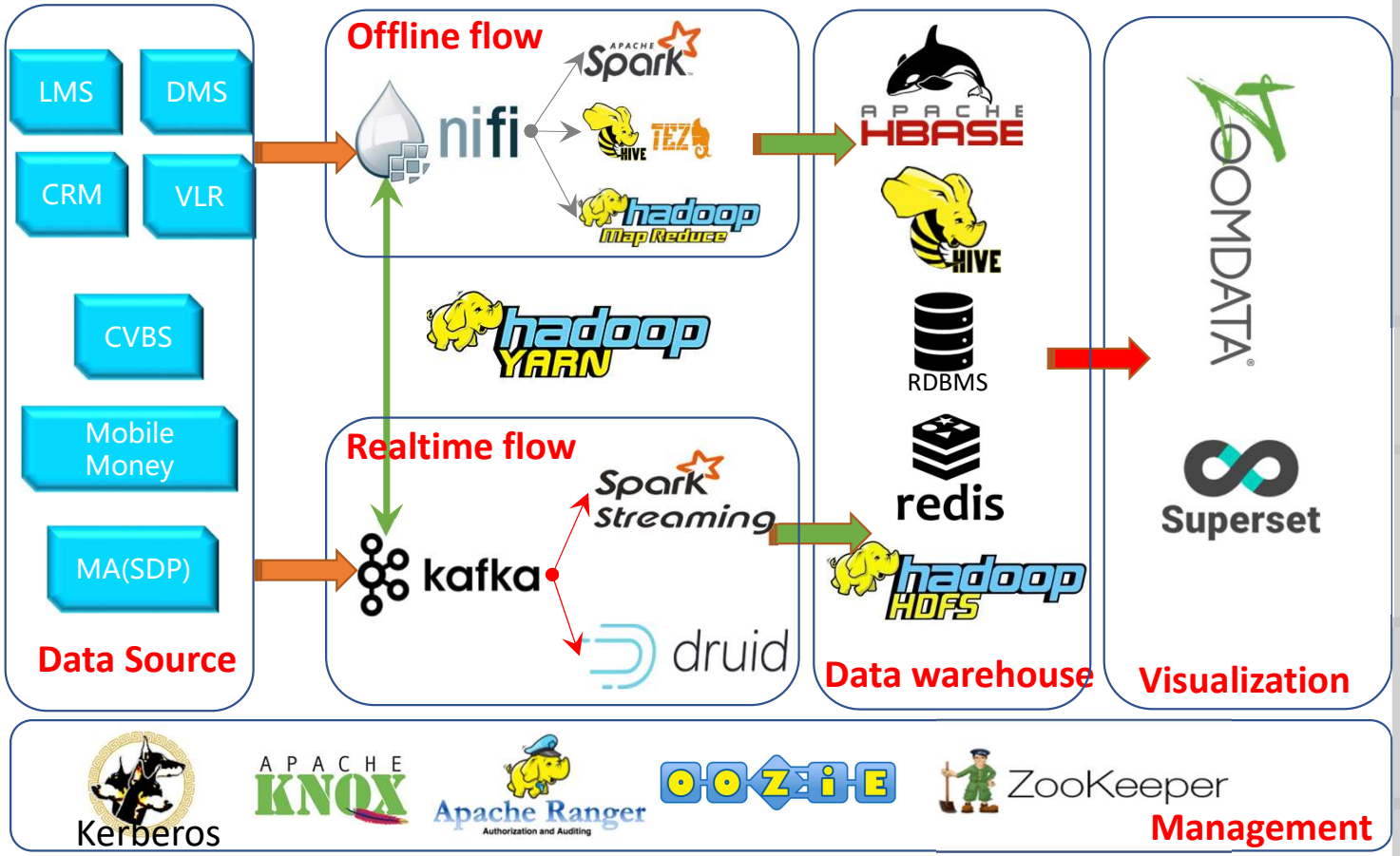
Hadoop Architecture

Aska Dynamics

Lambda Architecture For Hadoop



Data Source

All the valuable data from all MPT system

Offline Batch

Use NiFi as data ingestion tool to collect all the data from all MPT system, transfer to batch system like spark, hive/tez, MapReduce for offline calculation

Online Realtime

Use Kafka as Realtime streaming data ingestion tool to Realtime collect data from each system and pass to Spark Streaming or Druid for Realtime processing

Resource management

Use Apache Yarn as whole cluster Resource Management tool

Data warehouse

For Data Warehouse use Hbase/Redis as some on line Realtime services, use Hive/RDBMS as the offline services

