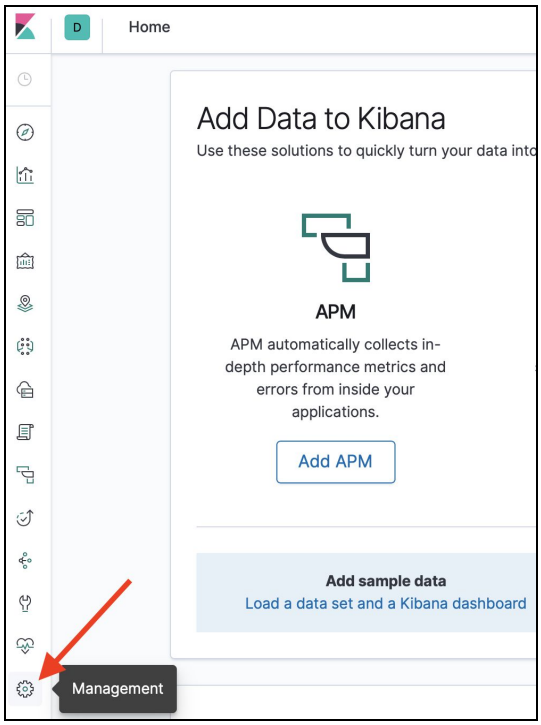
Elastic Lab

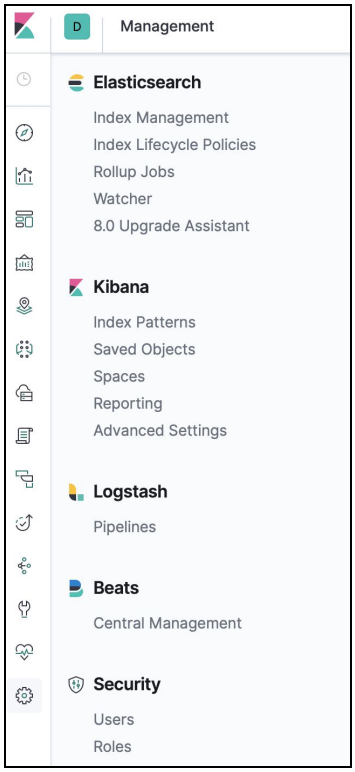
Let's explore the Elastic Stack

Lab 1: Working with Spaces and Role-based Access Control (RBAC)

In this section, we are going to explore Kibana Spaces as a way to organize and manage access to your visualizations and dashboards. We will also take a look at several sample data sets and use them to demonstrate how you can restrict access by role.

1. We'll start by opening the Kibana **Management** tab.

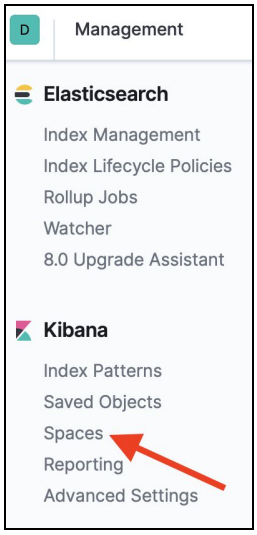




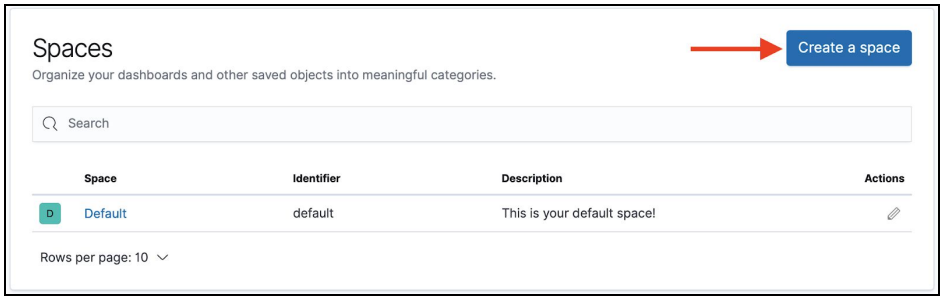
Before we jump in, looking at the management options you see, think through how you would create a **Space** for a **User** with a specific **Role**. Spend a few minutes exploring and then come back to the **Management** page.

How would you do it?

2. Open the Spaces Management page.

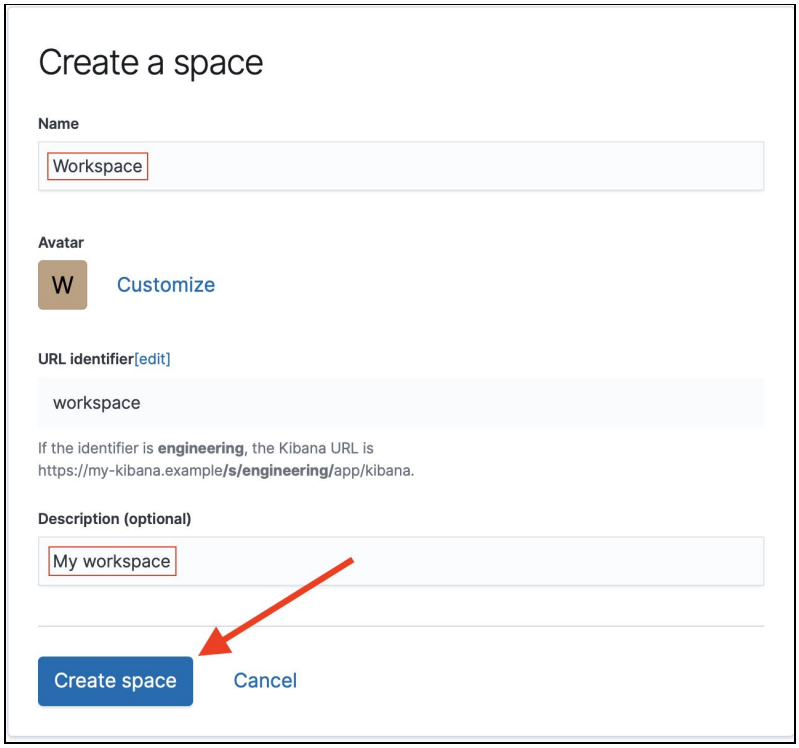


3. We will create three new Spaces:
   - Workspace
   - Business
   - Operations



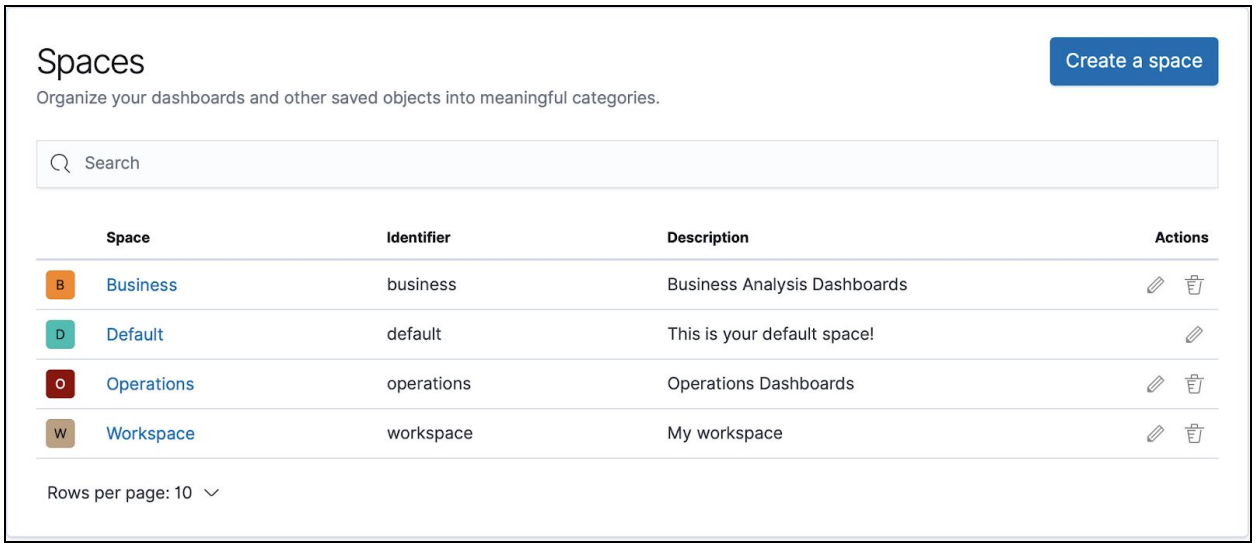
To start, click **Create a space**

4. Give your first new Space the name “Workspace” and add a description such as “My Workspace”.



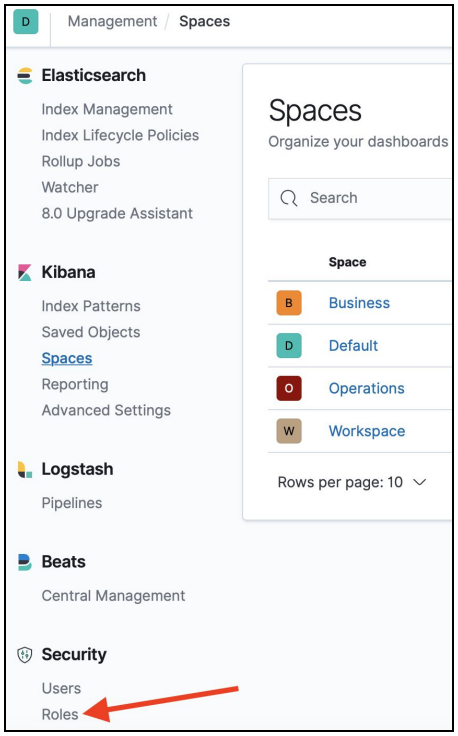
Then click **Create space**

5. Repeat this to create a “Business” space and an “Operations” space.



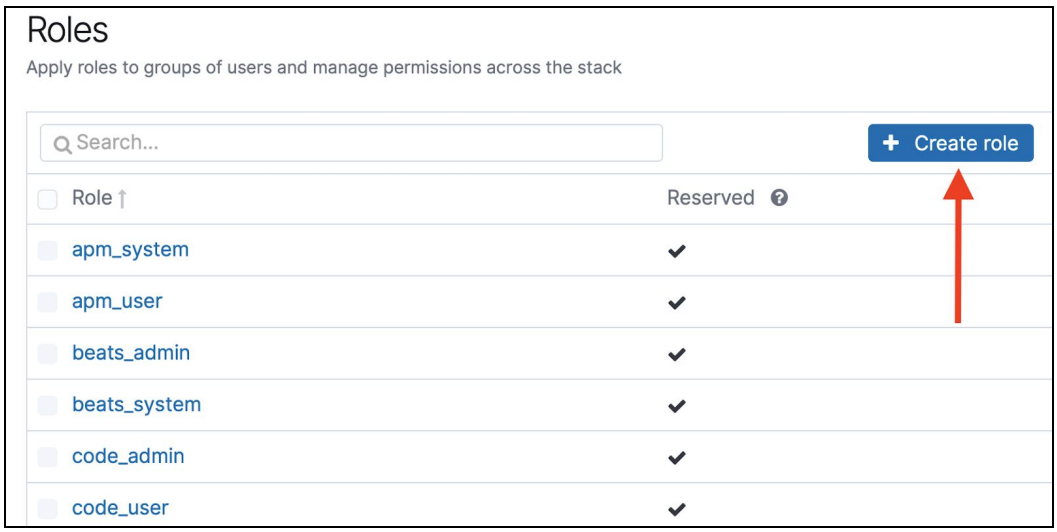
When your done, your **Spaces** management page should like something like the screen above.

6. Next we will create some new Roles. Let's click on the Roles link in the left navigation to get started.



Select **Roles** from the **Security** management section.

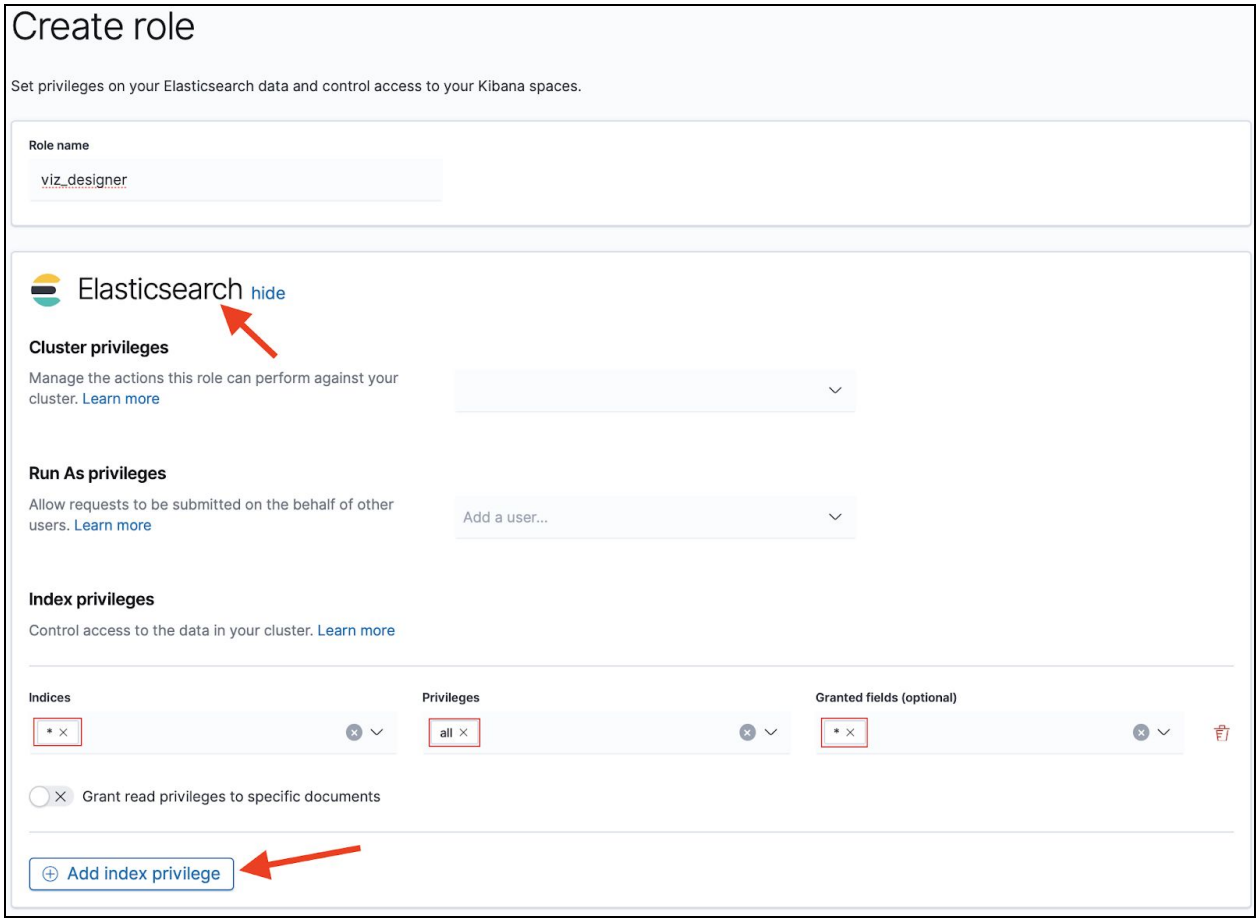
Here you see all of the default roles for the Elastic Stack.



Click on **+ Create role** to begin creating a new role.

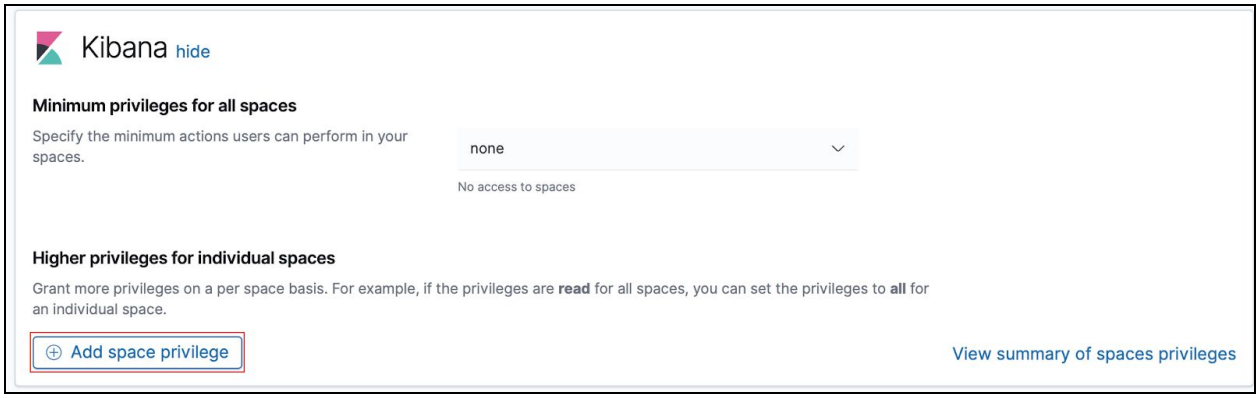
Enter “viz_designer” as the role name

As a visual designer, we won’t need any cluster privileges but we do need privileges for the indices in the cluster. To keep things simple, we’ll grant full access to all indices in the Elasticsearch cluster.

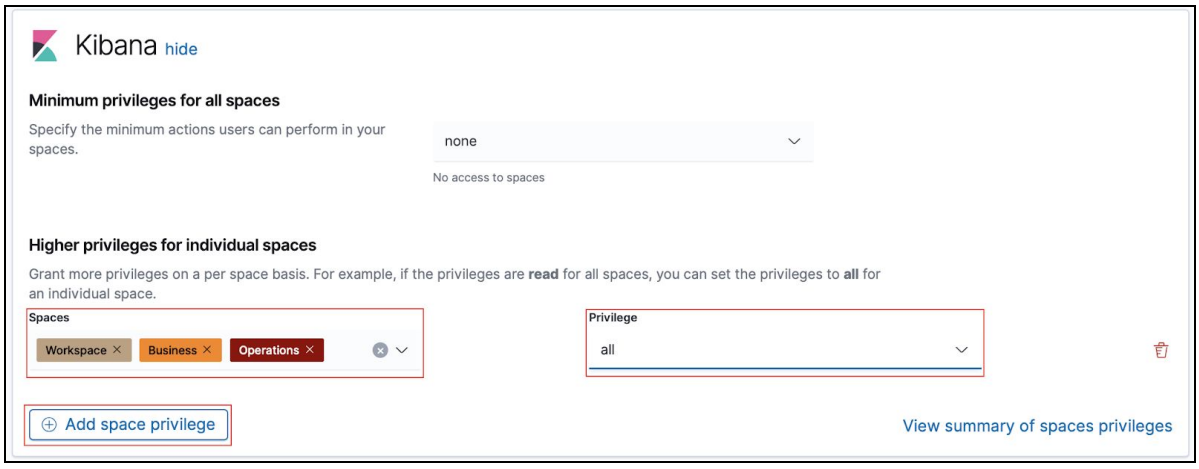


In the Elasticsearch section, specify all Indices by adding "*" in the **Indices** selector and select "all" in the **Privileges** selector. You can restrict permissions to specific fields in an index but we will keep the default "*" to allow access to all fields. If we wanted to add more index privileges, we could do that here; however, we will move on for now!

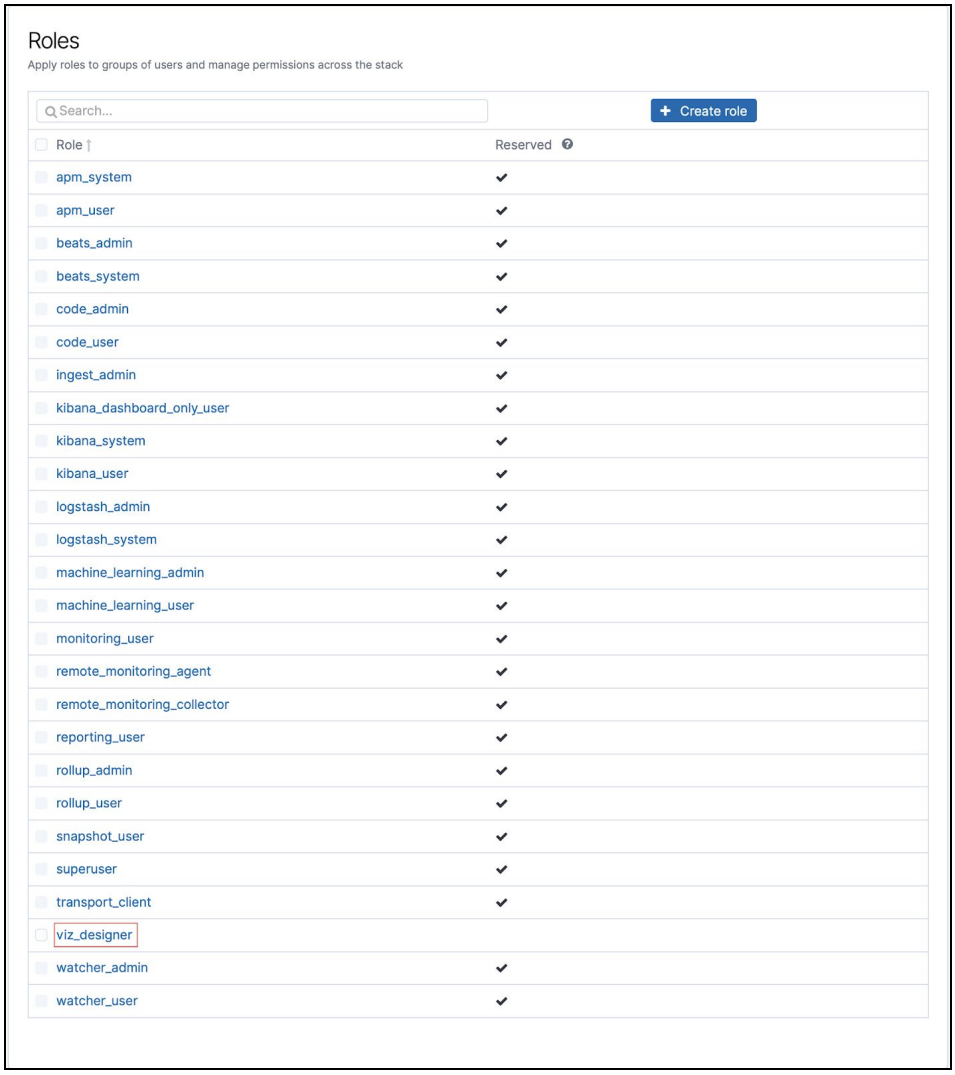
Next we'll move to the **Kibana** section to configure privileges for the spaces that we created above. Start by pressing **Add space privilege.**



As a visualization designer, we will need privileges for the **Workspace** space where we will do our design work, as well as the **Business** and **Operations** spaces so that we can populate those spaces with the visualizations and dashboards we create. We will grant full access to our three spaces.

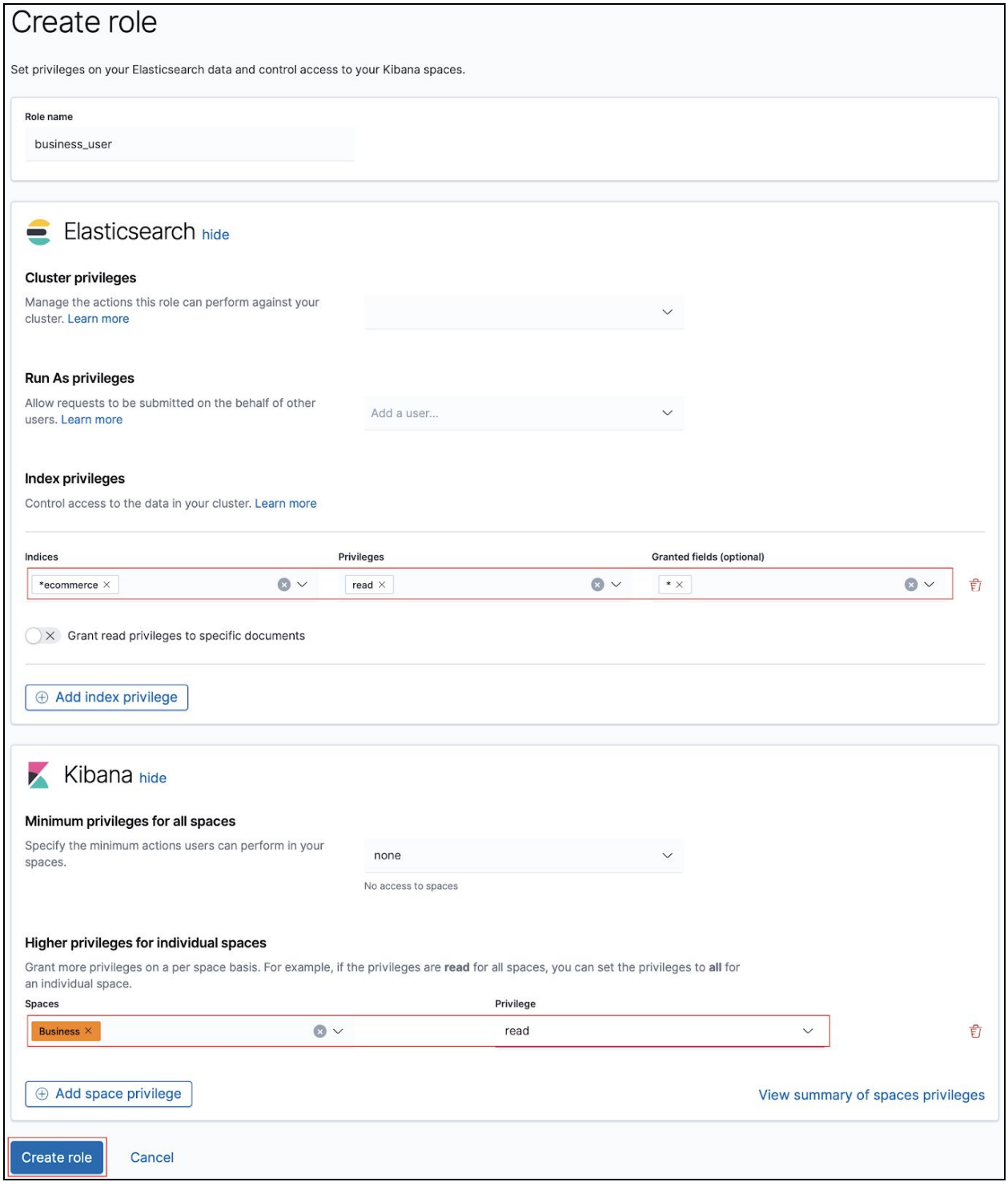


1. Press **Add space privilege**.
2. In the **Spaces** selector, add the **Workspace**, **Business**, and **Operations** spaces.
3. In the **Privileges** selector, choose **all**.
4. Finally, press **Create role** to create the **viz_designer** role.



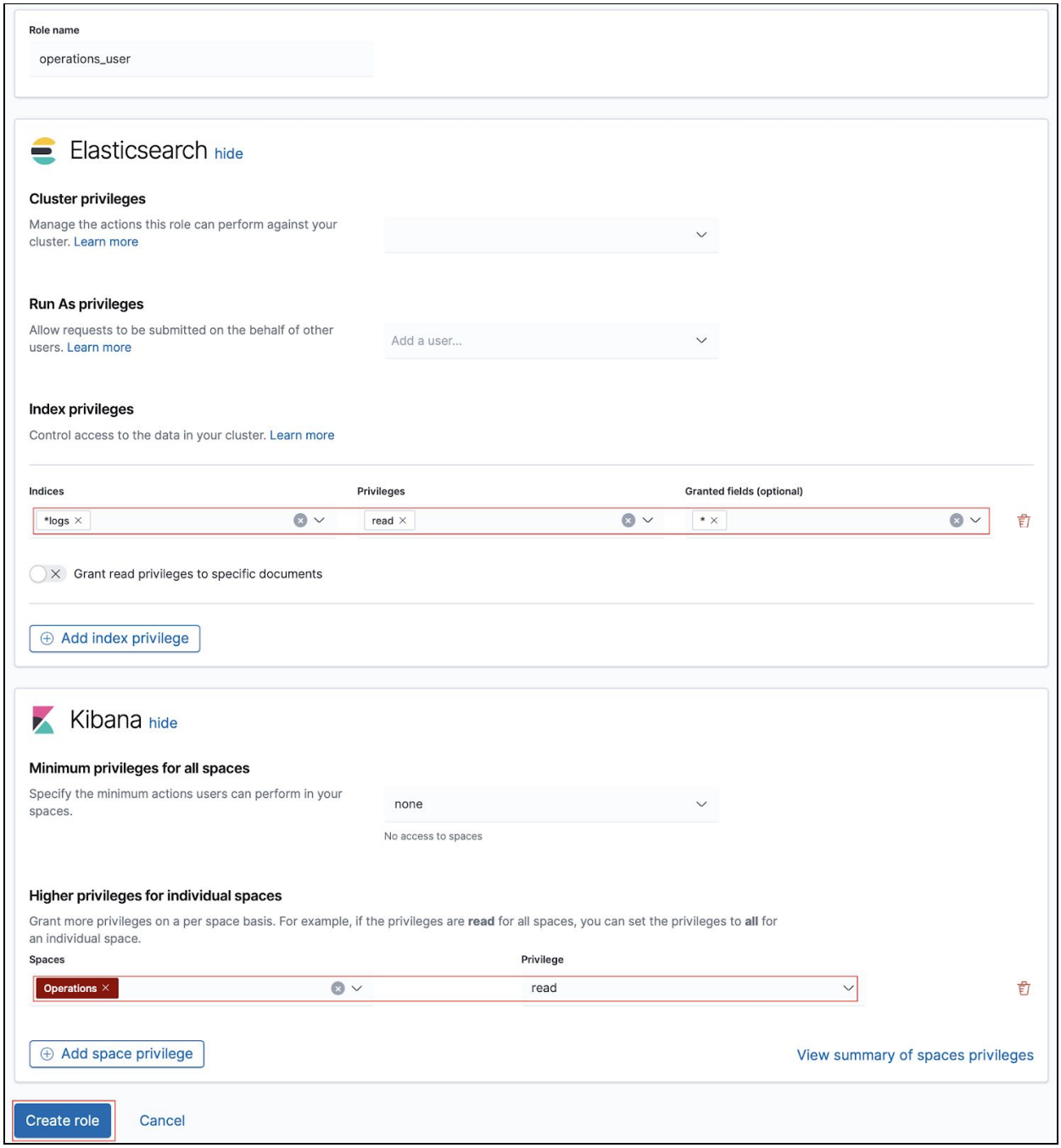
Next we will repeat this process to create our **business_user** and **operations_user** roles but these roles will only have access to their respective **Business** and **Operations** spaces and specific indices related to their roles.

