EPL660: Information Retrieval and Search Engines – Lab 7

University of Cyprus
Department of Computer Science

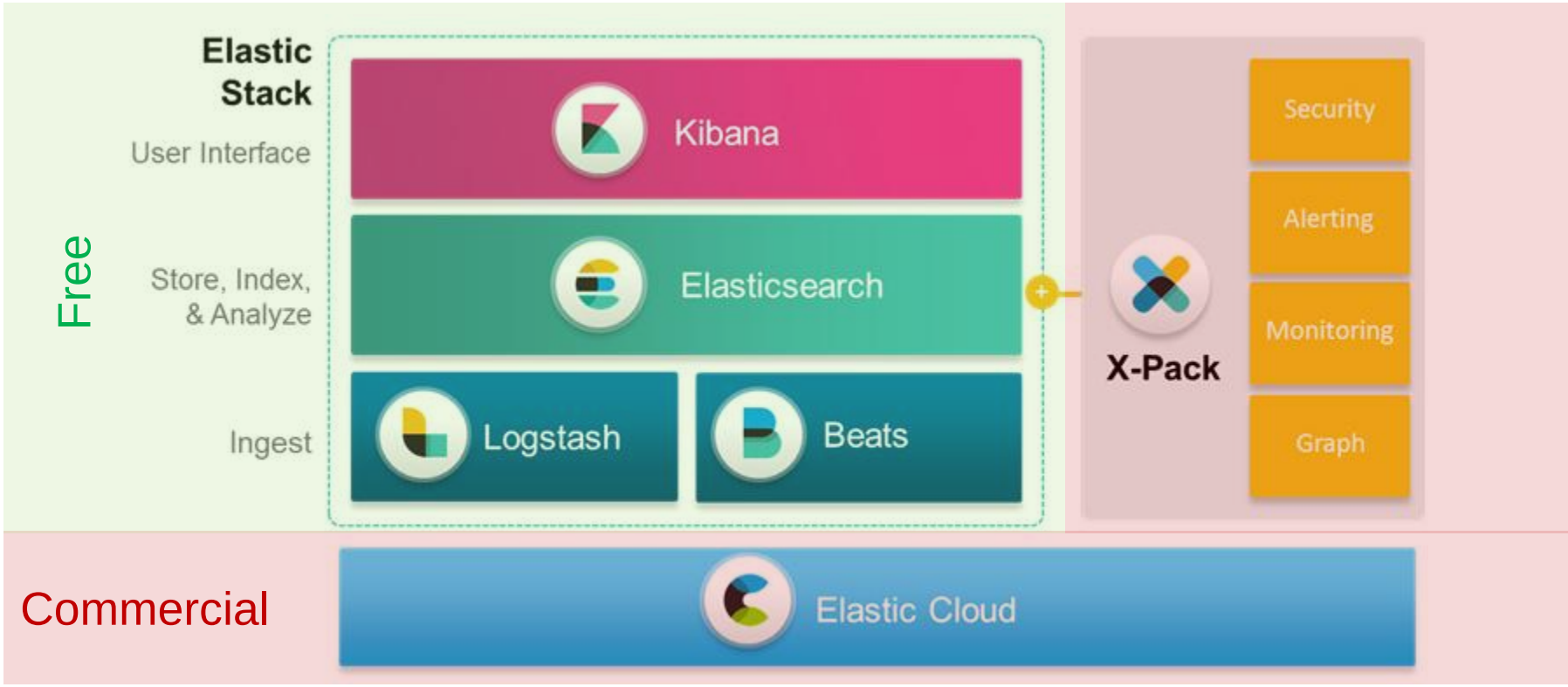
Παύλος Αντωνίου

Γραφείο: B109, ΘΕΕ01

Elasticsearch

- Free, open-source, document (json) oriented search and analytics engine built on top of Apache Lucene
- Developed in Java (cross-platform)
- Distributed (cloud-based)
- Scalable and highly available
- Java API + RESTful HTTP/JSON API
- Used for full-text search, structured search, analytics, or all three in combination
- Near Real-Time searching
  - slight latency (~ 1 sec) from the time you index a document until the time it becomes searchable

