COMP5339: Data Engineering

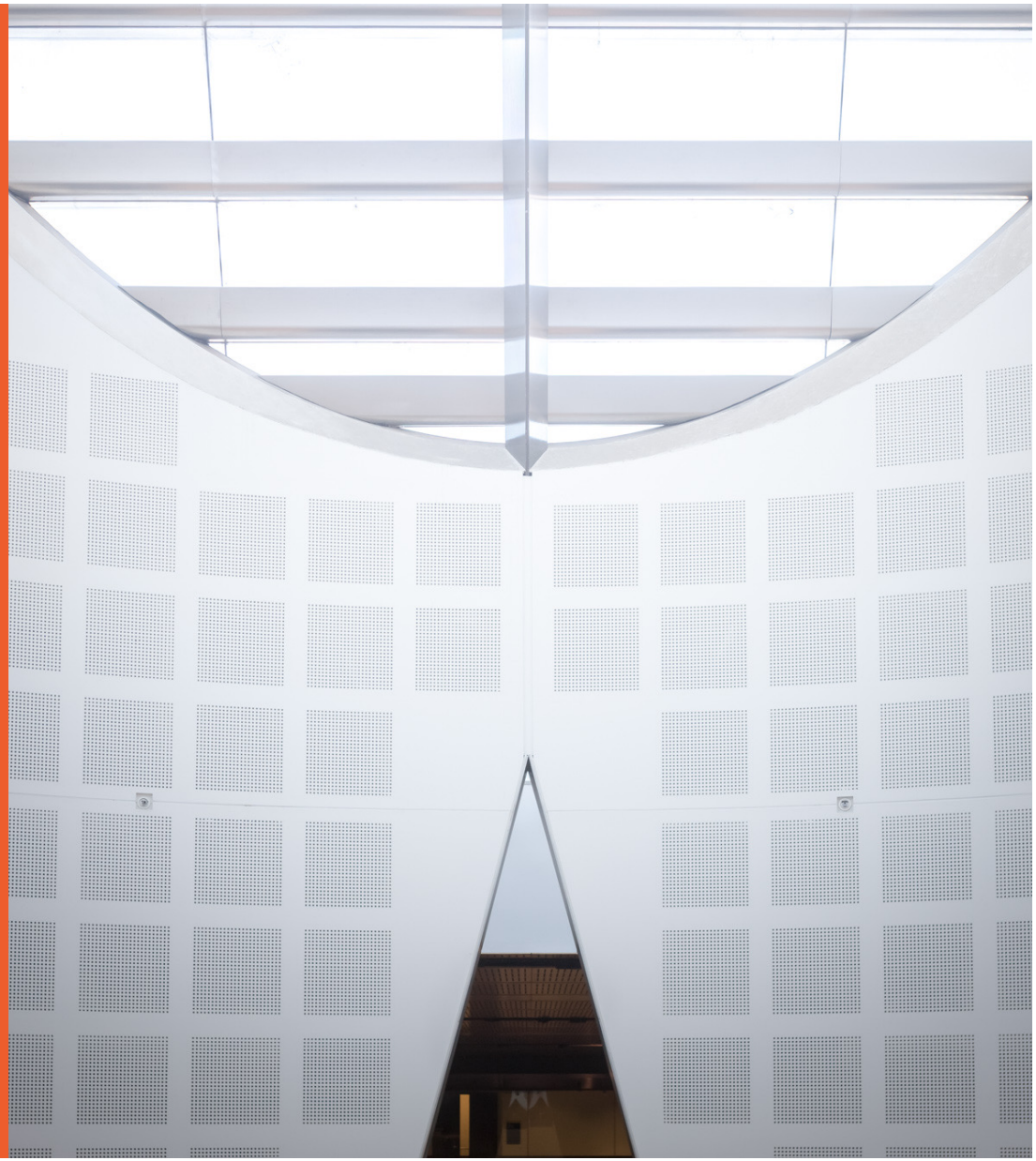
W12: Security and Privacy

Presented by

Uwe Roehm

School of Computer Science

THE UNIVERSITY OF SYDNEY



eHealth / Telemedicine Event Analysis

- Medical Data Analysis for Remote Health and Activities Monitoring
  - Wearable medical sensors and IoT allow for remote health monitoring
  - Event-driven
- Challenges:
  - Scalability for large populations
  - Data Privacy
  - Security

