Building a Powerful Data Tier
From open source datastores

Joey Lynch
jlynch@yelp.com
Data Shepherd
Yelp’s Mission

Connecting people with great local businesses.
Yelp Stats
As of Q2 2016

92M

108M

72%

32
Monitoring Cassandra at Scale

As of Q2 2016

Yelp 4+ years

True Zero Downtime HAProxy Reloads

HAProxy: Cornerstone of Reliable Websites

One primary goal of the infrastructure teams here at Yelp is to get as close to zero downtime as possible. This means that when users make requests for www.yelp.com we want to ensure that they get a response, and that they get a response as fast as possible. One way we do that at Yelp is by using HAProxy.

Scaling Elasticsearch to Hundreds of Developers

Yelp uses Elasticsearch to rapidly prototype and launch new search applications, and moving quickly at our scale raises challenges. In particular, we often encounter difficulty making changes to query logic without impacting users, as well as finding client library bugs, problems with multi-tenancy, and general reliability issues. As the number of engineers at Yelp writing ElasticSearch queries grew, our Search Infrastructure team was having difficulty supporting the multitude of ways engineers were finding to send queries to our ElasticSearch clusters. The infrastructure we designed for a single team to communicate with a single cluster did not scale to tens of teams and tens of clusters.
● Why datastores?
● Why datastores?
● The open source datastore landscape
Why datastores?  
The open source datastore landscape  
Building a datastore platform
● Why datastores?
● The open source datastore landscape
● Building a datastore platform
● Realities of distributed datastores
Why datastores?
The open source datastore landscape
Building a datastore platform
Realities of distributed datastores
Why?

Presentation tier
The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.

Logic tier
This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

Data tier
Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.

- This architecture is old as dirt

... nobody just does this anymore
Presentation tier
The top-most level of the application is the user interface. Its main function of the interface is to convert data and results to something that we understand.

Logic tier
This layer coordinates the application, processes commands, makes logical data evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

Data tier
Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.
Presentation tier
The top-most level of the application is the user interface. Its main function of the interface is to collect and results to something that the user can understand.

Logic tier
This layer coordinates the application, processes commands, makes logical decisions, performs evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

Data tier
Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing and then eventually sent to the user.
Scaling Databases: Choices

Smart Clients

“Primary” + failover + sharding

OR

“Replicas”

“NoSQL”

“NewSQL”
• Why datastores?
• The open source datastore landscape
• Building a datastore platform
• Realities of distributed (computing) datastores
# Landscape: Many Options

