

This tutorial shows how to log browser traffic and analyze it in real time. This is useful when you have a significant amount of logging from various sources and you want to debug issues or generate up-to-date statistics from the logs.

The tutorial describes how to send log information generated by an [NGINX](https://www.nginx.com/) web server to BigQuery using Fluentd, and then use BigQuery to analyze the log information. It assumes that you have basic familiarity with Google Cloud, Linux command lines, application log collection, and log analysis.

Logs are a powerful tool for providing a view into how large-scale systems and applications are performing. However, as the scale and complexity of these systems increases, it becomes a challenge to manage multiple logs that are distributed across a fleet of compute resources.

[Fluentd](http://www.fluentd.org/) is a popular open source log collector that aims to unify log collection across [many data sources](http://www.fluentd.org/datasources) and systems into a unified logging layer. Because it's hard to know in advance what data might be useful to analyze, a common approach is to log it all and sort through it later. But collecting and storing all of this data can get unwieldy, making it slow and difficult to get the answers you're looking for.

This is where the strengths of [BigQuery](/bigquery/) become useful for log insights. BigQuery is Google's fully managed, highly scalable, cloud data warehouse and analytical engine. It can perform queries on terabytes of logs in tens of seconds. This performance allows you to get the answers you need quickly to fix or improve your systems.

By default, you can stream [100,000 rows of log data per second](/bigquery/quotas#streaming_inserts) into BigQuery, and you can raise this limit by [requesting additional quota](https://console.cloud.google.com/iam-admin/quotas?service=bigquery.googleapis.com&metric=Streaming%20insert%20rows%20per%20second) (https://console.cloud.google.com/iam-admin/quotas?service=bigquery.googleapis.com&metric=Streaming%20insert%20rows%20per%20second)

Fluentd has an [output plugin](https://docs.fluentd.org/output) that can use [BigQuery as a destination](https://www.fluentd.org/plugins#google-cloud-platform) for storing the collected logs. Using the plugin, you can directly load logs into BigQuery in near real time from many servers. You can then easily visualize this data by creating a dashboard that's updated frequently inside Google Sheets or Google Data Studio.

- Run an NGINX web server on a Compute Engine instance.
- Install a Fluentd log collection agent.
- Configure Fluentd to do the following:
 - Collect NGINX traffic logs.
 - Forward the logs to BigQuery.
- Query the log messages using the BigQuery web UI.
- Run an analytics query on the logs using BigQuery.

This tutorial uses billable components of Google Cloud, including:

- Compute Engine
- BigQuery

Use the [Pricing Calculator](https://cloud.google.com/products/calculator/#id=60cf7877-16e3-4b07-ae48-1aa1fa93f975) (/products/calculator/#id=60cf7877-16e3-4b07-ae48-1aa1fa93f975) to generate a cost estimate based on your projected usage.

1. [Sign in](https://accounts.google.com/Login) (https://accounts.google.com/Login) to your Google Account.

If you don't already have one, [sign up for a new account](https://accounts.google.com/SignUp) (https://accounts.google.com/SignUp).

2. In the Cloud Console, on the project selector page, select or create a Cloud project.

★ **Note:** If you don't plan to keep the resources that you create in this procedure, create a project instead of selecting an existing project. After you finish these steps, you can delete the project, removing all resources associated with the project.

[Go to the project selector page](https://console.cloud.google.com/projectselector2/home/dashboard) (https://console.cloud.google.com/projectselector2/home/dashboard)

3. Make sure that billing is enabled for your Google Cloud project. [Learn how to confirm billing is enabled for your project](https://cloud.google.com/billing/docs/how-to/modify-project) (/billing/docs/how-to/modify-project).

When you finish this tutorial, you can avoid continued billing by deleting the resources you created. See [Cleaning up \(#clean-up\)](#) for more detail.

In this tutorial, you use the [Google Cloud Marketplace](https://console.cloud.google.com/marketplace) (https://console.cloud.google.com/marketplace) to create a Compute Engine instance preconfigured to run the NGINX web server.