Source: Shutterstock

Motivation

- Security is vital for data engineering
  - We often deal with sensitive data and information
  - Protecting sensitive data is not just a technical issue but a business-critical responsibility.

- Security key for data privacy
  - Recent high-level data breaches examples in 2022 - 2025:
    - Optus, Medibank, Latitude Financial and Qantas Frequent Flyer
    - Data of millions of customers affected
    - Latitude: 14 million customer records affected, including historic data more than 10 years old, incl. driver license numbers, addresses, DoB

Today's Agenda

- Data Analytics and Data Privacy
- Data Security
- Data Backup
- Choosing Technologies across the Data Engineering Lifecycle

Slide deck incorporates slides from Elliot Zhu's COMP5339 lecture in 2024 Sem2.

Data Privacy

Data Privacy

- Data privacy of significant legal importance in every country
- For example:
  - General Data Protection Regulation (GDPR) was passed in Europe in mid-2010s
    - GDPR also addresses the transfer of personal data outside the EU
  - California Consumer Privacy Act (CCPA), adopted on 28 June 2018, has many similarities with the GDPR
- Australian Privacy Principles (APP)
  - cornerstone of the privacy protection framework in the Privacy Act 1988
  - upheld by Office of the Australian Information Commissioner (OAIC), the independent national regulator for privacy and freedom of information.
  - https://www.oaic.gov.au/privacy/australian-privacy-principles

Personally Identifiable Information (PII)

- OAIC: "Information that, when used alone or with other relevant data, can identify an individual." – such as:
  - an individual's name, signature, address, phone number or date of birth
  - sensitive personal information
  - credit or financial information
  - employee records
  - photographs or video recording
  - internet protocol (IP) addresses
  - Voiceprints and facial recognition biometrics
  - location data from a mobile device

https://www.oaic.gov.au/privacy/your-privacy-rights/your-personal-information/what-is-personal-information

Sensitive Information

- **Sensitive information**, according to AOIC, is personal information that includes information or an opinion about an individual's:
  - racial or ethnic origin
  - political opinions or associations
  - religious or philosophical beliefs
  - trade union membership or associations
  - sexual orientation or practices
  - criminal record
  - health or genetic information
  - some aspects of biometric information.
- sensitive information has a higher level of privacy protection than other personal information

https://www.oaic.gov.au/privacy/your-privacy-rights/your-personal-information/what-is-personal-information

Special Data Protection Rules

- Some data has special protection laws that restrict where it can be stored or which processes have to be in place to manage / process such data
- Examples
  - Tax file number
  - Social security number (USA)
  - Credit card data
  - Health records
  - and other highly sensitive personal or financial data

Data Minimalism

- The best way to safeguard private and sensitive data is to avoid collecting it only unless absolutely necessary.
- Carefully consider data usage and gather sensitive data only if it is essential for downstream processes.
- Data engineers should proactively assess potential attack and leak scenarios when designing or using any data pipeline and storage system.
- Ensure the security of your access credentials at all times.

Right to be Forgotten

- The data minimalism principle should also apply for keeping historic data.
  - Recent major data breaches have exposes some PII of *former* customers
- Not a legal requirement in Australia (as of 2023)
- But GDPR for example explicitly states a "*right of erasure*"
  - GDPR Article 17 provides that a data subject has the right to request erasure of personal data related to them.
  - Article 21 of the GDPR allows an individual to object to processing personal information for marketing or non-service related purpose.

=> Therefore, data architecture should be designed to easily identify and erase outdated or unnecessary PII.

