Elasticsearch under the hood

Maintaining performance in a distributed system

Alexander Reelsen
@spinscale
alexander.reelsen@elasticsearch.com

elasticsearch.

?

Agenda

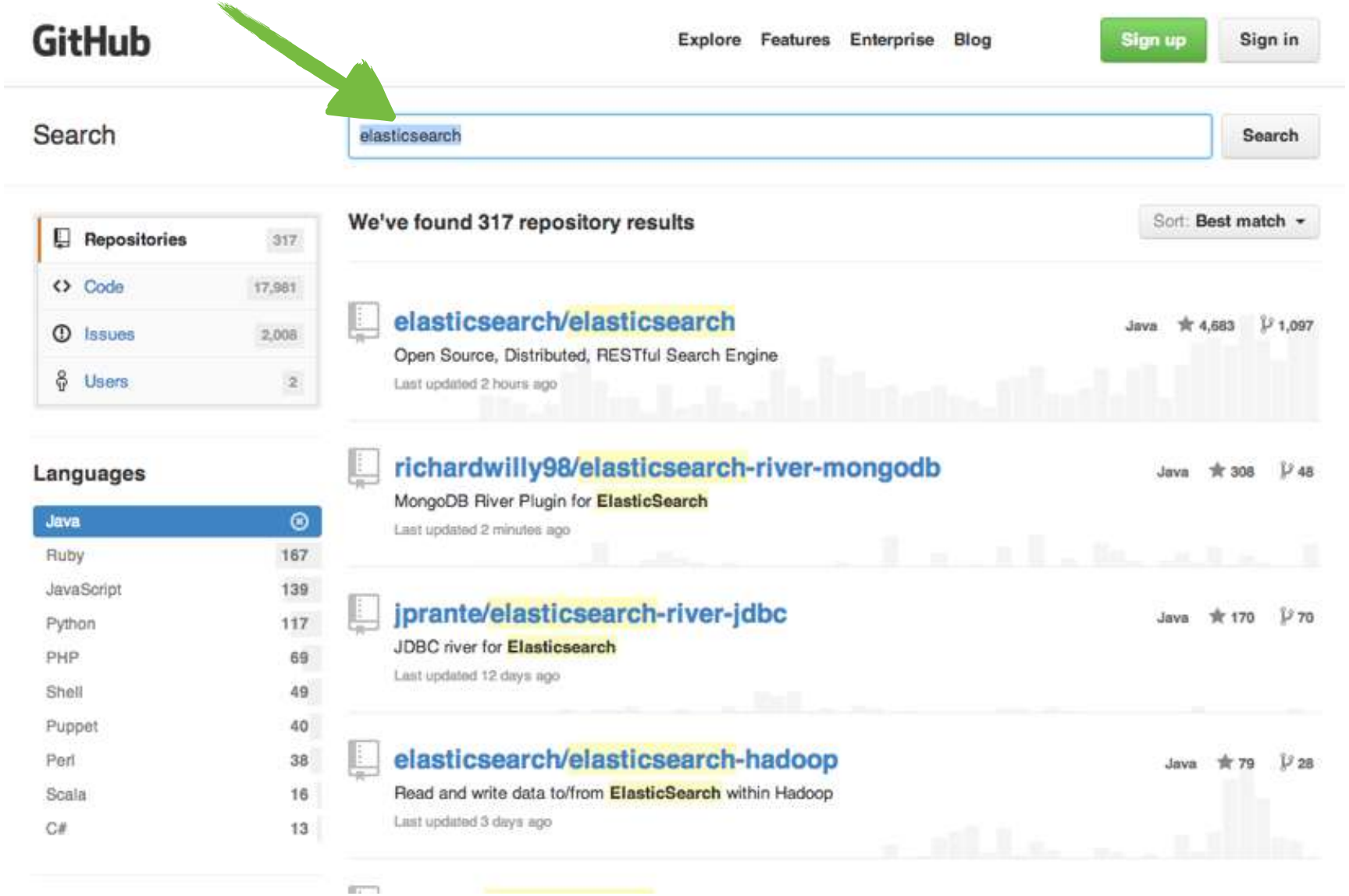
- Introduction, first steps, scalability
- Hardware & Operating system
- JVM
- Garbage collection
- Libraries
- Elasticsearch

about

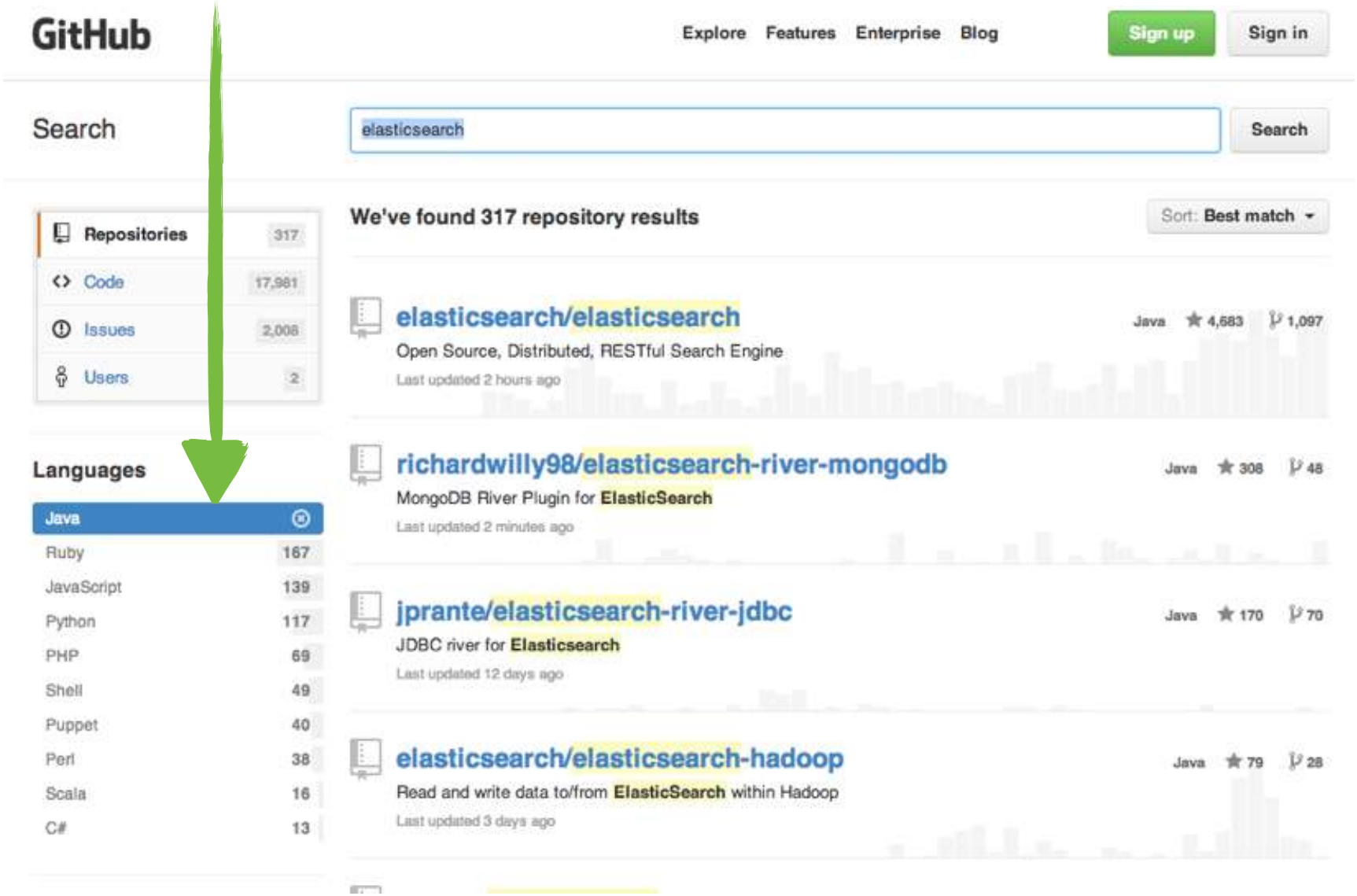
- Me
  - Interested in metrics, ops and the web
  - Likes the JVM
  - Working with elasticsearch since 2011
- Elasticsearch, founded in 2012
  - Products: Elasticsearch, Logstash, Kibana, Marvel
  - Professional services: Support & development subscriptions
  - Trainings

