Elastic Search
&
In-App Messaging

HTTP://BAYESANALTYIC.COM

JOE ELLSWORTH CTO, ALGORITHMS SCIENTIST & CONSULTANT

JOEE@BAYESANALTYIC.COM 206-601-2985

Bayes Analytic LLC

High performance distributed architecture.

- Greater than 300 queries per second

High availability Web

Complex Search

Geo-Location Search & Matching

Data Discovery

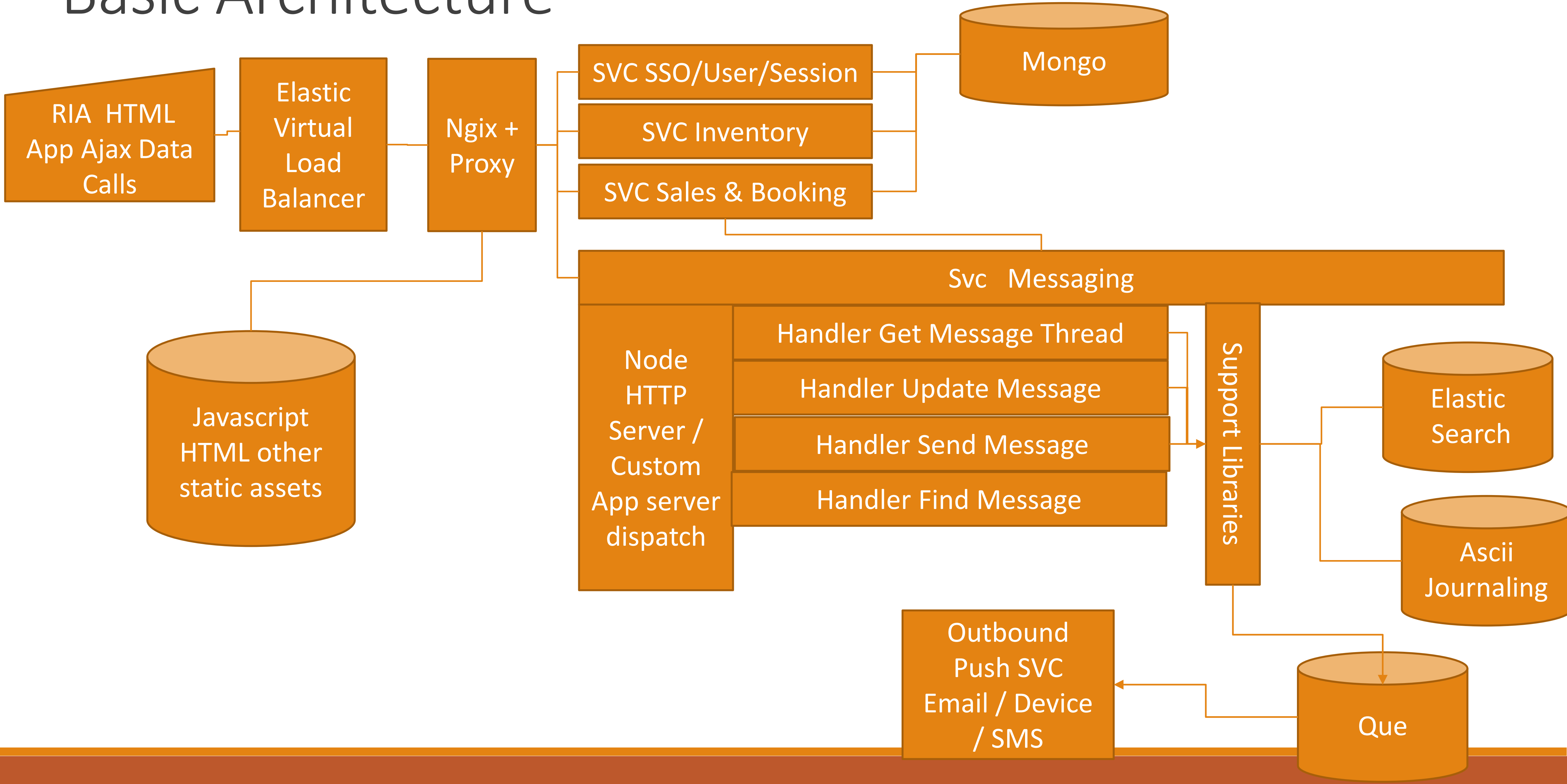
Predictive Engines for Search & Offers

Products

- Internal product – Stock and Forex Price prediction
- Open Source high performance Node libraries
- Open source Metadata server in pure Node
- Open source CNC Scripting
- Open source F# based Messaging – optimized form multi-reader very high performance and low latency.