Principle of Least Privilege

- Privileges and data access should be limited to what a person or system needs to complete a specific task.

- Apply the same principle to both humans and machines: grant them only the necessary access for their role, and restrict it to the required timespan.

- Implement granular access controls at the column, row and even cell-level for sensitive data.

- Consider masking PII and other sensitive information, and create views that contain only the data users need to access.

Processes

- Avoid security theatre
- Embed good security and privacy governance into your organization.
  - Real security requires *commitment* by team, not just ticking of forms for compliance.

- Keep it simple
- Regular security trainings

Data Analytics and Australian Privacy Principles

Key recommendations by OAIC regarding data analytics:

- Use de-identified data where possible.
- Embed good privacy governance into your organisation by taking privacy-by-design approach.
- Conduct Privacy Impact Assessments for your data analytics projects.
- Be open and transparent about your privacy practices.
- Know what you're collecting.
- Be careful with sensitive information.
- Establish grounds for new uses of information.
- Provide options.
- Ensure the accuracy of information.
- Protect information in line with your risk assessments.

https://www.oaic.gov.au/privacy/privacy-guidance-for-organisations-and-government-agencies/more-guidance?a=3086

Security

Slides based in parts on Elliot Zhu's COMP5339 slides from 2024sem2.

Scope

- **Assets**: Data objects (files, tables, views, rows) and the system managing them.
- **Threats**: Power failures, employee fraud, denial of service, unauthorised access.
- **Security** involves protecting assets, detecting breaches, minimising loss, and recovering.

Not all data is sensitive, but critical data requires robust protection.

Threats to Data Security

- **Unauthorised modification**: Sabotage, crime, or ignorance leading to data changes.
- **Unauthorised disclosure**: Sensitive data is leaked.
- **Loss of availability**: Denial of Service (DoS) preventing access to the database.
- **Commercial sensitivity**: Fraud often originates from internal threats.
- **Personal privacy & data protection**: Legislative controls require audit trails and careful handling.

Technology

- Patch and Update Systems
- Encryption – both **at-rest** and **in-transit**
  - Not a magic bullet as still threat of human security breach to obtain credentials
- Logging, Monitoring and Alerting
  - part of DataOps is to observe, detect and alert on incidents by automated monitoring, logging and alerting of peculiar events on
    - Access, Resources, Billing, Excess permissions
  - If possible, set up automatic anomaly detection
- Network Access
  - e.g. do not keep data in public available S3 buckets or other cloud storage locations
  - restrict APIs, databases, or SSH access to known networks

Encryption

- **Encryption at rest**
  - Server-side encryption for all data stored on servers, databases, cloud object containers
  - All laptops should have full disk encryption on by default in case a device gets stolen
  - Backups should be kept encrypted too
- **Encryption in transit** / over the wire
  - https nowadays the default for modern cloud APIs
  - be aware of how API keys are handled
  - be aware of security limitations of legacy tools or protocols (e.g. ftp is vulnerable to man-in-the-middle attacks)

Example: TLS (Transport Layer Security)

- **TLS (Transport Layer Security)** is a cryptographic protocol designed to provide secure communication over a computer network.
- Key Features:
  - **Encryption**: Ensures data privacy by encrypting communication between clients and servers, preventing unauthorised access.
  - **Authentication**: Verifies the identity of both parties (client and server) involved in the communication, ensuring that data is sent to the intended recipient.
  - **Data Integrity**: Protects data from being altered or tampered with during transmission, ensuring the content remains unchanged.
- Why TLS is critical:
  - Protects sensitive data such as passwords, credit card numbers, and personal information from eavesdropping or interception.
  - Widely used in securing web communications, APIs, and Delta Sharing for secure data transmission in data engineering pipelines.

