Copyright Notice

DeepLearning.AI

Introduction to Data Engineering

Week 3

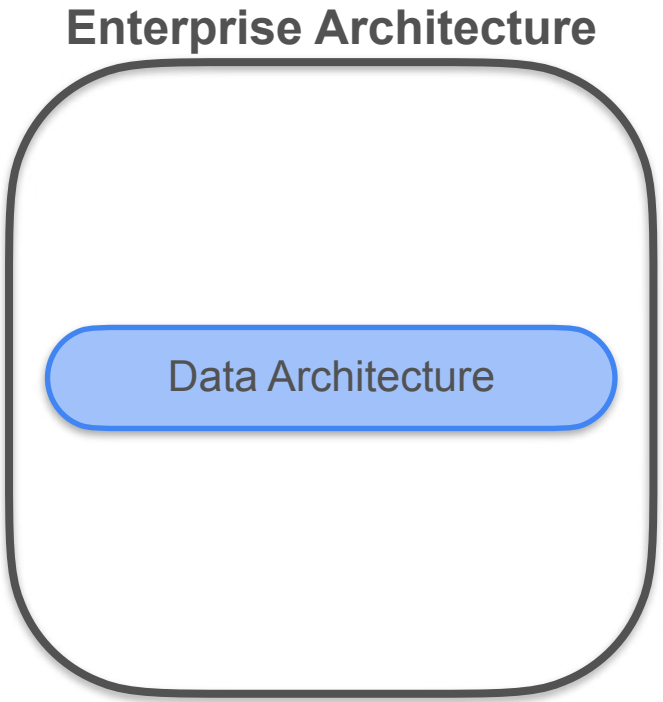
DeepLearning.AI

Data Architecture

Week 3 Overview

Week 3 Overview

- Enterprise & data architecture

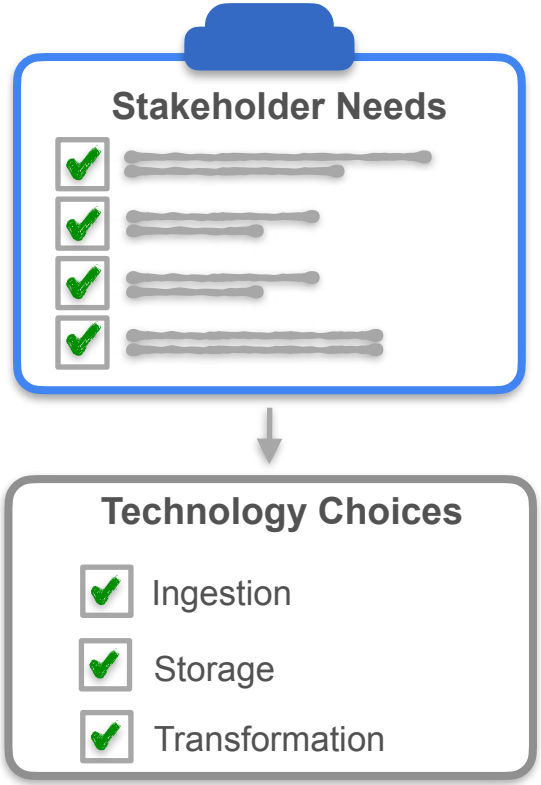


Week 3 Overview

- Enterprise & data architecture
- Specific architecture examples

Week 3 Overview

- Enterprise & data architecture
- Specific architecture examples
- Choosing technologies



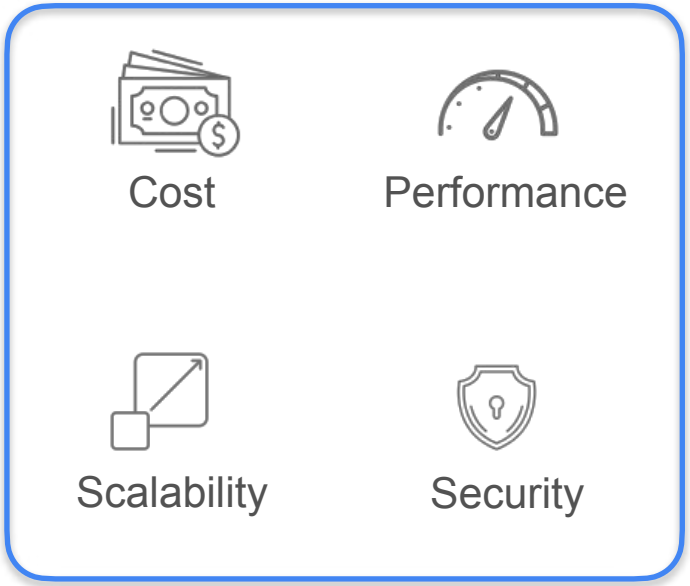
Week 3 Overview

- Enterprise & data architecture
- Specific architecture examples
- Choosing technologies
- Guiding architectural principles

Think like an architect!

Week 3 Overview

- Enterprise & data architecture
- Specific architecture examples
- Choosing technologies
- Guiding architectural principles
- Trade-off evaluation



Week 3 Overview

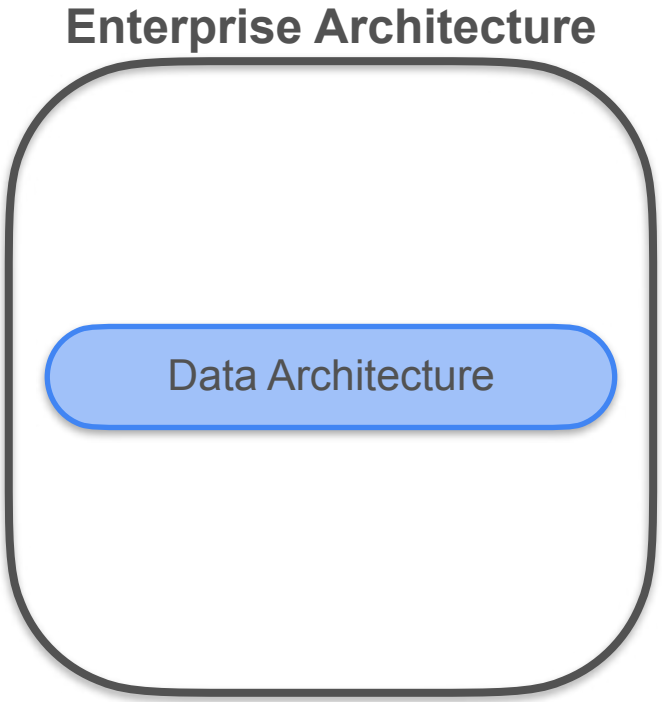
- Enterprise & data architecture
- Specific architecture examples
- Choosing technologies
- Guiding architectural principles
- Trade-off evaluation
- AWS Well-Architected Framework

DeepLearning.AI

Data Architecture

What is Data Architecture?

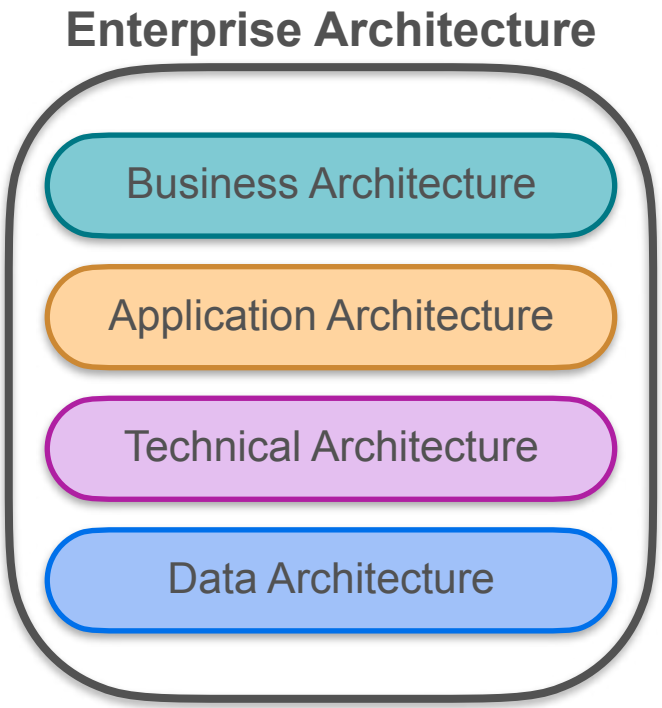
Enterprise Architecture



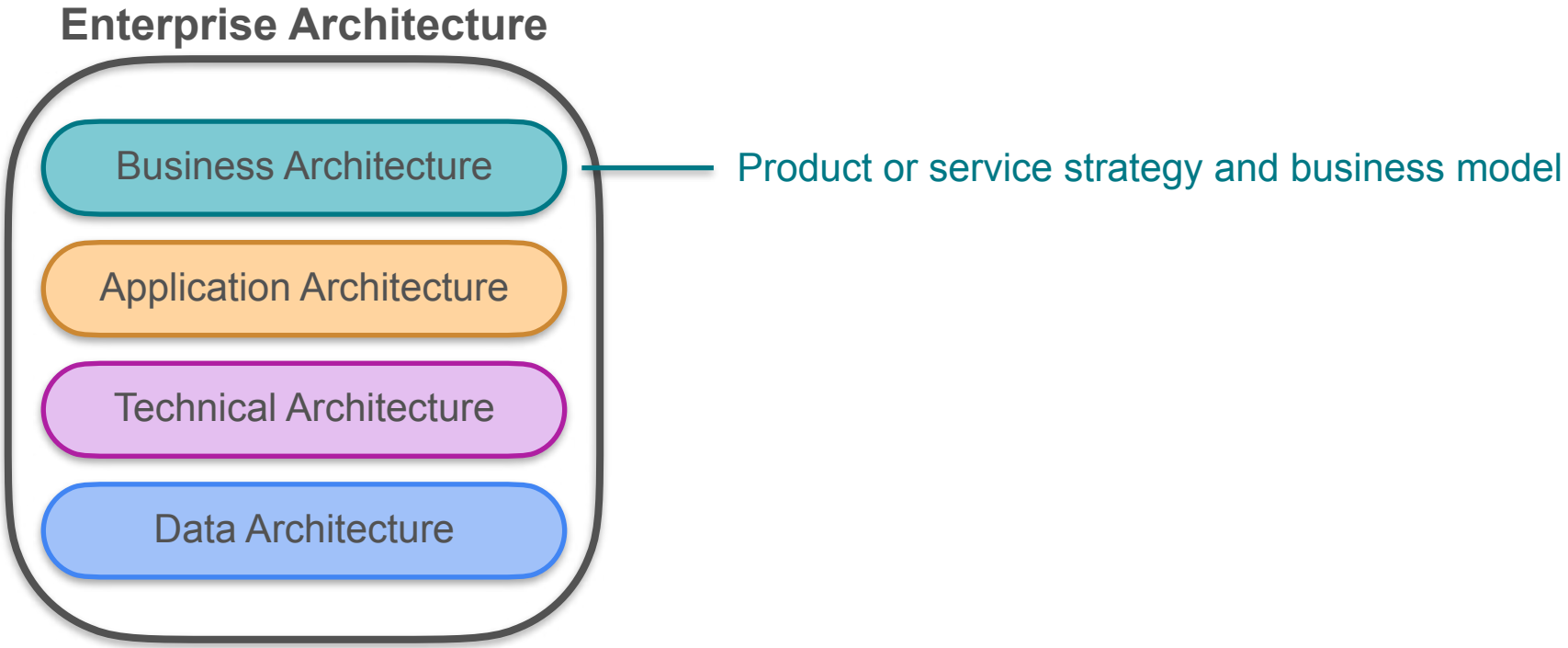
"the design of systems **to support change in an enterprise**, achieved by flexible and reversible decisions reached through a careful evaluation of trade-offs"

"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."

