How To Install Elasticsearch, Logstash, and Kibana (Elastic Stack) on Ubuntu 18.04

Ubuntu | Monitoring | Configuration Management | Elasticsearch | Ubuntu 18.04

By Justin Ellingwood and Vadym Kalsin
Published on November 6, 2018 342.6k

English

Not using Ubuntu 18.04?
Choose a different version or distribution.

Ubuntu 18.04

The author selected the Internet Archive to receive a donation as part of the Write for DOnations program.

Introduction

The Elastic Stack — formerly known as the *ELK Stack* — is a collection of open-source software produced by Elastic which allows you to search, analyze, and visualize logs generated from any source in any format, a practice known as *centralized logging*. Centralized logging can be very useful when attempting to identify problems with your servers or applications, as it allows you to search through all of your logs in a single place. It's also useful because it allows you to identify issues that span multiple servers by correlating their logs during a specific time frame.

The Elastic Stack has four main components:

- **Elasticsearch**: a distributed *RESTful* search engine which stores all of the collected data.
- **Logstash**: the data processing component of the Elastic Stack which sends incoming data to Elasticsearch.
- **Kibana**: a web interface for searching and visualizing logs.
- **Beats**: lightweight, single-purpose data shippers that can send data from hundreds or thousands of machines to either Logstash or Elasticsearch.

In this tutorial, you will install the Elastic Stack on an Ubuntu 18.04 server. You will learn how to install all of the components of the Elastic Stack — including Filebeat, a Beat used for forwarding and centralizing logs and files — and configure them to gather and visualize system logs. Additionally, because Kibana is normally only available on the `localhost`, we will use Nginx to proxy it so it will be accessible over a web browser. We will install all of these components on a single server, which we will refer to as our *Elastic Stack server*.

Note: When installing the Elastic Stack, you must use the same version across the entire stack. In this tutorial we will install the latest versions of the entire stack which are, at the time of this writing, Elasticsearch 7.6.1, Kibana 7.6.1, Logstash 7.6.1, and Filebeat 7.6.1.

Prerequisites

To complete this tutorial, you will need the following:

- An Ubuntu 18.04 server set up by following our Initial Server Setup Guide for Ubuntu 18.04, including a non-root user with sudo privileges and a firewall configured with `ufw`. The amount of CPU, RAM, and storage that your Elastic Stack server will require depends on the volume of logs that you intend to gather. For this tutorial, we will be using a VPS with the following specifications for our Elastic Stack server:

    - OS: Ubuntu 18.04
    - RAM: 4GB
    - CPU: 2
- Java 8 — which is required by Elasticsearch and Logstash — installed on your server. Note that Java 9 is not supported. To install this, follow the "Installing the Oracle JDK" section of our guide on how to install Java 8 on Ubuntu 18.04.
- Nginx installed on your server, which we will configure later in this guide as a reverse proxy for Kibana. Follow our guide on How to Install Nginx on Ubuntu 18.04 to set this up.

Additionally, because the Elastic Stack is used to access valuable information about your server that you would not want unauthorized users to access, it's important that you keep your server secure by installing a TLS/SSL certificate. This is optional but **strongly encouraged**.

However, because you will ultimately make changes to your Nginx server block over the course of this guide, it would likely make more sense for you to complete the Let's Encrypt on Ubuntu 18.04 guide at the end of this tutorial's second step. With that in mind, if you plan to configure Let's Encrypt on your server, you will need the following in place before doing so:

- A fully qualified domain name (FQDN). This tutorial will use `example.com` throughout. You can purchase a domain name on Namecheap, get one for free on Freenom, or use the domain registrar of your choice.
- Both of the following DNS records set up for your server. You can follow this introduction to DigitalOcean DNS for details on how to add them.
    - An A record with `example.com` pointing to your server's public IP address.
    - An A record with `www.example.com` pointing to your server's public IP address.

Step 1 — Installing and Configuring Elasticsearch

The Elastic Stack components are not available in Ubuntu's default package repositories. They can, however, be installed with APT after adding Elastic's package source list.

All of the Elastic Stack's packages are signed with the Elasticsearch signing key in order to protect your system from package spoofing. Packages which have been authenticated using the key will be considered trusted by your package manager. In this step, you will import the Elasticsearch public GPG key and add the Elastic package source list in order to install Elasticsearch.