Other Data Security and Privacy Aspects

1. **Data Storage Security:**
   - **Data encryption at rest**: Ensuring that data is encrypted while stored in databases, data lakes, or cloud storage to prevent unauthorised access.
   - **Access control mechanisms**: Implementing role-based or attribute-based access controls (RBAC/ABAC) to restrict who can access and modify data.
2. **Data Access Management:**
   - **Authentication and Authorisation**: Ensuring users and systems are properly authenticated and authorised to access sensitive data, often through multi-factor authentication (MFA) or OAuth mechanisms.
   - **Least Privilege Principle**: Granting minimal access rights necessary for users and systems to perform their roles to reduce risk.
3. **Data Masking and Anonymization:**
   - **Data masking**: Concealing sensitive data (e.g., personally identifiable information) with obfuscated data when access to full data is not required.
   - **Data anonymization**: Removing identifiable information from datasets to prevent unauthorised identification of individuals.

Other Aspects (cont'd)

4. Network Security:

- **Firewalls and VPNs**: Implementing network security measures like firewalls, virtual private networks (VPNs), and network access control lists (ACLs) to limit access to data from trusted networks.
- **Zero Trust Architecture**: Requiring verification of every device and user before granting network access to sensitive data systems.

5. Data Integrity and Validation:

- **Checksums and hashing**: Ensuring that data has not been altered during storage or transmission by using hash functions or checksums for integrity validation.
- **Data quality controls**: Ensuring data accuracy and consistency across different systems to avoid security and compliance issues.

6. Data Governance and Compliance:

- **GDPR, HIPAA, CCPA**: Adhering to regulations that govern how sensitive data should be handled, stored, and shared to avoid legal penalties.
- **Data retention policies**: Defining how long data should be retained and ensuring it is securely deleted when no longer needed.

Other Aspects (still cont'd)

7. Incident Response and Disaster Recovery:

- **Data breach response plans**: Implementing procedures to quickly respond to data breaches, containing the damage and preventing further unauthorised access.
- **Backups and disaster recovery**: Ensuring that data is regularly backed up and can be recovered in case of system failures or cyberattacks.

Backup & Recovery

Slides based on Elliot Zhu's COMP5339 slides from 2024sem2.

Disaster Prevention – Backups

- Always backup your data – regularly

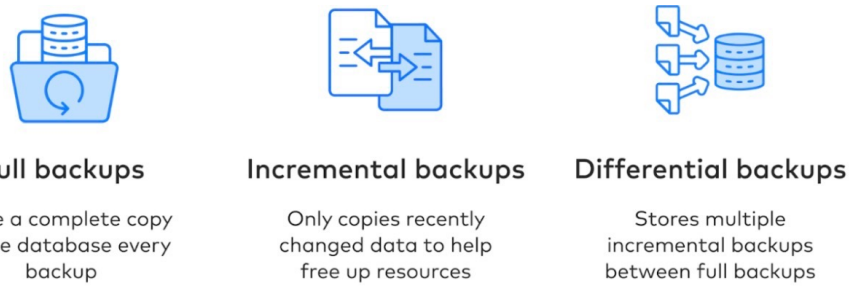
- 3 – 2 – 1 rule
  - At least 3 copies
  - On 2 different media
  - At least 1 off-premise

- Failures and accidents can always happen
- Also protects against ransomware attacks

Backup and Recovery

- **Backup**: Creating a copy of data to protect it from loss, corruption, or failure
- **Recovery**: Restoring data from a backup after a data loss event.
- Importance:
  - Ensures business continuity in case of failures, disasters, or cyberattacks.
  - Protects data integrity and availability.
- Key Concepts:
  - Backup Frequency
  - Recovery Point Objective (RPO)
  - Recovery Time Objective (RTO)

Types of Backup



- Full Backup:
  - Complete copy of all data.
    Advantages: Simple to restore, single source of truth.
  - **Disadvantages**: Time-consuming and requires large storage space.
- Incremental Backup:
  - Backs up only the changes since the last backup (incremental or full).
  - **Advantages**: Fast and requires less storage.
  - **Disadvantages**: Restoring can be complex as it requires chaining multiple backups.
- Differential Backup:
  - Backs up changes since the last full backup.
  - **Advantages**: Easier to restore than incremental, less storage than full backup.
  - **Disadvantages**: Takes longer than incremental, requires more storage.

