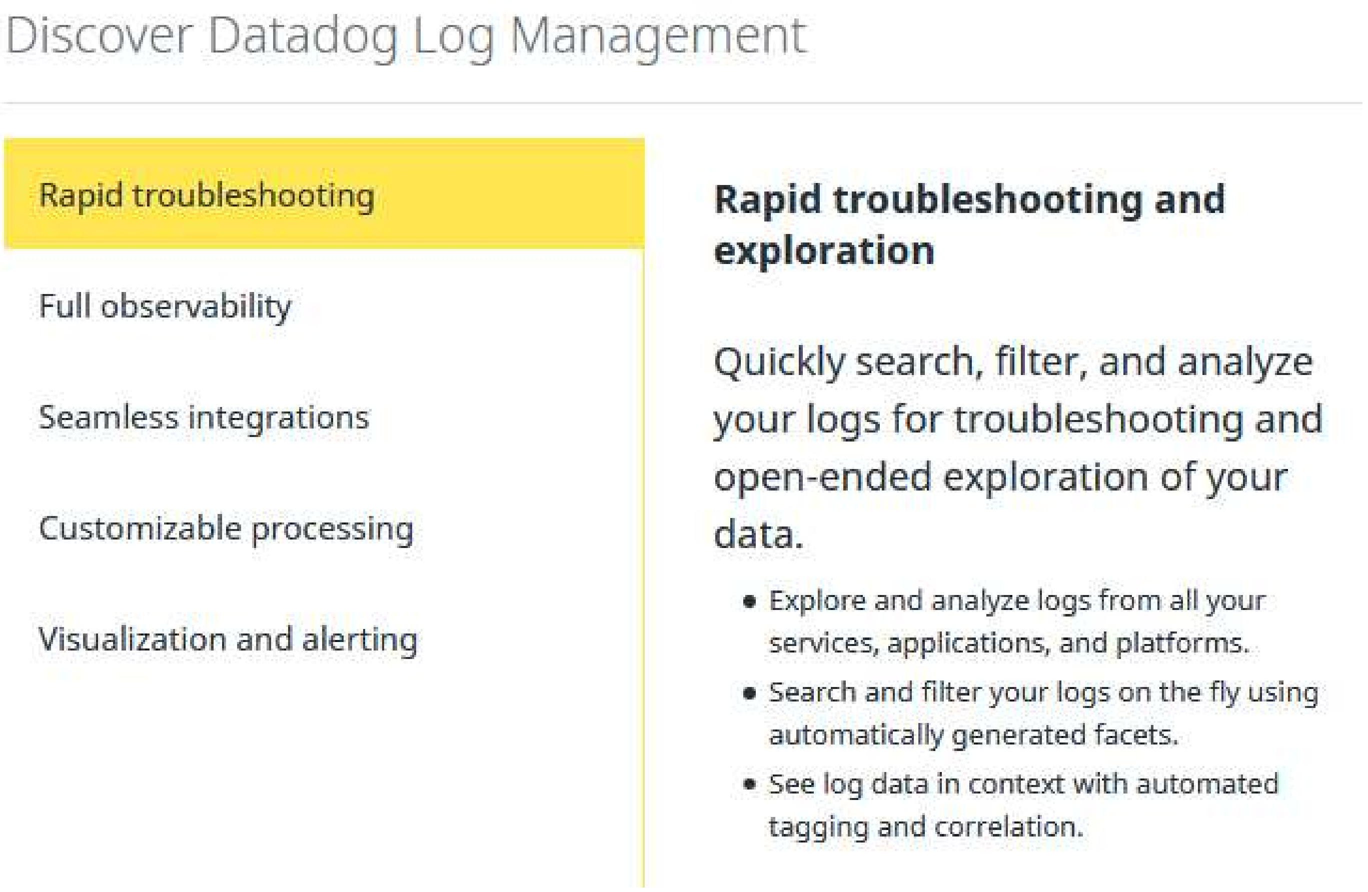
Analyzing Machine Data with Datadog

Get started log management with Datadog

Datadog Log Management: Rapid Troubleshooting



Discover Datadog Log Management

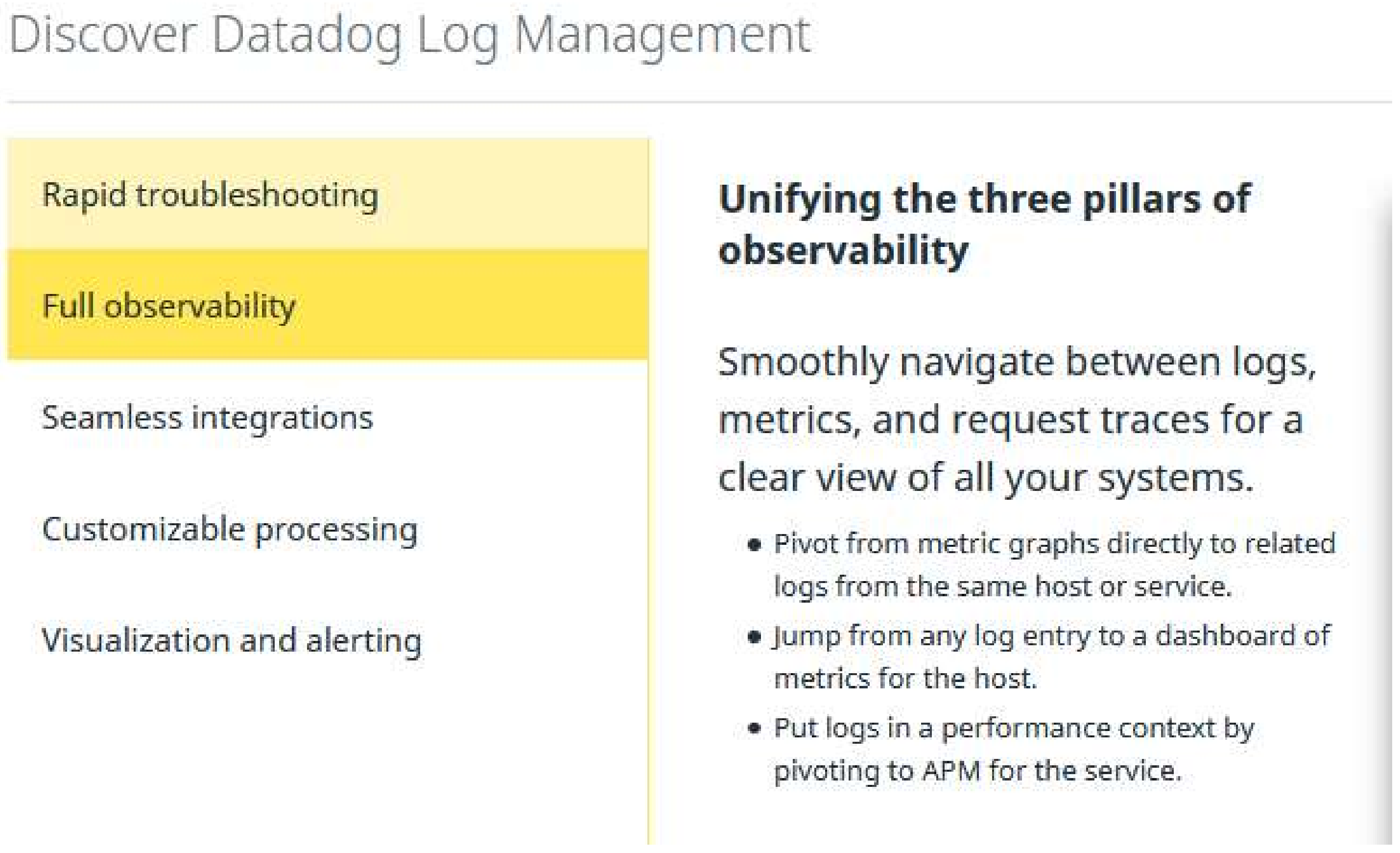
- Rapid troubleshooting
- Full observability
- Seamless integrations
- Customizable processing
- Visualization and alerting

Rapid troubleshooting and exploration

Quickly search, filter, and analyze your logs for troubleshooting and open-ended exploration of your data.

- Explore and analyze logs from all your services, applications, and platforms.
- Search and filter your logs on the fly using automatically generated facets.
- See log data in context with automated tagging and correlation.

Datadog Log Management: Full observability



Discover Datadog Log Management

Rapid troubleshooting

Full observability

Seamless integrations

Customizable processing

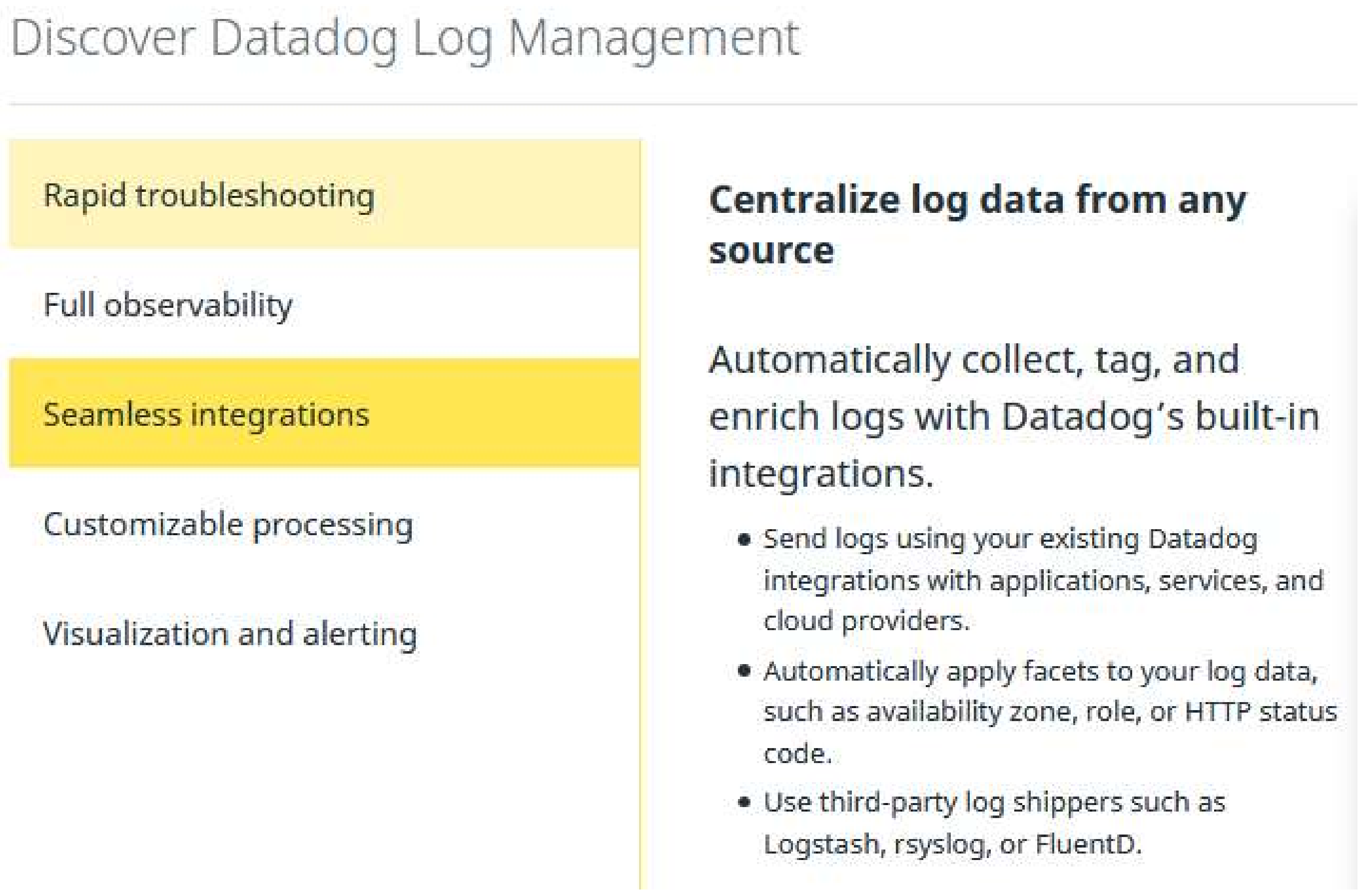
Visualization and alerting

Unifying the three pillars of observability

Smoothly navigate between logs, metrics, and request traces for a clear view of all your systems.

- Pivot from metric graphs directly to related logs from the same host or service.
- Jump from any log entry to a dashboard of metrics for the host.
- Put logs in a performance context by pivoting to APM for the service.