Introduction

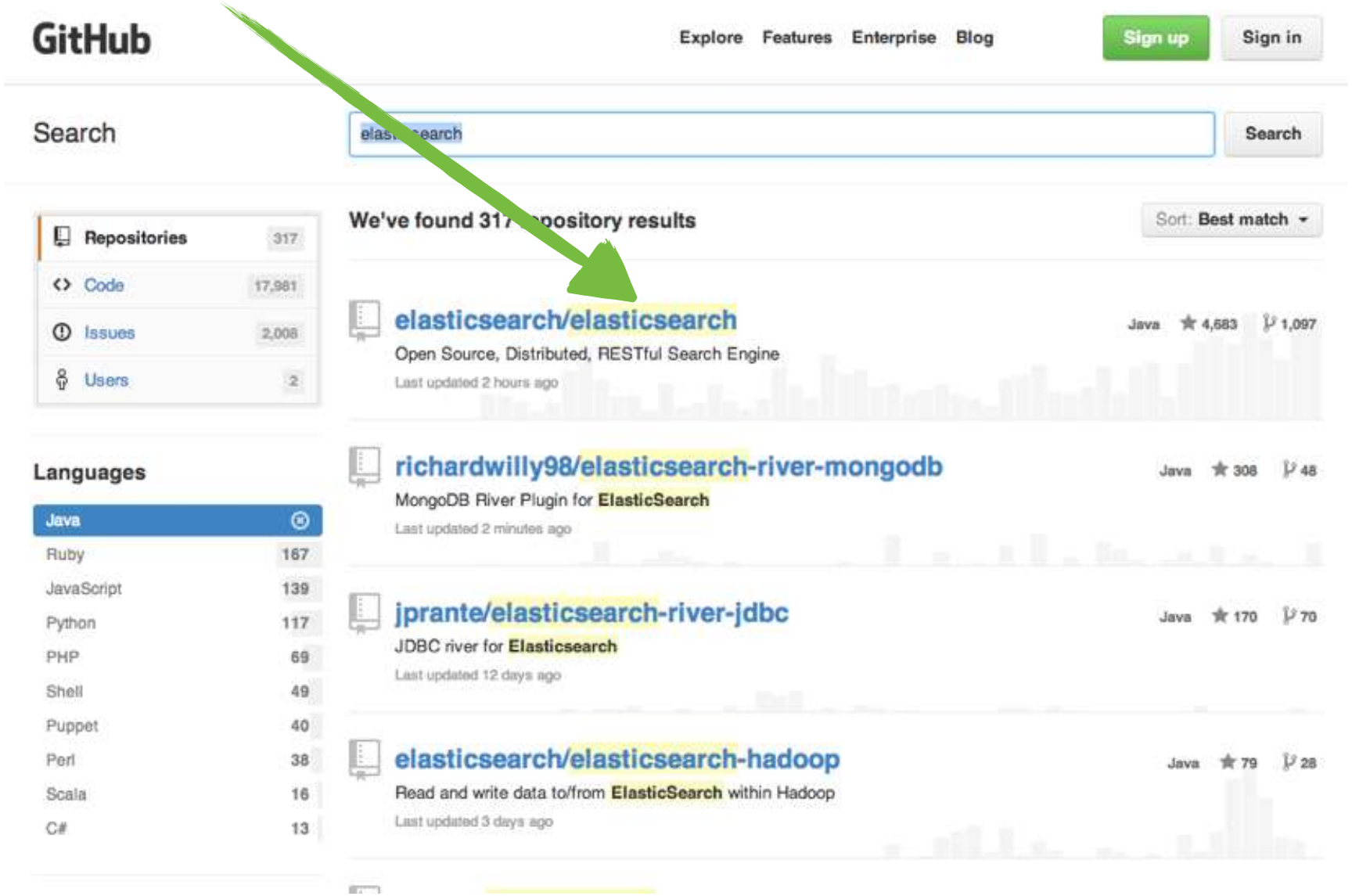
Unstructured search



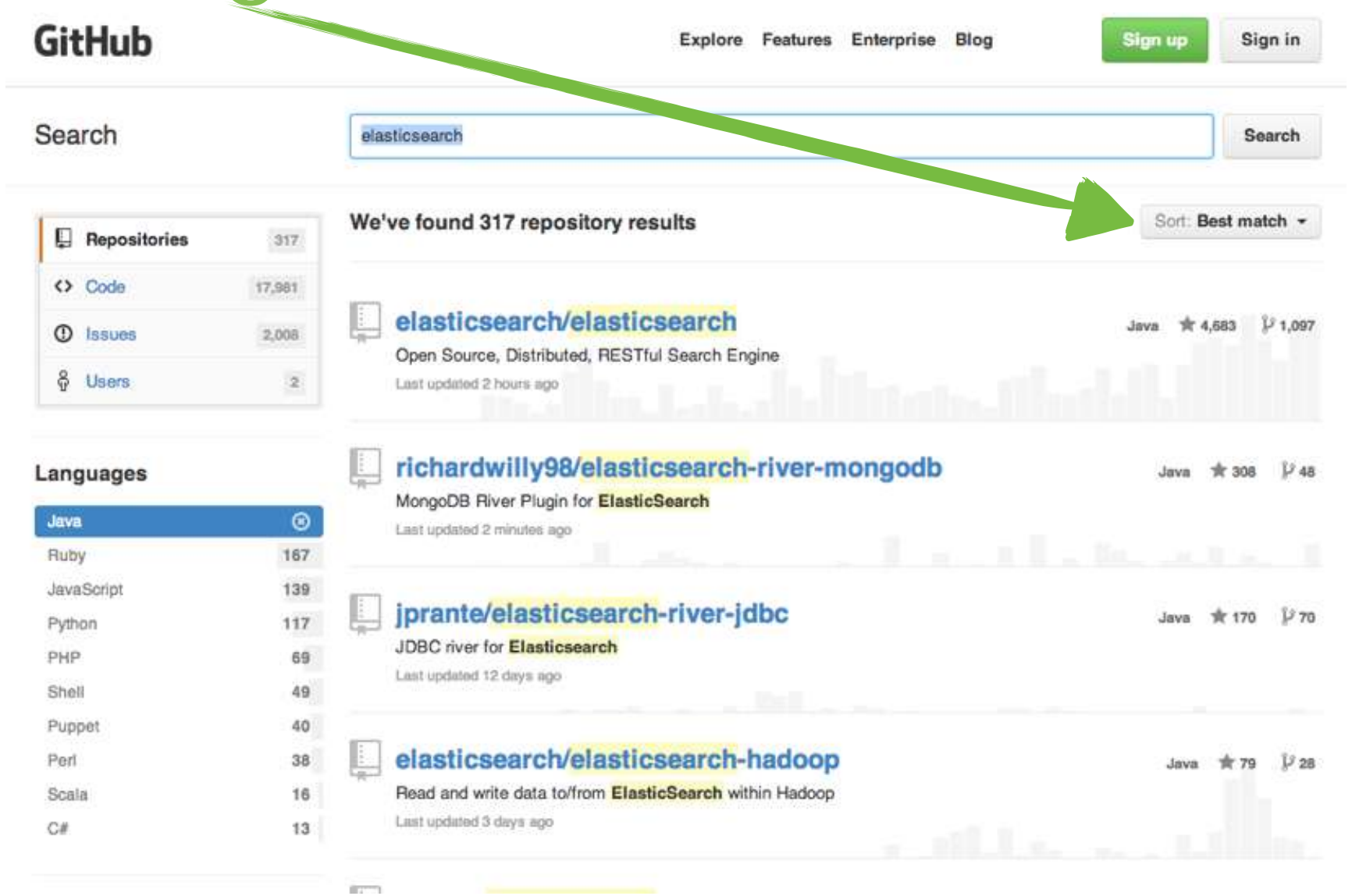
Structured search



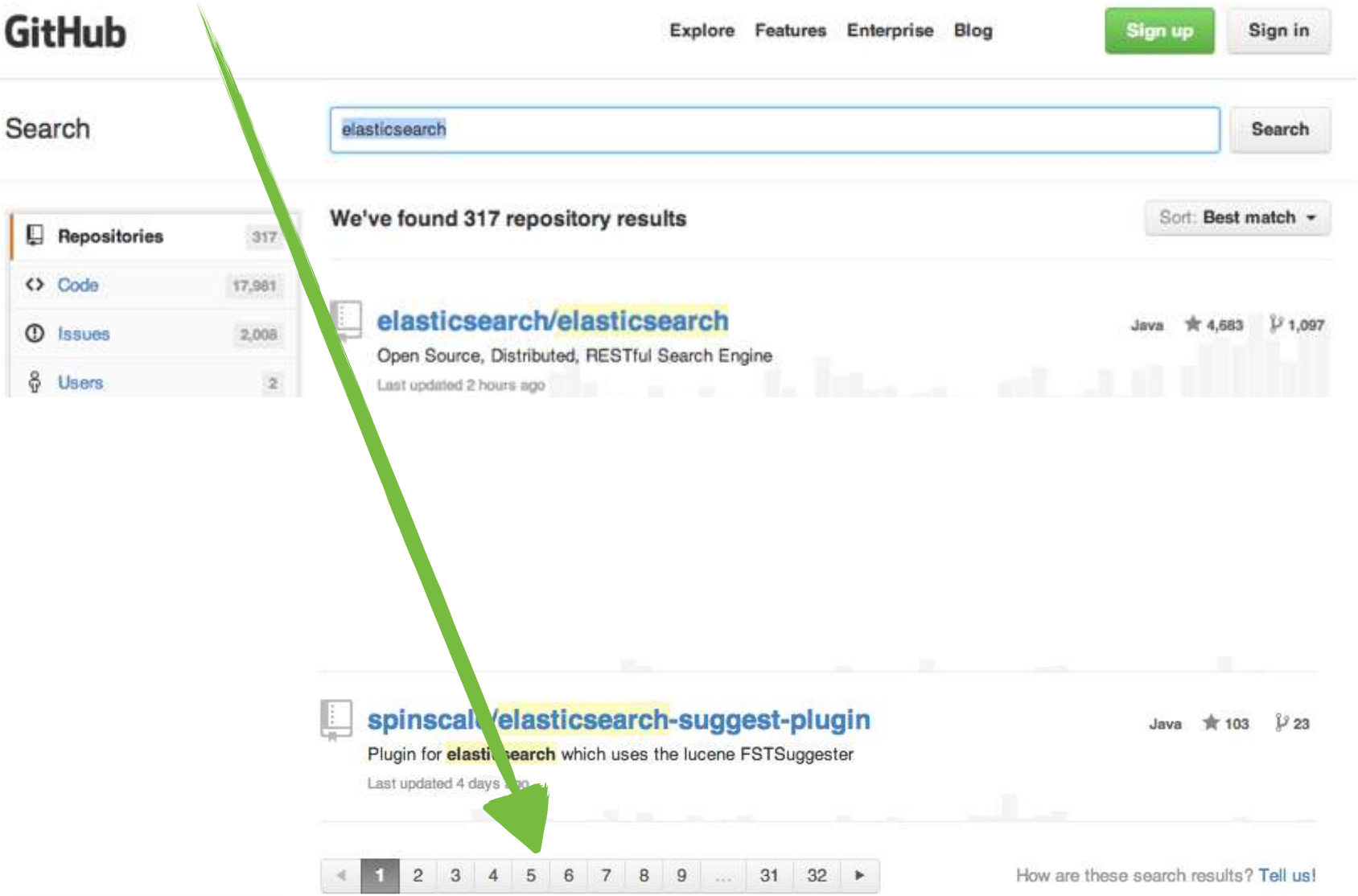
