Getting logs and metrics into metricstore

Murugan Kandaswamy, Senior Software engineer

Allan Yan, Principal Software engineer

Forward-Looking Statements

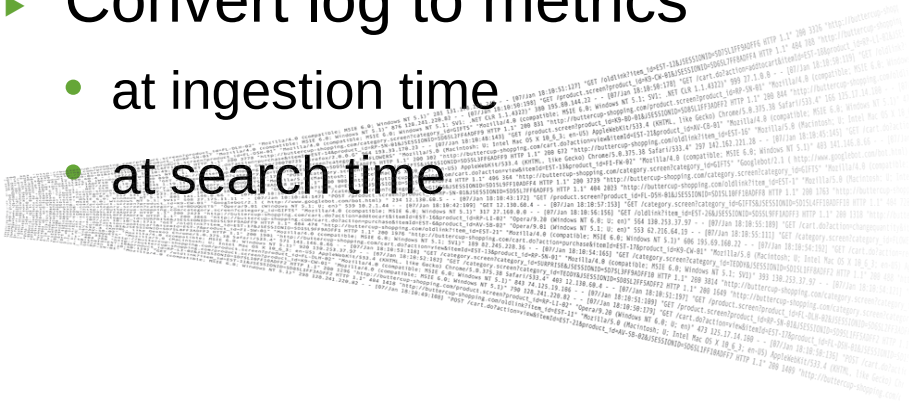
During the course of this presentation, we may make forward-looking statements regarding future events or the expected performance of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results could differ materially. For important factors that may cause actual results to differ from those contained in our forward-looking statements, please review our filings with the SEC.

The forward-looking statements made in this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, this presentation may not contain current or accurate information. We do not assume any obligation to update any forward-looking statements we may make. In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionality described or to include any such feature or functionality in a future release.

splunk> .conf18

Agenda

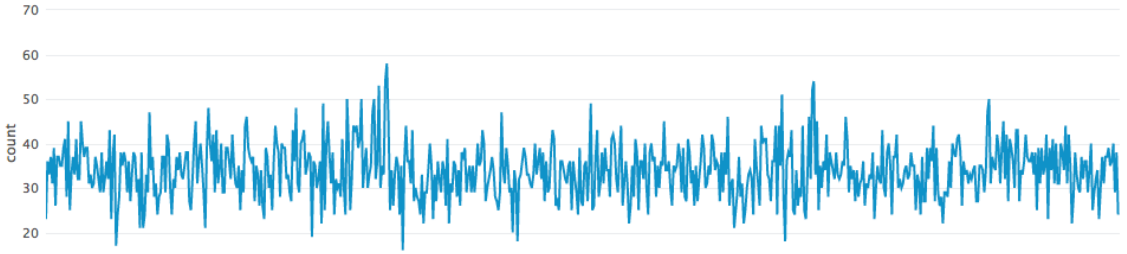
- Why metrics
  - Traditional way
  - New way
- Metrics data model
- Various ways to GDI for metrics
  - Configurations
  - Deployment topologies
- Convert log to metrics
  - at ingestion time
  - at search time



Why Metrics?

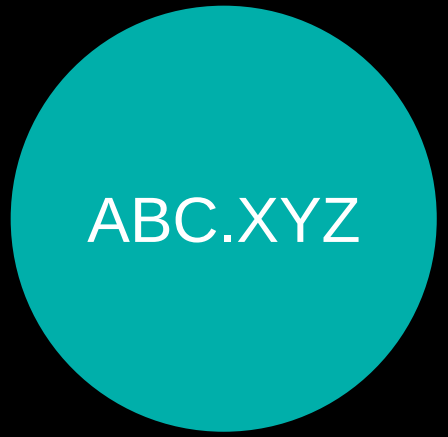
- **Logs**
  - Unstructured data
  - Needle in the haystack
  - Can tell you all about the “why”
  - Answers questions you might not even have yet
  - Very versatile

- **Metrics**
  - Structured Data
  - High volume
  - Easy way to do monitoring
  - You know what you want to measure
  - e.g. performance, CPU, Number of users, memory used, network latency, disk usage



Terminology - What is a Metric data point?

Time



Metric Name

`system.cpu.idle`

Measure

numeric data point

Dimensions

Host (10.1.1.100, web01.splunk.com)

Region (e.g., us-east-1, us-west-1, us-west-2, us-central1)

IntanceTypes (e.g., t2.medium, t2.large, m3.large)

Metric Time Series Examples

Server A

Server B

Dimensions				
metric_name	host	app	_time	_value
cpu.idle	A	foo	1	4.2
mem,free	B	bar	2	7.3
cpu.idle	A	foo	3	8.4
mem.free	A	baz	4	32

Different Colors Represent Distinct **Metric Time Series**. Each Row is a Single **Metric Data Point**, made up of a timestamp, measurement and a set of required and optional dimensions

Both of these data points belong to the same **Metric Time Series** because they share the exact same set of required and optional dimension key/value pairs:

metric_name=cpu.idle
host=A
app=foo

“Splunk provides one platform to analyze and investigate across both events and metrics”

Metrics – Before Splunk 7.0

- 06/29/2017 16:45:15.170 collection="Available Memory" object=Memory counter="Pages/sec" Value=264 host=10.0.8.156
- 06/29/2017 16:47:47.170 collection="MSExchangeIS_Mailbox" object="MSExchangeIS Mailbox" counter="Messages Submitted/sec" instance="_Total" Value=185.3656 host=10.0.8.156

Timestamp
Metric Name
Measurement Value
Dimensions

Metrics – Current Way

Ingest metrics natively

Metric Store

Ability to ingest and store metric measurements at scale

`mstats`

`tstats` equivalent to query time series from metrics indexes

Metrics Catalog

mcatalog and REST APIs to query lists of ingested metrics and dimensions

splunk> .conf18

Schema of a Metric Index

Field	Required	Description	Example
metric_name	•	The metric name.	os.cpu.user
_time	•	The timestamp of the metric in UNIX time notation.	
_value	•	The numeric value of the metric.	42.12345
<dim0>...<dimN>		**An arbitrary number of dimensions.**	**e.g. ip=10.2.1.166**
_dims	•	Dimension names. Dimensions indicate how metrics are split. Internal, should not be changed	
host	•	The origin host.	
index	•	The metrics index name.	
sourcetype	•	The data structure of the metric.	
source		The source of the metrics data.	