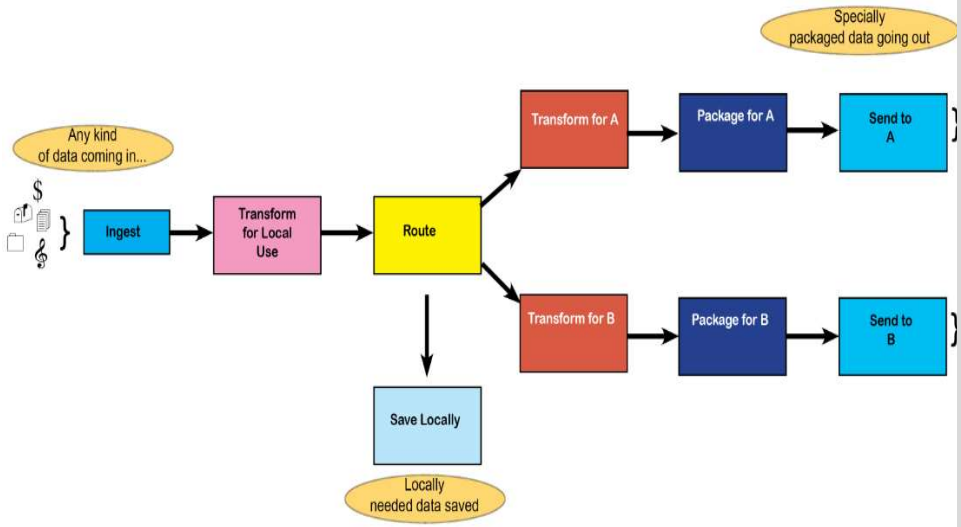
Data visualization

All Use ZoomData as the data visualization tool, Superset will help druid to do the data visualization

Data Security

Use Kerberos to keep all cluster save, Ranger to control all the user permission, setup SSL to encrypt the communication

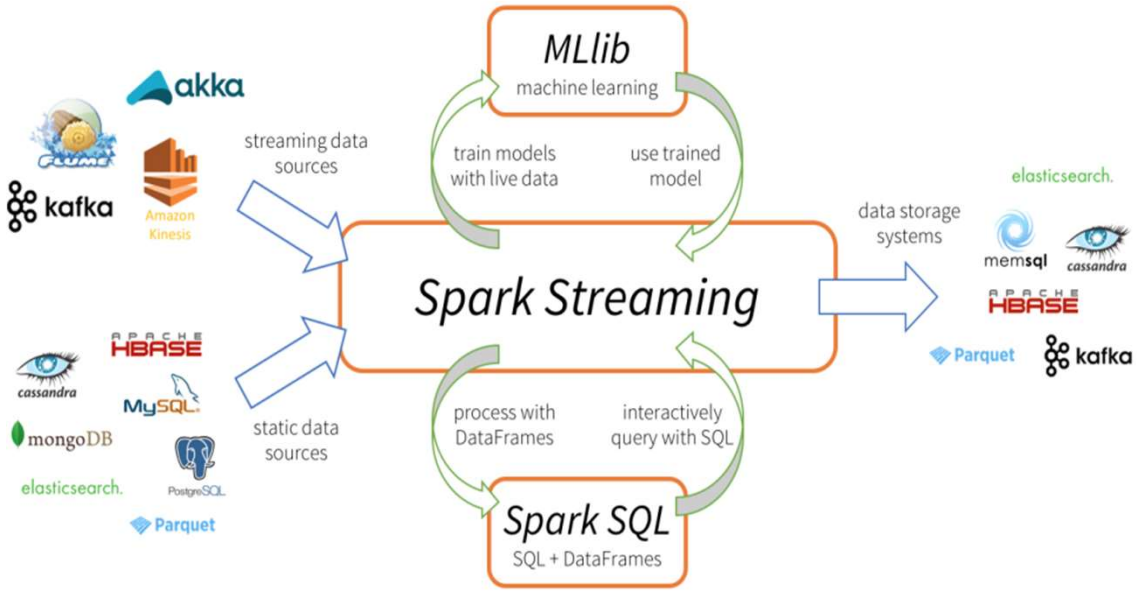
NiFi



- Apache NiFi supports powerful and scalable directed graphs of data routing, transformation, and system mediation logic.
- Features include:
  - **Web-based user interface** - covering design, control, feedback, and monitoring.
  - **Highly Configurable** - enables a balance between loss tolerance and guaranteed delivery, and low latency vs high throughput. Enables dynamic prioritization of flows, modification of flows at runtime, and back pressure thresholds, which specify amount of data that may exist in the queue, to avoid overrunning the system with data.
  - **Data Provenance** - enables tracking data flows from beginning to end.
  - **Extensible** - enables users to build their own processors and more. Enables rapid development and effective testing. Secure - supports SSL, SSH, HTTPS, encrypted content, and more. Provides multi-tenant authorization and internal policy management.

