

| | |
|--|----|
| 1. SiLK | 2 |
| 1.1 SiLK Installation | 2 |
| 1.2 Getting started | 3 |
| 1.3 Banana | 3 |
| 1.3.1 Banana Installation | 3 |
| 1.3.2 Dashboard Configuration | 4 |
| 1.3.2.1 Bettermap Panels | 4 |
| 1.3.2.2 Column Panels | 5 |
| 1.3.2.3 Filtering Panels | 5 |
| 1.3.2.4 Heatmap Panels | 5 |
| 1.3.2.5 Histogram Panels | 5 |
| 1.3.2.6 Hits Panels | 6 |
| 1.3.2.7 Map Panels | 7 |
| 1.3.2.8 Query Panels | 7 |
| 1.3.2.9 Range Facet Panels | 7 |
| 1.3.2.10 Table Panels | 8 |
| 1.3.2.11 Terms Panel | 8 |
| 1.3.2.12 Text Panels | 9 |
| 1.3.2.13 Ticker Panels | 9 |
| 1.3.2.14 Timepicker Panels | 9 |
| 1.3.3 Release Notes | 10 |
| 1.3.3.1 Release Notes for v1.3 | 10 |
| 1.3.3.2 Release Notes for v1.2 | 10 |
| 1.4 Solr Writer for Logstash | 10 |
| 1.4.1 Solr Writer for Logstash Initial Setup | 10 |
| 1.4.2 Indexing Content to Solr | 11 |
| 1.5 SiLK Examples and Recipes | 13 |
| 1.5.1 Apache Weblogs | 13 |
| 1.5.2 Search Analytics | 15 |

SiLK

SiLK stands for "Solr integrated with Logstash and Kibana". It includes a custom packaging of Solr, Banana and a Solr Writer for Logstash.

Banana is a data visualization tool that allows you to create dashboards to display content stored in your Solr indexes.

The Solr Writer for Logstash is a Logstash implementation geared toward indexing your log (or other content) to Solr.

Featured Pages

-  [Banana Installation](#)
-  [Getting started](#)
-  [SiLK Installation](#)
-  [Solr Writer for Logstash Initial Setup](#)

Search this documentation

Popular Topics

banana dashboard featured
histogram installation logstash panel recipe release-notes silk

SiLK Installation

The SiLK distribution provides Banana and the Solr Writer for Logstash in one package. The installation is straightforward: move the .tar.gz package to an intended location and unzip and untar it.

At that point, you can use Banana with your existing Solr implementation and can configure the Solr Writer for Logstash to index your log files. The following sections explain how to do this in more detail.

SiLK includes a distribution of Solr 4.8.1. To use the included instance of Solr, you only need to start Solr with `java -jar start.jar` command found in `$SiLK_HOME/SiLK-1.3/solr-4.8.1/SiLK`.

Banana Setup

While SiLK includes Solr 4.8.1, if you already have an instance of Solr running, you can simply copy the `$SiLK_HOME/banana` directory to the Solr webapp directory. The default location in a new Solr instance is `$SOLR_HOME/example/solr-webapp/webapp`, but the location may be different in your instance if you have moved Solr's webapp to another directory for your container. See also the section on [Banana Installation](#) for more information.

Changing the Default Domain and Port

If you want to change the port or the default domain, you should edit the `config.js` file found in the Banana webapp. Depending on how you started Banana, this file may be found in either `$SiLK_HOME/SiLK-1.3/solr-4.8.1/SiLK/banana-webapp/webapp` if you are running the included instance of Solr, or in `./solr-webapp/webapp/banana/src` if you have deployed Banana within your existing Solr webapp.

There is only one line that needs to be changed to modify both the domain and the port:

```
solr: "http://localhost:8983/solr/",
```

After saving the file, you will need to restart the container that is running the webapp.

It's also recommended to change the domain and port in the default dashboard to avoid errors being displayed to users. The location of the default dashboard also depends on how it was deployed and may be found in either of these locations:

- `$SiLK_HOME/SiLK-1.3/solr-4.8.1/SiLK/banana-webapp/webapp/apps/dashboards/dashboard.json` if deployed with the included Solr instance;
- `./solr-webapp/webapp/banana/src/app/dashboards/dashboard.json` if deployed within your existing Solr webapp container.

The section to be changed is found at the end of the file and looks like this:

```
"solr": {
  "server": "http://localhost:8983/solr/",
  "core_name": "collection1",
  "core_list": [
    "banana-int",
    "collection1",
    "logstash_logs",
    "logstash_logs_demo"
  ],
  "global_params": "&df=message"
}
```

Once Banana is set up, you can begin to work with [Dashboard Configuration](#).

Solr Writer for Logstash Setup

The Solr Writer for Logstash is a Logstash deployment with output options configured for Solr specifically. It is a standalone application, which means that it is not a webapp like Banana, but a process that is started to convert log data (or other types of data) into desired formats.

There are only a few short steps to setup Solr Writer for Logstash, detailed in the section [Solr Writer for Logstash Initial Setup](#).

Once your environment is ready, indexing data to your Solr instance is described in the section [Indexing Content to Solr with Logstash](#).

Getting started

In this section:

Banana

Banana is the name of the open source port of Kibana 3 available on GitHub at <https://github.com/LucidWorks/banana/>.

