Notes on the architecture, design, and data processes in openFDA

OpenFDA uses cutting edge technologies and is a pilot for how FDA can develop and deploy novel applications in the public cloud securely and efficiently in the future. In these notes, we provide a high-level plain language description of the logical architecture, data sources, data processing, data harmonization, and website technologies.

Logical Architecture

The architecture and technology were chosen to make openFDA scalable; quickly responsive; transferable to new technologies as they mature; easily accessible by application developers, researchers, and the general public; and transparent. The data are on the cloud that has been approved for federal use (Amazon Web Services East).[1] The figure shows openFDA is built using modern, open standards and leveraging open source and cloud technologies. The system uses modules that work together or in sequence and can be transparently replaced as technology improves. The target consumers are other applications that use openFDA; applications can query openFDA in the form of Uniform Resource Locators (URLs). Of course, researchers and the general public can create and run queries, as well.

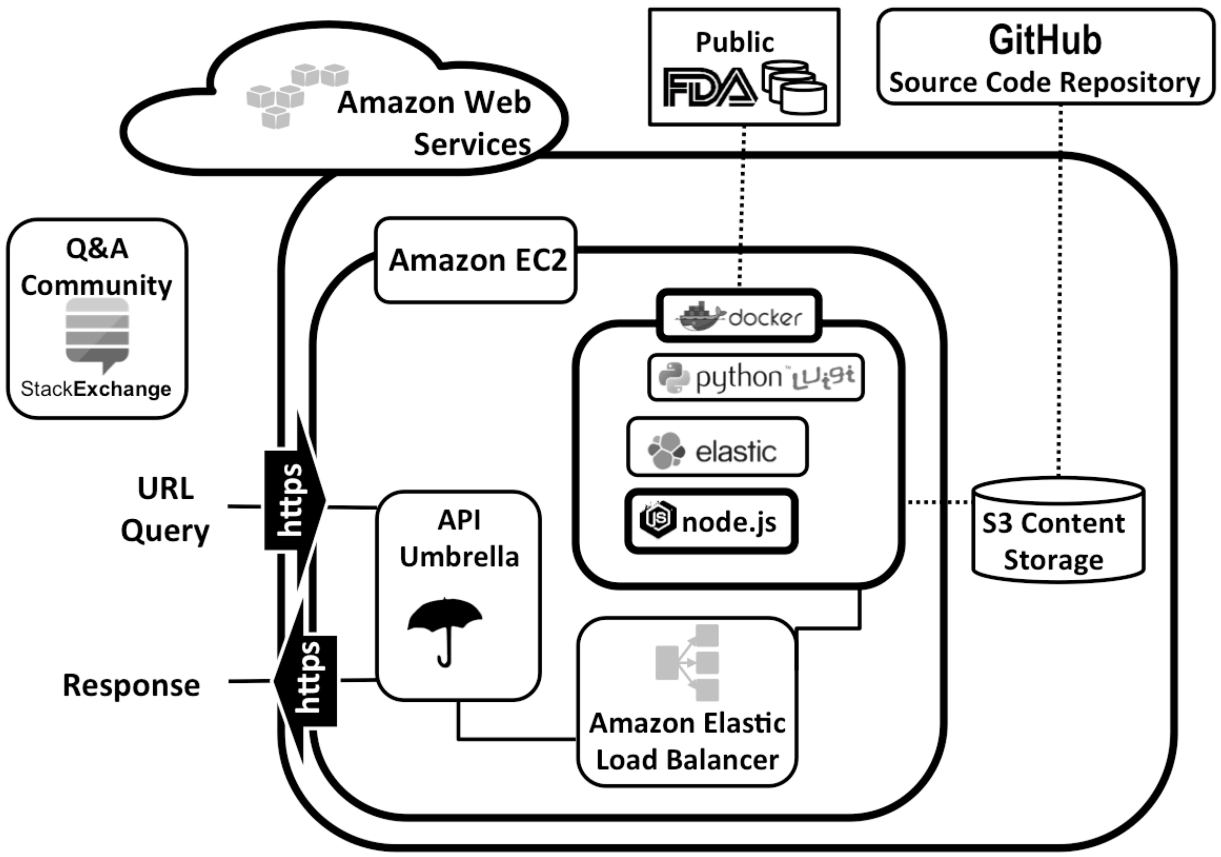

Figure. openFDA logical architecture.

