Control Center Planning Guide

Release 1.2.0

Zenoss, Inc.

www.zenoss.com

Control Center Planning Guide

Part Number: 1330.16.319

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*.

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 a Control Center cluster.
Control Center Installation Guide	Provides detailed procedures for installing and configuring a Control Center cluster.
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.

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 and part number, and as much information as possible about the context of your feedback. The part number appears at the end of the list of trademarks, at the front of this guide.

Introduction to Control Center 1

This chapter introduces Control Center, an open-source application service orchestrator based on *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 support 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 snapshot and rollback support
- Centralized logging through Logstash and Elasticsearch
- Support for database services and other persistent services
- Encrypted communications among all services and containers
- Delegate host authentication to prevent unauthorized system access

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

A collection of one or more software programs that have been converted into Docker containers.

cluster

The collection of hosts in one or more Control Center resource pools.

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

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 server for the Control Center browser interface. A system can be configured as delegate or master. Only one system in a Control Center cluster 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. The name of the default resource pool is `default`. Control Center uses resource pools (except the default pool) to run services on specific hosts.

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

(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 *cgroups 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 an instance of an application, 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 an LVM thin pool that is created on one or more block devices or partitions, or LVM volume groups.

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.

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 are supported. 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) 7.1 or 7.2 or CentOS 7.1 or 7.2
- support Advanced Encryption Standard (AES)

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 resources

You can deploy Control Center with multiple hosts or a single host.

Note An under-resourced master host cannot function properly. Ensure that you deploy Control Center and applications on a master host that meets the application's minimum requirements.

Multi-host deployments

Best results are achieved in production deployments that 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. The following table lists minimum resource requirements for the master host. Delegate hosts have additional requirements.

Resource	Minimum required
CPU cores ■ 64-bit only ■ real or virtual ■ must support Advanced Encryption Standard (AES)	4
Random-access memory (RAM)	16GB
Network interface controller must support TCP/IP and IPv4	1

Single-host deployments

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 load.

Resource	Minimum required
CPU cores - 64-bit only - real or virtual - must support Advanced Encryption Standard (AES)	4
Random-access memory (RAM)	20GB
Network interface controller must support TCP/IP and IPv4	1

Delegate host 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, refer to your application documentation.

Master host storage areas

The Control Center master host requires the following, separate storage areas:

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 required; the installation procedures include steps for using `serviced-storage` to create 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.

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 using `serviced-storage` to create 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.

In addition, the master host requires a dedicated swap area, and a root filesystem with a minimum of 10GB of free space in `/tmp`.

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 are supported. 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 35GB 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.

Operating system requirements 3

Control Center supports the 64-bit version of the following Linux distributions. All versions of the Linux kernel included in these releases, and all subsequent updates, are supported. However, for best performance, keep the kernel up-to-date.

- Red Hat Enterprise Linux (RHEL) 7.1 or later (7.2 is preferred)
- CentOS 7.1 or later (7.2 is preferred)

The RHEL/CentOS 7.x 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 and support the Minimal Install configuration when the NFS and Network Time Protocol (NTP) packages are also 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 belongs to a cluster.

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.

Possible communication conflicts are as follows:

- 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, however, 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, hosts in a Control Center cluster cannot run a general-purpose NFS server, and all 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.

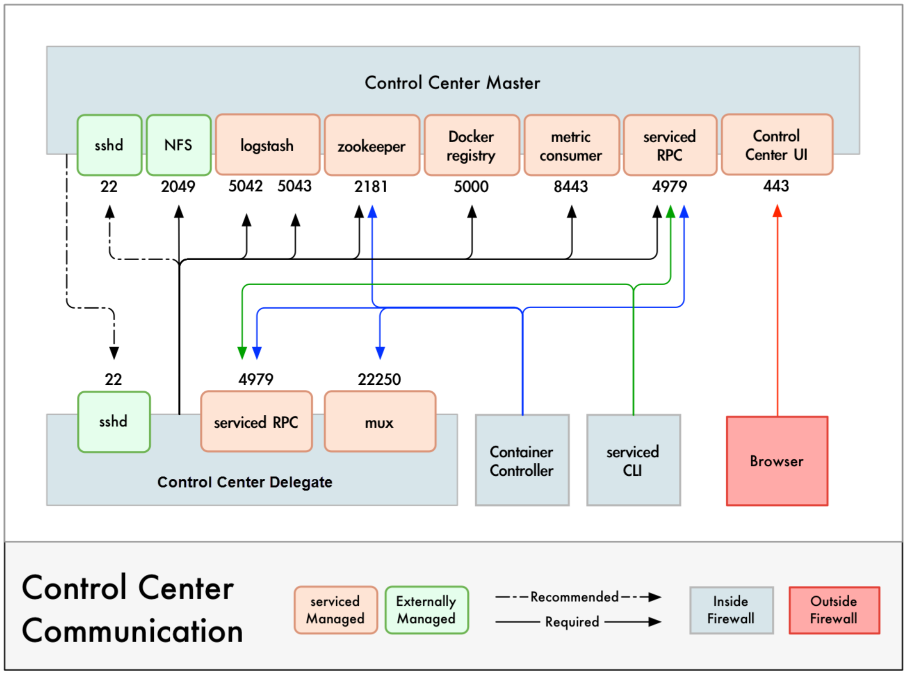
Control Center includes a virtual multiplexer (mux) that performs the following functions:

- 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.

