
# Fundamentals of Data Engineering

# Scenario

**Every stakeholder interacts with the data ecosystem differently: Business executives want results and trends. Data scientists need structured data to build models. Developers need usable interfaces (APIs).Customers want value and privacy. You, the data engineer, ensure the whole data flow works efficiently and ethically.**

Data Engineer

Wasting time & resources!


**System Requirements**

| Functional Requirements | Nonfunctional Requirements |
|---|---|

**Technology Choices**

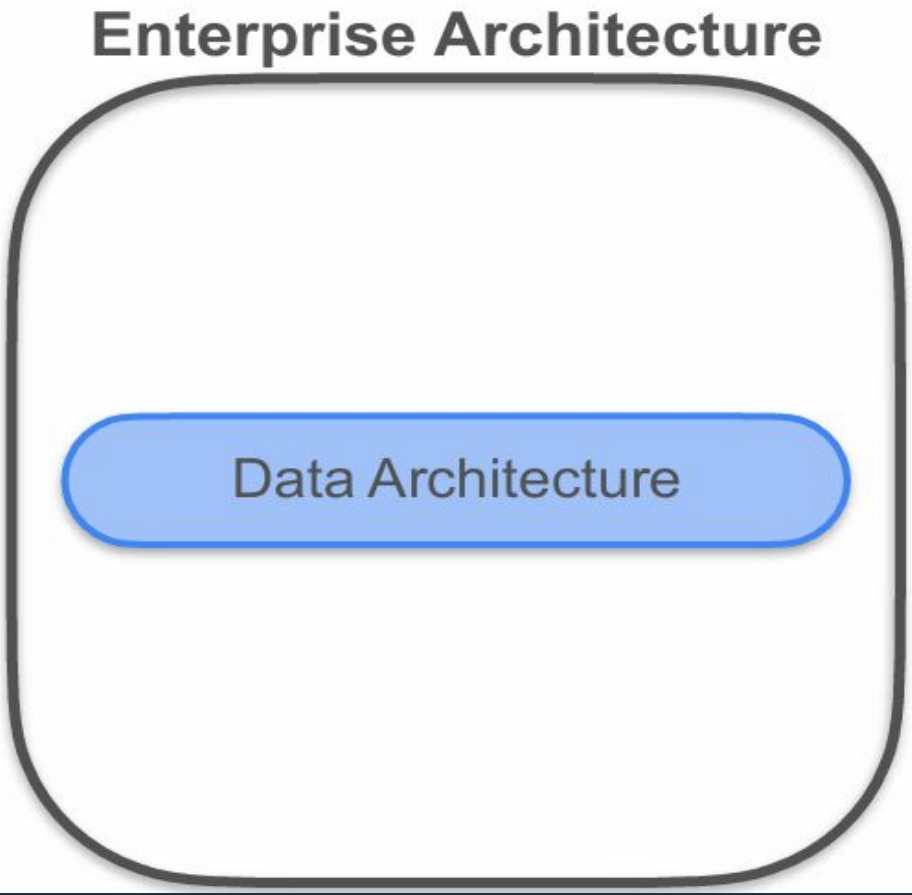
- Ingestion tool
- Storage system
- Transformation tool

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

# Module-2

1. Data Architecture,
2. Streaming Architectures,
3. Choosing the Right Technologies,
4. Stakeholder Management and Gathering Requirements, Translating Requirements to Architecture,
5. Implement Batch Pipeline to Serve Training Data,
6. Databricks

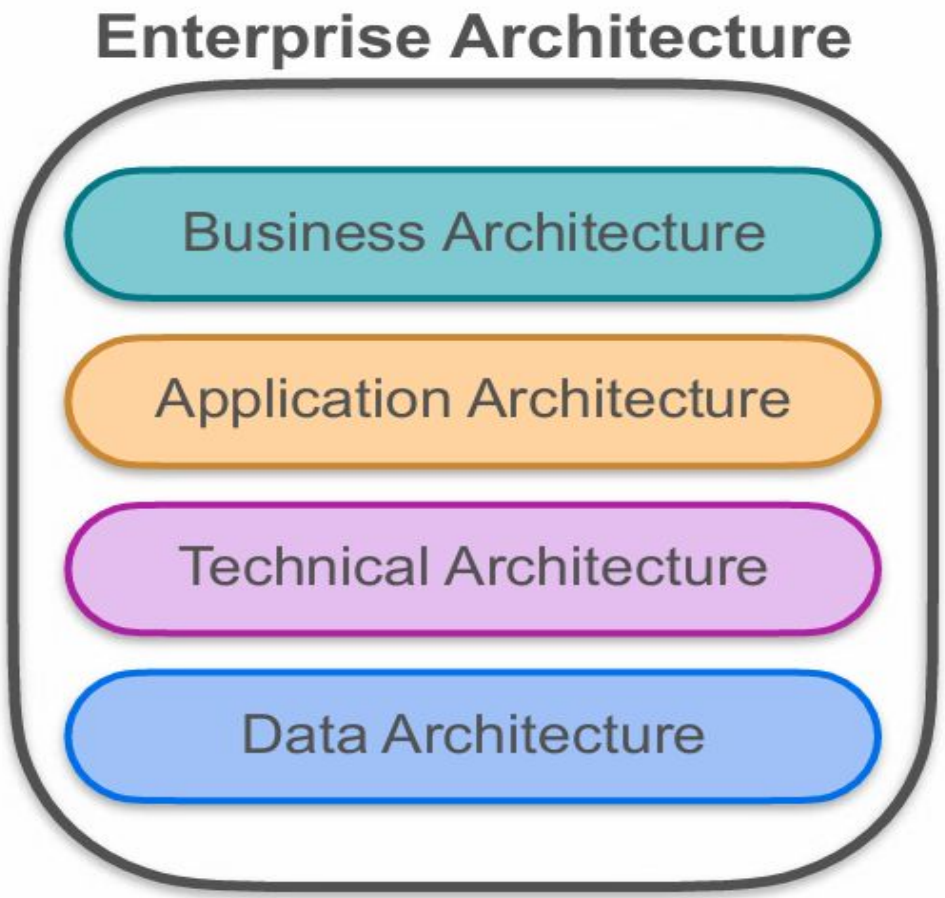
# Enterprise Architecture



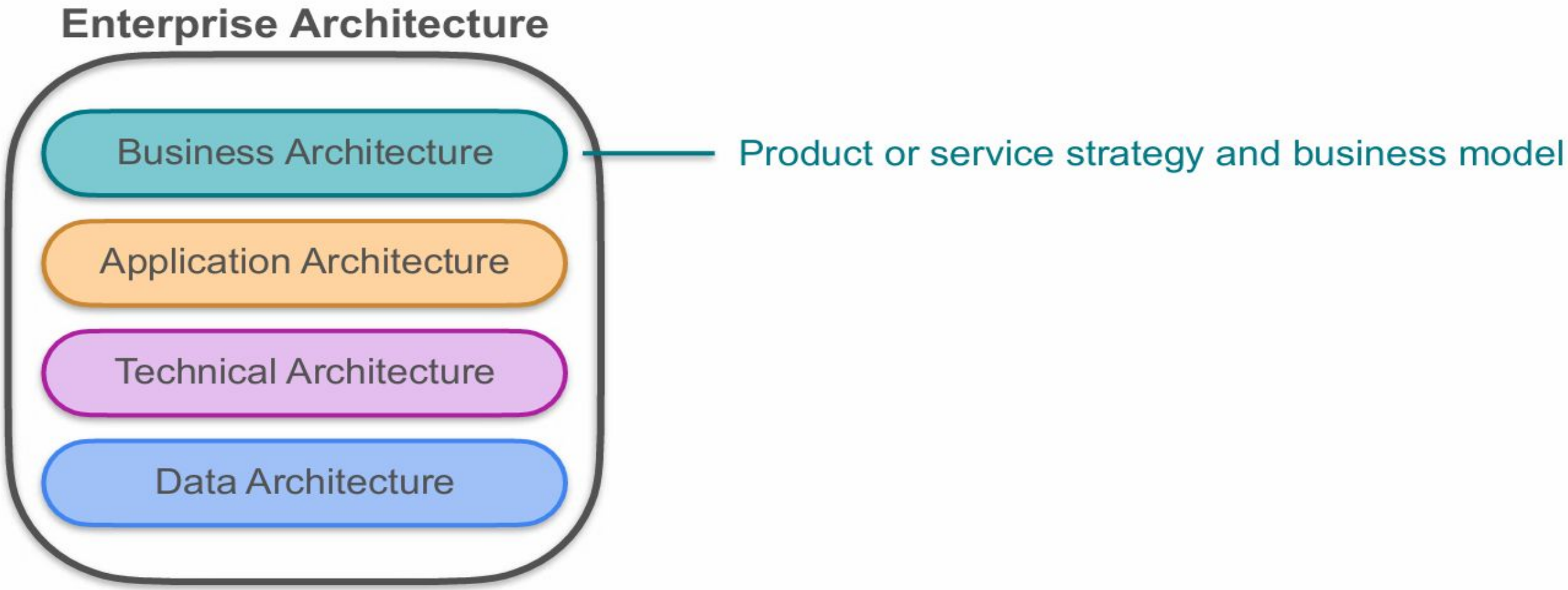

"the design of systems **to support change in an enterprise**, achieved by flexible and reversible decisions reached through a careful evaluation of trade-offs"

"the design of systems **to support the evolving data needs of an enterprise**, achieved by flexible and reversible decisions reached through a careful evaluation of trade-offs."

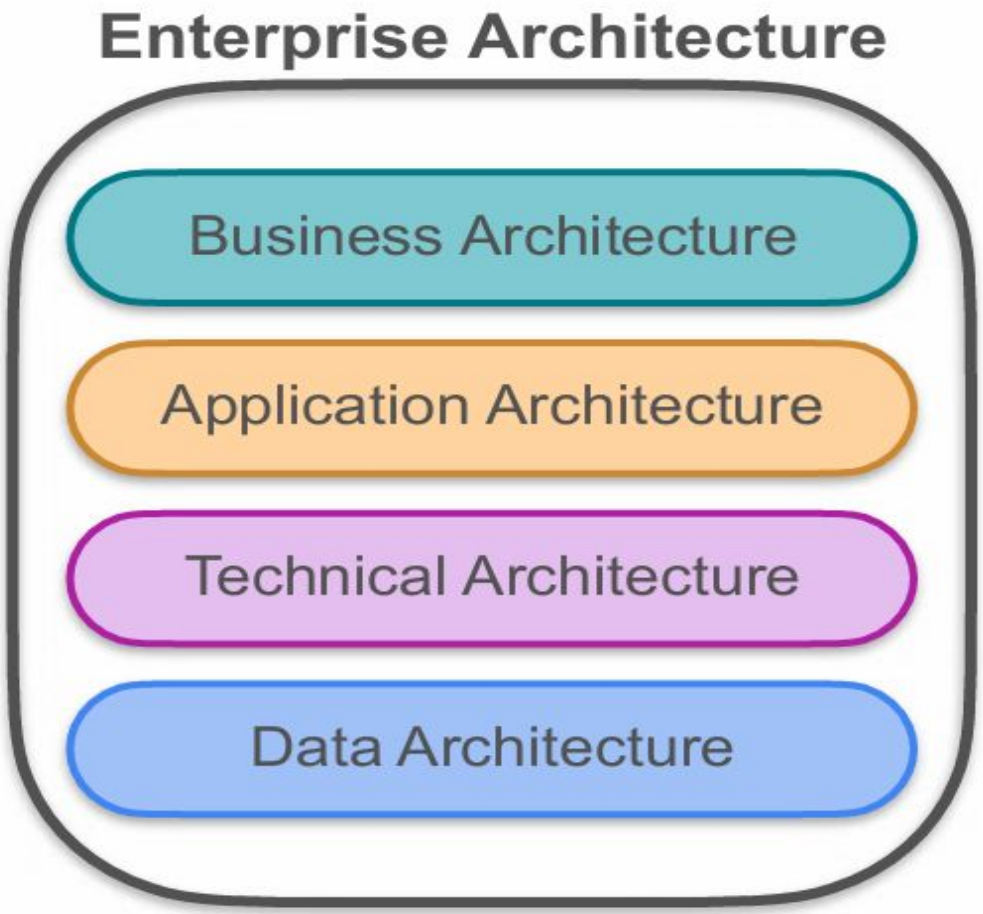
# Enterprise Architecture



# Enterprise Architecture

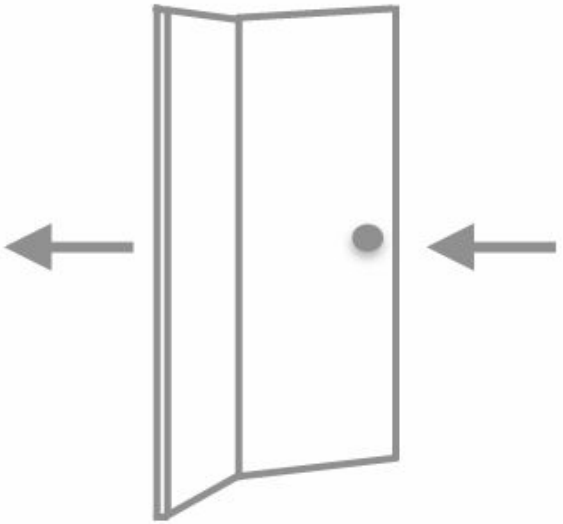


# Enterprise Architecture

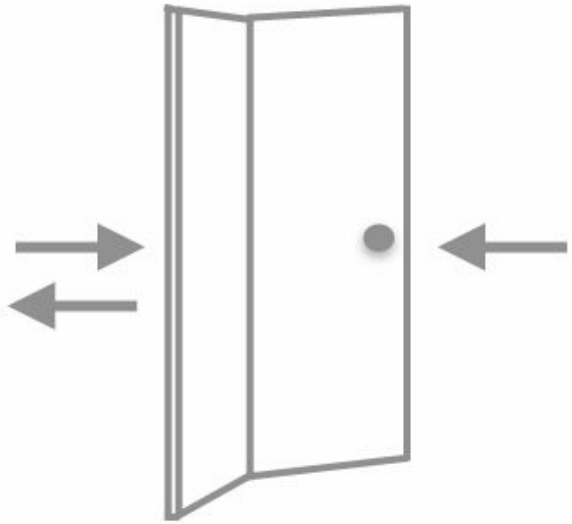



# One-Way and Two-Way Door Decisions

## Organization Decisions



One-way



Two-way

# One-Way and Two-Way Door Decisions

**Organization Decisions**

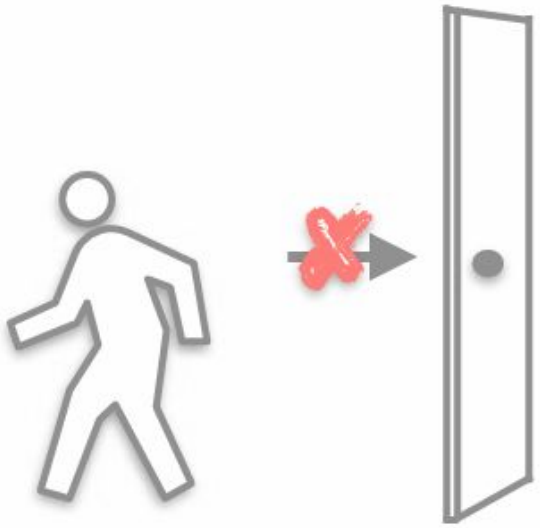
**One-way**

A decision that is almost impossible to reverse

**Two-way**

# One-Way and Two-Way Door Decisions

**Organization Decisions**



**One-way**

A decision that is almost impossible to reverse



**Two-way**

# One-Way and Two-Way Door Decisions

## Organization Decisions

**One-way**

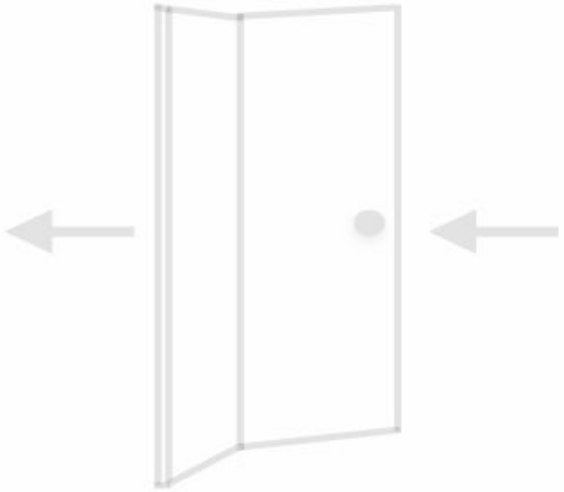
A decision that is almost impossible to reverse



**Two-way**

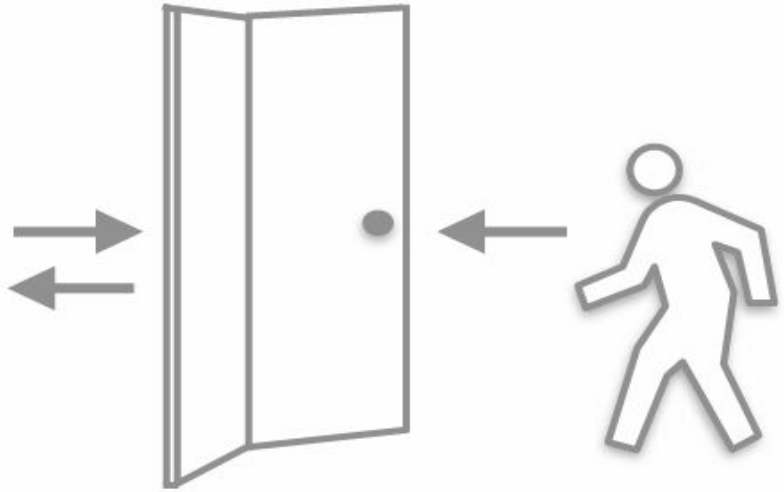
# One-Way and Two-Way Door Decisions

## Organization Decisions



**One-way**

A decision that is almost impossible to reverse

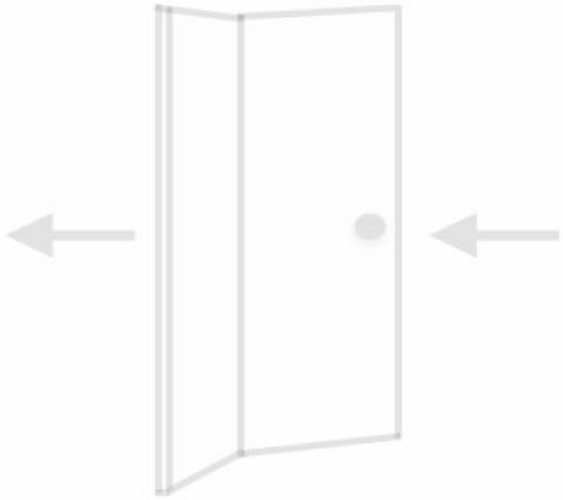


**Two-way**

An easily reversible decision

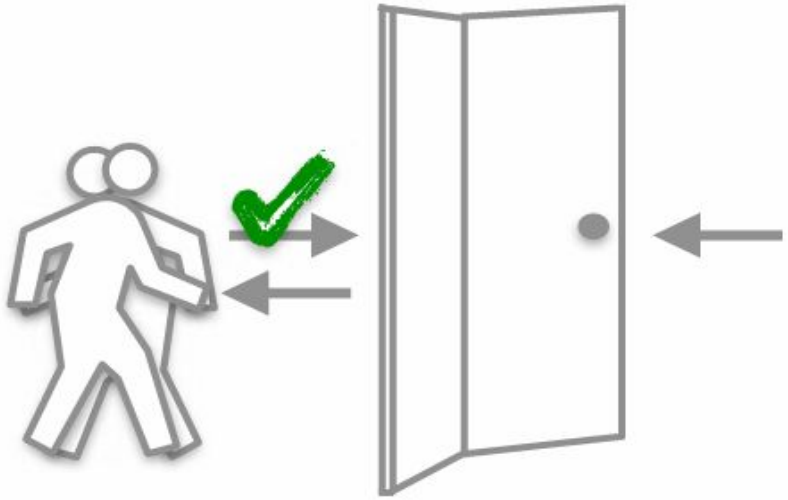
# One-Way and Two-Way Door Decisions

**Organization Decisions**



**One-way**

A decision that is almost impossible to reverse

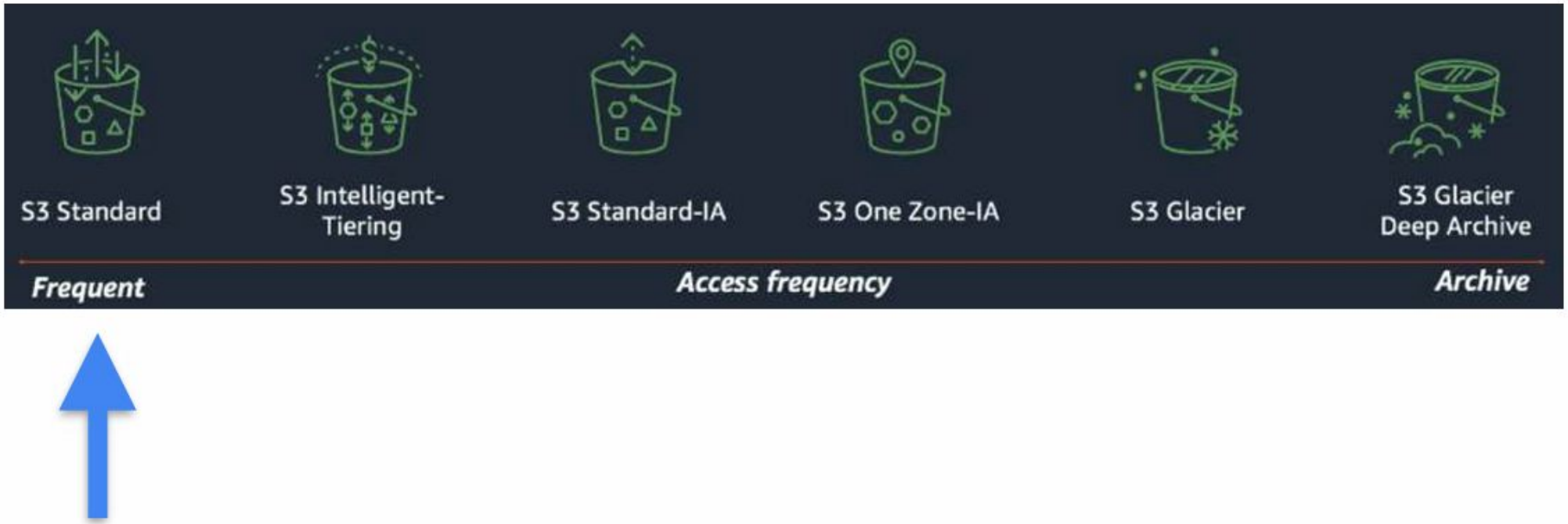


**Two-way**

An easily reversible decision

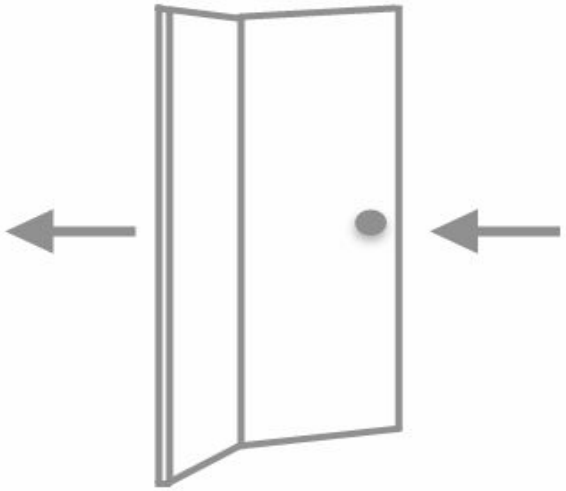
# Two-Way Door Decision

An example of a two-way door decision would be if you chose to store your data in the standard storage class in S three. Then later, if your storage needs change, you can transition to any other storage class for a fee. So, this decision is reversible.

