Control Center Planning Guide

Release 1.5.0

Zenoss, Inc.

www.zenoss.com

Control Center Planning Guide

Part Number: 1930.18.039.23

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

Contents

About this guide

Control Center Planning Guide provides detailed information about preparing to deploy Control Center. For information about planning a high-availability deployment, refer to the *Control Center Installation Guide*.

Supported clients and browsers

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

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

Related publications

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

Documentation feedback

To provide feedback about this document, or to report an error or omission, please send an email to `docs@controlcenter.io`. In the email, please include the document title (*Control Center Planning*

* Enterprise mode only; compatibility mode is not tested.

Guide) and part number (1930.18.039.23) and as much information as possible about the context of your feedback.

Change history

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

1930.18.039.23 (1.5.0)

Add a description of ZooKeeper.

Add example `lsblk` output from installed master and delegate hosts.

1930.17.311 (1.5.0)

Remove RHEL/CentOS 7.1, add 7.4.

1330.17.278 (1.4.1)

Update the tested operating system and kernel descriptions.

1330.17.268 (1.4.1)

Add this document history section.

1 Introduction to Control Center

This chapter introduces Control Center, an open-source application service orchestrator based on *Docker Community Edition* (Docker CE, or just Docker).

Control Center is a platform-as-a-service framework that can manage any Docker application, from a simple web application to a multi-tiered stateful application stack. Control Center is based on a service-oriented architecture, which enables applications to run as a set of distributed services spanning hosts, datacenters, and geographic regions.

Control Center relies on declarations of application requirements to integrate 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.

Features

Control Center includes the following key features:

- Intuitive HTML5 interface for deploying and managing applications
- Integrated backup and restore, and incremental snapshots and rollbacks
- Centralized logging through Logstash and Elasticsearch
- Integration with database services and other persistent services
- Encrypted communications among all services and containers
- Delegate host authentication to prevent unauthorized system access
- Storage monitoring and emergency shutdown of services to minimize the risk of data corruption
- Rolling restart of services to reduce downtime of multi-instance services
- Audit logging, including application audit logging

Terminology, internal services, and concepts

This section defines Control Center terminology, internal services that enable Control Center to function, and concepts that are used in this guide and other documentation.

application

One or more software services packaged in Docker containers.

delegate host

A host that runs the application services scheduled for the resource pool to which it belongs. A system can be configured as delegate or master.

Docker Community Edition (Docker CE, or just Docker)

An open-source application for building, shipping, and running distributed applications.

Elasticsearch

(Control Center internal service) A distributed, real-time search and analytics engine. Control Center uses it to index log files and store service definitions.

Kibana

(Control Center internal service) A browser-based user interface that enables the display and search of Elasticsearch databases, including the log files that Control Center monitors.

Logstash

(Control Center internal service) A log file collector and aggregator that forwards parsed log file entries to Elasticsearch.

master host

The host that runs the application services scheduler, the Docker registry, the distributed file system, and other internal services, including the HTTP server for the Control Center browser interface and application browser interface. A system can be configured as delegate or master. Only one Control Center host can be the master.

OpenTSDB

(Control Center internal service) A time series database that Control Center uses to store its service performance metrics.

resource pool

A collection of one or more hosts, each with its own compute, network, and storage resources. All of the hosts in a resource pool must have identical hardware resources, and must be located in the same data center and on the same subnet. If a resource pool host is a hypervisor guest system, all of the hosts in the resource pool must be guests of the same hypervisor host system.

service

A process and its supporting files that Control Center runs in a single container to provide specific functionality as part of an application.

serviced

The name of the Control Center service and a command-line client for interacting with the service.

tenant

An application that Control Center manages.

ZooKeeper (*Apache ZooKeeper*)

(Control Center internal service) A centralized service that Control Center uses for configuration maintenance, naming, distributed synchronization, and providing group services.

Docker fundamentals

This section summarizes *the architecture description provided by Docker* as customized for Control Center. For additional information, refer to the Docker site.

Docker provides convenient tools that make use of the *control groups feature of the Linux kernel* to develop, distribute, and run applications. Docker internals include images, registries, and containers.

Docker images

Docker images are read-only templates that are used to create Docker containers. Images are easy to build, and image updates are change layers, not wholesale replacements.

Docker registries

Docker registries hold images. Control Center uses a private Docker registry for its own images and application images.

Docker containers

Docker containers have everything needed to run a service instance, and are created from images. Control Center launches each service instance in its own Docker container.

Docker storage

Docker and Control Center data are stored in customized LVM thin pools that are created from one or more block devices or partitions, or from one or more LVM volume groups.

ZooKeeper and Control Center

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

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

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

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

Control Center application data storage

Control Center uses a dedicated LVM thin pool on the master host to store application data and snapshots of application data.

- The distributed file system (DFS) of each tenant application that `serviced` manages is stored in separate virtual devices. The initial size of each tenant device is copied from the base device, which is created during the initial startup of `serviced`.
- Snapshots of tenant data, used as temporary restore points, are stored in separate virtual devices, outside of tenant virtual devices. The size of a snapshot depends on the size of the tenant device, and grows over time.

