Carbon Black.

Cb Response Server/Cluster

Management Guide

Release 6.2

March 2018

Carbon Black.

Copyrights and Notices

Carbon Black acknowledges the use of the following third-party software in the Cb Response software product:

- Antlr python runtime - Copyright (c) 2010 Terence Parr
- Backbone routefilter - Copyright (c) 2012 Boaz Sender
- Backbone Upload - Copyright (c) 2014 Joe Vu, Homeslice Solutions
- Backbone Validation - Copyright (c) 2014 Thomas Pedersen, http://thedersen.com
- Backbone.js - Copyright (c) 2010–2014 Jeremy Ashkenas, DocumentCloud
- Beautifulsoup - Copyright (c) 2004–2015 Leonard Richardson
- Canvas2Image - Copyright (c) 2011 Tommy-Carlos Williams (http://github.com/devgeeks)
- Code Mirror - Copyright (c) 2014 by Marijn Haverbeke marijnh@gmail.com and others
- D3js - Copyright 2013 Mike Bostock. All rights reserved
- FileSaver - Copyright (c) 2011 Eli Grey.
- Font-Awesome - Copyright Font Awesome by Dave Gandy - http://fontawesome.io
- Fontello - Copyright (c) 2011 by Vitaly Puzrin
- Freewall - Copyright (c) 2013 Minh Nguyen.
- FullCalendar - Copyright (c) 2013 Adam Shaw
- Gridster - Copyright (c) 2012 Ducksboard
- Heredis - Copyright (c) 2009–2011, Salvatore Sanfilippo and Copyright (c) 2010–2011, Pieter Noordhuis
- Java memcached client - Copyright (c) 2006–2009 Dustin Sallings and Copyright (c) 2009–2011 Couchbase, Inc.
- Javascript Digest Auth - Copyright (c) Maricn Michalski (http://marcin-michalski.pl)
- Javascript marked - Copyright (c) 2011–2014, Christopher Jeffrey (https://github.com/chjj/)
- Javascript md5 - Copyright (c) 1998 - 2009, Paul Johnston & Contributors All rights reserved.
- Javascript modernizr - Copyright (c) 2009 - 2013 Modernizr
- Javascript zip - Copyright (c) 2013 Gildas Lormeau. All rights reserved.
- Jedis - Copyright (c) 2010 Jonathan Leibiusky
- Jmousewheel - Copyright (c) 2013 Brandon Aaron (http://brandon.aaron.sh)
- Joyride - Copyright (c) 1998 - 2014 ZURB, Inc. All rights reserved.
- JQuery - Copyright (c) 2014 The jQuery Foundation.
- JQuery cookie - Copyright (c) 2013 Klaus Hartl
- JQuery flot - Copyright (c) 2007–2014 IOLA and Ole Laursen
- JQuery Foundation - Copyright (c) 2013–2014 ZURB, inc.
- JQuery placeholder - Copyright (c) Mathias Bynens http://mathiasbynens.be/
- JQuery sortable - Copyright (c) 2012, Ali Farhadi
- Jquery sparkline - Copyright (c) 2009–2012 Splunck, Inc.
- JQuery spin - Copyright (c) 2011–2014 Felix Gnass [fgnass at neteye dot de]
- JQuery tablesorter - Copyright (c) Christian Bach.
- JQuery timepicker - Copyright (c) Jon Thornton, thornton.jon@gmail.com, https://github.com/jonthornton
- JQuery traffic cop - Copyright (c) Jim Cowart

- JQuery UI - Copyright (c) 2014 jQuery Foundation and other contributors
- jScrollPane - Copyright (c) 2010 Kelvin Luck
- Libcurl - Copyright (c) 1996 - 2014, Daniel Stenberg, daniel@haxx.se.
- libfreeimage.a - FreeImage open source image library.
- Meld3 - Supervisor is Copyright (c) 2006–2015 Agendaless Consulting and Contributors.
- moment.js - Copyright (c) 2011–2014 Tim Wood, Iskren Chernev, Moment.js contributors
- MonthDelta - Copyright (c) 2009–2012 Jess Austin
- Mwheelintent.js - Copyright (c) 2010 Kelvin Luck
- nginx - Copyright (c) 2002–2014 Igor Sysoev and Copyright (c) 2011–2014 Nginx, Inc.
- OpenSSL - Copyright (c) 1998–2011 The OpenSSL Project. All rights reserved.
- PostgreSQL - Portions Copyright (c) 1996–2014, The PostgreSQL Global Development Group and Portions Copyright (c) 1994, The Regents of the University of California
- PostgreSQL JDBC drivers - Copyright (c) 1997–2011 PostgreSQL Global Development Group
- Protocol Buffers - Copyright (c) 2008, Google Inc.
- pyperformance - Copyright 2014 Omer Gertel
- Pyrabbit - Copyright (c) 2011 Brian K. Jones
- Python decorator - Copyright (c) 2008, Michele Simionato
- Python flask - Copyright (c) 2014 by Armin Ronacher and contributors
- Python gevent - Copyright Denis Bilenko and the contributors, http://www.gevent.org
- Python gunicorn - Copyright 2009–2013 (c) Benoit Chesneau benoitc@e-engura.org and Copyright 2009–2013 (c) Paul J. Davis paul.joseph.davis@gmail.com
- Python haigha - Copyright (c) 2011–2014, Agora Games, LLC All rights reserved.
- Python hiredis - Copyright (c) 2011, Pieter Noordhuis
- Python html5 library - Copyright (c) 2006–2013 James Graham and other contributors
- Python Jinja - Copyright (c) 2009 by the Jinja Team
- Python kombu - Copyright (c) 2015–2016 Ask Solem & contributors. All rights reserved.
- Python Markdown - Copyright 2007, 2008 The Python Markdown Project
- Python netaddr - Copyright (c) 2008 by David P. D. Moss. All rights reserved.
- Python ordereddict - Copyright (c) Raymond Hettinger on Wed, 18 Mar 2009
- Python psutil - Copyright (c) 2009, Jay Loden, Dave Daeschler, Giampaolo Rodola'
- Python psycogreen - Copyright (c) 2010–2012, Daniele Varrazzo daniele.varrazzo@gmail.com
- Python redis - Copyright (c) 2012 Andy McCurdy
- Python Seasurf - Copyright (c) 2011 by Max Countryman.
- Python simplejson - Copyright (c) 2006 Bob Ippolito
- Python sqlalchemy - Copyright (c) 2005–2014 Michael Bayer and contributors. SQLAlchemy is a trademark of Michael Bayer.
- Python sqlalchemy-migrate - Copyright (c) 2009 Evan Rosson, Jan Dittberner, Domen Kozar
- Python tempita - Copyright (c) 2008 Ian Bicking and Contributors
- Python urllib3 - Copyright (c) 2012 Andy McCurdy
- Python werkzeug - Copyright (c) 2013 by the Werkzeug Team, see AUTHORS for more details.
- QUnitJS - Copyright (c) 2013 jQuery Foundation, http://jquery.org/
- redis - Copyright (c) by Salvatore Sanfilippo and Pieter Noordhuis
- Simple Logging Facade for Java - Copyright (c) 2004–2013 QOS.ch
- Six - Copyright (c) 2010–2015 Benjamin Peterson
- Six - yum distribution - Copyright (c) 2010–2015 Benjamin Peterson
- Spymemcached / Java Memcached - Copyright (c) 2006–2009 Dustin Sallings and Copyright (c) 2009–2011 Couchbase, Inc.
- Supervisord - Supervisor is Copyright (c) 2006–2015 Agendaless Consulting and Contributors.
- Switchery - Copyright (c) 2013–2014 Alexander Petkov
- Toastr - Copyright (c) 2012 Hans Fjallemark & John Papa.
- Underscore js - Copyright (c) 2009–2014 Jeremy Ashkenas, DocumentCloud and Investigative Reporters & Editors
- Zlib - Copyright (c) 1995–2013 Jean-loup Gailly and Mark Adler

Permission is hereby granted, free of charge, to any person obtaining a copy of the above third-party software and associated documentation files (collectively, the "Software"), to deal in the Software without restriction, including without limitation the rights to

use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notices and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE LISTED ABOVE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Carbon Black, Inc.
1100 Winter Street, Waltham, MA 02451 USA
Tel: 617.393.7400
Fax: 617.393.7499
Email: support@carbonblack.com
Web: http://www.carbonblack.com

Cb Response Server/Cluster Management Guide
Product Version: 6.2.1
Document Revision Date: March 13, 2018

Before You Begin

This preface provides a brief orientation to the *Cb Response Server/Cluster Management Guide*.

Sections

What this Document Covers

This document explains how to manage Cb Response servers and clusters. The following table summarizes the contents of this guide:

	Chapter	Description
1	Server Overview	Provides an overview of the Cb Response server technology stack, daemons, configuration, and logs.
2	Installing the Cb Response Server	Explains how to install/initialize a new Cb Response server, as well as how to upgrade, troubleshoot, and uninstall the server.
3	Server Backup and Restoration	Explains how to perform various backup and restore procedures.
4	Ports and Protocols	Provides a collection of tables that detail port and protocol information for several different server communications
5	Installing a Cb Response Cluster	Introduces Cb Response clusters and explains how to configure clusters, add minions to existing clusters, remove minion nodes from clusters, and upgrade cluster nodes
6	Using CBCLUSTER as a Non-Root User	Describes how to use the CBCLUSTER command as a non-root user.

Other Documentation

Visit the Carbon Black User eXchange website at https://community.carbonblack.com to locate documentation for tasks not covered in this guide as well as other documents maintained as a knowledge base for technical support solutions. Some of these documents are updated with every newly released build, while others are updated only for minor or major version changes. Documents include:

- *Cb Response Release Notes* – Provides information about new and modified features, issues resolved and general improvements in this release, and known issues and limitations. It also includes required or suggested preparatory steps before installing the server.
- *Cb Response Server Sizing Guide (OER)* – Describes performance and scalability considerations in deploying a Cb Response server.
- *Cb Response Server Configuration Guide (cb.conf)* – Describes the Cb Response server configuration file (`cb.conf`), including options, descriptions, and parameters.
- *Cb Response Server/Cluster Management Guide* – (this document) Describes how to install, manage, backup/restore, etc. a Cb Response server/cluster. This guide is for on-premises Cb Response installations only.
- *Cb Response User Guide* – Describes the Cb Response product and explains how to use all of its features and perform administration tasks.
- *Cb Response Unified View User Guide* – Describes how to install and manage Cb Response Unified View.

- *Cb Response Connector Guide* – Describes how to install, configure and maintain various Carbon Black connectors. A connector enables communication between a third-party product and Cb Response server.
- *Cb Response Integration Guide* – Provides information for administrators who are responsible for integrating Cb Response with various tools, such as Cb Protection, EMET, VDI, SSO, and more.
- *Cb Response API* – Documentation for the Cb Response REST API is located at https://developer.carbonblack.com/reference/enterprise-response. Documentation for the Python module that can be used for easy access to the REST API is hosted at https://cbapi.readthedocs.io.

Community Resources

The Carbon Black User eXchange website at https://community.carbonblack.com provides access to information shared by Carbon Black customers, employees and partners. It includes information and community participation for users of all Carbon Black products.

When you log into this resource, you can:

- Ask questions and provide answers to other users' questions.
- Enter a "vote" to bump up the status of product ideas.
- Download the latest user documentation.
- Participate in the Carbon Black developer community by posting ideas and solutions or discussing those posted by others.
- View the training resources available for Carbon Black products.

You must have a login account to access the User eXchange. Contact your Technical Support representative if you need to get an account.

Contacting Support

Carbon Black Technical Support offers several channels for resolving support questions:

Technical Support Contact Options
Carbon Black User eXchange: https://community.carbonblack.com
Email: support@carbonblack.com
Phone: 877.248.9098
Fax: 617.393.7499

Reporting Problems

When you call or email technical support, provide the following information to the support representative:

Required Information	Description
Contact	Your name, company name, telephone number, and email address
Product version	Product name and version number
Hardware configuration	Hardware configuration of the server or computer the product is running on (processor, memory, and RAM)
Document version	For documentation issues, specify the version of the manual you are using. The date and version of the document appear on the cover page, or for longer manuals, after the Copyrights and Notices section of the manual.
Problem	Action causing the problem, error message returned, and any other appropriate output
Problem severity	Critical, serious, minor, or enhancement

Contents

Chapter 1

Server Overview

This chapter provides an overview of the Cb Response server technology stack, daemons, configuration, and logs.