Enrichment



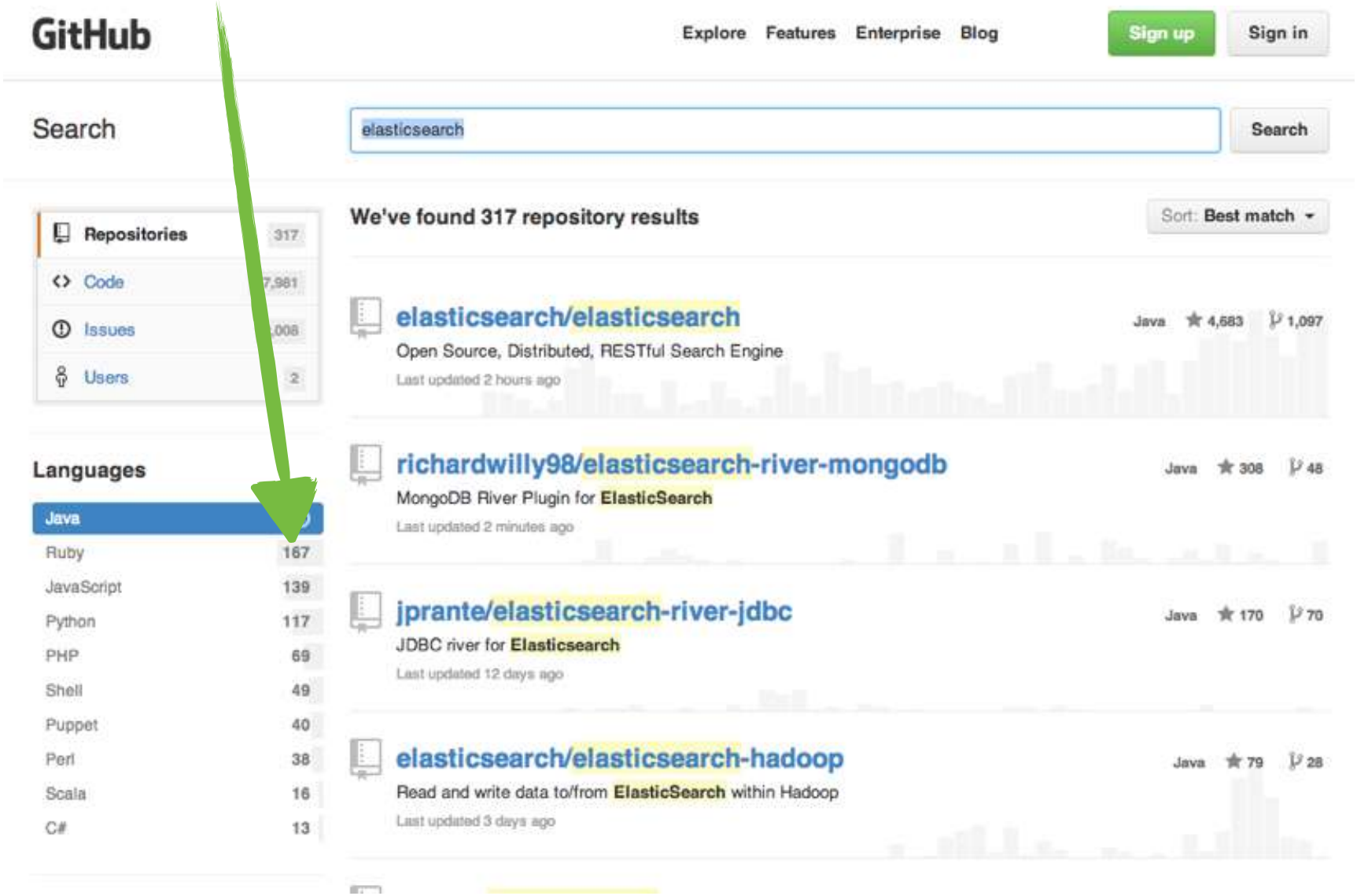
Sorting



Pagination

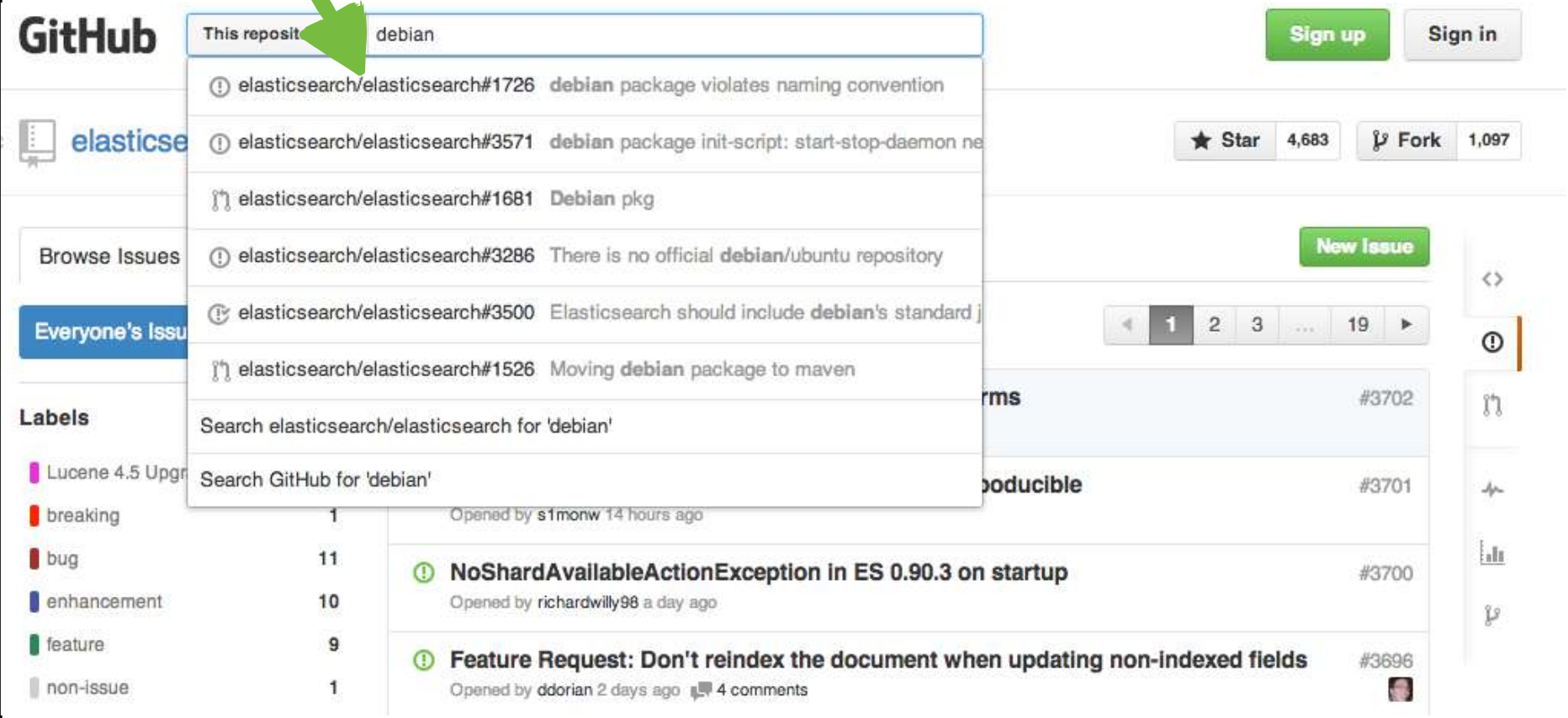


Aggregation



elasticsearch.

