Elastic Agent: Ingesting log files, how hard can it be?

Jan 20, 2026

Denis Rechkunov

elastic | The Search AI Company

Denis Rechkunov

Principal Software Engineer

Elastic Observability Beats

https://rdner.de

Logs are a foundation of Observability

Let’s talk about ingestion of log files

What’s the general idea?

Read a log line

We need a component that reads files line by line.

Map it to JSON

We need to somehow convert it to JSON.

Send it to Elasticsearch

We need to use an Elasticsearch client to send this JSON to an index.

Raise your hand if you think this is easy 

```
while read l
do curl -H"Authorization: ApiKey BASE64_API_KEY" \
  -XPOST localhost:9200/my-index/_doc \
  -HContent-Type:application/json \
  -d"$l"
done < data.ndjson
```

Nope, it’s hard.

Wouldn't it be fun to design it together really quick?

Let’s split the problem into these 3 parts:

Read a log line

We need a component that reads files line by line.

Map it to JSON

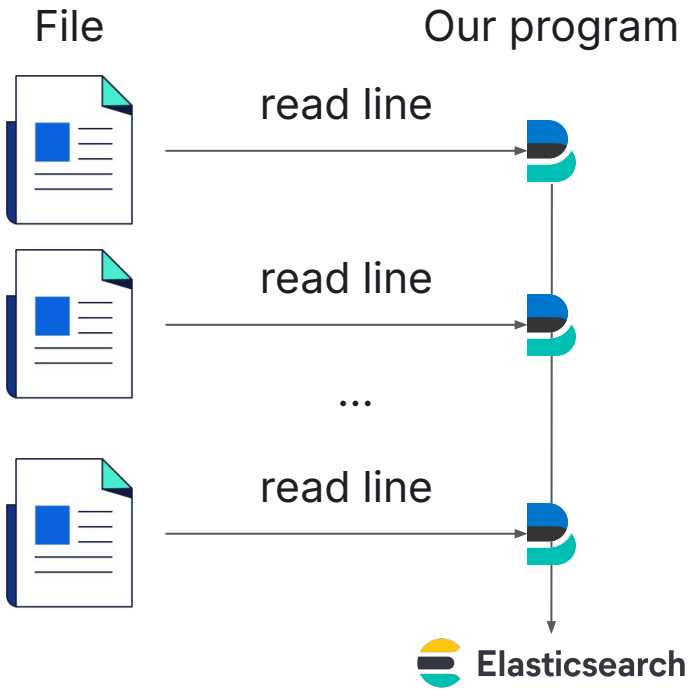
We need to somehow convert it to JSON.

Send it to Elasticsearch

We need to use an Elasticsearch client to send this JSON to an index.