To begin, run the following command to import the Elasticsearch public GPG key into APT:

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
```

Next, add the Elastic source list to the `sources.list.d` directory, where APT will look for new sources:

```
echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list
```

Next, update your package lists so APT will read the new Elastic source:

```
sudo apt update
```

Then install Elasticsearch with this command:

```
sudo apt install elasticsearch
```

Once Elasticsearch is finished installing, use your preferred text editor to edit Elasticsearch's main configuration file, `elasticsearch.yml`. Here, we'll use `nano`:

```
sudo nano /etc/elasticsearch/elasticsearch.yml
```

Note: Elasticsearch's configuration file is in YAML format, which means that indentation is very important! Be sure that you do not add any extra spaces as you edit this file.

Elasticsearch listens for traffic from everywhere on port `9200`. You will want to restrict outside access to your Elasticsearch instance to prevent outsiders from reading your data or shutting down your Elasticsearch cluster through the REST API. Find the line that specifies `network.host`, uncomment it, and replace its value with `localhost` so it looks like this:

/etc/elasticsearch/elasticsearch.yml

```
. . .
network.host: localhost
. . .
```

Save and close `elasticsearch.yml` by pressing `CTRL+X`, followed by `Y` and then `ENTER` if you're using `nano`. Then, start the Elasticsearch service with `systemctl`:

```
sudo systemctl start elasticsearch
```

Next, run the following command to enable Elasticsearch to start up every time your server boots:

```
sudo systemctl enable elasticsearch
```

You can test whether your Elasticsearch service is running by sending an HTTP request:

```
curl -X GET "localhost:9200"
```

You will see a response showing some basic information about your local node, similar to this:

```
Output
{
  "name" : "ElasticSearch",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "SMYhVWRiTwS1dF0pQ-h7SQ",
  "version" : {
    "number" : "7.6.1",
```

```
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "aa751e09be0a5072e8570670309b1f12348f023b",
    "build_date" : "2020-02-29T00:15:25.529771Z",
    "build_snapshot" : false,
    "lucene_version" : "8.4.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

Now that Elasticsearch is up and running, let's install Kibana, the next component of the Elastic Stack.

Step 2 — Installing and Configuring the Kibana Dashboard

According to the official documentation, you should install Kibana only after installing Elasticsearch. Installing in this order ensures that the components each product depends on are correctly in place.

Because you've already added the Elastic package source in the previous step, you can just install the remaining components of the Elastic Stack using `apt`:

```
sudo apt install kibana
```

Then enable and start the Kibana service:

```
sudo systemctl enable kibana
sudo systemctl start kibana
```

Because Kibana is configured to only listen on `localhost`, we must set up a reverse proxy to allow external access to it. We will use Nginx for this purpose, which should already be installed on your server.

First, use the `openssl` command to create an administrative Kibana user which you'll use to access the Kibana web interface. As an example we will name this account `kibanaadmin`, but to ensure greater security we recommend that you choose a non-standard name for your user that would be difficult to guess.

The following command will create the administrative Kibana user and password, and store them in the `htpasswd.users` file. You will configure Nginx to require this username and password and read this file momentarily:

```
echo "kibanaadmin:`openssl passwd -apr1`" | sudo tee -a /etc/nginx/htpasswd.users
```

Enter and confirm a password at the prompt. Remember or take note of this login, as you will need it to access the Kibana web interface.

Next, we will create an Nginx server block file. As an example, we will refer to this file as `example.com`, although you may find it helpful to give yours a more descriptive name. For instance, if you have a FQDN and DNS records set up for this server, you could name this file after your FQDN:

```
sudo nano /etc/nginx/sites-available/example.com
```

Add the following code block into the file, being sure to update `example.com` to match your server's FQDN or public IP address. This code configures Nginx to direct your server's HTTP traffic to the Kibana application, which is listening on `localhost:5601`. Additionally, it configures Nginx to read the `htpasswd.users` file and require basic authentication.

Note that if you followed the prerequisite Nginx tutorial through to the end, you may have already created this file and populated it with some content. In that case, delete all the existing content in the file before adding the following:

/etc/nginx/sites-available/example.com

```
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;
    }
}
```

When you're finished, save and close the file.

