Zenoss Core Upgrade Guide

Release 5.1.1

Zenoss, Inc.

www.zenoss.com

Zenoss Core Upgrade Guide

Part Number: 1091.16.068

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

Contents

About this guide

Zenoss Core Upgrade Guide provides detailed instructions for upgrading Zenoss Core from one minor or micro version to a more recent version.

Note Zenoss strongly recommends reviewing the *Zenoss Core Planning Guide* carefully before using this guide.

Related publications

Title	Description
Zenoss Core Administration Guide	Provides an overview of Zenoss Core architecture and features, as well as procedures and examples to help use the system.
Zenoss Core Configuration Guide	Provides required and optional configuration procedures for Zenoss Core, to prepare your deployment for monitoring in your environment.
Zenoss Core Installation Guide	Provides detailed information and procedures for creating deployments of Control Center and Zenoss Core.
Zenoss Core Planning Guide	Provides both general and specific information for preparing to deploy Zenoss Core.
Zenoss Core Release Notes	Describes known issues, fixed issues, and late-breaking information not already provided in the published documentation set.
Zenoss Core Upgrade Guide	Provides detailed information and procedures for upgrading deployments of Zenoss Core.

Additional information and comments

Zenoss welcomes your comments and suggestions regarding our documentation. To share your comments, please send an email to `docs@zenoss.com`. In the email, include the document title and part number. The part number appears at the end of the list of trademarks, at the front of this guide.

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 *Zenoss Core Release Notes*.

1091.16.068

Add a list of links to the overview of Part I.

After converting the storage driver, perform a full backup.

The Docker configuration file needs a longer startup timeout value, to work around a known Docker issue with the `devicemapper` driver. All Docker configuration steps now include adding `TimeoutSec=300`.

1091.16.067

A new part is added, for upgrading only Control Center. Both parts are renamed to reflect the addition.

The scope of supported upgrade paths is changed to reflect the micro release of Control Center.

All Docker configuration steps now add the storage driver flag (`-s devicemapper`) to the `/etc/sysconfig/docker` file.

All resource pool host upgrade procedures include a step to umnount the distributed file system before restarting `serviced`.

A link the the post-upgrade chapter is added to the end of upgrade procedures, if one is available.

1091.16.060.1

Upgrades are grouped in parts by scope. Each part contains a preparation chapter, chapters for the supported upgrade paths, and a post-upgrade chapter. Only the latest scope is in this version of the guide; previous scopes are in earlier versions.

A description of Zenoss Toolbox is included as an appendix.

1 Supported software and upgrade paths

Beginning with version 5.0.0, distributions of Zenoss Core include an additional component, Control Center. Each component is developed and maintained separately, and each has its own version number. This chapter identifies the combinations of component versions that Zenoss supports, and the supported upgrade paths between the combinations.

Supported combinations

The following table shows the Control Center and Zenoss Core release dates and the corresponding version combination for that release:

Release Date	Control Center	Zenoss Core
4 Mar 2016	1.1.2	5.1.1
29 Feb 2016	1.1.1	5.1.1
20 Feb 2016	1.0.10	5.0.10
02 Dec 2015	1.0.9	5.0.9
16 Nov 2015	1.0.8	5.0.8
10 Oct 2015	1.0.7	5.0.7
14 Sep 2015	1.0.6	5.0.6
05 Aug 2015	1.0.5	5.0.5
10 Jul 2015	1.0.4	5.0.4
27 May 2015	1.0.3	5.0.3
20 Apr 2015	1.0.2	5.0.2
03 Apr 2015	1.0.1	5.0.1
24 Feb 2015	1.0.0	5.0.0

Supported upgrade paths

Upgrade only Control Center

From combination	To combination
Control Center 1.1.1 and Zenoss Core 5.1.1	Control Center 1.1.2 and Zenoss Core 5.1.1

Upgrade Control Center and Zenoss Core

From combination	To combination
Control Center 1.0.6 and Zenoss Core 5.0.6	Control Center 1.1.2 and Zenoss Core 5.1.1
Control Center 1.0.7 and Zenoss Core 5.0.7	Control Center 1.1.2 and Zenoss Core 5.1.1
Control Center 1.0.8 and Zenoss Core 5.0.8	Control Center 1.1.2 and Zenoss Core 5.1.1
Control Center 1.0.9 and Zenoss Core 5.0.9	Control Center 1.1.2 and Zenoss Core 5.1.1
Control Center 1.0.10 and Zenoss Core 5.0.10	Control Center 1.1.2 and Zenoss Core 5.1.1

Recent upgrade paths

The following tables identify upgrade paths that are supported but documented only in previous editions of this guide.

Table 1: Upgrade to 1.0.10 / 5.0.10

From combination	To combination
Control Center 1.0.3 and Zenoss Core 5.0.3	Control Center 1.0.10 and Zenoss Core 5.0.10
Control Center 1.0.4 and Zenoss Core 5.0.4	Control Center 1.0.10 and Zenoss Core 5.0.10
Control Center 1.0.5 and Zenoss Core 5.0.5	Control Center 1.0.10 and Zenoss Core 5.0.10
Control Center 1.0.6 and Zenoss Core 5.0.6	Control Center 1.0.10 and Zenoss Core 5.0.10
Control Center 1.0.7 and Zenoss Core 5.0.7	Control Center 1.0.10 and Zenoss Core 5.0.10

From combination	To combination
Control Center 1.0.8 and Zenoss Core 5.0.8	Control Center 1.0.10 and Zenoss Core 5.0.10
Control Center 1.0.9 and Zenoss Core 5.0.9	Control Center 1.0.10 and Zenoss Core 5.0.10

Table 2: Upgrade to 1.0.3 / 5.0.3

From combination	To combination
Control Center 1.0.0 and Zenoss Core 5.0.0	Control Center 1.0.3 and Zenoss Core 5.0.3
Control Center 1.0.1 and Zenoss Core 5.0.1	Control Center 1.0.3 and Zenoss Core 5.0.3
Control Center 1.0.2 and Zenoss Core 5.0.2	Control Center 1.0.3 and Zenoss Core 5.0.3

Part I: Upgrading only Control Center

The chapters in this part provide instructions for upgrading Control Center without upgrading Zenoss Core.

Note Before upgrading only Control Center, make sure that you are upgrading to a supported combination of Control Center and Zenoss Core.

The following table identifies the supported combinations for upgrading only Control Center.

From combination	To combination
Control Center 1.1.1 and Zenoss Core 5.1.1	Control Center 1.1.2 and Zenoss Core 5.1.1

To perform an upgrade, select one of the following chapters:

Upgrading Control Center with internet access 1

This chapter includes detailed procedures for upgrading Control Center cluster hosts that have internet access. For hosts that do not have internet access, or that are configured for high-availability, use one of the other chapters in this part.

Stopping applications

This procedure stops all Control Center applications.

1. Log in to the Control Center master host as `root`, or as a user with superuser privileges.
2. Stop Zenoss Core, and then verify it is stopped.
   a. Stop Zenoss Core.

      ```
      serviced service stop Zenoss.core
      ```

   b. Verify the application is stopped.

      Repeat the following command until the `STATUS` column reads `Stopped`:

      ```
      serviced service status Zenoss.core
      ```

Upgrading the master host

This procedure upgrades the Control Center master host from version 1.1.1 to 1.1.2.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Verify that accidental upgrades of Docker are disabled.
   a. Check the Docker repository.

      ```
      grep enabled /etc/yum.repos.d/docker.repo
      ```

      If the result is `enabled=1`, perform the following substeps.
   b. Open `/etc/yum.repos.d/docker.repo` with a text editor.
   c. Change the value of the `enabled` key from `1` to `0`.
   d. Save the file and close the text editor.
3. Stop Control Center.