Elastic Stack

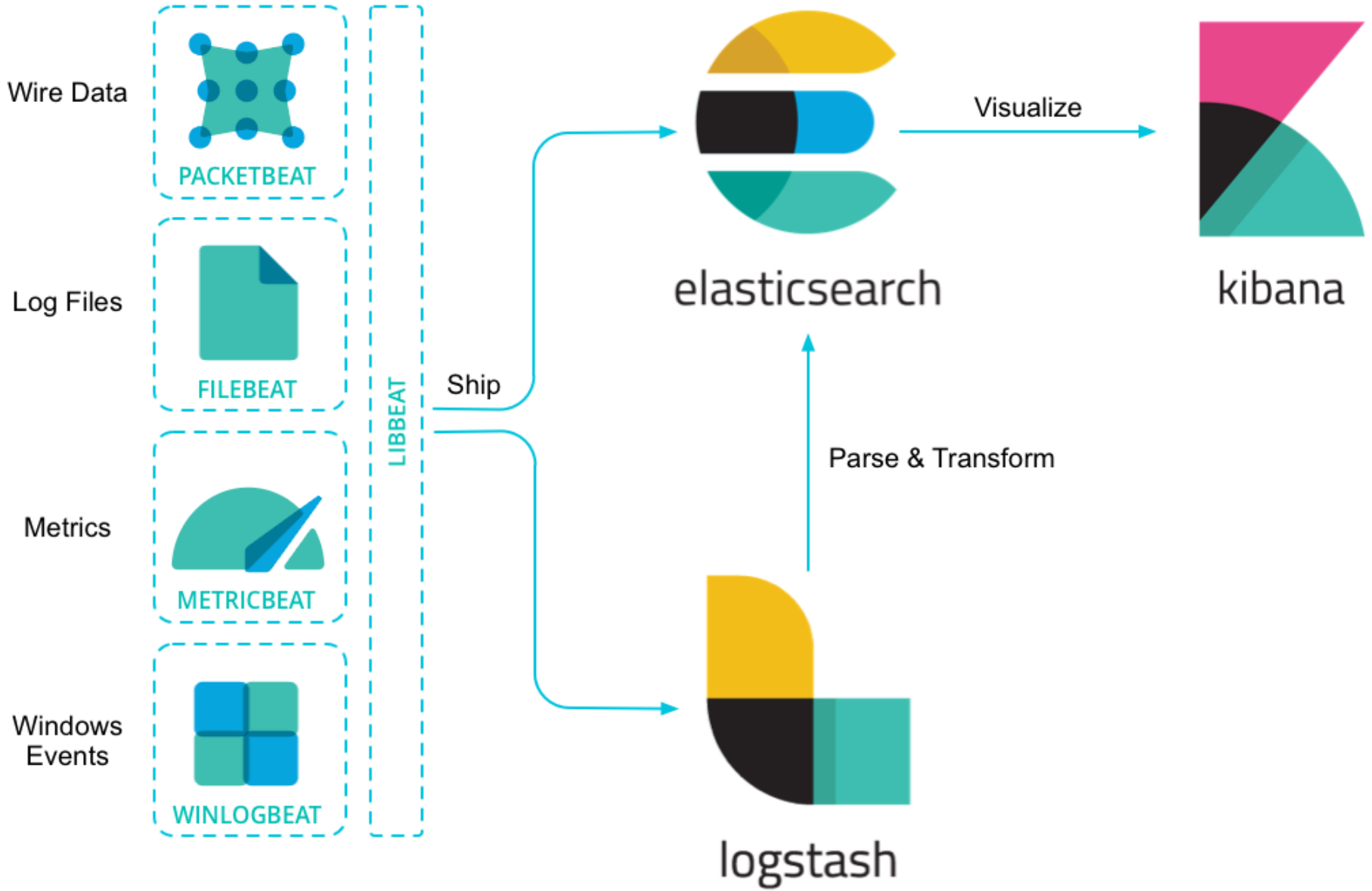


Elasticsearch is a search and analytics engine.

Rest Elastic Stack components

- kibana
  - Window into elastic stack. Enables visual exploration and real-time analysis of data in Elasticsearch
- logstash
  - Central dataflow engine for gathering, enriching, and unifying input data from various sources (beats, files, REST API) regardless of format or schema and sends it to your favorite stash (elasticsearch, files, email, REST API endpoint, tcp socket, mongodb, see more)
- beats
  - Forward host-based metrics and any data to Elasticsearch or Logstash
- x-pack
  - A single extension for Security for Elastic Stack, Alerting and notifications for the Elastic Stack, Monitoring for the Elastic Stack, Real-time Graph Analytics to enable new use cases for Elastic Stack
- cloud
  - Hosted Elasticsearch & Kibana on AWS and GCP

