Project 9 - Centralized Syslog Solution

Goal:

The goal for this project is to create a centralized syslog server that will allow you to store, graph, and search through the syslog messages from multiple servers. To do this, you'll be deploying the ELK stack. The components of the ELK stack are Elasticsearch, Logstash, and Kibana. Finally, you'll configure servers to send their messages to this new system.

Instructions:

Create a Virtual Machine

First, start a command line session on your local machine. Next, move into the working folder you created for this course.

```
cd linuxclass
```

Initialize the vagrant project using the usual process of creating a directory, changing into that directory, and running "vagrant init". We'll name this vagrant project "syslog".

```
mkdir syslog
cd syslog
vagrant init jasonc/centos8
```

Configure the Virtual Machine

Edit the Vagrantfile and set the hostname of the virtual machine to "syslog". Also, assign the IP address of 10.23.45.50 to the machine.

```
config.vm.hostname = "syslog"
config.vm.network "private_network", ip: "10.23.45.50"
```

Start the Virtual Machine

Now you're ready to start the VM and connect to it.

```
vagrant up
vagrant ssh
```

Install Elasticsearch

We'll be using Elasticsearch to store the syslog messages. Let's install Elasticsearch from an RPM.

```
sudo dnf install -y \
http://mirror.linuxtrainingacademy.com/elasticsearch/elasticsearch-7.9.2-x86_64.rpm
```

Internet download location:
https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.9.2-x86_64.rpm

We need to give our Elasticsearch cluster a name. Also, let's use our hostname for the node name. Edit the elasticsearch.yml file.

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

Append the following contents to the bottom of the file and save it.

```
cluster.name: syslog
node.name: syslog
```

Now we can start and enable the Elasticsearch service.

```
sudo systemctl start elasticsearch.service
sudo systemctl enable elasticsearch.service
```

Give Elasticsearch a minute or two to start. Then connect to its port of 9200 over HTTP using curl.

```
curl http://localhost:9200
```

Install Logstash

Let's install Logstash, so we have a way of receiving logs from systems and sending them to Elasticsearch. Logstash is a Java application, so we'll need to install Java first.

```
sudo dnf install -y java-1.8.0-openjdk
```

Now we can install Logstash.

```
sudo dnf install -y http://mirror.linuxtrainingacademy.com/logstash/logstash-7.9.2.rpm
```

Internet download location:
https://artifacts.elastic.co/downloads/logstash/logstash-7.9.2.rpm

Let's create the Logstash configuration. We'll place it in a file named `syslog.conf` in the `/etc/logstash/conf.d` directory.

```
sudo nano /etc/logstash/conf.d/syslog.conf
```

[This space intentionally left blank. Instructions continue on the following page.]

Paste the following contents into the file and save it. Be sure that all the characters pasted correctly.

```
input {
  syslog {
    type => syslog
    port => 5141
  }
}

filter {
  if [type] == "syslog" {
    grok {
      match => { "message" => "Accepted %{WORD:auth_method} for %{USER:username}
from %{IP:src_ip} port %{INT:src_port} ssh2" }
      add_tag => "ssh_successful_login"
    }
    grok {
      match => { "message" => "Failed %{WORD:auth_method} for %{USER:username} from
%{IP:src_ip} port %{INT:src_port} ssh2" }
      add_tag => "ssh_failed_login"
    }
    grok {
      match => { "message" => "Invalid user %{USER:username} from %{IP:src_ip}" }
      add_tag => "ssh_failed_login"
    }
  }
  geoip {
    source => "src_ip"
  }
}

output {
  elasticsearch { }
}
```

IMPORTANT NOTE: There are configuration lines start with "match => {" and end with "}". Everything between "match => {" and the closing curly brace "}" is on the same line. Due to the space limitation in this file, that content may appear to be on multiple lines, but it is not. If you include lines breaks in the match messages, they will never match because the actual logs do not include line breaks.

Here is the configuration again, but in a very small font showing the proper spacing and line breaks.

```
input {
  syslog {
    type => syslog
    port => 5141
  }
}

filter {
  if [type] == "syslog" {
    grok {
      match => { "message" => "Accepted %{WORD:auth_method} for %{USER:username} from %{IP:src_ip} port %{INT:src_port} ssh2" }
      add_tag => "ssh_successful_login"
    }
    grok {
      match => { "message" => "Failed %{WORD:auth_method} for %{USER:username} from %{IP:src_ip} port %{INT:src_port} ssh2" }
      add_tag => "ssh_failed_login"
    }
    grok {
      match => { "message" => "Invalid user %{USER:username} from %{IP:src_ip}" }
      add_tag => "ssh_failed_login"
    }
  }
  geoip {
    source => "src_ip"
  }
}

output {
  elasticsearch { }
}
```

The input section of the configuration causes Logstash to listen for syslog messages on port 5141. The filter section of the configuration allows Logstash to perform a bit of processing on the messages it receives that match the given patterns. For example, it extracts the authentication method, the username, the source IP address, and source port for ssh connection attempts. It also tags the messages with "ssh_successful_login" or "ssh_failed_login". This will make searching for data based on username, IP address, failed ssh login attempts, etc, quick and efficient. The output section tells logstash to store the messages into the Elasticsearch instance we just created.

Now we can start and enable the logstash service.

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

Logstash can take several seconds to start. You can confirm it started by looking at its log file.

```
cat /var/log/logstash/logstash-plain.log
```

Forwarding Syslog Messages to Logstash

Next, let's configure our local system to forward its syslog messages to Logstash. To do that, let's create a `logstash.conf` file in the `/etc/rsyslog.d` directory.

```
sudo nano /etc/rsyslog.d/logstash.conf
```

Place the following contents in the file and save the file.

```
*.* @10.23.45.50:5141
```

This will cause rsyslog to send a copy of every syslog message to Logstash. Restart rsyslog to enable this configuration.

```
sudo systemctl restart rsyslog
```

Check Elasticsearch

Logstash should now be receiving syslog messages from the local system and storing them in Elasticsearch. Let's look at the Elasticsearch indices. You should see an index for Logstash.

```
curl http://localhost:9200/_cat/indices?v
```

Over time, Logstash will create more indices in Elasticsearch. You'll be able to search across those indices without a problem with Kibana, which you will be installing in a minute.

Elasticsearch Troubleshooting

Elasticsearch can consume a lot of memory. If you are experiences issues, increase the amount of memory allocated to the virtual machine to at least 3 GB if possible. You can do this with a `config.vm.provider` block of configuration. Update your Vagrantfile to look like the following:

```
Vagrant.configure("2") do |config|
  config.vm.box = "jasonc/centos8"
  config.vm.hostname = "syslog"
  config.vm.network "private_network", ip: "10.23.45.50"
  config.vm.provider "virtualbox" do |vb|
    vb.memory = "3072"
  end
end
```

Creating a Cluster - For Informational Purposes Only - (OPTIONAL)

I also want to point out that if you see that the health of the index is yellow, it's because there is only one copy of the data for that index, and it's stored on this host. For our purposes, that is fine. For this project, we are going to operate with one copy of our data.

Please skip this clustering section and continue below if this is the first time you are working through this project. Clustering is an advanced procedure. It is not required to complete the project and this course. The following is primarily informational and what you might do in a production environment where you want to have multiple copies of the data.

If you want to eliminate single points of failure for your Elasticsearch cluster in a production environment, you can add another node. To do that, add the following line of configuration on your first host, so it can communicate with the outside network.

```
network.host: [10.23.45.50, _local_]
```

Next, add a line telling Elasticsearch where to look for additional nodes. You'll want to list all the IP addresses of all the nodes that will be in the Elasticsearch cluster.

```
discovery.zen.ping.unicast.hosts: ["10.23.45.50", "10.23.45.51"]
```

Remember to restart Elasticsearch after making configuration changes. The full elasticsearch.yml file will read:

```
    cluster.name: syslog
    node.name: syslog
    network.host: [10.23.45.50, _local_]
    discovery.zen.ping.unicast.hosts: ["10.23.45.50", "10.23.45.51"]
```

Next, install Elasticsearch on another server in the exact same manner as described above. Make sure to set the cluster name to be the same as it is on the first server. This new second server will automatically discover and join the cluster as long as it has the same cluster.name as the first node, and it finds a node in the discovery.zen.ping.unicast.hosts list.

Example configuration for an additional elasticsearch cluster member named syslog2 with an IP address of 10.23.45.51:

```
    cluster.name: syslog
    node.name: syslog2
    network.host: [10.23.45.51, _local_]
    discovery.zen.ping.unicast.hosts: ["10.23.45.50", "10.23.45.51"]
```

Install Kibana

Now that Elasticsearch has some data for us to search, we'll install Kibana.

```
sudo dnf install -y \
http://mirror.linuxtrainingacademy.com/kibana/kibana-7.9.2-x86_64.rpm
```

Internet download location:
https://artifacts.elastic.co/downloads/kibana/kibana-7.9.2-x86_64.rpm

By default, Kibana only listens on localhost. This means that you would not be able to connect to Kibana from outside the host. Let's change that so we can access Kibana using the VM's IP address. Open up the Kibana configuration file for editing.

```
sudo nano /etc/kibana/kibana.yml
```

Add this line of configuration.