OpenFDA is hosted in one of the secure cloud environments approved for federal use (Amazon Web Services US-East)[1] and uses Amazon Elastic Compute Cloud (EC2).[2] All of the content and data are encrypted to be read-only by the public. The platform is transportable to different cloud environments since it is deployed within a Docker container,[3] a complete file system that contains everything it needs to run the software. In addition, the data is portable to other software. Node.js is used within Docker as the open source, cross-platform runtime environment for server-side and networking applications. Node.js, using JavaScript, enables the creation of highly scalable fast webservers, has a simple and elegant programmer interface, and has a large library of open source modules.[4]

Git, a source control system, is used during the development and modification of any of the openFDA content (data processing steps, analysis code, encryption protocols, and settings for the other modules). All of the code is copied into the independent GitHub Source Code Repository for openFDA.[5] GitHub is a popular repository for open source code.

The public data are regularly drawn from public FDA files. Luigi Python (the open source version created by Spotify to handle massive volumes of digitized music)[6] and Elasticsearch[7] are both used to prepare and load the data. Python manages the workflow of data and software modules. Elasticsearch is a fast, scalable, full text JSON (see next sentence) database built upon the open source Lucene project that has an easy to use RESTful API (see next paragraph). The final form of each dataset is JavaScript Object Notation (JSON), an open standard data format that is independent of programming language and supported by many programming languages, including Python, JavaScript, and R.[8-9] All of the openFDA content is stored in Amazon Simple Storage Service (S3).[10]

The Application Programming Interfaces (APIs) contain the automation for accessing and using the data.[11] They are Representational State Transfer (REST) type, to take advantage of modularity of the code, the ability to cache queries and responses, the ability to layer services, and scalability.[12]

Queries are written in Lucene query syntax[13]. All queries begin with "https://api.fda.gov/" and go on to specify the database API name and then the search or count specifications. The path for queries is represented with solid lines in the figure. API Umbrella[14] is an open source tool that keeps track of query statistics, and administers the free user-specific API keys that allow heavy use. API Umbreall a receives the incoming query and fetches the results. If the query is a duplicate, the response is found in API Umbrella's historical query cache. Otherwise, Node.js and Elasticsearch are used to search and analyze the specified JSON dataset in cloud storage using real-time distributed methods. The load balancer balances all the jobs.[15] OpenFDA has been able to handle over 100 requests per second across millions of records, which are all built in an open source environment and can be adopted for other public health big data challenges.

StackExchange hosts questions, answers, and discussions related to openFDA. [16]

Data Source Details

Four main data sources are currently available in openFDA: adverse event reports for drugs and devices, recall reports for all products, and drug labeling.

- **Drug adverse event reports** includes approximately five million publicly available drug adverse event and medication error reports, of which almost 1.2 million reports are from 2013.[17] Since 2004, FDA has published online quarterly drug adverse reaction reports from the FDA Adverse Event Reporting System (FAERS).[18-19] The files listed on this webpage contain raw data extracts for the indicated time ranges, are not cumulative, and require reconstruction into a relational database. The historical files are in Standardized General Markup (SGM) format and the current ones are in Extensible Markup Language (XML). Part of processing the files requires preserving only the most recent record of a particular reported incident. LevelDB software[20] is used to do the filtering; LevelDB (with Snappy[21] compression, a fast data compression and decompression library) is compatible with Node.js, C++, and Python.