Next, enable the new configuration by creating a symbolic link to the `sites-enabled` directory. If you already created a server block file with the same name in the Nginx prerequisite, you do not need to run this command:

```
sudo ln -s /etc/nginx/sites-available/example.com /etc/nginx/sites-enabled/example.com
```

Then check the configuration for syntax errors:

```
sudo nginx -t
```

If any errors are reported in your output, go back and double check that the content you placed in your configuration file was added correctly. Once you see `syntax is ok` in the output, go ahead and restart the Nginx service:

```
sudo systemctl restart nginx
```

If you followed the initial server setup guide, you should have a UFW firewall enabled. To allow connections to Nginx, we can adjust the rules by typing:

```
sudo ufw allow 'Nginx Full'
```

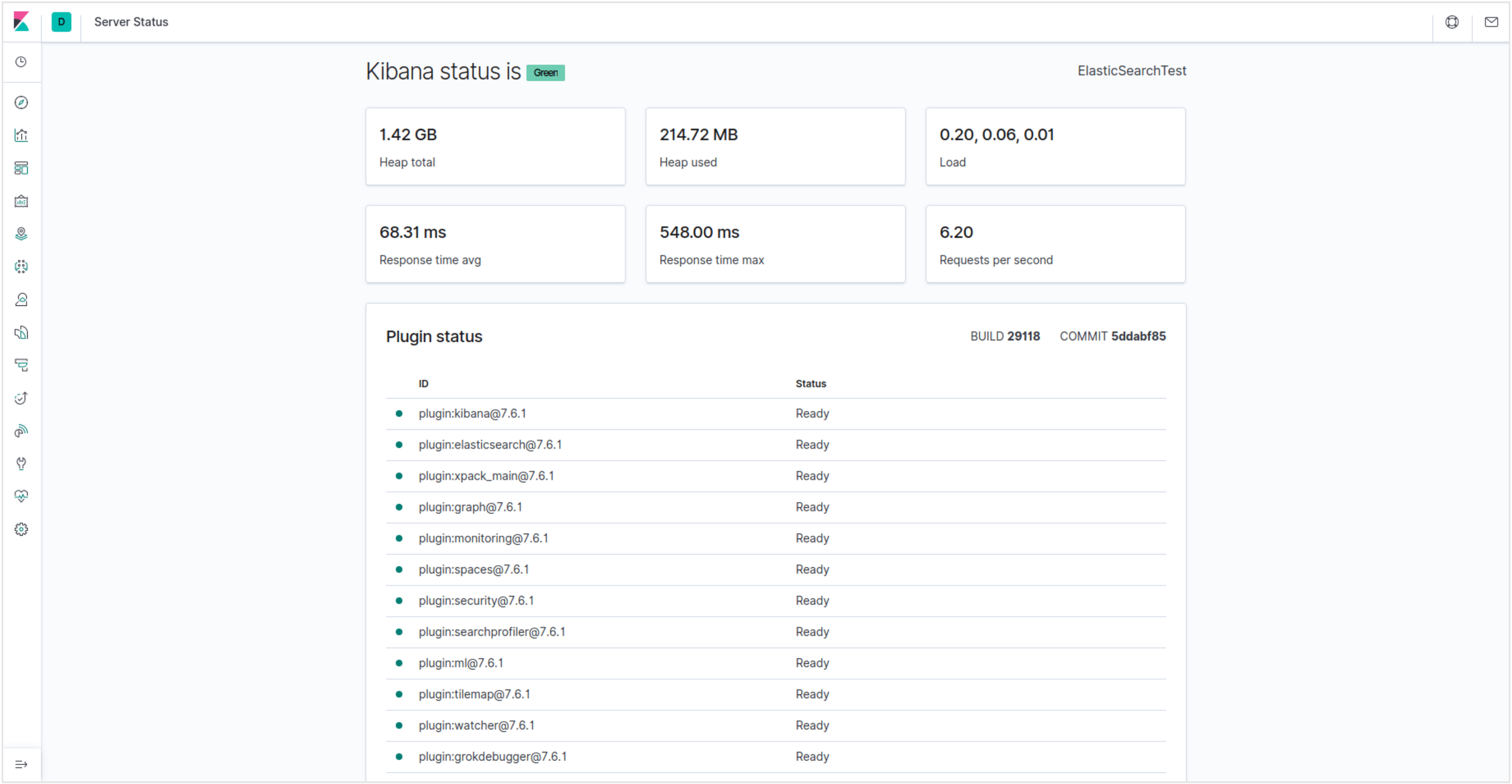
Note: If you followed the prerequisite Nginx tutorial, you may have created a UFW rule allowing the `Nginx HTTP` profile through the firewall. Because the `Nginx Full` profile allows both HTTP and HTTPS traffic through the firewall, you can safely delete the rule you created in the prerequisite tutorial. Do so with the following command:

```
sudo ufw delete allow 'Nginx HTTP'
```

