



Zenoss Core Installation Guide

Release 5.0.0

Zenoss, Inc.

www.zenoss.com

Zenoss Core Installation Guide

Copyright © 2015 Zenoss, Inc. All rights reserved.

Zenoss and the Zenoss logo are trademarks or registered trademarks of Zenoss, Inc., in the United States and other countries. All other trademarks, logos, and service marks are the property of Zenoss or other third parties. Use of these marks is prohibited without the express written consent of Zenoss, Inc., or the third-party owner.

Flash is a registered trademark of Adobe Systems Incorporated.

Oracle, the Oracle logo, Java, and MySQL are registered trademarks of the Oracle Corporation and/or its affiliates.

Linux is a registered trademark of Linus Torvalds.

RabbitMQ is a trademark of VMware, Inc.

SNMP Informant is a trademark of Garth K. Williams (Informant Systems, Inc.).

Sybase is a registered trademark of Sybase, Inc.

Tomcat is a trademark of the Apache Software Foundation.

VMware is a registered trademark or trademark of VMware, Inc. in the United States and/or other jurisdictions.

Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

All other companies and products mentioned are trademarks and property of their respective owners.

Part Number: 1051.15.030

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

Contents

Preface.....	4
 Chapter 1: Planning your deployment.....	5
Introduction to Zenoss Control Center.....	5
Deployment considerations.....	5
Hardware requirements.....	6
Operating system requirements.....	8
Security considerations.....	9
Packaging considerations.....	10
Supported clients and browsers.....	11
 Chapter 2: Installing on RHEL or CentOS hosts.....	12
Preparing the master host.....	12
Installing Zenoss Control Center on the master host.....	14
Preparing a resource pool host.....	16
Installing Zenoss Control Center on a resource pool host.....	17
 Chapter 3: Installing on Ubuntu hosts.....	20
Preparing the master host.....	20
Installing Zenoss Control Center on the master host.....	22
Preparing a resource pool host.....	23
Installing Zenoss Control Center on a resource pool host.....	25
 Chapter 4: Starting Zenoss Core.....	27
Enabling access to the Zenoss Control Center web interface.....	27
Deploying the Zenoss Core application.....	29
Changing the default passwords of application services.....	33
Starting the Zenoss Core application.....	33
 Appendix A: Mirroring repositories.....	35
Mirroring the Zenoss Docker Hub repository.....	35

Preface

Zenoss Core Installation Guide provides detailed instructions for installing Zenoss Core (Zenoss Core).

Related publications

Title	Description
<i>Zenoss Core Administration Guide</i>	Provides an overview of Zenoss Core architecture and features, as well as procedures and examples to help use the system.
<i>Zenoss Core Release Notes</i>	Describes known issues, fixed issues, and late-breaking information not already provided in the published documentation set.

Additional information and comments

If you have technical questions about this product that are not answered in this guide, visit the [Zenoss Support](#) site.

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.

1

Planning your deployment

This release features Zenoss Control Center, the open-source, application service orchestrator based on [Docker](#) and built by Zenoss. Zenoss Control Center greatly simplifies the installation, deployment, and management of Zenoss Core.

This chapter describes the platform considerations and requirements of Zenoss Control Center and Zenoss Core, for planning purposes. No procedures are included.

Introduction to Zenoss Control Center

A Zenoss Control Center cluster contains one or more resource pools—collections of compute, network, and storage resources (real or virtual hosts). The default resource pool includes the Zenoss Control Center master host. The master host schedules services to run on specific hosts, and maintains the local Docker repository and the volume of application data. Volume drivers create and manage snapshots, and perform rollbacks, so that Docker images and application data stay in sync.

Zenoss Control Center maps services to resource pools rather than to individual hosts. Also, Zenoss Control Center provides a distributed file system for application data. As a result, when a host in a resource pool fails, Zenoss Control Center simply starts a new instance of the service on a different host in the pool.

Deployment considerations

The features of Zenoss Control Center in this release affect deployments of Zenoss Core in the following ways.

- All hosts in a resource pool must have identical hardware resources (real or virtual). This ensures that each pool host can support the services assigned to the pool if all hosts but one fail.
- This release supports Docker version 1.3.3 only. If a more recent version of Docker is installed on a host, the installation process reverts it to 1.3.3.
- The default resource pool typically includes the stateful services of Zenoss Core. The performance of Zenoss Core is enhanced when the hosts in the default pool are equipped with high-performance storage subsystems. Since virtual storage subsystems are not supported, the highest-performance choice is solid-state drives.
- Docker supports running identical containers under a variety of Linux operating systems. Zenoss Control Center supports the Ubuntu, RHEL, and CentOS distributions, but combining hosts that are running different operating systems in a resource pool is currently untested and unsupported.
- Resource pools other than the default pool are known as distributed resource pools. You may create any number of distributed resource pools, and assign specific services to them. A common choice is to create distributed resource pools in different subnets or security zones, in order to deploy Zenoss Core collector instances closer to the monitored devices on the subnets.

- Zenoss Core collector services are stateless, so their storage requirements are very lightweight, compared to stateful services like MariaDB.
- The internal services of Zenoss Control Center include Logstash, a Docker repository, and OpenTSDB. If these services are already established in your environment, you may replace their default endpoints with the endpoints of your existing services.
- Currently, Zenoss Control Center does not provide failover of the master host. For failover support, install a [Red Hat cluster](#) in active/passive mode.
- By default, Zenoss Control Center deletes its log files after one day. You may increase the value by setting the `SERVICED_LOGSTASH_MAX_DAYS` variable in the defaults file, `/etc/default/serviced`.
- The default volume type for application data is `rsync`, which is sufficient for development deployments of Zenoss Control Center only. For all other uses, Zenoss supports only [Btrfs](#). The installation procedures include example steps for creating Btrfs file systems.

Hardware requirements

Zenoss Core and Zenoss Control Center require real or virtual hosts that implement the 64-bit version of the x86 instruction set, and support one of the required operating systems.

The default resource pool (containing the Zenoss Control Center master host) needs identical hosts that meet the following, minimum requirements:

- 4 CPU cores (64-bit only; real or virtual)
- 20GB RAM minimum
- 1 network interface controller (must support TCP/IP)
- 55GB to 80GB local storage (network file systems not supported)

The primary consumers of disk space on a master host are the local Docker registry and Zenoss Core data. The amount of disk space required by each consumer depends on the purpose of the deployment. For production deployments, application data requires relatively more space. For development or testing deployments, the Docker registry may require more space. In either deployment scenario, the amount of disk space dedicated to each consumer should not be less than 25GB, and the space must be local.

Note Copy-on-write file systems like Btrfs typically experience unrecoverable errors when the underlying disk space reaches capacity. Zenoss recommends allocating large reserves of disk space to Btrfs file systems.

Also, the master host requires adequate local or remote storage for backup and restoration files. The Zenoss Control Center web interface stores backup files in `$SERVICED_HOME/backups`, and restoration files in `$SERVICED_HOME/restore`. (The default value of `SERVICED_HOME` is `/opt/serviced`.) A typical backup file size is several gigabytes, and restoration file sizes are equivalent. The backup and restoration storage paths may be mounted to a remote file server.

The hosts in distributed resource pools need enough RAM and CPU resources to support the services assigned to the pool. Storage requirements also vary, but the minimum is approximately 55GB. The hardware specifications of all hosts in distributed resource pools should be identical.

The Docker, Zenoss Control Center, and Zenoss Core packages require approximately 5MB of disk space.

Note An under-resourced master host will not function properly. Please do not deploy Zenoss Control Center and Zenoss Core on a master host that does not meet the minimum requirements.

Finally, the network latency among all hosts in a resource pool should be less than 5 milliseconds.

Docker and Zenoss Control Center data storage

Internally, Docker uses an abstraction for data storage (volumes), and provides drivers that support specific file system types.

- On Ubuntu platforms, the preferred driver is `aufs`. The mainline Linux kernel does not support `aufs`, so Docker includes the necessary support in its own package. No separate primary or logical partition is needed for Docker data storage on Ubuntu platforms.
- On RHEL/CentOS platforms, the preferred driver is `btrfs`, which is supported in the mainline kernel. A separate primary or logical partition is needed for Docker data storage.