   ```
   systemctl stop serviced
   ```

4 Install the new version of Control Center.

```
yum --enablerepo=zenoss-stable install -y serviced-1.1.2
```

The installation preserves the existing version of the `serviced` configuration file, and installs the new one as `/etc/default/serviced.rpmnew`.

5 Delete the new Control Center configuration file.

There are no configuration file changes in this release.

```
rm /etc/default/serviced.rpmnew
```

6 Start Control Center.

```
systemctl start serviced
```

- **Single-host deployments:** You may log in to the Control Center browser interface, and then restart Zenoss Core, if desired.
- **Multi-host deployments:** Update all resource pool hosts (the next procedure).

Upgrading resource pool hosts

This procedure upgrades Control Center resource pool hosts from version 1.1.1 to 1.1.2.

Perform this procedure on each resource pool host in your deployment.

1 Log in to the resource pool host as `root`, or as a user with superuser privileges.
2 Verify that accidental upgrades of Docker are disabled.
   a Check the Docker repository.

   ```
   grep enabled /etc/yum.repos.d/docker.repo
   ```

   If the result is `enabled=1`, perform the following substeps.
   b Open `/etc/yum.repos.d/docker.repo` with a text editor.
   c Change the value of the `enabled` key from `1` to `0`.
   d Save the file and close the text editor.
3 Stop Control Center and Docker.

```
systemctl stop serviced && systemctl stop docker
```

4 Unmount the distributed file system (DFS).
   a Identify the file system specification to unmount.

   ```
   mount | awk '/serviced/ { print $1 }'
   ```

   b Unmount the DFS.
   Replace *DFS-Mount* with the file system specification returned in the previous substep:

   ```
   umount DFS-Mount
   ```

5 Install the new version of Control Center.

```
yum --enablerepo=zenoss-stable install -y serviced-1.1.2
```

The installation preserves the existing version of the `serviced` configuration file, and installs the new one as `/etc/default/serviced.rpmnew`.

6 Delete the new Control Center configuration file.
There are no configuration file changes in this release.

```
rm /etc/default/serviced.rpmnew
```

7 Start Control Center.

```
systemctl start serviced
```

Part II: Upgrading Control Center and Zenoss Core

The chapters in this part provide instructions for upgrading the combinations of Control Center and Zenoss Core shown in the following table.

From combination	To combination
Control Center 1.0.6 and Zenoss Core 5.0.6	Control Center 1.1.2 and Zenoss Core 5.1.1
Control Center 1.0.7 and Zenoss Core 5.0.7	Control Center 1.1.2 and Zenoss Core 5.1.1
Control Center 1.0.8 and Zenoss Core 5.0.8	Control Center 1.1.2 and Zenoss Core 5.1.1
Control Center 1.0.9 and Zenoss Core 5.0.9	Control Center 1.1.2 and Zenoss Core 5.1.1
Control Center 1.0.10 and Zenoss Core 5.0.10	Control Center 1.1.2 and Zenoss Core 5.1.1

For information about upgrading other combinations, see *Supported upgrade paths* on page 7.

To perform an upgrade, first review the information in *Preparing to upgrade* on page 15, and then proceed to *Upgrading Zenoss Core with internet access* on page 18. When you have completed the upgrade, proceed to *After upgrading* on page 35.

1 Preparing to upgrade

This chapter includes information about upgrading your deployment of Control Center and Zenoss Core, and procedures that prepare your deployment for the upgrade. The information and procedures in this chapter are independent of the specific upgrade procedures, which are detailed in subsequent chapters.

Storage changes

This release of Control Center includes a new storage driver for application data, named `devicemapper`. The new driver is based on *the Docker `devicemapper` storage driver*, which in turn is based on the *device mapper framework of the Linux kernel*.

The key feature of the drivers is their use of thin provisioning, a virtualization method that allocates data blocks only when data is written. (The traditional method is to allocate data blocks when a file system is created, before any data is written.) Thin provisioning enables snapshots, a time-efficient and space-efficient method of copying data, and enables making a device appear to have more physical data blocks than are actually available (as long as some blocks are unfilled). Also, thin-provisioned storage can be extended without having to move data from one physical partition to another.

For Docker and for Control Center, Logical Volume Manager (LVM) tools are used to create thin-provisioned storage, in thin pools. A thin pool includes an area for metadata (a small percentage of the total) and an area for the data itself. To simplify creating thin pools, this release of Control Center includes `serviced-storage`, a utility that calls the LVM tools. The utility may be used to create thin pools for use by Docker's `devicemapper` storage driver as well as for the Control Center `devicemapper` storage driver.

For Docker data storage, the recommended storage layout is a device mapper thin pool on one or more primary partitions. Docker's `devicemapper` storage driver may be used in `loop-lvm` mode, on loopback-mounted sparse files. However, `loop-lvm` mode is not recommended for production use, and Zenoss strongly recommends using a device mapper thin pool for Docker storage, in all deployment scenarios.

With this release, the recommended (and default) storage driver for Control Center application data is `devicemapper`. The recommended layout for application data storage is a device mapper thin pool on one or more primary partitions. Support for the `btrfs` storage driver is deprecated. The change is reflected in the Control Center configuration file, `/etc/default/serviced`: The default value of the *SERVICED_FS_TYPE* variable is `devicemapper`.

To convert application data from Btrfs file system storage to device mapper thin pool storage, the Control Center master host requires additional block storage. For more information about storage requirements in general, refer to the *Zenoss Core Planning Guide*. To determine how much data is stored in a Btrfs volume, see the next topic.

Note Control Center does not support restoring application data backups made with one storage driver to another driver. For example, a backup of Zenoss Core data that was stored on a Btrfs partition can not be restored to a `devicemapper` or `rsync` partition. In addition, snapshots and backups made with Control Center 1.0.x are not compatible with Control Center 1.1.x. Llikewise, snapshots and backups made with Control Center 1.1.x are not compatible with Control Center 1.0.x.

Control Center stores metadata for its `devicemapper` storage driver in `/opt/serviced/var/volumes`. The amount of storage required for the metadata rarely exceeds 1GB, so the directory no longer requires a separate, non-root file system (except for high-availability deployments, to mirror the metadata).

Preparing to convert the data storage driver

Perform this procedure to determine how much new storage is required to migrate application data from a Btrfs file system to a device mapper thin pool.

1. Log in to the Control Center master host as `root`, or as a user with superuser privileges.
2. Determine the amount of data stored in your Btrfs volume.

   The following command may not work for Btrfs volumes backed by RAID systems. For more information about determining the amount of used space on Btrfs volumes, refer to the *Btrfs FAQ*.

   ```
   btrfs filesystem df /opt/serviced/var/volumes
   ```

   Example result:

   ```
   Data, single: total=29.01GiB, used=23.8GiB
   System, single: total=4.00MiB, used=16.00KiB
   Metadata, single: total=264.00MiB, used=67.8MiB
   GlobalReserve, single: total=16.00MiB, used=0.00
   ```

   In this example, the used space for data and metadata total approximately 23.8G.

Use the size information to add storage to the Control Center master host. For more information, refer to the *Zenoss Core Planning Guide*.

Retaining a customized Control Center web server port

If you are using a port other than 443 for the Control Center web server, perform this procedure to retain the customization.

1. Log in to the Control Center master host as `root`, or as a user with superuser privileges.
2. Identify the port number of the Control Center web server.