Sections

Server Overview

This section describes the technology stack on a Cb Response server. Five major daemons exist in Cb Response server, as described in the following table:

Daemon	Description
cb-nginx	Used as an HTTP reverse proxy to internal daemons.
cb-coreservices	(Python, Gunicorn) All non-data application logic for HTTP transactions.
cb-datastore	(Java/Jetty) All incoming data, including event logs and binary files.
cb-solr	(Java/Jetty) Apache Solr, the primary data store.
cb-postgres	Traditional relational database.
cb-sensorservices	Handles all non-data sensor requests, such as sensor check-ins, registrations, and upgrades.

nginx is the only daemon with public sockets. The remaining daemons are bound to the Cb Response server using the default IP address, which is 127.0.0.1, and can only be accessed locally or by using the nginx reverse proxy.

nginx owns tcp/80 and tcp/443 and redirects to coreservices, cb-datastore or cb-sensorservices to the Cb Response web root:q based on the URL prefix, as described in the following table:

nginx	Redirects to
/	/var/www/cb/
/api/*	coreservices on tcp/5000
/sensor/*	sensorservices on tcp/6500 and 6501
/data/*	cb-datastore on tcp/9000

Note

coreservices handles /api/*

All /api/ URLs are used by the Cb Response console interface and by REST clients.

sensorservices handles /sensor/*

All /sensor/ URLs are used by the sensors that are pushing data. These URLs are isolated to allow binding a separate nginx server instance to tcp/443 on a public / DMZ interface for sensors that are outside of the internal network (for example, sensors on laptops used by traveling employees, and work laptops that are at employees' homes) without exposing the /api/ interfaces externally. You can isolate these URLs by using a simple nginx configuration change, as shown in the example in the file:

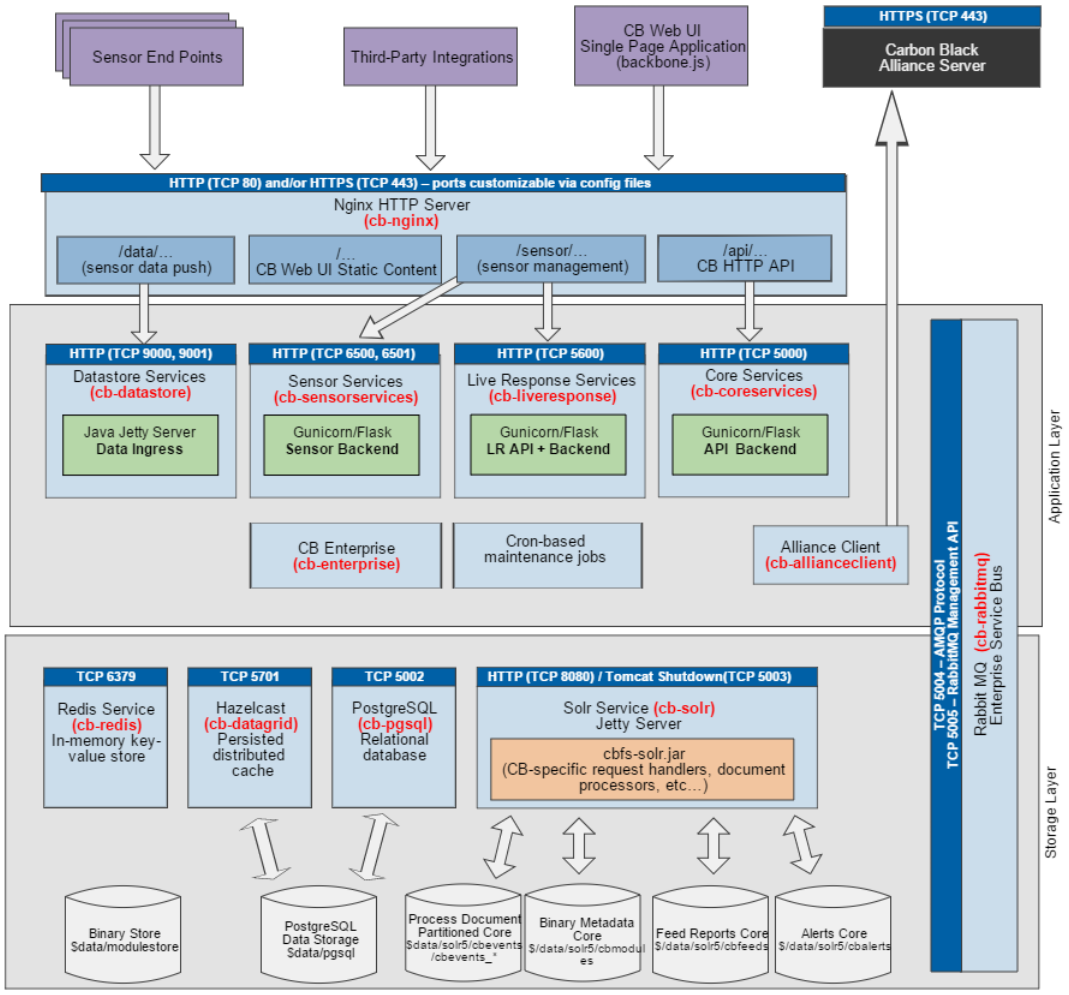
/etc/cb/nginx/conf.d/cb-multihome.conf.example

Note Listening ports are configured differently in a clustered setup. See cluster-specific documentation for more details.

In general, sensors first register and check into `sensorservices` by using `nginx`. If sensors have data, after they check in, they post event logs to `cb-datastore` by using `nginx`.

`cb-datastore` caches data for a few minutes before sending a collection of related data to `cb-solr`.

The following diagram illustrates the Cb Response server architecture at a high level:



Server Configuration

Daemon configuration data is generally static, and is stored in flat files that follow typical Linux conventions. Dynamic run-time configuration data is stored in PostgreSQL and configured by using the Cb Response console.

Most major configuration is done after the Cb Response server is installed, when the `cbinit` script is run. `cbinit` configures a combination of initial settings in both static configuration files and PostgreSQL. For more information about installing and configuring the Cb Response server, see Chapter 2, “Installing the Cb Response Server.”

The following table describes the major static configuration files.

Static Configuration File	Description
`/etc/cb/cb.conf`	Major Cb Response enterprise-wide settings.
`/etc/cb/solr5/core_conf`	This directory stores Apache Solr settings. Each subdirectory has individual solr core configurations.
`/var/cb/data/pgsql/postgresql.conf`	PostgreSQL settings.
`/etc/cb/coreservices-logger.conf`	Configuration of log settings for the Python `coreservices` daemon.
`/etc/cb/sensorservices-logger.conf`	Configuration of log settings for the Python `sensorservices` daemon.
`/etc/cb/cb-datastore/*`	Configuration of log settings for the Java `cb-datastore` daemon.
`/etc/cb/nginx/conf.d/cb.conf`	nginx server settings for defining IP addresses and ports to bind to listeners.
`/etc/sysconfig/iptables`	Standard iptables configuration for the server firewall.

The following table describes the secondary static configuration files:

Static Configuration File	Description
`/etc/cb/allianceclient-logger.conf`	Configuration of log settings for the Cb Response Alliance client daemon.
`/etc/cb/nginx/cb-nginx.conf`	Standard nginx server configurations that are specific to Cb Response.
`/etc/cb/solr5/solr.in.sh`	Jetty servlet settings for Solr

Server Logs

Log Overview

The Cb Response server uses log files extensively. These files provide both reassurance of system health and a record of activity under error conditions.

The following table describes critical logs that are on the Cb Response server:

Log	Description
`/var/log/cb/nginx/access.log` (and `error.log`)	`nginx` HTTP access and error logs for all sensor and API traffic.
`/var/log/cb/coreservices/debug.log`	Application logic for API traffic.
`/var/log/cb/sensorservices/debug.log`	Application logic for sensor traffic.
`/var/log/cb/datastore/debug.log`	Incoming sensor data cache.
`/var/log/cb/solr/debug.log`	Sensor data storage, indexing, and queries.

Troubleshooting

Error in the Cb Response Console Interface

Look in the log file: `/var/log/cb/coreservices/debug.log` for a Python stack trace with details. Contact your Carbon Black Technical Support representative for guidance.

Sensors Are Not Checking in

If you find that a sensor is not checking in, look in the log file: `/var/log/cb/nginx/access.log` for a request from the host in question. For example:

```
164.230.214.13 - - [20/Apr/2017:20:04:52 +0000(3.811)] "POST /
sensor/checkin/35998 HTTP/1.1" 502 166 "-" "sensors.vibrant-
pies.my.carbonblack.io" ">170.16.20.21:6501" "-" "-"
```

In the output example, there are several fields highlighted in red, and these are useful for diagnosing the issue:

- Sensor Id is reported after the checkin field. It is **35998** in the above case.
- The actual error code is reported in the field following HTTP/1.1 text. In the example above, **502** is the error code.
- In a clustered environment, some requests proxy calls to a different minion in the cluster. This minion's address is reported after the '>' character (**170.16.20.21** in the above case).

If you do find a checkin error, or an error to any other call with the "/sensor/" prefix, check the following log for more information about the error:

`/var/log/cb/sensorservices/debug.log`

If you see an error related to requests prefixed with "/data/", check the following log:

```
/var/log/cb/datastore/debug.log
```

In a cluster environment, you will need to look at log files on the node referenced in the nginx error log entry.

Alternatives

If sensors are not checking in but there are no entries in `access.log`, check `error.log`.

If the sensor SSL client certificates are not signed by the Certificate Authority (CA) in `/etc/cb/certs` and configured in `/etc/cb/nginx/conf.d/cb.conf`, `nginx` will refuse the request.

If there are no entries in error.log, check the status of sensor communications as described in the “Troubleshooting Sensors” section in the *Cb Response User Guide*.

Ensure that Everything is Working

Check the `nginx access.log` for ‘`200`’ HTTP response codes. `200` codes indicate that communications are working normally.

You can also check `/var/log/cb/solr/debug.log` for requests to the `/update` handler. For example:

```
INFO: [cbevents] webapp=/solr path=/update
params={wt=javabin&version=2} {add=[a2247de5-fa3b-7b80-0000-
000000000001(1448549632617480192), b6879e21-eff8-dbad-0000-
000000000001 (1448549632695074816), 3abdd29a-018b-3037-0000-
000000000001 (1448549632700317696), f405c732-b898-bedd-0000-
000000000001 (1448549632705560576), ... (203 adds)]} 0 20248
```

“`203 adds`” in this example indicates how many processes were just updated with one or more events. These requests should run smoothly and quickly on a production system.

Chapter 2

Installing the Cb Response Server

This chapter explains how to install/initialize a new Cb Response server, as well as how to upgrade, troubleshoot, and uninstall the server.

Sections

Overview

This chapter describes the steps for installing the Cb Response server. It covers new installations and server upgrades. You can complete the entire process in about ten minutes, assuming reasonable download speed.

The separate *Cb Response Server Sizing Guide* document provides guidelines for hardware and software required for the Cb Response server. Your environment must meet these requirements before you begin installation. See the Carbon Black User eXchange to locate this guide.

A Cb Response server installation consists of these main steps:

1. Obtain and install an RPM from Carbon Black. This RPM does not install the Cb Response server. It sets up a Yum repository and installs an SSL client certificate that allows the full Cb Response server to be downloaded and installed.
2. Install the Cb Response server. This is a two-step process that involves running the `yum install` command and the `cbinit` configuration script. The Cb Response server is downloaded when you run the `yum install` command.

For more information on `cbinit`, see *Automating cbinit* on the Carbon Black User eXchange.

When you have installed the server, you can then install sensors on the endpoints you intend to monitor. Instructions for installing and upgrading sensors are provided in the "Manage Sensors" chapter in the *Cb Response User Guide*.

Firewall and Connectivity Requirements

Internet connectivity through outbound TCP ports is required on the Cb Response Server system for the scenarios described in the following table.

Scenario	Description	Address
Cb Response Yum Repository	The RPM installer sets up a Yum repository.	yum.carbonblack.com:443
Cb Response Alliance Server and Cb Threat Intel	The Alliance Server and Cb Threat Intel provide threat intelligence and can enable further analysis of files on endpoints through Cb Threat Intel partners. To view all threat intelligence data, you need both addresses.	api.alliance.carbonblack.com:443 threatintel.bit9.com:443
CentOS Yum Repository	The standard CentOS Yum repository server used during Cb Response server installation to download standard packages	mirror.centos.org:80

Installing and Initializing a New Cb Response Server

This section describes the procedure for installing and initializing a new Cb Response server.

Caution The steps in this section are for a new installation only. If you already have the Cb Response Server installed, **do not perform these steps**. Instead, see "Server Upgrades and New Sensor Versions" on page 31.

Using the new installation procedure on an existing server will likely result in the loss of all data, including the configuration and event data that is collected from sensors.

Note Root-level permissions are required throughout the entire installation/configuration process. You will use `su` or `sudo` to enter the installation/initialize commands.

To install and initialize a new server:

1. Verify that the host machine on which you intend to install Cb Response server meets the hardware and software requirements specified in the *Cb Response Server Sizing Guide* you received from your Carbon Black representative.
2. Verify that the server has internet connectivity as specified in "Firewall and Connectivity Requirements" on page 20.
3. Contact "Community Resources," to procure an installation RPM for the Cb Response server.
4. Install the RPM:
   a. Run the following command:
      ```
      sudo rpm -ivh carbon-black-release-1.0.0-1.el6.x86_64.rpm
      ```
   b. (Optional) Verify that the Cb Response [cb] Yum repository was configured correctly. You can run this command to see the contents of the new Yum repository entry for Cb Response:
      ```
      cat /etc/yum.repos.d/CarbonBlack.repo.
      ```

```
[root@cb-enterprise-testing ~]# cat /etc/yum.repos.d/
CarbonBlack.repo

[CarbonBlack]
name=CarbonBlack
baseurl=https://yum.distro.carbonblack.io/enterprise/stable/
x86_64/
gpgcheck=1
enabled=1
metadata_expire=60
sslverify=1
sslclientcert=/etc/cb/certs/carbonblack-alliance-client.crt
sslclientkey=/etc/cb/certs/carbonblack-alliance-client.key
```

c. (Optional) You should see the Cb Response SSL certificates and keys in the following directory:

`/etc/cb/certs/`

5. Install the Cb Response server:

a. Verify that your computer's date and time settings are accurate. Incorrect date/time settings can cause failures in SSL negotiation, which is required for Yum downloads.

b. Run the following command:

`sudo yum install cb-enterprise`

```
[bsmith@localhost yum.repos.d]$ sudo yum install cb-enterprise
```

c. Install the CentOS GPG key if you are prompted to do so.

d. If your environment requires that outbound firewall exceptions be made, ensure that the exceptions documented in "Firewall and Connectivity Requirements" on page 20 are followed. You must also update `/etc/yum.repos.d/CentOS-Base.repo` to enable the baseurl of http://mirror.centos.org.