Blue = Internal or not directly writable

Getting Data In

Supported Approaches

4 ways to get metrics into Splunk

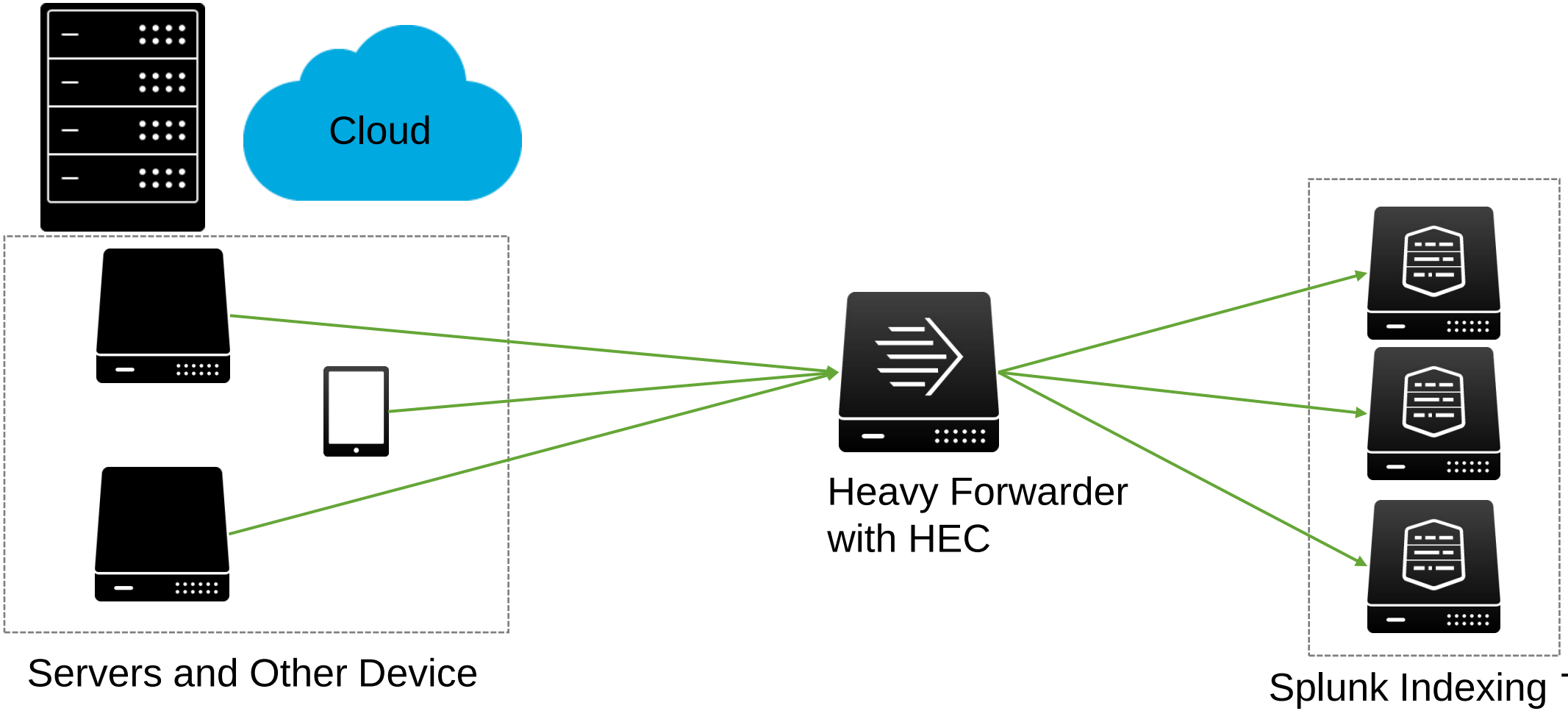
- HEC
- Statsd
- Collectd
- Log to Metrics

Quick Overview HEC

- use the HEC /collector REST API endpoint
  - Splunk host machine (IP address, host name, or load balancer name)
  - HTTP port number
  - HEC token value
  - Metrics data payload in JSON format
- Payload schema
  - Requires fields: **metric_name** and **_value** under **fields** field and **event** field set to "metric”.
  - Optional fields: **time**, **host**, **source**, **sourcetype**, and other dimension key/value pairs under fields field.
  - If **time** field is set, it must be in epoch time format.
- Example
  - curl -k https://localhost:**8088**/services/collector \ -H "Authorization: Splunk **b0221cd8-c4b4-465a-9a3c-273e3a75aa29**" -d '{"time": 1486683865.000,"event":"metric",,"fields":{"region":"us-west-1","datacenter":"us-west-1a","rack":"63", "_value":1099511627776,"metric_name":"total"}}'