Banana is a tool to create dashboards to visualize data you have stored in Solr. Commonly used with Logstash for log data, any content stored in a Solr index is eligible for visualization in a Banana dashboard.

In this section, you can find information on [Banana Installation](#) (if you have not already installed it as part of [SiLK Installation](#)), and detailed information on [Dashboard Configuration](#).

Banana Installation

These instructions are to install the Banana Web application to run within your existing Solr instance. If you are looking to work with the full SiLK stack, please see [SiLK Installation](#).

You can get Banana running with your Solr system in these easy steps:

1. Check out the banana Git repository, found at <https://github.com/LucidWorks/banana/>.
2. Run Solr at least once to create the webapp directories:

```
$ cd $SOLR_HOME/example
$ java -jar start.jar
```

3. Copy the `banana` directory to the same location you are running the Solr webapp from, such as `$SOLR_HOME/example/solr-webapp/webapp/`.
4. Browse to <http://localhost:8983/solr/banana/src/index.html#/dashboard>. If your Solr port is different from the default port of 8983, you can edit `banana/src/config.js` and enter the port you are using.

If you have not yet created the data collections and ingested log data into Solr, you will see an error message saying "Collection not found at ..". See the section [Solr Writer for Logstash](#) to learn how to import data into your Solr instance using Logstash.

If you want to save and load dashboards from Solr, copy either `solr-4.4.0/banana-int` (for Solr 4.4) or `solr-4.5.0/banana-int` (for Solr 4.5 and above) directories (as appropriate) into `$SOLR_HOME/example/solr` in order to setup the required core. Then restart Solr, and if it is not loaded already, load the core from Solr Admin (if you are using Solr Cloud, you will need to upload the configuration to ZooKeeper and then create the collection using that configuration).

Dashboard Configuration

The Banana dashboard is made up of several panels that are each configured to show the data you would like to see, in the format you would like to see it.

The panels are arranged in rows, and each row can be hidden. Each row can contain a number of panels; the exact number per row depends on the size of the panel.

- [Configuring Rows](#)
- [Adding & Editing Panels](#)



When configuring the dashboard, be sure to save your changes by clicking the **Save** icon at the upper right.

Configuring Rows

To add a row, click **Add a Row** beneath the last row.

This will bring up the Dashboard Settings menu. To add a row, enter a title for the new row, the height of the row, if it is editable, and then click **Create Row**. After the row is created, it will appear in the table of rows, and you can click the up or down arrows to arrange the new row with the existing rows. You can also use this screen to delete a row, if you'd like.

When you are done, click **Close** to get back to the dashboard.

Adding & Editing Panels

To add a panel to a row, click the **+** icon to the right of the last panel in a row; if this icon does not appear, the row is full.

You can also click the 'gear' icon to the left of the first panel. This will bring up the row configuration popup, where the last tab is 'Add Panel'.

When adding a panel, the configuration screen varies depending on the required properties for each panel type. From the Add Panel tab, all of the available properties of each type are displayed, allowing you to define the panel name, the data properties, and any queries that should be used to limit the data used in the panel.

When editing a panel, however, the view is split between three tabs: General, for name and size configuration; Panel, for the data properties; and Queries, for defining queries.

The following sections describe the types of panels that can be configured:

- [bettermap](#)
- [column](#)
- [filtering](#)
- [heatmap](#)
- [histogram](#)
- [hits](#)
- [map](#)
- [query](#)
- [rangeFacet](#)
- [table](#)
- [terms](#)
- [text](#)
- [ticker](#)
- [timepicker](#)

Bettermap Panels

The bettermap panel displays geographic points in clustered groups on a map.

This panel type does not use the terms facet and it does query sequentially. This means that it transfers more data and is generally heavier to compute, while showing less actual data

If you have a time filter, it will attempt to show to most recent points in your search, up to your defined limit.

Column Panels

A column panel allows you to add other panels inside it.

The properties of a column panel allow you to define the panels you would like included.

Choose the type of panel, and then define the properties according to the type chosen. When you have finished configuring the included panel, click Create Panel.

Below the Add Panel to Column area, the configured panels will be shown as a list, in the order they will appear on the dashboard. From this list, you can change the height of each panel, remove panels, change the display order, or temporarily hide panels.

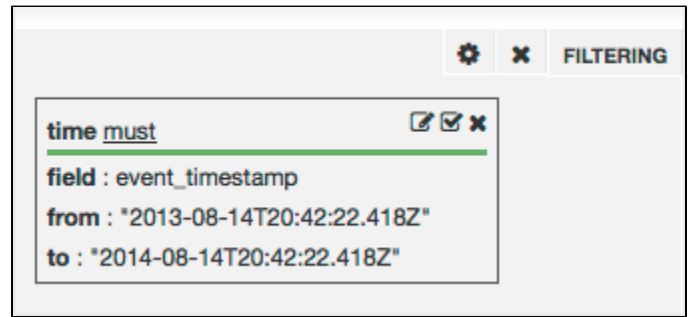


Filtering Panels

The filtering panel allows you to see the types of field or date limitations that have been applied to the dashboard.

It is a best practice to have a filtering panel on every dashboard so you can always see the limitations that are being applied to the data being displayed.

There are no properties to a filtering panel.



