**Installing the ELK Stack**

**(ElasticSearch, Logstash, Kibana)**

Contents

[Introduction 4](#_Toc480799397)

[Prerequisites 4](#_Toc480799398)

[Server Build out 4](#_Toc480799399)

[Install Oracle Java 1.8 (or latest) 5](#_Toc480799400)

[Install ElasticSearch 5](#_Toc480799401)

[Get the Elastic.co GPG Key 5](#_Toc480799402)

[Create a new yum repository file for ElasticSearch 5](#_Toc480799403)

[Install ElasticSearch 5](#_Toc480799404)

[Edit ElasticSearch Configuration 6](#_Toc480799405)

[Start ElasticSearch 8](#_Toc480799406)

[NOTES For Running 5.X on older Kernels (CentOS 6.X) 8](#_Toc480799407)

[Bootstrap Checks 8](#_Toc480799408)

[Proc ULIMITS 8](#_Toc480799409)

[Notes on Memory and Java Heap Size 8](#_Toc480799410)

[NOTES ON GARBAGE COLLECTION 9](#_Toc480799411)

[Install X-Pack 10](#_Toc480799412)

[Basic Cluster Checks 11](#_Toc480799413)

[These are the main config files for ES 12](#_Toc480799414)

[Install Kibana 12](#_Toc480799415)

[Create Repository for Kibana 12](#_Toc480799416)

[Install Kibana 13](#_Toc480799417)

[Modify Kibana Configuration 13](#_Toc480799418)

[Start and Enable Kibana Service 13](#_Toc480799419)

[Set Up Reverse Proxy for Web Access and Authentication 13](#_Toc480799420)

[Nginx as a Proxy 14](#_Toc480799421)

[Install Nginx 14](#_Toc480799422)

[Apache as a Proxy 17](#_Toc480799423)

[Install Apache 17](#_Toc480799424)

[Configure Apache 17](#_Toc480799425)

[Start Apache 18](#_Toc480799426)

[Install Logstash 18](#_Toc480799427)

[Add Logstash Repository 18](#_Toc480799428)

[Install Logstash Package 18](#_Toc480799429)

[Generate SSL Certificates 18](#_Toc480799430)

[Option 1: IP Address 19](#_Toc480799431)

[Option 2: FQDN (DNS) 19](#_Toc480799432)

[Configure Logstash 20](#_Toc480799433)

[Test your Logstash configuration 23](#_Toc480799434)

[Load Kibana Dashboards 23](#_Toc480799435)

[Load Filebeat Index Template in ElasticSearch 24](#_Toc480799436)

[Set Up Filebeat (Add Client Servers) 25](#_Toc480799437)

[Copy SSL Certificate 25](#_Toc480799438)

[Install Filebeat Package 26](#_Toc480799439)

[Create Filebeat Repository File: 26](#_Toc480799440)

[Install Filebeat with this command 26](#_Toc480799441)

[Configure Filebeat 26](#_Toc480799442)

[Test Filebeat Installation 29](#_Toc480799443)

[Connect to Kibana 29](#_Toc480799444)

# **Introduction**

This is a quick guide to get the ELK stack up and running on CentOS.

I pulled this stuff from websites like: <https://www.digitalocean.com/community/tutorials/how-to-install-elasticsearch-logstash-and-kibana-elk-stack-on-centos-7>

The basic gist of the install is that you use packages provided by Elastic.co and configure them for the environment.

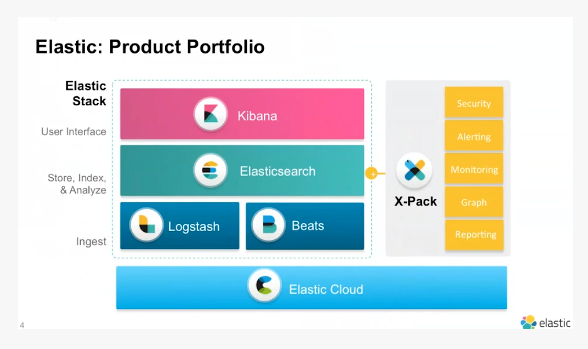


Figure 1 - ELK Stack Architecture

# Prerequisites

## Server Build out

The first step is to build the servers for your cluster. Make sure they:

* Are identical OS version
* Have network connectivity and DNS resolution
* Identical /etc/hosts files
* Can all talk to each other

## **Install Oracle Java 1.8 (or latest)**

Install Java 8 (Oracle version). You may have to tell it to use the new version using the “alternatives –config java” command.

If you download the latest version from Oracle, you can install as follows:

**sudo rpm –ivh jdk\_8u\_112.rpm**

# **Install ElasticSearch**

ElasticSearch can be installed with a package manager by adding Elastic's package repository.

Keep in mind with ES, the best load **performance is with more shards**, and **best query performance is with more replicas,** so you need to find a sweet spot with your setup.

## Get the Elastic.co GPG Key

Run the following command to import the Elasticsearch public GPG key into rpm:

**sudo rpm --import** [**http://packages.elastic.co/GPG-KEY-elasticsearch**](http://packages.elastic.co/GPG-KEY-elasticsearch)

## Create a new yum repository file for ElasticSearch

**vi /etc/yum.repos.d/elasticsearch.repo**

name=Elasticsearch repository for 2.x packages

baseurl=http://packages.elastic.co/elasticsearch/2.x/centos

gpgcheck=0

enabled=1

## Install ElasticSearch

**sudo yum -y install elasticsearch**

## Edit ElasticSearch Configuration

**sudo vi /etc/elasticsearch/elasticsearch.yml**

You will want to restrict outside access to your Elasticsearch instance (port 9200), so outsiders can't read your data or shutdown your Elasticsearch cluster through the HTTP API. Find the line that specifies network.host, uncomment it, and replace its value with "localhost" so it looks like this:

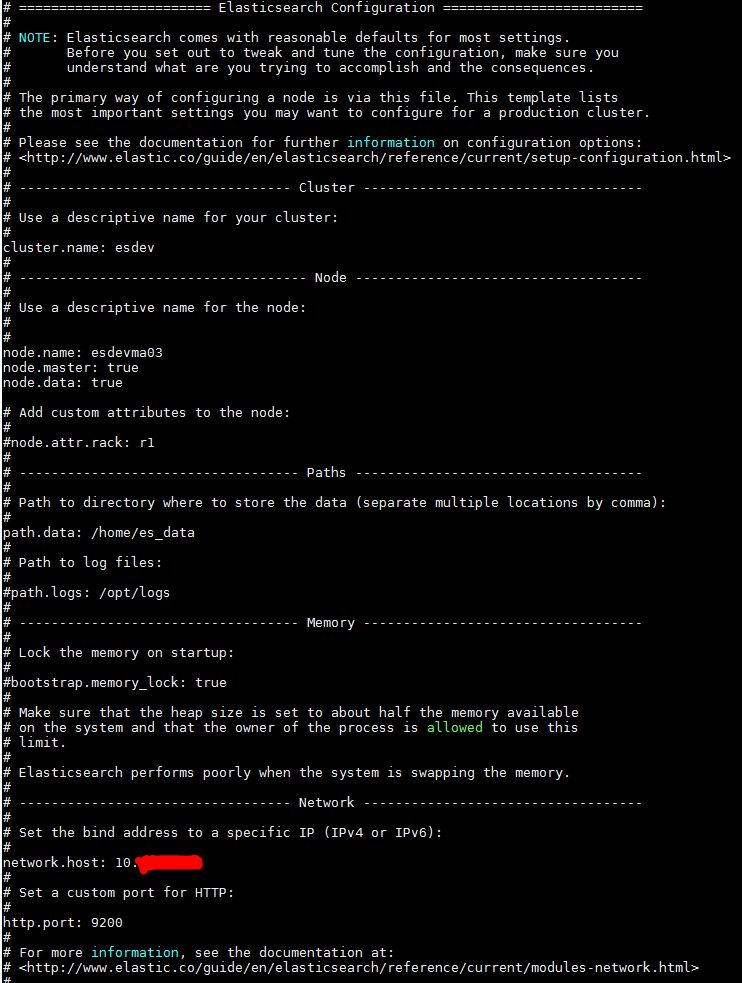


Figure 2 - Elasticsearch.yml Example

## Start ElasticSearch

**sudo systemctl start elasticsearch (RHEL 7)**

**sudo service elasticsearch start (RHEL6)**

**sudo systemctl enable elasticsearch (RHEL 7)**

**sudo chkconfig elasticsearch on (RHEL 6)**

\*\*REPEAT THOSE STEPS ON EACH NODE, BUT MODIFY /etc/elasticsearch/elasticsearch.yml accordingly for the cluster name and role.

## **NOTES For Running 5.X on older Kernels (CentOS 6.X)**

With older version of CentOS/RHEL running 5.X and newer, there are some kernel parameters that need to be modified.

You also need to turn of Swap by commenting out the swap mount in /etc/fstab

### Bootstrap Checks

/etc/elasticsearch/elasticsearch.yml add the following line:

**bootstrap.seccomp: false**

### Proc ULIMITS

/etc/security/limits.conf add the following line:

**elasticsearch - nproc 2048**

Java Options

/etc/sysconfig/elasticsearch uncomment and modify the line:

**ES\_JAVA\_OPTS=”-Xmx12g –Xms12g”**

## **Notes on Memory and Java Heap Size**

It’s possible to have too much heap enabled. The rule of thumb is you don’t want more than 31GB of Java Heap. Typically it’s half your RAM, but not more than 31GB. The reason for this is the garbage collection. Below is a typical log entry where this is happening. You see that it takes 29 seconds to regain the garbage:

[[2017-04-17T10:40:07,552][WARN ][o.e.m.j.](https://support.elastic.co/customers/s/feed/0D56100000qb9eDCAQ)**[JvmGc](https://support.elastic.co/customers/s/feed/0D56100000qb9eDCAQ)**[MonitorService] [tfbvag-deves2dn07] [gc][young][2368930][64052] duration [29.2s], collections [1]/[29.6s], total [29.2s]/[4.9h], memory [19.2gb]->[19.2gb]/[29.9gb], all\_pools {[young] [114.1mb]->[41.3mb]/[266.2mb]}{[survivor] [33.2mb]->[21.4mb]/[33.2mb]}{[old] [19.1gb]->[19.1gb]/[29.6gb]}  
[2017-04-17T10:40:07,553][WARN ][o.e.m.j.JvmGcMonitorService] [tfbvag-deves2dn07] [gc][2368930]](https://support.elastic.co/customers/s/feed/0D56100000qb9eDCAQ) **[overhead, spent [29.2s] collecting in the last [29.6s]](https://support.elastic.co/customers/s/feed/0D56100000qb9eDCAQ)**

## **NOTES ON GARBAGE COLLECTION**

Elasticsearch relies on garbage collection processes to free up heap memory. If you want to learn more about JVM garbage collection, check out [this guide](https://docs.oracle.com/javase/8/docs/technotes/guides/vm/gctuning/cms.html).

Because garbage collection uses resources (in order to free up resources!), you should keep an eye on its frequency and duration to see if you need to adjust the heap size. Setting the heap too large can result in long garbage collection times; these excessive pauses are dangerous because they can lead your cluster to mistakenly register your node as having dropped off the grid.

Java is a garbage-collected language, which means that the programmer does not manually manage memory allocation and deallocation. The programmer simply writes code, and the Java Virtual Machine (JVM) manages the process of allocating memory as needed, and then later cleaning up that memory when no longer needed.