Suggestions



Elasticsearch in 10 seconds

- Schema-free, REST & JSON based distributed document store
- Open Source: Apache License 2.0
- Zero configuration
- Written in Java, extensible

Installation & first steps

Zero configuration

```
$ wget https://download.elasticsearch.org/...
$ tar -xf elasticsearch-1.1.0.tar.gz
$ ./elasticsearch-1.1.0/bin/elasticsearch
...
[2014-01-19 14:53:11,508][INFO ][node] [Scanner] started
...
```

Is it alive?

```
» curl localhost:9200
{
  "status" : 200,
  "name" : "Scanner",
  "version" : {
    "number" : “1.1.0",
    "build_hash" : "e018cda7e7a32643d59e0ac3cdb412ccc239af04",
    "build_timestamp" : "2014-01-17T15:11:47Z",
    "build_snapshot" : true,
    "lucene_version" : “4.7.0"
  },
  "tagline" : "You Know, for Search"
}
```

Create…

```
» curl -XPUT localhost:9200/books/book/1 -d '
{
  "title" : "Elasticsearch - The definitive guide",
  "authors" : "Clinton Gormley",
  "started" : "2013-02-04",
  "pages" : 230
}'
```

Update…

```
» curl -XPUT localhost:9200/books/book/1 -d '
{
  "title" : "Elasticsearch - The definitive guide",
  "authors" : [ "Clinton Gormley", "Zachary Tong" ],
  "started" : "2013-02-04",
  "pages" : 230
}'
```

Delete…

```
» curl –X DELETE localhost:9200/books/book/1
```

Realtime GET…

```
» curl –X GET localhost:9200/books/book/1
» curl –X GET localhost:9200/books/book/1/_source
```

Search

```
» curl -XGET localhost:9200/books/_search?q=elasticsearch
```

```
{
  "took" : 2, "timed_out" : false,
  "_shards" : { "total" : 5, "successful" : 5, "failed" : 0 },
  "hits" : {
    "total" : 1, "max_score" : 0.076713204,
    "hits" : [ {
      "_index" : “books", "_type" : “book", "_id" : "1",
      "_score" : 0.076713204, "_source" : {
        "title" : "Elasticsearch - The definitive guide",
        "authors" : [ "Clinton Gormley", "Zachary Tong" ],
        "started" : “2013-02-04", "pages" : 230
      }
    } ]
  }
}
```

Search - Query DSL

```
» curl –XGET ‘localhost:9200/books/book/_search' –d '{
    "query": {
        "filtered" : {
            "query" : {
                "match": {
                    "text" :  {
                        "query" : “To Be Or Not To Be",
                        "cutoff_frequency" : 0.01
                    }
                }
            },
            "filter" : {
                "range": {
                  "price": {
                    "gte": 20.0
                    "lte": 50.0
       ...
    }
}'
```

Scalability

Distributed & scalable

- Replication
  Read scalability
  Removing SPOF

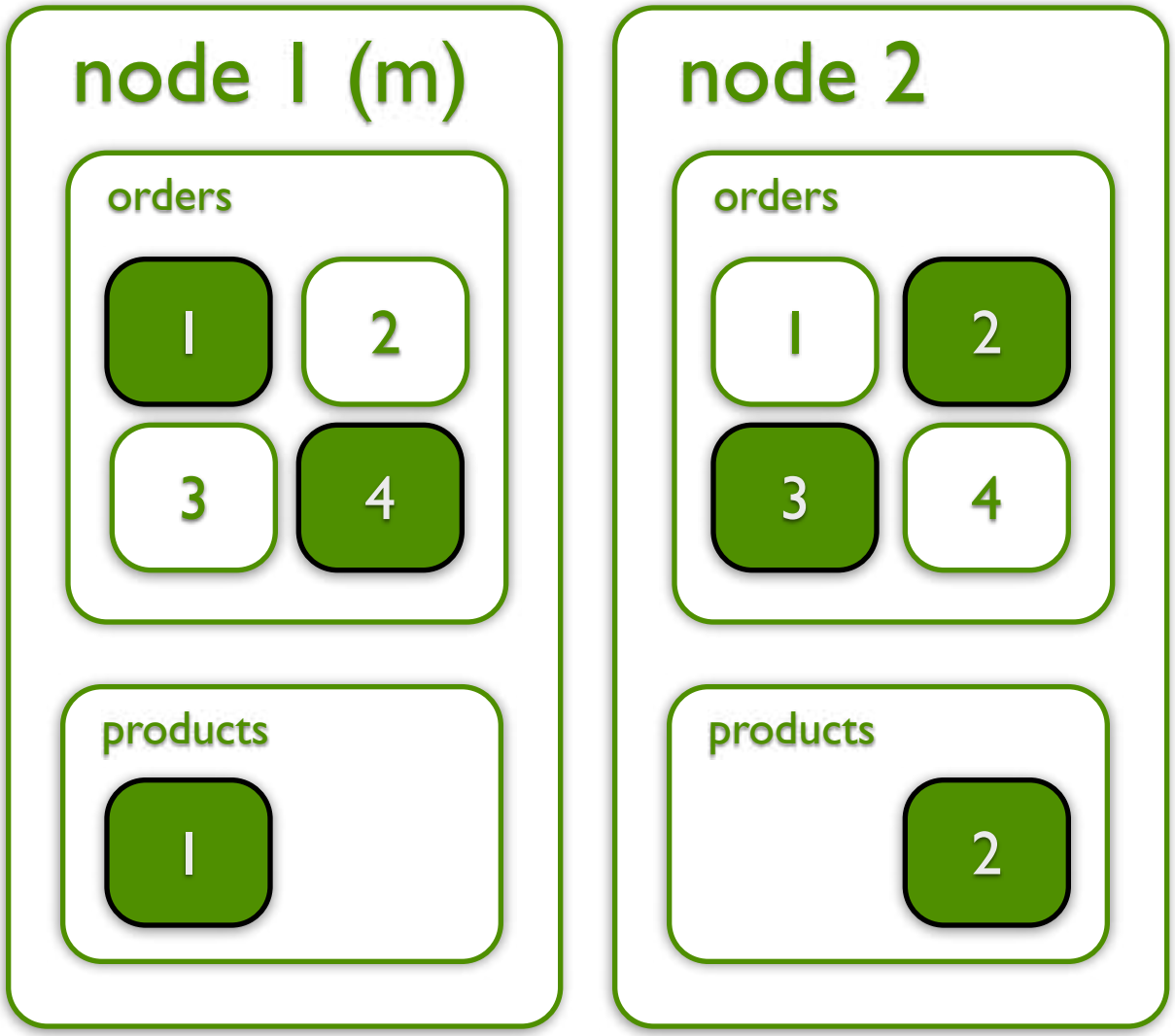
- Sharding
  Split logical data over several machines
  Write scalability
  Control data flows

