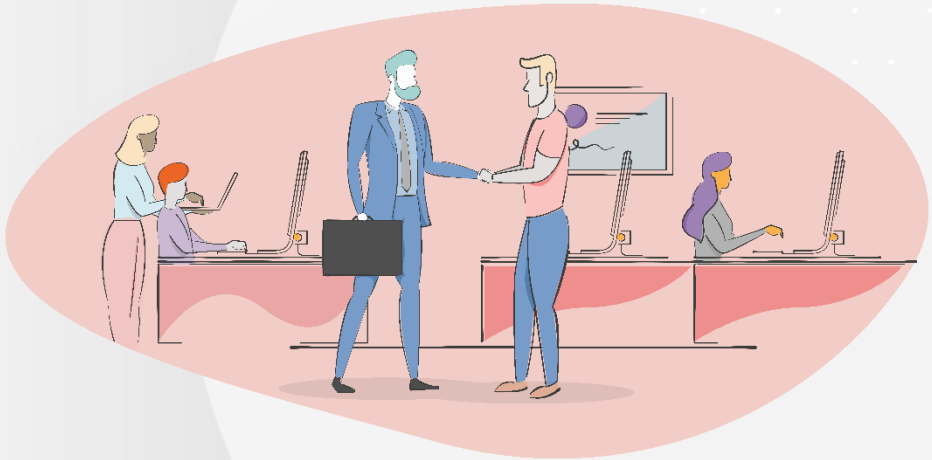
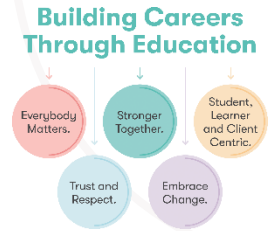
Introduction to reliable data structures



L5 Data Engineer Higher Apprenticeship
Module 1 / 12 (“**Data Fundamentals**”)
Topic 4 / 4

rev. 1 (2024)

Group discussion



Your data system strategy…

John A. Zachman. Early pioneer of enterprise architecture

"Seven thousand years of history would suggest the only known strategy for addressing complexity and change is architecture."

John Zachman

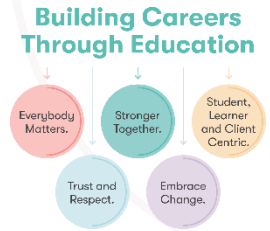
Group discussion

When working with data systems, what is your strategy for addressing:

Substantial increases in complexity and rapid increases in the rate of change?

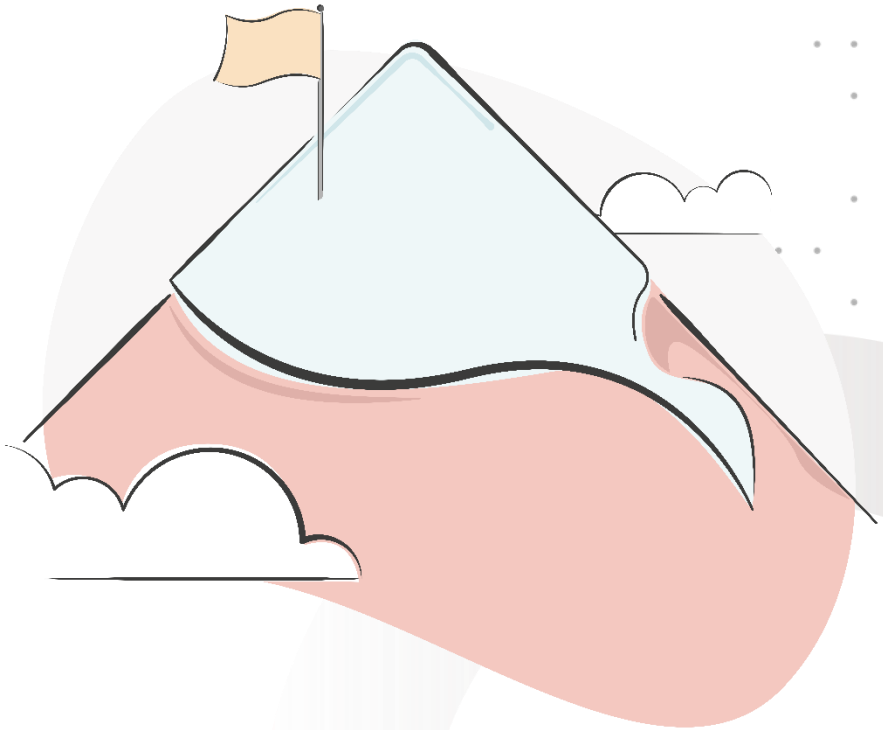
BPP

Learning outcomes



This webinar is designed to support the following learning outcomes:

- Model standardised big data ecosystems and architectures for enterprise data using visual approaches
- List the limitations of the Waterfall model in the context of data engineering projects
- Describe the benefits of the three-layered architecture pattern for organisational data management
- Describe how the microservices architecture style enables organisations to build flexible, scalable, and resilient applications
- Comprehend how to design rudimentary data products and how they add value to the organisation
- Define architectural governance and its role in ensuring reliability and compliance

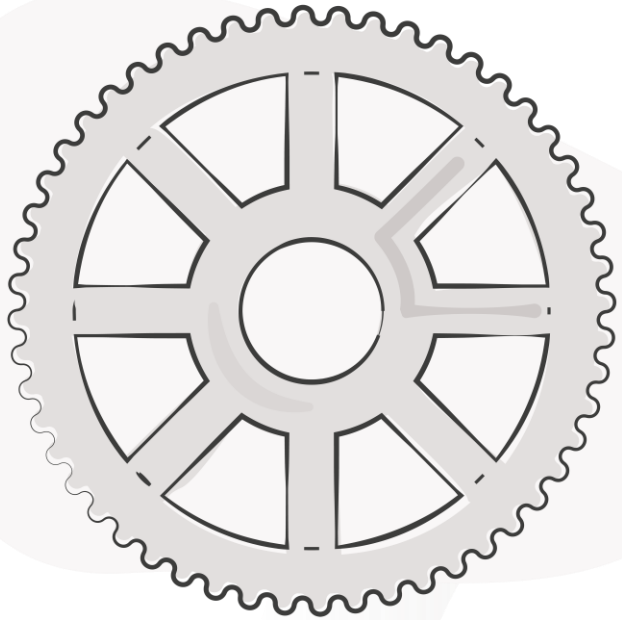


BPP

Webinar agenda

This webinar will cover the following:

- Design vs architecture diagrams and flow charts
- Monolithic vs microservices and UML component diagrams
- Creating architecture diagrams and applying lenses, layers, and chunks
- Publich cloud computing and architecture problem space
- TOGAF, Data Mesh and Data Culture



Webinar length: 3 hours

BPP

Knowledge check poll

Which of the following accurately distinguishes between an architecture diagram and a design diagram?

A. Architecture diagram focuses on system parts, while design diagram shows interaction

B. Architecture diagram covers system details; design diagram focuses on parts

Feedback: B – Let's make this a bit clear on the next slide!

Submit your responses to the chat!

BPP

Design vs architecture diagrams

Its easy to confuse the following two types of diagrams

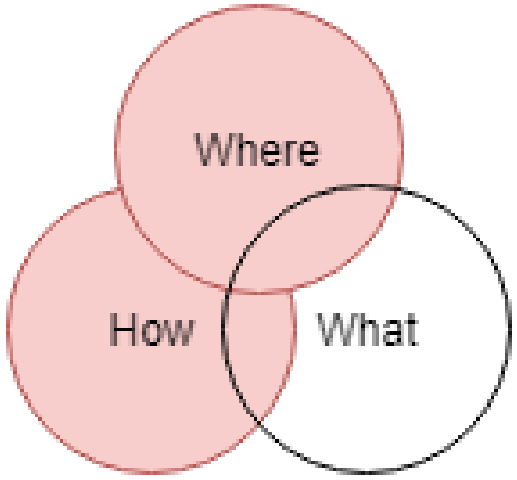
It is easy to confuse the following two types of diagrams:

An architecture diagram

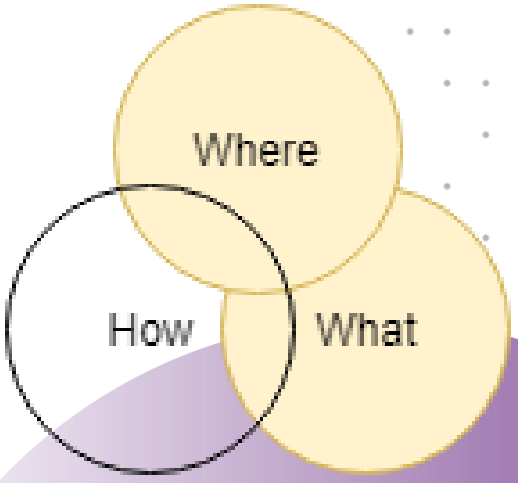
- Explains what you're building
- How stakeholders will interact with it
- The constraints of the system
- Is often drawn as layers

A design diagram

- Normally does not have stakeholders, constraints or layers
- Focuses on one part of the system and shows its building blocks



Design



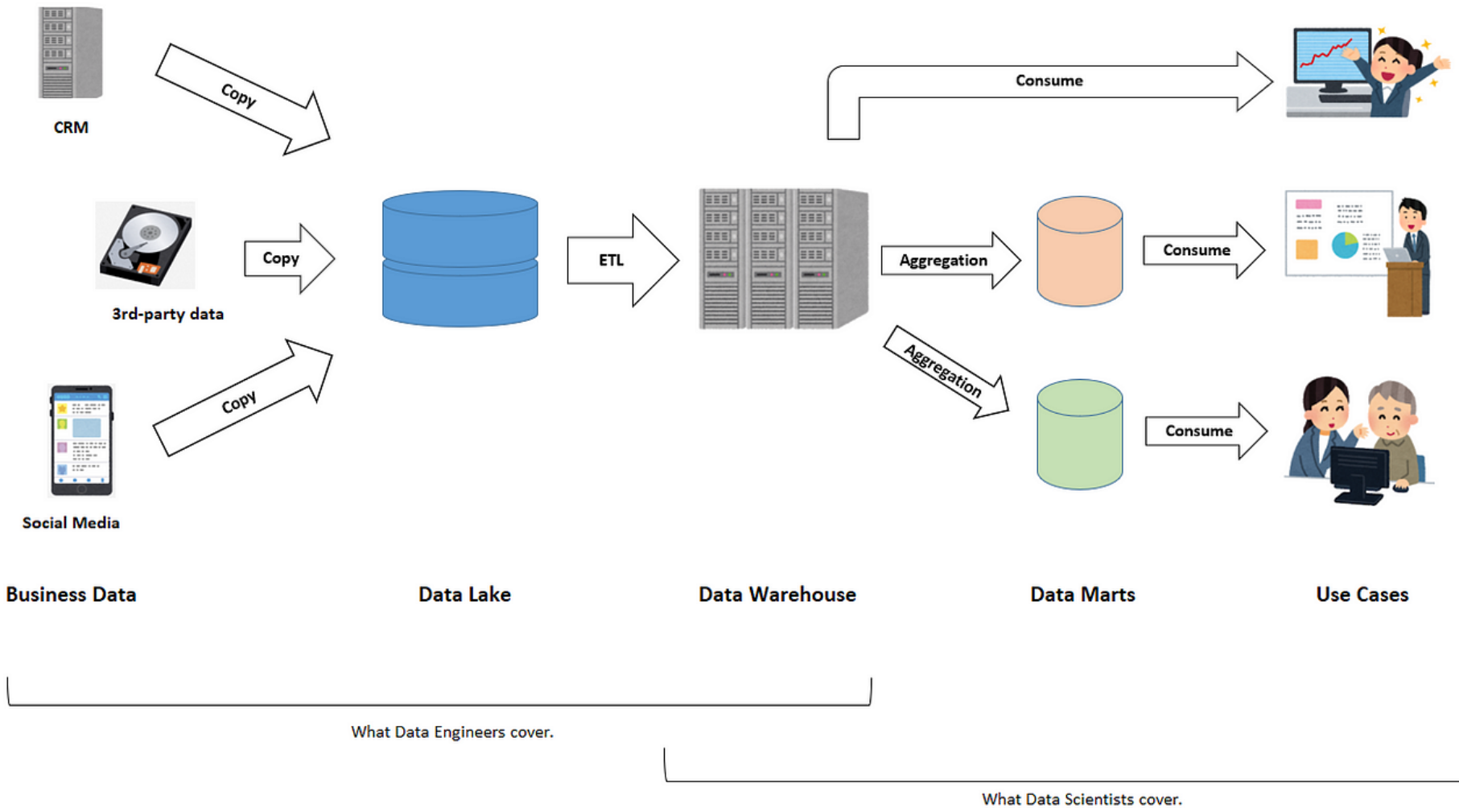
Architecture

The scrum process form start to finish

Image source: *LinkedIn*

A focus on architecture diagrams

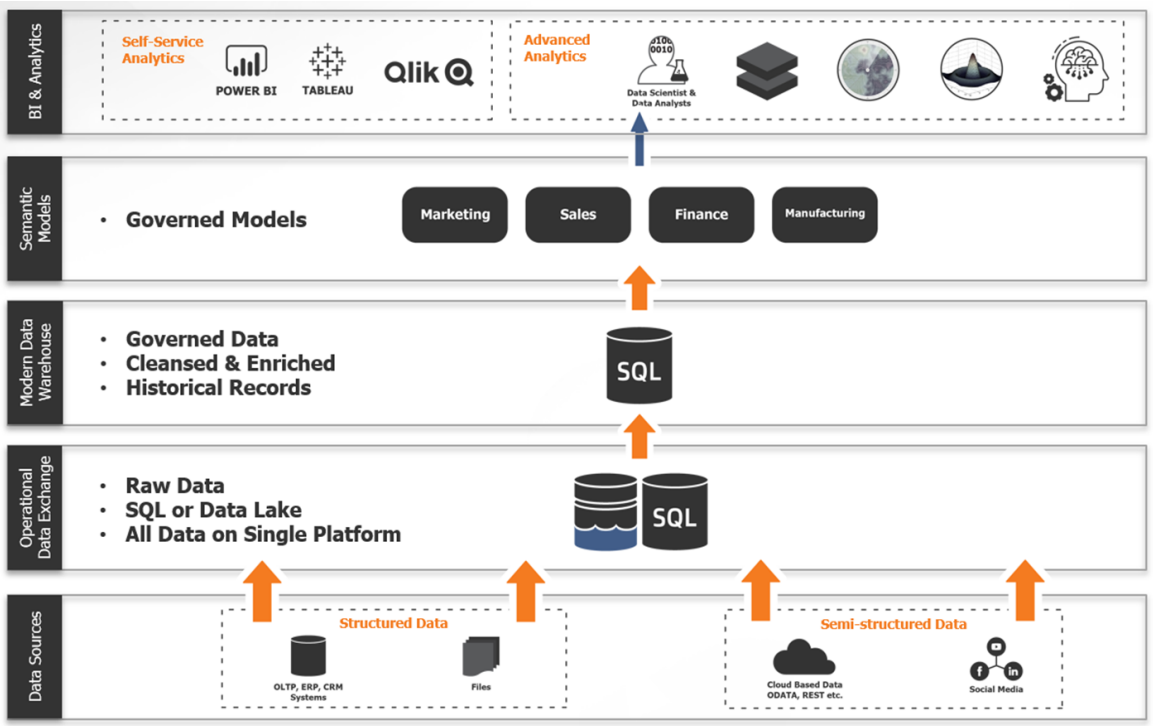
What are they designed to tell use?