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.

Figure 1: Port requirements for Control Center hosts



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 supported. By default, users must be members of the `wheel` group.
- 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 does not support *Security Enhanced Linux* in enforcing mode. The installation procedures include steps to set the mode to disabled.
- The `firewalld` service can conflict with Docker, and therefore, Control Center and the application it is managing.

Packaging and interface access 4

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

Packaging for customized installations

Control Center is distributed as a Redhat (`yum/rpm`) package. The package is available at public repositories maintained by Zenoss. The Docker images that Control Center requires are available at a public Zenoss *Docker Hub* repository. The default installation process requires internet access. All repositories can be mirrored, supporting offline installations.

The Control Center package requires approximately 50MB of storage space.

Supported clients and browsers

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

Client OS	Supported Browsers
Windows 7 and 8.1	Internet Explorer 11 (Enterprise mode only; compatibility mode is not supported.)
	Internet Explorer 10 (Compatibility mode is not supported.)
	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

Client OS	Supported Browsers
	Chrome 37 and above

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.

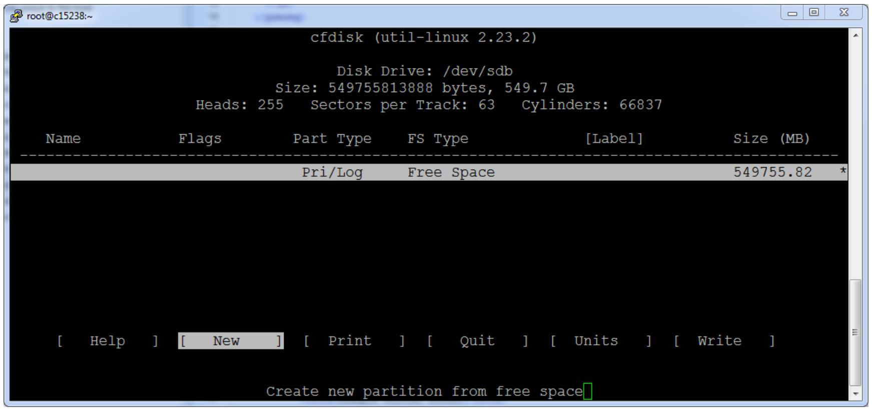
This procedure demonstrates how to create primary partitions on a disk. Each primary partition can be formatted as a file system or swap space, used in a device mapper thin pool, or reserved for future use. Each disk must have one primary partition, and can have up to four. If you are uncertain whether 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 2: Initial screen

   ```
   root@c15238:~
                                cfdisk (util-linux 2.23.2)

                                   Disk Drive: /dev/sdb
                            Size: 549755813888 bytes, 549.7 GB
                    Heads: 255   Sectors per Track: 63   Cylinders: 66837

       Name            Flags          Part Type    FS Type            [Label]          Size (MB)
    ------------------------------------------------------------------------------------------------
                                       Pri/Log     Free Space                           549755.82  *




       [   Help   ]  [   New    ]  [  Print   ]  [   Quit   ]  [  Units   ]  [  Write   ]

                            Create new partition from free space
   ```

   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.

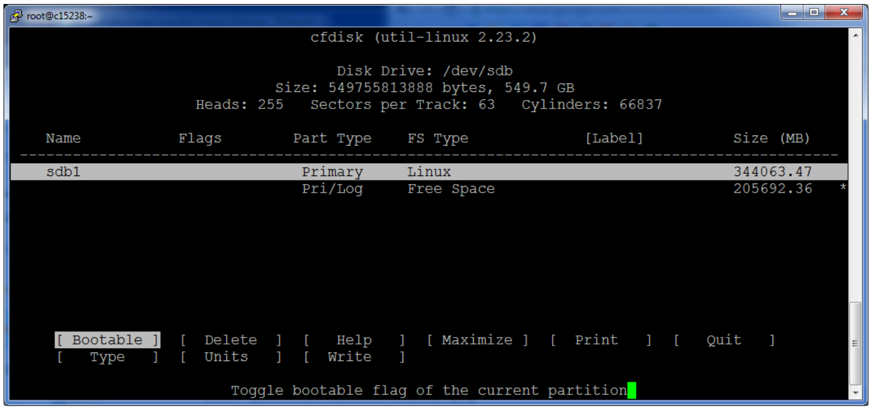
   a. Select the table entry with the value **Free Space** in the **FS Type** column.
   b. Select **[New]** and press **Enter**.
   c. Select **[Primary]** and press **Enter**.
   d. At the **Size (in MB)** prompt, enter the size of the partition to create in megabytes, and then press **Enter**.

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 3: One primary partition



```
                         cfdisk (util-linux 2.23.2)

                            Disk Drive: /dev/sdb
                     Size: 549755813888 bytes, 549.7 GB
               Heads: 255   Sectors per Track: 63   Cylinders: 66837

    Name          Flags        Part Type    FS Type             [Label]          Size (MB)
 ------------------------------------------------------------------------------------------
    sdb1                        Primary     Linux                                344063.47
                                Pri/Log     Free Space                           205692.36  *





     [ Bootable ]  [  Delete  ]  [   Help   ]  [ Maximize ]  [  Print   ]  [   Quit   ]
     [   Type   ]  [  Units   ]  [  Write   ]

                     Toggle bootable flag of the current 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
```