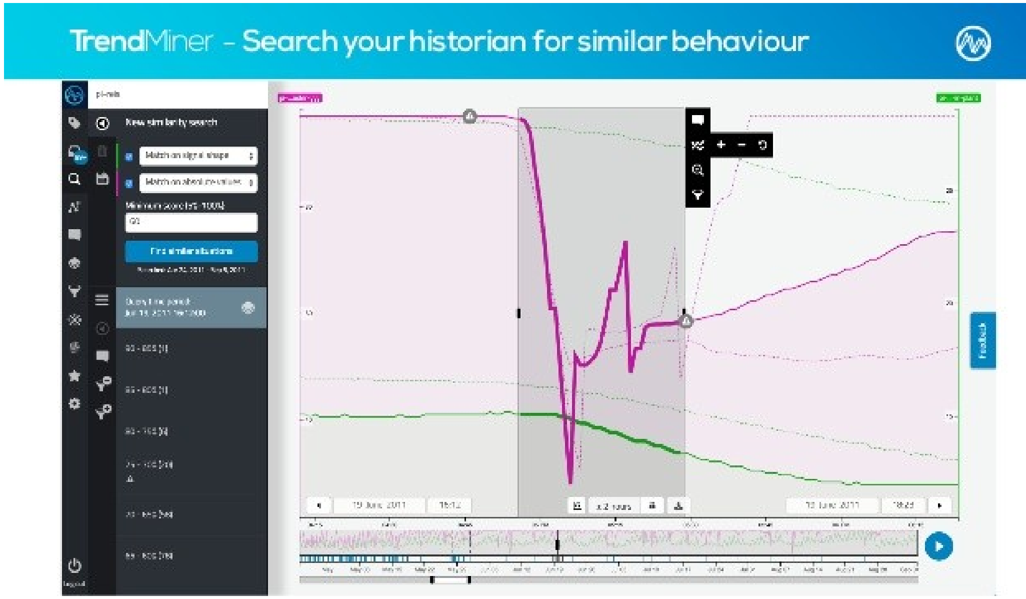
Monitoring and logging

About myself

- Studied computer science at KU Leuven
- 4 years of experience as software engineer at axxes
- Competence coach Java
- Ngdata: 10 months (Big data)
    - Hadoop
- Persgroep + truvo: 1 year
    - Spring services (micro services)
    - Handling automatisation + dockerizing environments
- Trendminer: 2 years
    - Improve analytics for process industry
    - Timeseries data (Ignite + Cassandra)

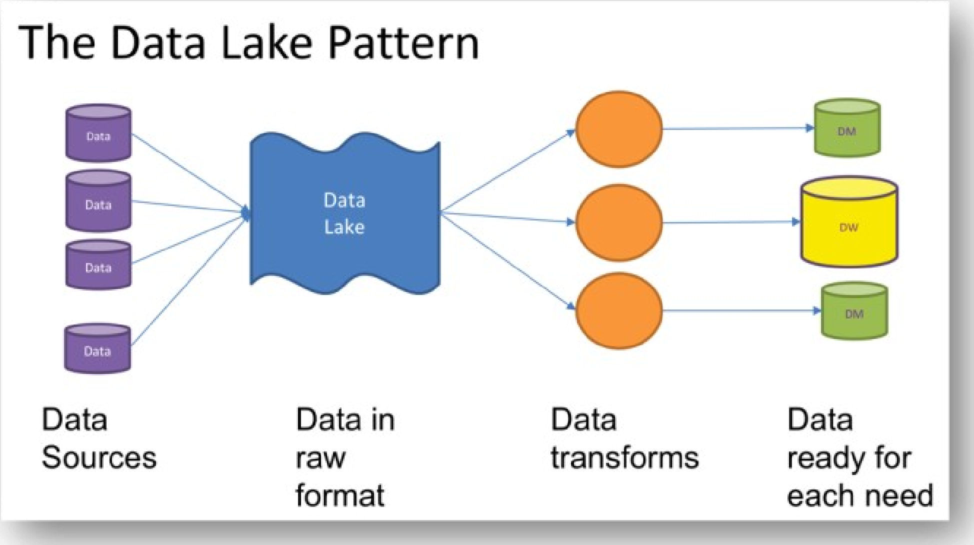
Big Data at Trendminer

- Perform searches through timeseries data
  - Volume: Second level resolution: 400Mb/year/tag
  - Searches:
    - Finding similar patterns
    - Linear regression
    - Predicting maintenance
    - Find abnormal situations
- Goal:
  - Reduce costs
  - Improve efficiency of plants



Big Data at Telenet

- Hadoop cluster: (9Tb memory, 6 Pb storage, 30+ nodes)
- Gather + transform data to suit business
- Batch + stream processing
- Technologies
    - Cloudera stack: hdfs, hive, kudu, impala, yarn
    - Processing data: spark, kafka streams



Who are you?

- Which was the best course so far?
- What is your experience with logging?
  - Did you diagnose issues through logging?
  - How did you analyze logs? Centralized/files on the server?
  - Logging best practices?

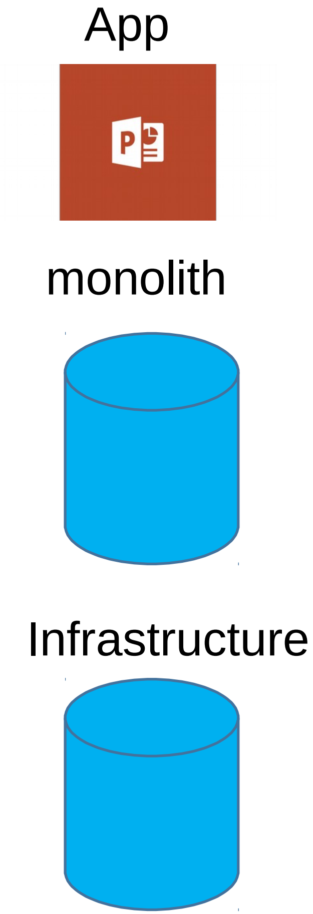
Monitoring and logging

Overview

- Logging
  - Intro
  - What/how/when/where
  - Best practices
- Metrics
  - Application metrics
  - User metrics
  - Best practices

Logging: Intro

- Traditional way of logging
  - Simple
  - Log to files
  - Analyze: look through log file
- Challenges (functional):
  - Log the right thing
    - Too much logging is annoying
    - Too little is even worse
  - How long to keep logs
    - Storage available?
    - Regulation?



App

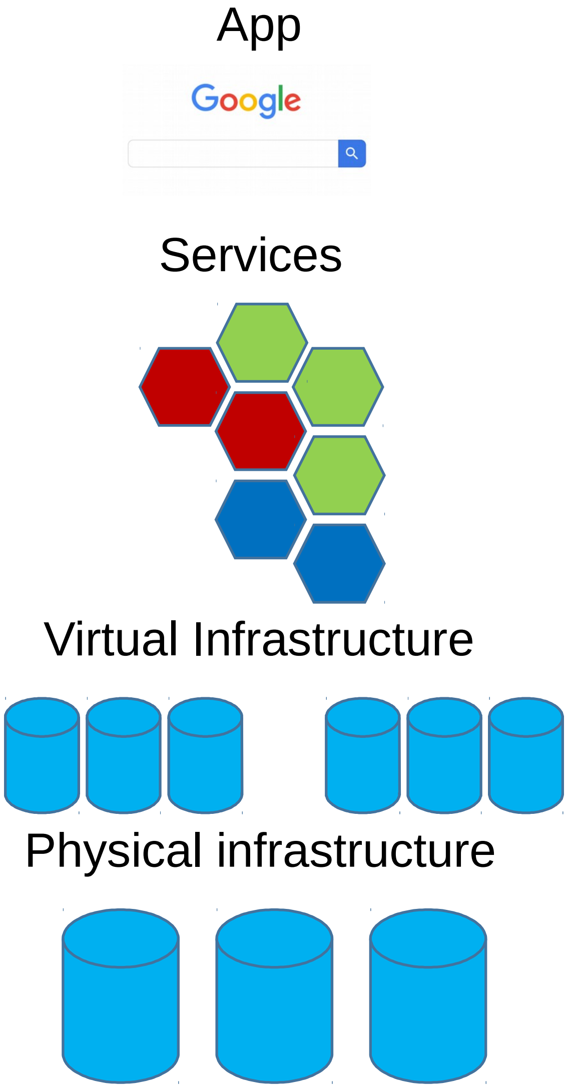
monolith

Infrastructure

Logging: Intro

- Microservices
  - Logs are spread over multiple servers
  - How to trace requests across services?
  - How to analyze all logs?
- Challenges (functional):
  - Log the right thing
  - How long to keep logs
- Challenges (technical)
  - Analyze logs efficiently
  - Trace logs across services