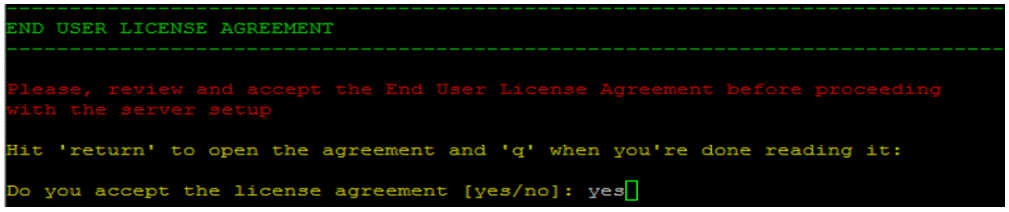
> **Note** Yum supports the use of web proxies. However, Carbon Black is not aware of a way to use Yum with NTLM-authenticated web proxies.

6. When the installation completes, initialize and configure the Cb Response server.

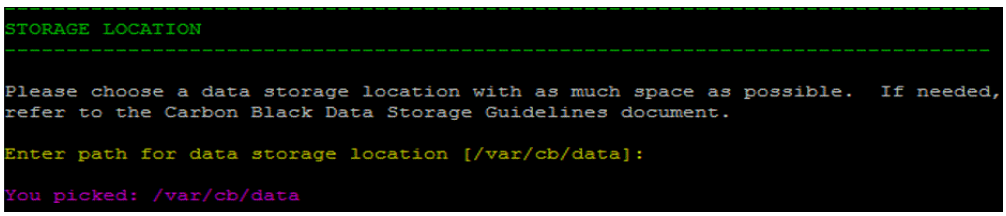
a. Run the following command:

`sudo /usr/share/cb/cbinit`

b. Press **[Return]** to open the EULA. When you are done reviewing it, enter `q` and then enter `yes`.



c. Select a storage location for your data and press **[Return]**.



> ***Note*** Per the “Carbon Black Response Server Sizing Guide”, the primary datastore is mapped by default to `/var/cb/data`.
>
> If you did not configure your storage per the recommendation, review your current file system mapping `(df -h)` with Carbon Black Support or Professional Services. Incorrect or insufficient disk configurations will prevent Cb Response from operating correctly.

d. Enter an initial Adminstrator account to log in and configure Cb Response. Enter values for **Username**, **First Name**, **Last Name**, **E-Mail**, **Password**, and **Confirm password**:

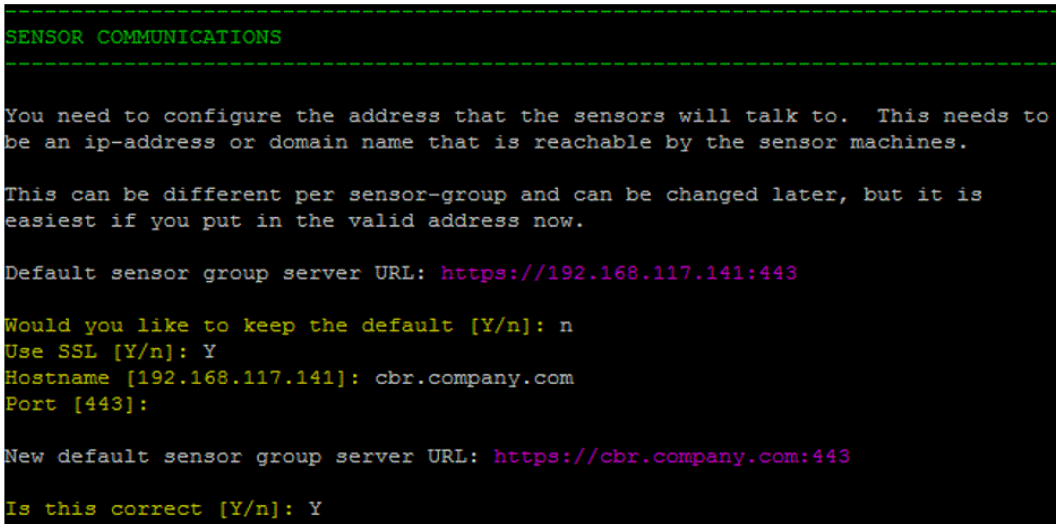
```
-------------------------------------------------------------------------------
ADMINISTRATOR ACCOUNT
-------------------------------------------------------------------------------


Here you configure your GLOBAL ADMINISTRATOR account.
This account is the most powerful account on the server.

Be sure to put a valid e-mail address if you want to take full advantage
of Carbon Black's notification system.

          Username: cbadmin
        First Name: CB
         Last Name: ADMIN
            E-Mail: cbadmin@carbonblack.com
          Password:
  Confirm password:

Verify Account Information:
          Username: cbadmin
        First Name: CB
         Last Name: ADMIN
            E-Mail: cbadmin@carbonblack.com

Is this correct [Y/n]: Y
```

e. Press **[Enter]** and then validate the account information by entering `y`.

f. In the **Sensor Communications** section, you define the address that the sensors will use to communicate back to the Cb Response server:

```
Would you like to keep the default [Y/n]: n
Use SSL [Y/n]: Y
Hostname [192.168.117.141]: cbr.company.com
Port [443]: return

If the Verify Account Information looks correct, Y
```

> **Note** The IP address of the server will be accessed via the default SSL port 433. A best practice is to use a DNS record that points to this IP address.
>
> Work with Carbon Black Support or Professional Services to make sure you understand the external connectivity options supported by Cb Response Server.



g. Review all prompts and configure sharing settings in accordance to your company's security policies. The recommended settings are provided here. You can change these settings at any time by accessing the Cb Response console and selecting your ***username*** **> Sharing Settings** in the top-right corner.

- `Do you want to enable communication with the Carbon Black Alliance? – Y`
  This enables the program to be supplemented with updated threat intelligence from Cb Threat Intel and Carbon Black's extended network of Cb Threat Intel partners.
- `Do you want your server to submit statistics and feedback information to Carbon Black? – Y`
  This enables the server to submit health statistics back to Cb Response. These are used by Carbon Black Support and Professional Services to determine how the allocated server is performing with our application.
- `Do you want the default sensor group to submit hashes to Carbon Black Alliance? – N`
  See the "Threat Intelligence Feeds" chapter in the *Cb Response User Guide* for more information on sharing hashes with Cb Response.

- Continue with current sharing settings? – Y

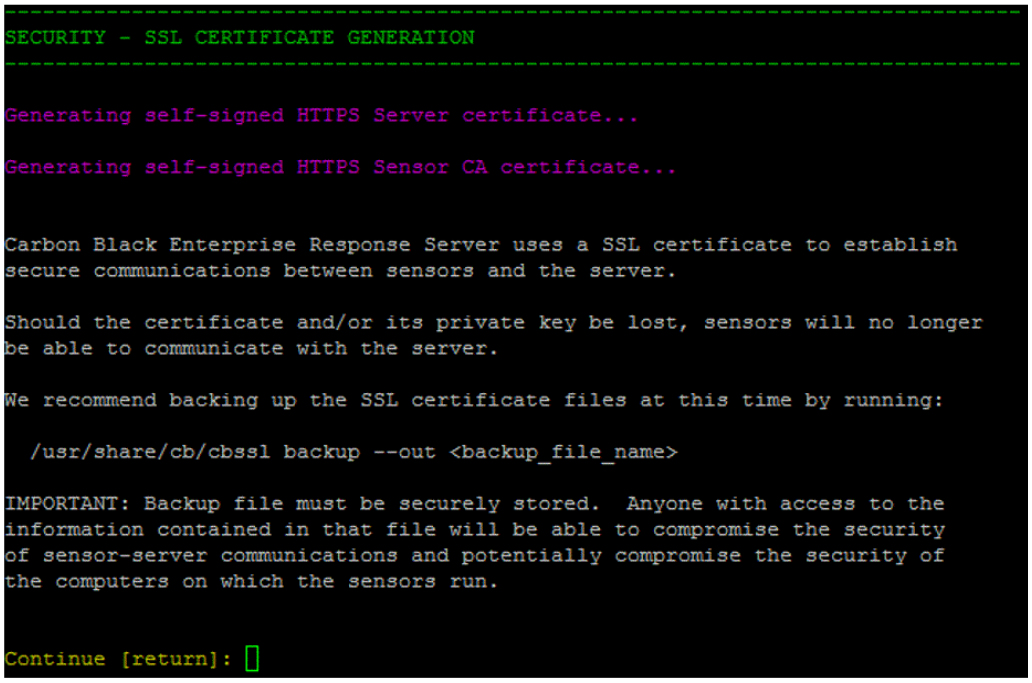
```
--------------------------------------------------------------------------------
CARBON BLACK ALLIANCE
--------------------------------------------------------------------------------


Do you want to enable communication with the Carbon Black Alliance and Threat
Intelligence Servers?

This option controls the main switch which allows the Carbon Black Enterprise
Response Server to connect out and establish communications with Carbon Black
cloud infrastructure.  This is required for a number of features which include
downloading threat intelligence feeds and reporting diagnostics data.
Enabling this switch does not enable any data transmission; each data stream
is controlled by separate, individual settings.

Your server must be able to communicate to
  https://api.alliance.carbonblack.com:443
  https://threatintel.bit9.com:443

Do you want to enable communication with the Carbon Black Alliance? [Y/n]: Y
--------------------------------------------------------------------------------
HELP IMPROVE YOUR CARBON BLACK EXPERIENCE
--------------------------------------------------------------------------------


We are constantly looking for ways to make the Carbon Black user experience
better.  Please help us achieve this goal by allowing automatic reporting of
usage, resource, and sensor statistics to our technology and support teams.

You can later change your mind, too, by going here:

    >>> Administration -> Sharing Settings

Do you want your server to submit statistics and feedback information back to Carbon Black? [Y/n]: Y


Be notified of any binary that could be a potential threat.  Information such as
the filename, MD5 hash and parent process will be shared with the Carbon Black
Alliance partners, including VirusTotal.

All information is anonymized to the extent reasonably practicable before being
shared with Carbon Black Alliance partners. The applicable terms and conditions
are set forth in and subject to your Carbon Black License Agreement. For further
information on what information is collected and shared by the Carbon Black
Alliance Server, please
visit https://www.carbonblack.com/solutions/carbon-black/collaboration/.

You can change this setting at any time in the server web console:

    >>> Administration -> Sharing Settings

If you enable this, you will then be prompted to either enable or disable the
uploading of unknown binaries.

Do you want the default sensor group to submit hashes to Carbon Black Alliance? [Y/n]: N

You have chosen to share data with Carbon Black Alliance
Please review your choices to make sure they are correct

Continue with current sharing settings? [Y/n]: Y
```

h. The **SSL Certificates** section is automated and requires no user input.

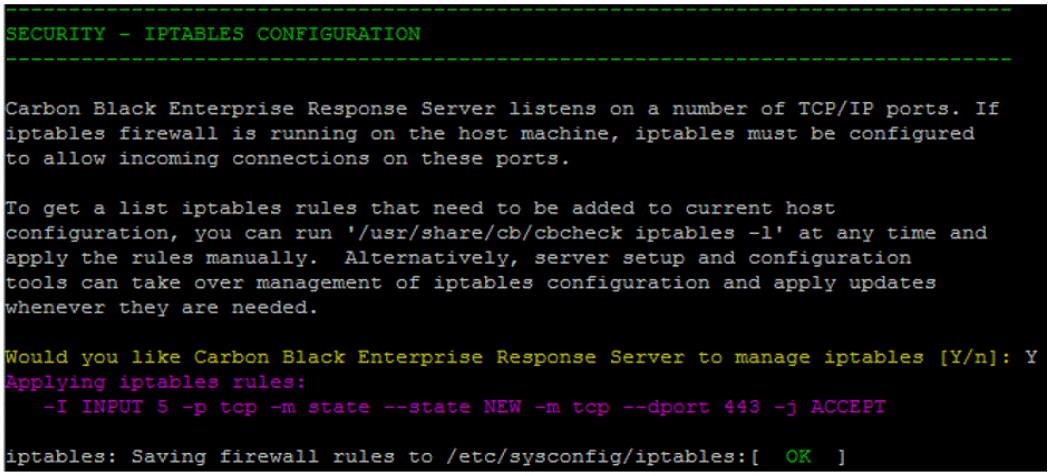
Note You can replace the certificates with those from your organization. Contact Carbon Black Support or Professional Services for more guidance.

