FAST LOG MANAGEMENT

For your infrastructure

EXOSCALE

@nicolas_frankel

ME, MYSELF AND I

- **Developer advocate**
  - Previously developer/architect consultant
- **DevOps-minded**

EXOSCALE

@nicolas_frankel

European alternative to the “big” cloud-computing players

- **Privacy-minded**
- **Great support**

The root of all evil

```
LOGGER.debug(
    "Cart price is now {}", cart.getPrice())
```

An improvement, but for whom?

```
if (LOGGER.isDebugEnabled()) {
   LOGGER.debug(
      "Cart price is now {}", cart.getPrice())
}
```

Ooops?!

```
if (LOGGER.isDebugEnabled()) {
   LOGGER.warn(
      "Cart price is now {}", cart.getPrice())
}
```

Lazy computation for the win!

```
LOGGER.debug(formatter ->
   formatter.format(
      "Cart price is now {}", cart.getPrice())
)
```

We are bound to the physical world...

- **SSD vs. HDD vs. NFS**
- **It takes time to actually write to a file**

EXOSCALE

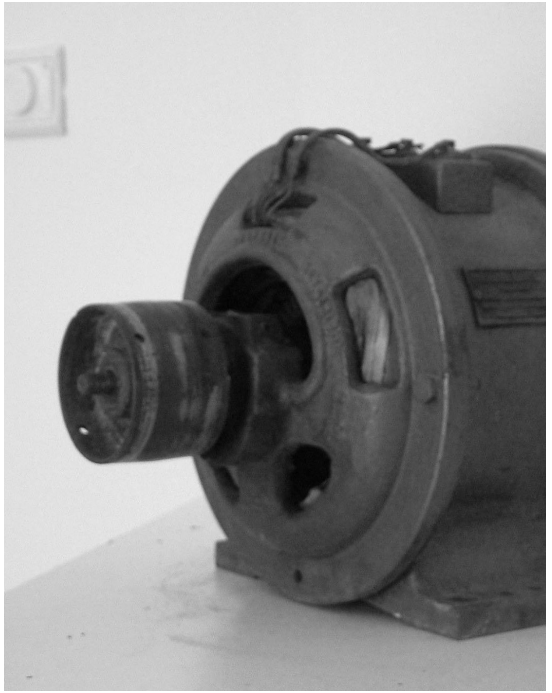
@nicolas_frankel

Writing process

1. **Open the stream**
2. **Write bytes**
3. **Close the stream**

Synchronous vs. asynchronous logging

- **By default, logging is blocking**
- **Most frameworks allow asynchronous logging**



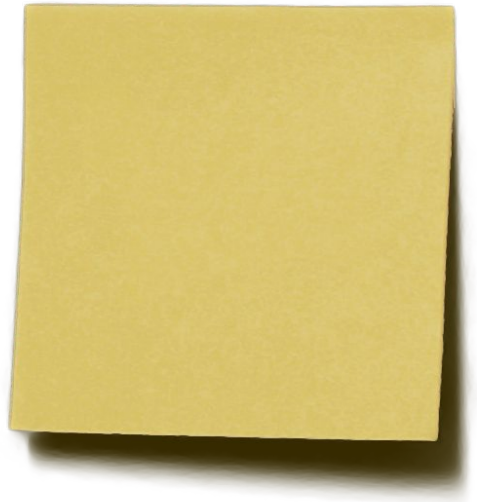
```
<configuration>
  <appender name="FILE" class="c.q.l.core.FileAppender">
    <file>myapp.log</file>
    <encoder>
      <pattern>%logger{35} - %msg%n</pattern>
    </encoder>
  </appender>
  <appender name="ASYNC" class="c.q.l.classic.AsyncAppender">
    <appender-ref ref="FILE" />
  </appender>
  <root level="DEBUG">
    <appender-ref ref="ASYNC" />
  </root>
</configuration>
```

Example: Logback configuration

- **Queue size**
- **Discarding threshold**
- **Never blocks:**
  - Drop messages vs block

Associated meta-data

- Timestamp
- Log level
- Thread name
- Class name
- Method name
- Line number in the file
- etc.