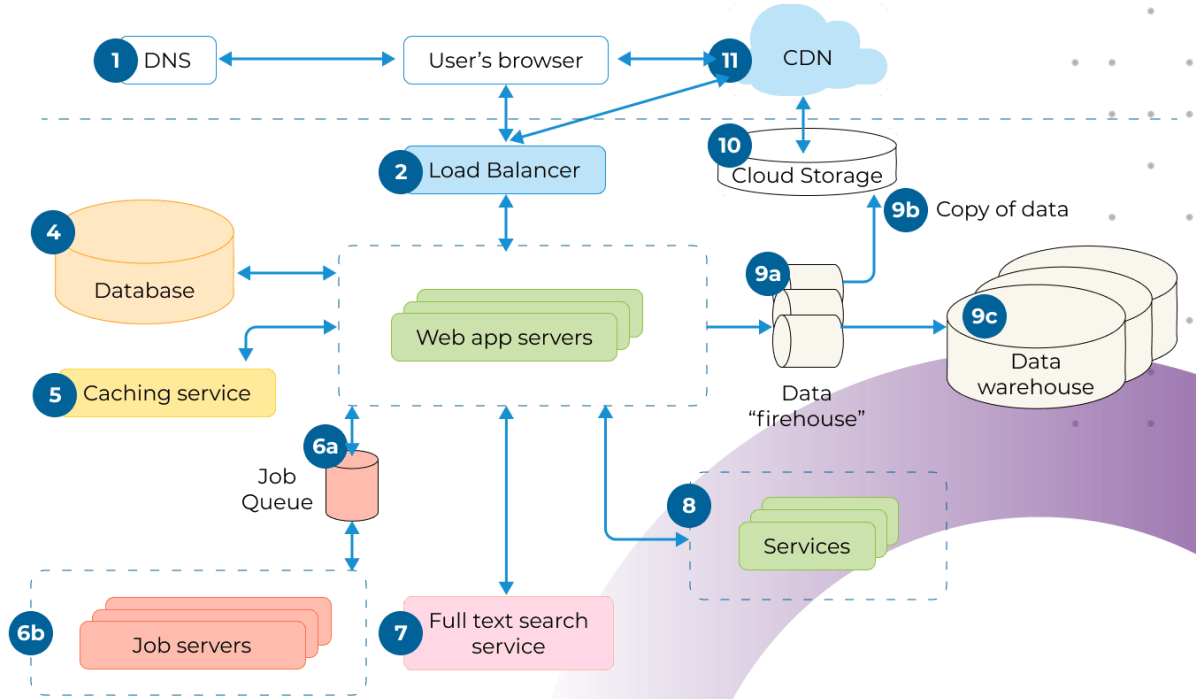
Architecture diagram

Representing architectures with layers

Enhancing system comprehensibility, maintainability, and adaptability

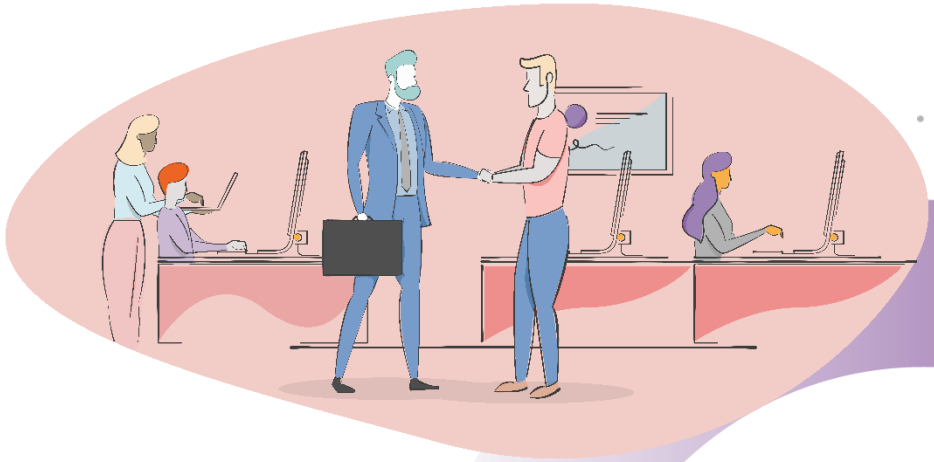


Data architecture layers example 1

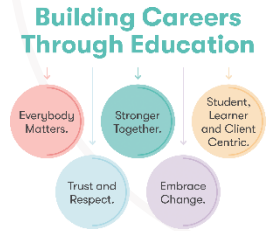


Data architecture layers example 2

Monolithic vs microservices and UML component diagrams



Why use diagrams at all?



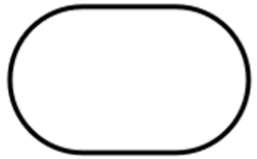
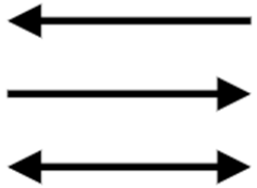
Diagrams are useful for the following reasons:

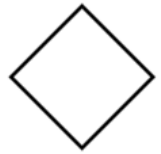
- Diagrams convey key system aspects
- No diagram is flawless; you select focal points
- Diagrams initiate discussions
- System architecture can be viewed through various lenses based on:
    - Job
    - Role
    - Expertise
    - Project stage

Reflective question

Exploring flowcharts

One of the most basic types of diagrams you can make…

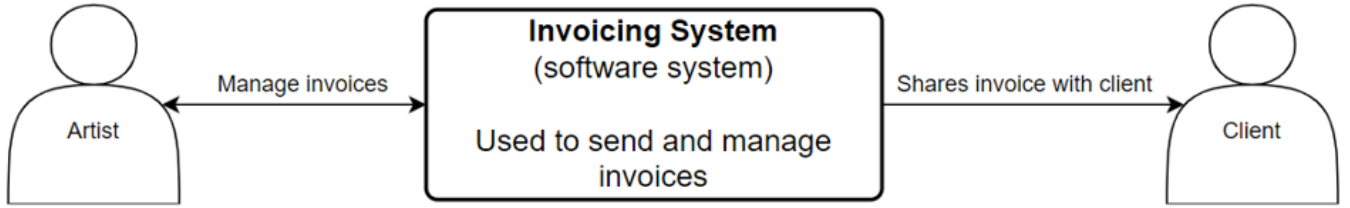
Start / End		Used to represent the starting point or terminal point of a flowchart
Flow lines		Connects components in a flowchart and indicates flow direction
Input / Output		Represents information or data that is transmitted or received

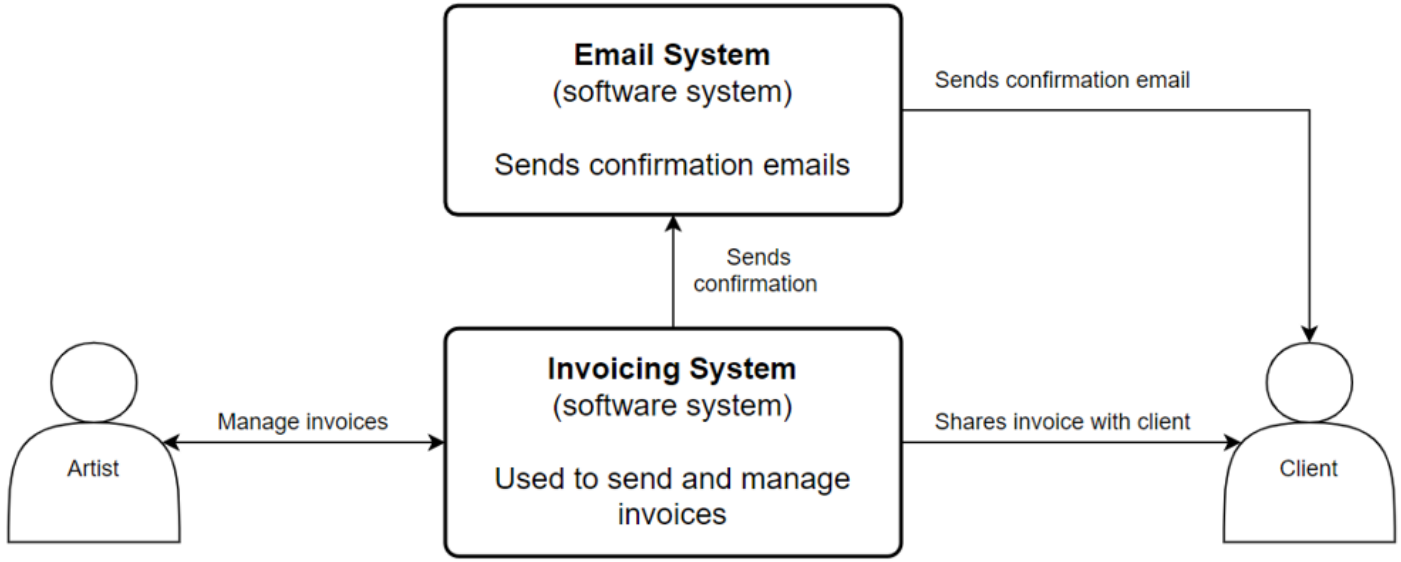
Decision		Represents checkpoints to evaluate conditions for making decisions
Process		Represents processes (e.g., mathematical operations)
Database		Represents databases
Person		Represents actors or users or a software system

Components of flow charts

Document the actors

Identify actors and document external system interactions for software clarity

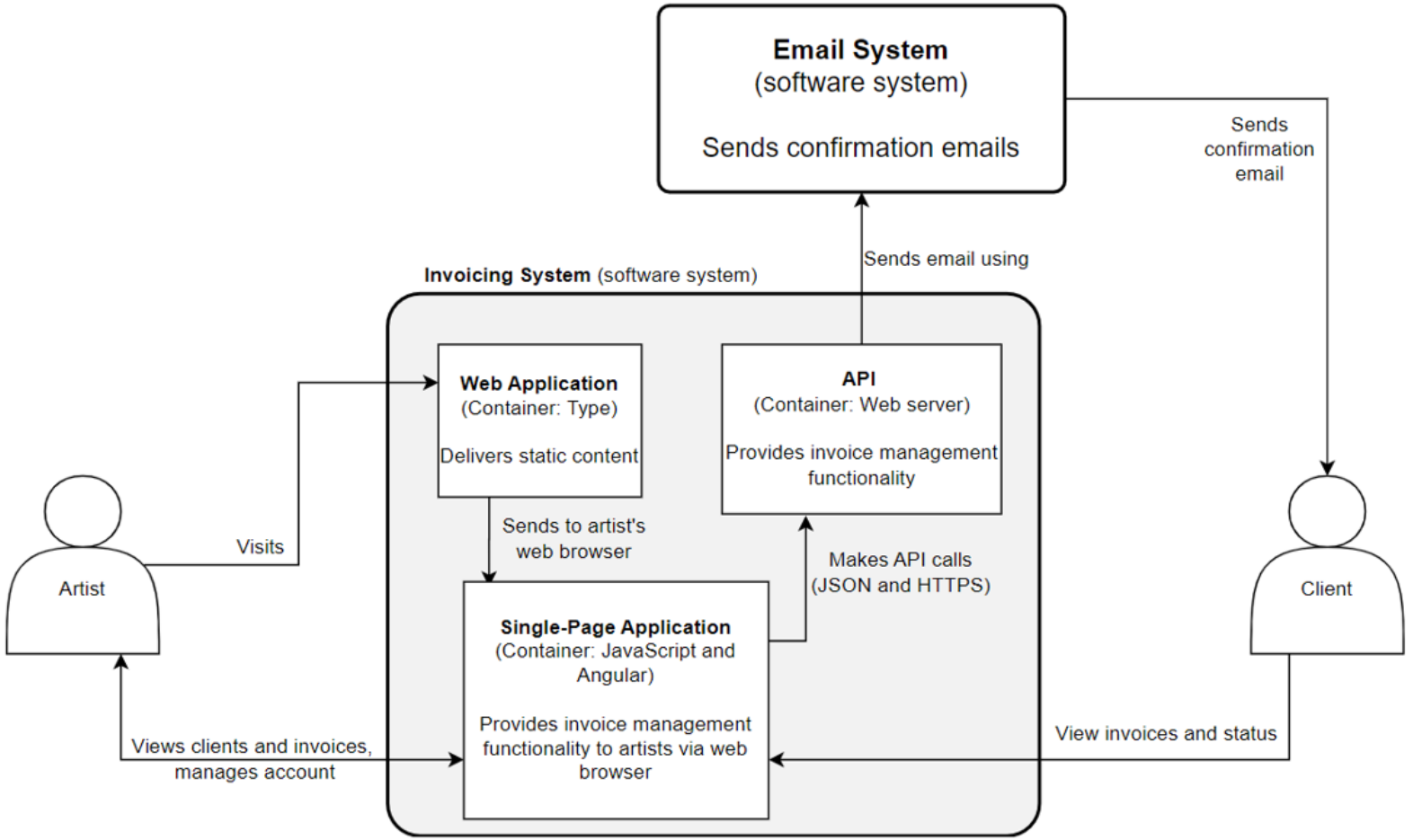




*Now that you know who your actors are, document any **external systems** interacting with the software system*