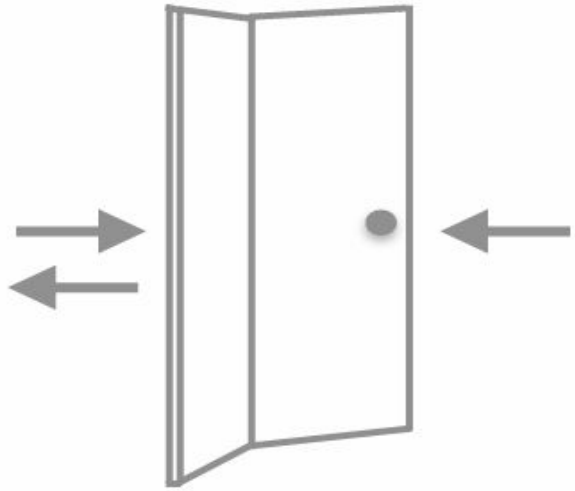


# Reversible Decisions

## Organization Decisions



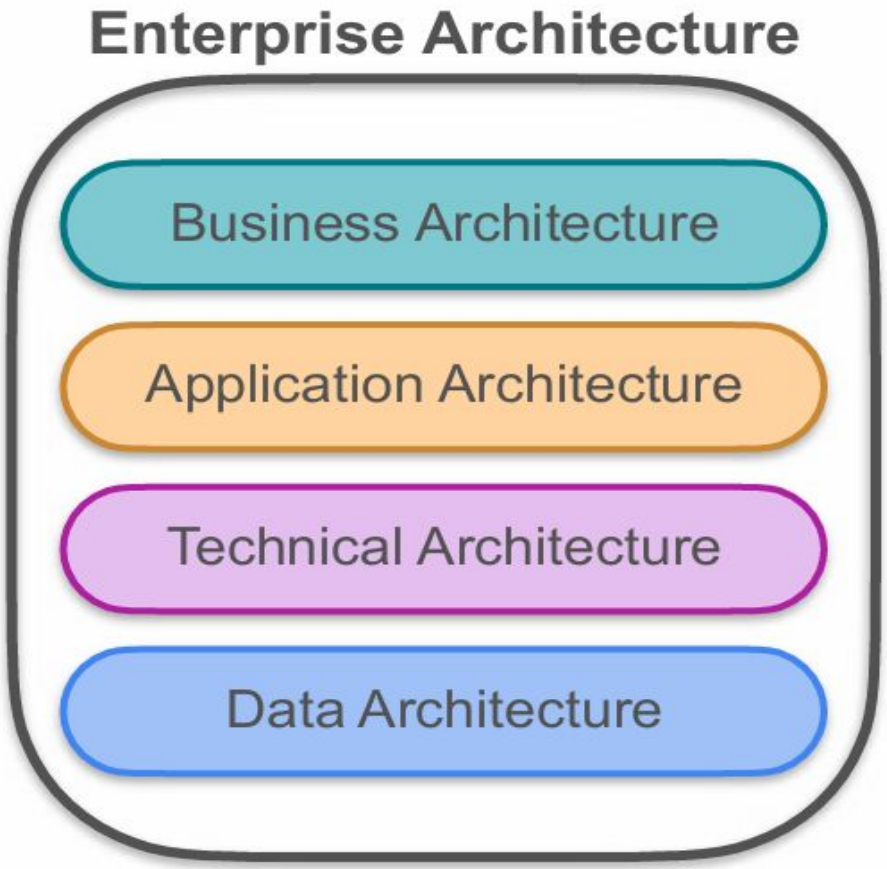
One-way



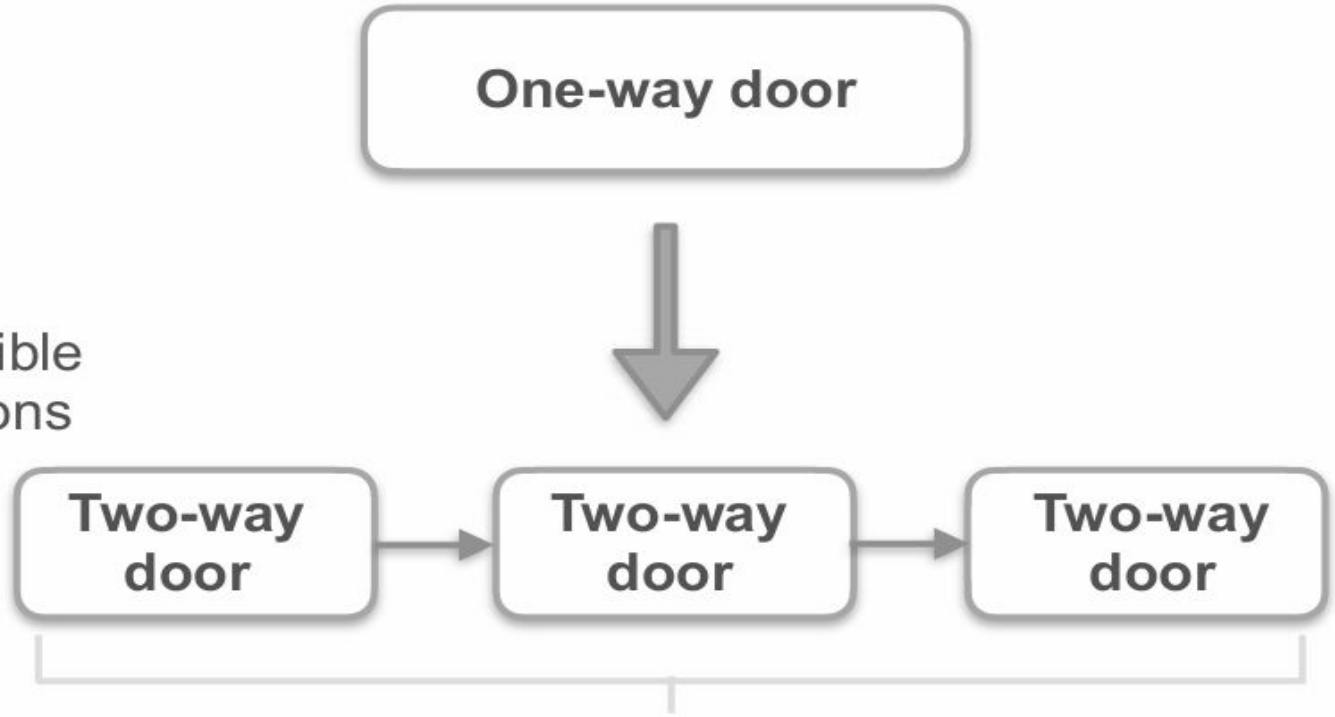
Two-way

Reversible Decisions

# Reversible Decisions




Reversible Decisions



# Conway's Law

"Any organization that designs a system will produce a design whose structure is a copy of the organization's communication structure."

- Melvin Conway

# Conway's Law

"Any organization that designs a system will produce a design whose structure is a copy of the organization's communication structure."

# Principles of Good Data Architecture

1. Choose common components wisely
2. Plan for failure!
3. Architect for scalability
4. Architecture is leadership
5. Always be architecting
6. Build loosely coupled systems
7. Make reversible decisions
8. Prioritize security
9. Embrace FinOps

# Principles of Good Data Architecture

1. Choose common components wisely
4. Architecture is leadership

**How data architecture impacts other teams and individuals**

5. Always be architecting
6. Build loosely coupled systems
7. Make reversible decisions

**Data architecture is an ongoing process**

2. Plan for failure!
3. Architect for scalability
8. Prioritize security
9. Embrace FinOps

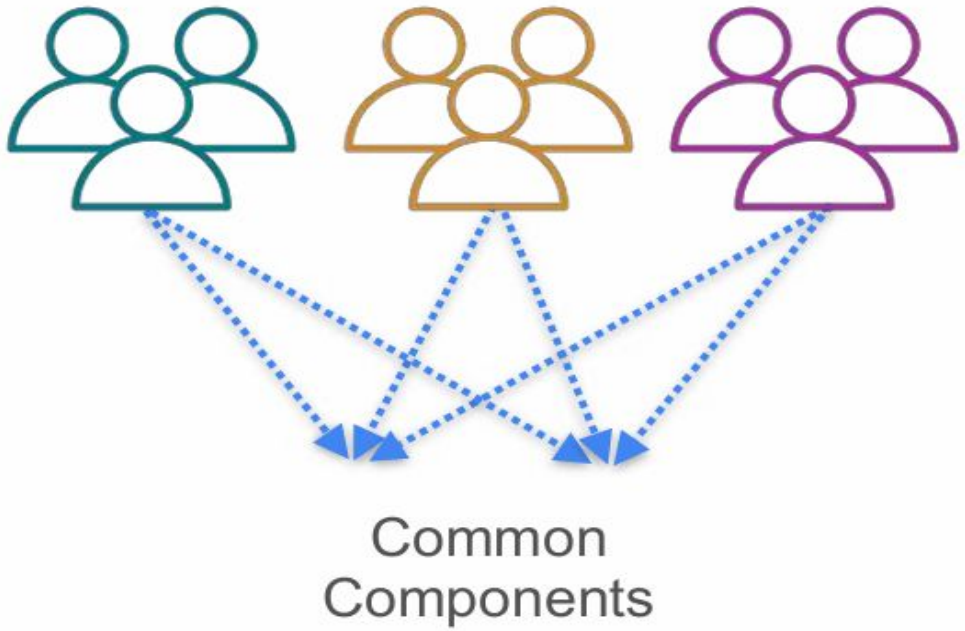
**Unspoken but understood priorities**

# Principles of Good Data Architecture

1. Choose common components wisely
4. Architecture is leadership

**How data architecture impacts other teams and individuals**

# Common Components




**Examples**

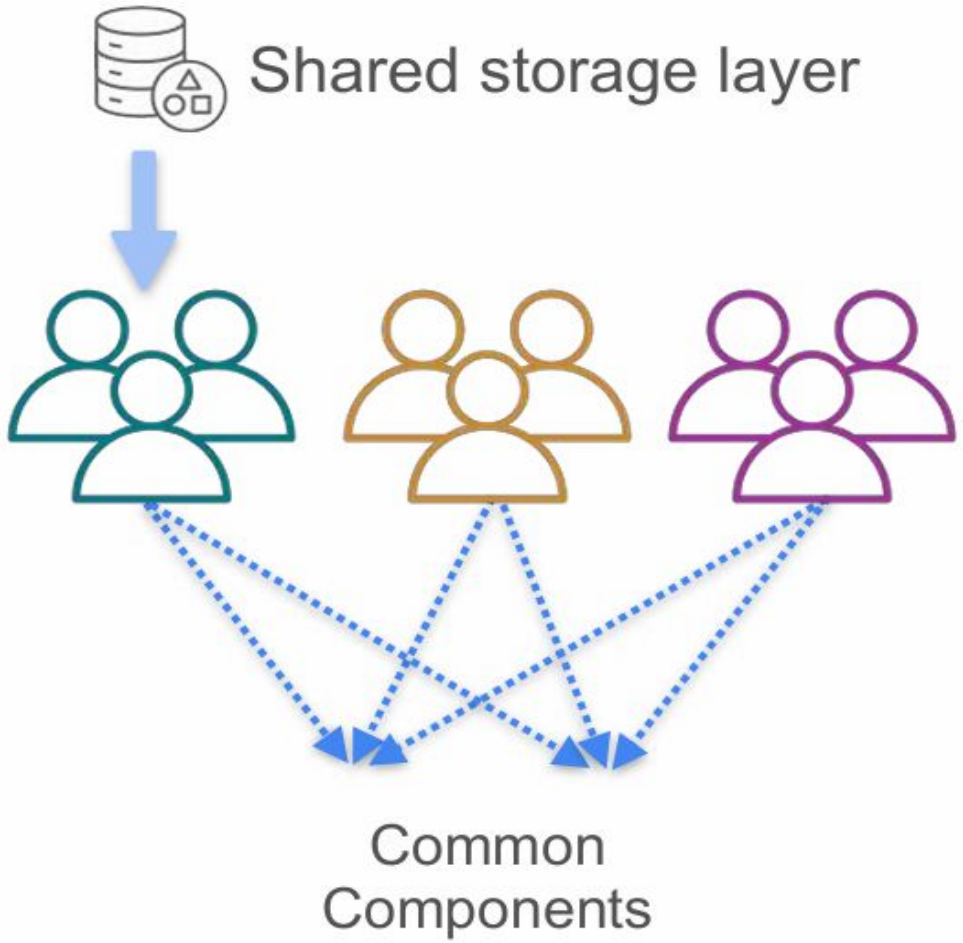
Object Storage

Version-Control Systems

Observability & Monitoring Systems

Processing Engines

# Common Components



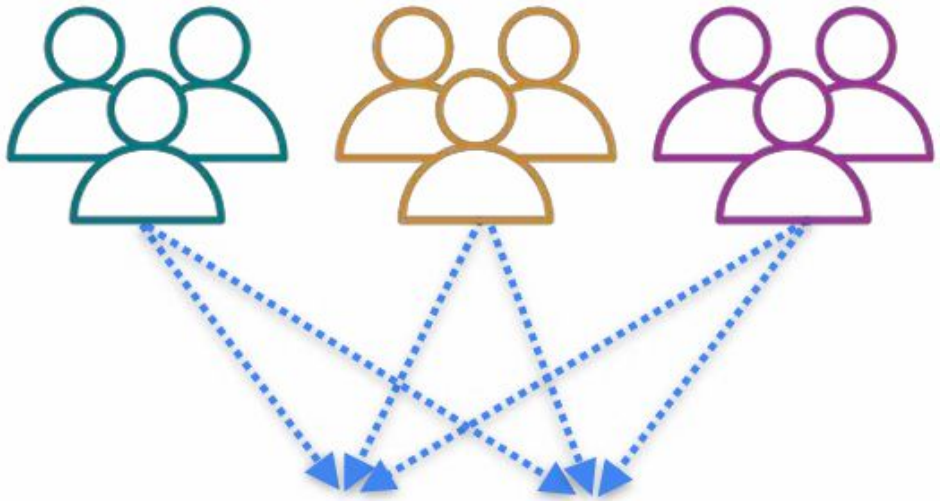

**Common components:**

- Facilitate team collaboration
- Break down silos

**“Wise” Choice:**

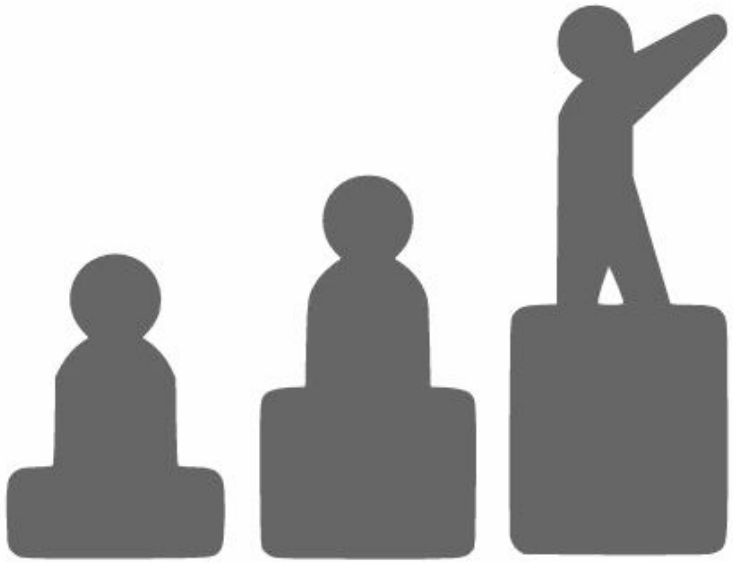
- Identify tools that benefit all teams
- Avoid a one-size-fits-all approach

# Architecture Leadership



Common Components

**Architecture is leadership**



Seek mentorship from data architects

# Application Programming Interface


**Amazon's API mandate required all teams to communicate through service interfaces (APIs), ensuring stable, predictable interactions regardless of internal complexity**

# Application Programming Interface

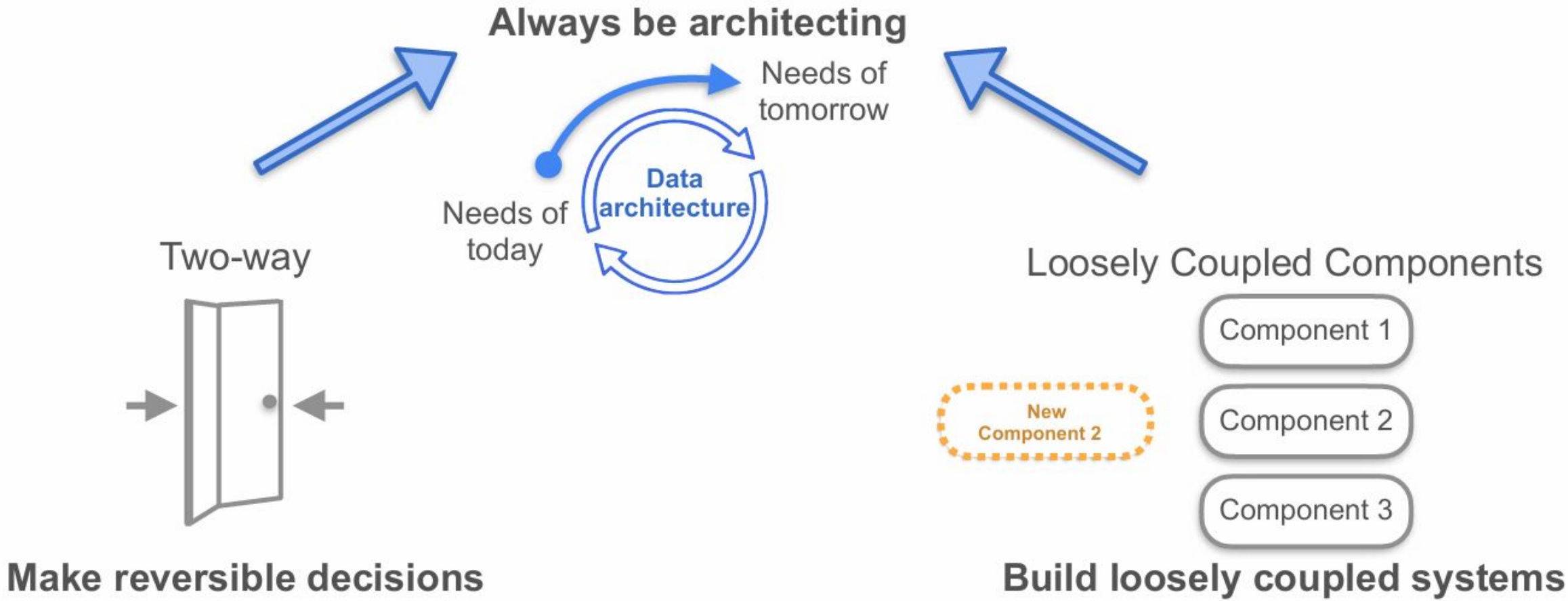
# Always Architecting

**Make reversible decisions**

**Build loosely coupled systems**

**Always be architecting**

# Always Architecting



# When Your Systems Fail

**Plan for failure**

**Architect for scalability**

**Prioritize security**

**Embrace FinOps**

# Plan for Failure

**Practical and Quantitative Approach**

System metrics:

**Availability** **Reliability** **Durability**

# Plan for Failure

**Availability**

The percentage of time an IT service or a component is expected to be in an operable state.

Amazon S3

| Examples of S3 Classes | S3 One Zone-IA | S3 Standard |
|---|---|---|
| Availability | 99.5% | 99.99% |
| Annual Downtime in hours | 44-hour downtime | 1-hour downtime |

# Plan for Failure

**Durability**

The ability of a storage system to withstand data loss due to hardware failure, software errors, or natural disasters.

Amazon S3

**Durability**

99.999999999%

(11 nines)

# Plan for Failure

**Reliability**

The probability of a particular service or component performing its intended function during a particular time interval

# Plan for Failure

**The maximum acceptable amount of time it should take to restore a system or application after a failure.** If your RTO is **2 hours**, your system must be back online within 2 hours after an outage.

**Recovery Time Objective**
**RTO**

The maximum acceptable time for a service or system outage

For example: Consider the Impact to customers

**Recovery Point Objective**
**RPO**

A definition of the acceptable state after recovery

For example: Consider the maximum acceptable data loss

# Plan for Failure

**The maximum acceptable amount of data loss measured in time.** If your RPO is 5 minutes, backups or replication must ensure you never lose more than 5 minutes of data.