Clients: Mostly medium sized companies with over 300M in revenue who are having performance, reliability or development velocity issues.   A small number of startups like Peerspace.

Basic Architecture


RIA HTML App Ajax Data Calls
Elastic Virtual Load Balancer
Ngix + Proxy
SVC SSO/User/Session
SVC Inventory
SVC Sales & Booking
Mongo
Javascript HTML other static assets
Svc Messaging
Node HTTP Server / Custom App server dispatch
Handler Get Message Thread
Handler Update Message
Handler Send Message
Handler Find Message
Support Libraries
Elastic Search
Ascii Journaling
Outbound Push SVC Email / Device / SMS
Que

More Typical Architecture

Master data source

Extractor / Event Bus

Delivery Queue

Multiple Fully Isolated Elastic search clusters

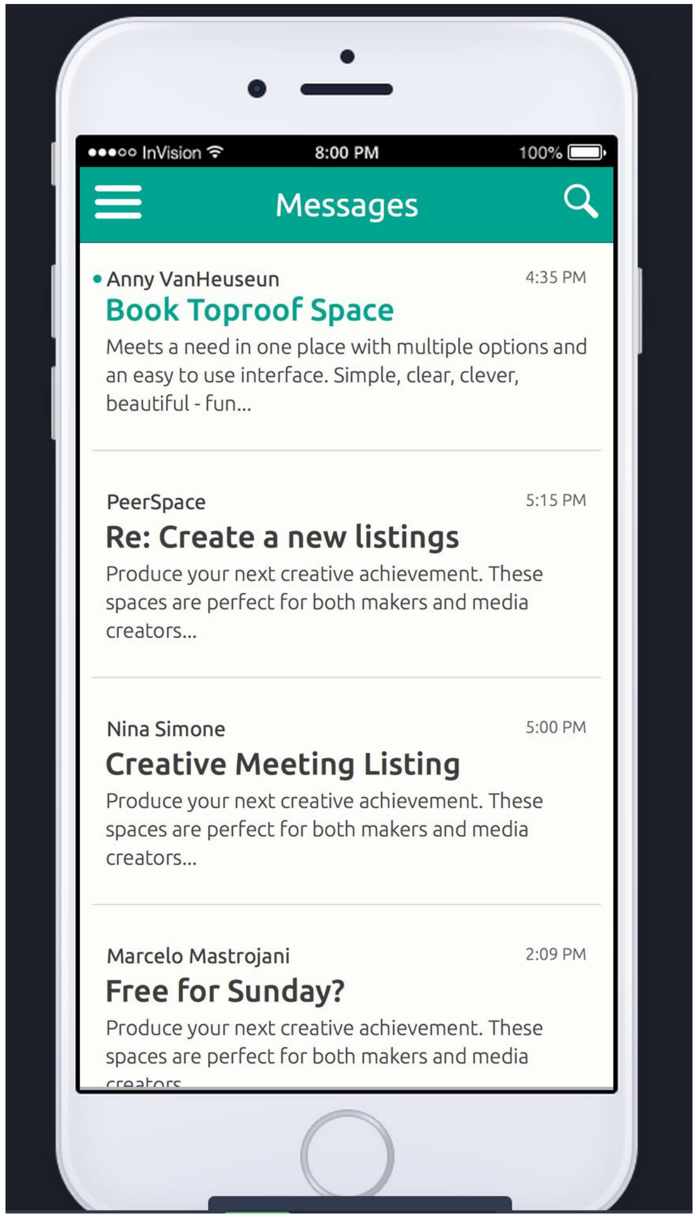
Elastic search is considered read only for the web app.

Normally has ASP tier -> Search Service -> Elastic Search

Sample Message Screen

Make Under utilized spaces available for other users

PeerSpace believes that changing your work surroundings via short-term use of inspiring spaces regularly boosts innovation for all types of professionals. To build a thriving marketplace, PeerSpace has partnered with businesses and other space owners to help them make more efficient use of their space, allowing them to safely and easily share it with other professionals online.