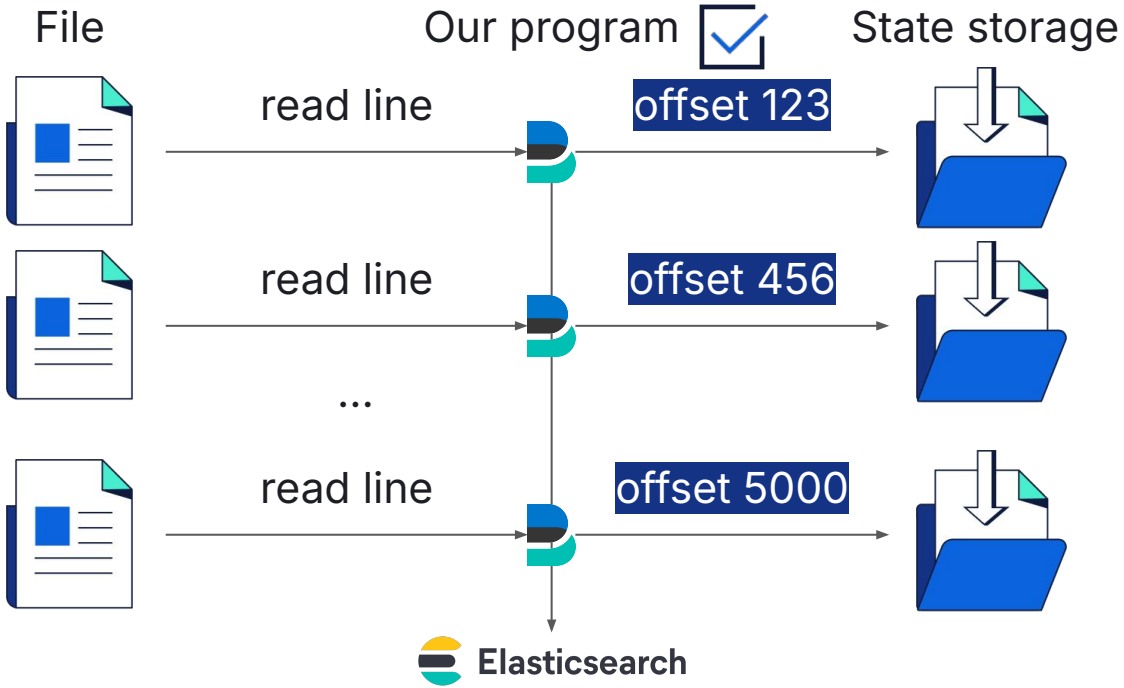
Part 1: Read a log line

Our First Approach



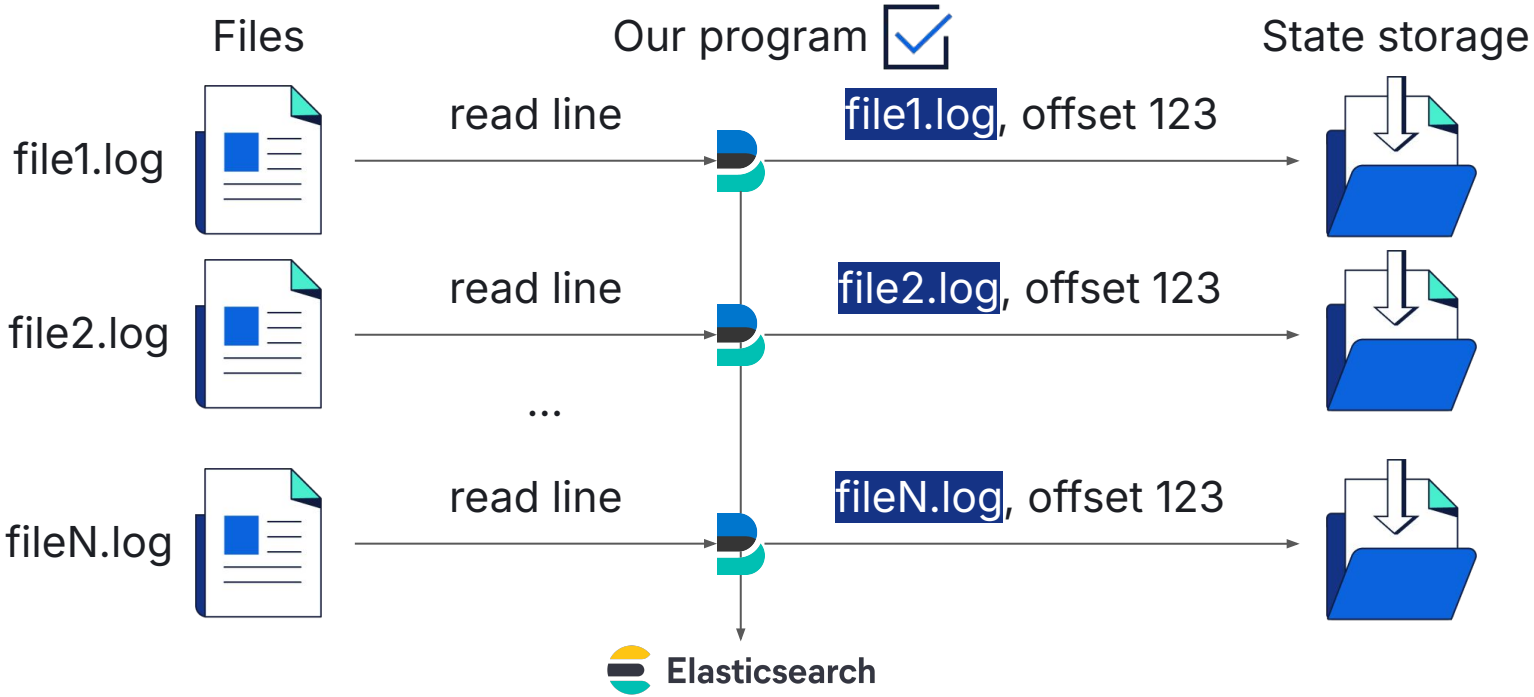

Uh-oh…
Someone restarted our program 🙄
We sent duplicates after the restart.
Credit: Heroes of Might and Magic III (videogame)

Offset Tracking




Uh-oh…
Someone added thousands of files to ingestion 🙄
We need to track multiple files now.
Credit: Heroes of Might and Magic III (videogame)

Per File Offset Tracking



Scaling Considerations

How to optimize for **thousands of files**

- State storage:
  - Key-Value disk storage.
  - Efficient writes, e.g. Write-ahead logging (WAL) with checkpointing.
  - In-memory state view or direct access to the on-disk storage if it's fast enough.
- Reading files:
  - OS has a file handle limit per process, sometimes as low as 256 (Mac).
  - Need to constantly close files on idle and re-open files on change.


Uh-oh...
The log rotation moved
“file1.log” to “old-file1.log”
🙄
We sent duplicates again.
Credit: Heroes of Might and Magic III (videogame)

Log Rotation

Challenges to pick a **stable file identifier**

- Throughout the lifetime of the file we need to have a stable identifier.
- Files can be:
  - renamed/moved
  - truncated
  - compressed to Gzip
  - removed
- We cannot rely on its filename as a stable identifier.

File Identification

Based on file system metadata

- It’s kind of a **coordinate** where the file is, **not** its **name.**
- For **Unix**-like systems, it’s the stat system call and a combination of:
  - device number (**st_dev**)
  - inode (**ino_t**)
- For **Windows**, it’s the file handler and a combination of:
  - **dwVolumeSerialNumber**
  - **nFileIndexLow**
  - **nFileIndexHigh**

More details with examples at https://www.elastic.co/docs/reference/beats/filebeat/file-identity

File Identification

Based on the content

- The content is what makes this file unique, nothing else affects the identity.
- Two main approaches:
  - **Hash** from the first **N bytes** – ingestion delay until the size is at least **N** bytes.
  - **Variable** length **match** until different – ingest right away, delay a file if collisions occur until it grows different.

More details with examples at https://www.elastic.co/docs/reference/beats/filebeat/file-identity

File Identification

Why is file system metadata unreliable (Unix)?

