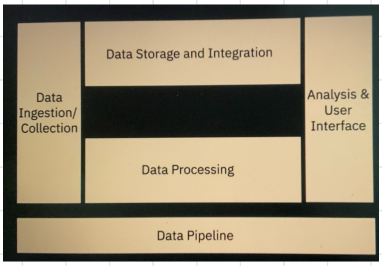
- **Apache Hadoop:** เป็นชุดเครื่องมือที่ช่วยจัดเก็บและประมวลผล Big Data แบบ distributed
  - → ใช้ Hadoop เพราะสามารถรองรับรูปแบบข้อมูลที่เกิดขึ้นใหม่ เช่น music streaming, video, social media sentiment ด้วยกฎโครงสร้างข้อมูล ที่ไม่ได้ถูกใช้ใน Data Warehouse
  - → stake-holder สามารถเข้าใช้งานได้แบบ Real time
  - → เพิ่มประสิทธิภาพและปรับปรุง costs ในคลังข้อมูลองค์กร โดยการรวบรวมข้อมูลทั้งหมดก่อน และย้ายข้อมูล "cold" data ที่หมายถึง data ที่ไม่ได้ถูกใช้บ่อย ไปยังระบบ Hadoop
  - ↳ **Hadoop Distributed File System (HDFS):** เป็นการจัดเก็บข้อมูลขนาดใหญ่ ที่ทำงานบน Hardware หลายตัวที่เชื่อมกันผ่าน Network ที่สามารถปรับขนาดได้และเชื่อถือได้
    - → แบ่ง Partition file บนหลาย node ช่วยให้ access และทำงานแบบ parallel ได้
    - → ทำสำเนา file block บน node ต่างๆ เพื่อป้องกัน data สูญหายและทนต่อข้อผิดพลาด
    - → HDFS จะทำการแบ่ง/ทำสำเนา ชิ้นส่วนต่างๆ ไปยัง server อื่นๆ ช่วยให้มีความพร้อมใช้งานเมื่อเกิด server ล้มเหลว ช่วยให้ Cluster แบ่งงานเป็นส่วนเล็กๆ แล้วรันงานแบบขนานบน server ช่วยให้ปรับขนาดได้ดี ช่วยให้ลดความแออัดของ Big Data
    - ↳ ข้อดี :
      - Recover ได้เร็วจากความล้มเหลวของ Hardware เพราะ HDFS ถูกสร้างให้ตรวจจับข้อผิดพลาดและการกู้คืนอัตโนมัติ
      - รองรับ Big Data เพราะสามารถเพิ่ม node ใน cluster ได้
      - พกพาง่าย เพราะพกพาได้บน Hardware หลายแบบและเข้ากันได้กับ Operation system หลายระบบ

- **Apache Hive:** Hive เป็น data warehouse software ที่ช่วยให้อ่าน, เขียน, และจัดการ Big Data โดยตรง ที่ถูกเก็บใน HDFS
  - ↳ HIVE base on Hadoop ทำให้การ query มีแนวโน้มจะมี latency สูงมาก
  - ↳ ไม่เหมาะกับ transection เพราะมันไม่เหมาะกับ Real-time
  - ↳ เหมาะกับ Data Warehouse เช่น ETL ช่วยให้เข้าถึงได้ง่ายผ่าน SQL

- **Apache Spark:** Spark เป็นเครื่องมือแบบประมวลผลที่ออกแบบมาสำหรับ App ต่างๆ, Interactive Analytics, Stream processing, ML, Data Integration และ ETL
  - ↳ ใช้ประโยชน์จากการประมวลผลจากหน่วยความจำ เพื่อเพิ่มความเร็ว แทนการอ่านจาก
  - ↳ มี Interface สำหรับ Python, R, Java, Scala และ SQL
  - ↳ สามารถทำงานได้โดยใช้ technology ของ Cluster แบบ Stand-alone เช่น Hadoop
  - ↳ สามารถเข้าถึงข้อมูลในแหล่งข้อมูลที่หลากหลาย เช่น HDFS และ HIVE ทำให้มีความหลากหลาย

Third Module

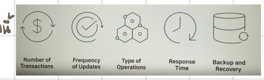
Layer of a Data Platform Architecture:



- **Data Ingestion/Collection layer** → ทำหน้าที่เชื่อมต่อกับระบบต้นทาง และนำข้อมูลจากระบบเหล่านี้มาสู่ Plat form ข้อมูล
  - ↳ main task : เชื่อมกับ data source , Transfer data จาก source ทำไปยัง data plat form ใน streaming , batch และทั้ง 2 modes
  - ↳ จัดการข้อมูลเกี่ยวกับ data collection ใน metadata repository
  - ↳ เมื่อเข้า data platform มันจะไป store และ Intregate
- **Data Storage and Integration layer:**
  - ↳ จัดเก็บข้อมูลเพื่อเตรียมไป processing และใช้ในระยะยาวต่อไป โดยทำ Transformation และ merge data ทั้ง logical และ physical ทำให้ data พร้อมใช้ทั้ง streaming และ batch mode
  - ↳ Storage layer ต้องเชื่อถือได้ , ปรับขนาดได้ , มีประสิทธิภาพสูง และคุ้มค่า
- **Data Processing** → Data Validation, Transformations , และใช้ logic ทางธุรกิจ เป็นสิ่งที่จะเกิดกับ layer นี้
  - ↳ layer นี้สามารถ อ่าน data ใน batch หรือ streaming mode จาก storage
  - ↳ จัดเตรียม data ให้ DA และ DS ไปใช้
  - ↳ Transformation บางอย่างเกิดใน layer นี้ ประกอบด้วย
    - การปรับ Structure โดยปรับรูปร่าง data และ schema อาจง่ายและยากแล้วแต่ความซับซ้อน หาก complex ก็จะมีการรวม data Table ด้วย Joins และ Unions
    - การ Normalization ที่ focus ที่การ clean database โดยเอา data ที่ไม่ใช้ออก และลดความซ้ำซ้อน และความไม่สอดคล้อง
    - Denormalization รวมข้อมูลหลาย table ไว้ใน table เดียว เพื่อให้ query มีประสิทธิภาพมากขึ้น เพื่อ report และ Analyse จะมีประสิทธิภาพขึ้นตาม
    - Data Cleaning แก้ไขความผิดปกติในข้อมูล เพื่อให้ข้อมูลน่าเชื่อถือสำหรับ downstream application
  - ↳ Storage และ processing อาจไม่แยก layer เสมอไป อาจเกิดใน layer เดียวกัน
    - ตัวอย่าง: Relational data สามารถถูกเก็บใน HDFS ก่อน จากนั้น ค่อยไป data processing ด้วย Spark
    - หรือ layer Processing อาจมาก่อน Store โดยที่อาจจะ Transform ก่อน Load เข้า storage ใน Database
- **Data Pipeline** → Data Ingestion , Data Storage และ Data Processing อยู่ใน Data pipeline ทั้งหมด อยู่ใน Data pipeline
  - ↳ รับผิดชอบในการนำไปใช้ และ maintain การไหลของข้อมูลให้ต่อเนื่อง

Designing of Data Stores:

- ↳ Data store หรือ Data Repository คือ data ที่ถูก collect, organize และ isolate (แยก) เพื่อ Analyse, ใช้ในองค์กร
- ↳ Data Repository มี Database, Data warehouse, Data Mart, Big data store และ Data lake
- ↳ Consideration for designing a data store: • ขึ้นกับ Type of data, Volume data, จุดประสงค์การใช้, Storage Consideration, และ Security Privacy และ Governance data
  - • Type of Data: Relational Database (RDBMSes) และ Non-Relational (NoSQL)
    - → RDBMSes เหมาะกับการใช้ data ที่เป็น Structure อยู่ในรูปแบบตาราง และมีการเขียน schema แล้ว
    - → NoSQL เหมาะกับ ไม่มีโครงสร้าง ที่ไม่มี schema และ free-form ไม่ต้องกำหนดความหมายและลักษณะไว้ล่วงหน้า
      - • key value
      - • Document based: ไม่เหมาะกับ complex search query และ การดำเนินการหลายอย่างพร้อมกัน
      - • Column based
      - • Graph based: ไม่เหมาะกับ data ที่มีปริมาณ analytics queries มาก ๆ
  - • Volume of Data: Data lake, Big Data Store
    - → Data lake: เก็บข้อมูลได้เยอะ เก็บจาก raw data มาเก็บได้เลย จาก source และเก็บได้ทั้ง RDBMS และ NoSQL
    - → Big Data Store: • เก็บ data ปริมาณมาก, ความเร็วสูง, หลากหลาย และต้องการ distributed processing
      - • Split large file ไปยัง computer ในหลาย ๆ เครื่อง ๆ ทำให้ใช้งานแบบ Parallel ได้
      - • คำนวณแบบ ขนานแต่ละ node ที่ data ถูกเก็บ
  - • Intend to use: พิจารณาหลายอย่าง



