Real-Time Data Flows with Apache NiFi

June 2016

Manish Gupta

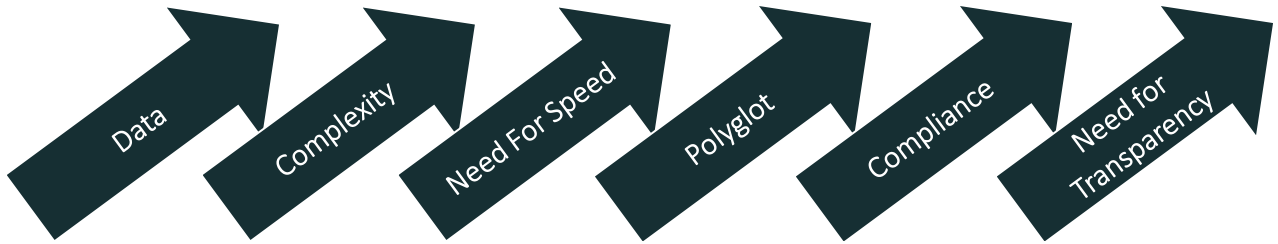
1. Data Flow Challenges in an Enterprise
2. Introduction to Apache NiFi
3. Core Features
4. Architecture
5. Demo – Simple Lambda Architecture
6. Use Cases
7. Q & A

01

Data Flow Challenges in an Enterprise

Connected Enterprises in a Distributed World

Evolution of Data Projects in an Enterprise



Simple VB/C++ applications working directly on Spreadsheet / Access databases.

OLTP applications (mostly web) using a RDBMS at backend, with some operational reporting capabilities.

Multiple OLTP applications exchanging data between themselves and trying to do BI reporting. But failed miserably in providing single version of truth

Emergence of EDW, ETL, BI, OLAP, MDM, Data Governance.

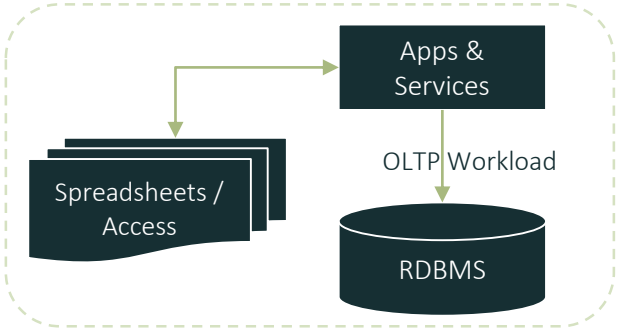
Issues in Scalability, Availability, Maintainability,

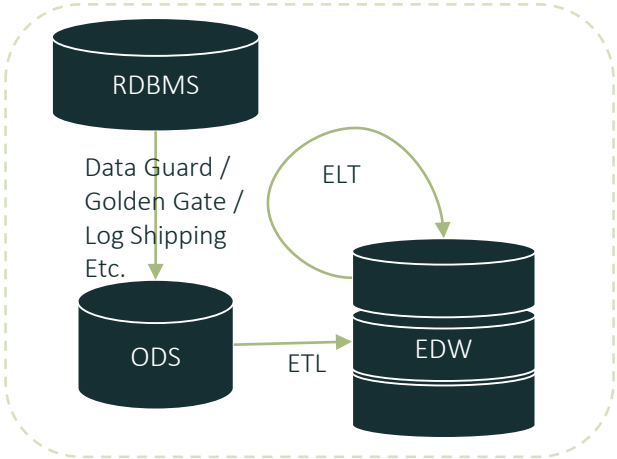
Performance etc. started showing up at totally different level.

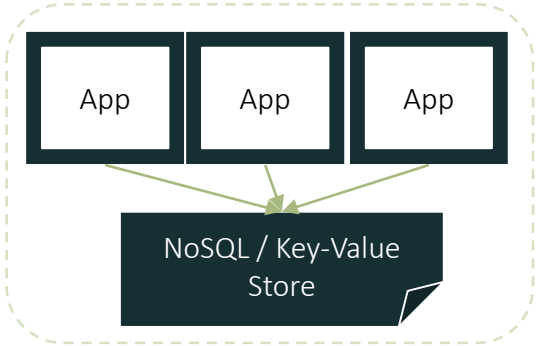
Emergence of MPP data warehouses, grid computing, Self-service BI tools and technology.

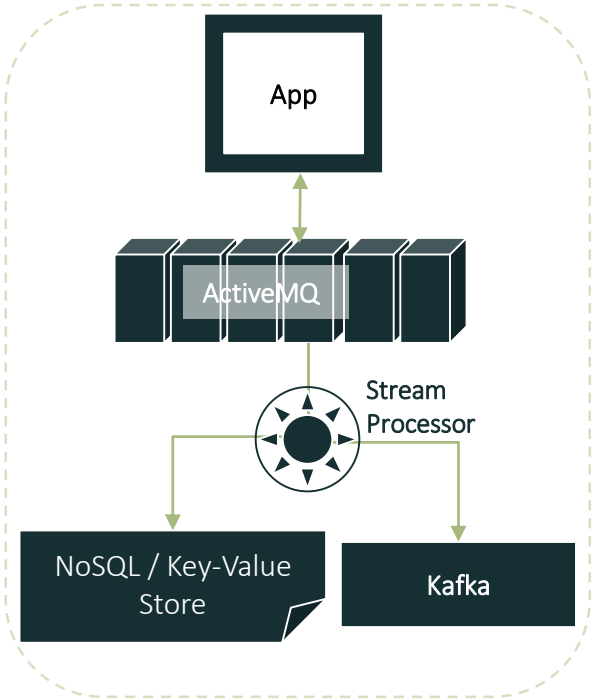
Big Data, Cloud/Hybrid Architecture, Real-Time analytics / Stream Processing, Document indexing/search, Log Capture & Analysis, Data Virtualization, NoSQL (Key-Value, Document Oriented, Column Family, Graph), In-memory database.

