoss manually. The required software and images are no longer available from the online Zenoss repository or Docker Hub. For more information, see *Downloading and staging required files* on page 13.

  The packages to download include a `yum` repository mirror that contains the required dependencies of Docker CE and Control Center. An operating system or kernel upgrade can install newer versions of the dependencies included in the mirror, and when you attempt to upgrade Docker CE or Control Center, `yum` will stop when it finds the newer versions. A new appendix in this document includes workarounds for the most common dependency conflicts.
- This release includes a new feature, `serviced` and application audit logging. **By default, audit logging requires 10GB of storage on the master host.** The upgrade process includes a procedure for adding space to the master host, if necessary. For more information about audit logging, refer to the *Control Center Reference Guide*.
- A new configuration variable, *SERVICED_LOG_PATH*, sets the location for audit logs. The default location is `/var/log/serviced`.
- When storage becomes critically low, Control Center initiates an *emergency shutdown* of applications and services while sufficient resources remain to take action to avoid data loss. Support for this feature requires specific minimum amounts of space in the Control Center thin pool for application data.

  Note Before upgrading, determine whether the thin pool is adequate and if necessary, resize it. For more information, see *Control Center application data storage requirements* on page 90.
- This release replaces Docker 1.9.0 with Docker Community Edition (CE) 17.03.1.
- The following new configuration variables are available in `/etc/default/serviced`, for tuning TCP/IP communications between ZooKeeper ensemble hosts and Control Center:

SERVICED_ZK_CONNECT_TIMEOUT
SERVICED_ZK_PER_HOST_CONNECT_DELAY
SERVICED_ZK_RECONNECT_START_DELAY
SERVICED_ZK_RECONNECT_MAX_DELAY

For more information, see *Control Center configuration file* on page 118.

- The Control Center RPM package includes a script that adds the `serviced` user group to a host, if necessary. Users must be members of the `serviced` group to use the command-line interface. For more information, see *User access control* on page 98.
- The minimum kernel version for Control Center hosts is `3.10.0-327.36.2`. For optimal results, the most recent kernel is recommended. To prevent dependency issues, updating the kernel or operating system is a step in the Docker update procedure.
- The `serviced` configuration file includes many new variables since version 1.1.1, and some deprecated variables. For more information, see *Control Center configuration variables* on page 109.
- All delegate communications are authenticated. To enable this feature, all existing hosts must install unique credentials, which are generated on the master host. The installation steps are included in the startup procedures.
- With delegate authentication, Control Center can control administrative and DFS access permissions at the resource pool level. During the upgrade, all existing resource pools are given both administrative and DFS access permissions. The post-upgrade chapter includes an optional procedure for removing permissions from a resource pool.
- In previous releases, the *SERVICED_NFS_CLIENT* variable was set on delegate hosts to prevent access to the DFS. In this release, *SERVICED_NFS_CLIENT* is deprecated in favor of setting DFS access permission at the resource pool level. To ease the transition to the new functionality, delegate host configurations that include the *SERVICED_NFS_CLIENT* variable are still supported.
- This release includes a new resource pool feature, the ability to set the length of time the scheduler waits for a disconnected delegate host to rejoin its pool before moving the services scheduled for the delegate to a different host in the pool. This feature is useful for remote resource pools that are connected through a high-latency, wide-area network. For more information, see *Setting the connection timeout of a resource pool* on page 85.
- Among other changes since version 1.9.0, Docker 17.03.1 includes a new storage subsystem. The initial startup takes a little longer, as the old layout is replaced with the new layout.

For more information about this release, refer to the *Control Center Release Notes*.

Upgrade best practices

The following list outlines recommended best practices for upgrading Control Center deployments:

1. Download a copy of the *Control Center Release Notes* for this release and review its contents. The latest information is included in that document.
2. Compare the Docker images that accompany this release with the images that accompany the installed release, and determine whether the image files need to be downloaded and installed. For more information, see *Releases and image tags* on page 107.
3. On delegate hosts, most of the upgrade steps are identical. Use `screen`, `tmux` or a similar program to establish sessions on each delegate host and perform the steps at the same time.
4. Review and verify the settings in delegate host configuration files (`/etc/default/serviced`) before starting the upgrade. Ideally, the settings on all delegate hosts are identical, except on ZooKeeper nodes and delegate hosts that do not mount the DFS.
5. Review the procedures in this guide before performing them. Every effort is made to avoid mistakes and anticipate needs; nevertheless, the instructions may be incorrect or inadequate for some requirements or environments.

6 Download and stage the required files for your upgrade. For more information, see *Downloading and staging required files* on page 13.

16 Stopping a Control Center deployment

This chapter includes procedures for stopping both single-host and multi-host Control Center deployments.

Note The procedures in this chapter assume that Control Center is the only source of Docker containers that are run on Control Center cluster hosts.

Stopping Control Center (single-host deployment)

Use this procedure to stop the Control Center service (`serviced`) in a single-host deployment.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. 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.
3. Stop the Control Center service.

   ```
   systemctl stop serviced
   ```

4. 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, stop. This procedure is complete.
- 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
```

c Display the identifiers of all containers, running and exited.

```
docker ps -qa
```

- If the command returns no result, stop. This procedure is complete.
- If the command returns a result, perform the remaining substeps.

d Disable the automatic startup of `serviced`.

```
systemctl disable serviced
```

e Reboot the host.

```
reboot
```

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

g Enable the automatic startup of `serviced`.

```
systemctl enable serviced
```

Stopping Control Center (multi-host deployment)

To stop Control Center in a multi-host deployment, perform the procedures in this section, in order.

Stopping a master host (multi-host deployment)

Use this procedure to stop the Control Center service (`serviced`) on the master host in a multi-host deployment.

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

2 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.

3 Stop the Control Center service.