Docker stores its data in `/var/lib/docker`.

Similarly, Zenoss Control Center uses the volumes abstraction for data storage, and provides `rsync` and `btrfs` drivers.

- For development deployments only, the `rsync` driver may be used.
- For all other deployment scenarios, Zenoss supports only the `btrfs` driver.

Zenoss Control Center stores application data in `/opt/serviced/var`.

Disk partitioning examples

The following tables provide example partitioning schemes that support the Docker and Zenoss Control Center data storage requirements. All sizes are suggested minimums.

The following table shows an example master host with separate partitions for Btrfs file systems. Optionally, you may create one large partition for Btrfs, and then create subvolumes for the two mount points.

Table 1: RHEL/CentOS master host (default pool)

Mount Point	Type	Size
<code>/</code>	Default (xfs)	15GB
<code>(none)</code>	swap	15GB
<code>/var/lib/docker</code>	btrfs	25GB
<code>/opt/serviced/var</code>	btrfs	25GB

Table 2: RHEL/CentOS resource pool host (default pool)

Mount Point	Type	Size
<code>/</code>	Default (xfs)	15GB
<code>(none)</code>	swap	15GB
<code>/var/lib/docker</code>	btrfs	25GB

Table 3: Ubuntu master host (default pool)

Mount Point	Type	Size
<code>/</code>	Default (ext4)	45GB
<code>(none)</code>	swap	15GB

Mount Point	Type	Size
/opt/serviced/var	btrfs	25GB

Table 4: Ubuntu resource pool host (default pool)

Mount Point	Type	Size
/	Default (ext4)	45GB
(none)	swap	15GB

Operating system requirements

Zenoss Control Center and Zenoss Core require the 64-bit version of Ubuntu 14.04 LTS, Red Hat Enterprise Linux (RHEL) 7.0, or CentOS 7.0. The versions of the Linux kernel included in these releases, and all subsequent updates, are supported. However, Zenoss encourages you to keep the kernel up-to-date.

Zenoss Control Center and Zenoss Core are tested on operating system platforms that are installed and configured with standard options.

- Ubuntu: The Ubuntu Server 14.04 LTS distribution provides only one configuration. In addition to the packages included in that configuration, the `openssh-server` and `btrfs-tools` packages are required.
- RHEL/CentOS: The 7.0 distributions provide a variety of server configurations. Docker and Zenoss Control Center require no specific package groups, and the Minimal Install configuration is tested and supported.

In multi-host configurations, Zenoss Control Center relies on the system clock to synchronize its actions. The installation procedures include steps to add the Network Time Protocol (NTP) daemon to all hosts. By default, the NTP daemon synchronizes the system clock by communicating with standard time servers available on the internet. You may configure the daemon to use a timeserver in your environment, instead.

Networking requirements

On startup, Docker creates the `docker0` virtual interface and selects an unused IP address and subnet (typically, 172.17.42.1/16) to assign to the interface. The virtual interface is used as a virtual Ethernet bridge, and automatically forwards packets among real and virtual interfaces attached to it. The host and all of its containers communicate among one another through this virtual bridge.

Docker can only check directly-connected routes, so the subnet it chooses for the virtual bridge may be inappropriate for your environment. To customize the virtual bridge subnet, refer to Docker's [advanced network configuration](#) article.

On RHEL/CentOS systems, [Firewalld](#) can conflict with Docker, and therefore, Zenoss Control Center and Zenoss Core. The following interactions illustrate the conflicts:

- The `firewalld` daemon removes the `DOCKER` chain from `iptables` when it starts or restarts.
- Under `systemd`, `firewalld` is started before Docker. However, if you start or restart `firewalld` while Docker is running, you need to restart Docker.

If you are using `Firewalld`, please ensure that it does not conflict with Docker.

If you are not using `Firewalld`, your firewall settings may still prevent communications over the Docker virtual bridge. This occurs when the `iptables` INPUT rules restrict most traffic. To ensure that the bridge works properly, append an INPUT rule to your `iptables` configuration that allows traffic on the bridge subnet. For

example, if `docker0` is bound to 172.17.42.1/16, then the following, non-specific command ensures that the bridge works.

```
iptables -A INPUT -d 172.17.0.0/16 -j ACCEPT
```

Note The preceding command is only an example. Please consult your networking specialist before modifying your `iptables` configuration.

Additional requirements

Zenoss Control Center requires a 16-bit, private IPv4 network for communications among hosts in its resource pools. The default network is 10.3/16. If this network is already in use in your environment, you may select any valid IPv4 16-bit address space during installation.

This release of Zenoss Control Center relies on Network File System (NFS) for its distributed file system implementation. For this reason, hosts in a Zenoss Control Center cluster may not run a general-purpose NFS server.

All hosts in Zenoss Control Center resource pools must

- be able to resolve the hostnames of all other resource pool hosts to IPv4 addresses
- respond with an IPv4 address other than 127.x.x.x when `ping Hostname` is invoked
- return a unique result from the `hostid` command

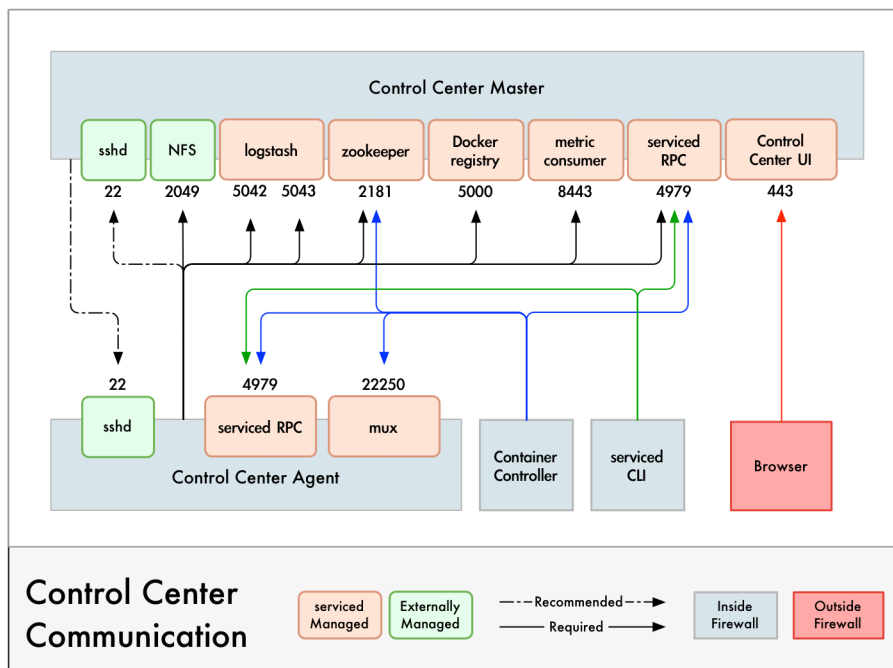
Security considerations

During installation, Zenoss Control Center has no knowledge of the port requirements of the applications it is to manage, so the installation procedure includes disabling the firewall. After both Zenoss Control Center and Zenoss Core are installed, you may close unused ports.

Zenoss Control Center includes a virtual multiplexer (mux), to aggregate the UDP and TCP traffic among the services it manages. The aggregation is opaque to services, and mux traffic is encrypted when it travels among containers on remote hosts. (Traffic among containers on the same host is not encrypted.) The mux, along with the distributed file system, enables Zenoss Control Center to deploy services to any pool host, rapidly. The mux also reduces the number of open ports required on a Zenoss Control Center host to a predictable set.

The following illustration identifies the ports that Zenoss Control Center requires its operations. All of the ports are configurable. All traffic is TCP.

Note Zenoss Control Center relies on the system clock to synchronize its actions, and indirectly, NTP, to synchronize clocks among multiple hosts. In the default configuration of `ntpd`, the firewalls of master and resource pool hosts need to support an incoming UDP connection on port 123.