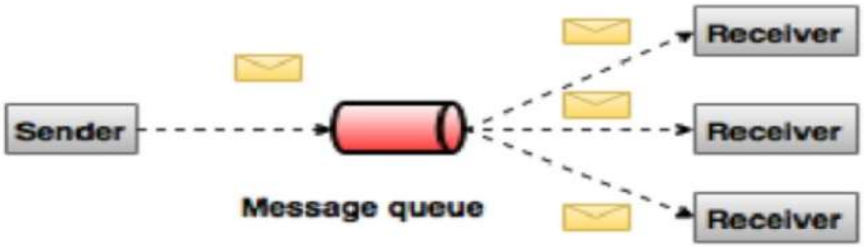
- Detailed document is available in the link below

Spark Streaming



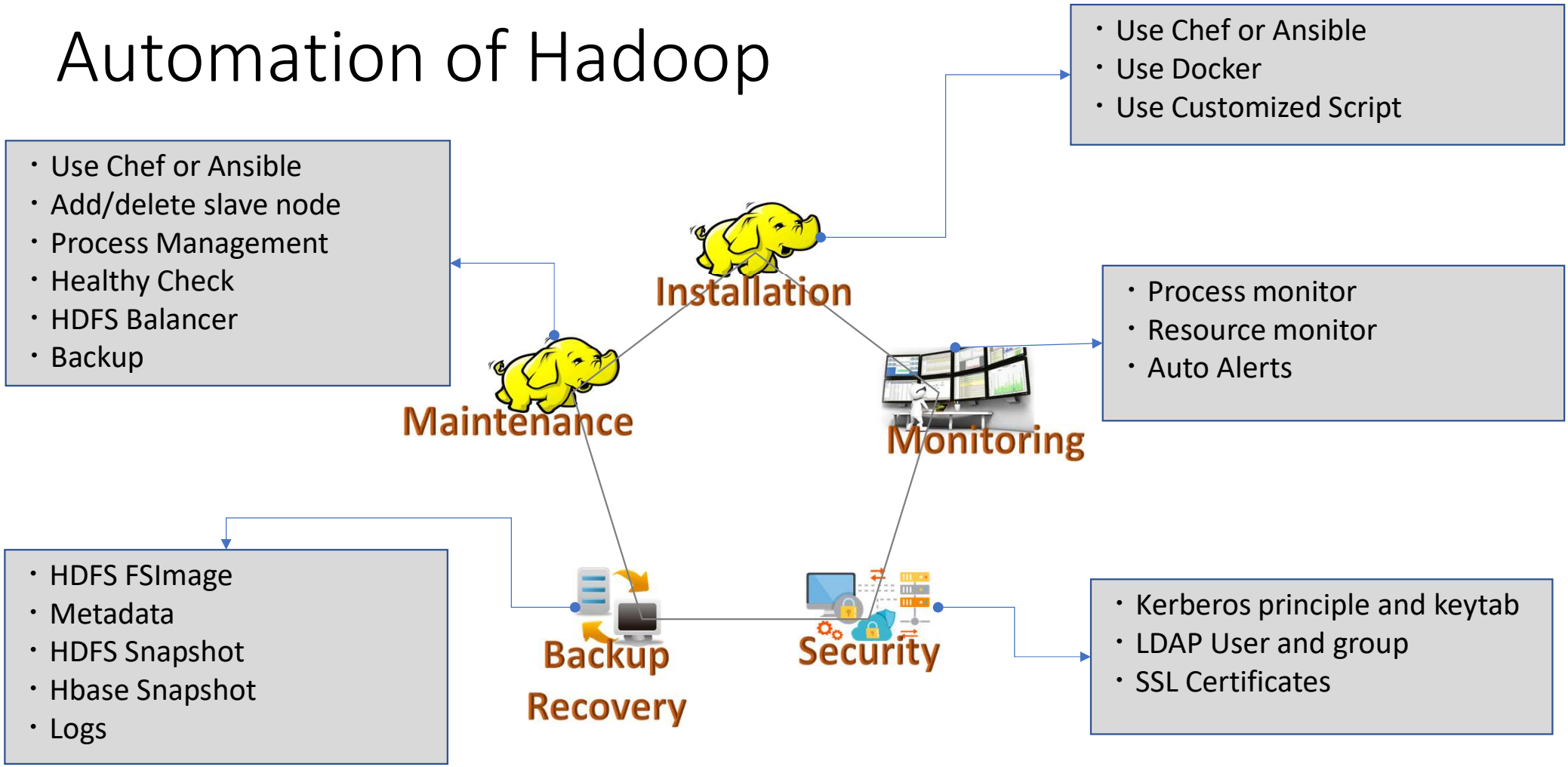
- Spark Streaming is different from other systems that either have a processing engine designed only for streaming or have similar batch and streaming APIs but compile internally to different engines.
- Spark's single execution engine and unified programming model for batch and streaming lead to some unique benefits over other traditional streaming systems.
- Four major aspects are:
  - Fast recovery from failures and stragglers
  - Better load balancing and resource usage
  - Combining of streaming data with static datasets and interactive queries
  - Native integration with advanced processing libraries (SQL, machine learning, graph processing)
- Only for real time data pull and processing.