Kibana is now accessible via your FQDN or the public IP address of your Elastic Stack server. You can check the Kibana server's status page by navigating to the following address and entering your login credentials when prompted:

```
http://your_server_ip/status
```

This status page displays information about the server's resource usage and lists the installed plugins.



Note: As mentioned in the Prerequisites section, it is recommended that you enable SSL/TLS on your server. You can follow this tutorial now to obtain a free SSL certificate for Nginx on Ubuntu 18.04. After obtaining your SSL/TLS certificates, you can come back and complete this tutorial.

Now that the Kibana dashboard is configured, let's install the next component: Logstash.

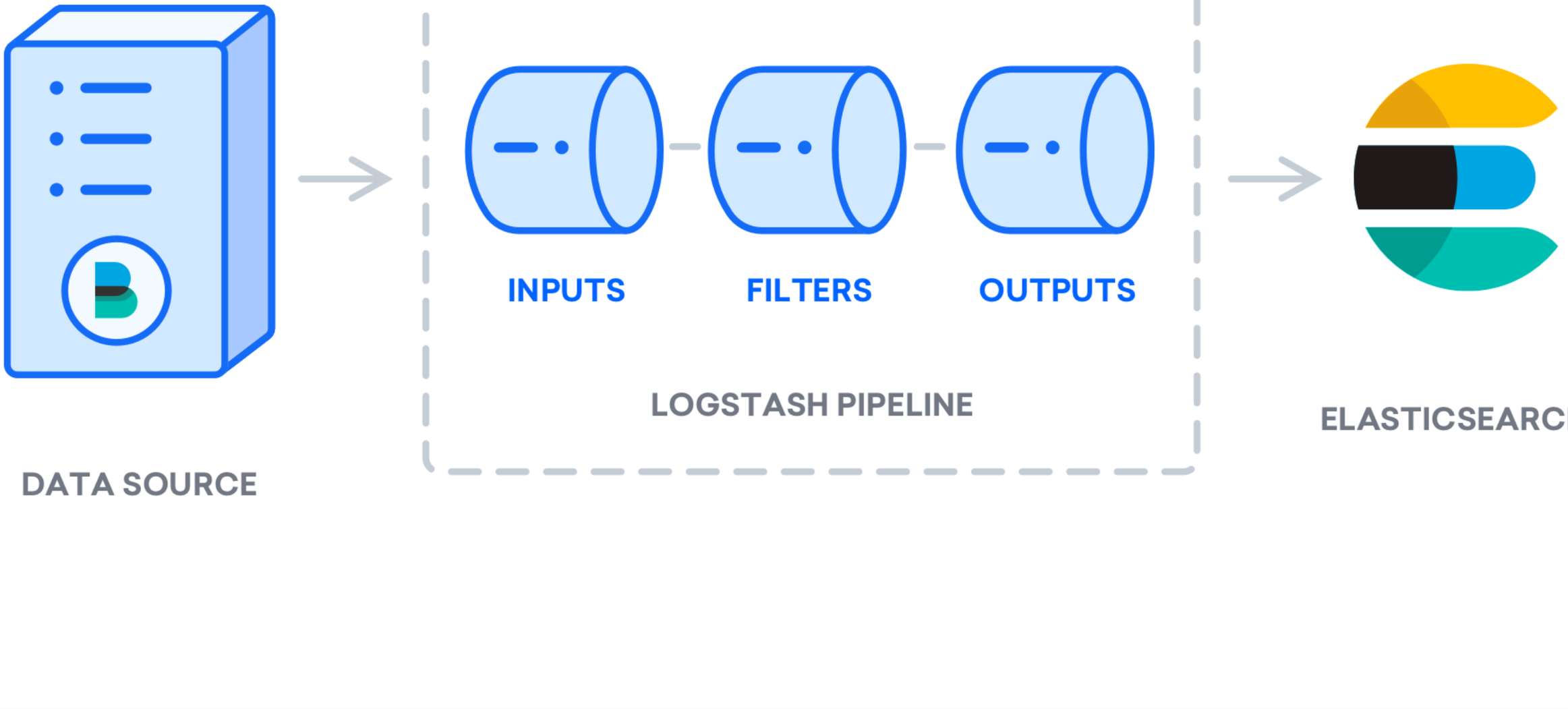
Step 3 — Installing and Configuring Logstash

Although it’s possible for Beats to send data directly to the Elasticsearch database, we recommend using Logstash to process the data. This will allow you to collect data from different sources, transform it into a common format, and export it to another database.