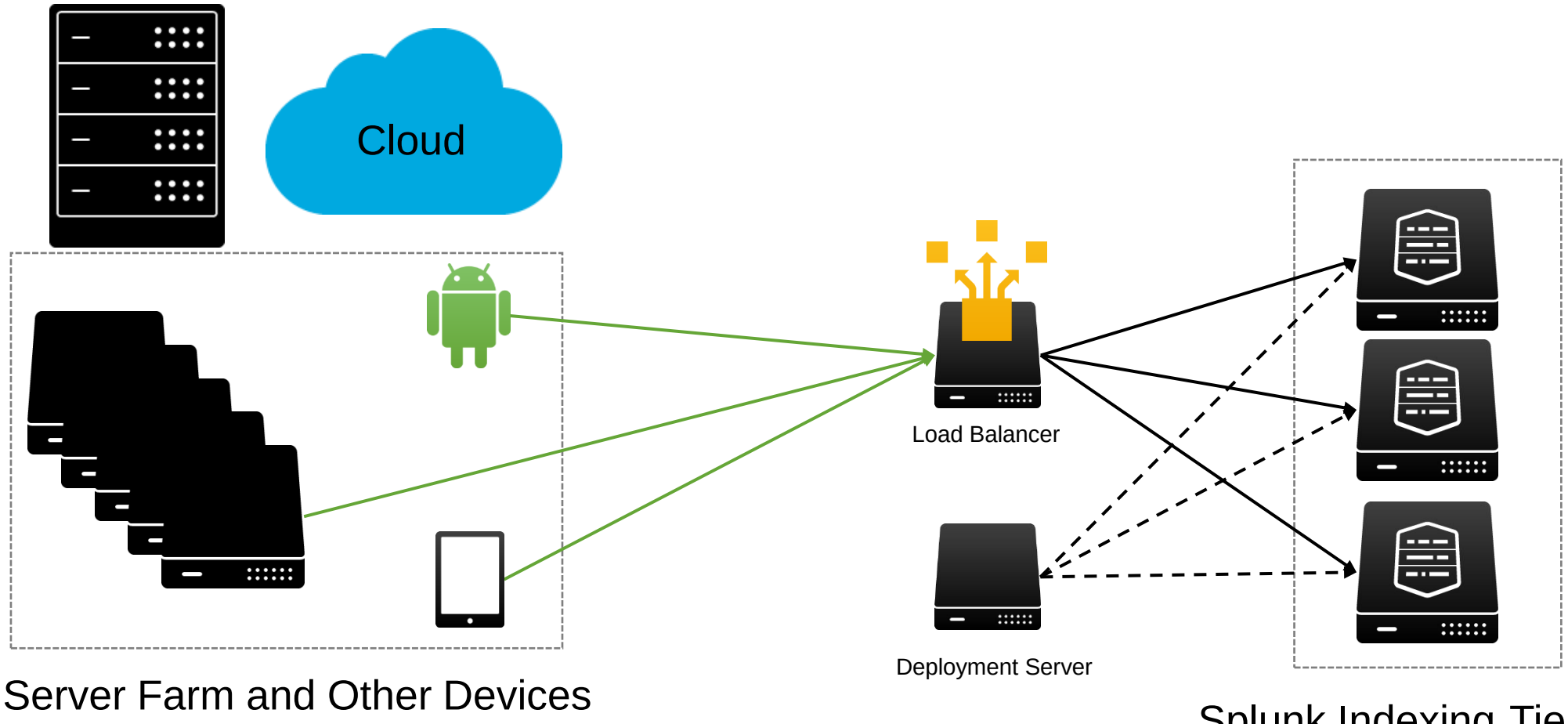
HEC Deployment Scenario 1

One HEC server, pool of indexers



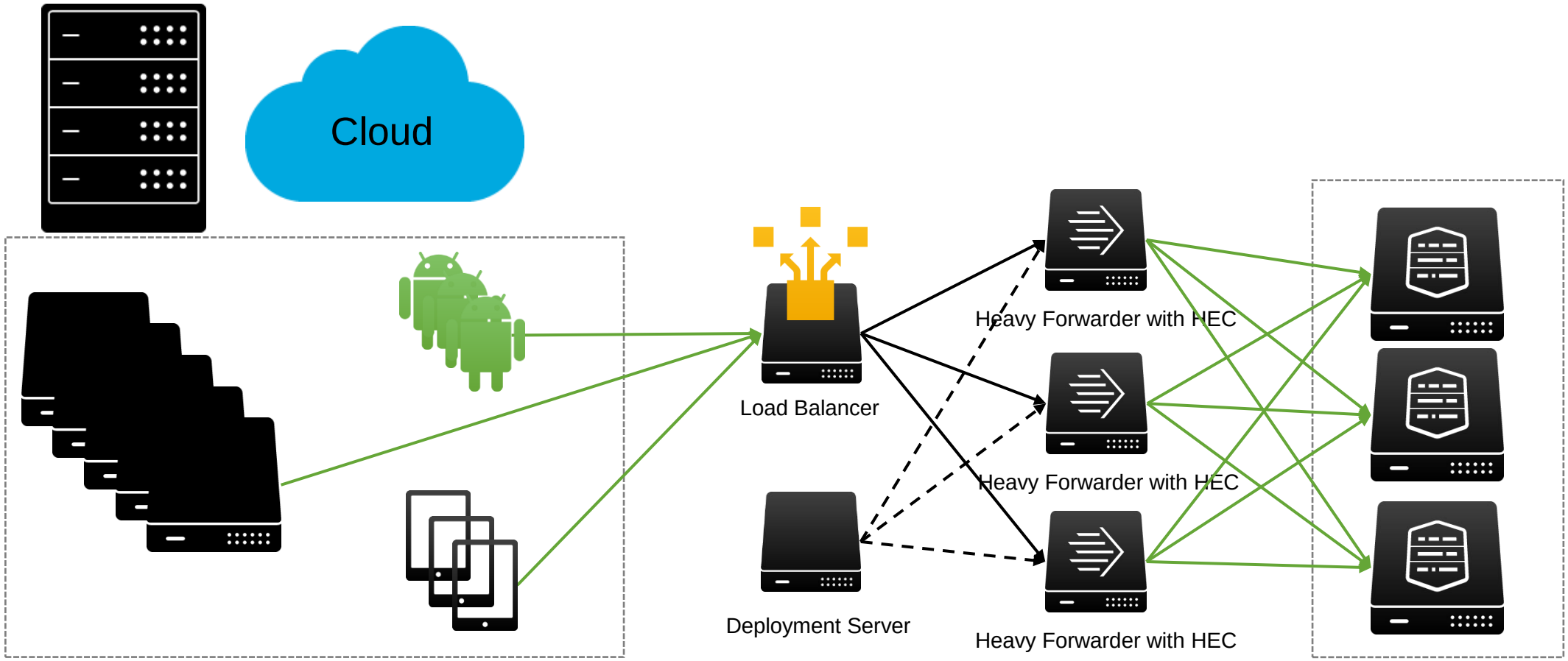
HEC Deployment Scenario 2

load balancer, no forwarder, pool of indexers, using deployment server



HEC Deployment Scenario 3

load balancer, multiple HEC instances with forwarders distributing the data to indexers



Quick Overview StatsD / Collectd

StatsD

- StatsD is a open source daemon that runs on the Node.js platform
- Primarily used to measure performance of **application code**
- Introduces statsD line metric protocol, often sent to UDP/TCP