1. In the Cloud Console, go to the Marketplace details page for the **Nginx (Google Click to Deploy)** image:

[Go the NGINX details page](https://console.cloud.google.com/marketplace/details/click-to-deploy-images/nginx) (https://console.cloud.google.com/marketplace/details/click-to-deploy-images/nginx)

2. Click **Launch on Compute Engine**.
3. If you're prompted, select the Google Cloud project to use.
4. Wait for the required APIs to be enabled.
5. Name the deployment `nginx`.
6. Select a zone that's in either a US region or a European region.
7. Under **Firewall** in the **Networking** section, make sure that the **Allow HTTP traffic** option is selected.
8. Read the terms listed under **Terms of Service**.
9. If you accept the terms, click **Deploy**.

The deployment details screen shows a new VM being created in the zone you specified, and the NGINX web server being installed.

10. Wait for the deployment to complete before continuing.

The virtual machine needs additional permissions in order to write to BigQuery. Modifying the VM permissions requires that you first shut down the VM.

1. In the Cloud Console, go to the Compute Engine VM Instances page.

[Go to the VM Instances page](https://console.cloud.google.com/compute/instances) (https://console.cloud.google.com/compute/instances)

You see a new VM instance called `nginx-vm`.

2. To open the details page for the VM, click the VM name.

3. Click the **Stop** button at the top of the page.

It takes a few moments to shut down the VM.

4. Click **Edit** at the top of the page.

5. Scroll down and change the BigQuery access to **Enabled**.

6. Click **Save** at the bottom of the page.

7. Click **Start** at the top of the page to restart the VM with the new permissions.

Don't close this page yet; you continue working in the page in the next section.

In this section of the tutorial, you install the Fluentd log collector and the Fluentd output plugin for BigQuery on the VM.

1. In the VM instance details page, click the SSH button to open a connection to the instance.

2. In the shell window on the VM, verify the version of Debian:

The result shows the Debian version that's running on the VM and its codename:

3. Go to the [Fluentd download page for Debian](https://docs.fluentd.org/installation/install-by-deb) (<https://docs.fluentd.org/installation/install-by-deb>) and find the installation command line for the Debian codename version.

For example, for Debian Stretch, you find the following command:

4. Copy the command into the shell window on your VM.

This command installs the [td-agent](https://docs.treasuredata.com/articles/td-agent) package on your VM. The package contains the Fluentd distribution.

5. Install the Fluentd-to-BigQuery plugin:

Fluentd works by [using input plugins](https://docs.fluentd.org/input) to collect logs generated by other applications and services. It parses this data into structured JSON records, which are then forwarded to any configured output plugins.

The [Fluentd NGINX access log parser](https://docs.fluentd.org/parser/nginx) reads the NGINX `access.log` files. The following listing shows an example record with fields and example values.

When you create a log destination table in BigQuery, the column names need to match the field names in the log record exactly. In the following procedure, make sure you use the names that are suggested so that the table column names are correct for later.

1. Open the BigQuery web UI:

[Go to the BigQuery Web UI](https://console.cloud.google.com/bigquery) (https://console.cloud.google.com/bigquery)

2. In the navigation panel, under **Resources**, click your project name.

3. In the details panel below the Query Editor, click **Create Dataset**.

4. For **Dataset ID**, enter `fluentd`.
5. For **Data Location**, select the multiregional location (US or Europe) where you created the NGINX instance.
6. Click **Create Dataset**.
7. In the navigation panel, under **Resources**, click the **fluentd** dataset.
8. In the details panel below the Query Editor, click **Create Table**.
9. For the table name, enter `nginx_access`.
10. Select the **Edit as Text** option.
11. Copy the following JSON column definition into the text box.

Notice that the column names exactly match the field names in the Fluentd log record. The `time` column value is added to the record by converting the log record timestamp into a BigQuery-compatible timestamp string.

12. In the **Partitioning** list, select **Partition by field: time**.

Partitioned tables divide large tables into smaller segments to improve query performance and control costs by reducing the amount of data that is read by a query. For more information, see [Introduction to partitioned tables \(/bigquery/docs/partitioned-tables\)](/bigquery/docs/partitioned-tables) in the BigQuery documentation.

13. Click **Create table** at the bottom.

The BigQuery table named `fluentd.nginx_access` is now available to receive log records.

14. To view table details, under **Resources > Details**, click the table name.

The details allow you to see information such as how many rows there are, and how much storage the table uses.