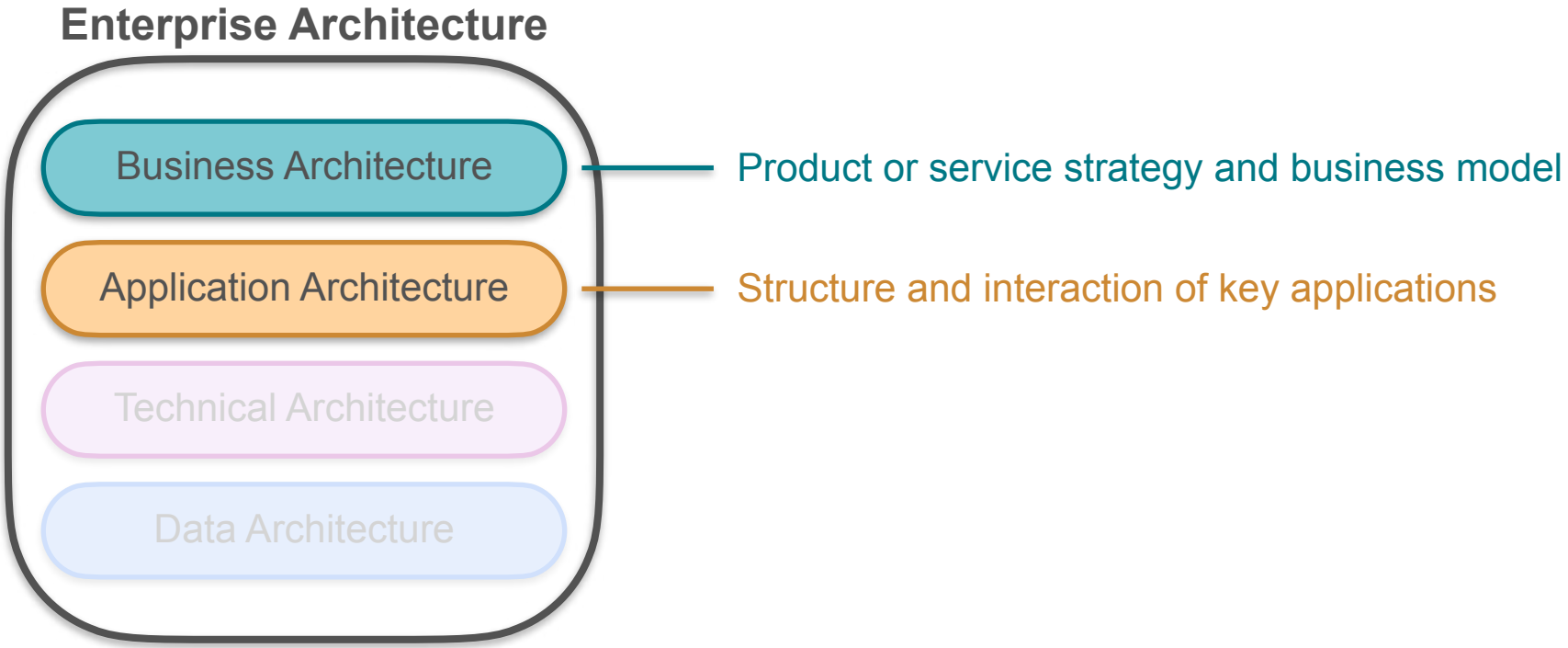
Enterprise Architecture



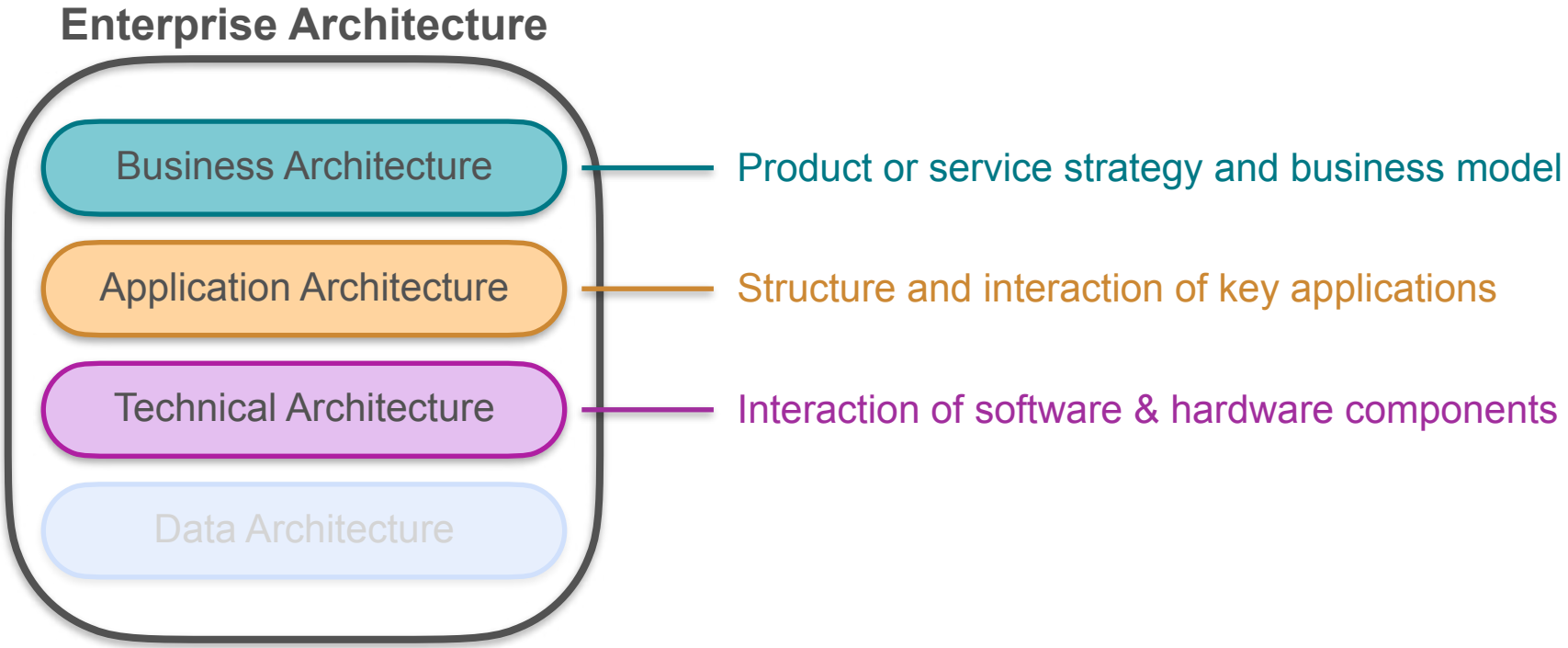
Enterprise Architecture



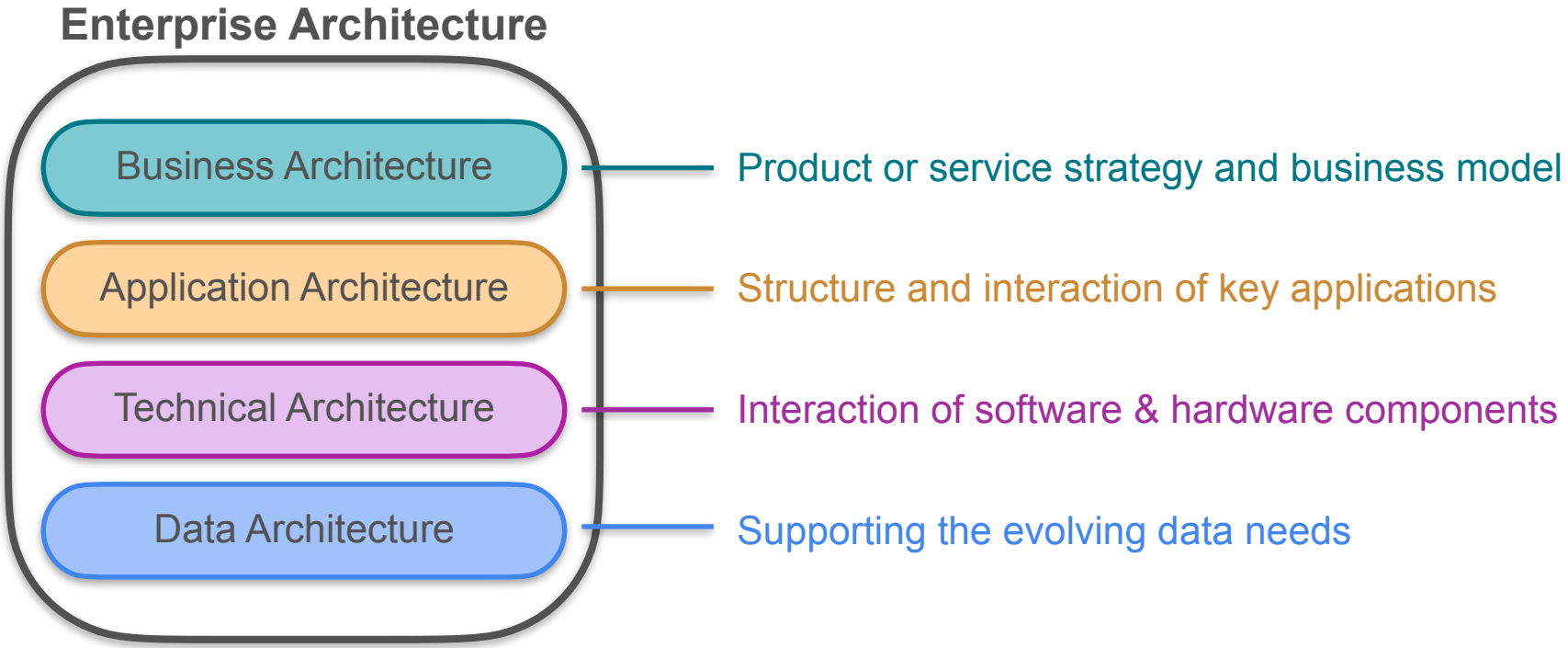
Enterprise Architecture



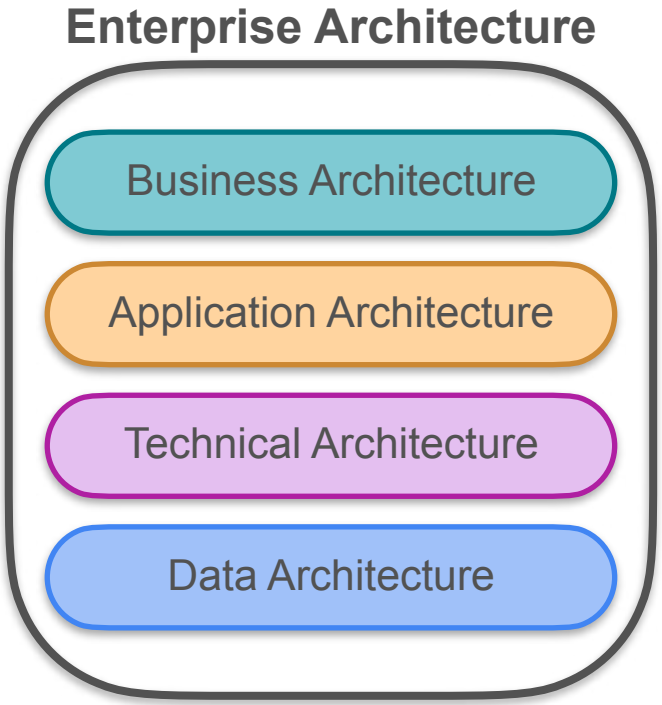
Enterprise Architecture



Enterprise Architecture

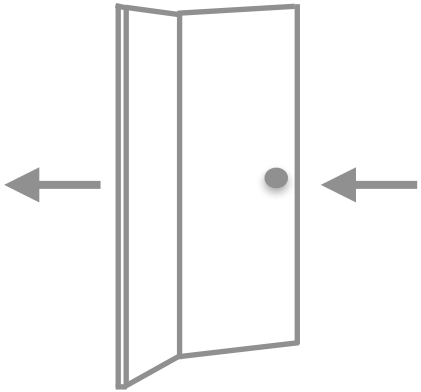


Enterprise Architecture

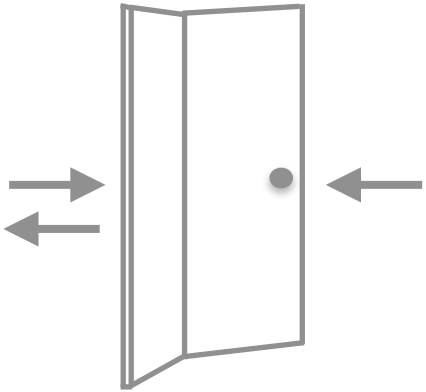


One-Way and Two-Way Door Decisions

Organization Decisions



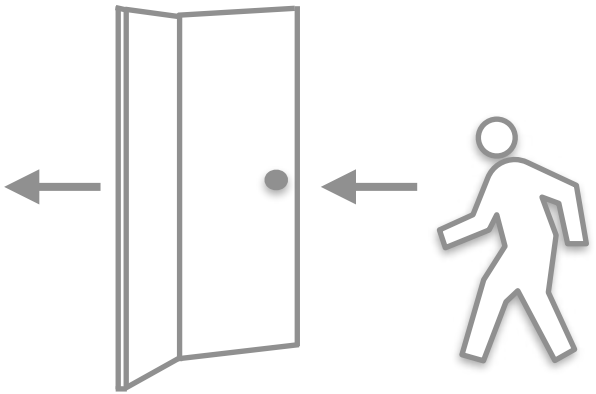
One-way



Two-way

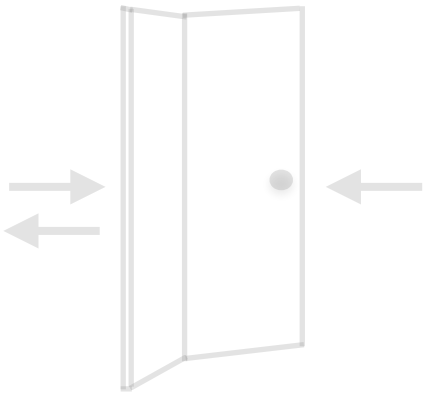
One-Way and Two-Way Door Decisions

Organization Decisions



One-way

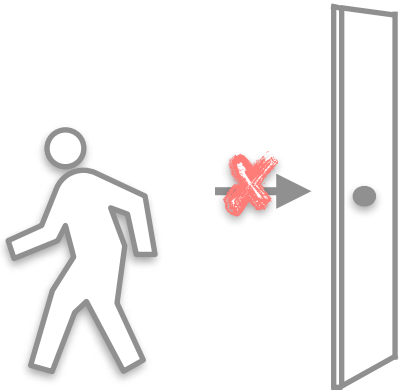
A decision that is almost impossible to reverse



Two-way

One-Way and Two-Way Door Decisions

Organization Decisions



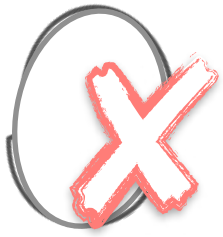
One-way

A decision that is almost impossible to reverse

Two-way

One-Way and Two-Way Door Decisions

Organization Decisions



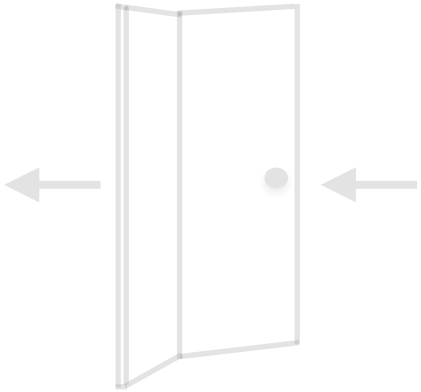
One-way

A decision that is almost impossible to reverse

Two-way

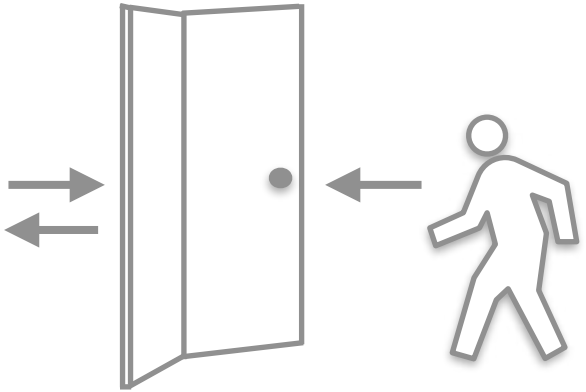
One-Way and Two-Way Door Decisions

Organization Decisions



One-way

A decision that is almost impossible to reverse

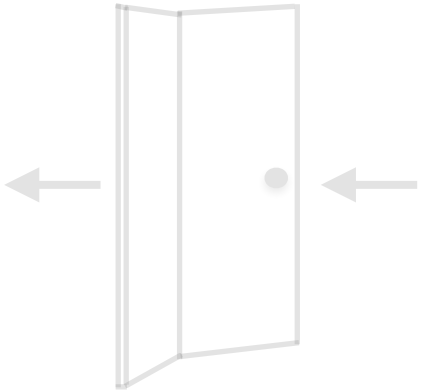


Two-way

An easily reversible decision

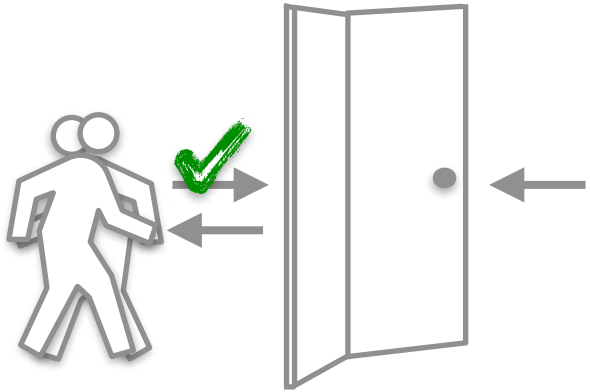
One-Way and Two-Way Door Decisions

Organization Decisions



One-way

A decision that is almost impossible to reverse



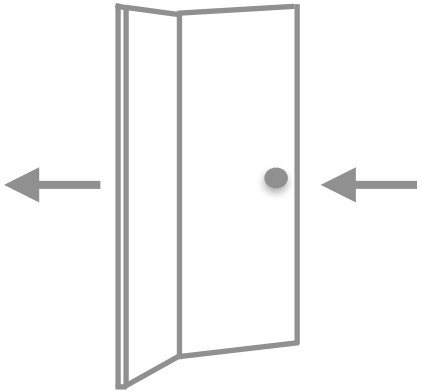
Two-way

An easily reversible decision

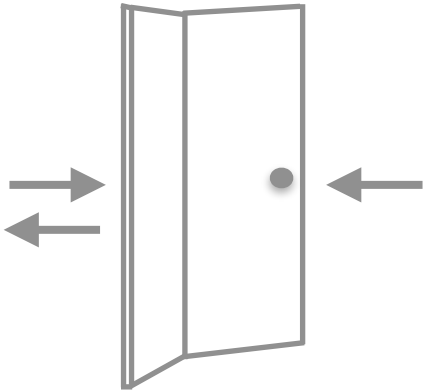
Two-Way Door Decision

Reversible Decisions

Organization Decisions



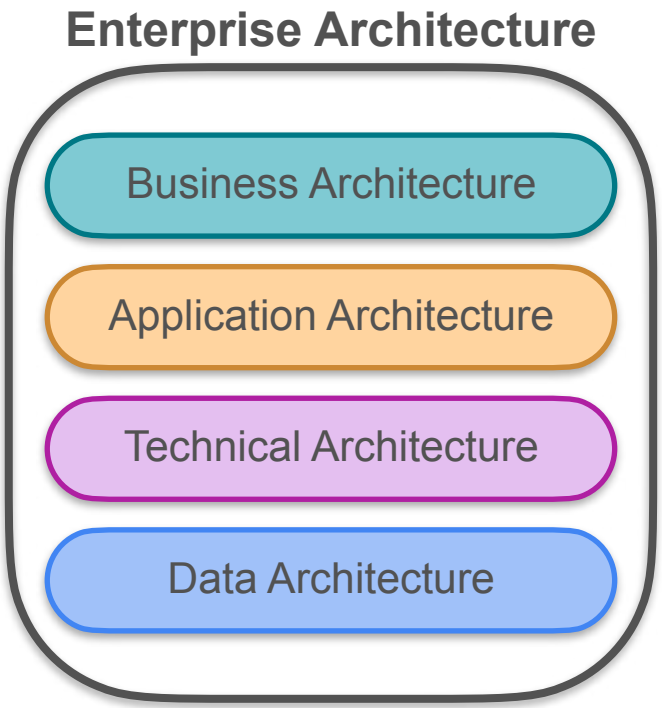
One-way



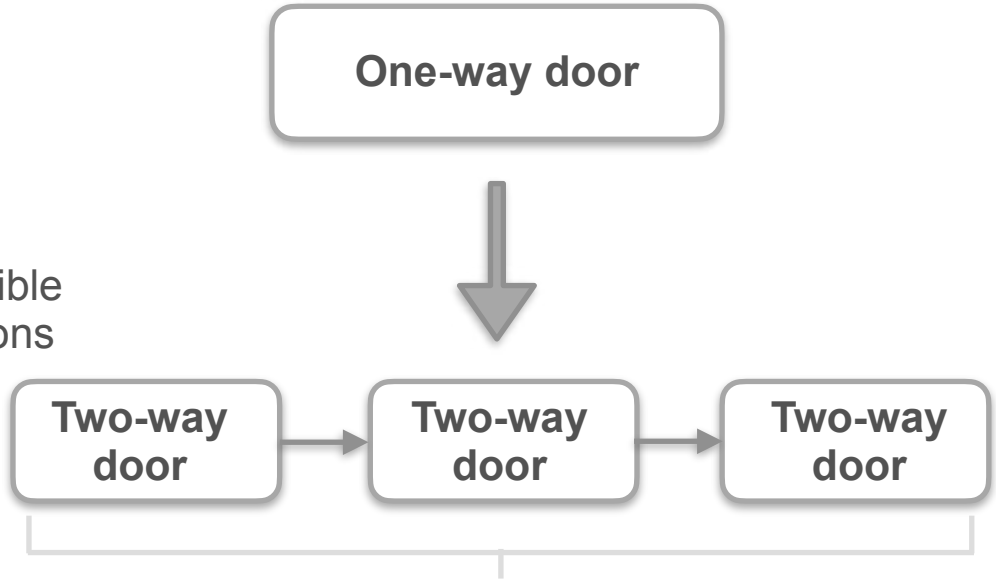
Two-way

Reversible Decisions

Reversible Decisions



Reversible Decisions



DeepLearning.AI

Data Architecture

Conway’s Law

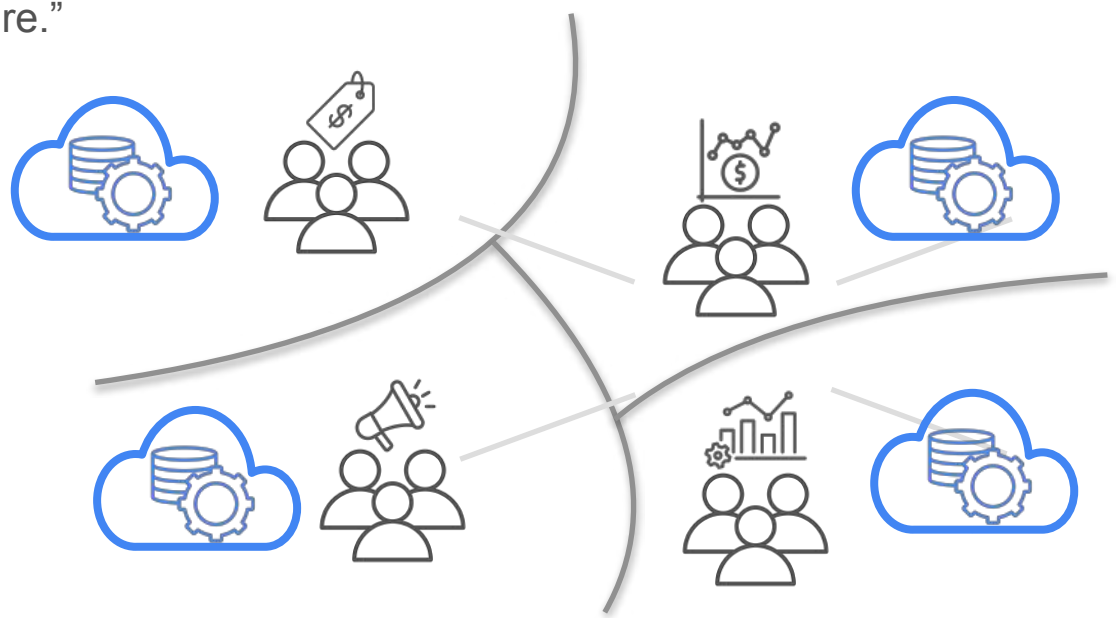
Conway's Law

"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."



DeepLearning.AI

Data Architecture

Principles of Good Data Architecture

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

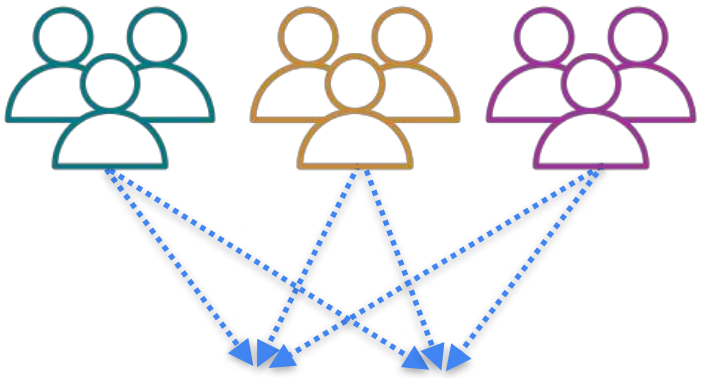
Unspoken but understood priorities

Principles of Good Data Architecture

1. Choose common components wisely
4. Architecture is leadership

How data architecture impacts other teams and individuals

Common Components



Common Components

Examples

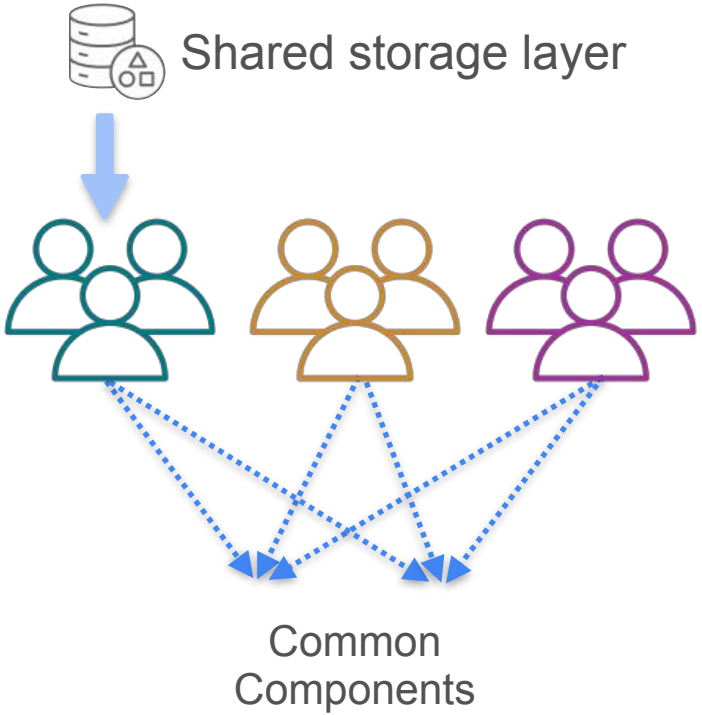
Object Storage

Version-Control Systems

Observability & Monitoring Systems

Processing Engines

Common Components



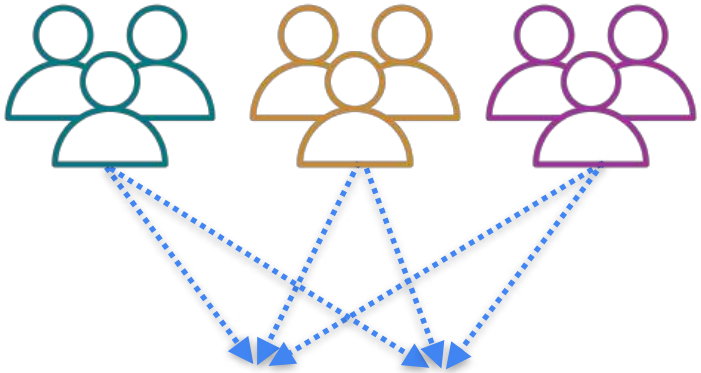
Common components:

- Facilitate team collaboration
- Break down silos

“Wise” Choice:

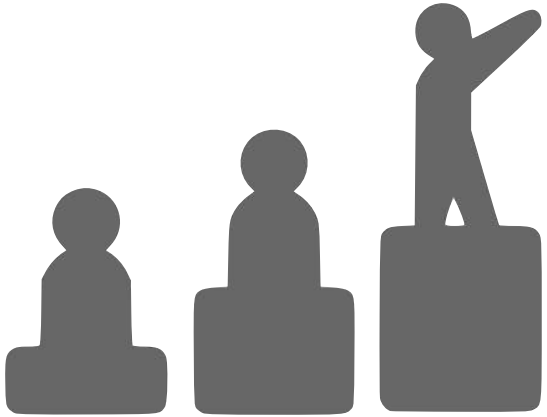
- Identify tools that benefit all teams
- Avoid a one-size-fits-all approach