Run the following script to create an encrypt backup of your certificates. The exact certificates are critical to disaster recovery efforts.

```
/usr/share/cb/cbssl backup --out
<backup_file_name>
```



i. In the **IP Tables** section, answer `Y`. This opens port 433 in the server's IP tables.



j. The **POSTGRESQL Database Setup** section is automated and requires no user input.

k. In the **Setup Complete** section, enter `Y` to start the services.

```
--------------------------------------------------------------------------------
SETUP COMPLETE!
--------------------------------------------------------------------------------


Server setup has COMPLETED successfully.

Do you want to start the services [Y/n]: Y
Starting cb-supervisord:                                   [  OK  ]
Starting cb-pgsql:                                         [  OK  ]
Starting cb-redis:                                         [  OK  ]
Starting cb-rabbitmq:                                      [  OK  ]
Starting cb-solr:                                          [  OK  ]
   Waiting for cb-solr to build the terms dictionary.
   Depending on index size this may take a while...
Starting cb-coreservices:                                  [  OK  ]
Starting cb-datastore:                                     [  OK  ]
Starting cb-liveresponse:                                  [  OK  ]
Starting cb-allianceclient:                                [  OK  ]
Starting cb-enterprise:                                    [  OK  ]
Starting cb-nginx:                                         [  OK  ]

--------------------------------------------------------------------------------
THANK YOU FOR INSTALLING CARBON BLACK ENTERPRISE RESPONSE!
--------------------------------------------------------------------------------
```

Note To confirm sensor-to-server communications are functioning properly:

1. Open Google Chrome and then launch your server:

   `https://<your_cber_server_url>`

2. Download a sensor and install it on an endpoint.

For more information on installing and managing sensors, see the "Manage Sensors" chapter in the Cb Response User Guide.

7. Configure your firewall if you have not already done so. There are many ways to configure your firewall. The following is just an example.

   a. Open port 443 if you did not allow the `cbinit` script to manage iptables for you.

```
[bsmith@localhost yum.repos.d]$ sudo vim /etc/sysconfig/iptables
     # Firewall configuration written by system-config-firewall
     # Manual customization of this file is not recommended.
     *filter
     :INPUT ACCEPT [0:0]
     :FORWARD ACCEPT [0:0]
     :OUTPUT ACCEPT [0:0]
     -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
     -A INPUT -p icmp -j ACCEPT
     -A INPUT -i lo -j ACCEPT
     -A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
     -A INPUT -j REJECT --reject-with icmp-host-prohibited
     -A FORWARD -j REJECT --reject-with icmp-host-prohibited
     # New additions to the IPTABLES for carbon black
     -A INPUT -m state --state NEW -m tcp -p tcp --dport 443 -j ACCEPT
     COMMIT
```

   b. (Optional) Open port 80 to allow use of web interface and sensor communications through an unsecured channel. This is not required and only recommended for exploration or troubleshooting. Connections to the web interface through port 80 are redirected to port 443.

8. Log into the Cb Response server web user interface at https://<your server address>/ and use the username and password that you set up in the `cbinit` script.

Note Google Chrome is the only supported browser for this release. Although not supported, internal testing indicates that Firefox, Opera, and IE10 or higher should work. However, IE browsers must not be in compatibility mode, and servers in the same subnet as the browser are automatically connected in this mode.

When the Cb Response server is installed, configured, and initialized, it should be accessible through the web interface on port 443 with a self-signed certificate. If you attempt to access the web interface through HTTP on port 80, the connection is redirected to port 443.

The next step, especially in a test environment, is to download and install one or more sensors to begin collecting data. Sensor installation is described in the “Manage Sensors” chapter in the *Cb Response User Guide.*

Upgrading a Cb Response Server

If you are upgrading the server, the procedure varies depending on whether:

- You are upgrading a standalone server or a clustered server.
- The database schema or Cb Threat Intel feed data must be migrated after the new server version is installed.

To upgrade a standalone server:

1. On the server, stop the Cb Response services:
   `sudo service cb-enterprise stop`
2. Update the Cb Response services:
   `sudo yum upgrade cb-enterprise`
3. Restart the Cb Response services:
   `sudo service cb-enterprise start`

To upgrade a clustered server:

1. On the Master server, navigate to the `cb` install directory (defaults to `/usr/share/cb`) and stop the Cb Response services:
   `sudo cbcluster stop`
2. Update the Cb Response services on all nodes:
   `sudo yum update cb-enterprise`
3. On the Master server, restart the Cb Response services:
   `sudo cbcluster start`

Upgrades to the Cb Response server will occasionally require you to use a utility called `cbupgrade` after executing `yum update cb-enterprise` to migrate the database schema or Cb Threat Intel feed data. The operator will be notified of this requirement when attempting to start the cb-enterprise services. In a clustered-server configuration, you must run the `cbupgrade` tool on all nodes before restarting the cluster. When running this utility in a clustered environment, be sure to answer `No` when asked to start the Cb Response services. You will need to use `cbcluster` to start the clustered server.

Server Upgrades and New Sensor Versions

A new server version might include new versions of the sensor. Check the current *Cb Response Release Notes* or contact support@carbonblack.com if there are any questions about this.

If a new sensor version is included, you must decide if you want to 1) only install server updates and not update all sensors or 2) only install server updates. Carbon Black recommends the first option. See the “Manage Sensors” chapter in the *Cb Response User Guide* for platform-specific sensor upgrade instructions.

Supporting Multiple Volumes for Event Data

This section explains how customers can add more storage to their existing Cb Response deployment after upgrading to the latest release. This involves adding multiple Solr data directories for cbevents cores. These directories can be added as mount points into new storage arrays, so that you can easily add more disk space. This can be easily configured. Basically, if you need more disk space, you attach a new volume, mount it into the Solr data directory, and the server starts using it automatically.

Naming Conventions

Solr uses new `cbevents` directories (mount points) if their name is prefixed with:

`cbevents*`

or

`_cbevents*`.

Warning The cbevents directory (without the suffix) is the default directory but does not need to remain on the original data partition. You can remove it if needed.

The following is an example of a valid multi-volume configuration:

```
[root@ip-172-31-14-184 solr5]# df
Filesystem     1K-blocks     Used Available Use% Mounted on
/dev/xvda1      32895856 10341996  20860168  34% /
tmpfs           31389104        0  31389104   0% /dev/shm
/dev/xvdb      206293688 28033360 167758184  15% /data
/dev/xvdf      206293688 35809668 159981876  19% /data/solr5/cbevents2
/dev/xvdg      226936188 63976332 151409136  30% /data/solr5/cbevents3
```

In this example, the default data drive is mounted to `/dev/xvdb`, and /data is configured as the data root inside of `cb.conf`. In addition, two more volumes are added and mounted to `/data/solr5/cbevents2` and `/data/solr5/cbevents3`.

Warning The system assigns the correct `user:group` upon cb-enterprise restart. If you created the mount points on a live server, ensure that the user assigned to the Cb Response server has write permissions on the mounted directory. Failure to do so causes the system to ignore the new mount points.

Another option to expand cbevents storage is to use symlink as follows:

1. Create a mount point in another location in the file system, such as `/data2`.
2. Create a symlink to the `cbevents*` directory inside the `solr5` directory that points to the mounted directory. For example:

   ```
   ln -s /data2 /var/cb/data/solr5/cbevents2
   ```
3. Ensure that the Cb Response user has write permissions in the mounted directory (`/data2`).

Using New Data Directories

This section discusses partitioning and purging relating to new data directories.

Partitioning

New data directories are used when the next partition occurs (every three days by default) or sooner if the current data disk is at risk of becoming full. The server uses simple heuristics in calculating when to partition and where to place the new event partition:

1. A new partition is created in the `cbevents*` directory with the most free space at the time of partitioning.
2. If the current data volume is more than 95% full and additional partitions exist that have more than 5% free space available, the server immediately partitions.

   You can control this threshold using the following configuration parameter:

   `SolrTimePartitioningFreeSpaceThresholdPerc`

- Rule 1 ensures that new volumes are used in a balanced fashion. As old data is aging out (being purged), some partitions free up. This ensures that free space is optimally used.
- Rule 2 ensures that the system uses fragmented disk space efficiently in case many `cbevents*` directories exist. For example, assume you have five volumes, and each has 20% free space. This could result in none of the volumes fitting into the three-day partition. The system will continue trying to use one of the partitions (up to its maximum available space) before moving to the next one. As a result, the server might end up with smaller partitions. However, this scenario should be rare.

Active and Read-Only Directories

Any `cbevents` directories prefixed with `cbevents*` will be used to create new cbevent partitions.

Any `cbevents` directories prefixed with `_cbevents*` will be used as read-only. They can be used to load existing partitions, but new partitions will not be created on it. This approach can be used when retiring old volumes. Old partitions will eventually be purged based on time. A second use for `_cbevents` prefix is for directories that are used only for "cold" partitions (old partitions that will only be loaded on demand).

Partition Purging

The system purges partitions based on disk space, time, or the maximum number of allowed partitions.

When purging based on disk space, a purging algorithm considers the overall amount of free disk space. For example, assume three 100 GB volumes exist, each with 30 GBs of free space. This gives you a total of 90 GBs of free space and a total disk space of 300 GBs. The total event data size is the sum of index sizes on all three volumes. (This could be less than 210 GBs since the main data volume may also contain store files and other data.)

The following shows how the current purging thresholds (in `cb.conf`) are interpreted when multiple volumes exist:

- `MaxEventStoreSizeInPercent` - Purge the oldest partition when the total sum of all event core sizes exceeds the given percentage of a total disk space (on all volumes).

- `MaxEventStoreSizeInMB` - Purge the oldest partition when the total event store size (on all volumes) exceeds the given threshold.
- `MinAvailableSizeInMB` - Purge the oldest partition when the total free disk space (on all volumes) falls below the given threshold.

Extending Disk Space on the Fly

You can add disk space on the fly without having to restart their Cb Response server. New directories are automatically used when a new partition occurs, avoiding any server downtime. As mentioned above, make sure that directory is given correct read+write permissions to the cb user.

Troubleshooting the Server

The following table shows the Cb Response server logs that are found in `/var/log/cb` and organized into subdirectories by component. Cb Response Server Logs:

Component	Description
allianceclient	The Alliance client communicates with the Cb Response Alliance server.
audit	Contains log files for the following activities: banning, sensor isolation, and live response. Also, if `EnableExtendedApiAuditLogging` is enabled in `cb.conf`, this directory also includes a user activity log file based on user-generated API calls in the console..
cbfs	Was the location of the datastore engine in earlier versions of Cb Response but is no longer used in versions 5.0.0 and later.
cbfs-http	Contains log files of the second generation of the Java datastore engine.
cli	Contains events pertaining to the Cb Response service commands used at the server console level.
coreservices	Provides access to functionality via web APIs to both the web interface and sensors. Nearly all interface issues should result in log entries for coreservices.
sensorservices	Provides entry-point for sensor registrations and checkins. Look for issues here if there are problems with sensor connectivity
datastore	Used for core event data processing and managing incoming sensor data.
enterprise	Used for event logging of the Cb Response service.
job-runner	The Cb Response server uses cron jobs to provide various scheduled maintenance, data trimming, and similar tasks.
liveresponse	Used to hold Cb Live Response session-related events.
nginx	The reverse proxy and SSL termination point for the Cb Response server.

Component	Description
notifications	The location of the syslog output for feeds and watchlists.
pgsql	The Cb Response server uses Postgres SQL to store administrative data. Event data gathered from the sensors is not stored in Posgres.
rabbitmq	The logging location for the rabbitmq component of the Cb Response server.
redis	The logging location for the redis component of the Cb Response server.
services	The logging location for the start/stop services of the Cb Response server.
solr	Used for indexes and stores data.
supervisord	The supervisord process utility is used to manage Cb Response server processes, handling startup and shutdown dependencies between the various server components and services.

The following table shows the scripts found in `/usr/share/cb`, most of which are diagnostics scripts; it only includes scripts that are used for diagnostics

Component	Description
cbbanning	Assists in managing the Cb Response server banning features. To get a list of available commands, run this command: `cbbanning commands`
cbstats	This utility provides access to the statistics collected by the Cb Response server.
cbsyslog	Provides an interface for testing Cb Response's notifications syslog output.
cbpost	This utility is used to send file(s) to the Alliance server; typically used during interaction with Carbon Black Technical Support.
py_runtime_info	Generates a runtime report that shows the stack trace, process memory map, and open file descriptors for the running Cb Response processes.
cbfeed_scrubber	Helps clean up feed tags on existing Solr documents.
cbinit	Used to configure a combination of initial settings during a Cb Response server installation.
cbdiag	Dumps verbose troubleshooting information, including logs and configuration, to a gzip archive. This file can be analyzed offline or provided to Carbon Black with support requests.
sql_stats	Contains outputs of various SQL database statistics; typically used during troubleshooting.
cbsolr	Used for indexes and stores data.

Component	Description
cbget	This utility is used to download or list files from Alliance server; typically used during interaction with Carbon Black Technical Support.
sensor_report	Generates a report that shows the status of every sensor communicating with Cb Response server. Optionally, it can be used to identify specific sensors that might require the attention of IT support personnel.

cbcheck	Assists in troubleshooting Cb Response server installation. To get a list of available commands, run this command:
	`cbcheck commands`
	to learn more about a specific command, run this command:
	`cbcheck <command> -h`

cbcluster	Used to manage clusters (not a diagnostic tool).
cb_rabbitmq-server.sh	This is a system utility and should never be run manually.
cbrabbitmqctl	A command-line interface that provides access to the Cb Response rabbitmq service.
pgsql_diag.sh	Prints diagnostic info about the CBER Postgres database
cbpasswd	Resets user's password. Can only be run as root.