Figure 1: Port requirements for Zenoss Control Center hosts

Additional considerations

- To gain access to the Zenoss Control Center web interface, users must have login accounts on the Zenoss Control Center master host. (Pluggable Authentication Modules (PAM) is supported.) By default, the users must be members of the `sudo` group (Ubuntu hosts) or the `wheel` group (RHEL/CentOS hosts). The default group may be changed by setting the `SERVICED_ADMIN_GROUP` variable, and the replacement group does not need superuser privileges.
- The `serviced` startup script sets the hard and soft open files limit to 1048576, but does not modify the `/etc/sysconfig/limits.conf` file.
- Zenoss Control Center does not support *Security Enhanced Linux* in enforcing mode. The installation procedures include steps to set the mode to permissive.
- On RHEL/CentOS systems, the `FirewallD` service can conflict with Docker, and therefore, Zenoss Control Center and Zenoss Core. For more information, see *Networking requirements* on page 8.

Packaging considerations

Zenoss Control Center is designed to support any application that includes one or more services in Docker containers. A service definition template contains the specifications of application services, in JSON format. The definition of each service includes the IDs of the Docker images needed to run the service.

Zenoss Control Center, and the service definition templates for Zenoss Core, are distributed as Debian (`apt`) and Redhat (`yum/rpm`) packages. The packages are available at public repositories maintained by Zenoss. The Docker images that Zenoss Core requires are available at the Zenoss *Docker Hub* repository. So, the default installation process requires internet access. However, all of the repositories may be mirrored, so offline installations are supported as well.

Note The Docker images for Zenoss Core are available in a public Zenoss repository at Docker Hub. No special permissions are required to pull the images.

Supported clients and browsers

The client operating systems and web browser combinations supported in this release.

- All browsers must have Adobe® Flash® Player 11 installed, or a more recent version.
- Compatibility mode is not supported in Internet Explorer.

Client OS	Supported Browsers
Windows 7 and 8.1	Internet Explorer 11 (enterprise mode is supported)
	Internet Explorer 10
	Firefox 30 and above
	Chrome 30 and above
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

2

Installing on RHEL or CentOS hosts

The procedures in this chapter install Zenoss Control Center and Zenoss Core on one or more Red Hat Enterprise Linux (RHEL) 7.0 or CentOS 7.0 hosts. Please review the information in [Planning your deployment](#) on page 5 before performing these procedures.

Note The procedures in this chapter include example steps to create *Btrfs* file systems. The steps may not be appropriate for your environment. To create Btrfs file systems in a different way, please refer to your operating system documentation.

Preparing the master host

Perform this procedure to prepare a Red Hat Enterprise Linux (RHEL) 7.0 or CentOS 7.0 host to function as the Zenoss Control Center master host.

- 1 Log in to the master host as `root`.
- 2 Verify that the host implements the 64-bit version of the x86 instruction set.

```
uname -m
```

- If the output is `x86_64`, the architecture is 64-bit. Proceed to the next step
 - If the output is `i386/i486/i586/i686`, the architecture is 32-bit. Stop this procedure and select a different host.
- 3 Disable the firewall.

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

- 4 Enable persistent storage for log files, if desired.

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

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

- 5 Create two Btrfs file systems, or one Btrfs file system with two subvolumes.

Note The following example procedure creates two file systems on two partitions, `/dev/sdb1` and `/dev/sdb2`. Modify the steps or values in this procedure as required for your environment.

- a** Create mount points for the Btrfs file systems.

```
mkdir -p /var/lib/docker /opt/serviced/var
```

- b** Create the file systems.

Replace the values of the partition variables with device paths that are correct for your environment.

```
DOCKER_PART=/dev/sdb1
APP_PART=/dev/sdb2
mkfs -t btrfs $DOCKER_PART
mkfs -t btrfs $APP_PART
```

- c** Add the new file systems to the `/etc/fstab` file.

```
echo "$DOCKER_PART /var/lib/docker btrfs rw,noatime,nodatacow 0 0" \
>> /etc/fstab
echo "$APP_PART /opt/serviced/var btrfs rw,noatime,nodatacow 0 0" \
>> /etc/fstab
```

- d** Mount the file systems, and then verify they mounted correctly.

```
mount -a
mount | grep 'docker|serviced'
```

The output of the preceding command should be similar to the following example.

```
/dev/sdb1 on /var/lib/docker type btrfs
(rw,noatime,seclabel,nodatasum,nodatacow,ssd,space_cache)
/dev/sdb2 on /opt/serviced/var type btrfs
(rw,noatime,seclabel,nodatasum,nodatacow,ssd,space_cache)
```

- 6** Disable Security-Enhanced Linux (SELinux), if installed.

- a** Determine whether SELinux is installed.

```
test -f /etc/selinux/config && grep '^SELINUX=' /etc/selinux/config
```

If the preceding commands return a result, SELinux is installed.

- b** Set the operating mode to disabled, and confirm the setting.

```
EXT=$(date +%j-%H%M%S")
sudo sed -i.${EXT} -e 's/^SELINUX=.*SELINUX=disabled/g' \
/etc/selinux/config && \
grep '^SELINUX=' /etc/selinux/config
```

- 7** Download and install the Zenoss repository package.

```
rpm -ivh http://get.zenoss.io/yum/zenoss-repo-1-1.x86_64.rpm
yum clean all
```

- 8** Install and start the Dnsmasq package.

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

- 9** Install the NTP package and start ntpd.

```
yum install -y ntp && systemctl enable ntpd && systemctl start ntpd
```

Note In testing, an unresolved issue prevents `ntpd` from restarting after a reboot. If this occurs in your environment, you must restart it manually.

- 10 Reboot the host.

```
sudo reboot
```

Proceed to the next procedure, [Installing Zenoss Control Center on the master host](#) on page 14.

Installing Zenoss Control Center on the master host

Perform this procedure to install Zenoss Control Center and Zenoss Core on the master host.

- 1 Log in to the master host as `root`.
- 2 Install Zenoss Control Center, Zenoss Core, and Docker, and then start Docker.

```
yum --enablerepo=zenoss-testing install -y zenoss-core-service
systemctl start docker
```

- 3 Configure and restart Docker.

- a Identify the IPv4 address and subnet Docker has selected for its virtual Ethernet bridge.

```
ip addr | grep -A 2 'docker0:' | grep inet
```

Note Typically, the address and subnet is 172.17.42.1/16. For more information about changing the selection, refer to Docker's [advanced network configuration](#) article.

- b Add the Btrfs and DNS flags to the Docker startup options.

If you change the virtual bridge subnet, replace the IP address in the following command.

```
echo 'DOCKER_OPTS="-s btrfs --dns=172.17.42.1"' \
>> /etc/sysconfig/docker
```

- c Add the current user to the Docker group.

```
usermod -aG docker $USER
```

- d Stop and restart Docker.

```
systemctl stop docker && systemctl start docker
```

- 4 Change the volume type for application data.

The `/etc/default/serviced` file contains variables that define the character of an instance of Zenoss Control Center. For more information about `serviced` configuration options, refer to *Zenoss Control Center Guide*.

The following `sed` command changes the value of the `SERVICED_FS_TYPE` variable from the default, `rsync`, to `btrfs`.

```
EXT=$(date +"%j-%H%M%S")
sudo sed -i.${EXT} \
-e 's|^#[^S]*\ (SERVICED_FS_TYPE=) .*$|\1btrfs|' \
/etc/default/serviced
```

- 5 Optional: Configure the master host for a multi-host deployment, if desired.

The default values in `/etc/default/serviced` configure Zenoss Control Center for a single-host deployment. To enable a multi-host deployment, change the following variables.

HOME

The path `docker` uses to locate the `.dockercfg` authentication file. Docker Hub credentials are stored in the file.

SERVICED_REGISTRY

Determines whether `serviced` uses a local registry to store Docker images. Set the value to 1, true.

SERVICED_AGENT

Determines whether a `serviced` instance acts as a resource pool host. A resource pool host runs application services scheduled for the resource pool to which it belongs. Set the value to 1, true.

SERVICED_MASTER

Determines whether a `serviced` instance acts as the Zenoss Control Center master host. The master host runs the application services scheduler and other internal services, including the web server for the Zenoss Control Center web interface. A `serviced` instance may be configured as both an agent and a master. Set the value to 1, true.