Distributed & scalable

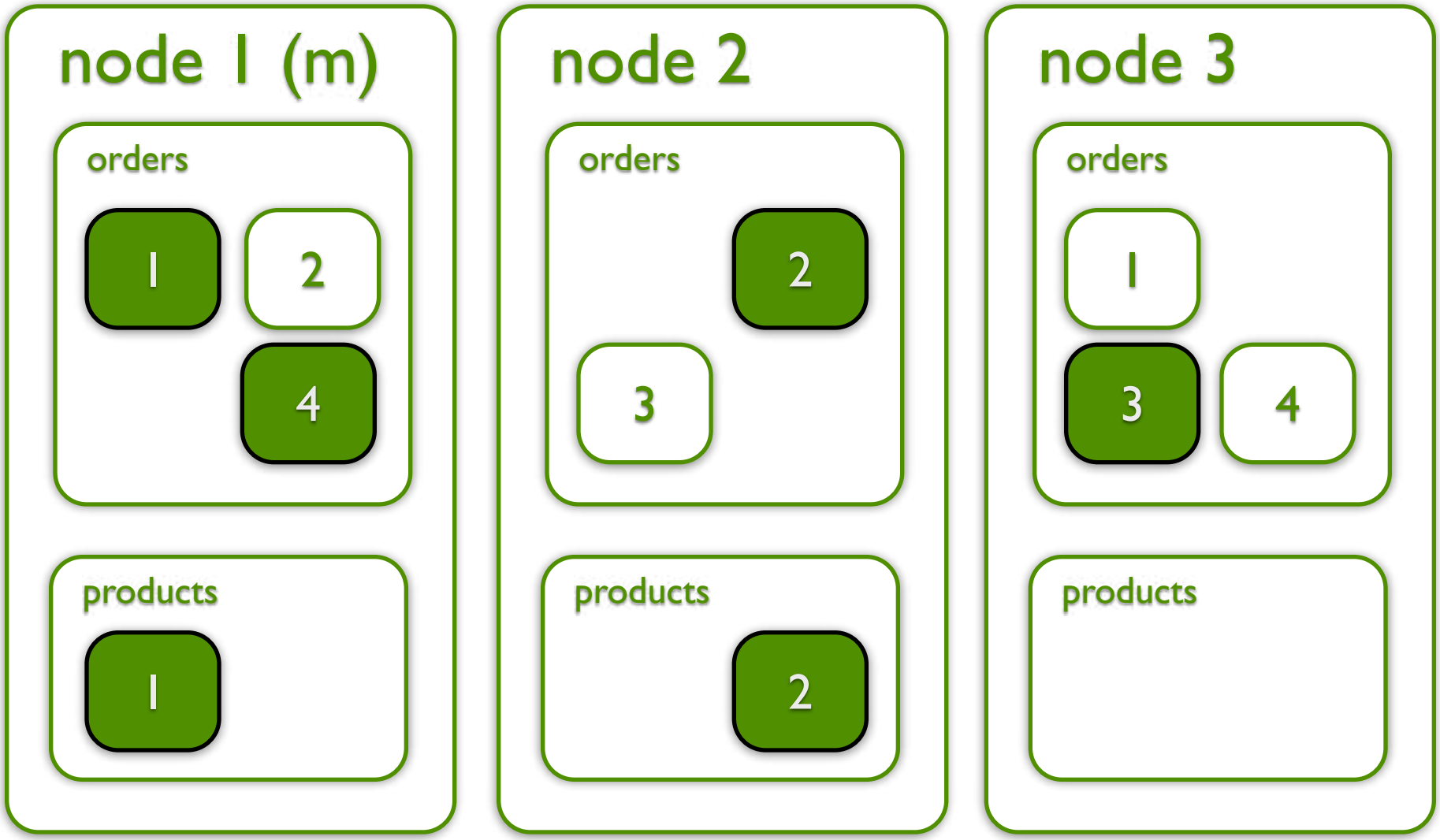

```
curl -X PUT localhost:9200/orders -d '{
  "settings.index.number_of_shards" : 4
  "settings.index.number_of_replicas" : 1
}'
```

```
curl -X PUT localhost:9200/products -d '{
  "settings.index.number_of_shards" : 2
  "settings.index.number_of_replicas" : 0
}'
```

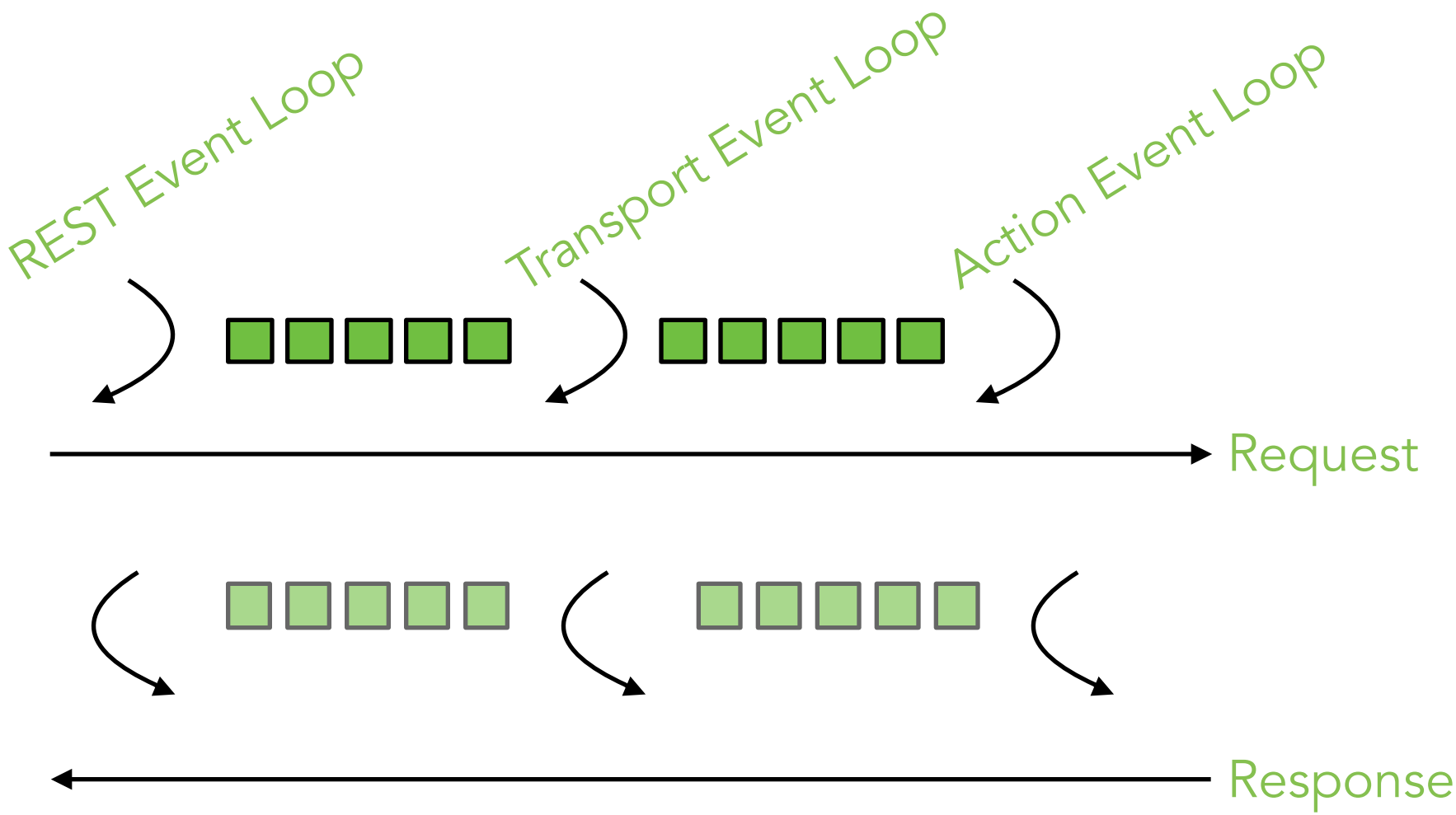
Distributed and scalable



Distributed & scalable



A request under the hood



Think async!

- Enforces event driven architecture
- Support for non-blocking model
- Enforce loose coupling
- Prefers push over pull
- Callback based concurrency
- Helps to avoid contention on resources / threads

Hardware & operating system

Quiz questions!

What is locked memory?

What is the best scheduler for SSDs?

Is TRIM supported on all filesystems?

Ever heard of mechanical sympathy?

Hardware

- Bigger is better? It depends...
- CPU: More cores, more parallel threads
- RAM: No limit
- Disk: SAN vs. local, SSD vs. spindle
- Bare metal vs. virtualization
  https://speakerdeck.com/elasticsearch/life-after-ec2

SSD

- TRIM
- Write amplification
- Garbage collection

- Coding for SSDs
  http://codecapsule.com/2014/02/12/coding-for-ssds-part-1-introduction-and-table-of-contents/

Operating system

- File system descriptors, file system cache
- Memlocked memory (mlockall)
- NUMA
  http://engineering.linkedin.com/performance/optimizing-linux-memory-management-low-latency-high-throughput-databases
  http://queue.acm.org/detail.cfm?id=2513149
- Never swap out if you need performance!
- OOM killer: Just dont...

JVM

Quiz question!

When does the JIT compiler start to optimize?

Are server/client vms different?

How big is the default thread stack size? How many threads fit in your HEAP?