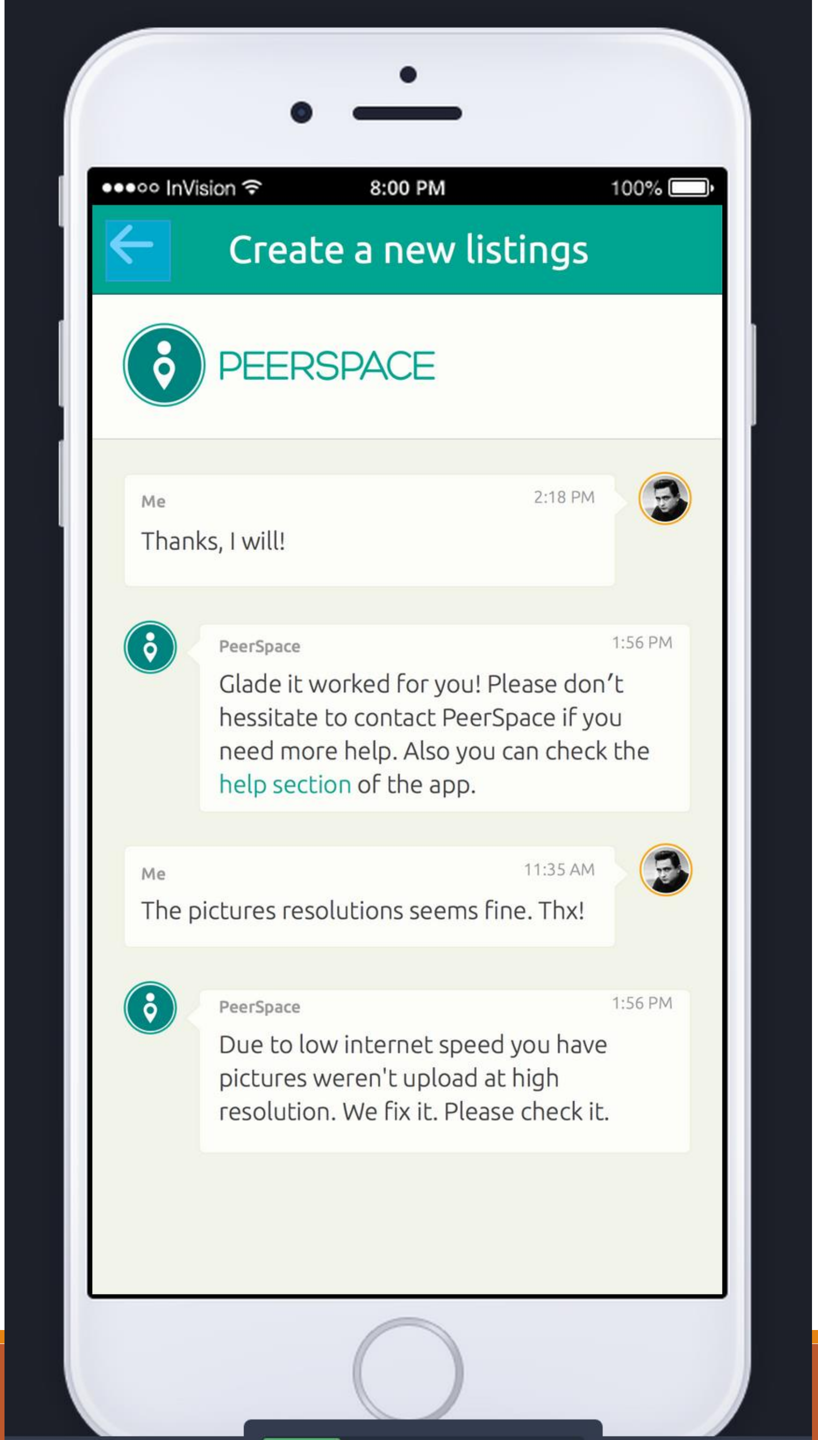
• Anny VanHeuseun 4:35 PM
Book Toproof Space
Meets a need in one place with multiple options and an easy to use interface. Simple, clear, clever, beautiful - fun...

PeerSpace 5:15 PM
Re: Create a new listings
Produce your next creative achievement. These spaces are perfect for both makers and media creators...

Nina Simone 5:00 PM
Creative Meeting Listing
Produce your next creative achievement. These spaces are perfect for both makers and media creators...