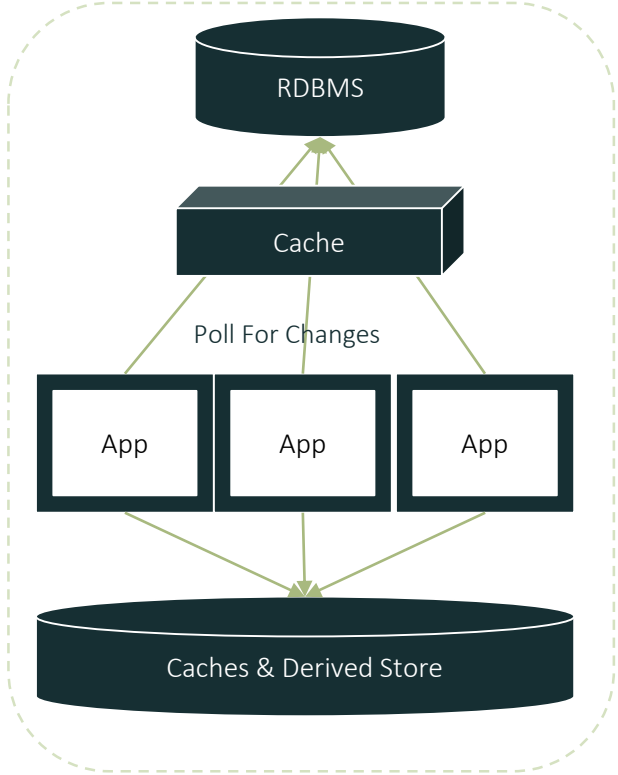
Application Architecture Pattern Silos

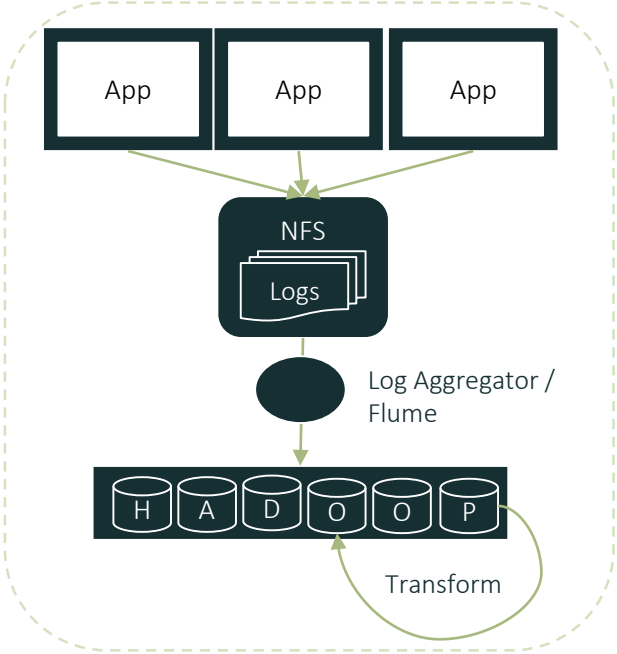


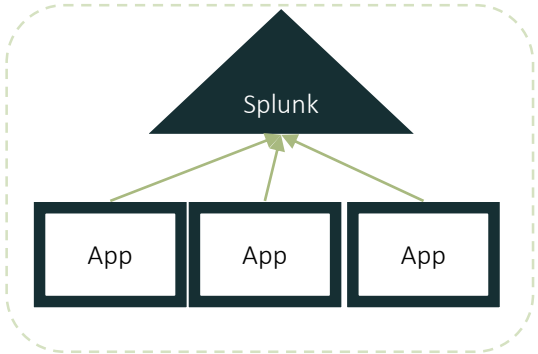


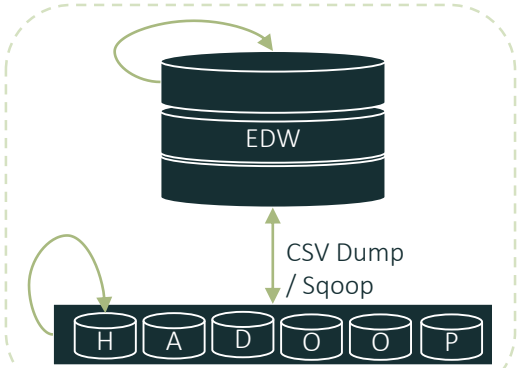


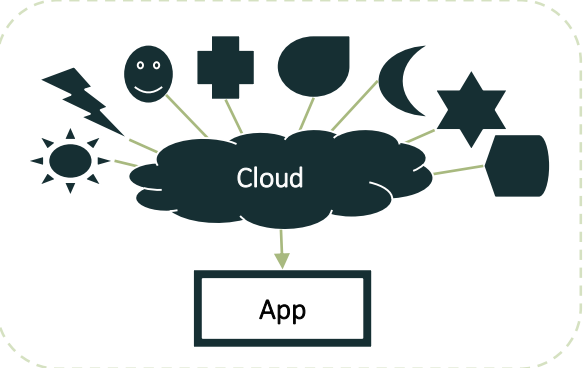




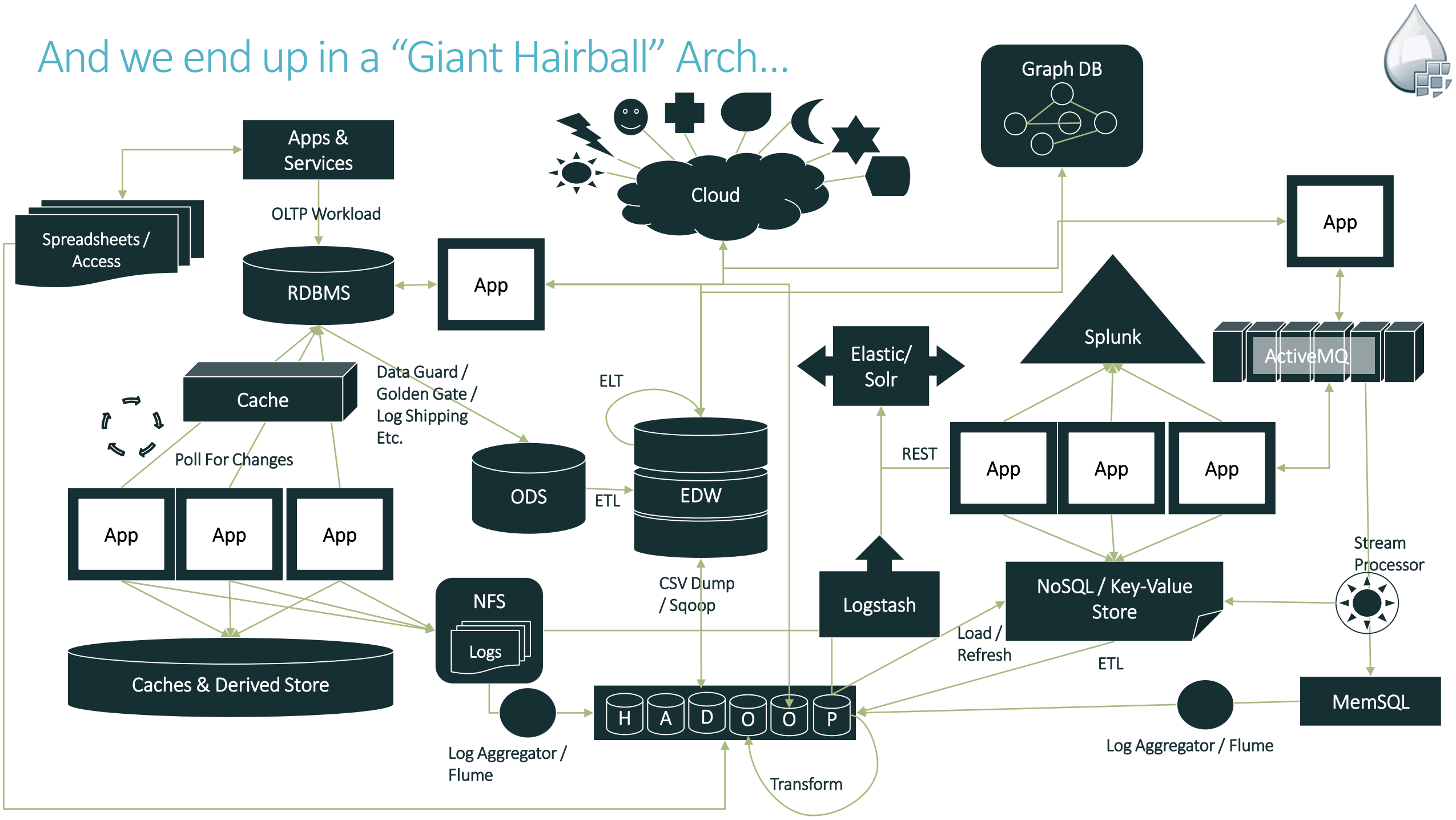








And we end up in a "Giant Hairball" Arch…



Data Ingestion Frameworks (excluding Traditional ETL Tools)

Apache Flume

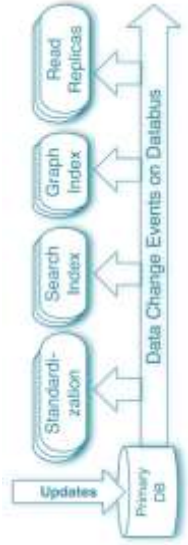
Apache Sqoop

Amazon Kinesis

Facebook Scribe

Apache Chukwa

Cloudera Morphlines



Fluentd

Mozilla Heka

Google Photon

Netflix Suro

Apache Kafka

Twitter Kestrel

Linkedin Databus

InfoSphere Streams

Apache Beam

Hadoop In, Hadoop Out

Elastic Logstash

Linkedin Gobblin

- Purpose built (Not designed with Universal Applicability)
- Induces lot of complexity in project architecture
- Hard to extend

Common Data Integration Problems

Size and Velocity

Messages in Streaming manner

Tiny to small files in micro batches time.

Small files in mini batches.

Medium to large files in batches.

Formats

CSV, TSV, PSV, TEXT, JSON, XML, XLS, XLSX, PDF, BMP, PRN

Avro, Protocol Buffer, Parquet, RC, ORC, Sequence File

Zip, GZIP, TAR, LZIP, 7z, RAR

Mediums

File Share, FTP, REST, HTTP, TCP, UDP

Schedule

Once, Every day, every hour, every minute, every second, continuous.

Mode

- Push / Pull / Poll

Asynchronous Operation Challenges

- Fast edge consumers + slow processors = everything breaks
- Process Message A first, all others can take a backseat.

Security

- Data should be secure – not just at rest, but in motion too.

Miscellaneous

- Can you route a copy of this to our NoSQL store as well after converting it to JSON.
- Ability to run from failure (checkpoint / rerun / replay)
- Merge small files to large files for Hadoop
- Break large files into smaller manageable chunks for NoSQL

02

Introduction

What is Apache NiFi, it's History, and some terminology.