Uninstalling a Cb Response Server

This section describes the procedure for uninstalling a Cb Response server from RHEL/CentOS.

You must remove the following Cb Response packages:

- `cb-enterprise`
- `python-cb-werkzeug`
- `python-cblib`
- `python-cb-coreservices`
- `cb-datastore`
- `python-cb-pysaml2`
- `cbui`
- `cb-swagger`
- `carbon-black-release`

Using the Yum utility, you can uninstall a Cb Response server in one of two ways. You can remove or erase the packages that constitute the Cb Response server. Instructions for both methods are provided here.

Using the Yum Utility to Remove the Cb Response Server

You can use the Yum utility to remove only those RPM packages that compromise the Cb Response software:

To uninstall a server:

1. Stop the Cb Response services with one of the following commands:

   ```
   sudo /usr/share/cb/cbcluster stop
   ```

   or

   ```
   sudo service cb-enterprise stop
   ```

2. Use the following Yum utility command to remove the previously listed Cb Response packages.

   ```
   sudo yum remove <package1> <package2> <packageN>
   ```

3. Manually remove the following `cb` directories:

   - `/var/www/cb/`
   - `/var/run/cb/`
   - `/var/log/cb/`
   - `/var/lib/cb/`
   - `/var/cb/`
   - `/usr/share/cb/`
   - `/usr/lib/python2.6/site-packages/cb/`
   - `/etc/cb/`

Using the Yum Utility to Erase the Cb Response Server and All its Dependencies

You can use the Yum utility to remove Cb Response packages as well as all other third-party packages upon which it depends.

Warning This procedure might remove packages that are also required by other software applications. Use caution when performing this procedure.

To uninstall a server:

1. Stop the Cb Response services with one of the following commands:

   ```
   sudo /usr/share/cb/cbcluster stop
   ```

   or

   ```
   sudo service cb-enterprise stop
   ```

2. Access the `yum.conf` file with the following command:

   ```
   vi /etc/yum.conf
   ```

3. Add this line to `yum.conf`:

   ```
   clean_requirements_on_remove=1
   ```

4. Enter the following:

   ```
   yum erase cb-enterprise
   ```

5. Enter the following:

   ```
   yum remove carbon-black-release
   ```

6. Manually remove the following `cb` directories:
   - `/var/www/cb/`
   - `/var/run/cb/`
   - `/var/log/cb/`
   - `/var/lib/cb/`
   - `/var/cb/`
   - `/usr/share/cb/`
   - `/usr/lib/python2.6/site-packages/cb/`
   - `/etc/cb/`

Chapter 3

Server Backup and Restoration

This chapter explains how to perform various backup and restore procedures.

Sections

Overview

This chapter provides procedures for backing up and restoring a Cb Response server. The procedures are designed to ensure a minimal amount of data loss in the event of a catastrophic failure.

Backup files and data should be stored on a different server than the one that is used for daily operations.

All procedures in this document are performed at the command prompt and require root-level access.

Note `/var/cb` is assumed to be the default installation data path throughout this appendix. If you have your datastore root configured in a different location, use that in place of what is provided here.

Note Network integrations will not be fully backed up. Any network integration needing a bridge or connector installed must be installed on the new restore server before restoring the configurations. Configuration items for network integrations will be restored throughout this appendix.

Restoration Servers

For all restoration options, it is assumed that a new server install has been performed with the same number of master and minion systems, each with the same number of Solr data shards in place on each system and configured to use the same hostname and IP addresses.

The same server version must also be used between backed up and restored systems.

To install a new system, see “Installing the Cb Response Server” on page 19.

All installation steps involve running `cbinit` (and `cbcluster add-node` for clustered systems). These must be completed before performing a system restoration.

Backup/Restore Script

An open-source script exists as part of the Carbon Black Developer Community that handles some of the following documented backup and restore tasks and saves you some typing. The use of this script is not covered in this appendix. If interested in this script, read through the script documentation stored on Github at https://github.com/cbcommunity/cb-administration-scripts/tree/master/backup_restore.

Backup

This section highlights two backup and restore methods:

- A configuration-only backup covers all configuration and sensor metadata for the system. It is designed to be minimal in storage size and allows the quickest return to operation in the event of a catastrophic server failure. It does not include any data that the sensors submit (event or binary data) or any feed or alert information from a running server. The configuration backup step is required for both backup and restore methods. For more information, see "Configuration Backup" on page 41.
- A data backup is a continuation of the configuration backup and includes all data stored by the server. Backing up this data ensures a full recovery of the system. Off-server storage requirements are much larger due to the full data set that is being backed up. Also, the time it takes to perform a restoration will likely be longer due to copying and unarchiving large amounts of data to a new system.

> **Note** `/cbdata/_backup/ServerName` is used in this procedure as an example for a backup data storage location. Determine the correct location for storing your backups and note the size requirements if also performing a data backup.

You must run all commands on the master and minion systems unless otherwise noted. Perform all steps on all standalone servers. For more information, see "Data Backup" on page 43.

Configuration Backup

A configuration backup only backups up the files and data required for a system restore with functioning sensors and without recorded data. This procedure is required for all backup options but can stand alone on its own. It is the quickest way to capture and restore data and takes up the smallest amount of disk space. Configuration backups can be performed with Cb Response in a running state.

To perform a configuration backup:

1. Change directories to the backup location:

   ```
   cd /cbdata/_backup/ServerName
   ```

2. Backup the configuration files by running these commands:

   a. Host file:

   ```
   tar -P --selinux -cvf cbhosts.tar /etc/hosts
   ```

   b. Yum files:

   ```
   tar -P --selinux -cvf cbyum.tar /etc/yum.repos.d sfd
   ```

   c. IP tables:

   ```
   tar -P --selinux -cvf cbiptables.tar /etc/sysconfig/iptables
   ```

   d. SSH configuration and keys:

   ```
   tar -P --selinux -cvf cbssh.tar /etc/ssh/
   ```

   e. Cb Response configuration:

   ```
   tar -P --selinux -cvf cbconfig.tar /etc/cb/
   ```

   f. Rsyslog configuration:

   ```
   tar -P --selinux -cvf cbrsyslog.tar /etc/rsyslog.conf
   ```

g. Rsyslog.d configuration:

```
tar -P --selinux -cvf cbrsyslogd.tar /etc/rsyslog.d/
```

h. Logrotate configuration:

```
tar -P --selinux -cvf cblogrotate.tar /etc/logrotate.d/cb
```

i. RabbitMQ cookie:

```
tar -P --selinux -cvf cbrabbitmqcookie.tar /var/cb/
.erlang.cookie
```

j. RabbitMQ node configuration:

```
tar -P --selinux -cvf cbrabbitmqnode.tar /var/cb/data/
rabbitmq
```

k. (Optional) SSH authorization keys:

> **Note** Perform this step only if you use trusted keys between systems in a cluster environment.

```
tar -P --selinux -cvf cbrootauthkeys.tar /root/.ssh/
authorized_keys
```

l. (Master only) Syslog CEF template:

```
tar -P --selinux -cvf cbceftemp.tar /usr/share/cb/
syslog_templates
```

m. (Optional - master only) Cb installer backups:

> **Note** Perform this step only if you manually installed additional versions of the sensor.

```
tar -P -selinux -cvf cbinstallers.tar /usr/share/cb/
coreservices/installers/
```

n. (Optional - master only) Custom syslog templates:

> **Note** Perform these steps only if you use custom syslog templates that are not stored in `/user/share/cb/syslog_templates`.

i. Locate all custom syslog templates paths in use by searching the following:

`/etc/cb/cb.conf file for any instance of SyslogTemplate=`

For example:

`WatchlistSyslogTemplateBinary`

and

`FeedIngressSyslogTemplateBinary`

ii. Take note of the file path after =.

For example:

```
WatchlistSyslogTemplateBinary=/var/custom/syslog/
watchlist_binary_custom.template
```

The path for this example would be `/var/custom/syslog`

iii. Tar each custom path that is identified using this command:

```
tar -P --selinux -cvf syslog_custom1.tar /var/custom/syslog
```

3. (Master only) Perform a backup of the Postgres Database:

> **Note** Perform this configuration only if you are performing a full backup. Otherwise, skip this step and execute the step for backing up a Postgres Database in "Data Backup" on page 43.

   a. Backup configuration:

```
pg_dump -C -Fp -f psqldump_config.sql cb -p 5002 \
--exclude-table-data=allianceclient_comm_history \
--exclude-table-data=allianceclient_uploads \
--exclude-table-data=allianceclient_pending_uploads \
--exclude-table-data=banning_sensor_counts \
--exclude-table-data=binary_status \
--exclude-table-data=cb_useractivity \
--exclude-table-
data=detect_dashboard_average_alert_resolution_history \
--exclude-table-data=detect_dashboard_binary_dwell_history \
--exclude-table-data=detect_dashboard_host_hygiene_history \
--exclude-table-data=investigations \
--exclude-table-data=maintenance_job_history \
--exclude-table-data=moduleinfo_events \
--exclude-table-data=mutex_watchlist_searcher \
--exclude-table-data=sensor_activity \
--exclude-table-data=sensor_comm_failures \
--exclude-table-data=sensor_driver_diagnostics \
--exclude-table-data=sensor_event_diagnostics \
--exclude-table-data=sensor_licensing_counts \
--exclude-table-data=sensor_queued_data_stats \
--exclude-table-data=sensor_resource_statuses \
--exclude-table-data=server_storage_stats \
--exclude-table-data=storefiles \
--exclude-table-data=tagged_events
```

   b. Backup users and groups:

```
pg_dumpall -p 5002 --roles-only -f psqlroles.sql
```

4. Copy the backup location off to a remote location.

Data Backup

A data backup captures all recorded data that is stored on a server and is required to complete a full restore of a functioning system. You must complete a "Configuration Backup" on page 41 before performing the steps in this section. The data captured here can be very large, depending on the amount of data retained by the system/cluster. Full backups require placing Cb Response in a stopped state.

To perform a data backup:

1. Stop all Cb Response services by running these commands:

   a. In a clustered server environment, on the master only, run:

```
/usr/share/cb/cbcluster stop
```

   b. In a standalone server environment, run:

```
service cb-enterprise stop
```

2. Change directories to the backup location:

   ```
   cd /cbdata/_backup/ServerName
   ```

3. Backup the Solr database by running this command:

   ```
   tar -P --selinux -cvf cbsolr.tar /var/cb/data/solr5/
   ```

4. Backup the module store:

   ```
   tar -P --selinux -cvf cbmodulestore.tar /var/cb/data/
   modulestore/
   ```

5. (Master only) Backup Postgres Database:

   a. Start the Postgres service:

      ```
      service cb-pgsql start
      ```

   b. Backup the Postgres database:

      ```
      pg_dump -C -Fp -f psqldump_full.sql cb -p 5002
      ```

   c. (This is a duplicate step from “Configuration Restore” on page 46.) Backup users and groups:

      ```
      pg_dumpall -p 5002 --roles-only -f psqlroles.sql
      ```

   d. Stop the Postgres service:

      ```
      service cb-pgsql stop
      ```

6. Start the Cb Response server by running these commands:

   a. In a clustered server environment, on the master only, run:

      ```
      /usr/share/cb/cbcluster start
      ```

   b. In a standalone server environment, run:

      ```
      service cb-enterprise start
      ```

7. Copy the backup location off to a remote location.

Restore

In order to restore a system, the following prerequisites must be met:

- A fresh Cb Response server (or an old snapshot) installation must be available on which to restore the backup files.
- The server must have the same configuration for the master and minions (for a clustered environment).
- The same hostname and IP addresses must be used.
- For clusters, the same number of Solr data shards must be in place on the correct minion servers before the configuration and data files can be restored to the server. For standalone servers, the same number of Solr data shards must be in place for the standalone server.
- The new server(s) must be installed on the same server version of the server(s) from which the backups were taken (for example, v5.2.0-4 -> v5.2.0-4).

If detailed configuration items are required to complete the install, use the following files, which are generated in “Configuration Backup” on page 41, to obtain the configurations from the backed-up server:

- The IP address used for each system. In `cbhosts.tar`, this is in `/etc/hosts`.
- The number of systems used and the master/minion assignment. In `cbconfig.tar`, this is in the `/etc/cb/cbcluster.conf` file.

- The number of Solr shards and the server assignment. In `cbconfig.tar`, this is in the `/etc/cb/cluster.conf` file.

To install a new system, see "Installing the Cb Response Server" on page 19. All installation steps that involve running `cbinit` (and `cbcluster add-node` for clustered environments) must be completed before performing a server restore. The configuration items chosen when running `cbinit` are overwritten after restoration of the configurations files is complete.

Failed Minion Cluster Restore

The procedure in this section helps you set up a fresh server install to restore one or more failed minions to an otherwise operational cluster.

The following prerequisites must be met before performing this procedure:

- The master server (at least) must still be functioning and backup data for the failed minions must be available.
- The servers on which the restored minions will be installed should be at the same base operating system level as the rest of the cluster with no Carbon Black software installed.