   ```
   egrep '^[^#]*SERVICED_UI_PORT' /etc/default/serviced
   ```

   - If the command returns no result, the web server port is the default port, 443. Do not perform the remaining steps in this procedure.
   - If the command returns a result, the web server port is the value to which the variable is set. Perform the remaining steps in this procedure.
3. Log in to the Control Center browser interface.
4. In the **Applications** table, click **Zenoss.core**.
5. In the application title line, click **Edit Variables**.

   Figure 1: The Edit Variables dialog
6. In the **Edit Variables** dialog, add the `controlplane-port` key.

Replace *Port-Number* with the port number of the *SERVICED_UI_PORT* variable in the Control Center configuration file, displayed previously.

```
controlplane-port Port-Number
```

7 In the **Edit Variables** dialog, click **Save Changes**.

Ensuring localhost resolution

Control Center requires an entry for `localhost` in `/etc/hosts`.

1 Log in to the Control Center master host as `root`, or as a user with superuser privileges.
2 Determine whether `127.0.0.1` is mapped to `localhost`.

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

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

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

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

Important information and recommendations

With this release, Control Center supports adding tags to snapshots. Tagged snapshots are not affected by the snapshot time-to-live setting—instead, tagged snapshots persist until explicitly removed. Before the Zenoss Core upgrade begins, the upgrade script creates and tags a snapshot. You can roll back to the pre-upgrade snapshot at any time.

Note Zenoss strongly recommends checking the integrity of Zenoss Core databases before performing an upgrade or installing a ZenPack. For more information, see *Using Zenoss Toolbox* on page 40.

This release requires RHEL/CentOS 7.1 or 7.2. Zenoss strongly recommends upgrading the operating system only at the step specified in each upgrade procedure. For hosts that do not have internet access, you must create and use a local mirror, or other media, to upgrade the operating system.

The procedures in this guide include instructions for the following configuration options, which may or may not be part of your deployment:

- Isolate the Control Center master host in its own resource pool.
- Create a ZooKeeper ensemble.

Both options improve reliability and require a multi-host deployment, and are strongly recommended. For more information, refer to the *Zenoss Core Planning Guide*.

For optimum results, Zenoss recommends reviewing the upgrade procedures for your deployment before performing the upgrade.

Upgrading Zenoss Core with internet access 2

This chapter includes detailed procedures for upgrading Control Center cluster hosts that have internet access. For hosts that do not have internet access, or that are configured for high-availability, use one of the other chapters in this part.

The upgrade process includes the following general steps:

1. Stop the application or applications that Control Center is managing, and then stop Control Center on the master host.
2. Upgrade Control Center on the master host.
3. Upgrade resource pool hosts.

Upgrading Control Center

This section describes how to upgrade Control Center on all of the hosts in a Control Center cluster. The upgrade process includes the following general steps:

1. Stop the application or applications that Control Center is managing, and then stop Control Center on the master host.
2. Save copies of Zenoss Core images, if necessary.
3. Upgrade Docker on the master host, and the operating system, if necessary.
4. Upgrade Control Center on the master host.
5. Upgrade resource pool hosts.

Note Zenoss strongly recommends checking the integrity of Zenoss Core databases before proceeding. For more information, see *Using Zenoss Toolbox* on page 40.

Stopping applications

This procedure stops all Control Center applications.

1. Log in to the Control Center master host as `root`, or as a user with superuser privileges.
2. Stop Zenoss Core, and then verify it is stopped.
   a. Stop Zenoss Core.

      ```
      serviced service stop Zenoss.core
      ```

   b. Verify the application is stopped.

Repeat the following command until the `STATUS` column reads `Stopped`:

```
serviced service status Zenoss.core
```

Saving copies of Zenoss Core images

This procedure ensures that Zenoss Core images are present and up-to-date in the Control Center registry.

1. Log in to the Control Center master host as `root`, or as a user with superuser privileges.
2. Determine whether Control Center is running.

   ```
   systemctl status serviced | grep Active
   ```

   If the result does not include `active (running)`, enter the following command:

   ```
   systemctl start serviced
   ```

3. Determine whether the *SERVICED_REGISTRY* variable is set.

   ```
   egrep '^[^#]*SERVICED_REGISTRY' /etc/default/serviced
   ```

   - If the result is `SERVICED_REGISTRY=1`, stop performing this procedure, and continue to the next.
   - If the result is `SERVICED_REGISTRY=0`, or if the command returns no result, complete the remaining steps of this procedure before continuing to the next.
4. Edit the Control Center configuration file.
   a. Open `/etc/default/serviced` in a text editor.
   b. Locate the *SERVICED_REGISTRY* declaration, and then set its value to `1`.
   c. Remove the number sign character (#) from the beginning of the line.
   d. Save the file, and then close the editor.
   e. Restart Control Center.

   ```
   systemctl stop serviced && systemctl start serviced
   ```

5. Synchronize the Docker and Control Center registries.

   ```
   serviced docker sync
   ```

   The synchronization may take approximately 20-30 minutes.

Upgrading Docker

This procedure upgrades Docker from version 1.5 or 1.8.2 to 1.9.0.

Note This release of Control Center requires RHEL/CentOS 7.1 or 7.2, and this procedure includes steps for upgrading the operating system, if necessary or desired. Zenoss recommends upgrading the operating system only at the step specified in this procedure.

---

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

   ```
   systemctl stop serviced
   ```

3 Determine which version of Docker is installed.

```
rpm -qa | grep docker
```

4 **Note** If `docker-engine-1.8` is installed, skip this step.

Remove `zenoss-docker`, and enable installation of the new version of Docker.

a Remove Docker, without removing Control Center.

```
rpm -e --nodeps zenoss-docker-1.5.0-2
```

b Add the Docker repository to the host's repository list.

```
cat > /etc/yum.repos.d/docker.repo <<-EOF
[dockerrepo]
name=Docker Repository
baseurl=https://yum.dockerproject.org/repo/main/centos/7
enabled=1
gpgcheck=1
gpgkey=https://yum.dockerproject.org/gpg
EOF
```

5 **Note** If `zenoss-docker-1.5` is installed, skip this step.

Remove Docker, and then verify that the Docker repository is enabled.

a Remove Docker, without removing Control Center.

```
rpm -e --nodeps docker-engine-1.8.2
```

b Check the Docker repository.

```
grep enabled /etc/yum.repos.d/docker.repo
```

If the result is `enabled=0`, perform the following substeps.

c Open `/etc/yum.repos.d/docker.repo` with a text editor.

d Change the value of the `enabled` key from `0` to `1`.

e Save the file and close the text editor.

6 Remove the Docker data partition.

a Identify the partition where `/var/lib/docker` is mounted.

```
mount | awk '/\/var\/lib\/docker/ { print $1 }'
```

b Unmount the partition.

Replace *Partition* with the device returned in the previous substep:

```
umount Partition
```

c Erase the XFS file system on the partition.

Replace *Partition* with the device returned previously:

```
wipefs -a Partition
```

The partition is now ready for use as a device mapper thin pool.

d Open `/etc/fstab` with a text editor.

e Remove the entry for `/var/lib/docker`.