Architecture Leadership



Common Components

Architecture is leadership



Seek mentorship from data architects

DeepLearning.AI

Data Architecture

Always Architecting

Application Programming Interface

Application Programming Interface

Amazon Web Services
(AWS)

Application
Programming Interface

stable and predictable

Loosely coupled system

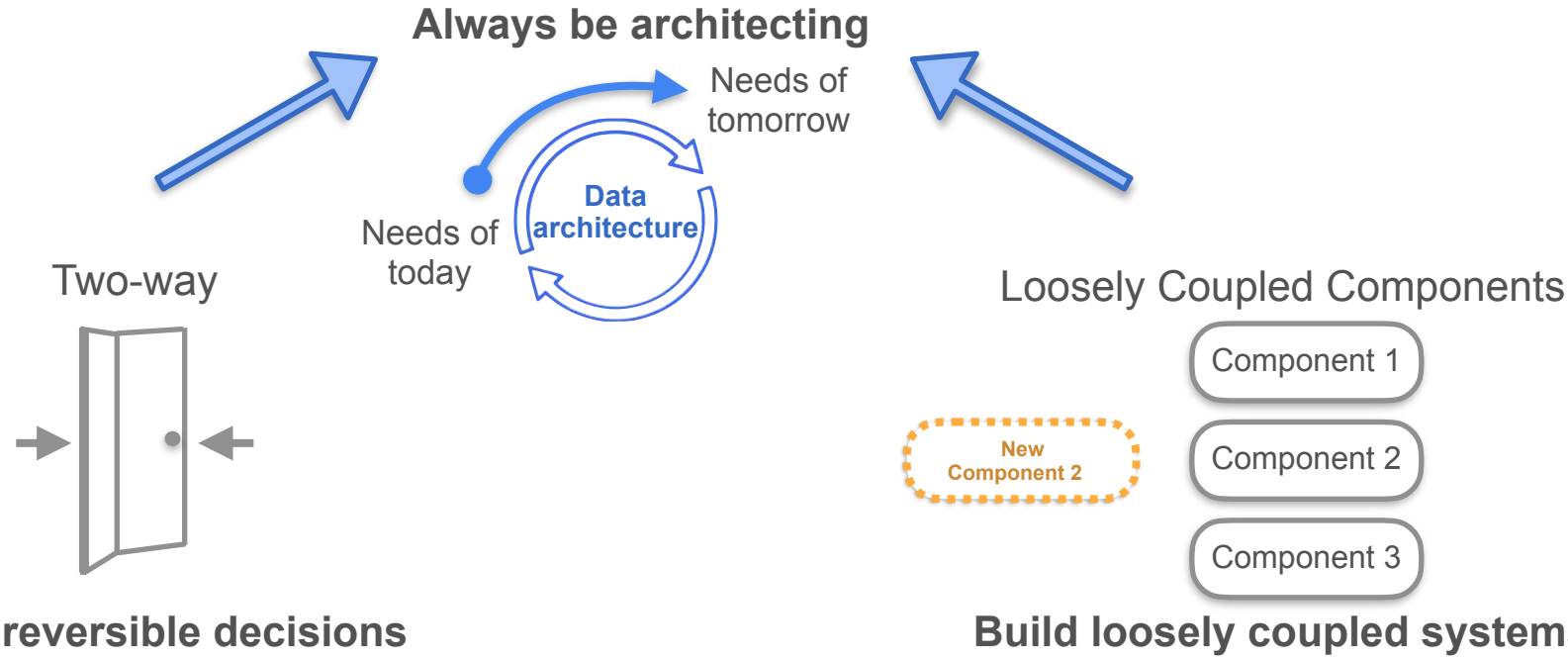
Always Architecting

Make reversible decisions

Build loosely coupled systems

Always be architecting

Always Architecting



DeepLearning.AI

Data Architecture

When Your Systems Fail

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

Reliability

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

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

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

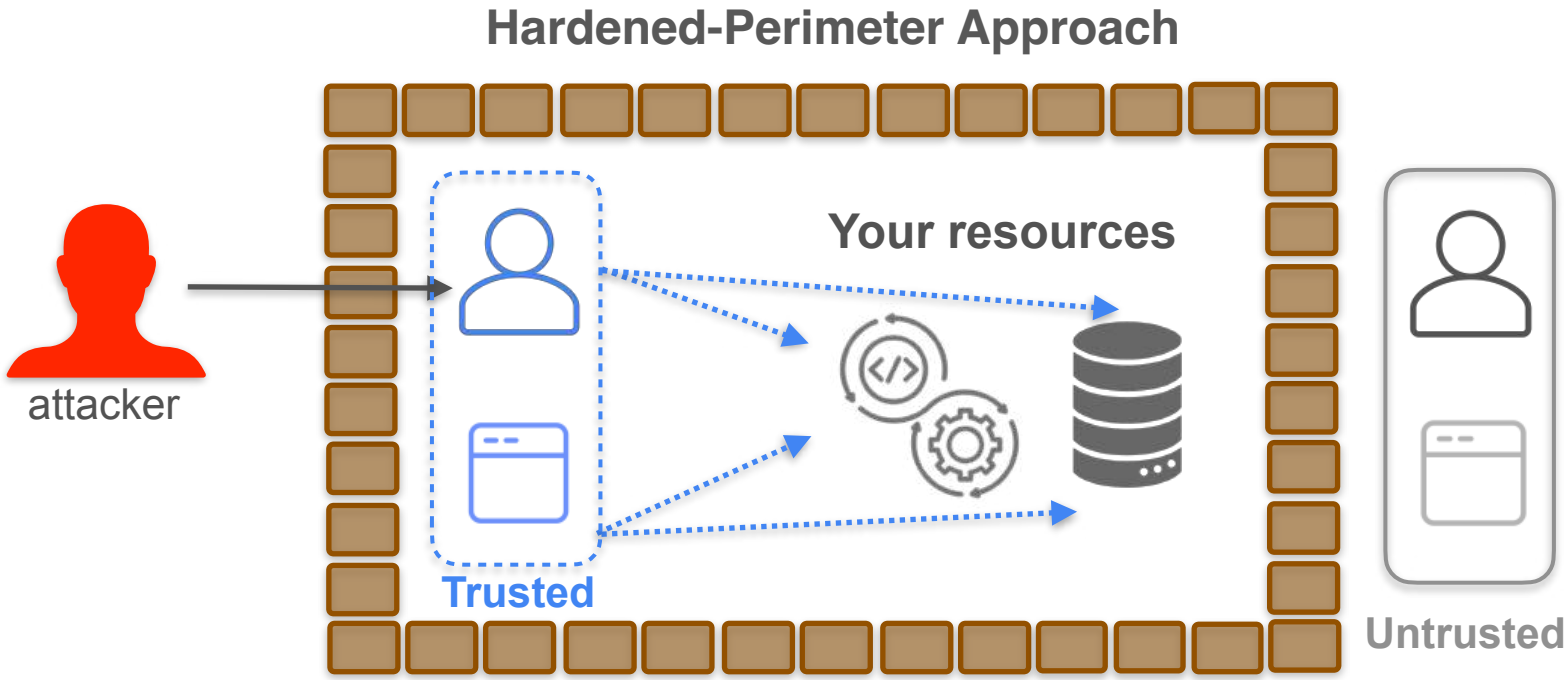
A definition of the acceptable state after recovery

For example: Consider the maximum acceptable data loss

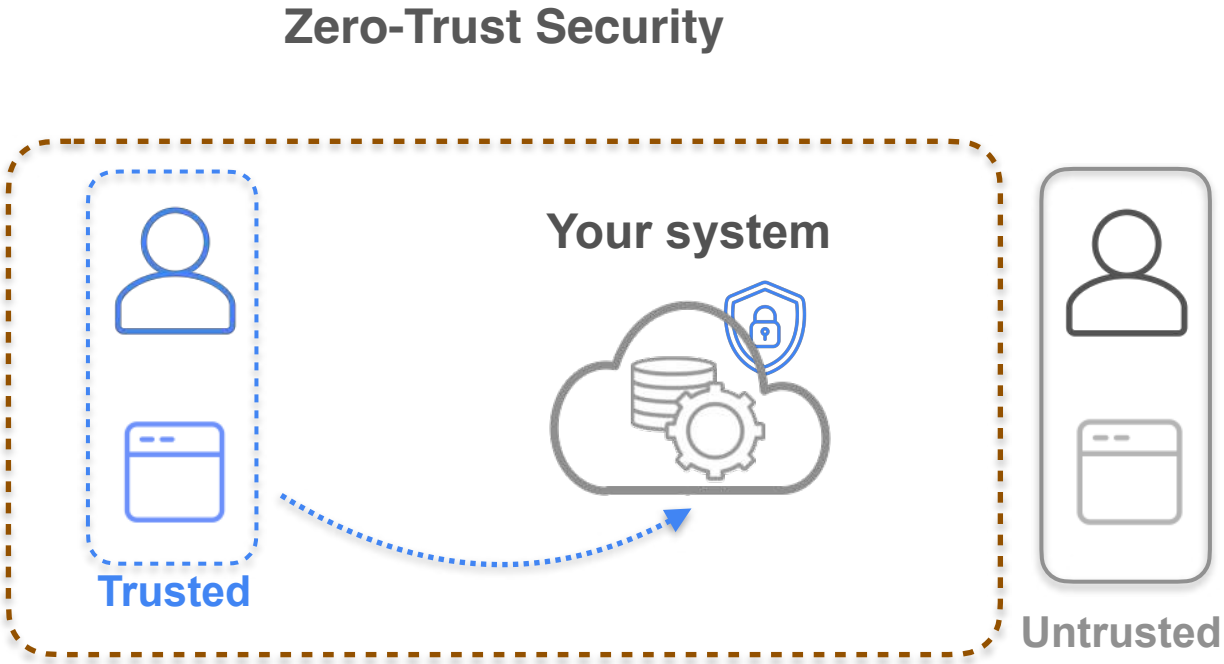
Prioritize Security

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

Prioritize Security



Prioritize Security

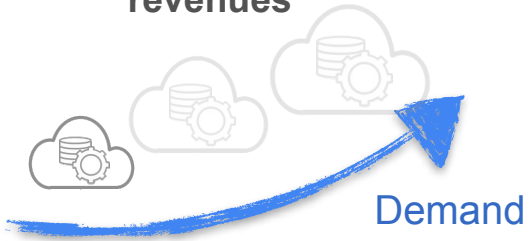


Architecting for Scalability & Embrace FinOps

Large Unforeseen Costs

OR

Missed Opportunities for revenues



Plan for Failure!

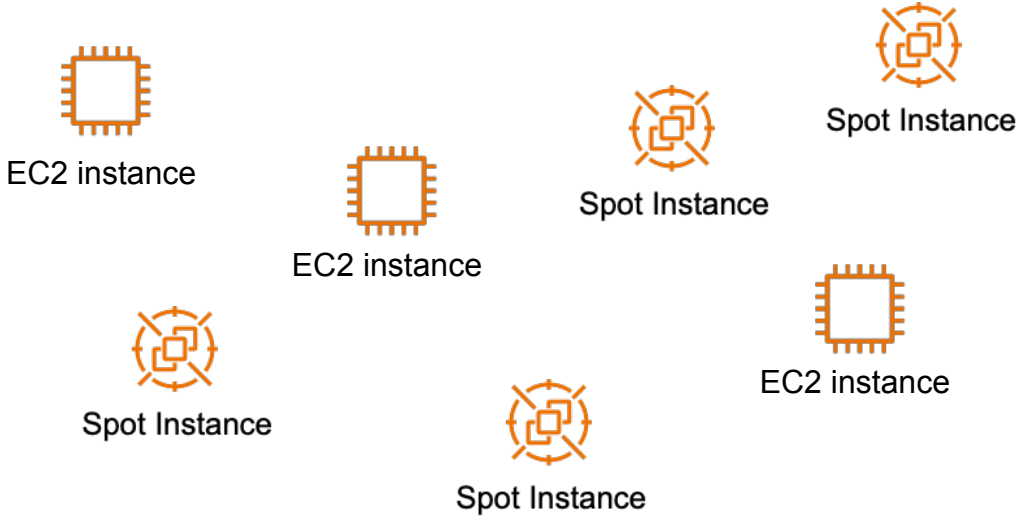
Embrace FinOps

Architect for scalability

Embrace FinOps

Manage cloud cost

How to optimize a daily job in terms of cost and performance?