```
systemctl stop serviced
```

4 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, stop. This procedure is complete.
- 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
```

c Display the identifiers of all containers, running and exited.

```
docker ps -qa
```

- If the command returns no result, stop. This procedure is complete.
- If the command returns a result, perform the remaining substeps.

d Disable the automatic startup of `serviced`.

```
systemctl disable serviced
```

e Reboot the host.

```
reboot
```

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

g Enable the automatic startup of `serviced`.

```
systemctl enable serviced
```

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
```

Upgrading a master host from 1.1.x to 1.4.1 17

Use this chapter to upgrade a Control Center 1.1.x master host to 1.4.1. Before performing the procedures in this chapter, download required software. See *Downloading and staging required files* on page 13.

Identifying storage for audit logging

The default configuration of Control Center audit logging requires 10GB of storage in `/var/log/serviced` on the master host. Use this procedure to determine whether sufficient space is available.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Display the amount of space available in `/var/log`.

   The `/var/log/serviced` directory does not exist until Control Center is upgraded.

   ```
   df -h /var/log
   ```

   Typically, `/var/log` is mounted on the root filesystem, `/`.
3. Display the amount of space available in `/tmp`.

   Docker requires 10GB of storage for temporary files, and the *Control Center Installation Guide* includes instructions to link the Docker temporary directory to `/tmp`.

   ```
   df -h /tmp
   ```

   Like `/var/log`, `/tmp` is typically mounted on the root filesystem, `/`.

If `/var/log` and `/tmp` each have 10GB of available storage, for a combined total of 20GB, then the master host has sufficient space for audit logging. Otherwise, you must choose one of the following alternatives:

- After upgrading Control Center, mount the `serviced` audit log directory on a larger local or remote file system, modify the settings in the `/opt/serviced/etc/logrotate.conf` configuration file.
- Use a `cron` job to copy the files to a larger local or remote file system.
- Forward the log files to a log management application.

Updating to Docker CE

Use this procedure to update Docker to Docker Community Edition (Docker CE) version 17.03.1.

1. Log in as `root`, or as a user with superuser privileges.
2. Remove the Docker repository description file.

Upgrades no longer require external repositories.

```
rm -f /etc/yum.repos.d/docker.repo
```

3 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 substeps.

b Disable automatic start of `serviced`.

```
systemctl disable serviced
```

c 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
```

d Log in as `root`, or as a user with superuser privileges.

e Enable automatic start of `serviced`.

```
systemctl enable serviced
```

4 Identify the name of the LVM thin pool for Docker.

```
docker info 2>/dev/null | grep 'Pool Name'
```

Example result:

```
 Pool Name: docker-docker--pool
```

Record the name for use in a subsequent step.

5 Back up the Docker environment file.

```
test -f /etc/sysconfig/docker \
  && mv /etc/sysconfig/docker /etc/sysconfig/docker.bak
```

6 Stop the Docker service.

```
systemctl stop docker
```

7 Remove Docker 1.9.0.

a Remove without checking dependencies.

```
rpm -e --nodeps docker-engine-1.9.0
```

b Clean the `yum` databases.

```
yum clean all
```

8 Install Docker CE 17.03.1.

```
yum install --enablerepo=zenoss-mirror docker-ce-17.03.1.ce
```

If `yum` returns an error due to dependency issues, see *Resolving package dependency conflicts* on page 104 for potential resolutions.

Configuring Docker

Use this procedure to configure Docker.

1. Log in as `root`, or as a user with superuser privileges.
2. Create a new Docker drop-in file.
   a. Create a directory for the drop-in file, if necessary.

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

   b. Create a backup of the drop-in file, if it exists.

```
test -f /etc/systemd/system/docker.service.d/docker.conf \
  && cp -p /etc/systemd/system/docker.service.d/docker.conf \
  /etc/systemd/system/docker.service.d/docker.conf.bak
```

   c. Create the new 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
```

3. Reload the `systemd` manager configuration.

```
systemctl daemon-reload
```

4. 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, identified in a previous step:

```
myPool="/dev/mapper/Thin-Pool-Device"
```

   b. Create variables for adding arguments to the Docker environment 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 environment 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 as Docker updates the storage layout.

5 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
```

6 Compare the previous version of the Docker environment file with the new version, and add any customizations for your deployment to the new version.

```
diff /etc/sysconfig/docker /etc/sysconfig/docker.bak
```

If you change this file, restart Docker with `systemctl restart docker`.

Optional: Changing the local Docker registry endpoint

Use this procedure to configure the master host with the endpoint of an alternative local Docker registry. Control Center includes a local Docker registry, but you may use an existing registry in your environment, if desired. For more information about configuring a local Docker registry, please refer to *Docker documentation*.

Note Changing the local Docker registry endpoint is rare. Perform this procedure only if you are sure it is necessary and the alternative local Docker registry is already available in your environment.

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:5000` with the endpoint of the local Docker registry.
      Use the IP address or fully-qualified domain name of the host and the port number.
   e. Save the file, and then close the editor.
3. Verify the settings.

```
grep -E '^\b*SERVICED' /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 following flag 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
```

Loading image files

Perform the steps in *Downloading and staging required files* on page 13 before performing this procedure.

Use this procedure to load images into the local Docker registry.

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

```
cd /root
```

3. Extract the images.

```
for image in install-*.run
do
  echo -n "$image: "
```

```
  ./$image
done
```

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

4 List the images in the registry.

```
docker images
```

The result should show one image for each archive file.

5 Optional: Delete the archive files.

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

Updating Control Center on the master host

Use this procedure to update Control Center on the master host to version 1.4.1.

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

2 Save the current `serviced` configuration file as a reference.

a Rename the file.

```
mv /etc/default/serviced /etc/default/serviced-pre-1.4.1
```

b Set permissions to read-only.

```
chmod 0440 /etc/default/serviced-pre-1.4.1
```

3 Remove the external Zenoss repository package.

Upgrades no longer require external repositories.

```
yum remove zenoss-repo-1-1.x86_64
```

4 Install the new version of Control Center.

```
yum install --enablerepo=zenoss-mirror \
  /opt/zenoss-repo-mirror/serviced-1.4.1-1.x86_64.rpm
```

If `yum` returns an error due to dependency issues, see *Resolving package dependency conflicts* on page 104 for potential resolutions.

5 Make a backup copy of the new configuration file.

a Copy the file.

```
cp /etc/default/serviced /etc/default/serviced-1.4.1-orig
```

b Set permissions to read-only.

```
chmod 0440 /etc/default/serviced-1.4.1-orig
```

6 Display the settings of the reference configuration file.

```
grep -E '^\b*SERVICED' /etc/default/serviced-pre-1.4.1
```

7 Open the new configuration file with a text editor, and then update the file for your environment.

For more information about configuring the master host, see *Control Center configuration variables* on page 109.

8 Update the log file management configuration file, if necessary.

The default configuration of the `/opt/serviced/etc/logrotate.conf` file now uses file size rather than file age to determine when to rotate logs. If you customized `/opt/serviced/etc/logrotate.conf` for your environment, your customizations are saved in `/opt/serviced/etc/logrotate.conf.bak`. Compare the new and old versions, and update the new one as desired.

18 Upgrading delegates from 1.1.x to 1.4.1

Use this chapter to upgrade Control Center 1.1.x delegate hosts to 1.4.1. Before performing the procedures in this chapter, download required software. See *Downloading and staging required files* on page 13.