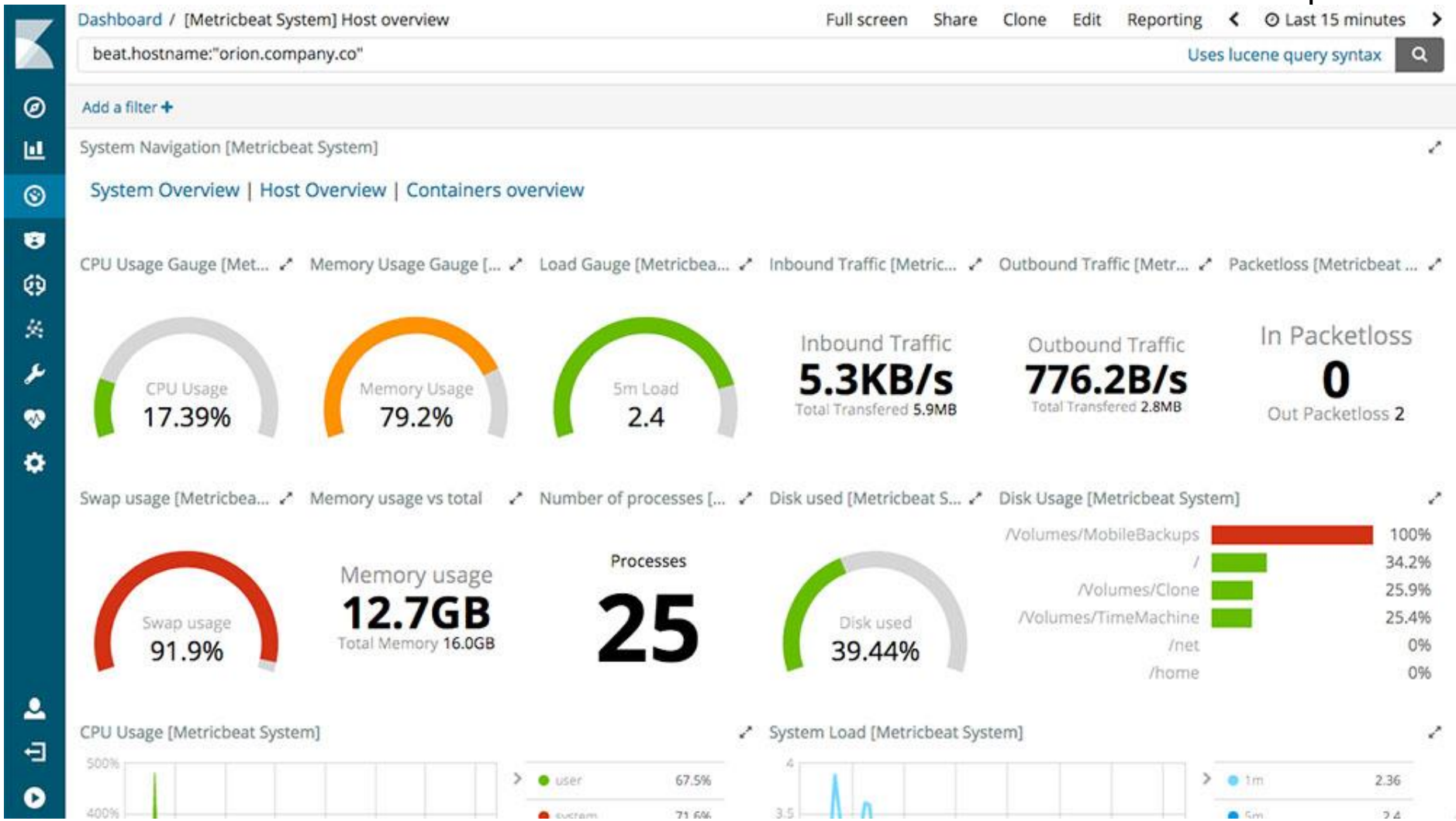
Elastic Stack in action



Kibana & Packetbeat in action

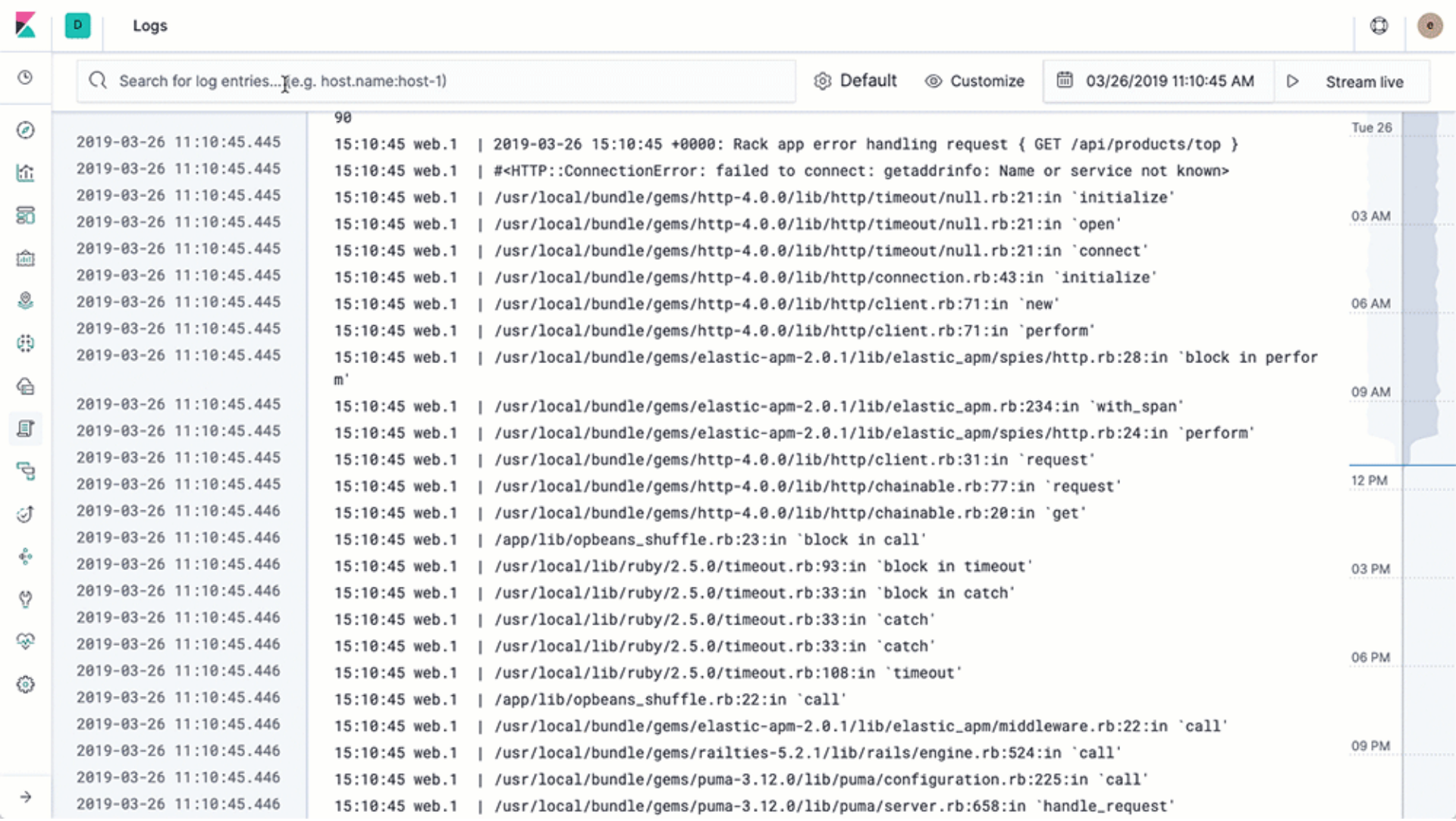
Packetbeat is a lightweight network packet analyzer that sends data from your hosts and containers to Logstash or Elasticsearch