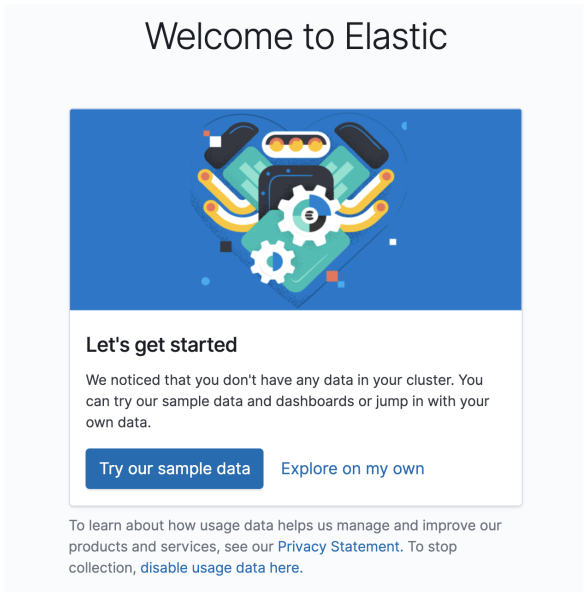
```
server.host: "10.23.45.50"
```

Now that we've configured Kibana, it's time to start it. We'll also enable it so that it starts on boot as well.

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

Once Kibana has been started, open a web browser on your local machine and visit this address: http://10.23.45.50:5601. Kibana operates on port 5601, so that's the port you'll connect to. It can take Kibana a few minutes to start, so please be patient.

You'll be presented with a welcome screen. Click on the "Explore on my own" link.

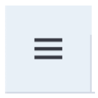

Welcome to Elastic

Let's get started

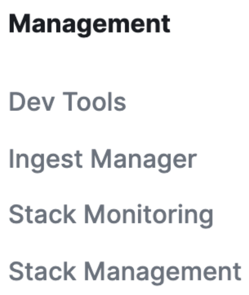
We noticed that you don't have any data in your cluster. You can try our sample data and dashboards or jump in with your own data.

To learn about how usage data helps us manage and improve our products and services, see our Privacy Statement. To stop collection, disable usage data here.

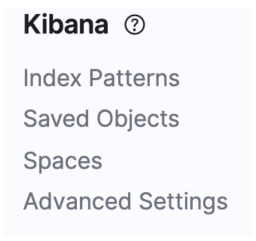
In the upper-left corner click on the menu icon. (Some people call the icon a Hamburger Menu icon.)



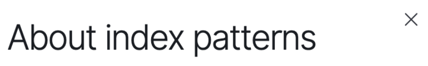
Scroll down. Under the "Management" section, click the "Stack Management" link.



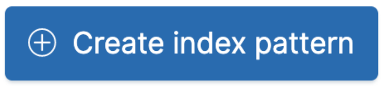
Scroll down. Under the "Kibana" section, click the "Index Patterns" link.



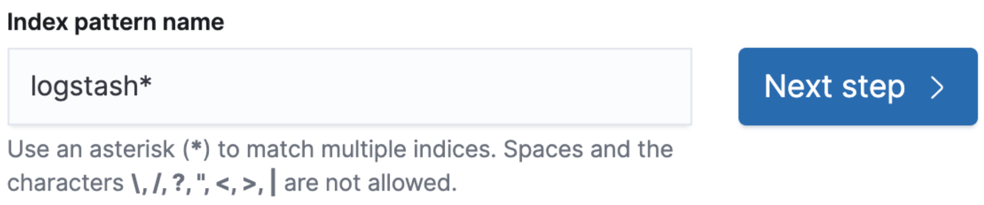
There will be a pop-up display labeled "About Index Patterns" in the right-hand side of your screen. Click the "X" to close it.



Now click on the "Create index pattern" button.

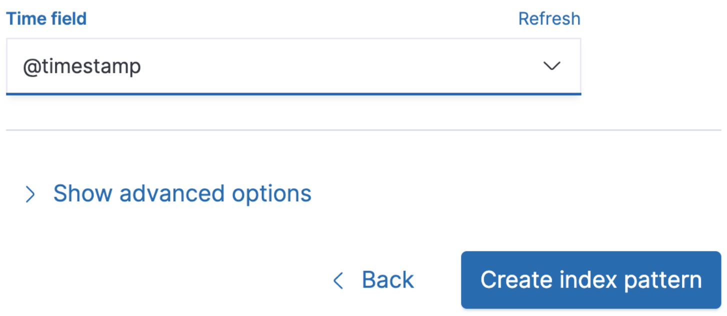


In the "Index pattern name" field, enter "logstash*" and then click the "Next Step" button. This tells Kibana to use any indices in Elasticsearch that start with "logstash".

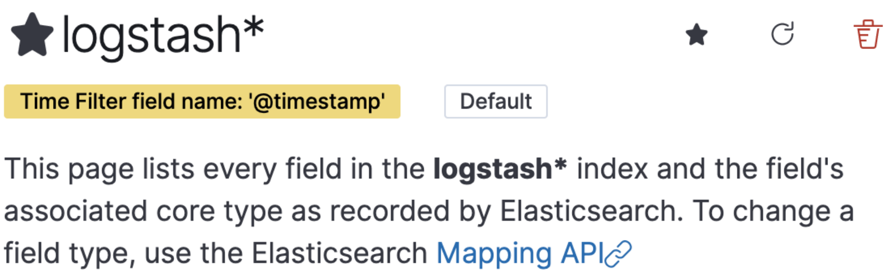


Use an asterisk (*) to match multiple indices. Spaces and the characters \, /, ?, ", <, >, | are not allowed.

In the "Time Field" dropdown menu, select "@timestamp and then click the "Create index pattern" button.

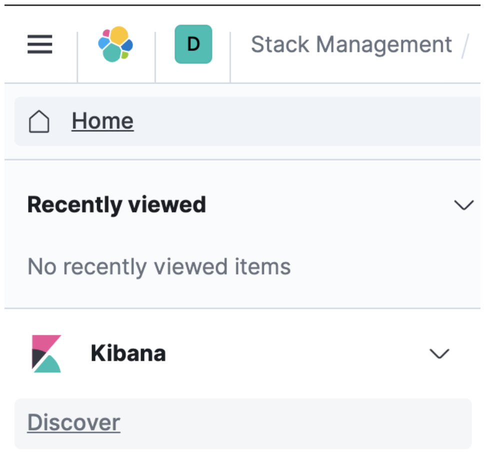


You'll be brought to a screen that shows information about the index pattern that you just created.



This page lists every field in the **logstash*** index and the field's associated core type as recorded by Elasticsearch. To change a field type, use the Elasticsearch Mapping API

Now you can start searching for log messages by clicking on the "Discover" link under "Kibana" in the right-hand menu.



Using Kibana

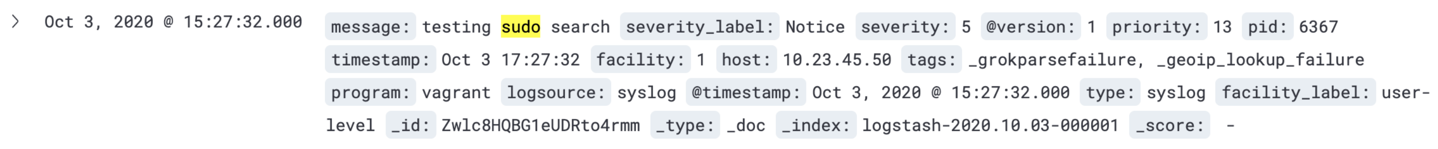
Return to your command line session. Let's use the logger command to send a message to syslog. Of course, this message will be sent to Logstash as well, and will ultimately be stored in Elasticsearch. Here is the command:

```
logger "testing sudo search"
```

This will send a syslog message of "testing sudo search". You can see it in /var/log/messages.

```
sudo grep testing /var/log/messages
```

Now, return to Kibana (http://10.23.45.50:5601/app/discover#/) and perform a search for "sudo". This returns all results that have the text "sudo" anywhere in its associated record.   Here's an example that shows the word "sudo" in the "message" field of the record -- the one we created with the logger command.

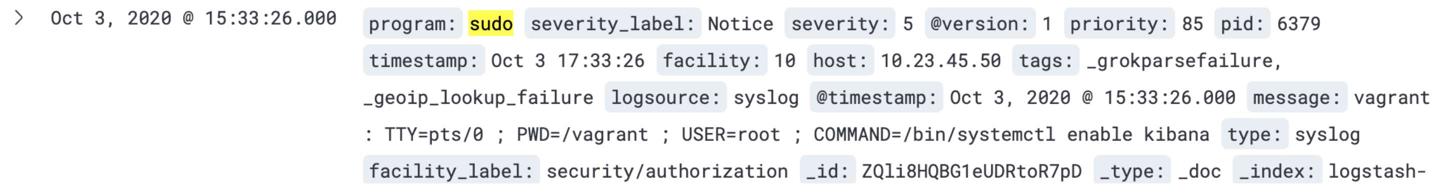


```
>  Oct 3, 2020 @ 15:27:32.000   message: testing sudo search severity_label: Notice severity: 5 @version: 1 priority: 13 pid: 6367
                                timestamp: Oct 3 17:27:32 facility: 1 host: 10.23.45.50 tags: _grokparsefailure, _geoip_lookup_failure
                                program: vagrant logsource: syslog @timestamp: Oct 3, 2020 @ 15:27:32.000 type: syslog facility_label: user-
                                level _id: Zwlc8HQBG1eUDRto4rmm _type: _doc _index: logstash-2020.10.03-000001 _score: -
```

You'll notice the various parts of the record. There is the message, the timestamp, the type, host, program, etc. You can use each one of these fields to narrow your search results. For example, let's search for sudo, but let's only display results that come from the sudo program. To do that, use the search string "program:sudo".