Updating to Docker CE

Use this procedure to update Docker to Docker Community Edition (Docker CE) version 17.03.1.

1. Log in as `root`, or as a user with superuser privileges.
2. Remove the Docker repository description file.

   Upgrades no longer require external repositories.

   ```
   rm -f /etc/yum.repos.d/docker.repo
   ```
3. 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 substeps.
   b. Disable automatic start of `serviced`.

      ```
      systemctl disable serviced
      ```
   c. 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
      ```
   d. Log in as `root`, or as a user with superuser privileges.
   e. Enable automatic start of `serviced`.

      ```
      systemctl enable serviced
      ```
4. Identify the name of the LVM thin pool for Docker.

   ```
   docker info 2>/dev/null | grep 'Pool Name'
   ```

Example result:

```
 Pool Name: docker-docker--pool
```

Record the name for use in a subsequent step.

5 Back up the Docker environment file.

```
test -f /etc/sysconfig/docker \
  && mv /etc/sysconfig/docker /etc/sysconfig/docker.bak
```

6 Stop the Docker service.

```
systemctl stop docker
```

7 Remove Docker 1.9.0.

a Remove without checking dependencies.

```
rpm -e --nodeps docker-engine-1.9.0
```

b Clean the `yum` databases.

```
yum clean all
```

8 Install Docker CE 17.03.1.

```
yum install --enablerepo=zenoss-mirror docker-ce-17.03.1.ce
```

If `yum` returns an error due to dependency issues, see *Resolving package dependency conflicts* on page 104 for potential resolutions.

Configuring Docker

Use this procedure to configure Docker.

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

2 Create a new Docker drop-in file.

a Create a directory for the drop-in file, if necessary.

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

b Create a backup of the drop-in file, if it exists.

```
test -f /etc/systemd/system/docker.service.d/docker.conf \
  && cp -p /etc/systemd/system/docker.service.d/docker.conf \
  /etc/systemd/system/docker.service.d/docker.conf.bak
```

c Create the new 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
```

3 Reload the `systemd` manager configuration.

```
systemctl daemon-reload
```

4 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, identified in a previous step:

```
myPool="/dev/mapper/Thin-Pool-Device"
```

b Create variables for adding arguments to the Docker environment 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 environment 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 as Docker updates the storage layout.

5 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
```

6 Compare the previous version of the Docker environment file with the new version, and add any customizations for your deployment to the new version.

```
diff /etc/sysconfig/docker /etc/sysconfig/docker.bak
```

If you change this file, restart Docker with `systemctl restart docker`.

Changing the local Docker registry endpoint

Use this procedure to configure the delegate host with the endpoint of the local Docker registry. Unless the master host is configured with an alternative local Docker registry, which is rare, the endpoint is the master host's hostname or IP address and port 5000.

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:5000` with the endpoint of the local Docker registry.

   If the master host is configured with an alternative local Docker registry, use the same endpoint here. Otherwise, just replace `localhost` with the IP address or fully-qualified domain name of the Control Center master host.
   e Save the file, and then close the editor.
3 Verify the settings.

```
grep -E '^\b*SERVICED' /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
```

Updating Control Center on delegate hosts

This procedure updates Control Center on delegate hosts to version 1.4.1.

Perform this procedure on each delegate host in your deployment.

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

2 Save the current `serviced` configuration file as a reference.

a Rename the file.

```
mv /etc/default/serviced /etc/default/serviced-pre-1.4.1
```

b Set permissions to read-only.

```
chmod 0440 /etc/default/serviced-pre-1.4.1
```

3 Remove the external Zenoss repository package.

Upgrades no longer require external repositories.

```
yum remove zenoss-repo-1-1.x86_64
```

4 Install the new version of Control Center.

```
yum install --enablerepo=zenoss-mirror \
  /opt/zenoss-repo-mirror/serviced-1.4.1-1.x86_64.rpm
```

If `yum` returns an error due to dependency issues, see *Resolving package dependency conflicts* on page 104 for potential resolutions.

5 Make a backup copy of the new configuration file.

a Copy the file.

```
cp /etc/default/serviced /etc/default/serviced-1.4.1-orig
```

b Set permissions to read-only.

```
chmod 0440 /etc/default/serviced-1.4.1-orig
```

6 Display the settings of the reference configuration file.

```
grep -E '^\b*SERVICED' /etc/default/serviced-pre-1.4.1
```

7 Open the new configuration file with a text editor, and then update the file for your environment.

Among other changes, the *SERVICED_NFS_CLIENT* variable is deprecated. Make note of the delegates that use the setting, and then rescind DFS access permission from the resource pool to which they belong. For more information, see *Updating resource pool permissions* on page 84.

For more information about configuring a delegate host, see *Control Center configuration variables* on page 109.

Updating the ZooKeeper image on ensemble nodes

Perform the steps in *Downloading and staging required files* on page 13 before performing this procedure.

Use this procedure to install a new Docker image for ZooKeeper on ZooKeeper ensemble nodes.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Identify the hosts in the ZooKeeper ensemble.

   ```
   grep -E '^\b*SERVICED_ZK=' /etc/default/serviced
   ```

   The result is a list of 3 or 5 hosts, separated by the comma character (`,`). The master host is always a node in the ZooKeeper ensemble.
3. Log in to a ZooKeeper ensemble node as `root`, or as a user with superuser privileges.
4. Change directory to `/root`.

   ```
   cd /root
   ```
5. Extract the ZooKeeper image.

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

   Image extraction begins when you press the **y** key.
6. Optional: Delete the archive file.

   ```
   rm -i ./install-zenoss-isvcs-zookeeper-v*.run
   ```
7. Repeat the preceding four steps on each delegate that is a node in the ZooKeeper ensemble.

19 Starting an upgraded deployment

This chapter includes procedures for starting single-host and multi-host Control Center deployments after upgrading to version 1.4.1.

Single-host deployments: *Starting and registering the master host (single-host deployment)* on page 77.

Multi-host deployments: Use the procedures in this section in the order shown in the following table.