Heatmap Panels

The heatmap panel provides a heat map for representing pivot facet counts.

Panel tab

The Panel tab defines what data is displayed in the panel and how it is displayed.

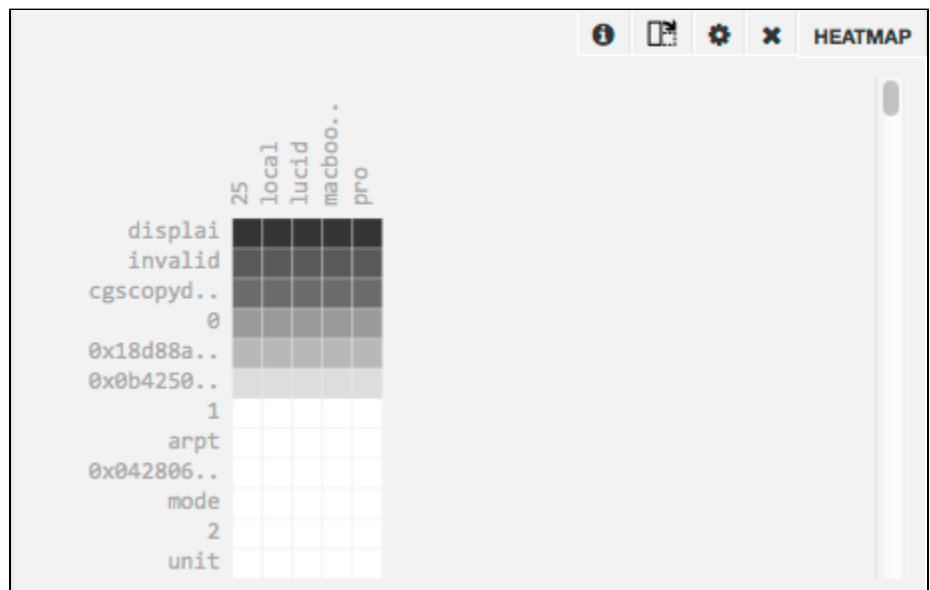
Row Field: The field from Solr that will provide data for rows.

Column Field: The field from Solr that will provide data for columns.

Rows Limit: The maximum number of rows to display.

Heatmap Color: The base color for the heatmap. The intensity of the color in any cell is proportional to the pivot facet count for that cell.

Transposed



Histogram Panels

The histogram panel provides a bucketed time series chart of the current query or queries, using Solr's range facets for data.

If using time-stamped indexes, this panel will query them sequentially to attempt to apply the lightest possible load to your Solr instance or cluster.

Configuration of this panel allows several parameters.

Mode

The value for the y-axis. The options are **count** or **values**.

When choosing values, you must select a field to use as the basis for the values. This field must have a numeric field type. You can also select a Group By Field, which cannot be a multi-valued field, and creates multiple charts.

Time_field

The value for the x-axis. This is the field to use for display of the time information for the histogram.

Chart Settings

There are several chart settings.

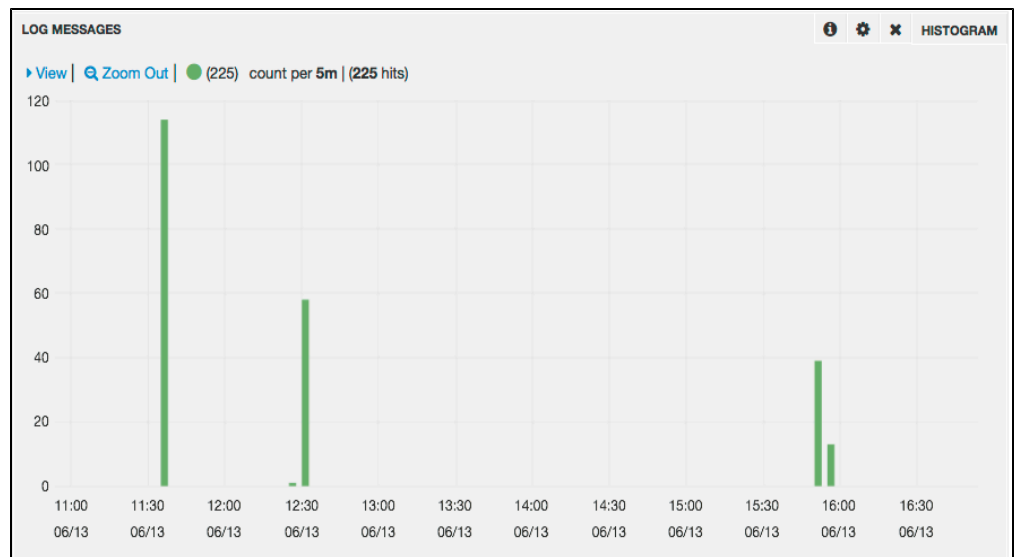
- **Bars:** Enable to show bars on the chart.
- **Lines:** Enable to show lines on the chart. When enabled a few more settings will be available:
 - **Line Fill:** The fill area, from 0-10.
 - **Line Width:** The width of the lines, in pixels.
 - **Smooth:** Enable to remove 0 values from lines.
- **Points:** Enable to show points on the chart.
- **Stack:** Enable to stack multiple series together.
- **Percent:** Enable to show the stack as a percentage of the total (only displayed when stack is enabled).
- **Legend:** Enable to show the legend.
- **xAxis:** Enable to display the x-axis values.
- **yAxis:** Enable to display the y-axis values.
- **Time correction:** If time correction should be applied to use the browser's timezone. Select 'utc' to always display times in UTC.
- **Selectable:**
- **Zoom Links:** Enable to allow users to zoom out in time.
- **View Options:** Enable to show an options section.
- **Auto-interval:** Enable to allow the chart to automatically scale the intervals.
- **Resolution:** When Auto-interval is enabled, a best effort will be made to show this number of bars.