7 Upgrade the operating system, if necessary.
   a Determine which release is installed.

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

   If the result includes `7.0`, perform the following substeps.
   b Disable the `serviced` service.

```
systemctl disable serviced
```

   c Upgrade the operating system.

```
yum clean all && yum update -y
```

   d Restart the operating system.

```
reboot
```

   e Log in to the Control Center cluster host as `root`, or as a user with superuser privileges.
   f Enable the `serviced` service.

```
systemctl enable serviced
```

8 Install Docker 1.9.0, and then disable accidental upgrades.
   a Install Docker 1.9.0.

```
yum install -y docker-engine-1.9.0
```

   The result may include a warning about `serviced` requirements, which may be ignored.
   b Open `/etc/yum.repos.d/docker.repo` with a text editor.
   c Change the value of the `enabled` key from `1` to `0`.
   d Save the file and close the text editor.
9 Edit the Docker service definition.
   a Open `/lib/systemd/system/docker.service` with a text editor.
   b Add the following lines immediately after the line that contains `[Service]`.

```
EnvironmentFile=-/etc/sysconfig/docker
TimeoutSec=300
```

   c Add *OPTIONS* to the `ExecStart` defintion.
   The result should look like the following example:

```
ExecStart=/usr/bin/docker daemon $OPTIONS -H fd://
```

   d Reload the `systemd` manager configuration.

```
systemctl daemon-reload
```

   e Configure the `docker` service to start when the system starts.

```
systemctl enable docker
```

10 Install the new version of Control Center.

Control Center includes a command that simplifies the process of creating a device mapper thin pool.

```
yum --enablerepo=zenoss-stable install -y serviced
```

The installation preserves the existing version of the `serviced` configuration file, and installs the new one as `/etc/default/serviced.rpmnew`.

11 Create a device mapper thin pool for Docker data.

a Identify the primary partition for the thin pool to create.

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

b Create the thin pool.

Replace *Path-To-Device* with the path of an unused primary partition:

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

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

12 Configure and start the Docker service.

a Create variables for adding arguments to the Docker configuration file.

The `--exec-opt` argument is a workaround for *a Docker issue* on RHEL/CentOS 7.x systems.

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

```
myDriver="-s devicemapper"
myFix="--exec-opt native.cgroupdriver=cgroupfs"
myFlag="--storage-opt dm.thinpooldev"
myPool="Thin-Pool-Device"
```

b Add the arguments to the Docker configuration file.

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

c Start or restart Docker.

```
systemctl restart docker
```

The initial startup takes up to a minute, and may fail. If the startup fails, repeat the previous command.

13 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, and replace *Bridge-Netmask* with the netmask `docker` selected:

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

For example, if the bridge subnet and netmask is 172.17.0.1/16, the flags to add are `--dns=172.17.0.1 --bip=172.17.0.1/16`.

Note Leave a blank space after the end of the thin pool device name, and make sure the double quote character (") is at the end of the line.

d Restart the Docker service.

```
systemctl restart docker
```

Configuring Control Center on the master host

This procedure upgrades Control Center on the master host.

1. Log in to the Control Center master host as `root`, or as a user with superuser privileges.
2. Manage the Control Center configuration files.
   a Change directory to the configuration file directory.

   ```
   cd /etc/default
   ```

   b Rename the existing configuration file, as a backup.

   ```
   mv serviced serviced.pre-1.1.2
   ```

   c Rename the new configuration file, and then create a copy of it.

   ```
   mv serviced.rpmnew serviced.orig-1.1.2
   cp serviced.orig-1.1.2 serviced
   ```
3. Copy settings from the previous Control Center configuration file to the new one.
   a Identify the customized variables in the pre-upgrade configuration file.

   ```
   egrep '^[^#]*SERVICED' serviced.pre-1.1.2
   ```

   b Open `/etc/default/serviced` with a text editor, and then customize the same variables that were customized in the pre-upgrade configuration file.

   The following variables are deprecated, and are not needed in the new configuration file:

   - *SERVICED_REGISTRY*
   - *SERVICED_VARPATH*

   c Add *SERVICED_DOCKER_REGISTRY* to the file.

   The variable specifies the host and port at which the local Docker registry is available.

   Replace *Hostname-Or-IP* with the hostname or IP address of the Control Center master host:

   ```
   SERVICED_DOCKER_REGISTRY=Hostname-Or-IP:5000
   ```

   d Save the file and close the text editor.
4. Start Control Center and watch its log file.

   ```
   systemctl start serviced && journalctl -flu serviced
   ```

   Control Center pulls images from Docker Hub to complete its update, and then copies Zenoss Core images from its registry to the Docker library.

- **Single-host deployments:** You may log in to the Control Center browser interface, and then restart Zenoss Core, if desired.

- **Multi-host deployments:** Update all resource pool hosts (the next procedure).

Upgrading Control Center on resource pool hosts

Perform this procedure on each resource pool host in a Control Center cluster.

1. Upgrade Docker.

   For more information, see *Upgrading Docker* on page 19.
2. Manage the Control Center configuration files.
   a. Change directory to the configuration file directory.

   ```
   cd /etc/default
   ```

   b. Rename the existing configuration file, as a backup.

   ```
   mv serviced serviced.pre-1.1.2
   ```

   c. Rename the new configuration file, and then create a copy of it.

   ```
   mv serviced.rpmnew serviced.orig-1.1.2
   cp serviced.orig-1.1.2 serviced
   ```
3. Copy settings from the previous Control Center configuration file to the new one.
   a. Identify the customized variables in the pre-upgrade configuration file.

   ```
   egrep '^[^#]*SERVICED' serviced.pre-1.1.2
   ```

   b. Open `/etc/default/serviced` with a text editor, and then customize the same variables that were customized in the pre-upgrade configuration file.

   The following variables are deprecated and are not required in the new configuration file:

   - *SERVICED_REGISTRY*
   - *SERVICED_VARPATH*

   c. Add *SERVICED_DOCKER_REGISTRY* to the file.

   The variable specifies the host and port at which the Docker registry on the master host is listening.

   Replace *Hostname-Or-IP* with the hostname or IP address of the Control Center master host:

   ```
   SERVICED_DOCKER_REGISTRY=Hostname-Or-IP:5000
   ```

   d. Save the file and close the text editor.
4. Unmount the distributed file system (DFS).
   a. Identify the file system specification to unmount.

   ```
   mount | awk '/serviced/ { print $1 }'
   ```

   b. Unmount the DFS.
   Replace *DFS-Mount* with the file system specification returned in the previous substep:

   ```
   umount DFS-Mount
   ```
5. Start Control Center.

   ```
   systemctl start serviced
   ```

When all resource pool hosts are upgraded, you may log in to the Control Center browser interface, and then restart Zenoss Core, if desired.

Converting the data storage driver

At this point in the upgrade, Control Center is upgraded on the master host and on all of the resource pool hosts. You may continue using Zenoss Core to monitor your infrastructure during the initial (and longest) part of the conversion process. However, Zenoss strongly recommends running Zenoss Core in this configuration only as long as it takes to migrate Btrfs data to the new thin pool.

The procedures in this section migrate Zenoss Core data from a Btrfs volume to a device mapper thin pool volume.

Starting the conversion

To perform this procedure, you need an unused primary partition that is large enough for your data. For more information, see *Preparing to convert the data storage driver* on page 16.

This procedure converts the majority of Zenoss Core data stored in a Btrfs file system and copies it to a device mapper thin pool. This procedure may be performed while Zenoss Core is monitoring your infrastructure.

1. Log in to the Control Center master host as `root`, or as a user with superuser privileges.
2. Create a thin pool for data storage.

   Replace *Partition* with the path of an unused primary partition:

   ```
   serviced-storage create-thin-pool serviced Partition
   ```

   On success, the result includes the name of the thin pool.
3. Initialize `serviced-storage` for the `btrfs` storage driver.

   ```
   serviced-storage init /opt/serviced/var/volumes btrfs
   ```
4. Synchronize the contents of the current data storage with the new thin pool.

   This step copies the current data to the new thin pool. A small amount of metadata is stored in the temporary directory you specify, which must have approximately 1GB of available space.

   Replace *Thin-Pool-Name* with the name of the thin pool created previously, and replace *Temporary-Directory* with the path of a temporary directory (for example, `/tmp/tmpdata`):

   ```
   serviced-storage sync -c -t devicemapper \
     -o dm.thinpooldev=Thin-Pool-Name \
     /opt/serviced/var/volumes Temporary-Directory
   ```

   After a pause to compute space requirements, the `serviced-storage` command displays both detail and summary information about its work. Depending on the amount of data to convert, this step may take several hours.

