Kibana:-

field: value → Starts: 200

field: " ____ " → exact phrase

field: *

NOT field: " ____ " → exclude this field from o/p.

AND, OR, >, <, >=, <=

? ⇒ OS: Windows
↳ Single letter

fields: (Value1 or Value2)

@timestamp > now - 1h → last 1hour.

@timestamp: now → current time.

@timestamp: [now - 7d TO Now]
↳ last 7day.

ES|QL → Elastic Search Query language:-

Limit = 5 → you can only 5 output.

where → filter out the data.

count() → overall count BY → splitting out.

|Stats count() BY Play_name.

|Stats total_count = count() BY Play_name.
↳ NewName ↳ func ↳ clause ↳ fieldname

|KEEP f1, f2, f3 - - - - → Keep field in the o/p.

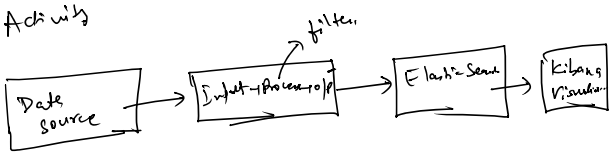
|sort f1 desc → Descending.
asc → Ascending

|Rename new_name = old_name.

Logstash:-

Ship → Transforms → store.

Data Processing pipeline that will do this Activity



Input stage:- Collect the data.

Input stage:- Collect the data.

Plugin:- beat, Kafka, tcp, udp, file.

Ex:-
```
input {
    beat {
        port = 5044
    }
}
```

Filter stage:-

Processing, parsing & enrichment of data happens.

grok – Parse the unstructured data using regex
drop – remove certain data.
mutate – Rename
date – Parse & Normalize timestamp
json – Parse json data.
translate – lookup
geoip – Add geographic info from ip address.

09/09/09

```
filter {
    grok {
        match = "{message = }"
    }
    date {
        match = [timestamp, "mmddyy"]
    }
}
```

Output stage:- Where to send the processed data.

O/P → kafka, elasticSearch, stdout etc.

ex:-
```
Output
{
    elasticSearch {
        host = 127.0.0.1:9200
        index = apache-log
    }
}
```

Pipeline:-

Input → Queue → Filter → output.

Parse | clean | enrich your data.

Input → collect the event

Queue → Temporary store data to memory queue / Persistent Disk queue.

Output → Destination like elasticSearch.

Logstash Internal memory

↳ memory Queue (Default)
↳ fast | store

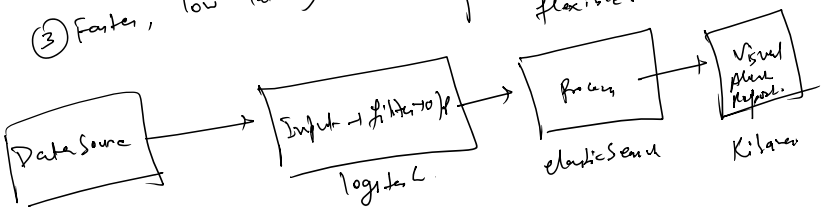
Logstash Internal memory

- memory Queue (Default)
  - fast | store
- Persistent Disk Queue → large memory, to avoid data loss.

ElasticSearch Pipeline vs logstash pipeline :-

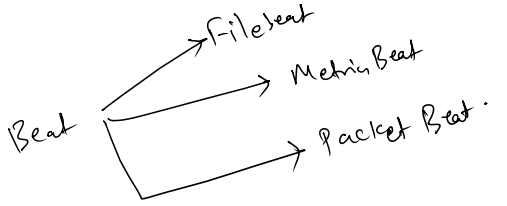
1. Simple pre-index transformation
2. simple | lightweight transformation steps
3. Faster, low latency

1. Complex data.
2. Heavier & complex data.
3. Heavier but more flexible.



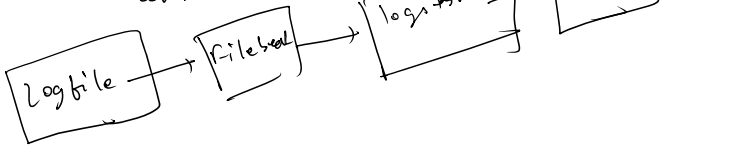
* Filebeat, Metricbeat & Packetbeat :-

Beat is a lightweight agent that will forward the data from source to logstash (heavy processing) or elastic search (Direct Indexing)



① Filebeat :-

Collect log files from the system containers or devices.




```
filebeat.inputs:
- type: log
  paths: /var/log/...
output.elasticsearch:
  hosts: [
```

② MetricBeat :-

Metric data like CPU, Disk, memory, ...

```
metricbeat.modules:
  - system
```

(2) Metricbeat:-

```
metricbeat.modules:
- module: system
  metricsets:
    - CPU
    - Disk
    - memory
  period: 10s
  hosts: ["localhost"]

output.elasticsearch:
  hosts: [
```

(3) Packetbeat:- Network Analysis.