**Recovery Time Objective**
**RTO**

ne maximum acceptable time for a service or system outage

For example: Consider the Impact to customers

**Recovery Point Objective**
**RPO**

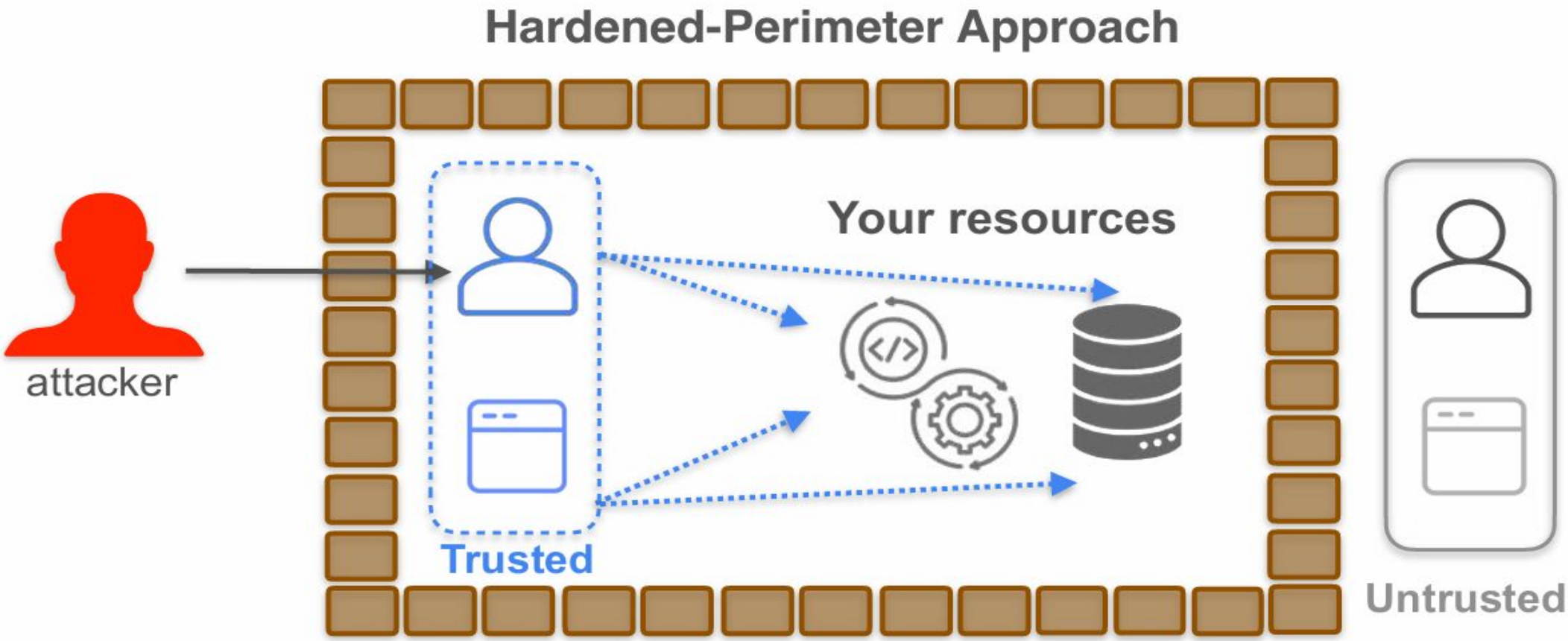
A definition of the acceptable state after recovery

For example: Consider the maximum acceptable data loss

# Prioritize Security

- Culture of security
- Principle of least privilege
- Zero-trust security

# Prioritize Security



# Prioritize Security

# Architecting for Scalability & Embrace FinOps

**Large Unforeseen Costs**

OR

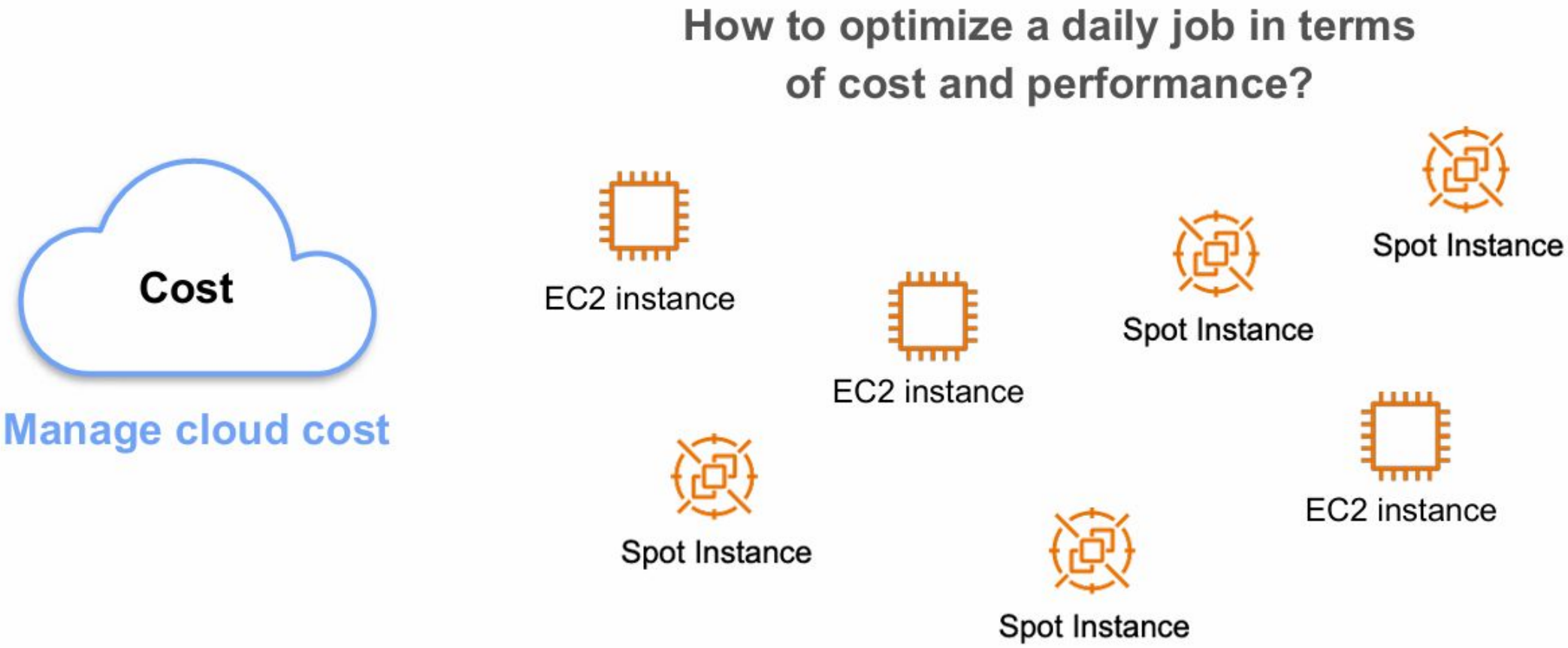
**Missed Opportunities for revenues**


**Plan for Failure!**

**Embrace FinOps**

**Architect for scalability**

# Embrace FinOps



# Embrace FinOps

**Pay-as-you-go models:**

- Cost-per-query model
- Cost-per-processing-capacity model


**Manage budgets, priorities and efficiency**

**Readily Scalable:**

# When Your Systems Fail

Plan for failure

Architect for scalability

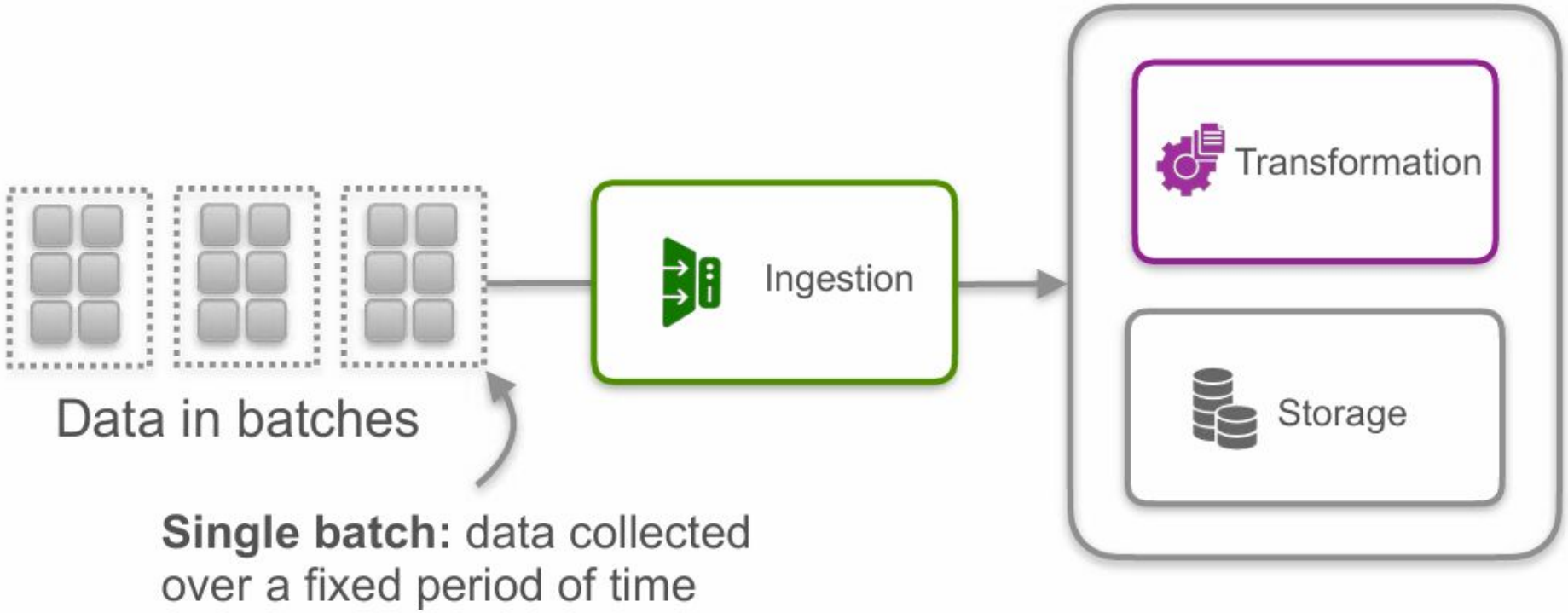
Prioritize security

Embrace FinOps

Serve the needs of your organization

# Batch Data Architecture

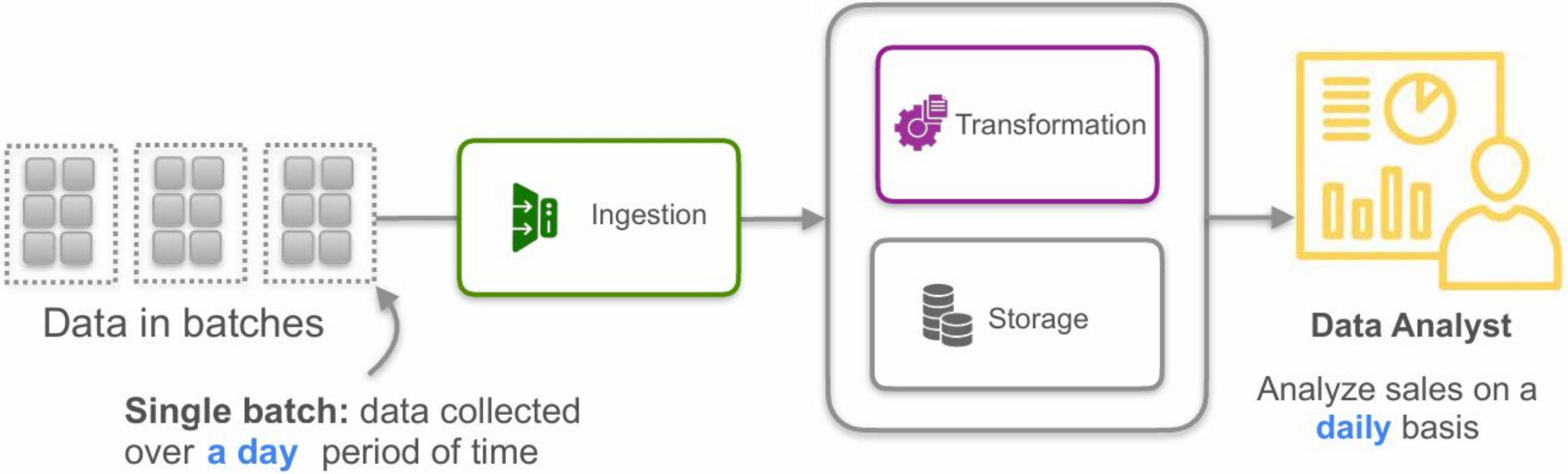
Real-time analysis is not critical



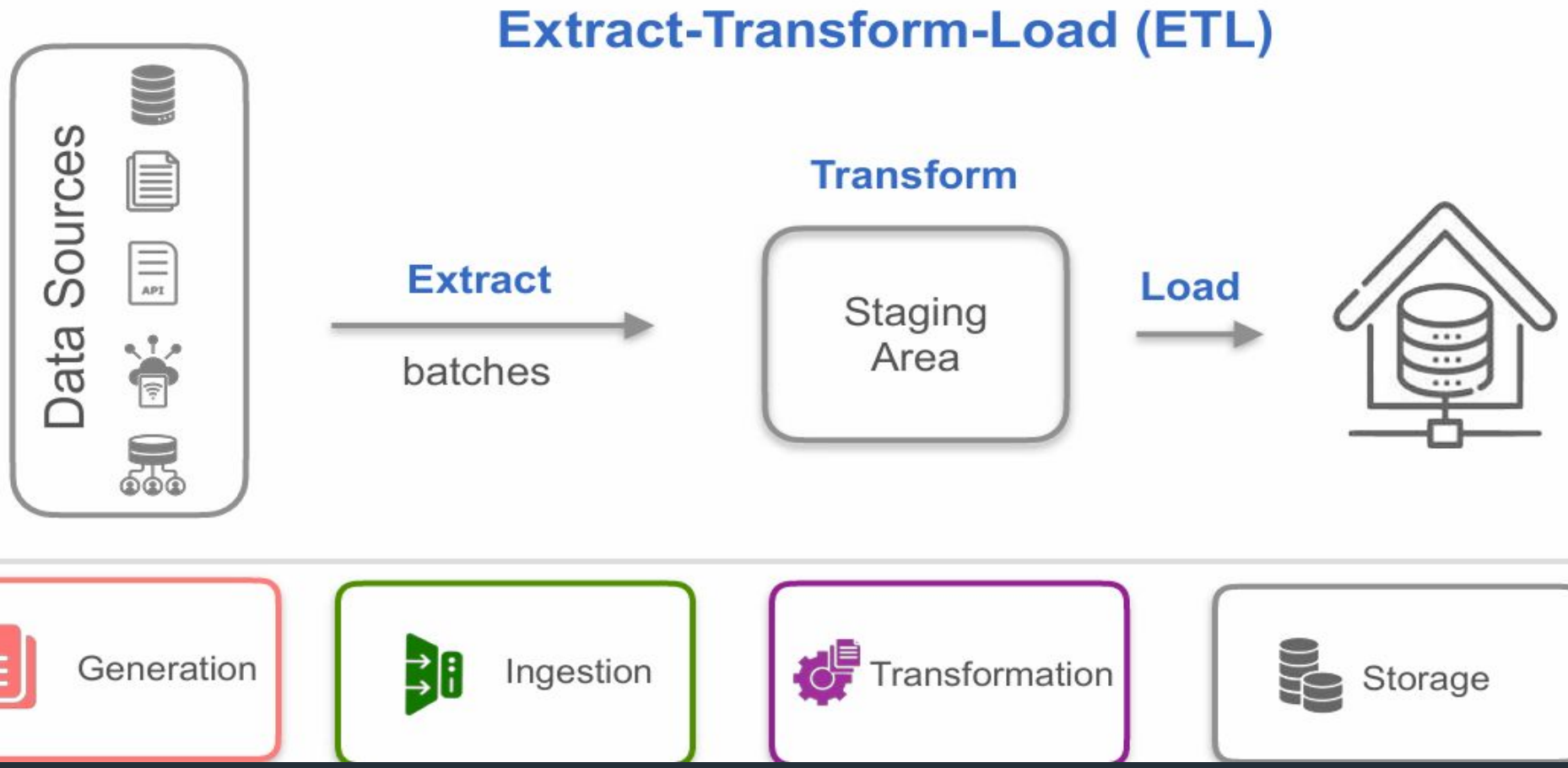

**Single batch:** data collected over a fixed period of time

# Batch Data Architecture

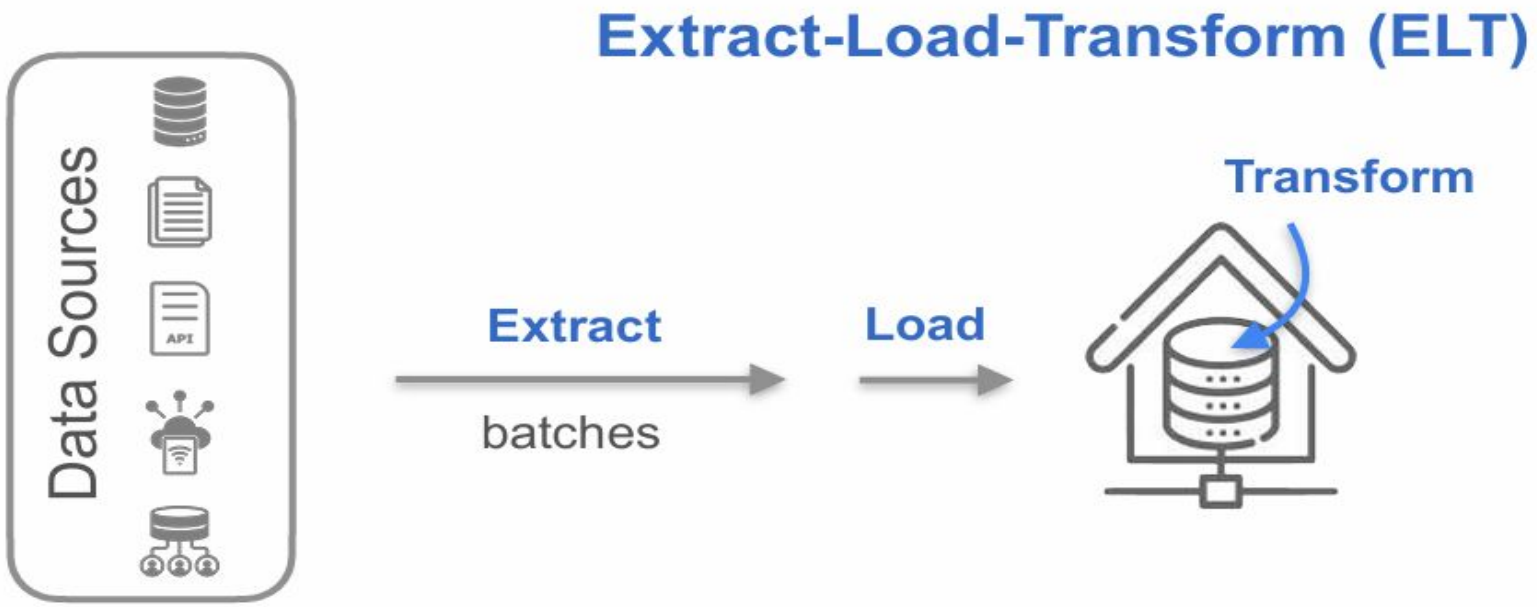
Real-time analysis is not critical



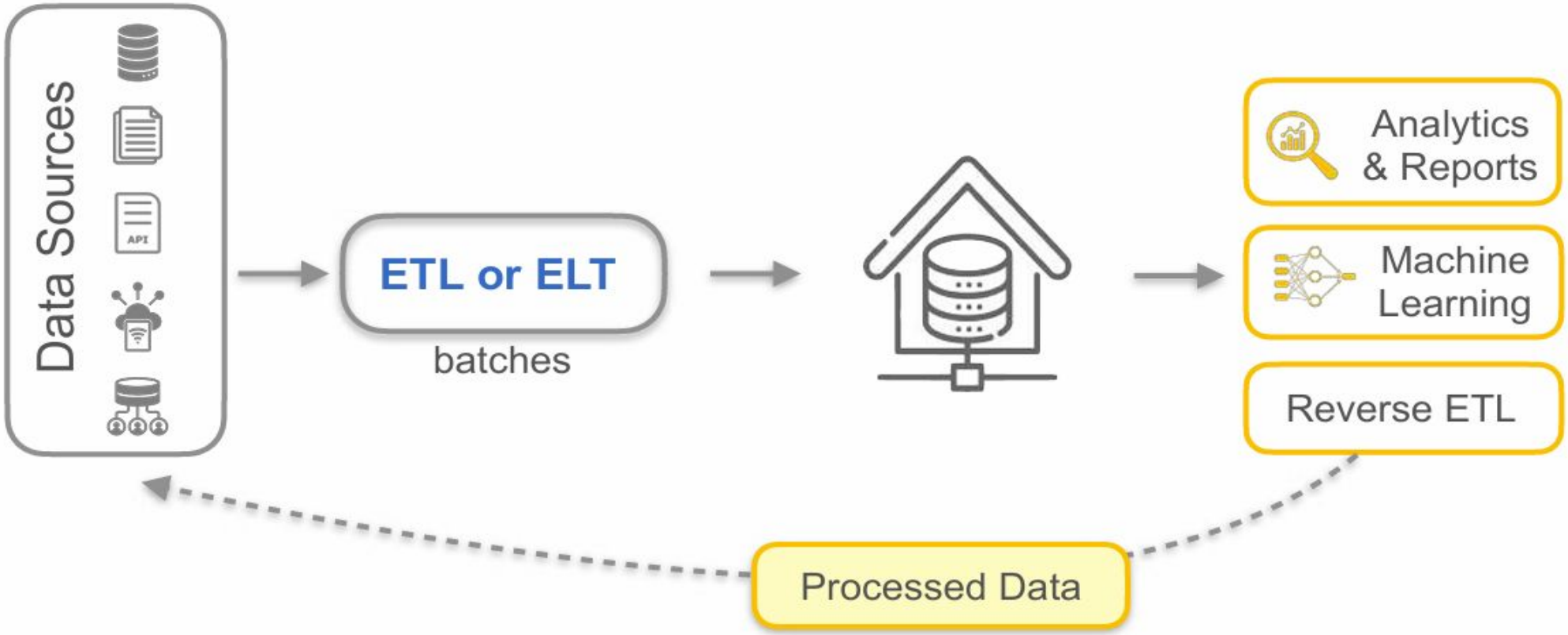
# Example of Batch Architectures



# Example of Batch Architectures

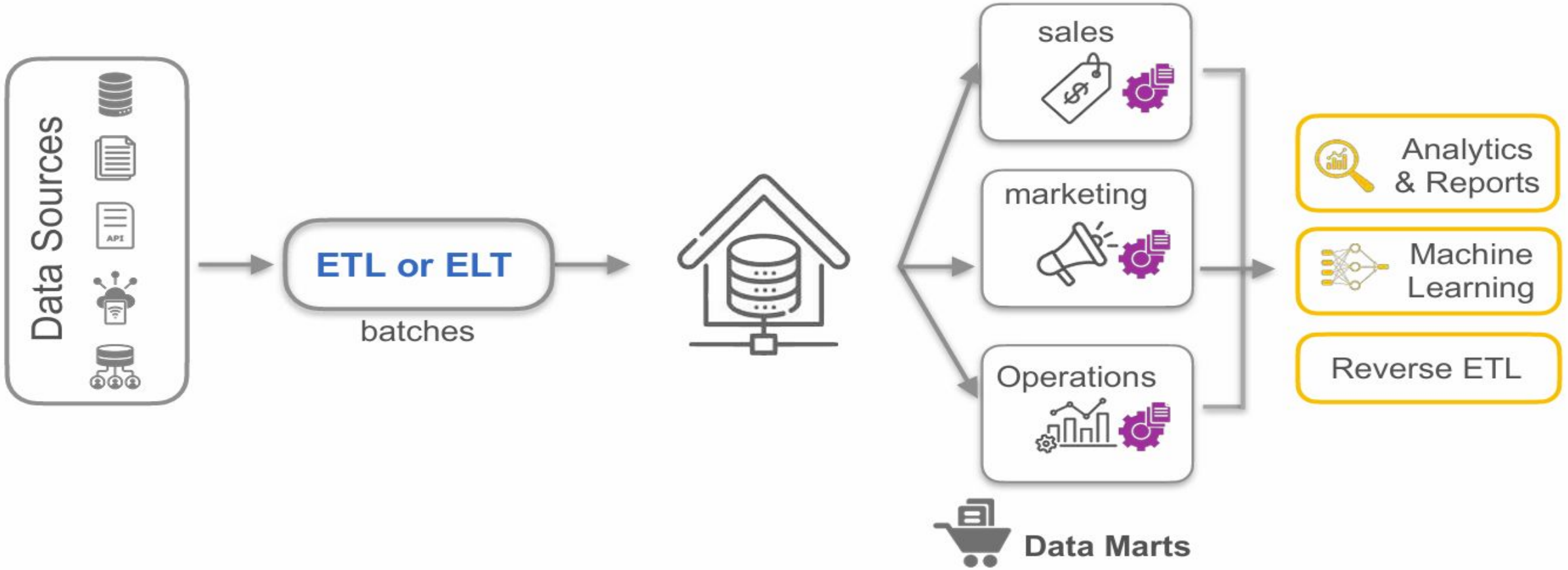



# Example of Batch Architectures

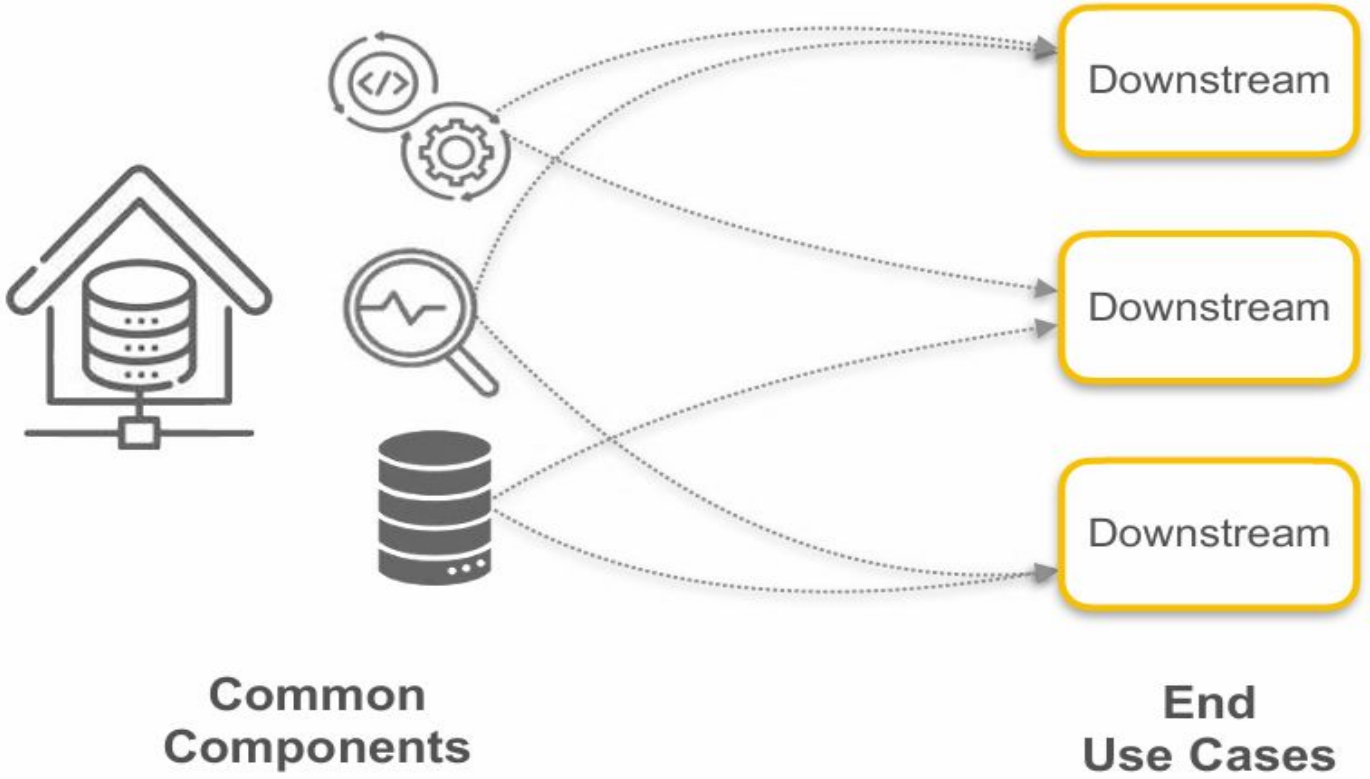


**From the main data warehouse, smaller data marts can be built.**
**Sales Mart** → Sales revenue, discounts, order history.
**Marketing Mart** → Campaign performance, leads.
**Operations Mart** → Shipping times, returns, warehouse stock.
**This makes reports faster and tailored for each department.**