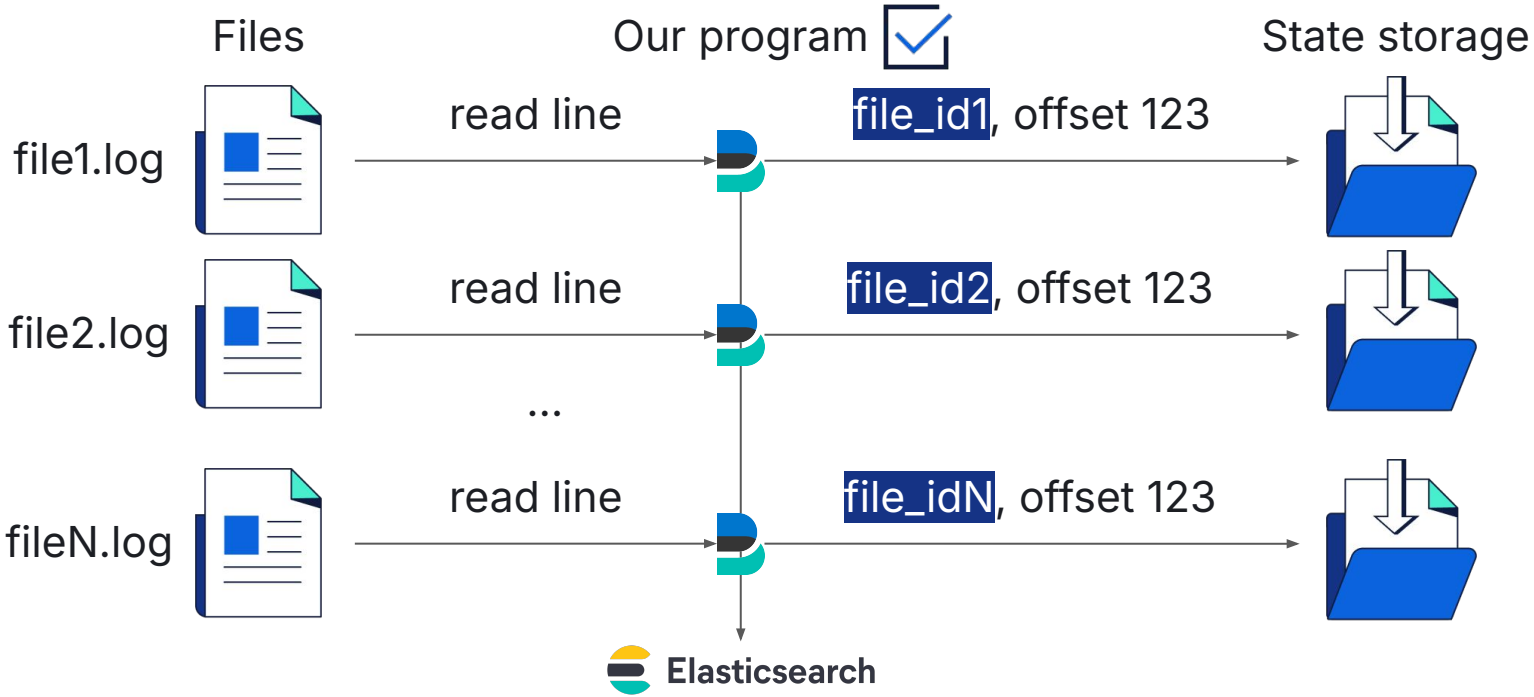
- Not every file system implements **inode** very well.
  Examples: NFS, FUSE filesystems.
- **inode** number gets reused for deleted files.
  Examples: containerized and virtualized environments.
- The **device ID/number** is unstable.
  Examples: when using the Linux LVM (Logical Volume Manager), device numbers are allocated dynamically at module load.
  (*refer to Persistent Device Numbers in the Red Hat Enterprise Linux documentation*)

File Identification

Better use the content-based identifiers

- Files need to be different in content (most of log lines have timestamps, and it’s enough).
- If the beginning of the file changed, it’s fair to assume it’s a new file.
- However, we have to delay ingestion of some files until we can create a **unique stable identifier** for them.

Per File Offset Tracking with ID



Part 2: Map it to JSON

A log line to a JSON document

Straight-forward

- Some logs are already in JSON, nothing to do.
- Some logs are not JSON and they need to be converted
  e.g. with Grok patterns (not to be confused with Grok AI🤦).
- 🎉

Credit: Heroes of Might and Magic III (videogame)