Kafka

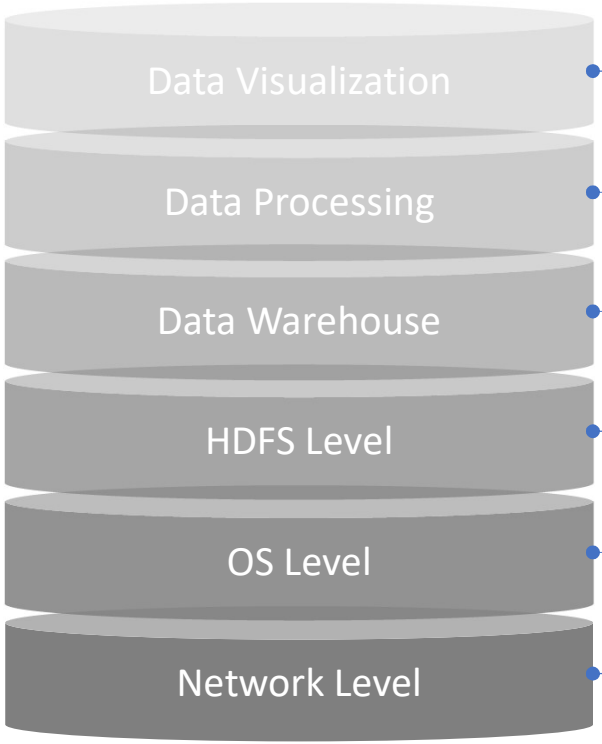


- Apache Kafka is a distributed publish-subscribe messaging system.
  - messages are persisted in a topic. Unlike point-to-point system, consumers can subscribe to one or more topic and consume all the messages in that topic. In the Publish-Subscribe system, message producers are called publishers and message consumers are called subscribers.
- Apache Kafka is a distributed publish-subscribe messaging system and a robust queue that can handle a high volume of data and enables you to pass messages from one end-point to another. Kafka is suitable for both offline and online message consumption. Kafka messages are persisted on the disk and replicated within the cluster to prevent data loss. Kafka is built on top of the Zookeeper synchronization service. It integrates very well with Apache Storm and Spark for real-time streaming data analysis.
- **Benefits**
  - **Reliability** − Kafka is distributed, partitioned, replicated and fault tolerance.
  - **Scalability** − Kafka messaging system scales easily without down time.
  - **Performance** − Kafka has high throughput for both publishing and subscribing messages. It maintains stable performance even many TB of messages are stored.
  - **Durability** − Kafka uses “Distributed commit log” which means messages persists on disk as fast as possible, hence it is durable.

Automation of Hadoop



Performance



- Do data management
- Validate with business user
- Keep Dashboard simple and digestible
- Use suitable filtering
- Choose suitable connection

- Choose suitable processing Engine
- Use Tungsten
- Understand your data
- Chose suitable Resource Manager
- Efficiently use data cache