When memory is allocated to a JVM process, it is allocated in a big chunk called the heap. The JVM then breaks the heap into two groups, referred to as generations:

Young (or Eden)

The space where newly instantiated objects are allocated. The young generation space is often quite small, usually 100 MB–500 MB. The young-gen also contains two survivor spaces.

Old

The space where older objects are stored. These objects are expected to be long-lived and persist for a long time. The old-gen is often much larger than the young-gen, and Elasticsearch nodes can see old-gens as large as 30 GB.

When an object is instantiated, it is placed into young-gen. When the young generation space is full, a young-gen garbage collection (GC) is started. Objects that are still "alive" are moved into one of the survivor spaces, and "dead" objects are removed. If an object has survived several young-gen GCs, it will be "tenured" into the old generation.

A similar process happens in the old generation: when the space becomes full, a garbage collection is started and dead objects are removed.

Nothing comes for free, however. Both the young- and old-generation garbage collectors have phases that "**stop the world**." During this time, the JVM literally halts execution of the program so it can trace the object graph and collect dead objects. During this stop-the-world phase, nothing happens. Requests are not serviced, pings are not responded to, and shards are not relocated. The world quite literally stops.

This isn’t a big deal for the young generation; its small size means GCs execute quickly. But the old-gen is quite a bit larger, and a slow GC here could mean 1s or even 15s of pausing—which is unacceptable for server software.

The garbage collectors in the JVM are very sophisticated algorithms and do a great job minimizing pauses. And Elasticsearch tries very hard to be garbage-collection friendly, by intelligently reusing objects internally, reusing network buffers, and enabling [Doc Values](https://www.elastic.co/guide/en/elasticsearch/guide/current/docvalues.html) by default. But ultimately, GC frequency and duration is a metric that needs to be watched by you, since it is the number one culprit for cluster instability.

A cluster that is frequently experiencing long GC will be a cluster that is under heavy load with not enough memory. These long GCs will make nodes drop off the cluster for brief periods. This instability causes shards to relocate frequently as Elasticsearch tries to keep the cluster balanced and enough replicas available. This in turn increases network traffic and disk I/O, all while your cluster is attempting to service the normal indexing and query load.

## **Install X-Pack Plugin**

X-Pack enables advanced features that are required for STIG mitigation at a minimum. If you want to use AD authentication, you need security from X-Pack (formerly called Shield).

First download the zip file for your version to each node:

**wget** [**https://artifacts.elastic.co/downloads/packs/x-pack/x-pack-5.3.0.zip**](https://artifacts.elastic.co/downloads/packs/x-pack/x-pack-5.3.0.zip)

Run the plug-in from your bin directory on each node. If you used the RPM package it’s /usr/share/elasticsearch/bin

/usr/share/elasticsearch/bin/elasticsearch-plugin install file:///path/to/file/x-pack-5.3.0.zip

**To uninstall run**

**/usr/share/elasticsearch/**bin/kibana-plugin remove x-pack

**Turn On/Off features you don’t want.**

By default, all X-Pack features are enabled. You can explicitly enable or disable X-Pack features in elasticsearch.yml and kibana.yml:

| **Setting** | **Description** |
| --- | --- |
| xpack.security.enabled | Set to false to disable X-Pack security. Configure in both elasticsearch.yml and kibana.yml. |
| xpack.monitoring.enabled | Set to false to disable X-Pack monitoring. Configure in both elasticsearch.yml and kibana.yml. |
| xpack.graph.enabled | Set to false to disable X-Pack graph. Configure in both elasticsearch.yml and kibana.yml. |
| xpack.watcher.enabled | Set to false to disable Watcher. Configure in elasticsearch.yml only. |
| xpack.reporting.enabled | Set to false to disable X-Pack reporting. Configure in kibana.yml only. |

Setting up Active Directory Authentication

NOTE: Once you enable an additional realm (ie for AD), you have to have an entry for the native realm for local accounts to work.

If you want to enable AD authentication, you need to add the AD realm to your /etc/elasticsearch/elasticsearch.yml file. An example is below:

xpack:

ssl.verification\_mode: none

security:

authc:

realms:

active\_directory:

type: active\_directory

order: 0

domain\_name: agora.local

url: "ldap://dc1.agora.local:389"

files:

role\_mapping: "/etc/elasticsearch/x-pack/role\_mapping.yml"

unmapped\_groups\_as\_roles: true

native:

type: native

In the above example, note we added the native realm at the bottom for local logins. We also point to the role\_mapping.yml file which maps your AD groups to your XPack roles. Below is an example of /etc/elasticsearch/x-pack/role\_mapping.yml file:

role\_mapping.yml

# Role mapping configuration file which has elasticsearch roles as keys

# that map to one or more user or group distinguished names

#roleA: this is an elasticsearch role

# - groupA-DN this is a group distinguished name

# - groupB-DN

# - user1-DN this is the full user distinguished name

#power\_user:

# - "cn=admins,dc=example,dc=com"

#user:

# - "cn=users,dc=example,dc=com"

# - "cn=admins,dc=example,dc=com"

# - "cn=John Doe,cn=other users,dc=example,dc=com"

superuser:

- "cn=IT\_ADMIN,ou=Groups,ou=AGORA,dc=agora,dc=local"

# - "cn=Mcafee John ctr,ou=Users,ou=TFB,OU=Sites,ou=AGORA,dc=agora,dc=local"

kibana\_user:

- "cn=TFB\_DEV\_USERS,ou=Groups,ou=TFB,ou=Sites,ou=AGORA,dc=agora,dc=local"

- "cn=Domain Users,cn=Users,dc=agora,dc=local"

sbir\_access\_role:

- "cn=CDM\_DEVELOPER,ou=Groups,ou=AGORA,dc=agora,dc=local"

#monitoring\_user:

#- "cn=Vanacore Michael CTR,ou=Users,ou=TFB,ou=Sites,ou=AGORA,dc=agora,dc=local"

# - "cn=Domain Users,dc=agora,dc=local"

## **Basic Cluster Checks**

On Clusters w/ HAProxy Running Port 8080 instead of 9200 and 8081 instead of 9300 and 80 instead of 5601.

using curl for api calls directly.