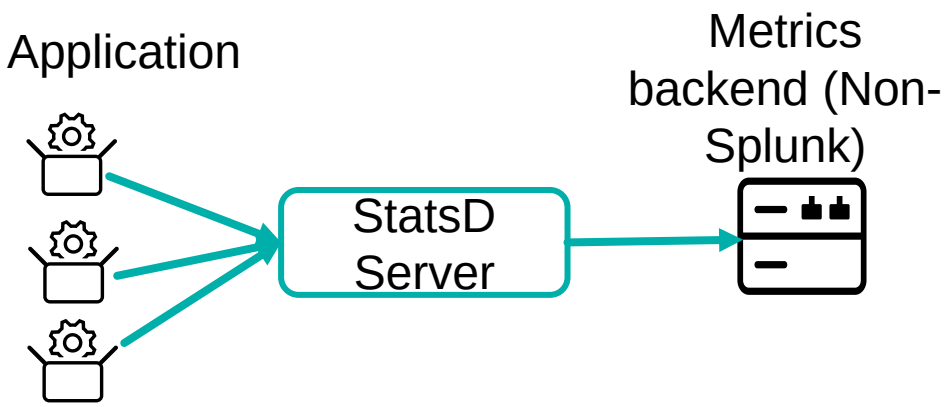
Collectd

- collectd is an open source daemon that collects performance metrics from a variety of sources.
- Primarily used to measure **system performance** (e.g. CPU, memory, disk, network etc)
- Can send data to various endpoint, e.g. HTTP(S)

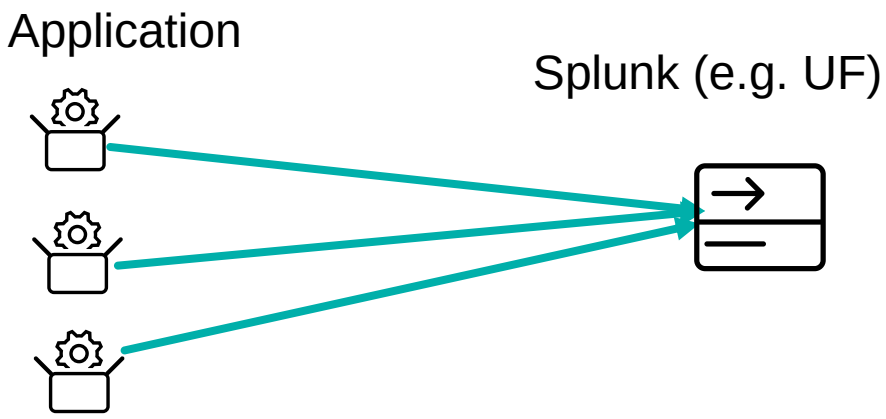
StatsD

- E.g. Instrumenting application code to track performance
- StatsD client libraries available in many programming languages
- “Fire and forget” via UDP

Traditional setup with StatsD



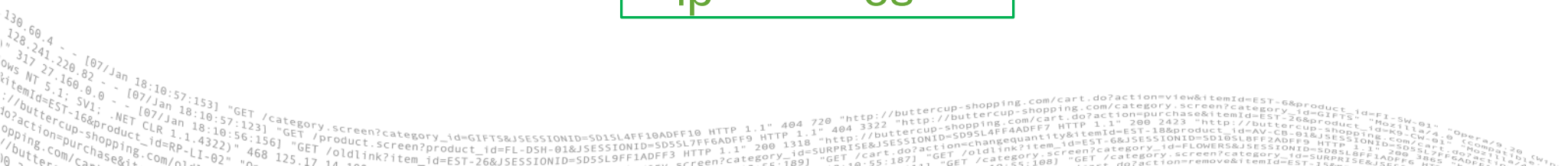
StatsD with Splunk



StatsD Protocol: All Variants are supported

StatsD sourcetype supports 3 different formats

1. StatsD line metric protocol:
   - metricname:value|type
   - **Example**
     performance.os.disk:1099511627776|g
2. StatsD support with Dimensions (Adjusted metric protocol)
   - metricname:value|type|#dim1:value1,dim2:value2
   - **Example**
     performance.os.disk:1099511627776|g|#region:us-west-1,datacenter:us-west-1a,rack:63,os:Ubuntu16.10,arch:x64,team:LON,service:6,
     service_version:0,service_environment:test,path:/dev/sda1,fstype:ext3
3. StatsD support with dimensions encoded in metric name (next slide)
   - **Example**
     mem.percent.used.10.2.3.4.windows:33|g
     ip os



StatsD Dimension extraction (cont'd)

- E.g. mem.percent.used.10.2.3.4.windows:33|g

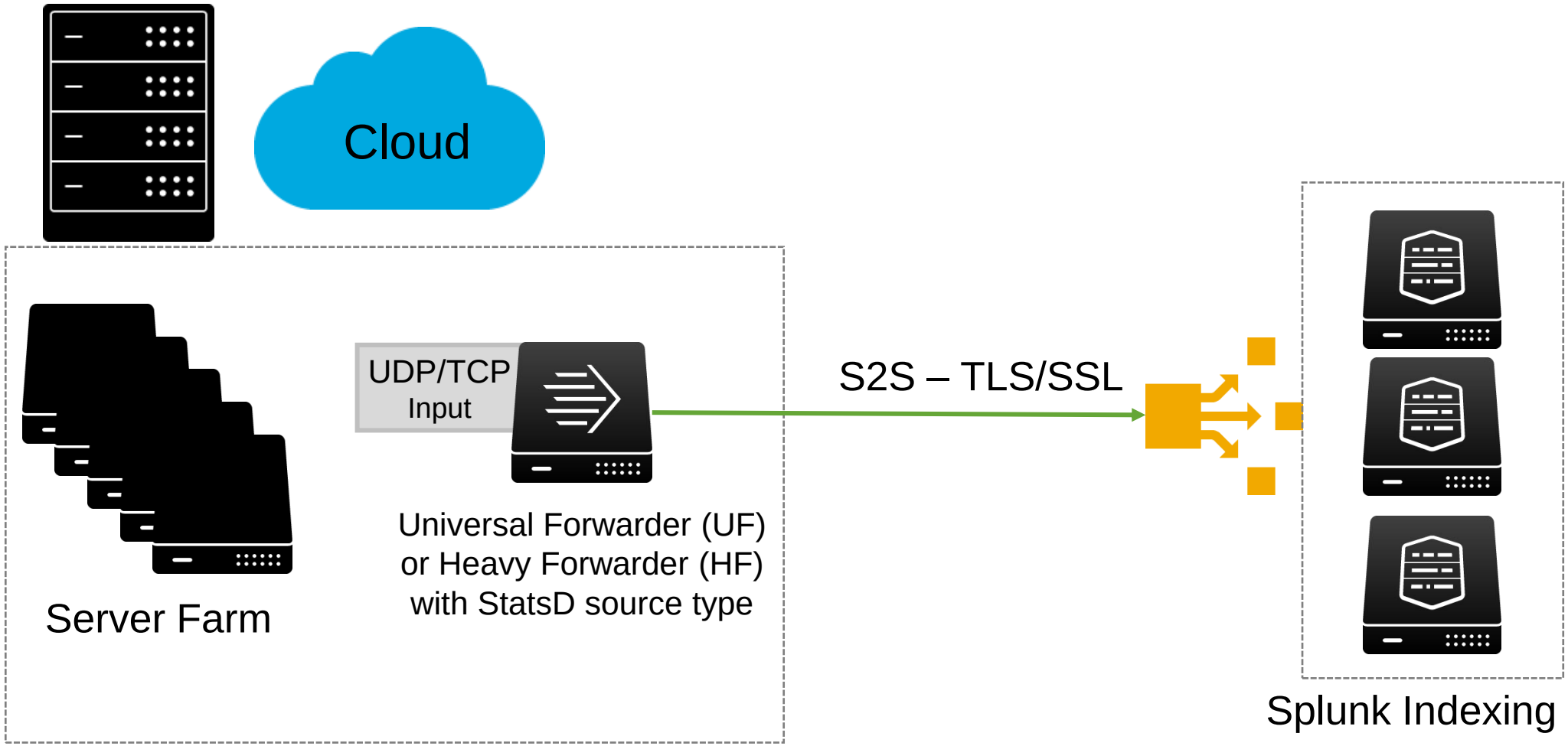
```
# props.conf

[my_custom_metrics_sourcetype]
METRIC_PROTOCOL = statsd
STATSD-DIM-TRANSFORMS = statsd-dims:my_custom_metrics_transform
```

```
# transforms.conf

[statsd-dims:my_custom_metrics_transform]
REGEX = (?<ip>\d{1,3}.\d{1,3}.\d{1,3}.\d{1,3})\.(?<os>\w+): REMOVE_DIMS_FROM_METRIC_NAME = true
```