- Safety report data contain semi-structured information about recalls, market withdrawals and safety alerts of FDA-regulated products archived in the Recall Enterprise System (RES)[22] since 2012. Recalling defective or dangerous products, by removing them from the market or correcting the problem, is one of the ways of protecting the public.[23] FDA provides various ways to access the recalls data, including an RSS feed, a Flickr stream, a search interface, weekly downloadable XML files (the ones used by openFDA), and weekly downloadable CSV files. The openFDA API provides a new option for easy and fast access. [24]
- **Labeling** data are composed of the updated Structured Product Labeling (SPL) data for 68,000 currently approved drugs. The labeling contains information necessary to inform healthcare providers about the safe and effective use of the drug for its approved use(s).[25] The FDA SPL staff make the updated SPL files in XML format available to the openFDA staff in parallel to sending them to the National Library of Medicine, where they are available to the public.[26]
- **Medical device adverse event reports** include over 4 million reports of serious injuries, deaths, and device malfunctions, from 1991 to the present. In recent years, the

Manufacturer and User Facility Device Experience (MAUDE) has been receiving several hundred thousand reports per year. The source is the public downloadable version of MAUDE, which is in multiple zip files composed of pipe-delimited text files that require reconstruction into a relational database.[27]

Processing of the Source Data

The public data sources are converted to flat JSON files. Both of the adverse event report source databases are in relational flat tables that are processed in openFDA to each form a large flat file with long records. The labelling source data are in XML format, with varying levels of hierarchy used for different records; the new flat table of records had to be designed after exploration of the extent of hierarchy in different sections of the individual records.

The recall reports source files are also in XML format that the openFDA process converts to a flat file**.**

Relational databases were popular for the last several decades because storage was relatively expensive. However, the complexity of relational databases raises the risk of inaccurate analysis strategies. Software designed for big data has made it feasible to quickly execute search and analysis commands on very large flat files.

Harmonization Process

To address issues related to differences in the structure of the three drug databases (adverse event reports, recalls, and labeling), openFDA features harmonization on drug identifiers (generic name, brand name, etc), to make it easier to both search for and understand the drug products returned by API queries. The additional “harmonization”, or “openFDA”, fields are created from the following four databases:

- NDC Directory.[28] OpenFDA uses application number, brand name, dosage form, generic name, manufacturer name, original packager indicator, NDC, type of drug product, route of administration, and active ingredients.
- SPL-Pharmacological Class Mappings.[29] OpenFDA uses all four types of pharmacologic class: mechanism of action, chemical structure, physiological effect, and approved indication class.
- SPL-RxNorm Mappings.[30] Synonym drug names are grouped into RxNorm “concepts”, and connected NDC, other drug names, ingredients, manufacturer, and pill attributes. OpenFDA uses the RxNorm Concept Unique Identifier that incorporates the drug concept, ingredients, strength, and dosage forms.
- Substance Registration System.[31] OpenFDA uses the Unique Ingredient Identifier.

The harmonized openFDA fields are then added to any record in the recalls, drug adverse event reports, and SPL flat files that match a field in the harmonization database. For recalls of drugs, the names of drugs and manufacturers, as well as NDC or UPC codes, were generally provided in free-text fields with other text. Regular expression-based extractors were built to identify this information for harmonization.

Website Technology

The design of the open.fda.gov website draws on best practices in agile development, intuitive user experience, and data visualization. Its aim is to provide a unified, consistent presentation for all datasets to facilitate ease of learning both about the APIs and the datasets themselves. The site is organized thematically around broad data types (drugs, devices, and foods), rather than around datasets or FDA's internal organization. This scheme is deliberate, in order to more closely align with website users' mental models of FDA data. The website is characterized by a combination of interactive programmer-oriented example queries, visualizations, and examples that explain the nature of the data and how to use the query syntax and JSON results. Unlike most API websites, it recognizes that non-technical members of the public have an interest in these data, and employs the design principle of progressive disclosure to provide multiple layers of information depth. Interactive data visualizations and examples, with plain language annotation, are oriented towards members of the public but usable by both programmer and non-programmer users of the website. Plain, straightforward language is used throughout, including in field-by-field documentation of the JSON results for each API endpoint.