#display the current health of the cluster

curl <http://mldev02:8080/_cluster/health?pretty=true>

#display current nodes in the cluster

curl <http://mldev02:8080/_cat/nodes>

#display the indices

curl <http://mldev02:8080/_cat/indices>

#display the shards

curl <http://mldev02:8080/_cat/shards>

#Get Version Info

curl -XGET 'http://localhost:9200'

If using xpack security:

curl –u tony –XGET <http://10.10.10.23:9200/_cat>

or

curl --user tony:password -XPUT 'localhost:9200/idx'

### **Troubleshooting**

1. Do a generic health check
   1. curl –u tony –XGET <http://10.10.10.123:9200/_cluster/health?pretty=true>
2. Then if not green, see if it’s cluster OS related (jvm heap or I/O) or load related (indices)
   1. curl –u tony –XGET <http://10.10.19.123:9200/_cat/indices>
3. If you think it may be OS related, check the logs,
   1. /opt/es/logs or /var/log/messages or wherever your elasticsearch.yml says you logs are
4. Check the nodes individually for specific things.
   1. curl -u elastic <http://10.10.10.21:9200/_nodes/esdevma01/stats/jvm> (or http or os)

Get HEAP Size

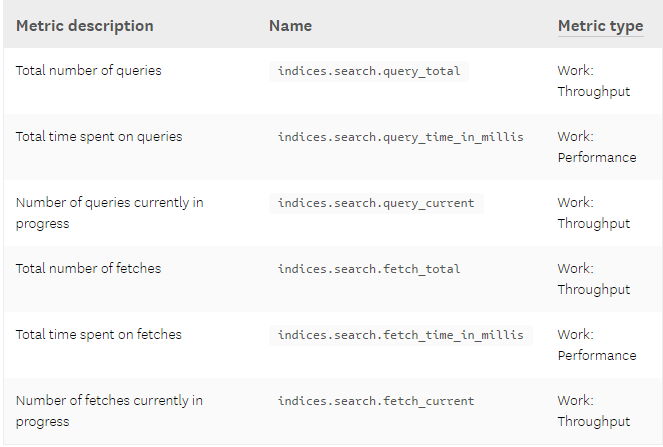
curl -u elastic -XGET <http://10.11.16.21:9200/_cat/nodes?h=heap.max>

You can then log into the servers that are down and **service elasticsearch start**

### These are the main config files for ES

|  |  |  |
| --- | --- | --- |
| home | /usr/share/elasticsearch | Home of elasticsearch installation |
| bin | /usr/share/elasticsearch/bin | Binary scripts including elasticsearch to start a node. |
| conf | /etc/elasticsearch | Configuration files elasticsearch.yml and logging.yml. |
| conf | /etc/sysconfig/elasticsearch | Environment variables including heap size, file descriptors. |
| data | /var/lib/elasticsearch | Default path to store elasticsearch data . |
| logs | /var/log/elasticsearch | Default log location. |
| plugins | /usr/share/elasticsearch/plugins | Plugin directory, all plugin installed in separate directory here. |
| script | /etc/elasticsearch/scripts | Default script files location. |

### Key Query Metric indices



| **Metric description** | **Name** | [**Metric type**](https://www.datadoghq.com/blog/monitoring-101-collecting-data/) |
| --- | --- | --- |
| Total number of queries | indices.search.query\_total | Work: Throughput |
| Total time spent on queries | indices.search.query\_time\_in\_millis | Work: Performance |
| Number of queries currently in progress | indices.search.query\_current | Work: Throughput |
| Total number of fetches | indices.search.fetch\_total | Work: Throughput |
| Total time spent on fetches | indices.search.fetch\_time\_in\_millis | Work: Performance |
| Number of fetches currently in progress | indices.search.fetch\_current | Work: Throughput |

#### Search performance metrics to watch

**Query load:** Monitoring the number of queries currently in progress can give you a rough idea of how many requests your cluster is dealing with at any particular moment in time. Consider alerting on unusual spikes or dips that may point to underlying problems. You may also want to monitor the size of the search thread pool queue, which we will explain in further detail [later on in this post](https://www.datadoghq.com/blog/monitor-elasticsearch-performance-metrics/#metrics-to-watch).

**Query latency:** Though Elasticsearch does not explicitly provide this metric, monitoring tools can help you use the available metrics to calculate the average query latency by sampling the total number of queries and the total elapsed time at regular intervals. Set an alert if latency exceeds a threshold, and if it fires, look for potential resource bottlenecks, or investigate whether you need to [optimize your queries](https://www.datadoghq.com/blog/elasticsearch-performance-scaling-problems/).

**Fetch latency:** The second part of the search process, the fetch phase, should typically take much less time than the query phase. If you notice this metric consistently increasing, this could indicate a problem with slow disks, [enriching of documents](https://www.elastic.co/guide/en/elasticsearch/guide/current/highlighting-intro.html) (highlighting relevant text in search results, etc.), or [requesting too many results](https://www.elastic.co/guide/en/elasticsearch/guide/current/pagination.html).

| **Metric description** | **Name** | [**Metric type**](https://www.datadoghq.com/blog/monitoring-101-collecting-data/) |
| --- | --- | --- |
| Total number of documents indexed | indices.indexing.index\_total | Work: Throughput |
| Total time spent indexing documents | indices.indexing.index\_time\_in\_millis | Work: Performance |
| Number of documents currently being indexed | indices.indexing.index\_current | Work: Throughput |
| Total number of index refreshes | indices.refresh.total | Work: Throughput |
| Total time spent refreshing indices | indices.refresh.total\_time\_in\_millis | Work: Performance |
| Total number of index flushes to disk | indices.flush.total | Work: Throughput |
| Total time spent on flushing indices to disk | indices.flush.total\_time\_in\_millis | Work: Performance |

# **Install Kibana**

The Kibana package shares the same GPG Key as ElasticSearch, and we already installed that public key.