Adding containers

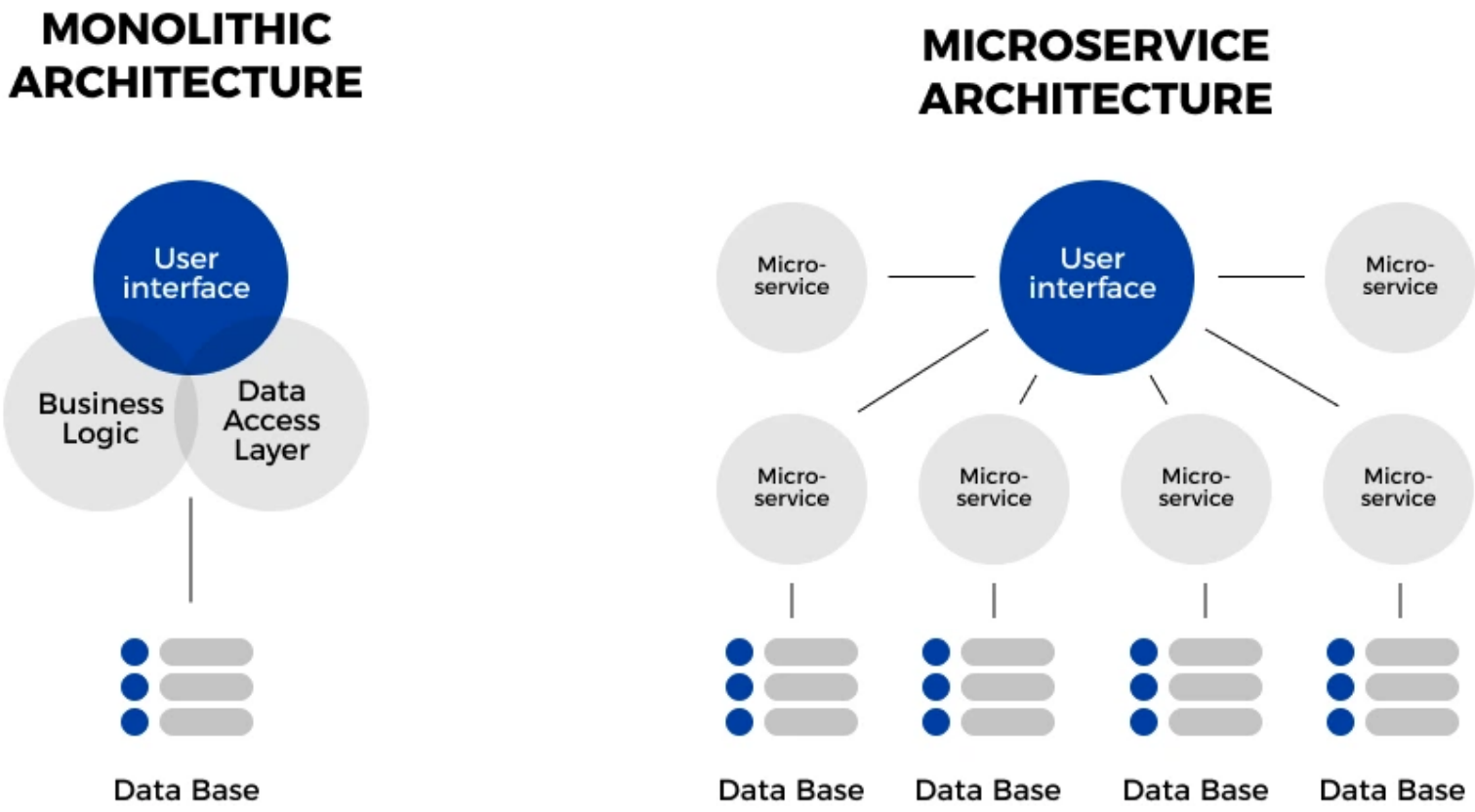
The benefits of adding containers in flowcharts



Improving flowchart clarity with containers for grouping and hierarchy

Monolithic vs microservices

Offering contrasting approaches to system design and development.



Comparing monolithic and microservices, ***Image source:*** *OpenLegacy*

Misuse deployment diagram

Threat orientated diagrams

- "Misuse Deployment Diagram" identifies security threats in data architecture
- Shows how attackers exploit weaknesses
- Includes servers, databases, networks, and controls
- Aids in proactive risk mitigation
- Ensures data protection and system integrity

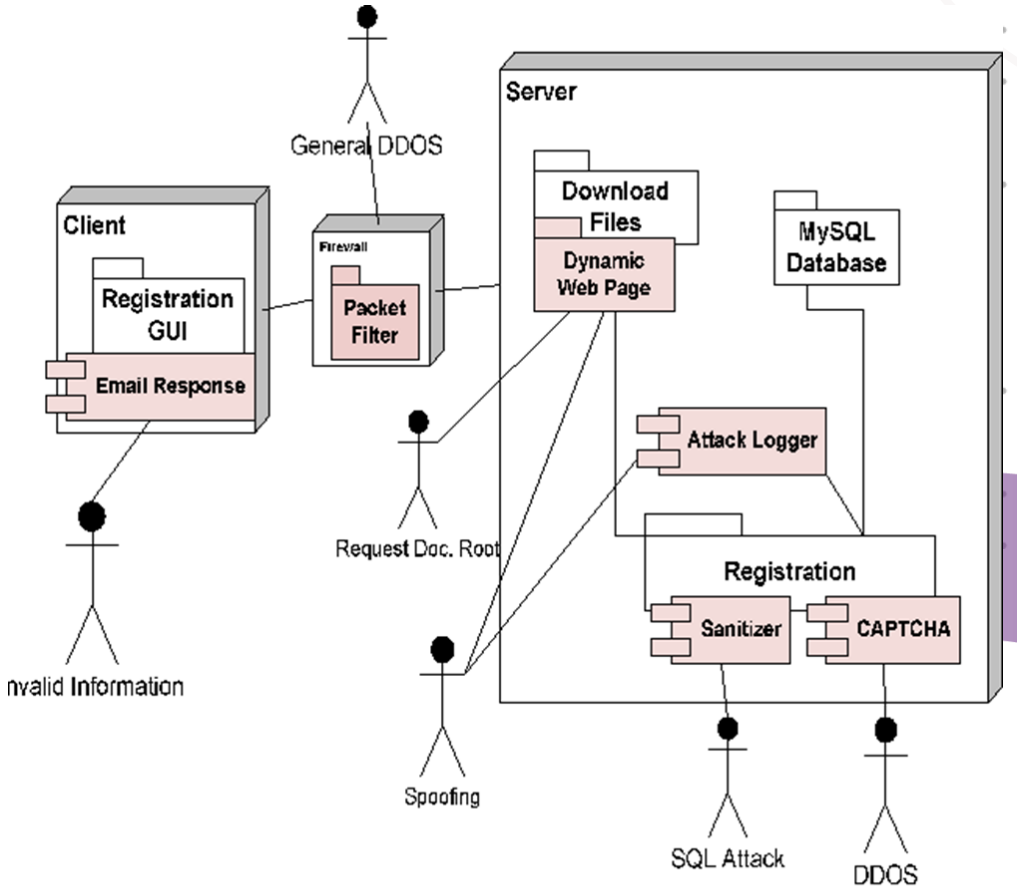
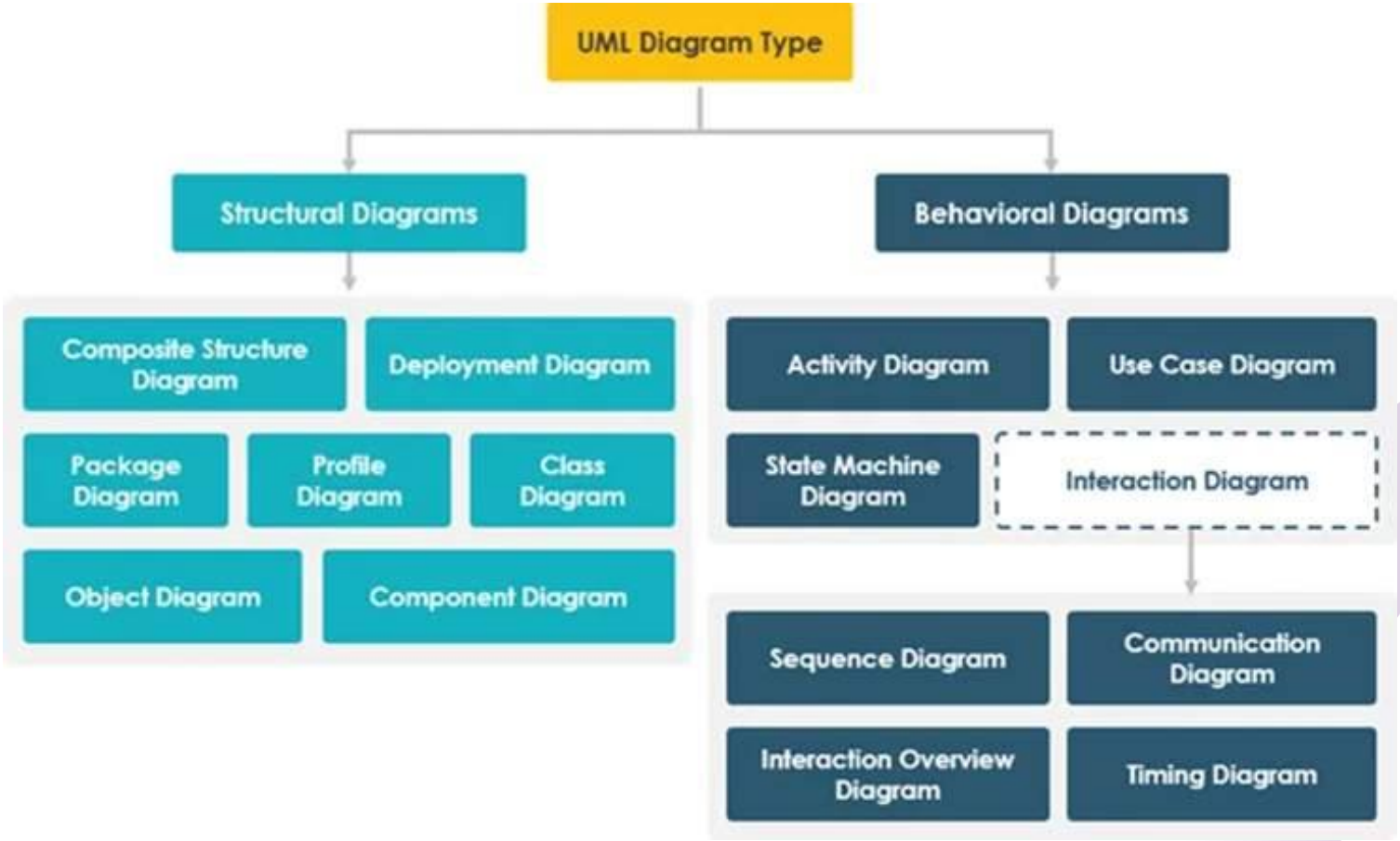


Figure 3. MDD for Web Registration

Proactively safeguarding data integrity with misuse deployment diagrams

UML component diagrams

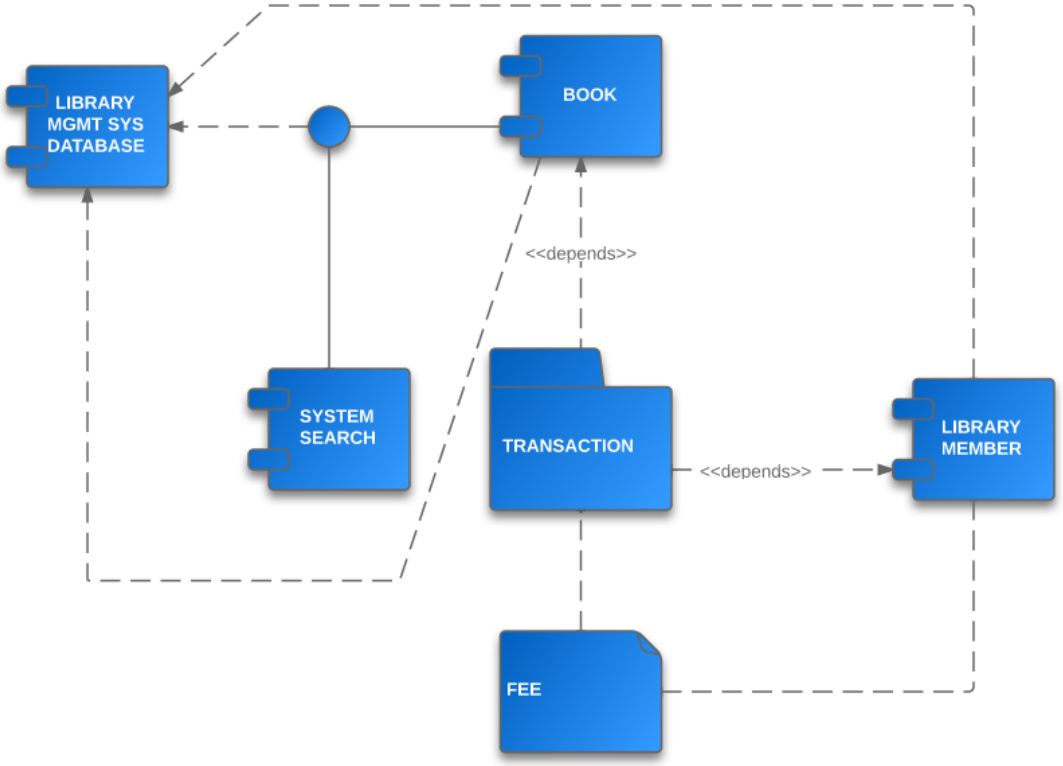
What is their purpose?