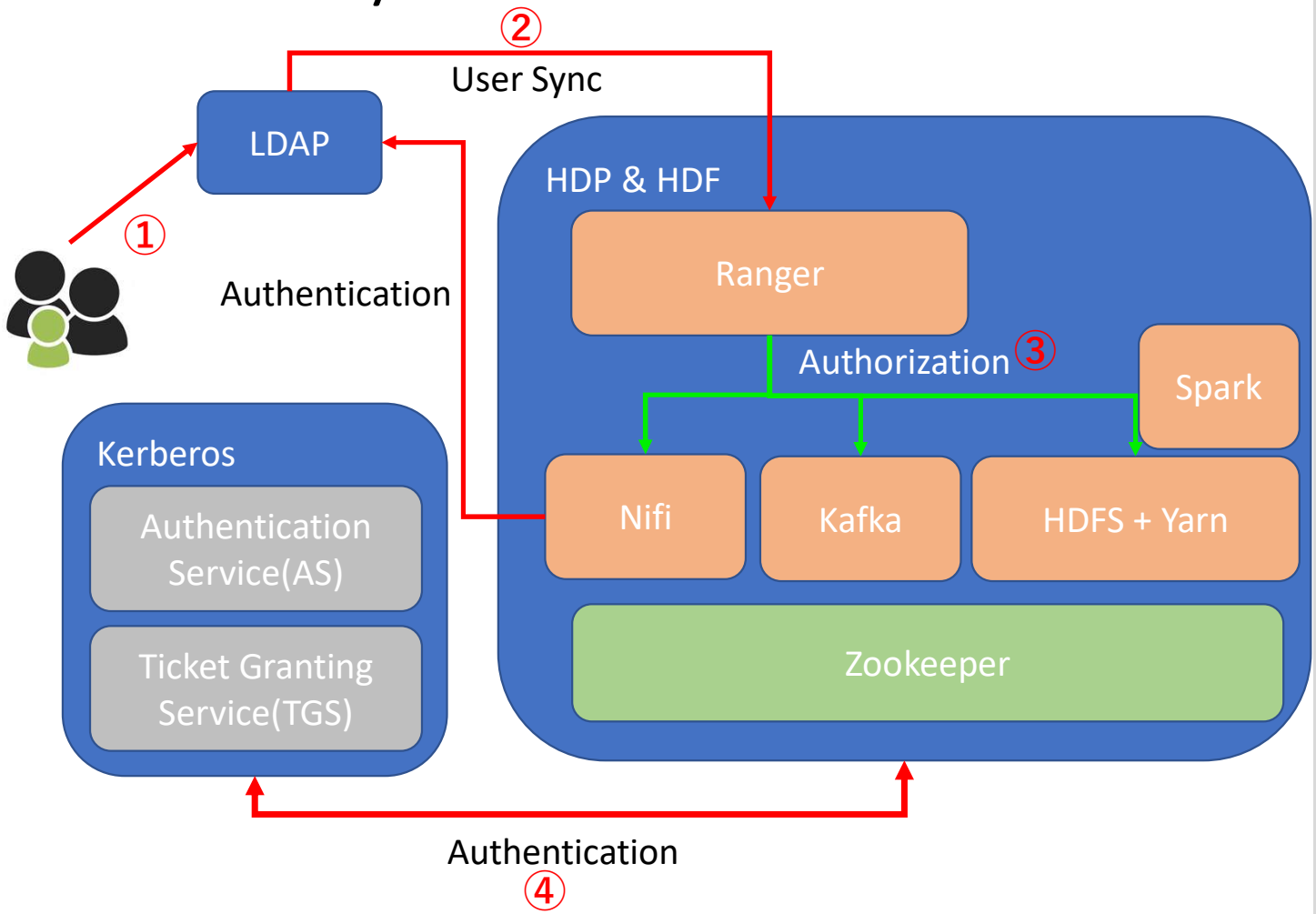
- Use neutral data model
- Use View
- Use Indexes
- Enforce prioritization
- Improve ETL Performance
- Capacity Plan and Management

- HDFS short circuit reads
- DataNode JVM
- HDFS block size
- HDFS File sizes
- Stale DataNodes
- mount options for data disks

- SELinux
- Openfiles
- Tune SSD Configurations
- Virtual Memory Usage
- Disable Host Swappiness
- Disable Transparent Huge Pages (THP)

- Isolated network
- Static IPs
- Reverse DNS setup
- 2 x 1 Gb & 1 x 10 Gb Ethernet
- Multiple RAC design