Create the "**`business_user`**" role with **read** privilege for the "**`*ecommerce`**" index and **read** privilege for the **Business** space as shown below.



Press **Create role** to finish creating the role.

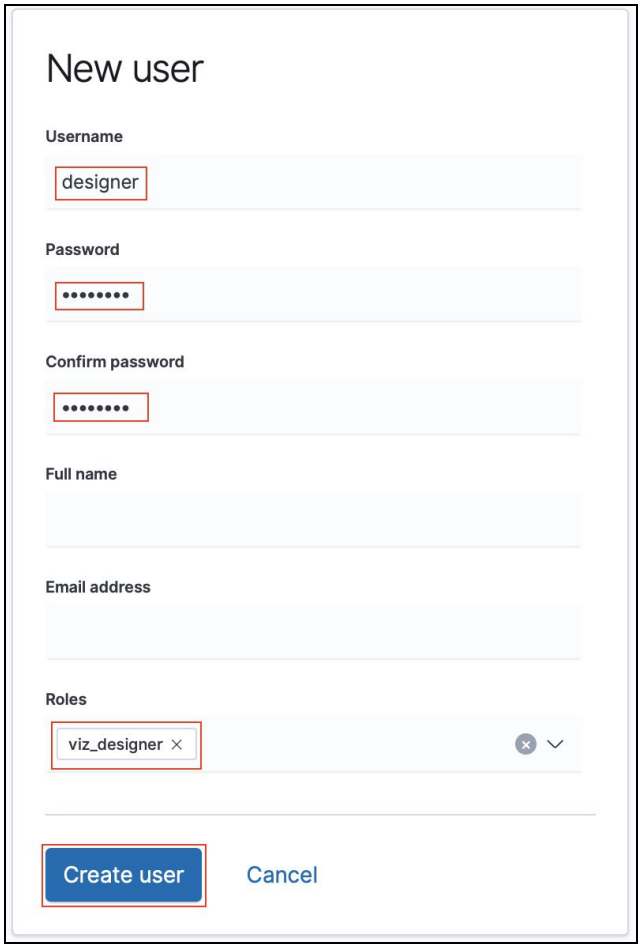
Finally, create the "`operations_user`" role with **read** privilege for the "`*logs`" index and **read** privilege for the **Operations** space as shown below.



Press **Create role** to finish creating the role.

7. Next we will create three users and assign the roles we just created to them. To keep things simple, we'll name the users after their respective roles. Click on **Users** in the left nav. This click on **Create new user** from the Users list.

Enter "**`designer`**" as the **Username.** Enter "**`changeme`**" in the **Password** and **Confirm password** fields, and select the role **viz_designer** for the **Roles** field. The **Full name** and **Email address** fields are optional.



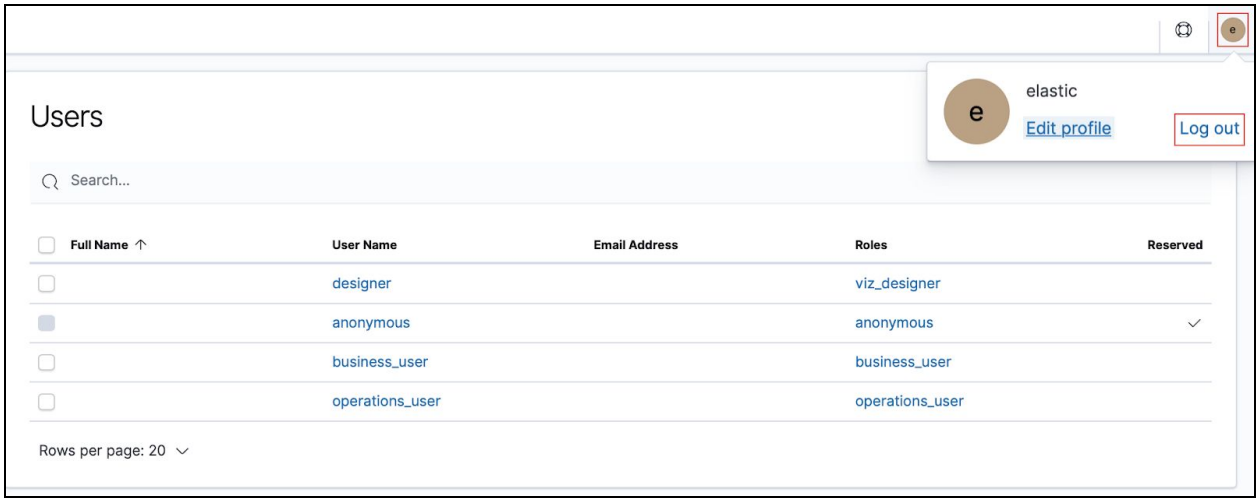
Press **Create user** to create this user.

Repeat this to create two more users:

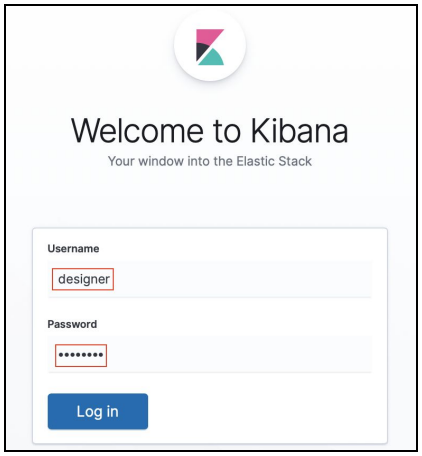
- User **Business User** with role **business_user**, and
- User **Operations User** with role **operations_user**.

You should now have three new users that look something like this:

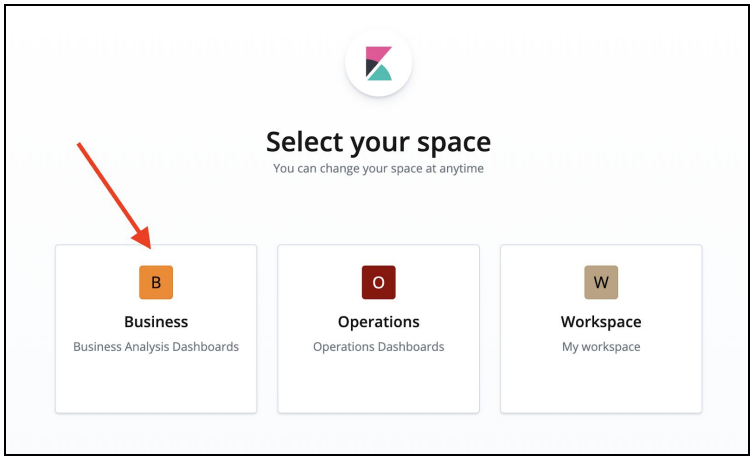
8. Now we will log out as the **elastic** user and log back in as the **Visual Designer** user so that we can add some sample data for each of the users. Recall that the **viz_designer** role has access to all three spaces that we created above. Click on the **icon** in the upper right corner of the screen and then click on **Log out**.



Log in as **designer** with the password you assigned earlier (**changeme**).

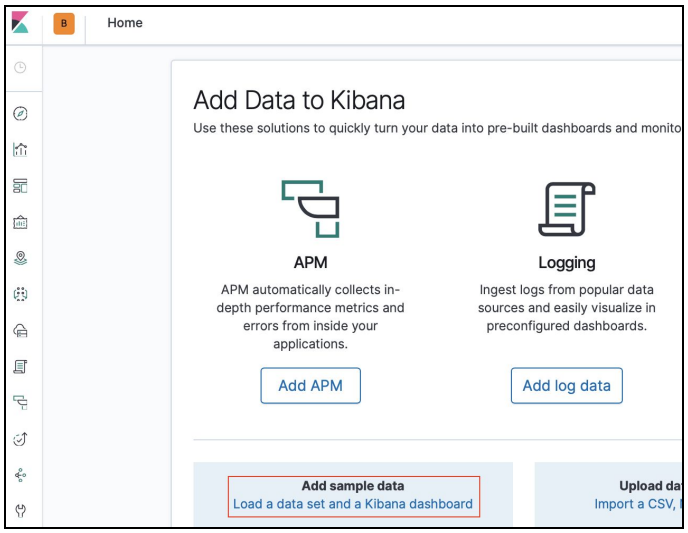


Select the **Business User** space so that you can add some sample ecommerce data to that space

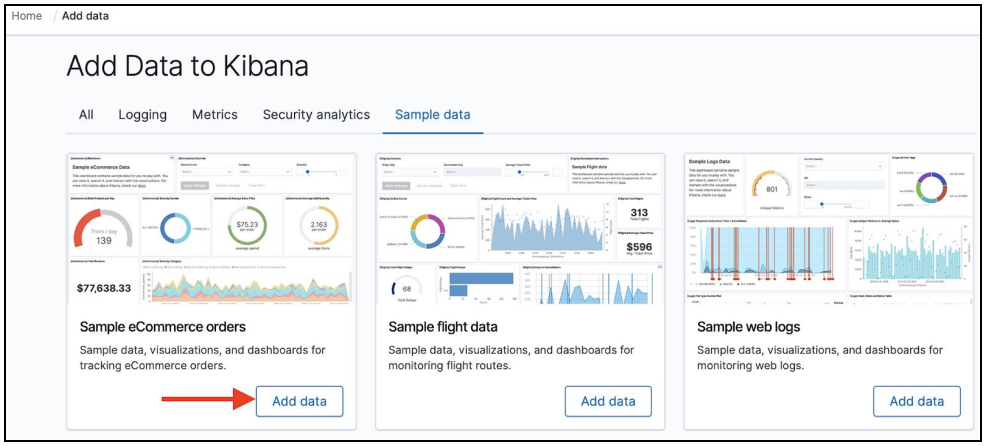


.

Select **Add sample data**.

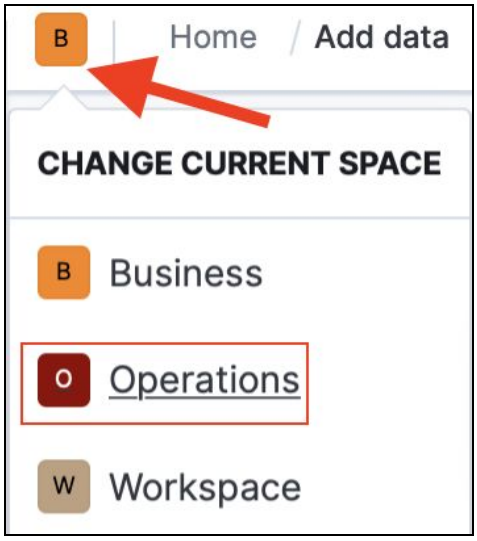


Since this is the **Business** space, we'll add some sample eCommerce data.



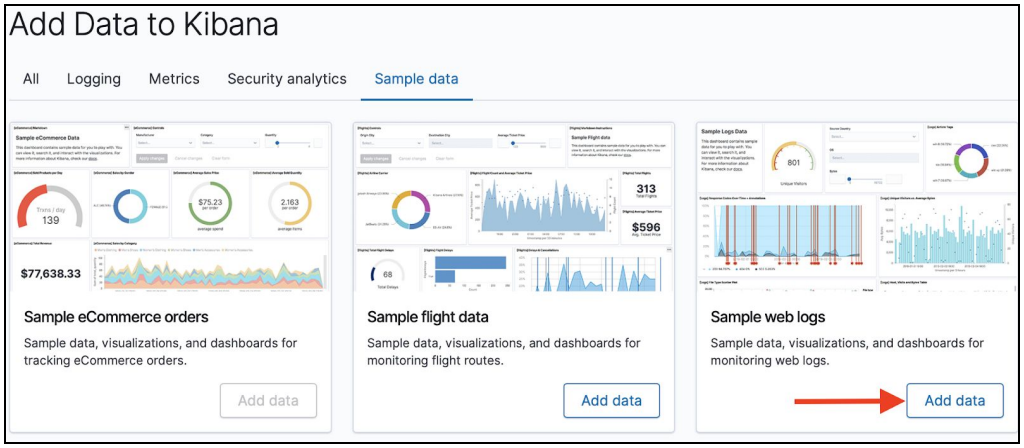
Press the **Add** button to add sample eCommerce data, visualizations, and dashboards to the **Business** space.

Now we'll switch over to the **Operations** space so that we can add some web log data to that space.



Press on the **Business** space in the left column to bring up the menu of available spaces and then select the **Operations** space.

Go back to **Add sample data** but this time add the sample **web log** data since we are in the **Operations** space.



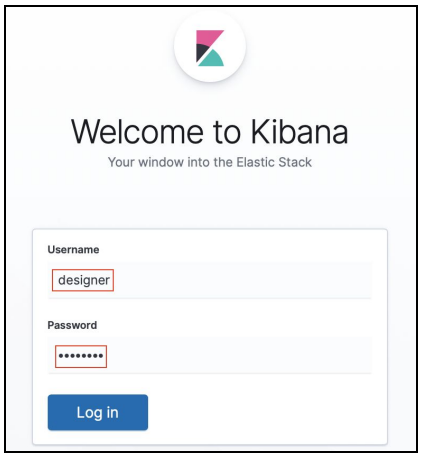
9. Now we will log out as the **Visual Designer** and log in as the **Business User**.

X. Congratulations! You are done with Section 2 of the lab! For the rest of the labs below, you will be working in the role of **visual_designer** so log out and log back in as the user **Visual Designer**.

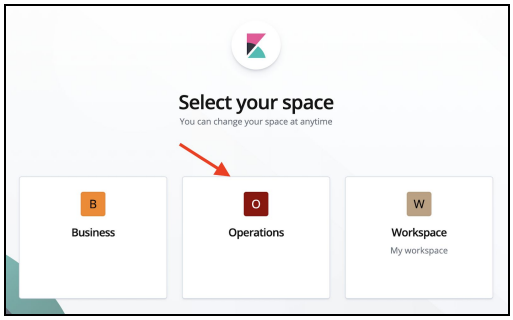
Lab 2: Learning by Reverse Engineering

In this section, we are going to take advantage of learning what others have done. We are big believers in building by this principle. It's a great way to learn. First, we are going to dive into the details of how visuals are built, next we will examine canvas, and then maps. In future labs, we will go over how to build visualizations and how to work with canvas to build amazing graphs; however, in this lab, please do sit back and appreciate what others have brought to the table. There is no reason you couldn't try to reconstruct any visualization that another person has built within Kibana: it's easy to do!

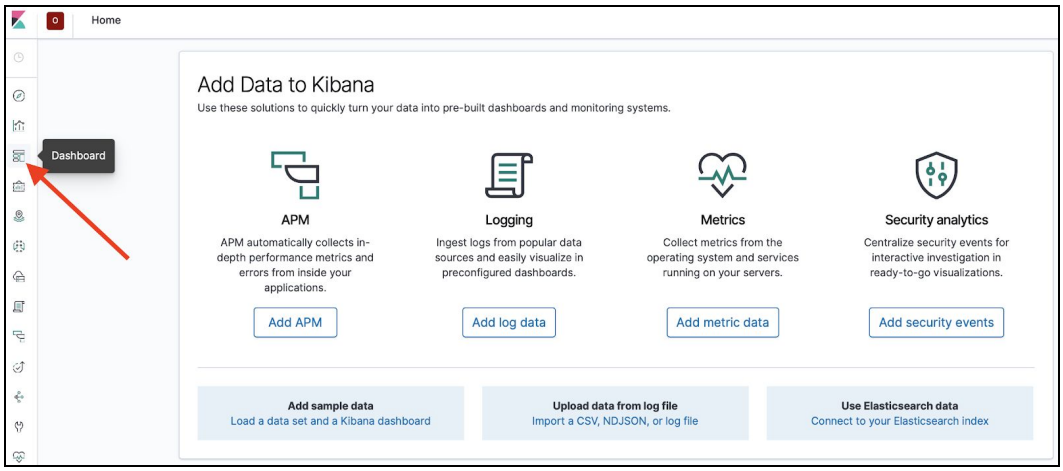
1. Log back in as **designer**



2. Select Operations. For the remainder of the lab we are going to work with logs as an administrator.

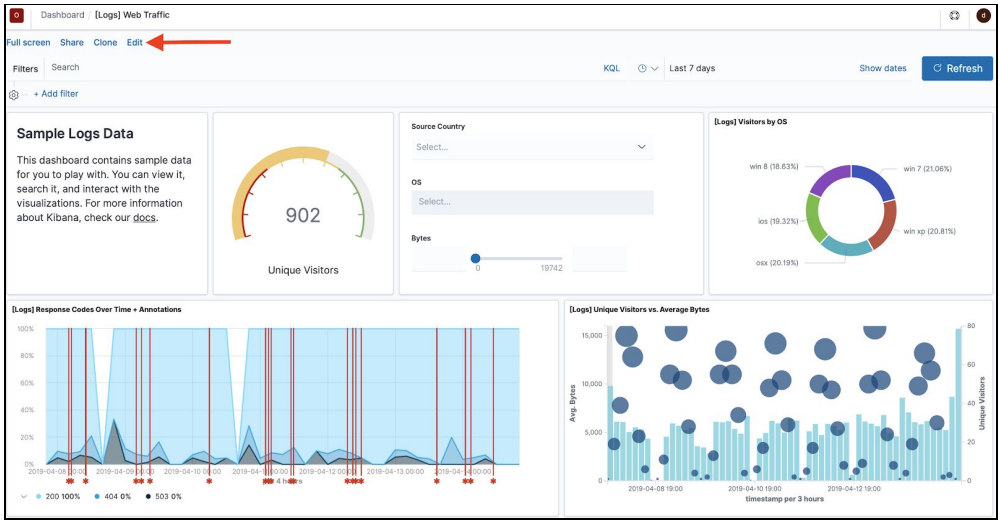


3. You will be sent to the splash page for your logs space. Note all the create assets you have at your disposal here. It's worth checking out at some point! You can get instructions on how load the 10's of beat modules with the click of a button and start ingesting log and metric data. You can learn how to get started with APM. There are a lot of directions we could take you here; however, we are running a visualization workshop (please do ask your instructor about logs and APM workshops!). In the meantime, let's take a look at the dashboard which shipped with the logs dataset. Click on **Dashboard**.

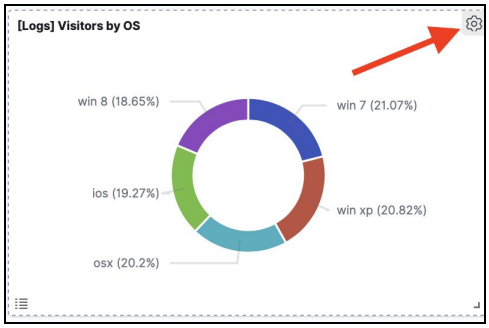


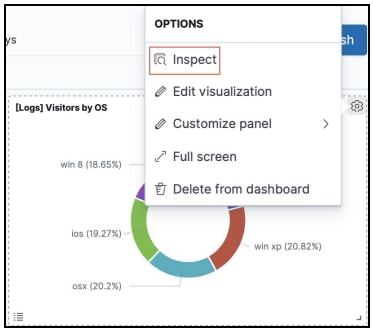
4. Click on the **[Logs] Web Traffic** dashboard

5. You are now looking at the dashboard for logs. At the top of the dashboard, there are controls to navigate time and view, as well as search; we will explore some of this in the discover lab. For now, we'd like to simply appreciate what someone else has built and learn from it. Notice all the visuals on this page. Each facet has a title. We would like to figure out how to build a pie chart ourselves. To do this, we are going to reverse engineer. Click on **Edit**. This will enable us to drill down into any visual we want to.

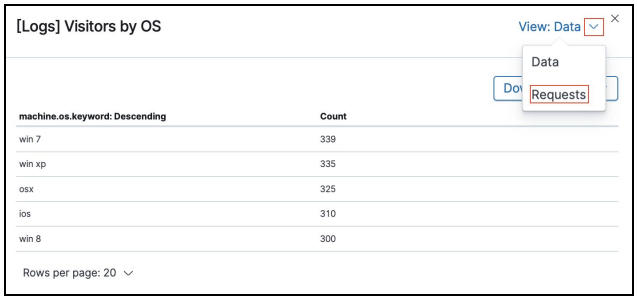


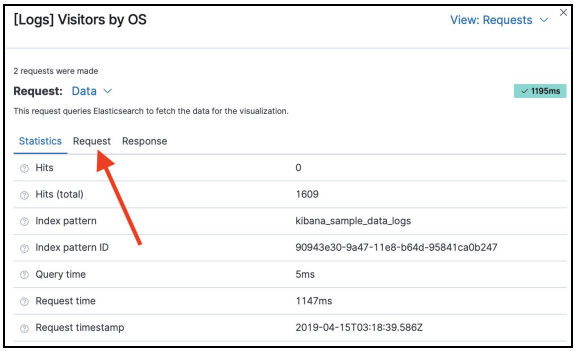
6a. In our case, we'd like to understand how the pie chart was built, but prior to doing that, let's take a look at the Elasticsearch query which provides the visualization with the data. Go ahead and click on the **gear** in the upper left corner of the visual and then click on **Inspect**.



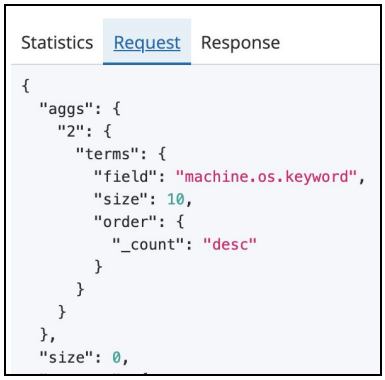


6a. These are the actual values which are used to light up the visual. Click on **Requests** in the dropdown to the right of **View Data**. Next, click on the **Request** link to look at the request.



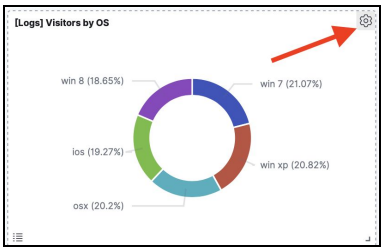


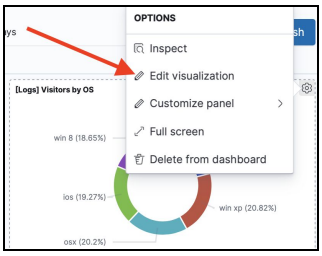
6e. We are now looking at the actual JSON which is posted to Elasticsearch in order to receive the response which can also be viewed here. Note that the request is clearly aggregating machine operating systems and ordering the top ten results in descending order.



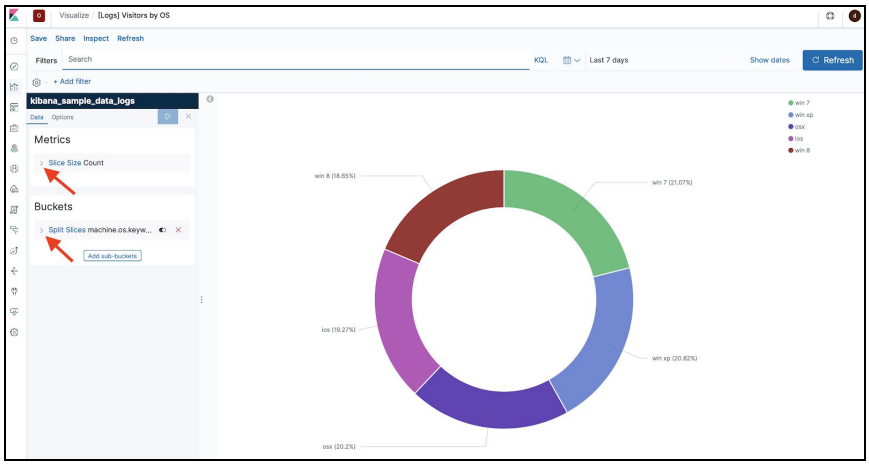
6f. If we wanted to, we could use that request and write our own API and render our own graphs. Folks within the Elastic community do that all day. But, guess what? No way are we going to do that here; after all, we have this great visualization tool called Kibana which does this for us. Let's dig into how the visualization was built. Click on the **X** to go back to the main dashboard screen.

7a. Now let's repeat the process; click the gear again, but this time pick **Edit Visualization**.

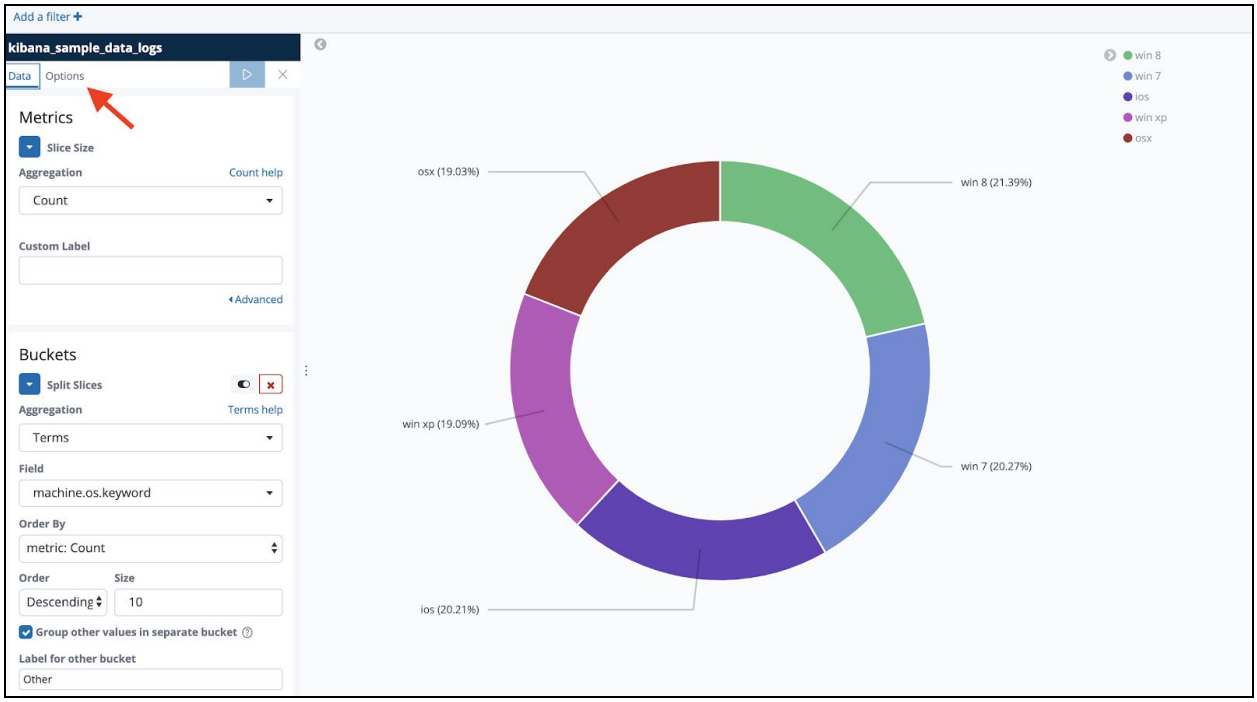


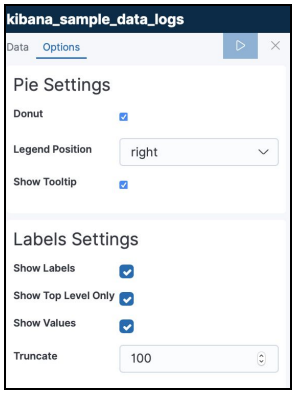


7c. We are sent straight to the makings of the visualization. Prior to investigating, let's expand the Metrics and the Buckets widgets by clicking on the arrows.



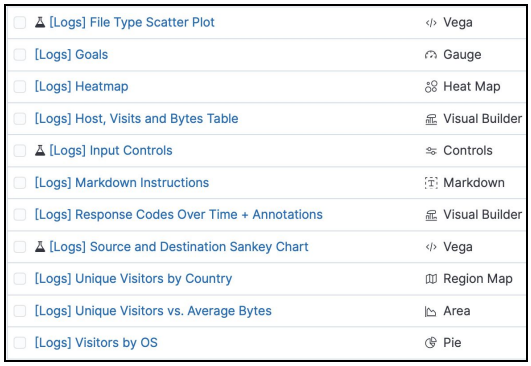
7d. Now we have a **full view** of the visualization for the pie chart in our sight. We can see that we are running an aggregate count of documents and bucketing those document counts up by operating system. We essentially have everything we need to reconstruct the pie chart ourselves. We could click **Options** to pick up the rest of the details, such as label details and legends.