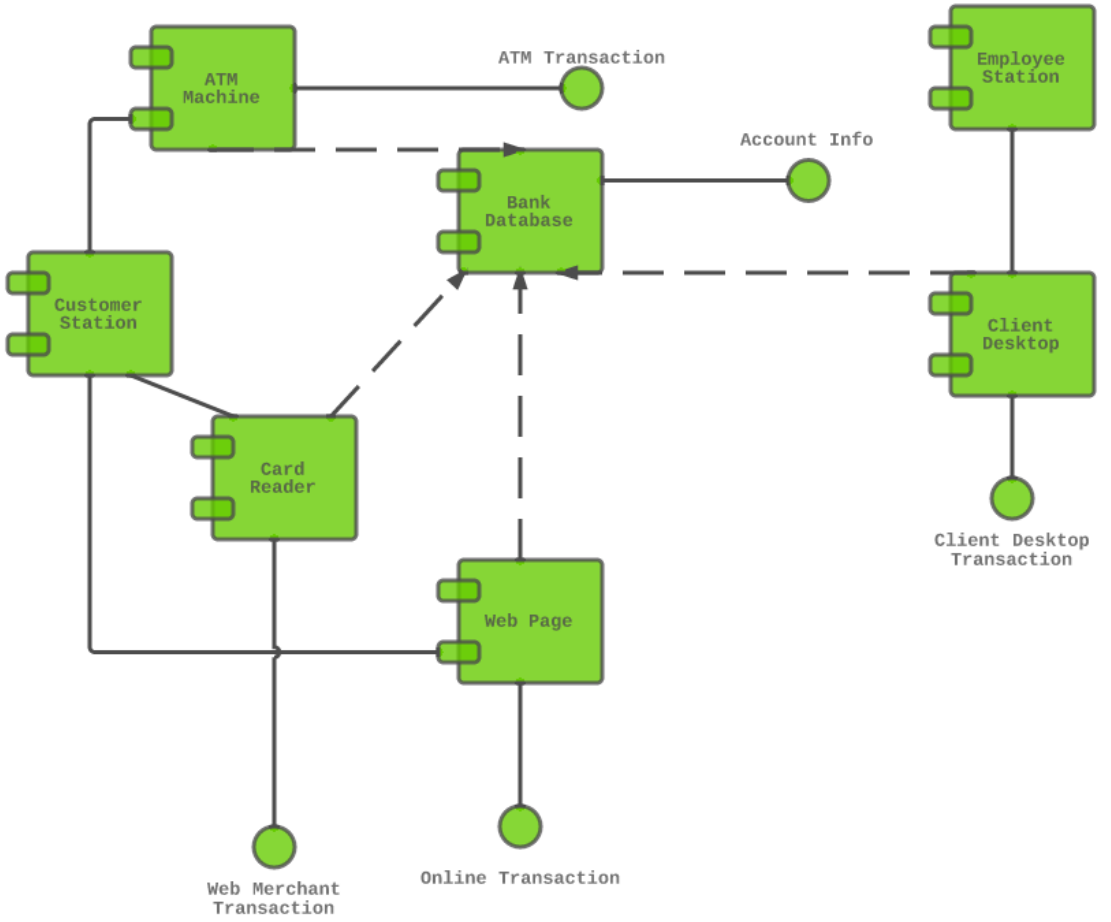
There is a wide variety of UML diagrams, and this chart shows you the different UML diagram types

System component diagrams

How do they work?



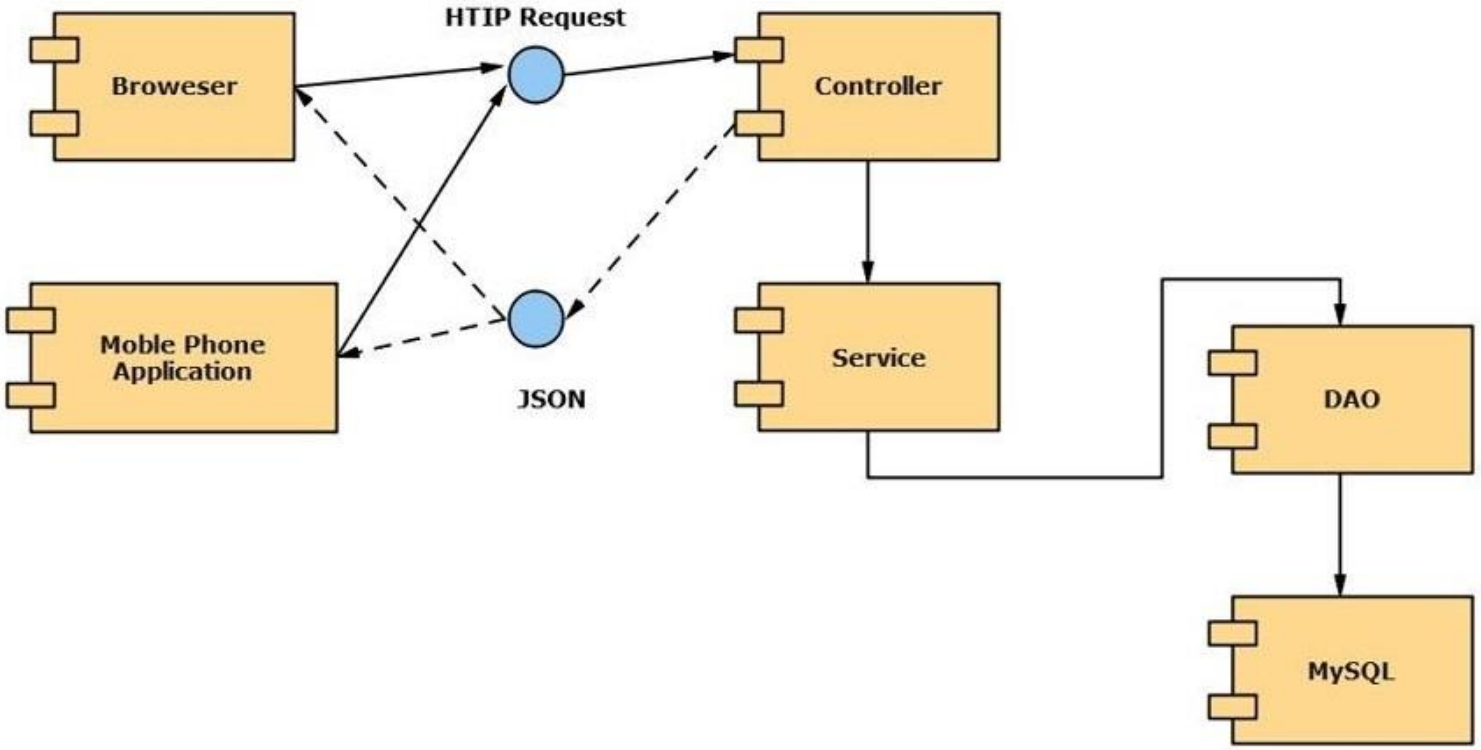
System component diagram example 1



System component diagram example 2

System component diagrams

How do they work?



System component diagram example 3

System component diagram key

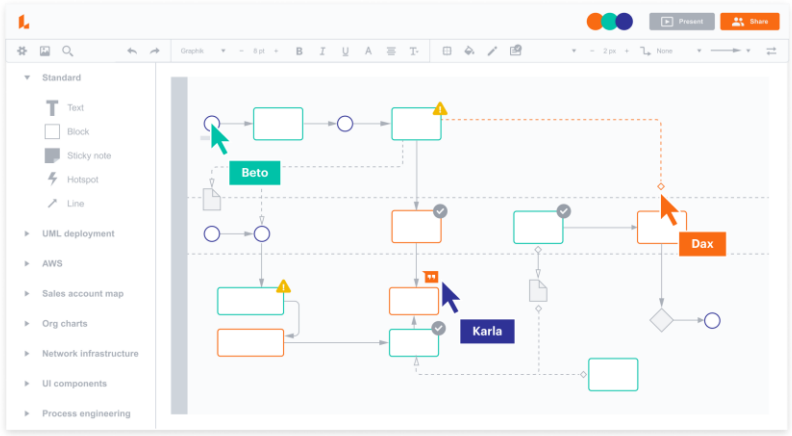
Group discussion

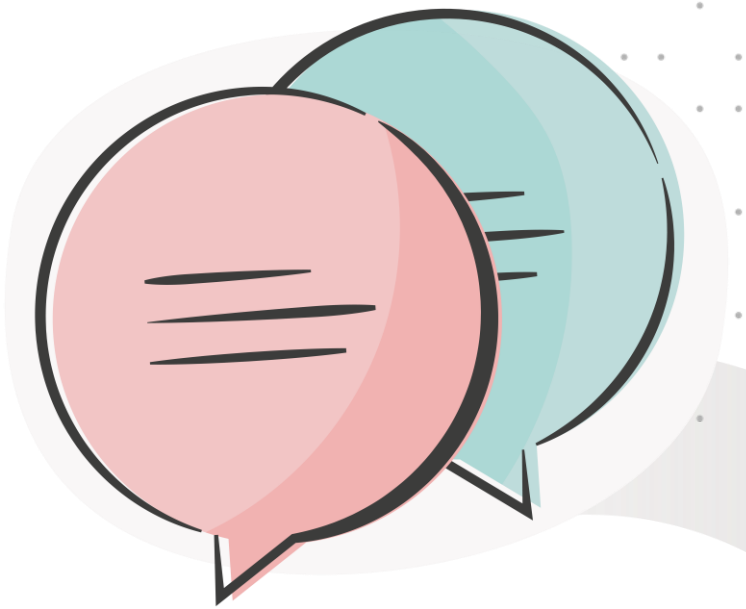
How to create diagrams…

Diagrams can be created using the following:

- LucidChart (online)
- Microsoft Visio (part of MS Office)
- Dia (free software)
- Draw.io (cloud based)

What do you use to create yours?



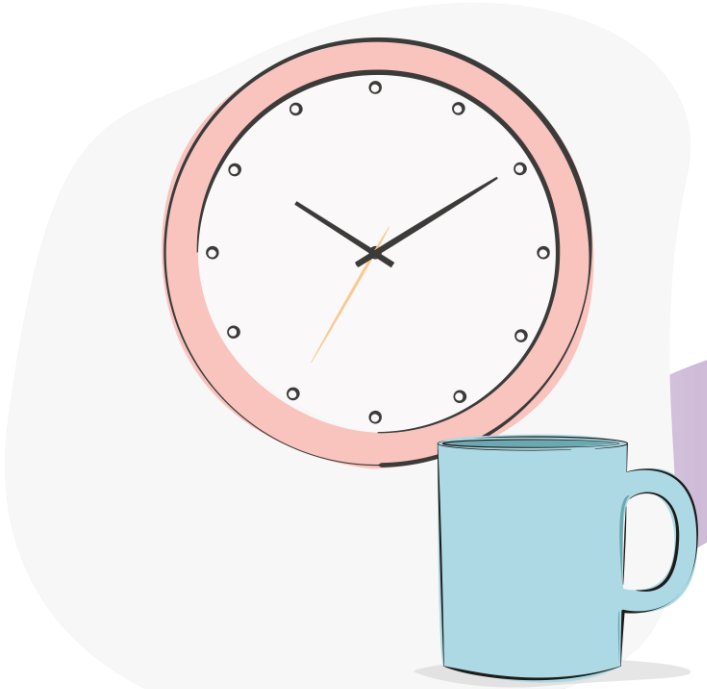


Group discussion

Lucidchart

BPP

Take a break!



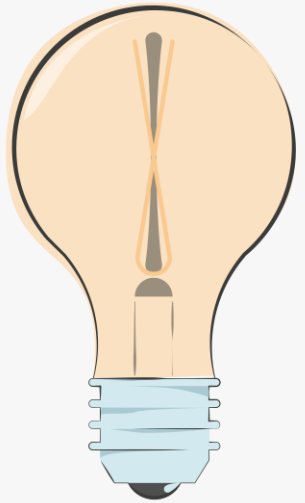
Design vs architecture diagrams and flowcharts

Tips for creating architecture diagrams

Remember the following…

Diagrams can be created using the following:

- Version Control your diagrams – they will evolve
- Simplify when possible, split when necessary
- Create logical groupings using polygons and colours
- Complement diagrams with descriptions
- Avoid too many acronyms
- You will normally have to explain the symbols you use