Tooltip Settings

The tooltip settings control the display of data when users hover over a line or bar on the chart.

- **Stacked Values:** When using stacked values, this defines if the data be displayed as cumulative, or as individual values.
- **Display Query:** If an alias is set, it will be shown in the tooltip. If no alias is set, enable this to show the entire query.

Hits Panels



Histogram Settings

General | **Panel** | **Queries**

Mode: count | **Time Field:** timestamp_dt

Chart Settings

| | | | | | | | |
|-------------------------------------|--------------------------|--------------------------|-------------------------------------|--------------------------|-------------------------------------|-------------------------------------|-------------------------------------|
| Bars | Lines | Points | Stack | Percent | Legend | xAxis | yAxis |
| <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |

Time correction: browser | **Selectable:** ☒ | **Zoom Links:** ☒ | **View Options:** ☒ | **Auto-interval:** ☒ | **Resolution:** 100

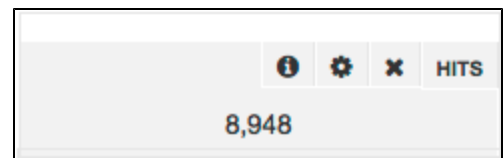
Tooltip Settings

Stacked Values: cumulative | **Display Query:** ☐

Close

The hits panel shows the total hits for the current query input to a query panel.

The properties allow providing a title for the panel, the style of results, and the font size.



Map Panels

A map panel displays a map of shaded regions using any field that contains a 2-letter country or US state code. Regions with more hits are shaded darker.

This uses the Solr terms facet, so it is important that you set it to the correct field.

The following properties are defined for a map panel:

Field: the Solr field that has the 2-letter codes that will be used for the location data.

Max: a maximum number of countries to plot. The default is 100.

Map: the style of map to display. If your data spans the world, you can choose the world map. If your data is focused on Europe or the US instead, you can choose the Europe or US maps respectively.

Mode: the approach to summarizing the data. You can choose count, mean, maximum, minimum or sum. In order to use any mode other than count, the field type must be numeric.

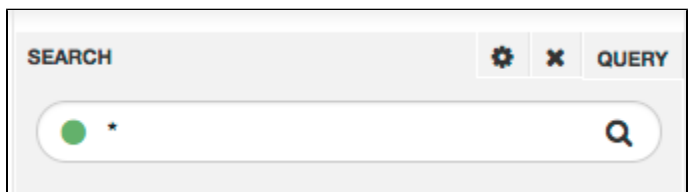
Decimal Points: the number of digits to display after a decimal points.

Query Panels

The query panel provides a search box to allow real-time filtering of data.


It is a best practice to include one of this type of panel on your dashboard. With this panel, you can add, remove, label, pin and color queries.

There are no specific properties for a query panel.



Range Facet Panels

A rangeFacet panel provides a histogram view across any numeric field using Solr's `range.facet` functionality. It is similar to the time series histogram, and allows selection of ranges and zooming in or out to the desired numeric range.

 Range selections in the panel are reflected across the entire dashboard.

The properties are very similar to the options for a histogram chart, and defined below.

rangeField

The rangeField defines the field in the index to use as the basis for the range. It must use an integer-based field type.

Chart Settings

The chart settings define how the data will be displayed. There are several options:

- **Bars:** Enable to show bars on the chart.
- **Lines:** Enable to show lines on the chart. When this option is chosen, additional options are available:
 - **Line Fill:** The fill area, from 0-10.
 - **Line Width:** The width of lines, in pixels.
 - **Smooth:** Enable to remove 0 values from lines.
- **Points:** Enable to show points on the chart.
- **Legend:** Enable to show the chart legend.
- **x-axis:** Enable to show the x-axis values.
- **y-axis:** Enable to show the y-axis values.
- **Selectable:**
- **Zoom Links:** Enable to allow users to zoom out in time.
- **View Options:** Enable to show an options section.

- **Minimum:** Define the minimum value to display.
- **Maximum:** Define the maximum value to display.
- **Interval:** Define the interval between displayed values.
- **Unit for Legend:**

Table Panels

The table panel allows you to create a table of field values from the Solr index that match the filters applied to the dashboard. While only the fields selected are displayed, clicking on any entry expands to show all fields of the document.