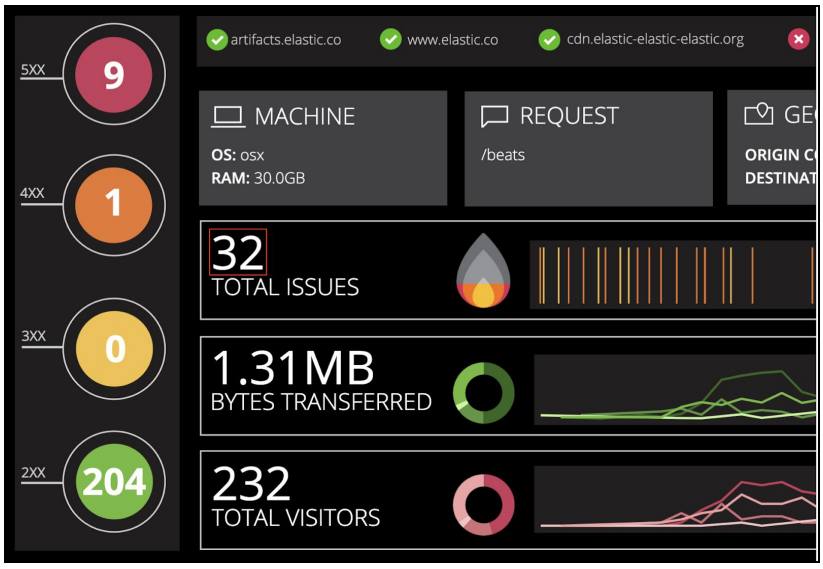
8a. Now, let's click on the **Visualize** link in the upper left of the screen.

8b. This will bring us to complete list of all visualizations available for us to begin to reverse engineer. We can now go shopping! Hey, why not? It's a free world! Please knock yourself out and call us when ready to move on.

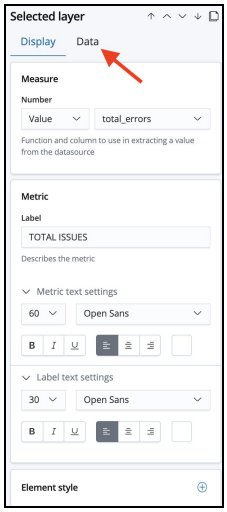


9a. This next section is **optional**. We are going to reverse engineer a visualization a **Canvas** workpad and take a look at how this was built.. Go ahead and click on **Canvas** in the left nav and then click on the **[Logs] Web Traffic** workpad.

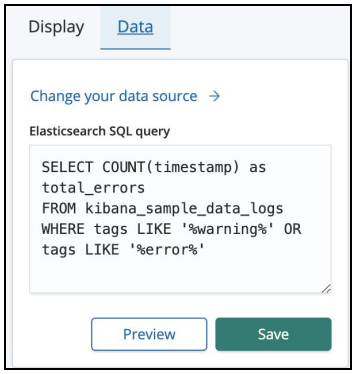
9c. This Canvas workpad is amazing! We want to know how it was built. Well we won't steal the thunder of a future lab just yet, but let's at least examine some of the components which make up this Infographic of live data (think slide with a value that is capable of refreshing live or being put on an operations monitor. Let's see how the number makes it to the page. Click on the **number of total issues** (in this case it's 32, your value may vary).



9d. Here, you can dive into the details of how this component is built or even try to use these values to reproduce, but we want to know how the data was glued into the visual, so next click on **Data**

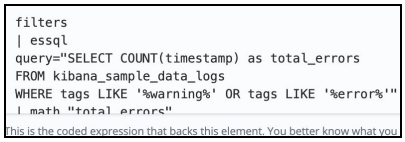


9e. You will see that the source of this data is in fact SQL which runs a count against the filtered records coming back from logs which have tags with the word warning or error in them. Pretty decent: now we have sample SQL which we could even repurpose.



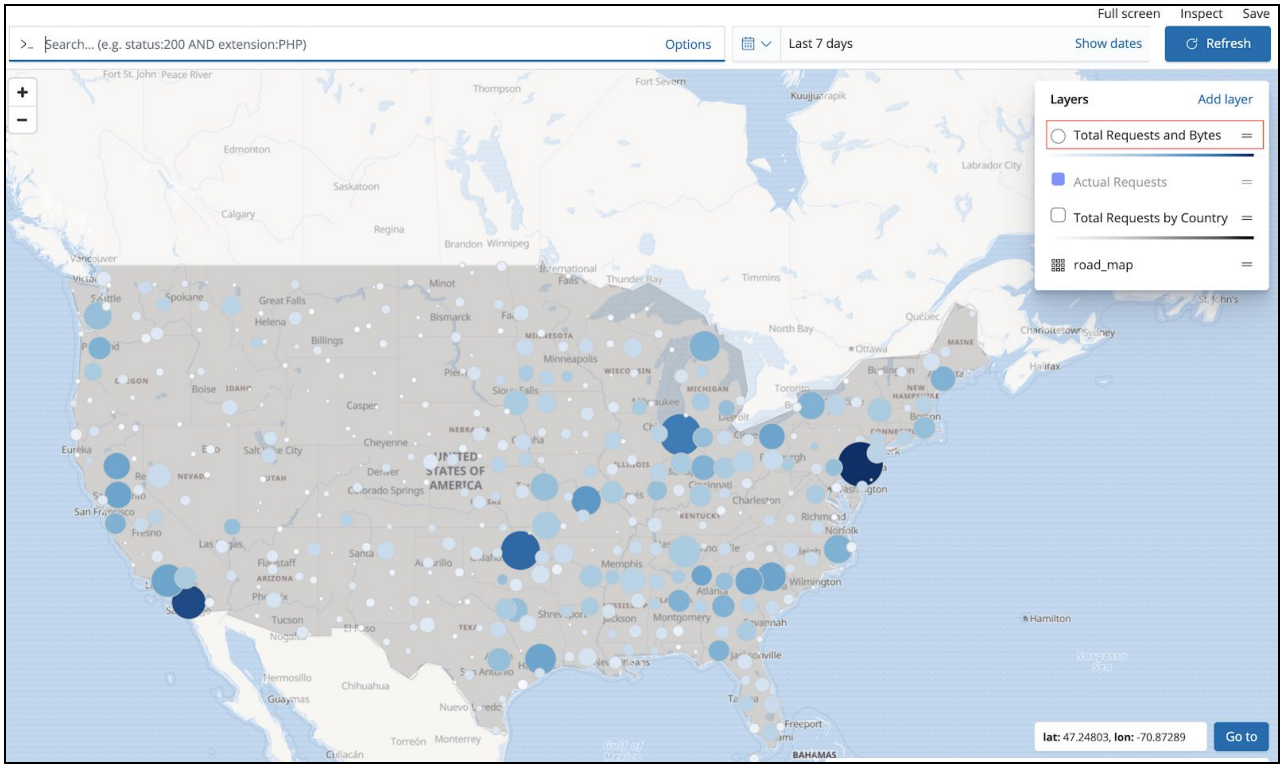
9f. Go ahead and clickon the **Expression editor** link in the bottom right. This will render the actual script which Canvas uses to render the screen **(lookup up term).**

9g. This actual script could be copied and pasted into a completely different environment if it has the same data and the visuals would render the same way. Alternatively, the expression could be tweaked to needs. Again, what a great place to copy, paste, and stash way for later!

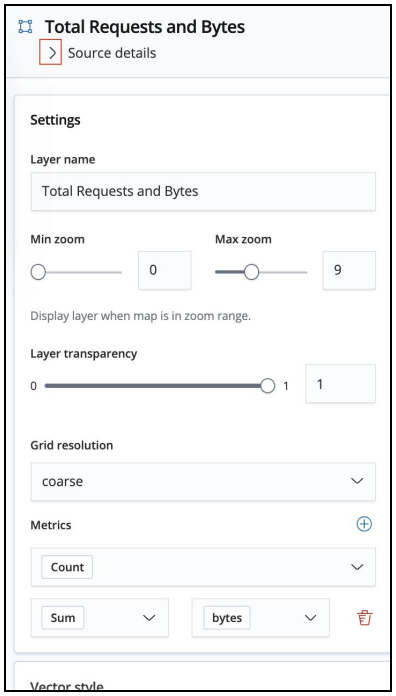


10a. Let's move on and optionally take a look at the Maps app within Kibana. Click on the **Maps** link in the left navigation. Then click on the **[Logs] Total Requests and Bytes** map link

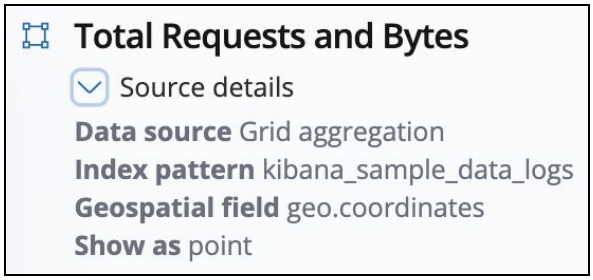
10c. Wow, look at this amazing example of map layers in action - fresh and ready for us to pull and scrape for our own use! Please do note: you will be able to learn how to build visualization maps into a dashboard in the next section, but this Maps app is a completely different beast: here we can actually join layers of data from potentially multiple sources into a single pane so that we can have a complete view of what's going on. In this case, all the layers which you can see listed in the upper right are sourced from either a map itself or the log data we've been working on. Click on the **Total Requests and Bytes** layer



10d. You can see all the visual settings which help to render this layer. The Kibana documentation has a great walkthrough of not just getting started with Maps but also of what all these knobs mean or….you could just turn the knobs and see what happens! Let's take a look at the source of this data. Click on the **> to the left of Source details**

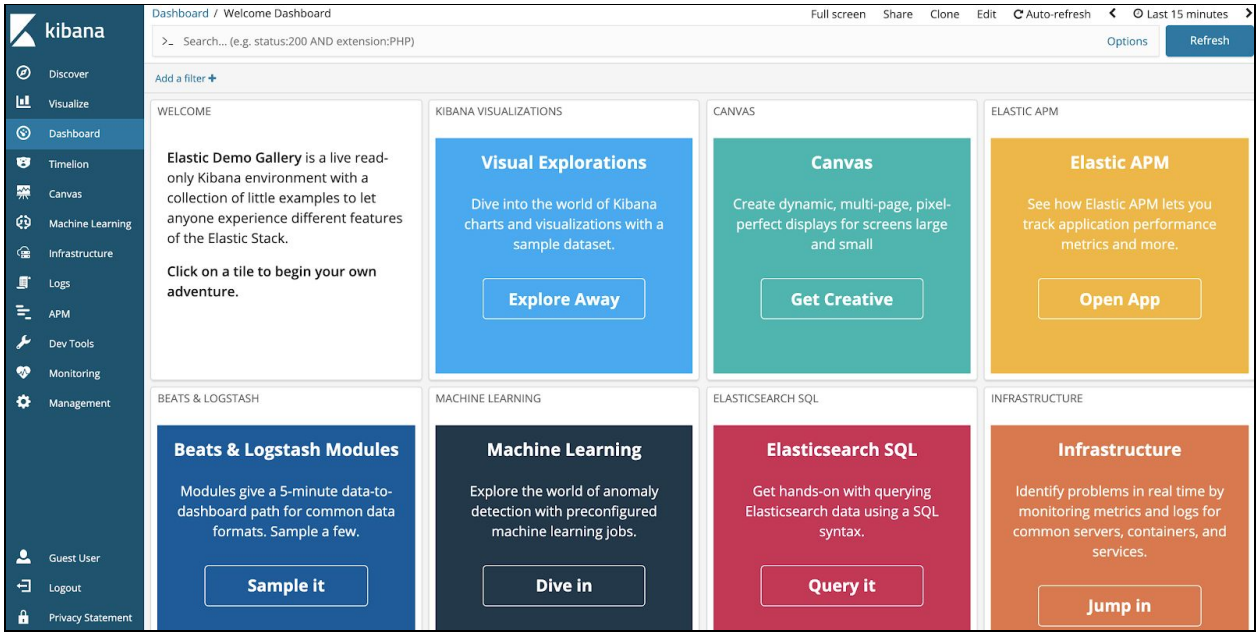


10e. Now we can see that the source of this layer is an aggregate of the logs by location of geopoints. Once again, we now have enough information to completely reconstruct this layer in our own Map. This is a great exercise to try out! Let's move on though!



11. The authors of this lab would be remiss if we did not share that there are some amazing sample dashboards at your fingertips at demo.elastic.co. You can navigate through and see all sorts of amazing visualizations at this location once again go shopping - now that you have the tools to examine and reproduce on your own; the possibilities are endless! There are other great examples of capabilities which we hardly touch on but are great to try out - for example,

examples of machine learning.



12. Congratulations! You are done with this section of the lab! Now, let's move on and learn how to start building some of these awesome visualizations!

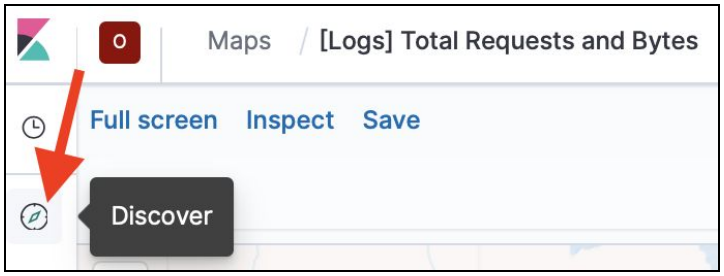
Lab 3: Discovering Data

We looked at dashboards and visualizations in section 3. Those tools serve their purpose. Want to monitor how things are going? Excellent, let's plot the trends. Need a rough overview of where unusual spikes are happening? Searching for geographic distribution of activities? These types of questions can be answered conveniently with graphics.

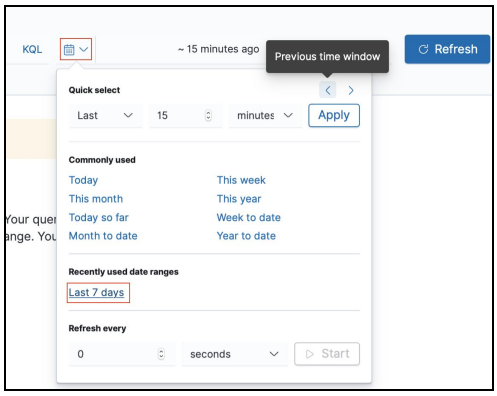
However, sometimes you just need a quick ad-hoc question answered. One way to access time series data in Elasticsearch is to post a query via a client or by using the DevTool app within Kibana. However, Kibana ships with a **Discover** app which makes searching for data easy.

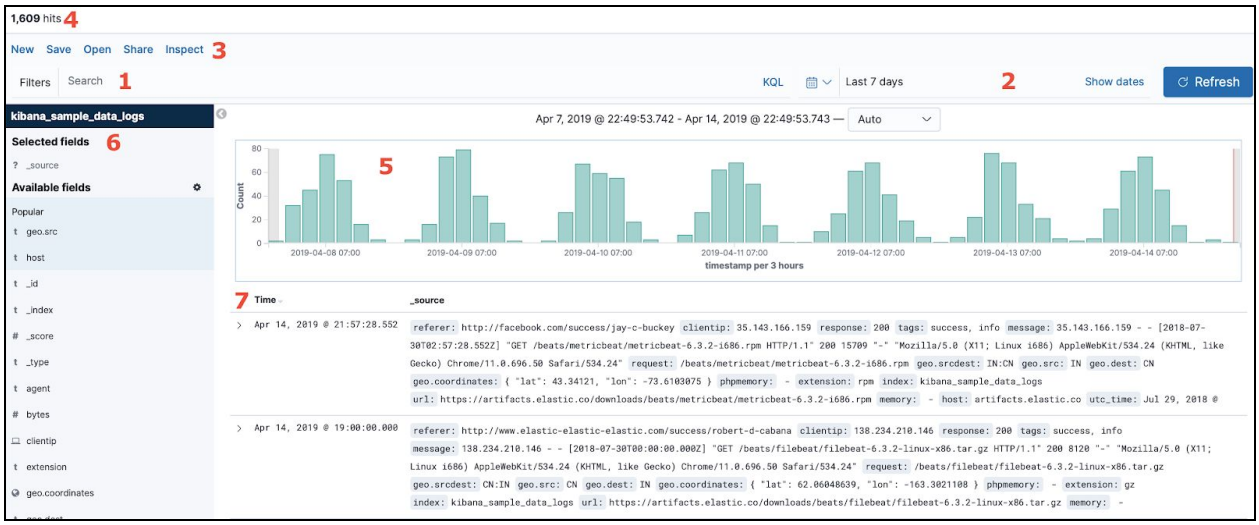
In this section, we are going to get used to looking at data within the discover app and then understand how we can ask questions of our data and get the answers really quickly.

1a. Click on Discover tab on left nav.



1a. Here is the dashboard. You might need to change time to **Last 7 days** in the upper right corner in order to see sample logs; we'll go over the time picker in a little bit. This is a common troubleshooting issue. Let's look at all the parts to this view.



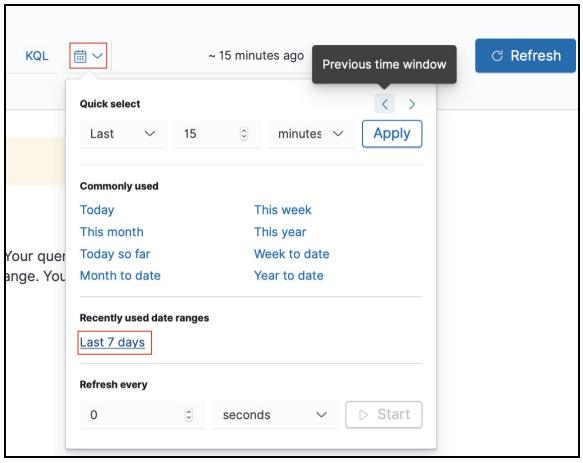


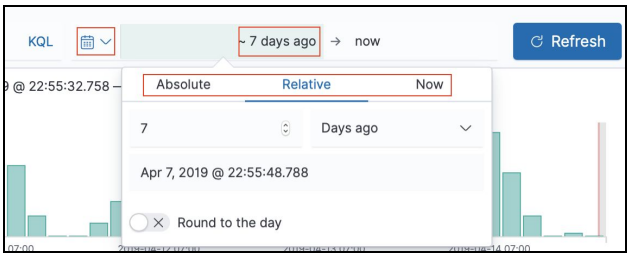
1. The **search box** - enables you to **filter** across all of the logs which are part of the kibana_sample_data_logs index pattern (an index pattern is a set of indexes with a shared naming convention - e.g. logs-* would be a typical name and all indices which start with "logs" would be included

2. The **date picker** - there are several ways to change dates. It's also possible to configure the screen to automatically refresh every x seconds
3. **File and viewing options** - searches, dashboards, visualizations can all be saved and opened later (saving searches and other visuals will be shown later)
4. Number of **records** - records render as a result of for provided filter and time range
5. Quick chart - graphical view of **document count** over time
6. **Fields** - document fields for a given index pattern can be searched and also added to the discovery screen (this will be shown later). In this case, the index pattern **kibana_sample_data_logs** is the index pattern we are viewing, as seen above the fields. **Note:** if you are seeing a different index pattern, be sure to click on it and select **kibana_sample_data_logs**.
7. **Raw documents** - these are ingested into the kibana_sample_data_logs index pattern; you can view these in table or JSON format. Documents render descending in time (unless sorted which will be shown later)

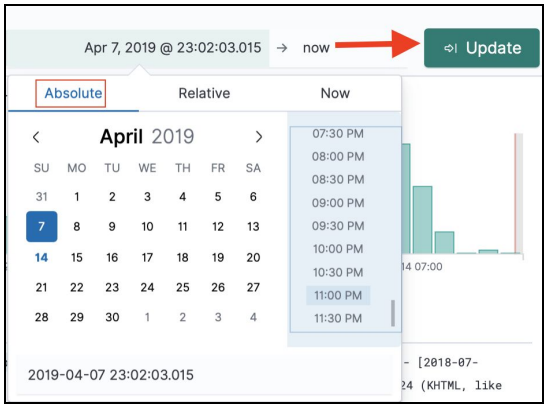
2a. Next, we are going to understand filtering time within Kibana. Recall earlier we clicked on the icon which brings us to recent and commonly used.

Click on the **actual time** (Last 7 days in this case) itself instead in the upper right corner, click on the time label again, and notice all the options. The widget immediately opens up to **Relative**.

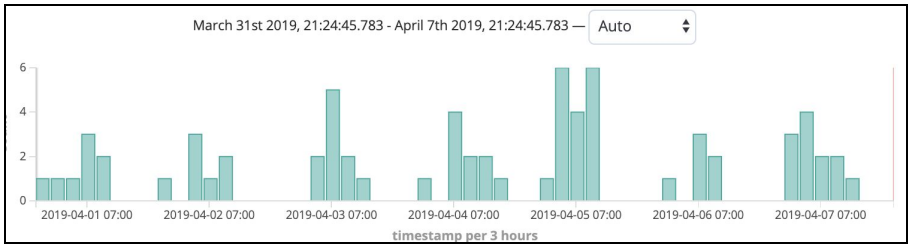




2b. Go ahead and click on **Absolute** and observe the traditional From and To tools. Try the **date pickers** out and click **Update**

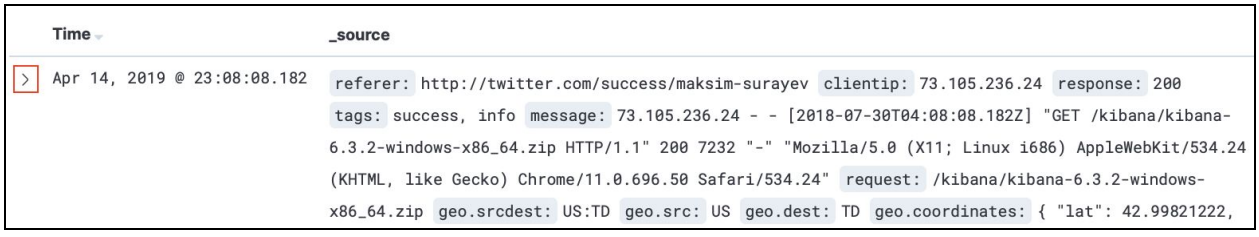


2c. You can also opt to drill down into a time range in the graph by clicking anywhere inside the graphic, dragging the mouse to the left or right, and the releasing mouse.



2d. Go ahead and click the back button on your browser until the time picker is back to Last 7 days (as you probably know by now you can click into the Quick Picker to for this!).

3a. We are going to now look at documents a bit. Pick the first document and click the icon to the right of it. This will expand the document.



3b. By default the view will open up in Table view, which is nice in order to summarize fields. However, for thoroughness, let's take a look at the document in JSON format by clicking JSON.

3c. As far as the sample log itself, we can see that it's a typical log with a timestamp and several other values split out. There are some great training classes which go over how to ingest and parse different kinds of metrics, APM, and logs into Elasticsearch. There is also a log workshop: please ask your instructor about this if interested!

3d. When done, go ahead and click the widget to the right of the log again to collapse.

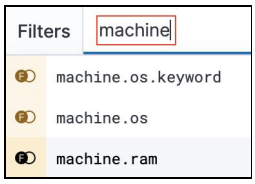
4a. Now, we are going to conduct our first search. Keep in mind every search starts with a question. Our question is going to be, “How many logs are running on Windows?” In order to get started, it’s quite simple: just type in **Windows** in the **Search** box and hit return.

4b. Congratulations! You have just conducts your first terms search! You actually ran a simple term search. You should see a reduction in the number of documents. Please note: there is a really awesome, super thorough class which will go over the basics of search called “Elasticsearch Engineer I”. We highly recommend checking that class out and it’s sequel also. You will get really good at search!

4c. Go ahead and expand a document again and look for the machine.os field. It should look similar to the below (your exact values may vary).

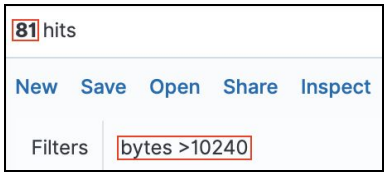
4d. We can actually filter for this field specifically. We are going to leverage Kibana Query Language (KQL) to do that - KQL is already enabled (you are able to turn KQL off and use Lucene style queries, if needed). KQL will enable auto-populate and ease of filling out filters. In the upper right corner of the screen, click on Options and then make sure **Turn on query features** is turn on

Now go to the search bar, start typing “machine”, and observe the field auto-population.

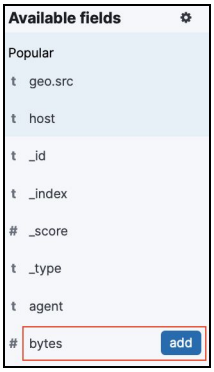


4f. Makes sure **machine.os.keyword : “win xp”** is typed in and hit return. Notice that the fields have now reduced down to 336 hits in this case (your results might vary slightly due to how sample data is deployed).

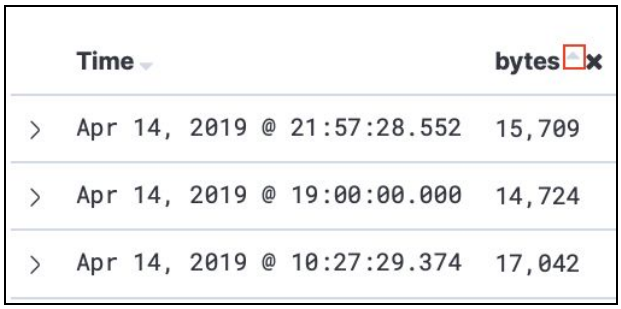
5a. We are going to ask another question now: “**how many documents are larger than 10 kilobytes?**” Type in **bytes > 10240** and hit **enter**. Notice that the number of documents is now down to 81.



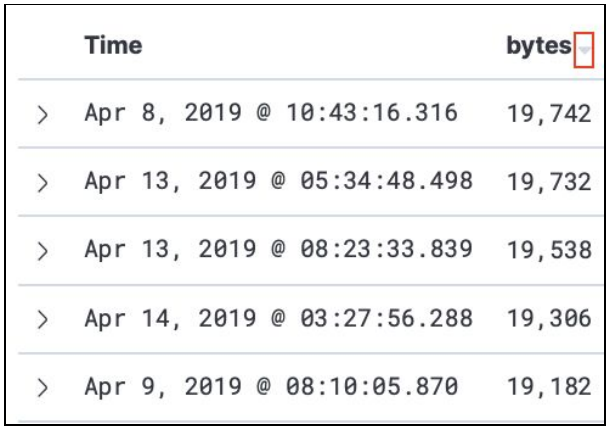
5b. We are going to make the byte fields a little bit more visible. On the left nav, hover over the **bytes** field (this field might be further down the page than this picture), notice the add button which pops up, click **add**.



5c. This results in a bytes column rendering to the right of the time stamp. Now, we are going to sort by that field, similar to how we would in a spreadsheet. Click on the **icon** to the right of the bytes column header. This puts the values in ascending order. We need the order in descending, so click the **icon again**.



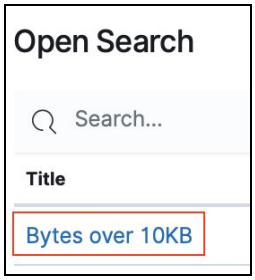
5d. The result should look similar to the below.



	Time	bytes
>	Apr 8, 2019 @ 10:43:16.316	19,742
>	Apr 13, 2019 @ 05:34:48.498	19,732
>	Apr 13, 2019 @ 08:23:33.839	19,538
>	Apr 14, 2019 @ 03:27:56.288	19,306
>	Apr 9, 2019 @ 08:10:05.870	19,182

6a. Now that we've answered our question, we'd like to hold onto all of our work. We can do that by clicking **Save** at the top. Title the "**saved search**" **Bytes over 10 KB**. Then, click **Confirm Save**.

6b. We can now pull this search up anytime we need it by clicking on Open. After you see the new Saved Search created, click on the x at the upper right to close out of the pop-up.



6c. There are other options. We can share this saved search easily by link. Click on **Share**. From here, you can copy the link to the clipboard, for example, in order to share the link. Just about every very view in discovery tab is represented in the URL and can just be copied and shared with someone. If security is enforced, the person who receives the link will need to log in.

6. Congratulations! You are done with this section of the lab! You can now go to the discover tab, pick an index pattern, and search for anything. One thing to note: when you are working with your own dataset, after you upload your data, you will need to define your index as part of

an index pattern (a set of index patterns with the same name - e.g. logs-03-11-*, in order to discover your data. Check out in the documents to learn more! You now have the basic tools, but this is a great walkthrough for your own data:
https://www.elastic.co/guide/en/kibana/current/tutorial-build-dashboard.html

---

Lab 4: Visualizing Data

In this section, we are actually going to build simple visualizations. You will have an opportunity below to build the following:

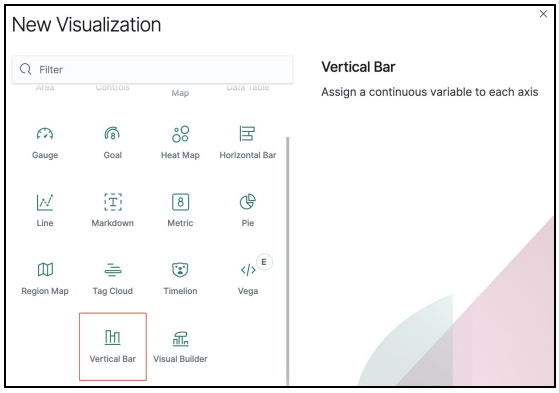
- Vertical bar (required)
- Control (required)
- Metric (required)
- Tag cloud (required)
- Geo (optional)
- Pie chart (optional)
- Table (optional)
- Heat map (optional)

1. If you fall short of time, at least you have a record for later; in the meantime, please try to complete the first four visualizations. For each visualization below, you can create a new visualization by clicking the **Visualize tab** in the left nav as you did in reverse engineering lab above and then clicking the plus sign. Occasionally, you will need to click on the **Visualize** link at the top left, in order to get back to the Visualizations list and click the **plus**. In order to avoid repetition, we will simply say, “now, let’s create a new visualization.”