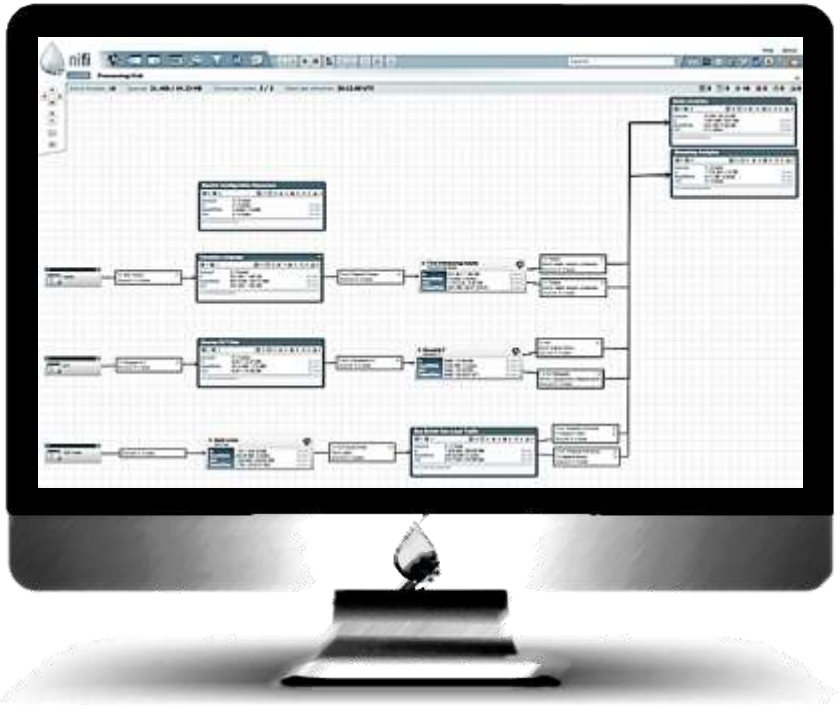
What is Apache NiFi

NiFi (short for “Niagara Files”) is a powerful enterprise grade dataflow tool that can collect, route enrich, transform and Process data in a scalable manner.

NiFi is based on the concepts of flow-based programming (FBP). FBP is a programming paradigm that defines applications as networks of "black box" processes, which exchange data across predefined connections by message passing, where the connections are specified externally to the processes.

Single combined platform for

- ✓ Data acquisition
- ✓ Simple event processing
- ✓ Transport and delivery
- ✓ Designed to accommodate highly diverse and complicated dataflows



- It has Visual command and control interface which allows you to define and manipulate data flows in real-time and with great agility.

Short History

- Developed by the National Security Agency (NSA) for over 8 years
- Open sourced in Nov 2014 (Apache). Major contributors were ex-NSA who formed a company named Onyara. Lead – Joe Witt.
- Become Apache Top Level Project in July 2015
- In August 2015, Hortonworks acquired Onyara
- In September 2015, Hortonworks released HDF 1.0 powered by NiFi. Current version is HDF 1.2
- HDF has got solid backing of Hortonworks.

Terminology

FlowFile (Information Packet)

Unit of data (each object) moving through the system

Content + Attributes (key/value pairs)

Processor (Black Box)

Performs the work, can access FlowFiles

Currently there are 135 different processors

Connection (Bounded Buffer)

Links between processors

Queues that can be dynamically prioritized

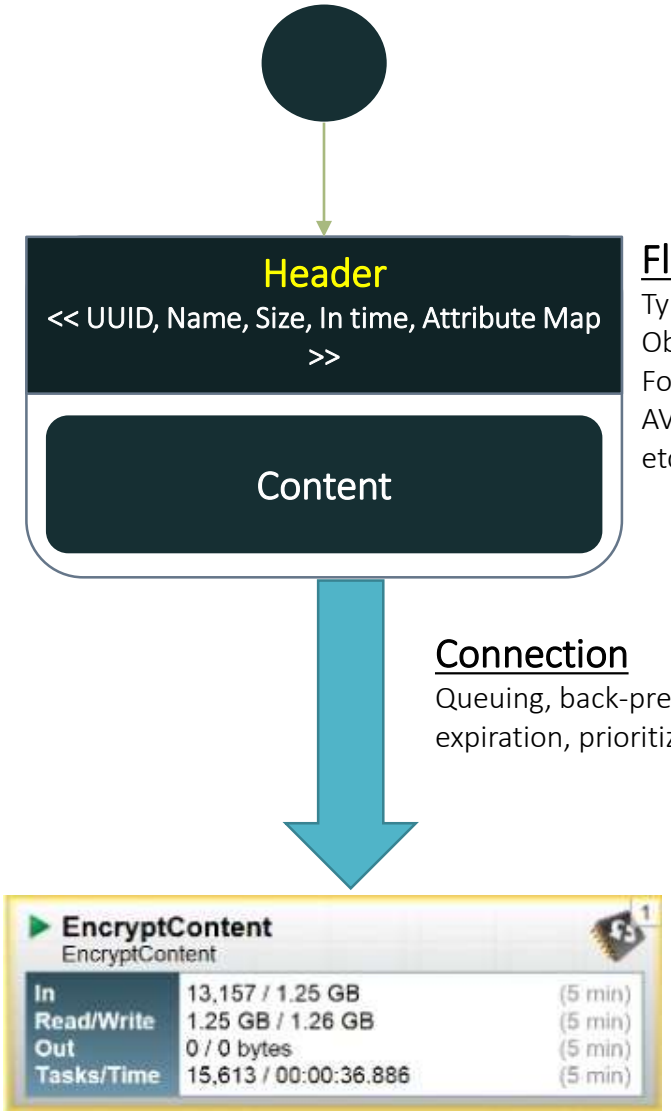
Process Group (Subnet)

Set of processors and their connections

Receive data via input ports, send data via output ports

Flow Controller (Scheduler)

Maintains the knowledge of how processes are connected and manages the threads and their allocation.



Flow File

Types (Events, Files, Objects, Messages etc.), Formats (JSON, XML, AVRO, Text, Proprietary etc.), Size (B to GBs)

Connection

Queuing, back-pressure, expiration, prioritization

Processor

Routing (Context/Content), Transformation (enrich, filter, convert, split, aggregate, custom), Mediation (push / pull), Scheduling

Apache NiFi is not

- NiFi is not a distributed computation Engine
  - An engine to do CEP (Complex event processing)
  - A computational framework to do distributed Joins or Rolling Window Aggregations the way Spark/Storm/Flink does.
  - Hence it's not based on Map Reduce / Spark or any other framework.

