orange™ Cyberdefense

Splunk Open Telemetry Collector for Kubernetes.

Step 1: **Prepare the enviroment**

Step 2: **Install Kubectel, Kubernetes and HELM**

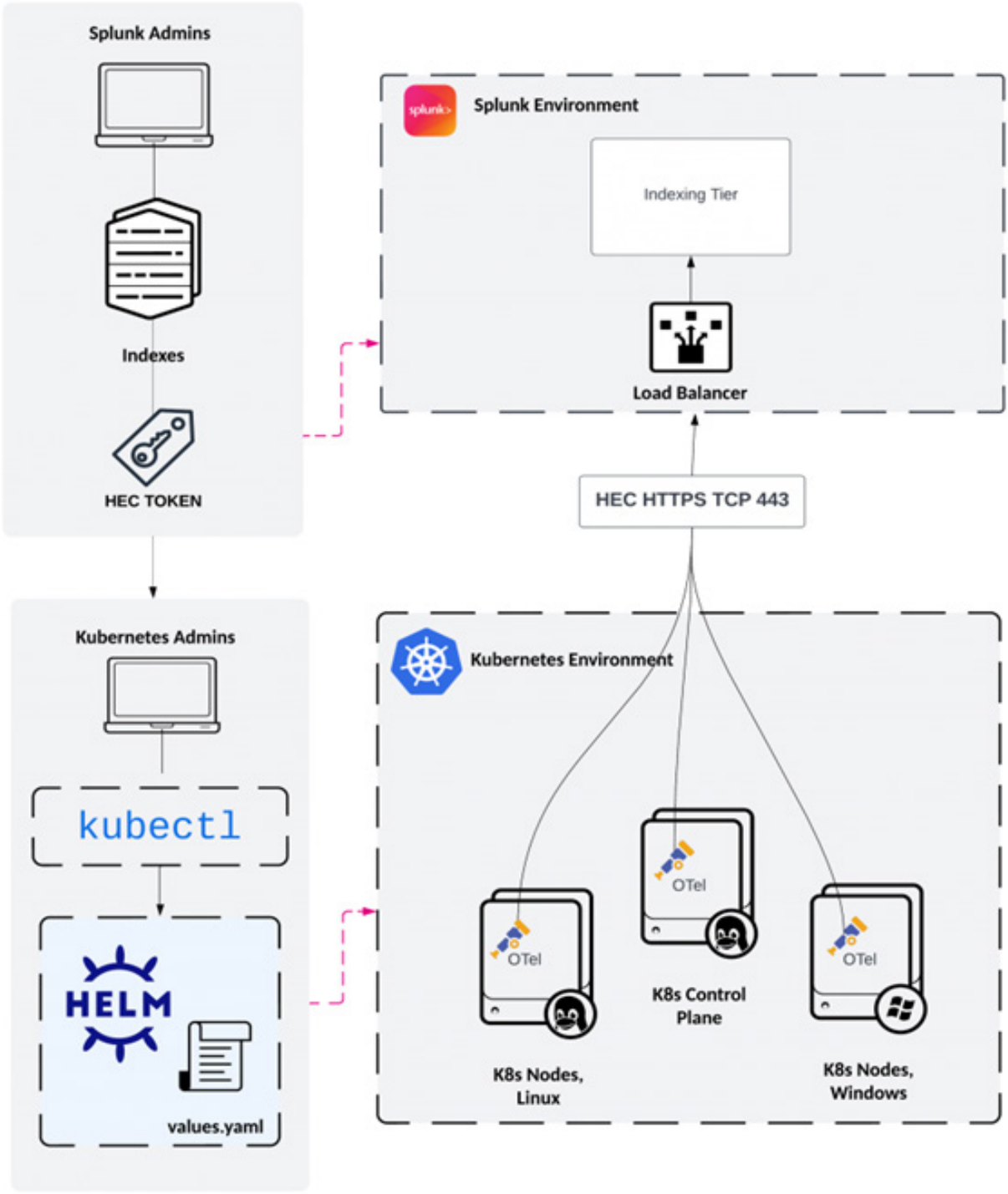
Step 3: **Install Otel Collector Helm Chart**

1. Pull OTel helm repo locally, by running the following script
2. Open the values file
3. Check your pods for errors, Get running pods
4. View cluster-receiver pod logs with the following command
5. Check Splunk

splunk>®

OpenTelemetry Collector for Kubernetes.