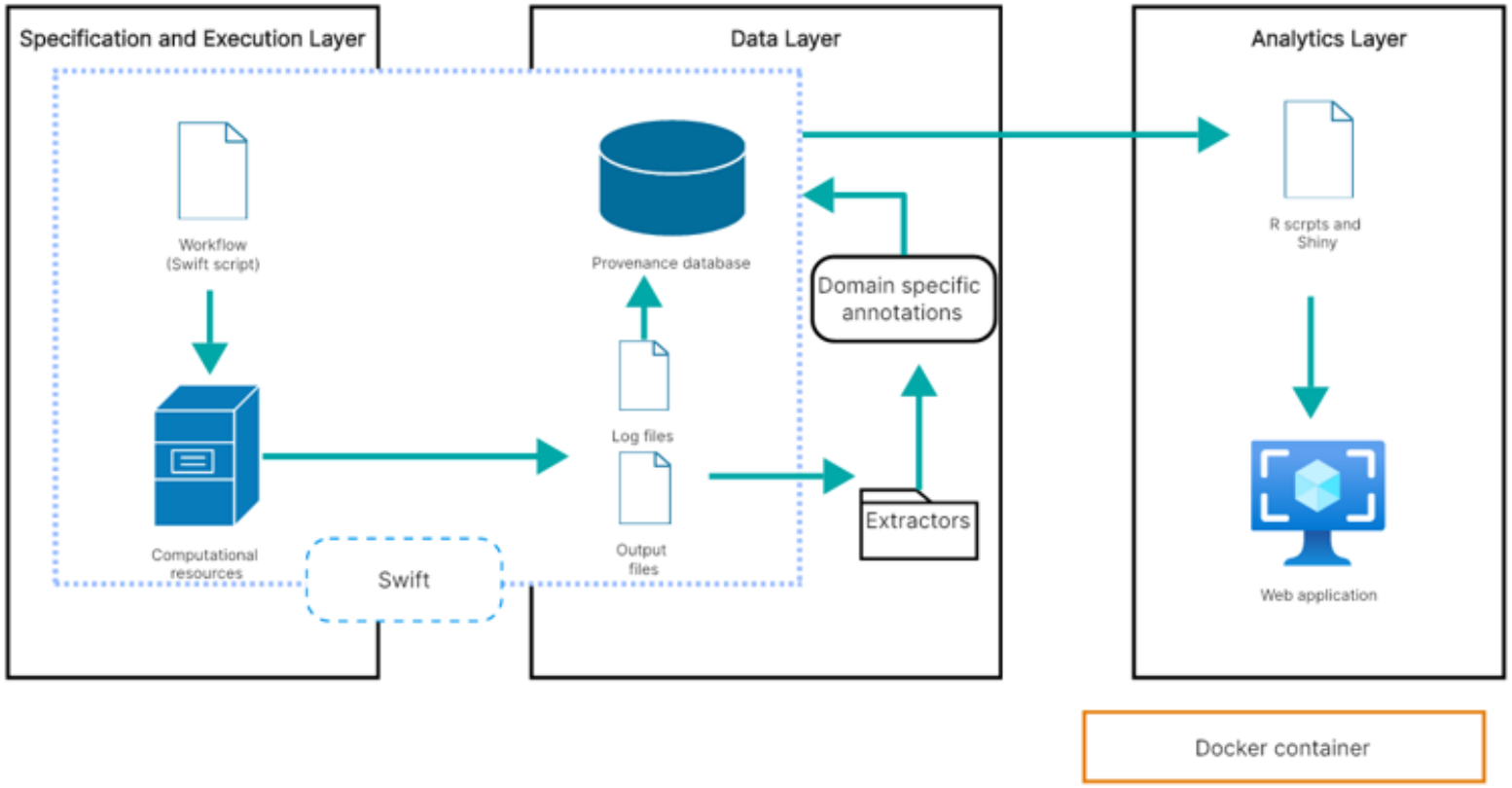


Mike’s handy hints!

BPP

Applying lenes, layers and chunks

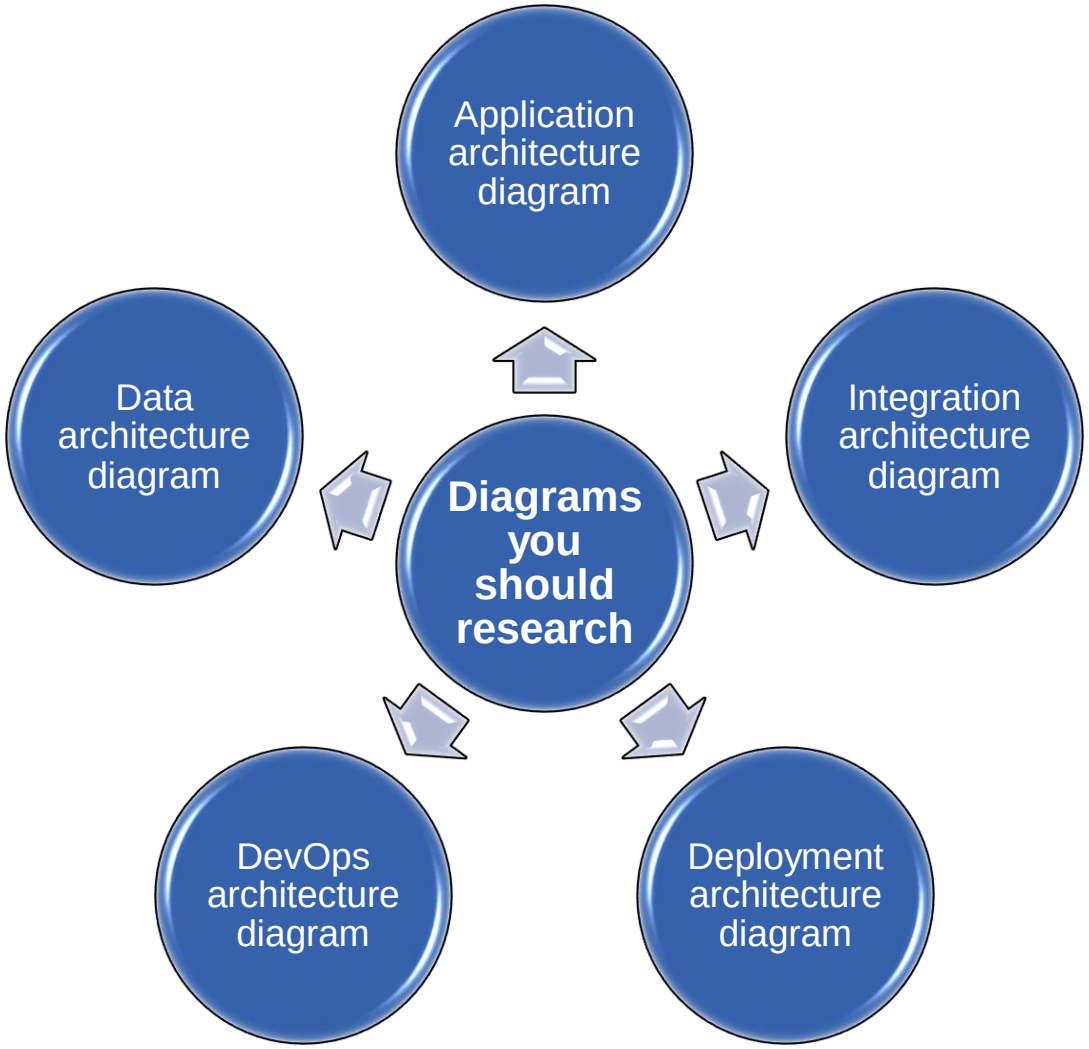
Exploring system architecture: Choosing lenses and layers



An example of conceptual architecture diagram

Other types of diagrams

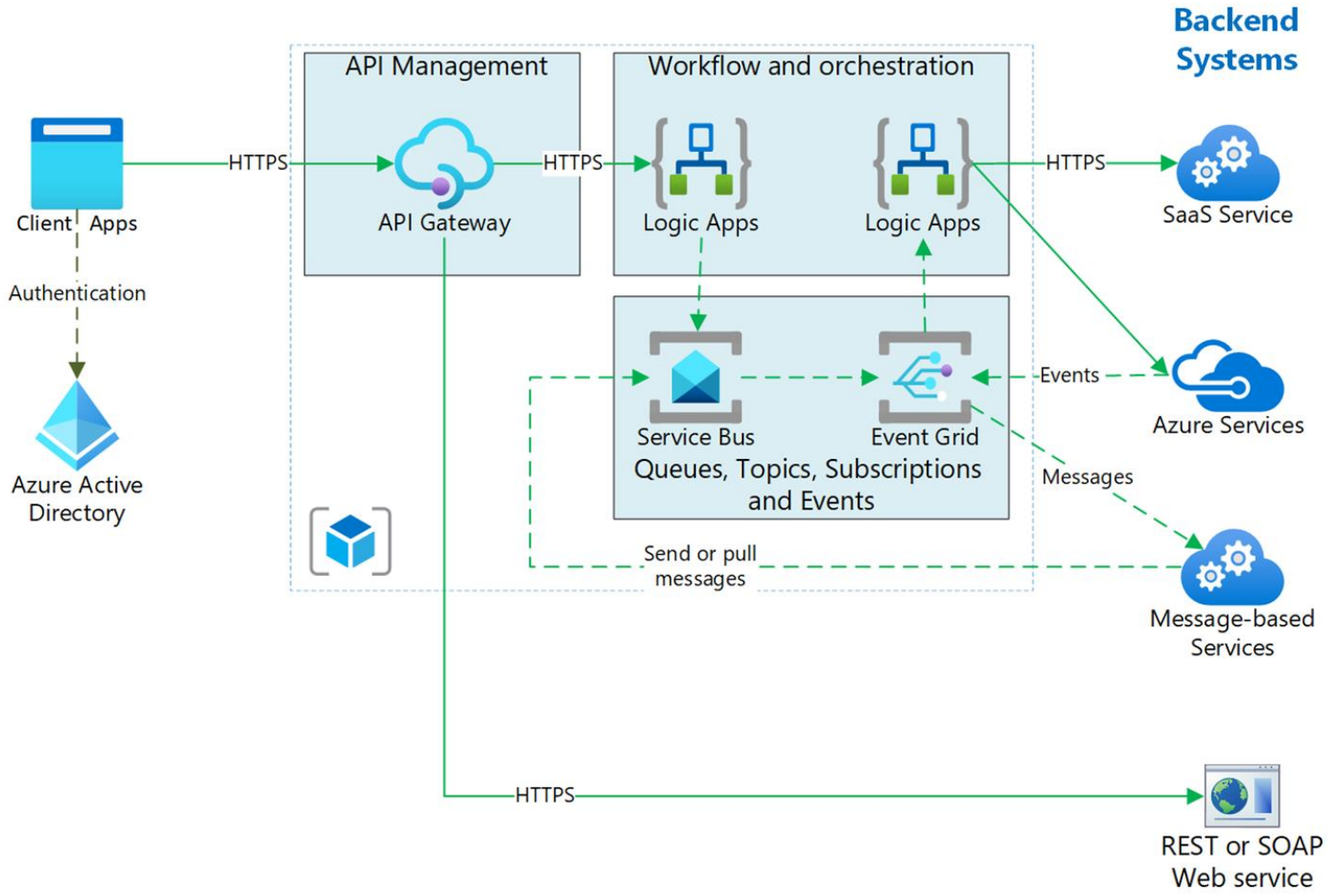
You should research the following…



Research suggestions!

Integration architecture diagram

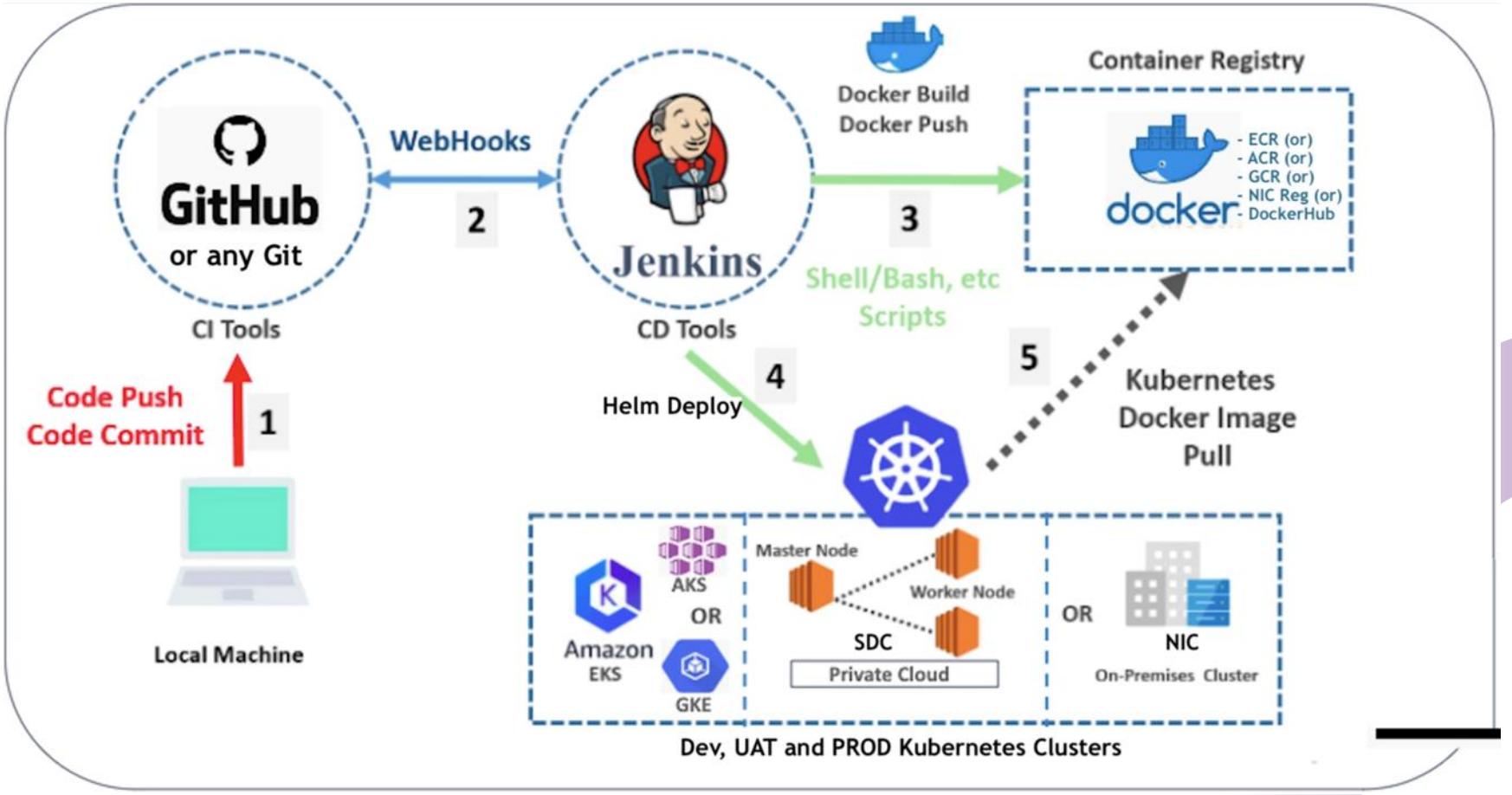
Exploring system architecture: Choosing lenses and layers