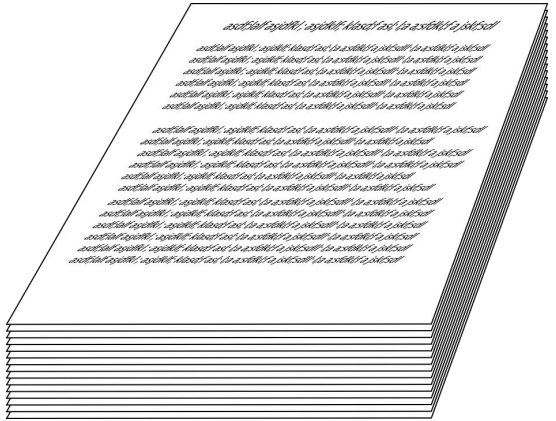


Some metadata is expensive to get

- ***e.g.* line number**
- **Better not compute it**
  - Then, writing it explicitly might be wrong

Logs aggregation

- **Logs by themselves are useless**
- **Centralized Logging Pattern**
  - Elasticsearch
  - Splunk
  - Graylog



Additional metadata

- **File**
- **Host/IP**
- **Environment**
  - *e.g.* “**DEV**” vs “**PROD**”
- **Cloud zone**
- **etc.**

EXOSCALE

@nicolas_frankel

Searching in logs

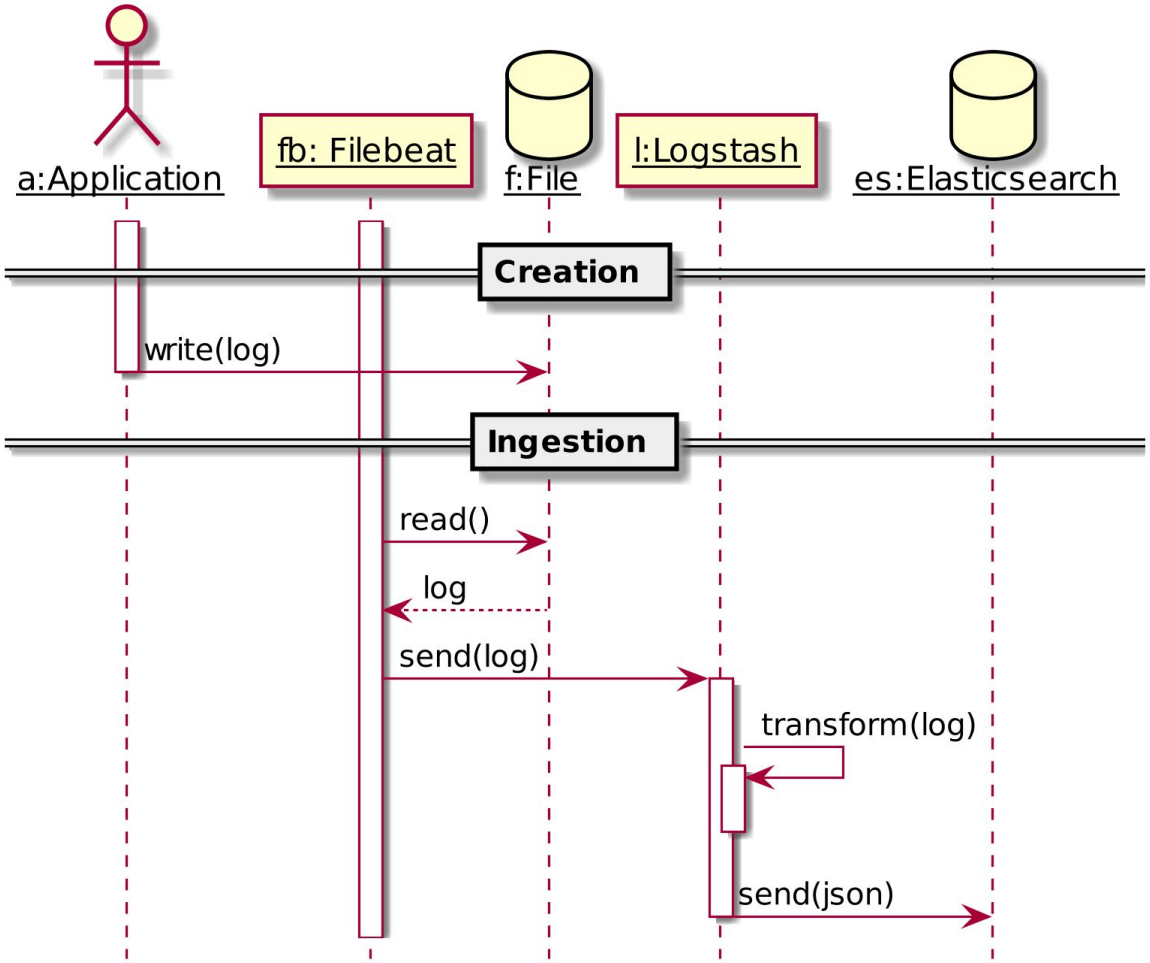
- **Logs are not the end!**
- **Searches are:**
  - “Find me all the logs that happened yesterday on JVMs in PROD”

