SCALING CHEF AUTOMATE

BEYOND 100,000 NODES

Joshua Hudson, Customer Engineer | **Thomas Cate**, Customer Engineer

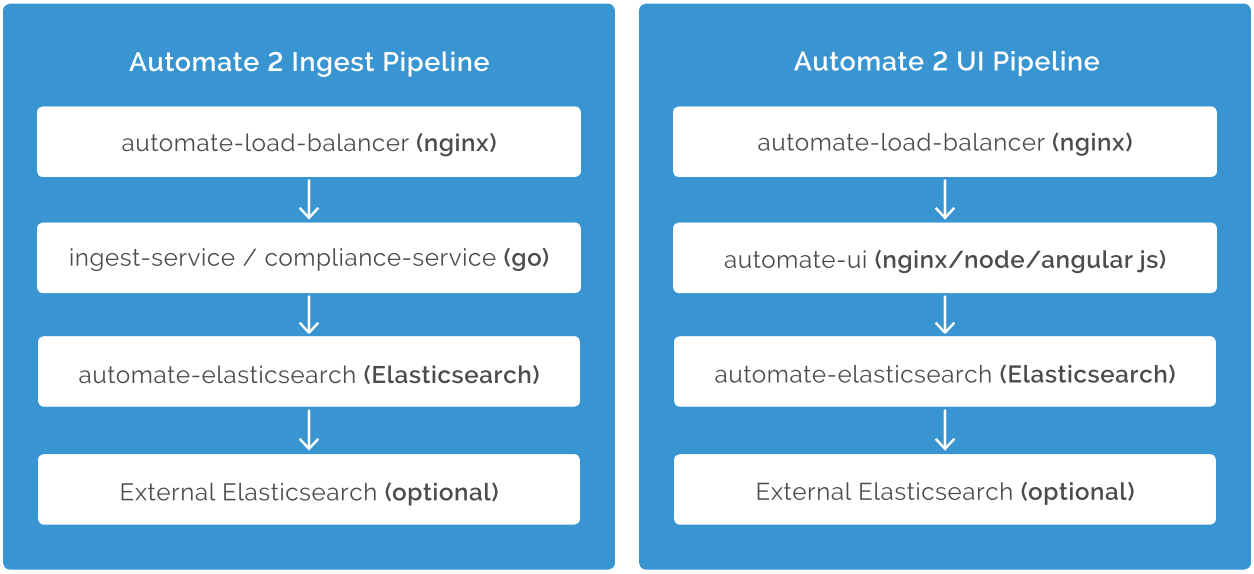
Published September, 2018

Chef Automate is an enterprise platform that allows developers, operations and security engineers to collaborate effortlessly on delivering application & infrastructure changes at the speed of business. Chef Automate provides actionable insights with enterprise scale and performance across multiple data centers and cloud providers.

Automation is essential at scale, and as the managed environment grows, visibility across all nodes is critical to maintaining compliance and enabling fast, problem-free software delivery. Chef software is in production in some of the world's largest, most-demanding environments. This paper describes proven approaches for scaling Chef Automate while detailing the architecture & hardware planning and performance tuning needed to run the system smoothly.

ARCHITECTURE COMPONENTS

A good way to think about the Chef Automate server is as a collection of microservice components underpinned by open source software including Elasticsearch, Systemd, Chef and the Linux kernel.