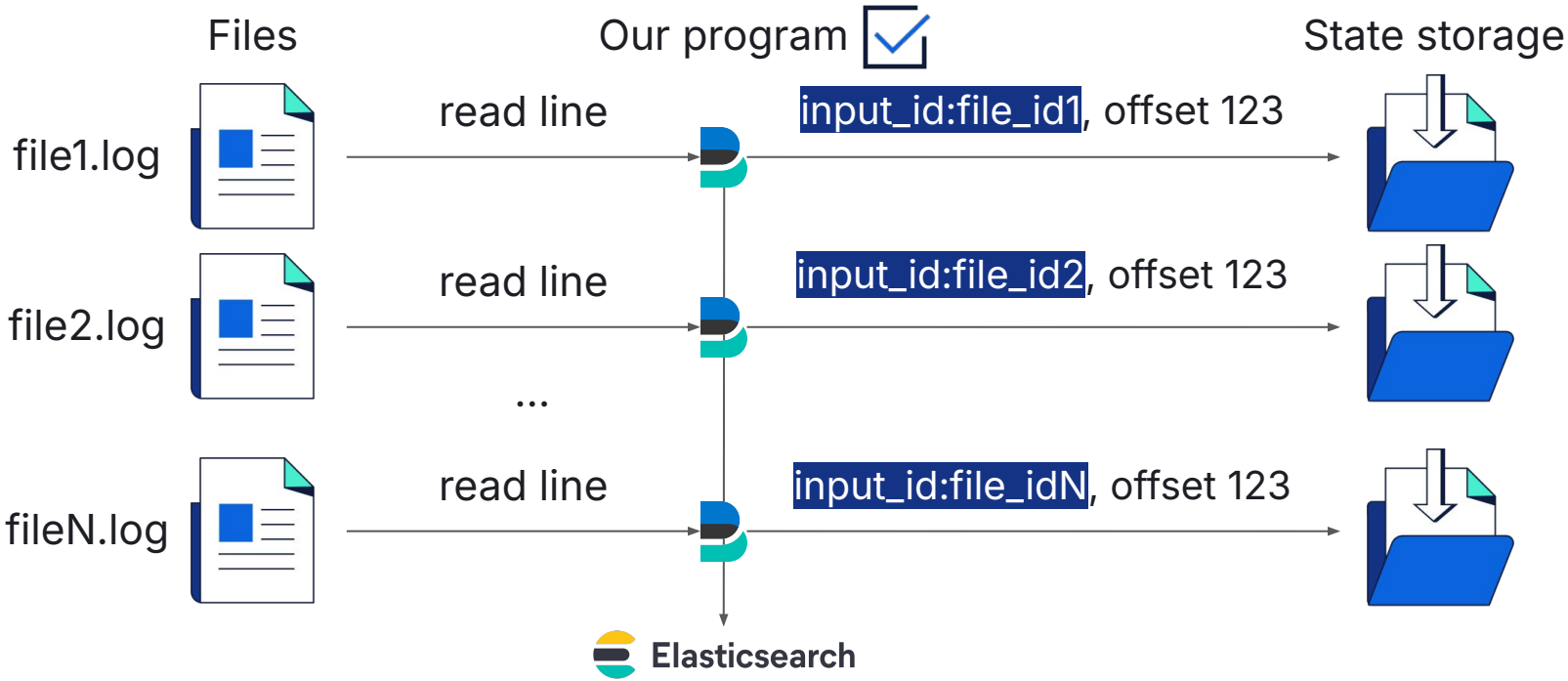
Processors

Requirements

- We introduce a processing pipeline (aka **input**) with dozens of processors.
  - add_field
  - drop_field
  - rename
  - decode
  - ... etc.
- Every processor must be:
  - lightweight – it's a **hot code path**
  - atomic – make backups before making changes
  - conditional – e.g. when/if/then/else


Uh-oh...
Users wish to process the same file differently by 2 different inputs.
Credit: Heroes of Might and Magic III (videogame)
Gem
Month: 1, Week: 4, pikabu.ru