Stopping applications

This procedure stops all Control Center applications.

1. Log in to the Control Center master host as `root`, or as a user with superuser privileges.
2. Stop Zenoss Core, and then verify it is stopped.
   a. Stop Zenoss Core.

      ```
      serviced service stop Zenoss.core
      ```

b Verify the application is stopped.

Repeat the following command until the `STATUS` column reads `Stopped`:

```
serviced service status Zenoss.core
```

Finalizing the conversion

This procedure completes the conversion started previously.

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

```
systemctl stop serviced
```

3. Synchronize the contents of the current data storage with the new thin pool.

   This step copies data created since the previous sync operation.

   Repeat the command you used before stopping the Control Center service:

```
serviced-storage sync -c -t devicemapper \
  -o dm.thinpooldev=Thin-Pool-Name \
  /opt/serviced/var/volumes Temporary-Directory
```

4. Unmount the Btrfs file system.

```
umount /opt/serviced/var/volumes
```

5. Unmount the export of the Btrfs file system.

```
umount -fl /exports/serviced_volumes_v2/*
systemctl restart nfs
```

6. Move the new metadata to `/opt/serviced/var/volumes`.

   a Disable the new thin pool.

   Replace *Thin-Pool-Name* with the name of the thin pool created previously, and replace *Temporary-Directory* with the path of your temporary directory:

```
serviced-storage disable Temporary-Directory \
  -o dm.thinpooldev=Thin-Pool-Name
```

   b Move the metadata contents to `/opt/serviced/var/volumes`.

   Replace *Temporary-Directory* with the path of your temporary directory:

```
mv Temporary-Directory/.devicemapper \
  Temporary-Directory/* /opt/serviced/var/volumes
```

7. Remove entries for Btrfs volumes from `/etc/fstab`, if necessary.

   a Determine whether `/etc/fstab` includes entries for Btrfs volumes.

```
awk '!/^.*#/ { print $1 }' /etc/fstab
```

   If the result includes `/opt/serviced/var/volumes` or `/exports/serviced_var_volumes_v2/`, perform the the following substeps.

   b Open `/etc/fstab` with a text editor.

c. Remove the entries that start with `/opt/serviced/var/volumes` and `/exports/serviced_var_volumes_v2/`.
d. Save the file, and then close the text editor.

8. Update the Control Center configuration file.
   a. Open `/etc/default/serviced` in a text editor.
   b. Find the *SERVICED_FS_TYPE* variable declaration, and then change the value from `btrfs` to `devicemapper`.
   c. Remove the number sign character (#) from the beginning of the line, if necessary.
   d. Add *SERVICED_DM_THINPOOLDEV* immediately after *SERVICED_FS_TYPE*. Replace *Thin-Pool-Name* with the name of the thin pool created previously,

```
SERVICED_DM_THINPOOLDEV=Thin-Pool-Name
```

   e. Save the file, and then close the editor.
9. Start the Control Center service.

```
systemctl start serviced
```

Verifying the conversion

This procedure verifies that the new data storage works properly with Zenoss Core version 1.0.x.

1. Log in to the Control Center browser interface.
2. Start Zenoss Core and all related applications.
3. Perform database integrity checks.

   For more information, see *Using Zenoss Toolbox* on page 40.
4. Create a backup.

   Previous backups are incompatible with the new storage driver.

When you are satisfied, continue with the Zenoss Core upgrade procedure.

ZooKeeper ensemble configuration

Control Center relies on *Apache ZooKeeper* to coordinate its services. The procedures in this section create a ZooKeeper ensemble of 3 nodes. To perform these procedures, you need a Control Center master host and a minimum of two resource pool hosts. Each resource pool host requires a separate primary partition for Control Center internal services, and each should have a static IP address. For more information about storage requirements, refer to the *Zenoss Core Planning Guide*.

Note Zenoss strongly recommends configuring a ZooKeeper ensemble for all production deployments.

A ZooKeeper ensemble requires a minimum of 3 nodes, and 3 nodes is sufficient for most deployments. A 5-node configuration improves failover protection during maintenance windows. Ensembles larger than 5 nodes are not necessary. An odd number of nodes is recommended, and an even number of nodes is strongly discouraged.

Note The Control Center ZooKeeper service requires consistently fast storage. Ideally, the primary partition for Control Center internal services is on a separate, high-performance device that has only one primary partition.

Stopping applications

This procedure stops all Control Center applications.

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

2 Stop Zenoss Core, and then verify it is stopped.

   a Stop Zenoss Core.

```
serviced service stop Zenoss.core
```

   b Verify the application is stopped.

   Repeat the following command until the `STATUS` column reads `Stopped`:

```
serviced service status Zenoss.core
```

Configuring the master nodes as ZooKeeper nodes

This procedure configures both Control Center master nodes as members of the ZooKeeper ensemble.

Note For accuracy, this procedure constructs Control Center configuration variables in the shell and appends them to `/etc/default/serviced`. The last step is to move the variables from the end of the file to more appropriate locations.

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

2 Create a variable for each Control Center host to include in the ZooKeeper ensemble.

   The variables are used in subsequent steps.

   Note Define the variables identically on the master host and on each resource pool host.

   Replace *Master-Host-IP* with the IP address of the Control Center master host, and replace *Pool-Host-A-IP* and *Pool-Host-B-IP* with the IP addresses of the Control Center resource pool hosts to include in the ensemble:

```
node1=Master-Host-IP
node2=Pool-Host-A-IP
node3=Pool-Host-B-IP
```

   Note ZooKeeper requires IP addresses for ensemble configuration.

3 Set the ZooKeeper node ID to `1`.

```
echo "SERVICED_ISVCS_ZOOKEEPER_ID=1" >> /etc/default/serviced
```

4 Specify the nodes in the ZooKeeper ensemble.
You may copy the following text and paste it in your console:

```
echo "SERVICED_ZK=${node1}:2181,${node2}:2181,${node3}:2181" \
  >> /etc/default/serviced
```

5 Specify the nodes in the ZooKeeper quorum.

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

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

```
q1="1@0.0.0.0:2888:3888"
q2="2@${node2}:2888:3888"
q3="3@${node3}:2888:3888"
echo "SERVICED_ISVCS_ZOOKEEPER_QUORUM=${q1},${q2},${q3}" \
  >> /etc/default/serviced
```

6 Clean up the Control Center configuration file.
   a Open `/etc/default/serviced` with a text editor.
   b Navigate to the end of the file, and cut the line that contains the *SERVICED_ZK* variable declaration at that location.

      The value of this declaration specifies 3 hosts.
   c Locate the *SERVICED_ZK* variable near the beginning of the file, and then delete the line it is on.

      The value of this declaration is just the master host.
   d Paste the *SERVICED_ZK* variable declaration from the end of the file in the location of the just-deleted declaration.
   e Navigate to the end of the file, and cut the line that contains the *SERVICED_ISVCS_ZOOKEEPER_ID* variable declaration at that location.
   f Locate the *SERVICED_ISVCS_ZOOKEEPER_ID* variable near the end of the file, and then delete the line it is on.

      This declaration is commented out.
   g Paste the *SERVICED_ISVCS_ZOOKEEPER_ID* variable declaration from the end of the file in the location of the just-deleted declaration.
   h Navigate to the end of the file, and cut the line that contains the *SERVICED_ISVCS_ZOOKEEPER_QUORUM* variable declaration at that location.
   i Locate the *SERVICED_ISVCS_ZOOKEEPER_QUORUM* variable near the end of the file, and then delete the line it is on.