Backup Strategies

- Backup Frequency:
  - **Daily, Weekly, Monthly**: Based on data criticality and business needs.
  - **Automated vs. Manual Backups**: Automated ensures consistency, reduces human error.
- Backup Location:
  - **On-Premise Backup**: Local storage devices like hard drives or network-attached storage (NAS).
  - **Cloud Backup**: Storing backups in cloud environments (e.g., AWS, Azure).
  - **Hybrid Backup**: Combines on-premise and cloud for higher redundancy.

Recovery Point Objective (RPO)

- The **maximum amount of data loss** acceptable in case of a failure, measured in time.
  - *Example*: An RPO of 1 hour means data backups must be made at least every hour.
- Importance:
  - Defines how often backups need to be performed.
  - Affects business operations and data recovery plans.
- Choosing an RPO:
  - Based on business-critical data: The more critical, the shorter the RPO.
  - Cost trade-offs: Shorter RPOs require more frequent backups and higher storage costs.

Recovery Time Objective (RTO)

- The **maximum amount of time** acceptable to restore operations after a failure.
  - *Example*: An RTO of 2 hours means the system must be fully restored within that time.

- Factors Affecting RTO:
  - **Backup Location**: Cloud backups may take longer to restore than local backups.
  - **Infrastructure Readiness**: Having the required hardware or virtual machines ready for recovery.
  - **Data Size**: Large volumes of data take longer to restore.

Backup Solutions for Distributed Systems

- Challenges:
  - Data distributed across multiple locations and databases.
  - Consistency between backups across distributed nodes.
- Solutions:
  - **Distributed Backup Tools**: Solutions like **Google Cloud Spanner** or **Amazon Aurora** have built-in distributed backup mechanisms.
  - **Replication**: Using replication as part of a backup strategy to maintain multiple copies of data across nodes.
- Impact on Concurrency:
  - During backups, concurrency control needs to ensure no conflicting transactions occur, especially with replicated data.

Backup and Recovery for Distributed Databases

- Replication as a Backup Strategy:
  - **Eager Replication**: Ensures data is immediately consistent across multiple nodes, reducing data loss but with higher overhead.
  - **Lazy Replication**: Asynchronous replication that allows eventual consistency but might risk some data loss during recovery.
- Backup Challenges in Distributed Systems:
  - **Consistency**: Ensuring that backups are consistent across nodes.
  - **Network Latency**: Recovering from geographically distributed backups can increase recovery time due to network delays.
- **Best Practice**: Combining **local replication** with **remote cloud backups** ensures both high availability and disaster recovery readiness.

Types of Recovery

- **Crash Recovery:**
  - Recovering from a system crash, often involves undoing incomplete transactions to bring the database to a consistent state.
  - **Techniques**: Using **logs** to roll back incomplete transactions.
- **Disaster Recovery:**
  - Recovering from a large-scale disaster, such as hardware failure, natural disasters, or cyberattacks.
  - Techniques:
    - **Failover systems**: Automatically switching to a backup system.
    - **Disaster recovery sites**: Maintaining a separate recovery location.
- **Point-in-Time Recovery (PITR):**
  - Restoring the database to a specific point in time, useful for undoing accidental deletions or corruption.

Backup and Recovery Best Practices

- **Regular Testing:**
  - Regularly test backups to ensure they work and can be restored quickly.
- **Multiple Copies:**
  - Keep **multiple copies** of backups (e.g., on-premise and cloud) for redundancy.
- **Encryption:**
  - Ensure backup data is encrypted both at rest and during transmission to protect from unauthorised access.
- **Version Control:**
  - Use **version control** on backups to maintain multiple restore points, helping with rollback to specific versions of data.