Offset Tracking with File ID and Input ID

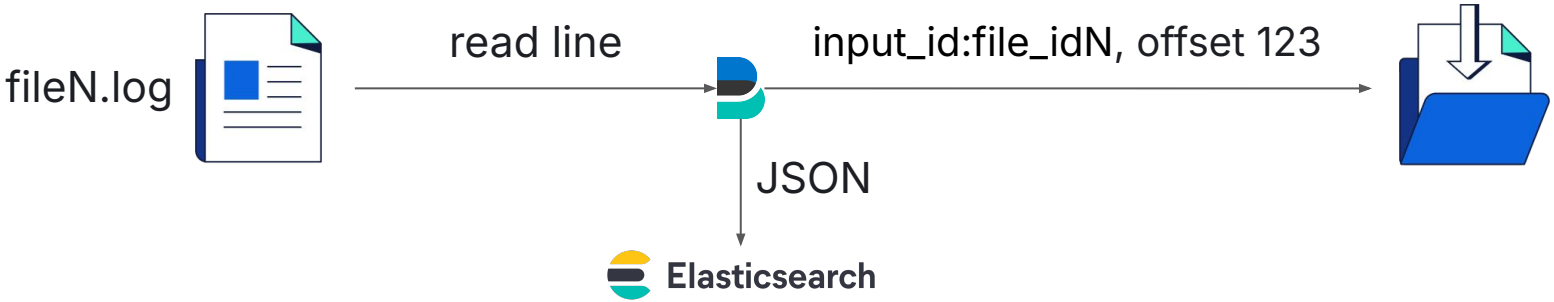


Part 3: Send it to Elasticsearch

Sending data to Elasticsearch

Straight-forward

- Use the official client, e.g. for Go.
- Send each JSON document to Elasticsearch.
- 🎉




Uh-oh…
Load spikes caused data loss.
The back pressure didn't let
us send data quick enough.
We need a queue.
Credit: Heroes of Might and Magic III (videogame)

Sending to Elasticsearch 

How to gain resilience

- Implement a queue: memory or disk.
- Combine JSON documents in batches and use the bulk API.
- Track acknowledgements for documents.
- Store offset **after** a document is **acknowledged**.
- Do exponential back-offs and retries.

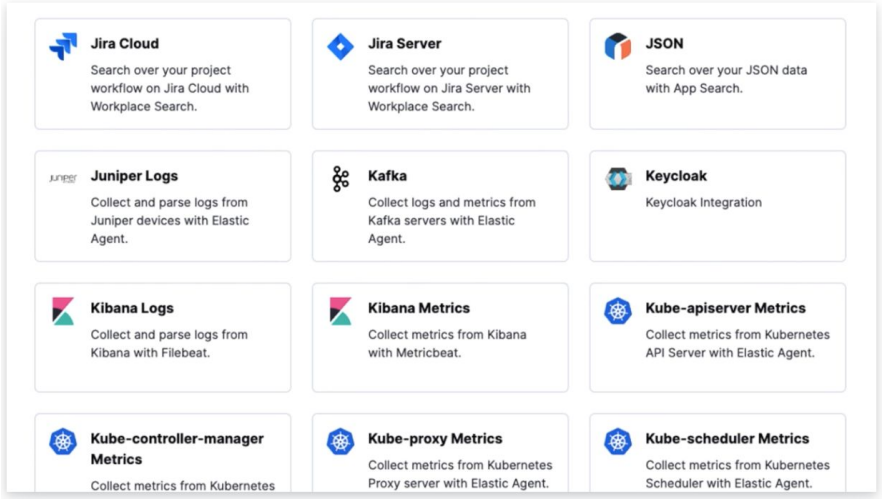

Uh-oh...
Users noticed that logs have established formats.
They want us to have presets.
Credit: Heroes of Might and Magic III (videogame)
Gem
Month: 1, Week: 4
pikabu.ru
kletka.net

Bonus Part: Integrations

Integrations

Logs have established formats, we need to:

- Manage the catalog of integration packages
- Version them
- Configure them



More at https://www.elastic.co/integrations


Uh-oh...
Users want centralized management for all of this across hundreds of machines.
Credit: Heroes of Might and Magic III (videogame)