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

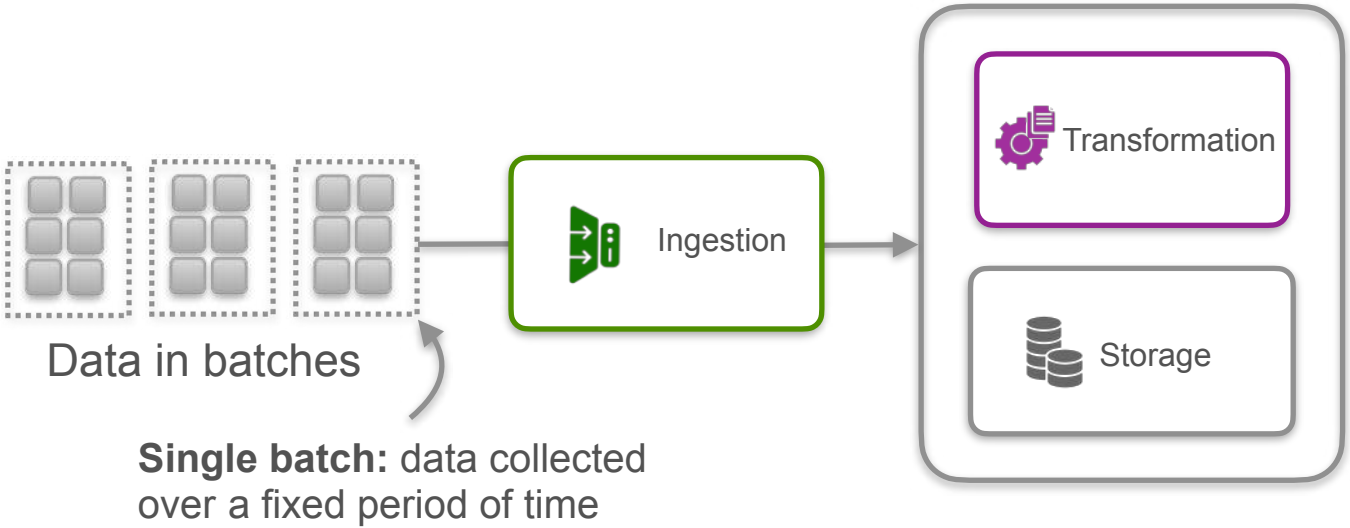
DeepLearning.AI

Data Architecture

Batch Architectures

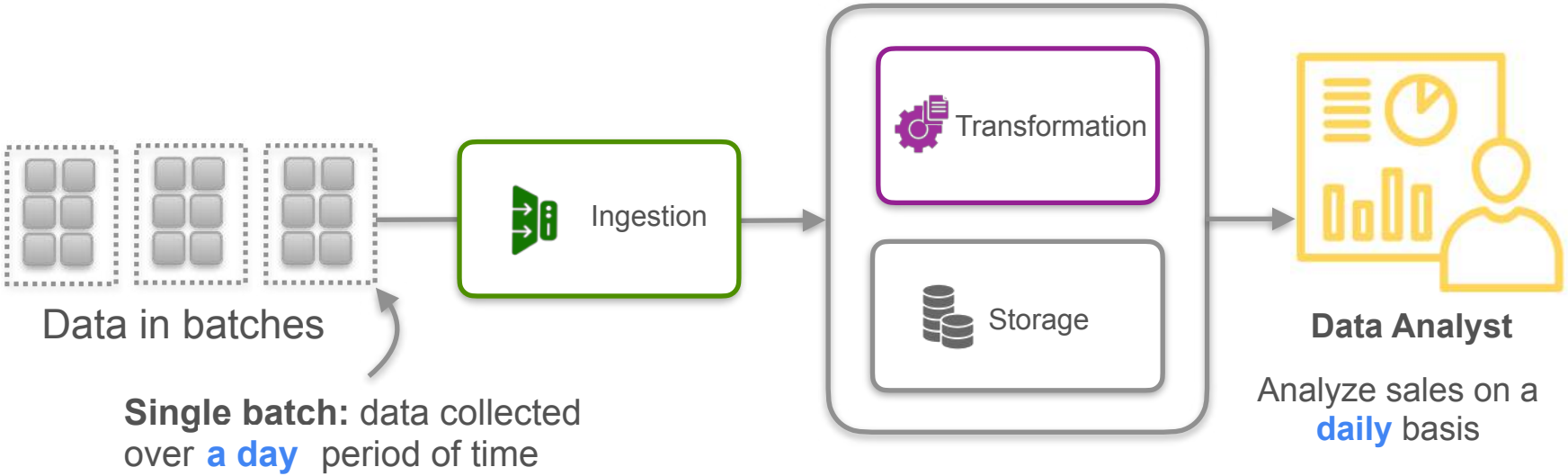
Batch Data Architecture

Real-time analysis is not critical

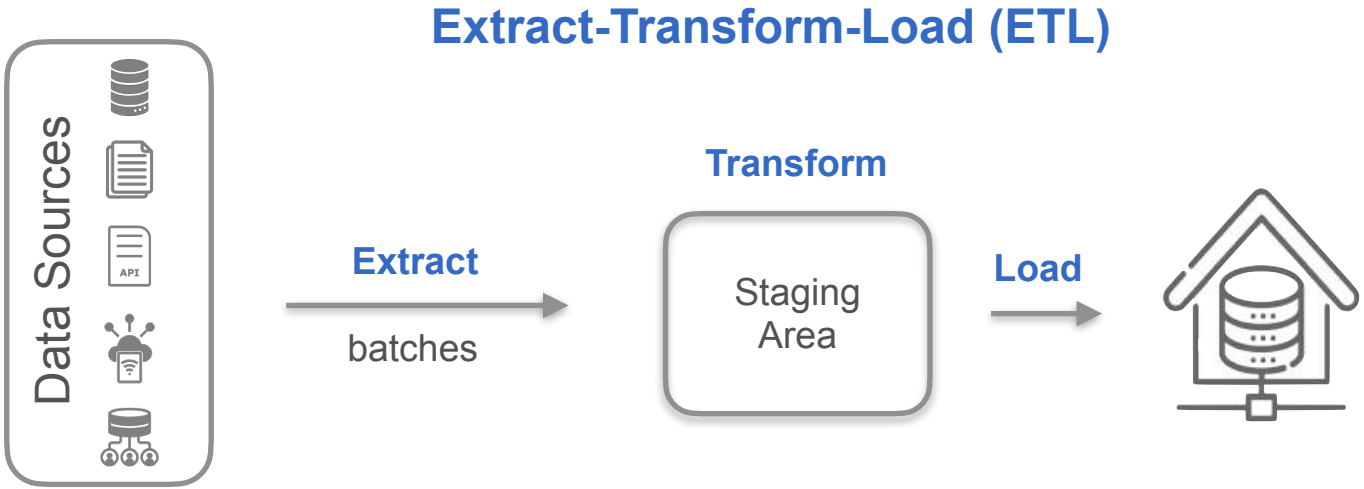


Batch Data Architecture

Real-time analysis is not critical



Example of Batch Architectures



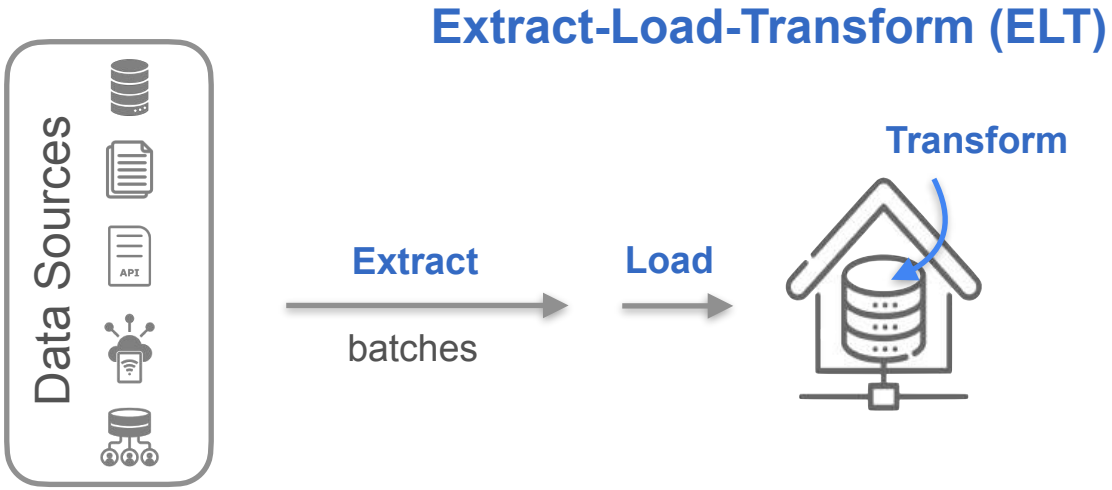
Generation

Ingestion

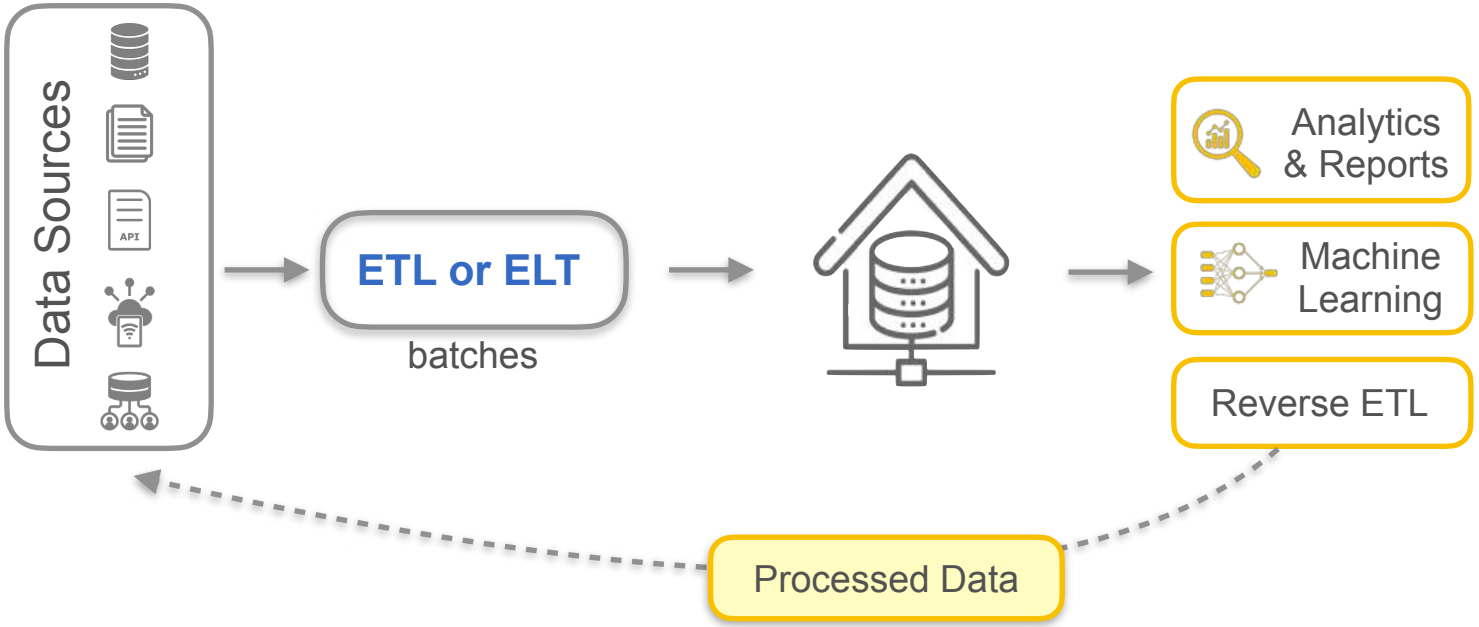
Transformation

Storage

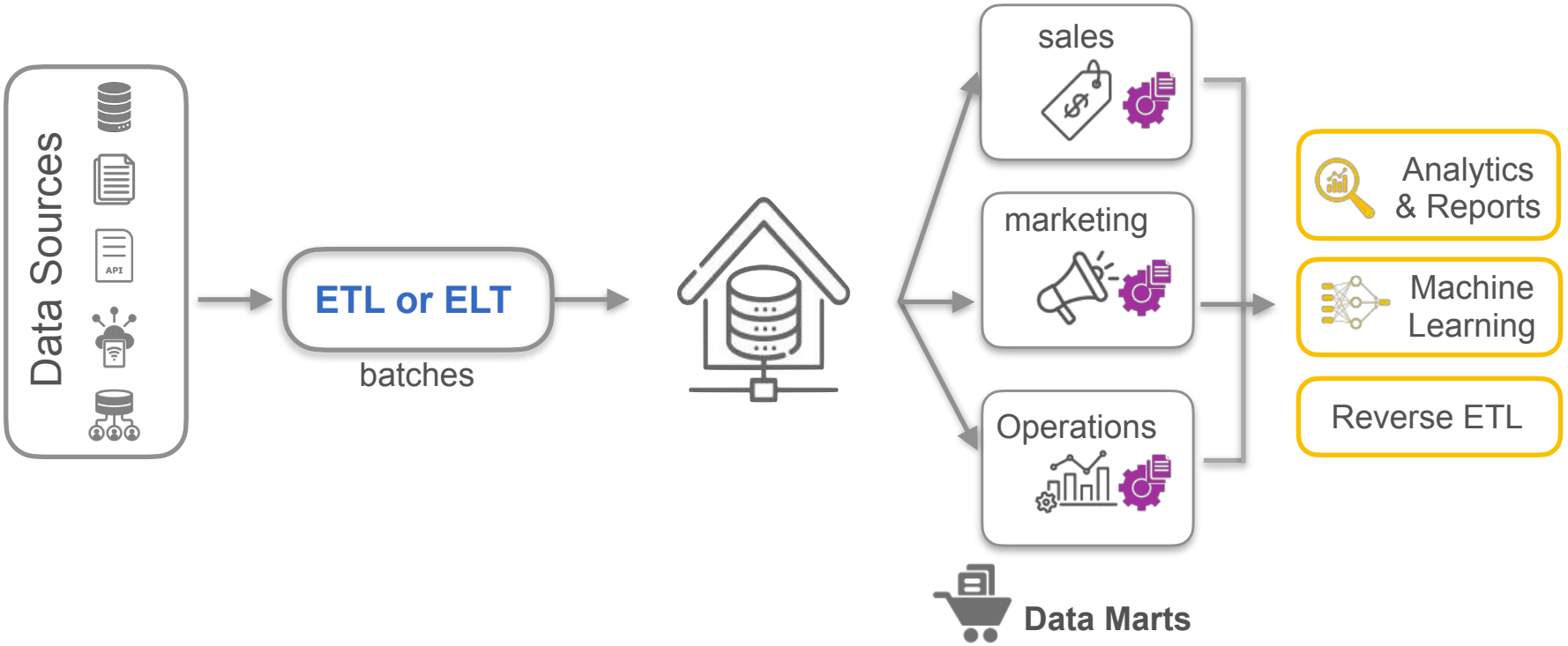
Example of Batch Architectures



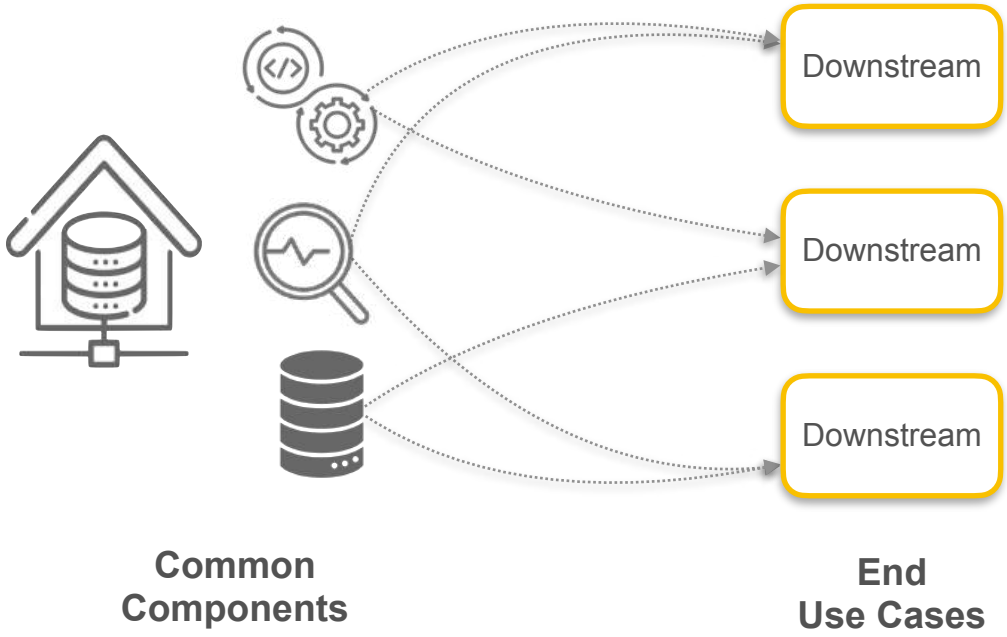
Example of Batch Architectures



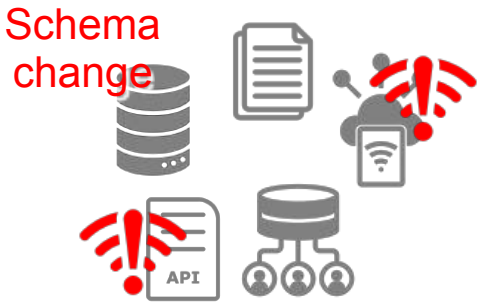
Example of Batch Architectures

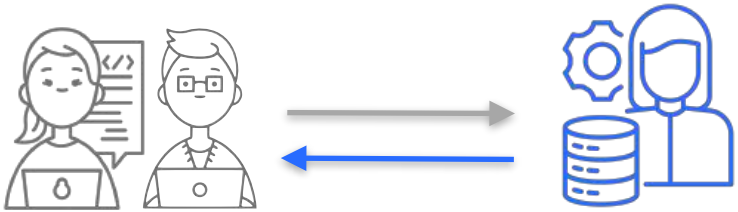


Choose Common Components



Planning for Failure

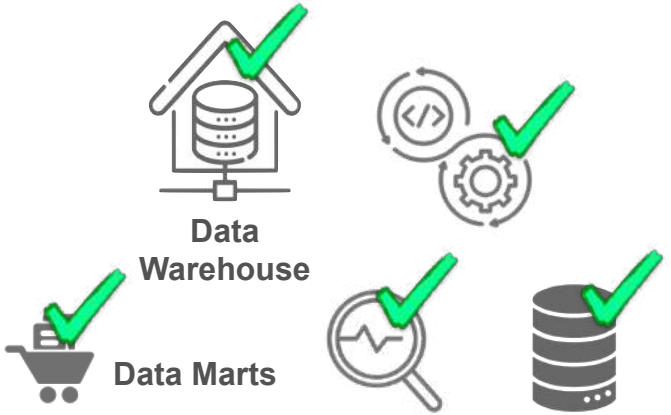




Source System Owners

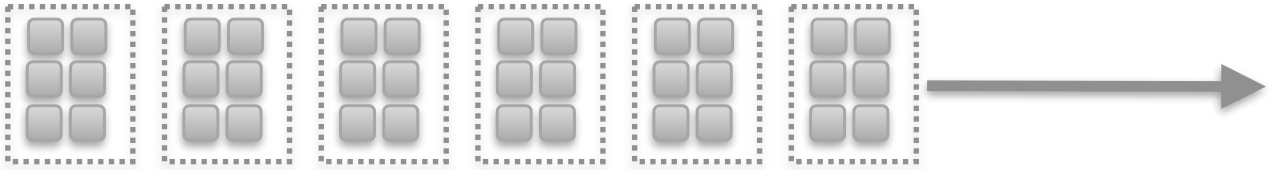
Data Engineer

Availability and reliability specs



Make Reversible Decisions

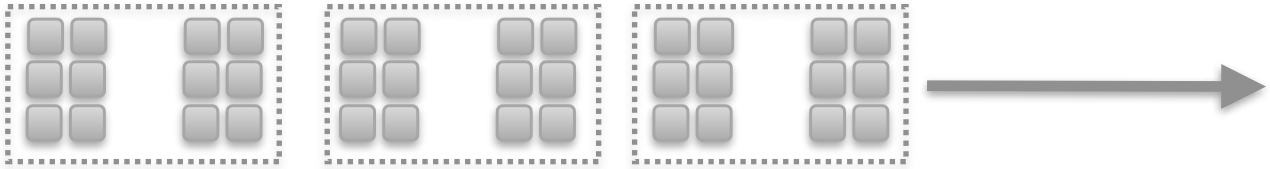
Build flexibility into your system



Ingest 1 day's worth of data

Make Reversible Decisions

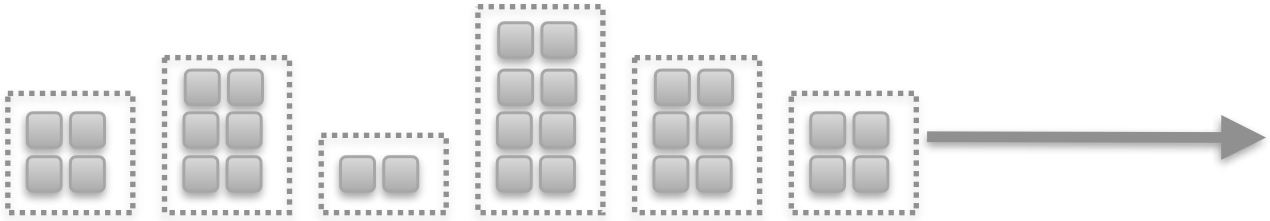
Build flexibility into your system



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

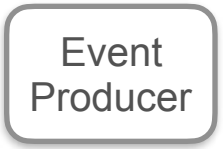
DeepLearning.AI

Data Architecture

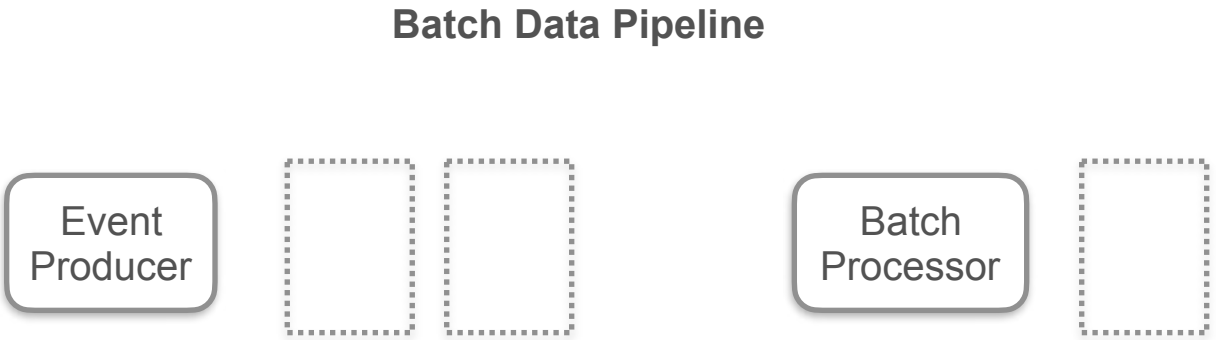
Streaming Architectures

Data - Stream of Events

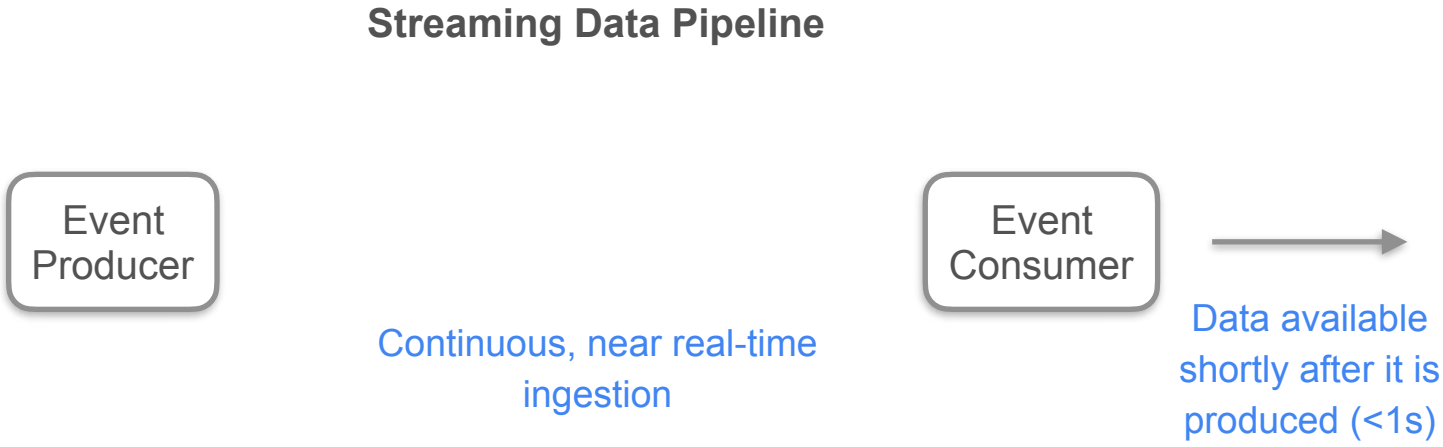
At its source, data is a continuous stream of events