Now we'll only get syslog messages generated by the sudo command. You will not find the message generated by the logger command used earlier, even though the word "sudo" was in the message. (For that, you could use this search: "message:sudo")

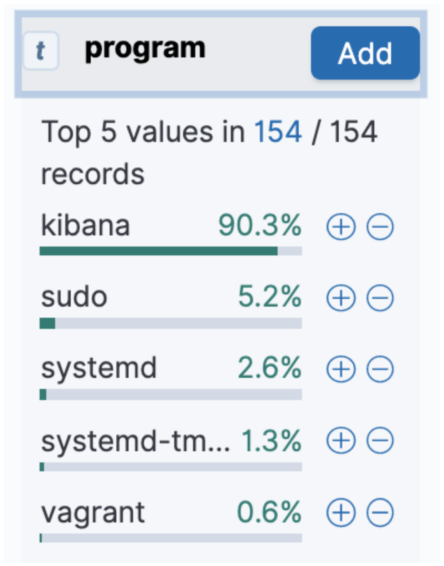
Here is an example result that matches "program:sudo".



```
>  Oct 3, 2020 @ 15:33:26.000   program: sudo severity_label: Notice severity: 5 @version: 1 priority: 85 pid: 6379
                                timestamp: Oct 3 17:33:26 facility: 10 host: 10.23.45.50 tags: _grokparsefailure,
                                _geoip_lookup_failure logsource: syslog @timestamp: Oct 3, 2020 @ 15:33:26.000 message: vagrant
                                : TTY=pts/0 ; PWD=/vagrant ; USER=root ; COMMAND=/bin/systemctl enable kibana type: syslog
                                facility_label: security/authorization _id: ZQli8HQBG1eUDRtoR7pD _type: _doc _index: logstash-
```

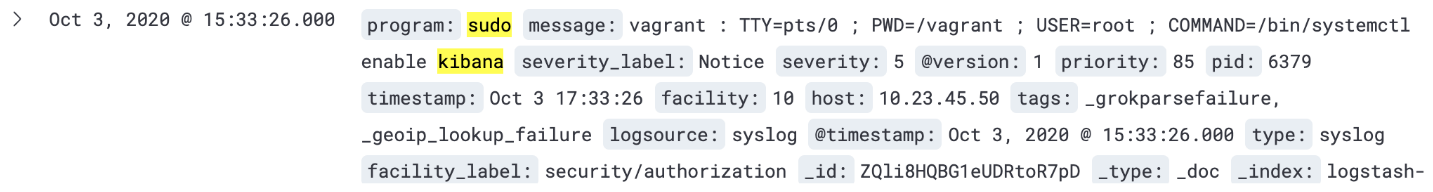
If you want to display all matches, simply enter in "*" in the search bar and hit enter. Do that now. (Alternatively, you could click the "New" link in the top left corner of the screen.)

Now, you can explore the data that is available in the fields by clicking on them on the left side of your screen. For example, if you click on "program" you'll see data that matches that field.



Let's do another search. This time, let's look for syslog messages generated by the sudo command that also contain the keyword "kibana." To do that, we'll use "AND" in our search. If you don't include "AND", Kibana will return results that match either of the conditions.

Here is the search: "program:sudo AND kibana". This is an example match that shows where the vagrant user ran "systemctl enable kibana".



```
> Oct 3, 2020 @ 15:33:26.000   program: sudo message: vagrant : TTY=pts/0 ; PWD=/vagrant ; USER=root ; COMMAND=/bin/systemctl
                               enable kibana severity_label: Notice severity: 5 @version: 1 priority: 85 pid: 6379
                               timestamp: Oct 3 17:33:26 facility: 10 host: 10.23.45.50 tags: _grokparsefailure,
                               _geoip_lookup_failure logsource: syslog @timestamp: Oct 3, 2020 @ 15:33:26.000 type: syslog
                               facility_label: security/authorization _id: ZQli8HQBG1eUDRtoR7pD _type: _doc _index: logstash-
```

Now let's create a graph. First, let's get some data to graph. Return to the command line and log out of the system and back in again a few time. This will create log entries for each of your connections.

```
exit
vagrant ssh
exit
vagrant ssh
exit
vagrant ssh
```

Now return to Kibana (http://10.23.45.50:5601). Click on the menu icon in the top-left corner. Under "Kibana" click the "Visualize" link. Now click on the "Create new visualization" button.

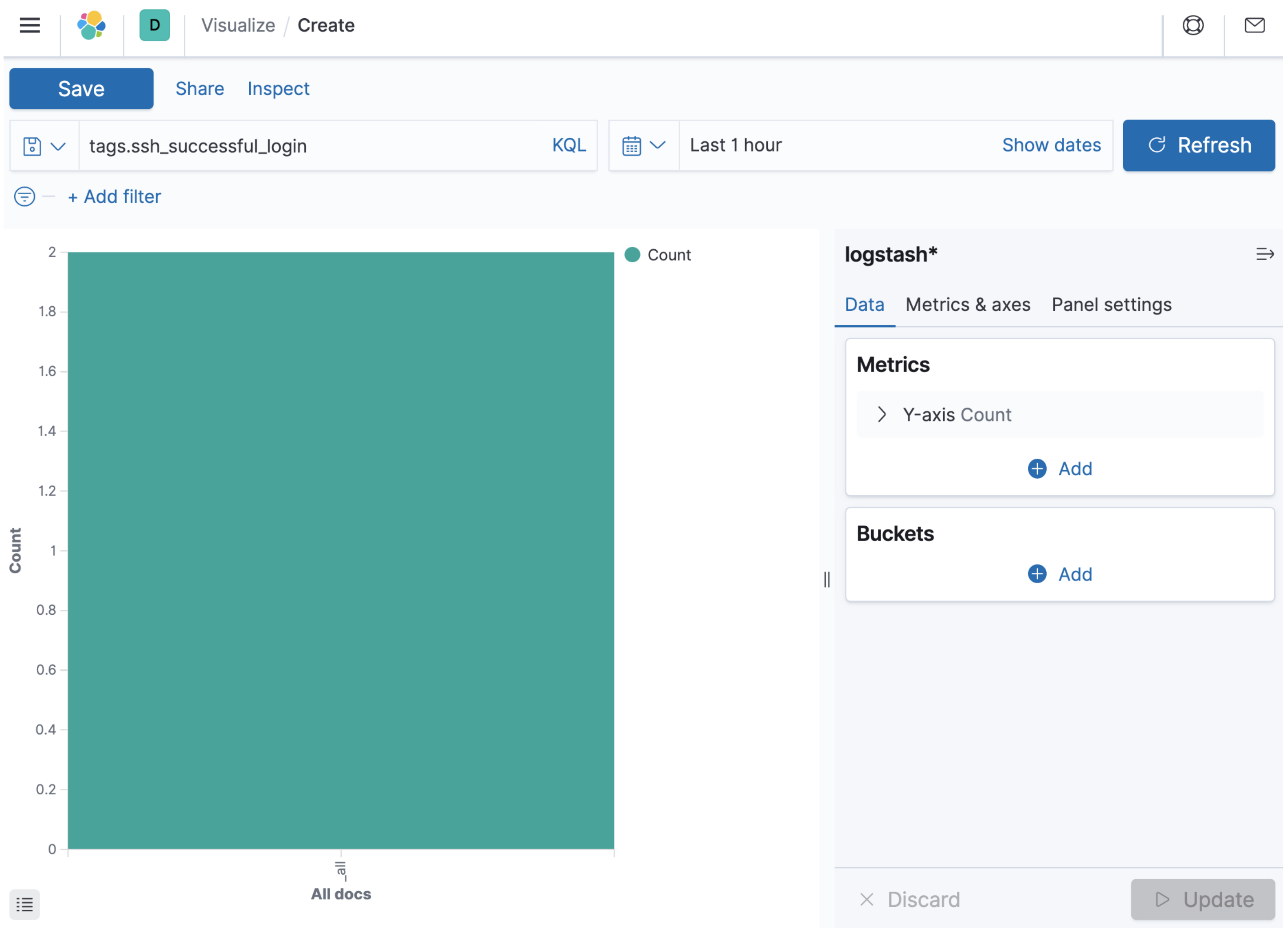
Scroll down and click "Vertical Bar".