Step		Procedure
1	Start/register the master host	*Starting and registering the master host (multi-host deployment)* on page 78
2	Start/register ZooKeeper ensemble hosts	*Updating delegate hosts with authentication* on page 79
3	Start the ZooKeeper ensemble	*Starting the ZooKeeper ensemble* on page 33
4	Start/register all remaining delegate hosts	*Updating delegate hosts with authentication* on page 79

Starting and registering the master host (single-host deployment)

Use this procedure to start the master host in a single-host deployment after upgrading Control Center to version 1.4.1. This procedure also includes steps to create and register the authentication credentials the master needs for its role as a delegate.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Determine whether `serviced` is configured to start when the system starts.

```
systemctl is-enabled serviced
```

   - If the result is `enabled`, proceed to the next step.
   - If the result is `disabled`, enter the following command:

```
systemctl enable serviced
```

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

```
grep -E '^\b*SERVICED' /etc/default/serviced
```

4 Start `serviced`, and then monitor the startup.

During this startup, `serviced` invokes `docker pull` to retrieve its updated images.

```
systemctl daemon-reload && systemctl start serviced \
  && journalctl -u serviced -f -o cat
```

Do not proceed to the next step until the following message is displayed:

```
Host Master successfully started
```

5 Obtain the host ID of the master host.

a Display the host IDs of all cluster hosts.

```
serviced host list | cut -c-85
```

b Record the host ID of the master host.

6 Create authentication credentials for the master host, and register the credentials.
Replace *Host-ID* with the host ID of the master host:

```
serviced key reset --register Host-ID
```

Starting and registering the master host (multi-host deployment)

Use this procedure to start the master host in a multi-host deployment after upgrading Control Center to version 1.4.1. This procedure also includes steps to create and register the authentication credentials the master needs for its role as a delegate.

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

2 Determine whether `serviced` is configured to start when the system starts.

```
systemctl is-enabled serviced
```

- If the result is `enabled`, proceed to the next step.
- If the result is `disabled`, enter the following command:

```
systemctl enable serviced
```

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

```
grep -E '^\b*SERVICED' /etc/default/serviced
```

4 Configure the master host to start independent of the ZooKeeper ensemble.

This step simplifies the process of registering the master host and its effect is temporary—a subsequent step restores the configuration file.

a Create a backup of the `serviced` configuration file.

```
cp -p /etc/default/serviced /etc/default/serviced.tmp
```

b Append a new declaration for the *SERVICED_ZK* variable.

```
echo "SERVICED_ZK=$(hostname -i):2181" >> /etc/default/serviced
```

5 Start `serviced`, and then monitor the startup.

During this startup, `serviced` invokes `docker pull` to retrieve its updated images.

```
systemctl start serviced && journalctl -u serviced -f -o cat
```

Do not proceed to the next step until the following message is displayed:

```
Host Master successfully started
```

6 Obtain the host ID of the master host.

a Display the host IDs of all cluster hosts.

```
serviced host list | cut -c-85
```

b Record the host ID of the master host.

7 Create authentication credentials for the master host, and register the credentials.
Replace *Host-ID* with the host ID of the master host:

```
serviced key reset --register Host-ID
```

8 Restore the `serviced` configuration file, and then restart the service.

a Restore the `serviced` configuration file.

```
mv /etc/default/serviced.tmp /etc/default/serviced
```

b Restart the service.

```
systemctl daemon-reload && systemctl start serviced \
  && journalctl -u serviced -f -o cat
```

9 Identify the hosts in the ZooKeeper ensemble.

```
grep -E '^\b*SERVICED_ZK=' /etc/default/serviced
```

The result is a list of 3 or 5 hosts, separated by the comma character (`,`). The master host is always a node in the ZooKeeper ensemble.

Updating delegate hosts with authentication

Starting with version 1.3.0, Control Center requires authentication tokens for all delegate communications. The tokens are based on RSA key pairs created by the master `serviced` instance. When you create a key pair for a delegate, `serviced` bundles its public key with the delegate's private key. The `serviced` instance on the delegate installs the credentials and uses them to sign messages with the required unique tokens.

Credentials are installed by using an SSH connection or a file.

- The command to create a key pair can initiate an SSH connection with a delegate and install credentials. 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 create a key pair creates a file containing the credentials. You can move the credentials file to the delegate host with any file transfer method, and then install it on the delegate.

The following procedures demonstrate how to create credentials and install them on a delegate.

Starting and registering a delegate using SSH

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 start a delegate host after upgrading Control Center to version 1.4.1. This procedure also includes steps to create and register the authentication credentials the delegate needs, through an SSH connection.

1. Log in to the delegate host as `root`, or as a user with superuser privileges.
2. Determine whether `serviced` is configured to start when the system starts.

   ```
   systemctl is-enabled serviced
   ```

   - If the result is `enabled`, proceed to the next step.
   - If the result is `disabled`, enter the following command:

     ```
     systemctl enable serviced
     ```

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

   ```
   grep -E '^\b*SERVICED' /etc/default/serviced
   ```

4. Start `serviced`, and then monitor the startup.

   During this startup, `serviced` invokes `docker pull` to retrieve its updated images.

   ```
   systemctl daemon-reload && systemctl start serviced \
     && journalctl -u serviced -f -o cat
   ```

   Do not proceed to the next step until the following message is displayed:

   ```
   Host Agent successfully started
   ```

5. Log out of the delegate host.
6. Log in to the master host as `root`, or as a user with superuser privileges.
7. Obtain the host ID of the delegate host started previously.
   a. Display the host IDs of all cluster hosts.

      ```
      serviced host list | cut -c-85
      ```

   b. Record the host ID of the delegate host.
8. Create authentication credentials for the delegate host, and register the credentials.

   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.

Replace *Host-ID* with the host ID of the delegate host started previously:

```
serviced key reset --register Host-ID
```

Starting and registering a delegate using a file

Use this procedure to start a delegate host after upgrading Control Center to version 1.4.1. This procedure also includes steps to create a credentials file and to use the file to register the delegate.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Obtain the host ID of the delegate host to start and register.
   a. Display the host IDs of all cluster hosts.

      ```
      serviced host list | cut -c-85
      ```

   b. Record the host ID of the delegate host.
3. Create authentication credentials for the delegate host.
   Replace *Host-ID* with the host ID of the delegate host identified in the preceding step:

   ```
   serviced key reset Host-ID
   ```

   The command creates a unique credentials file in the local directory.
4. Use a file transfer utility such as `scp` to copy the credentials file to the delegate host.
   Once copied to the delegate host, the credentials file is not needed on the master host and can be deleted.
5. Log in to the Control Center delegate host as `root`, or as a user with superuser privileges.
6. Determine whether `serviced` is configured to start when the system starts.

   ```
   systemctl is-enabled serviced
   ```

   - If the result is `enabled`, proceed to the next step.
   - If the result is `disabled`, enter the following command:

     ```
     systemctl enable serviced
     ```

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

   ```
   grep -E '^\b*SERVICED' /etc/default/serviced
   ```

8. Start `serviced`, and then monitor the startup.
   During this startup, `serviced` invokes `docker pull` to retrieve its updated images.