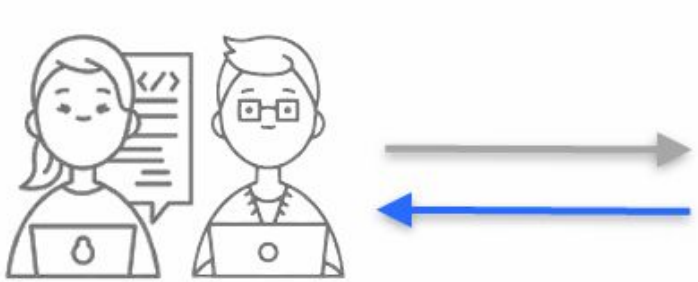
# Example of Batch Architectures



# Choose Common Components



# Planning for Failure

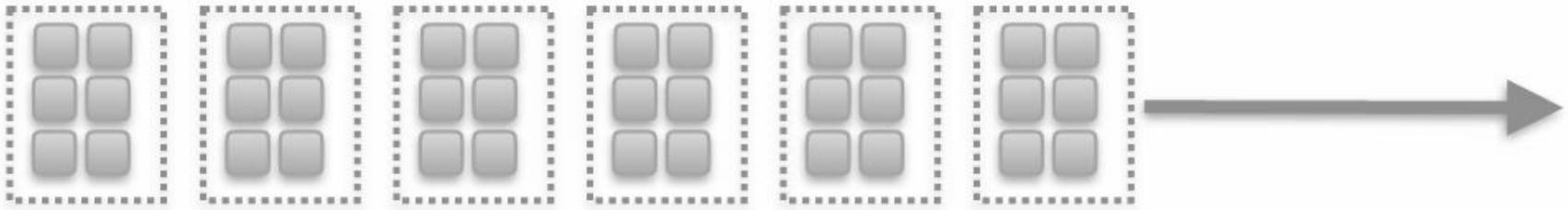
Source System Owners

Data Engineer

**Availability and reliability specs**

# Make Reversible Decisions

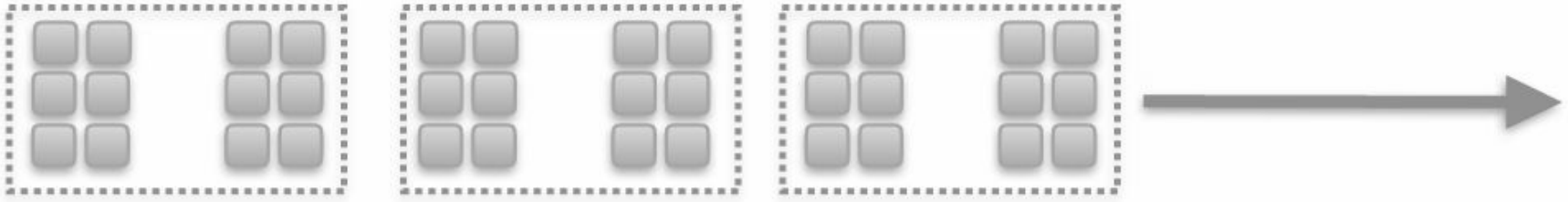
**Build flexibility into your system**



Ingest 1 day's worth of data

# Make Reversible Decisions

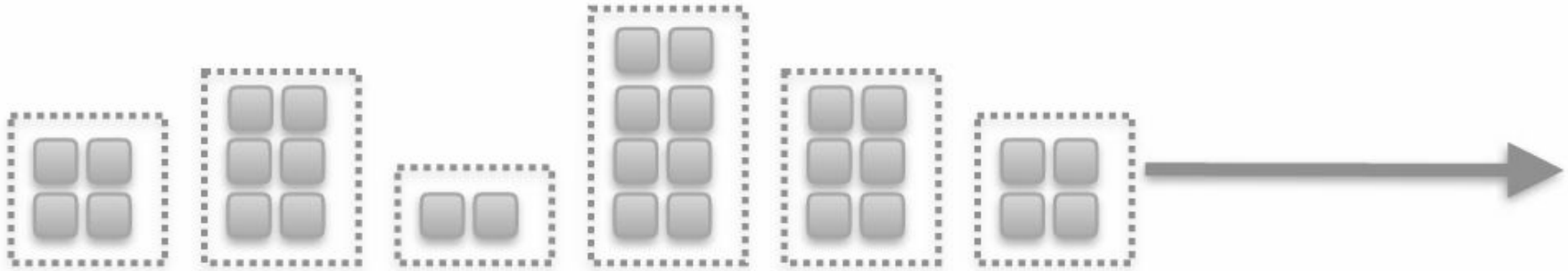
**Build flexibility into your system**



Ingest 2 day's worth of data

# Make Reversible Decisions

**Build flexibility into your system**



Ingest different
amounts of data

# Embrace FinOps

## Cost-Benefit Analysis

| | Component 1 | Component 2 |
|---|---|---|
| Performance | High | Low |
| Cost | High | Low |

Value for the business

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

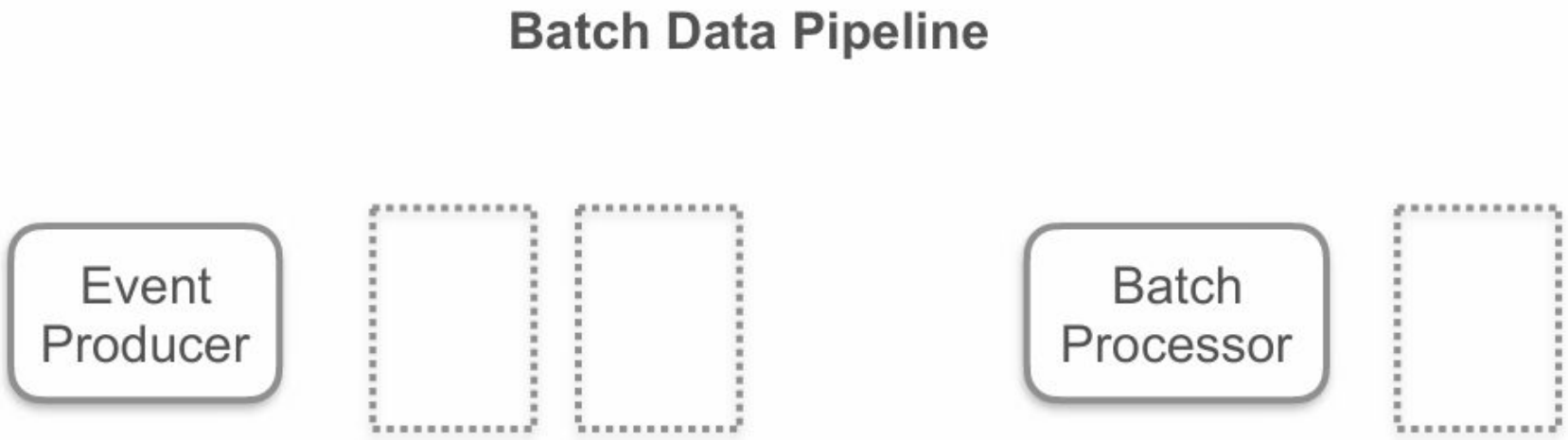
# Streaming Architectures

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

# Data - Stream of Events

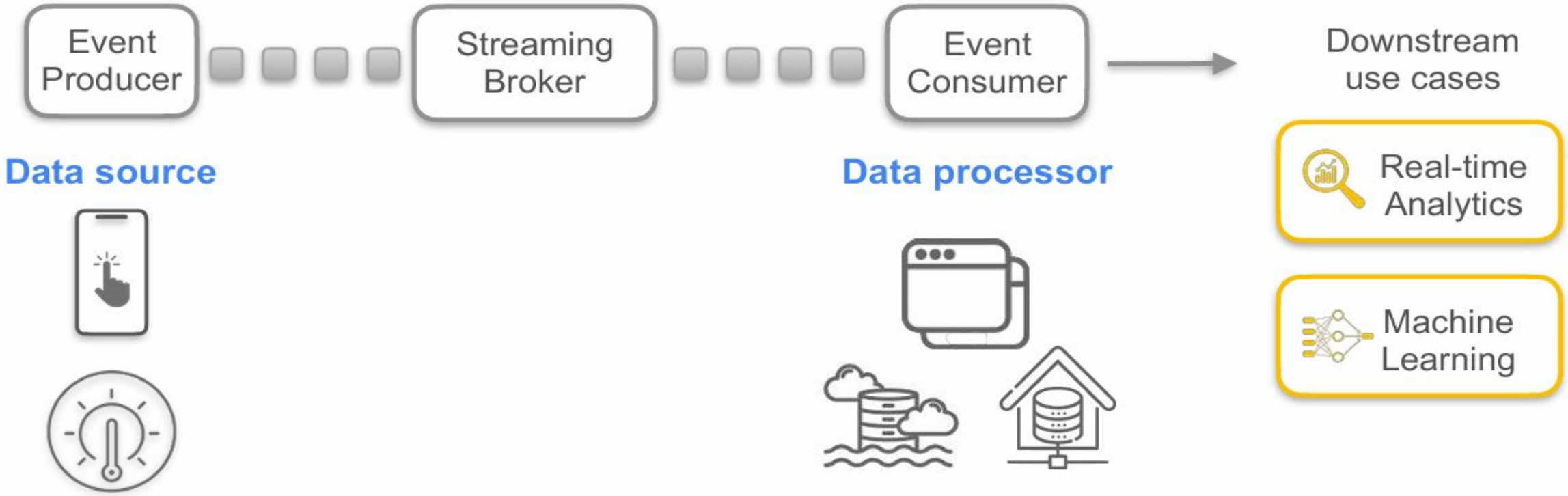
**At its source, data is a continuous stream of events**

# Data - Stream of Events



# Data - Stream of Events

# Streaming Systems



# Streaming Frameworks


Event Streaming Platform

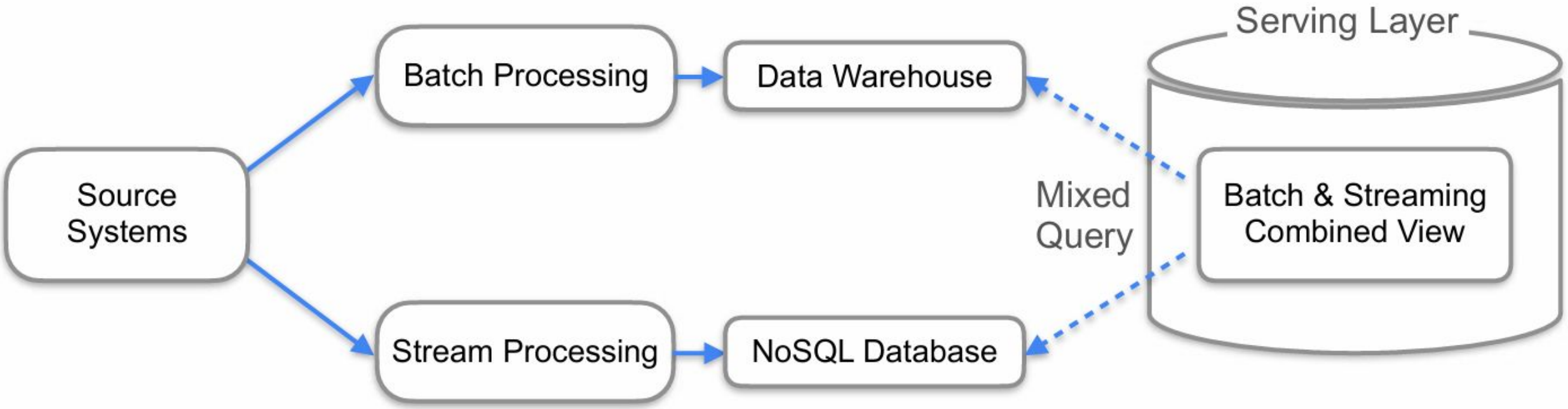


Streaming & Real-time Analytics

Batch + Real-time (dual pipelines)

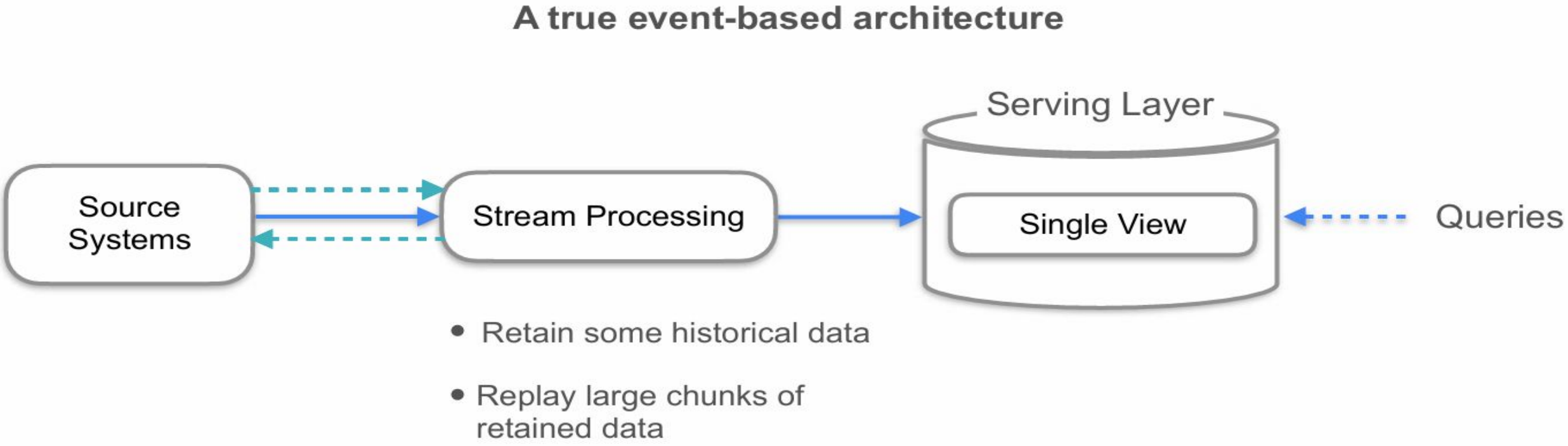
# Lambda Architecture

**Challenges: managing parallel systems with different code bases**



Kappa architecture is a data processing approach that simplifies data pipelines by handling both real-time and batch processing through a single, stream-based system. “All data is a stream; batch is just a bounded stream.”

# Kappa Architecture



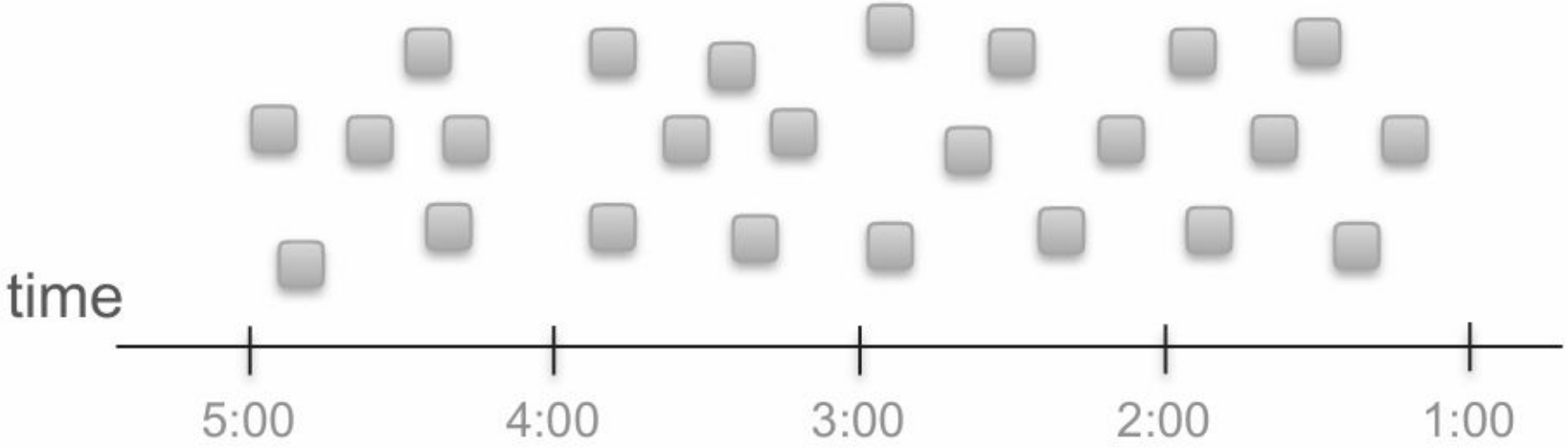
# Unified Batch & Streaming

**Unifying multiple code paths**

Google Dataflow

beam

**Data viewed as events**

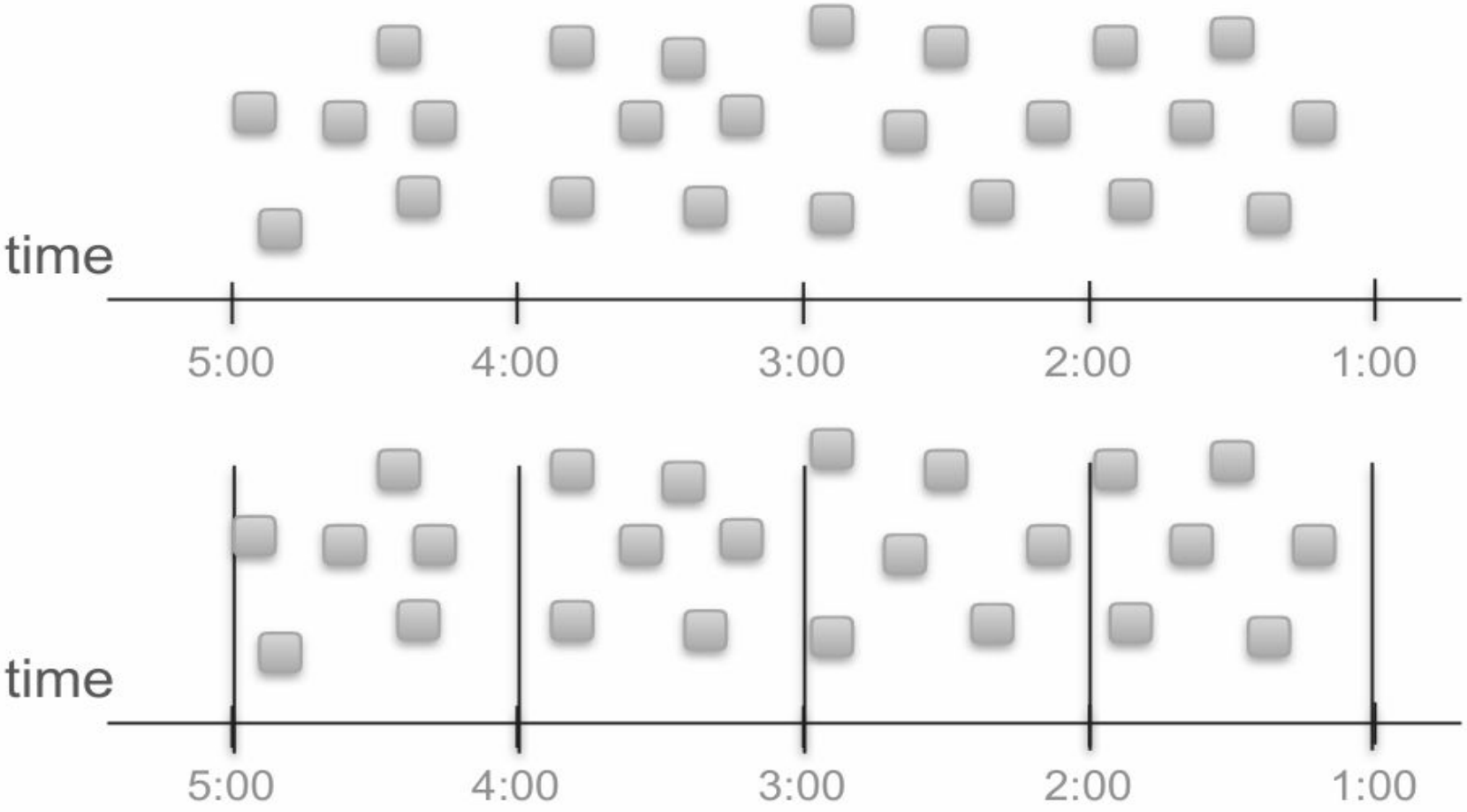


# Unified Batch & Streaming

Unbounded data:
real-time event streams

**Same code to process both types of data**

Bounded data:
data in batches



# Unified Batch & Streaming


RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

# Architecting for Compliance

# Architecting for Compliance

**General Data Protection Regulation (GDPR)**

*Enacted in European Union in 2018*

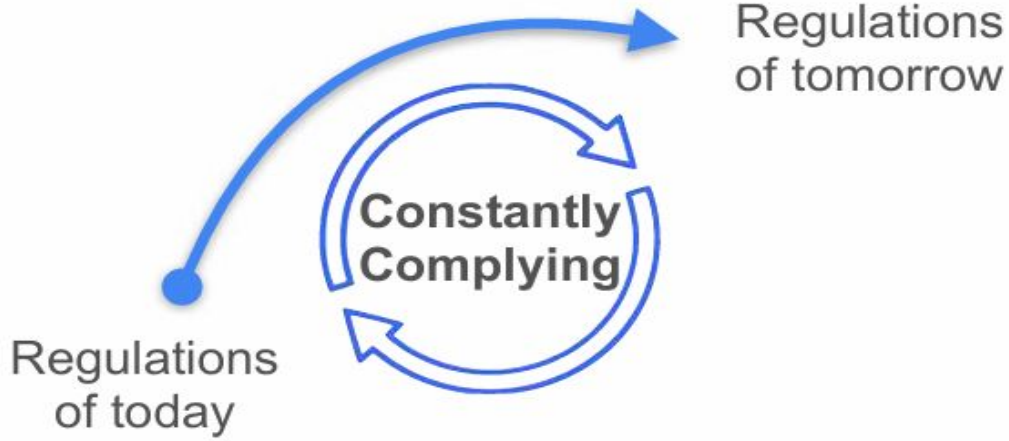
**Personal Data**

- Personal Identifiable Information (PII)
- Other information collectively used to identify an individual