You can also export the documents, if needed, to CSV, XML or JSON by clicking the "Export" icon in the upper right of the panel.

| message |
|---|
| Feb 25 15:29:47 Lucids-MacBook-Pro-25.local Microsoft Outlook[36342]: CGSCopyDisplayUIID: Invalid display 0x18d88a81 |
| Feb 27 03:13:39 Lucids-MacBook-Pro-25 kernel[0]: IOPPF: Sent cpu-plimit-notification last value 1 (rounded time weighted average 1) |
| Feb 27 04:13:50 Lucids-MacBook-Pro-25 kernel[0]: AppleCmIn::wakeEventHandlerThread |
| Feb 27 06:16:37 Lucids-MacBook-Pro-25.local WindowServer[91]: Display 0x03f003d: GL mask 0x2; bounds (4386, 0)[1 x 1], 1 modes available |
| Feb 24 16:37:58 Lucids-MacBook-Pro-25.local WindowServer[98]: Display 0x03f0040: GL mask 0x10; bounds (2464, 0)[1 x 1], 2 modes available |
| Feb 24 08:31:45 Lucids-MacBook-Pro-25.local Microsoft Word[375]: CGSCopyDisplayUIID: Invalid display 0x18d88a81 |
| Feb 23 22:40:09 Lucids-MacBook-Pro-25 kernel[0]: Wake reason: RTC (Alarm) |
| off-line, enabled, Vendor #####, Model #####, S/N #####, Unit 2, Rotation 0 |
| Feb 26 10:52:54 localhost kernel[0]: 6.30.223.154 (420397) |
| Feb 24 12:21:24 Lucids-MacBook-Pro-25.local Finder[203]: CGSCopyDisplayUIID: Invalid display 0x0b425001 |

In the display, you can add new fields on the fly, but clicking a field name from the list in the left side of the panel.

There are several properties you can configure:

Add Column

Enter the field(s) you would like to see in the table. You can enter as many fields as you'd like, but too many columns may require you to scroll horizontally to see all of your data.

Click the '+' button to add the field, and you should see it listed in the **Columns** section to the right. Click on a column to remove it from the list.

Options

The display options allow defining how the table is displayed.

- **Header:** Enable to see a header row in the table.
- **Sorting:** Enable to be able to sort the data in the table.
- **Sort:** Choose the field to sort on. Also choose the sort order by selecting the up or down carat to the right of the column name.
- **Font Size:** Choose the font size for the display of the data.
- **Trim Factor:** If the data is too long to fit in the column at your desired width, you can configure the point at which the data will be "trimmed" from display.

Paging

The paging options allow control over how to deal with large amounts of data that may be easier to work with on separate pages.

- **Show Controls:** Enable to show page forward and back options.
- **Overflow:** If the data expands beyond the height of the window, you can choose to **scroll** through the records, or enable **overflow** to expand the size of the panel to fit all of the data for that page.
- **Per Page:** The number of items to display on each page.
- **Page Limit:** The number of pages to display.
- **Pageable:** This will automatically update based on the values of Per Page and Page Limit, to show the total number of items that will be displayed in the table.

Terms Panel

The terms panel displays the results of a Solr facet as a pie chart, bar chart, or a table.

A statistics field can be displayed as min/max/mean/sum, faceted by the Solr facet field, again as a pie chart, bar chart or a table.

A terms panel takes several properties, described below.

- **Field:** The field to use as the basis of the facets that are used for display.
- **Length:** The maximum number of terms to display.
- **Order:** The sort order of the facets.
- **Style:** The style of chart, either **bar**, **pie** or **table**.
- **Legend:** If you choose bar or pie as the style, you can then choose no legend, or to display it above or below the data.

- **Font Size:** If you choose table, you will be given the option to define the font size.

- **Missing**

: Enable to display missing values.

- **Other**

:

- **Donut**

: If you choose pie chart as the style, you can choose to display the chart as a donut, with an empty circle in the middle.

- **Tilt**

: If you choose pie chart as the style, you can choose to tilt the chart as an added effect.

- **Labels**

: Enable to show labels in the chart for your data.

- **Mode**

: The mode for the data. Choose

count

, **mean**

min

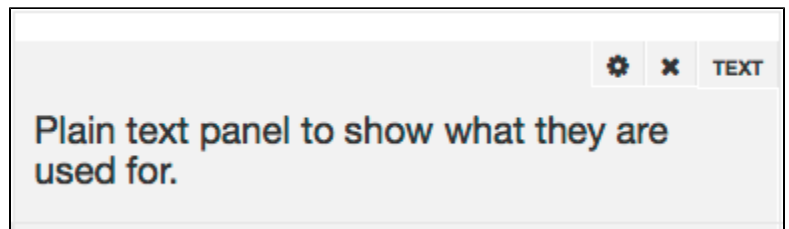
, **max**, or **sum**. If choosing any mode other than count, the Stats Field selected must be a numeric field.

- **Stats Field:** If you choose any mode other than count, you must then specify the field to use for statistics. This field must be a numeric field.
- **Display Precision:** Choose the number of digits to display after a decimal point, as appropriate.

Text Panels

A text panel is a free-text panel that allows you to add descriptions or other text on your dashboard. You can use plain text, [Markdown](#), or HTML.

When configuring the panel, choose the mode and then enter the text to display. When entering plain text, you can also define the font size to use with the display.



Ticker Panels

The ticker panel provides a stock-ticker style representation of how queries are moving over time.

When configuring a ticker panel, there is one primary property, "Time Ago", which defines the point in time to use as the basis for comparison.

For example, if the time is 1:10pm, your time picker was set to "Last 10m", and the Time Ago parameter was set to '1d', the panel would show how much the query results (from the [query panel](#)) have changed since 1:00 to 1:10pm yesterday.