Vertical Bar

Next, click on "logstash*".

logstash*

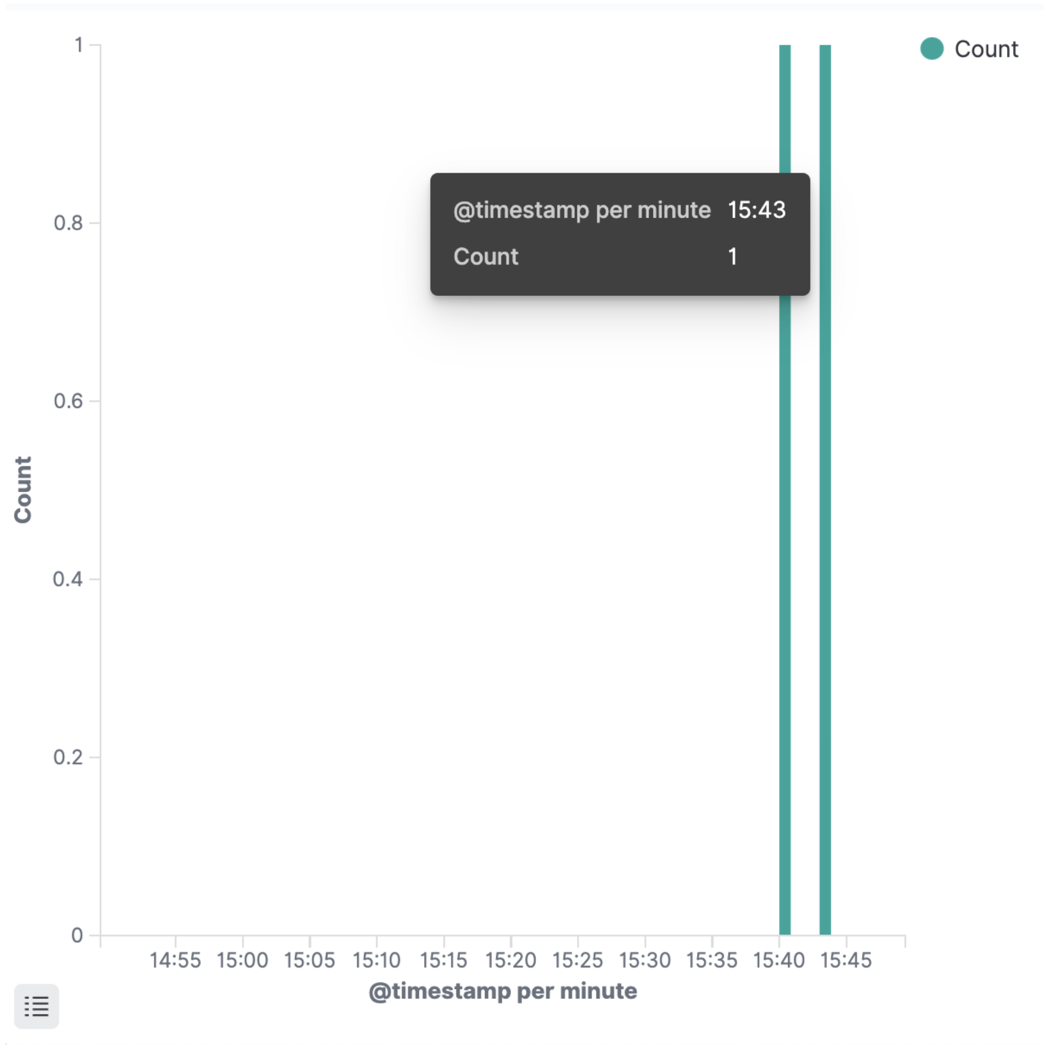
In the search bar, enter in "tags:ssh_successful_login" and click on the "Refresh" button. This will get more interesting in just a moment.



tags.ssh_successful_login

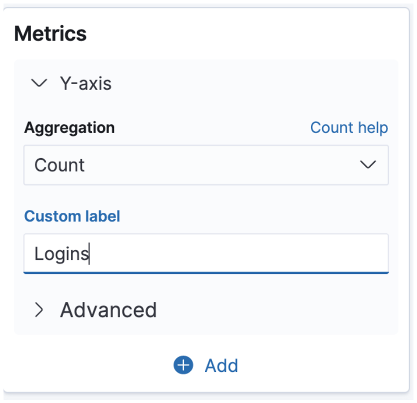
Last 1 hour

Under "Buckets", click on the "+ Add" link. The click on "X-Axis". Under aggregation, select "Date Histogram". To apply the changes, click the "Update" button at the bottom-right of your screen. Now you'll see that one big bar break into smaller bars. If you hover over the bar, it will tell you the number of times the search occurs during that time period. This graph shows that most recently, 1 login was detected.

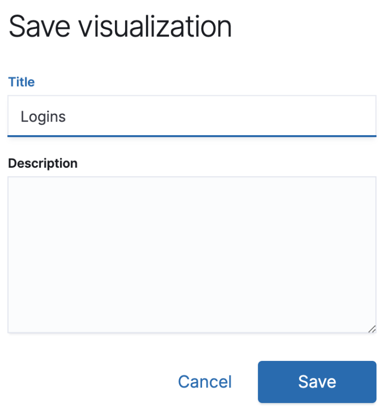


What we've done is visually represented the number of times someone logs into the system over a given period of time. You can use this technique to detect failed logins or any other thing you can imagine to search for in your data.

Let's pretty this graph up a bit. Under the "Metrics" Section, click on "Y-Axis". Now supply a custom label of "Logins". To apply this change, click the "Update" button in the bottom-right of your screen.

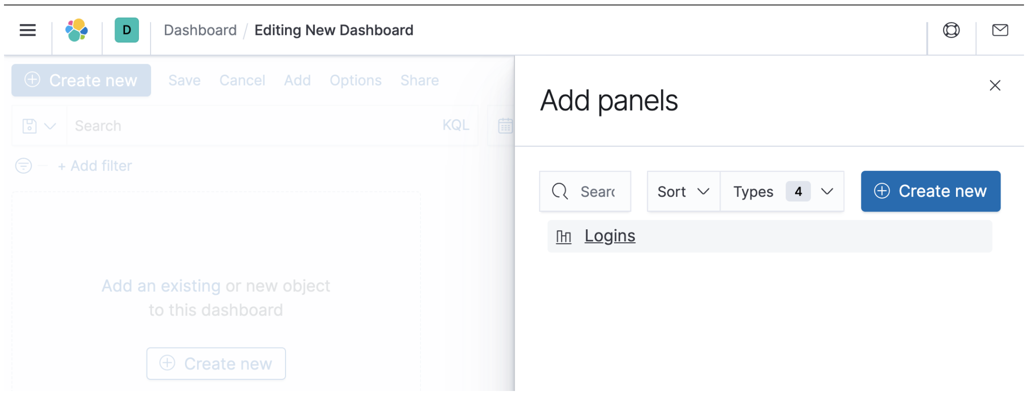


Finally, click the save icon in the top-left and give this graph a Title of "Logins". Click "save" to save.



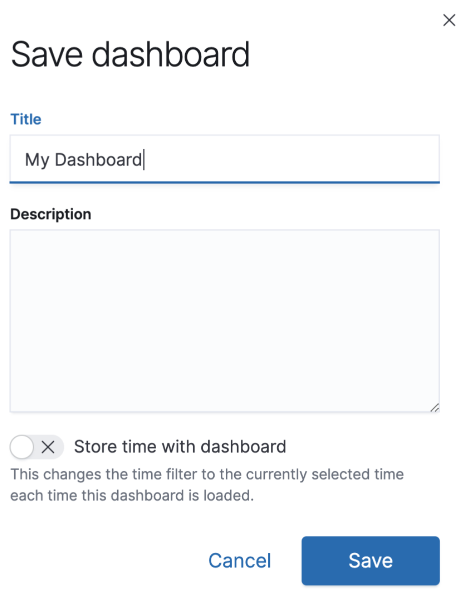
Let's add the visualization we just created to a dashboard. Click on the menu icon in the top-left corner. Under "Kibana" click the "Dashboard" link. Now click on the "Create new dashboard" button.

Click on the "Add an existing object" link. Now click on Logins. Next, click on the "X" to close the pop-up window.



You can stretch the logins graph to fit the width of your screen, or make it any size for that matter. Adjust the graph to whatever size you desire.

Next, click on the "save" link at the top of your screen and give this dashboard a name like "My Dashboard" and save it.


Save dashboard

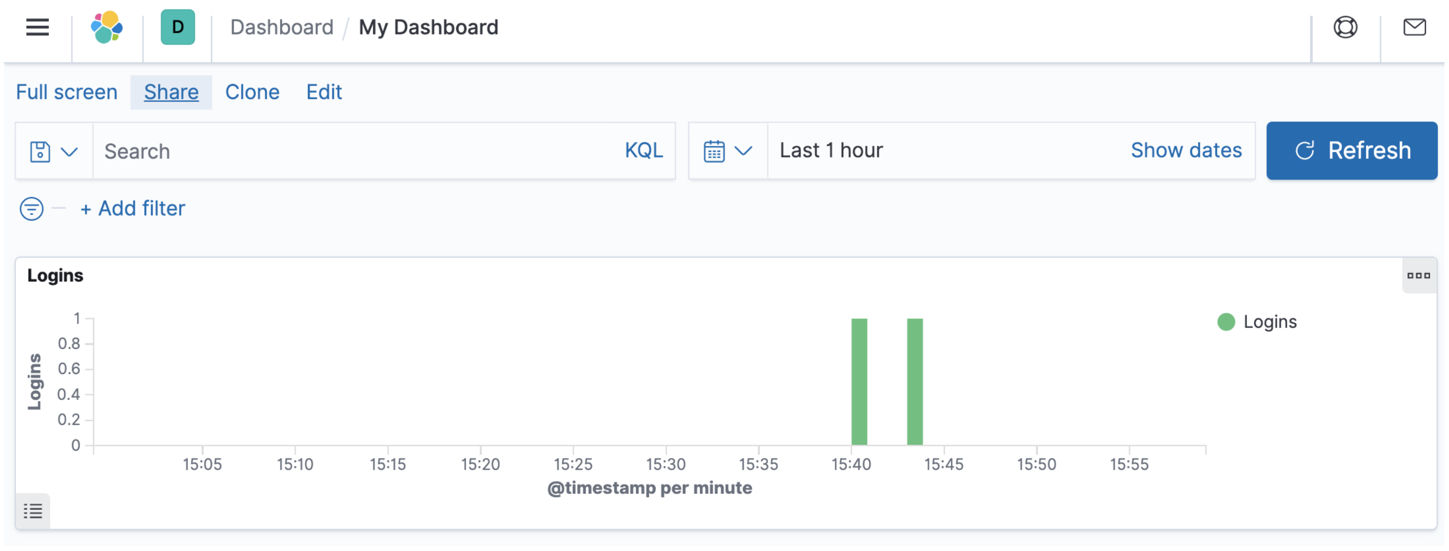
Title: My Dashboard

Description:

Store time with dashboard: off

This changes the time filter to the currently selected time each time this dashboard is loaded.