Datadog Log Management: Seamless Integration



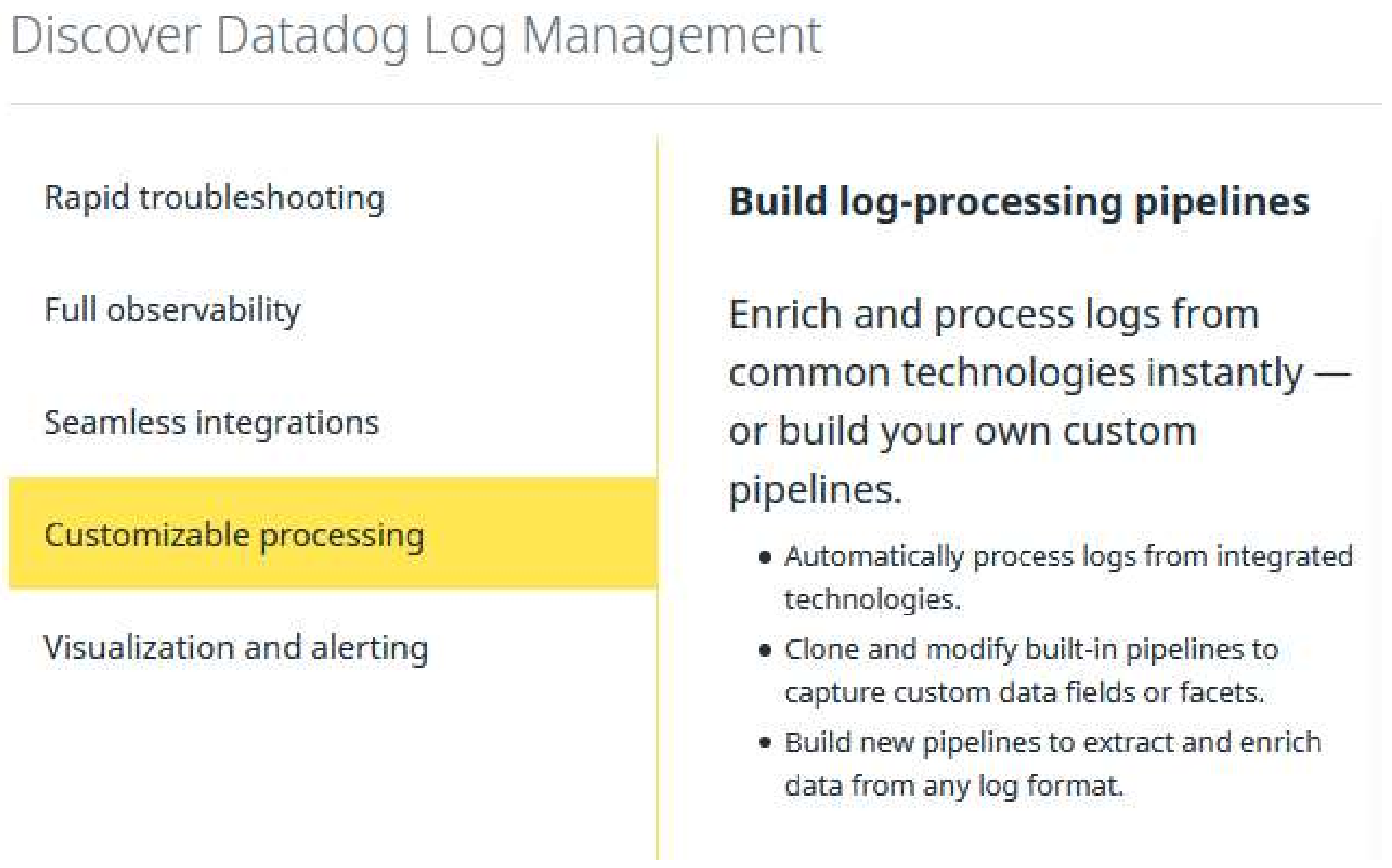
Discover Datadog Log Management

Centralize log data from any source

Automatically collect, tag, and enrich logs with Datadog's built-in integrations.

- Send logs using your existing Datadog integrations with applications, services, and cloud providers.
- Automatically apply facets to your log data, such as availability zone, role, or HTTP status code.
- Use third-party log shippers such as Logstash, rsyslog, or FluentD.

Datadog Log Management: Customizable Processing



Discover Datadog Log Management

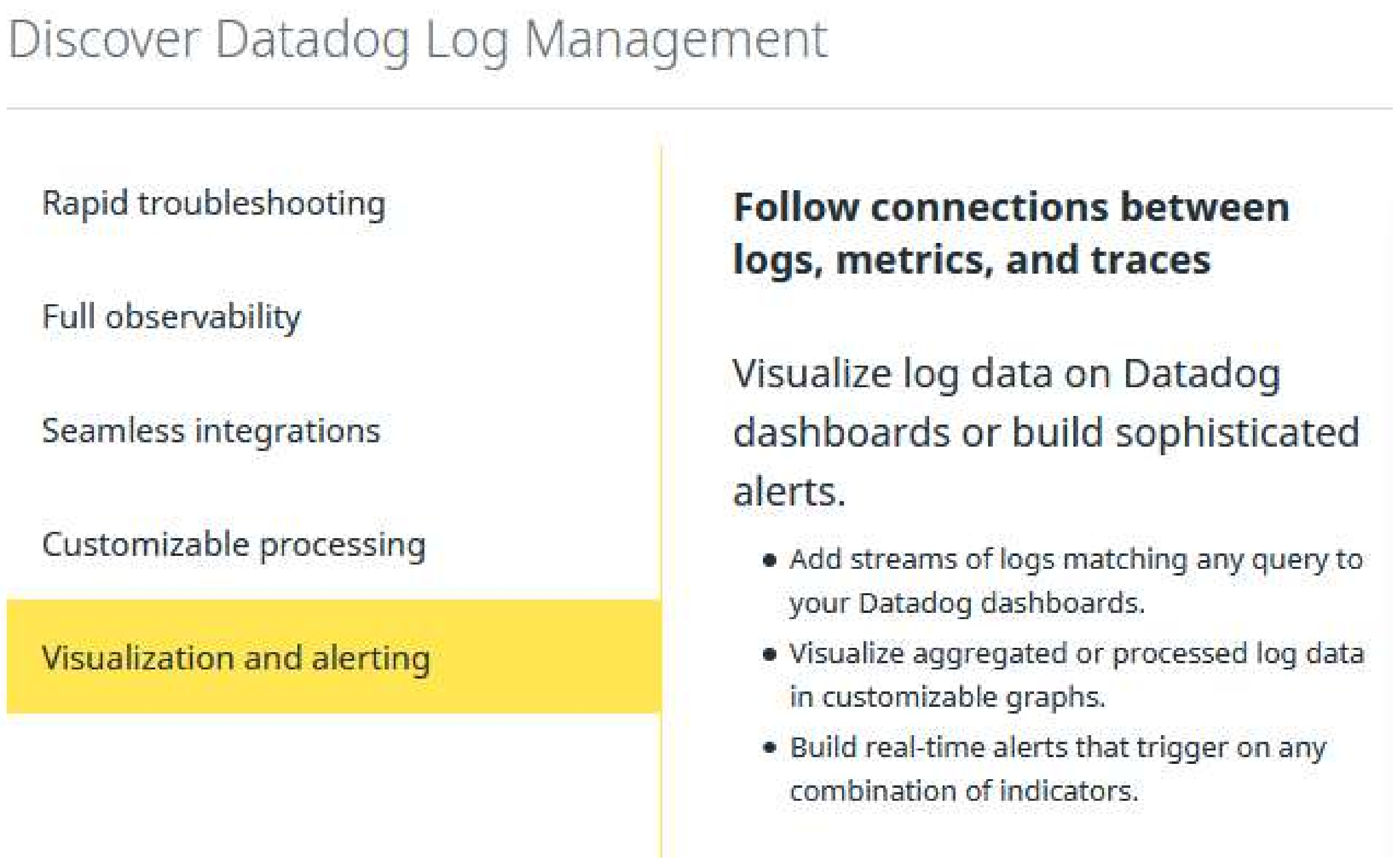
- Rapid troubleshooting
- Full observability
- Seamless integrations
- Customizable processing
- Visualization and alerting

Build log-processing pipelines

Enrich and process logs from common technologies instantly — or build your own custom pipelines.

- Automatically process logs from integrated technologies.
- Clone and modify built-in pipelines to capture custom data fields or facets.
- Build new pipelines to extract and enrich data from any log format.

Datadog Log Management: Visualization and Alerting



Discover Datadog Log Management

- Rapid troubleshooting
- Full observability
- Seamless integrations
- Customizable processing
- Visualization and alerting

Follow connections between logs, metrics, and traces

Visualize log data on Datadog dashboards or build sophisticated alerts.

- Add streams of logs matching any query to your Datadog dashboards.
- Visualize aggregated or processed log data in customizable graphs.
- Build real-time alerts that trigger on any combination of indicators.

Sending Logs to Datadog

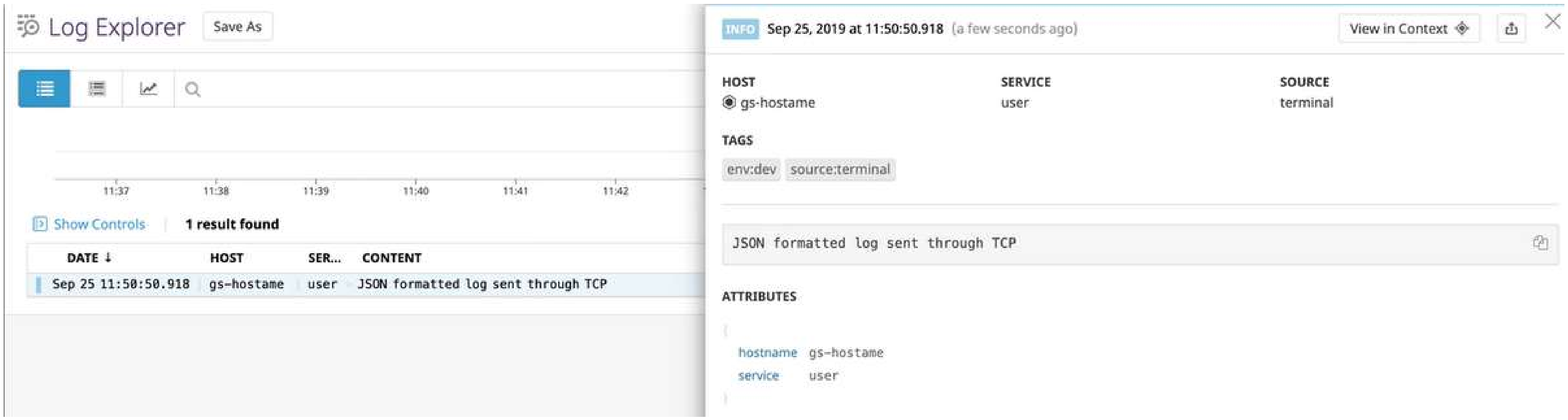
1. Sending logs manually
2. Send logs from a file

Sending Logs to Datadog: Sending logs manually

The secure TCP endpoint is intake.logs.datadoghq.com(or port 10514 for nonsecure connections).

```
telnet intake.logs.datadoghq.com 10514
<DATADOG_API_KEY> Plain text log sent through TCP
```

```
telnet intake.logs.datadoghq.com 10514
<DATADOG_API_KEY> {"message":"JSON formatted log sent through TCP", "ddtags":"env:dev", "ddsource":"terminal",
"hostname":"gs-hostame", "service":"user"}
```



1 result found

DATE ↓	HOST	SER...	CONTENT
Sep 25 11:50:50.918	gs-hostame	user	JSON formatted log sent through TCP

INFO Sep 25, 2019 at 11:50:50.918 (a few seconds ago)

HOST	SERVICE	SOURCE
gs-hostame	user	terminal

TAGS

env:dev source:terminal

```
JSON formatted log sent through TCP
```

ATTRIBUTES

```
{
  hostname  gs-hostame
  service   user
}
```

The TCP endpoint is not supported for this site -
us5.datadoghq.com.

```
curl -X POST "https://http-intake.logs.us5.datadoghq.com/api/v2/logs" \
-H "Content-Type: application/json" \
-H "DD-API-KEY: 061585f627bb034f27fc61cec6a35f3b" \
-d @- << EOF
[
  {
    "ddsource": "nginx",
    "ddtags": "env:staging,version:5.1",
    "hostname": "i-012345678",
    "message": "2019-11-19T14:37:58,995 INFO [process.name][20081] Hello World",
    "service": "payment"
  }
]
EOF
```

Sending Logs to Datadog: Send logs from a file

- Install the Datadog Agent

- Verify Datadog Agent Status and Look for "Logs Agent" which is not running.

```
$ sudo datadog-agent status
```

- Enable log collection

To enable log collection with the Agent, edit the datadog.yaml configuration file located at /etc/datadog-agent/datadog.yaml and set logs_enabled:true

- Monitor a custom file

```
$ touch log_file_to_monitor.log
$ echo "First line of log" >> log_file_to_monitor.log
$ sudo mkdir /etc/datadog-agent/conf.d/custom_log_collection.d/
$ sudo touch /etc/datadog-agent/conf.d/custom_log_collection.d/conf.yaml
logs:
  - type: file
    path: /home/ubuntu/log_file_to_monitor.log
    source: custom
    service: user
$ sudo service datadog-agent restart
```

- Validation. Verify Dsudo datadog-agent status
atadog Agent Status and Look for "Logs Agent" which is running.

```
$
```

- Add new logs to the file

```
$ echo "New line of log in the log file" >> log_file_to_monitor.log
```

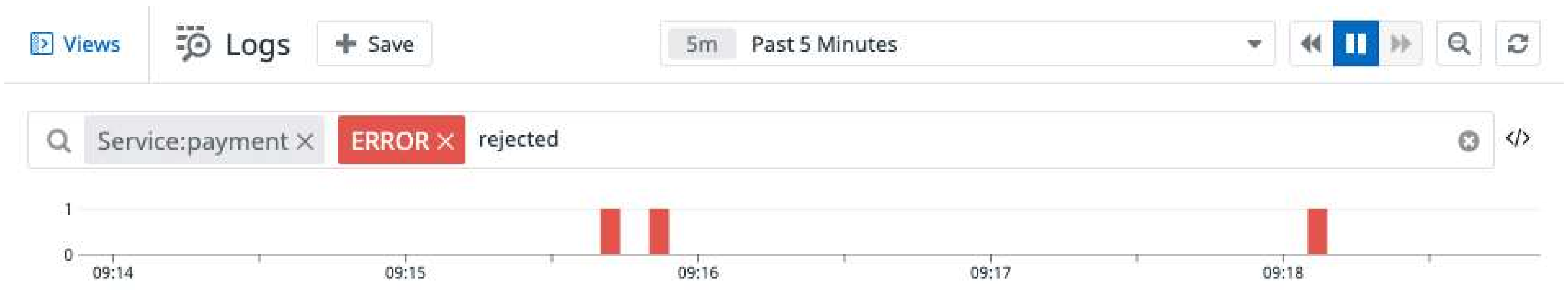
Explore Log

Log Explorer: Discover the Log Explorer view, and how to add Facets and Measures.
Search: Search through all of your logs.
Live Tail: See your ingested logs in real time across all your environments.
Analytics: Perform Log Analytics over your indexed logs.
Patterns: Spot Log Patterns by clustering your indexed logs together.
Saved Views: Use Saved Views to automatically configure your Log Explorer.