      This declaration is commented out.
   j Paste the *SERVICED_ISVCS_ZOOKEEPER_QUORUM* variable declaration from the end of the file in the location of the just-deleted declaration.
   k Save the file, and then close the text editor.
7 Verify the ZooKeeper environment variables.

```
egrep '^[^#]*SERVICED' /etc/default/serviced | egrep '(_ZOO|_ZK)'
```

Configuring a resource pool host as a ZooKeeper node

To perform this procedure, you need a resource pool host with an XFS file system on a separate partition. For more information, see the next topic.

This procedure configures a ZooKeeper ensemble on a resource pool host. Repeat this procedure on each Control Center resource pool host to add to the ZooKeeper ensemble.

1 Log in to the resource pool host as `root`, or as a user with superuser privileges.
2 Create a variable for each Control Center host to include in the ZooKeeper ensemble.

   The variables are used in subsequent steps.

   Note Define the variables identically on the master host and on each resource pool host.

   Replace *Master-Host-IP* with the IP address of the Control Center master host, and replace *Pool-Host-A-IP* and *Pool-Host-B-IP* with the IP addresses of the Control Center resource pool hosts to include in the ensemble:

```
node1=Master-Host-IP
node2=Pool-Host-A-IP
node3=Pool-Host-B-IP
```

   Note ZooKeeper requires IP addresses for ensemble configuration.

3 Set the ID of this node in the ZooKeeper ensemble.

For *Pool-Host-A-IP* (**node2**), use the following command:

```
echo "SERVICED_ISVCS_ZOOKEEPER_ID=2" >> /etc/default/serviced
```

For *Pool-Host-B-IP* (**node3**), use the following command:

```
echo "SERVICED_ISVCS_ZOOKEEPER_ID=3" >> /etc/default/serviced
```

4 Specify the nodes in the ZooKeeper ensemble.
You may copy the following text and paste it in your console:

```
echo "SERVICED_ZK=${node1}:2181,${node2}:2181,${node3}:2181" \
  >> /etc/default/serviced
```

5 Specify the nodes in the ZooKeeper quorum.

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

For *Pool-Host-A-IP* (**node2**), use the following commands:

```
q1="1@${node1}:2888:3888"
q2="2@0.0.0.0:2888:3888"
q3="3@${node3}:2888:3888"
echo "SERVICED_ISVCS_ZOOKEEPER_QUORUM=${q1},${q2},${q3}" \
  >> /etc/default/serviced
```

For *Pool-Host-B-IP* (**node3**), use the following commands:

```
q1="1@${node1}:2888:3888"
q2="2@${node2}:2888:3888"
q3="3@0.0.0.0:2888:3888"
echo "SERVICED_ISVCS_ZOOKEEPER_QUORUM=${q1},${q2},${q3}" \
  >> /etc/default/serviced
```

6 Set the *SERVICED_ISVCS_START* variable, and clean up the Control Center configuration file.

a Open `/etc/default/serviced` with a text editor.

b Locate the *SERVICED_ISVCS_START* variable, and then delete all but `zookeeper` from its list of values.

c Remove the number sign character (#) from the beginning of the line.

d Navigate to the end of the file, and cut the line that contains the *SERVICED_ZK* variable declaration at that location.

The value of this declaration specifies 3 hosts.

e Locate the *SERVICED_ZK* variable near the beginning of the file, and then delete the line it is on.

The value of this declaration is just the master host.

f Paste the *SERVICED_ZK* variable declaration from the end of the file in the location of the just-deleted declaration.

g Navigate to the end of the file, and cut the line that contains the *SERVICED_ISVCS_ZOOKEEPER_ID* variable declaration at that location.

h Locate the *SERVICED_ISVCS_ZOOKEEPER_ID* variable near the end of the file, and then delete the line it is on.

This declaration is commented out.

i Paste the *SERVICED_ISVCS_ZOOKEEPER_ID* variable declaration from the end of the file in the location of the just-deleted declaration.

j Navigate to the end of the file, and cut the line that contains the *SERVICED_ISVCS_ZOOKEEPER_QUORUM* variable declaration at that location.

k Locate the *SERVICED_ISVCS_ZOOKEEPER_QUORUM* variable near the end of the file, and then delete the line it is on.

This declaration is commented out.

l Paste the *SERVICED_ISVCS_ZOOKEEPER_QUORUM* variable declaration from the end of the file in the location of the just-deleted declaration.

m Save the file, and then close the text editor.

7 Verify the ZooKeeper environment variables.

```
egrep '^[^#]*SERVICED' /etc/default/serviced \
  | egrep '(_ZOO|_ZK|_STA)'
```

8 Pull the required Control Center ZooKeeper image from the master host.

a Identify the image to pull.

```
serviced version | grep IsvcsImages
```

Example result:

```
IsvcsImages: [zenoss/serviced-isvcs:v40 zenoss/isvcs-zookeeper:v3]
```

b Pull the Control Center ZooKeeper image.

Replace *Isvcs-ZK-Image* with the name and version number of the ZooKeeper image from the previous substep:

```
docker pull Isvcs-ZK-Image
```

Creating a file system for internal services

This procedure creates an XFS file system on a primary partition. For more information about primary partitions, refer to the *Zenoss Core Planning Guide*.

Note The Control Center ZooKeeper service requires consistently fast storage. Ideally, the primary partition for Control Center internal services is on a separate, high-performance device that has only one primary partition.

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

2 Identify the target primary partition for the file system to create.

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

For more information about the output of the `lsblk` command, and about creating primary partitions, refer to the *Zenoss Core Planning Guide*.

3 Create an XFS file system.
Replace *Partition* with the path of the target primary partition:

```
mkfs -t xfs Partition
```

4 Add an entry to the `/etc/fstab` file.
Replace *File-System-Specification* with the path of the primary partition used in the previous step:

```
echo "File-System-Specification \
  /opt/serviced/var/isvcs xfs defaults 0 0" >> /etc/fstab
```

5 Create the mount point for internal services data.

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

6 Mount the file system, and then verify it mounted correctly.

```
mount -a && mount | grep isvcs
```

Example result:

```
/dev/xvdb1 on /opt/serviced/var/isvcs type xfs
 (rw,relatime,seclabel,attr2,inode64,noquota)
```

Starting a ZooKeeper ensemble

This procedure starts a ZooKeeper ensemble.

1 Log in to the Control Center master host as `root`, or as a user with superuser privileges.
2 In a separate window, log in to the second node of the ZooKeeper ensemble (*Pool-Host-A-IP*).
3 In another separate window, log in to the third node of the ZooKeeper ensemble (*Pool-Host-B-IP*).
4 On the master host, stop and start `serviced`.

```
systemctl stop serviced && systemctl start serviced
```

5 On both resource pool hosts, stop and start `serviced`.

```
systemctl stop serviced && systemctl start serviced
```

6 On the master host, check the status of the ZooKeeper ensemble.

```
echo stat | nc localhost 2181 | grep Mode
echo stat | nc Pool-Host-A-IP 2181  | grep Mode
echo stat | nc Pool-Host-B-IP 2181  | grep Mode
```

7 Optional: Log in to the Control Center browser interface, and then start Zenoss Core and related applications, if desired.

The next procedure requires stopping Zenoss Core.

Master host isolation

Control Center enables or just performs rapid recovery from application service failures. When Control Center internal services and application services share a host, application failures can limit recovery options. Zenoss strongly recommends isolating the Control Center master host in a separate resource pool.

Note A master host in a separate resource pool requires fewer RAM and CPU resources than a master host that runs application services. For more information, refer to the *Zenoss Core Planning Guide*.