=> complexity increases



App

Services

Virtual Infrastructure

Physical infrastructure

Logging: Intro

- Environment/context of your application is important
  - Can you (developer) access the logs
    - Direct access
    - Indirect: someone needs to send them
  - Does the application have an outbound internet connection
  - Are you allowed to see the logs
    - Regulation?
    - Sensitive information?
      - Remove unnecessary sensitive info (password/username)
  - Can you access the server?
    - Change log levels
    - Intervene when there are issues

Logging: what?

- Exceptions?
- State of the application?
- SQL logging?
- Actions performed?

Logging: what?

Logging: what?

- Exceptions?
  - Interaction with external services?
  - Unexpected behavior?
    NullPointerException/ArythmeticException
  - Expected behavior?
    FileNotFoundException/CancellationException
- How?
  - General exception handler (1 place) = consistency
  - Logging content
    - Exception message
    - Stacktrace
    - Context

Logging: what?

- Application state (background batch process)
  - Log in which state we are
  - history shows state transitions
    - Essential when troubleshooting
    - Reason about what went wrong
- How?
  - Log level: depending on importance
- Trade-off
  - Amount of logging
  - Enough logging to diagnose issues

Logging: what?

- SQL logging
  - Framework support (hibernate)
  - Show the actual SQL statements performed
- How?
  - At most debug level (otherwise too much spam)
  - Useful when developing/not in production
  - Often sensitive information

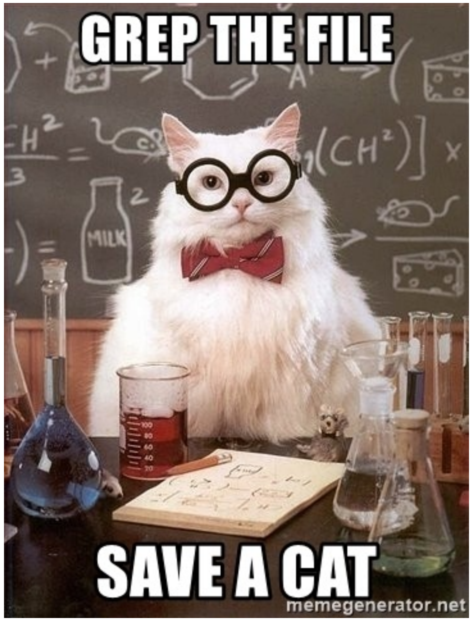
Logging: what?

- Actions performed?
  - Batch process/User interaction without event sourcing
  - Trace of when an operation happened
  - Be wary of amount of logging

Logging: regulations?

- We cannot log whatever we want
  - No passwords: not from the user nor the application (database access)
  - No username in the logs (depends on industry)
  - Other sensitive information
  - Impact on GDPR?
- What to do then
  - Obfuscate sensitive information
  - Make sure it is deterministic (same username to same id)
  - Think about the impact on analyzing issues (trace back to database dumps etc.)

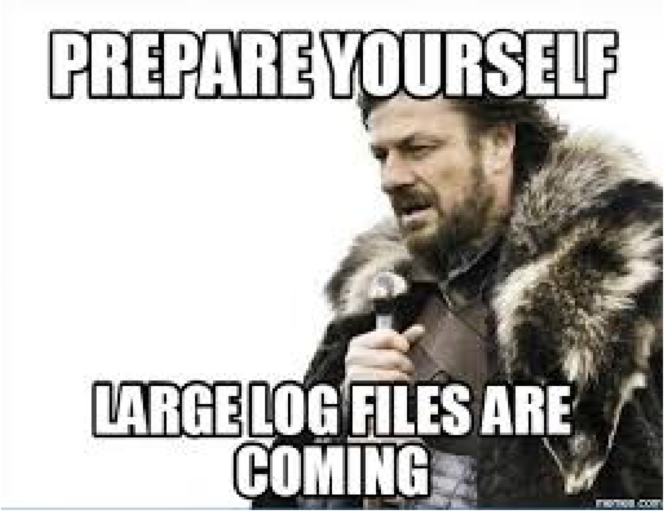
Logging: how?



- Log files are data
  - Structure them accordingly
  - You will analyze them often
    - Using grep/awk/cat
    - Using a query engine (kibana)
- Which one do you like
  - "Could not update entry 123 as t was not found in the database"
  - "error: entry does not exist, id: 123, table: users"
  - "{type: error, data: {message: entry does not exist, id: 123, table: users}}"
- Free text is generally a bad idea
  - Try to use keywords with meaning
  - Use a different log format for stacktraces
    - Easily filter them out

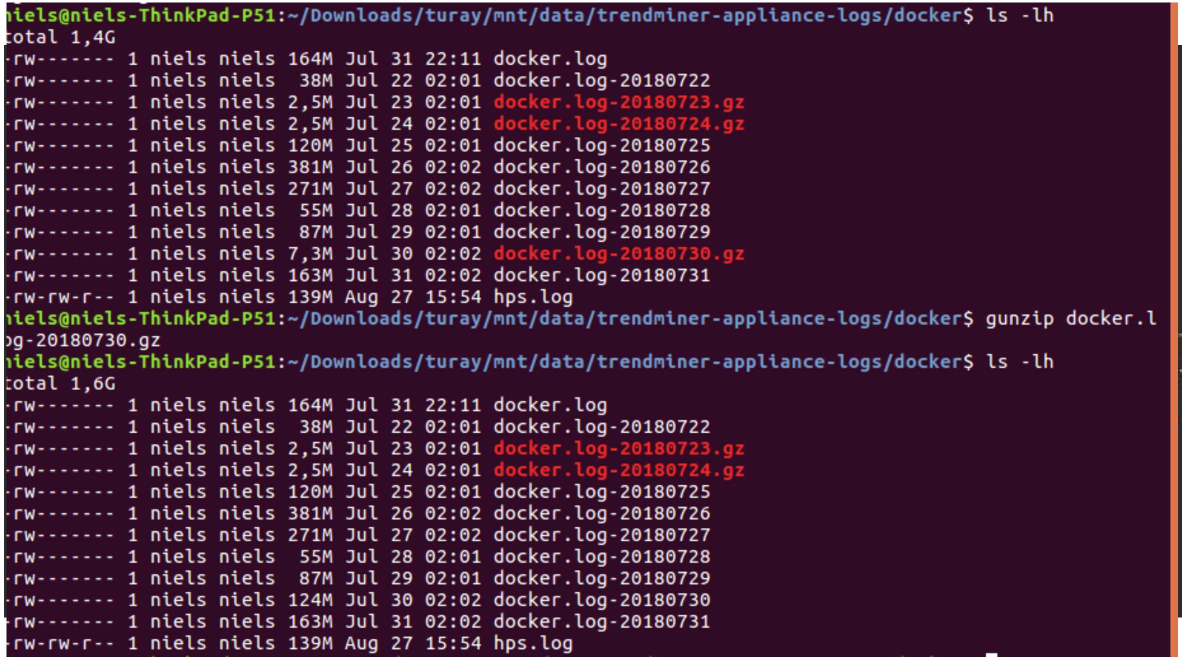
Logging: how much logging?