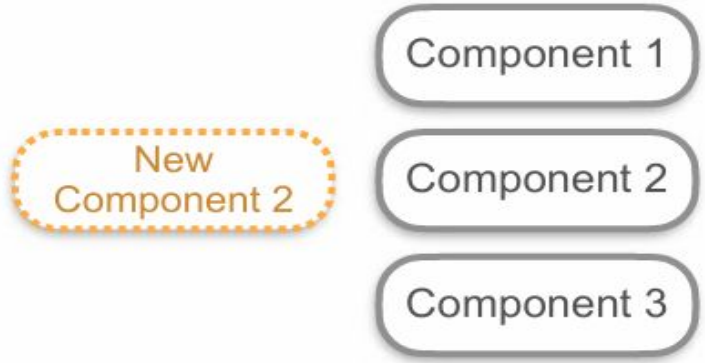
**Right to have your data deleted**


# Architecting for Compliance

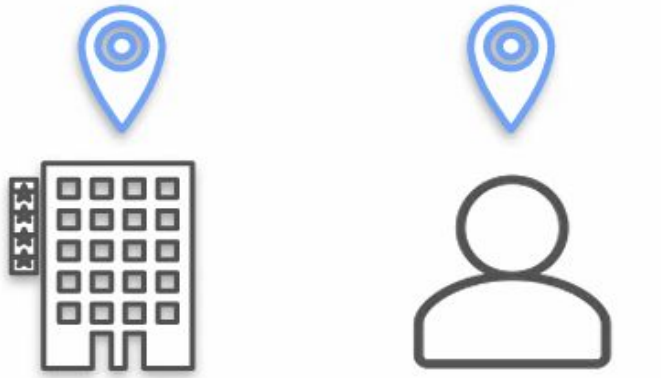
**GDPR and similar regulations have been enacted globally**




**Loosely Coupled Components**



# Architecting for Compliance




Industry

Health Insurance Portability and Accountability Act (HIPAA)

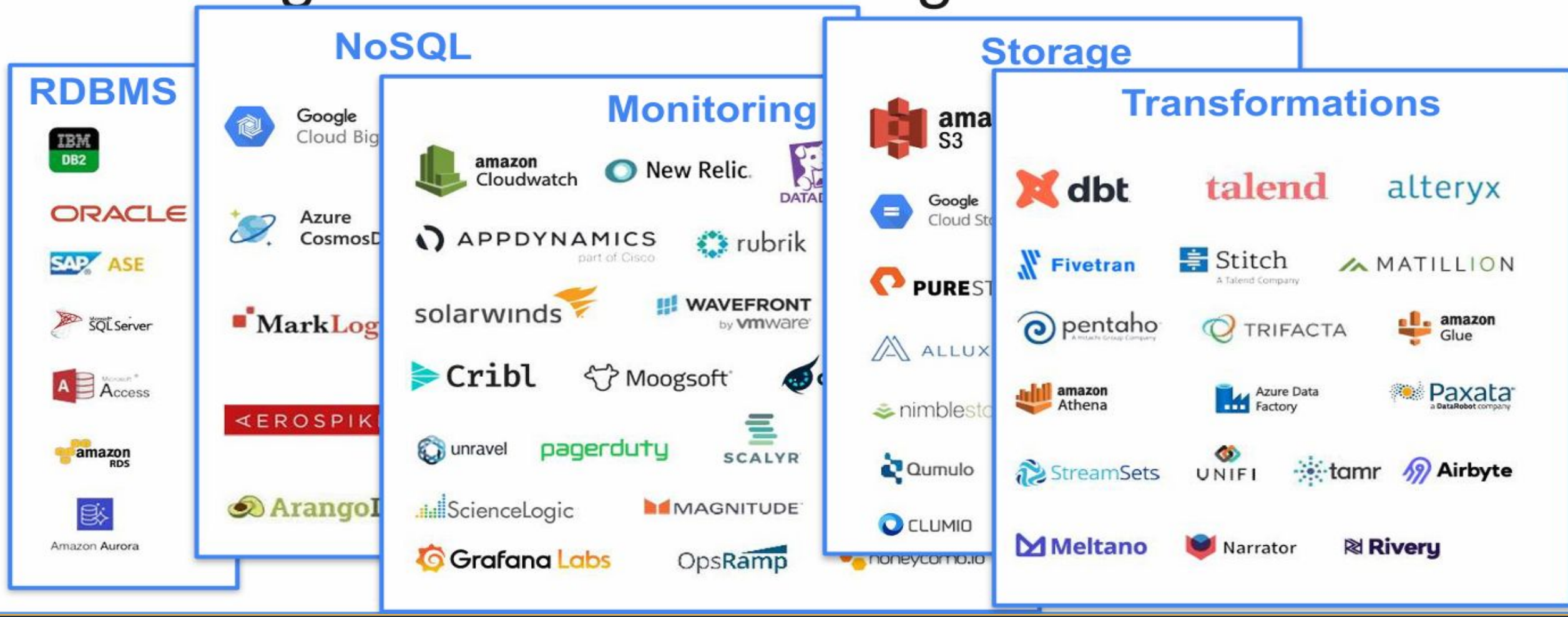
Financial Data

Sarbanes Oxley Act in the U.S

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

# Choosing tools and technologies

# Choosing Tools and Technologies



# Choosing Tools and Technologies

## Options of software solutions


**Open source software**

**Managed open source software**

**Proprietary software**

# Choosing Tools and Technologies

**Keep in mind the end goal!**

**Deliver high-quality data products**


Data Architecture

What Why

When

Tools

How

**Reality!**

**On-premises:** More control, but higher maintenance and upfront costs.
**Cloud:** Flexible and scalable, but recurring costs and vendor lock-in risks.
**Hybrid:** Balance between the two, but adds complexity.

# Choosing Tools and Technologies - Considerations

## Location

On-Premises

Cloud

Hybrid

## Other Considerations

Cost Optimization

Team's size & capabilities

Build

vs

Buy

# Location - On-Premises

On-Premises

Company **owns and maintains** the hardware and software for their data stack.

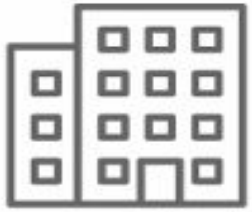
- Provisioning
- Maintaining
- Updating
- Scaling

# Location - Cloud


Cloud

- **Cloud provider** is responsible for **building and maintaining** the hardware in data centers
- You **rent** the compute and storage resources
- You don't need to maintain or provision any hardware

# Location



On-Premises

Migrating

Cloud

Regulations, security and client privacy concerns

Hybrid

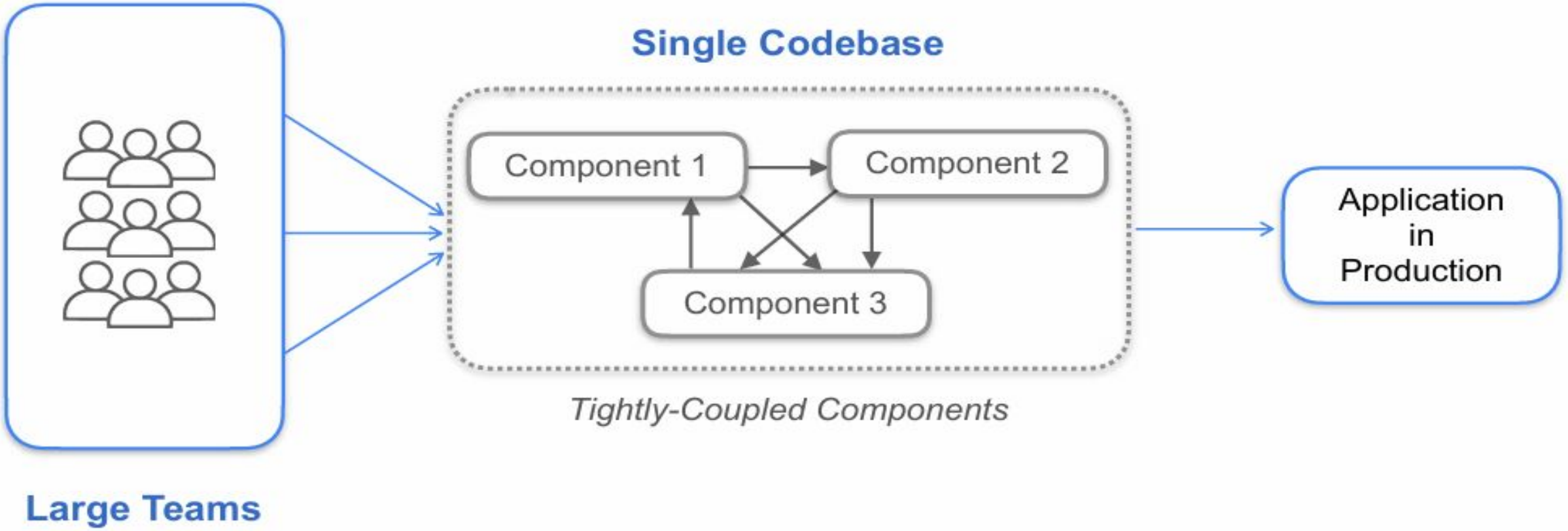
Flexibility and scalability

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

# Monolith versus modular systems

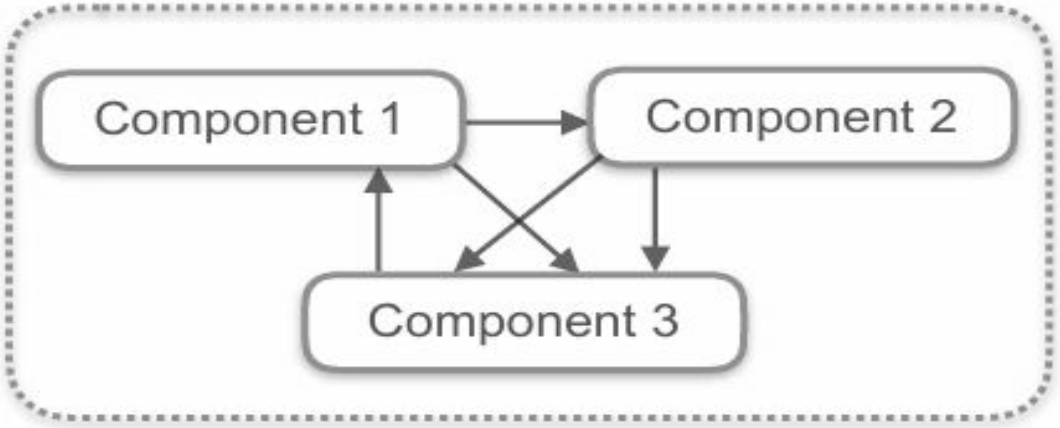
All components (UI, business logic, database access) are tightly integrated and deployed as a single unit. One failure can crash the entire application.

# Monolithic Systems



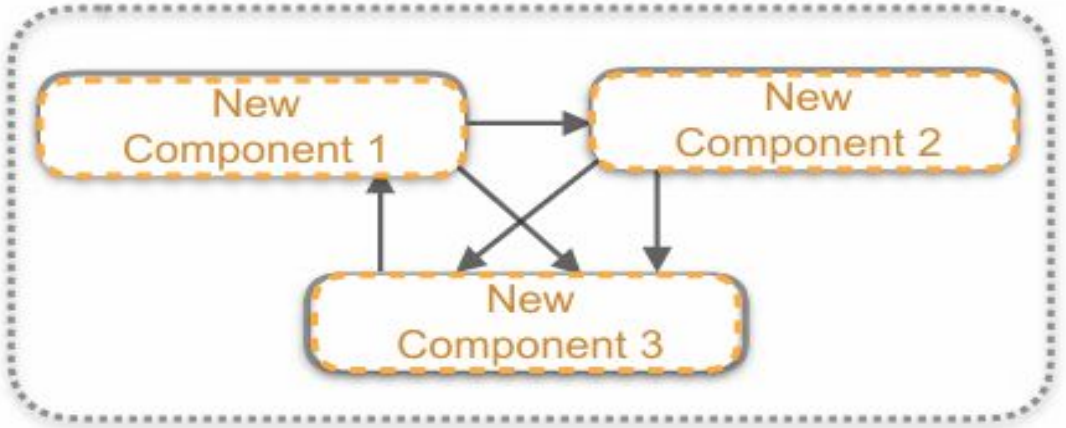
# Monolithic Systems

**Single Codebase**




*Tightly-Coupled Components*

**Hard to maintain**



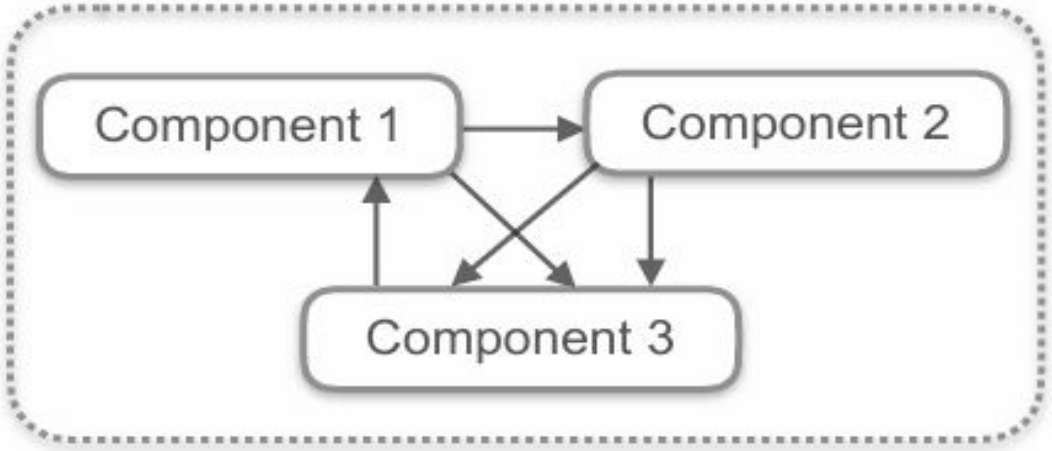

*Entire application has to be re-written*

- Easy to reason about and to understand
- Deal with one technology

Application divided into small, independent services. Failure isolated to a single service; others remain operational.

# Monolithic VS Modular Systems

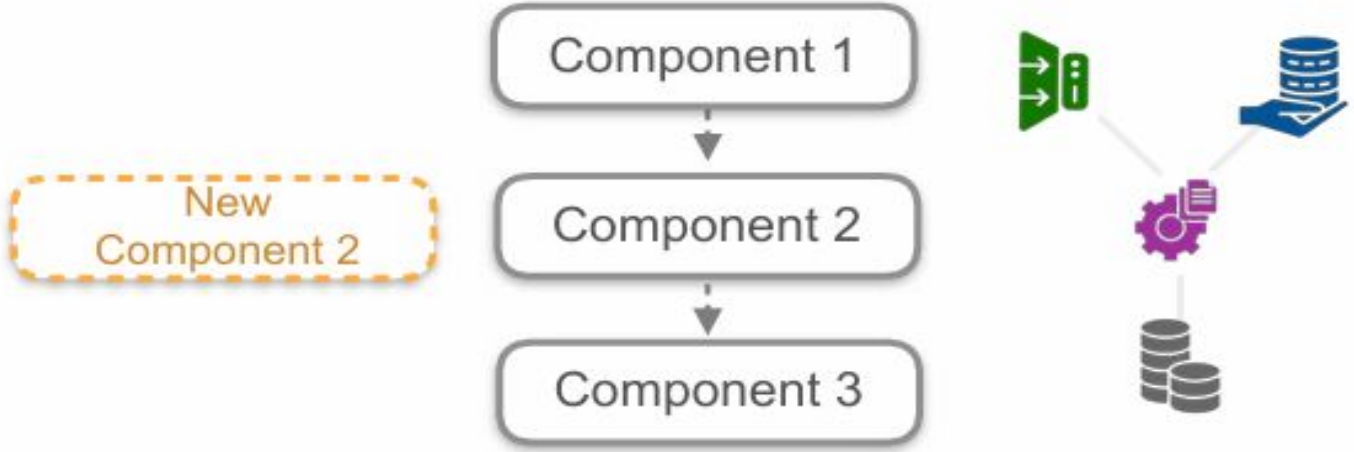
## Monolithic




*Tightly-Coupled Components*

- Easy to reason about and to understand
- Deal with one technology

## Modular




*Loosely-Coupled Components*

- Interoperability
- Flexible & reversible decisions
- Continuous improvement

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

# Cost Optimization and Business Value

# Cost Optimization

- Total Cost of Ownership (TCO)
- Total Opportunity Cost of Ownership (TOCO)
- FinOps

# Cost Optimization

**Total Cost of Ownership (TCO)**


The total estimated cost of a solution, project or initiative over its entire lifecycle.

## Direct Costs

Easy to identify costs, directly attributed to the development of a data product.

- Salaries
- Cloud bills
- Software subscriptions

## Indirect Costs (Overhead)

Expenses that are not directly attributed to the development of a data product.

- Network downtime
- IT support
- Loss of productivity

# Cost Optimization

**Total Cost of Ownership (TCO)**

The total estimated cost of a solution, project or initiative over its entire lifecycle.

Hardware & Software

## Capital Expenses (CapEx)

The payment made to purchase long-term fixed assets

# Cost Optimization

**Total Cost of Ownership (TCO)**

The total estimated cost of a solution, project or initiative over its entire lifecycle.

Hardware & Software

## Capital Expenses (CapEx)

The payment made to purchase long-term fixed assets


## Operational Expenses (OpEx)

Expense associated with running the day-to-day operations.

# Cost Optimization

**Total Cost of Ownership (TCO)**

The total estimated cost of a solution, project or initiative over its entire lifecycle.

## On-premises

### Capital Expenses (CapEx)

The payment made to purchase long-term fixed assets


## Cloud

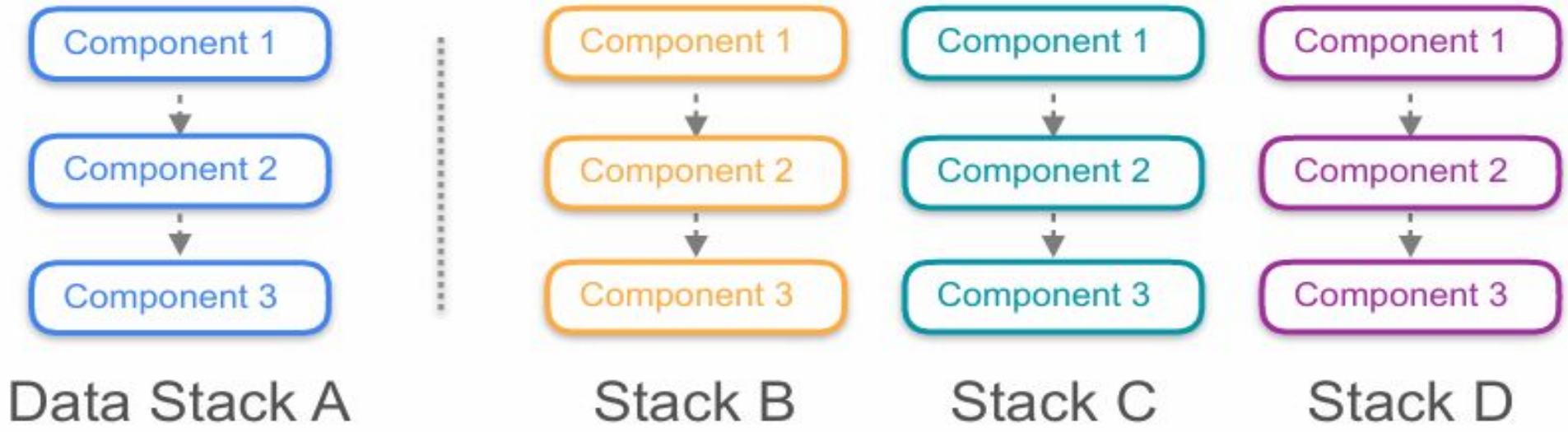
### Operational Expenses (OpEx)

Expense associated with running the day-to-day operations.

# Cost Optimization

**Total Opportunity Cost of Ownership (TOCO)**

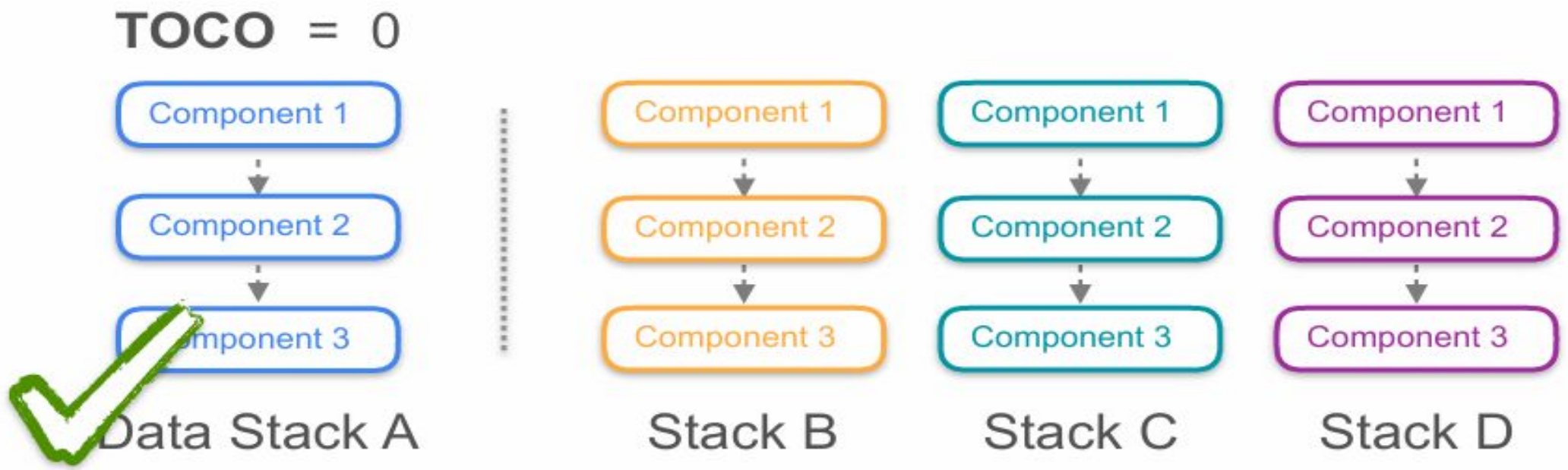
The cost of lost opportunities that you incur in choosing a particular tool or technology.