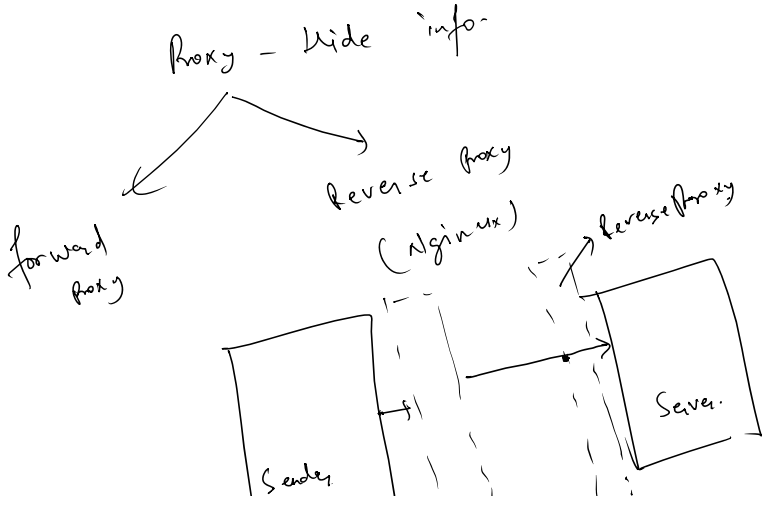
Use case: N/w latency, App. latency.

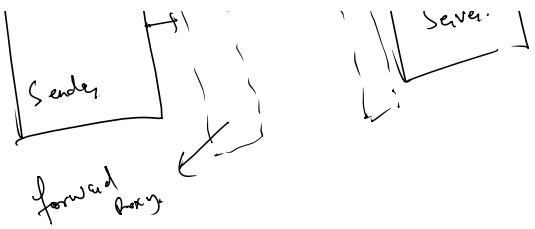
```
packet.interface.device: any
packet protocol
  - type: http
    ports: [80, 8080, 8000]
  - type: dns
    port: [32]
output.elasticsearch:
  hosts: [
```

Filebeat:-

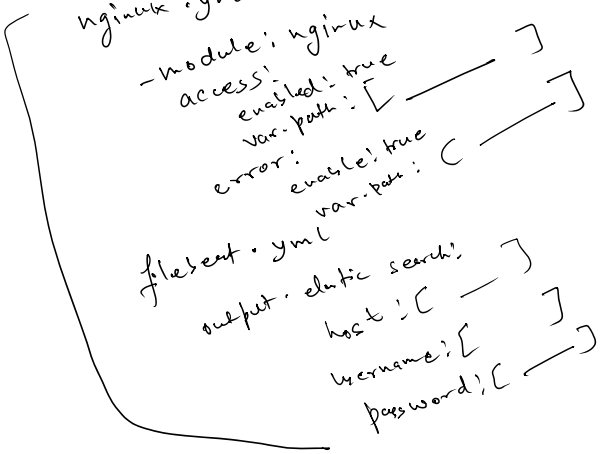
nginx
apache
System logs

Proxy - Hide info.





① access.log.
② error.log.



① Nginux
② filebeat.
③ Enable Nginux module in filebeat
④ nginux.yml
⑤ filebeat.yml → elastic Search

* Centralized management of Beats:-

Fleet → 8.x
6.x & 7.x → Centrd Mangmnt of Beats.

* Scaling & Performance:-

① cluster Architecture (Node, Master & Data Ingest)
① High Availability
② Backup.
③ load Management.

Index - logical namespace for document distribution.
→ Primary shards - Original data is stored.

Shards → Replica Shards – copy of the original shards is stored

1 index, 5 Primary shards, replica shards 1

total shards – 10

Node:- Each Node can perform different Activity

1. Master Node.
2. Data Node.
3. Ingest Node
4. Coordinating Node.

(1) Master Node – Coordination & Managment of Nodes.

ex:-
```
node.role : ["master"]
cluster-name : cluster-master1
node-name : master-node-1.
```

(2) Data Node:- Store & serve data in shards.

Handle indexing
Help in searching Process.

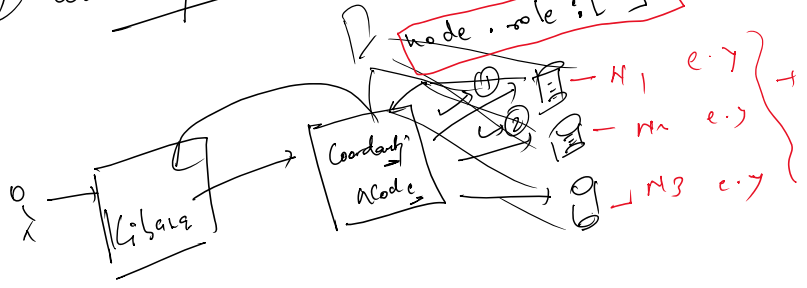
```
node.role : ["data"]
path.data :
path.log :
```