    - → Transactional System: ใช้กับปริมาณ data จำนวนมาก ต้อง อ่าน, เขียน และ update ได้
    - → Analytical System: ต้องการ complex queries ที่จะถูกนำไปใช้กับ ปริมาณข้อมูลขนาดใหญ่ ที่รองรับ data จาก transaction system ทำให้ต้องการความแน่นอน
    - → Schema design, indexing, และ partitioning strategy มีหน้าที่ใหญ่ที่ต่อ ๆ กัน ประสิทธิภาพ ของ system ขึ้นกับว่า data จะถูกนำไปใช้ยังไง
    - → Intend to use data • ช่วยเพิ่ม ความสามารถ ในการ scalability ในการพิจารณาในการออกแบบ เป็นความสามารถที่รองรับการโตขึ้นของข้อมูล, ผู้ใช้
      - • Normalization เป็นการจัดระเบียบข้อมูลใน data base เพื่อทำให้ ช่วยให้ พื้นที่ข้อมูล ถูกใช้ให้เกิดประโยชน์สูงสุด และ ช่วยให้ เข้าถึงข้อมูลได้ไวขึ้น
  - • Storage Considerations: from the perspective of storage
    - → Performance: มี ปริมาณงาน (Throughput) และ Latency
      - • Throughput: อัตราที่ ข้อมูลสามารถอ่าน และเขียน ไปยัง storage
      - • Latency: เวลาที่ใช้ในการ เข้าถึงข้อมูลที่เฉพาะใน storage
    - → Availability: Storage ต้องพร้อม เมื่อมีการ เข้าถึง ทุกครั้ง เมื่อเวลาต้องการ ไม่มีข้อยกเว้น และไม่ควร หยุดทำงาน
    - → Integrity: Data ต้องปลอดภัยจากการ corruption, สูญหาย และ การโจมตีจาก ภายนอก
    - → Recoverability: storage ควรมั่นใจว่า สามารถ recover data ได้ เมื่อ เกิดอะไรขึ้น แก่ไฟล์
  - • Privacy, Security, และ Governance: มี Access control, Multizone Encryption, Data Management และ Monitoring System
    - → กฎระเบียบ: GDPA, CCPA, HIPAA

Security: → ใช้ CIA Triad เป็นกลยุทธ์ ที่ใช้เพื่อ ความปลอดภัย มี 3 อย่าง

• C มาจาก Control การ ทำให้ คนที่ ไม่ได้รับอนุญาต • I มาจาก Integrity ตรวจสอบว่า resource เรา เชื่อถือได้ และ ไม่ถูกดัดแปลง • A มาจาก Availability ทำให้ มั่นใจว่า ผู้ใช้ที่ ได้รับอนุญาต เท่านั้นจะสามารถ เข้าถึงได้

↳ เอา CIA ไปใช้กับสิ่งต่าง ๆ: Infrastructure Security, Network Security, Application Security, และ Data security

- • Physicle Infrastructure Security:
  - → Access to the perimeter (ขอบเขต) ควรมี หน่วยความปลอดภัย ตาม Authentication
  - → มีการตรวจตรา พื้นที่ ตลอดเวลา ตลอด 24 hr.
  - → มีการใช้ พลังงานไฟฟ้า จากหลาย จุดไม่ใช่ แค่ไฟหลัก, จาก UPS เพื่อ ป้องกันไฟดับ
  - → กลไกการทำความร้อน / เย็น เพื่อ รักษาอุณหภูมิ และความชื้น
  - → คำนึงถึงความปลอดภัยจาก ธรรมชาติ ก่อนสร้าง โรงงาน ดูว่า มีน้ำท่วม หรืออะไรมั้ย
- • Network Security: สำคัญต่อการเชื่อมต่อ และ ข้อมูลให้ปลอดภัย
  - → Firewall ป้องกัน การเข้าถึง Network โดยไม่ได้รับอนุญาต
  - → Network Access Control เพื่อมั่นใจถึงความปลอดภัย ของปลายทาง โดย device ที่ ได้รับ อนุญาต เท่านั้นที่ จะเข้าได้
  - → Network Segmentation การแบ่ง Network เพื่อสร้าง silos
  - → Security Protocols ป้องกันไม่ให้ ผู้ไม่หวังดี เข้าถึงข้อมูลขณะส่ง
  - → Intrusion (การบุกรุก) Detection และ Intrusion Prevention ตรวจสอบ การรับข้อมูลผ่าน Input เพื่อ ตรวจสอบ ความพยายามในการบุกรุก และ ช่องโหว่
- • Application Security: รักษา Privacy ของลูกค้า และทำให้ App มั่นใจและตอบสนองดี
  - → Threat model เพื่อระบุ จุดอ่อน และรูปแบบการโจมตีที่อาจเกิด
  - → Secure design ช่วยลดความเสี่ยง
  - → Secure coding เป็นแนวทางปฏิบัติในการ code ที่ป้องกันช่องโหว่
  - → Security testing แก้ ไขปัญหา ก่อนที่จะ deploy App เพื่อยืนยันว่า ปัญหาเหล่านั้น ถูกแก้ไขแล้ว ความปลอดภัย

- • Data Security
  - → Data มีทั้ง อยู่ใน storage และระหว่างการส่ง เราต้องป้องกัน ทั้งหมดนี้
  - → มีการ Autholization ด้วยวิธีต่าง ๆ ว่า เราคือใคร และเป็นคนที่ต้อง จริงมั้ย โดยวิธี ตรวจสอบ password, ลายนิ้วมือ, Token
  - → Data ใน storage ที่เป็น Database หรือ Data warehouse ทาง physical อาจมีการ โจรกรรม หรือ ใช้รหัสกับ ข้อมูลได้
  - → Data ที่ transit อยู่ อาจมีการ encryp ด้วย https, ssl และ tls