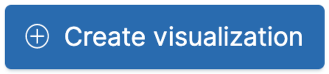
Here's what your dashboard might look like now.



From here you would repeat the process of discovering interesting log messages, making graphs that represent that data in a visually meaningful way, and finally collecting those graphs into a dashboard for a given purpose. For example, you could create a security dashboard that would include graphs for failed logins, password changes, sudo denied messages, and more.

Another Visualization

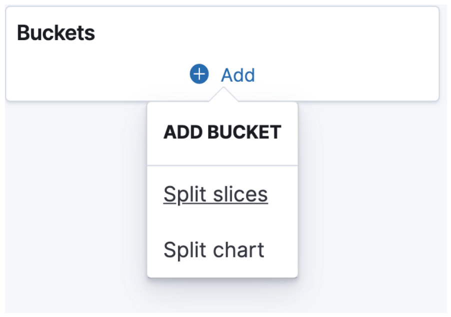
Let's determine what programs are generating the majority of our log messages. Click on the menu icon in the top-left corner. Under "Kibana" click the "Visualize" link. Now in the top-left corner, click the "Visualize" link. Now click on the "Create visualization" button.



Click on the "Pie Chart" icon.

Next, click on "logstash*".

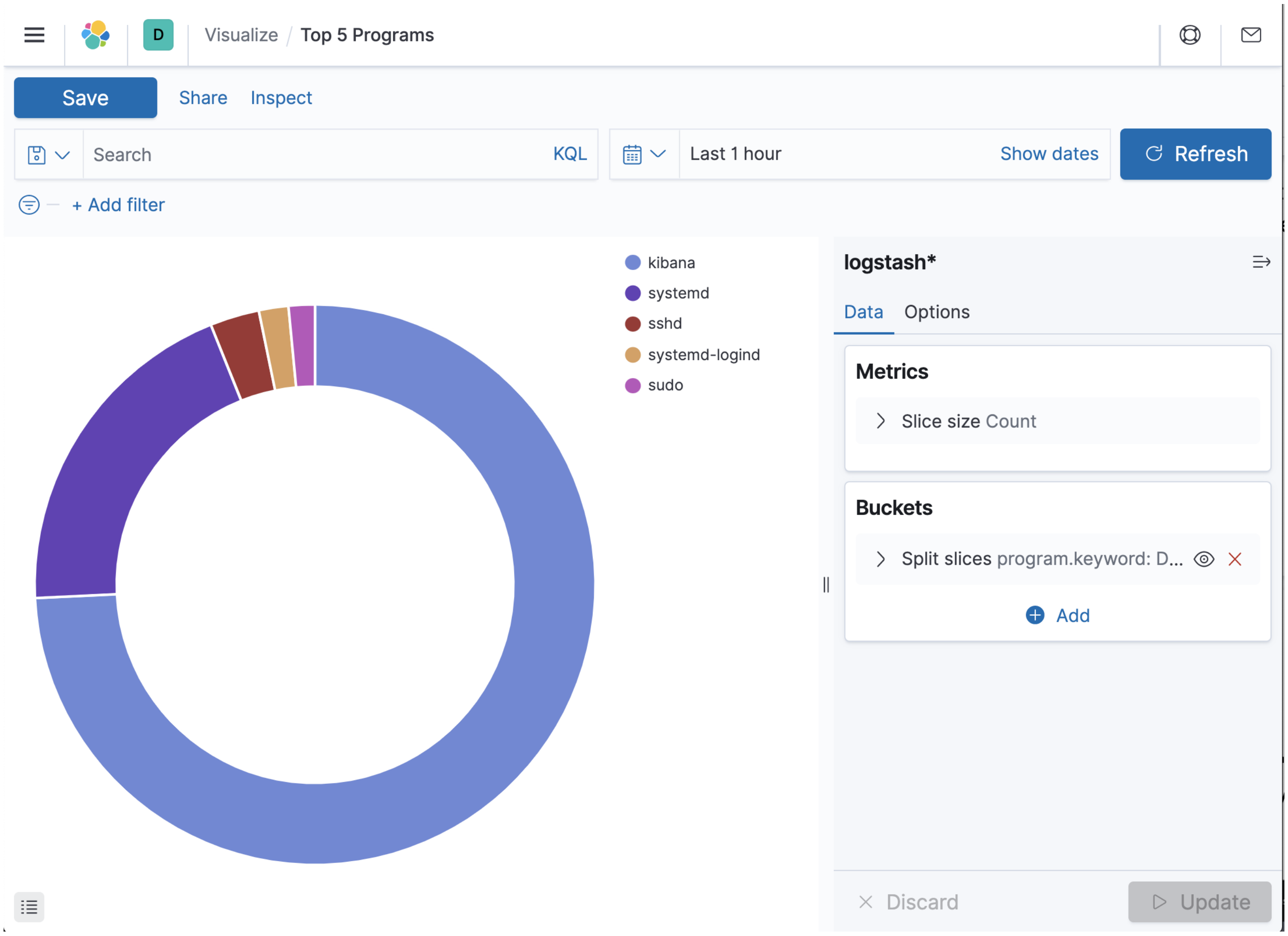
For this visualization, you don't need to filter the search, so leave the search field blank. Click the "+ Add" link under "Buckets" and click on "Split Slices".



Click the Aggregation drop-down and select "Terms", click the Field drop-down and select "program.keyword", then click the Size field and enter "5". Now click the "Update" button.

Save this visualization by clicking on the "Save" button. Give it a title of "Top 5 Programs". (Note, if only 4 programs are reporting logs during this time frame, the graph will show 4. If there are more than 5, then the graph will show the top 5.)

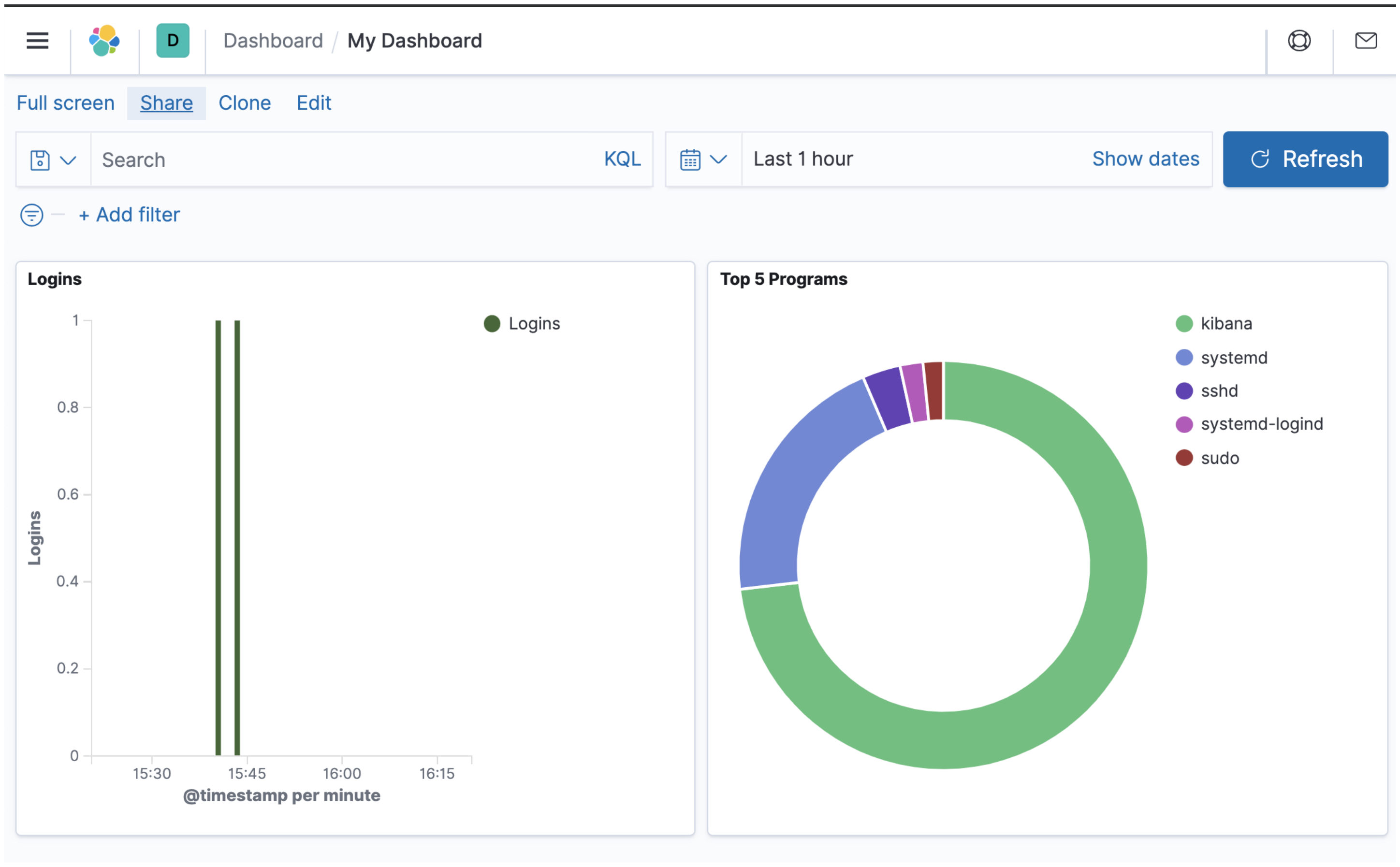
Your graph will now look something like this:



Now you can add it to your dashboard if you'd like. Click on the menu icon in the top-left corner. Under "Kibana" click the "Dashboard" link. Click on the "My Dashboard" link to display your dashboard.

