Server Overview

CB v5.0.0.150122.1654

January 22, 2015

Contents

Carbon Black Server

Overview

This is the technology stack on a Carbon Black server:

There are five major daemons in Carbon Black server:

- `cb-nginx` - used as HTTP reverse proxy to internal daemons
- `cb-coreservices` - Python, Gunicorn - all non-data application logic for HTTP transactions
- `cb-datastore` - Java/Tomcat - all incoming data (eventlogs and binary files)
- `cb-solr` - Java/Tomcat - Apache Solr, the primary data store
- `cb-postgres` - traditional relational database

`nginx` is the only daemon with public sockets. The remaining daemons are bound to `127.0.0.1` and can only be accessed locally or via the nginx reverse proxy. nginx owns `tcp/80` and `tcp/443` and redirects to `coreservices`, `cb-datastore` or the Cb web root:q based on the URL prefix:

- `/` -> `/var/www/cb/`
- `/api/*` -> `coreservices` on `tcp/5000`
- `/sensor/*` -> `coreservices` on `tcp/5000`
- `/data/*` -> `cb-datastore` on `tcp/9000`

Note: There are two URL prefixes for `coreservices`: `/api/*` and `/sensor/*`. All `/api/` URLs are used by the Web UI and REST clients. All `/sensor/` URLs are used by the sensors pushing data. These are isolated to allow binding a separate nginx server instance to `tcp/443` on a public / DMZ interface for sensors outside the internal network, (road warriors, work laptops at home, etc) without exposing the `/api/` interfaces to the world. This can be done with a simple `nginx` configuration change, an example is in `/etc/cb/nginx/conf.d/cb-multihome.conf.example`.

Note: Listening ports are somewhat different in a clustered setup. Please see cluster-specific documentation for more details.

In general, sensors first register and checkin to `coreservices` via `nginx`. If the sensor has data, following checkin it will `POST` event logs to `cb-datastore` via `nginx`. `cb-datastore` will cache data for a few minutes, before sending a collection of related data to `cb-solr`.

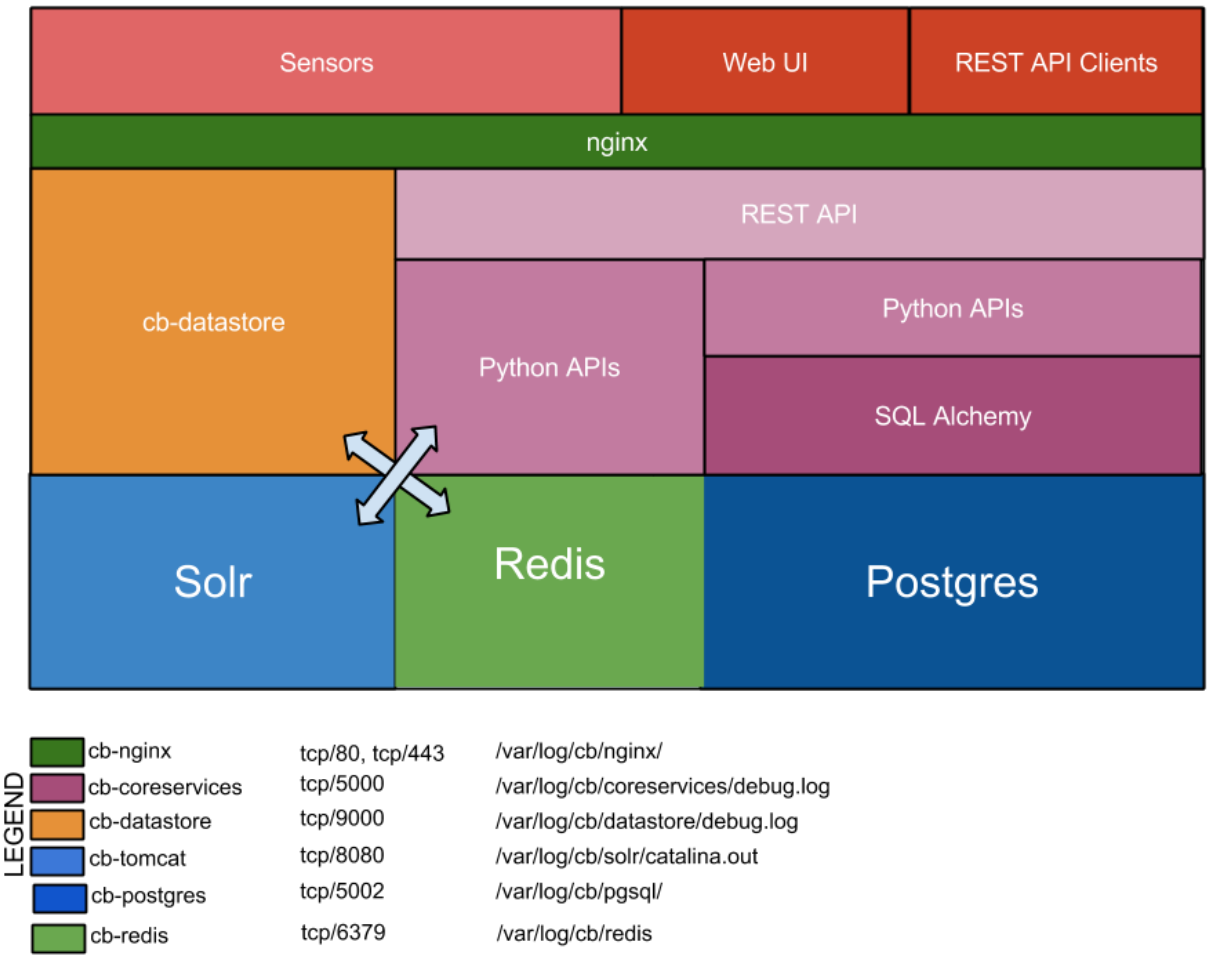

Figure 1: Cb Server Technology Stack

Server Configuration

Generally static, daemon configuration data is stored in flat files, following typical Linux convention. Dynamic run-time configuration data is stored in Postgres and configured via the Web UI.

Most major configuration is done during `cbinit` at initial server install. `cbinit` configures a combination of initial settings in both static configuration files and Postgres.

There are the following major static config files:

- `/etc/cb/cb.conf` - Major Carbon Black-wide settings
- `/usr/share/cb/setup/solr/cbevents/conf/solrconfig.xml` - Apache Solr settings
- `/var/cb/pgsql/postgresql.conf` - Postgres settings
- `/etc/cb/coreservices-logger.conf` - Configure log settings for the python `coreservices` daemon
- `/etc/cb/cb-datastore/*` - Configure log settings for the Java `cb-datastore` daemon
- `/etc/cb/nginx/conf.d/cb.conf` - nginx server settings, defines IPs and ports to bind listeners to
- `/etc/sysconfig/iptables` - Standard iptables configuration for the server firewall.

There are the following secondary static config files:

- `/etc/cb/allianceclient-logger.conf` - Configure log settings for the Alliance Client daemon
- `/etc/cb/nginx/cb-nginx.conf` - standard nginx server config, but tailored for Carbon Black
- `/etc/cb/solr/tomcat6.conf` - Tomcat webserver settings

Server Logs

Log overview

Enterprise Server makes heavy use of log files. These files provide both reassurance of system health and a record of activity under error conditions.

The are the following critical logs on a Carbon Black server:

- `/var/log/cb/nginx/access.log` (and `error.log`): nginx HTTP access logs for all sensor and API traffic.
- `/var/log/cb/coreservices/debug.log`: application logic for sensor and API traffic
- `/var/log/cb/cb-datastore/debug.log`: incoming sensor data cache
- `/var/log/cb/solr/catalina.out`: sensor data storage, indexing and queries

Where should I look if...

- **...I get an error in the Web UI?**

`/var/log/cb/coreservices/debug.log` - there will be a Python stack trace with details; your support rep can help.

- **...sensors are not checking in?**

Start with `/var/log/cb/nginx/access.log` - look for a request from the host in question: `10.1.64.109 - - [10/Oct/2013:18:28:48 -0400(0.224)] "POST /sensor/checkin HTTP/1.1" 200 109 "-" "-" "-"`

Note that v4.1+ sensors append the sensor ID to the URI. For example:

```
10.1.64.109 - - [10/Oct/2013:18:28:48 -0400(0.224)] "POST /sensor/checkin/529
HTTP/1.1" 200 109 "-" "-" "-"
```

The HTTP response code is the three-digit code immediately following `HTTP/1.1` (`200` above). If the code is `200`, everything is working. If you see a `5xx` error, check the `coreservices/debug.log` for `/sensor` requests or `cb-datastore/debug.log` for `/data/` requests.

If there are no entries in `access.log` check `error.log`. If the sensor SSL client certificates are not signed by the 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` confirm communications from the sensor perspective with the steps in sensor troubleshooting.

- **. . . I just want to make sure everything is working?**

Check nginx `access.log` for '200' HTTP response codes. 200s are good.

You can also check `solr/catalina.out` for requests to the `/update` handler:

`INFO: [cbevents] webapp=/solr path=/update params={wt=javabin&version=2} {add=[a2247de5-fa (1448549632617480192), b6879e21-eff8-dbad-0000-000000000001 (1448549632695074816), 3abdd29a-018b-3037-0000-000000000001 (1448549632700317696), f405c732-b898-bedd-0000-000000 (1448549632705560576), ... (203 adds)]} 0 20248` The "203 adds" indicates how many processes were just updated with 1 or more events. These should flow past at a healthy clip on a production system.