The following commands make the required edits to `/etc/default/serviced`.

```
EXT=$(date +%j-%H%M%S")
sudo sed -i.${EXT} -e 's|^#[^H]*\ (HOME=/root\)|\1|' \
-e 's|^#[^S]*\ (SERVICED_REGISTRY=)\.|\11|' \
-e 's|^#[^S]*\ (SERVICED_AGENT=)\.|\11|' \
-e 's|^#[^S]*\ (SERVICED_MASTER=)\.|\11|' \
/etc/default/serviced
```

- 6 Optional: Specify an alternate private subnet for Zenoss Control Center, if necessary.

By default, Zenoss Control Center uses the 10.3/16 private subnet for virtual IP addresses. If your environment already uses 10.3/16 for other purposes, select an unused subnet for Zenoss Control Center, and add it to the `/etc/default/serviced` file.

Note [RFC 1918](#) restricts private networks to the 10.0/24, 172.16/20, and 192.168/16 address spaces. However, Zenoss Control Center accepts any valid, 16-bit, IPv4 address space for its private network.

For example, to set the private subnet for Zenoss Control Center to 10.20/16, uncomment the `SERVICED_VIRTUAL_ADDRESS_SUBNET` variable, and then set its value to 10.20.

The following commands make the required edits to `/etc/default/serviced`.

```
SUBNET=10.20
EXT=$(date +%j-%H%M%S")
test ! -z "${SUBNET}" && \
sudo sed -i.${EXT} -e \
's|^#[^S]*\ (SERVICED_VIRTUAL_ADDRESS_SUBNET=)\.|\11|' \
/etc/default/serviced
```

Note All hosts in a Zenoss Control Center deployment must have the same value for the `SERVICED_VIRTUAL_ADDRESS_SUBNET` variable.

- 7 Start the Zenoss Control Center service.

```
systemctl start serviced
```

To monitor progress, enter the following command.

```
journalctl -u serviced -f
```

The `serviced` process invokes `docker` to download its internal services image from Docker Hub. The Zenoss Control Center web interface is unavailable until the internal services image is installed and its services are started. The process takes approximately 5-10 minutes.

- 8 Configure resource pool hosts or start Zenoss Core.
 - To configure resource pool hosts, proceed to [Preparing a resource pool host](#) on page 16.
 - To start Zenoss Core, proceed to [Starting Zenoss Core](#) on page 27.

Preparing a resource pool host

Perform this procedure to prepare a Red Hat Enterprise Linux (RHEL) 7.0 or CentOS 7.0 host as a Zenoss Control Center resource pool host.

- 1 Log in to the target host as `root`.
- 2 Verify that the host implements the 64-bit version of the x86 instruction set.

```
uname -m
```

- If the output is `x86_64`, the architecture is 64-bit. Proceed to the next step
 - If the output is `i386/i486/i586/i686`, the architecture is 32-bit. Stop this procedure and select a different host.
- 3 Disable the firewall.

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

- 4 Enable persistent storage for log files, if desired.

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

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

- 5 Create one Btrfs file system for Docker.

Note The following example procedure creates a file system on the `/dev/sdb1` partition. Modify the steps or values in this procedure as required for your environment.

- a Create a mount point for the Btrfs file system.

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

- b Create the file system.

```
DOCKER_PART=/dev/sdb1
mkfs -t btrfs $DOCKER_PART
```

- c Add the new file system to the `/etc/fstab` file.

```
echo "$DOCKER_PART /var/lib/docker btrfs rw,noatime,nodatacow 0 0" \
>> /etc/fstab
```


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

```
mount -a
mount | grep docker
```

The output of the preceding command should be similar to the following example.

```
/dev/sdb1 on /var/lib/docker type btrfs
(rw, noatime, seclabel, nodatasum, noatacow, ssd, space_cache)
```

- 6 Disable Security-Enhanced Linux (SELinux), if installed.

- a Determine whether SELinux is installed.

```
test -f /etc/selinux/config && grep '^SELINUX=' /etc/selinux/config
```

If the preceding commands return a result, SELinux is installed.

- b Set the operating mode to disabled, and confirm the setting.

```
EXT=$(date +"%j-%H%M%S")
sudo sed -i.${EXT} -e 's/^SELINUX=.*SELINUX=disabled/g' \
/etc/selinux/config && \
grep '^SELINUX=' /etc/selinux/config
```

- 7 Download and install the Zenoss repository package.

```
rpm -ivh http://get.zenoss.io/yum/zenoss-repo-1-1.x86_64.rpm
yum clean all
```

- 8 Install and start the Dnsmasq package.

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

- 9 Install the NTP package and start ntpd.

```
yum install -y ntp && systemctl enable ntpd && systemctl start ntpd
```

Note In testing, an unresolved issue prevents ntpd from restarting after a reboot. If this occurs in your environment, you must restart it manually.

- 10 Reboot the host.

```
sudo reboot
```

Proceed to the next procedure, [Installing Zenoss Control Center on a resource pool host](#) on page 17.

Installing Zenoss Control Center on a resource pool host

To perform this procedure, you need the Zenoss Control Center and Zenoss Core package files provided by your Zenoss representative.

- 1 Log in to the resource pool host as root.
- 2 Install Zenoss Control Center, Zenoss Core, and Docker, and then start Docker.

```
yum --enablerepo=zenoss-testing install -y zenoss-core-service
```

```
systemctl start docker
```

3 Configure the resource pool host for a multi-host deployment.

a Create a variable for the IP address or hostname of the master host.

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

```
MHOST=XX.XX.XX.XX
```

b Change variables in the Zenoss Control Center defaults file.

The default values in `/etc/default/serviced` configure Zenoss Control Center for a single-host deployment. To enable a multi-host deployment, uncomment and change the following variables.

HOME

The path `docker` uses to locate the `.dockercfg` authentication file. Docker Hub credentials are stored in the file.

SERVICED_REGISTRY

Determines whether `serviced` uses a local registry to store Docker images. Set the value to 1, true.

SERVICED_AGENT

Determines whether a `serviced` instance acts as a resource pool host. Set the value to 1, true.

SERVICED_MASTER

Determines whether a `serviced` instance acts as the Zenoss Control Center master host. Set the value to 0, false.

SERVICED_MASTER_IP

The IP address of the `serviced` instance configured as the master. Set the `SERVICED_MASTER_IP` variable to the IP address of the master host, then uncomment `SERVICED_MASTER_IP` and all other variables that reference it (`SERVICED_ZK`, `SERVICED_DOCKER_REGISTRY`, `SERVICED_ENDPOINT`, `SERVICED_LOG_ADDRESS`, `SERVICED_LOGSTASH_ES`, and `SERVICED_STATS_PORT`). Finally, replace the variable that reference `SERVICED_MASTER_IP` with the IP address of the master host.

The following commands make the required edits to `/etc/default/serviced`.

```
EXT=$(date +"%j-%H%M%S")
test ! -z "${MHOST}" && \
sed -i.${EXT} -e 's|^#|^#(HOME=/root\)|\1|' \
-e 's|^#|^#(SERVICED_REGISTRY=)\.|\1|' \
-e 's|^#|^#(SERVICED_AGENT=)\.|\1|' \
-e 's|^#|^#(SERVICED_MASTER=)\.|\10|' \
-e 's|^#|^#(SERVICED_MASTER_IP=)\.|\1|' \
-e '/=$SERVICED_MASTER_IP/ s|^#|^#|' \
-e 's|(\$SERVICED_MASTER_IP)|'${MHOST}'|' \
/etc/default/serviced
```

4 Optional: Specify an alternate private subnet for Zenoss Control Center, if necessary.

By default, Zenoss Control Center uses the 10.3/16 private subnet for virtual IP addresses. If your environment already uses 10.3/16 for other purposes, select an unused subnet for Zenoss Control Center, and add it to the `/etc/default/serviced` file.

Note [RFC 1918](#) restricts private networks to the 10.0/24, 172.16/20, and 192.168/16 address spaces. However, Zenoss Control Center accepts any valid, 16-bit, IPv4 address space for its private network.