This Splunk validated architecture (SVA) applies to Splunk Cloud Platform and Splunk Enterprise products.



OpenTelemetry Collector incorporates many additional benefits, including:

- Built on OpenTelemetry open standards.
- Improved logging scale with OTel logging versus Fluentd. See https://github.com/signalfx/splunk-otel-collector-chart/blob/main/docs/advanced-configuration.md#performance-of-native-opentelemetry-logs-collection
- Advanced metrics collection features.
- Advanced pipeline features including data manipulation.
- Support for trace collection.
- Support for Kubernetes annotations provides ability to route namespace and pod logs to certain indexes, set sourcetypes or even include or exclude logs from being monitored. See https://github.com/signalfx/splunk-otel-collector-chart/blob/main/docs/advanced-configuration.md#managing-log-ingestion-by-using-annotations
- Support for multiline logs via the filelog receiver's recombine operator. Users can define their line breaking rules in the collector to ensure multiline logs are properly rendered in Splunk. See : https://github.com/signalfx/splunk-otel-collector-chart/blob/main/docs/advanced-configuration.md#processing-multi-line-logs
- Access to OpenTelemetry components. See all components supported in the Splunk Helm chart at https://github.com/signalfx/splunk-otel-collector/blob/main/docs/components.md#components

To deploy the OpenTelemetry Collector, some configuration is required by both Splunk admins and Kubernetes admins.

The following sections describe a simple process of getting started with the Splunk OpenTelemetry Collector for Kubernetes (without using the load balancer). Of course a high-level process can be configured depending on the customer's environment and aims in terms of the configuration and use

Step 1: Prepare the enviroment.

1. First you are going to create 1 **events** index called otel_events and 1 **metrics** index calles otel_metrics

NOTE: YOU ARE BEING ASKED TO CREATE ONE EVENTS INDEX AND ONE METRICS INDEX! PLEASE CONFIGURE THE INDEX DATA TYPE ACCURATELY!!

See below the figures for more instructions on

2. To create the Http Event Collector, go to settings and Data Input, Select HEC Collector, then create a new **HEC Token**

Step 2: Install Kubectl, Kubernetes and HELM

Along with this document, you have been provided with a series of scripts to make your life easier when it comes to deploying all of the technologies that are needed for the lab.

1. **Do the following steps locally:**
    - Download and unzip the files in your local machine and desired directory.
    - Give execution permission to the following files:
      `chmod +x microk8s-deploy1.sh microk8s-deploy2.sh otel-deploy.sh`
    - Move the files to the VM where you will be deploying K8S, Kubernetes and HELM. (I do it with scp, but you will know better how to bring them in for your environment).
      `scp microk8s-values.yaml microk8s-deploy1.sh microk8s-deploy2.sh otel-deploy.sh<Insert User Here>@<insert IP here>:/home/splunker`

2. **Now, in the VM where you will be deploying the tools, (and counting that the scripts are already there):** `/microk8s-deploy1.sh`
3. **Then add microk8s to a new group:** `newgrp microk8s`
4. **Run the second deployment script:** `./microk8s-deploy2.sh`
5. **Sourcing files:** `source ~/.bash_aliases`

Step 3: Install Otel Collector Helm Chart

1. **Pull OTelm helm repo locally by running the following script:**

   `./otel-deploy.sh`

   This script will run the following commands on your behalf:

   ```
   microk8s helm3 repo add splunk-otel-collector-chart
   https://signalfx.
   github.io/splunk-otel-collector-chart
   microk8s kubectl create ns otel
   microk8s helm3 -n otel install my-splunk-otel-collector -f
   microk8s-values.yaml splunk-otel-collector-chart/splunk-otel-collector
   ```

2. **Open the values file yaml:**

```
nano values.yaml
```

Take a minute to look through the different sections contained in the default **values.yaml file.**

Note that the yaml file is sensitive to indentation and may cause parsing issues when deploying the helm chart in the next step.