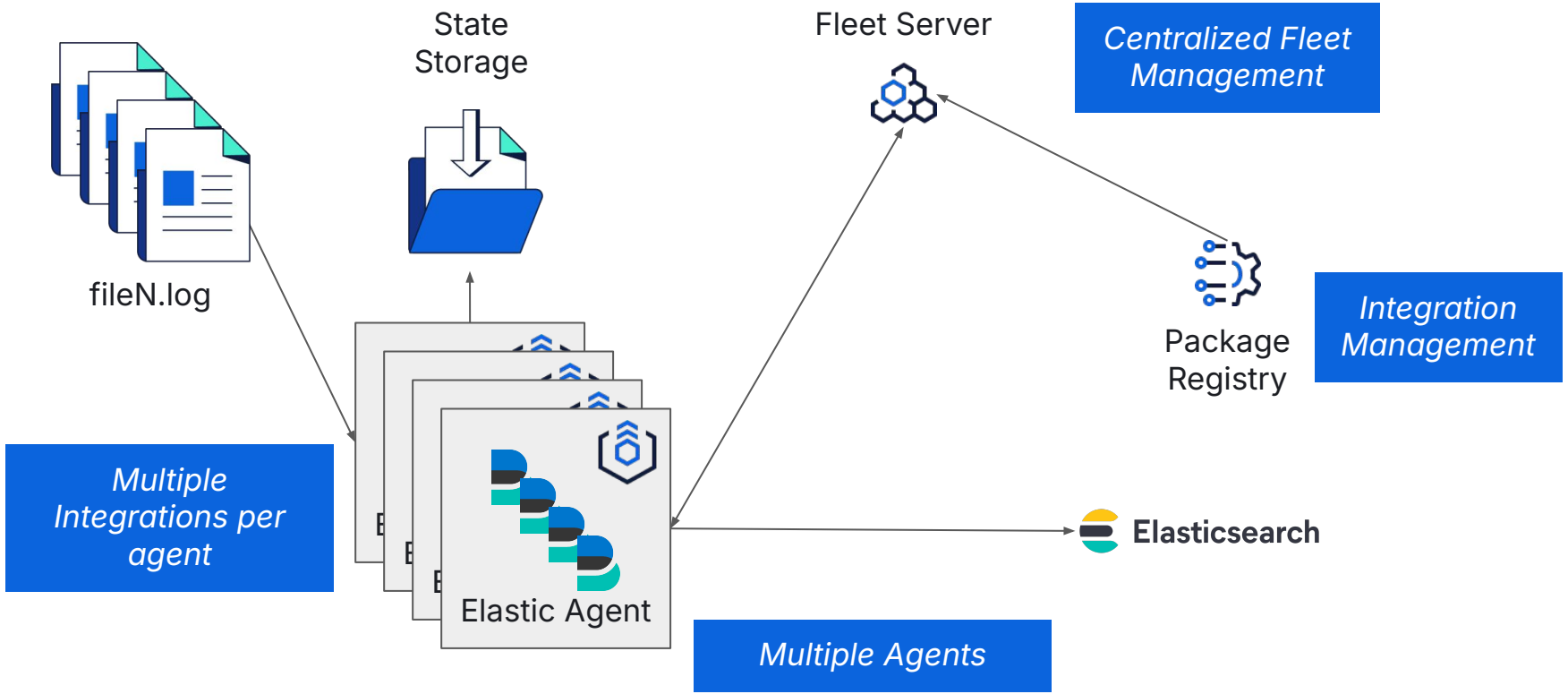
Fleet Management

Centralized management across hundreds of machines

- Every ingestion program is configured and supervised by a locally running service – agent.
- A centralized server manages these agents.
- They can be re-configured with integration packages in runtime.
- Agents report their metrics and state back to the Fleet server.

More at https://www.elastic.co/docs/reference/fleet

How do we solve it at Elastic?




Uh-oh…
Users want open standards
and no vendor-locking.
Credit: Heroes of Might and Magic III (videogame)

Elastic Agent as OpenTelemetry Collector

Elastic **D**istribution of **O**pen**T**elemetry (EDOT)

- Elastic Agent is also an OTel Collector.
- Beats (ingesting software running under the agent) are also OTel receivers.
- OTel components can be running side by side with Elastic-specific components.
- Elastic is actively contributing to OpenTelemetry.
- Using open standards makes it future-proof and removes vendor-locking.