Security Architecture



① Register user to LDAP system

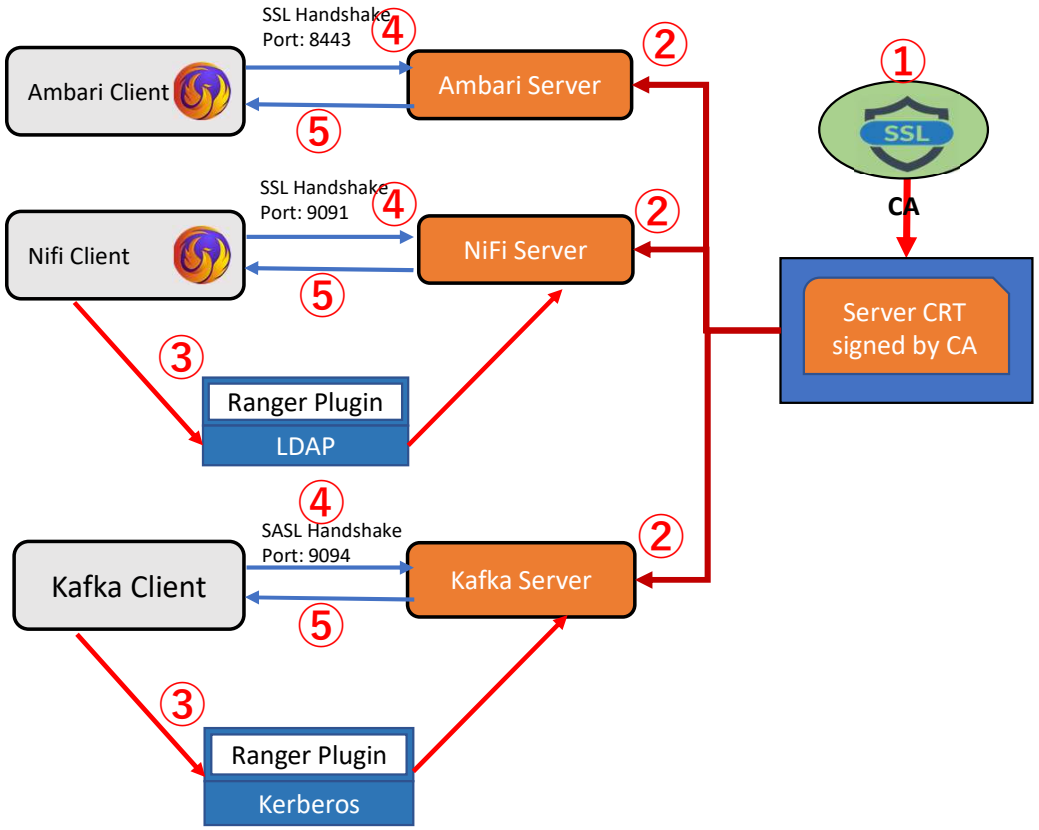
② Sync use from LDAP to Ranger

③ Create Ranger Policies to control Nifi, Kafka, HDFS, Yarn permission

④ Kerberos control the whole cluster security

Ambari/NIFI/Kafka SSL Security

- authentication and encryption



*SSL-> Encryption
*LDAP->User Authentication

① Apply Alliance SSL certificate

② Get Alliance certificate and install in each service server (Ambari, Nifi, Kafka)

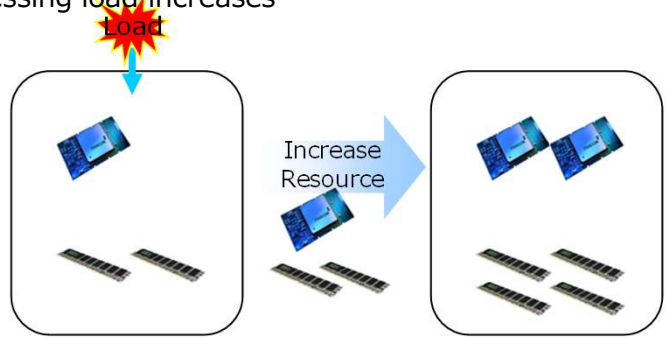
③ User Authentication By LDAP or Kerberos

④ SSL handshake with each server

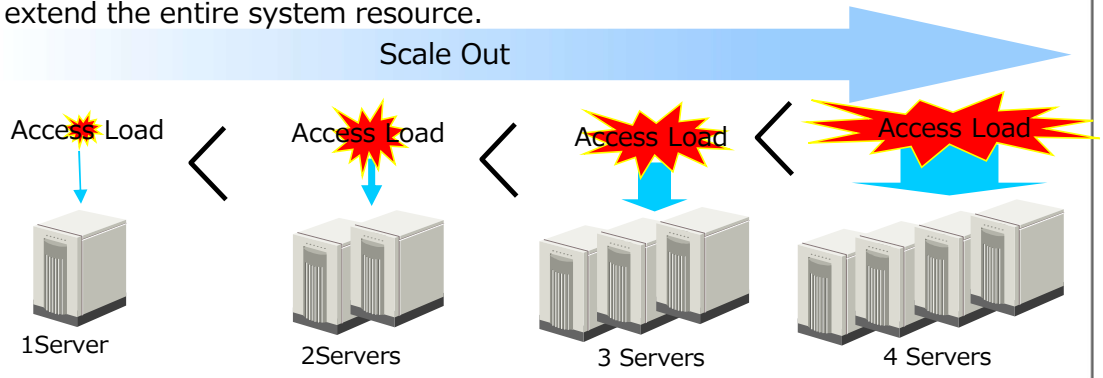
⑤ Communicate with encrypt pipe line

Hadoop Expansion

Scale Up
Expand the resource by increasing the Server's CPU, memory, etc.,
as the processing load increases



