**Elasticsearch Essential Training – LinkedIn Learning**

Introduction

Index documents and search documents – hence widely adapted by organizations of all sizes.

Based on JavaScript, Web Front End, Database Basics

1. Overview

Elasticsearch is a distributed restful search and analytics engine that helps with all kinds of use cases in today's technology landscape. For its data format, Elasticsearch uses JSON, which is the JavaScript Object Notation. And for its interface, it uses HTTP. Now, both of these are incredibly common on the web which makes Elasticsearch really flexible and can be integrated with almost any application out there. In working with Elasticsearch there are clients available in Java, .NET, Python, and many other languages. Across the landscape, Elasticsearch is by far the most popular enterprise search engine. In short, if you're looking to do search today and don't want to reinvent the wheel this is probably your best bet. Also, because of its incredible ability to scan documents and find information, it is becoming more useful for data scientists and analysts to understand what's happening or what insights can be gleaned from the data that is being stored inside of it.

A picture containing text, businesscard, screenshot, font

Description automatically generatedA screenshot of a computer

Description automatically generated with low confidence

Benefits of Elastic Stack (OpenSource):

A picture containing text, screenshot, font, rectangle

Description automatically generatedA screenshot of a phone

Description automatically generated with low confidence

- [Instructor] The Elastic Stack can be used for countless different use cases, however, I thought it would make sense just to talk about some of the most common ones that people are using it for today. A big one is security and log analytics. It's incredibly common to have logs dash pulling in your web server logs, or security logs from your firewall, for example. Throw them into an index where then, on top of that, you have dashboards, learning, and all kinds of things that you may want to be looking at in your real time with regards to what's going on with your traffic or your security systems. So, when it comes to analyzing security and log events, this is a fantastic tool because of how good it is at identifying things in real time. Another use case, which may not be totally obvious, is to use the Elastic Stack for marketing. Following along with this idea that we have web logs in Elastic Search that we can then run queries against, we can use that data to find things like drill paths, how people found our website and where they came from, and even what device they're using or what part of the world they're coming from. This way, the marketing team can really gain a lot of insight into the efforts that they have on trying to get people to our website to then sign up or buy a product, or whatever the case may be, and this is all made possible because of the data that's being ingested is so rich with information that helps our marketing team hone in their efforts and kind of use their dollars wisely. You can probably also guess that, operationally, this information and being able to react to it in real time would be very useful for your operation center. Now, whether that is monitoring a server, or a cluster of servers, that support a web app or something like a factory that has sensors all over with machines running, being able to ingest the data and then present it in alerts based on anomalies, or then kick off other actions, is really, really useful, especially if you have something that is running with a lot of volume, whether that's people on a website, people using an app, streaming videos, whatever the case may be. Being able to ingest data in near real time and react to it can really help smooth things out from an operational standpoint as well as identify weaknesses in your systems that you can then improve later on, and of course, there is Search, the actual ability to use Elastic Search as a search engine. Now, Elastic Search was built with the idea of providing a great search engine, and today, it is the most widely used one in enterprise space. Now, as you'll see, one of the key points of focus here is how this platform is easily able to parse search queries and retrieve related results based on basically any type of data that you may be looking for.

Elasticsearch Concepts

Cluster (collection of nodes), like developing, staging and production cluster.

A picture containing text, screenshot, rectangle, font

Description automatically generatedA screenshot of a computer

Description automatically generated with low confidence

- [Instructor] Before we dig in and start using Elasticsearch, I thought it would make sense to cover all the concepts here so that you have a general understanding of how the pieces fit together. We'll first have our cluster, and as you can imagine, a cluster is a collection of nodes. The cluster has a unique name with a default of Elasticsearch. A typical way that you would structure this is to have a development and staging and then a production cluster, but it's totally okay to just have a single cluster with a single node especially when you're just getting set up like this. Now, the node is a part of a cluster that stores the data and it provides search index capabilities and has its own unique name. Now, the nodes contain indexes, and an index is a collection of similar documents such as customer data, product information, or maybe orders from your website. Now, the node names are all in lowercase and you can have as many of them as you want. When you're doing almost anything in ElasticSearch you're going to be referencing an index so it's important to have a consistent naming pattern for all of them in your cluster. At the base unit, you have a document, which could be for a single customer or order, or an event, let's say coming from an app or a website. Now, these documents are stored in the JSON format and are physically residing inside of your index. Now, the index, in order to be scalable, has to be distributed and it does so using shards and replicas. Now, a replica is a segment of an index and a shard is a portion of that index. Because of its nature, a replica can never be located on the same node as the primary shard that it's a backup for. Now, the default when creating an index is to have just one shard and one replica. Now, this would equal one primary shard and one replica shard distributed across two different nodes. Now, if we visualize this, we start out with our cluster here which has several different nodes in it. We're zooming into node one where we have two indexes, our customer index and our orders index, which inside of those have the physical documents that have the information that we want to store and then use. Then we have our node two, which is our first shard, and this one also has a customer index and orders index being a backup for node one. Then if we wanted to round it out for kind of a complete setup, we would have three different replicas here and these replicas would be similar to nodes one and two but mostly serving as just backups.

1. Setup

Management > Dev Tools

GET: Retrieving Command

PUT: Creating Command

DELETE: Delete Indices

A screenshot of a computer

Description automatically generated

1. Loading Data

**BULK LOADING**

- Gear icon on domain > Applications > Copy endpoint on Elasticsearch > Paste temporarily to notepad/etc.

- Security > Go to Kabana > API keys > Give whatever name > Copy API key > Paste temporarily to notepad/etc.

1. # Get ES endpoint

2. # from https://cloud.elastic.co/home click gear icon next to cluster

3. # copy endpoint next to "Elasticsearch" under Applications and create variable in terminal

4.

5. ES\_HOST="<PASTE ENDPOINT HERE>"

6.

7. # now create API key

8. # click on security then in settings "kibana" link

9. # scroll down to API Keys then "Create API Key"

10. # create variable in terminal

11.

12. ES\_API\_KEY="<PASTE API KEY HERE>"

13.

14. # Create File with Requests (make sure to include new line at end of file)

15. # open VI and create a file called "reqs"

16. vi reqs

17.

18. # type "i" then copy/paste this data into vi

19.

20. { "index" : { "\_index" : "my-test", "\_id" : "1" } }

21. { "col1" : "val1"}

22. { "index" : { "\_index" : "my-test", "\_id" : "2" } }

23. { "col1" : "val2"}

24. { "index" : { "\_index" : "my-test", "\_id" : "3" } }

25. { "col1" : "val3" }

26.

27. # hit esc

28. # type :wq to save the file and exit vi

29. # type ls to see the file is there

30.

31.

32. # using curl, upload data file to cluster and create index

33.

34. curl -XPOST -i -k \

35. -H "Content-Type: application/x-ndjson" \

36. -H "Authorization: ApiKey $ES\_API\_KEY" \

37. $ES\_HOST/\_bulk --data-binary "@reqs"; echo

38.

39.

40. # Login to cluster in web portal, go to dev tools, look for indices and return our data

41. GET /\_cat/indices?v

42. GET /my-test

43. GET /my-test/\_doc/1 #\_doc/1 or 2 or 3 for id=1, 2, 3

44.

A screenshot of a computer

Description automatically generated

**LOADING SAMPLE DATA**

1. # Download from Elastic or use Exercise Files

2. # If necessary unzip the accounts.zip file

3. # https://download.elastic.co/demos/kibana/gettingstarted/accounts.zip

4.

5. # change to downloaded folder

6. cd <path to folder>

7.

8. # Inspect Accounts.json

9. head accounts.json

10.

11. # Load via curl, notice the endpoint which creates the 'bank' index

12. curl -XPOST -i -k \

13. -H "Content-Type: application/x-ndjson" \

14. -H "Authorization: ApiKey $ES\_API\_KEY" \

15. $ES\_HOST/bank/\_bulk?pretty --data-binary @accounts.json; echo #no need to enter/setup anything, here bank isn’t existed yet but you can directly run this command (bank will be created automatically).

16.

17.

18. # check inside ES, ctrl+enter to automatically click the play button

19. GET /\_cat/indices

20. GET /bank

21.

22. # set index pattern in Kibana

23. Management > Stack Management > Kibana > Data Views > Create data view > bank

24.

**A screenshot of a computer

Description automatically generated**

**SETTING DATA TYPES**

4 main datatypes: Core (binary, text, numeric, bool, range), Complex (array or object or nested array of JSON objects), Geo (shaped and point), specialized (IP address, autocomplete, tokens in string)

1. # Download sample logs from Elastic or use Exercise Files

2. # If necessary unzip/tar logs.jsonl.gz

3. # https://download.elastic.co/demos/kibana/gettingstarted/logs.jsonl.gz

4.

5.

6. # Add mapping for lat/lon geo properties for logs

7. PUT /logstash-2015.05.18

8. {

9. "mappings": {

10. "properties": {

11. "geo": {

12. "properties": {

13. "coordinates": {

14. "type": "geo\_point"

15. }

16. }

17. }

18. }

19. }

20. }

21.

22. # Create two more to simulate daily logs

23. PUT /logstash-2015.05.19

24. {

25. "mappings": {

26. "properties": {

27. "geo": {

28. "properties": {

29. "coordinates": {

30. "type": "geo\_point"

31. }

32. }

33. }

34. }

35. }

36. }

37. PUT /logstash-2015.05.20

38. {

39. "mappings": {

40. "properties": {

41. "geo": {

42. "properties": {

43. "coordinates": {

44. "type": "geo\_point"

45. }

46. }

47. }

48. }

49. }

50. }

51. # Check out structure of log data

52. head logs.jsonl

53.

54. # Import log files

55. curl -XPOST -i -k \

56. -H "Content-Type: application/x-ndjson" \

57. -H "Authorization: ApiKey $ES\_API\_KEY" \

58. $ES\_HOST/\_bulk?pretty --data-binary @logs.jsonl; echo

59.

60.

61. # Check ES for data

62. GET /\_cat/indices/logstash-\*

63.

64. # Change default index pattern in Kibana

65.

66. # Load Shakespeare data (download from elastic or find in the Exercise Files)

67. # Check out shakespeare.json

68. # https://download.elastic.co/demos/kibana/gettingstarted/shakespeare\_6.0.json

69.

70. head shakespeare.json

71.

72. # Shakespeare Schema

73. {

74. "line\_id": INT,

75. "play\_name": "String",

76. "speech\_number": INT,

77. "line\_number": "String",

78. "speaker": "String",

79. "text\_entry": "String",

80. }

81.

82. # Create Shakespeare index with data types

83. PUT /shakespeare

84. {

85. "mappings" : {

86. "properties" : {

87. "speaker" : {"type": "keyword" },

88. "play\_name" : {"type": "keyword" },

89. "line\_id" : { "type" : "integer" },

90. "speech\_number" : { "type" : "integer" }

91. }

92. }

93. }

94.

95. # Load Shakespeare data

96. curl -XPOST -i -k \

97. -H "Content-Type: application/x-ndjson" \

98. -H "Authorization: ApiKey $ES\_API\_KEY" \

99. $ES\_HOST/shakespeare/\_bulk?pretty --data-binary @shakespeare.json; echo

100.

101. # Check out index in ES

102. GET /shakespeare

103. GET /\_cat/indices

104. GET shakespeare/\_doc/2

105.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

1. Querying Data

Like SQL queries, except done on documents loaded in ES!

“match” – similar to WHERE clause in SQL

1. # show me everything

2. GET bank/\_search

3.

4. # find CA accounts only

5. GET bank/\_search

6. {

7. "query": {

8. "match": {

9. "state": "CA"

10. }

11. }

12. }

13.

14. # find "Techade" accounts in CA only

15. GET bank/\_search

16. {

17. "query": {

18. "bool": {

19. "must": [

20. { "match": {"state": "CA"} },

21. { "match": {"employer": "Techade"}}

22. ]

23. }

24. }

25. }

26.

27. # find non "Techade" accounts outside of CA

28. GET bank/\_search

29. {

30. "query": {

31. "bool": {

32. "must\_not": [

33. { "match": {"state": "CA"} },

34. { "match": {"employer": "Techade"}}

35. ]

36. }

37. }

38.

39.

40. # let's combine them to search for non "Techade" accounts inside CA

41. GET bank/\_search

42. {

43. "query": {

44. "bool": {

45. "must": [

46. {"match": {

47. "state": "CA"

48. }}

49. ],

50. "must\_not": [

51. { "match": {"employer": "Techade"}}

52. ]

53. }

54. }

55. }

56.

57. # Boost results for Smith

58. GET bank/\_search

59. {

60. "query": {

61. "bool": {

62. "should": [ #should match the state of CA and may have last name of Smith, but boost ‘Smith’ over the state (with priority 3) – hence differ than match as match is complete AND

63. { "match": {"state": "CA"} },

64. { "match": {

65. "lastname": { #subquery

66. "query": "Smith",

67. "boost": 3

68. }

69. }

70. }

71. ]

72. }

73. }

74. }

75.

A screenshot of a computer

Description automatically generated

“term” – look for numeric values and other keyword words.

1. # Term Query

2. GET bank/\_search

3. {

4. "query": {

5. "term": {

6. "account\_number": 516

7. }

8. }

9. }

10.

11. # Returns null because "state" is a text field (hence not an exact match)

12. GET bank/\_search

13. {

14. "query": {

15. "term": {

16. "state": "RI"

17. }

18. }

19. }

20.

21. # This works because it uses the "analysis" (match) process

22. GET bank/\_search

23. {

24. "query": {

25. "match": {

26. "state": "RI"

27. }

28. }

29. }

30.

31. # Terms can return multiple results (range queries

32. GET bank/\_search

33. {

34. "query": {

35. "terms": { #termS, not term if want to retrieve more than 1 account

36. "account\_number": [516,851] #Retrieve acc. No. 516 AND 851

37. }

38. }

39. }

40.

41.

42.

43.

44. # Range Queries

45. ## gte = Greater-than or equal to

46. ## gt = Greater-than

47. ## lte = Less-than or equal to

48. ## lt = Less-than

49.

50. # Show all accounts between 516 and 851, boosting the importance

51. GET bank/\_search

52. {

53. "query": {

54. "range": {

55. "account\_number": {

56. "gte": 516,

57. "lte": 851,

58. "boost": 2 #boosted for 516-851

59. }

60. }

61. }

62. }

63.

64. # Show all account holders older than 35

65. GET bank/\_search

66. {

67. "query": {

68. "range": {

69. "age": {

70. "gt": 35

71. }

72. }

73. }

74. }

75.

A screenshot of a computer

Description automatically generated

Analysis and Tokenization (break down string)

1. # Basic Example, each token captures one word correctly (with correct capitalization as well)

2. GET bank/\_analyze

3. {

4. "tokenizer" : "standard",

5. "text" : "The Moon is Made of Cheese Some Say"

6. }

7.

8. # Mixed String

9. GET bank/\_analyze

10. {

11. "tokenizer" : "standard",

12. "text" : "The Moon-is-Made of Cheese.Some Say$"

13. }

14.

15. # Use the letter tokenizer

16. GET bank/\_analyze

17. {

18. "tokenizer" : "letter",

19. "text" : "The Moon-is-Made of Cheese.Some Say$"

20. }

21.

22. # How about a URL

23. GET bank/\_analyze

24. {

25. "tokenizer": "standard",

26. "text": "you@example.com login at https://bensullins.com attempt"

27. }

28.

29. GET bank/\_analyze

30. {

31. "tokenizer": "uax\_url\_email",

32. "text": "you@example.com login at https://bensullins.com attempt"

33. }

34.

35. # Where it breaks, two fields with diff analyzers

36. PUT /idx1

37. {

38. "mappings": {

39. "properties": {

40. "title": {

41. "type": "text",

42. "analyzer" : "standard"

43. },

44. "english\_title": {

45. "type": "text",

46. "analyzer": "english"

47. }

48. }

49. }

50. }

51.

52. GET idx1 #different analyzer would have different result as they analyze data differently

53.

54. GET idx1/\_analyze #result in ‘bears’

55. {

56. "field": "title",

57. "text": "Bears"

58. }

59.

60. GET idx1/\_analyze #result in ‘bear’

61. {

62. "field": "english\_title",

63. "text": "Bears"

64. }

65.

A screenshot of a computer

Description automatically generated

1. Aggregation – Analysing Your Data

Basic Aggregations

1. # Count of Accounts by State

2. # Must be keyword field

3. GET bank/\_search

4. {

5. "size": 0, #only want the aggregations data, not the results (0 result should be returned)

6. "aggs": {

7. "states": {

8. "terms": {

9. "field": "state.keyword"

10. }

11. }

12. }

13. }

14.

15. # Add average balance in each state

16. # Nesting the metric inside the agg

17. GET bank/\_search

18. {

19. "size": 0,

20. "aggs": {

21. "states": {

22. "terms": {

23. "field": "state.keyword"

24. },

25. "aggs": {

26. "avg\_bal": {

27. "avg": {

28. "field": "balance"

29. }

30. }

31. }

32. }

33. }

34. }

35.

36. # Breakdown further with Nesting

37. GET bank/\_search

38. {

39. "size": 0,

40. "aggs": {

41. "states": {

42. "terms": {

43. "field": "state.keyword"

44. },

45. "aggs": {

46. "avg\_bal": {

47. "avg": {

48. "field": "balance"

49. }

50. },

51. "age":{

52. "terms": {

53. "field": "age"

54. }

55. }

56. }

57. }

58. }

59. }

60.

61. # Add avg\_price metric to lowest level

62. GET bank/\_search

63. {

64. "size": 0,

65. "aggs": {

66. "states": {

67. "terms": {

68. "field": "state.keyword"

69. },

70. "aggs": {

71. "avg\_bal": {

72. "avg": {

73. "field": "balance"

74. }

75. },

76. "age":{

77. "terms": {

78. "field": "age"

79. },

80. "aggs": {"avg\_bal": {"avg": {"field": "balance"} }

81. }

82. }

83. }

84. }

85. }

86. }

87.

88.

89. ## Get stats about bank balances

90. ## Size=1 to omit search results

91. GET bank/\_search

92. {

93. "size": 1,

94. "aggs": {

95. "balance-stats": {

96. "stats": {

97. "field": "balance"

98. }

99. }

100. }

101. }

102.

A screenshot of a computer

Description automatically generated

Filtering Aggregations

1. # Count of Accounts by State

2. # Must be keyword field

3. GET bank/\_search

4. {

5. "size": 0,

6. "aggs": {

7. "states": {

8. "terms": {

9. "field": "state.keyword"

10. }

11. }

12. }

13. }

14.

15. # This is the equivalent of using match\_all

16. GET bank/\_search

17. {

18. "size": 0,

19. "query": {

20. "match\_all": {}

21. },

22. "aggs": {

23. "states": {

24. "terms": {

25. "field": "state.keyword"

26. }

27. }

28. }

29. }

30.

31. # Aggs work in the context of the query, so apply a filter like normal

32. GET bank/\_search

33. {

34. "size": 0,

35. "query": {

36. "match": {

37. "state.keyword": "CA"

38. }

39. },

40. "aggs": {

41. "states": {

42. "terms": {

43. "field": "state.keyword"

44. }

45. }

46. }

47. }

48.

49. # You can also filter on terms

50. GET bank/\_search

51. {

52. "size": 0,

53. "query": {

54. "bool": {

55. "must": [

56. {"match": {"state.keyword": "CA"}},

57. {"range": {"age": {"gt": 35}}}

58. ]

59. }

60. },

61. "aggs": {

62. "states": {

63. "terms": {

64. "field": "state.keyword"

65. }

66. }

67. }

68. }

69.

70. # Lets add a metric back in

71. GET bank/\_search

72. {

73. "size": 0,

74. "query": {

75. "bool": {

76. "must": [

77. {"match": {"state.keyword": "CA"}},

78. {"range": {"age": {"gt": 35}}}

79. ]

80. }

81. },

82. "aggs": {

83. "states": {

84. "terms": {

85. "field": "state.keyword"

86. },

87. "aggs": {"avg\_bal": {"avg": {"field": "balance"} }}

88. }

89. }

90. }

91.

92.

93. # Look at state avg and global average

93b. # Aggregation on different level, can be difficult to achieve in programming languages and databases like SQL

94. GET bank/\_search

95. {

96. "size": 0,

97. "aggs": {

98. "state\_avg": { #level 1 aggregation

99. "terms": {

100. "field": "state.keyword"

101. },

102. "aggs": {"avg\_bal": {"avg": {"field": "balance"}}}

103. },

104. "global\_avg": { #level 2 aggregation

105. "global": {},

106. "aggs": {"avg\_bal": {"avg": {"field": "balance"}}}

107. }

108. }

109. }

110.

A screenshot of a computer program

Description automatically generated with medium confidence

A screenshot of a computer program

Description automatically generated with medium confidence

^global and state\_avg

Percentile and Histograms

1. # Look at the percentiles for the balances

2. GET bank/\_search

3. {

4. "size": 0,

5. "aggs": {

6. "pct\_balances": {

7. "percentiles": {

8. "field": "balance",

9. "percents": [

10. 1,

11. 5,

12. 25,

13. 50,

14. 75,

15. 95,

16. 99

17. ]

18. }

19. }

20. }

21. }

22.

23.

24. # We can use the percentile ranks agg for checking a individual values

25. GET bank/\_search

26. {

27. "size": 0,

28. "aggs": {

29. "bal\_outlier": {

30. "percentile\_ranks": {

31. "field": "balance",

32. "values": [35000,50000]

33. }

34. }

35. }

36. }

37.

38. # Similarly we can create a histogram

39. GET bank/\_search

40. {

41. "size": 0,

42. "aggs": {

43. "bals": {

44. "histogram": {

45. "field": "balance",

46. "interval": 500

47. }

48. }

49. }

50. }

51.

A screenshot of a computer

Description automatically generated

1. Presenting Your Insights

Kibana (Analytics) – Analyse Log Data.

**Kibana overview and setup**

Selecting transcript lines in this section will navigate to timestamp in the video

- So, along with Elasticsearch, the actual engine that stores our documents, and allows us to use APIs to put data in and pull it back out and manipulate it, do all the fun things we've been doing, there is Kibana, which is a really great way to interact with this data in a visual kind of interface, one that doesn't require a lot of coding. So, to get set up here, the way I want to get this working for you, is to analyze some log data. First, we're going to click on our little hamburger menu, and go down to Stack Management. Now, if you remember earlier, we actually imported some data here, and we'll go down to Kibana and Data Views, and that data was from Logstash. That's an actual engine that logs things, and has web requests coming to a web server. So, one of the key, kind of most common use cases for Elastic, probably some data that you have available at your organization. So, under Data Views here, what I want to do is click Create Data View, and I want to give it a name, logstash. You can give whatever you want but for me this makes the most sense. And I'll just start typing logstash-. So, the format of the data that we built before, each index was for a different day. So, you can kind of see the days listed here of the data we have available to us. I'm going to give it the index pattern of logstash-, so anything coming in that matches that pattern will automatically be added and then in Kibana, automatically update our visualizations and everything else. Now with this data, it has a timestamp associated with it, so that's really useful because that helps us do a lot of really interesting things on the analytics side. So, once we create our data view here, we'll click Save data view to Kibana. Now, once that's saved, you can see what it did here, but I'm less interested in actually digging into this, and I'm more interested in going up to Kibana itself. So, Kibana here on the Elastic Cloud is called Analytics, and in there I just want to hop over to the Discover tab. So, in the Discover tab, you'll see in the top left which data view we're looking at, and you can see these match the ones that were in there before. Here is the one that we just created for Logstash. We click on that, and I've already pre-selected the dates here, but if you haven't done that, or if you don't see any results up in the top right, you'll see the date filter for this, because this is a time-based dataset, there's the timestamp field. Go ahead and update that to May 17th, 2015 at 1700 hours, to May 20th, 2015 at 1600 hours, and you'll be seeing the same thing I am here, and this is just a nice way to kind of explore. I can click on Fields, I can see what's in them. I can even pull in field statistics, if I want, where it looks at each of the individual ones and sees the distribution among them. I mean, there's just so much and it's so easy to just pop in here to individual documents, as well as to look at individual fields. So, that is the basics of getting this set up. Now, because you're using Elastic Cloud, Kibana is already installed and configured and connected to your cluster. So, it is really nice and easy, and it's a great way to start learning how this works, without being burdened by all the different setup and configuration stuff, which you may not be interested in.

A screenshot of a computer

Description automatically generated

**Creating visualizations in Kibana**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] So here in Kibana, let's create some visualizations. I'm going to click on my hamburger menu and then go down to where it says visualize library. Here is where you actually create data visualizations. So we'll click on this big, beautiful button right there. And I'm going to use this feature here called Lens. I really like it because it is an easy way to explore the data. When you click on that, it gives you this interface here, but it doesn't have the right data view, so let's click on that and go down to Logstash. Now we have all of the data that is available in that data view, which came from those files we loaded previously. And the beauty of this is I can literally just drag things around and see what happens. I can change the chart type right here. And it's just a really nice, intuitive interface, much better than the previous version. So first, what I want to do is create a chart which shows me how many visits we're getting to our website over time. So a simple way to do that is to drag records over here. And I already have the dates filtered up in the top right, so if you're not seeing this exact thing, go ahead and change that there. And in there I can change the type of chart. And so I could do a line chart, which would be a very good one, or I could do an area chart, for example, if you want to make it a little bit nicer to look at. And this is a typical way that we would look at data over time. Usually what we're looking for visually is the steepness of the angle between the two data points, not necessarily the dots or the size of the bars, whatever. And so, you can do that here, and then of course we can change this if we wanted to, but for me, this is the simple way to create a time series analysis. So I'll click save here and I'll give it a title of Visits by Hour. I'm not going to put it on a dashboard yet, 'cause we'll look at how to do that later. I'll just click save and add to our library. So there we have it. Now if I click up in the top left, I go back to my library, I have my chart. I can jump back in, and there it is. Cool, so that's step one. Now let's create a map. I'll create another visualization. Here I'm just going to click on maps. In fact, if I were in Lens, and then I wanted to do maps, well, I'll show you what happens. You go in here, and we want to find our geo coordinates, which is the field we have. It's already typed, notice the different color here. I'm just going to drag that on. It's going to say, "Hey, I can't do that, sorry." Go ahead and drag it down here to open it in maps, and it jumped me over immediately. So I could have just started with maps, but I kind of like the idea of just always using the same Lens. And so now I'm here, it created a new layer, and within that layer I can do some things. So I'm going to zoom in here. We're a US-based company in this example. So all these dots, they don't make a whole lot of sense, right? It just looks like a mess. There's really nothing to it, so we need to edit that layer. I'll click on my Logstash layer here. I'll go down, and instead of using just vector tiles, I'm going to say "Show clusters when results exceed 10,000." So that way, we're going to get bigger bubbles with numbers on them. Ah, that's better, I can actually read that, and it kind of is more insightful. Then beyond that, I'm not just going to fill it with a solid color. Color is one of the key attributes we can use to affect the visualization, so I'm going to choose by value and then I'm going to choose a field here. For this, I'm just going to choose bytes, meaning where did most of the requests come from. I can also adjust the color of the border to a blue that kind of makes more sense, and there you have it. And I could also change this to an orange or a red, or whatever, but blue seems to be a fairly good one there. All right, so now we've been able to color code by a value, we've been able to manipulate that, and we were also able to change kind of the display, and so I'll save and close this. And what happens now is we're not set on these numbers. As I zoom in, you see that it starts to break it down further. This data is at latitude and longitude, so it actually gets pretty fine-grain when I start to zoom in there. And then as I zoom back out, it's going to aggregate that back up for me. Now of course you could spend countless hours formatting all of this. I'll leave that to you, but all that is done very simply in this interface here for the map. I'll click save and call this Traffic Locations. I'll choose none, and I'll save it, and now I have a map. Okay, so let's create some more visualizations here. I'm going to head back over to my visualize library. You can see the two we've just created. One is with Lens and the other is with maps. I'm going to click create. I'm going to go back to Lens. And this one I just want it to be total visits given the timeframe we're looking at. So I'm going to drag records on over. Gives me this guy, but I'm going to click on my visualization type and I'm going to go down to legacy metric. And there you go, count of records, 14,001. Now I'm going to click on this metric on the right side. And here, instead of count of records, I'm going to say total visits. I'll go ahead and click save and we'll call this Total Visits KPI. We're not going to put it on any specific dashboard yet. Click save and we are good to go. Now we're going to do this a couple more times. I'll go back to my visualize library, create visualization, Lens. This time I'm going to do clientip.keyword. Drag that on over. It gives me this, is not what I want. Give me that legacy metric again. Count of records. But here it went back to that one. I'm going to drag it on to unique count of clientip.keyword. Makes no sense, so instead we're going to call this Unique Visitors. Got it, save, say Unique Visitors KPI. Do not add it to any existing dashboard. Click save. We'll go back and do one more. And this one is going to be for the number of bytes. So if I start typing bytes, you can see it's a number here. I'll drag it over. Instead of vertical bar I want legacy metric. And that's actually fine, median of bytes. I don't mind that, but I want it to show is that bytes, is it kilobytes, megabytes, what is it? So the value format under appearance is going to be actually bytes. So there you go. Now I can leave it as median of bytes or I can change that title if I want. I'm fine with it. I'll click save and I'll say Median Bytes KPI. So KPI, if you're unfamiliar, is key performance indicator. A lot of times you think of it just as a single metric is a way to think of that. There's a lot more science behind it, but in common vernacular, that's what KPIs are. Okay, we've created essentially everything we need here to really get an understanding of the traffic to our website. So there's one more visualization I want to create, and it's just a list of the most popular pages here. I'm going to go into Lens and I'm going to search for title, give me the page title. Drag that on, gives me this thing which I don't want. I'll click on the type of chart and click table. And it tells me essentially how many visits were to each one of these pages. Now this doesn't make quite sense, so I'm going to click on this field here, this metric, and I'm going to go down. I want the 10 rows in descending order. And I'm going to say top 10 pages. Call it like that. So we've got top 10 pages. And then you can say rank by count of records is fine. We're going to not group as other ones as others so that way we don't have that big kind of funky thing on the bottom. And there we go, so now we have our top pages and we have a good list of them so people can actually see what it is. All right, so let's click save and say top pages. Not to any dashboard, click back on our visualization library, and we were able to create many visualizations here. And this new Lens feature is really great. It's almost Tableau-esque, if you're familiar with Tableau. But play around with this. Create some visualizations if you want. They've got lots of different kinds here, as well as when you click on it it'll show you other types that it recommends, and so I think this is a really great way in Kibana to create different visualizations.

**Creating dashboards in Kibana**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Another great feature in Kibana is the ability to create dashboards. So we already have bevy of wonderful data visualizations here, and what I want to do is essentially combine those to a single page, that is essentially what the definition of a dashboard is. It's a single page of charts explaining a single topic, letting the user kind of explore and understand what's happening in a single area. So to get started, we can click this little tooltip right here, but if that's not there, we'll click the hamburger menu dropdown and go over to Dashboard. From here, we will create a Dashboard, and here, we need to bring in some of our data visualizations. We already have our date filter up here, on the top right, which is really nice. So we'll click Add from library and we have all of them here. So I'm going to add the KPIs first, because I want to format those before I kind of bring any of the other ones in. When I click it, notice it just brings these ginormous tiles here which is definitely not what I want. So I'm going to bring this guy over, kind of drag it like that. I'll drag this next one just over here. It's very free flowing. I'm not sure I actually like this. I would kind of prefer it to be more structured than what it is, but easy to get going nonetheless and kind of nice that it's fluid. So, we'll drag. Let's see, total visits on top. That kind of makes sense. Maybe it's probably the most important metric. And then, unique visitors. There you go. We'll hit Cancel. All right, so we have kind of those three there. Let's bring in some new ones here. Let's start with our Visits by Hour, which is our chart. Very nice, that looks good. Now let's add in our Traffic Locations, which is our map. Good, just below that. Let's add in our other one here, Top Pages. And this one kind of went down there. Let's drag this guy off to the right. Actually, let's kind of move this first. We can drag this guy over. There we go, and kind of bring this guy up. So this can be something that a person would scroll through, see, so we got that there. Bring this guy down so it matches there, bing, and then a person can kind of scroll and see that stuff. Good to go. And then for the map, maps are generally good when they're bigger. So let's make this bad boy full screen here. So that way, a person can come here. They can see on the left, the different KPIs. They have this here. You could also put the KPIs on top. There's a lot of wiggle room in here. So you know, choose whatever layout you think works best for you. And there, essentially, is all you need to do. Of course, you can go way beyond this in terms of customizing how things look. There's, you know, different titles and things like that. Like, let's say I wanted to get rid of those. I can do that, or I could also get rid of this bit that's inside of the chart there to make this cleaner and nicer. But essentially, I have created a dashboard. Now, what I want to do next is click Save and we'll call this Web Traffic Main. This will be like our main dashboard here. Click Save, and now we have this dashboard. Really beautiful. As the data flows into Elasticsearch, this would automatically update. I have these dates specifically set here because we know we're just working with some sample data, but this could also be dynamically generated to be today, this week, last 15 minutes, et cetera, so it is pretty powerful. You know, this was just an example to show you how that's working but if you had real data flowing in, you would definitely want this to be more sort of real time. And that way, you could put it up on a display in your marketing area or maybe your network operation center, whatever the case may be. Now from here, we also have a few more options. We can share this, we can embed it, we can do PDF, we can do PNG. There's all kinds of ways you can send this to people. You can also have a sort of this view mode, which you can see. You can even go full screen, and this would be really nice if you had it up on sort of a network operations center. So dashboards in Kibana, really straightforward once you've already created your data visualizations, or you could actually just start with the dashboard and it would allow you to create visualizations from right there. But that is essentially it. Make sure to use color wisely on here and that the flow all makes sense, and you have the most important information highlighted for your users.

A screenshot of a computer

Description automatically generated

**Next steps with Elasticsearch**

Selecting transcript lines in this section will navigate to timestamp in the video

- As we've seen, Elasticsearch is much more than just a search engine. In fact, many data analysts, DevOps professionals, and marketing professionals rely on Elasticsearch daily for their work. The journey isn't over, however. There are many more concepts, tools, and platforms to understand before you become truly experienced in this platform. To continue this journey, I recommend checking out other related courses, such as Data Science Foundations: Data Engineering, SQL Tips Tricks and Techniques, Analyzing Big Data with Hive, and any of the top 10 courses I have out there, such as the ones on computational linguistics, machine learning, or business intelligence. With the skills you've learned here, and that you'll get with these additional courses, you are well on your way to becoming a truly powerful data scientist. Feel free to connect with me online, as well as remember, when you free the data, your mind will follow. Thanks for watching this course and I'll see you back here next time.