Tools for Backup and Recovery (Examples)

Database-Specific Tools:

- MySQL: mysqldump, MySQL Enterprise Backup.
- PostgreSQL: pg_dump, continuous archiving with pg_basebackup.
- SQL Server: Native tools for full, differential, and transaction log backups.

Cloud-Based Solutions:

- AWS Backup: Automated backups across AWS services.
- Google Cloud Backups: Managed backups for distributed databases (Cloud Spanner, Bigtable).
- Azure Backup: Scalable, cloud-native backup solutions.

Choosing Technologies across the Data Engineering Lifecycle

Considerations for Choosing Technologies

Architecture first, technology second

- Architecture is the high-level design, roadmap and blueprint of data systems to satisfy strategic aims.
- Tools are the tactical approaches to make the architecture a reality.

Some key considerations:

(can be remembered with the acronym ***MOBILIST***)

- **M**onolith/**M**odular
- **O**ptimising cost
- **B**uild vs buy
- **I**nteroperability
- **L**ocation
- **I**mmutability
- **S**peed to market
- **T**eam size

Example of Architecture before Technology: Enhanced Governance and Data Privacy

- Feature Stores are focusing on providing advanced governance, security, and compliance features. Managed feature stores like those from Microsoft include capabilities such as role-based access control (RBAC) and feature lineage tracking.

- Ensures that data pipelines comply with privacy regulations (e.g., GDPR, General Data Protection Regulation), and organisations can track and audit feature usage efficiently.

- Concrete technology or systems choices then later.

Monolith vs Modular Systems

Monolith

Features: a self-contained system

Pros: simplicity of everything in one place

(fewer moving parts, less context switching)

Cons: brittle and difficult to migrate away from

(updates/releases take longer, can be painful to escape

Modular

Features: decoupled, best-of-breed technologies communicating via APIs

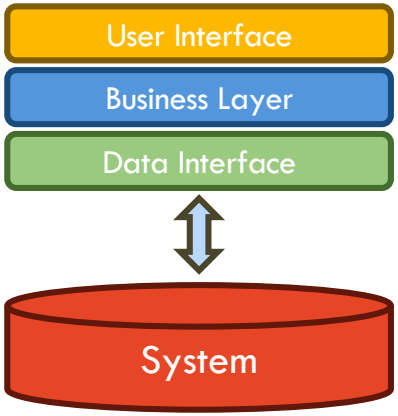
(each performing tasks at which they are uniquely great)

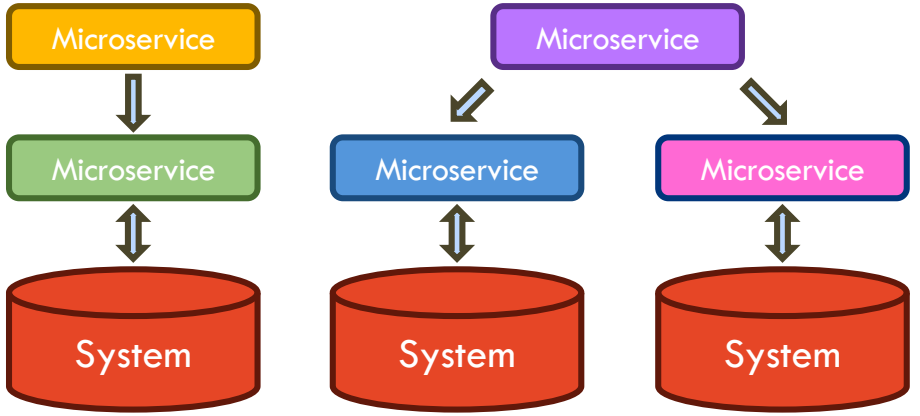
Pros: components are swappable

(ability to swap tools as technology changes is invaluable)

Cons: potentially countless systems

(poses complexities with maintaining them all)





Optimising Cost and Business Value

Budgets and time are finite, so cost is a major constraint

- Aim to optimise **Total Cost of Ownership (TCO)** and **opportunity cost**
- Learn to monitor costs to avoid surprises