JVM tricks

- Less than 32 GB of heap, allowing to use compressed pointers
- Serialize everything yourself (JVM versions tend to be incompatible)
- use server vm, allocate all memory on startup
- reduce thread stack size

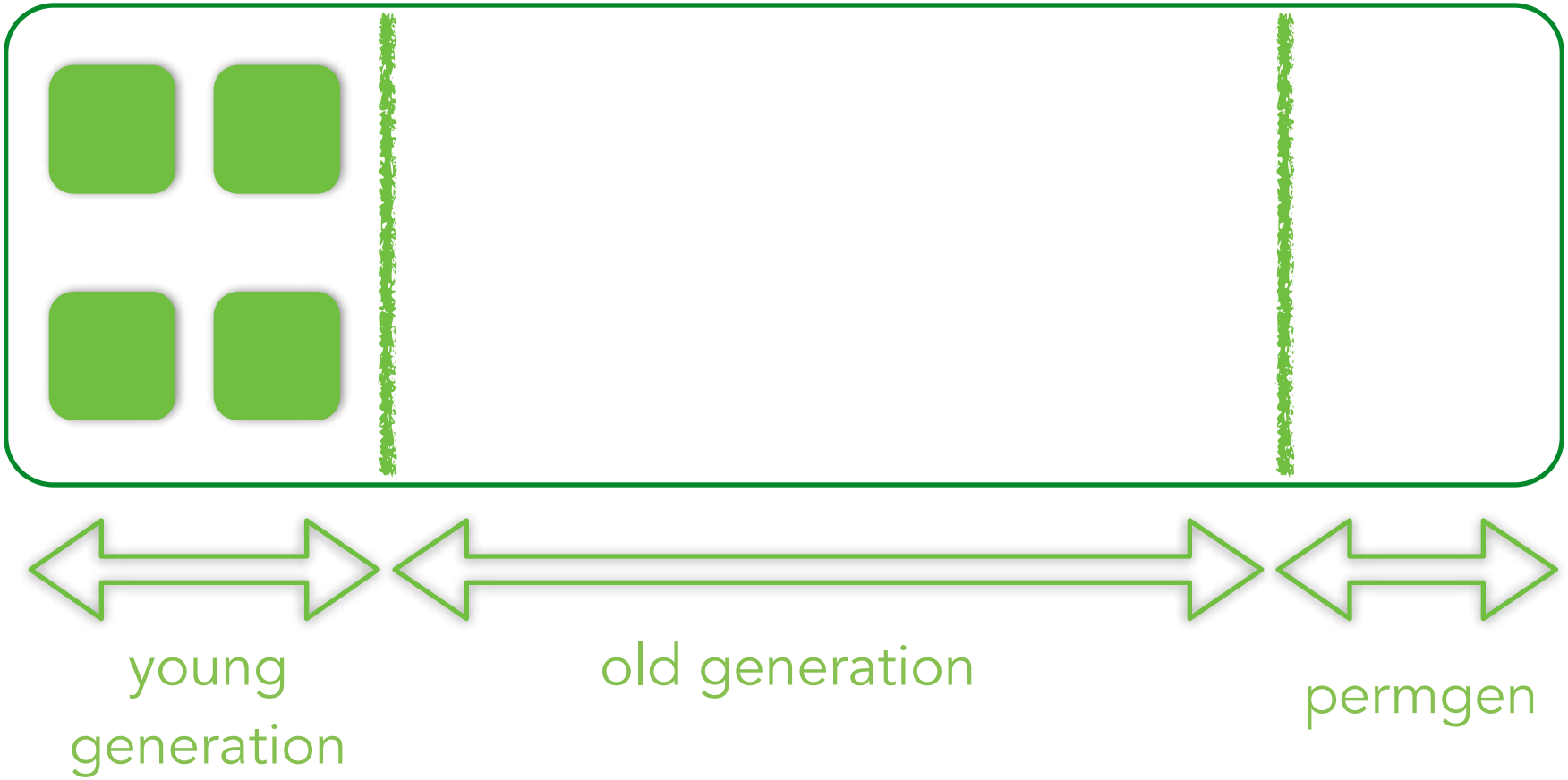
  http://rdiyewar-tech.blogspot.de/2013/02/outofmemoryerror-because-of-default.html

Threads

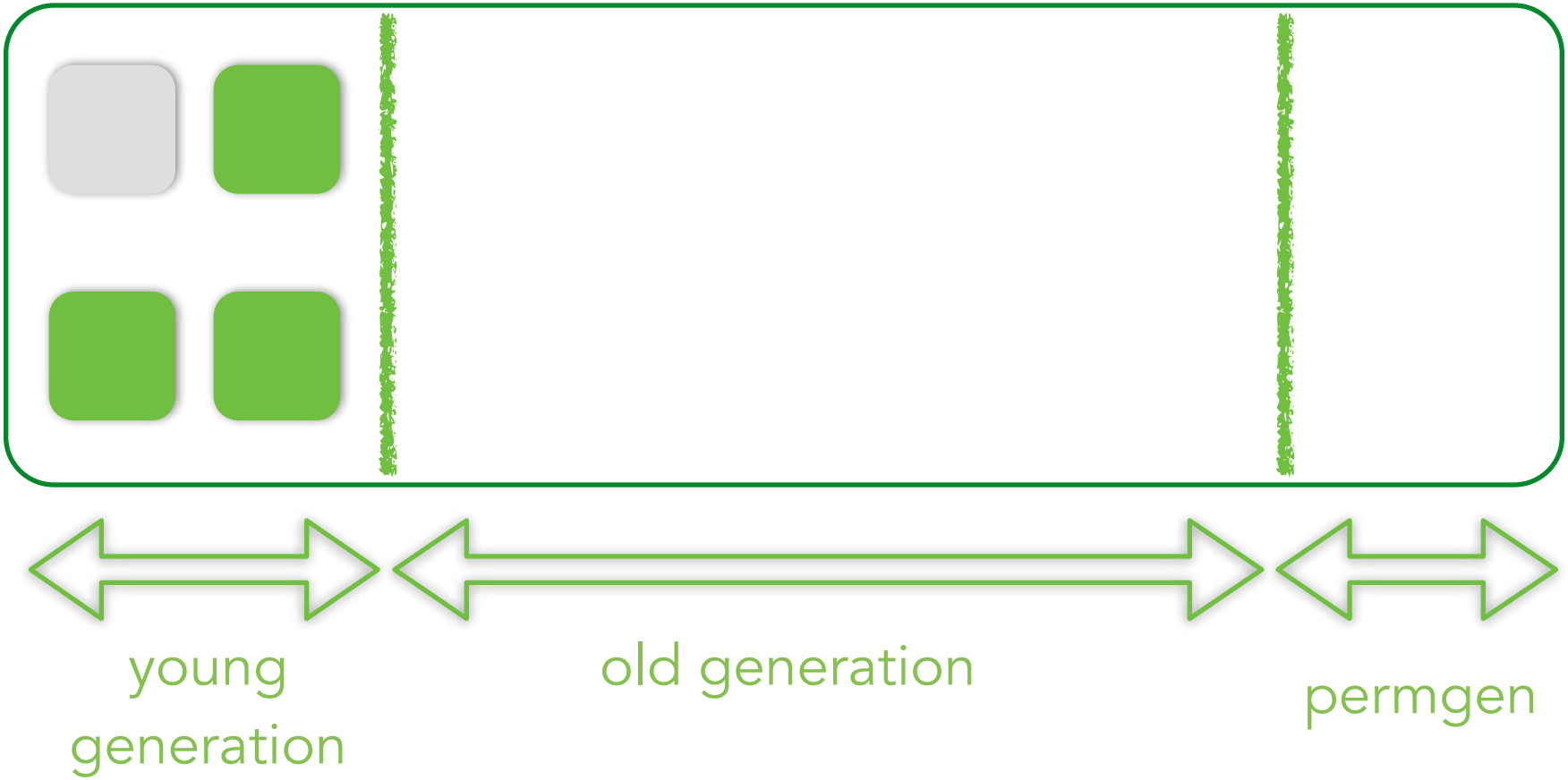
- JVM is good at managing threads, if it is not several thousands of them
- Not every task needs the same resources, one thread pool does not fit all
- Solution: Dedicated thread pools, based on the amount of available CPUs and their task complexity