The Control Center master host requires high-performance, persistent storage. Storage can be local or remote.

- For local storage, solid-state disk (SSD) devices are recommended.
- For remote storage, storage-area network (SAN) systems have been tested. High-performance SAN systems are recommended, as is assigning separate logical unit numbers (LUNs) for each mounted path.

The overall response times of master host storage affect the performance and stability of Control Center internal services and the applications it manages. For example, ZooKeeper (a key internal service) is sensitive to storage latency greater than 1000 milliseconds.

Note The physical devices associated with the application data thin pool must be persistent. If removable or re-connectable storage such as a SAN based on iSCSI is used, then the Device-Mapper Multipath feature of RHEL/CentOS must be configured and enabled.

Control Center includes the `serviced-storage` utility for creating and managing its thin pool. The `serviced-storage` utility can:

- use physical devices or partitions, or LVM volume groups, to create a new LVM thin pool for application data
- add space to a tenant device at any time
- identify and clean up orphaned snapshots
- create an LVM thin pool for Docker data

2 Hardware requirements

Control Center requires real or virtual master and delegate hosts that

- implement the 64-bit version of the x86 instruction set
- support Red Hat Enterprise Linux (RHEL) 7.x or CentOS 7.x
- support Advanced Encryption Standard (AES)
- include a network interface controller that supports TCP/IP and IPv4

Hardware resource and storage requirements for Control Center vary by role (master or delegate host) and by the services assigned to the resource pool to which a host belongs. This chapter provides minimum requirements for master hosts and delegate hosts. For more information about the requirements for your deployment, refer to your application documentation.

Master host CPU and RAM resources

You can create a multi-host or single-host deployment of Control Center, on real or virtual hosts.

Zenoss recommends that all production deployments include one Control Center master host and two or more delegate hosts. In this configuration, the master host runs in its own, separate resource pool, and runs only Control Center internal services. Delegate hosts run application services in other resource pools.

The master host in a multi-host deployment typically requires 4 real or virtual CPU cores and 16GB RAM. Very large multi-host deployments may require 8 CPU cores, but additional RAM is not required.

In a single-host deployment, the Control Center master host runs all Control Center internal services and all application services. The following table lists minimum resource requirements for the master host. Actual CPU and RAM requirements depend on the application and its load.

Table 1: Minimum resource requirements for single-host deployments

Software	CPU cores	RAM
Control Center (required)	4	16GB
Zenoss Resource Manager (example)	add 4	add 48GB
Zenoss Core (example)	add 0	add 8GB

Note An under-resourced master host cannot function properly. Deploy Control Center and the application it manages on a master host that meets or exceeds the combined minimum requirements.

Master host storage requirements are detailed in the following section.

Master host storage areas

A Control Center master host requires the following storage areas:

Root

Size: Depends on configuration

Type: XFS file system (typically)

Mount point: `/`

Resource: One or more block devices or partitions, or one or more LVM physical volumes.

Preparation: None. The file system is created when the operating system is installed.

Docker temporary

Size: 10GB

Type: XFS file system (typically)

Mount point: `/tmp`

Resource: One or more block devices or partitions, or one or more LVM physical volumes.

Preparation: Depends on configuration. Typically, additional space is allocated for the root filesystem, but a separate real or logical partition may be used. The *Control Center Installation Guide* includes instructions to link the Docker temporary directory to `/tmp`.

Swap

Size: 12GB to 16GB

Type: swap

Mount point: None

Resource: One or more block devices or partitions, one or more LVM physical volumes, or a special file on the root filesystem.

Preparation: Depends on configuration.

Docker data

Size: 50GB

Type: LVM thin pool

Mount point: None

Resource: One or more block devices or partitions, or one or more LVM physical volumes.

Preparation: None. The installation procedures include steps for creating the thin pool.

Control Center internal services data

Size: 50GB

Type: XFS file system

Mount point: `/opt/serviced/var/isvcs`

Resource: One or more block devices or partitions, or one or more LVM physical volumes.

Preparation: None. The installation procedures include steps for formatting the resource.

Control Center audit logging

Size: 10GB (default)

Type: XFS file system

Mount point: `/var/log/serviced`

Resource: One or more block devices or partitions, one or more LVM physical volumes, or a remote file server that is compatible with Linux.

Preparation: Depends on configuration. Typically, additional space is allocated for the root filesystem. For more information about audit logging, refer to the *Control Center Reference Guide*.

Application data

Size: 200GB suggested. The size should be twice as large as the base device, which determines the size of tenant virtual devices.

Type: LVM thin pool

Mount point: None

Resource: One or more block devices or partitions, or one or more LVM physical volumes.

Preparation: None. The installation procedures include steps for creating the thin pool.

Application data backups

Size: 150GB suggested. The size should be at least 150% of the size of the base device.

Type: XFS file system, or a Linux-compatible file server

Mount point: `/opt/serviced/var/backups`

Resource: One or more block devices or partitions, one or more LVM physical volumes, or a remote file server that is compatible with Linux.

Note When the storage for backups and Control Center internal services data use the same physical device, resource contention can cause failures during backup and restore operations. For optimal results, use separate physical devices.