The Fluentd configuration file `/etc/td-agent/td-agent.conf` defines the sources from which log data is collected, the outputs for the collected logs, and any filters. It also allows you to apply tags to the collected logs to define how they are processed and which output plugin they should be sent to.

In the following procedure, you configure Fluentd to do the following:

- Use the [tail input plugin](https://docs.fluentd.org/input/tail) (<https://docs.fluentd.org/input/tail>) to collect the NGINX logs as they are generated.
- Parse the log lines with the [NGINX log parser](https://docs.fluentd.org/parser/nginx) (<https://docs.fluentd.org/parser/nginx>).
- Send them to the [BigQuery output plugin](https://github.com/kaizenplatform/fluent-plugin-bigquery#configuration) (<https://github.com/kaizenplatform/fluent-plugin-bigquery#configuration>), which will insert them into the table you created.

To set up log collection and forwarding:

1. Use SSH to connect to the VM where NGINX is installed.
2. In the shell window, open a text editor as a root user (for example, open Vim or Nano) and edit the Fluentd agent configuration file `/etc/td-agent/td-agent.conf`. For example, use the following command to open the file in Vim:

3. At the bottom of the file, append the following lines to configure the tail input plugin to read the NGINX logs, parse them with the NGINX parser, and tag them with `nginx.access`:

4. Append the following lines to configure the BigQuery output plugin. Replace `[MY_PROJECT_NAME]` with the name of your Google Cloud project.

This configures Fluentd to do the following:

- For every log entry with tag `nginx.access`, Fluentd should use the `bigquery_insert` plugin to write the record to the BigQuery table.
- Authentication is done using the VM's service account.
- The original timestamp is added to the record as a string formatted in a way that BigQuery can convert to a timestamp.

5. Save the file and quit the editor.

6. Restart the Fluentd agent to apply the new configuration.

For more information on these configuration file parameters, including alternative methods for authentication, see the [BigQuery Fluentd plugin site](https://github.com/kaizenplatform/fluent-plugin-bigquery#configuration)

(<https://github.com/kaizenplatform/fluent-plugin-bigquery#configuration>) and the [Fluentd plugin documentation](https://docs.fluentd.org/parser) (<https://docs.fluentd.org/parser>).

Now that you have configured Fluentd, you can generate some NGINX log data and view it using BigQuery.

1. In the Cloud Console, go to the Compute Engine VM Instances page:

[Go to the VM instances page \(https://console.cloud.google.com/compute/instances\)](https://console.cloud.google.com/compute/instances)

2. Copy the external IP address of the `nginx-vm` VM instance.
3. In a separate browser tab, paste the IP address into the address box.

You see the default **Welcome to nginx!** page.

4. In the Cloud Console, go to the [BigQuery page \(https://console.cloud.google.com/bigquery\)](https://console.cloud.google.com/bigquery).
5. Copy the following query into the **Query Editor** pane and click **Run**:

You see a row in the **Query Results** panel that shows the access log record from your browser. Because your VM is accessible from the internet, you might also see access log rows from other remote hosts.

★ **Note:** If you are moving through this tutorial quickly, the first query results might not yet be available. If the results are empty, retry the query again in a few seconds. For more information on when streamed data becomes available, see the [Checking for data availability \(/bigquery/streaming-data-into-bigquery#dataavailability\)](#) section of the BigQuery documentation.

In this section, you run sample loads and then view the metrics for those loads using BigQuery. Performing these steps shows you that BigQuery can be used not only to read the logs, but to run analysis on them.

1. In the Cloud Console, start Cloud Shell.
2. Install the [ApacheBench \(https://httpd.apache.org/docs/2.4/programs/ab.html\)](https://httpd.apache.org/docs/2.4/programs/ab.html) (**ab**) web server benchmarking tool and related tools:
3. In Cloud Shell, generate a test load to run against the NGINX server. Replace `[IP_ADDRESS]` with the IP address of your VM.

This command uses the ApacheBench tool to generate 20 seconds of load against your NGINX server.