- NiFi doesn't have any dependency on any big data tool like Hadoop or zookeeper etc. All it needs is Java.

- It's not a full fledge ETL tool like Informatica / Pentaho / Talend / SSIS as of now. But it will be - eventually.

- It's not a long term Data storage tool. It only holds data temporarily for re-run / data provenance purposes.

- It's not a document indexer. It's indexing capabilities are only to help in troubleshooting / debugging.

03

Core Features

What are the core features and benefits of Apache NiFi?

Guaranteed Data Delivery

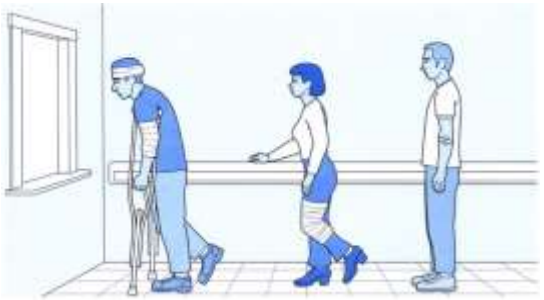
- Even at very high scale, delivery is guaranteed
- Persistent Write Ahead Log (Flow File Repository) and Data Partitioning (Content Repository) ensures this. They are together designed in a way that they allow:
  - Very high transaction rates
  - Effective load spreading
  - Copy-on-write scheme (for every change in data)
  - Pass-by-reference

Data Buffering w/ Back Pressure and Pressure Release

- Supports buffering of all queued data.
- Ability to back-pressure (Even if there is no load balancing, nodes can say “Back-Off” and other nodes in the pipeline pick up the slack.
- When backpressure is applied to a connection, it will cause the processor that is the source of the connection to stop being scheduled to run until the queue clears out. However, data will still queue up in that processor's incoming connections.

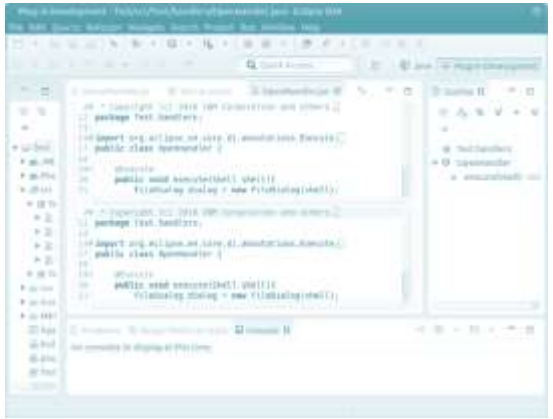
Prioritized Queuing

- NiFi allows the setting of one or more prioritization schemes for how data is retrieved from a queue.
- Oldest First, Newest first, Largest first, Smallest First, or custom scheme
- The default is oldest first



Designed for Extension

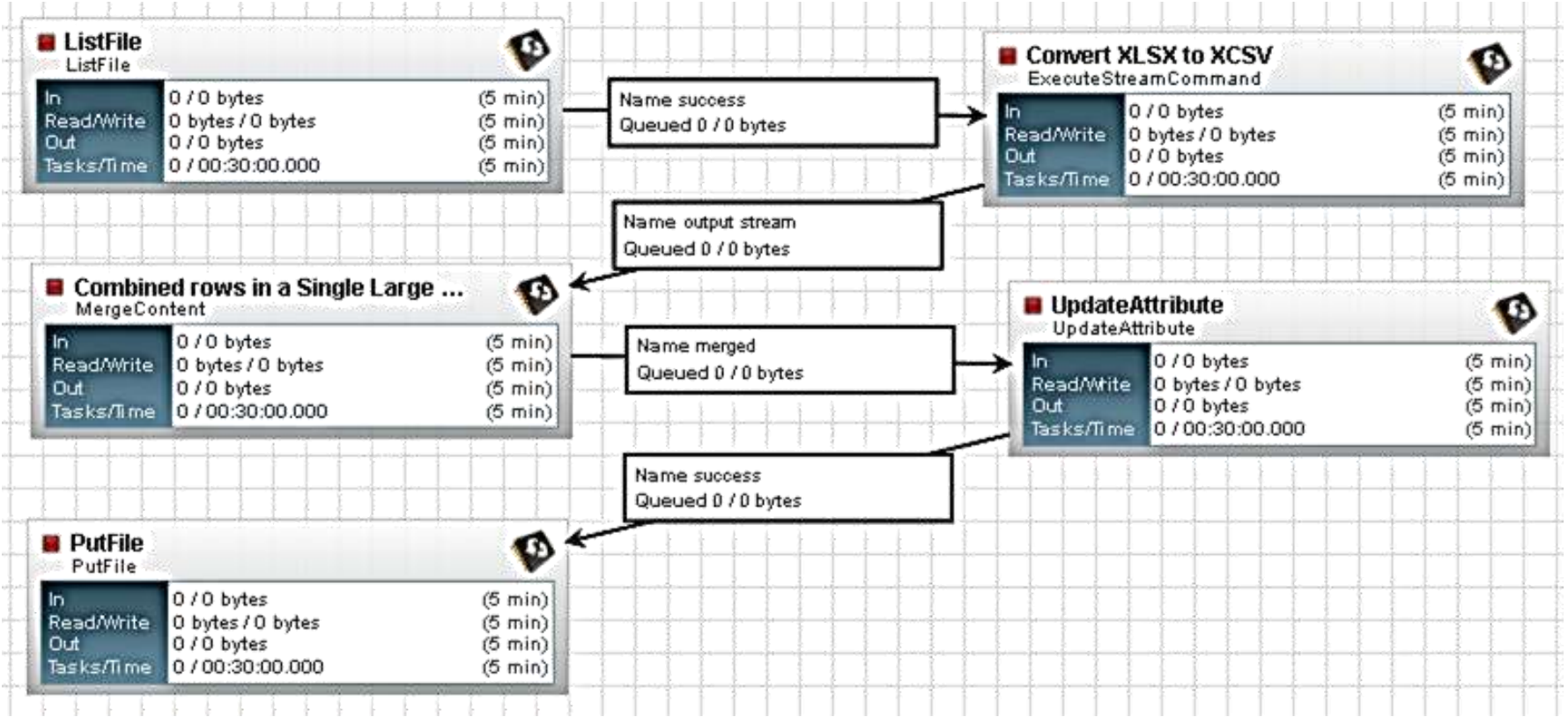
- NiFi by design is Highly Extensible.
- One can write *custom*:
  - ✓ Processor
  - ✓ Controller Service
  - ✓ Reporting Tasks
  - ✓ Prioritizer
  - ✓ User Interface