To perform the steps in this section, you need a Control Center master host and a minimum of one resource pool host. If you are upgrading a single-host deployment, skip this section.

Stopping applications

This procedure stops all Control Center applications.

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

2 Stop Zenoss Core, and then verify it is stopped.
   a Stop Zenoss Core.

```
serviced service stop Zenoss.core
```

   b Verify the application is stopped.
   Repeat the following command until the `STATUS` column reads `Stopped`:

```
serviced service status Zenoss.core
```

Moving the master host to new resource pool

This procedure creates a new resource pool for the Control Center master host, and then moves the master host to the new pool.

1 Log in to the master host as `root`, or as a user with superuser privileges.
2 Create a new resource pool named `master`.

```
serviced pool add master
```

3 Display the `serviced` identifiers of all Control Center cluster hosts.
serviced host list
The first column of the results table contains the `serviced` identifiers.
4 Remove the master host from the cluster.

Replace *Serviced-ID* with the ID of the master host:

```
serviced host remove Serviced-ID
```

5 Add the master host to the cluster, in the `master` resource pool.

Replace *Hostname-Or-IP* with the hostname or IP address of the Control Center master host:

```
serviced host add --memory=0 Hostname-Or-IP:4979 master
```

If you enter a hostname, all hosts in your Control Center cluster must be able to resolve the name, either through an entry in `/etc/hosts`, or through a nameserver on your network.

When the value of the memory flag is 0, Control Center uses up to 100% of non-system RAM for its processes, even if the host's memory increases or decreases. When the value is 100 or any other value, Control Center uses a fixed amount of memory; the amount is based on the percentage of available memory when the host is added to Control Center.

6 Update the Control Center configuration file.
   a Open `/etc/default/serviced` in a text editor.
   b Locate the *SERVICED_AGENT* declaration.
   c Change the value from 1 to 0.
   d Remove the number sign character (#) from the beginning of the line, if necessary.
   e Locate the *SERVICED_MASTER_POOLID* declaration.
   f Change the value from `default` to `master`.
   g Remove the number sign character (#) from the beginning of the line, if necessary.
   h Save the file, and then close the editor.

7 Restart the Control Center service.

```
systemctl restart serviced
```

8 Optional: After a few minutes, log in to the Control Center browser interface, and then start Zenoss Core, if desired.

The next procedure requires stopping Zenoss Core.

Upgrading Zenoss Core

Perform this procedure to upgrade Zenoss Core.

1 Log in to the Control Center master host as `root`, or as a user with superuser privileges.
2 Download the primary Docker image of Zenoss Core for this release

The download takes approximately 10-20 minutes.

```
docker run -it --rm -v /root:/mnt/root \
  zenoss/core_5.1:5.1.1 rsync -a /root/5.1.x /mnt/root
```

When the download completes, the `rsync` command copies scripts that perform the upgrade to `/root/5.1.x`.

3 Pull additional images for Zenoss Core from Docker Hub.

```
/root/5.1.x/pull-docker-images.sh
```

4 Start the upgrade script.

```
/root/5.1.x/upgrade-core-5.1.x.sh
```

The upgrade takes about 25-50 minutes.

5 Restart Zenoss Core.

Some Zenoss Core services are started during the upgrade, and they need to be restarted.

```
serviced service restart Zenoss.core
```

Proceed to *After upgrading* on page 35.

3 After upgrading

This chapter includes information about what to do after upgrading your deployment of Control Center and Zenoss Core.

Preventing OpenTSDB health check failures

Deployments that were upgraded to the combination of Control Center 1.0.3 and Zenoss Core 5.0.3 in the past could experience false negative OpenTSDB health checks. This procedure determines whether your deployment is affected, and prevents the health check failures.

1. Log in to the Control Center browser interface.
2. In the **Applications** table, click **Zenoss.core**.
3. Scroll down to the **Services** table, and then click **Infrastructure** > **opentsdb**.
4. In the **Actions** column of the **Configuration Files** table, click **Edit**.
5. In the **Edit Configuration** dialog, review the `tsd.storage.hbase.*` keys.

   The following list shows the incorrect and correct values for each affected key:

   tsd.storage.hbase.data_table

   Incorrect: `{{(parent (parent .)).ID}}-tsdb`

   Correct: `{{(parent (parent (parent .))).ID}}-tsdb`

   tsd.storage.hbase.uid_table

   Incorrect: `{{(parent (parent .)).ID}}-tsdb-uid`

   Correct: `{{(parent (parent (parent .))).ID}}-tsdb-uid`

   tsd.storage.hbase.meta_table

   Incorrect: `{{(parent (parent .)).ID}}-tsdb-meta`

   Correct: `{{(parent (parent (parent .))).ID}}-tsdb-meta`

   tsd.storage.hbase.tree_table

   Incorrect: `{{(parent (parent .)).ID}}-tsdb-tree`

   Correct: `{{(parent (parent (parent .))).ID}}-tsdb-tree`

   - If the values cause health check failures, perform the remaining steps of this procedure.
   - If the values do not cause health check failures, no additional steps are required.
6. Update the `tsd.storage.hbase.*` keys.

   Use the correct values listed in the previous step.
7. In the **Edit Configuration** dialog, click the **Save** button.

Checking ZooKeeper quorum keys

Some upgrades may have missed an upgrade to an OpenTSDB key. This procedure determines whether your deployment is affected, and corrects the oversight.

1. Log in to the Control Center browser interface.
2. In the **Applications** table, click **Zenoss.core**.
3. Scroll down to the **Services** table, and then click **Infrastructure** > **opentsdb**.
4. In the **Edit Configuration** dialog, review the value of the `tsd.storage.hbase.zk_quroum` key.

   In the following example, the value includes `localhost`:

   ```
   tsd.storage.hbase.zk_quorum = {{with $zks := (child (child (parent
    (parent .)) "HBase") "ZooKeeper").Instances }}{{ range (each
    $zks) }}localhost:{{plus 2181 .}}{{if ne (plus 1 .) $zks}},{{end}}
   {{end}}{{end}}
   ```

   The correct value of the `tsd.storage.hbase.zk_quroum` key replaces `localhost` with `127.0.0.1`.

   - If the value includes `localhost`, perform the remaining steps of this procedure.
   - If the value does not include `localhost`, no additional steps are required.
5. Update the value of the `tsd.storage.hbase.zk_quroum` key.

   Replace `localhost` with `127.0.0.1`.
6. In the **Edit Configuration** dialog, click the **Save** button.

Correcting a CentralQuery configuration file

Some upgrades may have an incorrect configuration for logging in a CentralQuery configuration file . This procedure determines whether your deployment is affected, and corrects the oversight.

1. Log in to the Control Center browser interface.
2. In the **Applications** table, click **Zenoss.core**.
3. Scroll down to the **Services** table, and then click **Zenoss** > **Metrics** > **CentralQuery**.
4. In the **Actions** column of the **Configuration Files** table, click the **Edit** control of the `/opt/zenoss/etc/central-query/configuration.yaml` file.
5. In the **Edit Configuration** dialog, review the declaration for `logging`.

   The following example shows the correct declaration:

   ```
   logging:
     level: INFO
     loggers:
       "org.zenoss": INFO
   ```

   - If the declaration for logging is not correct, perform the remaining steps of this procedure.
   - If the declaration for logging is correct, no additional steps are required.
6. Update the declaration for `logging`.

   Use the declaration in the following example:

   ```
   logging:
     level: INFO
     loggers:
       "org.zenoss": INFO
   ```
7. In the **Edit Configuration** dialog, click the **Save** button.

Adding ZooKeeper keys

Some upgrades may be missing two key-value pairs in a ZooKeeper configuration file. This procedure determines whether your deployment is affected, and corrects the oversight.