Preparation: None. The installation procedures include steps for formatting or mounting the resource.

The following example shows the disk configuration of a Control Center master host.

Figure 1: Example master host with 7 disks

```
# lsblk -ap --output=NAME,SIZE,TYPE,FSTYPE,MOUNTPOINT
NAME                                                                                                  SIZE TYPE FSTYPE      MOUNTPOINT
/dev/fd0                                                                                                4K disk
/dev/sda                                                                                             29.3G disk
##/dev/sda1                                                                                          29.3G part xfs         /
/dev/sdb                                                                                             48.8G disk
##/dev/sdb1                                                                                          48.8G part LVM2_member
  ##/dev/mapper/docker-docker--pool_tmeta                                                             900M lvm
  # ##/dev/mapper/docker-docker--pool                                                                42.2G lvm
  #   ##/dev/mapper/docker-8:1-42048-7a049d85cad6c528579e690364a55a5e45bd1d774d120913f63a7a66f781b8c5   45G dm   xfs         /var/lib/docker/devicemapper/mnt/7a049d85cad6c
  #   ##/dev/mapper/docker-8:1-42048-0f93eb8de36bba888d8618562f3c7196de1cf28702333d39c65cea9f7545f1c8   45G dm   xfs         /var/lib/docker/devicemapper/mnt/0f93eb8de36bb
  #   ##/dev/mapper/docker-8:1-42048-d84939118b6de8377a1ecdefd17104b099ef220436eaf283f4013e84faaa6493   45G dm   xfs         /var/lib/docker/devicemapper/mnt/d84939118b6de
  #   ##/dev/mapper/docker-8:1-42048-2daa75d92947e99b9cecbac1713bc26f230747af6954cab099d6510aa67004e0   45G dm   xfs         /var/lib/docker/devicemapper/mnt/2daa75d92947e
  #   ##/dev/mapper/docker-8:1-42048-e7277d3081955ed7e3a1e4b6ef004e9c71e4c1bd3895465ea4ac16dcba387ff5   45G dm   xfs         /var/lib/docker/devicemapper/mnt/e7277d3081955
  #   ##/dev/mapper/docker-8:1-42048-e91db5ccdcf21a43e319e70180e8d27e1ad449aaeb32a508558c3789fbc72a58   45G dm   xfs         /var/lib/docker/devicemapper/mnt/e91db5ccdcf21
  #   ##/dev/mapper/docker-8:1-42048-99104ce4a2d39c33a7f45567aa788e314a198367c52954ee8e0a18ae9a743ceb   45G dm   xfs         /var/lib/docker/devicemapper/mnt/99104ce4a2d39
  #   ##/dev/mapper/docker-8:1-42048-a29f148212846f3382fc8e45d2760c4f6ce9e979725a6debcedb3a98d42402a5   45G dm   xfs         /var/lib/docker/devicemapper/mnt/a29f148212846
  ##/dev/mapper/docker-docker--pool_tdata                                                            42.2G lvm
    ##/dev/mapper/docker-docker--pool                                                                42.2G lvm
      ##/dev/mapper/docker-8:1-42048-7a049d85cad6c528579e690364a55a5e45bd1d774d120913f63a7a66f781b8c5   45G dm   xfs         /var/lib/docker/devicemapper/mnt/7a049d85cad6c
      ##/dev/mapper/docker-8:1-42048-0f93eb8de36bba888d8618562f3c7196de1cf28702333d39c65cea9f7545f1c8   45G dm   xfs         /var/lib/docker/devicemapper/mnt/0f93eb8de36bb
      ##/dev/mapper/docker-8:1-42048-d84939118b6de8377a1ecdefd17104b099ef220436eaf283f4013e84faaa6493   45G dm   xfs         /var/lib/docker/devicemapper/mnt/d84939118b6de
      ##/dev/mapper/docker-8:1-42048-2daa75d92947e99b9cecbac1713bc26f230747af6954cab099d6510aa67004e0   45G dm   xfs         /var/lib/docker/devicemapper/mnt/2daa75d92947e
      ##/dev/mapper/docker-8:1-42048-e7277d3081955ed7e3a1e4b6ef004e9c71e4c1bd3895465ea4ac16dcba387ff5   45G dm   xfs         /var/lib/docker/devicemapper/mnt/e7277d3081955
      ##/dev/mapper/docker-8:1-42048-e91db5ccdcf21a43e319e70180e8d27e1ad449aaeb32a508558c3789fbc72a58   45G dm   xfs         /var/lib/docker/devicemapper/mnt/e91db5ccdcf21
      ##/dev/mapper/docker-8:1-42048-99104ce4a2d39c33a7f45567aa788e314a198367c52954ee8e0a18ae9a743ceb   45G dm   xfs         /var/lib/docker/devicemapper/mnt/99104ce4a2d39
      ##/dev/mapper/docker-8:1-42048-a29f148212846f3382fc8e45d2760c4f6ce9e979725a6debcedb3a98d42402a5   45G dm   xfs         /var/lib/docker/devicemapper/mnt/a29f148212846
/dev/sdc                                                                                             15.6G disk
##/dev/sdc1                                                                                          15.6G part swap        [SWAP]
/dev/sdd                                                                                             15.6G disk
##/dev/sdd1                                                                                          15.6G part xfs         /tmp
/dev/sde                                                                                             48.8G disk
##/dev/sde1                                                                                          48.8G part xfs         /opt/serviced/var/isvcs
/dev/sdf                                                                                            146.5G disk
##/dev/sdf1                                                                                          146.5G part xfs         /opt/serviced/var/backups
/dev/sdg                                                                                            195.3G disk
##/dev/sdg1                                                                                          195.3G part LVM2_member
  ##/dev/mapper/serviced-serviced--pool_tmeta                                                          1.8G lvm
  # ##/dev/mapper/serviced-serviced--pool                                                            172.3G lvm
  #   ##/dev/mapper/docker-8:1-33639769-33l4GPkupXkBInT0pbYYJO                                         90G dm   ext4        /exports/serviced_volumes_v2/a9hil5o55iy266s5l
  ##/dev/mapper/serviced-serviced--pool_tdata                                                       172.3G lvm
    ##/dev/mapper/serviced-serviced--pool                                                           172.3G lvm
      ##/dev/mapper/docker-8:1-33639769-33l4GPkupXkBInT0pbYYJO                                       90G dm   ext4        /exports/serviced_volumes_v2/a9hil5o55iy266s5l
/dev/sr0                                                                                           1024M rom
```