- Logging can impact performance
- Use log levels
  - Trace/debug/info/warn/error
  - specify log levels for specific packages
    - No debug logging of frameworks
    - Might use debug logging for your app
- Can you alter log level at runtime?
  - Can use more strict log levels



Logging: reduce size of log files?

- Rolling files (date suffix):
  - Maximum size (100 Mb)
  - Maximum file history to keep (10 files)
    => predictable storage requirements
  - Compress old logs (save storage)



```
iels@niels-ThinkPad-P51:~/Downloads/turay/mnt/data/trendminer-appliance-logs/docker$ ls -lh
total 1,4G
-rw------- 1 niels niels 164M Jul 31 22:11 docker.log
-rw------- 1 niels niels  38M Jul 22 02:01 docker.log-20180722
-rw------- 1 niels niels 2,5M Jul 23 02:01 docker.log-20180723.gz
-rw------- 1 niels niels 2,5M Jul 24 02:01 docker.log-20180724.gz
-rw------- 1 niels niels 120M Jul 25 02:01 docker.log-20180725
-rw------- 1 niels niels 381M Jul 26 02:02 docker.log-20180726
-rw------- 1 niels niels 271M Jul 27 02:02 docker.log-20180727
-rw------- 1 niels niels  55M Jul 28 02:01 docker.log-20180728
-rw------- 1 niels niels  87M Jul 29 02:01 docker.log-20180729
-rw------- 1 niels niels 7,3M Jul 30 02:02 docker.log-20180730.gz
-rw------- 1 niels niels 163M Jul 31 02:02 docker.log-20180731
-rw-rw-r-- 1 niels niels 139M Aug 27 15:54 hps.log
iels@niels-ThinkPad-P51:~/Downloads/turay/mnt/data/trendminer-appliance-logs/docker$ gunzip docker.l
og-20180730.gz
iels@niels-ThinkPad-P51:~/Downloads/turay/mnt/data/trendminer-appliance-logs/docker$ ls -lh
total 1,6G
-rw------- 1 niels niels 164M Jul 31 22:11 docker.log
-rw------- 1 niels niels  38M Jul 22 02:01 docker.log-20180722
-rw------- 1 niels niels 2,5M Jul 23 02:01 docker.log-20180723.gz
-rw------- 1 niels niels 2,5M Jul 24 02:01 docker.log-20180724.gz
-rw------- 1 niels niels 120M Jul 25 02:01 docker.log-20180725
-rw------- 1 niels niels 381M Jul 26 02:02 docker.log-20180726
-rw------- 1 niels niels 271M Jul 27 02:02 docker.log-20180727
-rw------- 1 niels niels  55M Jul 28 02:01 docker.log-20180728
-rw------- 1 niels niels  87M Jul 29 02:01 docker.log-20180729
-rw------- 1 niels niels 124M Jul 30 02:02 docker.log-20180730
-rw------- 1 niels niels 163M Jul 31 02:02 docker.log-20180731
-rw-rw-r-- 1 niels niels 139M Aug 27 15:54 hps.log
```

```
TimeBasedRollingPolicy">
leNamePattern>
```

Logging: logging libraries?

- Slf4j (API)
- API implementations
  - Logback
    - Add logback.xml in resources with loggin configuration
  - Log4j2/log4net
  - Commons-logging (do not use)
  - Util logging (issues when running app as executable jar)
- If you use spring
  - <include resource="org/springframework/boot/logging/logback/base.xml"/>
  - Add custom configuration

Logging: logback configuration?

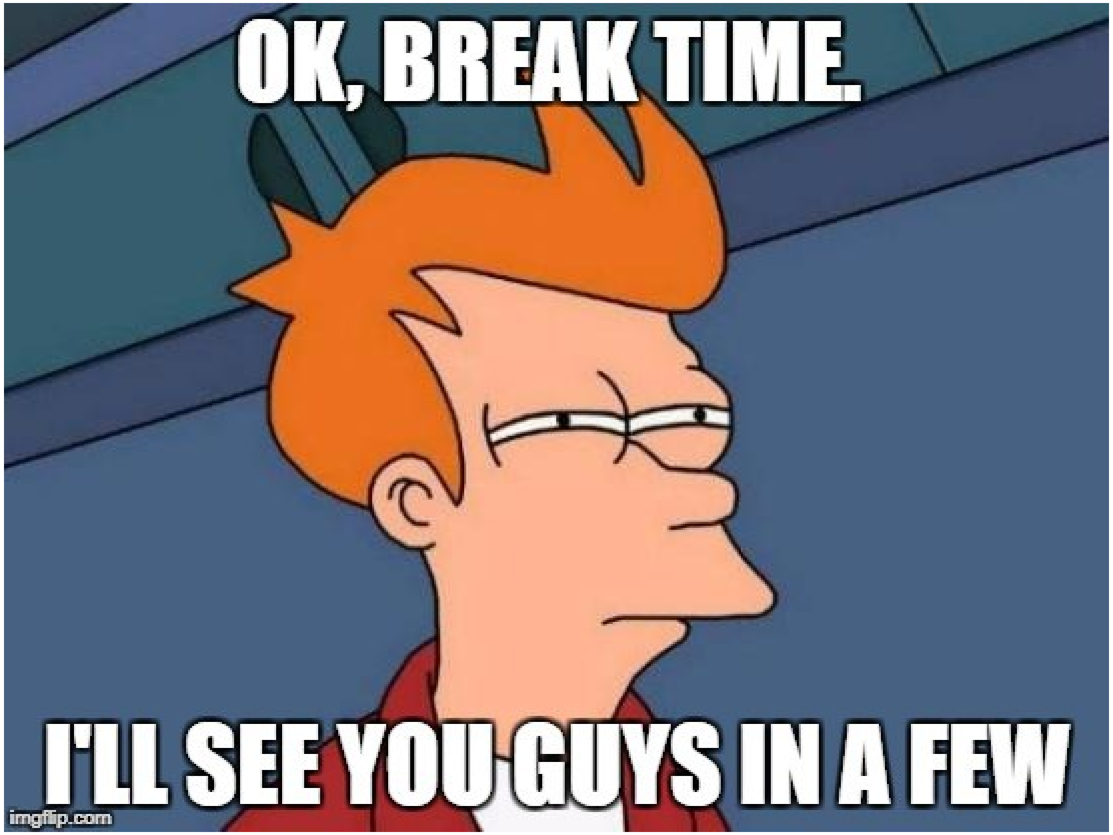
```
<configuration>
    <include resource="org/springframework/boot/logging/logback/defaults.xml" />
    <property name="CONSOLE_LOG_PATTERN" value="${CONSOLE_LOG_PATTERN:-%clr(%d{$
{LOG_DATEFORMAT_PATTERN:-yyyy-MM-dd HH:mm:ss.SSS}}){faint} %clr(${LOG_LEVEL_PATTERN:-%5p}) %clr(${PID:- })
{magenta} %clr(---){faint} %clr([%15.15t]){faint} %clr(%-40.40logger{39}){cyan} %clr(:){faint} %m%n$
{LOG_EXCEPTION_CONVERSION_WORD:-%wEx}}"/>
    <property name="FILE_LOG_PATTERN"
              value="${FILE_LOG_PATTERN:-%d{${LOG_DATEFORMAT_PATTERN:-yyyy-MM-dd HH:mm:ss.SSS}} $
{LOG_LEVEL_PATTERN:-%5p} ${PID:- } --- [%t] %-40.40logger{39} : %m%n${LOG_EXCEPTION_CONVERSION_WORD:-
%wEx}}"/>
    <property name="LOG_FILE" value="${LOG_FILE:-${LOG_PATH:-${LOG_TEMP:-${java.io.tmpdir:-/tmp}}}/spring.log}"/>
    <include resource="org/springframework/boot/logging/logback/console-appender.xml" />
    <include resource="org/springframework/boot/logging/logback/file-appender.xml" />
    <root level="INFO">
        <appender-ref ref="CONSOLE" />
        <appender-ref ref="FILE" />
    </root>

    <root level="WARN"/>
    <logger name="org.springframework.web" level="WARN"/>
</configuration>
```

Logging: best practices?