- These extensions are bundles in something called as NAR Files (*NiFi Archives).*

Visual Interface for Command and Control

- Drag and drop processors to build a flow
- Start, stop, and configure components in real time
- View errors and corresponding error messages
- View statistics and health of data flow
- Create templates of common processor & connections



Data Provenance (Not just Lineage)

NiFi Flow Data Provenance

Oldest event available: 07/29/2015 14:08:06 EDT

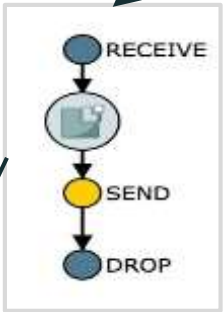
Filter | by component name

Displaying 1,000 of 1,000

Last updated: 21:12:00 EDT

Showing the most recent 1,000 of 62,293 events, please refine the search. Search

	Date/Time	Type	FlowFile Uuid	Size	Component Name	Component Type	
	07/29/2015 16:21:34.368 EDT	DROP	3b9f20bc-031e-4af8-ad8a-fedce...	158 bytes	PutSolrContentStream	PutSolrContentStream	
	07/29/2015 16:21:34.367 EDT	SEND	3b9f20bc-031e-4af8-ad8a-fedce...	158 bytes	PutSolrContentStream	PutSolrContentStream	
	07/29/2015 16:21:34.366 EDT	DROP	6f5036bc-1768-476d-9b6d-1f83...	2.15 KB	PutSolrContentStream	PutSolrContentStream	



- View attributes and content at given points in time (before and after each processor) !!!
- Records, indexes, and makes events available for display

Provenance Event

Details | Attributes | Content

Time	07/29/2015 16:21:34.367 EDT
Event Duration	00:00:00.001
Lineage Duration	00:00:00.117
Type	SEND
FlowFile Uuid	3b9f20bc-031e-4af8-ad8a-fedcef4e0099
File Size	158 bytes
Component Id	fa7b551f-c405-4fde-b004-0b0d69c03472
Component Name	PutSolrContentStream
Component Type	PutSolrContentStream
Transit Uri	solr://http://localhost:8984/solr/chronicle
Details	*No value set*

Benefits of Apache NiFi

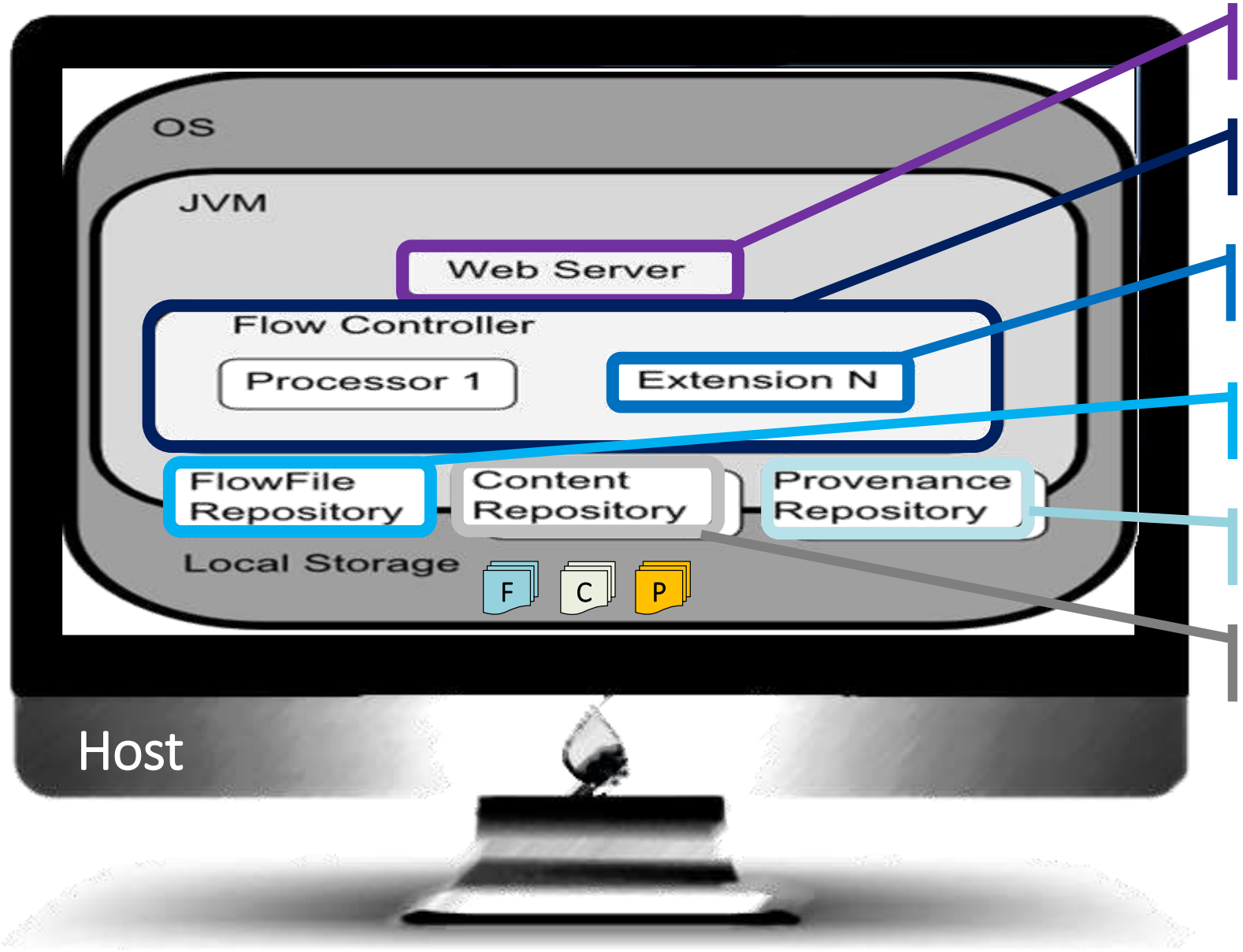
- Single data-source agnostic collection platform
- Intuitive, real-time visual user interface with drag-and-drop capabilities
- Powerful Data security capabilities from source to storage
- Highly granular data sharing policies
- Ability to react in real time by leveraging bi-directional data flows and prioritized data feeds
- Extremely scalable, extensible platform

