

Data Engineering Project

Crypto currency's Data Pipeline



วัตถุประสงค์ของโครงการ

- วางโครงสร้าง Data pipeline เพื่อสร้าง Automated Trading Bot บน Cryptocurrency ที่สามารถส่งสัญญาณคำสั่งซื้อ-ขายที่แม่นยำ และน่าเชื่อถือให้แก่นักลงทุนคริปโตโดยใช้ความรู้ทางทฤษฎีจากกระบวนการวิชา Data Engineering



การทำ Automate Trading Bot ต้องตอบคำถามอะไรบ้าง

1. จะใช้ภาษาอะไรในการทำ Data Pipeline?
2. ข้อมูลราคาของ Cryptocurrencyจะนำมาจากแหล่งไหน?
3. จะใช้ตัวช่วยอะไรในการจัดการและประมวลผลข้อมูล?
4. การจัดเก็บข้อมูลอย่างไร?
5. กลยุทธ์ของการซื้อ-ขายคืออะไร?
6. จะส่งคำสั่งซื้อ-ขายอย่างไร?
7. เครื่องมือที่ช่วยในการจัดการ Data Pipeline คืออะไร?
8. ตัวช่วยประมวลผลการดำเนินการควรเป็นอะไร ระหว่าง On-Cloud vs On-Premise?
9. ทดสอบอย่างไร?

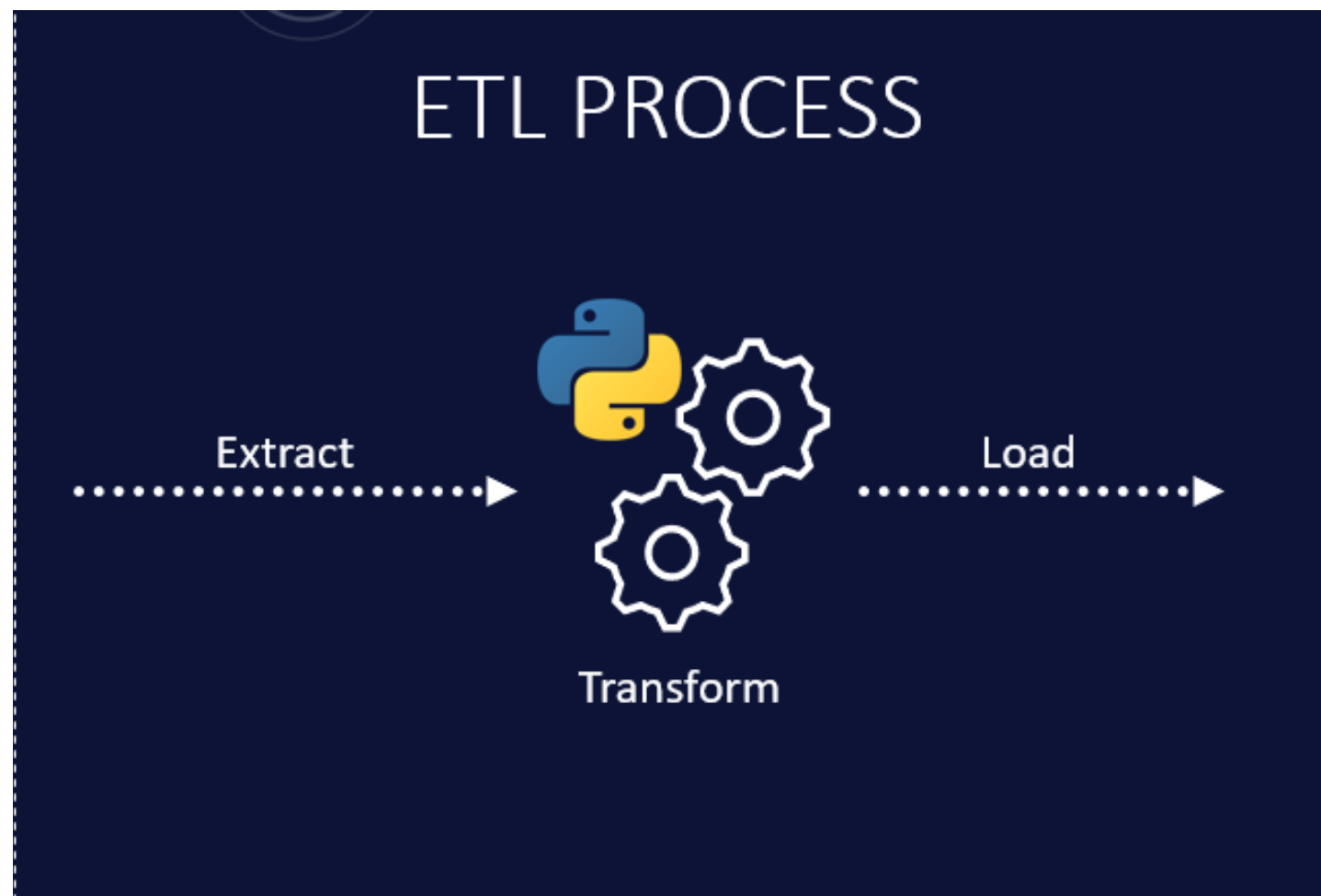




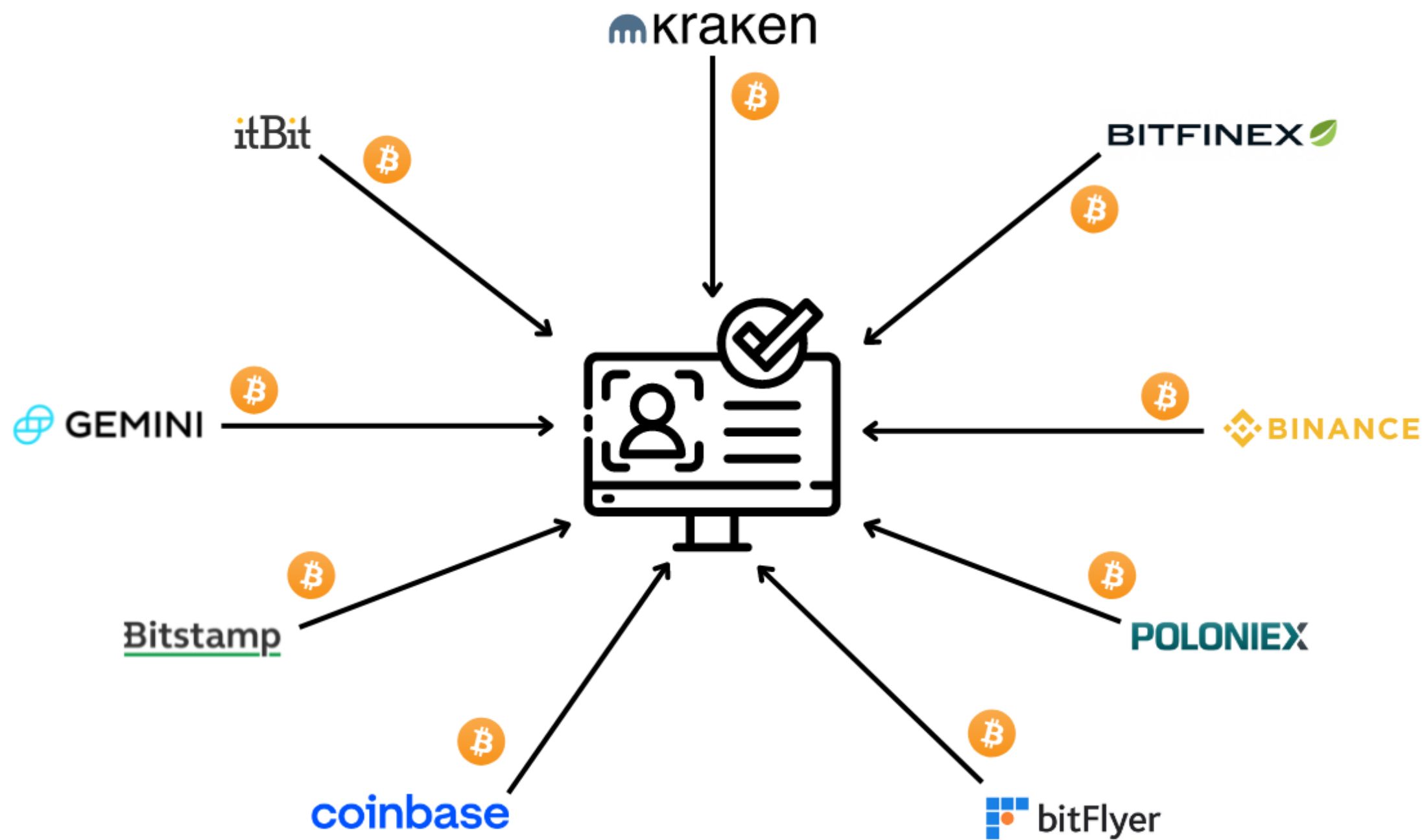
ภาษาที่เลือกใช้

เหตุผลที่เลือกภาษา Python มีดังนี้

- High level Programming ที่นิยมใช้ในงานหลายด้าน อาทิ สร้าง Automation Task
- สามารถโหลด, นำส่ง และ Clean ข้อมูลได้อย่างมีประสิทธิภาพ เหมาะแก่การทำ Big Data Processing
- ทำงานร่วมกับ Tech Stak ตัวอื่นได้ดี อาทิ Apache Spark
- เรียนรู้ได้ง่าย, ทีมพัฒนาที่รับงานต่อ ดูแลได้สะดวก