## Create Repository for Kibana

sudo vi /etc/yum.repos.d/kibana.repo

[kibana-4.4]

name=Kibana repository for 4.4.x packages

baseurl=http://packages.elastic.co/kibana/4.4/centos

gpgcheck=0

enabled=1

## Install Kibana

**sudo yum -y install kibana**

## Modify Kibana Configuration

**sudo vi /opt/kibana/config/kibana.yml**

In the Kibana configuration file, find the line that specifies server.host, and replace the IP address ("0.0.0.0" by default) with "localhost":

kibana.yml excerpt (updated)

server.host: "localhost"

Save and exit. This setting makes it so Kibana will only be accessible to the localhost. This is fine because we will install an Nginx reverse proxy, on the same server, to allow external access.

## Start and Enable Kibana Service

**sudo systemctl start kibana (RHEL 7)**

**sudo service kibana start (RHEL6)**

**sudo chkconfig kibana on (RHEL 6)**

**sudo systemctl enable kibana (RHEL 7)**

NOTE: You may have to modify the kibana user in the start up script if you find it’s not accessible. In **/etc/systemd/system/kibana.service** file change User: Kibana to User: root. I ran into it looked like it was started, but not accessible. This fixed that.

# Set Up Reverse Proxy for Web Access and Authentication

There are multiple ways to set up proxying for Kibana. You can set it up where when a user hits port 80, they are redirected to port 5601. Proxying also allows for authentication control. The two most common ways to accomplish this are to use Apache or Nginx.

## **Nginx as a Proxy**

### Install Nginx

Because we configured Kibana to listen on localhost, we must set up a reverse proxy to allow external access to it. We will use Nginx for this purpose.

**Note:** If you already have an Nginx instance that you want to use, feel free to use that instead. Just make sure to configure Kibana so it is reachable by your Nginx server (you probably want to change the host value, in /opt/kibana/config/kibana.yml, to your Kibana server's private IP address). Also, it is recommended that you enable SSL/TLS.

#### Install nginx Package

**sudo yum -y install nginx httpd-tools**

*NOTE: Nginx is available in the EPEL repository*

#### Create Kibana Admin User

Use htpasswd to create an admin user, called "kibanaadmin" (you could use another name), that can access the Kibana web interface:

**sudo htpasswd -c /etc/nginx/htpasswd.users kibanaadmin**

Enter a password at the prompt. Remember this login, as you will need it to access the Kibana web interface.

#### Edit nginx configuration for proxy

**sudo vi /etc/nginx/nginx.conf**

Find the default server block (starts with server {), the last configuration block in the file, and delete it. When you are done, the last two lines in the file should look like this:

nginx.conf excerpt

include /etc/nginx/conf.d/\*.conf;

}

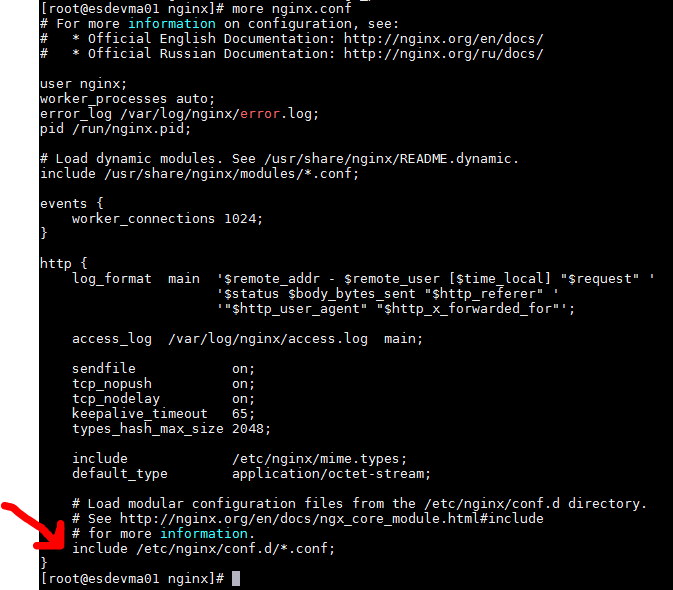


Figure 3 - nginx.conf example

Save and exit.

#### Create New Server Block

**sudo vi /etc/nginx/conf.d/kibana.conf**

Paste the following code block into the file. Be sure to update the server\_name to match your server's name:

/etc/nginx/conf.d/kibana.conf

**server {**

**listen 80;**

**server\_name example.com;**

**auth\_basic "Restricted Access";**

**auth\_basic\_user\_file /etc/nginx/htpasswd.users;**

**location / {**

**proxy\_pass http://localhost:5601;**

**proxy\_http\_version 1.1;**

**proxy\_set\_header Upgrade $http\_upgrade;**

**proxy\_set\_header Connection 'upgrade';**

**proxy\_set\_header Host $host;**

**proxy\_cache\_bypass $http\_upgrade;**

**}**

**}**

Save and exit. This configures Nginx to direct your server's HTTP traffic to the Kibana application, which is listening on localhost:5601. Also, Nginx will use the htpasswd.users file, that we created earlier, and require basic authentication.

