Agile
Containers
DevOps Automation
DevOps Linux
DevOps Services
Languages
Network
Scripting
Security
Solutions
Virtualization
Windows
Glib Examples
Go Examples
Javascript Examples
PHP Examples
Python Examples
Regex Examples

Apache
Ceph
Consul
CouchDB
Couchbase
ElasticSearch
Eureka
F5
GlusterFS
Graphite
Hadoop
Java
Kafka
MongoDB
MySQL
NFS
Nodejs
PgBouncer
Postgres
RabbitMQ
Redis Sentinel
Redis
Samba
Squid
Tomcat
Webserver
Zookeeper
etcd
htaccess Checker
htaccess
memcached
nginx
sqlite
zuul

ElasticSearch Cheat Sheet

Edit Cheat Sheet

REST API

Generic endpoint is on port 9200

Cluster Status

```
/_cat/health?v /_cat/nodes?v /_cat/indices?v /_cluster/health /_cluster/state # gives elected master and
shard initialisation status /_cluster/stats /_cluster/settings
```

Further introspection:

```
/_nodes/ /_nodes/process /_nodes/settings /_nodes/stats /_aliases /_warmers /_mappings
```

Indexes

```
GET /_cat/indices?v GET /<index name>?pretty PUT /<index name> DELETE /<index name> GET /_settings # Print
config for all indices
```

Copying indices using "reindex": It is possible to copy indices partially/fully from local as well as from remote indices:

```
POST /_reindex { "source": { "remote": { "host": "http://otherhost:9200", "username": "user", "password":
"pass" }, "index": "source", "query": { "match": { "test": "data" } } }, "dest": { "index": "dest" } }
```

Index Aliases

Endpoints for index aliases are quite messy

```
GET /_aliases?pretty POST /_aliases { "actions" : [ { "add" : { "index" : "<index>-000001", "alias" : "my-
<index>-alias" } } ] } DELETE /{index}/_alias/{name}
```

Trigger index rollover

```
POST /<alias>/_rollover { "conditions": [ "max_age": "3d", "max_docs": 1000000, "max_size": "30g" ] }
```

Shard Allocation

List unassigned shards

```
curl -s "<server>:9200/_cat/shards?v" | grep -E "UNASSIGNED|prirep"
```

Get info when shards are not allocated

```
GET /_cluster/allocation/explain
```

Retry allocation of shards (after retry limit reached)

```
GET /_cluster/reroute?retry_failed=true
```

Documents

```
GET /<index name>/external/1?pretty # Insert/Replace PUT /<index name>/external/1 { 'key': 'value' } #
Update POST /<index name>/external/1 { "doc": { 'count': 5 } } POST /<index name>/external/1 { "script":
"ctxt._source.count += 1" } DELETE /<index name>/external/1 DELETE /<index name>/external/_query { "query":
{ "match": { 'key': 'value' } }
```

Batch processing

```
POST /<index name>/external/_bulk {"index":{"_id":"1"}} {"key1": "value1"} {"index":{"_id":"2"}} {"key2":
"value2"} {"update":{"_id":"3"}} {"doc": { "key3": "value3" } {"delete":{"_id":"4"}} [...]
```

Queries

Just a simple search example to explain query building

```
GET /<index name>/external/_search?q=* POST /<index name>/external/_search { "query": { "match": {
"field1": "abcdef" } }, "sort": { "balance": { "order": "desc" } }, "from": 10, "size": 10, "_source":
["field1", "field2"] }
```

Management Tools

- Index retention: Curator
- Webadmin: Cerebro
- Auth: XPack Security (previously "Shield"), SearchGuard
- Alerting: Elastalert, Logagent, Sentinl
- Monitoring:
- by Elastic: Marvel, XPack

ELK Scaling Cheat Sheet

Sizing Examples

- Viki 2015
  - Ingest: 25k/s Access Logs
  - haproxy as Logstash LB
  - Logstash single-threaded filters, 4 Nodes (8 CPU, 16GB)
  - Logstash Forwarder Client with buffer log
  - Elasticsearch:
    - 20 Nodes (12 i7-3930k, 64GB, 3TB RAID0)
    - 20 shards, 4 replicas
    - 30GB heap
- Meltwater 2018
  - Search Volume: 3k/min complex search requests
  - Index Size: 3*10^6 articles, 100*10^6 social posts, 200TB
  - Elastischsearch:
    - 430 data nodes: i3.2xlarge, 64GB RAM
    - 3 master nodes
    - 40k shards, 100MB cluster state!
    - 26GB heap
- Etsy 2016
  - Index Size: overall 1.5PB
  - Ingest: 10^9 loglines/day, 400k/s peak
  - Elasticsearch:
    - 6 clusters, 141 Nodes (overall 4200 CPU Cores, 36TB)

Posts on Scaling:

- codecentric.de Tuning Hints
- hipages Engineering - Scaling ES
  - Scaling on index size (metrics: documents per shard, documents per node)
    - Change shards to trade search response time for search concurrency
    - Change nodes to trade resilience for memory usage
  - Scaling on search time and througput
    - Scalability Model
- Evolution of an ELK Setup
  1. ELK with 1 Logstash
  2. ELK with loadbalanced horizontally scaled Logstash
  3. Kafka in front of logstash to buffer spikes ELK
  4. Separation of client, data and master Elasticsearch nodes
- Determining the Number of Shards

General hints:

Note: credits for all those go to the post above. Consider this a compilation for ES begiinners.

- Set CPU scaling governor 'performance'
- Use SSDs with RAID 0
- Use HTTP transport protocol
- Change default mapping
  - Avoid raw fields
  - or make raw field "not_analyzed"
- Disable transparent huge pages
- Disable numad
- Disable swap, lock memory with bootstrap.mlockall: true
- Do not optimize JVM settings for max memory usage!
  - Try to live with 4GB heap
  - Ensure not to give more than 30GB RAM (sometimes only as much as 26GB) as JVM heap address compression stops with larger RAM

    - Check heap address mode by running with -XX:+UnlockDiagnosticVMOptions -XX:+PrintCompressedOopsMode and if you see "zero based Compressed Oops" you are fine
    - Check your heap usage curve. If you have a sawtooth give back the memory to the FS cache.
- When profiling
    - check for >15% ParNewGC
    - check SerialGC pauses
        - ensure you do not have the G1 garbage collector active
- Logstash:
    - On HW consider turning off hyperthreading
    - Increase flush_size
    - Increase idle*flush*time
    - Increase output workers
    - Finally increase pipeline batch size

Resilience

- Avoid split-brain by setting discovery.zen.minimum*master*nodes
- Monitor fielddata cache to avoid running in OOM killing your cluster

Monitoring

- Logstash Pipeline Monitoring using XPack + Kibana

Comment on Disqus