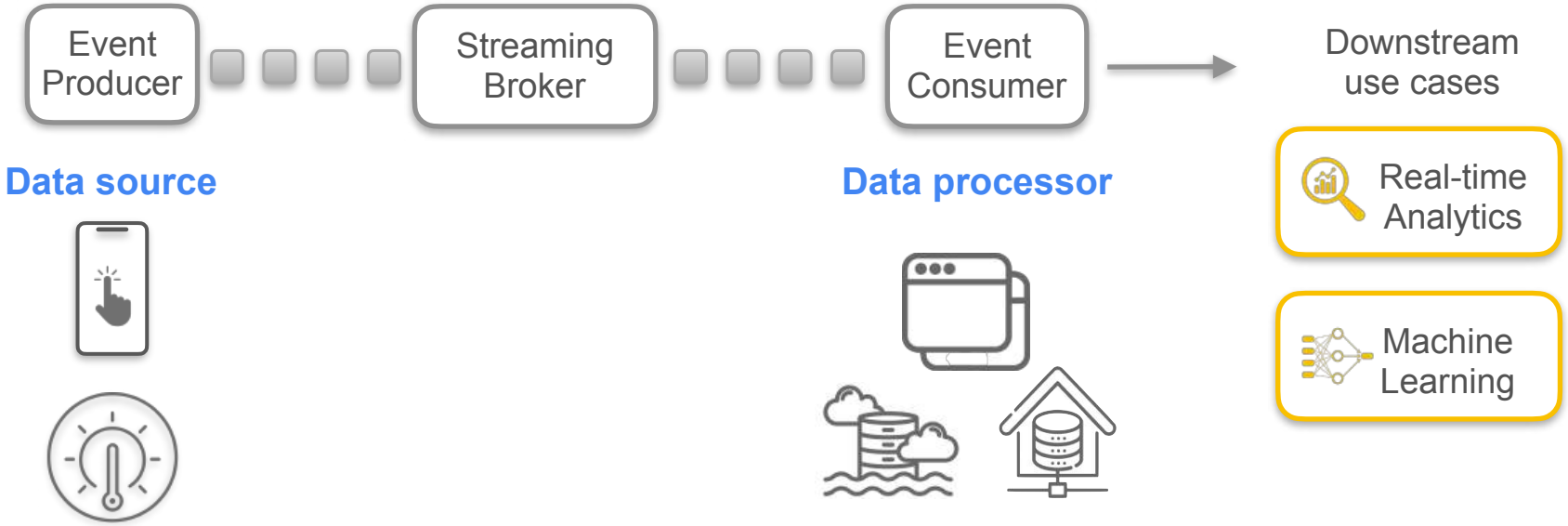
Data - Stream of Events



Data - Stream of Events



Streaming Systems



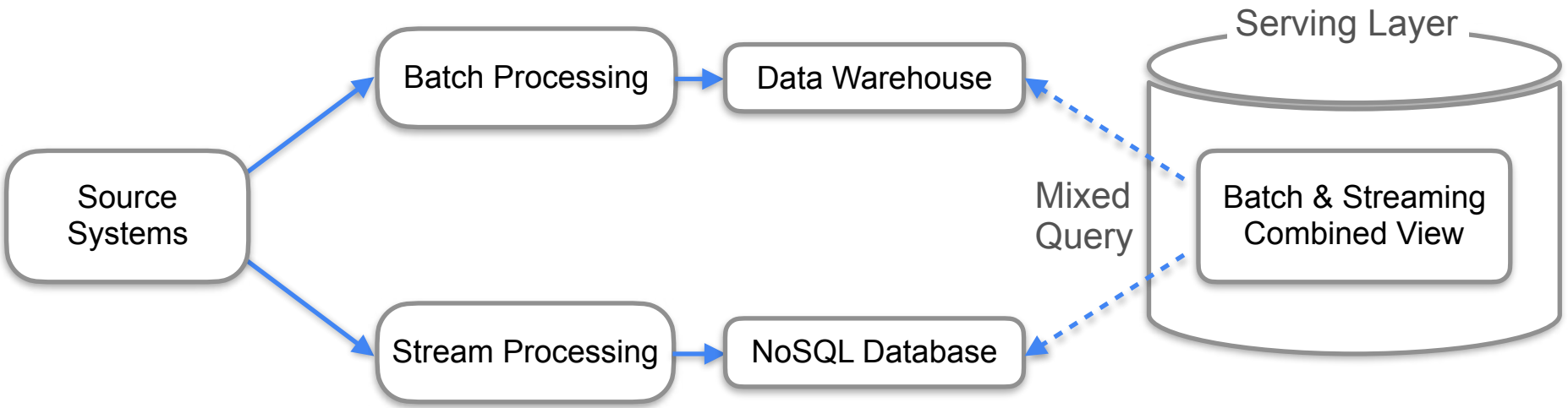
Streaming Frameworks

Event Streaming Platform

Streaming & Real-time Analytics

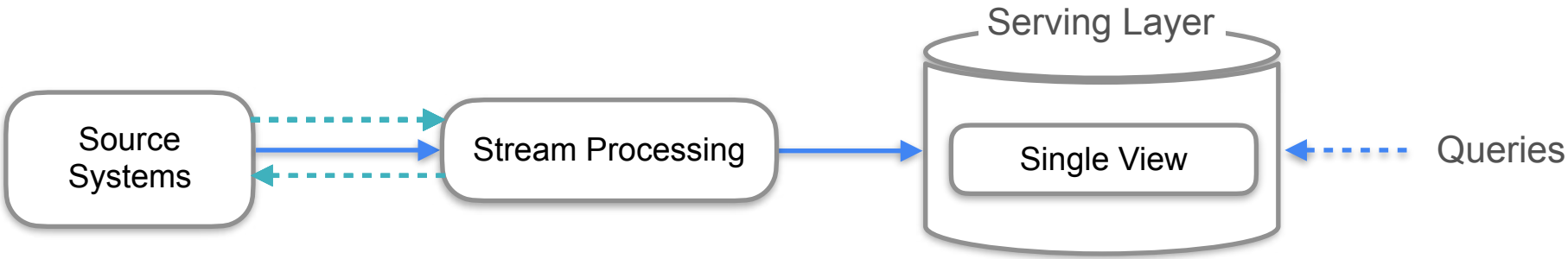
Lambda Architecture

Challenges: managing parallel systems with different code bases



Kappa Architecture

A true event-based architecture



- Retain some historical data
- Replay large chunks of retained data

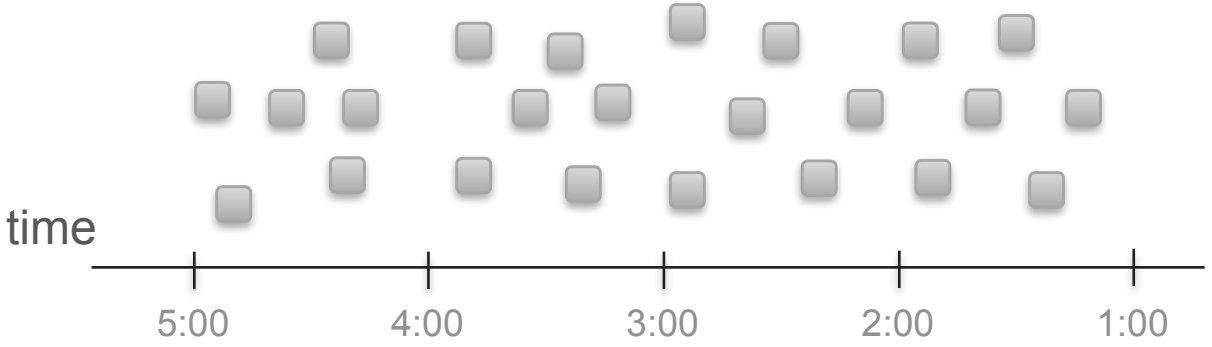
Unified Batch & Streaming

Unifying multiple code paths

Google Dataflow

beam

Data viewed as events

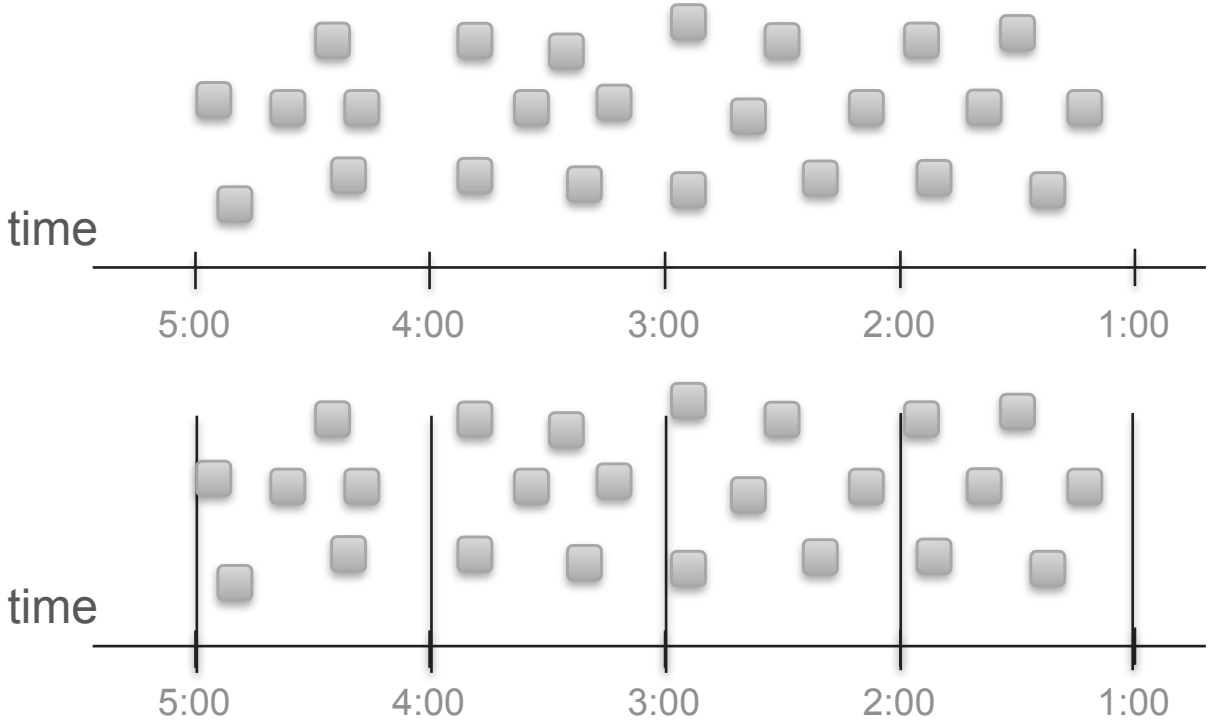


Unified Batch & Streaming

Unbounded data:
real-time event streams

Same code to process both types of data

Bounded data:
data in batches



Unified Batch & Streaming

Flink

DeepLearning.AI

Data Architecture

Architecting for Compliance

Architecting for Compliance

General Data Protection Regulation (GDPR)

Enacted in European Union in 2018

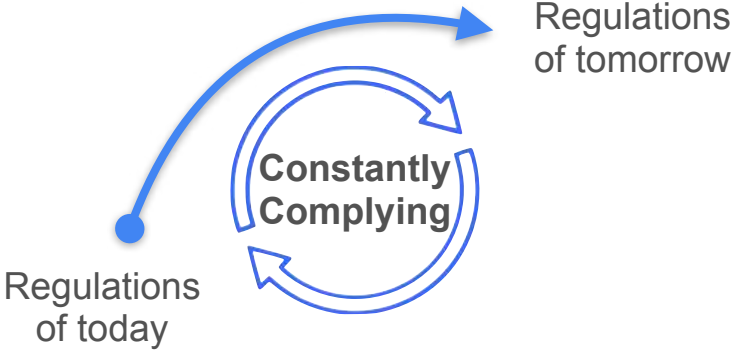
Personal Data

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

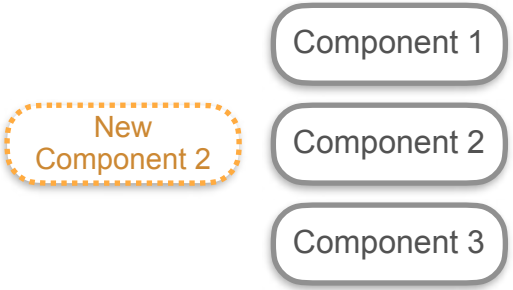
Right to have your data deleted

Architecting for Compliance

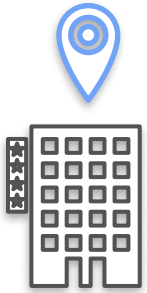
GDPR and similar regulations have been enacted globally



Loosely Coupled Components



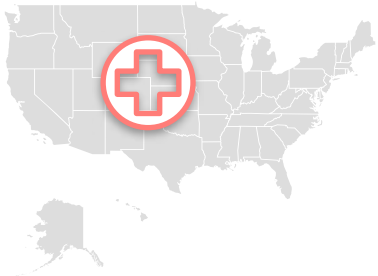
Architecting for Compliance



Company

Customer

Industry



Health Insurance Portability and Accountability Act (HIPAA)

Financial Data

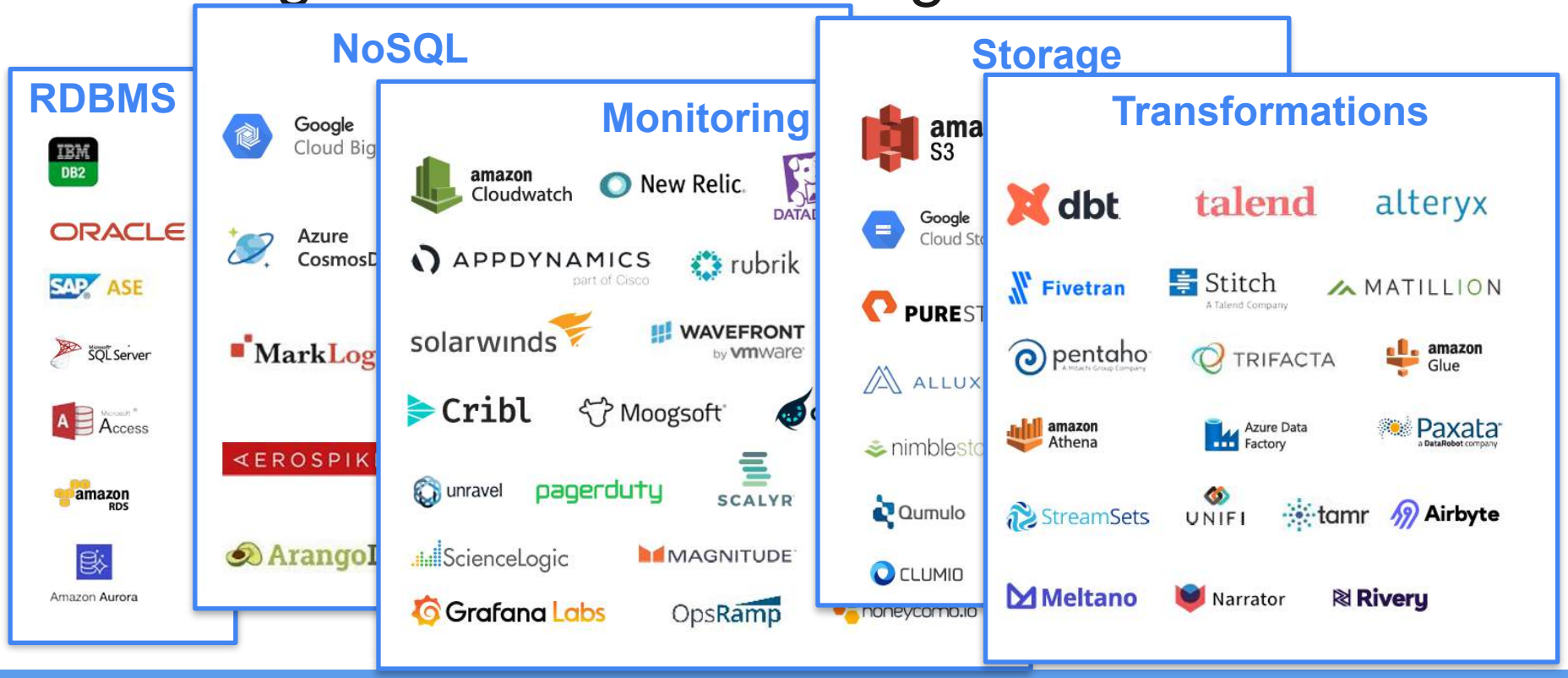
Sarbanes Oxley Act in the U.S

DeepLearning.AI

Choosing the Right Technologies

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!

Choosing Tools and Technologies - Considerations

Location

On-Premises

Cloud

Hybrid

Other Considerations

Cost Optimization

Team's size & capabilities

Build

vs

Buy

DeepLearning.AI

Choosing the Right Technologies

Location

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

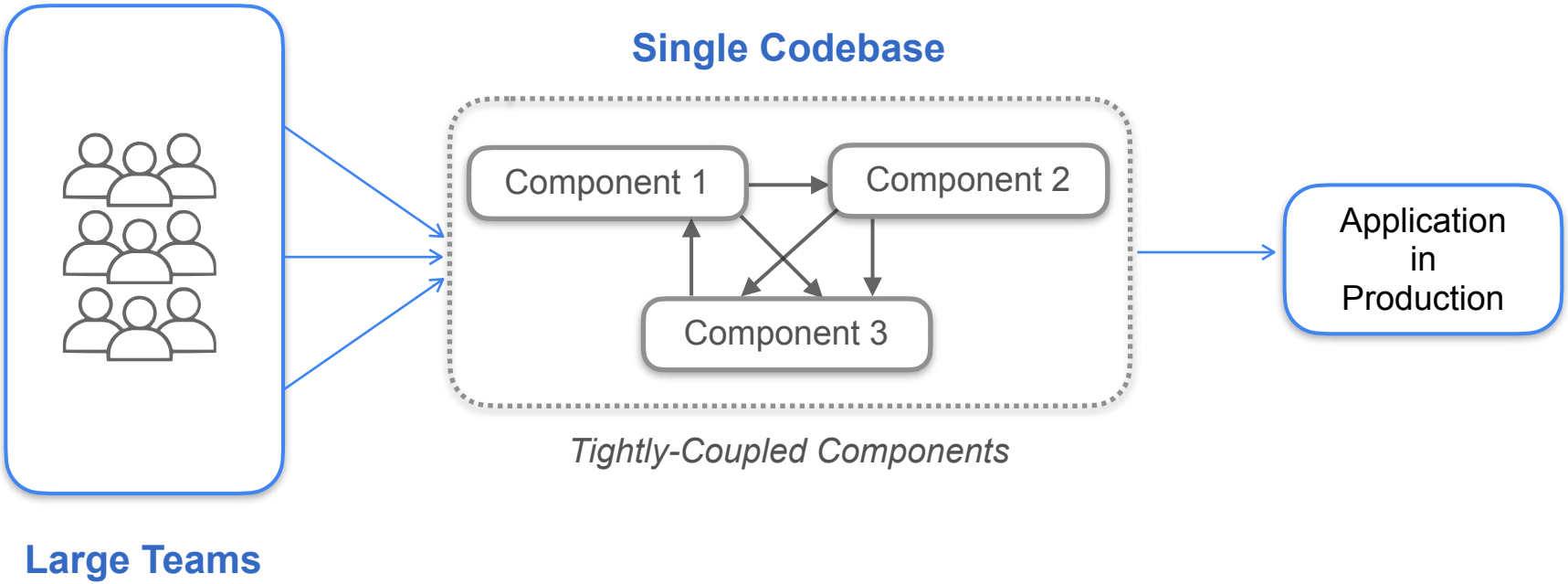
Flexibility and scalability

DeepLearning.AI

Choosing the Right Technologies

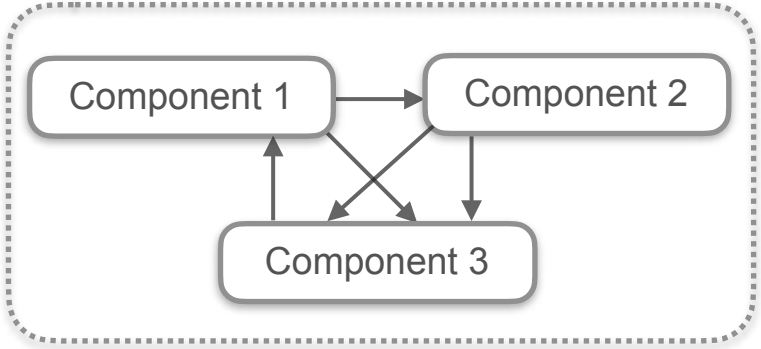
Monolith versus modular systems

Monolithic Systems



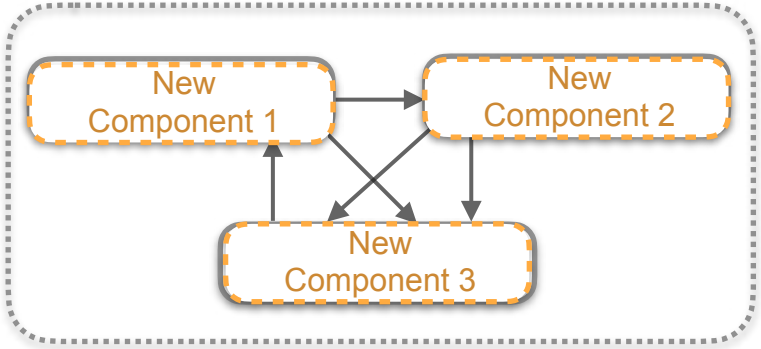
Monolithic Systems

Single Codebase



Tightly-Coupled Components

Hard to maintain

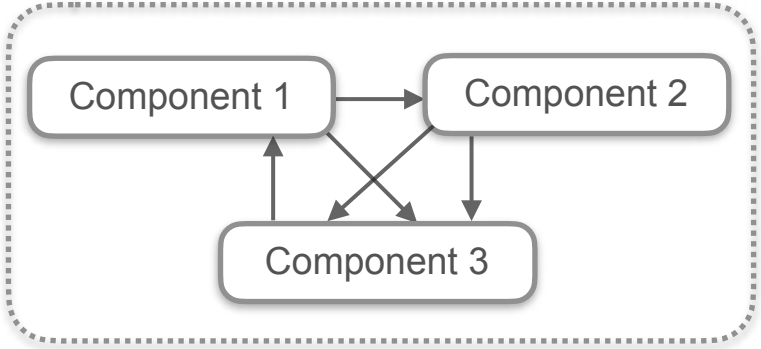


Entire application has to be re-written

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

Monolithic VS Modular Systems

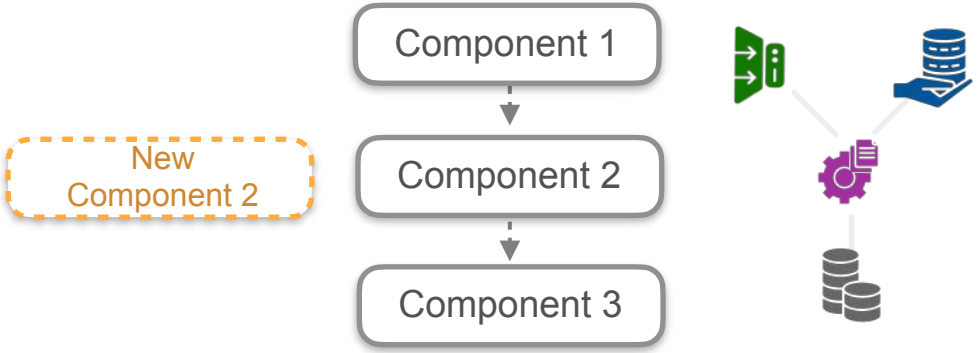
Monolithic



Tightly-Coupled Components

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

Modular



Loosely-Coupled Components

- Interoperability
- Flexible & reversible decisions
- Continuous improvement

DeepLearning.AI

Choosing the Right Technologies

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

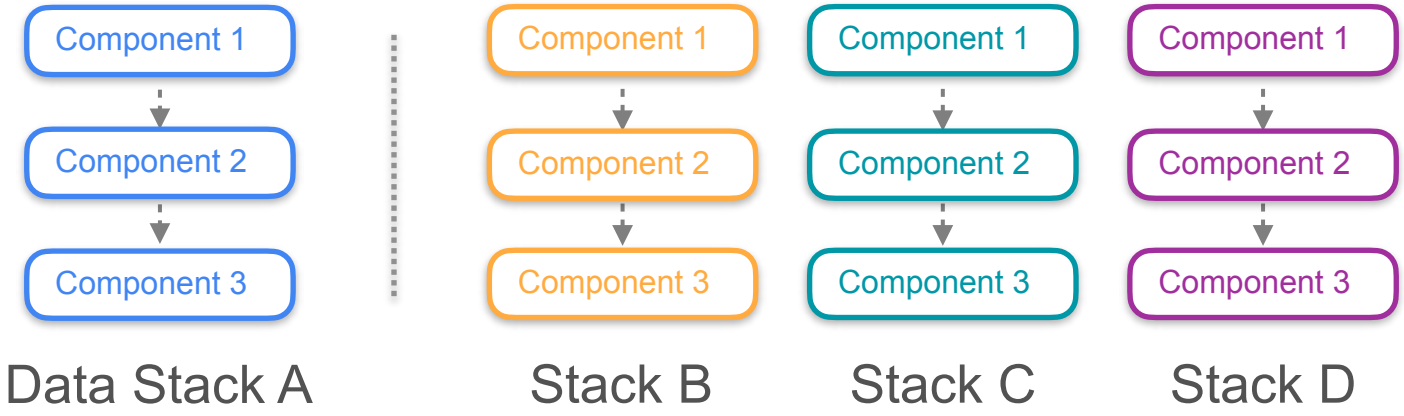
Operational Expenses (OpEx)