แหล่งข้อมูลที่นำมาใช้



เนื่องจากข้อมูลของCryptocurrency มีการเก็บข้อมูลแบบกระจาย และมีตลาด Exchange ที่มากมาย บางที่เชื่อมข้อมูลได้ แต่บางที่อาจทำไม่ได้

ดังนั้นทางทีมจึงพยายามคัดเลือก ผู้ให้บริการข้อมูล Cryptocurrency ที่ครอบคลุมปริมาณการซื้อขายทั่วโลก และต้องมีความเสถียรในการรับ-ส่ง ข้อมูลด้วย



แหล่งข้อมูลที่นำมาใช้



[ข้อดี]

- มีประวัติข้อมูล Crypto ที่ครอบคลุมมากที่สุด
- Support การแลกเปลี่ยนหลากหลายสกุลเงิน
- ข้อมูล Real-time
- การันตีการทำงานต่อเนื่องมีและ SLA
- ครอบคลุมตลาด 832 ตลาด ~ 80% ของการซื้อขายของโลก

[ข้อเสีย]

- เสียค่าใช้จ่ายการสมัครใช้บริการ
- ไม่เหมาะกับการใช้เรียกข้อมูลผ่านแบบ Batch



แหล่งข้อมูลที่น่ามาใช้



[ข้อดี]

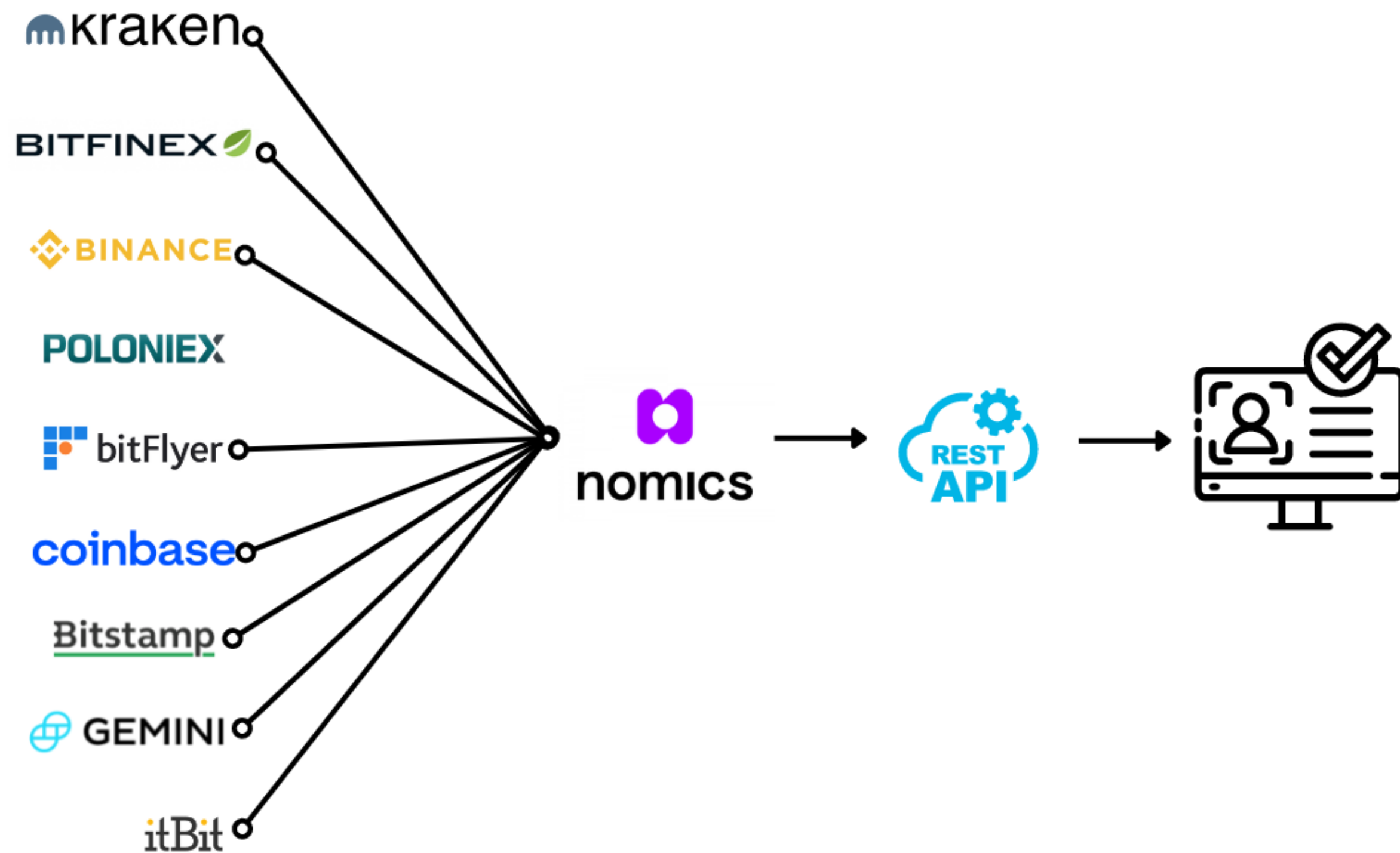
- มีบริการดึงข้อมูลแบบ Batch file
- มี historical data รายชั่วโมงให้ย้อนดูได้เป็นปี
- ค่าบริการค่อนข้างดีกว่าเจ้าอื่น
- มีข้อมูลข่าวรายวันให้ด้วย
- ครอบคลุมตลาด 311 แห่ง

[ข้อเสีย]

- ถ้าต้องการข้อมูลจำเพาะต้องจ่ายเงินเพิ่ม
- ไม่มีระบบ Customer Support



แหล่งข้อมูลที่น่ามาใช้



[ตัวอย่าง] การขอข้อมูลที่ใช้ในโครงการนี้

GET /currencies/ticker

Request samples

Shell

JavaScript

NodeJS

Ruby

Python

Copy

```
curl "https://api.nomics.com/v1/currencies/ticker?key=your-key-here&ids=BTC,ETH,XRP&interval=1d,30d&con"
```

[ตัวอย่าง] การขอข้อมูลที่ใช้ในโครงการนี้

Response samples

200

Content type
application/json

Copy Expand all Collapse all

```
"currency": "BTC",
"id": "BTC",
"status": "active",
"price": "8451.36516421",
"price_date": "2019-06-14T00:00:00Z",
"price_timestamp": "2019-06-14T12:35:00Z",
"symbol": "BTC",
"circulating_supply": "17758462",
"max_supply": "21000000",
"name": "Bitcoin",
"logo_url": "https://s3.us-east-2.amazonaws.com/nomics-api/static/images/currencies/btc.svg",
"market_cap": "150083247116.70",
"market_cap_dominance": "0.4080",
"transparent_market_cap": "150003247116.70",
"num_exchanges": "357",
"num_pairs": "42118",
"num_pairs_unmapped": "4591",
"first_candle": "2011-08-18T00:00:00Z",
"first_trade": "2011-08-18T00:00:00Z",
"first_order_book": "2017-01-06T00:00:00Z",
"first_priced_at": "2017-08-18T18:22:19Z",
"rank": "1",
"rank_delta": "0",
"high": "19404.81116899",
"high_timestamp": "2017-12-16",
- "1d": {
  "price_change": "269.75208019",
  "price_change_pct": "0.03297053",
  "volume": "1110989572.04",
```

```
"volume_change": "-24130098.49",
"volume_change_pct": "-0.02",
"market_cap_change": "4805518049.63",
"market_cap_change_pct": "0.03",
"transparent_market_cap_change": "4800518049.00",
"transparent_market_cap_change_pct": "0.0430",
- "volume_transparency": [
  - {
    "grade": "A",
    "volume": "2144455081.37",
    "volume_change": "-235524941.08",
    "volume_change_pct": "-0.10"
  },
  - {
    "grade": "B",
    "volume": "15856762.85",
    "volume_change": "-6854329.88",
    "volume_change_pct": "-0.30"
  }
]
```