# Cost Optimization

**Total Opportunity Cost of Ownership (TOCO)**

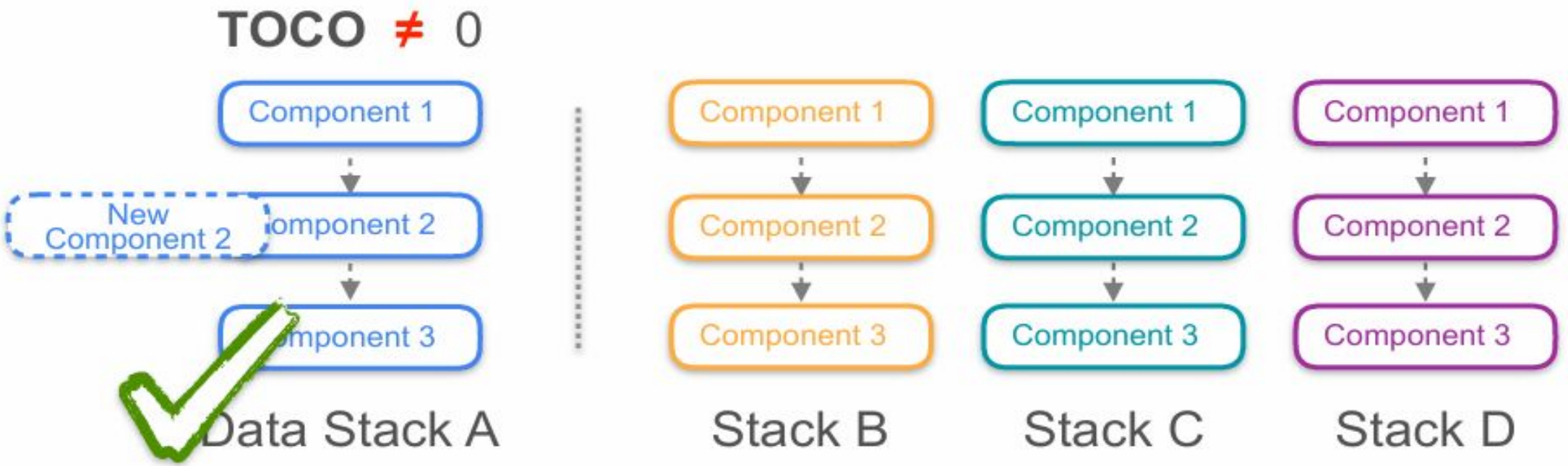
The cost of lost opportunities that you incur in choosing a particular tool or technology.



# Cost Optimization

**Total Opportunity Cost of Ownership (TOCO)**

The cost of lost opportunities that you incur in choosing a particular tool or technology.



# Cost Optimization

**Total Opportunity Cost of Ownership (TOCO)**

The cost of lost opportunities that you incur in choosing a particular tool or technology.

**Minimize TOCO**

**Build flexible systems**


*Loosely-Coupled Components*

**Recognize components that are likely to change**

- Immutable technologies

  Object storage, Networking, SQL

- Transitory technologies

  Stream processing, Orchestration, AI

# Cost Optimization

**FinOps**

Minimize TCO and TOCO

Maximize revenue generation opportunities

**Cloud**

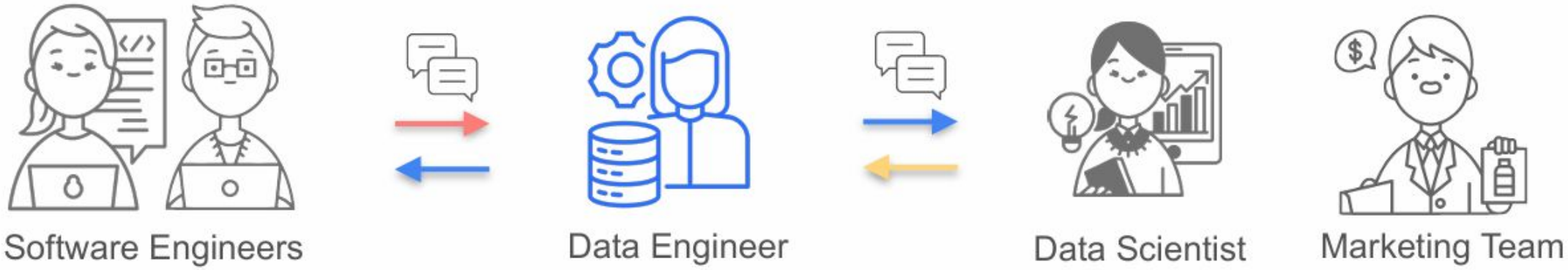
OpEx-first

- Flexible, pay-as-you-go technologies
- Modular options

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

# Stakeholder Management and Gathering Requirements

# Requirements Gathering



# Hierarchy of Needs


**What does success look like?**

- Growing revenue
- Growing market share
- Growing user base

**Individual employees**

- Proper resources, tools, & management
- Need robust data systems

**Requirements for the data system you will build & maintain**

# Hierarchy of Needs


**What does success look like?**

- Growing revenue
- Growing market share
- Growing user base

**Individual employees**

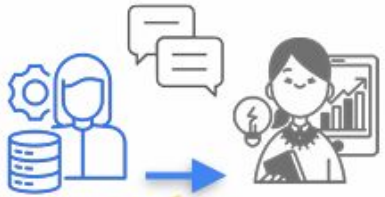
- Proper resources, tools, & management
- Need robust data systems

**Requirements for the data system you will build & maintain**

data engineering work must connect to stakeholder needs and overall business objectives, making requirements gathering essential. Ideally, this starts with leadership discussions (e.g., CEO, CTO, or CDO) to align with company goals.

# Requirements Gath...ring

CEO



Data Engineer

Data Scientist

**Business Goals**

**Stakeholder Needs**

**System Requirements**

*Functional*

Specific functionality:

What the system will be able to do

*Nonfunctional*

Characteristics or attributes:

Latency, scalability, reliability, cost, security

**What does success look like?**

- Growing revenue
- Growing market share
- Growing user base

**Individual employees**

- Proper resources, tools, & management
- Need robust data systems

**Requirements for the data system you will build & maintain**

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

# Breaking Down the Conversation with Marketing

# Documenting Requirements

# Documenting Requirements

**Business Goals**

Continue on the growth trajectory:

Focus on customer retention and loyalty, expand to new markets and new product offerings

**Stakeholder Needs**

**System Requirements**

*Functional*

*Nonfunctional*

# Documenting Requirements

## Business Goals

Continue on the growth trajectory:
Focus on customer retention and loyalty, expand to new markets and new product offerings

## Stakeholder Needs

## System Requirements

*Functional*

*Nonfunctional*

# Documenting Requirements

**Business Goals**

Continue on the growth trajectory:
Focus on customer retention and loyalty, expand to new markets and new product offerings

**Stakeholder Needs**

| | |
|---|---|
| *Analytics dashboard* | *Recommender System* |

**System Requirements**

| | |
|---|---|
| *Functional* | *Functional* |
| *Nonfunctional* | *Nonfunctional* |

# Documenting Requirements

## Business Goals

Continue on the growth trajectory:
Focus on customer retention and loyalty, expand to new markets and new product offerings

## Stakeholder Needs

*Analytics dashboard*

Marketing needs to know about demand spikes, with hourly dashboard updates

*Recommender System*

## System Requirements

*Functional*

The data system needs to serve transformed data that is no more than one hour old

*Nonfunctional*

*Functional*

*Nonfunctional*

# Documenting Requirements

## Business Goals

Continue on the growth trajectory:
Focus on customer retention and loyalty, expand to new markets and new product offerings

## Stakeholder Needs

### *Analytics dashboard*

Marketing needs to know about demand spikes, with hourly dashboard updates

### *Recommender System*

Marketing needs a system that recommends products based on browsing or purchase history and current cart contents

## System Requirements

### *Functional*

The data system needs to serve transformed data that is no more than one hour old

### *Functional*

The system needs to:

- Provide training data for recommender
- Ingest, transform, & serve user data to recommender
- Return model outputs back to sales platform

### *Nonfunctional*

### *Nonfunctional*

## Stakeholder Needs

- (from the data scientist + product marketing manager):

## Analytics Dashboards

- Need near real-time data (not 2-day-old data).
- Must detect **demand spikes** lasting a few hours to 1–2 days.
- **Hourly data updates** are sufficient to take timely action.

## Recommender System

- Move beyond "popular products of the week."
- Provide **personalized recommendations** using:
  - Purchase history
  - Browsing data
  - Items currently in cart

# Functional Requirements - Analytics Dashboard

**Functional Requirements:** System serves needed data in a timely manner

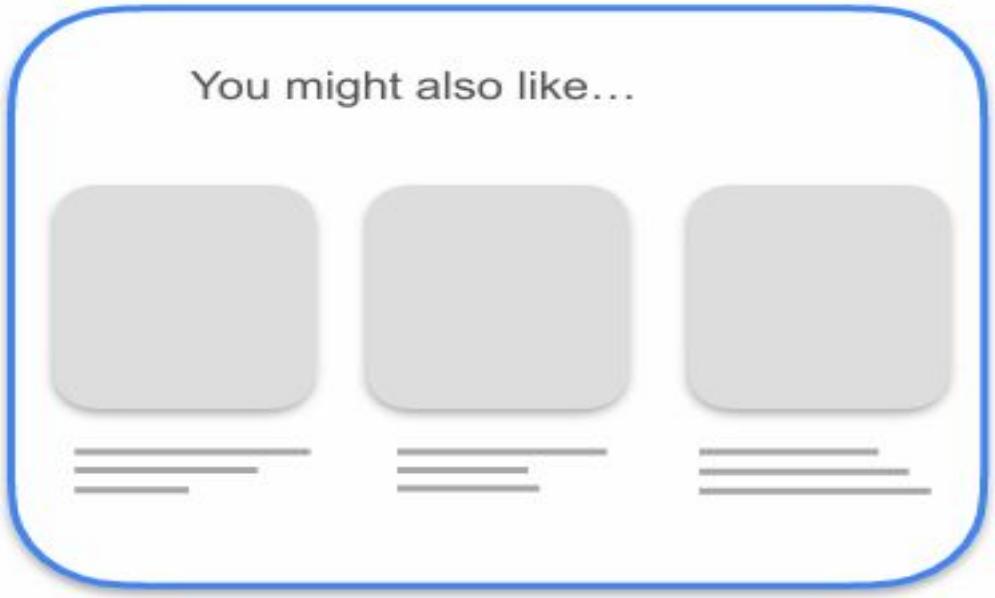
An analytics dashboard

**Dashboard features or metrics to display:**

- Responsibility of the data scientist
- Not functional requirements of your data system

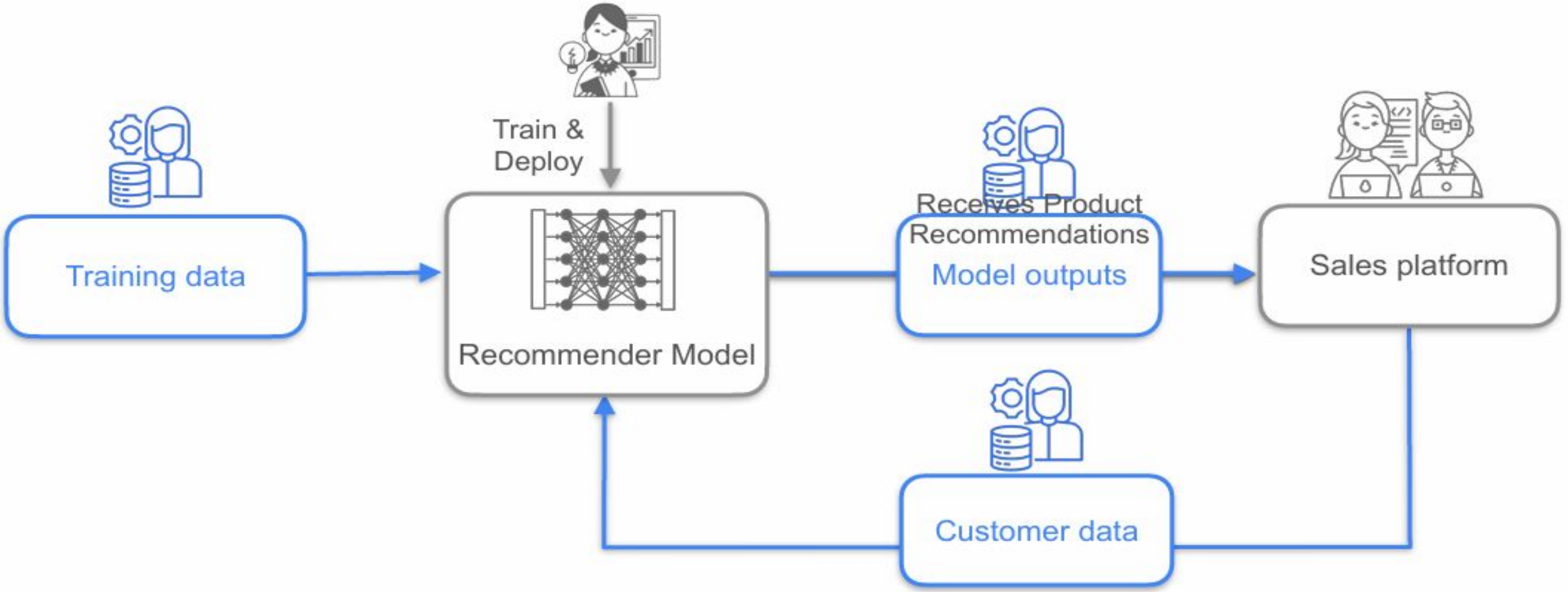
# Functional Requirements - Recommender System

## Functional Requirements




A recommender system

# Functional Requirements - Recommender System



## Functional Requirements (Examples)

### Dashboards

- *Requirement*: The data system must serve transformed data that is **no more than one hour old**.
- Note: Building the dashboards themselves is the responsibility of data scientists; the data engineer's role is ensuring timely, usable data delivery.

### Recommender System

- *Requirement*:
  - Provide training data for recommender model development.
  - Ingest, transform, and serve **user data** to the trained model.
  - Return model outputs (e.g., product IDs) back to the sales platform.

# Documenting Requirements

## Business Goals

Continue on the growth trajectory:

Focus on customer retention and loyalty, expand to new markets and new product offerings

## Stakeholder Needs

*Analytics dashboard*

Marketing needs to know about demand spikes, with hourly dashboard updates

*Recommender System*

Marketing needs a system that recommends products based on browsing or purchase history and current cart contents

## System Requirements

*Functional*

The data system needs to serve transformed data that is no more than one hour old

*Nonfunctional*

*Functional*

The system needs to:

- Provide training data for recommender
- Ingest, transform, & serve user data to recommender
- Return model outputs back to sales platform

*Nonfunctional*

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

# Documenting Nonfunctional Requirements

# Conversation Takeaways


- Ensure a degree of stability for read replica
- Provide notifications for system outages or changes to the database schema

# Documenting Requirements

## Business Goals

Continue on the growth trajectory:
Focus on customer retention and loyalty, expand to new markets and new product offerings

## Stakeholder Needs

### *Analytics dashboard*

Marketing needs to know about demand spikes, with hourly dashboard updates

### *Recommender System*

Marketing needs a system that recommends products based on browsing or purchase history and current cart contents

## System Requirements

### *Functional*

The data system needs to serve transformed data that is no more than one hour old

### *Functional*

The system needs to:

- Provide training data for recommender
- Ingest, transform, & serve user data to recommender
- Return model outputs back to sales platform

### *Nonfunctional*

### *Nonfunctional*

An analytics dashboard

## System Requirements

*Functional*

The data system needs to serve transformed data that is no more than one hour old

*Nonfunctional*

**Scalability and Latency**

System will be able to scale up to ingest, transform and serve the data volume expected with the maximum level of user activity, while staying within the latency requirements

**Reliability**

System will perform data quality checks to ensure data is conformant

**Maintainability**

The ingestion and transformation stages must be easily adaptable to accommodate any changes in the data schema

## A recommender system


"Real-time"?

## System Requirements

*Functional*

The system needs to:

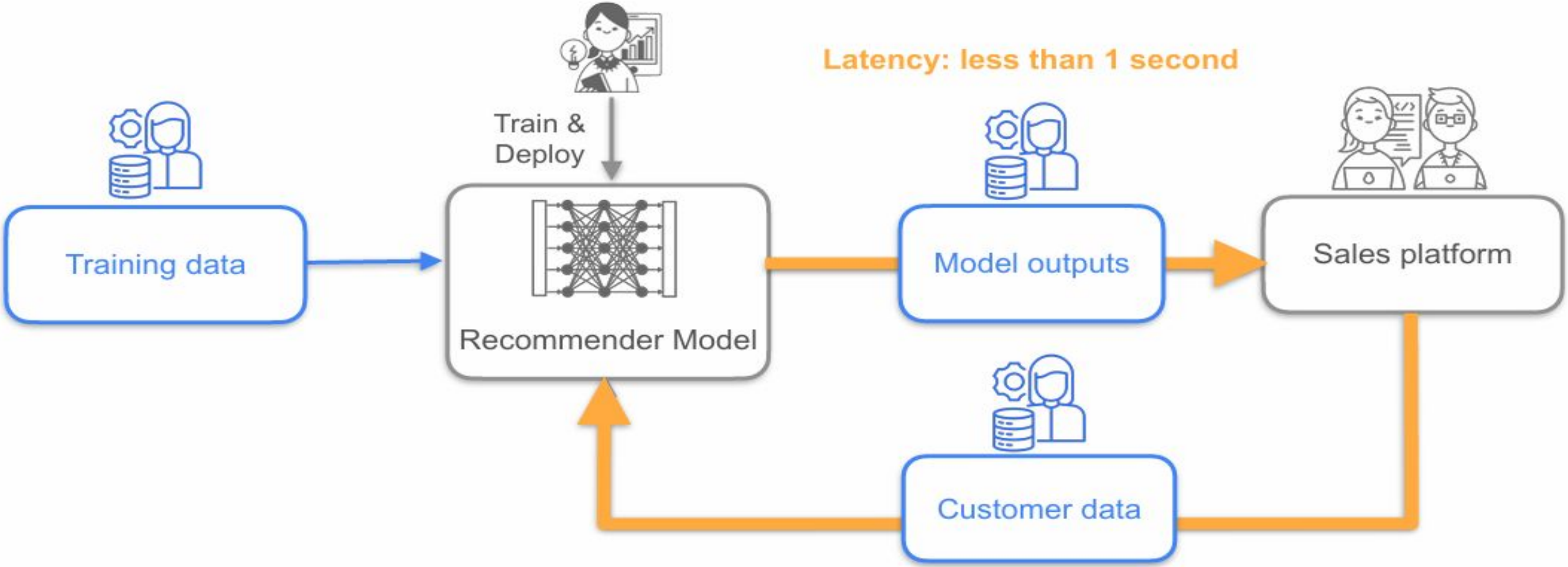
- Provide training data for recommender
- Ingest, transform, & serve user data to recommender
- Return model outputs back to sales platform

*Nonfunctional*

**Latency**

System must have a latency of less than 1 second from ingestion of user data to serving of recommendation data

# Functional Requirements - Recommender System



## A recommender system


## System Requirements

### *Functional*

The system needs to:

- Provide training data for recommender
- Ingest, transform, & serve user data to recommender
- Return model outputs back to sales platform

### *Nonfunctional*

**Latency**

System must have a latency of less than 1 second from ingestion of user data to serving of recommendation data

**Scalability**

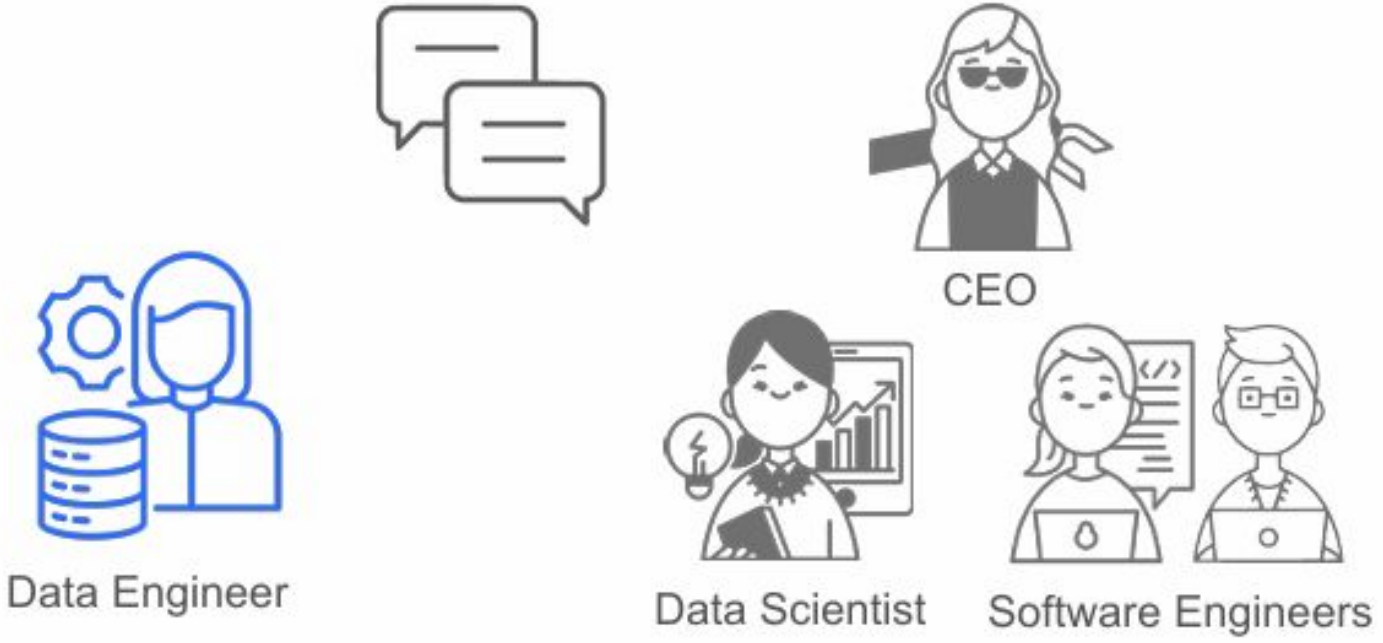
System must be able to scale up to the maximum number of concurrent users on the platform

**Reliability**

- System must return a set of recommendations within one second
- If the recommender pipeline fails it should default to serving a selection of the most popular products

# Main Takeaways

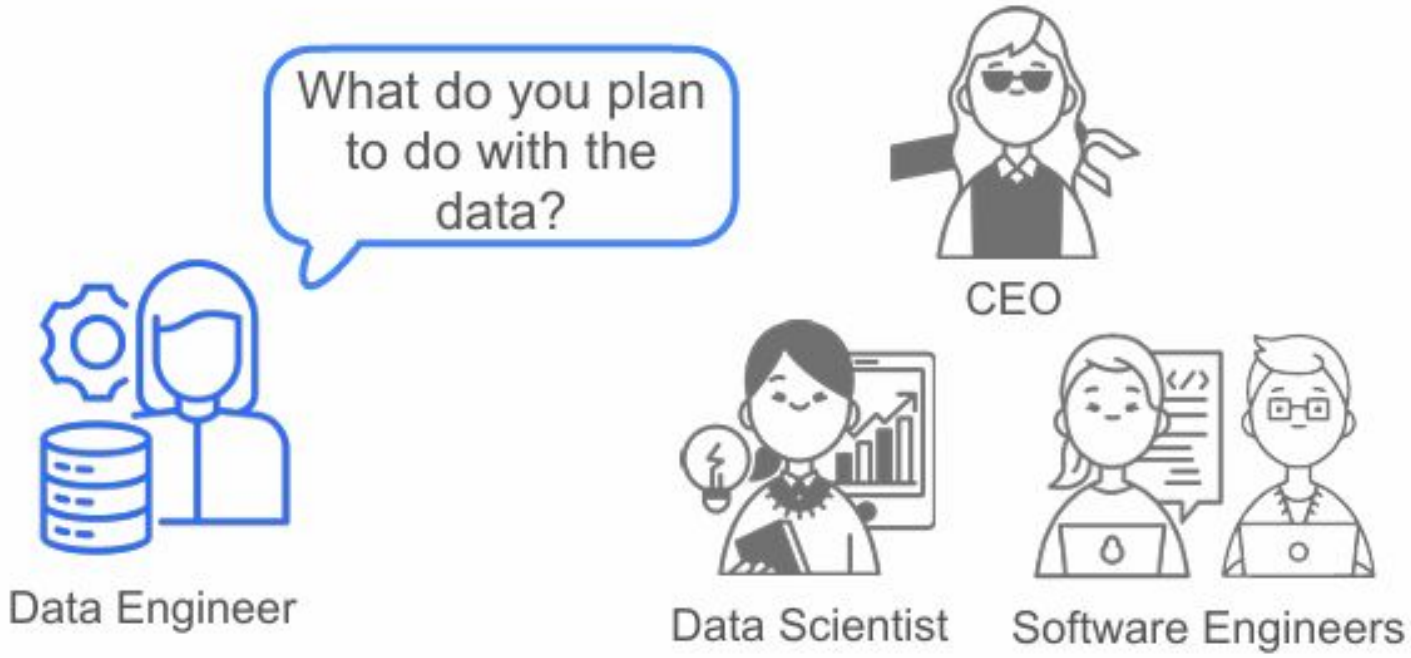
1. Identify the stakeholders, understand their needs and the broader goals of the business