   ```
   systemctl daemon-reload && systemctl start serviced \
     && journalctl -u serviced -f -o cat
   ```

   Do not proceed to the next step until the following message is displayed:

   ```
   Host Agent successfully started
   ```

9. Install the credentials.
   Replace *Credentials-File* with the pathname of the credentials file:

   ```
   serviced host register Credentials-File
   ```

10 Delete the credentials file.

The file is no longer needed on the delegate host.

Replace *Credentials-File* with the pathname of the credentials file:

```
rm Credentials-File
```

Starting the ZooKeeper ensemble

Use this procedure to start the ZooKeeper ensemble.

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

2 Determine whether `serviced` is configured to start when the system starts.

```
systemctl is-enabled serviced
```

- If the result is `enabled`, proceed to the next step.
- If the result is `disabled`, enter the following command:

```
systemctl enable serviced
```

3 Identify the hosts in the ZooKeeper ensemble.

```
grep -E '^\b*SERVICED_ZK=' /etc/default/serviced
```

The result is a list of 3 or 5 hosts, separated by the comma character (`,`). The master host is always a node in the ZooKeeper ensemble.

4 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.

5 On all 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 daemon-reload && systemctl start serviced
```

6 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 /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 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.

20 After upgrading from 1.1.x to 1.4.1

Perform the procedures in this chapter after Control Center is upgraded.

Updating resource pool permissions

Use this procedure to identify and change the permissions associated with one or more resource pools. During the upgrade to version 1.4.1, existing resource pools are assigned both administrative and DFS permissions.

In this release, the command that displays information about pools uses integer values for the Permissions field. The following list associates the values with the permissions they represent:

`1`, administrative permission
`2`, DFS access permission
`3`, both administrative and DFS access permissions

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

   ```
   serviced pool list -v | grep -E 'ID|Permissions'
   ```

   Example result:

   ```
   "ID": "default",
   "Permissions": 3,
   "ID": "master",
   "Permissions": 3,
   ```

   In the preceding example, the `default` and `master` resource pools have administrative permission and DFS access permission.
3. Optional: Remove DFS access permission from a pool.

   If you intend to remove both DFS access and administrative access permissions from a resource pool, you must remove DFS access permissions first.

   Replace *Pool-Name* with the name of a resource pool from the previous step:

   ```
   serviced pool set-permission --dfs=false Pool-Name
   ```
4. Optional: Remove administrative permission from a pool.

If you intend to remove both DFS access and administrative access permissions from a resource pool, you must remove DFS access permissions first.

Replace *Pool-Name* with the name of a resource pool from a previous step:

```
serviced pool set-permission --admin=false Pool-Name
```

Setting the connection timeout of a resource pool

Use this procedure to set the length of time the scheduler waits for a disconnected delegate host to rejoin its pool before moving the services scheduled for the delegate to a different host in the pool. This feature is useful for remote resource pools that are connected through a high-latency, wide-area network.

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
```

Removing unused images

Use this procedure to identify and remove unused Control Center images.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Identify the images associated with the installed version of `serviced`.

```
serviced version | grep Images
```

   Example result:

```
IsvcsImages: [zenoss/serviced-isvcs:v60 zenoss/isvcs-zookeeper:v10]
```

3. Display the `serviced` images in the local repository.

```
docker images | awk '/REPO|isvcs/'
```

   Example result (edited to fit):

```
REPOSITORY                      TAG       IMAGE ID
zenoss/serviced-isvcs           v40       88cd6c24cc82
zenoss/serviced-isvcs           v60       0aab5a2123f2
zenoss/isvcs-zookeeper          v3        46fa0a2fc4bf
```

```
zenoss/isvcs-zookeeper          v10       0ff3b3117fb8
```

The example result shows the current versions and one set of previous versions. Your result may include additional previous versions and will show different images IDs.

4 Remove unused images.
Replace *Image-ID* with the image ID of an image for a previous version.

```
docker rmi Image-ID
```

Repeat this command for each unused image.

Removing orphaned snapshot devices

Use this procedure to identify orphaned snapshot devices in the LVM thin pool for application data, and to remove them.

1 Log in to the master host as `root`, or as a user with superuser privileges.
2 Identify the location of tenant volumes.

```
grep -E '^\b*SERVICED_VOLUMES_PATH' /etc/default/serviced
```

- If the command returns a result, the location of tenant volumes is the value of the *SERVICED_VOLUMES_PATH* variable.
- If the command does not return a result, the location of tenant volumes is the default value of *SERVICED_VOLUMES_PATH*, `/opt/serviced/var/volumes`.

3 Identify the device of the `serviced` thin pool belongs.

```
ls /dev/mapper | grep serviced
```

Example result:

```
serviced-serviced--pool
serviced-serviced--pool_tdata
serviced-serviced--pool_tmeta
```

The first result is the thin pool device. The other results represent the data and metadata portions of the device.

4 Check for orphaned snapshot devices.
Replace *Thin-Pool-Device* with the name of the thin pool device from the previous step, and replace *Volumes-Path* with the location of tenant volumes:

```
serviced-storage -o dm.thinpooldev=/dev/mapper/Thin-Pool-Device \
  check Volumes-Path
```

- If the result is `No orphaned devices found`, stop. There are no orphaned snapshot devices to remove.
- If the result is `Orphaned devices were found`, perform the next step.

5 Remove orphaned snapshot devices.
Replace *Thin-Pool-Device* with the name of the thin pool device from the previous step, and replace *Volumes-Path* with the location of tenant volumes:

```
serviced-storage -o dm.thinpooldev=/dev/mapper/Thin-Pool-Device \
```

```
check -c Volumes-Path
```

Updating the OpenTSDB time-to-live value

Older versions of Control Center used a longer time-to-live (TTL) value for data maintained by the OpenTSDB database. The correct value for this version is 2592000 seconds (30 days). Use this procedure to update the TTL value, if necessary.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Start an interactive shell in the OpenTSDB container.

```
docker exec -it serviced-isvcs_opentsdb bash
```

3. Start an interactive HBase shell.

```
/opt/hbase/bin/hbase shell
```

Example result:

```
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 0.94.16, r1557241, Fri Jan 10 20:43:03 UTC 2014

hbase(main):001:0>
```

4. Display the current settings of the `tsdb` table.

```
describe 'tsdb'
```

   - If the result includes `TTL => '2592000'`, stop. Use the `exit` command twice, to end the HBase shell and then the shell in the OpenTSDB container.
   - If the result includes a larger value for the TTL setting, perform the remaining steps.