![DB Engine Ranking Table](http://db-engines.com/en/ranking)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>1.</td>
<td>1.</td>
<td>1.</td>
<td>Oracle</td>
<td>Relational DBMS</td>
<td>1425.56</td>
<td>-2.16</td>
<td>-37.81</td>
</tr>
<tr>
<td>2.</td>
<td>2.</td>
<td>2.</td>
<td>2.</td>
<td><strong>MySQL</strong></td>
<td>Relational DBMS</td>
<td>1354.03</td>
<td>-3.01</td>
<td>+76.28</td>
</tr>
<tr>
<td>3.</td>
<td>3.</td>
<td>3.</td>
<td>3.</td>
<td>Microsoft SQL Server</td>
<td>Relational DBMS</td>
<td>1211.55</td>
<td>+6.51</td>
<td>+113.72</td>
</tr>
<tr>
<td>4.</td>
<td>5.</td>
<td>5.</td>
<td>5.</td>
<td>PostgreSQL</td>
<td>Relational DBMS</td>
<td>316.35</td>
<td>+1.10</td>
<td>+30.18</td>
</tr>
<tr>
<td>5.</td>
<td>4.</td>
<td>4.</td>
<td>4.</td>
<td>MongoDB</td>
<td>Document store</td>
<td>316.00</td>
<td>-2.49</td>
<td>+15.43</td>
</tr>
<tr>
<td>6.</td>
<td>6.</td>
<td>6.</td>
<td>6.</td>
<td>DB2</td>
<td>Relational DBMS</td>
<td>181.19</td>
<td>-4.70</td>
<td>-27.95</td>
</tr>
<tr>
<td>7.</td>
<td>8.</td>
<td>8.</td>
<td>8.</td>
<td>Cassandra</td>
<td>Wide column store</td>
<td>130.49</td>
<td>+0.26</td>
<td>+2.89</td>
</tr>
<tr>
<td>8.</td>
<td>7.</td>
<td>7.</td>
<td>7.</td>
<td>Microsoft Access</td>
<td>Relational DBMS</td>
<td>123.31</td>
<td>-0.74</td>
<td>-22.68</td>
</tr>
<tr>
<td>9.</td>
<td>9.</td>
<td>9.</td>
<td>9.</td>
<td>SQLite</td>
<td>Relational DBMS</td>
<td>108.62</td>
<td>-1.24</td>
<td>+0.97</td>
</tr>
<tr>
<td>10.</td>
<td>10.</td>
<td>10.</td>
<td>10.</td>
<td>Redis</td>
<td>Key-value store</td>
<td>107.79</td>
<td>+0.47</td>
<td>+7.14</td>
</tr>
<tr>
<td>12.</td>
<td>13.</td>
<td>13.</td>
<td></td>
<td>Teradata</td>
<td>Relational DBMS</td>
<td>73.06</td>
<td>-0.57</td>
<td>-1.20</td>
</tr>
<tr>
<td>13.</td>
<td>11.</td>
<td>11.</td>
<td>11.</td>
<td>SAP Adaptive Server</td>
<td>Relational DBMS</td>
<td>69.16</td>
<td>-1.88</td>
<td>-17.36</td>
</tr>
<tr>
<td>15.</td>
<td>15.</td>
<td>15.</td>
<td>15.</td>
<td>HBase</td>
<td>Wide column store</td>
<td>57.81</td>
<td>+2.30</td>
<td>-1.22</td>
</tr>
<tr>
<td>16.</td>
<td>17.</td>
<td>17.</td>
<td>17.</td>
<td>FileMaker</td>
<td>Relational DBMS</td>
<td>55.35</td>
<td>+0.34</td>
<td>+4.35</td>
</tr>
<tr>
<td>17.</td>
<td>18.</td>
<td>18.</td>
<td>18.</td>
<td>Splunk</td>
<td>Search engine</td>
<td>51.29</td>
<td>+2.38</td>
<td>+9.06</td>
</tr>
<tr>
<td>18.</td>
<td>16.</td>
<td>16.</td>
<td>16.</td>
<td>Hive</td>
<td>Relational DBMS</td>
<td>48.82</td>
<td>+1.01</td>
<td>-4.71</td>
</tr>
<tr>
<td>19.</td>
<td>19.</td>
<td>19.</td>
<td>19.</td>
<td>SAP HANA</td>
<td>Relational DBMS</td>
<td>43.42</td>
<td>+0.68</td>
<td>+5.22</td>
</tr>
<tr>
<td>20.</td>
<td>25.</td>
<td>25.</td>
<td>25.</td>
<td>MariaDB</td>
<td>Relational DBMS</td>
<td>38.53</td>
<td>+1.65</td>
<td>+14.31</td>
</tr>
<tr>
<td>21.</td>
<td>21.</td>
<td>21.</td>
<td>21.</td>
<td>Neo4j</td>
<td>Graph DBMS</td>
<td>36.37</td>
<td>+0.80</td>
<td>+2.83</td>
</tr>
</tbody>
</table>

http://db-engines.com/en/ranking
WARNING
OPINIONS AHEAD
We’re going to look at 5

Relational Document Search Configuration Service

Key Value
Landscape: Relational

Tag Line: “It’s all just relations”

Nice: Transactions, read speedz, math!