ขนาดของข้อมูล

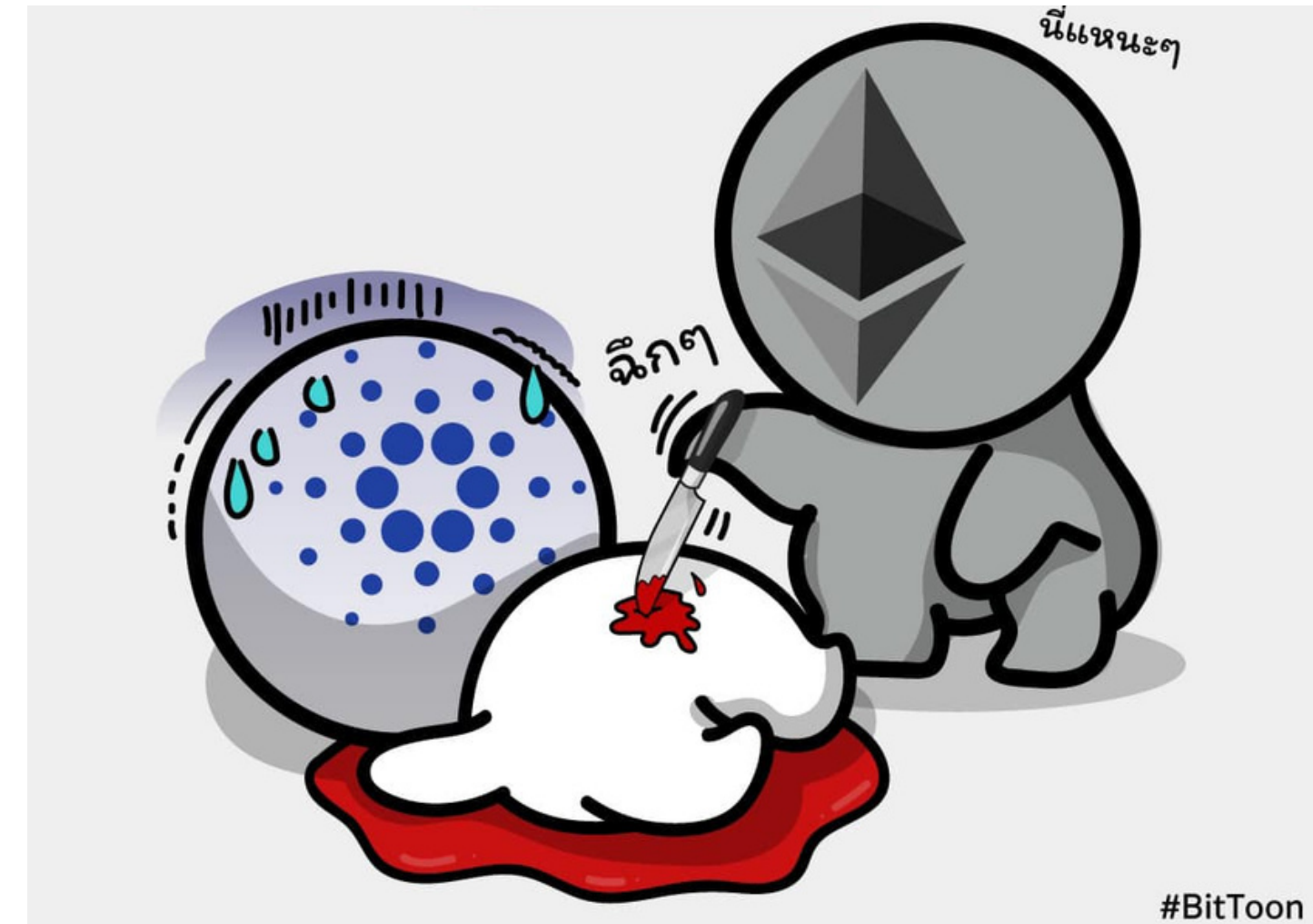
Raw Response

- JSON format
- Size ~ 226 KB

Estimate (Yearly)

- Trading Daily ~ 82 MB
- Trading Hourly ~ 1 GB
- Trading 5 Mins ~ 11.5 GB
- Trading 30 seconds ~ 232 GB

*** The highest frequency refresh rate of Nomics = 30 seconds



#BitToon

ตัวช่วยในการจัดการและประมวลผลข้อมูล



HADOOP เป็นระบบจัดเก็บข้อมูลที่สามารถจัดการกับความหลากหลายของข้อมูลได้ เช่น ข้อความ, รูปภาพ, วิดีโอ และ เสียง เป็นต้น

ซึ่งแต่ละชนิดนั้นจะมีโครงสร้างข้อมูล (Data Structure) ที่แตกต่างกัน อีกทั้งยังมีปริมาณที่เยอะขึ้น ทำให้เครื่องคอมพิวเตอร์เครื่องเดียวไม่สามารถทำงานได้อีกต่อไป

ตัวช่วยในการจัดการและประมวลผลข้อมูล

HADOOP

MapReduce

Others

YARN

HDFS

HADOOP ประกอบด้วย 2 ส่วนหลัก ดังนี้

1. HDFS (Hadoop Distributed File System)

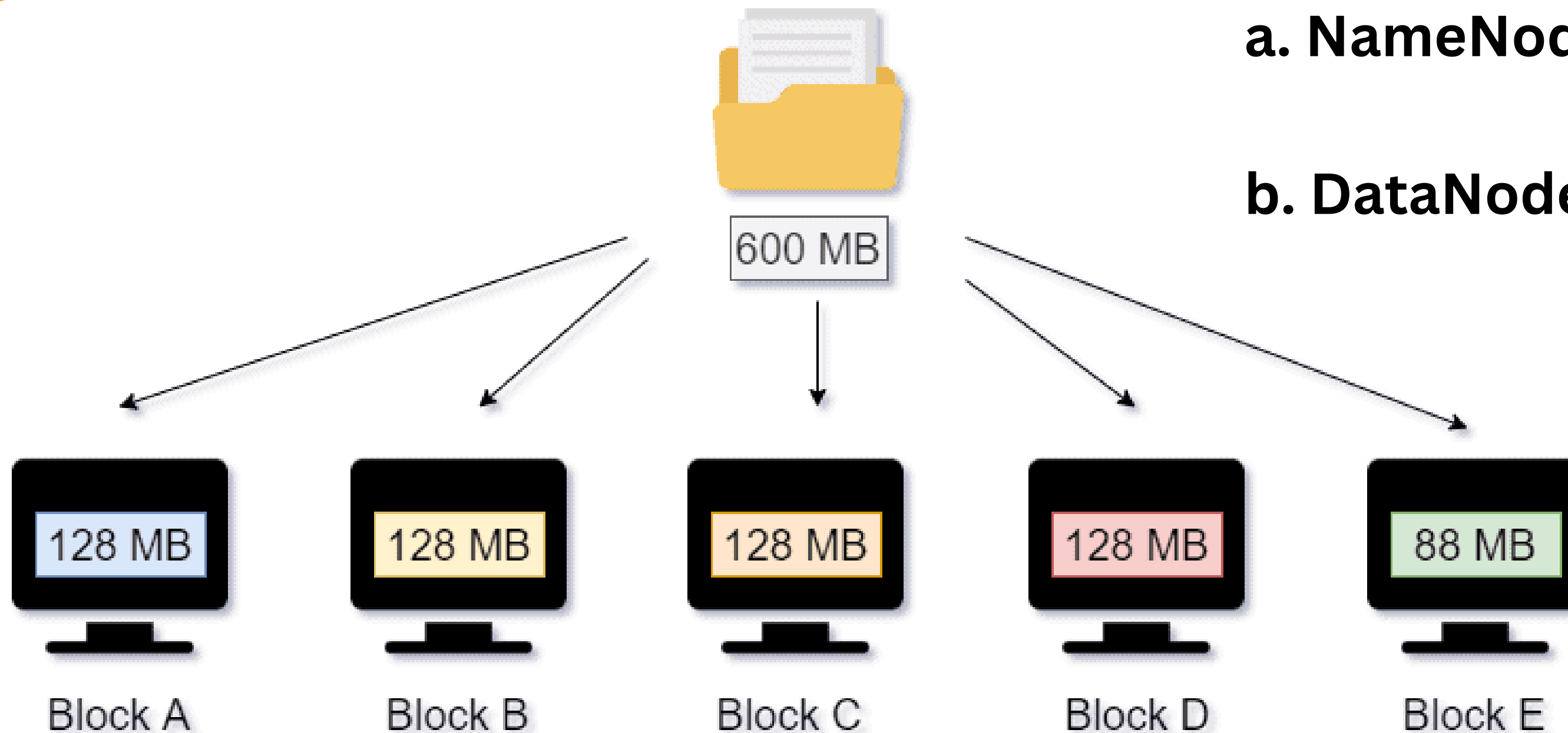
2. YARN (Yet Another Resource Negotiator)