To set up a fresh server install to restore one or more failed minions to an otherwise operational cluster:

1. Stop the cluster:

   ```
   /usr/share/cb/cbcluster stop
   ```

2. Start the Postgres service:

   ```
   service cb-pgsql start
   ```

3. Determine the `node_id` for missing minon(s) by querying Postgres:

   ```
   psql -p 5002 cb -c "select * from
   cluster_node_sensor_addresses;"
   ```

4. Delete the row from the `cluster_node_sensor_address` table, where the `node_id` is equal to the failed minion(s):

   `psql -p 5002 cb -c "delete` from `cluster_node_sensor_addresses` where `node_id = 2;"` where `2` is the number from the missing minion

5. Stop Postgres service:

   ```
   service cb-pgsql stop
   ```

6. Edit the `/etc/cb/cluster.conf` file as follows:

   a. In the `[Cluster]` section of the file, subtract `N` from `NodeCount` and NextSlaveAutoInc lines where `N` is the number of failed minions that are being restored.

   b. In the `[Master]` section of the file, add the the shard numbers from the `ProcSolrShards` line of the failed minions to the master's `ProcSolrShards` in a comma-separated list.

   c. Delete the `[SlaveN]` section for each failed minion.

   d. Save the file.

7. Add the minion(s) like you would as part of a normal install by running:

   ```
   /usr/share/cb/cbcluster add-node
   ```

   and providing the appropriate IP address or hostname and shard information.

8. The master will install `cb-enterprise` on the minion system and allow for normal operation.
9. Copy over the backup data generated from the backup process and follow the restore procedures in the next sections.
10. Start the cluster:

    ```
    /usr/share/cb/cbcluster start
    ```

Configuration Restore

This section contains a procedure for restoring the configuration files and data files. The restore should match what was performed in the backup procedures in “Backup” on page 41.

`/cddata/_backup/ServerName` is used in this procedure as an example for a backup data storage location. Determine the correct location for storing your backups and note the size requirement if you are also performing a data backup.

The following prerequisites must be met before performing this procedure:

- For clustered server environments, all commands in the following sections must be run for the master and minion servers unless otherwise noted.
- For standalone server environments, all commands in the following sections must be run as well.
- All restoration activities require that Cb Response be placed in a stopped state.

To perform a configuration restore:

1. Stop Cb Response by running these commands:
   a. (Master only) In a clustered server environment, run:

      ```
      /usr/share/cb/cbcluster stop
      ```

   b. In a standalone server environment, run:

      ```
      service cb-enterprise stop
      ```

2. Delete any stored ssh keys (files may not exist):

   ```
   rm /root/.ssh/known_hosts
   ```

3. Change directories to the backup location:

   ```
   cd /cbdata/_backup/ServerName
   ```

4. Restore configuration files as follows:
   a. Hosts file:

      ```
      tar -P -xvf cbhosts.tar
      ```

   b. Yum files:

      ```
      tar -P -xvf cbyum.tar
      ```

   c. IP table files:

      ```
      tar -P -xvf cbiptables.tar
      ```

   d. SSH keys:

      ```
      tar -P -xvf cbssh.tar
      ```

   e. Cb Response configuration:

      ```
      tar -P -xvf cbconfig.tar
      ```

      i. Clear the server.token:

      ```
      rm /etc/cb/server.token
      ```

ii. Grab the new server token:

```
python -c "from cb.alliance.token_manager import
SetupServerToken; SetupServerToken().set_server_token('/etc/
cb/server.token')"
```

f. Configure Rsyslog:

```
tar -P -xvf cbrsyslog.tar
```

g. Configure Rsyslog.d:

```
tar -P -xvf cbrsyslogd.tar
```

h. Configure Logrotate:

```
tar -P -xvf cblogrotate.tar
```

i. Configure Rabbitmq cookie:

```
tar -P -xvf cbrabbitmqcookie.tar
```

j. Configure Rabbitmq node:

```
tar -P -xvf cbrabbitmqnode.tar
```

k. (Optional) SSH authorization keys:

```
tar -P -xvf cbrootauthkeys.tar
```

l. (Master only) Syslog CEF template:

```
tar -P -xvf cbceftemp.tar
```

m. (Optional - Master only) Cb installer backups:

```
tar -P -xvf cbinstallers.tar
```

n. (Optional - Master only) Custom syslog templates - for each custom template TAR:

```
tar -P -xvf syslog_custom1.tar
```

5. (Master only) Backup Postgres database:

> **Note** If performing a full data backup, skip this step.

a. Start the Postgres database:

```
service cb-pgsql start
```

b. Drop the old database:

```
dropdb cb -p 5002
```

c. Restore roles:

> **Note** You will receive an error indicating that the account is already created.

```
psql template1 -p 5002 -f psqlroles.sql
```

d. Restore the data dump:

```
psql template1 -p 5002 -f psqldump_config.sql
```

e. Restore the database sequence values. Copy and paste the following into a file called `psqlcbvalues` on the server:

```
SELECT
pg_catalog.setval('allianceclient_comm_history_id_seq', 1,
false);
SELECT
pg_catalog.setval('allianceclient_pending_uploads_id_seq',
1, false);
```

```
SELECT pg_catalog.setval('allianceclient_uploads_id_seq', 1,
false);
SELECT pg_catalog.setval('cb_useractivity_id_seq', 1,
false);
SELECT
pg_catalog.setval('detect_dashboard_average_alert_resolution
_history_id_seq', 1, false);
SELECT
pg_catalog.setval('detect_dashboard_binary_dwell_history_id_
seq', 1, false);
SELECT
pg_catalog.setval('detect_dashboard_host_hygiene_history_id_
seq', 1, false);
SELECT pg_catalog.setval('investigations_id_seq', 1, false);
SELECT pg_catalog.setval('maintenance_job_history_id_seq',
1, false);
SELECT pg_catalog.setval('moduleinfo_events_id_seq', 1,
false);
SELECT pg_catalog.setval('sensor_activity_id_seq', 1,
false);
SELECT pg_catalog.setval('sensor_comm_failures_id_seq', 1,
false);
SELECT pg_catalog.setval('sensor_driver_diagnostics_id_seq',
1, false);
SELECT pg_catalog.setval('sensor_event_diagnostics_id_seq',
1, false);
SELECT pg_catalog.setval('sensor_licensing_counts_id_seq',
1, false);
SELECT pg_catalog.setval('sensor_queued_data_stats_id_seq',
1, false);
SELECT pg_catalog.setval('sensor_resource_statuses_id_seq',
1, false);
SELECT pg_catalog.setval('server_storage_stats_id_seq', 1,
false);
SELECT pg_catalog.setval('tagged_events_id_seq', 1, false);
```

f. Restore `psqlcbvalues`:

```
psql cb -p 5002 -f psqlcbvalues
```

g. Rebuild default investigations:

```
psql cb -p 5002 -c "INSERT INTO investigations VALUES
('1','Default Investigation',to_timestamp((select value from
cb_settings where key='ServerInstallTime'),'YYYY-MM-DD
hh24:mi:ss'),NULL,to_timestamp((select value from
cb_settings where key='ServerInstallTime'),'YYYY-MM-DD
hh24:mi:ss'),'Automatically Created at Installation Time');"
```

h. Remove all previous query-based feeds:

```
psql cb -p 5002 -c "delete from watchlist_entries where
group_id <> '-1';"
```

i. Clear the purge times:

```
psql cb -p 5002 -c "update cb_settings SET value = NULL where
key like 'EventPurgeEarliestTime%';"
```

j. Stop the Postgres database:

```
service cb-pgsql stop
```

6. Start the Cb Response services:

   Note If performing a data restoration, skip this step.

   a. (Master only) In a clustered server environment, run:
   ```
   /usr/share/cb/cbcluster start
   ```
   b. In a standalone server environment, run:
   ```
   service cb-enterprise start
   ```

Data Restore

This section contains a procedure for restoring the configuration files and data files. The restore should match what was performed in the backup procedures in "Backup" on page 41.

`/cddata/_backup/ServerName` is used in this procedure as an example for a backup data storage location. Determine the correct location for storing your backups and note the size requirement if you are also performing a data backup.

The following prerequisites must be met before performing this procedure:

- You must perform the procedure in "Configuration Restore" on page 46 before performing a data restore.
- For clustered server environments, all commands in the following sections must be run for the master and minion servers unless otherwise noted.
- For standalone server environments, all commands in the following sections must be run as well.

To perform a data restore:

1. Make sure that the Cb Response server is in a stopped state after performing the "Configuration Restore" on page 46.
2. Delete and restore the Solr data:
   ```
   rm -rf /var/cb/data/solr5/cbevents/0/data
   tar -P -xvf cbsolr.tar
   ```
3. Restore the module store by running these commands:
   ```
   rm -rf /var/cb/data/modulestore/*
   tar -P -xvf cbmodulestore.tar
   ```
4. (Master only) Restore the Postgres database by running these commands:
   a. Start the Postgres service:
   ```
   service cb-pgsql start
   ```
   b. Drop the old database:
   dropdb cb -p 5002
   c. Restore roles:

      Note You will receive an error indicating that the account is already created.

   ```
   psql template1 -p 5002 -f psqlroles.sql
   ```
   d. Restore the Postgres data:
   ```
   psql template1 -p 5002 -f psqldump_full.sql
   ```

e. Stop the Postgres service:

```
service cb-pgsql stop
```

5. Start the Cb Response services:

a. (Master only) In a clustered server environment, run:

```
/usr/share/cb/cbcluster start
```

b. In a standalone server environment, run:

```
service cb-enterprise start
```

Chapter 4

Ports and Protocols

This chapter provides port and protocol information for several different server communications.

Note Underlying IP addresses of any servers identified can change. Therefore, avoid listing any specific IP addresses; instead, configure your firewalls to use DNS names.

Communication	Port	Protocol	Comment
Management Station to Cb Response Server	TCP 22	SSH	A management station is a machine from which system administrators can SSH into the Cb Response Server and address any required administrative tasks or troubleshooting.
Management Station to Cb Response Server	TCP 443	HTTPS (configurable)	
Sensor to Cb Response Server	TCP 443	HTTPS (configurable)	NA

Communication	Port	Protocol	Comment
Master Cb Response Server to Minion Cb Response Server	TCP 22	SSH	NA
	TCP 4369	RabbitMQ	
	TCP 5701	datagrid	
	TCP 6379	REDIS	
	TCP 6500	sensorservices	
	TCP 6501	sensorservices	
	TCP 8080	SOLR	
	TCP 9000	CB data store	
	TCP 25004	RabbitMQ	
Minion Cb Response Server to Master Cb Response Server	TCP 4369	RabbitMQ	NA
	TCP 5002	POSTGRES	
	TCP 5600	liveresponse	
	TCP 5701	datagrid	
	TCP 6379	REDIS	
	TCP 6500	sensorservices	
	TCP 6501	sensorservices	
	TCP 8080	SOLR	
	TCP 25004	RabbitMQ	
Minion Cb Response Server to Minion Cb Response Server	TCP 4369	RabbitMQ	NA
	TCP 5701	datagrid	
	TCP 6500	sensorservices	
	TCP 6501	sensorservices	
	TCP 8080	SOLR	
	TCP 25004	RabbitMQ	

Communication	Port	Protocol	Comment
Cb Response Server to a CB Alliance Server	TCP 443	HTTPS	For URLs that can accept Cb Alliance Server communications, see one of the following: • api.alliance.carbonblack.com Points to the Cb Alliance Server and has single IPs behind it that will change for various reasons over time. • api2.alliance.carbonblack.com Points to the Cb Alliance Server and the single IP behind it may be different or the same as api..carbonblack.com. In addition, it may change for various reasons over time. Note that the IPs behind these servers are subject to change.
Cb Response Server to Cb Threat Intel	TCP 443	HTTPS	For a URL that accepts Cb Threat Intel communications, see threatintel.bit9.com. This URL has multiple elastic IPs behind it, and it points to the "next-gen" Cb Threat Intel infrastructure. Note that the IPs behind this URL are subject to change.

Communication	Port	Protocol	Comment
Cb Response Server to YUM Repositories	TCP 443	HTTPS	For an API on this type of communication, see yum.distro.carbonblack.io.
	TCP 80	HTTP	For an API on this type of communication, see mirror.centos.org or any other kind of enabled repository. Note that Cb Response uses the default CentOS configuration when installing base CentOS packages. Any communication supporting the installation or update of YUM repositories utilizes mirror.centos.org first, but that host is used only to identify the current mirror server list. After that, *any* of the available mirror servers might be chosen to download the actual packages. If this default CentOS behavior is a problem, ask your system administrator to change the CentOS configuration.

Chapter 5

Installing a Cb Response Cluster

This chapter introduces Cb Response clusters and explains how to configure clusters, add minions to existing clusters, remove minion nodes from clusters, and upgrade cluster nodes.

Note Some folders and scripts use the term “slave”, which is another term for “minion”.

Sections

Overview

A Cb Response cluster is a group of servers fulfilling certain roles, operating as a single Cb Response instance. Clusters are used to support a larger number of sensors and data than a single standalone server can support. This chapter provides details about setting up a Cb Response cluster.

Cb Response collects a large volume of data at varying degrees of velocity. As volume and velocity increase, the server begins to reach the limit of what a single server can support. In order to scale to support increasing volume and velocity, the infrastructure must grow horizontally. This means adding additional servers to the infrastructure. A cluster of Cb Response servers must be configured when the desired number of sensors, activity level, or amount of retention exceeds a certain threshold, which negatively impacts the performance of a single Cb Response server instance.

Horizontal scaling is used by Solr, which is the core component that Cb Response uses to store data. Solr is a big data solution that uses distributed indexes and distributes queries to those indexes. This increases query performance for high volume and velocity indexing. To support optimal scaling performance, the Cb Response infrastructure is modeled around this concept.