An example of conceptual architecture diagram

Deployment architecture diagram

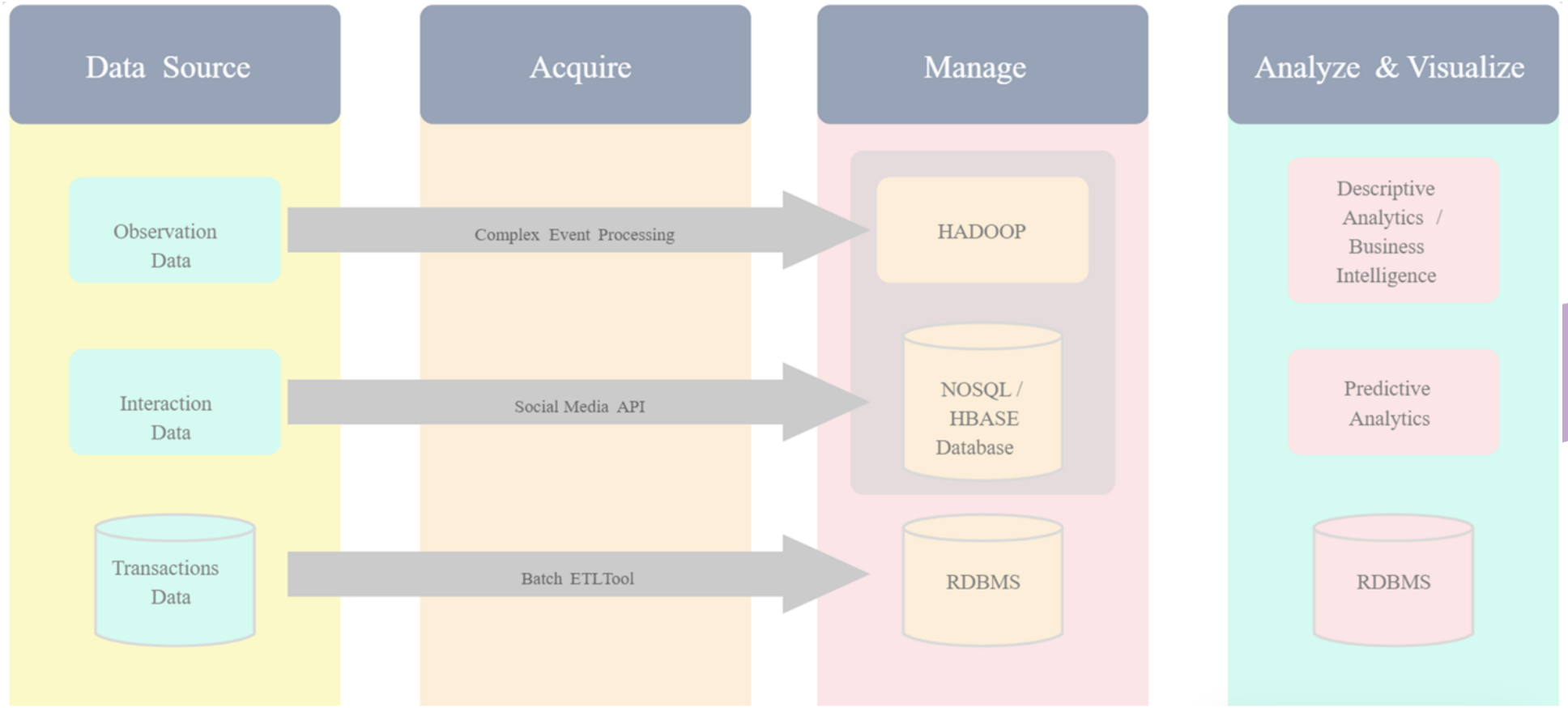
Exploring system architecture: Choosing lenses and layers



Visualising deployment architecture for scalability and security

Data architecture diagram

Understanding data architecture diagrams



Understanding the flow from acquisition to analysis in data architecture diagrams

Knowledge Check Poll

What is the primary purpose of UML component diagrams?

A. To show the physical deployment of the system

B. To represent the system's functional requirements

C. To illustrate the relationship between different components in a system

D. To describe the system's user interface

Feedback: C – The purpose of a UML component diagram is to show the relationship between different components in a system

Submit your responses to the chat!

BPP

Collaborate task

Scenario:

NextGen Fitness provides personalised fitness tracking and coaching services via a mobile app and wearable devices.

The system collects wearable data, stores it, and uses ML for coaching advice. Users can also connect with personal trainers.

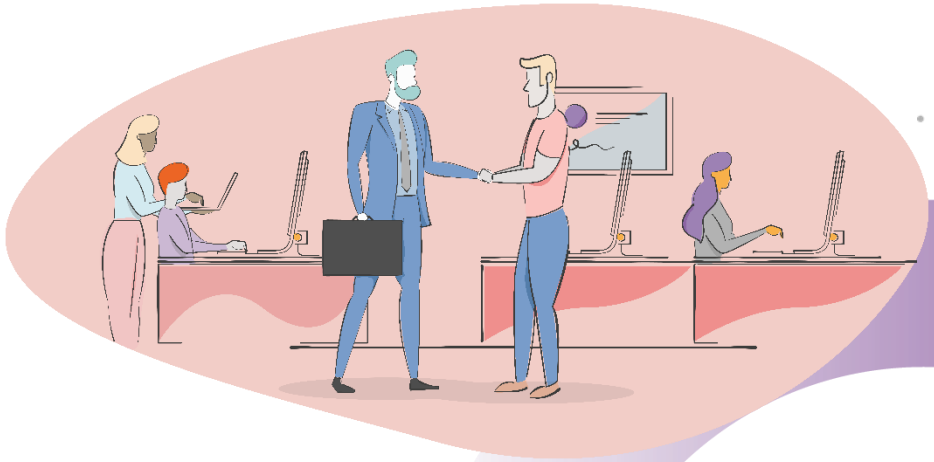
The brief:

1. Create a basic flowchart/architecture diagram for NextGen Fitness
2. Include actors: Users, Personal Trainers, Wearable Devices, Mobile App, ML Algorithms
3. Include containers: UI, Data Storage, Data Processing, Personalized Coaching, Trainer Communication
4. Show data flow and interactions
5. Use diagramming tools (LucidChart, Visio, etc.).

Collaborate task

BPP

Public cloud computing and architecture problem space

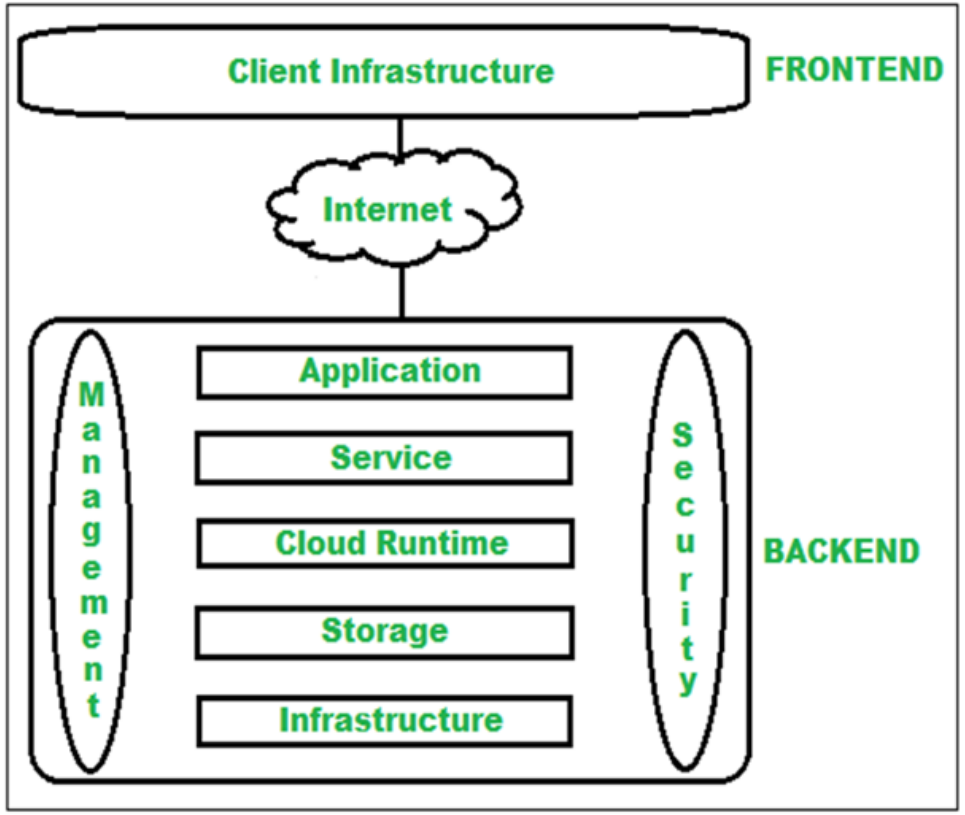


Architecture of public cloud computing

Working with public cloud computing for reliable data architectures

The following considerations are important:

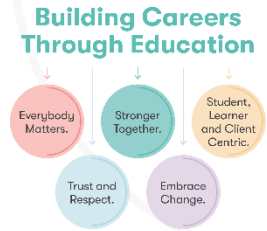
- Cloud storage
- Data processing
- Data integration
- Data security and compliance
- Scalability and performance
- Cost optimisation
- Automation and infrastructure as code
- Monitoring and logging



Unlocking the power of public cloud computing for reliable data architectures

BPP

Architecture problem space

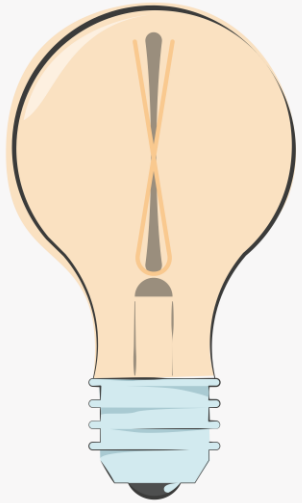


Helpful guidance…

BEFORE making decisions such as what technology is going to be used and the physical designs, you need to articulate the problem space for your architecture.

This is achieved by teams asking the following questions of the users:

- What is the current user experience or product?
- What are the problems?
- How can it be improved?
- What assumptions have you made?
- New product or user experience? How do you think your proposed design ideas support, change or extend current ways of doing things?



Mike’s handy hints!

BPP

Enterprise architecture

A framework

Problem
How to fully describe a project?

Difficulty
Work with an SME

1. An **Enterprise Architecture** (EA) shows project details, from business understanding to enterprise deployment.
2. An EA's artifacts includes models, documents, and specifications.
3. Example EAs include DoDAF (52 artifacts in 8 categories) the Zachman Framework (36 artifacts), and TOGAF.
4. Usually, only a subset of an EA's artifacts are created.

- Project concept
- Subject matter experts

→ **Enterprise Architecture Process** →

Project details at all levels for all customers

1. Select an Enterprise Architecture
2. Decide on which elements in the EA to create
   - A minimal list of artifacts could include
     - AV-1 : Overview and Summary Information
     - AV-2 : Integrated Dictionary
     - OV-1 : High Level Operational Concept Graphic – **most common**
     - OV-2 : Operational Node Connectivity Description
     - StdV-1 Standards Profile
     - SV-1 : System Interface Description
3. Create the artifacts and review with stakeholders

The **Zachman Framework** has (example instantiations shown below)
- **6 descriptive areas**: data, function, network, people, time, motivation
- **6 perspectives**: planner, owner, designer, builder, subcontractor, enterprise
- The 36 elements are arranged in a 6-by-6 grid

TOGAF (The Open Group Architecture Framework) uses 4 architecture domains: Applications, Business, Data, and Technical

BPP

Enterprise Architecture – Example – Phone App

- Consider creating a phone application
- Use the Zachman framework to show all needed artifacts.
- The 6 perspectives (rows) can be interpreted in several different ways; three are shown.
  - For example: “Objective /Scope” / “Contextual layer” / “Role: Planner”
- The cells in the 6-by-6 grid below contain only some of the items that would be in that cell.

6 perspectives – must be in this top down order

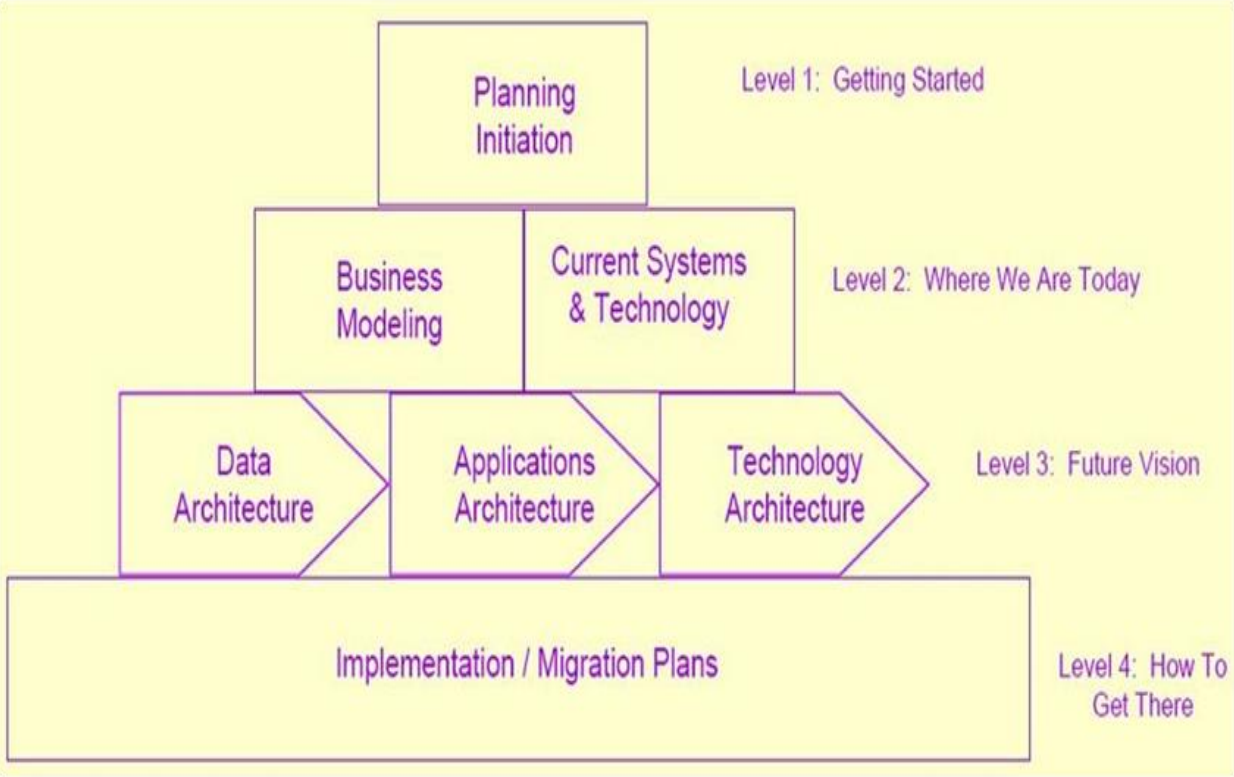
6 descriptive areas – can be in any order