Kibana & Metricbeat in action



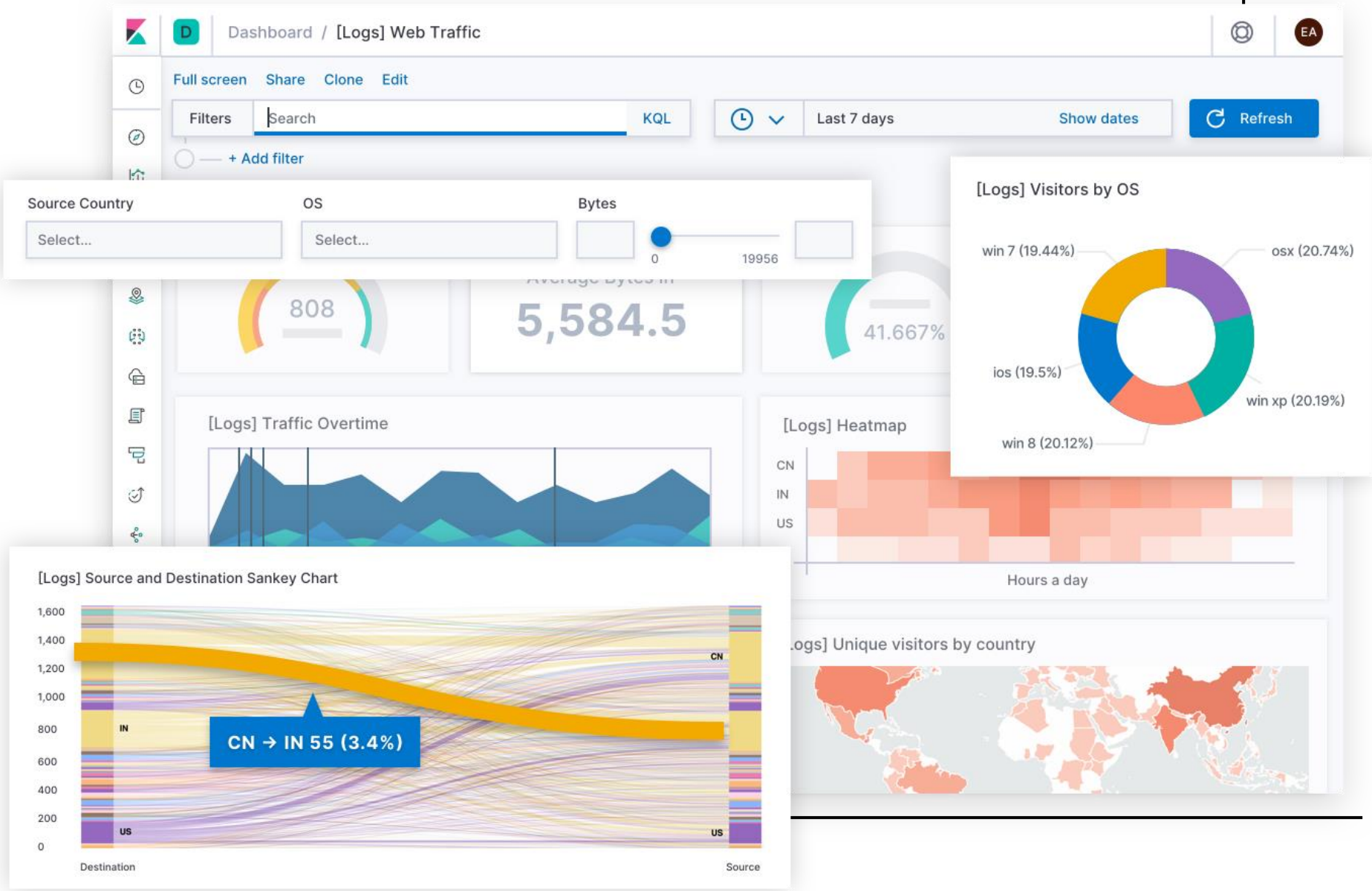
Collect metrics from your systems and services. From CPU to memory, Redis to NGINX, and much more, Metricbeat is a lightweight way to send system and service statistics.

Kibana & Filebeat in action



Filebeat offers a lightweight way to forward and centralize logs and files.

Kibana & Filebeat in action



De facto search solution

- Github uses Elasticsearch to index and query over 8 million repositories as well as indexing critical event data
- Wikipedia uses Elasticsearch to provide full-text search with highlighted search snippets, and *search-as-you-type* and *did-you-mean* suggestions
- Stack Overflow combines full-text search with geolocation queries and uses *more-like-this* to find related questions and answers
- Vimeo uses Elasticsearch to make navigation, exploration, and discovery of content as easy and awesome as possible

DB-Engines Ranking of Search Engines

☐ include secondary database models

21 systems in ranking, August 2020

Rank Aug 2020	Rank Jul 2020	Rank Aug 2019	DBMS	Database Model	Score Aug 2020	Score Jul 2020	Score Aug 2019
1.	1.	1.	Elasticsearch	Search engine, Multi-model	152.32	+0.73	+3.23
2.	2.	2.	Splunk	Search engine	89.91	+1.64	+4.03
3.	3.	3.	Solr	Search engine	51.69	+0.05	-7.43
4.	4.	4.	MarkLogic	Multi-model	12.22	+0.55	-2.25
5.	5.	↑ 6.	Microsoft Azure Search	Search engine	6.70	+0.15	-0.14
6.	6.	↓ 5.	Sphinx	Search engine	6.45	-0.09	-0.49
7.	↑ 8.	↑ 8.	Algolia	Search engine	6.14	+0.75	+1.29
8.	↓ 7.	↓ 7.	ArangoDB	Multi-model	5.73	-0.11	+0.61
9.	↑ 10.	↑ 10.	Virtuoso	Multi-model	2.65	+0.21	-0.41
10.	↓ 9.	↓ 9.	Amazon CloudSearch	Search engine	2.54	-0.13	-0.75
11.	11.	↑ 13.	Xapian	Search engine	0.82	+0.03	0.00
12.	12.	12.	CrateDB	Multi-model	0.79	+0.05	-0.12
13.	13.		Alibaba Cloud Log Service	Search engine	0.33	+0.03	
14.	14.	14.	SearchBlox	Search engine	0.32	+0.02	+0.05
15.	↑ 16.		Weaviate	Search engine, Multi-model	0.07	+0.01	
16.	↓ 15.	↓ 15.	Manticore Search	Search engine	0.06	-0.01	-0.01
17.	17.	↑ 18.	Exorbyte	Search engine	0.03	-0.02	-0.01
18.	18.	↓ 17.	searchxml	Multi-model	0.03	+0.00	-0.01
19.	19.	19.	FinchDB	Multi-model	0.02	-0.01	0.00
20.	20.	20.	Indica	Search engine	0.00	±0.00	±0.00
20.	20.		Rizhiyi	Search engine, Multi-model	0.00	±0.00	

Elasticsearch Characteristics

- Data sources
  - Elasticsearch accepts data from many different sources such as ActiveMQ, AWS SQS, DynamoDB (Amazon NoSQL), FileSystem, Git, JDBC, JMS, Kafka, LDAP, MongoDB, neo4j, RabbitMQ, Redis, Solr, Twitter, etc.
- Scalable and Distributed Operation
  - ElasticSearch is ***designed for the cloud***
    - simple to scale / attracts use cases where large clusters required
  - Elasticsearch has built-in cluster coordination subsystem
    - every search must be routed to all the right nodes to ensure that its results are accurate
    - every replica must be updated when you index or delete some documents.
    - every client request must be forwarded from the node that receives it to the nodes that can handle it.