- Use logging framework: work against API
    - Switch implementations when necessary
- Assign correct log-level + define log levels per package
- Invest in infrastructure to query logs
    - Structure your logs
    - Use ELK stack (see further)
- Use standard log structure:
    - thread name/time/context/event/result
- Log as much as you can (I always regret too little logs)
- Create dashboards if you have access to logs

Logging: Example?

- Logging application (consultants)
  - Customize logback
  - Spring boot actuator
  - Change logging at runtime


OK, BREAK TIME.
I'LL SEE YOU GUYS IN A FEW
imgflip.com

Logging: ELK stack why?

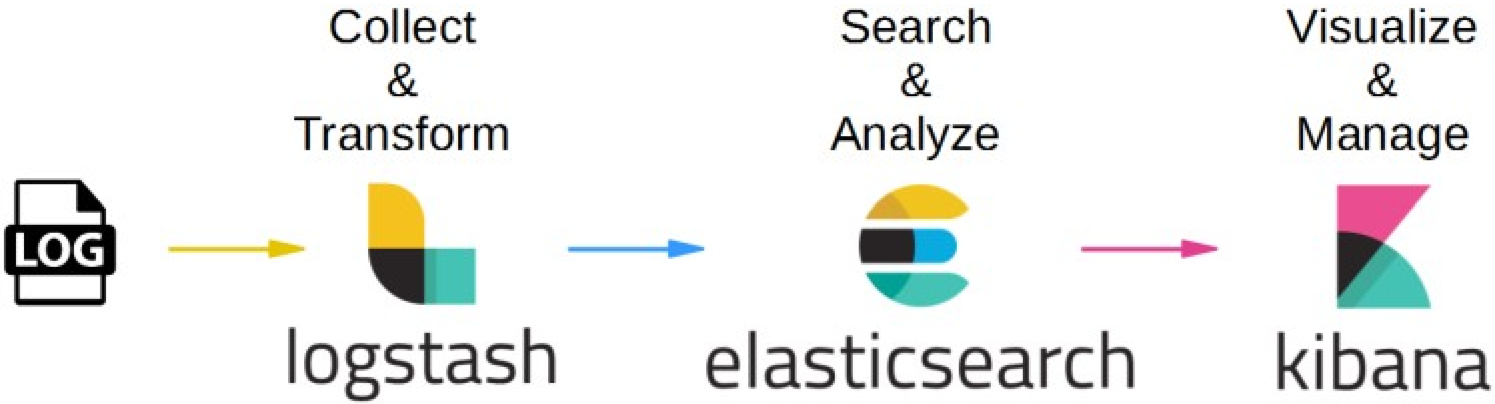
„Can you check the errors from yesterday between 15.02 and 15.07“

Logging: ELK stack why?

- Quickly analyzing logs
    - Central place for all logs
    - Unified structure across all applications
    - Powerful query functionality
- Dashboard for overview of application status
    - Visual instead of textual information
    - Example: number of errors: go up then alert developers

Logging: ELK stack

- A standard for managing log files in microservice environment
  - Parse log files and transfer them
  - Log files are sent to central database (elasticsearch)
  - Querying/analyzing/dashboards are created in kibana

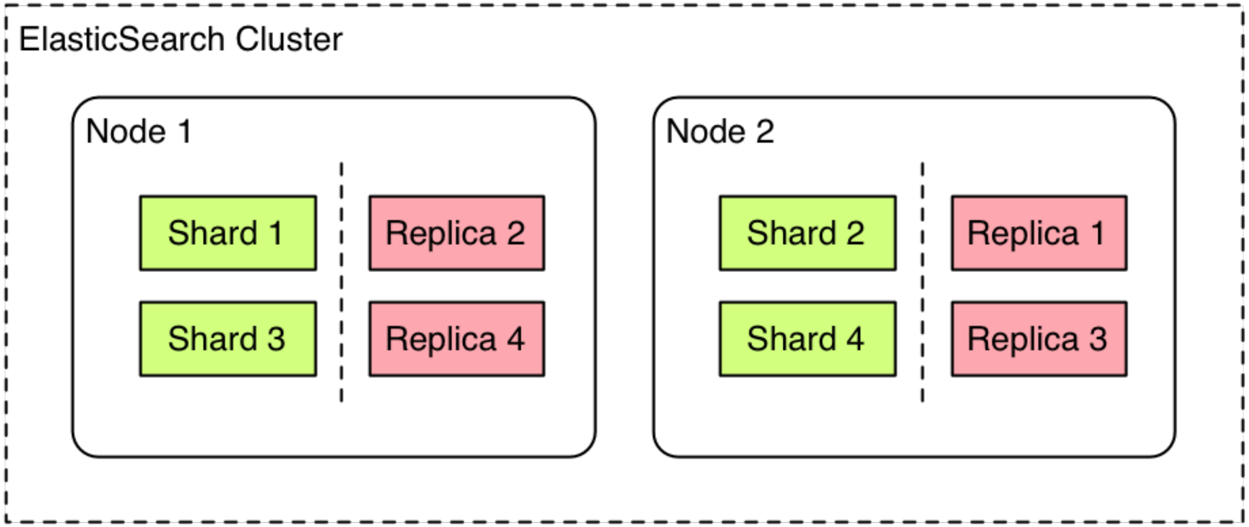


Logging: Elasticsearch

- Sharded, distributed json document store
- Supports full text search, geospatial search, …
- Good REST API to execute searches
- Java
- Apache lucene

Logging: Elasticsearch

- Concepts:
  - Cluster: set of nodes
  - Node: instance (java process
  - Index: collection of documents (exist many in a cluster)
  - Document: basic unit that can be searched/indexed
    → all properties in document can have a specific type
  - Shard: every index is split in multiple shards: distribute load across nodes

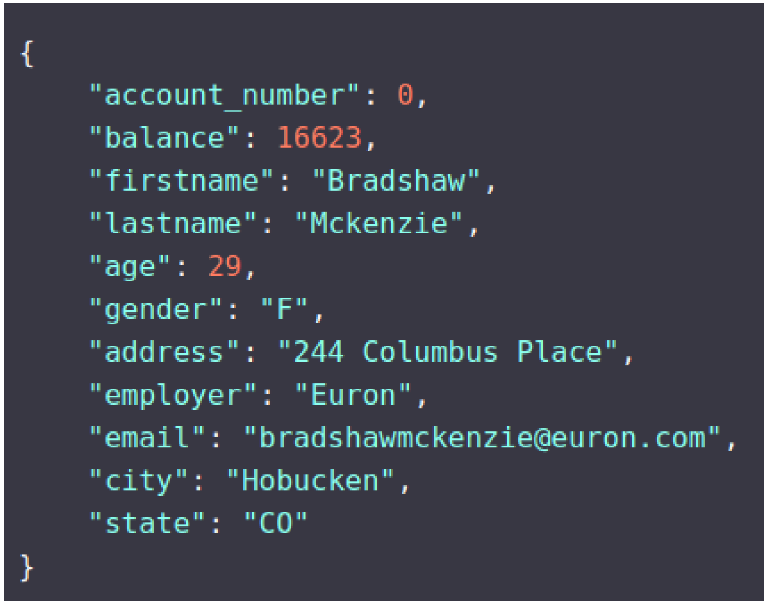


Demo: Elasticsearch

- Start elasticsearch:
  $ docker run -d -p 9200:9200 -p 9300:9300 -e "discovery.type=single-node" elasticsearch:latest
- Exercises:
  - Insert a customer document with your name
    - Query the document by id
    - Query the document by name