#### Start nginx

**sudo systemctl start nginx (RHEL 7)**

**sudo service nginx start (RHEL 6)**

**sudo systemctl enable nginx (RHEL 7)**

**sudo chkconfig nginx on (RHEL 6)**

**Note:** This tutorial assumes that SELinux is disabled. If this is not the case, you may need to run the following command for Kibana to work properly: **sudo setsebool -P httpd\_can\_network\_connect 1**

Kibana is now accessible via your FQDN or the public IP address of your ELK Server i.e. <http://yourhostname>. If you go there in a web browser, after entering the "kibanaadmin" credentials, you should see a Kibana welcome page which will ask you to configure an index pattern.

## **Apache as a Proxy**

The setup is very similar to nginx, but the format of the conf files is a bit different.

### Install Apache

**yum groupinstall –y web-server** (or just **yum install –y httpd**)

### Configure Apache

Go to **/etc/httpd/conf.d** and create two files: kibana.conf and ldap\_auth.conf

**Kibana.conf Example:**

**<VirtualHost \*:80>**

**Servername kibanaserver**

**ProxyPreserveHost On**

**ProxyRequests On**

**ProxyPass /** [**http://127.0.0.1:5601/**](http://127.0.0.1:5601/)

**PrxyPassReverse /** [**http://127.0.0.1/**](http://127.0.0.1/)

**</VirtualHost>**

**ldap\_auth.conf Example:**

**<Location />**

**AuthType Basic**

**AuthName “LDAP Service Acct”**

**AuthBasicProvider ldap**

**AuthzLDAPAuthoritative on**

**AuthLDAPURL “ldap://dcserver01.your.domain:3268/DC=your,DC=domain?sAMAccountName?sub?(objectClass=\*)”**

**AuthLDAPBindDN** [**serviceacct.svc@your.domain**](mailto:serviceacct.svc@your.domain)(whatever the name of your AD service account)

**AuthLDAPBindPassword “Y0urLD@pP@$$word”**

**Require user Anthony.crotwell bob.smith cecroam zf76554** (Whatever the AD Usernames are you want access)

**</Location>**

### Start Apache

**sudo service httpd start** (RHEL 6)

**sudo systemctl start httpd** (RHEL 7)

**sudo chkconfig httpd** on (RHEL 6)

**sudo systemctl enable httpd** (RHEL 7)

You should now be able to hit <http://kibanaserver.your.domain> and get Kibana.

## **Kibana Plugins**

There are a couple different methods to install plugins for Kibana. Some are as simple as running the install script, others you have to install node.js and then do a npm install.

### **Get node.js on the Kibana server**

Grab and execute the setup script. This script verifies your OS setup prior to node.js install.

**curl –silent –location** [**http://spacewalk/cobbler/repo\_mirror/node-js/setup\_6.x**](http://spacewalk/cobbler/repo_mirror/node-js/setup_6.x) **| bash -**

This will kick of the yum install, but you can manually do it by **yum install –y nodejs**

### **Install the plugin via node package manager**

The first step is to grab the plugin directory either via git or download the tar file, then run **npm install.**

**Example:**

cd KIBANA\_HOME/plugins

git clone https://github.com/virusu/3D\_kibana\_charts\_vis.git 3D\_kibana\_charts\_vis

cd 3D\_kibana\_charts\_vis

npm install

If you don’t have access to the github, you can download the tar file from Spacewalk and untar in the plugin directory (/usr/share/kibana/plugins)

Note: If the plug does not install due to unsupported version, you can modify the version number in the package.json file at the top level of the plugins directory to match your running version.

# **Install Logstash**

The Logstash package shares the same GPG Key as Elasticsearch, and we already installed that public key, so let's create and edit a new Yum repository file for Logstash:

sudo vi /etc/yum.repos.d/logstash.repo

## Add Logstash Repository

/etc/yum.repos.d/logstash.repo

[logstash-2.2]

name=logstash repository for 2.2 packages

baseurl=http://packages.elasticsearch.org/logstash/2.2/centos

gpgcheck=0

enabled=1

Save and exit.

## Install Logstash Package

**sudo yum -y install logstash**

## Generate SSL Certificates

Since we are going to use Filebeat to ship logs from our Client Servers to our ELK Server, we need to create an SSL certificate and key pair. The certificate is used by Filebeat to verify the identity of ELK Server. Create the directories that will store the certificate and private key with the following commands:

Now you have two options for generating your SSL certificates. If you have a DNS setup that will allow your client servers to resolve the IP address of the ELK Server, use **Option 2**. Otherwise, **Option 1** will allow you to use IP addresses.

### Option 1: IP Address

If you don't have a DNS setup—that would allow your servers, that you will gather logs from, to resolve the IP address of your ELK Server—you will have to add your ELK Server's private IP address to the subjectAltName (SAN) field of the SSL certificate that we are about to generate. To do so, open the OpenSSL configuration file:

sudo vi /etc/pki/tls/openssl.cnf

Find the [ v3\_ca ] section in the file, and add this line under it (substituting in the ELK Server's private IP address):

openssl.cnf excerpt

subjectAltName = IP: ELK\_server\_private\_ip

Save and exit.

Now generate the SSL certificate and private key in the appropriate locations (/etc/pki/tls/), with the following commands:

**cd /etc/pki/tls**

**sudo openssl req -config /etc/pki/tls/openssl.cnf -x509 -days 3650 -batch -nodes -newkey rsa:2048 -keyout private/logstash-forwarder.key -out certs/logstash-forwarder.crt**

The *logstash-forwarder.crt* file will be copied to all of the servers that will send logs to Logstash but we will do that a little later. Let's complete our Logstash configuration. If you went with this option, skip option 2 and move on to **Configure Logstash**.

### Option 2: FQDN (DNS)

If you have a DNS setup with your private networking, you should create an A record that contains the ELK Server's private IP address—this domain name will be used in the next command, to generate the SSL certificate. Alternatively, you can use a record that points to the server's public IP address. Just be sure that your servers (the ones that you will be gathering logs from) will be able to resolve the domain name to your ELK Server.