5. Disable the `tsdb` table.

```
disable 'tsdb'
```

6. Set the TTL value to 2592000 seconds (30 days).

```
alter 'tsdb', {NAME=>'t', TTL=>'2592000'}
```

7. Enable the `tsdb` table.

```
enable 'tsdb'
```

8. Display the current settings.

```
describe 'tsdb'
```

   - If the result includes `TTL => '2592000'`, proceed to the next step.
   - If the result includes a different value for the TTL setting, repeat the preceding steps.

9. End the interactive HBase shell.

```
exit
```

10 End the interactive shell in the OpenTSDB container.

```
exit
```

Managing maintenance scripts

The following topics provide information about the maintenance scripts that are installed when Control Center is installed on a host.

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.daily/serviced`

This script invokes `logrotate` daily, to manage the `/var/log/serviced.access.log` file.

This script is required on the master host and on all delegate hosts.

`/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.weekly/serviced-zenossdbpack`

This script invokes a `serviced` command weekly, which in turn invokes the database maintenance script for a Zenoss application. If the application is not installed or is offline, the command fails.

This script is required on the master host only.

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.daily/serviced`

This script invokes `logrotate` daily, to manage the `/var/log/serviced.access.log` file.

This script is required on the master host and on all delegate hosts.

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

This script should be removed from delegate hosts.

`/etc/cron.weekly/serviced-zenossdbpack`
: This script should be removed from delegate hosts.

Control Center application data storage requirements

A

Control Center uses an LVM thin pool to store tenant (application) data. LVM thin pools include separate storage areas for data and for metadata. For each tenant it manages, Control Center maintains a separate virtual device (a volume) in the data storage area of its LVM thin pool.

To ensure consistency, Control Center requires the following, minimum amount of free space in its thin pool:

- 3GiB available for each tenant volume
- 3GiB available in the data storage area
- 62MiB available in the metadata storage area
- The total amount of metadata storage must be 1% of total data storage

Examining application data storage status

Beginning with release 1.3.0, Control Center initiates an emergency shutdown when the minimum required amounts of free space are not available in the `serviced` thin pool or tenant volumes.

Use this procedure to display the amount of free space in a Control Center thin pool and tenant volumes, to determine how much space is available in the LVM volume group that contains the thin pool, and to determine whether additional steps are required.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Display the amount of space available in the `serviced` thin pool.

```
serviced volume status
```

Figure 1: Example output with highlighted values

```
Status for volume /opt/serviced/var/volumes:

Driver:      devicemapper
Driver Type: direct-lvm
Volume Path: /opt/serviced/var/volumes

Thin Pool
---------
Logical Volume:               serviced-serviced--pool
Metadata (total/used/avail):  2.2 GiB / 64.18 MiB (2.85%) / 2.137 GiB (97.15%)
Data (total/used/avail):      200 GiB   / 30.576 GiB (15%)   / 169.424 GiB (85%)

3ir81rg1h8b09ipowynz3qx2f Application Data
------------------------------------------
Volume Mount Point:            /opt/serviced/var/volumes/3ir81rg1h8b09ipowynz3qx2f
Filesystem (total/used/avail): 98.31 GiB / 1.511 GiB (1.5%) / 91.78 GiB (93%)
Virtual device size:           100 GiB
```

The result includes detailed information about the `serviced` thin pool and each tenant volume.

- If the amount of free space in the `serviced` thin pool is sufficient, stop. No further action is required.
- If the amount of free space in the data or metadata portions of the `serviced` thin pool is not sufficient, perform the following steps.

3 Identify the volume group to which the `serviced` thin pool belongs.

```
lvs --options=lv_name,vg_name,lv_size
```

The volume group associated with `serviced-pool` contains the `serviced` thin pool.

4 Display the amount of free space in the volume group that contains the `serviced` thin pool. Replace *Volume-Group* with the name of the volume group identified in the previous step:

```
vgs --no-headings --options=vg_free Volume-Group
```

- If the amount of free space in the volume group is not sufficient to increase one or both of the storage areas of the `serviced` thin pool to their required minimums, add physical or logical storage to the volume group. For more information, refer to your operating system documentation.
- If the amount of free space in the volume group is sufficient to increase one or both of the storage areas of the `serviced` thin pool to their required minimums, proceed to the next step

5 Increase the free space in one or both areas of the `serviced` thin pool.

- To increase the amount of space in the metadata area, proceed to *Adding space to the metadata area of a Control Center thin pool* on page 91.
- To increase the amount of space in the data area, proceed to *Adding space to the data area of a Control Center thin pool* on page 92.
- To increase the size of a tenant volume, which relies on the data area of the thin pool, proceed to *Adding space to a tenant volume* on page 92.

Adding space to the metadata area of a Control Center thin pool

Use this procedure to increase the amount of space in the metadata area of a Control Center thin pool.

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

2 Display information about LVM logical volumes on the host.

```
lvs --options=lv_name,vg_name,lv_size
```

Typically, the logical volume is `serviced-pool` and the containing volume group is `serviced`.

3 Add space to the metadata storage area of the `serviced` thin pool.
In the following command:

- Replace *Size* with the amount of space to add (in megabytes) and the units identifier (`M`).
- Replace *Volume-Group* with the name of the LVM volume group identified in the previous step.
- Replace *Logical-Volume* with the name of the logical volume identified in the previous step.

```
lvextend -L+SizeM Volume-Group/Logical-Volume_tmeta
```

Adding space to the data area of a Control Center thin pool

Use this procedure to increase the amount of space in the data area of a Control Center thin pool.

1 Log in to the master host as `root`, or as a user with superuser privileges.
2 Display information about LVM logical volumes on the host.

```
lvs --options=lv_name,vg_name,lv_size
```

Typically, the logical volume is `serviced-pool` and the containing volume group is `serviced`.

3 Add space to the data storage area of the `serviced` thin pool.
In the following command:

- Replace *Total-Size* with the sum of the existing device size plus the space to add to the device, in gigabytes. Include the units identifier, `G`.
- Replace *Volume-Group* with the name of the LVM volume group identified in the previous step.
- Replace *Logical-Volume* with the name of the logical volume identified in the previous step.

```
lvextend -L+Total-SizeG Volume-Group/Logical-Volume
```

Adding space to a tenant volume

Use this procedure to increase the size of a tenant volume in a Control Center thin pool.

1 Log in to the master host as `root`, or as a user with superuser privileges.
2 Identify the tenant device to resize.

```
serviced volume status
```

3 Display the device mapper name of the `serviced` thin pool.

```
grep -E '^\b*SERVICED_DM_THINPOOLDEV' /etc/default/serviced \
  | sed -e 's/.*=//'
```

Typically, the name is `/dev/mapper/serviced-serviced--pool`.

4 Increase the size of the tenant device.
In the following command:

- Replace *Device-Mapper-Name* with the device mapper name of the thin pool.