ตัวช่วยในการจัดการและประมวลผลข้อมูล

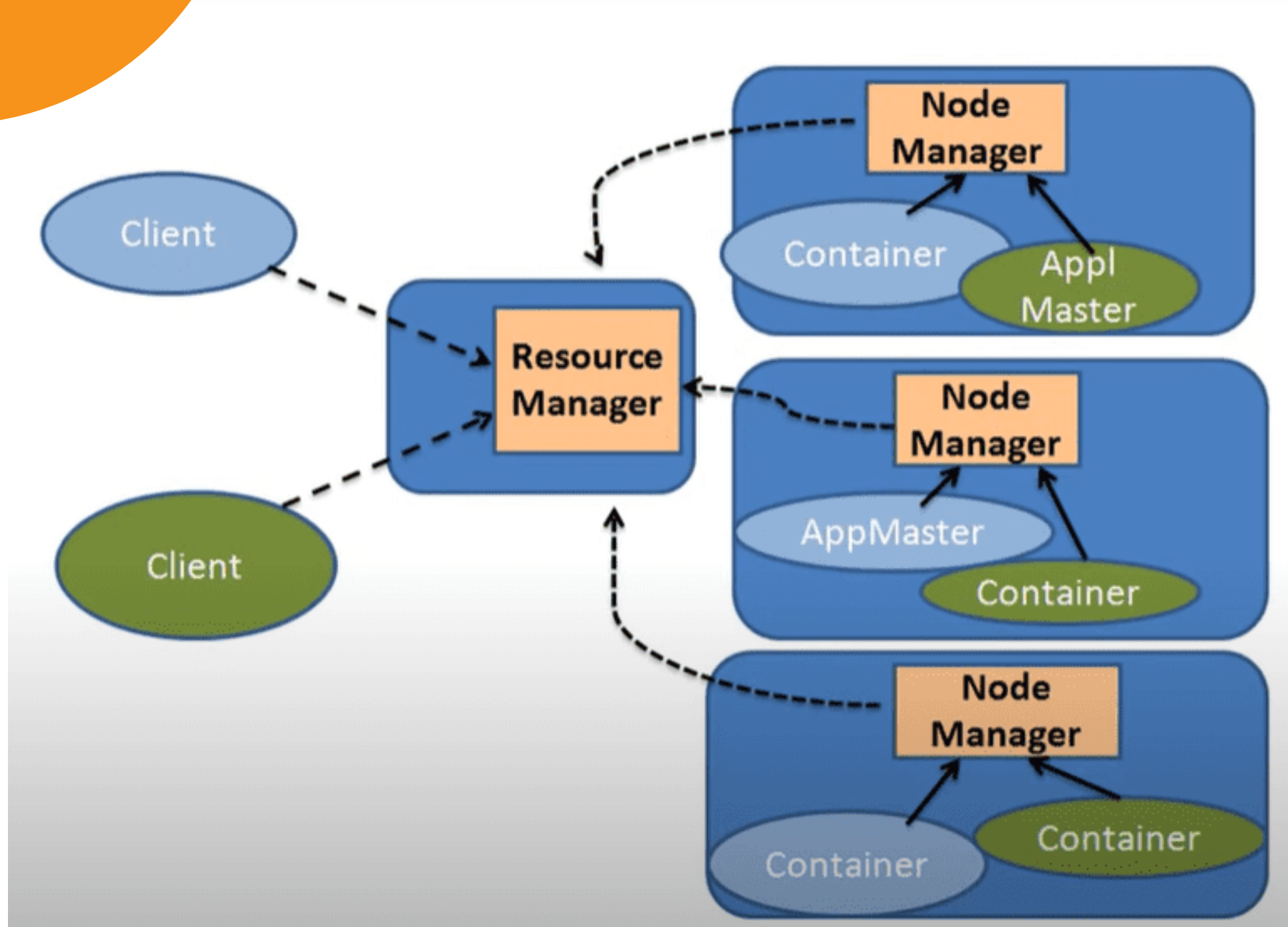
HDFS ประกอบด้วย 2 ส่วนย่อย ดังนี้

a. NameNode (Master Node)

b. DataNode (Slave Node)



ตัวช่วยในการจัดการและประมวลผลข้อมูล

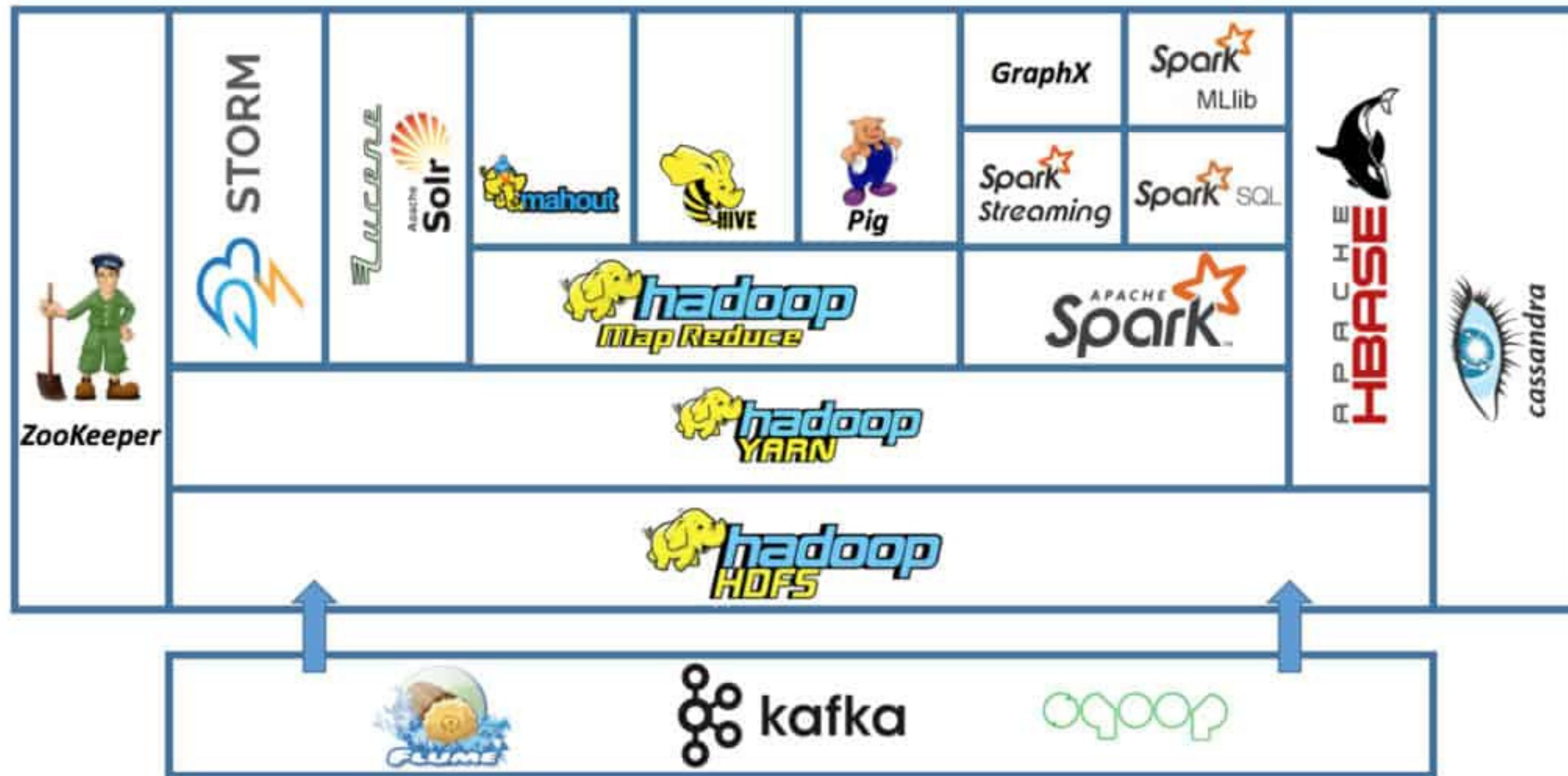


YARN มีส่วนประกอบหลัก ดังนี้

- **Client**
- **Resource Manager**
- **Node Manager**
- **Container**
- **Application Manager**

ตัวช่วยในการจัดการและประมวลผลข้อมูล

Hadoop Ecosystem





ตัวช่วยในการจัดการและประมวลผลข้อมูล



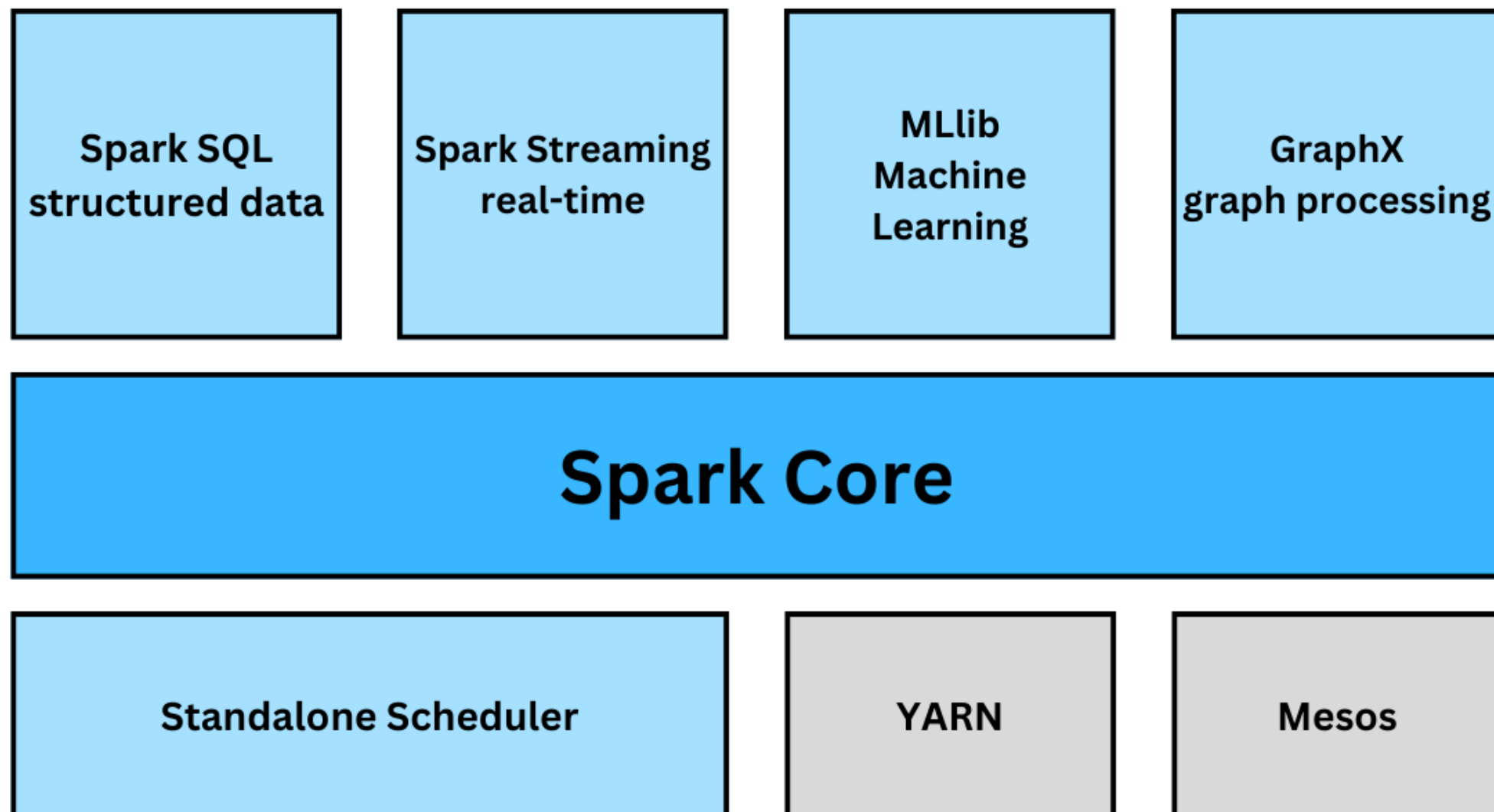
เป็นเครื่องมือหนึ่งที่ถูกดีไซน์ให้สามารถทำงานแบบกลุ่มได้ โดยที่เชื่อมต่อระบบการทำงานของคอมพิวเตอร์เข้าด้วยกัน หรือเรียกว่า “Cluster computing platform” ซึ่งสามารถกระจายงานที่ต้องทำไปยังเครื่องอื่น ๆ ภายในระบบได้ ทำให้เราสามารถประมวลผลข้อมูลขนาดใหญ่แบบเต็มประสิทธิภาพ หรือแบบ real-time ไปพร้อมๆ กันได้