Less Nice: Less scalable writes, bad at non-relational queries, $$

```
SELECT *
FROM
    review r INNER JOIN
    user u on r.user_id = u.id
WHERE u.first_name = 'Joey'
ORDER BY u.id LIMIT 20;
```
Landscape: Document

Tag Line: “Data is really bags of keys”

Nice: Less impedance mismatch, easy to setup

Less Nice: Correctness is hard

mongoDB

CouchDB

RethinkDB
Landscape: Key Value, in memory*

Tag Line: “It’ll be fast when it’s in RAM”

Nice: So fast, relatively easy to scale

Less Nice: often (ab)used as a primary database

* redis offers eventual persistence
Landscape: Key Value, durable

Tag Line: “Bigtable was cool”
Nice: So much write speedz, horizontally scale
Less Nice: Slow reads, CAP is confusing?, “query” modeling

<table>
<thead>
<tr>
<th>Row</th>
<th>1</th>
<th>4</th>
<th>17</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row 0</td>
<td>1</td>
<td>4</td>
<td>17</td>
<td>24</td>
</tr>
<tr>
<td>Row 1</td>
<td>89</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row 2</td>
<td>45</td>
<td>78</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row 3</td>
<td>32</td>
<td>34</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>Row 4</td>
<td>66</td>
<td>89</td>
<td>89</td>
<td></td>
</tr>
<tr>
<td>Row 5</td>
<td>28</td>
<td>50</td>
<td>75</td>
<td>99</td>
</tr>
</tbody>
</table>

Fast
Landscape: Search Engines

Tag Line: “Index everything”
Nice: Text search, aggregations,
Less Nice: Maybe consistent, reading your writes is a feature

| Brutus  | → | 1  | 2  | 4  | 11 | 31 | 45 | 173 | 174 |
| Caesar  | → | 1  | 2  | 4  | 5  | 6  | 16 | 57  | 132 |
| Calpurnia | → | 2  | 31 | 54 | 101 |

Dictionary

Postings
Landscape: Coordination Services

Tag Line: “Your distributed locks are just files”

Nice: Finally a consistent datastore, ephemeral nodes and watches are sweet

Less Nice: Availability is actually pretty nice
Joey’s guide to picking a datastore

1. Does it satisfy the business requirement?
2. Does it have a solid community?
3. Does it plug into your stack?
4. How badly broken does it appear?

“slightly broken”  “horribly broken”
Yelp Stack?
Open Source Datastores @ Yelp

MySQL
Memcache
Lucene
Scribe

1 datacenter
~80 developers
~1 “service”
Open Source Datastores @ Yelp

MySQL
Memcache
Lucene
Scribe

2011
1 datacenter
~80 developers
~1 “service”

2012
2 datacenters
~100 developers

2013

2014
~5 “services”
Case Study: Distributed Configuration

**Business Need:** Distributed locks, master election, service registry

**Deciding Factors:** Zookeeper just works.

**Regret:** Four letter words are not really an API
Case Study: Search Engine

**Business Need:** Full text search

**Deciding Factors:** API, features, community

**Regret:** Elasticsearch sorta loses data a lot
What does it enable?

- Homepage
- Business Search
- Your Next Review Awaits
- Event/Talk/List
What does it enable?

- Search Page
  - Business Search
  - Business Highlights
  - Delivery + Reservation
  - Member/Talk/Event/Lists
Open Source Datastores @ Yelp

MySQL     Zookeeper
Memcache   Elasticsearch
Lucene     Scribe

2011
1 datacenter
~80 developers
~1 “service”

2012
2 datacenters
~100 developers
~5 “services”
Open Source Datastores @ Yelp

MySQL
Memcache
Lucene
Scribe

Zookeeper
Elasticsearch
Kafka

2011
1 datacenter
~80 developers
~1 “service”

2012
2 datacenters
~100 developers
~5 “services”

2013
4 datacenters
~200 developers
~30 “services”

2014
Open Source Datastores @ Yelp

MySQL  Memcache  Zookeeper  Elasticsearch  Kafka

<table>
<thead>
<tr>
<th>Year</th>
<th>1 datacenter</th>
<th>2 datacenters</th>
<th>4 datacenters</th>
<th>6 datacenters</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011</td>
<td>1 “service”</td>
<td>~80 developers</td>
<td>~100 developers</td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td>2 “services”</td>
<td>~5 “services”</td>
<td>~200 developers</td>
<td>~400 developers</td>
</tr>
<tr>
<td>2013</td>
<td></td>
<td>~30 “services”</td>
<td></td>
<td>~100 “services”</td>
</tr>
<tr>
<td>2014</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Open Source Datastores @ Yelp