Log Explorer

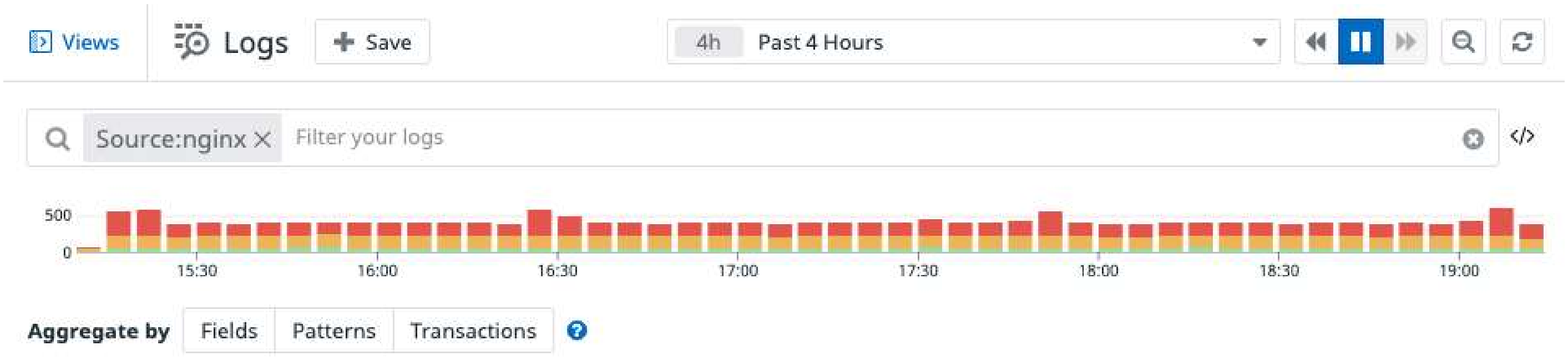
Log Explore: Filters Log

The search filter consists of a timerange and a search query mixing key:value and full-text search. For example, the search query service:payment status:error rejected over a Past 5 minutes timerange:



Log Explore: Aggregate and Measure

Aggregate queried logs into higher-level entities in order to derive or consolidate information. Logs can be valuable as individual events, but sometimes valuable information lives in a subset of events. In order to expose this information, aggregate your logs. Aggregations are supported for indexed logs only



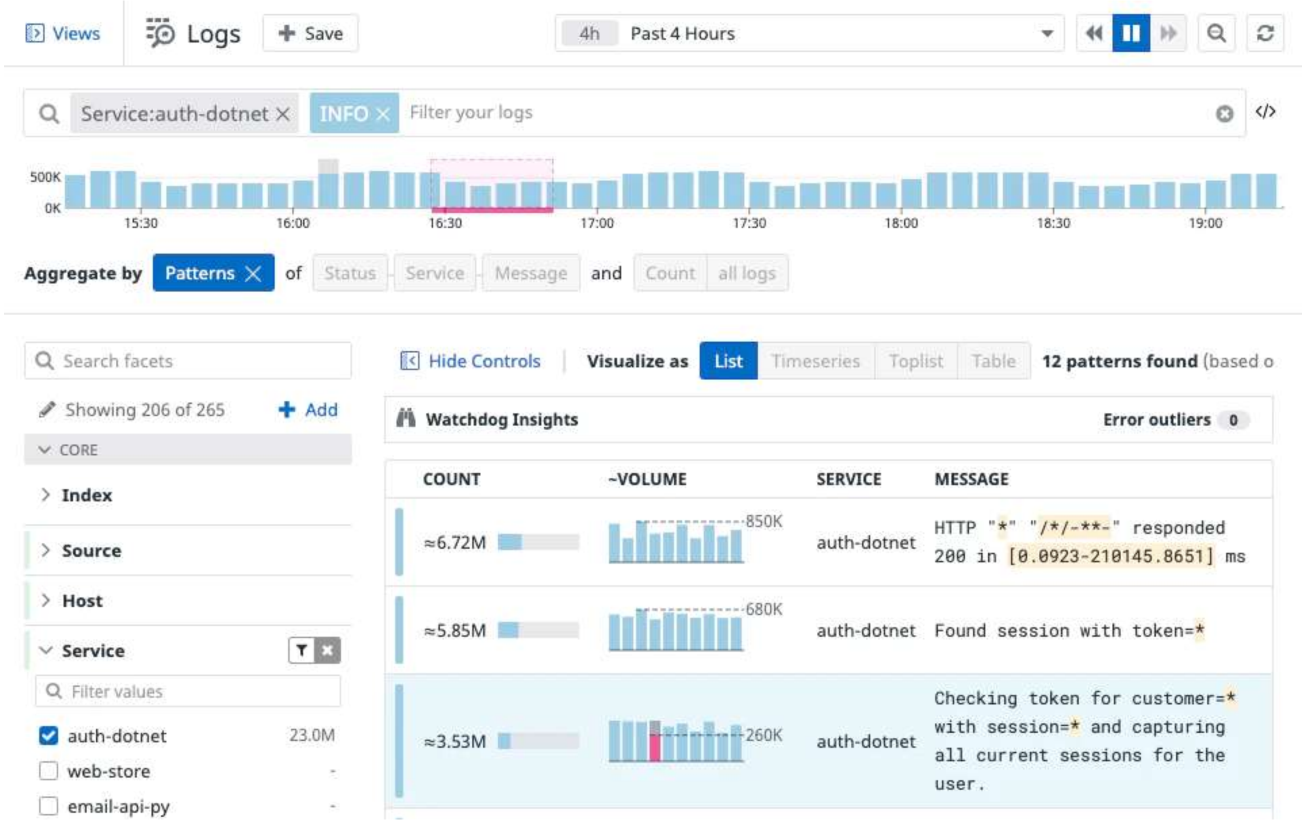
Log Explore: Aggregate -> Fields

With fields aggregation, all logs matching the query filter are aggregated into groups based on the value of one or multiple log facets. On top of these aggregates, you can extract the following measures:

- count of logs per group
- unique count of coded values for a facet per group
- statistical operations (min, max, avg, and percentiles) on numerical values of a facet per group

Log Explore: Aggregate -> Patterns

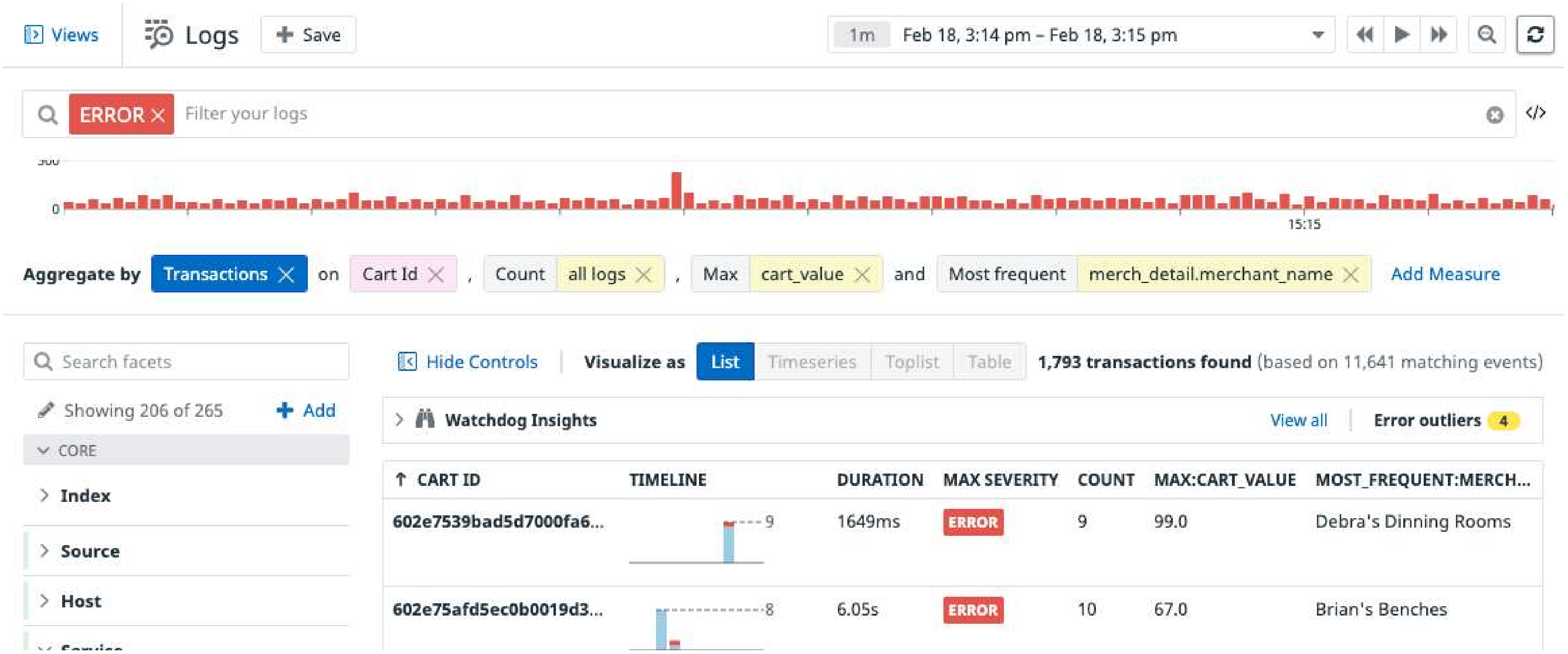
With pattern aggregation, logs that have a message with similar structures, belong to the same service and have the same status are grouped altogether. The patterns view is helpful for detecting and filtering noisy error patterns that could cause you to miss other issues:



COUNT	~VOLUME	SERVICE	MESSAGE
≈6.72M	850K	auth-dotnet	`HTTP "*" "/*/-**-" responded 200 in [0.0923-210145.8651] ms`
≈5.85M	680K	auth-dotnet	`Found session with token=*`
≈3.53M	260K	auth-dotnet	`Checking token for customer=* with session=* and capturing all current sessions for the user.`

Log Explore: Aggregate -> Transactions

Transactions aggregate indexed logs according to instances of a sequence of events, such as a user session or a request processed across multiple micro-services. For example, an e-commerce website groups log events across various user actions, such as catalog search, add to cart, and checkout, to build a transaction view using a common attribute such as requestId or orderId.



Aggregate by Transactions on Cart Id, Count all logs, Max cart_value and Most frequent merch_detail.merchant_name Add Measure

Visualize as List Timeseries Toplist Table **1,793 transactions found** (based on 11,641 matching events)

CART ID	TIMELINE	DURATION	MAX SEVERITY	COUNT	MAX:CART_VALUE	MOST_FREQUENT:MERCH...
602e7539bad5d7000fa6...	9	1649ms	ERROR	9	99.0	Debra's Dinning Rooms
602e75afd5ec0b0019d3...	8	6.05s	ERROR	10	67.0	Brian's Benches

Log Explore: Aggregate -> Visualize

Lists - The columns displayed in list of aggregates are columns derived from the aggregation.

Timeseries - Visualize the evolution of a single measure (or a facet unique count of values) over a selected time frame, and (optionally) split by an available facet.

Toplists - Visualize the top values from a facet according to the chosen measure.

Nested tables - Visualize the top values from a facet according to a chosen measure (the first measure you choose in the list), and display the value of additional measures for elements appearing in this table.