Garbage collection

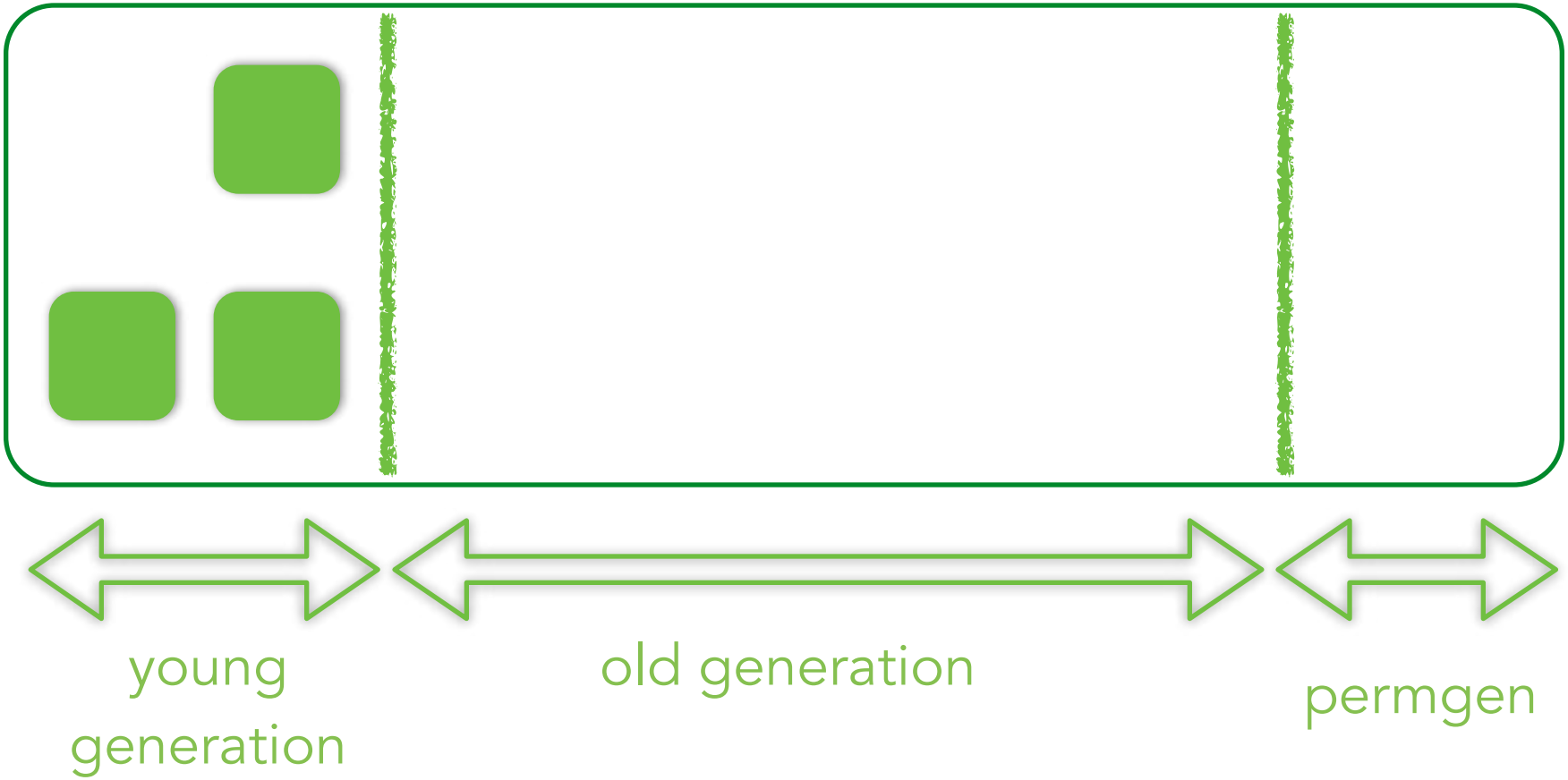
Basics



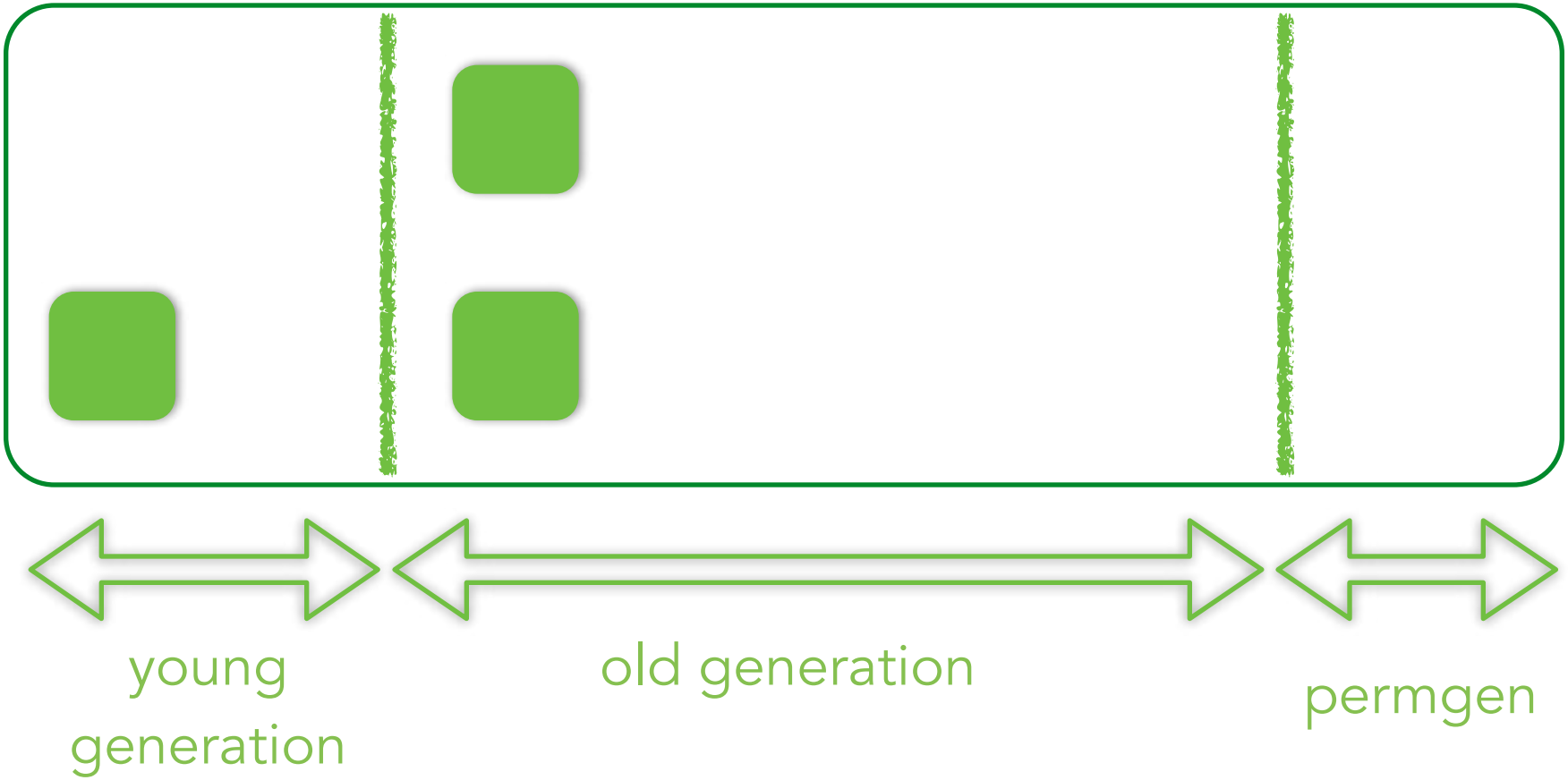
Basics



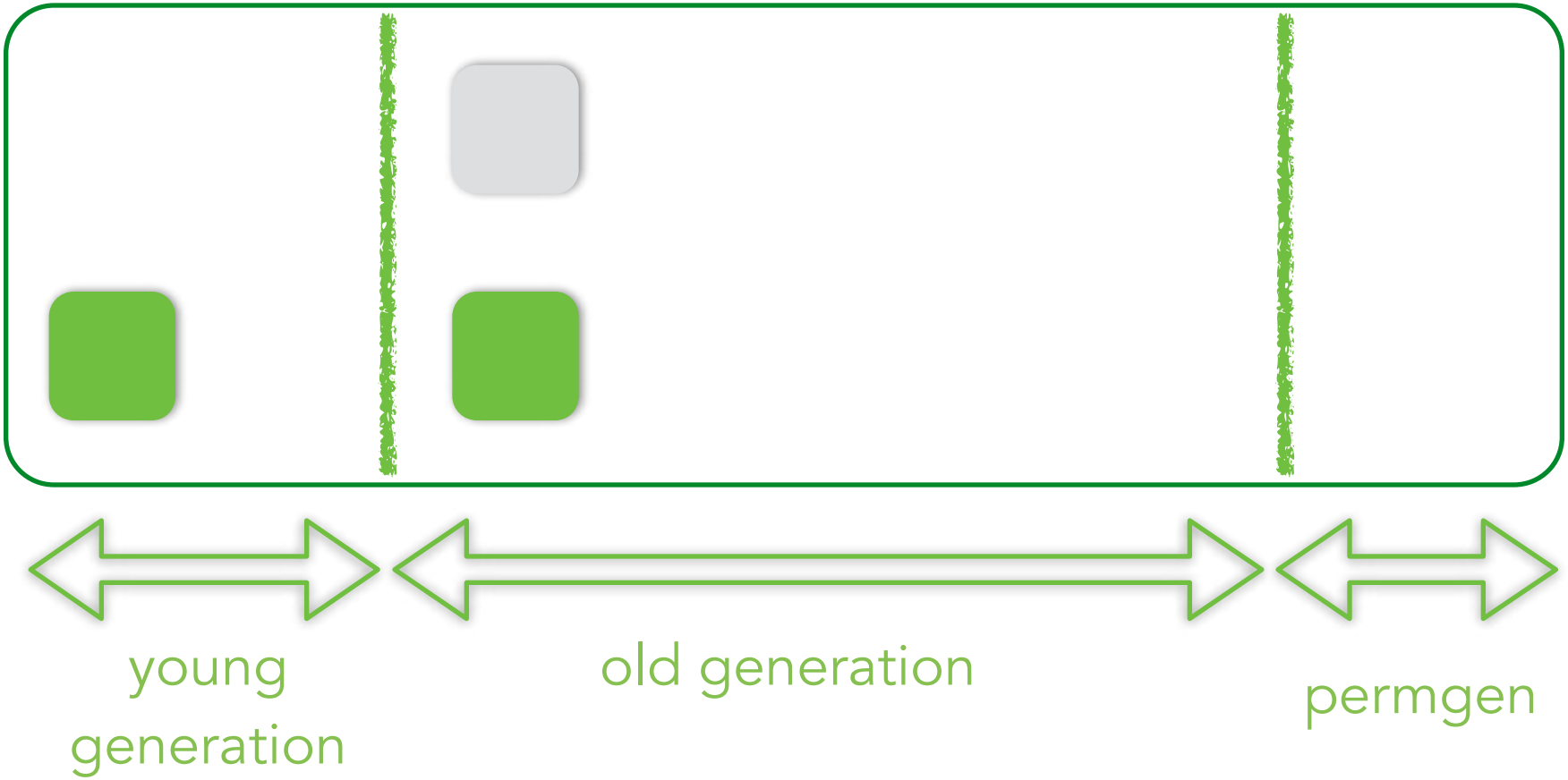
Basics



Promotion to old space



Marking in old space



Sweeping

Avoiding/Improving GC

- Create less objects
- Stream data in to avoid object creation and keeping objects in memory (young gen)
- `-XX:CMSInitiatingOccupancyFraction=75`
- Long GCs can result in nodes dropping out of the cluster and master reelections and data shifting (often happens due to GC pressure)