In the example at the right, the timepicker has been set to 30 days, and the ticker panel is set to 1 day. The 0% tells me that there have been no new documents added to my index in the last day.



Timepicker Panels

The timepicker panel allows control over the time range filters.

If you have time-based data, or if you're using time stamped indices, you need one of these



You can configure the panel with several properties:

- **Default Mode**

: The options are

relative

, which provides a series of relative timeframes (such as 30 days ago, 1 year, etc.); **absolute**, where you define the start and end dates; or **since**, where you define only the starting date, with the current date assumed.

- **Time Field:** The field to use for time-based data.
- **Relative Time Options:** When the mode is set to relative, you can provide a comma-separated list of relative time options, such as "5m,1h,2d". If you use the default range, you should set the panel to span at least 6, to prevent the time selections from overrunning the edges of the panel.
- **Default Timespan:** The time option that should be selected as a default.
- **Auto-refresh:** When the mode is set to either relative or since, you may want your dashboard to automatically refresh with the latest data. These options allow you to configure auto-refresh:
 - **Enable:** Select to enable auto-refresh.
 - **Interval:** The interval, in seconds, to refresh.
 - **Minimum Interval:** The minimum interval, in seconds, to refresh.

Release Notes

Release Notes for v1.3

Release Notes for v1.2

Banana 1.2: Released on 11 May 2014

Following release 1.1, we have addressed a number of user requests, including:

1. This release provides panels for representing geo-spatial data—a `map` module that provides a heat map-style representation based on two-letter country codes or US state codes, and a `bettermap` module that provides a clustered representation of location (`LatLngType`) data.
2. The `Table` Module now has a `Save` button that enables you to save to csv, JSON or XML formats so that you can use other tools like Microsoft Excel for further analysis. The number of rows downloaded will be equal to number of “pageable” hits configured in the *Paging* tab within the *Table Panel Configuration Menu* (accessed by clicking on the cog wheel icon near the top right of the table panel).
3. You can now control whether a dashboard can be saved and/or edited from the *Editable* checkbox in the *General* tab, and the *Controls* tab, both within the *Dashboard Configurator* (accessed from the cog-wheel icon to very top and right of dashboard).
4. We have added a *hits* panel that provides you with the number of matching results returned while using the global query parameters. This is useful if you want to make the number prominent or if you are not using the histogram panel prominently.
5. You can now provide additional *Global Query Parameters* that apply to all panels of the dashboard from the *Solr* tab in the *Dashboard Configurator*. Among other uses, this feature is invaluable for:
 - a. Specifying a custom query parser (Solr query parameter: `&defType`) or search handler (`&q`)
 - b. Specifying a user type for use in custom business rules at the Solr server.
 - c. Specifying default search fields (`&df`)
6. We fixed a bug in the *values* mode within the *histogram* module, where missing values were previously assumed to be zero. This led to jagged graphs when the “group by” option was used. We no longer set them to zero but rather have the individual lines skip the missing values.
7. In the *Absolute Time* and *Since* modes, the *timepicker* used to skip back one day if your browser time was behind UTC. This issue has now been fixed.
8. Banana 1.1 hardcoded certain default search fields (i.e., `df`) to work with our LogStash output writer. Specifically, it hardcoded a `df=message`. This means that your old dashboards may not be fetching query results with Banana 1.2, though they were doing so with 1.1. To fix this, add a *Global Query Parameter* `&df=message` (or whatever field you want to search on) within the *Dashboard Configurator*. Alternately, you can set the default search field in your `solrconfig` (recommended).

Solr Writer for Logstash

The Solr Writer for Logstash is an implementation of Logstash specifically designed for output to Solr.

Solr Writer for Logstash Initial Setup

If you are not using the Solr Writer for Logstash from the [SiLK distribution](https://github.com/LucidWorks/solrlogmanager/), you should clone the Github repository <https://github.com/LucidWorks/solrlogmanager/> and copy the `logstash_deploy` directory to any location.

Solr Writer for Logstash Setup

Create a Collection for the Event Data

The first step is to create a collection to hold the event data.

It's recommended to create the collection so it uses the managed schema functionality of Solr. Also referred to as "schemaless", this allows fields to be created in the schema on the fly, as it were, saving you from needing to create the fields in advance of indexing your log files. More information about the managed schema is found in the Solr Reference Guide section [Schemaless Mode](#).

If you do choose to allow Solr to determine the appropriate fields as log events are being indexed, there are two fields that must be defined in your schema. These fields are 'timestamp' and 'version'. If these fields do not exist in your schema, they will be added at start of the Logstash job with a prefix defined in the configuration file for the log processing (such as 'logstash_version' or 'event_timestamp'). If created automatically, these fields will have the following configuration:

- Timestamp
 - type = tdate
 - stored = true
 - indexed = true
- Version
 - type = long
 - stored = true
 - indexed = true



The 'version' field discussed here is not the same as the `_version_` field that is required for SolrCloud and associated internal tlog processing.

Place Files in Proper Directories

First, make sure that the `logstash_deploy` directory contains a `.jar` file called `lucidworks.jar`. If it does not, you will need to find it and

Then, copy the file `lucidworks_solr_lsv133.rb` to the Logstash outputs directory. If using the `logstash_deploy` directory, the default location for the output is `./logstash_deploy/logstash/output` and you should find the file already there.