Export - At any moment, and depending on your current aggregation, export your exploration as a: Saved View, Dashboard widget, Monitor, Metric, CSV & Share

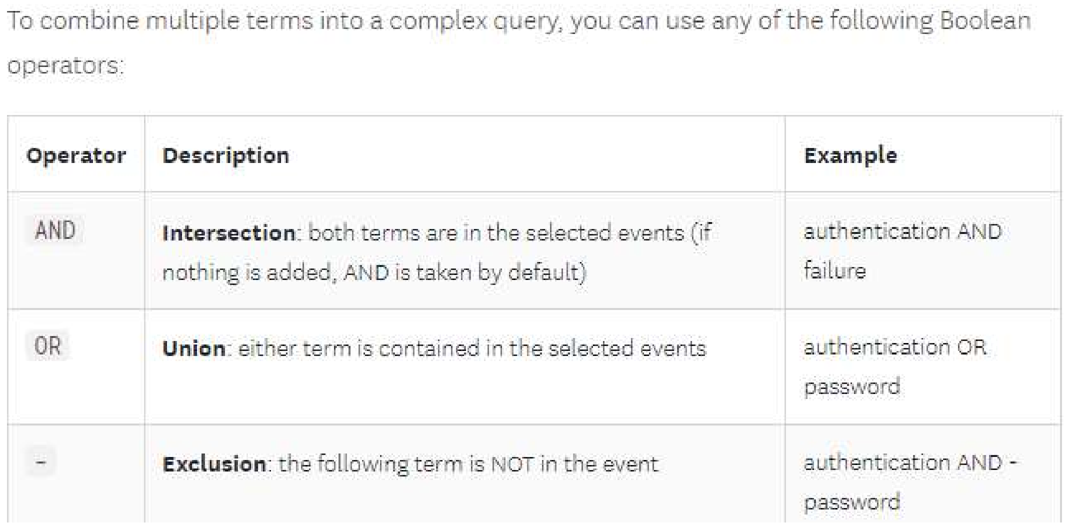
Search Log

Search Syntax

A query filter is composed of terms and operators.

There are two types of terms:

- A single term is a single word such as test or hello.
- A sequence is a group of words surrounded by double quotes, such as "hello dolly".



To combine multiple terms into a complex query, you can use any of the following Boolean operators:

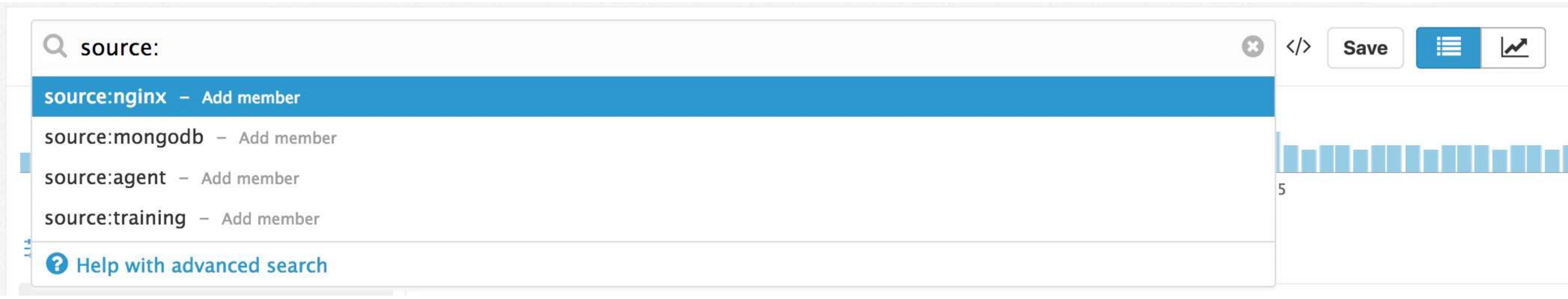
Operator	Description	Example
`AND`	**Intersection**: both terms are in the selected events (if nothing is added, AND is taken by default)	authentication AND failure
`OR`	**Union**: either term is contained in the selected events	authentication OR password
`-`	**Exclusion**: the following term is NOT in the event	authentication AND - password

Search Syntax: AND OR

To combine multiple terms into a complex query, you can use any of the following Boolean operators:

Operator	Description	Example
`AND`	**Intersection**: both terms are in the selected events (if nothing is added, AND is taken by default)	authentication AND failure
`OR`	**Union**: either term is contained in the selected events	authentication OR password
`-`	**Exclusion**: the following term is NOT in the event	authentication AND -password

Search Syntax: Autocomplete



source:

source:nginx – Add member

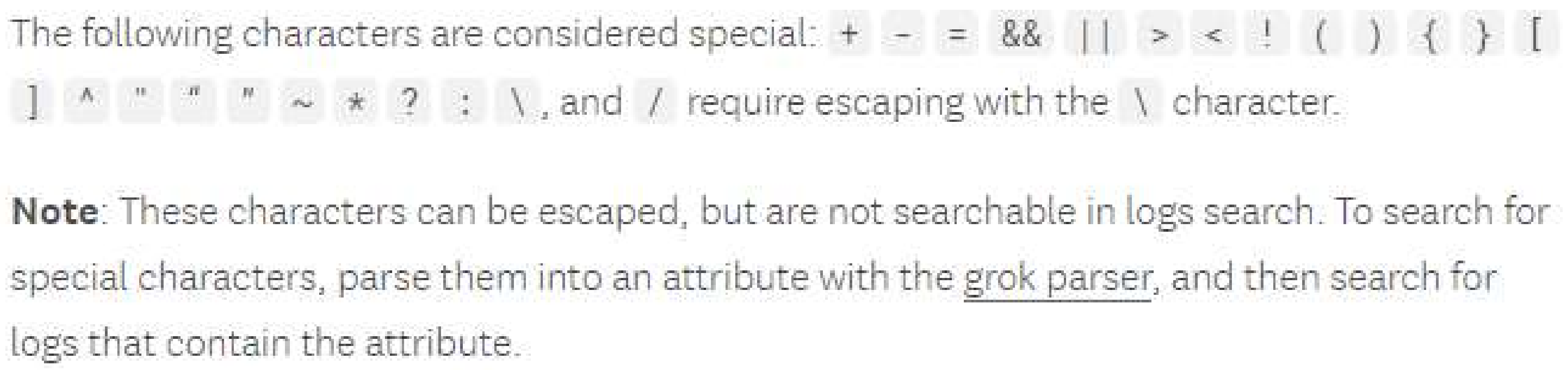
source:mongodb – Add member

source:agent – Add member

source:training – Add member

Help with advanced search

Search Syntax: Escaping of special characters



The following characters are considered special: `+` `-` `=` `&&` `||` `>` `<` `!` `(` `)` `{` `}` `[` `]` `^` `"` `"` `"` `~` `*` `?` `:` `\`, and `/` require escaping with the `\` character.

Note: These characters can be escaped, but are not searchable in logs search. To search for special characters, parse them into an attribute with the grok parser, and then search for logs that contain the attribute.

Search Syntax: Attributes search

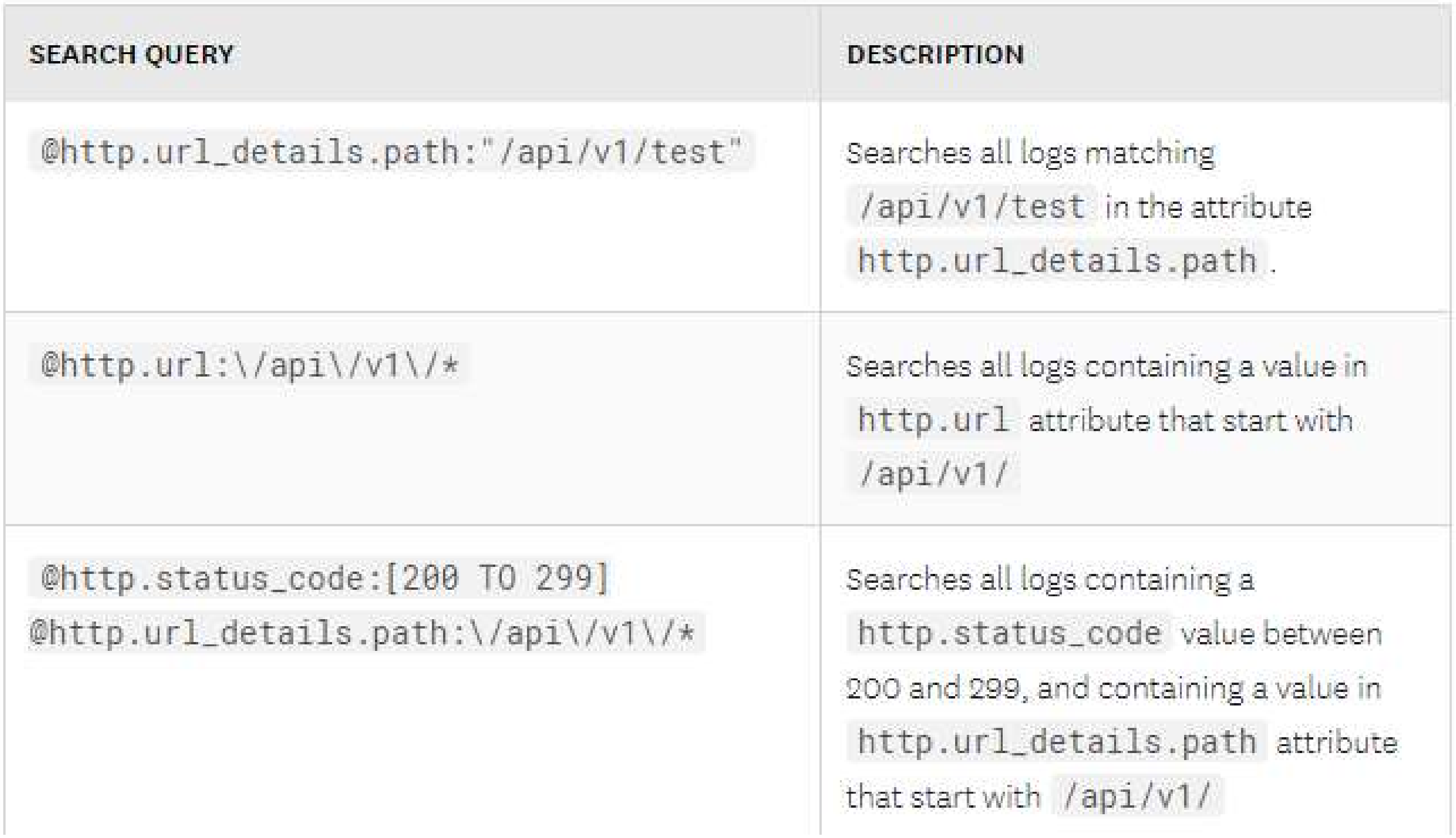
Message attribute search
To search for logs that contain user=JaneDoe in the message attribute use the following search:

```
user\=JaneDoe
```

Facets search
To search on a specific attribute, first add it as a facet and then add @ to specify you are searching on a facet. For instance, if your facet name is url and you want to filter on the url value www.datadoghq.com, enter:

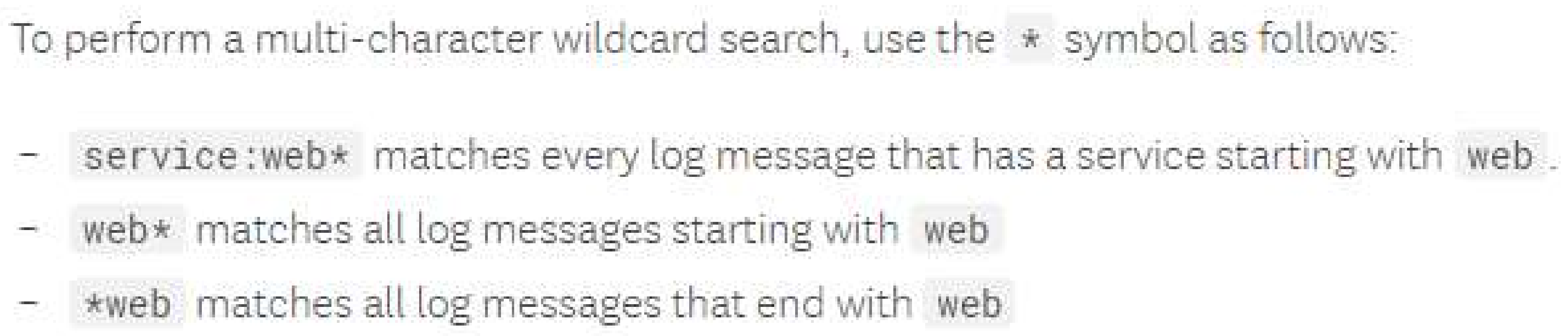
```
@url:www.datadoghq.com
```

Search Syntax: Attributes search



SEARCH QUERY	DESCRIPTION
`@http.url_details.path:"/api/v1/test"`	Searches all logs matching `/api/v1/test` in the attribute `http.url_details.path`.
`@http.url:\/api\/v1\/*`	Searches all logs containing a value in `http.url` attribute that start with `/api/v1/`
`@http.status_code:[200 TO 299]` `@http.url_details.path:\/api\/v1\/*`	Searches all logs containing a `http.status_code` value between 200 and 299, and containing a value in `http.url_details.path` attribute that start with `/api/v1/`