MySQL  Memcache  Zookeeper  Elasticsearch
Lucene  Scribe  Kafka

2011  
1 datacenter  
~80 developers  
~1 “service”

2012  
2 datacenters  
~100 developers  
~5 “services”

2013  
4 datacenters  
~200 developers  
~30 “services”

2014  
6 datacenters  
~400 developers  
~100 “services”
Case Study: Durable Key Value

**Business Need:** All the writes, globally

**Deciding Factors:** Community, global writes

**Regret:** Python driver is a bummer sometimes
Open Source Datastores @ Yelp

MySQL
Memcache
Lucene
Scribe

Zookeeper
Elasticsearch

Kafka

Cassandra

2011
1 datacenter
~80 developers
~1 “service”

2012
2 datacenters
~100 developers
~5 “services”

2013
4 datacenters
~200 developers
~30 “services”

2014
6 datacenters
~400 developers
~100 “services”
● Why datastores?
● The open source datastore landscape

● Building a datastore platform
● Realities of distributed datastores
Building the Platform

1. Deploy
2. Configure
3. Monitor
4. Discover
5. Tooling
Deploy

- Super easy to spin up an ASG
- Slightly unstable, but getting there!
Deploy

It's a cassandra node!
Configure

External Node Classifiers

An external node classifier is an arbitrary script or application which can tell Puppet which classes a node should have. It can replace or work in concert with the node definitions in the main site manifest (site.pp).

Depending on the external data sources you use in your infrastructure, building an external node classifier can be a valuable way to extend Puppet.

```erb
class elasticsearch_cluster ( $cluster_name, $role, $es_version, ) {

class cassandra_cluster ( $cluster_name, $version, $team, $start_cassandra = true, ) {

class zookeeper_cluster ( $cluster_name, $team, $version, $datastore, ) {
```
Configure

- Functions help
- Hierarchical config is better
Configure - Hierarchy

```yaml
---
version: 4
data_dir: data
hierarchy:
- name: cluster_name
  backend: yaml
  path: clusters/%{::cassandra_cluster::cluster_name}
- name: version
  backend: yaml
  path: version_defaults/%{::cassandra_cluster::version}
- name: default_cluster
  backend: yaml
  path: default_cluster
```
Configure - Hierarchy

- clusters
  - ad_backend.yaml
  - generic.yaml
  - kairos.yaml
  - profilistic.yaml
  - zipkin.yaml

- default_cluster.yaml
- version_defaults
  - 2.1.13.yaml
  - 2.1.4.yaml
  - 2.2.5.yaml
  - 2.2.7.yaml
Configure - Flexible

```
{
  "phi_convict_threshold": 12,
  "stream_throughput_outboundMegabits_per_sec": 500
}
```

```
{
  "auto_bootstrap": true,
  "client_encryption_options": {
    "enabled": true,
    "keystore": "/nail/etc/cassandra/keystore",
    "keystore_password": "",
    "require_client_auth": true,
    "truststore": "/nail/etc/cassandra/client_truststore",
    "truststore_password": ""
  },
  "incremental_backups": true,
  "server_encryption_options": {
    "internode_encryption": "all",
    "keystore": "/nail/etc/cassandra/keystore",
    "keystore_password": "",
    "truststore": "/nail/etc/cassandra/server_truststore",
    "truststore_password": ""
  }
}
```
Page: Cluster Health
Ticket: Node Health
IRC (/dev/null): everything else that breaks
Discover

- **configure_synapse**
- **configure_nerve**
- HAProxy
- Zookeeper
- Nerve
- hacheck
- Database