- Replace *Tenant-ID* with the identifier of the tenant device.
- Replace *Total-Size* with the sum of the existing device size plus the space to add to the device, in gigabytes. Include the units identifier, `G`.

```
serviced-storage resize -d /opt/serviced/var/volumes \
  -o dm.thinpooldev=Device-Mapper-Name Tenant-ID Total-SizeG
```

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
```

User access control

C

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 cluster 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 Control Center supports using 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) is supported and 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.

Configuring a private master NTP server D

Control Center requires a common time source. The procedures in this appendix configure a private master *NTP* server to synchronize the system clocks of all hosts in a Control Center cluster.

A private master server is only required when a multi-host deployment does not have internet access and no time synchronization servers are available behind the firewall. Single-host deployments do not require time synchronization, and deployments with internet access can rely on the default public time servers that are configured in `/etc/ntp.conf`.

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 or to use a time synchronization server that is available behind the firewall. For more information, refer to the VMware documentation for your version of vSphere.

The procedures in this appendix assume that Control Center is not installed. If it is installed, stop the `serviced` service before performing configuring NTP.

Configuring an NTP master server

Use this procedure to configure an NTP master server on the Control Center master host. Perform this procedure only if the host does not have internet access.

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

1. Log in to the Control Center master 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:

```
# Use the local clock
server 127.127.1.0 prefer
fudge  127.127.1.0 stratum 10
driftfile /var/lib/ntp/drift
```

```
broadcastdelay 0.008

# Give localhost full access rights
restrict 127.0.0.1

# Grant access to client hosts
restrict Address-Range mask Netmask nomodify notrap
```

c Replace *Address-Range* with the range of IPv4 network addresses that are allowed to query this NTP server.

For example, the following IP addresses are assigned to the hosts in an Control Center cluster:

```
203.0.113.10
203.0.113.11
203.0.113.12
203.0.113.13
```

For the preceding addresses, the value for *Address-Range* is `203.0.113.0`.

d Replace *Netmask* with the IPv4 network mask that corresponds with the address range.
For example, a valid network mask for `203.0.113.0` is `255.255.255.0`.

e Save the file and exit the editor.

4 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
```

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 delegate host in your Control Center cluster.

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

---

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 IPv4 address of the host where the NTP server is running (the Control Center master host).

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
```

Resolving package dependency conflicts

E

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 releases and images

F

This appendix associates Control Center releases with the tags of the required Docker images for each release. The images provide the virtual containers of the Control Center internal services and the ZooKeeper service. In addition, this appendix includes a procedure for identifying installed images.

Releases and image tags

Release 1.4

Release	Internal services image tag	ZooKeeper image tag
Control Center 1.4.1	zenoss/serviced-isvcs:v60	zenoss/isvcs-zookeeper:v10
Control Center 1.4.0	zenoss/serviced-isvcs:v60	zenoss/isvcs-zookeeper:v10

Release 1.3

Release	Internal services image tag	ZooKeeper image tag
Control Center 1.3.3	zenoss/serviced-isvcs:v56	zenoss/isvcs-zookeeper:v8
Control Center 1.3.2	zenoss/serviced-isvcs:v56	zenoss/isvcs-zookeeper:v8
Control Center 1.3.2	zenoss/serviced-isvcs:v56	zenoss/isvcs-zookeeper:v8
Control Center 1.3.1	zenoss/serviced-isvcs:v56	zenoss/isvcs-zookeeper:v8
Control Center 1.3.0	zenoss/serviced-isvcs:v56	zenoss/isvcs-zookeeper:v8

Release 1.2

Release	Internal services image tag	ZooKeeper image tag
Control Center 1.2.3	zenoss/serviced-isvcs:v54	zenoss/isvcs-zookeeper:v8
Control Center 1.2.2	zenoss/serviced-isvcs:v54	zenoss/isvcs-zookeeper:v8
Control Center 1.2.1	zenoss/serviced-isvcs:v54	zenoss/isvcs-zookeeper:v8
Control Center 1.2.0	zenoss/serviced-isvcs:v53	zenoss/isvcs-zookeeper:v8

Identifying installed Docker images

Use this procedure to identify the local Docker images for Control Center that are installed on a host.

1. Log in to the Control Center host as `root`, or as a user with superuser privileges.
2. Display the local Docker images for Control Center.

```
docker images | awk '/isvcs/ { print $1, " ", $2}'
```

- If the installed image versions are higher than the versions that accompany a release, the images need to be updated. The upgrade procedures include steps for installing the required images.
- If the installed image versions are not higher than the versions that accompany a release, the images do not need to be updated.

Removing unused images

Use this procedure to identify and remove unused Control Center images.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Identify the images associated with the installed version of `serviced`.

```
serviced version | grep Images
```

Example result:

```
IsvcsImages: [zenoss/serviced-isvcs:v60 zenoss/isvcs-zookeeper:v10]
```

3. Display the `serviced` images in the local repository.

```
docker images | awk '/REPO|isvcs/'
```

Example result (edited to fit):

```
REPOSITORY                      TAG       IMAGE ID
zenoss/serviced-isvcs           v40       88cd6c24cc82
zenoss/serviced-isvcs           v60       0aab5a2123f2
zenoss/isvcs-zookeeper          v3        46fa0a2fc4bf
zenoss/isvcs-zookeeper          v10       0ff3b3117fb8
```

The example result shows the current versions and one set of previous versions. Your result may include additional previous versions and will show different images IDs.

4. Remove unused images.
Replace *Image-ID* with the image ID of an image for a previous version.