Delegate host CPU and RAM resources

Delegate hosts require adequate resources to support all application services assigned to the host's resource pool. All delegate hosts in a resource pool need identical resources. For more information about required CPU and RAM resources, refer to your application documentation.

Delegate host storage requirements

Like master hosts, Control Center delegate hosts require high-performance, persistent storage. Storage can be local or remote.

- For local storage, solid-state disk (SSD) devices are recommended.
- For remote storage, storage-area network (SAN) systems have been tested. High-performance SAN systems are recommended, as is assigning separate logical unit numbers (LUNs) for each mounted path.

The following storage configuration is recommended for delegate hosts:

- root filesystem with a minimum of 30GB of storage, formatted with XFS
- dedicated swap area
- one or more block devices or partitions, or one or more LVM physical volumes, with a total of 50GB of space. The installation procedures include steps for using `serviced-storage` to create an LVM thin pool for Docker data.

The following example shows the disk configuration of a Control Center delegate host.

Figure 2: Example delegate host with 4 disks

```
# lsblk -ap --output=NAME,SIZE,TYPE,FSTYPE,MOUNTPOINT
NAME                                                                                                  SIZE TYPE FSTYPE      MOUNTPOINT
/dev/fd0                                                                                               4K disk
/dev/sda                                                                                            29.3G disk
##/dev/sda1                                                                                          29.3G part xfs         /
/dev/sdb                                                                                            48.8G disk
##/dev/sdb1                                                                                          48.8G part LVM2_member
  ##/dev/mapper/docker-docker--pool_tmeta                                                            900M lvm
  # ##/dev/mapper/docker-docker--pool                                                                42.2G lvm
  #   ##/dev/mapper/docker-8:1-16786822-2a6c277141569d87eb95e34d1f8b856271c71a81cd4a07e9e09a552bdc6ded5a   45G dm   xfs         /var/lib/docker/devicemapper/mnt/2a6c2771415
  #   ##/dev/mapper/docker-8:1-16786822-579a2fb361130fb926f2045444c128efe8fa7cac2d85120e94b265196eee1b13   45G dm   xfs         /var/lib/docker/devicemapper/mnt/579a2fb3611
  #   ##/dev/mapper/docker-8:1-16786822-c95ca5f04a4e07d2084299fd544a524c4f4d6c0a8631717e4a4e898b382e2ef6   45G dm   xfs         /var/lib/docker/devicemapper/mnt/c95ca5f04a4
  #   ##/dev/mapper/docker-8:1-16786822-e9733ecdda1d3b3ea16a6e4b6385d2b00fc1cf332e0bca26efac2e884858cbd4   45G dm   xfs         /var/lib/docker/devicemapper/mnt/e9733ecdda1
  #   ##/dev/mapper/docker-8:1-16786822-947636812ed0592739d755b0591abb8f51cf6dc25982afe221021cf76d9cf142   45G dm   xfs         /var/lib/docker/devicemapper/mnt/947636812ed
  #   ##/dev/mapper/docker-8:1-16786822-d65a56e4300079a2d5bcf99d649b11c9cee8a23103775aec21ca593a65b2d130f   45G dm   xfs         /var/lib/docker/devicemapper/mnt/d65a56e4300
  #   ##/dev/mapper/docker-8:1-16786822-3d1fb13572cf0fb0c1736fcb84eed91b934263b76f317197dd9c6f6e74e68b9f   45G dm   xfs         /var/lib/docker/devicemapper/mnt/3d1fb13572c
  #   ##/dev/mapper/docker-8:1-16786822-542e3e08ccccc53ab0094bc87bee57405324bb37dd1205398f2978e9eb974cc6   45G dm   xfs         /var/lib/docker/devicemapper/mnt/542e3e08ccc
  #   ##/dev/mapper/docker-8:1-16786822-3beb71206399ef166df725a31ca770a85791a30f5174ed90414c22535ca8dc7f   45G dm   xfs         /var/lib/docker/devicemapper/mnt/3beb7120639
  #   ##/dev/mapper/docker-8:1-16786822-1c88db1b91c8164fcb62603a7be87583aae88449e466eb983db662d471a87fb8   45G dm   xfs         /var/lib/docker/devicemapper/mnt/1c88db1b91c
  #   ##/dev/mapper/docker-8:1-16786822-3a246ac134c631e4740fd9a4ecee63d3e9e2442c580ac779ddc8915c224ec0e9   45G dm   xfs         /var/lib/docker/devicemapper/mnt/3a246ac134c
  #   ##/dev/mapper/docker-8:1-16786822-045933b4418dc31765d8b9583def38d5887846423fc8a0fc8142c4dc03e8a558   45G dm   xfs         /var/lib/docker/devicemapper/mnt/045933b4418
  #   ##/dev/mapper/docker-8:1-16786822-e420dfc0229c744c0b03f186b5eec5e2f0f2dc002b52fde38b23d5cffafcd3b2   45G dm   xfs         /var/lib/docker/devicemapper/mnt/e420dfc0229
  #   ##/dev/mapper/docker-8:1-16786822-918a6f927345ac96069f811b7ce9206ac702997210640404458ea5a200e79dde6   45G dm   xfs         /var/lib/docker/devicemapper/mnt/918a6f92734
  #   ##/dev/mapper/docker-8:1-16786822-a7efd4f15c38e906b2d34ce97f95b7a41be3ea3c022f70a12eb530b6d9eea81a   45G dm   xfs         /var/lib/docker/devicemapper/mnt/a7efd4f15c3
  #   ##/dev/mapper/docker-8:1-16786822-17b847994e2209acbad7f19e00f1f3947dcae6489e6150a269ddbecd367ad68c   45G dm   xfs         /var/lib/docker/devicemapper/mnt/17b847994e2
  #   ##/dev/mapper/docker-8:1-16786822-3564e748e1476950c051afdbcf924d7a5e513cee79f69ee95649eea864733651   45G dm   xfs         /var/lib/docker/devicemapper/mnt/3564e748e14
  ##/dev/mapper/docker-docker--pool_tdata                                                            42.2G lvm
    ##/dev/mapper/docker-docker--pool                                                                42.2G lvm
      ##/dev/mapper/docker-8:1-16786822-2a6c277141569d87eb95e34d1f8b856271c71a81cd4a07e9e09a552bdc6ded5a   45G dm   xfs         /var/lib/docker/devicemapper/mnt/2a6c2771415
      ##/dev/mapper/docker-8:1-16786822-579a2fb361130fb926f2045444c128efe8fa7cac2d85120e94b265196eee1b13   45G dm   xfs         /var/lib/docker/devicemapper/mnt/579a2fb3611
      ##/dev/mapper/docker-8:1-16786822-c95ca5f04a4e07d2084299fd544a524c4f4d6c0a8631717e4a4e898b382e2ef6   45G dm   xfs         /var/lib/docker/devicemapper/mnt/c95ca5f04a4
      ##/dev/mapper/docker-8:1-16786822-e9733ecdda1d3b3ea16a6e4b6385d2b00fc1cf332e0bca26efac2e884858cbd4   45G dm   xfs         /var/lib/docker/devicemapper/mnt/e9733ecdda1
      ##/dev/mapper/docker-8:1-16786822-947636812ed0592739d755b0591abb8f51cf6dc25982afe221021cf76d9cf142   45G dm   xfs         /var/lib/docker/devicemapper/mnt/947636812ed
      ##/dev/mapper/docker-8:1-16786822-d65a56e4300079a2d5bcf99d649b11c9cee8a23103775aec21ca593a65b2d130f   45G dm   xfs         /var/lib/docker/devicemapper/mnt/d65a56e4300
      ##/dev/mapper/docker-8:1-16786822-3d1fb13572cf0fb0c1736fcb84eed91b934263b76f317197dd9c6f6e74e68b9f   45G dm   xfs         /var/lib/docker/devicemapper/mnt/3d1fb13572c
      ##/dev/mapper/docker-8:1-16786822-542e3e08ccccc53ab0094bc87bee57405324bb37dd1205398f2978e9eb974cc6   45G dm   xfs         /var/lib/docker/devicemapper/mnt/542e3e08ccc
      ##/dev/mapper/docker-8:1-16786822-3beb71206399ef166df725a31ca770a85791a30f5174ed90414c22535ca8dc7f   45G dm   xfs         /var/lib/docker/devicemapper/mnt/3beb7120639
      ##/dev/mapper/docker-8:1-16786822-1c88db1b91c8164fcb62603a7be87583aae88449e466eb983db662d471a87fb8   45G dm   xfs         /var/lib/docker/devicemapper/mnt/1c88db1b91c
      ##/dev/mapper/docker-8:1-16786822-3a246ac134c631e4740fd9a4ecee63d3e9e2442c580ac779ddc8915c224ec0e9   45G dm   xfs         /var/lib/docker/devicemapper/mnt/3a246ac134c
      ##/dev/mapper/docker-8:1-16786822-045933b4418dc31765d8b9583def38d5887846423fc8a0fc8142c4dc03e8a558   45G dm   xfs         /var/lib/docker/devicemapper/mnt/045933b4418
      ##/dev/mapper/docker-8:1-16786822-e420dfc0229c744c0b03f186b5eec5e2f0f2dc002b52fde38b23d5cffafcd3b2   45G dm   xfs         /var/lib/docker/devicemapper/mnt/e420dfc0229
      ##/dev/mapper/docker-8:1-16786822-918a6f927345ac96069f811b7ce9206ac702997210640404458ea5a200e79dde6   45G dm   xfs         /var/lib/docker/devicemapper/mnt/918a6f92734
      ##/dev/mapper/docker-8:1-16786822-a7efd4f15c38e906b2d34ce97f95b7a41be3ea3c022f70a12eb530b6d9eea81a   45G dm   xfs         /var/lib/docker/devicemapper/mnt/a7efd4f15c3
      ##/dev/mapper/docker-8:1-16786822-17b847994e2209acbad7f19e00f1f3947dcae6489e6150a269ddbecd367ad68c   45G dm   xfs         /var/lib/docker/devicemapper/mnt/17b847994e2
      ##/dev/mapper/docker-8:1-16786822-3564e748e1476950c051afdbcf924d7a5e513cee79f69ee95649eea864733651   45G dm   xfs         /var/lib/docker/devicemapper/mnt/3564e748e14
/dev/sdc                                                                                            15.6G disk
##/dev/sdc1                                                                                         15.6G part swap        [SWAP]
/dev/sdd                                                                                            15.6G disk
##/dev/sdd1                                                                                         15.6G part xfs         /tmp
/dev/sr0                                                                                          1024M rom
```