Now generate the SSL certificate and private key, in the appropriate locations (/etc/pki/tls/...), with the following command (substitute in the FQDN of the ELK Server):

* **cd /etc/pki/tls**
* **sudo openssl req -subj '/CN=ELK\_server\_fqdn/' -x509 -days 3650 -batch -nodes -newkey rsa:2048 -keyout private/logstash-forwarder.key -out certs/logstash-forwarder.crt**

The *logstash-forwarder.crt* file will be copied to all of the servers that will send logs to Logstash but we will do that a little later. Let's complete our Logstash configuration.

## Configure Logstash

Logstash configuration files are in the JSON-format, and reside in /etc/logstash/conf.d. The configuration consists of three sections: inputs, filters, and outputs.

Create a configuration file called 02-beats-input.conf and set up our "filebeat" input:

* sudo vi /etc/logstash/conf.d/02-beats-input.conf

Insert the following **input** configuration:

02-beats-input.conf

input {

beats {

port => 5044

ssl => true

ssl\_certificate => "/etc/pki/tls/certs/logstash-forwarder.crt"

ssl\_key => "/etc/pki/tls/private/logstash-forwarder.key"

}

}

Save and quit. This specifies a beats input that will listen on tcp port 5044, and it will use the SSL certificate and private key that we created earlier.

Now create a configuration file called 10-syslog-filter.conf, where we will add a filter for syslog messages:

* sudo vi /etc/logstash/conf.d/10-syslog-filter.conf

Insert the following syslog **filter** configuration:

10-syslog-filter.conf

filter {

if [type] == "syslog" {

grok {

match => { "message" => "%{SYSLOGTIMESTAMP:syslog\_timestamp} %{SYSLOGHOST:syslog\_hostname} %{DATA:syslog\_program}(?:\[%{POSINT:syslog\_pid}\])?: %{GREEDYDATA:syslog\_message}" }

add\_field => [ "received\_at", "%{@timestamp}" ]

add\_field => [ "received\_from", "%{host}" ]

}

syslog\_pri { }

date {

match => [ "syslog\_timestamp", "MMM d HH:mm:ss", "MMM dd HH:mm:ss" ]

}

}

}

Save and quit. This filter looks for logs that are labeled as "syslog" type (by Filebeat), and it will try to use grok to parse incoming syslog logs to make it structured and query-able.

Lastly, we will create a configuration file called 30-elasticsearch-output.conf:

* sudo vi /etc/logstash/conf.d/30-elasticsearch-output.conf

Insert the following **output** configuration:

/etc/logstash/conf.d/30-elasticsearch-output.conf

output {

elasticsearch {

hosts => ["localhost:9200"]

sniffing => true

manage\_template => false

index => "%{[@metadata][beat]}-%{+YYYY.MM.dd}"

document\_type => "%{[@metadata][type]}"

}

}

Save and exit. This output basically configures Logstash to store the beats data in ElasticSearch which is running at localhost:9200, in an index named after the beat used (filebeat, in our case).

If you want to add filters for other applications that use the Filebeat input, be sure to name the files so they sort between the input and the output configuration (i.e. between 02- and 30-).

## Test your Logstash configuration

**sudo service logstash configtest**

It should display Configuration OK if there are no syntax errors. Otherwise, try and read the error output to see what's wrong with your Logstash configuration.

Restart and enable Logstash to put our configuration changes into effect:

**sudo systemctl restart logstash (RHEL 7)**

**sudo service logstash restart (RHEL 6)**

**sudo chkconfig logstash on (RHEL 6)**

**sudo systemctl enable logstash (RHEL 7)**

# Load Kibana Dashboards

Elastic provides several sample Kibana dashboards and Beats index patterns that can help you get started with Kibana. Although we won't use the dashboards in this tutorial, we'll load them anyway so we can use the Filebeat index pattern that it includes.

First, download the sample dashboards archive to your home directory:

cd ~

curl -L -O https://download.elastic.co/beats/dashboards/beats-dashboards-1.1.0.zip

Install the unzip package with this command:

**sudo yum -y install unzip**

Next, extract the contents of the archive:

**unzip beats-dashboards-\*.zip**

And load the sample dashboards, visualizations and Beats index patterns into Elasticsearch with these commands:

cd beats-dashboards-\*

./load.sh

These are the index patterns that we just loaded:

[packetbeat-]YYYY.MM.DD

[topbeat-]YYYY.MM.DD

[filebeat-]YYYY.MM.DD

[winlogbeat-]YYYY.MM.DD

When we start using Kibana, we will select the Filebeat index pattern as our default.

# Load Filebeat Index Template in ElasticSearch

Because we are planning on using Filebeat to ship logs to ElasticSearch, we should load a Filebeat index template. The index template will configure ElasticSearch to analyze incoming Filebeat fields in an intelligent way.

First, download the Filebeat index template to your home directory:

cd ~

curl -O https://gist.githubusercontent.com/thisismitch/3429023e8438cc25b86c/raw/d8c479e2a1adcea8b1fe86570e42abab0f10f364/filebeat-index-template.json

Then load the template with this command:

curl -XPUT 'http://localhost:9200/\_template/filebeat?pretty' -d@filebeat-index-template.json

If the template loaded properly, you should see a message like this:

Output:

{

"acknowledged" : true

}

Now that our ELK Server is ready to receive Filebeat data, let's move onto setting up Filebeat on each client server.

# Set Up Filebeat (Add Client Servers)

Do these steps for each **CentOS or RHEL 7** server that you want to send logs to your ELK Server. For instructions on installing Filebeat on Debian-based Linux distributions (e.g. Ubuntu, Debian, etc.).

# Copy SSL Certificate

On your **ELK Server**, copy the SSL certificate created earlier to your **Client Server** (substitute the client server's address, and your own login):

**scp /etc/pki/tls/certs/logstash-forwarder.crt user@client\_server\_private\_address:/tmp**

After providing your login's credentials, ensure that the certificate copy was successful. It is required for communication between the client servers and the ELK Server.