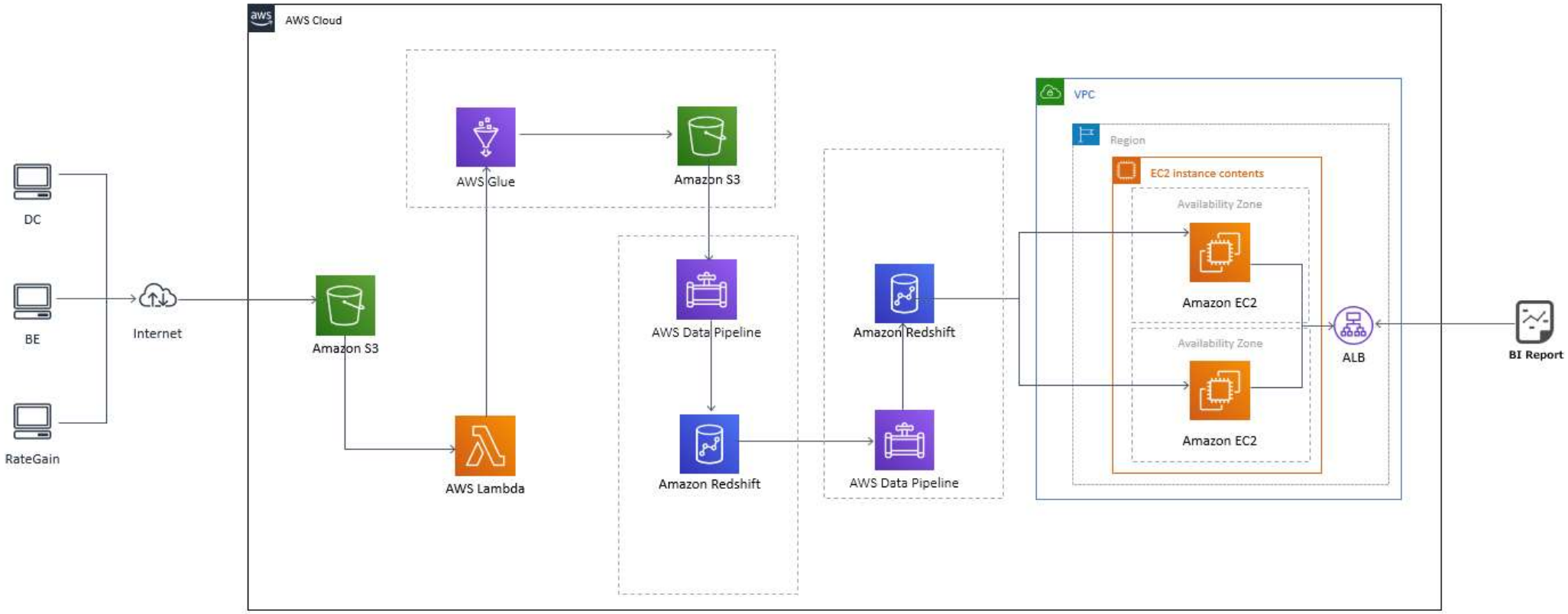
Scale Out
In accordance with the increase in the access load, add more servers to extend the entire system resource.



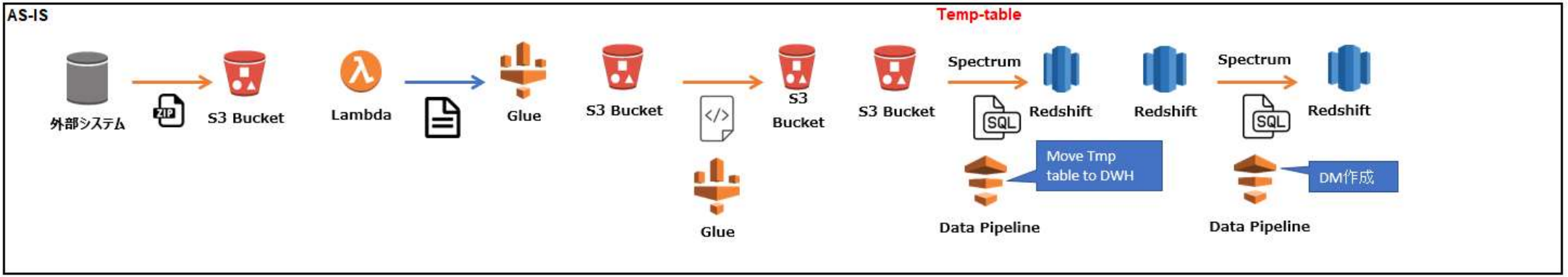
Target Server	Expand Target	Strategy
Ambari Node	CPU	Usage:70%+ Scale up
	Memory	Usage:70%+ Scale up
	Disk	Usage:70%+ Scale up
Master Node	CPU	Usage:70%+ Scale up
	Memory	Heap:70%+ Scale up
	Disk	Usage:70%+ Scale up
Slave Node	CPU	Usage:70%+ Scale out
	Memory	Usage:70%+ Scale out
	Disk	HDFS:70%+ Scale out

DWH On AWS

AS-IS Architecture

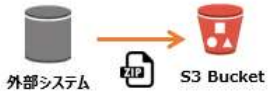
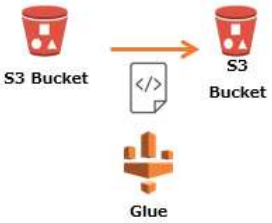
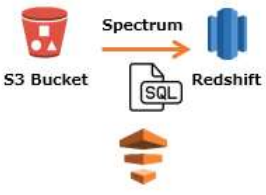
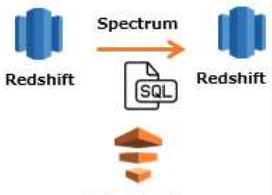
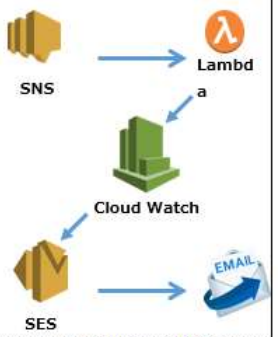