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

Cost Optimization

Total Opportunity Cost of Ownership (TOCO)

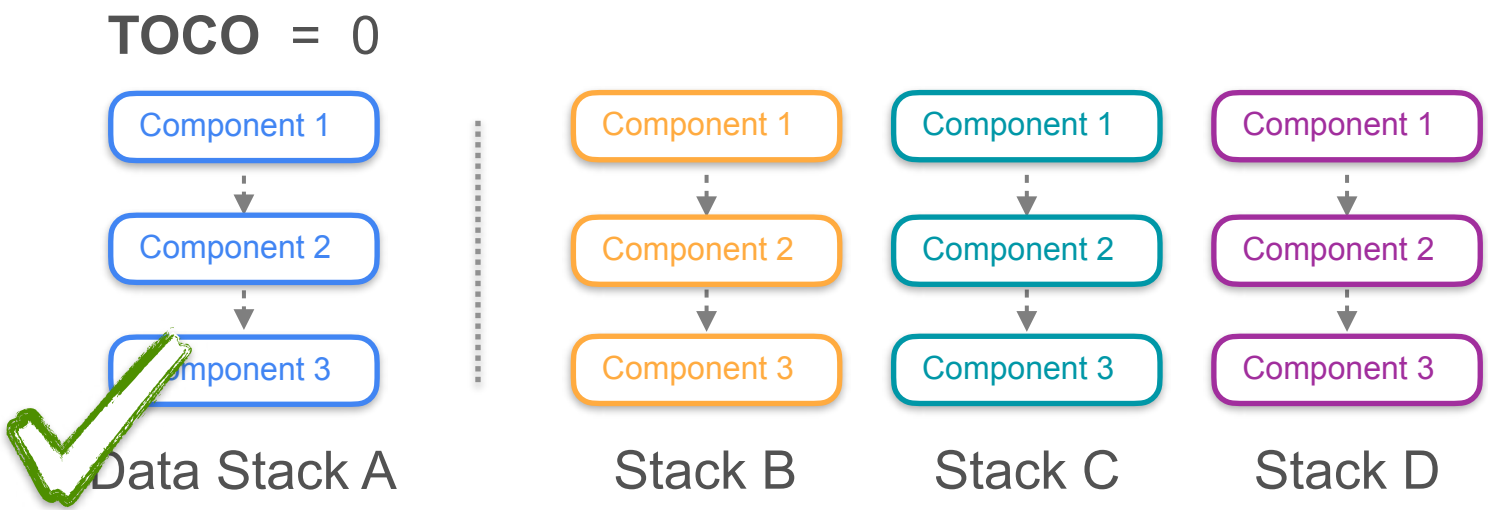
The cost of lost opportunities that you incur in choosing a particular tool or technology.



Cost Optimization

Total Opportunity Cost of Ownership (TOCO)

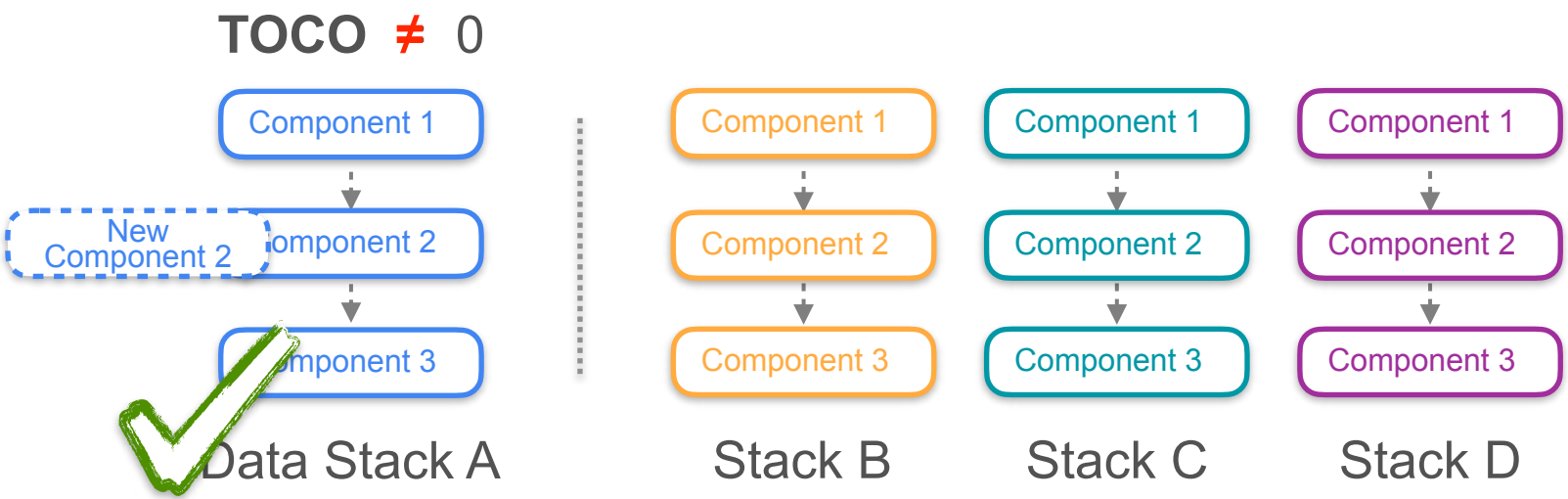
The cost of lost opportunities that you incur in choosing a particular tool or technology.



Cost Optimization

Total Opportunity Cost of Ownership (TOCO)

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



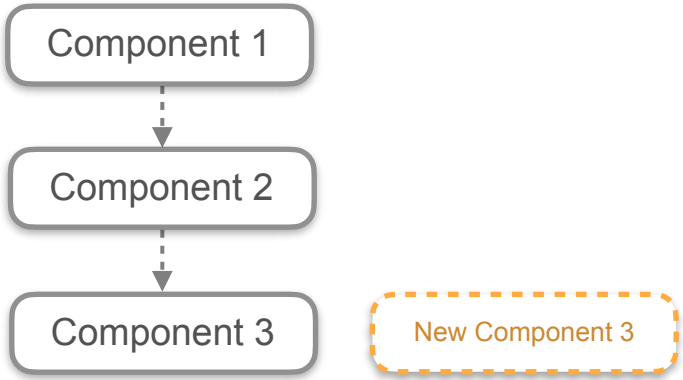
Cost Optimization

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

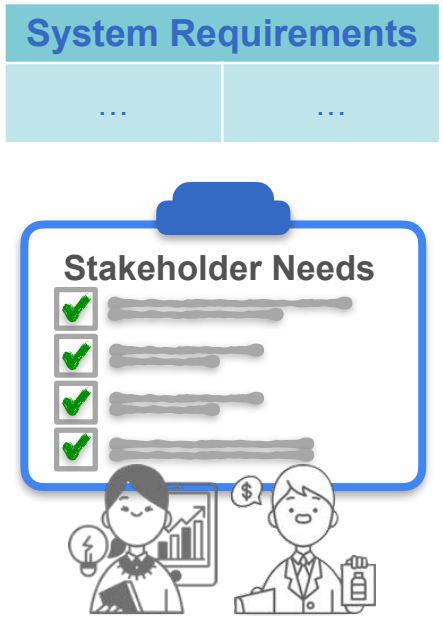
- Flexible, pay-as-you-go technologies
- Modular options

DeepLearning.AI

Choosing the Right Technologies

Build vs Buy

Build Your Own Solution



Customize your own solution

Benefits:

- Get exactly the solution you need
- Avoid licensing fees
- Avoid being at the mercy of a vendor

Use Existing Solution

- Open-Source *(community)*
- Commercial Open-Source *(vendor)*
- Proprietary Non-Open-Source

Considerations

Your team

- Does your team have the bandwidth and capabilities to implement an open source solution?
- Are you a small team? Could using a managed or proprietary service free up your time?

Cost

- How much are licensing fees associated with managed or proprietary services?
- What is the total cost to build and maintain a system?

Business Value

- Do you get some advantage by building your own system compared to a managed service?
- Are you avoiding undifferentiated heavy lifting?

DeepLearning.AI

Choosing the Right Technologies

Server, Container, and Serverless Compute Options

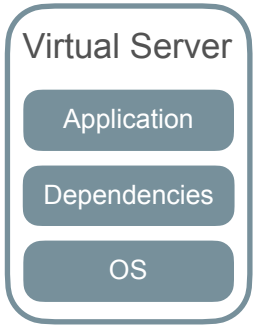
Server

Container

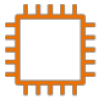
Serverless

You set up and manage the server

- Update the OS
- Install / update packages
- Patch software
- Networking, scaling, and security



Example:

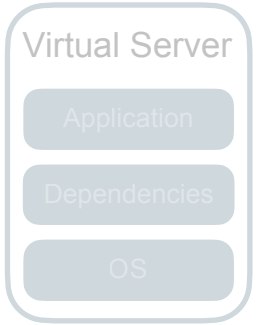


EC2 instance

Server

You set up and manage the server

- Update the OS
- Install / update packages
- Patch software
- Networking, scaling, and security



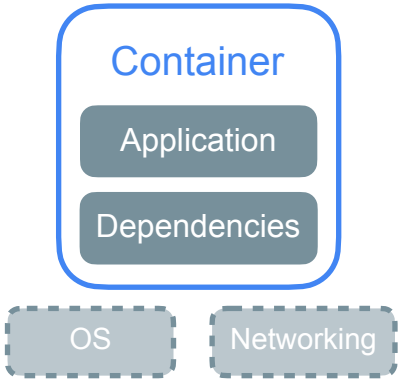
Example:

EC2 instance

Container

Modular unit that packages code and dependencies to run on a server

- Lightweight & portable
- You set up the application code and dependencies

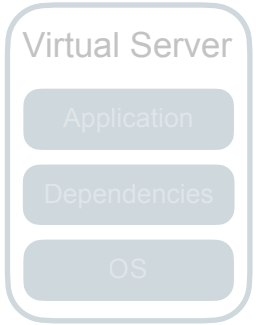


Serverless

Server

You set up and manage the server

- Update the OS
- Install / update packages
- Patch software
- Networking, scaling, and security



Example:

EC2 instance

Container

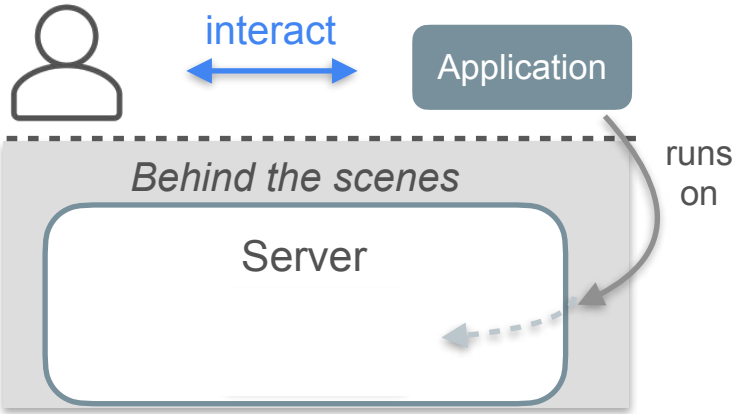
Modular unit that packages code and dependencies to run on a server

- Lightweight & portable
- You set up the application code and dependencies

Serverless

You don't need to set up or maintain the server

- Automatic scaling
- Availability & fault-tolerance
- Pay-as-you-go



Amazon Serverless Services

Serverless services you've worked with

Amazon Athena

AWS Glue

Service that popularized serverless services

AWS Lambda

Run code in response to an event

Advantages of serverless services

- Execute small chunks of code on as-needed basis
- Run services on as-needed basis
- Pay a little bit each time you run code or use a service

When To Use Serverless Services?

Cloud cost

Example:

AWS Lambda

Expensive in a **high event rate** environment

Model & Monitor

- Event rates
- Event duration
- Cost per event

Limitations

- Limits on execution frequency, concurrency, and duration
- Use container if your application can't function within limits

Best for:

Simple & discrete tasks

Not good when:

Many parts require a lot of compute or memory power

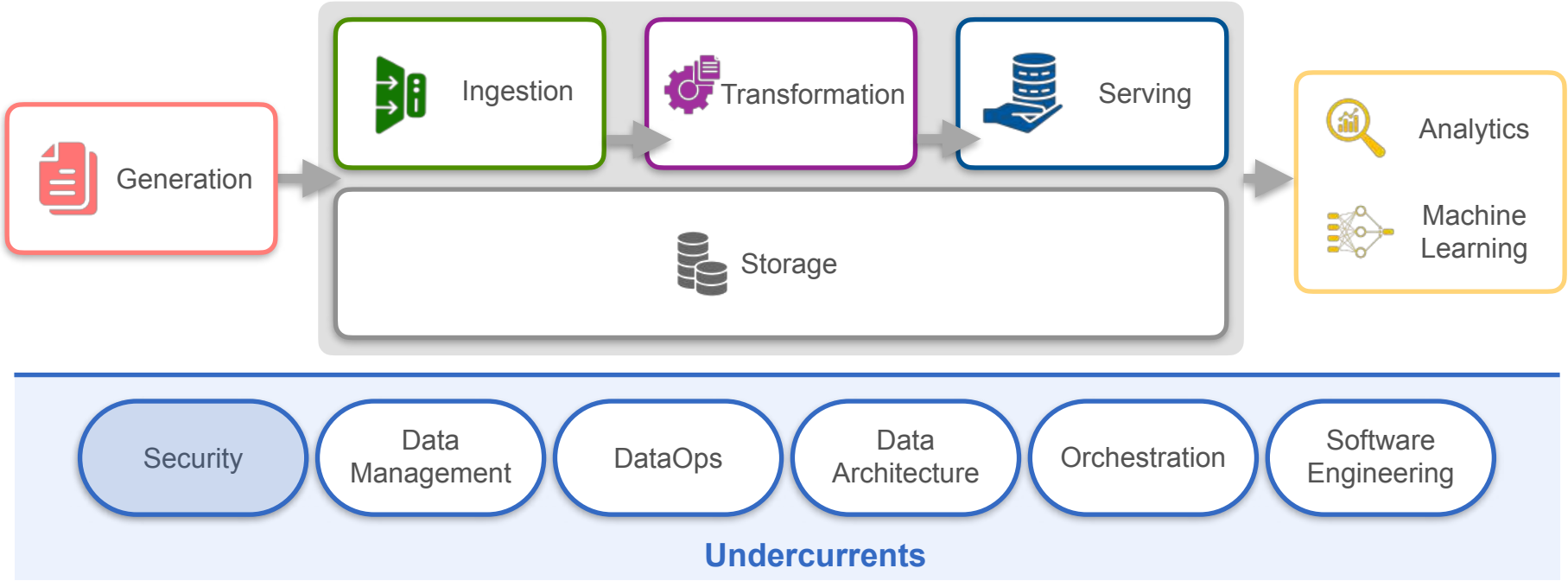
Container orchestration

DeepLearning.AI

Choosing the Right Technologies

How the Undercurrents Impact Your Decisions

The Undercurrents



Security

What are the security features of the tool?

Use tools from reputable sources.

Know where your tools come from!

Data Management

How are data governance practices implemented by the tool provider?

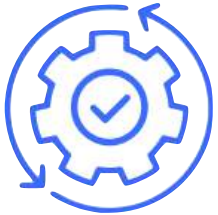
Data Breach

Compliance

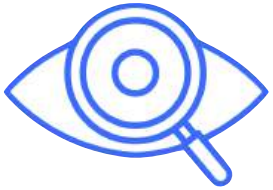
Data Quality

DataOps

What features does the tool offer in terms of automation and monitoring?



Automation

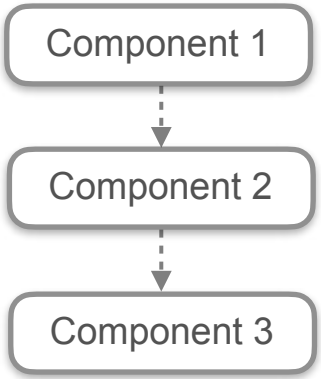


Monitoring

Service Level Agreement (SLA)

Data Architecture

Does the tool provide modularity and interoperability?



Loosely-Coupled Components

Orchestration

Software Engineering

How much do you want to do?

Team's capabilities

Business Value

Build

vs

Buy

Avoid undifferentiated heavy lifting!

DeepLearning.AI

Investigating your architecture (on AWS)

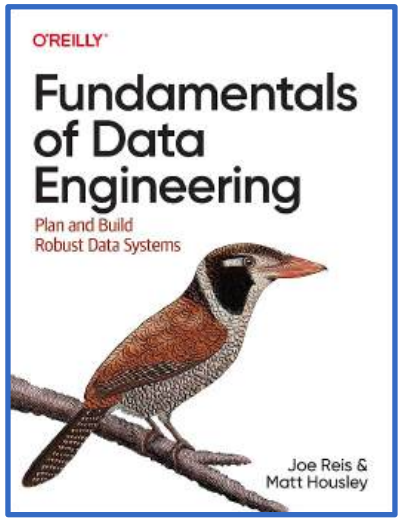
Intro to the AWS Well-Architected Framework

AWS Well-Architected Framework

Set of principles and best practices that help you build:

scalable & **robust** architectures on AWS.