Install Logstash with this command:

```
sudo apt install logstash
```

After installing Logstash, you can move on to configuring it. Logstash’s configuration files are written in the JSON format and reside in the `/etc/logstash/conf.d` directory. As you configure it, it’s helpful to think of Logstash as a pipeline which takes in data at one end, processes it in one way or another, and sends it out to its destination (in this case, the destination being Elasticsearch). A Logstash pipeline has two required elements, `input` and `output`, and one optional element, `filter`. The input plugins consume data from a source, the filter plugins process the data, and the output plugins write the data to a destination.



Create a configuration file called `02-beats-input.conf` where you will set up your Filebeat input:

```
sudo nano /etc/logstash/conf.d/02-beats-input.conf
```

Insert the following `input` configuration. This specifies a `beats` input that will listen on TCP port `5044`.

/etc/logstash/conf.d/02-beats-input.conf

```
input {
  beats {
    port => 5044
  }
}
```

Save and close the file. Next, create a configuration file called `10-syslog-filter.conf`, where we will add a filter for system logs, also known as *syslogs*:

```
sudo nano /etc/logstash/conf.d/10-syslog-filter.conf
```

Insert the following syslog filter configuration. This example system logs configuration was taken from official Elastic documentation. This filter is used to parse incoming system logs to make them structured and usable by the predefined Kibana dashboards:

/etc/logstash/conf.d/10-syslog-filter.conf

```
filter {
  if [fileset][module] == "system" {
    if [fileset][name] == "auth" {
      grok {
        match => { "message" => ["%{SYSLOGTIMESTAMP:[system][auth][timestamp]} %{SYSLOGHOST:[system][auth][hostname]} sshd(?:\[%{POSINT:[system][auth][pid]}\])?: %{DATA:[system][auth][ssh][eve
                  "%{SYSLOGTIMESTAMP:[system][auth][timestamp]} %{SYSLOGHOST:[system][auth][hostname]} sshd(?:\[%{POSINT:[system][auth][pid]}\])?: %{DATA:[system][auth][ssh][event]} user %{DAT
                  "%{SYSLOGTIMESTAMP:[system][auth][timestamp]} %{SYSLOGHOST:[system][auth][hostname]} sshd(?:\[%{POSINT:[system][auth][pid]}\])?: Did not receive identification string from %{
                  "%{SYSLOGTIMESTAMP:[system][auth][timestamp]} %{SYSLOGHOST:[system][auth][hostname]} sudo(?:\[%{POSINT:[system][auth][pid]}\])?: \s*%{DATA:[system][auth][user]} :( %{DATA:[sy
                  "%{SYSLOGTIMESTAMP:[system][auth][timestamp]} %{SYSLOGHOST:[system][auth][hostname]} groupadd(?:\[%{POSINT:[system][auth][pid]}\])?: new group: name=%{DATA:system.auth.groupa
                  "%{SYSLOGTIMESTAMP:[system][auth][timestamp]} %{SYSLOGHOST:[system][auth][hostname]} useradd(?:\[%{POSINT:[system][auth][pid]}\])?: new user: name=%{DATA:[system][auth][user]
                  "%{SYSLOGTIMESTAMP:[system][auth][timestamp]} %{SYSLOGHOST:[system][auth][hostname]} %{DATA:[system][auth][program]}(?:\[%{POSINT:[system][auth][pid]}\])?: %{GREEDYMULTILINE:
      pattern_definitions => {
        "GREEDYMULTILINE"=> "(.|\n)*"
      }
      remove_field => "message"
    }
    date {
      match => [ "[system][auth][timestamp]", "MMM  d HH:mm:ss", "MMM dd HH:mm:ss" ]
    }
    geoip {
      source => "[system][auth][ssh][ip]"
      target => "[system][auth][ssh][geoip]"
    }
  }
  else if [fileset][name] == "syslog" {
    grok {
      match => { "message" => ["%{SYSLOGTIMESTAMP:[system][syslog][timestamp]} %{SYSLOGHOST:[system][syslog][hostname]} %{DATA:[system][syslog][program]}(?:\[%{POSINT:[system][syslog][pid]}\
      pattern_definitions => { "GREEDYMULTILINE" => "(.|\n)*" }
      remove_field => "message"
    }
    date {
      match => [ "[system][syslog][timestamp]", "MMM  d HH:mm:ss", "MMM dd HH:mm:ss" ]
    }
  }
```

```
  }
}
```