3 Operating system requirements

Control Center has tested the 64-bit version of the following Linux distributions:

- Red Hat Enterprise Linux (RHEL) 7.2, 7.3, or 7.4
- CentOS 7.2, 7.3, or 7.4

Kernel versions `3.10.0-327.22.2.el7.x86_64` and higher are tested. For best performance, keep the kernel up-to-date.

The RHEL/CentOS 7 distributions provide a variety of server configurations. Control Center is tested on operating system platforms that are installed and configured with standard options. Docker and Control Center are tested on the Minimal Install configuration with the NFS and Network Time Protocol (NTP) packages installed.

Control Center relies on the system clock to synchronize its actions. The installation procedures include steps to add the NTP daemon to all hosts. By default, the NTP daemon synchronizes the system clock by communicating with standard time servers available on the internet. Use the default configuration or configure the daemon to use a time server in your environment.

Note Because Control Center relies on the system clock, while an application is running, do not pause a virtual machine that is running Control Center.

Networking

On startup, Docker creates the `docker0` virtual interface and selects an unused IP address and subnet (typically, 172.17.0.1/16) 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 through this virtual bridge.

Docker can only check directly connected routes, so the subnet it chooses for the virtual bridge might be inappropriate for your environment. To customize the virtual bridge subnet, refer to Docker's *advanced network configuration* article.

The following list highlights potential communication conflicts:

- If you use a firewall utility, ensure that it does not conflict with Docker. The default configurations of firewall utilities such as *FirewallD* include rules that can conflict with Docker, and therefore Control Center. 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 must restart Docker.
- Even if you do not use a firewall utility, your firewall settings might still prevent communications over the Docker virtual bridge. This issue occurs when `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 a command like the following example would ensure that the bridge works.

Note Before modifying your `iptables` configuration, consult your networking specialist.

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

Additional requirements and considerations

Control Center requires a 16-bit, private IPv4 network for virtual IP addresses. The default network is 10.3/16, but during installation you can select any valid IPv4 16-bit address space.

Before installation, add DNS entries for the Control Center master host and all delegate hosts. Verify that all hosts in Control Center resource pools can

- Resolve the hostnames of all other delegate hosts to IPv4 addresses. For example, if the public IP address of your host is 192.0.2.1, then the `hostname -i` command should return 192.0.2.1.
- Respond with an IPv4 address other than `127.x.x.x` when `ping` *`Hostname`* is invoked.
- Return a unique result from the `hostid` command.

Control Center relies on Network File System (NFS) for its distributed file system implementation. Therefore, Control Center hosts cannot run a general-purpose NFS server, and all Control Center hosts require NFS.

Security

Port information in this section applies only to Control Center. See the documentation for your application for additional port requirements.