2. Vertical Bar (Important)

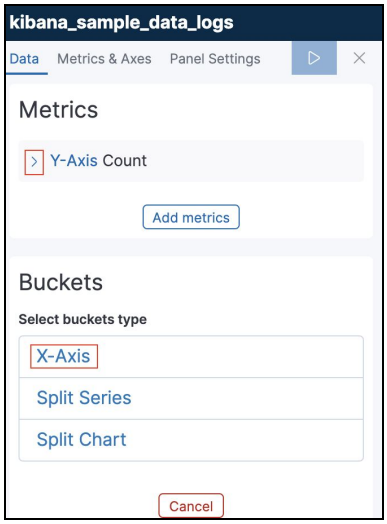
2a. In this section we are going to build out a vertical bar. Why would we build a vertical bar chart? In our case, we would like to ask the question, “how can I visualize the breakdown of count of response codes streamed in time?”.

2b. Now, let's create a new visualization. Select **Vertical Bar**.

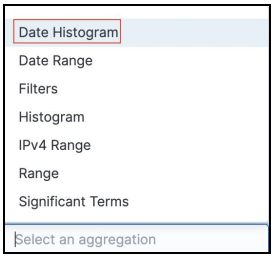


2c. Select the **kibana_sample_data_logs** Index pattern by clicking on it.

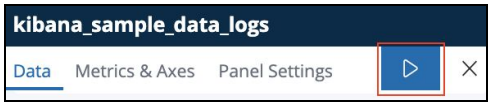
2d. Make sure your time window is Last 7 days (go back to discovery lab if you don't know how to do this). On the left hand side, click on the Metrics Y-Axis collapse **arrow**. Also, click on bucket **X-axis**..

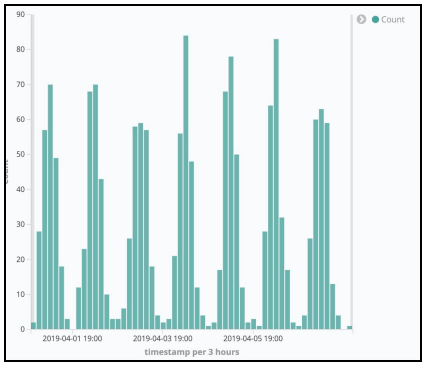


2e. Click on **Aggregation** picklist and select **Date Histogram**.

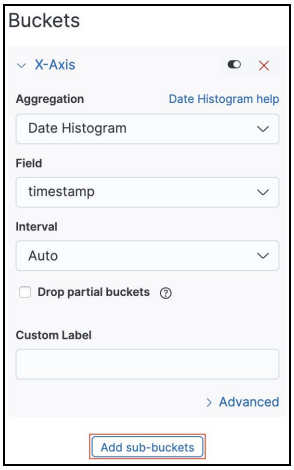


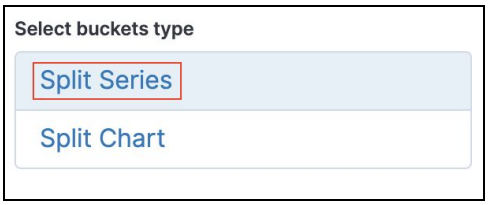
2f. Click on the arrow icon at the top of the left nav to see how the build is coming along.



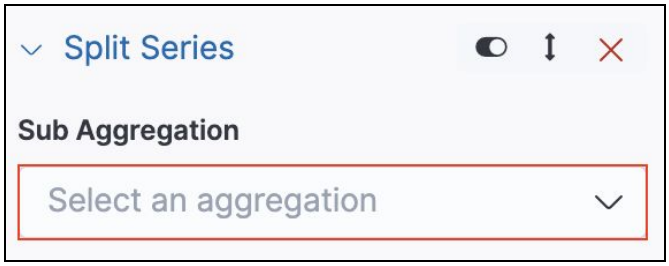


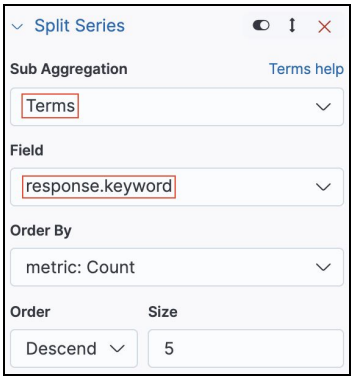
2g. We are now cooking with grease - so far so good! Counts are nice but I'd actually like to see a breakdown by response code. In order to do that, we click on **Add sub-buckets**. Next, click on **Split Series**.



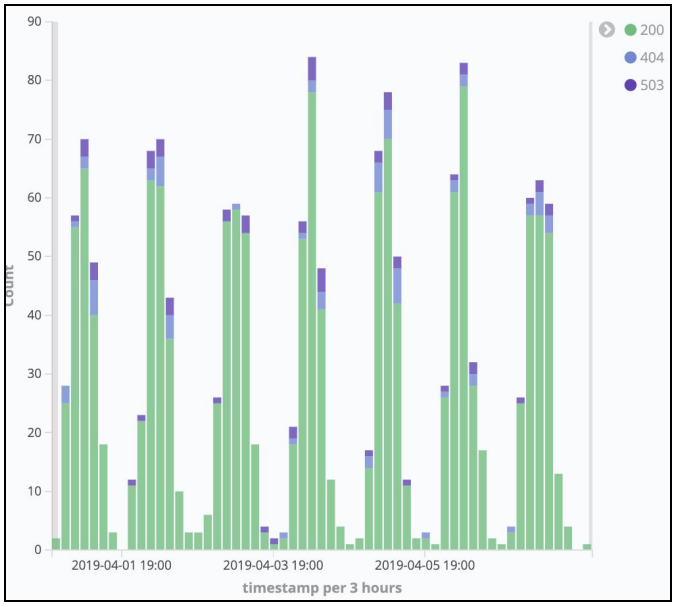


2h. Select **Terms aggregation** and the field **response.keyword .** Basically, this is how we can split up means a count of discrete items over time, which is what we are doing here. The discrete item we are splitting up is the response code. The keyword is required for fields you want to bucket up by aggregation.





2i. Click on the arrow **icon** and notice your chart just light up! We did it!



2j. Click on **Save**

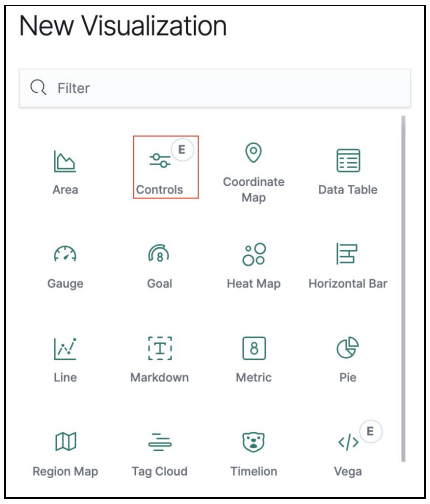
2k. Give it a title. Call it **Requests Over Time By Response Code** and click **Confirm Save**

2l. You are done! Pretty easy right? We just split up response codes over time and got a nice breakdown of the count. There are other visuals which can accomplish the same thing - for example, TSVB (in the advanced workshop section) has a similar concept. In general, for a lot of folks, if you want a fast breakdown over time, the bars are the fastest way to get there.

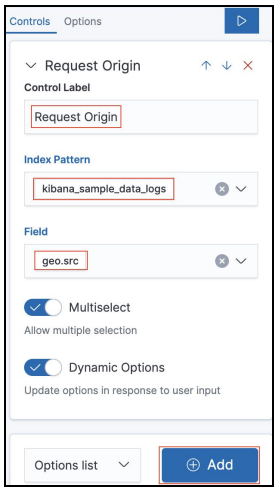
3. Controls (Important)

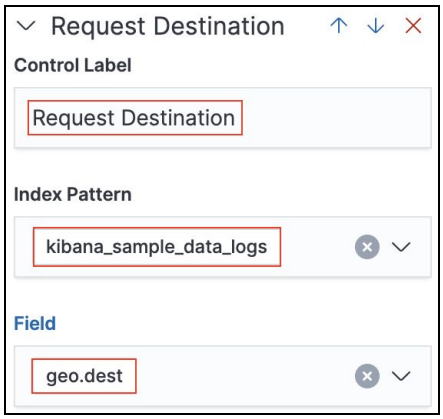
3a. Often, folks who research data want a quick way to filter data logically into buckets or facets, during a search. This type of term-based aggregate search is so common Elastic has built out a Control visualization to filter data quickly on a dashboard. This visualization will answer the question, “How can view all logs with a specific destination and/or source country on the dashboard itself?”

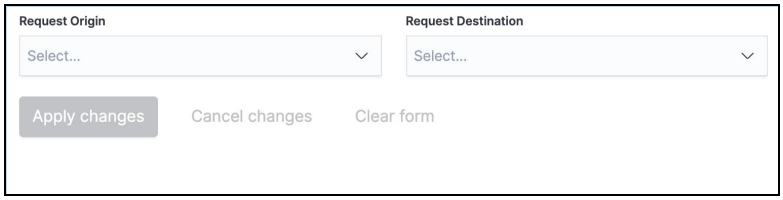
3b. Now, let’s create a new visualization. Select **Controls**.



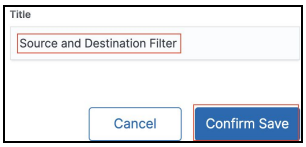
3c. Make sure Options list is selected and click on the **Add** button. Type in **Request Origin** to **Control Label** field. Select **kibana_sample_data_logs** as the **Index Pattern,** and **geo.src** as the **Field**. Then click **Add** again. This will add a new filter below. The last filter was for source, so now we’ll filter by destination. Enter **Request Destination** in **Control Label**, **kibana_sample_data_logs** as the **Index Pattern**, and **geo.dest** as **Field**. Go ahead and click the **render arrow** to test. To test, type in or select values from drop down lists.







3d. Click **Save** at top when satisfied visualization is ready, entitle object **Source and Destination Filter** and click **Confirm Save**.

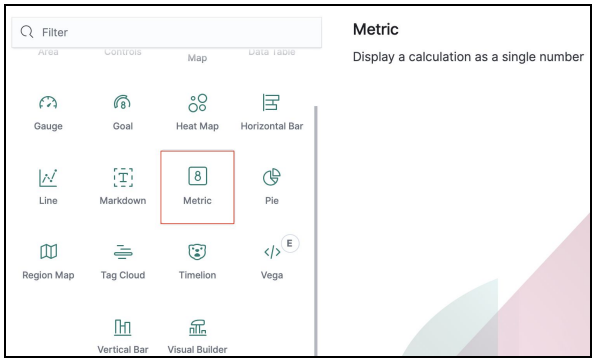


3e. We are done with this visualization which will really come in handy during the dashboard lab. How else can we filter data? We can certainly bucket up data using the vertical bar approach. We can also consider using the Discover tab and save a search, but then we wouldn't be looking at a dashboard. This visualization is popular!

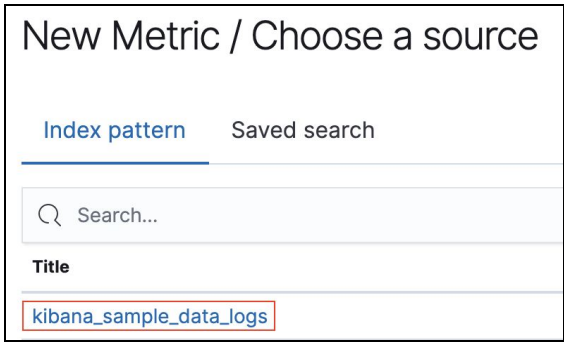
4. Metric (Important)

4a. Moving on, we are now going to think about a visualization which comes in handy for calculating statistical information on the fly across time-series data. And the data sizes we are referring to here could go into the TB's even PB's: Elasticsearch is just that good at running calculations across large datasets. How does this get done? By storing data into memory off-heap (remember Elasticsearch is a Java-based solution) in a distributed way. As a result, calculations and faceting of data can be accomplished at the speed of thought. In our case, here, we will wonder, "how many unique visitors are exploring our website?" This is a simple tally of Unique count. Elasticsearch can do this at lightning speed and with ease. This is a SEARCH problem! Let's get started answering our question.

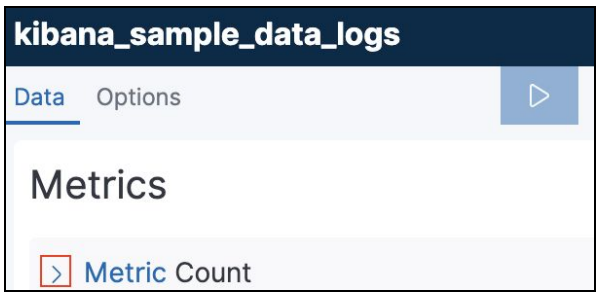
4b. Create a new visualization. Select **Metric**.



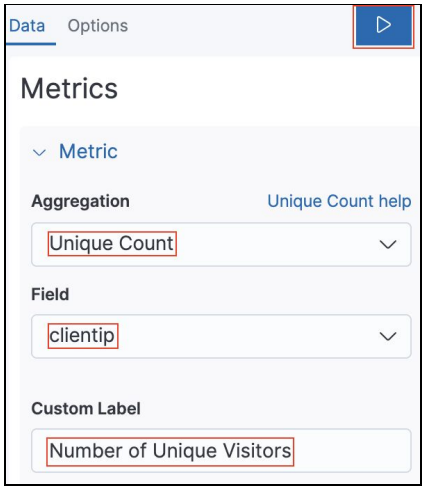
4d. Select **kibana_sample_data_logs** as the **Index pattern**. Note: we could be running this same metric against data in a saved search, instead.



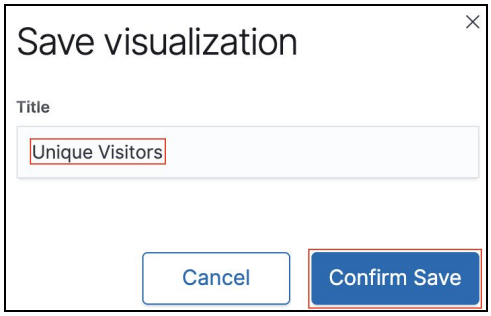
4e. Expand the metric box by clicking **>** .



4f. Select **Unique Count** for **Aggregation** type, but notice all the other aggregation types to choose from in the dropdown! Select **clientip** for the **Field**, and type in **Number of Unique Visitors** for the **Custom Label**. When done, go ahead and click on the blue **arrow** in the upper right and notice the metric you've created; note, you might have to adjust the time range in the upper right in order to have any metrics. That was quick!



4g. Let's go ahead and save this visualization as **Unique Visitors**.

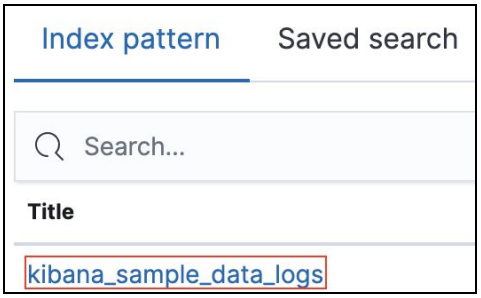


4h. The aggregation we are run above finds the cardinality of the clientip. To understand more about the actual request, you can click on the **Inspect** link at the top and navigate to the request. You can run a search off of the index pattern within DevTools with this request as the body and see the same responses for the given filtered time range.

5. Tag Cloud (Important)

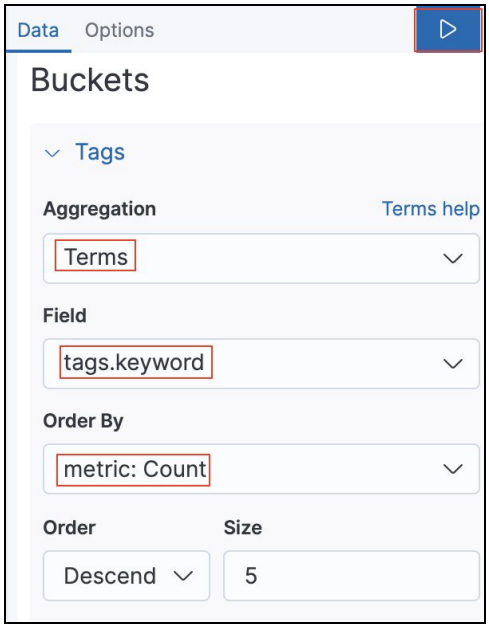
5a. Sometimes, we want a run a count on a number of documents which can be bucketed into different categories. There are lots of ways to show these types of buckets. One such way is to go with the tag cloud which provides a nice way of visually ranking concentration of aggregate. In our lab, we'll ask the question, "How do I representing the popularity of tags within a document?"

5b. Select visualization Tag Cloud to get started and then pick kibana_sample_data_logs as the Index pattern.



5c. Expand the **arrow** to examine the aggregation type. Nothing to change here: we are going to count documents which apply. When done examining options for search, click on the **Tags** link within the **Buckets** options.

5d. Choose **Terms** for the **Aggregation**, **tags.keyword** for the **Field**, and **metric: Count** for **Order By**. Then click the **arrow** to see what you have. There are a lot of really great other options we are skipping over which are documented and which training would cover in depth; this is a cookshow :) Your tag cloud should like something like this.

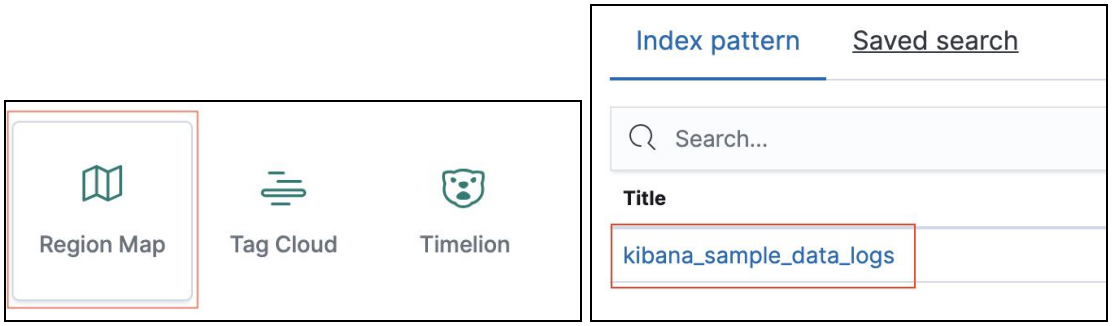


5e. Save this visualization. Call it **Tags**.

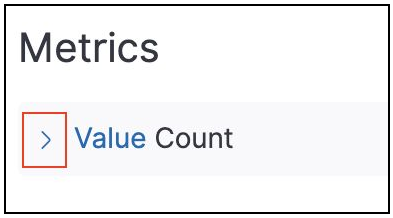
6. Geospatial (Optional)

6a. In this lab, we will create a geospatial visualization that shows unique visitors to our website on a world map. To do this, we will use a **unique count** aggregation of client ip addresses and bucket them by country of origin. Once rendered on the world map, the darker the color of the country, the more unique visitors from that country.

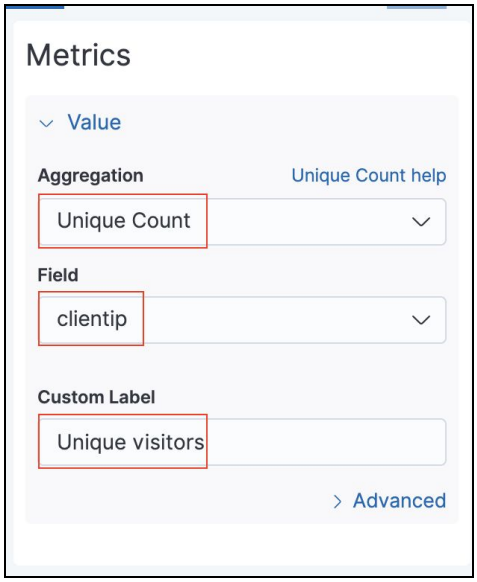
6b. Select visualization **Region Map** to get started and then pick **kibana_sample_data_logs** as the Index pattern.



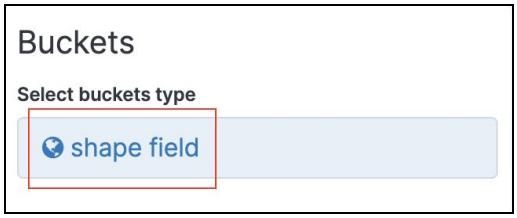
6c. Expand the Metrics **Value** section.



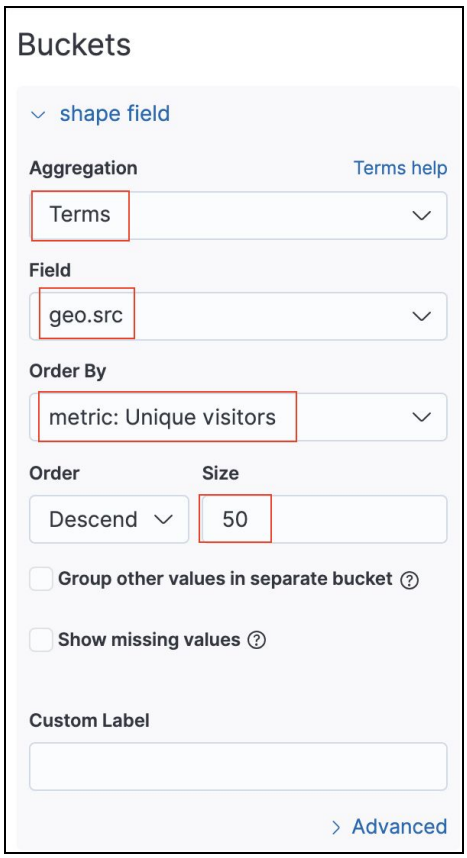
6d. Select **Unique Count** as the type of Aggregation, **clientip** as the Field that we will be aggregating, and enter a **Custom Label** such as “Unique visitors”



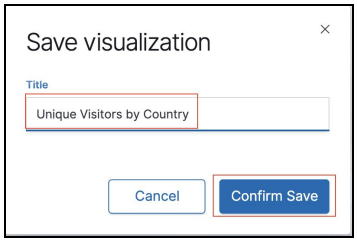
6e. Next we will configure the Bucket Aggregation. In the **Buckets** section expand **shape field.**



6f. Then select **Terms** for the **Aggregation**, **geo.src** (country of origin) as the **Field** and in the **Order By** drop down, select **metric: Unique visitors**. Finally, change the size to **50** and hit the blue render arrow to see the results.



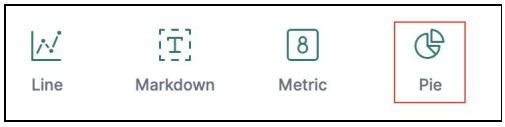
6g. Save this visualization and give it a name such as **Unique Visitors by Country**.

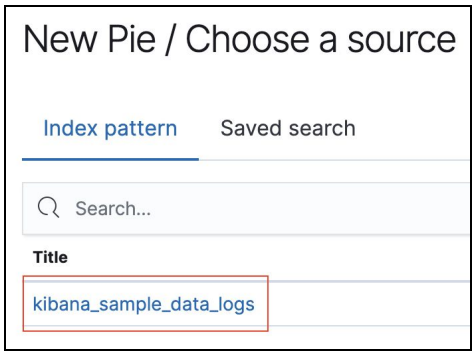


7. Pie Chart (Optional)

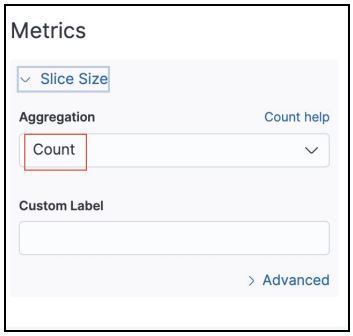
7a. In this lab we will create a pie chart with data from our **kibana_sample_data_logs** index. For example, maybe we are interested in a pie chart view of the operating systems being used to access our website among the top 5 countries.

7b. Start by selecting the **Pie** visualization and then pick **kibana_sample_data_logs** as the Index pattern.



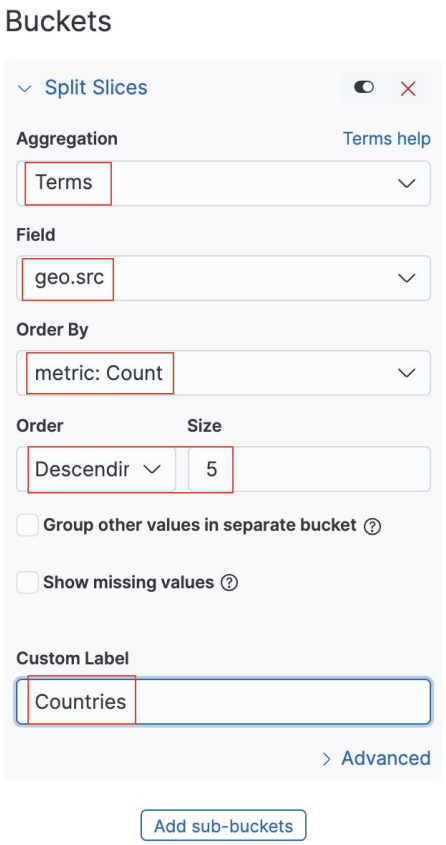


7c. Expand the **Metrics** section and configure it for **Count** aggregation.



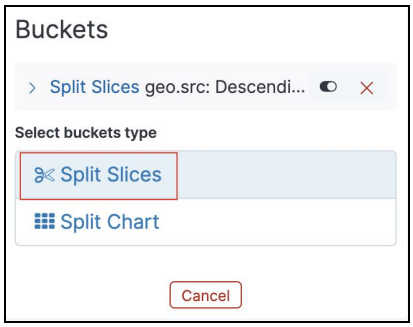
7d. The next step in creating our pie chart is to create the slices by country. To do this we will use the **Split Slices** for the bucket type in the **Buckets** configuration section.

7e. Select a **Terms** aggregation on the **geo.src** field and use **metric: Count** for the **Order By** section. Use **Descending** for the order and set the size to **5** to limit our pie chart to only the top 5 countries. We can label this ring of slices by specifying “Countries” as a custom label. Next, press the blue render arrow to show the results.

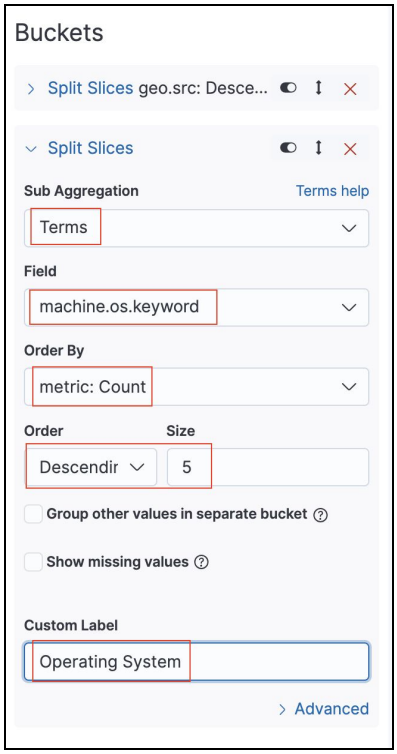


7f. Next we’ll add another ring of slices to our pie chart that will show the operating systems in use for each country. You can collapse the Countries **Split Slices** section that you just created by clicking on its arrow and then clicking on **Add sub-buckets**.

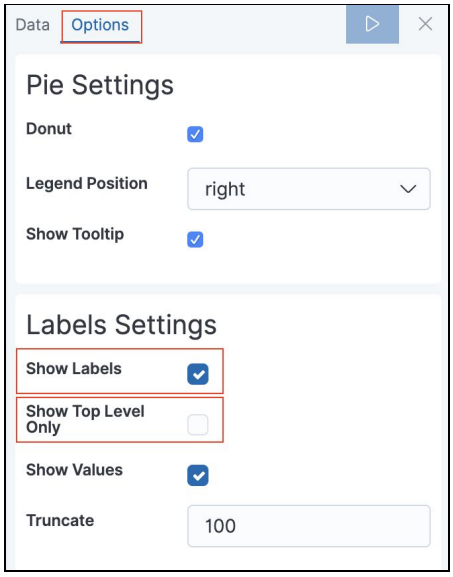
7g. Again select **Split Slices**.



7h. Select a **Terms** aggregation on the **machine.os.keyword** field and use **metric: Count** for the **Order By** section. Use **Descending** for the order and set the size to **5** to limit our pie chart to only the top 5 operating systems. We can label this ring of slices by specifying “Operating System” as a custom label. Next, press the blue render button to show your results.



7i. We can configure several options for our pie chart. Select **Options**, select **Show Labels** and unselect **Show Top Level Only** to label each slice in both the Countries and Operating System rings in the pie chart. Next, press the blue render button to see the results.

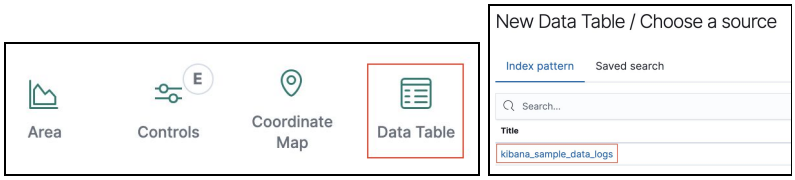


7j. Finally, save your new pie chart visualization and give it a title such as “OS by Country”.

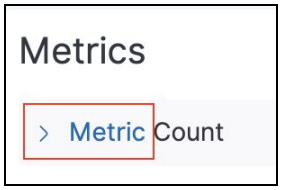
8. Data Table (Optional)

8a. In this lab we will create a table with data from our **kibana_sample_data_logs** index. For example, maybe we are interested in a tabular view of the top 20 countries accessing our website and we’d like to show how many unique visitors accessed the site from each country as well as the total data volume transferred in bytes to each country.

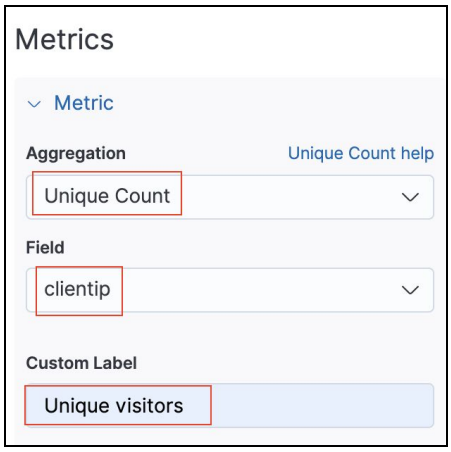
8b. Start by selecting the **Data Table** visualization and then pick **kibana_sample_data_logs** as the Index pattern.



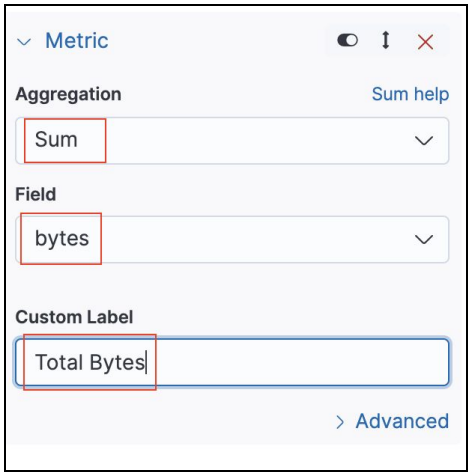
8c. Expand the **Metrics** section and configure it for **Unique Count** aggregation on the **clientip** field to show the number of unique visitors to the website. This will be one column in our table. Finally, provide a heading for the column by entering a custom label such as “Unique visitors”. Press the blue render button to see your results.



=

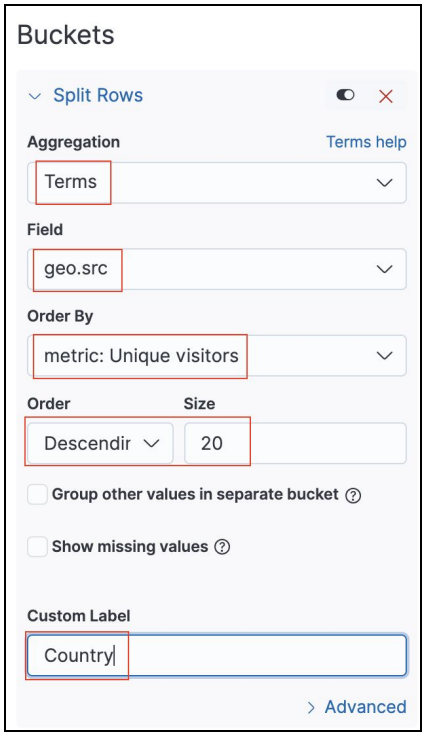


8d. Next click on **Add metrics** to add another column in the table. For this column we will use the **Sum** aggregation on the **bytes** field to show total bytes transferred. Again, we can provide a column heading by providing a custom label such as “Total Bytes”. Render your results.

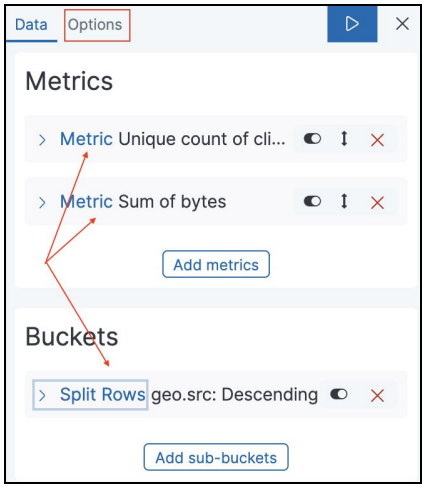


8e. The next step in creating our table is to create the rows of the table which will show unique visitors and bytes transferred by country. To do this we will use the **Split Rows** for the bucket type in the **Buckets** configuration section.

8f. Select a **Terms** aggregation on the **geo.src** field and use **metric: Unique visitors** for the **Order By** section. Use **Descending** to order the rows from most unique visitors to least and set the size to **20** to limit our table to only the top 20 countries. Again, we can set the column heading by specifying “Country” as a custom label. Press the blue render arrow to see the results.

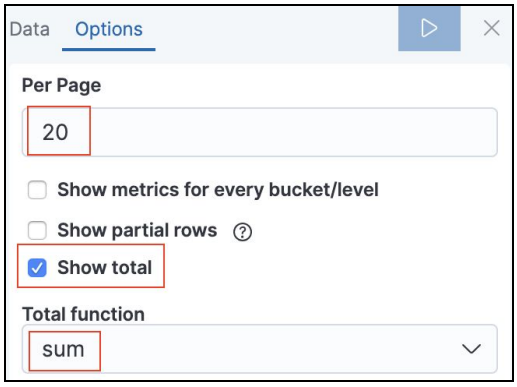


8g. As a last step, we will configure a few **Options** for this table.

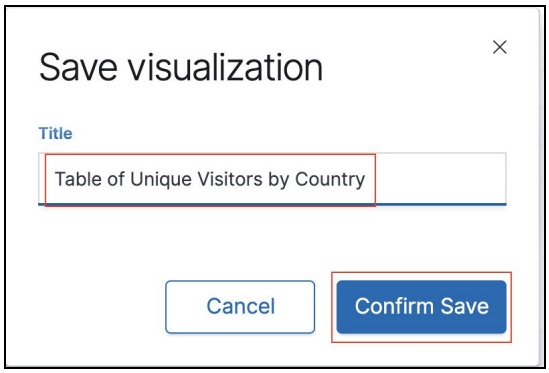


Note that in this screenshot, the Metrics and Buckets configurations have been collapsed by clicking the Metric and Split Rows section as indicated by the red arrows.

8j. In the **Options** section, set the number of rows **Per Page** to 20 to show the entire table. Select **Show Total** to show the total visitors and total number of bytes transferred at the bottom of the table. Select **Sum** as the Total function.



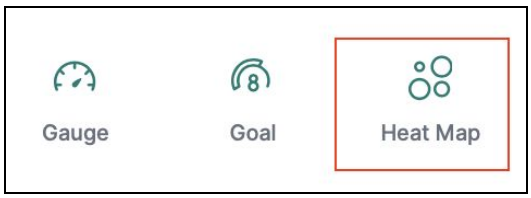
8k. Save your data table visualization and give it a name such as “Table of Unique Visitors by Country”. Remember to click **Confirm Save**.



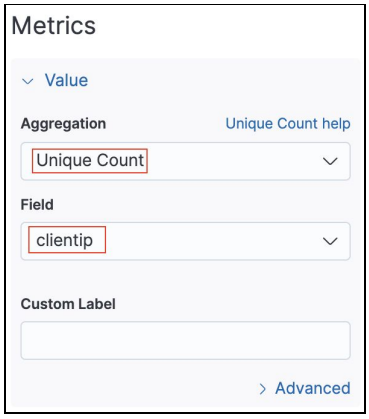
9. Heat Map(Optional)

9a. For this next visualization, we are going to ask the question, “how can I cross-reference overlapping buckets of results?” In our example, we can compare client visits per country versus hours of the day, in order to understand peak traffic during the day by region. We will do this with a Heat Map.

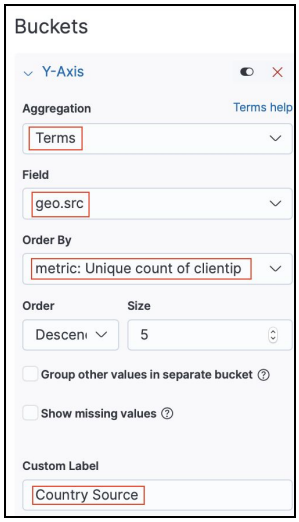
9b. Select the **Heat Map** visualization option and the **kibana_sample_data_logs** index pattern .

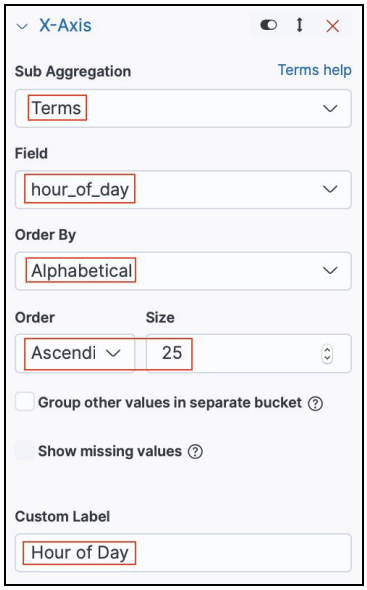


9c. We going to identify Unique Counts of clientips.

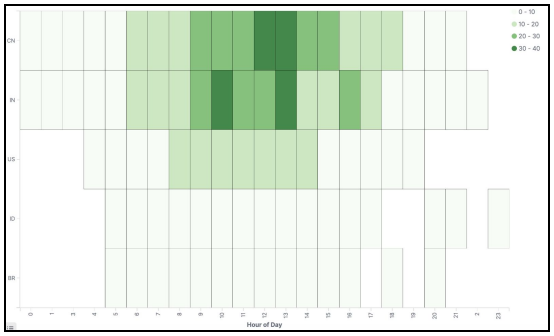


9d. And then we'll categorize those unique counts by source location and hour of the day.





9e. After clicking the arrow in order to render the visualization, you should see something like the following. Go ahead and **Save** your visualization.



9f. Sometimes, looking back at the source document on the discovery tab can help to remind everything which is being done during the aggregate analysis; alternatively, you can click on **Inspect** at the top of the visualization screen and study the rendered request in JSON format, much like we did in the reverse engineering lab. See if you can find your way to the request again! Again, great to be able to leverage that request some day in your own application, as needed!

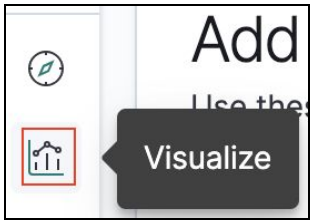
6. Congratulations! You are done with this section of the lab!

Lab 5: Building Out Dashboards

In this section, we'll spend some time placing the visualizations you built as well as the saved search into a new dashboard.

Dashboards are awesome! They help you to easily cross-correlate data visually across multiple data sources within the same time frame. Be able to do this opens up a world of possibilities which operations centers use on live monitors, today. Other organizations go so far as to build weekly alerts which ship out pre-built PDF's of selected dashboards. Some folks merely find dashboards a great place as a starting point for investigating an issue or monitoring how KPI's or some other metrics are doing. Let's get started….

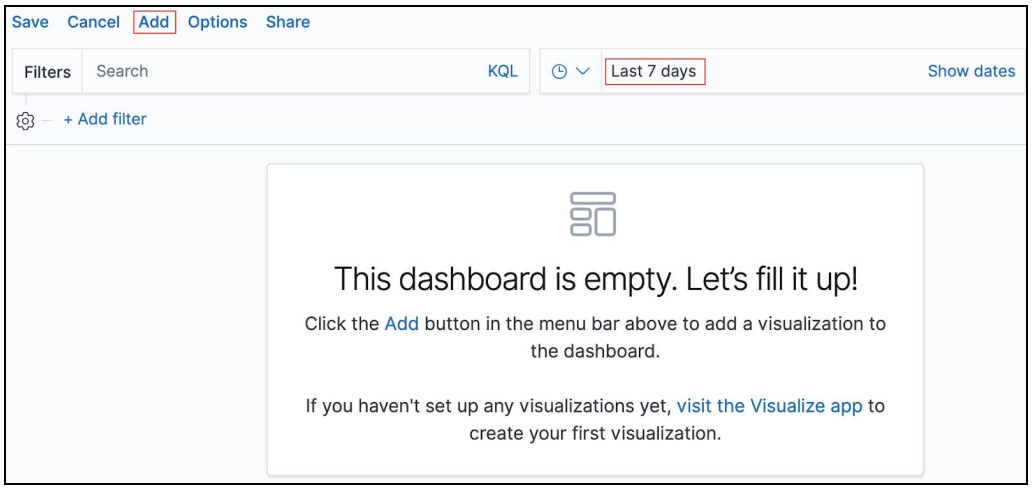
1. In review, you should have at least four visualizations built. You can actually see these on the Visualization list by clicking on **Visualize** and scrolling to the bottom of the list.



2. We are going to take these bottom four which you just built and make a dashboard out of them. First, click on **Dashboard** on the left nav. If the shipped logs dashboard is still showing, you can click on **Dashboard** in the breadcrumb at the top of the page to see the list of dashboards. There should only be one dashboard listed - [Logs] Web Traffic.

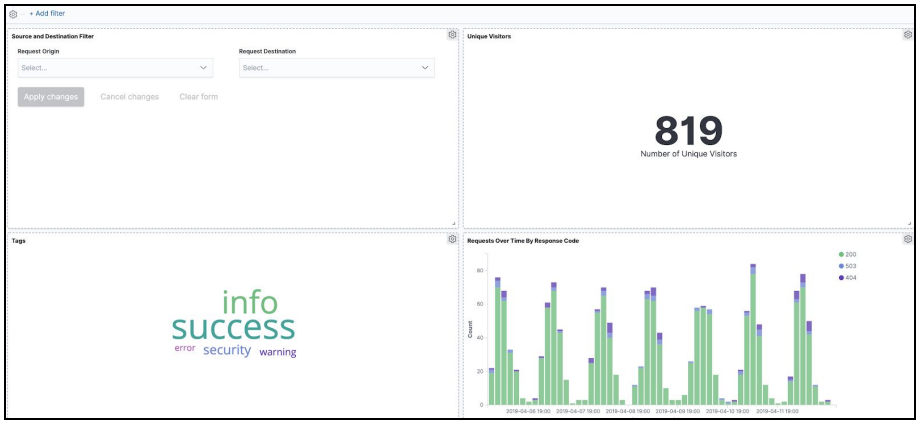
3. Click on **Create New Dashboard** to start off our new screen.

4. The new dashboard should have a default time of **Last 7 days** and should be empty: let's fill up the screen! Click on **Add** in the middle of the screen or in the top left corner. Change the time, if needed; the key is to be able to see log data render within the visualizations after adding.

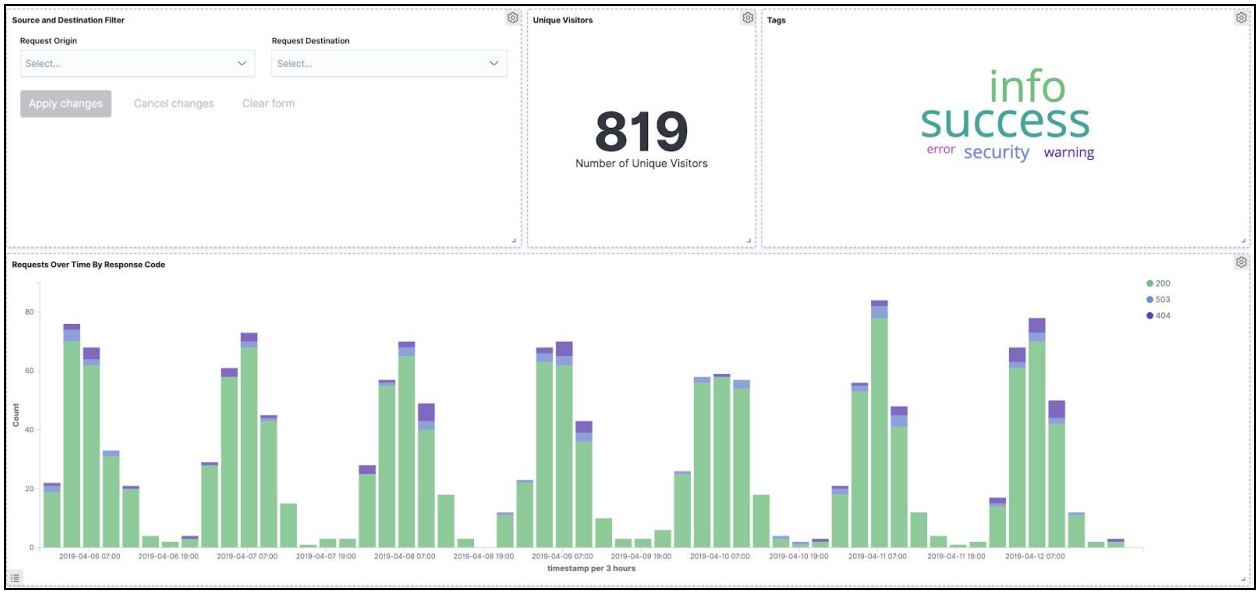


5. The Add Panels screen should be showing. Scroll to the bottom of the list and click on the four visualizations to add them. Start with Source and Destination Filter and go downward. Notice as you click that the visualizations are being populated onto the dashboard in the background. When you are done, **X** out of the **Add Panels** console.

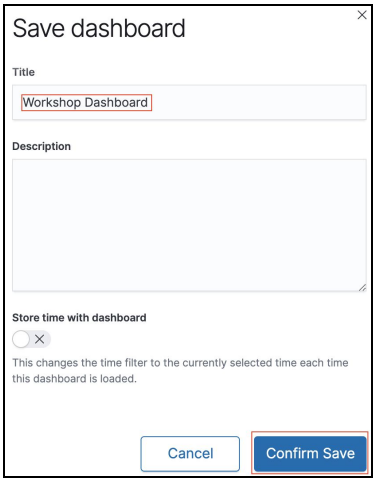
6. Your screen should look similar to the below. Notice that you can reshape and drag and drop the position of each visualization.



7 Here is the final dashboard as rearranged:



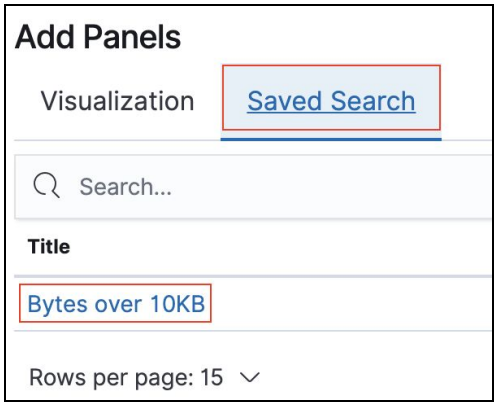
8. We are done - easy right? Definitely. That said we could have put some of the other visualizations onto the screen. There are other features we could add which we won't in this lab - like add markup language to link to another dashboard. Remember: all dashboards are accessible by merely copying the link and sharing. The only left to do with this work is save it. Click on **Save** at the top, Give your dashboard a **title**, and click **Confirm Save**.



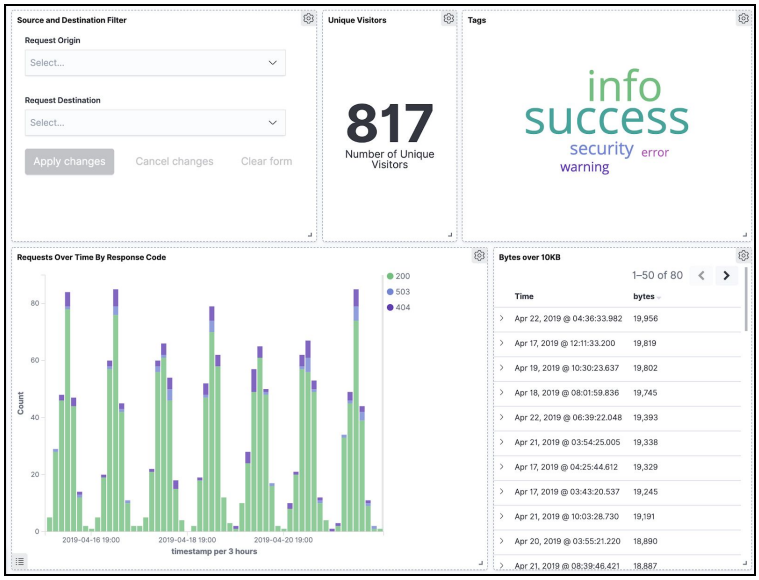
9. **Saved Searches (Optional)**

9a. In this section, we are going to take the Saved Search called **Bytes over 10 K** we created in the Reverse Engineering lab above and add it to the dashboard. If you missed the saved search, please do check it out above! It's really easy to filter your index or index pattern in the Discover Tab and then save the search for future use. In this case, that use is cross-correlating the Saved Search with other visualizations.

9b. To start, put your dashboard back into **Edit** mode and then click on **Add** again. Next, click on the Saved Search Tab and select your Saved Search.



9c. Here is the final result after squishing the visual to share the bottom half the dashboard:



6. Congratulations! You are done with this section of the lab!

Lab 6: Alerting

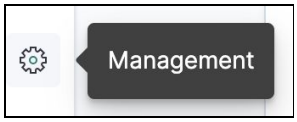
In this section, we are going to try out an alert. In order for this lab to work, your **email address should be whitelisted**, as guided in the first lab where you built out a cluster and set up your profile on ElasticSearch Service.

Well, why would we discuss alerting in a visualization workshop? Our experience has been that users who want to learn how to build visualizations would also like to be able to configure alerts. Often times, these very same teams are being provided a space for their particular team to do their work with their own set of data which they have access to. As a result, we are all about making sure the visualization user can make an alert.

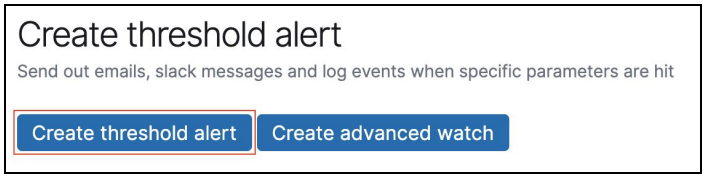
What is an alert? An alert let's you well...notify a user via email, slack, or an API call somewhere else that a certain condition has been met on data stored in and queried from Elasticsearch.

In this exercise, we are going to configure an alert to identify that bytes in a request have exceeded a threshold of 10000. Then, we'll actually trigger the alert.

1. **Important:** Be sure to log in with your **Elastic user**. The designer user has not been given rights to send an alert, yet. Once logged in, go to the **Management** console on the left hand nav and click **Watcher**.

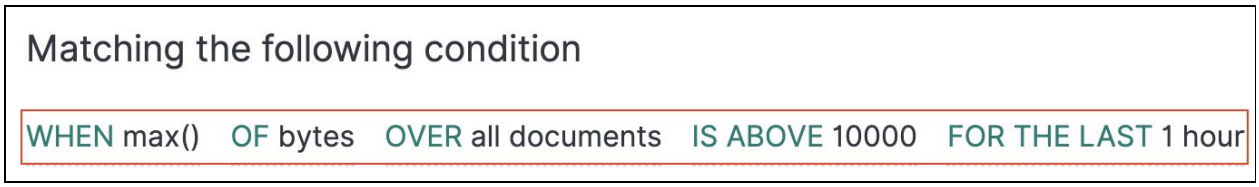


2. There are two kinds of Alerts we can configure with this console. One is more advanced and requires getting deeper into Elasticsearch to fully benefit from. Since learning queries is a separate workshop or class (Elasticsearch Engineer I and II plus certification is a great way to go, btw!), let's go with **Create threshold alert** for now.

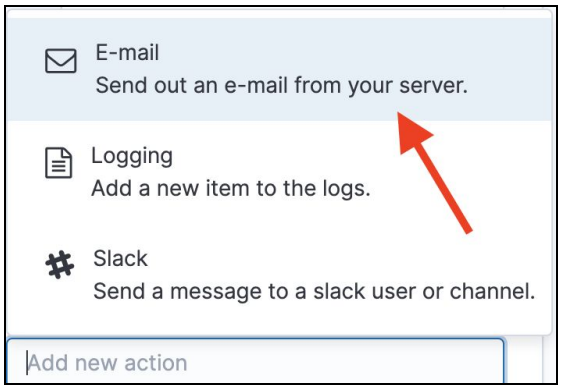


3. Name the alert **bytes_are_high**, select the **kibana_sample_data_logs** as the index we will query, pick **timestamp**, and set the watch to run every **30 seconds**.

4. Further down the alert creation page, click in the right location and modify to look for the **max()** of the **bytes** field (within the logs index) for all values over the last **1 hour**.

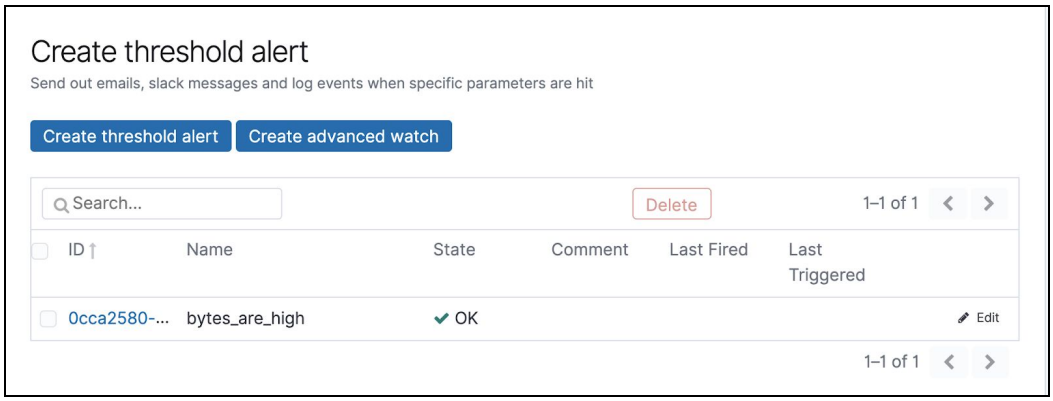


5. Scrolling down further, select **email** as the form of notification.



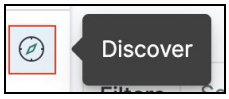
6. Enter your **email address** (please no spam!), appropriate **text** in the body, and click **Save**. If you'd like to test your ability to receive emails, you can click on the test fire button.

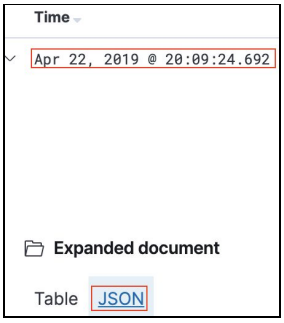
7. Here is the alert console. We can see everything is “OK”.



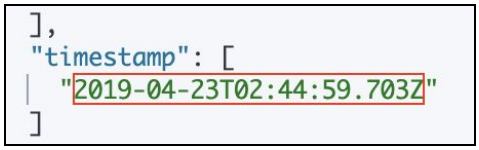
9. Now we need to trigger this alert. Though technically, the sample logs we’ve loaded will have future log data, this data is updated every 15 minutes. If you are like the author, you don’t want to wait 15 minutes. We could write a Python program stream data; we could also steam some data using Filebeat or Logstassh. However, we are going to take a simpler approach and leave more complex approaches to Log labs or up to the user to try out, since streaming data to Elasticsearch is easy but not in scope. Instead, we’ll simply post a new document into the DevTool console within Kibana which will in turn trigger an alert.

First we are going to need the most current time in UTC format. In order to pull that information, we are going to go to the **Discovery** tab and open up the top document (which is also the most recent and has been sent within the last 15 minutes).

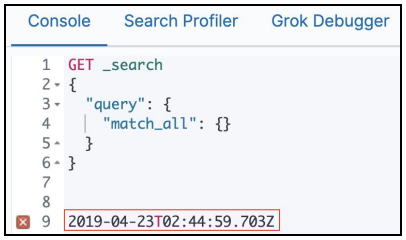




10. **Copy** the value of the **timestamp** within the JSON of this document to the clipboard (make sure you are in the **kibana_sample_data_logs** index).



9. Go to DevTools, click the Button **Get to Work** to get started, and then paste the correctly formatted timestamp value in UTC into the console.



10. Next, copy and paste the following alert and replace the timestamp value with the UTC value. Modify this timestamp as needed to make the time trigger within the 5 minutes window created. After copying/pasting, click send/request button.

Start copying below

```
POST kibana_sample_data_logs/_doc/
{
        "referer" : "http://twitter.com/success/ravish-malhotra",
        "clientip" : "27.253.217.206",
        "response" : "200",
        "tags" : [
          "success",
          "security"
```

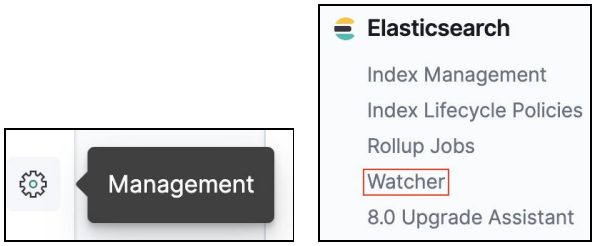
```
      ],
      "message" : """27.253.217.206 - - [2018-08-17T07:55:43.689Z] "GET /kibana/kibana-6.3.2-linux-x86_64.tar.gz HTTP/1.1" 200 9795 "-"
"Mozilla/5.0 (X11; Linux i686) AppleWebKit/534.24 (KHTML, like Gecko) Chrome/11.0.696.50 Safari/534.24""",
      "request" : "/kibana/kibana-6.3.2-linux-x86_64.tar.gz",
      "geo" : {
       "srcdest" : "CN:PK",
       "src" : "CN",
       "dest" : "PK",
       "coordinates" : {
        "lat" : 35.84980556,
        "lon" : -97.41560833

       }
      },
      "phpmemory" : null,
      "extension" : "gz",
      "index" : "kibana_sample_data_logs",
      "url" : "https://artifacts.elastic.co/downloads/kibana/kibana-6.3.2-linux-x86_64.tar.gz",
      "memory" : null,
      "host" : "artifacts.elastic.co",
      "utc_time" : "2018-08-17T07:55:43.689Z",
      "machine" : {
       "ram" : 19327352832,
       "os" : "win xp"
      },
      "agent" : "Mozilla/5.0 (X11; Linux i686) AppleWebKit/534.24 (KHTML, like Gecko) Chrome/11.0.696.50 Safari/534.24",
      "ip" : "27.253.217.206",
      "bytes" : 20000,
      "timestamp" : "2019-04-26T07:55:43.689Z"
 }
```


End copying above

…

11. After you send the request, the document will get inserted into Elasticsearch. At that point, you will be able to go back to the Watcher console within the Management App and see the alert firing. You should also receive the email alert.



11. At this point, you can go back to the Watcher console and disable your alert, unless you'd like to be reminded every so often what a great job you did! (Technically, you can control the repeating of the same alert).

12. Congratulations! You are done with this section of the lab!

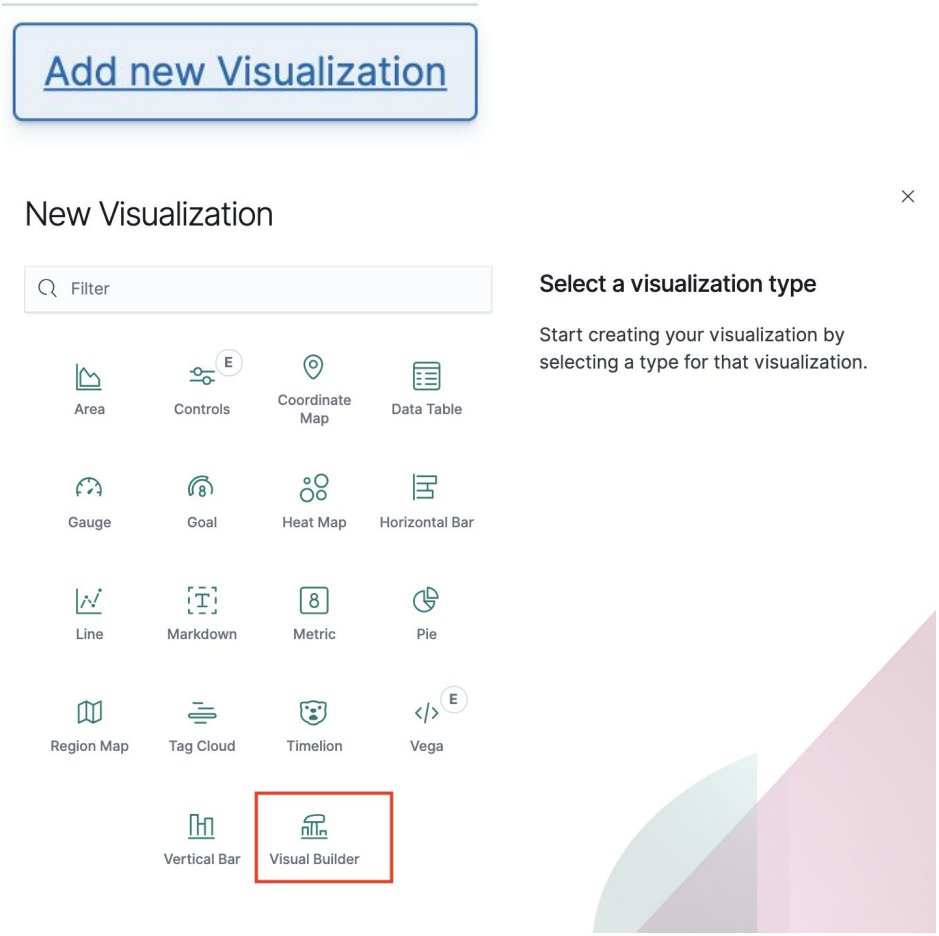
Lab 7: Getting Deeper in Visuals

In this section, you are going to explore some of the more advanced visualizations available in Kibana - Time Series Visual Builder, Timelion, and Vega. These visualizations give you the ability to pull data from multiple indices, apply basic mathematical equations, and write your own custom queries against Elasticsearch and visualize them in whichever way you see fit.

1. Let's start with creating a dashboard. Click on "**Dashboard**" menu link in Kibana and once inside the section click on "**Create New Dashboard**".

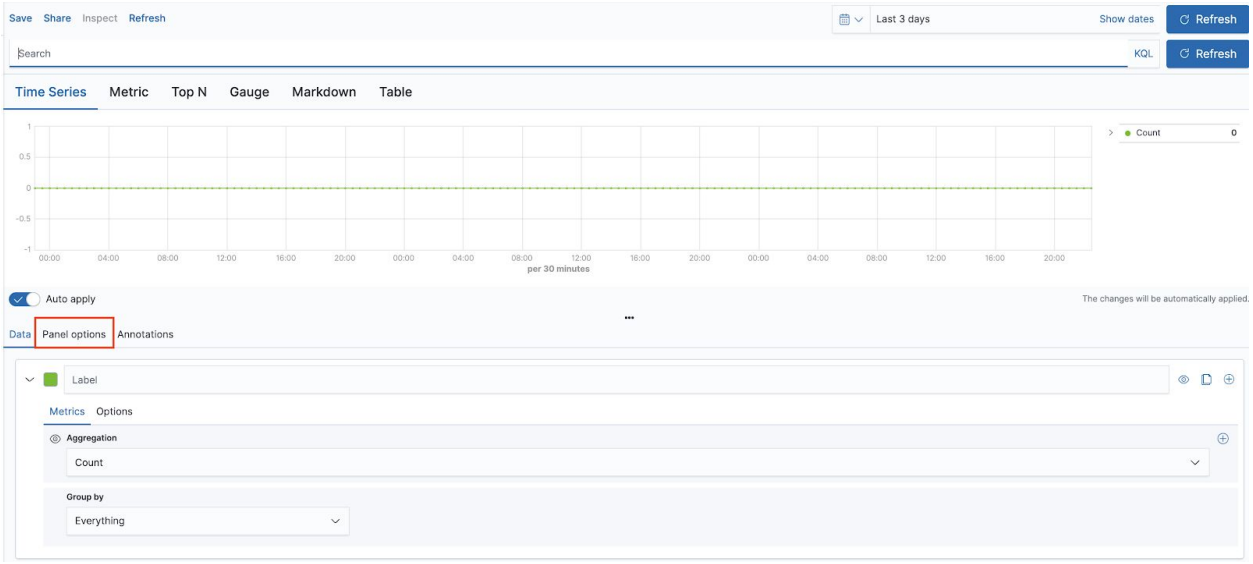
2. Click on “Add” link in the top left corner

2a. Click on “**Add new Visualization**” and select “**Visual Builder**”.

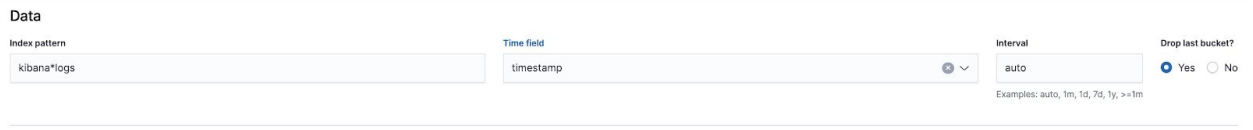


2b. Let’s have a basic view of all the responses that were served from our web server and let’s split them by response code. First we need to define which data we are planning to work with.

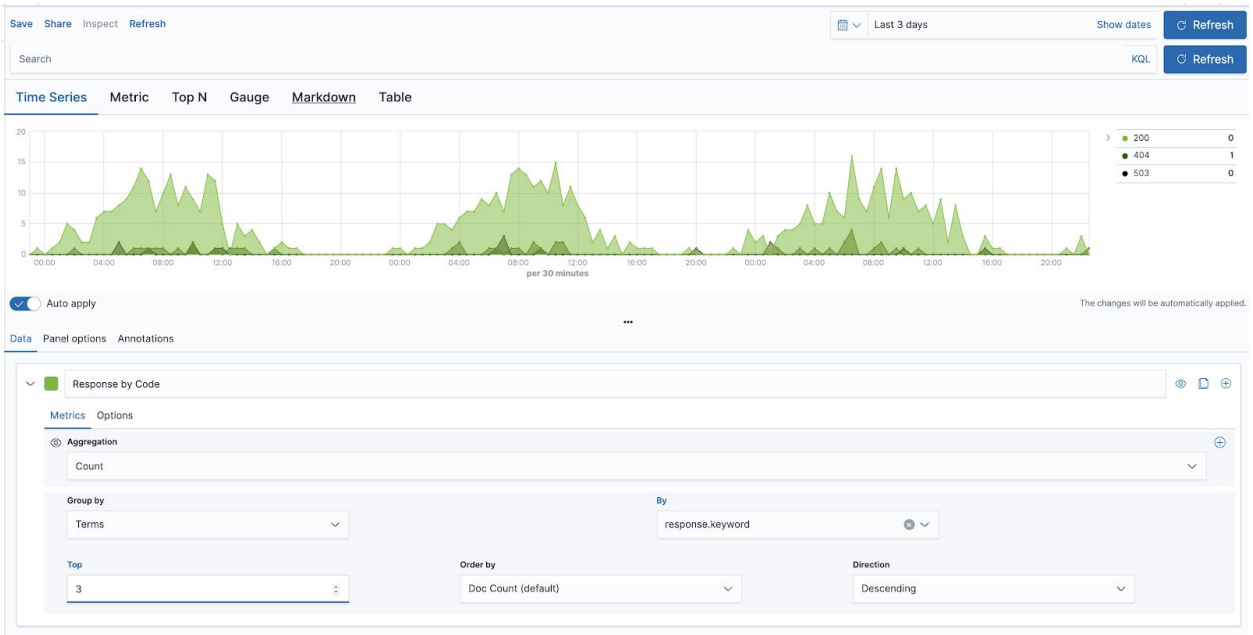
Click on **Panel Options**:



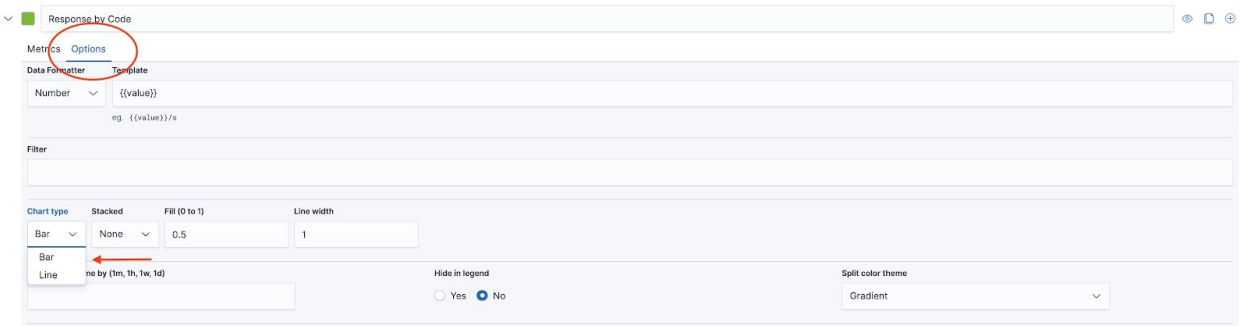
Inside Data section as index specify “**kibana*logs**” and as time field specify “**timestamp**”



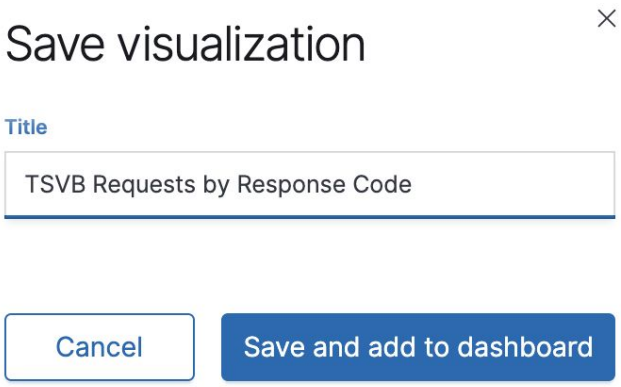
2c. Go back to **Data** section (link next to Panel Options) and let’s do the following manipulations there: In the **Label Section** type in “**Response by Code**”, in the **Aggregation** select **“Count”**, and then Group By “**Terms**”. This will give you the option to select by Field - “**response.keyword**”. And since we have only 3 response codes - in **Top put “3”**.

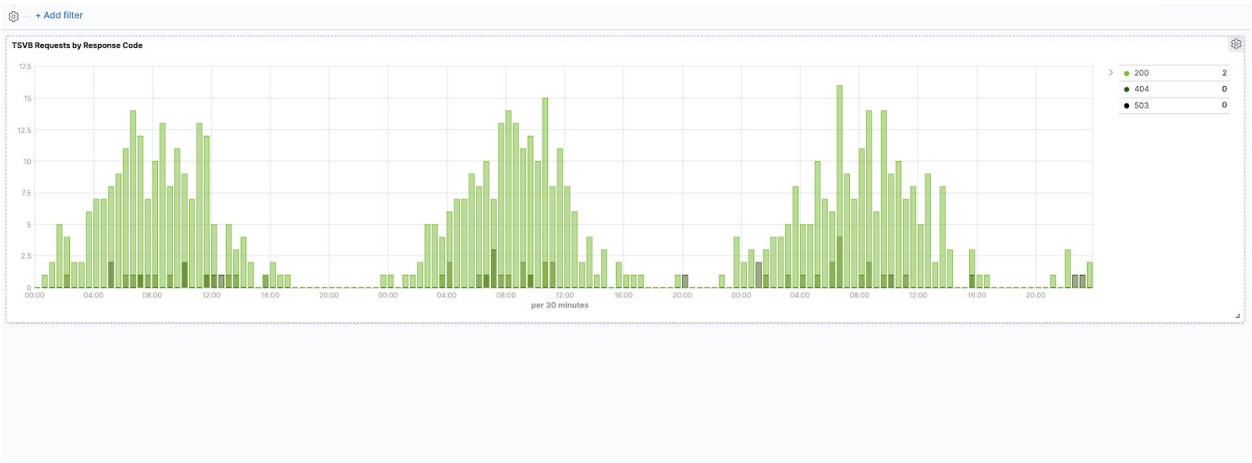


2d. In the “**Options**” section change the Chart Type to “**Bar**”



2d. Save this visualization (“Save” link in the top left corner), give it a name and notice how it appears right away on the Dashboard. (if you are not seeing populated results, make sure to check your time picker).





Extend the visualization across the whole screen to make it look like the image above.

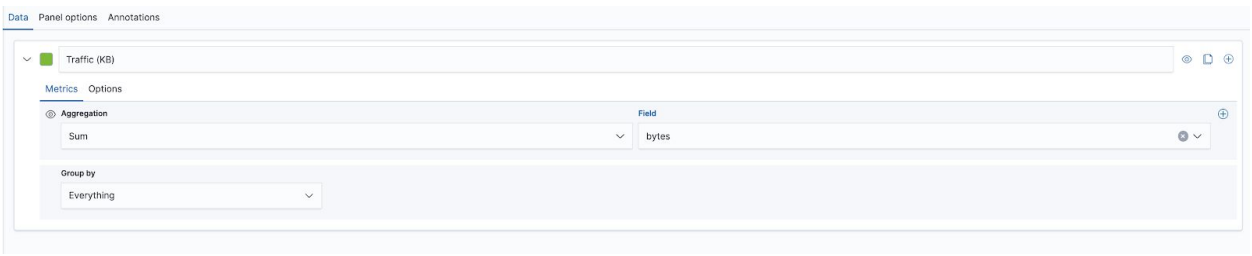
3. In the previous section we have created a simple Visual Builder visualization. In this section we will create a similar visualization, but we will do some math with the data that we have. Let's assume we would like to display the overall traffic of the site by buckets. How do we do that? For each log entry we have a field "bytes" which specifies traffic generated by this particular request/response. So overall traffic per bucket of time is simply the sum of all of those bytes! And if we want to see a value in kilobytes instead of bytes then we simply divide that sum by 1000.
Here is how we can do this in Visual Builder.

3a. On our dashboard similarly to step 2 **add a new Visual Builder** visualization. Once on the visualization specify "**kibana*logs**" as data source and "**timestamp**" as time field (see steps 2 for details on how to do this)

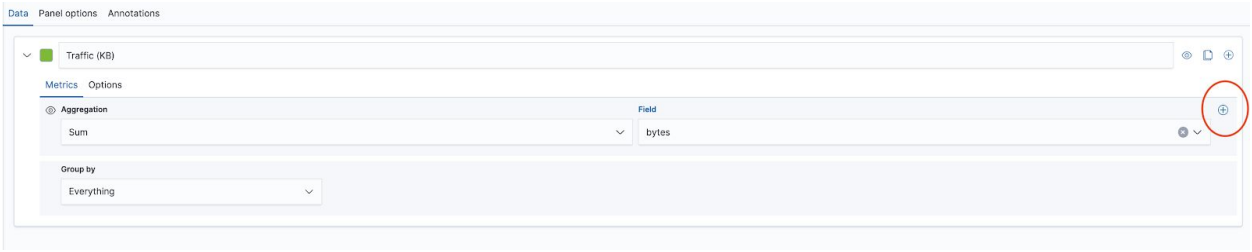
3b. On **Data** set the following values:

Label: Traffic (KB)

Aggregation: Sum, Field: bytes

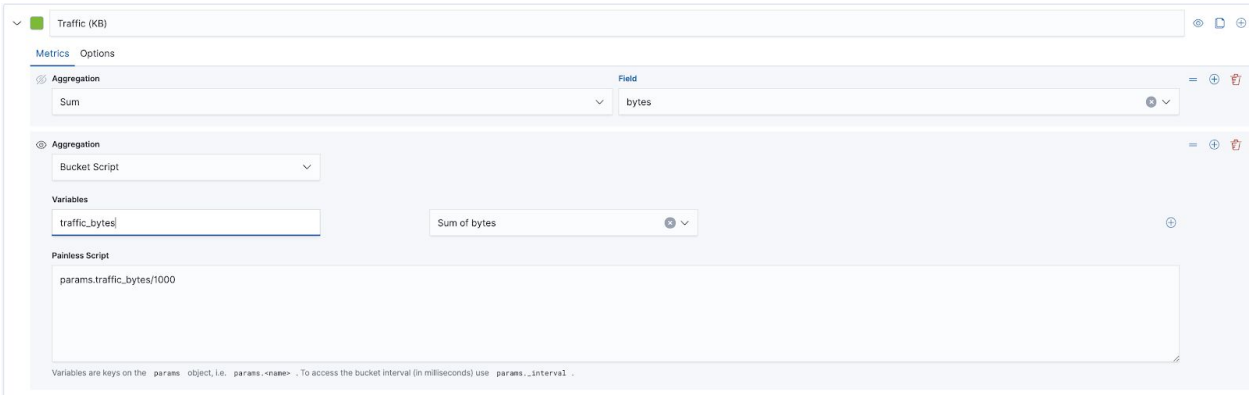


3c. Click on the “**+**” sign above field to add a new value.



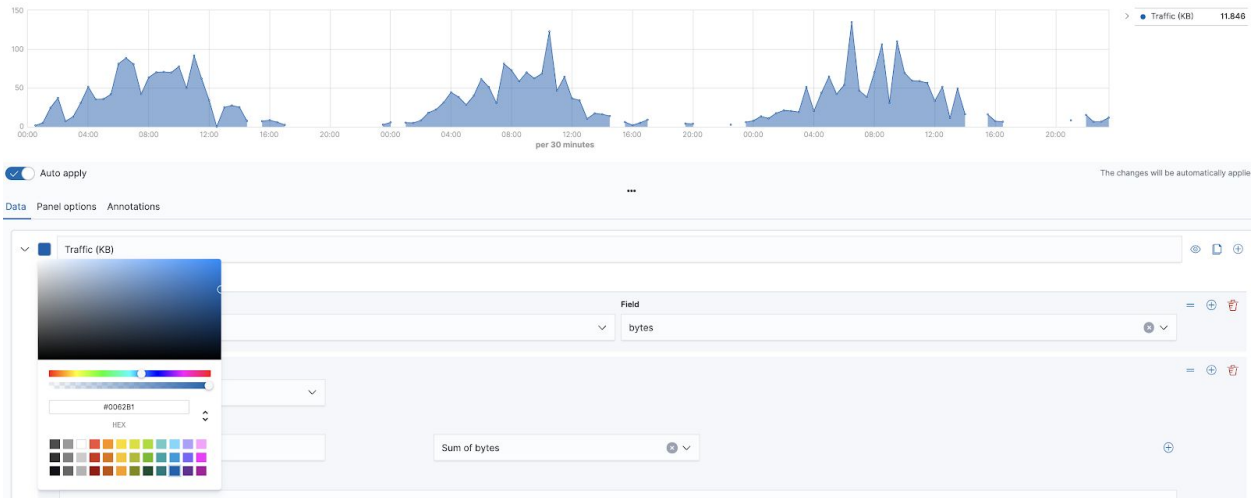
3d. Select:

Aggregation : Bucket Script
Name the variable: **traffic_bytes**, for metric select: **Sum of bytes**
In the painless script type in: **params.traffic_bytes/1000**

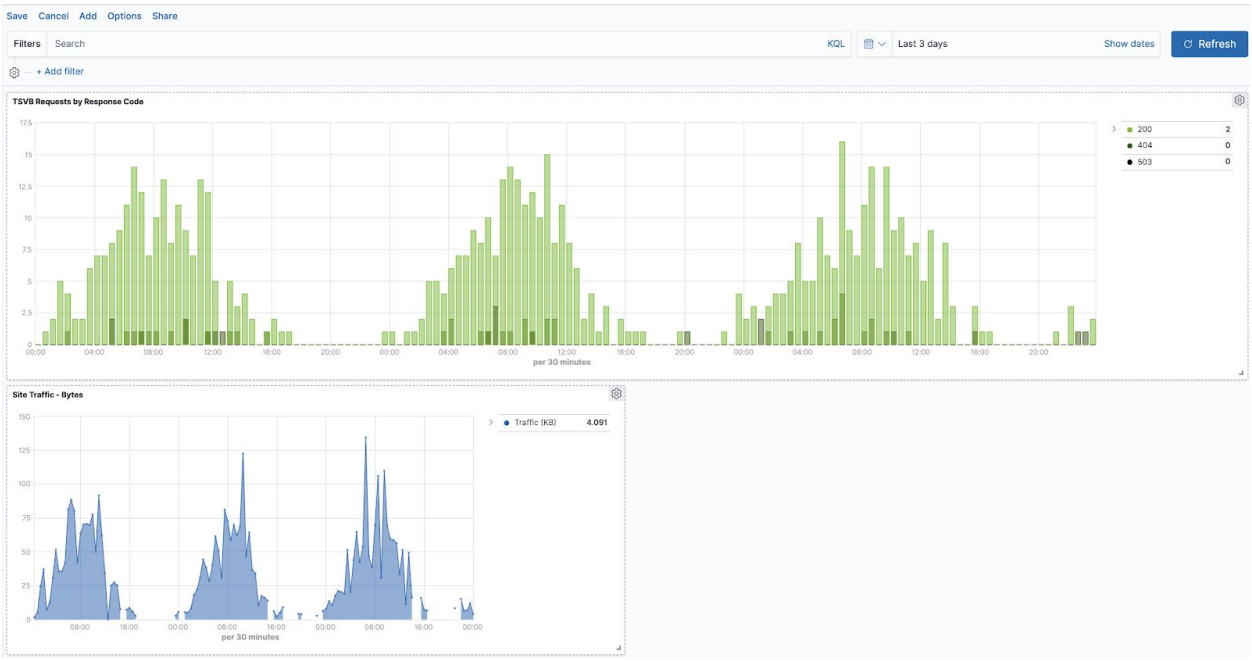


Note that the scale of your visualization has changed, now you’re displaying traffic per buckets in KBs.

3e. Next to Label that says “**Traffic**” change the color to be different from the previous visualization. In this example we are selecting a blue color.



3e. Save the visualization similarly to the way in step 2 and observe it appearing on your dashboard. Name it something along the lines of "**Site Traffic - bytes"**



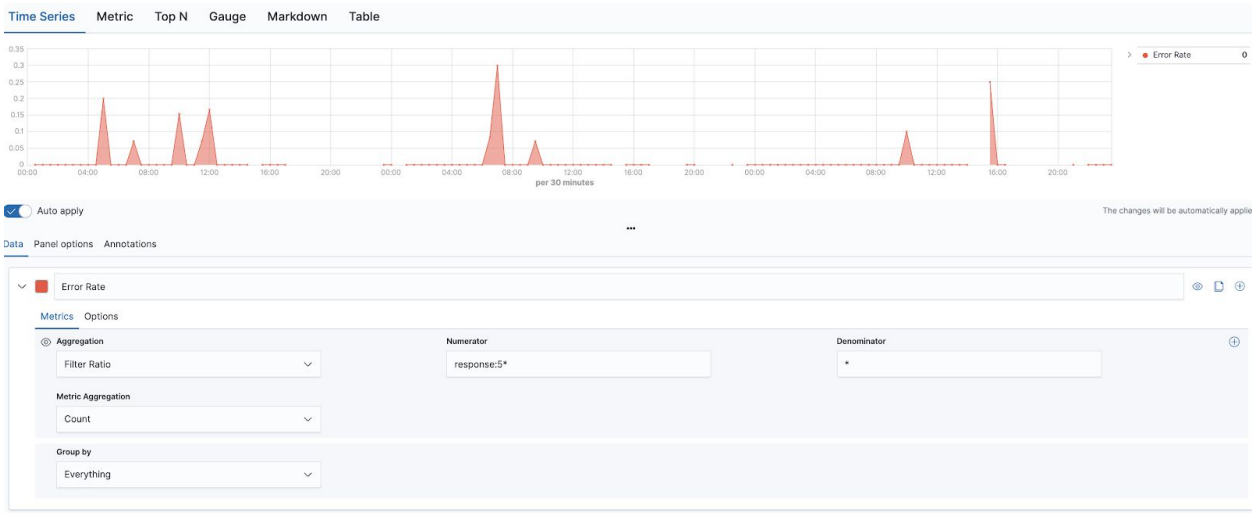
4. Suppose you would like to calculate the error rate of your site based on the logs. How do we do that? 5xx response code stands for Internal Server Error. So the ratio of this response code vs everything else is your error percentage!

4a. On our dashboard similarly to step 2 add a new Visual Builder visualization. Once on the visualization specify “**kibana*logs**” as data source and “**timestamp**” as time field (see steps 2 for details on how to do this)

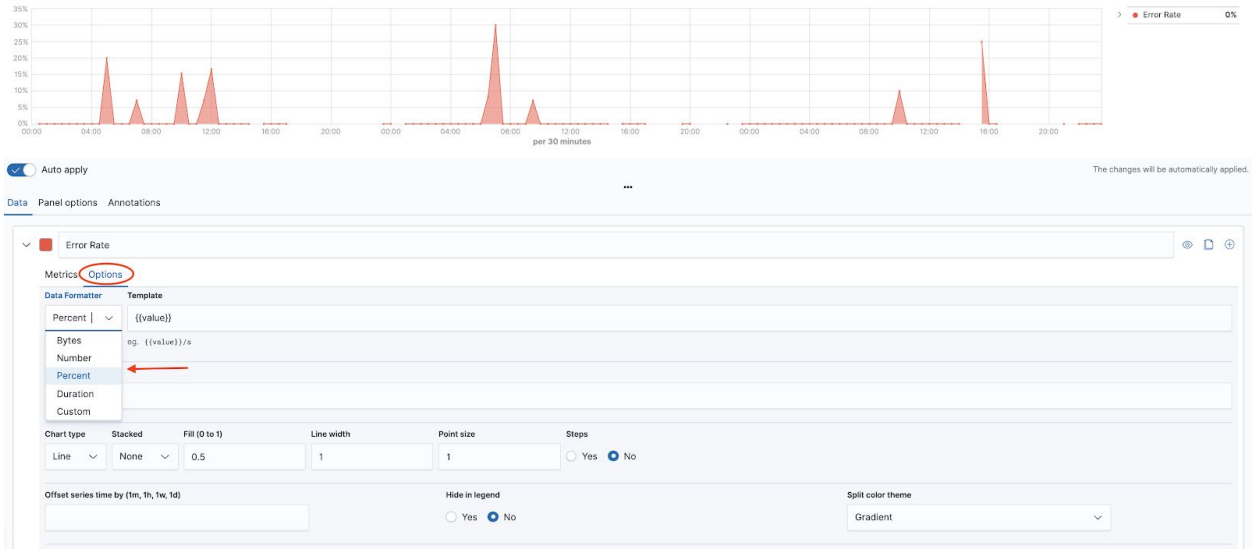
4b. In the Label section enter “**Error Rate**” and select the color red.

4c Select the following:

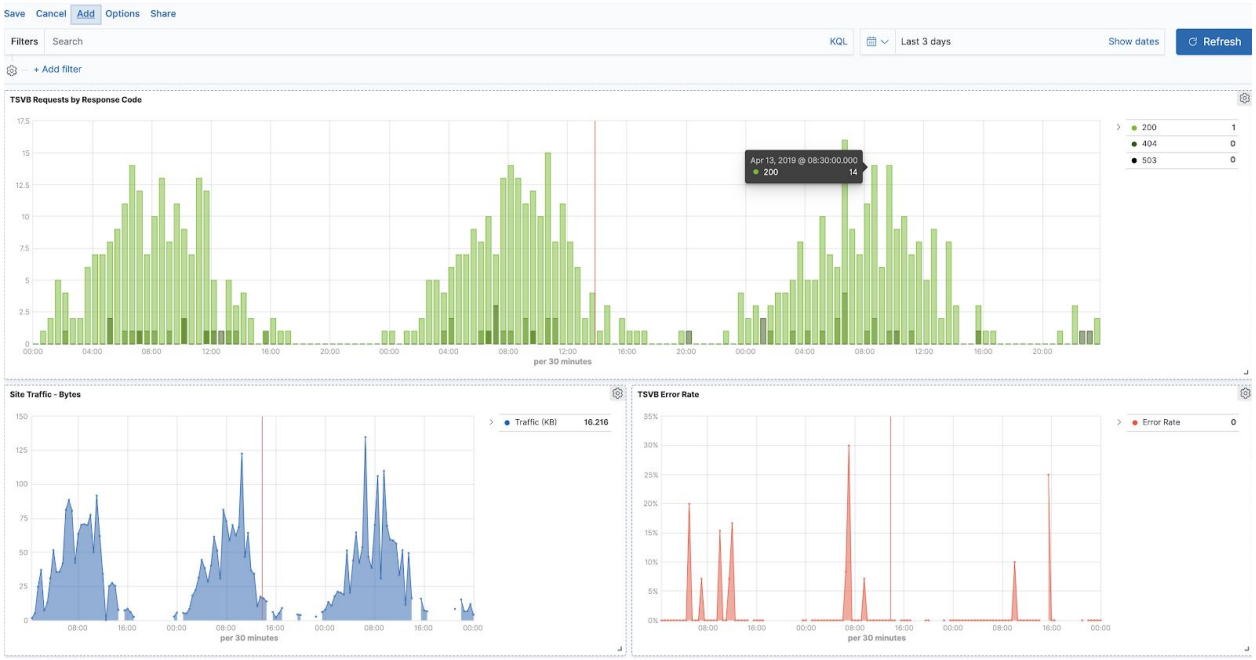
Aggregation: Filter Ratio
Numerator: response:5*
Denominator: leave as *



4d. The resulting visualization shows error percentage. But note how the axis is showing decimal values instead of percentage. Let's fix that! In "**Options**" section select "**Percent**" for Data Formatter.

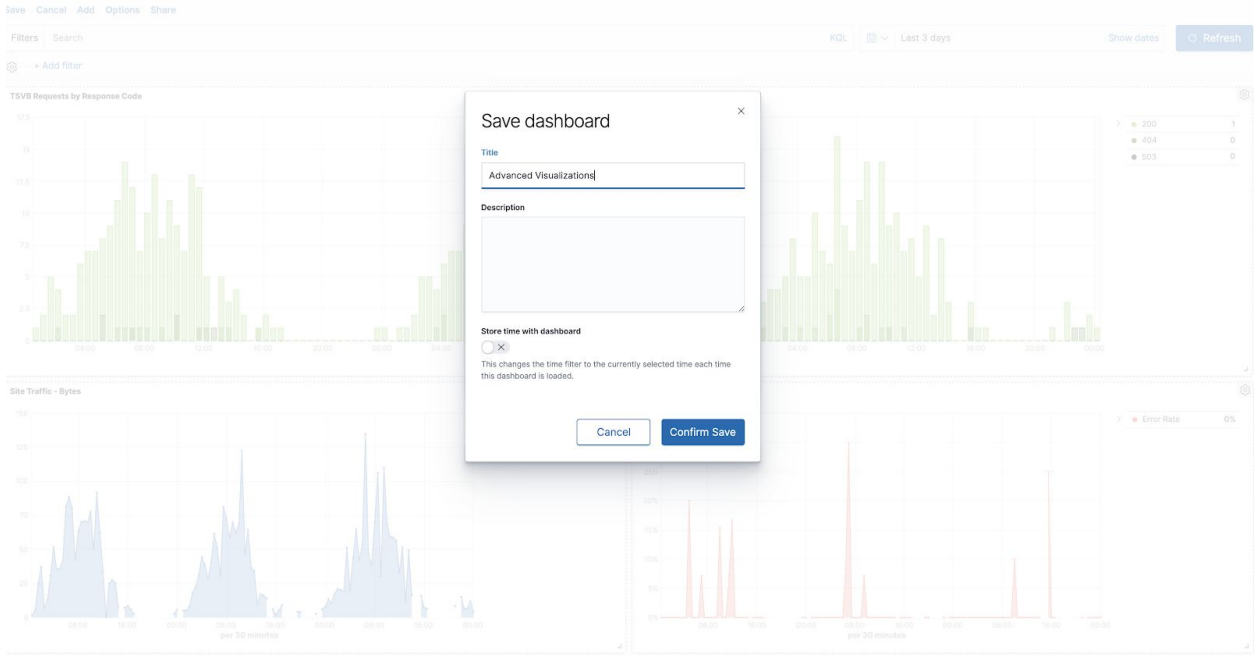


4d. Save this visualization similarly to previous steps and observe it show up on the dashboard. Name the visualization "**TSVB Error Rate**"

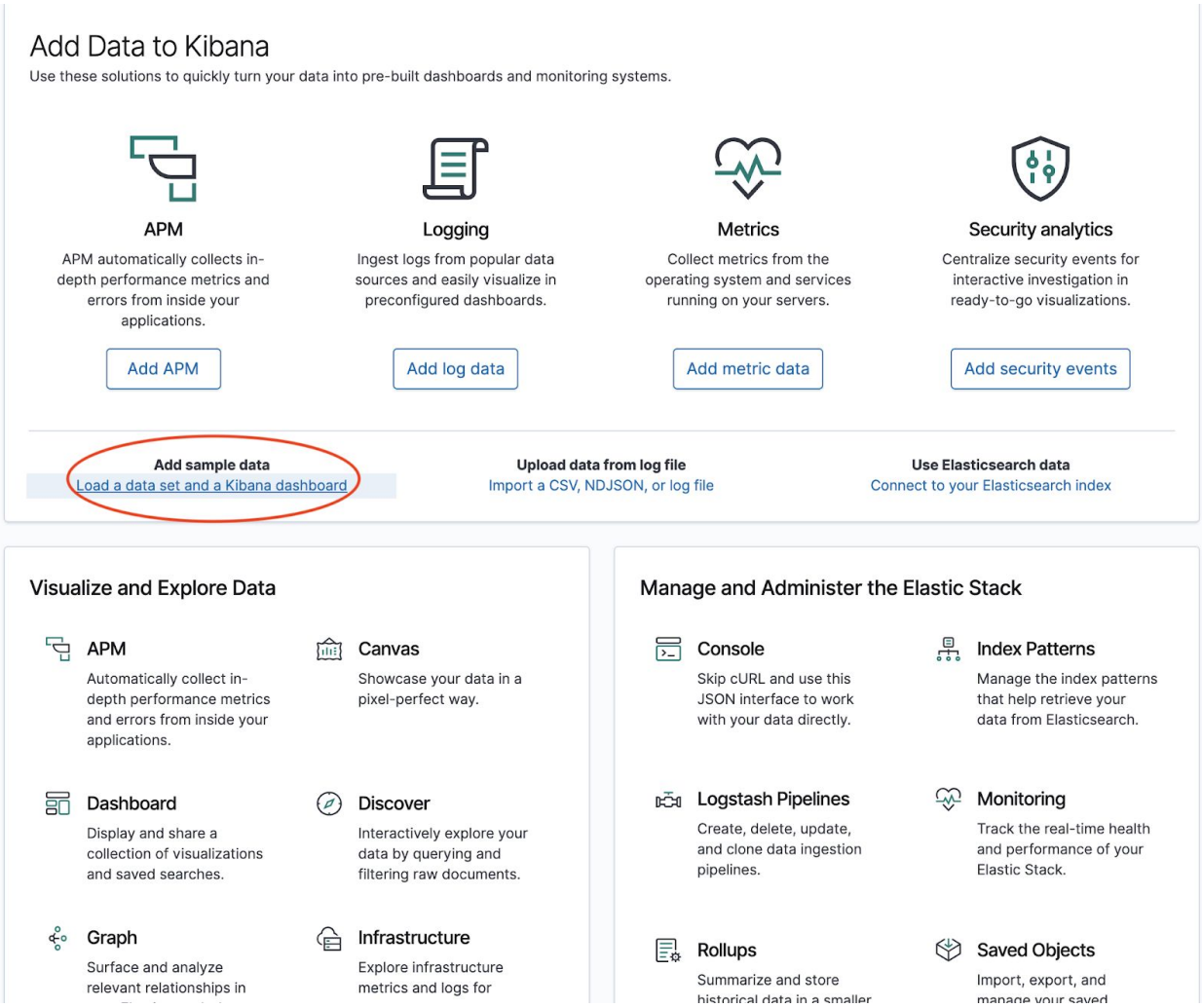


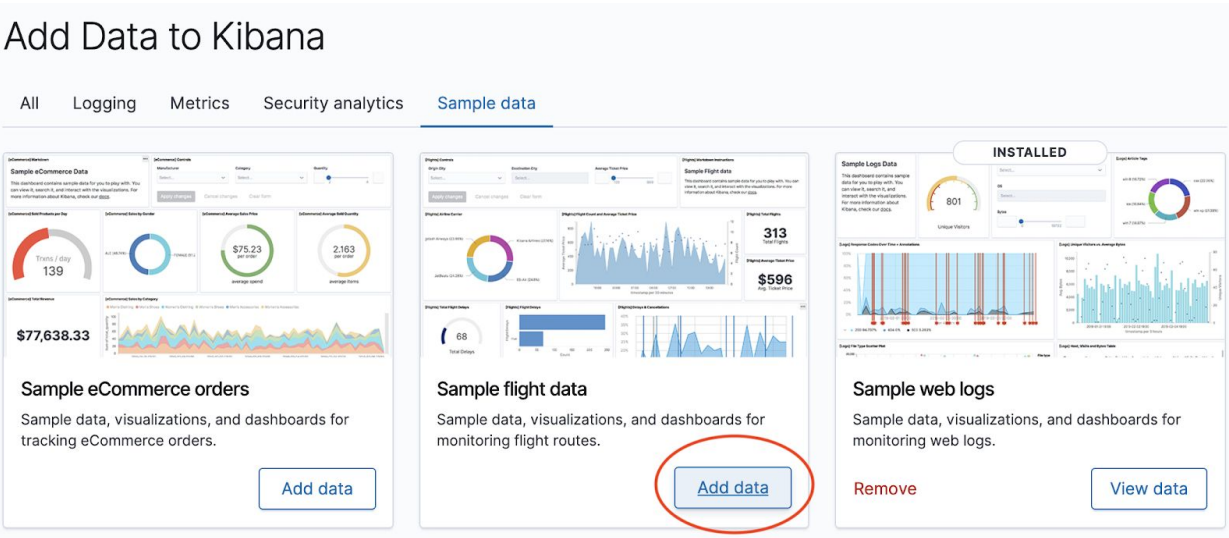
5. In this subsection we are going to explore the capabilities of Timelion. Similarly to Visual Builder, Timelion gives you the ability to visualize multiple data sources on one chart and do interesting mathematical operations with them. We will not dig deep into this visualization today, but will show how to visualize data from multiple data sources on it.

5a. Save your current dashboard (Save button next to Add).

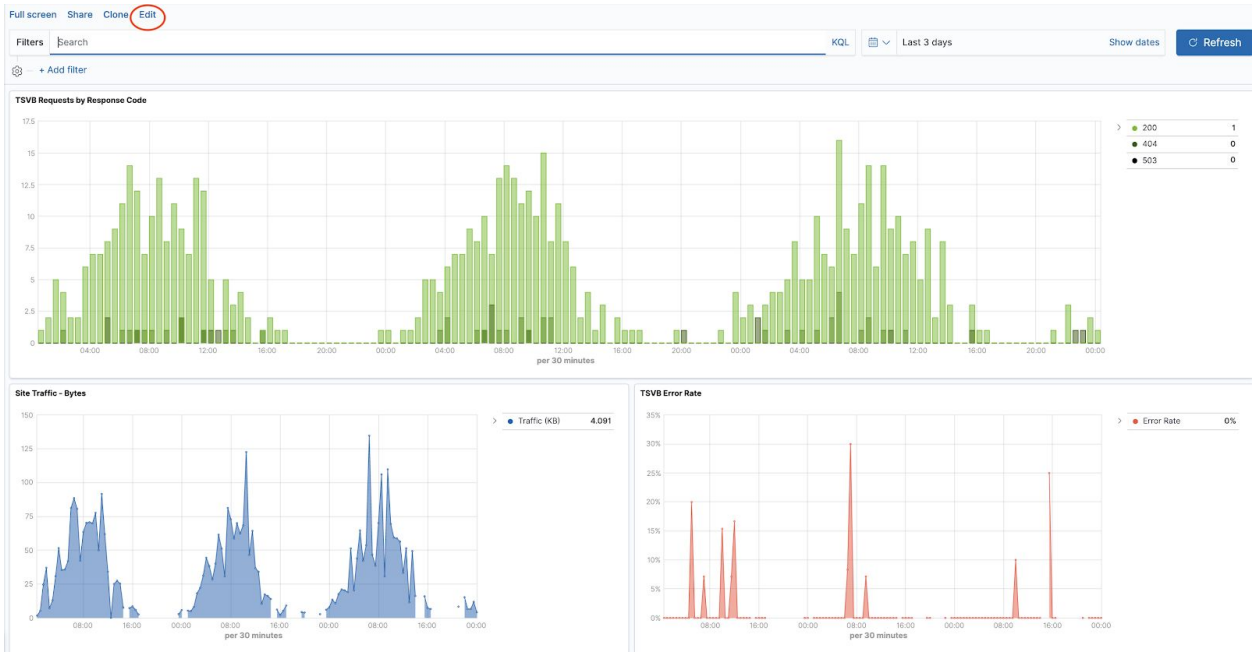


5b. Go to **Kibana Landing Page** and similarly to adding Logs sample data click to add Flights sample Data (Or click the Kibana logo in the very top left corner)

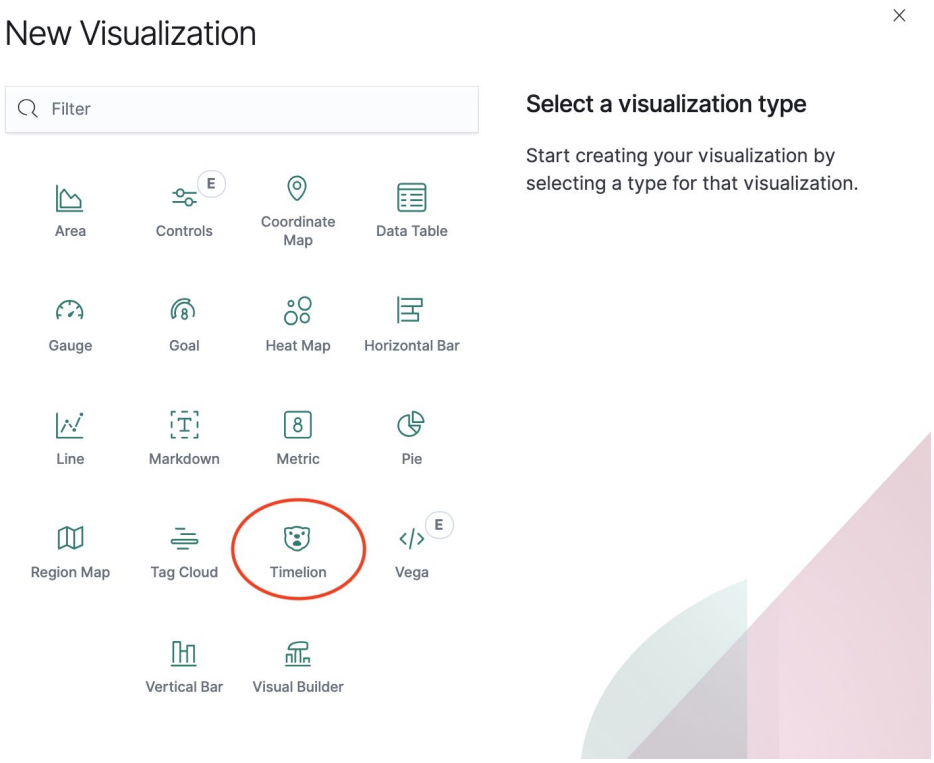

Add Data to Kibana
Use these solutions to quickly turn your data into pre-built dashboards and monitoring systems.
APM
APM automatically collects in-depth performance metrics and errors from inside your applications.
Add APM
Logging
Ingest logs from popular data sources and easily visualize in preconfigured dashboards.
Add log data
Metrics
Collect metrics from the operating system and services running on your servers.
Add metric data
Security analytics
Centralize security events for interactive investigation in ready-to-go visualizations.
Add security events
Add sample data
Load a data set and a Kibana dashboard
Upload data from log file
Import a CSV, NDJSON, or log file
Use Elasticsearch data
Connect to your Elasticsearch index
Visualize and Explore Data
APM
Automatically collect in-depth performance metrics and errors from inside your applications.
Canvas
Showcase your data in a pixel-perfect way.
Dashboard
Display and share a collection of visualizations and saved searches.
Discover
Interactively explore your data by querying and filtering raw documents.
Graph
Surface and analyze relevant relationships in
Infrastructure
Explore infrastructure metrics and logs for
Manage and Administer the Elastic Stack
Console
Skip cURL and use this JSON interface to work with your data directly.
Index Patterns
Manage the index patterns that help retrieve your data from Elasticsearch.
Logstash Pipelines
Create, delete, update, and clone data ingestion pipelines.
Monitoring
Track the real-time health and performance of your Elastic Stack.
Rollups
Summarize and store
Saved Objects
Import, export, and


Add Data to Kibana
All
Logging
Metrics
Security analytics
Sample data
$77,638.33
$75.23
2,163
313
$596
68
INSTALLED
801
Sample eCommerce orders
Sample data, visualizations, and dashboards for tracking eCommerce orders.
Add data
Sample flight data
Sample data, visualizations, and dashboards for monitoring flight routes.
Add data
Sample web logs
Sample data, visualizations, and dashboards for monitoring web logs.
Remove
View data

5c. Go back to our **Dashboard** and click on **Edit**.



5d. Similarly to previous steps click on the **Add new visualization**, but this time select **Timelion**.



5e. On Timelion visualization **enter the following code** inside the window:

```
 .es(index=kibana*logs, timefield=timestamp).color(red),
.es(index=kibana_sample_data_flights, timefield=timestamp).color(green)
```

What we are telling Timelion to do above is to get data from 2 different indices and color one of them in red and another one in green.

5d. We can also include queries in them to just show the information we are interested in and label it properly. Replace the previous code with the following and observe how it changes.

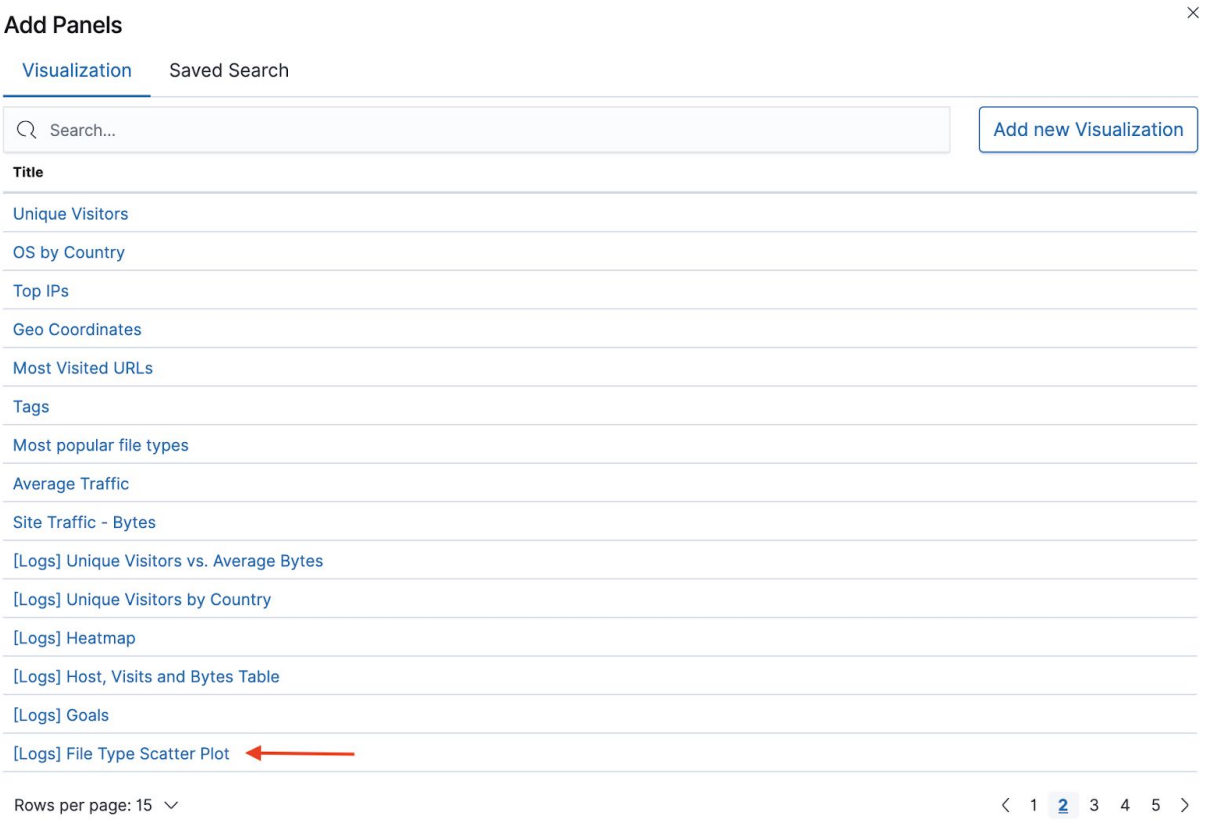
```
.es(index=kibana*logs, timefield=timestamp).color(red).label("Logs
Volume"),
.es(index=kibana_sample_data_flights, timefield=timestamp, q="Carrier
:JetBeats").color(green).label("JetBeats Flights")
```

5e. We can do a lot with Timelion - it probably deserves a workshop on its own. For full list of Timeioin capabilities checkout https://www.elastic.co/guide/en/kibana/current/timelion.html

Meanwhile **save** your visualization similarly to all the other visualizations and watch it appear on your Dashboard.

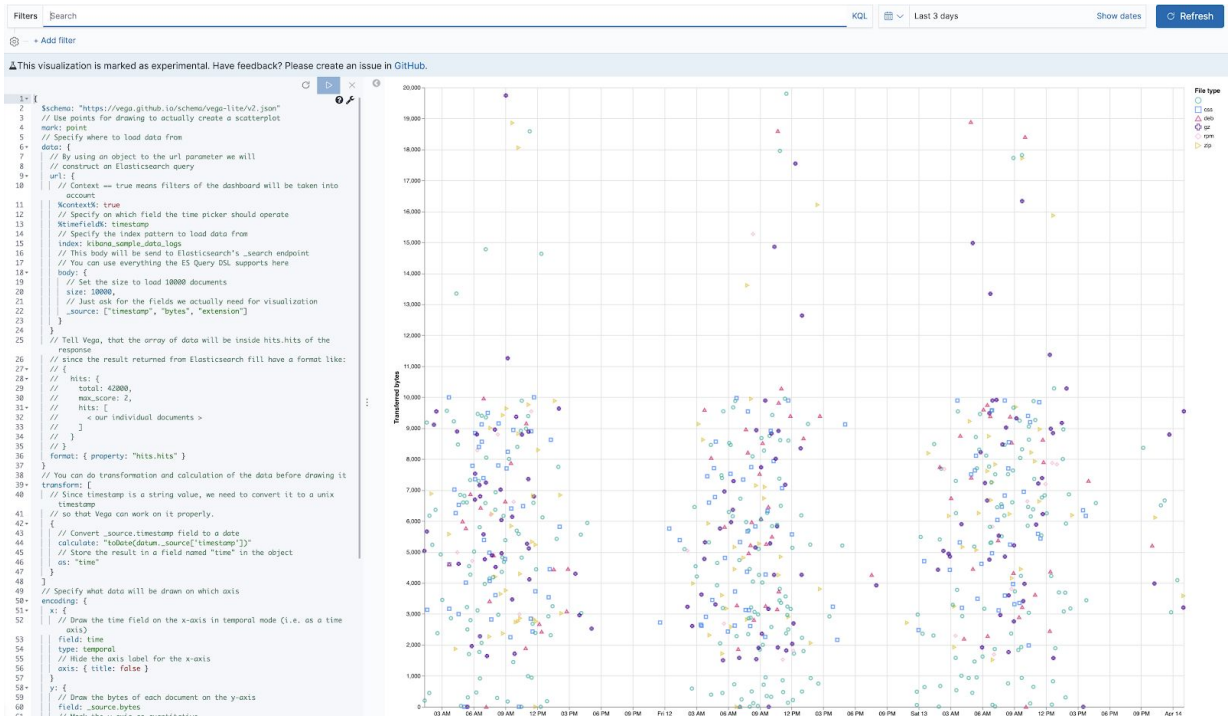
6. And last, but not least let's play a little with Vega visualizations. The Vega plugin gives you complete freedom to come up with your own visualizations and the ability to define your own custom Elasticsearch queries to fetch the data. This is probably the most complex, but also the most powerful visualization type available to Kibana users and also deserves a course of its own. Our sample dashboards come with few sample visualizations, let's add one of them and understand how it is built.

6a. **Add new visualization**, but this time select the existing one - **File Type Scatter Plot.**



6b. On your dashboard you will have a new visualization appear. **Click on its Options, and then select “Edit visualization”**

6c. Inside the visualization read the code and the comments that explain each section well.



6d. (Optional) In the “_data” section in the “_source” add field “tags”, change the “color” section to be based on “_source.tags”, apply the changes. Note that few colors have added, as there are more combinations of tags vs file extensions. Complete code below, feel free to copy-paste it. If you apply this change save the visualization and go back go to your dashboard.

```
{
  $schema: "https://vega.github.io/schema/vega-lite/v2.json"
  // Use points for drawing to actually create a scatterplot
  mark: point
  // Specify where to load data from
  data: {
    // By using an object to the url parameter we will
    // construct an Elasticsearch query
    url: {
      // Context == true means filters of the dashboard will be taken into account
      %context%: true
      // Specify on which field the time picker should operate
      %timefield%: timestamp
      // Specify the index pattern to load data from
      index: kibana_sample_data_logs
      // This body will be send to Elasticsearch's _search endpoint
      // You can use everything the ES Query DSL supports here
      body: {
        // Set the size to load 10000 documents
        size: 10000,
        // Just ask for the fields we actually need for visualization
        _source: ["timestamp", "bytes", "extension","tags"]
      }
    }
    // Tell Vega, that the array of data will be inside hits.hits of the response
    // since the result returned from Elasticsearch fill have a format like:
    // {
    //   hits: {
    //     total: 42000,
    //     max_score: 2,
    //     hits: [
    //       < our individual documents >
    //     ]
    //   }
    // }
    format: { property: "hits.hits" }
  }
  // You can do transformation and calculation of the data before drawing it
  transform: [
    // Since timestamp is a string value, we need to convert it to a unix timestamp
    // so that Vega can work on it properly.
```

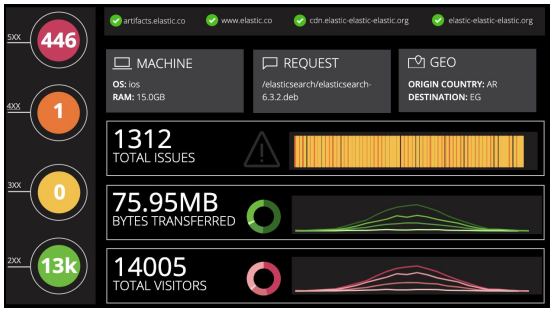
```
    {
      // Convert _source.timestamp field to a date
      calculate: "toDate(datum._source['timestamp'])"
      // Store the result in a field named "time" in the object
      as: "time"
    }
  ]
  // Specify what data will be drawn on which axis
  encoding: {
    x: {
      // Draw the time field on the x-axis in temporal mode (i.e. as a time axis)
      field: time
      type: temporal
      // Hide the axis label for the x-axis
      axis: { title: false }
    }
    y: {
      // Draw the bytes of each document on the y-axis
      field: _source.bytes
      // Mark the y-axis as quantitative
      type: quantitative
      // Specify the label for this axis
      axis: { title: "Transferred bytes" }
    }
    color: {
      // Make the color of each point depend on the _source.extension field
      field: _source.tags
      // Treat different values as completely unrelated values to each other.
      // You could switch this to quantitative if you have a numeric field and
      // want to create a color scale from one color to another depending on that
      // field's value.
      type: nominal
      // Rename the legend title so it won't just state: "_source.extension"
      legend: { title: 'File type' }
    }
    shape: {
      // Also make the shape of each point dependent on the extension.
      field: _source.extension
      type: nominal
    }
  }
}
```

Observe changes taking effect on Vega visualization compared to its original view.

7. Congratulations! You are done with this section of the lab!

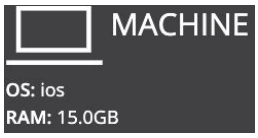
Lab 8: Building Infographics

In this lab, we will recreate some elements in web log sample Kibana Canvas dashboard which we looked at in the Reverse Engineering lab, above. Canvas is a great way to portray your live data and convey information to peers and external stakeholders.



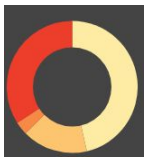
Off of that workpad, we are going to build the following:

A count of response codes in the 500's using SQL

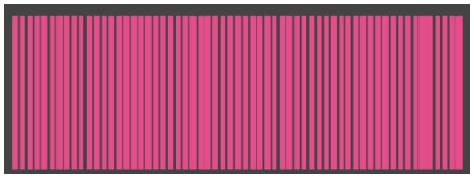


RAM for a given operating system on a machine using raw documents

The percent of error logs as calculated using SQL



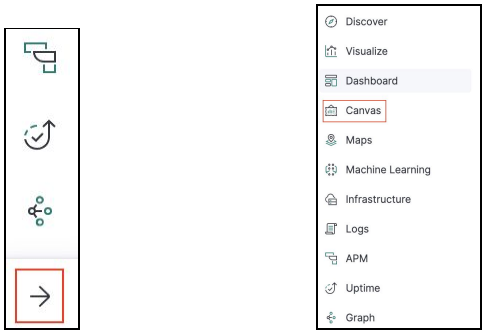
Pie chart showing visitors grouped by host using SQL



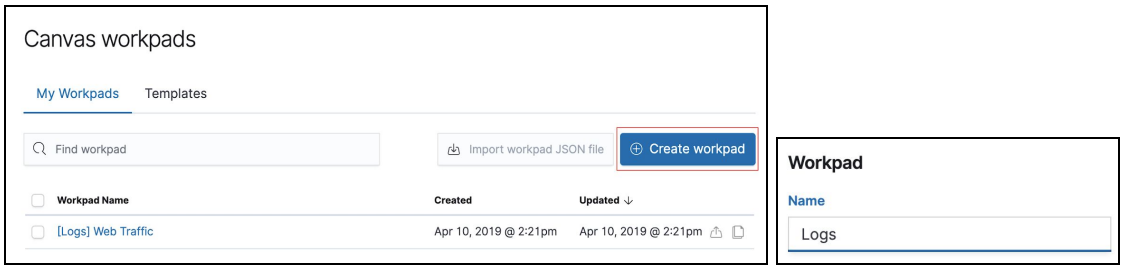
Line chart of concentration of errors over time using SQL

Well, let's get started!