Garbage collectors

- Serial, Parallel, ParallelOld
- CMS - Concurrent mark-and-sweep
- G1
- Pauseless GC (Shenandoah, Azul)
- Going off-heap
  Using java.misc.Unsafe & handle memory allocation yourself

Libraries

Guice

- dependency injection container
- allows to create infrastructure for plugins
- singletons can be created eager (on startup)

HPPC

- Guava is awesome, except for performance
- Meet High Performance Parallel Collections
  http://labs.carrotsearch.com/hppc.html
- Mapping updates (use of ImmutableOpenMap)
  built on top of ObjectObjectOpenHashMap
  1000 properties: 0.2 seconds (was 5.1)
  2000 properties: 1.2 seconds (was 25)
  5000 properties: 4.2 seconds (was 231)
  10000 properties: 83.8 seconds (never finished before)

Lucene

- Writes are append-only (segments are immutable)
  Allows the file system cache to kick in for huge segments
  Lock-free read access
- Rate limiting on write
  Saves IO and CPU
- Packed* classes, ordinals

Lucene

- Filter caching per segment
- Field data caching per segment
- FSTs

  Blazing fast in-memory structures, allow thousands of qps
  Allow for complex searches like prefix/fuzzy searches or intersections

Sigar

- Great helping library for getting all kinds of stats
- Output can vary on operating systems

Jackson

- Stable and fast streaming JSON parser
- Supports YAML and SMILE
- New and also claims to be lightning fast
  https://github.com/RichardHightower/boon/wiki

Elasticsearch

Reuse objects

- Page-based cache recycling (old gen!)
- Reusing netty buffers
- Fielddata

Node-to-Node communication

- Maintaining different channels with different priorities

  IMMEDIATE, URGENT, HIGH, NORMAL, LOW, LANGUID

- Binary protocol

- TCP connections are held open

Preventing OOM

- Fielddata is number one OOM reason
- Circuit breaker
- Doc-value based field data

Threadpools

- Merging
- Networking
- Indexing
- Searching
- Bulk
- Management
- Snapshot/Restore
- Get
- Refresh
- Warmer
- Optimize
- Percolate
- Suggest
- Flush

Transaction log

- Search is near real-time, background thread makes data available for search every second (by default)
- Creating a new segment after every document is indexed: too expensive
- So, how to do realtime GET, when it is not searchable?
- Solution: write data into additional data structure, that is easy to write to disk, yet very cheap to lookup until data is written into lucene

Keeping GET requests fast

- After a refresh new data is written into a lucene index, thus the transaction log is cleared
- How does a GET request look like now?
- Naive: Searching for a type and an ID in a shard, which in turn consists of segments
- Needing to search each segment does not scale!

Keeping GET requests fast

- Welcome bloom filters!

  Check out impls in Guava, Elasticsearch
  http://www.infoq.com/presentations/scalability-data-mining

- Dependent on hash function and number of functions

- Can tell exactly if an element is NOT in a list

Keeping GET requests fast

- Solution: Maintaining an additional bloom filter data structure per segment
- Implemented as own postings format via Lucene
- Results only in n segment lookups (fast!) instead of need to search each segment
- At the price of higher memory

Percentile Aggregations

- Elasticsearch 1.1.x features a percentile aggregations, allowing to easily find out the distribution of a value in your data
- Great to find outliers

  Think HTTP response times (average and median is not too useful)
- Naive implementation does not scale

Percentile Aggregations

- Solution: Using T-Digests
  https://github.com/tdunning/t-digest/blob/master/docs/t-digest-paper/histo.pdf
- Trading in accuracy for memory savings
- Accuracy is configurable, at the cost of memory and speed
- Default (worst case!): 480kB for a percentile aggregation (per shard, per bucket)

Cardinality Aggregations

- Calculating the amount of distinct values in a field
- Naive approach: Set containing all the values
- Enter HyperLogLog++

HyperLogLog++

- Configurable precision, which decides on how to trade memory for accuracy,
- Excellent accuracy on low-cardinality sets
- Fixed memory usage: no matter if there are tens or billions of unique values, memory usage only depends on the configured precision

HyperLogLog++: Precompute hashes

- Every aggregation run will compute hashes and use those
- You can precompute that on index to lower execution time
- Hashing is fast on numeric fields, rather an edge-case optimisation

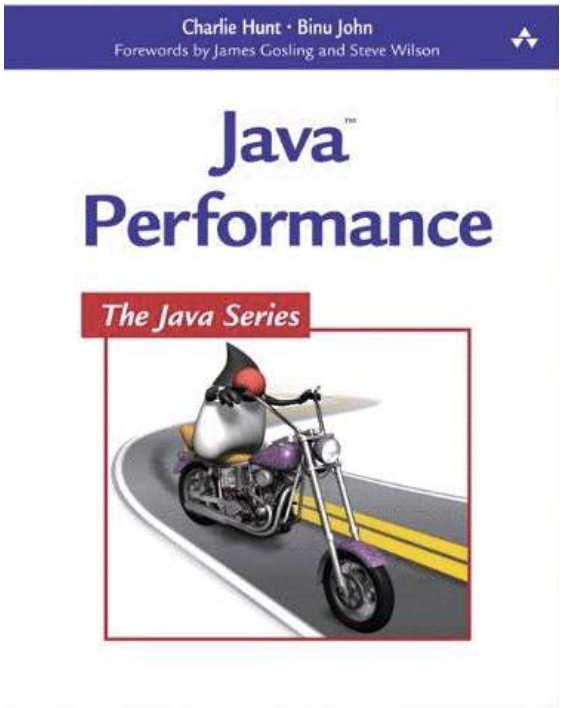
Summary

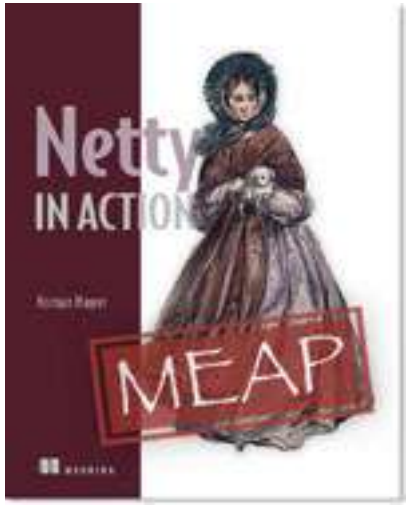
Summary

- Monitor all the things
- Know your full stack, it is invaluable
  Hardware, OS, Environment, Language, Protocols, Libraries
- Do not trust other people's numbers! Fake your own!
- Probalistic data structures are awesome
  ... unless you are a bank/insurance company or need exact numbers

Resources

Learn this stuff!





Resources

http://jprante.github.io/2012/11/28/Elasticsearch-Java-Virtual-Machine-settings-explained.html

https://plumbr.eu/blog/what-garbage-collector-are-you-using
https://plumbr.eu/blog/g1-vs-cms-vs-parallel-gc

http://www.slideshare.net/aragozin/garbage-collection-in-jvm
https://github.com/aragozin/jvm-tools

https://github.com/brettwooldridge/HikariCP/wiki/Down-the-Rabbit-Hole

Resources

http://www.artima.com/underthehood/flowP.html

http://en.wikipedia.org/wiki/Switch_case#Compilation

http://www.elasticsearch.org/blog/disk-based-field-data-a-k-a-doc-values/

http://static.googleusercontent.com/media/research.google.com/fr//pubs/archive/40671.pdf

Thanks for listening

Q & A

P.S. We're hiring
http://elasticsearch.com/about/jobs
http://elasticsearch.com/support

Alexander Reelsen
@spinscale
alexander.reelsen@elasticsearch.com

elasticsearch.