Now, click the "Edit" link at the top of your screen. Now click the "Add" link at the top of your screen. Finally, click on the "Top 5 Programs" to add that graph to the dashboard. Click the "X" in the "Add Panels" pop-up to dismiss it.

Arrange the two graphs in your dashboard to your liking. Click on the "save" link at the top of your screen and confirm the save by clicking the "Save" button. Now your dashboard will look something like this:



Adding Even More Hosts

Let's configure the kanboard machine to send its log messages to our centralized syslog server. Open up a new command session, start the kanboard machine, and connect to it.

```
cd linuxclass
cd kanboard
vagrant up
vagrant ssh
```

Let's configure this system to forward its syslog messages to the centralized syslog server. To do that, let's create a logstash.conf file in the /etc/rsyslog.d directory.

```
sudo nano /etc/rsyslog.d/logstash.conf
```

Place the following contents in the file and save the file.

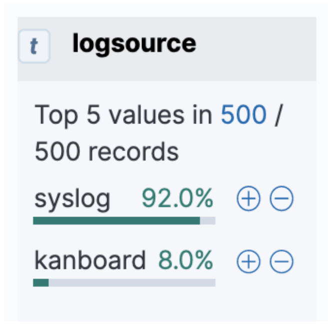
```
*.* @10.23.45.50:5141
```

This will cause rsyslog to send a copy of every syslog message to Logstash on the syslog server. Restart rsyslog to enable this configuration.

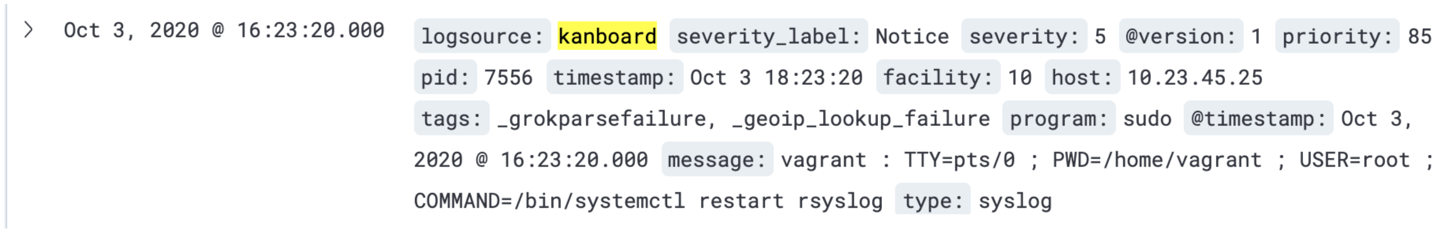
```
sudo systemctl restart rsyslog
```

Now return to the Kibana web interface (http://10.23.45.50:5601/app/discover#/). Click on the menu icon and then click on the "Discover" link under the "Kibana" section of the menu.

In the left-hand field menu, click on "logsource." You should now see the kanboard host appear in addition to the syslog host.

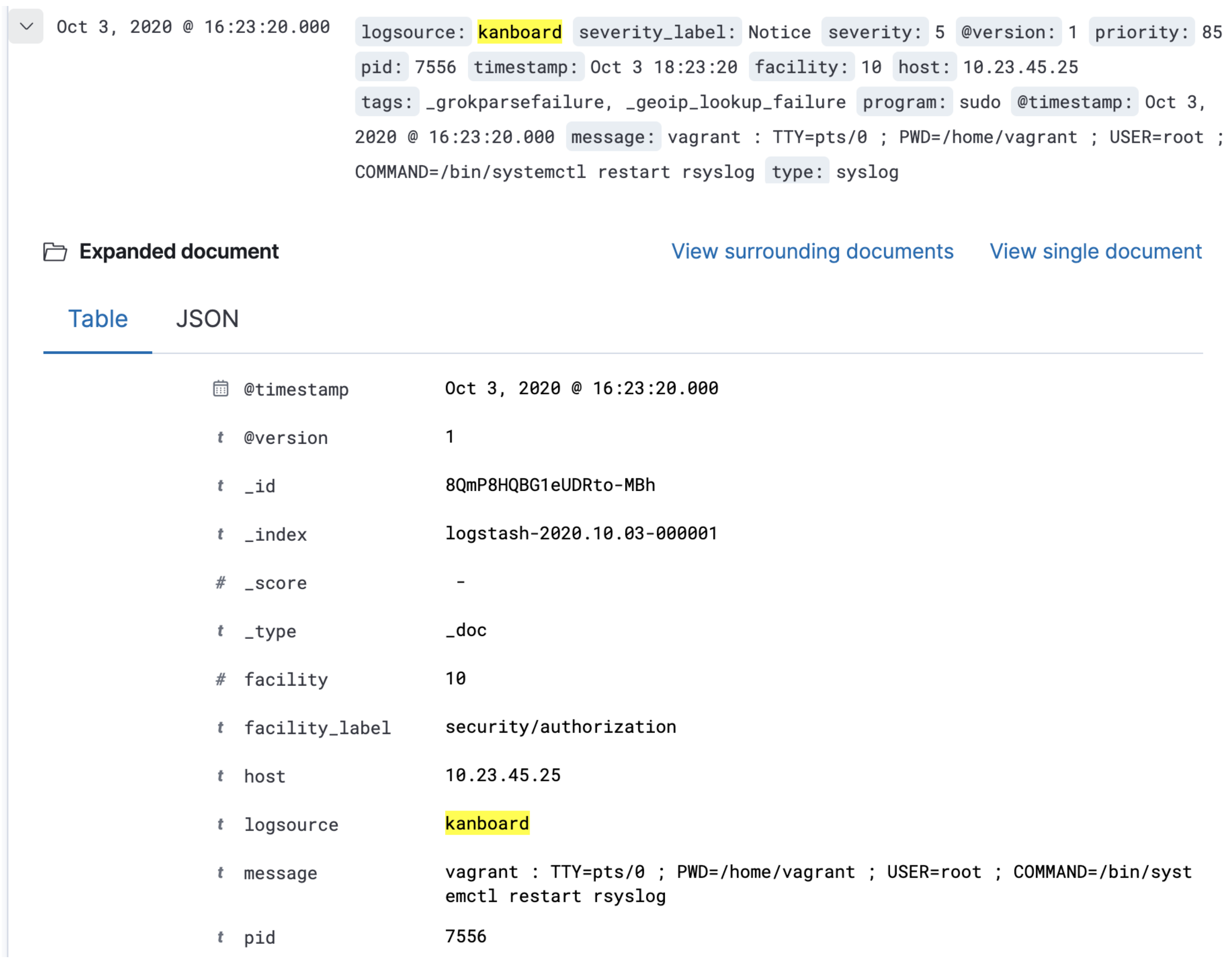


Now you can search across multiple hosts in one single place. If you need to narrow your search to one host, you can do that as well. Let's use this search to only display the messages coming from the kanboard host: "logsource:kanboard". You can also use "host:10.23.45.25". Here is an example message:



```
> Oct 3, 2020 @ 16:23:20.000   logsource: kanboard severity_label: Notice severity: 5 @version: 1 priority: 85
                               pid: 7556 timestamp: Oct 3 18:23:20 facility: 10 host: 10.23.45.25
                               tags: _grokparsefailure, _geoip_lookup_failure program: sudo @timestamp: Oct 3,
                               2020 @ 16:23:20.000 message: vagrant : TTY=pts/0 ; PWD=/home/vagrant ; USER=root ;
                               COMMAND=/bin/systemctl restart rsyslog type: syslog
```

By the way, you can click on the arrow next to the message to get even more details about. Here's what that looks like:



```
Oct 3, 2020 @ 16:23:20.000   logsource: kanboard severity_label: Notice severity: 5 @version: 1 priority: 85
                             pid: 7556 timestamp: Oct 3 18:23:20 facility: 10 host: 10.23.45.25
                             tags: _grokparsefailure, _geoip_lookup_failure program: sudo @timestamp: Oct 3,
                             2020 @ 16:23:20.000 message: vagrant : TTY=pts/0 ; PWD=/home/vagrant ; USER=root ;
                             COMMAND=/bin/systemctl restart rsyslog type: syslog
```

```
@timestamp        Oct 3, 2020 @ 16:23:20.000
@version          1
_id               8QmP8HQBG1eUDRto-MBh
_index            logstash-2020.10.03-000001
_score             -
_type             _doc
facility          10
facility_label    security/authorization
host              10.23.45.25
logsource         kanboard
message           vagrant : TTY=pts/0 ; PWD=/home/vagrant ; USER=root ; COMMAND=/bin/syst
                  emctl restart rsyslog
pid               7556
```

You can clearly see that the host is 10.23.45.25, the logsource is kanboard, etc.