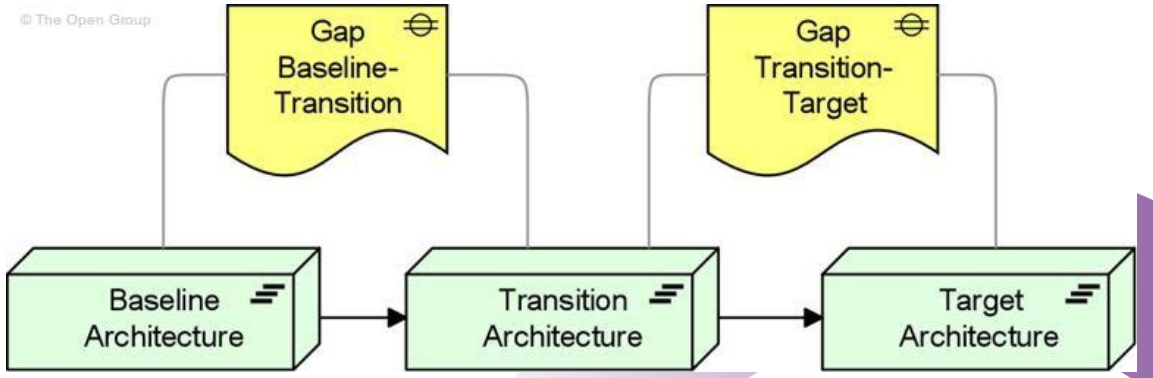
		What	How	Where	Who	When	Why
		Data	*Function*	*Network*	*People*	*Time*	*Motivation*
(1)	**Objective/Scope** Contextual layer *Role: Planner*	Business vision & goals	Business processes	Business locations	Departments involved	Future products road map	User needs. app business case
(2)	**Enterprise model** Conceptual layer *Role: Owner*	Short term goals	App financing, hiring, training	Project locations	Stakeholders buy-in plan	Product release timeline	App alignment with other offerings
(3)	**System logic** Logical layer *Role: Designer, Architect, or General Manager*	App look and feel	System architecture (e.g., support capabilities)	System connectivity	User interface design	Master schedule	App functionality
(4)	**Technology model** Physical model *Role: Builder, General Contractor, or Local Manager*	Platform description, wireframe model	App requirements	Technology architecture (e.g., component libraries)	Skill identification	Development milestones	Define function capabilities
(5)	**Detailed representation** Detailed model *Role: Scientist, Engineer, Subcontractor, or Programmer*	Interface definitions, database schema, code	App design	Communications architecture	Security design	Implementation model (e.g., scrum)	Motivate team to create sucessful product
(6)	**Functioning result** Enterprise release *Role: End user*	User data needs	Usage instructions	User locations (e.g., sales roll-out plan)	Market segmentation	App responsiveness	Motivation for end-users to obtain and use app

BPP

Time-based partitions

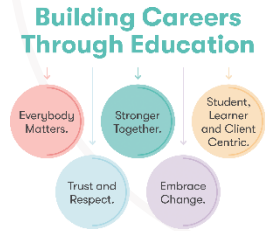
Building the foundation for scalable data architecture across architectural levels





Evolving from the current state of the system to the desired future state

Knowledge Check Poll



Which of the following is **NOT** a key consideration when working with public cloud computing for reliable data architectures?

A. Data security and compliance

B. Scalability and performance

C. Cost optimisation

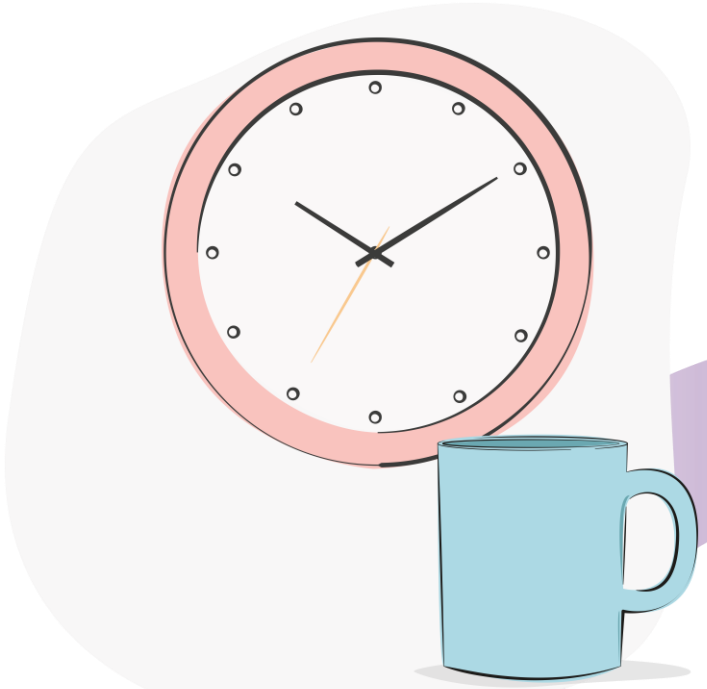
D. Physical server maintenance

Feedback: D – Physical server maintenance is typically handled by the cloud service provider.

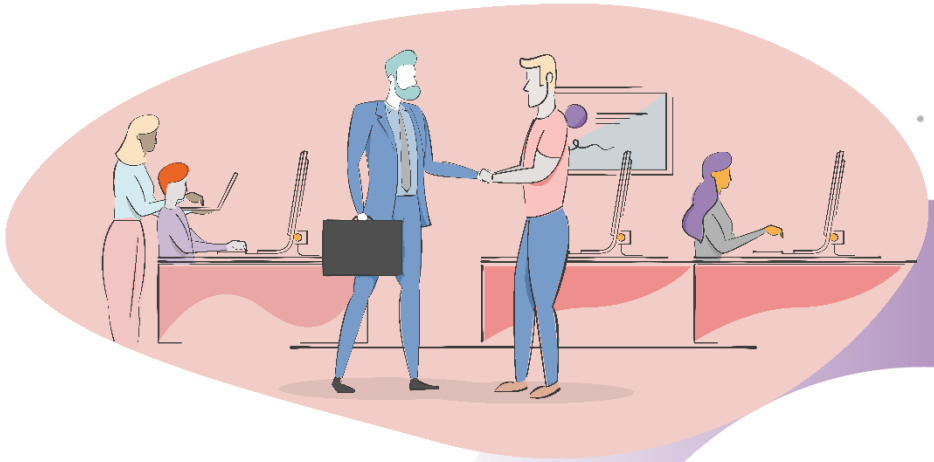
Submit your responses to the chat!

BPP

Take a break!



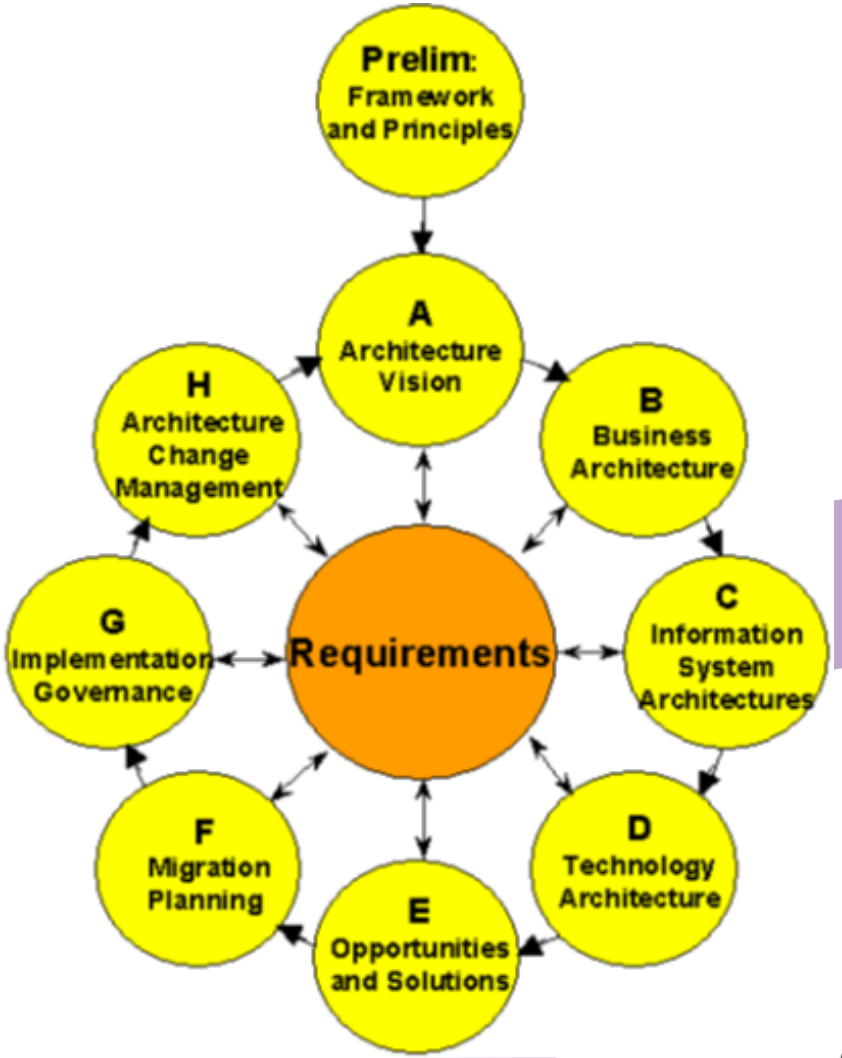
TOGAF, Data Mesh, and Data Culture



The Open Group Architecture Framework

Understanding TOGAF

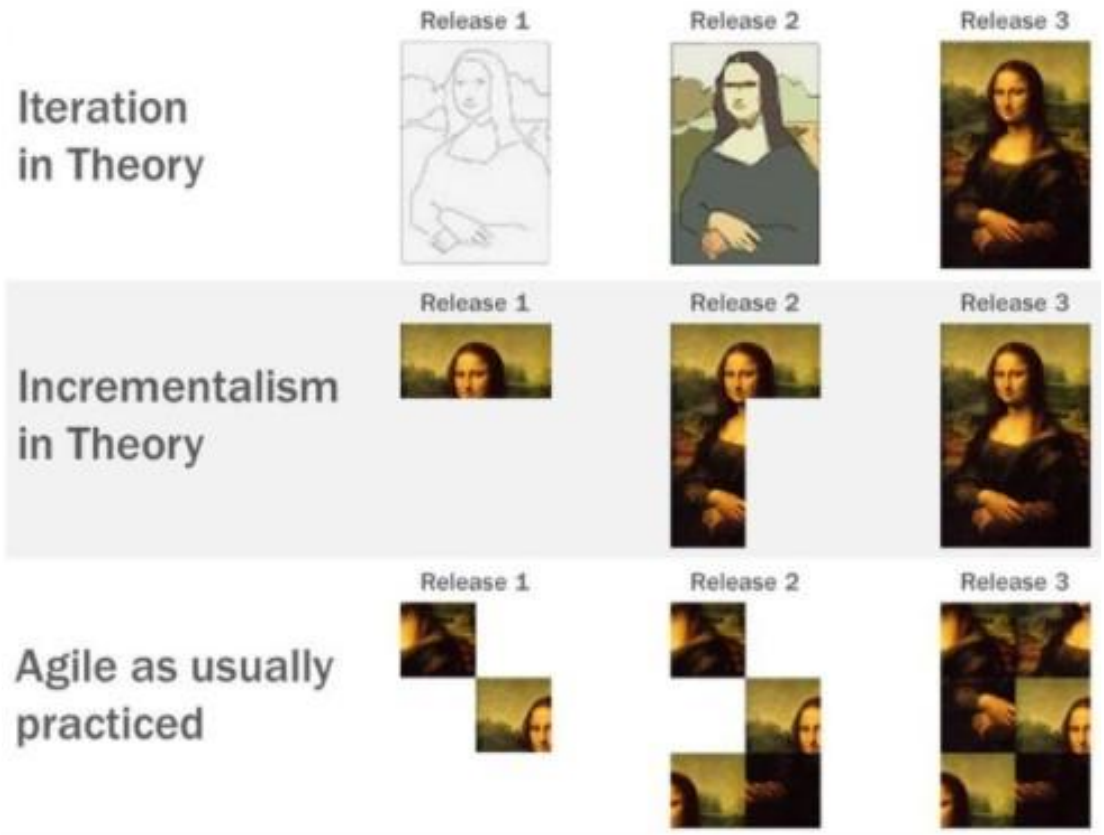
- TOGAF, developed by The Open Group, stems from the US Department of Defense's Technical Architecture Framework for Information Management (TAFIM)
- The Open Group received explicit permission from the DoD to build upon TAFIM and create TOGAF
- TOGAF has evolved through successive versions annually by The Open Group Architecture Forum
- Notable versions include TOGAF 7 (2001) and TOGAF 8 (2002)
- Each version is publicly available on The Open Group's website, emphasising transparency in architectural standards development



A visual representation of TOGAF

The Open Group Architecture Framework

Is it relevant in an Agile world?