For example, to set the private subnet for Zenoss Control Center to 10.20/16, uncomment the `SERVICED_VIRTUAL_ADDRESS_SUBNET` variable, and then set its value to 10.20.

The following commands make the required edits to `/etc/default/serviced`.

```
SUBNET=10.20
EXT=$(date +"%j-%H%M%S")
test ! -z "${SUBNET}" && \
sudo sed -i.${EXT} -e \
's|^#[^S]*\ (SERVICED_VIRTUAL_ADDRESS_SUBNET=) .*|\1'${SUBNET}'|' \
/etc/default/serviced
```

Note All hosts in a Zenoss Control Center deployment must have the same value for the `SERVICED_VIRTUAL_ADDRESS_SUBNET` variable.

- 5 Start the Zenoss Control Center service.

```
systemctl start serviced
```

To monitor progress, enter the following command.

```
journalctl -u serviced -f
```

- 6 Configure additional hosts or start Zenoss Core.

- To configure additional hosts, return to [Preparing a resource pool host](#) on page 16.
- To start Zenoss Core, proceed to [Starting Zenoss Core](#) on page 27.

3

Installing on Ubuntu hosts

The procedures in this chapter install Zenoss Control Center and Zenoss Core on one or more Ubuntu 14.04 LTS hosts. Please review the information in [Planning your deployment](#) on page 5 before performing these procedures.

In addition, the procedures in this chapter require access to public Zenoss repositories or local mirrors of Zenoss repositories. For information about creating local mirrors, see [Mirroring repositories](#) on page 35.

Finally, these procedures include example steps to create a [Btrfs](#) partition on a separate disk, which may not be appropriate for your environment. To create a Btrfs partition in a different way, please refer to your operating system documentation.

Preparing the master host

Follow this procedure to prepare an Ubuntu host for Zenoss Control Center.

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

```
uname -m
```

- If the output is `x86_64`, the architecture is 64-bit. Proceed to the next step
- If the output is `i386/i486/i586/i686`, the architecture is 32-bit. Stop this procedure and select a different host.

- 3 Disable the firewall.

```
sudo ufw disable
```

- 4 Install the `btrfs-tools` package, if necessary.

```
dpkg -s btrfs-tools >/dev/null 2>&1 \  
|| sudo apt-get install -y btrfs-tools
```

- 5 Create a Btrfs file system for application data.

Note The following example procedure creates a file system on the `/dev/sdb1` partition. Modify the steps or values in this procedure as required for your environment.

- a** Create a mount point for the Btrfs file system.

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

- b** Format the partition with Btrfs.

```
sudo mkfs.btrfs /dev/sdb1
```

- c** Add the new file system to the `/etc/fstab` file.

```
sudo sh -c 'echo \
"/dev/sdb1 /opt/serviced/var btrfs rw,noatime,nodatacow 0 0" \
>> /etc/fstab'
```

- d** Mount the filesystem, and then verify it mounted correctly.

```
sudo mount -a
mount | grep serviced
```

The output of the preceding command should be similar to the following example.

```
/dev/sdb1 on /opt/serviced/var type btrfs
(rw,noatime,seclabel,nodatasum,nodatacow,ssd,space_cache)
```

- 6** Disable Security-Enhanced Linux (SELinux), if installed.

- a** Determine whether SELinux is installed.

```
test -f /etc/selinux/config && grep '^SELINUX=' /etc/selinux/config
```

If the preceding commands return a result, SELinux is installed.

- b** Set the operating mode to disabled, and confirm the setting.

```
EXT=$(date +"%j-%H%M%S")
sudo sed -i.${EXT} -e 's/^SELINUX=.*SELINUX=disabled/g' \
/etc/selinux/config && \
grep '^SELINUX=' /etc/selinux/config
```

- 7** Install *Docker*.

```
curl -sSL https://get.docker.io/ubuntu/ | sudo sh
```

- 8** Add the current user account to the `docker` group.

```
sudo usermod -aG docker $USER
```

- 9** Install the Zenoss OpenPGP public key.

```
sudo apt-key adv --keyserver keys.gnupg.net --recv-keys AA5A1AD7
```

- 10** Add the Zenoss repository to the list of repositories.

If you are using a local mirror of the public Zenoss repository, replace the value of the `REPO` variable with your mirror's address in the following commands.

```
REPO=http://get.zenoss.io/apt/ubuntu
sudo sh -c 'echo "deb [ arch=amd64 ] '${REPO}' trusty universe" \
> /etc/apt/sources.list.d/zenoss.list'
```

- 11 Update the Ubuntu repository database.

```
sudo apt-get update
```

- 12 If you are setting up a multi-host deployment, install the NTP package and start ntpd.

```
sudo apt-get install -y ntp
```

- 13 Reboot the host.

```
sudo reboot
```

Installing Zenoss Control Center on the master host

Perform this procedure to install Zenoss Control Center and Zenoss Core on the master host.

- 1 Log in to the target host as a user with `sudo` and `docker` privileges.
- 2 Install the Zenoss Core service template.

```
sudo apt-get -t zenoss-testing install -y zenoss-core-service
```

- 3 Change the volume type for application data.

The `/etc/default/serviced` file contains variables that define the character of an instance of Zenoss Control Center. For more information about `serviced` configuration options, refer to *Zenoss Control Center Guide*.

The following `sed` command changes the value of the `SERVICED_FS_TYPE` variable from the default, `rsync`, to `btrfs`.

```
EXT=$(date +"%j-%H%M%S")
sudo sed -i.${EXT} \
-e 's|^#[^S]*\ (SERVICED_FS_TYPE=) .*$|\1btrfs|' \
/etc/default/serviced
```

- 4 Optional: Configure the master host for multi-host deployment, if desired.

The default values in the Zenoss Control Center defaults file, `/etc/default/serviced`, configure Zenoss Control Center for a single-host deployment. To enable a multi-host deployment, uncomment and change the following variables.

HOME

The path `docker` uses to locate the `.dockercfg` authentication file. Docker Hub credentials are stored in the file.

SERVICED_REGISTRY

Determines whether `serviced` uses a local registry to store Docker images. Set the value to 1, true.

SERVICED_AGENT

Determines whether a `serviced` instance acts as a resource pool host. A resource pool host runs application services scheduled for the resource pool to which it belongs. Set the value to 1, true.

SERVICED_MASTER

Determines whether a `serviced` instance acts as the Zenoss Control Center master host. The master host runs the application services scheduler and other internal services, including the web server for the Zenoss Control Center web interface. A `serviced` instance may be configured as both an agent and a master. Set the value to 1, true.

The following commands make the required edits to `/etc/default/serviced`.

```
EXT=$(date +%j-%H%M%S")
sudo sed -i.${EXT} -e 's|^#\[H\]*\ (HOME=/root\)|\1|' \
-e 's|^#\[S\]*\ (SERVICED_REGISTRY=)\ .|\11|' \
-e 's|^#\[S\]*\ (SERVICED_AGENT=)\ .|\11|' \
-e 's|^#\[S\]*\ (SERVICED_MASTER=)\ .|\11|' \
/etc/default/serviced
```

- 5 Optional: Specify an alternate private subnet for Zenoss Control Center, if necessary.

By default, Zenoss Control Center uses the 10.3/16 private subnet for virtual IP addresses. If your environment already uses 10.3/16 for other purposes, select an unused subnet for Zenoss Control Center, and add it to the `/etc/default/serviced` file.

Note [RFC 1918](#) restricts private networks to the 10.0/24, 172.16/20, and 192.168/16 address spaces. However, Zenoss Control Center accepts any valid, 16-bit, IPv4 address space for its private network.

For example, to set the private subnet for Zenoss Control Center to 10.20/16, uncomment the `SERVICED_VIRTUAL_ADDRESS_SUBNET` variable, and then set its value to 10.20.