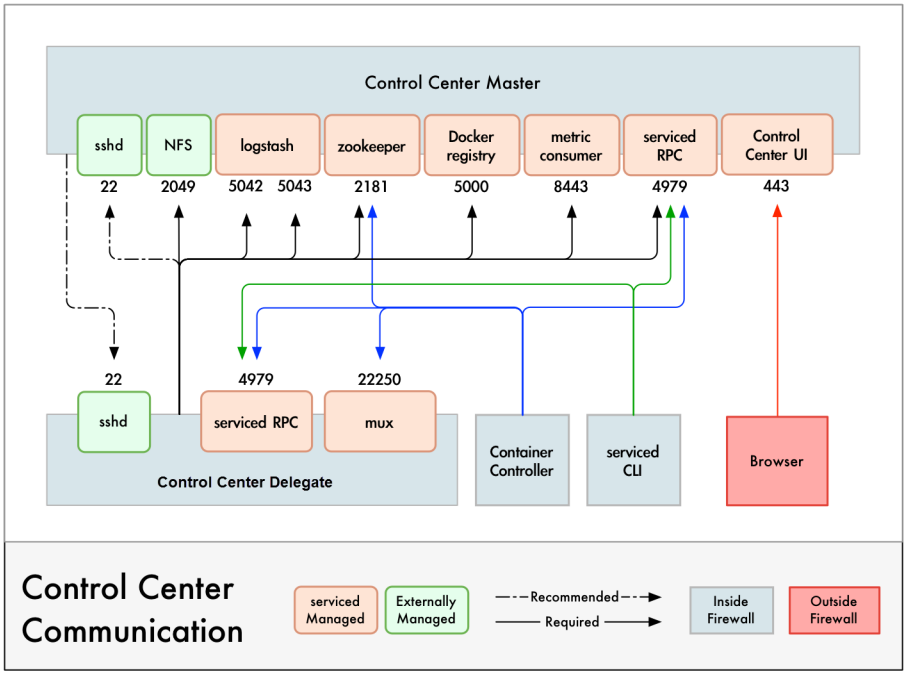
During installation, Control Center has no knowledge of the port requirements of the applications it is to manage, so the installation procedure includes disabling the firewall. After both Control Center and an application are installed, you can close unused ports.

Control Center includes a virtual multiplexer (mux) that performs the following functions:

- Aggregates 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.)
- Along with the distributed file system, enables Control Center to quickly deploy services to any pool host.
- Reduces the number of open ports required on a Control Center host to a predictable set.

The following figure identifies the ports that Control Center requires. All traffic is TCP. Except for port 4979, all ports are configurable.

Figure 3: Port requirements for Control Center hosts



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 delegate hosts must support an incoming UDP connection on port 123.

Additional requirements and considerations

- To install Control Center, you must log in as `root`, or as a user with superuser privileges.
- Access to the Control Center browser interface requires a login account on the Control Center master host. Pluggable Authentication Modules (PAM) is tested. By default, users must be members of the `wheel` group. 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. The script does not modify the `/etc/sysconfig/limits.conf` file.
- Control Center has been tested with *Security Enhanced Linux* enabled.

Delegate host authentication

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

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

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

4 Interface access and packaging

This chapter specifies the tested clients for its browser interface and describes how Control Center is packaged and distributed.

Supported clients and browsers

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

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

Packaging and distribution

Control Center is distributed as a Redhat (`yum`/`rpm`) package. The package is available at for download from Zenoss. The Docker images that Control Center requires are available for download from Zenoss as well.

The Control Center RPM package requires approximately 50MB of storage space. The Docker images for Control Center require approximately 1.35GB of storage space.

* Enterprise mode only; compatibility mode is not tested.

Storage management on Linux hosts A

This appendix includes basic procedures for managing storage on a Linux host.

Identifying storage devices and their configuration

This procedure identifies block storage devices attached to a host and demonstrates how devices are configured.

1. Log in to the target host as `root`, or as a user with superuser privileges through a terminal session.
2. Display the block storage devices attached to the host and their configuration.

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

The following example result shows three disks (`sda`, `sdb`, and `sdc`) and one CD/DVD drive (`sr0`).

- Disk `sda` has two primary partitions:
  - Partition `sda1` is devoted to `/boot`, and formatted with the *XFS* file system.
  - Partition `sda2` includes the following logical volumes, which are managed by LVM:
    - a swap volume, formatted as such
    - a volume for root (`/`), formatted as XFS
    - a volume for `/home`, formatted as XFS

Example result:

```
NAME                     SIZE TYPE FSTYPE      MOUNTPOINT
sda                      128G disk
|-sda1                   500M part xfs         /boot
|-sda2                 127.5G part LVM2_member
  |-centos_c15246-swap  24.8G lvm  swap
  |-centos_c15246-root    50G lvm  xfs         /
  |-centos_c15246-home  52.6G lvm  xfs         /home
sdb                      768G disk
sdc                      768G disk
sr0                     1024M rom
```

For more information about `lsblk`, enter `man lsblk`.

Creating primary partitions

To perform this procedure, you need:

- The password of the `root` user account on a Linux host or a user account that belongs to the `wheel` group.
- A Linux host with at least one local or remote disk.