The website was designed in an iterative fashion, incorporating feedback from both internal and external stakeholders. Since its launch, changes have been made to clarify documentation in response to feedback from the open source community.

Publicly available data provided through openFDA are in the public domain with a CC0 Public Domain Dedication[32]. The website was built with open source software: Jekyll for overall structure,[33] Bootstrap for responsive design including mobile compatibility,[34] Grunt for optimizing JavaScript,[35] and LESS/CSS[36] and D3[37] and C3[38] for data visualization.

Conclusion

OpenFDA brings a new model of big data search and analytics across disparate and complex sources by simplifying dataset structures and using modular open source technology.

By: Taha A. Kass-Hout, MD, Roselie A. Bright, ScD, Adam Baker

References

1. FedRAMP Compliant Systems. *FedRAMP, US General Services Administration.* https://www.fedramp.gov/marketplace/compliant-systems/. Accessed in July 2015.
2. Amazon EC2. *Amazon*. 2015. http://aws.amazon.com/ec2/?sc_channel=PS&sc_campaign=acquisition_US&sc_publisher=bing&sc_medium=ec2_b&sc_content=ec2_bmm&sc_detail=+amazon%20+ec2&sc_category=ec2&sc_segment=7005634148&sc_matchtype=p&sc_country=US&s_kwcid=AL!4422!10!7005634148!50080015136&ef_id=VLAuEwAABfo1-Hwu:20150717225734:s. Accessed in July 2015.
3. Build, Ship, Run. *Docker, Inc*. https://www.docker.com/. Accessed in July 2015.
4. Node.js. *Node.js Foundation*. 2015. https://nodejs.org/. Accessed in July 2015.
5. FDA/openFDA. *GitHub, Inc*. 2015. https://github.com/FDA/openfda. Accessed in July 2015.
6. Luigi. *Python Software Foundation*. 2014. https://pypi.python.org/pypi/luigi. Accessed in July 2015.
7. Elasticsearch: Search & Analyze Data in Real Time. *Elasticsearch*. 2015. https://www.elastic.co/products/elasticsearch. Accessed in July 2015.
8. Introducing JSON. *JSON*. http://json.org. Accessed in July 2015.
9. The R Project for Statistical Computing. *The R Foundation*. http://www.r-project.org/. Accessed in July 2015.

10. Amazon Simple Storage Service Developer Guide. *AWS Documentation*. 2015. http://docs.aws.amazon.com/AmazonS3/latest/dev/Welcome.html. Accessed in July 2015.

11. Orenstein D. Application Programming Interface. *Computerworld*. January 10, 2000. http://www.computerworld.com/article/2593623/app-development/application-programming-interface.html. Accessed in July 2015.

12. Kay R. Representational State Transfer (REST). *Computer World*. August 6, 2007. http://www.computerworld.com/article/2552929/networking/representational-state-transfer--rest-.html. Accessed in July 2015.

13. Lucene. *Apache Software Foundation*. June 21, 2013. https://lucene.apache.org/core/2_9_4/queryparsersyntax.html. Accessed in July 2015.

14. API Umbrella. http://apiumbrella.io/. Accessed in July 2015.

15. Elastic Load Balancing. *Amazon Web Services*. Aws.amazon.com/elasticloadbalancing/. Accessed in July 2015.

16. StackExchange. http://stackexchange.com/search?q=openFDA. Accessed in July 2015.

17. Reports Received and Reports Entered into FAERS by Year. *Food and Drug Administration*. August 6, 2014. http://www.fda.gov/Drugs/GuidanceComplianceRegulatoryInformation/Surveillance/AdverseDrugEffects/ucm070434.htm in July 2015. Accessed in July 2015.