ตัวช่วยในการจัดการและประมวลผลข้อมูล

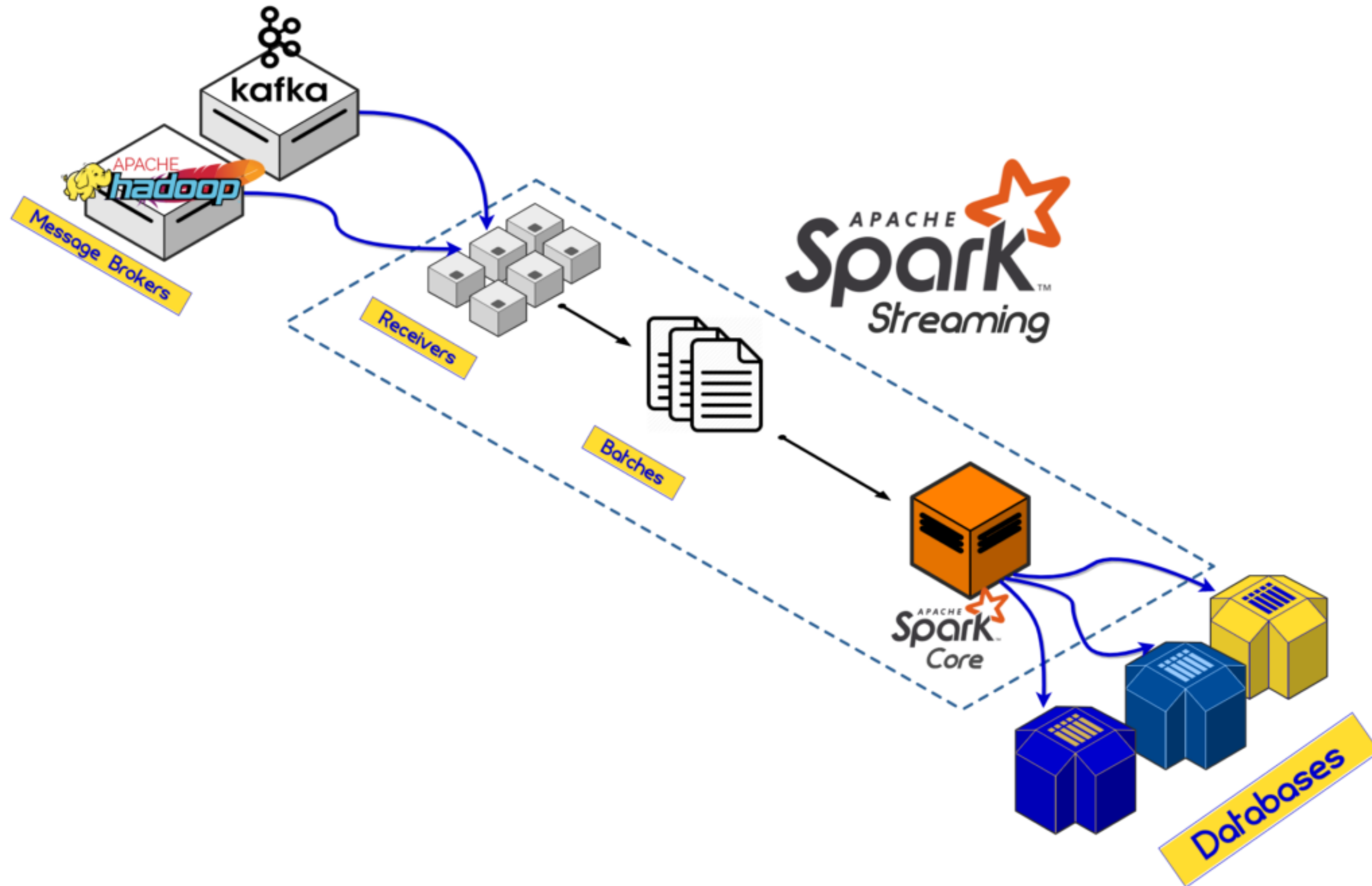


Spark มีส่วนประกอบหลัก ดังนี้



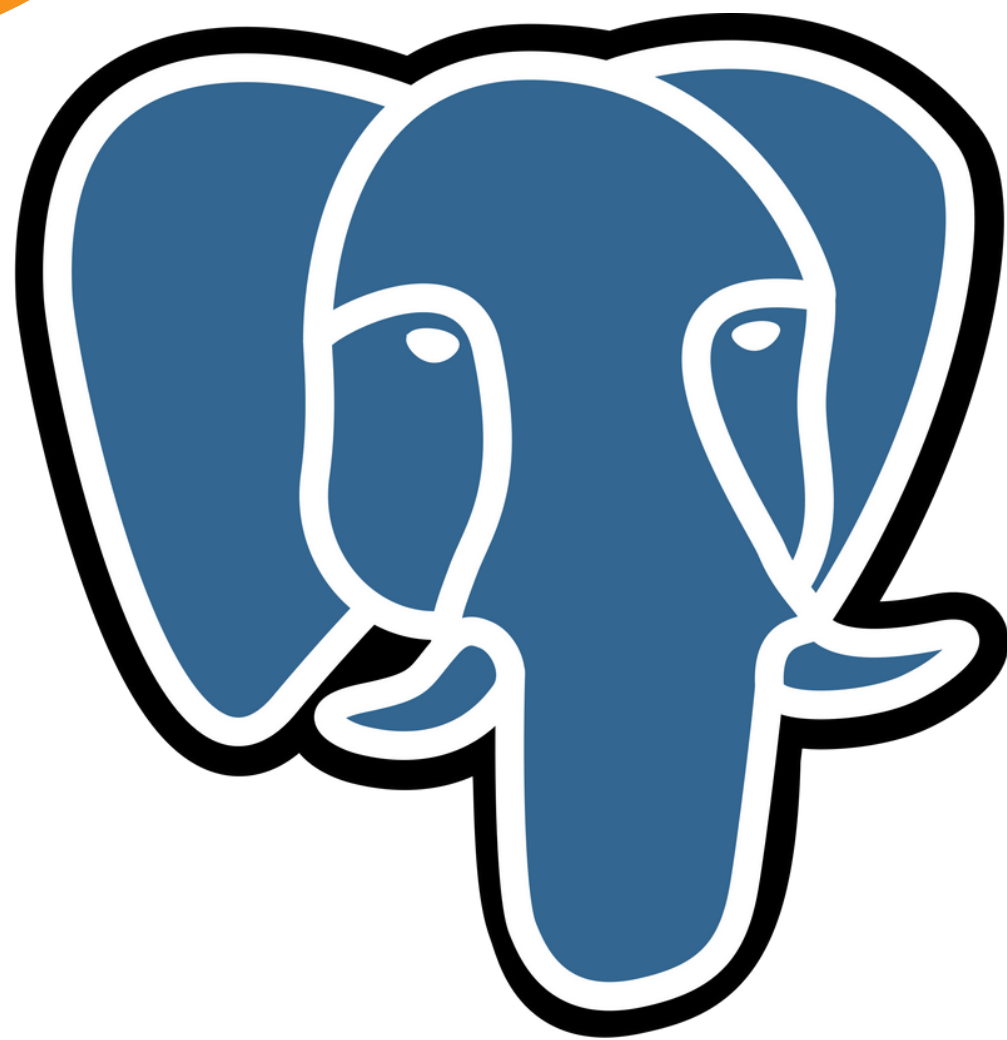
- **Spark Core**
- **Spark SQL**
- **Spark Streaming**
- **MLlib**
- **GraphX**
- **Cluster Managers**

ตัวช่วยในการจัดการและประมวลผลข้อมูล



การจัดเก็บข้อมูล (Data Store)

1. Relational Database



Postgre SQL

[จุดเด่น]

- เป็น open-source object-relational system ที่ใช้งานแบบเดียวกับภาษา SQL
- รองรับการจัดการด้วยหลากหลายภาษา อาทิ Python
- สามารถรองรับการขยายตัวของข้อมูลในอนาคตได้
- มี Feature ต่างๆ ในการตรวจสอบความถูกต้องของข้อมูล Database

[จุดด้อย]

- ประสิทธิภาพในระดับเมตริก และความเร็วอาจไม่ไวมาก
- ระบบ RDBMS ใช้เวลาสูง ในการบำรุงรักษา

การจัดเก็บข้อมูล (Data Store)

2. NoSQL Database



[จุดเด่น]

- เป็น database แบบ Document-Oriented
- ใช้ระบบการจัดการ memory แบบเดียวกับ cached memory ใน Linux
- ใช้ภาษา JavaScript เป็นคำสั่งในการจัดการข้อมูล