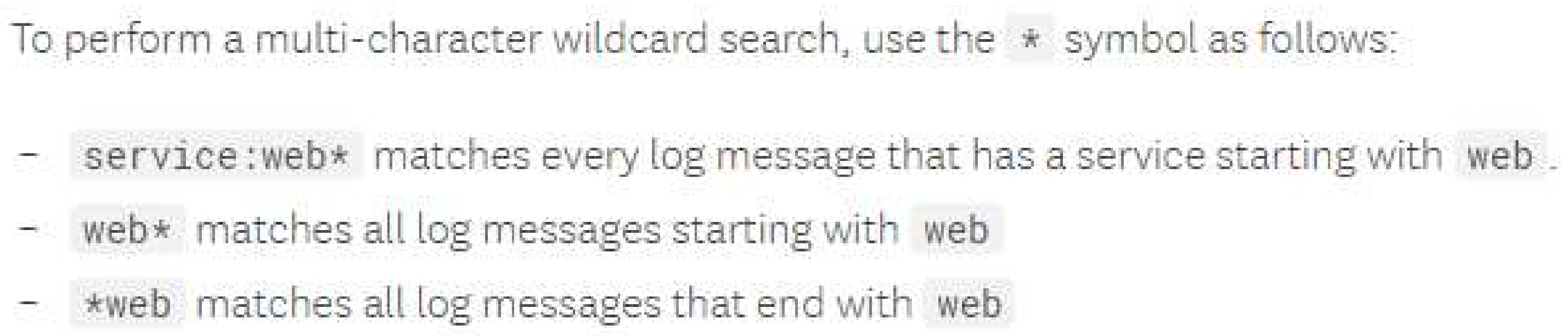
Search Syntax: Wildcards



To perform a multi-character wildcard search, use the `*` symbol as follows:

- `service:web*` matches every log message that has a service starting with `web`.
- `web*` matches all log messages starting with `web`
- `*web` matches all log messages that end with `web`

Search Syntax: Wildcards



To perform a multi-character wildcard search, use the `*` symbol as follows:

- `service:web*` matches every log message that has a service starting with `web`.
- `web*` matches all log messages starting with `web`
- `*web` matches all log messages that end with `web`

Search Syntax: Numerical values

Use `<`, `>`, `<=`, or `>=` to perform a search on numerical attributes. For instance, retrieve all logs that have a response time over 100ms with:

```
@http.response_time:>100
```

You can search for numerical attribute within a specific range. For instance, retrieve all your 4xx errors with:

```
@http.status_code:[400 TO 499]
```

Search Syntax: Tags

Your logs inherit tags from hosts and integrations that generate them. They can be used in the search and as facets as well:

- `test` is searching for the string "test".
- `env:(prod OR test)` matches all logs with the tag `env:prod` or the tag `env:test`
- `(env:prod AND -version:beta)` matches all logs that contain tag `env:prod` and that do not contain tag `version:beta`

If your tags don't follow tags best practices and don't use the `key:value` syntax, use this search query:

- `tags:<MY_TAG>`

Forwarding / Flushing Metrics to Datadog Cloud from restricted outbound traffic

- Using a web proxy, such as Squid or Microsoft Web Proxy, that is already deployed to your network
- Using HAProxy (if you want to proxy more than 16-20 Agents through the same proxy)
- Using the Agent as a proxy (for up to 16 Agents per proxy, only on Agent v5)
- Using Prometheus Pushgateway - Refer - https://openapm.io/landscape/collector/prometheus-push-gateway

Pipelines

What is Pipelines?

Datadog automatically parses JSON-formatted logs.

When logs are not JSON-formatted, you can add value to your raw logs by sending them through a **processing pipeline.**

What is Pipelines?

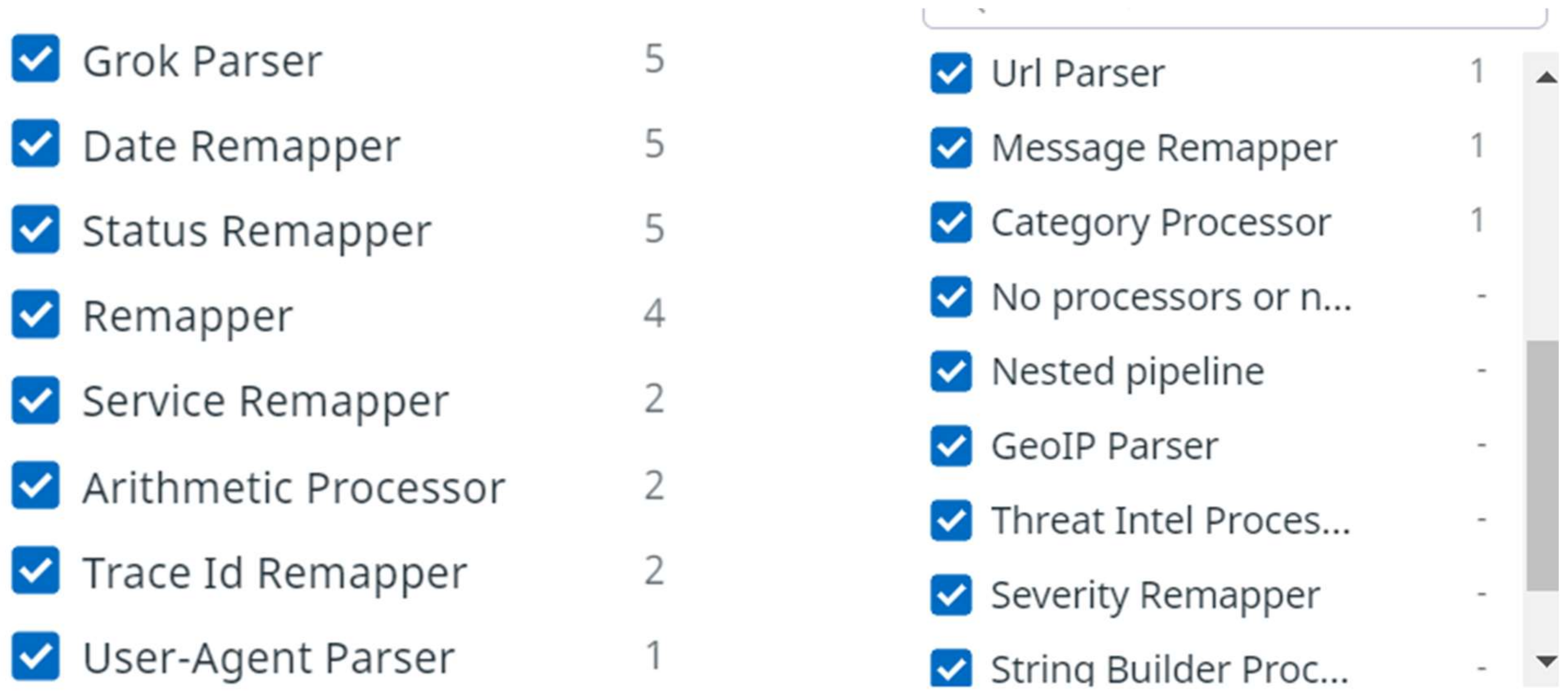
Pipelines take logs from a wide **variety of formats** and translate them into a **common format** in Datadog.

Implementing a log pipelines and processing strategy is beneficial as it introduces an **attribute naming convention** for your organization.

What is Pipelines?

With pipelines, logs are parsed and enriched by chaining them sequentially **through processors**. This extracts meaningful information or attributes from semi-structured text to reuse as facets.
Each log that comes through the pipelines is tested against every **pipeline filter**. If it matches a filter, then all the processors are applied sequentially before moving to the next pipeline.

Processors



✓	Grok Parser	5
✓	Date Remapper	5
✓	Status Remapper	5
✓	Remapper	4
✓	Service Remapper	2
✓	Arithmetic Processor	2
✓	Trace Id Remapper	2
✓	User-Agent Parser	1

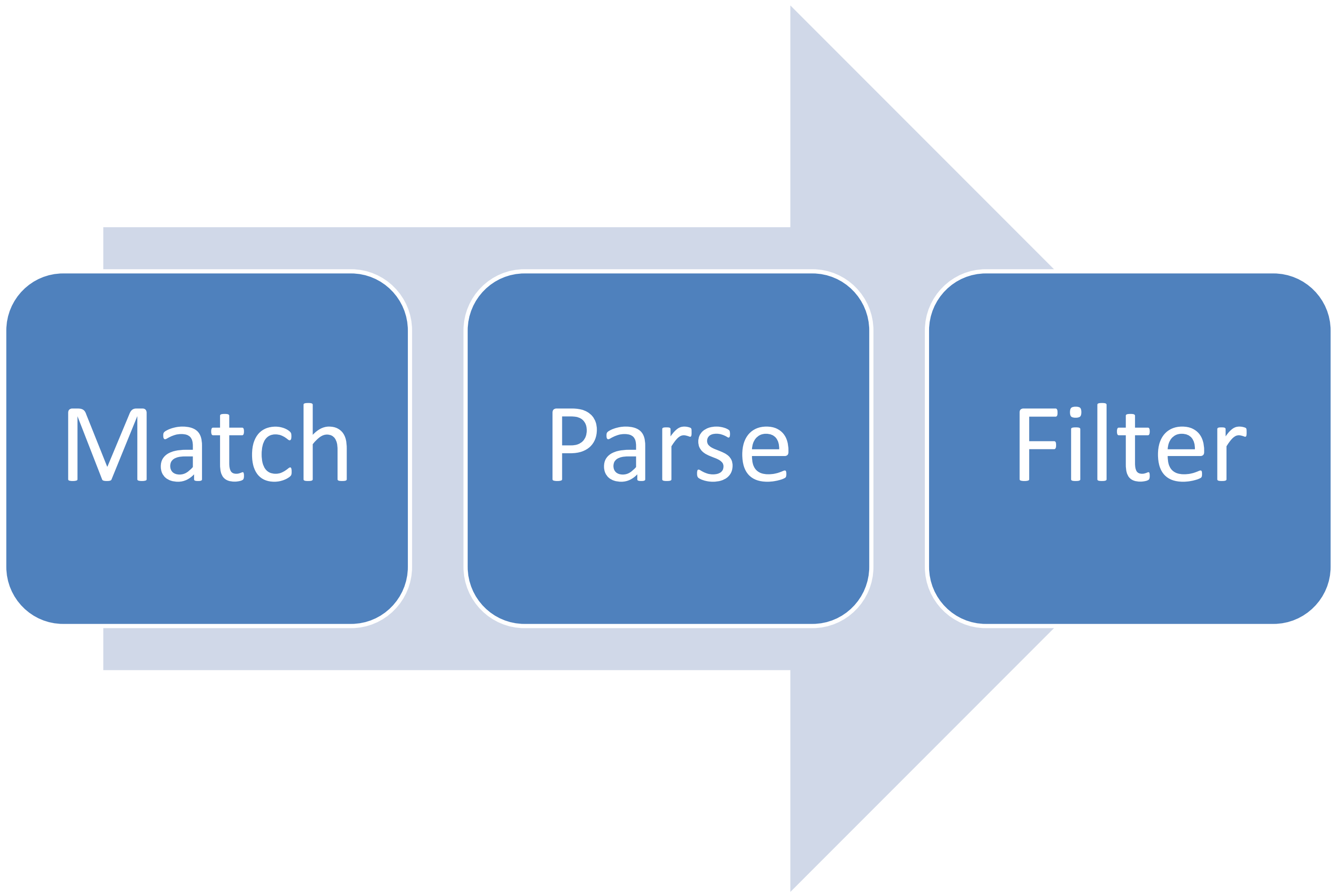
✓	Url Parser	1
✓	Message Remapper	1
✓	Category Processor	1
✓	No processors or n...	-
✓	Nested pipeline	-
✓	GeoIP Parser	-
✓	Threat Intel Proces...	-
✓	Severity Remapper	-
✓	String Builder Proc...	-