More at https://www.elastic.co/docs/reference/edot-collector

Elastic Contributions to OpenTelemetry

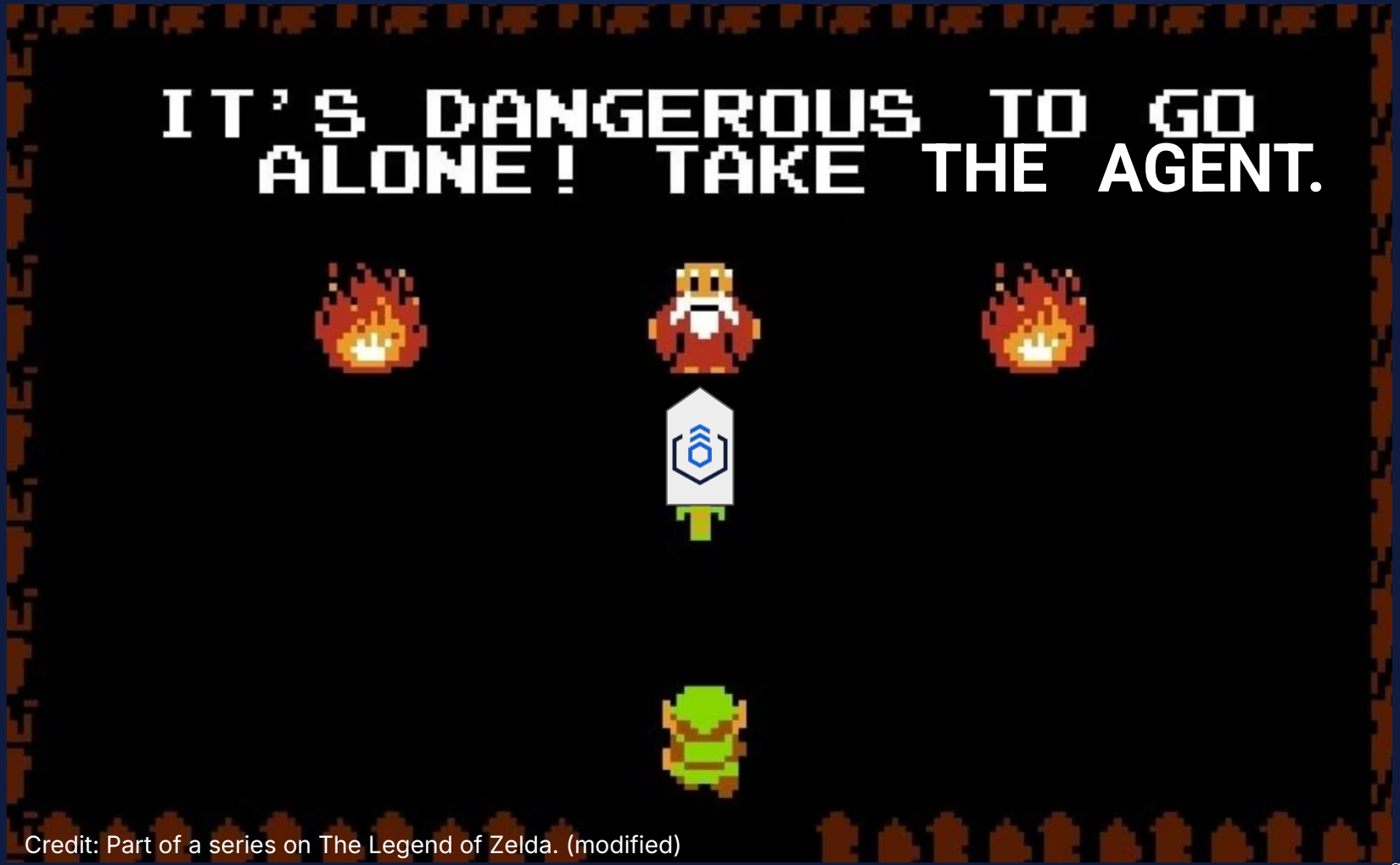
As of January 2026:

OpenTelemetry Companies statistics (Contributions, Range: Last decade), bot...

Rank	Company	Number
	All	1301377
1	Splunk Inc.	290032
2	Microsoft Corporation	192429
3	LightStep Inc.	68899
4	Dynatrace LLC	63942
5	Elasticsearch Inc.	55090
6	Google LLC	46846
7	Amazon	43405
8	Datadog Inc	41711

Source https://opentelemetry.devstats.cncf.io/d/5/companies-table

I guess the point is…



Credit: Part of a series on The Legend of Zelda. (modified)

Thank you!

My contacts

Slides