Now, on your **Client Server**, copy the ELK Server's SSL certificate into the appropriate location (/etc/pki/tls/certs):

sudo mkdir -p /etc/pki/tls/certs

sudo cp /tmp/logstash-forwarder.crt /etc/pki/tls/certs/

# Install Filebeat Package

## Create Filebeat Repository File:

* sudo vi /etc/yum.repos.d/elastic-beats.repo

Add the following repository configuration:

/etc/yum.repos.d/elastic-beats.repo

[beats]

name=Elastic Beats Repository

baseurl=https://packages.elastic.co/beats/yum/el/$basearch

enabled=1

gpgcheck=0

Save and exit.

Install Filebeat with this command

**sudo yum -y install filebeat**

## Configure Filebeat

Now configure Filebeat to connect to Logstash on our ELK Server. This section will step you through modifying the example configuration file that comes with Filebeat. When you complete the steps, you should have a file that looks something like [this](https://gist.githubusercontent.com/thisismitch/3429023e8438cc25b86c/raw/de660ffdd3decacdcaf88109e5683e1eef75c01f/filebeat.yml-centos).

On **Client Server**, create and edit Filebeat configuration file:

**sudo vi /etc/filebeat/filebeat.yml**

Near the top of the file, you will see the prospectors section, which is where you can define **prospectors** that specify which log files should be shipped and how they should be handled. Each prospector is indicated by the - character.

Modify the existing prospector to send secure and messages logs to Logstash. Under paths, comment out the - /var/log/\*.log file. This will prevent Filebeat from sending every .log in that directory to Logstash. Then add new entries for syslog and auth.log. It should look something like this when you're done:

filebeat.yml excerpt 1 of 5

...

paths:

- /var/log/secure

- /var/log/messages

# - /var/log/\*.log

...

Then find the line that specifies document\_type:, uncomment it and change its value to "syslog". It should look like this after the modification:

filebeat.yml excerpt 2 of 5

...

document\_type: syslog

...

This specifies that the logs in this prospector are of type **syslog** (which is the type that our Logstash filter is looking for).

If you want to send other files to your ELK server, or make any changes to how Filebeat handles your logs, feel free to modify or add prospector entries.

Next, under the output section, find the line that says elasticsearch:, which indicates the Elasticsearch output section (which we are not going to use). **Delete or comment out the entire Elasticsearch output section** (up to the line that says logstash:).

Find the commented out Logstash output section, indicated by the line that says #logstash:, and uncomment it by deleting the preceding #. In this section, uncomment the hosts: ["localhost:5044"]line. Change localhost to the private IP address (or hostname, if you went with that option) of your ELK server:

filebeat.yml excerpt 3 of 5

### Logstash as output

logstash:

# The Logstash hosts

hosts: ["ELK\_server\_private\_IP:5044"]

This configures Filebeat to connect to Logstash on your ELK Server at port 5044 (the port that we specified an input for earlier).

Directly under the hosts entry, and with the same indentation, add this line:

filebeat.yml excerpt 4 of 5

bulk\_max\_size: 1024

Next, find the tls section, and uncomment it. Then uncomment the line that specifies certificate\_authorities, and change its value to ["/etc/pki/tls/certs/logstash-forwarder.crt"]. It should look something like this:

filebeat.yml excerpt 5 of 5

...

tls:

# List of root certificates for HTTPS server verifications

certificate\_authorities: ["/etc/pki/tls/certs/logstash-forwarder.crt"]

This configures Filebeat to use the SSL certificate that we created on the ELK Server.

Save and quit.

Now start and enable Filebeat to put our changes into place:

sudo systemctl start filebeat

sudo systemctl enable filebeat

Again, if you're not sure if your Filebeat configuration is correct, compare it against this [example Filebeat configuration](https://gist.githubusercontent.com/thisismitch/3429023e8438cc25b86c/raw/de660ffdd3decacdcaf88109e5683e1eef75c01f/filebeat.yml-centos).

Now Filebeat is sending your syslog messages and secure files to your ELK Server! Repeat this section for all of the other servers that you wish to gather logs for.

## Test Filebeat Installation

If your ELK stack is setup properly, Filebeat (on your client server) should be shipping your logs to Logstash on your ELK server. Logstash should be loading the Filebeat data into Elasticsearch in a date-stamped index, filebeat-YYYY.MM.DD.

On your **ELK Server**, verify that Elasticsearch is indeed receiving the data by querying for the Filebeat index with this command:

curl -XGET 'http://localhost:9200/filebeat-\*/\_search?pretty'

You should see a bunch of output that looks like this:

Sample Output:

...

{

"\_index" : "filebeat-2016.01.29",

"\_type" : "log",

"\_id" : "AVKO98yuaHvsHQLa53HE",

"\_score" : 1.0,

"\_source":{"message":"Feb 3 14:34:00 rails sshd[963]: Server listening on :: port 22.","@version":"1","@timestamp":"2016-01-29T19:59:09.145Z","beat":{"hostname":"topbeat-u-03","name":"topbeat-u-03"},"count":1,"fields":null,"input\_type":"log","offset":70,"source":"/var/log/auth.log","type":"log","host":"topbeat-u-03"}

}

...

If your output shows 0 total hits, Elasticsearch is not loading any logs under the index you searched for, and you should review your setup for errors. If you received the expected output, continue to the next step.

# Connect to Kibana

When you are finished setting up Filebeat on all of the servers that you want to gather logs for, let's look at Kibana, the web interface that we installed earlier.

In a web browser, go to the FQDN or public IP address of your ELK Server. After entering the "kibanaadmin" credentials, you should see a page prompting you to configure a default index pattern: