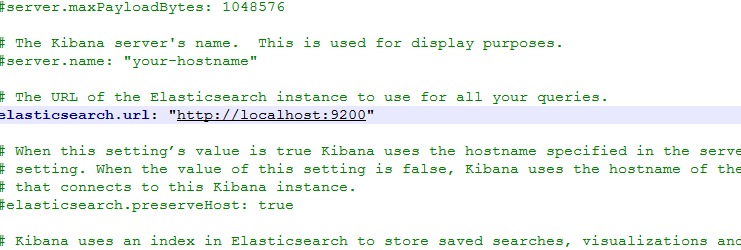
# Download Kibana

Download and unzip Kibana

<https://www.elastic.co/downloads/kibana>

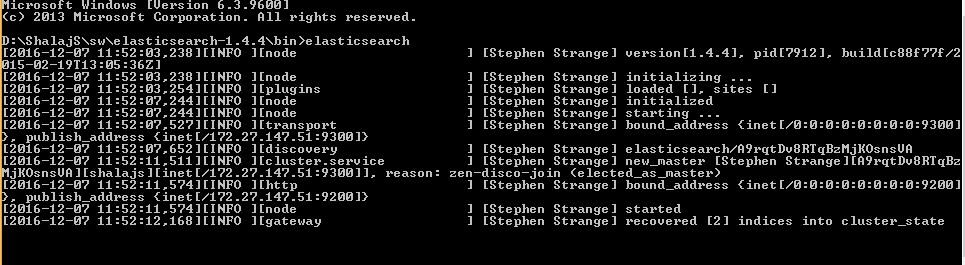
(D:\ShalajS\sw\kibana-5.0.2-windows-x86)

Open config/kibana.yml in an editor and Set elasticsearch.url to point at your Elasticsearch instance

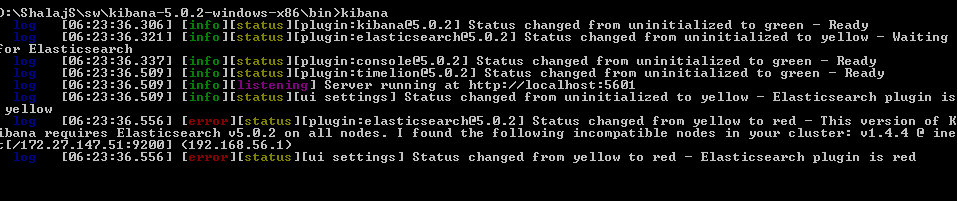


***Note: By default, Kibana connects to the Elasticsearch instance running on localhost. To connect to a different Elasticsearch instance, modify the Elasticsearch URL in the kibana.yml configuration file and restart Kibana , so if kibana and elasticsearch is running on same machine we do not need to add elasticsearch url***

Run elasticsearch

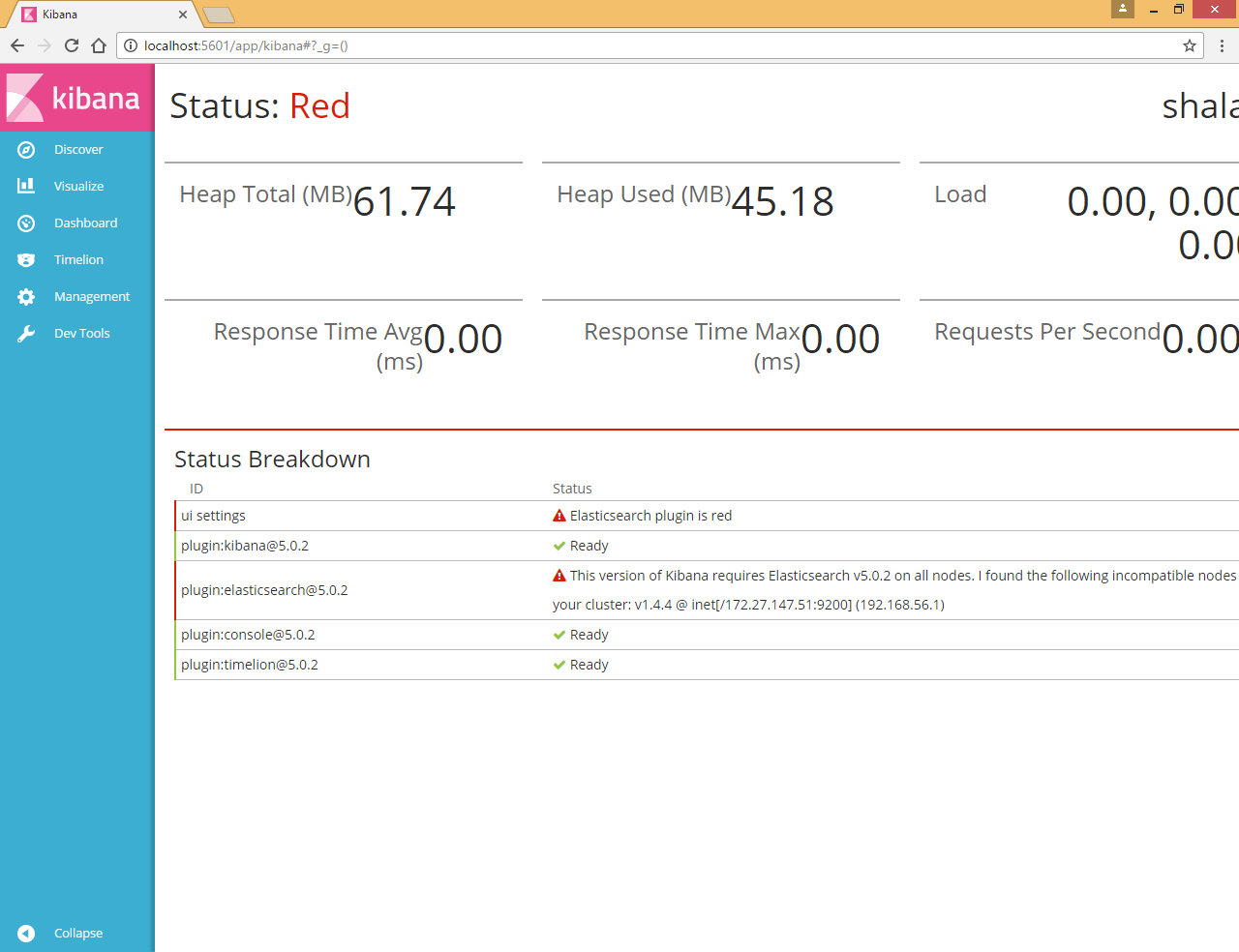


Run bin/kibana (or bin\kibana.bat on Windows)



Open

<http://localhost:5601>

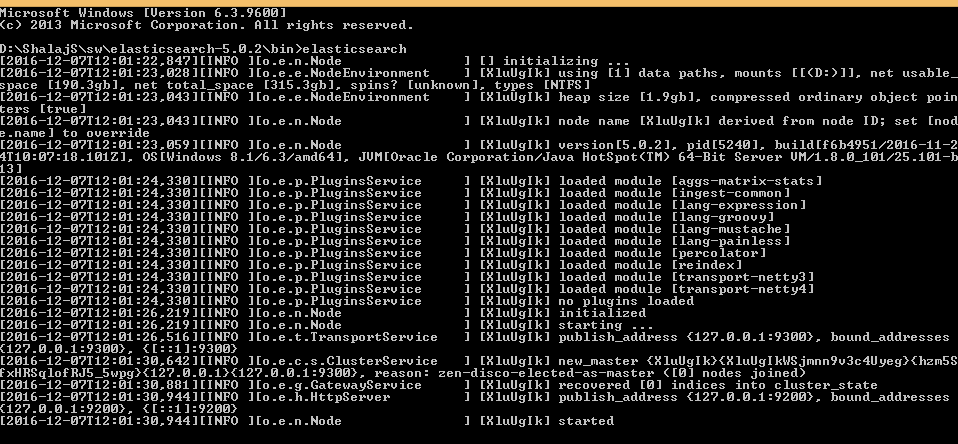


Kibana 5.x requires **Elasticsearch v5.0.2** so need to download new version of elasticsearch

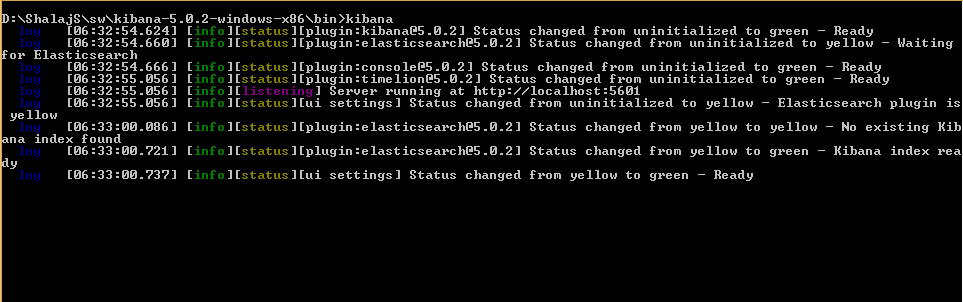
<https://www.elastic.co/downloads/past-releases/elasticsearch-5-0-2>

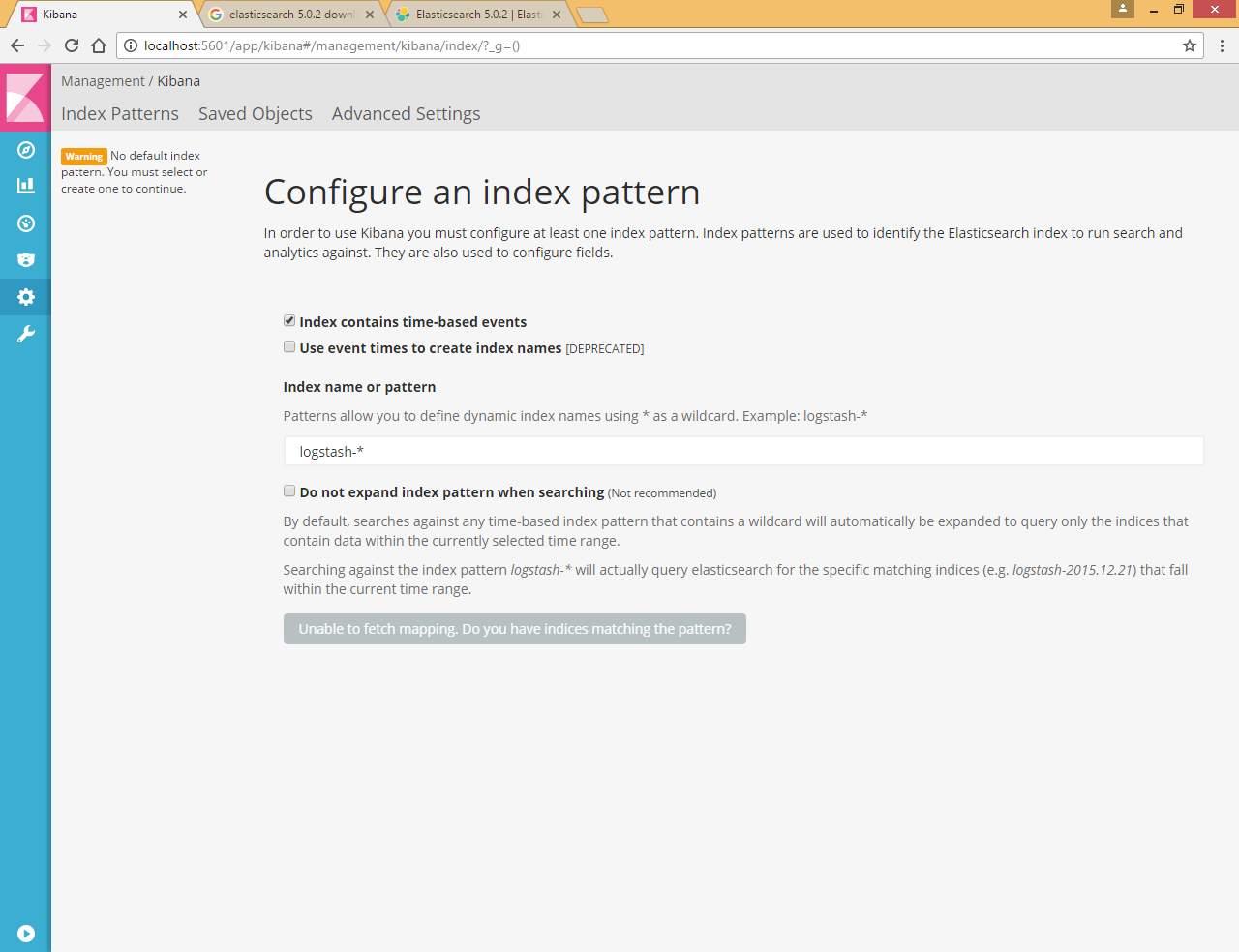
first stop kibana and elastic search

start new version of elasticsearch



Now start kibana





# Load Data in elastic Search

First we need to create new index and type in elasticsearch and need to load some data, so that we can visualize data in elasticsearch

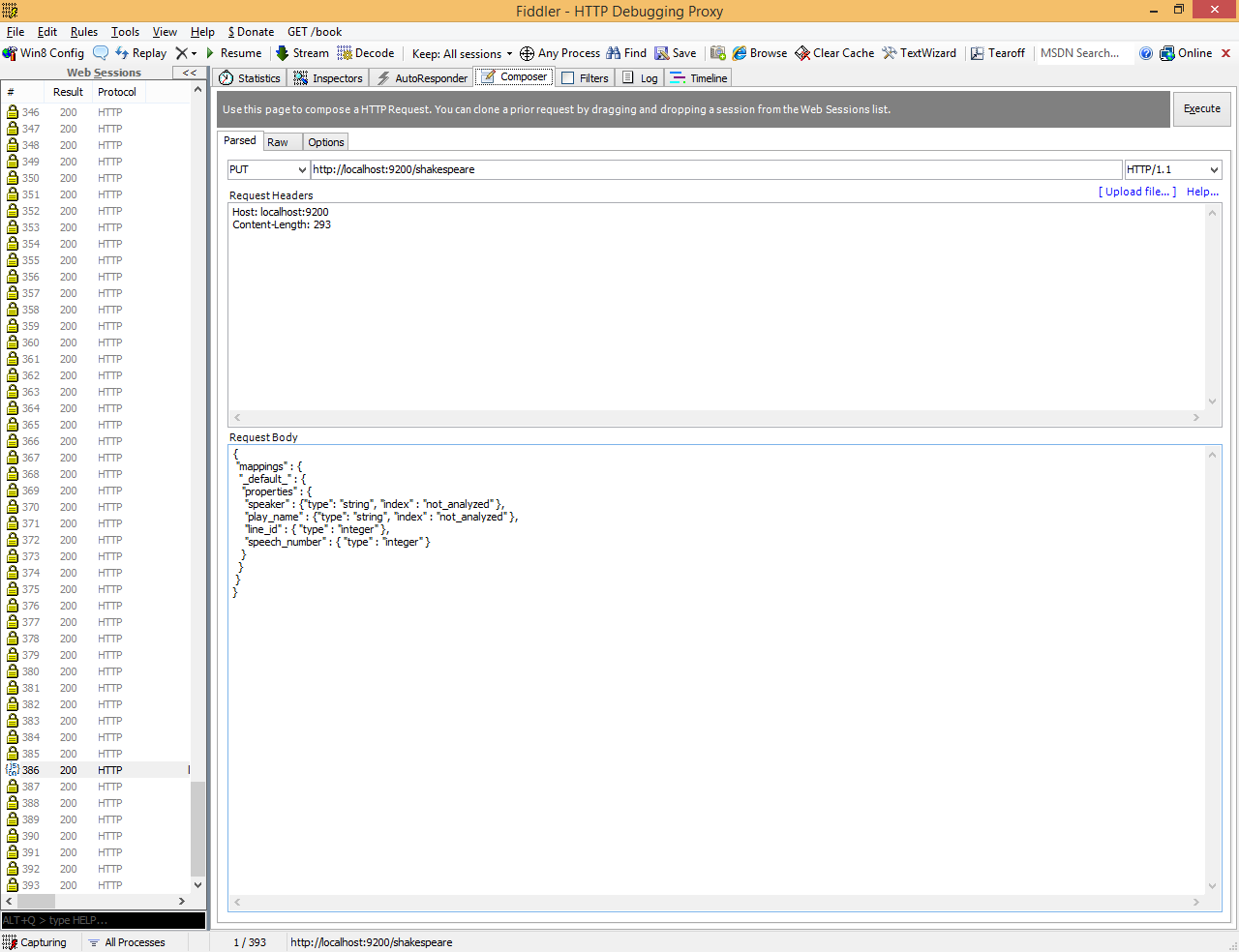
Refer <https://www.elastic.co/guide/en/kibana/current/introduction.html>

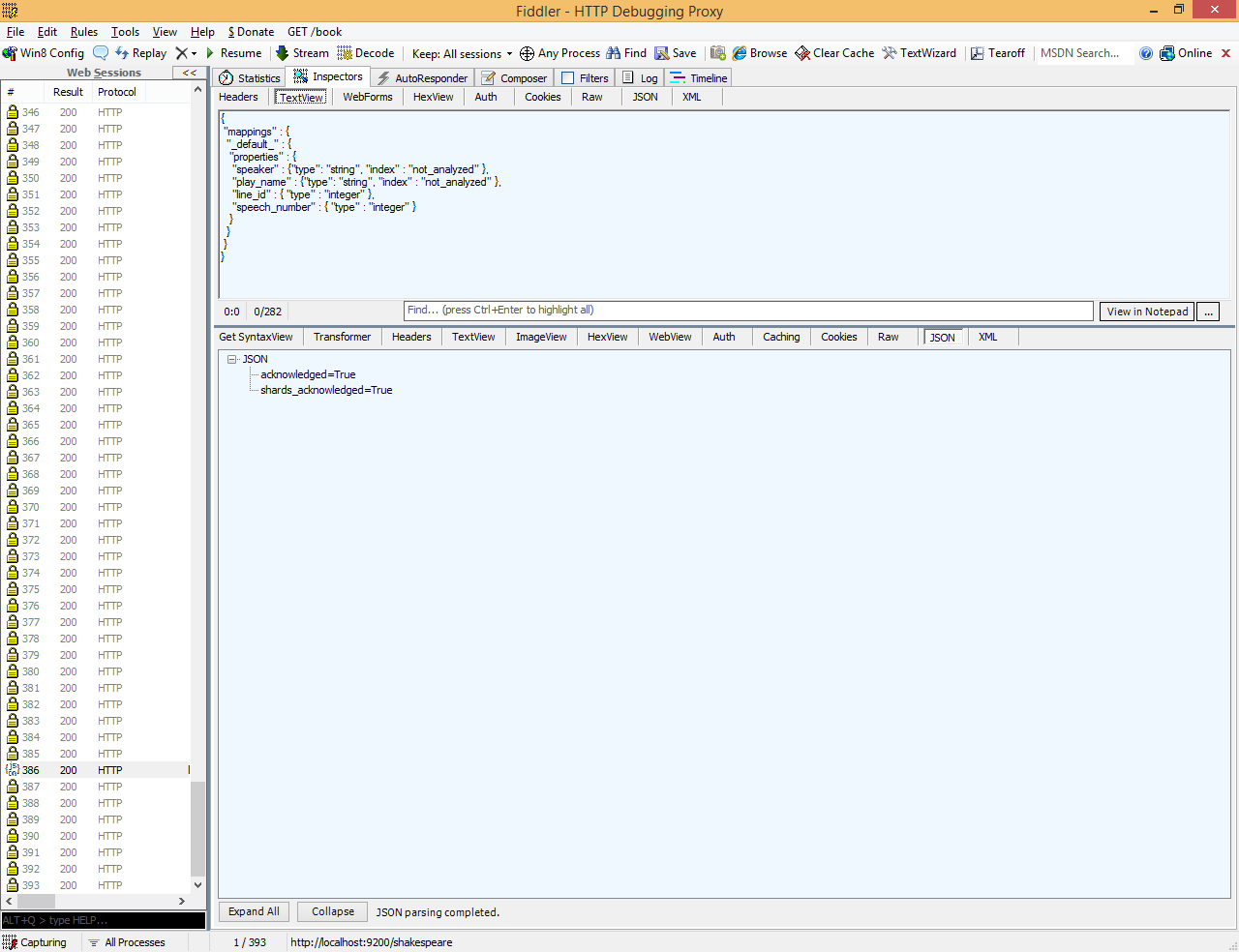
We have three datasets shakespeer.json,accounts.json and log file



Before we load the Shakespeare and logs data sets, we need to set up mappings for the fields. Mapping divides the documents in the index into logical groups and specifies a field’s characteristics, such as the field’s searchability or whether or not it’s tokenized, or broken up into separate words.

|  |
| --- |
| curl -XPUT http://localhost:9200/shakespeare -d '  {  "mappings" : {  "\_default\_" : {  "properties" : {  "speaker" : {"type": "string", "index" : "not\_analyzed" },  "play\_name" : {"type": "string", "index" : "not\_analyzed" },  "line\_id" : { "type" : "integer" },  "speech\_number" : { "type" : "integer" }  }  }  }  }  '; |





This mapping specifies the following qualities for the data set:

* The *speaker* field is a string that isn’t analyzed. The string in this field is treated as a single unit, even if there are multiple words in the field.
* The same applies to the *play\_name* field.
* The *line\_id* and *speech\_number* fields are integers.

The logs data set requires a mapping to label the latitude/longitude pairs in the logs as geographic locations by applying the geo\_point type to those fields.

Use the following commands to establish geo\_point mapping for the logs:

curl -XPUT http://localhost:9200/logstash-2015.05.18 -d '

{

"mappings": {

"log": {

"properties": {

"geo": {

"properties": {

"coordinates": {

"type": "geo\_point"

}

}

}

}

}

}

}

';

curl -XPUT http://localhost:9200/logstash-2015.05.19 -d '

{

"mappings": {

"log": {

"properties": {

"geo": {

"properties": {

"coordinates": {

"type": "geo\_point"

}

}

}

}

}

}

}

';

curl -XPUT http://localhost:9200/logstash-2015.05.20 -d '

{

"mappings": {

"log": {

"properties": {

"geo": {

"properties": {

"coordinates": {

"type": "geo\_point"

}

}

}

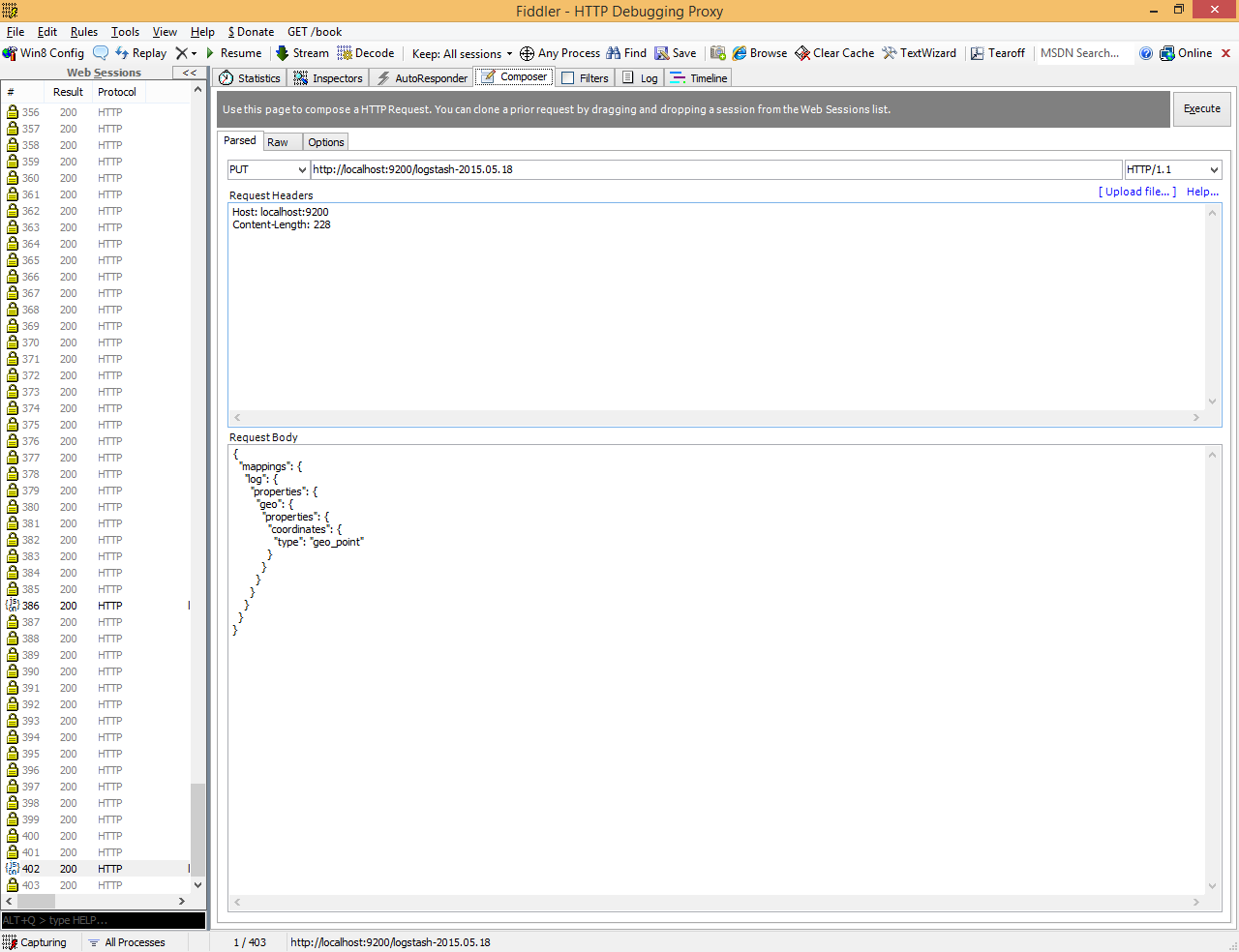
}

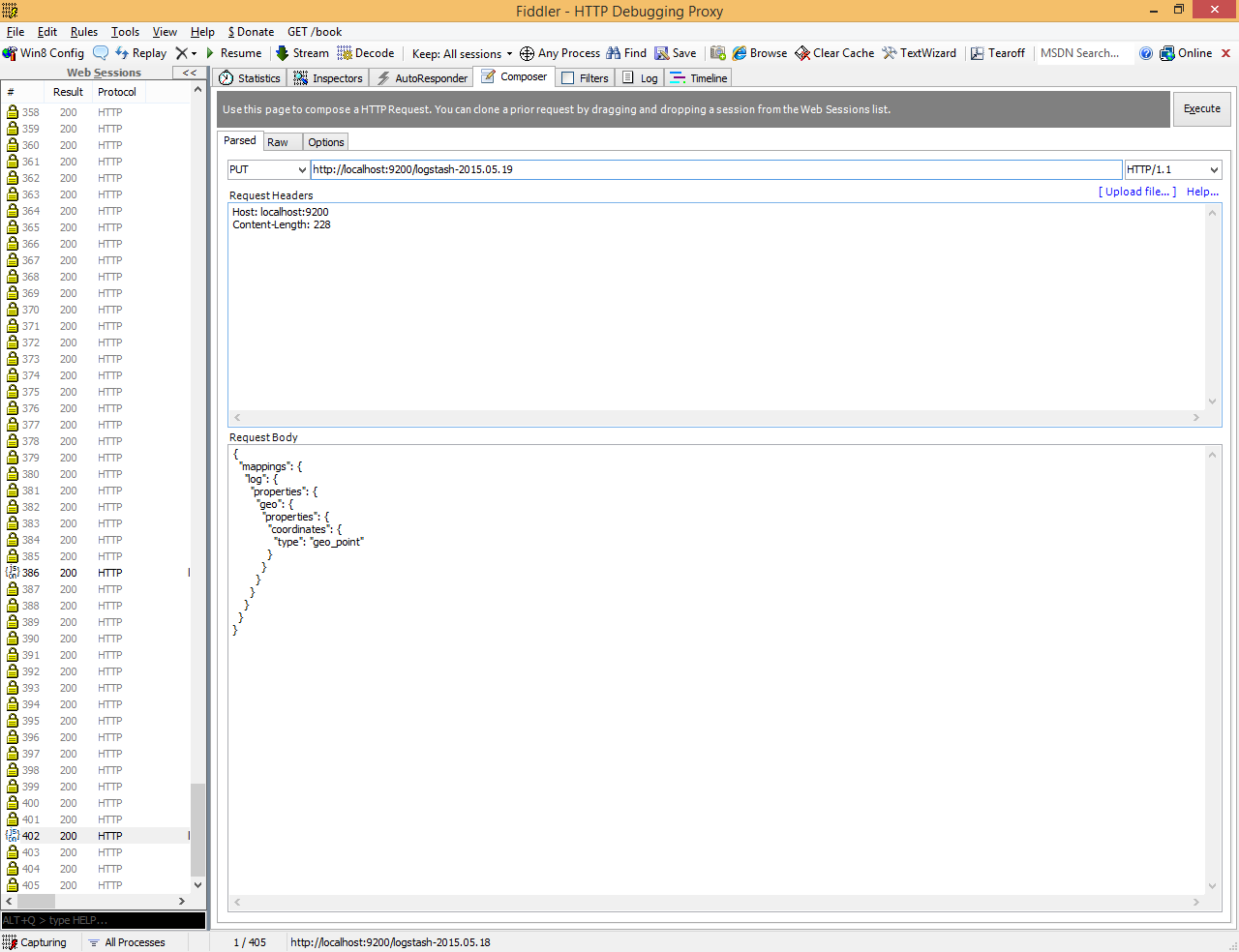
}

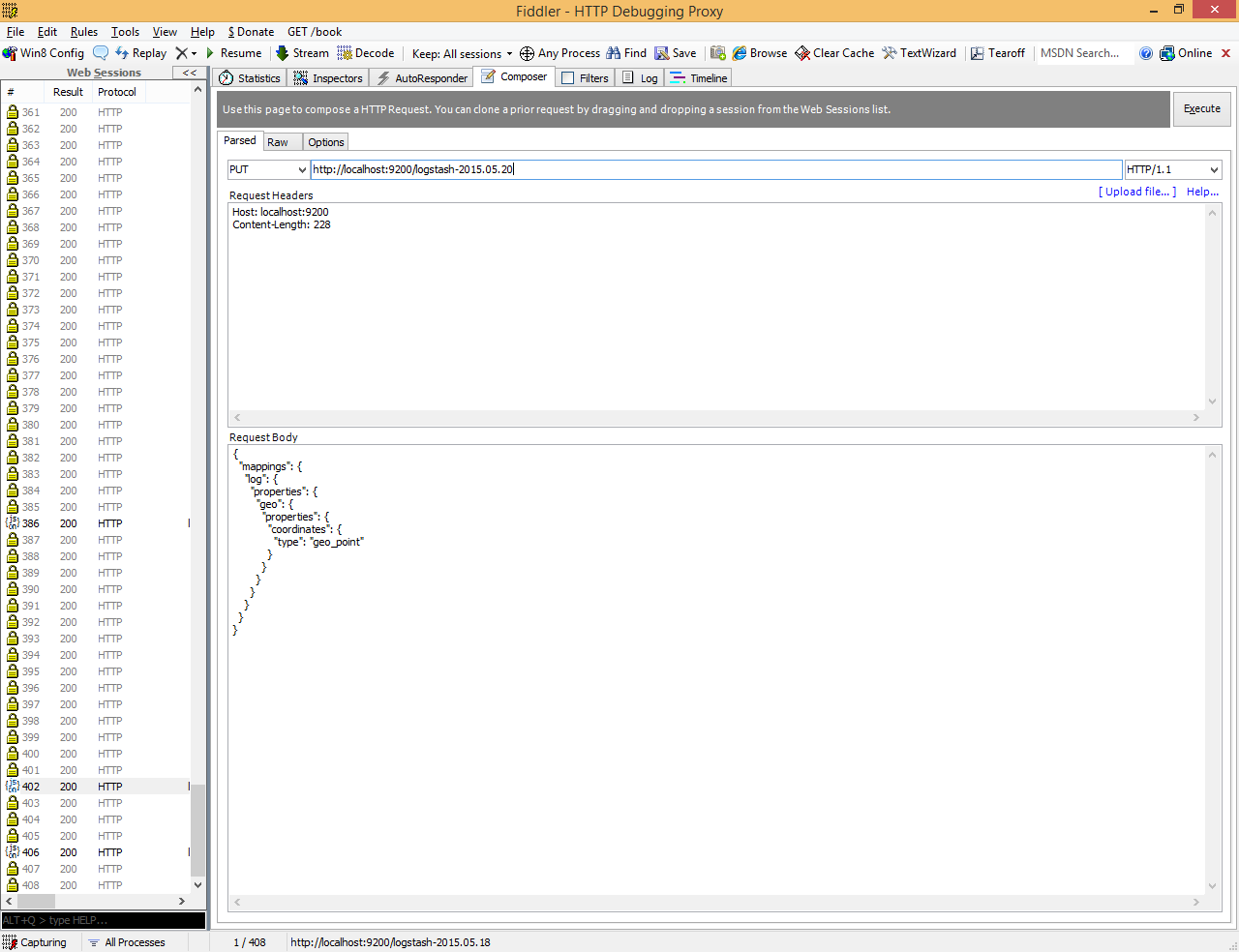
}

}

';





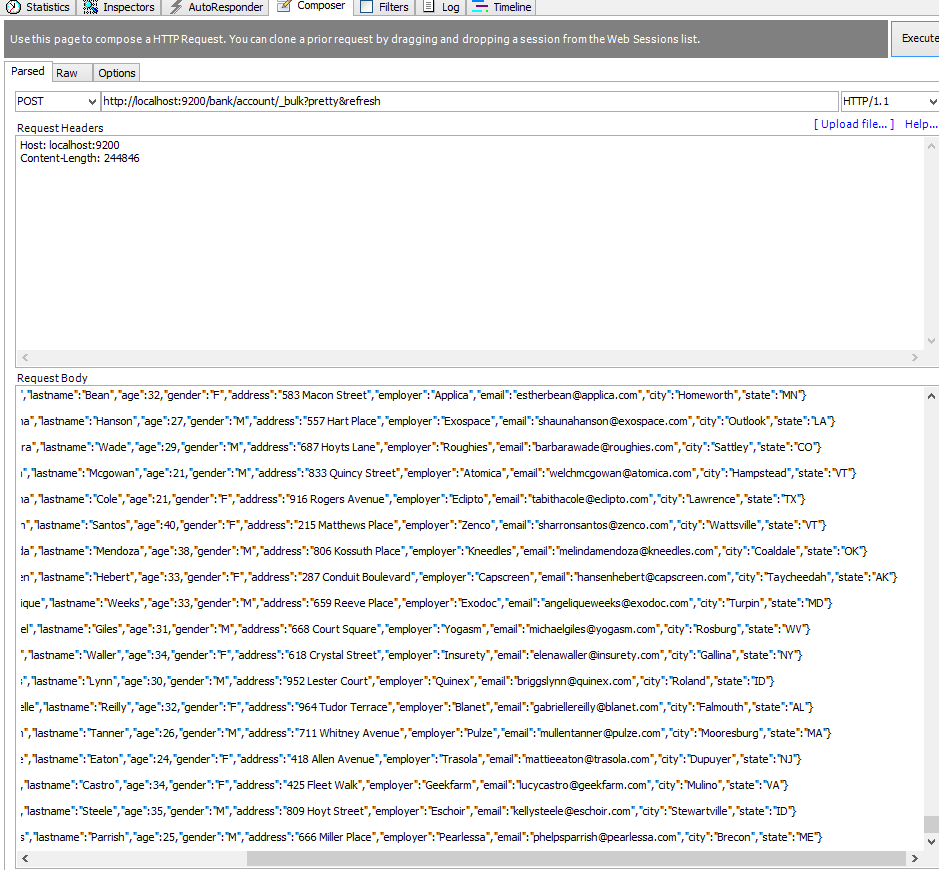


The accounts data set doesn’t require any mappings, so at this point we’re ready to use the Elasticsearch [bulk](https://www.elastic.co/guide/en/elasticsearch/reference/5.0/docs-bulk.html) API to load the data sets with the following commands:

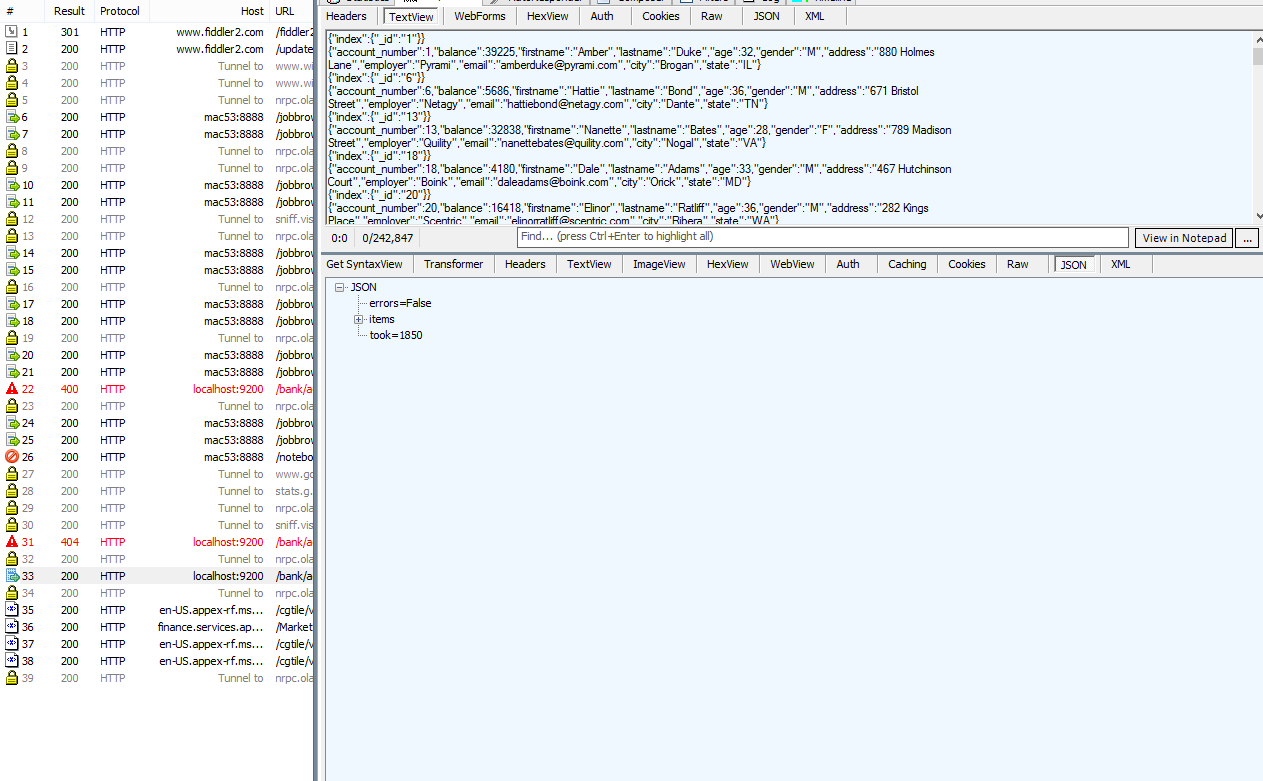
## Load accounts.json file

Open fiddler and post bulk value (<http://localhost:9200/bank/account/_bulk?pretty&refresh>)

Place content of accounts.json in Request body



Now click on localhost:9200 at left frame and see result



So we have created index **bank** and type **account** under elasticsearch

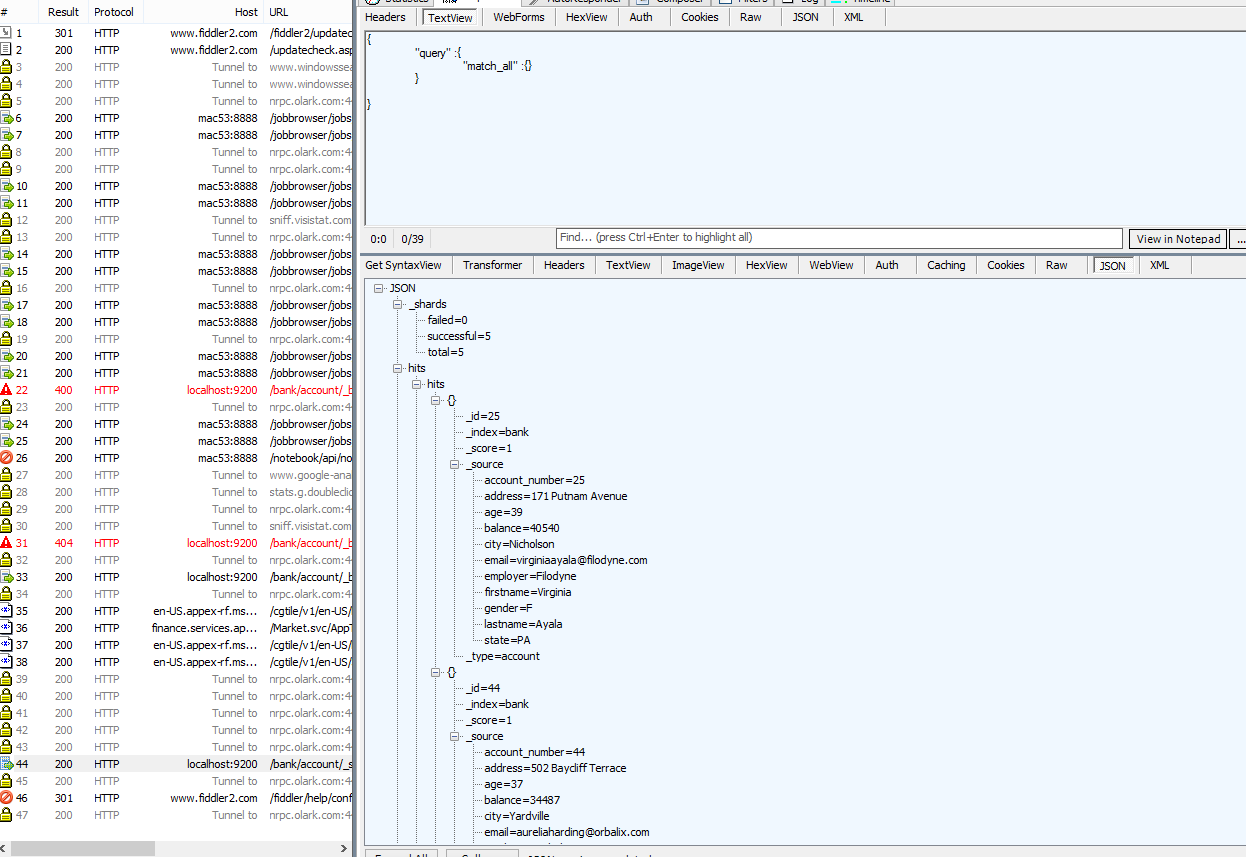
To see all values fire post request <http://localhost:9200/bank/account/_search>

With content

|  |
| --- |
| {  "query" :{  "match\_all" :{}  }    } |



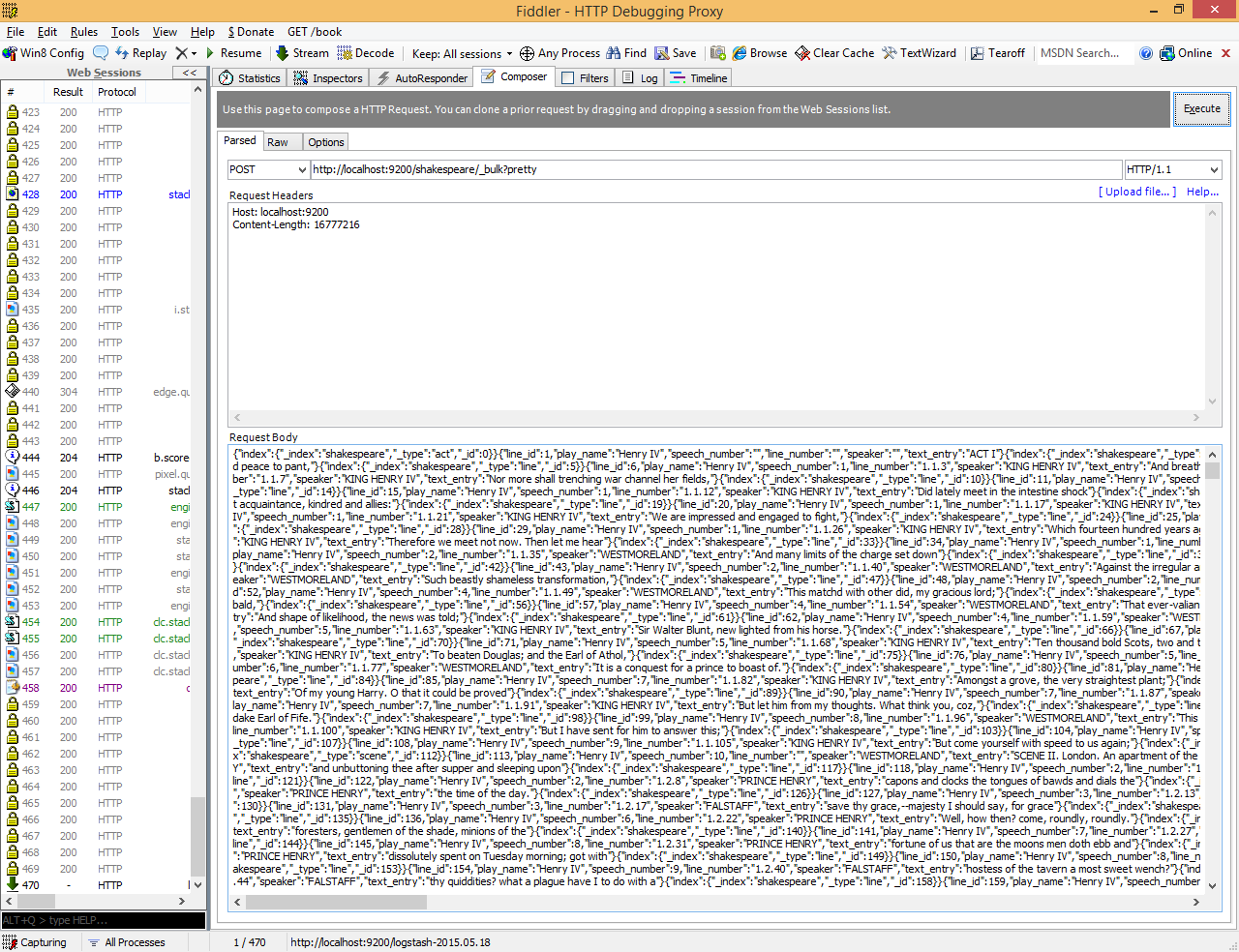
Check output

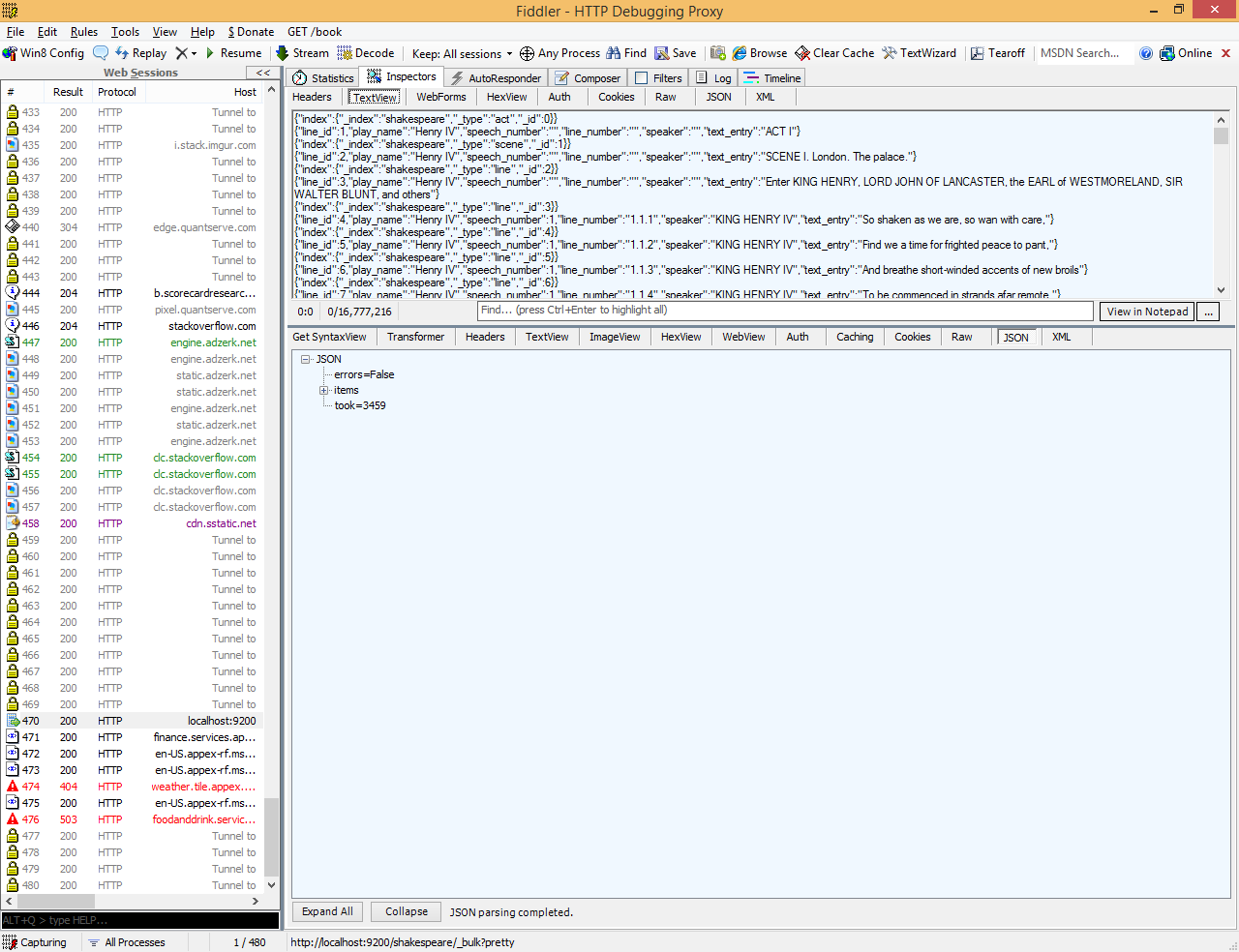


It will show all rows

## Load shakespeer.json file

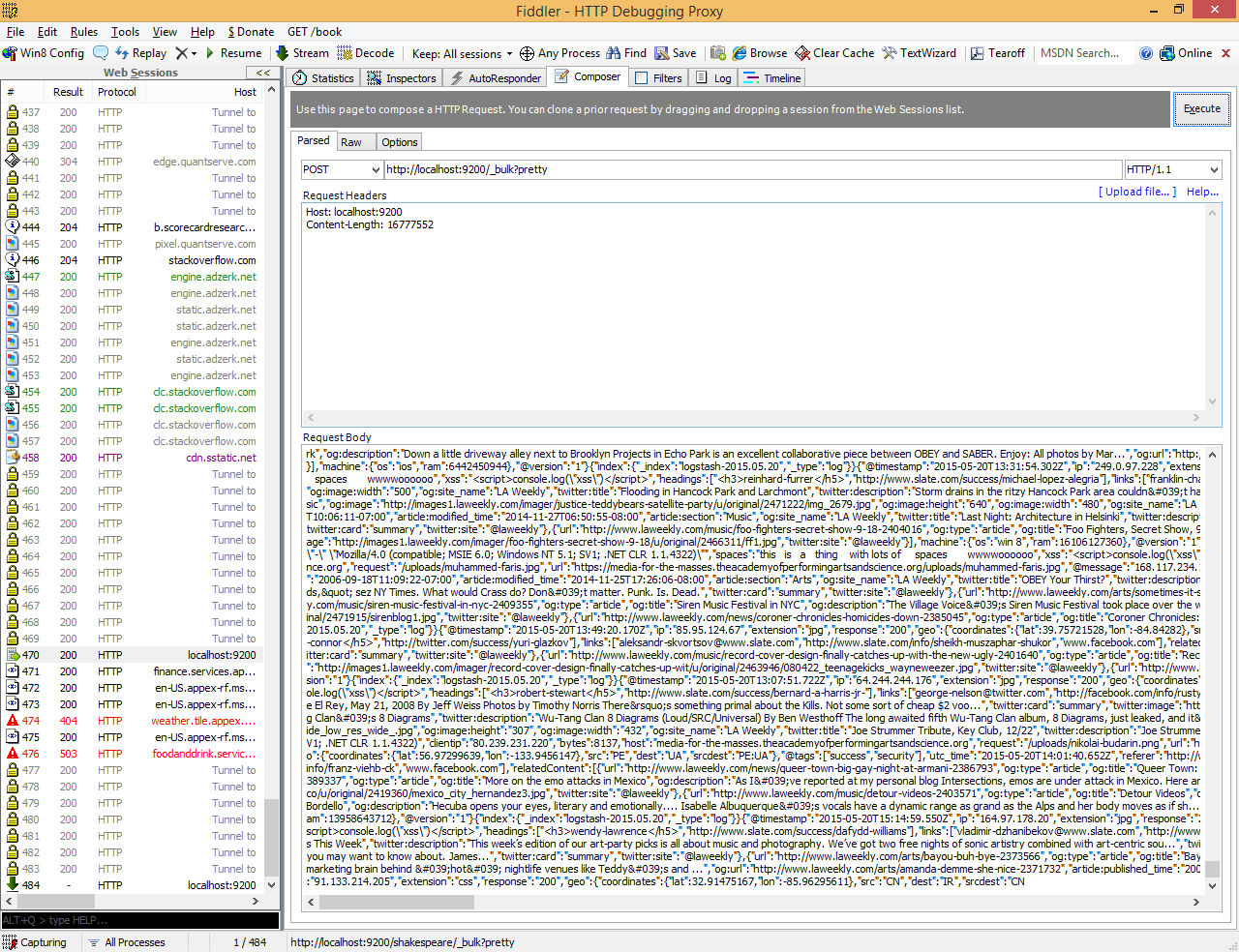
curl -XPOST 'localhost:9200/shakespeare/\_bulk?pretty' --data-binary @shakespeare.json

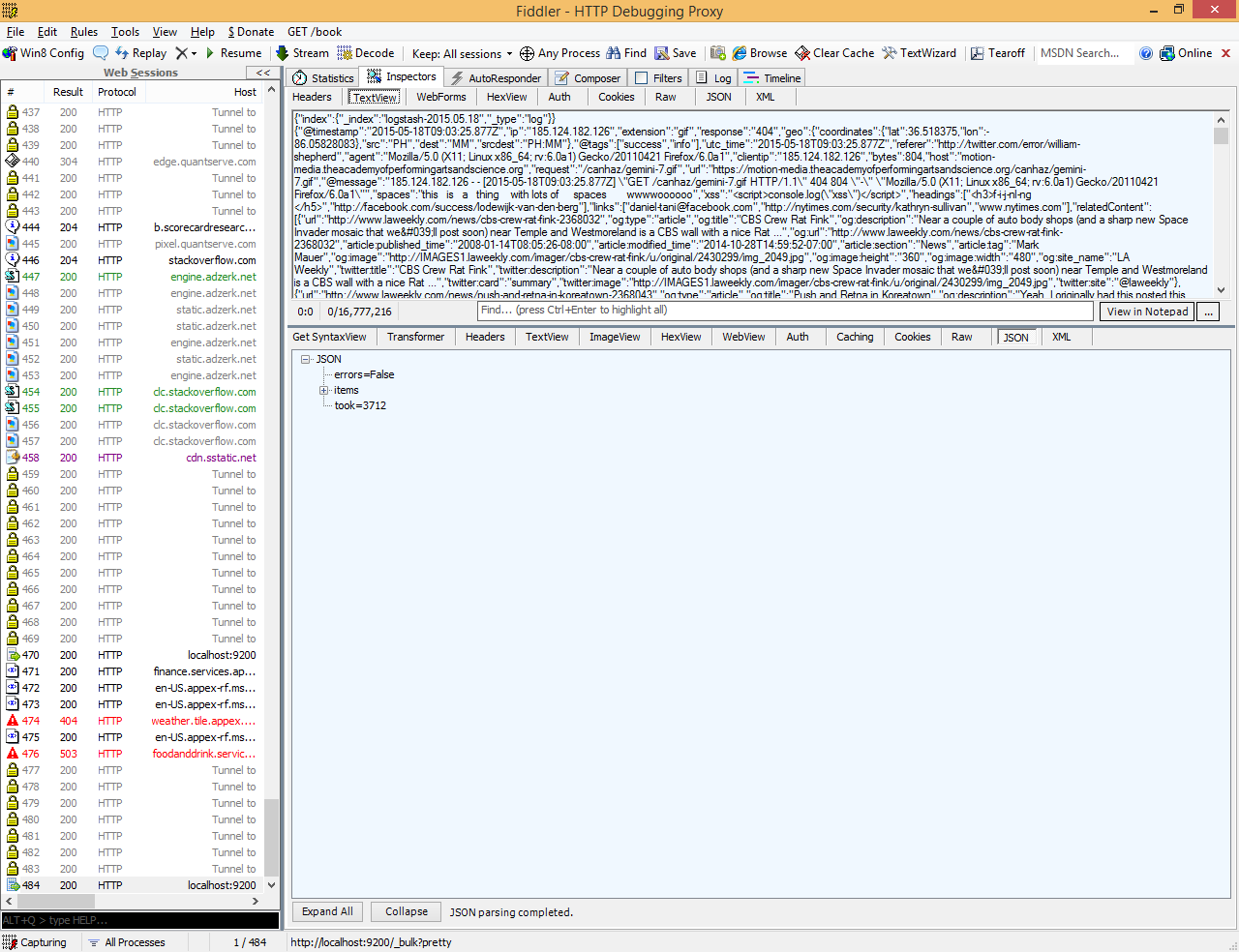




## Load logs.json file

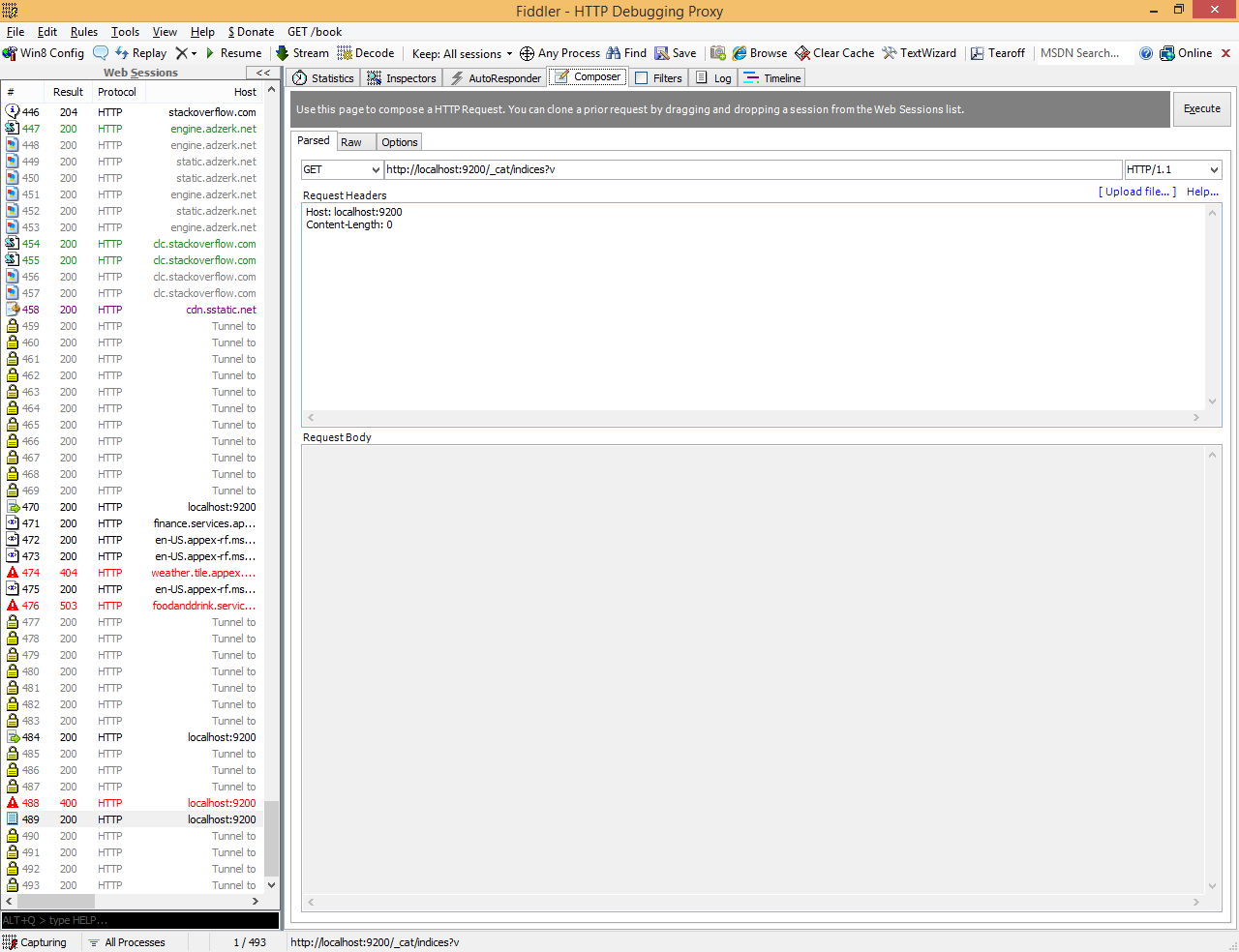
curl -XPOST 'localhost:9200/\_bulk?pretty' --data-binary @logs.jsonl

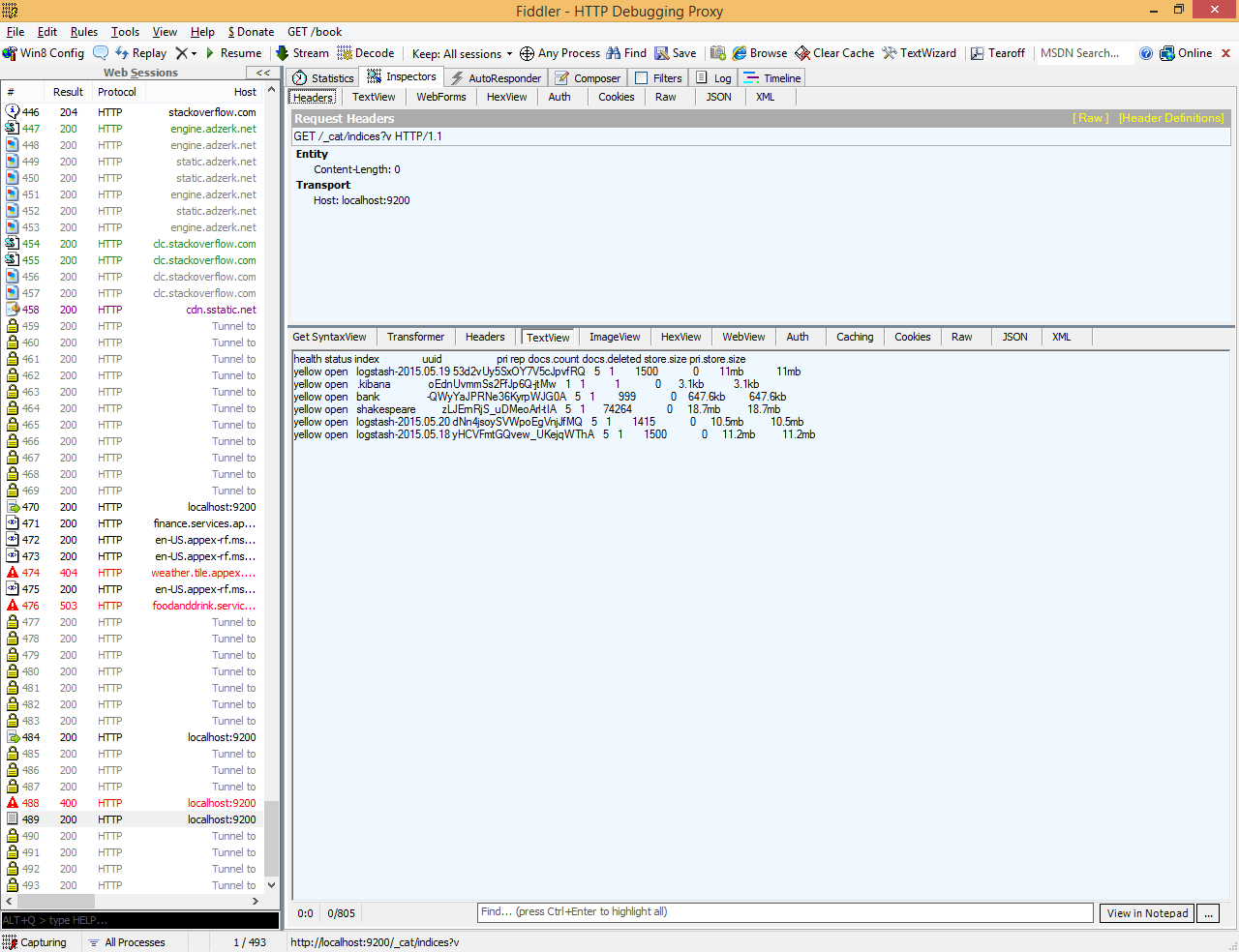




curl 'localhost:9200/\_cat/indices?v'

in fiddler execute get request





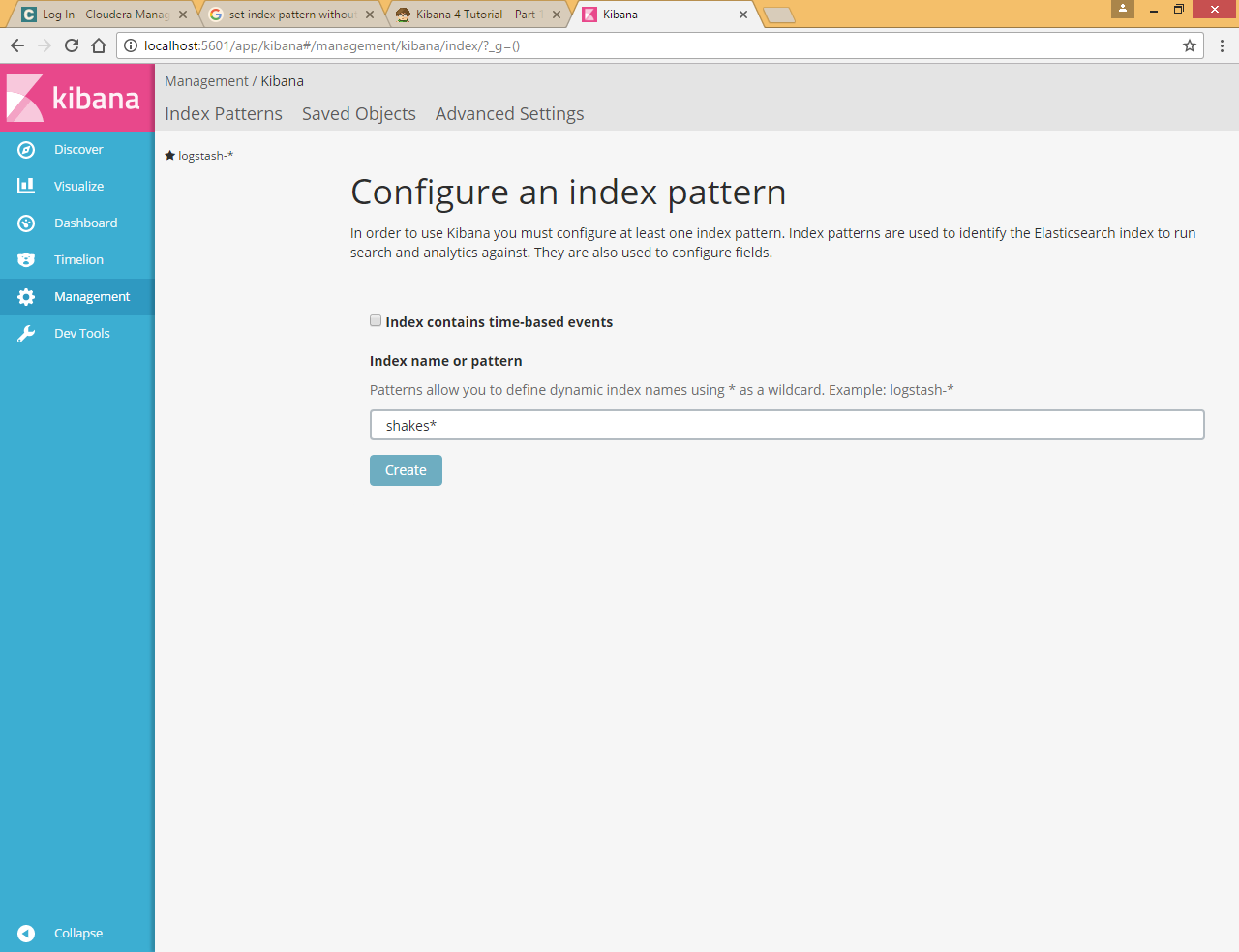
|  |
| --- |
| health status index uuid pri rep docs.count docs.deleted store.size pri.store.size  yellow open logstash-2015.05.19 53d2vUy5SxOY7V5cJpvfRQ 5 1 1500 0 11mb 11mb  yellow open .kibana oEdnUvmmSs2FfJp6Q-jtMw 1 1 1 0 3.1kb 3.1kb  yellow open bank -QWyYaJPRNe36KyrpWJG0A 5 1 999 0 647.6kb 647.6kb  yellow open shakespeare zLJErnRjS\_uDMeoArl-tIA 5 1 74264 0 18.7mb 18.7mb  yellow open logstash-2015.05.20 dNn4jsoySVWpoEgVnjJfMQ 5 1 1415 0 10.5mb 10.5mb  yellow open logstash-2015.05.18 yHCVFmtGQvew\_UKejqWThA 5 1 1500 0 11.2mb 11.2mb |

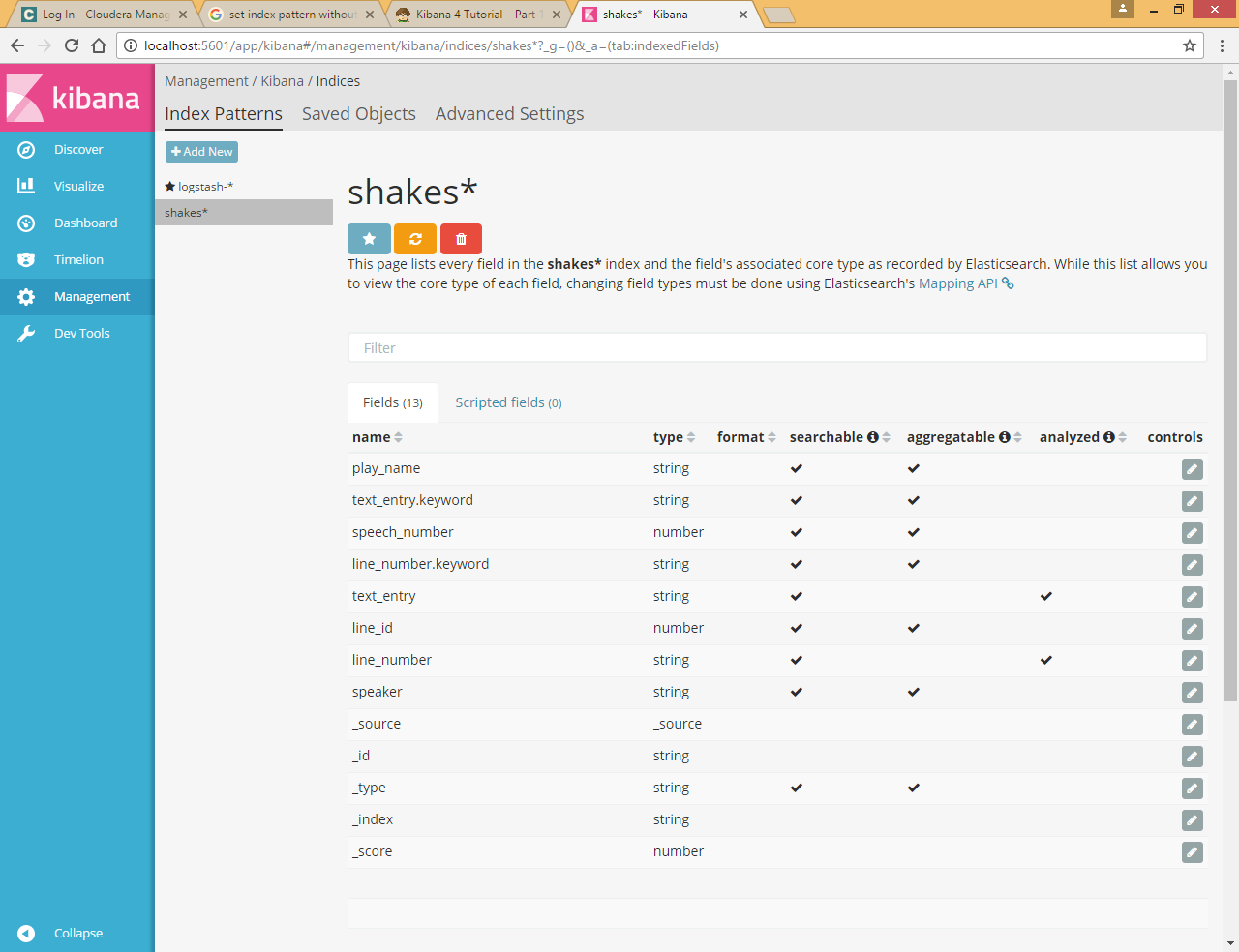
# Define Index on kibana

each set of data loaded to Elasticsearch has an index pattern. In the previous section, the Shakespeare data set has an index named shakespeare, and the accounts data set has an index named bank. An index pattern is a string with optional wildcards that can match multiple indices. For example, in the common logging use case, a typical index name contains the date in MM-DD-YYYY format, and an index pattern for May would look something like logstash-2015.05\*.

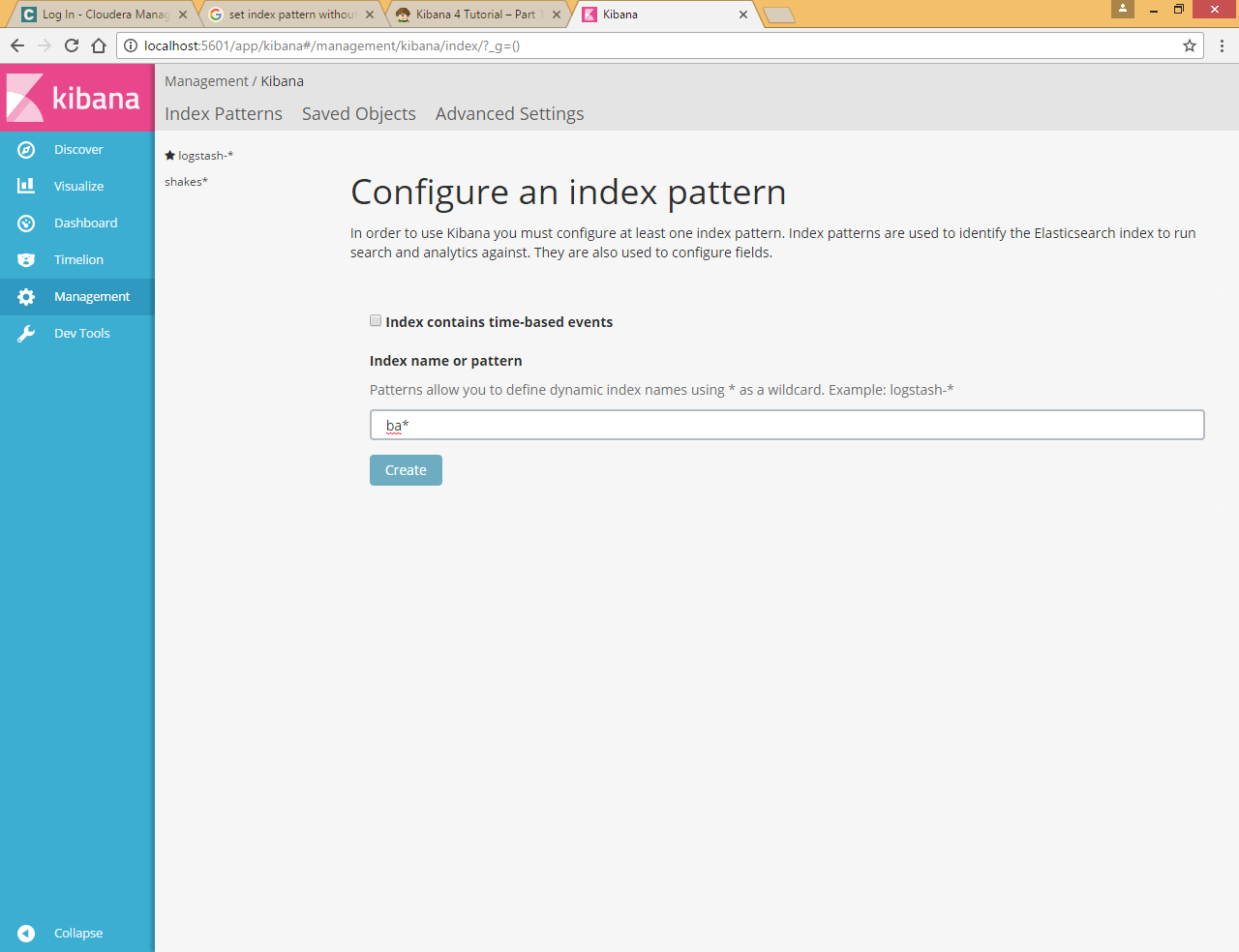
For this tutorial, any pattern that matches the name of an index we’ve loaded will work. Open a browser and navigate to localhost:5601. Click the **Management** application, then the **Index Patterns** link. Click **Add New** to define a new index pattern. Two of the sample data sets, the Shakespeare plays and the financial accounts, don’t contain time-series data. Make sure the **Index contains time-based events** box is unchecked when you create index patterns for these data sets. Specify shakes\* as the index pattern for the Shakespeare data set and click **Create** to define the index pattern, then define a second index pattern named ba\*.

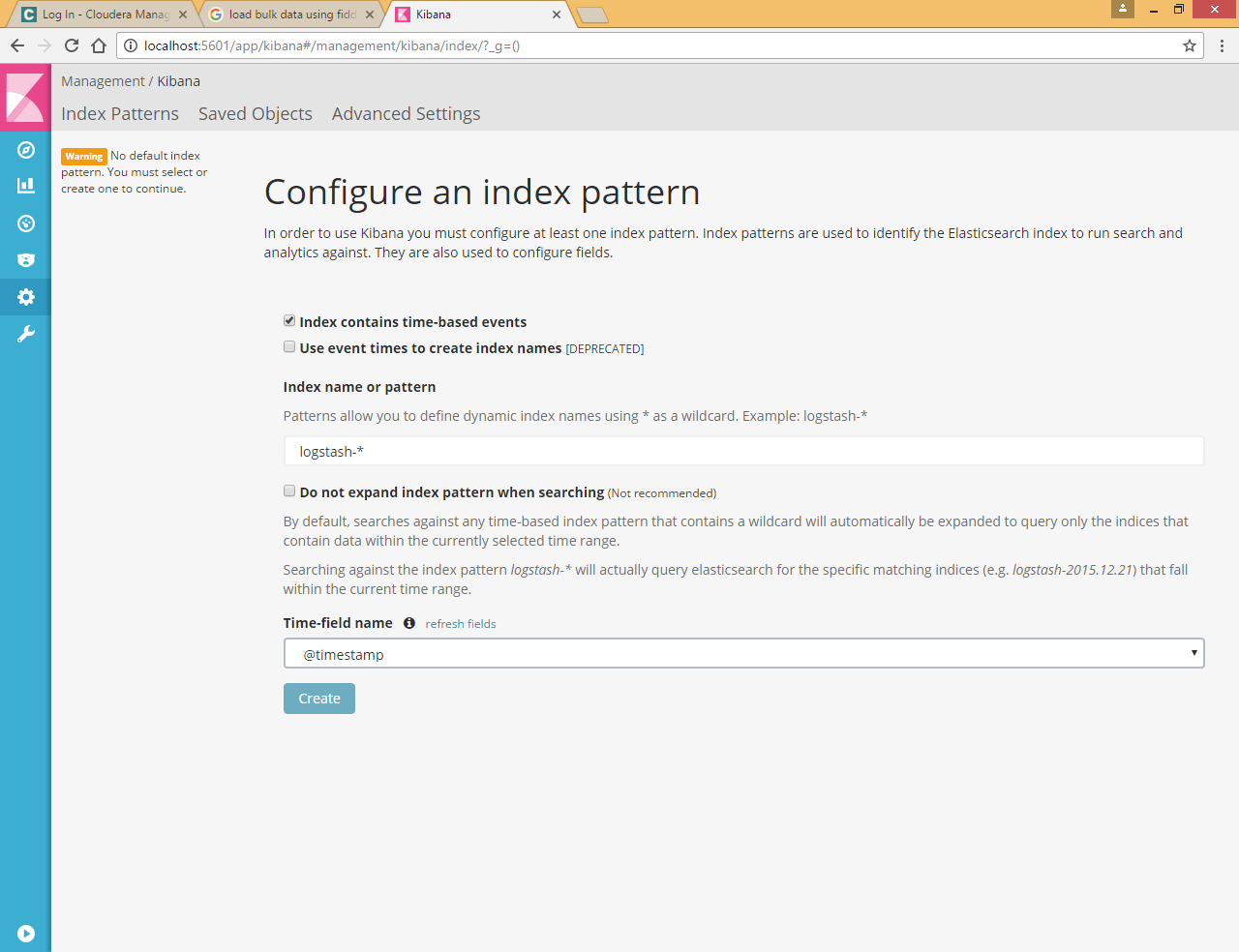
The Logstash data set does contain time-series data, so after clicking **Add New** to define the index for this data set, make sure the **Index contains time-based events** box is checked and select the @timestamp field from the **Time-field name** drop-down.





Click Add new to add new index pattern

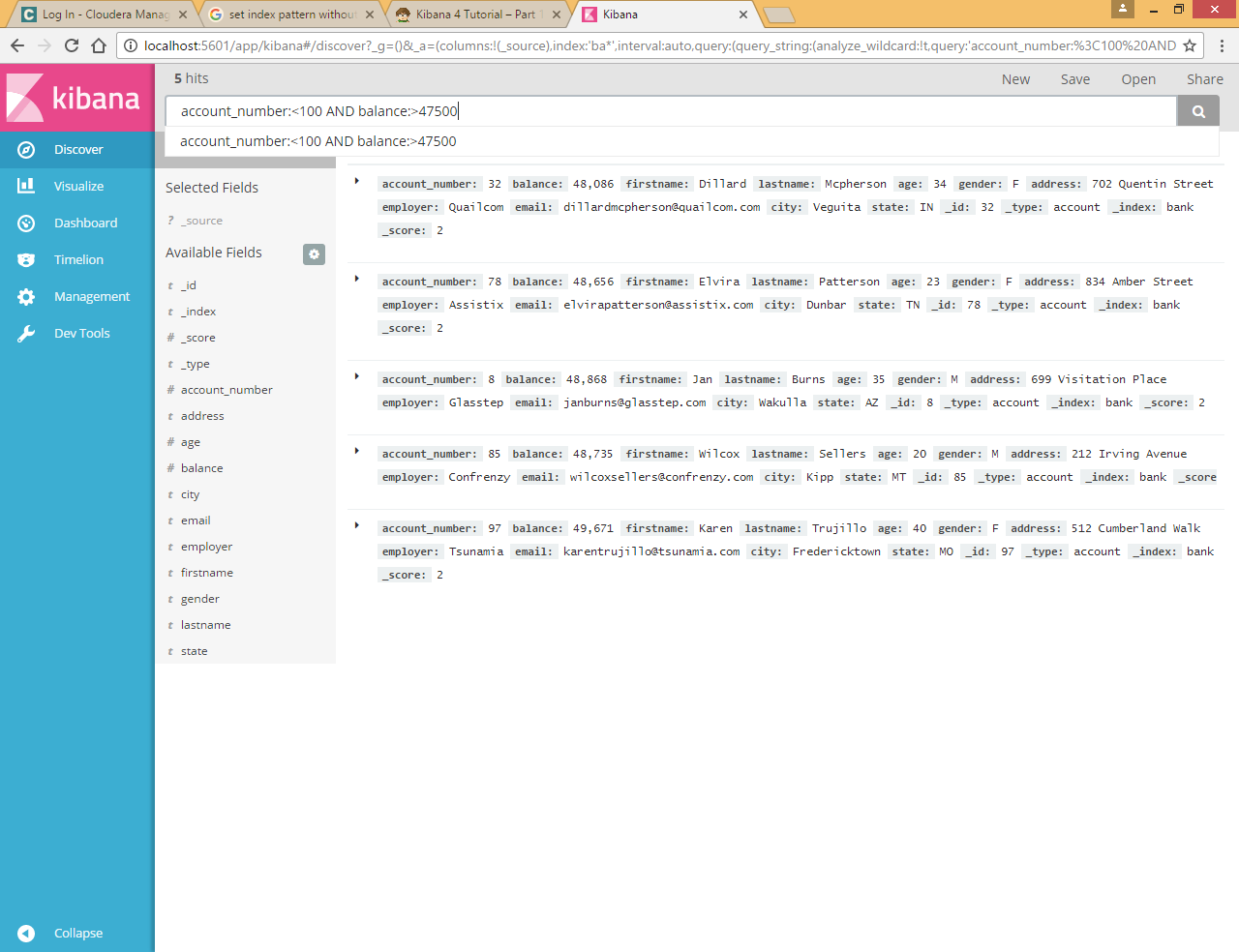




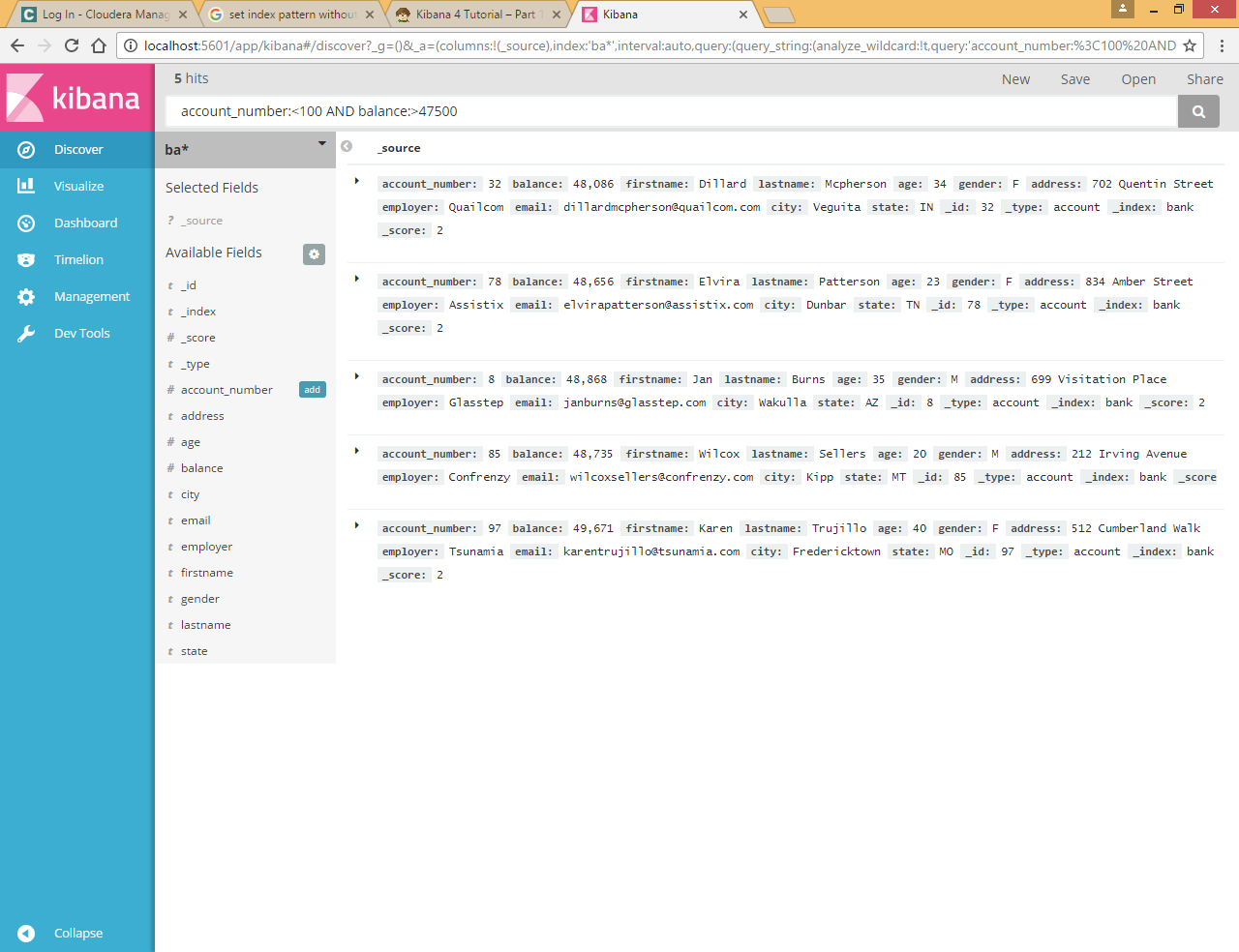
# Discovering Data

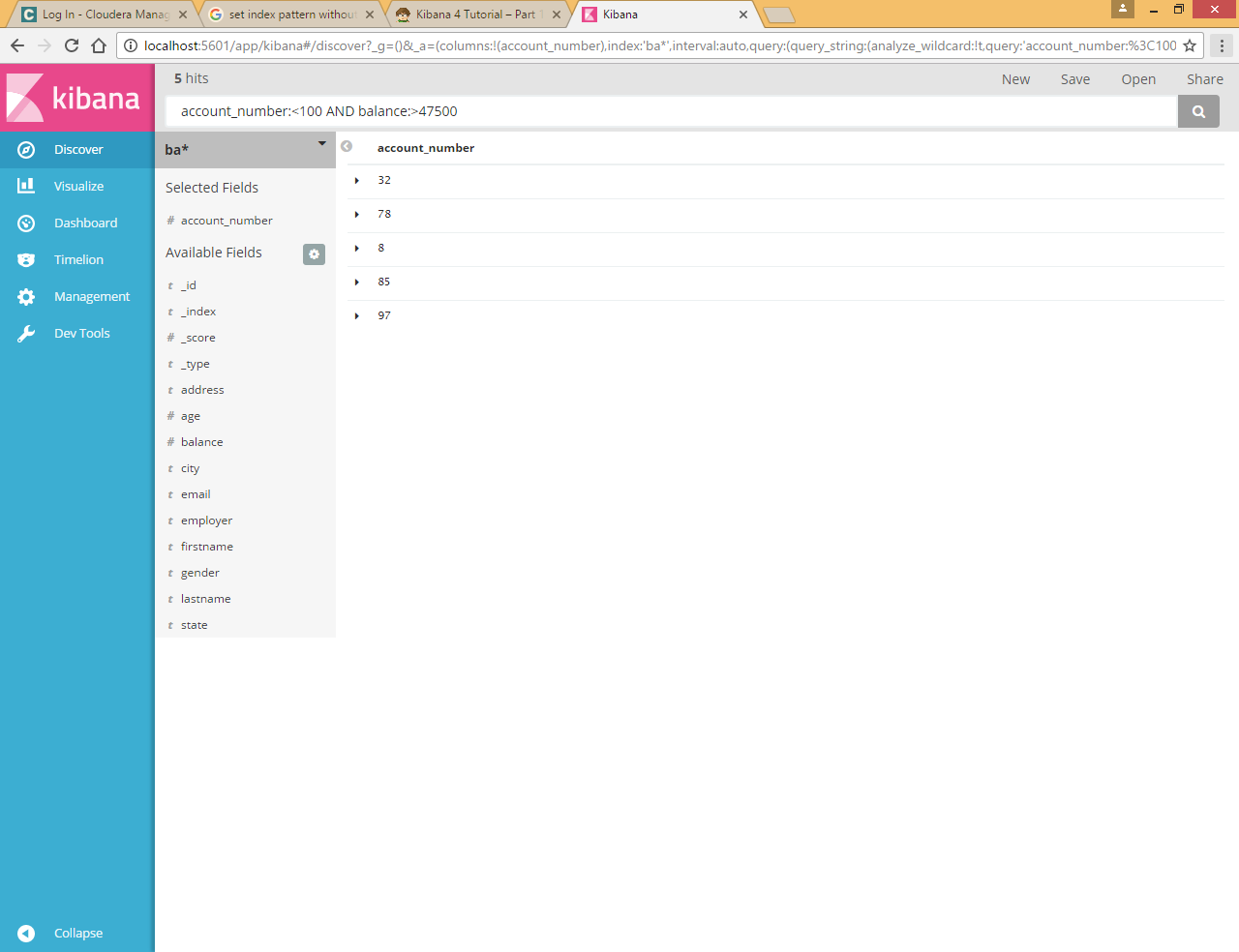
To try it out, select the ba\* index pattern and enter the following query string in the query bar:

account\_number:<100 AND balance:>47500

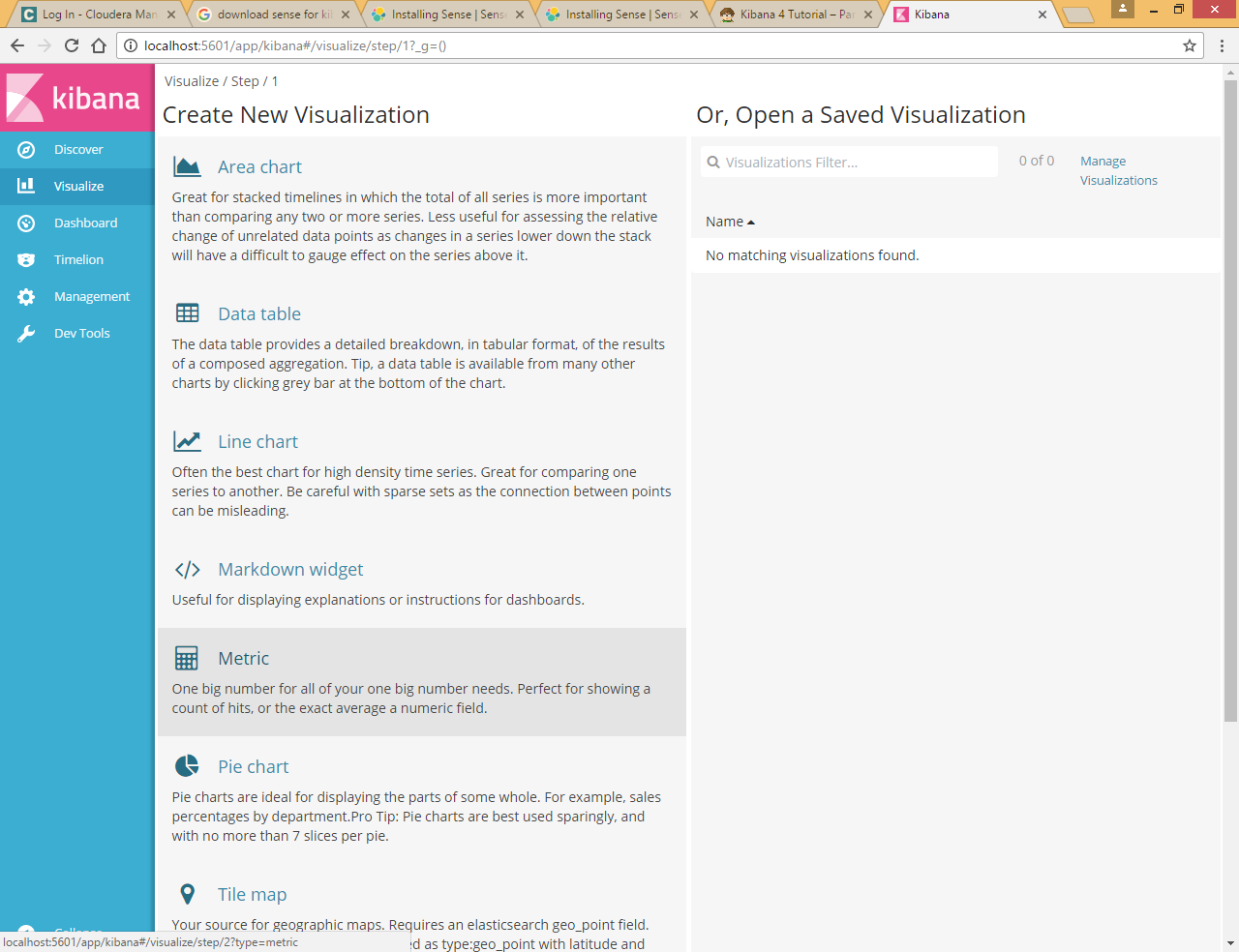


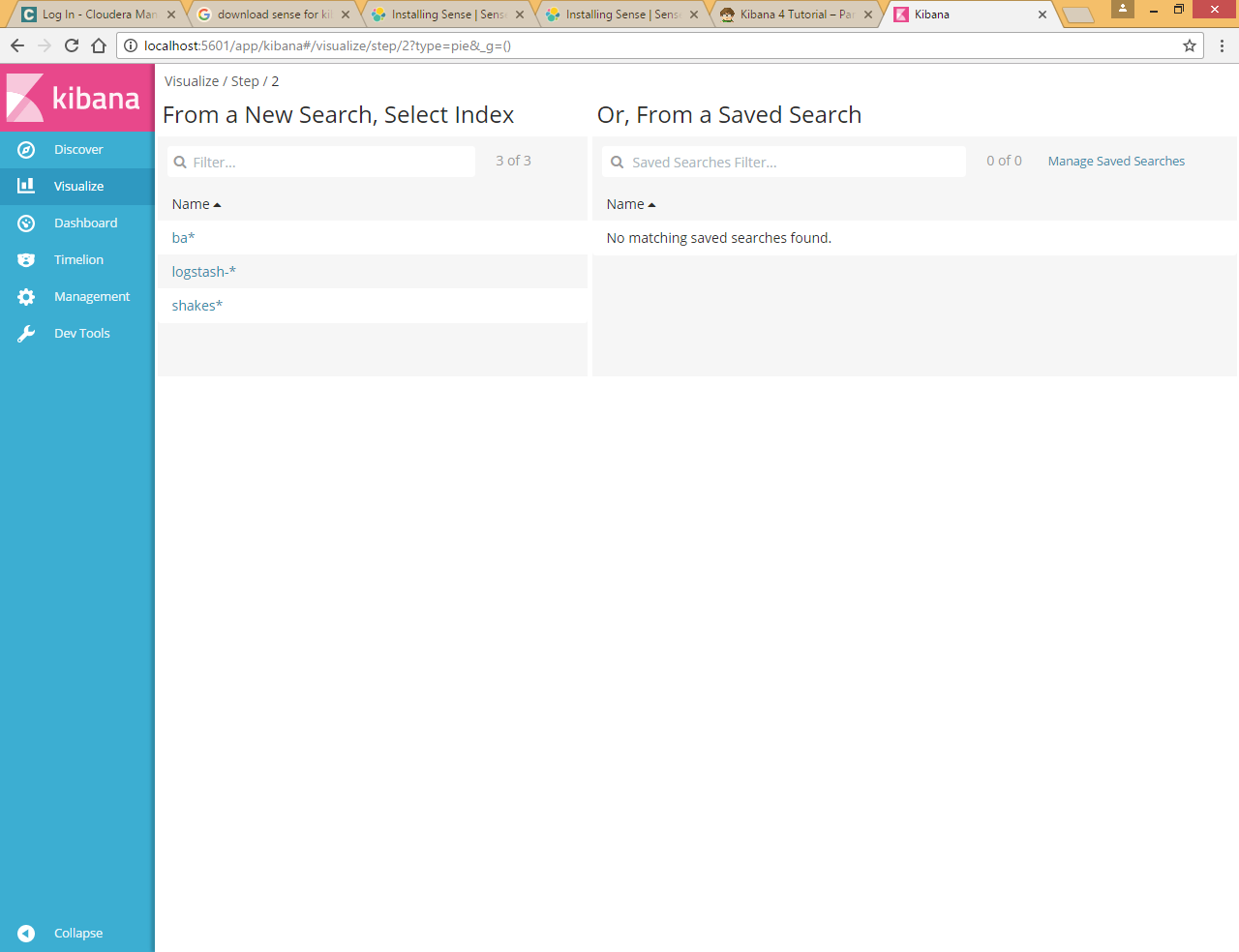
By default, all fields are shown for each matching document. To choose which document fields to display, hover over the Available Fields list and click the **add** button next to each field you want include. For example, if you add just the account\_number, the display changes to a simple list of five account numbers:





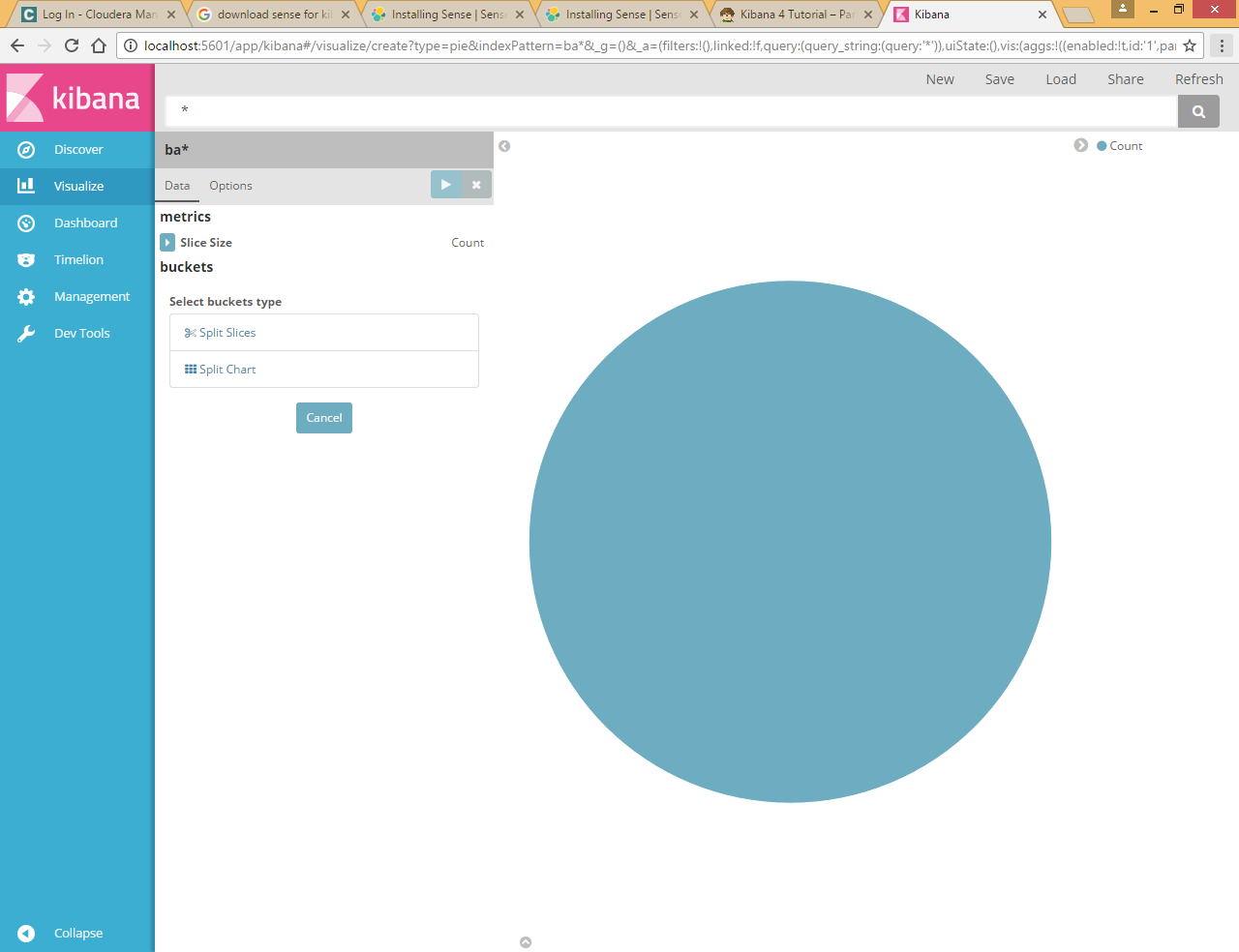
# Visualize





To get started, click **Pie chart** in the list of visualizations. You can build visualizations from saved searches, or enter new search criteria. To enter new search criteria, you first need to select an index pattern to specify what indices to search. We want to search the account data, so select the ba\*index pattern.

The default search matches all documents. Initially, a single "slice" encompasses the entire pie:



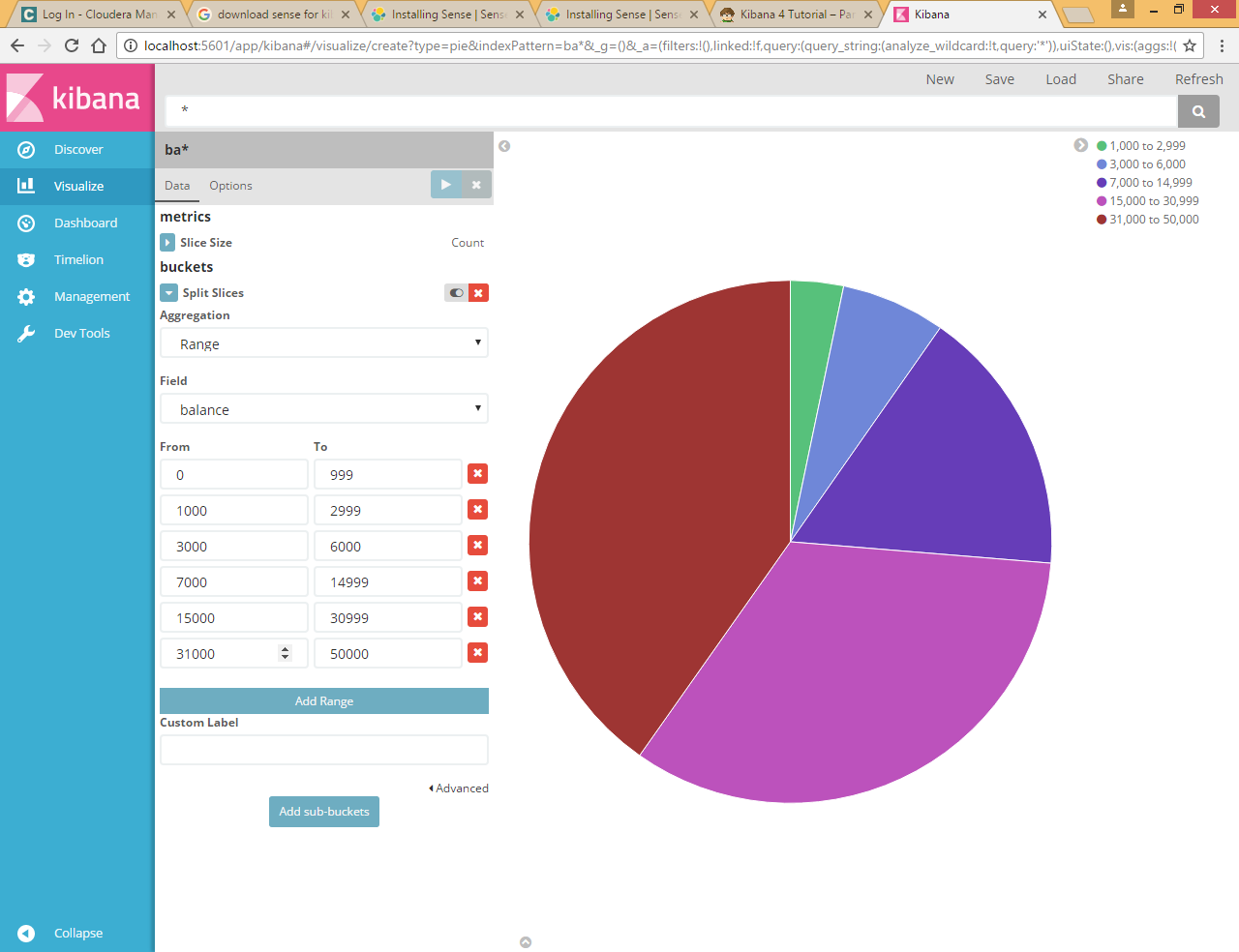
To specify what slices to display in the chart, you use an Elasticsearch [bucket aggregation](https://www.elastic.co/guide/en/elasticsearch/reference/5.0/search-aggregations.html). A bucket aggregation simply sorts the documents that match your search criteria into different categories, aka buckets. For example, the account data includes the balance of each account. Using a bucket aggregation, you can establish multiple ranges of account balances and find out how many accounts fall into each range.

To define a bucket for each range:

1. Click the **Split Slices** buckets type.
2. Select **Range** from the **Aggregation** list.
3. Select the **balance** field from the **Field** list.
4. Click **Add Range** four times to bring the total number of ranges to six.
5. Define the following ranges:
6. 0 999
7. 1000 2999
8. 3000 6999
9. 7000 14999
10. 15000 30999

31000 50000

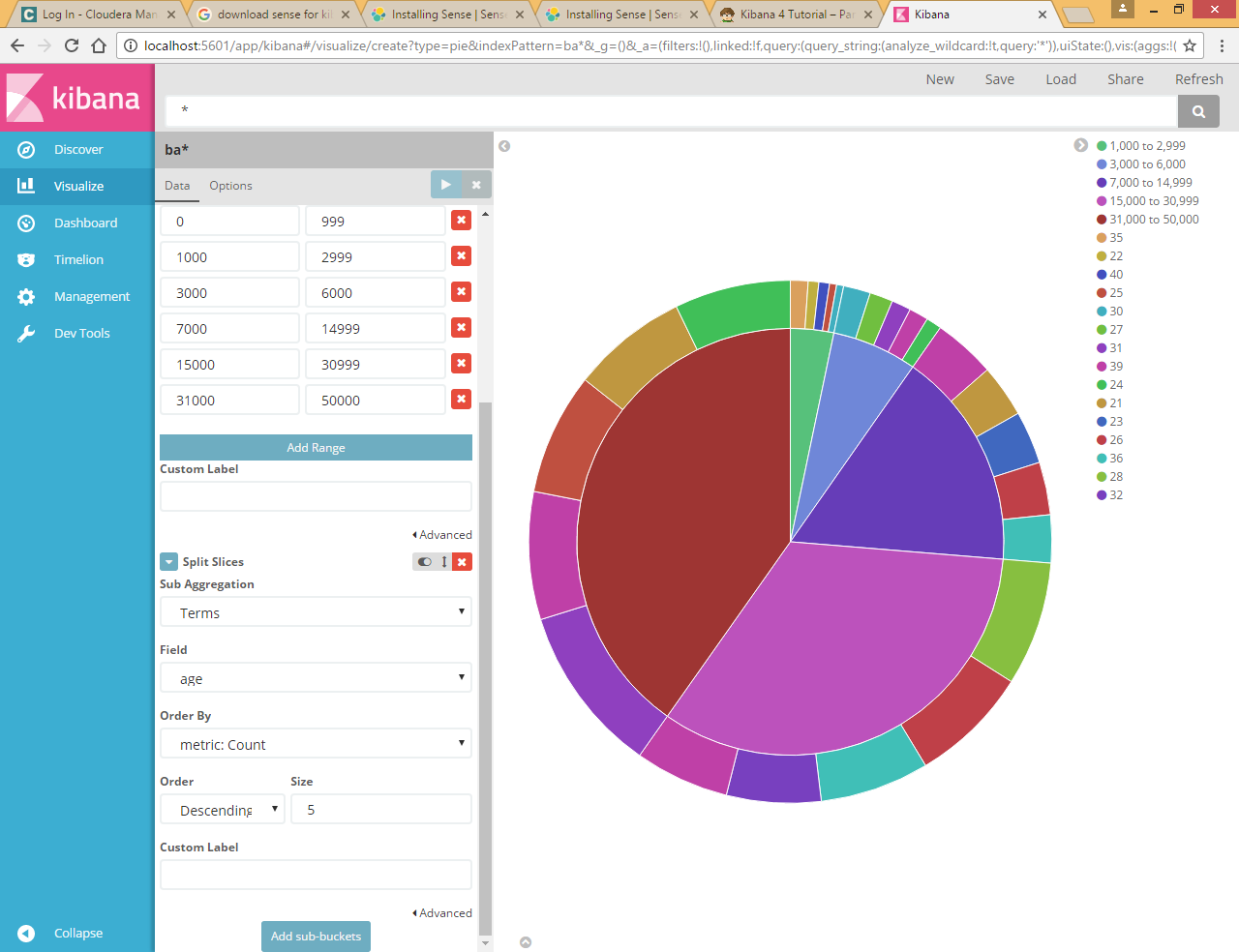
1. Click **Apply changes** images/apply-changes-button.png to update the chart.



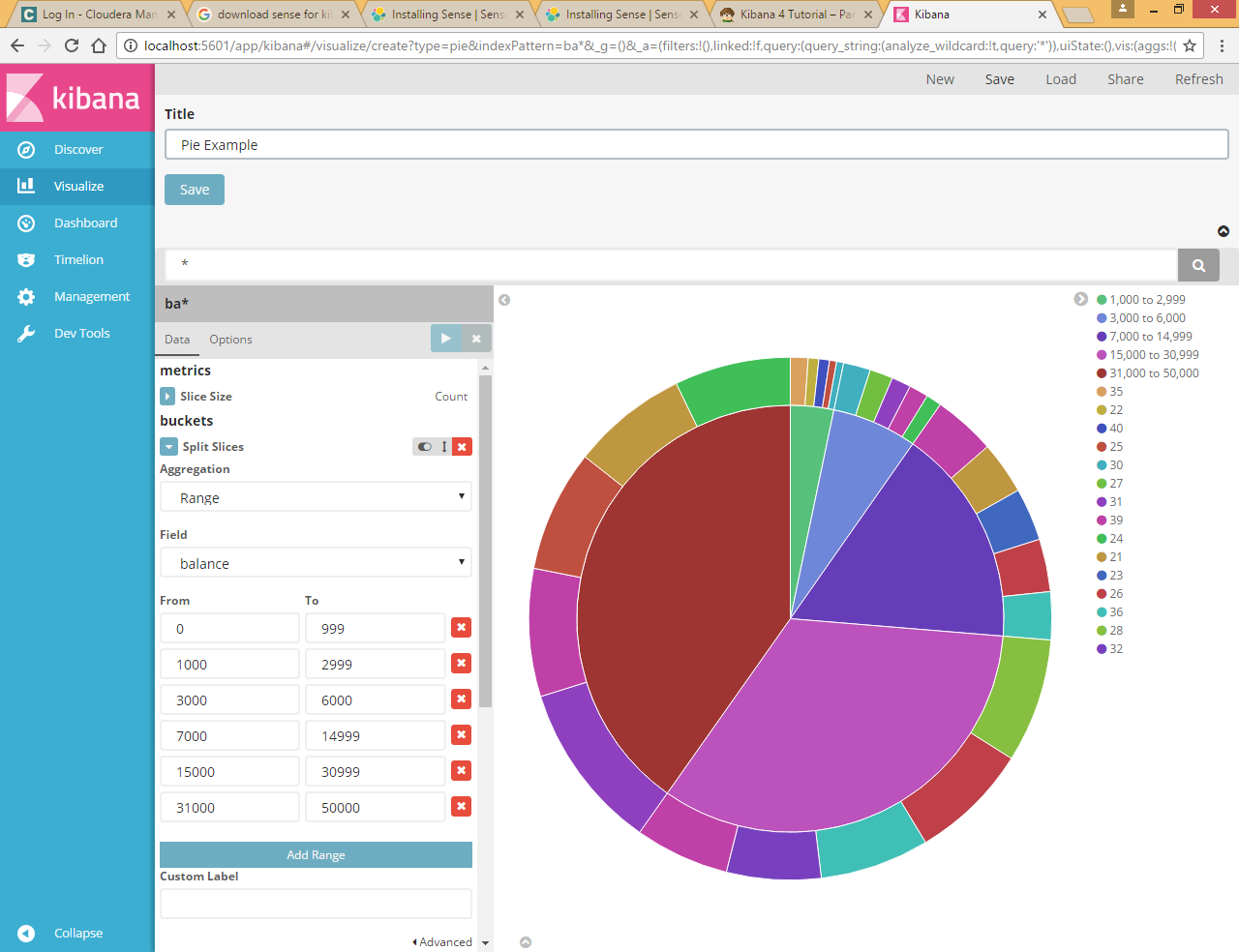
Let’s take a look at another dimension of the data: the account holder’s age. By adding another bucket aggregation, you can see the ages of the account holders in each balance range:

1. Click **Add sub-buckets** below the buckets list.
2. Click **Split Slices** in the buckets type list.
3. Select **Terms** from the aggregation list.
4. Select **age** from the field list.
5. Click **Apply changes** images/apply-changes-button.png.

Now you can see the break down of the account holders' ages displayed in a ring around the balance ranges.

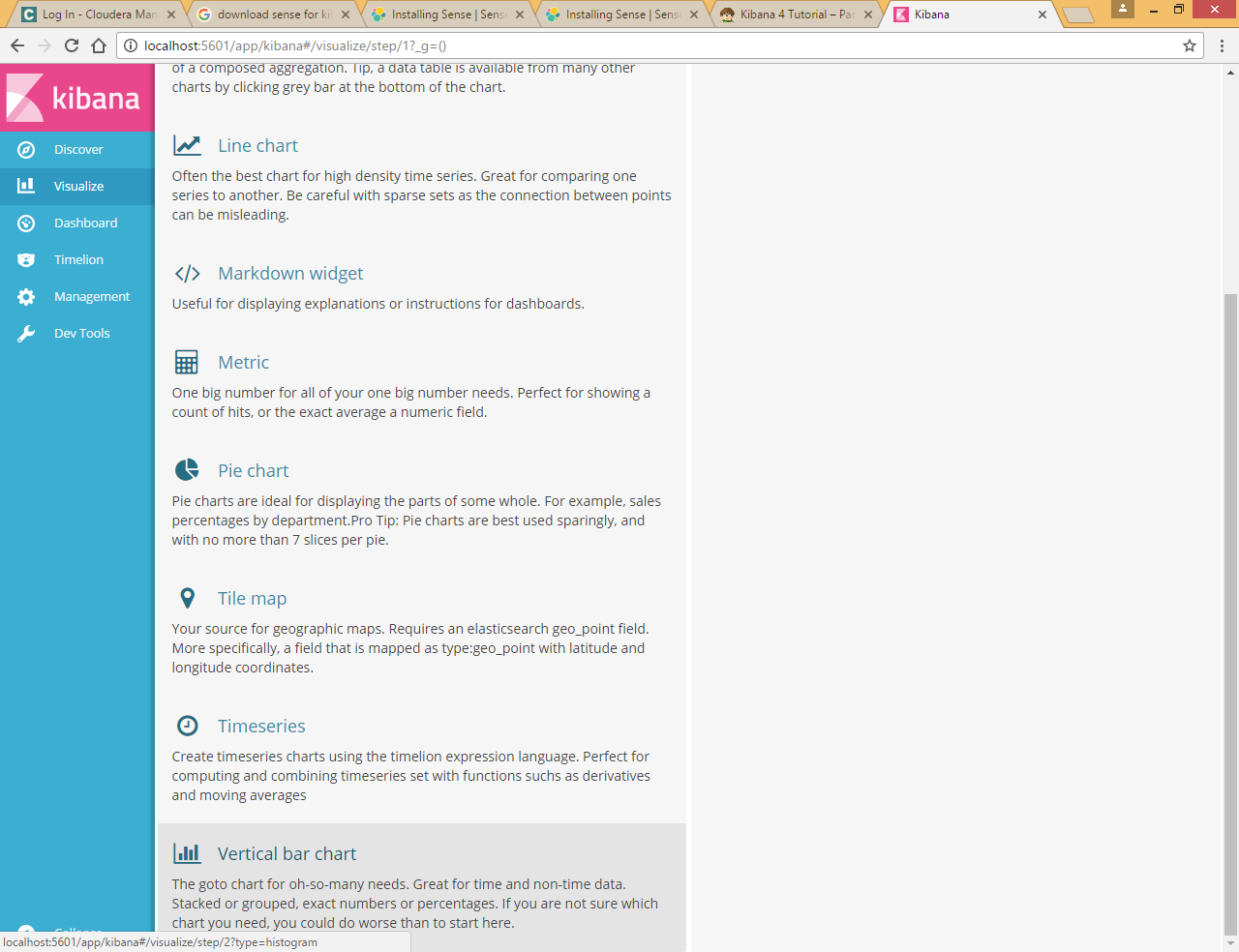


To save this chart so we can use it later, click **Save** and enter the name Pie Example.

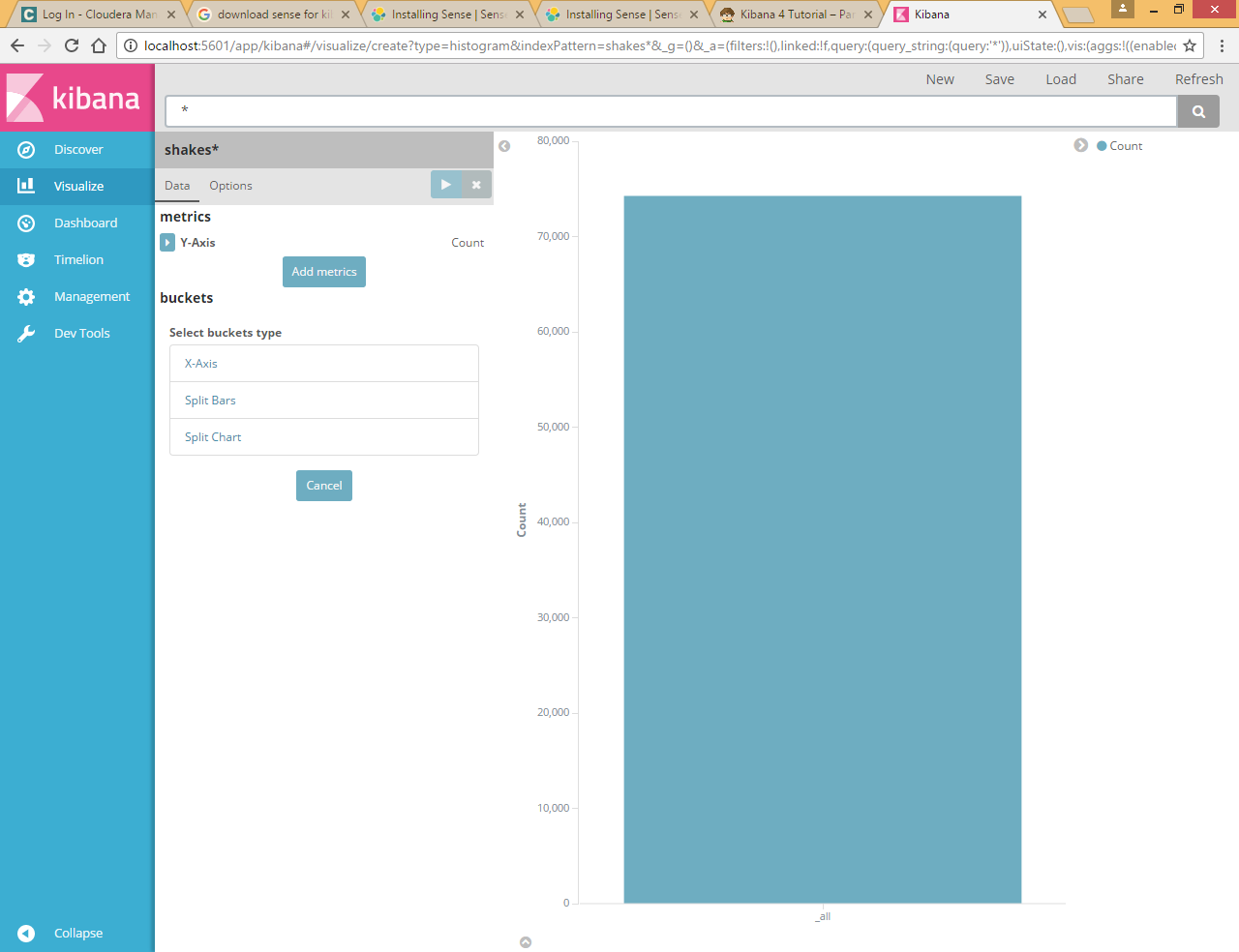


Next, we’re going to look at data in the Shakespeare data set. Let’s find out how the plays compare when it comes to the number of speaking parts and display the information in a bar chart:

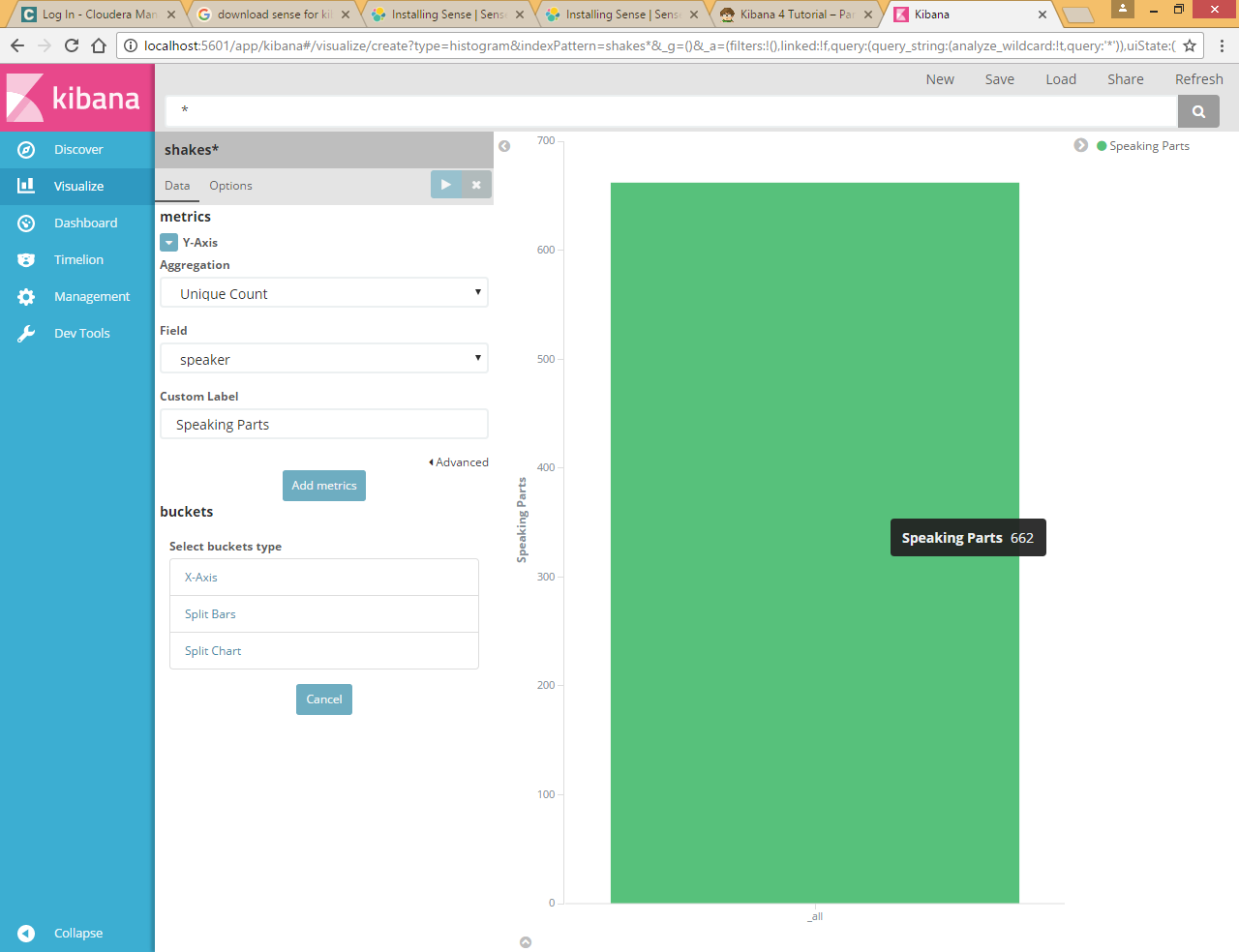
1. Click **New** and select **Vertical bar chart**.



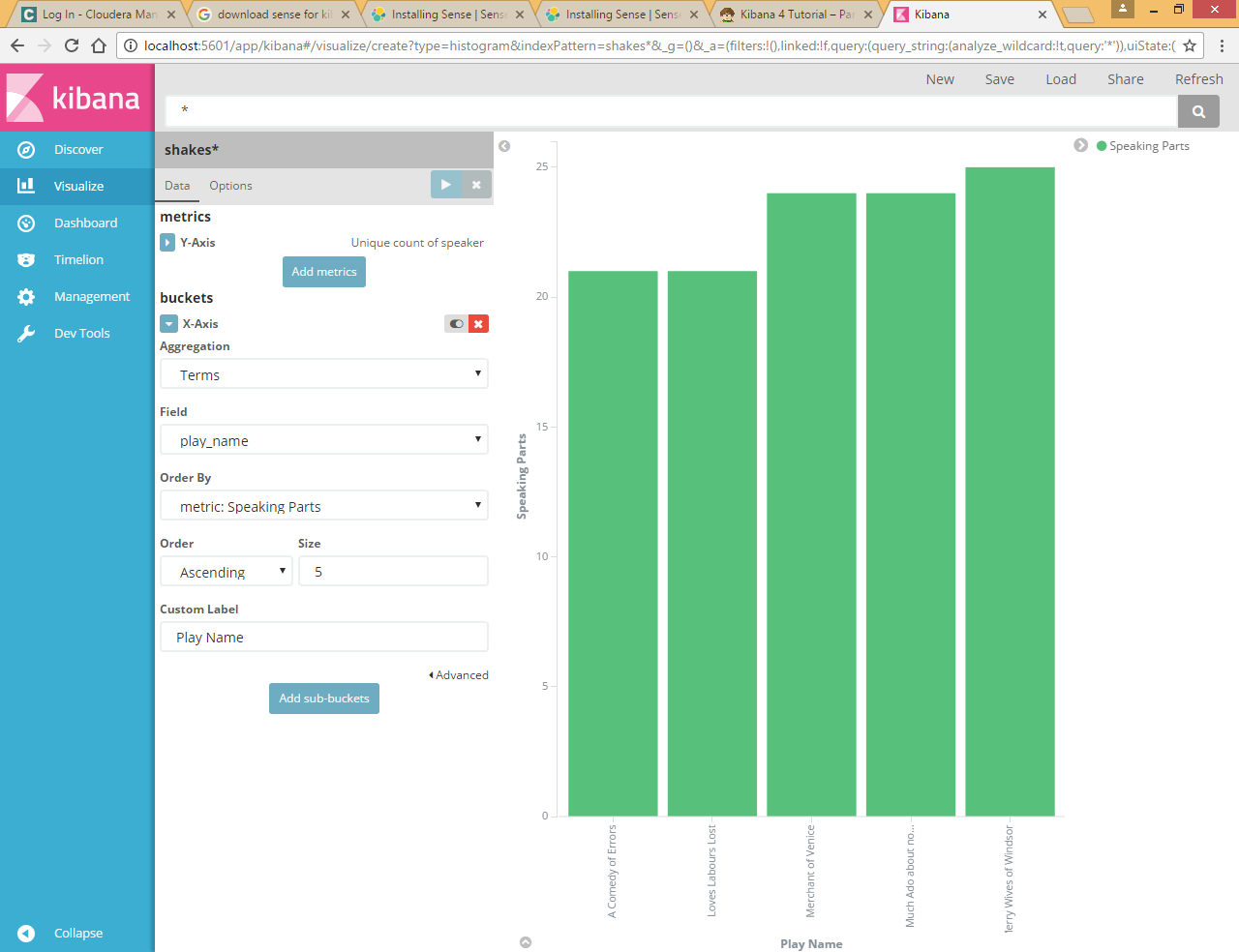
1. Select the shakes\* index pattern. Since you haven’t defined any buckets yet, you’ll see a single big bar that shows the total count of documents that match the default wildcard query.



1. To show the number of speaking parts per play along the y-axis, you need to configure the Y-axis [metric aggregation](https://www.elastic.co/guide/en/elasticsearch/reference/5.0/search-aggregations.html). A metric aggregation computes metrics based on values extracted from the search results. To get the number of speaking parts per play, select the **Unique Count** aggregation and choose **speaker** from the field list. You can also give the axis a custom label, Speaking Parts.



1. To show the different plays long the x-axis, select the X-Axis buckets type, select **Terms** from the aggregation list, and choose **play\_name** from the field list. To list them alphabetically, select **Ascending** order. You can also give the axis a custom label, Play Name.
2. Click **Apply changes** images/apply-changes-button.png to view the results.

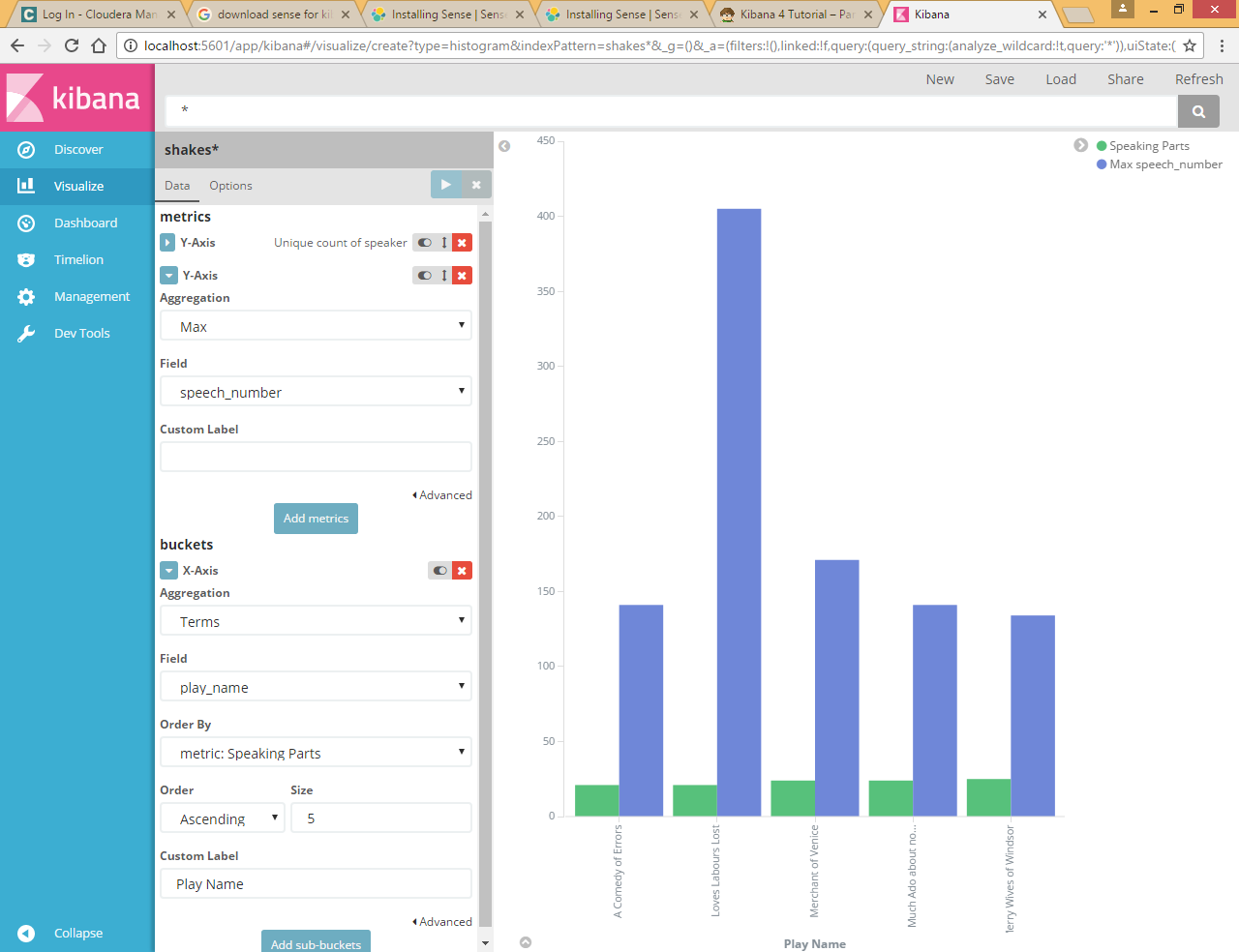


Notice how the individual play names show up as whole phrases, instead of being broken down into individual words. This is the result of the mapping we did at the beginning of the tutorial, when we marked the **play\_name** field as not analyzed.

Hovering over each bar shows you the number of speaking parts for each play as a tooltip. To turn tooltips off and configure other options for your visualizations, select the Visualization builder’s **Options** tab.

Now that you have a list of the smallest casts for Shakespeare plays, you might also be curious to see which of these plays makes the greatest demands on an individual actor by showing the maximum number of speeches for a given part.

1. Click **Add metrics** to add a Y-axis aggregation.
2. Choose the **Max** aggregation and select the **speech\_number** field.
3. Click **Options** tab and change the **Bar Mode** to **grouped**.
4. Click **Apply changes** images/apply-changes-button.png. Your chart should now look like this:



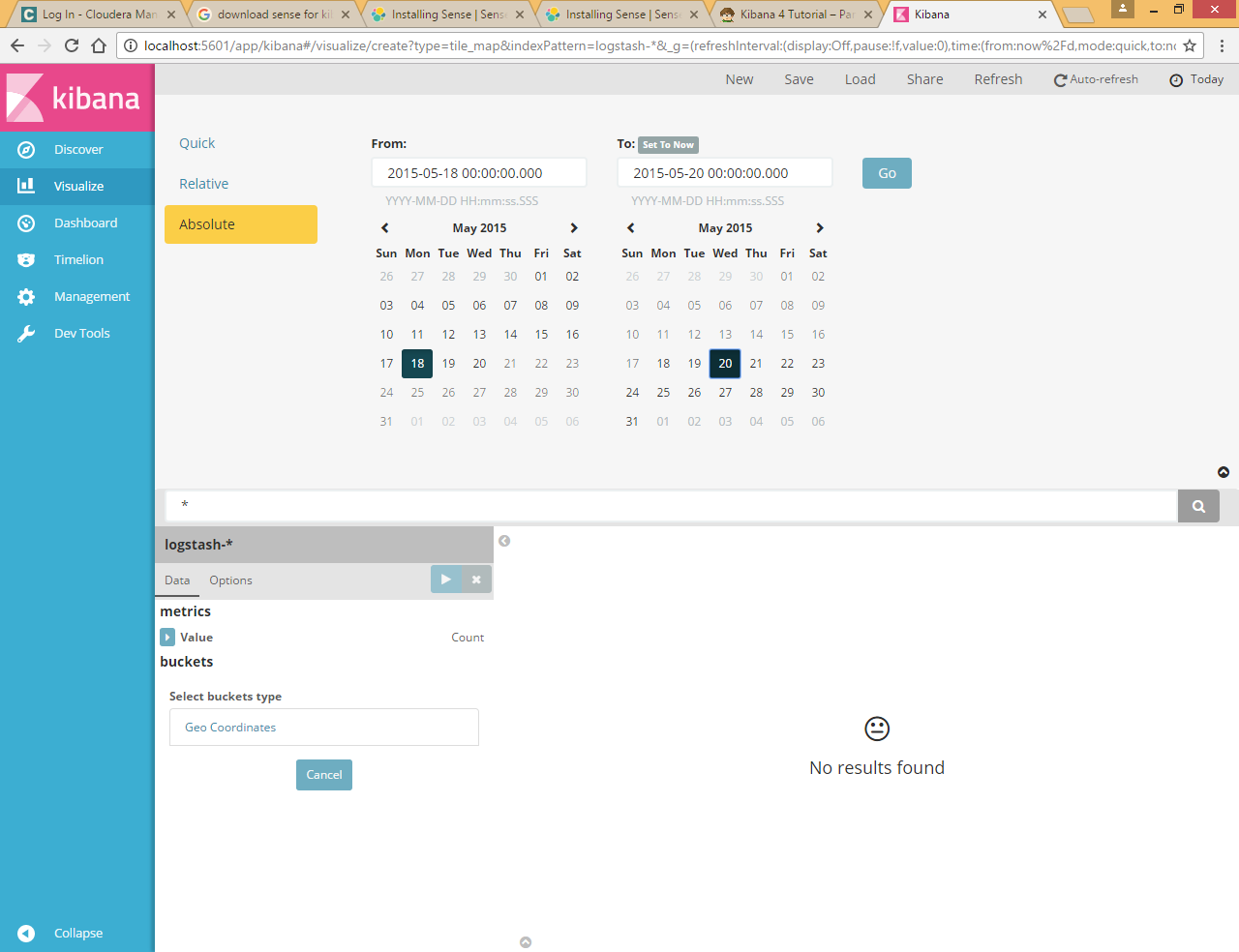
As you can see, Love’s Labours Lost has an unusually high maximum speech number, compared to the other plays, and might therefore make more demands on an actor’s memory.

Note how the **Number of speaking parts** Y-axis starts at zero, but the bars don’t begin to differentiate until 18. To make the differences stand out, starting the Y-axis at a value closer to the minimum, go to Options and select **Scale Y-Axis to data bounds**.

Save this chart with the name Bar Example.

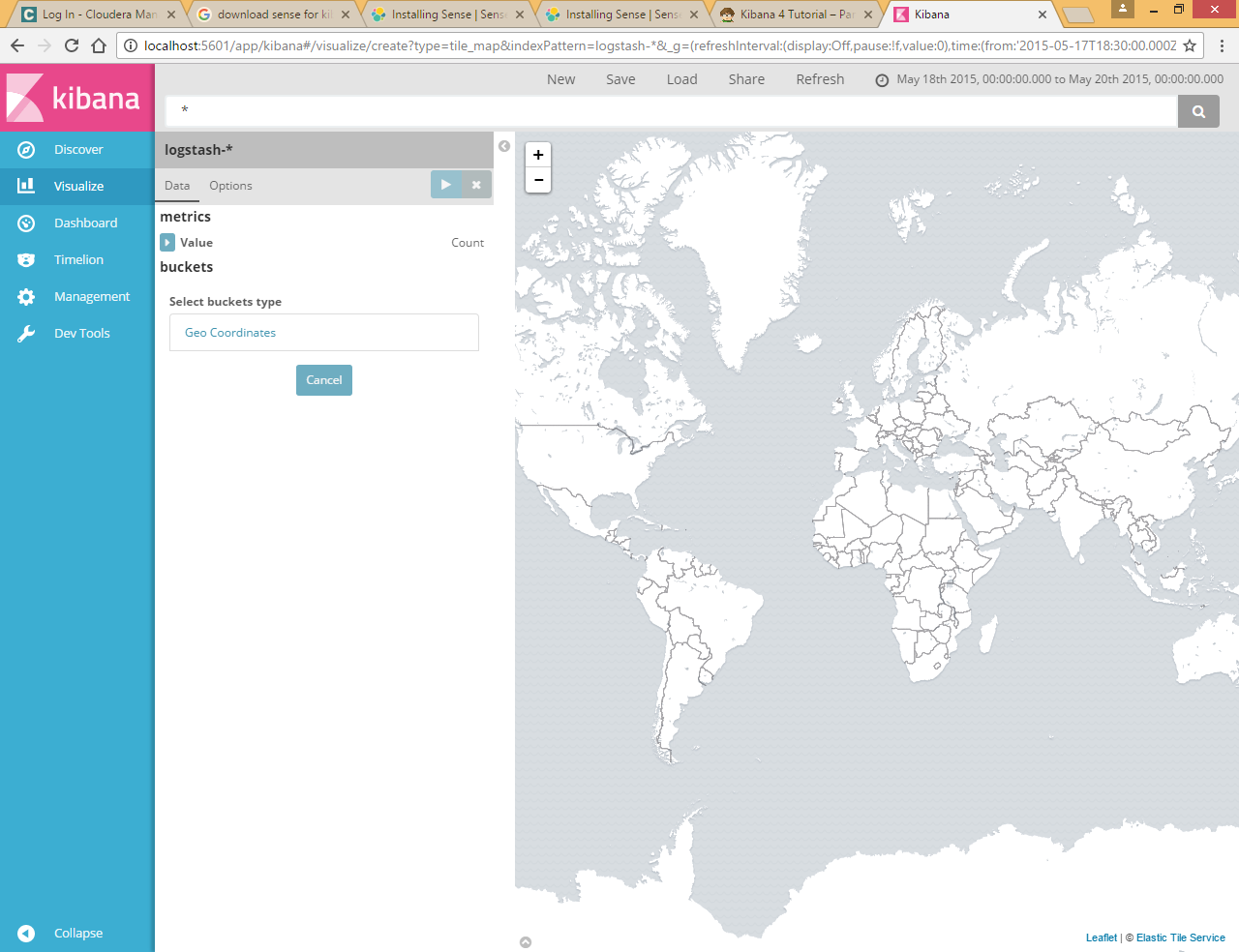
Next, we’re going to use a tile map chart to visualize geographic information in our log file sample data.

1. Click **New**.
2. Select **Tile map**.
3. Select the logstash-\* index pattern.
4. Set the time window for the events we’re exploring:
5. Click the time picker in the Kibana toolbar.
6. Click **Absolute**.
7. Set the start time to May 18, 2015 and the end time to May 20, 2015.

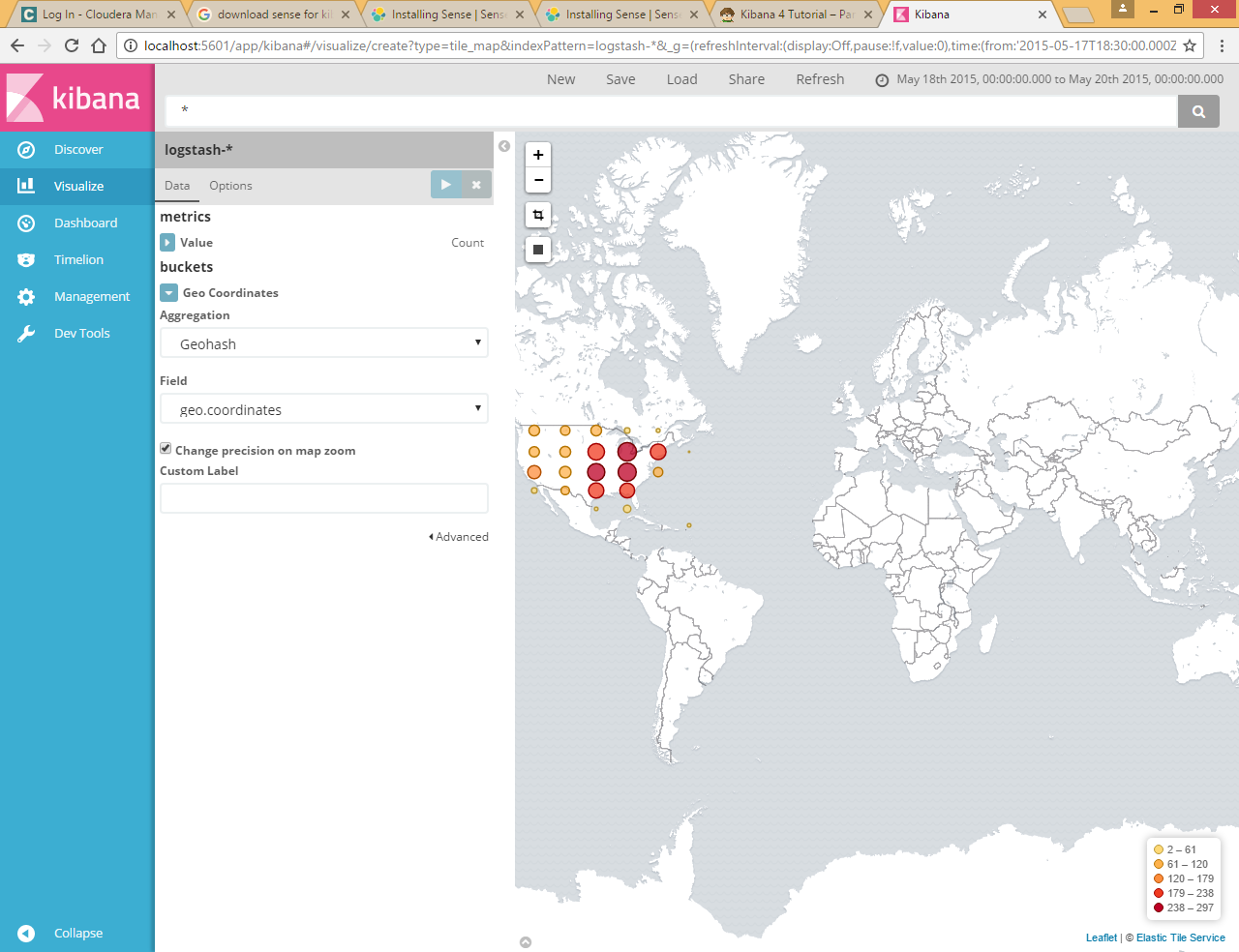


Once you’ve got the time range set up, click the **Go** button and close the time picker by clicking the small up arrow in the bottom right corner.

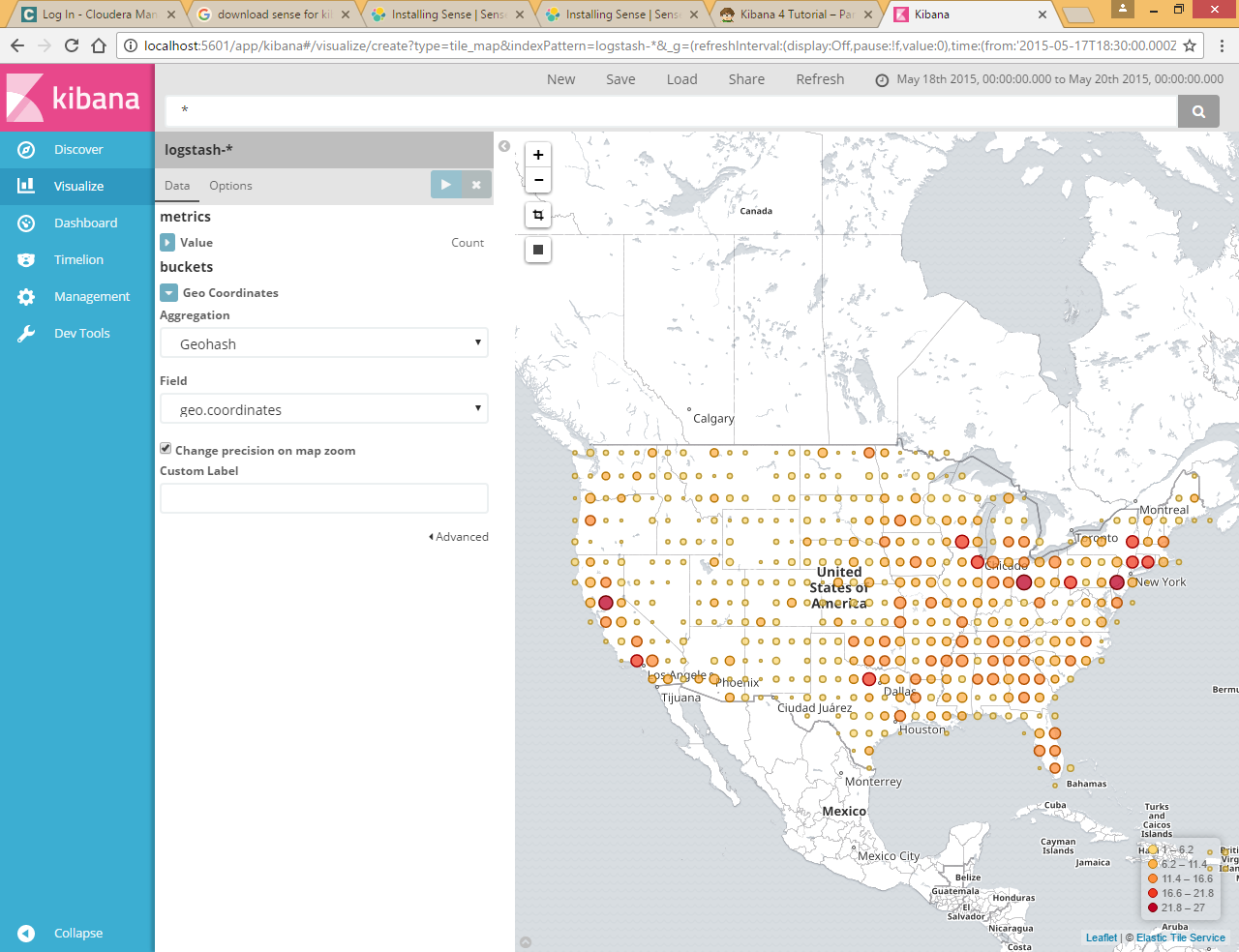
You’ll see a map of the world, since we haven’t defined any buckets yet:



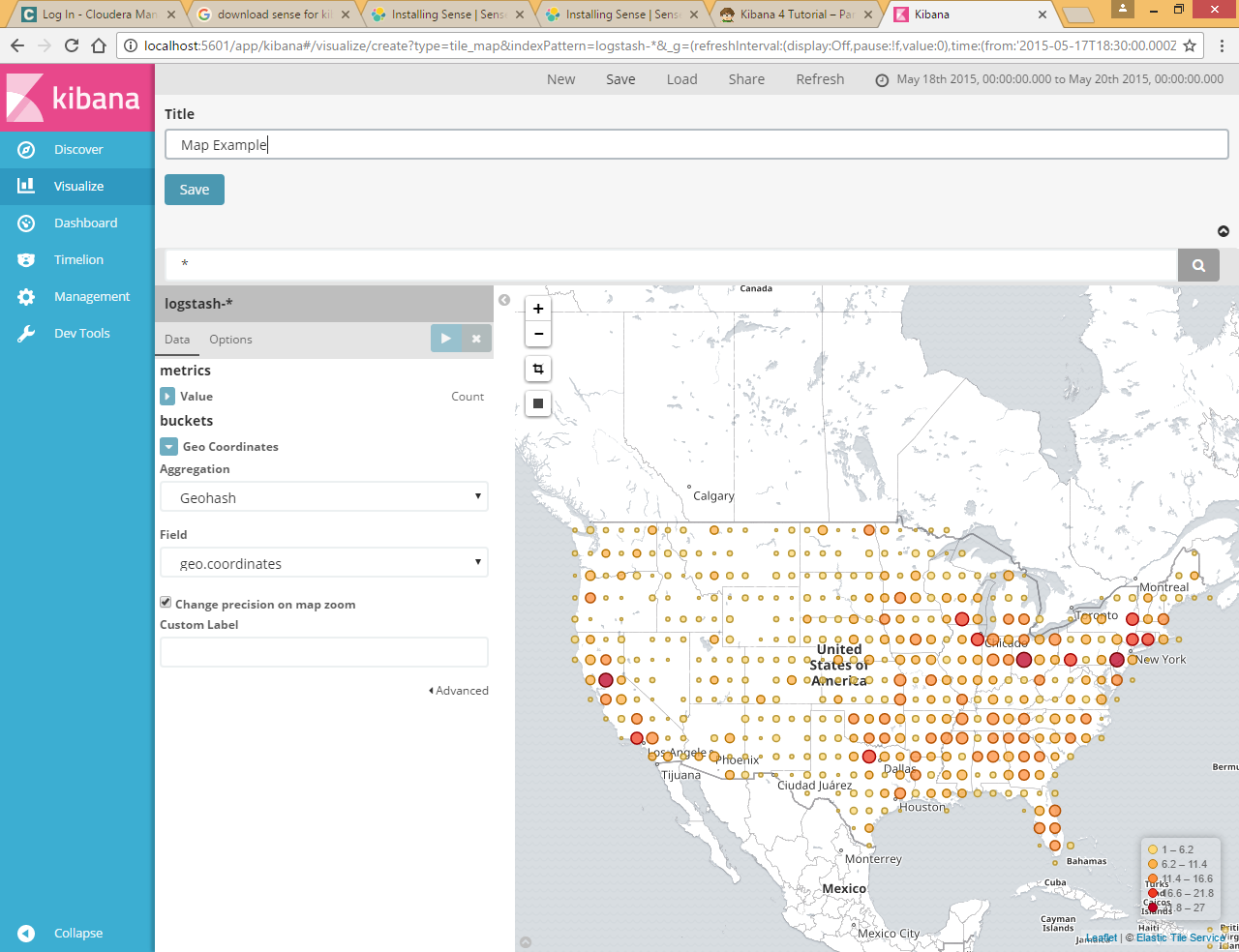
To map the geo coordinates from the log files select **Geo Coordinates** as the bucket and click **Apply changes** images/apply-changes-button.png. Your chart should now look like this:



You can navigate the map by clicking and dragging, zoom with the images/viz-zoom.png buttons, or hit the **Fit Data Bounds** images/viz-fit-bounds.png button to zoom to the lowest level that includes all the points. You can also include or exclude a rectangular area by clicking the **Latitude/Longitude Filter** images/viz-lat-long-filter.png button and drawing a bounding box on the map. Applied filters are displayed below the query bar. Hovering over a filter displays controls to toggle, pin, invert, or delete the filter.



Save this map with the name Map Example.



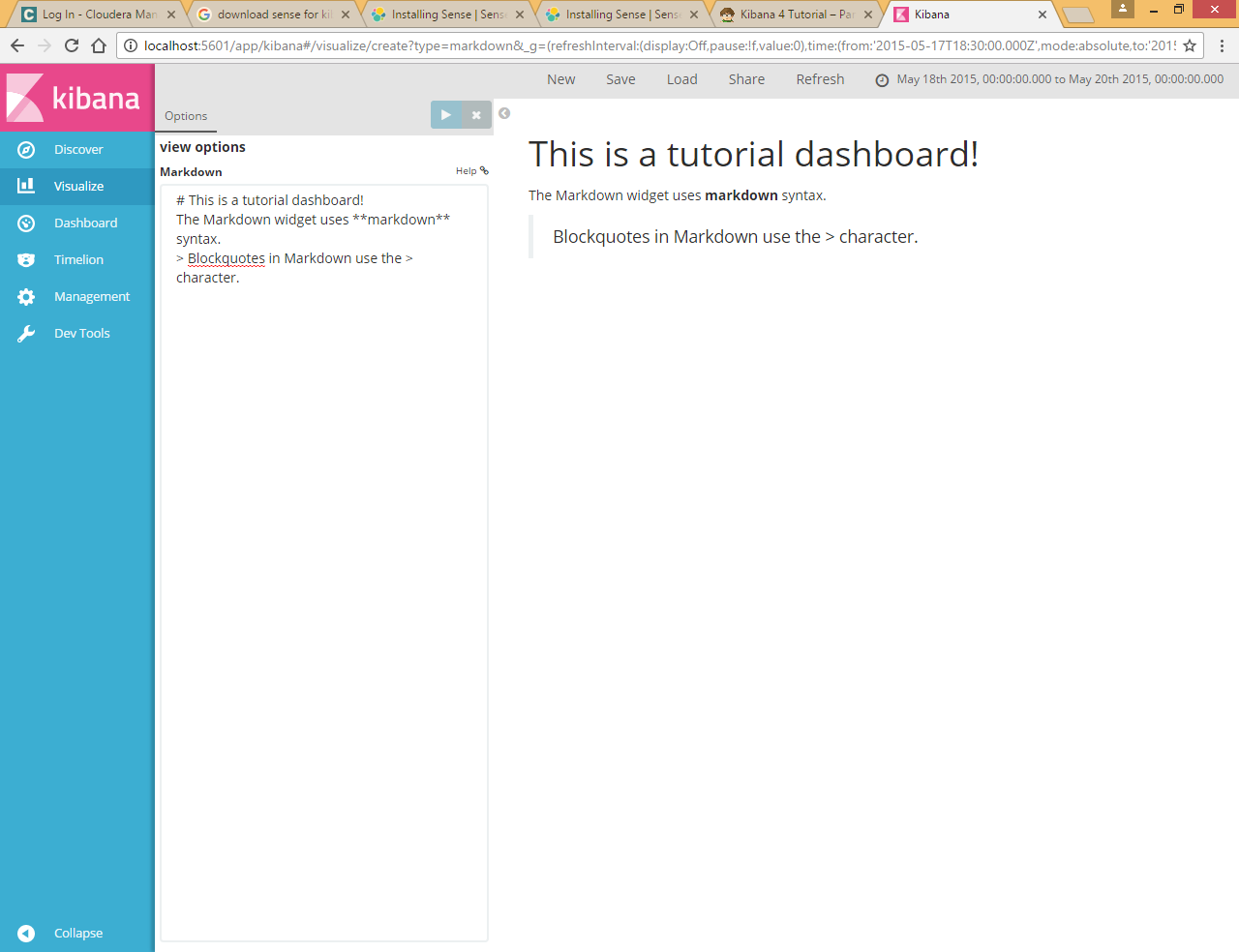
Finally, create a Markdown widget to display extra information:

1. Click **New**.
2. Select **Markdown widget**.
3. Enter the following text in the field:

# This is a tutorial dashboard!

The Markdown widget uses \*\*markdown\*\* syntax.

> Blockquotes in Markdown use the > character.

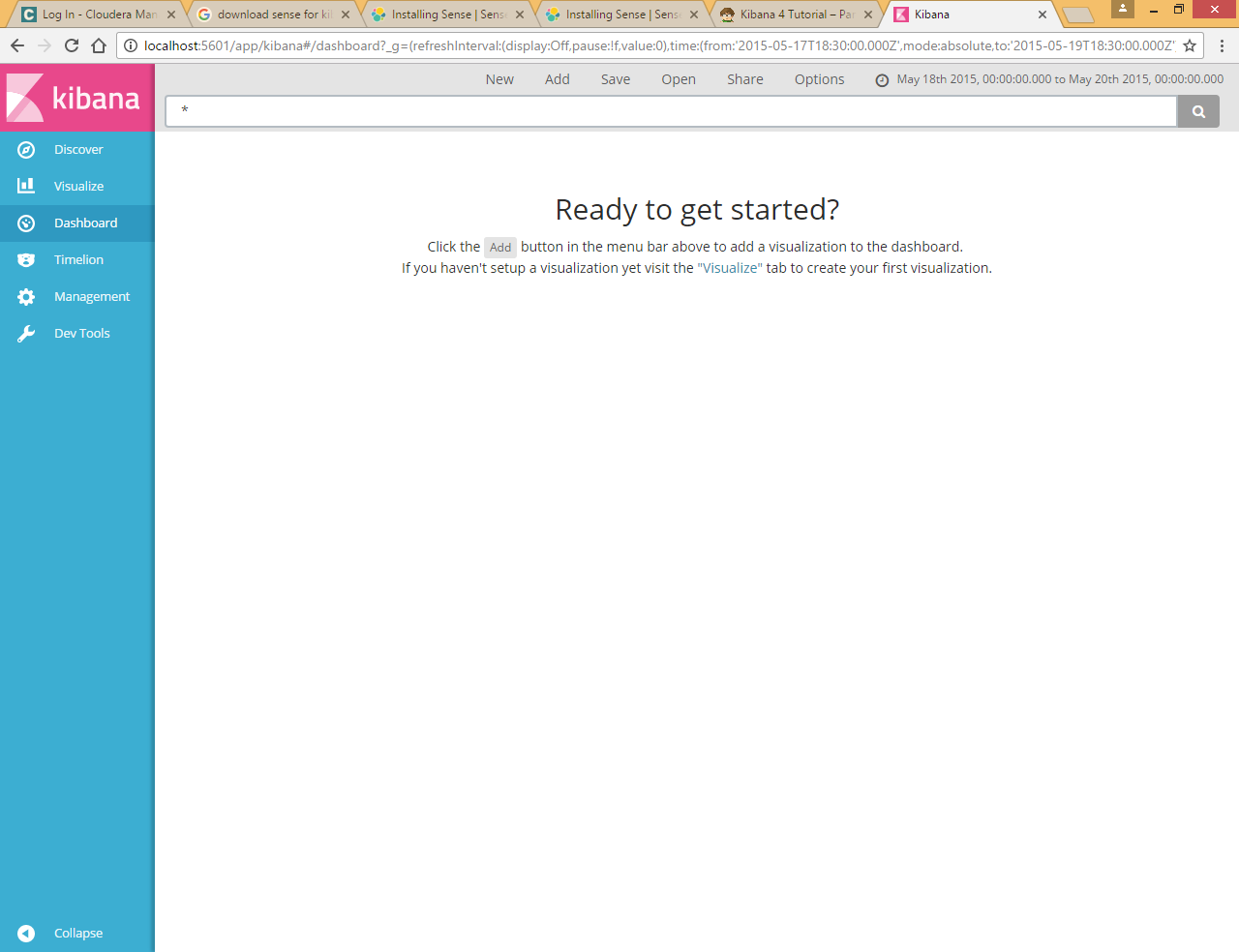


Save this visualization with the name Markdown Example.

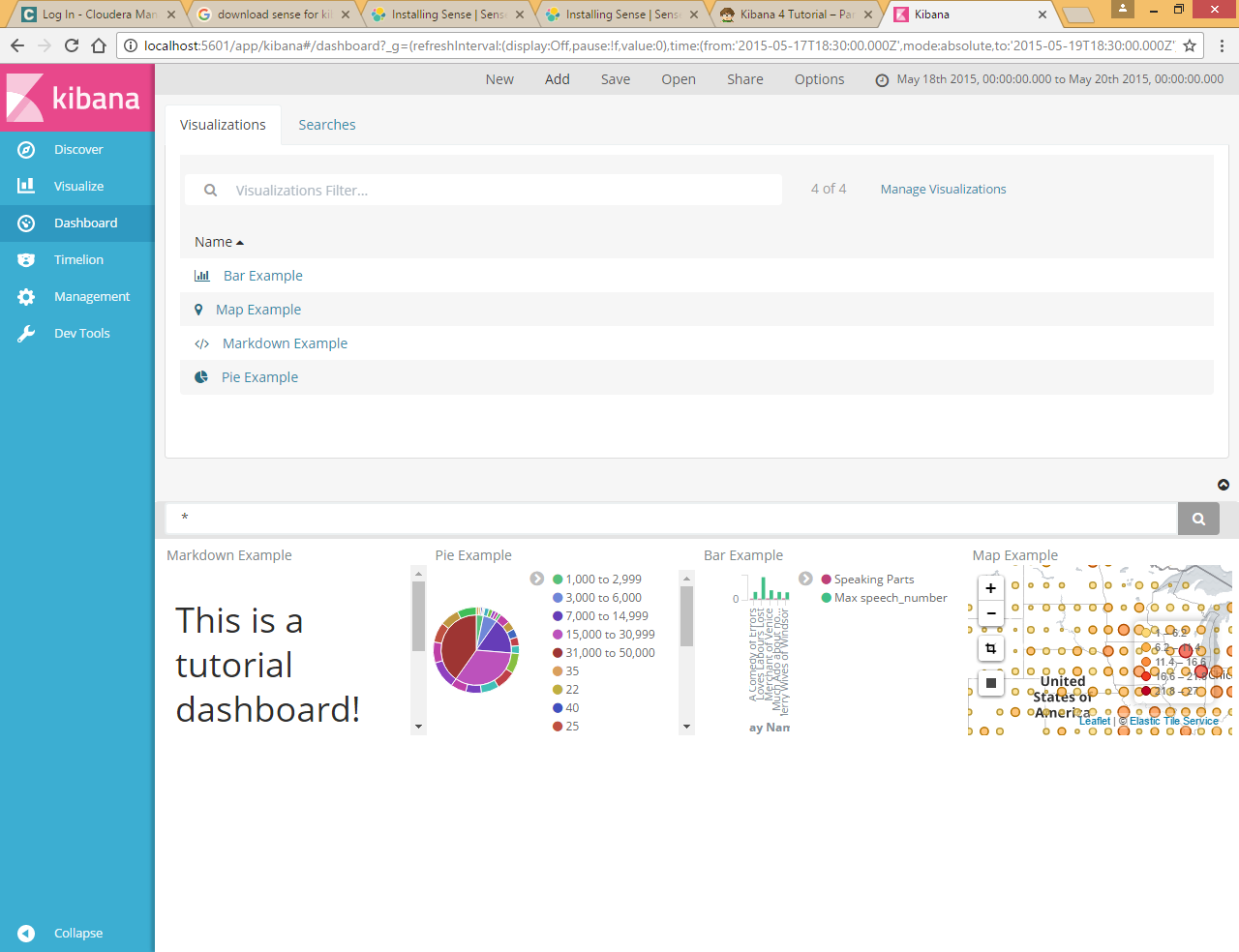
# Putting it all Together with Dashboards

A dashboard is a collection of visualizations that you can arrange and share. To build a dashboard that contains the visualizations you saved during this tutorial:

1. Click **Dashboard** in the side navigation.
2. Click **Add** to display the list of saved visualizations.

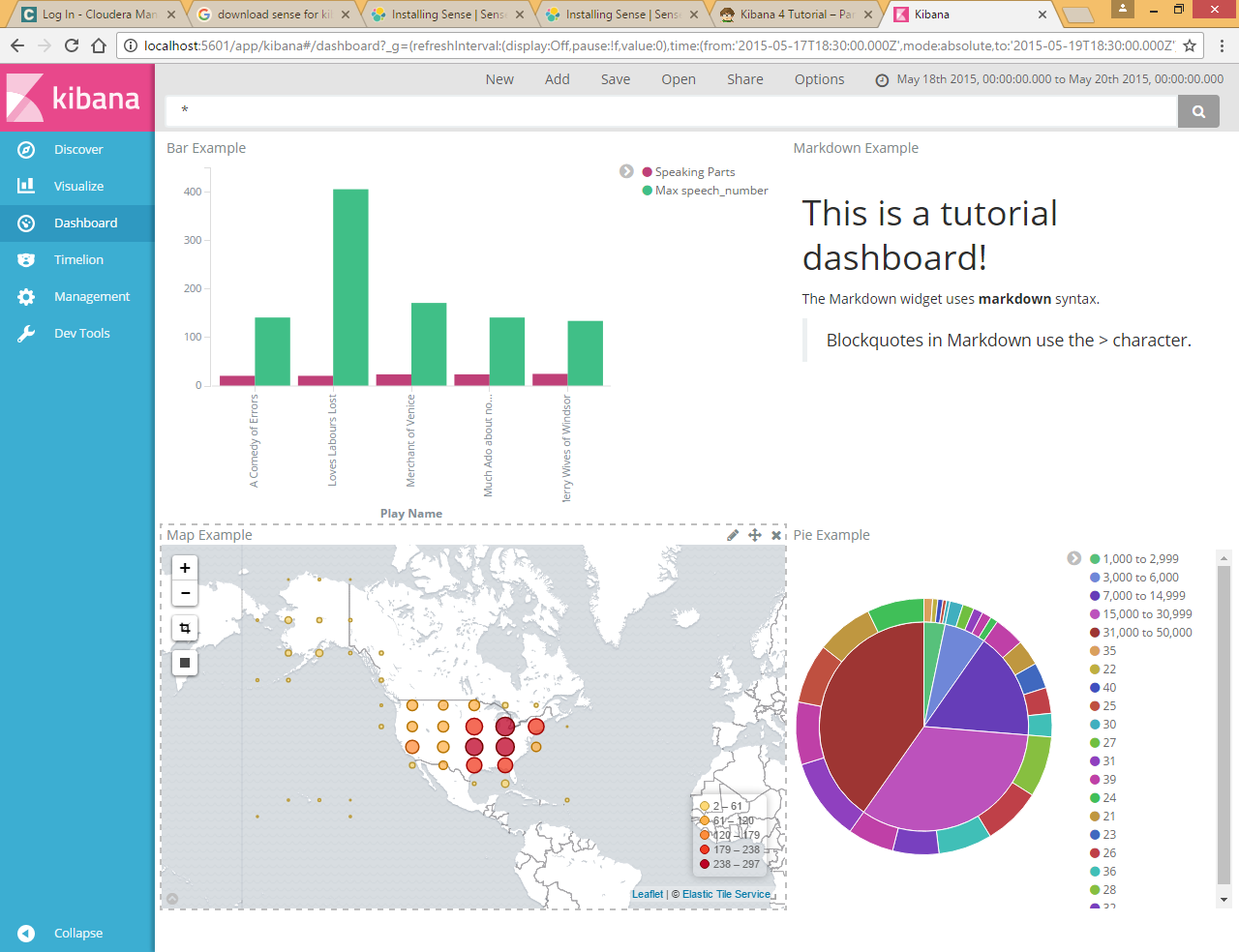


1. Click *Markdown Example*, *Pie Example*, *Bar Example*, and *Map Example*, then close the list of visualizations by clicking the small up-arrow at the bottom of the list.

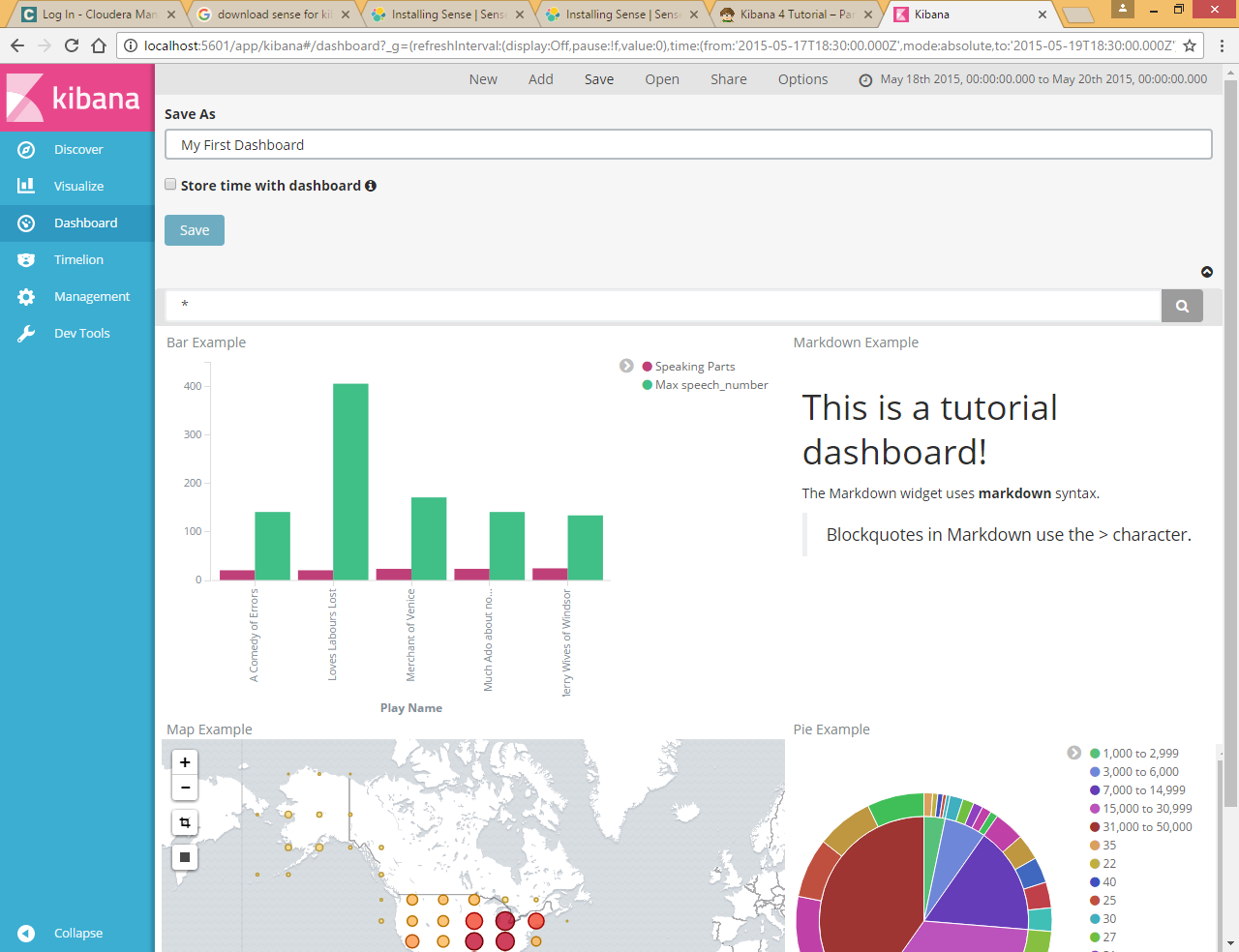


Hovering over a visualization displays the container controls that enable you to edit, move, delete, and resize the visualization.

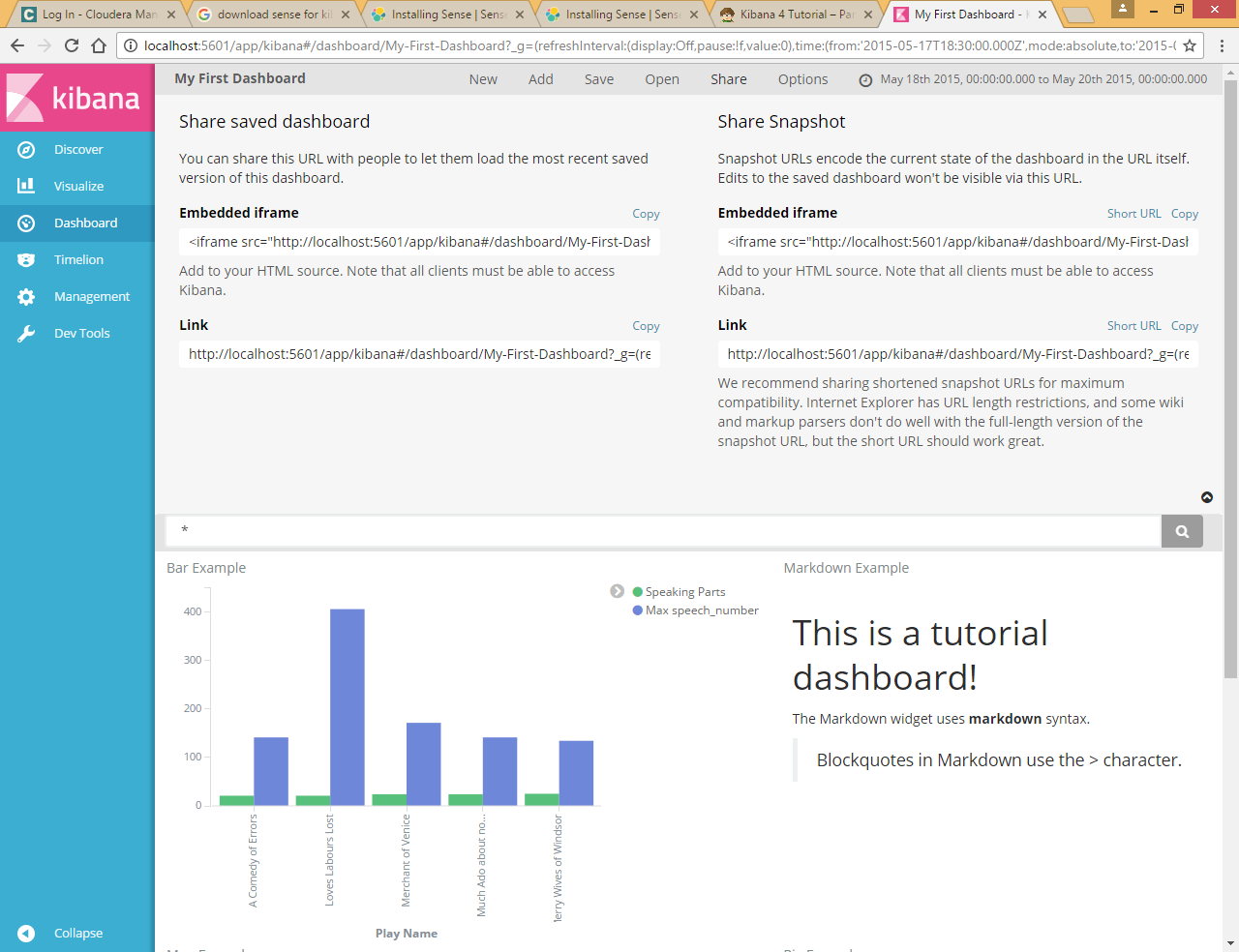
Your sample dashboard should end up looking roughly like this:



To get a link to share or HTML code to embed the dashboard in a web page, save the dashboard and click **Share**.



Click share



Copy html link

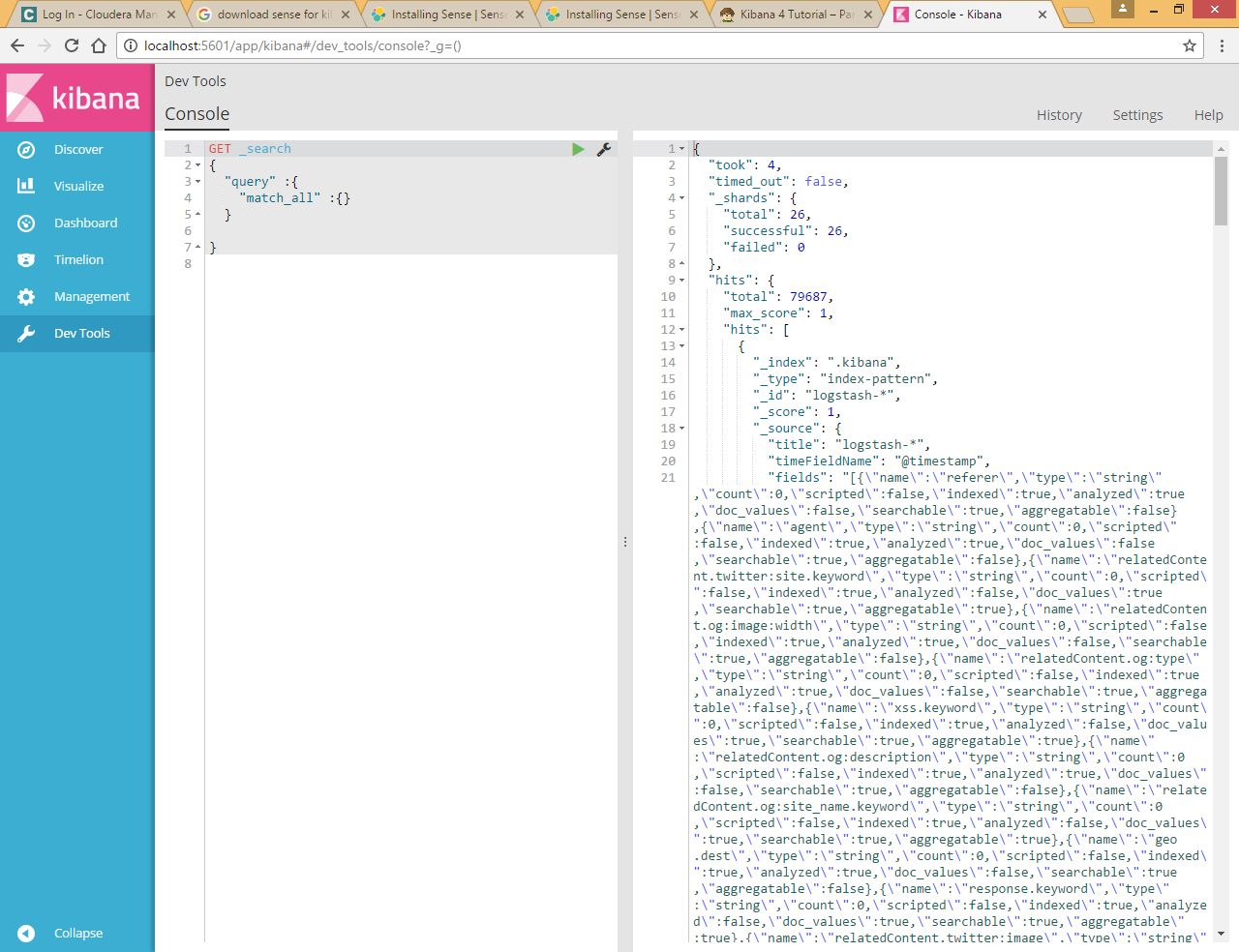
<http://localhost:5601/app/kibana#/dashboard/My-First-Dashboard?_g=(refreshInterval%3A(display%3AOff%2Cpause%3A!f%2Cvalue%3A0)%2Ctime%3A(from%3A'2015-05-17T18%3A30%3A00.000Z'%2Cmode%3Aabsolute%2Cto%3A'2015-05-19T18%3A30%3A00.000Z'))>

# Dev tool to use REST API for elasticsearch

We can use Dev tools to call the deferent method like PUT POST GET etc.

Click on Dev Tools

Return all result from all indexes



To search the keyword **300** enter **GET \_search?q=300** on consoled and execute