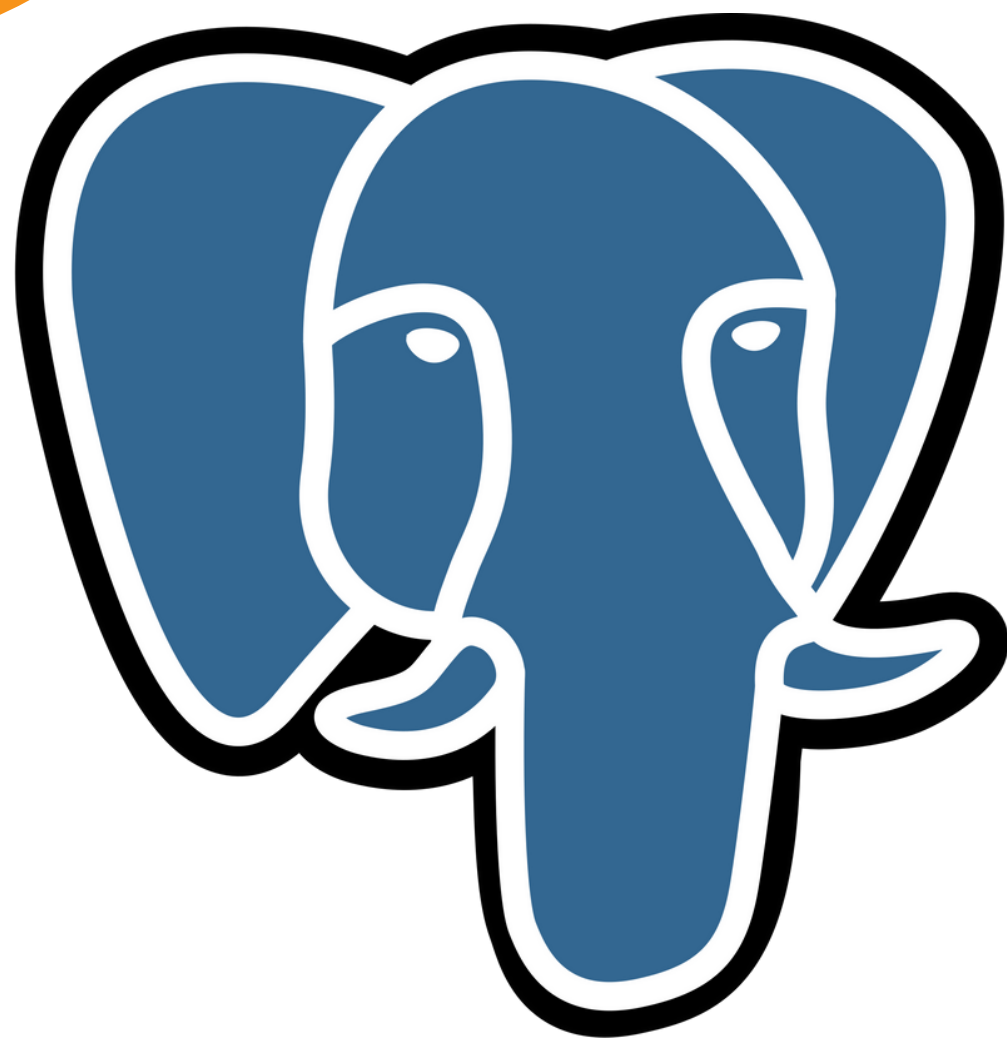
[จุดด้อย]

- ไม่ซัพพอร์ตการทำธุรกรรม
- การ Join ข้อมูลใน MongoDB นั้น ไม่ง่าย
- กินพื้นที่การเก็บข้อมูลมากกว่า MySQL
- ยากต่อการ handle เพราะขาดคุณสมบัติ ACID ทำให้อาจถูก corrupt ได้ง่าย

การจัดเก็บข้อมูล (Data Store)

เหตุผลที่เลือก PostgreSQL

- การแปลงข้อมูล (Data integrity) จากหลากหลายช่องทาง PostgreSQL จะพิจารณา การ integrate ข้อมูลของคุณ ด้วยการแนะนำข้อจำกัด และ ควบคุมข้อมูลที่คุณใส่เข้าไป
- มี Performance ในการทำงานคู่ขนานในการ queries
- ให้คุณเลือกเก็บข้อมูลได้หลากหลาย เพื่อความไม่มีลิมิตของคุณ
- ด้วยออฟชั่น replication ข้อมูลของคุณจึงปลอดภัยหายห่วง
- Support of non-relational data พวกไฟล์ JSON, XML, Hstore และ Cstore documents



Postgre SQL

กลยุทธ์การซื้อขาย

ทางทีมได้ใช้เครื่องมือที่เรียกว่า CDC Action Zone Version3 มาเป็น Indicator ในการหาคำสั่งซื้อขายโดย CDC Action Zone เป็นการคำนวณหาค่า exponential moving averages (EMA) สองค่าได้แก่ EMA 12 วัน และ 26 วัน แล้วนำค่าที่ได้มาเปรียบเทียบเพื่อหาสัญญาณซื้อ หรือสัญญาณขาย โดยค่า EMA สามารถคำนวณได้จากสมการด้านล่างนี้

**Exponential
Moving Average**

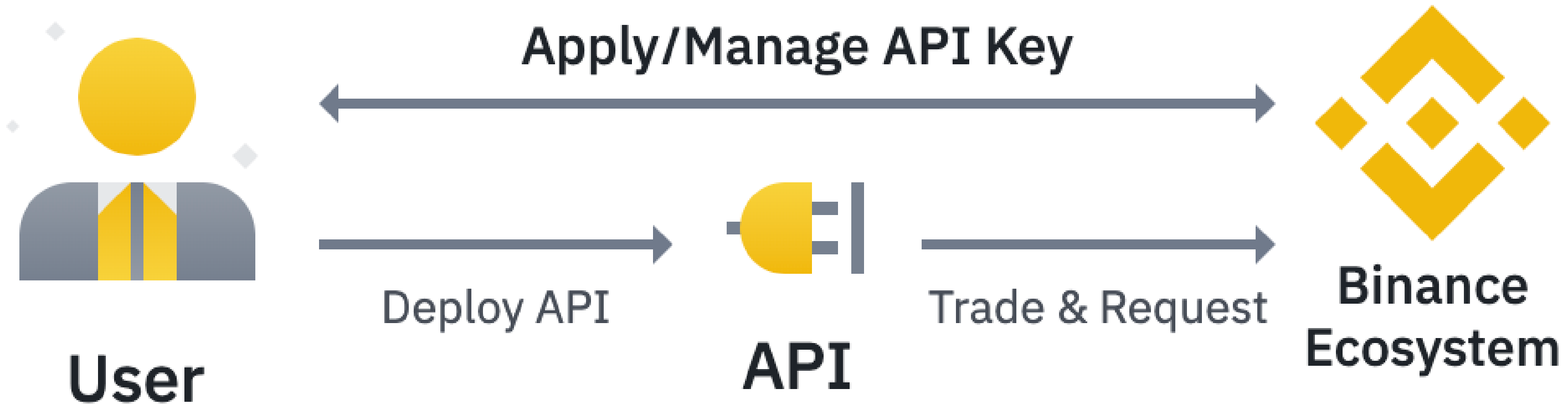
$$= C - P \times \frac{2}{(n + 1)} + P$$







กลยุทธ์การซื้อขาย






การส่งคำสั่งซื้อ-ขาย



การส่งคำสั่งซื้อ-ขาย

 Buy Crypto  Markets Trade Derivatives Earn Finance NFT 

Wallet Orders   Downloads English | USD 

Buy & sell Crypto in minutes

Join the world's largest crypto exchange

[Buy Now](#)

BNB/BUSD +4.92% 349.9 \$349.90	BTC/BUSD +1.49% 20,584.80 \$20,584.80	DOGE/BUSD -10.44% 0.11937 \$0.11937000	MATIC/BUSD +13.83% 1.1010 \$1.10	ETH/BUSD +2.05% 1,576.88 \$1,576.88
--	---	--	--	---









 Complete Fiat Transactions with USD, AUD or NGN to Get Up to 10,000 BUSD Per Day 11-04 | [More](#) >

 Special Notice about Binance.com in Singapore [More](#)

 Special Notice About Binance Markets Limited [More](#)

 **Lite Referral** >
Get 100 USDT






Apache
Airflow

การสร้าง Data Pipeline

เป็นเครื่องมือหนึ่งในการสร้าง Data Pipeline โดยเป็น Open Source Platform ตัวหนึ่งที่ใช้สามารถเขียนโปรแกรมที่จะมาควบคุมการไหลของ Workflow และสามารถคอยเฝ้าดูการไหลได้อีกด้วย ซึ่งถูกพัฒนาโดย Airbnb และเริ่มใช้งานมาตั้งแต่ปี 2015 ซึ่งนอกจากสร้าง Data Pipelines ได้แล้ว มันยังสามารถนำไปสร้าง หรือ พัฒนา ETL (Extract-Transform-Load), Machine Learning และ Predictive ได้อีกด้วย



การสร้าง Data Pipeline

 Airflow

DAGs

Security

Browse

Admin

Docs

09:49 CET (+01:00)





































































































NL

DAGs

All 26Active 5Paused 21

Filter DAGs by tag

Search DAGs

DAG	Owner	Runs	Schedule	Last Run	Recent Tasks	Actions	Links
 example_bash_operator example example2	airflow	  	0 0 * * *	2020-11-20, 09:47:09 	           	  	...
 example_complex example example2 example3	airflow	  	None	2020-11-20, 09:37:11 	           	  	...
 example_skip_dag example	airflow	  	1 day, 0:00:00	2020-11-20, 09:47:27 	           	  	...
 test_utils example	airflow	  	None	2020-11-20, 09:36:46 	           	  	...
 tutorial example	airflow	  	1 day, 0:00:00	2020-11-20, 09:33:50 	           	  	...