Find and Update the following values with the created HEC token in Step 1-2 :

```
clusterName: „<yourDesiredName>“
endpoint: „https://<yourSplunkInstance>:8088/services/collector“
token: „00000000-0000-0000-0000-000000000000“ #Your HEC token goes here
index: „otel_events“
metricsIndex: „otel_metrics“
insecureSkipVerify: true
metricsEnabled: true
logsEngine: otel
containerRuntime: „containerd“
excludeAgentLogs: false
```

When complete, save and quit.

3. **Check your pods for errors, Get running pods**

```
kubectl -n otel get pods
```

Output should show that both your “agent” and “cluster-receiver” are both running. Be patient, it may take a minute.

```
splunker@so1:~$ kubectl -n otel get pods
NAME                                                        READY   STATUS    RESTARTS      AGE
mmodestino-tko-otel-splunk-otel-collector-k8s-cluster-rece9h94v   1/1     Running   1 (13h ago)   24h
mmodestino-tko-otel-splunk-otel-collector-agent-ptvmt       1/1     Running   1 (13h ago)   19h
```

You can also use the “watch flag” to keep an eye on the pods as they do their thing…

```
kubectl -n otel get pods -w
```

Press “ctrl + c” to stop the watch.

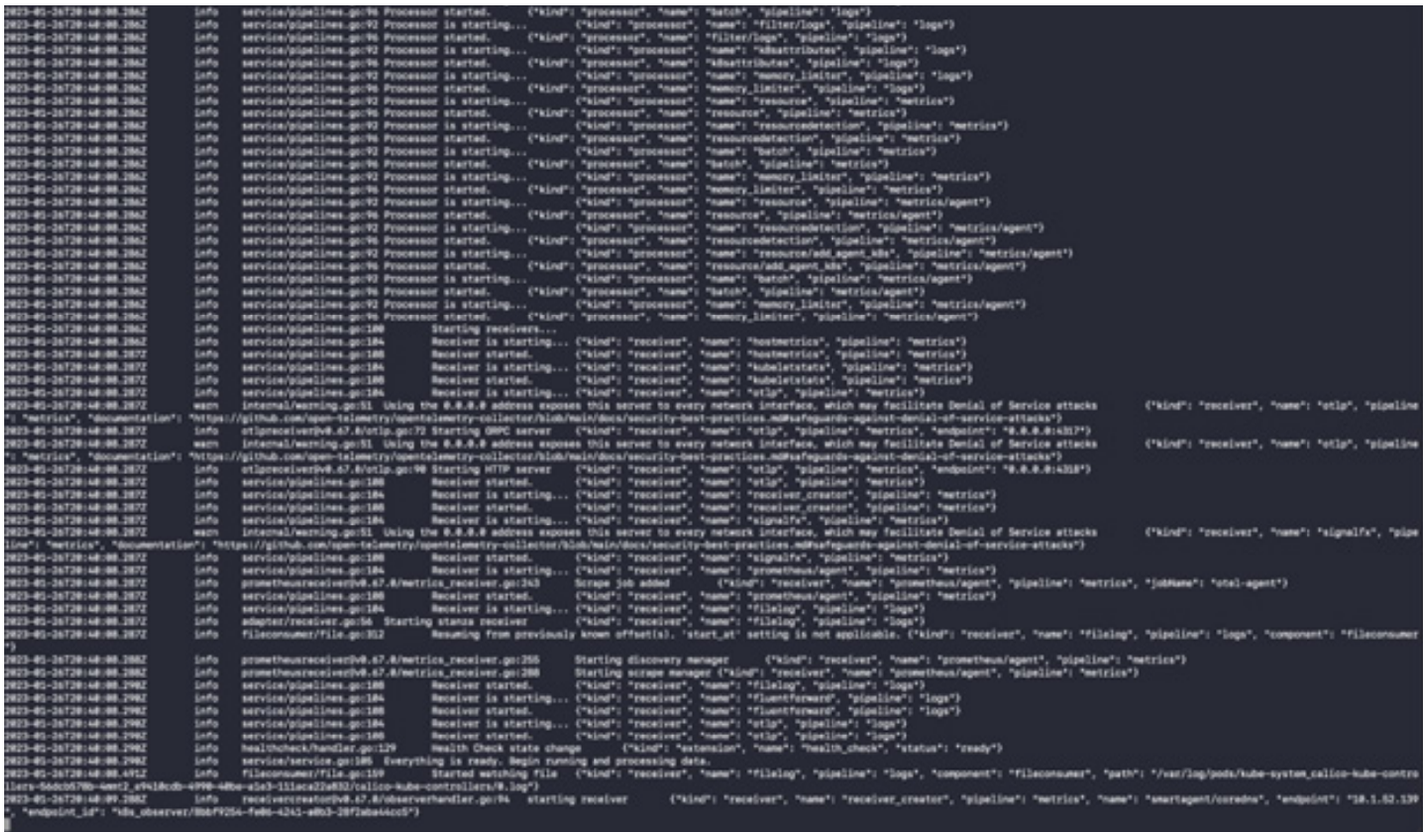
```
ctrl + c
```

Now, Check both your pods for errors.

```
kubectl -n otel logs -f <collector-agent-pod>
```

Replace <collector-agent-pod> with the agent pod name from previous command.

You should see startup logs showing the collector is running correctly. When running smoothly the pod will not log much info. It should look something like this:



Press “ctrl+c” to exit followtail view

4. **View cluster-receiver pod logs with the following command**

```
kubectl -n otel logs -f <collector-cluster-reciever-pod>
```

press “ctrl” + “c” to exit followtail view

If there is an issue with the configuration, you will see a constant stream of warnings or errors. You will need to go back and fix your configuration then use the following command to deploy your changes:

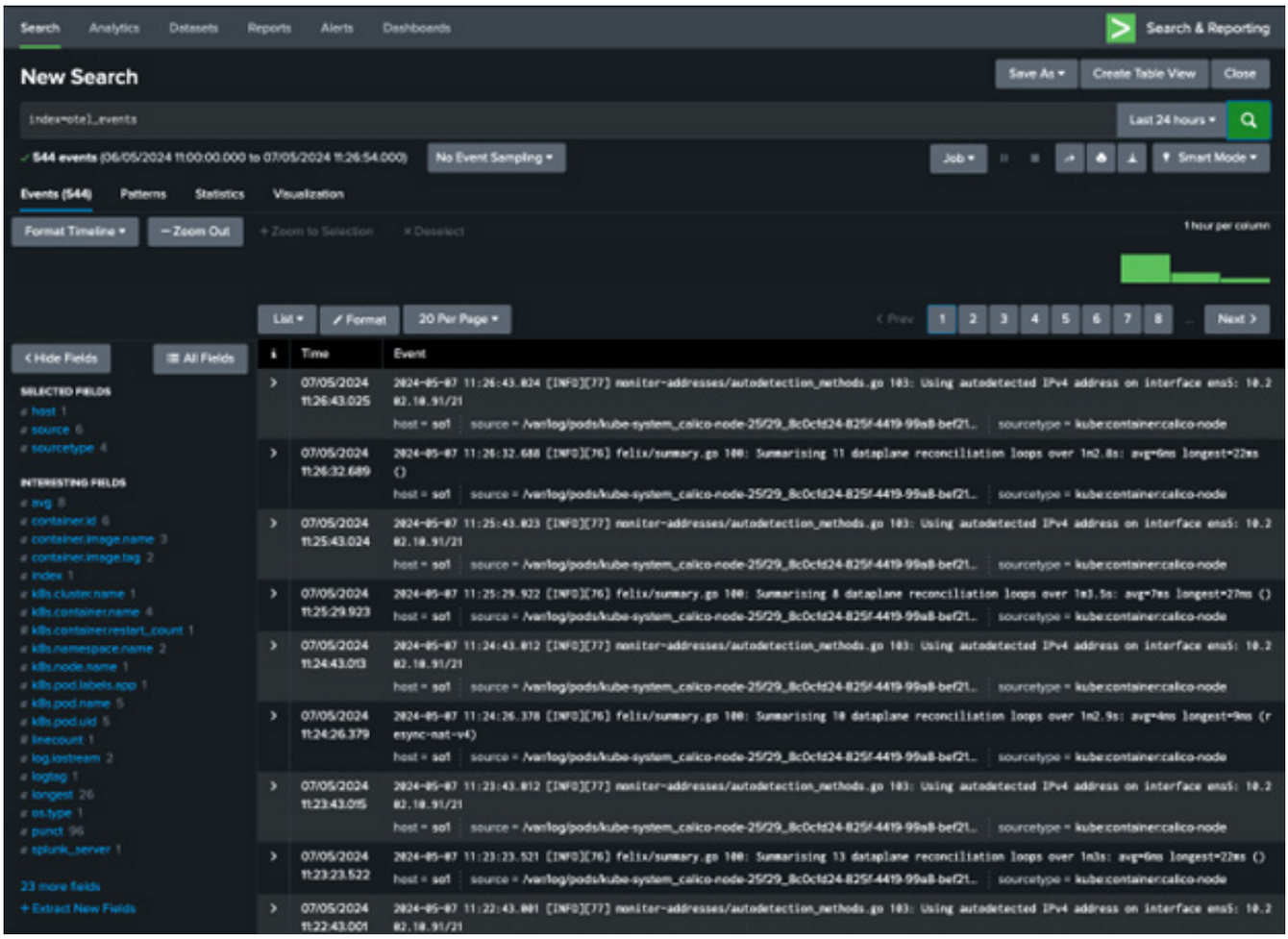
```
helm -n otel upgrade <yourDesiredName> -f values.yaml splunk-otel-collector-chart/splunk-otel-collector
```