```
docker rmi Image-ID
```

Repeat this command for each unused image.

Control Center configuration variables

G

This appendix includes the content in the following list:

1 *Removed variables (at version 1.2.0)* on page 109
2 *New variables (at version 1.2.0)* on page 110
3 *Master host configuration variables* on page 111
4 *Delegate host configuration variables* on page 114
5 *Universal configuration variables* on page 116
6 *Best practices for configuration files* on page 118
7 *Control Center configuration file* on page 118

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

Removed variables (at version 1.2.0)

The variables in the following table were present in version 1.1.1 and are not present in version 1.2.0. The variables are shown in the order in which they appeared in `/etc/default/serviced`.

Variable	Explanation
TMP	Not POSIX compliant. Replaced by *TMPDIR*.
SERVICED_AGENT	*SERVICED_MASTER* is sufficient to specify the host role.
SERVICED_VARPATH	*SERVICED_ISVCS_PATH*, *SERVICED_VOLUMES_PATH*, and *SERVICED_BACKUPS_PATH* are more specific and preferred.
SERVICED_MONITOR_DFS_MASTER_INTERVAL *SERVICED_MONITOR_DFS_MASTER_RESTART* *SERVICED_MONITOR_DFS_REMOTE_UPDATE_INTERVAL*	Control Center now uses different mechanisms to monitor DFS status.

New variables (at version 1.2.0)

The variables in the following table were not present in version 1.1.1 and are present in version 1.2.0. The variables are shown in the order in which they appear in `/etc/default/serviced`.

Variable	Purpose
TMPDIR	Replacement for *TMP*.
SERVICED_MASTER_IP	The IP address of the master host.
SERVICED_MASTER_POOLID	The name of the resource pool to which the master host belongs.
SERVICED_RPC_TLS_MIN_VERSION *SERVICED_RPC_TLS_CIPHERS*	Additional options for encrypting RPC communications.
SERVICED_UI_POLL_FREQUENCY	The frequency at which browser interface clients poll the server.
SERVICED_MUX_DISABLE_TLS *SERVICED_MUX_TLS_MIN_VER* *SERVICED_MUX_TLS_CIPHERS*	Additional options for encrypting RPC communications.
SERVICED_DM_ARGS *SERVICED_DM_BASESIZE* *SERVICED_DM_LOOPDATASIZE* *SERVICED_DM_LOOPMETADATASIZE* *SERVICED_DM_THINPOOLDEV*	Options for the `devicemapper` storage driver.
SERVICED_STORAGE_STATS_UPDATE_INTERVAL	The number of seconds between polls of kernel statistics about the application data thin pool.
SERVICED_LOGSTASH_CYCLE_TIME	The amount of time between logstash purges.
SERVICED_SVCSTATS_CACHE_TIMEOUT	The lifetime of data in the browser interface cache for statistics about services.
SERVICED_NFS_CLIENT	DEPRECATED: Prevent a delegate host from mounting the DFS. In version 1.2.0, delegate host access to the DFS is controlled through resource pool permissions.
SERVICED_DOCKER_DNS	The Docker DNS configuration for containers.
SERVICED_SNAPSHOT_USE_PERCENT	The amount of free space in the application data thin pool, used to determine whether the pool can store a new snapshot.
SERVICED_ZK_SESSION_TIMEOUT	The number of seconds the ZooKeeper leader waits before flushing an inactive connection.
SERVICED_ES_STARTUP_TIMEOUT	The number of seconds to wait for the Elasticsearch service to start.
SERVICED_RPC_DIAL_TIMEOUT	The number of seconds until an RPC connection attempt times out.
SERVICED_AUTH_TOKEN_EXPIRATION	The lifetime of a delegate host revocation certificate.

Variable	Purpose
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.

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.

Storage variables

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

- Use one of the first two groups of variables but not both.
- Before starting the master host for the first time, you might need to change the defaults of the third group.
- Typically, the defaults of the last two groups of variables are not changed until Control Center has managed an application for a while and a need arises.

The *SERVICED_STORAGE_STATS_UPDATE_INTERVAL* variable sets the interval for collecting kernel statistics about the application data thin pool. Its default value is unlikely to require a change until a need arises.

Variable	Description	Purpose
SERVICED_FS_TYPE *SERVICED_DM_ARGS* *SERVICED_DM_BASESIZE* *SERVICED_DM_THINPOOLDEV*	The specifications of a `devicemapper`-based application data storage resource for production use.	Provide basic information about the data storage resource.
SERVICED_FS_TYPE *SERVICED_DM_LOOPDATASIZE* *SERVICED_DM_LOOPMETADATASIZE* *SERVICED_ALLOW_LOOP_BACK*	The specifications of a `devicemapper`-based application data storage resource for development use.	Provide basic information about the data storage resource.
SERVICED_ISVCS_PATH *SERVICED_VOLUMES_PATH* *SERVICED_BACKUPS_PATH*	The data storage paths of separate functional components of Control Center internal services.	Enable separate storage areas for one or more components. The default installation process puts all three components on the same device.
SERVICED_SNAPSHOT_TTL *SERVICED_SNAPSHOT_USE_PERCENT* *SERVICED_MAX_DFS_TIMEOUT*	The snapshot retention interval, the percentage of the data storage thin pool that is unused, and the snapshot attempt timeout interval.	Prevent the creation of snapshots that are too large to fit the thin pool.
SERVICED_LOGSTASH_MAX_DAYS *SERVICED_LOGSTASH_MAX_SIZE* *SERVICED_LOGSTASH_CYCLE_TIME*	The variables that manage the amount of space used by the application log storage service.	Prevent application logs from filling the storage device that logstash uses.

Internal services endpoint variables

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

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

Variable	Endpoint	Description
SERVICED_DOCKER_REGISTRY	(varies)	The local Docker registry for Control Center internal services images and application images.
SERVICED_ENDPOINT	*Master-Host*`:4979`	The `serviced` RPC server. The endpoint port number must match the value of *SERVICED_RPC_PORT*.
SERVICED_LOG_ADDRESS	*Master-Host*`:5042`	The *logstash* service.
SERVICED_LOGSTASH_ES	*Master-Host*`:9100`	The Elasticsearch service for logstash.
SERVICED_STATS_PORT	*Master-Host*`:8443`	The `serviced` metrics consumer service.
SERVICED_AUTH_TOKEN_EXPIRATION	(none)	The length of time a delegate authentication token is valid.

RPC service variables

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

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.

Internal services endpoint variables

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

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

Variable	Endpoint	Description
SERVICED_DOCKER_REGISTRY	(varies)	The local Docker registry for Control Center internal services images and application images.
SERVICED_ENDPOINT	*Master-Host*:4979	The `serviced` RPC server. The endpoint port number must match the value of *SERVICED_RPC_PORT*.
SERVICED_LOG_ADDRESS	*Master-Host*:5042	The *logstash* service.
SERVICED_LOGSTASH_ES	*Master-Host*:9100	The Elasticsearch service for logstash.
SERVICED_STATS_PORT	*Master-Host*:8443	The `serviced` metrics consumer service.
SERVICED_AUTH_TOKEN_EXPIRATION	(none)	The length of time a delegate authentication token is valid.

RPC service variables

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

Variable	Description
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 cluster 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.

Variable	Description
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.

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 cluster 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 cluster hosts.

SERVICED_UI_PORT

Default: `:443`

The port on which the HTTP server listens for requests. The value may be expressed as follows:

- *IP-Address*:*Port-Number*
- :*Port-Number*
- *Port-Number*

All Control Center cluster hosts must have the same value for this variable.

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 cluster 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 cluster 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` is supported.

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 (`,`).