Statsd Deployment



Collectd

- E.g. Tracking infrastructure performance (CPU, Memory, Network, Disk etc)
- ~100 frontend plugins
- Send to HEC via write_http plugin
- https://collectd.org

- **Example Frontend Plugins**
  - CPU
  - df
  - Disk
  - Interface
  - Load
  - Memory
  - Network
  - Protocols
  - Swap
  - Tcpconns
  - Thermal
  - Uptime
- **Relevant Backend Plugins**
  - write_http

GDI: collectd write_http plugin

- Sample write_http event

{
"values":[**98.93638411944**],
"dstypes":["**gauge**"],
"dsnames":["**value**"],
"time":1474401106.556,
"interval":10.000,
"host":"C5819124-66AE-4B28-8E13-914C3961E46C",
"plugin":"cpu",
"plugin_instance":"0",
"type":"**cpu**",
"type_instance":"**idle**”
}

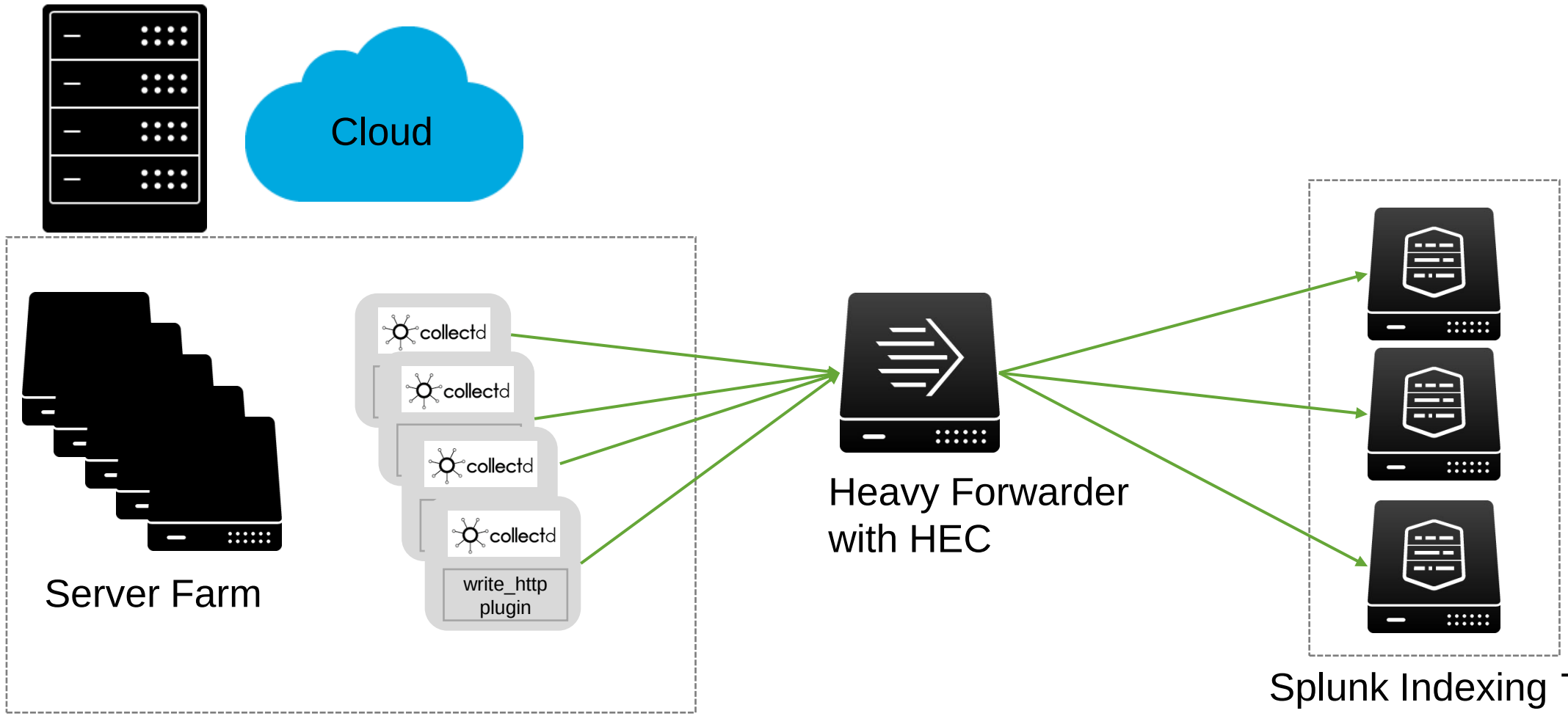
- Sample Result
  - metric_name = **cpu.idle.value**
  - _value = **98.93638411944**
  - **plugin_instance** = **0** (=CPU core # 0)
  - **metric_type** = **gauge**