Adding Sample Data

I put a server on the Internet for one month without a firewall and collected the syslog messages that were generated. I've narrowed that data down to just one day's worth of logs. Run the following command to download and insert that data into your Elasticsearch instance.

Note: while this script is running Kibana and Elasticsearch will be unavailable. This process will take a few minutes to complete.

```
curl -fsSL http://mirror.linuxtrainingacademy.com/elasticsearch/es-sample.sh | sudo bash
```

You can verify that the data was successfully imported into Elasticsearch by looking at its indices.

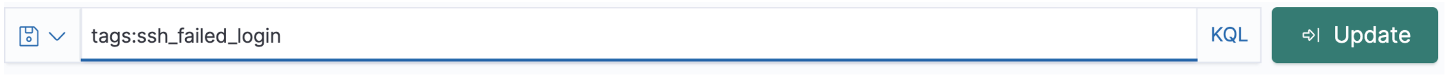
```
curl http://localhost:9200/_cat/indices?v
```

You will now see a new index named "logstash-2020.04.15" which contains the logs from the system described above.

Using the Sample Data

Let's create a map of where all the failed ssh attempts are coming from. Click on the menu icon in the top-left corner. Under "Kibana" click the "Maps" link.

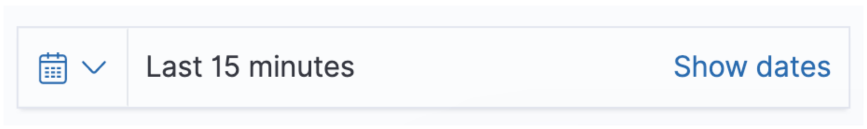
In the search bar, enter in "tags:ssh_failed_login" and click the "Update" button.


tags:ssh_failed_login

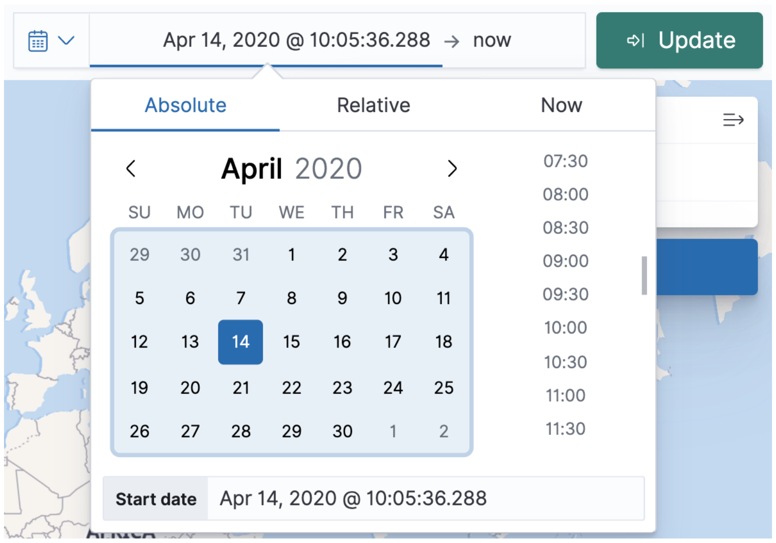
This will narrow our results to just the failed ssh logins.

Now, let's set our search range to ensure we can see the messages from 04/15/2020.

In the Time Picker menu, click "Show Dates".

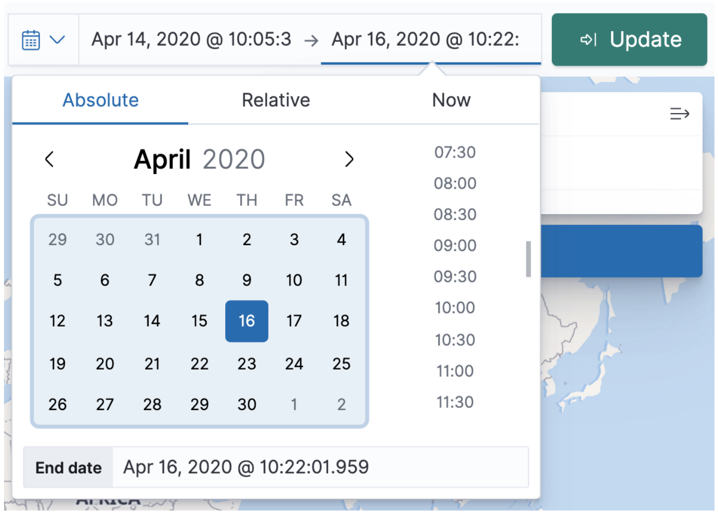


Click on the left-hand side of the of date picker to bring up a date selector menu. Click "Absolute". Use the calendar to select the date of April 14, 2020. This is the start date.