**Configuration**
- Synapse
- Client

**Registration**
- Synapse
- Zookeeper
- Nerve

**Healthchecks**
- Zookeeper
- Nerve

**Discover**
- HAPerxoy
Discover More

- **Synapse**
  - `configure_synapse`
  - Configuration

- **Zookeeper**
  - Configuration
  - Registration

- **Nerve**
  - `configure_nerve`
  - Configuration
  - Healthchecks

- **Client**
  - Configuration

- **JSON files**

- **hacheck**
  - Healthchecks

- **Database**

---

See more details in the diagram.
Discover More

cat /var/run/synapse/services/elasticsearch-testing173.main.json

| jq . |
| [] |
| {} |
| "name": "10-40-11-48-uswest1adegr" |
| "host": "10.40.11.48" |
| "port": 14900 |
| "id": 1174 |
| "weight": 16 |
| {} |
| "name": "10-40-11-118-uswest1adegr" |
| "host": "10.40.11.118" |
| "port": 14900 |
| "id": 1164 |
| "weight": 16 |
Tooling

1. Restarts
2. Monitoring
3. (adv) Schema mgmt
4. (adv) Autoscaling

```bash
~ $ es_tool cat nodes h=ip,heap.percent,ram.percent,load,master

<table>
<thead>
<tr>
<th>ip</th>
<th>heap.percent</th>
<th>ram.percent</th>
<th>load</th>
<th>master</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.40.18.235</td>
<td>42</td>
<td>70</td>
<td>0.06</td>
<td>*</td>
</tr>
<tr>
<td>10.40.11.48</td>
<td>58</td>
<td>67</td>
<td>0.34 m</td>
<td></td>
</tr>
<tr>
<td>10.40.11.118</td>
<td>60</td>
<td>68</td>
<td>0.03 m</td>
<td></td>
</tr>
<tr>
<td>10.40.20.214</td>
<td>52</td>
<td>70</td>
<td>0.04 m</td>
<td></td>
</tr>
</tbody>
</table>
```
Schemas

jlynch@10-40-11-48-uswest1adevc:/nail/etc/datastores$ ls
account_search/  ad_sales_recommendation/  data_quality/  new_business/
activity_feed/  biz_dup_queue/  distsys_streaming/  ordering/
bizfeed/  business_features/  geocoder/  OWNERS*
business_location_search/  business_match/  geolocator/  payload/
business_name_search/  business_similarity_search/  highlights/  payments/
community/  community-data/  kafka_benchmark/  platform/
contributions/  kafka_guinea_pigs/  kafka/  profilistic/
crm_list/  kairos/  kashserv/  README.md*

reservation_pages/  reviewboard/  reviewuggestion/
search_indexing/  serendipity/
small_search/
test_results/
user_engagement/
zipkin/
Schemas

multiple datastores!

"where does it go?"

Ownership!
Schemas

```bash
jlynch@10-40-11-48-uswest1devc:~$ es_tool apply_schemata --help
usage: es_tool apply_schemata [-h] [--force]
    [--schema-location SCHEMA_LOCATION]
    [--blacklist [BLACKLIST [BLACKLIST ...]]]

apply_schemata applies all schemas from corresponding schema groups that are tracked at `--schema-location`, but undeployed on the elasticsearch cluster.

optional arguments:
   -h, --help            show this help message and exit
   --force               Force the action ignoring whether this host is actually responsible for applying schemas on this cluster.
   --schema-location SCHEMA_LOCATION
                           Base path where schema files for all schema groups exist, defaults to /nail/etc/datastores.
   --blacklist [BLACKLIST [BLACKLIST ...]]
                           List of schema groups to blacklist. Schemas corresponding to these schema groups shall not be monitored.

jlynch@10-40-27-177-uswest1devc:~$ cassandra_tool apply_schemata --help
usage: cassandra_tool apply_schemata [-h] [--force]
    [--default-replication-factor DEFAULT_REPLICATION_FACTOR]
    [--schema-location SCHEMA_LOCATION]
    [--cqlshrc CQLSHRC]
    [--blacklist [BLACKLIST [BLACKLIST ...]]]

apply_schemata applies all schemas from corresponding schema groups that are tracked at `--schema-location`, but undeployed on the cassandra cluster.

optional arguments:
   -h, --help            show this help message and exit
   --force               Force the action ignoring whether this host is actually responsible for applying schemas on this cassandra cluster.
   --default-replication-factor DEFAULT_REPLICATION_FACTOR
                           Default replication factor for keyspaces if not overridden, defaults to 3.
   --schema-location SCHEMA_LOCATION
                           Base path where schema files for all schema groups exist, defaults to /nail/etc/datastores.
   --cqlshrc CQLSHRC
                           Path to cqlshrc configuration file containing information like ssl options, authentication options, etc ...
   --blacklist [BLACKLIST [BLACKLIST ...]]
                           List of schema groups to blacklist. Schemas corresponding to these schema groups shall not be
```
Autoscaling