Note distinction between expenses:

- **Capital expenditure** (capex) requires up-front investment
  (especially common before the cloud, with companies purchasing hardware/software up front with large acquisition contracts)
- **Operational expenditure** (opex) is gradual and spread out over time
  (much more flexible and common with the rise of cloud-based services)

Build vs Buy

Build

Pros: End-to-end control of the solution

Cons: Resource constraints and expertise

Worthwhile: If provides competitive advantage

Buy

Pros: Avoids "re-inventing the wheel"

Cons: At the mercy of a vendor / community

Worthwhile: Where system not key advantage

When purchasing, there are a few types of **software**:

- Open Source: Made available for general use, typically with specific licensing terms and at no cost
  - Community Managed (e.g. Spark): Consider its maturity, popularity, roadmap and documentation
  - Commercial (e.g. Databricks) : Consider its added-value, delivery model, pricing and support
- Independent: Proprietary solutions won't be as transparent, but can work quite well
  - Consider interoperability, documentation, pricing and longevity

Interoperability

Describes how systems **connect**, exchange information and interact

- Data pipelines typically use more than one technology.
- When evaluating two technologies A and B, how easily do those integrate?
  - Is seamless integration already baked into a technology or system?
  - If not, how much manual configuration will be required to integrate it?
- Design for modularity so that technologies can be swapped out later.

Systems often connect via **APIs**

For a publicly available example of what this can look like, feel free to investigate the Canvas API Documentation

Location

- Shift towards the cloud in recent decades.
- Principle places to run technology stack:
  - **On Premises:** company owns, and are operationally responsible for, their hardware and the software that runs on it (need to manage upgrade cycles every few years, ensure they have enough hardware to handle usage peaks, etc)
  - **Cloud:** company rents hardware and managed services from a cloud provider (e.g. AWS, Azure, Google Cloud), allowing engineers to quickly launch projects without long lead time hardware planning (resources can be reserved on extremely short-term basis, and can be dynamically scaled)
  - Approaches can be **hybrid cloud** (i.e. a mix of on-premise and cloud) or **multicloud** (deploying workloads across multiple cloud environments to leverage different benefits of different systems), though these should only be used if there's a compelling reason

Speed to Market

- In technology, speed to market wins.
  - For data pipelines, this means choosing the right technologies that help deliver features and data fast while maintaining high-quality standards and security.
- Perfect is the enemy of good – deliver value early and often.
  - Agile approaches help with tight feedback loop of launching, evaluating, iterating, and making improvements.
- Choose tools that help you move quickly, reliably and securely.

Apache Tools for Scalable Data Engineering

	Tool	Purpose	Execution	Scope
Distributed Storage & Processing	Apache Hadoop	Distributed storage & batch processing	Runs MapReduce jobs across a cluster	Large-scale storage (HDFS) & processing
	Apache Spark	Unified analytics engine for large-scale data	In-memory processing for both batch and streaming data	Fast data processing, ML, and analytics
Streaming & Real-Time Processing	Apache Flink	Stream and batch data processing	Low-latency processing for real-time data streams	Stream-first architecture with batch mode
	Apache Kafka	Distributed streaming platform	Real-time event streaming & message brokering	High-throughput distributed event streaming
Data Pipelines	Apache Beam	Unified model for batch and streaming pipelines	Executes on multiple runners (e.g., Flink, Spark, Dataflow)	Flexible pipelines for stream/ batch data
Orchestration & Workflow Management	Apache Airflow	Workflow orchestration and scheduling	Executes DAGs across workers to manage task dependencies	Scheduling & monitoring of data workflows
Dataflow Automation	Apache Nifi	Dataflow automation and management	Automates data movement between systems	Low-latency data integration
NoSQL Databases	Apache HBase	Distributed NoSQL database (columnar storage)	Provides real-time reads/writes for big data	Random access to large datasets