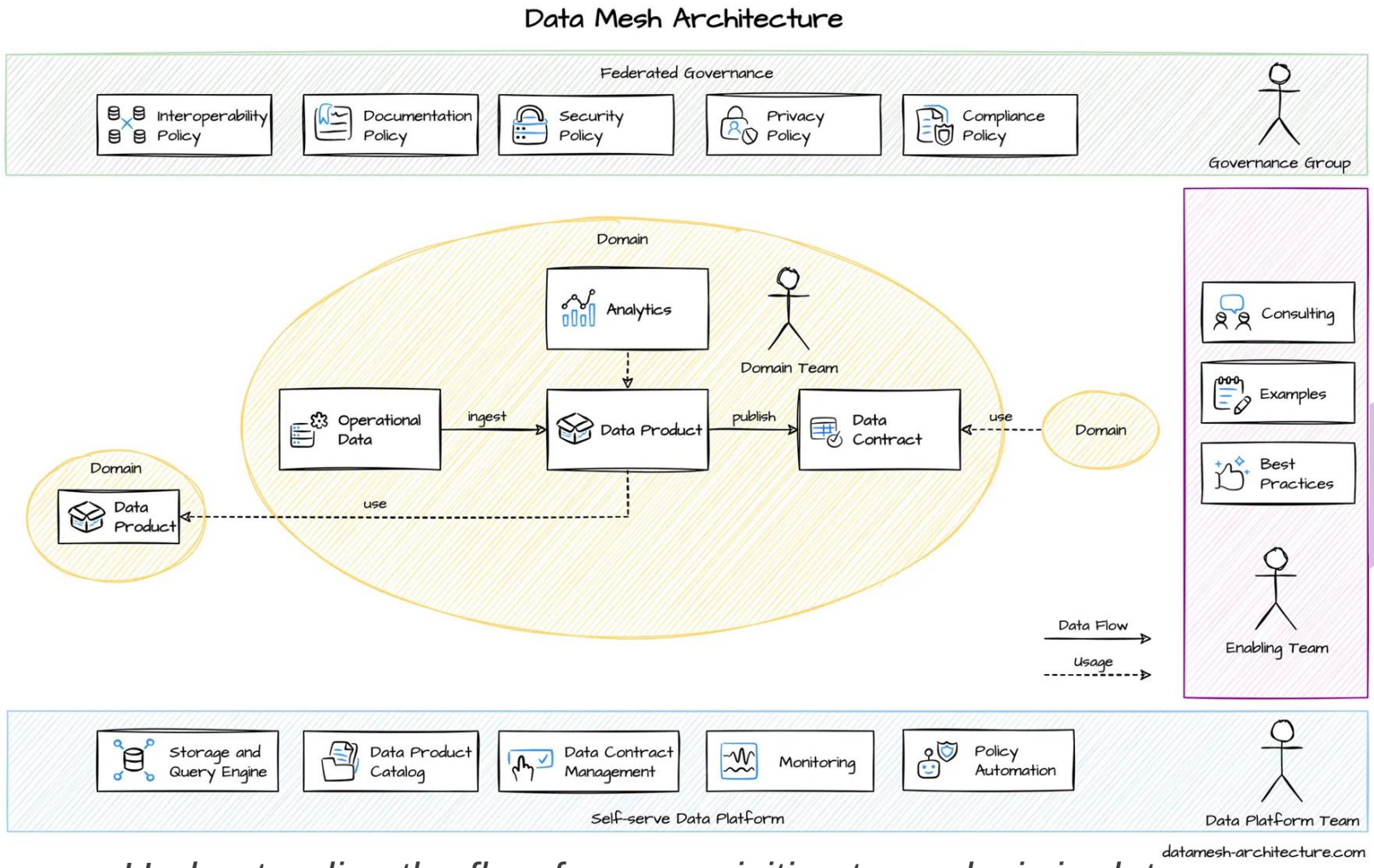


- ✓ TOGAF supports agility
- ✓ TOGAF is iterative
- ✓ Just another style of architecture

Understanding the flow from acquisition to analysis in data architecture diagrams

Data Mesh

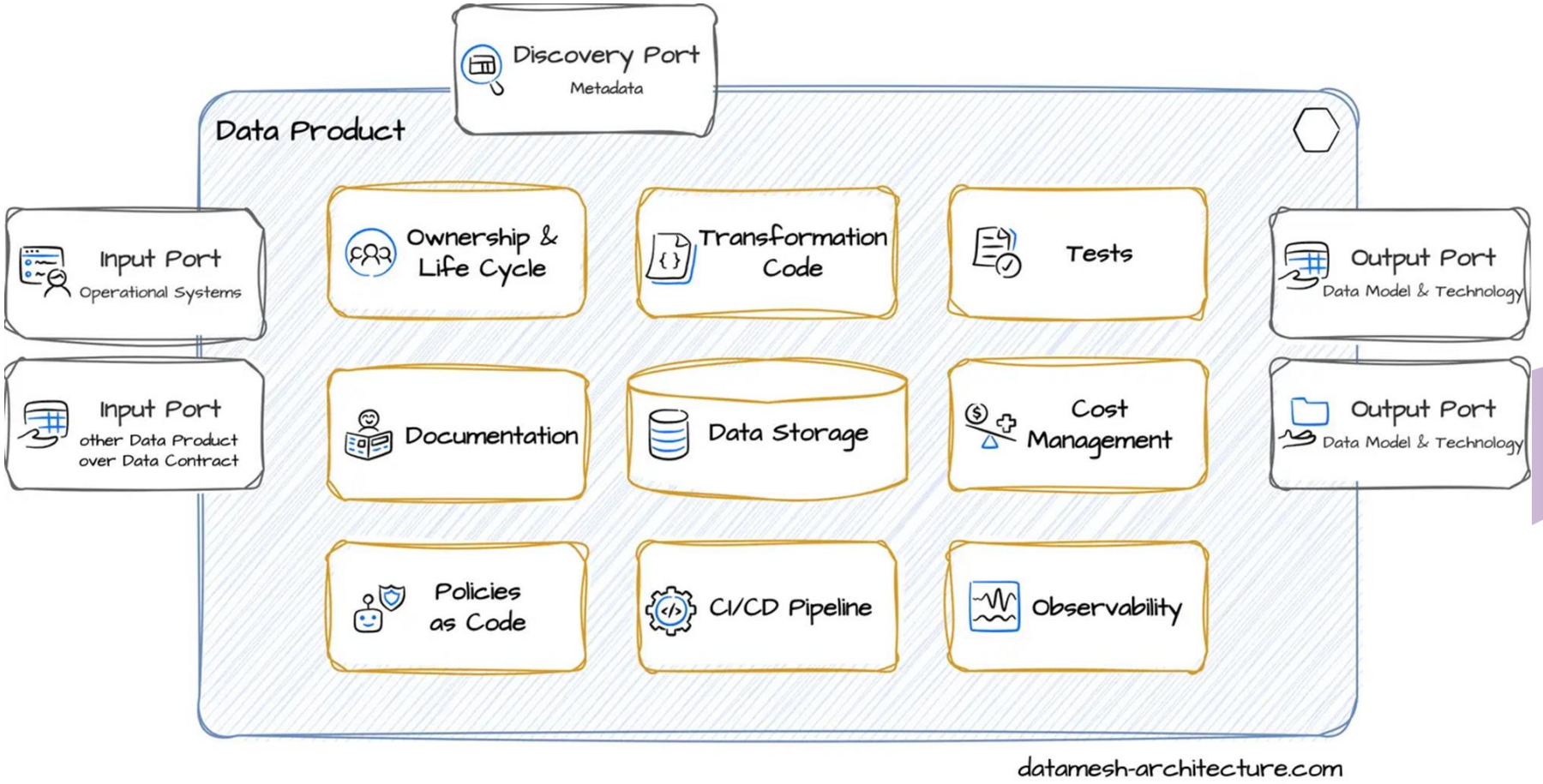
Exploring the architecture



Understanding the flow from acquisition to analysis in data architecture diagrams

Data mesh

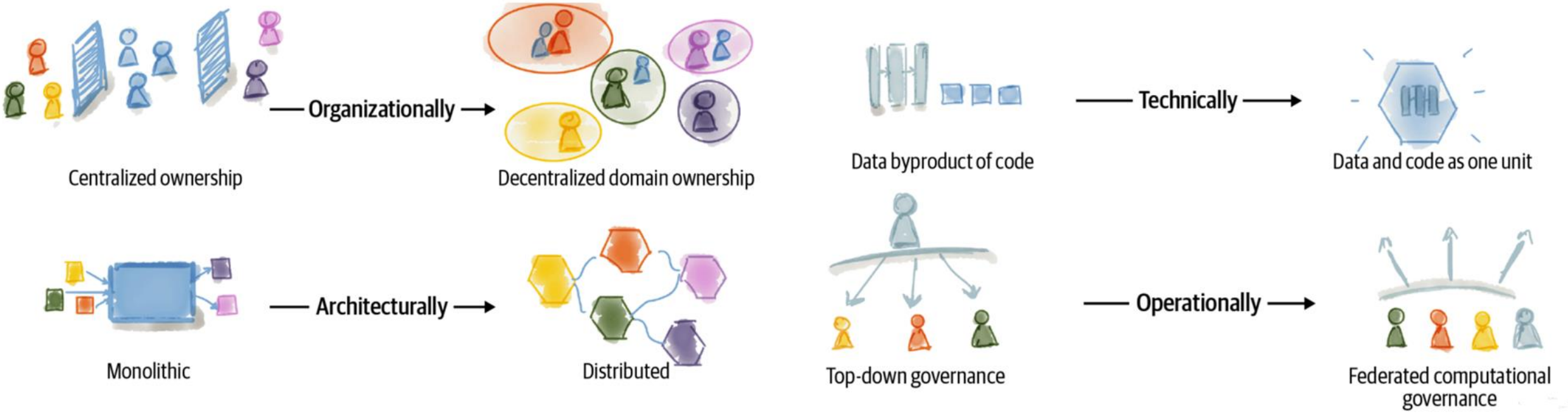
In the context of data products



Empowering data management: *Understanding the role of data products*

Data Mesh

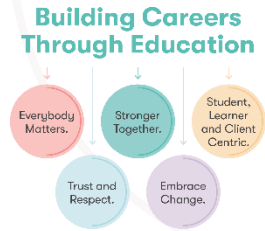
Embracing data decentralisation



Embracing data decentralisation with data mesh: Shifting ownership, architecture, technology, and operations

Practical lab

Apply hands-on exercise briefing

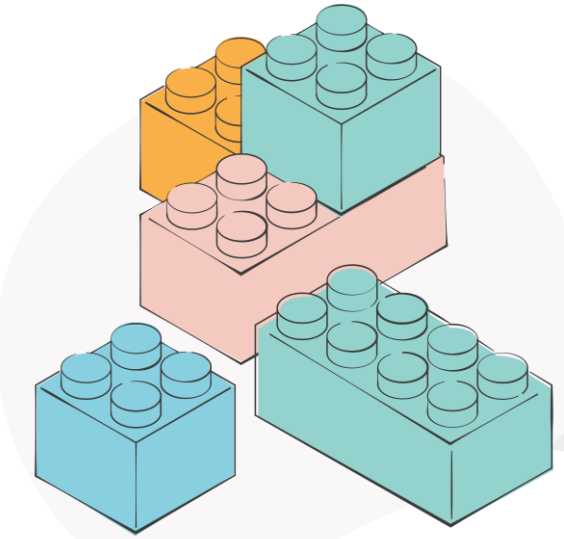


Your homework (off-the-job)

Your off-the-job apply activities for this topic are as follows:

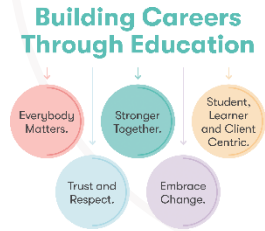
1. Creating database diagrams with LucidChart
2. Analysing monolithic vs micoservices architectures
3. Designing a data product
4. Applying TOGAF principles to data governance

Further details can be found here and in the Hub for this topic.



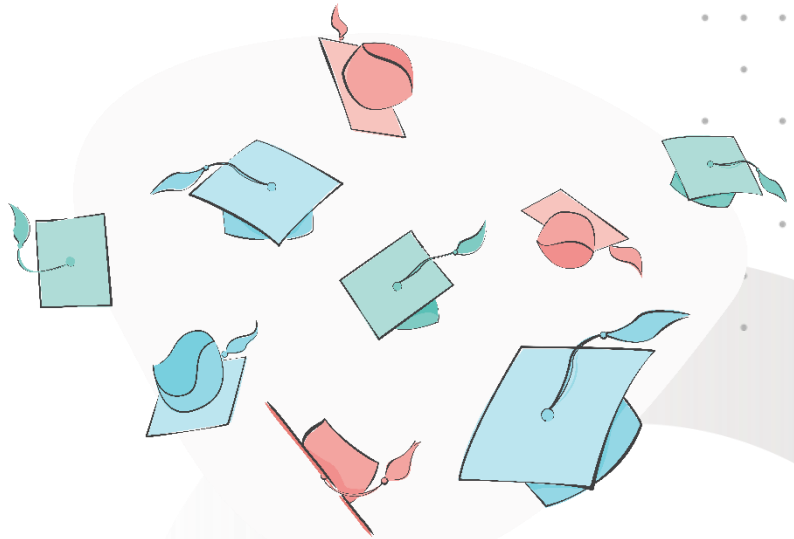
Apply exercises

Key Learning Summary



The key takeaways from this session are as follows:

- Architecture diagrams are crucial for managing complexity and change in data systems
- Diagrams help convey key system aspects, initiate discussions, and provide different perspectives
- Microservices architecture enables flexibility, scalability, and resilience
- Applying lenses, layers, and chunks enhances system comprehensibility and adaptability
- Public cloud computing offers powerful capabilities for reliable data architectures
- Data Mesh promotes decentralised data ownership and empowers domain-driven data management
- Fostering a data-driven culture is essential for successful data initiatives



BPP

Thank you

Do you have any questions, comments, or feedback?