Marcelo Mastrojani 2:09 PM
Free for Sunday?
Produce your next creative achievement. These spaces are perfect for both makers and media creators

Sample Conversation Thread


●●●○○ InVision 8:00 PM 100%

Create a new listings

PEERSPACE

Me 2:18 PM

Thanks, I will!

PeerSpace 1:56 PM

Glade it worked for you! Please don't hessitate to contact PeerSpace if you need more help. Also you can check the help section of the app.

Me 11:35 AM

The pictures resolutions seems fine. Thx!

PeerSpace 1:56 PM

Due to low internet speed you have pictures weren't upload at high resolution. We fix it. Please check it.

Why Middle service

Application specific simplified queries.

Application specific responses

Isolate complexity especially when single query requires multiple chained searches.

Limit connection demands against elastic search.

Keep most of the API technology Agnostic so we can swap engines.

Maintain flexibility to allow manual shading to spread load across Elastic Search.

Simple Thread Query

http://127.0.0.1:9602/api/messages/threadtofrom?to=64b69968c69e63060063b664&from=help&v=elast_query,count

What it looks like as Elastic Search Query

POST: http://127.0.0.1:9599/_search

```
{"size":999,
 "sort":[{"created":{"order":"desc"}},
{"from.ssoid":{"order":"desc"}},
{"space.id":{"order":"desc"}},
{"listing_id":{"order":"desc"}},
{"booking_id":{"order":"desc"}},
{"created":{"order":"desc"}}],
"query":{"filtered":{"filter":{"bool":{"should":[{"bool":{"must":[{"term":{"from.ssoid":"help"}},{"t
erm":{"to.ssoid":"63b68068c69e560860a6b664"}}]}},{"bool":{"must":[{"term":{"from.ssoid":“64
b69067a69e620601a6b664"}},{"term":{"to.ssoid":"help"}}]}}]}}}}}
```

Elastic Search Queries in Node.js

```
  var aggs = {
      "from": {
        "terms": {
          "field": "from.ssoid"
        }
      },
      "to": {
        "terms": { "field": "to.ssoid" }
      },
      "is_read"  : {
        "terms" : { "field" : "is_read" },
        "aggs"  : { "mtype" :  { "terms" : { "field" : "mtype"}
},
                  "prefs" :  { "terms" : { "field" : "prefs"} }
                }
      }
    }
```

```
var tquery = {
   "size": tin.size,
   "sort": [
       { "created": { "order": "desc" } },
       { "from.ssoid": { "order": "desc" } },
       { "space.id": { "order": "desc" } },
       { "listing_id": { "order": "desc" } },
       { "booking_id": { "order": "desc" } },
       { "created": { "order": "desc" } },
       //{ "created": { "order": "desc" } }
   ],
   "query": {
     "filtered": {
       "filter": {
         "bool": {
           // value of filter added below
         }   }     }   } }
```

Supporting Views

http://127.0.0.1:9602/api/messages/threadtofrom?to=542e92d7259e24025027&from=help

http://127.0.0.1:9602/api/messages/threadtofrom?to=542e92d7259e24025027&from=help&v=elast_query

http://127.0.0.1:9602/api/messages/threadtofrom?to=542e92d7259e24025027&from=help&v=count

http://127.0.0.1:9602/api/messages/threadtofrom?to=542e92d7259e24025027&from=help&v=count,elast_query,msg_detail

Elastic Search Good Bad Ugly

Good:

Rapid development

Extremely flexible with emerging requirements to add fields.

Nearly transparent syntax between Node and Elalstic

Aggs module is extremely helpful

Search functionality supported esoteric requirements more expensive to support in traditional DB.

Ugly:

Simple architecture may have write scaling issues.

Query after write on refresh forces cache invalidation that could reduce max TPS.

Lack ofAlter table requires dump, transform, reload script when it occurs but luckily seldom encountered.

Under heavy load Elastic search will panic with horrible 7,000ms or worse search responses.

Query language is simply more cumbersome and way more verbose than equivalent Lucene / Solr Syntax.

Recovery Journaling

Never, Ever trust any binary Storage system.   Must have ASCII Based human readable journal to rebuild.