アーキテクチャ比較

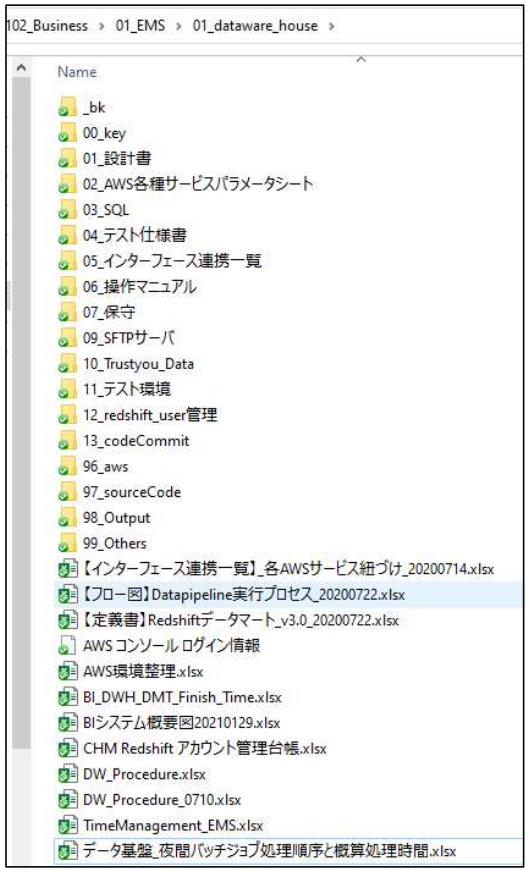

AS-IS
Temp-table
外部システム
S3 Bucket
Lambda
Glue
S3 Bucket
S3 Bucket
Glue
S3 Bucket
Spectrum
Redshift
Redshift
Spectrum
Redshift
SQL
SQL
Move Tmp table to DWH
DM作成
Data Pipeline
Data Pipeline


新構成図
DWH
同上
S3 Bucket
外部テーブル
Redshift
Redshift
Spectrum
Aurora
SQL
DM作成
Data Pipeline

処理流れ

Step	1	2	3	4	5	6	7
	データウェアハウスにデータを保存するプロセス				外部システムで使うためのデータを作成してデータマートへ保存、データを外部へ提供するプロセス		
やっていること	ソースシステム、予約システムなどから約16種類のファイルがS3の各フォルダへ入れられる（各ファイルは圧縮されている）	S3に新しいファイルが入ったことをLambdaが感知し、圧縮ファイルを解凍して、ファイル名ごとにGlueのJobを呼び出す	Lamdaが呼び出したGlueのJobには、解凍されたCSVファイルをRedshiftが読み込めるparquet形式のファイルに変換して、S3の指定フォルダへ保存する	Glueが作成したparquet形式のファイルを、Data PipelineがSpectrumを通してRedshiftのテーブルへ書き込む（各テーブルごとに使用するSQL文はS3に保管してある）	Redshiftに保存されたテーブルをData Pipelineが合わせて、加工して、計算して、データマート用のテーブルに書き込む（各テーブルごとに使用するSQL文はS3に保管してある）	BIレポートシステムが稼働しているEC2からRedshiftのデータマート用テーブルを参照してレポートを作成する	各処理プロセスが正常終了しなかった場合、SNSがそれを感知して、Lambdaを起動、Cloud Watchに定義してある基準毎にSESを使って管理者へ通知メールを送信する
ツール							
動作頻度	日次バッチ処理で自動アップロードされる	Lambdaがアップロードファイルを感知して、自動実行される	LambdaがGlueを呼び出して、自動実行される	日次バッチ処理で自動実行される	日次バッチ処理で自動実行される	日次バッチ処理で自動更新される	エラー発生の都度、自動送信される
備考	S3のフォルダ構造は、S3タブのシートを参照。 どんな種類のファイルが送られてくるのか？は、現担当業者から資料が送られてくる予定。	Lambdaの内容は、Lambdaタブのシートを参照。	GlueのJobは、GlueJobタブのシートを参照。	Data Pipelineの内容は、Data Pipelineタブのシートを参照。			Cloud Watchの動作は、Cloud Watchタブのシートを参照。 監視する内容は、監視対象メトリクス一覧タブのシートを参照。

DWHドキュメントセット



- 01_設計書：Redshiftテーブル定義書など
- 02_AWS各種サービスパラメータシート：各AWSサービスの設計、パラメータシート
- 03_SQL：DDLなど
- 保守または操作マニュアルなど

お客様情報がありますので、共有するのは難しい。
申し訳ございません。

Thank you very much!