See the *Cb Response Server Sizing Guide (OER)* document for more information on server performance and hardware requirements.

Cluster Architecture

Cb Response cluster servers have two roles:

- *Master* – Also known as a head-node
- *Minion* – Also known as an index(ing) node

Each cluster membership role fulfills specific functions that encompass a single Cb Response instance. At their fundamental level, all cluster nodes are a functioning Cb Response server with specific functionalities and internal components configured to perform its membership role.

The following table provides a matrix for Cb Response services and certain components compared to each role:

Service/Component	Standalone	Master	Minion
cb-nginx	Yes	Yes	Yes
cb-datastore	Yes	Yes	Yes
cb-coreservices	Yes	Yes	Yes
cb-sensorservices	Yes	Yes	Yes
cb-liveresponse	Yes	Yes	No
cb-allianceclient	Yes	Yes	Yes
cb-datagrid	Yes	Yes	Yes
cb-enterprise	Yes	Yes	Yes
- ThreatIntel	Yes	Yes	No

Service/Component	Standalone	Master	Minion
- Statistics	Yes	Yes	Yes
cb-redis	Yes	Yes	Yes
cb-pgsql	Yes	Yes	No
cb-solr	Yes	Yes	Yes
- Process data (cbevents core shards)	Yes	No	Yes
- Binary Info data (cbmodules core)	Yes	Yes	Yes*
- Feed data (cbfeeds core)	Yes	Yes	No
cb-rabbitmq	Yes	Yes	Yes
Binary (modulestore)	Yes	No	Yes

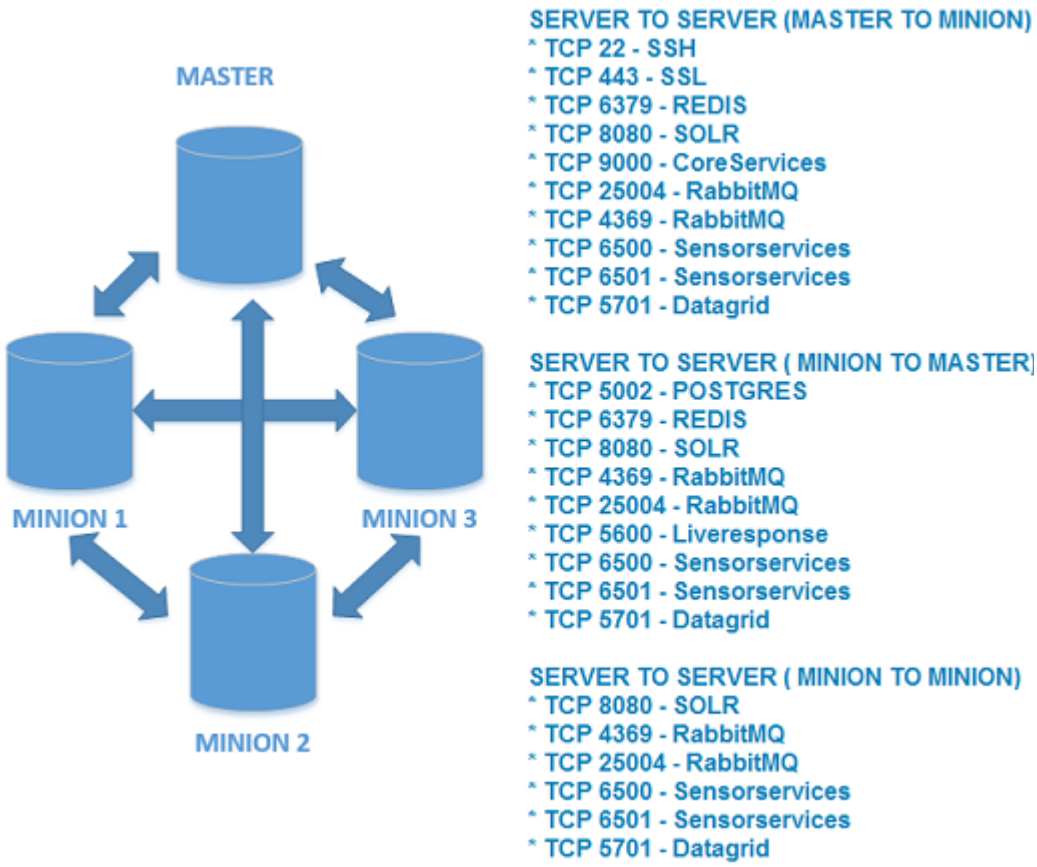
* replicated by Master

Solr clustering is performed by breaking up a single Solr core into multiple cores called shards. Shards, which are identified by a numerical integer starting at 0, are then evenly distributed to the indexing nodes (minions) for individual management and distributed querying.

For smaller cluster implementations, the master can also perform the role of an indexing server and fulfill all roles similar to a standalone server. However, for larger organizations with more than 60,000 sensors or four minions, the cluster must have a dedicated master that is not performing minion duties.

When Cb Response is clustered, internal nodal communication must occur so that the application behaves as a single instance. Most internal components can operate in this distributed capacity by making standard calls to the other components and/or nodes.

The following diagram illustrates inter-nodal cluster communications:



In order for RabbitMQ to function properly as a single system, it must also be clustered. RabbitMQ is an application message bus used to exchange data (or messages) between the processes, application, and servers. Creating a RabbitMQ cluster allows Cb Response to exchange and queue internal (server process and application) and inter-nodal (node-to-node) messaging for proper operation as a single Cb Response instance.

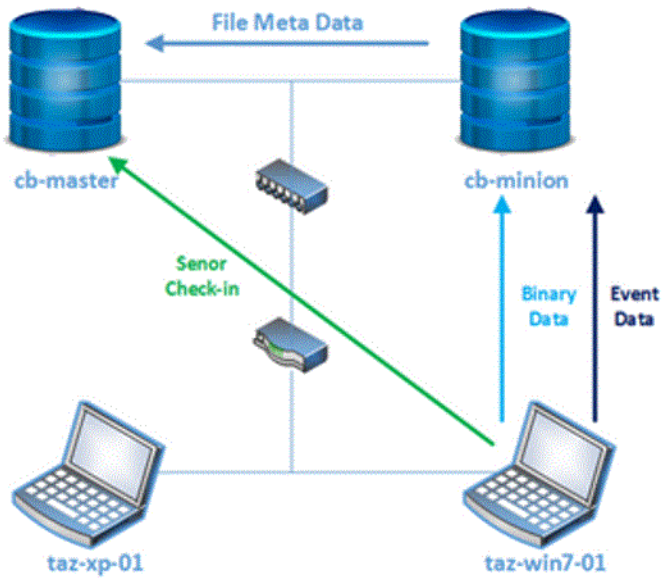
Cluster Operation

A Cb Response cluster must operate with multiple servers in the same way as a standalone instance. As a result, each role must be responsible for certain aspects of a Cb Response instance.

The master contains the user interface and is the primary frontend for the API and most integrations. As users navigate the web console, standard API endpoint calls will be made to the master, which in turn queries the appropriate backend storage location. When process data is queried, the master distributes the query to the minions and renders the aggregated results. However, if a binary, threat intel, or alert search is performed, it only queries its local Solr cores.

The master also contains the only instance of the PostgreSQL database that houses most of the application-specific configuration and certain state information that is used for sensor management. The master is in charge of managing the sensor configuration and communications, which include the Cb Live Response capabilities. Managing sensors not only requires sensor state information but also minion state metrics to independently distribute minion bandwidth to allow its assigned sensors to submit data.

The minions serve as the primary ingestion point for sensor data. When a sensor is told to send data to the minion by the master, the minion receives the data and begins to ingest the data for storage. A minion will store all process-related data in its cbevents Solr core(s), where it manages its indexes separately from the rest of the cluster. This data is retrieved when a distributed query is initiated by the master. Binary data is forwarded to the master for storage and management. This is only done once per unique binary per sensor. Unlike the binary metadata, the copies of the binaries are stored locally on the minion. The copies are only submitted and stored once per Cb Response cluster or instance. The distribution of the binary storage depends on the sensor to which that binary was submitted. This data flow is illustrated in the following figure:



Configuring Cb Response Clusters

This section describes how to configure a Cb Response cluster containing a master node and any number of minion nodes. This process involves two generic Cb Response server installs, which have no client software configured on them initially. The goal is to create a Cb Response cluster with `cb-master` as the master node and `cb-minion` as the minion node.

You must execute the initial `cbinit`, which runs on the master node.

If the node is a master and the cluster has more than three nodes, add the following parameter to instruct the master node *not* to store events:

```
--no-solr-events
```

> **Note** The following instructions assume that you have installed the Cb Response RPM and have run the `yum install cb-enterprise` command on the master node. The minion is simply a generic install of CentOS. For more information, see "Installing and Initializing a New Cb Response Server" on page 21.

To set up the cluster configuration:

1. On the master node, issue the `cbinit` command with the correct `--no-solr-events` flag defined, based on your installation. In this example, we will instruct the master node to not store events.

   ```
   /usr/share/cb/cbinit --no-solr-events
   ```

   > **Note** You will use the following Cb Response cluster management command-line tool options to initiate the cluster configuration:
   >
   > `[root@cb--?master~]# /usr/share/cb/cbcluster`
   >
   > Usage: `cbcluster COMMAND [CMD OPTIONS]`
   >
   > Available commands are:
   >
   > `help` – Display this help screen
   >
   > `start` – Start the cluster
   >
   > `stop` – Stop the cluster
   >
   > `status` – Get the running status of the cluster
   >
   > `add-node` – Add a minion node to the cluster
   >
   > `change-node` – Change parameters of the existing cluster node
   >
   > `remove-node` – Remove a minion node from the cluster

2. From `cb-master`, run the following command to initiate the cluster configuration (see "Best Practices" on page 63):

   ```
   [root@cb-master~]# /usr/share/cb/cbcluster add-node
   ```

3. Enter the following information when prompted:
   - For the hostname or IP address of the remote node, enter the IP address of the server to become a minion node. For example:`172.xx.xxx.xxx`.
   - For the password of the server that will become a minion node, enter the root password for the server in question; do not enter the Cb Response password.

   ```
   The Cb Response software is now  installed on each minion. The
   yum repo configuration used on the master is used on each node
   added.
   ```

4. When the minion nodes have all been configured, start the cluster services:

   ```
   [root@cb-master~]# /usr/share/cb/cbcluster start
   ```

5. You can now view the results in the config file:

   ```
   [root@cb-master~]# cd /etc/cb/
   ```

```
[root@cb-master~]# less cluster.conf

#################################################################
#############
#
# /etc/cb/cluster.conf:
# This file contains Cb Response server cluster configuration,
# which includes the list of participating nodes and Solr shards
present on
#     every one of those nodes.
#
# NOTE: The contents of this file are being managed by
# /usr/share/cb/cbcluster command line tool and any changes
made here may
# be overwritten next time that tool is used.
#
#################################################################
#############

[Cluster]
NodeCount=2
NextSlaveAutoInc=2

[Master]
Host=172.16.100.110
HasEvents=False
User=root

[Slave1] Host=172.16.100.111
HasEvents=True
User=root
```

6. Log into Cb Response.
7. In the top-right corner of the console, select your ***username*** **> Settings**:

8. In the Settings panel, select **Server Nodes** to view the server nodes in a cluster:

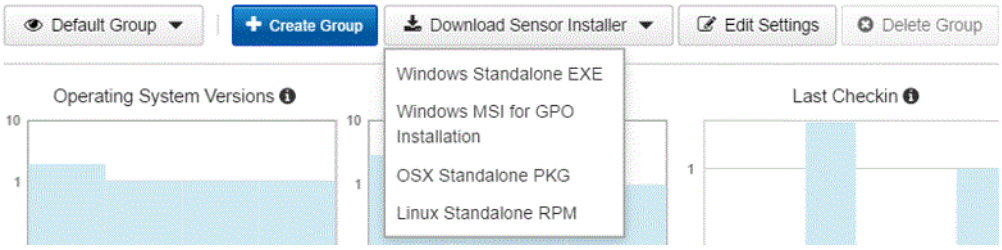
Sensor Install and Verification

The next procedure involves adding sensors to the cluster node(s) and verifying afterward that the installed sensors appear in the Cb Response console on the master node. All sensors report to the master and have an assigned node for reporting process events and binary data.

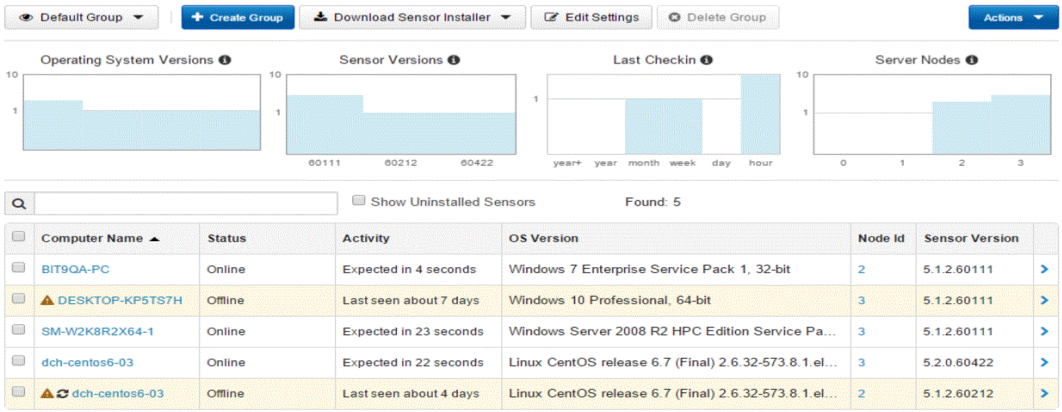
To install sensors and verify they appear in Cb Response:

1. Log into Cb Response.
2. In the left-navigation menu, select **Sensors**.
3. Select **Download Installer** and select the sensor you want to install:

> **Note** See the “Installing Sensors” chapter in the Cb Response User Guide information on installing sensors.