AWS Well-Architected Framework



Principles of good data architecture

Inspired by

AWS Well-Architected

DeepLearning.AI

Investigating your architecture (on AWS)

The AWS Well-Architected Framework

AWS Well-Architected Framework

- Operational Excellence
- Performance Efficiency
- Security
- Cost Optimization
- Reliability
- Sustainability

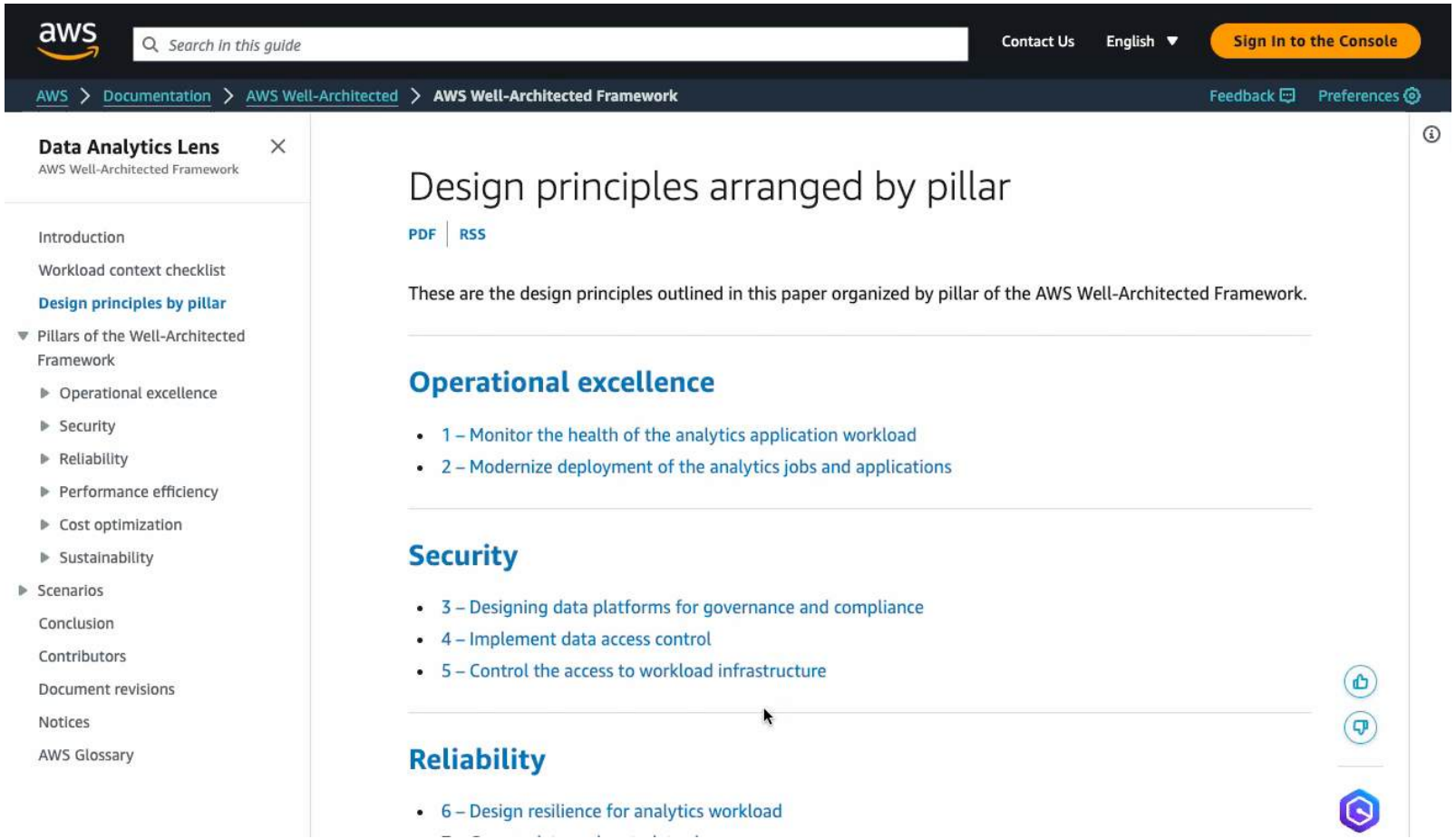
AWS Well-Architected

Operational Excellence	• How you can develop and run your workloads on AWS more effectively • Monitor your systems to gain insight into your operations • Continuously improve your processes and procedures to deliver business value
Security	• How to take advantage of cloud technologies to protect your data, systems, and assets
Reliability	• Everything from designing for reliability to planning for failure and adapting to change
Performance Efficiency	• Taking a data-driven approach to building high-performance architecture • Evaluating the ability of computing resources to efficiently meet system requirements • How you can maintain that efficiency as demand changes and technologies evolve
Cost Optimization	• Building systems to deliver maximum business value at the lowest possible price point • Use AWS Cost Explorer and Cost Optimization Hub to make comparisons and get recommendations about how to optimize cost for your systems
Sustainability	• Consider the environmental impact of the workloads you're running on the cloud • Reducing energy consumption and increasing efficiency across all components of your system

AWS Well-Architected

- Operational Excellence
- Security
- Reliability
- Performance Efficiency
- Cost Optimization
- Sustainability

- Set of principles and questions
- Helps you design and operate reliable, secure, efficient, cost-effective, and sustainable systems in the cloud
- Helps you think through the pros and cons of different architecture choices

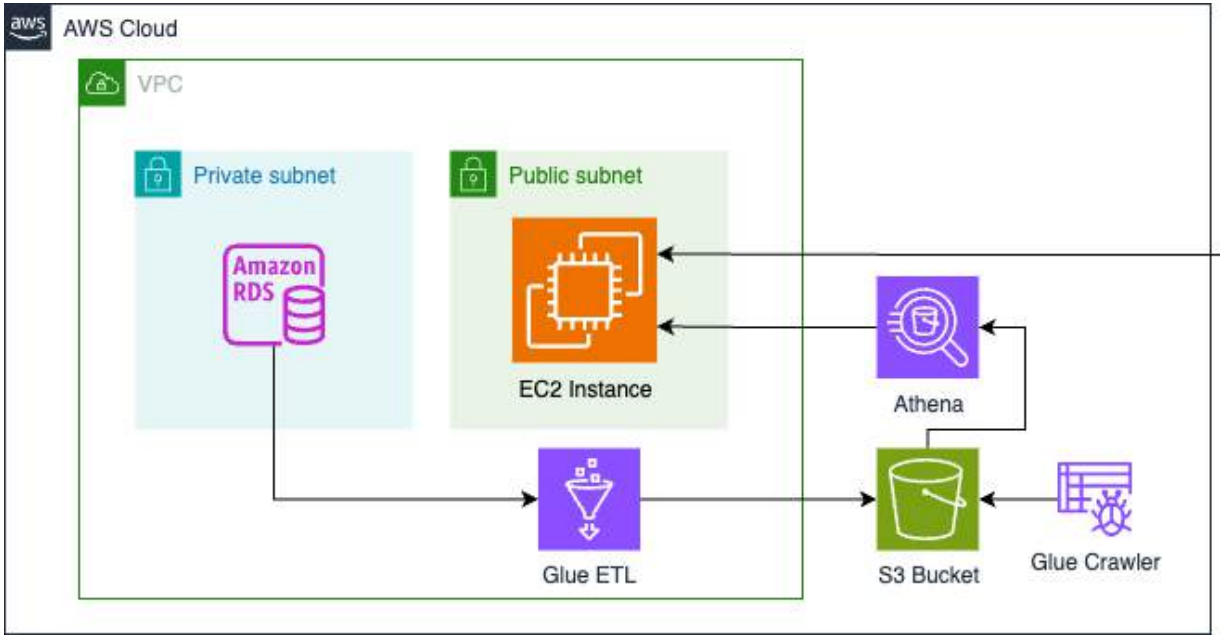

aws
Search in this guide
Contact Us
English
Sign In to the Console
AWS > Documentation > AWS Well-Architected > AWS Well-Architected Framework
Feedback
Preferences
Data Analytics Lens
AWS Well-Architected Framework
Introduction
Workload context checklist
Design principles by pillar
Pillars of the Well-Architected Framework
Operational excellence
Security
Reliability
Performance efficiency
Cost optimization
Sustainability
Scenarios
Conclusion
Contributors
Document revisions
Notices
AWS Glossary
Design principles arranged by pillar
PDF
RSS
These are the design principles outlined in this paper organized by pillar of the AWS Well-Architected Framework.
Operational excellence
1 – Monitor the health of the analytics application workload
2 – Modernize deployment of the analytics jobs and applications
Security
3 – Designing data platforms for governance and compliance
4 – Implement data access control
5 – Control the access to workload infrastructure
Reliability
6 – Design resilience for analytics workload

DeepLearning.AI

Lab Walkthrough

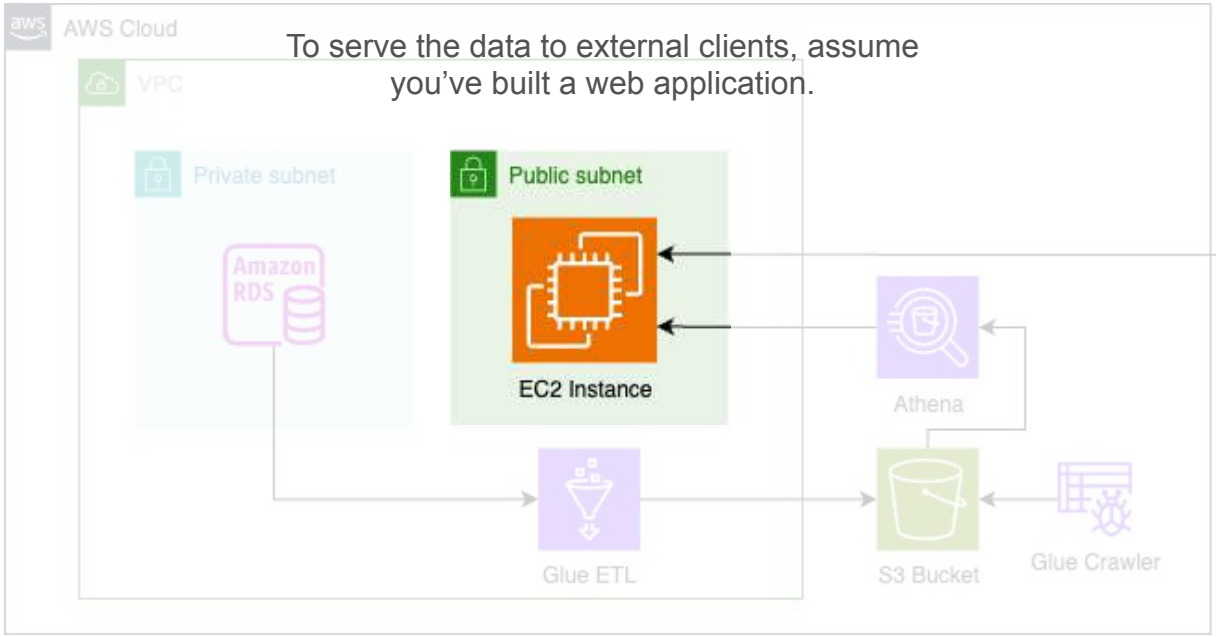
Introduction to the Lab

Week 3 Lab



- Serve embedded dashboards to clients
- Share analytics data through third-party data sharing platforms

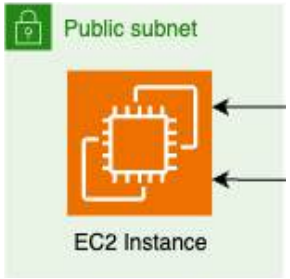
Week 3 Lab



- Serve embedded dashboards to clients
- Share analytics data through third-party data sharing platforms

Week 3 Lab

To serve the data to external clients, assume you've built a web application.



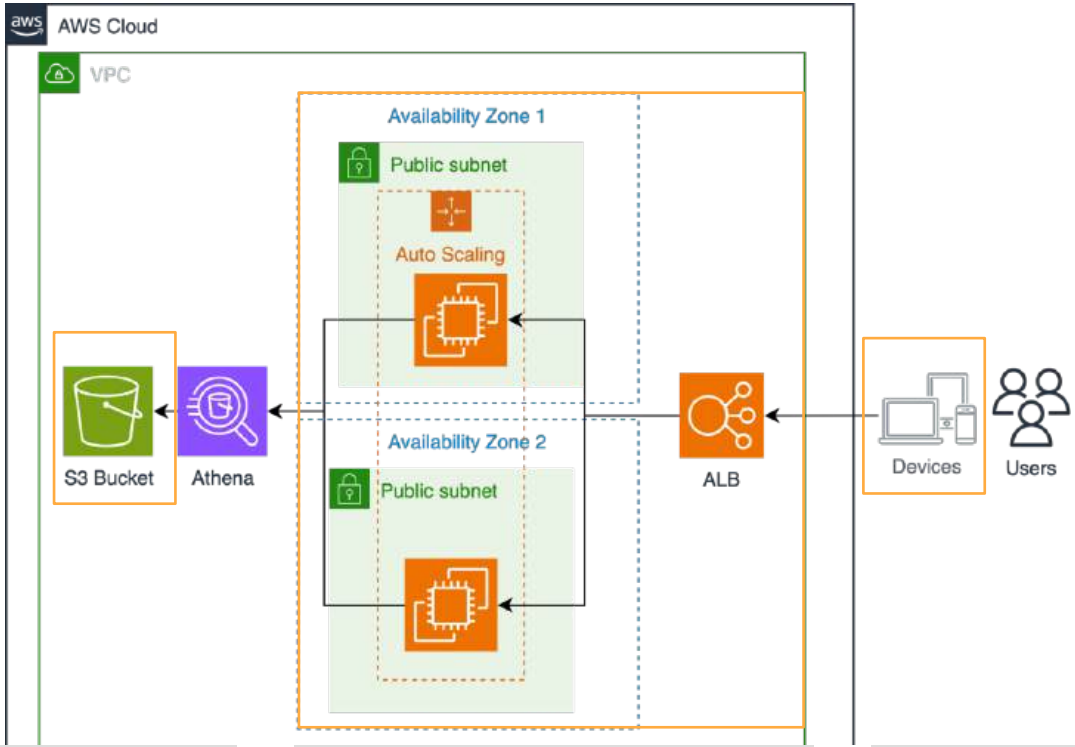
In this lab, ensure that the web app:

- Capable of scaling to meet the clients' needs
- Uses computing resources efficiently
- Designed in a secure and reliable way

Principles of good data architecture.

AWS well-architected framework.

Architectural Diagram



Data Tier

The web-app extracts data from the data tier.

Logic Tier

It hosts the application logic that processes the clients' requests.

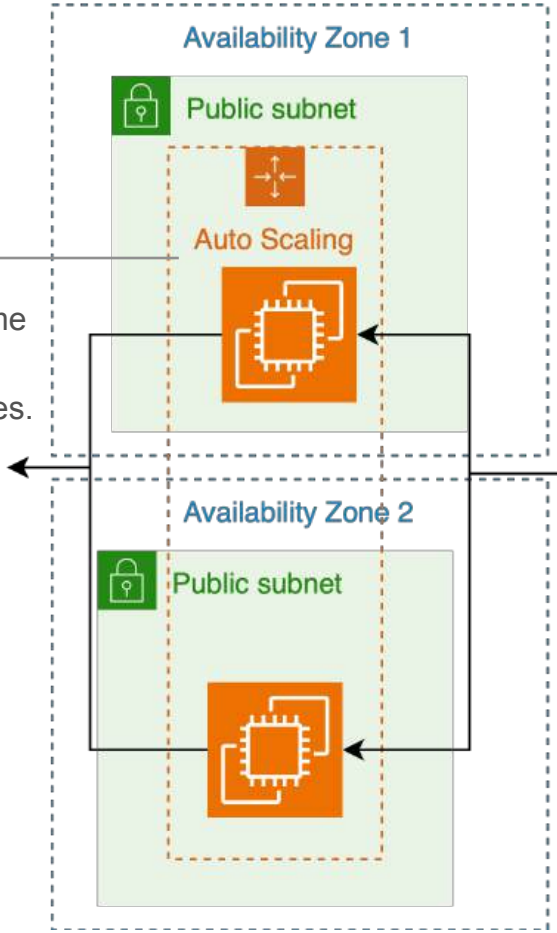
Presentation Tier

An interface that displays results to the clients (ex., webpage)

Logic Tier

EC2 Auto Scaling Group

- Collection of EC2 instances that run the same logic.
- Requests are distributed across the instances.
- The number of instances can automatically increase or decrease.
- Configured to start with 2 instances.

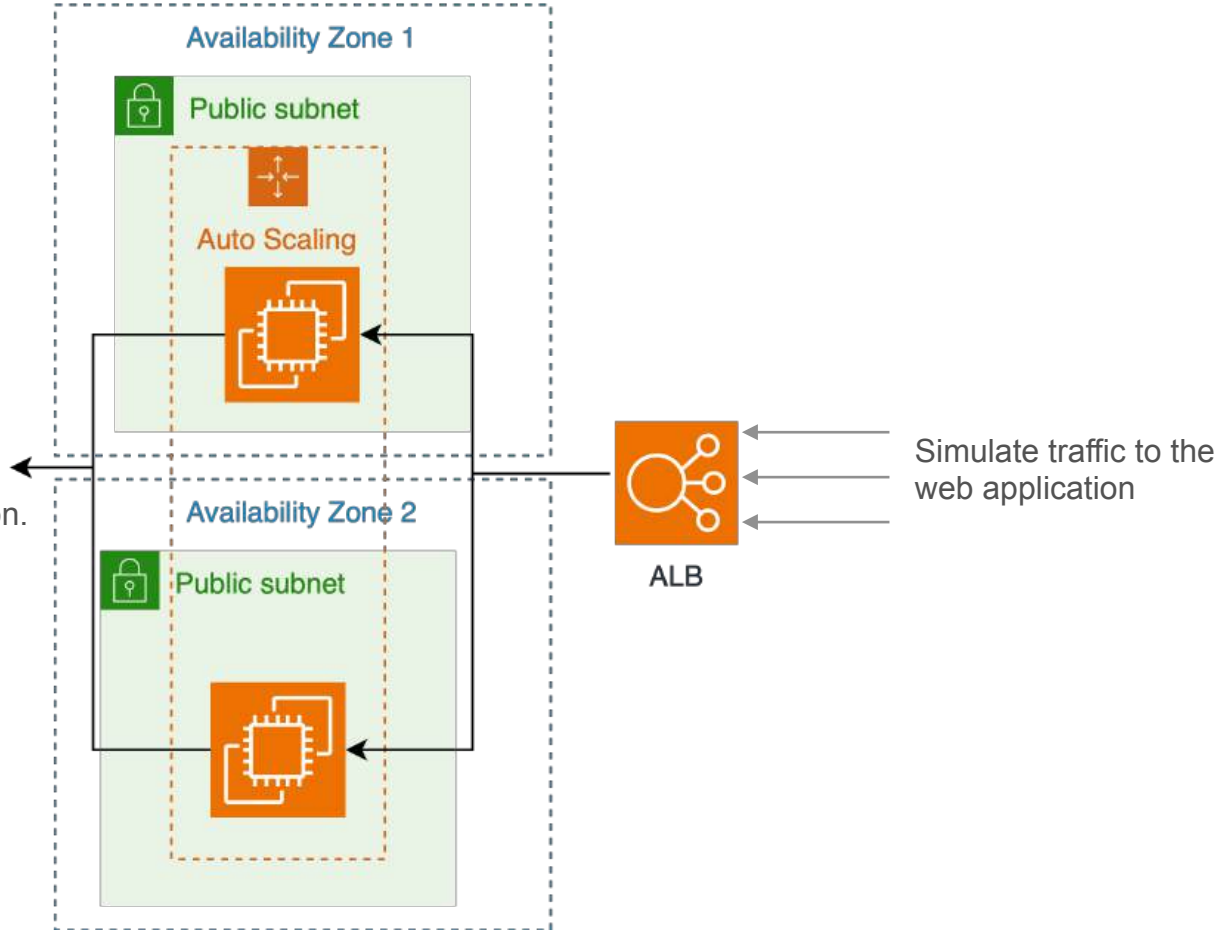


Application Load Balancer

- Distributes the incoming traffic across the instances
- Single point of contact for clients

Week 3 Lab

- Assess the scalability of the app.
- Monitor computing resources and network activity.
- Configure EC2 instances to enable performance efficiency and cost optimization.
- Adjust the security options of the load balancer.