From this point, your next step is to set up your configuration files to index content, as described in the next section [Indexing Content to Solr](#).

Indexing Content to Solr

If you have completed the initial setup as described in the section [Solr Writer for Logstash Initial Setup](#), you can start indexing content to your Solr instance.

Note that you should already have created a collection in Solr, or identified an existing collection, that you want to use to store the log events index.

Modifying the Logstash Configuration File

The Logstash configuration file defines the location of the logs to be processed, how events should be parsed, and where the output should be sent.

Solr Writer for Logstash includes a configuration file that can be modified for your environment. The file is `lw_solr.conf` and is found in the `logstash_deploy` directory.

There are several properties to modify in the configuration file, described below.

Path to input files

The `lw_solr.conf` file starts with the input definitions, as shown in this example:

```

input {
  file {
    type => "syslog"
    exclude => [ "*.gz", "*.zip", "*.tgz" ]
    # FYI - Logstash does not always recurse the directory hierarchy correctly
    on Windows
    # unless the path is all lowercase.
    path => [ "/logfilePath/**/*./*" ]
    sincedb_path => "/dev/null"
    start_position => "beginning"
  }
}

```

To modify the path, change the location in line 8 to the correct location of the log files to be processed.

Field definitions

The next section of the file addresses filtering content.

```

# Add name=value pairs as fields
filter {
  kv{
    add_field => [ "User_{user}_says", "Hello world, from %{src_ip}" ]
    add_tag => [ "new tag1", "new tag2" ]
  }
}

```

Output location

Finally, the output location defines the location of your Solr instance.

In the `lw_solr.conf` file, the output is defined as the following:

```

output {
  stdout { debug => true codec => "rubydebug" }
  lucidworks_solr_lsv133 { collection_host => "localhost" collection_port => "8983"
collection_name => "logstash_logs" field_prefix => "event_" force_commit => false
flush_size => 100 idle_flush_time => 1 }
}

```

Line 21 of this file contains the definitions for output to a Solr instance. To modify this for your environment, you should change several properties:

- `collection_host`: the address of your Solr instance, defined as a string. If it is not defined, it defaults to 'localhost'.
- `collection_port`: the port of your Solr instance, defined as a string. If it is not defined, it defaults to '8983'.
- `collection_name`: the name of the Solr collection the log events will be indexed to.
- `field_prefix`: the prefix to use when adding the timestamp and version fields (as described in [Solr Writer for Logstash Initial Setup](#)). If it is not defined, it defaults to 'logstash_'. In this case, the `lw_solr.conf` file has defined it as 'event_'.
- `force_commit`: if true, a commit request will be sent to Solr with each batch of documents uploaded. If false, which is the default behavior, a commit will only occur when the Solr instance is configured to commit.
- `flush_size`: the number of documents to queue as a batch before writing to Solr. This uses Logstash's stud event buffering. If it is not defined, it will default to 100 events.
- `idle_flush_time`: the amount of time, in seconds, to wait from the last buffer flush before another flush is executed. This flush will be done even if the number of buffered events is less than the defined `flush_size`.

Note also that this definition includes a 'stdout' output as well. This will output the processing to the console window. With the `debug => true` option as defined there, each document will also be output to the console window.

Once the process has finished, your processed log data will be in your Solr instance and available for queries.

Launching the Logstash JAR

Once your configuration is complete, you launch the processing of your log files with the following command:

```
java -jar logstash-1.3.3-flatjar.jar agent -f lw_solr.conf -p .
```

Note that this command ends with a period ('.'). This is required for the processing to run.

SiLK Examples and Recipes

To show how SiLK can be used for different use cases, we have provided a couple of examples, described below.

Apache Weblogs

Using SiLK, it is possible to index your Apache weblogs (from access_log or other log file produced by httpd).

Logstash provides the conversion from the raw log file format to documents that can be indexed by Solr, and Banana provides the data visualization tools. The source code for this example can be found in the Lucidworks Github repository at <https://github.com/LucidWorks/silkusecases/tree/master/apacheweblogs>.

This example includes adding a location data

Indexing the Log Files

As described in the section [Indexing Content to Solr](#), when you start an indexing job with Logstash, you define a configuration file for the location of the raw data, the conversion of the data and the outputs for the data.

The sample provided in the silkusecases repository (at https://github.com/LucidWorks/silkusecases/blob/master/apacheweblogs/silk_apachelog_with_geoip) can be modified in the following ways:

Input

The input section defines where to find the data to be processed.