plugin_instance is currently the only dimension extracted in addition to the default available dimensions
host, source, sourcetype, index

splunk> .conf18

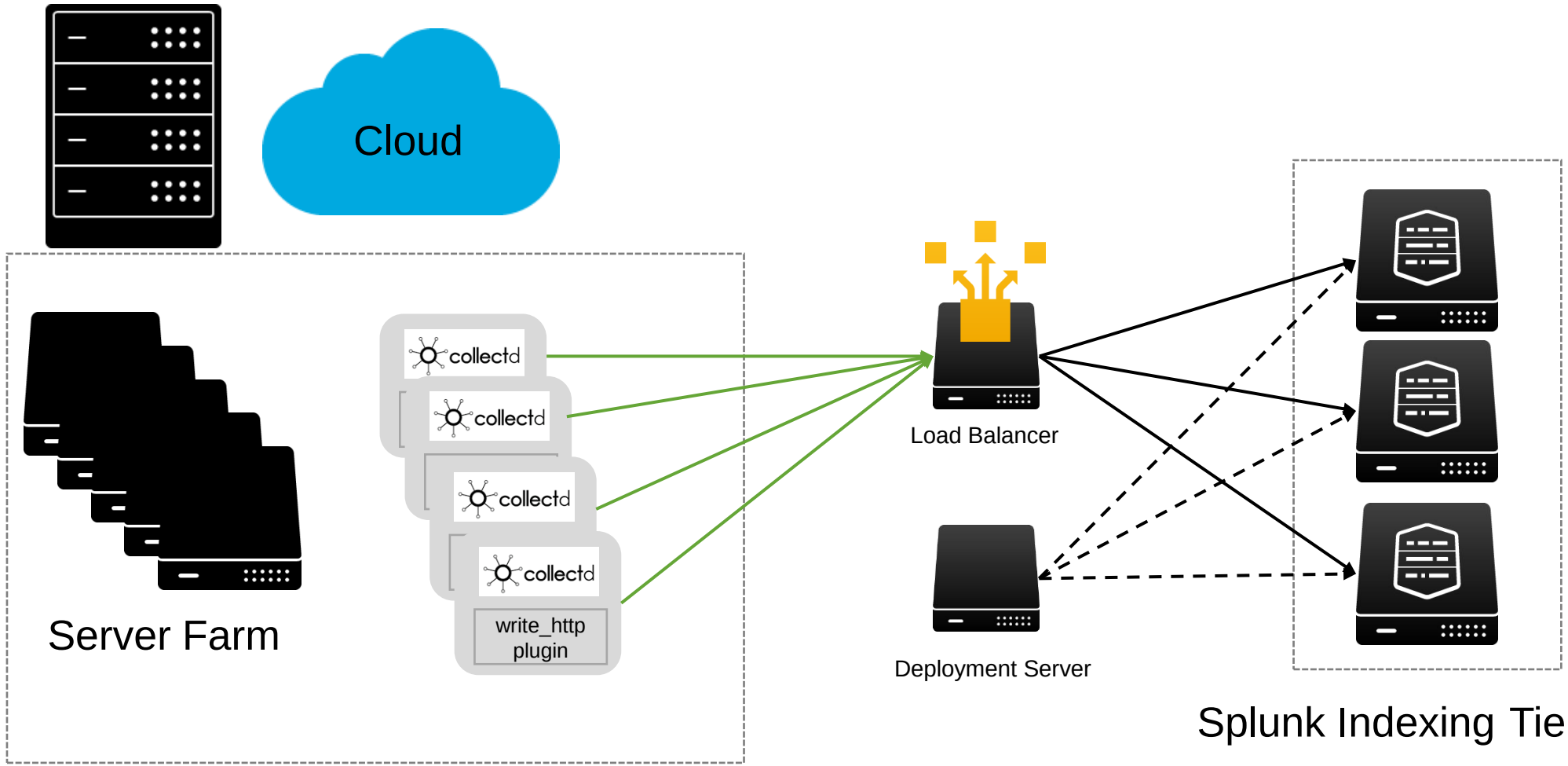
Collectd Deployment Scenario 1

One HEC server, pool of indexers



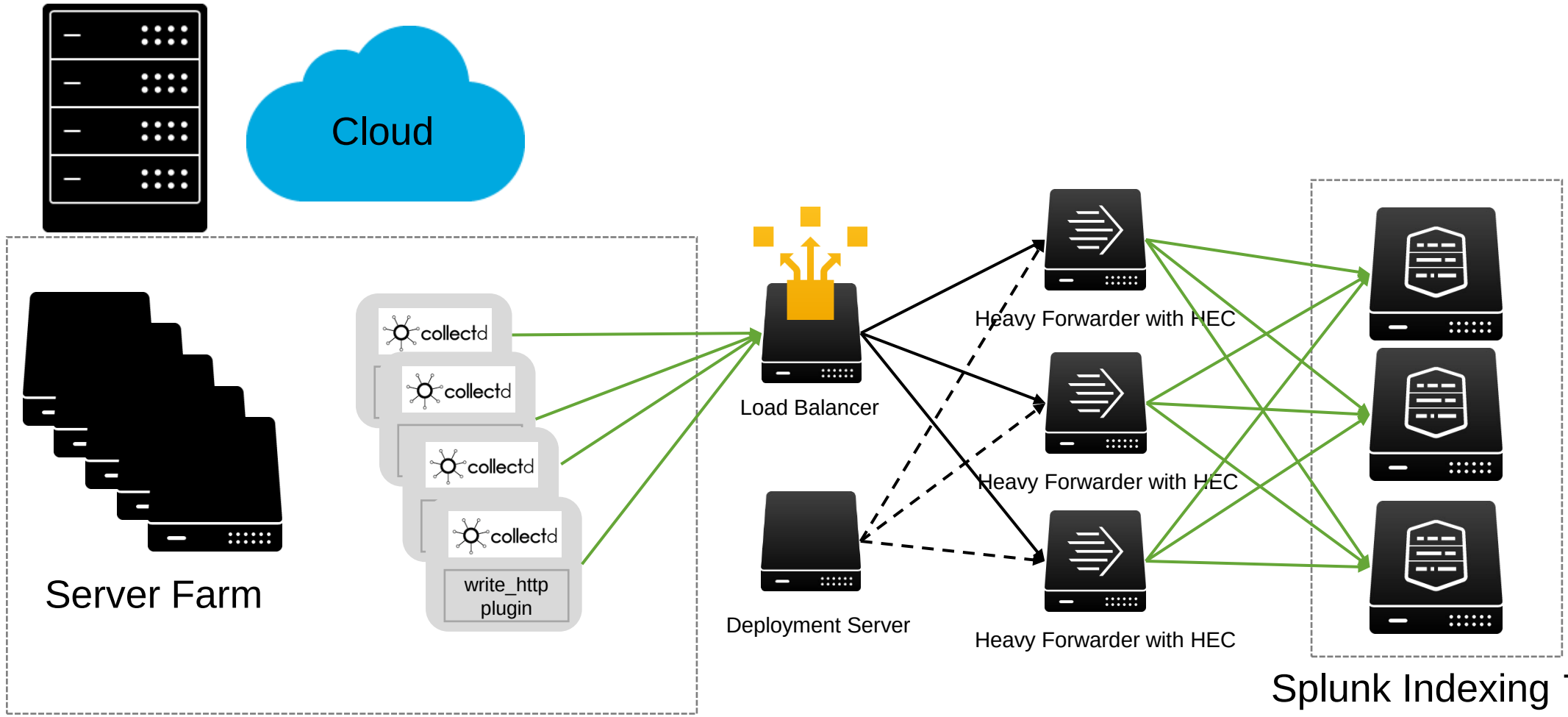
Collectd Deployment Scenario 2

load balancer, no forwarder, pool of indexers, using deployment server



Collectd Deployment Scenario 3

load balancer, multiple HEC instances with forwarders distributing the data to indexers



Why Log to Metrics?

Ingest log natively as metrics

- Logs which contain metric data like windows performance monitor typically has multiple metrics per log line.
- Before Log to Metrics feature, there was no way of extracting multiple metrics from an event even by defining custom source types(i.e props/transforms configuration).
- Log to metrics feature enables user to extract multiple metrics from an event.

Log to Metrics

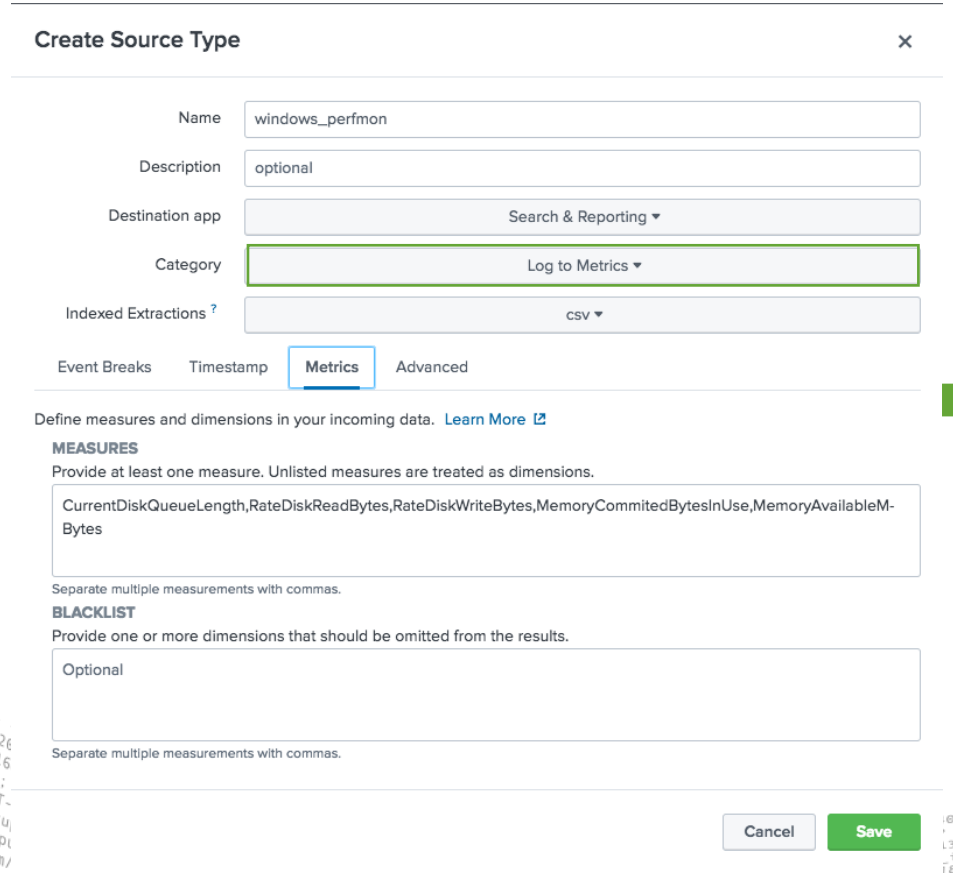
- Log to Metrics is available in Splunk Enterprise 7.2 and Splunk Cloud.
- This enables user to natively ingest log sources which has multiple metrics per event directly into metric store.
- User has to list all the measurements to be indexed in metric store.
- Also, Splunk lets user discard some unnecessary high cardinality dimensions present in the log data.