The following commands make the required edits to `/etc/default/serviced`.

```
SUBNET=10.20
EXT=$(date +%j-%H%M%S")
test ! -z "${SUBNET}" && \
sudo sed -i.${EXT} -e \
's|^#\[S\]*\ (SERVICED_VIRTUAL_ADDRESS_SUBNET=)\ .*|\1'${SUBNET}'|' \
/etc/default/serviced
```

Note All hosts in a Zenoss Control Center deployment must have the same value for the `SERVICED_VIRTUAL_ADDRESS_SUBNET` variable.

- 6 Start the Zenoss Control Center service.

```
sudo start serviced
```

The `serviced` process invokes `docker` to download its internal services image from Docker Hub. The Zenoss Control Center web interface is unavailable until the internal services image is installed and its services are started. The process takes approximately 5-10 minutes.

- 7 Configure resource pool hosts or start Zenoss Core.
 - To configure resource pool hosts, proceed to [Preparing a resource pool host](#) on page 23.
 - To start Zenoss Core, proceed to [Starting Zenoss Core](#) on page 27.

Preparing a resource pool host

Follow this procedure to prepare an Ubuntu host for Zenoss Control Center.

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

```
uname -m
```

- If the output is `x86_64`, the architecture is 64-bit. Proceed to the next step

- If the output is `i386/i486/i586/i686`, the architecture is 32-bit. Stop this procedure and select a different host.

3 Disable the firewall.

```
sudo ufw disable
```

4 Install the `btrfs-tools` package, if necessary.

```
dpkg -s btrfs-tools >/dev/null 2>&1 \
|| sudo apt-get install -y btrfs-tools
```

5 Disable Security-Enhanced Linux (SELinux), if installed.

- a** Determine whether SELinux is installed.

```
test -f /etc/selinux/config && grep '^SELINUX=' /etc/selinux/config
```

If the preceding commands return a result, SELinux is installed.

- b** Set the operating mode to disabled, and confirm the setting.

```
EXT=$(date +"%j-%H%M%S")
sudo sed -i.${EXT} -e 's/^SELINUX=.*SELINUX=disabled/g' \
/etc/selinux/config && \
grep '^SELINUX=' /etc/selinux/config
```

6 Install *Docker*.

```
curl -sSL https://get.docker.io/ubuntu/ | sudo sh
```

7 Add the current user account to the `docker` group.

```
sudo usermod -aG docker $USER
```

8 Install the Zenoss OpenPGP public key.

```
sudo apt-key adv --keyserver keys.gnupg.net --recv-keys AA5A1AD7
```

9 Add the Zenoss repository to the list of repositories.

If you are using a local mirror of the public Zenoss repository, replace the value of the `REPO` variable with your mirror's address in the following commands.

```
REPO=http://get.zenoss.io/apt/ubuntu
sudo sh -c 'echo "deb [ arch=amd64 ] '${REPO}' trusty universe" \
> /etc/apt/sources.list.d/zenoss.list'
```

10 Update the Ubuntu repository database.

```
sudo apt-get update
```

11 Install the NTP package and start `ntpd`.

```
sudo apt-get install -y ntp
```

12 Reboot the host.

```
sudo reboot
```


Proceed to the next procedure, [Installing Zenoss Control Center on a resource pool host](#) on page 25.

Installing Zenoss Control Center on a resource pool host

- 1 Log in to the target host as a user with `sudo` and `docker` privileges.
- 2 Install the Zenoss Core service template.

```
sudo apt-get -t zenoss-testing install -y zenoss-core-service
```

- 3 Configure the resource pool host for a multi-host deployment.

- a Create a variable for the IP address or hostname of the master host.

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

```
MHOST=XX.XX.XX.XX
```

- b Change variables in the Zenoss Control Center defaults file.

The default values in `/etc/default/serviced` configure Zenoss Control Center for a single-host deployment. To enable a multi-host deployment, uncomment and change the following variables.

HOME

The path `docker` uses to locate the `.dockercfg` authentication file. Docker Hub credentials are stored in the file.

SERVICED_REGISTRY

Determines whether `serviced` uses a local registry to store Docker images. Set the value to 1, true.

SERVICED_AGENT

Determines whether a `serviced` instance acts as a resource pool host. Set the value to 1, true.

SERVICED_MASTER

Determines whether a `serviced` instance acts as the Zenoss Control Center master host. Set the value to 0, false.

SERVICED_MASTER_IP

The IP address of the `serviced` instance configured as the master. Set the `SERVICED_MASTER_IP` variable to the IP address of the master host, then uncomment `SERVICED_MASTER_IP` and all other variables that reference it (`SERVICED_ZK`, `SERVICED_DOCKER_REGISTRY`, `SERVICED_ENDPOINT`, `SERVICED_LOG_ADDRESS`, `SERVICED_LOGSTASH_ES`, and `SERVICED_STATS_PORT`).

The following commands make the required edits to `/etc/default/serviced`.

```
EXT=$(date +"%j-%H%M%S")
test ! -z "${MHOST}" && \
sed -i.${EXT} -e 's|^#[^H]*\ (HOME=/root\)|\1|' \
-e 's|^#[^S]*\ (SERVICED_REGISTRY=)\.|\1|' \
-e 's|^#[^S]*\ (SERVICED_AGENT=)\.|\1|' \
-e 's|^#[^S]*\ (SERVICED_MASTER=)\.|\10|' \
-e 's|^#[^S]*\ (SERVICED_MASTER_IP=)\.|\1|' "${MHOST}" \
-e '/=$SERVICED_MASTER_IP/ s|^#[^S]*|' \
/etc/default/serviced
```

- 4 Optional: Specify an alternate private subnet for Zenoss Control Center, if necessary.

By default, Zenoss Control Center uses the 10.3/16 private subnet for virtual IP addresses. If your environment already uses 10.3/16 for other purposes, select an unused subnet for Zenoss Control Center, and add it to the `/etc/default/serviced` file.

Note [RFC 1918](#) restricts private networks to the 10.0/24, 172.16/20, and 192.168/16 address spaces. However, Zenoss Control Center accepts any valid, 16-bit, IPv4 address space for its private network.

For example, to set the private subnet for Zenoss Control Center to 10.20/16, uncomment the `SERVICED_VIRTUAL_ADDRESS_SUBNET` variable, and then set its value to 10.20.

The following commands make the required edits to `/etc/default/serviced`.

```
SUBNET=10.20
EXT=$(date +%j-%H%M%S")
test ! -z "${SUBNET}" && \
sudo sed -i.${EXT} -e \
's|^#[^S]*\ (SERVICED_VIRTUAL_ADDRESS_SUBNET=) .*|\1'${SUBNET}'|' \
/etc/default/serviced
```

Note All hosts in a Zenoss Control Center deployment must have the same value for the `SERVICED_VIRTUAL_ADDRESS_SUBNET` variable.

- 5 Start the Zenoss Control Center service.

```
sudo start serviced
```

- 6 Configure additional hosts or start Zenoss Core.

- To configure additional hosts, return to [Preparing a resource pool host](#) on page 23.
- To start Zenoss Core, proceed to [Starting Zenoss Core](#) on page 27.

4

Starting Zenoss Core

The procedures in this chapter demonstrate how to start Zenoss Core for the first time. Some of the procedures are optional, depending on whether your deployment is single-host or multi-host. For best results, perform the procedures in the order in which they appear.

Enabling access to the Zenoss Control Center web interface

- 1 Log in to the master host as `root`.
- 2 Add users to the administrative group.

To gain access to the Zenoss Control Center web interface, users must be members of the Zenoss Control Center administrative group on the master host. The default administrative group is the system group. (On Ubuntu hosts, the system group is `sudo`; on RHEL and CentOS hosts, it is `wheel`.) You may add users to the system group, or designate a regular group as the administrative group.

- To add users to the system group, enter one of the following commands for each user to add.
 - Ubuntu hosts: `sudo usermod -aG sudo User`
 - RHEL/CentOS hosts: `sudo usermod -aG wheel User`
- To designate a regular group as the administrative group, see [Designating a regular group as the administrative group](#) on page 28.

- 3 Enable access to services.

Zenoss Control Center proxies the IP addresses of the services it manages, because the addresses change during normal operations. To simplify access, Zenoss Control Center provides virtual host aliases for the Zenoss Core services it manages.