Some Example Solutions for Data Sharing

Solution	Description	Use Cases
Apache Arrow Flight	A high-performance data transport framework designed for large data transfers, supporting secure and fast communication.	Efficient data transport between systems and applications
Snowflake Data Sharing	Enables secure, governed sharing of live data with consumers inside or outside of an organization using Snowflake's platform.	Data sharing within Snowflake environments
AWS Data Exchange	A service to securely find, subscribe to, and share third-party data in the AWS ecosystem.	Sharing and subscribing to third-party data on AWS
Google Cloud BigQuery Data Transfer Service	Enables secure data movement from external data sources into BigQuery for analytics.	Transferring data into BigQuery for analysis
Azure Data Share	A fully managed service for securely sharing data between organizations in Azure, offering data governance and control.	Secure data sharing across Azure environments
Hadoop Distributed File System (HDFS)	Secure distributed storage for big data, supporting encryption and permissions to control access to shared data.	Data storage and sharing in big data environments
SFTP (Secure File Transfer Protocol)	Secure file transfer solution that provides encrypted data sharing between servers using SSH-based security.	File-based data sharing across systems

Google Cloud Storage Examples

Aspect	Cloud Spanner	Bigtable	Cloud SQL
Data Model	Relational, schema-based	NoSQL, column-oriented	Relational, schema-based (MySQL, PostgreSQL, SQL Server)
Query Language	Full SQL support (JOINs, indexes, ACID transactions)	No SQL, row key lookups, range scans	Full SQL support, ACID transactions, indexes, JOINs
Use Cases	Global transactional systems (e.g., financial, ERP)	Real-time analytics, IoT, time-series data	Traditional web apps, CMS, OLTP systems, small to medium databases
Transactions	Multi-row ACID transactions, strong consistency	Single-row transactions, eventual consistency	Multi-row ACID transactions, strong consistency
Scalability	Horizontally scalable, globally distributed	Petabyte scale, ultra-low latency	Vertically scalable (up to a point), limited horizontal scalability
Replication	Synchronous, multi-region, strong consistency	Asynchronous, regional or multi-region, eventual consistency	Synchronous/asynchronous replication, regional availability
Performance	Low-latency transactions, suited for consistency	Ultra-low latency, suited for high-throughput workloads	Moderate performance, suitable for transactional workloads
Cost	Higher due to complex transactions and global consistency	More cost-effective for massive, non-relational datasets	Cost-effective for smaller databases, higher costs for large deployments
Availability	Global replication with strong consistency	Regional replication, eventual consistency	Regional availability, optional high availability configurations

Team Size and Capabilities

- Previous slides gave an overview of some of today's technologies (also see Matt Turck's Machine Learning, AI and Data landscape (cf. Week 1), https://www.mattturck.com/mad-landscapes)
- But remember: **Architecture first, technology second**

- Choices not only have to fit the architecture and any internal technical constraints, but also need to match a team's technical skills and capabilities.
- Team's size determines bandwidth that can be dedicated to complex solutions.
- Especially for small teams, it helps to use familiar technologies and workflows, or SaaS (Software as a Service) solutions.
- Investing time in learning a new technology must have a clear benefit and goal, otherwise risks of never been used in production.

Summary and Outlook

THE UNIVERSITY OF SYDNEY

Summary

- Security should be on the top of the mind when designing a data architecture or building new data pipelines
- Treat data as if it is your own wallet or smartphone
- Check relevant data privacy regulations and manage data accordingly
- Know basic security practices

Next Steps

- Week 12: Tutorial Quiz 2
  - **individual individual assessment worth 5%** of COMP5339
  - in-person, written quiz during your assigned tutorial time. Duration: 30 minutes. **You have to attend your tutorial this week to sit the quiz.**
- Week 13: UoS Review and Examination Tips
- End of Week 13: Submission deadline for Assignment 2
  - due on **Friday 7 November, 23:59**

- Final Exam: Friday 21 November 2025, 1pm
- USS Feedback form: https://student-surveys.sydney.edu.au