Once you deploy your config updates, return to the previous step to check your pods are running and there are no errors in your pod logs.

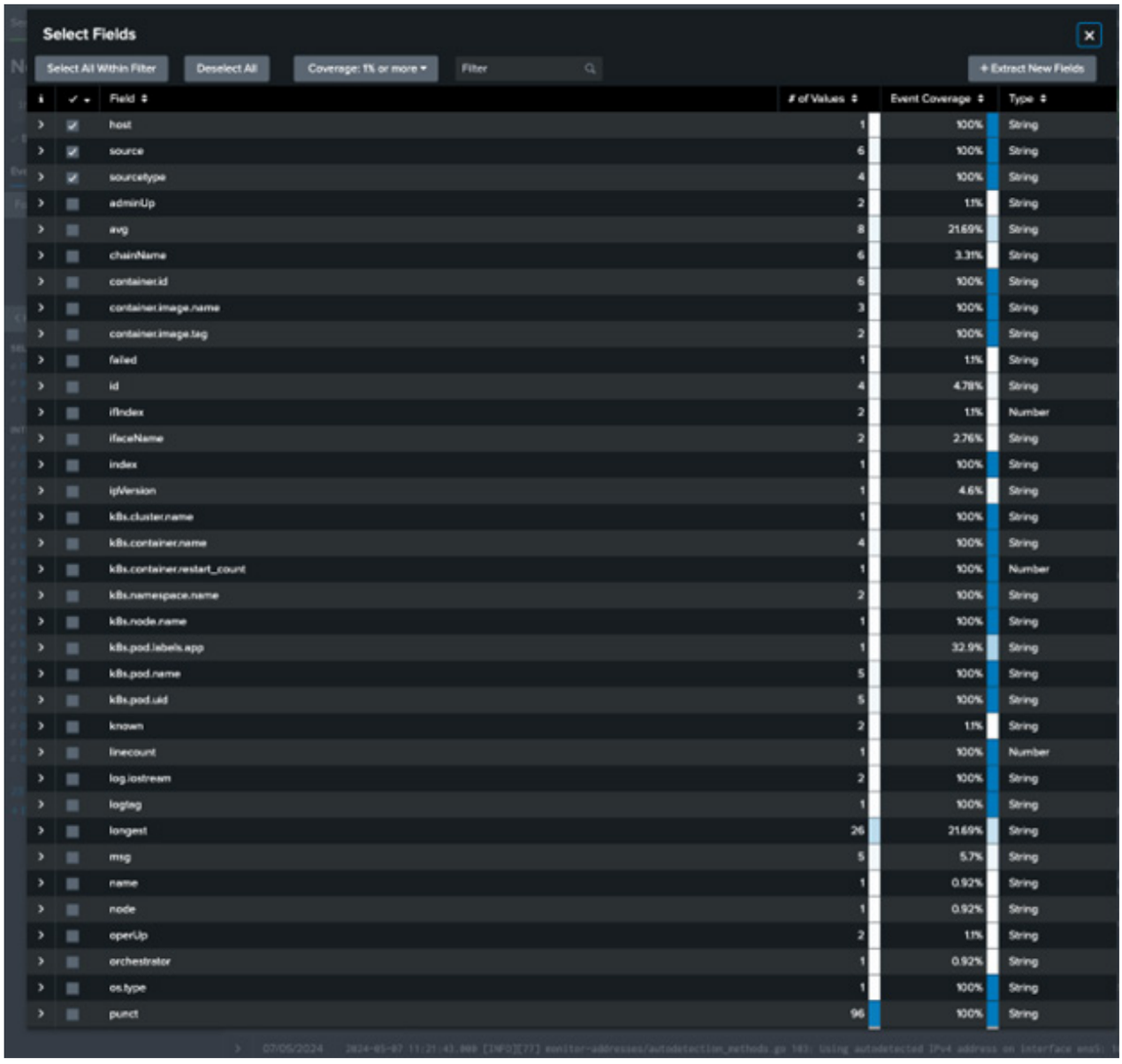
5. Check Splunk