Demo: Elasticsearch

- Insert accounts dataset
  curl -H "Content-Type: application/json" -XPOST "localhost:9200/bank/_doc/_bulk" --data-binary "@accounts.json"
- Query documents sorted by account_number
  - alter number of results
- Query documents with mill address and only display balance
- Query documents for anyone who is 40

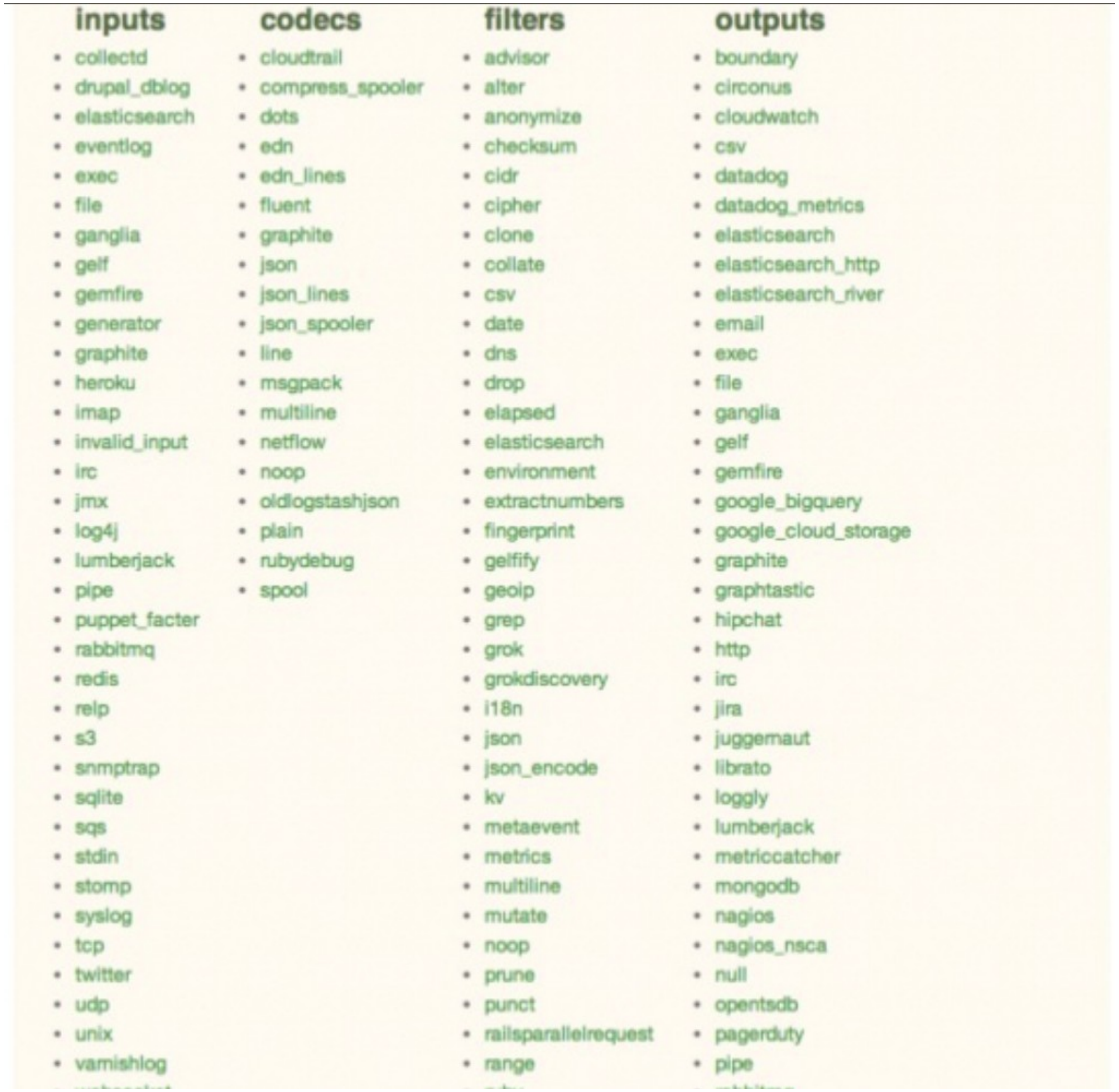


```
{
    "account_number": 0,
    "balance": 16623,
    "firstname": "Bradshaw",
    "lastname": "Mckenzie",
    "age": 29,
    "gender": "F",
    "address": "244 Columbus Place",
    "employer": "Euron",
    "email": "bradshawmckenzie@euron.com",
    "city": "Hobucken",
    "state": "CO"
}
```

Logging: Logstash

- Plumbing for your logs
  - Different log structure
  - Different date formats
- Many different inputs exists
  - file,kafka queue,...
- Filtering/parsing of logs
- Many different outputs exists
  - Redis, elasticsearch, file

Logging: Logstash



inputs

- collectd
- drupal_dblog
- elasticsearch
- eventlog
- exec
- file
- ganglia
- gelf
- gemfire
- generator
- graphite
- heroku
- imap
- invalid_input
- irc
- jmx
- log4j
- lumberjack
- pipe
- puppet_facter
- rabbitmq
- redis
- relp
- s3
- snmptrap
- sqlite
- sqs
- stdin
- stomp
- syslog
- tcp
- twitter
- udp
- unix
- varnishlog

codecs

- cloudtrail
- compress_spooler
- dots
- edn
- edn_lines
- fluent
- graphite
- json
- json_lines
- json_spooler
- line
- msgpack
- multiline
- netflow
- noop
- oldlogstashjson
- plain
- rubydebug
- spool

filters

- advisor
- alter
- anonymize
- checksum
- cidr
- cipher
- clone
- collate
- csv
- date
- dns
- drop
- elapsed
- elasticsearch
- environment
- extractnumbers
- fingerprint
- gelfify
- geoip
- grep
- grok
- grokdiscovery
- i18n
- json
- json_encode
- kv
- metaevent
- metrics
- multiline
- mutate
- noop
- prune
- punct
- railsparallelrequest
- range

outputs

- boundary
- circonus
- cloudwatch
- csv
- datadog
- datadog_metrics
- elasticsearch
- elasticsearch_http
- elasticsearch_river
- email
- exec
- file
- ganglia
- gelf
- gemfire
- google_bigquery
- google_cloud_storage
- graphite
- graphtastic
- hipchat
- http
- irc
- jira
- juggernaut
- librato
- loggly
- lumberjack
- metriccatcher
- mongodb
- nagios
- nagios_nsca
- null
- opentsdb
- pagerduty
- pipe

Logging: Logstash

```
input {
    file {
      path => "/tmp/access_log"
      start_position => "beginning"
  }
}

filter {
  grok {
    match => { "message" => "%{COMBINEDAPACHELOG}" }
  }
  date {
    match => [ "timestamp" , "dd/MMM/yyyy:HH:mm:ss Z" ]
  }
}

output {
  elasticsearch { hosts => ["localhost:9200"] }
  stdout { codec => rubydebug }
}
```

Logging: Grok basics

- Combines text patters into something that matches your log
  - %{SYNTAX:SEMANTIC}
    - Syntaxt: name of pattern that will match your text
    - Semantic: identifier you give to that text being matched