1. Log in as **visual designer** user and pick **operations**. Click on the expand icon at the bottom, and then click on Canvas on the right-hand side:

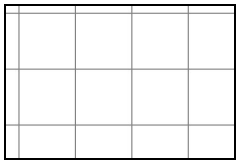


2. Click to create a new workpad and then give Canvas a title in upper right part of the new workpad; clicking away from the textbox will persist new title.



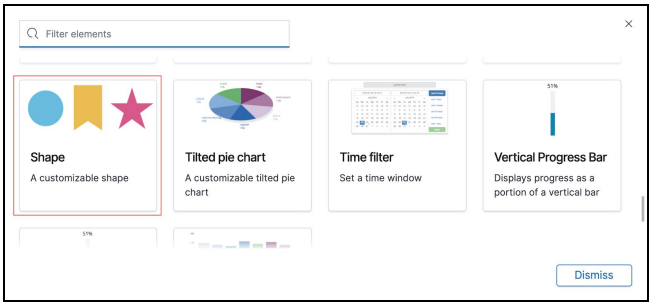
3. Show Grid:

3a.Before we start adding elements to our workpad, let's enable the grid lines by using the keyboard shortcut alt+g (option+g on Mac).

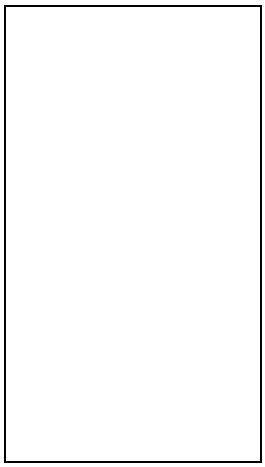


4. Add a panel for background

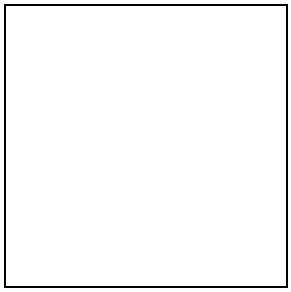
4a. Now, for the first element. We are going to build a response code count. Let's start by clicking on **Add element,** scroll down, and select **Shape**



4b. Square shape is the default so we don't have to change the shape here. Disable **Maintain aspect** ratio. Set color for **fill** to #221F20, **border** #777777, and **width** 0. Colors can be set by clicking on swatch.



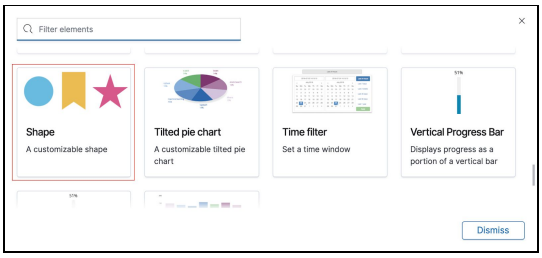
4c. Expand the shape by dragging the corners of the shape and move it to the desired location using the grid on our workpad.



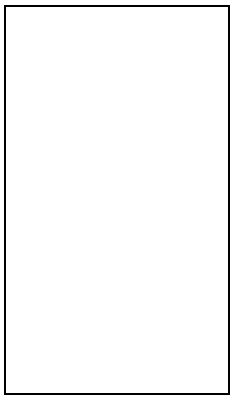
4d. If desiring precision, consider checking out the sample Logs Workpad provided with the dataset. Reverse engineering is a great learning tool! This is what we are striving towards building in our first effort over steps 4 through 9:

5. Decorative outer circle

5a. The circle outline is a circle shape element with no color fill and border width of 2. Click on **Add element** and select Shape. Choose circle in Select a shape

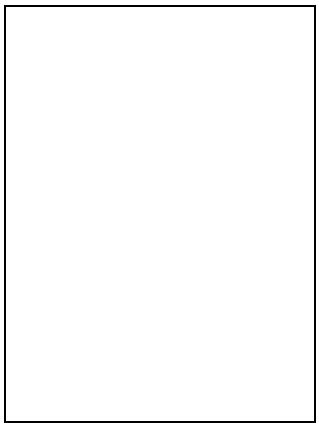


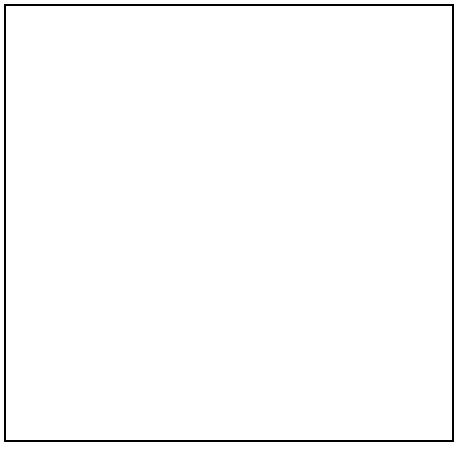
5b. Select **Circle shape** by clicking on default shape and switching. Set **fill** to **#221F20**, **border #777777**, and **border width 2**. Leave Maintain aspect ratio. Size and move the circle inside the panel we just created.



6. Decorative inner circle

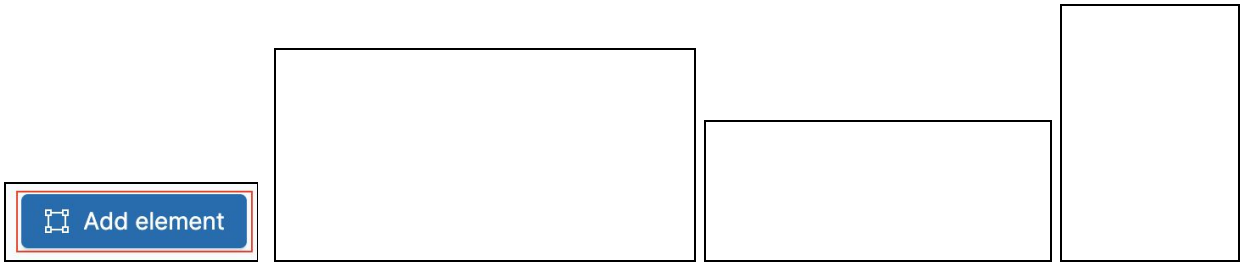
6a. This is very similar to what we just did with the outer circle except we will need to change the **fill color** to **#C83C5B** and border to transparent with **0 width**. Keep the default border color.





7. Count of 5xx errors

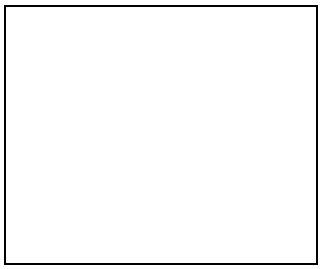
7a. Click on **Add element** and select **Metric**. Go to the Selected layer panel on the right and click on the **Data** tab and click on **Change your data source**. Select **Elasticsearch SQL**.



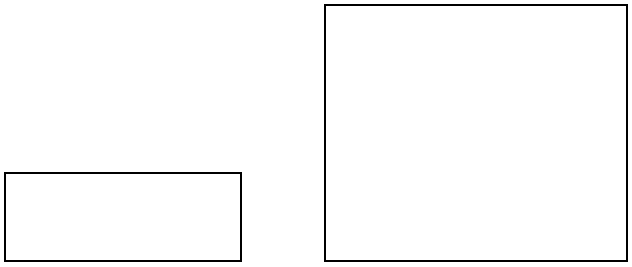
7b. Paste the following text into the Elasticsearch query textbox. We are getting a count of response code that are equal to or greater than 500

```
SELECT COUNT(*) as response_code FROM kibana_sample_data_logs WHERE response >= 500
```

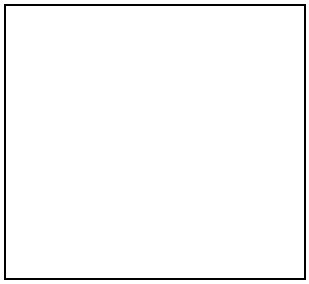
Click on **Save** when done.



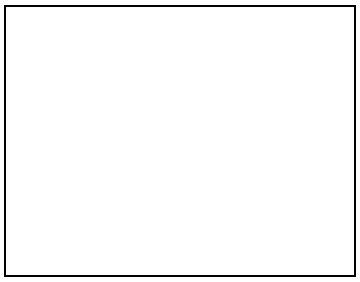
7c. Let's go back to the **Display** tab. In the Measure panel, select Value under the Number heading and response_code for select column. Delete the Metric label element since we will be labeling the element outside



7e. In Metric text settings, set our font to size 48, open sans font style, and white text color. Delete Label text settings section since we have removed label already.

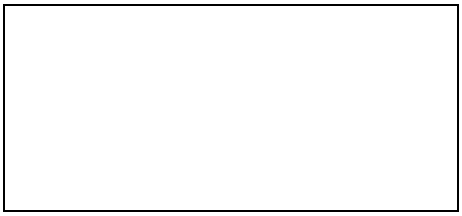


7f. Drag the Metric into the circle and adjust spacing to fit.

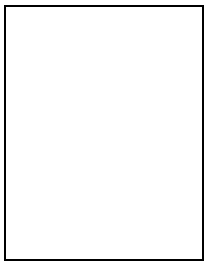


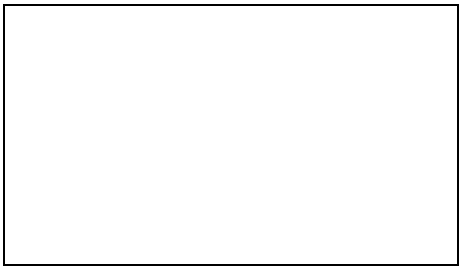
8. 5xx response code label

8a. We are labeling the element outside of the outer cycle. It will be a markdown element. Click on Add element and select Markdown



8b. Go to Markdown content in the Markdown panel under the Display tab. Replace the text there with 5xx and click Apply Click on the little + button at the upper right hand corner of the Display tab and select **text settings**. Set font size to 18, open sans font style, and white text color.

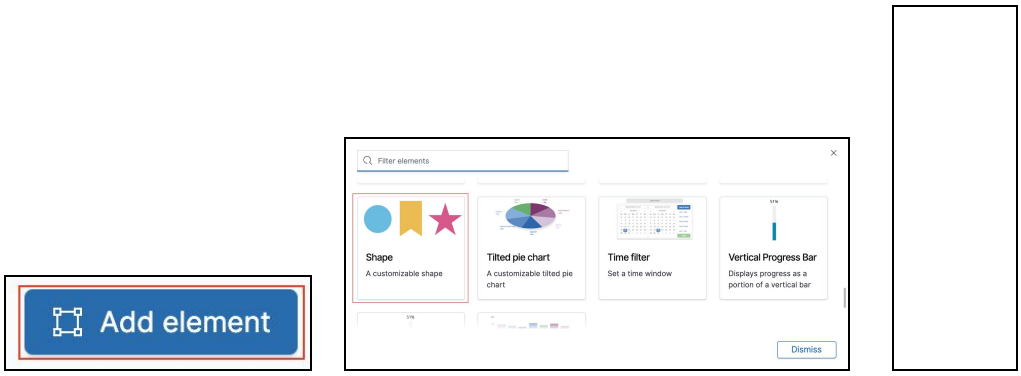




8c. Reposition Markdown element to fit into graphic.

9. Platform for 5xx Label

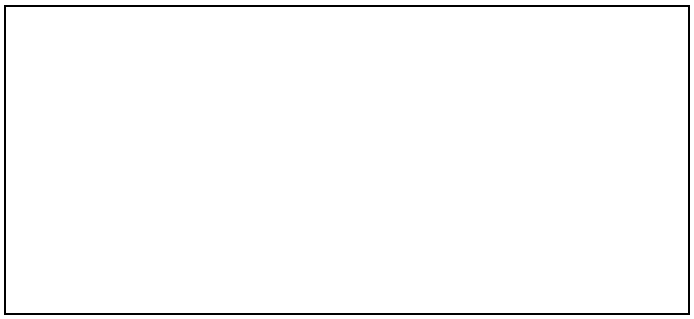
9a. We will add a platform below the 5xx label. It is a very narrow rectangle shape. Click on Add element and select Shape. Square shape is the default. Disable Maintain aspect ratio as we need a rectangle. Fill the shape and border with color #CFD0D2 and set border width to 2.



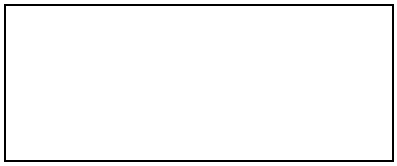
9b. Adjust the shape and move it to the right position. This graphic is done! The 4xx's and 3xx's can be done the same way.

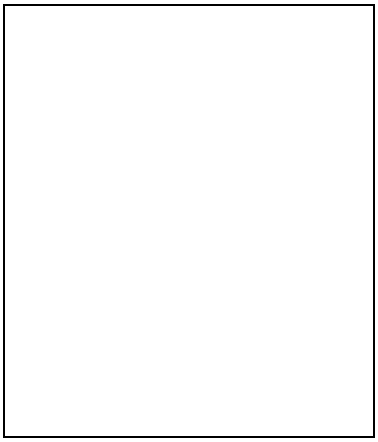
10. Markdown content from Elasticsearch raw documents

10a. The computer and the label are static elements, one image and one markdown. The OS and RAM are more interesting as they are data driven. Click on Add element and select Markdown.

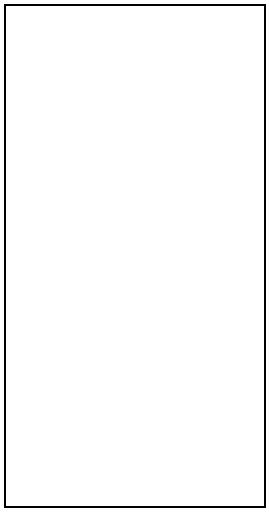


10b. In the Selected layer panel, select the Data tab. Click on Change your data source. Select Elasticsearch raw documents

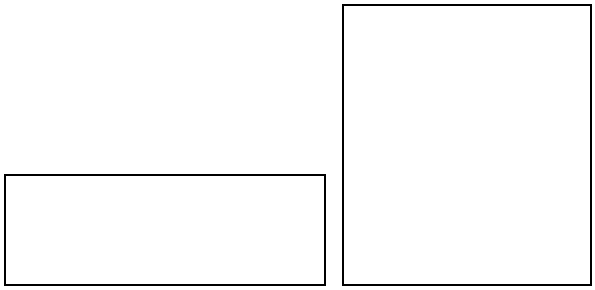




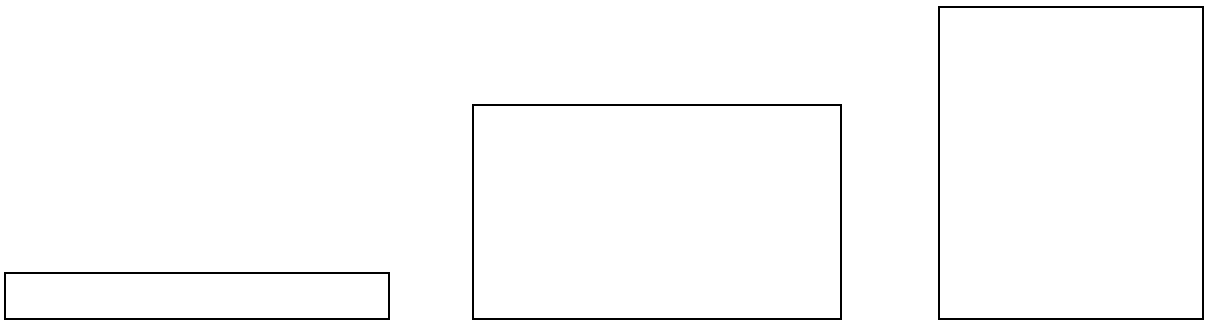
10c. Change the **index** to **kibana_sample_data_log**s and set the sort field to **@timestamp.** In **Fields**, select **machine.os** and **machine.ram.** Click on **Save**.



10d. Click on the Display tab. In the Markdown content, replace the text with `**OS:**` click on Apply.

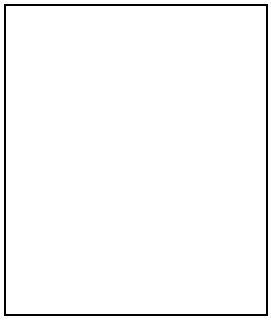


10e. Click on the little + button to the right of markdown at the upper right hand corner of the Display tab and select Markdown content. Add `{getCell "machine.os"}` to the textbox below the Markdown content section we just added and click on Apply

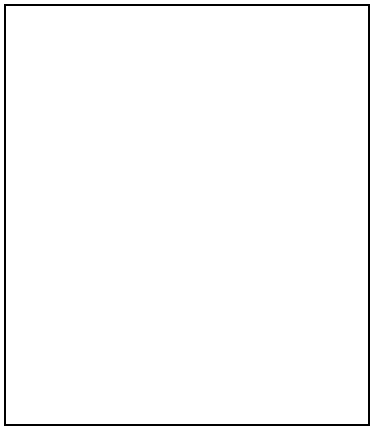


10f. Click on the little + button at the upper right hand corner of the Display tab and select Markdown content. Add the following to the Markdown content textbox. Click on **Apply**.

```
 \
**RAM:**
```



10g. Click on the little + button at the upper right hand corner of the Display tab and select Markdown content . Add the following to the Markdown content `{getCell "machine.ram" | formatNumber "00.0b"}`. Click on Apply.



10h. Click on the Expression editor to fix the incorrect text. Locate

```
 markdown "**OS:**"  {getCell "machine.os"} "\\
 **RAM:** " " {getCell \"machine.ram\" | formatNumber \"00.0b\"}"
```

Replace with

```
 markdown "**OS:** " {getCell "machine.os"} "\
 **RAM:** " {getCell "machine.ram" | formatNumber "00.0b"}
```

10i. Click run and click on close. Your graphic should look like this:

11. Image Reveal

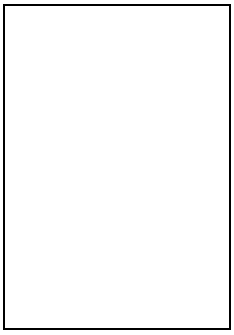
Image reveal allow you to represent a percentage in two different shades of the same image. One image is partially revealed based on a calculated percentage.

11a. Click on Add element and select Image reveal.

11b. Import the images we will be using, Fire gray.svg (http://bit.ly/elastic_viz_lab_firegray)

and Fire color.svg (http://bit.ly/elastic_viz_lab_firecolor) . In the Selected layer panel of the Display tab, expand Image upload if it is not expanded already. Click on Import. Drag the **Fire gray.svg** file to Select or drag and drop a file. Click on Import again. Drag the **Fire color.svg** file to Select or drag and drop a file.

11c. Click on the **Data** tab in the Selected layer. Click on **Change your data source** and select **Elasticsearch SQL**

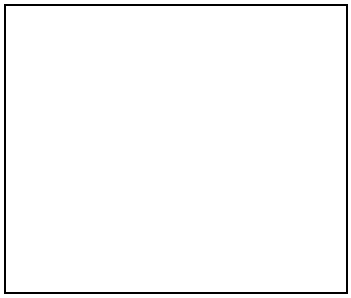


11d. In Elasticsearch SQL query textbox, replace the text with the following SQL query

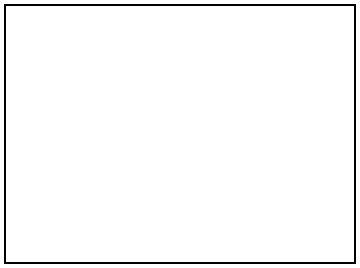
```
SELECT COUNT(*) as total_errors FROM kibana_sample_data_logs
WHERE tags LIKE '%warning%' OR tags LIKE '%error%'
```

We are counting the number of errors and warnings in kibana_sample_data_logs

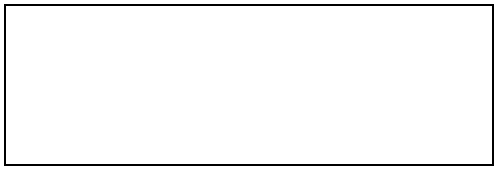
Click on **Save**

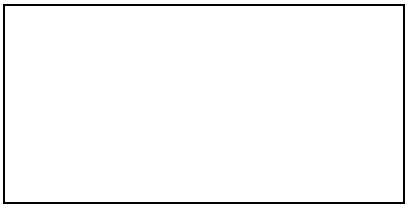


11e. Click on the **Display** tab. Select the **Fire color image** as the Reveal image.



11f. Click on the little **+** button to the right of Reveal image and select **Background image**. Select **Fire gray** as the background image. Leave Reveal from set to bottom.

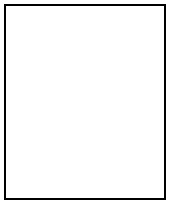




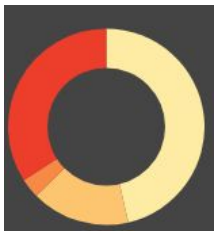
11g. Click on the **Expression editor** to add a math function using the results of our Elasticsearch SQL query. Replace `math "..."` with `| math "sum(total_errors / 4000)"`

Please note **4000** is an arbitrary number. You may have to come up with your own but the result of the math function must be less than 1 or 100% so we know what percentage of the image to hide/show. Click on **Run.**

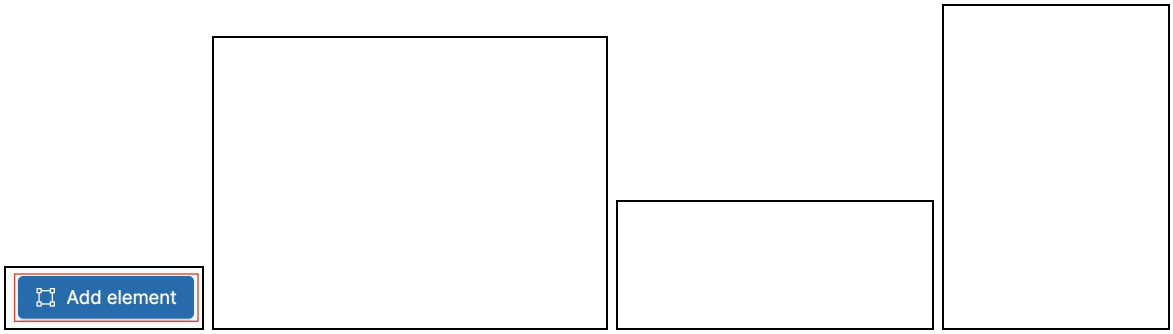
11h. The reveal will render percent of errors out of 4000 using reveal image. After a little resizing, your Reveal Image should look like the following. Toy with the 4000 value to see different fraction.



12. Pie/Donut Chart

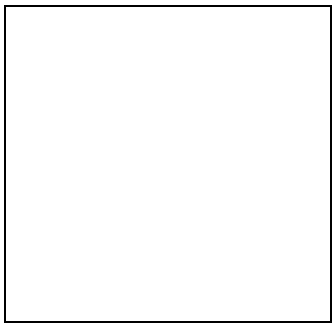


12a. Click on Add element and select Donut chart. In the Selected layer Panel, click on the Data tab. Click on Change your data source and select Elasticsearch SQL

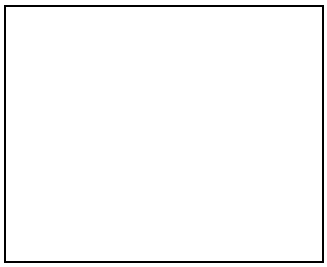


12b. We are counting the number of visitors per host in the kibana_sample_data_logs index. In Elasticsearch SQL query textbox, replace the text with the following SQL query. Click on Save

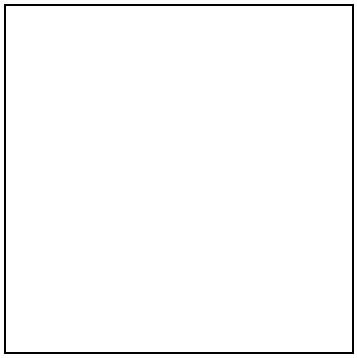
```
SELECT COUNT(*) as total_visitors, host FROM kibana_sample_data_logs GROUP BY host
```

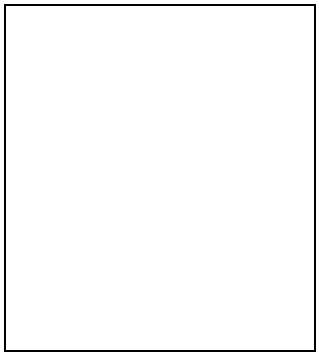


12c. Click on the Display tab. Set Slice Labels to Value and select column to host. Set Slice Angle to Value and select column to total_visitors



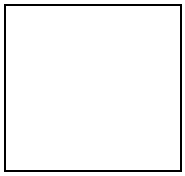
12d. In Chart style, click on the little + button and select Color palette. Click on the Color palette band, scroll down, and set the color to Elastic Orange



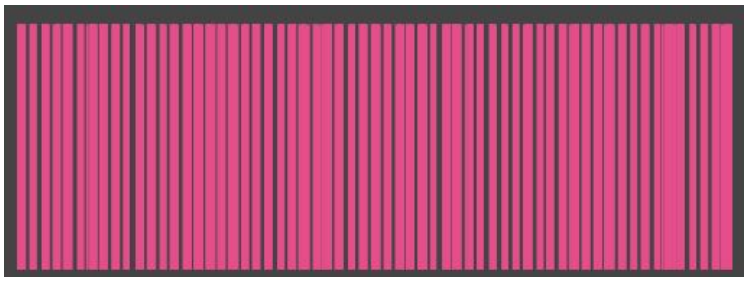


12e. In chart style, if we set inner radius to 0, then it is just a pie chart. Leave the radius at 60 so it is a donut chart. Leave the label off. Set the Legend position to off to create a cleaner look.

12f. Your final product can be resize and should look like this:

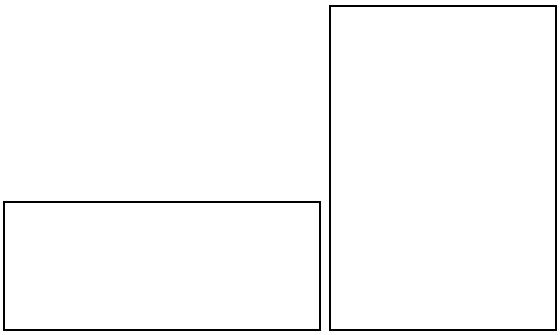


13. Line Chart



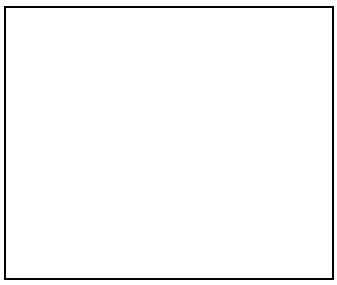
The above charts are of the same type, line chart. In the former, we represented our data with bar and the latter with lines. We will choose to build the first chart.

13a. Click on Add element and select Line chart. In the Selected layer Panel, click on the Data tab. Click on Change your data source and select Elasticsearch SQL

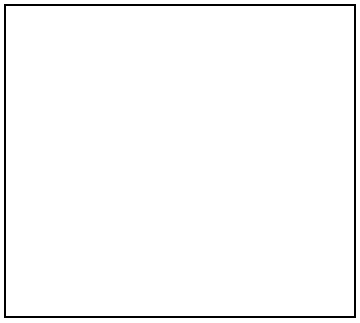


13b. We are counting the number of errors over time in the kibana_sample_data_logs index. In Elasticsearch SQL query textbox, replace the text with the following **SQL query** and click **Save**.

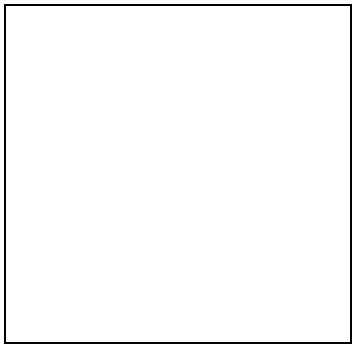
```
SELECT COUNT(*) as total_errors, timestamp FROM kibana_sample_data_logs WHERE tags LIKE '%error%' GROUP BY timestamp ORDER BY timestamp DESC
```



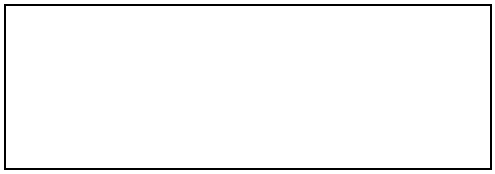
13c. Click on the Display tab. In dimensions and measures, set X-axis to Value and timestamp. Set y-axis to Value and total_errors

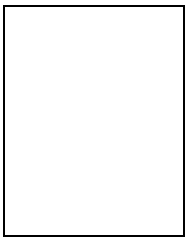


13d. Click on the little + button under Chart style and select Legend position. Click the + button again but this time select X-axis. Repeat again and select the Y-axis. Set the Legend position to hidden. Turn off the X-axis. Turn off the Y-axis



13e. Expand Default style. Set Line to None, Bar to 1, and Point to None. Click the Color block to change the color to **#e74b8b** for Bar representation.





13f. Here is the final result. Take a look at the original **Canvas Log Workpad** we were emulating. How close did you come? More importantly, take a look at the line chart and convince yourself by clicking around on the controls that you could in fact build a similar line chart! **SUCCESS!**