Navigate back to Splunk Web and the Search & Reporting app. Use the default search time range to ensure you are seeing events.

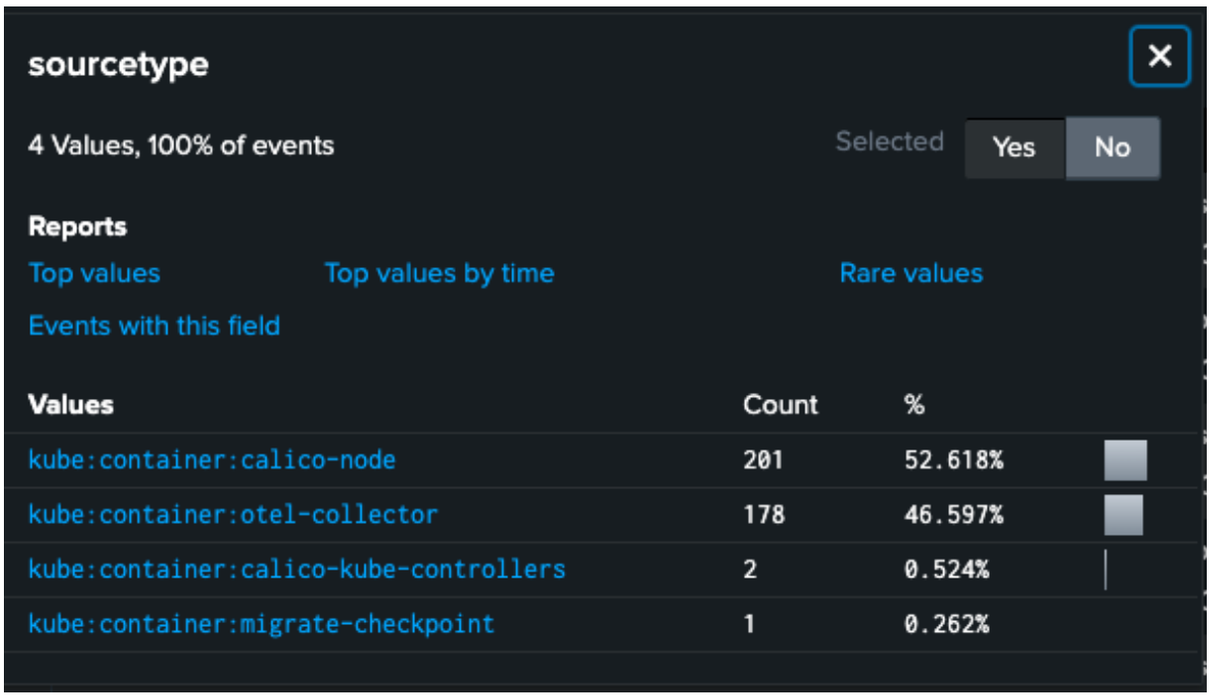
```
index=otel_events
```



If you click on more fields, you'll see:

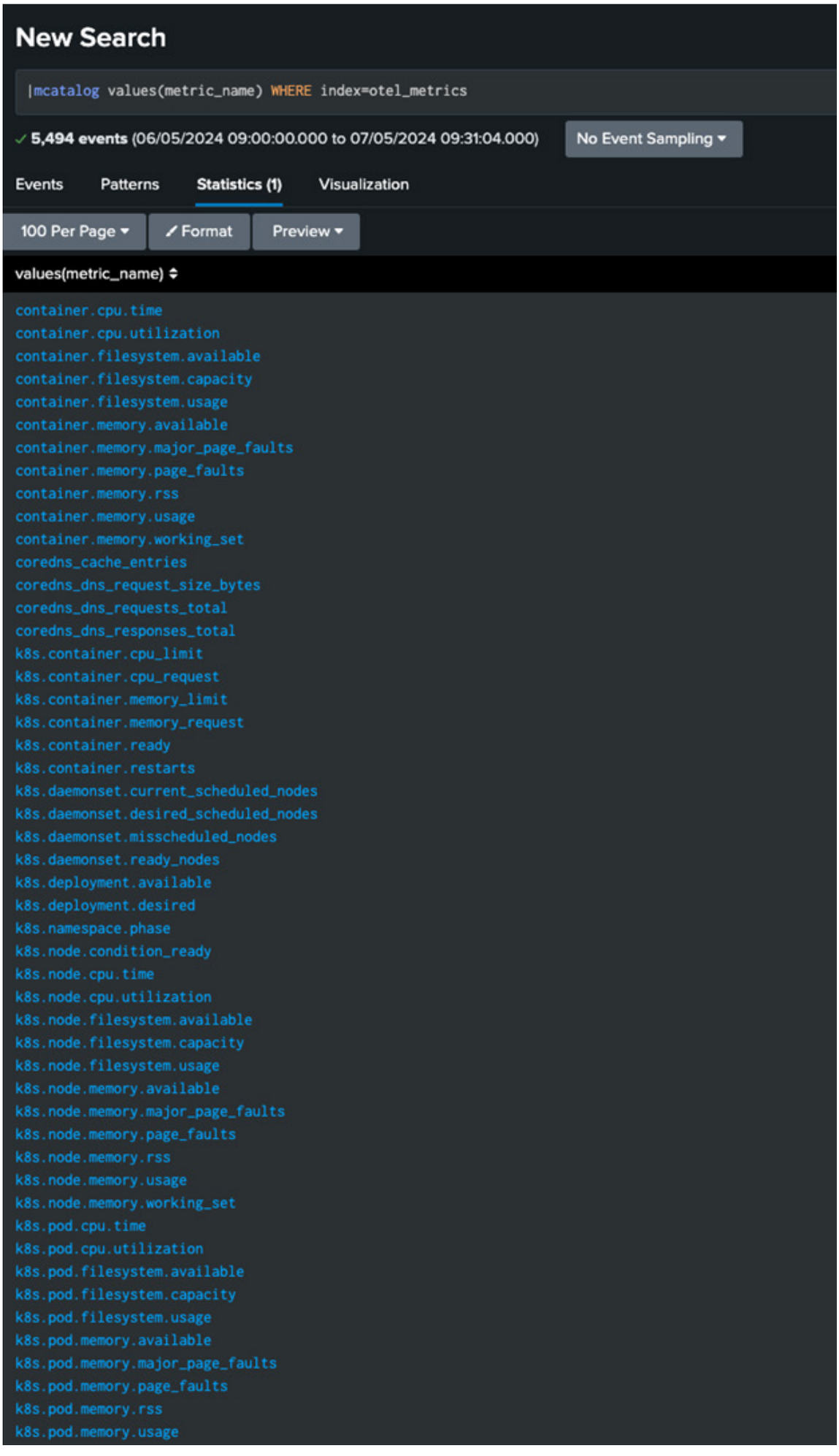


The sourcetypes are:



The type of metrics that can be displayed using the query:
`mcatalog values(metric_name) WHERE index=otel_metrics`

```
system.paging.faults
system.paging.operations
system.processes.count
system.processes.created
up
```


New Search
|mcatalog values(metric_name) WHERE index=otel_metrics
5,494 events (06/05/2024 09:00:00.000 to 07/05/2024 09:31:04.000)
No Event Sampling
Events
Patterns
Statistics (1)
Visualization
100 Per Page
Format
Preview
values(metric_name)
container.cpu.time
container.cpu.utilization
container.filesystem.available
container.filesystem.capacity
container.filesystem.usage
container.memory.available
container.memory.major_page_faults
container.memory.page_faults
container.memory.rss
container.memory.usage
container.memory.working_set
coredns_cache_entries
coredns_dns_request_size_bytes
coredns_dns_requests_total
coredns_dns_responses_total
k8s.container.cpu_limit
k8s.container.cpu_request
k8s.container.memory_limit
k8s.container.memory_request
k8s.container.ready
k8s.container.restarts
k8s.daemonset.current_scheduled_nodes
k8s.daemonset.desired_scheduled_nodes
k8s.daemonset.misscheduled_nodes
k8s.daemonset.ready_nodes
k8s.deployment.available
k8s.deployment.desired
k8s.namespace.phase
k8s.node.condition_ready
k8s.node.cpu.time
k8s.node.cpu.utilization
k8s.node.filesystem.available
k8s.node.filesystem.capacity
k8s.node.filesystem.usage
k8s.node.memory.available
k8s.node.memory.major_page_faults
k8s.node.memory.page_faults
k8s.node.memory.rss
k8s.node.memory.usage
k8s.node.memory.working_set
k8s.pod.cpu.time
k8s.pod.cpu.utilization
k8s.pod.filesystem.available
k8s.pod.filesystem.capacity
k8s.pod.filesystem.usage
k8s.pod.memory.available
k8s.pod.memory.major_page_faults
k8s.pod.memory.page_faults
k8s.pod.memory.rss
k8s.pod.memory.usage

```
k8s.pod.network.errors
k8s.pod.network.io
k8s.pod.phase
k8s.replicaset.available
k8s.replicaset.desired
otelcol_exporter_queue_capacity
otelcol_exporter_queue_size
otelcol_exporter_send_failed_log_records
otelcol_exporter_send_failed_metric_points
otelcol_exporter_sent_log_records
otelcol_exporter_sent_metric_points
otelcol_otelsvc_k8s_ip_lookup_miss
otelcol_otelsvc_k8s_namespace_added
otelcol_otelsvc_k8s_namespace_updated
otelcol_otelsvc_k8s_pod_added
otelcol_otelsvc_k8s_pod_deleted
otelcol_otelsvc_k8s_pod_table_size
otelcol_otelsvc_k8s_pod_updated
otelcol_process_cpu_seconds
otelcol_process_memory_rss
otelcol_process_runtime_heap_alloc_bytes
otelcol_process_runtime_total_alloc_bytes
otelcol_process_runtime_total_sys_memory_bytes
otelcol_process_uptime
otelcol_processor_accepted_log_records
otelcol_processor_accepted_metric_points
otelcol_processor_dropped_log_records
otelcol_processor_dropped_metric_points
otelcol_processor_filter_logs_filtered
otelcol_processor_refused_log_records
otelcol_processor_refused_metric_points
otelcol_receiver_accepted_log_records
otelcol_receiver_accepted_metric_points
otelcol_receiver_refused_log_records
otelcol_receiver_refused_metric_points
otelcol_scraper_errored_metric_points
otelcol_scraper_scraped_metric_points
scrape_duration_seconds
scrape_samples_post_metric_relabeling
scrape_samples_scraped
scrape_series_added
system.cpu.load_average.15m
system.cpu.load_average.1m
system.cpu.load_average.5m
system.cpu.time
```

Navigate to the Analytics tab and confirm you see metrics, you can check there the different dashboards, set analysis filters, and find the data that you want to analyze