4. In the Cloud Console, go to the BigQuery page:

[Go to the BigQuery page](https://console.cloud.google.com/bigquery) (<https://console.cloud.google.com/bigquery>)

5. Get a list of ApacheBench requests using the following query in the **Query Editor** pane:

6. Run the following SQL command to calculate the number of requests for each response code for each second:

The values for each second should be approximately equal to the **Requests per second** line in the ApacheBench output that you saw earlier.

As mentioned, BigQuery streaming inserts (as used in this tutorial) make the data available to be queried within a few seconds. There is a [small charge](#) ([/bigquery/pricing#streaming_pricing](#)) for this method, and there are [limits applied](#) ([/bigquery/quotas#streaming_inserts](#)) to the size and frequency of the inserts.

For higher logging volumes, you might want to use BigQuery batch loading. As the name suggests, this loads the data using a lower-priority batch job. With batch loading, it takes more time for the data to load and become available than with streaming inserts. However, there is no charge for batch loading.

As with streaming inserts, there are limits (/bigquery/quotas#load_jobs) to the frequency of batch load jobs—most importantly, 1,000 load jobs per table per day, and 50,000 load jobs per project per day.

To implement batch loading, you use the bigquery_load (<https://github.com/kaizenplatform/fluent-plugin-bigquery#load>) Fluentd plugin. This plugin uses a Fluentd buffer (<https://docs.fluentd.org/buffer>) to collect a set of logs in files up to a specified limit on time or size before sending them as a batch to BigQuery.

To use batch loading, do the following:

1. Edit the Fluentd configuration file `/etc/td-agent/td-agent.conf` as a root user.
2. Replace the BigQuery output plugin configuration you specified earlier with the following lines. Replace `[MY_PROJECT_NAME]` with the name of your Google Cloud project.

This configuration specifies that the bigquery_load output plugin should be used instead of the bigquery_insert plugin.

The buffer section specifies that the log data is buffered in the VM until up to 1 GB of logs have been collected or up to one hour has passed. The buffered data is then loaded into BigQuery. For

more information on the configuration parameters, see the [plugin documentation](https://github.com/kaizenplatform/fluent-plugin-bigquery#configuration) (<https://github.com/kaizenplatform/fluent-plugin-bigquery#configuration>).

If at any point you suspect that the agent is not collecting or delivering logs, you can check its state by running the following command in the shell window of the Fluentd VM:

The output shows the active state of the service, and the last few log lines from the service:

If the agent doesn't start, the most likely cause is an error in the Fluentd configuration file. This is reported in the Fluentd log file, which you can view by using the following command in the VM shell window:

For example, the following output shows an error when the name of a plugin is not correct:

To avoid incurring charges to your Google Cloud Platform account for the resources used in this tutorial:



Caution: Deleting a project has the following effects:


- **Everything in the project is deleted.** If you used an existing project for this tutorial, when you delete it, you also delete any other work you've done in the project.
- **Custom project IDs are lost.** When you created this project, you might have created a custom project ID that you want to use in the future. To preserve the URLs that use the project ID, such as an **appspot.com** URL,

delete selected resources inside the project instead of deleting the whole project.

If you plan to explore multiple tutorials and quickstarts, reusing projects can help you avoid exceeding project quota limits.

1. In the Cloud Console, go to the **Manage resources** page.

[Go to the Manage resources page](https://console.cloud.google.com/iam-admin/projects) (https://console.cloud.google.com/iam-admin/projects)

2. In the project list, select the project you want to delete and click **Delete**  .

3. In the dialog, type the project ID, and then click **Shut down** to delete the project.

- Learn about using [Stackdriver Logging](/logging/) (/logging/), the Google Kubernetes Engine service for managed log collection, storage, and analysis.
- Learn how to [customize Kubernetes Engine Stackdriver logs with Fluentd](/solutions/customizing-stackdriver-logs-fluentd) (/solutions/customizing-stackdriver-logs-fluentd).
- Learn how to [visualize BigQuery data in a Jupyter notebook](/bigquery/docs/visualize-jupyter) (/bigquery/docs/visualize-jupyter).
- Learn how to [process logs at scale using Dataflow](/solutions/processing-logs-at-scale-using-dataflow) (/solutions/processing-logs-at-scale-using-dataflow).
- Try out other Google Cloud features for yourself. Have a look at our [tutorials](/docs/tutorials) (/docs/tutorials).