« < 1 > »

Showing 1-5 of 5 DAGs

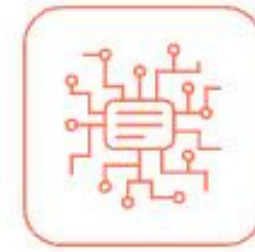


การสร้าง Data Pipeline



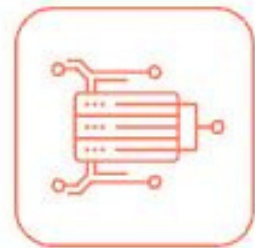
Scalable

Airflow has a modular architecture and uses a message queue to orchestrate an arbitrary number of workers.
Airflow is ready to scale to infinity.



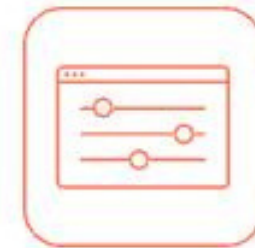
Dynamic

Airflow pipelines are defined in Python, allowing for dynamic pipeline generation. This allows for writing code that instantiates pipelines dynamically.



Extensible

Easily define your own operators and extend libraries to fit the level of abstraction that suits your environment.



Elegant

Airflow pipelines are lean and explicit. Parametrization is built into its core using the powerful Jinja templating engine.



การสร้าง Data Pipeline

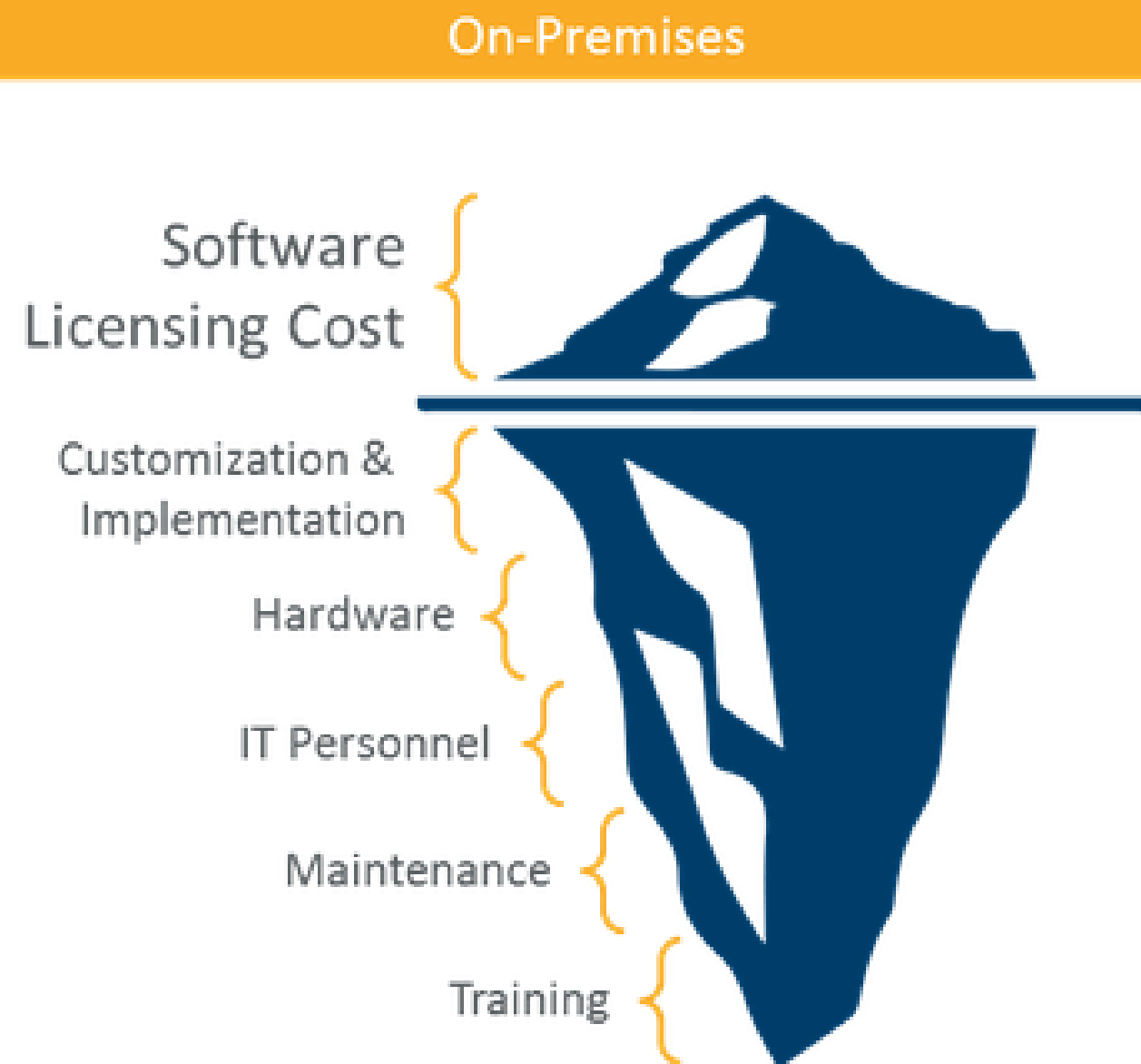
เหตุผลที่เลือก



- User Interface: มีความเป็น User Friendly สูงทำให้เข้าใจส่วนต่างๆ ได้ง่าย
- Python: เราสามารถเขียนด้วยภาษา Python เพื่อกำหนดงานต่าง ๆ ใน Data Pipelines ได้
- General purpose: เราสามารถใช้สร้าง Data Pipelines แบบไหนก็ได้
- Easy to add new functionality: ง่ายต่อการเพิ่มฟังก์ชันใหม่ๆ
- Easy to monitor: ติดตามผลการทำงานของ Workflow ใน State ต่างๆ ได้ง่าย
- Scale: ระบบถูกออกแบบมาให้ขยายขอบเขตการทำงานได้ง่าย
- มี Community ของผู้ใช้งานที่ใหญ่ ซึ่งง่ายต่อการปรึกษาหาทางออกได้รวดเร็ว

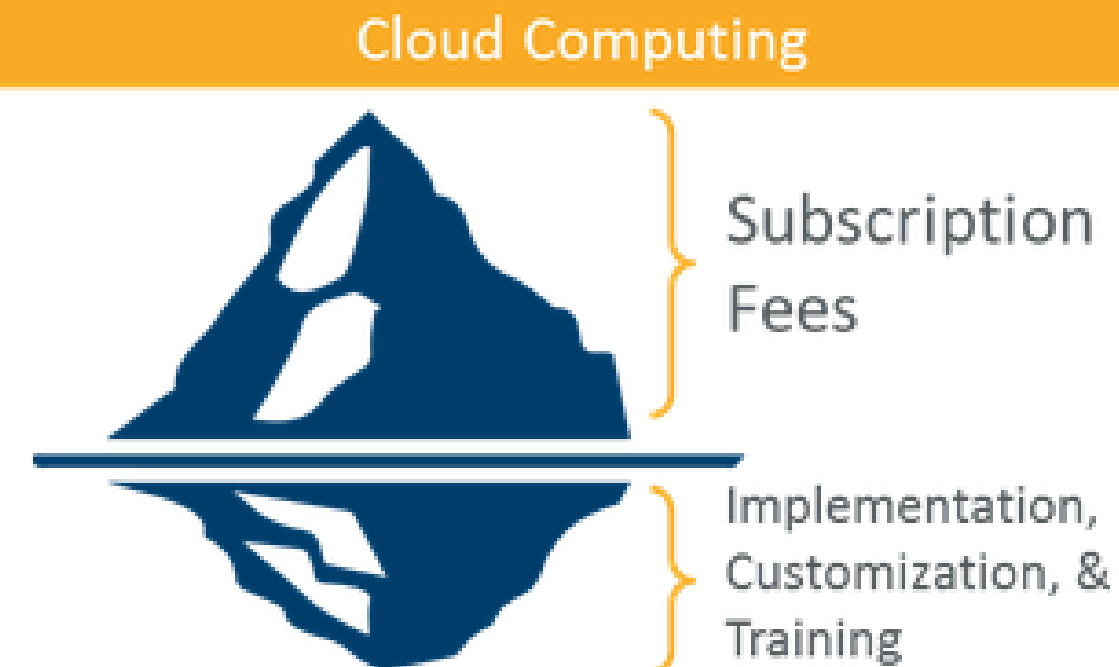


On-Cloud vs On-Premise



Ongoing Costs

- Apply patches, upgrades
- Downtime
- Performance tuning
- Rewrite customizations
- Rewrite integrations
- Upgrade dependent applications
- Ongoing burden on IT (hardware)
- Maintain/upgrade network
- Maintain/upgrade security
- Maintain/upgrade database



Ongoing Costs

- Subscription fees
- Training
- Configuration
- System Administration



On-Cloud vs On-Premise

แพลตฟอร์มผู้ให้บริการ Cloud Computing มาเป็นเวลานาน ด้วยบริการที่หลากหลายและครอบคลุมกว่า 175 บริการ เช่น ด้าน Analytic, Blockchain, Game, Internet of Things, Machine Learning, Quantum Technology ทำให้สามารถตอบสนองโจทย์ผู้ใช้ระดับองค์กรได้เป็นอย่างดี