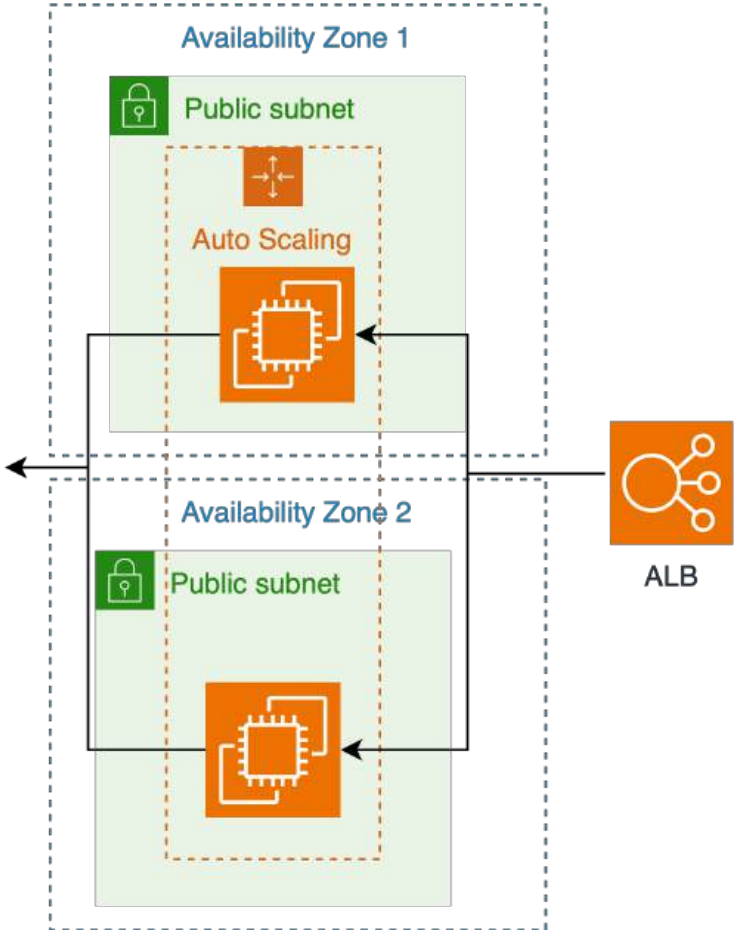
DeepLearning.AI

Lab Walkthrough

Monitoring the Web App

Monitoring the Web App

3. Getting the address of the web application
4. Monitoring CPU usage and networking activity

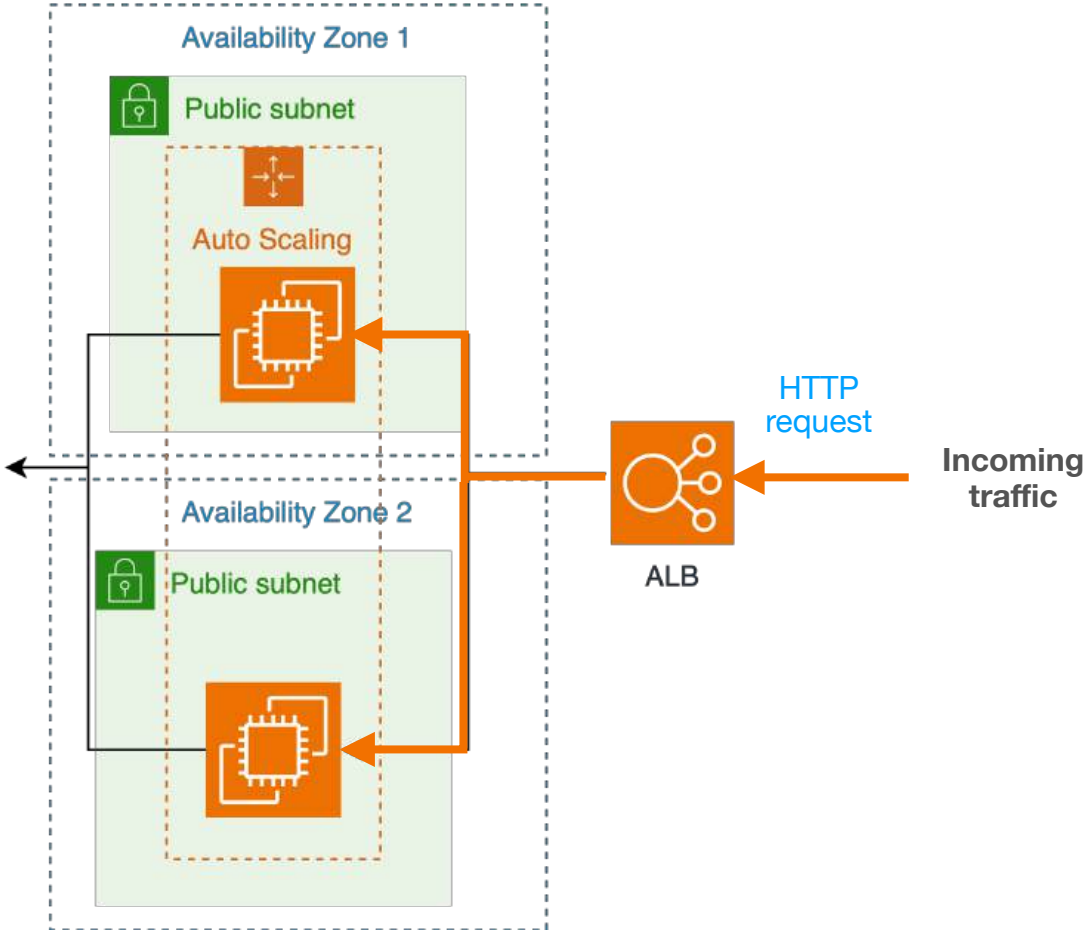


Monitoring the Web App

Monitor the incoming traffic and usage of computing resources.

DataOps Observability

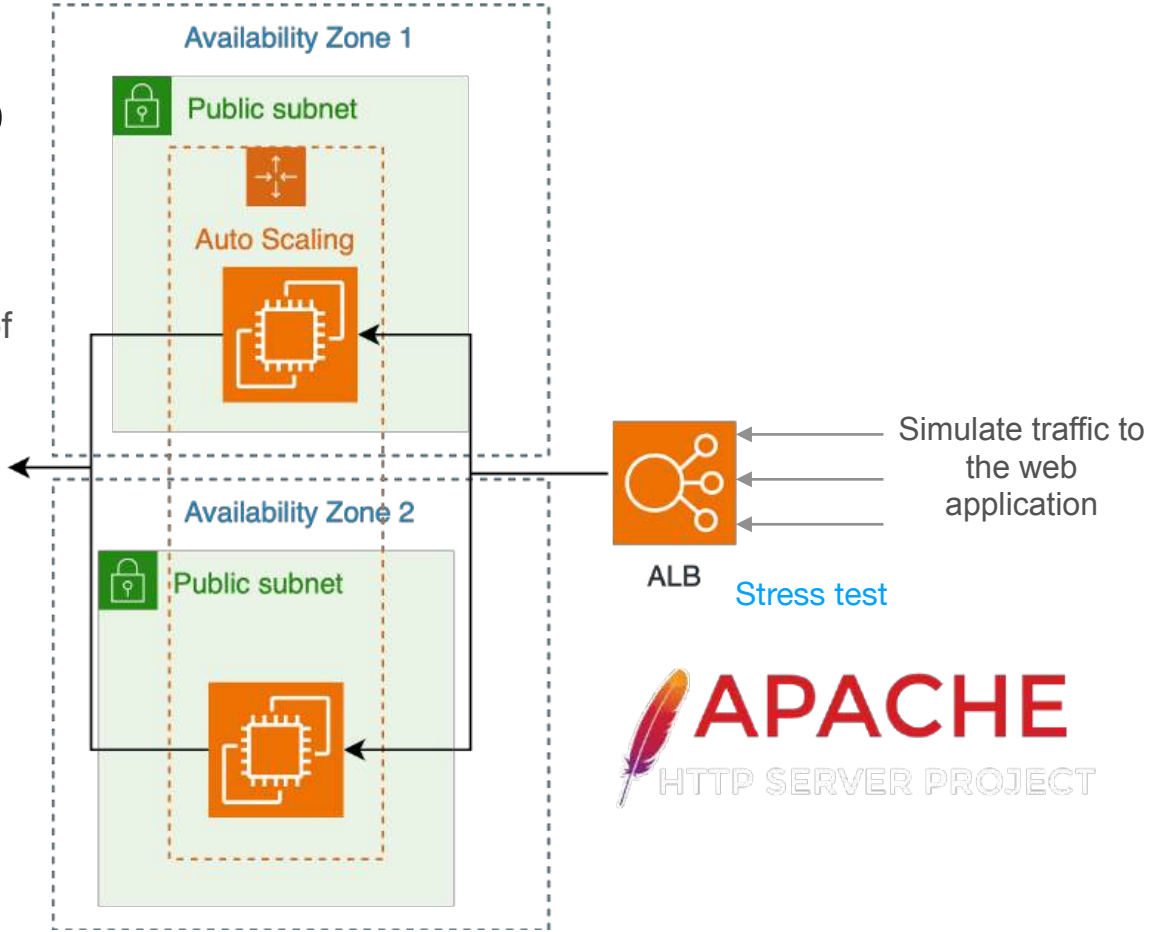
Operational Excellence



Monitoring the Web App

Monitor the incoming traffic and usage of computing resources.

Amazon CloudWatch



DeepLearning.AI

Lab Walkthrough

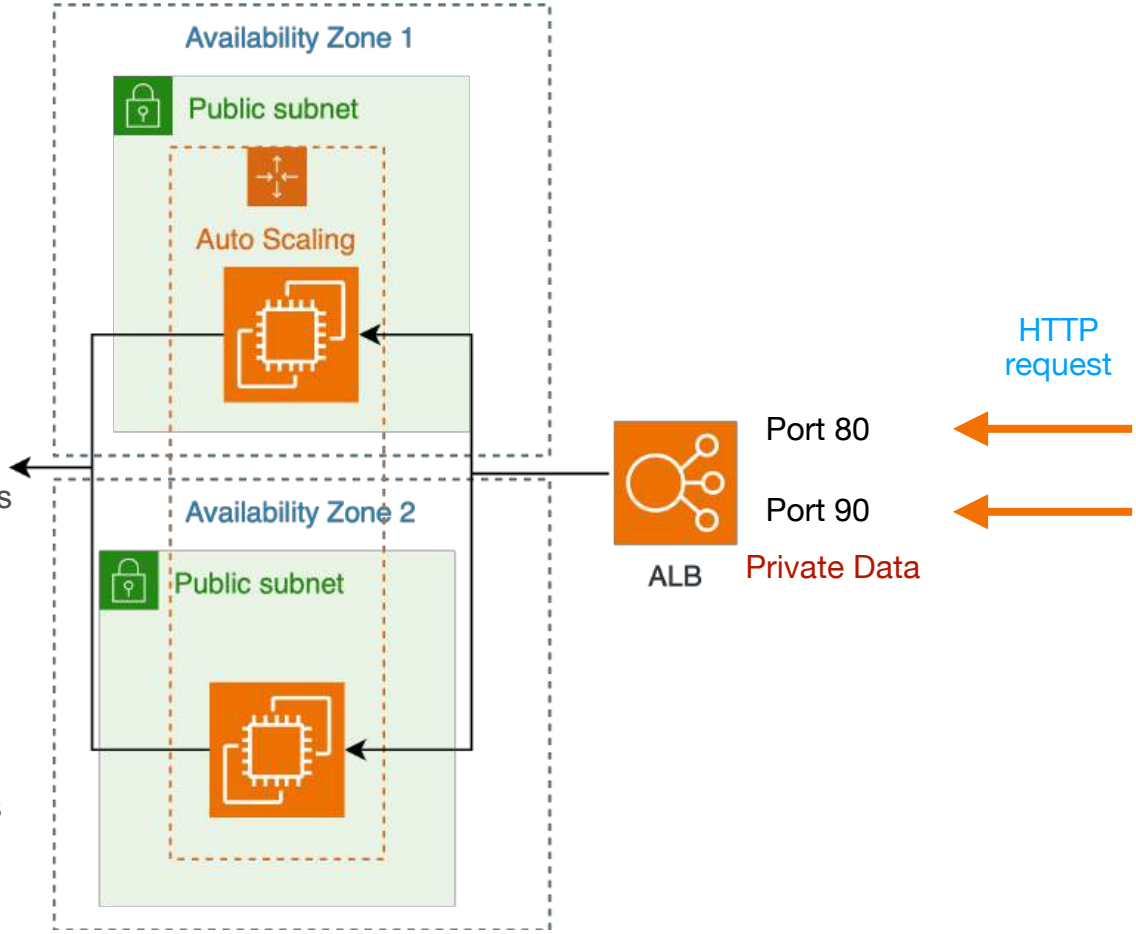
Applying the Principles of Good Data Architecture

Security

Prioritize Security

Security Pillar

- Configure the ALB to only receive certain types of requests.
- A request needs the address and the port number.
- A port number is a virtual identifier that applications use to differentiate between types of traffic

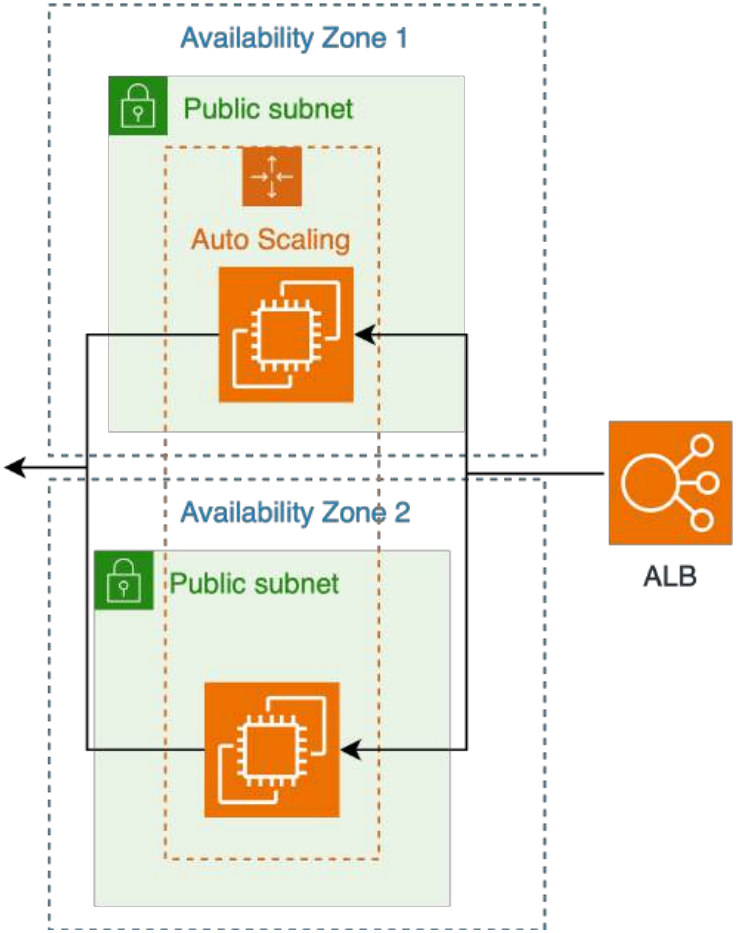


Reliability

Plan for failure

Reliability Pillar

If something goes wrong with one of the availability zones, the requests can still be processed by the EC2 of the other zone.

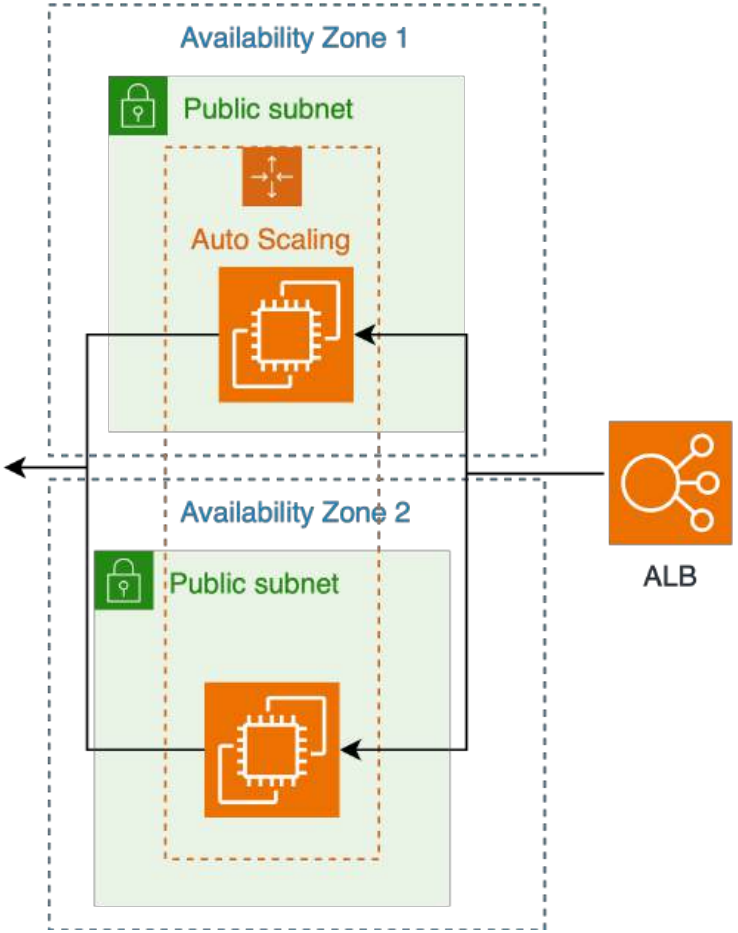


Cost Optimization

Embrace FinOps

Cost Optimization Pillar

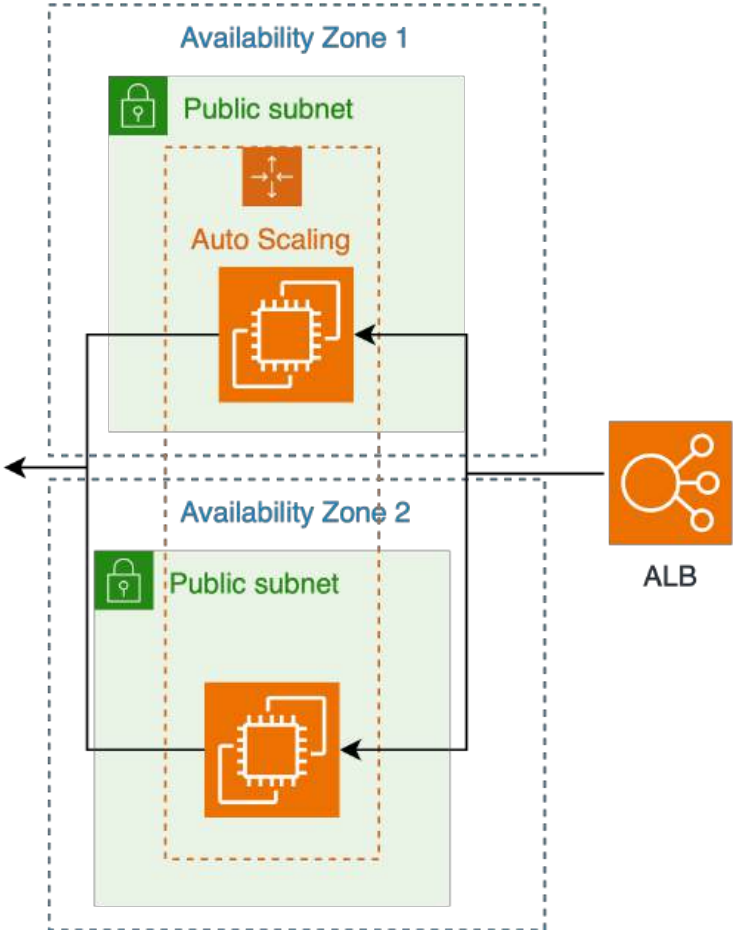
- Use the right resources efficiently.
- Currently, t3.micro instances are used.
- Switch to t3.nano.
- Modify the Launch template of the auto scaling group.



Scalability

Architecting for scalability

- With auto scaling groups, you only pay for what you use.
- Extra EC2 resources will be automatically removed when demand decreases.



DeepLearning.AI

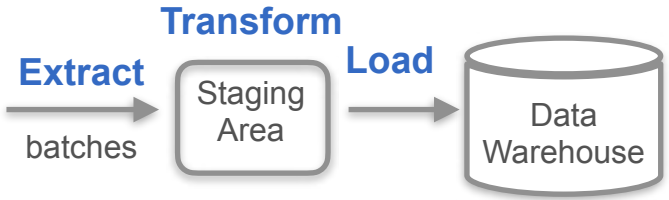
Data Architecture

Week 3 Summary

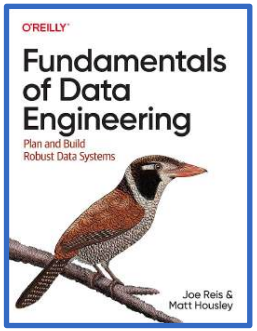
Week 3 Summary

Enterprise Architecture

- Business Architecture
- Application Architecture
- Technical Architecture
- Data Architecture



Principles of good data architecture



Hands-on Lab

aws

AWS Well-Architected