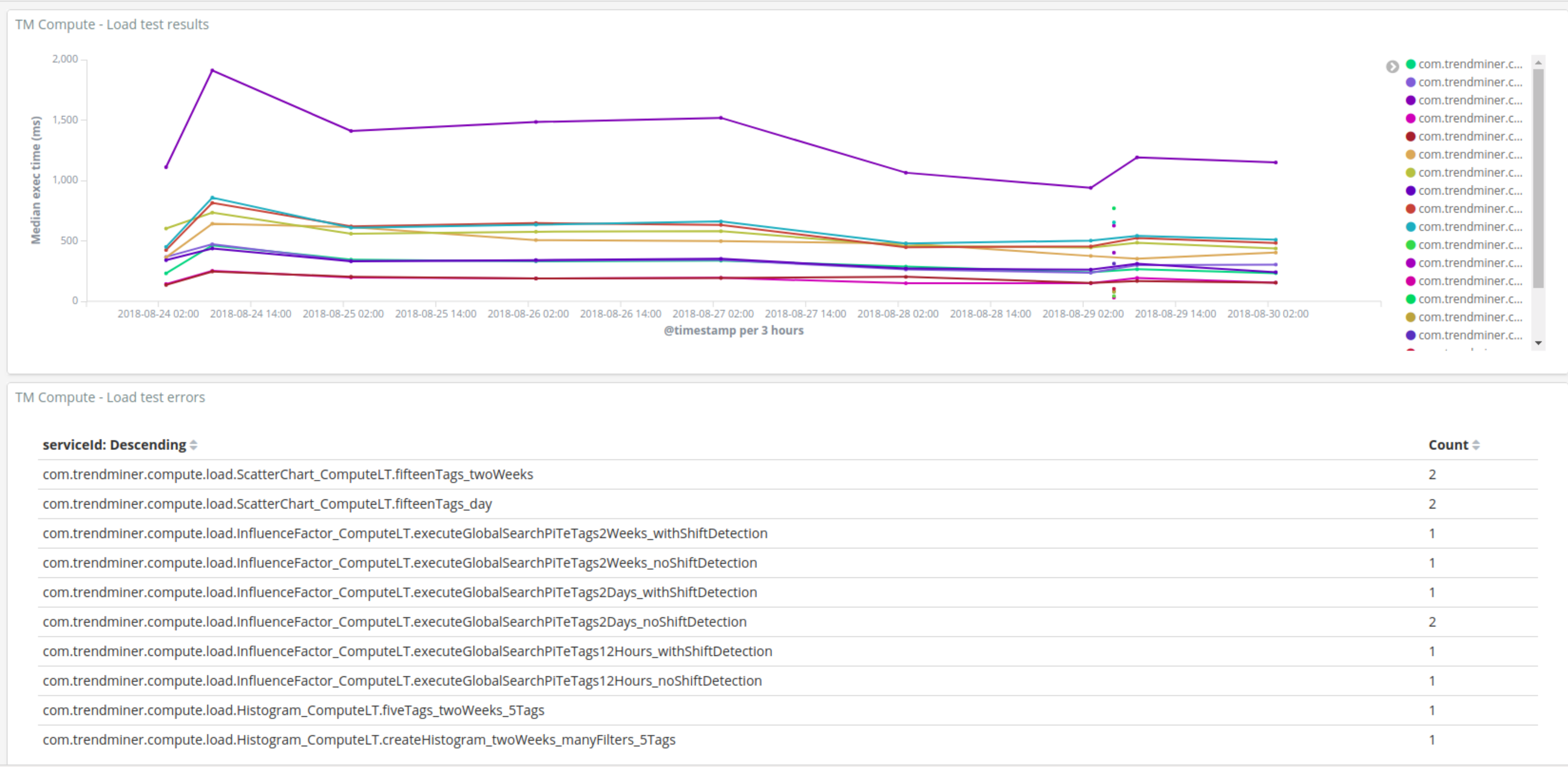
Examples

```
55.3.244.1 GET /index.html 15824 0.043
```

```
%{IP:client} %{WORD:method} %{URIPATHPARAM:request} %{NUMBER:bytes} %
{NUMBER:duration}
```

Logging: Kibana

- Configurable dashboard
- Real-time functionality
- Slice your logs stored in elasticsearch



serviceId: Descending	Count
com.trendminer.compute.load.ScatterChart_ComputeLT.fifteenTags_twoWeeks	2
com.trendminer.compute.load.ScatterChart_ComputeLT.fifteenTags_day	2
com.trendminer.compute.load.InfluenceFactor_ComputeLT.executeGlobalSearchPiTeTags2Weeks_withShiftDetection	1
com.trendminer.compute.load.InfluenceFactor_ComputeLT.executeGlobalSearchPiTeTags2Weeks_noShiftDetection	1
com.trendminer.compute.load.InfluenceFactor_ComputeLT.executeGlobalSearchPiTeTags2Days_withShiftDetection	1
com.trendminer.compute.load.InfluenceFactor_ComputeLT.executeGlobalSearchPiTeTags2Days_noShiftDetection	2
com.trendminer.compute.load.InfluenceFactor_ComputeLT.executeGlobalSearchPiTeTags12Hours_withShiftDetection	1
com.trendminer.compute.load.InfluenceFactor_ComputeLT.executeGlobalSearchPiTeTags12Hours_noShiftDetection	1
com.trendminer.compute.load.Histogram_ComputeLT.fiveTags_twoWeeks_5Tags	1
com.trendminer.compute.load.Histogram_ComputeLT.createHistogram_twoWeeks_manyFilters_5Tags	1

Logging: ELK stack example

- Setup ELK stack locally with docker-compose
  - Kibana
  - Elasticsearch
  - Logstash
- Configure logstash to analyze logs of consultants application
  - Errors when key constraint violation
  - Debug entries when validation fails
  - Illustrate kibana

Logging: example metric beat

- Setup with docker-compose
    - Kibana
    - Elasticsearch
    - Metricbeat: aggregate metrics from host machine
- Look at the configuration
- Show the visualizations/charts

Application metrics

Application metrics: What?

- Measure health of each service
  - Response times
  - Load on the service
  - GC/memory usage
- Measure performance of each service
  - Interaction with other service
  - Job execution time
- Measure node status
  - Allocation of memory
  - Load on the system

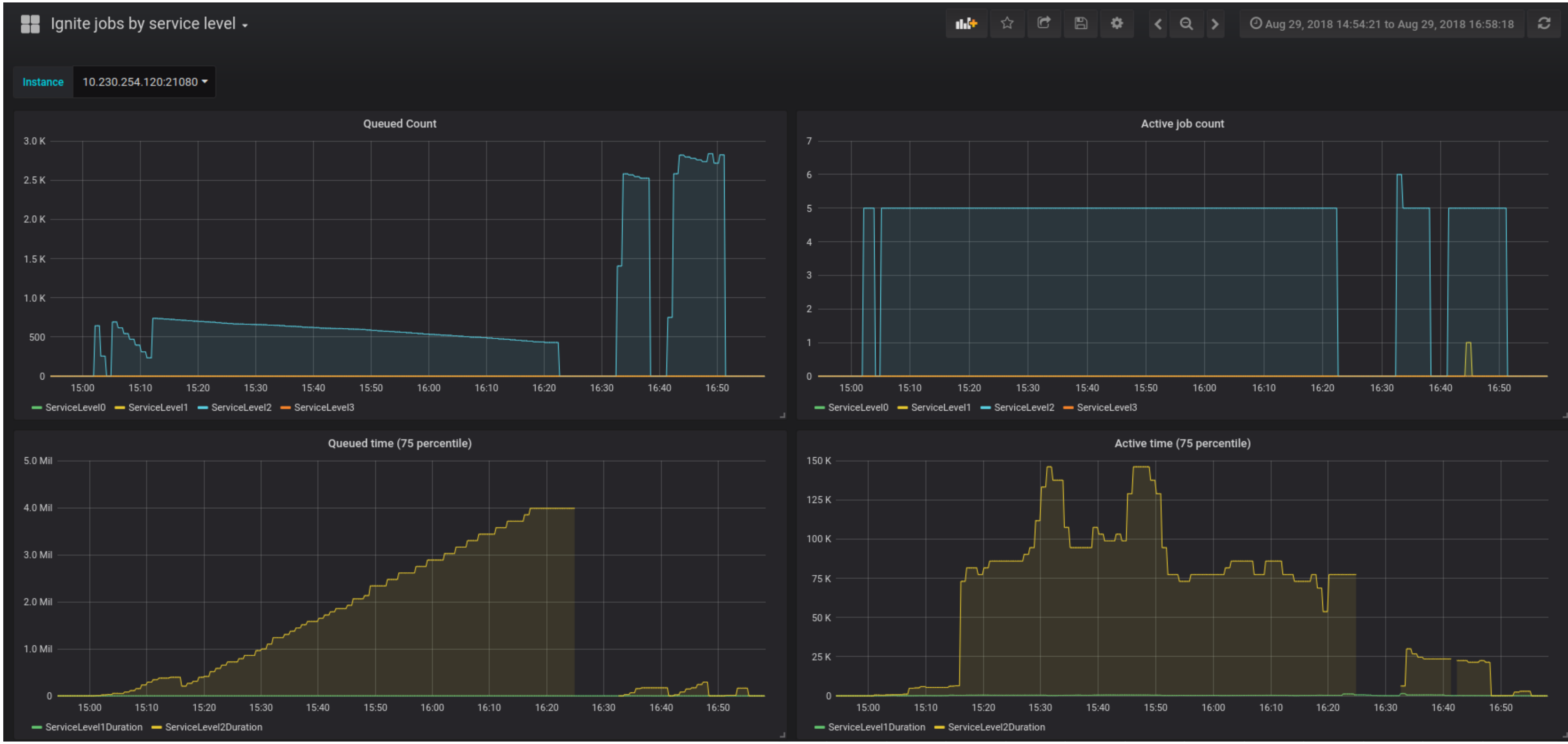
Application metrics: Why?