# Main Takeaways

1. Identify the stakeholders, understand their needs and the broader goals of the business

2. Ask open-ended questions



# Main Takeaways

1. Identify the stakeholders, understand their needs and the broader goals of the business
2. Ask open-ended questions
3. Document all of your findings

# Evaluation of Trade Offs

**Timeline**

Stakeholders want you to build them a data system as quickly as possible

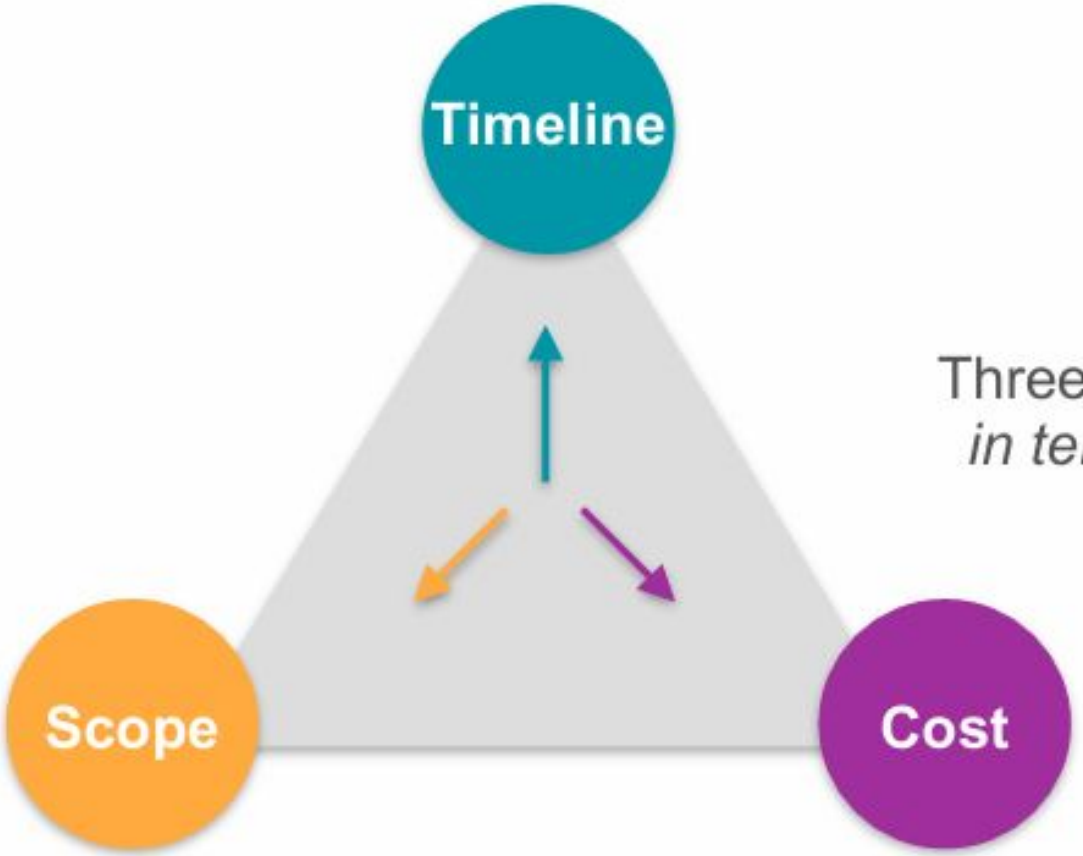
**Cost**

You might be working within a limited budget

**Scope**

Features of the system

# Iron Triangle

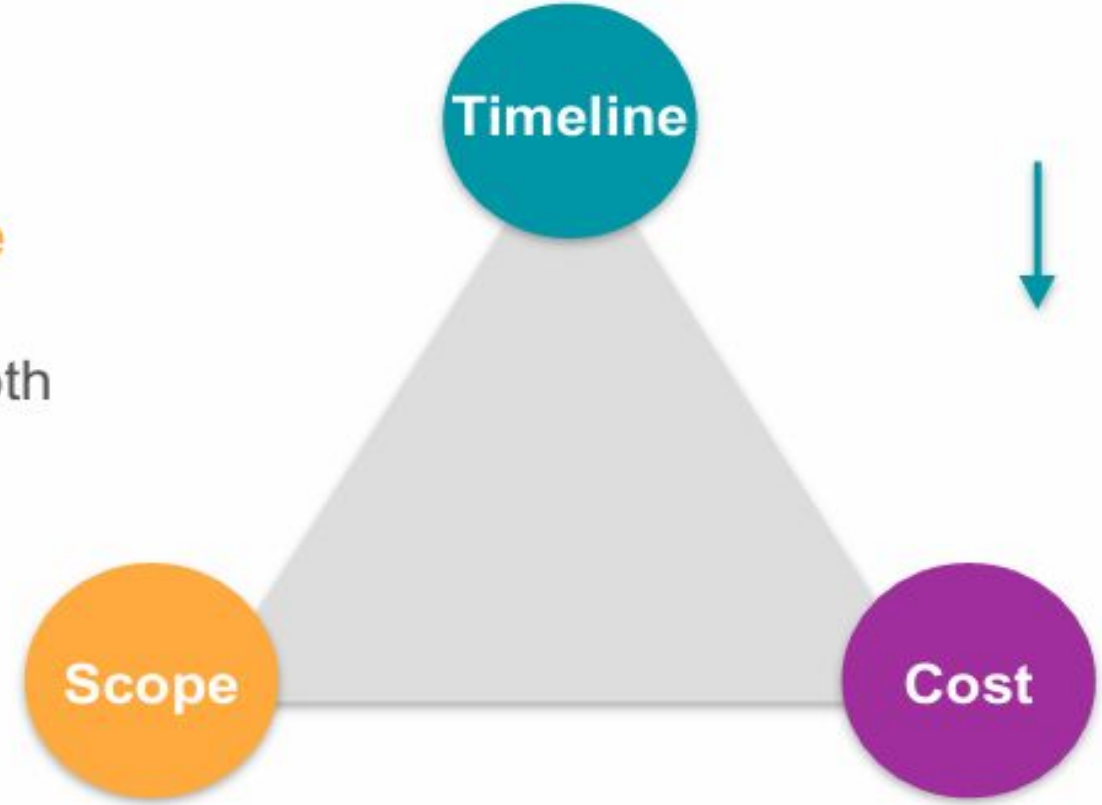



Three project aspects that are *in tension with one another*!

# Iron Triangle

**Increase in scope**

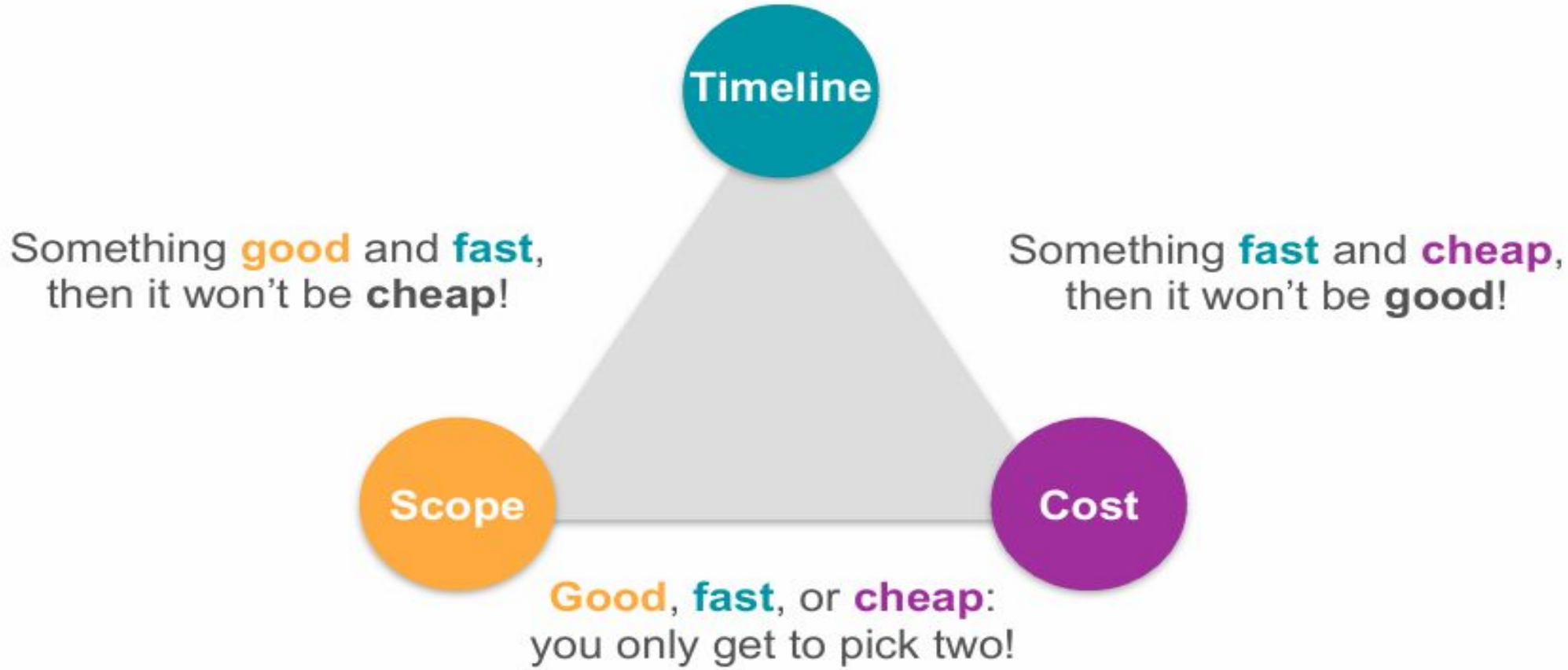
**Timeline** or **cost** or both must also increase




**Shorten the timeline**

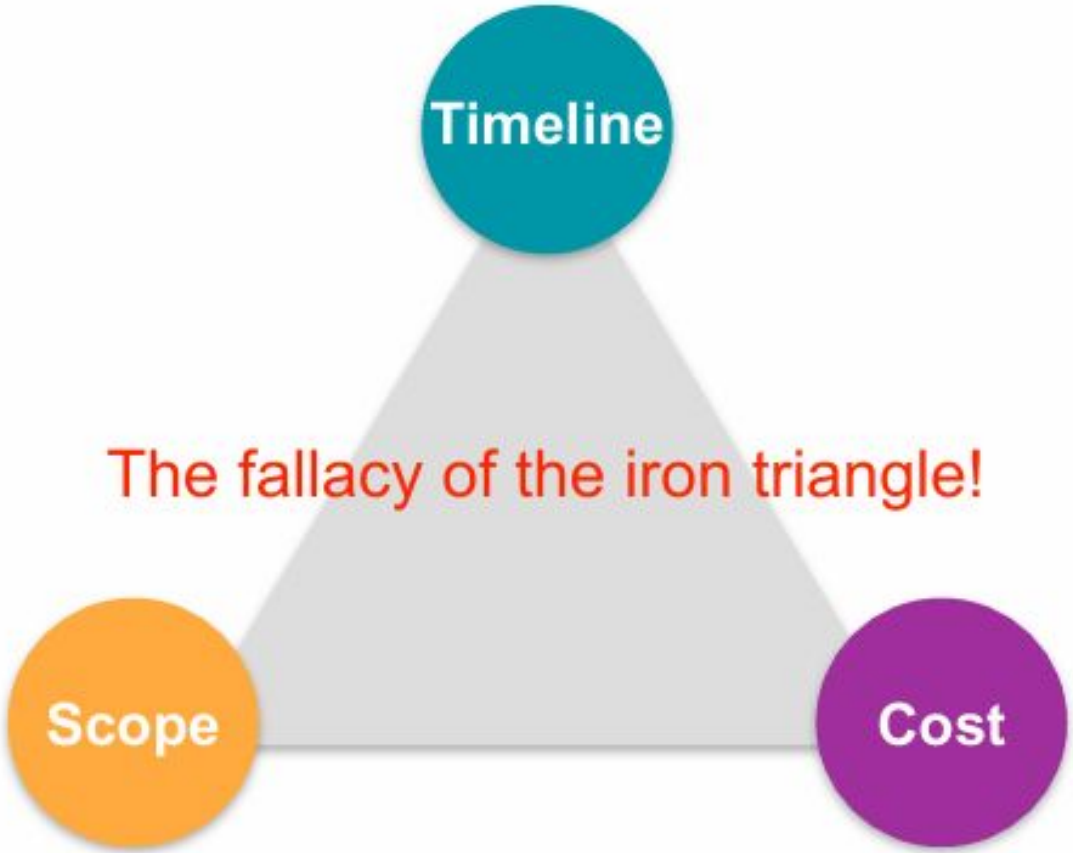
Increase the **cost** or reduce **scope**

# Iron Triangle



# Iron Triangle

Projects done as quickly as possible!



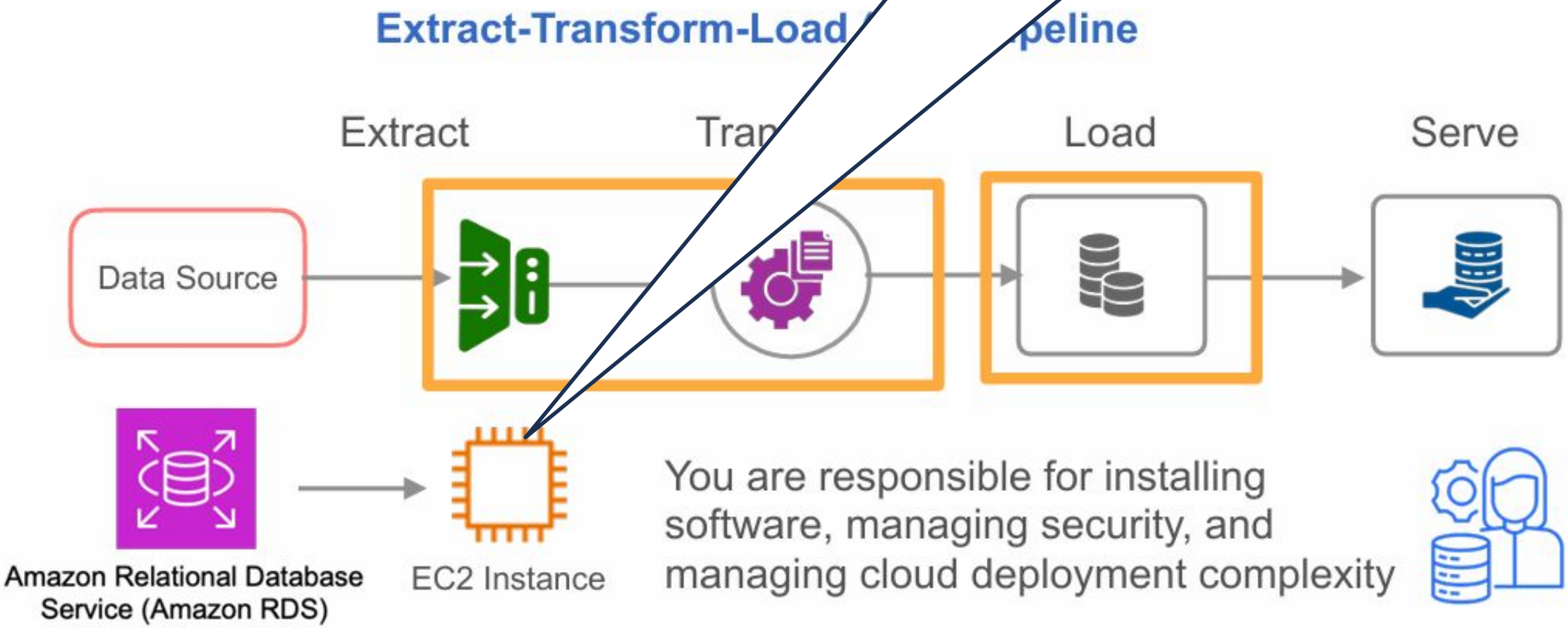

The fallacy of the iron triangle!

Projects done well!

Projects are within budget constraints!

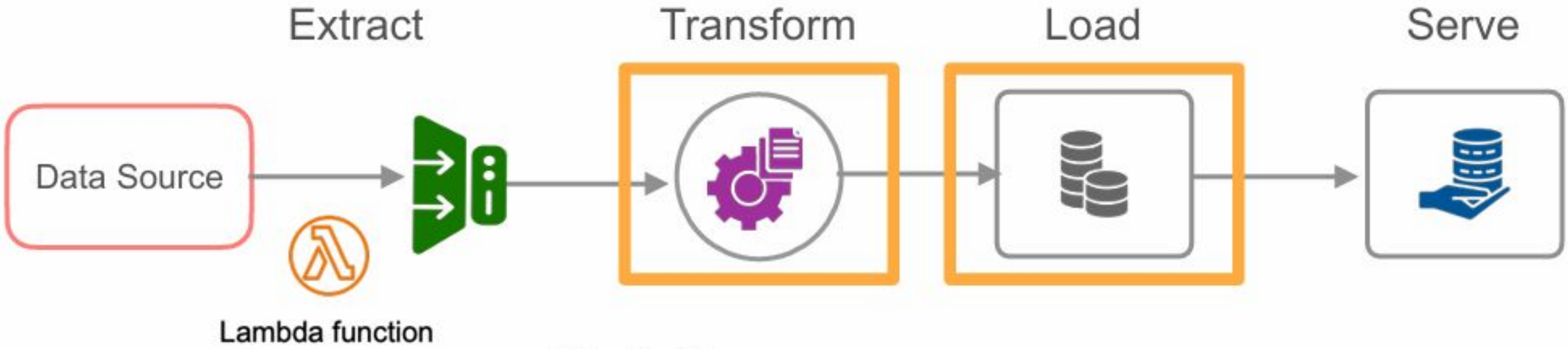
**Custom EC2 scripts: Possible, but involves heavy management.**

# AWS Services for Batch Pipelines



# AWS Services for Batch Pipelines

## Extract-Transform-Load (ETL) Pipeline




AWS Lambda

### Limitations

- 15-minute timeout for each function call
- Memory and CPU allocation for each function
- Requires you to write custom code for your use case

# Serverless Tools for Batch Processing

**Difference:** Tradeoffs between control vs convenience

Amazon EMR

AWS Glue ETL

## More control

- Designed as a big data tool




## More convenience

- Can handle big data with additional features

AWS Glue Crawler

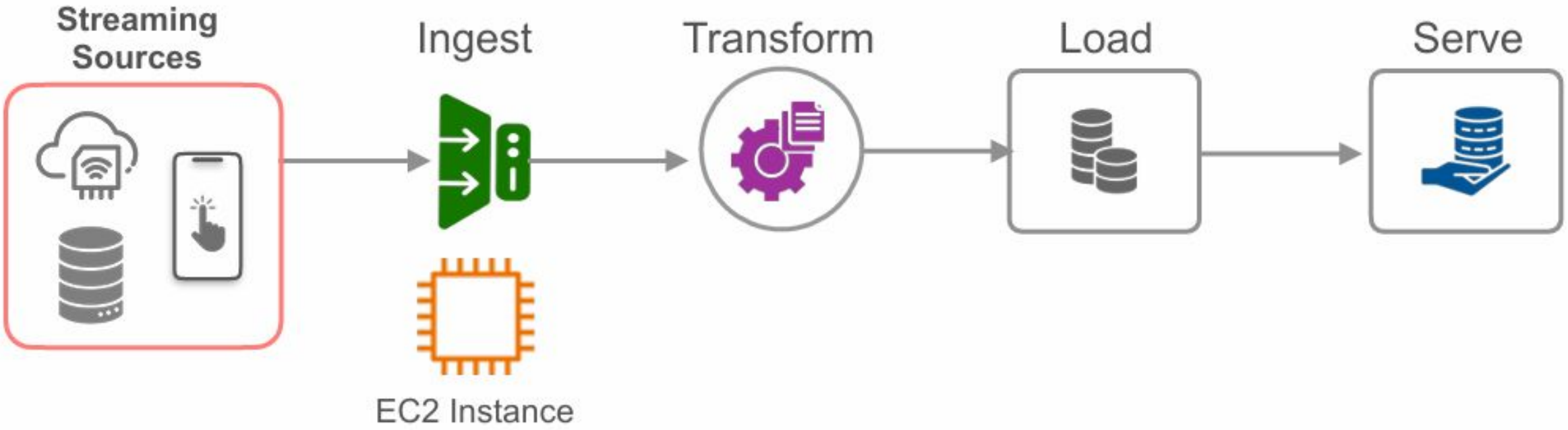
- Automatically discover & classify data
- Create metadata

AWS Glue Data Catalog

- A central repository with info about all your data assets

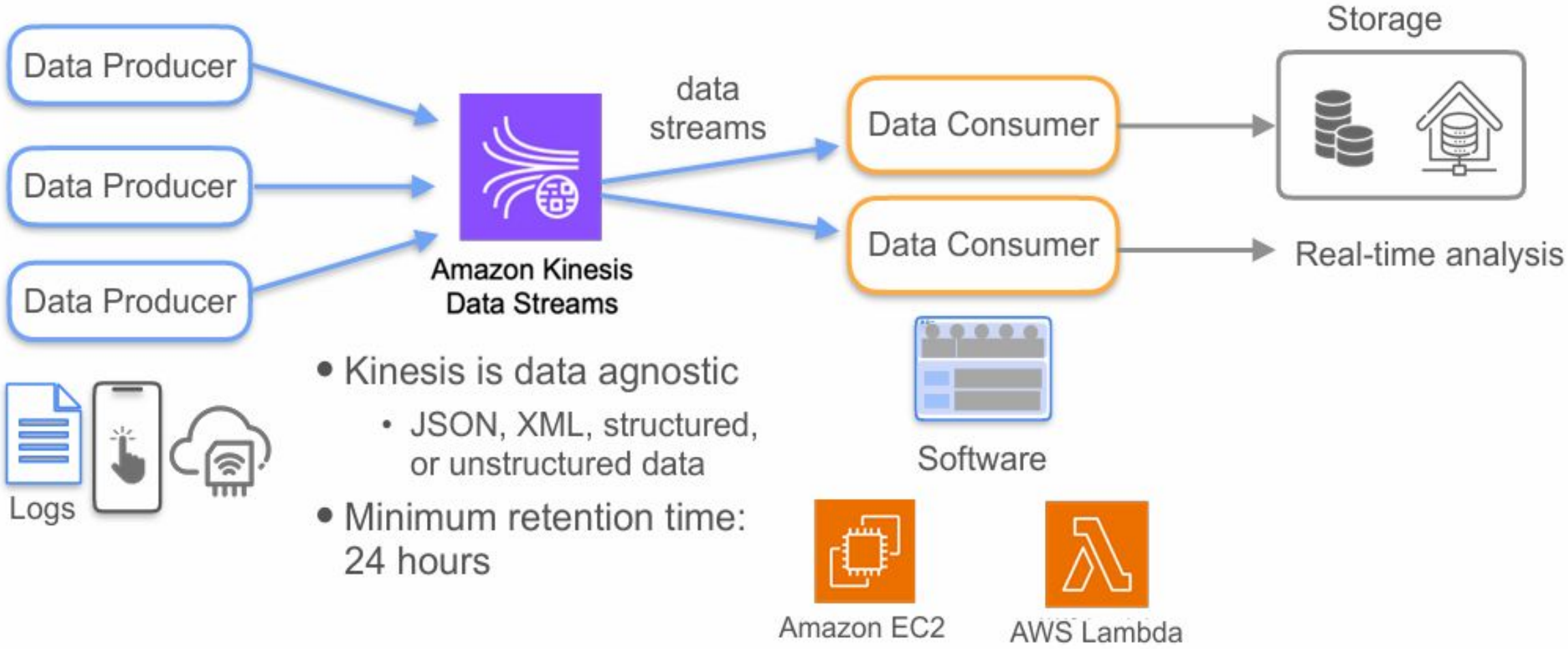
- Can use Glue visual ETL tool to design your pipeline