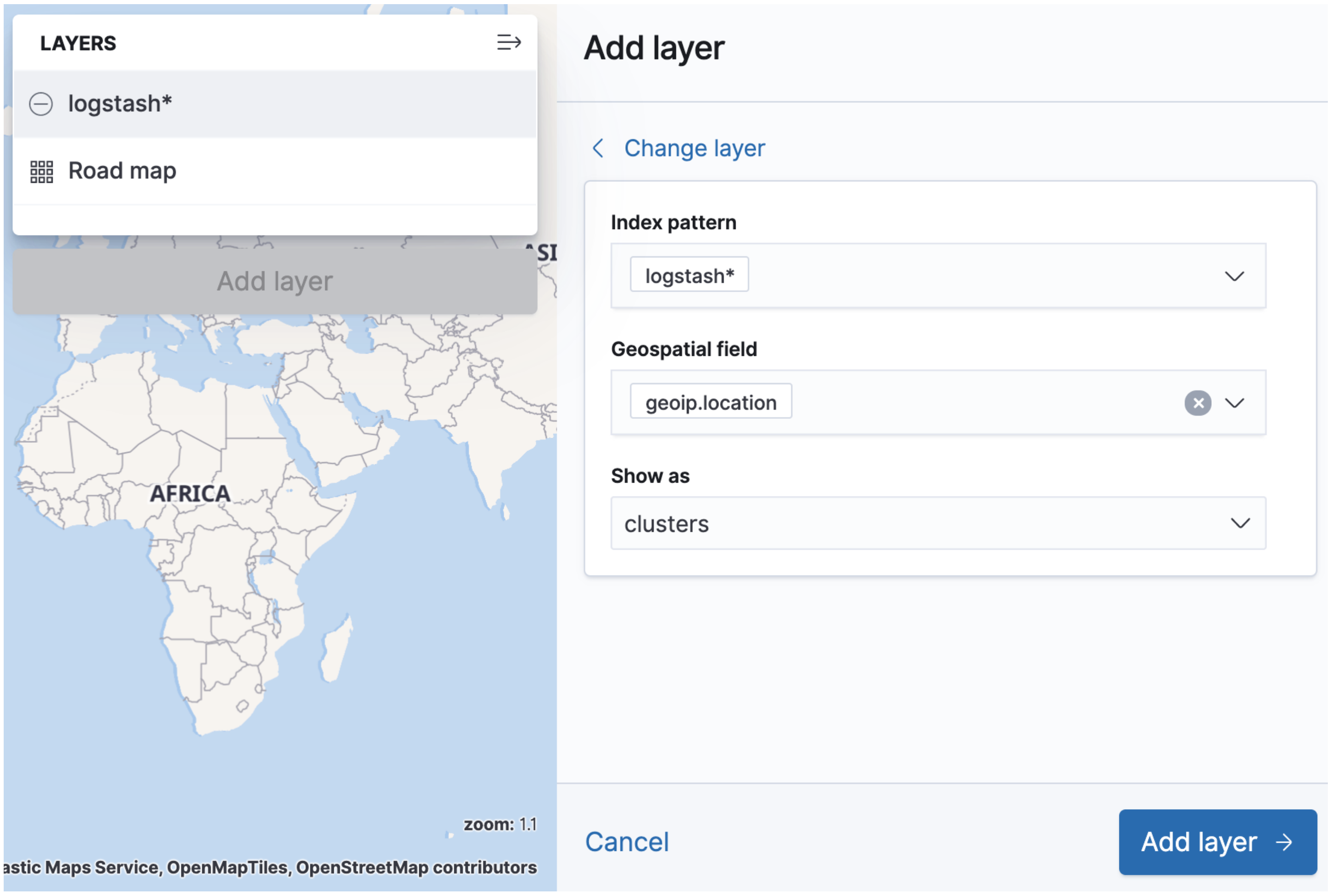

Apr 14, 2020 @ 10:05:36.288 → now

Start date Apr 14, 2020 @ 10:05:36.288

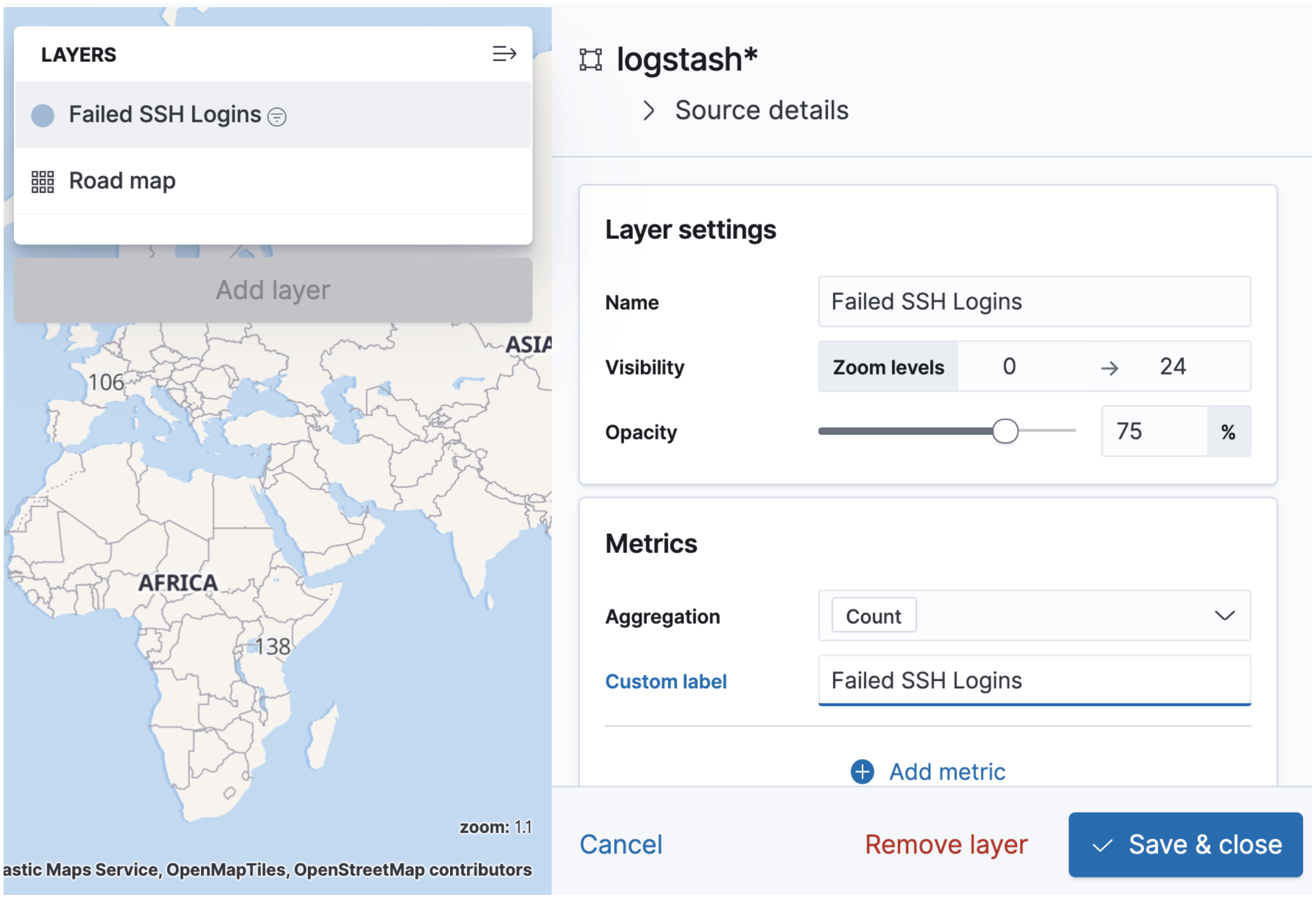
Next, let's set our end date. Click on the right-hand side of the date picker where it says "now". Click "Absolute". Use the calendar to select the date of April 16, 2020. Finally, click the Update button.


Apr 14, 2020 @ 10:05:3 → Apr 16, 2020 @ 10:22:
Update
Absolute
Relative
Now
April 2020
End date
Apr 16, 2020 @ 10:22:01.959

Click the "Add Layer" button, then click "Clusters and Grids". In the "Index pattern" drop-down menu, select "logstash*". Next, click "Add Layer".



In the "Name" field, enter "Failed SSH Logins". In the "Custom label" field also enter "Failed SSH Logins".



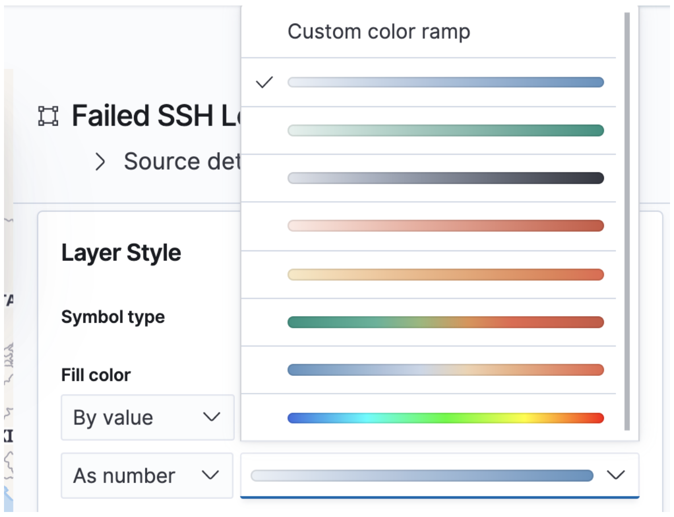
Layer settings

Name: Failed SSH Logins
Visibility: Zoom levels 0 → 24
Opacity: 75 %

Metrics

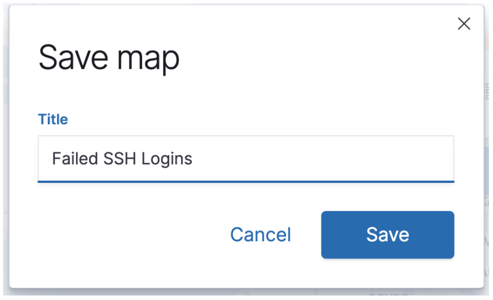
Aggregation: Count
Custom label: Failed SSH Logins

The default color is very close to the map color, so let's change it to something that is more contrasting. Scroll down to the "Layer Style". Click the color next to "number". A menu of color choices will appear. Select the color at the very bottom. Click the "Save & close" button.



Now we should see some circles and numbers on our map. These clusters represent the number of failed SSH attempts in that given area. The larger the cluster, the more failed ssh attempts have been made from that area.

Click the "Save" link at the top of your screen and give this graph a name such as "Failed SSH Logins", for example. Click the "Save" button.



Save map

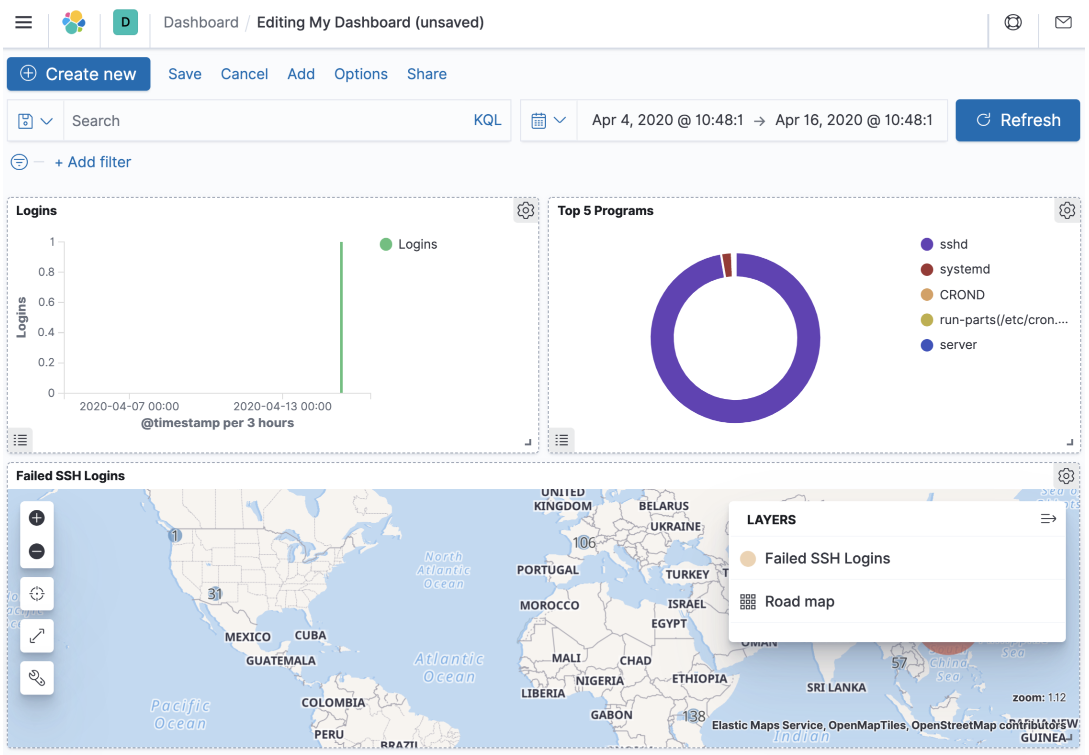
Title

Failed SSH Logins

Now you can add the map to your dashboard if you'd like. Click on the menu icon in the top-left corner. Under "Kibana" click the "Dashboard" link. Click on the "My Dashboard" link to display your dashboard.

Now, click the "Edit" link at the top of your screen. Now click the "Add" link at the top of your screen. Finally, click on the "Failed SSH Logins" link to add the map to the dashboard. Click the "X" in the "Add Panels" pop-up to dismiss it.

Arrange the graphs in your dashboard to your liking. Click on the "save" link at the top of your screen and confirm the save by clicking the "Save" button. Now your dashboard will look something like this:



Dashboard / Editing My Dashboard (unsaved)

Explore

From here, you can configure other hosts to send their messages to your syslog server. You can also dig through some logs and look for interesting things. Feel free to create more graphs and come up with your own dashboards.