Save and close the file when finished.

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

```
sudo nano /etc/logstash/conf.d/30-elasticsearch-output.conf
```

Insert the following `output` configuration. Essentially, this output configures Logstash to store the Beats data in Elasticsearch, which is running at `localhost:9200`, in an index named after the Beat used. The Beat used in this tutorial is Filebeat:

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

```
output {
  elasticsearch {
    hosts => ["localhost:9200"]
    manage_template => false
    index => "%{[@metadata][beat]}-%{[@metadata][version]}-%{+YYYY.MM.dd}"
  }
}
```

Save and close the file.

If you want to add filters for other applications that use the Filebeat input, be sure to name the files so they're sorted between the input and the output configuration, meaning that the file names should begin with a two-digit number between `02` and `30`.

Test your Logstash configuration with this command:

```
sudo -u logstash /usr/share/logstash/bin/logstash --path.settings /etc/logstash -t
```

If there are no syntax errors, your output will display `Configuration OK` after a few seconds. If you don't see this in your output, check for any errors that appear in your output and update your configuration to correct them.

If your configuration test is successful, start and enable Logstash to put the configuration changes into effect:

```
sudo systemctl start logstash
sudo systemctl enable logstash
```

Now that Logstash is running correctly and is fully configured, let's install Filebeat.

Step 4 — Installing and Configuring Filebeat

The Elastic Stack uses several lightweight data shippers called Beats to collect data from various sources and transport them to Logstash or Elasticsearch. Here are the Beats that are currently available from Elastic:

- Filebeat: collects and ships log files.
- Metricbeat: collects metrics from your systems and services.
- Packetbeat: collects and analyzes network data.
- Winlogbeat: collects Windows event logs.
- Auditbeat: collects Linux audit framework data and monitors file integrity.
- Heartbeat: monitors services for their availability with active probing.

In this tutorial we will use Filebeat to forward local logs to our Elastic Stack.

Install Filebeat using `apt`:

```
sudo apt install filebeat
```

Next, configure Filebeat to connect to Logstash. Here, we will modify the example configuration file that comes with Filebeat.

Open the Filebeat configuration file:

```
sudo nano /etc/filebeat/filebeat.yml
```

Note: As with Elasticsearch, Filebeat's configuration file is in YAML format. This means that proper indentation is crucial, so be sure to use the same number of spaces that are indicated in these instructions.

Filebeat supports numerous outputs, but you'll usually only send events directly to Elasticsearch or to Logstash for additional processing. In this tutorial, we'll use Logstash to perform additional processing on the data collected by Filebeat. Filebeat will not need to send any data directly to Elasticsearch, so let's disable that output. To do so, find the `output.elasticsearch` section and comment out the following lines by preceding them with a `#`:

/etc/filebeat/filebeat.yml

```
...
#output.elasticsearch:
  # Array of hosts to connect to.
  #hosts: ["localhost:9200"]
...
```

Then, configure the `output.logstash` section. Uncomment the lines `output.logstash:` and `hosts: ["localhost:5044"]` by removing the `#`. This will configure Filebeat to connect to Logstash on your Elastic Stack server at port `5044`, the port for which we specified a Logstash input earlier:

/etc/filebeat/filebeat.yml

```
output.logstash:
  # The Logstash hosts
  hosts: ["localhost:5044"]
```

Save and close the file.

The functionality of Filebeat can be extended with Filebeat modules. In this tutorial we will use the system module, which collects and parses logs created by the system logging service of common Linux distributions.

Let’s enable it:

```
sudo filebeat modules enable system
```

You can see a list of enabled and disabled modules by running:

```
sudo filebeat modules list
```

You will see a list similar to the following:

```
Output
Enabled:
system
```