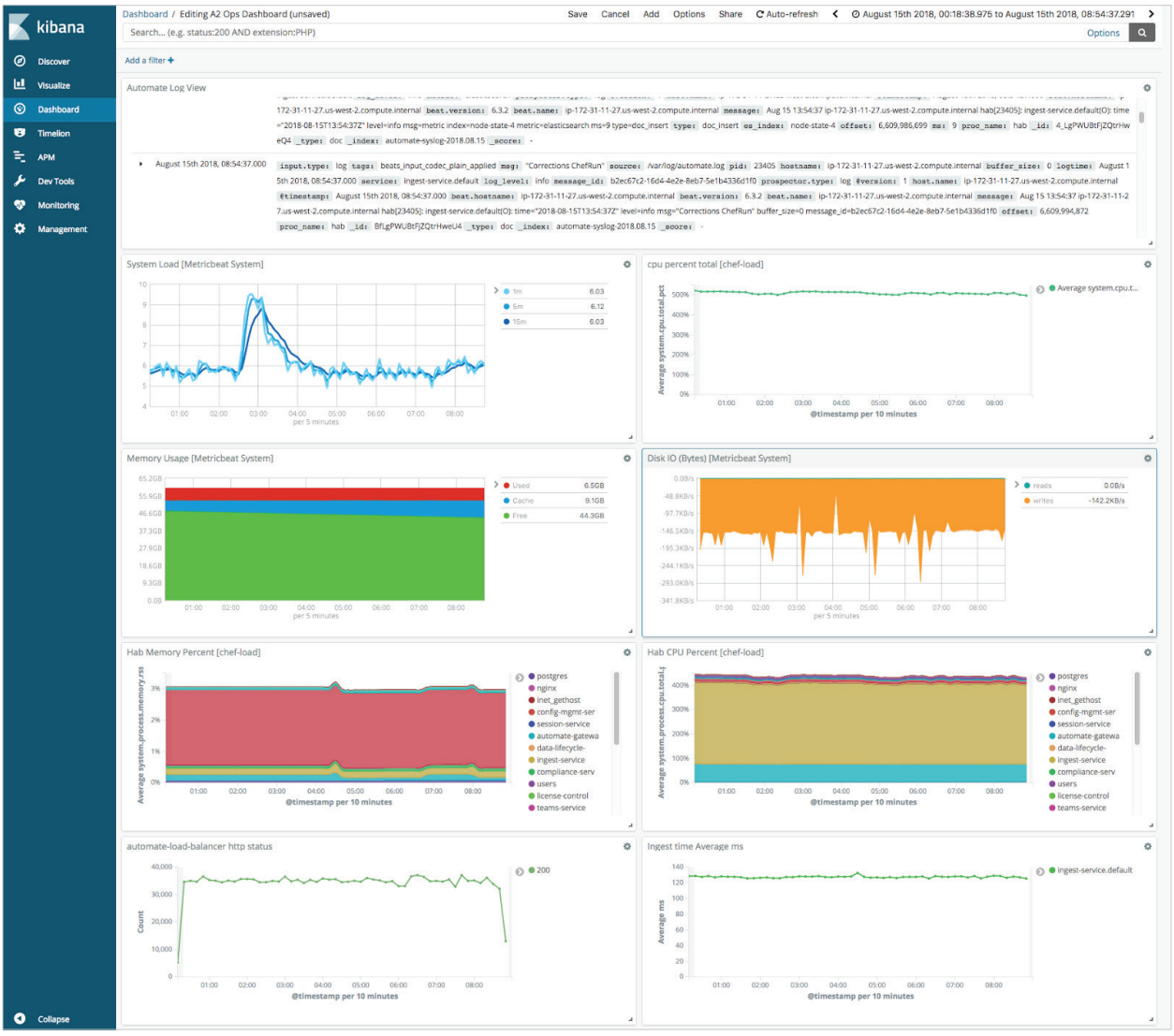


It's important to understand the performance characteristics, monitoring, and troubleshooting of Linux systems, and of Chef Automate's open source components, including NGINX, Elasticsearch, and systemd-journald. The proprietary code in Chef Automate consists of microservices written in Go.

In Chef Automate, data flows from a service called "data collector" in Nginx to Elasticsearch as shown in the diagram below. These services are the "critical path" for your Chef data and must be monitored and tuned in order to alleviate data processing bottlenecks.

MONITORING

Collecting excellent metrics are fundamental to making architecture or tuning changes — those are not only real-time metrics, but historical trends and the ability to correlate various



metrics. It's important to remember that the architecture and sizing recommendations below are a starting point, however real-life results will vary based on a myriad of factors. Data-driven decision making based on reliable metrics and trends will enable you to increase performance while decreasing costs.

At Chef, we're big fans of Open Source monitoring solutions such as Sensu, Graphite, the Elastic stack and the Influx Data Platform; these are the tools we use to monitor our own software when we run it in production.

We encourage you to make your own metrics collection decisions, but these are the most important metrics we've found to measure when evaluating the effectiveness of your Chef Automate system:

- **API Request and Error rates** - - Automate uses Nginx as a load balancer. All user interactions and data ingest comes in through here. As load begins to climb, it is most important to measure and graph the response times (in milliseconds) and error rates (non 200 messages) over time. You can find these messages with something like (journalctl -u chef-automate | grep automate-load-balancer). In this message we can see that we got a success (200) and the whole response including nginx took 0.021 seconds. The second 200 and time are what nginx saw from the Go upstream services.

```
Aug 15 15:44:38 ip-172-31-11-27.us-west-2.compute.internal hab[23405]: automate-load-balancer.default(O): - [15/Aug/2018:15:44:38 +0000]  "POST /data-collector/v0 HTTP/1.1" 200 "0.021" 0 "-" "Go-http-client/1.1" "172.31.11.27:2000" "200" "0.020" 622
```

- **Elasticsearch Post time** - The amount of time it takes for Elasticsearch to ingest the data is measured by the POST time to the _bulk API endpoint - as shown in this ingest-service log (journalctl -u chef-automate | grep ingest-service). POST times above a few seconds may indicate that Elasticsearch is suffering from insufficient disk speed, Java tunings, or CPU/memory constraints. In this example the POST time is 12 milliseconds. Looking at these messages isolates what you're looking at to only Go=>Elasticsearch, simplifying your troubleshooting.

```
Aug 15 13:54:37 ip-172-31-11-27.us-west-2.compute.internal hab[23405]: ingest-service.default(O): time="2018-08-15T13:54:37Z" level=info msg=metric index=converge-history-2018.08.15 metric=elasticsearch ms=12 type=doc_insert
```

- **Java VM (JVM) Metrics** - The data storage and search engine is Java based and tuned by default for smaller systems. The most important metric to watch is the amount of time the JVM spends paused due to "stop the world" garbage collection (GC) events. If this value becomes a significant amount of time (1% or higher) it indicates that the JVM Heap size is too small (frequent GC events) or too large (infrequent but very slow GC events).

- **System disk latency** - The creators of Elasticsearch recommend using Flash or Flash-accelerated local disk storage for optimum performance. The use of magnetic and network based storage incur additional latency as measurable by disk latency (average time it takes to service a request) and disk queue length (the number of IOs waiting to be serviced). Significant spikes in latency or disk queue length can impact many of the above metrics.

- **System disk utilization** - A full disk can cause significant problems for data storage systems used by Chef Automate: Elasticsearch, and PostgreSQL. If disks are not monitored, situations can arise including significant downtime, data corruption and data loss.

- **System CPU utilization** - Number of cores, processor "steal time" indicating contention on VM systems.

ARCHITECTURE/COST CONSIDERATIONS

Chef's recommended architectures below are based on a number of observations we've seen during customer deployments and internal testing. You may opt to decrease your hardware requirements by adjusting some of these site-wide parameters, or increase your data granularity which will increase the hardware requirements.

CHEF CONVERGE INTERVAL & OVERALL CHEF CLIENT RUNS PER MINUTE

Although the total number of nodes is an important measure, the data processing speeds of Chef Automate will be primarily determined by the rate of Chef Client Runs per Minute (CCR/m). This also includes InSpec if configured to do so (see section on Compliance data below). The CCR/m rate can be greatly impacted by adjusting the Chef client's converge interval (how frequently does Chef run) and the Splay (level of randomization so that all chef clients don't run at the exact same second).

Our calculations below are based on the default Chef Client settings, which are a 30 minute converge interval and a 5 minute splay - in order to maximally smooth the server load. Cutting either your converge interval or splay in half will double the server load and processing requirements. From a calculation standpoint, it is simplest to say that 5,000 nodes at a 30 minute converge interval are equivalent to 10,000 nodes at a 60 minute interval.

DATA RETENTION PERIOD

Chef Automate allows users to control the retention period of its data. Our calculations are based on a 90-day retention period for both Chef run and Compliance data, as this is a common value in regulatory documents. Customers may choose to increase the retention period (thus increasing storage requirements) or decrease it. A simple calculation is that a doubling of the data retention period will double the amount of data stored.

For more information on controlling Chef Automate's data retention, see: https://automate.chef.io/docs/configuration/#data-retention

COMPLIANCE DATA

The audit cookbook is used to collect Compliance (InSpec) data during the Chef client run. When the audit cookbook is used, significantly more data is sent to Chef Automate's data-collector service. Since typically this data is not as urgent, we tested with the audit cookbook configured to run daily. Configuring this to run more often will put more load on the compliance-service and automate-elasticsearch service.

SIZE OF NODE OHAI DATA

Every chef client runs Ohai, a tool for collecting system data. In Enterprise environments the Ohai data can become quite large. Two common causes are the ohai and sessions plugins, which are enabled by default in pre-14 versions of chef-client. These plugins can gather excess system information about non-local users (passwd) and system sessions on linux machines that bloat node objects with non-valuable data. If you are using a chef-client version earlier than Chef 14, we recommend disabling these ohai plugins in your client settings.

For more information on disabling ohai plugins or restricting their data from being sent to Chef Automate, see: https://docs.chef.io/ohai.html#ohai-settings-in-client-rb

SYSTEM ARCHITECTURE RECOMMENDATIONS

ASSUMPTIONS

- **30 minute Chef Client converge interval with a 5 minute splay:**
    - It's important to note that if you choose to do a more frequent converge interval (say 15 minutes instead of 30) then you must double the node count in the calculation.
- **90-day data retention period:**
    - The data storage requirement scales up and down proportionately if you wish to increase or decrease the retention period.
- **Daily compliance scans:**
    - We assume all nodes are running compliance scans with the audit cookbook. If you enable `['audit']['interval']['enabled'] = true` it will run daily.

HARDWARE PLANNING

Elasticsearch is a big data platform that automatically distributes the data storage and processing workload across all cluster members using sharding. It also automatically provides redundancy for all data stored (with replica shards), allowing customers to optimize their storage solutions for speed rather than redundancy. When using a multi-node Elasticsearch cluster, you can lose any single member without losing any data.

Storage

Elastic strongly recommends the use of Flash or SSD storage that is local to the machine running it (not NAS or SAN). In cloud environments, we've found the best performance from machine types with local SSD storage (AWS I3 or D2, Azure Ls series) , however, the SSD-based network storage options (AWS provisioned-IOPS EBS and Azure Premium Storage) provided acceptable latency in our tests but allowed much larger volume sizes.

In on-prem VM environments we recommend using direct-attached SSD arrays or SAN systems that can provide guaranteed bandwidth reserved for Elasticsearch. For larger datasets, physical machines optimized for big data workloads may be more economical. The storage must be able to handle at least **1000 sustained IOPs per Elasticsearch node with an average latency of 2ms or less**.

For storage space, the largest amount of data we observed stored was 12 MB per client node per day (split among the Elasticsearch servers). This grows linearly, to illustrate with an example:

- To retain 90 days of data for one node, you need 1.08 GB of storage space.
- If you store this on a multi node cluster it will require 2.16 GB, since two copies of each document will be stored in the cluster.

CPU

On the Chef Automate server, the primary consumers of CPU resources are the ingest and compliance services. These services are much more efficient than the previous Logstash implementation Automate 1 uses. However, if you have a very large estate of nodes sending data to Automate, these processes will require the most CPU usage.

Memory

The largest consumer of memory on all of your servers will be Elasticsearch. Giving more memory to Elasticsearch will generally increase UI performance, as Elasticsearch will be serving data from memory instead of disk. Postgres is installed on the Automate box, but it is only used for smaller amounts of data related to compliance profiles and authentication.

Network

Elasticsearch documentation states: *Avoid clusters that span multiple data centers, even if the data centers are collocated in close proximity. Always avoid clusters that span large geographic distances.*

Gigabit Ethernet is okay for most installations, until the data ingest rate begins to reach 400 Mbps (50 MB/s). At that point all systems including the Automate server should be upgraded to 10GbE NICs

.

SERVER SIZING:

This is the upper limit in number of nodes these instances can support based on a 30-minute converge interval, 24 hour compliance interval and 90 day data retention period.

NODES	**CCR/ MIN**	**Frontend Server Specs** (Tested ec2 instance)	**Frontend Storage** (TB of SSD)	**Elasticsearch** Server Count	**Elasticsearch Server Specs** (Test ec2 instance)	**Elasticsearch Per Node Storage** (TB of SSD each)	**Elasticsearch Total Storage** (Total TB)
5,000	8.3	4 CPU 16 GB (m4.xlarge)	6	0	N/A	N/A	N/A
20,000	322	16 CPU 64 GB (m4.4xlarge)	22	0	N/A	N/A	N/A
60,000	996	16 CPU 64 GB (m4.4xlarge)	0.5	3	8 CPU 64 GB (r4.2xlarge)	44	132
100,000	1610	32 CPU 64 GB (m4.10xlarge)	0.5	5	8 CPU 64 GB (r4.2xlarge)	44	220

Note 1: External Elasticsearch doubles storage requirements, since the cluster keeps 2 copies of each document.

Note 2: Recommended specs also vary slightly from tested specs, for example, the m4.10xlarge has a lot more than 64GB of RAM, but wasn't using most of it.

PERFORMANCE TUNING

The following performance tuning settings are required in order to achieve the desired throughput rates on the recommended hardware.

ELASTICSEARCH

In Chef Automate the biggest impact in user perception of performance is how responsive Elasticsearch is. All of your historical node and compliance data is stored here, and UI is built on top of queries to this data store.

Multiple Elasticsearch Nodes

The chart above gave some examples of where it might make sense to scale Elasticsearch horizontally. Another concern this covers is data resiliency. In a multi-node Elasticsearch deployment you have 2 copies of all data in the cluster, meaning a single Elasticsearch node going down does not take down Chef Automate. Keep in mind though that a cluster is not a replacement for maintaining backups.

JVM settings

Elastic has very good documentation for recommended heap settings:
https://www.elastic.co/guide/en/elasticsearch/reference/6.2/heap-size.html

Applying these settings to Chef Automate is fairly easy. Generally we recommend that you set Elasticsearch heap size to 50% of the total system RAM, with a maximum of 28GB. This is why the largest dedicated Elasticsearch servers we test with have 64GB of RAM. That gives the OS plenty for disk caches and let's us set the maximum heap Elastic recommends.

Unlike previous versions, when you connect Chef Automate 2 to an external Elasticsearch cluster it will spin up its own copy of Elasticsearch to route queries and communicate with the cluster. We do recommend increasing the heap size of this Chef Automate managed Elasticsearch instance as well, even if you're using an external cluster.

For example, in the 60,000 node example above you would configure the Chef Automate frontend server like so.

```
[elasticsearch.v1.sys.runtime]
 heapsize = "16g"
```

Then, on the external Elasticsearch servers, update the /etc/elasticsearch/jvm.options file to set these two options

```
-Xms28g
-Xmx28g
```

Minimum Master Nodes

When running an external Elasticsearch cluster we recommend that you set minimum_master_nodes to (ClusterSize/2)+1 to ensure that you never end up with a split brain scenario where different nodes end up with a different view of the world. For our recommended cluster sizes this is 2 for 3 node clusters and 4 for 6 node clusters.

Configure minimum_master_nodes in your /etc/elasticsearch/elasticsearch.yml before starting your cluster.

```
discovery.zen.minimum_master_nodes: 2
```

If you need to change this on a live cluster, for example if you expand from 3 to 6 elasticsearch cluster nodes. You can set it with curl on any node in your cluster. Once set on a single node the setting will apply to all.

```
cat > /tmp/elastic-settings.json
{
    "persistent" : {
        "discovery.zen.minimum_master_nodes" : 4
    }
}
```

```
curl -XPUT http://`hostname`:9200/_cluster/settings -d @/tmp/elastic-settings.json
```

File Handles

Elasticsearch's performance may be limited by the maximum number of file descriptors it can open. Chef Automate will configure this for its built in Elasticsearch. However, for any external Elasticsearch servers you set up, this is typically set by the limits.conf configuration file in Linux and tested using the `ulimit -n` command. To adjust this setting, see the documentation for your operating system.

For more information, see:
https://www.elastic.co/guide/en/elasticsearch/guide/current/_file_descriptors_and_mmap.html

Indexing Throttle

Elasticsearch will throttle indexing while segments are merging. By default, this is set very conservatively. We set this to 100MB which is Elastic's recommend value for SSD storage:

```
cat > /tmp/elastic-settings.json
{
    "persistent" : {
        "indices.store.throttle.max_bytes_per_sec" : "100mb"
    }
}
```

```
curl -XPUT http://`hostname`:9200/_cluster/settings -d @/tmp/elastic-settings.json
```

For more information see:
https://www.elastic.co/guide/en/elasticsearch/guide/current/indexing-performance.html#segments-and-merging

LINUX SERVERS

Modern Linux/kernel

The Chef Automate server itself will require systemd, which generally means at least rhel/cent/oel 7.x or Ubuntu 16.04 or greater. We recommend using the same operating system for your external Elasticsearch machines.

SELinux (RHEL)

On RHEL-based systems we have observed up to a 20% performance penalty in IO and process intensive services when SELinux is enabled. We recommend disabling it, or else increasing hardware to compensate.

```
# to immediately disable selinux
setenforce 0
# To make the change persist through reboots
sed -i 's/SELINUX=enforcing/SELINUX=permissive/g' /etc/selinux/config
```

RHEL boot-time kernel tuning:

In order to take advantage of modern Flash devices, the following changes must be made on RHEL-based systems (they are the default on Ubuntu 16.04).

- Enable Multi-queue I/O scheduling for SCSI
- Set the "noop" disk I/O scheduler
- Disable Transparent Huge Pages

Adjust the following line in your GRUB configuration file, `/etc/default/grub`:
(changes bolded for emphasis)

```
GRUB_CMDLINE_LINUX="console=ttyS0,115200n8 console=tty0 net.ifnames=0
biosdevname=0 crashkernel=auto scsi_mod.use_blk_mq=Y elevator=noop transparent_
hugepage=never"
```

Linux kernel VM tuning

The following settings are recommended by the Chef Automate development team to increase system responsiveness under load (reboot required):

```
cat > /etc/sysctl.d/chef-highperf.conf <<EOF
vm.swappiness=10
vm.max_map_count=262144
vm.dirty_ratio=20
vm.dirty_background_ratio=30
vm.dirty_expire_centisecs=30000
EOF
```

Filesystem

Chef Automate 2 stores all of its data under /hab, for this data store we recommend using the the XFS filesystem combined with the LVM (Linux Volume Manager). XFS provides many performance advantages over the ext4 filesystem. To format a new volume appropriately:

```
# Create LVM LV /dev/chef-vg/chef-lv
pvcreate /dev/xvdb
vgcreate chef-vg /dev/xvdb
lvcreate -n chef-lv -l 80%VG chef-vg
# create xfs
mkfs.xfs /dev/chef-vg/chef-lv
# mount
mkdir -p /var/opt
mount /dev/chef-vg/chef-lv /var/opt
```

It is worth noting that AWS EBS volumes are limited to 16TB in size, and other storage solutions may also present per-volume limits. In those cases we recommend building a RAID-0 stripe set of multiple volumes before formatting with LVM+XFS, like so:
http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/raid-config.html

WANT TO LEARN MORE?

Chef offered a training course at ChefConf 2018 called "Operating Chef Automate v 2.0". You can find this all with other Chef Training offerings at https://training.chef.io.

General Elasticsearch configuration for Automate 2.
https://automate.chef.io/docs/configuration/#general-elasticsearch-configuration

Explanation of how Chef Automate 2's internal Elasticsearch functions, with external clusters.
https://www.elastic.co/guide/en/elasticsearch/reference/6.2/modules-node.html#coordinating-node

How to upgrade your Elasticsearch cluster.
https://www.elastic.co/guide/en/elasticsearch/reference/6.2/setup-upgrade.html

Air-gapped install for Automate 2.
https://automate.chef.io/docs/airgapped_installation/

Data-collection for Automate 2.
https://automate.chef.io/docs/data-collection/