Processors

Grok parser
Log date remapper
Log status remapper
Service remapper
Log message remapper
Remapper
URL parser
User-Agent parser
Category processor
Arithmetic processor
String builder processor
GeoIP parser
Lookup processor
Trace remapper

https://docs.datadoghq.com/logs/log_configuration/processors

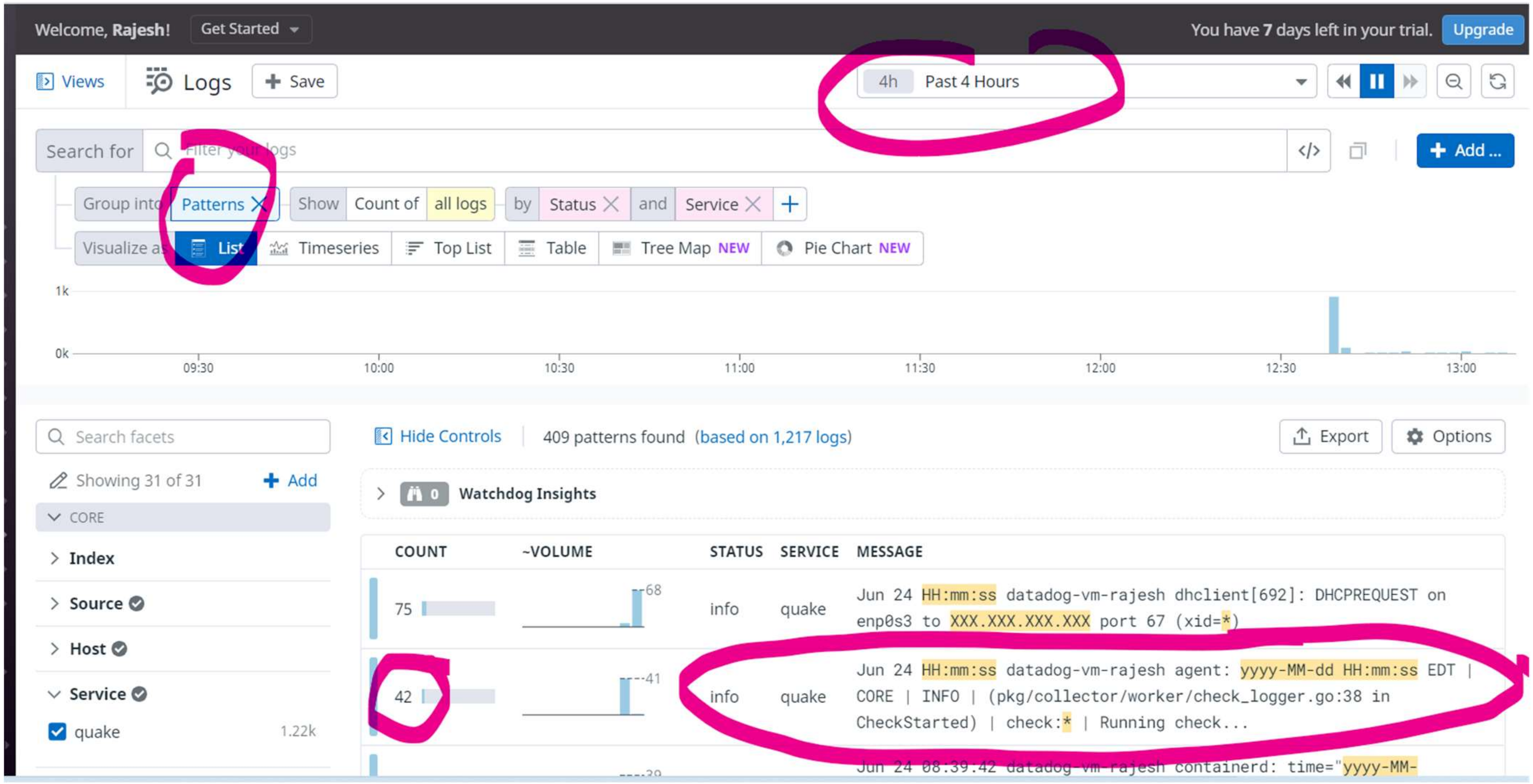
Pipelines Workflow



You can write parsing rules with the `%{MATCHER:EXTRACT:FILTER}` syntax:

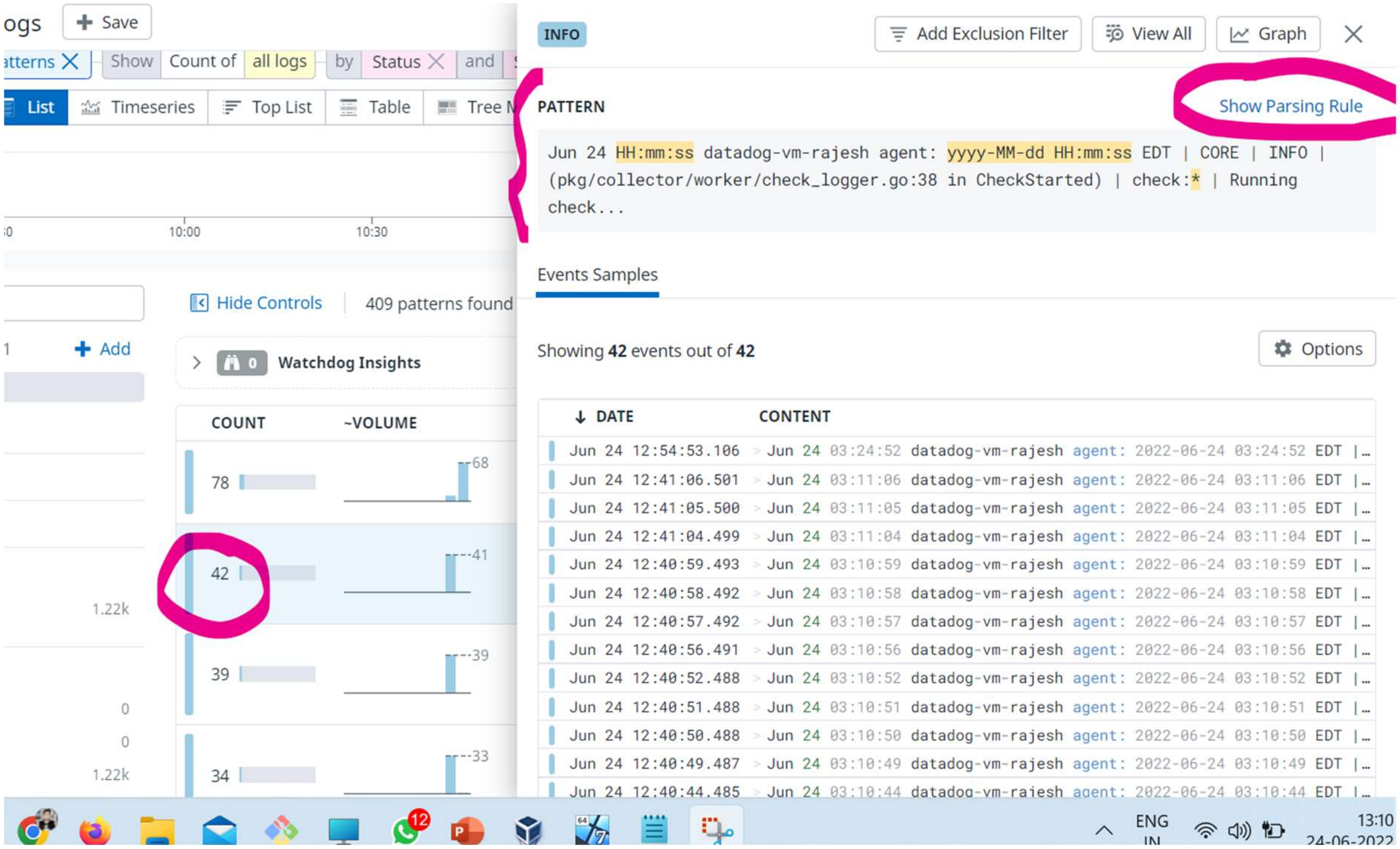
- **Matcher**: A rule (possibly a reference to another token rule) that describes what to expect (number, word, notSpace, etc.).
- **Extract** (optional): An identifier representing the capture destination for the piece of text matched by the *Matcher*.
- **Filter** (optional): A post-processor of the match to transform it.

Step 1 – Know a Log Pattern



COUNT	~VOLUME	STATUS	SERVICE	MESSAGE
75	68	info	quake	Jun 24 HH:mm:ss datadog-vm-rajesh dhclient[692]: DHCPREQUEST on enp0s3 to XXX.XXX.XXX.XXX port 67 (xid=*)
42	41	info	quake	Jun 24 HH:mm:ss datadog-vm-rajesh agent: yyyy-MM-dd HH:mm:ss EDT | CORE | INFO | (pkg/collector/worker/check_logger.go:38 in CheckStarted) | check:* | Running check...
	39			Jun 24 08:39:42 datadog-vm-rajesh containerd: time="yyyy-MM-

Step 2 – Understand a Pattern & Parsing Rule

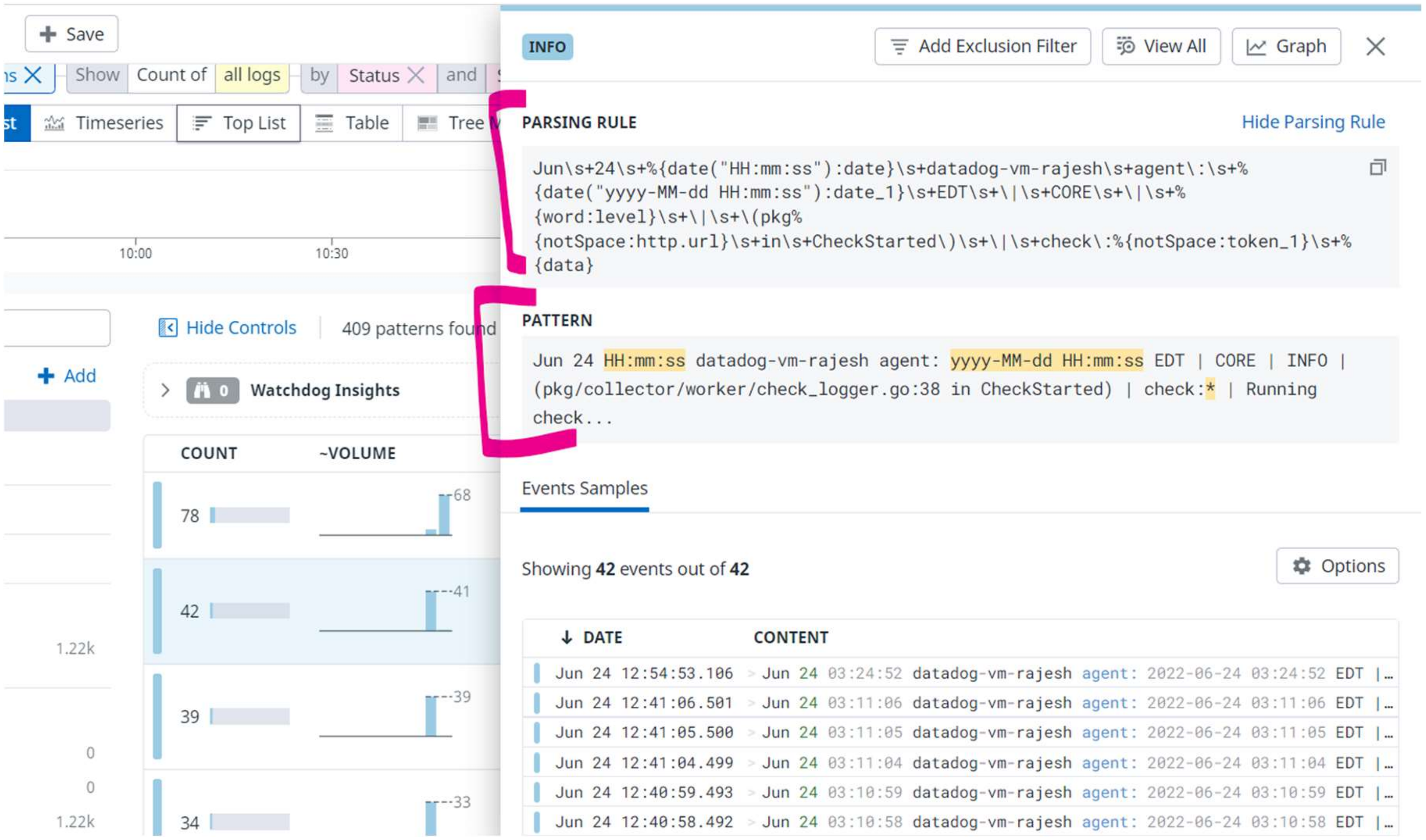


```
Jun 24 HH:mm:ss datadog-vm-rajesh agent: yyyy-MM-dd HH:mm:ss EDT | CORE | INFO |
(pkg/collector/worker/check_logger.go:38 in CheckStarted) | check:* | Running
check...
```

Showing **42** events out of **42**

```
↓ DATE                  CONTENT
Jun 24 12:54:53.106   Jun 24 03:24:52 datadog-vm-rajesh agent: 2022-06-24 03:24:52 EDT |…
Jun 24 12:41:06.501   Jun 24 03:11:06 datadog-vm-rajesh agent: 2022-06-24 03:11:06 EDT |…
Jun 24 12:41:05.500   Jun 24 03:11:05 datadog-vm-rajesh agent: 2022-06-24 03:11:05 EDT |…
Jun 24 12:41:04.499   Jun 24 03:11:04 datadog-vm-rajesh agent: 2022-06-24 03:11:04 EDT |…
Jun 24 12:40:59.493   Jun 24 03:10:59 datadog-vm-rajesh agent: 2022-06-24 03:10:59 EDT |…
Jun 24 12:40:58.492   Jun 24 03:10:58 datadog-vm-rajesh agent: 2022-06-24 03:10:58 EDT |…
Jun 24 12:40:57.492   Jun 24 03:10:57 datadog-vm-rajesh agent: 2022-06-24 03:10:57 EDT |…
Jun 24 12:40:56.491   Jun 24 03:10:56 datadog-vm-rajesh agent: 2022-06-24 03:10:56 EDT |…
Jun 24 12:40:52.488   Jun 24 03:10:52 datadog-vm-rajesh agent: 2022-06-24 03:10:52 EDT |…
Jun 24 12:40:51.488   Jun 24 03:10:51 datadog-vm-rajesh agent: 2022-06-24 03:10:51 EDT |…
Jun 24 12:40:50.488   Jun 24 03:10:50 datadog-vm-rajesh agent: 2022-06-24 03:10:50 EDT |…
Jun 24 12:40:49.487   Jun 24 03:10:49 datadog-vm-rajesh agent: 2022-06-24 03:10:49 EDT |…
Jun 24 12:40:44.485   Jun 24 03:10:44 datadog-vm-rajesh agent: 2022-06-24 03:10:44 EDT |…
```