Elasticsearch Characteristics

- Searching
  - Elasticsearch is often used for analytical querying, filtering, and grouping (e.g. aggregations)
    - other than indexing text very well, can also index numbers, dates, geographical coordinates, and almost any other datatype
  - Elasticsearch is always trying to make queries more efficient (through methods including the lowering of memory footprint and CPU usage) and improve performance at both the Lucene and Elasticsearch levels.
  - Elasticsearch is a better choice for applications that require **not only text search but** also **complex timeseries search** and **aggregations** (similar to GROUP BY)

Elasticsearch Characteristics

- Elasticsearch is very popular among newer developers due to its ease of use
- Elasticsearch is a very good option for cloud and distributed environments that need good scalability and performance

Drawbacks

- Fast for fetching the first 100s-1000s of docs. When querying ~ 10000 results it gets relatively slow
- Used for simple analytics using ES aggregations feature. Run queries such as: `"Filter all documents from 2016-04-23, and return the sum of the field 'pages' from all those documents"`
  Works fast because filter and aggregation only run on indexed data. Anything more complex (mass data manipulation, joins, or window functions for example) will not run so well.

Building blocks

Term	Description
Cluster	A group of nodes that holds all data
Node	A single machine (server) that holds some data and participates on the cluster's indexing and querying
Index	A collection of documents that have similar characteristics. An index can be divided into multiple pieces called shards.
Document	Basic unit of information that can be indexed. Expressed in JSON. Contains fields in key/value pair(s)
Shard	Primary shard and replica(s) may exist. Each Elasticsearch shard is a Lucene index with an upper limit of docs. Sharding is important for 2 reasons: • Horizontal splitting/scaling of content volume • Allows to distribute and parallelize operations across shards (potentially on multiple nodes) thus increasing performance/throughput

Shards

- Number of primary shards and replica shards can be defined per index at the time index is created
- After the index is created, you may change the number of replica shards dynamically anytime but you cannot change the number of primary shards without re-indexing data
- Ideal number of shards should be based on the amount of data in an index.
  - optimal shard should hold 30-50GB of data.
    - For example, if you expect to accumulate around 300GB of application logs in a day, having around 10 shards in that index would be reasonable.

Building blocks

Cluster

Node 1

Index 1: shard1_primary

Type

Document1	Field1	Field2	Field3
Document2	Field1	Field2	Field3

Node 2

Index 1: shard1_replica

Type

Document1	Field1	Field2	Field3
Document2	Field1	Field2	Field3

Mapping

- Mapping is the process of defining how a document, and the fields it contains, are stored and indexed
- For instance, use mappings to define:
  - which string fields should be treated as full text fields.
  - which fields contain numbers, dates, or geolocations.
  - the format of date values.
  - custom rules to control the mapping for dynamically added fields.

Mapping

- Mapping definition has:
  - Metadata fields
    - customize how a document's associated metadata is treated
    - examples: document's _index, _id, and _source fields.
  - Fields
    - A mapping contains a list of fields or properties pertinent to the document. Each field has its own data type.

See more at https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping.html

Mapping

- Index mapping

Example: A mapping for newsgroups index, when the type of the inserted document is _doc

```
{
  "newsgroups": {
      "mappings": {
          "properties": {
            "full_name": {
              "type":  "text"
            },
            "department": {
              "type": "text"
            }
          }
      }
  }
}
```

A mapping type contains a list of fields or properties pertinent to the document. Each field has a data type such as text, keyword, date, long, double, boolean or ip.

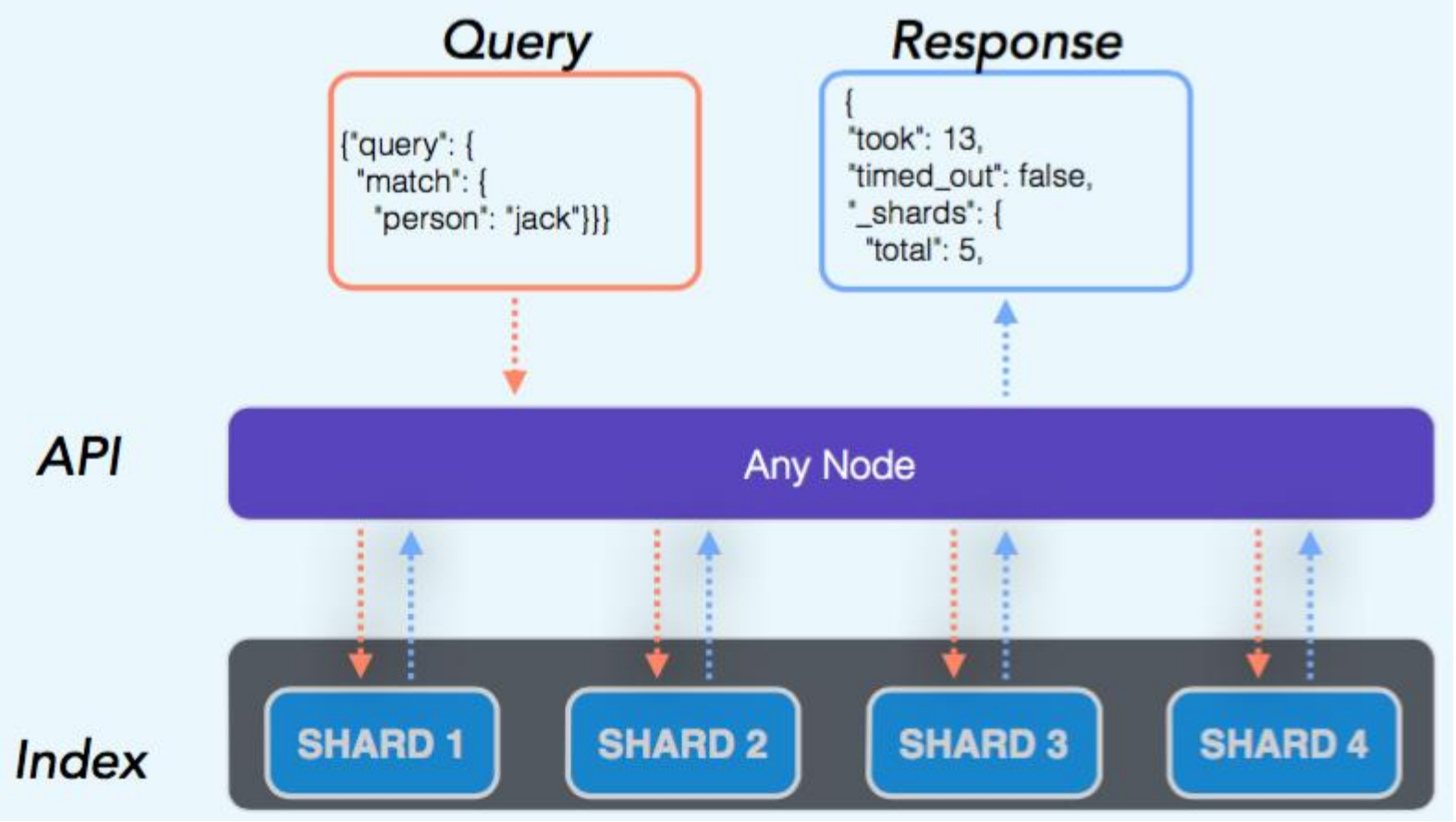
Document to be indexed

```
{ "_index": "newsgroups", "full_name": "john smith", "department":
"computer science"}
```