- Health check
  - Used to detect whether service is up (consul, eureka)
  - Load balance over multiple services
- Measure performance of application
  - detect which operations are slow and why
  - Reject/short-circuit calls if too many are queued
- Helps to diagnose why the application is slow
  - Can be that the host is overloaded (host metrics)
  - A dependent service is not responding
  - The application is doing GC (not enough memory/leak)
- Analyze non functional requirements
  - Define service levels based on them

Application metrics: How?

- Spring-boot: micrometer
  - Counters
  - Histograms: timing of operation
  - Expose these metrics
    - JMX
    - Web page
- Prometheus
  - Time series database
  - Scrape service endpoints
- Grafana
  - Visualize timeseries

Application metrics: Grafana



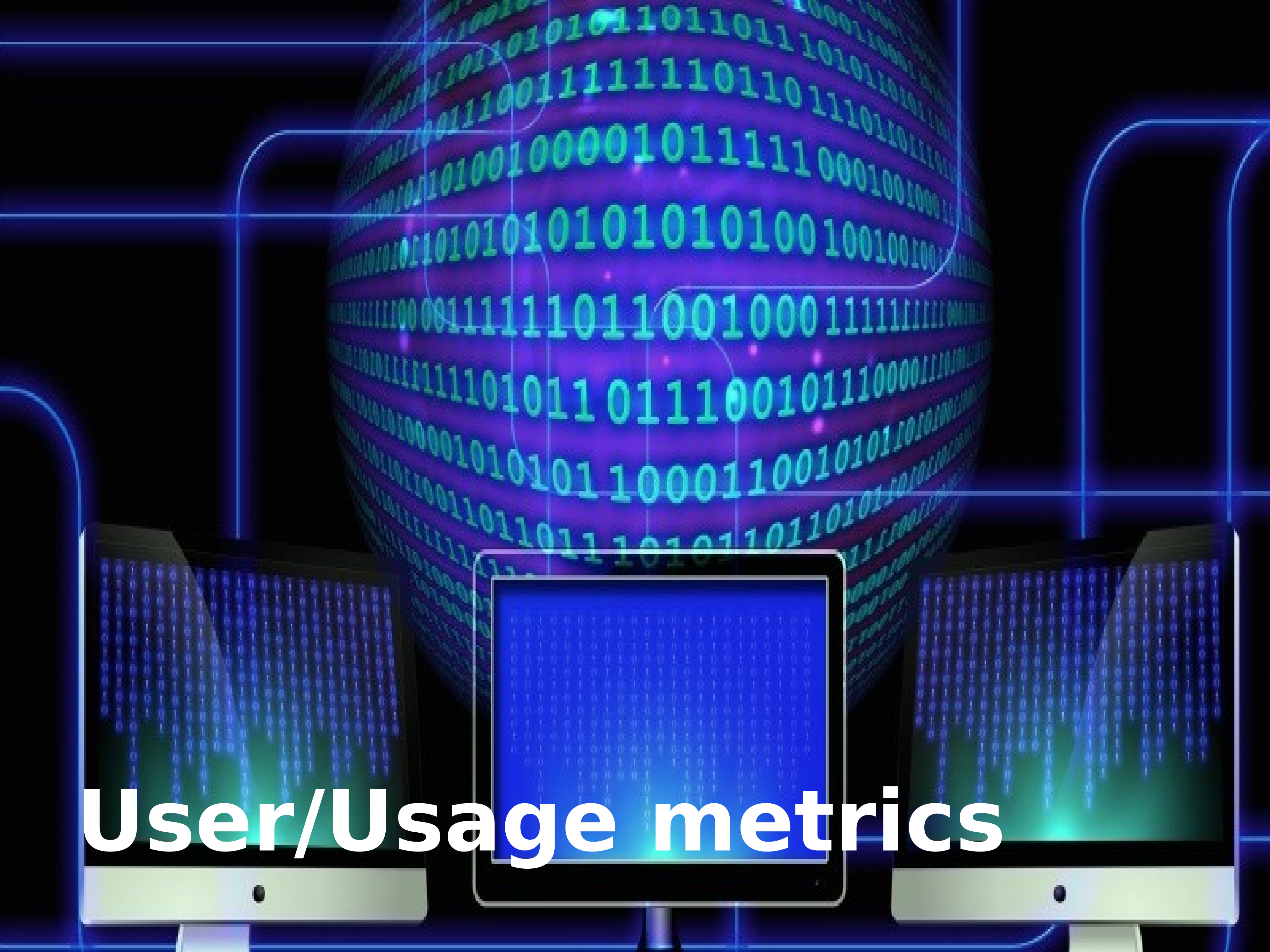
Ignite jobs by service level

Instance 10.230.254.120:21080

Aug 29, 2018 14:54:21 to Aug 29, 2018 16:58:18

Application metrics: Example

- Add metrics to application
- Scrape metrics with prometheus
- Display metrics with grafana