To enable access to Zenoss Core services, add the virtual host aliases to a DNS server, or to the `/etc/hosts` file (or its equivalent) of client systems from which you wish to gain access to services.

The following line shows the syntax of the entry to add to a DNS server or `/etc/hosts` file.

```
IP-Address Host.Domain Host zenoss5x.Host hbase.Host opentsdb.Host
rabbitmq.Host
```

The following example shows an entry for a master host at IP address 10.10.1.2, named mypc, in the big.io domain.

```
10.1.1.2 mypc.big.io mypc zenoss5x.mypc hbase.mypc opentsdb.mypc
rabbitmq.mypc
```

Designating a regular group as the administrative group

Perform this procedure to designate a non-system group on the Zenoss Control Center master host as the administrative group. Membership in the administrative group is required to gain access to the Zenoss Control Center web interface.

- 1 Log in to the master host as root.
- 2 Create a variable for the group to designate as the administrative group.
In this example, the name of group to create is serviced. You may choose any name, or use an existing group.

```
GROUP=serviced
```

- 3 Create a new group, if necessary.
 - Ubuntu hosts: `addgroup ${GROUP}`
 - RHEL/CentOS hosts: `groupadd ${GROUP}`
- 4 Add one or more existing users to the group to designate as the administrative group.
Repeat the following command for each user to add.

```
sudo usermod -aG ${GROUP} User
```

- 5 Change the value of the `SERVICED_ADMIN_GROUP` variable in `/etc/default/serviced`.

```
EXT=$(date +"%j-%H%M%S")
test ! -z "${GROUP}" && sudo sed -i.${EXT} -e \
's|^#[^S]*\ (SERVICED_ADMIN_GROUP=) .*$|\1'${GROUP}'|' \
/etc/default/serviced || \
echo "*** GROUP undefined; no edit performed"
```

- 6 Prevent root users and members of the sudo or wheel groups from gaining access to the Zenoss Control Center web interface.

```
EXT=$(date +"%j-%H%M%S")
sudo sed -i.${EXT} \
-e 's|^#[^S]*\ (SERVICED_ALLOW_ROOT_LOGIN=) .*$|\10|' \
/etc/default/serviced
```

- 7 Restart Zenoss Control Center.
 - Ubuntu hosts: `sudo stop serviced && sudo start serviced`
 - RHEL/CentOS hosts: `systemctl stop serviced && systemctl start serviced`

Zenoss Core browser interfaces

The following table describes the Zenoss Core browser interfaces available through the Zenoss Control Center master host.

Note For each interface address, substitute the hostname of your master host for *Master*.

Address	Interface Name	Description
<code>https://Master</code>	Zenoss Control Center	The browser interface of the Zenoss Docker-based management system. Access to this interface is restricted to users with accounts on the master host. The users must be members of the <code>docker</code> and <code>sudo</code> groups.
<code>https://zenoss5.Master</code>	Zenoss Core	The browser interface of Zenoss Core.
<code>https://hbase.Master</code>	Apache HBase	The browser interface of the Apache HBase instance in which OpenTSDB stores device monitoring data.
<code>https://opentsdb.Master</code>	OpenTSDB	The browser interface of the OpenTSDB database which Zenoss Core uses to manage and manipulate device monitoring data.
<code>https://rabbitmq.Master</code>	RabbitMQ	The administrative interface of the <i>RabbitMQ</i> service.

Editing the hosts file of a Windows 7 system

- 1 Log in to the Windows 7 system as a user with administrative privileges.
- 2 From the **Start** menu, highlight **All Programs > Accessories > Notepad**.
- 3 Right click, and then select **Run as administrator**.
- 4 From the Notepad **File** menu, select **Open**.
- 5 In the **File name** field of the **Open** window, enter `C:\Windows\System32\drivers\etc\hosts`.
- 6 Insert the entry for your Zenoss Control Center master host at the end of the file.
- 7 Save the file and exit Notepad.

Deploying the Zenoss Core application

Perform this procedure after enabling client access.

- 1 Log in to the Zenoss Control Center web interface.

For example, if the master host is `mypc`, then the address of the Zenoss Control Center web interface is `https://mypc`.

Note Access to this interface is restricted to users with accounts on the Zenoss Control Center master host.

The **Deployment Wizard** displays upon initial login.

Note If you are using an appliance virtual machine, the **Deployment Wizard** might not display. If so, simply add a host to the default pool, and then deploy the application.

2 Add a host to the default resource pool.

For this step, the host to add is the Zenoss Control Center master host. Add resource pool hosts after completing the Deployment Wizard.

- a In the **Host and Port** field, enter the hostname or IP address of the Zenoss Control Center master host, followed by a colon character (:), and then 4979.
 - If you enter a hostname, all hosts in your Zenoss Control Center deployment must be able to resolve the name, either through an entry in `/etc/hosts` or through a nameserver on your network.
 - Port 4979 is the default port Zenoss Control Center uses to communicate among hosts in resource pools.
- b In the **Resource Pool ID** field, select default from the list, and then click **Next**.

3 Select an application to deploy.

- a Mark the `Zenoss.core (v5.0)` checkbox.

Application	Memory Required
<input type="checkbox"/> Zenoss.core (v5.0)	9.00 GB

- b Click **Next**.

4 Select a resource pool for the application.

- a Select the radio button of the default pool.

Deployment Wizard

✓ Step 1
Add Host

✓ Step 2
Select Applications

Step 3
Select Resource Pool

Step 4
Deploy Applications

You're not logged in to Docker Hub. Any application dependencies on private repositories will cause deployment to fail.

Select the resource pool to install to:

Resource Pool	Description	Memory	CPU Cores
<input checked="" type="radio"/> default		0.00 GB	0

Back Next

- b Click **Next**.
- 5 Deploy the application to the resource pool.
 - a In the **Deployment ID** field, enter an identifier for this deployment of the application.

Deployment Wizard

✓ Step 1
Add Host

✓ Step 2
Select Applications

✓ Step 3
Select Resource Pool

Step 4
Deploy Applications

You're not logged in to Docker Hub. Any application dependencies on private repositories will cause deployment to fail.

Zenoss.core has been configured for resource pool 'default'.

Deployment ID

Dev, Test, Production, etc.

Back Deploy Deploy and Start

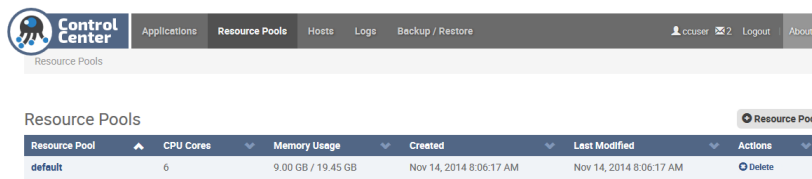
- b Click **Deploy**.
- Zenoss Control Center pulls the application's images from the Zenoss repository at Docker Hub, which takes several minutes.

Note If you are using an appliance virtual machine, the images are already in the local registry, and this step takes less than a minute.

- c When the **Deployment Wizard** completes, refresh the browser window to update the **Applications** table.
- 6 Optional: Add hosts to the default resource pool or other resource pools, if desired.
 - To add hosts to the default resource pool, see [Adding a host to a resource pool](#) on page 32.
 - To add hosts to other resource pools, see [Creating a resource pool](#) on page 31.

Creating a resource pool

- 1 Log in to the Zenoss Control Center web interface.
- 2 In the Zenoss Control Center menu bar, click **Resource Pools**.



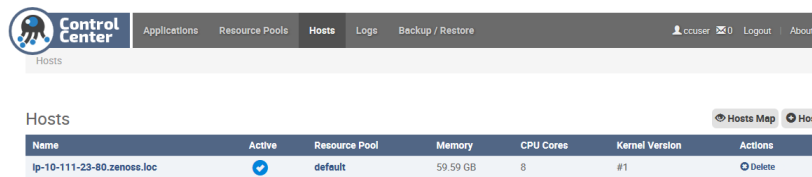
- 3 On the **Resource Pools** page, click the **+Resource Pool** button, located at the right side of the page.