Dynamic mappings

- Mappings can be explicitly created when creating an index (e.g. via REST API call)
- To index a doc, you don't have to first create an index, define a mapping type, and define your fields
- Every time a document contains new mapping type and new fields, those end up in index's mappings
  - Lucene StandardAnalyzer for automatic type guessing
- However, defining too many fields in an index is a condition that can lead to a mapping explosion, which can cause out of memory errors and difficult situations to recover from
  - Use settings to limit the number of field mappings

Query and response

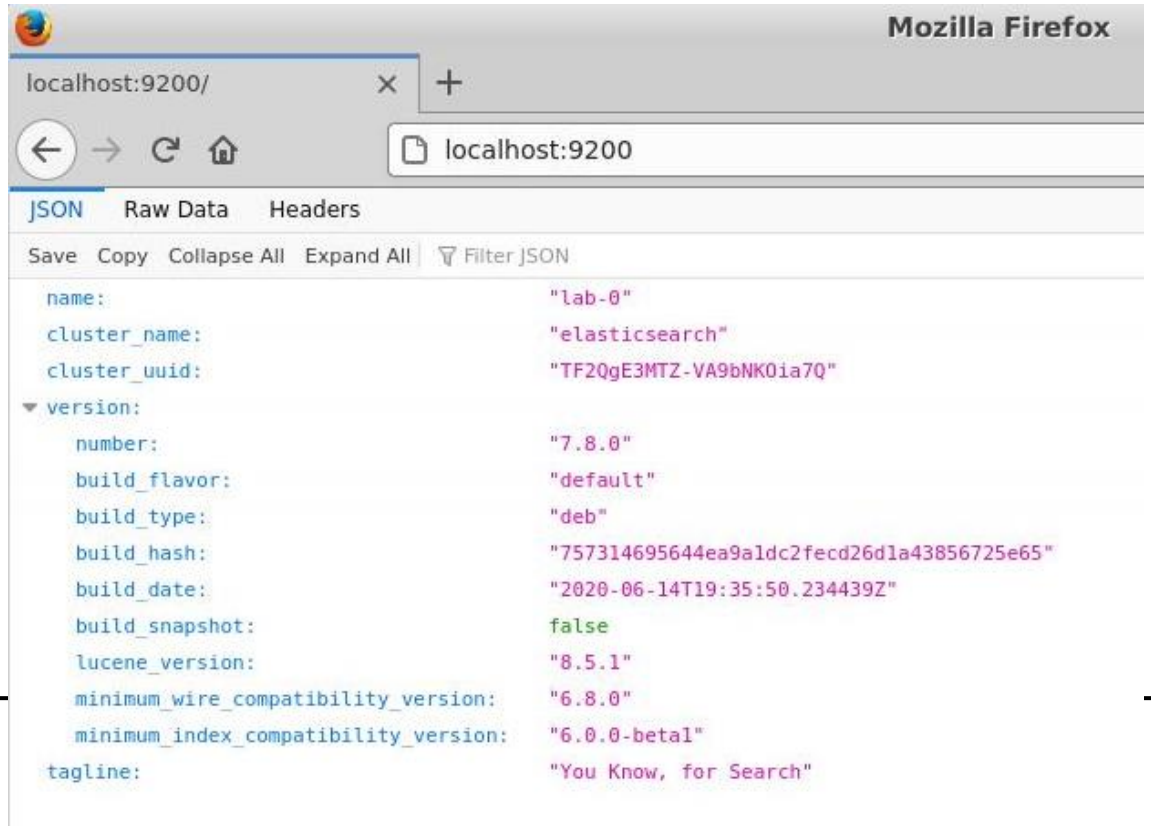


Hands on: ElasticSearch

- Elasticsearch v7.8.0 installed on VM
- Kibana installed on VM
- Logstash installed on VM

Hands on: ElasticSearch

- Start Elasticsearch as a service
  - `sudo service elasticsearch start`
- Check if Elasticsearch is working:
  - http://localhost:9200



Hands on: MetricBeat & Kibana

- Metricbeat helps you monitor your servers and the services they host by collecting metrics from the operating system and services.

- Steps to be followed to use MetricBeat:
  - install Metricbeat on each system you want to monitor
  - specify the metrics you want to collect
  - send the metrics to Elasticsearch
  - visualize the metrics data in Kibana