Schema

On read vs. on write

Example architecture: the Elastic Stack

- **Filebeat**
  - Read log files
- **Logstash**
  - Parsing of log messages



EXOSCALE

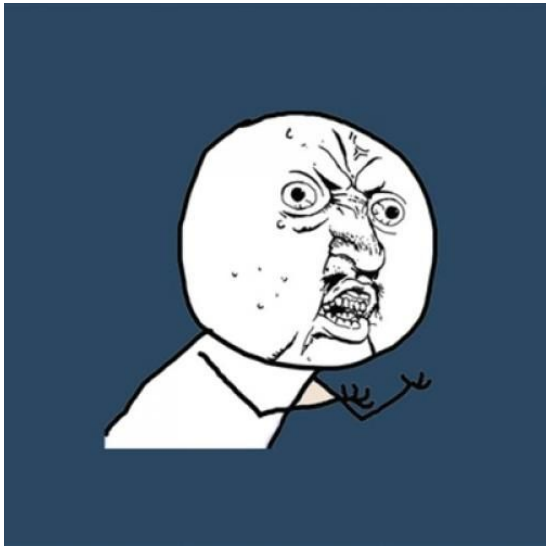
@nicolas_frankel

```
2018-12-17 13:56:54.906  INFO
1 --- [  restartedMain]
c.e.configmgmt.demo.DemoAppli
cation        : Started
DemoApplication in 3.833
seconds (JVM running for
4.303)
```