Step 3 – Copy Parsing Rule & Pattern



PARSING RULE

```
Jun\s+24\s+%{date("HH:mm:ss"):date}\s+datadog-vm-rajesh\s+agent\:\s+%
{date("yyyy-MM-dd HH:mm:ss"):date_1}\s+EDT\s+\|\s+CORE\s+\|\s+%
{word:level}\s+\|\s+\(pkg%
{notSpace:http.url}\s+in\s+CheckStarted\)\s+\|\s+check\:%{notSpace:token_1}\s+%
{data}
```

PATTERN

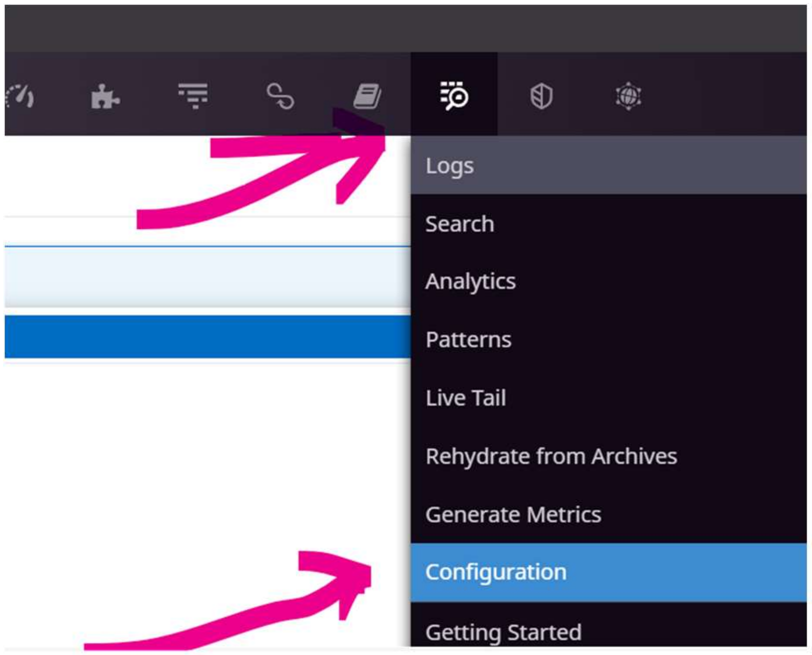
```
Jun 24 HH:mm:ss datadog-vm-rajesh agent: yyyy-MM-dd HH:mm:ss EDT | CORE | INFO |
(pkg/collector/worker/check_logger.go:38 in CheckStarted) | check:* | Running
check...
```

Showing **42** events out of **42**

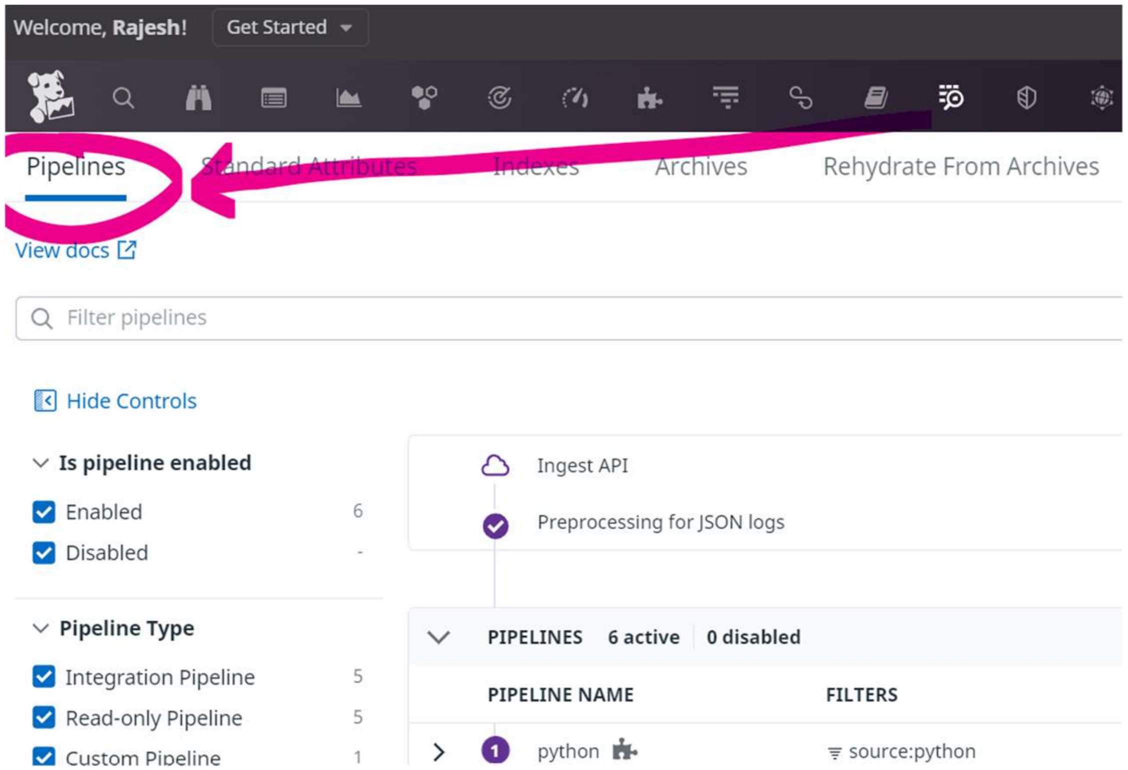
COUNT	~VOLUME
78	68
42	41
39	39
34	33

```
↓ DATE                  CONTENT
Jun 24 12:54:53.106  >  Jun 24 03:24:52 datadog-vm-rajesh agent: 2022-06-24 03:24:52 EDT |…
Jun 24 12:41:06.501  >  Jun 24 03:11:06 datadog-vm-rajesh agent: 2022-06-24 03:11:06 EDT |…
Jun 24 12:41:05.500  >  Jun 24 03:11:05 datadog-vm-rajesh agent: 2022-06-24 03:11:05 EDT |…
Jun 24 12:41:04.499  >  Jun 24 03:11:04 datadog-vm-rajesh agent: 2022-06-24 03:11:04 EDT |…
Jun 24 12:40:59.493  >  Jun 24 03:10:59 datadog-vm-rajesh agent: 2022-06-24 03:10:59 EDT |…
Jun 24 12:40:58.492  >  Jun 24 03:10:58 datadog-vm-rajesh agent: 2022-06-24 03:10:58 EDT |…
```

Step 4 – Navigate to Pipeline



Logs
Search
Analytics
Patterns
Live Tail
Rehydrate from Archives
Generate Metrics
Configuration
Getting Started



Welcome, **Rajesh**! Get Started

Pipelines Standard Attributes Indexes Archives Rehydrate From Archives

View docs

Filter pipelines

Hide Controls

Is pipeline enabled

✓ Enabled	6
✓ Disabled	-

Pipeline Type

✓ Integration Pipeline	5
✓ Read-only Pipeline	5
✓ Custom Pipeline	1

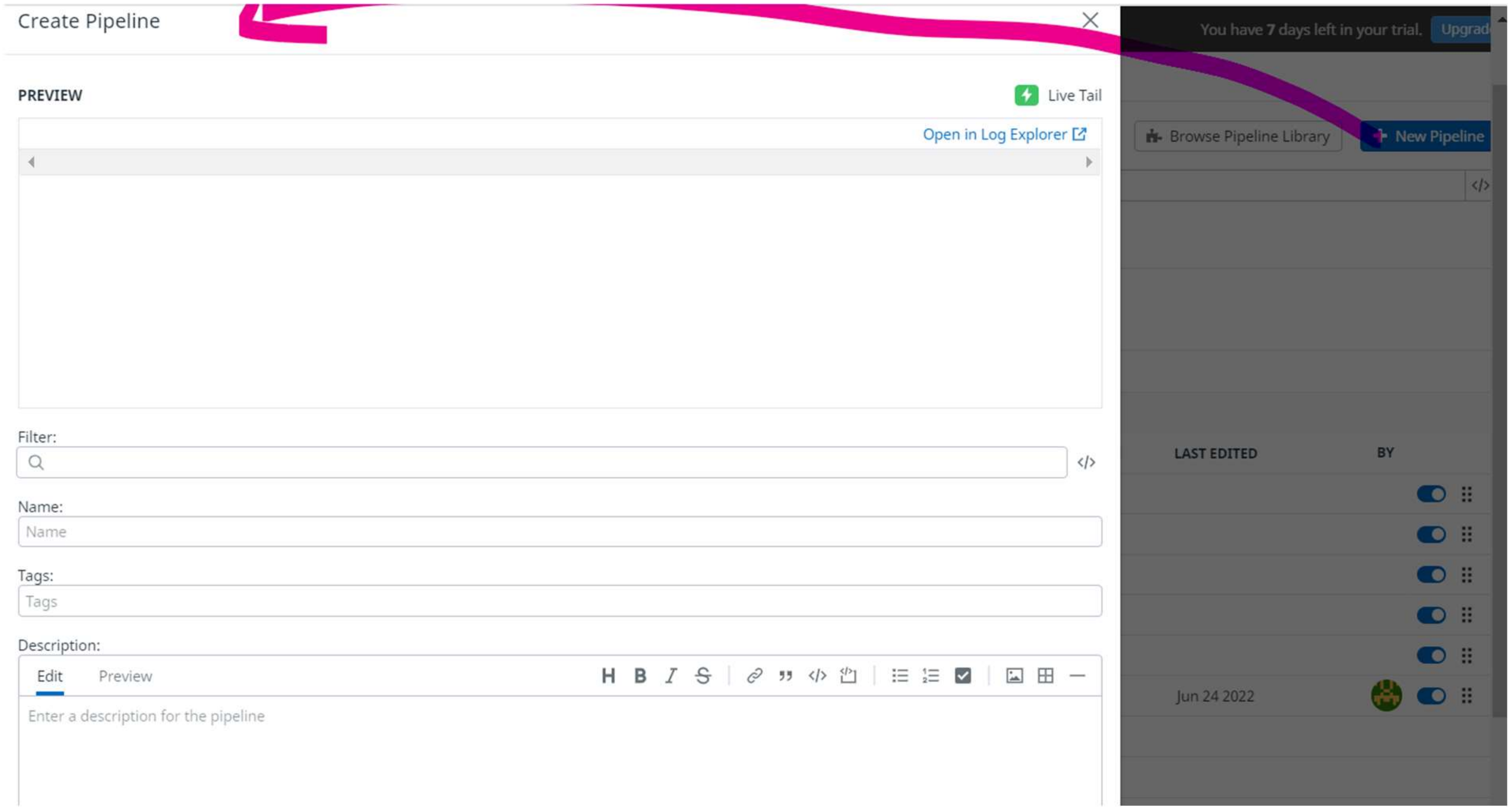
Ingest API

Preprocessing for JSON logs

PIPELINES 6 active 0 disabled

PIPELINE NAME	FILTERS
1 python	source:python

Step 5 – Understand a Pattern & Parsing Rule



Create Pipeline

PREVIEW

Live Tail

Open in Log Explorer

Filter:

Name:

Name

Tags:

Tags

Description:

Edit Preview

Enter a description for the pipeline

You have 7 days left in your trial. Upgrad

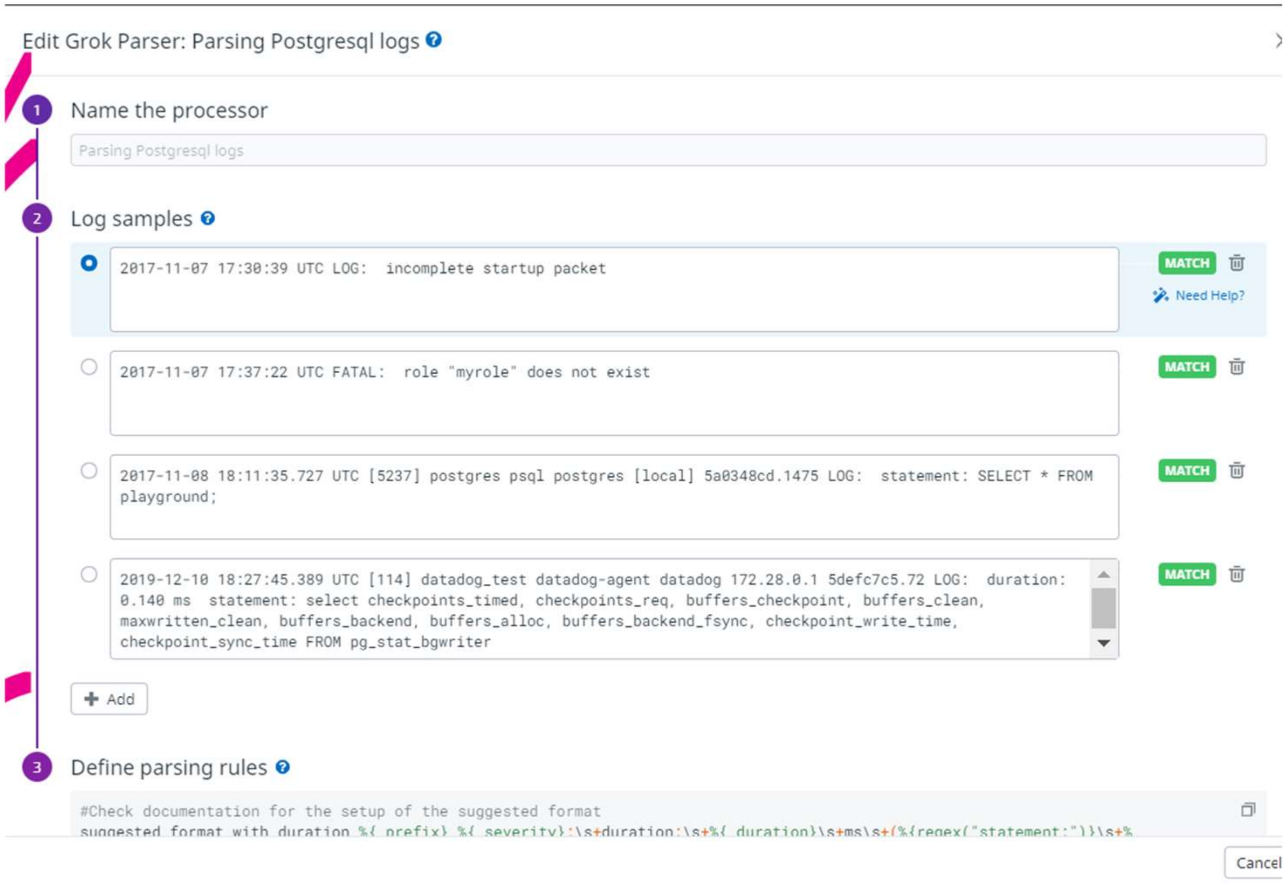
Browse Pipeline Library

New Pipeline

LAST EDITED BY

Jun 24 2022

Step 6 – Understand a Pattern & Parsing Rule



Edit Grok Parser: Parsing Postgresql logs

1 Name the processor

Parsing Postgresql logs

2 Log samples

```
2017-11-07 17:30:39 UTC LOG:  incomplete startup packet
```

MATCH · Need Help?

```
2017-11-07 17:37:22 UTC FATAL:  role "myrole" does not exist
```