Hands on: MetricBeat & Kibana

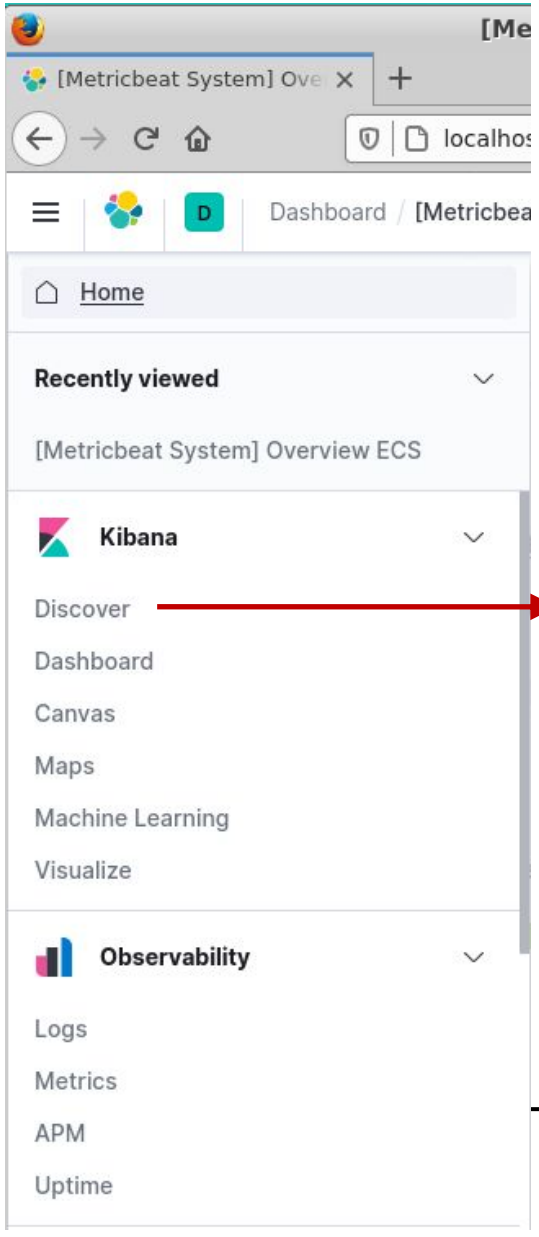
- Install Metricbeat on VM
  - Follow steps 1-4 to install Kibana using APT repositories https://www.elastic.co/guide/en/beats/metricbeat/current/setup-repositories.html
- Start Kibana as a service
  - `sudo service kibana start`
- Set up the Kibana dashboards for Metricbeat
  - `sudo metricbeat setup --dashboards`
    - Kibana must be running and reachable
- See list of available modules that collect metrics
  - `sudo metricbeat modules list`

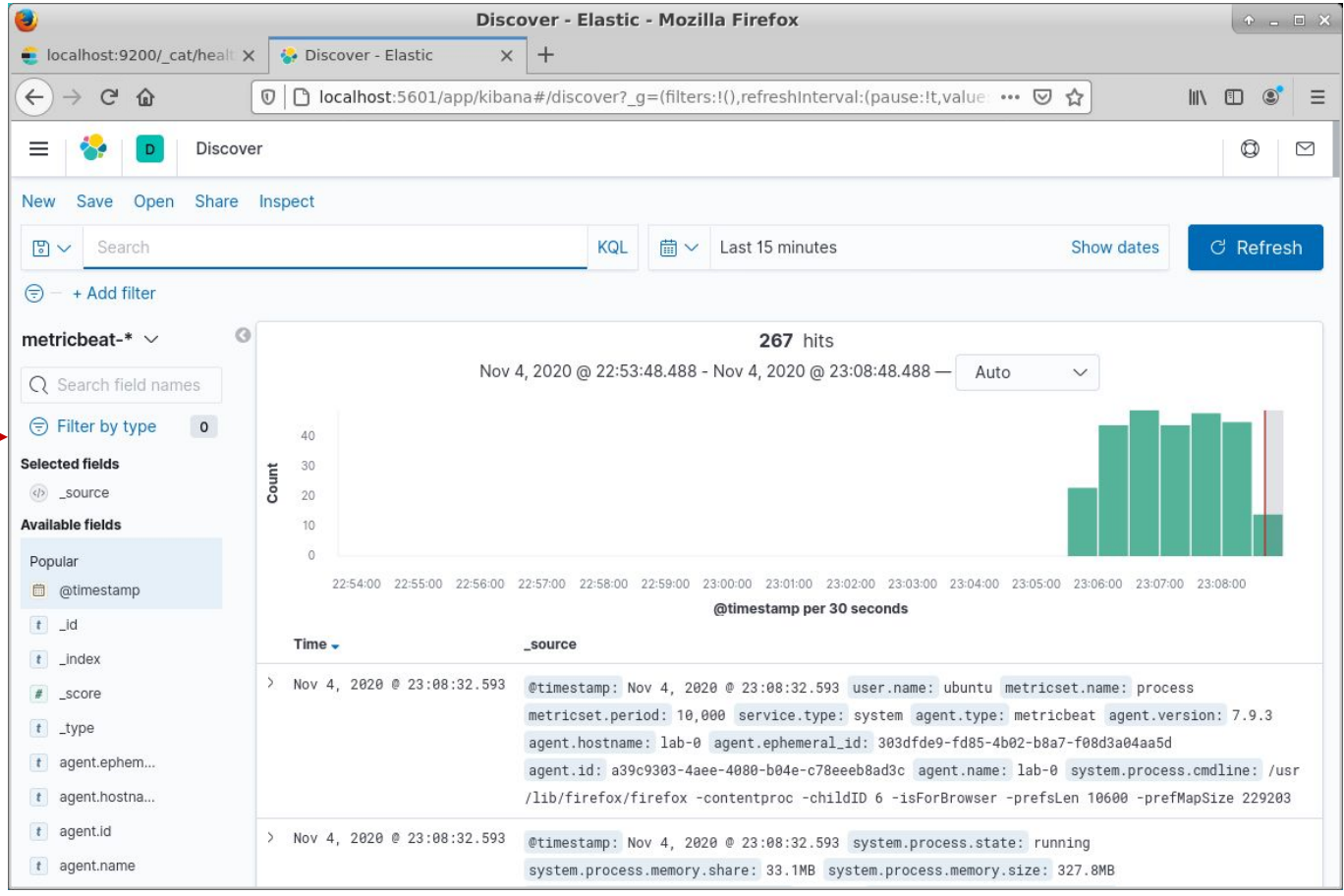
Hands on: MetricBeat & Kibana

- You can enable one or more modules:
  - `sudo metricbeat modules enable apache mysql`
  - If you accept the default configuration without enabling additional modules, Metricbeat collects system metrics only.