- Over the last 20 years I have billed over ½ million $ for emergency recovery of supposedly high stability systems such as Oracle, Progress,   MS SQL,   Berkley, Lucene. When they fail it is often with silent creeping corruption that quite often crosses many backups.
- ASCII journals have saved me more than once when forced to abandon a promising product that wasn't quite ready for primetime. And would have saved my clients. Due to the fact that we have the ASCII journal a elastic search corruption event is not a huge risk. They are cheap and massively scalable.

Saves every record JSON source saved in elastic search.

Must be technology agnostic

Simple JSON files limited to about 10 Meg per file.

Auto join smaller files to 2 Gig limit for backup

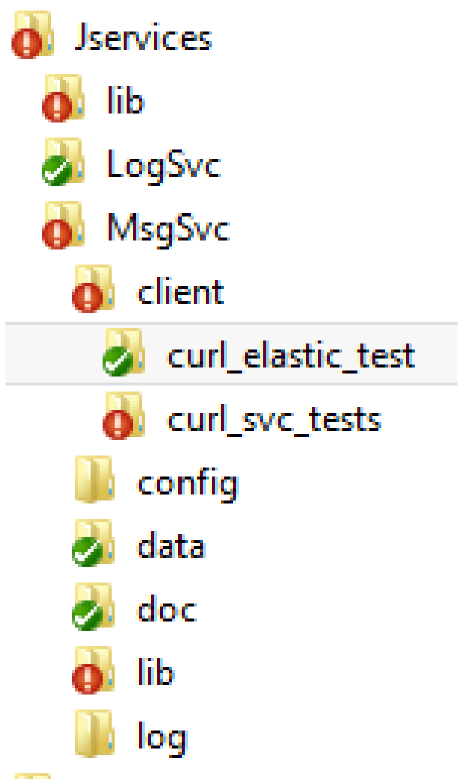
Hook to auto replication system for off box replication. Can also use queue like Kafka.

Can index filename and starting offset to easily retrieve any version of any object by ID

Periodic de-dupe can be used to reduce space if versioning is not needed.

Just used this to reload an elastic search instance when needed to change underlying object type.

Example Curl Svc Calls – essential docs



Jservices
lib
LogSvc
MsgSvc
client
curl_elastic_test
curl_svc_tests
config
data
doc
lib
log

```
curl -X GET
"http://127.0.0.1:9602/api/messages/threadtofrom?to=2363163d26f83633a63f637&from=f8b29620026c21e62e1e664a672&v=elast_query"
```

Running many stand along elastic search instances.

```
..\..\elasticsearch\bin\elasticsearch.bat -DES_HEAP_SIZE=1G -
Des.path.data=../../elastic9599/data -Des.path.logs=../../elastic9599/logs -
Djava.io.tmpdir=../../elastic9599/tmp -Des.path.cluster=a9599 -
Des.mlockall=true -Des.vm.swappiness=1  -Des.http.port=9599 -
Des.transport.tcp.port=9598 –Des.cluster.name=elastic9599
```

Allows many different elastic search instances that are all completely isolated for different clients on the save server.

Caution: Keep your Data and log directories as virtual mounts for the Docker image or it will cause no end grief.

Also used to Bypass JVM GC issues on Big Boxes – Has allowed more than Double Net TPS with multiple search listeners

Minimal Mapping Strategy

```
{
  "settings" : { "number_of_shards" : 8 },
  "mappings" : {
    "msg" : {
      "_source" : { "enabled" : true},
      "properties" : {
        "id" : { "type" : "string", "analyzer":"keyword"},
        "space" : {
          "properties" : {
            "id" : { "type" : "string", "analyzer":"keyword"},
            "loc" :  {"type":"geo_point", "lat_lon": true, "validate":true,
                      "normalize":false, "geohash":true, "geohash_prefix": true}
      } } } } } }
```

Core Principal: Allow new Fields to be added by client without changing our Service

Our Core Libraries (MIT Open Source)

Robust HTTP to Elastic Search.

Connection limit management

Automatic basic parsing for Rest/JSON services

Advanced control of handlers

MDS – Metadata Service All Native Node

Designed to support high speed metadata fragments not returned by the Search engine.

Forward Cached repository nearly as fast as Memcache but massively scalable to many T.

Out performs any other Metadata server we have tested.

- At about 30K requests per second with average doc size of 8K and 90 million docs.
- On AWS instance with 8 cores and 128GB of RAM.

Runs up to saturation on a 1 Gig network when used with 8 Cores.

Maximal leverage of Linux file system cache.