4. After you have installed all sensors on the cluster node(s), verify that they appear in the Cb Response console on the master node by again navigating to **Sensors** in the left navigation menu:



Best Practices

When configuring a Cb Response cluster, Carbon Black recommends that you refer to the *Cb Response Server Sizing Guide*, which describes performance and scalability considerations in deploying Cb Response.

Adding Minions to Existing Clusters

To add minions to existing clusters:

1. Verify that master and minion are on the same version of Cb Response.
2. Login to the master server and stop the cluster services:

   ```
   /usr/share/cb/cbcluster stop
   ```
3. Check the status to ensure the cluster services have stopped:

   ```
   /usr/share/cb/cbcluster status
   ```
4. Run the cluster add node command:

   ```
   /usr/share/cb/cbcluster add-node
   ```
5. On the master server, start up the cluster:

   ```
   /usr/share/cb/cbcluster start
   ```

Removing Minions from Existing Clusters

Removing unneeded minions from a cluster reduces the overall size of the cluster and simplifies the cluster deployment. This improves the cluster performance by reducing network overhead.

Best Practices

Be sure to adhere to these best practices before attempting to remove minions from an existing cluster. Consult the *Cb Response Server Sizing (OER) Guide* to ensure that the reduced cluster has the capacity to support the total number of endpoints.

- The remaining minions must have enough disk space to contain the full data.
  - Calculate the daily usage of the disk per sensor and re-calculate the required disk space to new minions for full retention. Here is an example:

    Assume that you want to collapse the six-minion cluster that has 40K endpoints down to three minions and have retention for 30 days. This is supported by Cb Response v6.0, because each minion will still have less than the maximum of 18,750 endpoints.

    You have calculated the current daily disk usage per sensor to be 20MB (dividing the total cluster data volume storage with the current daily retention and number of endpoints).

    The new total storage requires a minimum of 20MB * 40K (endpoints) * 30 (days) = 24 TB (or 8 TB per node).

    Add 20-30% to the available disk space on top of the calculated amount to allow for increased sensor activity in the future.
  - Additional extra disk space must be provided for desired cold storage in days.
- The master cluster node must have enough disk space to contain binary files for all removed minions. Binary files typically take less space than events but should still be taken into account. You can calculate the required extra space on the master by logging into Cb Response, navigating to **Server Dashboard** in the left navigation menu, and viewing the **Storage Statistics** for the minions. Add up the total binary size on the minions that will be removed to obtain the maximum additional space required on the master node.
- The remaining minions must have enough CPU and memory space to support the sensor load (per OER).

Read-Only Minions

Before removing minions from an existing cluster, you must convert them to a read-only state. Read-only minions are searchable throughout the API and user interface but are not used for sensor checkins or event/datastore pushes. While minions are in the read-only state, the system copies the binary files to the master.

The event data is not added to the minions and existing data is purged periodically as in normal operations. As a result, the read-only minions become completely inactive after the retention period. After the desired data retention period expires, you can safely remove the read-only minions.

Note Read-only minions can be reverted to active minions at any time if needed.

Removing Minions

Perform these steps to mark/unmark minions as read-only and to remove a cluster minion. If the node_id parameter is required, it can be found inside `/etc/cb/cluster.cfg` for a given node.

To mark minions as read-only:

1. Stop the cluster.
2. For each minion that you want to remove, run the following command:

   ```
   cbcluster change-node -N {node_id} -R True
   ```
3. (Optional) If you have an eventless master node and want it to contain events, run the following command:

   ```
   cbcluster change-node -E True
   ```
4. Start the cluster.

To unmark minions as read-only:

1. Stop the cluster.
2. For each minion that you want to remove, run the following command:

   ```
   cbcluster change-node -N {node_id} -R False
   ```
3. (Optional) If you want to convert master node back to eventless, run the following command:

   ```
   cbcluster change-node -E False
   ```
4. Start the cluster.

To remove a cluster minion:

1. Stop the cluster.
2. For each minion that you want to remove, run the following command:

   ```
   cbcluster remove-node -N {node_id}
   ```
3. Start the cluster.

Removed minions are no longer part of the cluster and can be shut down and de-provisioned.

Note Only read-only minions can be removed.

The system can remove minion only if all storefiles have been copied to the master node. If this is the case, an error is printed stating how many more files must be moved; you can try to remove the minion at a later time.

Upgrading Cluster Nodes

Currently, the procedure for upgrading cluster nodes involves manually upgrading each cluster node following a similar procedure to upgrading a standalone Cb Response server (discussed in “Upgrading a Cb Response Server” on page 31). Additional steps are also required to ensure that the cluster nodes are properly communicating with each other.

To upgrade cluster nodes:

1. Login to the master node and stop the cluster:

   ```
   /usr/share/cb/cbcluster stop
   ```
2. Login to each of the machines in the cluster and upgrade cb-enterprise:

   ```
   yum upgrade cb-enterprise
   ```

3. Login to each of the cluster nodes and upgrade their data schema to the latest version:

   ```
   /usr/share/cb/cbupgrade
   ```

   Warning At the end of the upgrade, the tool will ask if you want to start services. Do NOT start the services.

4. With new software version, communications port requirements may change, As a result, you must check if the iptables settings need to be updated:
   a. If you are managing iptables yourself, run this command to identify which rules need to be added:

      ```
      /usr/share/cb/cbcheck iptables -l
      ```

   b. If you want the iptables settings to be adjusted automatically, run this command to have the utility apply those settings:

      ```
      usr/share/cb/cbcheck iptables --apply
      ```

5. After all nodes have been upgraded (but not started), start the entire cluster at once by logging into the master node and issuing this command:

   ```
   /usr/share/cb/cbcluster start
   ```

This step is important to perform after an upgrade. This is due to the fact that when the cluster is started using the `cbcluster` tool, it will redistribute configuration files that must be synchronized across all nodes within the cluster. If a new software version introduces configuration changes, this will ensure that each minion node has the updates.

Chapter 6

Using CBCLUSTER as a Non-Root User

This chapter describes how to use the CBCLUSTER command as a non-root user.

Sections

Overview

`cbcluster` is a command line utility used to manage and configure Cb Response clusters on the master node. It supports the following options:

- `cbcluster start`
- `cbcluster stop`
- `cbcluster status`
- `cbcluster add-node`

These commands run in *root* context and require users to have *sudo* privileges in order to invoke `/usr/share/cb/cbcluster` on the master node. Additionally, the `cbcluster add-node` command, which is used to add a new minion node to a cluster, connects and remotely executes on the minion a series of commands for cluster configuration.

Notes

- Please ensure that the Master and Minion are on the same version of Cb Response prior to running the `cbcluster add-node` command. If the versions are not the same, you will see an error message.
- The `cbcluster reshard` command is deprecated and no longer supported.

As of Cb Response version 5.1.1, it is possible to define a non-root user as the remote user for minion communication and execution. Previously, when adding a minion node to a cluster, the `cbcluster` utility required availability of root user on the minion node. In this release, this requirement has been relaxed, and minion nodes can be configured as a non-root user.

This chapter describes the required privileges for the non-root user and its use during cluster setup.

Required User Privileges

Before invoking `cbcluster` to connect to a minion as a non-root user, the remote user on the minion needs to have certain assigned privileges:

- SSH access to the minion node
- Sudo privileges for the commands listed below; the user MUST be configured to run with NOPASSWD.

To use the `cbcluster` commands with a non-root user, add the following entries to your sudoers file:

```
## Required sudo privileges on minion to run cbcluster add-node
Cmnd_Alias HOSTNAME = /bin/hostname
Cmnd_Alias CB_INIT = /usr/share/cb/cbinit
Cmnd_Alias YUM_INSTALL_CB = /usr/bin/yum install cb-enterprise -y
Cmnd_Alias YUM_INSTALL_RSYNC = /usr/bin/yum install rsync -y
Cmnd_Alias MKDIR_ETC_CB = /bin/mkdir /etc/cb --mode=755
Cmnd_Alias MKDIR_ETC_CB_CERTS = /bin/mkdir /etc/cb/certs --mode=755
Cmnd_Alias COPY_ALLIANCE_CRT = /usr/bin/rsync --remove-source-files -
-verbose /tmp/.cb_tmp/carbonblack-alliance-client.crt /etc/cb/certs/
carbonblack-alliance-client.crt
Cmnd_Alias COPY_SERVER_CRT = /usr/bin/rsync --remove-source-files --
verbose /tmp/.cb_tmp/cb-server.crt /etc/cb/certs/cb-server.crt
Cmnd_Alias COPY_CLIENT_CA_CRT = /usr/bin/rsync --remove-source-files
--verbose /tmp/.cb_tmp/cb-client-ca.crt /etc/cb/certs/cb-client-
ca.crt
Cmnd_Alias COPY_ALLIANCE_KEY = /usr/bin/rsync --remove-source-files -
-verbose /tmp/.cb_tmp/carbonblack-alliance-client.key /etc/cb/certs/
carbonblack-alliance-client.key
Cmnd_Alias COPY_SERVER_KEY = /usr/bin/rsync --remove-source-files --
verbose /tmp/.cb_tmp/cb-server.key /etc/cb/certs/cb-server.key
Cmnd_Alias COPY_CLIENT_CA_KEY = /usr/bin/rsync --remove-source-files
--verbose /tmp/.cb_tmp/cb-client-ca.key /etc/cb/certs/cb-client-
ca.key
Cmnd_Alias COPY_CB_REPO = /usr/bin/rsync --remove-source-files --
verbose /tmp/.cb_tmp/CarbonBlack.repo /etc/yum.repos.d/
CarbonBlack.repo
Cmnd_Alias COPY_CLUSTER_CONF = /usr/bin/rsync --remove-source-files -
-verbose /tmp/.cb_tmp/cluster.conf /etc/cb/cluster.conf
Cmnd_Alias COPY_ERLANG_COOKIE = /usr/bin/rsync --remove-source-files
--verbose /tmp/.cb_tmp/.erlang.cookie /var/cb/.erlang.cookie
Cmnd_Alias COPY_SERVER_LIC = /usr/bin/rsync --remove-source-files --
verbose /tmp/.cb_tmp/server.lic /etc/cb/server.lic
Cmnd_Alias COPY_SERVER_TOKEN = /usr/bin/rsync --remove-source-files -
-verbose /tmp/.cb_tmp/server.token /etc/cb/server.token
Cmnd_Alias CBCHECK_IP_TABLES = /usr/share/cb/cbcheck iptables --apply
Cmnd_Alias CB_ENTERPRISE = /etc/init.d/cb-enterprise
Cmnd_Alias CAT_VERSION = /bin/cat /usr/share/cb/VERSION
Cmnd_Alias CBUPGRADE = /usr/share/cb/cbupgrade --non-interactive
Cmnd_Alias CBUPGRADE_CHECK = /usr/share/cb/cbupgrade --check

my_user       ALL=(ALL)  NOPASSWD: HOSTNAME, CB_INIT, YUM_INSTALL_CB,
YUM_INSTALL_RSYNC, MKDIR_ETC_CB, MKDIR_ETC_CB_CERTS,
COPY_ALLIANCE_CRT, COPY_SERVER_CRT, COPY_CLIENT_CA_CRT,
COPY_ALLIANCE_KEY, COPY_SERVER_KEY, COPY_CLIENT_CA_KEY, COPY_CB_REPO,
COPY_CLUSTER_CONF, COPY_ERLANG_COOKIE, COPY_SERVER_LIC,
COPY_SERVER_TOKEN, CBCHECK_IP_TABLES, CB_ENTERPRISE, CAT_VERSION,
CBUPGRADE, CBUPGRADE_CHECK
```

See https://community.carbonblack.com/docs/DOC-5692 for a version of these entries formatted so that you can copy and paste them into your environment.

If any of the required permissions are not configured, the `cbcluster` command will prompt for the missing permissions during initial validation.

Defining Users

> **Note** To avoid potential conflicts, do not select “cb” for a username.

There are two ways to invoke `cbcluster` and configure minion nodes as a user other than root:

- If this is a new minion which has not been added to the cluster, the new user can be configured when running add-node, by including the new add-node option:

  `--user=<arg>`

  For example:

  `$ /usr/share/cb/cbcluster add-node --hostname <my_host> --user <my_user>`

  This will configure `cbcluster` to perform all remote actions for `<my_host>` as `<my_user>`. The /`etc/cb/cluster.conf` file will be updated to reflect the new configuration with the key-value pair "`User=<my_user>`".
- If this is an existing minion, the `/etc/cb/cluster.conf` file can be modified directly. Open the file in an editor, and for each minion node, add the key-value pair "`User=<my_user>`" where `<my_user>` is the actual username you want `cbcluster` to use when connecting to that minion.