# AWS Services for Streaming Pipelines



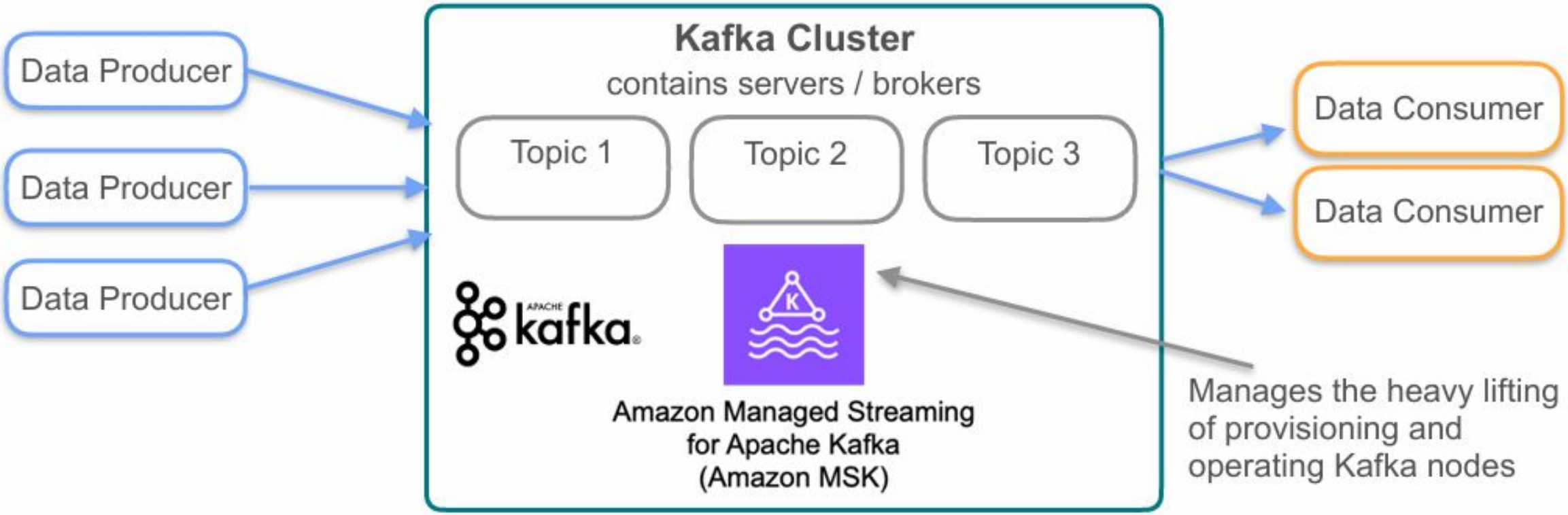

You are responsible for installing software, managing security, and managing cloud deployment complexity

# AWS Services for Streaming Pipelines




- Kinesis is data agnostic
  - JSON, XML, structured, or unstructured data
- Minimum retention time: 24 hours

# AWS Services for Streaming Pipelines




**Kafka Data Plane**

An interface you interact with that MSK manages for you to create topics, produce, and consume data

# AWS Services for Streaming Pipelines

Both can scale up to handle petabyte-level data volumes with millisecond latency

Amazon Managed Streaming for Apache Kafka (Amazon MSK)

Amazon Kinesis Data Streams

## More control

- Used for Kafka clusters
- High degree of flexibility and control

## More convenience

- User-friendly
- Reduced operational overhead

# Databricks

- The **main purpose of Databricks** is to provide a **unified, scalable, cloud-based platform** for:

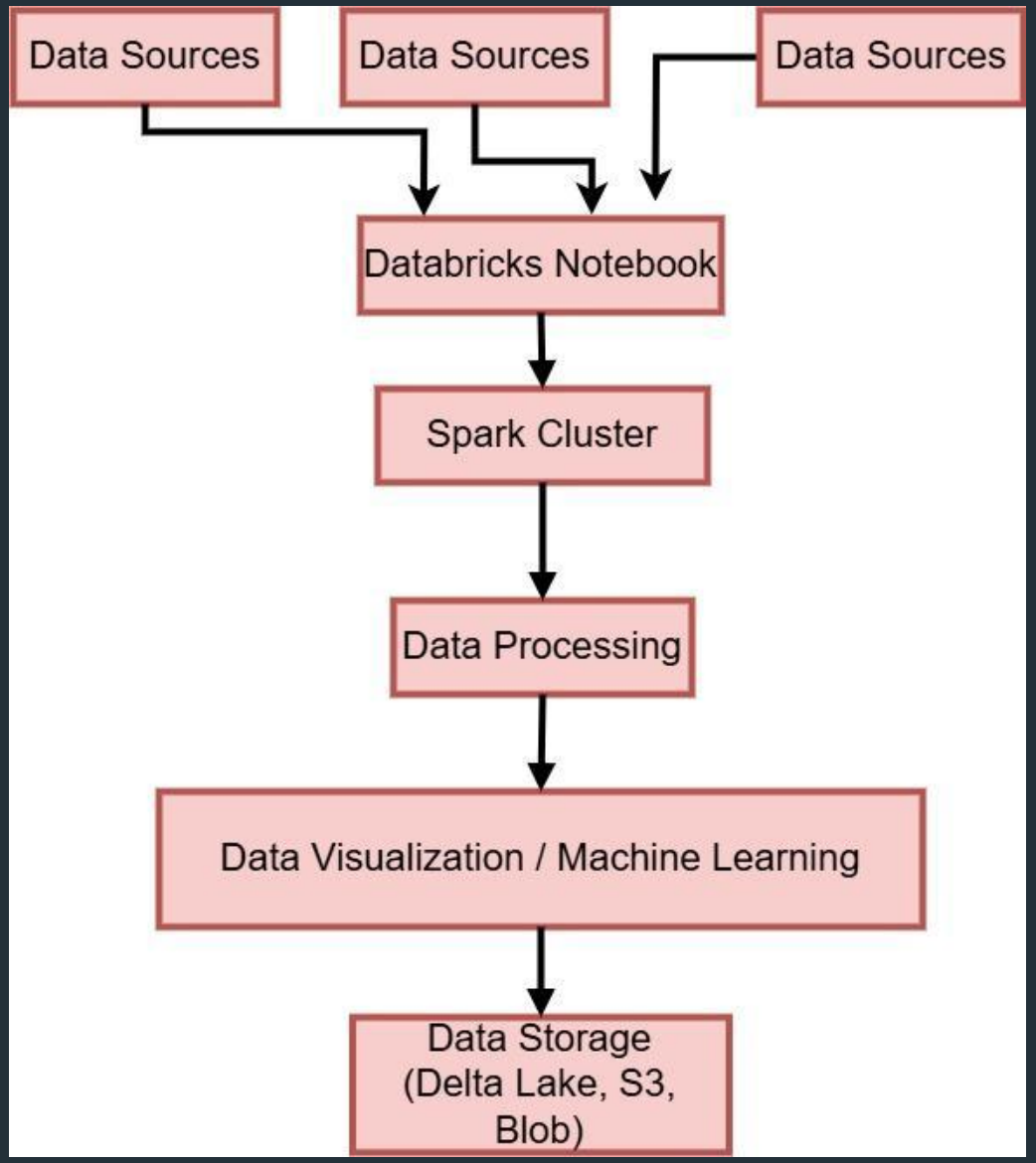
  **Data Engineering**

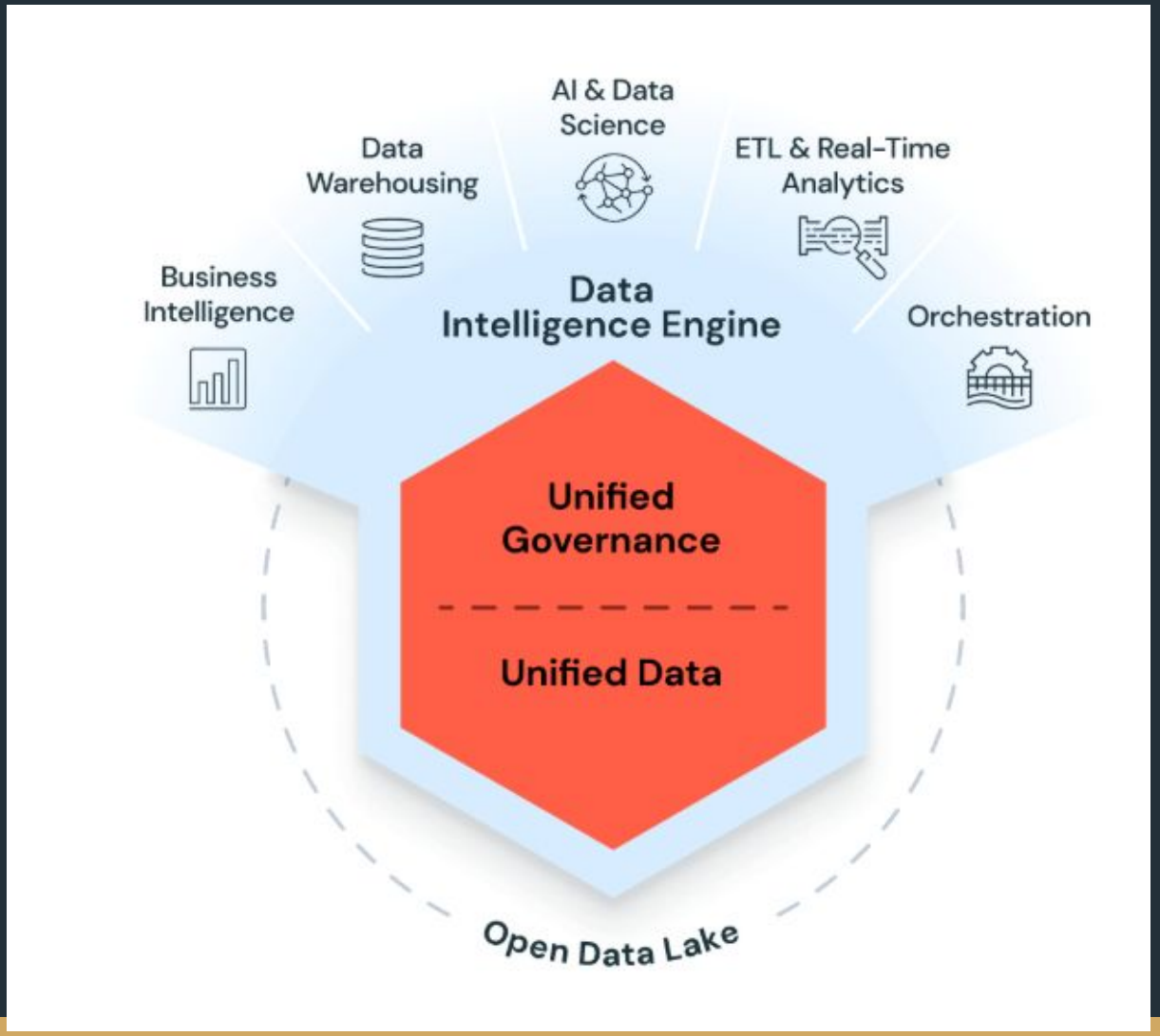
  **Data Science**

  **Machine Learning**

  **Big Data Analytics**

- It simplifies working with **large volumes of data** and makes it easier to build **end-to-end data pipelines** without worrying about complex infrastructure.


Data Sources
Data Sources
Data Sources
Databricks Notebook
Spark Cluster
Data Processing
Data Visualization / Machine Learning
Data Storage
(Delta Lake, S3,
Blob)


AI & Data
Science
Data
Warehousing
ETL & Real-Time
Analytics
Business
Intelligence
Data
Intelligence Engine
Orchestration
Unified
Governance
Unified Data
Open Data Lake

1. Data Sources → Load raw data into the system.
2. Databricks Notebook → Code written to process and analyze data.
3. SparkCluster → Parallel processing engine that executes the notebook commands.
4. Data Processing → Data cleaning, transformation, aggregation.
5. Visualization/ML → Analyze processed data, create charts, build machine learning models.
6. Data Storage → Save the final outputs for future access or reporting.

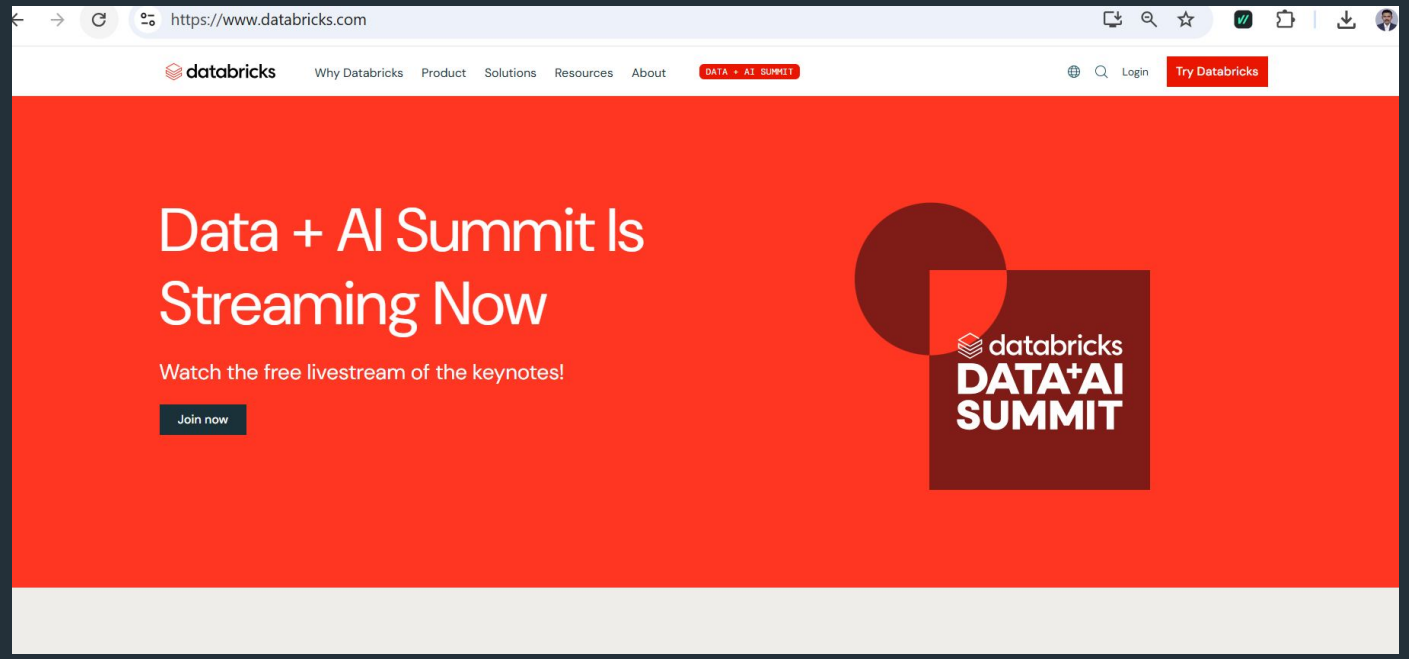
# Key Use Cases:

1. ETL/ELT pipelines
2. Real-time analytics
3. Machine learning (AutoML, model serving)
4. BI and SQL analytics
5. GenAI development (e.g., with vector databases, LLM integration)

# How to use

https://www.databricks.com/

Step1. Use this link and create account



Click this for free account

https://login.databricks.com/?dbx_source=www&itm_data=databricks-web-nav&intent=SIGN_UP&l=en-EN&tuuid=390703...

## Build AI with the Databricks Data Intelligence Platform

- Express setup gives you instant access and free serverless credits
- Create high-quality Generative AI applications
- Simplify and automate data ingestion and ETL
- Query your data with common language

TRUSTED BY

MICHELIN Square Walgreens Mercedes Benz

### Use express setup

Quickly get hands-on with Databricks. We'll manage your account and cloud infrastructure.

Continue with Express Setup

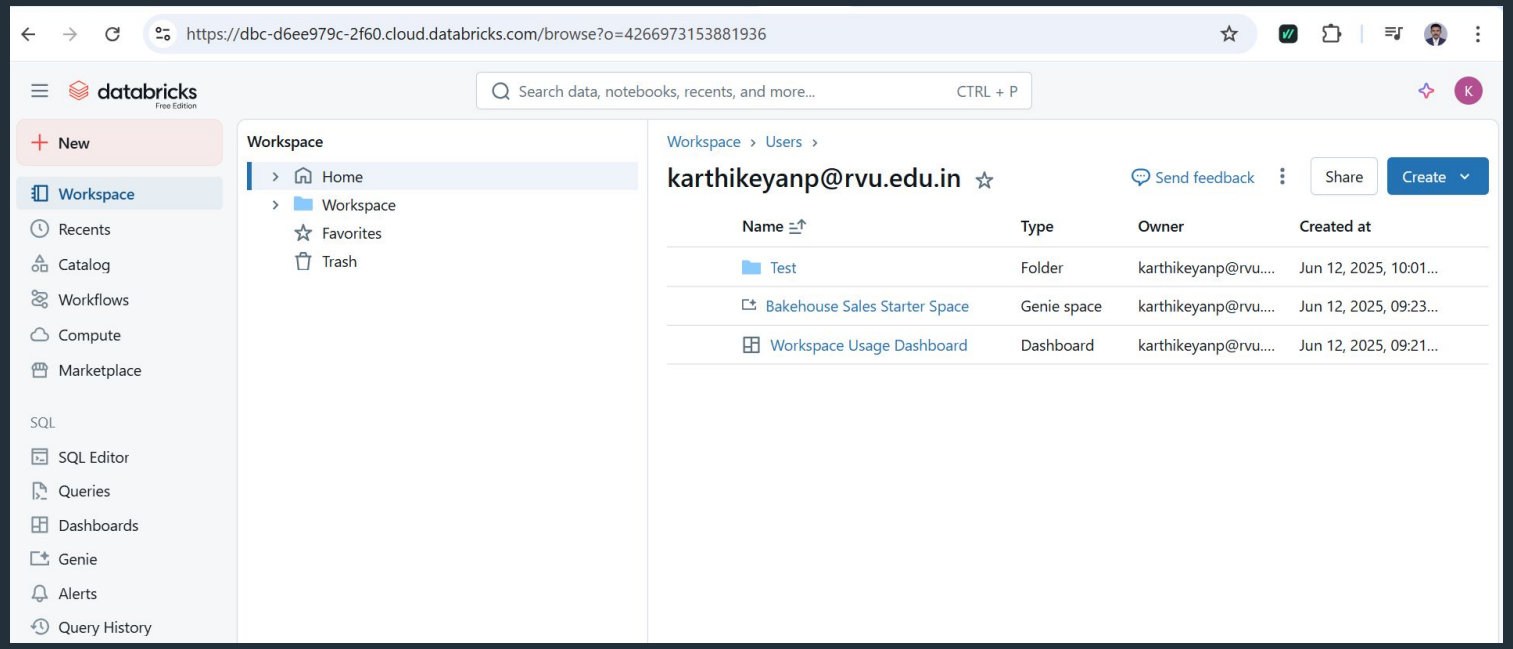
### Use your existing cloud account

If you have significant data volume in your cloud account or want to manage compute and storage.

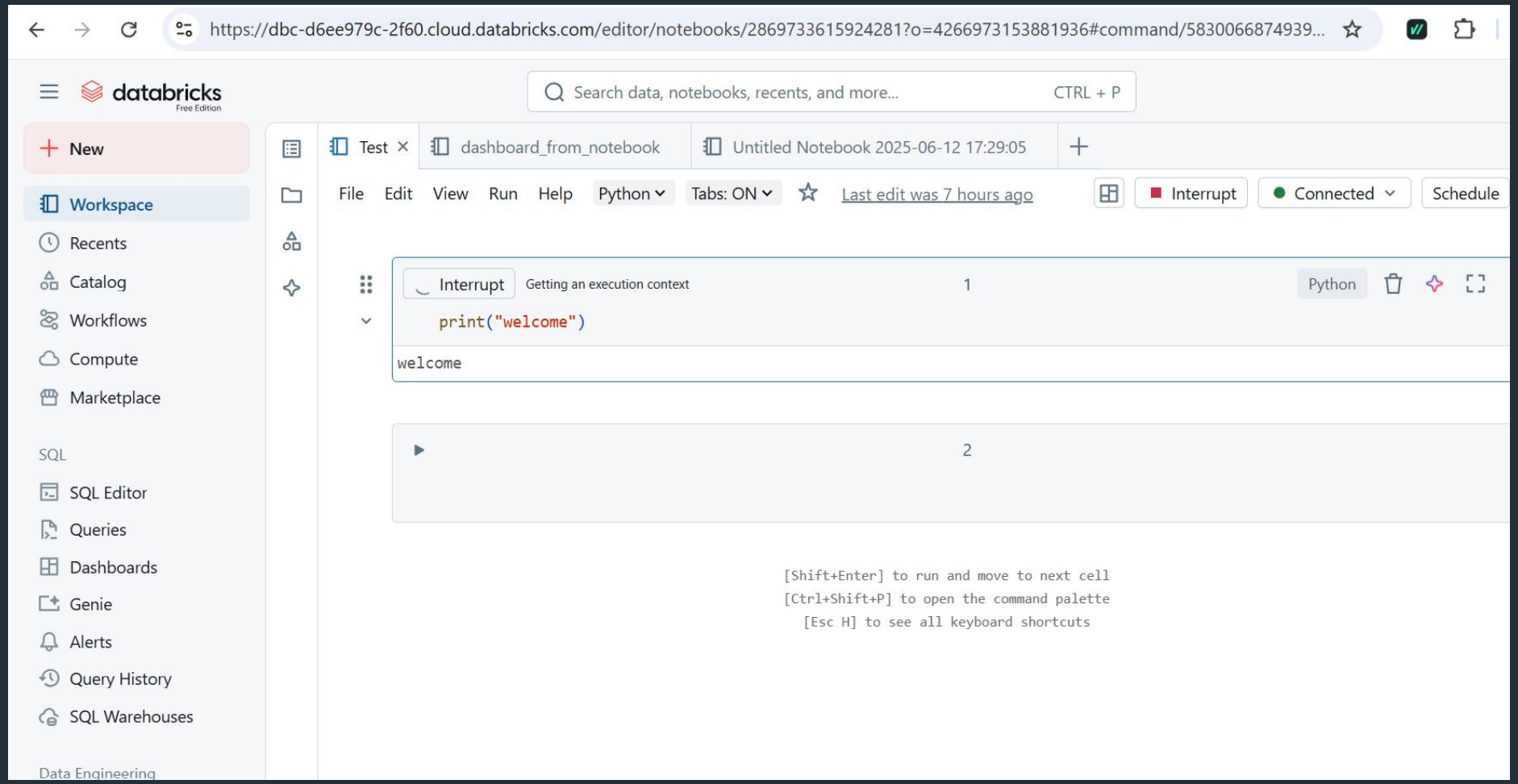
# How To Create Notebook In Databricks

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

## Step 2: Open the Workspace On the left sidebar, click on Workspace and create Test Workspace.



# Step 3: Create Notebook Click on the dropdown (right-click) in the workspace folder where you want to create the notebook. Click Create → Notebook.



**Step 4: Select Language Choose your programming language**: Python ,SQL, Scala and R

**Step 5: Start Writing Code Use cells to write:** Python code with %python (or directly if default is Python)SQL queries with %sql Markdown text with %md

**Step 6: Run the Notebook**

Click **Run All** to execute all cells.

Or click the **Run button** on individual cells.