How to ingest Log as Metrics?

- Sample CSV performance monitoring data:

```
"Timestamp",            "CurrentDiskQueueLength","RateDiskReadBytes","RateDiskWriteBytes","MemoryCommittedBytesInUse","MemoryAvailableMBytes"
"08/23/2018 01:07:44.922","0",                   "0",               "7787.6846449231989","28.924132238357359",      "2527"
"08/23/2018 01:07:54.931","0",                   "324588.9108980139","126919.63560882151","28.930770565972374",      "2524"
"08/23/2018 01:08:04.927","0",                   "0",               "219209.36279744742","28.928099975205978",      "2526"
```

- New Create Sourcetype UI workflow(with Log to Metrics):




```
root@so1:/opt/splunk# cat etc/apps/search/local/props.conf
[windows_perfmon]
DATETIME_CONFIG =
FIELD_QUOTE = "
INDEXED_EXTRACTIONS = csv
METRIC-SCHEMA-TRANSFORMS = metric-schema:windows_perfmon_1533521596641
NO_BINARY_CHECK = true
category = Log to Metrics
pulldown_type = 1
root@so1:/opt/splunk#
root@so1:/opt/splunk# cat etc/apps/search/local/transforms.conf
[metric-schema:windows_perfmon_1533521596641]
METRIC-SCHEMA-MEASURES = CurrentDiskQueueLength,RateDiskReadBytes,RateDiskWriteBytes,MemoryCommittedBytesInUse,MemoryAvailableMBytes
root@so1:/opt/splunk#
```

splunk> .conf18

How to ingest sophisticated log data as metrics?