MATCH

```
2017-11-08 18:11:35.727 UTC [5237] postgres psql postgres [local] 5a0348cd.1475 LOG:  statement: SELECT * FROM
playground;
```

MATCH

```
2019-12-10 18:27:45.389 UTC [114] datadog_test datadog-agent datadog 172.28.0.1 5defc7c5.72 LOG:  duration:
0.140 ms  statement: select checkpoints_timed, checkpoints_req, buffers_checkpoint, buffers_clean,
maxwritten_clean, buffers_backend, buffers_alloc, buffers_backend_fsync, checkpoint_write_time,
checkpoint_sync_time FROM pg_stat_bgwriter
```

MATCH

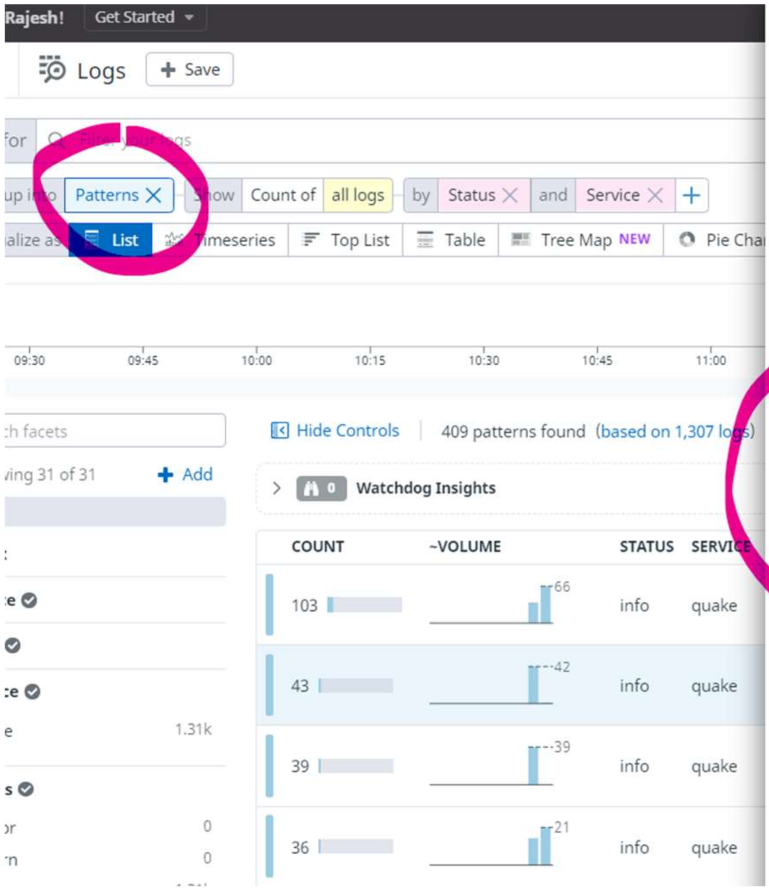
+ Add

3 Define parsing rules

```
#Check documentation for the setup of the suggested format
suggested_format_with_duration %{_prefix} %{_severity}:\s+duration:\s+%{_duration}\s+ms\s+(%{regex("statement:")}\s+%
```

Cancel

Step 8 – Results - Before



Rajesh! Get Started

Logs + Save

for Filter your logs

up into Patterns × Show Count of all logs by Status × and Service × +

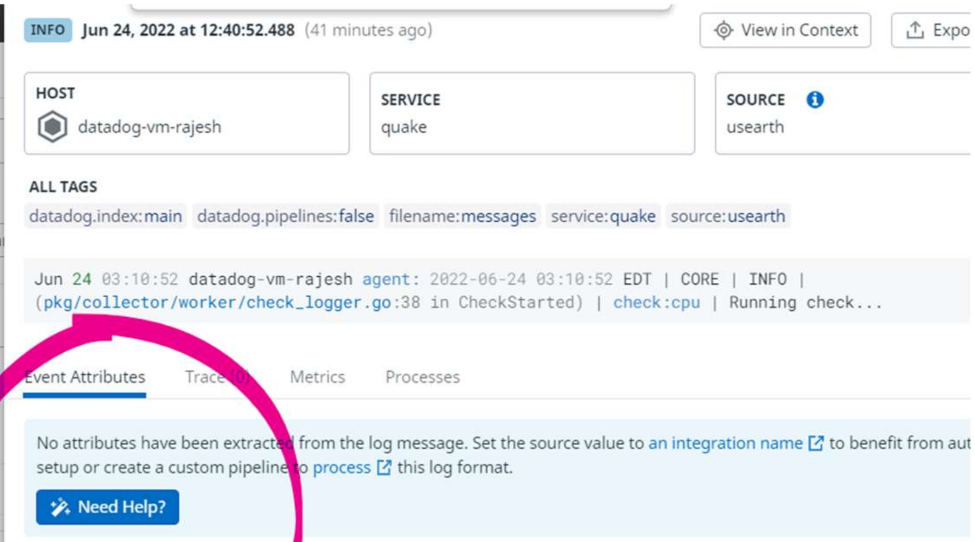
alize as List Timeseries Top List Table Tree Map NEW Pie Char

09:30 09:45 10:00 10:15 10:30 10:45 11:00

Hide Controls 409 patterns found (based on 1,307 logs)

Watchdog Insights

COUNT	~VOLUME	STATUS	SERVICE
103	66	info	quake
43	42	info	quake
39	39	info	quake
36	21	info	quake



INFO Jun 24, 2022 at 12:40:52.488 (41 minutes ago) View in Context Expo

HOST: datadog-vm-rajesh

SERVICE: quake

SOURCE: usearth

ALL TAGS

datadog.index:main datadog.pipelines:false filename:messages service:quake source:usearth

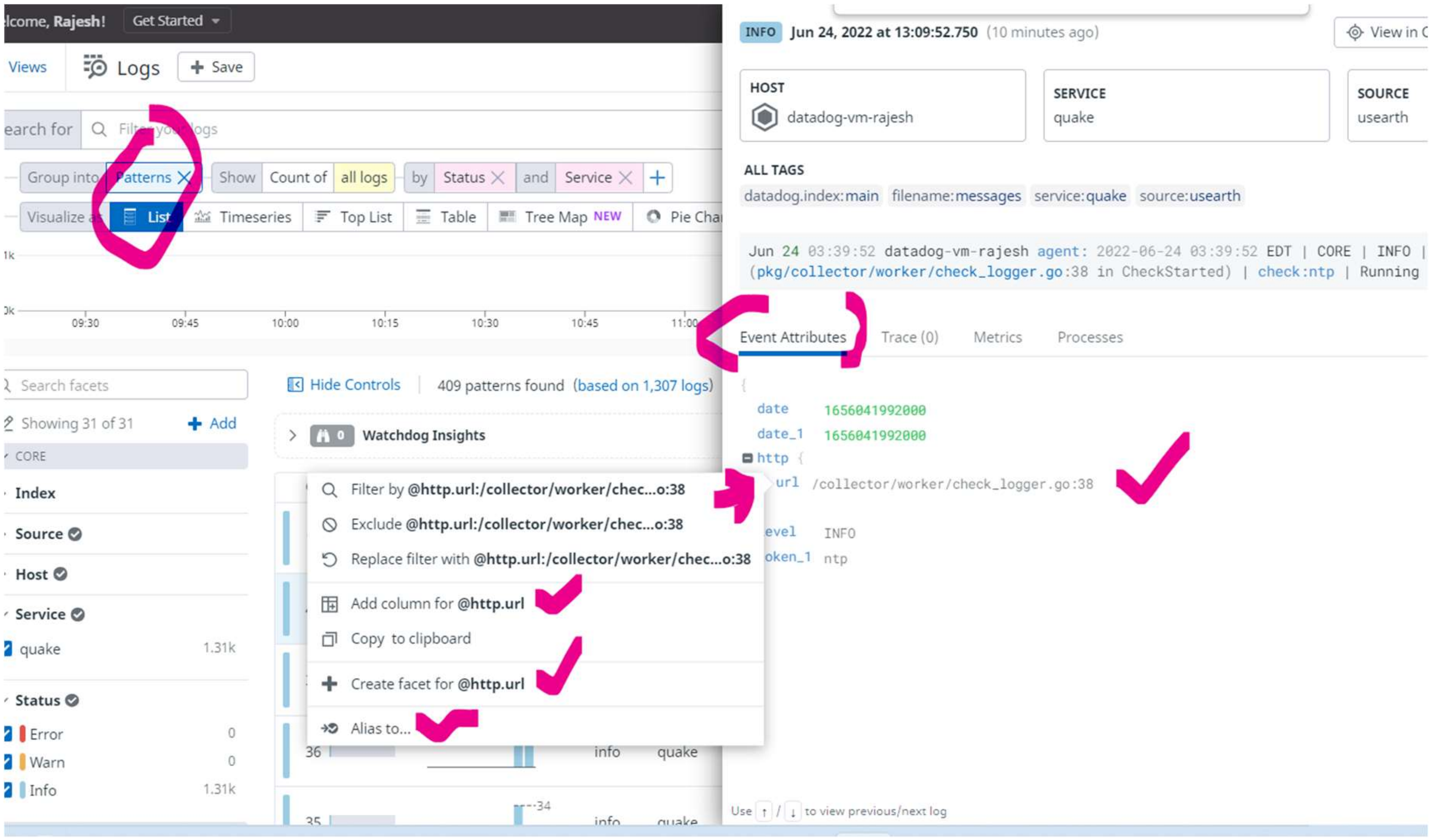
```
Jun 24 03:10:52 datadog-vm-rajesh agent: 2022-06-24 03:10:52 EDT | CORE | INFO |
(pkg/collector/worker/check_logger.go:38 in CheckStarted) | check:cpu | Running check...
```

Event Attributes Traces (0) Metrics Processes

No attributes have been extracted from the log message. Set the source value to an integration name to benefit from aut setup or create a custom pipeline to process this log format.

Need Help?

Step 9 – Results - After



lcome, Rajesh! Get Started

Views Logs + Save

earch for Filter your logs

Group into Patterns ✕ Show Count of all logs by Status ✕ and Service ✕ +

Visualize as List Timeseries Top List Table Tree Map NEW Pie Cha

Search facets

Showing 31 of 31 + Add

CORE

Index

Source

Host

Service

quake	1.31k

Status

Error	0
Warn	0
Info	1.31k

Hide Controls 409 patterns found (based on 1,307 logs)

Watchdog Insights

Filter by @http.url:/collector/worker/chec...o:38

Exclude @http.url:/collector/worker/chec...o:38

Replace filter with @http.url:/collector/worker/chec...o:38

Add column for @http.url

Copy to clipboard

Create facet for @http.url

Alias to...

INFO Jun 24, 2022 at 13:09:52.750 (10 minutes ago) View in C

HOST	SERVICE	SOURCE
datadog-vm-rajesh	quake	usearth

ALL TAGS

datadog.index:main filename:messages service:quake source:usearth

```
Jun 24 03:39:52 datadog-vm-rajesh agent: 2022-06-24 03:39:52 EDT | CORE | INFO |
(pkg/collector/worker/check_logger.go:38 in CheckStarted) | check:ntp | Running
```

Event Attributes Trace (0) Metrics Processes

```
{
  date     1656041992000
  date_1   1656041992000
http {
  url  /collector/worker/check_logger.go:38

evel    INFO
oken_1  ntp
```

Use ↑ / ↓ to view previous/next log