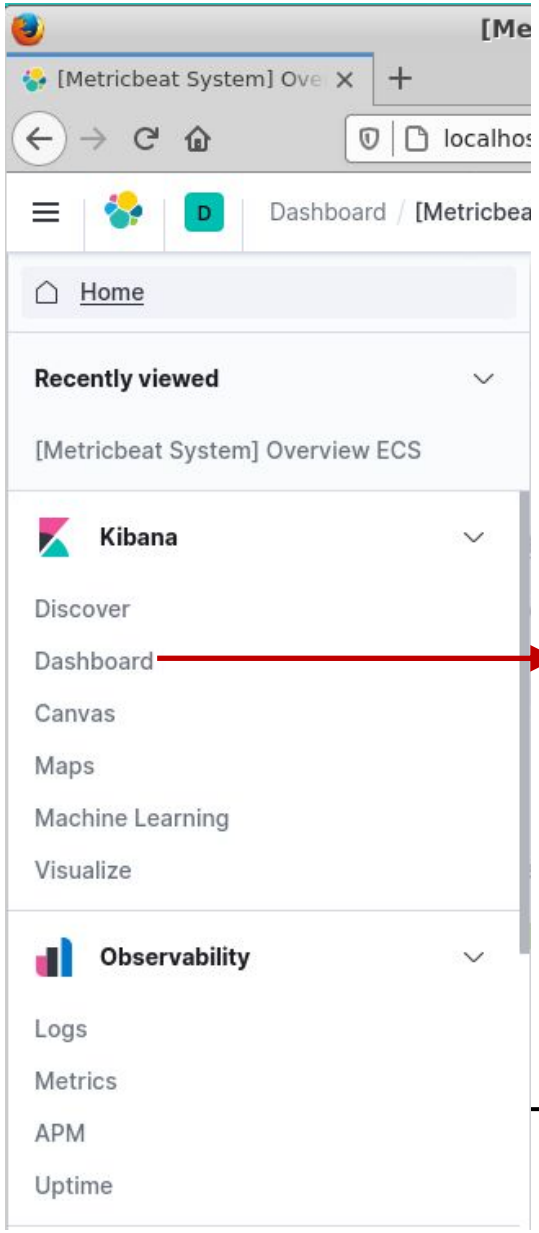
- Start Metricbeat as a service
  - `sudo service metricbeat start`

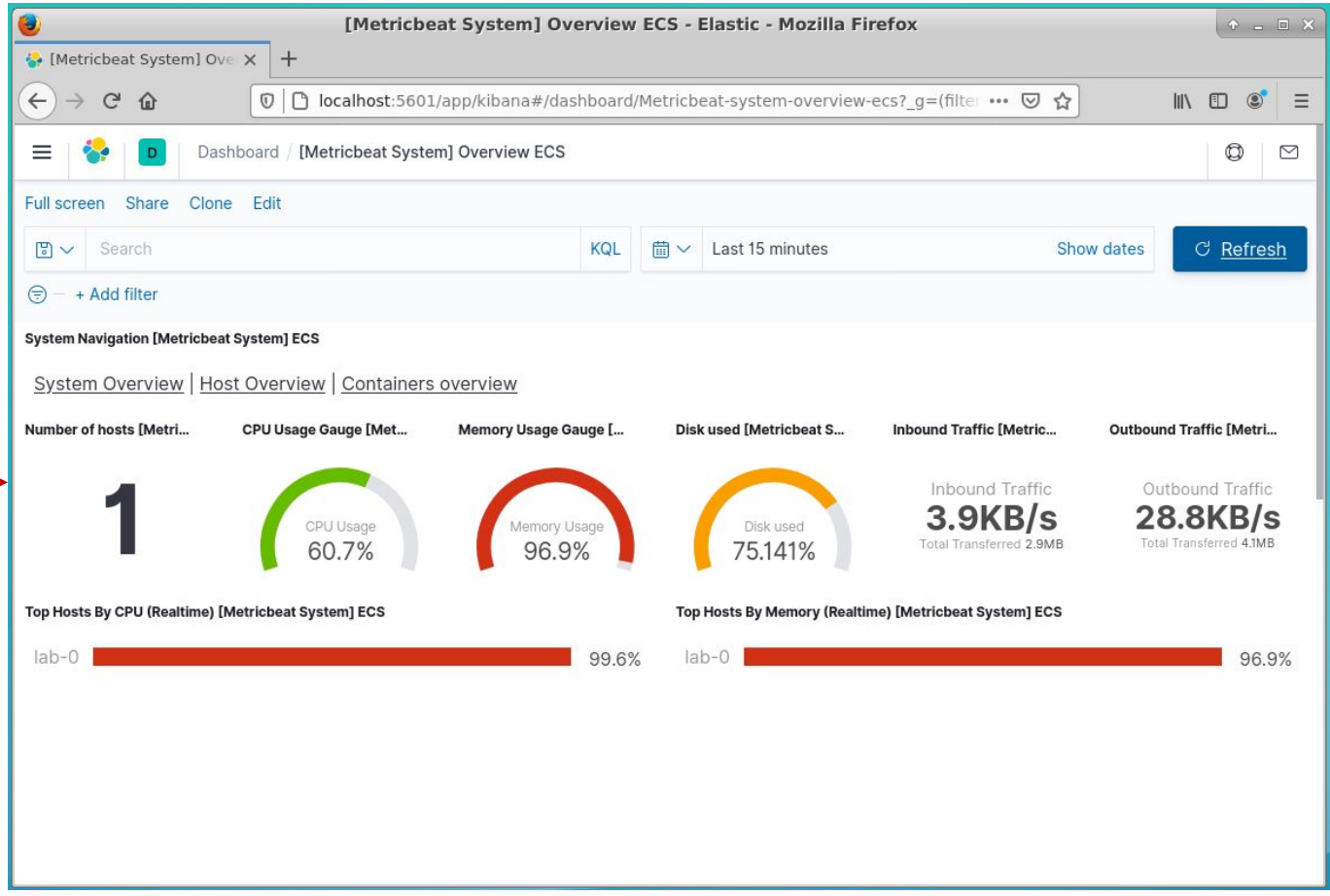
Kibana – Discover page





Kibana – Dashboard page





Experimenting with Logstash

- Logstash activation as a service
  - sudo service logstash start
- Alternative way to start logstash: using binaries
  - cd /usr/share/logstash/bin
  - sudo ./logstash -e 'input { stdin {} } output { stdout {} }'
    - -e flag enables specifying configuration directly from command line
    - Pipeline in the example takes input from the standard input, stdin, and moves that input to the standard output, stdout, in a structured format.

After starting Logstash, wait until you see "Successfully started Logstash API endpoint" and enter messages at the command prompt