18. The Adverse Event Reporting System (AERS): Older Quarterly Data Files. *Food and Drug Administration.* August 15, 2013.

http://www.fda.gov/Drugs/GuidanceComplianceRegulatoryInformation/Surveillance/AdverseDrugEffects/ucm083765.htm. Accessed in July 2015.

19. FDA Adverse Event Reporting System (FAERS): Latest Quarterly Data Files. *Food and Drug Administration.* June 16, 2015. http://www.fda.gov/Drugs/GuidanceComplianceRegulatoryInformation/Surveillance/AdverseDrugEffects/ucm082193.htm . Accessed in July 2015..

20. LevelDB. *LevelDB*. http://leveldb.org/. Accessed in July 2015.

21. Snappy: a fast compressor/decompressor. *Google Project Hosting*. http://code.google.com/p/snappy/. Accessed in July 2015.

22. Enforcement Reports. *Food and Drug Administration*. July 15, 2015. http://www.fda.gov/Safety/Recalls/EnforcementReports/default.htm. Accessed in July 2015.

23. FDA 101: Product Recalls – From First Alert to Effectiveness Checks. *Food and Drug Administration*. Updated April 29, 2015. http://www.fda.gov/ForConsumers/ConsumerUpdates/ucm049070.htm. Accessed in July 2015.

24. Kass-Hout T. OpenFDA provides ready access to recall data. *Food and Drug Administration.* August 8, 2014. https://open.fda.gov/update/openfda-provides-ready-access-to-recall-data. Accessed in July 2015.

25. Kass-Hout T. Providing easy public access to prescription drug, over-the-counter drug, and biological product labeling. *Food and Drug Administration*. August 18, 2014. https://open.fda.gov/update/drug-product-labeling/. Accessed in July 2015.

26. DailyMed. *National Library of Medicine, US National Institutes of Health*. http://dailymed.nlm.nih.gov/dailymed/. Accessed in July 2015.

27. Manufacturer and User Facility Device Experience Database – (MAUDE). *Food and Drug Administration.* May 7, 2015. http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/PostmarketRequirements/ReportingAdverseEvents/ucm127891.htm. Accessed in July 2015.

28. National Drug Code Database Background Information. *Food and Drug Administration*. June 14, 2012. http://www.fda.gov/drugs/developmentapprovalprocess/ucm070829.htm. Accessed in July 2015.

29. SPL Resources: Download all mapping files. *National Library of Medicine, US National Institutes of Health*. http://local-dailymed.nlm.nih.gov/dailymed/spl-resources-all-mapping-files.cfm. Accessed in July 2015.

30. RxNorm Overview. *National Library of Medicine, US National Institutes of Health*. January 5, 2015. http://www.nlm.nih.gov/research/umls/rxnorm/overview.html. Accessed in July 2015.

31. UNII List Download. Substance Registration System – Unique Ingredient Identifier (UNII). *National Library of Medicine, US National Institutes of Health*. Updated March 2015. http://fdasis.nlm.nih.gov/srs/jsp/srs/uniiListDownload.jsp. Accessed in July 2015.

32. Creative Commons Corp., CCO 1.0 Universal. https://creativecommons.org/publicdomain/zero/1.0/legalcode Accessed in July 2015.

33. Preston-Werner T. Transform your plain text into static websites and blogs. *Jeckyll*. 2015. Jekyllrb.com/. Accessed in July 2015.

34. Bootstrap. *Bootstrap*. Getbootstrap.com/. Accessed in July 2015.

35. GRUNT The JavaScript Task Runner. *Gruntjs*. Gruntjs.com/. Accessed in July 2015.

36. Getting started. *LESS/CSS*. Lesscss.org/#. Accessed in July 2015.

37. D3: Data-Driven Documents. *D3js.* D3js.org/. Accessed in July 2015.

38. Tanaka M. C3.js: D3-based reusable chart library. *C3js*. 2014. C3js.org/. Accessed in July 2015.