04

Architecture

High level architecture (single machine), Primary components

Single Node



Host NiFi's HTTP-based command and control API.

Real Brain. Provide and manage threads. Scheduling.

Runs within JVM. Processor / Controller Service / Reporting Service / U I/ Prioritizer.

State of about a given FlowFile which is presently active in the flow. WAL.

All provenance event data is stored. Saved on FS. Indexed / Searchable.

Actual content bytes of a given FlowFile. Blocks of data in FS. More than 1 FS (partitions)

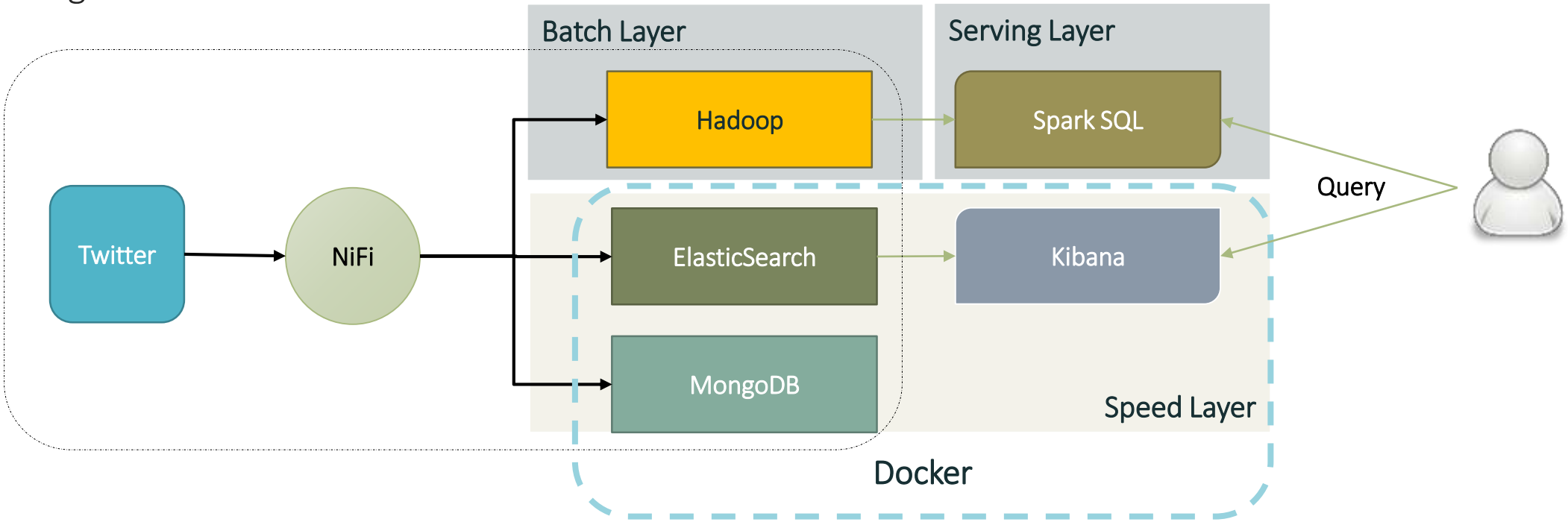
05

Demo

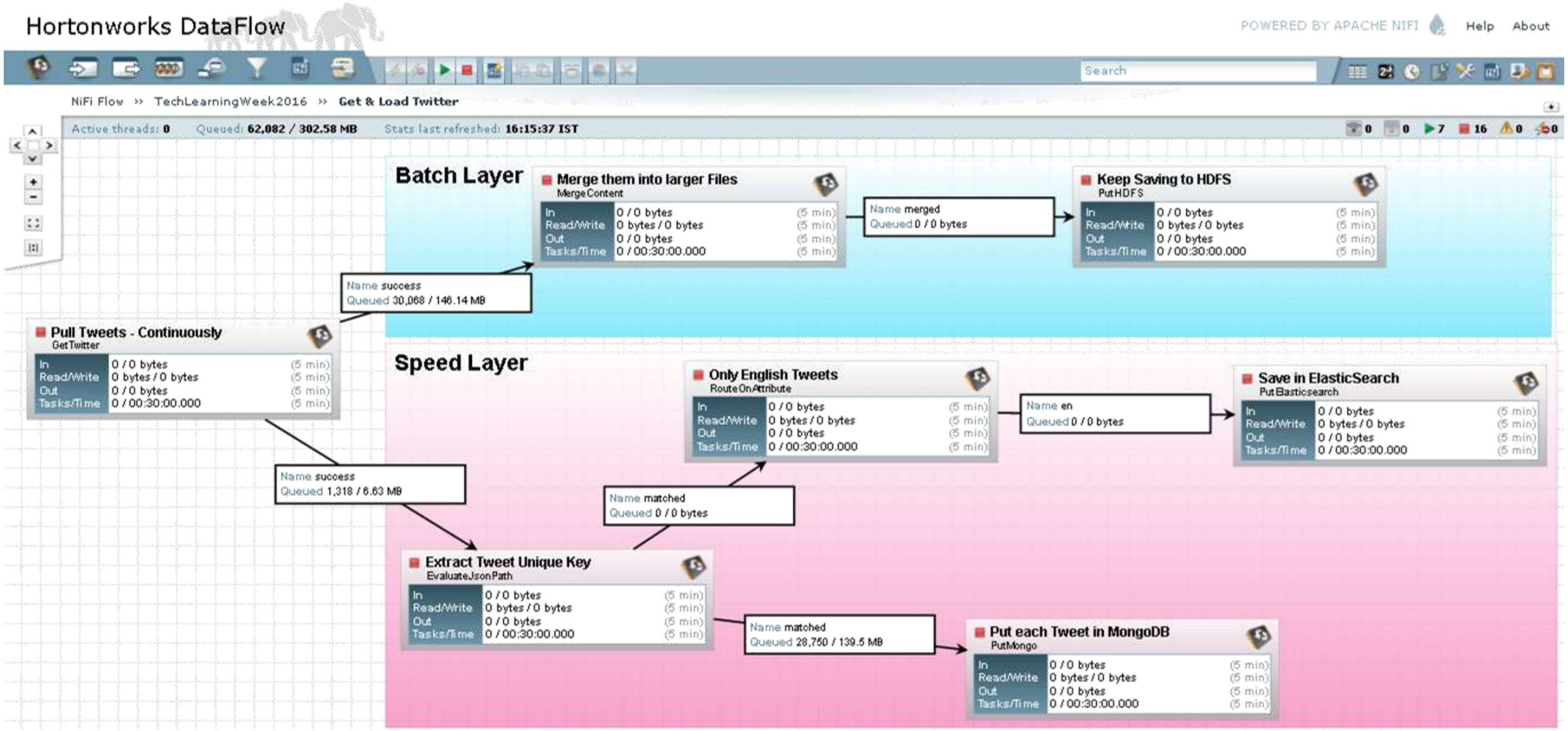
Simple λ Architecture

Demo (Simple λ Architecture using NiFi)

1. Start NiFi (or HDF). ElasticSearch, Kibana and MongoDB on Docker. Create HDFS destination table.
2. Explore NiFi UI
3. Pull data from twitter
4. Route & Deliver to ElasticSearch, Mongo and Hadoop in real time
5. Explore NiFi capabilities
6. Design Dashboard on real-time data



WYSIWYG…!!!


Hortonworks DataFlow
POWERED BY APACHE NIFI
Help
About
Search
NiFi Flow » TechLearningWeek2016 » Get & Load Twitter
Active threads: 0
Queued: 62,082 / 302.58 MB
Stats last refreshed: 16:15:37 IST
Batch Layer
Merge them into larger Files
MergeContent
In 0 / 0 bytes (5 min)
Read/Write 0 bytes / 0 bytes (5 min)
Out 0 / 0 bytes (5 min)
Tasks/Time 0 / 00:30:00.000 (5 min)
Name merged
Queued 0 / 0 bytes
Keep Saving to HDFS
PutHDFS
In 0 / 0 bytes (5 min)
Read/Write 0 bytes / 0 bytes (5 min)
Out 0 / 0 bytes (5 min)
Tasks/Time 0 / 00:30:00.000 (5 min)
Name success
Queued 30,068 / 146.14 MB
Pull Tweets - Continuously
GetTwitter
In 0 / 0 bytes (5 min)
Read/Write 0 bytes / 0 bytes (5 min)
