

**Download and extract**

To deploy on top of Kubernetes we are going to grab the Fluent Bit Helm charts from the open-source repository <https://github.com/fluent/helm-charts>.

**“wget https://github.com/fluent/helm-charts/releases/download/fluent-bit-0.20.9/fluent-bit-0.20.9.tgz**

**tar -xzvf fluent-bit-0.20.9.tgz**

**cd fluent-bit”**

**Modify the Values File**

We will modify the [values.yaml](https://github.com/fluent/helm-charts/blob/757f8e26184c6bb886950dae7bcda6e2a74a5526/charts/fluent-bit/values.yaml#L309) file, in the root of the fluent-bit folder, in two key places. These changes will be used to formulate our Fluent Bit configuration. The Helm chart itself will deploy a [daemonset](https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/) on each node responsible for log collection.

Under the filters key we are going to add the same [nest filter](https://docs.fluentbit.io/manual/pipeline/filters/nest) as our earlier post to move all fields under the log column.

While the JSON type is great for the dynamic parts of the logs, we cannot currently use the columns it creates for primary keys. [Primary keys](https://clickhouse.com/docs/en/guides/improving-query-performance/sparse-primary-indexes/sparse-primary-indexes-intro) are a key component of accelerating query performance in ClickHouse and should broadly match those columns on which we are most likely to apply filters. For purposes of example, we will later create our table with a primary key on the host, pod\_name and timestamp fields - filtering logs by the host, pod and timestamp seems like a reasonably common usage pattern for Kubernetes logs as a first pass. Obviously, this may vary depending on your use case and typical diagnosis paths, and we encourage users to read about optimizing and configuring primary keys [here](https://clickhouse.com/docs/en/guides/improving-query-performance/sparse-primary-indexes/sparse-primary-indexes-design). To use these columns as primary keys we must therefore move them out of the JSON log field to the base of the message. To achieve this we use a [lua filter](https://docs.fluentbit.io/manual/pipeline/filters/lua). This is currently necessary as the lift feature of the nest filter is not selective and the [modify filter](https://docs.fluentbit.io/manual/pipeline/filters/modify) is unfortunately [not supported on nested fields](https://github.com/fluent/fluent-bit/issues/2152#issuecomment-1049615508). To achieve this, we set the luaScripts key in our [values.yaml](https://github.com/fluent/helm-charts/blob/757f8e26184c6bb886950dae7bcda6e2a74a5526/charts/fluent-bit/values.yaml#L276) file:

|  |
| --- |
| luaScripts:  functions.lua: |  function set\_fields(tag, timestamp, record)  if record['log'] and record['log']['kubernetes'] then  record['host'] = record['log']['kubernetes']['host']  record['pod\_name'] = record['log']['kubernetes']['pod\_name']  record['labels'] = record['log']['kubernetes']['labels']  record['annotations'] = record['log']['kubernetes']['annotations']  record['stream'] = record['log']['kubernetes']['stream']  record['log']['kubernetes'] = nil  return 2, timestamp, record  else  return 0, timestamp, record -- Return 0 if 'kubernetes' field is not present  end  end |

Then make the following changes in the filters section, thus Our filters configuration becomes:

## https://docs.fluentbit.io/manual/pipeline/filters

|  |
| --- |
| filters: |  [FILTER]  Name kubernetes  Match kube.\*  Merge\_Log On  Keep\_Log Off  K8S-Logging.Parser On  K8S-Logging.Exclude On  [FILTER]  Name nest  Match \*  Operation nest  Wildcard \*  Nest\_under log  [FILTER]  Name lua  Match \*  script /fluent-bit/scripts/functions.lua  call set\_fields |

In the [Output section](https://github.com/fluent/helm-charts/blob/757f8e26184c6bb886950dae7bcda6e2a74a5526/charts/fluent-bit/values.yaml#L319) we are going to replace the default Elasticsearch configuration with the ClickHouse HTTP output. Be sure to replace the host, port and http\_passwd parameters with your [ClickHouse Cloud](https://clickhouse.cloud/signUp?glxid=faa2caf4-8750-4e94-b161-a828da5e3384&experiments=mktg-website-nav-cta-btn%3A0) settings. As a reminder, users can access the HTTP settings from the connection settings of a [ClickHouse Cloud](https://clickhouse.cloud/signUp?glxid=faa2caf4-8750-4e94-b161-a828da5e3384&experiments=mktg-website-nav-cta-btn%3A0) service.

Note: We’ve used a separate table kube for the data vs. the original jsonlogs table used in our earlier post. We create this below.

## https://docs.fluentbit.io/manual/pipeline/outputs

|  |
| --- |
| outputs: |  [OUTPUT]  name http  tls off  match \*  host <YOUR CLICKHOUSE CLOUD HOST>  port 8123  URI /?query=INSERT+INTO+fluentbit.kube+FORMAT+JSONEachRow  format json\_stream  json\_date\_key timestamp  json\_date\_format epoch  http\_user default  http\_passwd <YOUR PASSWORD> |

**Creating the table**

In preparation for the logs, we need to create the table in ClickHouse.

**“CREATE DATABASE fluentbit”**

After creating the database, we are required to enable the JSON object type via the experimental flag allow\_experimental\_object\_type, or in [ClickHouse Cloud](https://clickhouse.cloud/signUp?glxid=faa2caf4-8750-4e94-b161-a828da5e3384&experiments=mktg-website-nav-cta-btn%3A0) opening a support case:

**“SET allow\_experimental\_object\_type = 1”**

Once set, we can create the table with the provided structure. Note how we specify our primary key via the ORDER BY clause. Explicitly declaring our host and pod\_name columns on the root of the message, rather than relying on ClickHouse to infer them dynamically as simply String within the JSON column, allows us to define their types more tightly - for both we use LowCardinality(String) improving their compression and query performance due to reduced IO. We create the usual log column which will contain any other fields in the message.

**CREATE TABLE fluentbit.kube**

**(**

**timestamp DateTime,**

**log JSON,**

**host LowCardinality(String),**

**pod\_name LowCardinality(String),**

**labels LowCardinality(String),**

**annotations LowCardinality(String),**

**)**

**Engine = MergeTree ORDER BY tuple(host, pod\_name, timestamp)**

Once created, we can deploy Fluent Bit to send our Kubernetes logs.

**Applying the Helm Chart**

We can now deploy the helm chart using the following command in the fluent-bit directory:

“**Helm install my\_release\_name .”**

Then the fluentbit is deployed and collects the logs and sends it to the clickhouse server.

You can check the logs aggregating in clickhouse by running the following command in clickhouse server

**“SELECT count()**

**FROM fluentbit.kube”**

A screenshot of a computer program

Description automatically generated And it will shows the result as follows, where the count of logs changed everytime when the command is executed showing the collection of logs in the database.

For more details refer the following link: <https://clickhouse.com/blog/kubernetes-logs-to-clickhouse-fluent-bit>