From the provided example, you can simply modify the "path" property to the correct location.

```
input {
  file {
    path =>
"/Users/ravikrishnamurthy/Documents/src/demo_log_generator/mysamplelogs/lucidfind/weblogs/find_searchhub_org_access*"
    exclude => ["*.gz", "*.zip", "*.tgz"]
    type => "apache-access"
    sincedb_path =>
"/Users/ravikrishnamurthy/Documents/agents/SiLK-1.1/solrWriterForLogStash/logstash_deploy/apachelogs.sincedb"
    start_position => "beginning"
  }
}
```

Filter

In the provided example, the filter is using several filters together.

grok

The grok filter is included with Logstash by default. Here we have defined the log type as "apache-access", and defined the pattern.

```
grok {  
  
  type => "apache-access"  
  patterns_dir => "./patterns"  
  # See the following URL for a complete list of named patterns  
  # logstash/grok ships with by default:  
  # https://github.com/logstash/logstash/tree/master/patterns  
  #  
  # The grok filter will use the below pattern and on successful match use  
  # any captured values as new fields in the event.  
  
  pattern => "%{COMBINEDAPACHELOG}"  
}
```

There is nothing you need to change here.

date

In this section, we are defining what dates will look like in the documents. We have defined the date type as "apache-access" and defined the date pattern and what field it will go into.

```
date {  
  type => "apache-access"  
  # Try to pull the timestamp from the 'timestamp' field (parsed above with  
  # grok). The apache time format looks like: "18/Aug/2011:05:44:34 -0700"  
  match => [ "timestamp", "dd/MMM/yyyy:HH:mm:ss Z" ]  
}
```

Unless you have configured your Apache logs to record dates in a different format, it's unlikely there is anything to change here.

geoip

```
geoip{  
  source => ["clientip"]  
}
```

mutate

This section defines how we will take the client IP data and parse it into state, country and/or region codes for the index.

```

mutate{
  #timestamp is a field used in LWS. Need to delete it here
  remove_field => [ "timestamp"]

  #flatten the geoip JSON object into Solr fields
  # two letter country code used by map module
  rename =>["[geoip][country_code2]", "country"]
  #three letter country code, not used at the moment
  rename =>["[geoip][country_code3]", "country3"]
  rename =>["[geoip][city_name]", "city"]
  rename =>["[geoip][continent_code]", "continent"]

  # location field for use in clustering and range faceting by LatLon coordinates
  add_field => {"geolocation" => "%{[geoip][latitude]},%{[geoip][longitude]}"}

}

#in map module we want to get two letter state codes for US geography only. Use
region codes elsewhere
if [country] == "US" {
  mutate{
    rename =>["[geoip][region_name]", "state"]
  }
}
else{
  mutate{
    rename =>["[geoip][region_name]", "region"]
  }
}

#now that we have flattened it there is no need for the geoip field.
mutate{
  remove_field => ["geoip"]
}

```

Output

```

output {
  stdout { debug => true codec => "rubydebug"}
  lucidworks_solr_lsv133 { collection_host => "localhost" collection_port => "8888"
collection_name => "lucidfindapachelogs" field_prefix => "event_" force_commit =>
false flush_size => 100 idle_flush_time => 1 }
}

```

Visualizing the Data

Search Analytics

Once you have built your search application with Apache Solr, the next step is to get information about how it's being used by your target audience. Using SiLK, you can ingest the log files from your application and use dashboards to analyze the data.

Ideally, you would be able to answer the following questions:

- What are people searching for?
- What queries return zero hits (or a low number of hits)?
- Are there queries that are slow to return results? Are there trends that indicate a need for more hardware or better tuning?
- What is the overall system performance? Are we experiencing downtime?

This recipe provides some approaches to answering these questions. The source code for this recipe can be found at <https://github.com/LucidWorks/silkusecases/tree/master/searchanalytics>.

Indexing Solr Log Files

As described in the section [Indexing Content to Solr](#), we can use Solr Writer for Logstash to index Solr's log files. This requires a configuration file to tell Logstash how to process the files.

The example provided in the silkusecases Github repository (found at https://github.com/LucidWorks/silkusecases/blob/master/searchanalytics/silk_solrlogs.conf) can be customized for your environment. Let's step through this file as an example:



If you are using LucidWorks Search instead of Solr, you can use https://github.com/LucidWorks/silkusecases/blob/master/searchanalytics/silk_lwslogs.conf instead.

input

```
input {
  file {
    type => "solrlog"
    path => [
      "/Users/ravikrishnamurthy/Documents/demos/slk-4.7.0/solr-4.7.0/SiLK/logs/*" ]
    #path => [
      "/Users/ravikrishnamurthy/Documents/demos/slk-4.7.0/solrWriterForLogStash/logstash_deploy/test.log" ]
    exclude => ["*.gz", "*.zip", "*.tgz"]
    sincedb_path => "/dev/null"
    start_position => "beginning"
  }
}
```

filter


```

filter {
  if [type] == "solrlog" {

    grok {
      match => ["message", "INFO %{DATA} %{TIMESTAMP_ISO8601:received_at};
%{DATA}; \[%{DATA:collection}\] webapp=%{DATA:webapp} path=%{DATA:searchhandler}
params=%{DATA}q=%{DATA:queryterms}[%&]l%{DATA} hits=%{BASE10NUM:hits}
status=%{BASE10NUM:status} QTime=%{BASE10NUM:qtime}"]
    }

    if ("_grokparsefailure" in [tags]) {
      drop{}
    }

    date {
      # Try to pull the time stamp from the 'received_at' field (parsed above with
grok)
      match => [ "received_at", "yyyy-MM-dd HH:mm:ss.SSS" ]
    }
  }
}

```

output

```

output {
  stdout { debug => true codec => "rubydebug"}
  #lucidworks_solr_lsv133 { collection_host => "localhost" collection_port => "8888"
collection_name => "solrlogs" field_prefix => "event_" force_commit => false
flush_size => 1000 idle_flush_time => 1 }
}

```

Analytics Dashboard