User/Usage metrics

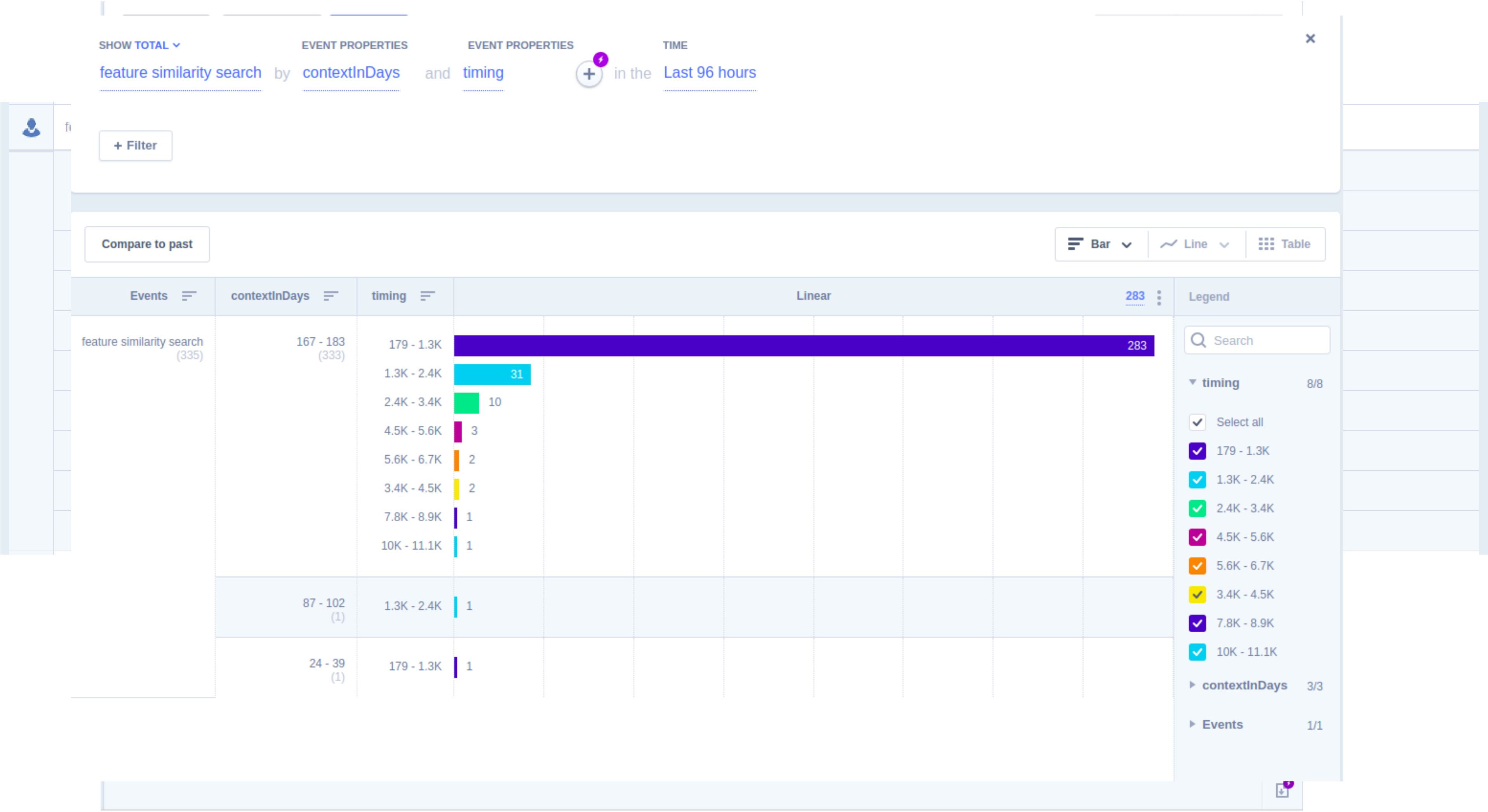
User metrics: Why?

- Gain insights into how users use the application
  - Useful when optimizing functionality
    - Optimize for 95% of use (e.g. Recommendations)
  - Analyze A/B testing: what is preferred
  - Gives info in which feature is rarely used
    - User cannot find it? → make it easily accessible
    - User does not know it? → document it
- Create user profiles
  - User who likes sports/politics/financial news
  - Based on this profile you can personalize the content

User metrics: How?

- Google analytics/mixpanel
- Log actions that a user takes
    - Provide context information: who + what + extra properties
    - Richer content than is possible in metrics
- Similar to how logging is handled
    - Central library that logs events
    - Use a pre-defined structure for all events
    - Easy analysis afterwards

User metrics: Example



SHOW TOTAL ⌄ feature similarity search by EVENT PROPERTIES contextInDays and EVENT PROPERTIES timing + in the TIME Last 96 hours ×

+ Filter

Compare to past

Bar ⌄ | Line ⌄ | Table

Events	contextInDays	timing	Linear 283
feature similarity search (335)	167 - 183 (333)	179 - 1.3K	283
		1.3K - 2.4K	31
		2.4K - 3.4K	10
		4.5K - 5.6K	3
		5.6K - 6.7K	2
		3.4K - 4.5K	2
		7.8K - 8.9K	1
		10K - 11.1K	1
	87 - 102 (1)	1.3K - 2.4K	1
	24 - 39 (1)	179 - 1.3K	1

Legend

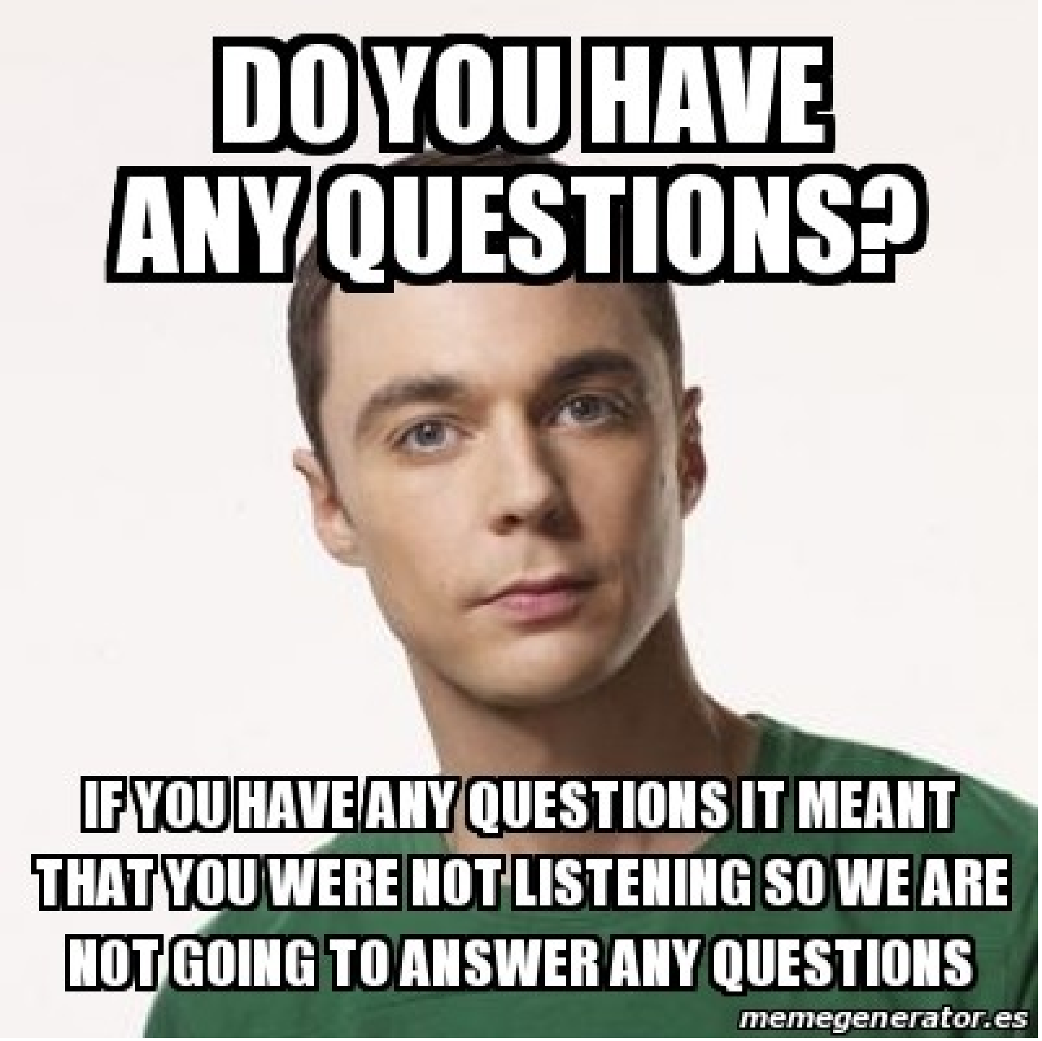
Search

▾ timing 8/8

- ☑ Select all
- ☑ 179 - 1.3K
- ☑ 1.3K - 2.4K
- ☑ 2.4K - 3.4K
- ☑ 4.5K - 5.6K
- ☑ 5.6K - 6.7K
- ☑ 3.4K - 4.5K
- ☑ 7.8K - 8.9K
- ☑ 10K - 11.1K

▸ contextInDays 3/3

▸ Events 1/1


DO YOU HAVE
ANY QUESTIONS?
IF YOU HAVE ANY QUESTIONS IT MEANT
THAT YOU WERE NOT LISTENING SO WE ARE
NOT GOING TO ANSWER ANY QUESTIONS
memegenerator.es

Feedback

https://bit.ly/2P83fYU