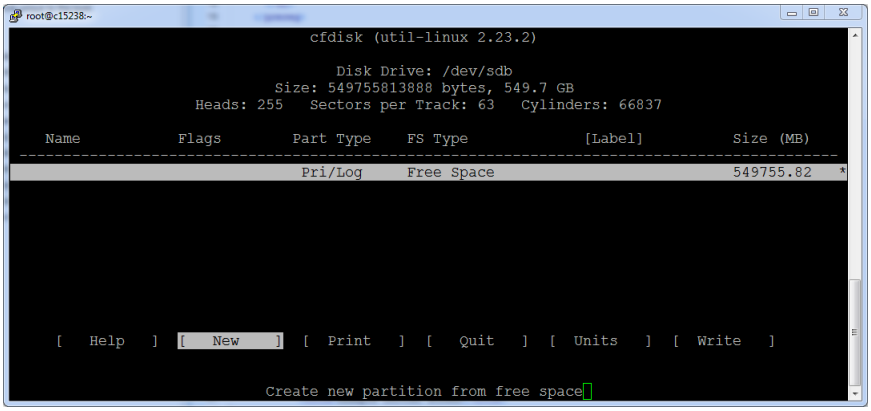
This procedure demonstrates how to create primary partitions on a disk. Each primary partition can be formatted as a file system or swap space, used in a device mapper thin pool, or reserved for future use. Each disk must have one primary partition, and can have up to four. If you are uncertain whether a disk is partitioned, see the preceding topic.

Note Data present on the disk you select is destroyed by this procedure. Before proceeding, ensure that data is backed up or no longer needed.

1. Log in to the target host as `root`, or as a user with superuser privileges.
2. Start the partition table editor for the target disk.
   In this example, the target disk is `/dev/sdb`, and it has no entries in its partition table.

```
cfdisk /dev/sdb
```

Figure 4: Initial screen



The `cfdisk` command provides a text user interface (TUI) for editing the partition table. The following list describes how to navigate through the interface:

- To select an entry in the table, use the up and down arrow keys. The current entry is highlighted.
- To select a command from the menu at the bottom of the interface, use the left and right arrows or **Tab** and **Shift-Tab**. The current command is highlighted.
- To choose a command, press **Enter**.
- To return to the previous level of the menu, press **Esc**.
- To exit the interface, select **Quit** and then press **Enter**.

For more information about `cfdisk`, enter `man cfdisk`.

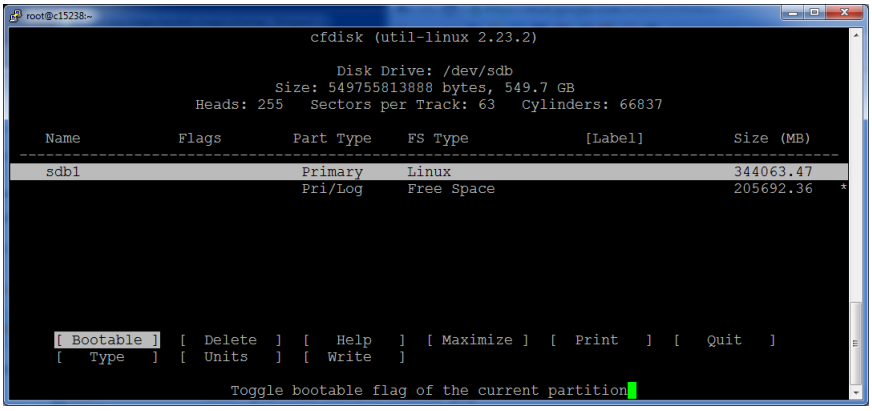
3. Create a new partition.
   Repeat the following substeps to create up to four primary partitions.
   a. Select the table entry with the value **Free Space** in the **FS Type** column.
   b. Select **[New]** and press **Enter**.
   c. Select **[Primary]** and press **Enter**.
   d. At the **Size (in MB)** prompt, enter the size of the partition to create in megabytes, and then press **Enter**.

To accept the default value (all free space on the disk), press **Enter**.

e **Note** If you created a single partition that uses all available disk space, skip this substep.

Optional: Select **[Beginning]** and press **Enter**.

Figure 5: One primary partition



4 Write the partition table to disk, and then exit the partition table editor.

a Select **[Write]** and press **Enter**.

b At the confirmation prompt, enter `yes` and then press **Enter**.

You can ignore the warning about a bootable partition.

c Select **[Quit]** and press **Enter**.

Creating a swap partition

To perform this procedure, you need:

- A host with one or more local disks, with at least one unused primary partition.
- The password of the `root` account on the host or a user that is a member of the `wheel` group.

Perform this procedure to configure a primary partition on a local disk as swap space. Typically, configuring one swap partition or swap file on each local disk maximizes swap space performance.

Note This procedure does not use LVM tools to create a swap space. For more information about LVM, refer to your operating system documentation.

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

2 Identify one or more primary partitions for use as swap space.

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

3 Create and enable swap space on each target primary partition.

a Disable swapping on all swap devices.

```
swapoff -a
```

b Create swap space.

Repeat the following command for each primary partition to use as swap space.

Replace *Device* with the path of a primary partition:

```
mkswap Device
```

c Update the file system table.

Repeat the following command for each swap partition created in the previous substep.

Replace *Device* with the path of a swap partition:

```
echo "Device  swap  swap  defaults  0  0" >> /etc/fstab
```

d Enable swapping on all swap devices.

```
swapon -a
```