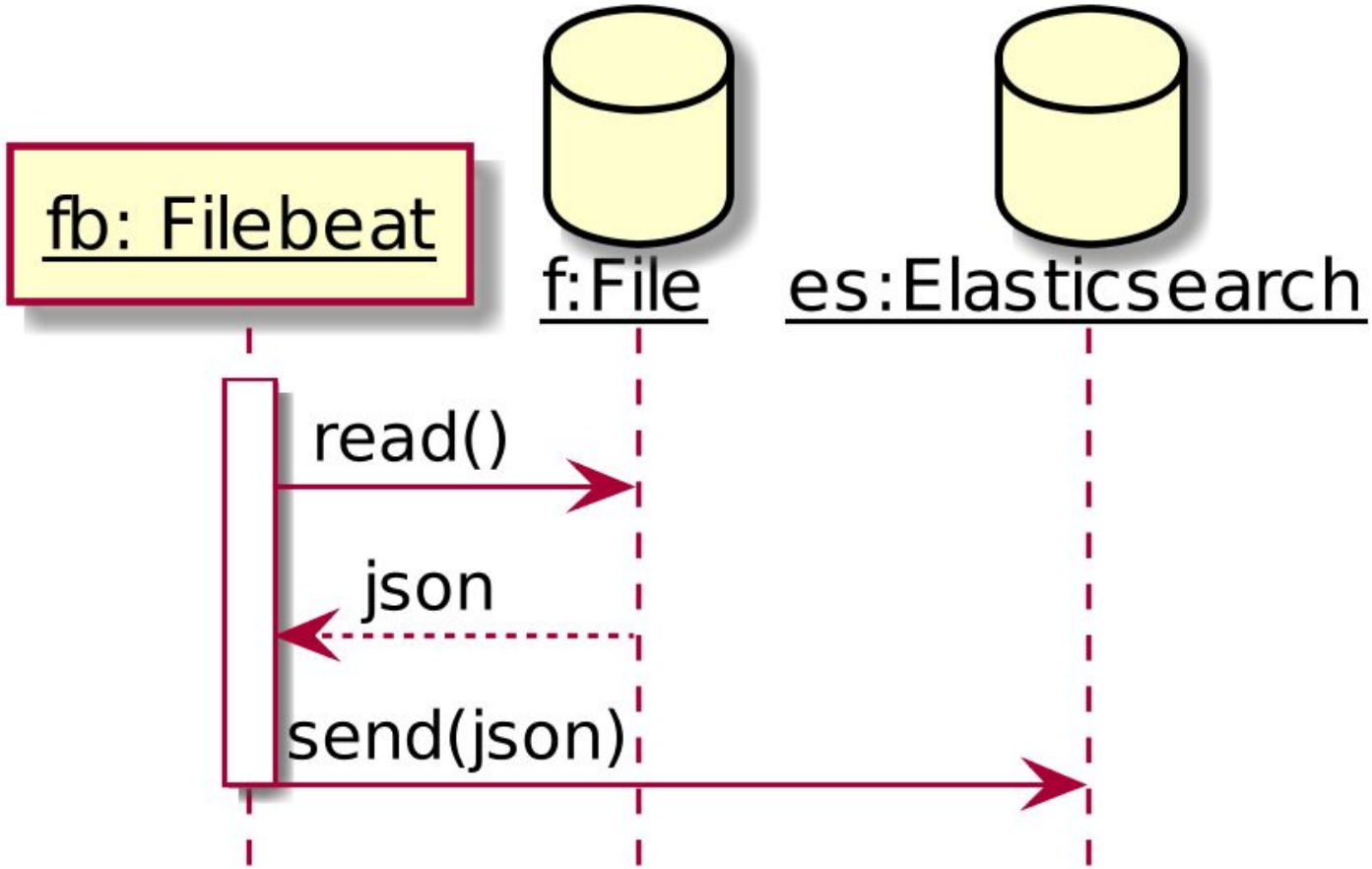
```
{
 "date": [[ "18-12-17" ]],
 "MONTHDAY": [[ "18" ]],
 "MONTHNUM": [[ "12" ]],
 "YEAR": [[ "17" ]],
 "time": [[ "13:56:54.906" ]],
 "HOUR": [[ "13" ]],
 "MINUTE": [[ "56" ]],
 "SECOND": [[ "54.906" ]],
 "level": [[ "INFO" ]],
 "threadName": [[ "restartedMain" ]],
 "class": [[ "c.e.configmgmt.demo.DemoApplication" ]],
 "message":
   [[ "Started DemoApplication in 3.833 seconds (JVM running for
4.303)" ]]
}
```

Why?!

We don’t actually need Logstash if we produce JSON directly!



```
{
 "date": "18-12-17",
 "time": "13:56:54.906",
 "level": "INFO",
 "thread": "restartedMain",
 "class":
"c.e.configmgmt.demo.DemoApplication",
 "message": "Started DemoApplication in
3.833 seconds (JVM running for 4.303)"
}
```