- A sample key-value log data below has the two sets of measurements.

Log with 2 sets of measurements

```
04-08-2018 00:57:21.500 -0700 INFO  Metrics - group=queue, location=sf, corp=splunk, name=udp_queue, max_size_kb=0, current_size_kb=0, current_size=0, largest_size=0, smallest_size=0
04-08-2018 00:57:21.500 -0700 INFO  Metrics - group=queue, location=sf, corp=splunk, name=aggqueue, max_size_kb=1024, current_size_kb=1, current_size=5, largest_size=35, smallest_size=0
04-08-2018 00:57:21.500 -0700 INFO  Metrics - group=queue, location=sf, corp=splunk, name=auditqueue, max_size_kb=500, current_size_kb=0, current_size=0, largest_size=1, smallest_size=0
04-08-2018 00:57:52.492 -0700 INFO  Metrics - group=pipeline, name=indexerpipe, processor=indexin, cpu_seconds=0, executes=171, cumulative_hits=2214401
04-08-2018 00:57:52.492 -0700 INFO  Metrics - group=pipeline, name=indexerpipe, processor=index_thruput, cpu_seconds=0, executes=171, cumulative_hits=2214401
04-08-2018 00:57:52.492 -0700 INFO  Metrics - group=pipeline, name=indexerpipe, processor=indexandforward, cpu_seconds=0, executes=171, cumulative_hits=2214401
```

- How to generate unique metric_name prefix for logs without it?
  - Use Ingest time eval feature(Also, available in Splunk Enterprise 7.2).
  - Ingest eval can be used to remap the field name ‘group::<value>’ as ‘metric_name::<value>’.

```
root@so1:/opt/splunk# cat etc/system/local/props.conf
[metrics_log]
METRIC-SCHEMA-TRANSFORMS = metric-schema:extract_metrics
TRANSFORMS-metricslog = field_extraction,eval_pipeline
root@so1:/opt/splunk#
root@so1:/opt/splunk# cat etc/system/local/transforms.conf
[eval_pipeline]
INGEST_EVAL = metric_name=group

[metric-schema:extract_metrics]
METRIC-SCHEMA-MEASURES-queue = max_size_kb,current_size_kb,current_size,largest_size,smallest_size
METRIC-SCHEMA-BLACKLIST-DIMS-queue = location,corp
METRIC-SCHEMA-MEASURES-pipeline = cpu_seconds,executes,cumulative_hits
root@so1:/opt/splunk#
```

splunk> .conf18

Verify Ingested Log to Metrics data

- Run **mcatalog** command to verify all measurements and dimensions got indexed into metricstore.

```
|mcatalog values(metric_name) where index=metrics_log
```

✓ 24 events (before 9/4/18 7:27:27.000 AM) No Event Sampling

Events Patterns **Statistics (1)** Visualization

20 Per Page Format Preview

values(metric_name)
pipeline.cpu_seconds
pipeline.cumulative_hits
pipeline.executes
queue.current_size
queue.current_size_kb
queue.largest_size
queue.max_size_kb
queue.smallest_size

```
|mcatalog values(_dims) where index=metrics_log
```

✓ 24 events (before 9/4/18 7:45:38.000 AM) No Event Sampling

Events Patterns **Statistics (1)** Visualization

20 Per Page Format Preview

values(_dims)
group
name
processor

splunk> .conf18

Log to Metrics REST API

- New REST endpoint: /services/data/transforms/metric-schema
- This endpoint lets you configure multiple measures and dimensions to be blacklisted for transforming log to metrics data
- Supported POST request params

Parameter	Details
name	name of the metric schema stanza
field_names	comma separated list of measures to be extracted from an event
blacklist_dimensions (optional)	comma separated list of dimensions to be blacklisted from an event
metric_name_prefix (optional)	used when each event has different set of field_names and blacklist_dimensions. If events such as CSV data have the same set of field_names and blacklist dimensions, then metric_name_prefix can be ignored.

Log to Metrics – search time

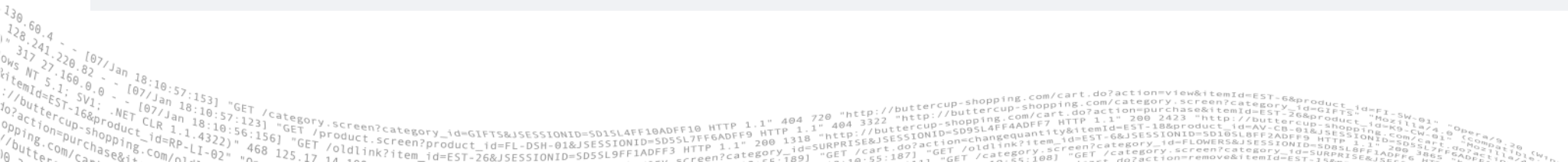
- search command **mcollect** and **meventcollect** to re-ingest already indexed log events into metrics index
  - Similar to summary indexing using `collect`
  - No additional license cost
  - use **meventcollect** for simple events search
  - use **mcollect** turning report output to metrics (e.g. `stats, timechart`)
- Example

```
index=mylog ERROR | stats count BY type | rename count AS _value type AS metric_name | mcollect index=mymetrics
```

All time

```
index=mylog | eval prefix = group + "." + name | meventcollect index=my_metric_index split=true prefix_field=prefix name group
```

All time



splunk> .conf18

Key Takeaways

Approach (sourcetype/category/search command)	Use case
HEC (http_event_collector_metrics)	preformatted JSON data
Statsd (statsd)	monitoring application performance
Collectd (collectd_http)	monitoring system performance
Log to Metrics (category)	extract one or more measurements from an event at index time
Use search command (mcollect)	extract one or more measurements from an event at search time

Thank You

Don't forget to rate this session in the .conf18 mobile app

Q&A

splunk> .conf18