Auto Scaling lifecycle hooks enable you to perform custom actions as Auto Scaling launches or terminates instances. For example, you could install or configure software on newly launched instances, or download log files from an instance before it terminates.

This tool provides the ability to handle events raised by AWS autoscaling lifecycle policies.

optional arguments:

- -h, --help: show this help message and exit
- --zookeeper-connection-string ZK_CONN_STRING: Zookeeper connection string of the form '<host1>:<port>,<host2>:<port>' used to co-ordinate handling of events among instances of the cluster. It is highly recommended to use this option to prevent race conditions.
- --max-timeout MAX_TIMEOUT: Maximum time in number of hours to wait before abandoning the event, defaults to 6
- --sqs-queue-region SQS_QUEUE_REGION: AWS region for the event source SQS queue, defaults to us-west-1
- --heartbeat-timeout HEARTBEAT_TIMEOUT: Heartbeat timeout of the associated lifecycle hook, in seconds. Defaults to 3600
Why datastores?
The open source datastore landscape
Building a datastore platform
Realities of distributed datastores
Reality: Data coming in

Sources of Truth
- MySQL
- Kafka
- Zookeeper

Derived Data
- Elasticsearch
- Cassandra
- Memcache

- ETLs
- RBR
- Batches
- Indexers
Better: “Data pipeline”
Billions of Messages a Day - Yelp's Real-time Data Pipeline

This is the first post in a series covering Yelp's real-time streaming data infrastructure. Our series explores in-depth how we stream MySQL updates in real-time with an exactly-once guarantee, how we automatically track & migrate schemas, how we process and transform streams, and finally how we connect all of this into datastores like Redshift and Salesforce.

Read the posts in the series:
- Billions of Messages a Day - Yelp's Real-time Data Pipeline
- Streaming MySQL tables in real-time to Kafka
- More Than Just a Schema Store
- PaaStorm: A Streaming Processor
- Data Pipeline: Salesforce Connector

Reality: Data going out

Sources of Truth
MySQL
Kafka
Zookeeper

Derived Data
Elasticsearch
Cassandra
Memcache
Better: Proxies

- Sources of Truth
  - MySQL
  - Kafka
  - Zookeeper

- Derived Data
  - Elasticsearch
  - Cassandra
  - Memcache
Our Solution

Search Infrastructure at Yelp has been employing a proxy service we call Apollo to separate the concerns of the developers from the implementation details so that now our infrastructure looks like this:

Reality: Operations

- Development (Software Engineering)
- QA (Quality Assurance)
- DevOps
- Operations

DBA
Better: DevOps

Developers ought be responsible for their datastores

You’re going to need:
- Docs
- Lectures
- Support channels
Better: DevOps

Topic for #cassandra is "y/cassandra | susanne, shar, or krp should be able to help. Ping one of them! | <jarvis> why didn't we just write the data correctly the first time"
Takeaways

● You have lots of choices
● Don’t choose them all
● Distribute responsibility
● Prepare for change
● One size fits...some
fb.com/YelpEngineers
@YelpEngineering
engineeringblog.yelp.com
github.com/yelp
Photo Attribution

- Latency Numbers Every Programmer Should Know, Jonas Bonér
  https://gist.github.com/jboner/2841832
- Posting list; Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008.
- Devops venn diagram, Kharnagy
  https://commons.wikimedia.org/wiki/File:Devops.svg
- Server rack, unknown author
  http://www.clipartkid.com/server-rack-cabinet-clip-art-at-clker-com-vector-clip-art-online-IPTtWc-clipart/
- Kitten, Skitterphoto