1. ผู้ให้บริการ
2. ด้านราคาและค่าบริการ
3. ด้านความปลอดภัย



On-Cloud vs On-Premise



บริการ Cloud Computing จาก Microsoft ซึ่งมีฟังก์ชันและฟีเจอร์ย่อยหลากหลาย เรียกได้ว่าเป็นบริการ Cloud Server ที่ครบครันเจ้าหนึ่ง ทั้งเครื่องมือด้าน AI, Machine Learning, Blockchain, DevOps, Internet of Things, Storage, Web และบริการอื่น ๆ อีกมากมาย

1. ฟีเจอร์ที่ให้บริการ
2. ด้านราคาและค่าบริการ
3. ด้านความปลอดภัย



On-Cloud vs On-Premise

ผู้ให้บริการ Cloud Server จาก Google ที่เรารู้จักคุ้นเคยกันเป็นอย่างดี ซึ่งมีฟีเจอร์การทำงานที่หลากหลาย เนื่องจากเป็นบริษัทขนาดใหญ่อยู่แล้ว จึงทำให้มีบริการที่หลากหลายไม่แพ้ Azure หรือ AWS

Google Cloud

1. ฟีเจอร์ที่ให้บริการ
2. ด้านราคาและค่าบริการ
3. ด้านความปลอดภัย

On-Cloud vs On-Premise

เหตุผลที่เลือก



Google Cloud

หลังจากที่เปรียบเทียบทั้ง 3 บริษัทแล้ว ทางทีมได้ตัดสินใจเลือกใช้บริการ Google Cloud Platform ที่มีบริการที่ชื่อว่า “Google Compute Engine” หรือเรียกย่อ ๆ ว่า GCE ซึ่งเป็นบริการเครื่องคอมพิวเตอร์เซิร์ฟเวอร์ให้กับลูกค้าที่ต้องการนำไปทำงานต่าง ๆ เป็นระบบที่เราสามารถสร้าง เครื่องคอมพิวเตอร์เซิร์ฟเวอร์ได้อย่างรวดเร็ว ลดเวลาจากสัปดาห์มาเหลือแค่หลักนาที่เท่านั้น ซึ่งบริการนี้ทาง Google Cloud Platform มีสเปคของเครื่องคอมพิวเตอร์เซิร์ฟเวอร์ที่หลากหลาย เราสามารถสร้างเองได้ตามความเหมาะสมกับประเภทของงานที่ใช้ ไม่ว่าจะเป็นสเปคของ ซีพียู ขนาดของหน่วยความจำ หรือ ที่เก็บข้อมูล รวมถึงกำหนดระบบปฏิบัติการที่ต้องการใช้ได้ อาทิ เช่น Windows หรือ Linux เป็นต้น

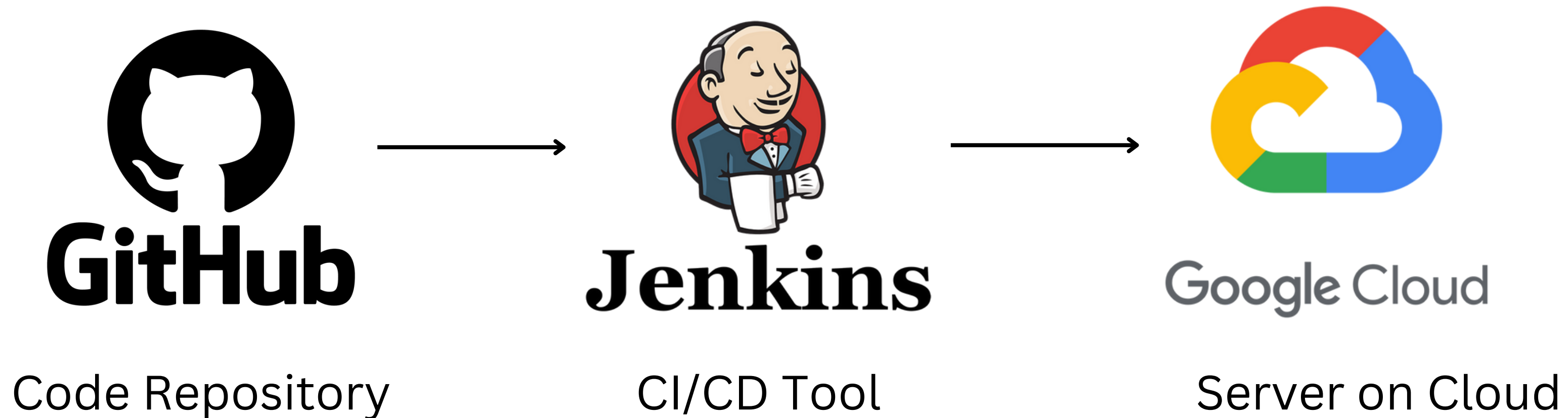


Test Case: Unit Testing

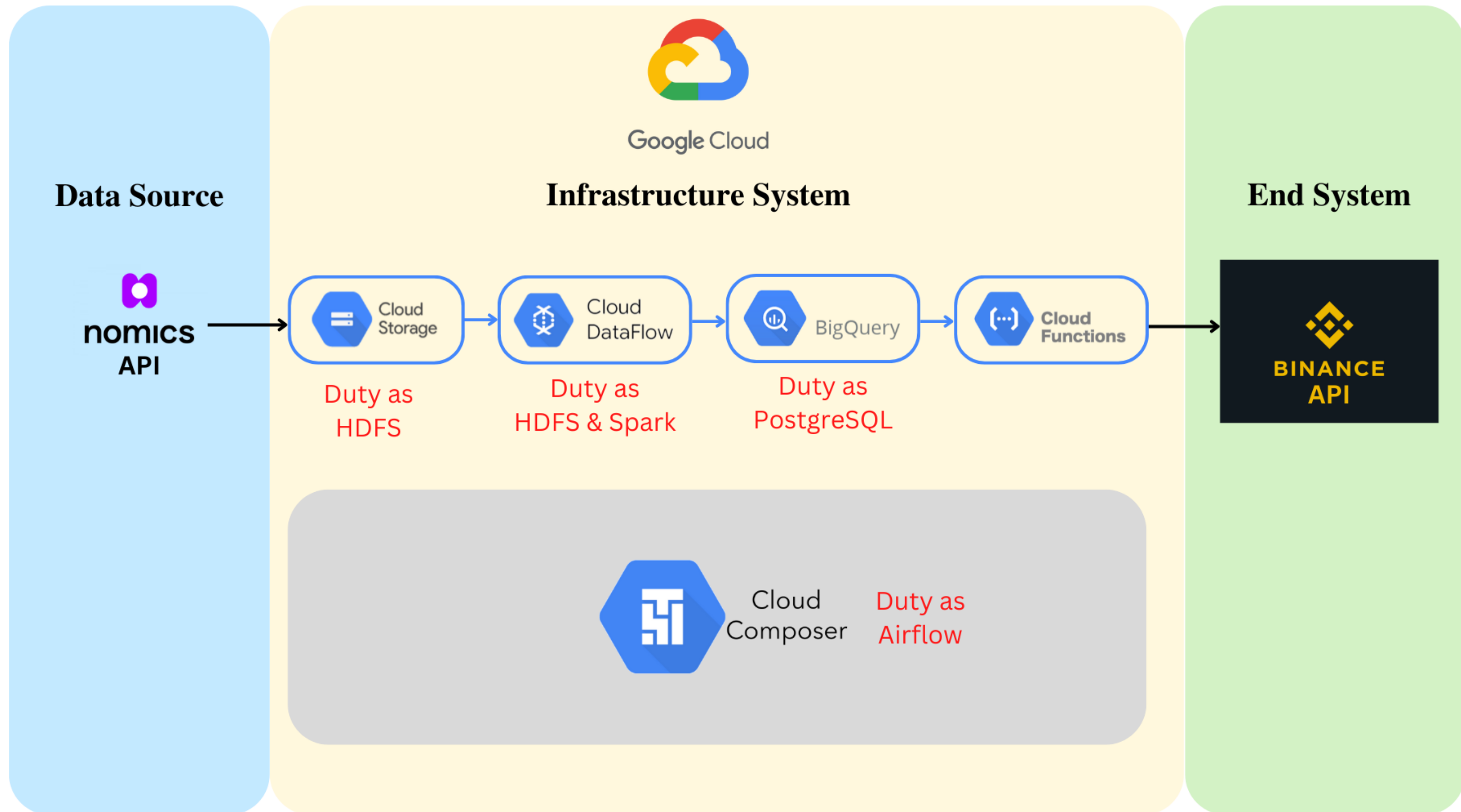
- Data Inquiry
 - Testcase: การเชื่อมต่อ APIs ไปยัง Nomics API
 - Expected Result: ได้รับข้อมูลตรงตามกับ Format ที่กำหนด + Status Code: 200
- EMA Calculation Engine
 - Testcase: ใส่ข้อมูลราคา Cryptocurrency
 - Expected Result: ค่า EMA12, 26 เทียบกับ ค่า EMA ที่คาดการณ์
- Database
 - Testcase: Insert ข้อมูลลงไปใน Database
 - Expected Result: Status Success
- Trading Engine
 - Testcase: ใส่ข้อมูลราคา EMA 12, 26
 - Expected Result: การแสดงผลคำสั่งซื้อขาย
- Exchange APIs Connectivity
 - Testcase: การเชื่อมต่อ APIs ไปยัง Binance API
 - Expected Result: ได้รับข้อมูลพอร์ต + Status Code: 200

CICD Process

- โปรแกรมจะต้องผ่านการทดสอบ Unit Testing ทุกครั้งก่อนที่จะถูก Deploy ขึ้นไปบน Google Cloud Platform ผ่าน CICD Tools ได้แก่ Jenkins



Final Data Pipeline



THANK YOU
FOR
YOUR KIND
ATTENTION!