Out 0 / 0 bytes (5 min)
Tasks/Time 0 / 00:30:00.000 (5 min)
Speed Layer
Only English Tweets
RouteOnAttribute
In 0 / 0 bytes (5 min)
Read/Write 0 bytes / 0 bytes (5 min)
Out 0 / 0 bytes (5 min)
Tasks/Time 0 / 00:30:00.000 (5 min)
Name en
Queued 0 / 0 bytes
Save in ElasticSearch
PutElasticsearch
In 0 / 0 bytes (5 min)
Read/Write 0 bytes / 0 bytes (5 min)
Out 0 / 0 bytes (5 min)
Tasks/Time 0 / 00:30:00.000 (5 min)
Name success
Queued 1,318 / 6.63 MB
Name matched
Queued 0 / 0 bytes
Extract Tweet Unique Key
EvaluateJsonPath
In 0 / 0 bytes (5 min)
Read/Write 0 bytes / 0 bytes (5 min)
Out 0 / 0 bytes (5 min)
Tasks/Time 0 / 00:30:00.000 (5 min)
Name matched
Queued 28,750 / 139.5 MB
Put each Tweet in MongoDB
PutMongo
In 0 / 0 bytes (5 min)
Read/Write 0 bytes / 0 bytes (5 min)
Out 0 / 0 bytes (5 min)
Tasks/Time 0 / 00:30:00.000 (5 min)

06

Use Cases

Some Scenarios

Some Use Cases

Building Ingestion and Delivery layers in IoT Solutions

Ingestion tier in Lambda Architecture (for feeding both speed and batch layers)

Ingestion tier in Data Lake Architectures

Cross Geography Data Replication in a secure manner

Integrating on premise system to on cloud system (Hybrid Cloud Architecture)

Simplifying existing Big Data architectures which are currently using Flume, Kafka, Logstash, Scribe etc. or custom connectors.

Developing Edge nodes for Trade repositories.

Enterprise Data Integration platform

And many more…

07

Q & A

Reference

https://nifi.apache.org/

http://hortonworks.com/products/data-center/hdf/

https://github.com/apache/nifi

https://twitter.com/apachenifi

Thank You

@manishpedia

https://in.linkedin.com/in/manishgforce