(3) Ingest Node:- Preprocess the index data.

```
node.role :["ingest"]
```

(4) Coordinating Node:- Act as Query / Load Balancing.

```
node.role :[ ]
```



→ Snapshot & Backup for recovery:-

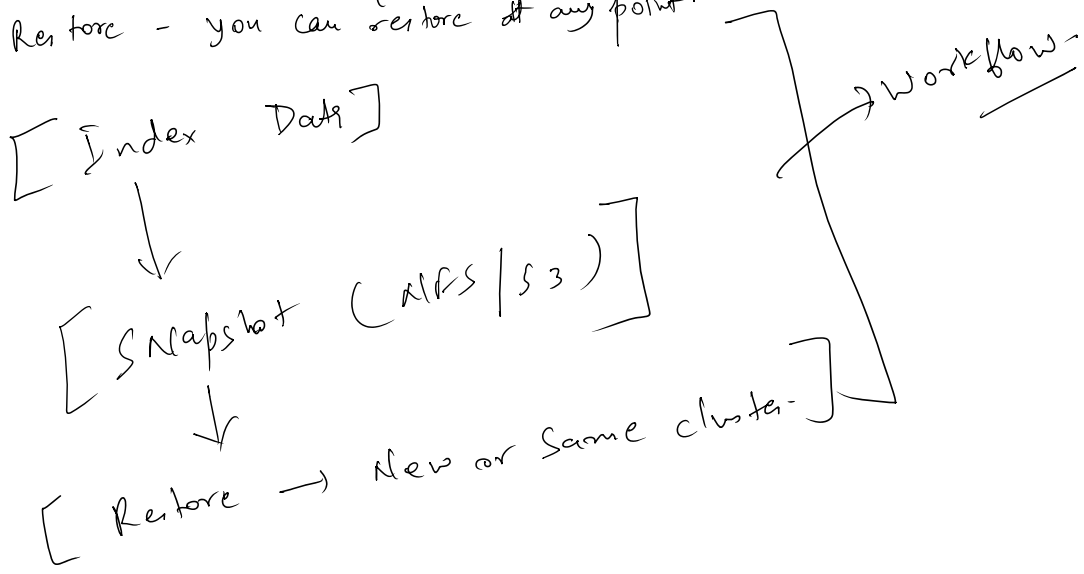
* Snapshot & Backup for recovery :-

Create the copy of the cluster.

(1) Index Data (Primary & replica shards)
(2) Index Setting, setting.
(3) Restore at any point.

Snapshots are:-

(1) Incremental.
(2) Non-blocking - Not effect the existing running cluster.
(3) Restore - you can restore at any point.



(1) elasticsearch.yml
path.repo = "[ ]"

(2) Register.

* Monitoring using X-Pack or metricBeats

Elastic search - latency, Schema, Node, ---

X-Pack:-

... feature to monitor the elas

h'cSeard -

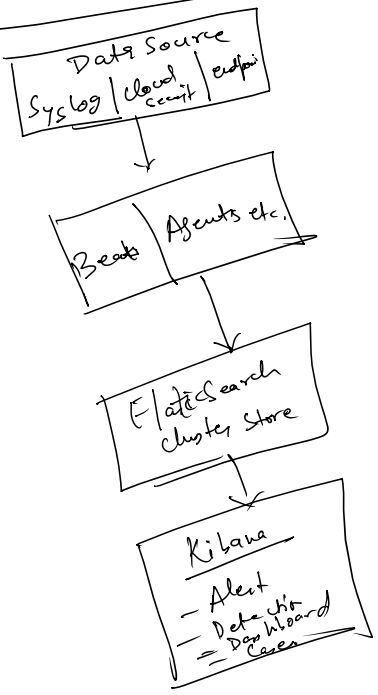
X-Pack:-

(1) Inbuilt feature to monitor the elas
(2) collect the internal metrics of logstash, elastic search, Kibana & beat

elasticSearch.yml

↳ xpack.monitoring.collection.enabled : true.

SIEM & Security use case (Elastic Search) :—



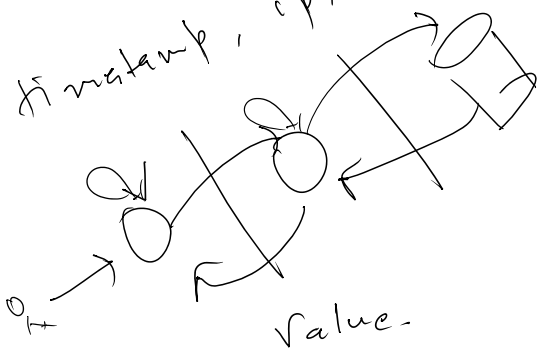
(1) Windows eventlogs. — Winlogbeat
(2) linux Syslog. — filebeat
(3) Network Devices — Filebeat
(4) Cloud Platform — Filebeat cloud module.
(5) Endpoint — Elastic Agent
(6) Application. — Filebeat json module.

(6) Application.

Metric – CPU = 45%.

log – timestamp, ip,

trace –



Aggregate value.

SLI –

SLO –

SLA – Agreement b/w two party.

30min – P1 → SLA

25min → P1 → Team

SLO:

Error Budget:

99.5% → SLO.

0.5% → EB

ELK vs Splunk vs Datadog.