- 4 In the **Resource Pool** field of the **Add Resource Pool** dialog, enter a name for the new resource pool.
- 5 Optional: In the **Description** field, enter a brief description of the purpose of the new resource pool, and then click **Add Resource Pool**.

To add hosts to the new resource pool, see [Adding a host to a resource pool](#) on page 32.

Adding a host to a resource pool

- 1 Log in to the Zenoss Control Center web interface.
- 2 In the Zenoss Control Center menu bar, click **Hosts**.



- 3 On the **Hosts** page, click the **+Host** button, located at the right side of the page.

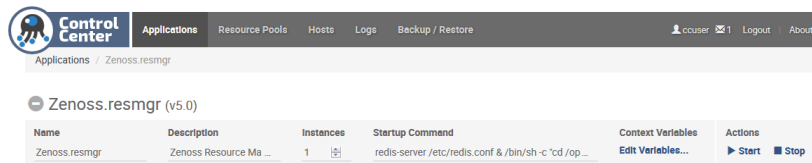
- 4 In the **Host and Port** field of the **Add Host** dialog, enter the hostname or IP address of a Zenoss Control Center resource pool host, followed by a colon character (:), and then 4979.
- 5 In the **Resource Pool ID** field, select a resource pool from the list, and then click **Add Host**.

Changing the default passwords of application services

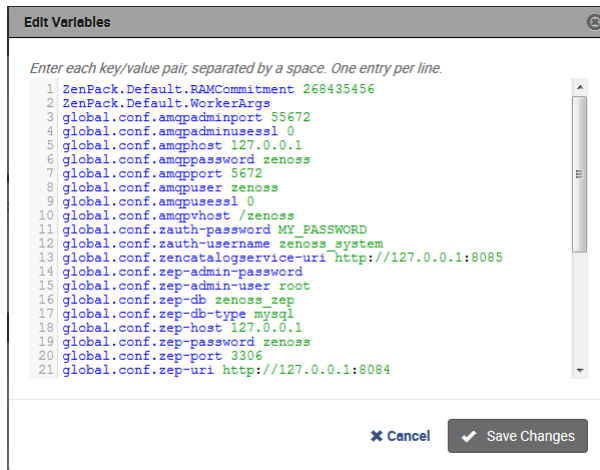
Zenoss Core includes several services with independent authentication systems, each of which has a default password defined by Zenoss. This procedure identifies the default passwords to change, and demonstrates how to gain access to and edit global application variables.

Note You may change the default passwords of services other than the ones identified in this procedure. However, Zenoss recommends not changing user account names.

- 1 Log in to the Zenoss Control Center web interface.
- 2 In the **Application** column of the **Applications** table, click the name of the application to secure.



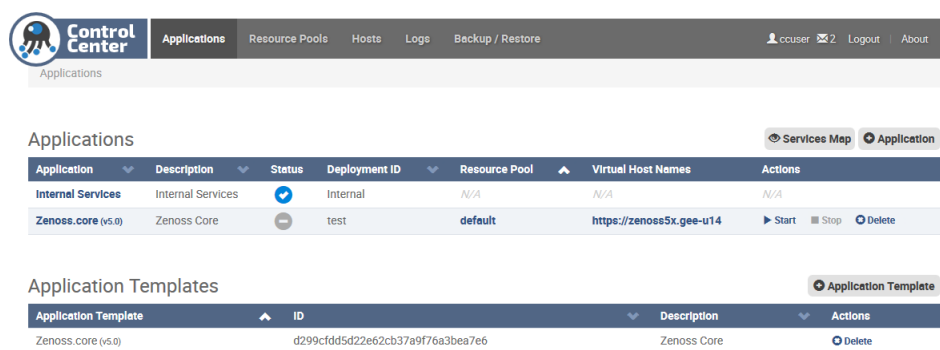
- 3 In the **Context Variables** column of the application instance table, click **Edit Variables...**



- 4 Change the default password of the RabbitMQ service.
 - a In the **Edit Variables** dialog, locate the `global.conf.amqppassword` variable.
 - b Replace the default value, `zenoss`, with a new password.
- 5 Change the default password of the Zenoss authentication proxy.
 - a In the **Edit Variables** dialog, locate the `global.conf.zauth-password` variable.
 - b Replace the default value, `MY_PASSWORD`, with a new password.
- 6 Click **Save Changes**.

Starting the Zenoss Core application

- 1 Log in to the Zenoss Control Center web interface.
- 2 At the top of the page, click **Applications**.



Control Center

Applications | Resource Pools | Hosts | Logs | Backup / Restore

ccuser 2 Logout About

Applications

Applications Services Map Application

Application	Description	Status	Deployment ID	Resource Pool	Virtual Host Names	Actions
Internal Services	Internal Services	✓	Internal	N/A	N/A	N/A
Zenoss.core (v5.0)	Zenoss Core	—	test	default	https://zenoss5x.gee-u14	Start Stop Delete

Application Templates Application Template

Application Template	ID	Description	Actions
Zenoss.core (v5.0)	d299cfd5d5d22e62cb37a9f76a3bea7e6	Zenoss Core	Delete

- In the **Actions** column, click the **Start** control.

To monitor the startup process, click the application's name, in the **Application** column, and then scroll down the page.

A

Mirroring repositories

The procedure in this section enable installing Zenoss Control Center and Zenoss Core on hosts that do not have an internet connection.

Mirroring the Zenoss Docker Hub repository

You may use a transfer host with synchronous access to both the internet and the Zenoss Control Center master host to mirror the Zenoss Docker Hub repository.

To perform this procedure:

- A Zenoss Control Center master host must be installed and running in your environment.
- A separate, transfer host must have simultaneous network access to the internet and to the Zenoss Control Center master host.
- The transfer host must implement the 64-bit version of the x86 instruction set.
- The transfer host and the master host must be able to resolve each other's fully-qualified domain names (FQDN).

Preparing a transfer host

This procedure is written for the bash shell on Ubuntu 14.04 LTS Linux hosts.

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

```
uname -m
```

- If the output is `x86_64`, the architecture is 64-bit. Proceed to the next step
 - If the output is `i386/i486/i586/i686`, the architecture is 32-bit. Stop this procedure and select a different host.
- 3 Install *Docker*.

```
curl -sSL https://get.docker.io/ubuntu/ | sudo sh
```

- 4 Add the current user account to the `docker` group.

```
sudo usermod -aG docker $USER
```

- 5 Determine the installed version of Python.

```
python --version
```

The required minimum version for this procedure is 2.7, which is included in Ubuntu 14.04 by default.

- 6 Download the mirror script from the Zenoss Control Center master host.

```
curl -k -O https://Master-Host-FQDN/static/scripts/template-mirror.py
```

- 7 Add execute permission to the mirror script.

```
chmod +x template-mirror.py
```

- 8 Reboot the host.

```
sudo reboot
```

Mirroring with synchronous internet and master host access

- 1 Log in to the transfer host as a user with `sudo` and `docker` privileges.
- 2 Copy the service definition template file from the Zenoss Control Center master host to the transfer host.

```
scp Master-Host-FQDN:/opt/serviced/templates/Template.json .
```

- 3 Invoke the mirror script.

```
./template-mirror.py ./Template.json Master-Host-FQDN:5000 \  
./Template_Master-Host-FQDN.json
```

The mirror script

- 1 pulls the Docker images listed in the original template into a Docker index on the transfer host
- 2 pushes the images in the local index to the Docker registry on the Zenoss Control Center master host
- 3 tags the images on the master host with the FQDN of the master host
- 4 creates a new template, referencing the new image tags
- 4 Copy the new template to the master host.

```
scp ./Template_Master-Host-FQDN.json \  
Master-Host-FQDN:/opt/serviced/templates
```

- 5 Log in to the Zenoss Control Center master host as a user with `sudo` and `docker` privileges.
- 6 Remove the template added when serviced was installed.

```
ID=$(serviced template list | tail -1 | awk '{ print $1 }'); echo $ID  
serviced template remove $ID
```

- 7 Add the new template to Zenoss Control Center.

```
serviced template add \  
/opt/serviced/templates/Template_Master-Host-FQDN.json
```

This step makes the new template available for deployment in the Zenoss Control Center web interface.