1. Log in to the Control Center browser interface.
2. In the **Applications** table, click **Zenoss.core**.
3. Scroll down to the **Services** table, and then click **Infrastructure** > **HBase** > **ZooKeeper**.
4. In the **Actions** column of the **Configuration Files** table, click the **Edit** control of the `/etc/zookeeper.cfg` file.
5. In the **Edit Configuration** dialog, review the contents.

   The following example shows the key-value pairs that may be missing:

   ```
   autopurge.snapRetainCount=3
   autopurge.purgeInterval=1
   ```

   - If the key-value pairs are missing, perform the remaining steps of this procedure.
   - If the key-value pairs are present, no additional steps are required.
6. Update the contents of the configuration file.

   Add the content of the following example:

   ```
   autopurge.snapRetainCount=3
   autopurge.purgeInterval=1
   ```
7. In the **Edit Configuration** dialog, click the **Save** button.

Increase Zope server threads

Some upgrades may have the value of the Zope server threads key set to 1 instead of the default, 4. This procedure determines whether your deployment is affected, and corrects the oversight.

1. Log in to the Control Center browser interface.
2. In the **Applications** table, click **Zenoss.core**.
3. Scroll down to the **Services** table, and then click **Zenoss** > **User Interface** > **Zope**.
4. In the **Actions** column of the **Configuration Files** table, click the **Edit** control of the `/opt/zenoss/etc/zope.conf` file.
5. In the **Edit Configuration** dialog, determine whether the `zserver-threads` key is set.

   Typically, the `zserver-threads` key is about 20% past the beginning of the file.

   - If the `zserver-threads` key is set and the value is 1, perform the remaining steps of this procedure.
   - If the `zserver-threads` key is set and the value is 4, no additional steps are required.
   - If the `zserver-threads` key is not set, the value defaults to 4, and no additional steps are required.
6. Change the value of the `zserver-threads` key from 1 to 4.
7. In the **Edit Configuration** dialog, click the **Save** button.

Updating the daily maintenance script

The `/etc/cron.daily/serviced` script installed with previous releases of Control Center invokes the `/opt/serviced/bin/serviced-container-cleanup` script to maintain the Btrfs file system of `/var/lib/docker`. If `/etc/cron.daily/serviced` includes the cleanup script invocation, remove the invocation.

Creating a weekly maintenance script

The Zenoss Core databases require regular maintenance to perform optimally. This procedure creates a script for `cron` to run once a week, to perform the required maintenance.

1. Log in to the Control Center master host as `root`, or as a user with superuser privileges.
2. Create a shell script for `cron` to invoke.
   a. Open `/etc/cron.weekly/serviced` with a text editor.
      The file is empty.
   b. Add the following content to the file.

```
#!/bin/sh

/bin/serviced service run zope zenossdbpack
```

   c. Save the file, and then close the text editor.
3. Set file permissions.

```
chmod 0755 /etc/cron.weekly/serviced
```

Deleting the pre-upgrade snapshot

Before the Zenoss Core upgrade begins, the upgrade script creates and tags a snapshot of the system. Tagged snapshots persist until they are explicitly removed. When you are satisfied the new release is working properly, delete the pre-upgrade snapshot.

1. Log in to the Control Center master host as `root`, or as a user with superuser privileges.
2. Display a list of all Control Center snapshots, with their tags.

```
serviced snapshot list -t
```

   Example result:

```
Snapshot                          Description  Tags
xm5mtezbyo2_20160211-220535.480                preupgrade-resmgr-5.1.1
```

3. Delete the pre-upgrade snapshot.

   Replace *Snapshot-ID* with the identifier of the pre-upgrade snapshot returned in the previous step:

```
serviced snapshot remove Snapshot-ID
```

Deleting the pre-upgrade images

This upgrade retains Zenoss Core images in the Docker library. When you are satisfied the new release is working properly, delete the old images.

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

```
docker images
```

3. Remove unneeded images.

Replace *Image-ID* with the identifier of the image to remove:

```
docker rmi Image-ID
```

Repeat this procedure on each resource pool host.

Deleting the pre-upgrade registry

This release of Control Center includes a new registry for application images. When you are satisfied the new release is working properly, delete the old registry.

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

   ```
   cd /opt/serviced/var/isvcs/docker-registry
   ```
3. Check the sizes of the two registries.

   ```
   du -sh ./*
   ```

   Each registry contains about 2GB or 3GB of images.
4. Delete the pre-upgrade registry.

   ```
   rm -rf ./registry
   ```

Using Zenoss Toolbox A

This appendix describes how to install and use Zenoss Toolbox.

Zenoss Toolbox tools

The Zenoss Toolbox tools examine key Zenoss Core components for common issues affecting data integrity. Zenoss recommends running the following tools, in order, before upgrading Zenoss Core:

1. The *zodbscan* tool quickly scans the Zope Object Database (ZODB) to provide a preliminary indication of the health of the database, and to determine whether the database needs to be compressed with `zenossdbpack` before upgrading.
2. The *findposkeyerror* tool checks objects and their relationships, and provides options for fixing errors.
3. The *zenrelationscan* tool checks only ZenRelations between objects.
4. The *zencatalogscan* tool checks ZODB object catalogs, which speed up web interface access.

The tools are run inside a Zope container, and the log files for each command are found in `$ZENHOME/log/toolbox`.

Downloading Zenoss Toolbox with internet access

This procedure describes how to download Zenoss Toolbox to a Control Center master host that has internet access.

1. Log in to the Control Center master host as `root`, or as a user with superuser privileges.
2. Create a temporary directory, and change the current working directory to the temporary directory.

   The directory must be local (not mounted).

   ```
   mkdir /tmp/toolbox && cd /tmp/toolbox
   ```
3. Download Zenoss Toolbox.

   ```
   myUrl=https://github.com/zenoss/zenoss.toolbox/archive/master.zip
   curl -sL --insecure -o master.zip $myUrl
   ```
4. Change the directory and file permissions.

   The directory and file must be readable, writable, and executable by all users.

   ```
   chmod -R 777 /tmp/toolbox
   ```

Installing Zenoss Toolbox

This procedure describes how to install Zenoss Toolbox for use in Control Center Zope containers.

1. Log in to the Control Center master host as `root`, or as a user with superuser privileges.
2. Start a shell as the `zenoss` user in a Zope container.
   a. Change directory to the tempoarary location of the Zenoss Toolbox `master.zip` file.

```
cd /tmp/toolbox
```

   b. Start an interactive shell in a Zope container and save a snapshot named `InstallZenossToolbox`.

```
mySnap=InstallZenossToolbox
serviced service shell -i -s $mySnap zope bash
```

   c. Switch user to `zenoss`.

```
su - zenoss
```

3. Install Zenoss Toolbox, and then exit the container.
   a. Install Zenoss Toolbox.

```
easy_install /mnt/pwd/master.zip
```

   b. Exit the `zenoss` user account.

```
exit
```

   c. Exit the Zope container.

```
exit
```

4. Commit the named snapshot.

```
serviced snapshot commit $mySnap
```

5. Restart the Zope service.

```
serviced service restart zope
```

Running Zenoss Toolbox tools

1. Log in to the Control Center master host as a user with `serviced` CLI privileges.
2. Start an interactive session in a Zope container.

```
serviced service attach zope/0
```

3. Switch user to `zenoss`.

```
su - zenoss
```

4. Run the Zenoss Toolbox tools, in order.
   For more information about the tools, see *Zenoss Toolbox tools* on page 40.

5 Exit the `zenoss` user account.

```
exit
```

6 Exit the Zope container.

```
exit
```