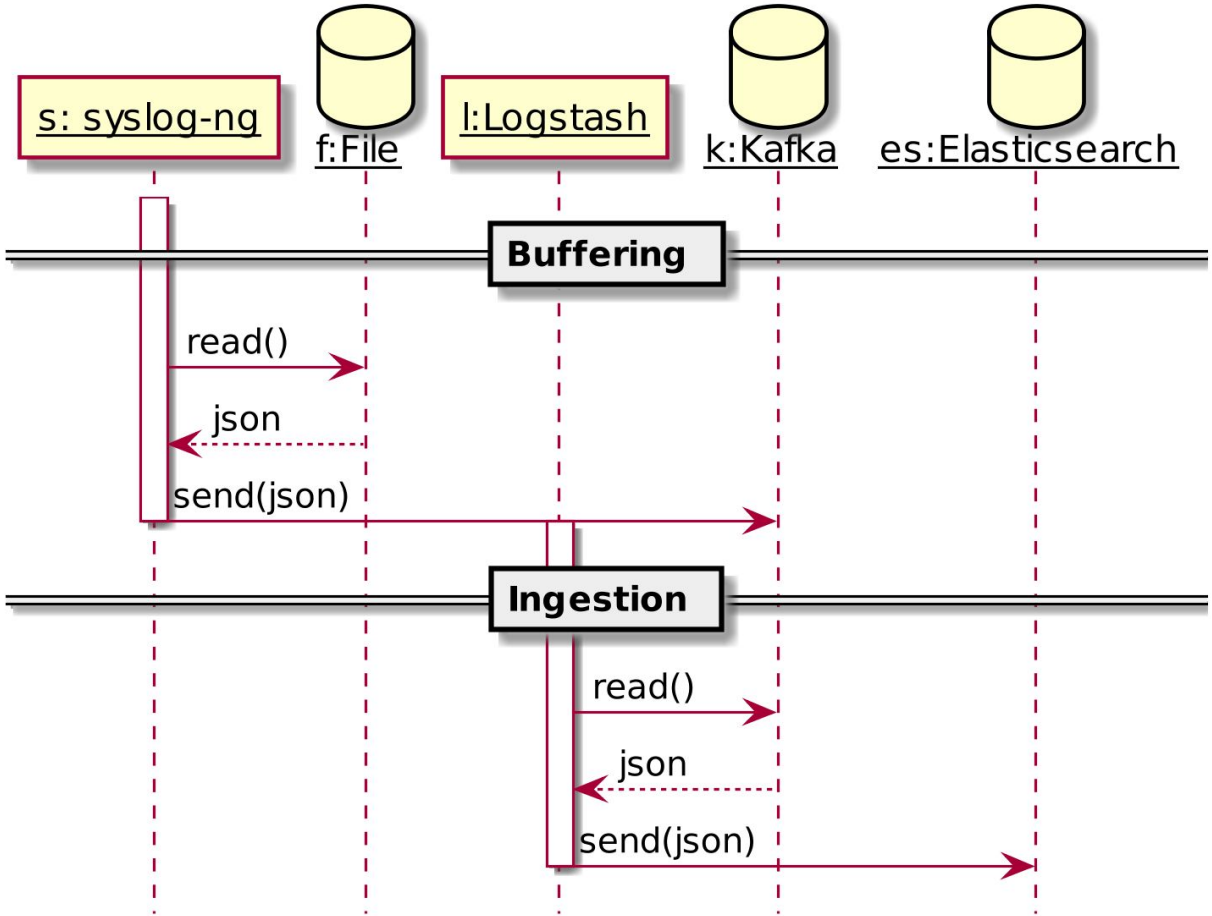

fb: Filebeat
f:File
es:Elasticsearch
read()
json
send(json)

Food for thoughts

- **Events vs. log files**
- **Configuration hot reload**

Logging @ Exoscale

- **syslog-ng**
- **Kafka**


s: syslog-ng
f:File
l:Logstash
k:Kafka
es:Elasticsearch
Buffering
read()
json
send(json)
Ingestion
read()
json
send(json)

Summary

1. **Pass computations instead of results to log statements**
2. **Consider the physical file system to log to**
3. **Go asynchronous if speed counts more than reliability**
4. **Don’t use expensive meta-data**
    - Consider hot reloading configuration just in case
5. **Schema on write is slower, but the alternative is worse**
6. **Send JSON directly**

Takeaway

It’s everyone’s responsibility to have fast logs:

- **Developers**
- **Ops**
- **Architects**

EXOSCALE

@nicolas_frankel

Takeaway #2

It’s a matter of trade-offs:

- **Speed**
- **Reliability**
- **Custom context**

THANKS!

- https://blog.frankel.ch/
- https://exoscale.com/syslog/
- @nicolas_frankel

EXOSCALE

@nicolas_frankel